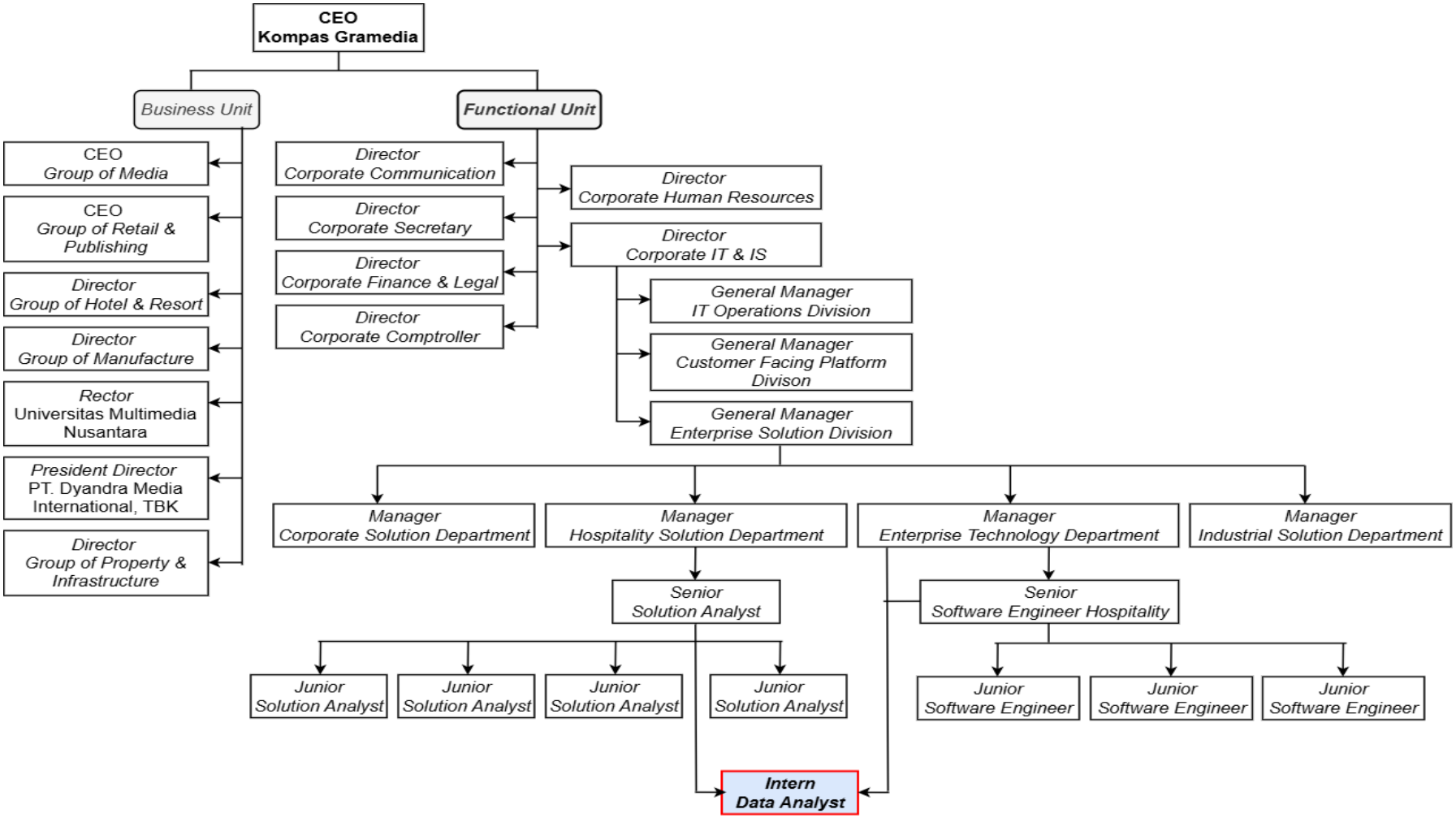
BAB III

PELAKSANAAN KERJA MAGANG

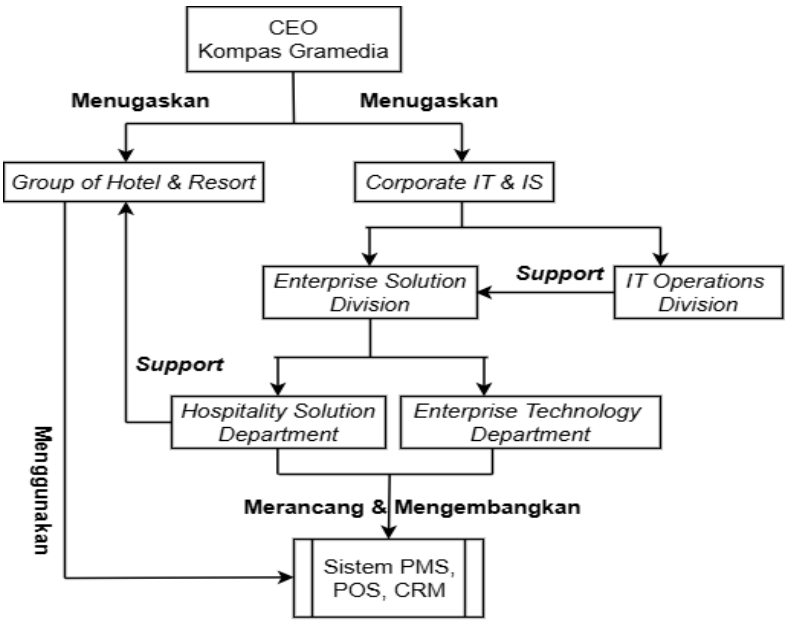
3.1. Kedudukan dan Koordinasi

Kedudukan dan koordinasi yang jelas dalam proyek magang di Kompas Gramedia sangat penting untuk memastikan peran setiap tim terdefinisi dengan baik sehingga dapat mendukung kolaborasi dan mencapai tujuan bersama. Posisi peserta magang adalah sebagai *data analyst* di *Corporate* IT & IS, *Enterprise Solution Division*, *Hospitality Solution Department* yang berperan penting dalam mendukung transformasi digital Kompas Gramedia. *Hospitality Solution Department* sendiri merupakan bagian dari *Enterprise Solution Division* yang bertugas untuk mengelola sistem informasi operasional hotel PT. Grahawita Santika, serta mengembangkan teknologi yang meningkatkan kinerja hotel dan mendukung pengambilan keputusan berbasis data. Peserta magang ditempatkan di tim yang fokus pada pengolahan dan analisis data melalui pembuatan *dashboard* interaktif, dengan bimbingan intensif dari Bapak Stefanus Linardi (*Senior Solution Analyst*) sebagai mentor, dan pengawasan oleh Bapak Anwari Rahman (Manajer *Enterprise Technology Department*) sebagai supervisor. Peran peserta magang sebagai *data analyst* sangat penting dalam mengelola *big data*, membangun *pipeline* data, dan membuat visualisasi data melalui *dashboard* yang kemudian di-*deploy* ke sistem. Dalam stuktur kedudukan, terdapat dua manajer yang mengawasi proyek, yaitu Manajer *Hospitality Solution Department* yang bertanggung jawab atas *business analyst*, dan Manajer *Enterprise Technology Department* yang bertanggung jawab atas *software engineer*. Sebagai satu-satunya *data analyst intern*, peserta magang berkolaborasi dengan *business analyst* untuk memahami kebutuhan bisnis dan *software engineer* untuk mengembangkan solusi teknis, memastikan data yang dikumpulkan dan dianalisis dapat disajikan dengan relevan dan berguna untuk pengambilan keputusan manajerial. Struktur kedudukan diilustrasikan dalam Gambar 3.1. Struktur yang terorganisir memastikan setiap tim memiliki fokus dan tanggung jawab yang jelas, sehingga memudahkan koordinasi dan kolaborasi dalam perencanaan serta pelaksanaan proyek.



Gambar 3.1 Bagan Detail Struktur Kedudukan Peserta Magang

Gambar 3.1 menunjukkan CEO Kompas Gramedia (KG) sebagai pimpinan tertinggi yang membawahi seluruh unit bisnis dan fungsional, dengan *Director* yang bertanggung jawab atas pengelolaan divisinya. Di bawah *Director Corporate* IT & IS, terdapat *General Manager Enterprise Solution Division* yang mengawasi manajer dan staf. Staf terdiri dari *solution analyst* dan *software engineer*. *Hospitality Solution Department* bertanggung jawab merancang solusi teknologi untuk mendukung operasional *Group of Hotel & Resorts*, khususnya dalam sistem pengelolaan data tamu dan layanan perhotelan. Peran *data analyst intern* berfokus pada proses *Extract*, *Transform*, *Load* (ETL) dan pengolahan data dari berbagai sumber untuk mempersiapkannya sebagai bahan visualisasi dalam *dashboard*. Gambar 3.2 menggambarkan alur kerja kolaborasi antar tim dalam proyek sistem *hospitality*. CEO KG menugaskan *Group of Hotel & Resorts* (*business unit*) dan *Corporate* IT & IS (*functional unit*). *Corporate* IT & IS, melalui *Enterprise Solution Division*, bertanggung jawab merancang dan mengembangkan sistem perhotelan seperti *Property Management System* (PMS), *Point of Sale* (POS), *Customer Relationship Management* (CRM), dan didukung oleh IT *Operations Division* untuk *monitoring* infrastruktur. *Hospitality Solution Department* berperan untuk menganalisis kebutuhan bisnis sistem bersama *Group of Hotel & Resorts*, sementara *Enterprise Technology Department* merancang aspek teknis sistem.

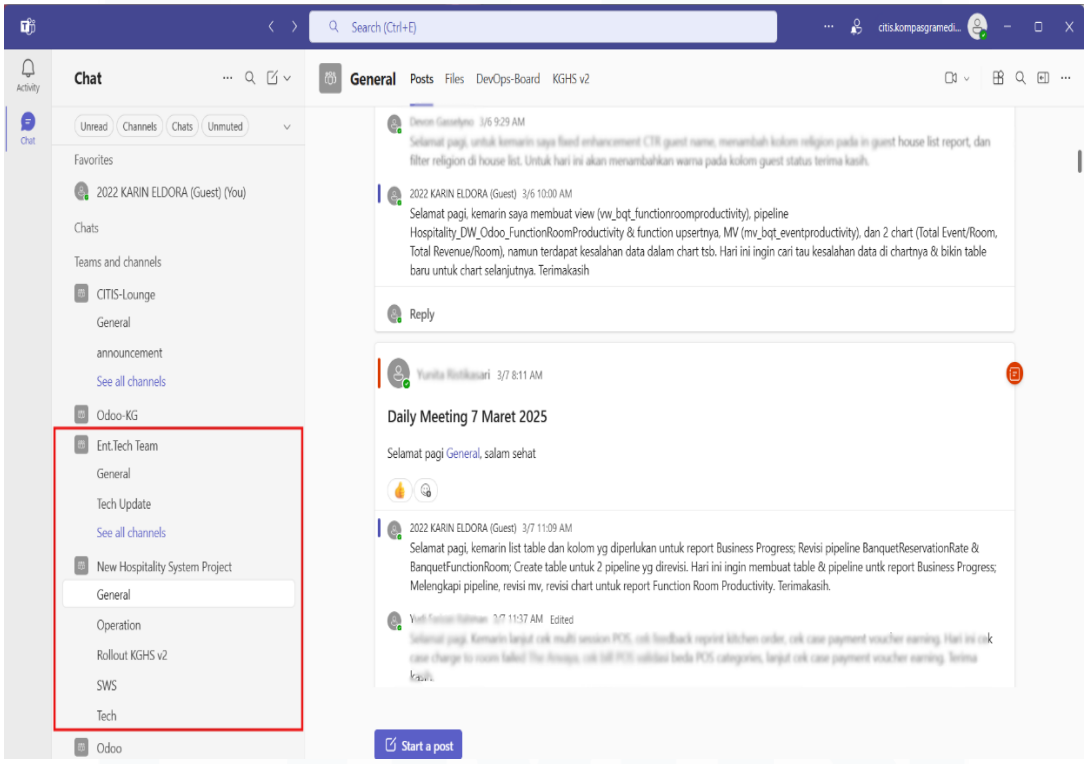


Gambar 3.2 Kolaborasi Peran dalam Proyek Sistem *Hospitality* Kompas Gramedia

Alur kerja proyek magang dimulai dari *Group of Hotel & Resorts* yang mengidentifikasi kebutuhan visualisasi data dan mengajukan *request* pembuatan *dashboard*. *Request* ini kemudian didiskusikan bersama manajer dan mentor magang, dimana mentor magang (*Senior Solution Analyst*) menerima keputusan pembuatan *dashboard* dan menganalisis permintaan serta sumber data yang ada. Setelah itu, mentor magang berdiskusi dengan peserta magang (*data analyst intern*) dan memberikan penugasan *dashboard*. Peserta magang selanjutnya menganalisis kebutuhan data, mempersiapkan *dataset*, serta merancang dan membuat *dashboard*. Hasil *dashboard* kemudian dikonsultasikan, dan mentor akan melakukan *crosscheck*. Jika penugasan *dashboard* tidak sesuai, peserta magang akan merevisi. Jika sesuai, *dashboard* di-*deploy* ke sistem PMS. Setelah *deployment*, dilakukan *user acceptance testing* dan *review code* oleh mentor. Jika fitur dinamis berfungsi, *dashboard* didiskusikan dan dipresentasikan kepada *Group of Hotel & Resorts*. Sebaliknya, jika terdapat masalah, peserta magang akan diminta untuk merevisi kode di sistem PMS. Setelah presentasi, peserta magang menerima evaluasi dan apresiasi atas hasil kerja yang telah dilakukan. Manajer hotel kemudian menilai apakah *dashboard* sudah sesuai untuk digunakan dalam pengambilan keputusan. Jika hasilnya memenuhi standar, proyek dianggap selesai. Proses ini diilustrasikan dalam Gambar 3.5 yang menggambarkan tahapan alur kerja secara lengkap. Koordinasi dan kolaborasi yang baik menekankan pentingnya komunikasi di setiap tahap proyek, yang didukung oleh penggunaan berbagai *platform* komunikasi.

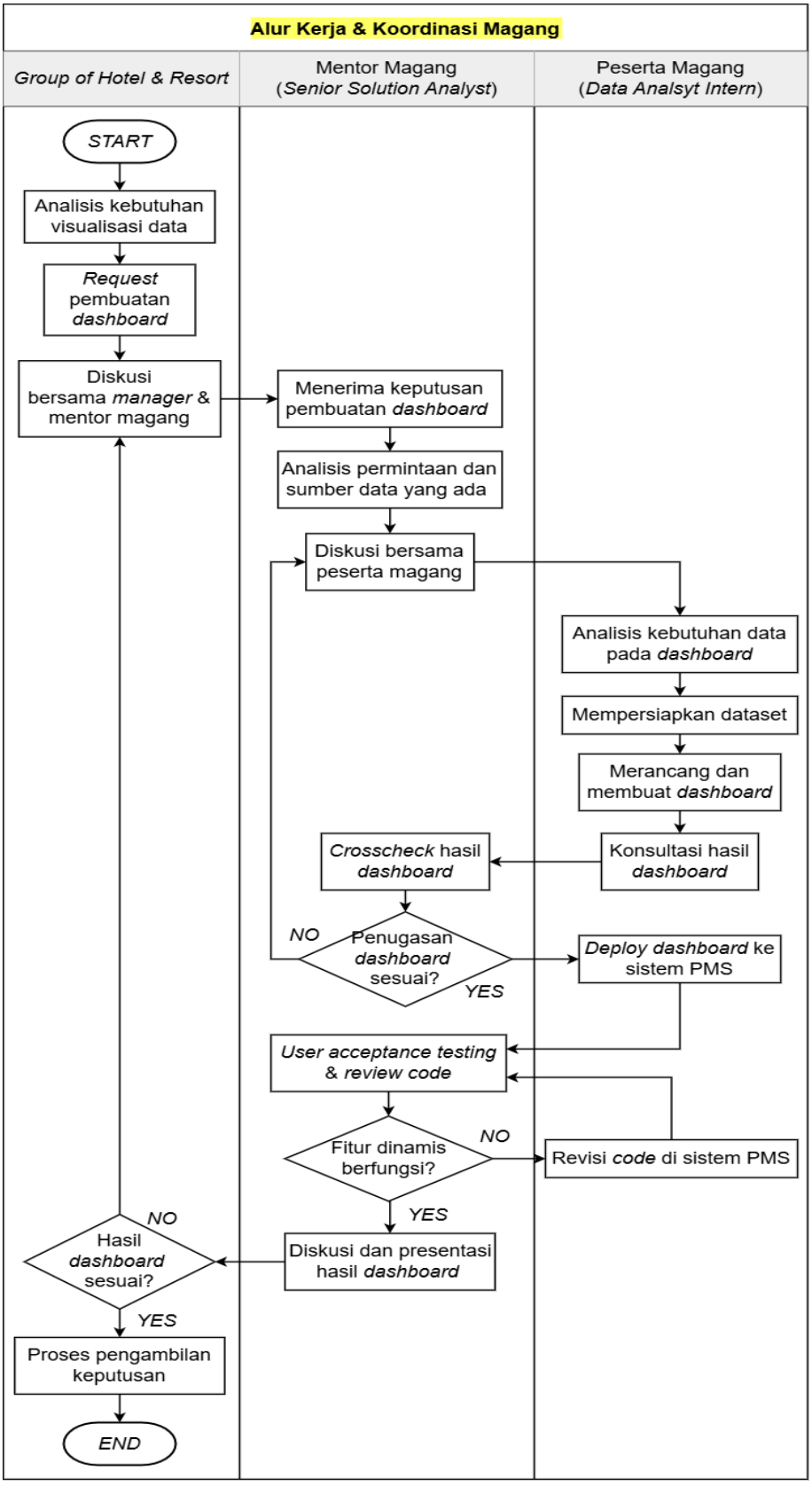
Koordinasi antara anggota tim di *Corporate* IT & IS dilakukan melalui berbagai *platform* komunikasi, seperti pada Gambar 3.4. Microsoft Teams menjadi *platform* utama yang digunakan untuk koordinasi antar divisi dan departemen. *Platform* ini membantu semua anggota tim untuk berinteraksi melalui beberapa saluran komunikasi, seperti grup diskusi, panggilan video, dan pesan pribadi. Gambar 3.3 menunjukkan tampilan Microsoft Teams yang digunakan dalam koordinasi, yang juga mencakup ruang kerja untuk berbagai grup yang relevan dengan proyek magang ini. Microsoft Teams mendukung diskusi antara peserta magang dengan mentor dan supervisor, serta koordinasi dengan rekan kerja lainnya melalui fitur *personal chat*. Peserta magang menggunakan dua *channel team chat*,

yakni Ent.*Tech Team* dan *New Hospitality System Project* yang digunakan untuk pembaruan tugas dan klarifikasi proyek secara *real-time*. Selain Microsoft Teams, WhatsApp (WA) digunakan untuk komunikasi administratif dan perizinan yang bersifat pribadi atau di luar konteks pekerjaan. Untuk keperluan komunikasi dengan unit bisnis lain, seperti *Corporate Human Resources*, atau untuk keperluan administrasi dokumen, Gmail digunakan sebagai saluran komunikasi resmi. Dengan berbagai *platform* ini, koordinasi antara peserta magang, mentor, supervisor, dan tim lainnya dapat berjalan lancar.



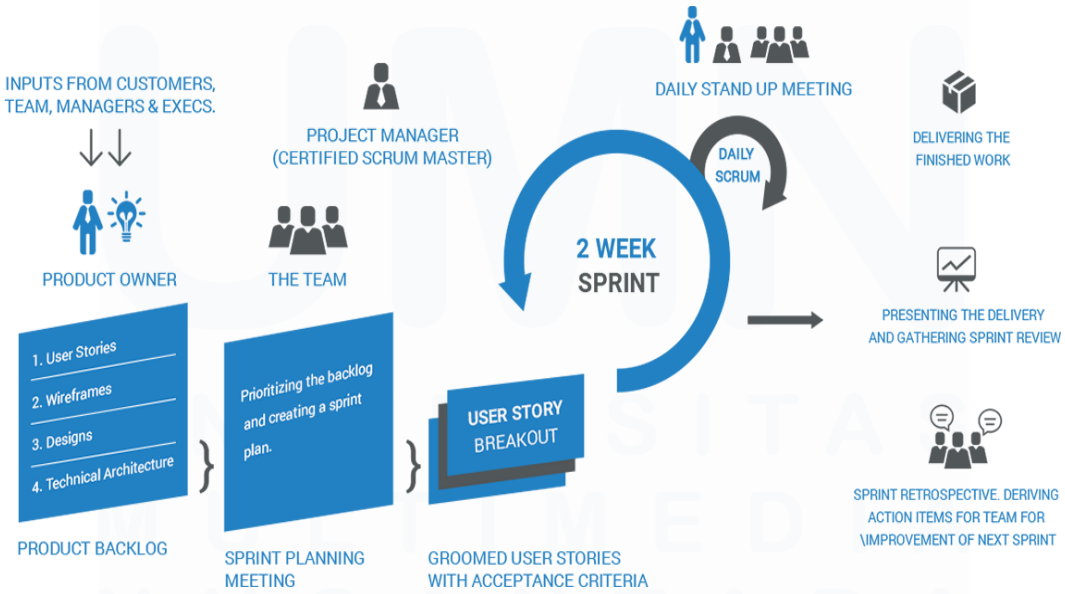
Gambar 3.3 Tampilan MS. Teams *Corporate* IT & IS

Gambar 3.4 *Platfrom* Komunikasi Utama, dari kiri: MS. Teams, WA, Gmail



Gambar 3.5 Alur Kerja dan Koordinasi Magang

Koordinasi dalam tim *Corporate* IT & IS dilakukan menggunakan SCRUM *Framework*, yang merupakan metode berbasis *agile* yang mendukung pengelolaan proyek secara terstruktur, iteratif, dan kolaboratif melalui pengembangan produk bertahap. Gambar 3.6 menggambarkan alur SCRUM *Framework* di *Hospitality Solution Department*, sementara Tabel 3.1 merinci jenis-jenis *meeting*, termasuk *daily stand-up*, *sprint planning*, *sprint retrospective*, dan *data discussion*. Pada *daily stand-up*, anggota tim melaporkan kemajuan tugas, merencanakan tugas hari itu, dan mengidentifikasi hambatan yang muncul. Pada *sprint planning*, tim menetapkan tujuan *sprint*, menentukan hasil yang ingin dicapai, dan memperbarui *backlog* tugas yang perlu ditangani. Selama *sprint*, diadakan *sprint review* untuk mengevaluasi dan mendemonstrasikan hasil *incremental product*, diikuti oleh *sprint retrospective* di akhir, dimana tim merefleksikan proses, mengevaluasi kekuatan dan kelemahan, serta merencanakan tindakan untuk meningkatkan kinerja di *sprint* berikutnya. Untuk *data discussion*, dilakukan secara insidental antara mentor dan peserta magang yang membahas kemajuan, tantangan, dan pengembangan *dashboard*. Setelah pengerjaan *dashboard* selesai, peserta magang mempresentasikan proses *end-to-end* kepada mentor dan menerima evaluasi serta apresiasi atas hasil yang telah dikerjakan.



Gambar 3.6 Detail Alur SCRUM *Framework*

Sumber: [38]

Tabel 3.1 Koordinasi *Meeting* dalam *Hospitality Solution Department*

Jenis *Meeting*	Jadwal	Anggota	Keterangan
Daily Stand-Up	Setiap Hari Pukul 08.30	Seluruh tim *Hospitality Solution Department*	• Memberikan *update* tentang pekerjaan yang telah diselesaikan sebelumnya. • Menyampaikan progress mengenai kegiatan yang sedang dikerjakan. • Menyusun rencana tugas yang akan dikerjakan pada hari tersebut. • Membahas isu atau permasalahan yang ada.
Sprint Planning	Setiap 2 Minggu Pukul 09.30	Tim *Hospitality Solution*, tim *Enterprise Technology*, tim IT *Operations*	• Menjelaskan detail *backlog* sesuai prioritas, dimulai dari *critical*, *high*, *medium*, *low*. • Presentasi *backlog* untuk *back office* dan *front office*. • Diskusi urgensi, *pain points*, *request*, dan *expected deliverables* untuk setiap *backlog*. • Menentukan tim yang akan mengerjakan *backlog* yang belum di-*assign*. • Memberikan *update* mengenai status *backlog* yang telah diambil, seperti *approved*, *committed*, *ready to test/deploy*, *test*, *resolved/ready to prod*, dan *done*.
Sprint Retrospective	Setiap 2 Minggu Pukul 11.00	Tim *Hospitality Solution*, tim *Enterprise Technology*	• Merefleksikan dan mencatat apa yang berjalan dengan baik dan yang tidak. • Mengevaluasi hal-hal yang perlu dipertahankan dan yang perlu diperbaiki. • Merencanakan tindakan untuk meningkatkan kinerja di *sprint* berikutnya. • Mengerjakan *team assessment* dan mendiskusikan hasilnya.
Data Discussion	Insidental	Peserta magang dan mentor	• Melakukan diskusi dan analisis permintaan kebutuhan *dashboard* dari *user*. • Memberikan *update* tentang pengembangan *dashboard* yang sedang dikerjakan dan rencana pengembangan berikutnya. • Menjelaskan proses ETL dan pembentukan *end-to-end dataset*. • Melakukan presentasi dan konsultasi hasil *dashboard*, serta meminta *feedback*. • *Crosscheck* dan validasi nilai data pada *dashboard*.

3.2. Tugas dan Uraian Kerja Magang

Program magang sebagai *data analyst intern* di Kompas Gramedia, tepatnya di *Enterprise Technology Department*, *Hospitality Solution Division*, *Corporate* IT & IS, dilaksanakan selama 120 hari, dari 03 Februari hingga 31 Juli 2025. Selama magang, peserta bertanggung jawab atas pengolahan data dari berbagai hotel yang dimiliki oleh Kompas Gramedia, yang berada di bawah naungan PT. Grahawita Santika. Tugas utama peserta magang meliputi proses *Extract, Transform, Load* (ETL) untuk menyalin dan mentransformasikan data dari berbagai sistem ke *data warehouse* terpusat, pengelolaan data dengan PostgreSQL dan DBeaver, serta pembuatan *dashboard* di Apache Superset. Salah satu proyek utamanya adalah melakukan ETL untuk menyalin data dari *database* odoo dan PMS ke *data warehouse* hospitalitydwh menggunakan *pipeline* Azure Data Factory. Selain itu, peserta magang membuat dan mengelola *materialized views* serta parameter metrik untuk *charts*. *Dashboard* berisi informasi penting dalam bentuk grafik dan tabel yang mudah dipahami untuk mempermudah pengambilan keputusan oleh *Board of Directors* hotel. Peserta magang juga mengimplementasikan teknis untuk *embedding dashboard* ke dalam sistem *Property Management System* (PMS). Berikut adalah tugas dan tanggung jawab peserta magang secara umum:

1. Mengidentifikasi tantangan atau masalah bisnis dan memberikan solusi berbasis data.
2. Melakukan proses ETL dengan membangun *pipeline* data yang *scalable* dan *reliable* menggunakan Azure Data Factory.
3. Mengelola *data warehouse* dengan PostgreSQL untuk analisis data dan pembuatan *materialized view* sebagai sumber *dataset*.
4. Membuat visualisasi data dalam bentuk *dashboard* yang jelas, informatif, dan mudah dipahami menggunakan Apache Superset.
5. Melakukan *deployment dashboard* ke dalam sistem PMS menggunakan *Application Programming Interface* (API) dan *Software Development Kit* (SDK) melalui Microsoft Visual Studio.
6. Bekerja sama dengan *business analyst* dan *software engineer* untuk menghasilkan analisis yang relevan.

Tabel 3.2 menunjukkan *timeline*, rincian tugas, dan progres kegiatan selama empat bulan magang sebagai *data analyst intern* di *Hospitality Solution Department* Kompas Gramedia. Kegiatan pertama adalah orientasi program dan pengenalan lingkungan kerja yang dilakukan pada awal magang. Pada tahap ini, dilakukan penjelasan mengenai profil perusahaan, visi, misi, nilai-nilai Kompas Gramedia, serta struktur organisasi yang berlaku di *Corporate* IT & IS. Selain itu, peserta magang juga diberi pemahaman tentang *jobdesk* yang akan dijalankan dan ekspektasi selama magang, serta diberikan akses untuk mengeksplorasi *tools* komunikasi Microsoft Teams dan Azure DevOps, serta cara koordinasi yang dilakukan. Kegiatan berikutnya adalah instalasi dan konfigurasi berbagai *tools* yang digunakan dalam pekerjaan sehari-hari, termasuk Azure Data Factory, DBeaver untuk PostgreSQL, Apache Superset, dan Microsoft Visual Studio yang dilanjutkan dengan eksplorasi terhadap sistem dan fitur-fitur yang ada di *Property Management System* (PMS) serta *Customer Relationship Management* (CRM).

Kegiatan utama yang menjadi fokus adalah proses *Extract, Transform, Load* (ETL), dimana peserta magang merancang dan membuat berbagai alur *pipeline* data untuk menyalin data dari *database* odoo dan PMS ke *data warehouse* hospitalitydwh. Beberapa *pipeline* yang dirancang termasuk data *banquet*, data kebutuhan *dashboard* CRM, dan perancangan *job pipeline* dan *refresh Materialized Views* (MV) untuk memastikan data yang terintegrasi dapat diakses secara *real-time*. Setelah proses ETL selesai, peserta magang berfokus pada pembuatan dan desain *dashboard* menggunakan Apache Superset, dimana beberapa *dashboard* seperti *Banquet Dashboard* dan CRM Menu *Dashboard* dirancang dengan berbagai menu, filter, *charts*, dan *metrics* yang sesuai dengan kebutuhan manajerial hotel. Pada tahap akhir, dilakukan proses *embedding dashboard* ke dalam sistem PMS menggunakan *Application Programming Interface* (API) dan *Software Development Kit* (SDK), memastikan bahwa semua *dashboard* yang telah dikembangkan dapat diakses secara langsung dalam sistem operasional hotel.

Tabel 3.2 Rincian Aktivitas Kerja Magang pada *Hospitality Solution Department*

No.	Nama Kegiatan	Tanggal Mulai	Tanggal Berakhir	Rangkuman Pekerjaan
On Boarding and Tools Introduction				
1.	Orientasi program dan lingkungan kerja.	03/02/2025	04/02/2025	• Pengenalan profil perusahaan, termasuk nilai, visi, dan misi Kompas Gramedia. • Memahami struktur organisasi dan peran setiap unit dalam *Corporate* IT & IS. • Penjelasan *jobdesk*, *tools*, tanggung jawab dan ekspektasi selama magang.
2.	Eksplorasi *tools* komunikasi dan alur koordinasi divisi	03/02/2025	06/02/2025	• Pemberian akses dan eksplorasi alur koordinasi menggunakan Microsoft Teams. • Pemberian akses dan eksplorasi alur koordinasi menggunakan Azure DevOps.
3.	Instalasi, konfigurasi, dan eksplorasi *tools*	05/02/2025	17/02/2025	• Instalasi dan konfigurasi *tools* Azure Data Factory serta eksplorasi fitur yang ada. • Instalasi dan konfigurasi *tools* DBeaver (PostgreSQL) serta eksplorasi fitur yang ada. • Instalasi dan konfigurasi *tools* Apache Superset serta eksplorasi fitur yang ada. • Instalasi dan konfigurasi *tools* Microsoft Visual Studio serta eksplorasi fitur yang ada.
4.	Eksplorasi sistem, fitur, dan data pada sistem hotel	18/02/2025	24/02/2025	• Eksplorasi sistem, fitur, dan *code* PMS sekaligus memahami data yang dihasilkan. • Eksplorasi sistem, fitur, dan *dashboard* CRM sekaligus memahami data yang diperlukan.
ETL *Process*				
5.	Alur *pipeline data banquet* untuk kebutuhan *banquet dashboard*	25/02/2025	07/03/2025	• Merancang dan membuat alur *pipeline* Hospitality_DW_Odoo_BanquetFunctionRoom. • Merancang dan membuat alur *pipeline* Hospitality_DW_Odoo_BanquetPackage. • Merancang dan membuat alur *pipeline* Hospitality_DW_Odoo_BanquetReservation. • Merancang dan membuat alur *pipeline* Hospitality_DW_Odoo_BanquetReservationBill. • Merancang dan membuat alur *pipeline* Hospitality_DW_Odoo_BanquetReservationEvent. • Merancang dan membuat alur *pipeline* Hospitality_DW_Odoo_BanquetReservationEventAdditionalResources.

No.	Nama Kegiatan	Tanggal Mulai	Tanggal Berakhir	Rangkuman Pekerjaan
				• Merancang dan membuat alur *pipeline* Hospitality_DW_Odoo_BanquetReservationRate. • Merancang dan membuat alur *pipeline* Hospitality_DW_Odoo_CatalogEventType. • Merancang dan membuat alur *pipeline* Hospitality_DW_Odoo_HrEmployee. • Merancang dan membuat alur *pipeline* Hospitality_DW_Odoo_ResPartner.
6.	Alur *pipeline* untuk kebutuhan *dashboard* CRM	10/03/2025	21/03/2025	• Merancang dan membuat alur *pipeline* Hospitality_DW_ManagerReport. • Merancang dan membuat alur *pipeline* Hospitality_DW_RoomCountSheet. • Merancang dan membuat alur *pipeline* Hospitality_DW_BudgetMarketSegment.
7.	Alur *job pipeline* dan *pipeline refresh* MV	24/03/2025	09/04/2025	• Merancang dan membuat alur *job pipeline* Hospitality_Job_DailyReport. • Merancang dan membuat alur *pipeline* Hospitality_DW_RefreshMVDaily.
Designing and Developing Visualization Dashboards				
8.	*Hospitality-Analytics Banquet Dashboard*	04/04/2025	23/04/2025	• Merancang dan membuat *Banquet Dashboard* submenu *Summary*, termasuk membuat MV, *metrics*, dan *charts* yang diperlukan. • Merancang dan membuat *Banquet Dashboard* submenu *Company*, termasuk membuat MV, *metrics*, dan *charts* yang diperlukan. • Merancang dan membuat *Banquet Dashboard* submenu *Sales & Room Productivity*, termasuk membuat MV, *metrics*, dan *charts* yang diperlukan. • Merancang dan membuat *Banquet Dashboard* submenu *Reservation*, termasuk membuat MV, *metrics*, dan *charts* yang diperlukan.
9.	*Hospitality-Analytics* CRM Menu *Dashboard*	10/04/2025	25/04/2025	• Merancang dan membuat *dashboard Market Segment by Brand* yang berkaitan dengan CRM Menu, termasuk membuat MV, *metrics*, dan *charts* yang diperlukan. • Merancang dan membuat *dashboard Market Segment with Budget* yang berkaitan dengan CRM Menu, termasuk membuat MV, *metrics*, dan *charts* yang diperlukan.

No.	Nama Kegiatan	Tanggal Mulai	Tanggal Berakhir	Rangkuman Pekerjaan
				• Merancang dan membuat *dashboard Average Room Rate Statistic* yang berkaitan dengan CRM Menu, termasuk membuat MV, *metrics*, dan *charts* yang diperlukan. • Merancang dan membuat *dashboard Occupancy Statistic* yang berkaitan dengan CRM Menu, termasuk membuat MV, *metrics*, dan *charts* yang diperlukan. • Merancang dan membuat *dashboard Hotel Revenue Statistic* yang berkaitan dengan CRM Menu, termasuk membuat MV, *metrics*, dan *charts* yang diperlukan. • Merancang dan membuat *dashboard Room Revenue Statistic* yang berkaitan dengan CRM Menu, termasuk membuat MV, *metrics*, dan *charts* yang diperlukan. • Merancang dan mesmbuat *dashboard Geographical Origin Of Business* yang berkaitan dengan CRM Menu, termasuk membuat MV, *metrics*, dan *charts* yang diperlukan.
***Embedding Dashboards into Property Management System* (PMS)**				
10.	*Deploy dashboard* ke sistem PMS menggunakan API dan SDK	28/04/2025	09/05/2025	• *Deploy dashboard Hotel Statistic* ke sistem PMS menggunakan API dan SDK. • *Deploy dashboard Reservation & OTA* ke sistem PMS menggunakan API dan SDK. • *Deploy dashboard Room Statistic* ke sistem PMS menggunakan API dan SDK. • *Deploy dashboard Revenue Statistic* ke sistem PMS menggunakan API dan SDK. • *Deploy dashboard Outlet Statistics* ke sistem PMS menggunakan API dan SDK. • *Deploy* seluruh *dashboard* CRM ke sistem PMS menggunakan API dan SDK.

3.2.1. *On Boarding* – Orientasi Program dan Lingkungan Kerja

Gambar 3.7 Kantor *Functional Unit* Kompas Gramedia

Pada tanggal 03 Februari 2025, program magang resmi dimulai dengan orientasi pengenalan terhadap lingkungan kerja dan struktur organisasi perusahaan. Kantor *functional unit* Kompas Gramedia (KG) dapat dilihat pada Gambar 3.7. Kegiatan orientasi ini bertujuan untuk memberikan pemahaman menyeluruh mengenai profil dan sejarah perusahaan, nilai-nilai, visi dan misi, serta pengenalan terhadap berbagai unit yang ada dalam *Corporate* IT & IS (CITIS), yang terdiri dari tiga divisi yakni *Enterprise Solution Division*, IT *Operations Division*, dan *Customer Facing Platform Division*. Peserta magang ditempatkan pada *Enterprise Solution Division*, *Hospitality Solution Department*. Seluruh peserta magang diperkenalkan dengan karyawan yang bekerja di divisi ini, yang terdiri dari berbagai *role*, seperti direktur, *general manager*, manajer, hingga staf dengan total 110 orang. Tim CITIS KG dapat dilihat pada Gambar 3.8. Kemudian, peserta magang diantarkan ke meja kerja dan dibantu untuk mengakses koneksi internet. Kegiatan ini memberikan pemahaman tentang peran masing-masing unit dalam mendukung operasional perusahaan, serta bagaimana koordinasi antar divisi dapat berjalan dengan lancar. Dokumentasi selama *on boarding* dapat dilihat pada Gambar 3.9.

Gambar 3.8 Tim *Corporate* IT & IS (CITIS) Kompas Gramedia

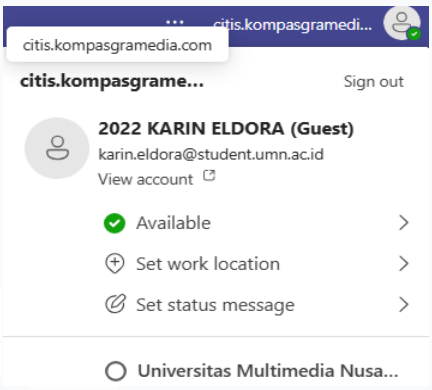
Peserta magang kemudian mengikuti *briefing* lebih lanjut yang dipimpin oleh Bapak Anwari Rahman (Manajer *Enterprise Solution*) selaku supervisor magang dan Bapak Stefanus Linardi (*Senior Solution Analyst*) selaku mentor lapangan, yang menjelaskan lebih detail mengenai tugas, tanggung jawab, serta ekspektasi yang harus dipenuhi selama program magang. Peserta magang juga diberi akses ke aplikasi komunikasi yang digunakan, seperti Microsoft Teams dan Azure DevOps, serta pemahaman penjelasan mengenai etika, aturan, dan budaya kerja yang berlaku di perusahaan. Selain itu, peserta magang turut diperkenalkan dengan lingkungan kantor secara langsung, termasuk berkeliling untuk mengenal berbagai ruang kerja dan fasilitas yang tersedia. Ibu Eris selaku sekretaris CITIS mengundang peserta magang ke dalam WhatsApp *Group* CITIS yang juga menjadi sarana komunikasi dan sumber informasi terkait kegiatan dan *update* yang ada. Kegiatan *on boarding* ini sangat penting karena dapat memfasilitasi peserta magang untuk beradaptasi dengan lingkungan kerja di KG, memahami struktur organisasi, mendapatkan akses yang diperlukan, serta memahami kultur dan tujuan perusahaan untuk melanjutkan ke tugas berikutnya.

Gambar 3.9 Dokumentasi Kegiatan *On Boarding* Kompas Gramedia

3.2.2. *Tools Introduction* – Eksplorasi *Tools* Komunikasi dan Alur Koordinasi Divisi

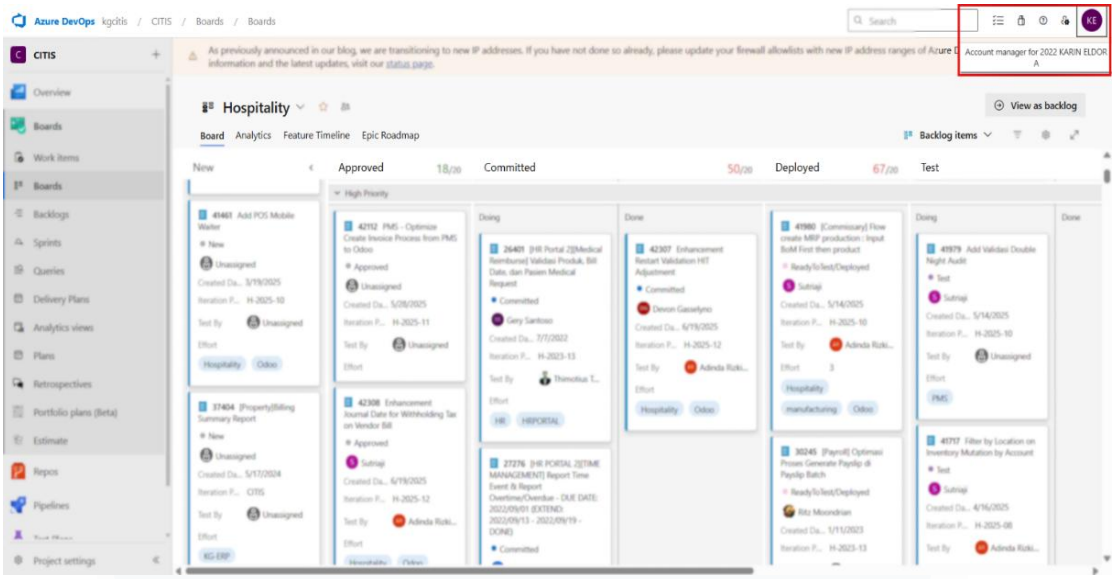
Pada program magang ini, peserta diberikan akses ke beberapa *tools* komunikasi yang penting untuk mendukung koordinasi dan kelancaran proyek. Salah satu *tools* utama yang digunakan adalah Microsoft Teams, yang berfungsi sebagai *platform* komunikasi utama untuk seluruh tim di *Corporate* IT & IS (CITIS). Gambar 3.10 menunjukkan akun Microsoft Teams peserta magang, yang mana merupakan akun Teams *student* UMN yang telah terhubung dengan Teams resmi milik citis.kompasgramedia.com. Microsoft Teams CITIS seperti pada Gambar 3.3 digunakanan untuk berinteraksi dengan berbagai cara, seperti melalui grup diskusi, panggilan video, dan pesan pribadi. Dalam aplikasi ini, peserta magang aktif berinteraksi dalam dua *channel*, yaitu Ent.*Tech Team* dan *New Hospitality System Project* yang digunakan untuk pembaruan tugas dan klarifikasi proyek. Melalui Teams, peserta magang dapat

berkomunikasi langsung dengan mentor, supervisor, serta rekan-rekan lainnya dalam tim untuk membahas berbagai proyek yang sedang berlangsung.



Gambar 3.10 Akun Microsoft Teams Peserta Magang

Peserta magang juga diberikan akses ke Azure DevOps sebagai *tools* komunikasi lainnya. Azure DevOps merupakan *platform* pengembangan *software* yang sangat berguna untuk manajemen proyek dan pengawasan *backlog*. Melalui Azure DevOps, peserta magang bisa memantau dan melacak perkembangan tugas-tugas yang ada, seperti melihat *backlog* proyek, *source code*, serta *repository* untuk penyimpanan dokumen dan kode aplikasi. Azure DevOps memiliki fitur utama "*boards*" yang berfungsi sebagai papan kanban atau scrum yang menampilkan *work items* dengan berbagai status untuk melacak kemajuan proyek. Setiap *work item* dilengkapi dengan identifikasi unik, deskripsi, informasi penugasan, dan estimasi waktu penyelesaian. Fitur ini memfasilitasi kolaborasi tim, meningkatkan transparansi, dan mendukung pemantauan proyek secara *real-time*. Selain *boards*, Azure DevOps juga menyediakan "*backlogs*" untuk mengelola pekerjaan prioritas, "*sprints*" untuk perencanaan iterasi, "*queries*" untuk mencari *work items*, "*delivery plans*" untuk rencana rilis proyek, dan "*pipelines*" untuk otomatisasi *build* dan *deployment*, yang penting untuk memastikan kelancaran *deploy dashboard* ke sistem PMS. Dengan fitur-fitur ini, peserta magang dapat lebih mudah beradaptasi dan berkontribusi dalam proyek. Gambar 3.11 menampilkan *user interface* dari Azure DevOps CITIS KG.



Gambar 3.11 Tampilan Azure DevOps CITIS KG

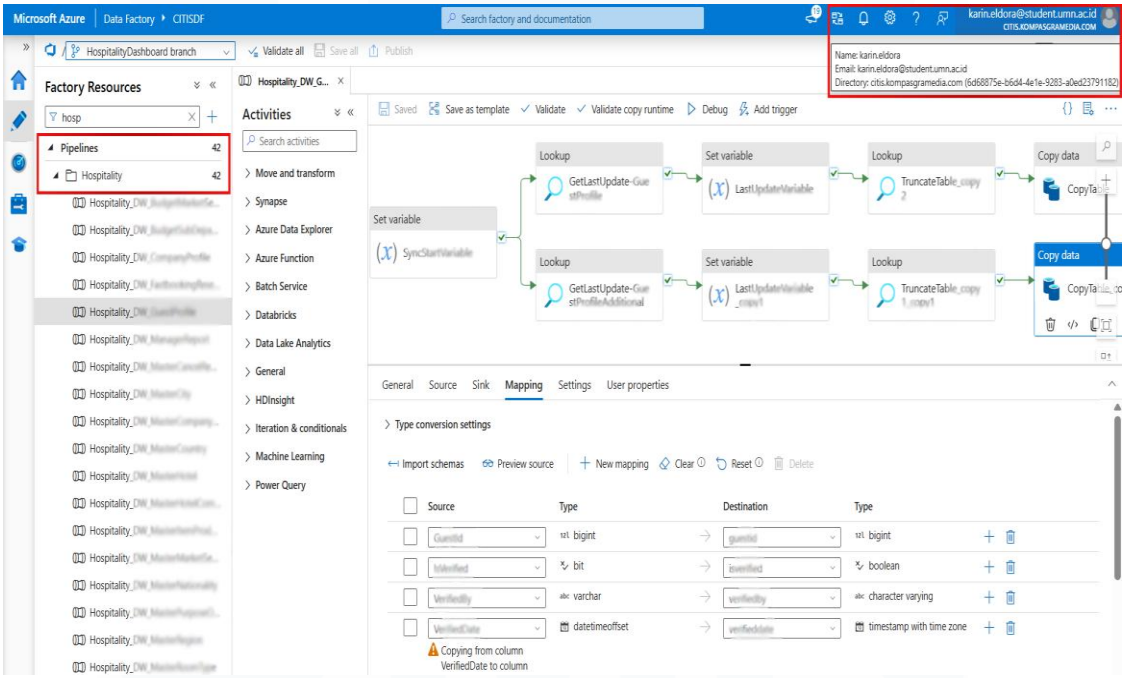
3.2.3. *Tools Introduction* – Instalasi, Konfigurasi, dan Eksplorasi *Tools*

1. *Tools* Azure Data Factory

Gambar 3.12 Logo Azure Data Factory

Azure Data Factory (ADF) dengan logo pada Gambar 3.12, merupakan *platform cloud* berbasis *Extract, Transform, Load* (ETL) yang disediakan oleh Microsoft Azure untuk membangun, menjadwalkan, dan mengelola *pipelines* data [15]. ADF sangat berguna untuk mengintegrasikan data dari berbagai sumber seperti *cloud*, *on-premises*, dan SaaS serta melakukan transformasi data sebelum memuatnya ke penyimpanan seperti *data warehouse* atau *data lakes*. Salah satu fitur utama ADF adalah membantu pengguna untuk merancang *pipeline* tanpa perlu menulis banyak kode, sekaligus menyediakan pemantauan dan peringatan untuk memastikan *pipeline* berjalan lancar. Selama magang, ADF digunakan untuk melakukan *Extract, Transform, Load* (ETL) dengan tujuan utama menyalin data dari *database* odoo dan PMS ke *data warehouse* hospitalitydwh melalui *pipeline* yang dibuat secara khusus. Dalam konteks ini, ADF berfungsi sebagai alat

orkestrasi data yang mendukung proses ETL *scalable* dan *reliable*. Setiap *pipeline* dibuat untuk mengatur alur kerja data yang dimulai dengan pengambilan data (*extraction*) dari *database* sumber, kemudian *transformasi* sesuai kebutuhan, dan akhirnya memuat data (*load*) yang telah terstruktur ke dalam *data warehouse* tujuan. *Activity* dalam *pipeline* ini termasuk operasi seperti *Lookup*, *Set Variable*, *Truncate Table*, dan *Copy Data* yang memastikan data diolah dengan benar dan siap digunakan. Penggunaan ADF dalam proyek ini terintegrasi dengan PostgreSQL dan Microsoft SQL Server melalui DBeaver menggunakan Hospitality-Vnet-IntegrationRuntime. Gambar 3.13 menampilkan *user interface* dari Azure Data Factory CITIS KG.



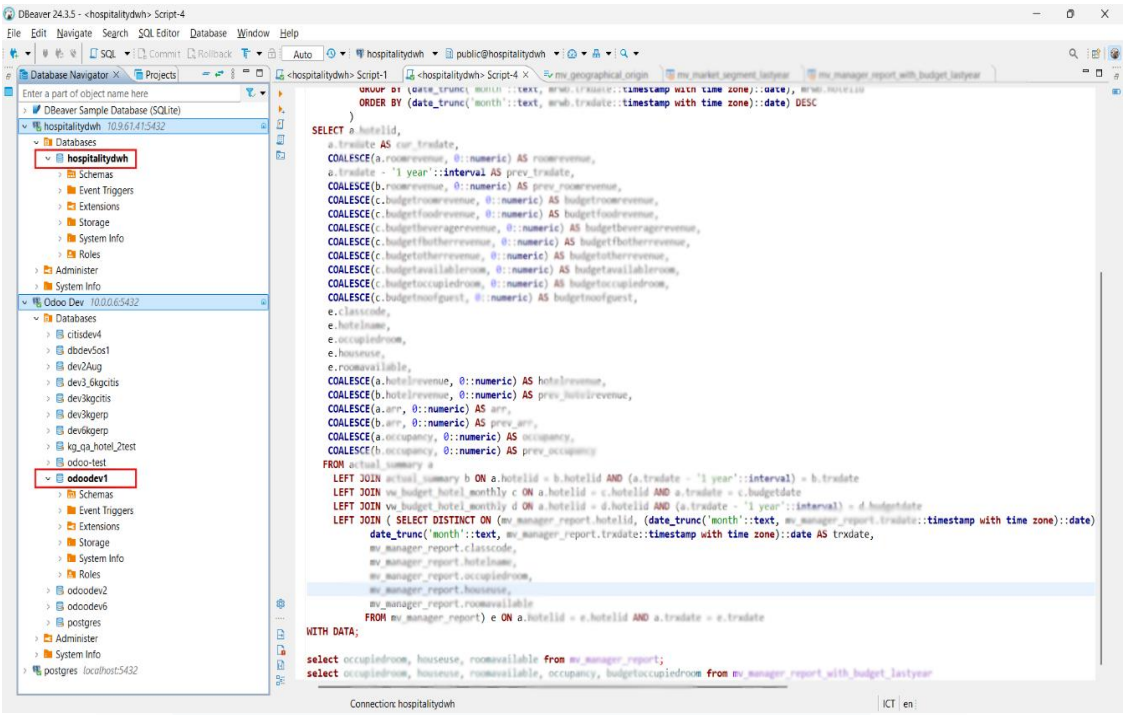
Gambar 3.13 Tampilan Azure Data Factory CITIS KG

2. *Tools* DBeaver (PostgreSQL)

Gambar 3.14 Logo dari kiri: DBeaver, PostgreSQL

DBeaver (PostgreSQL) dengan logo pada Gambar 3.14, merupakan alat manajemen *database* yang sangat berguna dalam pengelolaan data, terutama dalam konteks pengolahan data menggunakan PostgreSQL. Sebagai GUI (*Graphical User Interface*), DBeaver membantu pengguna untuk berinteraksi dengan *database* dengan cara yang lebih mudah dan praktis tanpa memerlukan kode SQL yang kompleks [39]. Dalam konteks magang, DBeaver digunakan sebagai *workbench* utama untuk mengelola *database* odoo (sumber *database*) dan *data warehouse* hospitalitydwh (*data warehouse* tujuan) menggunakan PostgreSQL. Melalui DBeaver, proses ETL menggunakan Azure Data Factory dapat dilakukan dengan cepat, serta mendukung pembuatan dan pengelolaan *Materialized View* (MV) yang digunakan sebagai sumber *dataset* untuk *dashboard* di Apache Superset.

DBeaver juga memberikan fitur editor SQL dengan *auto-completion* dan *syntax highlighting* yang memudahkan dalam menulis *query* yang diperlukan, serta kemampuan untuk mengekspor dan mengimpor data dalam berbagai format. Kemampuan DBeaver dalam mengelola skema *database*, memverifikasi integritas data, dan mengeksekusi *query* dengan cepat sangat penting untuk memastikan kelancaran dan ketepatan data yang digunakan dalam *dashboard*, mendukung peserta magang dalam mengelola *database*, memverifikasi hasil ETL, dan menyiapkan *dataset* untuk visualisasi *dashboard*. Untuk menghubungkan DBeaver dengan PostgreSQL, peserta magang dibantu oleh mentor dan diberikan akses oleh supervisor untuk menambahkan koneksi baru, memilih PostgreSQL sebagai tipe koneksi, serta memasukkan detail *host*, *port*, nama *database*, dan kredensial *login*. Gambar 3.15 menampilkan *user interface* dari DBeaver dan *database hospitality* KG.



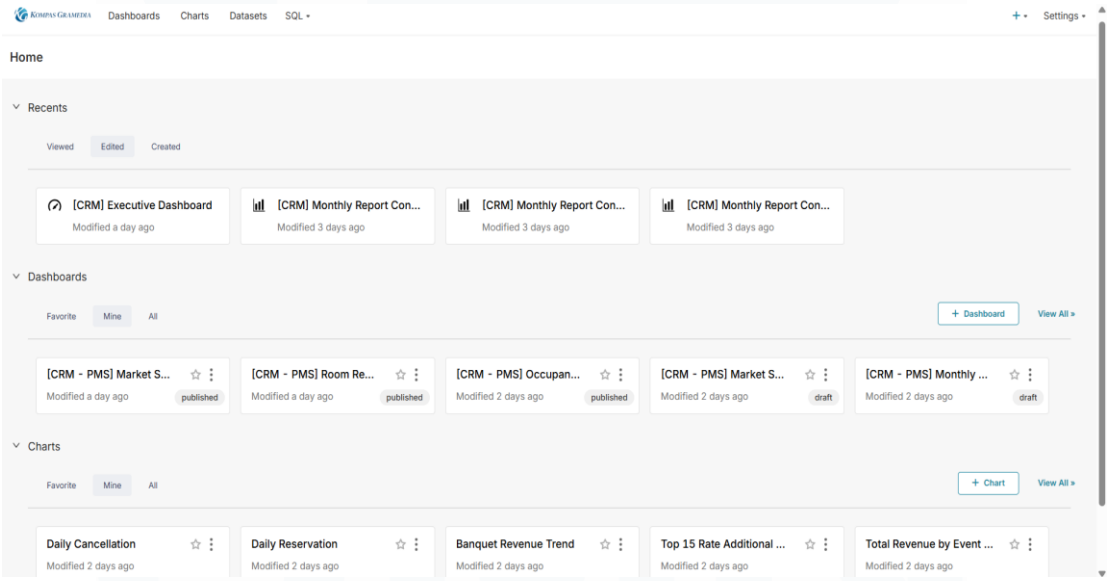
Gambar 3.15 Tampilan DBeaver dan Database PostgreSQL *Hospitality* KG

3. *Tools* Apache Superset

Gambar 3.16 Logo Apache Superset

Apache Superset dengan logo pada Gambar 3.16, merupakan *platform business intelligence* (BI) *open-source* yang modern, fleksibel, dan sangat skalabel untuk eksplorasi data yang cepat dan pembuatan visualisasi *dashboard* interaktif [40]. Sebagai proyek inkubasi Apache, Superset dapat menghubungkan berbagai sumber data (termasuk PostgreSQL, MySQL, Oracle, BigQuery, Snowflake, dll). Superset mendukung eksplorasi data *ad-hoc* melalui SQL Lab, pembuatan visualisasi dengan *chart builder* yang menawarkan berbagai tipe *chart*, serta penyusunan *layout dashboard* menggunakan *dashboard editor* untuk kebutuhan analitik. Setelah dilakukan instalasi dan konfigurasi, *tools* ini digunakan untuk eksplorasi data, membuat *charts*, serta merancang *dashboard*. Pada lingkungan kerja *Hospitality* KG, Superset menjadi *platform* yang digunakan untuk membuat

berbagai tipe *chart*, seperti grafik reservasi harian, tren pendapatan *banquet*, dan statistik tingkat hunian kamar, yang semua ini digunakan untuk mendukung pengambilan keputusan *Board of Directors* hotel. Dalam prosesnya, data yang telah melalui proses ETL menggunakan ADF akan dikelola di DBeaver (PostgreSQL) dalam bentuk *Materialized Views* (MV) yang kemudian menjadi sumber data dalam *dashboard*. Superset memberikan fleksibilitas dalam mendefinisikan metrik, menulis *query* SQL untuk eksplorasi data, serta mengelola *dataset* yang dapat digunakan oleh berbagai tim. Superset juga menyediakan ID khusus untuk *dashboard* dan ID filter yang dapat digunakan untuk mempermudah proses *embedding dashboard* ke dalam sistem *Property Management System* (PMS). Gambar 3.17 menampilkan *user interface* dari Apache Superset *hospitality* KG.



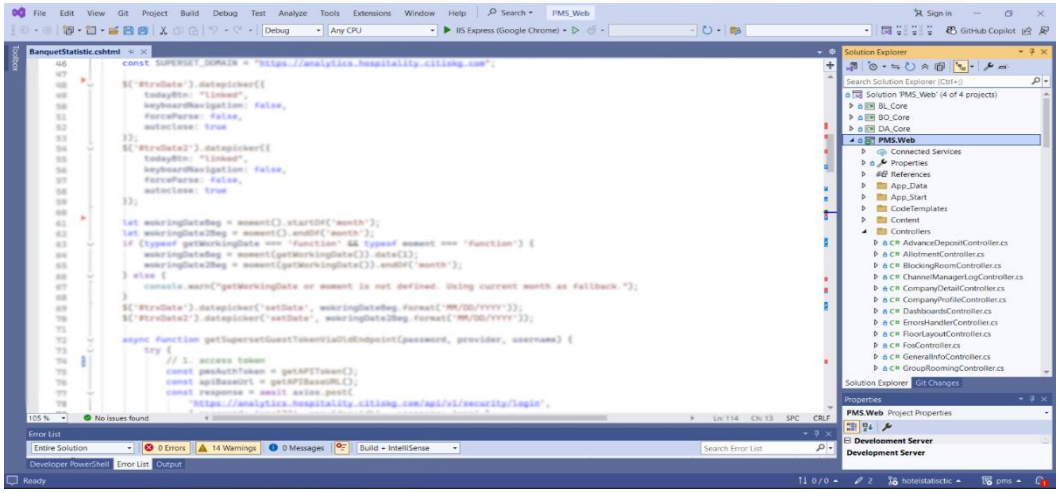
Gambar 3.17 Tampilan Apache Superset *Hospitality* KG

4. *Tools* Microsoft Visual Studio

Gambar 3.18 Logo Microsoft Visual Studio

Microsoft Visual Studio dengan logo pada Gambar 3.18, merupakan *Integrated Development Environment* (IDE) yang dirancang untuk

mendukung seluruh siklus hidup pengembangan *software*. Visual Studio menyediakan lingkungan terpadu bagi para *developer* untuk menulis, menguji, men-debug, dan me-*deploy* aplikasi untuk berbagai *platform*, termasuk web, *desktop*, *mobile*, dan *cloud* [41]. IDE ini mendukung berbagai bahasa pemrograman seperti C#, Visual Basic .NET, C++, Python, JavaScript, TypeScript, dan banyak lagi. Dalam magang ini, Visual Studio digunakan untuk mengembangkan dan mengelola kode yang diperlukan untuk meng-*embed dashboard* Superset ke dalam sistem *Property Management System* (PMS) Kompas Gramedia. Dengan fitur seperti *IntelliSense* untuk *autocompletion*, *debugger* untuk memperbaiki masalah, dan kemampuan integrasi dengan sistem kontrol versi seperti Git, Visual Studio menyediakan semua *tools* yang diperlukan untuk memastikan bahwa *dashboard* dapat diakses dan berfungsi dengan baik dalam sistem. Selain itu, Visual Studio juga membantu pengguna untuk memodifikasi baik *frontend* maupun *backend* dari sistem PMS, memastikan bahwa visualisasi data dari Superset dapat ditampilkan dengan mulus menggunakan metode seperti *Application Programming Interface* (API) dan *Software Development Kit* (SDK). Proses ini memastikan bahwa hasil analitik yang telah diciptakan dapat langsung diterapkan dalam sistem operasional hotel. Gambar 3.19 menampilkan *user interface* dari Microsoft Visual Studio *hospitality* KG.



Gambar 3.19 Tampilan Microsoft Visual Studio *Hospitality* KG

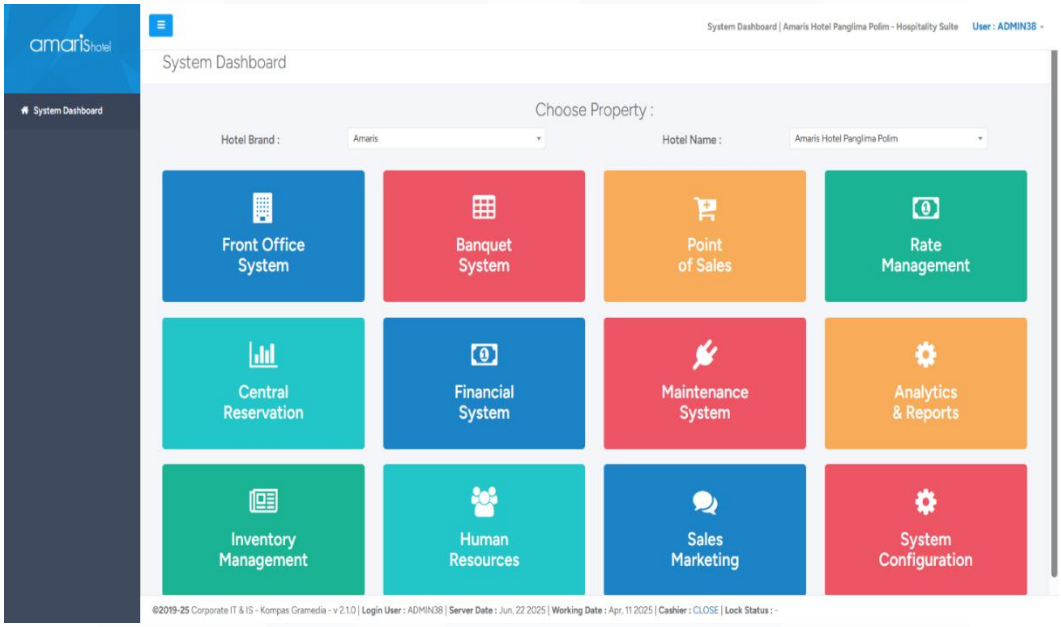
3.2.4. *Tools Introduction* – Eksplorasi Sistem, Fitur, dan Data pada Sistem Hotel

1. ***Property Management System* (PMS)**

Gambar 3.20 Tampilan Awal Login PMS *Hospitality* KG

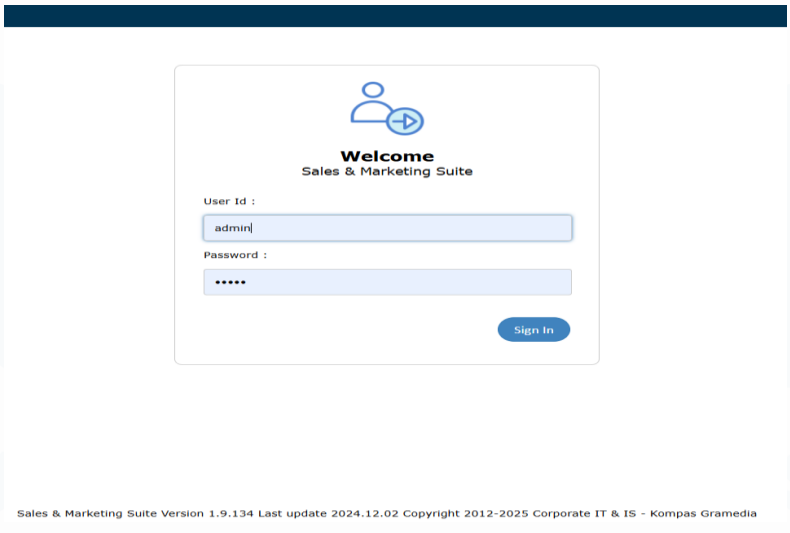
Property Management System (PMS) dengan tampilan login awal seperti pada Gambar 3.20, merupakan *software* yang digunakan untuk mengelola berbagai aspek operasional hotel, termasuk pemesanan, *check-in* & *check-out* tamu, pengelolaan inventaris kamar, penagihan, dan akuntansi. Sistem ini menjadi pusat kendali yang menyatukan berbagai modul yang saling terintegrasi, seperti *Front Office*, Reservasi, *Housekeeping*, *Point of Sale* (POS), Keuangan, dan *Customer Relationship Management* (CRM), yang semuanya bertujuan untuk meningkatkan kinerja operasional dan pengalaman tamu. Dalam konteks magang di Kompas Gramedia, PMS yang digunakan oleh hotel-hotel di bawah PT. Grahawita Santika berfungsi sebagai sumber data utama yang terhubung dengan sistem POS Odoo. *Dashboard* yang dibuat di Superset akan diintegrasikan langsung ke dalam PMS, khususnya dalam modul "*Analytics* & *Reports*", yang membantu manajemen hotel untuk mengakses *insight* data secara *real-time* tanpa perlu berpindah *platform*. Pengintegrasian *dashboard* dalam PMS memberikan kemudahan bagi pengambil keputusan untuk menggunakan data yang telah

diproses, mendorong pengambilan keputusan yang lebih cepat dan tepat berdasarkan informasi yang lebih jelas dan terstruktur. Gambar 3.21 menampilkan *user interface* dari PMS *hospitality* KG.



Gambar 3.21 Tampilan PMS *Hospitality* KG

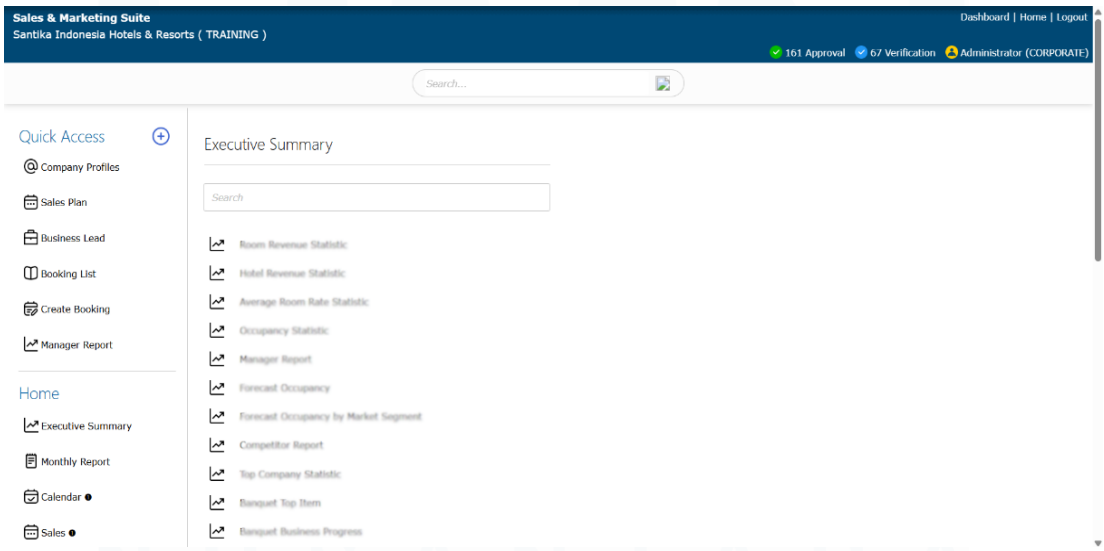
2. *Customer Relationship Management* (CRM)



Gambar 3.22 Tampilan Awal Login Sistem CRM *Hospitality* KG

Customer Relationship Management (CRM) dengan tampilan *login* awal seperti pada Gambar 3.22, merupakan sistem yang digunakan untuk mengelola dan menganalisis interaksi pelanggan dan data sepanjang siklus

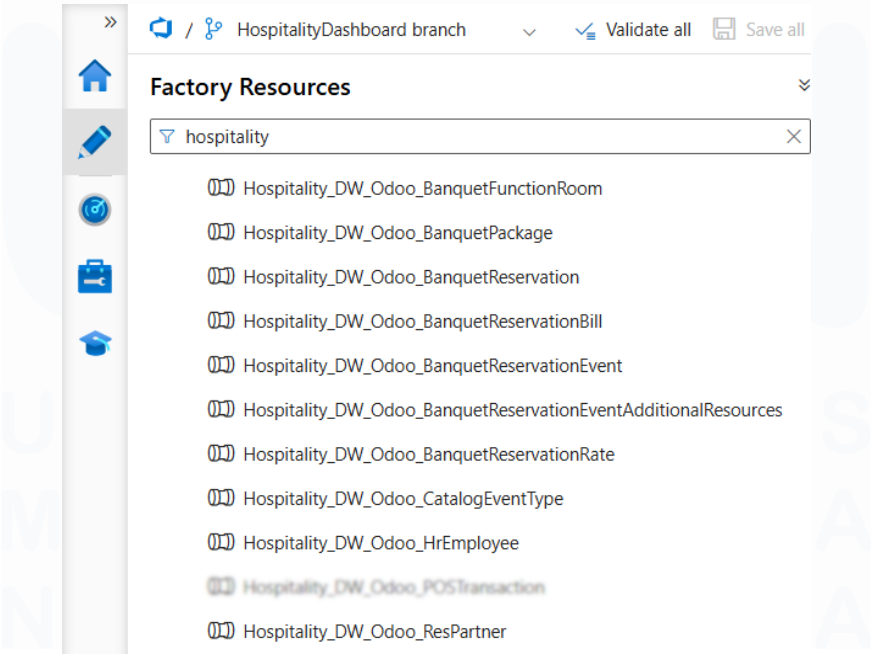
hidup pelanggan dengan tujuan meningkatkan hubungan bisnis dengan pelanggan, membantu retensi pelanggan, dan mendorong pertumbuhan penjualan. Sistem CRM dalam industri perhotelan mengkonsolidasikan informasi pelanggan dari berbagai saluran komunikasi ke dalam satu *platform* terpusat, yang penting untuk memberikan pengalaman *personal*, mengelola program loyalitas, menargetkan kampanye pemasaran, dan memantau umpan balik tamu. Di Kompas Gramedia, sistem CRM yang digunakan oleh Santika Indonesia *Hotels & Resorts* mendukung akses cepat ke berbagai fungsi terkait penjualan dan pemasaran, seperti "*Sales Plan*", "*Business Lead*", dan "*Booking List*". Tampilan dari sistem CRM ini menampilkan *dashboard* yang diperlukan oleh manajemen untuk menilai kinerja dan mengambil keputusan strategis. Dalam konteks magang, *dashboard* yang ada dalam CRM menjadi dasar pembuatan *dashboard* yang akan dipindahkan ke dalam sistem PMS untuk konsolidasi data yang lebih praktis. Dengan memindahkan *dashboard* dari CRM ke PMS, Kompas Gramedia bertujuan mengintegrasikan seluruh laporan analitik data ke dalam satu *platform* terpusat untuk mempermudah akses bagi *Board of Directors* dan meningkatkan kinerja operasional. Gambar 3.23 menampilkan *user interface* dari sistem CRM *hospitality* KG.



Gambar 3.23 Tampilan Sistem CRM *Hospitality* KG

3.2.5. ETL *Process* – Alur *Pipeline Data Banquet* Untuk Kebutuhan *Banquet Dashboard*

Proses *Extract, Transform, Load* (ETL) merupakan salah satu tugas penting yang dilakukan sebelum perancangan *dashboard* untuk operasional perhotelan di Kompas Gramedia, khususnya dalam hal ini terkait dengan data *banquet*. Pada tahap ini, data mentah yang berasal dari berbagai sumber seperti dari *database* odoo perlu dipindahkan ke dalam *data warehouse* hospitalitydwh untuk kebutuhan *dashboard*. Dalam konteks ini, beberapa alur *pipeline* data dirancang menggunakan Azure Data Factory untuk menyalin, mentransformasikan, dan memuat data dari sumber *database* odoo ke dalam *data warehouse* hospitalitydwh. Dalam proses ETL data *banquet*, peserta magang membangun sepuluh *pipeline* dengan tujuan yang berbeda untuk mengalirkan berbagai sumber data, yang ditunjukkan pada Gambar 3.24. Proses ETL ini mempersiapkan data dari berbagai aspek *banquet* yang melibatkan banyak entitas, seperti ruangan, paket, reservasi, dan transaksi untuk kemudian digunakan sebagai sumber data yang penting bagi pembuatan *materialized views* di PostgreSQL. Tabel 3.3 menunjukkan rincian fokus dari masing-masing sepuluh *pipeline* dalam proses ETL data *banquet*.

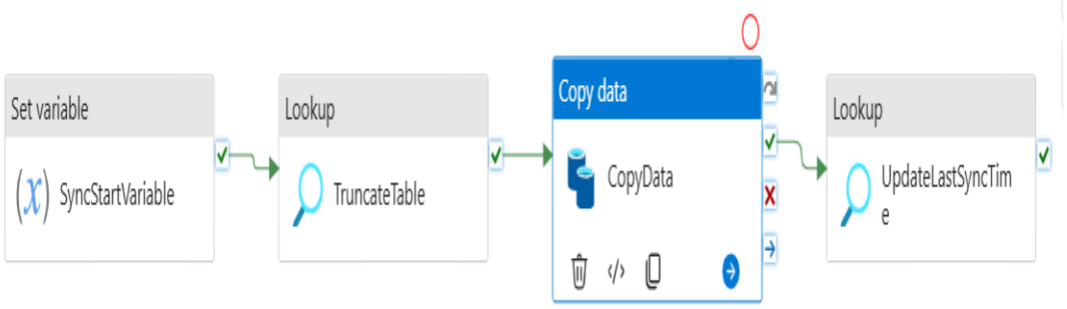


Gambar 3.24 Daftar *Pipeline* ETL Data *Banquet*

Tabel 3.3 Rincian Fokus *Pipeline Banquet*

Nama *Pipeline*	*Database* Sumber	Tabel Sumber	*Database* Tujuan	Tabel Tujuan
BanquetFunctionRoom	odoo	banquet_function_room	hospitalitydwh	odoo_bqtfunctionroom
BanquetPackage	odoo	banquet_package	hospitalitydwh	odoo_bqtpackage
BanquetReservation	odoo	banquet_reservation	hospitalitydwh	odoo_bqtreservation
BanquetReservationBill	odoo	banquet_reservation_bill	hospitalitydwh	odoo_bqtreservationbill
BanquetReservationEvent	odoo	banquet_reservation_event	hospitalitydwh	odoo_bqtreservationevent
BanquetReservationEvent AdditionalResources	odoo	banquet_reservation_event_additional_resources	hospitalitydwh	odoo_bqtreservationevent additionalresources
BanquetReservationRate	odoo	banquet_reservation_rate	hospitalitydwh	odoo_bqtreservationrate
CatalogEventType	odoo	catalog_eventtype	hospitalitydwh	odoo_catalogeventtype
HrEmployee	odoo	hr_employee	hospitalitydwh	odoo_hremployee
ResPartner	odoo	res_partner	hospitalitydwh	odoo_respartner

3.2.5.1. *Pipeline* Hospitality_DW_Odoo_BanquetFunctionRoom

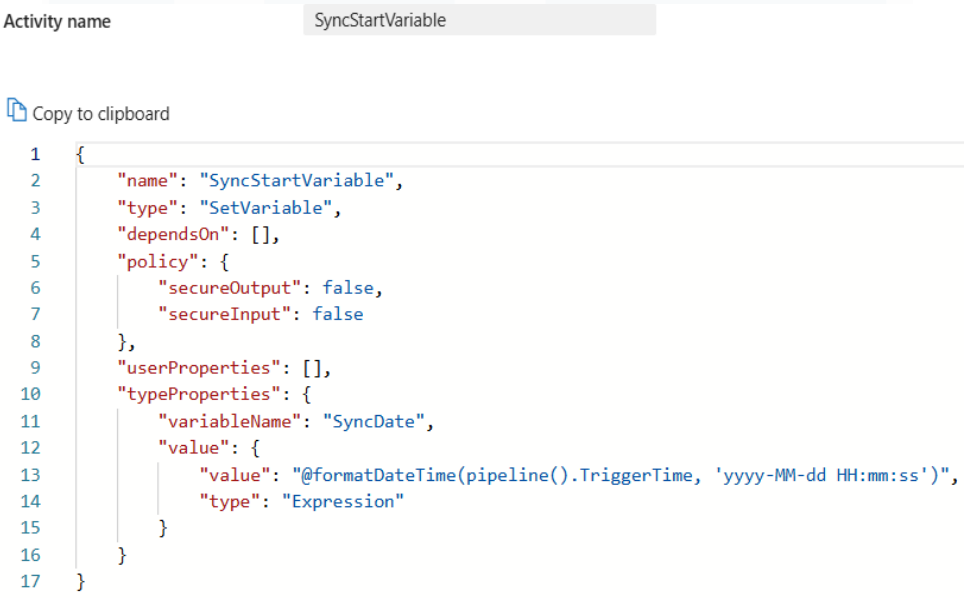


Gambar 3.25 Alur *Pipeline* Hospitality_DW_Odoo_BanquetFunctionRoom

Alur *pipeline* Hospitality_DW_Odoo_BanquetFunctionRoom seperti pada Gambar 3.25 dirancang untuk menyalin dan mentransformasikan data ruangan *banquet* dan *function room* dari *database* odoo ke *data warehouse* hospitalitydwh. Proses *Extract, Transform, Load* (ETL) dimulai dengan *activity Set variable* yang mengatur parameter awal, seperti waktu sinkronisasi terakhir, yang memastikan bahwa data yang diproses adalah yang terbaru. Selanjutnya, *pipeline* menggunakan *activity Lookup* untuk mengambil data terbaru dari sumber atau memeriksa keberadaan data yang diperlukan, diikuti oleh langkah *TruncateTable* untuk membersihkan tabel tujuan di hospitalitydwh agar data yang baru dapat dimasukkan tanpa duplikasi. Proses

inti dari *pipeline* ini adalah *activity CopyData*, yang mentransfer data aktual dari tabel banquet_function_room (odoo) ke tabel tujuan odoo_bqtfunctionroom (hospitalitydwh), dengan pengaturan pemetaan yang tepat antara kolom sumber dan kolom tujuan, serta penyesuaian tipe data yang diperlukan. Setelah proses pemindahan data selesai, *pipeline* diakhiri dengan *Lookup* untuk memperbarui *timestamp* sinkronisasi dan memastikan bahwa data *function room* yang baru telah berhasil disalin.

a. *Activity SyncStartVariable*



Gambar 3.26 *Code Activity SyncStartVariable* dalam *Pipeline* BanquetFunctionRoom

Activity SyncStartVariable dengan kode seperti pada Gambar 3.26 berfungsi untuk menginisialisasi sebuah variabel *pipeline* yang akan digunakan untuk menandai waktu mulai sinkronisasi. Variabel yang dibuat diberi nama "*SyncDate*". Nilai dari variabel "*SyncDate*" ini diisi secara dinamis menggunakan ekspresi *@formatDateTime(pipeline().TriggerTime, 'yyyy-MM-dd HH:mm:ss')*, yang berarti waktu pemicu (*trigger*) *pipeline* saat ini akan diformat menjadi *string* tanggal dan waktu (tahun-bulan-tanggal jam:menit:detik). Dengan demikian, setiap kali *pipeline* ini berjalan, variabel *SyncDate* akan menyimpan *timestamp* yang akurat kapan proses ETL data dimulai. Langkah ini termasuk ke dalam tahap *extract*

(*preparation phase*) karena berperan sebagai penanda waktu yang akan digunakan dalam proses penarikan data terbaru dari sumber.

b. *Activity TruncateTable*

Activity name: TruncateTable

Copy to clipboard

```
{
    "name": "TruncateTable",
    "type": "Lookup",
    "dependsOn": [
        {
            "activity": "SyncStartVariable",
            "dependencyConditions": [
                "Succeeded"
            ]
        }
    ],
    "policy": {
        "timeout": "0.00:30:00",
        "retry": 5,
        "retryIntervalInSeconds": 30,
        "secureOutput": false,
        "secureInput": false
    },
    "userProperties": [],
    "typeProperties": {
        "source": {
            "type": "AzurePostgreSqlSource",
            "query": "SELECT truncate_table('odoo_bqtfunctionroom')",
            "partitionOption": "None"
        },
        "dataset": {
            "referenceName": "Hospitality_DW_SyncTableLog",
            "type": "DatasetReference"
        }
    }
```

Gambar 3.27 *Code Activity TruncateTable* dalam *Pipeline* BanquetFunctionRoom

Activity TruncateTable bertipe *Lookup* dengan kode seperti pada Gambar 3.27 digunakan untuk menjalankan *query* SQL ke *data warehouse* tujuan (hospitalitydwh). *Activity* ini bergantung pada keberhasilan *SyncStartVariable*, memastikan *SyncDate* sudah terset. *Query* yang dieksekusi adalah *SELECT truncate_table('odoo_bqtfunctionroom')*, yang memanggil *function* kustom PostgreSQL bernama *truncate_table*. *Function truncate_table* dengan *query* seperti pada Gambar 3.28 ini didefinisikan dalam DBeaver (PostgreSQL) yang bertujuan untuk mengosongkan (*truncates*) tabel yang namanya diberikan sebagai parameter

(odoo_bqtfunctionroom). Dengan demikian, sebelum data baru disalin, *activity* ini memastikan tabel odoo_bqtfunctionroom di hospitalitydwh benar-benar bersih dari data lama. Langkah ini termasuk ke dalam tahap *load* (*preparation phase*) karena berfungsi membersihkan tabel tujuan agar proses pemuatan data berikutnya berlangsung tanpa duplikasi atau konflik.

Gambar 3.28 *Query Function truncate_table*

c. Activity CopyData

Activity CopyData dengan kode seperti pada Gambar 3.29 adalah inti dari proses ETL data yang berfungsi untuk menyalin data dari *source database* ke *destination database*. *Activity* ini bergantung pada keberhasilan *TruncateTable*, memastikan tabel tujuan sudah kosong sebelum penyalinan dimulai. Sumber data adalah PostgreSQL odoo, sementara tujuan data adalah PostgreSQL hospitalitydwh, dengan strategi *CopyCommand* untuk mempercepat transfer data. Pada bagian *translator* menunjukkan bahwa ada pemetaan kolom otomatis dan konversi tipe data yang terjadi selama proses penyalinan, memastikan data bersih dan siap pakai di *data warehouse* tujuan. *Activity* ini termasuk dalam kategori *extract, transform, load* sekaligus karena ADF menjalankan penarikan data dari *source* (*extract*), melakukan konversi dan pemetaan kolom (*transform*), serta menyimpannya ke dalam *destination table* (*load*) secara terintegrasi dalam satu proses.

Gambar 3.29 *Code Activity CopyData* dalam *Pipeline* BanquetFunctionRoom

d. *Activity UpdateLastSyncTime*

Activity UpdateLastSyncTime bertipe *Lookup* dengan kode seperti pada Gambar 3.30 adalah langkah terakhir dalam *pipeline* yang bergantung pada keberhasilan *activity CopyData*. Tujuannya adalah untuk memperbarui log sinkronisasi di *data warehouse* hospitalitydwh setelah data berhasil disalin. *Query* yang dieksekusi adalah *SELECT update_synctablelog('odoo_bqtfunctionroom','@{variables('SyncDate')}');* yang memanggil *function* PostgreSQL *update_synctablelog*. *Function* dengan *query* seperti pada Gambar 3.31 ini dirancang untuk memperbarui *lastupdate* dari tabel dw_synctablelog dengan *SyncDate* yang diambil dari

variabel *pipeline*, menggunakan nama tabel (odoo_bqtfunctionroom) sebagai kunci. Jika nama tabel belum ada di dw_synctablelog, *function* ini akan memasukkan entri baru. Dengan demikian, *activity* ini mencatat waktu terakhir sinkronisasi data untuk odoo_bqtfunctionroom, memberikan jejak audit dan status terkini dari proses ETL. *Activity* ini termasuk kategori *post-load* (*administrative step*) karena berfungsi mencatat waktu sinkronisasi terakhir untuk mendukung proses *extract* berikutnya agar hanya memproses data terbaru.

Activity name UpdateLastSyncTime

Copy to clipboard

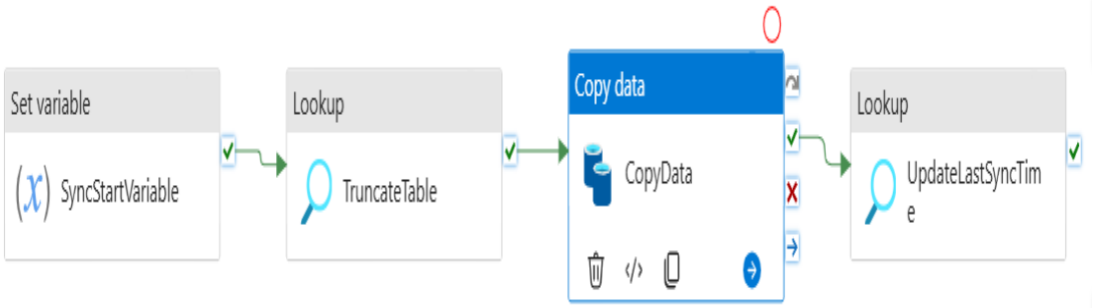
```
{
    "name": "UpdateLastSyncTime",
    "type": "Lookup",
    "dependsOn": [
        {
            "activity": "CopyData",
            "dependencyConditions": [
                "Succeeded"
            ]
        }
    ],
    "policy": {
        "timeout": "0.00:30:00",
        "retry": 5,
        "retryIntervalInSeconds": 30,
        "secureOutput": false,
        "secureInput": false
    },
    "userProperties": [],
    "typeProperties": {
        "source": {
            "type": "AzurePostgreSqlSource",
            "query": {
                "value": "SELECT update_synctablelog('odoo_bqtfunctionroom', '@{variables('SyncDate')}');",
                "type": "Expression"
            },
            "partitionOption": "None"
        },
        "dataset": {
            "referenceName": "Hospitality_DW_SyncTableLog",
            "type": "DatasetReference"
        },
        "firstRowOnly": false
    }
}
```

Gambar 3.30 *Code Activity UpdateLastSyncTime* dalam *Pipeline* BanquetFunctionRoom

```
CREATE OR REPLACE FUNCTION public.update_synctablelog(synctable text, updatedate timestamp without time zone)
 RETURNS void
 LANGUAGE plpgsql
AS $function$
BEGIN
  IF EXISTS (
    SELECT 1
    FROM dw_synctablelog
    WHERE tablename = synctable
  ) THEN
    UPDATE dw_synctablelog
    SET lastupdate = updatedate
    WHERE tablename = synctable;
  ELSE
    INSERT INTO dw_synctablelog (tablename, lastupdate)
    VALUES (synctable, updatedate);
  END IF;
END;
$function$
;
```

Gambar 3.31 *Query Function update_synctablelog*

3.2.5.2. *Pipeline* Hospitality_DW_Odoo_BanquetPackage



Gambar 3.32 Alur *Pipeline* Hospitality_DW_Odoo_BanquetPackage

Alur *pipeline* Hospitality_DW_Odoo_BanquetPackage seperti pada Gambar 3.32 dirancang untuk menyalin dan mentransformasikan data paket *banquet* dan fasilitas acara dari *database* odoo ke *data warehouse* hospitalitydwh, mencakup detail seperti harga, kapasitas, fasilitas, dan tautan ke banquet_function_room jika paket terkait dengan ruangan tertentu. Proses dimulai dengan *activity Set variable*, yang menetapkan parameter awal seperti waktu mulai sinkronisasi data untuk memastikan bahwa data yang diproses adalah yang terbaru. Selanjutnya, *activity Lookup* digunakan untuk mengambil data relevan dari sumber eksternal, yang mendukung pengecekan status atau nilai tertentu yang diperlukan dalam *pipeline*. Tahap berikutnya adalah *TruncateTable*, yang mengosongkan tabel tujuan di hospitalitydwh agar data

yang baru dapat dimasukkan tanpa mengganggu integritas data sebelumnya. Proses utama dalam *pipeline* ini adalah *CopyData*, yang menyalin data dari tabel banquet_package (odoo) ke tabel odoo_bqtpackage (hospitalitydwh), dengan pemetaan yang jelas antara kolom sumber dan kolom tujuan. Setelah proses pemindahan data selesai, *activity Lookup* yang terakhir digunakan untuk memperbarui *timestamp* sinkronisasi, menandakan bahwa data paket *banquet* telah berhasil disalin.

Proses alur *pipeline* Odoo_BanquetPackage mirip dengan Odoo_BanquetFunctionRoom, namun berbeda pada *query*, tabel, dan alur *activity CopyData*. Secara umum, *activity SyncStartVariable*, *TruncateTable*, dan *UpdateLastSyncTime* memiliki penjelasan serupa dengan perbedaan pada nama tabel yang disalin. *SyncStartVariable* menggunakan variabel *SyncDate* untuk menandai waktu sinkronisasi, *TruncateTable* menjalankan *query SELECT truncate_table('odoo_bqtpackage')* untuk membersihkan data pada tabel odoo_bqtpackage, dan *UpdateLastSyncTime* memperbarui *timestamp* sinkronisasi di dw_synctablelog berdasarkan tabel odoo_bqtpackage dengan *query SELECT update_synctablelog('odoo_bqtpackage', '@{variables('SyncDate')}')*. Perbedaan utamanya terletak pada *activity CopyData*, dimana setiap pipeline memetakan dan menyalin data sesuai kebutuhan tabel yang dituju. Berikut penjelasan dari *activity CopyData* pada *pipeline* Odoo_BanquetPackage:

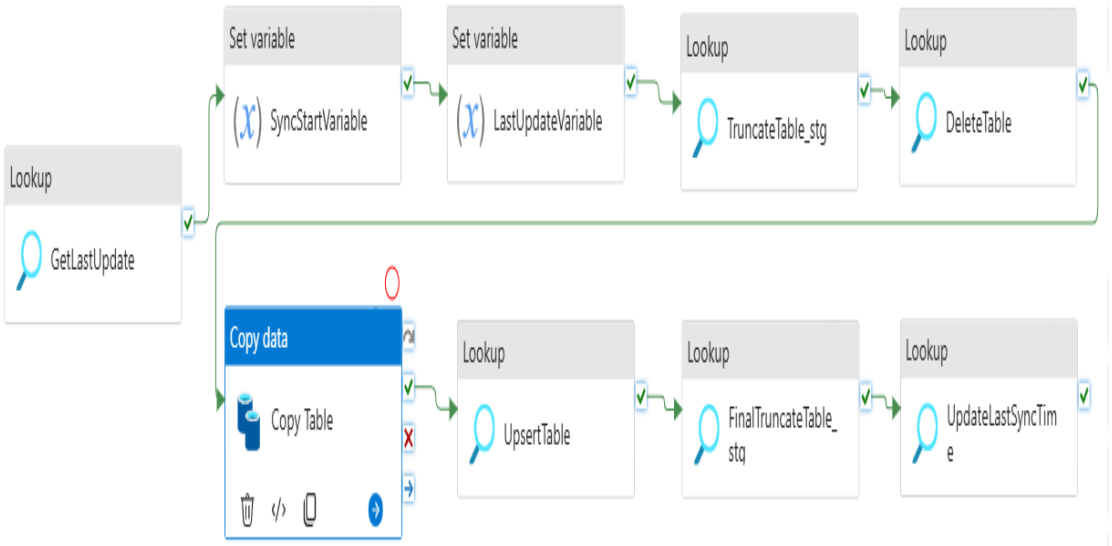
a. *Activity CopyData*

Activity CopyData dengan kode seperti pada Gambar 3.33 adalah inti dari proses ETL data yang berfungsi untuk menyalin data paket *banquet* dari *database* odoo ke *data warehouse* hospitalitydwh. Proses penyalinan ini baru akan dimulai setelah tabel tujuan berhasil dikosongkan oleh *activity TruncateTable*. Sumber data adalah *dataset* banquet_package dari *database* odoo dan akan disalin ke *dataset* banquet_package di *data warehouse* hospitalitydwh. Selama proses ini, ADF secara otomatis melakukan pemetaan dan penyesuaian tipe data antar kolom sumber dan tujuan melalui

properti *translator* (termasuk *mappings* dan *typeConversionSettings*) dalam *activity Copy Data*. *Activity* ini termasuk dalam kategori *extract*, *transform*, *load* sekaligus karena ADF menjalankan penarikan data dari *source* (*extract*), melakukan konversi dan pemetaan kolom (*transform*), serta menyimpan hasilnya ke dalam *destination table* (*load*) secara terintegrasi dalam satu proses.

Gambar 3.33 *Code Activity CopyData* dalam *Pipeline* BanquetPackage

3.5.2.3. *Pipeline* Hospitality_DW_Odoo_BanquetReservation



Gambar 3.34 Alur *Pipeline* Hospitality_DW_Odoo_BanquetReservation

Alur *pipeline* Hospitality_DW_Odoo_BanquetReservation seperti pada Gambar 3.34 dirancang untuk menyalin dan mentransformasikan data detail pemesanan *event* atau acara *banquet* dari *database source* odoo ke *data warehouse destination* hospitalitydwh menggunakan strategi pembaruan inkremental (*upsert*), yang memfokuskan pada data baru atau yang telah diubah sejak sinkronisasi terakhir. Tahap pertama adalah *activity Lookup GetLastUpdate*, yang mengambil *timestamp* sinkronisasi terakhir untuk menentukan data yang perlu diproses. Tahap kedua, *pipeline* mencatat waktu sinkronisasi dengan *Set variable SyncStartVariable*. Pada tahap ketiga, nilai dari *GetLastUpdate* diambil melalui *Set variable LastUpdateVariable*. Tahap keempat adalah *Lookup TruncateTable_stg*, yang mengosongkan tabel *staging* agar data baru dapat dimuat tanpa mengganggu tabel utama. Tahap kelima, *DeleteTable* menghapus data lama dari tabel utama untuk memastikan hanya data terbaru yang tersimpan. Tahap keenam, *Copy Table* menyalin data dari odoo ke tabel *staging* dengan filter *LastUpdateVariable* untuk memastikan hanya data yang diperbarui atau baru yang dipindahkan. Setelah itu, pada tahap ketujuh, *Lookup UpsertTable* melakukan operasi *upsert* untuk memperbarui atau memasukkan data baru ke dalam tabel utama BanquetReservation di

hospitalitydwh. Tahap kedelapan, *FinalTruncate_stg* mengosongkan tabel *staging*, dan tahap kesembilan adalah *UpdateLastSyncTime* memperbarui *timestamp* sinkronisasi terakhir.

a. *Activity GetLastUpdate*

Activity name GetLastUpdate

Copy to clipboard

```
{
    "name": "GetLastUpdate",
    "type": "Lookup",
    "dependsOn": [],
    "policy": {
        "timeout": "0.00:30:00",
        "retry": 5,
        "retryIntervalInSeconds": 30,
        "secureOutput": false,
        "secureInput": false
    },
    "userProperties": [],
    "typeProperties": {
        "source": {
            "type": "AzurePostgreSqlSource",
            "query": "select lastupdate from dw_synctablelog where tablename = 'odoo_bqtreservation'",
            "partitionOption": "None"
        },
        "dataset": {
            "referenceName": "Hospitality_DW_SyncTableLog",
            "type": "DatasetReference"
        }
    }
}
```

Gambar 3.35 *Code Activity GetLastUpdate* dalam *Pipeline* BanquetReservation

Activity GetLastUpdate bertipe *Lookup* dengan kode seperti pada Gambar 3.35 berfungsi untuk mengambil waktu pembaruan terakhir dari tabel dw_synctablelog yang berada di hospitalitydwh. *Query* SQL yang dijalankan adalah *select lastupdate from dw_synctablelog where tablename = 'odoo_bqtreservation'*, yang secara spesifik mengambil *lastupdate* untuk entri odoo_bqtreservation. Nilai *lastupdate* ini kemudian akan digunakan oleh *pipeline* untuk menentukan data inkremental mana yang perlu disalin dari *database* odoo. *Activity* ini krusial untuk implementasi ETL secara cepat dan hanya memproses perubahan. Langkah ini termasuk dalam kategori *extract* (*preparation phase*) karena digunakan untuk mengambil parameter batas waktu data terbaru yang akan diekstrak dari sistem sumber.

b. *Activity SyncStartVariable*

Activity SyncStartVariable bertipe *Set Variable* dengan kode seperti pada Gambar 3.36 berfungsi untuk menginisialisasi sebuah variabel *pipeline* yang akan digunakan untuk menandai waktu mulai sinkronisasi. Variabel yang dibuat diberi nama "*SyncDate*", dan *activity* ini bergantung pada keberhasilan *GetLastUpdate*. Nilai dari variabel "*SyncDate*" diisi secara dinamis menggunakan ekspresi *@formatDateTime(pipeline().TriggerTime, 'yyyy-MM-dd HH:mm:ss')*, yang berarti waktu pemicu *pipeline* saat ini akan diformat menjadi *string* tanggal dan waktu. Dengan demikian, *SyncDate* akan menyimpan *timestamp* akurat kapan proses ETL data reservasi *banquet* dimulai. Langkah ini termasuk dalam kategori *extract* (*preparation phase*) karena berfungsi mencatat waktu awal proses ETL sebagai parameter untuk audit dan *logging*.

Activity name GetLastUpdate

Copy to clipboard

```
{
    "name": "GetLastUpdate",
    "type": "Lookup",
    "dependsOn": [],
    "policy": {
        "timeout": "0.00:30:00",
        "retry": 5,
        "retryIntervalInSeconds": 30,
        "secureOutput": false,
        "secureInput": false
    },
    "userProperties": [],
    "typeProperties": {
        "source": {
            "type": "AzurePostgreSqlSource",
            "query": "select lastupdate from dw_synctablelog where tablename = 'odoo_bqtreservation'",
            "partitionOption": "None"
        },
        "dataset": {
            "referenceName": "Hospitality_DW_SyncTableLog",
            "type": "DatasetReference"
        }
    }
}
```

Gambar 3.36 *Code Activity SyncStartVariable* dalam *Pipeline* BanquetReservation

c. *Activity LastUpdateVariable*

Activity LastUpdateVariable bertipe *Set Variable* dengan kode seperti pada Gambar 3.37 berfungsi untuk menyimpan *timestamp* pembaruan terakhir yang diperoleh dari *activity* sebelumnya. Variabel yang dibuat

diberi nama "*LastUpdate*", dan *activity* ini bergantung pada keberhasilan *SyncStartVariable*. Nilainya diambil dari *output activity GetLastUpdate* melalui ekspresi *@formatDateTime(activity('GetLastUpdate').output.firstRow.lastupdate, 'yyyy-MM-dd HH:mm:ss')*, yang berarti nilai *lastupdate* dari baris pertama hasil *GetLastUpdate* diformat menjadi *string* tanggal dan waktu. Dengan demikian, variabel *LastUpdate* menyimpan *timestamp* dari sinkronisasi data reservasi *banquet* sebelumnya. Langkah ini termasuk dalam kategori *extract* (*preparation phase*) karena menyimpan nilai *timestamp* terakhir sebagai parameter filter untuk proses pengambilan data berikutnya.

Activity name LastUpdateVariable

Copy to clipboard

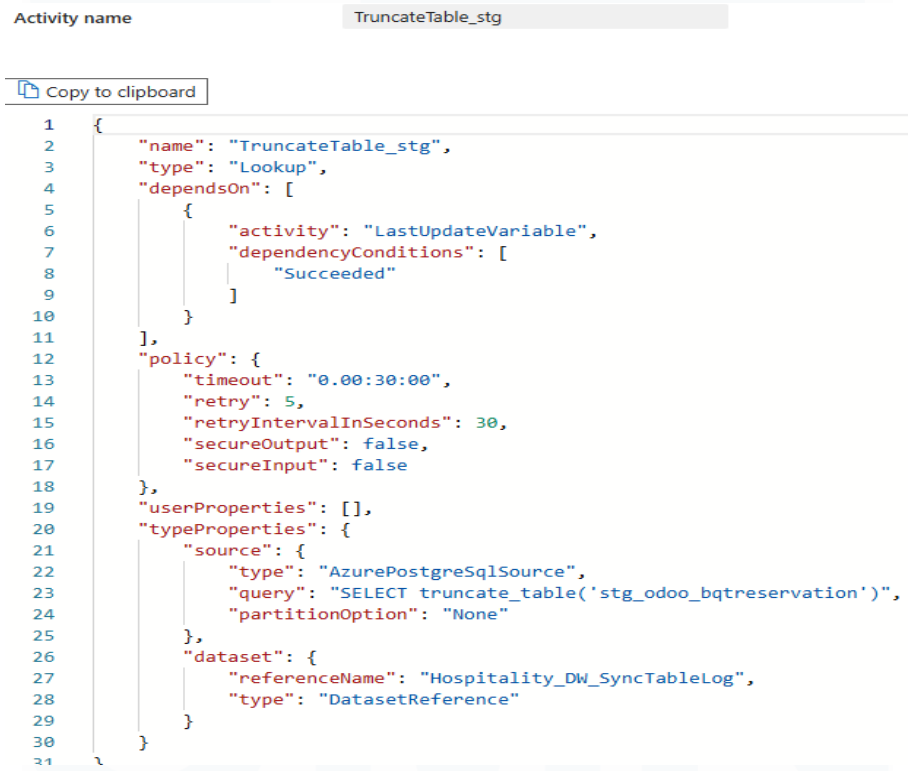
```
{
    "name": "LastUpdateVariable",
    "type": "SetVariable",
    "dependsOn": [
        {
            "activity": "SyncStartVariable",
            "dependencyConditions": [
                "Succeeded"
            ]
        }
    ],
    "policy": {
        "secureOutput": false,
        "secureInput": false
    },
    "userProperties": [],
    "typeProperties": {
        "variableName": "LastUpdate",
        "value": {
            "value": "@formatDateTime(activity('GetLastUpdate').output.firstRow.lastupdate, 'yyyy-MM-dd HH:mm:ss')",
            "type": "Expression"
        }
    }
}
```

Gambar 3.37 *Code Activity LastUpdateVariable* dalam *Pipeline* BanquetReservation

d. *Activity TruncateTable_stg*

Activity TruncateTable_stg bertipe *Lookup* dengan kode seperti pada Gambar 3.38 bergantung pada keberhasilan *LastUpdateVariable*, *activity* ini memastikan variabel waktu terakhir *update* sudah disiapkan. Fungsi utamanya adalah mengosongkan tabel *staging* yang bernama stg_odoo_bqtreservation di hospitalitydwh. *Query* yang dieksekusi adalah *SELECT truncate_table('stg_odoo_bqtreservation')*, yang memanggil

function kustom PostgreSQL *truncate_table*. *Function truncate_table* dengan *query* seperti pada Gambar 3.28, secara spesifik menjalankan perintah *TRUNCATE TABLE* pada nama tabel yang diberikan (stg_odoo_bqtreservation), menghapus semua baris data di dalamnya secara cepat. Dengan demikian, *activity* ini memastikan tabel *staging* bersih sebelum data reservasi *banquet* yang baru atau diperbarui disalin. Langkah ini termasuk dalam tahap *load* (*preparation phase*) karena berfungsi sebagai persiapan agar proses pemuatan data baru ke *staging* berjalan tanpa tumpang tindih dengan data lama.

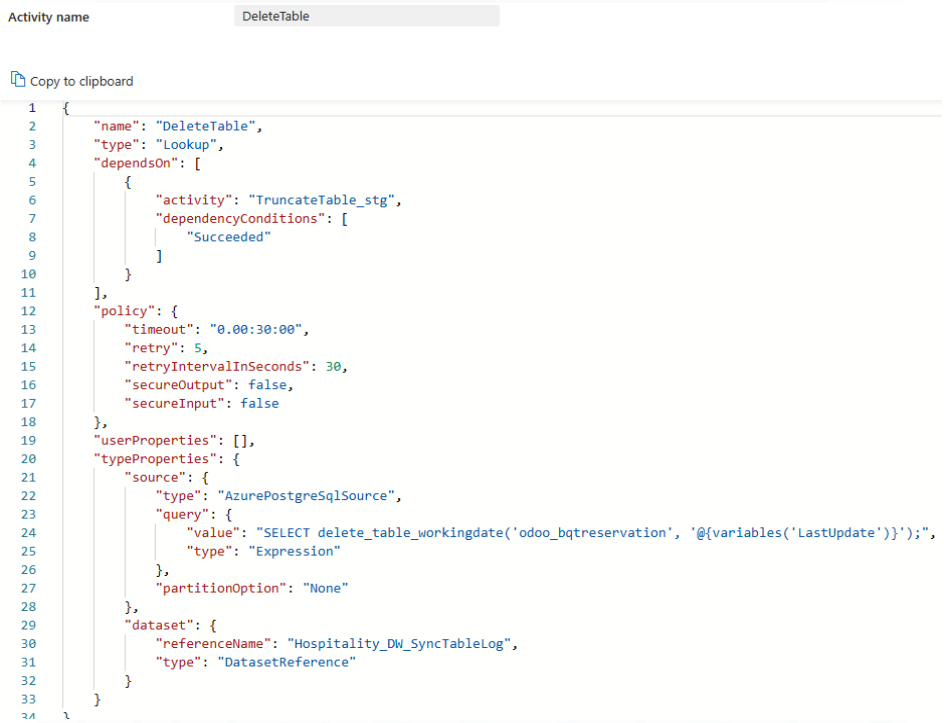


Gambar 3.38 *Code Activity TruncateTable_stg* dalam *Pipeline* BanquetReservation

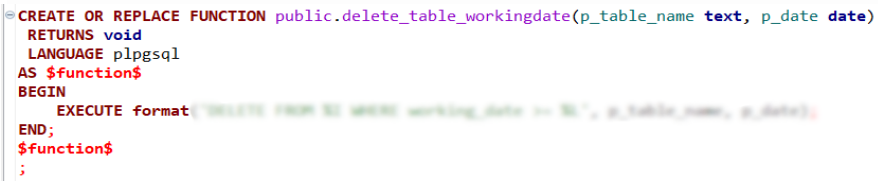
e. *Activity DeleteTable*

Activity DeleteTable bertipe *Lookup* dengan kode seperti pada Gambar 3.39 bergantung pada keberhasilan *TruncateTable_stg*, memastikan tabel *staging* sudah dikosongkan. Tujuan utamanya adalah menghapus data lama dari tabel reservasi *banquet* di hospitalitydwh yang akan digantikan oleh data yang lebih baru. *Query* yang dieksekusi adalah *SELECT*

delete_table_workingdate('odoo_bqtreservation','@{variables('LastUpdate')}'); yang memanggil *function* kustom PostgreSQL *delete_table_workingdate*. *Function delete_table_workingdate* dengan *query* seperti pada Gambar 3.40 ini dirancang untuk menghapus baris dari tabel tujuan (odoo_bqtreservation) dimana *working_date* lebih besar atau sama dengan *LastUpdate* yang diberikan sebagai parameter. Dengan demikian, *activity* ini secara selektif menghapus data yang telah diproses atau diperbarui dari periode waktu tertentu untuk mempersiapkan tabel utama untuk proses *upsert* data inkremental berikutnya. Langkah ini termasuk dalam tahap *load* (*preparation phase*) karena bertujuan membersihkan data lama pada tabel utama agar proses *upsert* data dapat berjalan dengan benar tanpa duplikasi.



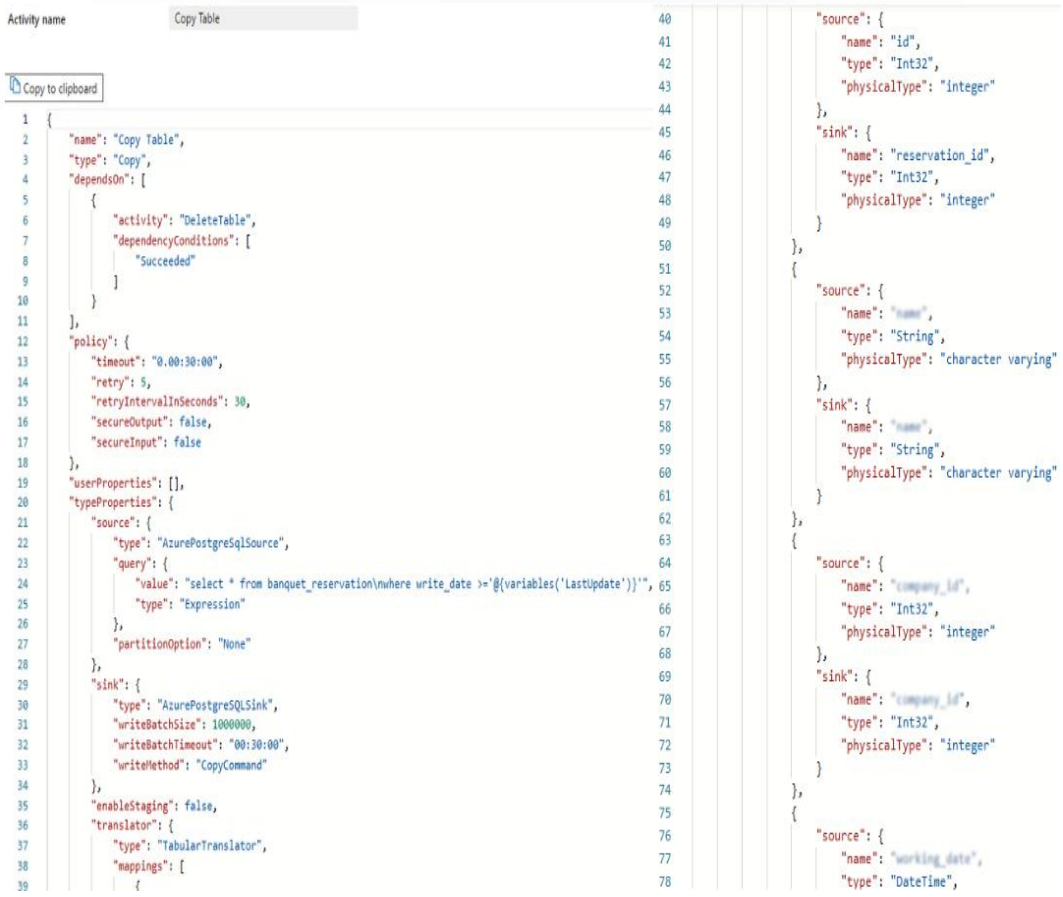
Gambar 3.39 *Code Activity DeleteTable* dalam *Pipeline* BanquetReservation



Gambar 3.40 *Query Function delete_table_workingdate*

f. *Activity Copy Table*

Activity Copy Table dengan kode seperti pada Gambar 3.41 adalah langkah utama untuk menyalin data reservasi *banquet* yang terbaru atau yang diperbarui dari *database* odoo ke tabel *staging* di hospitalitydwh. *Activity* ini bergantung pada keberhasilan *DeleteTable* yang sudah membersihkan data lama. *Query select * from banquet_reservation where write_date >='@{variables('LastUpdate')}'* secara selektif memilih hanya baris data yang memiliki *write_date* lebih baru dari *LastUpdate* terakhir, memastikan hanya data inkremental yang disalin. Data ini kemudian disalin ke *dataset* odoo_bqtreservation (hospitalitydwh), dengan pemetaan kolom dan konversi tipe data yang terjadi secara otomatis. Langkah ini mencakup proses *extract* dengan filter berdasarkan *LastUpdateVariable*, *transform* ringan berupa pemetaan dan konversi tipe data, serta *load* data ke tabel *staging* secara bersamaan dalam satu *activity*.



Gambar 3.41 *Code Activity Copy Table* dalam *Pipeline* BanquetReservation

g. *Activity UpsertTable*

Activity UpsertTable bertipe *Lookup* dengan kode seperti pada Gambar 3.42 bergantung pada keberhasilan *Copy Table*, memastikan data reservasi *banquet* inkremental sudah tersimpan di tabel *staging*. Tujuan utamanya adalah untuk melakukan operasi *upsert*, yaitu memperbarui data yang sudah ada atau menyisipkan data baru, ke tabel odoo_bqtreservation di hospitalitydwh. *Query* yang dieksekusi adalah *SELECT upsert_odoo_bqtreservation();*, yang memanggil *function* PostgreSQL *upsert_odoo_bqtreservation()*. *Function* dengan *query* seperti pada Gambar 3.43 ini berisi logika *INSERT ... ON CONFLICT* (reservation_id) *DO UPDATE SET ...*, yang berarti data dari tabel *staging* (stg_odoo_bqtreservation) akan disisipkan ke tabel utama. Jika reservation_id sudah ada (konflik), baris yang ada akan diperbarui dengan nilai-nilai terbaru dari data *staging*, memastikan tabel tujuan selalu berisi data reservasi *banquet* yang paling baru. Langkah ini merupakan proses

load yang melibatkan operasi *upsert* untuk memasukkan data baru atau memperbarui data yang sudah ada di tabel utama berdasarkan *primary key* atau ID.

Activity name: UpsertTable

Copy to clipboard

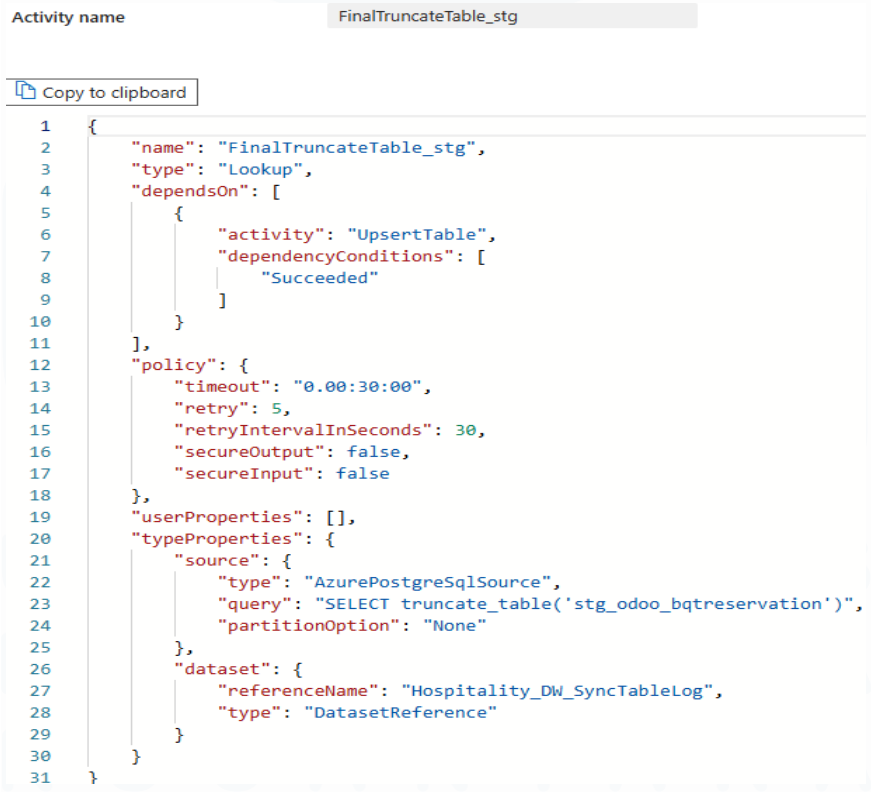
```
{
    "name": "UpsertTable",
    "type": "Lookup",
    "dependsOn": [
        {
            "activity": "Copy Table",
            "dependencyConditions": [
                "Succeeded"
            ]
        }
    ],
    "policy": {
        "timeout": "0.00:30:00",
        "retry": 5,
        "retryIntervalInSeconds": 30,
        "secureOutput": false,
        "secureInput": false
    },
    "userProperties": [],
    "typeProperties": {
        "source": {
            "type": "AzurePostgreSqlSource",
            "query": "SELECT upsert_odoo_bqtreservation(); ",
            "partitionOption": "None"
        },
        "dataset": {
            "referenceName": "Hospitality_DW_SyncTableLog",
            "type": "DatasetReference"
        }
    }
}
```

Gambar 3.42 *Code Activity UpsertTable* dalam *Pipeline* BanquetReservation

```
CREATE OR REPLACE FUNCTION public.upsert_odoo_bqtreservation()
 RETURNS void
 LANGUAGE plpgsql
AS $function$
BEGIN
  INSERT INTO odoo_bqtreservation (
    reservation_id,
    name,
    company_id,
    working_date,
    state,
    banquet_status,
    payment_state,
    reservation_date,
    event_type_id,
    currency_id,
    arrival_date,
    departure_date,
    total_balance_bill,
    total_amount,
    total_billed,
    total_net_transaction,
    is_definite,
    partner_id,
    )
    SELECT
      reservation_id,
      name,
      company_id,
      working_date,
      state,
      banquet_status,
      payment_state,
      reservation_date,
      event_type_id,
      currency_id,
      arrival_date,
      departure_date,
      total_balance_bill,
      total_amount,
      total_billed,
      total_net_transaction,
      is_definite,
      partner_id,
```

Gambar 3.43 *Query Function upsert_odoo_bqtreservation()*

h. *Activity FinalTruncateTable_stg*



Gambar 3.44 *Code Activity FinalTruncateTable_stg* dalam *Pipeline* BanquetReservation

Activity FinalTruncateTable_stg bertipe *Lookup* dengan kode seperti pada Gambar 3.44 bergantung pada keberhasilan *UpsertTable*, memastikan proses *upsert* data telah selesai. Fungsi utamanya adalah mengosongkan kembali tabel *staging* yang bernama stg_odoo_bqtreservation di hospitalitydwh. *Query* yang dieksekusi adalah *SELECT truncate_table('stg_odoo_bqtreservation')*, yang memanggil *function truncate_table* dengan *query* seperti pada Gambar 3.28 untuk melakukan operasi *TRUNCATE TABLE*. Dengan demikian, *activity* ini membersihkan tabel *staging* setelah data berhasil diproses agar siap untuk siklus ETL berikutnya. Langkah ini termasuk dalam tahap *load* (*post-load cleanup*) yang berfungsi membersihkan tabel *staging* setelah proses *upsert* selesai untuk memastikan kesiapan *staging* untuk siklus ETL berikutnya tanpa memengaruhi data utama.

i. *Activity UpdateLastSyncTime*

Activity UpdateLastSyncTime bertipe *Lookup* dengan kode seperti pada Gambar 3.45 adalah langkah terakhir dalam *pipeline* yang bergantung pada keberhasilan *FinalTruncateTable*_stg. Tujuannya adalah untuk memperbarui catatan waktu sinkronisasi terakhir untuk data reservasi *banquet* di dw_synctablelog hospitalitydwh. *Query* yang dieksekusi, *SELECT update_synctablelog('odoo_bqtreservation', '@{variables('SyncDate')}');*, memanggil *function* PostgreSQL *update_synctablelog()* dengan nama tabel odoo_bqtreservation dan waktu mulai *pipeline* (*SyncDate*). *Function* dengan *query* seperti pada Gambar 3.31 ini akan memperbarui kolom *lastupdate* di dw_synctablelog untuk odoo_bqtreservation jika sudah ada, atau menambahkan entri baru jika belum. Hal tersebut memastikan catatan *timestamp* pembaruan data reservasi *banquet* selalu terkini. Langkah ini termasuk dalam tahap *post-load* (*administrative step*) yang berfungsi mencatat waktu terakhir proses ETL sebagai referensi untuk sinkronisasi data selanjutnya.

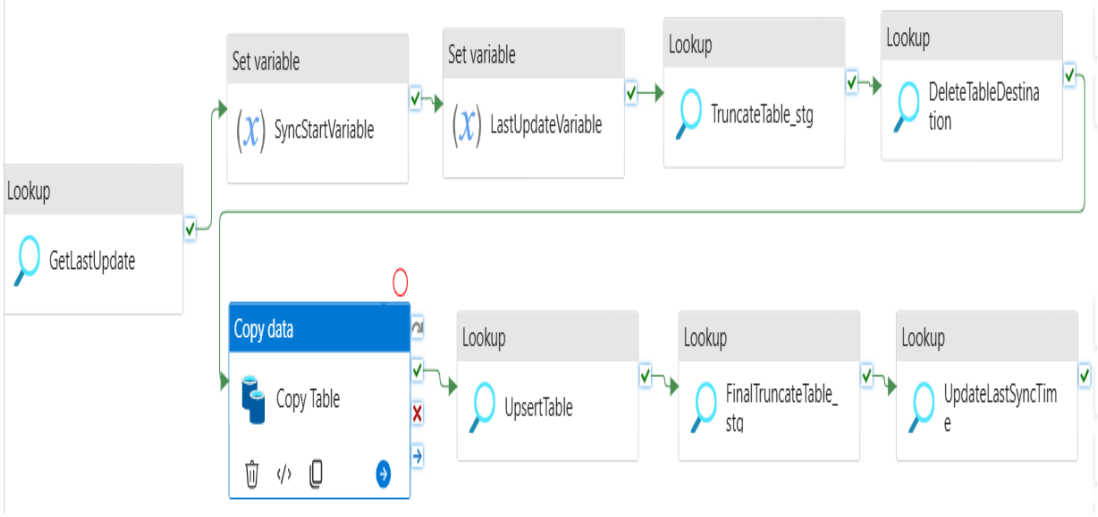
Activity name UpdateLastSyncTime

Copy to clipboard

```
{
    "name": "UpdateLastSyncTime",
    "type": "Lookup",
    "dependsOn": [
        {
            "activity": "FinalTruncateTable_stg",
            "dependencyConditions": [
                "Succeeded"
            ]
        }
    ],
    "policy": {
        "timeout": "0.00:30:00",
        "retry": 5,
        "retryIntervalInSeconds": 30,
        "secureOutput": false,
        "secureInput": false
    },
    "userProperties": [],
    "typeProperties": {
        "source": {
            "type": "AzurePostgreSqlSource",
            "query": {
                "value": "SELECT update_synctablelog('odoo_bqtreservation', '@{variables('SyncDate')}');",
                "type": "Expression"
            },
            "partitionOption": "None"
        },
        "dataset": {
            "referenceName": "Hospitality_DW_SyncTableLog",
            "type": "DatasetReference"
        },
        "firstRowOnly": false
    }
}
```

Gambar 3.45 *Code Activity UpdateLastSyncTime* dalam *Pipeline* BanquetReservation

3.2.5.4. *Pipeline* Hospitality_DW_Odoo_BanquetReservationBill



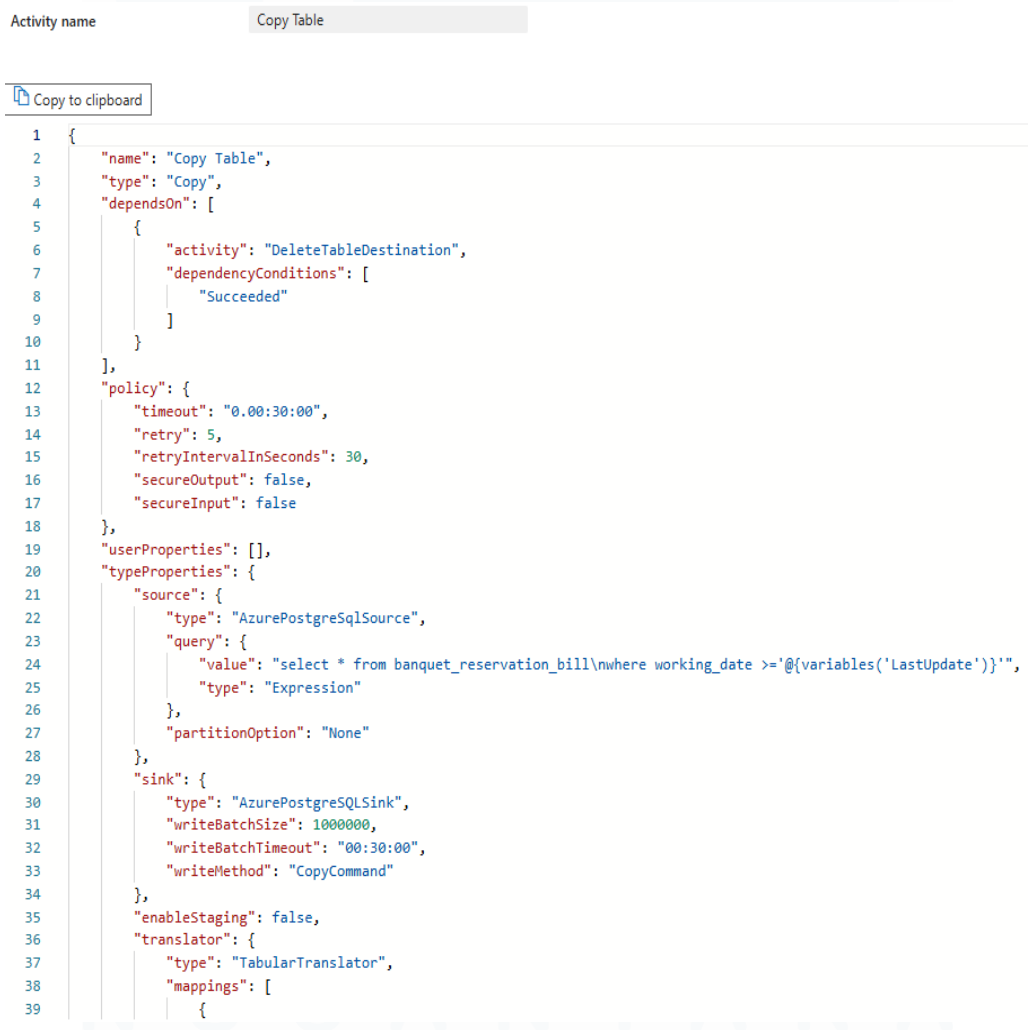
Gambar 3.46 Alur *Pipeline* Hospitality_DW_Odoo_BanquetReservationBill

Alur *pipeline* Hospitality_DW_Odoo_BanquetReservationBill seperti pada Gambar 3.46 dirancang untuk menyalin dan mentransformasikan data perincian tagihan finansial dari setiap reservasi *banquet* dari *database source* odoo ke *data warehouse destination* hospitalitydwh dengan pendekatan inkremental (*upsert*), yang memprioritaskan pembaruan data yang sudah ada atau penambahan data baru. Tahap pertama dimulai dengan *Lookup* untuk mengambil *timestamp* dari log sinkronisasi terakhir pada tabel dw_synctablelog, yang digunakan untuk menyaring data baru atau yang berubah sejak sinkronisasi terakhir. Pada tahap kedua, *pipeline* mencatat waktu pemicu dengan *Set variable SyncStartVariable*, diikuti tahap ketiga dengan *Set variable LastUpdateVariable*, yang berfungsi sebagai filter data inkremental. Tahap keempat, *Lookup TruncateTable_stg* mengosongkan tabel *staging* agar data baru dapat dimuat tanpa mengganggu tabel utama, diikuti tahap kelima *Lookup DeleteTableDestination*, yang menghapus data lama dari tabel utama BanquetReservationBill. Pada tahap keenam, *Copy Table* mentransfer data tagihan yang terfilter ke tabel *staging*. Tahap ketujuh, *Lookup UpsertTable* memastikan data baru atau yang diperbarui dimasukkan ke dalam *data warehouse* hospitalitydwh. Setelah proses *upsert*, tahap kedelapan *FinalTruncateTable_stg*, mengosongkan kembali tabel *staging*, dan *pipeline* diakhiri dengan tahap kesembilan *UpdateLastSyncTime*, yang memperbarui *timestamp* sinkronisasi terakhir pada dw_synctablelog, memastikan data tagihan reservasi *banquet* tetap akurat.

Proses alur *pipeline* Odoo_BanquetReservationBill mirip dengan Odoo_BanquetReservation, namun berbeda pada *query*, tabel, *activity Copy Table*, dan *activity UpsertTable*. Secara umum, *activity GetLastUpdate*, *SyncStartVariable*, *LastUpdateVariable*, *TruncateTable_stg*, *DeleteTableDestination*, *FinalTruncateTable_stg*, dan *UpdateLastSyncTime* memiliki penjelasan serupa dengan *pipeline* BanquetReservation, hanya berbeda pada nama tabel yang disalin. *Activity GetLastUpdate* digunakan untuk mengambil *timestamp* pembaruan terakhir dari tabel dw_synctablelog yang spesifik untuk odoo_bqtreservationbill, guna menentukan data

inkremental yang perlu disalin. *Activity TruncateTable_stg* membersihkan tabel stg_odoo_bqtreservationbill di hospitalitydwh sebelum data baru disalin, sementara *activity DeleteTableDestination* menghapus data lama dari tabel odoo_bqtreservationbill di hospitalitydwh berdasarkan *timestamp LastUpdate* dari *GetLastUpdate*. *Activity UpdateLastSyncTime* memperbarui *timestamp* sinkronisasi di dw_synctablelog tabel odoo_bqtreservationbill. Perbedaan utamanya terletak pada *activity Copy Table* dan *UpsertTable*, dimana masing-masing *pipeline* memetakan dan menyalin data sesuai dengan tabel yang dituju. Berikut penjelasan dari *activity Copy Table* dan *UpsertTable* pada *pipeline* Odoo_BanquetReservationBill:

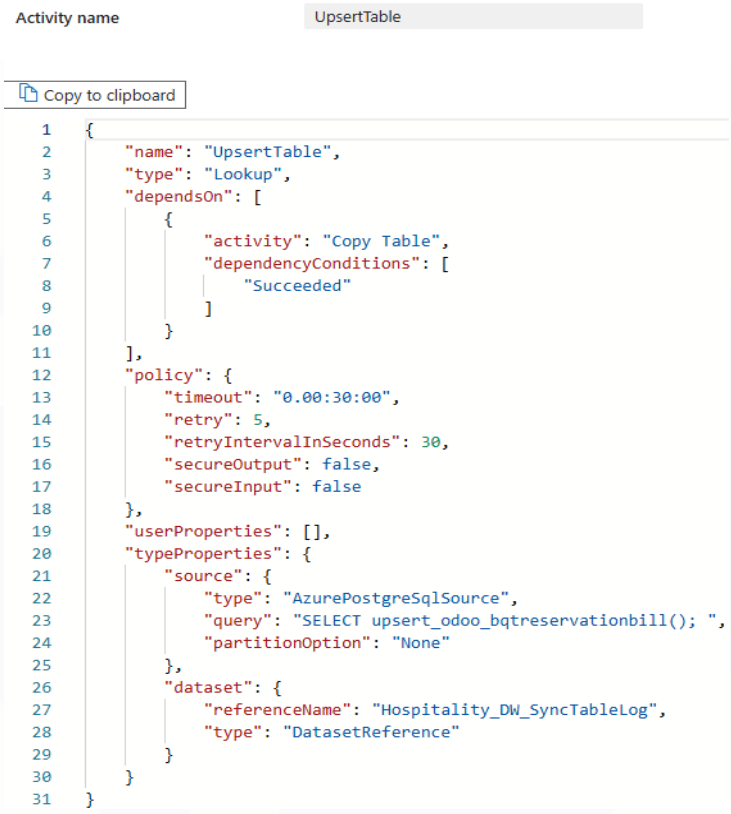
a. *Activity Copy Table*



Gambar 3.47 *Code Activity Copy Table* dalam *Pipeline* BanquetReservationBill

Activity Copy Table dengan kode seperti pada Gambar 3.47 adalah inti dari proses ETL yang berfungsi untuk memindahkan data tagihan reservasi *banquet* yang terbaru atau yang diperbarui dari *database* odoo ke tabel *staging* di hospitalitydwh. *Activity* ini bergantung pada keberhasilan *DeleteTableDestination*, memastikan tabel utama telah dibersihkan dari data lama yang akan diganti. *Query select * from banquet_reservation_bill where working_date >='@{variables('LastUpdate')}'* secara spesifik memilih baris data yang *working_date* nya lebih baru dari *LastUpdate* terakhir, memfokuskan pada transfer data inkremental. Data yang disaring ini kemudian disalin ke *dataset* stg_odoo_bqtreservationbill dengan penyesuaian tipe data dan pemetaan kolom terjadi secara otomatis. Langkah ini mencakup proses *extract* dengan filter berdasarkan *LastUpdateVariable*, *transform* ringan berupa pemetaan dan konversi tipe data, serta *load* data ke tabel *staging* secara bersamaan dalam satu *activity*.

b. *Activity UpsertTable*



Gambar 3.48 *Code Activity UpsertTable* dalam *Pipeline* BanquetReservation

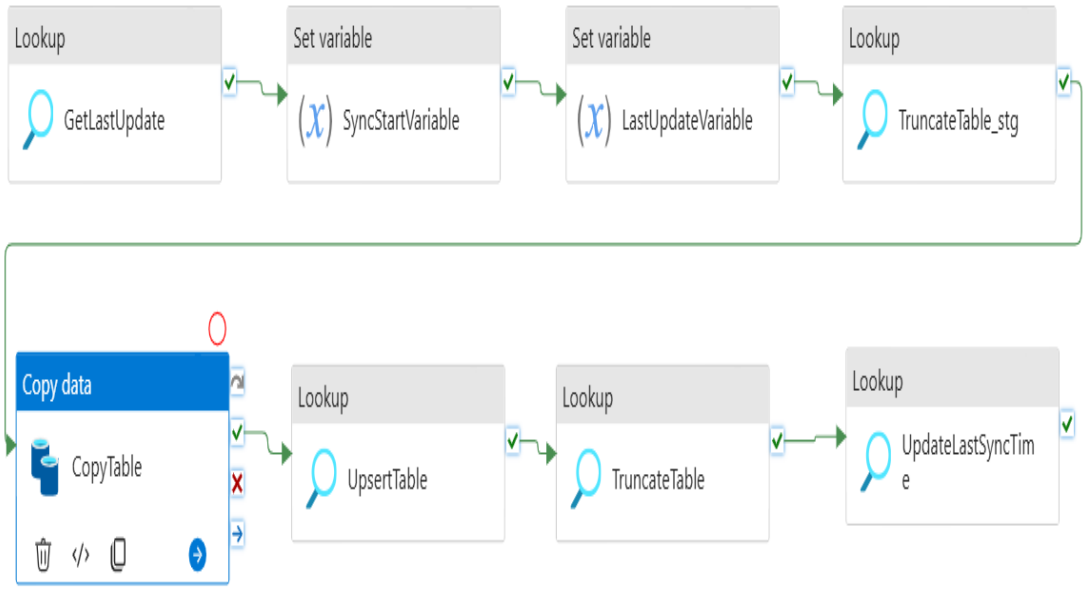
Activity UpsertTable bertipe *Lookup* dengan kode seperti pada Gambar 3.48 bergantung pada keberhasilan *Copy Table*, memastikan data tagihan reservasi *banquet* inkremental sudah tersedia di tabel *staging*. Tujuan utamanya adalah untuk melakukan operasi *upsert* pada data tagihan reservasi *banquet* ke tabel odoo_bqtreservationbill di hospitalitydwh. *Query* yang dieksekusi adalah *SELECT upsert_odoo_bqtreservationbill();*, yang memanggil *function* PostgreSQL *upsert_odoo_bqtreservationbill()*. *Function* dengan *query* seperti pada Gambar 3.49 ini berisi logika *INSERT ... ON CONFLICT* (rb_id) *DO UPDATE SET ...*, yang berarti data dari tabel *staging* (stg_odoo_bqtreservationbill) akan disisipkan ke tabel utama. Jika rb_id sudah ada, baris yang ada akan diperbarui dengan nilai-nilai terbaru dari data *staging*, memastikan tabel tujuan selalu berisi data tagihan reservasi *banquet* yang paling baru. Langkah ini merupakan proses *load* yang melibatkan operasi *upsert* untuk memasukkan data baru atau

memperbarui data yang sudah ada di tabel utama berdasarkan *primary key* atau ID.

```
CREATE OR REPLACE FUNCTION public.upsert_odoo_bqtreservationbill()
 RETURNS void
 LANGUAGE plpgsql
AS $function$
BEGIN
  INSERT INTO odoo_bqtreservationbill (
    rb_id,
    reservation_id,
    company_id,
    state,
    working_date,
    subtotal_amount_rate,
    subtotal_amount_additional,
    total_balance,
    subtotal_payment,
    total_bill,
    is_charge_room,
    total_food,
    total_beverage,
    total_other,
    is_residential,
    invoice_id,
    temp_payment_type
  )
  SELECT
    rb_id,
    reservation_id,
    company_id,
    state,
    working_date,
    subtotal_amount_rate,
    subtotal_amount_additional,
    total_balance,
    subtotal_payment,
    total_bill,
    is_charge_room,
    total_food,
    total_beverage,
    total_other,
    is_residential,
    invoice_id,
    temp_payment_type
  FROM stg_odoo_bqtreservationbill
  WHERE rb_id IS NOT NULL
  ON CONFLICT (rb_id)
  DO UPDATE SET
    reservation_id = EXCLUDED.reservation_id,
    company_id = EXCLUDED.company_id,
    state = EXCLUDED.state,
    working_date = EXCLUDED.working_date,
    subtotal_amount_rate = EXCLUDED.subtotal_amount_rate,
    subtotal_amount_additional = EXCLUDED.subtotal_amount_additional,
    total_balance = EXCLUDED.total_balance,
    subtotal_payment = EXCLUDED.subtotal_payment,
    total_bill = EXCLUDED.total_bill,
    is_charge_room = EXCLUDED.is_charge_room,
    total_food = EXCLUDED.total_food,
    total_beverage = EXCLUDED.total_beverage,
    total_other = EXCLUDED.total_other,
    is_residential = EXCLUDED.is_residential,
    invoice_id = EXCLUDED.invoice_id,
    temp_payment_type = EXCLUDED.temp_payment_type;
END;
$function$
;
```

Gambar 3.49 *Query Function upsert_odoo_bqtreservationbill()*

3.2.5.5. *Pipeline* Hospitality_DW_Odoo_BanquetReservationEvent



Gambar 3.50 Alur *Pipeline* Hospitality_DW_Odoo_BanquetReservationEvent

Alur *pipeline* Hospitality_DW_Odoo_BanquetReservationEvent seperti pada Gambar 3.50 dirancang untuk menyalin dan mentransformasikan data informasi spesifik setiap *event* atau sesi dalam reservasi *banquet* dari *database source* odoo ke *data warehouse destination* hospitalitydwh menggunakan pendekatan inkremental (*upsert*) agar datanya selalu terkini. Tahap pertama dimulai dengan *activity GetLastUpdate* yang mengambil *timestamp* pembaruan terakhir dari log sinkronisasi. Tahap kedua yakni *SyncStartVariable* yang mencatat waktu pemicu *pipeline*, diikuti oleh tahap ketiga yaitu *LastUpdateVariable* yang menyimpan *timestamp* terakhir untuk menyaring data inkremental. Tahap keempat, *TruncateTable_stg* mengosongkan tabel *staging* di hospitalitydwh untuk memuat data mentah, yang kemudian disalin ke tabel *staging* pada tahap kelima melalui *CopyTable*. Pada tahap keenam, *UpsertTable* melakukan *upsert* data ke tabel odoo_bqtreservationevent untuk memastikan data terbaru menggantikan yang lama. Tahap ketujuh, *TruncateTable* membersihkan data lama dan tahap kedelapan *UpdateLastSyncTime*, memperbarui *timestamp* sinkronisasi di dw_synctablelog untuk menjaga jejak audit yang akurat.

Proses alur *pipeline* Odoo_BanquetReservationEvent mirip dengan Odoo_BanquetReservation, namun terdapat perbedaan pada *query*, tabel, serta *activity Copy Table* dan *UpsertTable*. Secara umum, *activity GetLastUpdate*, *SyncStartVariable*, *LastUpdateVariable*, *TruncateTable_stg*, *TruncateTable*, dan *UpdateLastSyncTime* memiliki penjelasan yang serupa dengan *pipeline* BanquetReservation, hanya berbeda pada nama tabel yang disalin. *Activity GetLastUpdate* digunakan untuk mengambil *timestamp* pembaruan terakhir dari tabel dw_synctablelog yang spesifik untuk odoo_bqtreservationevent, guna menentukan data inkremental yang perlu disalin. *Activity TruncateTable_stg* dan *TruncateTable* mengosongkan tabel stg_odoo_bqtreservationevent di hospitalitydwh sebelum dan sesudah data baru disalin. *Activity UpdateLastSyncTime* memperbarui *timestamp* sinkronisasi di dw_synctablelog tabel odoo_bqtreservationevent. Perbedaan utama terdapat pada activity *DeleteTableDestination*, yang tidak diperlukan

dalam *pipeline* Hospitality_DW_Odoo_BanquetReservationEvent. Selain itu, perbedaan juga terletak pada *Copy Table* dan *UpsertTable*, yang masing-masing *pipeline* memetakan dan menyalin data sesuai dengan tabel yang dituju. Berikut penjelasan lebih lanjut mengenai *activity CopyTable* dan *UpsertTable* dalam *pipeline* Odoo_BanquetReservationEvent:

a. *Activity CopyTable*

Gambar 3.51 *Code Activity CopyTable* dalam *Pipeline* BanquetReservationEvent

Activity CopyTable dengan kode seperti pada Gambar 3.51 adalah inti dari proses ETL yang berfungsi untuk menyalin data *event* reservasi *banquet* yang terbaru atau yang diperbarui dari *database* odoo ke tabel *staging* di hospitalitydwh. *Activity* ini bergantung pada keberhasilan *TruncateTable_stg*, memastikan tabel *staging* sudah bersih dahulu. *Query select * from banquet_reservation_event where write_date>='@{variables('LastUpdate')}'* secara spesifik memilih baris data yang *write_date* nya lebih baru dari *LastUpdate* terakhir. Data yang disaring ini kemudian disalin ke *dataset* odoo_bqtreservationevent (hospitalitydwh), dengan penyesuaian tipe data dan pemetaan kolom otomatis. Langkah ini mencakup proses *extract* dengan filter berdasarkan *LastUpdateVariable*, *transform* berupa pemetaan dan konversi tipe data, serta *load* data ke tabel *staging* secara bersamaan dalam satu *activity*.

b. *Activity UpsertTable*

Activity name: UpsertTable

Copy to clipboard

```
{
    "name": "UpsertTable",
    "type": "Lookup",
    "dependsOn": [
        {
            "activity": "CopyTable",
            "dependencyConditions": [
                "Succeeded"
            ]
        }
    ],
    "policy": {
        "timeout": "0.00:30:00",
        "retry": 5,
        "retryIntervalInSeconds": 30,
        "secureOutput": false,
        "secureInput": false
    },
    "userProperties": [
        {
            "name": "Timeout",
            "value": "600"
        },
        {
            "name": "CommandTimeout",
            "value": "0"
        }
    ],
    "typeProperties": {
        "source": {
            "type": "AzurePostgreSqlSource",
            "query": "SELECT upsert_odoo_bqtreservationevent();",
            "partitionOption": "None"
        },
        "dataset": {
            "referenceName": "Hospitality_DW_SyncTableLog",
            "type": "DatasetReference"
        },
        "firstRowOnly": false
    }
}
```

Gambar 3.52 *Code Activity UpsertTable* dalam *Pipeline* BanquetReservationEvent

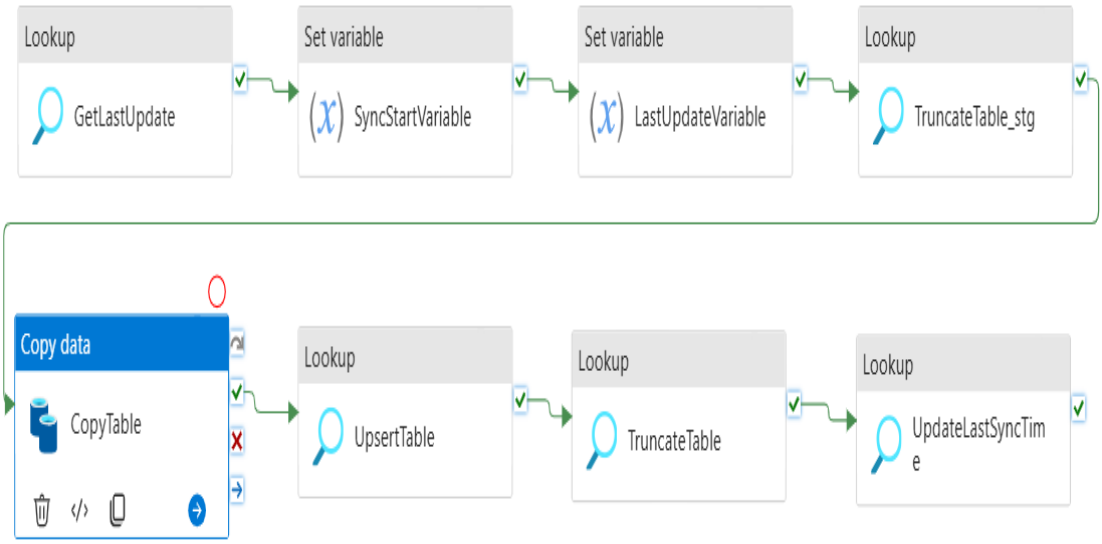
Activity UpsertTable bertipe *Lookup* dengan kode seperti pada Gambar 3.52 bergantung pada keberhasilan *CopyTable*, memastikan data *event* reservasi *banquet* inkremental sudah tersedia di tabel *staging*. Tujuan

utamanya adalah untuk melakukan operasi *upsert* pada data *event* reservasi *banquet* ke tabel odoo_bqtreservationevent di hospitalitydwh. *Query* yang dieksekusi adalah *SELECT upsert_odoo_bqtreservationevent();*, yang memanggil *function* PostgreSQL *upsert_odoo_bqtreservationevent()*. *Function* dengan *query* seperti pada Gambar 3.53 ini berisi logika *INSERT ... ON CONFLICT* (re_id) *DO UPDATE SET ...*, yang berarti data dari tabel stg_odoo_bqtreservationevent akan disisipkan ke tabel utama. Jika re_id sudah ada, baris yang ada akan diperbarui dengan nilai-nilai terbaru dari data *staging* agar tabel tujuan selalu berisi data *event* reservasi *banquet* yang paling baru. Langkah ini merupakan proses *load* yang melibatkan operasi *upsert* untuk memasukkan data baru atau memperbarui data yang sudah ada di tabel utama berdasarkan *primary key* atau ID.

```
CREATE OR REPLACE FUNCTION public.upsert_odoo_bqtreservationevent()
 RETURNS void
 LANGUAGE plpgsql
AS $function$
BEGIN
  INSERT INTO odoo_bqtreservationevent (
    re_id,
    reservation_id,
    company_id,
    function_room_id,
    event_function_id,
    eventdate,
    price_event,
    function_room_amount,
    attendees,
    beo_issued
  )
  SELECT
    re_id,
    reservation_id,
    company_id,
    function_room_id,
    event_function_id,
    eventdate,
    price_event,
    function_room_amount,
    attendees,
    beo_issued
  FROM stg_odoo_bqtreservationevent
  WHERE company_id IS NOT NULL
    AND eventdate IS NOT NULL
    AND reservation_id IS NOT NULL
    AND function_room_id IS NOT NULL
    AND event_function_id IS NOT NULL
  ON CONFLICT (re_id)
  DO UPDATE SET
    reservation_id = EXCLUDED.reservation_id,
    company_id = EXCLUDED.company_id,
    function_room_id = EXCLUDED.function_room_id,
    event_function_id = EXCLUDED.event_function_id,
    eventdate = EXCLUDED.eventdate,
    price_event = EXCLUDED.price_event,
    function_room_amount = EXCLUDED.function_room_a
    attendees = EXCLUDED.attendees,
    beo_issued = EXCLUDED.beo_issued;
END;
$function$
;
```

Gambar 3.53 *Query Function upsert_odoo_bqtreservationbill()*

3.2.5.6. *Pipeline* Odoo_BanquetReservationEventAdditionalResources



Gambar 3.54 Alur *Pipeline* Odoo_BanquetReservationEventAdditionalResources

Alur *pipeline* Odoo_BanquetReservationEventAdditionalResources seperti pada Gambar 3.54 dirancang untuk menyalin dan mentransformasikan data perincian biaya dan kuantitas item tambahan seperti makanan dan minuman yang digunakan dalam setiap *event* reservasi *banquet* dari *database source* odoo ke *data warehouse destination* hospitalitydwh menggunakan pendekatan inkremental (*upsert*) agar datanya selalu terkini. Tahap pertama dimulai dengan *activity GetLastUpdate* yang mengambil *timestamp* pembaruan terakhir dari log sinkronisasi. Tahap kedua yakni *SyncStartVariable* yang mencatat waktu pemicu *pipeline*, diikuti oleh tahap ketiga yaitu *LastUpdateVariable* yang menyimpan *timestamp* terakhir untuk menyaring data inkremental. Tahap keempat, *TruncateTable_stg* mengosongkan tabel *staging* di hospitalitydwh untuk memuat data mentah, yang kemudian disalin ke tabel *staging* pada tahap kelima melalui *CopyTable*. Pada tahap keenam, *UpsertTable* melakukan *upsert* data ke tabel odoo_bqtreservationeventadditionalresources untuk memastikan data terbaru menggantikan yang lama. Tahap ketujuh, *TruncateTable* membersihkan data lama dan tahap kedelapan *UpdateLastSyncTime*, memperbarui *timestamp* sinkronisasi di dw_synctablelog untuk menjaga jejak audit yang akurat.

Proses alur *pipeline* BanquetReservationEventAdditionalResources mirip dengan Odoo_BanquetReservation, namun terdapat perbedaan pada *query*, tabel, serta *activity Copy Table* dan *UpsertTable*. Secara umum, *activity GetLastUpdate*, *SyncStartVariable*, *LastUpdateVariable*, *TruncateTable_stg*, *TruncateTable*, dan *UpdateLastSyncTime* memiliki penjelasan yang serupa dengan *pipeline* BanquetReservation, hanya berbeda pada nama tabel yang disalin. *Activity GetLastUpdate* digunakan untuk mengambil *timestamp* pembaruan terakhir dari tabel dw_synctablelog yang spesifik untuk odoo_bqtreservationeventadditionalresources, guna menentukan data inkremental yang perlu disalin. *Activity TruncateTable_stg* dan *TruncateTable* mengosongkan tabel stg_odoo_bqtreservationeventadditionalresources di hospitalitydwh sebelum dan sesudah data baru disalin. *Activity UpdateLastSyncTime* memperbarui *timestamp* sinkronisasi di dw_synctablelog tabel odoo_bqtreservationeventadditionalresources. Perbedaan utama terdapat pada *activity DeleteTableDestination*, yang tidak diperlukan dalam *pipeline* Odoo_BanquetReservationEventAdditionalResources. Selain itu, perbedaan juga terletak pada *CopyTable* dan *UpsertTable*, yang masing-masing *pipeline* memetakan dan menyalin data sesuai dengan tabel yang dituju. Berikut penjelasan lebih lanjut mengenai *activity CopyTable* dan *UpsertTable* dalam *pipeline* Odoo_BanquetReservationEventAdditionalResources:

a. *Activity CopyTable*

Activity CopyTable dengan kode seperti pada Gambar 3.55 adalah langkah utama untuk menyalin data sumber daya tambahan dari *event* reservasi *banquet* yang terbaru atau yang diperbarui dari *database* odoo ke tabel *staging* di hospitalitydwh. *Activity* ini bergantung pada keberhasilan *TruncateTable_stg*, memastikan tabel *staging* sudah bersih sebelum penyalinan. *Query select * from banquet_reservation_event_additional_resources where write_date>='@{variables('LastUpdate')}'* secara spesifik memilih baris data dengan *write_date* lebih baru dari *LastUpdate* terakhir, fokus pada

transfer data inkremental. Data ini kemudian disalin ke *dataset* odoo_bqtreservationeventadditionalresources dengan penyesuaian tipe data dan pemetaan kolom secara otomatis. Langkah ini mencakup proses *extract* dengan filter berdasarkan *LastUpdateVariable*, *transform* ringan berupa pemetaan dan konversi tipe data, serta *load* data ke tabel *staging* secara bersamaan dalam satu *activity*.

Activity name CopyTable

Copy to clipboard


```
{
    "name": "CopyTable",
    "type": "Copy",
    "dependsOn": [
        {
            "activity": "TruncateTable_stg",
            "dependencyConditions": [
                "Succeeded"
            ]
        }
    ],
    "policy": {
        "timeout": "0.00:30:00",
        "retry": 5,
        "retryIntervalInSeconds": 30,
        "secureOutput": false,
        "secureInput": false
    },
    "userProperties": [],
    "typeProperties": {
        "source": {
            "type": "AzurePostgreSqlSource",
            "query": {
                "value": "select * from banquet_reservation_event_additional_resources\nwhere write_date>='@{variables('LastUpdate')}'",
                "type": "Expression"
            },
            "partitionOption": "None"
        },
        "sink": {
            "type": "AzurePostgreSQLSink",
            "writeBatchSize": 1000000,
            "writeBatchTimeout": "00:30:00",
            "writeMethod": "CopyCommand"
        },
        "enableStaging": false,
        "translator": {
            "type": "TabularTranslator",
            "mappings": [
                {
```

Gambar 3.55 *Code Activity CopyTable* dalam BanquetReservationEventAdditionalResources

b. *Activity UpsertTable*

Activity UpsertTable bertipe *Lookup* dengan kode seperti pada Gambar 3.56 bergantung pada keberhasilan *CopyTable*, memastikan data sumber daya tambahan *event* reservasi *banquet* inkremental sudah tersedia di tabel *staging*. Tujuan utamanya adalah untuk melakukan operasi *upsert* pada data sumber daya tambahan ke tabel odoo_bqtreservationeventadditionalresources di hospitalitydwh. *Query* yang dieksekusi adalah *SELECT upsert_odoo_bqtreservationeventadditionalresources();*, yang memanggil *function* PostgreSQL *upsert_odoo_bqtreservationeventadditionalresources()*. *Function* dengan *query* seperti pada Gambar 3.57 ini berisi logika *INSERT ... ON CONFLICT*

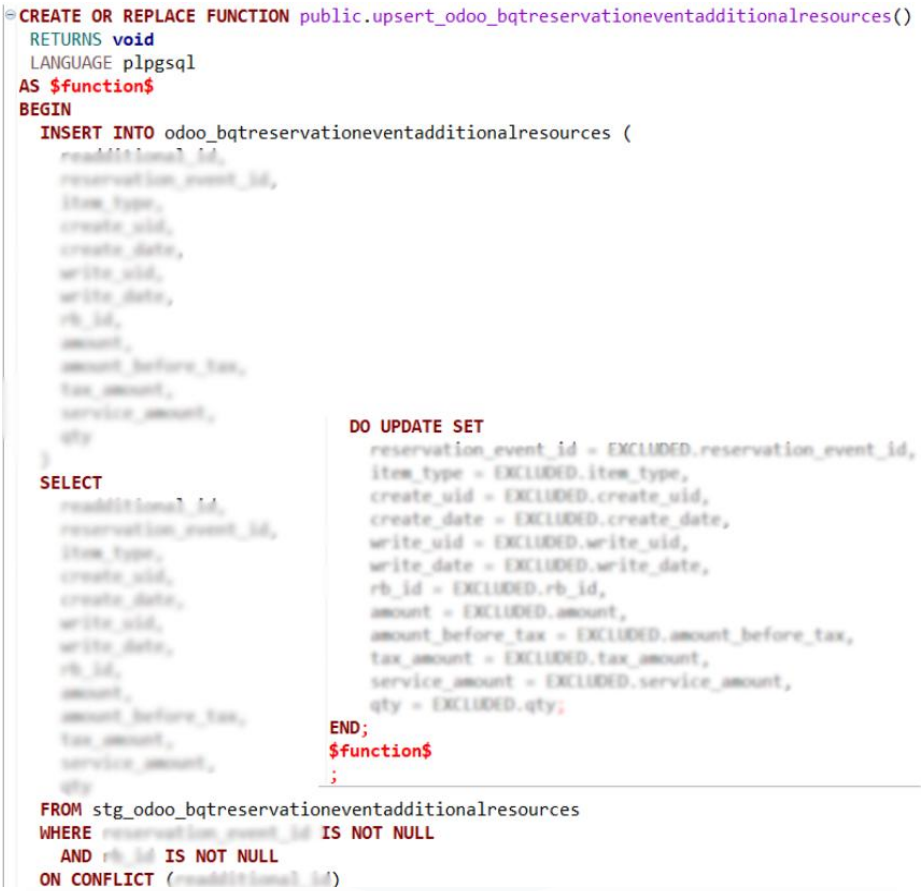
(readditional_id) *DO UPDATE SET* ..., yang berarti data dari tabel stg_odoo_bqtreservationeventadditionalresources akan disisipkan ke tabel utama. Jika readditional_id sudah ada, baris yang ada akan diperbarui dengan nilai-nilai terbaru dari data *staging*, memastikan tabel tujuan selalu berisi data sumber daya tambahan *event* reservasi *banquet* yang paling baru. Langkah ini merupakan proses *load* yang melibatkan operasi *upsert* untuk memasukkan data baru atau memperbarui data yang sudah ada di tabel utama berdasarkan *primary key* atau ID.

Activity name UpsertTable

Copy to clipboard

```
{
    "name": "UpsertTable",
    "type": "Lookup",
    "dependsOn": [
        {
            "activity": "CopyTable",
            "dependencyConditions": [
                "Succeeded"
            ]
        }
    ],
    "policy": {
        "timeout": "0.00:30:00",
        "retry": 5,
        "retryIntervalInSeconds": 30,
        "secureOutput": false,
        "secureInput": false
    },
    "userProperties": [
        {
            "name": "Timeout",
            "value": "600"
        },
        {
            "name": "CommandTimeout",
            "value": "0"
        }
    ],
    "typeProperties": {
        "source": {
            "type": "AzurePostgreSqlSource",
            "query": "SELECT upsert_odoo_bqtreservationeventadditionalresources();",
            "partitionOption": "None"
        },
        "dataset": {
            "referenceName": "Hospitality_DW_SyncTableLog",
            "type": "DatasetReference"
        },
        "firstRowOnly": false
```

Gambar 3.56 *Code Activity UpsertTable* dalam BanquetReservationEventAdditionalResources



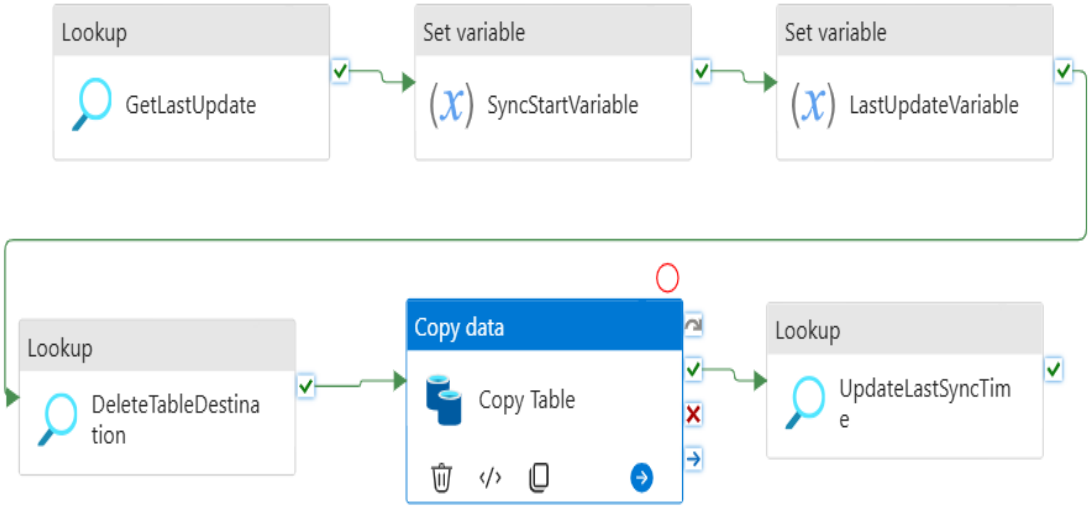
```
CREATE OR REPLACE FUNCTION public.upsert_odoo_bqtreservationeventadditionalresources()
 RETURNS void
 LANGUAGE plpgsql
AS $function$
BEGIN
  INSERT INTO odoo_bqtreservationeventadditionalresources (
    readditional_id,
    reservation_event_id,
    item_type,
    create_uid,
    create_date,
    write_uid,
    write_date,
    rb_id,
    amount,
    amount_before_tax,
    tax_amount,
    service_amount,
    qty
  )
  SELECT
    readditional_id,
    reservation_event_id,
    item_type,
    create_uid,
    create_date,
    write_uid,
    write_date,
    rb_id,
    amount,
    amount_before_tax,
    tax_amount,
    service_amount,
    qty
  FROM stg_odoo_bqtreservationeventadditionalresources
  WHERE reservation_event_id IS NOT NULL
    AND rb_id IS NOT NULL
  ON CONFLICT (readditional_id)
  DO UPDATE SET
    reservation_event_id = EXCLUDED.reservation_event_id,
    item_type = EXCLUDED.item_type,
    create_uid = EXCLUDED.create_uid,
    create_date = EXCLUDED.create_date,
    write_uid = EXCLUDED.write_uid,
    write_date = EXCLUDED.write_date,
    rb_id = EXCLUDED.rb_id,
    amount = EXCLUDED.amount,
    amount_before_tax = EXCLUDED.amount_before_tax,
    tax_amount = EXCLUDED.tax_amount,
    service_amount = EXCLUDED.service_amount,
    qty = EXCLUDED.qty;
END;
$function$
;
```

Gambar 3.57 *Query Function upsert_odoo_bqtreservationeventadditionalresources()*

3.2.5.7. *Pipeline* Hospitality_DW_Odoo_BanquetReservationRate

Alur *pipeline* Hospitality_DW_Odoo_BanquetReservationRate seperti pada Gambar 3.58 dirancang untuk menyalin dan mentransformasikan data perincian biaya paket dan jumlah kehadiran reservasi *banquet* dari *database source* odoo ke *data warehouse destination* hospitalitydwh. Alur *pipeline* pertama dimulai dengan *activity GetLastUpdate*, yang mengambil *timestamp* terakhir dari log sinkronisasi untuk mengidentifikasi data baru atau yang berubah. Tahap kedua yakni *SyncStartVariable* yang mencatat waktu pemicu sinkronisasi, diikuti oleh tahap ketiga *LastUpdateVariable* yang menyimpan *timestamp* terakhir untuk menyaring data inkremental. Pada tahap keempat, *DeleteTableDestination* digunakan untuk menghapus data lama dari tabel odoo_bqtreservationrate di hospitalitydwh, memastikan hanya data terbaru yang disalin. Pada tahap kelima, *Copy Table* akan menyalin data tarif reservasi

banquet yang terbaru atau telah dimodifikasi ke tabel utama. Tahap keenam *UpdateLastSyncTime*, yang memperbarui *timestamp* sinkronisasi terakhir di dw_synctablelog untuk menjaga catatan pembaruan yang akurat.



Gambar 3.58 Alur *Pipeline* Hospitality_DW_Odoo_BanquetReservationRate

Proses alur *pipeline* Hospitality_DW_Odoo_BanquetReservationRate mirip dengan Hospitality_DW_Odoo_BanquetReservation, namun berbeda pada *query*, tabel, dan *activity Copy Table*. Secara umum, *activity GetLastUpdate*, *SyncStartVariable*, *LastUpdateVariable*, *DeleteTableDestination*, dan *UpdateLastSyncTime* memiliki penjelasan serupa dengan *pipeline* BanquetReservation, hanya berbeda pada nama tabel yang disalin. *GetLastUpdate* mengambil *timestamp* pembaruan terakhir dari tabel dw_synctablelog untuk odoo_bqtreservationrate, yang digunakan untuk menentukan data inkremental yang perlu disalin. *DeleteTableDestination* dengan *function query* seperti pada Gambar 3.59, digunakan untuk menghapus data lama dari tabel odoo_bqtreservationrate di hospitalitydwh berdasarkan *timestamp LastUpdate* yang diperoleh dari *GetLastUpdate*. *UpdateLastSyncTime* memperbarui *timestamp* sinkronisasi di dw_synctablelog untuk tabel odoo_bqtreservationrate. Perbedaan utamanya terletak pada *Copy Table*, yang memetakan dan menyalin data sesuai dengan tabel odoo_bqtreservationrate.

```
CREATE OR REPLACE FUNCTION public.delete_table_date(p_table_name text, p_date date)
 RETURNS void
 LANGUAGE plpgsql
AS $function$
BEGIN
    EXECUTE format('DELETE FROM %I WHERE date >= %L', p_table_name, p_date);
END;
$function$
;
```

Gambar 3.59 *Query Function delete_table_date()*

a. *Activity Copy Table*

Activity Copy Table dengan kode seperti pada Gambar 3.60 adalah langkah utama untuk menyalin data tarif reservasi *banquet* yang terbaru atau yang diperbarui dari *database* odoo ke *data warehouse* hospitalitydwh. *Activity* ini bergantung pada keberhasilan *DeleteTableDestination*, memastikan tabel tujuan telah dikosongkan dari data lama yang akan diganti. *Query select * from banquet_reservation_rate where date >='@{variables('LastUpdate')}'* secara spesifik memilih baris data dengan *date* nya lebih baru dari *LastUpdate* terakhir. Data yang disaring ini kemudian disalin ke *dataset* odoo_bqtreservationrate, dengan penyesuaian tipe data dan pemetaan kolom terjadi secara otomatis. Langkah ini mencakup proses *extract* dengan filter berdasarkan *LastUpdateVariable*, *transform* ringan berupa pemetaan dan konversi tipe data, serta *load* data ke tabel *staging* secara bersamaan dalam satu *activity*.

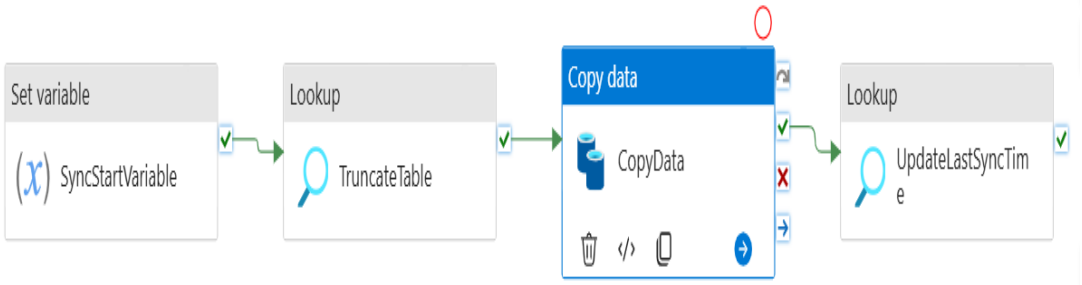
Activity name: Copy Table

Copy to clipboard

```
{
    "name": "Copy Table",
    "type": "Copy",
    "dependsOn": [
        {
            "activity": "DeleteTableDestination",
            "dependencyConditions": [
                "Succeeded"
            ]
        }
    ],
    "policy": {
        "timeout": "0.00:30:00",
        "retry": 5,
        "retryIntervalInSeconds": 30,
        "secureOutput": false,
        "secureInput": false
    },
```

Gambar 3.60 *Code Activity Copy Table* dalam *Pipeline* Odoo_BanquetReservationRate

3.2.5.8. *Pipeline* Hospitality_DW_Odoo_CatalogEventType



Gambar 3.61 Alur *Pipeline* Hospitality_DW_Odoo_CatalogEventType

Alur *pipeline* Hospitality_DW_Odoo_CatalogEventType seperti pada Gambar 3.61 dirancang untuk menyalin dan mentransformasikan data jenis-jenis acara atau *event* yang dapat diselenggarakan. Proses ini mentransfer data dari *database* sumber odoo ke *data warehouse* tujuan hospitalitydwh. Tahap pertama dimulai dengan *activity Set variable* bernama *SyncStartVariable*, yang

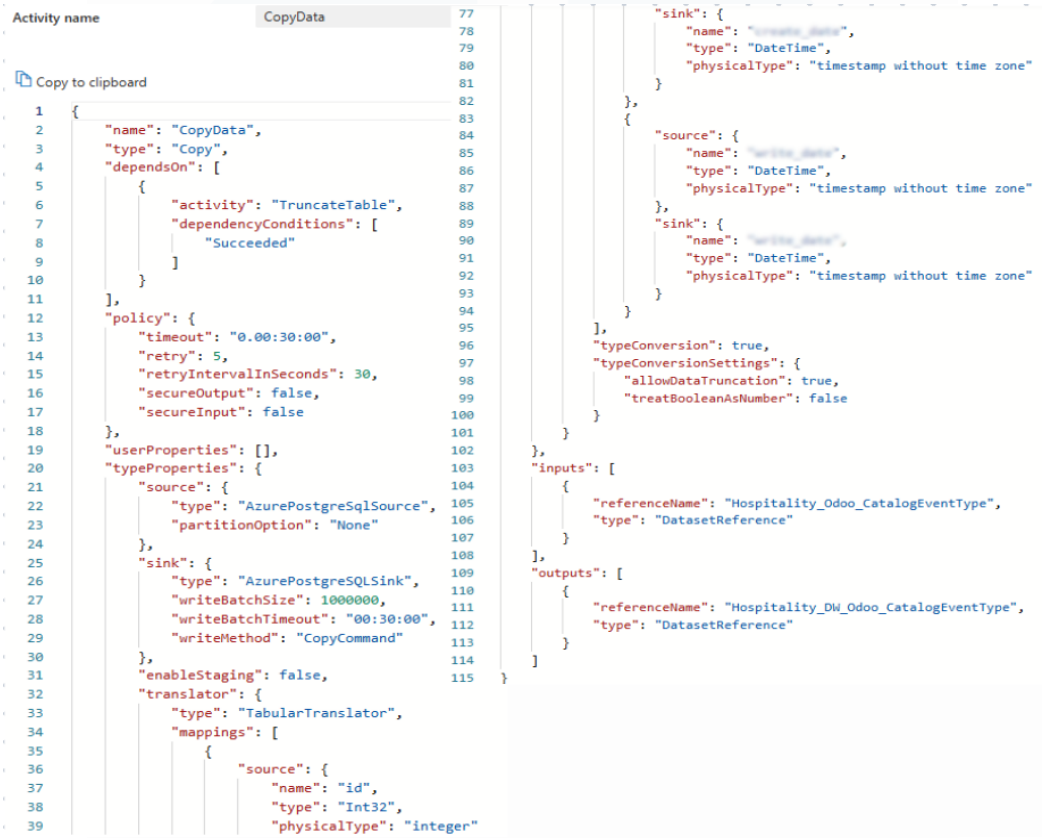
mencatat waktu pemicu sinkronisasi. Tahap kedua dilakukan *TruncateTable* yang mengosongkan tabel odoo_catalogeventtype di hospitalitydwh untuk memastikan hanya data terbaru yang disalin. Pada tahap ketiga, *activity CopyData* akan menyalin data jenis acara dari odoo ke tabel tujuan, menggantikan data yang ada. Tahap keempat adalah *UpdateLastSyncTime*, yang memperbarui *timestamp* sinkronisasi terakhir di dw_synctablelog menggunakan waktu yang tercatat pada *SyncStartVariable*.

Proses alur *pipeline* Hospitality_DW_Odoo_CatalogEventType mirip dengan Odoo_BanquetFunctionRoom, namun berbeda pada *query*, tabel, dan alur *activity CopyData*. Secara umum, *activity SyncStartVariable*, *TruncateTable*, dan *UpdateLastSyncTime* memiliki penjelasan serupa dengan perbedaan pada nama tabel yang disalin. *SyncStartVariable* menggunakan variabel *SyncDate* untuk menandai waktu sinkronisasi, *TruncateTable* digunakan untuk membersihkan data pada tabel odoo_catalogeventtype, dan *UpdateLastSyncTime* memperbarui *timestamp* sinkronisasi di dw_synctablelog berdasarkan tabel odoo_catalogeventtype. Perbedaan utamanya terletak pada *activity CopyData*, dimana setiap *pipeline* memetakan dan menyalin data sesuai kebutuhan tabel yang dituju. Berikut penjelasan dari *activity CopyData* pada *pipeline* Odoo_CatalogEventType:

a. *Activity CopyData*

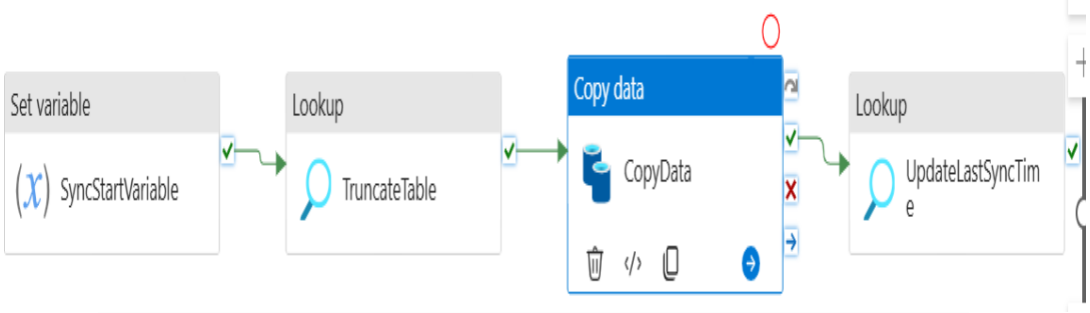
Activity CopyData dengan kode seperti pada Gambar 3.62 adalah langkah utama untuk menyalin data jenis *event* katalog dari *database* odoo ke hospitalitydwh. *Activity* ini bergantung pada keberhasilan *TruncateTable*, memastikan tabel tujuan sudah dikosongkan secara menyeluruh sebelum penyalinan dimulai. Data disalin dari *dataset* catalog_eventtype (odoo) dan dimuat ke *dataset* odoo_catalogeventtype (hospitalitydwh), tanpa menggunakan filter inkremental pada *write_date*. Selama proses ini, ADF secara otomatis melakukan pemetaan dan konversi tipe data antar kolom sumber dan tujuan. *Activity* ini termasuk dalam kategori *extract*, *transform*, *load* sekaligus karena ADF menjalankan

penarikan data dari *source* (*extract*), melakukan konversi dan pemetaan kolom (*transform*), serta menyimpan hasilnya ke dalam *destination table* (*load*) secara terintegrasi dalam satu proses.



Gambar 3.62 *Code Activity CopyData* dalam *Pipeline* Odoo_CatalogEventType

3.2.5.9. *Pipeline* Hospitality_DW_Odoo_HrEmployee



Gambar 3.63 Alur *Pipeline* Hospitality_DW_Odoo_HrEmployee

Alur *pipeline* Hospitality_DW_Odoo_HrEmployee seperti pada Gambar 3.63 dirancang untuk menyalin dan mentransformasikan data detail identitas dan status karyawan yang berasal dari *database* sumber odoo ke *data*

warehouse tujuan hospitalitydwh. Tahap pertama dimulai dengan *activity Set variable* bernama *SyncStartVariable*, yang mencatat waktu pemicu sinkronisasi. Tahap kedua dilakukan *TruncateTable* yang mengosongkan tabel odoo_hremployee di hospitalitydwh untuk memastikan hanya data terbaru yang disalin. Pada tahap ketiga, *activity CopyData* akan menyalin data detail identitas dan status karyawan dari odoo ke tabel tujuan, menggantikan data yang ada. Tahap keempat adalah *UpdateLastSyncTime*, yang memperbarui *timestamp* sinkronisasi terakhir di dw_synctablelog menggunakan waktu yang tercatat pada *SyncStartVariable*.

Proses alur *pipeline* Hospitality_DW_Odoo_HrEmployee mirip dengan Odoo_BanquetFunctionRoom, namun berbeda pada *query*, tabel, dan alur *activity CopyData*. Secara umum, *activity SyncStartVariable*, *TruncateTable*, dan *UpdateLastSyncTime* memiliki penjelasan serupa dengan perbedaan pada nama tabel yang disalin. *SyncStartVariable* menggunakan variabel *SyncDate* untuk menandai waktu sinkronisasi, *TruncateTable* digunakan untuk membersihkan data pada tabel odoo_hremployee, dan *UpdateLastSyncTime* memperbarui *timestamp* sinkronisasi di dw_synctablelog berdasarkan tabel odoo_hremployee. Perbedaan utamanya terletak pada *activity CopyData*, dimana setiap *pipeline* memetakan dan menyalin data sesuai kebutuhan tabel yang dituju. Berikut penjelasan dari *activity CopyData* pada *pipeline* Hospitality_DW_Odoo_HrEmployee:

a. *Activity CopyData*

Activity CopyData dengan kode seperti pada Gambar 3.64 adalah langkah utama untuk menyalin data karyawan dari *database* odoo ke *data warehouse* hospitalitydwh. *Activity* ini bergantung pada keberhasilan *TruncateTable*, memastikan tabel tujuan sudah dikosongkan secara menyeluruh sebelum penyalinan dimulai. Data disalin dari *dataset* hr_employee (odoo) dan dimuat ke *dataset* odoo_hremployee (hospitalitydwh) tanpa menggunakan filter inkremental. Selama proses ini, ADF secara otomatis melakukan pemetaan dan konversi tipe data antar

kolom sumber dan tujuan melalui properti *translator* (termasuk *mappings* dan *typeConversionSettings*). *Activity* ini termasuk dalam kategori *extract*, *transform*, *load* sekaligus karena ADF menjalankan penarikan data dari *source* (*extract*), melakukan konversi dan pemetaan kolom (*transform*), serta menyimpan hasilnya ke dalam *destination table* (*load*) secara terintegrasi dalam satu proses.

Activity name CopyData

Copy to clipboard

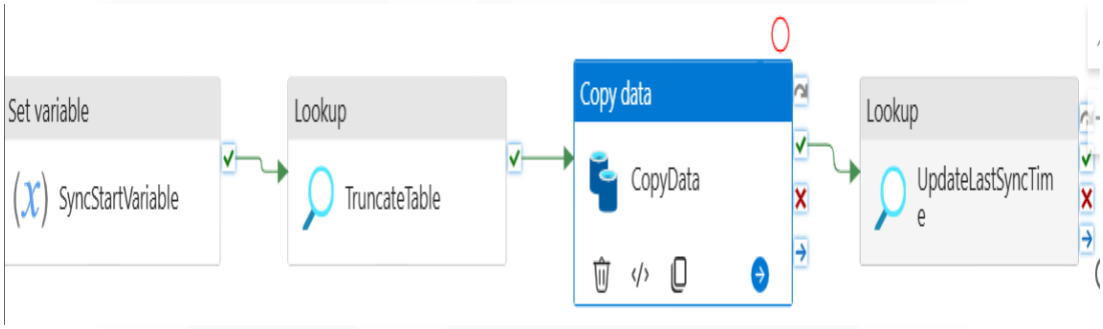

```
{
    "name": "CopyData",
    "type": "Copy",
    "dependsOn": [
        {
            "activity": "TruncateTable",
            "dependencyConditions": [
                "Succeeded"
            ]
        }
    ],
    "policy": {
        "timeout": "0.00:30:00",
        "retry": 5,
        "retryIntervalInSeconds": 30,
        "secureOutput": false,
        "secureInput": false
    },
    "userProperties": [],
    "typeProperties": {
        "source": {
            "type": "AzurePostgreSqlSource",
            "partitionOption": "None"
        },
        "sink": {
            "type": "AzurePostgreSQLSink",
            "writeBatchSize": 1000000,
            "writeBatchTimeout": "00:30:00",
            "writeMethod": "CopyCommand"
        },
        "enableStaging": false,
        "translator": {
            "type": "TabularTranslator",
            "mappings": [
                {
                    "source": {
                        "name": "id",
                        "type": "Int32",
                        "physicalType": "integer"
                    },
                    "sink": {
                        "name": "[redacted]",
                        "type": "String",
                        "physicalType": "character varying"
                    }
                },
                {
                    "source": {
                        "name": "[redacted]",
                        "type": "Boolean",
                        "physicalType": "boolean"
                    },
                    "sink": {
                        "name": "[redacted]",
                        "type": "Boolean",
                        "physicalType": "boolean"
                    }
                }
            ],
            "typeConversion": true,
            "typeConversionSettings": {
                "allowDataTruncation": true,
                "treatBooleanAsNumber": false
            }
        }
    },
    "inputs": [
        {
            "referenceName": "Hospitality_Odoo_HrEmployee",
            "type": "DatasetReference"
        }
    ],
    "outputs": [
        {
            "referenceName": "Hospitality_DW_Odoo_HrEmployee",
            "type": "DatasetReference"
        }
    ]
}
```


Gambar 3.64 *Code Activity CopyData* dalam *Pipeline* Hospitality_DW_Odoo_HrEmployee

3.2.5.10. *Pipeline* Hospitality_DW_Odoo_ResPartner

Alur *pipeline* Hospitality_DW_Odoo_ResPartner seperti pada Gambar 3.65 dirancang untuk menyalin dan mentransformasikan data dari *database* sumber odoo ke *data warehouse* tujuan hospitalitydwh yang berisi informasi tentang mitra atau organisasi yang bekerja sama dengan hotel, termasuk perusahaan dan institusi. Tahap pertama dimulai dengan *activity Set variable* bernama *SyncStartVariable*, yang mencatat waktu pemicu sinkronisasi. Tahap

kedua dilakukan *TruncateTable* yang mengosongkan tabel odoo_respartner di hospitalitydwh untuk memastikan hanya data terbaru yang disalin. Pada tahap ketiga, *activity CopyData* akan menyalin data tentang mitra atau organisasi yang bekerja sama dengan hotel dari odoo ke tabel tujuan, menggantikan data yang ada. Tahap keempat adalah *UpdateLastSyncTime*, yang memperbarui *timestamp* sinkronisasi terakhir di dw_synctablelog menggunakan waktu yang tercatat pada *SyncStartVariable*.



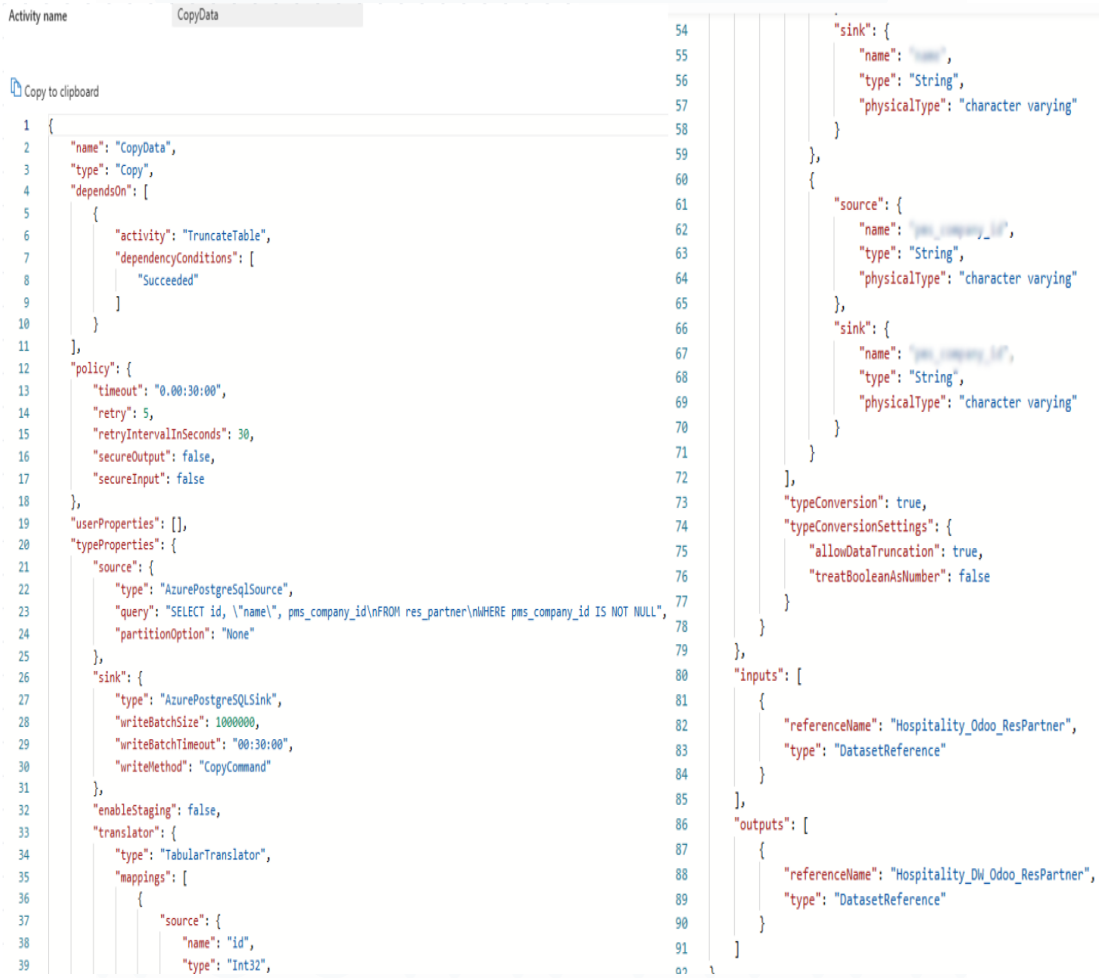
Gambar 3.65 Alur *Pipeline* Hospitality_DW_Odoo_ResPartner

Proses alur *pipeline* Hospitality_DW_Odoo_ResPartner mirip dengan Odoo_BanquetFunctionRoom, namun berbeda pada *query*, tabel, dan alur *activity CopyData*. Secara umum, *activity SyncStartVariable*, *TruncateTable*, dan *UpdateLastSyncTime* memiliki penjelasan serupa dengan perbedaan pada nama tabel yang disalin. *SyncStartVariable* menggunakan variabel *SyncDate* untuk menandai waktu sinkronisasi, *TruncateTable* digunakan untuk membersihkan data pada tabel odoo_respartner, dan *UpdateLastSyncTime* memperbarui *timestamp* sinkronisasi di dw_synctablelog berdasarkan tabel odoo_respartner. Perbedaan utamanya terletak pada *activity CopyData*, dimana setiap *pipeline* memetakan dan menyalin data sesuai kebutuhan tabel yang dituju. Berikut penjelasan dari *activity CopyData* pada *pipeline* Hospitality_DW_Odoo_ResPartner:

a. *Activity CopyData*

Activity CopyData dengan kode seperti pada Gambar 3.66 adalah langkah utama untuk menyalin data *partner* (pelanggan/vendor) dari *database* odoo ke hospitalitydwh. *Activity* ini bergantung pada keberhasilan

TruncateTable, memastikan tabel tujuan sudah bersih sebelum penyalinan dimulai. Dalam *pipeline* ini terdapat *query* khusus seperti *SELECT id, "name", pms_company_id FROM res_partner WHERE pms_company_id IS NOT NULL* yang berguna untuk spesifik memilih kolom tertentu dan memfilter data dimana pms_company_id tidak *null*. Data yang disaring ini kemudian disalin ke *dataset* odoo_respartner, dengan penyesuaian tipe data dan pemetaan kolom terjadi secara otomatis melalui properti *translator* (termasuk *mappings* dan *typeConversionSettings*). *Activity* ini termasuk dalam kategori *extract*, *transform*, *load* sekaligus karena ADF menjalankan penarikan data dari *source* (*extract*), melakukan konversi dan pemetaan kolom (*transform*), serta menyimpan hasilnya ke dalam *destination table* (*load*) secara terintegrasi dalam satu proses.



Gambar 3.66 *Code Activity CopyData* dalam *Pipeline* Hospitality_DW_Odoo_ResPartner

3.2.6. ETL *Process* – Alur *Pipeline* Untuk Kebutuhan *Dashboard* CRM

Proses *Extract, Transform, Load* (ETL) sangat penting untuk memastikan integrasi data yang lancar antar sistem. Sistem *Customer Relationship Management* (CRM) *Hospitality* Kompas Gramedia menyajikan berbagai *dashboard* yang berisi informasi tentang interaksi pelanggan. Dalam proyek magang ini, salah satu tugas utamanya adalah memindahkan *dashboard* dari sistem CRM ke dalam sistem *Property Management System* (PMS) agar dapat diakses oleh manajemen hotel. Namun, data dalam sistem CRM tidak dapat langsung digunakan oleh sistem lain tanpa proses ETL yang tepat untuk menyalin, membersihkan, dan menyelaraskan data ke dalam *data warehouse* sebagai sumber visualisasi. Oleh karena itu, menyalin dan mentransformasikan data ke dalam *data warehouse* terpusat (hospitalitydwh) menjadi krusial agar data dapat diproses lebih lanjut dan digunakan dalam sistem lain, seperti *Property Management System* (PMS). Peserta magang bertugas merancang dan membangun alur *pipeline* data untuk menyalin data dari sistem CRM ke hospitalitydwh. *Pipeline* ini tidak hanya menyalin data, tetapi juga memastikan bahwa data yang diproses siap digunakan untuk pembuatan *dashboard* yang dapat diakses melalui PMS hotel. Tabel 3.4 menunjukkan rincian fokus dari ketiga *pipeline* dalam proyek CRM.

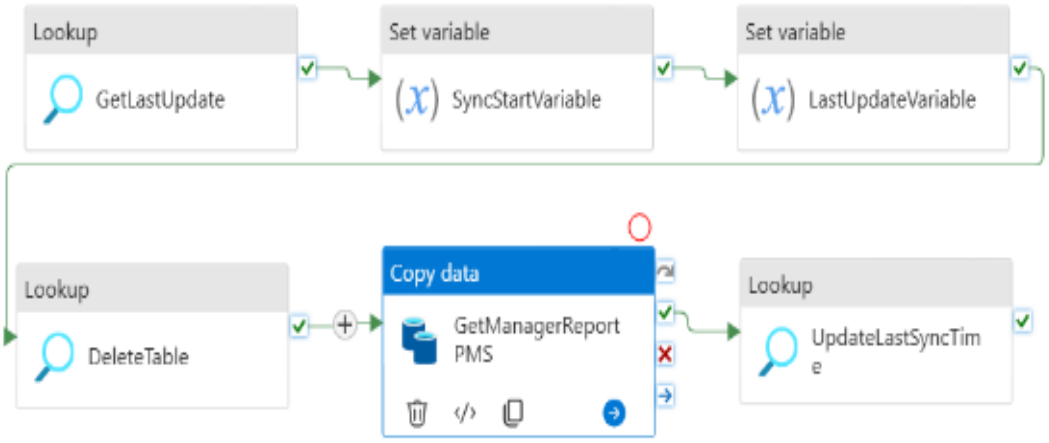
Tabel 3.4 Rincian Fokus *Pipeline* Untuk Kebutuhan *Dashboard* CRM

Nama *Pipeline*	*Database* Sumber	Tabel Sumber	*Database* Tujuan	Tabel Tujuan
ManagerReport	PMS	repManagerReport	hospitalitydwh	pms_repmanagerreport
RoomCountSheet	PMS	repRoomRevenueRoomCountSheet	hospitalitydwh	pms_reproomrevenueroomcountsheet
BudgetMarketSegment	PMS	mstMarketSegmentBudget	hospitalitydwh	pms_mstmarketsegmentbudget

3.2.6.1. *Pipeline* Hospitality_DW_ManagerReport

Alur *pipeline* Hospitality_DW_ManagerReport seperti pada Gambar 3.67 dirancang sebagai komponen utama dalam proses ETL untuk memperbarui data laporan manajerial pada *dashboard* CRM. Tahap pertama

dimulai dengan *activity GetLastUpdate* yang mengambil informasi waktu sinkronisasi terakhir guna memastikan hanya data terbaru yang diproses. Hasilnya digunakan dalam tahap kedua dan ketiga untuk menetapkan dua variabel penting, *SyncStartVariable* dan *LastUpdateVariable*, yang berfungsi sebagai penanda waktu pemrosesan dan acuan validasi. Tahap keempat berupa *DeleteTable* digunakan untuk menghapus isi tabel tujuan guna mencegah duplikasi atau inkonsistensi saat data baru ditambahkan. Tahap kelima, yaitu *GetManagerReportPMS* yang menyalin data dari *database* sumber PMS ke *data warehouse* tujuan hospitalitydwh dengan struktur tabel yang telah disesuaikan. Setelah pemindahan data selesai, tahap keenam yaitu *UpdateLastSyncTime* mencatat waktu sinkronisasi terbaru sebagai referensi untuk proses ETL selanjutnya. Seluruh tahapan berjalan secara linier dan saling bergantung untuk menjaga integritas data.



Gambar 3.67 Alur *Pipeline* Hospitality_DW_ManagerReport

Proses alur *pipeline* Hospitality_DW_ManagerReport mirip dengan Hospitality_DW_Odoo_BanquetReservation, namun berbeda pada *query*, tabel, dan *activity Copy Data*. Secara umum, *activity GetLastUpdate*, *SyncStartVariable*, *LastUpdateVariable*, *DeleteTableDestination*, dan *UpdateLastSyncTime* memiliki penjelasan serupa dengan *pipeline* BanquetReservation, hanya berbeda pada nama tabel yang disalin. *GetLastUpdate* mengambil *timestamp* pembaruan terakhir dari tabel dw_synctablelog untuk pms_repmanagerreport, yang digunakan untuk

menentukan data inkremental yang perlu disalin. *DeleteTableDestination* dengan *function query* seperti pada Gambar 3.68, digunakan untuk menghapus data lama dari tabel pms_repmanagerreport di hospitalitydwh berdasarkan *timestamp LastUpdate* yang diperoleh dari *GetLastUpdate*. *UpdateLastSyncTime* memperbarui *timestamp* sinkronisasi di dw_synctablelog untuk tabel pms_repmanagerreport. Perbedaan utamanya terletak pada *activity GetManagerReportPMS*, yang memetakan dan menyalin data sesuai dengan tabel pms_repmanagerreport.

```
CREATE OR REPLACE FUNCTION public.delete_table_trxdate(p_table_name text, p_date date)
 RETURNS void
 LANGUAGE plpgsql
AS $function$
BEGIN
    EXECUTE format('[illegible]', p_table_name, p_date);
END;
$function$
;
```

Gambar 3.68 *Query Function delete_table_trxdate()*

a. *Activity GetManagerReportPMS*

Activity GetManagerReportPMS bertipe *copy data* dengan kode seperti pada Gambar 3.69 bertugas menyalin data laporan manajer dari *database* sumber (PMS) ke *data warehouse* tujuan (hospitalitydwh). Data sumber diambil dari tabel repManagerReport dengan *query* SQL yang hanya menyertakan data dari hotel-hotel tertentu (selain 999 dan 900) dan data yang tanggal transaksinya (*TrxDate*) lebih baru atau sama dengan waktu pembaruan terakhir yang disimpan dalam variabel *LastUpdate*. Data ini kemudian disalin ke hospitalitydwh melalui fitur *sink* PostgreSQL dengan *batch size* besar dan *write method CopyCommand*. *Mapping* kolom dari *source* ke *sink* digunakan untuk memastikan penamaan dan tipe data yang konsisten. Langkah ini mencakup proses *extract* dengan filter berdasarkan *LastUpdateVariable*, *transform* ringan berupa pemetaan dan konversi tipe data, serta *load* data ke tabel pms_repmanagerreport secara bersamaan dalam satu *activity*.

Activity name: GetManagerReportPMS

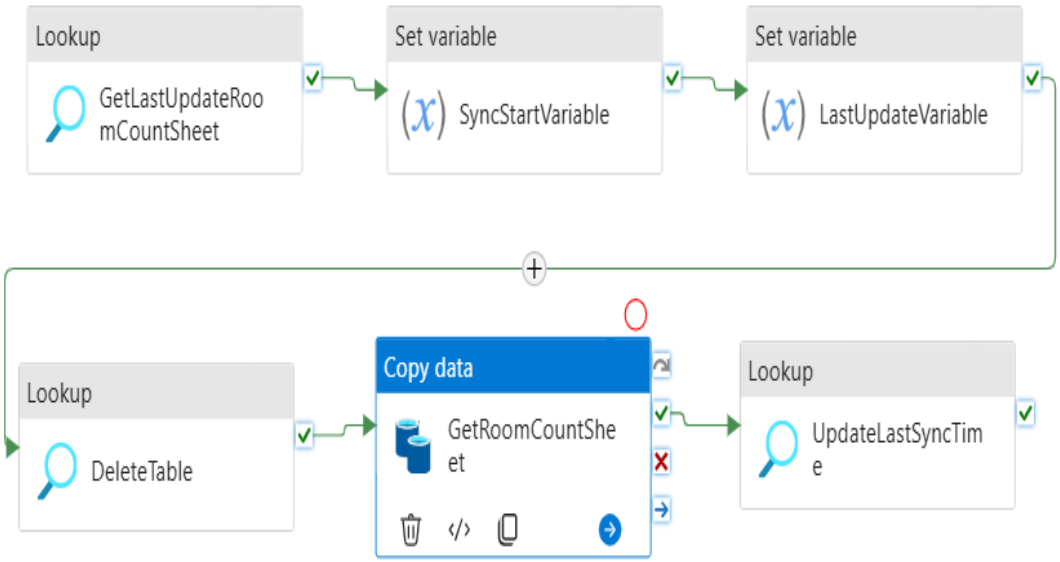
Copy to clipboard

```
{
    "name": "GetManagerReportPMS",
    "type": "Copy",
    "dependsOn": [
        {
            "activity": "DeleteTable",
            "dependencyConditions": [
                "Succeeded"
            ]
        }
    ],
    "policy": {
        "timeout": "0.00:30:00",
        "retry": 5,
        "retryIntervalInSeconds": 30,
        "secureOutput": false,
        "secureInput": false
    },
    "userProperties": [],
    "typeProperties": {
        "source": {
            "type": "AzureSqlSource",
            "sqlReaderQuery": {
                "value": "select * from repManagerReport\nwhere HotelId <> '999' and HotelId <> '900' and TrxDate >= '@{variables('LastUpdate')}'",
                "type": "Expression"
            },
            "queryTimeout": "02:00:00",
            "partitionOption": "None"
        },
        "sink": {
            "type": "AzurePostgreSQLSink",
            "writeBatchSize": 1000000,
            "writeBatchTimeout": "00:30:00",
            "writeMethod": "CopyCommand"
        },
        "enableStaging": false,
        "translator": {
            "type": "TabularTranslator",
            "mappings": [
                {
                    "source": {
                        "name": "HotelId",
                        "type": "Int32",
                        "physicalType": "int"
                    },
                    "sink": {
                        "name": "hotelid",
                        "type": "Int32",
                        "physicalType": "integer"
                    }
                },
                {
                    "source": {
                        "name": "[redacted]",
                        "type": "DateTime",
                        "physicalType": "datetime"
                    },
                    "sink": {
                        "name": "[redacted]",
                        "type": "DateTime",
                        "physicalType": "timestamp without time zone"
                    }
                },
                {
                    "source": {
                        "name": "[redacted]",
                        "type": "Int32",
                        "physicalType": "int"
                    },
                    "sink": {
                        "name": "[redacted]",
                        "type": "Int32",
                        "physicalType": "integer"
                    }
                },
                {
                    "source": {
                        "name": "[redacted]",
                        "type": "Decimal",
                        "physicalType": "money",
                        "scale": 4,
                        "precision": 19
                    },
                    "sink": {
                        "name": "[redacted]",
                        "type": "Decimal",
                        "physicalType": "money"
                    }
                }
            ],
            "typeConversion": true,
            "typeConversionSettings": {
                "allowDataTruncation": true,
                "treatBooleanAsNumber": false
            }
        }
    },
    "inputs": [
        {
            "referenceName": "Hospitality_PMS_ManagerReport",
            "type": "DatasetReference"
        }
    ],
    "outputs": [
        {
            "referenceName": "Hospitality_DW_ManagerReport",
            "type": "DatasetReference"
        }
    ]
```

Gambar 3.69 *Code Activity GetManagerReportPMS* dalam *Pipeline* DW_ManagerReport

3.2.6.2. *Pipeline* Hospitality_DW_RoomCountSheet

Alur *pipeline* Hospitality_DW_RoomCountSheet seperti ditunjukkan pada Gambar 3.70 dirancang sebagai komponen utama dalam proses ETL untuk memperbarui data harian transaksi dan pemakaian kamar hotel. Tahap pertama diawali dengan *activity GetLastUpdateRoomCountSheet* yang mengambil waktu sinkronisasi terakhir agar hanya data terbaru yang diproses. Informasi ini digunakan pada tahap kedua dan ketiga untuk menetapkan dua variabel penting, *SyncStartVariable* dan *LastUpdateVariable* sebagai penanda waktu pemrosesan dan acuan validasi. Pada tahap keempat, *DeleteTable* digunakan untuk menghapus isi tabel tujuan guna menghindari duplikasi atau inkonsistensi saat data baru dimuat. Tahap kelima, *GetRoomCountSheet*, menyalin data dari *database* sumber PMS ke *data warehouse* tujuan hospitalitydwh dengan struktur tabel yang telah disesuaikan. Setelah data berhasil dipindahkan, tahap keenam yaitu *UpdateLastSyncTime* mencatat waktu sinkronisasi terbaru sebagai referensi untuk proses berikutnya. Seluruh tahapan dijalankan secara linier dan saling bergantung untuk menjaga integritas data.



Gambar 3.70 Alur *Pipeline* Hospitality_DW_RoomCountSheet

Proses alur *pipeline* Hospitality_DW_RoomCountSheet memiliki struktur yang mirip dengan Hospitality_DW_Odoo_BanquetReservation,

namun berbeda pada *query*, tabel, dan *activity Copy Data*. Secara umum, *activity GetLastUpdateRoomCountSheet*, *SyncStartVariable*, *LastUpdateVariable*, *DeleteTable*, dan *UpdateLastSyncTime* memiliki fungsi serupa dengan *pipeline* BanquetReservation, hanya berbeda pada nama tabel yang diproses. *GetLastUpdateRoomCountSheet* mengambil timestamp pembaruan terakhir dari tabel dw_synctablelog untuk pms_reproomrevenueroomcountsheet, yang digunakan sebagai acuan pemrosesan data inkremental. *DeleteTable*, dengan query seperti pada Gambar 3.68, menghapus data lama dari tabel pms_reproomrevenueroomcountsheet di hospitalitydwh berdasarkan *timestamp* dari *GetLastUpdate*. Kemudian *UpdateLastSyncTime* untuk memperbarui *timestamp* sinkronisasi pada dw_synctablelog untuk tabel tersebut. Perbedaan utama *pipeline* ini terletak pada *activity GetRoomCountSheet*, yang menyalin dan memetakan data sesuai struktur tabel pms_reproomrevenueroomcountsheet.

a. *Activity GetRoomCountSheet*

Activity GetRoomCountSheet bertipe *copy data* dengan kode seperti pada Gambar 3.71 dirancang untuk mentransfer data detail kamar hotel dari *database* PMS ke *data warehouse* hospitalitydwh. Sumber data diambil dari tabel repRoomRevenueRoomCountSheet dengan memfilter data hotel (*HotelId*) yang bukan 999 dan 900, serta hanya menyalin data dengan tanggal transaksi (*TrxDate*) yang lebih baru atau sama dengan nilai *LastUpdate* dari variabel *pipeline*. Data ini kemudian dimasukkan ke *sink* PostgreSQL (DB hospitalitydwh) menggunakan *CopyCommand*, dengan melakukan *mapping* data yang memastikan semua detail terkait kamar seperti tarif, diskon, dan pendapatan ditransfer dengan benar. Langkah ini mencakup proses *extract* dengan filter berdasarkan *LastUpdateVariable*, *transform* ringan berupa pemetaan dan konversi tipe data, serta *load* data ke tabel pms_reproomrevenueroomcountsheet secara bersamaan dalam satu *activity*.

Activity name GetRoomCountSheet

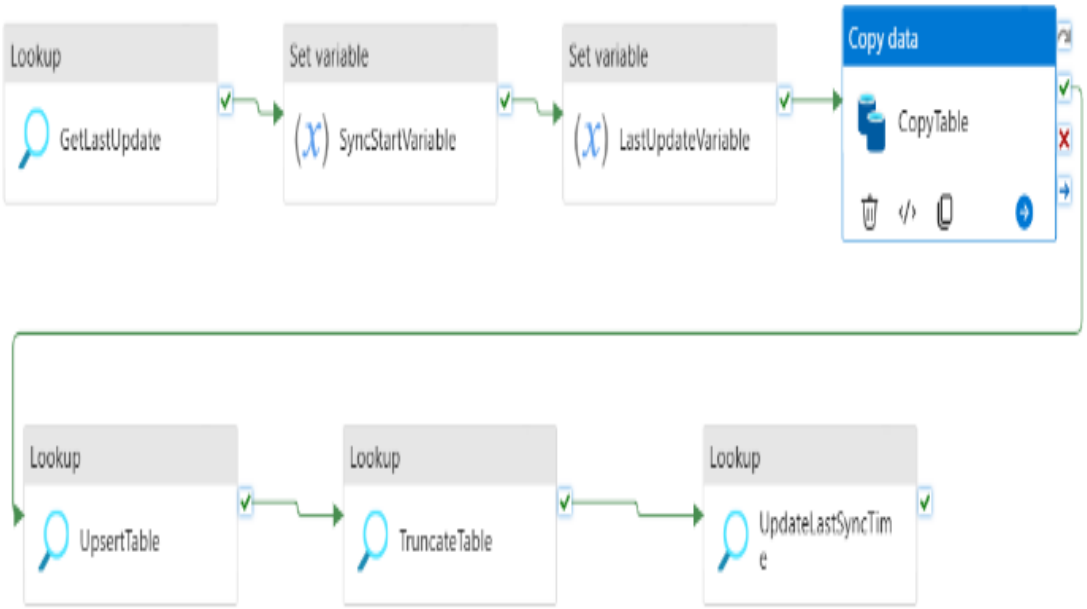
Copy to clipboard

```
{
    "name": "GetRoomCountSheet",
    "type": "Copy",
    "dependsOn": [
        {
            "activity": "DeleteTable",
            "dependencyConditions": [
                "Succeeded"
            ]
        }
    ],
    "policy": {
        "timeout": "0.00:30:00",
        "retry": 5,
        "retryIntervalInSeconds": 30,
        "secureOutput": false,
        "secureInput": false
    },
    "userProperties": [],
    "typeProperties": {
        "source": {
            "type": "AzureSqlSource",
            "sqlReaderQuery": {
                "value": "select * from repRoomRevenueRoomCountSheet\nwhere HotelId <> '999' and HotelId <> '900' and TrxDate >= '@{variables('LastUpdate')}'\n\n",
                "type": "Expression"
            },
            "queryTimeout": "02:00:00",
            "partitionOption": "None"
        },
        "sink": {
            "type": "AzurePostgreSQLSink",
            "writeBatchSize": 1000000,
            "writeBatchTimeout": "00:30:00",
            "writeMethod": "CopyCommand"
        },
        "enableStaging": false.
        "translator": {
            "type": "TabularTranslator",
            "mappings": [
                {
                    "source": {
                        "name": "HotelId",
                        "type": "Int32",
                        "physicalType": "int"
                    },
                    "sink": {
                        "name": "hotelid",
                        "type": "Int32",
                        "physicalType": "integer"
                    }
                },
                {
                    "source": {
                        "name": "[redacted]",
                        "type": "DateTime",
                        "physicalType": "datetime"
                    },
                    "sink": {
                        "name": "[redacted]",
                        "type": "DateTime",
                        "physicalType": "timestamp without time zone"
                    }
                },
                {
                    "source": {
                        "name": "[redacted]",
                        "type": "Int64",
                        "physicalType": "bigint"
                    },
                    "sink": {
                        "name": "[redacted]",
                        "type": "Int64",
                        "physicalType": "double precision"
                    }
                },
                {
                    "source": {
                        "name": "[redacted]",
                        "type": "Double",
                        "physicalType": "float"
                    },
                    "sink": {
                        "name": "[redacted]",
                        "type": "Double",
                        "physicalType": "double precision"
                    }
                }
            ],
            "typeConversion": true,
            "typeConversionSettings": {
                "allowDataTruncation": true,
                "treatBooleanAsNumber": false
            }
        }
    },
    "inputs": [
        {
            "referenceName": "Hospitality_PMS_RoomCountSheet",
            "type": "DatasetReference"
        }
    ],
    "outputs": [
        {
            "referenceName": "Hospitality_DW_RoomCountSheet",
            "type": "DatasetReference"
        }
    ]
}
```

Gambar 3.71 *Code Activity GetRoomCountSheet* dalam *Pipeline* DW_RoomCountSheet

3.2.6.3. *Pipeline* Hospitality_DW_BudgetMarketSegment

Alur *pipeline* Hospitality_DW_BudgetMarketSegment seperti pada Gambar 3.72 dirancang untuk memperbarui data rencana anggaran pendapatan kamar dan jumlah malam kamar per segmen pasar dari *database* PMS ke hospitalitydwh. Tahap pertama dimulai dengan *GetLastUpdate* untuk mengambil informasi waktu pembaruan terakhir, sehingga hanya data terbaru yang diproses. Waktu ini kemudian disimpan dalam dua variabel: *SyncStartVariable* pada tahap kedua sebagai penanda awal sinkronisasi, dan *LastUpdateVariable* pada tahap ketiga sebagai acuan data terakhir yang telah diproses. Selanjutnya, tahap keempat *CopyTable* menyalin data dari *database* sumber ke *staging area* di hospitalitydwh sebagai tempat penampungan awal. Tahap kelima *UpsertTable*, memperbarui atau menambahkan data ke tabel target agar informasi anggaran tiap segmen pasar tetap akurat dan konsisten. Setelah itu, tahap keenam *TruncateTable* mengosongkan *staging table* untuk mencegah duplikasi. Terakhir, tahap ketujuh *UpdateLastSyncTime* mencatat waktu sinkronisasi terkini sebagai referensi untuk proses berikutnya.



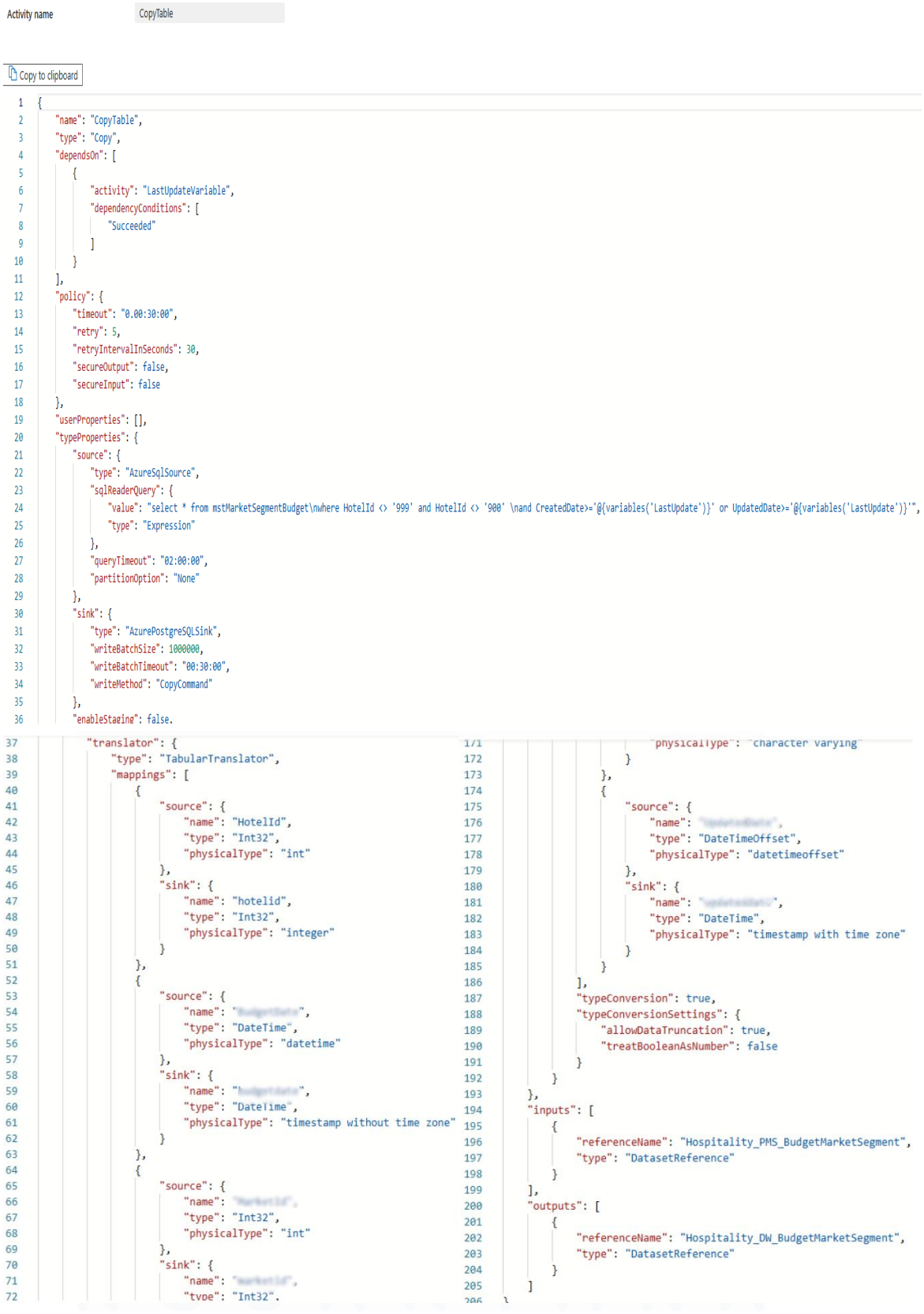
Gambar 3.72 Alur *Pipeline* Hospitality_DW_BudgetMarketSegment

Proses alur *pipeline* Hospitality_DW_BudgetMarketSegment mirip dengan Odoo_BanquetReservation, namun berbeda pada *query*, tabel, *activity*

CopyTable, dan *activity UpsertTable*. Secara umum, *activity GetLastUpdate*, *SyncStartVariable*, *LastUpdateVariable*, *TruncateTable*, dan *UpdateLastSyncTime* memiliki penjelasan serupa dengan *pipeline* BanquetReservation, hanya berbeda pada nama tabel yang disalin. *Activity GetLastUpdate* digunakan untuk mengambil *timestamp* pembaruan terakhir dari tabel dw_synctablelog yang spesifik untuk pms_mstmarketsegmentbudget untuk menentukan data inkremental yang perlu disalin. *Activity TruncateTable* membersihkan tabel stg_mstmarketsegmentbudget di hospitalitydwh sebelum data baru disalin. *Activity UpdateLastSyncTime* memperbarui *timestamp* sinkronisasi di dw_synctablelog tabel pms_mstmarketsegmentbudget. Perbedaan utamanya terletak pada *activity CopyTable* dan *UpsertTable*, dimana masing-masing *pipeline* memetakan dan menyalin data sesuai dengan tabel yang dituju. Berikut penjelasan dari *activity CopyTable* dan *UpsertTable* pada *pipeline* Hospitality_DW_BudgetMarketSegment:

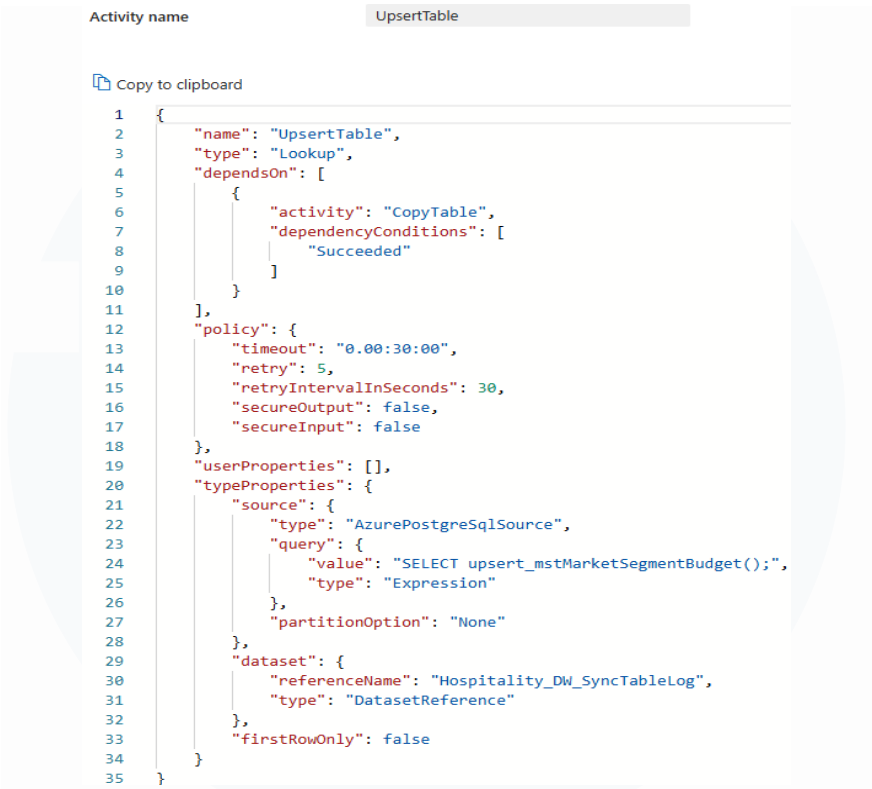
a. *Activity CopyTable*

Activity CopyTable dengan kode seperti pada Gambar 3.73 adalah inti dari proses ETL yang berfungsi untuk menyalin data anggaran segmen pasar dari *database* sumber (PMS) ke *data warehouse* tujuan (hospitalitydwh). *Activity* ini secara selektif mengambil data inkremental dari tabel mstMarketSegmentBudget berdasarkan *CreatedDate* atau *UpdatedDate* yang lebih baru dari *LastUpdateVariable*, sambil mengecualikan *HotelId* tertentu. Data yang disaring ini kemudian disisipkan ke *sink* PostgreSQL hospitalitydwh menggunakan metode *CopyCommand*. Proses ini memastikan bahwa data anggaran segmen pasar yang masuk ke hospitalitydwh sudah bersih dan terbaru. Langkah ini mencakup proses *extract* dengan filter berdasarkan *LastUpdateVariable*, *transform* ringan berupa pemetaan dan konversi tipe data, serta *load* data ke tabel pms_mstmarketsegmentbudget secara bersamaan dalam satu *activity*.



Gambar 3.73 *Code Activity CopyTable* dalam *Pipeline* BudgetMarketSegment

b. *Activity UpsertTable*



Gambar 3.74 *Code Activity UpsertTable* dalam *Pipeline* BudgetMarketSegment

Activity UpsertTable bertipe *Lookup* dengan kode seperti pada Gambar 3.74 bergantung pada keberhasilan *CopyTable*. Tujuan utamanya adalah untuk melakukan pembaruan atau penyisipan data pada tabel anggaran segmen pasar di *data warehouse* hospitalitydwh. Pembaruan ini dicapai dengan memanggil *function* PostgreSQL *upsert_mstMarketSegmentBudget()* melalui sebuah *query*. *Function* dengan *query* seperti pada Gambar 3.75 ini mengimplementasikan logika *INSERT ... ON CONFLICT* (...) *DO UPDATE SET* ..., yang berarti data dari tabel *staging* stg_mstMarketSegmentBudget akan dimasukkan ke tabel tujuan pms_mstMarketSegmentBudget. Apabila kombinasi kolom yang diperlukan sudah ada, baris yang sesuai akan diperbarui dengan nilai-nilai terkini dari data *staging*, memastikan tabel tujuan selalu berisi data anggaran segmen pasar yang paling baru. Langkah ini merupakan proses *load* yang

melibatkan operasi *upsert* untuk memasukkan data baru atau memperbarui data yang sudah ada di tabel utama berdasarkan *primary key* atau ID.

```
CREATE OR REPLACE FUNCTION public.upsert_mstmarketsegmentbudget()
 RETURNS void
 LANGUAGE plpgsql
AS $function$
BEGIN
  INSERT INTO pms_mstMarketSegmentBudget (
    hotelid,
    budgetdate,
    marketid,
    revision,
    roomnight,
    roomrevenue,
    remark,
    rowchange,
    createdby,
    createddate,
    updatedby,
    updateddate
  )
  SELECT hotelid,
    budgetdate,
    marketid,
    revision,
    roomnight,
    roomrevenue,
    remark,
    rowchange,
    createdby,
    createddate,
    updatedby,
    updateddate
  FROM stg_mstMarketSegmentBudget
  WHERE marketid IS NOT NULL AND hotelid IS NOT NULL AND budgetdate IS NOT NULL
  ON CONFLICT (marketid, hotelid, budgetdate,revision)
  DO UPDATE SET
    revision = EXCLUDED.revision,
    roomnight = EXCLUDED.roomnight,
    roomrevenue = EXCLUDED.roomrevenue,
    remark = EXCLUDED.remark,
    rowchange = EXCLUDED.rowchange,
    createdby = EXCLUDED.createdby,
    createddate = EXCLUDED.createddate,
    updatedby = EXCLUDED.updatedby,
    updateddate = EXCLUDED.updateddate;
END;
$function$
;
```

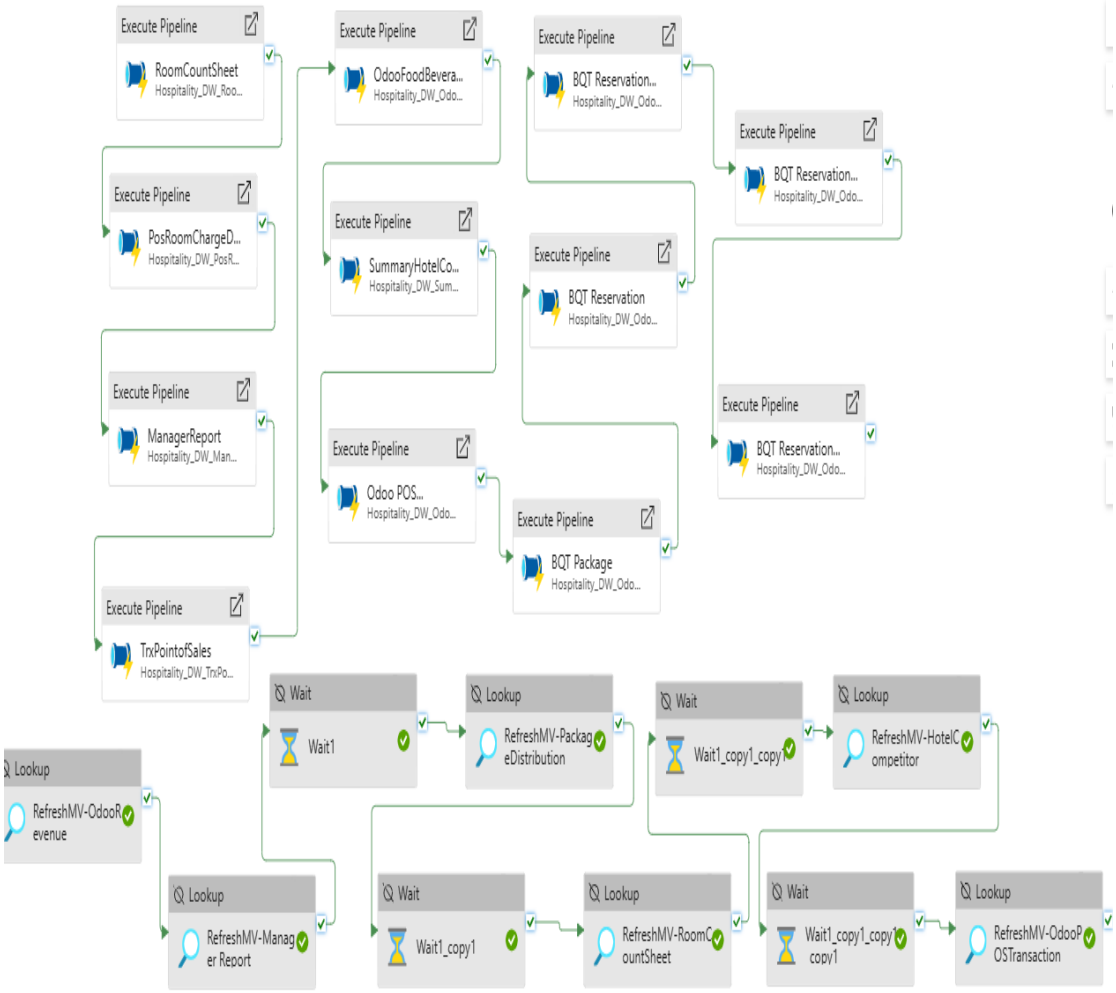
Gambar 3.75 *Query Function upsert_mstMarketSegmentBudget()*

3.2.7. ETL *Process* – Alur *Job Pipeline* dan *Pipeline Refresh* MV

Proses automasi dalam siklus *Extract, Transform, Load* (ETL) memiliki peran penting dalam memastikan data pada *data warehouse* hospitalitydwh selalu terbarui dan dapat diandalkan sebagai sumber utama untuk *dashboard* Apache Superset. Pembaruan data secara berkala diperlukan agar informasi yang ditampilkan mencerminkan kondisi operasional hotel terkini. Untuk mendukung hal ini, dibutuhkan dua alur kerja tambahan, yaitu *job pipeline* dan *pipeline refresh* MV. *Job pipeline* berfungsi menjalankan serangkaian proses ETL secara terjadwal dan berurutan tanpa intervensi manual, sedangkan *pipeline refresh* MV memperbarui isi *materialized view* agar tetap sinkron dengan data terbaru di *database*. Kedua *pipeline* ini saling melengkapi dalam menjaga stabilitas dan integrasi alur pembaruan data dalam sistem operasional *Hospitality Solution*. Perancangan *pipeline* automasi ini menjadi syarat utama

sebelum membuat *dashboard* visualisasi, karena seluruh komponen *dashboard* sangat bergantung pada data terbaru hasil proses ETL.

3.2.7.1. *Job Pipeline* Hospitality_Job_DailyReport



Gambar 3.76 Alur *Job Pipeline* Hospitality_Job_DailyReport

Alur *job pipeline* Hospitality_Job_DailyReport seperti pada Gambar 3.76 dirancang sebagai *master pipeline* harian yang berfungsi mengoordinasikan proses eksekusi sejumlah sub-*pipeline* ETL dan pembaruan *materialized view* (MV) secara terjadwal. Alur *pipeline* ini terdiri atas dua bagian utama, yaitu kelompok *activity Execute Pipeline* di bagian atas yang berjalan paralel, dan kelompok *activity Lookup-Wait* di bagian bawah yang berjalan sekuensial. Pada bagian atas, *pipeline* mengaktifkan dua belas sub-*pipeline* berbeda yang masing-masing *pipeline* menangani proses *extract*,

transform, dan *load* (ETL) data dari berbagai sumber *database* menuju satu *data warehouse* hospitalitydwh. Seluruh sub-*pipeline* ini dijalankan secara paralel agar menghemat waktu proses dan memaksimalkan pemanfaatan *resources* di Azure Data Factory. Setelah seluruh proses ETL selesai, alur berpindah ke tahap bawah yang diawali dengan *activity Lookup* untuk mengeksekusi perintah *refresh* pada masing-masing *materialized view* (MV). Setelah perintah *refresh* MV, proses dilanjutkan dengan *activity Wait* untuk menahan sementara eksekusi sebelum melanjutkan ke *Lookup* berikutnya. Pola ini diulang beberapa kali untuk memastikan setiap *materialized view* diperbarui secara berurutan dan tidak tumpang tindih.

a. *Activity Execute Pipeline*

Activity name BQT Reservation

Copy to clipboard

```
{
    "name": "BQT Reservation",
    "type": "ExecutePipeline",
    "dependsOn": [
        {
            "activity": "BQT Package",
            "dependencyConditions": [
                "Succeeded"
            ]
        }
    ],
    "policy": {
        "secureInput": false
    },
    "userProperties": [],
    "typeProperties": {
        "pipeline": {
            "referenceName": "Hospitality_DW_Odoo_BanquetReservation",
            "type": "PipelineReference"
        },
        "waitOnCompletion": true
    }
}
```

Gambar 3.77 Salah Satu *Code Activity Execute Pipeline* dalam *Pipeline* Job_DailyReport

Activity Execute Pipeline dengan kode seperti pada Gambar 3.77 bertugas untuk mengeksekusi *pipeline* lain bernama "Hospitality_DW_Odoo_BanquetReservation". Eksekusi *activity* ini hanya akan berjalan jika *activity* sebelumnya, yaitu "BQT *Package*", berhasil dijalankan (*dependency condition: Succeeded*). Parameter *waitOnCompletion*: *true* memastikan bahwa *pipeline* utama

Hospitality_Job_DailyReport akan menunggu hingga *pipeline* yang dieksekusi ini selesai sepenuhnya sebelum melanjutkan ke langkah berikutnya. *Activity* ini merupakan bagian dari proses ETL harian untuk menarik dan mengolah data reservasi *banquet* dari odoo ke dalam *data warehouse* hospitalitydwh. Dalam *job pipeline* Hospitality_Job_DailyReport terdapat banyak *activity* dengan tipe *Execute Pipeline*, secara prinsip semuanya memiliki fungsi inti yang sama, yakni menjalankan *pipeline* lain sebagai bagian dari rangkaian proses ETL harian.

b. *Activity Lookup*

Activity name: RefreshMV-Manager Report

Copy to clipboard

```
{
    "name": "RefreshMV-Manager Report",
    "description": "",
    "type": "Lookup",
    "state": "Inactive",
    "onInactiveMarkAs": "Succeeded",
    "dependsOn": [
        {
            "activity": "RefreshMV-OdooRevenue",
            "dependencyConditions": [
                "Succeeded"
            ]
        }
    ],
    "policy": {
        "timeout": "0.00:15:00",
        "retry": 2,
        "retryIntervalInSeconds": 30,
        "secureOutput": false,
        "secureInput": false
    },
    "userProperties": [
        {
            "name": "commandtimeout",
            "value": "600"
        },
        {
            "name": "timeout",
            "value": "600"
        }
    ],
    "typeProperties": {
        "source": {
            "type": "AzurePostgreSqlSource",
            "query": {
                "value": "SELECT refresh_mv('mv_manager_report');",
                "type": "Expression"
            },
            "partitionOption": "None"
        },
        "dataset": {
            "referenceName": "Hospitality_DW_SyncTableLog",
            "type": "DatasetReference"
        }
    }
}
```

Gambar 3.78 Salah Satu *Code Activity Lookup* dalam *Pipeline* Job_DailyReport

Activity bertipe *Lookup* dengan nama "RefreshMV-Manager Report" seperti ditunjukkan pada Gambar 3.78 bertugas untuk menjalankan *lookup* yang memanggil query SQL untuk melakukan *refresh* pada *materialized view* (MV). *Activity* ini akan dieksekusi setelah *activity* "RefreshMV-OdooRevenue" berhasil dijalankan (*dependency condition: Succeeded*). *Function refresh_mv()* dengan *query* seperti pada Gambar 3.79 ini digunakan untuk me-*refresh materialized view* berdasarkan nama yang diterima sebagai parameter (dalam hal ini, "mv_manager_report"). *Activity* ini diatur untuk menunggu dengan *timeout* selama 15 menit dan melakukan

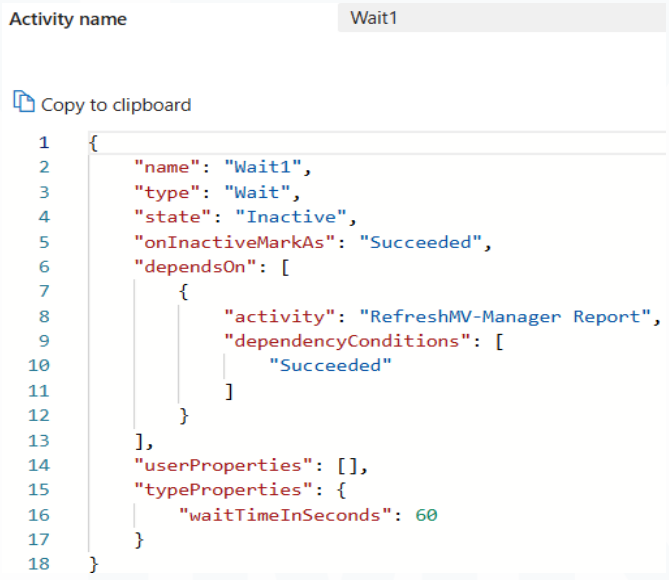
dua kali percobaan ulang jika diperlukan, serta menggunakan *dataset* yang terhubung dengan Hospitality_DW_SyncTableLog. Dalam *job pipeline* Hospitality_Job_DailyReport terdapat banyak *activity* dengan tipe *Lookup*, secara prinsip semuanya memiliki fungsi inti yang sama.

```
CREATE OR REPLACE FUNCTION public.refresh_mv(mv_name text)
 RETURNS void
 LANGUAGE plpgsql
AS $function$
BEGIN
    EXECUTE format('REFRESH MATERIALIZED VIEW %I', mv_name);

END;
$function$
;
```

Gambar 3.79 *Query Function refresh_mv()*

c. *Activity Wait*

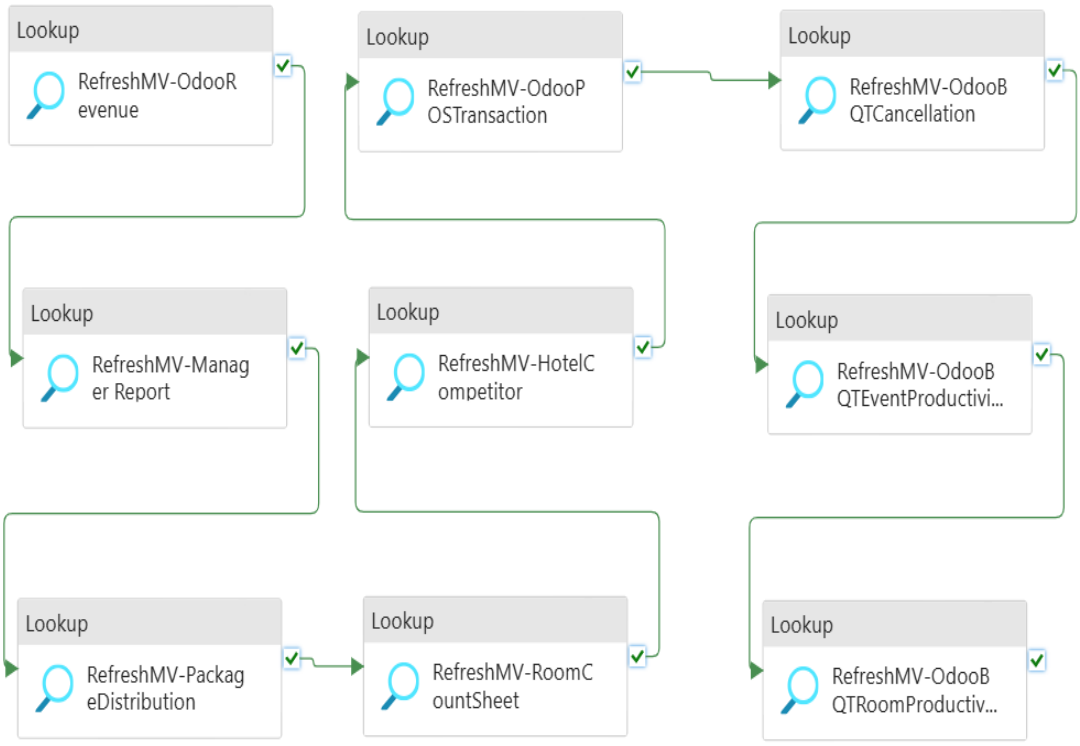


Gambar 3.80 Salah Satu *Code Activity Wait* dalam *Pipeline* Job_DailyReport

Activity Wait dengan kode seperti pada Gambar 3.80 bertugas untuk menunggu selama 60 detik sebelum melanjutkan ke langkah berikutnya. *Activity* ini hanya akan dieksekusi setelah *activity RefreshMV* berhasil dijalankan. Waktu tunggu yang ditentukan adalah 60 detik, yang diatur dalam properti *waitTimeInSeconds*. *Activity* ini membantu untuk memberi jeda waktu sebelum proses lebih lanjut, memastikan bahwa semua proses sebelumnya telah selesai dengan benar.

3.2.7.2. *Pipeline* Hospitality_DW_RefreshMVDaily

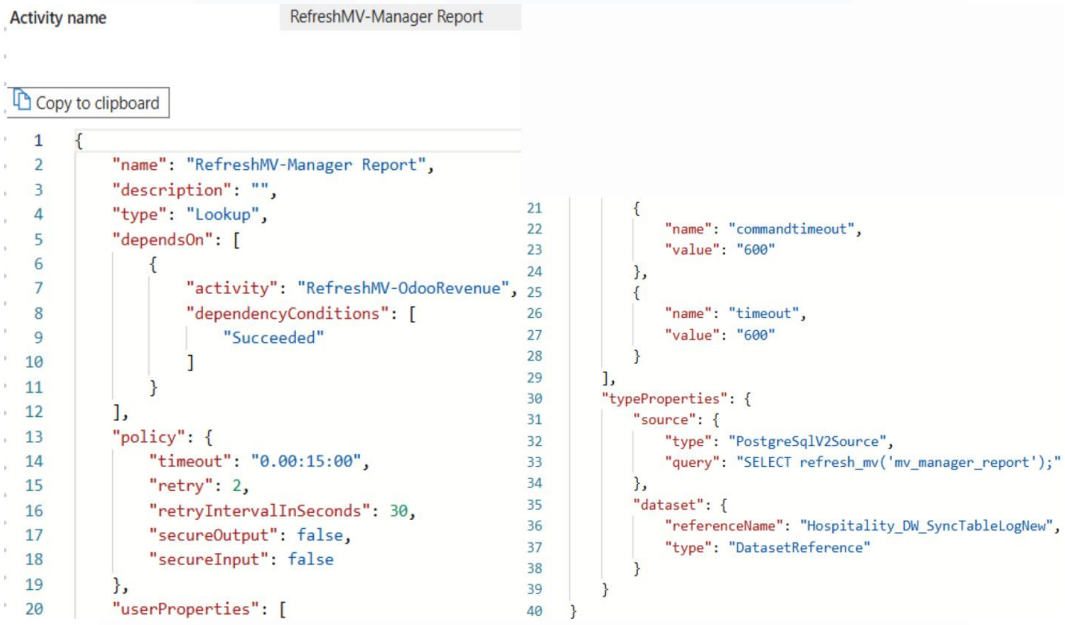
Alur *pipeline* Hospitality_DW_RefreshMVDaily seperti pada Gambar 3.81 dirancang untuk memperbarui sejumlah *materialized view* (MV) secara terjadwal setiap hari sebagai bagian dari proses penyiapan data untuk *dashboard* operasional hotel. Alur *pipeline* ini terdiri dari deretan *activity Lookup* yang terhubung secara linear dan menjalankan perintah *refresh* terhadap berbagai MV yang menyimpan data penting seperti pendapatan Odoo, performa *manager report*, data distribusi paket, informasi kompetitor dan transaksi POS, dan lainnya. Setiap *Lookup* akan mengeksekusi *query* SQL tertentu yang memaksa PostgreSQL memperbarui isi MV agar mencerminkan data terkini yang sebelumnya telah diproses melalui *pipeline* ETL. Struktur berurutan pada *pipeline* ini menunjukkan bahwa *refresh* MV dilakukan secara bertahap berdasarkan prioritas dan ketergantungan data antar MV. Pembuatan *pipeline* ini memperkuat pendekatan automasi penuh dalam alur kerja data harian, dimana tidak ada intervensi manual dibutuhkan untuk memperbarui informasi visual yang digunakan oleh tim manajemen hotel.



Gambar 3.81 Alur *Pipeline* Hospitality_DW_RefreshMVDaily

a. *Activity RefreshMV*

Activity RefreshMV dengan kode seperti pada Gambar 3.82 bertugas untuk menjalankan *lookup* yang memanggil *query* SQL untuk melakukan *refresh* pada *materialized view* (MV). *Activity* ini akan dijalankan setelah *activity* "RefreshMV-OdooRevenue" berhasil dijalankan (*dependency condition: Succeeded*). *Query* yang digunakan adalah *SELECT refresh_mv('mv_manager_report');*, yang memanggil *function refresh_mv()* dengan *query* pada Gambar 3.79 untuk me-*refresh* MV tersebut. *Dataset* yang digunakan untuk *activity* ini adalah Hospitality_DW_SyncTableLogNew, yang mendukung pencatatan dan pelacakan status eksekusi *activity* ini. Dalam *pipeline* Hospitality_DW_RefreshMVDaily terdapat banyak *activity* dengan tipe *Lookup*, secara prinsip semuanya memiliki fungsi inti yang sama.



Gambar 3.82 Salah Satu *Code Activity RefreshMV* dalam *Pipeline* DW_RefreshMVDaily

3.2.8. *Developing Dashboards – Hospitality-Analytics Banquet Dashboard*

Pembangunan *Hospitality-Analytics Banquet Dashboard* merupakan tahap lanjutan setelah proses *Extract, Transform, Load* (ETL) berhasil dijalankan melalui *pipeline* data *banquet* yang dirancang menggunakan Azure Data Factory. *Pipeline* ini berfungsi untuk menyalin, menyelaraskan, dan

memuat data *banquet* dari sistem *Point of Sales* (POS) Odoo ke dalam *data warehouse* hospitalitydwh. Setelah data tersimpan secara konsisten dan terstruktur, tahap berikutnya adalah pengolahan visualisasi data menggunakan Apache Superset. *Dashboard Banquet* dirancang untuk memberikan *insight* terkait aktivitas penyewaan ruang acara hotel, mencakup total reservasi, profil perusahaan penyewa, tren penjualan, produktivitas ruang berdasarkan periode waktu tertentu, dan lainnya. *Dashboard* ini terdiri dari empat submenu utama yakni *Summary*, *Company*, *Sales & Room Productivity*, dan *Reservation* yang masing-masing memerlukan pembuatan *materialized views* (MV), perhitungan *metrics*, serta visualisasi dalam bentuk grafik dan tabel. Penyusunan *dashboard* dimulai dari pembuatan MV sebagai sumber *dataset*, yang kemudian dijadikan dasar pembentukan *metrics* dan *charts* sesuai kebutuhan analisis tiap submenu. Setiap submenu menyajikan perspektif analitis yang berbeda untuk mendukung pengambilan keputusan operasional dan strategis oleh manajemen hotel. Keberhasilan visualisasi ini sangat bergantung pada ketepatan dan keberlangsungan proses ETL, karena data yang diproses dalam tahap ETL menjadi fondasi visualisasi *dashboard* di Apache Superset. Integrasi antara proses ETL dan pembuatan *dashboard* membentuk satu kesatuan alur kerja dalam sistem *Business Intelligence* unit *Hospitality* Kompas Gramedia, yang memudahkan *Board of Directors* mengakses informasi secara cepat, akurat, dan relevan. Rincian fokus tiap submenu dalam *Banquet Dashboard* disajikan pada Tabel 3.5.

Tabel 3.5 Rincian Fokus Submenu dalam *Banquet Dashboard*

Submenu	***Dataset***	**Tujuan Visualisasi**
Summary	mv_bqt_eventproductivity	Menampilkan performa pendapatan dan aktivitas acara hotel, mencakup total *event*, jumlah tamu, tren *revenue*, komposisi pendapatan, detail paket dan pembatalan, serta tipe acara
	mv_bqt_cancellation	
Company	mv_bqt_eventproductivity	Menampilkan perusahaan penyumbang dan penyebab kerugian terbesar berdasarkan jumlah *event* dan total pendapatan
	mv_bqt_cancellation	
	mv_bqt_eventproductivity	

Submenu	*Dataset*	Tujuan Visualisasi
Sales & Room Productivity	mv_bqt_roomproductivity	Mengevaluasi kontribusi tim *sales* dan pemanfaatan ruang *meeting* terhadap jumlah *event* dan total pendapatan yang dihasilkan
Reservation	mv_bqt_eventproductivity	Memantau pola reservasi harian, tingkat pembatalan, dan rentang waktu antara pemesanan hingga pelaksanaan event
	mv_bqt_cancellation	

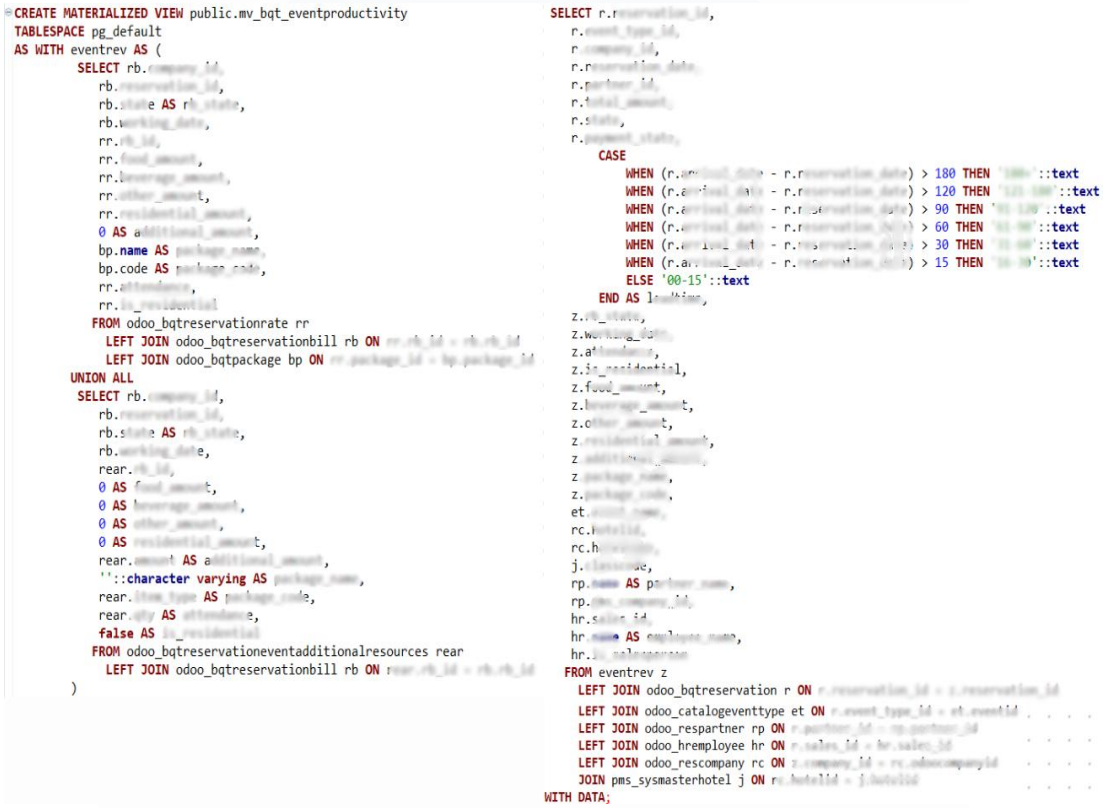
3.2.8.1. *Banquet Dashboard* Submenu *Summary*

Pembangunan submenu *Summary* pada *Banquet Dashboard* bertujuan untuk menyajikan ringkasan kinerja kegiatan *banquet* hotel secara menyeluruh melalui visualisasi interaktif yang mudah dipahami, sehingga memberikan pandangan strategis kepada *Board of Directors* terkait performa banquet sebagai salah satu sumber pendapatan utama dalam bisnis perhotelan. Berikut adalah input, *process*, dan *output* dari *Banquet Dashboard* submenu *Summary*:

a. Input

Penyusunan visualisasi pada submenu *Summary* dimulai dengan tahap penyediaan input berupa *dataset* terstruktur. *Dataset* yang digunakan dalam submenu ini berasal dari dua *Materialized View* (MV), yaitu mv_bqt_eventproductivity dan mv_bqt_cancellation yang dibentuk dari hasil integrasi beberapa tabel pada sistem *Point of Sales* (POS) Odoo. Perancangan MV dilakukan menggunakan perintah SQL dan disimpan dalam PostgreSQL hospitalitydwh untuk memastikan keterbaruan data secara periodik dan kestabilan performa *query* pada Superset. *Dataset* mv_bqt_eventproductivity dengan *query* seperti pada Gambar 3.83 dirancang untuk menggabungkan dan menyatukan dua jenis data, yaitu data tagihan reservasi (*bill*) serta data tambahan acara (*additional resources*), melalui proses *UNION ALL* pada tabel odoo_bqtreservationrate dan odoo_bqtreservationeventadditionalresources yang kemudian dihubungkan dengan berbagai referensi tambahan seperti odoo_bqtreservation, odoo_catalogeventtype, odoo_rescompany, dan odoo_hremployee.

Penggunaan *join* terhadap tabel pms_sysmasterhotel juga diperlukan untuk memperoleh informasi *classcode* dan pengelompokannya.



Gambar 3.83 *Query Materialized View* mv_bqt_eventproductivity

Dataset lainnya adalah mv_bqt_cancellation dengan *query* seperti pada Gambar 3.84 dibentuk dari tabel odoo_bqtreservation yang difilter berdasarkan status pembatalan (*state = 'release'*) serta diperkaya dengan informasi tambahan dari odoo_bqtreservationrate, odoo_rescompany, odoo_catalogeventtype, dan odoo_respartner untuk menangkap seluruh data yang relevan terhadap reservasi yang dibatalkan, termasuk nominal nilai pembatalan, waktu pembatalan (*canceldate*), jenis *event*, serta asal perusahaan penyewa. Kedua *materialized view* tersebut memuat seluruh atribut penting yang diperlukan dalam perhitungan metrik dan pembuatan visualisasi, seperti total pendapatan, *attendance*, tipe paket, status reservasi, nama event, serta dimensi waktu seperti *working_date* dan *reservation_date* yang menjadi kunci filter dalam penggunaan *dashboard*.

```
CREATE MATERIALIZED VIEW public.mv_bqt_cancellation
TABLESPACE pg_default
AS SELECT r.reservation_id,
    r.event_type_id,
    r.state,
    r.company_id,
    r.working_date,
    r.reservation_date,
    r.total_amount,
        CASE
            WHEN (r.arrival_date - r.reservation_date) > 180 THEN '180+'::text
            WHEN (r.arrival_date - r.reservation_date) > 120 THEN '121-180'::text
            WHEN (r.arrival_date - r.reservation_date) > 90 THEN '91-120'::text
            WHEN (r.arrival_date - r.reservation_date) > 60 THEN '61-90'::text
            WHEN (r.arrival_date - r.reservation_date) > 30 THEN '31-60'::text
            WHEN (r.arrival_date - r.reservation_date) > 15 THEN '16-30'::text
            ELSE '00-15'::text
        END AS lead_time,
    (r.write_date + make_interval(hours => j.timezone))::date AS canceldate,
    et.event_name,
    rc.hotelid,
    rc.hotelname,
    j.classcode,
    rr.is_residential,
    rr.amount AS reservation_amount,
    rp.pms_company_id,
    rp.name AS partner_name
   FROM odoo_bqtreservation r
     LEFT JOIN odoo_catalogeventtype et ON r.event_type_id = et.eventid
     LEFT JOIN odoo_rescompany rc ON r.company_id = rc.odoocompanyid
     LEFT JOIN odoo_bqtreservationrate rr ON r.reservation_id = rr.reservation_id
     LEFT JOIN odoo_respartner rp ON r.partner_id = rp.partner_id
     JOIN pms_sysmasterhotel j ON rc.hotelid = j.hotelid
  WHERE rc.hotelid <> 999
WITH DATA;
```

Gambar 3.84 *Query Materialized View* mv_bqt_cancellation

b. *Process*

Perancangan visualisasi pada submenu *Summary* memerlukan tahapan pemrosesan data berbasis logika agregasi dan kalkulasi metrik dari tabel hasil proses ETL. Daftar kalkulasi metrik dan filter yang digunakan untuk membuat *Banquet Dashboard* submenu *Summary* disajikan pada Tabel 3.6. *Dataset* utama yang digunakan adalah mv_bqt_eventproductivity yang dirancang untuk menghimpun data operasional *banquet* dari sistem *Point of Sales* (POS) Odoo dan menyusunnya dalam format tabel analitis. Pada tahap awal, metrik dasar seperti jumlah *event*, total tamu, dan pendapatan kotor dihitung menggunakan *aggregate function*, seperti *COUNT(DISTINCT reservation_id)* untuk jumlah *event* dan *SUM(attendance)* untuk total tamu. Pendapatan dihitung dari penjumlahan beberapa komponen biaya yang kemudian dibagi dengan 1.21 untuk menghilangkan komponen pajak 21% sesuai standar akuntansi KG *Hospitality*. Selanjutnya, dilakukan transformasi metrik evaluatif seperti *Average Rate per Event* dan *Average Check*, yang dihitung di Apache Superset melalui fitur *custom metric*

builder. *Average Rate per Event* diperoleh dari pembagian total pendapatan tanpa pajak dengan jumlah *event* dalam periode tertentu, sementara *Average Check* dihitung dari total pendapatan dibagi jumlah tamu. Seluruh metrik ini difilter menggunakan parameter spesifik seperti status reservasi (*checkout* atau *checkin*), tanggal transaksi (*working_date*), tipe hotel (*classcode*), dan nama hotel (*hotelname*).

Tabel 3.6 Rincian Metriks dan Filter pada *Banquet Dashboard* Submenu *Summary*

Nama *Chart*	Kalkulasi Metriks	Filter
Total Event (*Big Number*)	column: # reservation_id aggregate: COUNT_DISTINCT	state IN ('checkout', 'checkin') 2025-05-01 ≤ working_date < 2... classcode IN ('PREMIERE') hotelname IN ('Hotel Santika Prem...
Total Attendance (*Big Number*)	`SUM(attendance)`	state IN ('checkout', 'checkin') 2025-05-01 ≤ working_date < 2... classcode IN ('PREMIERE') hotelname IN ('Hotel Santika Prem...

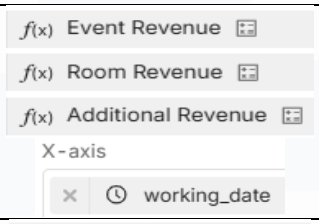
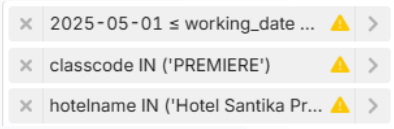
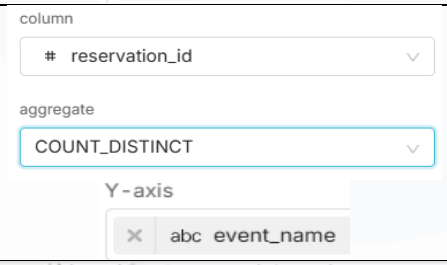
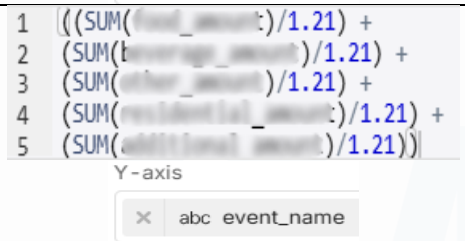
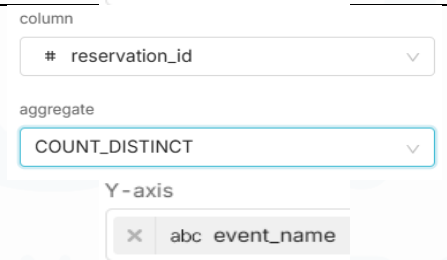
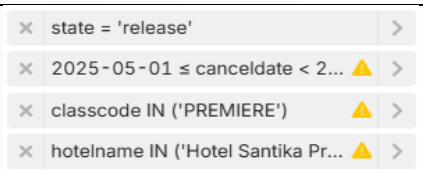
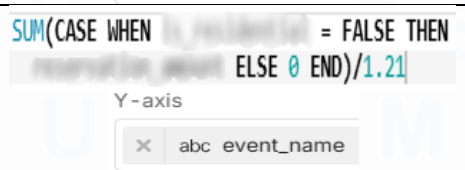
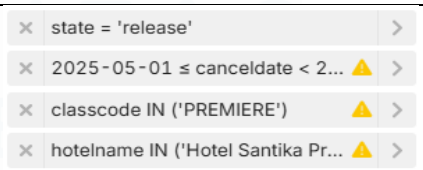
Total Revenue (*Big Number*)	`((SUM( )/1.21) +`
	`(SUM( )/1.21) +`
	`(SUM( )/1.21) +`
	`(SUM( )/1.21) +`
	`(SUM( )/1.21))`
		state IN ('checkout', 'checkin')
		2025-05-01 ≤ working_date < 2...
		classcode IN ('PREMIERE')
		hotelname IN ('Hotel Santika Prem...

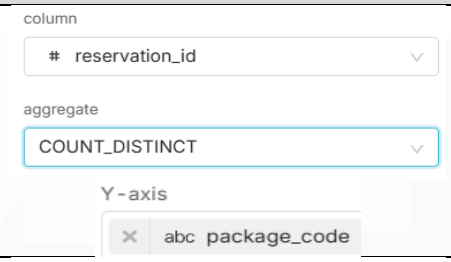
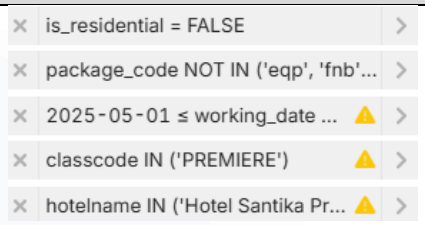
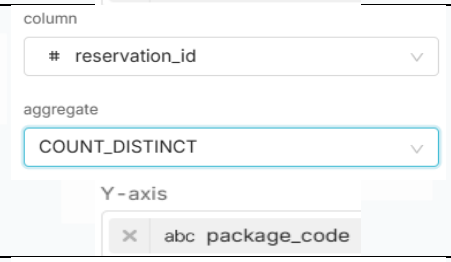
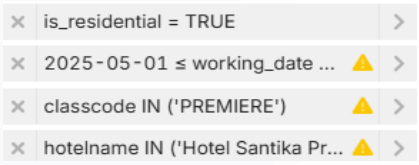
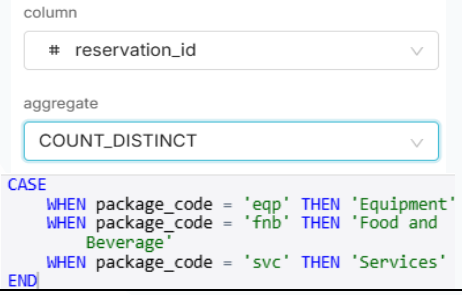
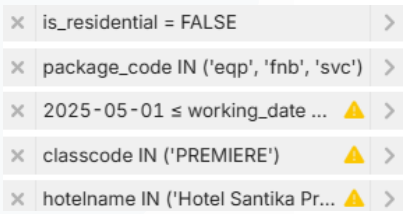
Event Revenue (*Big Number*)	`(SUM( )/1.21) +`
	`(SUM( )/1.21) +`
	`(SUM( )/1.21)`
		state IN ('checkout', 'checkin')
		2025-05-01 ≤ working_date < 2...
		classcode IN ('PREMIERE')
		hotelname IN ('Hotel Santika Prem...

Average Rate/Event (*Big Number*)	`((SUM( )/1.21) +`
	`(SUM( )/1.21) +`
	`(SUM( )/1.21) +`
	`(SUM( )/1.21) +`
	`(SUM( )/1.21)) / COUNT(DISTINCT`
		state IN ('checkout', 'checkin')
		2025-05-01 ≤ working_date < 2...
		classcode IN ('PREMIERE')
		hotelname IN ('Hotel Santika Prem...

Average Check (*Big Number*)	`((SUM( )/1.21) +`
	`(SUM( )/1.21) +`
	`(SUM( )/1.21) +`
	`(SUM( )/1.21) +`
	`(SUM( )/1.21)) / SUM( )`
		state IN ('checkout', 'checkin')
		2025-05-01 ≤ working_date < 2...
		classcode IN ('PREMIERE')
		hotelname IN ('Hotel Santika Prem...

Room Revenue (*Big Number*)	`SUM(CASE WHEN = TRUE THEN` `ELSE 0 END) / 1.21`	state IN ('checkout', 'checkin') 2025-05-01 ≤ working_date < 2... classcode IN ('PREMIERE') hotelname IN ('Hotel Santika Prem...

Nama *Chart*	Kalkulasi Metriks	Filter
Additional Revenue (*Big Number*)		
Banquet Revenue Trend (*Line Chart*)		
Food vs Beverage Revenue (*Bar Chart*)		
Event Type Details (*Bar Chart*)		
Total Revenue by Event (*Bar Chart*)		
Event Type Detail (*Cancellation*) (*Bar Chart*)		
Total Revenue by Event (*Cancellation*) (*Bar Chart*)		

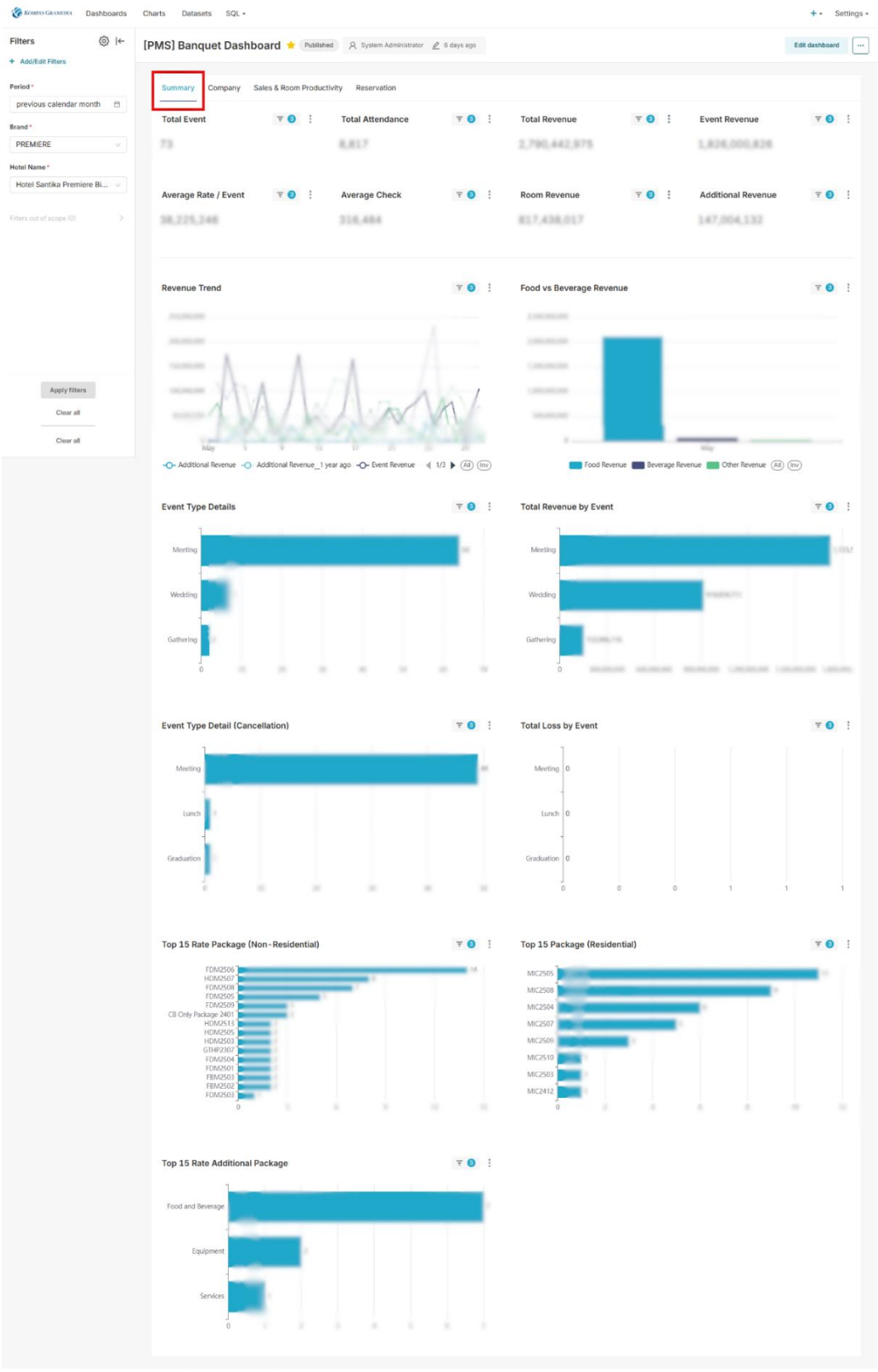
Nama *Chart*	Kalkulasi Metriks	Filter
Top 15 Rate Package (*Non-Residential*) (*Bar Chart*)		
Top 15 Rate Package (*Residential*) (*Bar Chart*)		
Top 15 Rate Additional Package (*Non-Residential*) (*Bar Chart*)		

Proses selanjutnya adalah perancangan visualisasi dalam bentuk *charts* untuk merepresentasikan metrik yang telah dikalkulasi dalam submenu *Summary*. Jenis visualisasi yang digunakan mencakup *Big Number*, *Line Chart*, dan *Bar Chart*, disesuaikan dengan karakteristik data yang disajikan. Metrik seperti Total *Event*, Total *Attendance*, Total *Revenue*, *Event Revenue*, serta pendapatan berdasarkan tipe (*room*, *additional*, *average rate*, dan *average check*) ditampilkan dalam format *Big Number* untuk menonjolkan angka utama di halaman utama *dashboard*. *Line Chart* digunakan untuk menampilkan *Banquet Revenue Trend* dalam bentuk tiga garis yakni *Event Revenue*, *Room Revenue*, dan *Additional Revenue* yang menggambarkan kontribusi masing-masing komponen terhadap total pendapatan harian berdasarkan sumbu waktu *working_date*. *Bar Chart* digunakan untuk menyajikan detail pendapatan berdasarkan tipe *event* dan paket layanan, seperti pada *Event Type Details* dan *Total Revenue by Event*, yang dibentuk dari agregasi per *event_name* menggunakan metrik

COUNT(DISTINCT reservation_id) dan total pendapatan bebas pajak. Visualisasi seperti *Top 15 Rate Package (Non-Residential)* dan *Top 15 Rate Additional Package (Non-Residential)* disusun melalui segmentasi berdasarkan kondisi *is_residential* dan pengelompokan *package_code* menggunakan logika *CASE* dalam SQL. *Charts Event Type Detail* (*Cancellation*) dan *Total Revenue by Event* (*Cancellation*) menggunakan dataset mv_bqt_cancellation untuk reservasi yang dibatalkan. Visualisasi pembatalan ini difilter berdasarkan *state = 'release'* dan *canceldate* pada periode yang sama.

c. *Output*

Submenu *Summary* pada *Banquet Dashboard* menghasilkan berbagai *output* visualisasi yang menyajikan performa keseluruhan aktivitas penyewaan ruang acara hotel dalam periode tertentu. Hasil *Banquet Dashboard* submenu *Summary* ditampilkan pada Gambar 3.85 yang mencakup indikator utama seperti *total event*, jumlah tamu, dan total pendapatan yang ditampilkan dalam bentuk *big number chart* untuk memberikan representasi nilai agregat secara ringkas. Selain itu, terdapat *line chart* yang menampilkan tren pendapatan berdasarkan kategori *Event*, *Room*, dan *Additional Revenue* dari hari ke hari, serta *bar chart* yang memperlihatkan distribusi pendapatan berdasarkan jenis acara dan jenis paket yang digunakan. Informasi tambahan terkait pembatalan reservasi, termasuk jumlah event yang dibatalkan dan nilai kerugian pendapatan, turut ditampilkan menggunakan chart terpisah yang bersumber dari *dataset* khusus untuk pembatalan. Seluruh *output* tersebut akan di-*embed* ke dalam sistem PMS dan digunakan oleh manajemen hotel, khususnya *Board of Directors* untuk mengidentifikasi performa bisnis *banquet* secara cepat, mengevaluasi kontribusi jenis acara terhadap pendapatan, serta menetapkan kebijakan penjualan dan strategi pemasaran berdasarkan pola pemesanan dan tipe layanan yang paling banyak digunakan.



Gambar 3.85 *Output Banquet Dashboard* Submenu *Summary*

3.2.8.2. *Banquet Dashboard* Submenu *Company*

Pembangunan submenu *Company* pada *Banquet Dashboard* bertujuan untuk menampilkan kontribusi dan potensi kerugian dari masing-masing perusahaan penyewa ruang acara hotel melalui visualisasi *bar chart* interaktif yang membandingkan jumlah *event*, pendapatan, serta pembatalan yang terjadi. Berikut adalah input, *process*, dan *output* dari *Banquet Dashboard* submenu *Company*:

a. Input

Pembangunan submenu *Company* dalam *Banquet Dashboard* turut menggunakan dua *Materialized View* (MV), yaitu mv_bqt_eventproductivity dengan *query* seperti pada Gambar 3.83 dan mv_bqt_cancellation dengan *query* seperti pada Gambar 3.84 yang sebelumnya telah dijelaskan pada bagian *Banquet Dashboard* Submenu *Summary*. Meskipun menggunakan sumber *dataset* yang sama, perbedaan terletak pada fokus analisis, fungsi visualisasi, serta kolom yang dimanfaatkan untuk membentuk metrik dan dimensi pada masing-masing *chart*. Submenu *Summary* menitikberatkan pada performa keseluruhan acara, sedangkan submenu *Company* dirancang untuk menyoroti kontribusi dan kerugian yang ditimbulkan oleh masing-masing perusahaan penyewa ruang acara hotel. Kolom-kolom yang digunakan dalam submenu *Company* lebih terfokus pada *pms_company_id* dan *partner_name* yang digabungkan untuk membentuk identitas perusahaan, serta *reservation_id* dan *reservation_amount* yang digunakan dalam penghitungan total *event* dan total *revenue*. Selain itu, pemanfaatan *state*, *canceldate*, dan *working_date* sebagai filter waktu dan status reservasi menjadi elemen penting dalam membatasi cakupan data agar sesuai dengan periode analisis yang ditentukan.

b. *Process*

Perancangan visualisasi pada submenu *Company* memerlukan tahapan pemrosesan data berbasis logika agregasi dan kalkulasi metrik dari tabel

hasil proses ETL. Daftar kalkulasi metrik dan filter yang digunakan untuk membuat *Banquet Dashboard* submenu *Company* disajikan pada Tabel 3.7. Dua *dataset* yang digunakan yaitu mv_bqt_eventproductivity dan mv_bqt_cancellation, masing-masing menyimpan informasi terkait *event* yang terlaksana dan yang dibatalkan. Metrik utama untuk menghitung jumlah *event* perusahaan dirumuskan dengan *COUNT(DISTINCT reservation_id)* sebagai representasi total reservasi yang terhubung dengan perusahaan tertentu. Metrik pendapatan dihitung dari penjumlahan komponen tertentu kemudian dibagi 1.21 untuk menormalkan nilai sebelum pajak sesuai kebijakan akuntansi internal. Untuk pembatalan pendapatan, perhitungan dilakukan secara terpisah menggunakan *SUM(CASE WHEN is_residential = TRUE/FALSE THEN reservation_amount ELSE 0 END) / 1.21* untuk membedakan antara *event residential* dan *non-residential*. Seluruh metrik dilengkapi filter seperti *state* (*checkout, checkin, release*), *working_date*, *canceldate*, *classcode*, dan *hotelname* untuk memastikan data yang digunakan hanya mencakup transaksi aktual pada periode tertentu.

Tabel 3.7 Rincian Metriks dan Filter pada *Banquet Dashboard* Submenu *Company*

Nama *Chart*	Kalkulasi Metriks	Filter
Top 15 Company (*Event*) (*Bar Chart*)	column: # reservation_id aggregate: COUNT_DISTINCT `CONCAT(pms_company_id,' - ',left(partner_name, 30))`	state IN ('checkout', 'checkin') 2025-05-01 ≤ working_date < 2... classcode IN ('PREMIERE') hotelname IN ('Hotel Santika Prem...
Top 15 Company (*Revenue*) (*Bar Chart*)	`((SUM(...)/1.21) + (SUM(...)/1.21) + (SUM(...)/1.21) + (SUM(...)/1.21) + (SUM(...)/1.21))` `CONCAT(pms_company_id,' - ',left(partner_name, 30))`	state IN ('checkout', 'checkin') 2025-05-01 ≤ working_date < 2... classcode IN ('PREMIERE') hotelname IN ('Hotel Santika Prem...
Top 15 Company Cancellation (*Event*) (*Bar Chart*)	column: # reservation_id aggregate: COUNT_DISTINCT `CONCAT(pms_company_id,' - ',left(partner_name, 30))`	state = 'release' 2025-05-01 ≤ canceldate < 2... classcode IN ('PREMIERE') hotelname IN ('Hotel Santika Pr...

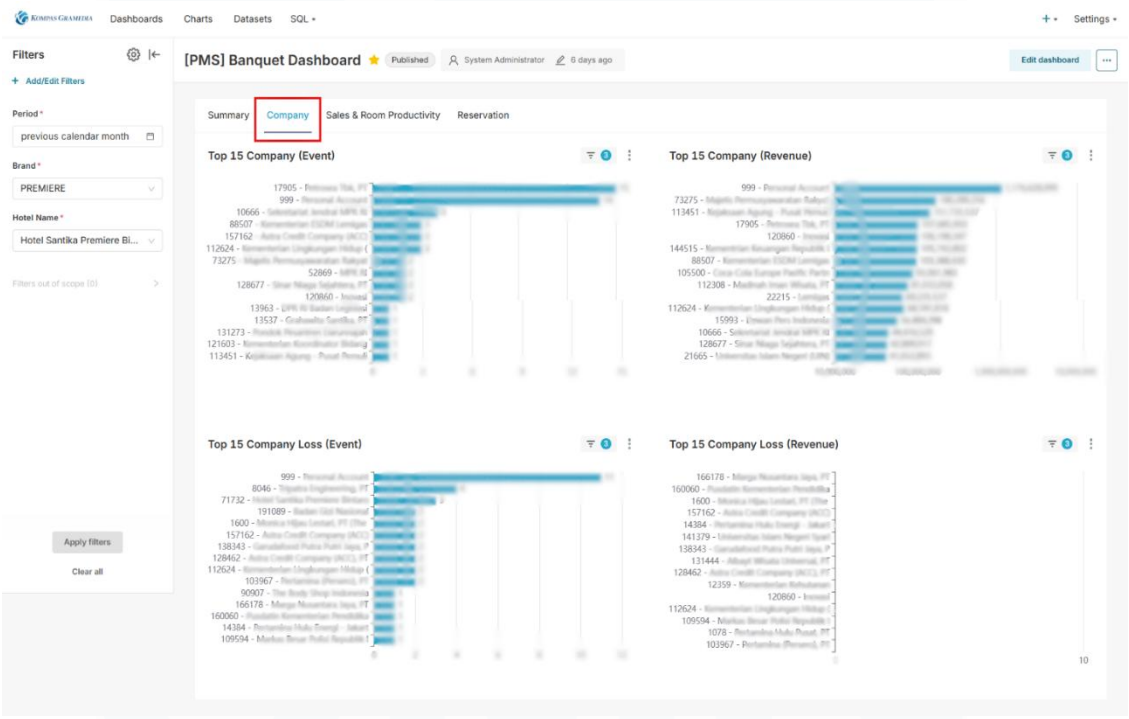
Nama *Chart*	Kalkulasi Metriks	Filter
Top 15 *Company Cancellation* (*Revenue*) (*Bar Chart*)	(SUM(CASE WHEN [redacted] = FALSE THEN [redacted] ELSE 0 END)/1.21) + (SUM(CASE WHEN [redacted] = TRUE THEN [redacted] ELSE 0 END)/1.21) CONCAT(pms_company_id,' - ',left(partner_name, 30))	state = 'release' 2025-05-01 ≤ canceldate < 2... classcode IN ('PREMIERE') hotelname IN ('Hotel Santika Pr...

Proses selanjutnya adalah pembuatan *chart* visual di Apache Superset untuk menyajikan metrik yang telah dihitung ke dalam format yang mudah dianalisis oleh manajemen. Empat visualisasi utama disusun dalam bentuk *bar chart*, yaitu *Top 15 Company* (*Event*), *Top 15 Company* (*Revenue*), *Top 15 Company Cancellation* (*Event*), dan *Top 15 Company Cancellation* (*Revenue*). Setiap *chart* menggunakan sumbu Y yang menggabungkan kode dan nama perusahaan melalui *CONCAT(pms_company_id, ' - ', LEFT(partner_name, 30))* untuk menjaga identifikasi yang ringkas namun tetap informatif. Sumbu X merepresentasikan hasil perhitungan metrik sesuai fokus visualisasi, seperti total *event* atau pendapatan sebelum pajak. Semua *chart* difilter agar hanya menampilkan data dari hotel tertentu dalam periode transaksi atau pembatalan tertentu. Tampilan *bar chart* disusun menurun berdasarkan nilai metrik sehingga menampilkan 15 perusahaan teratas yang paling berkontribusi maupun paling banyak membatalkan *event*.

c. *Output*

Output dari submenu *Company* pada *Banquet Dashboard* terdiri dari empat *bar chart* yang menampilkan 15 perusahaan teratas berdasarkan jumlah *event*, total pendapatan, jumlah pembatalan *event*, dan nilai pembatalan pendapatan. Hasil *Banquet Dashboard* submenu *Company* ditampilkan pada Gambar 3.86 yang dirancang untuk memberikan perbandingan kinerja antar perusahaan mitra dalam menggunakan fasilitas ruang acara hotel, baik dari sisi kontribusi maupun potensi kerugian. Visualisasi ini membantu manajemen mengidentifikasi perusahaan dengan tingkat utilisasi tinggi maupun frekuensi pembatalan tinggi yang dapat memengaruhi stabilitas pendapatan. Informasi yang tersaji digunakan untuk

mendukung strategi hubungan mitra bisnis, seperti penawaran eksklusif, penyesuaian kebijakan reservasi, atau evaluasi kontrak kerja sama. *Output dashboard* ini akan di-*embed* ke dalam sistem operasional hotel melalui fitur *embedding* Apache Superset, sehingga dapat diakses langsung melalui *Property Management System* (PMS) oleh manajemen dan tim *marketing*, khususnya *Board of Directors* untuk mempercepat pengambilan keputusan berbasis data. Seluruh visualisasi bersifat dinamis dan diperbarui secara berkala mengikuti siklus data bulanan yang diproses melalui *pipeline* ETL, sehingga perubahan performa perusahaan mitra dapat terpantau dan ditindaklanjuti secara tepat waktu.



Gambar 3.86 *Output Banquet Dashboard* Submenu *Company*

3.2.8.3. *Banquet Dashboard* Submenu *Sales & Room Productivity*

Pembangunan submenu *Sales & Room Productivity* pada *Banquet Dashboard* bertujuan untuk mengevaluasi kontribusi individu tenaga penjual serta pemanfaatan ruang meeting terhadap jumlah *event* dan pendapatan *banquet* hotel. Berikut adalah input, *process*, dan *output* dari *Banquet Dashboard* submenu *Sales & Room Productivity*:

a. Input

Pembangunan submenu *Sales & Room Productivity* dalam *Banquet Dashboard* menggunakan dua sumber input berupa *Materialized View* (MV) mv_bqt_eventproductivity dan mv_bqt_roomproductivity yang disimpan dalam *data warehouse* hospitalitydwh. Meskipun mv_bqt_eventproductivity telah dijelaskan sebelumnya pada submenu *Summary*, penggunaannya pada submenu ini difokuskan untuk mengevaluasi kontribusi individu *sales person* dan ruang *meeting* terhadap jumlah *event* dan pendapatan, berbeda dari fokus analisis pada performa umum *banquet* hotel. Kolom seperti *employee_name* dan *is_salesperson* digunakan untuk mengidentifikasi peran tenaga penjual dalam proses reservasi. Sementara itu, mv_bqt_roomproductivity dengan *query* seperti pada Gambar 3.87 secara khusus digunakan untuk mengukur tingkat pemanfaatan ruang *meeting*, yang dibentuk melalui proses *LEFT JOIN* dari beberapa tabel seperti odoo_bqtreservationevent, odoo_bqtfunctionroom, odoo_bqtreservation, dan odoo_bqtreservationbill, serta informasi tambahan dari active_hotel dan odoo_catalogeventtype.

```
CREATE MATERIALIZED VIEW public.mv_bqt_roomproductivity
 TABLESPACE pg_default
 AS SELECT re.[illegible],
    re.[illegible],
    re.[illegible],
    re.[illegible],
    re.[illegible],
    re.[illegible],
    re.[illegible],
    re.[illegible],
    re.[illegible],
    fr.[illegible] AS [illegible],
    fr.[illegible] AS [illegible],
    fr.[illegible],
    r.[illegible],
    r.[illegible],
    ah.[illegible],
    ah.[illegible],
    ah.[illegible],
    et.[illegible],
    rb.[illegible],
    rb.[illegible]
   FROM odoo_bqtreservationevent re
     LEFT JOIN odoo_bqtfunctionroom fr ON [illegible]
     LEFT JOIN odoo_bqtreservation r ON [illegible]
     LEFT JOIN odoo_bqtreservationbill rb ON [illegible] AND [illegible]
     LEFT JOIN active_hotel ah ON [illegible]
     LEFT JOIN odoo_catalogeventtype et ON [illegible]
 WITH DATA;
```

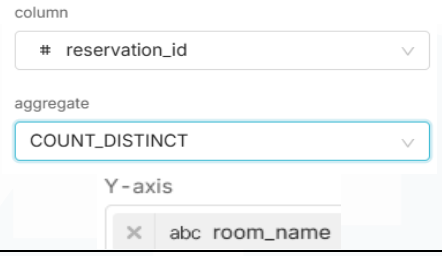
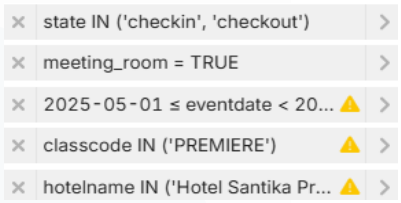
Gambar 3.87 *Query Materialized View* mv_bqt_roomproductivity

b. *Process*

Perancangan visualisasi pada submenu *Sales & Room Productivity* memerlukan tahapan pemrosesan data berbasis logika agregasi dan kalkulasi metrik dari tabel hasil proses ETL. Daftar kalkulasi metrik dan filter yang digunakan untuk membuat *Banquet Dashboard* submenu *Sales & Room Productivity* disajikan pada Tabel 3.8. *Dataset* mv_bqt_eventproductivity digunakan untuk menilai kontribusi tenaga penjual, sementara *dataset* mv_bqt_roomproductivity untuk mengukur pemanfaatan ruang *meeting*, keduanya berdasarkan jumlah reservasi dan pendapatan. Metrik Total *Event* per *Salesperson* dihitung menggunakan *COUNT(DISTINCT reservation_id)* yang dikategorikan berdasarkan *employee_name*, dengan filter status reservasi (*state*) berupa *checkout* dan *checkin*, serta hanya mencakup data dengan *is_salesperson = TRUE*. Metrik Total *Revenue* per *Salesperson* berasal dari penjumlahan lima komponen biaya yang masing-masing dibagi 1.21 untuk menormalkan nilai tanpa pajak sesuai standar akuntansi KG *Hospitality*. Adapun metrik Total *Event* per *RoomMeeting* dihitung menggunakan *COUNT(DISTINCT reservation_id)* dari mv_bqt_roomproductivity, dikelompokkan berdasarkan *room_name*, dan difilter dengan *meeting_room = TRUE* serta *eventdate* sesuai periode tertentu. Seluruh metrik difilter secara konsisten berdasarkan tanggal transaksi, tipe hotel, dan nama hotel.

Tabel 3.8 Rincian Metriks dan Filter pada *Banquet Dashboard* Submenu *Sales & Room*

Nama *Chart*	Kalkulasi Metriks	Filter
Total Event / Sales Person (*Bar Chart*)	column # reservation_id aggregate COUNT_DISTINCT Y-axis abc employee_name	state IN ('checkout', 'checkin') is_salesperson IN (TRUE) 2025-05-01 ≤ working_date ... classcode IN ('PREMIERE') hotelname IN ('Hotel Santika Pr...
Total Revenue / SalesPerson (*Bar Chart*)	((SUM( )/1.21) + (SUM( )/1.21) + (SUM( )/1.21) + (SUM( )/1.21) + (SUM( )/1.21)) Y-axis abc employee_name	state IN ('checkout', 'checkin') is_salesperson IN (TRUE) 2025-05-01 ≤ working_date ... classcode IN ('PREMIERE') hotelname IN ('Hotel Santika Pr...

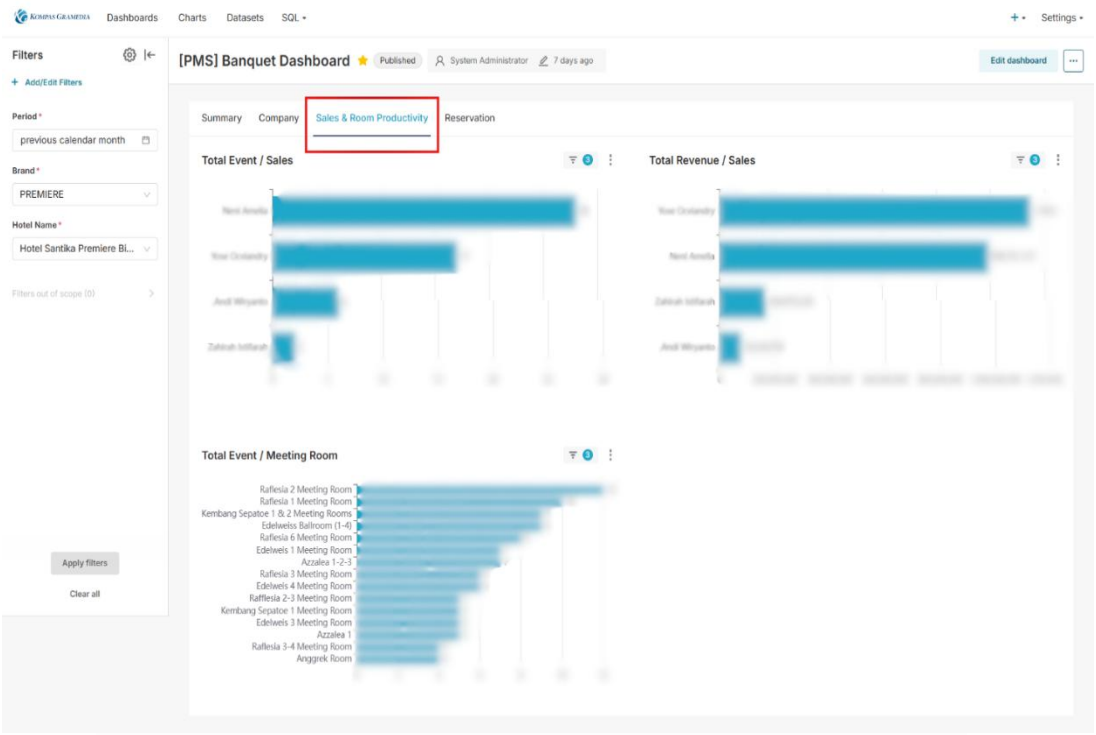
Nama *Chart*	Kalkulasi Metriks	Filter
Total Event / RoomMeeting (*Bar Chart*)		

Proses selanjutnya adalah pembuatan visualisasi dalam bentuk *bar chart* di Apache Superset untuk menyajikan hasil kalkulasi metrik dari setiap kategori secara informatif dan mudah dipahami. Tiga visualisasi utama dikembangkan dalam submenu ini, yaitu Total *Event* per *Salesperson*, Total *Revenue* per *Salesperson*, dan Total *Event* per *RoomMeeting*, dengan sumbu Y mewakili dimensi kategori seperti *employee_name* atau *room_name*, dan sumbu X menampilkan nilai metrik utama. Pemilihan *bar chart* didasarkan pada kemampuannya dalam memperlihatkan perbandingan antar entitas secara jelas, seperti mengidentifikasi tenaga penjual dengan performa terbaik atau ruang *meeting* yang paling sering digunakan. Setiap visualisasi dilengkapi dengan filter dinamis di Superset untuk menyaring data berdasarkan periode transaksi (*working_date* atau *eventdate*), status reservasi (*state*), jenis hotel (*classcode*), dan nama hotel (*hotelname*). Secara default, tampilan *bar chart* diurutkan secara menurun berdasarkan nilai metrik tertinggi, sehingga memudahkan identifikasi kontributor utama dalam pendapatan dan pemanfaatan aset.

c. *Output*

Output dari submenu *Sales & Room Productivity* berupa tiga visualisasi *bar chart* yang menyajikan kinerja tenaga penjual dan tingkat pemanfaatan ruang *meeting* berdasarkan data aktual periode tertentu. Hasil *Banquet Dashboard* submenu *Sales & Room Productivity* ditampilkan pada Gambar 3.88. Visualisasi pertama *Total Event per Salesperson*, menampilkan jumlah reservasi yang berhasil ditangani oleh masing-masing *sales* untuk mengukur produktivitas individu dalam tim penjualan. Visualisasi kedua *Total Revenue per Salesperson*, menunjukkan kontribusi pendapatan bersih

tiap *sales* terhadap total pendapatan *banquet* hotel. Visualisasi ketiga *Total Event per RoomMeeting*, menggambarkan frekuensi penggunaan masing-masing ruang *meeting* sebagai lokasi acara. Ketiga chart ini ditampilkan secara langsung dalam halaman submenu Apache Superset dan diperbarui otomatis setiap awal bulan berdasarkan filter *working_date* dan *eventdate*. *Output dashboard* ini akan di-*embed* ke dalam sistem operasional hotel melalui fitur *embedding* Apache Superset, sehingga dapat diakses langsung melalui *Property Management System* (PMS) oleh *Board of Directors*, *General* Manajer, dan tim operasional *banquet* untuk keperluan *monitoring* performa, evaluasi target bulanan, serta perumusan strategi penyewaan ruangan dan alokasi sumber daya di periode berikutnya.



Gambar 3.88 *Output Banquet Dashboard* Submenu *Sales & Room Productivity*

3.2.8.4. *Banquet Dashboard* Submenu *Reservation*

Pembangunan submenu *Reservation* pada *Banquet Dashboard* bertujuan untuk menyajikan analisis tentang pola reservasi dan pembatalan harian, serta jeda waktu antara pemesanan dan pelaksanaan acara. Berikut

adalah input, *process*, dan *output* dari *Banquet Dashboard* submenu *Reservation*:

a. Input

Pembangunan submenu *Reservation* dalam *Banquet Dashboard* turut menggunakan dua *Materialized View* (MV), yaitu mv_bqt_eventproductivity dengan *query* seperti pada Gambar 3.83 dan mv_bqt_cancellation dengan *query* seperti pada Gambar 3.84 yang sebelumnya telah dijelaskan pada bagian *Banquet Dashboard* Submenu *Summary*. Meskipun menggunakan sumber *dataset* yang sama, perbedaan terletak pada fokus analisis, fungsi visualisasi, serta kolom yang dimanfaatkan untuk membentuk metrik dan dimensi pada masing-masing *chart*. Pada submenu ini, mv_bqt_eventproductivity dimanfaatkan untuk menganalisis pola reservasi harian serta menghitung rentang waktu antara tanggal pemesanan dan pelaksanaan acara menggunakan kolom *reservation_date* dan *leadtime*. Sementara itu, mv_bqt_cancellation digunakan untuk memantau tren pembatalan harian melalui kolom *canceldate* dan status *state = 'release'* untuk mengukur volatilitas serta potensi kerugian akibat pembatalan reservasi. Visualisasi yang ditampilkan dalam submenu ini hanya mengambil atribut yang relevan guna menyusun metrik *Daily Reservation*, *Daily Cancellation*, dan *Reservation to Check In* (*Days*).

b. *Process*

Perancangan visualisasi pada submenu *Reservation* memerlukan tahapan pemrosesan data berbasis logika agregasi dan kalkulasi metrik dari tabel hasil proses ETL. Daftar kalkulasi metrik dan filter yang digunakan untuk membuat *Banquet Dashboard* submenu *Reservation* disajikan pada Tabel 3.9. Dua *dataset* yang digunakan yaitu mv_bqt_eventproductivity dan mv_bqt_cancellation, masing-masing menyimpan informasi terkait event yang terlaksana dan yang dibatalkan. Metrik pertama yaitu jumlah reservasi harian, dihitung menggunakan *COUNT(DISTINCT reservation_id)* yang

dikelompokkan berdasarkan *reservation_date*, dengan filter *classcode*, *hotelname*, dan *period* tertentu. Metrik kedua jumlah pembatalan harian, dihitung dari dataset mv_bqt_cancellation dengan metode serupa berdasarkan canceldate, difokuskan pada status *state = 'release'*. Metrik ketiga *Reservation* to *Check In (Days)*, digunakan untuk mengukur jeda waktu antara tanggal reservasi dan pelaksanaan acara berdasarkan kolom leadtime, dihitung dengan *COUNT(DISTINCT reservation_id)* dan difilter untuk status selain *release* dalam periode reservation_date tertentu. Seluruh metrik dihitung langsung di Apache Superset melalui *custom metric builder* dan dirancang untuk merepresentasikan performa operasional dari sisi perilaku reservasi, tren pembatalan, serta kecepatan pengambilan keputusan pelanggan.

Tabel 3.9 Rincian Metriks dan Filter pada *Banquet Dashboard* Submenu *Reservation*

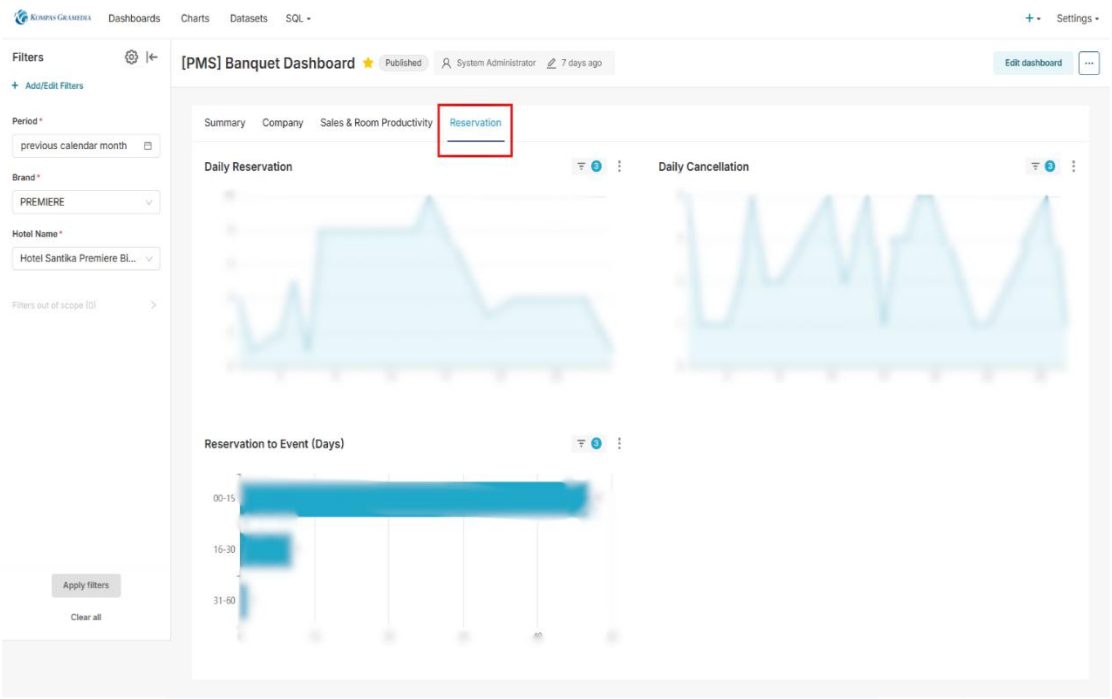
Nama *Chart*	Kalkulasi Metriks	Filter
Daily Reservation (*Area Chart*)	column: # reservation_id aggregate: COUNT_DISTINCT X-axis: reservation_date	2025-05-01 ≤ reservation_date ... classcode IN ('PREMIERE') hotelname IN ('Hotel Santika Pre...
Daily Cancellation (*Area Chart*)	column: # reservation_id aggregate: COUNT_DISTINCT X-axis: canceldate	state = 'release' 2025-05-01 ≤ canceldate < 2... classcode IN ('PREMIERE') hotelname IN ('Hotel Santika Pr...
Reservation to Check In (*Days*) (*Bar Chart*)	column: # reservation_id aggregate: COUNT_DISTINCT Y-axis: abc leadtime	state <> 'release' 2025-05-01 ≤ reservation_da... classcode IN ('PREMIERE') hotelname IN ('Hotel Santika Pr...

Proses selanjutnya adalah pembuatan chart visual di Apache Superset untuk menyajikan metrik yang telah dihitung ke dalam format yang mudah dianalisis oleh manajemen. Visualisasi *Daily Reservation* dan *Daily*

Cancellation menggunakan *Area Chart* dengan sumbu X berupa *reservation_date* dan *canceldate*, serta sumbu Y menunjukkan jumlah reservasi atau pembatalan per hari, untuk menggambarkan fluktuasi permintaan dan dinamika pembatalan ruang acara dalam periode tertentu. *Visualisasi Reservation to Check In* (*Days*) berbentuk *Bar Chart* dengan sumbu Y berupa *leadtime* dan sumbu X merepresentasikan jumlah reservasi berdasarkan jeda waktu antara pemesanan dan pelaksanaan event. Seluruh grafik dibangun secara manual melalui Superset. Masing-masing visualisasi dilengkapi dengan filter interaktif seperti periode tanggal, nama hotel, *classcode*, dan status reservasi untuk mendukung analisis yang fleksibel dan responsif.

c. *Output*

Output dari submenu *Reservation* pada *Banquet Dashboard* seperti yang terlihat pada Gambar 3.89, berupa tampilan visual interaktif yang menyajikan tiga grafik utama: *Daily Reservation, Daily Cancellation*, dan *Reservation to Check In (Days)* dalam format *Area Chart* dan *Bar Chart*. Grafik *Daily Reservation* menampilkan tren volume pemesanan ruang acara hotel per hari, *Daily Cancellation* menggambarkan pola pembatalan dalam periode yang sama, sementara *Reservation to Check In (Days)* menunjukkan distribusi jeda waktu antara pemesanan dan pelaksanaan acara. Seluruh visualisasi merupakan hasil kalkulasi metrik terhadap data reservasi dan pembatalan yang telah diproses melalui *pipeline* ETL sebelumnya. *Output* ini digunakan oleh manajemen hotel untuk memahami perilaku pelanggan serta mengevaluasi kesiapan internal dalam merespons perubahan jadwal acara secara mendadak. *Output dashboard* ini akan di-*embed* ke dalam sistem operasional hotel melalui fitur *embedding* Apache Superset untuk mendukung pengambilan keputusan harian yang cepat dan berbasis data.



Gambar 3.89 *Output Banquet Dashboard* Submenu *Reservation*

3.2.9. *Developing Dashboards – Hospitality-Analytics* CRM Menu *Dashboard*

Pengembangan *Hospitality-Analytics* CRM Menu *Dashboard* merupakan kelanjutan dari proses integrasi data melalui *pipeline* ETL yang berfungsi menyalin dan menyelaraskan data dari sistem *Customer Relationship Management* (CRM) ke dalam *data warehouse* hospitalitydwh. *Pipeline* ini dibangun dengan Azure Data Factory dan memproses data dari *Property Management System* (PMS) sebagai sumber utama, karena mayoritas *dataset* dalam *dashboard* CRM berasal dari PMS. Perpindahan *dashboard* ke dalam PMS bertujuan untuk mengintegrasikan seluruh fungsi analitik ke dalam satu sistem terpadu yang digunakan oleh berbagai unit operasional hotel karena CRM hanya dapat diakses oleh tim *sales*, sementara PMS digunakan oleh *front office*, manajer operasional, dan pihak hotel lainnya. Untuk mendukung kebutuhan akses lintas fungsi ini, *dashboard* CRM direplikasi dan diintegrasikan langsung ke dalam PMS melalui *tools* Apache Superset yang terhubung dengan *data warehouse* hospitalitydwh. Proses ini mencakup pembuatan visualisasi, pembangunan *materialized views* (MV), perhitungan

metrik analitis, dan desain tampilan *dashboard*. Semua tahap ini memastikan bahwa hasil migrasi *dashboard* tetap akurat, konsisten, dan relevan bagi pengguna PMS dalam pengambilan keputusan. Rincian fokus tiap *dashboard* dalam proyek CRM Menu *Dashboard* disajikan pada Tabel 3.10.

Tabel 3.10 Rincian Fokus *Dashboard* dalam Proyek CRM Menu *Dashboard*

Dashboard	***Dataset***	**Tujuan Visualisasi**
Market Segment by Brand	mv_market_segment_last year	Membandingkan performa jumlah kamar, pendapatan kamar, dan rata-rata harga kamar berdasarkan *market segment* hotel, serta menganalisis tren dan proporsi historis maupun target anggaran
Market Segment with Budget	mv_market_segment_last year	Membandingkan kinerja okupansi, pendapatan, dan tarif rata-rata kamar berdasarkan segmentasi pasar hotel terhadap target anggaran dan capaian historis
Average Room Rate Statistic	mv_manager_report_with_budget_lastyear	Menganalisis perbandingan tarif rata-rata kamar berdasarkan *brand* hotel terhadap target anggaran dan capaian historis dalam periode tertentu
Occupancy Statistic	mv_manager_report_with_budget_lastyear	Menganalisis tingkat hunian kamar hotel berdasarkan *brand* dengan membandingkan capaian aktual terhadap target anggaran dan performa tahun sebelumnya
Hotel Revenue Statistic	mv_manager_report_with_budget_lastyear	Membandingkan pendapatan hotel aktual, anggaran, dan capaian tahun sebelumnya, serta menganalisis pertumbuhan dan pencapaian anggaran
Room Revenue Statistic	mv_manager_report_with_budget_lastyear	Membandingkan pendapatan kamar aktual, anggaran, dan capaian tahun sebelumnya, serta menghitung pertumbuhan dan pencapaian anggaran
Geographical Origin Of Business	mv_geographical_origin	Membandingkan performa *room night* berdasarkan asal geografis (Indonesia dan Internasional) dengan data aktual dan perbandingan tahun sebelumnya

3.2.9.1. CRM Menu *Dashboard – Market Segment by Brand*

Dashboard *Market Segment by Brand* dirancang untuk menyajikan performa bisnis hotel berdasarkan segmentasi pasar yang diklasifikasikan

menurut *brand*. Penyajian informasi dalam *dashboard* ini dibagi ke dalam empat submenu utama agar analisis dapat dilakukan secara lebih terfokus dan menyeluruh terhadap aspek volume, pendapatan, tarif rata-rata, serta rincian detail data dari ketiga aspek tersebut. Keempat submenu tersebut terdiri dari *Room Night*, *Room Revenue*, *Average Room Rate*, dan *Detail*.

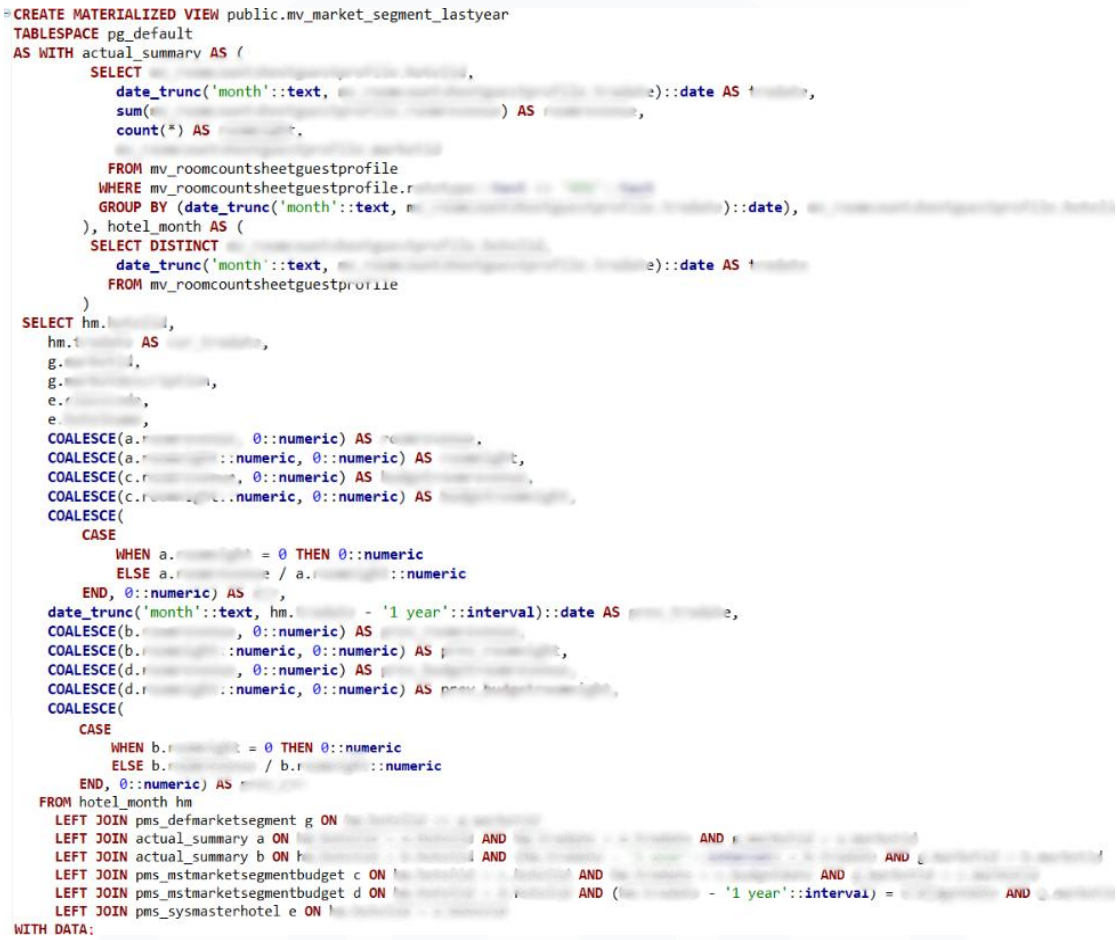
1. *Market Segment by Brand* Submenu *Room Night*

Pembangunan submenu *Room Night* pada *dashboard Market Segment by Brand* bertujuan untuk menyajikan visualisasi interaktif terkait distribusi jumlah kamar terjual berdasarkan segmentasi pasar hotel, guna memberikan pandangan strategis terhadap dinamika volume tamu dari berbagai sumber pasar. Berikut adalah input, *process*, dan *output* dari *Market Segment by Brand* submenu *Room Night*:

a. Input

Penyusunan visualisasi pada submenu *Room Night* dimulai dengan penyediaan input berupa *dataset* yang telah diolah dalam bentuk *materialized view* bernama mv_market_segment_lastyear dengan *query* seperti pada Gambar 3.90 yang memuat data historis, aktual, dan target anggaran berdasarkan agregasi bulanan jumlah kamar terjual (*room night*) per segmen pasar hotel. Data utama berasal dari tabel mv_roomcountsheetguestprofile yang merekam transaksi kamar berdasarkan tanggal dan segmentasi tamu, lalu diolah menggunakan *function date_trunc* dan *count(*)* untuk menghitung total kamar yang dihuni setiap bulan. Informasi ini diperkaya dengan data anggaran dari tabel pms_mstmarketsegmentbudget untuk menghitung target kinerja, serta referensi segmentasi pasar dari pms_defmarketsegment untuk klasifikasi *market segment*. Untuk memungkinkan perbandingan kinerja tahun berjalan dengan periode yang sama di tahun sebelumnya, dilakukan *join* dengan parameter waktu yang dimundurkan satu tahun menggunakan *function* interval PostgreSQL. Struktur hotel seperti *hotelname* dan

classcode diambil dari tabel pms_sysmasterhotel untuk mendukung fitur penyaringan visualisasi.



```
CREATE MATERIALIZED VIEW public.mv_market_segment_lastyear
TABLESPACE pg_default
AS WITH actual_summary AS (
         SELECT
            date_trunc('month'::text, )::date AS ,
            sum( ) AS ,
            count(*) AS ,
           FROM mv_roomcountsheetguestprofile
          WHERE mv_roomcountsheetguestprofile.
          GROUP BY (date_trunc('month'::text, )::date),
        ), hotel_month AS (
         SELECT DISTINCT
            date_trunc('month'::text, )::date AS
           FROM mv_roomcountsheetguestprofile
        )
 SELECT hm.,
    hm. AS ,
    g.,
    g.,
    e.,
    e.,
    COALESCE(a. 0::numeric) AS ,
    COALESCE(a. ::numeric, 0::numeric) AS ,
    COALESCE(c., 0::numeric) AS ,
    COALESCE(c. ::numeric, 0::numeric) AS ,
    COALESCE(
        CASE
            WHEN a. = 0 THEN 0::numeric
            ELSE a. / a. ::numeric
        END, 0::numeric) AS ,
    date_trunc('month'::text, hm. - '1 year'::interval)::date AS ,
    COALESCE(b., 0::numeric) AS ,
    COALESCE(b. ::numeric, 0::numeric) AS ,
    COALESCE(d., 0::numeric) AS ,
    COALESCE(d. ::numeric, 0::numeric) AS ,
    COALESCE(
        CASE
            WHEN b. = 0 THEN 0::numeric
            ELSE b. / b. ::numeric
        END, 0::numeric) AS
   FROM hotel_month hm
     LEFT JOIN pms_defmarketsegment g ON
     LEFT JOIN actual_summary a ON  AND  AND
     LEFT JOIN actual_summary b ON  AND  AND
     LEFT JOIN pms_mstmarketsegmentbudget c ON  AND  AND
     LEFT JOIN pms_mstmarketsegmentbudget d ON  AND ( - '1 year'::interval) =  AND
     LEFT JOIN pms_sysmasterhotel e ON
WITH DATA;
```

Gambar 3.90 *Query Materialized View* mv_market_segment_lastyear

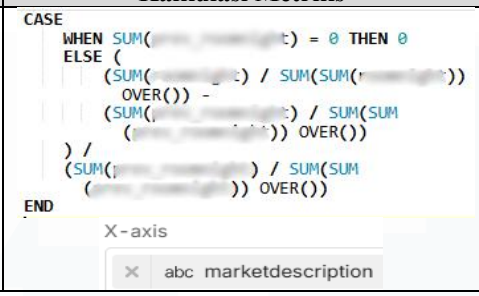
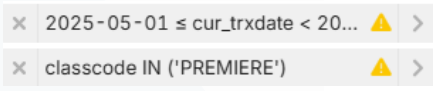
b. *Process*

Perancangan visualisasi pada submenu *Room Night* memerlukan tahapan pemrosesan data berbasis logika agregasi dan kalkulasi metrik dari tabel hasil proses ETL. Daftar kalkulasi metrik dan filter yang digunakan untuk membuat *Market Segment by Brand* Submenu *Room Night* disajikan pada Tabel 3.11. *Dataset* yang digunakan adalah mv_market_segment_lastyear yang telah dibentuk dalam tahap sebelumnya. Proses dimulai dengan agregasi data *roomnight*, *budgetroomnight*, dan *prev_roomnight* menggunakan *function SUM()* untuk menghasilkan metrik *Actual*, *Budget*, dan *Last Year*. Selain metrik tersebut,

dihitung pula metrik pertumbuhan dengan rumus *((SUM(roomnight) - SUM(prev_roomnight)) / SUM(prev_roomnight))*, serta rasio kontribusi masing-masing segmen terhadap total *roomnight* menggunakan *SUM(roomnight) / SUM(SUM(roomnight)) OVER ()*. Perbandingan rasio antar tahun digunakan untuk menghitung *growth ratio* yang menunjukkan perubahan proporsi segmen dibandingkan tahun sebelumnya. Seluruh metrik dihitung langsung di Apache Superset melalui fitur *custom metric* dengan filter cur_trxdate difokuskan pada periode tertentu dan penyaringan berdasarkan *brand* hotel (*classcode*).

Tabel 3.11 Rincian Metriks dan Filter pada *Market Segment by Brand* Submenu *Room Night*

Nama *Chart*	Kalkulasi Metriks	Filter
Room Night (*Bar Chart*)	column ACTUAL # roomnight aggregate SUM column BUDGET # budgetroomnight aggregate SUM column LAST YEAR # prev_roomnight aggregate SUM X-axis abc marketdescription	2025-05-01 ≤ cur_trxdate < 20... classcode IN ('PREMIERE')
Growth Room Night (*Bar Chart*)	CASE WHEN SUM([redacted]) = 0 THEN 0 ELSE (SUM([redacted]) - SUM([redacted])) / SUM([redacted]) END X-axis abc marketdescription	2025-05-01 ≤ cur_trxdate < 20... classcode IN ('PREMIERE')
Ratio Room Night (*Bar Chart*)	ACTUAL CASE WHEN SUM(SUM([redacted])) OVER () = 0 THEN 0 ELSE SUM([redacted]) / NULLIF(SUM(SUM([redacted])) OVER (), 0) END LAST YEAR CASE WHEN SUM(SUM([redacted])) OVER () = 0 THEN 0 ELSE SUM([redacted]) / NULLIF(SUM(SUM([redacted])) OVER (), 0) END X-axis abc marketdescription	2025-05-01 ≤ cur_trxdate < 20... classcode IN ('PREMIERE')

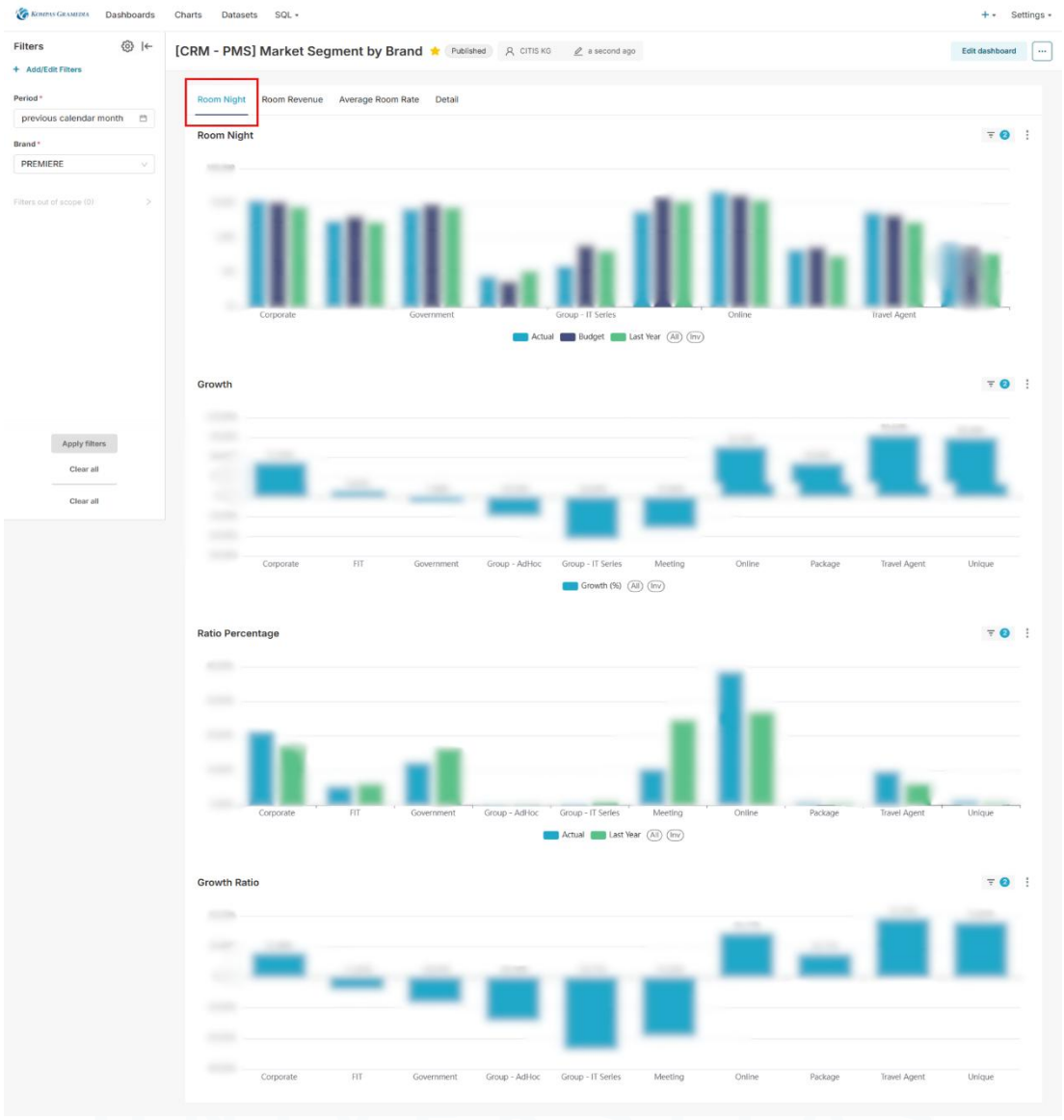
Nama *Chart*	Kalkulasi Metriks	Filter
Growth Ratio Room Night (*Bar Chart*)		

Proses selanjutnya adalah perancangan visualisasi dalam bentuk *bar chart* untuk merepresentasikan metrik yang telah dikalkulasi dalam submenu *Room Night*. Dalam submenu ini, dirancang empat *chart* yakni *Market Segment Room Night*, *Growth Room Night*, *Ratio Room Night*, dan *Growth Ratio Room Night* yang masing-masing menampilkan total volume, tingkat pertumbuhan, distribusi kontribusi, serta perubahan proporsi antar segmen pasar. Seluruh chart menggunakan dimensi *marketdescription* sebagai sumbu X untuk mengelompokkan data berdasarkan nama segmen pasar, dengan tipe *bar chart* dipilih karena lebih jelas dalam memperlihatkan perbandingan nilai antar kategori. Filter yang diterapkan mencakup rentang tanggal transaksi tertentu dan *classcode* tertentu.

c. *Output*

Submenu *Room Night* pada *dashboard Market Segment by Brand* menyajikan *output* visualisasi berbentuk *bar chart* yang menampilkan distribusi jumlah kamar terjual berdasarkan segmentasi pasar hotel dalam periode waktu tertentu. Hasil *Market Segment by Brand* submenu *Room Night* ditampilkan pada Gambar 3.91 yang mencakup empat aspek utama yakni nilai aktual, target anggaran, pencapaian tahun sebelumnya, serta persentase pertumbuhan dan kontribusi masing-masing segmen terhadap total *room night*. *Chart* seperti *Market Segment Room Night*, *Growth Room Night*, *Ratio Room Night*, dan *Growth Ratio Room Night* dirancang untuk memberikan gambaran menyeluruh tentang dinamika perubahan volume tamu dari berbagai segmen. Seluruh *output* disajikan secara interaktif melalui Apache Superset dan akan di-*embed* langsung ke dalam sistem

Property Management System (PMS) hotel. Informasi ini menjadi acuan utama bagi manajemen dan tim operasional dalam menganalisis pergerakan volume kamar berdasarkan sumber pasar, mengidentifikasi tren pertumbuhan atau penurunan, serta menyesuaikan strategi penjualan dan alokasi kamar untuk periode mendatang.



Gambar 3.91 *Output Market Segment by Brand* Submenu *Room Night*

2. *Market Segment by Brand* Submenu *Room Revenue*

Pembangunan submenu *Room Revenue* pada dashboard *Market Segment by Brand* bertujuan untuk menampilkan performa pendapatan kamar

hotel berdasarkan segmentasi pasar secara komprehensif melalui visualisasi interaktif. Berikut adalah input, *process*, dan *output* dari *Market Segment by Brand* submenu *Room Revenue*:

a. Input

Pembangunan submenu *Room Revenue* dalam *dashboard Market Segment by Brand* menggunakan *dataset* yang sama dengan submenu *Room Night*, yaitu *materialized view* bernama mv_market_segment_lastyear yang dibentuk melalui proses ETL dan *query* SQL seperti pada Gambar 3.90. Meskipun *dataset* yang digunakan identik, perbedaan terletak pada fokus visualisasi, fungsi analitis, dan kolom yang digunakan dalam pembentukan metrik. Submenu *Room Revenue* berfokus pada pendapatan kamar, sehingga menggunakan kolom *roomrevenue*, *budgetroomrevenue*, dan *prev_roomrevenue*. *Dataset* ini juga mencakup nilai target anggaran dan pencapaian historis untuk analisis tren pendapatan berdasarkan segmentasi pasar. Informasi mengenai tanggal transaksi (cur_trxdate) dan *brand* hotel (*classcode*) tetap digunakan untuk mendukung fitur penyaringan data.

b. *Process*

Perancangan visualisasi pada submenu *Room Revenue* memerlukan tahapan pemrosesan data berbasis logika agregasi dan kalkulasi metrik dari tabel hasil proses ETL. Daftar kalkulasi metrik dan filter yang digunakan untuk membuat *dashboard Market Segment by Brand* submenu *Room Revenue* disajikan pada Tabel 3.12. *Dataset* yang digunakan yaitu mv_market_segment_lastyear. Perhitungan metrik dilakukan langsung di Apache Superset menggunakan fitur *custom metric* dengan kolom *roomrevenue*, *budgetroomrevenue*, dan *prev_roomrevenue* sebagai dasar agregasi. Nilai aktual pendapatan dihitung dengan *SUM(roomrevenue)*, target anggaran dengan *SUM(budgetroomrevenue)*, dan nilai historis dengan *SUM(prev_roomrevenue)*. Selain metrik dasar, disusun metrik pertumbuhan menggunakan rumus *(SUM(roomrevenue) - SUM(prev_roomrevenue)) / SUM(prev_roomrevenue)*, serta analisis

kontribusi tiap segmen melalui *SUM(roomrevenue) / SUM(SUM(roomrevenue)) OVER ()*. Perbandingan kontribusi antar tahun digunakan untuk menghitung *growth ratio*. Seluruh kalkulasi difilter berdasarkan periode transaksi tertentu dan kode *brand* hotel tertentu untuk menghasilkan visualisasi yang lebih kontekstual dan terfokus.

Tabel 3.12 Rincian Metriks dan Filter pada *Market Segment by Brand* Submenu *Room Revenue*

Nama *Chart*	Kalkulasi Metriks	Filter
Room Revenue (*Bar Chart*)	column ACTUAL: # roomrevenue; aggregate: SUM column BUDGET: # budgetroomrevenue; aggregate: SUM column LAST YEAR: # prev_roomrevenue; aggregate: SUM X-axis: abc marketdescription	2025-05-01 ≤ cur_trxdate < 20... classcode IN ('PREMIERE')
Growth Room Revenue (*Bar Chart*)	CASE SUM([prev_roomrevenue]) WHEN 0 THEN 0 ELSE (SUM([roomrevenue]) - SUM([prev_roomrevenue])) / SUM([prev_roomrevenue]) END X-axis: abc marketdescription	2025-05-01 ≤ cur_trxdate < 20... classcode IN ('PREMIERE')
Ratio Room Revenue (*Bar Chart*)	ACTUAL: CASE WHEN SUM(SUM([roomrevenue])) OVER () = 0 THEN 0 ELSE SUM([roomrevenue]) / NULLIF(SUM(SUM([roomrevenue])) OVER (), 0) END LAST YEAR: CASE WHEN SUM(SUM([prev_roomrevenue])) OVER () = 0 THEN 0 ELSE SUM([prev_roomrevenue]) / NULLIF(SUM(SUM([prev_roomrevenue])) OVER (), 0) END X-axis: abc marketdescription	2025-05-01 ≤ cur_trxdate < 20... classcode IN ('PREMIERE')
Growth Ratio Room Revenue (*Bar Chart*)	CASE WHEN SUM([prev_roomrevenue]) = 0 THEN 0 ELSE ((SUM([roomrevenue]) / SUM(SUM([roomrevenue])) OVER()) - (SUM([prev_roomrevenue]) / SUM(SUM([prev_roomrevenue])) OVER())) / (SUM([prev_roomrevenue]) / SUM(SUM([prev_roomrevenue])) OVER()) END X-axis: abc marketdescription	2025-05-01 ≤ cur_trxdate < 20... classcode IN ('PREMIERE')

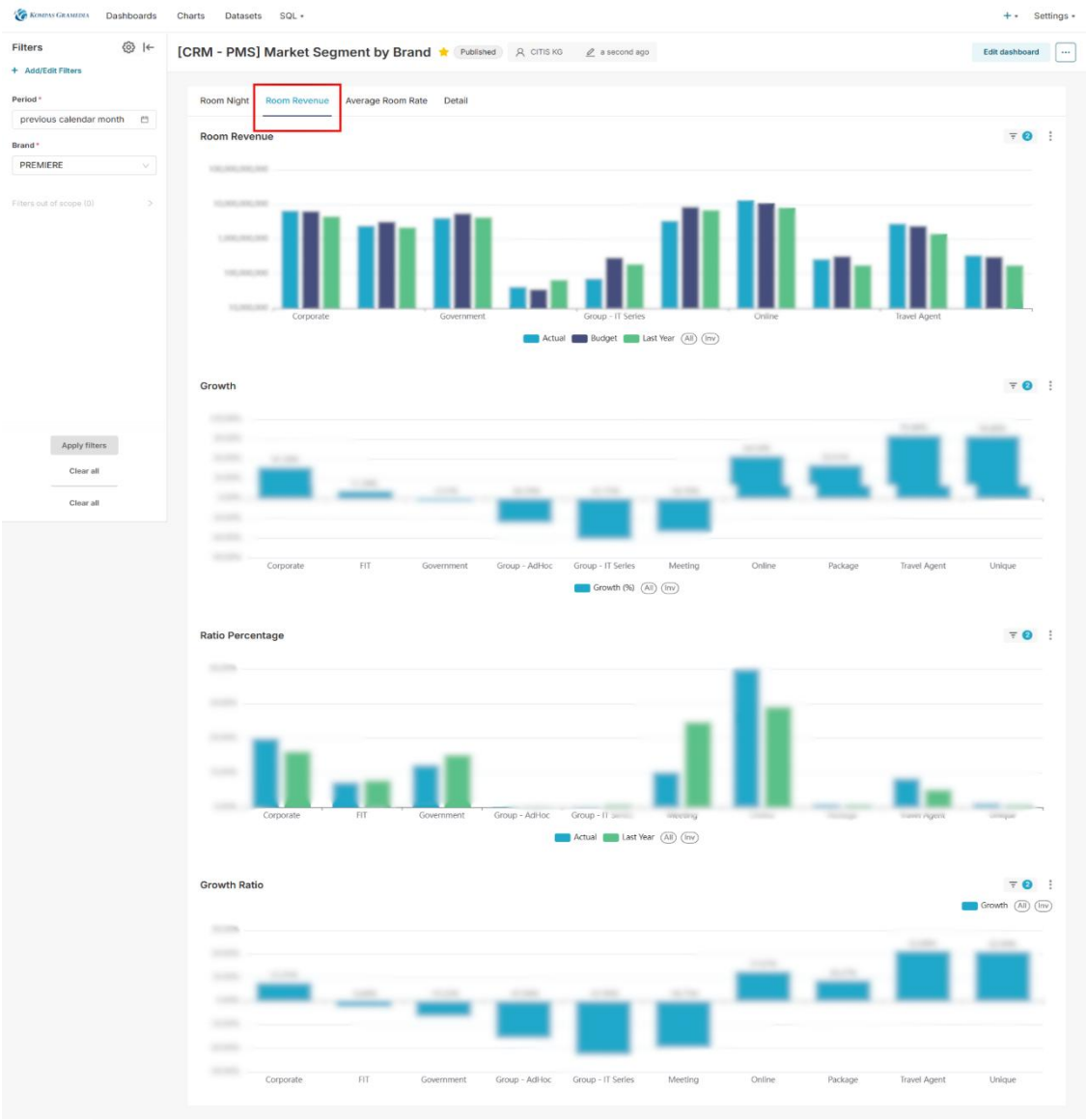
Proses selanjutnya adalah pembangunan visualisasi dalam bentuk *bar chart* untuk merepresentasikan hasil kalkulasi metrik pendapatan

berdasarkan segmentasi pasar. Dalam submenu ini, disusun empat *chart* utama yakni *Market Segment Room Revenue*, *Growth Room Revenue*, *Ratio Room Revenue*, dan *Growth Ratio Room Revenue* yang masing-masing memiliki fungsi analitis spesifik. *Chart* pertama menampilkan perbandingan nilai aktual, target anggaran, dan capaian tahun sebelumnya, *chart* kedua menunjukkan persentase pertumbuhan pendapatan antar segmen, *chart* ketiga menggambarkan proporsi kontribusi setiap segmen terhadap total pendapatan kamar, dan *chart* keempat memperlihatkan perubahan rasio kontribusi segmen antara periode berjalan dan periode yang sama tahun sebelumnya. Seluruh *chart* menggunakan dimensi *marketdescription* sebagai sumbu X, dengan jenis visualisasi *bar chart* dipilih karena efektif dalam memperlihatkan perbandingan nilai antar kategori.

c. *Output*

Submenu *Room Revenue* pada *dashboard Market Segment by Brand* menghasilkan *output* visualisasi berupa *bar chart* seperti yang terlihat pada Gambar 3.92 yang menyajikan kinerja pendapatan kamar berdasarkan segmentasi pasar hotel dalam periode tertentu. Hasil visualisasi ditampilkan melalui empat *chart*, yaitu *Market Segment Room Revenue*, *Growth Room Revenue*, *Ratio Room Revenue*, dan *Growth Ratio Room Revenue*, yang masing-masing menampilkan nilai aktual, target anggaran, perbandingan tahun sebelumnya, rasio kontribusi segmen, serta persentase pertumbuhan antar segmen. Semua visualisasi menggunakan sumbu X berbasis *marketdescription* untuk membandingkan performa pendapatan tiap segmen secara visual dan mudah dipahami. *Output* ini disusun dalam format interaktif melalui Apache Superset dan difilter berdasarkan tanggal transaksi dan *classcode* hotel. Seluruh hasil visualisasi tersebut akan diintegrasikan ke dalam sistem *Property Management System* (PMS) agar dapat diakses langsung oleh berbagai fungsi operasional hotel, termasuk manajer unit, tim *front office*, hingga pihak manajemen pusat. Informasi yang ditampilkan dalam *dashboard* ini berfungsi sebagai acuan utama

dalam mengevaluasi kontribusi tiap segmen pasar terhadap total pendapatan kamar, serta mendukung proses pengambilan keputusan dalam menyusun strategi harga, promosi, dan alokasi kamar secara lebih terarah.



Gambar 3.92 *Output Market Segment by Brand* Submenu *Room Revenue*

3. *Market Segment by Brand* Submenu *Average Room Rate*

Pembangunan submenu *Average Room Rate* (ARR) pada *dashboard Market Segment by Brand* bertujuan untuk menyajikan analisis rata-rata harga kamar per segmen pasar hotel dalam periode tertentu, dengan fokus pada nilai

aktual, target anggaran, dan perbandingan historis. Berikut adalah input, *process*, dan *output* dari *Market Segment by Brand* submenu ARR:

a. Input

Pembangunan submenu *Average Room Rate* (ARR) dalam *dashboard Market Segment by Brand* menggunakan *dataset* yang sama dengan submenu *Room Night*, yaitu *materialized view* bernama mv_market_segment_lastyear yang dibentuk melalui proses ETL dan *query* SQL seperti pada Gambar 3.90. Meskipun *dataset* yang digunakan sama, fokus analisis pada submenu ARR berbeda karena ditujukan untuk mengevaluasi rata-rata harga kamar per segmen pasar. Hal ini tercermin dari penggunaan kolom yang relevan dengan pendapatan dan volume kamar, yaitu *roomrevenue* dibagi *roomnight* untuk nilai aktual, *budgetroomrevenue* dibagi *budgetroomnight* untuk target anggaran, serta *prev_roomrevenue* dibagi *prev_roomnight* untuk capaian tahun sebelumnya. Seluruh input tersebut mendukung analisis tarif rata-rata kamar secara lebih detail guna membandingkan strategi harga antar segmen pasar dan mengidentifikasi perubahan harga secara historis.

b. *Process*

Perancangan visualisasi pada submenu *Average Room Rate* (ARR) memerlukan tahapan pemrosesan data berbasis kalkulasi metrik analitis dari hasil agregasi kolom pendapatan dan jumlah kamar terjual pada *dataset* mv_market_segment_lastyear. Daftar kalkulasi metrik dan filter yang digunakan untuk membuat *dashboard Market Segment by Brand* submenu ARR disajikan pada Tabel 3.13. Metrik dasar ARR dihitung langsung di Apache Superset melalui *custom metric*, dengan rumus pembagian antara *roomrevenue* dan *roomnight* untuk nilai aktual, *budgetroomrevenue* dibagi *budgetroomnight* untuk target anggaran, serta *prev_roomrevenue* dibagi *prev_roomnight* untuk nilai historis. Metrik pertumbuhan ARR disusun dari selisih nilai aktual dan historis yang dibagi dengan nilai historis, sementara metrik kontribusi dihitung dari rasio ARR per segmen terhadap total ARR

semua segmen menggunakan *function OVER()*. Seluruh metrik disusun secara terstruktur dan difilter berdasarkan periode transaksi tertentu serta kode *brand* hotel tertentu agar hasil analisis lebih terarah dan sesuai kebutuhan.

Tabel 3.13 Rincian Metriks dan Filter pada *Market Segment by Brand* Submenu ARR

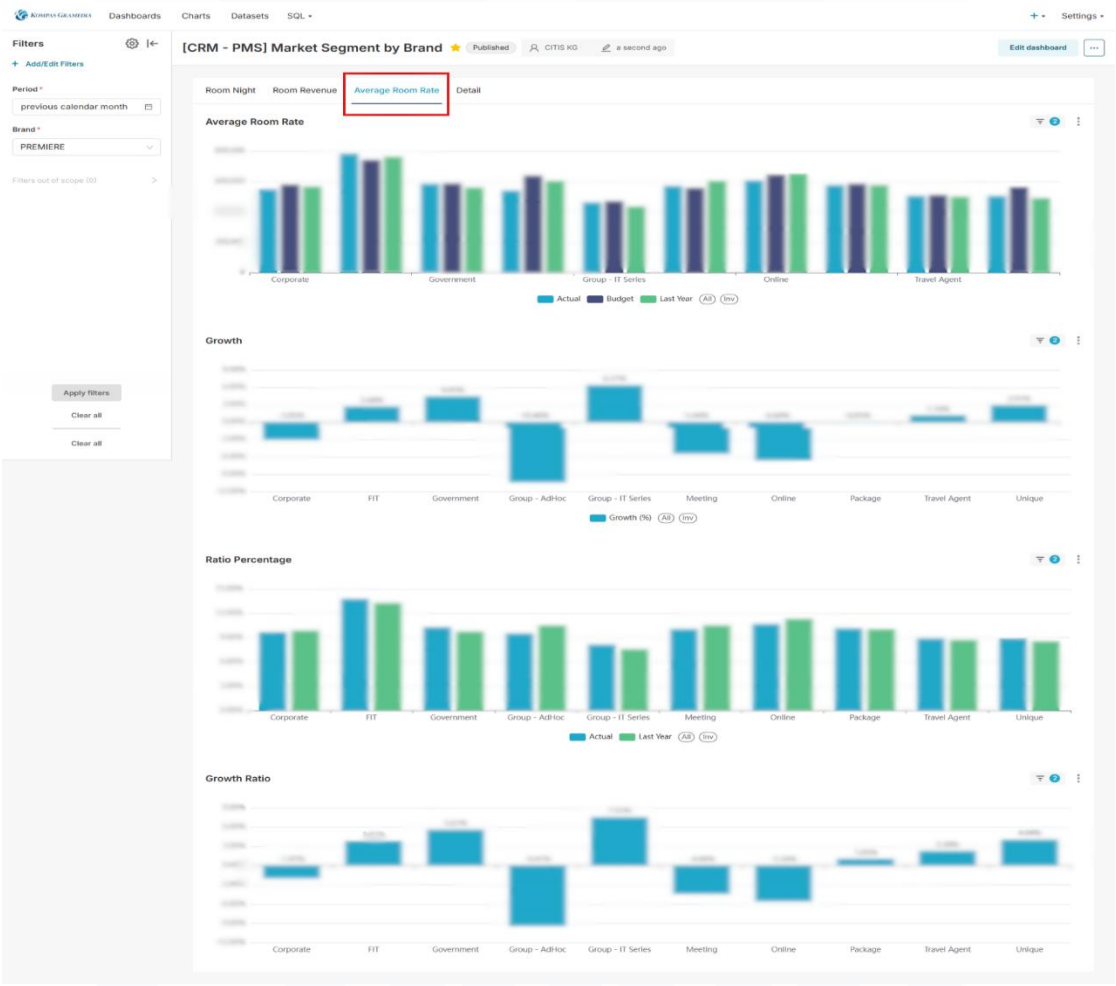
Nama *Chart*	Kalkulasi Metriks	Filter
Average Room Rate (*Bar Chart*)	ACTUAL CASE SUM(...) WHEN 0 THEN 0 ELSE SUM(...) / SUM(...) END BUDGET CASE SUM(...) WHEN 0 THEN 0 ELSE SUM(...) / SUM(...) END X-axis abc marketdescription LAST YEAR CASE SUM(...) WHEN 0 THEN 0 ELSE SUM(...) / SUM(...) END	2025-05-01 ≤ cur_trxdate < 20... classcode IN ('PREMIERE')
Growth Average Room Rate (*Bar Chart*)	((CASE SUM(...) WHEN 0 THEN 0 ELSE SUM(...) / SUM(...))END) - (CASE SUM(...) WHEN 0 THEN 0 ELSE SUM(...) / SUM (...) END)) / NULLIF((CASE WHEN SUM(...) = 0 THEN 0 ELSE SUM(...) / SUM (...) END), 0) X-axis abc marketdescription	2025-05-01 ≤ cur_trxdate < 20... classcode IN ('PREMIERE')
Ratio Average Room Rate (*Bar Chart*)	ACTUAL CASE WHEN SUM(...) = 0 THEN 0 ELSE (SUM(...) / SUM(...)) / NULLIF(SUM(SUM(...) / NULLIF(SUM (...), 0)) OVER (), 0) END X-axis abc marketdescription LAST YEAR CASE WHEN SUM(...) = 0 THEN 0 ELSE (SUM(...) / SUM (...) / NULLIF(SUM(SUM(...) / NULLIF (SUM(...), 0)) OVER (), 0) END	2025-05-01 ≤ cur_trxdate < 20... classcode IN ('PREMIERE')
Growth Ratio Average Room Rate (*Bar Chart*)	CASE WHEN SUM(...) = 0 THEN 0 ELSE (((SUM(...) / SUM(...)) / NULLIF(SUM(SUM(...) / NULLIF(SUM(...), 0)) OVER (), 0)) - ((SUM(...) / SUM (...)) / NULLIF(SUM(SUM(...) / NULLIF(SUM(...), 0)) OVER (), 0))) / ((SUM(...) / SUM (...)) / NULLIF(SUM(SUM(...) / NULLIF(SUM(...), 0)) OVER (), 0)) END	2025-05-01 ≤ cur_trxdate < 20... classcode IN ('PREMIERE')

Proses selanjutnya adalah pembangunan visualisasi berbasis *bar chart* untuk menyajikan metrik ARR dalam bentuk visual yang mudah dipahami oleh pengguna. Submenu ini terdiri dari empat *chart* utama yakni *Market Segment Average Room Rate*, *Growth Average Room Rate*, R*atio Average Room Rate*, dan *Growth Ratio Average Room Rate* yang masing-masing menampilkan nilai rata-rata aktual, pertumbuhan tahunan, rasio kontribusi, serta perubahan kontribusi ARR antar periode. Seluruh *chart* menggunakan *marketdescription* sebagai sumbu X untuk mengelompokkan data berdasarkan segmen pasar hotel, dengan *bar chart* yang memudahkan perbandingan visual antar segmen. Filter cur_trxdate dan *classcode* diterapkan untuk menjaga konsistensi waktu dan *brand* dalam analisis. Visualisasi ini tidak hanya menyampaikan metrik ARR secara numerik, tetapi juga menunjukkan kontribusi tiap segmen terhadap perubahan rata-rata harga kamar, sehingga menjadi dasar evaluasi harga dan penyusunan kebijakan tarif yang lebih tepat sesuai kondisi pasar.

c. *Output*

Submenu *Average Room Rate* (ARR) pada dashboard *Market Segment by Brand* menghasilkan *output* visualisasi berupa empat *bar chart* utama yakni *Market Segment Average Room Rate*, *Growth Average Room Rate*, *Ratio Average Room Rate*, dan *Growth Ratio Average Room Rate* yang menyajikan rata-rata harga kamar per segmen pasar hotel dalam periode tertentu. Visualisasi *dashboard* seperti pada Gambar 3.93 ini mencakup nilai aktual, target anggaran, perbandingan historis, kontribusi relatif tiap segmen, serta persentase pertumbuhannya dibandingkan periode sebelumnya. Seluruh *chart* menggunakan dimensi *marketdescription* sebagai sumbu X dan telah difilter berdasarkan waktu transaksi serta *brand* hotel tertentu untuk memastikan analisis yang terarah. *Output dashboard* dibangun secara interaktif menggunakan Apache Superset dan di-*embed* ke dalam sistem *Property Management System* (PMS), sehingga dapat diakses oleh tim operasional hotel seperti manajer unit, supervisor *front office*, dan manajemen pusat. Informasi ARR yang ditampilkan berperan penting

dalam membantu evaluasi strategi penetapan harga, perbandingan tarif antar segmen pasar, serta penyusunan kebijakan promosi yang sesuai dengan daya beli dan karakteristik masing-masing pasar.



Gambar 3.93 *Output Market Segment by Brand* Submenu *Average Room Rate*

4. *Market Segment by Brand* Submenu Detail

Pembangunan submenu Detail pada *dashboard Market Segment by Brand* bertujuan untuk menyajikan performa hotel berdasarkan segmentasi pasar melalui tiga indikator utama, yaitu jumlah kamar terjual, pendapatan kamar, dan rata-rata harga kamar yang ditampilkan dalam bentuk tabel analitis secara terperinci. Berikut adalah input, *process*, dan *output* dari *Market Segment by Brand* submenu Detail:

a. Input

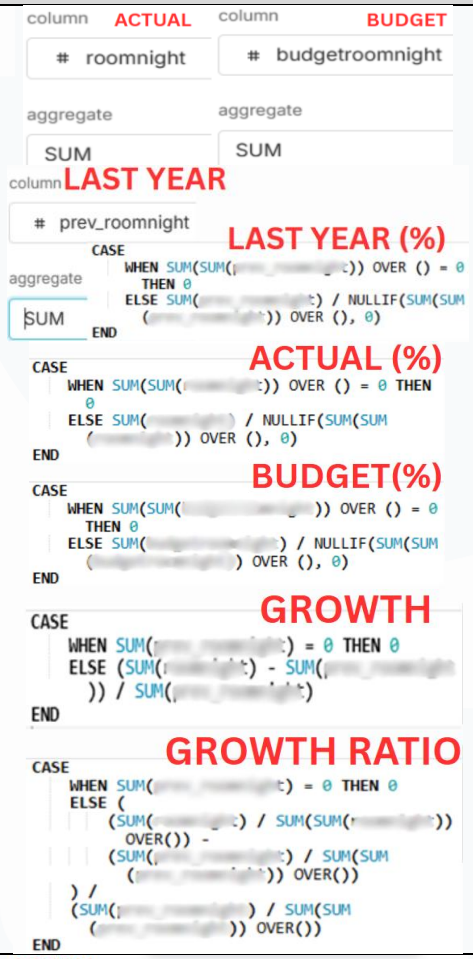
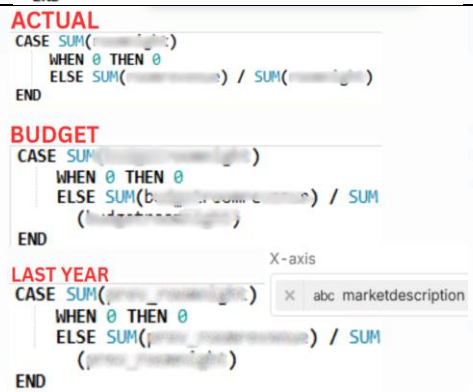
Pembangunan submenu Detail dalam *dashboard Market Segment by Brand* menggunakan *dataset* yang sama dengan submenu *Room Night*, yaitu *materialized view* bernama mv_market_segment_lastyear yang dibentuk melalui proses ETL dan *query* SQL seperti pada Gambar 3.90. Meskipun struktur *dataset* identik, perbedaannya terletak pada tujuan visualisasi, fungsi analisis, dan kolom-kolom yang digunakan dalam pembentukan metrik. Submenu Detail menyajikan kombinasi tiga aspek utama yakni *room night*, *room revenue*, dan *average room rate* dalam tampilan tabel. Penggunaan kolom seperti *roomnight*, *roomrevenue*, serta rasio di antara keduanya mencerminkan konteks metrik aktual, anggaran, historis, kontribusi, pertumbuhan, hingga rasio pertumbuhan. Seluruh nilai diklasifikasikan berdasarkan dimensi *marketdescription* untuk menampilkan performa tiap segmen pasar secara granular, dan difilter berdasarkan waktu transaksi serta kode *brand* hotel agar visualisasi yang dihasilkan relevan dengan kebutuhan analisis manajerial.

b. *Process*

Perancangan metrik dalam submenu Detail memerlukan kalkulasi analitis yang kompleks terhadap data dalam *materialized view* mv_market_segment_lastyear, yang merupakan hasil integrasi dari sistem *Property Management System* (PMS) melalui *pipeline* ETL dan diselaraskan dalam struktur *data warehouse*. Kalkulasi mencakup tiga aspek utama yakni jumlah kamar terjual (*room night*), pendapatan kamar (*room revenue*), dan tarif rata-rata kamar (*average room rate*) berdasarkan segmentasi pasar yang diidentifikasi melalui kolom *marketdescription*. Daftar kalkulasi metrik dan filter yang digunakan untuk membuat *dashboard Market Segment by Brand* submenu Detail disajikan pada Tabel 3.14. Setiap aspek dianalisis dalam konteks aktual, anggaran (*budget*), dan historis (*last year*), serta dibandingkan secara proporsional dan temporal. Fungsi agregasi seperti *SUM()* digunakan untuk perhitungan metrik numerik, sementara *CASE WHEN* dan *NULLIF* diterapkan untuk menghindari kesalahan pembagian dengan nol pada rasio dan persentase.

Nilai pertumbuhan dihitung dari selisih antara nilai aktual dan historis yang dibagi nilai historis, sedangkan *growth ratio* digunakan untuk membandingkan kontribusi segmen saat ini dengan kontribusi historisnya secara proporsional, guna menilai perubahan daya saing antar segmen pasar.

Tabel 3.14 Rincian Metriks dan Filter pada *Market Segment by Brand* Submenu *Detail*

Nama *Chart*	Kalkulasi Metriks	Filter
Room Night (*Table*)		2025-05-01 ≤ cur_trxdate < 20... classcode IN ('PREMIERE')
Average Room Rate (*Table*)		2025-05-01 ≤ cur_trxdate < 20... classcode IN ('PREMIERE')

Nama *Chart*	Kalkulasi Metriks	Filter
Room Revenue (*Table*)	ACTUAL column roomrevenue aggregate SUM BUDGET column budgetroomrevenue aggregate SUM LAST YEAR column prev_roomrevenue aggregate SUM LAST YEAR (%) ACTUAL (%) BUDGET (%)	2025-05-01 ≤ cur_trxdate < 20... classcode IN ('PREMIERE')

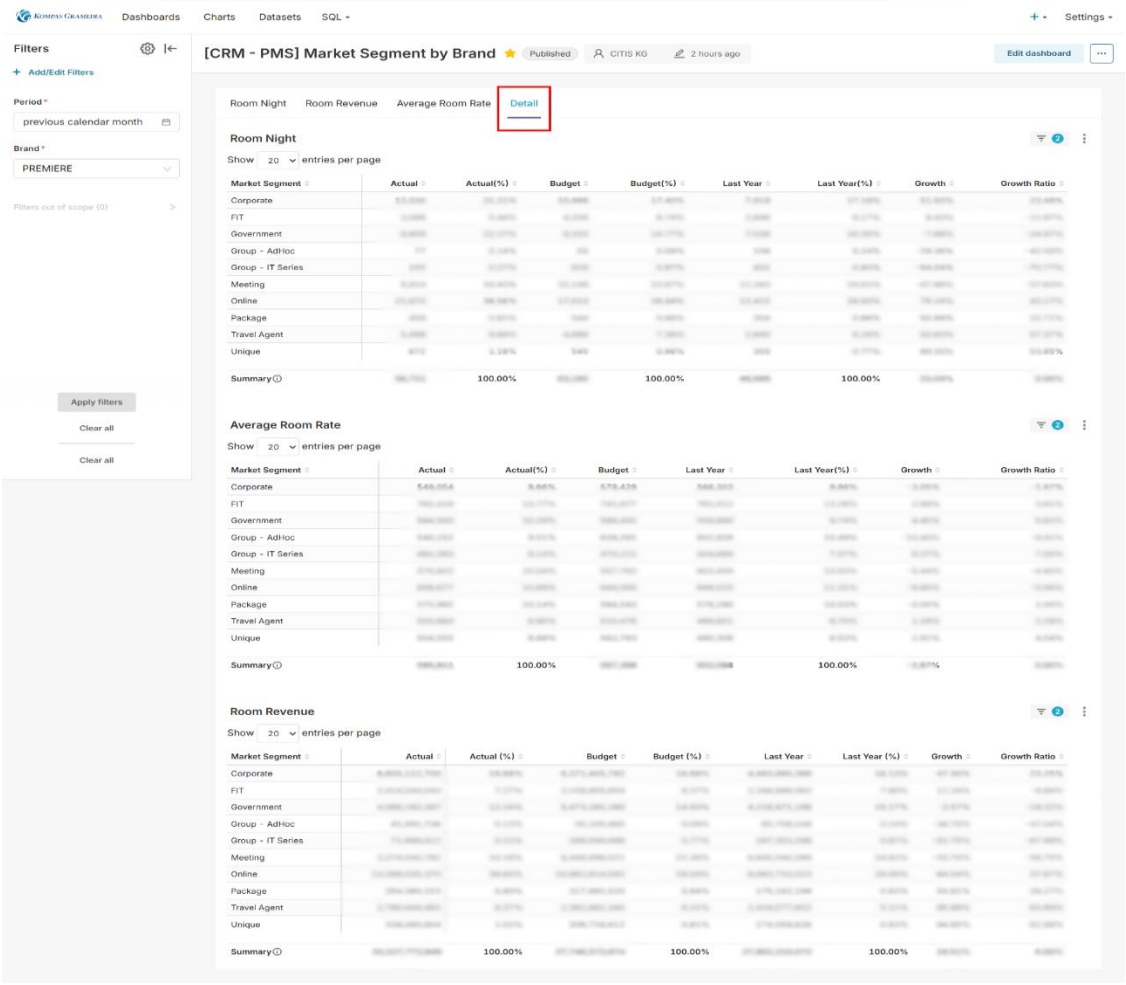
Nama *Chart*	Kalkulasi Metriks	Filter
	GROWTH CASE SUM([redacted]) WHEN 0 THEN 0 ELSE (SUM([redacted]) - SUM ([redacted])) / SUM ([redacted]) END GROWTH RATIO CASE WHEN SUM([redacted]) = 0 THEN ELSE ((SUM([redacted]) / SUM(SUM ([redacted])) OVER()) - (SUM([redacted]) / SUM(SUM ([redacted])) OVER())) / (SUM([redacted]) / SUM(SUM ([redacted])) OVER()) END	

Proses selanjutnya adalah pembangunan visualisasi dalam bentuk *table* untuk menyajikan data numerik yang terstruktur dan mudah dibandingkan antar segmen. Tiga tabel utama ditampilkan untuk masing-masing aspek yakni *Room Night*, *Room Revenue*, dan *Average Room Rate* dengan *marketdescription* sebagai sumbu X untuk memperlihatkan perbandingan kinerja antar segmen pasar hotel. Pemilihan bentuk tabel dilakukan karena mampu menampilkan delapan metrik sekaligus dalam satu tampilan secara detail berserta *values* datanya, yaitu nilai aktual, persentase aktual, anggaran, persentase anggaran, nilai historis, persentase historis, pertumbuhan nilai, dan rasio pertumbuhan. Penyusunan dilakukan melalui fitur *Chart Builder* di Apache Superset, dengan logika kalkulasi yang dipetakan dari SQL ke dalam *custom metric* secara langsung atau melalui *materialized view* di level *database*. Filter yang digunakan konsisten di ketiga chart.

c. *Output*

Submenu Detail pada *dashboard Market Segment by Brand* menghasilkan *output* visualisasi seperti yang terlihat pada Gambar 3.94 yang mencakup tiga tabel dengan indikator performa hotel berdasarkan segmentasi pasar, yaitu jumlah kamar terjual (*room night*), pendapatan kamar (*room revenue*), dan rata-rata harga kamar (*average room rate*). Setiap tabel memuat delapan metrik untuk masing-masing dimensi,

termasuk nilai aktual, anggaran, historis, persentase kontribusi terhadap total, tingkat pertumbuhan, dan rasio pertumbuhan dibandingkan tahun sebelumnya. Visualisasi ini dirancang untuk memberikan representasi granular terhadap performa tiap market segment yang diklasifikasikan berdasarkan kolom *marketdescription*, sehingga memungkinkan analisis komparatif secara horizontal antar segmen dan vertikal antar periode waktu. *Output* ini akan diintegrasikan langsung ke dalam sistem *Property Management System* (PMS) melalui fitur *embed* dari Apache Superset agar dapat diakses oleh tim operasional hotel. Tujuan utama penyajian *output* ini adalah untuk mendukung evaluasi performa pemasaran, pemantauan pencapaian target, serta pengambilan keputusan terkait penyesuaian harga dan segmentasi pasar yang lebih tepat sasaran.



Gambar 3.94 *Output Market Segment by Brand* Submenu *Detail*

3.2.9.2. *CRM Menu Dashboard – Market Segment with Budget*

Pembangunan *dashboard Market Segment with Budget* bertujuan untuk menyajikan perbandingan kinerja jumlah kamar terjual, pendapatan kamar, dan tarif rata-rata kamar berdasarkan segmentasi pasar hotel dalam konteks aktual, anggaran, dan historis tahun sebelumnya. Berikut adalah input, *process*, dan *output* dari *dashboard Market Segment with Budget*:

a. Input

Penyusunan visualisasi pada *dashboard Market Segment with Budget* dimulai dengan tahap penyediaan input berupa *dataset materialized view* mv_market_segment_lastyear, sama seperti pada submenu *Room Night* di *dashboard Market Segment by Brand*. *Dataset* dengan query SQL seperti pada Gambar 3.90 ini dibentuk melalui proses ETL yang mengintegrasikan beberapa tabel, seperti mv_roomcountsheetguestprofile, pms_mstmarketsegmentbudget, dan pms_sysmasterhotel. Meskipun struktur data dan *query*-nya identik, perbedaannya terletak pada fungsi analisis, fokus visualisasi, dan kolom metrik yang digunakan. *Dashboard Market Segment with Budget* menyajikan perbandingan performa jumlah kamar, pendapatan kamar, dan tarif rata-rata terhadap target anggaran dan capaian tahun sebelumnya. Kolom metrik seperti *roomnight*, *roomrevenue*, *budgetroomnight*, dan *budgetroomrevenue* digunakan dalam kombinasi agregasi untuk menghasilkan nilai aktual, anggaran, dan varians berdasarkan dimensi *marketdescription*. Seluruh input difilter berdasarkan parameter waktu, nama hotel, dan *classcode* agar data yang ditampilkan relevan.

b. *Process*

Perancangan visualisasi pada *dashboard Market Segment with Budget* memerlukan tahapan kalkulasi berbasis logika agregasi dari *dataset* mv_market_segment_lastyear, yang memuat data historis, aktual, dan anggaran. Daftar kalkulasi metrik dan filter yang digunakan untuk membuat *dashboard Market Segment with Budget* disajikan pada Tabel 3.14. Data ini diproses untuk menghasilkan metrik dalam lima kategori yaitu *Actual*, *Last*

Year, *Budget*, *Variance with Last Year*, dan *Variance with Budget*. Metrik dasar seperti *Room Night* dan *Revenue* dihitung menggunakan *function* agregat *SUM()*, sementara *Average Room Rate* diperoleh dari pembagian antara pendapatan dan jumlah kamar terjual. Transformasi lanjutan seperti *Room Night* (%) dan *Revenue* (%) dihitung melalui pembagian nilai segmen terhadap total keseluruhan menggunakan *SUM*(...) *OVER* (). Untuk analisis varians, nilai aktual dikurangkan dengan nilai pembanding (*budget* atau *last year*), lalu dihitung persentase selisihnya terhadap nilai pembanding. Ekspresi *CASE WHEN* dan *NULLIF* digunakan untuk menghindari pembagian dengan nol, sehingga mencegah kesalahan logika dalam visualisasi.

Tabel 3.15 Rincian Metriks dan Filter pada *Dashboard Market Segment with Budget*

Nama *Chart*	Kalkulasi Metriks	Filter
Actual (*Table*)	ROOM NIGHT: # roomnight; aggregate: SUM REVENUE: # roomrevenue; aggregate: SUM ROOM NIGHT (%): SUM( ) / SUM(SUM( )) OVER () AVERAGE ROOM RATE: CASE SUM( ) WHEN 0 THEN 0 ELSE SUM( ) / SUM( ) END REVENUE (%): SUM( ) / SUM(SUM( )) OVER ()	2025-05-01 ≤ cur_trxdate < 20... classcode IN ('PREMIERE') hotelname IN ('Hotel Santika Pre...
Last Year (*Table*)	ROOM NIGHT: # prev_roomnight; aggregate: SUM REVENUE: # prev_roomrevenue; aggregate: SUM ROOM NIGHT (%): SUM( ) / SUM(SUM( )) OVER ()	2025-05-01 ≤ cur_trxdate < 20... classcode IN ('PREMIERE') hotelname IN ('Hotel Santika Pre...

Nama *Chart*	Kalkulasi Metriks	Filter

	AVERAGE ROOM RATE
	`CASE SUM( ) WHEN 0 THEN 0 ELSE SUM( ) / SUM ( ) END`
	REVENUE (%)
	`SUM( ) / SUM(SUM ( )) OVER ()`	

Budget (*Table*)	**ROOM NIGHT** # budgetroomnight
	REVENUE # budgetroomrevenue
	aggregate SUM
	aggregate SUM
	ROOM NIGHT (%)
	`SUM( ) / SUM(SUM( )) OVER ()`
	AVERAGE ROOM RATE
	`CASE SUM( ) WHEN 0 THEN 0 ELSE SUM( ) / SUM ( ) END`
	REVENUE (%)
	`SUM( ) / SUM(SUM ( )) OVER ()`
		2025-05-01 ≤ cur_trxdate < 20...
		classcode IN ('PREMIERE')
		hotelname IN ('Hotel Santika Pre...

Variance with Last Year (*Table*)	**ROOM NIGHT**
	`sum( )-sum( )`
	ROOM NIGHT (%)
	`CASE WHEN SUM( ) = 0 THEN 0 ELSE (SUM( ) - SUM(prev_roomnight )) / SUM( ) END`
	AVERAGE ROOM RATE
	`(CASE WHEN SUM( ) = 0 THEN 0 ELSE SUM( ) / SUM( ) END) - (CASE WHEN SUM( ) = 0 THEN 0 ELSE SUM( ) / SUM ( ) END)`
	REVENUE
	`sum( )-sum( )`
		2025-05-01 ≤ cur_trxdate < 20...
		classcode IN ('PREMIERE')
		hotelname IN ('Hotel Santika Pre...

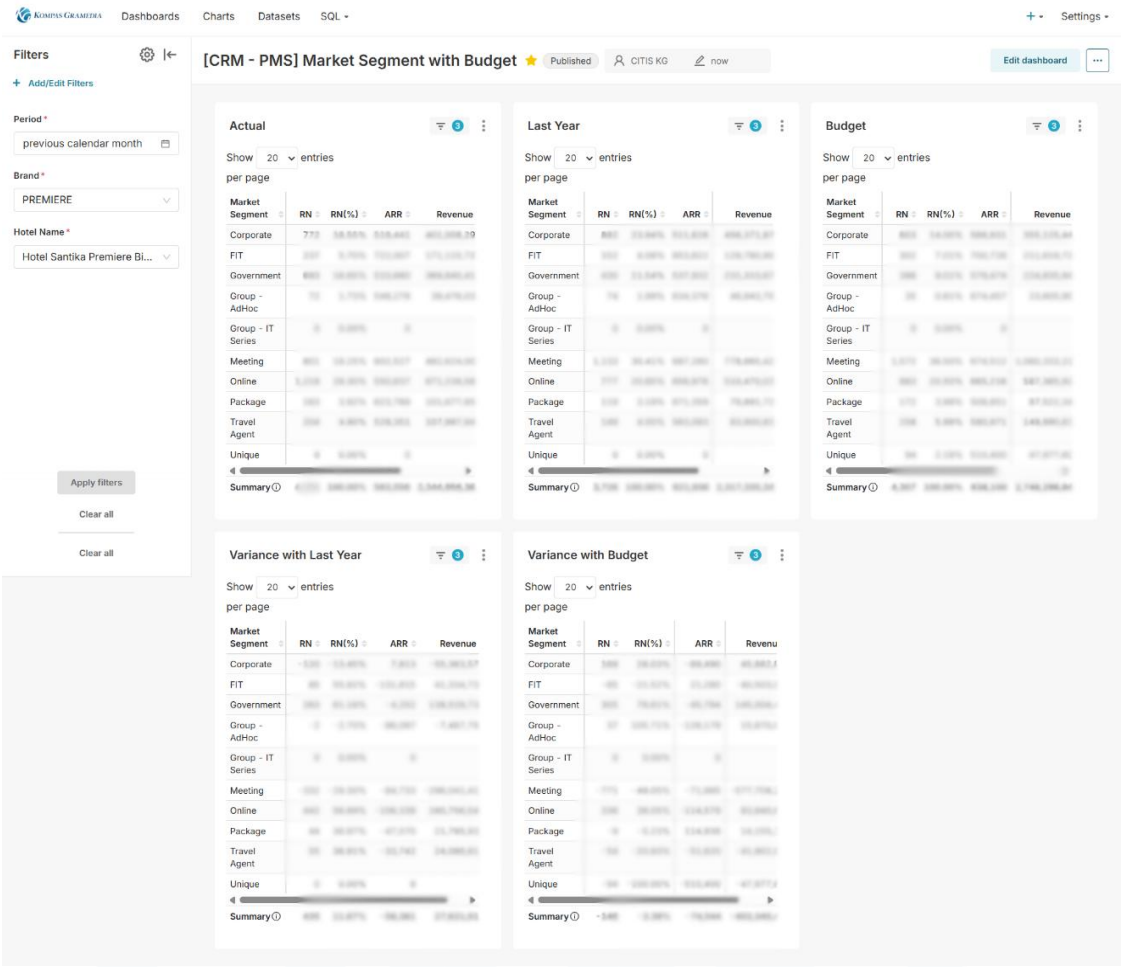
Variance with Budget (*Table*)	**ROOM NIGHT**
	`sum( )-sum( )`
	ROOM NIGHT (%)
	`CASE WHEN SUM( ) = 0 THEN 0 ELSE (SUM( ) - SUM( )) / SUM( ) END`
	AVERAGE ROOM RATE
	`(CASE WHEN SUM( ) = 0 THEN 0 ELSE SUM( ) / SUM( ) END) - (CASE WHEN SUM( ) = 0 THEN 0 ELSE SUM( ) / SUM ( ) END)`
	REVENUE
	`sum( )-sum( )`
		2025-05-01 ≤ cur_trxdate < 20...
		classcode IN ('PREMIERE')
		hotelname IN ('Hotel Santika Pre...

Proses selanjutnya adalah penyusunan hasil kalkulasi metrik pada *dashboard Market Segment with Budget* yang disajikan dalam lima *table* yang masing-masing merepresentasikan kondisi performa aktual, capaian tahun sebelumnya, target anggaran, serta selisih terhadap kedua komponen pembanding tersebut. Seluruh chart dibangun menggunakan fitur *chart builder* di Apache Superset dengan *marketdescription* sebagai sumbu X, agar visualisasi terfokus pada perbandingan segmentasi pasar. Setiap tabel menampilkan lima metrik utama yakni *Room Night*, *Room Night* (%), *Average Room Rate* (ARR), *Revenue*, dan *Revenue* (%) yang disusun sejajar untuk memudahkan pembacaan dan interpretasi oleh pengguna akhir. Semua *chart* diterapkan dengan filter yang konsisten, mencakup periode waktu transaksi tertentu, *brand* hotel tertentu, dan nama hotel tertentu, sehingga visualisasi yang dihasilkan relevan dengan kebutuhan analisis kinerja operasional hotel.

c. *Output*

Dashboard Market Segment with Budget menghasilkan lima *table chart* yang menyajikan perbandingan kinerja segmentasi pasar hotel berdasarkan jumlah kamar terjual, tarif rata-rata kamar, dan pendapatan kamar dalam tiga konteks utama (aktual, historis tahun lalu, dan target anggaran). Hasil *dashboard Market Segment with Budget* ditampilkan pada Gambar 3.95 yang mana setiap *chart*-nya menampilkan metrik numerik dan proporsional, yang mencakup nilai absolut, persentase kontribusi, serta selisih terhadap anggaran dan capaian tahun sebelumnya, baik dalam bentuk nilai murni maupun persentase varians. Seluruh *output* disusun berdasarkan dimensi *marketdescription* untuk merepresentasikan performa tiap segmen pasar secara horizontal, dan difilter menggunakan parameter waktu, nama hotel, serta *classcode* agar analisis relevan. Visualisasi ini akan diintegrasikan langsung ke dalam sistem *Property Management System* (PMS) melalui fitur *embed* dari Apache Superset, sehingga dapat diakses oleh pihak terkait seperti *General* Manajer, *Revenue* Manajer, dan *Business Analyst* untuk memantau pencapaian target, menganalisis kinerja pasar, serta menetapkan

kebijakan harga dan strategi penjualan berdasarkan selisih antara realisasi dan rencana.



Gambar 3.95 *Output Dashboard Market Segment with Budget*

3.2.9.3. *CRM Menu Dashboard – Average Room Rate Statistic*

Pembangunan *dashboard Average Room Rate Statistic* bertujuan untuk menyajikan perbandingan rata-rata harga kamar hotel berdasarkan tiga indikator utama yaitu nilai anggaran (*Budget*), realisasi (*Actual*), dan capaian tahun sebelumnya (*Last Year*) untuk setiap kategori *brand* hotel. Berikut adalah input, *process*, dan *output* dari *dashboard Average Room Rate Statistic*:

a. Input

Dasar input penyusunan visualisasi pada *dashboard Average Room Rate Statistic* berasal dari *dataset* mv_manager_report_with_budget_lastyear

yang dirancang melalui proses agregasi dan penggabungan beberapa tabel dalam *data warehouse* hospitalitydwh. *Dataset* dengan *query* seperti pada Gambar 3.96 ini dibentuk menggunakan perintah SQL pada PostgreSQL dalam bentuk materialized view, yang mengintegrasikan data historis aktual dari mv_manager_report, data anggaran dari vw_budget_hotel_monthly, serta atribut klasifikasi hotel seperti *classcode* dan *hotelname*. Proses pembuatan MV diawali dengan pembuatan CTE untuk menghitung nilai aktual seperti total pendapatan kamar (*roomrevenue*), total pendapatan hotel (*hotelrevenue*), rata-rata harga kamar (ARR), dan tingkat hunian kamar (*occupancy*) dalam lingkup waktu bulanan berdasarkan kolom *trxdate*. Selanjutnya, dilakukan *self join* untuk menarik data historis dari periode yang sama tahun sebelumnya, kemudian dilanjutkan dengan penggabungan data anggaran berdasarkan *hotelid* dan *trxdate* yang sesuai. Hasil penggabungan ini dilengkapi dengan data referensi seperti jumlah kamar terisi (*occupiedroom*), penggunaan kamar internal (*houseuse*), dan total ketersediaan kamar (*roomavailable*).

Gambar 3.96 *Query Materialized View* mv_manager_report_with_budget_lastyear

b. *Process*

Perancangan metrik pada *dashboard Average Room Rate Statistic* dimulai dengan kalkulasi tiga nilai kunci yakni *Budget*, *Actual*, dan *Last Year* berdasarkan *dataset* mv_manager_report_with_budget_lastyear. Daftar kalkulasi metrik dan filter yang digunakan untuk membuat *dashboard Average Room Rate Statistic* disajikan pada Tabel 3.15. Metrik *Budget* dihitung melalui pembagian *SUM(budgetroomrevenue)* dengan *SUM(budgetoccupiedroom)* untuk memperoleh rata-rata tarif kamar yang direncanakan. Nilai *Actual* diperoleh dari *SUM(roomrevenue)* dibagi *(SUM(occupiedroom) - SUM(houseuse))*, dimana *houseuse* dikurangkan untuk mengecualikan penggunaan kamar internal dari nilai okupansi. Metrik *Last Year* dihitung menggunakan *SUM(previous_arr)* yang diambil dari data historis pada bulan yang sama tahun sebelumnya, hasil *join* terhadap mv_manager_report setahun sebelumnya. Filter yang diterapkan adalah berdasarkan periode transaksi tertentu (*curr_trxdate*).

Tabel 3.16 Rincian Metriks dan Filter pada *Dashboard Average Room Rate Statistic*

Nama *Chart*	Kalkulasi Metriks	Filter
Average Room Rate Statistic (*Bar Chart*)	BUDGET CASE WHEN SUM([redacted]) = 0 THEN 0 ELSE SUM([redacted]) / SUM ([redacted]) END ACTUAL CASE WHEN (SUM([redacted]) - SUM([redacted])) = 0 THEN 0 ELSE SUM([redacted]) / (SUM([redacted]) - SUM([redacted])) END LAST YEAR column # prev_arr aggregate SUM	X 2025-05-01 ≤ cur_trxdate < 20...

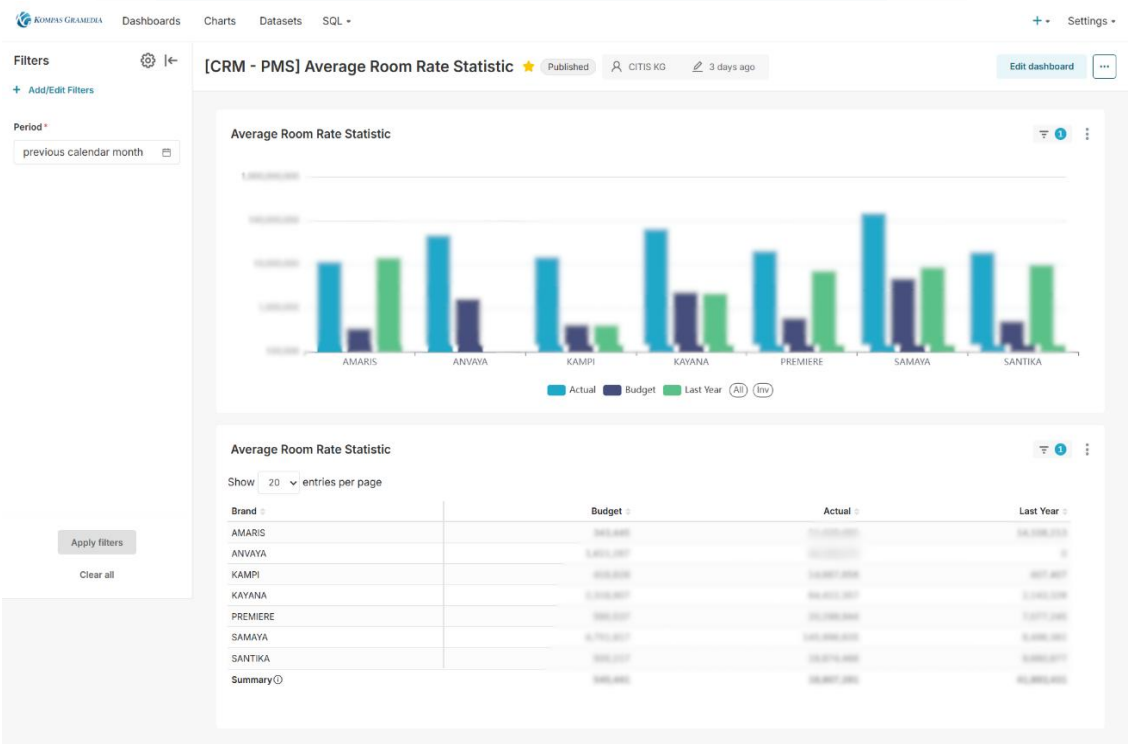
Nama *Chart*	Kalkulasi Metriks	Filter
Average Room Rate Statistic (*Table*)	BUDGET CASE WHEN SUM(...) = 0 THEN 0 ELSE SUM(...) / SUM (...) END ACTUAL CASE WHEN (SUM(...) - SUM(...)) = 0 THEN 0 ELSE SUM(...) / (SUM(...) - SUM(...)) END LAST YEAR column # prev_arr aggregate SUM	2025-05-01 ≤ cur_trxdate < 20...

Proses selanjutnya adalah penyusunan visualisasi pada *dashboard Average Room Rate* (ARR) *Statistic* melalui dua jenis *chart* yaitu *Bar Chart* dan *Table*, dengan sumbu horizontal berbasis *classcode* yang merepresentasikan *brand* hotel. *Bar Chart* digunakan untuk menampilkan perbandingan visual antara nilai *Budget*, *Actual*, dan *Last Year* dalam format batang sejajar untuk setiap *brand*, sehingga memudahkan identifikasi kinerja tertinggi atau terendah dalam periode tertentu. Setiap batang merepresentasikan hasil kalkulasi metrik, dikelompokkan berdasarkan *classcode* dan diwarnai sesuai kategori waktu (*budget*, *actual*, atau *historical*). *Table Chart* disusun dalam bentuk tabel interaktif dengan baris berdasarkan *brand* hotel dan kolom berisi ketiga metrik utama *budget* ARR, *actual* ARR, dan *previous* ARR untuk memberikan referensi numerik yang lebih spesifik. Kombinasi kedua jenis chart ini dirancang untuk memenuhi kebutuhan manajerial dalam bentuk ringkasan visual sekaligus mendukung analisis operasional yang memerlukan data kuantitatif secara detail.

c. *Output*

Dashboard Average Room Rate (ARR) *Statistic* menghasilkan *output* visualisasi seperti pada Gambar 3.97 yang menyajikan perbandingan rata-rata tarif kamar berdasarkan tiga indikator (*Budget*, *Actual*, *Last Year*) untuk masing-masing kategori *brand* hotel. Visualisasi ditampilkan melalui dua jenis *chart* utama, yaitu *bar chart* dan *table chart*. *Bar chart* menampilkan

ARR secara visual dalam bentuk batang sejajar yang diklasifikasikan berdasarkan *classcode*, memudahkan pengguna mengidentifikasi perbedaan performa antar *brand* hotel. *Table chart* menyajikan rincian numerik *values* dari ketiga metrik ARR dalam format tabular, yang mendukung analisis komparatif secara presisi untuk keperluan laporan dan evaluasi manajerial. Seluruh visualisasi akan diintegrasikan langsung ke dalam sistem *Property Management System* (PMS) melalui fitur *embedding* menggunakan Apache Superset, sehingga dapat diakses secara *real-time* oleh manajemen hotel seperti *General Manajer*, *Revenue Manajer*, dan *Board of Directors* untuk menyusun strategi harga, mengevaluasi pencapaian anggaran, dan merumuskan kebijakan peningkatan pendapatan kamar berdasarkan kinerja historis dan proyeksi anggaran.



Gambar 3.97 *Output Dashboard Average Room Rate Statistic*

3.2.9.4. *CRM Menu Dashboard – Occupancy Statistic*

Pembangunan *dashboard Occupancy Statistic* bertujuan untuk menyajikan analisis tingkat hunian kamar berdasarkan perbandingan antara

metrik anggaran, aktual, dan capaian tahun sebelumnya, dengan fokus pada klasifikasi hotel. Berikut adalah input, *process*, dan *output* dari *dashboard Occupancy Statistic*:

a. Input

Input pada *dashboard Occupancy Statistic* menggunakan *dataset* yang sama dengan *Average Room Rate Statistic*, yaitu mv_manager_report_with_budget_lastyear dengan *query* seperti pada Gambar 3.96. Meskipun berasal dari sumber dan struktur *query* yang sama, fokus penggunaan data pada *dashboard Occupancy Statistic* berbeda, yaitu untuk menganalisis tingkat hunian kamar berdasarkan tiga indikator *Budget*, *Actual*, dan *Last Year*. Proses agregasi dalam *materialized view* ini menekankan kalkulasi jumlah kamar terisi (*budgetoccupiedroom*, *occupancy*, dan *prev_occupancy*) terhadap total kamar tersedia (*budgetavailableroom*) dalam periode tertentu. Variabel-variabel tersebut disusun dalam kolom khusus hasil transformasi dari data aktual, historis, dan anggaran yang telah disesuaikan dengan struktur waktu (*trxdate*) dan klasifikasi hotel (*classcode*).

b. *Process*

Perancangan metrik pada *dashboard Occupancy Statistic* dimulai dengan menetapkan formula perhitungan tingkat hunian kamar berdasarkan tiga kategori utama yakni *Budget*, *Actual*, dan *Last Year*. Daftar kalkulasi metrik dan filter yang digunakan untuk membuat *dashboard Occupancy Statistic* disajikan pada Tabel 3.16. Metrik *Budget* dihitung menggunakan ekspresi *CASE* untuk membagi total kamar yang direncanakan terisi (*budgetoccupiedroom*) dengan jumlah kamar yang direncanakan tersedia (*budgetavailableroom*), disertai pengamanan terhadap kondisi pembagi nol. Metrik *Actual* diperoleh dari agregasi nilai kolom *occupancy*, yang sebelumnya telah dihitung dalam *materialized view*. Sementara itu, metrik *Last Year* dihitung dari total *prev_occupancy*, yaitu tingkat hunian kamar pada periode yang sama di tahun sebelumnya. Ketiga metrik ini dirancang

menggunakan *custom metric builder* di Apache Superset dengan mengacu pada data hasil agregasi bulanan berdasarkan *trxdate* dan segmentasi berdasarkan *classcode*.

Tabel 3.17 Rincian Metriks dan Filter pada *Dashboard Occupancy Statistic*

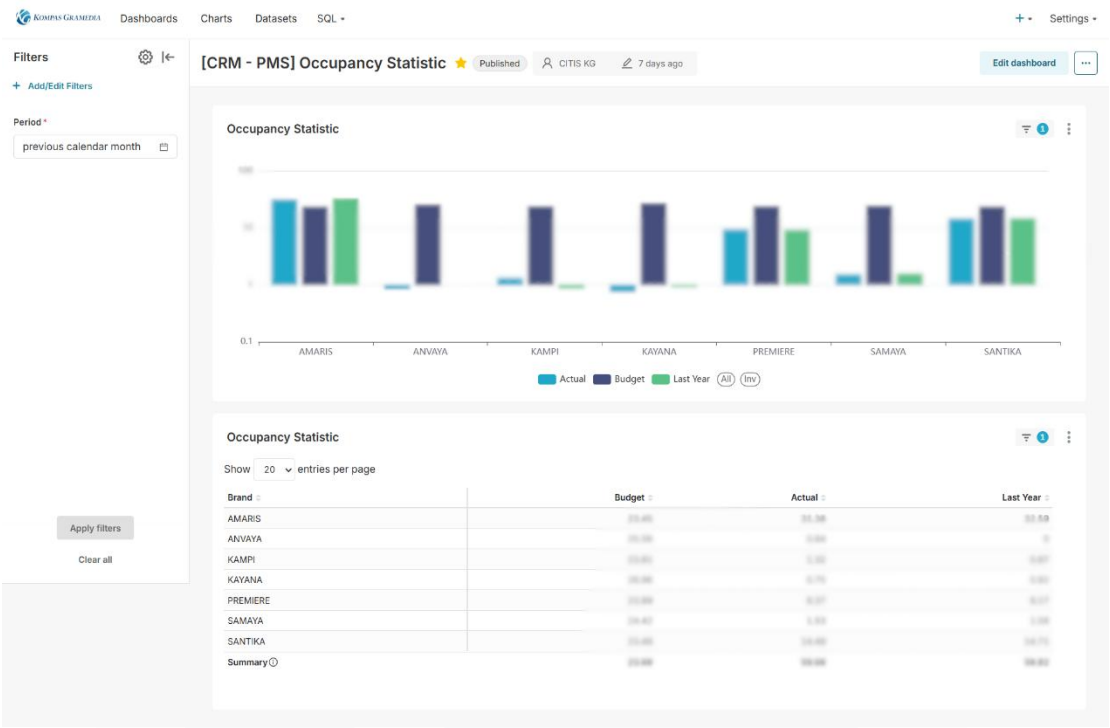
Nama *Chart*	Kalkulasi Metriks	Filter
Occupancy Statistic (*Bar Chart*)	BUDGET CASE WHEN SUM([redacted]) = 0 THEN 0 ELSE ROUND(SUM([redacted])::numeric / SUM([redacted]), 2) END column ACTUAL # occupancy aggregate SUM column LAST YEAR # prev_occupancy aggregate SUM	2025-05-01 ≤ cur_trxdate < 20...
Occupancy Statistic (*Table*)	BUDGET CASE WHEN SUM([redacted]) = 0 THEN 0 ELSE ROUND(SUM([redacted])::numeric / SUM([redacted]), 2) END column ACTUAL # occupancy aggregate SUM column LAST YEAR # prev_occupancy aggregate SUM	2025-05-01 ≤ cur_trxdate < 20...

Proses selanjutnya adalah pengembangan visualisasi pada *dashboard Occupancy Stastistic* menggunakan dua jenis chart, yaitu *Bar Chart* dan *Table*. *Bar Chart* memperlihatkan perbandingan tingkat okupansi antar *brand* hotel dalam bentuk batang vertikal yang menggambarkan selisih antar metrik *Budget*, *Actual*, dan *Last Year* dalam satuan persentase. *Table* menyajikan detail numerik ketiga metrik dengan presisi dua desimal untuk memudahkan interpretasi. Filter waktu diterapkan untuk membatasi data pada periode tertentu, memastikan konsistensi evaluasi performa bulanan. Seluruh visualisasi disusun dengan menyesuaikan komponen *chart* di Apache Superset, seperti penentuan sumbu X pada *classcode*, pemilihan *dataset* mv_manager_report_with_budget_lastyear, dan penerapan filter

dinamis berbasis *cur_trxdate*. Integrasi kedua *chart* dalam satu *dashboard* membantu pengguna untuk mengakses informasi okupansi hotel secara komprehensif dalam satu tampilan.

c. *Output*

Dashboard Occupancy Statistic yang terlihat pada Gambar 3.98 menghasilkan visualisasi perbandingan tingkat hunian kamar berdasarkan klasifikasi hotel untuk periode tertentu, dengan menggunakan dua *chart* yakni *bar chart* dan *table chart*. Kedua format ini menyajikan tiga indikator utama (*Budget*, *Actual*, dan *Last Year*) untuk memudahkan pengguna dalam membandingkan performa okupansi aktual, target anggaran, dan capaian historis secara tersegmentasi berdasarkan *classcode*. Grafik batang menggambarkan deviasi antar metrik dalam persentase, sementara tabel menyajikan *values* detail data numerik dengan akurasi dua desimal. Seluruh visualisasi akan di-*embed* dalam sistem PMS menggunakan fitur *embed dashboard* Superset, yang memudahkan akses langsung oleh manajer operasional, supervisor hotel, dan *Board of Directors*. *Output dashboard* ini digunakan untuk evaluasi kinerja hunian kamar, analisis tren okupansi bulanan, serta penyesuaian strategi pemasaran dan alokasi kamar berdasarkan data historis dan proyeksi anggaran.



Gambar 3.98 *Output Dashboard Occupancy Statistic*

3.2.9.5. *CRM Menu Dashboard – Hotel Revenue Statistic*

Pembangunan *dashboard Hotel Revenue Statistic* bertujuan untuk membandingkan pendapatan hotel aktual, anggaran, dan capaian tahun sebelumnya, serta menghitung pertumbuhan dan pencapaian anggaran untuk menganalisis kinerja keuangan hotel dalam periode tertentu. *Output* visualisasi ini disajikan melalui *bar chart* dan *pivot table* untuk memudahkan perbandingan antar *brand* hotel dan memantau kinerja pendapatan secara lebih terperinci. Berikut adalah input, *process*, dan *output* dari *dashboard Hotel Revenue Statistic*:

a. Input

Input visualisasi pada *dashboard Hotel Revenue Statistic* menggunakan *dataset* yang sama dengan *Average Room Rate Statistic*, yaitu mv_manager_report_with_budget_lastyear dengan *query* seperti pada Gambar 3.96 yang dibentuk dalam *materialized view* pada PostgreSQL. Meskipun berasal dari sumber *dataset* dan struktur *query* yang sama, fokus analisis *dashboard* ini berbeda karena lebih menekankan pada total

pendapatan hotel, mencakup *room revenue*, *food revenue*, *beverage revenue*, FB *other revenue*, dan *other revenue*. Kolom-kolom ini berasal dari proses ETL yang menggabungkan data historis, anggaran bulanan, dan atribut klasifikasi hotel yang kemudian digunakan untuk menghitung metrik seperti *Budget*, *Actual*, *Last Year*, *Growth*, dan *Achievement*.

b. *Process*

Perhitungan metrik pada *dashboard Hotel Revenue Statistic* melibatkan agregasi komponen tabel pendapatan hotel dari *dataset* mv_manager_report_with_budget_lastyear. Daftar kalkulasi metrik dan filter yang digunakan untuk membuat *dashboard Hotel Revenue Statistic* disajikan pada Tabel 3.17. Metrik *Budget* dihitung dari penjumlahan lima jenis pendapatan anggaran. Metrik *Actual* diambil dari *hotelrevenue*, sementara *Last Year* diperoleh dari *prev_hotelrevenue* sebagai pembanding. Metrik *Growth* dihitung dengan rumus (*Actual* - *Last Year*) / *Last Year* * 100 untuk mendapatkan persentase perubahan kinerja antar tahun. Metrik *Achievement* dihitung dari persentase pencapaian pendapatan aktual terhadap total pendapatan anggaran, dengan menggunakan logika *CASE WHEN* untuk menghindari pembagian dengan nol. Seluruh metrik difilter berdasarkan *cur_trxdate* dengan rentang waktu tertentu sesuai kebutuhan.

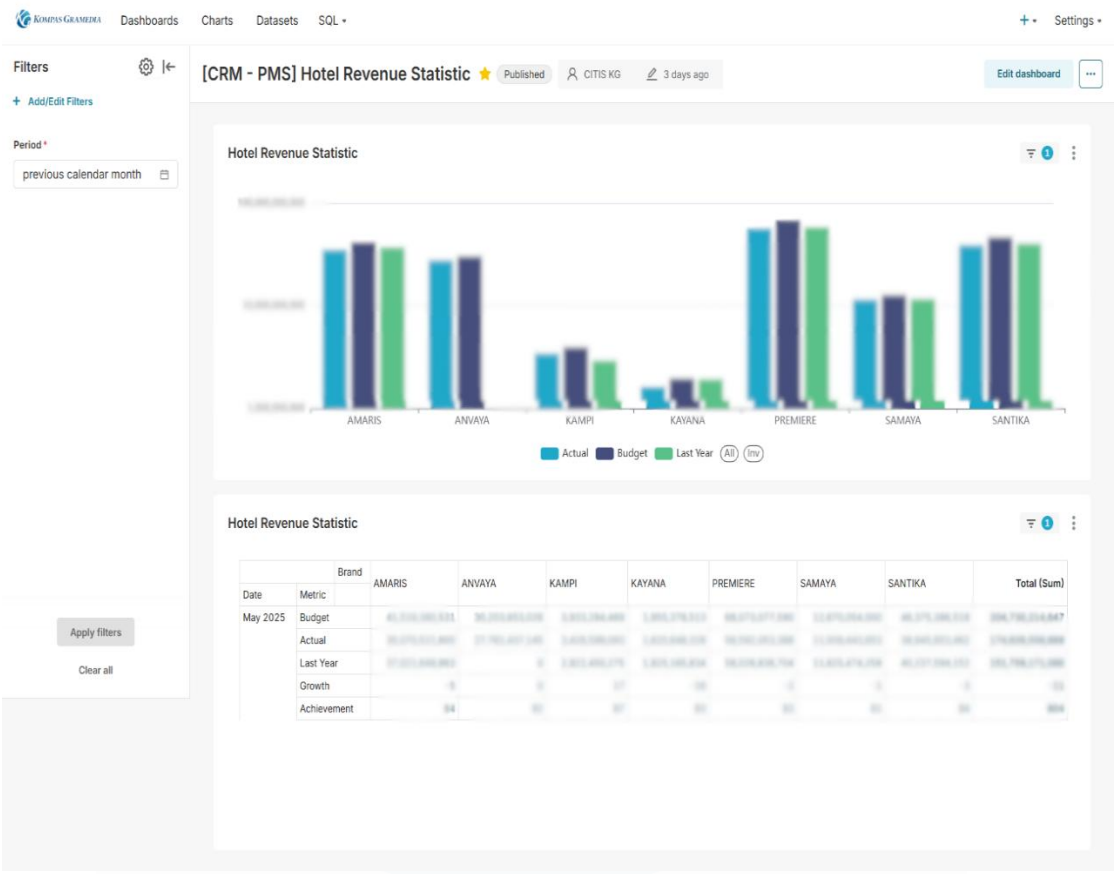
Tabel 3.18 Rincian Metriks dan Filter pada *Dashboard Hotel Revenue Statistic*

Nama *Chart*	Kalkulasi Metriks	Filter
Hotel Revenue Statistic (*Bar Chart*)	BUDGET SUM( )+SUM( )+SUM( )+SUM( )+SUM( ) column ACTUAL # hotelrevenue aggregate SUM column LAST YEAR # prev_hotelrevenue aggregate SUM	2025-05-01 ≤ cur_trxdate < 20...

Nama *Chart*	Kalkulasi Metriks	Filter
Hotel Revenue Statistic (*Pivot Table*)	BUDGET SUM(...)+SUM(...)+SUM(...)+SUM(...)+SUM(...) column ACTUAL: # hotelrevenue; aggregate: SUM column LAST YEAR: # prev_hotelrevenue; aggregate: SUM GROWTH ROUND(CASE WHEN SUM(...) = 0 OR SUM(...) IS NULL THEN 0.0 ELSE ((SUM(...) - SUM(...)) * 100.0 / SUM(...)) END, 2) ACHIEVEMENT ROUND(CASE WHEN (SUM(...) + SUM(...) + SUM(...) + SUM(...)) = 0 THEN 0.0 ELSE ((SUM(...) + SUM(...) + SUM(...) + SUM(...)) / (SUM(...) + SUM(...) + SUM(...) + SUM(...))) * 100.0) END, 2)	2025-05-01 ≤ cur_trxdate < 20...

Proses selanjutnya adalah pengembangan visualisasi pada *dashboard Hotel Revenue Statistic* menggunakan dua tipe chart yakni *Bar Chart* dan *Pivot Table*, yang dirancang melalui antarmuka Apache Superset. *Bar Chart* menampilkan perbandingan nilai *Budget*, *Actual*, *Last Year* berdasarkan *classcode* dalam bentuk batang vertikal untuk memudahkan identifikasi perbedaan kinerja antar jenis hotel. *Pivot Table* menyajikan data tabular dengan struktur baris *cur_trxdate* dan kolom *classcode*, serta menerapkan metrik yang telah dihitung sebelumnya pada setiap sel tabel. Fitur *Apply metrics on rows* pada *Pivot Table* mendukung perbandingan nilai metrik terhadap waktu dan jenis hotel secara bersamaan. Semua visualisasi disesuaikan dengan filter waktu tertentu dan akan di-*embed* ke dalam sistem PMS untuk memantau performa keuangan hotel.

c. *Output*



Gambar 3.99 *Output Dashboard Hotel Revenue Statistic*

Dashboard Hotel Revenue Statistic seperti yang terlihat pada Gambar 3.99 menyajikan visualisasi ringkasan performa pendapatan hotel berdasarkan *brand* hotel (*classcode*) untuk periode waktu tertentu menggunakan *bar chart* dan *pivot table*. *Bar chart* memperlihatkan perbandingan performa antar *classcode* untuk metrik *Budget*, *Actual*, *Last Year* yang memudahkan identifikasi perbedaan capaian pendapatan. *Pivot table* menyajikan informasi lebih granular dalam format tabel untuk *monitoring* dan pelacakan angka *Budget*, *Actual*, *Last Year*, *Growth*, dan *Achievement* berdasarkan waktu dan *brand* hotel. Seluruh visualisasi difilter berdasarkan rentang waktu tertentu dan hasil visualisasi akan di-*embed* dalam sistem PMS agar dapat diakses langsung oleh manajer hotel, supervisor operasional, dan tim *corporate* sebagai dasar pengambilan keputusan berbasis data aktual.

3.2.9.6. *CRM Menu Dashboard – Room Revenue Statistic*

Pembangunan *dashboard Room Revenue Statistic* bertujuan untuk membandingkan pendapatan kamar aktual, anggaran, dan performa tahun sebelumnya, serta menghitung pertumbuhan dan pencapaian anggaran untuk menganalisis kinerja pendapatan kamar hotel dalam periode tertentu. Berikut adalah input, *process*, dan *output* dari *dashboard Room Revenue Statistic*:

a. Input

Input pada *dashboard Room Revenue Statistic* menggunakan *dataset* yang sama dengan *Average Room Rate Statistic*, yaitu mv_manager_report_with_budget_lastyear dengan *query* seperti yang terlihat pada Gambar 3.96, yang dibentuk melalui pembuatan *materialized view* (MV) menggunakan perintah SQL pada PostgreSQL. Meskipun berasal dari sumber data dan struktur *query* yang sama, fokus penggunaan data pada *dashboard Room Revenue Statistic* berbeda, yaitu pada total pendapatan kamar (*roomrevenue*) yang dianalisis melalui tiga sudut pandang yakni capaian aktual, anggaran, dan performa tahun sebelumnya. Kolom yang digunakan dalam MV mencakup *roomrevenue*, *budgetroomrevenue*, dan *prev_roomrevenue*, yang dihitung secara bulanan. Semua input disiapkan dalam struktur *dataset* yang telah dioptimalkan dan difilter berdasarkan parameter waktu transaksi (*cur_trxdate*) tertentu.

b. *Process*

Perhitungan metrik pada *dashboard Room Revenue Statistic* dimulai dengan penyusunan formula kalkulasi untuk *dataset* mv_manager_report_with_budget_lastyear. Daftar kalkulasi metrik dan filter yang digunakan untuk membuat *dashboard Room Revenue Statistic* disajikan pada Tabel 3.19. Tiga metrik utama (*Budget*, *Actual*, *Last Year*) dihitung menggunakan *function* agregat *SUM()* terhadap kolom *budgetroomrevenue*, *roomrevenue*, dan *prev_roomrevenue*. Dua metrik evaluatif *Growth* dan *Achievement*, ditambahkan untuk mengukur persentase perubahan dan pencapaian target anggaran. Metrik *Growth*

dihitung dengan rumus (*Actual - Last Year*) / *Last Year* * 100 untuk mendapatkan persentase perubahan kinerja antar tahun. Metrik *Achievement* dihitung dari persentase pencapaian pendapatan aktual terhadap total pendapatan anggaran, dengan menggunakan logika *CASE WHEN* untuk menghindari pembagian dengan nol.

Tabel 3.19 Rincian Metriks dan Filter pada *Dashboard Room Revenue Statistic*

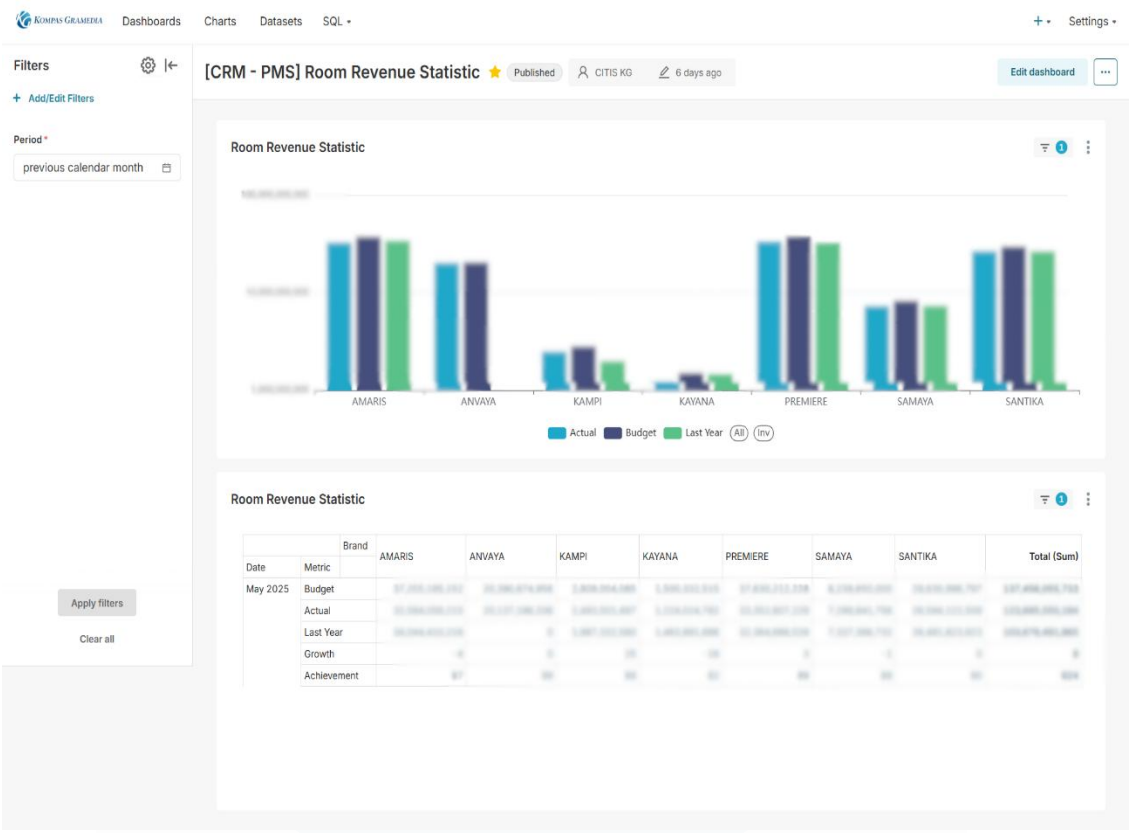
Nama *Chart*	Kalkulasi Metriks	Filter
Room Revenue Statistic (*Bar Chart*)	column **BUDGET** # budgetroomrevenue aggregate SUM column **ACTUAL** # roomrevenue aggregate SUM column **LAST YEAR** # prev_roomrevenue aggregate SUM	2025-05-01 ≤ cur_trxdate < 20...
Room Revenue Statistic (*Pivot Table*)	column **BUDGET** # budgetroomrevenue aggregate SUM column **ACTUAL** # roomrevenue aggregate SUM column **LAST YEAR** # prev_roomrevenue aggregate SUM **GROWTH** ROUND(CASE WHEN SUM(...) = 0 OR SUM (...) IS NULL THEN 0.0 ELSE ((SUM(...) - SUM (...)) * 100.0 / SUM (...)) END, 2) **ACHIEVEMENT** ROUND(CASE WHEN SUM(...) = 0 OR SUM (...) IS NULL THEN 0.0 ELSE ((SUM(...) / SUM (...) * 100.0) END, 2)	2025-05-01 ≤ cur_trxdate < 20...

Proses selanjutnya adalah pengembangan visualisasi pada *dashboard Room Revenue Statistic* menggunakan dua jenis grafik yakni *Bar Chart* dan *Pivot Table*, keduanya dirancang melalui Apache Superset. *Bar Chart*

membandingkan nilai anggaran, capaian aktual, dan performa tahun sebelumnya secara horizontal berdasarkan pengelompokan *classcode*. Sumbu X menggunakan *classcode* untuk membedakan jenis hotel, sementara sumbu Y menampilkan nilai agregat dari ketiga metrik utama (*Budget*, *Actual*, *Last Year*). *Pivot Table* menyajikan data dalam format tabel dinamis, dengan *cur_trxdate* sebagai baris dan *classcode* sebagai kolom, serta metrik *Growth* dan *Achievement* diterapkan pada baris untuk memberikan tampilan numerik yang lengkap. Visualisasi ini mendukung analisis cepat terhadap tren pendapatan kamar dan evaluasi performa berdasarkan klasifikasi hotel.

c. *Output*

Dashboard Room Revenue Statistic seperti yang terlihat pada Gambar 3.100 menghasilkan visualisasi yang menunjukkan kinerja pendapatan kamar hotel berdasarkan kategori *classcode* untuk periode waktu tertentu. Visualisasi terdiri dari dua *chart* yakni *Bar Chart* yang membandingkan agregat pendapatan aktual (*Actual*), target anggaran (*Budget*), dan capaian tahun sebelumnya (*Last Year*), serta *Pivot Table* yang menampilkan rincian data *Budget*, *Actual*, *Last Year* dengan persentase pertumbuhan (*Growth*) dan tingkat pencapaian anggaran (*Achievement*). Semua data difilter berdasarkan *cur_trxdate* untuk memastikan hanya data periode waktu tertentu yang ditampilkan. *Bar Chart* memudahkan perbandingan nominal pendapatan antar kategori hotel, sementara *Pivot Table* menyediakan analisis numerik yang lebih spesifik. Visualisasi ini akan diintegrasikan ke dalam sistem *Property Management System* (PMS) dan dapat diakses oleh manajemen operasional, seperti *Hotel* Manajer dan *Board of Directors*, untuk evaluasi pendapatan, identifikasi penyimpangan terhadap target anggaran, dan penyusunan strategi optimalisasi pendapatan kamar di periode berikutnya.



Gambar 3.100 *Output Dashboard Room Revenue Statistic*

3.2.9.7. *CRM Menu Dashboard – Geographical Origin Of Business*

Pembangunan *dashboard Geographical Origin Of Business* bertujuan untuk menganalisis kontribusi asal geografis tamu hotel terhadap total okupansi kamar, dengan memisahkan data berdasarkan wilayah domestik dan internasional dalam periode waktu tertentu. Berikut adalah input, *process*, dan *output* dari *dashboard Geographical Origin Of Business*:

a. Input

Input pada *dashboard Geographical Origin Of Business* menggunakan *dataset* mv_geographical_origin, yang mengumpulkan data asal geografis tamu hotel berdasarkan tingkat okupansi kamar (*room night*). *Dataset* dengan *query* SQL seperti pada Gambar 3.101 ini dibentuk melalui agregasi data dari tabel mv_roomcountsheetguestprofile dalam sistem PMS, dengan *filtering* untuk mengabaikan entri bertipe *ratetype* = 'HOU'. Agregasi dilakukan berdasarkan kombinasi atribut *hotelid*, *classcode*,

countrydescription, *cityid*, dan *cityname* dalam rentang waktu yang ditentukan. Selanjutnya, dibuat *materialized view* yang membandingkan kinerja *room night* pada bulan berjalan (*cur_trxdate*) dengan bulan yang sama tahun sebelumnya (*prev_trxdate*) berdasarkan hotel dan kota asal tamu. Proses ini menghasilkan *dataset* yang menyimpan informasi aktual dan historis kontribusi kota terhadap total okupansi, baik untuk tamu domestik maupun internasional.

```
CREATE MATERIALIZED VIEW public.mv_geographical_origin
TABLESPACE pg_default
AS WITH actual_summary AS (
         SELECT mv_roomcountsheetguestprofile.hotelid,
            date_trunc('month'::text, mv_roomcountsheetguestprofile.trxdate)::date AS trxdate,
            count(*) AS roomnight,
            mv_roomcountsheetguestprofile.countrydescription,
            mv_roomcountsheetguestprofile.cityid,
            mv_roomcountsheetguestprofile.cityname,
            mv_roomcountsheetguestprofile.classcode,
            mv_roomcountsheetguestprofile.hotelname
           FROM mv_roomcountsheetguestprofile
          WHERE mv_roomcountsheetguestprofile.ratetype::text <> 'HOU'::text
          GROUP BY (date_trunc('month'::text, mv_roomcountsheetguestprofile.trxdate)::date),
mv_roomcountsheetguestprofile.hotelid, mv_roomcountsheetguestprofile.countrydescription,
mv_roomcountsheetguestprofile.cityid, mv_roomcountsheetguestprofile.cityname,
mv_roomcountsheetguestprofile.classcode, mv_roomcountsheetguestprofile.hotelname
        )
 SELECT a.hotelid,
    a.trxdate AS cur_trxdate,
    a.classcode,
    a.hotelname,
    a.countrydescription,
    a.cityid,
    a.cityname,
    COALESCE(a.roomnight::numeric, 0::numeric) AS roomnight,
    date_trunc('month'::text, a.trxdate - '1 year'::interval)::date AS prev_trxdate,
    COALESCE(b.roomnight::numeric, 0::numeric) AS prev_roomnight
   FROM actual_summary a
     LEFT JOIN actual_summary b ON a.hotelid = b.hotelid AND (a.trxdate - '1 year'::interval) = b.trxdate
AND a.cityid = b.cityid
WITH DATA;
```

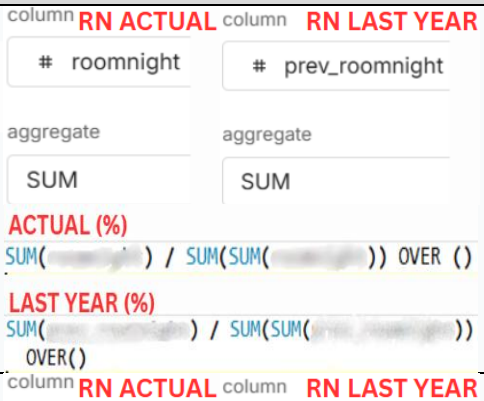
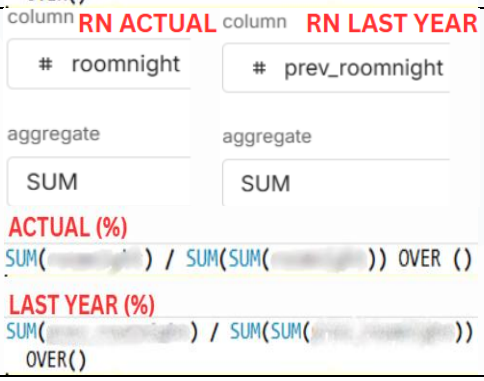
Gambar 3.101 *Query Materialized View* mv_geographical_origin

b. *Process*

Perhitungan metrik pada *dashboard Geographical Origin Of Business* dimulai dengan menghitung total *room night* aktual dan tahun sebelumnya menggunakan fungsi agregat *SUM(roomnight)* dan *SUM(prev_roomnight)* untuk masing-masing kota. Daftar kalkulasi metrik dan filter yang digunakan untuk membuat *dashboard Geographical Origin Of Business* disajikan pada Tabel 3.20. Selanjutnya, kontribusi setiap kota terhadap total okupansi dihitung melalui persentase dengan *function OVER ()*. Perhitungan ini dilakukan di Apache Superset menggunakan *custom metric builder* dan *window function* untuk menghitung proporsi masing-masing

kota. Semua metrik difilter berdasarkan waktu transaksi tertentu, *brand* hotel (*classcode*), dan hotel *name* agar visualisasi relevan dengan konteks unit hotel terkait.

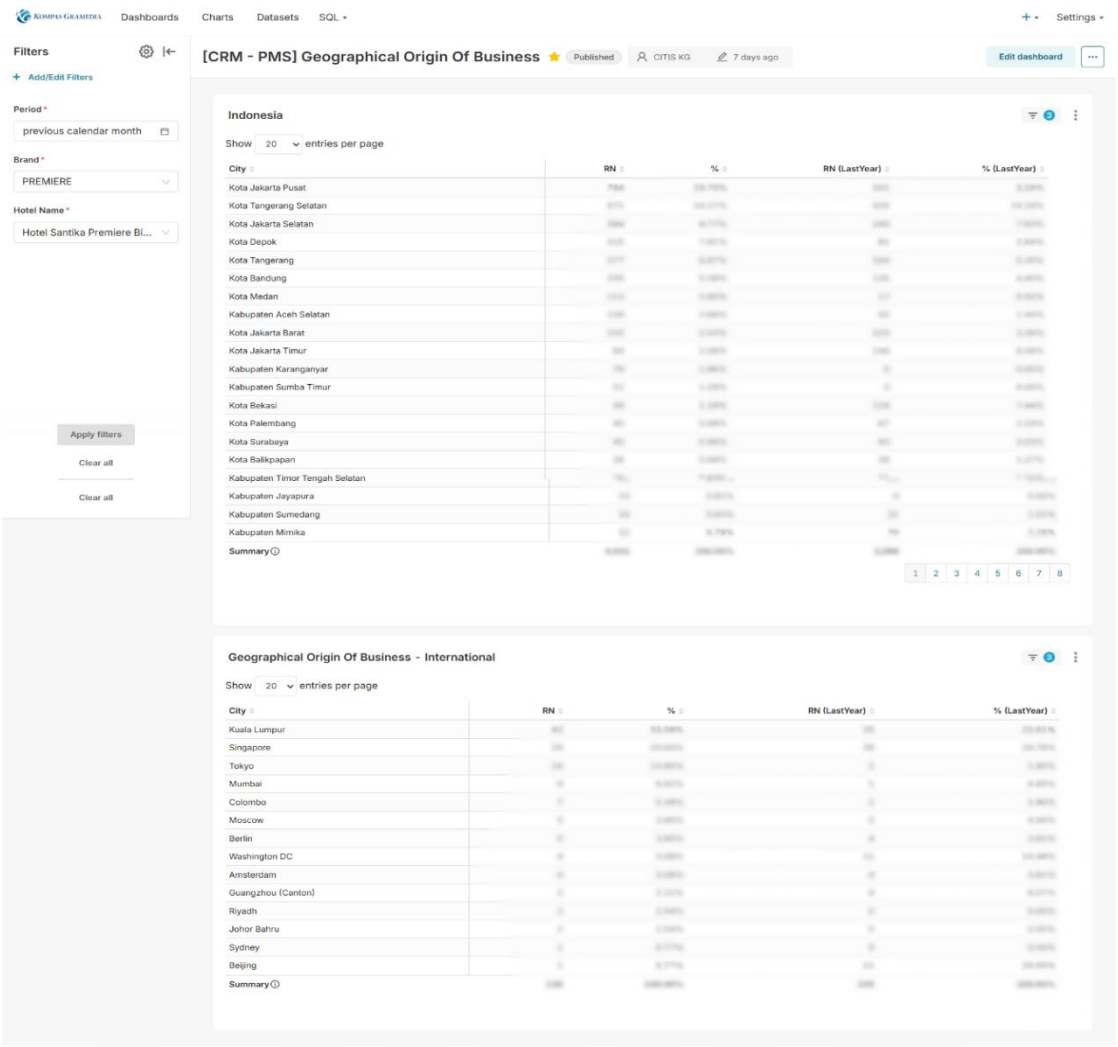
Tabel 3.20 Rincian Metriks dan Filter pada *Dashboard Geographical Origin Of Business*

Nama *Chart*	Kalkulasi Metriks	Filter
Geographical Origin Of Business - Indonesia (*Table*)		
Geographical Origin Of Business - International (*Table*)		

Proses selanjutnya adalah pengembangan visualisasi pada *dashboard Geographical Origin of Business* menggunakan tabel untuk mengorganisasi hasil analisis berdasarkan kategori wilayah yang dibagi menjadi dua bagian yakni Indonesia dan *International*. Setiap tabel menampilkan daftar kota yang diurutkan berdasarkan kontribusinya terhadap total *room night*, dengan *cityname* sebagai dimensi utama yang disusun vertikal menggunakan *chart table* di Apache Superset. Visualisasi ini menampilkan empat metrik utama yakni *Room Night Actual*, *Actual* (%), *Room Night Last Year*, dan *Last Year* (%). Filter diterapkan untuk memisahkan data domestik dan internasional berdasarkan *countrydescription*, serta filter waktu dan *brand* hotel diterapkan sesuai dengan kebutuhan pengguna. Hasil visualisasi

ini digunakan untuk mengevaluasi kontribusi geografis tamu hotel terhadap total okupansi berdasarkan asal wilayah.

c. Output



Gambar 3.102 *Output Dashboard Geographical Origin Of Business*

Dashboard Geographical Origin of Business seperti yang terlihat pada Gambar 3.102 menyajikan visualisasi dalam bentuk tabel yang menggambarkan kontribusi kota-kota asal tamu terhadap okupansi kamar hotel dalam periode waktu tertentu. *Output* ini dibagi menjadi dua bagian yakni domestik dan internasional. Setiap bagian menampilkan *Room Night Actual*, *Room Night Last Year*, serta persentasenya terhadap total berdasarkan metrik *SUM(roomnight)* dan *SUM(prev_roomnight)*. Data

disusun berdasarkan nama kota (*cityname*) untuk memudahkan identifikasi wilayah dengan kontribusi tinggi atau penurunan kinerja dibandingkan tahun sebelumnya. Visualisasi ini memberikan wawasan terkait sebaran pasar tamu lokal dan internasional, dan hasil dashboard akan diintegrasikan langsung ke dalam sistem *Property Management System* (PMS) untuk diakses oleh manajer operasional, tim *front office*, dan manajemen hotel. Tujuannya adalah mendukung pemantauan performa okupansi wilayah secara *real-time* dan memperkuat pengambilan keputusan terkait strategi promosi dan segmentasi pasar berdasarkan asal tamu.

3.2.10. *Embedding Dashboards* – *Deploy Dashboard* ke Sistem PMS Menggunakan API dan SDK

Pemanfaatan *Application Programming Interface* (API) dalam proses *embedding dashboard* berperan penting sebagai jembatan komunikasi antara aplikasi eksternal dan sistem Apache Superset. API berfungsi untuk mengautentikasi akses pengguna, meminta token otorisasi, serta mengambil data atau *resource* yang akan ditampilkan dalam antarmuka sistem lain seperti PMS (*Property Management System*). Kode terkait proses *embedding dashboard* menggunakan API dapat dilihat pada Gambar 3.104, yang mana dalam proses integrasi ini, API Superset digunakan untuk melakukan permintaan login dengan metode POST ke *endpoint* server, yang kemudian memberikan akses token sebagai identitas akses. Setelah memperoleh akses token, sistem melakukan permintaan token CSRF (*Cross Site Request Forgery*) melalui *endpoint* server yang dibutuhkan untuk melakukan permintaan lanjut seperti pengambilan *guest token*. *Guest token* ini diperlukan agar pengguna dapat mengakses *dashboard* Superset yang di-*embed* tanpa login langsung ke Superset, namun tetap dalam otorisasi yang dikendalikan. Semua komunikasi dilakukan secara *secured* dengan protokol HTTPS dan format data JSON, serta disertai *header* otentikasi yang sesuai standar API modern. Dalam proyek *embedding dashboard* pada sistem PMS, proses login, otorisasi, dan pengambilan *guest token* menggunakan alur ini dijalankan melalui *script* berbasis *axios* yang terintegrasi langsung dengan kode *backend*

sistem. Tahapan ini menjadi fondasi penting sebelum *dashboard* dapat ditampilkan secara utuh melalui mekanisme *Software Development Kit* (SDK) pada sisi *frontend*.

```
async function getSupersetGuestTokenViaOldEndpoint(password, provider, username) {
    try {
        // 1. access token
        const pmsAuthToken = getAPIToken();
        const apiBaseUrl = getAPIBaseURL();
        const response = await axios.post(
            'https://analytics.hospitality.citilink.com/api/v1/security/login',
            { password: 'pms123', provider:'db' , username: 'pms' },
            {
                headers: {
                    'Authorization': 'Bearer ' + pmsAuthToken,
                    'Content-Type': 'application/json'
                }
            }
        );
        const accessToken = response.data.access_token;
        if (!accessToken) {
            throw new Error("Access token not found in PMS backend response.");
        }
        console.log("Access token received:", accessToken.substring(0, 10) + "...");

        // 2. CSRF token
        const csrfResponse = await axios.get(
            `${SUPERSET_DOMAIN}/api/v1/security/csrf_token/`,
            {
                headers: {
                    'Authorization': `Bearer ${accessToken}`,
                    'Content-Type': 'application/json',
                    'Accept': 'application/json'
                },
                withCredentials: true
            }
        );
        const csrfToken = csrfResponse.data?.result;
        if (!csrfToken) {
            console.error("CSRF response:", JSON.stringify(csrfResponse.data, null, 2));
            console.error("CSRF headers:", JSON.stringify(csrfResponse.headers, null, 2));
            throw new Error("CSRF token not found in response.");
        }

    // 3. guest token
    const supersetResponse = await axios.post(
        `${SUPERSET_DOMAIN}/api/v1/security/guest_token/`,
        {
            user: {
                username: "",
                first_name: "",
                last_name: ""
            },
            resources: [{
                type: "dashboard",
                id: DASHBOARD_ID
            }],
            rls: []
        },
        {
            headers: {
                'Authorization': `Bearer ${accessToken}`,
                'Content-Type': 'application/json',
                'X-CSRFToken': csrfToken,
                'Accept': 'application/json',
                ...(sessionCookie && { 'Cookie': Array.isArray(sessionCookie) ? sessionCookie.join('; ') : sessionCookie })
            },
            withCredentials: true
        }
    );
    const guestToken = supersetResponse.data?.token;
    if (!guestToken) {
        console.error("Guest token response:", JSON.stringify(supersetResponse.data, null, 2));
        throw new Error("Guest token not found in response.");
    }
    console.log("Guest token received:", guestToken.substring(0, 10) + "...");
    return guestToken;
```

Gambar 3.103 *Code Embedding Dashboard with* API *GuestToken*

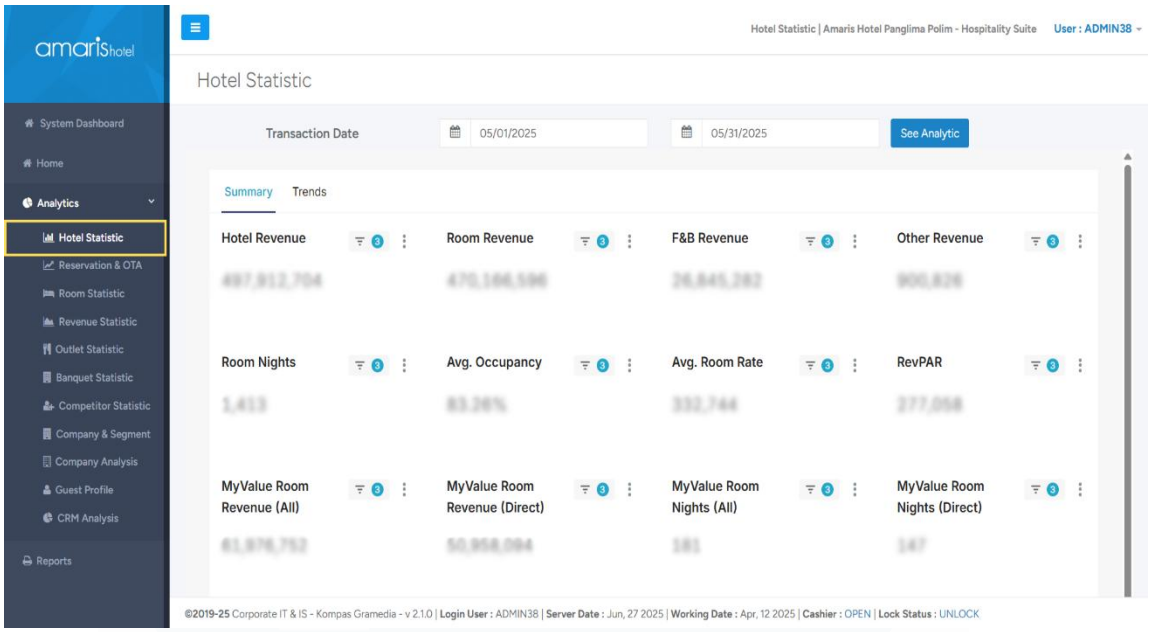
Penggunaan *Software Development Kit* (SDK) dari Superset menjadi komponen utama dalam proses visualisasi *dashboard* secara langsung di dalam sistem PMS setelah proses otorisasi API selesai dilakukan. SDK Superset bekerja dengan prinsip *client-side rendering* menggunakan *library* JavaScript bernama *@superset-ui/embedded-sdk* yang menyediakan *function embedDashboard* untuk menyisipkan *dashboard* ke dalam elemen HTML tertentu pada halaman web. *Function* tersebut memerlukan beberapa parameter penting seperti ID *dashboard*, *domain* Superset, *mount point* elemen HTML, token otorisasi dari API, konfigurasi tampilan (misalnya menyembunyikan judul atau *tab*), serta parameter URL tambahan seperti nama hotel, kelas hotel, dan rentang waktu transaksi yang dikirim dari *form input* di sistem PMS. Kode terkait proses *embedding dashboard* menggunakan SDK dapat dilihat pada Gambar 3.104. Dalam kasus proyek di Kompas Gramedia, *embedding* dilakukan ke sistem PMS milik unit hotel menggunakan komponen HTML div sebagai kontainer *dashboard*, dengan alur *on-click* tombol yang memicu proses pengambilan token dan pemanggilan *function embedDashboard*. Seluruh interaksi dilakukan secara dinamis menggunakan jQuery dan moment.js untuk mengelola format tanggal. Dengan pendekatan ini, *dashboard* Superset dapat ditampilkan langsung di PMS tanpa memerlukan login terpisah, mendukung kebutuhan visualisasi lintas peran di level operasional maupun manajerial hotel, serta memastikan pengalaman pengguna yang lebih ringkas dan terintegrasi antar sistem internal.

```
<script src="https://unpkg.com/%40superset-ui/embedded-sdk"></script>

try {
    const guestToken = await getSupersetGuestTokenViaOldEndpoint(
    await supersetEmbeddedSdk.embedDashboard({
        id: DASHBOARD_ID,
        supersetDomain: SUPERSET_DOMAIN,
        mountPoint: document.getElementById("dashboard-container"),
        fetchGuestToken: () => Promise.resolve(guestToken),
```

Gambar 3.104 *Code Embedding Dashboard with* SDK Superset

3.2.10.1. *Embedding Dashboards – Deploy Dashboard Hotel Statistic*

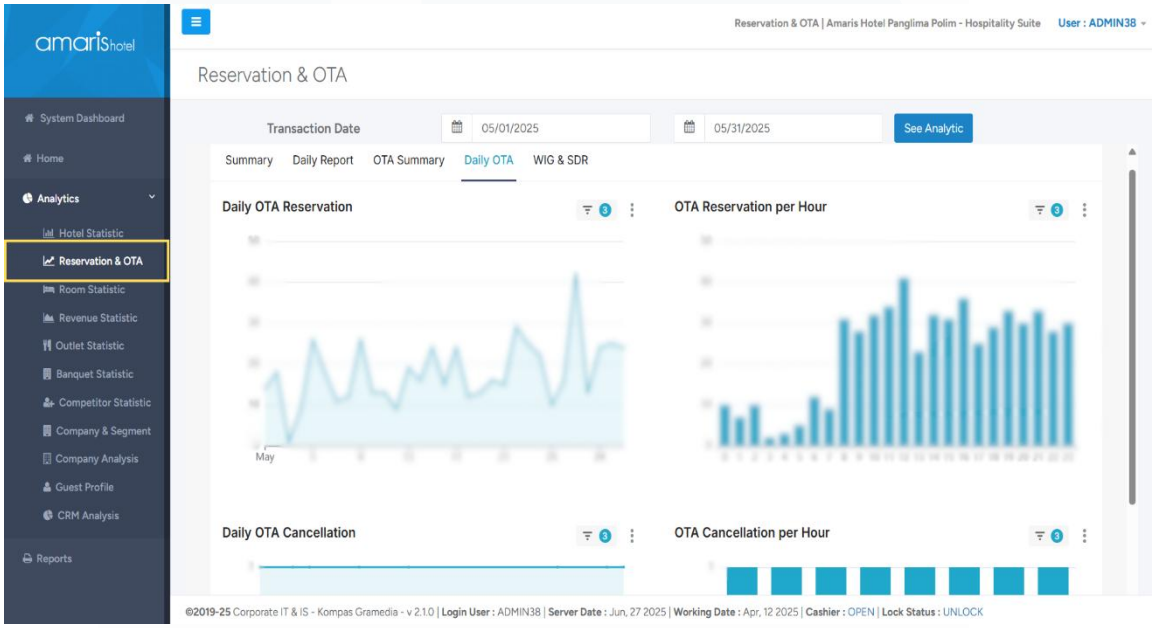


Gambar 3.105 Hasil *Deployment Dashboard Hotel Statistic*

Deployment dashboard Hotel Statistic ke dalam sistem *Property Management System* (PMS) dilakukan melalui proses *embedding* menggunakan API dan SDK dari Apache Superset, yang diintegrasikan ke menu *Analytics & Reports*. Proses ini memanfaatkan fitur *embedded dashboard* Superset dengan DASHBOARD_ID yang digunakan dalam kode *frontend* berbasis JavaScript melalui *library @superset-ui/embedded-sdk* di Microsoft Visual Studio. *Dashboard* berinteraksi menggunakan *library axios* untuk komunikasi API dan *moment.js* untuk pengelolaan format tanggal. Struktur kode mirip dengan konsep *embedding* seperti yang telah dijelaskan sebelumnya dan terlihat pada Gambar 3.103 dan Gambar 3.104, dengan perbedaan pada parameter yang digunakan, seperti DASHBOARD_ID untuk *Hotel Statistic* dan input parameter seperti *param_time_range*, *param_hotel_name*, dan *param_hotel_class*. Proses dimulai dengan otorisasi token API Superset, pengambilan CSRF token, dan pembuatan *guest token* untuk akses *embed dashboard*. Fitur filter dinamis membantu pemilihan periode waktu tanpa keluar dari PMS. Hasil *embed dashboard* ini memudahkan manajer hotel, supervisor, dan staf administrasi dalam

menganalisis data keuangan dan performa hotel secara terpadu, mempercepat pengambilan keputusan tanpa mengakses Superset secara terpisah. Visualisasi hasil akhir *deployment dashboard Hotel Statistic* dapat dilihat pada Gambar 3.105.

3.2.10.2. *Embedding Dashboards – Deploy Dashboard Reservation* & OTA

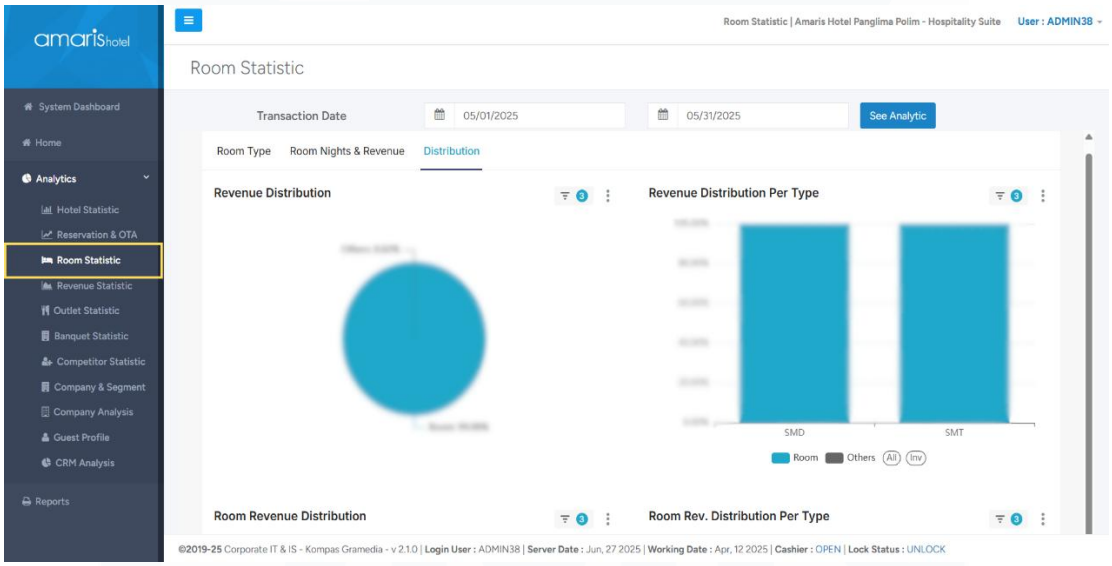


Gambar 3.106 Hasil *Deployment Dashboard Reservation* & OTA

Embedding Reservation & OTA *Dashboard* ke dalam sistem *Property Management System* (PMS) dilakukan melalui integrasi Apache Superset ke menu *Analytics & Reports* di PMS menggunakan API dan SDK. Proses dimulai dengan pengambilan DASHBOARD_ID dari fitur *embed* Superset, yang dimasukkan ke dalam kode JavaScript di Microsoft Visual Studio menggunakan *library @superset-ui/embedded-sdk*. Autentikasi dilakukan melalui login Superset API untuk memperoleh *access* token dan CSRF token, diikuti dengan permintaan *guest* token agar *dashboard* dapat diakses tanpa login ulang. Filter dinamis seperti tanggal transaksi menggunakan *library* moment.js dan *datepicker* akan membantu pengguna memilih periode untuk memvisualisasikan data reservasi dan OTA (*Online Travel Agent*). Struktur kode *embedding* ini mirip dengan proses *embed dashboard* lainnya, seperti yang telah dijelaskan sebelumnya dan terlihat pada Gambar 3.103 dan Gambar

3.104 dengan penyesuaian pada parameter URL dan ID *dashboard*. *Dashboard* ini menyajikan metrik seperti *Total Reservation*, *Group Reservation*, *Cancellation*, *Avg*. *Stay Period*, dan lainnya, serta tren harian dalam submenu seperti *Daily Report*, OTA *Summary*, dan *Daily* OTA. Data berasal dari sistem reservasi hotel yang telah diproses ETL dan divisualisasikan dalam *big number chart*, *line chart*, *area chart*, *bar chart*, dan *table*. *Dashboard* interaktif ini diakses langsung oleh pengguna PMS, seperti *front office* supervisor dan *revenue* manajer, untuk memantau performa reservasi dan OTA secara menyeluruh, mempercepat respon terhadap pola pembatalan, dan mendukung perencanaan promosi OTA. Visualisasi hasil akhir *deployment dashboard Reservation* & OTA dapat dilihat pada Gambar 3.106.

3.2.10.3. *Embedding Dashboards – Deploy Dashboard Room Statistic*

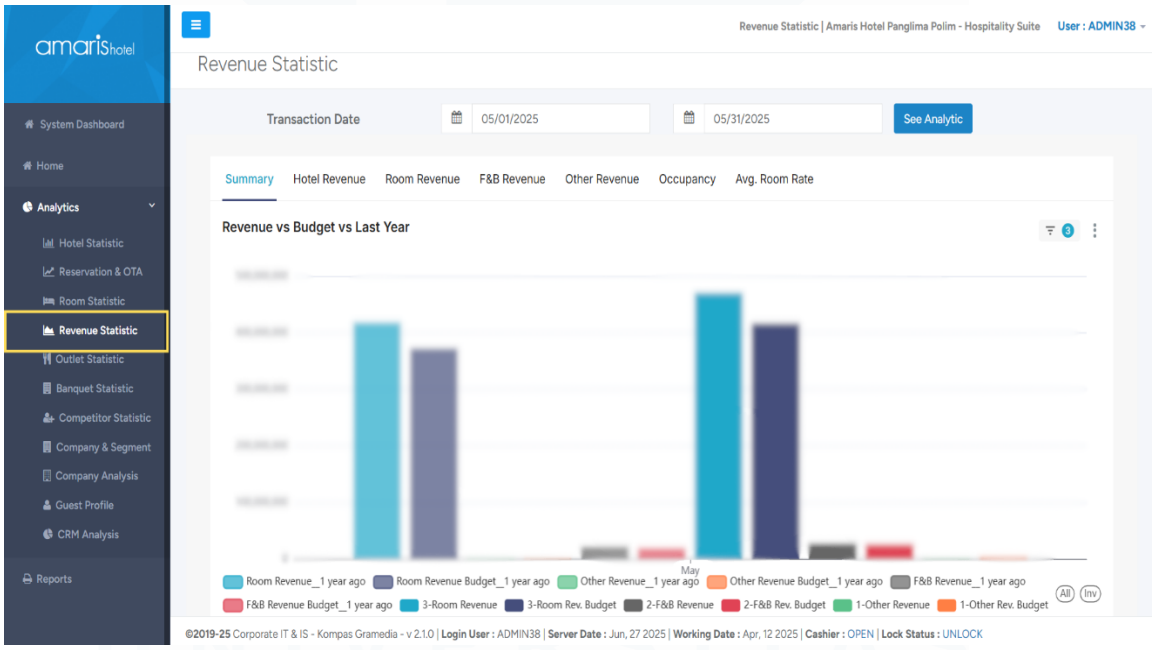


Gambar 3.107 Hasil *Deployment Dashboard Room Statistic*

Embedding Room Statistic Dashboard dilakukan dengan mengintegrasikan visualisasi dari Apache Superset ke dalam sistem PMS pada menu *Analytics & Reports* menggunakan API dan SDK yang mendukung autentikasi token dan parameter dinamis. Proses dimulai dengan pengambilan DASHBOARD_ID dari fitur "*Embed this dashboard*" di Superset, yang kemudian dimasukkan ke dalam *script* JavaScript di Microsoft Visual Studio menggunakan *library @superset-ui/embedded-sdk*. Akses ke API Superset

diperlukan untuk memperoleh *access* token, CSRF token, dan *guest* token agar *dashboard* dapat ditampilkan langsung dalam PMS tanpa login terpisah. *Dashboard* ini dilengkapi dengan filter dinamis untuk rentang tanggal transaksi, hotel *brand*, dan hotel *name* yang membantu pengguna memilih periode analisis distribusi pendapatan kamar berdasarkan tipe dan jenis *room night* di hotel tertentu. Struktur kode mirip dengan konsep *embedding* seperti yang telah dijelaskan sebelumnya dan terlihat pada Gambar 3.103 dan Gambar 3.104 dengan perbedaan pada parameter URL dan ID *dashboard* sesuai dengan nama dan isi visualisasi. Manfaat utama *embedding* ini adalah menyajikan data pendapatan kamar secara *real-time* dan interaktif di sistem PMS agar dapat diakses oleh staf operasional hotel, mendukung analisis performa per tipe kamar, dan pola kontribusi pendapatan harian. Visualisasi hasil akhir *deployment dashboard Room Statistic* dapat dilihat pada Gambar 3.107.

3.2.10.4. *Embedding Dashboards – Deploy Dashboard Revenue Statistic*



Gambar 3.108 Hasil *Deployment Dashboard Revenue Statistic*

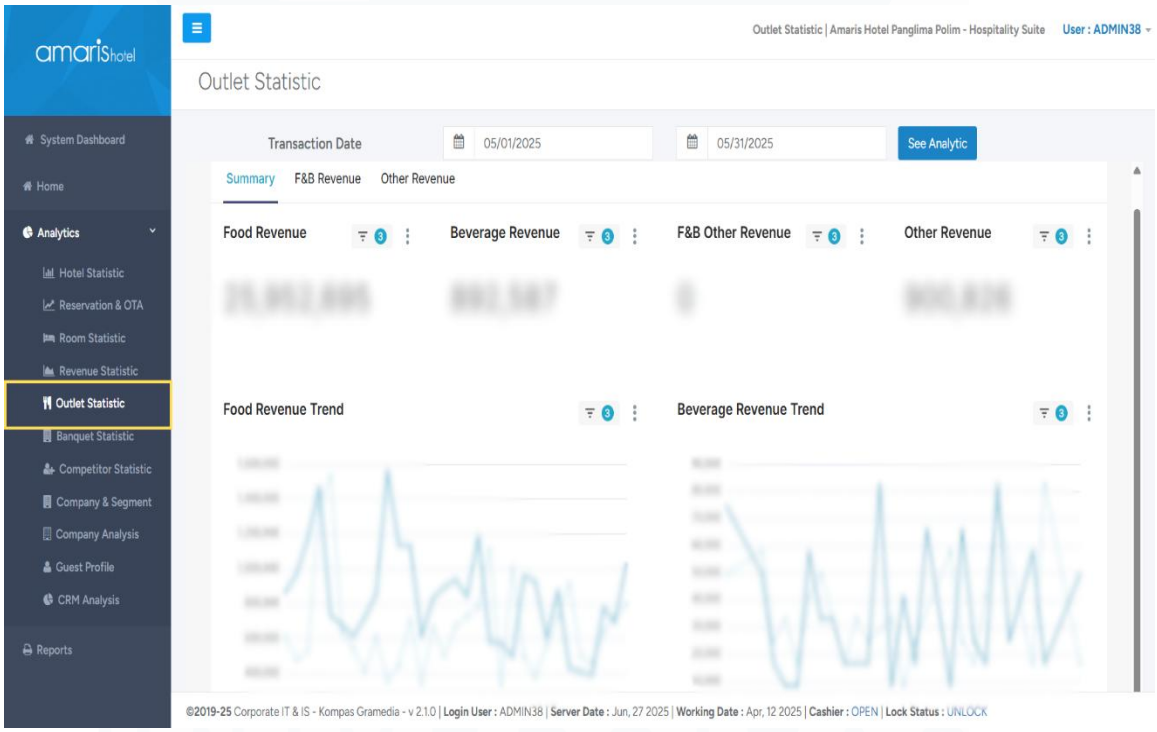
Deployment Revenue Statistic Dashboard dilakukan dengan menyisipkan tampilan visualisasi Superset ke dalam sistem PMS menggunakan API dan SDK Superset yang diintegrasikan dalam kode berbasis

Visual Studio. *Dashboard* ini diintegrasikan ke dalam menu *Analytics & Reports* pada sistem PMS hotel KG *Hospitality* dan dirancang untuk menampilkan perbandingan pendapatan aktual, anggaran, dan capaian tahun sebelumnya. Proses *embedding* dimulai setelah *dashboard* dipublikasikan di Apache Superset, kemudian DASHBOARD_ID diperoleh melalui fitur *Embed Dashboard* dari Superset, yang menghasilkan UUID untuk referensi dalam kode integrasi. Proses ini dimulai dengan mengakses *endpoint* login API Superset untuk memperoleh *access* token, CSRF token, dan *guest* token yang diperlukan oleh SDK untuk merender *dashboard* di halaman HTML. Pengguna dapat memilih rentang tanggal transaksi melalui komponen input tanggal yang terhubung dengan parameter *param_time_range* pada Superset, serta filter seperti klasifikasi hotel dan nama hotel pada URL *dashboard*. Visualisasi interaktif yang dihasilkan menampilkan grafik *Revenue* vs *Budget* vs *Last Year*, mencakup kategori pendapatan seperti *Room Revenue*, F&B *Revenue*, dan *Other Revenue*, yang disandingkan dengan nilai anggaran dan historis tahun sebelumnya. Hasil ini memberikan gambaran kinerja keuangan yang responsif untuk mendukung pengambilan keputusan manajerial hotel. Visualisasi hasil akhir *deployment dashboard Revenue Statistic* dapat dilihat pada Gambar 3.108.

3.2.10.5. *Embedding Dashboards – Deploy Dashboard Outlet Statistic*

Deployment Outlet Statistic Dashboard dilakukan dengan menyisipkan visualisasi data dari Apache Superset ke dalam sistem PMS menggunakan API dan SDK Superset yang diintegrasikan dalam *platform* Visual Studio. *Dashboard* ini ditempatkan pada menu *Analytics & Reports* untuk menampilkan informasi pendapatan *Food Revenue*, *Beverage Revenue*, F&B *Other Revenue*, dan *Other Revenue* dalam format angka dan grafik tren. Pengguna menentukan periode transaksi melalui *date picker*, yang diteruskan sebagai parameter *param_time_range* dalam proses *embedding*. Proses dimulai dengan pengambilan DASHBOARD_ID dari fitur *Embed Dashboard* Superset, yang menghasilkan UUID untuk digunakan dalam kode integrasi, dengan otentikasi dilakukan melalui API Superset untuk memperoleh *access*

token, CSRF token, dan *guest* token. Visualisasi dirender menggunakan metode *embedDashboard()* dari SDK Superset dan disisipkan dalam elemen HTML pada halaman target. Struktur kode mirip dengan konsep *embedding* seperti yang telah dijelaskan sebelumnya dan terlihat pada Gambar 3.103 dan Gambar 3.104, namun disesuaikan dengan parameter dan konfigurasi untuk *Outlet Statistic*. *Output dashboard* ini menunjukkan total pendapatan berdasarkan kategori serta grafik tren harian untuk *Food Revenue*, *Beverage Revenue*, dan *Other Revenue* yang dapat diakses langsung oleh pengguna sistem PMS. Implementasi *dashboard* ini memberikan gambaran menyeluruh terhadap performa F&B *outlet* hotel dalam tampilan responsif dan terhubung langsung dengan data terkini. Visualisasi hasil akhir *deployment dashboard Outlet Statistic* dapat dilihat pada Gambar 3.109.

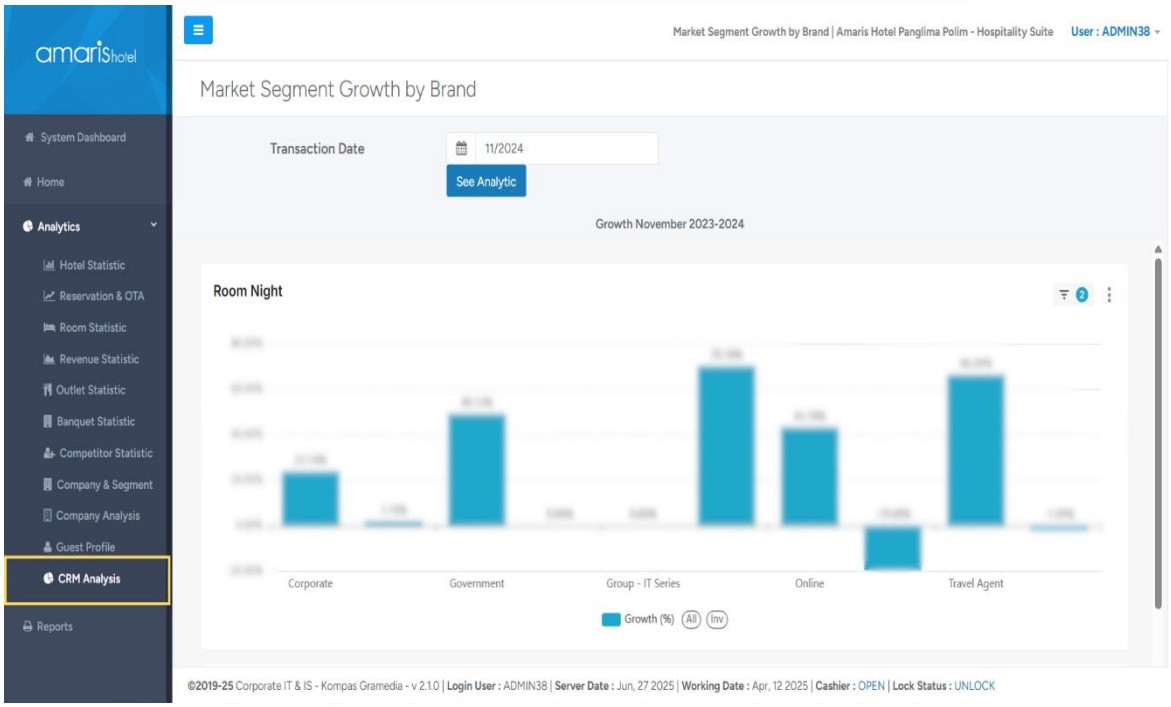


Gambar 3.109 Hasil *Deployment Dashboard Outlet Statistic*

3.2.10.6. *Embedding Dashboards – Deploy Dashboard* CRM

Deployment CRM - *Market Segment Growth by Brand Dashboard* dilakukan melalui integrasi Apache Superset ke dalam sistem PMS menggunakan metode *embedding* dengan kombinasi API dan SDK Superset,

dan dikembangkan di *platform* Microsoft Visual Studio. *Dashboard* ini ditempatkan di menu *Analytics & Reports* dan menyajikan metrik pertumbuhan segmen pasar berdasarkan *brand* hotel, khususnya untuk *Room Night* yang dibandingkan antara tahun tertentu. Pengguna dapat memilih bulan dan tahun transaksi melalui *date picker*, yang dikirim sebagai parameter *param_time_range* untuk menyesuaikan periode visualisasi. DASHBOARD_ID diperoleh melalui fitur *Embed Dashboard* Superset, yang menghasilkan UUID unik untuk identifikasi dalam kode integrasi. Proses autentikasi melibatkan pengambilan *access* token melalui API Superset, CSRF token, dan pembuatan *guest* token yang digunakan oleh SDK Superset untuk merender *dashboard* di halaman sistem PMS. *Output*-nya berupa grafik batang yang menunjukkan nilai pertumbuhan untuk masing-masing segmen pasar dengan persentase positif dan negatif yang menggambarkan performa *brand*. Hasil visualisasi ini digunakan oleh tim manajemen untuk menganalisis tren pertumbuhan segmen pasar, yang berdampak pada strategi pemasaran dan segmentasi pelanggan. Visualisasi hasil akhir *deployment dashboard* CRM - *Market Segment Growth by Brand* dapat dilihat pada Gambar 3.110.



Gambar 3.110 Hasil *Deployment Dashboard* CRM

Seluruh *dashboard* yang dikembangkan untuk kebutuhan unit bisnis *hospitality* telah berhasil diintegrasikan ke dalam sistem PMS pada menu "*Analytics & Reports*", dan saat ini telah digunakan secara aktif oleh Kompas Gramedia, khususnya oleh *Group of Hotel & Resorts*. *Dashboard* ini dimanfaatkan oleh *Board of Directors* serta para manajer hotel sebagai alat bantu dalam proses pengambilan keputusan berbasis data, yang mencakup analisis performa hotel, pendapatan, serta berbagai metrik operasional lainnya. Pemanfaatan *dashboard* ini turut dibahas dalam *meeting* bersama setiap dua minggu sekali dengan para *stakeholder* seperti yang ditunjukkan pada Gambar 3.111, dimana mereka mengevaluasi hasil *dashboard* untuk menentukan langkah-langkah strategis yang perlu diambil ke depannya.

Gambar 3.111 Bukti *Meeting* Implementasi dan Pembahasan Hasil *Dashboard*

3.2. Kendala yang Ditemukan

Kegiatan magang tidak terlepas dari tantangan yang menjadi bagian penting dalam proses pembelajaran dan pengembangan diri di lingkungan kerja profesional. Selama menjalani magang, terdapat beberapa kendala yang perlu dihadapi dan diselesaikan secara bertahap untuk mendukung kelancaran proyek yang dijalankan, antara lain:

1. Keterbatasan pengetahuan terhadap mekanisme kerja proyek yang berbeda dengan pembelajaran di kampus, terutama dalam penggunaan *software* dan proses teknis yang baru. Kurangnya pengalaman praktis dalam merancang *pipeline*, mengelola *data warehouse*, dan *deployment dashboard* membuat proses adaptasi memerlukan waktu lebih lama.
2. Pemahaman struktur data menjadi tantangan karena setiap sistem memiliki *database* berbeda dan banyak tabel yang tidak terdokumentasi. Hal ini menghambat proses *join* antar tabel dan memperlambat pembuatan *query* ETL, sehingga berdampak pada keterlambatan visualisasi data.
3. Gangguan jaringan dan konektivitas server sering terjadi saat mengakses Apache Superset, dengan *error network timeout* atau *connection refused*. Gangguan ini menghambat proses perancangan dan penyimpanan *dashboard* karena harus menunggu server kembali normal.
4. Keterbatasan komunikasi dengan mentor akibat jadwal dinas luar kota yang padat menghambat progres pekerjaan karena respon sering tertunda. Selain itu, minimnya rekan kerja yang menguasai *tools* memperlambat beberapa tugas yang membutuhkan bantuan teknis dari mentor.

3.3. Solusi atas Kendala yang Ditemukan

Setiap tantangan yang muncul selama pelaksanaan magang menjadi peluang untuk mengembangkan kemampuan adaptasi dan *problem solving* dalam lingkungan kerja. Berbagai upaya dilakukan secara bertahap dan terarah untuk mengatasi kendala-kendala yang telah dijelaskan pada Subbab 3.3, di antaranya adalah sebagai berikut:

1. Pendalaman *tools* selama magang dilakukan secara mandiri melalui *platform* pembelajaran *online* seperti dokumentasi resmi *tools* tersebut, forum komunitas, dan tutorial video di YouTube. Fokus pembelajaran ini mencakup pemahaman alur kerja *end-to-end* dalam pembuatan *dashboard hospitality*, mulai dari *pipeline* ETL di Azure Data Factory, pembuatan *materialized views* dengan PostgreSQL di DBeaver, hingga visualisasi data di Apache Superset. Selain itu, mempelajari proyek-proyek terdahulu dari

mentor juga memberikan referensi praktis yang mempercepat adaptasi dan meningkatkan kemampuan teknis secara bertahap.

2. Kendala memahami struktur data di sistem PMS, POS, dan CRM diatasi dengan mengeksplorasi struktur tabel dan kolom melalui DBeaver, lalu mendokumentasikannya dalam catatan pribadi. Pemahaman relasi antar tabel divalidasi melalui diskusi dengan mentor untuk mempercepat pembuatan *query* dan meningkatkan akurasi pipeline serta *dashboard*.
3. Masalah koneksi ke server Apache Superset diatasi dengan melapor ke mentor dan menghubungi tim IT Ops untuk penanganan teknis. Sambil menunggu server normal, peserta magang mengerjakan tugas lain agar waktu tetap produktif dan pekerjaan tidak tertunda.
4. Kendala komunikasi dengan mentor diatasi dengan menjadwalkan diskusi online melalui Microsoft Teams dan rutin mengirimkan progres pekerjaan melalui *chat* agar tetap ter-*monitoring*. Jika ada pertanyaan teknis namun tidak *urgent*, peserta magang melakukan eksplorasi mandiri dan mencatat hal-hal penting untuk dibahas saat mentor kembali.