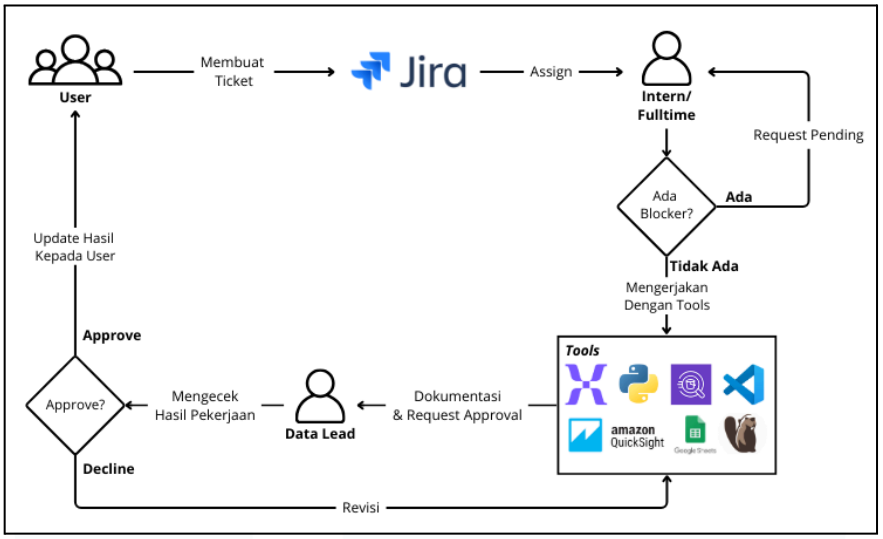
BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Tim Data di Ruparupa dipimpin oleh seorang *Data Lead* yang membawahi tiga sub-divisi, yaitu *Data Engineer, Data Scientist,* dan *Data Analyst*. Setiap sub-divisi memiliki peran dan tanggung jawab masing-masing dalam mendukung pengelolaan dan analisis data perusahaan. *Data Engineer* bertanggung jawab dalam mengelola infrastruktur data dan mengolah data mentah menjadi bentuk yang lebih terstruktur. *Data Scientist* berperan dalam analisis mendalam dengan menggunakan *machine learning* dan AI untuk menghasilkan insight bisnis yang dapat mendukung pengambilan keputusan. Sementara itu, *Data Analyst* bertugas dalam interpretasi data serta penyajian hasil analisis dalam bentuk laporan atau visualisasi data. Dalam struktur ini, posisi *Data Engineer Intern* disupervisi langsung oleh seorang *Data Enginee*r dan bertanggung jawab dalam membantu proses ekstraksi, transformasi, dan pemuatan data (ETL), membersihkan serta mengelola data agar siap digunakan untuk analisis lebih lanjut, serta mendukung pengembangan pipeline data yang efisien. Selain itu, *Data Engineer Intern* juga bertugas dalam pemantauan performa sistem data dan melakukan debugging jika terjadi kendala dalam proses pengolahan data.

Alur kerja di tim Data Ruparupa dijelaskan dalam Gambar 3.1, di mana proses dimulai ketika seorang *user* memiliki kebutuhan terkait data. *User* tersebut akan membuat tiket di JIRA, yaitu *software* manajemen proyek yang digunakan untuk mencatat, mengorganisir, dan melacak tugas-tugas yang ada. Setelah tiket dibuat, tiket tersebut akan muncul di JIRA dan kemudian akan di-*assign* kepada *intern* atau karyawan tetap sesuai dengan kebutuhan dan tingkat kesulitan tugas tersebut. Penugasan tiket ini biasanya dilakukan setiap hari pada pukul 09.30 WIB dalam *meeting huddle*, yaitu pertemuan rutin tim Data melalui Google Meet yang bertujuan untuk membahas progres pekerjaan sebelumnya, rencana kerja hari ini, serta mendiskusikan kendala yang dihadapi oleh masing-masing anggota tim.



Gambar 3.1 Alur Kerja Tim Data Ruparupa

Setelah tiket di-*assign*, *intern* atau karyawan tetap akan berkomunikasi terlebih dahulu dengan *user* untuk memastikan pemahaman yang jelas terkait permintaan dalam tiket. Jika terdapat kendala atau *blocker* yang menghambat penyelesaian tugas, maka tiket akan masuk ke status "*Pending for Approval*" agar tidak mempengaruhi SLA (*Service Level Agreement*) pengerjaan tiket. SLA sendiri merupakan batas waktu penyelesaian tugas yang ditetapkan selama 48 jam sejak tiket di-*assign* untuk memastikan efisiensi kerja tim. Hal ini penting untuk menjaga *workflow* tetap lancar dan memastikan setiap tugas dapat diselesaikan sesuai tenggat waktu yang telah ditetapkan.

Jika tidak terdapat *blocker*, *assignee* dapat langsung mengerjakan tiket menggunakan berbagai *tools* sesuai dengan kebutuhan, seperti Mixpanel, Visual Studio Code (VS Code), DBeaver, Google Sheets, Amazon QuickSight, AWS Athena, dan lainnya. Penggunaan *tools* ini bertujuan untuk memastikan data dapat diolah dengan optimal sesuai dengan kebutuhan user dan standar perusahaan. Setelah tiket selesai dikerjakan, *assignee* wajib melakukan dokumentasi di Airtable serta memperbarui status tiket di JIRA untuk memastikan bahwa seluruh tim Data dapat melihat perkembangan dan memastikan transparansi dalam pekerjaan. Dokumentasi ini sangat penting agar pekerjaan dapat ditinjau kembali oleh tim lain jika diperlukan dan untuk mendukung efektivitas dalam koordinasi pekerjaan.

Setelah proses dokumentasi selesai, tiket akan masuk ke status "*Waiting for Approval*" dan diperiksa oleh *Data Lead* untuk memastikan bahwa pekerjaan sudah sesuai dengan kebutuhan. Jika ditemukan kesalahan atau kekurangan dalam pengerjaan tiket, maka *Data Lead* akan mengembalikannya untuk direvisi oleh *assignee* agar hasil akhirnya sesuai dengan standar yang telah ditentukan. Namun, jika pekerjaan telah sesuai dengan standar dan tidak ada kesalahan, maka tiket akan diperbarui kepada *user* yang bersangkutan sebagai tanda bahwa tugas telah selesai dikerjakan. Proses ini memastikan bahwa setiap pekerjaan yang dilakukan oleh tim Data Ruparupa memiliki kualitas yang baik dan memenuhi kebutuhan *user* secara optimal.

Proses ini akan terus diulang setiap kali *user* memiliki kebutuhan baru terkait data, sehingga koordinasi dalam tim dapat berjalan dengan baik. Sistem kerja ini membantu memastikan bahwa setiap tiket dikerjakan dengan efisien, transparan, dan sesuai dengan SLA yang telah ditetapkan. Dengan adanya alur kerja yang jelas dan terdokumentasi dengan baik, tim Data Ruparupa dapat terus meningkatkan performa mereka dalam memberikan layanan analisis dan pengolahan data. Selain itu, mekanisme ini juga membantu menciptakan lingkungan kerja yang lebih produktif dan kolaboratif dalam tim.

3.2 Tugas dan Uraian Kerja Magang

Tugas utama yang dilakukan oleh *Data Engineer Intern* di PT. Omni Digitama Internusa (Ruparupa) mencakup kegiatan *maintenance* terhadap sistem data yang sudah ada, termasuk merawat, menyesuaikan, serta melakukan *debugging* dan perbaikan *bug* pada *dataset*, sistem otomatisasi, dan berbagai proses data lainnya. *Maintenance* ini bertujuan untuk memastikan seluruh infrastruktur data berjalan dengan optimal serta mendukung kebutuhan analisis dan operasional perusahaan. Selain itu, pengembangan dan pengelolaan pipeline data dilakukan oleh intern, begitu pula dengan optimasi query untuk meningkatkan efisiensi pengolahan data. Proses ekstraksi, transformasi, dan pemuatan data (ETL) juga dibantu oleh *intern*. *Intern* juga dilibatkan dalam monitoring sistem untuk mengidentifikasi potensi

masalah, serta dilakukan optimasi pada pipeline data oleh *intern* agar kinerjanya tetap efisien dan mampu menangani volume data yang besar.

Tidak hanya terbatas pada pemeliharaan, *Data Engineer Intern* juga diberikan beberapa proyek, baik dalam skala kecil maupun besar. Proyek-proyek ini melibatkan pengolahan data dalam jumlah besar, pengembangan sistem otomatisasi baru, atau peningkatan kinerja sistem yang sudah ada guna meningkatkan efisiensi kerja tim Data. Selama pengerjaan baik proyek atau *maintenance*, *intern* selalu mendapatkan supervisi serta bimbingan langsung dari *Data Engineer full-time* untuk memastikan bahwa tugas berjalan dengan baik serta menjadi pengalaman pembelajaran yang berharga dalam memahami cara kerja sistem data di dunia industri.

Selain tugas teknis, *intern* juga memiliki tanggung jawab dalam dokumentasi pekerjaan yang telah dilakukan. Dokumentasi ini dicatat melalui tiket di JIRA agar seluruh perubahan dan perbaikan yang dilakukan dapat ditelusuri dengan jelas, serta di Airtable untuk mempermudah akses informasi oleh seluruh tim Data. Dokumentasi yang baik membantu menjaga transparansi kerja, memudahkan evaluasi terhadap proyek yang telah dikerjakan, serta memastikan bahwa setiap pembaruan pada sistem dapat dipahami oleh seluruh tim untuk mendukung pengembangan lebih lanjut. Dengan adanya kombinasi antara tugas *maintenance*, proyek strategis, dan dokumentasi, program magang ini memberikan pengalaman yang komprehensif dalam peran seorang *Data Engineer* di lingkungan kerja nyata. Rincian lebih lanjut mengenai uraian pekerjaan yang dilakukan selama periode magang dapat dilihat pada Tabel 3.1.

Tabel 3.1 Uraian Tugas dan Waktu Pelaksanaan Magang

No	Deskripsi Kegiatan	Minggu ke-	Tanggal Mulai	Tanggal Selesai
1	Pengenalan lingkungan serta *transfer knowledge* mengenai sistem kerja di Kawan Lama dan Ruparupa.	1	3 Februari 2025	7 Februari 2025
Database & Data Warehouse Management				
2	Mengatur akses data di MySQL Data Warehouse, Rapod, dan	2 - 20	10 Februari 2025	27 Juni 2025

No	Deskripsi Kegiatan	Minggu ke-	Tanggal Mulai	Tanggal Selesai
	Quicksight (akun, izin akses).			
3	Setup *Data Warehouse* dan *Data Lake*.	1 & 5	4 Februari 2025	3 Maret 2025
4	Memvalidasi dan memperbarui *dataset* yang gagal *refresh*.	2 - 20	10 Februari 2025	27 Juni 2025
Data Cleaning, Integration & Analysis				
5	Menggabungkan data dari berbagai sumber menjadi sebuah *dataset*.	2 - 20	5 Februari 2025	27 Juni 2025
6	*Maintenance dataset* dengan menambah kolom, mengubah logika pemrosesan, dan memastikan akurasi data.	1 - 20	4 Februari 2025	27 Juni 2025
Automation & Workflow Optimization				
7	Mengelola skrip Python untuk otomasi proses data.	3 - 20	17 Februari 2025	27 Juni 2025
8	Mengerjakan proyek *Reconciliation Data Validator for Shipper Comparison*.	3 - 7	21 Februari 2025	18 Maret 2025
9	Memperbaharui *script Tokopedia Data Extraction & Automation*.	6	10 Februari 2025	14 Februari 2025
10	Mengerjakan proyek *Appsflyer Onelink Master Data Automation*.	8 - 9	27 Maret 2025	11 April 2025
11	Memantau otomasi di Make.com.	2 - 20	10 Februari 2025	27 Juni 2025
12	Mengelola pembaruan token API sesuai jadwal.	2, 4, 6, 8, 10, 12, 14, 16, 18, 20	10 Februari 2025	24 Juni 2025
Documentation & Reporting				
13	Membuat dan memperbarui dokumentasi terkait definisi data, *workflow*, serta *dataset queries*.	1 - 20	3 Februari 2025	27 Juni 2025
14	Membuat ERD untuk membantu pemahaman mengenai *database* perusahaan.	1 - 2	3 Februari 2025	14 Februari 2025

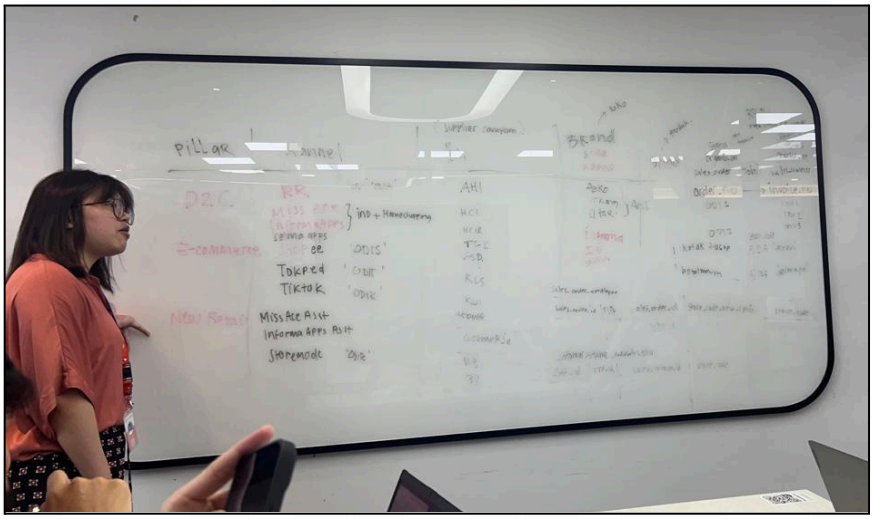
(Sumber olahan peneliti, 2025)

3.2.1 Pengenalan Lingkungan Serta *Transfer Knowledge* Mengenai Sistem Kerja di Kawan Lama dan Ruparupa (Minggu ke 1)

Pada minggu pertama magang di Ruparupa, para mahasiswa *intern* mendapatkan pengenalan mengenai lingkungan kerja serta nilai-nilai perusahaan induk yaitu Kawan Lama Group. Sesi orientasi ini berlangsung selama tiga jam dalam bentuk *lecture* seperti pada Gambar 3.2 yang membahas berbagai aspek penting, seperti etos kerja, budaya perusahaan, struktur organisasi, serta fasilitas yang tersedia bagi karyawan. Selain itu, sesi ini juga memperkenalkan berbagai pilar bisnis atau *Business Unit* (BU) yang beroperasi di bawah Kawan Lama Group seperti yang ada pada Gambar 3.3 untuk memberikan pemahaman lebih mendalam mengenai cakupan bisnis perusahaan. Setelah sesi orientasi, *intern* diperkenalkan lebih dalam dengan lingkungan kerja Ruparupa di lantai 2, termasuk bertemu dengan tim data, memahami struktur organisasi, serta mengeksplorasi berbagai *channel* perusahaan dan sistem yang digunakan dalam pengelolaan serta analisis data, sebagaimana tercantum dalam Gambar 3.4.

Gambar 3.2 Sesi Orientasi dan *Lecture* Hari Pertama Magang di Kawan Lama Group

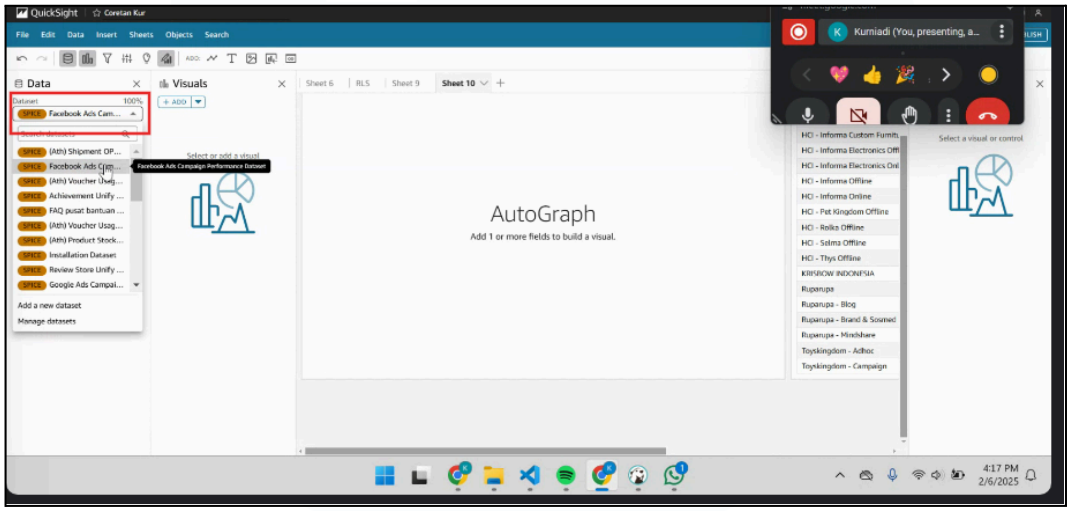
Gambar 3.3 Pilar Bisnis Kawan Lama Group



Gambar 3.4 Sesi Pengenalan BU dan *Channel* Ruparupa

Pada hari kedua hingga kelima, *transfer knowledge* dilakukan oleh *Data Engineer Intern* sebelumnya di Ruparupa melalui materi yang disampaikan langsung atau melalui Google Meet, seperti tercantum dalam Gambar 3.5. Materi yang diajarkan mencakup berbagai aspek teknis dalam *data engineering*, seperti penggunaan SQL untuk manipulasi data, pembangunan *pipeline data* dengan teknologi seperti Apache Airflow, serta konsep *data warehousing* untuk mengelola data dalam skala besar. Selain itu, sesi ini juga membahas optimalisasi performa *query* agar lebih efisien, implementasi *workflow* otomatisasi, serta penerapan *best practices* dalam manajemen data guna memastikan kualitas dan konsistensi informasi yang diolah. *Intern* juga diberikan kesempatan untuk

menerapkan langsung materi yang dipelajari melalui studi kasus yang relevan dengan kebutuhan bisnis, mulai dari integrasi data dari berbagai sumber, proses pembersihan data untuk meningkatkan akurasi, hingga visualisasi data guna mendukung pengambilan keputusan. Setiap proses dan temuan yang diperoleh diwajibkan untuk dicatat sebagai dokumentasi serta sebagai bekal yang dapat digunakan untuk referensi dan pengembangan lebih lanjut di masa depan.



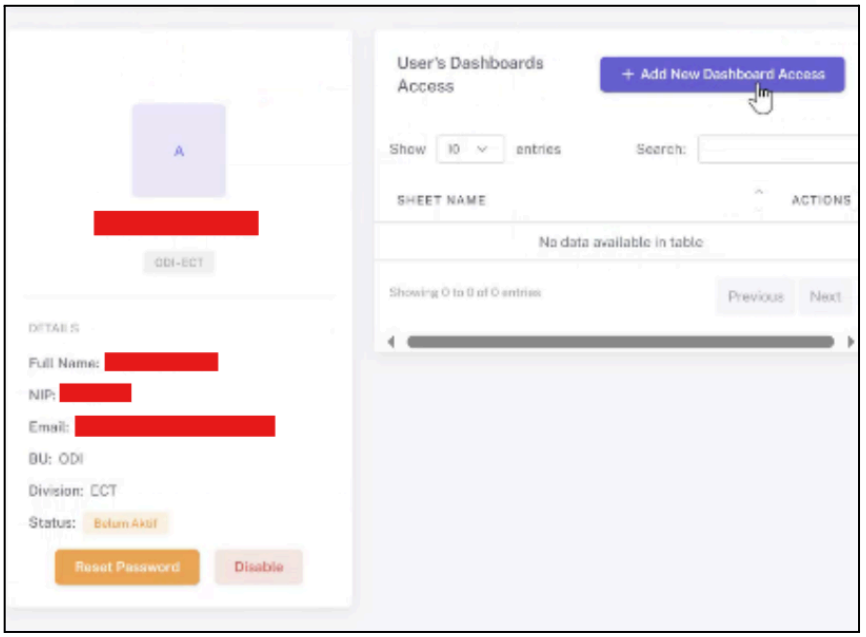
Gambar 3.5 Proses *Transfer Knowledge* di Ruparupa Melalui Google Meet

3.2.2 Mengatur Akses Data di Mysql Data Warehouse, Rapod, dan Quicksight (Akun, Izin Akses) (Minggu ke 2 - 20)

Sebagai seorang *Data Engineer*, menjaga keamanan data merupakan aspek yang sangat penting dalam mengelola infrastruktur data perusahaan. Oleh karena itu, *intern* bertanggung jawab dalam mengatur dan mengelola akses pengguna ke beberapa *platform* utama seperti MySQL Data Warehouse, Rapod, dan Amazon QuickSight. Pengelolaan akses ini bertujuan untuk memastikan bahwa hanya pengguna yang berwenang yang dapat mengakses data tertentu, sehingga integritas dan keamanan informasi tetap terjaga.

Rapod merupakan *website* yang dikembangkan oleh tim *Data Engineer* untuk mempermudah akses ke *dashboard* Amazon QuickSight tanpa memerlukan akun QuickSight secara langsung. *Platform* ini dirancang agar pengguna dari tim non-teknis dapat mengakses dan menganalisis data dengan lebih mudah tanpa perlu memahami teknis penggunaan QuickSight. Dalam pengelolaan akses pengguna di Rapod, *intern* bertugas untuk membuat akun baru untuk *user*,

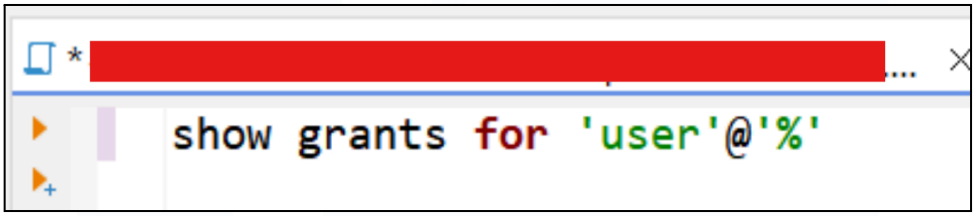
menambahkan dan menghapus akses ke *dashboard*, serta menghapus *user* yang tidak lagi memiliki kepentingan dalam penggunaan *platform* ini. Proses ini dapat dilihat pada Gambar 3.6, yang menunjukkan bagaimana manajemen akses pengguna di dalam Rapod. Jika seorang pengguna telah *resign* atau tidak lagi memerlukan akses, maka akun pengguna tersebut perlu dihapus untuk menjaga keamanan sistem.



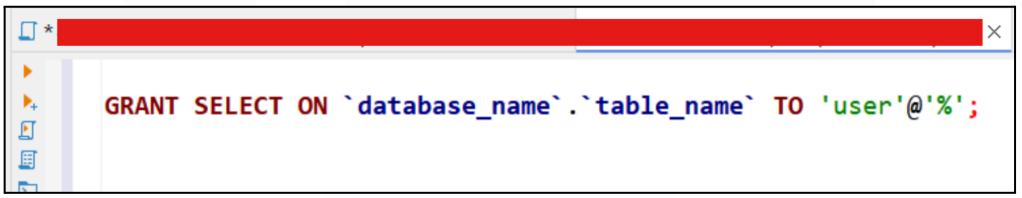
Gambar 3.6 Manajemen Akses Pengguna di Rapod

Dalam mengatur akses pengguna ke *database* perusahaan yang berada di MySQL Data Warehouse, DBeaver digunakan sebagai *software* utama. Hak akses *Data Engineer Intern* dalam DBeaver terbagi menjadi *data reader*, yang memungkinkan pengguna hanya dapat membaca data, dan *data writer*, yang memungkinkan pengguna melakukan perubahan pada data. *Intern* bertanggung jawab untuk melihat hak akses pengguna pada *data reader* menggunakan *query* tertentu untuk memastikan akses mereka sesuai kebutuhan sebelum memberikan izin tambahan. *Query* ini ditunjukkan dalam Gambar 3.7, yang menggambarkan bagaimana cara mengecek akses *database* dan tabel yang dimiliki oleh *user*. Jika ada pengguna yang memerlukan akses ke *database* tertentu, *intern* perlu menjalankan *query* seperti yang terlihat pada Gambar 3.8, yang berfungsi untuk memberikan akses kepada *user* agar dapat mengakses data sesuai peran dan

tugasnya. Selain memberikan akses, *intern* juga bertanggung jawab untuk menghapus akses pengguna yang sudah tidak memerlukannya, baik karena perubahan tugas, pergantian tim, atau pengunduran diri (*resign*). Proses penghapusan akses ini dapat dilihat pada Gambar 3.9, yang menunjukkan bagaimana *intern* menghapus akses pengguna yang tidak lagi relevan. Dengan adanya kontrol akses ini, perusahaan dapat memastikan bahwa hanya pihak yang berwenang yang dapat mengakses dan mengelola data dalam sistem.



Gambar 3.7 *Query* Mengecek Akses *User*



Gambar 3.8 *Query* untuk Memberikan Akses Kepada *User*

Gambar 3.9 *Query* Menghapus Akses *User*

Selain mengelola akses di MySQL Data Warehouse, *intern* juga bertanggung jawab dalam mengatur akses pengguna di Amazon QuickSight, yang berfungsi sebagai *Data Lake* dan tempat penyimpanan *dataset* yang digunakan oleh *Data Analyst* maupun *Data Engineer*. Platform ini memungkinkan pengguna untuk membuat dan menyimpan *dataset* baru ketika ada perubahan atau kebutuhan tambahan dalam analisis data. *Intern* memiliki tugas untuk menentukan siapa saja yang dapat mengakses suatu *dataset*, sehingga data tetap aman dan hanya dapat digunakan oleh pihak yang berwenang. Pada Gambar 3.10, dapat dilihat bagaimana mengelola akses pengguna ke suatu *dataset* di Amazon QuickSight.

Selain itu, *intern* juga dapat membantu dalam pembuatan *dataset* baru yang akan digunakan untuk analisis lebih lanjut sesuai dengan kebutuhan tim. Dengan pengelolaan akses yang terstruktur, penggunaan Amazon QuickSight dapat lebih optimal dalam mendukung pengambilan keputusan berbasis data.

Gambar 3.10 Manajemen Akses *Dataset* di Amazon QuickSight

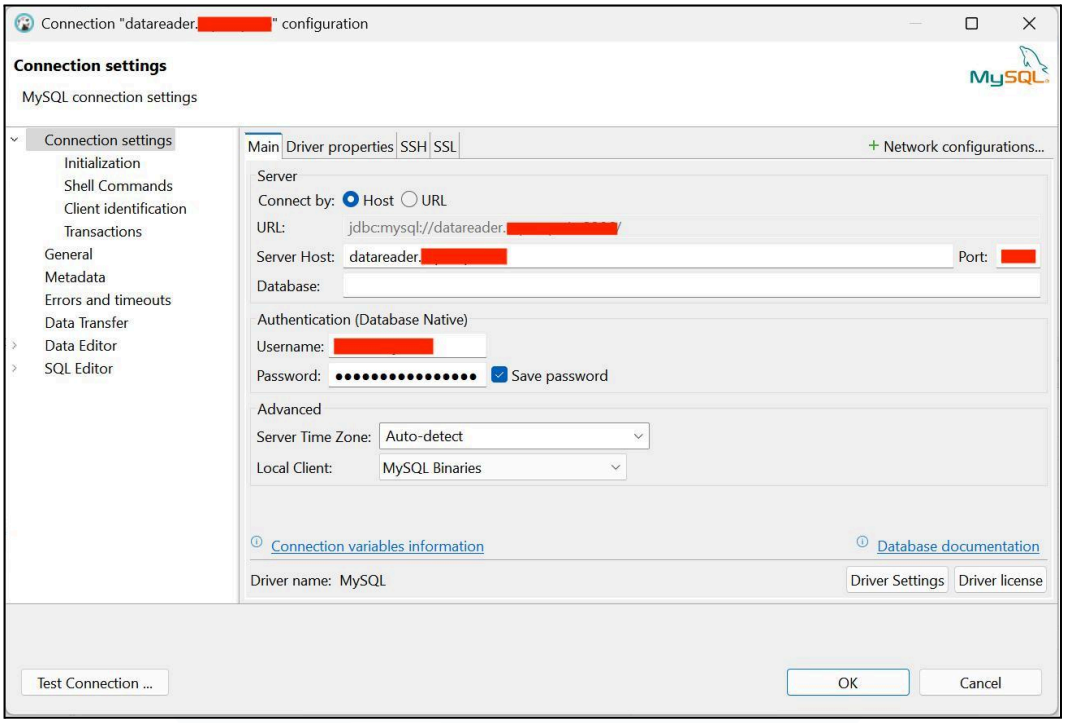
Tugas *maintenance* ini berlangsung sepanjang periode magang dan menjadi bagian penting dalam memastikan sistem tetap berjalan dengan aman dan terkendali. Dengan adanya sistem pengelolaan akses yang baik pada Rapod, MySQL Data Warehouse (DBeaver), dan Amazon QuickSight, perusahaan dapat memastikan bahwa keamanan data tetap terjaga. *Intern* memiliki peran penting dalam memastikan bahwa setiap pengguna hanya memiliki akses sesuai dengan peran dan kebutuhannya. Hal ini tidak hanya meningkatkan efisiensi kerja tetapi juga mencegah potensi kebocoran atau penyalahgunaan data yang dapat merugikan perusahaan.

3.2.3 Setup *Data Warehouse* Serta *Data Lake* (Minggu ke 1 dan 5)

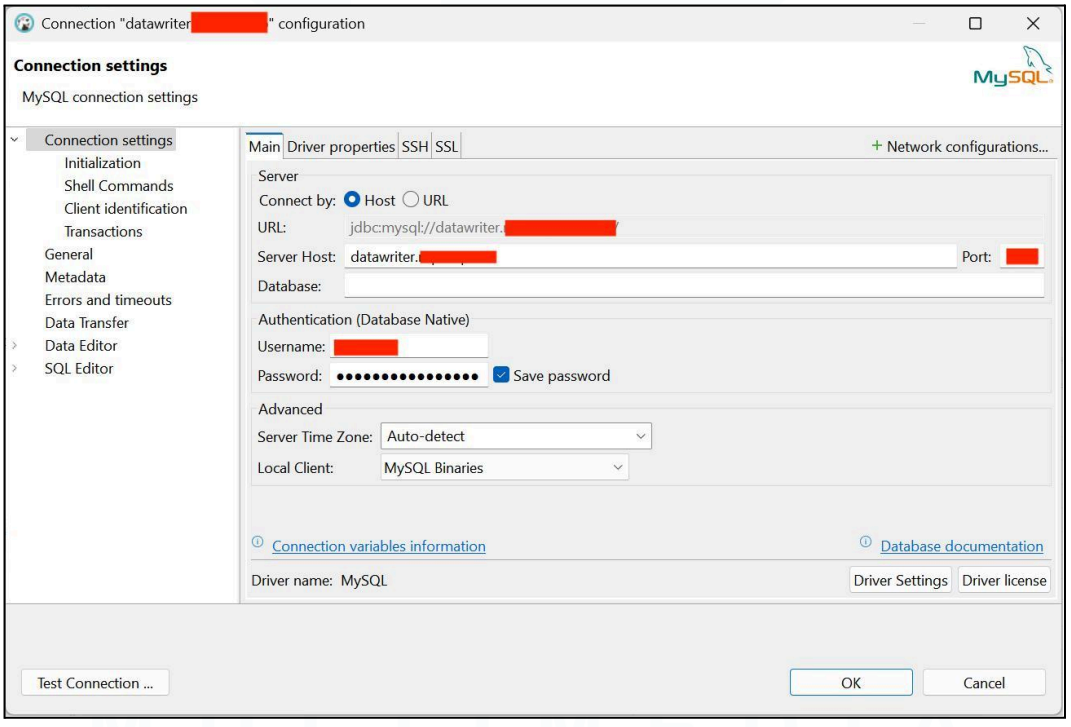
Data Warehouse dan *Data Lake* merupakan dua komponen penting dalam pengelolaan data di Ruparupa. *Data Warehouse* adalah sistem penyimpanan data yang terstruktur dan dioptimalkan untuk analisis bisnis, di mana data telah melalui proses transformasi dan disusun dalam format yang terorganisir. Sementara itu, *Data Lake* adalah penyimpanan data dalam bentuk mentah (*raw data*) yang dapat menyimpan berbagai jenis data, baik yang terstruktur, semi-terstruktur, maupun tidak terstruktur, sehingga memungkinkan fleksibilitas dalam analisis data untuk

kepentingan kedepannya. Kedua sistem ini berperan krusial dalam mendukung pengambilan keputusan berbasis data di perusahaan.

Agar dapat mengakses *Data Warehouse* di Ruparupa, diperlukan proses *setup* di DBeaver, sebuah *software* manajemen database yang memungkinkan koneksi ke berbagai jenis database. Setup ini mencakup beberapa konfigurasi utama, yaitu pengaturan *server host*, *database native authentication*, dan konfigurasi *SSH tunnel*. Proses ini dilakukan baik untuk *data reader* (seperti yang ditunjukkan pada Gambar 3.11) maupun untuk *data writer* (seperti yang terlihat pada Gambar 3.12). Setiap konfigurasi harus dilakukan dengan benar untuk memastikan bahwa pengguna dapat terhubung ke *database* dengan aman dan tanpa kendala teknis. Selain *Data Warehouse*, akses ke *Data Lake* di Ruparupa dilakukan melalui Amazon QuickSight dan AWS Athena, yang memungkinkan pengguna untuk mengelola dan menganalisis data dalam skala besar. Setup *Data Lake* mencakup pembuatan akun menggunakan *email* Ruparupa di AWS Athena dan Amazon QuickSight, sehingga pengguna dapat mengakses data yang diperlukan untuk analisis.



Gambar 3.11 *Setup Data Reader*



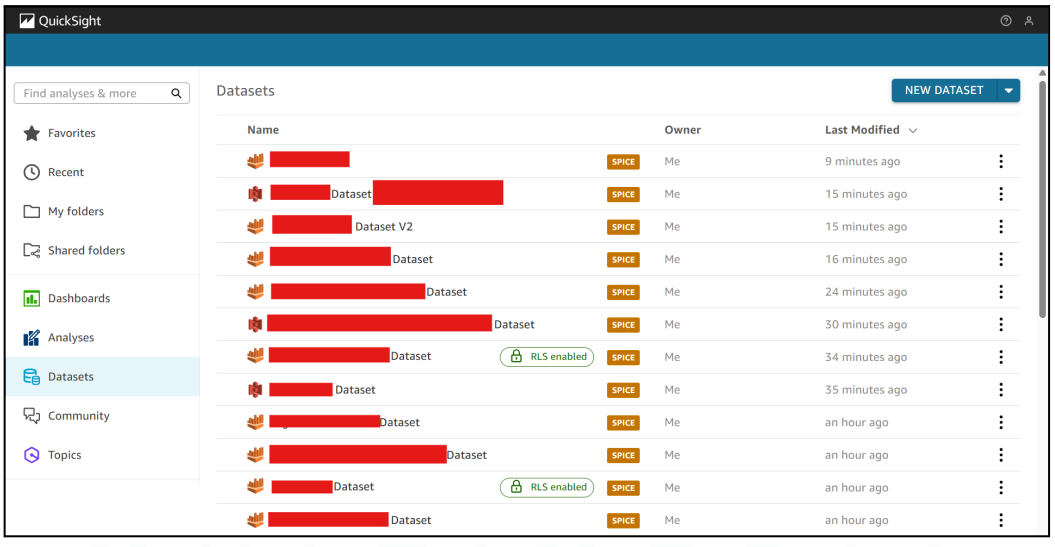
Gambar 3.12 *Setup Data Writer*

Data Engineer Intern juga bertugas untuk mengatur hak akses bagi *Data Scientist Intern* dan *Data Analyst Intern*. Hak akses ini diberikan dengan

melakukan *setup data reader*, yang memungkinkan pengguna membaca data tanpa memiliki izin untuk mengubahnya. *Setup* ini penting agar setiap tim dapat mengakses informasi yang dibutuhkan untuk analisis tanpa mengganggu integritas data dalam sistem. Proses *setup* koneksi dan hak akses ini dilakukan dalam dua tahap, yaitu pada awal magang di bulan Februari 2025 menggunakan laptop pribadi, serta dilakukan kembali di awal bulan Maret 2025 setelah *intern* mendapatkan laptop kantor. *Setup* ulang ini diperlukan untuk memastikan semua konfigurasi tetap berjalan dengan baik pada perangkat baru yang telah disediakan oleh perusahaan.

3.2.4 Memvalidasi dan Memperbarui Dataset yang Gagal *Refresh* (Minggu ke 2 - 20)

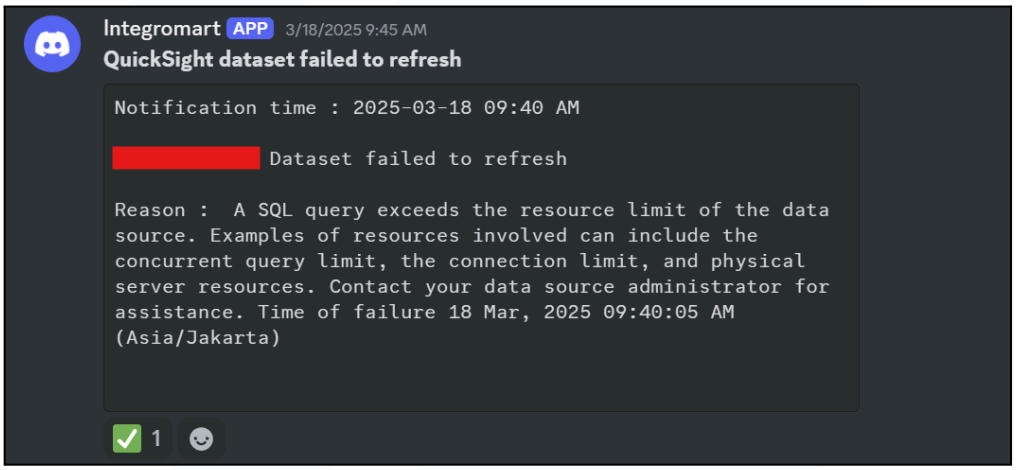
Dataset yang telah dibuat oleh tim *Data Engineer* disimpan di Amazon QuickSight seperti yang ditunjukkan pada Gambar 3.13. Untuk memastikan bahwa dataset selalu menggunakan data terbaru, setiap *dataset* umumnya memiliki jadwal *refresh* secara berkala. Proses *refresh* ini bertujuan untuk memperbarui *dataset* dengan data terbaru dari *Data Lake*, sehingga analisis yang dilakukan oleh pengguna tetap akurat dan relevan.



Gambar 3.13 *Dataset Management* di Amazon QuickSight

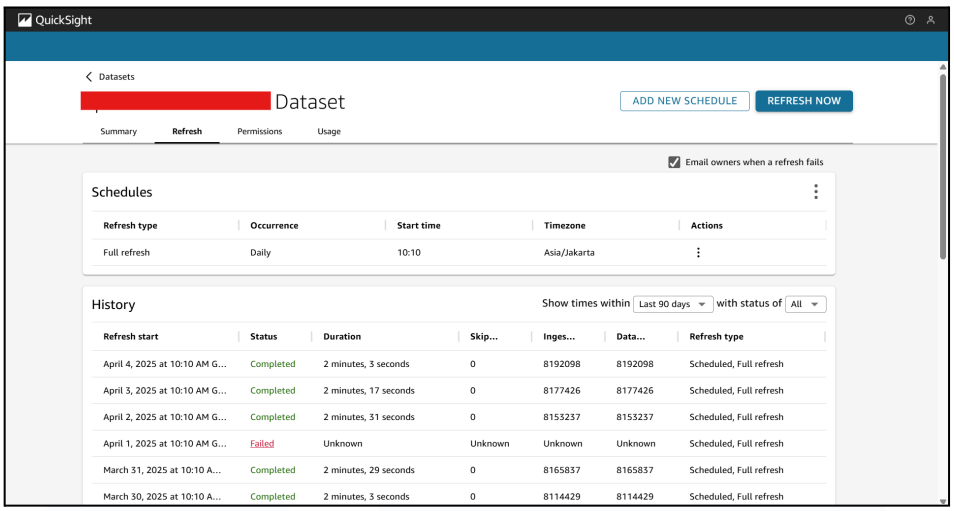
Namun, dalam praktiknya, sering terjadi kendala di mana dataset gagal melakukan *refresh*. Kegagalan ini dapat disebabkan oleh berbagai faktor, seperti SQL *query* yang melebihi batas sumber daya (*A SQL query exceeds the resource*

limit of the data source), kapasitas penyimpanan SPICE yang tidak mencukupi (*This data exceeds your current SPICE capacity*), atau terlalu banyak baris data yang tidak valid (*There are too many invalid rows*). Notifikasi mengenai kegagalan *refresh* ini biasanya dikirimkan melalui pesan di Discord, seperti yang dapat dilihat pada Gambar 3.14.



Gambar 3.14 Pesan *Error Dataset* QuickSight di Discord

Dalam situasi seperti ini, *Data Engineer Intern* bertugas untuk melakukan *refresh* ulang *dataset* yang gagal atau melakukan *debugging* jika diperlukan. Jika penyebab kegagalan hanyalah batas sumber daya atau kapasitas SPICE, maka dapat langsung dilakukan *refresh* secara manual agar *dataset* kembali normal. Namun, jika terdapat indikasi *invalid rows*, perlu dilakukan *debugging* dengan mengecek kesalahan pada *dataset*, memperbaiki data yang bermasalah, dan memastikan *dataset* dapat di-*refresh* tanpa *error*. Contoh kasus berhasilnya proses refresh ulang dapat dilihat pada Gambar 3.15, di mana status dataset kembali menjadi aktif setelah dilakukan penanganan.

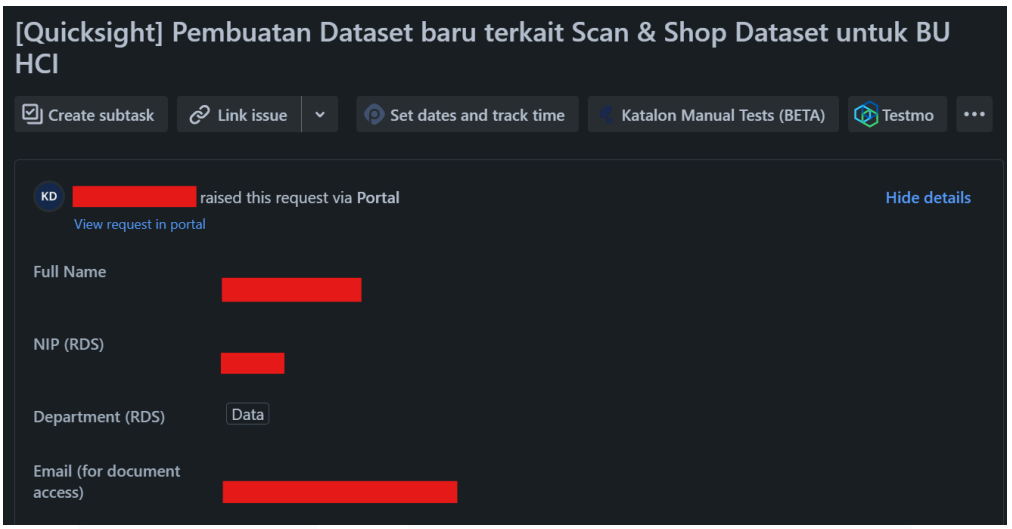


Gambar 3.15 *Dataset* yang Berhasil Di-*refresh* Kembali

Setelah dataset berhasil di-*refresh* atau diperbaiki, reaksi *checklist* (✅) diberikan pada pesan Discord untuk menandakan bahwa *error* telah ditangani dengan baik seperti pada Gambar 3.14. Proses monitoring dan *maintenance* ini dilakukan secara terus-menerus dengan memantau notifikasi di Discord, guna memastikan bahwa semua *dataset* di Amazon QuickSight tetap berjalan dengan baik dan tidak mengalami kendala dalam proses *refresh*.

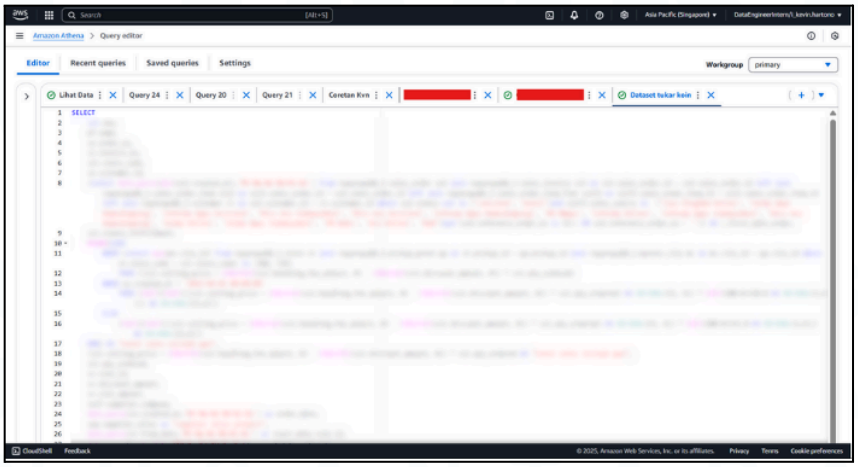
3.2.5 Menggabungkan Data Dari Berbagai Sumber Menjadi Sebuah *Dataset* (Minggu ke 2 - 20)

Dalam rangka mendukung proses analisis data oleh tim *Data Analyst*, *Data Scientist*, maupun *stakeholder* non-teknis, D*ata Engineer Intern* memiliki tanggung jawab dalam pembuatan *dataset* yang menggabungkan data dari berbagai sumber. Proses ini dilakukan karena kebutuhan analisis seringkali melibatkan informasi dari beberapa tabel atau bahkan lintas *database*. Dengan menggabungkan data yang tersebar ke dalam satu kesatuan *dataset*, proses analisis menjadi lebih efisien dan relevan sesuai konteks yang dibutuhkan oleh *user*. Permintaan untuk pembuatan *dataset* ini biasanya diajukan melalui sistem *ticketing* JIRA seperti pada Gambar 3.16, yang kemudian akan ditindaklanjuti oleh *Data Engineer* sesuai dengan prioritas dan urgensinya.



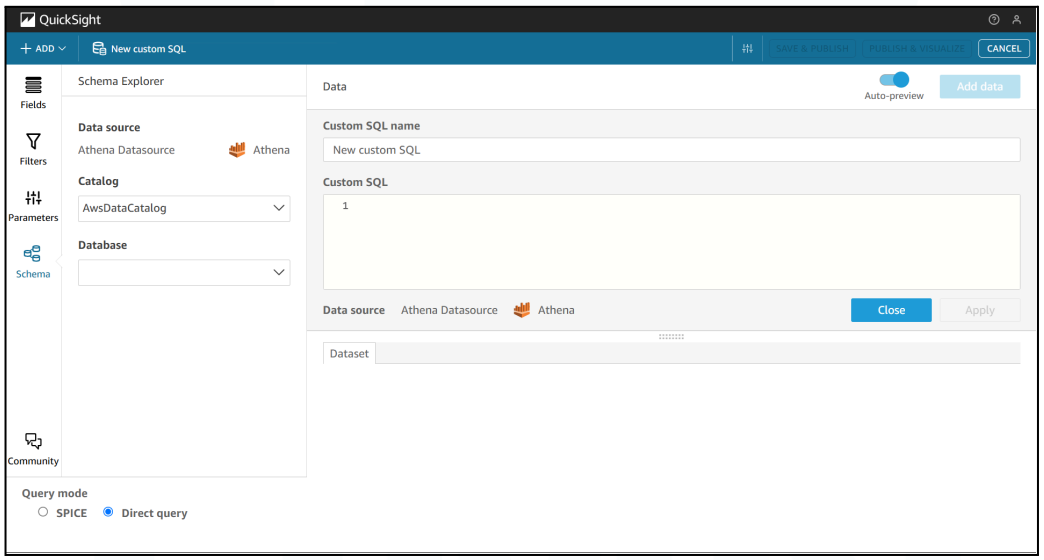
Gambar 3.16 Contoh Tiket Permintaan Pembuatan *Dataset* di JIRA

Setelah menerima tiket permintaan dari *user*, langkah awal yang dilakukan oleh *intern* adalah membuat SQL *query* di AWS Athena untuk mengolah data dari *Data Lake*. Athena dipilih karena kemampuannya dalam melakukan *query* langsung terhadap data mentah di S3 tanpa perlu proses ekstraksi manual. *Query* yang dapat menggabungkan tabel disusun dari berbagai sumber sesuai kebutuhan yang dijelaskan *user*, termasuk pengolahan seperti *join* antar tabel, *filtering*, dan *formatting*. Hasil dari *query* ini kemudian akan ditinjau terlebih dahulu untuk memastikan bahwa struktur dan isi data telah sesuai dengan ekspektasi *user* sebelum dilanjutkan ke tahap visualisasi. Contoh proses ini ditunjukkan dalam Gambar 3.17, yang menampilkan tampilan *query builder* pada AWS Athena.



Gambar 3.17 Tampilan *Query Builder* di AWS Athena

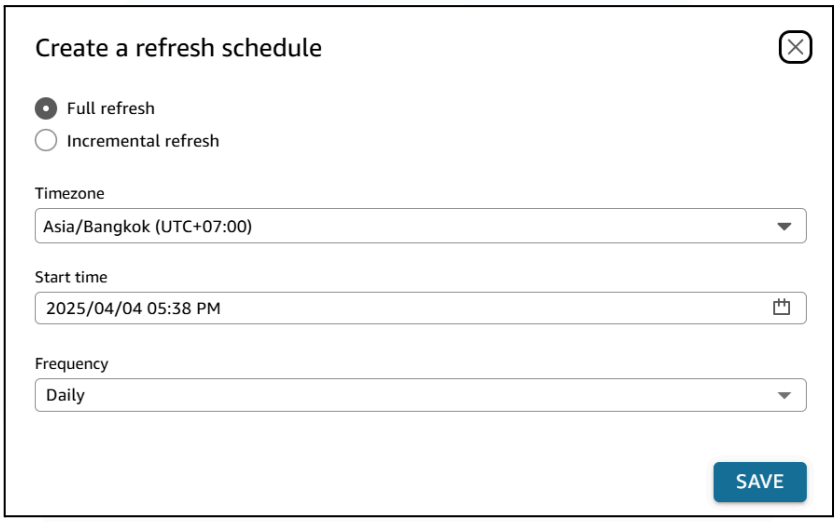
Jika data hasil *query* telah sesuai, proses selanjutnya adalah membuat *dataset* di Amazon QuickSight sebagai *platform* untuk menampung dan mengelola hasil pengolahan data tersebut. Pembuatan dataset dimulai dengan memilih opsi "*New Dataset*" di halaman utama QuickSight, lalu memilih sumber data berupa Athena Datasource. Setelah itu, dilakukan *preview* data dengan mengganti nama *dataset*, serta menyalin SQL *query* dari AWS Athena ke kolom *editor* yang tersedia di QuickSight seperti yang ada pada Gambar 3.18. Mode *query* kemudian diubah menjadi SPICE (*Super-fast, Parallel, In-memory Calculation Engine*) untuk mempercepat pemrosesan dan mendukung performa analisis yang lebih baik. Setelah semua langkah tersebut dilakukan, *dataset* akan disimpan dan di-*publish* dengan memilih tombol "*Save and Publish*".



Gambar 3.18 Tampilan Pembuatan *Dataset* Baru di Amazon QuickSight

Setelah *dataset* berhasil dibuat, tahap berikutnya adalah menambahkan jadwal *refresh* (*schedule refresh*) seperti yang ditunjukkan pada Gambar 3.19 untuk memastikan data di dalam *dataset* selalu terbarui. Jadwal *refresh* sangat penting terutama untuk *dataset* yang digunakan secara rutin dalam *dashboard* operasional atau pelaporan mingguan. Proses pengaturan ini umumnya melibatkan diskusi terlebih dahulu dengan *supervisor* untuk menentukan jenis *refresh* yang akan digunakan, yaitu *full refresh* atau *incremental refresh*, tergantung dari ukuran *dataset* dan kebutuhan frekuensi pembaruan. Jika *dataset* berukuran besar dan waktu pemrosesan menjadi pertimbangan, maka *incremental refresh* sering kali

dipilih sebagai solusi yang efisien. Setelah mendapatkan persetujuan *supervisor*, penjadwalan juga akan dikonfirmasi ke *user* agar sesuai dengan kebutuhan penggunaan data mereka.



Gambar 3.19 Penjadwalan *Refresh Dataset* di Amazon QuickSight

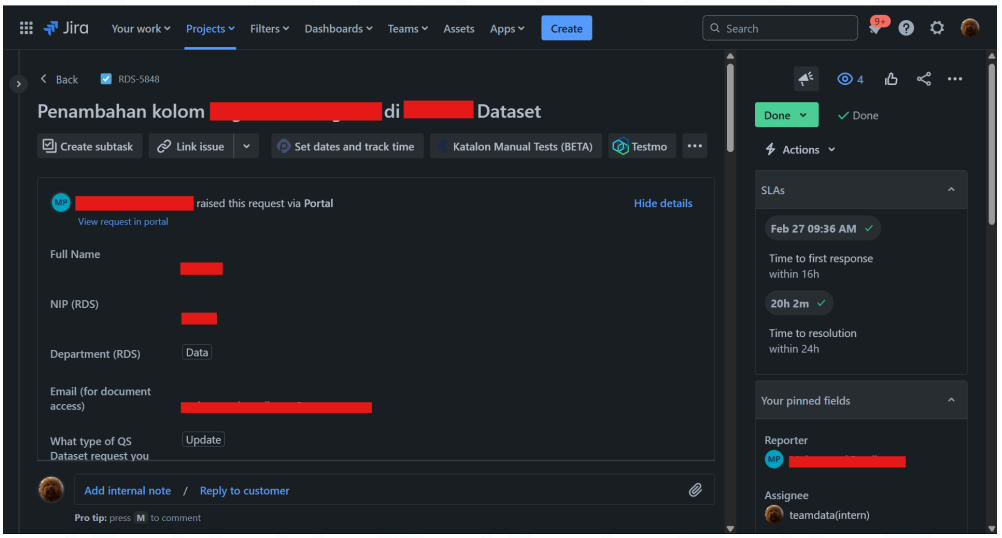
Setelah semua pengaturan selesai, konfigurasi disimpan dan dilakukan uji coba *refresh* untuk memastikan bahwa jadwal dan koneksi *dataset* berjalan dengan baik. Jika semua proses berjalan lancar, maka *dataset* siap digunakan untuk pembuatan visualisasi dashboard di Amazon QuickSight, atau digunakan sebagai sumber analisis lanjutan. Proses ini menggambarkan bagaimana *Data Engineer Intern* berperan aktif dalam proses ETL dan integrasi data, sekaligus menjembatani kebutuhan analisis antara tim teknis dan non-teknis dengan menyediakan *dataset* yang siap pakai dan mudah divisualisasikan.

3.2.6 *Maintenance Dataset* Dengan Menambah Kolom, Mengubah Logika Pemrosesan, dan Memastikan Akurasi Data (Minggu ke 1 - 20)

Selain membuat dan *refresh dataset*, *Data Engineer Intern* di Ruparupa memiliki tanggung jawab untuk melakukan pemeliharaan terhadap *dataset* yang telah digunakan oleh berbagai tim. Kebutuhan *user* terhadap data sering mengalami perubahan, sehingga *dataset* perlu disesuaikan secara berkala. Pemeliharaan ini dilakukan untuk menjaga kesesuaian isi *dataset* dengan kebutuhan yang ada di lapangan. Salah satu bentuk pemeliharaan yang dilakukan adalah menambah kolom baru atau mengubah logika *query* berdasarkan

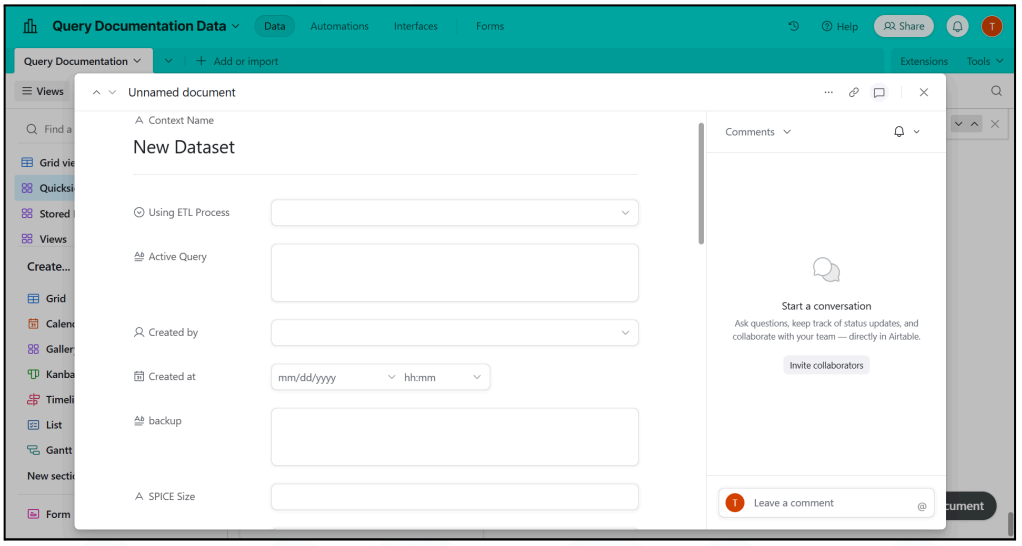
permintaan *user*. Tugas menambah kolom ini merupakan tugas paling awal dari proses *transfer knowledge* yang dijalani oleh *intern* di awal masa magang. Melalui proses ini, struktur *dataset* dan proses kerja yang digunakan oleh tim data dapat dipahami secara lebih mendalam.

Permintaan untuk menambah kolom biasanya diajukan melalui tiket di JIRA. Contoh dari permintaan ini dapat dilihat pada Gambar 3.20, yang menunjukkan bahwa *user* membutuhkan tambahan kolom tertentu untuk mendukung proses analisis. Setelah permintaan diterima, *dataset* yang sudah tersedia di Amazon QuickSight dibuka untuk mengidentifikasi *query* asal yang digunakan untuk membuat *dataset* tersebut. Langkah selanjutnya adalah mengambil dan menyalin *query* tersebut ke dalam AWS Athena. Penambahan kolom dilakukan sesuai dengan permintaan ke dalam *query* dan melakukan penyesuaian logika jika diperlukan. *Query* tersebut kemudian dijalankan untuk memastikan bahwa kolom berhasil ditambahkan dan data yang ditampilkan sesuai dengan kebutuhan *user*. Jika hasilnya sudah sesuai, langkah selanjutnya adalah kembali ke Amazon QuickSight dan memilih opsi untuk mengedit *dataset* yang bersangkutan. Pada tahap ini, *query* yang telah diperbarui dari Athena akan disalin ke dalam *query editor* di QuickSight, menggantikan *query* sebelumnya. Setelah itu, *dataset* diperbarui dan dipublikasikan ulang agar kolom baru bisa langsung digunakan dalam *dashboard* yang terkait.



Gambar 3.20 Contoh Permintaan Penambahan Kolom pada *Dataset* melalui JIRA

Setelah pembaruan selesai dilakukan, semua perubahan dicatat ke dalam *platform* dokumentasi yang digunakan oleh tim, yaitu Airtable. Gambar 3.21 menunjukkan tampilan dari halaman dokumentasi di Airtable yang mencantumkan *query* lama di bagian "*backup*" dan *query* yang sudah diperbarui di bagian "*Active Query*". Dokumentasi ini dilakukan untuk mencatat seluruh aktivitas yang telah dikerjakan, termasuk perubahan pada *query*. Dengan mencatat perubahan, semua anggota tim dapat menelusuri riwayat modifikasi *dataset* dan memahami alasan di balik setiap pembaruan. Hal ini juga membantu proses evaluasi apabila terjadi kendala pada masa mendatang.

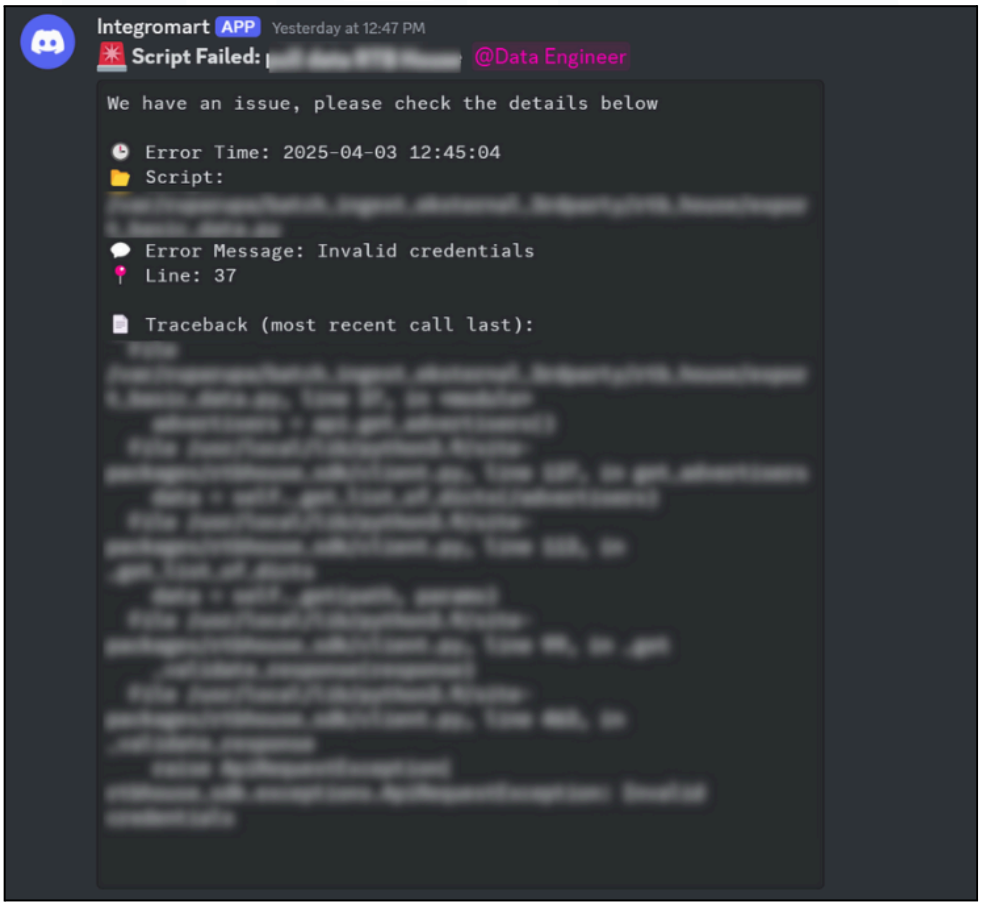


Gambar 3.21 Tampilan Dokumentasi *Query Dataset* di Airtable

3.2.7 Mengelola Skrip Python untuk Otomasi Proses Data (Week 3 - 20)

Dalam mendukung efisiensi proses pengolahan data di Ruparupa, *Data Engineer Intern* bertugas mengelola berbagai skrip Python yang digunakan untuk mengotomasi alur kerja. Otomasi ini dirancang untuk mengurangi ketergantungan pada eksekusi manual, mempercepat pengiriman informasi, dan meningkatkan akurasi proses. Terdapat berbagai jenis otomasi yang dijalankan melalui skrip Python, mulai dari penarikan data melalui *query* SQL yang hasilnya dikirim langsung ke *email user*, proses *scraping* data dari *website* untuk disimpan sebagai tabel di *Data Lake*, hingga penarikan data dari sistem eksternal melalui API. Setiap skrip disesuaikan dengan kebutuhan spesifik pengguna atau tim terkait.

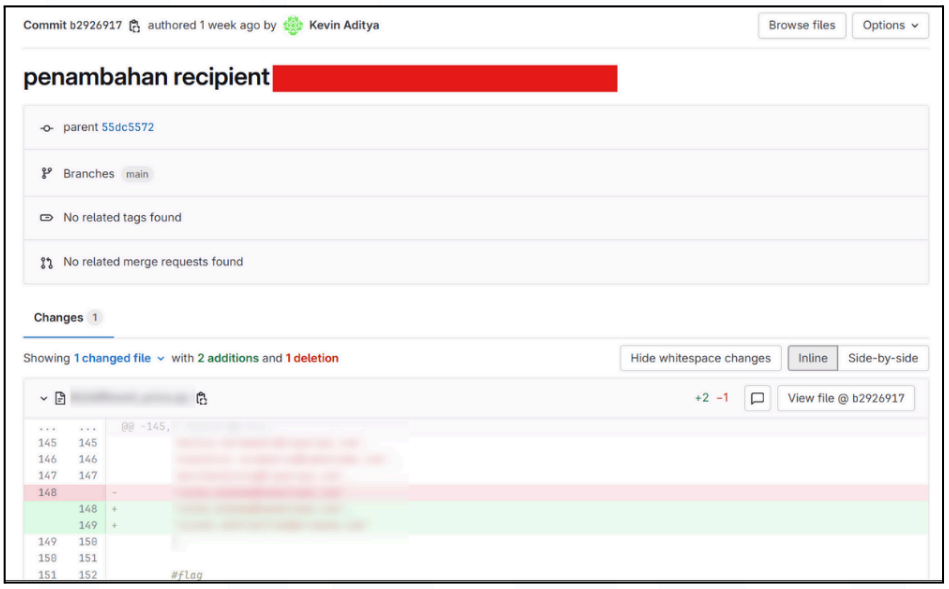
Agar proses otomasi tetap berjalan sebagaimana mestinya, diperlukan kegiatan *maintenance* secara rutin terhadap skrip-skrip yang telah berjalan. Seiring waktu, berbagai *error* dapat muncul akibat perubahan struktur data, tampilan *website*, atau *endpoint* API yang digunakan. Ketika terjadi gangguan atau *error* dalam eksekusi skrip otomasi, notifikasi *error* umumnya dikirimkan melalui Discord, sebagaimana ditunjukkan pada Gambar 3.22. Notifikasi ini menjadi penanda untuk segera melakukan pemeriksaan dan perbaikan terhadap skrip yang bermasalah.



Gambar 3.22 Notifikasi *Error* Otomasi pada Discord

Langkah pertama yang dilakukan adalah menelusuri skrip terkait yang tersimpan di *repository* GitLab. Setelah mengidentifikasi bagian dari skrip yang menyebabkan *error*, dilakukan proses *debugging* untuk mengecek penyebab kegagalan. Perbaikan dilakukan dengan memperbarui *logic* atau menyesuaikan struktur kode agar sesuai dengan format data terbaru atau perubahan sistem

eksternal. Skrip yang telah diperbaiki kemudian diuji kembali untuk memastikan bahwa masalah sudah teratasi dan hasil otomasi kembali normal. Setelah proses perbaikan selesai, tidak lupa untuk memberikan komentar yang menjelaskan perubahan yang dilakukan pada bagian *commit* di GitLab. Komentar ini bertujuan untuk mendokumentasikan alasan dan konteks dari setiap pembaruan yang dilakukan. Setelah komentar dituliskan dan perubahan dipastikan benar, skrip akan di-*push* ke *repository* GitLab. Ilustrasi proses ini dapat dilihat pada Gambar 3.23. Dokumentasi ini penting untuk menjaga transparansi dan memudahkan anggota tim lain dalam melakukan *review* atau tindak lanjut di masa mendatang.



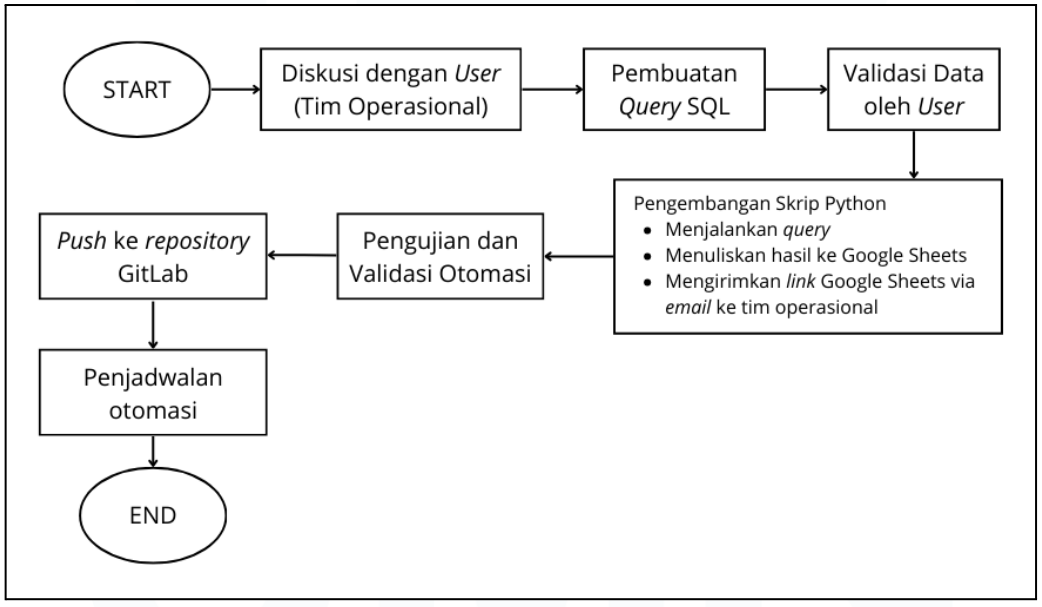
Gambar 3.23 *Commit* Perubahan Skrip Python di GitLab

3.2.8 Mengerjakan Proyek *Reconciliation Data Validator for Shipper Comparison* (Minggu ke 3 - 7)

Sebelumnya, proses rekonsiliasi dilakukan secara manual melalui penarikan data dari Amazon QuickSight yang volumenya besar dan tidak disesuaikan dengan kebutuhan analisis. Hal ini menyebabkan proses verifikasi data menjadi tidak efisien dan memakan waktu. Oleh karena itu, dibutuhkan solusi otomasi agar data yang dibutuhkan dapat langsung dikirimkan secara rutin dan terfilter kepada tim operasional melalui *email*. Proyek ini memiliki urgensi tinggi karena secara langsung mendukung akurasi dalam pencocokan data *shipment* dengan

vendor dan berkontribusi pada efisiensi kerja tim operasional dalam menghadapi volume pengiriman yang besar.

Seperti yang tertera pada Gambar 3.24, proses diawali dengan sesi diskusi antara *intern* dan *user* untuk memahami kebutuhan bisnis, alur rekonsiliasi yang berjalan, serta sumber data yang perlu digunakan. Berdasarkan hasil diskusi tersebut, *query* SQL disusun untuk membentuk *dataset* rekonsiliasi sesuai permintaan *user*. Pada Gambar 3.25 ditunjukkan bahwa *query* tersebut menggabungkan data dari beberapa tabel dan *database* untuk menghasilkan informasi yang dibutuhkan. Setelah hasil *query* dikonfirmasi valid, penyusunan skrip otomatisasi berbasis Python dilakukan untuk menulis data hasil *query* ke Google Sheets dan mengirimkan tautannya melalui *email* kepada tim operasional setiap bulan.



Gambar 3.24 *Flow* Proses Otomasi Pengiriman Data Rekonsiliasi untuk Tim Operasional

Gambar 3.25 Potongan *Query* SQL untuk *Dataset* Rekonsiliasi Pengiriman

Skrip *Python* pada *Gambar 3.26* terdiri dari beberapa *function* yang memiliki peran spesifik. *Function auth_gspread()* digunakan untuk melakukan autentikasi terhadap layanan *Google Sheets* dengan memanfaatkan kredensial dari *Service Account* Google. Proses autentikasi ini memuat berkas *JSON* dari sistem *secrets manager*, menentukan cakupan (*scope*) akses *API*, dan menghasilkan objek kredensial yang dapat digunakan untuk mengakses *spreadsheet*. Selanjutnya, *function clear_sheet(sheet*) digunakan untuk menghapus seluruh isi lembar kerja *Google Sheet* sebelum data baru dituliskan, guna memastikan tidak ada data lama yang tertinggal. *Function* utama *write_to_google_sheets(data)* digunakan untuk melakukan koneksi ke *Google Sheets* menggunakan objek hasil autentikasi, memilih *worksheet* berdasarkan nama, serta mengatur format tampilan seperti teks *bold*, ukuran *font*, dan *alignment*. Fungsi ini juga menggunakan *library set_with_dataframe()* untuk menuliskan data *Data Frame* ke dalam *Google Sheet* secara langsung dan terstruktur.

```python
# Load kredensial dari Google Service Account JSON
def auth_gspread():
    google_cred = json.loads(secrets_google.get_secret())
    scope = ["https://spreadsheets.google.com/feeds", "https://www.googleapis.com/auth/drive"]
    creds = ServiceAccountCredentials.from_json_keyfile_dict(google_cred, scope)
    return gspread.authorize(creds)

# Fungsi untuk menghapus semua data di Sheet
def clear_sheet(sheet):
    sheet.clear()

# Fungsi untuk menulis data ke Google Sheet
def write_to_google_sheets(data):
    gc = auth_gspread()
    sh = gc.open_by_key(SPREADSHEET_ID)

    try:
        worksheet = sh.worksheet(SHEET_NAME)
        worksheet.clear()  # Bersihkan data sebelum menulis ulang
    except gspread.exceptions.WorksheetNotFound:
        print(f"{SHEET_NAME} tidak ditemukan, membuat sheet baru.")
        worksheet = sh.add_worksheet(title=SHEET_NAME, rows="30000", cols="12")

    # Format sheet
    worksheet.format('A1:Z1', {'textFormat': {'bold': True, 'fontSize': 10}, "wrapStrategy": "CLIP"})
    worksheet.format('A2:Z', {"wrapStrategy": "CLIP", "horizontalAlignment": "LEFT"})

    # Tulis data ke sheet
    set_with_dataframe(worksheet, data)
    print(f"Data berhasil ditulis ke {SHEET_NAME}.")
```

Gambar 3.26 Skrip Python untuk Autentikasi dan Penulisan Data ke Google Sheets

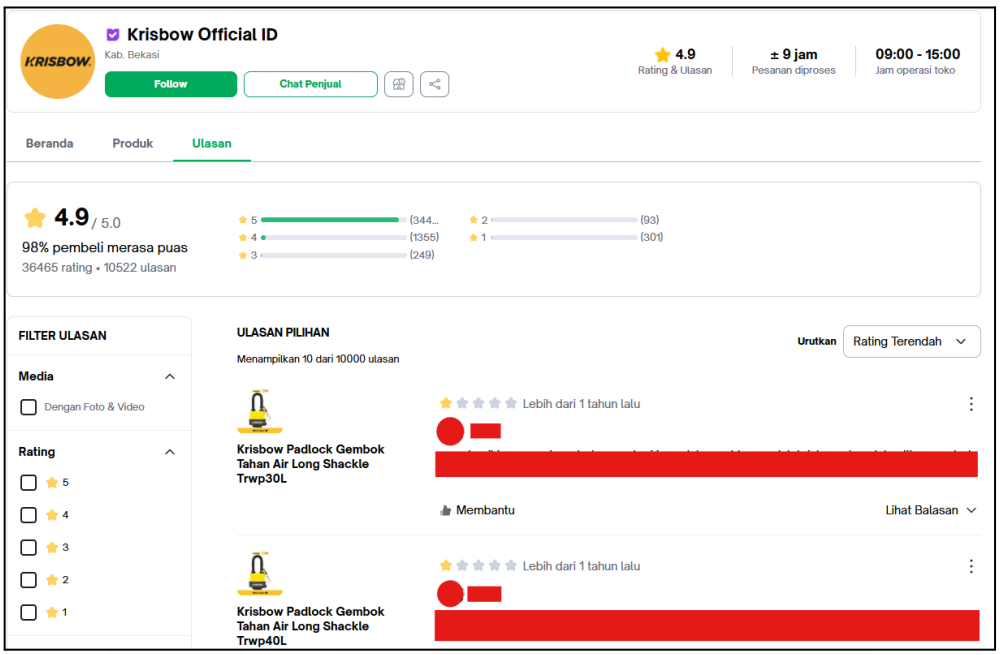
Setelah data berhasil ditulis ke Google Sheet selanjutnya *link* akan dikirimkan kepada *user* melalui *email* menggunakan *function* yang ada pada Gambar 3.27, skrip akan menghitung periode laporan berdasarkan tanggal eksekusi. Periode laporan dihitung dua bulan ke belakang dari tanggal saat skrip dijalankan, misalnya untuk eksekusi di bulan Maret, maka data yang dikirim adalah periode bulan Januari sampai Februari. *Link* dari Google Sheet yang telah terisi data akan dimasukkan ke dalam isi *email* dalam format teks dan HTML. Isi email disusun secara terstruktur, menjelaskan tujuan dari data yang dikirim, periode yang dicakup, serta memberikan catatan tambahan untuk membantu *user* memahami konteks data. Subjek *email* juga disesuaikan dengan periode data agar lebih informatif. *Email* tersebut dikirim menggunakan layanan Amazon SES, dengan daftar penerima yang telah ditentukan oleh tim operasional. Setelah *email* berhasil dikirim, skrip akan di *push* ke *repository* GitLab. Penjadwalan otomatisasi untuk menjalankan skrip ini setiap bulan selanjutnya dikelola oleh *Data Engineer full-time* menggunakan *scheduler* yang telah ditentukan.

```python
write_to_google_sheets(data_df)

# Hitung periode laporan
today = datetime.today()
start_period = (today.replace(day=1) - relativedelta(months=2)).strftime('%d %B %Y')
end_period = (today.replace(day=1) - timedelta(days=1)).strftime('%d %B %Y')

# Link Google Sheet
gsheet_link = f"https://docs.google.com/spreadsheets/d/{SPREADSHEET_ID}/edit#gid=0"

# Isi email
message = #Pesan Email

SUBJECT = f'Monthly Report: Shipping Reconciliation Data Periode {start_period} - {end_period}'

BODY_HTML = #Isi message email dalam format HTML

RECIPIENTS = #Email User

SENDER = #Address must be verified with Amazon SES.

# Kirim email
send_email.send_email(SENDER, RECIPIENTS, SUBJECT, message, BODY_HTML)
```

Gambar 3.27 Skrip Python untuk Mengirim *Link* Google Sheet ke *Email User*

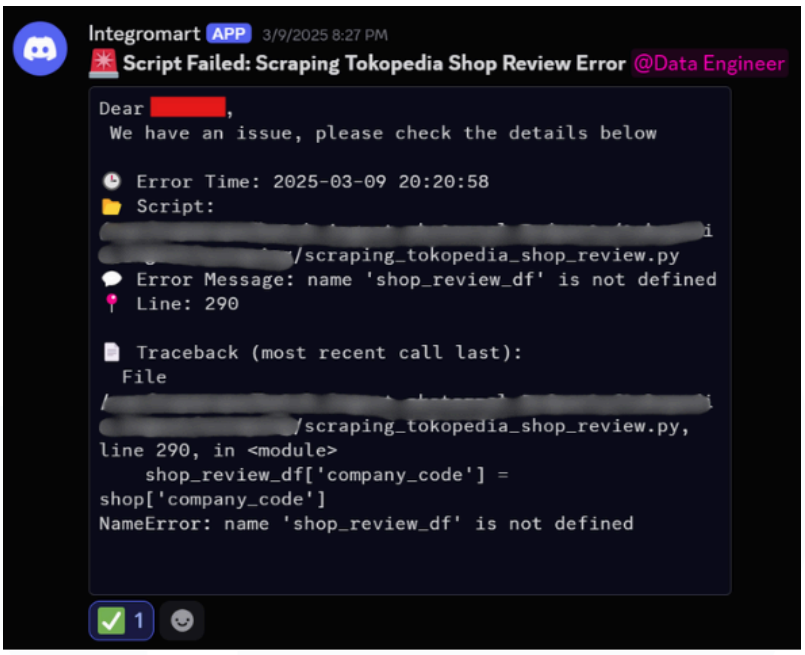
3.2.9 Memperbaharui Skrip *Scraping Tokopedia Shop Review* (Minggu ke 6)

Proyek *Scraping Tokopedia Shop Review* bertujuan untuk melakukan ekstraksi data ulasan toko yang menjual produk dari BU Kawan lama *Group* di Tokopedia (Gambar 3.28) secara otomatis menggunakan Python. Skrip ini digunakan untuk membantu tim *Data Analyst* dalam memantau kualitas pelayanan toko dan mengidentifikasi keluhan pelanggan melalui ulasan secara *real-time*. Otomatisasi ini penting karena mempercepat proses pengumpulan data yang sebelumnya dilakukan secara manual dan mengurangi risiko *human error*. Dengan data yang diperoleh, tim dapat mengambil tindakan lebih cepat dalam menanggapi isu-isu pelanggan serta menyusun strategi perbaikan layanan yang lebih tepat sasaran.



Gambar 3.28 Tampilan Halaman *Tokopedia Shop Review*

Skrip Python ini sebelumnya telah berhasil digunakan, namun mengalami *error* ketika struktur halaman Tokopedia mengalami perubahan seperti yang terlihat pada Gambar 3.29. Perubahan ini terjadi karena adanya pembaruan pada tampilan halaman ulasan di *website* Tokopedia, yang secara tidak langsung mempengaruhi struktur HTML-nya. Akibat dari perubahan tersebut, elemen-elemen HTML yang sebelumnya menjadi acuan dalam proses ekstraksi data menjadi tidak ditemukan lagi oleh skrip. Hal ini menyebabkan variabel *shop_review_df* tidak dapat terdefinisi, yang pada akhirnya membuat keseluruhan proses scraping gagal berjalan.



Gambar 3.29 Notifikasi *Error Scraping Tokopedia Shop Review* pada Discord

Sebagai langkah perbaikan, dilakukan penyesuaian terhadap *selector* CSS pada skirp agar kembali sesuai dengan struktur HTML terbaru di halaman ulasan Tokopedia seperti pada Gambar 3.30. Selain itu, waktu tunggu (*sleep*) pada elemen *list_button_block_membantu_lihat_balasan* juga ditingkatkan dari sebelumnya 2 dan 1 detik menjadi 3 dan 2 detik. Penambahan waktu tunggu ini bertujuan untuk memberikan waktu lebih bagi halaman web dalam memuat elemen-elemen dinamis, sehingga dapat menghindari *error* seperti *Element Click Intercepted Exception* yang sering terjadi ketika elemen belum sepenuhnya dimuat atau tertutup oleh elemen lain.

Gambar 3.30 Perubahan yang Dilakukan Pada Skrip *Scraping Tokopedia Shop Review*

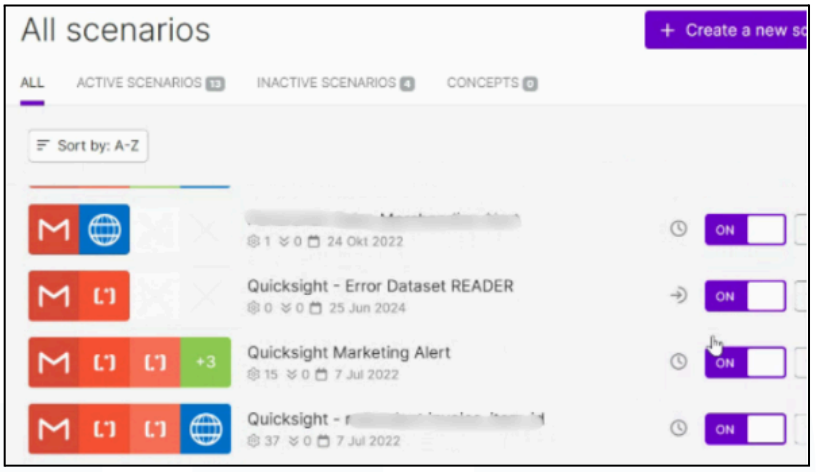
Sebagai peningkatan tambahan, dilakukan penambahan blok kode untuk mendukung *setup browser* pada sistem operasi Windows, mengingat beberapa pengguna internal menggunakan sistem operasi tersebut. Selain itu, skrip juga dimodifikasi untuk menampilkan jumlah data yang berhasil dikumpulkan oleh variabel *shop_review_df* dan *shop_review_summary_today_df* seperti pada Gambar 3.31. Tujuan dari penambahan *log* jumlah data adalah untuk memastikan apakah hasil *scraping* menghasilkan data atau tidak, sehingga proses *debugging* dan validasi dapat dilakukan dengan lebih cepat dan efisien. Setelah seluruh perbaikan diterapkan, skrip kemudian di-*push* ke *repository* GitLab untuk didokumentasikan dan dapat digunakan dalam proses otomasi selanjutnya. Dengan pembaruan ini, proses *scraping* ulasan Tokopedia kembali berjalan tanpa kendala dan data dapat dikumpulkan secara otomatis sesuai kebutuhan.

```
    for shop in tokopedia_shop_sheet_dict:
        print('scraping ' + shop['shop_name'] + '...')
        attempts = 0
@@ -297,7 +311,8 @@ try:
        shop_review_df['date'] = pd.to_datetime(shop_review_df['date'])
        shop_review_df['feedback'] = shop_review_df['feedback'].astype('str')
        shop_review_df['feedback_date'] = pd.to_datetime(shop_review_df['feedback_date'])
-
+        print(f"data count for shop review {shop['shop_name']}: {len(shop_review_df)}")
+
        unique_dates = pd.to_datetime(shop_review_df['date'].unique
        ()).strftime('%Y-%m-%d')

@@ -316,7 +331,7 @@ try:
            )
            print(f"Uploaded {file_name} to S3 bucket {settings['BUCKET_NAME']}")

-        #GET SHOPEE SHOP REVIEW SUMMARY
+        #GET SHOP REVIEW SUMMARY
        shop_review_summary_today_df['company_code'] = shop['company_code']
        shop_review_summary_today_df['shop_name'] = shop['shop_name']

@@ -326,6 +341,7 @@ try:
        shop_review_summary_today_df['log_date'] = pd.to_datetime(shop_review_summary_today_df['log_date'])
        shop_review_summary_today_df['rating_count'] = shop_review_summary_today_df['rating_count'].astype('int')

+        print(f"data count for shop review summary {shop['shop_name']}: {len(shop_review_summary_today_df)}")
        #UPLOAD RESULT TO S3
        file_name = (datetime.now()).strftime('%Y-%m-%d') + '_' + shop['shop_name'] + '_' + 'shop_review_summary.parquet'
        parquet_buffer_2 = BytesIO()
```

Gambar 3.31 Kode Penambahan *Log* untuk Menampilkan Jumlah Data Hasil *Scraping*

3.2.10 Memantau Otomasi di Make.com (Minggu ke 2 - 20)

Dalam mendukung kelancaran proses kerja tim Data Engineer, terutama dalam pemantauan data dan deteksi *error*, digunakan sebuah *platform* otomasi bernama Make.com. Make.com merupakan web berbasis *low-code* yang memungkinkan pengguna membuat alur kerja otomasi secara visual seperti pada Gambar 3.32. Salah satu skenario otomasi penting yang dibangun melalui platform ini adalah pengiriman notifikasi terkait *error* ke Discord tim Data Ruparupa, sehingga tim dapat segera menindaklanjuti informasi tersebut secara *real-time*. Otomasi ini sangat membantu dalam menjaga kualitas data dan meminimalisir keterlambatan dalam merespons isu yang muncul di lapangan.



Gambar 3.32 Skenario Otomasi Menggunakan Make.com

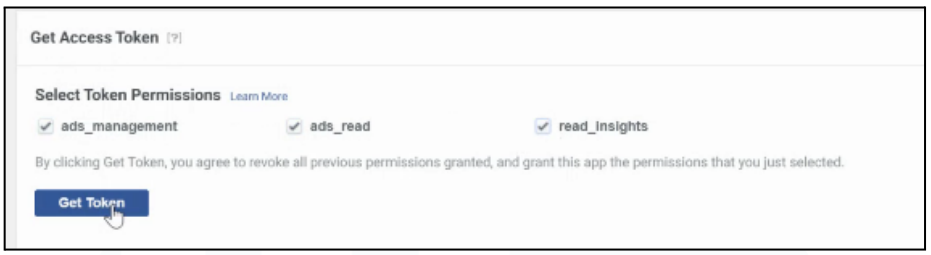
Tugas *intern* adalah memastikan bahwa skenario otomasi berjalan dengan lancar tanpa hambatan. Salah satu hal yang perlu diperhatikan adalah masa berlaku otorisasi akses ke layanan eksternal seperti Gmail, karena Make.com membutuhkan koneksi ter-autentikasi untuk dapat mengakses dan mengirim notifikasi email. Dalam kasus ini, koneksi Gmail *supervisor* digunakan sebagai akun pengirim notifikasi. *S*tatus otorisasi ini perlu dicek secara berkala melalui menu *More* > *Connection* > Gmail Google di Make.com, untuk memastikan apakah koneksi masih aktif atau perlu diperbarui (*reauthorize*).

Jika terdapat tanda bahwa koneksi telah atau akan segera kedaluwarsa, maka langkah *reauthorize* harus segera dilakukan. Proses ini melibatkan *login* ulang ke akun Gmail *supervisor* melalui laptop *intern*, lalu melakukan klik *Reauthorize* > *Advanced* > *Continue*, dan menunggu proses verifikasi hingga selesai. Sangat penting untuk memastikan bahwa *reauthorize* dilakukan sebelum koneksi benar-benar mati, karena jika tidak, maka seluruh skenario otomasi yang bergantung pada koneksi tersebut akan berhenti berjalan. Oleh karena itu, pemantauan dan perpanjangan koneksi secara proaktif menjadi sangat penting supaya sistem notifikasi tetap aktif dan mendukung proses bisnis secara berkelanjutan.

3.2.11 Mengelola Pembaruan Token API Sesuai Jadwal (Minggu ke 2, 4, 6, 8, 10, 12, 14, 16, 18, dan 20)

Untuk menjaga keberlanjutan dan kestabilan proses pengambilan data dari *platform* pihak ketiga seperti Facebook, sangat penting untuk secara rutin memperbarui API Token yang digunakan dalam proses otomasi. Token ini berfungsi sebagai kredensial akses yang memungkinkan sistem untuk terhubung dengan *Facebook Marketing API* dan menarik data iklan yang diperlukan oleh tim marketing atau analitik. Tanpa pembaruan berkala, token dapat kadaluarsa, yang akan menyebabkan proses integrasi data menjadi gagal atau berhenti. Oleh karena itu, update token ini dijadwalkan secara rutin setiap tanggal 10 dan 25 setiap bulannya.

Flow pembaruan token dilakukan melalui beberapa langkah sederhana namun krusial. Pertama, pengguna perlu mengakses *Data Ruparupa API > Marketing API > Tools*, lalu mencentang tiga opsi permissions, yaitu *ads_management, ads_read,* dan *read_insights* seperti pada Gambar 3.33. Setelah itu, klik tombol *"Get Token"* untuk menghasilkan token baru yang kemudian disalin dengan memastikan tidak ada spasi atau karakter yang terpotong. Token yang telah disalin selanjutnya ditempelkan ke dalam sistem penyimpanan, yaitu AWS Secrets Manager, dengan mengakses *AWS > Secret Manager > secrets > EC2 Secret Credential*. Di sana, pengguna mengedit nilai dari parameter *Facebook Token* dengan menempelkan token baru. Perlu dipastikan kembali tidak ada kesalahan salin tempel seperti spasi tambahan agar sistem dapat membaca token dengan benar. Pembaruan token secara rutin ini bertujuan untuk memastikan seluruh proses integrasi dan otomasi pengambilan data iklan tetap berjalan tanpa hambatan. Proses ini mendukung pengambilan keputusan yang cepat dan berbasis data, terutama bagi tim pemasaran dan pengelolaan campaign digital.



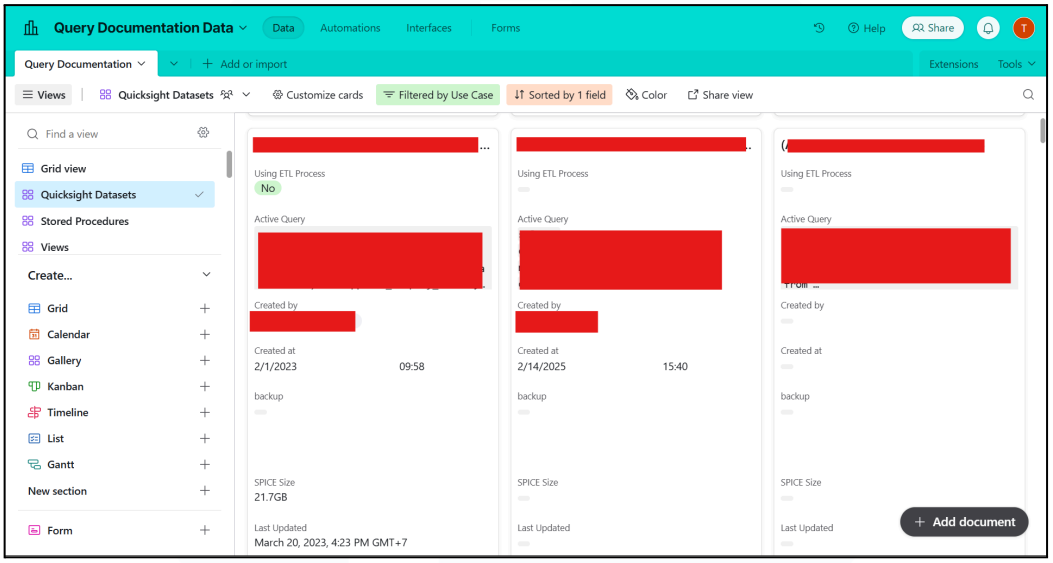
Gambar 3.33 Opsi *Permissions* Token API Facebook Ads

3.2.12 Membuat dan Memperbarui Dokumentasi Terkait Definisi Data, *Workflow*, Serta *Dataset Queries* (Minggu ke 1 - 20)

Untuk meningkatkan pemahaman terhadap data serta memperlancar kolaborasi antar tim, dokumentasi yang jelas dan terstruktur sangat diperlukan. Dokumentasi menjadi aspek krusial dalam ekosistem data karena tidak hanya membantu tim data dalam menjaga konsistensi dan akurasi informasi, tetapi juga menjadi panduan bagi tim non-teknis dalam menafsirkan data secara tepat. Di Ruparupa, dokumentasi berperan penting dalam memastikan bahwa setiap proses, definisi, dan transformasi data dapat dipahami secara menyeluruh oleh berbagai pihak, termasuk pihak yang tidak terlibat langsung dalam pembuatan *dataset*. Selain itu, dokumentasi juga memudahkan proses audit, *troubleshooting*, dan *onboarding* anggota tim baru, sehingga pengetahuan teknis tidak hilang atau terputus meskipun terjadi rotasi tim atau pergantian personel. Dengan adanya dokumentasi yang komprehensif, proses pengambilan keputusan berbasis data dapat dilakukan secara lebih cepat, tepat, dan berlandaskan informasi yang seragam.

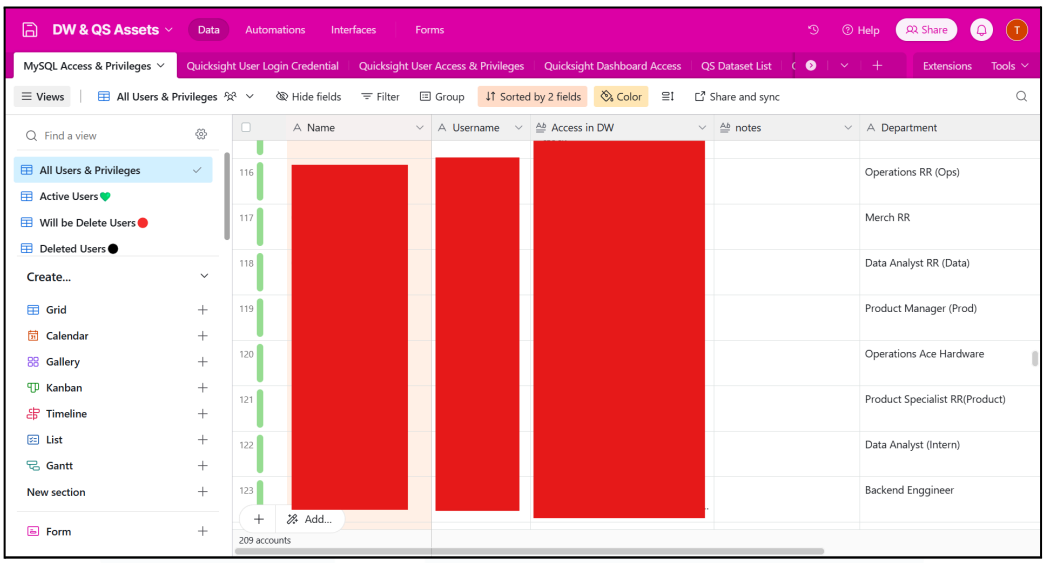
Sebagian besar dokumentasi terkait pengelolaan data di Ruparupa dilakukan menggunakan Airtable, sebuah *platform* berbasis *cloud* yang menggabungkan kemudahan tampilan *spreadsheet* dengan kekuatan fitur *relational database*. Airtable memungkinkan penyusunan dokumentasi secara terstruktur dan fleksibel, sehingga setiap informasi dapat diorganisasi dan diakses dengan mudah oleh berbagai pihak yang terlibat. Dalam Airtable, berbagai bentuk dokumentasi telah dilakukan untuk mendukung transparansi dan akuntabilitas dalam pengelolaan data. Sebagaimana ditunjukkan pada Gambar 3.34, dokumentasi *query* SQL disusun untuk menjelaskan logika di balik *dataset* yang digunakan dalam

dashboard maupun kebutuhan operasional. Hal ini bertujuan agar seluruh tim, baik teknis maupun non-teknis, dapat memahami sumber data, transformasi yang dilakukan, dan cara kerja dataset tersebut secara menyeluruh.



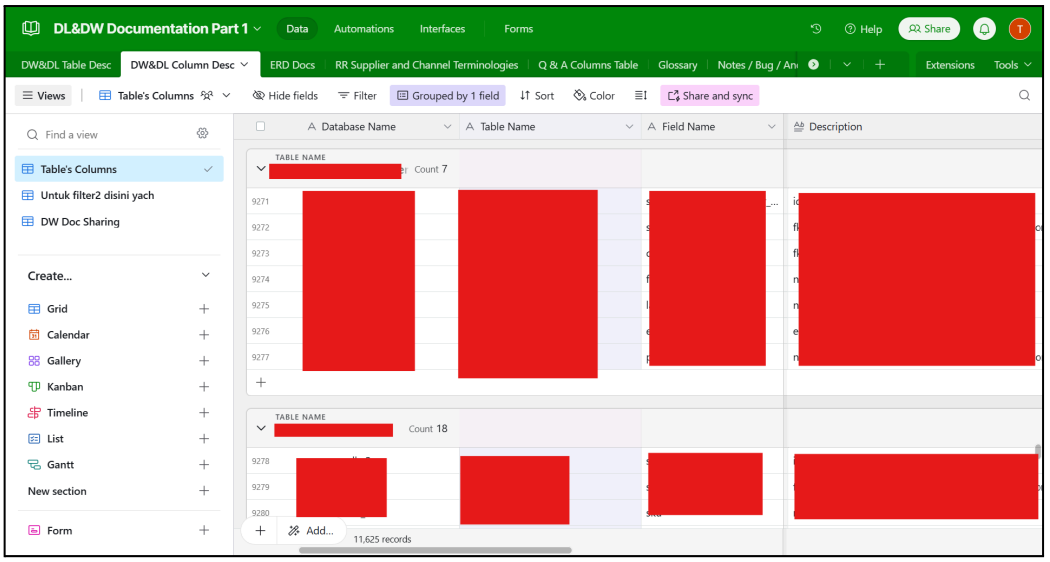
Gambar 3.34 Dokumentasi SQL *Query* untuk *Dataset* Operasional dan *Dashboard*

Selanjutnya pada Gambar 3.35, terdokumentasi informasi terkait akses *database* oleh pengguna, termasuk siapa saja yang memiliki hak akses dan jenis izin yang diberikan. Dokumentasi ini penting untuk memastikan keamanan data serta mendukung proses audit internal. Melalui pencatatan ini, kontrol terhadap akses data menjadi lebih terstruktur, sehingga risiko kebocoran atau penyalahgunaan data dapat ditekan. Selain itu, dokumentasi ini juga mempermudah proses pemberian atau pencabutan akses seiring perubahan kebutuhan tim.



Gambar 3.35 Dokumentasi Hak Akses *Database* Pengguna

Pada Gambar 3.36, ditampilkan dokumentasi definisi kolom untuk tabel-tabel dalam *Data Warehouse*. Penjelasan mengenai arti kolom, tipe data, serta relasi antar tabel sangat membantu dalam proses analisis dan pengembangan lebih lanjut. Dokumentasi ini berfungsi sebagai data *dictionary* yang menjadi rujukan utama, khususnya bagi anggota tim baru atau pihak yang tidak terlibat langsung dalam pembuatan dataset. Dengan pemahaman struktur data yang seragam, komunikasi antar tim menjadi lebih efisien dan tepat sasaran.

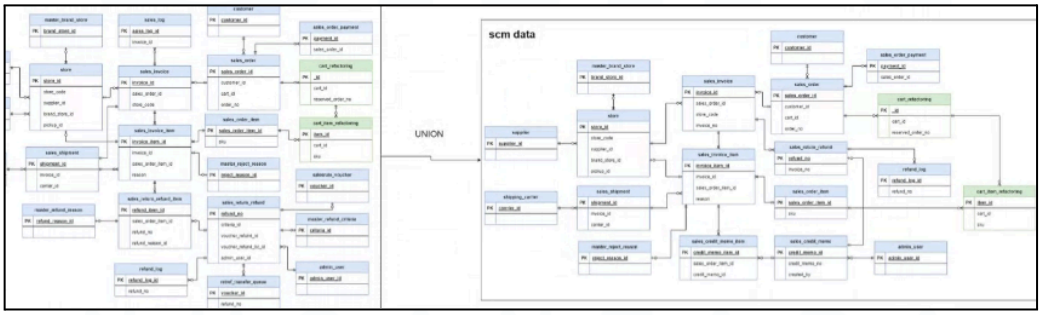


Gambar 3.36 Dokumentasi Definisi Kolom untuk Tabel-tabel Dalam *Data Warehouse*

Dokumentasi yang dibuat di Airtable memberikan kontribusi besar dalam membangun ekosistem data yang terstruktur, transparan, dan mudah dipelihara. Selain mempercepat proses *onboarding*, dokumentasi ini juga mendukung keberlangsungan proyek jangka panjang dengan memastikan bahwa pengetahuan teknis tidak hilang saat terjadi pergantian tim. Dengan kata lain, dokumentasi menjadi pondasi penting dalam menjaga keberlanjutan, efisiensi, dan kualitas dalam pengelolaan data di Ruparupa.

3.2.13 Membuat ERD untuk Membantu Pemahaman Mengenai Database Perusahaan (Week 1 - 2)

Pemahaman struktur dan hubungan antar tabel dalam *database* Ruparupa menjadi hal yang sangat penting, terutama bagi *intern* yang baru bergabung dan perlu beradaptasi dengan cepat. Untuk mendukung proses pembelajaran ini, dibuatlah *Entity Relationship Diagram* (ERD), yakni sebuah representasi visual yang menggambarkan hubungan antar entitas dalam *database*. Diagram ini membantu dalam mengilustrasikan bagaimana satu tabel berelasi dengan tabel lainnya, sehingga dapat mempercepat pemahaman mengenai alur data serta proses bisnis yang mendasarinya. Salah satu contoh ERD yang telah disusun ditampilkan pada Gambar 3.37, yang menggambarkan hubungan antar tabel dalam salah satu dataset penting di perusahaan. Pembuatan ERD tersebut didasarkan pada analisis langsung terhadap struktur database dan relasi antar tabel yang ditemukan melalui eksplorasi data. Salah satu pendekatan yang digunakan untuk memahami keterkaitan tabel adalah dengan menjalankan query SQL tertentu, seperti pada Gambar 3.38.

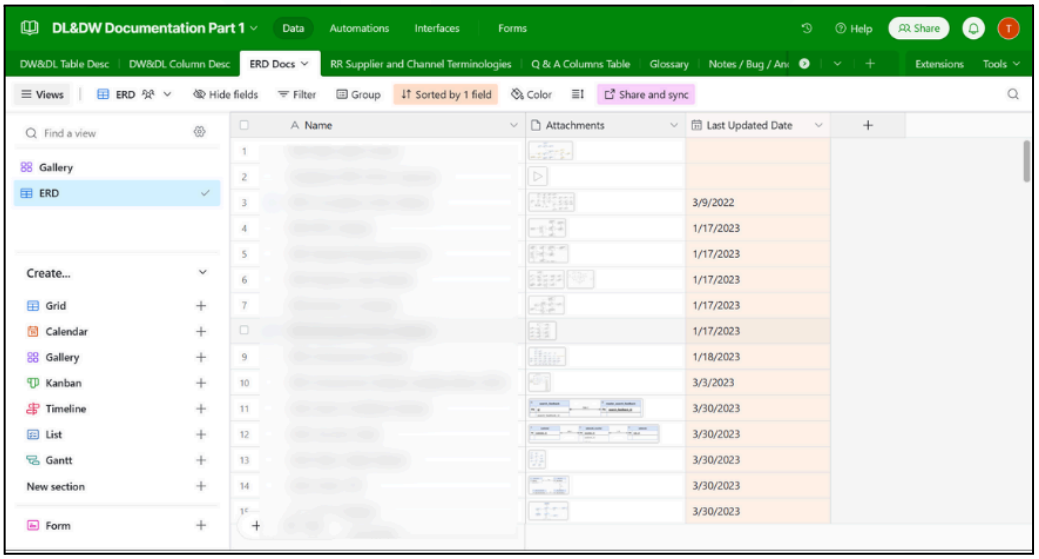


Gambar 3.37 Contoh *Entity Relationship Diagram* (ERD) untuk *Dataset* Internal Ruparupa

```
select 'primary key tabel A', count('primary key tabel B') as counts
from '                .tabel B' left join '                .tabel A' on 'tabel A.penghubung' = 'tabel B.penghubung'
group by 'primary key tabel A'
having count('tabel B.penghubung') < 1
```

Gambar 3.38 *Query* SQL untuk Mengecek Relasi antar Tabel dalam Pembuatan ERD

Query ini digunakan untuk mengidentifikasi apakah terdapat baris pada tabel A yang tidak memiliki relasi dengan data pada tabel B. Dengan metode LEFT JOIN, seluruh data dari tabel A akan ditampilkan meskipun tidak ada padanan di tabel B. Proses ini sangat berguna dalam memverifikasi hubungan antar tabel serta menemukan data yang mungkin tidak terhubung sebagaimana mestinya. Validasi ini memberikan landasan yang kuat dalam penyusunan ERD agar hasil visualisasi benar-benar mencerminkan struktur data yang aktual di dalam sistem. Sebagai bagian dari dokumentasi, seluruh ERD yang telah dibuat juga disimpan secara sistematis di Airtable, yang berfungsi sebagai pusat dokumentasi teknis tim data. Salah satu tampilan dokumentasi ERD yang disimpan di Airtable dapat dilihat pada Gambar 3.39.



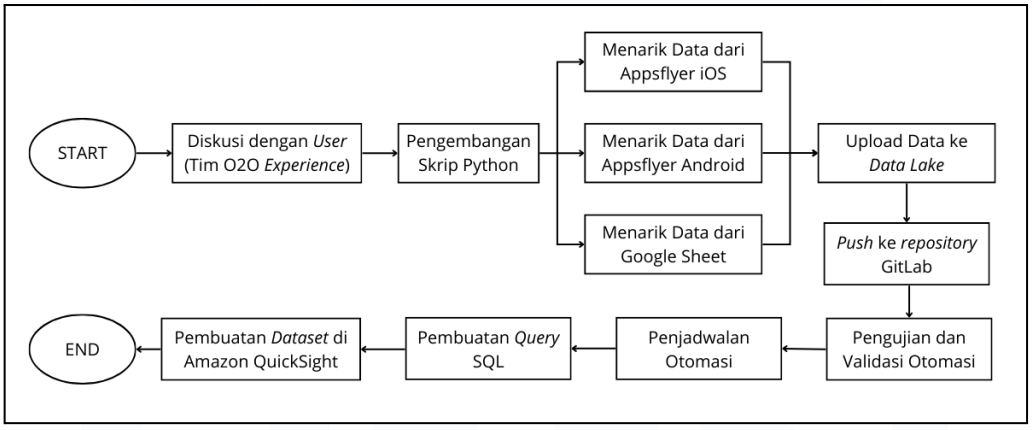
Gambar 3.39 Dokumentasi Visual ERD pada *Platform* Airtable

3.2.14 Mengerjakan Proyek *Appsflyer Onelink Master Data Automation* (Week ke 8 - 9)

Appsflyer merupakan *platform attribution* yang digunakan untuk mengukur dan menganalisis performa kampanye pemasaran digital, khususnya pada kanal aplikasi *mobile*. Salah satu fitur penting dari platform ini adalah *Onelink*, yaitu *link* universal yang mengarahkan pengguna secara otomatis ke platform yang

sesuai (iOS, Android, atau *browser*) berdasarkan perangkat yang digunakan. Tim *O2O Experience* di Ruparupa memanfaatkan fitur ini untuk membuat berbagai *campaign* yang bertujuan meningkatkan akuisisi dan konversi pengguna. Oleh karena itu, dibutuhkan data historis terkait performa *Onelink* seperti jumlah klik per *campaign* dan metrik lainnya, yang dapat digunakan untuk proses evaluasi dan optimasi strategi campaign berikutnya.

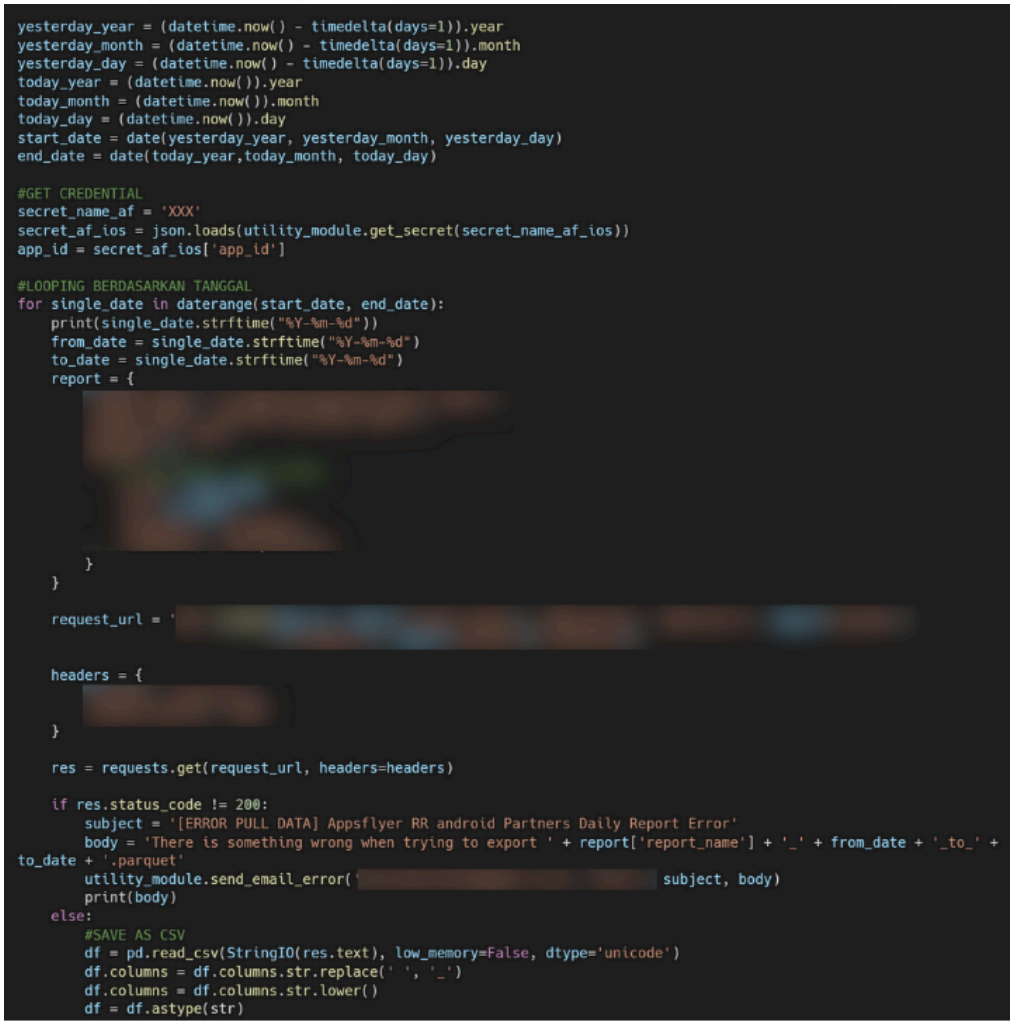
Dalam rangka memenuhi kebutuhan tersebut, perlu dibuat sebuah proses otomasi pengambilan data dari Appsflyer secara berkala seperti pada Gambar 3.40. Tahapan awal dari proyek ini dimulai dengan sesi diskusi bersama tim *O2O Experience* guna memahami kebutuhan pengguna dan menentukan data apa saja yang relevan untuk ditarik dari Appsflyer. Hasil diskusi ini menjadi dasar dalam menyusun skrip otomasi menggunakan Python.



Gambar 3.40 *Flow* Otomasi *Appsflyer Onelink Master Data*

Proses penarikan data dari Appsflyer dilakukan dengan menembak API untuk masing-masing *platform*, seperti yang ditunjukkan pada Gambar 3.41. Skrip Python yang digunakan memanfaatkan modul requests untuk melakukan HTTP GET *request* ke *endpoint* Appsflyer API. Tanggal awal dan akhir ditentukan secara dinamis dengan menggunakan fungsi *datetime* dan *timedelta*, sehingga data dapat diperbarui setiap hari. Selanjutnya, *API key* dan *App ID* diambil secara aman dari AWS Secrets Manager, dan hasil respons yang berbentuk CSV kemudian dibaca ke dalam bentuk *dataframe* menggunakan pandas. Data tersebut selanjutnya disimpan dalam format *Parquet* dan diunggah ke *Data Lake* menggunakan layanan AWS S3, agar bisa digunakan oleh tim *Data Analyst* secara

efisien dan terstruktur. Proses selanjutnya adalah melakukan *push* skrip ke repository GitLab untuk menjaga *versioning* serta kolaborasi tim. Otomasi ini juga melewati tahap pengujian dan validasi untuk memastikan bahwa data yang dihasilkan sesuai dan tidak mengalami *error*. Setelah validasi berhasil, penjadwalan dilakukan agar skrip dapat berjalan secara otomatis setiap hari atau dalam periode tertentu tanpa intervensi manual.

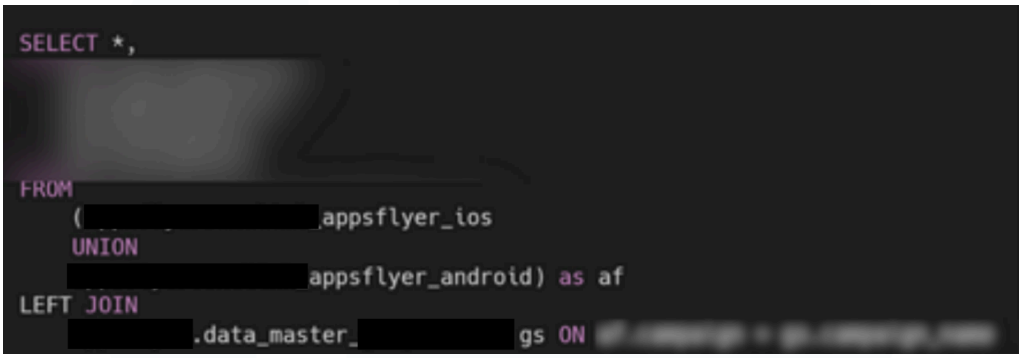


```
yesterday_year = (datetime.now() - timedelta(days=1)).year
yesterday_month = (datetime.now() - timedelta(days=1)).month
yesterday_day = (datetime.now() - timedelta(days=1)).day
today_year = (datetime.now()).year
today_month = (datetime.now()).month
today_day = (datetime.now()).day
start_date = date(yesterday_year, yesterday_month, yesterday_day)
end_date = date(today_year,today_month, today_day)

#GET CREDENTIAL
secret_name_af = 'XXX'
secret_af_ios = json.loads(utility_module.get_secret(secret_name_af_ios))
app_id = secret_af_ios['app_id']

#LOOPING BERDASARKAN TANGGAL
for single_date in daterange(start_date, end_date):
    print(single_date.strftime("%Y-%m-%d"))
    from_date = single_date.strftime("%Y-%m-%d")
    to_date = single_date.strftime("%Y-%m-%d")
    report = {
        }
    }

    request_url = '

    headers = {

    }

    res = requests.get(request_url, headers=headers)

    if res.status_code != 200:
        subject = '[ERROR PULL DATA] Appsflyer RR android Partners Daily Report Error'
        body = 'There is something wrong when trying to export ' + report['report_name'] + '_' + from_date + '_to_' +
to_date + '.parquet'
        utility_module.send_email_error(                                subject, body)
        print(body)
    else:
        #SAVE AS CSV
        df = pd.read_csv(StringIO(res.text), low_memory=False, dtype='unicode')
        df.columns = df.columns.str.replace(' ', '_')
        df.columns = df.columns.str.lower()
        df = df.astype(str)
```

Gambar 3.41 Skrip Penarikan Data Appsflyer via API

Setelah proses unggah ke *data lake* selesai, dibuatlah *query* SQL untuk menggabungkan seluruh sumber data yang tersedia. *Query* ini dibuat dan dijalankan menggunakan Amazon Athena seperti yang digambarkan pada Gambar 3.42. Struktur *query* terdiri dari gabungan (*UNION*) data dari Appsflyer Android dan iOS, yang kemudian di-*left join* dengan *data master campaign* dari Google

Sheet berdasarkan kolom referensi tertentu. *Dataset* hasil *query* ini menjadi sumber utama dalam proses pembuatan *dashboard* di Amazon QuickSight. *Dashboard* ini kemudian digunakan oleh tim *O2O Experience* untuk memonitor performa kampanye, mengukur efektivitas *Onelink*, dan mengambil keputusan strategis berbasis data.



Gambar 3.42 *Query* Penggabungan Data Appsflyer dan Google Sheet

3.3 Kendala yang Ditemukan

Selama menjalani masa magang sebagai *Data Engineer Intern* di Ruparupa, terdapat berbagai tantangan yang dihadapi dalam proses adaptasi, pengerjaan tugas, maupun kolaborasi dengan tim lintas fungsi. Kendala-kendala ini muncul baik dari sisi teknis maupun non-teknis, dan menjadi bagian penting dalam proses pembelajaran. Berikut adalah beberapa kendala utama yang ditemukan selama periode magang beserta penjelasan dan dampaknya terhadap proses kerja:

1. Kurangnya Pemahaman Data di Kalangan Pengguna Divisi Non-Teknis (*User*)

   Salah satu kendala yang sering muncul adalah kurangnya pemahaman dari beberapa *user* terhadap data yang tersedia. Ketidaktahuan mengenai struktur data, sumber data, atau definisi metrik menyebabkan terjadinya miskomunikasi antara *user* dan tim data. Hal ini menghambat proses permintaan data, interpretasi hasil, maupun pengambilan keputusan berbasis data karena adanya ketidaksesuaian ekspektasi dan realita dari data yang dianalisis.
2. Dokumentasi Data yang Belum Lengkap

Masalah lain yang cukup signifikan adalah kurangnya dokumentasi yang lengkap dan terstruktur terkait *dataset*, *query*, serta definisi kolom. Beberapa informasi penting tersebar di berbagai tempat atau tidak terdokumentasi dengan baik, sehingga menyulitkan dalam menelusuri asal-usul data, memahami logika *query*, atau mengetahui kegunaan kolom tertentu dalam tabel. Hal ini memperlambat proses kerja dan meningkatkan risiko kesalahan pemahaman terhadap data.

3. Keterbatasan Spesifikasi Perangkat Kerja

   Perangkat laptop yang digunakan selama masa magang memiliki keterbatasan dari sisi spesifikasi, khususnya dalam hal kapasitas RAM dan kecepatan pemrosesan. Ketika harus menjalankan *query* dengan skala besar atau mengakses data yang kompleks, perangkat sering mengalami kendala performa seperti *lagging*, *not responding*, atau bahkan *crash*. Hal ini mengganggu kelancaran pekerjaan sehari-hari, terutama dalam proses eksplorasi dan pembersihan data.

4. Terbatasnya Pemahaman Awal Terhadap Struktur *Database* Internal

   Di awal periode magang, pemahaman terhadap struktur dan relasi antar tabel dalam database perusahaan masih sangat minim. Kurangnya informasi awal mengenai arsitektur *database* membuat proses pengerjaan tugas menjadi lebih lambat karena perlu waktu ekstra untuk memahami alur data dan koneksi antar tabel. Hal ini menyulitkan dalam mengeksekusi *query* yang tepat serta menginterpretasikan hasil data dengan benar pada tahap-tahap awal pengerjaan proyek.

3.4 Solusi atas Kendala yang Ditemukan

Berbagai tantangan berhasil dihadapi dengan solusi yang disesuaikan dengan konteks permasalahan. Solusi-solusi ini diterapkan secara bertahap, baik melalui pendekatan teknis maupun kolaboratif lintas fungsi. Dengan adanya upaya perbaikan ini, proses kerja menjadi lebih efisien, komunikasi antar tim semakin baik, dan pemahaman terhadap data semakin kuat. Berikut adalah solusi yang

diambil untuk mengatasi kendala-kendala utama yang dihadapi selama masa magang:

1. Meningkatkan Literasi Data di Kalangan User Non-Teknis

   Untuk mengatasi kendala kurangnya pemahaman dari pihak user terhadap struktur dan konteks data, dilakukan pendekatan kolaboratif seperti menjelaskan definisi metrik dan asumsi data secara langsung ketika menerima permintaan data. Ajakan *meeting* atau diskusi singkat bersama *user* juga diinisiasi untuk membahas secara langsung kebutuhan data mereka serta memberikan klarifikasi apabila ditemukan ambiguitas dalam permintaan. Dengan memberikan konteks tambahan terhadap output data, proses pengambilan keputusan menjadi lebih akurat. Upaya ini juga membantu memperkecil gap pemahaman yang dapat menyebabkan miskomunikasi.

2. Menyusun Dokumentasi Data yang Lebih Terstruktur

   Masalah dokumentasi yang belum lengkap dapat ditangani dengan memperbarui dokumentasi di Airtable, *platform* yang digunakan sebagai pusat dokumentasi tim data. Penyusunan ulang dokumentasi terkait *query*, definisi kolom, dan konteks penggunaan dataset dilakukan secara langsung. Proses ini dilakukan secara konsisten agar setiap anggota tim memiliki referensi yang sama ketika mengakses atau memproses data.

3. Mengoptimalkan Penggunaan Perangkat Kerja dengan Spesifikasi Terbatas

   Untuk mengatasi keterbatasan RAM dan kecepatan pemrosesan perangkat kerja, digunakan strategi optimasi *query* dengan membatasi *scope* data (menggunakan *LIMIT*), hanya memilih kolom yang dibutuhkan, dan menghindari operasi berat seperti *JOIN* yang kompleks. Waktu *idle* seperti saat istirahat siang juga dimanfaatkan untuk menjalankan proses berat agar tidak mengganggu *workflow* harian.

4. Mempercepat Pemahaman Struktur *Database* Melalui Visualisasi ERD

   Keterbatasan pemahaman terhadap relasi antar tabel dalam *database* ditanggulangi dengan membuat *Entity Relationship Diagram* (ERD) dari

beberapa dataset penting. ERD ini membantu memperjelas struktur dan alur data, serta memudahkan saat harus mengeksekusi *query* yang melibatkan banyak tabel.