BAB III

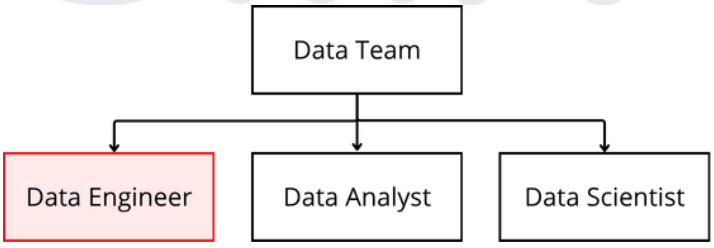
PELAKSANAAN KERJA

3.1 Kedudukan dan Koordinasi

3.1.1 Kedudukan

Di Ruparupa, Tim Data memegang peranan krusial sebagai pilar strategis di bawah naungan divisi *Data Analytics*. Misi utamanya adalah mengubah data mentah menjadi wawasan berharga yang mendorong efisiensi operasional dan memperkaya pengalaman pelanggan. Tanggung jawab tim ini mencakup *scope* yang luas, mulai dari membangun infrastruktur data yang kokoh dan mengembangkan model prediktif, hingga menyajikan analisis akurat untuk berbagai unit bisnis. Melalui pendekatan berbasis fakta ini, Tim Data memastikan setiap keputusan strategis memiliki landasan kuat, sehingga memungkinkan Ruparupa untuk terus berinovasi dan unggul dalam persaingan industri *e-commerce*.

Secara struktural, tim ini dipimpin oleh seorang *Data Lead* yang bertanggung jawab atas keseluruhan strategi dan manajemen data perusahaan. Tim ini terbagi menjadi tiga unit utama dengan spesialisasi berbeda: *Data Scientist, Data Engineer,* dan *Data Analyst*. Secara keseluruhan, tim yang solid ini beranggotakan 16 orang, dengan rincian struktur yang dapat dilihat pada Gambar 3.1.



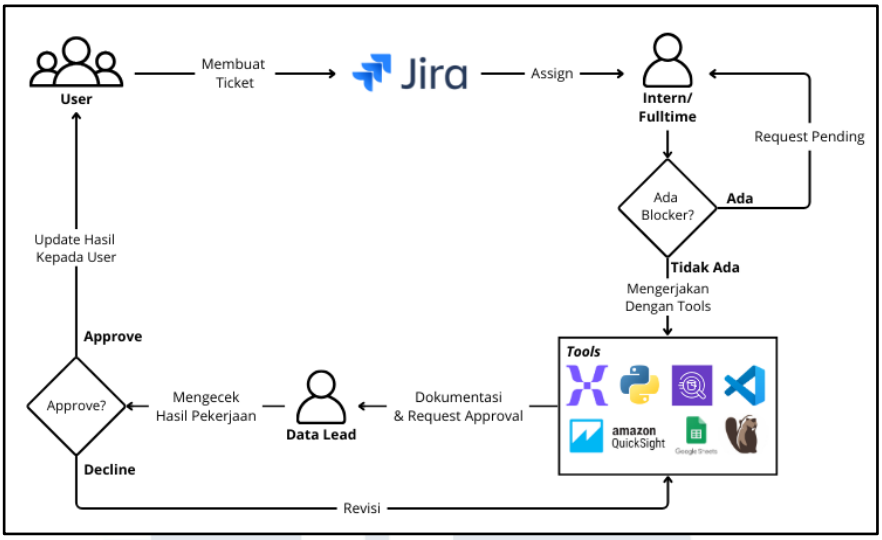
Gambar 3.1 Struktur Tim Data Ruparupa

Struktur tim ini dirancang sebagai sebuah ekosistem sinergis dengan tiga pilar fungsional yang bertujuan untuk mengubah aset data menjadi nilai bisnis yang nyata. Berperan sebagai fondasi, divisi *Data Engineer*, yang terdiri dari tiga karyawan tetap dan tiga *intern*, bertanggung jawab penuh atas

arsitektur dan aliran data. *Data Engineer* bertugas dalam merancang, membangun, dan memelihara *pipeline* data yang efisien, memastikan setiap informasi mengalir secara lancar, bersih, dan andal dari berbagai sumber hingga siap untuk diolah lebih lanjut oleh tim lain. Data yang telah disiapkan ini kemudian dimanfaatkan oleh tim *Data Scientist*, yang terdiri dari satu karyawan tetap dan satu *intern*, untuk melakukan eksplorasi mendalam dan melihat ke masa depan. Dengan menerapkan model statistik dan *machine learning* yang canggih, mereka menggali pola-pola tersembunyi dan membangun algoritma prediktif yang mampu menjawab tantangan bisnis paling kompleks. Sementara itu, sebagai penghubung antara data dan kebutuhan bisnis, divisi *Data Analyst*, yang merupakan unit terbesar dengan formasi tiga karyawan tetap dan empat *intern*, bertindak sebagai penerjemah utama data. *Data Analyst* mengubah angka dan tren yang kompleks menjadi narasi bisnis yang selaras melalui laporan serta *dashboard* interaktif, sehingga para *stakeholder* dapat dengan mudah memahami performa masa lalu dan mengidentifikasi peluang di masa depan.

3.1.2 Koordinasi

Semua pekerjaan di tim ini bersifat *on-demand*, yang berarti dimulai ketika ada *user* dari divisi lain yang membutuhkan data. *User* tersebut harus membuat tiket permintaan secara resmi melalui JIRA. JIRA adalah *software* manajemen proyek yang sangat penting bagi tim untuk mencatat, mengelola, dan melacak status semua tugas agar tidak ada yang terlewat. Setelah tiket dibuat, tiket itu akan otomatis masuk ke dalam antrean kerja tim. Tiket ini kemudian akan di-*assign* kepada *intern* atau karyawan tetap, biasanya disesuaikan dengan beban kerja dan tingkat kesulitan tugasnya. Proses pembagian tugas ini biasanya dilakukan saat *meeting huddle* harian pada pukul 09.30 WIB. *Huddle* adalah rapat koordinasi singkat tim Data melalui Google Meet. Tujuannya adalah untuk membahas *progress* pekerjaan kemarin, menyusun rencana kerja untuk hari ini, dan mencari solusi bersama jika ada anggota tim yang mengalami kesulitan atau hambatan. Gambar 3.2 menjelaskan alur kerja harian di Tim Data Ruparupa.



Gambar 3.2 Skema Kerja Tim Data Ruparupa

Setelah seorang anggota tim (*assignee*) mendapatkan alokasi tiket, langkah pertama yang wajib dilakukan adalah menghubungi *user* pembuat tiket. Komunikasi awal ini sangat penting untuk melakukan konfirmasi dan memastikan *assignee* mengerti dengan jelas apa yang diminta oleh *user*, sehingga dapat menghindari salah pengerjaan di kemudian hari. Jika saat proses pengerjaan ditemukan ada kendala atau blocker, misalnya data yang dibutuhkan belum tersedia, akses ke sistem tertentu belum diberikan, atau perlu menunggu data dari tim lain, a*ssignee* akan mengubah status tiket di JIRA menjadi "*Pending for Approva*l". Status ini berfungsi untuk menjeda perhitungan waktu *Service Level Agreement* (SLA) tiket tersebut. SLA adalah standar batas waktu pengerjaan tiket yang ditetapkan tim, yaitu 48 jam kerja. Perubahan status ini penting agar target waktu pengerjaan tim tetap adil dan wajar, serta *workflow* keseluruhan tidak terganggu oleh faktor eksternal.

Jika tidak ada kendala dan permintaan *user* sudah jelas, *assignee* bisa langsung mengerjakan tiket tersebut. Berbagai tools teknis digunakan sesuai dengan kebutuhan tugasnya untuk mengolah data. Contohnya, tim menggunakan Visual Studio Code (VS Code) untuk menulis *script* Python, dan DBeaver untuk menjalankan *query SQL* ke *database*. Untuk kebutuhan visualisasi data dan pembuatan *dashboard*, tim memakai Amazon QuickSight. Selain itu, AWS Athena juga sering digunakan untuk mengambil

data mentah langsung dari *data lake* perusahaan. Setelah pekerjaan teknisnya selesai, ada dua hal administrasi yang wajib dilakukan. *Assignee* harus memperbarui status tiket di JIRA dan membuat dokumentasi lengkap di Airtable. Dokumentasi ini sangat penting karena mencatat proses kerja, *query* yang digunakan, dan hasilnya, agar transparan dan bisa dilihat kembali oleh anggota tim lain jika diperlukan di masa depan.

Sesudah proses dokumentasi di Airtable selesai, status tiket di JIRA diubah menjadi "*Waiting for Approval*". Ini adalah tanda bahwa pekerjaan sudah selesai dari sisi assignee dan siap untuk diperiksa. Pada tahap ini, Data Lead atau anggota tim senior akan melakukan *review* atas hasil pekerjaan tersebut. Proses review ini adalah bentuk *quality control* untuk mengecek apakah hasilnya sudah benar, akurat secara data, dan yang terpenting, sudah sesuai dengan permintaan awal *user*. Jika ditemukan ada kesalahan atau kekurangan, tiket akan dikembalikan ke *assignee* untuk diperbaiki. Namun, jika hasil pekerjaan sudah sesuai standar dan tidak ada kesalahan, *Data Lead* akan memberikan *approval*. Setelah disetujui, tiket akan ditutup dan *user* pembuat tiket akan diberi tahu bahwa permintaannya sudah selesai dikerjakan.

Proses kerja ini akan terus berulang setiap kali ada permintaan data baru yang masuk dari *user*. Alur kerja yang teratur ini sangat membantu tim dalam berkoordinasi dan mengelola beban kerja harian. Adanya sistem tiket JIRA dan proses *review* ini memastikan setiap tiket dikerjakan secara efisien, progresnya transparan bagi semua anggota, dan dapat selesai sesuai target SLA yang ditentukan. Dengan *workflow* yang jelas dan kebiasaan dokumentasi yang rapi di Airtable, Tim Data Ruparupa dapat menjaga performa kerjanya tetap baik dan konsisten. Sistem kerja yang terstruktur ini sangat membantu menciptakan lingkungan kerja yang lebih produktif, akuntabel, dan kolaboratif bagi seluruh anggota tim.

3.2 Tugas yang Dilakukan

Pelaksanaan program magang mencakup berbagai tugas yang relevan dengan posisi Data Engineer Intern. Pekerjaan yang dilakukan tidak hanya bersifat operasional harian, tetapi juga meliputi beberapa proyek jangka panjang yang berkontribusi langsung pada operasional tim data. Seluruh kegiatan didokumentasikan untuk melacak kemajuan dan memastikan pemenuhan tanggung jawab. Tabel 3.1 di bawah ini menyajikan rincian lengkap dari kegiatan tersebut beserta periode waktu pelaksanaannya. Daftar tugas ini dikelompokkan ke dalam beberapa kategori utama, seperti *Data Cleaning, Integration & Analysis, Automation & Workflow Optimization,* serta *Cost Optimization & Efficiency*. Tabel ini memberikan gambaran umum yang jelas mengenai cakupan pekerjaan yang ditangani selama periode magang dari Juli hingga Oktober 2025.

Tabel 3.1 Rincian Kegiatan serta Waktu Pelaksanaan Magang

No	Deskripsi Kegiatan	Minggu ke-	Tanggal Mulai	Tanggal Selesai
Data Cleaning, Integration & Analysis				
1	Menggabungkan data dari berbagai sumber menjadi sebuah *dataset*.	1 - 20	1 Juli 2025	31 Oktober 2025
2	*Maintenance dataset* dengan menambah kolom, mengubah logika pemrosesan, dan memastikan akurasi data.	1 - 20	1 Juli 2025	31 Oktober 2025
Design, develop, and maintain scalable ETL/ELT pipelines				
3	Mengerjakan proyek pembuatan *pipeline* ETL untuk data penjualan dari *Seller Center* TikTok	1 - 4	1 Juli 2025	31 Juli 2025
4	Mengerjakan proyek pembuatan *pipeline* ELT untuk ingestasi data Freshdesk ke *Data Lake*.	5 - 8	1 Agustus 2025	31 Agustus 2025
Automation & Workflow Optimization				
5	Mengerjakan proyek implementasi AI untuk sistem *tagging* produk di *marketplace*.	9 - 12	1 September 2025	30 September 2025

No	Deskripsi Kegiatan	Minggu ke-	Tanggal Mulai	Tanggal Selesai
6	Mengerjakan proyek otomatisasi pembuatan laporan Data Order DC dari *Data Lake* ke Google Drive.	13 - 16	1 Oktober 2025	31 Oktober 2025
7	*Maintenance* Otomasi Proses Data Menggunakan Skrip Python	1 - 20	1 Juli 2025	31 Oktober 2025
Cost Optimization & Efficiency				
8	Mengerjakan proyek migrasi dataset QuickSight dari SPICE ke Direct Query Athena untuk optimasi biaya.	1 - 12	1 Juli 2025	30 September 2025

(Sumber olahan peneliti, 2025)

Seperti yang ditunjukkan pada tabel di atas, tanggung jawab pekerjaan mencakup lingkup yang luas dalam siklus pengelolaan data. Terdapat tugas-tugas yang bersifat berkelanjutan, seperti *maintenance dataset* dan pembaruan dokumentasi, yang dilakukan sepanjang periode magang. Di sisi lain, terdapat penugasan untuk mengerjakan proyek-proyek spesifik dengan tenggat waktu yang jelas. Contohnya adalah pembuatan *pipeline ETL* untuk data TikTok di bulan Juli dan proyek implementasi AI untuk *tagging* produk di bulan September. Variasi pekerjaan ini memberikan pengalaman praktis yang komprehensif. Penjelasan yang lebih mendalam dan rinci mengenai proses pelaksanaan beberapa proyek utama tersebut akan diuraikan pada bagian Uraian Pelaksanaan Kerja. Bagian tersebut akan menjabarkan langkah-langkah pengerjaan proyek dari awal hingga akhir.

3.3 Uraian Pelaksanaan Kerja

Data Engineer Intern di Ruparupa berfokus pada pengelolaan *life cycle* data dilaksanakan secara menyeluruh untuk menjamin ketersediaan informasi yang akurat bagi kebutuhan analisis bisnis. Proses *cleaning* dan integrasi data dari berbagai sumber yang terpecah dilakukan secara intensif hingga terbentuk *dataset* yang siap diolah. Selain itu, perancangan serta pemeliharaan *pipeline* data, baik dengan metode ETL (*Extract, Transform, Load*) maupun ELT (*Extract, Load, Transform*), dikerjakan secara rutin untuk memastikan aliran data ke *Data Warehouse* dan *Data Lake* berjalan tanpa hambatan. Pemeliharaan berkala terhadap *dataset* juga terus dilakukan, termasuk penyesuaian logika pemrosesan dan validasi

data, agar kualitas infrastruktur data tetap terjaga dan dapat diandalkan oleh tim operasional maupun analis.

Berbagai *software* digunakan untuk mendukung seluruh aktivitas teknis dan pengolahan data tersebut. Bahasa pemrograman Python dikelola menggunakan Visual Studio Code (VS Code) untuk keperluan skrip otomatisasi dan transformasi data yang kompleks. Akses dan manipulasi *database* dilakukan melalui aplikasi DBeaver dengan menjalankan perintah *query SQL*, sementara layanan Amazon QuickSight dan AWS Athena dimanfaatkan untuk kebutuhan visualisasi data serta pengambilan data mentah langsung dari *cloud*. Selain itu, *platform* manajemen proyek JIRA digunakan untuk pencatatan *ticket* atau tugas, sedangkan Airtable dimanfaatkan sebagai pusat dokumentasi, dan Google Sheets digunakan untuk pelaporan data operasional.

Di samping tugas operasional harian, berbagai proyek turut diselesaikan dengan fokus pada inovasi teknologi dan efisiensi sistem. Salah satu pencapaian utama adalah penerapan teknologi *Artificial Intelligence* untuk mengotomatisasi proses *tagging* produk di *marketplace*, sehingga akurasi pencarian produk meningkat signifikan. Upaya optimasi biaya infrastruktur *cloud* juga direalisasikan melalui migrasi penyimpanan data visualisasi dari memori SPICE ke metode *Direct Query* menggunakan AWS Athena, yang terbukti lebih hemat sumber daya. Tidak hanya itu, sistem pelaporan manual juga digantikan oleh skrip otomatisasi yang dirancang untuk mengirimkan laporan data operasional dari *Data Lake* langsung ke Google Drive secara terjadwal. Setiap tugas yang dikerjakan dicatat secara rinci melalui sistem tiket di JIRA agar beban kerja dan status penyelesaian dapat terpantau dengan jelas.

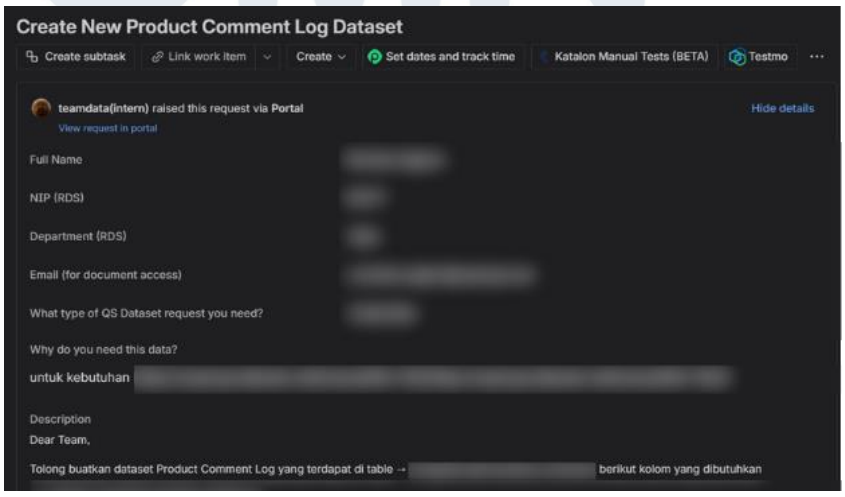
3.3.1 Proses Pelaksanaan

Selama periode magang, kegiatan kerja difokuskan pada pengelolaan *data life cycle* serta pengembangan infrastruktur data yang *scalable* untuk mendukung kebutuhan analisis bisnis dan operasional perusahaan. Terdapat beberapa proyek utama dan aktivitas *maintenance* yang menjadi fokus pelaksanaan, yaitu Pembuatan Pipeline ETL TikTok Shop yang berfungsi mengotomatisasi integrasi laporan penjualan manual dari Google Drive ke

Data Lake, Ingestasi Data Freshdesk yang bertujuan mengamankan data historis tiket layanan pelanggan melalui mekanisme ELT (*Extract, Load, Transform*), Implementasi AI Product Tagging yang memanfaatkan teknologi *Generative AI* Google Gemini untuk mempercepat proses pemberian kata kunci produk di *marketplace*, Otomasi Laporan Data Order DC yang digunakan untuk menyederhanakan distribusi data audit transaksi dari *Data Lake* langsung ke Google Drive pengguna, serta Optimasi Biaya Infrastruktur melalui migrasi *dataset* QuickSight dari memori SPICE ke metode *Direct Query* AWS Athena. Kelima proyek ini menjadi landasan penerapan *modern data engineering* dalam meningkatkan efisiensi operasional, menjamin akurasi data, serta menekan biaya infrastruktur *cloud* perusahaan secara signifikan

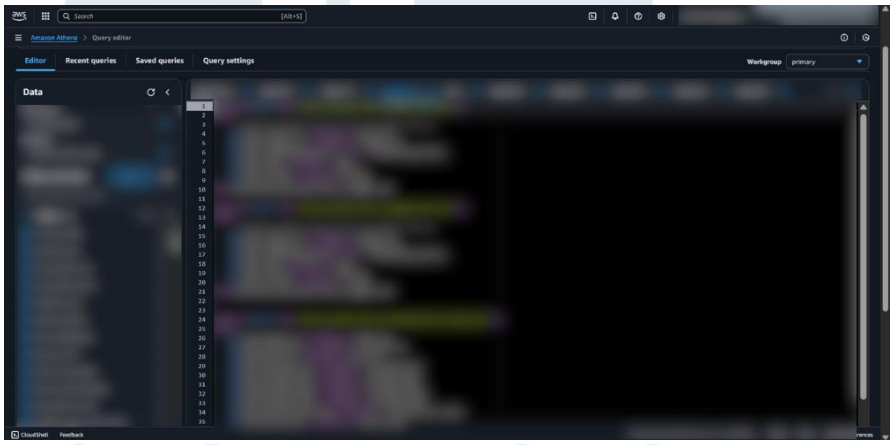
3.3.1.1 Menggabungkan Data dari Berbagai Sumber Menjadi Sebuah *Dataset*

Dataset merupakan kumpulan data terstruktur yang berfungsi sebagai sumber utama dalam proses analisis oleh tim data, khususnya *Data Analyst* dan *Data Scientist*. Dalam proses ini, *Data Engineer Intern* bertugas menggabungkan data dari berbagai sumber yang terpisah menjadi satu kesatuan dataset yang siap digunakan. Permintaan pembuatan *dataset* baru selalu diajukan melalui tiket pada platform JIRA, sebagaimana ditunjukkan pada Gambar 3.3.



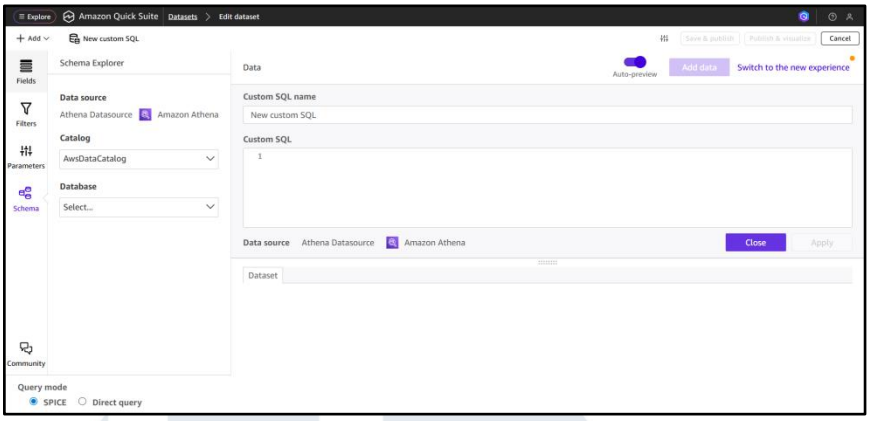
Gambar 3.3 Tampilan Tiket Permintaan Pembuatan *Dataset* Baru di JIRA

Setelah tiket diterima, logika pemrosesan serta kebutuhan tabel dan kolom dipelajari agar sesuai dengan permintaan bisnis. Diskusi teknis sering dilakukan bersama *user* untuk memastikan pemahaman yang menyeluruh dan mencegah kesalahan interpretasi. Setelah kebutuhan dipahami, *query* disusun menggunakan bahasa SQL Presto pada layanan AWS Athena, seperti yang terlihat pada Gambar 3.4. Data yang dihasilkan dari *query* tersebut wajib divalidasi, baik secara mandiri maupun bersama *supervisor*, untuk menjamin akurasi angka dan informasinya.

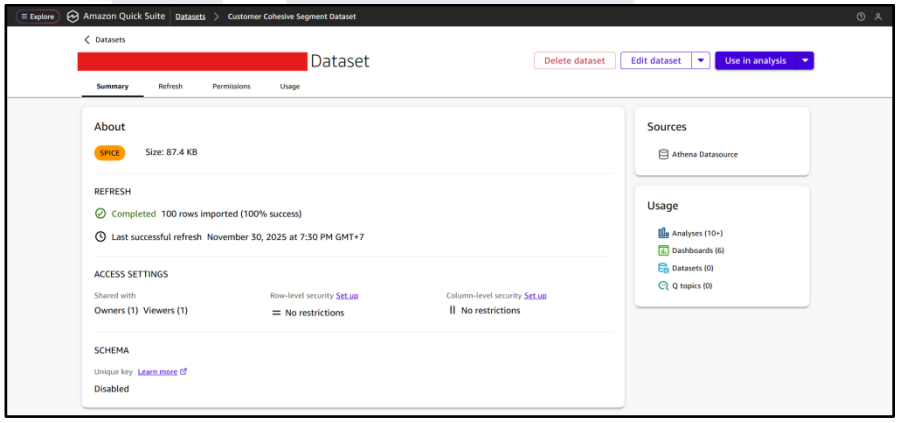


Gambar 3.4 Proses Penyusunan Query SQL pada AWS Athena

Langkah selanjutnya adalah pembuatan dataset di AWS QuickSight agar dapat digunakan untuk visualisasi. Proses ini dimulai dengan memilih opsi "*New Dataset*" dan menetapkan AWS Athena sebagai sumber data. *Query* yang telah disusun di Athena kemudian disalin ke editor QuickSight, seperti yang ditampilkan pada Gambar 3.5. Mode penyimpanan diubah menjadi SPICE (*Super-fast, Parallel, In-memory Calculation Engine*) guna mempercepat performa pemrosesan data. *Dataset* kemudian disimpan dan dipublikasikan melalui tombol "*Save and Publish*". Gambar 3.6 memperlihatkan tampilan saat *dataset* berhasil dibuat.

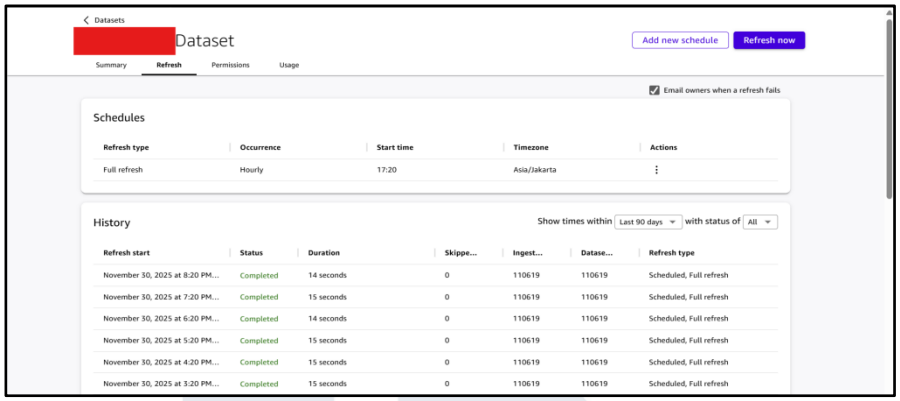


Gambar 3.5 Tampilan Editor *Custom SQL* pada Amazon QuickSight

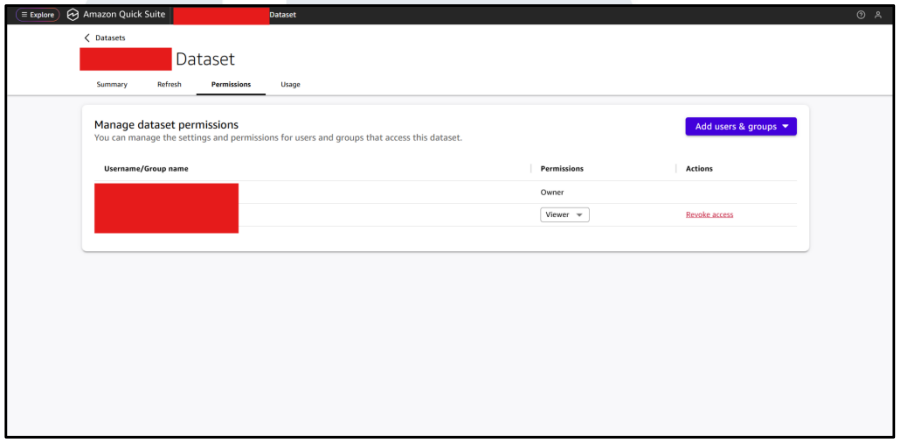


Gambar 3.6 Tampilan *Dataset* yang Telah Berhasil Dibuat

Agar data selalu terbaru, jadwal pembaruan otomatis (*schedule refresh*) ditambahkan seperti pada Gambar 3.7. Penentuan metode pembaruan, baik *full refresh* maupun *incremental refresh*, didiskusikan terlebih dahulu dengan *supervisor* berdasarkan ukuran data dan kebutuhan frekuensi pembaruan. Setelah disetujui, pengaturan izin akses (*permissions*) diterapkan pada akun *user* terkait, seperti yang ditunjukkan pada Gambar 3.8, untuk menjaga keamanan dan privasi data.

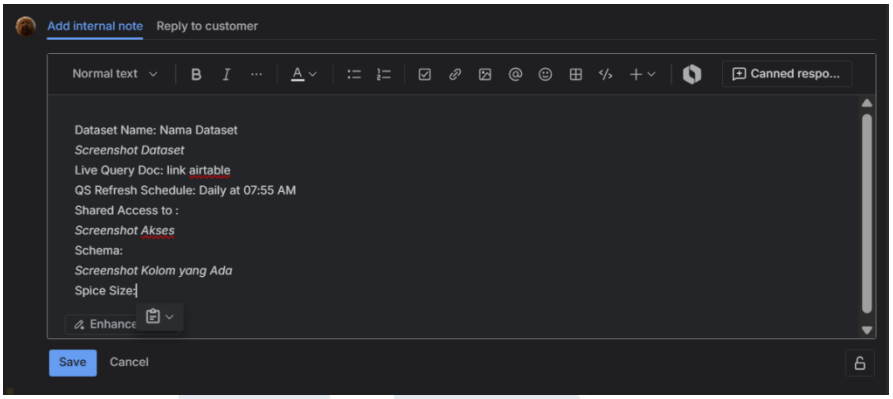


Gambar 3.7 Konfigurasi Jadwal Pembaruan (*Schedule Refresh*) Dataset



Gambar 3.8 Pengaturan Izin Akses (*Permissions*) Dataset untuk *User*

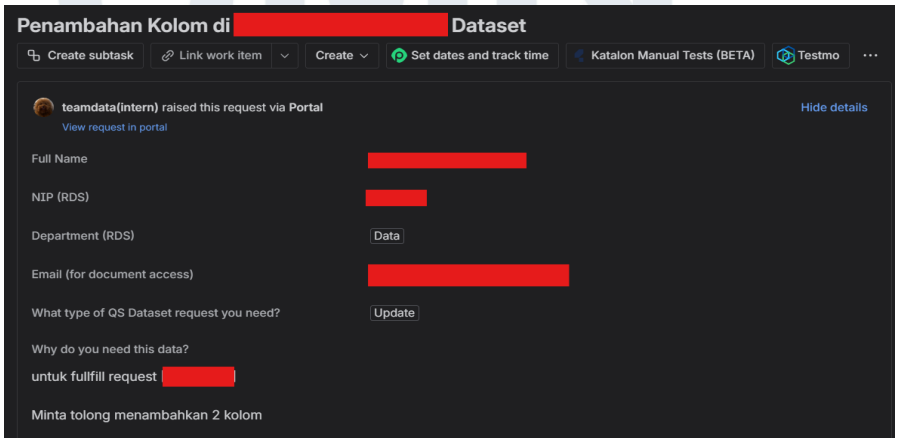
Tahap akhir dari seluruh rangkaian ini adalah dokumentasi teknis. Query SQL, logika yang digunakan, serta atribut penting lainnya dicatat pada kolom komentar tiket JIRA, seperti terlihat pada Gambar 3.9. Dokumentasi ini bertujuan memudahkan proses pemeriksaan oleh *supervisor* serta pelacakan kembali di masa depan jika terjadi kendala. Seluruh proses ini mencerminkan peran *Data Engineer Intern* dalam menjembatani kebutuhan teknis dan bisnis melalui penyediaan data yang terintegrasi dan siap pakai. Pembuatan *dataset* ini dijalankan secara berkelanjutan setiap kali terdapat permintaan tiket baru dari *user*, serta dilaksanakan secara konsisten mulai dari awal hingga akhir periode magang berlangsung.



Gambar 3.9 Dokumentasi Teknis Pengerjaan *Dataset* pada Komentar JIRA

3.3.1.2 *Maintenance Dataset* dengan Menambah Kolom, Mengubah Logika Pemrosesan, dan Memastikan Akurasi Data

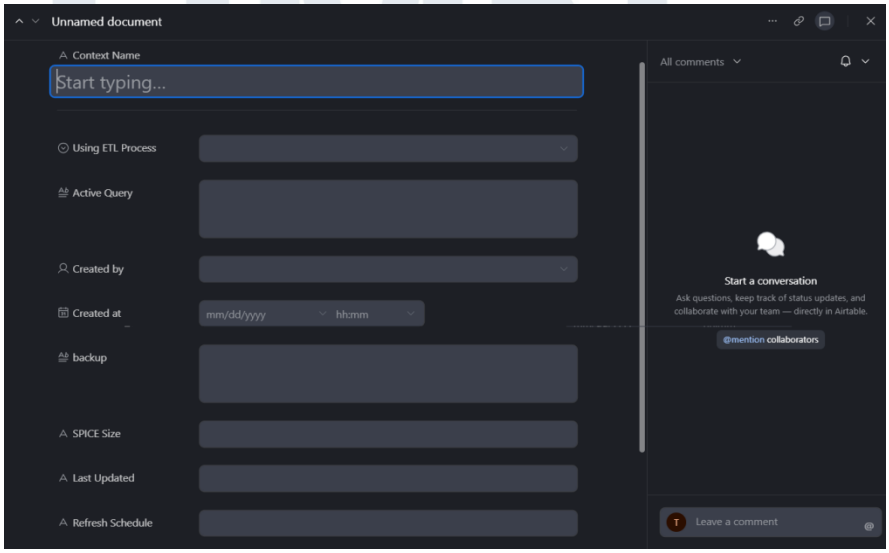
Tugas pengelolaan data tidak hanya terbatas pada pembuatan *dataset* baru, melainkan mencakup *maintenance* rutin terhadap *dataset* yang sudah berjalan secara operasional. *Request* pemeliharaan ini sering kali diajukan oleh *user* bisnis untuk menyesuaikan struktur data dengan kebutuhan analisis yang terus berkembang. Lingkup pekerjaan ini meliputi penambahan kolom baru, penghapusan atribut yang tidak lagi relevan, penggantian *logic* pemrosesan, hingga perubahan tingkat *granularity* dari data tersebut. Contoh *ticket* permintaan perubahan struktur data dari *user* dapat dilihat pada Gambar 3.10.



Gambar 3.10 Tampilan *Ticket* JIRA Terkait Permintaan Penambahan Kolom *Dataset*

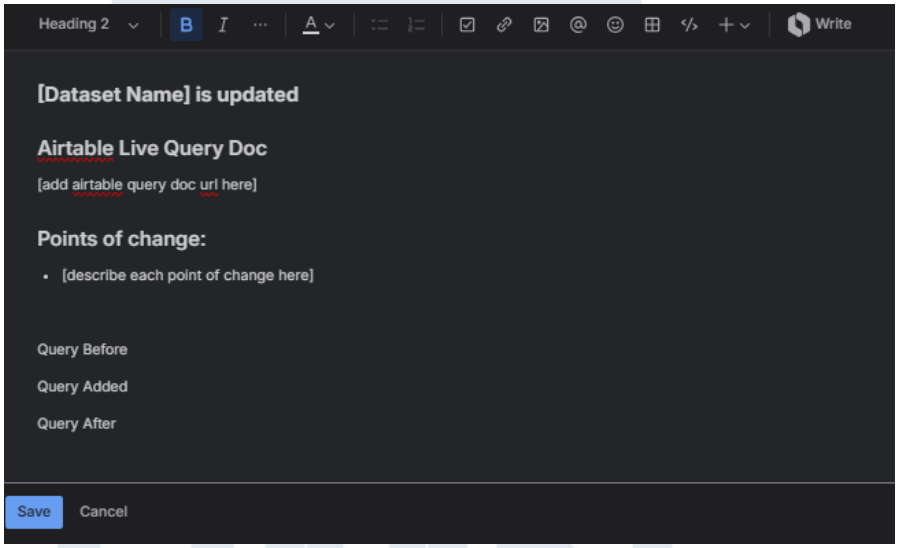
Setelah *ticket* diterima, analisis awal dilakukan terhadap *logic* baru yang diminta guna menyamakan pemahaman teknis dengan *objective user*. Diskusi kerap dilakukan sebelum mengerjakan *ticket* untuk mencegah terjadinya kesalahan atau *miscommunication*. Proses pengubahan *query* kemudian dilakukan melalui editor AWS Athena seperti pada Gambar 3.4, yang memungkinkan pengujian *logic* baru secara lebih fleksibel sebelum diterapkan ke AWS Quicksight. Hasil modifikasi data tersebut wajib divalidasi, baik dilakukan secara mandiri maupun melalui pengecekan bersama *supervisor*, untuk memastikan akurasi data tetap terjaga.

Apabila hasil validasi telah dikonfirmasi aman, *query* yang telah diperbarui segera di-*update* ke dalam *dataset* di AWS QuickSight. Langkah selanjutnya adalah melakukan pembaruan dokumentasi teknis pada *platform* Airtable, seperti yang ditunjukkan pada Gambar 3.11. Mekanisme dokumentasi ini dilakukan dengan memindahkan *query* versi lama ke kolom *backup query* dan menempatkan *query* baru pada kolom *active query*. Penyimpanan versi kode ini bertujuan agar sistem memiliki riwayat perubahan yang jelas dan data lama dapat di-*restore* jika sewaktu-waktu diperlukan.



Gambar 3.11 Dokumentasi *Update Query* SQL pada *Platform* Airtable

Selain di Airtable, perubahan juga didokumentasikan secara *detail* pada *comment* ticket JIRA sebagai bentuk transparansi pekerjaan. Dokumentasi ini mencakup perbandingan antara skrip *query before, query added,* dan *query after*, serta poin-poin penjelasan mengenai bagian mana saja yang telah dimodifikasi. Contoh format catatan perubahan teknis atau *changelog* ini ditampilkan pada Gambar 3.12. Pencatatan yang rapi ini bertujuan untuk memudahkan penelusuran masalah di masa depan serta membantu seluruh tim memahami riwayat evolusi dari sebuah dataset.

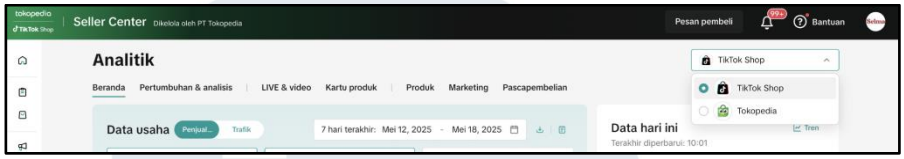


Gambar 3.12 Catatan Perubahan *Logic Query* pada *Comment* JIRA

3.3.1.3 Proyek Pembuatan *Pipeline* ETL untuk Data Penjualan Dari Seller Center Tiktok

Unit bisnis Kawan Lama Group terus memperluas jangkauan pasarnya melalui strategi *omnichannel*, salah satunya dengan memanfaatkan *platform* TikTok Shop yang kini telah terintegrasi dengan Tokopedia. Tim *Merchandising* memiliki kebutuhan mendesak untuk memantau performa penjualan dari Seller Center TikTok secara berkala guna menentukan strategi stok dan promosi yang tepat. Namun, proses pengumpulan data ini sebelumnya dilakukan secara manual dengan mengunduh laporan satu per satu setiap bulannya, yang tidak hanya memakan waktu tetapi juga rentan

terhadap kesalahan manusia atau *human error*. Selain itu, tantangan teknis muncul karena satu toko di *Seller Center* kini dapat memiliki dua *channel* penjualan berbeda, yaitu TikTok dan Tokopedia, yang harus dipisahkan secara logika di *backend* data. Kompleksitas sumber data yang ditampilkan pada antarmuka Seller Center dapat dilihat pada Gambar 3.13.



Gambar 3.13 Tampilan Antarmuka *Seller Center* TikTok Shop

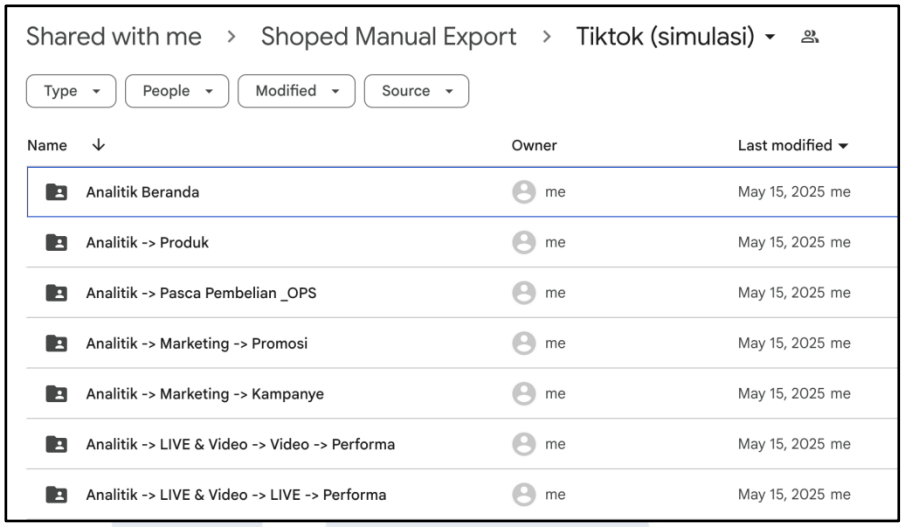
Dalam *project* ini, pengembangan difokuskan pada perancangan dan pembangunan *pipeline* ETL (*Extract, Transform, Load*) untuk mengotomatisasi aliran data tersebut. Sistem ini dibangun untuk menjembatani proses manual yang dilakukan oleh tim bisnis dengan sistem *Data Lake* perusahaan melalui mekanisme *ingestion* berbasis *cloud storage*. Lingkup pengerjaan meliputi perancangan struktur direktori penyimpanan, pembuatan Standar Operasional Prosedur (SOP) penamaan *file* agar dapat dibaca oleh sistem, serta penulisan skrip Python untuk memproses data mentah menjadi *dataset* yang terstruktur. Tujuannya adalah memastikan bahwa data yang masuk ke dalam sistem analitik memiliki format yang konsisten dan dapat langsung digunakan (*ready-to-use*) oleh tim *Data Analyst* tanpa perlu pembersihan ulang.

Tahap pertama dalam eksekusi proyek ini adalah penetapan alur kerja standar bagi *user* dalam melakukan penyetoran data. *Flowchart* disusun untuk mewajibkan pengguna mengubah nama file hasil unduhan sesuai dengan format baku SalesSource_StoreName_ReportType_StartDate-EndDate.xlsx sebelum mengunggahnya. Standarisasi ini menjadi aspek krusial karena skrip ekstraksi data akan membaca metadata seperti nama toko, jenis laporan, dan periode waktu langsung dari nama file

tersebut untuk keperluan partisi data. Jika format penamaan tidak sesuai, sistem dirancang untuk menolak file tersebut guna menjaga integritas data di sistem hilir. Alur kerja pengunggahan data yang telah dirancang ditunjukkan secara rinci pada Gambar 3.14.

Gambar 3.14 Alur Kerja Standardisasi dan Upload Data ke Google Drive

Setelah *file* diunggah ke dalam folder Google Drive yang telah disiapkan, seperti yang terlihat pada Gambar 3.15, skrip otomatisasi akan dijalankan untuk memproses pemetaan *channel* penjualan. Logika kondisional khusus diimplementasikan untuk menangani isu dualitas *channel* pada TikTok Shop, di mana toko yang bermigrasi dari Tokopedia (seperti STORA Storage) akan dicatat sebagai *channel* "Tokopedia", sedangkan toko asli TikTok (seperti Selma ID) akan dicatat sebagai *channel* "Tiktok". Logika pemetaan ini, yang ditampilkan pada Gambar 3.16, sangat penting untuk memastikan pendapatan tercatat di buku besar unit bisnis yang tepat. Data yang telah melalui proses transformasi dan validasi ini kemudian disimpan sebagai *dataset* baru dengan nama "Gdrive Tiktok Dataset".

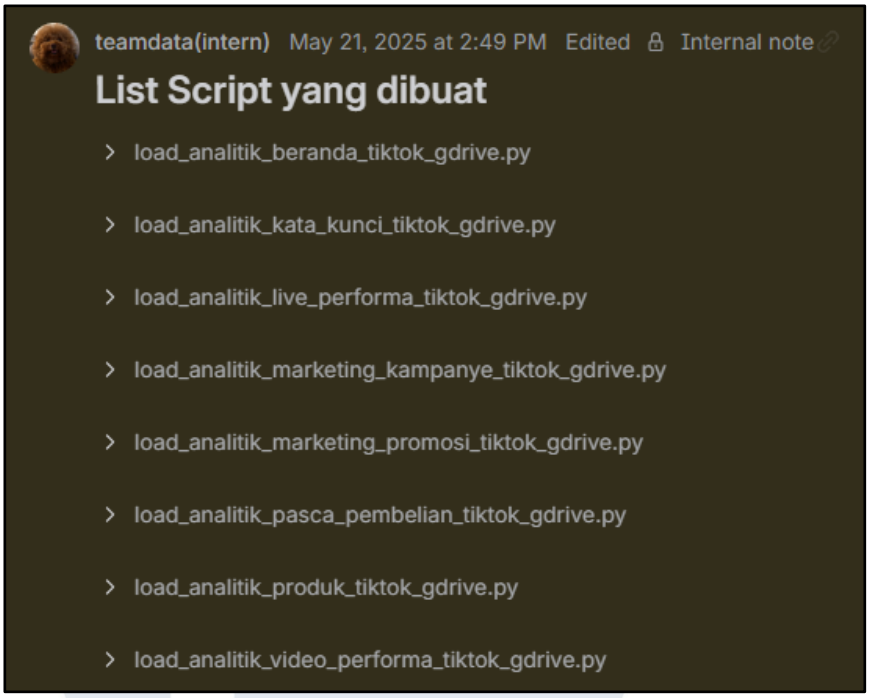


Gambar 3.15 Struktur Direktori Penyimpanan Data pada Google Drive

msv_shop_name	msv_vendor	channel
STORA Storage	tiktok-tokopedia	Tokopedia
Selma ID	tiktok	Tiktok

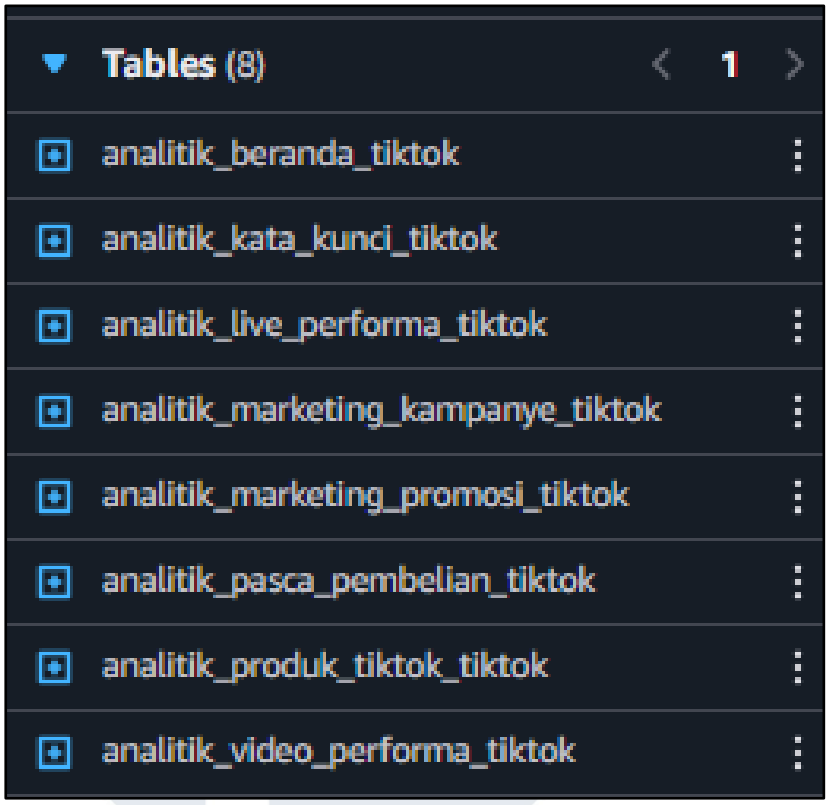
Gambar 3.16 Logika Pemetaan *Channel* Penjualan pada Dataset

Implementasi teknis dari *pipeline* ini dibangun menggunakan bahasa pemrograman Python yang memanfaatkan layanan *cloud computing* AWS. Serangkaian skrip Python dikembangkan secara modular untuk menangani berbagai jenis laporan secara spesifik, sebagaimana ditampilkan pada Gambar 3.17. Skrip-skrip ini berfungsi untuk mengekstraksi data dari Google Drive, melakukan transformasi data sesuai logika bisnis yang telah ditetapkan, dan memuat (*load*) hasilnya ke dalam AWS S3 sebagai lapisan penyimpanan data (*Data Lake*). Penggunaan skrip modular memungkinkan proses pemeliharaan (*maintenance*) menjadi lebih mudah jika terjadi perubahan format pada salah satu jenis laporan di masa mendatang.



Gambar 3.17 Daftar Skrip Python untuk Pemrosesan Data

Setelah data berhasil dimuat ke dalam AWS S3, tahap akhir melibatkan penggunaan layanan AWS Athena untuk memodelkan data tersebut agar dapat diakses menggunakan kueri SQL standar. Data yang tersimpan di S3 diregistrasikan sebagai tabel eksternal di Athena, sehingga memungkinkan tim analis untuk mengakses data yang sudah bersih dan terstruktur secara langsung. Hasil akhir dari proses ETL ini berupa sekumpulan tabel siap pakai, seperti analitik_beranda_tiktok dan analitik_produk_tiktok, yang terlihat pada antarmuka Athena di Gambar 3.18. Dengan tersedianya tabel-tabel ini, ketergantungan pada proses unduh manual berhasil dihilangkan, dan akurasi data untuk pengambilan keputusan bisnis dapat lebih terjamin.

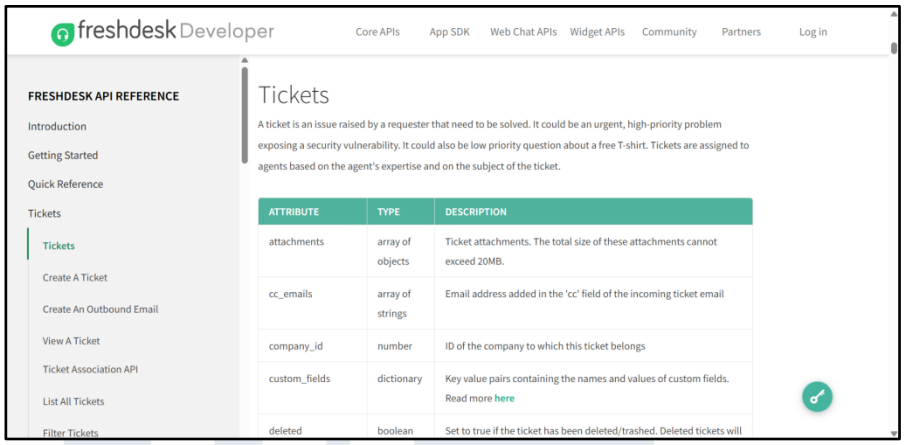


Gambar 3.18 Tampilan Tabel Output pada AWS Athena

3.3.1.4 Proyek pembuatan *pipeline* ELT untuk ingestasi data Freshdesk ke *Data Lake*

Freshdesk merupakan *platform customer service* yang digunakan Ruparupa untuk mengelola interaksi dengan konsumen, termasuk penanganan tiket keluhan dan pertanyaan melalui email. Seiring dengan peningkatan status layanan Freshdesk ke versi berbayar, tim operasional menghadapi kebutuhan mendesak untuk mengamankan data historis interaksi agar pemantauan metrik krusial seperti *Customer Satisfaction Score* (CSAT) tetap dapat dilakukan secara berkesinambungan. Tanpa mekanisme penyimpanan yang tepat, risiko kehilangan akses terhadap data historis tiket, performa agen, dan survei kepuasan pelanggan menjadi sangat tinggi. Oleh karena itu, diperlukan solusi *backend* yang mampu menarik data dari Freshdesk secara otomatis dan memusatkannya ke dalam infrastruktur data perusahaan. Dokumentasi teknis API Freshdesk yang menjadi

acuan utama dalam pengembangan integrasi ini ditampilkan pada Gambar 3.19.



Gambar 3.19 Dokumentasi API *Developer* Freshdesk

Dalam proyek ini, pengembangan difokuskan pada perancangan *pipeline* ELT (*Extract, Load, Transform*) yang menghubungkan sistem Freshdesk dengan *Data Lake* perusahaan. Berbeda dengan metode tradisional, pendekatan ELT memprioritaskan pemuatan data mentah secara langsung ke dalam penyimpanan target sebelum dilakukan proses transformasi atau pembersihan. Metode ini dipilih untuk menjaga integritas data orisinal di dalam *Data Lake*, sehingga memberikan fleksibilitas tinggi bagi tim data untuk mengubah logika transformasi di masa mendatang tanpa perlu mengulang proses ekstraksi dari sumber awal. Selain itu, ELT juga meningkatkan efisiensi waktu pemrosesan, yang sangat krusial dalam menangani batasan *rate limit* API pihak ketiga. Tugas yang dilakukan mencakup identifikasi *endpoint* API yang relevan, pengembangan logika ekstraksi data, serta manajemen penyimpanan data dalam format yang efisien. *Objective* utama dari proyek ini adalah memindahkan sembilan jenis laporan utama ke dalam AWS S3, untuk memastikan ketersediaan data operasional bagi tim *Data Analyst* tanpa membebani kinerja sistem produksi Freshdesk.

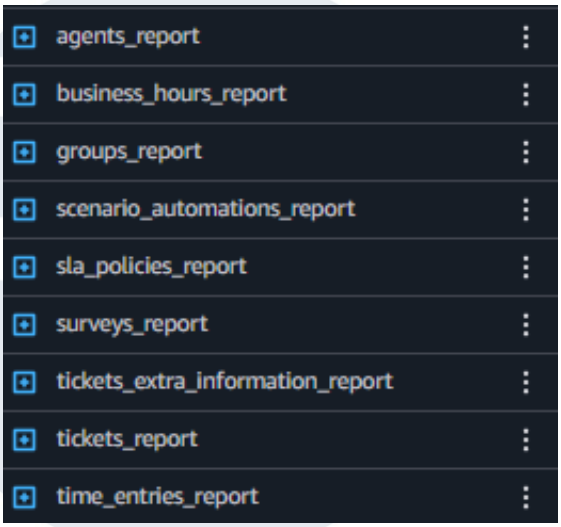
Setelah mengidentifikasi *endpoint* API, dilakukan dengan mengembangkan serangkaian skrip Python modular, seperti

freshdesk_extract_ticket.py dan freshdesk_extract_agent.py, yang masing-masing bertugas menangani *endpoint* spesifik. Skrip ini dirancang dengan mekanisme penanganan kesalahan (*error handling*) yang kuat, termasuk pengelolaan *rate limit* API dan sistem *pagination* untuk memastikan seluruh data terambil tanpa terputus. Selain itu, proses transformasi data dilakukan secara langsung, meliputi konversi zona waktu dari UTC ke WIB (UTC+7), normalisasi kolom JSON yang bertingkat (*flattening*), serta standardisasi tipe data menjadi string untuk menjaga konsistensi. Seluruh data yang telah diproses kemudian disimpan dalam format Parquet di AWS S3.

Ekstraksi data mencakup sembilan jenis *report* yang merepresentasikan dimensi operasional layanan pelanggan secara menyeluruh. Laporan utama yang ditarik meliputi tickets_report dan tickets_extra_information_report yang berfungsi sebagai rekam jejak historis interaksi tiket, serta time_entries_report yang mencatat rincian durasi penyelesaian masalah untuk setiap tiket. Untuk analisis kinerja sumber daya manusia, tabel agents_report dan groups_report digunakan untuk memetakan profil agen dan struktur tim, sementara business_hours_report memberikan konteks waktu operasional yang valid. Aspek krusial terkait kepuasan pelanggan dan kepatuhan standar layanan dipantau melalui surveys_report yang menjadi basis data utama perhitungan *Customer Satisfaction Score* (CSAT), serta sla_policies_report yang menampung parameter aturan tingkat layanan (*Service Level Agreement*). Selain itu, scenario_automations_report turut diintegrasikan ke dalam Data Lake guna memungkinkan evaluasi terhadap efektivitas skenario otomatisasi alur kerja yang telah diterapkan dalam sistem.

Tahap akhir adalah penyediaan akses data melalui layanan AWS Athena, yang memungkinkan data di S3 dikueri menggunakan sintaks SQL standar. Tabel-tabel eksternal didefinisikan di Athena untuk merepresentasikan setiap jenis laporan yang telah diekstraksi, seperti

tickets_report, agents_report, dan surveys_report, sebagaimana terlihat pada Gambar 3.20. Untuk menjaga kebaruan data, seluruh proses eksekusi skrip dijadwalkan secara otomatis setiap hari. Dengan tersedianya tabel-tabel ini di *Data Lake*, tim operasional kini dapat melakukan analisis historis dan perhitungan CSAT secara mandiri dan akurat, terlepas dari perubahan status lisensi pada *platform* sumber.



Gambar 3.20 Daftar Tabel Freshdesk pada AWS Athena

3.3.1.5 Proyek Implementasi AI untuk Sistem *Tagging* Produk di *Marketplace*

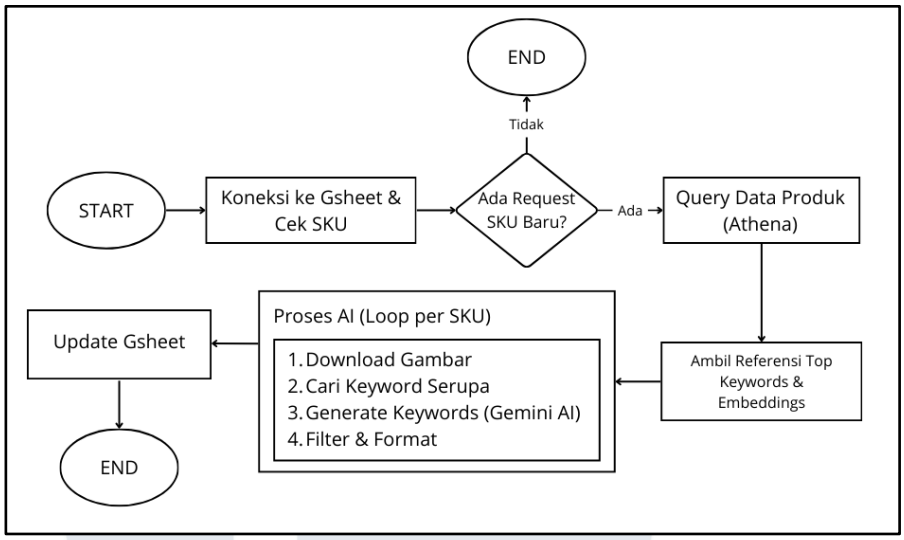
Dalam ekosistem *marketplace* yang sangat kompetitif, visibilitas produk pada *search engine* menjadi faktor penentu utama keberhasilan penjualan. Tim bisnis menghadapi tantangan besar dalam mengoptimalkan *Search Engine Optimization* (SEO) untuk ribuan SKU produk yang terus bertambah setiap harinya, di mana setiap produk memerlukan *tag* atau kata kunci yang spesifik agar mudah ditemukan oleh pembeli. Proses pemberian kata kunci ini sebelumnya dilakukan secara manual oleh tim *Merchandising*, yang memakan waktu rata-rata sekitar satu menit untuk setiap SKU guna meriset dan mengetik kata kunci yang relevan. Selain inefisiensi waktu, metode manual ini sering kali menghasilkan inkonsistensi data dan rentan terhadap kesalahan manusia, seperti penggunaan kata-kata

promosi (contoh: "diskon", "termurah") yang dilarang oleh algoritma *marketplace*. Tanpa adanya sistem otomatisasi yang cerdas, potensi produk untuk muncul di halaman pencarian utama menjadi terbatas, yang berdampak langsung pada rendahnya tingkat konversi penjualan (*Conversion Rate* atau CVR).

Dalam proyek ini, *objective* utama berfokus pada pengembangan solusi otomatisasi berbasis kecerdasan buatan (*Artificial Intelligence*) untuk mempercepat dan menstandarisasi proses *tagging* produk. Tugas yang diberikan mencakup perancangan arsitektur *pipeline* data yang menjembatani basis data perusahaan di AWS Athena dengan kemampuan analisis visual dari layanan *Generative AI* Google Gemini. Pengembangan dilakukan dengan mengembangkan skrip Python yang mampu memproses data gambar dan teks secara massal, serta mengintegrasikan hasilnya kembali ke dalam lembar kerja operasional Google Spreadsheet yang digunakan oleh tim bisnis. Inisiatif ini bertujuan untuk mengubah proses manual yang repetitif menjadi alur kerja otomatis yang akurat dan terukur. Penerapan teknologi ini diharapkan dapat memberikan efisiensi operasional yang signifikan sekaligus meningkatkan standar kualitas data produk di seluruh *channel* penjualan.

Mekanisme kerja sistem ini dimulai dengan proses identifikasi SKU produk yang belum memiliki kelengkapan atribut kata kunci pada Google Spreadsheet yang berfungsi sebagai antarmuka pengguna. Skrip Python dirancang untuk membaca daftar SKU tersebut dan kemudian melakukan penarikan informasi detail produk, termasuk nama, kategori, dan tautan gambar (*image URL*), dari *Data Lake* perusahaan menggunakan kueri SQL melalui layanan AWS Athena. Data gambar visual menjadi komponen krusial dalam proses ini karena analisis visual memberikan konteks fungsi produk yang lebih kaya dibandingkan hanya mengandalkan teks nama produk semata. Alur logika pemrosesan data dari awal hingga akhir, mulai

dari koneksi data hingga pembaruan hasil, digambarkan secara sistematis pada *flowchart* di Gambar 3.21.



Gambar 3.21 *Flow Tagging* Produk Berbasis AI

Fokus utama dalam proyek ini adalah pemanfaatan model *Large Language Model* (LLM) Google Gemini untuk menghasilkan kata kunci yang relevan berdasarkan analisis gabungan antara gambar produk dan kategori. Sebuah *prompt* yang kompleks dikembangkan untuk mengarahkan AI agar berfokus pada intensi pencarian pembeli (*search intent*) dan fungsi utama produk, serta menghindari penggunaan kata sifat yang subjektif. Sistem ini juga mengintegrasikan data referensi berupa "*Top Search Keywords*" dari historis pencarian pengguna untuk meningkatkan relevansi hasil prediksi. Selain itu, diterapkan mekanisme filter otomatis yang secara agresif memvalidasi keluaran AI guna menghapus *forbidden words* seperti kata promosi, nama *brand* pesaing, atau kata yang terlalu umum sebagaimana tercantum pada Gambar 3.22.

Kata-kata yang terlalu umum	Kata-kata yang perlu dilarang	Kata-kata brand	Kata-kata berulang
Furnitur	Kompak	Fischer	Brand
Elektronik	Merangsang	Jenga	Color
Peralatan Dapur	Ribet	Rubik, rubiks	Dimensi produk
Perlengkapan Dapur	Pistol	Velcro	Kapasitas
Gadget	eek	Scotch Tape	Daya
Otomotif		Sellotape	Material
Rumah Tangga		Post-it	
Home Improvement		Tupperware	
Hobi & Gaya Hidup		Ziploc	
Perawatan & Kecantikan		Teflon	
Aksesoris Fashion		Thermos	
Alat Pengaman		Instant Pot	
		Bubble Wrap	
		Styrofoam	
		Pampers	
		Betadine	

Gambar 3.22 Aturan *Filter* Kata Kunci dan Daftar Kata Terlarang

Setelah proses generasi dan validasi kata kunci selesai, hasil akhirnya diformat menjadi rangkaian *tag* standar yang digabungkan dengan nama produk. Data yang telah matang ini kemudian ditulis kembali secara otomatis ke kolom yang sesuai di Google Spreadsheet menggunakan API *gspread*, sehingga tim operasional dapat langsung memanfaatkannya. Dampak dari implementasi sistem ini sangat signifikan terhadap efisiensi kerja tim *Merchandising*; durasi pengerjaan *tagging* berhasil dipangkas drastis dari satu menit per SKU secara manual menjadi hanya satu detik per SKU menggunakan otomatisasi. Selain efisiensi waktu, kualitas kata kunci yang dihasilkan AI yang lebih relevan dan sesuai dengan intensi pembeli terbukti berkontribusi positif terhadap peningkatan *Conversion Rate* (CVR) dari hasil pencarian SKU di *marketplace*. Contoh hasil implementasi penamaan produk akhir dapat dilihat pada Gambar 3.23.

Gambar 3.23 Hasil *Tagging* Produk di *Marketplace*

3.3.1.6 Proyek Otomatisasi Pembuatan Laporan Data *Order* dari *Data Lake* ke Google Drive

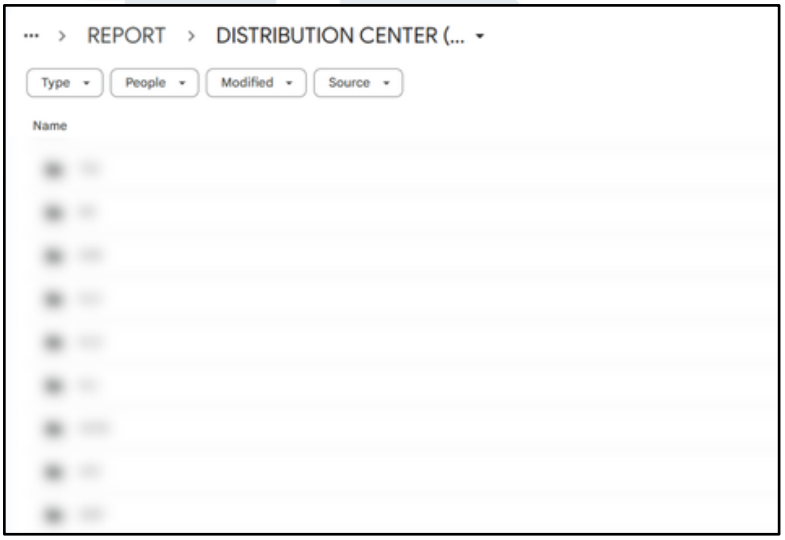
Tim *Internal Audit* memiliki kebutuhan krusial untuk memperoleh visibilitas menyeluruh terhadap siklus transaksi di

Distribution Center (DC) guna keperluan audit. Data yang diminta mencakup rekam jejak transaksi secara *end-to-end*, mulai dari pesanan masuk (*order creation*) hingga proses penyelesaian akhir, termasuk jika terjadi retur atau pengembalian dana (*refund/return*). Data ini diminta untuk disajikan dalam satu tabel terintegrasi agar memudahkan proses penelusuran. Sebelumnya, penyediaan data semacam ini sering kali terkendala oleh fragmentasi sumber data antar sistem yang berbeda dan proses konsolidasi manual yang memakan waktu. Oleh karena itu, diperlukan solusi otomatis yang dapat menyuplai data mentah secara rutin setiap bulan melalui *platform* penyimpanan awan yang mudah diakses dan aman, yaitu Google Drive.

Proyek ini berfokus pada pengembangan mekanisme otomatisasi pelaporan yang menghubungkan infrastruktur *Data Lake* perusahaan dengan kebutuhan aksesibilitas pengguna. Tugas utama meliputi perancangan alur kerja data (*workflow*), penulisan skrip ekstraksi data menggunakan Python, serta integrasi dengan API Google Drive untuk manajemen *file*. *Objective* dari proyek ini adalah menghilangkan intervensi manual dalam penyiapan laporan rutin, memastikan data selalu tersedia tepat waktu setiap tanggal 2 di bulan berjalan, dan terorganisir dengan rapi berdasarkan *business unit*. Melalui pendekatan ini, aliran data audit dijamin konsistensi dan kelengkapannya, memungkinkan tim audit untuk fokus pada analisis substansial daripada menghabiskan waktu untuk pengumpulan dan pembersihan data.

Tahap awal adalah pengembangan skrip Python yang memiliki logika pemrosesan berjenjang. Alur kerja skrip dimulai dengan fungsi pembersihan otomatis yang memindai folder tujuan di Google Drive dan memindahkan laporan versi bulan lalu ke *folder trash* untuk mencegah duplikasi atau kebingungan versi data. Selanjutnya, skrip menginisialisasi koneksi ke layanan AWS Athena untuk

mengeksekusi kueri SQL kompleks yang menggabungkan berbagai tabel transaksional menjadi satu *dataset* utuh. Data hasil kueri ini kemudian diproses menggunakan *library* Pandas untuk dipartisi berdasarkan Unit Bisnis (BU) dan periode waktu, lalu dikonversi menjadi fail berformat Excel (.xlsx). *File-file* tersebut akhirnya diunggah secara otomatis ke dalam struktur folder spesifik untuk setiap unit bisnis sebagaimana ditampilkan pada Gambar 3.24.



Gambar 3.24 Struktur *Folder* Laporan Data *Order* DC pada Google Drive

3.3.1.7 *Maintenance* Otomasi Proses Data Menggunakan Skrip Python

Dalam upaya mendukung efisiensi operasional perusahaan, kebutuhan akan pemrosesan data yang cepat dan akurat menjadi sangat krusial. Proses manual dalam penarikan dan pengolahan data seringkali dinilai tidak efisien, memakan waktu lama, dan memiliki risiko tinggi terhadap *human error*. Oleh karena itu, strategi otomasi diterapkan secara luas untuk meminimalkan ketergantungan pada eksekusi manual serta mempercepat distribusi informasi kepada pengguna bisnis. Implementasi ini mencakup berbagai jenis alur kerja, mulai dari eksekusi kueri SQL rutin yang hasilnya dikirimkan via email, teknik *web scraping* untuk mengumpulkan data pasar, hingga integrasi data sistem eksternal melalui API. Otomasi ini menjadi

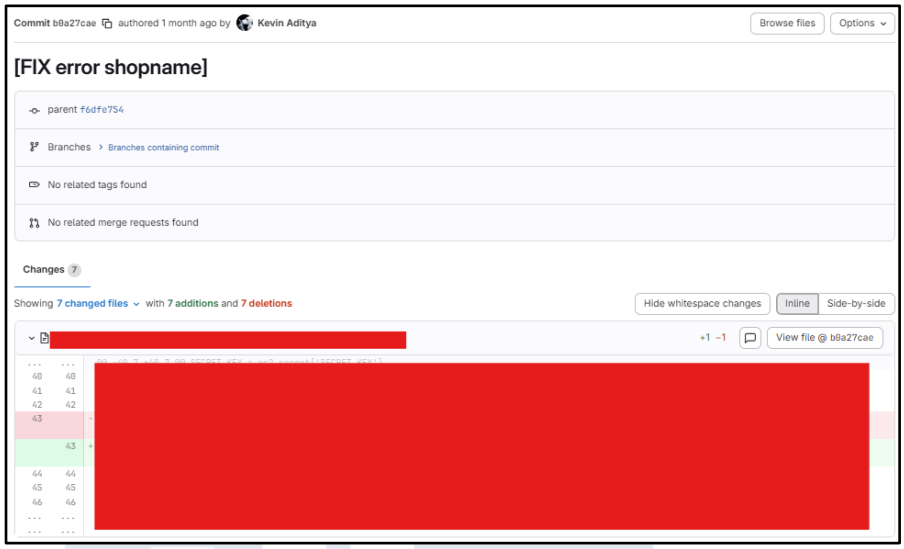
fondasi operasional data yang menjamin ketersediaan informasi tepat waktu bagi berbagai divisi.

Data Engineer Intern berperan dalam pengelolaan dan pengembangan skrip Python tersebut. Tanggung jawab yang diemban tidak hanya terbatas pada pembuatan skrip baru sesuai permintaan pengguna, tetapi juga mencakup pemeliharaan stabilitas sistem yang sudah berjalan. Skrip-skrip ini dirancang secara modular agar mudah dikelola dan disesuaikan dengan kebutuhan spesifik, baik untuk keperluan laporan harian maupun ingestasi data ke dalam *Data Lake*. Penguasaan terhadap logika pemrograman dan pemahaman mendalam mengenai struktur data menjadi kompetensi utama yang diterapkan dalam tugas ini.

Agar proses otomasi tetap berjalan optimal, kegiatan *maintenance* dilakukan secara disiplin dan berkala. Gangguan pada eksekusi skrip tidak dapat dihindari karena adanya faktor eksternal seperti perubahan struktur halaman *website* target, pembaruan *endpoint* API, atau perubahan skema data sumber. Untuk memitigasi dampak dari gangguan tersebut, sistem pemantauan otomatis telah diintegrasikan ke dalam alur kerja. Ketika sebuah skrip mengalami kegagalan eksekusi, sistem secara otomatis mengirimkan notifikasi peringatan yang berisi detail *error* ke Discord atau Telegram milik tim, sebagaimana ditunjukkan pada Gambar 3.25. Mekanisme peringatan dini ini memungkinkan tim data untuk merespons insiden dengan cepat sebelum berdampak pada *user*.

Gambar 3.25 Notifikasi *Error* Otomasi pada Telegram

Proses penanganan insiden dimulai dengan penelusuran skrip terkait pada repositori kode untuk mengidentifikasi akar permasalahan. *Debugging* dilakukan secara mendalam untuk menentukan apakah kegagalan disebabkan oleh kesalahan logika internal atau perubahan pada sistem eksternal, yang kemudian diikuti dengan *bug fixing*. Setelah perbaikan selesai dan diverifikasi, kode yang telah diperbarui diunggah kembali (*push*) ke *repository* GitLab perusahaan. Setiap perubahan kode wajib disertai dengan pesan commit yang deskriptif untuk menjelaskan konteks perbaikan yang dilakukan, seperti yang terlihat pada Gambar 3.26. Dokumentasi perubahan ini sangat vital untuk menjaga transparansi riwayat kode dan memudahkan kolaborasi antar anggota tim dalam jangka panjang.



Gambar 3.26 Riwayat *Commit* Perbaikan pada GitLab

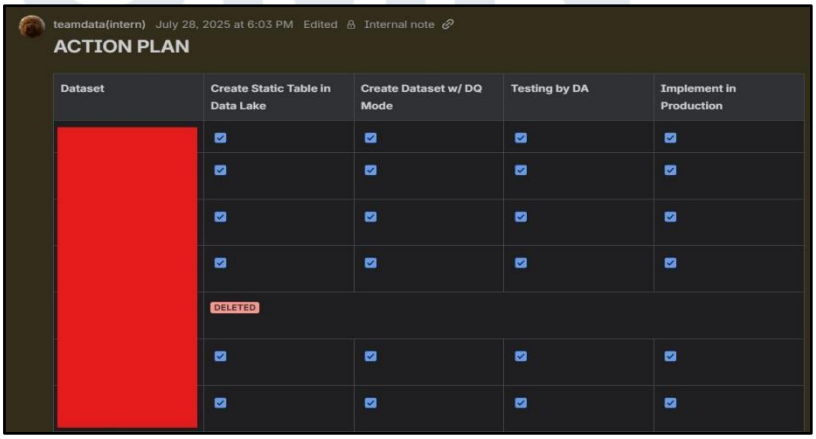
3.3.1.8 Proyek migrasi *dataset* QuickSight dari SPICE ke Direct Query Athena untuk optimasi biaya

Ruparupa menggunakan Amazon QuickSight sebagai alat visualisasi data utama untuk mendukung pengambilan keputusan bisnis. Namun, tantangan muncul dari biaya penggunaan memori SPICE (*Super-fast, Parallel, In-memory Calculation Engine*) yang terus membengkak seiring bertambahnya volume data historis. Banyak *dataset* berukuran besar, seperti data penjualan *omnichannel* dari tahun 2016 hingga 2022, tersimpan di SPICE meskipun datanya bersifat statis dan jarang berubah. Kondisi ini mengakibatkan anggaran operasional *cloud* yang tidak efisien dikarenakan perusahaan harus membayar biaya penyimpanan memori berkinerja tinggi yang sebenarnya tidak diperlukan untuk data yang sudah tidak diperbaharui lagi. Oleh karena itu, inisiatif migrasi ke metode *Direct Query* menggunakan Athena menjadi prioritas mendesak untuk mereduksi biaya tanpa mengorbankan aksesibilitas data bagi pengguna bisnis.

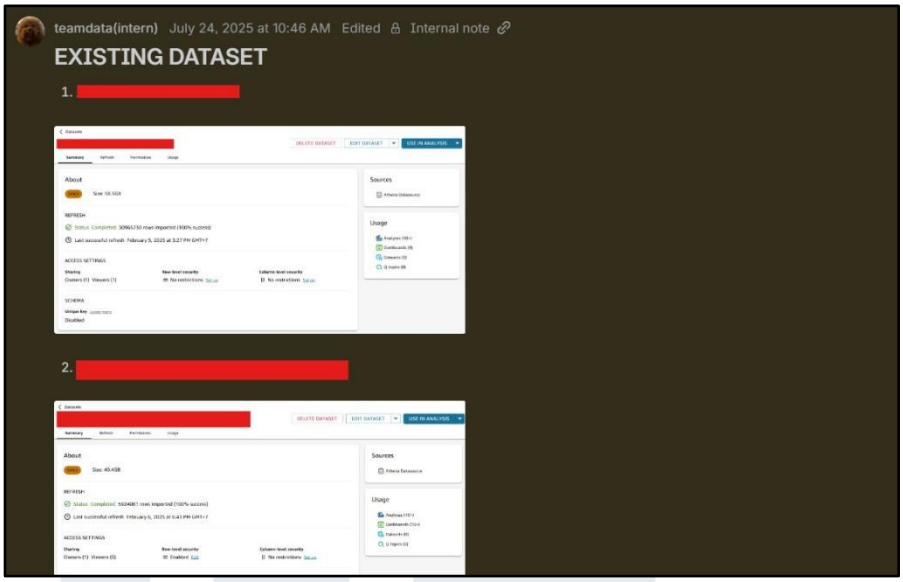
Proyek optimalisasi biaya ini berfokus pada eksekusi migrasi arsitektur penyimpanan data di level *backend* QuickSight. Tugas utama meliputi analisis penggunaan *dataset* yang sudah ada,

perancangan tabel statis di *Data Lake*, serta konfigurasi ulang koneksi data dari mode SPICE ke *Direct Query*. Diperlukan ketelitian tinggi untuk memastikan bahwa perubahan metode penarikan data ini tidak merusak *dashboard* atau analisis yang sedang digunakan oleh tim bisnis. Selain itu, tugas juga mencakup dokumentasi teknis yang rinci dan validasi data untuk menjamin bahwa angka yang tampil setelah migrasi tetap akurat 100% dibandingkan data sebelumnya.

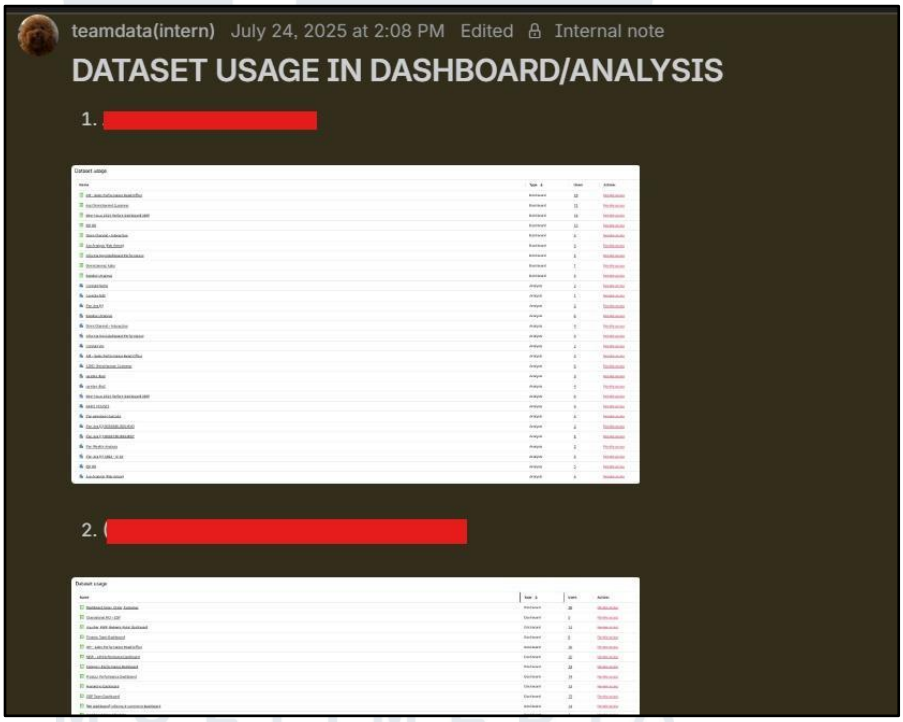
Tahap awal dimulai dengan pemetaan terhadap *dataset* yang membebani kapasitas SPICE. Daftar *dataset* prioritas disusun dalam sebuah *Action Plan* yang memuat status pengerjaan mulai dari pembuatan tabel statis, pengujian, hingga implementasi produksi, sebagaimana terlihat pada Gambar 3.27. Langkah selanjutnya adalah mengidentifikasi spesifikasi *dataset* yang sudah ada, termasuk ukuran penyimpanan dan frekuensi pembaruan data, untuk menentukan strategi migrasi yang tepat seperti yang ditunjukkan pada Gambar 3.28. Sebelum perubahan dilakukan, pengecekan dampak (*impact analysis*) wajib dilakukan untuk mengetahui *dashboard* dan analisis mana saja yang terhubung dengan *dataset* tersebut. Daftar penggunaan *dataset* di berbagai laporan bisnis dipetakan secara rinci pada Gambar 3.29 untuk memastikan komunikasi yang tepat kepada para pemangku kepentingan sebelum migrasi dijalankan.



Gambar 3.27 *Action Plan* Migrasi *Dataset*



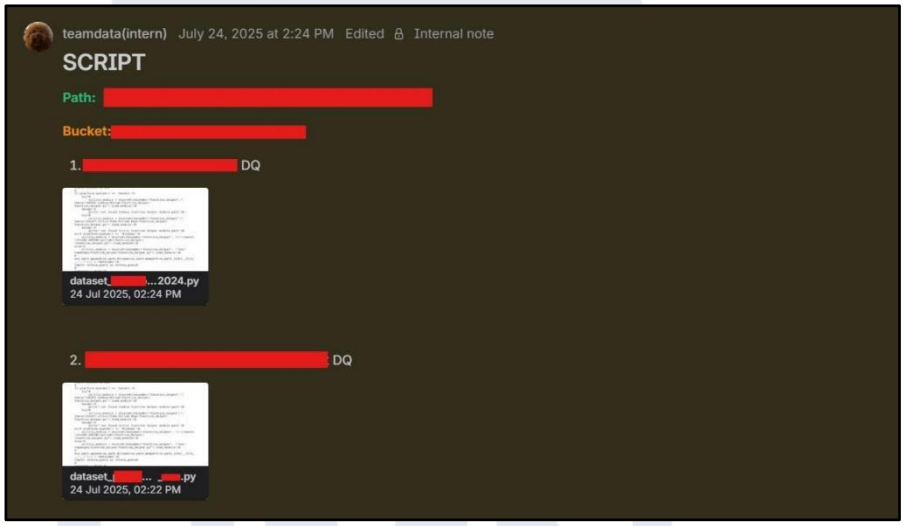
Gambar 3.28 Identifikasi Dataset Eksisting Berbasis SPICE



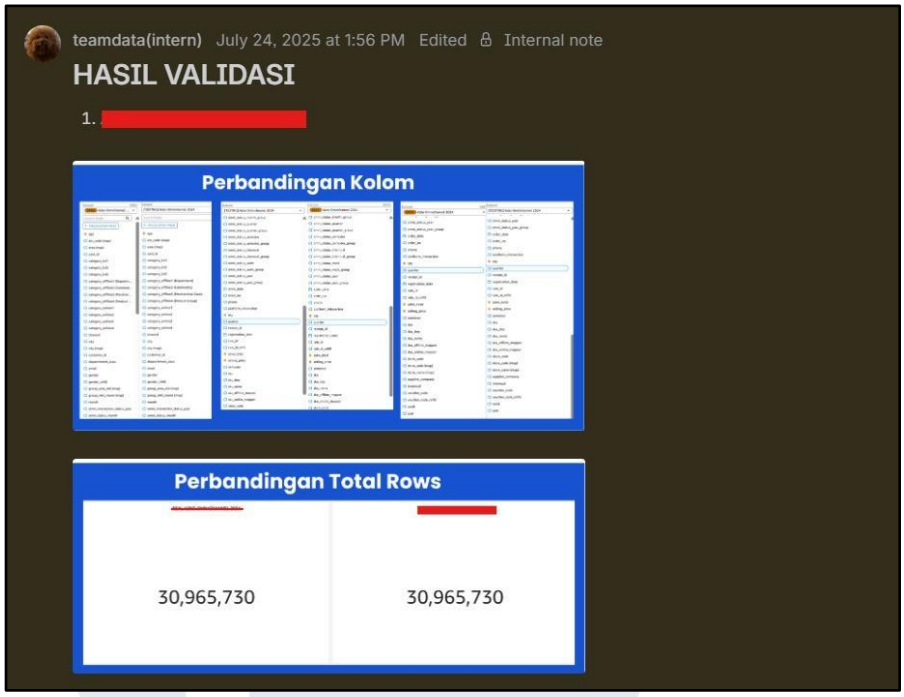
Gambar 3.29 Analisis Dampak Penggunaan *Dataset* pada *Dashboard*

Proses migrasi dilakukan dengan membuat *dataset* baru yang menggunakan metode *Direct Query* ke tabel statis di Athena, yang ditandai dengan sufiks "DQ" pada nama *dataset* seperti terlihat pada Gambar 3.30. Setelah *dataset* baru terbentuk, dilakukan proses validasi data dengan membandingkan total baris dan nilai kolom

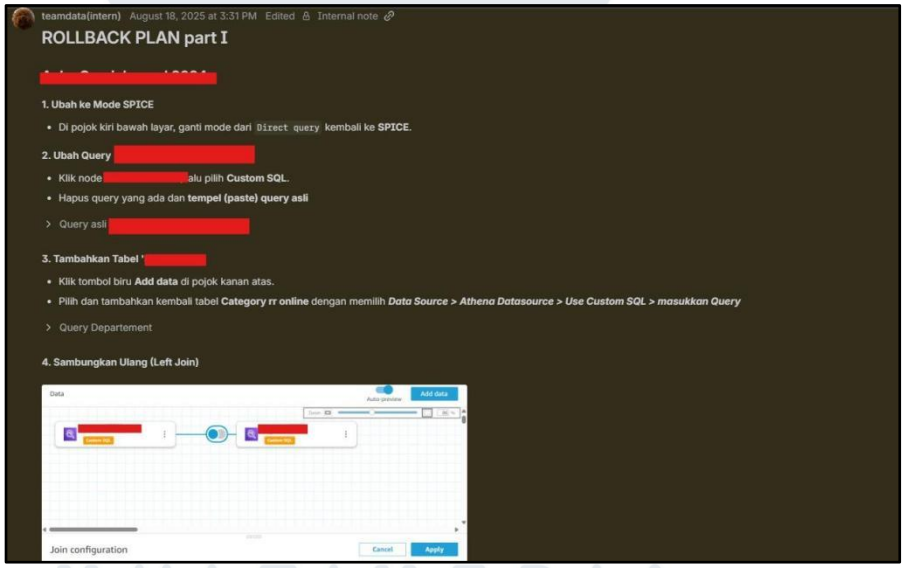
antara *dataset* lama berbasis SPICE dan *dataset* baru berbasis *Direct Query*. Hasil komparasi pada Gambar 3.31 menunjukkan kesesuaian data yang identik, menandakan bahwa migrasi berhasil tanpa distorsi informasi. Sebagai langkah mitigasi risiko, prosedur *rollback* juga disiapkan secara mendetail. Rencana ini, yang terdokumentasi pada Gambar 3.32, memuat instruksi langkah demi langkah untuk mengembalikan konfigurasi ke mode SPICE dan memulihkan *query* SQL asli jika terjadi kendala performa atau *error* setelah implementasi.



Gambar 3.30 Konfigurasi *Dataset* Baru dengan *Direct Query*



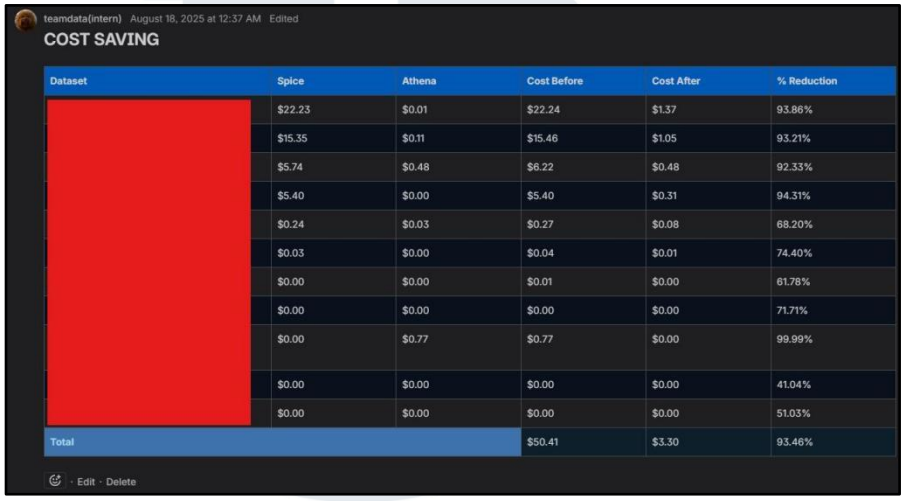
Gambar 3.31 Validasi Data Antara SPICE dan *Direct Query*



Gambar 3.32 Rencana *Rollback* Migrasi

Implementasi proyek ini menghasilkan dampak efisiensi finansial yang sangat signifikan bagi biaya operasional infrastruktur data perusahaan. Dengan memindahkan data historis yang bersifat statis dari memori mahal SPICE ke penyimpanan murah berbasis S3/Athena, biaya bulanan untuk *dataset* target berhasil dipangkas secara drastis. Tabel kalkulasi penghematan biaya pada Gambar 3.33

menunjukkan penurunan biaya total dari $50.41 menjadi hanya $3.30 per bulan. Angka ini merepresentasikan persentase pengurangan biaya sebesar 93,46%, membuktikan efektivitas strategi *Direct Query* untuk pengelolaan data arsip jangka panjang. Keberhasilan ini memberikan ruang penyimpanan lebih besar bagi tim untuk mengalokasikan kapasitas SPICE bagi data operasional yang lebih dinamis dan membutuhkan performa tinggi.


teamdata(intern) August 18, 2025 at 12:37 AM Edited

COST SAVING

Dataset	Spice	Athena	Cost Before	Cost After	% Reduction
	$22.23	$0.01	$22.24	$1.37	93.86%
	$15.35	$0.11	$15.46	$1.05	93.21%
	$5.74	$0.48	$6.22	$0.48	92.33%
	$5.40	$0.00	$5.40	$0.31	94.31%
	$0.24	$0.03	$0.27	$0.08	68.20%
	$0.03	$0.00	$0.04	$0.01	74.40%
	$0.00	$0.00	$0.01	$0.00	61.78%
	$0.00	$0.00	$0.00	$0.00	71.71%
	$0.00	$0.77	$0.77	$0.00	99.99%
	$0.00	$0.00	$0.00	$0.00	41.04%
	$0.00	$0.00	$0.00	$0.00	51.03%
Total			$50.41	$3.30	93.46%

Edit · Delete

Gambar 3.33 Tabel Kalkulasi Penghematan Biaya Setelah Migrasi

3.3.2 Kendala yang Ditemukan

Selama menjalani masa magang sebagai *Data Engineer Intern* di Ruparupa, terdapat berbagai tantangan yang dihadapi dalam proses pengerjaan tugas maupun kolaborasi dengan tim lintas divisi. Kendala-kendala ini muncul baik dari sisi teknis maupun non-teknis, dan menjadi bagian penting dalam proses pembelajaran serta evaluasi kinerja. Berikut adalah rincian kendala utama yang ditemukan selama periode magang beserta penjelasannya:

1. Spesifikasi Permintaan *User* yang Kerap Berubah

   Salah satu tantangan terbesar yang sering dihadapi dalam pengerjaan tiket adalah ketidakkonsistenan permintaan dari sisi *user* bisnis. Sering kali, definisi kebutuhan atau logika data yang telah disepakati di awal berubah di tengah proses pengembangan, sehingga mengharuskan perombakan ulang pada skrip atau struktur tabel yang telah dibangun. Perubahan

spesifikasi yang mendadak ini tidak hanya mengganggu alur kerja yang telah direncanakan, tetapi juga membuang sumber daya waktu dan komputasi yang seharusnya dapat dialokasikan untuk tugas lain. Akibatnya, efisiensi pengerjaan menjadi menurun dan target penyelesaian tiket (SLA) berisiko terlampaui karena proses iterasi revisi yang berulang-ulang tanpa finalisasi parameter yang jelas sejak awal.

2. Respon Lambat dari *User* yang Menghambat Alur Kerja

   Kendala signifikan lainnya berkaitan dengan alur komunikasi, di mana respon yang lambat dari pihak user sering kali menjadi penghambat utama kelancaran penyelesaian tugas. Dalam banyak kasus, proses pengembangan teknis sangat bergantung pada konfirmasi mendetail terkait logika bisnis, pemberian akses ke folder data tertentu, atau validasi hasil data sebelum dapat dilanjutkan ke tahap produksi. Ketika user tidak segera memberikan umpan balik atau persetujuan yang dibutuhkan karena kesibukan operasional mereka, status tiket terpaksa tertahan dalam posisi pending atau blocked untuk jangka waktu yang tidak menentu. Penundaan ini secara langsung menciptakan *bottleneck* dalam antrean pekerjaan tim data, yang pada akhirnya menunda keseluruhan timeline proyek dan mempengaruhi metrik produktivitas individu.

3. Kurangnya Ketelitian dalam Penulisan Kode dan Logika

   Selain faktor eksternal, tantangan juga muncul dari aspek internal, yaitu kurangnya ketelitian dalam penulisan kode pemrograman atau penyusunan logika pemrosesan data. Tingkat kompleksitas skrip Python dan kueri SQL yang tinggi menuntut presisi yang mutlak, di mana kesalahan kecil seperti *typo* pada nama variabel atau kekeliruan logika filter dapat menyebabkan kegagalan fatal pada eksekusi *pipeline*. Ketidaktelitian ini sering kali baru terdeteksi saat skrip dijalankan secara penuh atau setelah data diverifikasi, yang kemudian memaksa dilakukannya proses *debugging* ulang yang memakan waktu cukup lama.

3.3.3 Solusi atas Kendala yang Ditemukan

Dari kendala-kendala yang telah dihadapi, muncul solusi untuk mengatasi kendala atau kesulitan tersebut, yakni:

1. Melakukan komunikasi yang lebih intensif dan terstruktur dengan *user* pada tahap awal penerimaan tiket (*requirement gathering*). Hal ini dilakukan dengan cara mengkonfirmasi kembali detail logika dan spesifikasi data secara tertulis pada tiket JIRA sebelum proses pengembangan dimulai. Dokumentasi kebutuhan yang jelas di awal berfungsi sebagai acuan baku yang menyepakati batasan pengerjaan, sehingga dapat meminimalisir risiko permintaan perubahan spesifikasi yang mendadak di tengah jalan dan menjaga agar pengerjaan tetap sesuai dengan *timeline*.
2. Menerapkan strategi manajemen waktu yang proaktif dengan melakukan *follow-up* secara berkala kepada *user* melalui saluran komunikasi resmi perusahaan. Apabila respon masih belum didapatkan dalam batas waktu wajar, inisiatif diambil dengan mengalihkan fokus sementara untuk mengerjakan tiket lain yang ada dalam antrean agar produktivitas tidak terhenti. Jika hambatan tersebut bersifat krusial dan mendesak, permasalahan akan dikonsultasikan kepada *supervisor* untuk mendapatkan arahan atau bantuan koordinasi lebih lanjut.
3. Membangun standar verifikasi mandiri yang lebih ketat sebelum mengajukan *code review* atau *deployment*. Proses ini mencakup pelaksanaan uji coba skrip secara lokal dengan berbagai skenario data untuk memastikan logika berjalan sesuai harapan. Selain itu, membiasakan diri untuk melakukan pengecekan ulang baris demi baris terhadap penamaan variabel dan sintaks SQL menjadi langkah preventif yang efektif untuk menekan angka kesalahan akibat ketidaktelitian dan menjamin kualitas kode yang dihasilkan.