BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

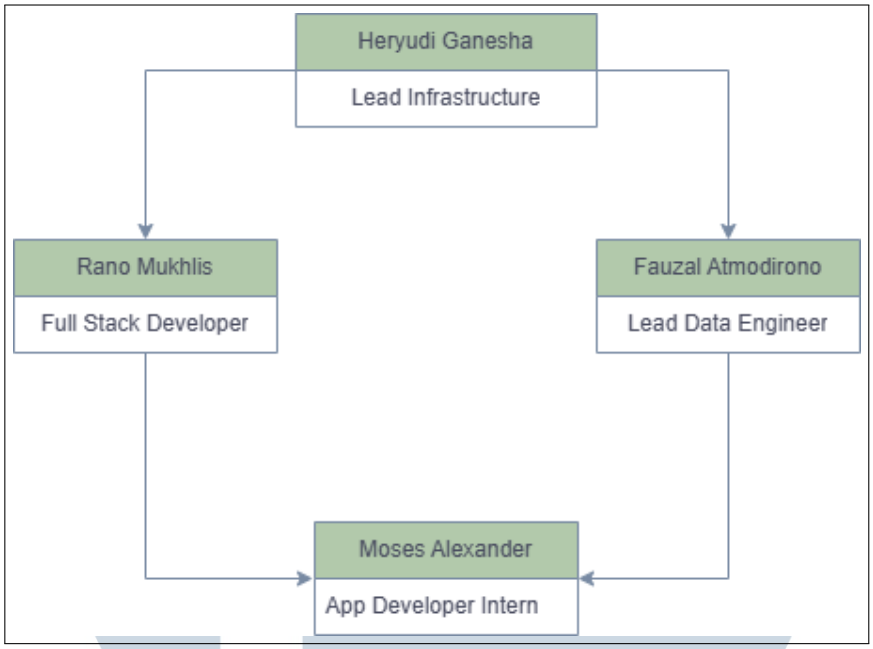
Dalam kerja magang di PT Devoteam G Cloud Services, berbagai tugas dikoordinasikan dan dilaksanakan sesuai dengan peran dan tanggung jawab yang ditetapkan. Berikut ini adalah penjabarannya

Program magang dijalani di PT Devoteam G Cloud Services sebagai *App Developer Intern*, di bawah naungan *Professional Service Department*. Departemen ini memiliki peran penting dalam merancang dan mengimplementasikan solusi digital berbasis kebutuhan klien dengan pendekatan profesional, efisien, dan berorientasi pada teknologi *cloud*.

Dalam kedudukan tersebut, tanggung jawab utama yang dijalankan adalah sebagai berikut:

1. Perancangan, pengembangan, dan pemeliharaan aplikasi berbasis web dan mobile dilakukan, mencakup sisi *front-end* maupun *back-end*.
2. Desain, pengelolaan, serta optimalisasi database dilakukan, termasuk integrasi dan/atau pembangunan *Application Programming Interface* (API) sesuai kebutuhan sistem.
3. Proses pengujian aplikasi dilaksanakan untuk memastikan kualitas, fungsionalitas, dan stabilitas, serta perbaikan bug ditangani berdasarkan hasil pengujian atau pelaporan dari pengguna.

Selama program magang, koordinasi dilakukan secara langsung dengan *Full Stack Developer* dan *Lead Data Engineer* dari *Professional Service Department* sebagai pembimbing teknis dalam pengerjaan proyek. Kedudukan dan hubungan koordinasi ini dapat dilihat pada diagram struktur organisasi yang ditampilkan pada gambar 3.1.



Gambar 3.1. Kedudukan di PT Devoteam G Cloud Services

Proses koordinasi dilaksanakan secara daring dengan mengutamakan efisiensi dan fleksibilitas komunikasi, menyesuaikan dengan sistem kerja yang diterapkan di PT Devoteam G Cloud Services. Adapun media utama yang digunakan dalam proses koordinasi adalah sebagai berikut:

1. Google Chat (Spaces): Dimanfaatkan untuk penyampaian laporan *daily progress*, diskusi terkait kendala teknis, serta pendokumentasian arahan dan pembagian tugas dalam satu ruang kerja tim secara terstruktur.
2. Pesan Pribadi (Direct Chat): Digunakan untuk komunikasi langsung satu lawan satu dengan *Full Stack Developer* maupun *Lead Data Engineer* terkait konfirmasi penyelesaian tugas, permintaan *review*, atau klarifikasi teknis yang lebih mendalam.

Melalui metode koordinasi ini, kontribusi aktif terhadap proyek dapat diberikan, ekspektasi kerja dari masing-masing pembimbing dapat dipahami, dan penyelesaian setiap tugas dapat dipastikan sesuai dengan kualitas serta ketepatan waktu yang diharapkan.

3.2 Tugas yang Dilakukan

Keterlibatan dalam proyek migrasi database Smartfren ke Google BigQuery dilakukan sebagai bagian dari pelaksanaan kegiatan magang. Dalam proyek ini, peran yang dijalankan meliputi proses migrasi basis data milik Smartfren dari sistem *on-premise* Greenplum menuju layanan *cloud-native* Google BigQuery. Migrasi ini merupakan bagian dari strategi modernisasi infrastruktur data yang bertujuan untuk meningkatkan efisiensi, skalabilitas, dan kecepatan pemrosesan data. PT Devoteam G Cloud Services bertindak sebagai vendor resmi sekaligus mitra teknis dalam pelaksanaan proyek tersebut. Melalui proyek ini, pengalaman langsung diperoleh dalam memahami struktur data Greenplum, melakukan analisis skema dan *query*, serta berkontribusi dalam proses validasi dan pengoptimalan data yang dimigrasikan ke lingkungan BigQuery. Proyek ini juga menjadi sarana pengenalan terhadap ekosistem Google Cloud Platform (GCP), serta praktik terbaik dalam pengelolaan data berskala besar dan integrasi layanan *cloud*.

Selama proses migrasi berlangsung, sistem Greenplum tetap berfungsi sebagai sumber utama (*master*) yang aktif memproses dan menghasilkan data operasional. Oleh karena itu, validasi terhadap hasil migrasi dilakukan secara bertahap dan berkelanjutan, dengan membandingkan hasil output fungsi di Greenplum dan BigQuery hingga didapatkan kesesuaian satu banding satu. Pendekatan ini bertujuan memastikan bahwa data dan logika fungsional yang telah dikonversi ke BigQuery tetap akurat dan konsisten dengan sistem sumbernya.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang diuraikan seperti pada Table 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

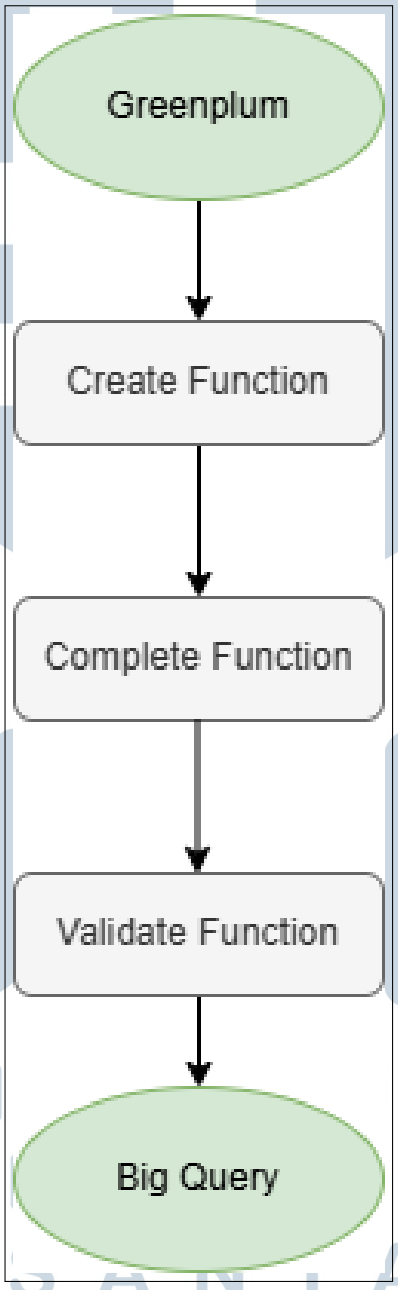
Minggu Ke -	Pekerjaan yang dilakukan
1	Belajar mandiri dengan menggunakan akses dari google partner skillboost.
2	Mendalami framework FastAPI dengan mempelajari kode yang telah disediakan oleh pengembang senior, serta mendapatkan arahan langsung dari mentor di PT Devoteam Cloud Services.
	Lanjutan pada halaman berikutnya

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang (lanjutan)

Minggu Ke -	Pekerjaan yang dilakukan
3	Mengerjakan proyek HRIS Devoteam, mencakup integrasi antara sisi frontend dan backend melalui pemanfaatan endpoint yang tersedia, serta implementasi fitur seperti pada dashboard, halaman history pengguna dan membuat halaman untuk admin melihat semua history leave.
4	Menambahkan fitur untuk melihat riwayat cuti (leave history) dengan opsi penyaringan berdasarkan urutan naik atau turun (ascending maupun descending). Serta membuat dokumentasi endpoint menggunakan postman untuk projek BNI.
5	Mulai melakukan Migrasi database smartfren pada tahap create function.
6	Melanjutkan Migrasi database smartfren pada tahap create function.
7	Melanjutkan Migrasi database smartfren pada tahap create function.
8	Mulai melakukan Migrasi database smartfren pada tahap complete function.
9	Melanjutkan Migrasi database smartfren pada tahap complete function.
10	Melanjutkan Migrasi database smartfren pada tahap complete function.
11	Mulai melakukan Migrasi database smartfren pada tahap Validate function.
12	Melanjutkan melakukan Migrasi database smartfren pada tahap Validate function.
13	Melanjutkan melakukan Migrasi database smartfren pada tahap Validate function.
14	Melanjutkan melakukan Migrasi database smartfren pada tahap Validate function.
15	Melakukan End to End testing.
16	Melanjutkan End to End testing.

3.3.1 Proses Migrasi

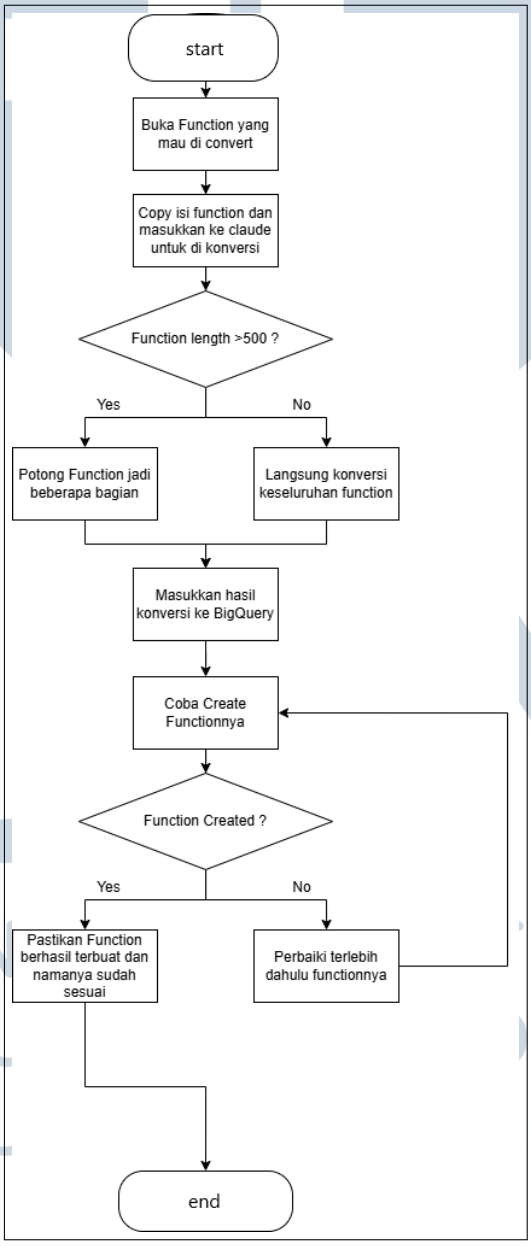
Proses migrasi ini dilakukan secara bertahap melalui beberapa fungsi dan tahapan validasi. Berikut adalah penjabaran proses migrasi yang dilakukan, sesuai dengan alur pada Gambar 3.2.



Gambar 3.2. Alur Migrasi Data dari Greenplum ke BigQuery

A Create Function

Salah satu tahapan penting dalam proses migrasi data adalah pembuatan ulang fungsi (*create function*) dari sistem sumber, yaitu Greenplum, ke dalam format yang sesuai dengan lingkungan BigQuery. Fungsi-fungsi ini semula ditulis menggunakan *PostgreSQL* dan perlu dikonversi ke dalam sintaks *Standard SQL* milik Google agar dapat dijalankan di BigQuery sebagai *stored procedure routine*. Alur tahapan validasi secara umum dapat dilihat pada gambar 3.3.



Gambar 3.3. Flowchart Create Function

```sql
SELECT DISTINCT(partition_date)
FROM dataset.table
WHERE partition_date >= '2025-02-01'
  AND partition_date < '2025-03-01';
```

Kode 3.1: Query cek tanggal

Kode 3.1 menunjukkan query yang digunakan untuk memverifikasi ketersediaan data berdasarkan tanggal.

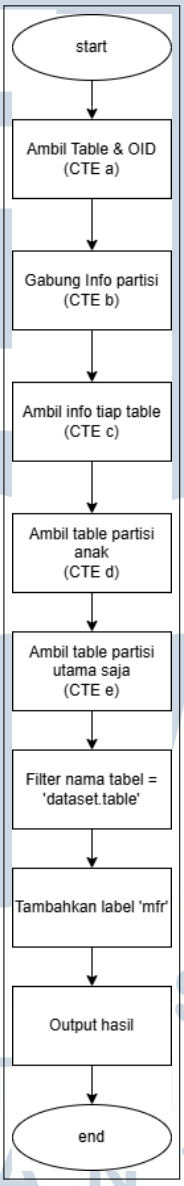
```sql
WITH a AS (
    SELECT c.oid table_id, t.schemaname, t.tablename
    FROM pg_tables t
    JOIN pg_class c
        ON t.tablename = c.relname
), b AS (
    SELECT DISTINCT x.*,
        CASE
            WHEN y.inhparent IS NOT NULL AND y.inhrelid IS NOT
    NULL
            THEN TRUE
            ELSE FALSE
        END is_partitiontable,
        z.columnname partitionkey,
        p.partition_period
    FROM a AS x
    LEFT JOIN pg_inherits AS y
        ON x.table_id = y.inhparent
    LEFT JOIN pg_partition_columns AS z
        ON x.schemaname = z.schemaname
        AND x.tablename = z.tablename
    LEFT JOIN (
        SELECT parent_table AS parent_id,
               child_table AS child_id,
               schemaname,
               tablename,
               CASE
                   WHEN tablename ~* 'prt_d' THEN 'daily'
                   WHEN tablename ~* 'prt_m' THEN 'monthly'
                   ELSE NULL
               END AS partition_period
        FROM (
            SELECT * FROM (
                SELECT * FROM a
            ) AS x
```

```
            JOIN (
                SELECT inhparent parent_table,
                       MAX(inhrelid) child_table
                FROM pg_inherits
                GROUP BY 1
            ) AS y
            ON x.table_id = y.child_table
        ) AS m
    ) AS p
    ON x.table_id = p.parent_id
), c AS (
    SELECT DISTINCT
        schemaname schema_name,
        tablename table_name,
        FIRST_VALUE(is_partitiontable) OVER (
            PARTITION BY schemaname, tablename
            ORDER BY is_partitiontable DESC
        ) is_partitiontable,
        FIRST_VALUE(partitionkey) OVER (
            PARTITION BY schemaname, tablename
            ORDER BY is_partitiontable DESC
        ) partition_key,
        FIRST_VALUE(partition_period) OVER (
            PARTITION BY schemaname, tablename
            ORDER BY is_partitiontable DESC
        ) partition_period
    FROM b
), d AS (
    SELECT DISTINCT x.*, y.inhrelid, y.inhparent
    FROM a AS x
    LEFT JOIN pg_inherits AS y
        ON x.table_id = y.inhrelid
    WHERE y.inhrelid IS NOT NULL
), e AS (
    SELECT * FROM c AS x
    WHERE NOT EXISTS (
        SELECT 1
        FROM d AS y
        WHERE x.schema_name = y.schemaname
          AND x.table_name = y.tablename
    )
)
SELECT 'mfr' AS sources, *
```

```
78 FROM e
79 WHERE e.schema_name || '.' || e.table_name IN ('dataset.table')
80 ORDER BY table_name;
```

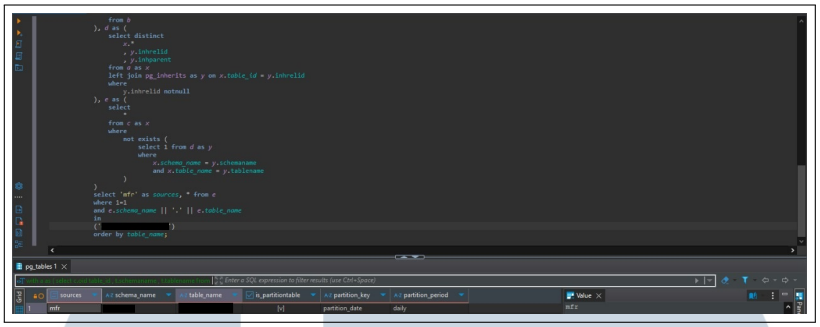
Kode 3.2: Query partisi table

Query pada Kode 3.2 ini digunakan untuk mengecek apakah table yang digunakan adalah table partisi atau bukan. Proses cara pengecekannya dapat di lihat pada gambar 3.4



Gambar 3.4. Alur pencarian tipe partisi table

Contoh penulisan table partisi pada Greenplum dapat dilihat pada gambar 3.5



Gambar 3.5. Contoh table partisi pada Greenplum

Dengan jumlah fungsi yang sangat besar, mencapai sekitar 1500 fungsi, proses konversi dilakukan secara efisien dengan bantuan model kecerdasan buatan. Tim Machine Learning menyediakan bantuan berupa penggunaan *Vertex AI Claude 3.7 Sonnet*, yaitu model *AI* yang dioptimalkan untuk memahami struktur dan logika fungsi *SQL*, lalu melakukan translasi otomatis dari *PostgreSQL* ke BigQuery.

Fungsi-fungsi tersebut diekstraksi dari dokumen yang tersimpan di Google Drive, kemudian dikonversi melalui antarmuka *Vertex AI*. Dalam proses ini, digunakan prompt khusus yang berisi instruksi konversi secara sistematis, termasuk parameter proyek, nama *dataset*, serta aturan perubahan konvensi penamaan. Hasil konversi yang dihasilkan kemudian ditinjau ulang, baik secara otomatis maupun manual, untuk memastikan tidak ada kesalahan sintaks atau perbedaan logika dibandingkan fungsi aslinya.

Setelah proses verifikasi selesai, hasil konversi dipindahkan ke editor BigQuery untuk dilakukan eksekusi sebagai *stored procedure*. Validasi akhir juga dilakukan, seperti memastikan konsistensi nama fungsi antara Greenplum dan BigQuery, pengecekan struktur table partisi agar sesuai dengan format BigQuery, serta keakuratan penulisan identifier table dalam format *project.dataset.table*.

Langkah awal yang wajib dilakukan dalam proses validasi migrasi adalah memastikan bahwa struktur table partisi sudah sesuai dengan struktur yang digunakan di BigQuery. Berikut adalah contoh penulisan table partisi pada Greenplum Kode 3.3.

```
1 FROM nama_dataset.nama_table_1_prt_d_(tanggal)
```

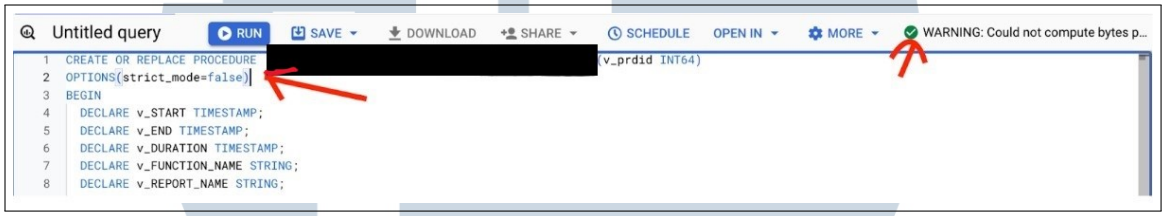
Kode 3.3: Contoh Table Partisi Greenplum

sementara berikut adalah cara penulisan table partisi di BigqueryKode 3.4.

```
1 FROM nama_dataset.nama_table
2 WHERE partition_date = '(tanggal)'
```

Kode 3.4: Contoh Table Partisi BigQuery

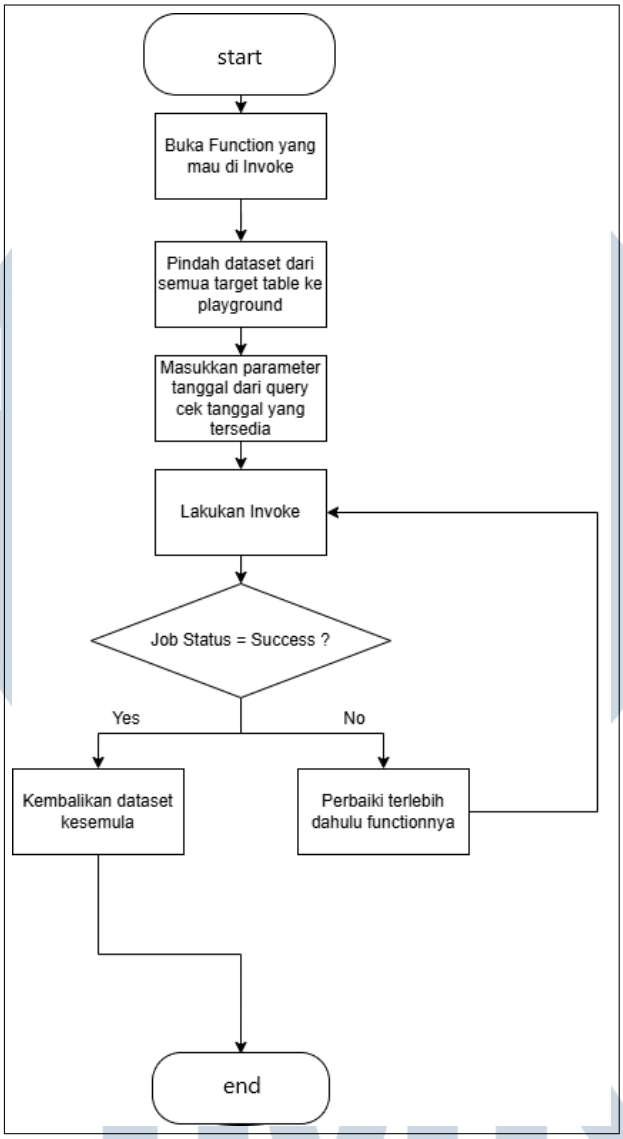
Selain itu, perlu dipastikan bahwa struktur ID table BigQuery (*project.dataset.table*) telah dituliskan secara tepat untuk menghindari error pada saat pemanggilan fungsi. Jika ditemukan bahwa ada table yang tidak tersedia, maka nama dataset dan table tersebut perlu dicatat dalam dokumen *tracker*, khususnya pada sheet *list_table*. Dalam kasus ini, agar fungsi tetap dapat dibuat dan dijalankan di BigQuery meskipun ada table yang hilang, disarankan untuk menambahkan opsi *OPTIONS(strict_mode=false)* sebagaimana ditunjukkan pada Gambar 3.6.



Gambar 3.6. Penggunaan OPTIONS(strict_mode=false)

B Complete Function

Setelah fungsi berhasil dibuat di BigQuery, tahap berikutnya adalah proses penyempurnaan dan pengujian untuk memastikan bahwa fungsi tersebut dapat berjalan sebagaimana mestinya tanpa menimbulkan kesalahan atau dampak negatif pada data produksi. Tahap ini penting dilakukan untuk menjamin konsistensi dan integritas logika fungsi hasil migrasi. Alur tahapan validasi secara umum dapat dilihat pada gambar 3.7.



Gambar 3.7. Flowchart Complete Function

Penyempurnaan dimulai dengan mengakses fungsi yang telah dibuat melalui *editor* di BigQuery, kemudian dilakukan penelusuran terhadap seluruh elemen dalam fungsi, terutama bagian-bagian yang melibatkan operasi *query* terhadap table. Fokus utama adalah memastikan bahwa semua *dependency table—table-table* yang disebut dalam klausa SELECT atau operasi lainnya—benar-benar tersedia di lingkungan BigQuery. Jika ada table yang belum tersedia, maka informasi tersebut dicatat dalam sistem pelacakan proyek, dan daftar tablenya ditambahkan ke lembar kerja referensi (*list_table*) untuk ditindaklanjuti lebih lanjut.

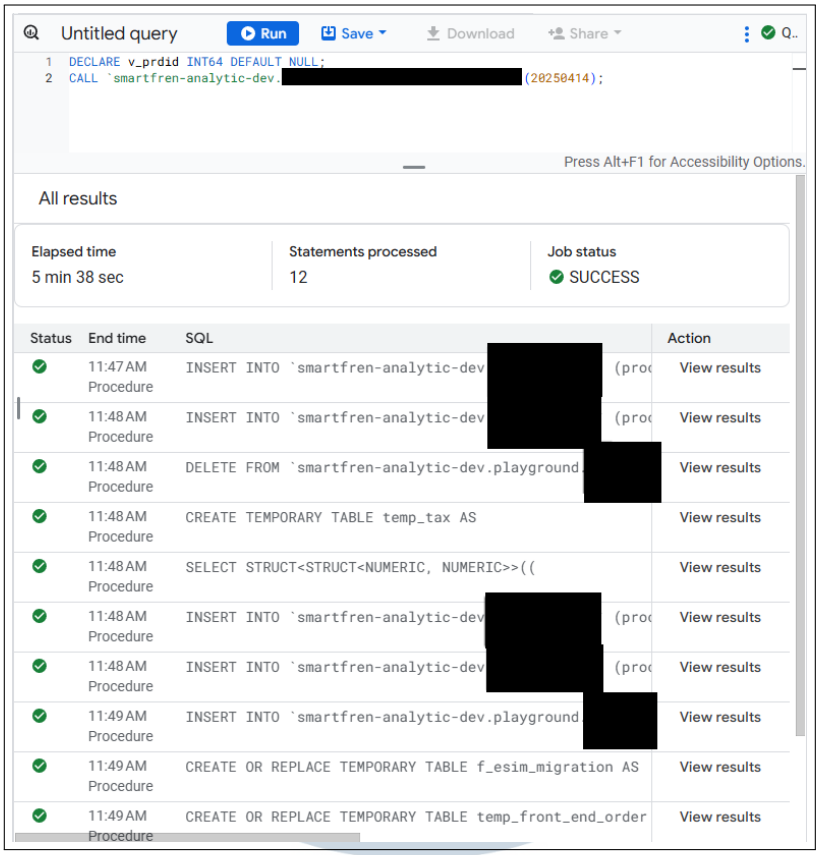
Selain ketersediaan table, perhatian khusus juga diberikan terhadap perintah-perintah destruktif dalam fungsi, seperti *INSERT INTO, TRUNCATE*

TABLE, DROP TABLE, DELETE FROM, UPDATE, dan CREATE OR REPLACE TABLE. Perintah semacam ini berpotensi mengubah atau menghapus data, sehingga pengujian fungsi yang mengandungnya tidak boleh langsung dijalankan pada table produksi.

Sebagai langkah mitigasi, replika dari *table-table* yang digunakan dalam fungsi dibuat dalam dataset khusus bernama *playground*. Replika ini hanya memuat sebagian data (misalnya *LIMIT 100*) dan difungsikan sebagai lingkungan aman untuk melakukan pengujian. Dengan begitu, fungsi dapat dijalankan secara bebas tanpa risiko terhadap data utama.

Proses pengujian dilakukan dengan memanggil fungsi tersebut menggunakan parameter yang telah ditentukan sebelumnya. Biasanya digunakan parameter waktu atau tanggal dalam rentang tertentu, seperti antara Desember 2024 hingga Februari 2025, tergantung pada kebutuhan fungsi. Selama proses pengujian, apabila muncul error, fungsi dikaji ulang dan diperbaiki sampai dapat dijalankan dengan sukses.

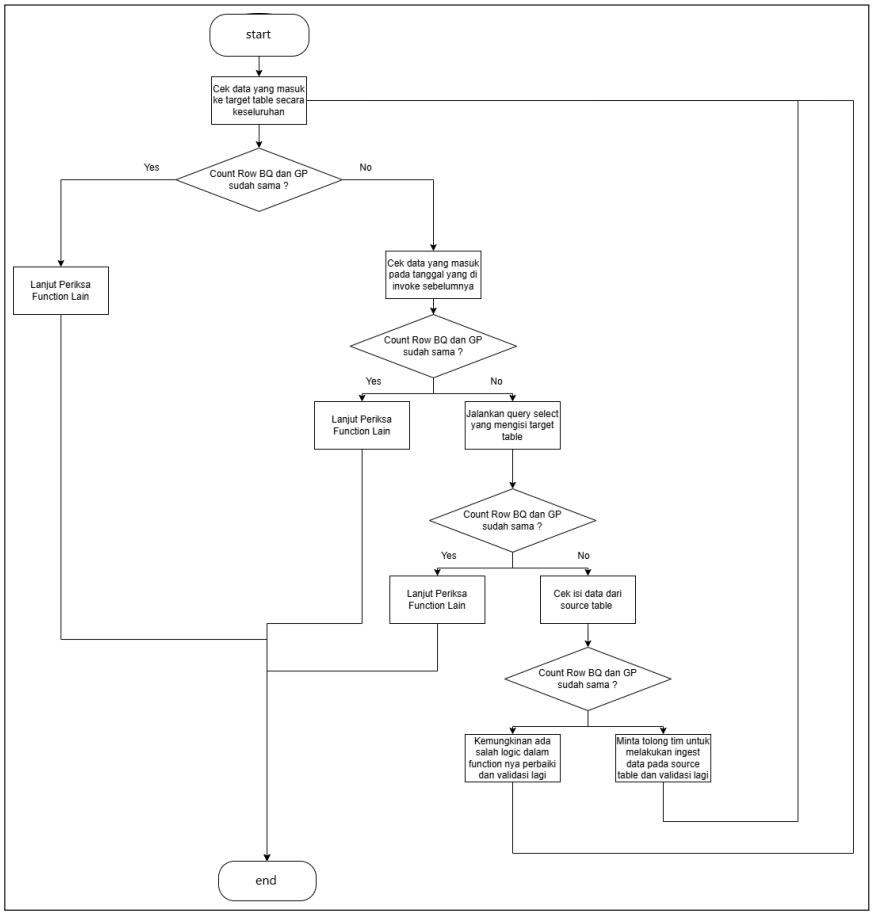
Setelah pengujian dinyatakan berhasil dan fungsi terbukti berfungsi dengan baik seperti Pada Gambar 3.8, nama *dataset* yang semula diubah ke *playground* dikembalikan ke *dataset* aslinya. Terakhir, fungsi disimpan kembali sebagai versi final dan siap digunakan dalam pipeline produksi.



Gambar 3.8. Hasil test invoke berhasil

C Validasi Fungsi

Setelah proses konversi fungsi dari Greenplum ke BigQuery dinyatakan selesai dan tidak menghasilkan error saat dieksekusi, tahap selanjutnya adalah melakukan proses validasi. Validasi berfungsi untuk memastikan bahwa fungsi yang telah dikembangkan memberikan hasil yang setara antara lingkungan lama (Greenplum) dan lingkungan baru (BigQuery), baik dari sisi sintaksis maupun logika fungsional. Alur tahapan validasi secara umum dapat dilihat pada gambar 3.9.



Gambar 3.9. Flowchart Validate Function

Proses ini melibatkan pembandingan output fungsi berdasarkan parameter tertentu, terutama tanggal, yang sebelumnya telah ditentukan dari analisis sumber data. Validasi dilakukan terhadap hasil eksekusi fungsi di kedua sistem menggunakan data yang identik. Perbandingan mencakup verifikasi jumlah baris, isi data, serta konsistensi terhadap ekspektasi bisnis. Validasi yang berhasil ditandai dengan kesamaan output antara Greenplum dan BigQuery tanpa ada penyimpangan yang signifikan.

Untuk mendukung proses ini, digunakan beberapa perangkat lunak, seperti DBeaver dan ZeroTier. DBeaver memfasilitasi akses ke database Greenplum secara langsung, memungkinkan tim untuk mengeksekusi fungsi dan mengambil data referensi dari sistem lama. Sementara itu, ZeroTier digunakan untuk membangun koneksi aman ke jaringan internal Smartfren, termasuk akses ke *Airflow* dan Greenplum melalui VPN.

Ketersediaan dan konfigurasi koneksi ke kedua lingkungan ini menjadi kunci untuk memastikan proses validasi berjalan dengan lancar. Data hasil

eksekusi fungsi dikumpulkan, lalu dibandingkan menggunakan query dasar seperti *COUNT(*)* untuk jumlah baris, serta pengecekan nilai data secara sampling. Seluruh hasil pengujian kemudian didokumentasikan dalam ETL Tracker pada bagian *Preparation Validation*, yang mencakup informasi seputar nama database, nama fungsi, table target, sumber data, rentang tanggal pengujian, dan status sinkronisasi data.

C.1 Tiga Layer Validasi

Proses validasi dilakukan secara bertahap melalui tiga *layer* untuk memastikan integritas dan kesesuaian data antar platform:

1. Layer 1 - Count Rows Target Table (GP vs BQ):
   Pada tahap ini, jumlah baris pada table target di Greenplum dibandingkan dengan jumlah baris di table target yang sama di BigQuery berdasarkan rentang tanggal tertentu. Ini bertujuan memastikan bahwa data yang berhasil dimuat di kedua sistem berada dalam cakupan yang sama. Contoh query yang digunakan untuk proses ini dapat dilihat pada Kode 3.5.

```
select count(*) from dataset.table
  WHERE partition_key = 'tanggal invoke sebelumnya'
```

Kode 3.5: Query cek isi data Target table dengan tanggal spesifik

2. Layer 2 - Count Rows Query Insert (GP vs BQ):
   Validasi dilakukan terhadap hasil dari *query SELECT* yang digunakan sebagai dasar untuk mengisi (*insert*) table target. Hasil dari eksekusi *query* ini di kedua sistem dibandingkan untuk memastikan bahwa logika transformasi yang digunakan menghasilkan jumlah baris yang sama sebelum dilakukan proses *insert*.

3. Layer 3 - Count Rows Data Source (GP vs BQ):
   Lapisan validasi terakhir adalah memeriksa jumlah baris dari data sumber yang digunakan dalam fungsi. Hal ini penting untuk memastikan bahwa data mentah yang digunakan sebagai input oleh fungsi sudah konsisten di kedua *platform*, sehingga tidak menimbulkan perbedaan akibat perbedaan data sumber. Contoh *query* yang digunakan untuk proses ini dapat dilihat pada Kode 3.6.

```sql
-- jika tablenya non partition
select count(*) from dataset.table

-- jika table partition daily
select count(*) from dataset.table
WHERE partition_key = 'tanggal invoke sebelumnya'

-- jika table partition monthly
select count(*) from dataset.table
WHERE date_trunc(partition_key, month) = 'tanggal invoke
  sebelumnya'
```

Kode 3.6: Query cek isi data table source dengan tanggal spesifik

Penting untuk dicatat bahwa validasi bukan sekadar formalitas akhir, melainkan tahap penting dalam menjamin keberhasilan migrasi. Validasi yang menyeluruh memastikan bahwa setiap fungsi tidak hanya dapat dieksekusi, tetapi juga menghasilkan *output* yang sesuai dengan kebutuhan bisnis secara konsisten dan akurat. Dengan demikian, proses migrasi tidak hanya tuntas secara teknis, tetapi juga telah melalui uji kualitas yang ketat.

3.4 Spesifikasi Sistem

Berikut adalah perangkat lunak dan perangkat keras yang digunakan sebagai penunjang dalam pelaksanaan magang serta pengerjaan proyek migrasi fungsi dari Greenplum ke BigQuery. Perangkat lunak yang digunakan selama proses pengerjaan proyek meliputi:

1. Google Cloud Console
2. Google Drive
3. Google Meet
4. Google Docs
5. Zerotier
6. DBeavear

Spesifikasi perangkat keras yang digunakan adalah sebagai berikut:

1. Operating System: Windows 10 Pro 64-bit
2. Processor: AMD Ryzen 5 3600 6-Core Processor
3. GPU: NVIDIA GTX 1650
4. RAM: 16 GB DDR5
5. Storage: 1 TB HDD, 128 TB SSD

3.5 Kendala dan Solusi

Dalam proses migrasi fungsi dari Greenplum ke BigQuery, terdapat berbagai tantangan teknis yang dihadapi. Salah satu kesulitan utama yang dirasakan adalah jumlah fungsi yang sangat banyak, masing-masing dengan tingkat kompleksitas yang berbeda-beda. Tidak jarang, sebuah fungsi memiliki query yang sangat panjang dan kompleks, bahkan mencapai ribuan baris kode SQL. Hal ini tidak hanya menyulitkan proses pemahaman struktur logika fungsi, tetapi juga memperbesar potensi terjadinya kesalahan saat proses konversi dilakukan secara langsung. Kompleksitas lainnya muncul dari banyaknya dependensi terhadap table-table antara fungsi satu dengan lainnya. Struktur yang saling bergantung ini menuntut perhatian ekstra dalam memastikan bahwa setiap bagian fungsi dapat berjalan dengan baik tanpa mengganggu fungsi lain yang terkait.

Untuk mengatasi tantangan tersebut, solusi yang diterapkan adalah dengan melakukan proses konversi secara bertahap atau sepotong-sepotong. Artinya, query dalam fungsi yang kompleks dibagi menjadi beberapa bagian kecil yang lebih mudah dianalisis dan ditransformasikan. Pendekatan ini memungkinkan tim untuk lebih fokus dalam memahami dan menerjemahkan setiap logika bisnis yang ada, serta mempermudah proses debugging apabila terjadi error. Selain itu, dengan memecah fungsi menjadi bagian-bagian lebih kecil, proses validasi juga dapat dilakukan secara lebih terstruktur. Setiap bagian yang telah berhasil dikonversi dan divalidasi dapat dijadikan fondasi untuk melanjutkan konversi bagian selanjutnya. Pendekatan ini terbukti efektif dalam meningkatkan akurasi konversi serta mempercepat proses migrasi secara keseluruhan.