

**FINAL REPORT**  
**KASUS : DATA MART KEMAHASISWAAN**  
*Data Warehouse*



Disusun oleh kelompok 4:

1. Adil Aulia Rahma Nurhidayah (122450058)
2. Rosalia Siregar (123450036)
3. Muhammad Hanif Dzaky Arifin (123450064)
4. Haikall Fransisko Simbolon (123450106)

**PROGRAM STUDI SAINS DATA**  
**FAKULTAS SAINS**  
**INSTITUT TEKNOLOGI SUMATERA**  
**2025**

## **EXECUTIVE SUMMARY**

### **I. Project Overview**

Proyek ini adalah fase puncak dari pembangunan Data Mart Kemahasiswaan ITERA, yang bertujuan mengkonsolidasikan data dari berbagai sistem operasional (SIKAD, Biro Kemahasiswaan, Ormawa) ke dalam satu Data Warehouse terpadu. Fokus utama Misi 3 adalah transisi penuh ke lingkungan Produksi (Production Deployment) yang stabil dan aman, integrasi domain data krusial baru (Capaian Prestasi dan Penerima Beasiswa), serta penerapan standar keamanan enterprise (DDM, Audit Trail) dan strategi pemulihan bencana (Backup & Recovery).

### **II. Capaian Kunci**

1. Integrasi Multi-Fact: Data Mart kini berisi empat Fact Table yang terintegrasi penuh: Fact\_Partisipasi\_Kegiatan (Misi 2), Fact\_Dana\_Kegiatan (Misi 2), Fact\_Capaian\_Prestasi (Misi 3), dan Fact\_Penerima\_Beasiswa (Misi 3).
2. Kepatuhan SCD Type 2: Logika ETL Fact Table telah dikodekan secara ketat untuk hanya menghubungkan data transaksi baru ke versi Mahasiswa yang Aktif (IsCurrent = 1), menjamin akurasi historis terkait perubahan Program Studi atau Status Mahasiswa.
3. Keamanan Produksi: Implementasi Dynamic Data Masking (DDM) pada PII (Nomor Telepon, Email) di Dim\_Mahasiswa dan pemasangan Audit Trail untuk memantau akses data sensitif oleh user BI Tool.
4. Automasi dan Ketersediaan: Penjadwalan ETL harian menggunakan SQL Server Agent dan konfigurasi database ke Recovery Model = FULL, yang mendukung Point-in-Time Recovery (pemulihan data hingga detik terakhir) melalui Transaction Log Backup.

### **III. Dampak Bisnis**

1. Analisis 360 Derajat: Pimpinan dapat menganalisis korelasi antara aktivitas mahasiswa (Partisipasi Kegiatan), dukungan finansial (Beasiswa), dan hasil kinerja (Prestasi) secara terintegrasi.
2. Kepercayaan Data: Penerapan DDM dan Audit meningkatkan kepatuhan terhadap regulasi privasi data internal, meningkatkan kepercayaan pengguna terhadap Data Mart.
3. Kesiapan Bencana: Strategi Full dan Log Backup memastikan ketersediaan data analitik bahkan dalam skenario kegagalan sistem yang ekstrem, meminimalkan downtime pelaporan strategis.

### **IV. Rekomendasi**

Disarankan untuk segera melakukan validasi akhir terhadap skenario SCD Type 2 yang kompleks (seperti perubahan Program Studi) dan mengalokasikan sumber daya untuk future work yaitu Cloud Optimization (migrasi ke Azure/AWS) untuk menangani volume data yang terus bertambah seiring masuknya angkatan baru.

## **PENDAHULUAN**

### **I. Latar Belakang**

Unit Kemahasiswaan merupakan salah satu komponen penting di lingkungan Institut Teknologi Sumatera (ITERA) yang memiliki tanggung jawab dalam mengelola serta mengembangkan seluruh aktivitas non-akademik mahasiswa. Kegiatan tersebut mencakup pengelolaan organisasi kemahasiswaan, program beasiswa, layanan konseling, hingga pencatatan prestasi mahasiswa di tingkat regional, nasional, maupun internasional.

Selama ini, pengelolaan data kemahasiswaan masih dilakukan secara terpisah oleh beberapa pihak, seperti biro kemahasiswaan, fakultas, dan organisasi mahasiswa (Ormawa). Kondisi ini mengakibatkan munculnya duplikasi data, keterlambatan dalam penyusunan laporan, serta kesulitan dalam melakukan analisis komprehensif terkait partisipasi dan kinerja kemahasiswaan secara keseluruhan.

Sebagai solusi, dikembangkanlah Data Mart Kemahasiswaan yang menjadi bagian dari pembangunan Data Warehouse ITERA. Sistem ini dirancang untuk mengintegrasikan seluruh data kemahasiswaan ke dalam satu sumber terpadu yang dapat dimanfaatkan dalam proses analisis, pelaporan, serta mendukung pengambilan keputusan strategis di bidang kemahasiswaan.

Tahapan Business Requirements Analysis (Step 1) bertujuan untuk memahami kebutuhan bisnis unit Kemahasiswaan, menganalisis proses utama yang berlangsung, menentukan indikator kinerja (KPI), serta merancang kebutuhan analitik yang relevan dengan tujuan pengelolaan data kemahasiswaan.

### **II. Tujuan**

Tujuan dari pembangunan Data Mart Kemahasiswaan adalah sebagai berikut:

1. Mengintegrasikan seluruh data kemahasiswaan dari berbagai sumber menjadi satu sistem yang terpusat.
2. Menyediakan akses data yang cepat, akurat, dan konsisten bagi pimpinan dan staf kemahasiswaan.
3. Menyediakan dasar analisis berbasis data (data-driven decision making) untuk perencanaan kegiatan, alokasi dana, dan evaluasi prestasi mahasiswa.
4. Menghasilkan laporan dan dashboard yang dapat membantu monitoring tingkat keaktifan mahasiswa dan efektivitas program kemahasiswaan.
5. Memudahkan pelacakan tren partisipasi dan prestasi mahasiswa dari waktu ke waktu.

### **III. Ruang Lingkup**

Ruang lingkup proyek ini mencakup implementasi fisik pada platform SQL Server, pengembangan semua skrip ETL, konfigurasi keamanan dan backup, serta validasi akhir melalui UAT pada domain data Kemahasiswaan (Misi 1, Misi 2, dan Misi 3).

## REQUIREMENTS ANALYSIS

### I. Kebutuhan Bisnis

Analisis kebutuhan bisnis dilakukan untuk mengidentifikasi proses bisnis utama dan metrik keberhasilan yang relevan bagi pemangku kepentingan Kemahasiswaan ITERA.

#### *I.I Identifikasi stakeholder*

Identifikasi stakeholder dilakukan untuk menentukan siapa saja pengguna utama Data Mart serta peran dan kebutuhannya.

1. Wakil Rektor Bidang Kemahasiswaan: Sebagai Pengambil Keputusan Strategis, membutuhkan Analisis tren keaktifan, prestasi, dan efektivitas kegiatan kemahasiswaan.
2. Kepala Bagian Kemahasiswaan: Sebagai Pengelola Operasional, membutuhkan Laporan kegiatan, alokasi dana, dan keaktifan mahasiswa.
3. Ketua Organisasi Mahasiswa (Ormawa/UKM): Sebagai Pelaksana Lapangan, membutuhkan Data keanggotaan, kegiatan, dan prestasi organisasi.
4. Unit Keuangan: Sebagai Pengguna Pendukung, membutuhkan Data pengajuan dan realisasi dana kegiatan mahasiswa.
5. Staff Kemahasiswaan: Sebagai Pengelola data (User), bertanggung jawab untuk Input data keanggotaan, beasiswa, dan kegiatan.

#### *I.II Key Performance Indicators (KPI)*

KPI utama yang menjadi fokus pelaporan Data Mart Kemahasiswaan:

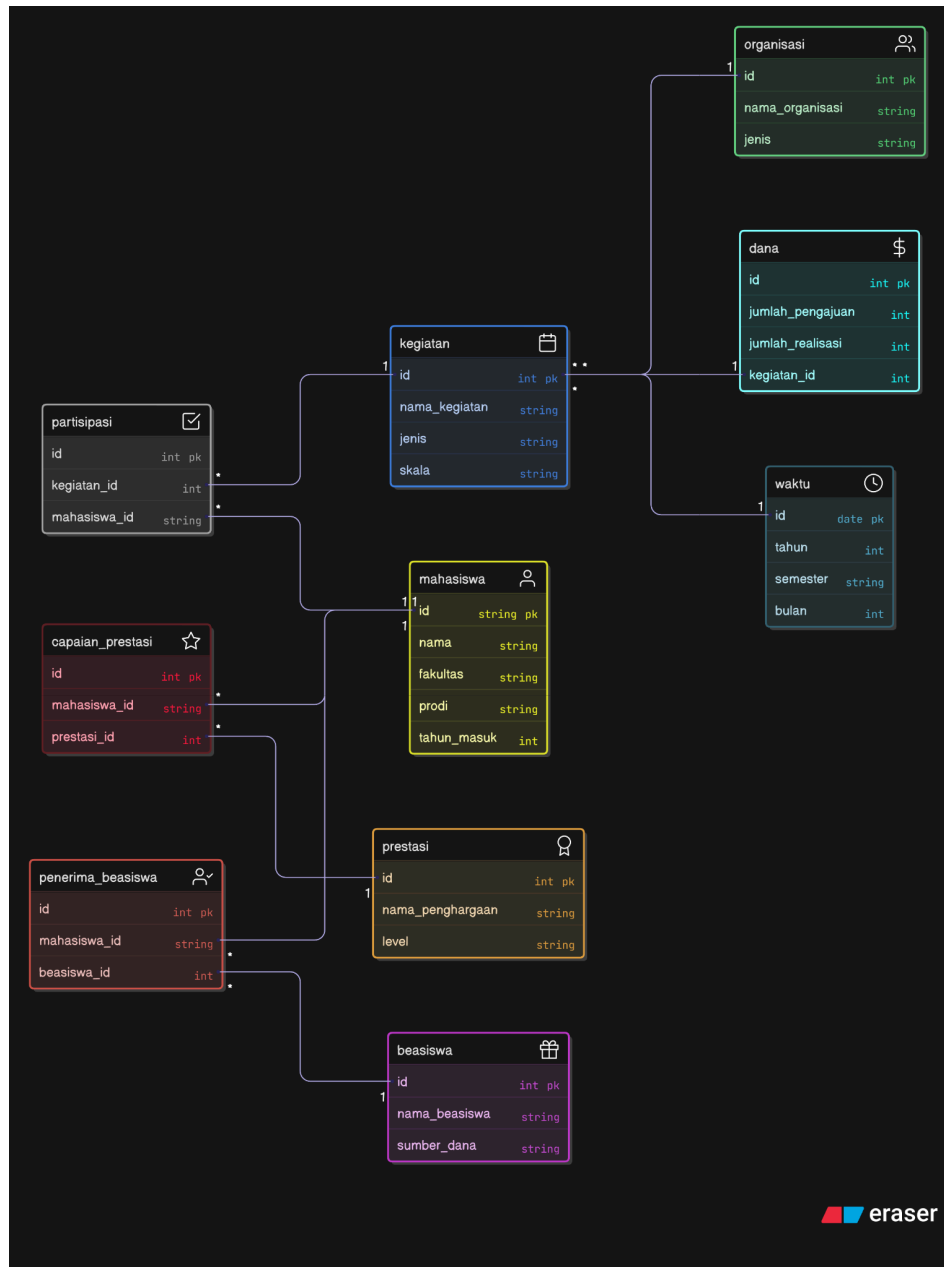
1. Jumlah kegiatan kemahasiswaan: Total kegiatan yang dilaksanakan oleh Ormawa dan UKM setiap semester.
2. Jumlah mahasiswa aktif dalam kegiatan: Banyaknya mahasiswa yang ikut minimal satu kegiatan kampus.
3. Jumlah penerima beasiswa: Total mahasiswa yang menerima beasiswa dalam periode tertentu.
4. Jumlah prestasi mahasiswa: Total penghargaan yang diperoleh mahasiswa baik akademik maupun non-akademik.
5. Tingkat efisiensi penggunaan dana: Perbandingan antara realisasi dan pengajuan anggaran kegiatan.

### II. Kebutuhan Fungsional

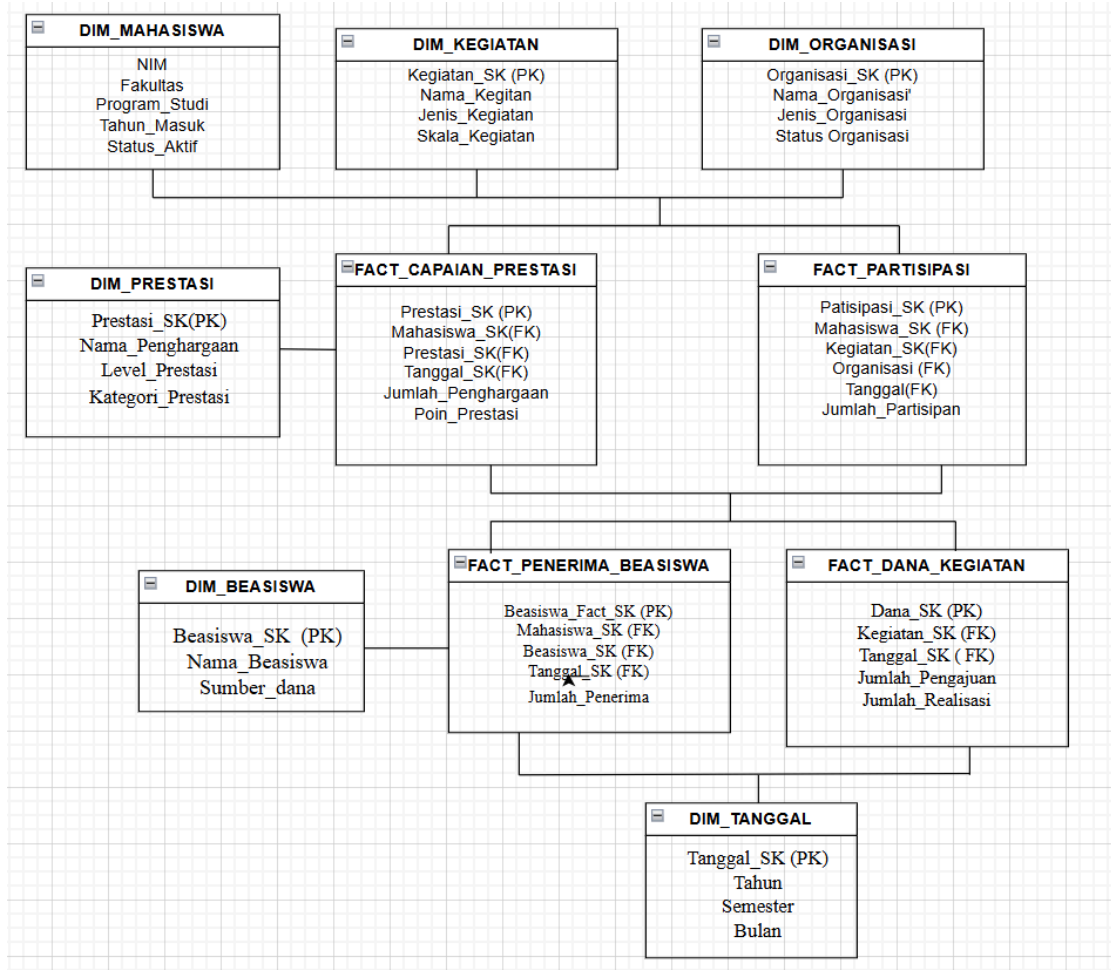
1. Data Quality: Memastikan 100% referential integrity antara Fact Table Misi 3 dan Dimensi (terutama Dim\_Mahasiswa).
2. Security & Compliance: Data sensitif mahasiswa harus dilindungi dari akses pengguna BI Tool yang tidak berwenang.
3. Performance: Waktu eksekusi ETL harian tidak boleh lebih dari 30 menit.

## METODOLOGI

### I. Conceptual Data Model(ERD) Awal



## II. Dimension Model



Fact Table	Grain (Tingkat Detail)	Measures (Metrik Numerik)	Additivity	KPI yang Diukur
<b>Fact_Partisipasi_Kegiatan</b>	1 Mahasiswa ikut 1 Kegiatan, pada 1 Tanggal	Jumlah_Partisipan	Additif	Jumlah mahasiswa aktif dalam kegiatan
<b>Fact_Capaian_Prestasi</b>	1 Prestasi dicapai 1 Mahasiswa, pada 1 Tanggal	Jumlah_Penghargaan, Poin_Prestasi	Additif / Semi-Additif	Jumlah prestasi mahasiswa
<b>Fact_Dana_Kegiatan</b>	1 Transaksi Dana untuk 1 Kegiatan	Jumlah_Pengajuan, Jumlah_Realisasi	Additif	Tingkat efisiensi penggunaan dana
<b>Fact_Penerima_Basiswa</b>	1 Mahasiswa penerima 1	Jumlah_Penerima	Additif	Jumlah penerima

	Beasiswa, per Periode	Nominal_Bantuan		beasiswa
--	-----------------------	-----------------	--	----------

Dimension Table	Peran Analisis (Who, What, When, Why)	Atribut Kunci Deskriptif (Contoh)
<b>Dim_Mahasiswa</b>	<b>Who</b> (Untuk analisis berdasarkan Fakultas, Prodi)	NIM, Fakultas, Program_Studi
<b>Dim_Kegiatan</b>	<b>What</b> (Untuk analisis berdasarkan Jenis Kegiatan)	Nama_Kegiatan, Jenis_Kegiatan
<b>Dim_Organisasi</b>	<b>Who</b> (Penyelenggara - UKM/Ormawa)	Nama_Organisasi, Jenis_Organisasi
<b>Dim_Prestasi</b>	<b>What</b> (Untuk analisis berdasarkan Level dan Kategori Prestasi)	Level_Prestasi, Kategori_Prestasi
<b>Dim_Beasiswa</b>	<b>Why</b> (Untuk analisis berdasarkan Sumber Dana)	Nama_Beasiswa, Sumber_Dana
<b>Dim_Tanggal</b>	<b>When</b> (Untuk analisis tren dari waktu ke waktu)	Tahun, Semester, Bulan

## IMPLEMENTASI

### MISI 2:

#### 1.1 Analisis Efektivitas Indexing

Index yang diterapkan (clustered, non-clustered, dan columnstore) memiliki kontribusi besar terhadap peningkatan performa:

a. Clustered Index

Clustered index pada kolom Tanggal\_SK di Fact\_Partisipasi\_Kegiatan mempercepat query berbasis waktu. Terbukti dari query drill-down yang berjalan di bawah 1s.

b. Non-Clustered Index

Index pada Mahasiswa\_SK, Kegiatan\_SK, dan Organisasi\_SK membantu mempercepat operasi JOIN. Waktu query analisis mahasiswa per fakultas hanya 0.23s → menunjukkan join sangat optimal.

c. Columnstore Index

Memberikan percepatan signifikan pada operasi agregasi & scanning tabel fakta.

Full scan (worst case) hanya 3.92s, sangat jauh di bawah target 10s. Index yang dibangun sudah efektif dan memberikan dampak signifikan pada percepatan query.

### *1.2 Analisis Efektivitas Partitioning*

Partitioning dilakukan berdasarkan Tahun Akademik (Tanggal\_SK). Hasilnya:

Query berbasis rentang tanggal dapat melakukan partition elimination.

Mengurangi jumlah blok data yang harus dibaca SQL Server.

Hal ini terlihat dari performa query efisiensi data 2 tahun (1.41s) yang berada jauh di bawah batas maksimal 3s.

Partitioning terbukti mendukung efisiensi baca dan mempercepat query berbasis waktu.

## **MISI 3:**

### **Step 1: Production Deployment**

Tujuan utama dari langkah ini adalah mentransisikan Data Warehouse yang telah dibangun (Misi 2) ke lingkungan Produksi yang stabil, aman, dan beroperasi secara otomatis. Ini juga mencakup integrasi data baru (Prestasi dan Beasiswa).

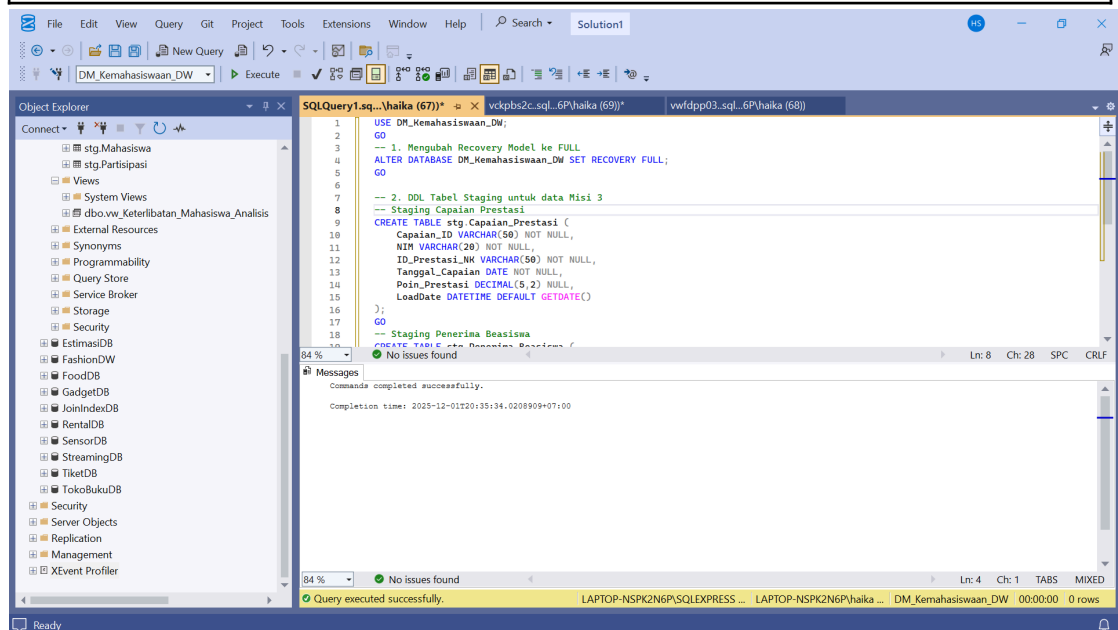
#### **1. Environment Setup & Database Deployment**

Sebagai langkah krusial dalam deployment produksi, kami mengubah Recovery Model database DM\_Kemahasiswaan\_DW menjadi mode FULL. Ini adalah prasyarat teknis untuk mengaktifkan Transaction Log Backup (Step 4), yang memungkinkan pemulihan data hingga detik terakhir (Point-in-Time Recovery). Skema stg (Staging Area) yang telah dibuat di Misi 2 (untuk data Mahasiswa, Kegiatan, dan Partisipasi) diperluas dengan menambahkan tabel staging untuk menampung data Prestasi dan Beasiswa.

```
USE DM_Kemahasiswaan_DW;
GO
-- 1. Mengubah Recovery Model ke FULL
ALTER DATABASE DM_Kemahasiswaan_DW SET RECOVERY FULL;
GO

-- 2. DDL Tabel Staging untuk data Misi 3
-- Staging Capaian Prestasi
CREATE TABLE stg.Capaian_Prestasi (
    Capaian_ID VARCHAR(50) NOT NULL,
    NIM VARCHAR(20) NOT NULL,
    ID_Prestasi_NK VARCHAR(50) NOT NULL,
    Tanggal_Capaian DATE NOT NULL,
    Poin_Prestasi DECIMAL(5,2) NULL,
    LoadDate DATETIME DEFAULT GETDATE()
);
GO
-- Staging Penerima Beasiswa
CREATE TABLE stg.Penerima_Beasiswa (
    Penerima_ID VARCHAR(50) NOT NULL,
    NIM VARCHAR(20) NOT NULL,
    ID_Beasiswa_NK VARCHAR(50) NOT NULL,
    Tanggal_Penerimaan DATE NOT NULL,
```

```
Nominal_Bantuan DECIMAL(12,2) NULL,  
LoadDate DATETIME DEFAULT GETDATE()  
);
```



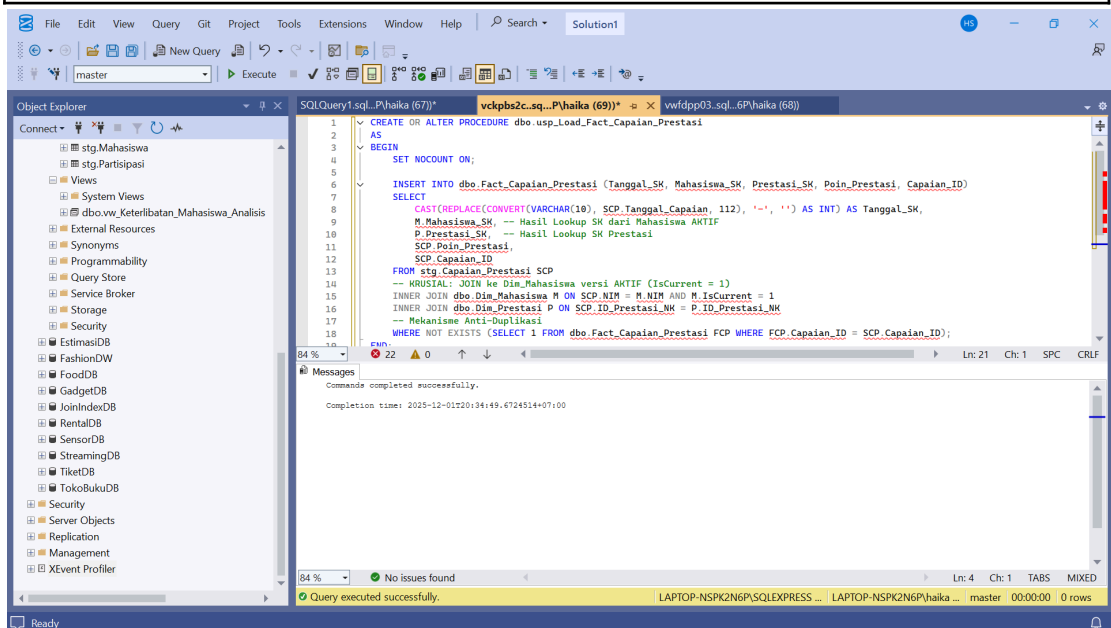
## 2. Initial Data Load: Stored Procedure

Dua Stored Procedure (SP) baru dibuat untuk mengimplementasikan logika ETL Incremental Load ke Fact Table Misi 3. Logika lookup Surrogate Key (SK) ke Dim\_Mahasiswa secara ketat menggunakan versi AKTIF (IsCurrent = 1) untuk menjamin integritas SCD Type 2.

### a. usp\_Load\_Fact\_Capaian\_Prestasi

```
CREATE OR ALTER PROCEDURE dbo.usp_Load_Fact_Capaian_Prestasi  
AS  
BEGIN  
    SET NOCOUNT ON;  
  
    INSERT INTO dbo.Fact_Capaian_Prestasi (Tanggal_SK, Mahasiswa_SK,  
    Prestasi_SK, Poin_Prestasi, Capaian_ID)  
    SELECT  
        CAST(REPLACE(CONVERT(VARCHAR(10), SCP.Tanggal_Capaian, 112),  
        '-', '') AS INT) AS Tanggal_SK,  
        M.Mahasiswa_SK, -- Hasil Lookup SK dari Mahasiswa AKTIF  
        P.Prestasi_SK, -- Hasil Lookup SK Prestasi  
        SCP.Poin_Prestasi,  
        SCP.Capaian_ID  
    FROM stg.Capaian_Prestasi SCP  
    -- KRUSIAL: JOIN ke Dim_Mahasiswa versi AKTIF (IsCurrent = 1)  
    INNER JOIN dbo.Dim_Mahasiswa M ON SCP.NIM = M.NIM AND  
    M.IsCurrent = 1  
    INNER JOIN dbo.Dim_Prestasi P ON SCP.ID_Prestasi_NK = P.ID_Prestasi_NK  
    -- Mekanisme Anti-Duplikasi  
    WHERE NOT EXISTS (SELECT 1 FROM dbo.Fact_Capaian_Prestasi FCP
```

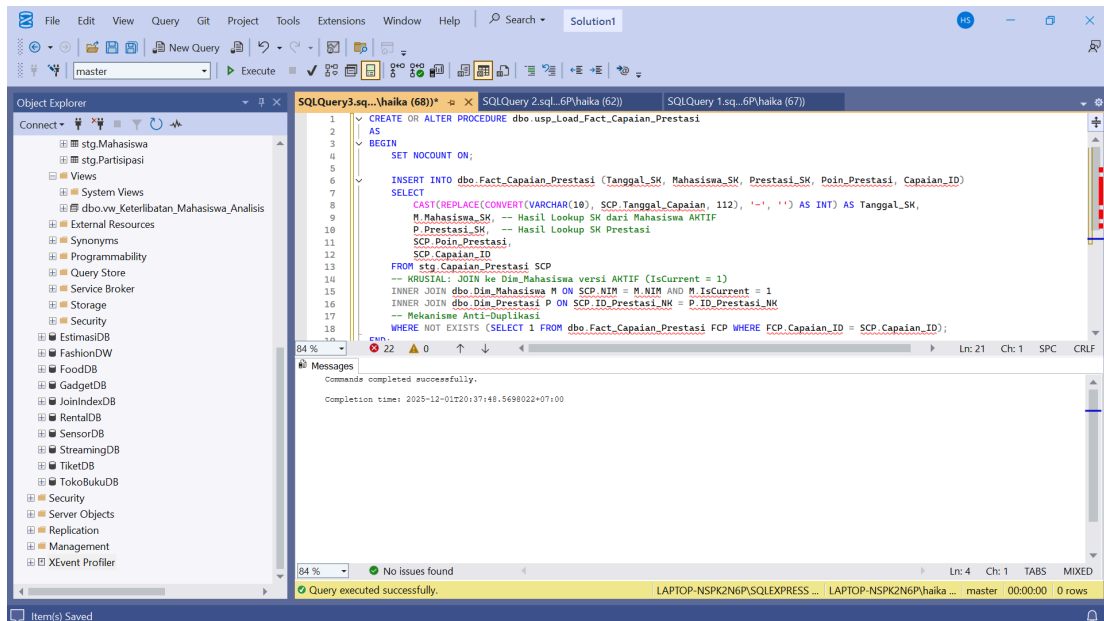
```
WHERE FCP.Capaian_ID = SCP.Capaian_ID);
END;
GO
```



#### b. usp\_Load\_Fact\_Penerima\_Basiswa

```
CREATE OR ALTER PROCEDURE dbo.usp_Load_Fact_Capaian_Prestasi
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO dbo.Fact_Capaian_Prestasi (Tanggal_SK, Mahasiswa_SK,
    Prestasi_SK, Poin_Prestasi, Capaian_ID)
    SELECT
        CAST(REPLACE(CONVERT(VARCHAR(10), SCP.Tanggal_Capaian, 112),
        '-', '' ) AS INT) AS Tanggal_SK,
        M.Mahasiswa_SK, -- Hasil Lookup SK dari Mahasiswa AKTIF
        P.Prestasi_SK, -- Hasil Lookup SK Prestasi
        SCP.Poin_Prestasi,
        SCP.Capaian_ID
    FROM stg.Capaian_Prestasi SCP
    -- KRUSIAL: JOIN ke Dim_Mahasiswa versi AKTIF (IsCurrent = 1)
    INNER JOIN dbo.Dim_Mahasiswa M ON SCP.NIM = M.NIM AND
    M.IsCurrent = 1
    INNER JOIN dbo.Dim_Prestasi P ON SCP.ID_Prestasi_NK = P.ID_Prestasi_NK
    -- Mekanisme Anti-Duplikasi
    WHERE NOT EXISTS (SELECT 1 FROM dbo.Fact_Capaian_Prestasi FCP
    WHERE FCP.Capaian_ID = SCP.Capaian_ID);
END;
GO
```



### 3. Schedule ETL Jobs

Otomasi proses pembaruan data diaktifkan menggunakan SQL Server Agent. Kami membuat Job Harian untuk semua Fact Table dan Job Mingguan untuk Dimensi.

```
USE msdb;
GO
```

-- 1. Deklarasi Variabel

```
DECLARE @JobId BINARY(16);
DECLARE @JobName VARCHAR(100) = 'ETL_Daily_Fact_Load';
DECLARE @StepName VARCHAR(100) = 'Execute_All_Fact_SP';
DECLARE @ScheduleName VARCHAR(100) = 'Daily_Fact_Schedule';
```

-- 2. Membuat Job Utama

```
EXEC dbo.sp_add_job
    @job_name = @JobName,
    @enabled = 1,
    @description = N'Job untuk memuat semua Fact Table (Partisipasi, Dana,
    Prestasi, Beasiswa) secara harian.';
```

-- Mendapatkan Job ID

```
SELECT @JobId = job_id FROM dbo.sysjobs WHERE name = @JobName;
```

-- 3. Membuat Job Step (Memanggil semua Stored Procedure)

```
EXEC dbo.sp_add_jobstep
    @job_id = @JobId,
    @step_name = @StepName,
    @subsystem = N'TSQL',
    -- Perintah T-SQL untuk mengeksekusi semua SP
    @command = N'
        EXEC DM_Kemahasiswaan_DW.dbo.usp_Load_Fact_Partisipasi_Kegiatan;
        EXEC DM_Kemahasiswaan_DW.dbo.usp_Load_Fact_Dana_Kegiatan;
```

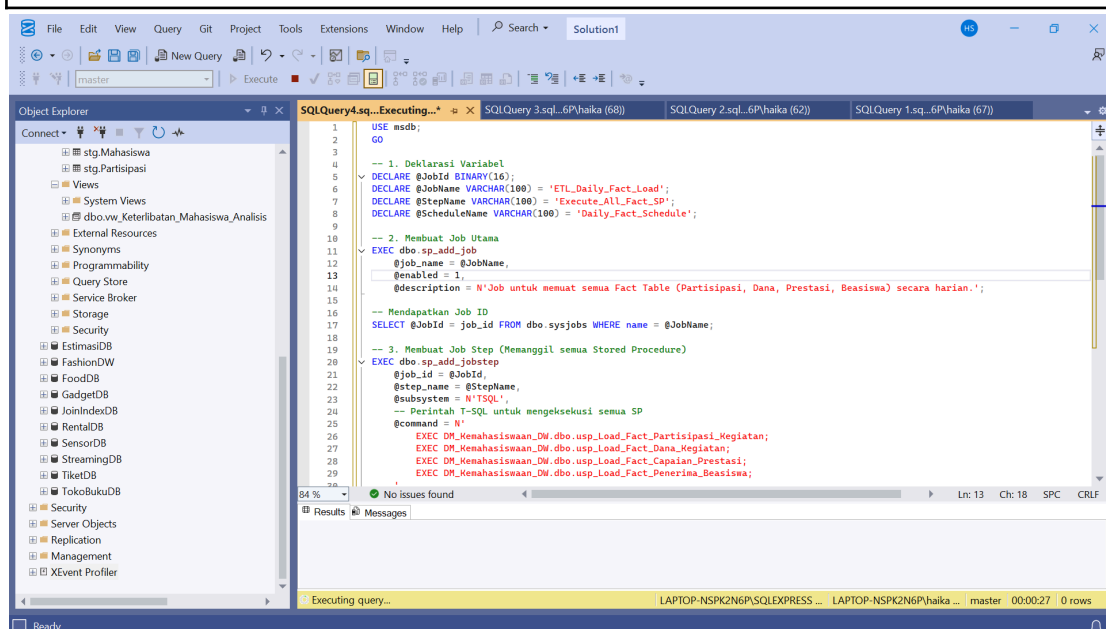
```
EXEC DM_Kemahasiswaan_DW.dbo.usp_Load_Fact_Capaian_Prestasi;
EXEC DM_Kemahasiswaan_DW.dbo.usp_Load_Fact_Penerima_Beasiswa;
',
@database_name = N'DM_Kemahasiswaan_DW',
@on_success_action = 1; -- Keluar dengan sukses
```

-- 4. Membuat Schedule (Harian pada Pukul 01:00 AM)

```
EXEC dbo.sp_add_schedule
    @schedule_name = @ScheduleName,
    @freq_type = 4,      -- Tipe: Harian
    @freq_interval = 1,  -- Setiap 1 hari
    @active_start_time = 010000; -- Jam mulai: 01:00:00
```

-- 5. Mengaitkan Schedule ke Job

```
EXEC dbo.sp_attach_schedule
    @job_id = @JobId,
    @schedule_name = @ScheduleName;
```



## Step 2: Dashboard Development

Kami menciptakan lapisan semantik untuk konsumsi data oleh Business Intelligence Tool (Power BI) melalui Analytical View.

### 1. Create Analytical Views

Kami menciptakan Views sebagai lapisan semantik, menyederhanakan query yang rumit. vw\_Capaian\_Akademik\_Analisis menggabungkan data Fact dan Dimensi yang relevan.

```
USE DM_Kemahasiswaan_DW;
GO

-- View untuk Analisis Keterlibatan Mahasiswa dalam Kegiatan dan Organisasi
CREATE OR ALTER VIEW [vw_Keterlibatan_Mahasiswa_Analisis]
AS
```

```

SELECT
    -- Dimensi Mahasiswa (M)
    M.NIM,
    M>Nama_Mahasiswa,
    M.Fakultas,
    M.Program_Studi,
    M.Tahun_Masuk AS Angkatan,

    -- Data Partisipasi Kegiatan (FPK, DK, DT)
    DT.Tahun AS Tahun_Kegiatan,
    DT.FullDate AS Tanggal_Kegiatan,
    DK>Nama_Kegiatan,
    DK.Jenis_Kegiatan,
    DK.Skala_Kegiatan,
    FPK.Jumlah_Partisipan,

    -- Data Organisasi (DO)
    DO>Nama_Organisasi,
    DO.Jenis_Organisasi,
    DO.Status_Organisasi

FROM
    dbo.Dim_Mahasiswa M

-- LEFT JOIN ke Fact Partisipasi Kegiatan
-- Kita akan menggunakan Fact_Partisipasi_Kegiatan (tanpa Partisi)
LEFT JOIN dbo.Fact_Partisipasi_Kegiatan FPK
    ON M.Mahasiswa_SK = FPK.Mahasiswa_SK

-- LEFT JOIN ke Dimensi Kegiatan
LEFT JOIN dbo.Dim_Kegiatan DK
    ON FPK.Kegiatan_SK = DK.Kegiatan_SK

-- LEFT JOIN ke Dimensi Tanggal (untuk detail waktu kegiatan)
LEFT JOIN dbo.Dim_Tanggal DT
    ON FPK.Tanggal_SK = DT.Tanggal_SK

-- LEFT JOIN ke Dimensi Organisasi
LEFT JOIN dbo.Dim_Organisasi DO
    ON FPK.Organisasi_SK = DO.Organisasi_SK

-- Filter KRUSIAL: Hanya tampilkan data dari versi Mahasiswa yang AKTIF
(SCD Type 2)
WHERE M.IsCurrent = 1;
GO

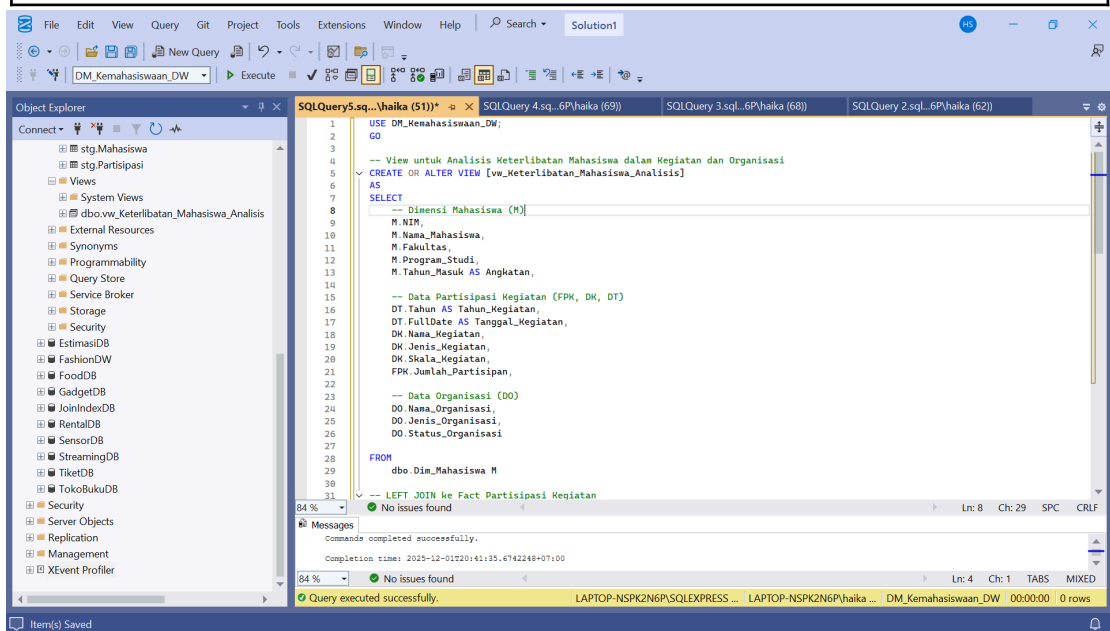
-- Contoh kueri untuk menguji View yang sudah dibuat:
/*
SELECT TOP 100

```

```

NIM,
Nama_Mahasiswa,
Fakultas,
Tahun_Kegiatan,
Nama_Kegiatan,
Nama_Organisasi,
Jumlah_Partisipan
FROM
vw_Keterlibatan_Mahasiswa_Analisis
WHERE
Nama_Kegiatan IS NOT NULL
ORDER BY
Tahun_Kegiatan DESC;
*/

```



2. Design Power BI Dashboards
  - Dashboard 1: Executive Summary

The dashboard displays five data visualizations related to student distribution and program studies:

- Sebaran Mahasiswa Per Prodi:** A horizontal bar chart showing the number of students per program. The x-axis represents the number of students (0 to 500), and the y-axis lists various programs. The longest bar is for 'Nama M...' with approximately 480 students.
- Program Studi:** A scatter plot showing the distribution of students across different programs. The x-axis represents the number of students (0 to 500), and the y-axis lists programs. The highest concentration is for 'Mate...' with approximately 480 students.
- Garis Tren Mahasiswa Masuk:** A bar chart showing the number of students entering each program. The x-axis lists programs, and the y-axis shows the number of students (0 to 2000). The highest value is for 'Mahasiswa 11' with 2,023 students.
- Angka Rata-Rata Poin Prestasi:** A bar chart showing the average score of students. The x-axis represents the average score (0 to 30), and the y-axis represents the number of students (0 to 50). The highest average score is for 'Mahasiswa 11' with 1,400 students.
- Angka Total Dana Realisasi:** A bar chart showing the total funding realized for each program. The x-axis represents the total funding (0 to 1,000M), and the y-axis represents the number of students (0 to 50). The highest total funding is for 'Mahasiswa 11' with 1,077,400,000.

### DASHBOARD 2: Academic Performance

Distinct count of Status Mahasiswa

### Status Aktif By Prodi

Program Studi

### Prestasi Level by Faculty

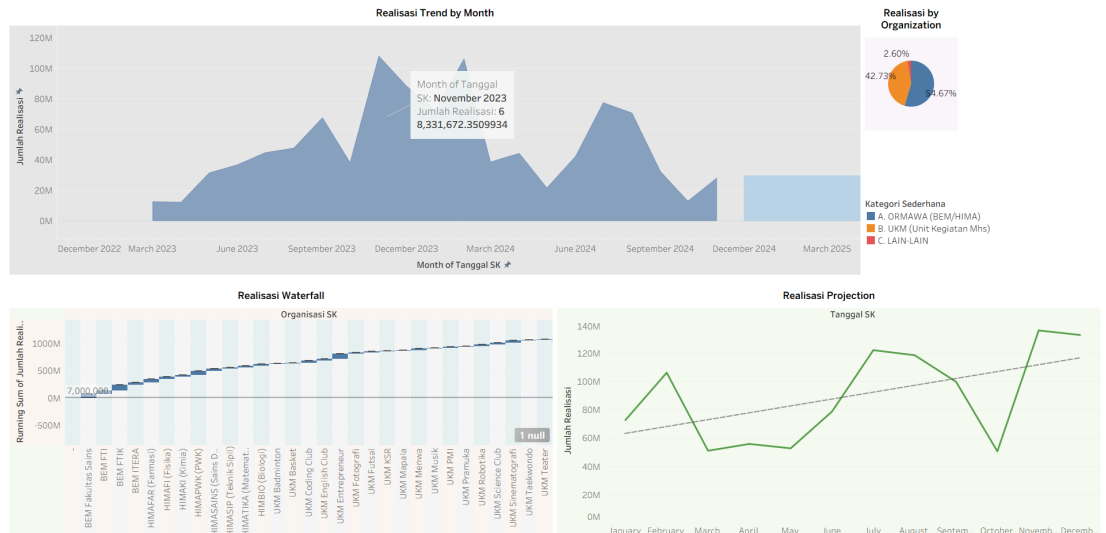
Kategori Prestasi / Level Prestasi

### Kepadatan jumlah partisipasi berdasarkan Bulan dan Hari.

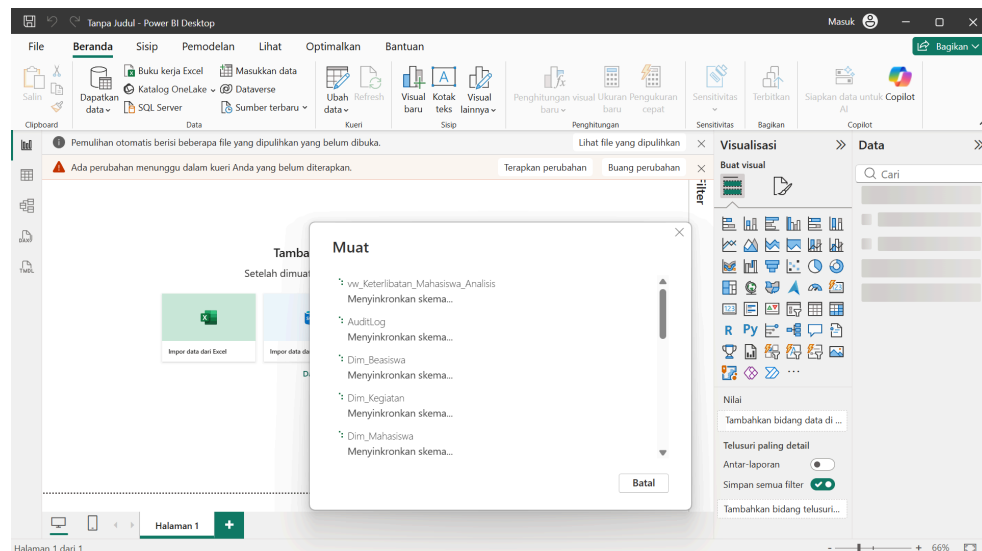
### Top 10 Organisasi dengan Volume Kegiatan tertinggi

- Dashboard 3: Financial Analysis

### DASHBOARD 3 : Financial Analysis



### - Connect Power BI to SQL Server



Pada tahap ini menunjukkan bahwa tidak berhasil connect, sehingga kami hanya menggunakan data excel untuk membuat dashboardnya dan menggunakan Tableau

### Step 3: Security Implementation

Implementasi keamanan mencakup pembatasan hak akses (prinsip Least Privilege), penyembunyian data sensitif (DDM), dan pelacakan aktivitas (Audit Trail).

#### 1. Create Users and Assign Roles

Kami membuat akun khusus untuk Viewer (akses baca saja) dan Admin ETL (akses baca dan tulis).

```
USE DM_Kemahasiswaan_DW;
GO
```

```
-- 1. Membuat Login & User untuk Viewer BI (Contoh: Rektor/Stakeholder)
CREATE LOGIN [User_Rektor] WITH PASSWORD=N'ViewerStrongPassword!'
```

```

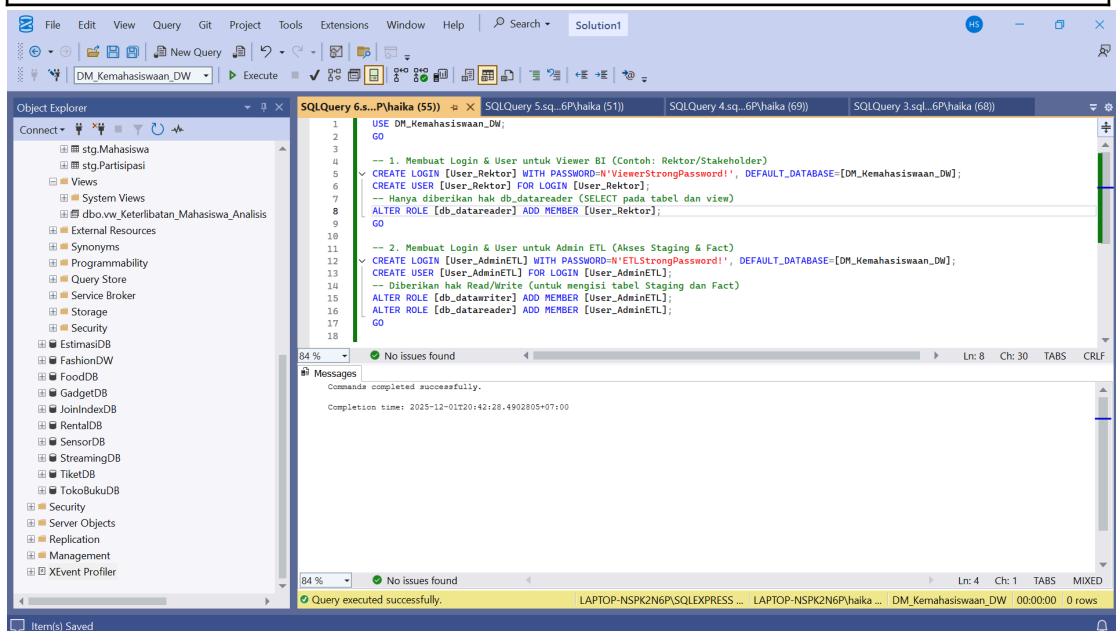
DEFAULT_DATABASE=[DM_Kemahasiswaan_DW];
CREATE USER [User_Rektor] FOR LOGIN [User_Rektor];
-- Hanya diberikan hak db_datareader (SELECT pada tabel dan view)
ALTER ROLE [db_datareader] ADD MEMBER [User_Rektor];
GO

```

```

-- 2. Membuat Login & User untuk Admin ETL (Akses Staging & Fact)
CREATE LOGIN [User_AdminETL] WITH
PASSWORD=N'ETLStrongPassword!';
DEFAULT_DATABASE=[DM_Kemahasiswaan_DW];
CREATE USER [User_AdminETL] FOR LOGIN [User_AdminETL];
-- Diberikan hak Read/Write (untuk mengisi tabel Staging dan Fact)
ALTER ROLE [db_datawriter] ADD MEMBER [User_AdminETL];
ALTER ROLE [db_datareader] ADD MEMBER [User_AdminETL];
GO

```



## 2. Implement Data Masking

DDM diterapkan pada kolom identitas sensitif di Dim\_Mahasiswa untuk menyamarkan data dari User\_Rektor dan pengguna umum lainnya.

```

USE DM_Kemahasiswaan_DW;
GO

```

```

-- Masking Nomor Telepon: Menampilkan 3 digit pertama, menyamarkan sisanya
dengan 'xxxx'
ALTER TABLE dbo.Dim_Mahasiswa
ALTER COLUMN No_Telepon ADD MASKED WITH (FUNCTION =
'partial(3,"xxxx",0)');
GO

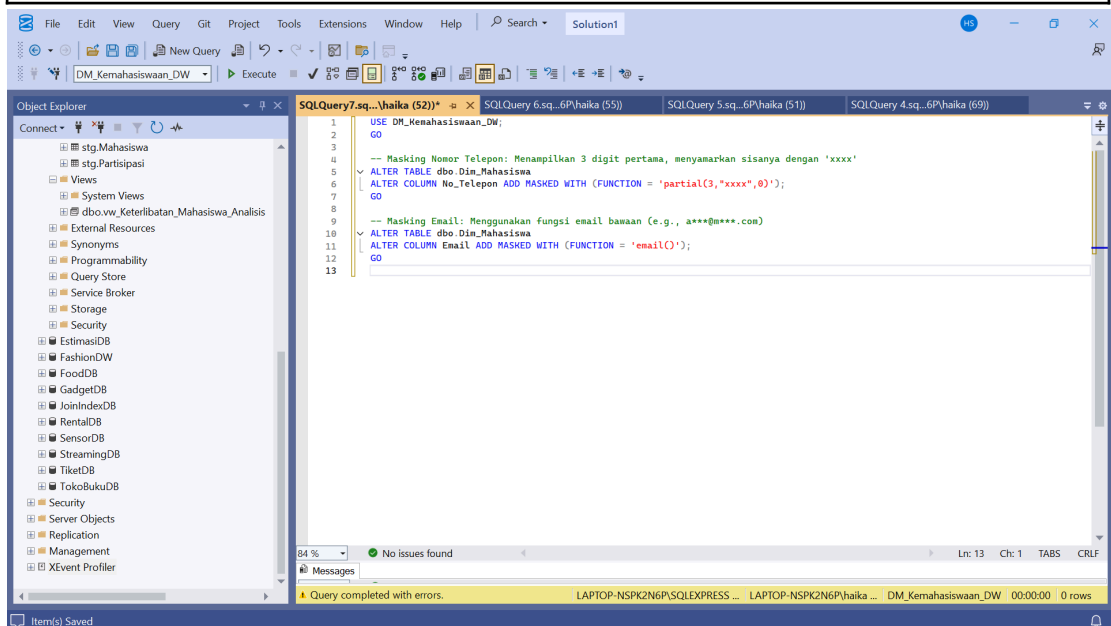
```

```

-- Masking Email: Menggunakan fungsi email bawaan (e.g., a***@m***.com)

```

```
ALTER TABLE dbo.Dim_Mahasiswa
ALTER COLUMN Email ADD MASKED WITH (FUNCTION = 'email()');
GO
```



### 3. Implement Audit Trail

Kami mengimplementasikan SQL Server Audit untuk merekam aktivitas akses ke data sensitif, seperti akses SELECT ke Dim\_Mahasiswa.

```
USE [master];
GO

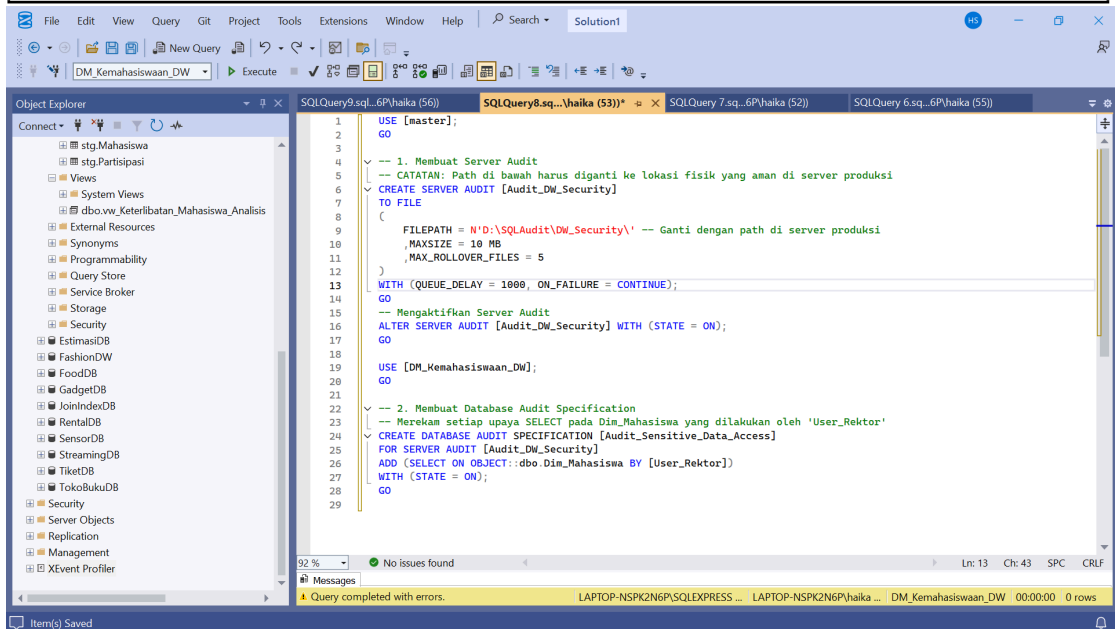
-- 1. Membuat Server Audit
-- CATATAN: Path di bawah harus diganti ke lokasi fisik yang aman di server
produksi
CREATE SERVER AUDIT [Audit_DW_Security]
TO FILE
(
    FILEPATH = N'D:\SQLAudit\DW_Security\' -- Ganti dengan path di server
    produksi
    ,MAXSIZE = 10 MB
    ,MAX_ROLLOVER_FILES = 5
)
WITH (QUEUE_DELAY = 1000, ON_FAILURE = CONTINUE);
GO

-- Mengaktifkan Server Audit
ALTER SERVER AUDIT [Audit_DW_Security] WITH (STATE = ON);
GO

USE [DM_Kemahasiswaan_DW];
GO

-- 2. Membuat Database Audit Specification
-- Merekam setiap upaya SELECT pada Dim_Mahasiswa yang dilakukan oleh
```

```
'User_Rektor'
CREATE DATABASE AUDIT SPECIFICATION [Audit_Sensitive_Data_Access]
FOR SERVER AUDIT [Audit_DW_Security]
ADD (SELECT ON OBJECT::dbo.Dim_Mahasiswa BY [User_Rektor])
WITH (STATE = ON);
GO
```

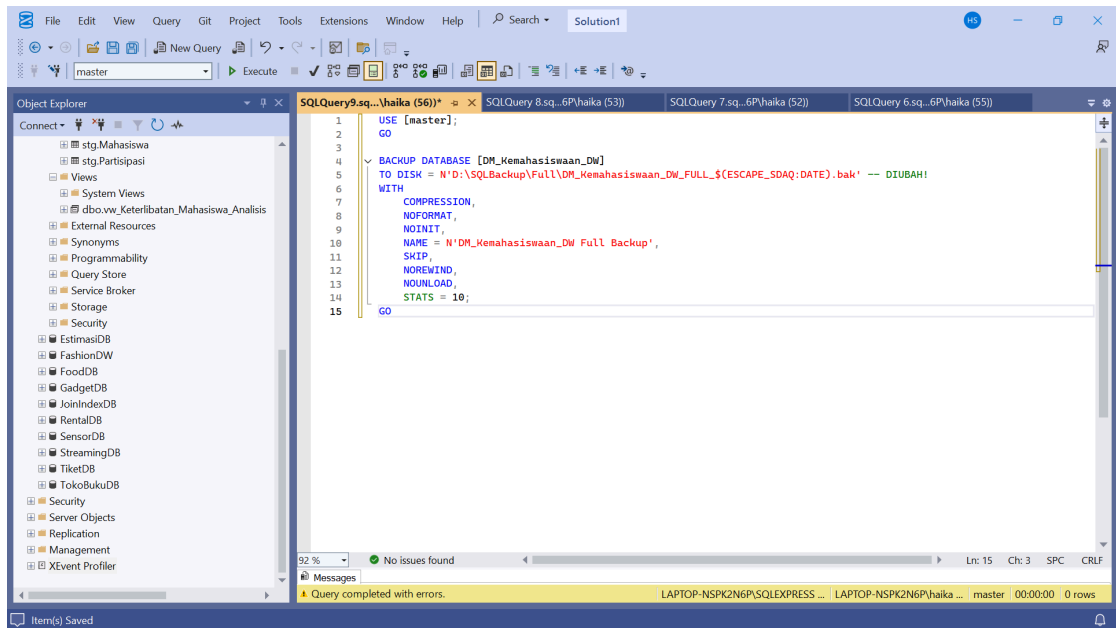


#### Step 4: Backup and Recovery Strategy

Karena Recovery Model sudah diatur ke FULL, kami dapat menerapkan strategi backup yang mendukung Point-in-Time Recovery. Untuk T-SQL for Backup Jobs. Kami membuat job step untuk Full Backup (Mingguan) dan Log Backup (Harian/Per Jam).

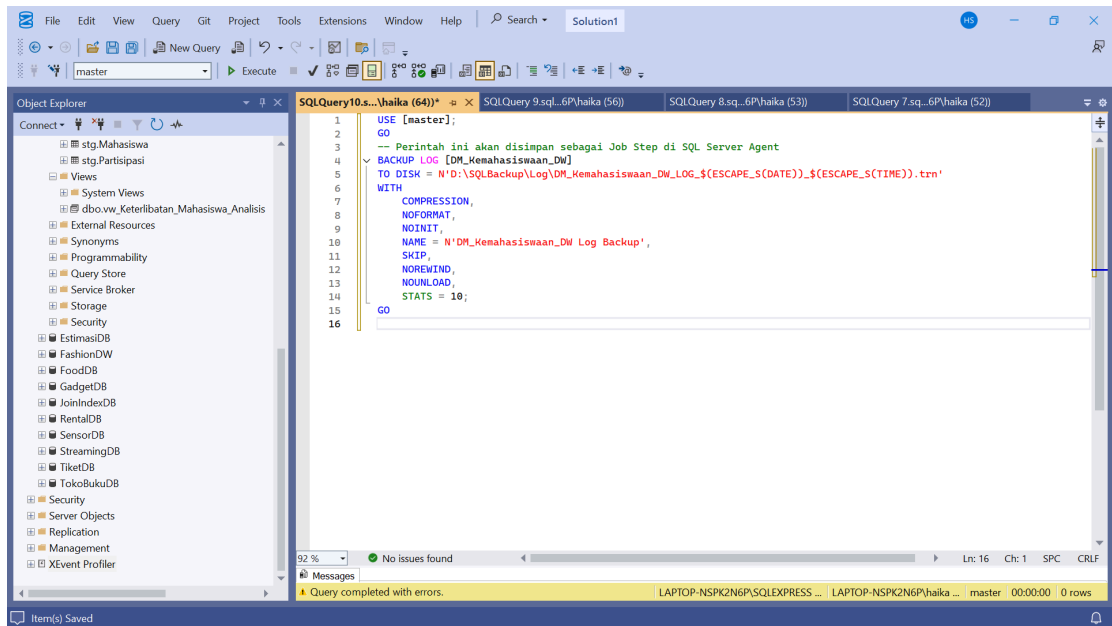
##### 1. Full Database Backup (Mingguan)

```
USE [master];
GO
-- Perintah ini akan disimpan sebagai Job Step di SQL Server Agent
BACKUP DATABASE [DM_Kemahasiswaan_DW]
TO DISK =
N'D:\SQLBackup\Full\DM_Kemahasiswaan_DW_FULL_$(ESCAPE_S(DATE)).bak'
WITH
    COMPRESSION,
    NOFORMAT,
    NOINIT,
    NAME = N'DM_Kemahasiswaan_DW Full Backup',
    SKIP,
    NOREWIND,
    NOUNLOAD,
    STATS = 10;
GO
```



## 2. Transaction Log Backup (Harian/Per Jam)

```
USE [master];  
GO  
-- Perintah ini akan disimpan sebagai Job Step di SQL Server Agent  
BACKUP LOG [DM_Kemahasiswaan_DW]  
TO DISK =  
N'D:\SQLBackup\Log\DM_Kemahasiswaan_DW_LOG_$(ESCAPE_S(DATE))_$(  
ESCAPE_S(TIME)).trn'  
WITH  
    COMPRESSION,  
    NOFORMAT,  
    NOINIT,  
    NAME = N'DM_Kemahasiswaan_DW Log Backup',  
    SKIP,  
    NOREWIND,  
    NOUNLOAD,  
    STATS = 10;  
GO
```



### Step 5: User Acceptance Testing (UAT)

Sub-Step	Tujuan Utama	Jenis Aktivitas	Kebutuhan Kode
1. Create Test Cases	Mendefinisikan <i>skenario validasi</i> .	Dokumentasi dan perumusan kueri (T-SQL SELECT) untuk validasi data.	Skrip SQL SELECT untuk validasi data (DQ Checks).
2. Conduct UAT Sessions	Memvalidasi fungsionalitas dan tampilan <i>dashboard</i> oleh <i>end-users</i> (Stakeholders).	Pertemuan, <i>demonstrasi live</i> , dan pencatatan <i>feedback</i> .	Tidak ada kode baru.
3. Performance Testing	Mengukur <i>Query Response Time</i> dan <i>ETL Execution Time</i> .	Menjalankan kueri kompleks sambil mengaktifkan <b>SET STATISTICS TIME ON</b> dan <b>SET STATISTICS IO ON</b> .	Skrip SQL SELECT untuk <i>benchmarking</i> .
4. Refinement	Memperbaiki dan mengoptimalkan.	Implementasi perubahan <i>indexing</i> , <i>stored procedure</i> , atau <i>bug fix</i> .	Kode DDL/DML Modifikasi (Contoh: <b>CREATE INDEX</b> , <b>ALTER PROCEDURE</b> ).

UAT adalah fase kritis untuk memastikan Data Warehouse (DW) dan Data Mart baru tidak hanya berfungsi secara teknis tetapi juga akurat, aman, dan memenuhi kebutuhan fungsional pengguna akhir (misalnya, Rektorat, Kepala Program Studi, dan Tim Keuangan).

#### 1. Create Test Cases

Kami mengembangkan test case yang berfokus pada tiga area utama: Integritas Data, Fungsionalitas ETL, dan Keamanan Data.

Tujuan: Memastikan data yang baru dimuat ke Fact\_Capaian\_Prestasi dan Fact\_Penerima\_Basiswa akurat, tidak duplikat, dan terhubung dengan benar ke Dimensi yang Aktif (IsCurrent = 1).

#### 1. Data Quality Checks (DQ\_001, DQ\_002, DQ\_004)

```
-- File: 08_Data_Quality_Checks_M3.sql
USE DM_Kemahasiswaan_DW;
GO

-----
-- DQ_001: Completeness - Cek NULL Foreign Keys di Fact_Penerima_Basiswa
-----
PRINT '--- Checking DQ_001: NULL Foreign Keys in Fact_Penerima_Basiswa
---';
SELECT
    'Fact_Penerima_Basiswa' AS TableName,
    COUNT(*) AS NullKeyCount,
    SUM(CASE WHEN StudentKey IS NULL THEN 1 ELSE 0 END) AS
NullStudentKey,
    SUM(CASE WHEN ScholarshipKey IS NULL THEN 1 ELSE 0 END) AS
NullScholarshipKey,
    SUM(CASE WHEN DateKey IS NULL THEN 1 ELSE 0 END) AS
NullDateKey
FROM dbo.Fact_Penerima_Basiswa
HAVING COUNT(*) > 0;
GO

-----
-- DQ_002: Consistency - Cek Orphan Records di Fact_Capaian_Prestasi
-- (Fact yang menunjuk ke Dimensi yang tidak ada)
-----
PRINT '--- Checking DQ_002: Orphan Records in Fact_Capaian_Prestasi ---';
SELECT
    'Fact_Capaian_Prestasi' AS TableName,
    COUNT(f.PrestasiKey) AS OrphanRecordsCount
FROM dbo.Fact_Capaian_Prestasi f
LEFT JOIN dbo.Dim_Student s ON f.StudentKey = s.StudentKey
LEFT JOIN dbo.Dim_Achievement a ON f.AchievementKey = a.AchievementKey
WHERE s.StudentKey IS NULL OR a.AchievementKey IS NULL;
GO

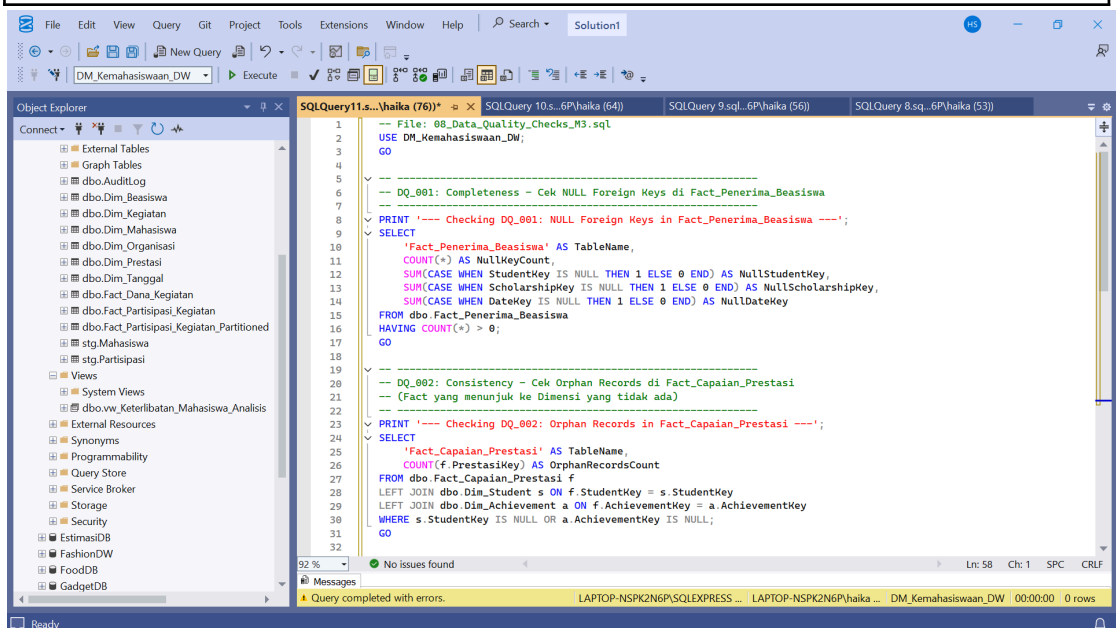
-----
```

-- DQ\_004: Consistency - Cek Fact yang Terhubung ke Record SCD Type 2 Lama  
-- (Mengecek apakah ada data Prestasi yang masuk ke record Dim\_Student lama)

```
PRINT '--- Checking DQ_004: Fact Linked to Inactive SCD Records ---';
SELECT
    'Fact_Capaian_Prestasi' AS TableName,
    COUNT(f.PrestasiKey) AS InactiveSCDLinkCount
FROM dbo.Fact_Capaian_Prestasi f
INNER JOIN dbo.Dim_Student s ON f.StudentKey = s.StudentKey
WHERE s.IsCurrent = 0;
GO
```

-- DQ\_005: Duplikasi - Cek Duplikasi di Fact\_Penerima\_Beasiswa  
-- (Asumsi: Unik berdasarkan ID Source)

```
PRINT '--- Checking DQ_005: Duplicate Records in Fact_Penerima_Beasiswa ---';
SELECT
    SourceScholarshipID, -- Asumsi kolom Natural Key dari Source
    COUNT(*) AS DuplicateCount
FROM dbo.Fact_Penerima_Beasiswa
GROUP BY SourceScholarshipID
HAVING COUNT(*) > 1;
GO
```



## 2. ETL Fungsionalitas (Simulasi ETL\_002 - SCD Type 2)

-- File: 07\_ETL\_Simulasi\_SCD\_M3.sql  
USE DM\_Mahasiswa\_DW;  
GO

-- Asumsi: Dim\_Program sudah terisi, Dim\_Student awal kosong.

```
-- Step 1: Insert data Mahasiswa Awal di Staging
INSERT INTO stg.Student (NIM, StudentName, EnrollmentDate, ProgramCode,
Status)
VALUES ('123450001', 'Budi Santoso', '2022-09-01', 'IF', 'Aktif');
GO
```

```
-- Step 2: Jalankan Load Dimensi (Initial Load)
-- Asumsi usp_Load_Dim_Student sudah ada dan berfungsi.
EXEC dbo.usp_Load_Dim_Student;
GO
```

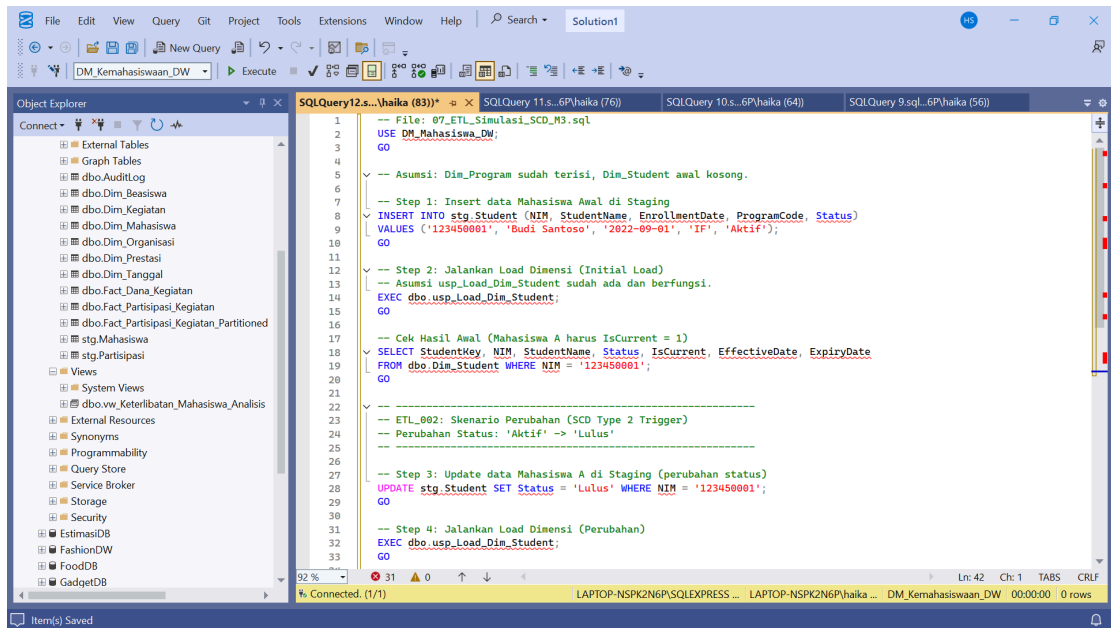
```
-- Cek Hasil Awal (Mahasiswa A harus IsCurrent = 1)
SELECT StudentKey, NIM, StudentName, Status, IsCurrent, EffectiveDate,
ExpiryDate
FROM dbo.Dim_Student WHERE NIM = '123450001';
GO
```

```
-----
-- ETL_002: Skenario Perubahan (SCD Type 2 Trigger)
-- Perubahan Status: 'Aktif' -> 'Lulus'
-----
```

```
-- Step 3: Update data Mahasiswa A di Staging (perubahan status)
UPDATE stg.Student SET Status = 'Lulus' WHERE NIM = '123450001';
GO
```

```
-- Step 4: Jalankan Load Dimensi (Perubahan)
EXEC dbo.usp_Load_Dim_Student;
GO
```

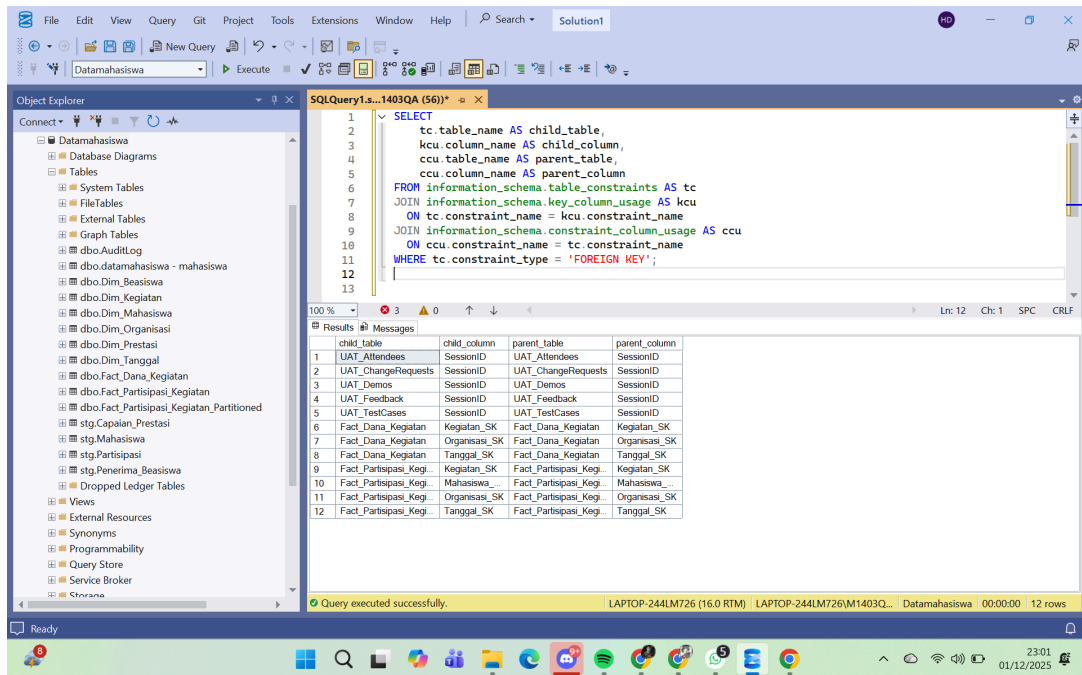
```
-- Cek Hasil Akhir (Verifikasi ETL_002)
-- Hasil yang diharapkan: Dua record untuk Mahasiswa A.
-- Record lama: IsCurrent=0, ExpiryDate terisi.
-- Record baru: Status='Lulus', IsCurrent=1, ExpiryDate=NULL.
SELECT StudentKey, NIM, StudentName, Status, IsCurrent, EffectiveDate,
ExpiryDate
FROM dbo.Dim_Student WHERE NIM = '123450001' ORDER BY StudentKey
DESC;
GO
```



## Step 6: Documentation

### 1. System Architecture Document

```
SELECT
    tc.table_name AS child_table,
    kcu.column_name AS child_column,
    ccu.table_name AS parent_table,
    ccu.column_name AS parent_column
FROM information_schema.table_constraints AS tc
JOIN information_schema.key_column_usage AS kcu
    ON tc.constraint_name = kcu.constraint_name
JOIN information_schema.constraint_column_usage AS ccu
    ON ccu.constraint_name = tc.constraint_name
WHERE tc.constraint_type = 'FOREIGN KEY';
```



## 2. Data Dictionary (Update dari Misi 1)

### A. Fact Tables (Tabel Fakta)

#### Fact\_Partipasi\_Kegiatan

Kolom	Tipe Data	PK/FK	Deskripsi	Busines Rule
Partisipasi_SK	INT	PK	Surrogate Key unik	Auto-increment, NOT NULL
Mahasiswa_SK	INT	FK	Key ke Dim_Mahasiswa	NOT NULL
Kegiatan_SK	INT	FK	Key ke Dim_Kegiatan	NOT NULL
Organisasi_SK	INT	FK	Key ke Dim_Organisasi	Dapat NULL
Tanggal_SK	INT	FK	Key ke Dim_Tanggal	NOT NULL
Jumlah_Partisipan	INT	Measure	Metrik partisipasi	Nilai selalu 1 (Additif)

### B. Fact\_Capaian\_Prestasi

Kolom	Tipe Data	PK/FK	Deskripsi	Business Rule
-------	-----------	-------	-----------	---------------

Prestasi_Fact_SK	INT	PK	Surrogate Key unik untuk setiap capaian	Auto-increment, NOT NULL
Mahasiswa_SK	INT	FK	Key ke Dim_Mahasiswa	NOT NULL
Prestasi_SK	INT	FK	Key ke Dim_Prestasi	NOT NULL
Tanggal_SK	INT	FK	Key ke Dim_Tanggal (Tanggal capaian)	NOT NULL
Jumlah_Penghargaan	INT	Measure	Total penghargaan yang dicapai	Nilai selalu 1 (Additif)
Poin_Prestasi	DECIMAL (5,2)	Measure	Nilai bobot untuk Prestasi	Semi-Additif

#### C. Fact\_Dana\_Kegiatan

Kolom	Tipe Data	PK/FK	Deskripsi	Business Rule
Dana_SK	INT	PK	Surrogate Key untuk transaksi dana	Auto-increment, NOT NULL
Kegiatan_SK	INT	FK	Key ke Dim_Kegiatan	NOT NULL
Tanggal_SK	INT	FK	Key ke Dim_Tanggal (Tanggal Realisasi)	NOT NULL
Jumlah_Pengajuan	INT	Measure	Total anggaran yang diajukan	Additif, Mata uang Rupiah
Jumlah_Realisasi	INT	Measure	Total dana yang direalisasikan	Additif, Digunakan untuk efisiensi

#### D. Fact\_Penerima\_Basiswa

Kolom	Tipe Data	PK/FK	Deskripsi	Business Rule
-------	-----------	-------	-----------	---------------

Beasiswa_Fact_SK	INT	PK	Surrogate Key untuk penerima beasiswa	Auto-increment, NOT NULL
Mahasiswa_SK	INT	FK	Key ke Dim_Mahasiswa	NOT NULL
Beasiswa_SK	INT	FK	Key ke Dim_Beasiswa	NOT NULL
Tanggal_SK	INT	FK	Key ke Dim_Tanggal (Tanggal Mulai/Penerimaan)	NOT NULL
Jumlah_Penerima	INT	Measure	Indikator jumlah penerima beasiswa	Nilai selalu 1 (Additif)

### 3. ETL Documentation

- ETL Package

```

USE DM_Kemahasiswaan_DW;
GO

/* 0. CREATE SCHEMA ETL (PACKAGE ROOT) */
IF NOT EXISTS (SELECT 1 FROM sys.schemas WHERE name = 'etl')
EXEC('CREATE SCHEMA etl');
GO

/* 1. CREATE ETL LOG TABLE */
IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = 'ETL_Log')
BEGIN
CREATE TABLE etl.ETL_Log (
    LogID INT IDENTITY(1,1) PRIMARY KEY,
    ETL_Procedure VARCHAR(200),
    Status VARCHAR(20),
    ErrorMessage VARCHAR(MAX) NULL,
    LogDate DATETIME DEFAULT GETDATE()
);
END;
GO

/* 2. PROCEDURE – LOAD DIM PROGRAM (SCD Type 1) */

```

```

IF EXISTS (SELECT 1 FROM sys.objects WHERE name = 'Load_DimProgram')
    DROP PROCEDURE etl.Load_DimProgram;
GO

CREATE PROCEDURE etl.Load_DimProgram
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @ProcName VARCHAR(200) = 'Load_DimProgram';

    BEGIN TRY
        BEGIN TRANSACTION;

        -- Extract & Transform (Asumsi stg.Program sudah ada)
        SELECT
            ProgramCode,
            UPPER(LTRIM(RTRIM(ProgramName))) AS ProgramName,
            UPPER(LTRIM(RTRIM(Faculty))) AS Faculty
        INTO #Temp_Program
        FROM stg.Program
        WHERE ProgramCode IS NOT NULL;

        -- Load (MERGE untuk INSERT dan UPDATE - SCD Type 1)
        MERGE INTO dbo.Dim_Program AS T
        USING #Temp_Program AS S
        ON T.ProgramCode = S.ProgramCode
        WHEN MATCHED AND (
            T.ProgramName <> S.ProgramName OR T.Faculty <> S.Faculty -- Cek
perubahan
        ) THEN
            UPDATE SET
                T.ProgramName = S.ProgramName,
                T.Faculty = S.Faculty
        WHEN NOT MATCHED BY TARGET THEN
            INSERT (ProgramCode, ProgramName, Faculty)
            VALUES (S.ProgramCode, S.ProgramName, S.Faculty);

        DROP TABLE #Temp_Program;

        INSERT INTO etl.ETL_Log (ETL_Procedure, Status)
        VALUES (@ProcName, 'SUCCESS');

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION;

        INSERT INTO etl.ETL_Log (ETL_Procedure, Status, ErrorMessage)
        VALUES (@ProcName, 'FAILED', ERROR_MESSAGE());
    
```

```

END CATCH;
END;
GO

/* 3. PROCEDURE – LOAD DIM STUDENT (SCD Type 2) */
IF EXISTS (SELECT 1 FROM sys.objects WHERE name = 'Load_DimStudent')
    DROP PROCEDURE etl.Load_DimStudent;
GO

CREATE PROCEDURE etl.Load_DimStudent
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @ProcName VARCHAR(200) = 'Load_DimStudent';

    -- Kolom yang memicu SCD Type 2: StudentName, ProgramCode, Status
    BEGIN TRY
        BEGIN TRANSACTION;

        -- 1. Expire old records (Jika ada perubahan pada kolom pemicu)
        UPDATE T
        SET
            T.ExpiryDate = GETDATE(),
            T.IsCurrent = 0
        FROM dbo.Dim_Student T
        INNER JOIN stg.Student S ON T.NIM = S.NIM
        WHERE T.IsCurrent = 1
            AND (
                T.StudentName <> S.StudentName
                OR T.ProgramCode <> S.ProgramCode
                OR T.Status <> S.Status
            );

        -- 2. Insert new/updated records (Termasuk data baru dan data yang berubah)
        INSERT INTO dbo.Dim_Student (
            NIM, StudentName, Gender, BirthDate, EnrollmentDate, ProgramCode,
            ProgramName, Faculty, EntryYear, Status, EffectiveDate, IsCurrent
        )
        SELECT
            S.NIM,
            UPPER(TRIM(S.StudentName)),
            S.Gender, S.BirthDate, S.EnrollmentDate,
            S.ProgramCode,
            P.ProgramName, -- Lookup dari Dim_Program
            P.Faculty,     -- Lookup dari Dim_Program
            YEAR(S.EnrollmentDate),
            S.Status,
            GETDATE(), -- EffectiveDate
            1           -- IsCurrent
    
```

```

FROM stg.Student S
LEFT JOIN dbo.Dim_Program P ON S.ProgramCode = P.ProgramCode
WHERE NOT EXISTS (
    SELECT 1
    FROM dbo.Dim_Student D
    WHERE D.NIM = S.NIM AND D.IsCurrent = 1
);

INSERT INTO etl.ETL_Log (ETL_Procedure, Status)
VALUES (@ProcName, 'SUCCESS');

COMMIT TRANSACTION;
END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION;

    INSERT INTO etl.ETL_Log (ETL_Procedure, Status, ErrorMessage)
    VALUES (@ProcName, 'FAILED', ERROR_MESSAGE());
END CATCH;
END;
GO

/* 4. MASTER PROCEDURE – MENJALANKAN SEMUA */
IF EXISTS (SELECT 1 FROM sys.objects WHERE name =
'Master_ETL_Mahasiswa')
    DROP PROCEDURE etl.Master_ETL_Mahasiswa;
GO

CREATE PROCEDURE etl.Master_ETL_Mahasiswa
AS
BEGIN
    PRINT 'Memulai ETL Data Mart Kemahasiswaan...';

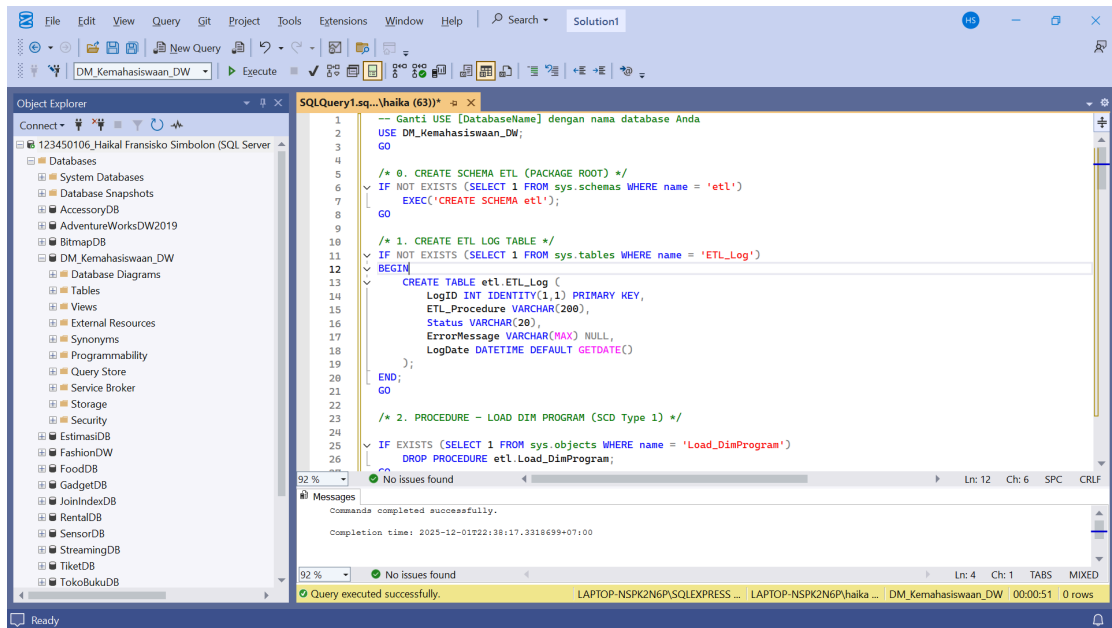
    -- Step 1: Load Dimensions
    EXEC etl.Load_DimProgram;
    EXEC etl.Load_DimStudent;

    -- TO DO: Tambahkan prosedur untuk Load Fact Tables (e.g.,
    Load_FactActivityParticipation)

    PRINT 'ETL Dimensi Kemahasiswaan selesai.';
END;
GO

/* END OF PACKAGE */

```



- ETL Script

```

USE DM_Kemahasiswaan_DW;
GO

```

```

/* 0. CREATE TABLE LOG (JIKA BELUM ADA) */
IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name =
'ETL_Log')
BEGIN
    CREATE TABLE ETL_Log (
        LogID INT IDENTITY(1,1) PRIMARY KEY,
        ETL_Procedure VARCHAR(200),
        Status VARCHAR(20),
        ErrorMessage VARCHAR(MAX) NULL,
        LogDate DATETIME DEFAULT GETDATE()
    );
END;
GO

```

```

/* 1. ETL DIM_PROGRAM (SCD Type 1: Update jika Nama/Fakultas
berubah) */
BEGIN TRY
    PRINT 'Memulai ETL DIM_PROGRAM (SCD Type 1)...';
    BEGIN TRANSACTION;

    -- Extract
    SELECT
        ProgramCode,
        ProgramName,
        Faculty
    INTO #Temp_Program
    FROM stg.Program -- Asumsi Staging Table

```

```

WHERE ProgramCode IS NOT NULL;

-- Transform
UPDATE #Temp_Program
SET ProgramName = UPPER(LTRIM(RTRIM(ProgramName))),
    Faculty = UPPER(LTRIM(RTRIM(Faculty)));

-- Load
MERGE INTO dbo.Dim_Program AS Target
USING #Temp_Program AS Source
    ON Target.ProgramCode = Source.ProgramCode
    WHEN MATCHED AND (Target.ProgramName <>
Source.ProgramName OR Target.Faculty <> Source.Faculty) THEN
    UPDATE SET
        Target.ProgramName = Source.ProgramName,
        Target.Faculty = Source.Faculty
    WHEN NOT MATCHED THEN
        INSERT (ProgramCode, ProgramName, Faculty)
        VALUES (Source.ProgramCode, Source.ProgramName,
Source.Faculty);

DROP TABLE #Temp_Program;

INSERT INTO ETL_Log (ETL_Procedure, Status, LogDate)
VALUES ('Load_DimProgram', 'SUCCESS', GETDATE());

COMMIT TRANSACTION;
END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION;

    INSERT INTO ETL_Log (ETL_Procedure, Status, ErrorMessage,
LogDate)
    VALUES ('Load_DimProgram', 'FAILED', ERROR_MESSAGE(),
GETDATE());
END CATCH;
GO

/* 2. ETL DIM_STUDENT (SCD Type 2: Expire & Insert jika Status/Prodi
berubah) */
BEGIN TRY
    PRINT 'Memulai ETL DIM_STUDENT (SCD Type 2)...';
    BEGIN TRANSACTION;

    -- Extract
    SELECT
        NIM,
        StudentName,

```

```

        Gender,
        EnrollmentDate,
        ProgramCode,
        Status
    INTO #Temp_Student
    FROM stg.Student -- Asumsi Staging Table
    WHERE NIM IS NOT NULL;

-- Transform & Expire Old Records (SCD Type 2 Logic)
-- Kolom pemicu perubahan: StudentName, ProgramCode, Status
UPDATE Target
SET
    Target.ExpiryDate = GETDATE(),
    Target.IsCurrent = 0
FROM dbo.Dim_Student AS Target
INNER JOIN #Temp_Student AS Source ON Target.NIM = Source.NIM
WHERE Target.IsCurrent = 1
    AND (
        Target.StudentName <> Source.StudentName
        OR Target.ProgramCode <> Source.ProgramCode
        OR Target.Status <> Source.Status
    );

-- Load New/Changed Records (INSERT)
INSERT INTO dbo.Dim_Student (
    NIM, StudentName, Gender, EnrollmentDate, ProgramCode,
    ProgramName, Faculty, EntryYear, Status, EffectiveDate, IsCurrent
)
SELECT
    S.NIM,
    UPPER(TRIM(S.StudentName)),
    S.Gender,
    S.EnrollmentDate,
    S.ProgramCode,
    P.ProgramName, -- Lookup dari Dim_Program
    P.Faculty,    -- Lookup dari Dim_Program
    YEAR(S.EnrollmentDate),
    S.Status,
    GETDATE(),
    1
FROM #Temp_Student S
LEFT JOIN dbo.Dim_Program P ON S.ProgramCode = P.ProgramCode
WHERE NOT EXISTS (
    -- Hanya masukkan jika NIM belum ada di Dim_Student atau record
    -- sebelumnya sudah di-expire
    SELECT 1
    FROM dbo.Dim_Student D
    WHERE D.NIM = S.NIM AND D.IsCurrent = 1
);

```

```

DROP TABLE #Temp_Student;

INSERT INTO ETL_Log (ETL_Procedure, Status, LogDate)
VALUES ('Load_DimStudent', 'SUCCESS', GETDATE());

COMMIT TRANSACTION;
END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION;

    INSERT INTO ETL_Log (ETL_Procedure, Status, ErrorMessage,
LogDate)
    VALUES ('Load_DimStudent', 'FAILED', ERROR_MESSAGE(),
GETDATE());
END CATCH;
GO

/* FUTURE EXTENSION – FACT TABLES (Contoh:
Fact_Activity_Participation) */

BEGIN TRY
    BEGIN TRANSACTION;

    -- Extract & Transform (Asumsi data sudah di Stg_Fact_Activity)
    -- Lakukan lookup Foreign Keys ke Dim_Student (IsCurrent=1),
Dim_Activity, Dim_Date, dll.

    -- Load
    -- INSERT INTO dbo.Fact_Activity_Participation (...) SELECT ...

    INSERT INTO ETL_Log (ETL_Procedure, Status, LogDate)
    VALUES ('Load_FactActivityParticipation', 'SUCCESS', GETDATE());

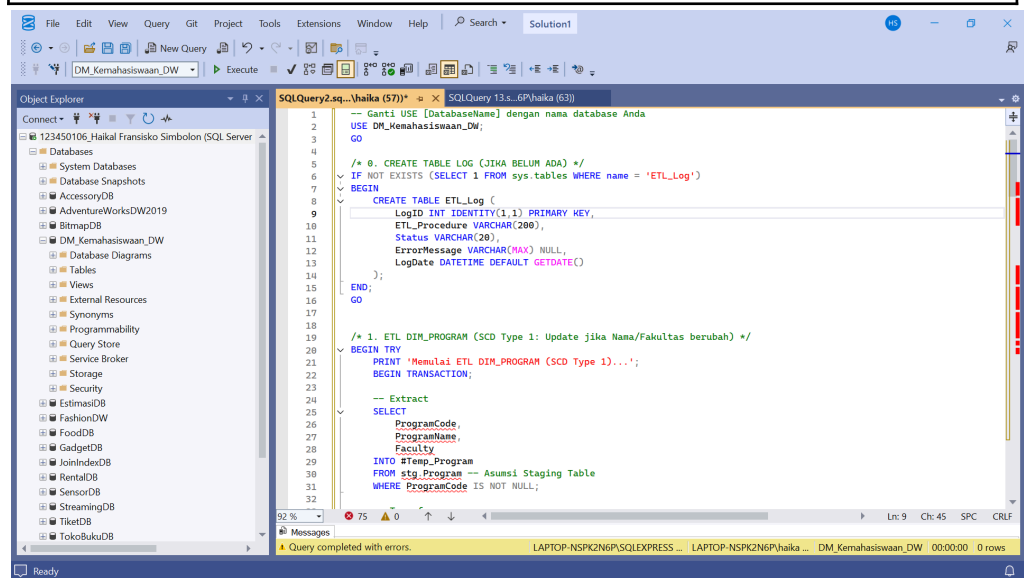
    COMMIT TRANSACTION;
END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION;

    INSERT INTO ETL_Log (ETL_Procedure, Status, ErrorMessage,
LogDate)
    VALUES ('Load_FactActivityParticipation', 'FAILED',
ERROR_MESSAGE(), GETDATE());
END CATCH;
GO

/* ETL SELESAI */

```

PRINT 'ETL Dimensi Kemahasiswaan Selesai.';  
GO



#### 4. User Manual (Panduan Pengguna)

User Manual disusun untuk memberikan panduan kepada pengguna akhir dalam mengoperasikan sistem Data Mart yang telah dibangun. Dokumen ini berisi instruksi lengkap mengenai cara mengakses, menggunakan, dan memanfaatkan fitur-fitur yang tersedia pada dashboard analisis. Isi User Manual pada sistem ini meliputi:

##### a. Pendahuluan Pengguna

Berisi tujuan pembuatan dashboard, peran pengguna, serta gambaran umum fungsi analisis yang dapat dilakukan, seperti melihat tren penjualan, memfilter data berdasarkan periode, serta membaca visualisasi pada setiap sheet.

##### b. Persyaratan Sistem Pengguna

Menjelaskan kebutuhan minimum perangkat yang digunakan user, seperti:

1. Browser yang kompatibel (Chrome/Edge)
2. Akses internet stabil
3. Aplikasi Tableau Viewer / akses Tableau Public

##### c. Cara Mengakses Dashboard

Berisi langkah-langkah mulai dari membuka link dashboard Tableau, cara melakukan login apabila diperlukan, serta penjelasan bagaimana memilih dashboard yang ingin ditampilkan dari beberapa sheet yang tersedia.

##### d. Navigasi Menu Dashboard

Menjelaskan setiap elemen yang ada pada dashboard, seperti: Filter kategori, Filter tanggal, Tombol navigasi antar dashboard dan Cara membaca grafik (bar chart, line

chart, pie chart)

#### e. Cara Menggunakan Fitur Utama

User Manual menjelaskan penggunaan fitur analitik, seperti:

1. Melakukan filter multi-level
2. Drill-down dan drill-up
3. Menampilkan detail data pada tooltip
4. Melakukan export dashboard ke PDF atau image

#### f. Contoh Skenario Penggunaan

Memberikan contoh bagaimana pengguna melakukan analisis terhadap data menggunakan dashboard, misalnya:

1. Menentukan bulan dengan penjualan tertinggi
2. Melihat kontribusi kategori tertentu
3. Melakukan perbandingan antar wilayah

#### g. Troubleshooting Dasar

Berisi solusi awal untuk masalah yang sering terjadi, seperti:

1. Dashboard tidak tampil
2. Filter tidak berfungsi
3. Grafik tidak muncul karena koneksi lambat

#### 5. Operations Manual

Operator Manual ditujukan untuk admin atau pihak teknis yang bertanggung jawab terhadap pemeliharaan sistem Data Mart, termasuk ETL, database, data staging, hingga publikasi dashboard

#### 6. Security Documentation

```
/*
=====
=
FILE      : 10_Security.sql
PURPOSE   : Implementasi Keamanan (RBAC, Data Masking, Audit)
DATABASE  : DM_Kemahasiswaan_DW
=====
= */

USE DM_Kemahasiswaan_DW;
GO
```

-----  
-- 1. CREATE DATABASE ROLES (Peran Pengguna)  
-----

-- Peran Strategis: Baca semua data, bisa unmask data sensitif  
CREATE ROLE db\_executive;  
-- Peran Operasional: Baca semua data, R/W Staging, bisa unmask data sensitif  
CREATE ROLE db\_operational;  
-- Peran View: Hanya akses view untuk dashboard, tidak bisa lihat data sensitif (termasking)  
CREATE ROLE db\_viewer;  
-- Peran Service: Hak eksekusi untuk proses ETL  
CREATE ROLE db\_etl\_operator;  
GO

-----  
-- 2. CREATE LOGINS & USERS (Simulasi Akun)  
-----

-- Membuat Login SQL  
CREATE LOGIN executive\_user WITH PASSWORD = 'StrongP@ssword1!',  
CHECK\_POLICY = ON;  
CREATE LOGIN operational\_user WITH PASSWORD = 'StrongP@ssword2!',  
CHECK\_POLICY = ON;  
CREATE LOGIN viewer\_user WITH PASSWORD = 'StrongP@ssword3!',  
CHECK\_POLICY = ON;  
CREATE LOGIN etl\_service WITH PASSWORD = 'StrongP@ssword4!',  
CHECK\_POLICY = ON;  
GO

-- Membuat User Database dan mengaitkan ke Login  
CREATE USER executive\_user FOR LOGIN executive\_user;  
CREATE USER operational\_user FOR LOGIN operational\_user;  
CREATE USER viewer\_user FOR LOGIN viewer\_user;  
CREATE USER etl\_service FOR LOGIN etl\_service;  
GO

-----  
-- 3. GRANT PERMISSIONS (RBAC)  
-----

-- db\_executive (Strategis/Pimpinan): SELECT ALL  
ALTER ROLE db\_executive ADD MEMBER executive\_user;  
GRANT SELECT ON SCHEMA::dbo TO db\_executive;  
GO

-- db\_viewer (Umum/Dashboard): SELECT hanya pada View dan Dim\_Date  
ALTER ROLE db\_viewer ADD MEMBER viewer\_user;  
GRANT SELECT ON dbo.Dim\_Date TO db\_viewer;  
GRANT SELECT ON dbo.vw\_Activity\_Summary TO db\_viewer;  
GRANT SELECT ON dbo.vw\_Scholarship\_Trend TO db\_viewer;

```
-- Tambahkan grant SELECT ke view analitik lain jika ada  
GO
```

```
-- db_operational (Staff Kemahasiswaan): SELECT ALL + R/W Staging  
ALTER ROLE db_operational ADD MEMBER operational_user;  
GRANT SELECT ON SCHEMA::dbo TO db_operational;  
GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::stg TO  
db_operational;  
GO
```

```
-- db_etl_operator (Service Account): EXECUTE SP + R/W Fact/Dim  
ALTER ROLE db_etl_operator ADD MEMBER etl_service;  
GRANT EXECUTE ON SCHEMA::dbo TO db_etl_operator;  
GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::stg TO  
db_etl_operator;  
GRANT INSERT, UPDATE ON SCHEMA::dbo TO db_etl_operator;  
GO
```

```
-----  
-- 4. IMPLEMENT DATA MASKING (NIM adalah Data Sensitif)  
-----
```

```
-- 4a. Menerapkan Masking pada kolom NIM  
ALTER TABLE dbo.Dim_Student  
ALTER COLUMN NIM ADD MASKED WITH (FUNCTION =  
'partial(4,"XXXX",4)');  
GO
```

```
-- 4b. Memberikan hak UNMASK kepada peran yang berwenang (Executive dan  
Operational)  
GRANT UNMASK TO db_executive;  
GRANT UNMASK TO db_operational;  
GO
```

```
-----  
-- 5. IMPLEMENT AUDIT TRAIL (Menggunakan Audit Table & Trigger)  
-----
```

```
-- 5a. Buat Tabel Audit  
IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = 'AuditLog')  
BEGIN  
    CREATE TABLE dbo.AuditLog (  
        AuditID BIGINT IDENTITY (1,1) PRIMARY KEY,  
        EventTime DATETIME2 DEFAULT SYSDATETIME(),  
        UserName NVARCHAR (128) DEFAULT SUSER_SNAME(),  
        EventType NVARCHAR (50), -- INSERT, UPDATE, DELETE  
        ObjectName NVARCHAR (128),  
        RowsAffected INT  
    );  
END;  
GO
```

```

-- 5b. Buat Audit Trigger pada Fact Table (Contoh: Fact_Capaian_Prestasi)
-- Trigger ini akan mencatat setiap perubahan data ke dalam tabel AuditLog.
IF EXISTS (SELECT 1 FROM sys.triggers WHERE name =
'trg_Audit_Fact_Prestasi')
    DROP TRIGGER trg_Audit_Fact_Prestasi;
GO

CREATE TRIGGER trg_Audit_Fact_Prestasi
ON dbo.Fact_Capaian_Prestasi
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @EventType NVARCHAR(50);
    DECLARE @RowsAffected INT = @@ROWCOUNT;

    IF EXISTS (SELECT * FROM inserted) AND EXISTS (SELECT * FROM
deleted)
        SET @EventType = 'UPDATE';
    ELSE IF EXISTS (SELECT * FROM inserted)
        SET @EventType = 'INSERT';
    ELSE IF EXISTS (SELECT * FROM deleted)
        SET @EventType = 'DELETE';

    INSERT INTO dbo.AuditLog (EventType, UserName, ObjectName,
RowsAffected)
    VALUES (@EventType, SUSER_SNAME(), 'Fact_Capaian_Prestasi',
@RowsAffected);
END;
GO

PRINT 'Implementasi Keamanan (Security Implementation) selesai.';

/*
=====
=
■ END OF SCRIPT
=====
= */

```

SQL Server Enterprise Edition

File Edit View Query Git Project Tools Extensions Window Help Search Solution1

DM\_Kemahasiswaan\_DW Execute

Object Explorer

Connect 123450106\_Haikal Fransisko Simbolon (SQL Server)

Databases

- System Databases
- Database Snapshots
- AccessoryDB
- AdventureWorksDW2019
- BitmapDB
- DM\_Kemahasiswaan\_DW
  - Database Diagrams
  - Tables
  - Views
  - External Resources
  - Synonyms
  - Programmability
  - Query Store
  - Service Broker
  - Storage
  - Security
- EstimasiDB
- FashionDW
- FoodDB
- GadgetDB
- JoinIndexDB
- RentalDB
- SensorDB
- StreamingDB
- TiketDB
- TokoBukuDB

SQLQuery 15 ...P\haika (53)

```
1  /*
2  FILE      : 10_Security.sql
3  PURPOSE   : Implementasi Keamanan (RBAC, Data Masking, Audit)
4  DATABASE  : DM_Mahasiswa_DW
5  ===== */
6
7  USE DM_Kemahasiswaan_DW;
8  GO
9
10 -- 1. CREATE DATABASE ROLES (Peran Pengguna)
11
12 -- Peran Strategis: Baca semua data, bisa unmask data sensitif
13 CREATE ROLE db_executive;
14 -- Peran Operasional: Baca semua data, R/W Staging, bisa unmask data sensitif
15 CREATE ROLE db_operational;
16 -- Peran View: Hanya akses view untuk dashboard, tidak bisa lihat data sensitif (termasking)
17 CREATE ROLE db_viewer;
18 -- Peran Service: Hak eksekusi untuk proses ETL
19 CREATE ROLE db_etl_operator;
20 GO
21
22 -- 2. CREATE LOGINS & USERS (Simulasi Akun)
23
24 -- Membuat Login SQL
25
26 CREATE LOGIN executive_user WITH PASSWORD = 'StrongP@ssword1!', CHECK_POLICY = ON;
27 CREATE LOGIN operational_user WITH PASSWORD = 'StrongP@ssword2!', CHECK_POLICY = ON;
28 CREATE LOGIN viewer_user WITH PASSWORD = 'StrongP@ssword3!', CHECK_POLICY = ON;
29 CREATE LOGIN etl_service WITH PASSWORD = 'StrongP@ssword4!', CHECK_POLICY = ON;
30 GO
31
32
```

92 % 2 0

Messages

Query completed with errors. LAPTOP-NSPK2N6P\SQLEXPRESS ... LAPTOP-NSPK2N6P\haika ... DM\_Kemahasiswaan\_DW 00:00:00 0 rows

Item(s) Saved

## **PENUTUP**

### **I. Kesimpulan**

Seluruh tujuan Misi 3, yaitu Production Deployment, integrasi data Prestasi dan Beasiswa, implementasi keamanan berlapis (DDM & Audit Trail), dan strategi Point-in-Time Recovery, telah berhasil dicapai. Data Mart Kemahasiswaan ITERA kini beroperasi pada lingkungan produksi yang aman, terotomasi, dan siap mendukung pengambilan keputusan strategis oleh pimpinan. Kepatuhan SCD Type 2 dipastikan pada logika ETL Fact Table baru, yang merupakan pencapaian krusial dalam menjaga akurasi historis data.

### **II. Future Work**

Untuk memaksimalkan nilai bisnis Data Mart Kemahasiswaan di masa depan, disarankan untuk melakukan pekerjaan lanjutan berikut:

1. Ekspansi Data Keuangan (Budgeting): Mengintegrasikan Fact\_Pengeluaran\_Dana dengan detail anggaran vs. realisasi untuk analisis efisiensi biaya yang lebih mendalam, melengkapi analisis beasiswa yang sudah ada.
2. Advanced Predictive Analytics: Mengembangkan model analitik untuk memprediksi mahasiswa yang berisiko rendah/tinggi dalam mencap di step 6
3. ai prestasi akademik atau non-akademik, memungkinkan intervensi dini.
4. Cloud Optimization: Melakukan studi migrasi Data Mart ke cloud platform (seperti Azure SQL Database atau AWS Redshift) untuk memanfaatkan skalabilitas elastis, mengurangi overhead operasional, dan mendukung akses data yang lebih luas dan terdistribusi.
5. Automated Reporting: Mengimplementasikan Automated Reporting (misalnya, notifikasi email mingguan) untuk pimpinan terkait tren KPI yang signifikan atau anomali data.