

**KELOMPOK 2:**

1. Abit Ahmad Oktarian (122450042)
2. Tesalonika Hutajulu (123450033)
3. Rahmah Gustriana Deka (123450102)
4. Sarah Wasti (123450057)

**UNIT: Akademik****MISI 2: DESAIN FISIKAL DAN DEVELOPMENT****Step 1: Physical Database Design**

## 1. Database Setup

Query:

```
CREATE DATABASE akademik
GO
use akademik
GO
```

Kode diatas digunakan untuk membuat database bernama Akademik, lalu menggunakannya untuk proses selanjutnya.

## 2. Create Dimension Tables

Query:

```
-- Buat Tabel Dimensi
-- Dim_Program
CREATE TABLE dbo.Dim_Program (
    ProgramKey     INT IDENTITY(1,1) PRIMARY KEY,
    ProgramCode    VARCHAR(20) NOT NULL,
    ProgramName    VARCHAR(100) NOT NULL,
    Faculty        VARCHAR(100) NOT NULL,
    CONSTRAINT UQ_Dim_Program_Code UNIQUE (ProgramCode)
);
GO

-- Dim_Date
CREATE TABLE dbo.Dim_Date (
    DateKey        INT PRIMARY KEY,      -- yyyyymmdd
    [Date]          DATE NOT NULL,
    [Day]           INT NOT NULL,
    [Month]         INT NOT NULL,
```

```

[Quarter] INT NOT NULL,
[Year] INT NOT NULL,
SemesterCode VARCHAR(20) NULL,
CONSTRAINT UQ_Dim_Date UNIQUE ([Date]),
CONSTRAINT CHK_Dim_Date_Month CHECK ([Month] BETWEEN 1 AND 12),
CONSTRAINT CHK_Dim_Date_Quarter CHECK ([Quarter] BETWEEN 1 AND 4)
);
GO

-- Dim_Semester
CREATE TABLE dbo.Dim_Semester (
    SemesterKey INT IDENTITY(1,1) PRIMARY KEY,
    SemesterCode VARCHAR(20) NOT NULL,
    StartDate DATE NOT NULL,
    EndDate DATE NOT NULL,
    CONSTRAINT UQ_Dim_Semester_Code UNIQUE (SemesterCode),
    CONSTRAINT CHK_Dim_Semester_Dates CHECK (EndDate > StartDate)
);
GO

-- Dim_Student
CREATE TABLE dbo.Dim_Student (
    StudentKey INT IDENTITY(1,1) PRIMARY KEY,
    StudentNaturalID VARCHAR(20) NOT NULL, -- NIM
    FullName VARCHAR(100) NOT NULL,
    DOB DATE NOT NULL,
    Gender CHAR(1) NOT NULL CHECK (Gender IN ('M','F')),
    EntryYear INT NOT NULL,
    [Status] VARCHAR(20) NOT NULL,
    ProgramNaturalID VARCHAR(20) NOT NULL,
    CONSTRAINT UQ_Dim_Student_NIM UNIQUE (StudentNaturalID),
    CONSTRAINT CHK_Dim_Student_EntryYr CHECK (EntryYear >= 1900)
);
GO

-- Dim_Course
CREATE TABLE dbo.Dim_Course (
    CourseKey INT IDENTITY(1,1) PRIMARY KEY,
    CourseCode VARCHAR(20) NOT NULL,
    CourseName VARCHAR(150) NOT NULL,
    Credits INT NOT NULL,
    ProgramNaturalID VARCHAR(20) NOT NULL,
    CONSTRAINT UQ_Dim_Course_Code UNIQUE (CourseCode),

```

```

CONSTRAINT CHK_Dim_Course_Credits CHECK (Credits > 0)
);
GO

-- Dim_Instructor
CREATE TABLE dbo.Dim_Instructor (
    InstructorKey      INT IDENTITY(1,1) PRIMARY KEY,
    InstructorNaturalID VARCHAR(20) NOT NULL,
    [Name]              VARCHAR(100) NOT NULL,
    [Rank]              VARCHAR(50) NULL,
    Dept                VARCHAR(100) NULL,
    FTE                 DECIMAL(3,2) NOT NULL DEFAULT 1.00,
    CONSTRAINT UQ_Dim_Instructor_ID UNIQUE (InstructorNaturalID),
    CONSTRAINT CHK_Dim_Instructor_FTE CHECK (FTE BETWEEN 0 AND 1)
);
GO

```

Setelah itu dibuat berbagai dimension tables seperti program studi, tanggal, semester, mahasiswa, mata kuliah, dan dosen. Setiap tabel dimensi memiliki atribut penting, aturan validasi, serta unique constraint untuk menjaga kualitas data.

### **Memasukkan data ke dalam tabel dimensi**

Query:

```

-- ISI DATA KEDALAM DIMENSI
-- Dim_Program
INSERT INTO dbo.Dim_Program (ProgramCode, ProgramName, Faculty)
VALUES
('TI', 'Teknik Informatika',      'Fakultas Sains dan Teknologi'),
('SI', 'Sistem Informasi',        'Fakultas Sains dan Teknologi'),
('MI', 'Manajemen Informatika',   'Fakultas Vokasi'),
('KM', 'Kimia',                  'Fakultas Sains dan Teknologi'),
('FI', 'Fisika',                  'Fakultas Sains dan Teknologi');
GO

-- Dim_Date (tahun 2018–2025)
DECLARE @Start DATE = '2018-01-01';
DECLARE @End   DATE = '2025-12-31';

WHILE @Start <= @End
BEGIN
    INSERT INTO dbo.Dim_Date (DateKey, [Date], [Day], [Month], [Quarter], [Year],
    SemesterCode)

```

```

VALUES (
    CONVERT(INT, FORMAT(@Start, 'yyyyMMdd')),
    @Start,
    DATEPART(DAY, @Start),
    DATEPART(MONTH, @Start),
    DATEPART(QUARTER, @Start),
    DATEPART(YEAR, @Start),
    NULL -- bisa diupdate nanti sesuai mapping semester
);

SET @Start = DATEADD(DAY, 1, @Start);
END;
GO

-- Dim_Semester
INSERT INTO dbo.Dim_Semester (SemesterCode, StartDate, EndDate)
VALUES
('2018-Ganjil', '2018-08-01', '2019-01-31'),
('2018-Genap', '2019-02-01', '2019-07-31'),
('2019-Ganjil', '2019-08-01', '2020-01-31'),
('2019-Genap', '2020-02-01', '2020-07-31'),
('2020-Ganjil', '2020-08-01', '2021-01-31'),
('2020-Genap', '2021-02-01', '2021-07-31'),
('2021-Ganjil', '2021-08-01', '2022-01-31'),
('2021-Genap', '2022-02-01', '2022-07-31'),
('2022-Ganjil', '2022-08-01', '2023-01-31'),
('2022-Genap', '2023-02-01', '2023-07-31'),
('2023-Ganjil', '2023-08-01', '2024-01-31'),
('2023-Genap', '2024-02-01', '2024-07-31');
GO

-- Dim_Instructor
INSERT INTO dbo.Dim_Instructor (
    InstructorNaturalID, [Name], [Rank], Dept, FTE
)
VALUES
('INS001', 'Dr. Andi', 'Lektor Kepala', 'Teknik Informatika', 1.00),
('INS002', 'Dr. Budi', 'Lektor', 'Sistem Informasi', 1.00),
('INS003', 'Dr. Citra', 'Lektor', 'Manajemen Informatika', 0.75),
('INS004', 'Dr. Diana', 'Guru Besar', 'Kimia', 1.00),
('INS005', 'Dr. Eko', 'Asisten Ahli', 'Fisika', 0.50),
('INS006', 'Dr. Fitri', 'Lektor', 'Teknik Informatika', 1.00),
('INS007', 'Dr. Gita', 'Lektor', 'Sistem Informasi', 1.00),

```

```

('INS008', 'Dr. Hasan',    'Lektor Kepala', 'Kimia',      1.00),
('INS009', 'Dr. Intan',    'Asisten Ahli',   'Fisika',      0.80),
('INS010', 'Dr. Joko',     'Lektor',        'Manajemen Informatika', 1.00);
GO

```

```

-- Dim_Course
INSERT INTO dbo.Dim_Course (CourseCode, CourseName, Credits, ProgramNaturalID)
VALUES
('TI101', 'Pengantar Pemrograman', 3, 'TI'),
('TI102', 'Struktur Data', 3, 'TI'),
('TI201', 'Basis Data', 3, 'TI'),
('SI101', 'Dasar Sistem Informasi', 3, 'SI'),
('SI102', 'Analisis Proses Bisnis', 3, 'SI'),
('MI101', 'Algoritma dan Logika', 3, 'MI'),
('KM101', 'Kimia Dasar', 3, 'KM'),
('KM201', 'Kimia Organik', 3, 'KM'),
('FI101', 'Fisika Dasar', 3, 'FI'),
('FI201', 'Fisika Modern', 3, 'FI');
GO

```

```

--Dim_Student (1.000 mahasiswa)
;WITH N AS (
    SELECT TOP (1000)
        ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS n
    FROM sys.all_objects
)
INSERT INTO dbo.Dim_Student (
    StudentNaturalID, FullName, DOB, Gender, EntryYear, [Status], ProgramNaturalID
)
SELECT
    CONCAT('NIM', RIGHT('000000' + CAST(n AS VARCHAR(6)), 6)) AS StudentNaturalID,
    CONCAT('Mahasiswa ', n) AS FullName,
    DATEADD(DAY, -1 * (ABS(CHECKSUM(NEWID())) % 9000), '2000-01-01') AS DOB,
    CASE WHEN ABS(CHECKSUM(NEWID())) % 2 = 0 THEN 'M' ELSE 'F' END AS
Gender,
    2018 + (ABS(CHECKSUM(NEWID())) % 6) AS EntryYear, -- 2018–2023
    CASE
        WHEN ABS(CHECKSUM(NEWID())) % 100 < 80 THEN 'Aktif'
        WHEN ABS(CHECKSUM(NEWID())) % 100 < 90 THEN 'Cuti'
        ELSE 'Lulus'
    END AS [Status],
    CASE

```

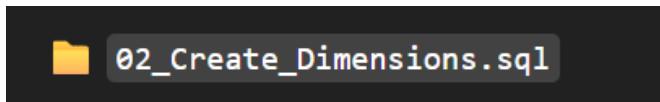
```

WHEN n % 5 = 1 THEN 'TI'
WHEN n % 5 = 2 THEN 'SI'
WHEN n % 5 = 3 THEN 'MI'
WHEN n % 5 = 4 THEN 'KM'
ELSE 'FI'
END AS ProgramNaturalID;
GO

```

Setelah semua dimensi dibuat, kode tersebut memasukkan data contoh, mulai dari daftar program studi, kalender tanggal secara lengkap dari 2018 hingga 2025, daftar semester, instruktur, mata kuliah, hingga 1000 mahasiswa dengan atribut acak seperti gender, DOB, entry year, status, dan program.

Dokumentasi:



Dimensi program

```

-- Dim_Program
CREATE TABLE dbo.Dim_Program (
    ProgramKey INT IDENTITY(1,1) PRIMARY KEY,
    ProgramCode VARCHAR(20) NOT NULL,
    ProgramName VARCHAR(100) NOT NULL,
    Faculty VARCHAR(100) NOT NULL,
    CONSTRAINT UQ_Dim_Program_Code UNIQUE (ProgramCode)
);
GO

```

Dimensi Date

The screenshot shows the Object Explorer on the left with the database 'akademik' selected. The central pane displays a SQL query window titled 'SQLQuery1...Pongo (76)\*'. The query creates the 'Dim\_Date' table with columns for DateKey (INT PRIMARY KEY), Date (DATE NOT NULL), Day (INT NOT NULL), Month (INT NOT NULL), Quarter (INT NOT NULL), Year (INT NOT NULL), and SemesterCode (VARCHAR(20) NULL). It includes constraints: UQ\_Dim\_Date for the Date column, CHK\_Dim\_Date\_Month for Month between 1 and 12, and CHK\_Dim\_Date\_Quarter for Quarter between 1 and 4. The command concludes with 'GO'. The status bar at the bottom right indicates 'Ln: 14, Ch: 3'.

```
-- Dim_Date
CREATE TABLE dbo.Dim_Date (
    DateKey INT PRIMARY KEY,          -- yyyyymmdd
    [Date] DATE NOT NULL,
    [Day] INT NOT NULL,
    [Month] INT NOT NULL,
    [Quarter] INT NOT NULL,
    [Year] INT NOT NULL,
    SemesterCode VARCHAR(20) NULL,
    CONSTRAINT UQ_Dim_Date UNIQUE ([Date]),
    CONSTRAINT CHK_Dim_Date_Month CHECK ([Month] BETWEEN 1 AND 12),
    CONSTRAINT CHK_Dim_Date_Quarter CHECK ([Quarter] BETWEEN 1 AND 4)
);
GO
```

## Dimensi Semester

The screenshot shows the Object Explorer on the left with the database 'akademik' selected. The central pane displays a SQL query window titled 'SQLQuery1.sq...Pongo (76)\*'. The query creates the 'Dim\_Semester' table with columns for SemesterKey (INT IDENTITY(1,1) PRIMARY KEY), SemesterCode (VARCHAR(20) NOT NULL), StartDate (DATE NOT NULL), and EndDate (DATE NOT NULL). It includes constraints: UQ\_Dim\_Semester\_Code for SemesterCode, and CHK\_Dim\_Semester\_Dates for EndDate > StartDate. The command concludes with 'GO'. The status bar at the bottom right indicates 'Ln: 10, Ch: 3'.

```
-- Dim_Semester
CREATE TABLE dbo.Dim_Semester (
    SemesterKey INT IDENTITY(1,1) PRIMARY KEY,
    SemesterCode VARCHAR(20) NOT NULL,
    StartDate DATE NOT NULL,
    EndDate DATE NOT NULL,
    CONSTRAINT UQ_Dim_Semester_Code UNIQUE (SemesterCode),
    CONSTRAINT CHK_Dim_Semester_Dates CHECK (EndDate > StartDate)
);
GO
```

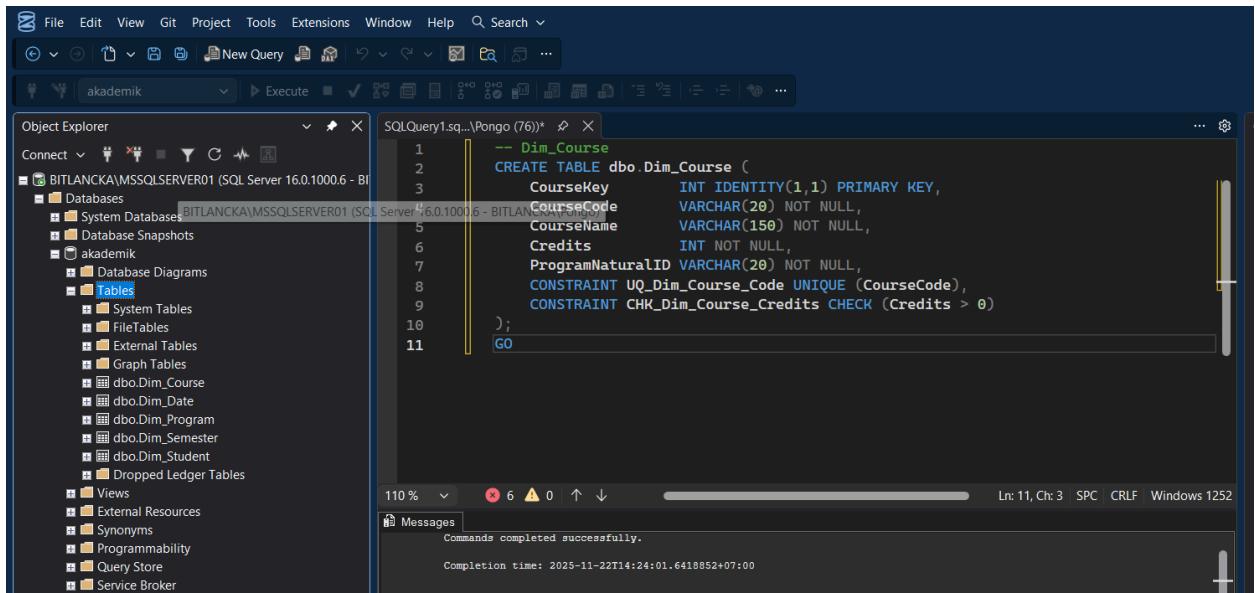
## Dimensi Student

The screenshot shows the SQL Server Management Studio (SSMS) interface. On the left, the Object Explorer pane displays the database structure for 'akademik'. In the center, the SQL Query Editor window contains the following T-SQL code:

```
-- Dim_Student
CREATE TABLE dbo.Dim_Student (
    StudentKey      INT IDENTITY(1,1) PRIMARY KEY,
    StudentNaturalID VARCHAR(20) NOT NULL,      -- NIM
    FullName        VARCHAR(100) NOT NULL,
    DOB             DATE NOT NULL,
    Gender          CHAR(1) NOT NULL CHECK (Gender IN ('M','F')),
    EntryYear       INT NOT NULL,
    [Status]         VARCHAR(20) NOT NULL,
    ProgramNaturalID VARCHAR(20) NOT NULL,
    CONSTRAINT UQ_Dim_Student_NIM      UNIQUE (StudentNaturalID),
    CONSTRAINT CHK_Dim_Student_EntryYr CHECK (EntryYear >= 1900)
);
GO
```

The status bar at the bottom indicates the command completed successfully with a completion time of 2023-11-22T14:23:46.8022447+07:00.

## Dimensi Course

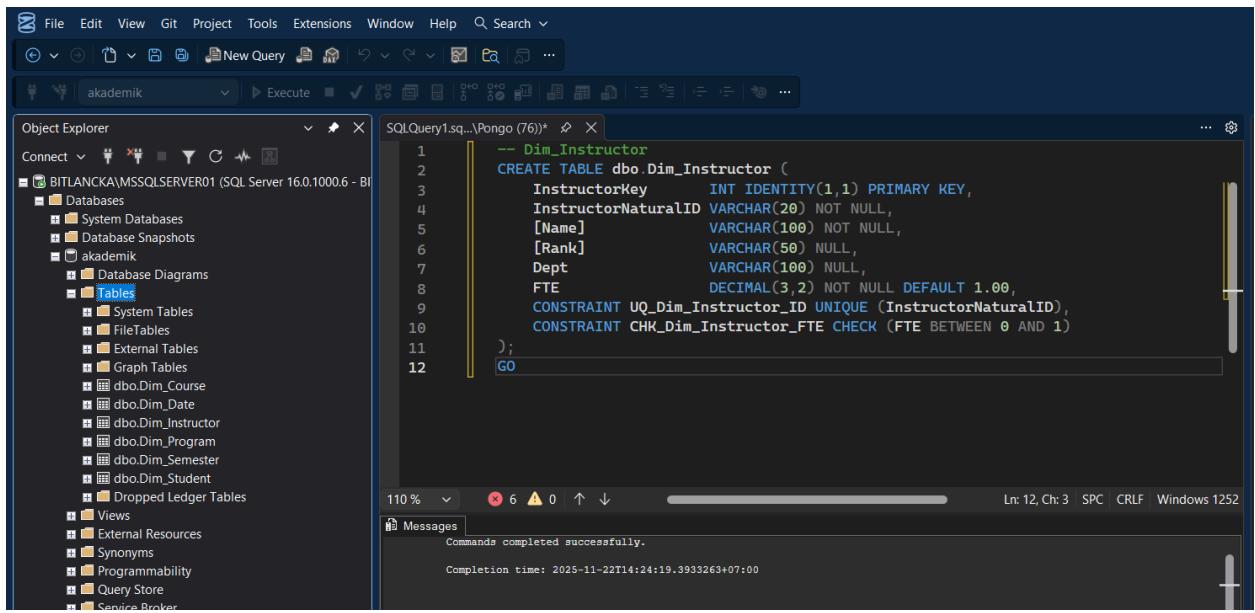


The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left shows a database named 'akademik' containing various tables like 'dbo.Dim\_Course', 'dbo.Dim\_Date', etc. The central pane displays a T-SQL script for creating the 'Dim\_Course' table:

```
-- Dim_Course
CREATE TABLE dbo.Dim_Course (
    CourseKey INT IDENTITY(1,1) PRIMARY KEY,
    CourseCode VARCHAR(20) NOT NULL,
    CourseName VARCHAR(150) NOT NULL,
    Credits INT NOT NULL,
    ProgramNaturalID VARCHAR(20) NOT NULL,
    CONSTRAINT UQ_Dim_Course_Code UNIQUE (CourseCode),
    CONSTRAINT CHK_Dim_Course_Credits CHECK (Credits > 0)
);
GO
```

The status bar at the bottom indicates the command completed successfully with a completion time of 2025-11-22T14:24:01.6418852+07:00.

## Dimensi Instructor



The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left shows a database named 'akademik' containing various tables like 'dbo.Dim\_Instructor', 'dbo.Dim\_Date', etc. The central pane displays a T-SQL script for creating the 'Dim\_Instructor' table:

```
-- Dim_Instructor
CREATE TABLE dbo.Dim_Instructor (
    InstructorKey INT IDENTITY(1,1) PRIMARY KEY,
    InstructorNaturalID VARCHAR(20) NOT NULL,
    [Name] VARCHAR(100) NOT NULL,
    [Rank] VARCHAR(50) NULL,
    Dept VARCHAR(100) NULL,
    FTE DECIMAL(3, 2) NOT NULL DEFAULT 1.00,
    CONSTRAINT UQ_Dim_Instructor_ID UNIQUE (InstructorNaturalID),
    CONSTRAINT CHK_Dim_Instructor_FTE CHECK (FTE BETWEEN 0 AND 1)
);
GO
```

The status bar at the bottom indicates the command completed successfully with a completion time of 2025-11-22T14:24:19.3933263+07:00.

### 3. Create Fact Tables

Rincian data untuk tiap tabel fact adalah sebagai berikut:

- fact enrollment: 20.000 data
- fact\_admission: 8000 data
- fact\_graduation: 3000 data

Query:

```
-- Buat Tabel Fakta
-- Fact_Enrollment
CREATE TABLE dbo.Fact_Enrollment (
    EnrollmentID INT IDENTITY(1,1) PRIMARY KEY,
    StudentKey INT NOT NULL,
    CourseKey INT NOT NULL,
    SemesterKey INT NOT NULL,
    InstructorKey INT NOT NULL,
    DateKey INT NOT NULL,
    Grade VARCHAR(2) NULL,
    NumericGrade DECIMAL(4,2) NULL,
    AttendanceRate DECIMAL(5,2) NULL,
    [Status] VARCHAR(20) NULL,
    CONSTRAINT FK_Fact_Enroll_Student FOREIGN KEY (StudentKey) REFERENCES
    dbo.Dim_Student(StudentKey),
    CONSTRAINT FK_Fact_Enroll_Course FOREIGN KEY (CourseKey) REFERENCES
    dbo.Dim_Course(CourseKey),
    CONSTRAINT FK_Fact_Enroll_Semester FOREIGN KEY (SemesterKey)
    REFERENCES dbo.Dim_Semester(SemesterKey),
    CONSTRAINT FK_Fact_Enroll_Instructor FOREIGN KEY (InstructorKey)
    REFERENCES dbo.Dim_Instructor(InstructorKey),
    CONSTRAINT FK_Fact_Enroll_Date FOREIGN KEY (DateKey) REFERENCES
    dbo.Dim_Date(DateKey),
    CONSTRAINT CHK_Fact_Enroll_Grade CHECK (NumericGrade BETWEEN 0 AND
    4),
    CONSTRAINT CHK_Fact_Enroll_Attend CHECK (AttendanceRate BETWEEN 0 AND
    100)
);
GO

-- Fact_Graduation
CREATE TABLE dbo.Fact_Graduation (
    GraduationID INT IDENTITY(1,1) PRIMARY KEY,
    StudentKey INT NOT NULL,
```

```

DateKey      INT NOT NULL,      -- mis. tanggal yudisium
ProgramKey    INT NOT NULL,
GraduationDateKey INT NOT NULL,      -- tanggal wisuda
GPA          DECIMAL(4,2) NOT NULL,
TotalCredits  INT NOT NULL,
StudyDuration INT NULL,           -- bulan
Honors        VARCHAR(50) NULL,
ThesisScore   DECIMAL(5,2) NULL,
CONSTRAINT FK_Fact_Grad_Student FOREIGN KEY (StudentKey)
REFERENCES dbo.Dim_Student(StudentKey),
CONSTRAINT FK_Fact_Grad_Date   FOREIGN KEY (DateKey)      REFERENCES
dbo.Dim_Date(DateKey),
CONSTRAINT FK_Fact_Grad_GradDate FOREIGN KEY (GraduationDateKey)
REFERENCES dbo.Dim_Date(DateKey),
CONSTRAINT FK_Fact_Grad_Program FOREIGN KEY (ProgramKey)
REFERENCES dbo.Dim_Program(ProgramKey),
CONSTRAINT CHK_Fact_Grad_GPA   CHECK (GPA BETWEEN 0 AND 4),
CONSTRAINT CHK_Fact_Grad_Credits CHECK (TotalCredits > 0)
);
GO

-- Fact_Admission
CREATE TABLE dbo.Fact_Admission (
AdmissionID   INT IDENTITY(1,1) PRIMARY KEY,
StudentKey     INT NOT NULL,
ProgramKey     INT NOT NULL,
ApplicationDateKey INT NOT NULL,
AdmissionDateKey INT NOT NULL,
TestScore      DECIMAL(5,2) NULL,
InterviewScore DECIMAL(5,2) NULL,
HighSchoolGPA  DECIMAL(4,2) NULL,
AdmissionStatus VARCHAR(20) NOT NULL,
AdmissionType   VARCHAR(50) NULL, -- SNMPTN/SBMPTN/Mandiri
ProcessingDays  INT NULL,
CONSTRAINT FK_Fact_Adm_Student FOREIGN KEY (StudentKey)
REFERENCES dbo.Dim_Student(StudentKey),
CONSTRAINT FK_Fact_Adm_Program FOREIGN KEY (ProgramKey)
REFERENCES dbo.Dim_Program(ProgramKey),
CONSTRAINT FK_Fact_Adm_AppDate FOREIGN KEY (ApplicationDateKey)
REFERENCES dbo.Dim_Date(DateKey),
CONSTRAINT FK_Fact_Adm_AdmDate FOREIGN KEY (AdmissionDateKey)
REFERENCES dbo.Dim_Date(DateKey),
CONSTRAINT CHK_Fact_Adm_Test   CHECK (TestScore    >= 0),

```

```

CONSTRAINT CHK_Fact_Adm_Interview CHECK (InterviewScore >= 0),
CONSTRAINT CHK_Fact_Adm_HSGPA    CHECK (HighSchoolGPA BETWEEN 0 AND
4)
);
GO

```

Kode selanjutnya membuat tiga tabel fakta utama, yakni Fact\_Enrollment, Fact\_Graduation, dan Fact\_Admission. Masing-masing tabel menghubungkan *surrogate keys* dari dimensi dan menyimpan data transaksi seperti KRS mahasiswa, informasi kelulusan, dan proses penerimaan mahasiswa baru.

### Memasukkan data

```

-- data untuk Fact_Admission 8.000 baris
USE DM_AKADEMIK_DW;
GO

DECLARE @RowCountAdm INT = 8000; -- ubah kalau mau jumlah lain

;WITH Numbers AS (
    SELECT TOP (@RowCountAdm)
        ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS n
    FROM sys.all_objects ao1
    CROSS JOIN sys.all_objects ao2
)
INSERT INTO dbo.Fact_Admission (
    StudentKey,
    ProgramKey,
    ApplicationDateKey,
    AdmissionDateKey,
    TestScore,
    InterviewScore,
    HighSchoolGPA,
    AdmissionStatus,
    AdmissionType,
    ProcessingDays
)
SELECT
    s.StudentKey,
    p.ProgramKey,
    app.DateKey      AS ApplicationDateKey,
    adm.DateKey      AS AdmissionDateKey,
    -- skor 0–100 (dua desimal)

```

```

        CAST((ABS(CHECKSUM(NEWID())) % 10001) / 100.0 AS DECIMAL(5,2)) AS
TestScore,
        CAST((ABS(CHECKSUM(NEWID())) % 10001) / 100.0 AS DECIMAL(5,2)) AS
InterviewScore,
        -- HighSchoolGPA 2.00–4.00
        CAST((200 + (ABS(CHECKSUM(NEWID())) % 201)) / 100.0 AS DECIMAL(4,2)) AS
HighSchoolGPA,
        CASE
            WHEN randStatus.RandStatus < 60 THEN 'Accepted'
            WHEN randStatus.RandStatus < 85 THEN 'Rejected'
            ELSE 'Waiting List'
        END AS AdmissionStatus,
        CASE
            WHEN randType.RandType < 35 THEN 'SNMPTN'
            WHEN randType.RandType < 70 THEN 'SBMPTN'
            WHEN randType.RandType < 90 THEN 'Mandiri'
            ELSE 'Beasiswa'
        END AS AdmissionType,
        DATEDIFF(DAY, app.[Date], adm.[Date]) AS ProcessingDays
FROM Numbers n
-- ambil student + program natural ID-nya
CROSS APPLY (
    SELECT TOP 1 StudentKey, ProgramNaturalID
    FROM dbo.Dim_Student
    ORDER BY NEWID()
) AS s
-- map ProgramNaturalID -> ProgramKey
CROSS APPLY (
    SELECT TOP 1 ProgramKey
    FROM dbo.Dim_Program
    WHERE ProgramCode = s.ProgramNaturalID
    ORDER BY ProgramKey
) AS p
-- pilih application date (dibatasi supaya +60 hari masih dalam range dim tanggal)
CROSS APPLY (
    SELECT TOP 1 DateKey, [Date]
    FROM dbo.Dim_Date
    WHERE [Date] <= '2025-10-31'
    ORDER BY NEWID()
) AS app
-- random offset 0–60 hari
CROSS APPLY (
    SELECT ABS(CHECKSUM(NEWID())) % 61 AS DaysOffset

```

```

) AS ofs
-- admission date = application date + offset
CROSS APPLY (
    SELECT DateKey, [Date]
    FROM dbo.Dim_Date
    WHERE [Date] = DATEADD(DAY, ofs.DaysOffset, app.[Date])
) AS adm
-- random status dan admission type
CROSS APPLY (SELECT ABS(CHECKSUM(NEWID())) % 100 AS RandStatus) AS
randStatus
CROSS APPLY (SELECT ABS(CHECKSUM(NEWID())) % 100 AS RandType) AS
randType;
GO

-- check hasilnya
SELECT COUNT(*) AS JumlahAdmission
FROM dbo.Fact_Admission;

-- data untuk Fact_Graduation 3.000 baris
USE DM_AKADEMIK_DW;
GO

DECLARE @RowCountGrad INT = 3000; -- ubah kalau mau jumlah lain

;WITH Numbers AS (
    SELECT TOP (@RowCountGrad)
        ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS n
    FROM sys.all_objects ao1
    CROSS JOIN sys.all_objects ao2
)
INSERT INTO dbo.Fact_Graduation (
    StudentKey,
    DateKey,
    ProgramKey,
    GraduationDateKey,
    GPA,
    TotalCredits,
    StudyDuration,
    Honors,
    ThesisScore
)

```

```

SELECT
    s.StudentKey,
    yud.DateKey      AS DateKey,          -- tanggal yudisium
    p.ProgramKey,
    grad.DateKey     AS GraduationDateKey, -- tanggal wisuda
    -- GPA 2.00–4.00
    CAST((200 + (ABS(CHECKSUM(NEWID())) % 201)) / 100.0 AS DECIMAL(4,2)) AS
GPA,
    -- total SKS 120–160
    120 + (ABS(CHECKSUM(NEWID())) % 41) AS TotalCredits,
    -- lama studi 36–72 bulan
    36 + (ABS(CHECKSUM(NEWID())) % 37) AS StudyDuration,
    CASE
        WHEN randHonors.RandHonors < 50 THEN NULL
        WHEN randHonors.RandHonors < 80 THEN 'Cum Laude'
        WHEN randHonors.RandHonors < 95 THEN 'Magna Cum Laude'
        ELSE 'Summa Cum Laude'
    END AS Honors,
    -- nilai skripsi 60–100
    CAST((6000 + (ABS(CHECKSUM(NEWID())) % 4001)) / 100.0 AS DECIMAL(5,2)) AS
ThesisScore
FROM Numbers n
-- ambil student + ProgramNaturalID
CROSS APPLY (
    SELECT TOP 1 StudentKey, ProgramNaturalID, EntryYear
    FROM dbo.Dim_Student
    WHERE [Status] = 'Lulus' OR [Status] = 'Aktif'
    ORDER BY NEWID()
) AS s
-- map ProgramNaturalID -> ProgramKey
CROSS APPLY (
    SELECT TOP 1 ProgramKey
    FROM dbo.Dim_Program
    WHERE ProgramCode = s.ProgramNaturalID
    ORDER BY ProgramKey
) AS p
-- pilih graduation date antara 2022–2025
CROSS APPLY (
    SELECT TOP 1 DateKey, [Date]
    FROM dbo.Dim_Date
    WHERE [Date] BETWEEN '2022-01-01' AND '2025-12-31'
    ORDER BY NEWID()
) AS grad

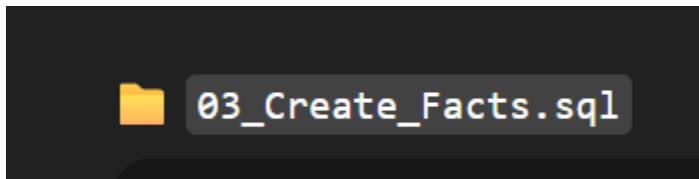
```

```
-- offset yudisium 0–90 hari sebelum wisuda
CROSS APPLY (
    SELECT ABS(CHECKSUM(NEWID())) % 91 AS OffsetDays
) AS ofs
CROSS APPLY (
    SELECT TOP 1 DateKey, [Date]
    FROM dbo.Dim_Date
    WHERE [Date] = DATEADD(DAY, -ofs.OffsetDays, grad.[Date])
    ORDER BY [Date]
) AS yud
CROSS APPLY (
    SELECT ABS(CHECKSUM(NEWID())) % 100 AS RandHonors
) AS randHonors;
GO

-- check jumlahnya
SELECT COUNT(*) AS JumlahGraduation
FROM dbo.Fact_Graduation;
```

Setelah struktur DW selesai, kode melakukan proses *data generation* untuk membuat ribuan baris data fiktif dengan menggunakan randomisasi terkontrol (melalui NEWID, RAND, dan CROSS APPLY), sehingga menghasilkan dataset analitis yang realistik.

Dokumentasi:



Fakta Admission

Fakta Admission

```

-- Fact_Admission
CREATE TABLE dbo.Fact_Admission (
    AdmissionID INT IDENTITY(1,1) PRIMARY KEY,
    StudentKey INT NOT NULL,
    ProgramKey INT NOT NULL,
    ApplicationDateKey INT NOT NULL,
    AdmissionDateKey INT NOT NULL,
    TestScore DECIMAL(5,2) NULL,
    InterviewScore DECIMAL(5,2) NULL,
    HighSchoolGPA DECIMAL(4,2) NULL,
    AdmissionStatus VARCHAR(20) NOT NULL,
    AdmissionType VARCHAR(50) NULL, -- SNMPTN/SBMPTN/Mandiri
    ProcessingDays INT NULL,
    CONSTRAINT FK_Fact_Adm_Student FOREIGN KEY (StudentKey) REFERENCES dim_student,
    CONSTRAINT FK_Fact_Adm_Program FOREIGN KEY (ProgramKey) REFERENCES dim_program,
    CONSTRAINT FK_Fact_Adm_AppDate FOREIGN KEY (ApplicationDateKey) REFERENCES dim_date,
    CONSTRAINT FK_Fact_Adm_AdmDate FOREIGN KEY (AdmissionDateKey) REFERENCES dim_date,
    CONSTRAINT CHK_Fact_Adm_Test CHECK (TestScore >= 0)
)
Ln 22, Ch 3 (1186 chars, 22 lines) SPC CRLF Windows 1252

```

Messages

Commands completed successfully.

Completion time: 2025-11-22T14:25:25.1656017+07:00

## Fakta Graduation

Fakta Graduation

```

-- Fact_Graduation
CREATE TABLE dbo.Fact_Graduation (
    GraduationID INT IDENTITY(1,1) PRIMARY KEY,
    StudentKey INT NOT NULL,
    DateKey INT NOT NULL, -- mis. tanggal yudisium
    ProgramKey INT NOT NULL,
    GraduationDateKey INT NOT NULL, -- tanggal wisuda
    GPA DECIMAL(4,2) NOT NULL,
    TotalCredits INT NOT NULL,
    StudyDuration INT NULL, -- bulan
    Honors VARCHAR(50) NULL,
    ThesisScore DECIMAL(5,2) NULL,
    CONSTRAINT FK_Fact_Grad_Student FOREIGN KEY (StudentKey) REFERENCES dim_student,
    CONSTRAINT FK_Fact_Grad_Date FOREIGN KEY (DateKey) REFERENCES dim_date,
    CONSTRAINT FK_Fact_Grad_GradDate FOREIGN KEY (GraduationDateKey) REFERENCES dim_date,
    CONSTRAINT FK_Fact_Grad_Program FOREIGN KEY (ProgramKey) REFERENCES dim_program,
    CONSTRAINT CHK_Fact_Grad_GPA CHECK (GPA BETWEEN 0 AND 4),
    CONSTRAINT CHK_Fact_Grad_Credits CHECK (TotalCredits > 0)
)
Ln 20, Ch 3 (1110 chars, 20 lines) SPC CRLF Windows 1252

```

Messages

Commands completed successfully.

Completion time: 2025-11-22T14:25:04.7789951+07:00

## Fakta Enrollment

```
-- Fact_Enrollment
CREATE TABLE dbo.Fact_Enrollment (
    EnrollmentID INT IDENTITY(1,1) PRIMARY KEY,
    StudentKey INT NOT NULL,
    CourseKey INT NOT NULL,
    SemesterKey INT NOT NULL,
    InstructorKey INT NOT NULL,
    DateKey INT NOT NULL,
    Grade VARCHAR(2) NULL,
    NumericGrade DECIMAL(4,2) NULL,
    AttendanceRate DECIMAL(5, 2) NULL,
    [Status] VARCHAR(20) NULL,
    CONSTRAINT FK_Fact_Enroll_Student FOREIGN KEY (StudentKey) REFERENCES dbo.Student,
    CONSTRAINT FK_Fact_Enroll_Course FOREIGN KEY (CourseKey) REFERENCES dbo.Course,
    CONSTRAINT FK_Fact_Enroll_Semester FOREIGN KEY (SemesterKey) REFERENCES dbo.Semester,
    CONSTRAINT FK_Fact_Enroll_Instructor FOREIGN KEY (InstructorKey) REFERENCES dbo.Instructor,
    CONSTRAINT FK_Fact_Enroll_Date FOREIGN KEY (DateKey) REFERENCES dbo.Date,
    CONSTRAINT CHW_Fact_Enroll_Grade CHECK (NumericGrade BETWEEN 0 AND 4)
)
Ln: 21, Ch: 3 (1145 chars, 21 lines) SPC CRLF Windows 1252
```

## Step 2: Indexing Strategy

Setelah itu dibuat berbagai indeks, baik *nonclustered* maupun *columnstore*, untuk memastikan performa analitik tetap cepat meskipun data besar.

### 1. Clustered Index on Fact Table

```
-- 1. NONCLUSTERED INDEXES (DateKey + PK)
-- (PK tetap clustered)
```

```
CREATE NONCLUSTERED INDEX IX_Fact_Enrollment_DateKey
ON dbo.Fact_Enrollment(DateKey, EnrollmentID);
GO
```

```
CREATE NONCLUSTERED INDEX IX_Fact_Graduation_DateKey
ON dbo.Fact_Graduation(GraduationDateKey, GraduationID);
GO
```

```
CREATE NONCLUSTERED INDEX IX_Fact_Admission_DateKey
ON dbo.Fact_Admission(AdmissionDateKey, AdmissionID);
GO
```

### 2. Non-Clustered Indexes

```
-----  
-- 2. NONCLUSTERED INDEXES (join + analytics)  
-----  
  
-- Fact_Enrollment  
CREATE NONCLUSTERED INDEX IX_Fact_Enroll_Student  
ON dbo.Fact_Enrollment(StudentKey)  
INCLUDE (CourseKey, SemesterKey, NumericGrade);  
GO  
  
CREATE NONCLUSTERED INDEX IX_Fact_Enroll_Course  
ON dbo.Fact_Enrollment(CourseKey)  
INCLUDE (StudentKey, NumericGrade, AttendanceRate);  
GO  
  
CREATE NONCLUSTERED INDEX IX_Fact_Enroll_Semester  
ON dbo.Fact_Enrollment(SemesterKey, DateKey)  
INCLUDE (StudentKey, CourseKey);  
GO  
  
-- Fact_Graduation  
CREATE NONCLUSTERED INDEX IX_Fact_Grad_Student  
ON dbo.Fact_Graduation(StudentKey)  
INCLUDE (GPA, TotalCredits, StudyDuration);  
GO  
  
CREATE NONCLUSTERED INDEX IX_Fact_Grad_Program  
ON dbo.Fact_Graduation(ProgramKey, GraduationDateKey);  
GO  
  
-- Fact_Admission  
CREATE NONCLUSTERED INDEX IX_Fact_Adm_Student  
ON dbo.Fact_Admission(StudentKey)  
INCLUDE (AdmissionStatus, AdmissionType);  
GO  
  
CREATE NONCLUSTERED INDEX IX_Fact_Adm_Program  
ON dbo.Fact_Admission(ProgramKey, AdmissionDateKey)  
INCLUDE (AdmissionStatus);  
GO
```

### 3. Columnstore Index (for large fact tables)

```

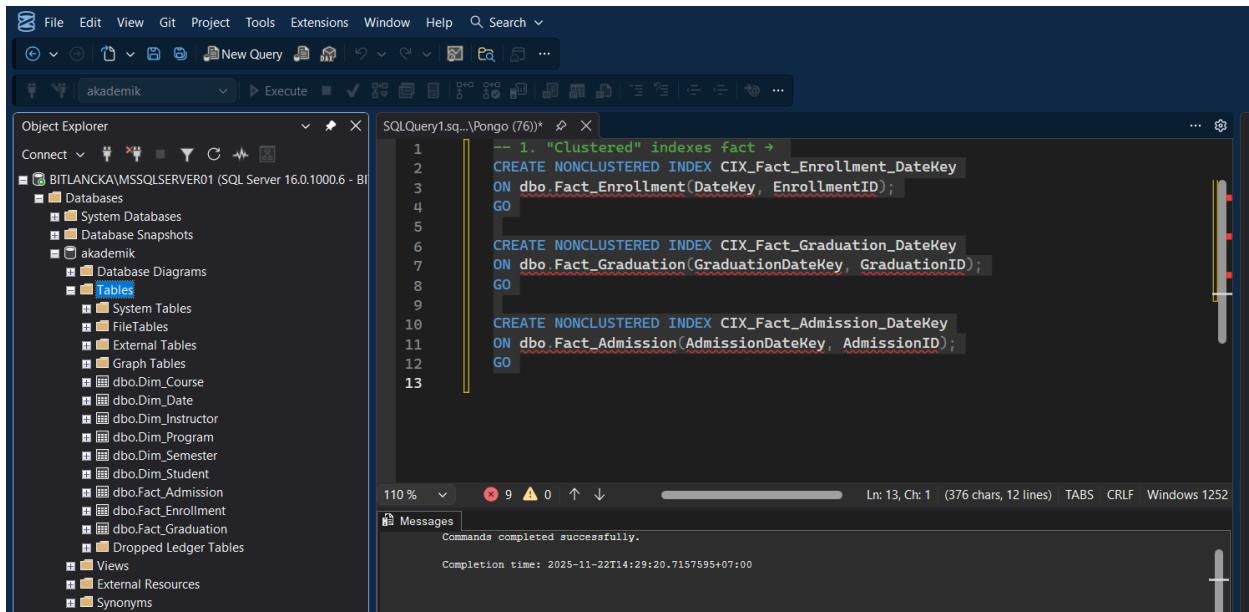
-----
-- 3. COLUMNSTORE INDEX (analytical)
-----
IF OBJECT_ID('dbo.Fact_Enrollment_Partitioned', 'U') IS NOT NULL
    DROP TABLE dbo.Fact_Enrollment_Partitioned;
GO

CREATE TABLE dbo.Fact_Enrollment_Partitioned (
    EnrollmentID INT IDENTITY(1,1) NOT NULL,
    StudentKey INT NOT NULL,
    CourseKey INT NOT NULL,
    SemesterKey INT NOT NULL,
    InstructorKey INT NOT NULL,
    DateKey INT NOT NULL,
    Grade VARCHAR(2) NULL,
    NumericGrade DECIMAL(4,2) NULL,
    AttendanceRate DECIMAL(5,2) NULL,
    [Status] VARCHAR(20) NULL,
    CONSTRAINT PK_Fact_Enrollment_Part
        PRIMARY KEY CLUSTERED (DateKey, EnrollmentID)
) ON PS_AcademicYear(DateKey);
GO

```

## Dokumentasi:

### 1. Clustered Index on Fact Table



The screenshot shows the SSMS interface with the following details:

- Object Explorer:** Shows the database structure for 'akademik' database, including tables like 'dbo.Dim\_Course', 'dbo.Dim\_Instructor', etc.
- Query Editor:** Contains T-SQL code for creating clustered indexes on fact tables.
- Messages:** Displays the message "Commands completed successfully." and the completion time.

```

-- 1. "Clustered" indexes fact
CREATE NONCLUSTERED INDEX CIX_Fact_Enrollment_DateKey
ON dbo.Fact_Enrollment(DateKey, EnrollmentID);
GO

CREATE NONCLUSTERED INDEX CIX_Fact_Graduation_DateKey
ON dbo.Fact_Graduation(GraduationDateKey, GraduationID);
GO

CREATE NONCLUSTERED INDEX CIX_Fact_Admission_DateKey
ON dbo.Fact_Admission(AdmissionDateKey, AdmissionID);
GO

```

## 2. Non-Clustered Indexes

The screenshot shows the SQL Server Management Studio (SSMS) interface. The Object Explorer on the left shows the database structure, including tables like Fact\_Enrollment, Fact\_Graduation, and Fact\_Admission. The central query editor window displays T-SQL code for creating non-clustered indexes:

```
1 -- Fact_Enrollment
2 CREATE NONCLUSTERED INDEX IX_Fact_Enroll_Student
3 ON dbo.Fact_Enrollment(StudentKey)
4 INCLUDE (CourseKey, SemesterKey, NumericGrade);
5 GO
6
7 CREATE NONCLUSTERED INDEX IX_Fact_Enroll_Course
8 ON dbo.Fact_Enrollment(CourseKey)
9 INCLUDE (StudentKey, NumericGrade, AttendanceRate);
10 GO
11
12 CREATE NONCLUSTERED INDEX IX_Fact_Enroll_Semester
13 ON dbo.Fact_Enrollment(SemesterKey)
14 INCLUDE (StudentKey, CourseKey);
15 GO
16
17 -- Fact_Graduation
18 CREATE NONCLUSTERED INDEX IX_Fact_Grad_Student
19 ON dbo.Fact_Graduation(StudentKey)
20 INCLUDE (GPA, TotalCredits, StudyDuration);
21 GO
22
23 CREATE NONCLUSTERED INDEX IX_Fact_Grad_Program
24 ON dbo.Fact_Graduation(ProgramKey);
25 GO
26
27 -- Fact_Admission
28 CREATE NONCLUSTERED INDEX IX_Fact_Ad_Student
29 ON dbo.Fact_Admission(StudentKey)
30 INCLUDE (AdmissionStatus, AdmissionType);
31 GO
32
33 CREATE NONCLUSTERED INDEX IX_Fact_Ad_Program
34 ON dbo.Fact_Admission(ProgramKey);
35 INCLUDE (AdmissionStatus);
36 GO
37
```

The status bar at the bottom indicates "Ln: 37, Ch: 3 (1029 chars, 37 lines) TABS CRLF Windows 1252".

## 3. Columnstore Index (for large fact tables)

The screenshot shows the SQL Server Management Studio (SSMS) interface. The Object Explorer on the left shows the database structure, including tables like Fact\_Enrollment, Fact\_Graduation, and Fact\_Admission. The central query editor window displays T-SQL code for creating a columnstore index:

```
-- CREATE columnstore index (FIXED)
CREATE NONCLUSTERED COLUMNSTORE INDEX NCCIX_Fact_Enrollment
ON dbo.Fact_Enrollment
(
    DateKey,
    StudentKey,
    CourseKey,
    SemesterKey,
    InstructorKey,
    NumericGrade,
    AttendanceRate,
    [Status]
);
GO
```

The status bar at the bottom indicates "Ln: 5, Ch: 13 SPC CRLF Windows 1252".

### Step 3: Partitioning Strategy

Kode juga membuat *partitioning* berdasarkan tahun akademik agar tabel fakta bisa di-*scale* dengan efisien. Misalkan yang akan dipartisi adalah Fact\_Enrolment per tahun akademik berbasis DateKey (yyyymmdd)

Query:

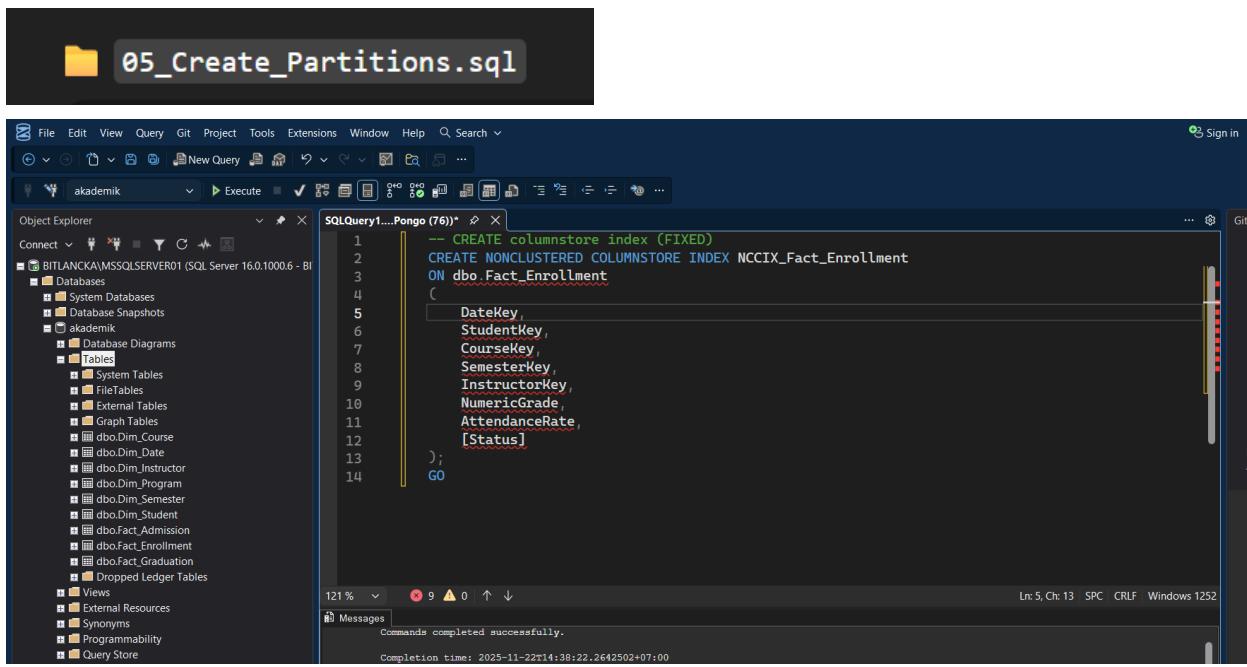
```
-- Partitioning

-- Partition function for Academic Year
CREATE PARTITION FUNCTION PF_AcademicYear (INT)
AS RANGE RIGHT FOR VALUES
(
    20200801, -- 2020/2021
    20210801, -- 2021/2022
    20220801, -- 2022/2023
    20230801, -- 2023/2024
    20240801, -- 2024/2025
    20250801 -- 2025/2026
);
GO

CREATE PARTITION SCHEME PS_AcademicYear
AS PARTITION PF_AcademicYear
ALL TO ([PRIMARY]);
GO

-- Example partitioned version of Fact_Enrollment
CREATE TABLE dbo.Fact_Enrollment_Partitioned (
    EnrollmentID INT IDENTITY(1,1) PRIMARY KEY,
    StudentKey INT NOT NULL,
    CourseKey INT NOT NULL,
    SemesterKey INT NOT NULL,
    InstructorKey INT NOT NULL,
    DateKey INT NOT NULL,
    Grade VARCHAR(2),
    NumericGrade DECIMAL(4,2),
    AttendanceRate DECIMAL(5,2),
    [Status] VARCHAR(20)
) ON PS_AcademicYear(DateKey);
GO
```

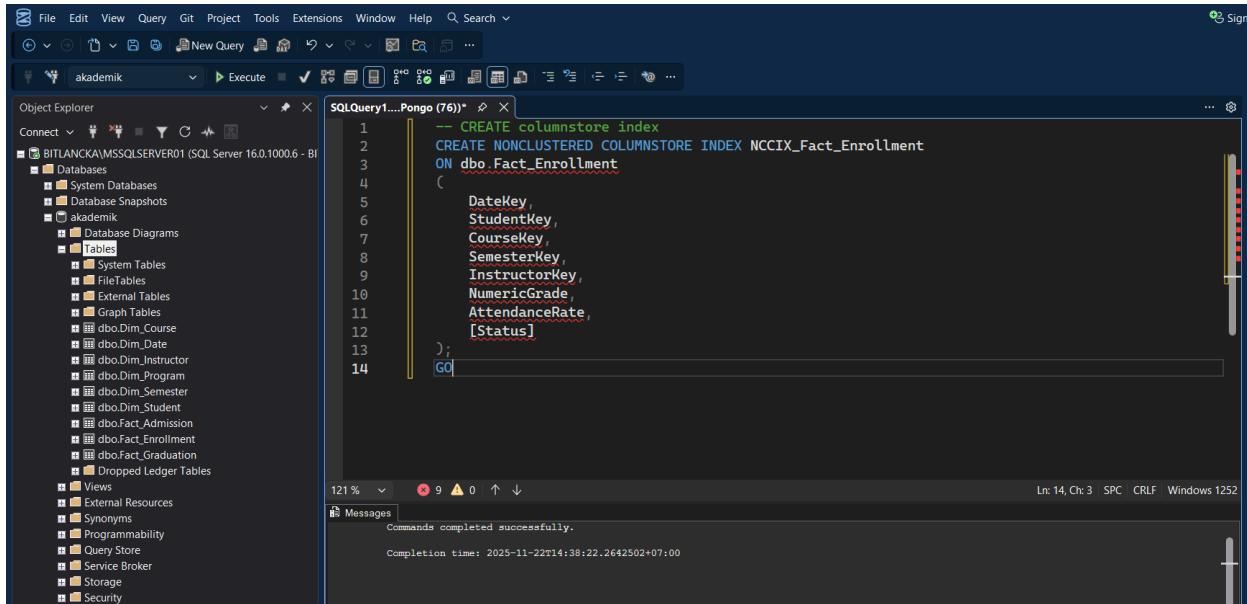
## Dokumentasi:



The screenshot shows the SQL Server Management Studio (SSMS) interface. The title bar displays "05\_Create\_Partitions.sql". The Object Explorer on the left shows the database "akademik" with its tables, including "dbo.Fact\_Enrollment". The main window contains a query editor titled "SQLQuery1...Pongo (76)\*" with the following T-SQL code:

```
-- CREATE columnstore index (FIXED)
CREATE NONCLUSTERED COLUMNSTORE INDEX NCCIX_Fact_Enrollment
ON dbo.Fact_Enrollment
(
    DateKey,
    StudentKey,
    CourseKey,
    SemesterKey,
    InstructorKey,
    NumericGrade,
    AttendanceRate,
    [Status]
);
GO
```

The status bar at the bottom right indicates "Ln: 5, Ch: 13 SPC CRLF Windows 1252".

This screenshot shows the same SSMS interface and query editor. The title bar still displays "05\_Create\_Partitions.sql". The main window contains the same T-SQL code as above, but the status bar at the bottom right now indicates "Ln: 14, Ch: 3 SPC CRLF Windows 1252".

The screenshot shows the SQL Server Management Studio (SSMS) interface. In the Object Explorer, a database named 'akademik' is selected. In the center pane, a query window titled 'SQLQuery1...Pongo (76)\*' contains the following T-SQL code:

```
1 CREATE PARTITION SCHEME PS_AcademicYear
2 AS PARTITION PF_AcademicYear
3 ALL TO ([PRIMARY]);
4 GO
```

The status bar at the bottom right indicates 'Ln: 4, Ch: 3 TABS CRLF Windows 1252'. Below the query window, the 'Messages' pane displays the output:

```
Partition scheme 'PS_AcademicYear' has been created successfully. 'PRIMARY' is marked as the next used filegroup in partition scheme 'PS_AcademicYear'.
Completion time: 2025-11-22T14:32:14.8822605+07:00
```

This screenshot shows the continuation of the SSMS session. The Object Explorer still shows the 'akademik' database. The central query window now contains the following T-SQL code:

```
-- Partition function by DateKey (threshold: 1 Agustus setiap tahun ajaran)
CREATE PARTITION FUNCTION PF_AcademicYear (INT)
AS RANGE RIGHT FOR VALUES
(
    20200801, -- 2020/2021
    20210801, -- 2021/2022
    20220801, -- 2022/2023
    20230801, -- 2023/2024
    20240801, -- 2024/2025
    20250801 -- 2025/2026
);
GO
```

The status bar at the bottom right indicates 'Ln: 13, Ch: 1 SPC CRLF Windows 1252'. The 'Messages' pane shows the command completed successfully.

## Step 4: ETL Design

Kode kemudian mendefinisikan *staging area* serta *stored procedure* untuk proses ETL, termasuk proses MERGE untuk sinkronisasi data dimensi dan proses insert terkontrol ke tabel fakta.

### 1. ETL Architecture Design

Memakai OPSI 2: T-SQL Stored Procedure karena lebih mudah dikerjakan.

06\_Create\_Staging.sql

## 2. Create Staging Tables

Query:

```
-----  
-- Staging Area  
-----  
CREATE SCHEMA stg;  
GO  
  
-- Staging Program  
CREATE TABLE stg.Program (  
    ProgramCode VARCHAR(20),  
    ProgramName VARCHAR(100),  
    Faculty    VARCHAR(100),  
    LoadDate   DATETIME DEFAULT GETDATE()  
);  
GO  
  
-- Staging Student  
CREATE TABLE stg.Student (  
    StudentNaturalID VARCHAR(20),  
    FullName      VARCHAR(100),  
    DOB          DATE,  
    Gender        CHAR(1),  
    EntryYear     INT,  
    [Status]      VARCHAR(20),  
    ProgramNaturalID VARCHAR(20),  
    LoadDate     DATETIME DEFAULT GETDATE()  
);  
GO  
  
-- Staging Course  
CREATE TABLE stg.Course (  
    CourseCode    VARCHAR(20),  
    CourseName    VARCHAR(150),  
    Credits       INT,  
    ProgramNaturalID VARCHAR(20),  
    LoadDate     DATETIME DEFAULT GETDATE()  
);  
GO  
  
-- Staging Instructor  
CREATE TABLE stg.Instructor (
```

```

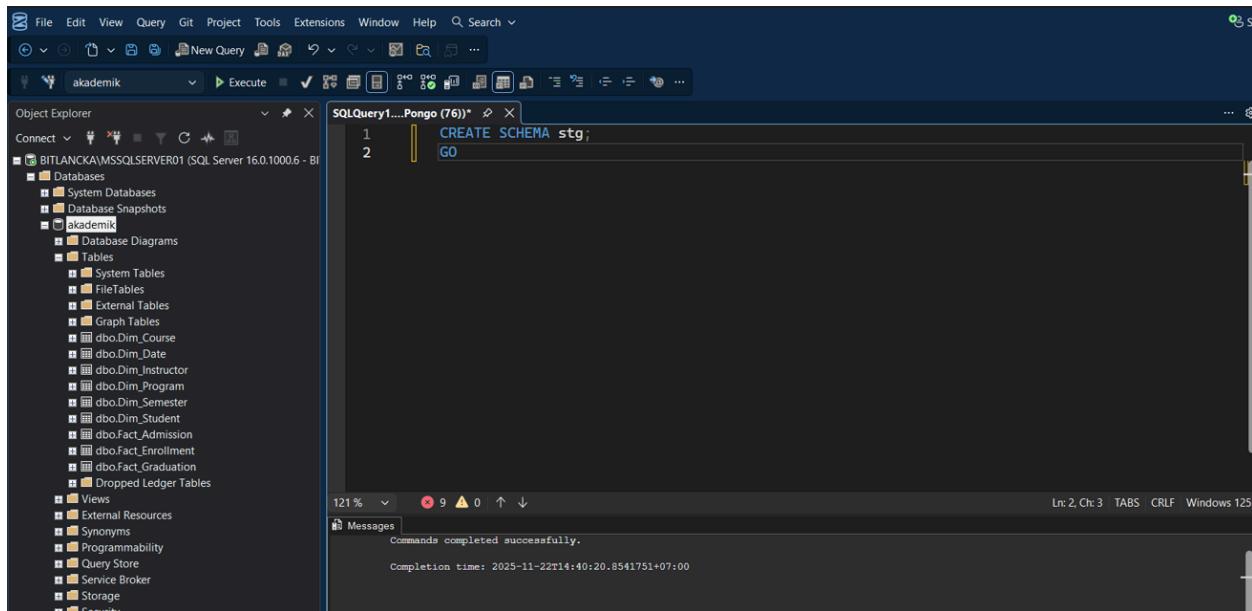
InstructorNaturalID VARCHAR(20),
[Name]      VARCHAR(100),
[Rank]       VARCHAR(50),
Dept        VARCHAR(100),
FTE         DECIMAL(3,2),
LoadDate    DATETIME DEFAULT GETDATE()
);
GO

-- Staging Enrollment (Fact)
CREATE TABLE stg.Enrollment (
    StudentNaturalID VARCHAR(20),
    CourseCode     VARCHAR(20),
    InstructorID   VARCHAR(20),
    SemesterCode   VARCHAR(20),
    DateKey        INT,
    Grade          VARCHAR(2),
    NumericGrade   DECIMAL(4,2),
    AttendanceRate DECIMAL(5,2),
    [Status]        VARCHAR(20),
    LoadDate       DATETIME DEFAULT GETDATE()
);
GO

```

## Dokumentasi:

### Create Schema Staging



The screenshot shows the SSMS interface with the following details:

- Object Explorer:** Shows the database structure under "akademik".
- SQL Query Editor:** Contains the following T-SQL code:
 

```
CREATE SCHEMA stg;
GO
```
- Messages Window:** Displays the output:
 

```
Commands completed successfully.
Completion time: 2025-11-22T14:40:20.8541751+07:00
```

Create Table Staging Student

The screenshot shows the SQL Server Management Studio (SSMS) interface. The Object Explorer on the left shows a connection to 'BITLANCKA\MSSQLSERVER01 (SQL Server 16.0.1000.6 - BI)'. Under the 'Tables' node for the 'akademik' database, there is a new table named 'stg.Student'. The SQL Query window on the right contains the following code:

```
-- Staging Student
CREATE TABLE stg.Student (
    StudentNaturalID VARCHAR(20),
    FullName         VARCHAR(100),
    DOB              DATE,
    Gender           CHAR(1),
    EntryYear        INT,
    [Status]         VARCHAR(20),
    ProgramNaturalID VARCHAR(20),
    LoadDate         DATETIME DEFAULT GETDATE()
);
```

The 'Messages' pane at the bottom shows the output: 'Commands completed successfully.' and 'Completion time: 2025-11-22T14:40:45.6726239+07:00'.

## Create Table Staging Program

The screenshot shows the SQL Server Management Studio (SSMS) interface. The Object Explorer on the left shows a connection to 'BITLANCKA\MSSQLSERVER01 (SQL Server 16.0.1000.6 - BI)'. Under the 'Tables' node for the 'akademik' database, there is a new table named 'stg.Program'. The SQL Query window on the right contains the following code:

```
-- Staging Program
CREATE TABLE stg.Program (
    ProgramCode   VARCHAR(20),
    ProgramName  VARCHAR(100),
    Faculty      VARCHAR(100),
    LoadDate     DATETIME DEFAULT GETDATE()
);
GO
```

The 'Messages' pane at the bottom shows the output: 'Commands completed successfully.' and 'Completion time: 2025-11-22T14:41:01.4358043+07:00'.

## Create Table Staging Course

The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'akademik' database and its tables. The central pane displays a query window titled 'SQLQuery1....Pongo (76)\*'. The query creates a table named 'stg.Course' with columns: CourseCode (VARCHAR(20)), CourseName (VARCHAR(150)), Credits (INT), ProgramNaturalID (VARCHAR(20)), and LoadDate (DATETIME with a default value of GETDATE()). A yellow box highlights the 'GO' command at the end of the script. The status bar at the bottom indicates the command completed successfully with a completion time of 2025-11-22T14:41:19.7620584+07:00.

```
-- Staging Course
CREATE TABLE stg.Course (
    CourseCode      VARCHAR(20),
    CourseName     VARCHAR(150),
    Credits        INT,
    ProgramNaturalID VARCHAR(20),
    LoadDate       DATETIME DEFAULT GETDATE()
);
GO
```

## Create Table Staging Instructor

The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'akademik' database and its tables. The central pane displays a query window titled 'SQLQuery1.sq..\\Pongo (76)\*'. The query creates a table named 'stg.Instructor' with columns: InstructorNaturalID (VARCHAR(20)), [Name] (VARCHAR(100)), [Rank] (VARCHAR(50)), Dept (VARCHAR(100)), FTE (DECIMAL(3,2)), and LoadDate (DATETIME with a default value of GETDATE()). A yellow box highlights the 'GO' command at the end of the script. The status bar at the bottom indicates the command completed successfully with a completion time of 2025-11-22T14:41:33.2642476+07:00.

```
-- Staging Instructor
CREATE TABLE stg.Instructor (
    InstructorNaturalID VARCHAR(20),
    [Name]           VARCHAR(100),
    [Rank]            VARCHAR(50),
    Dept              VARCHAR(100),
    FTE                DECIMAL(3,2),
    LoadDate          DATETIME DEFAULT GETDATE()
);
GO
```

## Create Table Staging Enrollment

The screenshot shows the SSMS interface with the Object Explorer on the left and a query editor window on the right. The query editor contains the following SQL code:

```
-- Staging Enrollment
CREATE TABLE stg.Enrollment (
    StudentNaturalID VARCHAR(20),
    CourseCode      VARCHAR(20),
    InstructorID   VARCHAR(20),
    SemesterCode    VARCHAR(20),
    DateKey         INT,
    Grade           VARCHAR(2),
    NumericGrade    DECIMAL(4,2),
    AttendanceRate DECIMAL(5,2),
    [Status]        VARCHAR(20),
    LoadDate        DATETIME DEFAULT GETDATE()
);
GO
```

The Messages pane at the bottom indicates that the command completed successfully.

### 3. ETL Mapping Document

+ [ETL\\_Mapping](#)

## Step 5: ETL Implementation (SSIS)

1. Create SSIS Project
2. Package 1: Load Dimensions

### Load Dim Program

Query:

```
-----  
-- ETL Logic  
-----  
  
-- Load Dim_Program  
CREATE OR ALTER PROCEDURE dbo.usp_Load_Dim_Program  
AS  
BEGIN
```

```

MERGE dbo.Dim_Program AS tgt
USING (
    SELECT DISTINCT ProgramCode, ProgramName, Faculty
    FROM stg.Program
) AS src
ON tgt.ProgramCode = src.ProgramCode
WHEN MATCHED THEN
    UPDATE SET
        ProgramName = src.ProgramName,
        Faculty      = src.Faculty
WHEN NOT MATCHED BY TARGET THEN
    INSERT (ProgramCode, ProgramName, Faculty)
    VALUES (src.ProgramCode, src.ProgramName, src.Faculty);
END;
GO

```

Dokumentasi:

```

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

    MERGE dbo.Dim_Program AS tgt
    USING (
        SELECT DISTINCT ProgramCode, ProgramName, Faculty
        FROM stg.Program
    ) AS src
    ON tgt.ProgramCode = src.ProgramCode
    WHEN MATCHED THEN
        UPDATE SET
            tgt.ProgramName = src.ProgramName,
            tgt.Faculty      = src.Faculty
    WHEN NOT MATCHED BY TARGET THEN
        INSERT (ProgramCode, ProgramName, Faculty)
        VALUES (src.ProgramCode, src.ProgramName, src.Faculty);
END;
GO

```

## Load Dim Student

Query:

```
-- Load Dim_Student
CREATE OR ALTER PROCEDURE dbo.usp_Load_Dim_Student
AS
BEGIN
    MERGE dbo.Dim_Student AS tgt
    USING (
        SELECT DISTINCT
            StudentNaturalID, FullName, DOB, Gender,
            EntryYear, [Status], ProgramNaturalID
        FROM stg.Student
    ) AS src
    ON tgt.StudentNaturalID = src.StudentNaturalID
    WHEN MATCHED THEN
        UPDATE SET
            FullName      = src.FullName,
            DOB          = src.DOB,
            Gender        = src.Gender,
            EntryYear     = src.EntryYear,
            [Status]      = src.[Status],
            ProgramNaturalID = src.ProgramNaturalID
    WHEN NOT MATCHED BY TARGET THEN
        INSERT (StudentNaturalID, FullName, DOB, Gender,
                EntryYear, [Status], ProgramNaturalID)
        VALUES (src.StudentNaturalID, src.FullName, src.DOB,
                src.Gender, src.EntryYear, src.[Status],
                src.ProgramNaturalID);
END;
GO
```

Dokumantasi:

**Load Dim\_Student (versi simple, tanpa SCD2)**

```

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

    MERGE dbo.Dim_Student AS tgt
    USING (
        SELECT DISTINCT
            StudentNaturalID,
            FullName,
            DOB,
            Gender,
            EntryYear,
            [Status],
            ProgramNaturalID
        FROM stg.Student
    ) AS src
    ON tgt.StudentNaturalID = src.StudentNaturalID
    WHEN MATCHED AND (
        tgt.FullName      => src.FullName
        OR tgt.DOB       => src.DOB
        OR tgt.Gender    => src.Gender
        OR tgt.EntryYear => src.EntryYear
        OR tgt.[Status]  => src.[Status]
        OR tgt.ProgramNaturalID => src.ProgramNaturalID
    ) THEN
        UPDATE SET
            tgt.FullName      = src.FullName,
            tgt.DOB          = src.DOB,
            tgt.Gender       = src.Gender,
            tgt.EntryYear    = src.EntryYear,
            tgt.[Status]     = src.[Status],
            tgt.ProgramNaturalID = src.ProgramNaturalID
    WHEN NOT MATCHED BY TARGET THEN
        INSERT (StudentNaturalID, FullName, DOB, Gender, EntryYear, [Status], ProgramNaturalID)
        VALUES (src.StudentNaturalID, src.FullName, src.DOB, src.Gender,
                src.EntryYear, src.[Status], src.ProgramNaturalID);
    END;
GO

```

56 % 49 0 ↑ ↓

Messages

Command completed successfully.

### 3. Package 2: Load Facts

#### Load Fact Enrollment

Query:

```
-- Load Fact_Enrollment
CREATE OR ALTER PROCEDURE dbo.usp_Load_Fact_Enrollment
AS
BEGIN
    INSERT INTO dbo.Fact_Enrollment (
        StudentKey, CourseKey, SemesterKey, InstructorKey,
        DateKey, Grade, NumericGrade, AttendanceRate, [Status]
    )
    SELECT
        s.StudentKey,
        c.CourseKey,
        sem.SemesterKey,
        i.InstructorKey,
        e.DateKey,
        e.Grade,
```

```

e.NumericGrade,
e.AttendanceRate,
e.[Status]
FROM stg.Enrollment e
JOIN dbo.Dim_Student s ON e.StudentNaturalID = s.StudentNaturalID
JOIN dbo.Dim_Course c ON e.CourseCode = c.CourseCode
JOIN dbo.Dim_Instructor i ON e.InstructorID = i.InstructorNaturalID
JOIN dbo.Dim_Semester sem ON e.SemesterCode = sem.SemesterCode
WHERE NOT EXISTS (
    SELECT 1
    FROM dbo.Fact_Enrollment fe
    WHERE fe.StudentKey = s.StudentKey
    AND fe.CourseKey = c.CourseKey
    AND fe.DateKey = e.DateKey
);
END;
GO

```

Dokumentasi:

```

CREATE OR ALTER PROCEDURE dbo.usp_Load_Fact_Enrollment
AS
BEGIN
    SET NOCOUNT ON;
    INSERT INTO dbo.Fact_Enrollment (
        StudentKey,
        CourseKey,
        SemesterKey,
        InstructorKey,
        DateKey,
        Grade,
        NumericGrade,
        AttendanceRate,
        [Status]
    )
    SELECT
        e.StudentKey,
        c.CourseKey,
        sem.SemesterKey,
        i.InstructorKey,
        e.DateKey,
        e.Grade,
        e.NumericGrade,
        e.AttendanceRate,
        e.[Status]
    FROM stg.Enrollment e
    INNER JOIN dbo.Dim_Student s ON e.StudentNaturalID = s.StudentNaturalID
    INNER JOIN dbo.Dim_Course c ON e.CourseCode = c.CourseCode
    INNER JOIN dbo.Dim_Instructor i ON e.InstructorID = i.InstructorNaturalID
    INNER JOIN dbo.Dim_Semester sem ON e.SemesterCode = sem.SemesterCode
    WHERE NOT EXISTS (
        SELECT 1
        FROM dbo.Fact_Enrollment f
        WHERE f.StudentKey = e.StudentKey
        AND f.CourseKey = e.CourseKey
        AND f.InstructorKey = e.InstructorID
        AND f.SemesterKey = e.SemesterCode
        AND f.Datekey = e.Datekey
    );
END;
GO

```

Object Explorer

- BITLANKA\SQLSERVER01 (SQL Server 16.0.1000.1)
- Databases
  - System Databases
  - Database Snapshots
  - akademik
- Tables
  - File Tables
  - External Tables
  - Graph Tables
  - dbo.Dim\_Course
  - dbo.Dim\_Date
  - dbo.Dim\_Instructor
  - dbo.Dim\_Program
  - dbo.Dim\_Semester
  - dbo.Dim\_Student
  - dbo.Fact\_Admission
  - dbo.Fact\_Enrollment
  - dbo.Fact\_Graduation
  - stg.Course
  - stg.Enrollment
  - stg.Instructor
  - stg.Program
  - stg.Student
  - Dropped Ledger Tables
- Views
- External Resources
- Synonyms
- Programmability
- Query Store
- Service Broker
- Storage
- Security

Messages

Command completed successfully.  
Completion time: 2020-11-20 10:18:10.0000000 +07:00

## 4. Master Package

The screenshot shows the SSMS interface with the following details:

- Title Bar:** Master ETL
- Toolbar:** File, Edit, View, Git, Project, Tools, Extensions, Window, Help, Search
- Object Explorer:** Connected to BITLANCKA\MSSQLSERVER01 (SQL Server 16.0.1000.6). The tree view shows the database structure under the 'akademik' database, including Tables, Views, and other objects.
- Query Editor:** The query window contains the following T-SQL code for creating a stored procedure:

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

    BEGIN TRY
        BEGIN TRAN;
        -- Load dimensions
        EXEC dbo.usp_Load_Dim_Program;
        EXEC dbo.usp_Load_Dim_Student;
        -- Tambah: usp_Load_Dim_Course, usp_Load_Dim_Instructor, usp_Load_Dim_Semester, Dim_Date
        -- Load facts
        EXEC dbo.usp_Load_Fact_Enrollment;
        -- Tambah lagi nanti: usp_Load_Fact_Graduation, usp_Load_Fact_Admission

        -- Update statistics sederhana
        UPDATE STATISTICS dbo.Dim_Student;
        UPDATE STATISTICS dbo.Fact_Enrollment;

        COMMIT TRAN;
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0 ROLLBACK TRAN;
        DECLARE @Err NVARCHAR(4000) = ERROR_MESSAGE();
        RAISERROR(@Err, 16, 1);
    END CATCH;
END;
GO
```

- Messages Window:** Shows the command completed successfully and the compilation time.
- Status Bar:** Query executed successfully.

## Step 6: Data Quality Assurance

### 1. Data Quality Checks

Bagian akhir kode menyediakan *data quality checks* seperti pengecekan completeness, orphan records, invalid values, dan duplikasi, serta menambahkan contoh query analitik untuk menguji performa dan melihat insight seperti trend enrollment, distribusi GPA, funnel admission, hingga hubungan antara absensi dan nilai.

## 5. Data Quality Checks (Step 6)

08\_Data\_Quality\_Checks.sql

Query:

```
-----  
-- Data Quality Checks  
-----  
-- 1. Completeness check  
SELECT COUNT(*) AS NullName  
FROM dbo.Dim_Student  
WHERE FullName IS NULL;  
GO  
  
-- 2. Orphan check Fact <-> Dimension  
SELECT COUNT(*) AS Orphan_Students  
FROM dbo.Fact_Enrollment f  
LEFT JOIN dbo.Dim_Student d ON f.StudentKey = d.StudentKey  
WHERE d.StudentKey IS NULL;  
GO  
  
-- 3. Numeric grade validity  
SELECT COUNT(*) AS InvalidGrades  
FROM dbo.Fact_Enrollment  
WHERE NumericGrade NOT BETWEEN 0 AND 4;  
GO  
  
-- 4. Duplicate fact combinations  
SELECT StudentKey, CourseKey, SemesterKey, COUNT(*) AS Cnt  
FROM dbo.Fact_Enrollment  
GROUP BY StudentKey, CourseKey, SemesterKey  
HAVING COUNT(*) > 1;  
GO
```

The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'akademik'. The central pane displays a T-SQL script named 'SQLQuery1.sql' with the following code:

```
-- 1. Completeness: cek NULL di Dim_Student
SELECT
    Dim_Student AS TableName,
    COUNT(*) AS TotalRows,
    SUM(CASE WHEN StudentNaturalID IS NULL THEN 1 ELSE 0 END) AS NullNIM,
    SUM(CASE WHEN FullName IS NULL THEN 1 ELSE 0 END) AS NullName,
    SUM(CASE WHEN Gender IS NULL THEN 1 ELSE 0 END) AS NullGender
FROM dbo.Dim_Student;
GO
```

The results pane shows a single row of data from the query:

TableName	TotalRows	NullNIM	NullName	NullGender
Dim_Student	0	NULL	NULL	NULL

At the bottom, a message indicates the query was executed successfully.

Output menunjukkan bahwa pada tabel Dim\_Student tidak ada nilai NULL.

The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'akademik'. The central pane displays a T-SQL script named 'SQLQuery1...Pongo (76)\*' with the following code:

```
-- 2. Consistency: Orphan fact vs dim
SELECT
    COUNT(*) AS OrphanStudentFact
FROM dbo.Fact_Enrollment f
LEFT JOIN dbo.Dim_Student s ON f.StudentKey = s.StudentKey
WHERE s.StudentKey IS NULL;
GO
```

The results pane shows a single row of data from the query:

OrphanStudentFact
0

At the bottom, a message indicates the query was executed successfully.

Hasil pengecekan konsistensi data menunjukkan bahwa tidak terdapat *orphan record* pada tabel *Fact\_Enrollment*. Seluruh nilai *StudentKey* pada tabel fakta berhasil mencocokkan data yang ada pada tabel *Dim\_Student*, sehingga integritas referensial antara fakta dan dimensi berada dalam kondisi baik.

The screenshot shows the SQL Server Management Studio (SSMS) interface. On the left is the Object Explorer pane, which displays the database structure for 'BITLANCKA\MSSQLSERVER01' (SQL Server 16.0.1000.6). It includes nodes for Databases, Tables (containing System Tables, FileTables, External Tables, Graph Tables, and various fact and dimension tables like Dim\_Course, Dim\_Instructor, Dim\_Program, Dim\_Semester, Dim\_Student, Fact\_Admission, Fact\_Enrollment, Fact\_Graduation, and Student-related tables), Views, Synonyms, Programmability, Query Store, Service Broker, Storage, Security, and Server Objects. A 'Dropped Ledger Tables' node is also present. The main area shows a query window titled 'SQLQuery1....Pongo (76)\*'. The query is:

```
-- 3. Accuracy: numeric grade di luar range
SELECT COUNT(*) AS InvalidNumericGrade
FROM dbo.Fact_Enrollment
WHERE NumericGrade < 0 OR NumericGrade > 4;
GO
```

The results pane shows a single row with the value '0' under the column 'InvalidNumericGrade'. Below the results pane, a message bar indicates 'Query executed successfully.' and shows the connection details: BITLANCKA\MSSQLSERVER01 (16... | BITLANCKA\Pongo (76).

Tidak ditemukan nilai NumericGrade yang berada di luar rentang 0 sampai 4. Semua nilai NumericGrade pada tabel Fact\_Enrollment valid dan sesuai standar penilaian

The screenshot shows the SSMS interface with the following details:

- Object Explorer:** Shows the database structure for 'BITLANCKA\MSSQLSERVER01'. Under the 'akademik' database, it lists 'Tables' (including 'dbo.Fact\_Enrollment'), 'Views', and other system objects.
- SQLQuery1....Pongo (76)\*:** This is the active query window. It contains a T-SQL script labeled '4. Duplicate business key' which counts rows where multiple combinations of StudentKey, CourseKey, and SemesterKey occur. The script is as follows:

```
-- 4. Duplicate business key
SELECT
    StudentKey, CourseKey, SemesterKey,
    COUNT(*) AS DuplicateCount
FROM dbo.Fact_Enrollment
GROUP BY StudentKey, CourseKey, SemesterKey
HAVING COUNT(*) > 1;
GO
```

- Results Window:** Below the query window, there is a table header row with columns: StudentKey, CourseKey, SemesterKey, and DuplicateCount. The main body of the table is currently empty, indicating no duplicates were found.
- Status Bar:** At the bottom, it says 'Query executed successfully.' and shows the connection information: BITLANCKA\MSSQLSERVER01 (16...).

Pengecekan duplikasi pada tabel Fact\_Enrollment menunjukkan bahwa tidak terdapat kombinasi StudentKey, CourseKey, dan SemesterKey yang muncul lebih dari satu kali. Hal ini menandakan bahwa data transaksi enrollment tidak memiliki duplikasi dan sudah memenuhi aturan grain yang ditetapkan.

```

-- 5. Rekonsiliasi jumlah record antara staging vs warehouse
SELECT 'stg.Enrollment' AS Source, COUNT(*) AS RecordCount
FROM stg.Enrollment
UNION ALL
SELECT 'Fact_Enrollment' AS Source, COUNT(*) AS RecordCount
FROM dbo.Fact_Enrollment;
GO

```

The screenshot shows the SSMS interface with the following details:

- Object Explorer:** Shows the database structure for 'BITLANKA\MSQLSERVER01 (SQL Server 16.0.1000.6)' under the 'akademik' database.
- SQL Query Window:** Title: 'SQLQuery1....Pongo (76)\*'. It contains the provided T-SQL script.
- Results Grid:** Shows the output of the query. There are two rows: 'stg.Enrollment' with a RecordCount of 0, and 'Fact\_Enrollment' with a RecordCount of 0.
- Status Bar:** Shows 'Query executed successfully.' and other execution details.

Hasil rekonsiliasi antara tabel staging dan tabel warehouse menunjukkan bahwa kedua tabel, baik stg.Enrollment maupun Fact\_Enrollment, memiliki jumlah record 0. Hal ini menandakan bahwa belum terdapat data yang masuk ke staging, atau proses ETL untuk memuat data ke tabel Fact\_Enrollment belum dijalankan

## Step 7: Performance Testing

1. Create Test Queries
2. Analyze Query Plans
3. Performance Benchmarks

Query:

```

-----
-- Performance Testing
-----

-- 1. Average grade per program per year
SELECT
    p.ProgramName,
    d.[Year],
    AVG(f.NumericGrade) AS AvgGrade
FROM dbo.Fact_Enrollment f
JOIN dbo.Dim_Date d ON f.DateKey = d.DateKey

```

```

JOIN dbo.Dim_Course c ON f.CourseKey = c.CourseKey
JOIN dbo.Dim_Program p ON c.ProgramNaturalID = p.ProgramCode
GROUP BY p.ProgramName, d.[Year]
ORDER BY d.[Year], p.ProgramName;

-- 2. Monthly enrollment trend
SELECT
    d.[Year], d.[Month],
    COUNT(*) AS TotalEnrollments
FROM dbo.Fact_Enrollment f
JOIN dbo.Dim_Date d ON f.DateKey = d.DateKey
GROUP BY d.[Year], d.[Month]
ORDER BY d.[Year], d.[Month];
GO

-- 3. – Funnel Admission → Enrollment → Graduation per Program per Tahun
;WITH Adm AS (
    SELECT
        a.ProgramKey,
        d.[Year] AS AdmissionYear,
        COUNT(DISTINCT a.StudentKey) AS TotalApplicants
    FROM dbo.Fact_Admission a
    JOIN dbo.Dim_Date d ON a.AdmissionDateKey = d.DateKey
    GROUP BY a.ProgramKey, d.[Year]
),
Enr AS (
    SELECT
        c.ProgramNaturalID,
        d.[Year] AS EnrollmentYear,
        COUNT(DISTINCT f.StudentKey) AS TotalEnrolled
    FROM dbo.Fact_Enrollment f
    JOIN dbo.Dim_Course c ON f.CourseKey = c.CourseKey
    JOIN dbo.Dim_Date d ON f.DateKey = d.DateKey
    GROUP BY c.ProgramNaturalID, d.[Year]
),
Grad AS (
    SELECT
        g.ProgramKey,
        d.[Year] AS GraduationYear,
        COUNT(DISTINCT g.StudentKey) AS TotalGraduated
    FROM dbo.Fact_Graduation g
    JOIN dbo.Dim_Date d ON g.GraduationDateKey = d.DateKey
    GROUP BY g.ProgramKey, d.[Year]
)

```

```

)
SELECT
    p.ProgramName,
    a.AdmissionYear,
    a.TotalApplicants,
    ISNULL(e.TotalEnrolled, 0) AS TotalEnrolled,
    ISNULL(g.TotalGraduated, 0) AS TotalGraduated,
    CASE WHEN a.TotalApplicants = 0
        THEN 0 ELSE 1.0 * ISNULL(e.TotalEnrolled, 0) / a.TotalApplicants END AS
    Conv_Adm_to_Enroll,
    CASE WHEN a.TotalApplicants = 0
        THEN 0 ELSE 1.0 * ISNULL(g.TotalGraduated, 0) / a.TotalApplicants END AS
    Conv_Adm_to_Grad
FROM Adm a
JOIN dbo.Dim_Program p ON a.ProgramKey = p.ProgramKey
LEFT JOIN Enr e
    ON p.ProgramCode = e.ProgramNaturalID
    AND a.AdmissionYear = e.EnrollmentYear
LEFT JOIN Grad g
    ON a.ProgramKey = g.ProgramKey
    AND a.AdmissionYear = g.GraduationYear
ORDER BY p.ProgramName, a.AdmissionYear;
GO

```

-- 4: Distribusi lama studi dan GPA per program

```

SELECT
    p.ProgramName,
    COUNT(*) AS TotalGraduates,
    AVG(g.GPA) AS AvgGPA,
    MIN(g.GPA) AS MinGPA,
    MAX(g.GPA) AS MaxGPA,
    AVG(CAST(g.StudyDuration AS FLOAT)) AS AvgStudyMonths,
    MIN(g.StudyDuration) AS MinStudyMonths,
    MAX(g.StudyDuration) AS MaxStudyMonths
FROM dbo.Fact_Graduation g
JOIN dbo.Dim_Program p ON g.ProgramKey = p.ProgramKey
GROUP BY p.ProgramName
ORDER BY p.ProgramName;
GO

```

-- 5: Top 10 mahasiswa per program (GPA + SKS)

```

;WITH GradDetail AS (
    SELECT

```

```

g.StudentKey,
s.StudentNaturalID,
s.FullName,
p.ProgramName,
g.GPA,
g.TotalCredits,
ROW_NUMBER() OVER (
    PARTITION BY p.ProgramName
    ORDER BY g.GPA DESC, g.TotalCredits DESC
) AS rn
FROM dbo.Fact_Graduation g
JOIN dbo.Dim_Student s ON g.StudentKey = s.StudentKey
JOIN dbo.Dim_Program p ON g.ProgramKey = p.ProgramKey
)
SELECT
    ProgramName,
    rn AS RankInProgram,
    StudentNaturalID,
    FullName,
    GPA,
    TotalCredits
FROM GradDetail
WHERE rn <= 10
ORDER BY ProgramName, rn;
GO

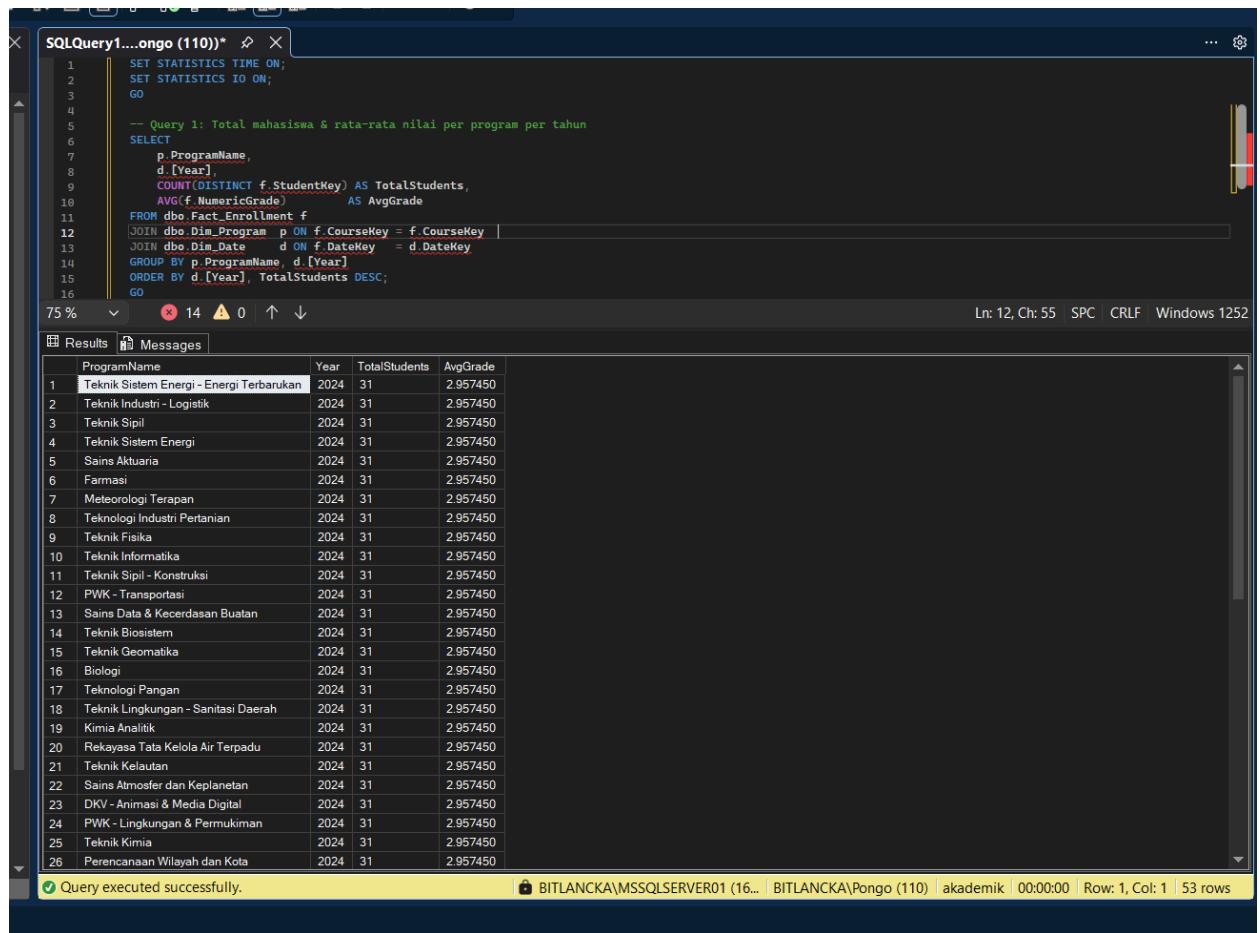
-- 6: Pengaruh Attendance terhadap NumericGrade
;WITH EnrBucket AS (
    SELECT
        CASE
            WHEN AttendanceRate < 60 THEN '< 60%'
            WHEN AttendanceRate < 75 THEN '60–74%'
            WHEN AttendanceRate < 90 THEN '75–89%'
            ELSE '>= 90%'
        END AS AttendanceBucket,
        NumericGrade
    FROM dbo.Fact_Enrollment
    WHERE NumericGrade IS NOT NULL
)
SELECT
    AttendanceBucket,
    COUNT(*) AS TotalEnrollments,
    AVG(NumericGrade) AS AvgGrade,

```

```
MIN(NumericGrade)      AS MinGrade,  
MAX(NumericGrade)      AS MaxGrade  
FROM EnrBucket  
GROUP BY AttendanceBucket  
ORDER BY  
CASE AttendanceBucket  
WHEN '< 60%' THEN 1  
WHEN '60–74%' THEN 2  
WHEN '75–89%' THEN 3  
WHEN '>= 90%' THEN 4  
END;  
GO
```

```
-- 7: Trend enrollment per semester dan program  
SELECT  
    sem.SemesterCode,  
    d.[Year],  
    p.ProgramName,  
    COUNT(*)          AS TotalEnrollments,  
    COUNT(DISTINCT f.StudentKey) AS DistinctStudents,  
    AVG(f.NumericGrade) AS AvgGrade  
FROM dbo.Fact_Enrollment f  
JOIN dbo.Dim_Semester sem ON f.SemesterKey = sem.SemesterKey  
JOIN dbo.Dim_Date d   ON f.DateKey    = d.DateKey  
JOIN dbo.Dim_Course c  ON f.CourseKey  = c.CourseKey  
JOIN dbo.Dim_Program p  ON c.ProgramNaturalID = p.ProgramCode  
GROUP BY sem.SemesterCode, d.[Year], p.ProgramName  
ORDER BY d.[Year], sem.SemesterCode, p.ProgramName;  
GO
```

## Output:



The screenshot shows a SQL Server Management Studio (SSMS) window with a query editor and a results grid. The query editor contains T-SQL code to calculate the total number of students and average grade per program per year. The results grid displays 53 rows of data, each representing a program with its name, year, total students, and average grade.

ProgramName	Year	TotalStudents	AvgGrade
Teknik Sistem Energi - Energi Terbarukan	2024	31	2.957450
Teknik Industri - Logistik	2024	31	2.957450
Teknik Spil	2024	31	2.957450
Teknik Sistem Energi	2024	31	2.957450
Sains Aktuaria	2024	31	2.957450
Farmasi	2024	31	2.957450
Meteorologi Terapan	2024	31	2.957450
Teknologi Industri Pertanian	2024	31	2.957450
Teknik Fisika	2024	31	2.957450
Teknik Informatika	2024	31	2.957450
Teknik Spil - Konstruksi	2024	31	2.957450
PWK - Transportasi	2024	31	2.957450
Sains Data & Kecerdasan Buatan	2024	31	2.957450
Teknik Biosistem	2024	31	2.957450
Teknik Geomatika	2024	31	2.957450
Biologi	2024	31	2.957450
Teknologi Pangan	2024	31	2.957450
Teknik Lingkungan - Sanitasi Daerah	2024	31	2.957450
Kimia Analitik	2024	31	2.957450
Rekayasa Tata Kelola Air Terpadu	2024	31	2.957450
Teknik Kelautan	2024	31	2.957450
Sains Atmosfer dan Keplanetan	2024	31	2.957450
DKV - Animasi & Media Digital	2024	31	2.957450
PWK - Lingkungan & Permukiman	2024	31	2.957450
Teknik Kimia	2024	31	2.957450
Perencanaan Wilayah dan Kota	2024	31	2.957450

	ProgramName	Year	TotalStudents	AvgGrade	
29	Teknik Geofisika	2024	31	2.957450	
30	Teknik Perkeretaapian	2024	31	2.957450	
31	Sains Data	2024	31	2.957450	
32	Informatika - Cyber Security	2024	31	2.957450	
33	Arsitektur Urban	2024	31	2.957450	
34	Tahap Persiapan Bersama	2024	31	2.957450	
35	Teknik Mesin	2024	31	2.957450	
36	Teknologi Pangan - Keamanan Pangan	2024	31	2.957450	
37	Teknik Industri	2024	31	2.957450	
38	Sains Lingkungan Kelautan	2024	31	2.957450	
39	Arsitektur	2024	31	2.957450	
40	Fisika	2024	31	2.957450	
41	DKV - Branding & Identitas Visual	2024	31	2.957450	
42	Informatika - Software Development	2024	31	2.957450	
43	Fisika Instrumentasi	2024	31	2.957450	
44	Arsitektur Lanskap	2024	31	2.957450	
45	Teknik Lingkungan	2024	31	2.957450	
46	Matematika Terapan & Statistika	2024	31	2.957450	
47	Desain Komunikasi Visual	2024	31	2.957450	
48	Matematika	2024	31	2.957450	
49	Teknik Sipil - Transportasi	2024	31	2.957450	
50	Teknik Elektro	2024	31	2.957450	
51	Teknik Mesin - Energi	2024	31	2.957450	
52	Biologi Mikrobiologi	2024	31	2.957450	
53	Fisika Material	2024	31	2.957450	

✓ Query executed successfully.

|  BITLANCK

Results Messages

ProgramName	Year	TotalStudents	AvgGrade
14 Teknik Biosistem	2024	31	2.957450
15 Teknik Geomatika	2024	31	2.957450
16 Biologi	2024	31	2.957450
17 Teknologi Pangan	2024	31	2.957450
18 Teknik Lingkungan - Sanitasi Daerah	2024	31	2.957450
19 Kimia Analitik	2024	31	2.957450
20 Rekayasa Tata Kelola Air Terpadu	2024	31	2.957450
21 Teknik Kelautan	2024	31	2.957450
22 Sains Atmosfer dan Keplanetan	2024	31	2.957450
23 DKV - Animasi & Media Digital	2024	31	2.957450
24 PWK - Lingkungan & Permukiman	2024	31	2.957450
25 Teknik Kimia	2024	31	2.957450
26 Perencanaan Wilayah dan Kota	2024	31	2.957450
27 Kimia	2024	31	2.957450
28 Teknik Industri - Manufaktur	2024	31	2.957450
29 Teknik Geofisika	2024	31	2.957450
30 Teknik Perkeretaapian	2024	31	2.957450
31 Sains Data	2024	31	2.957450
32 Informatika - Cyber Security	2024	31	2.957450
33 Arsitektur Urban	2024	31	2.957450
34 Tahap Persiapan Bersama	2024	31	2.957450
35 Teknik Mesin	2024	31	2.957450
36 Teknologi Pangan - Keamanan Pangan	2024	31	2.957450
37 Teknik Industri	2024	31	2.957450
38 Sains Lingkungan Kelautan	2024	31	2.957450
39 Arsitektur	2024	31	2.957450

Query executed successfully.

BITLANCKA\BITLANCKA (16...) | BITLANCKA\Ponco (110) | akademik | 00:00:00 | Row: 1, Col: 1 | 53 rows

Query pertama digunakan untuk menghitung rata-rata nilai (NumericGrade) mahasiswa pada setiap program studi untuk setiap tahun akademik. Dengan menggabungkan tabel fact enrollment, dimensi tanggal, dan dimensi program, query ini membantu melihat *kualitas akademik rata-rata* tiap program dari tahun ke tahun. Hasilnya dapat digunakan untuk analisis performa program, tren peningkatan atau penurunan IP mahasiswa, serta menjadi indikator mutu pendidikan

The screenshot shows the SSMS interface with the following details:

- Object Explorer:** Shows the database structure under "BITLANCKA\MSSQLSERVER01 (SQL Server 16.0.1000.6)".
- Query Editor:** Titled "SQLQuery1...Pongo (67)\*", it contains the following T-SQL code:

```
-- Query 2: Trend enrollment bulanan
SELECT
    d.[Year],
    d.[Month],
    COUNT(*) AS TotalEnrollments
FROM dbo.Fact_Enrollment f
JOIN dbo.Dim_Date d ON f.DateKey = d.DateKey
GROUP BY d.[Year], d.[Month]
ORDER BY d.[Year], d.[Month];
GO
```
- Results Pane:** Displays the output of the query in a table format.

	Year	Month	TotalEnrollments
1	2024	6	20000

Query ini digunakan untuk melihat jumlah pendaftaran/enrollment per bulan pada setiap tahun. Data ini bermanfaat untuk menganalisis *pola musiman* pendaftaran mahasiswa, misalnya apakah ada lonjakan pada awal semester atau penurunan pada bulan tertentu. Dengan tren bulanan ini, institusi dapat melakukan perencanaan kapasitas kelas, evaluasi jadwal akademik, atau kebutuhan sumber daya.

```

-- Query 3: Funnel Admission > Enrollment > Graduation
WITH Adm AS (
    SELECT
        a.ProgramKey,
        d.[Year] AS AdmissionYear,
        COUNT(DISTINCT a.StudentKey) AS TotalApplicants
    FROM dbo.Fact_Admission a
    JOIN dbo.Dim_Date d ON a.AdmissionDateKey = d.DateKey
    GROUP BY a.ProgramKey, d.[Year]
),
Enr AS (
    SELECT
        c.ProgramNaturalID,
        d.[Year] AS EnrollmentYear,
        COUNT(DISTINCT f.StudentKey) AS TotalEnrolled
    FROM dbo.Fact_Enrollment f
    JOIN dbo.Dim_Course c ON f.CourseKey = c.CourseKey
    JOIN dbo.Dim_Date d ON f.DateKey = d.DateKey
    GROUP BY c.ProgramNaturalID, d.[Year]
),
Grad AS (
    SELECT
        g.ProgramName,
        g.AdmissionYear,
        COUNT(g.StudentKey) AS TotalGraduated
    FROM dbo.Fact_Graduation g
    WHERE g.GraduationDateKey = (SELECT MAX(DateKey) FROM dbo.Dim_Date)
    GROUP BY g.ProgramName, g.AdmissionYear
)
SELECT
    p.ProgramName,
    p.AdmissionYear,
    Adm.TotalApplicants,
    Enr.TotalEnrolled,
    Grad.TotalGraduated,
    Enr.TotalEnrolled / Adm.TotalApplicants AS Conv. Adm. to Enroll,
    Grad.TotalGraduated / Adm.TotalApplicants AS Conv. Adm. to Grad
FROM Adm
JOIN Enr ON Adm.ProgramKey = Enr.ProgramNaturalID
JOIN Grad ON Adm.ProgramKey = Grad.ProgramName
ORDER BY p.ProgramName, p.AdmissionYear;

```

Query ketiga membangun *funnel analisis*, yaitu alur dari pendaftaran (admission) ke keterdaftar dalam mata kuliah (enrollment) hingga kelulusan (graduation). Analisis funnel seperti ini sangat penting untuk melihat *rasio konversi* pada setiap tahapan pendidikan. Misalnya, berapa banyak dari pendaftar yang benar-benar masuk kuliah, dan berapa banyak dari mereka yang berhasil lulus. Data ini membantu mengukur efektivitas program studi serta menemukan potensi masalah seperti dropout.

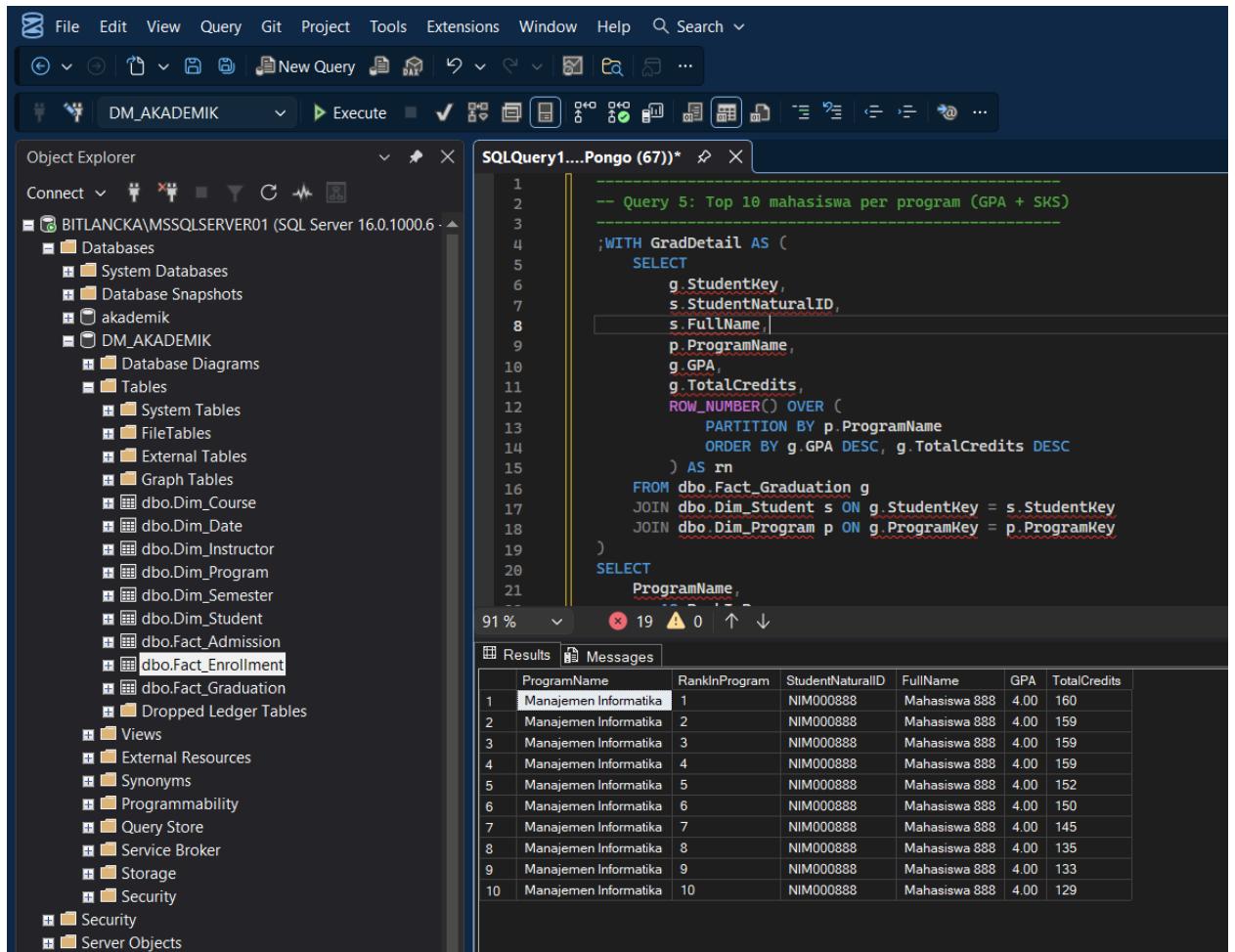
```

-- Query 4: Distribusi lama studi dan GPA per program
SELECT
    p.ProgramName,
    COUNT(*) AS TotalGraduates,
    AVG(g.GPA) AS AvgGPA,
    MIN(g.GPA) AS MinGPA,
    MAX(g.GPA) AS MaxGPA,
    AVG(CAST(g.StudyDuration AS FLOAT)) AS AvgStudyMonths,
    MIN(g.StudyDuration) AS MinStudyMonths,
    MAX(g.StudyDuration) AS MaxStudyMonths
FROM dbo.Fact_Graduation g
JOIN dbo.Dim_Program p ON g.ProgramKey = p.ProgramKey
GROUP BY p.ProgramName
ORDER BY p.ProgramName;
GO

```

Query ini menghitung distribusi lama studi dan GPA (IPK) bagi mahasiswa yang telah lulus pada setiap program. Hasilnya mencakup jumlah lulusan, rata-rata IPK, IPK minimum-maksimum, serta lama studi rata-rata hingga lulus. Output ini penting untuk

menganalisis *mutu lulusan*, konsistensi performa akademik, serta efektivitas kurikulum suatu program studi.



The screenshot shows the SSMS interface with the following details:

- Object Explorer:** Shows the database structure under "BITLANKA\MYSQLSERVER01 (SQL Server 16.0.1000.6)". It includes Databases, Tables (with System Tables, FileTables, External Tables, Graph Tables), Views, Synonyms, Programmability, Query Store, Service Broker, Storage, Security, and Server Objects.
- SQLQuery1...Pongo (67)\*:** The query window contains a T-SQL script. The code is a common table expression (CTE) named "GradDetail" that ranks students by GPA and total credits. It then selects the top 10 rows for each program. The results are displayed in the "Results" tab.
- Results:** The output table shows 10 rows of data for the program "Manajemen Informatika". The columns are: ProgramName, RankInProgram, StudentNaturalID, FullName, GPA, and TotalCredits.

	ProgramName	RankInProgram	StudentNaturalID	FullName	GPA	TotalCredits
1	Manajemen Informatika	1	NIM000888	Mahasiswa 888	4.00	160
2	Manajemen Informatika	2	NIM000888	Mahasiswa 888	4.00	159
3	Manajemen Informatika	3	NIM000888	Mahasiswa 888	4.00	159
4	Manajemen Informatika	4	NIM000888	Mahasiswa 888	4.00	159
5	Manajemen Informatika	5	NIM000888	Mahasiswa 888	4.00	152
6	Manajemen Informatika	6	NIM000888	Mahasiswa 888	4.00	150
7	Manajemen Informatika	7	NIM000888	Mahasiswa 888	4.00	145
8	Manajemen Informatika	8	NIM000888	Mahasiswa 888	4.00	135
9	Manajemen Informatika	9	NIM000888	Mahasiswa 888	4.00	133
10	Manajemen Informatika	10	NIM000888	Mahasiswa 888	4.00	129

Query ini menyusun daftar 10 mahasiswa terbaik pada setiap program berdasarkan GPA dan jumlah SKS. Dengan menggunakan ranking per program, institusi dapat dengan mudah mengidentifikasi mahasiswa berprestasi, memberikan penghargaan, atau memanfaatkan data ini untuk laporan akreditasi. Query ini juga menunjukkan kualitas lulusan unggulan di masing-masing program.

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
-- Query 6: Pengaruh Attendance terhadap NumericGrade
;WITH EnrBucket AS (
    SELECT
        CASE
            WHEN AttendanceRate < 60 THEN '< 60%'
            WHEN AttendanceRate < 75 THEN '60-74%'
            WHEN AttendanceRate < 90 THEN '75-89%'
            ELSE '>= 90%'
        END AS AttendanceBucket,
        NumericGrade
    FROM dbo.Fact_Enrollment
    WHERE NumericGrade IS NOT NULL
)
SELECT
    AttendanceBucket,
    COUNT(*) AS TotalEnrollments,
    AVG(NumericGrade) AS AvgGrade,
    MIN(NumericGrade) AS MinGrade,
    MAX(NumericGrade) AS MaxGrade

```

AttendanceBucket	TotalEnrollments	AvgGrade	MinGrade	MaxGrade
< 60%	11896	2.972461	0.00	4.00
60-74%	2988	2.947255	0.00	4.00
75-89%	2975	2.976941	0.00	4.00
>= 90%	2141	2.966978	0.00	4.00

Query keenam mengelompokkan mahasiswa berdasarkan persentase kehadiran (AttendanceRate) ke dalam bucket (misalnya <60%, 60–74%, 75–89%, ≥90%). Kemudian dihitung rata-rata nilai, nilai minimum, dan maksimum pada tiap kelompok. Analisis ini bertujuan melihat apakah *tingkat kehadiran berpengaruh terhadap nilai*, dan biasanya menunjukkan korelasi positif antara kehadiran tinggi dan nilai lebih baik.

Query keenam mengelompokkan mahasiswa berdasarkan persentase kehadiran (AttendanceRate) ke dalam bucket (misalnya <60%, 60–74%, 75–89%, ≥90%). Kemudian dihitung rata-rata nilai, nilai minimum, dan maksimum pada tiap kelompok. Analisis ini bertujuan melihat apakah *tingkat kehadiran berpengaruh terhadap nilai*, dan biasanya menunjukkan korelasi positif antara kehadiran tinggi dan nilai lebih baik.

The screenshot shows the SQL Server Management Studio (SSMS) interface. The title bar reads "SQLQuery1....Pongo (67)". The left pane is the Object Explorer, showing the connection to "BITLANCA\SQLSERVER01 (SQL Server 16.0.1000.6)" and the database "DM\_AKADEMIK". The right pane contains a query window with the following T-SQL code:

```
-- Query 7: Trend enrollment per semester dan program
SELECT
    sem.SemesterCode,
    d.[Year],
    p.ProgramName,
    COUNT(*) AS TotalEnrollments,
    COUNT(DISTINCT f.StudentKey) AS DistinctStudents,
    AVG(f.NumericGrade) AS AvgGrade
FROM
    dbo.Fact_Enrollment f
JOIN
    dbo.Dim_Semester sem ON f.SemesterKey = sem.SemesterKey
JOIN
    dbo.Dim_Date d ON f.DateKey = d.DateKey
JOIN
    dbo.Dim_Course c ON f.CourseKey = c.CourseKey
JOIN
    dbo.Dim_Program p ON c.ProgramNaturalID = p.ProgramCode
GROUP BY sem.SemesterCode, d.[Year], p.ProgramName
ORDER BY d.[Year], sem.SemesterCode, p.ProgramName;
GO
```

The results pane shows a single row of data:

	SemesterCode	Year	ProgramName	TotalEnrollments	DistinctStudents	AvgGrade
1	2019-Genap	2024	Fisika	20000	1	2.968775

Query terakhir memberikan gambaran jumlah enrollment, jumlah mahasiswa unik, dan rata-rata nilai pada setiap semester untuk tiap program studi. Data ini membantu memantau perkembangan jumlah peserta kelas per semester, mengidentifikasi perubahan jumlah mahasiswa, dan menganalisis performa akademik berdasarkan semester. Informasi ini bermanfaat untuk perencanaan akademik dan monitoring kapasitas program.