PROGRAM STUDI SAINS DATA

FAKULTAS SAINS

INSTITUT TEKNOLOGI SUMATERA

TUGAS BESAR

PEMBANGUNAN DATA MART

PERGUDANGAN DATA

Kode Mata Kuliah	: SD25-31007
Nama Mata Kuliah	: Pergudangan Data
Bobot SKS	: 3 (2-1)
Tahun Ajaran	: 2025
Bobot Penilaian	: 30% dari nilai akhir

Lampung Selatan
2025

Contents

1 PENDAHULUAN

1.1 Deskripsi Tugas Besar

Tugas Besar Pergudangan Data merupakan proyek kelompok yang dirancang untuk memberikan pengalaman komprehensif dalam merancang, membangun, dan mengimplementasikan data mart dari tahap konseptual hingga produksi. Mahasiswa akan bekerja secara kolaboratif untuk membangun solusi business intelligence yang mendukung pengambilan keputusan berbasis data pada unit kerja di lingkungan Institut Teknologi Sumatera (ITERA).

1.2 Tujuan Pembelajaran

Tugas besar ini bertujuan untuk:

1. Menerapkan konsep desain konseptual, logikal, dan fisikal gudang data dalam proyek nyata
2. Mengimplementasikan arsitektur data mart menggunakan teknologi standar industri
3. Membangun sistem ETL untuk integrasi data dari berbagai sumber
4. Menerapkan best practices dalam pengembangan data warehouse
5. Mengembangkan kemampuan kolaborasi tim dan project management
6. Menghasilkan dokumentasi teknis yang komprehensif

1.3 Capaian Pembelajaran Mata Kuliah (CPMK)

Tugas besar ini mencakup seluruh CPMK mata kuliah:

- **CPMK 072**: Mempersiapkan data dengan membersihkan, mentransformasi, dan memformat data mentah agar siap digunakan untuk analisis (Misi 2 & 3)
- **CPMK 092**: Mengimplementasikan arsitektur manajemen data sesuai dengan desain arsitektur data menggunakan teknologi yang tepat (Misi 1, 2, & 3)
- **CPMK 102**: Menerapkan metode, sumber daya, alat komputasi modern standar industri yang sesuai untuk aktivitas praktik sains data (Misi 2 & 3)

1.4 Bobot Penilaian

Komponen	**Bobot Misi**	**Bobot Total (30%)**
Misi 1: Desain Konseptual & Logikal	30%	9%
Misi 2: Desain Fisikal & Development	35%	10.5%
Misi 3: Implementasi Produksi	35%	10.5%
Total	**100%**	**30%**

1.5 Pembagian Kelompok dan Domain

Setiap kelompok terdiri dari 3-4 mahasiswa dan ditugaskan untuk membangun data mart untuk unit kerja spesifik di ITERA. Pembagian kelompok mengacu pada dokumen ”Pembagian-Kelompok-Tubes-DW.pdf” dengan unit sebagai berikut:

- Kelompok 1: Fakultas Sains
- Kelompok 2: Akademik
- Kelompok 3: Non Akademik
- Kelompok 4: Kemahasiswaan
- Kelompok 5: LPPM
- Kelompok 6: LPMPP
- Kelompok 7: Keuangan
- Kelompok 8: Kepegawaian
- Kelompok 9: Keamanan Siber
- Kelompok 10: Satuan Pengawas Internal
- Kelompok 11: Satu Data
- Kelompok 12: Kebun Raya
- Kelompok 13: K3L
- Kelompok 14: Sarana dan Prasarana
- Kelompok 15: Fakultas Teknologi Industri
- Kelompok 16: Fakultas Teknologi Infrastruktur dan Kewilayahan
- Kelompok 17: Biro Akademik (Perencanaan)
- Kelompok 18: Biro Akademik (Akademik)
- Kelompok 19: Biro Akademik (Umum)

2 TEKNOLOGI DAN TOOLS

2.1 Platform Utama

1. **SQL Server di Azure VM**
   - Database engine untuk data warehouse
   - SQL Server Integration Services (SSIS) untuk ETL
   - SQL Server Analysis Services (SSAS) untuk OLAP (opsional)
   - SQL Server Reporting Services (SSRS) untuk reporting (opsional)
2. **GitHub**
   - Version control untuk kode SQL, ETL scripts
   - Dokumentasi proyek (README, Wiki)
   - Issue tracking dan project management
   - Kolaborasi tim

2.2 Tools Pendukung

- **Desain & Modeling**: SQL Server Data Tools (SSDT), draw.io, Lucidchart, ERD-Plus
- **ETL Development**: SSIS, Azure Data Factory (alternatif)
- **Database Management**: SQL Server Management Studio (SSMS), Azure Data Studio
- **BI & Visualization**: Power BI Desktop, Tableau Public (opsional)
- **Data Quality**: SSIS Data Quality Services, SQL profiling tools
- **Documentation**: Microsoft Word/LaTeX, Markdown untuk GitHub

2.3 Pemilihan Arsitektur

Kelompok dapat memilih salah satu pendekatan arsitektur:

2.3.1 Kimball (Dimensional Modeling)

+ Bottom-up approach
+ Fokus pada proses bisnis spesifik
+ Star schema yang simpel dan performa tinggi
+ Implementasi cepat (iterative)
– Potensi redundansi data

2.3.2 Inmon (Enterprise Data Warehouse)

+ Top-down approach
+ Integrasi enterprise-wide
+ Single source of truth
+ Normalized structure

− Kompleksitas tinggi, implementasi lama

2.3.3 Data Vault

+ Agile dan scalable
+ Historical tracking yang baik
+ Flexible untuk perubahan
+ Audit trail built-in

− Learning curve yang tinggi

Rekomendasi: Untuk tugas besar ini, **Kimball** direkomendasikan karena scope yang lebih terfokus dan implementasi yang lebih praktis untuk proyek semesteran.

3 MISI 1: DESAIN KONSEPTUAL DAN LOGIKAL

3.1 Deskripsi Misi

Misi 1 fokus pada tahap awal perancangan data mart, yaitu memahami kebutuhan bisnis, merancang model konseptual, dan mengembangkan desain logikal yang akan menjadi blueprint implementasi.

3.2 Tujuan Misi

- Memahami proses bisnis dan kebutuhan analitik unit kerja
- Mengidentifikasi sumber data dan metrik bisnis yang relevan
- Merancang model dimensional yang optimal
- Mendokumentasikan desain konseptual dan logikal secara komprehensif

3.3 Deliverables

1. Dokumen Requirements Analysis
2. Entity Relationship Diagram (ERD)
3. Dimensional Model (Star/Snowflake Schema)
4. Data Dictionary
5. Dokumentasi GitHub (README.md)

3.4 Timeline

Durasi: 1 minggu

Minggu	Aktivitas	Output
1	Kick-off, analisis domain	Pemahaman bisnis, identifikasi stakeholder
2	Requirements gathering	Dokumen requirements, data sources
3	Desain konseptual	ERD, identifikasi fakta/dimensi
4	Desain logikal	Star/snowflake schema, data dictionary

3.5 Langkah Pengerjaan Detail

3.5.1 Step 1: Business Requirements Analysis

Tujuan: Memahami kebutuhan bisnis dan analitik unit kerja

Aktivitas:

1. **Identifikasi Stakeholders**

   - Tentukan siapa pengguna utama data mart (manajer, staff, pimpinan)
   - Identifikasi decision makers yang akan menggunakan insight

2. **Analisis Proses Bisnis**
   - Dokumentasikan proses bisnis utama unit kerja
   - Identifikasi key performance indicators (KPI)
   - Tentukan metrik bisnis yang perlu dimonitor

3. **Kebutuhan Analitik**
   - Daftar pertanyaan bisnis yang perlu dijawab
   - Jenis laporan yang dibutuhkan (daily, weekly, monthly)
   - Level agregasi dan granularitas data

Contoh untuk Unit Akademik:

```
Proses Bisnis:
1. Penerimaan Mahasiswa Baru
2. Perkuliahan dan Pengajaran
3. Evaluasi Akademik
4. Kelulusan

KPI:
- Jumlah mahasiswa aktif per program studi
- Rata-rata IPK per angkatan
- Tingkat kelulusan tepat waktu
- Rasio dosen:mahasiswa

Kebutuhan Analitik:
- Dashboard monitoring mahasiswa aktif
- Analisis performa akademik per prodi
- Trend penerimaan mahasiswa 5 tahun terakhir
- Analisis drop-out rate
```

Output:

- Dokumen Business Requirements (5-10 halaman)
- Daftar KPI dan metrik bisnis
- Use case diagram (opsional)

3.5.2 Step 2: Data Source Identification

Tujuan: Mengidentifikasi dan menganalisis sumber data yang tersedia
Aktivitas:

1. **Identifikasi Sumber Data**
   - Database operasional (OLTP)
   - File Excel/CSV

- External APIs
- Manual data entry

2. **Data Source Analysis**
    - Struktur dan schema data source
    - Volume dan growth rate data
    - Kualitas data (completeness, accuracy)
    - Frekuensi update data
3. **Data Profiling**
    - Analisis distribusi data
    - Identifikasi null values dan outliers
    - Deteksi duplikasi
    - Konsistensi format data

Output:

Data Source	Type	Volume	Update Frequency
SIAKAD DB	SQL Server	100K rows	Real-time
Absensi Excel	CSV	50K rows	Daily
...	...	...	...

3.5.3 Step 3: Conceptual Design - ERD

Tujuan: Membuat model konseptual yang merepresentasikan entitas bisnis
Aktivitas:

1. **Identifikasi Entitas**
    - Entitas utama (mahasiswa, dosen, mata kuliah, dll)
    - Atribut untuk setiap entitas
    - Primary keys dan foreign keys
2. **Definisi Relationships**
    - One-to-One, One-to-Many, Many-to-Many
    - Kardinalitas relationships
    - Optional vs mandatory relationships
3. **Gambar ERD**
    - Gunakan tools: draw.io, Lucidchart, ERDPlus
    - Gunakan notasi standar (Crow's Foot atau Chen)
    - Pastikan ERD jelas dan mudah dipahami

Output:

- ERD diagram (high-resolution image atau PDF)
- Deskripsi setiap entitas dan relationship
- Dokumen asumsi dan business rules

3.5.4 Step 4: Logical Design - Dimensional Model

Tujuan: Mengkonversi model konseptual ke dimensional model untuk data warehouse
Aktivitas:

1. **Identifikasi Fact Tables**
   - Tentukan business process yang akan dimodel
   - Pilih grain (level detail) fact table
   - Identifikasi measures (metrik numerik)
   - Klasifikasi additivity (additive, semi-additive, non-additive)
2. **Identifikasi Dimension Tables**
   - Dimension yang mendukung analisis (Who, What, Where, When, Why, How)
   - Atribut deskriptif untuk filtering dan grouping
   - Hierarki dalam dimensi (untuk drill-down/roll-up)
3. **Desain Star/Snowflake Schema**
   - Gambar dimensional model
   - Tunjukkan fact table di tengah
   - Dimension tables mengelilingi fact
   - Tandai foreign keys dan relationships
4. **Penentuan Surrogate Keys**
   - Gunakan integer surrogate keys untuk semua dimensi
   - Definisikan naming convention (contoh: DimCustomer_SK)
   - Rencanakan strategy untuk SCD (Slowly Changing Dimensions)

Contoh untuk Unit Akademik:

```
Fact Table: Fact_Enrollment
- EnrollmentKey (PK, surrogate key)
- DateKey (FK to Dim_Date)
- StudentKey (FK to Dim_Student)
- CourseKey (FK to Dim_Course)
- InstructorKey (FK to Dim_Instructor)
- ProgramKey (FK to Dim_Program)
- Grade (measure)
- Credits (measure)
- TuitionFee (measure)
- AttendanceRate (measure)
```

```

Dimension Tables:
1. Dim_Date (Time dimension)
2. Dim_Student (Student demographics)
3. Dim_Course (Course details)
4. Dim_Instructor (Faculty information)
5. Dim_Program (Academic programs)
```

Output:

- Star/Snowflake schema diagram
- Tabel spesifikasi fact dan dimension tables
- Data dictionary lengkap

3.5.5 Step 5: Data Dictionary

Tujuan: Mendokumentasikan setiap elemen data secara detail

Format Data Dictionary:

Table	Column	Data Type	PK/FK	Description	Business Rule
Fact_Enrollment	EnrollmentKey	INT	PK	Unique identifier	Auto-increment
Fact_Enrollment	Grade	DECIMAL(3,2)		Student grade	0.00 - 4.00
Dim_Student	StudentKey	INT	PK	Surrogate key	Auto-increment
Dim_Student	NIM	VARCHAR(20)	Natural Key	Student ID	Unique, NOT NULL
...	...	...	...	...	...

3.5.6 Step 6: GitHub Repository Setup

Tujuan: Menyiapkan repository untuk kolaborasi dan version control

Struktur Repository:

```
repository-name/
  README.md
  docs/
     01-requirements/
        business-requirements.md
        data-sources.md
     02-design/
        ERD.png
        dimensional-model.png
        data-dictionary.xlsx
     presentations/
  sql/
     schema/
  etl/
  reports/
```

README.md Template:

```
# Data Mart - [Nama Unit]
Tugas Besar Pergudangan Data - Kelompok [X]

```

```
## Team Members
- NIM - Nama (Role)
- ...

## Project Description
Brief description of the data mart project

## Business Domain
Description of the business unit

## Architecture
- Approach: Kimball/Inmon/Data Vault
- Platform: SQL Server on Azure VM
- ETL: SSIS

## Key Features
- Fact tables: ...
- Dimension tables: ...
- KPIs: ...

## Documentation
- [Business Requirements](docs/01-requirements/)
- [Design Documents](docs/02-design/)

## Timeline
- Misi 1: [Tanggal]
- Misi 2: [Tanggal]
- Misi 3: [Tanggal]
```

3.6 Kriteria Penilaian Misi 1

Aspek	**Kriteria**	**Bobot**
Requirements Analysis	Kelengkapan analisis kebutuhan bisnis, identifikasi KPI, kebutuhan analitik yang jelas dan terukur	20%
Conceptual Design	Kualitas ERD: kelengkapan entitas, accuracy relationships, clarity diagram	20%
Dimensional Model	Desain star/snowflake schema yang tepat, pemilihan grain yang optimal, identifikasi measures yang relevan	30%
Data Dictionary	Kelengkapan dokumentasi, kejelasan definisi, konsistensi naming convention	15%
Documentation	Kualitas dokumentasi GitHub, struktur repository, README yang informatif	10%
Presentation	Kemampuan menjelaskan desain, menjawab pertanyaan, profesionalisme	5%
Total		**100%**

4 MISI 2: DESAIN FISIKAL DAN DEVELOPMENT

4.1 Deskripsi Misi

Misi 2 fokus pada implementasi fisik dari desain logikal, pembangunan database, dan pengembangan proses ETL untuk mengisi data warehouse.

4.2 Tujuan Misi

- Mengimplementasikan desain logikal ke dalam SQL Server
- Membangun proses ETL yang robust dan efficient
- Menerapkan teknik optimasi performa (indexing, partitioning)
- Melakukan data quality assurance
- Menyiapkan infrastruktur untuk deployment produksi

4.3 Deliverables

1. SQL DDL Scripts (CREATE TABLE, INDEX, PARTITION)
2. ETL Package (SSIS atau scripts)
3. Data Quality Report
4. Performance Optimization Documentation
5. Testing Documentation
6. Updated GitHub Repository

4.4 Timeline

Durasi: 1 minggu

Minggu	**Aktivitas**	**Output**
5	Physical design, DDL development	SQL scripts untuk schema
6	ETL design	ETL architecture, mapping document
7	ETL implementation	Working ETL packages
8	Optimization	Indexed, partitioned tables
9	Testing	Test results, quality report

4.5 Langkah Pengerjaan Detail

4.5.1 Step 1: Physical Database Design

Tujuan: Mengimplementasikan dimensional model ke SQL Server
Aktivitas:

1. **Database Setup**

```
-- Create Database
CREATE DATABASE DM_[UnitName]_DW
ON PRIMARY
(
    NAME = N'DM_[UnitName]_DW_Data',
    FILENAME = N'D:\Data\DM_[UnitName]_DW_Data.mdf',
    SIZE = 1GB,
    MAXSIZE = UNLIMITED,
    FILEGROWTH = 256MB
)
LOG ON
(
    NAME = N'DM_[UnitName]_DW_Log',
    FILENAME = N'E:\Logs\DM_[UnitName]_DW_Log.ldf',
    SIZE = 256MB,
    MAXSIZE = 2GB,
    FILEGROWTH = 64MB
);
GO

USE DM_[UnitName]_DW;
GO
```

2. **Create Dimension Tables**

```
-- Dimension Table: Dim_Date
CREATE TABLE dbo.Dim_Date (
    DateKey INT PRIMARY KEY NOT NULL,
    FullDate DATE NOT NULL,
    DayNumberOfWeek TINYINT NOT NULL,
    DayName VARCHAR(10) NOT NULL,
    DayNumberOfMonth TINYINT NOT NULL,
    DayNumberOfYear SMALLINT NOT NULL,
    WeekNumberOfYear TINYINT NOT NULL,
    MonthName VARCHAR(10) NOT NULL,
    MonthNumber TINYINT NOT NULL,
    Quarter TINYINT NOT NULL,
    QuarterName VARCHAR(6) NOT NULL,
    Year SMALLINT NOT NULL,
    IsWeekend BIT NOT NULL,
    IsHoliday BIT NOT NULL,
    HolidayName VARCHAR(50) NULL,
    AcademicYear VARCHAR(9) NULL,
    Semester TINYINT NULL,
```

```sql
    CONSTRAINT CK_Dim_Date_Quarter CHECK (Quarter BETWEEN 1 AND 4),
    CONSTRAINT CK_Dim_Date_Month CHECK (MonthNumber BETWEEN 1 AND 12)
);
GO

-- Dimension Table: Dim_Student (Example)
CREATE TABLE dbo.Dim_Student (
    StudentKey INT IDENTITY(1,1) PRIMARY KEY NOT NULL,
    NIM VARCHAR(20) UNIQUE NOT NULL,
    StudentName VARCHAR(100) NOT NULL,
    Gender CHAR(1) CHECK (Gender IN ('M','F')),
    BirthDate DATE,
    EnrollmentDate DATE NOT NULL,
    ProgramCode VARCHAR(10) NOT NULL,
    ProgramName VARCHAR(100) NOT NULL,
    Faculty VARCHAR(100) NOT NULL,
    EntryYear SMALLINT NOT NULL,
    Status VARCHAR(20) NOT NULL, -- Active, Graduated, Drop-out, etc.
    -- SCD Type 2 attributes
    EffectiveDate DATE NOT NULL DEFAULT GETDATE(),
    ExpiryDate DATE NULL,
    IsCurrent BIT NOT NULL DEFAULT 1,
    -- Metadata
    CreatedDate DATETIME DEFAULT GETDATE(),
    ModifiedDate DATETIME DEFAULT GETDATE()
);
GO

CREATE INDEX IX_Dim_Student_NIM ON dbo.Dim_Student(NIM);
CREATE INDEX IX_Dim_Student_Program ON dbo.Dim_Student(ProgramCode);
CREATE INDEX IX_Dim_Student_Current ON dbo.Dim_Student(IsCurrent)
    WHERE IsCurrent = 1;
GO
```

3. **Create Fact Tables**

```sql
-- Fact Table: Fact_Enrollment
CREATE TABLE dbo.Fact_Enrollment (
    EnrollmentKey BIGINT IDENTITY(1,1) PRIMARY KEY NOT NULL,
    -- Foreign Keys to Dimensions
    DateKey INT NOT NULL,
    StudentKey INT NOT NULL,
    CourseKey INT NOT NULL,
    InstructorKey INT NOT NULL,
    ProgramKey INT NOT NULL,
    -- Degenerate Dimensions
    EnrollmentID VARCHAR(50) NOT NULL,
    ClassCode VARCHAR(20) NOT NULL,
    -- Measures
    Credits DECIMAL(3,1) NOT NULL,
    Grade DECIMAL(3,2) NULL,
    GradePoint DECIMAL(3,2) NULL,
    AttendanceCount INT DEFAULT 0,
    AttendanceRate DECIMAL(5,2) NULL,
    TuitionFee DECIMAL(12,2) NOT NULL,
    -- Flags
    IsPassed BIT NULL,
    IsDropped BIT DEFAULT 0,
    -- Metadata
    SourceSystem VARCHAR(50) NOT NULL,
    LoadDate DATETIME DEFAULT GETDATE(),
    -- Foreign Key Constraints
    CONSTRAINT FK_Fact_Enrollment_Date
        FOREIGN KEY (DateKey) REFERENCES dbo.Dim_Date(DateKey),
    CONSTRAINT FK_Fact_Enrollment_Student
        FOREIGN KEY (StudentKey) REFERENCES dbo.Dim_Student(StudentKey),
    CONSTRAINT FK_Fact_Enrollment_Course
        FOREIGN KEY (CourseKey) REFERENCES dbo.Dim_Course(CourseKey),
    CONSTRAINT FK_Fact_Enrollment_Instructor
        FOREIGN KEY (InstructorKey) REFERENCES dbo.Dim_Instructor(InstructorKey),
    CONSTRAINT FK_Fact_Enrollment_Program
        FOREIGN KEY (ProgramKey) REFERENCES dbo.Dim_Program(ProgramKey)
);
GO
```

Output:

- File SQL: `01_Create_Database.sql`
- File SQL: `02_Create_Dimensions.sql`
- File SQL: `03_Create_Facts.sql`

4.5.2 Step 2: Indexing Strategy

Tujuan: Mengoptimalkan query performance dengan indexing yang tepat
Aktivitas:

1. **Clustered Index on Fact Table**

```sql
-- Rebuild clustered index dengan partitioning (optional)
CREATE CLUSTERED INDEX CIX_Fact_Enrollment_DateKey
ON dbo.Fact_Enrollment(DateKey, EnrollmentKey);
GO
```

2. **Non-Clustered Indexes**

```sql
-- Index untuk foreign keys (join optimization)
CREATE NONCLUSTERED INDEX IX_Fact_Enrollment_Student
ON dbo.Fact_Enrollment(StudentKey)
INCLUDE (Credits, Grade);
GO

CREATE NONCLUSTERED INDEX IX_Fact_Enrollment_Course
ON dbo.Fact_Enrollment(CourseKey)
INCLUDE (Grade, Credits);
GO

CREATE NONCLUSTERED INDEX IX_Fact_Enrollment_Program
ON dbo.Fact_Enrollment(ProgramKey, DateKey)
INCLUDE (StudentKey, Grade);
GO

-- Covering index untuk common queries
CREATE NONCLUSTERED INDEX IX_Fact_Enrollment_Covering
ON dbo.Fact_Enrollment(DateKey, ProgramKey)
INCLUDE (StudentKey, Credits, Grade, TuitionFee);
GO
```

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

```sql
-- Columnstore index untuk analytical queries
CREATE NONCLUSTERED COLUMNSTORE INDEX NCCIX_Fact_Enrollment
ON dbo.Fact_Enrollment
(
    DateKey, StudentKey, CourseKey, ProgramKey,
    Credits, Grade, TuitionFee, AttendanceRate
);
GO
```

Output: File SQL 04_Create_Indexes.sql

4.5.3 Step 3: Partitioning Strategy

Tujuan: Meningkatkan manageability dan performance untuk large tables

Aktivitas:

```sql
-- Create Partition Function (by 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
CREATE PARTITION SCHEME PS_AcademicYear
AS PARTITION PF_AcademicYear
ALL TO ([PRIMARY]);
GO

-- Create Partitioned Fact Table
CREATE TABLE dbo.Fact_Enrollment_Partitioned (
    -- columns same as before
) ON PS_AcademicYear(DateKey);
GO
```

Output: File SQL `05_Create_Partitions.sql`

4.5.4 Step 4: ETL Design

Tujuan: Merancang proses Extract, Transform, Load yang efisien
Aktivitas:

1. **ETL Architecture Design**
   - Source to Staging: Extract raw data
   - Staging to Integration: Data cleansing dan transformation
   - Integration to Data Warehouse: Load ke fact dan dimension tables

2. **Create Staging Tables**

```sql
-- Staging Schema
CREATE SCHEMA stg;
GO

-- Staging Table for Student Data
CREATE TABLE stg.Student (
    NIM VARCHAR(20),
    StudentName VARCHAR(100),
    Gender CHAR(1),
    BirthDate DATE,
    EnrollmentDate DATE,
    ProgramCode VARCHAR(10),
    Status VARCHAR(20),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO

-- Staging Table for Enrollment Data
CREATE TABLE stg.Enrollment (
    EnrollmentID VARCHAR(50),
    NIM VARCHAR(20),
    CourseCode VARCHAR(20),
    InstructorCode VARCHAR(20),
    Semester VARCHAR(10),
    AcademicYear VARCHAR(9),
    Credits DECIMAL(3,1),
    Grade DECIMAL(3,2),
    TuitionFee DECIMAL(12,2),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO
```

3. **ETL Mapping Document**

Source	**Source Column**	**Target**	**Target Column**	**Transformation**
SIAKAD.dbo.mahasiswa	nim	Dim_Student	NIM	Direct mapping
SIAKAD.dbo.mahasiswa	nama	Dim_Student	StudentName	UPPER(TRIM(nama))
SIAKAD.dbo.mahasiswa	jenis_kelamin	Dim_Student	Gender	CASE WHEN jenis_kelamin='Laki-laki' THEN 'M' ELSE 'F' END
...	...	...	...	...

Output:

- Dokumen ETL Architecture
- ETL Mapping Spreadsheet
- File SQL: `06_Create_Staging.sql`

4.5.5 Step 5: ETL Implementation

Tujuan: Mengimplementasikan ETL menggunakan SSIS atau T-SQL scripts

Opsi 1: Menggunakan SSIS (Recommended)

1. **Create SSIS Project**
   - Buka SQL Server Data Tools (SSDT)
   - Create new Integration Services Project
   - Beri nama: ETL_[UnitName]_DW
2. **Package 1: Load Dimensions**
   - Data Flow Task: Extract dari source
   - Derived Column: Transformasi data
   - Lookup Transformation: SCD Type 2 handling
   - Slowly Changing Dimension: Insert/Update dimension
3. **Package 2: Load Facts**
   - Data Flow Task: Extract enrollment data
   - Lookup Transformations: Resolve dimension keys
   - Derived Column: Calculate measures
   - OLE DB Destination: Load to fact table

4. **Master Package**

    - Sequence Container: Truncate staging
    - Execute SQL Task: Disable indexes
    - Execute Package Task: Load dimensions
    - Execute Package Task: Load facts
    - Execute SQL Task: Rebuild indexes
    - Execute SQL Task: Update statistics

Opsi 2: Menggunakan T-SQL Stored Procedures

```sql
-- Stored Procedure: Load Dim_Student with SCD Type 2
CREATE PROCEDURE dbo.usp_Load_Dim_Student
AS
BEGIN
    SET NOCOUNT ON;

    -- Expire old records
    UPDATE d
    SET
        ExpiryDate = GETDATE(),
        IsCurrent = 0,
        ModifiedDate = GETDATE()
    FROM dbo.Dim_Student d
    INNER JOIN stg.Student s ON d.NIM = s.NIM
    WHERE d.IsCurrent = 1
        AND (
            d.StudentName <> s.StudentName OR
            d.Status <> s.Status OR
            d.ProgramCode <> s.ProgramCode
        );

    -- Insert new records (new students or changed records)
    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,
        p.Faculty,
        YEAR(s.EnrollmentDate),
        s.Status,
        GETDATE(),
        1
    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
    );

END;
GO

-- Stored Procedure: Load Fact_Enrollment
CREATE PROCEDURE dbo.usp_Load_Fact_Enrollment
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO dbo.Fact_Enrollment (
        DateKey, StudentKey, CourseKey, InstructorKey, ProgramKey,
        EnrollmentID, ClassCode,
        Credits, Grade, GradePoint, TuitionFee,
        IsPassed, SourceSystem
    )
    SELECT
        CAST(CONVERT(VARCHAR(8), s.EnrollmentDate, 112) AS INT) AS DateKey,
        ds.StudentKey,
        dc.CourseKey,
        di.InstructorKey,
        dp.ProgramKey,
        s.EnrollmentID,
        s.ClassCode,
        s.Credits,
        s.Grade,
        CASE
            WHEN s.Grade >= 3.50 THEN 4.00
            WHEN s.Grade >= 3.00 THEN 3.50
            WHEN s.Grade >= 2.50 THEN 3.00
            ELSE s.Grade
        END AS GradePoint,
        s.TuitionFee,
        CASE WHEN s.Grade >= 2.00 THEN 1 ELSE 0 END AS IsPassed,
        'SIAKAD' AS SourceSystem
    FROM stg.Enrollment s
    INNER JOIN dbo.Dim_Student ds ON s.NIM = ds.NIM AND ds.IsCurrent = 1
    INNER JOIN dbo.Dim_Course dc ON s.CourseCode = dc.CourseCode
    INNER JOIN dbo.Dim_Instructor di ON s.InstructorCode = di.InstructorCode
    INNER JOIN dbo.Dim_Program dp ON ds.ProgramCode = dp.ProgramCode
    WHERE NOT EXISTS (
        SELECT 1
        FROM dbo.Fact_Enrollment f
        WHERE f.EnrollmentID = s.EnrollmentID
    );

END;
GO

-- Master ETL Procedure
CREATE PROCEDURE dbo.usp_Master_ETL
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        -- Step 1: Load Dimensions
        EXEC dbo.usp_Load_Dim_Student;
        EXEC dbo.usp_Load_Dim_Course;
        EXEC dbo.usp_Load_Dim_Instructor;

        -- Step 2: Load Facts
        EXEC dbo.usp_Load_Fact_Enrollment;

        -- Step 3: Update Statistics
        UPDATE STATISTICS dbo.Dim_Student;
        UPDATE STATISTICS dbo.Fact_Enrollment;

        COMMIT TRANSACTION;

        PRINT 'ETL Completed Successfully';
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0
            ROLLBACK TRANSACTION;

        DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
        RAISERROR(@ErrorMessage, 16, 1);
    END CATCH
END;
GO
```

Output:

- SSIS Project files (.dtsx) atau
- SQL Scripts: `07_ETL_Procedures.sql`
- ETL Execution Log

4.5.6 Step 6: Data Quality Assurance

Tujuan: Memastikan kualitas data yang dimuat ke data warehouse
Aktivitas:

1. **Data Quality Checks**

```sql
-- Check 1: Completeness - NULL values
SELECT
    'Dim_Student' AS TableName,
    COUNT(*) AS TotalRows,
    SUM(CASE WHEN NIM IS NULL THEN 1 ELSE 0 END) AS NullNIM,
    SUM(CASE WHEN StudentName 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;

-- Check 2: Consistency - Referential Integrity
SELECT
    COUNT(*) AS OrphanRecords
FROM dbo.Fact_Enrollment f
LEFT JOIN dbo.Dim_Student s ON f.StudentKey = s.StudentKey
WHERE s.StudentKey IS NULL;

-- Check 3: Accuracy - Valid ranges
SELECT
    COUNT(*) AS InvalidGrades
FROM dbo.Fact_Enrollment
WHERE Grade < 0 OR Grade > 4.00;

-- Check 4: Duplicates
SELECT
    EnrollmentID,
    COUNT(*) AS DuplicateCount
FROM dbo.Fact_Enrollment
GROUP BY EnrollmentID
HAVING COUNT(*) > 1;

-- Check 5: Record Counts Reconciliation
SELECT
    'Source' AS DataSource,
    COUNT(*) AS RecordCount
FROM SIAKAD.dbo.enrollment
UNION ALL
SELECT
    'Warehouse' AS DataSource,
    COUNT(*) AS RecordCount
FROM dbo.Fact_Enrollment;
```

2. **Create Data Quality Dashboard**

   - Tabel audit untuk tracking data quality metrics

- Stored procedure untuk generate quality report
- Alert jika quality threshold tidak terpenuhi

Output:

- Data Quality Report (PDF/Excel)
- File SQL: `08_Data_Quality_Checks.sql`
- Quality metrics summary

4.5.7 Step 7: Performance Testing

Tujuan: Menguji dan mengoptimalkan performa query

Aktivitas:

1. **Create Test Queries**

```sql
-- Query 1: Total students per program
SET STATISTICS TIME ON;
SET STATISTICS IO ON;

SELECT
    p.ProgramName,
    COUNT(DISTINCT f.StudentKey) AS TotalStudents,
    AVG(f.Grade) AS AvgGrade,
    SUM(f.Credits) AS TotalCredits
FROM dbo.Fact_Enrollment f
INNER JOIN dbo.Dim_Program p ON f.ProgramKey = p.ProgramKey
INNER JOIN dbo.Dim_Date d ON f.DateKey = d.DateKey
WHERE d.AcademicYear = '2024/2025'
GROUP BY p.ProgramName
ORDER BY TotalStudents DESC;

-- Query 2: Monthly enrollment trend
SELECT
    d.Year,
    d.MonthNumber,
    d.MonthName,
    COUNT(DISTINCT f.EnrollmentKey) AS TotalEnrollments,
    SUM(f.TuitionFee) AS TotalRevenue
FROM dbo.Fact_Enrollment f
INNER JOIN dbo.Dim_Date d ON f.DateKey = d.DateKey
GROUP BY d.Year, d.MonthNumber, d.MonthName
ORDER BY d.Year, d.MonthNumber;
```

2. **Analyze Query Plans**

    - Review execution plans
    - Identify missing indexes
    - Check for table scans

Query Type	**Target**	**Actual**	**Status**
Simple Aggregation	<1s	0.5s	Pass
Complex Join	<3s	2.1s	Pass
Drill-down Analysis	<2s	1.8s	Pass
Full Scan Report	<10s	8.5s	Pass

- Optimize join orders

3. **Performance Benchmarks**

Output:

- Performance Test Report
- Query optimization recommendations
- File SQL: `09_Test_Queries.sql`

4.6 Kriteria Penilaian Misi 2

Aspek	**Kriteria**	**Bobot**
Physical Design	Kualitas SQL DDL, proper constraints, naming conventions, database organization	20%
Indexing & Optimization	Strategi indexing yang tepat, partitioning (if applicable), query optimization	20%
ETL Implementation	Functionality, robustness, error handling, efficiency, completeness	30%
Data Quality	Completeness, accuracy, consistency, quality checks implementation	15%
Performance	Query response time, ETL execution time, resource utilization	10%
Documentation	Code documentation, GitHub repository, technical documentation	5%
Total		**100%**

5 MISI 3: IMPLEMENTASI PRODUKSI

5.1 Deskripsi Misi

Misi 3 fokus pada deployment data mart ke lingkungan produksi, pembangunan laporan/dashboard, implementasi keamanan, dan operasionalisasi sistem.

5.2 Tujuan Misi

- Men-deploy data mart ke production environment
- Membangun dashboard dan reporting untuk end-users
- Mengimplementasikan security dan access control
- Menyiapkan monitoring dan maintenance procedures
- Melakukan user acceptance testing (UAT)
- Menyusun dokumentasi operasional lengkap

5.3 Deliverables

1. Production Database di Azure VM
2. BI Dashboard/Reports (Power BI atau Tableau)
3. Security Implementation Documentation
4. User Manual
5. Operations Manual
6. Final Presentation
7. Complete GitHub Repository

5.4 Timeline

Durasi: 10 Hari

Minggu	Aktivitas	Output
10	Production deployment	Working production database
11	Dashboard development	Interactive dashboards
12	Security implementation	Access control, audit trail
13	UAT dan refinement	Tested and refined system
14	Documentation & presentation	Final deliverables

5.5 Langkah Pengerjaan Detail

5.5.1 Step 1: Production Deployment

Tujuan: Men-deploy database dan ETL ke Azure VM production
Aktivitas:

1. **Environment Setup**
    - Provision Azure VM dengan SQL Server
    - Configure firewall rules
    - Setup SQL Server services
    - Configure backup locations
2. **Database Deployment**

```
-- Deploy using SSMS or sqlcmd
-- Execute all SQL scripts in sequence:
:r 01_Create_Database.sql
:r 02_Create_Dimensions.sql
:r 03_Create_Facts.sql
:r 04_Create_Indexes.sql
:r 05_Create_Partitions.sql
:r 06_Create_Staging.sql
:r 07_ETL_Procedures.sql
:r 08_Data_Quality_Checks.sql
```

3. **Initial Data Load**
    - Execute full ETL untuk historical data
    - Verify data integrity
    - Document load statistics
4. **Schedule ETL Jobs**

```
-- Create SQL Server Agent Job
USE msdb;
GO

EXEC sp_add_job
    @job_name = N'ETL_Daily_Load',
    @enabled = 1,
    @description = N'Daily ETL load for Data Mart';
GO

EXEC sp_add_jobstep
    @job_name = N'ETL_Daily_Load',
    @step_name = N'Execute Master ETL',
    @subsystem = N'TSQL',
    @command = N'EXEC dbo.usp_Master_ETL;',
    @database_name = N'DM_[UnitName]_DW',
    @retry_attempts = 3,
    @retry_interval = 5;
GO

EXEC sp_add_schedule
    @schedule_name = N'Daily at 2 AM',
    @freq_type = 4, -- Daily
    @freq_interval = 1,
    @active_start_time = 020000; -- 02:00 AM
GO

EXEC sp_attach_schedule
    @job_name = N'ETL_Daily_Load',
    @schedule_name = N'Daily at 2 AM';
GO

EXEC sp_add_jobserver
    @job_name = N'ETL_Daily_Load';
GO
```

Output:

- Deployment checklist
- Production connection strings
- ETL schedule documentation

5.5.2 Step 2: Dashboard Development

Tujuan: Membangun interactive dashboard untuk end-users

Aktivitas:

1. **Create Analytical Views**

```
-- View: Student Performance Summary
CREATE VIEW dbo.vw_Student_Performance AS
SELECT
    ds.NIM,
    ds.StudentName,
    ds.ProgramName,
    ds.Faculty,
    ds.EntryYear,
    COUNT(DISTINCT f.EnrollmentKey) AS TotalCourses,
    AVG(f.Grade) AS GPA,
    SUM(f.Credits) AS TotalCredits,
    SUM(CASE WHEN f.IsPassed = 1 THEN f.Credits ELSE 0 END) AS PassedCredits,
    CAST(SUM(CASE WHEN f.IsPassed = 1 THEN f.Credits ELSE 0 END) * 100.0
         / NULLIF(SUM(f.Credits), 0) AS DECIMAL(5,2)) AS PassRate
FROM dbo.Fact_Enrollment f
INNER JOIN dbo.Dim_Student ds ON f.StudentKey = ds.StudentKey
```

```sql
WHERE ds.IsCurrent = 1
GROUP BY ds.NIM, ds.StudentName, ds.ProgramName, ds.Faculty, ds.EntryYear;
GO

-- View: Program Analytics
CREATE VIEW dbo.vw_Program_Analytics AS
SELECT
    dp.ProgramName,
    dp.Faculty,
    d.AcademicYear,
    d.Semester,
    COUNT(DISTINCT f.StudentKey) AS TotalStudents,
    COUNT(DISTINCT f.EnrollmentKey) AS TotalEnrollments,
    AVG(f.Grade) AS AvgGrade,
    SUM(f.TuitionFee) AS TotalRevenue,
    CAST(SUM(CASE WHEN f.IsPassed = 1 THEN 1 ELSE 0 END) * 100.0
         / COUNT(*) AS DECIMAL(5,2)) AS PassRate
FROM dbo.Fact_Enrollment f
INNER JOIN dbo.Dim_Program dp ON f.ProgramKey = dp.ProgramKey
INNER JOIN dbo.Dim_Date d ON f.DateKey = d.DateKey
GROUP BY dp.ProgramName, dp.Faculty, d.AcademicYear, d.Semester;
GO
```

2. **Design Power BI Dashboards**

Dashboard 1: Executive Summary

- KPI Cards: Total Students, Avg GPA, Total Revenue
- Line Chart: Enrollment trend over time
- Bar Chart: Students per program
- Map: Student distribution by region

Dashboard 2: Academic Performance

- Stacked Bar: Pass rate by program
- Heat Map: Grade distribution
- Table: Top performing students
- Gauge: Graduation rate vs target

Dashboard 3: Financial Analysis

- Area Chart: Revenue trend
- Pie Chart: Revenue by program
- Waterfall: Revenue breakdown
- Forecast: Revenue projection

Connect Power BI to SQL Server

- Get Data → SQL Server

- Server: [Azure VM IP/Hostname]
- Database: DM_[UnitName]_DW
- Import Mode vs DirectQuery (recommended: DirectQuery untuk real-time)
- Load tables, views, atau write custom DAX queries

3. **Implement Interactivity**
   - Slicers: Academic Year, Semester, Program, Faculty
   - Drill-through: From summary to detail
   - Cross-filtering between visuals
   - Bookmarks untuk different views
   - Row-level security (if needed)

Output:

- Power BI file (.pbix)
- Published dashboard to Power BI Service (optional)
- Dashboard screenshots untuk dokumentasi

5.5.3 Step 3: Security Implementation

Tujuan: Mengimplementasikan access control dan audit trail

Aktivitas:

1. **Create User Roles**

```sql
-- Create Database Roles
CREATE ROLE db_executive;
CREATE ROLE db_analyst;
CREATE ROLE db_viewer;
CREATE ROLE db_etl_operator;
GO

-- Grant Permissions for Executive
GRANT SELECT ON SCHEMA::dbo TO db_executive;
GRANT EXECUTE ON dbo.usp_Master_ETL TO db_executive;
GO

-- Grant Permissions for Analyst
GRANT SELECT ON SCHEMA::dbo TO db_analyst;
GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::stg TO
   db_analyst;
GO

-- Grant Permissions for Viewer (Read-only)
GRANT SELECT ON dbo.vw_Student_Performance TO db_viewer;
GRANT SELECT ON dbo.vw_Program_Analytics TO db_viewer;
GRANT SELECT ON dbo.Dim_Date TO db_viewer;
GO
```

```sql

-- Grant Permissions for ETL Operator
GRANT EXECUTE ON SCHEMA::dbo TO db_etl_operator;
GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::stg TO db_etl_operator;
GRANT INSERT ON SCHEMA::dbo TO db_etl_operator;
GO
```

2. Create Users and Assign Roles

```sql
-- Create SQL Logins
CREATE LOGIN executive_user WITH PASSWORD = 'StrongP@ssw0rd!';
CREATE LOGIN analyst_user WITH PASSWORD = 'StrongP@ssw0rd!';
CREATE LOGIN viewer_user WITH PASSWORD = 'StrongP@ssw0rd!';
CREATE LOGIN etl_service WITH PASSWORD = 'StrongP@ssw0rd!';
GO

-- Create Database Users
USE DM_[UnitName]_DW;
GO

CREATE USER executive_user FOR LOGIN executive_user;
CREATE USER analyst_user FOR LOGIN analyst_user;
CREATE USER viewer_user FOR LOGIN viewer_user;
CREATE USER etl_service FOR LOGIN etl_service;
GO

-- Assign to Roles
ALTER ROLE db_executive ADD MEMBER executive_user;
ALTER ROLE db_analyst ADD MEMBER analyst_user;
ALTER ROLE db_viewer ADD MEMBER viewer_user;
ALTER ROLE db_etl_operator ADD MEMBER etl_service;
GO
```

3. Implement Data Masking

```sql
-- Dynamic Data Masking untuk kolom sensitif
ALTER TABLE dbo.Dim_Student
ALTER COLUMN Email ADD MASKED WITH (FUNCTION = 'email()');

ALTER TABLE dbo.Dim_Student
ALTER COLUMN Phone ADD MASKED WITH (FUNCTION = 'partial(0,"XXX-XXX-",4)');

-- Grant UNMASK permission untuk specific roles
GRANT UNMASK TO db_executive;
GRANT UNMASK TO db_analyst;
GO
```

4. Implement Audit Trail

```sql
-- Create Audit Table
CREATE TABLE dbo.AuditLog (
    AuditID BIGINT IDENTITY(1,1) PRIMARY KEY,
    EventTime DATETIME2 DEFAULT SYSDATETIME(),
    UserName NVARCHAR(128) DEFAULT SUSER_SNAME(),
    EventType NVARCHAR(50), -- SELECT, INSERT, UPDATE, DELETE
    SchemaName NVARCHAR(128),
    ObjectName NVARCHAR(128),
    SQLStatement NVARCHAR(MAX),
    RowsAffected INT,
    IPAddress VARCHAR(50),
    ApplicationName NVARCHAR(128) DEFAULT APP_NAME()
);
GO

-- Create Audit Trigger (example for Fact_Enrollment)
CREATE TRIGGER trg_Audit_Fact_Enrollment
ON dbo.Fact_Enrollment
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @EventType NVARCHAR(50);
    DECLARE @RowsAffected INT;

    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';

    SET @RowsAffected = @@ROWCOUNT;

    INSERT INTO dbo.AuditLog (EventType, SchemaName, ObjectName, RowsAffected)
    VALUES (@EventType, 'dbo', 'Fact_Enrollment', @RowsAffected);
END;
GO

-- Enable SQL Server Audit (Server-level)
CREATE SERVER AUDIT DataWarehouse_Audit
TO FILE
(   FILEPATH = N'D:\Audit\',
    MAXSIZE = 100 MB,
    MAX_ROLLOVER_FILES = 10
)
WITH (ON_FAILURE = CONTINUE);
```

```sql
GO

ALTER SERVER AUDIT DataWarehouse_Audit WITH (STATE = ON);
GO

-- Create Database Audit Specification
CREATE DATABASE AUDIT SPECIFICATION DataWarehouse_DB_Audit
FOR SERVER AUDIT DataWarehouse_Audit
ADD (SELECT, INSERT, UPDATE, DELETE ON SCHEMA::dbo BY public);
GO

ALTER DATABASE AUDIT SPECIFICATION DataWarehouse_DB_Audit WITH (
    STATE = ON);
GO
```

Output:

- Security implementation document
- User access matrix
- File SQL: `10_Security.sql`

5.5.4 Step 4: Backup and Recovery Strategy

Tujuan: Menyiapkan backup untuk disaster recovery

Aktivitas:

```sql
-- Full Backup
BACKUP DATABASE DM_[UnitName]_DW
TO DISK = N'D:\Backup\DM_[UnitName]_DW_Full.bak'
WITH
    COMPRESSION,
    INIT,
    NAME = N'Full Database Backup',
    STATS = 10;
GO

-- Differential Backup
BACKUP DATABASE DM_[UnitName]_DW
TO DISK = N'D:\Backup\DM_[UnitName]_DW_Diff.bak'
WITH
    DIFFERENTIAL,
    COMPRESSION,
    INIT,
    NAME = N'Differential Database Backup',
    STATS = 10;
GO

-- Transaction Log Backup
BACKUP LOG DM_[UnitName]_DW
TO DISK = N'D:\Backup\DM_[UnitName]_DW_Log.trn'
```

```sql
WITH
    COMPRESSION,
    INIT,
    NAME = N'Transaction Log Backup',
    STATS = 10;
GO

-- Schedule Backup Jobs
-- Full Backup: Weekly (Sunday 2 AM)
-- Differential Backup: Daily (2 AM)
-- Transaction Log Backup: Every 6 hours

-- Backup to Azure Blob Storage (Optional)
CREATE CREDENTIAL [AzureStorageCredential]
WITH IDENTITY = 'SHARED ACCESS SIGNATURE',
SECRET = '<SAS_TOKEN>';
GO

BACKUP DATABASE DM_[UnitName]_DW
TO URL = N'https://[storage_account].blob.core.windows.net/backups/DM_[UnitName]_DW.bak'
WITH CREDENTIAL = 'AzureStorageCredential',
COMPRESSION;
GO
```

Output:

- Backup schedule documentation
- Recovery procedures document
- File SQL: `11_Backup.sql`

5.5.5 Step 5: User Acceptance Testing

Tujuan: Validasi sistem dengan end-users

Aktivitas:

1. **Create Test Cases**

Test ID	Scenario	Expected Result	Status	Notes
TC001	View total students per program	Dashboard shows correct count	Pass	
TC002	Filter by academic year	Data filtered correctly	Pass	
TC003	Export report to Excel	File downloads successfully	Pass	
TC004	Drill-down from program to student	Detail view appears	Pass	
TC005	ETL runs successfully	Data updated in warehouse	Pass	
...	...	...	...	...

2. **Conduct UAT Sessions**
   - Schedule meeting dengan stakeholders
   - Demonstrasi dashboard dan features
   - Collect feedback dan bug reports

- Document change requests

3. **Performance Testing**
   - Test dengan concurrent users
   - Measure dashboard load time
   - Test ETL with production data volume
   - Identify and fix bottlenecks
4. **Refinement**
   - Fix bugs dari UAT
   - Implement requested features
   - Optimize slow queries
   - Update documentation

Output:

- UAT Test Results document
- Bug report dan resolution log
- Change request log

5.5.6 Step 6: Documentation

Tujuan: Menyiapkan dokumentasi lengkap untuk operasional

Dokumen yang Diperlukan:

1. **System Architecture Document**
   - High-level architecture diagram
   - Technology stack
   - Data flow diagram
   - Deployment architecture
2. **Data Dictionary** (Update dari Misi 1)
   - Complete table and column definitions
   - Business rules
   - Data lineage
3. **ETL Documentation**
   - ETL process flow
   - Transformation rules
   - Error handling procedures
   - Schedule and dependencies

4. **User Manual**
    - How to access dashboard
    - How to navigate reports
    - How to apply filters
    - How to export data
    - FAQ section
5. **Operations Manual**
    - Daily operations checklist
    - How to monitor ETL jobs
    - How to troubleshoot common issues
    - Backup and recovery procedures
    - Contact information for support
6. **Security Documentation**
    - User roles and permissions
    - Access request procedures
    - Password policies
    - Audit trail queries

Output:

- Complete documentation package (PDF)
- Updated GitHub README.md dengan links ke semua dokumentasi
- Quick reference guides

5.5.7 Step 7: Final Presentation

Tujuan: Mempresentasikan hasil proyek kepada dosen dan stakeholders
Struktur Presentasi (20-25 menit):

1. **Introduction** (2 menit)
    - Team introduction
    - Business domain overview
    - Project objectives
2. **Requirements & Design** (3 menit)
    - Business requirements summary
    - Key KPIs identified
    - Dimensional model overview
    - Architecture choice (Kimball/Inmon/Vault)

3. **Implementation** (5 menit)
    - Database schema
    - ETL process
    - Data volume and quality
    - Challenges and solutions
4. **Dashboard Demo** (8 menit)
    - Live demonstration of dashboards
    - Show key features and interactivity
    - Demonstrate different user scenarios
    - Show performance metrics
5. **Technical Highlights** (3 menit)
    - Optimization techniques used
    - Security implementation
    - Backup strategy
    - Monitoring and maintenance
6. **Lessons Learned & Future Work** (2 menit)
    - Key takeaways
    - Challenges overcome
    - Potential improvements
    - Scalability considerations
7. **Q&A** (5 menit)

Presentasi Requirements:

- PowerPoint slides (max 20 slides)
- Live demo (tidak boleh video recording)
- Semua anggota tim harus presentasi
- Siapkan backup plan jika koneksi bermasalah

Output:

- Presentation slides (PDF)
- Demo script
- Q&A preparation document

5.6 Kriteria Penilaian Misi 3

Aspek	**Kriteria**	**Bobot**
Production Deployment	Successfully deployed, operational, stable, proper configuration	15%
Dashboard Quality	Visual appeal, usability, interactivity, insights provided, performance	25%
Security Implementation	Role-based access control, data masking, audit trail, compliance	15%
User Acceptance	UAT results, user feedback, bug fixes, feature completeness	15%
Documentation	Completeness, clarity, usability, organization, professionalism	15%
Presentation	Content, delivery, demo quality, Q&A responses, team coordination	15%
Total		**100%**

6 PEDOMAN UMUM

6.1 GitHub Best Practices

1. **Commit Messages**

```
# Good commit messages:
git commit -m "Add: Fact_Enrollment table with indexing"
git commit -m "Fix: ETL procedure null handling issue"
git commit -m "Update: Data dictionary with new columns"
git commit -m "Docs: Add UAT test cases document"

# Bad commit messages:
git commit -m "update"
git commit -m "fix bug"
git commit -m "changes"
```

2. **Branch Strategy**

   - `main`: Production-ready code
   - `develop`: Integration branch
   - `feature/[name]`: Feature development
   - `hotfix/[name]`: Emergency fixes

3. **Pull Requests**

   - Always create PR untuk merge ke main
   - Minimal 1 reviewer before merge
   - Include description of changes
   - Link related issues

4. **.gitignore**

```
# SQL Server files
*.mdf
*.ldf
*.bak

# SSIS files
*.dtproj.user
**/bin/
**/obj/

# Power BI
*.pbix.tmp

# Credentials
**/config/*.json
**/secrets/

```

```
# Logs
*.log
*.txt
```

6.2 Naming Conventions

6.2.1 Database Objects

Object Type	Prefix	Example
Fact Table	Fact_	Fact_Enrollment
Dimension Table	Dim_	Dim_Student
Staging Table	stg.	stg.Student
View	vw_	vw_Student_Performance
Stored Procedure	usp_	usp_Load_Dim_Student
Function	fn_	fn_Calculate_GPA
Index (Non-Clustered)	IX_	IX_Fact_Enrollment_Student
Index (Clustered)	CIX_	CIX_Fact_Enrollment_DateKey
Primary Key	PK_	PK_Dim_Student
Foreign Key	FK_	FK_Fact_Enrollment_Student

6.2.2 Column Names

- Use PascalCase: `StudentName`, `EnrollmentDate`
- Surrogate keys: `[TableName]Key` (e.g., `StudentKey`)
- Foreign keys: Same as referenced column name
- Boolean: Prefix dengan `Is`, `Has`, `Can`
- Dates: Suffix dengan `Date` atau `Time`

6.3 Code Review Checklist

☐ Code follows naming conventions

☐ SQL scripts are idempotent (can run multiple times)

☐ Proper error handling implemented

☐ No hardcoded values (use variables/parameters)

☐ Comments for complex logic

☐ No SQL injection vulnerabilities

☐ Indexes are appropriate and not excessive

☐ Transactions are properly managed

☐ Test data is not committed to repository

6.4 Common Pitfalls to Avoid

1. **Over-Engineering**
    - Jangan membuat desain terlalu kompleks
    - Start simple, iterate based on needs
    - Remember: YAGNI (You Aren't Gonna Need It)
2. **Ignoring Data Quality**
    - Selalu validate data sebelum load
    - Implement proper error handling
    - Document data quality issues
3. **Poor Documentation**
    - Document as you go, not at the end
    - Keep documentation in sync with code
    - Use clear and concise language
4. **Neglecting Performance**
    - Test with realistic data volumes
    - Monitor query execution plans
    - Optimize before deploying to production
5. **Security Afterthought**
    - Implement security from the start
    - Never store passwords in plain text
    - Follow principle of least privilege

6.5 Time Management Tips

1. **Weekly Team Meetings**
    - Schedule fixed meeting time
    - Review progress and blockers
    - Assign tasks for next week
    - Document decisions
2. **Use Project Management Tools**
    - GitHub Projects/Issues
    - Trello, Asana, atau Notion
    - Track progress visually
    - Set deadlines for milestones

3. **Parallel Workstreams**
   - Divide tasks among team members
   - Some can work on ETL while others on dashboard
   - Regular integration points
   - Clear interfaces between components
4. **Buffer Time**
   - Always add 20-30% buffer to estimates
   - Account for unexpected issues
   - Leave time for testing and refinement

7 TEMPLATE DELIVERABLES

7.1 README.md Template (GitHub)

```
# Data Mart [Nama Unit] - Institut Teknologi Sumatera

## Project Overview
Brief description of the data mart, its purpose, and business value.

## Team Members
| NIM | Name | Role | Email |
|-----|-------|------|-------|
| 123450xxx | Name 1 | Project Lead & Database Designer | email@example.com |
| 123450xxx | Name 2 | ETL Developer | email@example.com |
| 123450xxx | Name 3 | BI Developer | email@example.com |
| 123450xxx | Name 4 | Documentation & QA | email@example.com |

## Business Domain
Description of the unit/department being analyzed:
- Key business processes
- Stakeholders
- Decision-making needs

## Objectives
1. Objective 1
2. Objective 2
3. Objective 3

## Key Performance Indicators (KPIs)
- KPI 1: Description
- KPI 2: Description
- KPI 3: Description

## Architecture
- **Approach**: Kimball Dimensional Modeling
- **Database**: SQL Server 2019 on Azure VM
- **ETL**: SQL Server Integration Services (SSIS)
- **Visualization**: Power BI Desktop
- **Version Control**: GitHub

### Data Model
![Dimensional Model](docs/02-design/dimensional-model.png)

**Fact Tables:**
- Fact_[Name]: Description

**Dimension Tables:**
- Dim_Date: Time dimension
- Dim_[Name]: Description
- ...
```

~~~

Repository Structure

```
 README.md
 docs/
    01-requirements/
    02-design/
    03-implementation/
    presentations/
 sql/
    01_Create_Database.sql
    02_Create_Dimensions.sql
    03_Create_Facts.sql
    ...
 etl/
    packages/
    scripts/
 dashboards/
    PowerBI files
 tests/
     test scripts
```

Getting Started

Prerequisites
- SQL Server 2019 or higher
- SQL Server Management Studio (SSMS)
- Power BI Desktop
- Access to Azure VM (provided by instructor)

Installation
1. Clone repository:
   ```bash
   git clone https://github.com/[org]/[repo].git
   ```

2. Execute SQL scripts in order:
   ```bash
   sqlcmd -S [server] -d [database] -i sql/01_Create_Database.sql
   sqlcmd -S [server] -d [database] -i sql/02_Create_Dimensions.sql
   ...
   ```

3. Deploy SSIS packages or run ETL scripts

4. Open Power BI dashboard and refresh data

Dashboards
- **Executive Dashboard**: High-level KPIs and trends
~~~

```
- **Operational Dashboard**: Detailed analysis
- **Custom Reports**: Ad-hoc analysis capabilities

## Documentation
- [Business Requirements](docs/01-requirements/business-requirements.md)
- [Data Dictionary](docs/02-design/data-dictionary.xlsx)
- [ETL Documentation](docs/03-implementation/etl-documentation.md)
- [User Manual](docs/03-implementation/user-manual.pdf)
- [Operations Manual](docs/03-implementation/operations-manual.pdf)

## Testing
- Data Quality Tests: [Link to test results]
- Performance Tests: [Link to test results]
- UAT Results: [Link to UAT report]

## Project Timeline
- **Misi 1** (Weeks 1-4): Completed [Date]
- **Misi 2** (Weeks 5-9): Completed [Date]
- **Misi 3** (Weeks 10-14): Completed [Date]

## Security
- Role-based access control implemented
- Data masking for sensitive information
- Audit trail for all data modifications
- See [Security Documentation](docs/03-implementation/security.md)

## License
This project is developed as part of academic coursework at Institut Teknologi Sumatera.

## Acknowledgments
- Program Studi Sains Data, Fakultas Sains, ITERA
- Course Instructor: [Name]
- Stakeholders from [Unit Name]

## Contact
For questions or issues, please contact the team lead or create an issue in this repository.
```

7.2 Final Report Outline

1. **Cover Page**
   - Judul proyek
   - Logo ITERA
   - Team members
   - Tanggal
2. **Executive Summary** (1-2 halaman)

- Project overview
- Key achievements
- Business impact
- Recommendations

3. **Introduction**
    - Background
    - Objectives
    - Scope
    - Methodology
4. **Requirements Analysis**
    - Business requirements
    - Functional requirements
    - Non-functional requirements
    - Data sources
5. **Design**
    - Conceptual model (ERD)
    - Logical model (dimensional model)
    - Physical design
    - Architecture diagram
6. **Implementation**
    - Database implementation
    - ETL process
    - Dashboard development
    - Security implementation
7. **Testing & Validation**
    - Data quality results
    - Performance testing
    - UAT results
    - Bug fixes
8. **Deployment & Operations**
    - Production deployment
    - Backup strategy
    - Monitoring

  - Maintenance procedures

9. **Results & Discussion**
   - Key findings
   - Business insights
   - Performance metrics
   - Challenges and solutions

10. **Conclusions & Future Work**
    - Summary of achievements
    - Lessons learned
    - Limitations
    - Recommendations for future enhancements

11. **References**

12. **Appendices**
    - Complete data dictionary
    - SQL scripts (selected)
    - Dashboard screenshots
    - Test results

8 PENILAIAN DAN GRADING

8.1 Bobot Komponen Penilaian

Komponen	Bobot Misi	Bobot Total	Tenggat
Misi 1: Desain Konseptual & Logikal	30%	9%	Minggu 1
Misi 2: Desain Fisikal & Development	35%	10.5%	Minggu 2
Misi 3: Implementasi Produksi	35%	10.5%	Minggu 3
Total Tugas Besar	**100%**	**30%**	

8.2 Sistem Penilaian

- **Skala Nilai**: 0-100 untuk setiap misi
- **Passing Grade**: Minimal 56 untuk setiap misi
- **Late Submission Penalty**: -10% per hari keterlambatan (maksimal 3 hari)
- **Plagiarism**: Nilai 0 jika terdeteksi plagiarisme

8.3 Kriteria Penilaian Kelompok

Grade	Kriteria
A (86-100)	Exceptional work. Semua deliverables lengkap dan berkualitas tinggi. Implementasi production-ready dengan dokumentasi excellent. Dashboard sangat interaktif dan insightful.
B (71-85)	Good work. Deliverables lengkap dengan kualitas baik. Implementasi functional dengan dokumentasi yang memadai. Dashboard interaktif dan useful.
C (56-70)	Satisfactory work. Deliverables ada tetapi beberapa aspek kurang optimal. Implementasi bekerja dengan beberapa issues minor. Dashboard basic namun functional.
D (< 56)	Unsatisfactory work. Deliverables tidak lengkap atau kualitas rendah. Implementasi tidak bekerja dengan baik atau banyak bugs. Dashboard tidak memenuhi requirements.

8.4 Peer Evaluation

Setiap anggota kelompok akan mengevaluasi kontribusi rekan setimnya:

- **Factor**: 0.7 - 1.3

- **Kriteria**:
  - Kontribusi terhadap deliverables
  - Partisipasi dalam meeting
  - Komunikasi dan kolaborasi
  - Kualitas pekerjaan
  - Responsiveness
- **Final Individual Score** = Group Score × Peer Factor

8.5 Submission Requirements

8.5.1 Misi 1 Submission

1. GitHub Repository URL
2. Dokumen Requirements Analysis (PDF)
3. ERD (High-resolution image/PDF)
4. Dimensional Model Diagram
5. Data Dictionary (Excel/PDF)
6. Presentation Slides (PDF)

8.5.2 Misi 2 Submission

1. Updated GitHub Repository
2. Complete SQL Scripts (zipped)
3. SSIS Project files atau ETL scripts
4. Data Quality Report (PDF)
5. Performance Test Results (PDF)
6. Technical Documentation (PDF)

8.5.3 Misi 3 Submission

1. Production Database Credentials
2. Power BI Dashboard File (.pbix)
3. Published Dashboard URL (if applicable)
4. Complete Documentation Package (PDF)
   - User Manual
   - Operations Manual
   - Security Documentation

5. Final Report (PDF, 30-50 halaman)
6. Presentation Slides (PDF)
7. Complete GitHub Repository (final version)

8.6 Submission Method

- **Primary**: GitHub Repository (provide URL)
- **Secondary**: Upload ZIP file ke LMS ITERA
- **Naming Convention**:

```
Kelompok[X]_Misi[Y]_[UnitName]_[Date].zip

Example:
Kelompok01_Misi1_FakultasSains_20250215.zip
Kelompok01_Misi2_FakultasSains_20250322.zip
Kelompok01_Misi3_FakultasSains_20250426.zip
```

9 REFERENSI DAN SUMBER BELAJAR

9.1 Buku Referensi

1. Vaisman, A., & Zimányi, E. (2014). *Data warehouse systems: Design and implementation*. Springer.

2. Taniar, D., & Rahayu, W. (2022). *Data warehousing and analytics: Fueling the data engine*. Springer Nature.

3. Kimball, R., & Ross, M. (2013). *The data warehouse toolkit: The definitive guide to dimensional modeling* (3rd ed.). Wiley.

4. Inmon, W. H. (2005). *Building the data warehouse* (4th ed.). Wiley.

5. Linstedt, D., & Olschimke, M. (2015). *Building a scalable data warehouse with Data Vault 2.0*. Morgan Kaufmann.

9.2 Dokumentasi Teknis

1. Microsoft SQL Server Documentation
   - https://docs.microsoft.com/en-us/sql/
   - SQL Server Database Engine
   - Integration Services (SSIS)
   - Analysis Services (SSAS)

2. Microsoft Power BI Documentation
   - https://docs.microsoft.com/en-us/power-bi/
   - Power BI Desktop
   - DAX Reference
   - Best practices

3. Azure SQL Database
   - https://docs.microsoft.com/en-us/azure/sql-database/
   - Security best practices
   - Performance tuning

4. GitHub Documentation
   - https://docs.github.com/
   - Git basics
   - Collaboration workflow

9.3 Online Resources

1. **Kimball Group**: https://www.kimballgroup.com/
   - Design tips
   - Best practices
   - Case studies
2. **SQLServerCentral**: https://www.sqlservercentral.com/
   - Articles and tutorials
   - Scripts repository
   - Community forums
3. **SQLBI**: https://www.sqlbi.com/
   - Power BI best practices
   - DAX patterns
   - Data modeling
4. **Stack Overflow**: https://stackoverflow.com/
   - SQL Server tag
   - SSIS tag
   - Power BI tag

9.4 Video Tutorials

1. Microsoft Learn (https://learn.microsoft.com/)
   - SQL Server learning paths
   - Power BI learning paths
   - Azure fundamentals
2. LinkedIn Learning
   - Data warehouse design courses
   - SQL Server administration
   - Power BI essentials
3. YouTube Channels
   - Guy in a Cube (Power BI)
   - WiseOwlTutorials (SQL Server)
   - Pragmatic Works (SSIS, SSAS, SSRS)

10 FREQUENTLY ASKED QUESTIONS (FAQ)

10.1 General Questions

Q1: Berapa ukuran minimal data yang harus dimuat ke data warehouse?
A1: Minimal 10,000 records dalam fact table utama. Namun, lebih banyak data (50K+ records) akan memberikan hasil analisis yang lebih bermakna.

Q2: Apakah boleh menggunakan sample data atau synthetic data?
A2: Ya, jika data real tidak tersedia atau sensitif. Namun, data harus realistis dan mencerminkan karakteristik bisnis yang sesungguhnya.

Q3: Berapa jumlah minimal tabel fakta dan dimensi?
A3: Minimal 1 fact table dan 4 dimension tables (termasuk Dim_Date). Untuk proyek yang lebih kompleks, bisa lebih dari itu.

Q4: Apakah wajib menggunakan SSIS untuk ETL?
A4: Tidak wajib. Bisa menggunakan T-SQL stored procedures, Python scripts, atau Azure Data Factory. Yang penting proses ETL berfungsi dengan baik dan terdokumentasi.

10.2 Technical Questions

Q5: Bagaimana cara mendapatkan akses ke Azure VM?
A5: Koordinator kelompok akan menerima credentials dari instructor. Atau bisa menggunakan Azure for Students subscription.

Q6: Apakah boleh menggunakan database lokal (bukan Azure)?
A6: Ya, untuk development. Namun untuk Misi 3 (production), harus di-deploy ke Azure VM atau cloud platform.

Q7: Bagaimana mengatasi error ”Login failed for user” di SQL Server?
A7: Pastikan:

- SQL Server Authentication diaktifkan
- User sudah dibuat dengan password yang benar
- Firewall rules sudah di-configure
- Connection string sudah benar

Q8: Bagaimana cara meng-handle Slowly Changing Dimensions (SCD)?
A8: Untuk tugas ini, implementasi SCD Type 2 (recommended):

- Tambahkan kolom: EffectiveDate, ExpiryDate, IsCurrent
- Expire record lama, insert record baru saat ada perubahan
- Lihat contoh di Section Misi 2, Step 5

Q9: Power BI Desktop tidak bisa connect ke Azure SQL Server. Kenapa?
A9: Pastikan:

- IP address Anda sudah ditambahkan ke Azure Firewall
- Gunakan SQL Server authentication (bukan Windows auth)
- Port 1433 tidak diblok
- Connection string format: [servername].database.windows.net

10.3 Project Management Questions

Q10: Bagaimana membagi tugas antar anggota kelompok?
A10: Rekomendasi pembagian:

- Anggota 1: Requirements analysis, dimensional modeling, project lead
- Anggota 2: Database design, SQL scripts, indexing
- Anggota 3: ETL development, data quality
- Anggota 4: Dashboard development, documentation

Namun semua anggota harus terlibat di semua fase.

Q11: Bagaimana jika ada anggota kelompok yang tidak aktif?
A11: Dokumentasikan kontribusi setiap anggota. Pada peer evaluation, berikan score yang sesuai dengan kontribusi masing-masing. Konsultasi dengan instructor jika masalah serius.

Q12: Bolehkah konsultasi dengan dosen di luar jam kuliah?
A12: Ya. Jadwalkan office hours dengan instructor melalui email atau appointment system.

10.4 Submission Questions

Q13: Format dokumen yang harus disubmit apa saja?
A13:

- Dokumen: PDF (NOT Word)
- Diagram: PNG/JPG high-resolution atau PDF
- Spreadsheet: Excel (.xlsx)
- SQL Scripts: .sql files
- SSIS: .dtsx files
- Power BI: .pbix files
- GitHub: Repository URL

Q14: Apakah ada template untuk dokumentasi?
A14: Ya, template disediakan di Section 8 (Template Deliverables) dokumen ini.

Q15: Bagaimana cara submit jika file terlalu besar?
A15:

- Upload file besar (Power BI, SSIS) ke Google Drive/OneDrive
- Share link di README.md GitHub
- Submit ZIP file kecil dengan scripts dan dokumentasi
- Jangan commit file data (.mdf, .ldf) ke GitHub

11 LAMPIRAN

11.1 Lampiran A: Checklist Progress Tracking

11.1.1 Misi 1 Checklist

- □ Kick-off meeting dilaksanakan
- □ Business requirements terdokumentasi
- □ Data sources teridentifikasi
- □ KPIs terdefinisi
- □ ERD sudah dibuat
- □ Fact tables teridentifikasi
- □ Dimension tables teridentifikasi
- □ Grain terdefinisi
- □ Dimensional model digambar
- □ Data dictionary lengkap
- □ GitHub repository dibuat
- □ README.md terpenuhi
- □ Dokumentasi diupload
- □ Presentasi Misi 1 dilaksanakan

11.1.2 Misi 2 Checklist

- □ Database dibuat di SQL Server
- □ Dimension tables dibuat
- □ Fact tables dibuat
- □ Primary keys dan foreign keys defined
- □ Indexes dibuat
- □ Partitioning implemented (if applicable)
- □ Staging tables dibuat
- □ ETL design terdokumentasi
- □ ETL implemented (SSIS atau scripts)
- □ Data loaded successfully

□ Data quality checks dilakukan

□ Performance testing dilakukan

□ SQL scripts committed ke GitHub

□ Technical documentation lengkap

11.1.3 Misi 3 Checklist

□ Production database deployed

□ ETL jobs scheduled

□ Historical data loaded

□ Analytical views dibuat

□ Power BI connected ke database

□ Dashboard Executive dibuat

□ Dashboard Operational dibuat

□ Interactivity implemented

□ User roles dibuat

□ Permissions di-assign

□ Data masking implemented

□ Audit trail implemented

□ Backup strategy implemented

□ UAT conducted

□ Bugs fixed

□ User manual completed

□ Operations manual completed

□ Final report completed

□ Presentation slides ready

□ Final presentation dilaksanakan

Term	Definition
Data Mart	Subset dari data warehouse yang fokus pada area bisnis atau departemen tertentu
Data Warehouse	Central repository untuk integrated data dari berbagai sources
Dimensional Model	Data modeling technique untuk data warehouse menggunakan facts dan dimensions
ETL	Extract, Transform, Load - process untuk memindahkan data dari source ke warehouse
Fact Table	Table yang menyimpan measurements, metrics, atau facts dari business process
Dimension Table	Table yang menyimpan descriptive attributes untuk analyzing facts
Grain	Level of detail atau atomicity dari data dalam fact table
Star Schema	Dimensional model dengan fact table di center dan dimensions mengelilinginya
Snowflake Schema	Dimensional model dengan normalized dimensions
SCD (Slowly Changing Dimension)	Dimension yang berubah pelan over time
Surrogate Key	System-generated key (biasanya integer) untuk unique identification
Natural Key	Business key yang naturally identifies record
OLAP	Online Analytical Processing - technology untuk complex queries
OLTP	Online Transaction Processing - system untuk day-to-day operations
Measure	Numeric value dalam fact table yang bisa di-aggregate
Additive Measure	Measure yang bisa dijumlahkan across all dimensions
Semi-additive Measure	Measure yang bisa dijumlahkan across some dimensions (e.g., balance)
Non-additive Measure	Measure yang tidak bisa dijumlahkan (e.g., ratios, percentages)
Conformed Dimension	Dimension yang shared across multiple data marts
Degenerate Dimension	Dimension attribute yang stored dalam fact table

11.2 Lampiran B: Glossary

11.3 Lampiran C: Contoh Dimensional Models by Domain

11.3.1 Akademik

Fact Tables:

- Fact_Enrollment: Student course enrollments
- Fact_Academic_Performance: Grades and achievements
- Fact_Thesis: Thesis supervision and completion

Dimensions:

- Dim_Date, Dim_Student, Dim_Course, Dim_Instructor, Dim_Program, Dim_Department

11.3.2 Keuangan

Fact Tables:

- Fact_Transactions: Financial transactions
- Fact_Budget: Budget allocation and utilization
- Fact_Payment: Payment processing

Dimensions:

- Dim_Date, Dim_Account, Dim_Department, Dim_Vendor, Dim_PaymentMethod, Dim_BudgetCategory

11.3.3 Kepegawaian

Fact Tables:

- Fact_Headcount: Employee headcount snapshots
- Fact_Attendance: Employee attendance
- Fact_Performance: Performance reviews

Dimensions:

- Dim_Date, Dim_Employee, Dim_Department, Dim_Position, Dim_Location

11.3.4 Sarana dan Prasarana

Fact Tables:

- Fact_Asset_Utilization: Facility and equipment usage
- Fact_Maintenance: Maintenance activities
- Fact_Request: Service requests

Dimensions:

- Dim_Date, Dim_Asset, Dim_Location, Dim_AssetType, Dim_Requester, Dim_MaintenanceType

Dokumen ini merupakan panduan resmi untuk Tugas Besar Mata Kuliah Pergudangan Data
Program Studi Sains Data, Fakultas Sains
Institut Teknologi Sumatera
Tahun Ajaran 2025