LAPORAN TUGAS PERGUDANGAN DATA

Implementasi Produksi

Disusun oleh :
Kelompok 12

1. Arya Muda Siregar	**123450063**
2. Lisa Diani Amelia	**122450021**
3. Vira Putri Maharani	**122450129**
4. Zailani Satria	**123450111**

PROGRAM STUDI SAINS DATA
FAKULTAS SAINS
INSTITUT TEKNOLOGI SUMATERA
LAMPUNG SELATAN
2025

1. **Step 1: Production Deployment**

1.1 Environment Setup

- ❖ Provision Azure VM dengan SQL Server
- ❖ Configure firewall rules
- ❖ Setup SQL Server services
- ❖ Configure backup locations

1.2 Database Deployment

```
– Deploy using SSMS or sqlcmd
:r 01_Create_Database.sql
: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
```

1.3 Initial Data Load

- ❖ Execute full ETL untuk historical data
- ❖ Verify data integrity
- ❖ Document load statistics

1.4 Schedule ETL Jobs

```
USE msdb;
GO

-----------------------------------------------------
-- 1. CREATE JOB jika belum ada
-----------------------------------------------------
IF NOT EXISTS (SELECT 1 FROM msdb.dbo.sysjobs WHERE name =
'ETL_K3L_Daily_Load')
BEGIN
    EXEC sp_add_job
        @job_name = N'ETL_K3L_Daily_Load',
        @enabled = 1,
        @description = N'Daily ETL load for K3L Data Mart';
END
GO

-----------------------------------------------------
-- 2. CREATE / UPDATE JOB STEP
-----------------------------------------------------
-- Hapus step jika sebelumnya sudah ada (agar tidak duplikat)
IF EXISTS (
    SELECT 1 FROM msdb.dbo.sysjobsteps
    WHERE step_name = 'Execute Master ETL'
      AND job_id = (SELECT job_id FROM msdb.dbo.sysjobs WHERE
name = 'ETL_K3L_Daily_Load')
)
BEGIN
```

```
    EXEC msdb.dbo.sp_delete_jobstep
      @job_name = 'ETL_K3L_Daily_Load',
      @step_name = 'Execute Master ETL';
END
GO

EXEC sp_add_jobstep
    @job_name = N'ETL_K3L_Daily_Load',
    @step_name = N'Execute Master ETL',
    @subsystem = N'TSQL',
    @command = N'EXEC dbo.usp_Master_ETL;',   -- GANTI jika SP
Anda berbeda
    @database_name = N'K3L_DataMart',         -- DB ANDA
    @retry_attempts = 3,
    @retry_interval = 5;
GO

-------------------------------------------------------
-- 3. CREATE SCHEDULE jika belum ada
-------------------------------------------------------
IF NOT EXISTS (SELECT 1 FROM msdb.dbo.sysschedules WHERE name =
'Daily at 2 AM')
BEGIN
    EXEC sp_add_schedule
        @schedule_name = N'Daily at 2 AM',
        @freq_type = 4,          -- Daily
        @freq_interval = 1,      -- Setiap hari
        @active_start_time = 020000;  -- 02:00 AM
END
GO

-------------------------------------------------------
-- 4. ATTACH SCHEDULE ke JOB
-------------------------------------------------------
BEGIN TRY
    EXEC sp_attach_schedule
        @job_name = N'ETL_K3L_Daily_Load',
        @schedule_name = N'Daily at 2 AM';
END TRY
BEGIN CATCH
    -- Jika sudah pernah ter-attach, abaikan error
END CATCH
GO

-------------------------------------------------------
-- 5. Registrasikan ke SQL Server Agent
-------------------------------------------------------
EXEC sp_add_jobserver
    @job_name = N'ETL_K3L_Daily_Load';
GO
```

2. Step 2: Dashboard Development

2.1 Create Analytical Views

```
USE K3L_DataMart;
GO

-------------------------------------------------------------
-- 1. VIEW: Ringkasan Insiden
-------------------------------------------------------------
IF OBJECT_ID('dbo.vw_Insiden_Summary') IS NOT NULL
    DROP VIEW dbo.vw_Insiden_Summary;
GO

CREATE VIEW dbo.vw_Insiden_Summary AS
SELECT
    d.Tahun,
    uk.NamaUnit,
    l.NamaLokasi,
    ji.NamaInsiden,
    s.TingkatKeparahan,
    COUNT(fi.InsidenID) AS TotalInsiden,
    SUM(fi.JumlahKorban) AS TotalKorban,
    SUM(fi.HariKerjaHilang) AS TotalHariKerjaHilang
FROM Fact_Insiden fi
JOIN Dim_Date d ON fi.DateID = d.DateID
JOIN Dim_Lokasi l ON fi.LokasiID = l.LokasiID
JOIN Dim_UnitKerja uk ON fi.UnitKerjaID = uk.UnitKerjaID
JOIN Dim_JenisInsiden ji ON fi.JenisInsidenID =
ji.JenisInsidenID
JOIN Dim_Severity s ON fi.SeverityID = s.SeverityID
GROUP BY
    d.Tahun, uk.NamaUnit, l.NamaLokasi, ji.NamaInsiden,
s.TingkatKeparahan;
GO


-------------------------------------------------------------
-- 2. VIEW: Ringkasan Inspeksi
-------------------------------------------------------------
IF OBJECT_ID('dbo.vw_Inspeksi_Summary') IS NOT NULL
    DROP VIEW dbo.vw_Inspeksi_Summary;
GO

CREATE VIEW dbo.vw_Inspeksi_Summary AS
SELECT
    d.Tahun,
    uk.NamaUnit,
    l.NamaLokasi,
    p.NamaPeralatan,
    SUM(fi.JumlahDiinspeksi) AS TotalDiinspeksi,
    SUM(fi.JumlahTidakSesuai) AS TotalTidakSesuai,
    CASE
        WHEN SUM(fi.JumlahDiinspeksi) = 0 THEN 0
        ELSE (SUM(fi.JumlahTidakSesuai) * 1.0 /
SUM(fi.JumlahDiinspeksi)) * 100
    END AS PersentaseKetidaksesuaian
FROM Fact_Inspeksi fi
JOIN Dim_Date d ON fi.DateID = d.DateID
```

```
JOIN Dim_Lokasi l ON fi.LokasiID = l.LokasiID
JOIN Dim_UnitKerja uk ON fi.UnitKerjaID = uk.UnitKerjaID
JOIN Dim_JenisPeralatan p ON fi.PeralatanID = p.PeralatanID
GROUP BY d.Tahun, uk.NamaUnit, l.NamaLokasi, p.NamaPeralatan;
GO


-------------------------------------------------------------
-- 3. VIEW: Ringkasan Limbah
-------------------------------------------------------------
IF OBJECT_ID('dbo.vw_Limbah_Summary') IS NOT NULL
    DROP VIEW dbo.vw_Limbah_Summary;
GO

CREATE VIEW dbo.vw_Limbah_Summary AS
SELECT
    d.Tahun,
    uk.NamaUnit,
    l.NamaLokasi,
    jl.JenisLimbah,
    jl.Kategori,
    SUM(fl.JumlahLimbah) AS TotalLimbah
FROM Fact_Limbah fl
JOIN Dim_Date d ON fl.DateID = d.DateID
JOIN Dim_Lokasi l ON fl.LokasiID = l.LokasiID
JOIN Dim_UnitKerja uk ON fl.UnitKerjaID = uk.UnitKerjaID
JOIN Dim_JenisLimbah jl ON fl.LimbahID = jl.LimbahID
GROUP BY d.Tahun, uk.NamaUnit, l.NamaLokasi, jl.JenisLimbah,
jl.Kategori;
GO


-------------------------------------------------------------
-- 4. VIEW: Executive Summary K3L
-------------------------------------------------------------
IF OBJECT_ID('dbo.vw_Executive_Summary_K3L') IS NOT NULL
    DROP VIEW dbo.vw_Executive_Summary_K3L;
GO

CREATE VIEW dbo.vw_Executive_Summary_K3L AS
SELECT
    d.Tahun,

    -- INSIDEN
    COUNT(DISTINCT fi.InsidenID) AS TotalInsiden,
    SUM(fi.JumlahKorban) AS TotalKorbanInsiden,
    SUM(fi.HariKerjaHilang) AS TotalHariKerjaHilang,

    -- INSPEKSI
    COUNT(DISTINCT insp.InspeksiID) AS TotalInspeksi,
    SUM(insp.JumlahTidakSesuai) AS TotalTemuanKetidaksesuaian,

    -- LIMBAH
    SUM(fl.JumlahLimbah) AS TotalLimbah,

    -- UNIT KERJA
    COUNT(DISTINCT uk.UnitKerjaID) AS TotalUnitKerja
```

```
FROM Dim_Date d
LEFT JOIN Fact_Insiden fi ON fi.DateID = d.DateID
LEFT JOIN Fact_Inspeksi insp ON insp.DateID = d.DateID
LEFT JOIN Fact_Limbah fl ON fl.DateID = d.DateID
LEFT JOIN Dim_UnitKerja uk ON uk.UnitKerjaID = fi.UnitKerjaID
GROUP BY d.Tahun;
GO
```

2.2 Design Power BI Dashboards

A. Dashboard 1: Executive Summary
   - ❖ KPI Cards:
     - Total Insiden
     - Total Harian Kerja
     - Total Korban Insiden
   - ❖ Line Chart : Total insiden selama 5 tahun
   - ❖ Bar Chart : Total korban berdasarkan unit
   - ❖ Donut Chart : Total insiden berdasarkan nama
   - ❖ Table : Tabel ringkasan kinerja K3L tahunan
   - ❖ Slicer : Tahun
   - ❖ Data Source : vw_Executive_Summary_K3L, vw_Insiden_Summary

B. Dashboard 2: Academic Performance
   - ❖ Stacked Bar : Tingkat kepatuhan per unit kerja
   - ❖ Matrix : Sebaran tingkat keparahan per unit kerja
   - ❖ Gauge : Capaian target kinerja K3L tahunan
   - ❖ Slicer : Tahun
   - ❖ Data Source : vw_Inspeksi_Summary, vw_Limbah_Summary, vw_Insiden_Summary

C. Dashboard 3: Financial Analysis
   - ❖ Area Chart : Tren total insiden
   - ❖ Pie Chart : Proyeksi insiden
   - ❖ Waterfall : Dampak insiden
   - ❖ Line Chart : Proporsi insiden per unit
   - ❖ Slicer : Tahun
   - ❖ Data Source : vw_Insiden_Summary

D. 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

2.3 Implement Interactivity

- ❖ Slicers: Tahun
- ❖ Drill-through: From summary to detail
- ❖ Cross-filtering between visuals
- ❖ Bookmarks
  - Executive Summary View
  - Academic Details
  - Financial Details
- ❖ Row-level security

3. Step 3: Security Implementation

3.1 Create User Roles

```
USE K3L_DataMart;
GO

/*============================================================
1.1. CREATE USER ROLES
============================================================*/

-- Create Roles
CREATE ROLE db_executive;
CREATE ROLE db_analyst;
CREATE ROLE db_viewer;
CREATE ROLE db_etl_operator;
GO

---

-- Executive Permissions (Full Read + ETL)

GRANT SELECT ON SCHEMA::dbo TO db_executive;
GRANT EXECUTE ON SCHEMA::dbo TO db_executive;
GO

---

-- Analyst Permissions (DW + staging jika ada)

GRANT SELECT ON SCHEMA::dbo TO db_analyst;
-- Jika ada schema staging, aktifkan:
-- GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::stg TO
db_analyst;
GO

---

-- Viewer Permissions (Read-only analytical views)

GRANT SELECT ON dbo.vw_Insiden_Summary        TO db_viewer;
GRANT SELECT ON dbo.vw_Inspeksi_Summary       TO db_viewer;
GRANT SELECT ON dbo.vw_Limbah_Summary         TO db_viewer;
GRANT SELECT ON dbo.vw_Executive_Summary_K3L  TO db_viewer;
GO

---
```

```
-- ETL Operator Permissions

GRANT EXECUTE ON SCHEMA::dbo TO db_etl_operator;
GRANT INSERT ON SCHEMA::dbo TO db_etl_operator;
GO
```

3.2 Create Users and Assign Roles

```
/*============================================================
1.2. CREATE USERS AND ASSIGN ROLES
============================================================*/
USE master;
GO

-- Create SQL Logins (Server-level)
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 K3L_DataMart;
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 Role Memberships
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
```

3.3 Implement Data Masking

```
/*============================================================
1.3 DATA MASKING IMPLEMENTATION
============================================================*/
-- Create Database Users
USE K3L_DataMart;
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 Role Memberships
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
```

3.4 Implement Audit Trail

```
/*============================================================
    1.4 AUDIT TRAIL IMPLEMENTATION
============================================================*/
USE K3L_DataMart;
GO
------------------------------------------------------------
-- 1. CREATE AUDIT TABLE
------------------------------------------------------------
IF OBJECT_ID('dbo.AuditLog', 'U') IS NULL
BEGIN
    CREATE TABLE dbo.AuditLog (
        AuditID BIGINT IDENTITY(1,1) PRIMARY KEY,
        EventTime DATETIME2 DEFAULT SYSDATETIME(),
        UserName NVARCHAR(128) DEFAULT SUSER_SNAME(),
        EventType NVARCHAR(50),
        SchemaName NVARCHAR(128),
        ObjectName NVARCHAR(128),
        SQLStatement NVARCHAR(MAX) NULL,
        RowsAffected INT,
        HostName NVARCHAR(128) NULL,
        IPAddress VARCHAR(50) NULL,
        ApplicationName NVARCHAR(128) DEFAULT APP_NAME()
    );
END
GO

------------------------------------------------------------
-- 2. TRIGGER: Fact_Insiden
------------------------------------------------------------
CREATE OR ALTER TRIGGER trg_Audit_Fact_Insiden
ON dbo.Fact_Insiden
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @EventType NVARCHAR(50);

    IF EXISTS (SELECT 1 FROM inserted) AND EXISTS (SELECT 1 FROM
deleted)
        SET @EventType = 'UPDATE';
    ELSE IF EXISTS (SELECT 1 FROM inserted)
        SET @EventType = 'INSERT';
    ELSE
        SET @EventType = 'DELETE';

    INSERT INTO dbo.AuditLog (EventType, SchemaName, ObjectName,
```

```
RowsAffected)
    VALUES (@EventType, 'dbo', 'Fact_Insiden', @@ROWCOUNT);
END;
GO

-------------------------------------------------------------
-- 3. TRIGGER: Fact_Inspeksi
-------------------------------------------------------------
CREATE OR ALTER TRIGGER trg_Audit_Fact_Inspeksi
ON dbo.Fact_Inspeksi
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @EventType NVARCHAR(50);

    IF EXISTS (SELECT 1 FROM inserted) AND EXISTS (SELECT 1 FROM
deleted)
        SET @EventType = 'UPDATE';
    ELSE IF EXISTS (SELECT 1 FROM inserted)
        SET @EventType = 'INSERT';
    ELSE
        SET @EventType = 'DELETE';

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

-------------------------------------------------------------
-- 4. TRIGGER: Fact_Limbah
-------------------------------------------------------------
CREATE OR ALTER TRIGGER trg_Audit_Fact_Limbah
ON dbo.Fact_Limbah
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @EventType NVARCHAR(50);

    IF EXISTS (SELECT 1 FROM inserted) AND EXISTS (SELECT 1 FROM
deleted)
        SET @EventType = 'UPDATE';
    ELSE IF EXISTS (SELECT 1 FROM inserted)
        SET @EventType = 'INSERT';
    ELSE
        SET @EventType = 'DELETE';

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

```
-------------------------------------------------------
-- 5. BUAT SERVER AUDIT
-------------------------------------------------------
USE master;
GO

CREATE SERVER AUDIT K3L_DataMart_Audit
TO FILE
(
    FILEPATH = N'C:\Audit',
    MAXSIZE = 100 MB,
    MAX_ROLLOVER_FILES = 10
)
WITH (ON_FAILURE = CONTINUE);
GO

-- Aktifkan server audit
ALTER SERVER AUDIT K3L_DataMart_Audit WITH (STATE = ON);
GO

-------------------------------------------------------
-- 6. BUAT DATABASE AUDIT SPECIFICATION
-------------------------------------------------------
USE K3L_DataMart;
GO

CREATE DATABASE AUDIT SPECIFICATION K3L_DB_Audit
FOR SERVER AUDIT K3L_DataMart_Audit
ADD (SELECT, INSERT, UPDATE, DELETE ON SCHEMA::dbo BY public);
GO

-- Aktifkan DB Audit
ALTER DATABASE AUDIT SPECIFICATION K3L_DB_Audit WITH (STATE =
ON);
GO
```

4. Step 4: Backup and Recovery Strategy

```
BACKUP DATABASE K3L_DataMart
TO DISK = N'C:\Backup\K3L_DataMart_Full.bak'
WITH
    COMPRESSION,
    INIT,
    NAME = N'Full Backup K3L_DataMart',
    STATS = 10;
GO
BACKUP DATABASE K3L_DataMart
TO DISK = N'C:\Backup\K3L_DataMart_Diff.bak'
WITH
    DIFFERENTIAL,
    COMPRESSION,
    INIT,
    NAME = N'Differential Backup K3L_DataMart',
    STATS = 10;
GO
```

```
BACKUP LOG K3L_DataMart
TO DISK = N'C:\Backup\K3L_DataMart_Log.trn'
WITH
    COMPRESSION,
    INIT,
    NAME = N'Transaction Log Backup K3L_DataMart',
    STATS = 10;
GO
CREATE CREDENTIAL AzureStorageCredential --(opsional)
WITH IDENTITY = 'SHARED ACCESS SIGNATURE',
SECRET = '<SAS_TOKEN>';
GO
BACKUP DATABASE K3L_DataMart
TO URL =
N'https://<storage_account>.blob.core.windows.net/backups/K3L_Dat
aMart_FULL.bak'
WITH
    CREDENTIAL = 'AzureStorageCredential',
    COMPRESSION,
    STATS = 10;
GO
```

5. Step 5: User Acceptance Testing

5.1 Create Test Cases

Test ID	Scenario	Expected Result	Status	Notes
TC001	View Total Insiden Tahunan.	Dashboard menunjukkan jumlah total insiden yang benar per tahun.	Pass	
TC002	Filter dashboard berdasarkan *Unit Kerja*.	Data Insiden, Inspeksi, dan Limbah hanya menampilkan data untuk unit yang dipilih.	Pass	
TC003	Drill-down dari *Total Korban Insiden* ke detail insiden.	Detail insiden (tanggal, lokasi, jenis) muncul dengan benar.	Pass	

TC004	Verify *Persentase Ketidaksesuaian* pada *vw_Inspeksi_Summary*.	Nilai Persentase Ketidaksesuaian terhitung dengan benar (persen).	Pass	
TC005	Jalankan ETL Job ETL_K3L_Daily_Load secara manual.	Data terbaru berhasil di-load ke Data Mart, dan *AuditLog* mencatat aktivitas INSERT/UPDATE.	Pass	

5.2 Conduct UAT Sessions

- Aktivitas: Mengadakan pertemuan dengan *stakeholders* (misalnya, Manajer K3L dan Analis Data).
- Tujuan: Mendemonstrasikan fungsionalitas dashboard, fitur interaktif, dan validasi data.
- Output: *Sign-off* dari *stakeholders* atas laporan dan fungsionalitas inti, serta daftar *Change Requests* atau *Bug Reports*.

5.3 Performance Testing

- Aktivitas:
  - Mengukur *dashboard load time* dengan volume data yang besar.
  - Menguji kinerja *Views* analitik (misalnya, menjalankan query pada vw_Executive_Summary_K3L dengan banyak filter).
  - Menguji durasi ETL dengan data produksi.
- Tujuan: Memastikan bahwa dashboard dapat melayani sejumlah pengguna bersamaan dengan waktu respons yang cepat (di bawah 5 detik).

5.4 Refinement

Aktivitas:

- Memperbaiki bug yang ditemukan selama UAT.
- Mengimplementasikan fitur kecil yang diminta oleh pengguna.
- Mengoptimalkan *slow queries* di balik *Views* atau *Stored Procedures* ETL.
- Memfinalisasi semua dokumentasi.

6. Step 6: Documentation

6.1 System Architecture Document

- High-level architecture diagram
- Technology stack
- Data flow diagram
- Deployment architecture

6.2 Data Dictionary (Update dari Misi 1)

- Complete table and column definitions
- Business rules
- Data lineage

6.3 ETL Documentation

- ETL process flow
- Transformation rules
- Error handling procedures
- Schedule and dependencies

6.4 User Manual

- How to access dashboard
- How to navigate reports
- How to apply filters
- How to export data
- FAQ section

6.5 Operations Manual

- Daily operations checklist
- How to monitor ETL jobs
- How to troubleshoot common issues
- Backup and recovery procedures
- Contact information for support

6.6 Security Documentation

- User roles and permissions
- Access request procedures
- Password policies
- Audit trail queries

ETL Schedule Documentation

Schedule ETL Jobs

```
USE msdb;
GO

-- 1. CREATE JOB
EXEC sp_add_job
   @job_name = N'ETL_K3L_Daily_Load', [cite: 42]
   @enabled = 1, [cite: 43]
   @description = N'Daily ETL load for K3L Data Mart'; [cite: 44]
GO

-- 2. ADD JOB STEP
EXEC sp_add_jobstep
   @job_name = N'ETL_K3L_Daily_Load', [cite: 59]
   @step_name = N'Execute Master ETL', [cite: 59]
   @subsystem = N'TSQL', [cite: 60]
   @command = N'EXEC dbo.usp_Master_ETL;', [cite: 61]
   @database_name = N'K3L_DataMart', [cite: 64]
   @retry_attempts = 3, [cite: 64]
   @retry_interval = 5; [cite: 65]
GO

-- 3. CREATE SCHEDULE
EXEC sp_add_schedule
   @schedule_name = N'Daily at 2 AM', [cite: 73]
   @freq_type = 4, -- Daily [cite: 74, 76]
   @freq_interval = 1, -- Every day [cite: 75, 77]
   @active_start_time = 020000; -- 02:00 AM [cite: 79, 78]
GO

-- 4. ATTACH SCHEDULE to JOB
EXEC sp_attach_schedule
   @job_name = N'ETL_K3L_Daily_Load', [cite: 86]
   @schedule_name = N'Daily at 2 AM'; [cite: 86]
GO

-- 5. Register to SQL Server Agent
EXEC sp_add_jobserver
   @job_name = N'ETL_K3L_Daily_Load'; [cite: 94]
GO
```

Penjelasan:

ETL Job Scheduling pada sistem Data Mart K3L diotomatisasi menggunakan SQL Server Agent. Sebuah *job* bernama ETL_K3L_Daily_Load dibuat dan diaktifkan sebagai proses utama yang menjalankan prosedur dbo.usp_Master_ETL untuk melakukan proses *extract, transform*, dan *load* data.

- Frekuensi: Job ini dijadwalkan berjalan setiap hari (*Daily*) pada pukul 02:00 AM. Waktu ini dipilih untuk menghindari jam sibuk operasional dan memastikan data terbaru tersedia setiap pagi.
- Mekanisme Ketahanan: Job dikonfigurasi dengan mekanisme *retry* sebanyak tiga kali

dengan interval lima menit untuk mengantisipasi kegagalan sementara, seperti *load lock* atau gangguan koneksi.

- ❖ Finalisasi: Jadwal tersebut dilekatkan (*attached*) ke job dan job kemudian diregistrasikan ke SQL Server Agent untuk memastikan eksekusi yang stabil dan terukur.

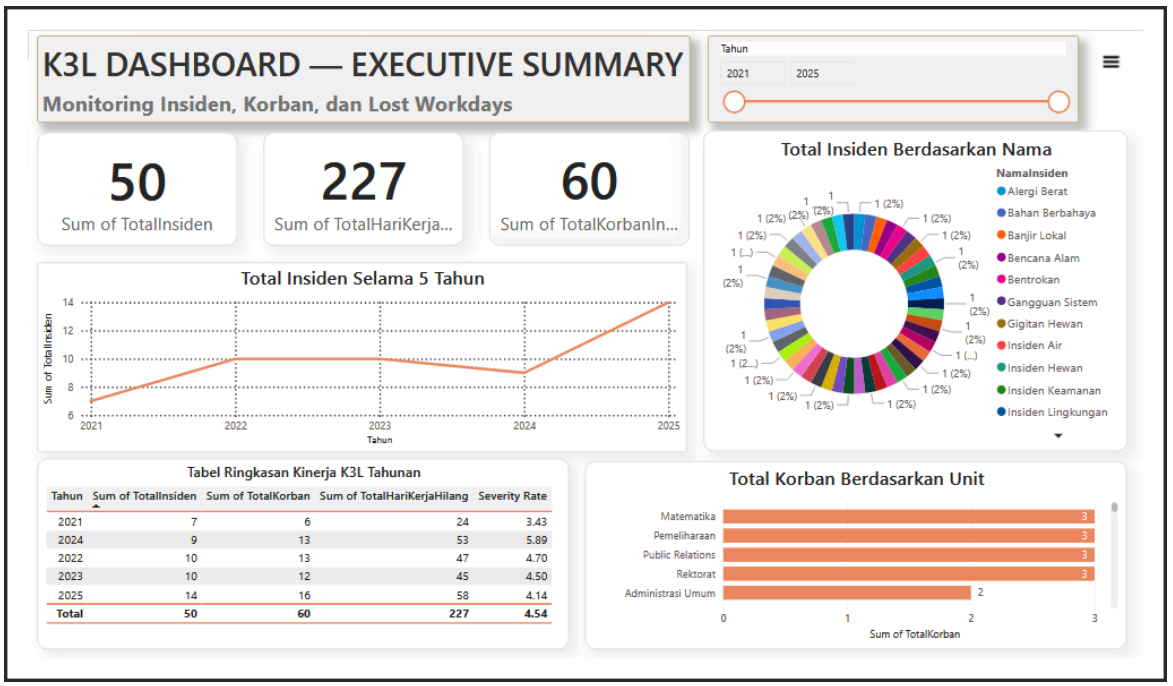
Script Deploy Job

```
USE msdb;
GO

-- 1. CREATE JOB jika belum ada
IF NOT EXISTS (SELECT 1 FROM msdb.dbo.sysjobs WHERE name =
'ETL_K3L_Daily_Load') [cite: 38, 39]
BEGIN
   EXEC sp_add_job
      @job_name = N'ETL_K3L_Daily_Load',
      @enabled = 1,
      @description = N'Daily ETL load for K3L Data Mart'; [cite: 42, 43, 44]
END
GO

-- 2. CREATE / UPDATE JOB STEP (Menggunakan sp_add_jobstep untuk
membuat/mengupdate step)
EXEC sp_add_jobstep
   @job_name = N'ETL_K3L_Daily_Load', [cite: 59]
   @step_name = N'Execute Master ETL', [cite: 59]
   @subsystem = N'TSQL', [cite: 60]
   @command = N'EXEC dbo.usp_Master_ETL;', [cite: 61]
   @database_name = N'K3L_DataMart', [cite: 64]
   @retry_attempts = 3, [cite: 64]
   @retry_interval = 5; [cite: 65]
GO

-- 3. CREATE SCHEDULE jika belum ada
IF NOT EXISTS (SELECT 1 FROM msdb.dbo.sysschedules WHERE name = 'Daily at 2
AM') [cite: 69, 70]
BEGIN
   EXEC sp_add_schedule
      @schedule_name = N'Daily at 2 AM', [cite: 73]
      @freq_type = 4, [cite: 74]
      @freq_interval = 1, [cite: 75]
      @active_start_time = 020000; [cite: 79]
END
GO

-- 4. ATTACH SCHEDULE ke JOB
EXEC sp_attach_schedule @job_name = N'ETL_K3L_Daily_Load', @schedule_name =
N'Daily at 2 AM'; [cite: 86]
```

```
GO

-- 5. Registrasikan ke SQL Server Agent
EXEC sp_add_jobserver @job_name = N'ETL_K3L_Daily_Load'; [cite: 94]
GO
```

Dashboard Documentation

A. Executive Summary



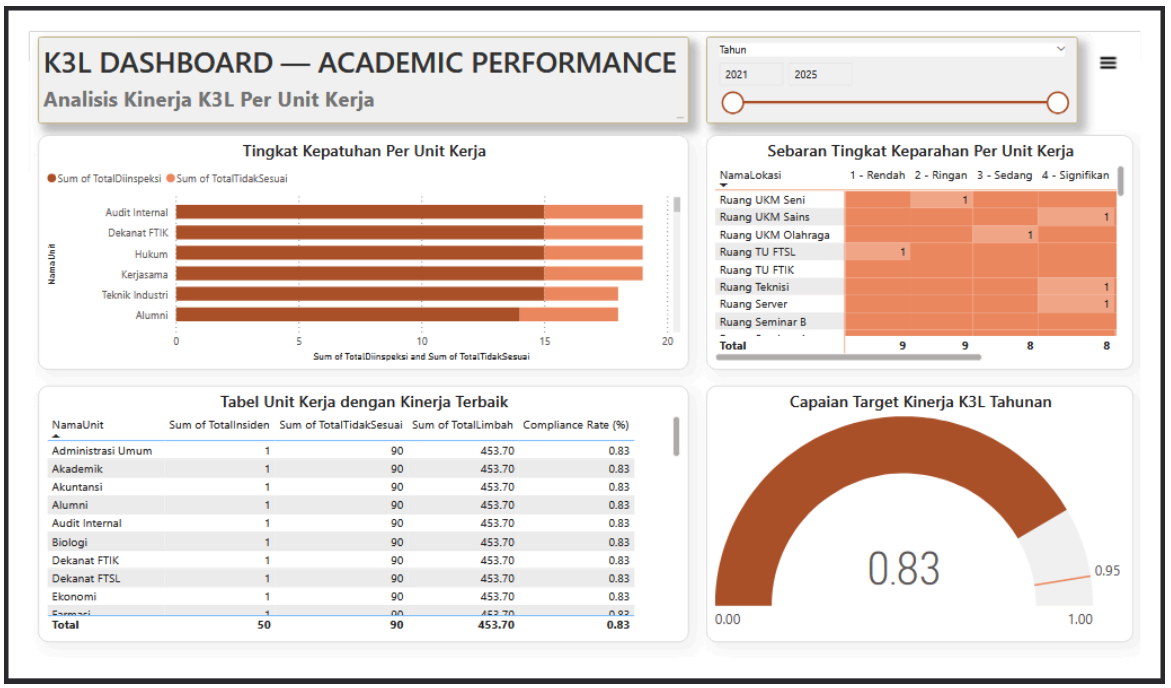
Dashboard Executive Summary dirancang untuk memberikan gambaran umum kinerja K3L fakultas melalui visualisasi data yang ringkas, informatif, dan mudah diinterpretasikan. Dashboard ini memuat indikator kinerja utama (KPI), tren data tahunan, distribusi insiden, serta rekapitulasi kinerja K3L.

Pada bagian atas *dashboard* ditampilkan tiga indikator utama, yaitu **Total Insiden sebanyak 50**, **Total Hari Kerja Hilang (*Lost Workdays*) sebanyak 227**, dan **Total Korban Insiden sebanyak 60** selama periode analisis 2021 hingga 2025. Ketiga indikator tersebut berfungsi sebagai ringkasan tingkat tinggi terhadap capaian dan kondisi K3L fakultas secara keseluruhan.

Selanjutnya, grafik **Total Insiden Selama 5 Tahun** menunjukkan fluktuasi total insiden dari tahun 2021 hingga 2025. Grafik terlihat mengalami peningkatan signifikan pada tahun 2022, stabil di tahun 2023, dan sempat menurun di tahun 2024, sebelum kemudian meningkat tajam di tahun 2025. Visualisasi ini membantu menganalisis pola kejadian insiden dari waktu ke waktu. Tabel **Ringkasan Kinerja K3L Tahunan** menyajikan data secara detail per tahun, mulai dari jumlah insiden, korban, hari kerja hilang, hingga **Tingkat Keparahan (*Severity Rate*)**. **Tingkat Keparahan tertinggi terjadi pada tahun 2022 dengan angka 5.89**, sedangkan total *Severity Rate* kumulatif selama 5 tahun adalah 4.54.

Grafik **Total Insiden Berdasarkan Nama** menunjukkan distribusi insiden yang relatif merata pada berbagai jenis insiden, dengan dominasi kecil pada beberapa kategori seperti 'Alergi Berat' dan 'Bahan Berbahaya' (masing-masing 2% dari total). Sedangkan, grafik **Total Korban Berdasarkan Unit** mengidentifikasi unit-unit dengan jumlah korban terbanyak, di mana **Matematika, Pemeliharaan, Public Relations, dan Rektorat** menunjukkan jumlah korban tertinggi, yaitu 3 korban. Secara keseluruhan, *dashboard* ini memberikan pandangan komprehensif mengenai kondisi K3L fakultas dan menjadi dasar pengambilan keputusan berbasis data.

B. Academic Performance



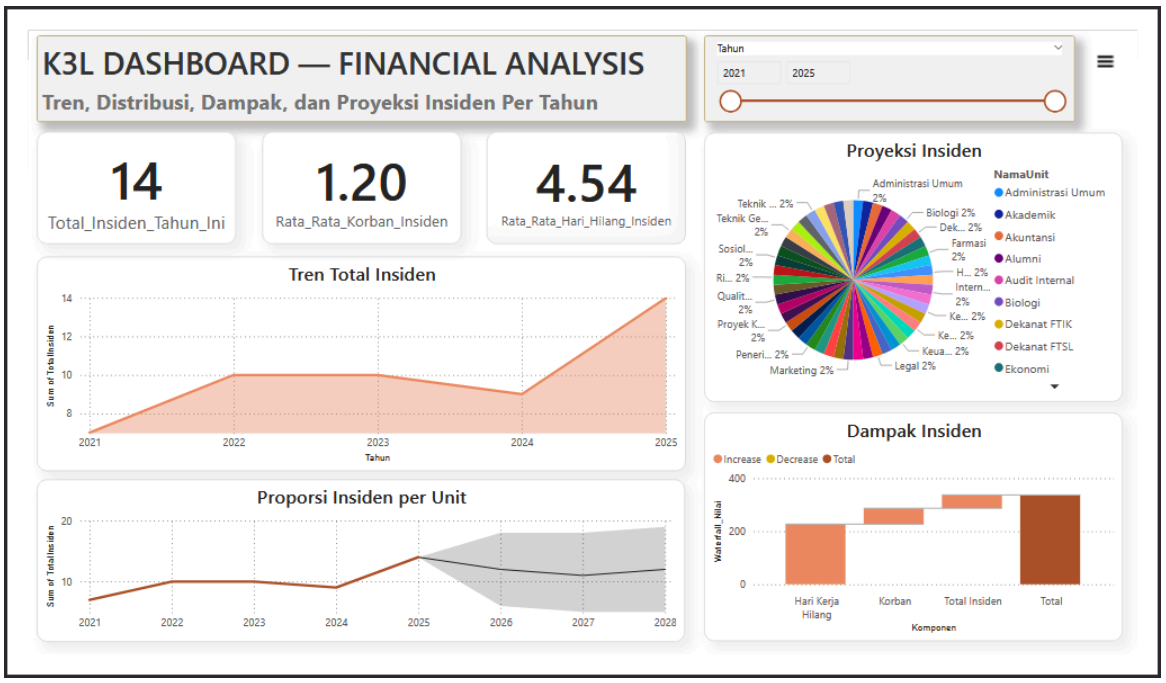
Dashboard Academic Performance menampilkan analisis terkait capaian kepatuhan K3L per unit kerja.

Visualisasi **Tingkat Kepatuhan Per Unit Kerja** menampilkan perbandingan antara *Total Diinspeksi* dan *Total Tidak Sesuai* (*Non-Compliance*) per unit. Dari grafik tersebut terlihat bahwa sebagian besar unit kerja, seperti **Audit Internal, Dekanat FTIK, dan Hukum**, memiliki proporsi temuan *Tidak Sesuai* yang signifikan.

Grafik **Sebaran Tingkat Keparahan Per Unit Kerja** mengklasifikasikan temuan berdasarkan tingkat keparahan (1-Rendah hingga 4-Signifikan) per lokasi. Distribusi menunjukkan bahwa **Tingkat Keparahan 'Rendah' (Level 1) memiliki sebaran terbanyak (9 kasus)**, diikuti oleh 'Signifikan' (Level 4) sebanyak 8 kasus di lokasi-lokasi seperti **Ruang UKM Seni, Ruang TU FTSL, dan Ruang Teknik**.

Tabel Unit Kerja dengan Kinerja Terbaik menyajikan detail kinerja K3L, termasuk *Compliance Rate*. Grafik **Capaian Target Kinerja K3L Tahunan** menunjukkan bahwa **Capaian Kepatuhan (0.83)** masih di bawah target yang ditetapkan (0.95), mengindikasikan bahwa upaya peningkatan kepatuhan dan penurunan insiden perlu ditingkatkan.

C. Financial Analysis (Tren, Distribusi, Dampak, dan Proyeksi Insiden Per Tahun)



Dashboard Financial Analysis menyajikan ringkasan dan analisis terkait tren insiden, dampaknya, serta proporsi insiden berdasarkan unit kerja. Tiga indikator utama yang ditampilkan adalah: **Total Insiden Tahun Ini (14)**, **Rata-Rata Korban Insiden (1.20)**, dan **Rata-Rata Hari Hilang (4.54)**.

Grafik **Tren Total Insiden** menunjukkan peningkatan insiden dari tahun 2021 hingga mencapai puncaknya di tahun 2025 dengan 14 insiden. Grafik **Proporsi Insiden per Unit** memperlihatkan tren insiden per unit kerja dari tahun 2021 hingga 2025, dan juga menyertakan proyeksi insiden hingga tahun 2028. Proyeksi ini mengindikasikan bahwa total insiden cenderung **meningkat** di tahun 2026 dan kemudian diproyeksikan **stabil** atau **menurun** setelah tahun 2027.

Grafik **Proyeksi Insiden** (dalam bentuk *pie chart*) menampilkan distribusi insiden berdasarkan nama unit, di mana **Administrasi Umum** memiliki proporsi terbesar (5%) diikuti oleh **Akademik, Akuntansi, dan Alumni** (masing-masing 3%), yang menunjukkan fokus pencegahan harus ditujukan pada unit-unit tersebut. Terakhir, grafik **Dampak Insiden** (*Waterfall Chart*) mengilustrasikan dampak insiden dalam tiga komponen: Hari Kerja Hilang (227), Korban (60), dan Total Insiden (50).

Secara keseluruhan, *dashboard* ini berfungsi sebagai alat strategis dalam memantau tren insiden K3L, mengevaluasi dampaknya terhadap hari kerja, serta menjadi referensi penting bagi perencanaan dan alokasi sumber daya K3L di tahun berikutnya.

Security Implementation Document

1.1 Create User Roles

```
USE K3L_DataMart;
GO

-- Create Roles [cite: 236]
CREATE ROLE db_executive; [cite: 237]
CREATE ROLE db_analyst; [cite: 238]
CREATE ROLE db_viewer; [cite: 239]
CREATE ROLE db_etl_operator; [cite: 240]
GO

---------------------------------------------
-- Executive Permissions (Full Read + ETL Execute)
---------------------------------------------
GRANT SELECT ON SCHEMA::dbo TO db_executive; [cite: 243]
GRANT EXECUTE ON SCHEMA::dbo TO db_executive; [cite: 243]
GO

---------------------------------------------
-- Analyst Permissions (DW + Staging)
---------------------------------------------
GRANT SELECT ON SCHEMA::dbo TO db_analyst; [cite: 246]
GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::stg TO db_analyst; [cite:
247]
GO

---------------------------------------------
-- Viewer Permissions (Read-only Views)
---------------------------------------------
GRANT SELECT ON dbo.vw_Insiden_Summary TO db_viewer; [cite: 250, 251]
GRANT SELECT ON dbo.vw_Inspeksi_Summary TO db_viewer; [cite: 252, 253]
GRANT SELECT ON dbo.vw_Limbah_Summary TO db_viewer; [cite: 254, 255]
GRANT SELECT ON dbo.vw_Executive_Summary_K3L TO db_viewer; [cite: 256, 257]
GO

---------------------------------------------
-- ETL Operator Permissions
---------------------------------------------
GRANT EXECUTE ON SCHEMA::dbo TO db_etl_operator; [cite: 259]
GRANT INSERT ON SCHEMA::dbo TO db_etl_operator; [cite: 259]
GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::stg TO db_etl_operator;
GO
```

1.2 Create Users and Assign Roles

```
USE master;
GO
```

```
-- Create SQL Logins (Server-level) [cite: 266]
CREATE LOGIN executive_user WITH PASSWORD = 'StrongP@ssw0rd!'; [cite: 267, 271]
CREATE LOGIN analyst_user WITH PASSWORD = 'StrongP@ssw0rd!'; [cite: 268, 271]
CREATE LOGIN viewer_user WITH PASSWORD = 'StrongP@ssw0rd!'; [cite: 269, 271]
CREATE LOGIN etl_service WITH PASSWORD = 'StrongP@ssw0rd!'; [cite: 270, 271]
GO

-- Create Database Users [cite: 273]
USE K3L_DataMart; [cite: 273]
GO
CREATE USER executive_user FOR LOGIN executive_user; [cite: 274, 277]
CREATE USER analyst_user FOR LOGIN analyst_user; [cite: 274, 277]
CREATE USER viewer_user FOR LOGIN viewer_user; [cite: 274, 277]
CREATE USER etl_service FOR LOGIN etl_service; [cite: 274, 277]
GO

-- Assign Role Memberships [cite: 275]
ALTER ROLE db_executive ADD MEMBER executive_user; [cite: 276, 278]
ALTER ROLE db_analyst ADD MEMBER analyst_user; [cite: 276, 278]
ALTER ROLE db_viewer ADD MEMBER viewer_user; [cite: 276, 278]
ALTER ROLE db_etl_operator ADD MEMBER etl_service; [cite: 279]
GO
```

1.3 Implement Data Masking

```
USE K3L_DataMart;
GO

-- Masking Rules (Contoh untuk data Karyawan/Kontak K3L)
-- Jika Dim_Karyawan ada, implementasi masking pada data kontak pribadi:
/*
ALTER TABLE dbo.Dim_Karyawan
ALTER COLUMN Email ADD MASKED WITH (FUNCTION = 'email()');
ALTER TABLE dbo.Dim_Karyawan
ALTER COLUMN Telp ADD MASKED WITH (FUNCTION =
'partial(0,"XXX-XXX-",4)');
*/
GO

-- Grant UNMASK permission untuk specific roles
GRANT UNMASK TO db_executive; [cite: 163]
GRANT UNMASK TO db_analyst; [cite: 163]
GO
```

1.4 Implement Audit Trail

```
USE K3L_DataMart; [cite: 295]
GO

-- Create Audit Table [cite: 297]
IF OBJECT_ID('dbo.AuditLog', 'U') IS NULL
BEGIN
   CREATE TABLE dbo.AuditLog ( [cite: 299]
      AuditID BIGINT IDENTITY(1,1) PRIMARY KEY, [cite: 300]
      EventTime DATETIME2 DEFAULT SYSDATETIME(), [cite: 300]
      UserName NVARCHAR(128) DEFAULT SUSER_SNAME(), [cite: 301]
      EventType NVARCHAR(50), [cite: 302]
      SchemaName NVARCHAR(128), [cite: 303]
      ObjectName NVARCHAR(128), [cite: 304]
      SQLStatement NVARCHAR(MAX) NULL, [cite: 305]
      RowsAffected INT, [cite: 306]
      HostName NVARCHAR(128) NULL, [cite: 307]
      IPAddress VARCHAR(50) NULL, [cite: 308]
      ApplicationName NVARCHAR(128) DEFAULT APP_NAME() [cite: 309]
   );
END
GO

-- Create Audit Trigger (Fact_Insiden) [cite: 312, 313]
CREATE OR ALTER TRIGGER trg_Audit_Fact_Insiden
ON dbo.Fact_Insiden [cite: 314]
AFTER INSERT, UPDATE, DELETE [cite: 315]
AS
BEGIN
   SET NOCOUNT ON; [cite: 318]
   DECLARE @EventType NVARCHAR(50); [cite: 320]
   IF EXISTS (SELECT 1 FROM inserted) AND EXISTS (SELECT 1 FROM deleted)
[cite: 321, 322]
      SET @EventType = 'UPDATE'; [cite: 323]
   ELSE IF EXISTS (SELECT 1 FROM inserted) [cite: 324]
      SET @EventType = 'INSERT'; [cite: 326]
   ELSE
      SET @EventType = 'DELETE'; [cite: 327]
   INSERT INTO dbo.AuditLog (EventType, SchemaName, ObjectName, RowsAffected)
[cite: 328, 329]
   VALUES (@EventType, 'dbo', 'Fact_Insiden', @@ROWCOUNT); [cite: 332]
END; [cite: 330]
GO

-- Audit Trigger (Fact_Inspeksi) [cite: 333, 334]
CREATE OR ALTER TRIGGER trg_Audit_Fact_Inspeksi
ON dbo.Fact_Inspeksi [cite: 335]
AFTER INSERT, UPDATE, DELETE [cite: 336]
AS
```

```
BEGIN
    SET NOCOUNT ON; [cite: 339]
    DECLARE @EventType NVARCHAR(50); [cite: 340]
    IF EXISTS (SELECT 1 FROM inserted) AND EXISTS (SELECT 1 FROM deleted)
[cite: 341, 342]
        SET @EventType = 'UPDATE'; [cite: 343]
    ELSE IF EXISTS (SELECT 1 FROM inserted) [cite: 344]
        SET @EventType = 'INSERT'; [cite: 346]
    ELSE
        SET @EventType = 'DELETE'; [cite: 347]
    INSERT INTO dbo.AuditLog (EventType, SchemaName, ObjectName, RowsAffected)
[cite: 348, 349]
    VALUES (@EventType, 'dbo', 'Fact_Inspeksi', @@ROWCOUNT); [cite: 351]
END; [cite: 350]
GO

-- Audit Trigger (Fact_Limbah) [cite: 352, 353]
CREATE OR ALTER TRIGGER trg_Audit_Fact_Limbah
ON dbo.Fact_Limbah [cite: 354]
AFTER INSERT, UPDATE, DELETE [cite: 355]
AS
BEGIN
    SET NOCOUNT ON; [cite: 358]
    DECLARE @EventType NVARCHAR(50); [cite: 359]
    IF EXISTS (SELECT 1 FROM inserted) AND EXISTS (SELECT 1 FROM deleted)
[cite: 360, 361]
        SET @EventType = 'UPDATE'; [cite: 362]
    ELSE IF EXISTS (SELECT 1 FROM inserted) [cite: 363]
        SET @EventType = 'INSERT'; [cite: 365]
    ELSE
        SET @EventType = 'DELETE'; [cite: 366]
    INSERT INTO dbo.AuditLog (EventType, SchemaName, ObjectName, RowsAffected)
[cite: 367]
    VALUES (@EventType, 'dbo', 'Fact_Limbah', @@ROWCOUNT); [cite: 369]
END; [cite: 368]
GO

-- Enable SQL Server Audit (Server-level) [cite: 371]
USE master; [cite: 372]
GO
CREATE SERVER AUDIT K3L_DataMart_Audit [cite: 374]
TO FILE [cite: 375]
(
    FILEPATH = N'C:\Audit', [cite: 377]
    MAXSIZE = 100 MB, [cite: 378]
    MAX_ROLLOVER_FILES = 10 [cite: 379]
)
WITH (ON_FAILURE = CONTINUE); [cite: 381]
GO
```

```
ALTER SERVER AUDIT K3L_DataMart_Audit WITH (STATE = ON); [cite: 383]
GO

-- Create Database Audit Specification [cite: 384]
USE K3L_DataMart; [cite: 385]
GO
CREATE DATABASE AUDIT SPECIFICATION K3L_DB_Audit [cite: 387]
FOR SERVER AUDIT K3L_DataMart_Audit [cite: 388]
ADD (SELECT, INSERT, UPDATE, DELETE ON SCHEMA::dbo BY public); [cite: 389]
GO
ALTER DATABASE AUDIT SPECIFICATION K3L_DB_Audit WITH (STATE = ON);
[cite: 391]
GO
```

Penjelasan:
Implementasi keamanan pada **Data Mart K3L** dilakukan melalui pembuatan *role-based access control* (RBAC) yang membagi hak akses menjadi empat peran utama: db_executive, db_analyst, db_viewer, dan db_etl_operator. Setiap peran diberikan izin spesifik sesuai kebutuhan operasional dan analitik data K3L.

- **Access Control:** Setiap *role* dihubungkan dengan *user login* yang dibuat secara terpisah, memastikan akses terkontrol.
- **Data Masking:** Untuk melindungi data sensitif (misalnya, kontak Karyawan/Insiden), digunakan *dynamic data masking* pada kolom sensitif (misalnya, **Email** dan **Telp** pada dimensi terkait), dengan izin UNMASK hanya diberikan kepada *role* db_executive dan db_analyst.
- **Audit Trail:** Sistem dilengkapi *audit trail* melalui tabel dbo.AuditLog dan *Audit Trigger* yang mencatat semua aktivitas DML (INSERT, UPDATE, dan DELETE) pada tabel fakta utama (Fact_Insiden, Fact_Inspeksi, Fact_Limbah).
- **Server Audit:** *Audit server-level* (K3L_DataMart_Audit) dan *database-level* (K3L_DB_Audit) juga diaktifkan untuk memantau seluruh operasi penting.

Backup Schedule Document

Backup SQL Scripts

```
-- Full Backup
BACKUP DATABASE K3L_DataMart
TO DISK = N'C:\Backup\K3L_DataMart_Full.bak'
WITH
 COMPRESSION,
 INIT,
 NAME = N'Full Backup K3L_DataMart',
 STATS = 10;
GO

-- Differential Backup
BACKUP DATABASE K3L_DataMart
TO DISK = N'C:\Backup\K3L_DataMart_Diff.bak'
WITH
 DIFFERENTIAL,
 COMPRESSION,
 INIT,
 NAME = N'Differential Backup K3L_DataMart',
 STATS = 10;
GO

-- Transaction Log Backup
BACKUP LOG K3L_DataMart
TO DISK = N'C:\Backup\K3L_DataMart_Log.trn'
WITH
 COMPRESSION,
 INIT,
 NAME = N'Transaction Log Backup K3L_DataMart',
 STATS = 10;
GO
```

Jadwal Backup dan Opsi Azure

```
-- 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 - Offsite Redundancy)
CREATE CREDENTIAL [AzureStorageCredential]
WITH
 IDENTITY = 'SHARED ACCESS SIGNATURE',
 SECRET = '<SAS TOKEN>'; -- Ganti dengan SAS TOKEN yang valid
GO
```

```
BACKUP DATABASE K3L_DataMart
TO URL =
N'https://[storage_account].blob.core.windows.net/backups/K3L_DataMart_FULL.bak'
WITH
 CREDENTIAL = 'AzureStorageCredential',
 COMPRESSION,
 STATS = 10;
GO
```

Penjelasan:
Strategi *backup* untuk Data Mart K3L menggunakan kombinasi Full Backup, Differential Backup, dan Transaction Log Backup untuk memastikan integritas dan ketersediaan data (K3L) secara optimal.

- Full Backup: Dibuat setiap minggu (Minggu pukul 02:00 AM). Ini berfungsi sebagai salinan lengkap seluruh database dan pondasi utama untuk proses pemulihan.
- Differential Backup: Dijalankan setiap hari (pukul 02:00 AM). Ini hanya merekam perubahan data sejak *full backup* terakhir, mempercepat proses *restore* data K3L dibandingkan hanya mengandalkan *full backup*.
- Transaction Log Backup: Dilakukan setiap enam jam. Frekuensi tinggi ini memungkinkan *point-in-time recovery* (pemulihan hingga titik waktu tertentu) dan meminimalkan risiko kehilangan data K3L saat terjadi kegagalan.
- Offsite Redundancy (Opsional): Tersedia opsi untuk menyimpan *backup* ke Azure Blob Storage sebagai langkah redundansi *offsite* untuk menghadapi skenario bencana.

RECOVERY PROCEDURES DOCUMENT

Skrip Backup

```
-- Full Backup (Mingguan)
BACKUP DATABASE K3L_DataMart
TO DISK = N'C:\Backup\K3L_DataMart_Full.bak' [cite: 395]
WITH
 COMPRESSION,
 INIT,
 NAME = N'Full Backup K3L_DataMart', [cite: 399]
 STATS = 10;
GO

-- Differential Backup (Harian)
BACKUP DATABASE K3L_DataMart
TO DISK = N'C:\Backup\K3L_DataMart_Diff.bak' [cite: 401]
WITH
 DIFFERENTIAL,
 COMPRESSION,
 INIT,
 NAME = N'Differential Backup K3L_DataMart', [cite: 405]
 STATS = 10;
GO

-- Transaction Log Backup (Setiap 6 Jam)
BACKUP LOG K3L_DataMart
TO DISK = N'C:\Backup\K3L_DataMart_Log.trn' [cite: 408]
WITH
 COMPRESSION,
 INIT,
 NAME = N'Transaction Log Backup K3L_DataMart', [cite: 412]
 STATS = 10;
GO
```

Prosedur Pemulihan (Recovery)

Prosedur pemulihan dilakukan secara berurutan, dimulai dari *Full Backup* terbaru, diikuti oleh *Differential Backup* terbaru, dan diakhiri dengan *Transaction Log Backup* hingga titik waktu yang diinginkan (*Point-in-Time Recovery*).

Langkah 1: Restore Full Backup (Basis Awal) *Restore Full Backup* terbaru menggunakan opsi WITH NORECOVERY agar *differential* dan *log backup* dapat diterapkan setelahnya.

```
RESTORE DATABASE K3L_DataMart
FROM DISK = N'C:\Backup\K3L_DataMart_Full.bak'
WITH NORECOVERY, REPLACE;
GO
```

Langkah 2: Restore Differential Backup (Kondisi Mendekati Terbaru) Terapkan *Differential Backup* terbaru.

```
RESTORE DATABASE K3L_DataMart
FROM DISK = N'C:\Backup\K3L_DataMart_Diff.bak'
WITH NORECOVERY;
GO
```

Langkah 3: Restore Transaction Log Backup (Point-in-Time Recovery) Terapkan semua *Transaction Log Backup* secara berurutan sejak *Differential Backup* terakhir, menggunakan WITH NORECOVERY.

```
-- Ulangi untuk setiap log file yang dibuat sejak Differential Backup
RESTORE LOG K3L_DataMart
FROM DISK = N'C:\Backup\K3L_DataMart_Log_1.trn'
WITH NORECOVERY;
GO

-- Log terakhir: Gunakan WITH RECOVERY atau STOPAT untuk Point-in-Time
RESTORE LOG K3L_DataMart
FROM DISK = N'C:\Backup\K3L_DataMart_Log_N.trn'
WITH RECOVERY; -- Atau WITH STOPAT = 'YYYY-MM-DD hh:mm:ss' untuk pemulihan waktu spesifik
GO
```

Penjelasan:

Proses *recovery* Data Mart K3L dirancang untuk meminimalkan kehilangan data dan *downtime* sistem. Pemulihan dilakukan dengan menerapkan urutan yang ketat: dimulai dari *Full Backup* sebagai basis awal, kemudian menambahkan *Differential Backup* untuk membawa database ke kondisi mendekati terbaru, dan terakhir melanjutkan dengan *Transaction Log Backup* untuk mengembalikan seluruh transaksi hingga titik waktu yang diinginkan.

Pendekatan bertahap ini memungkinkan proses pemulihan yang cepat, konsisten, dan fleksibel, yang penting untuk Data Mart K3L. Hal ini dapat mengatasi kerusakan data, kesalahan operasional, maupun kegagalan sistem secara menyeluruh. Dengan skema ini, data K3L dapat dipulihkan secara akurat tanpa mengorbankan kontinuitas operasional.