LAPORAN PROSES PEMBANGUNAN DATA WAREHOUSE – DATA MART FTIK

Disusun Oleh:
MUHAMMAD ZAKY ZAIDDAN (122440119)
EIGI ARTAMEVIA (123450011)
SALSABILA PUTRI MAHARANI (123450070)
HANIFAH INAYA SANI (123450123)

PROGRAM STUDI SAINS DATA

FAKULTAS SAINS

INSTITUT TEKNOLOGI SUMATERA

2025

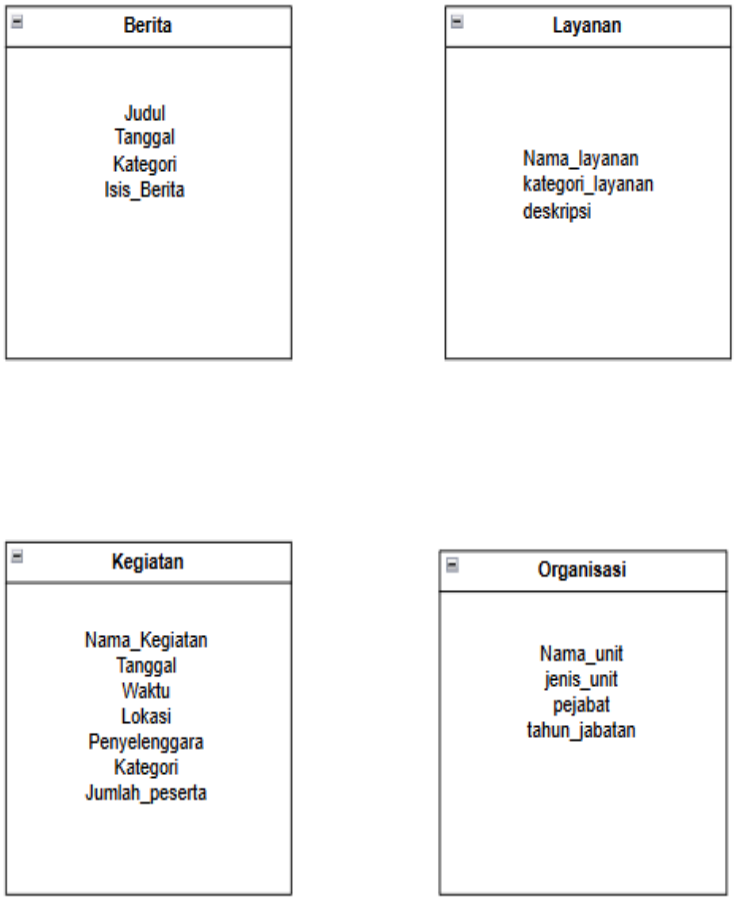
3. Desain ERD & ETL Mapping

A. Skema Awal (Staging Area)

Berikut adalah tabel *staging* yang berfungsi sebagai area pendaratan data mentah (Source) sebelum proses transformasi (ETL) dimulai:

```
USE DWMart_FTIK;
GO

CREATE SCHEMA stg;
GO

CREATE TABLE stg.Berita (
    judul NVARCHAR(300),
    tanggal_raw NVARCHAR(50),
    kategori_raw NVARCHAR(100),
    isi_berita NVARCHAR(MAX),
    url NVARCHAR(300),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO

CREATE TABLE stg.Kegiatan (
    nama_kegiatan NVARCHAR(300),
    tanggal_raw NVARCHAR(50),
    waktu_raw NVARCHAR(50),
    lokasi NVARCHAR(200),
    penyelenggara NVARCHAR(200),
    kategori_raw NVARCHAR(100),
    jumlah_peserta_raw NVARCHAR(50),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO

CREATE TABLE stg.Prodi (
    nama_prodi NVARCHAR(200),
```

```
    jenjang NVARCHAR(50),
    deskripsi NVARCHAR(MAX),
    kepala_prodi NVARCHAR(200),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO

CREATE TABLE stg.Layanan (
    nama_layanan NVARCHAR(200),
    kategori_layanan_raw NVARCHAR(200),
    deskripsi NVARCHAR(MAX),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO

CREATE TABLE stg.Organisasi (
    nama_unit NVARCHAR(200),
    jenis_unit_raw NVARCHAR(100),
    pejabat NVARCHAR(200),
    tahun_jabatan_raw NVARCHAR(50),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO
```



Schema staging ini memang tidak memiliki relasi karena fungsinya hanya sebagai tempat penampungan data mentah (*raw data*) yang diambil langsung dari sumber sebelum dibersihkan atau ditransformasi. Pada tahap ini, struktur data dibiarkan apa adanya tanpa penentuan primary key, foreign key, atau hubungan antar tabel agar proses ekstraksi menjadi lebih fleksibel, cepat, dan tidak terikat aturan integritas. Relasi baru akan dibentuk nanti pada tahap selanjutnya.

B. ETL Procedures

```
CREATE PROCEDURE usp_Load_Dim_Waktu
AS
BEGIN
   INSERT INTO DIM_WAKTU (tanggal, hari, bulan, tahun, minggu_ke)
   SELECT
      s.tanggal,
      DATENAME(WEEKDAY, s.tanggal),
      MONTH(s.tanggal),
      YEAR(s.tanggal),
      DATEPART(WEEK, s.tanggal)
   FROM stg.Waktu s
   WHERE NOT EXISTS (SELECT 1 FROM DIM_WAKTU d WHERE
d.tanggal = s.tanggal);

   PRINT 'DIM_WAKTU loaded.';
END;
GO

CREATE PROCEDURE usp_Load_Dim_Prodi
AS
BEGIN
   INSERT INTO DIM_PRODI (nama_prodi, jenjang, deskripsi)
   SELECT s.nama_prodi, s.jenjang, s.deskripsi
   FROM stg.Prodi s
   WHERE NOT EXISTS (SELECT 1 FROM DIM_PRODI d WHERE
d.nama_prodi = s.nama_prodi);

   PRINT 'DIM_PRODI loaded.';
END;
GO

CREATE PROCEDURE usp_Load_Dim_Kategori
AS
BEGIN
   INSERT INTO DIM_KATEGORI (nama_kategori)
```

```
    SELECT nama_kategori
    FROM stg.Kategori s
    WHERE NOT EXISTS (
        SELECT 1 FROM DIM_KATEGORI d
        WHERE d.nama_kategori = s.nama_kategori
    );

    PRINT 'DIM_KATEGORI loaded.';
END;
GO

CREATE PROCEDURE usp_Load_Dim_Layanan
AS
BEGIN
    INSERT INTO DIM_LAYANAN (nama_layanan, kategori_layanan,
deskripsi)
    SELECT nama_layanan, kategori_layanan, deskripsi
    FROM stg.Layanan s
    WHERE NOT EXISTS (SELECT 1 FROM DIM_LAYANAN d
WHERE d.nama_layanan = s.nama_layanan);

    PRINT 'DIM_LAYANAN loaded.';
END;
GO

CREATE PROCEDURE usp_Load_Dim_Organisasi
AS
BEGIN
    INSERT INTO DIM_ORGANISASI (nama_unit, jenis_unit, pejabat)
    SELECT nama_unit, jenis_unit, pejabat
    FROM stg.Organisasi s
    WHERE NOT EXISTS (SELECT 1 FROM DIM_ORGANISASI d
WHERE d.nama_unit = s.nama_unit);

    PRINT 'DIM_ORGANISASI loaded.';
END;
```

```
GO

CREATE PROCEDURE usp_Load_Fact_Kegiatan
AS
BEGIN
   INSERT INTO FACT_KEGIATAN
   (
      waktu_id, prodi_id, layanan_id, kategori_id, organisasi_id,
      jumlah_peserta, durasi_menit
   )
   SELECT
      w.waktu_id,
      p.prodi_id,
      l.layanan_id,
      k.kategori_id,
      o.organisasi_id,
      s.jumlah_peserta,
      s.durasi_menit
   FROM stg.Kegiatan s
   LEFT JOIN DIM_WAKTU w ON w.tanggal = s.tanggal
   LEFT JOIN DIM_PRODI p ON p.nama_prodi = s.nama_prodi
   LEFT JOIN DIM_LAYANAN l ON l.nama_layanan = s.nama_layanan
   LEFT JOIN DIM_KATEGORI k ON k.nama_kategori =
s.nama_kategori
   LEFT JOIN DIM_ORGANISASI o ON o.nama_unit = s.nama_unit;

   PRINT 'FACT_KEGIATAN loaded.';
END;
GO

CREATE PROCEDURE usp_Load_Master_ETL
AS
BEGIN
   EXEC usp_Load_Dim_Waktu;
   EXEC usp_Load_Dim_Prodi;
   EXEC usp_Load_Dim_Layanan;
```

```
    EXEC usp_Load_Dim_Kategori;
    EXEC usp_Load_Dim_Organisasi;

    EXEC usp_Load_Fact_Kegiatan;

    PRINT 'MASTER ETL SUCCESS';
END;
GO
```

1. Prosedur Load Dimensi (INSERT Non-Repeating Data)

Prosedur-prosedur ini bertanggung jawab untuk mengisi tabel dimensi. Logika utamanya adalah **menghindari duplikasi** dengan hanya memasukkan baris yang *belum ada* di tabel dimensi target, berdasarkan *Natural Key*-nya.

➔ **usp_Load_Dim_Waktu**
- **Fungsi:** Memuat data waktu (tanggal) ke dalam tabel DIM_WAKTU.
- **Sumber Data:** stg.Waktu.
- **Transformasi:** Melakukan *parsing* dan ekstraksi elemen waktu dari kolom tanggal di *staging*:
  - hari: Diambil nama hari (DATENAME(WEEKDAY, s.tanggal)).
  - bulan, tahun: Diambil nilai numerik bulan dan tahun.
  - minggu_ke: Diambil nomor minggu dalam setahun (DATEPART(WEEK, s.tanggal)).
- **Filter Duplikasi:** Menggunakan WHERE NOT EXISTS untuk memastikan tanggal yang sama tidak dimasukkan dua kali ke DIM_WAKTU.

➔ **usp_Load_Dim_Prodi**
- **Fungsi:** Memuat dimensi Program Studi.
- **Sumber Data:** stg.Prodi.
- **Transformasi:** Data nama_prodi, jenjang, dan deskripsi diambil langsung dari *staging*.
- **Filter Duplikasi:** Hanya baris dengan nama_prodi baru yang akan dimasukkan ke DIM_PRODI.

➔ **usp_Load_Dim_Kategori**
- **Fungsi:** Memuat dimensi Kategori (misalnya, kategori Berita atau Kegiatan).
- **Sumber Data:** stg.Kategori.
- **Filter Duplikasi:** Hanya baris dengan nama_kategori baru yang akan dimasukkan ke DIM_KATEGORI.

➔ **usp_Load_Dim_Layanan**

- **Fungsi:** Memuat dimensi Layanan.
- **Sumber Data:** stg.Layanan.
- **Filter Duplikasi:** Hanya baris dengan nama_layanan baru yang akan dimasukkan ke DIM_LAYANAN.

➔ **usp_Load_Dim_Organisasi**

- **Fungsi:** Memuat dimensi Organisasi/Unit Kerja.
- **Sumber Data:** stg.Organisasi.
- **Filter Duplikasi:** Hanya baris dengan nama_unit baru yang akan dimasukkan ke DIM_ORGANISASI.

2. Prosedur Load Fakta (Key Lookup)

Prosedur ini bertanggung jawab untuk memuat metrik ke tabel fakta, menggantikan *Natural Key* (seperti nama_prodi atau tanggal) dengan *Surrogate Key* (*_id atau *_key).

➔ **usp_Load_Fact_Kegiatan**

- **Fungsi:** Memuat data transaksi (fakta) Kegiatan ke dalam FACT_KEGIATAN.
- **Sumber Data:** stg.Kegiatan.
- **Transformasi Kunci:** Melakukan **LOOKUP** (menggunakan LEFT JOIN) ke semua tabel dimensi (DIM_WAKTU, DIM_PRODI, DIM_LAYANAN, DIM_KATEGORI, DIM_ORGANISASI) berdasarkan *Natural Key* yang sesuai (s.tanggal, s.nama_prodi, dll.) untuk mengambil *Surrogate Key* (waktu_id, prodi_id, dll.).
- **Metrik:** Memuat metrik kuantitatif langsung dari *staging* (jumlah_peserta, durasi_menit).
- **Catatan:** Dalam implementasi ETL yang ketat, baris yang gagal di-*lookup* salah satu *key* (hasil JOIN adalah NULL) biasanya akan diabaikan atau dicatat untuk *error handling*.

3. Prosedur Master (Orkestrasi)

➔ **usp_Load_Master_ETL**

- **Fungsi:** Mengorkestrasi seluruh proses ETL dalam urutan yang benar.
- **Urutan Eksekusi (Kritis):**
  1. **Load Dimensi:** Semua prosedur dimensi (usp_Load_Dim_*) harus dijalankan **terlebih dahulu**. Ini memastikan bahwa semua *Surrogate Key* tersedia saat tabel fakta diisi.
  2. **Load Fakta:** Prosedur fakta (usp_Load_Fact_Kegiatan) dijalankan **terakhir** karena ia bergantung pada keberadaan *key* di semua dimensi.
- **Hasil:** Mencetak pesan sukses setelah semua tahapan selesai.

C. Data Quality Checks

```
-- check null values

SELECT
   'DIM_WAKTU' AS TableName,
   COUNT(*) AS TotalRows,
   SUM(CASE WHEN tanggal IS NULL THEN 1 ELSE 0 END) AS NullTanggal,
   SUM(CASE WHEN hari IS NULL THEN 1 ELSE 0 END) AS NullHari,
   SUM(CASE WHEN bulan IS NULL THEN 1 ELSE 0 END) AS NullBulan,
   SUM(CASE WHEN tahun IS NULL THEN 1 ELSE 0 END) AS NullTahun
FROM DIM_WAKTU;

SELECT
   'DIM_PRODI' AS TableName,
   COUNT(*) AS TotalRows,
   SUM(CASE WHEN nama_prodi IS NULL THEN 1 ELSE 0 END) AS
NullNamaProdi,
   SUM(CASE WHEN jenjang IS NULL THEN 1 ELSE 0 END) AS NullJenjang
FROM DIM_PRODI;

SELECT
   'DIM_LAYANAN' AS TableName,
   COUNT(*) AS TotalRows,
   SUM(CASE WHEN nama_layanan IS NULL THEN 1 ELSE 0 END) AS
NullNamaLayanan
FROM DIM_LAYANAN;

SELECT
   'DIM_KATEGORI' AS TableName,
   COUNT(*) AS TotalRows,
   SUM(CASE WHEN nama_kategori IS NULL THEN 1 ELSE 0 END) AS
NullNamaKategori
FROM DIM_KATEGORI;

SELECT
   'DIM_ORGANISASI' AS TableName,
   COUNT(*) AS TotalRows,
   SUM(CASE WHEN nama_unit IS NULL THEN 1 ELSE 0 END) AS NullUnit
FROM DIM_ORGANISASI;

SELECT
```

```
    'FACT_KEGIATAN' AS TableName,
    COUNT(*) AS TotalRows,
    SUM(CASE WHEN waktu_id IS NULL THEN 1 ELSE 0 END) AS NullWaktu,
    SUM(CASE WHEN kategori_id IS NULL THEN 1 ELSE 0 END) AS NullKategori
FROM FACT_KEGIATAN;
```

```
1  -- check null values
2
```

75 % | No issues found | Ln: 3 | Ch: 8 | SPC | CRLF

Results | Messages

	TableName	TotalRows	NullTanggal	NullHari	NullBulan	NullTahun
1	DIM_WAKTU	0	NULL	NULL	NULL	NULL

	TableName	TotalRows	NullNamaProdi	NullJenjang
1	DIM_PRODI	0	NULL	NULL

	TableName	TotalRows	NullNamaLayanan
1	DIM_LAYANAN	0	NULL

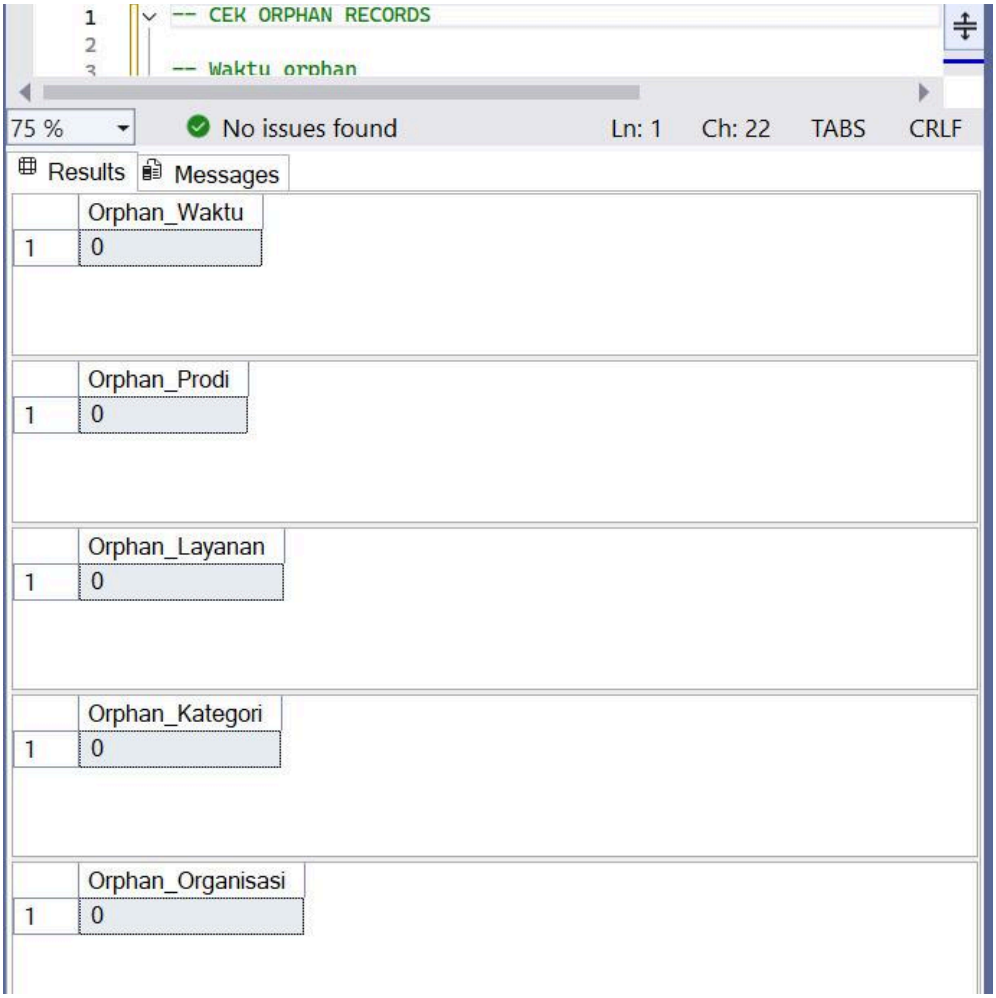
	TableName	TotalRows	NullNamaKategori
1	DIM_KATEGORI	0	NULL

	TableName	TotalRows	NullUnit
1	DIM_ORGANISASI	0	NULL

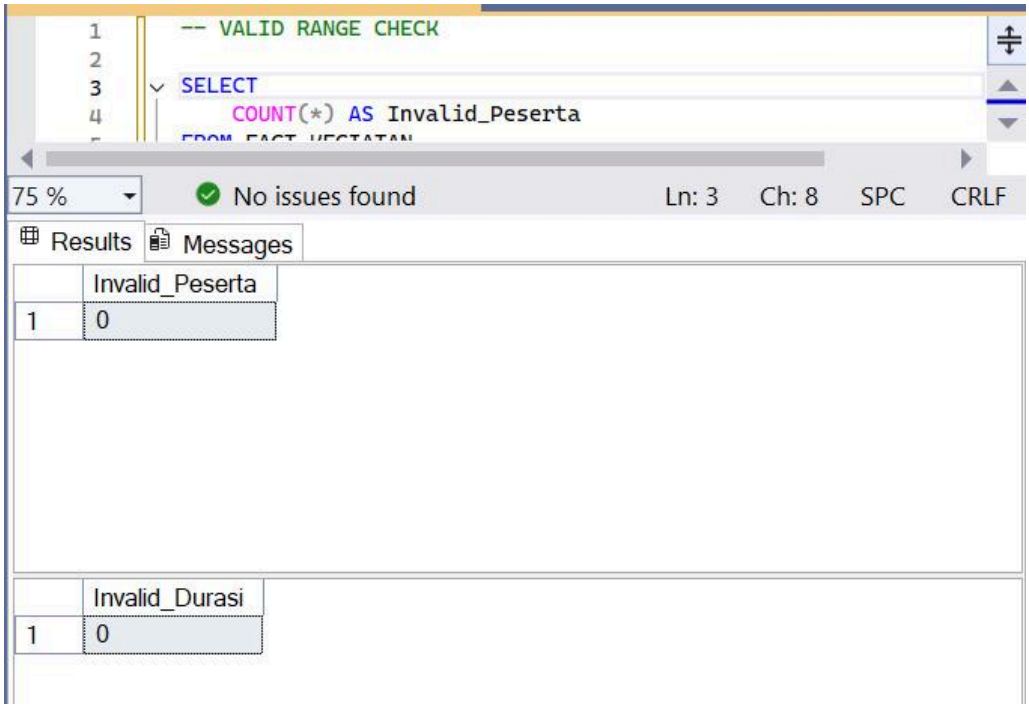
	TableName	TotalRows	NullWaktu	NullKategori
1	FACT_KEGIATAN	0	NULL	NULL

```
-- CEK ORPHAN RECORDS

-- Waktu orphan
SELECT COUNT(*) AS Orphan_Waktu
FROM FACT_KEGIATAN f
LEFT JOIN DIM_WAKTU d ON f.waktu_id = d.waktu_id
WHERE d.waktu_id IS NULL;

-- Prodi orphan
SELECT COUNT(*) AS Orphan_Prodi
FROM FACT_KEGIATAN f
LEFT JOIN DIM_PRODI d ON f.prodi_id = d.prodi_id
WHERE f.prodi_id IS NOT NULL AND d.prodi_id IS NULL;
```

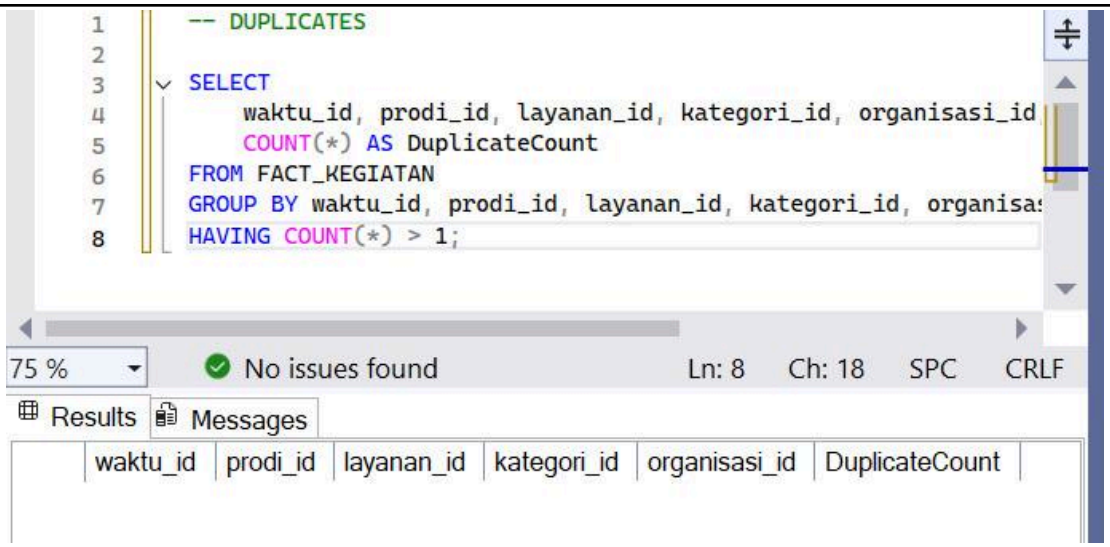
```
-- Layanan orphan
SELECT COUNT(*) AS Orphan_Layanan
FROM FACT_KEGIATAN f
LEFT JOIN DIM_LAYANAN d ON f.layanan_id = d.layanan_id
WHERE f.layanan_id IS NOT NULL AND d.layanan_id IS NULL;

-- Kategori orphan
SELECT COUNT(*) AS Orphan_Kategori
FROM FACT_KEGIATAN f
LEFT JOIN DIM_KATEGORI d ON f.kategori_id = d.kategori_id
WHERE f.kategori_id IS NOT NULL AND d.kategori_id IS NULL;

-- Organisasi orphan
SELECT COUNT(*) AS Orphan_Organisasi
FROM FACT_KEGIATAN f
LEFT JOIN DIM_ORGANISASI d ON f.organisasi_id = d.organisasi_id
WHERE f.organisasi_id IS NOT NULL AND d.organisasi_id IS NULL;
```



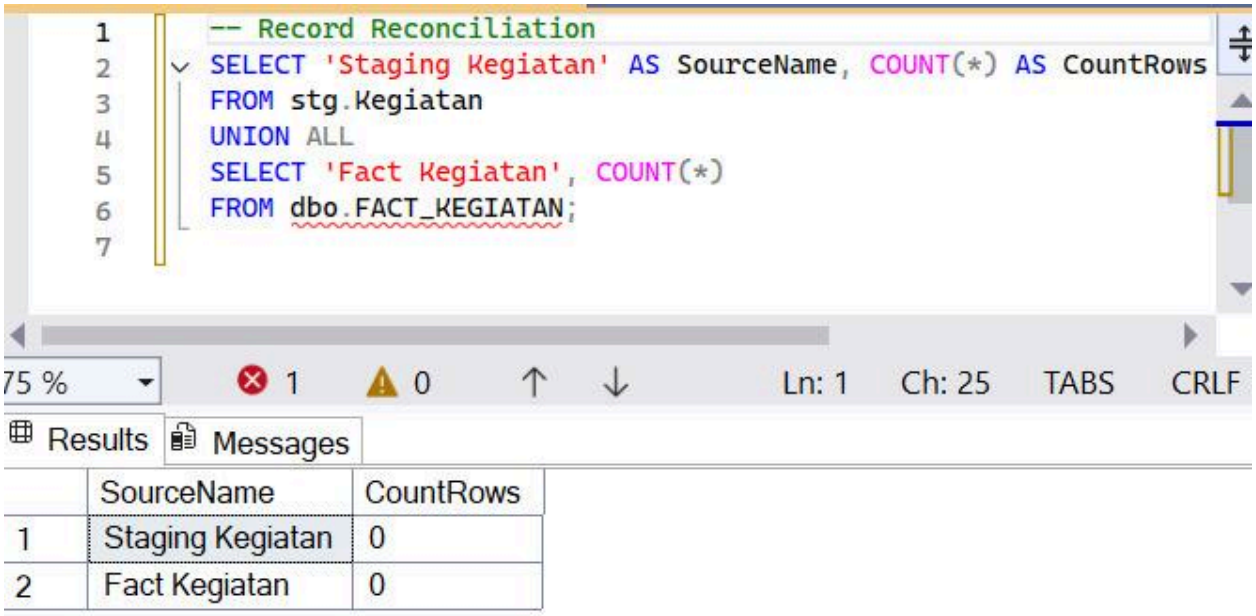

```
-- VALID RANGE CHECK

SELECT
   COUNT(*) AS Invalid_Peserta
FROM FACT_KEGIATAN
WHERE jumlah_peserta < 0
  OR jumlah_peserta > 10000;   -- batas aman

SELECT
   COUNT(*) AS Invalid_Durasi
FROM FACT_KEGIATAN
WHERE durasi_menit < 0
  OR durasi_menit > 1440;  -- max 24 jam
```




```
-- DUPLICATES

SELECT
   waktu_id, prodi_id, layanan_id, kategori_id, organisasi_id,
   COUNT(*) AS DuplicateCount
FROM FACT_KEGIATAN
GROUP BY waktu_id, prodi_id, layanan_id, kategori_id, organisasi_id
HAVING COUNT(*) > 1;
```



```
-- RECORD COUNT RECONCILIATION

SELECT 'Staging Kegiatan' AS SourceName, COUNT(*) AS CountRows
FROM stg.Kegiatan
UNION ALL
SELECT 'Fact Kegiatan', COUNT(*)
FROM FACT_KEGIATAN;
```



	SourceName	CountRows
1	Staging Kegiatan	0
2	Fact Kegiatan	0

1. Pemeriksaan Nilai Null (Null Value Check)

Tujuan: Memastikan kolom-kolom kritis (terutama *Natural Key* pada Dimensi dan atribut wajib lainnya) tidak mengandung nilai NULL. Nilai NULL yang tidak diharapkan dapat menyebabkan kegagalan dalam *join* atau menghasilkan laporan yang tidak akurat.

Target Query	Kolom yang Diperiksa	Pentingnya Pemeriksaan
DIM_WAKTU	tanggal, hari, bulan, tahun	Nilai-nilai ini harus lengkap dan valid karena digunakan untuk navigasi waktu dalam analisis.

DIM_PRODI	nama_prodi, jenjang	nama_prodi adalah *Natural Key* yang digunakan untuk mengidentifikasi dan mengisi tabel fakta.
DIM_LAYANAN	nama_layanan	*Natural Key* layanan wajib ada.
DIM_KATEGORI	nama_kategori	*Natural Key* kategori wajib ada.
DIM_ORGANISASI	nama_unit	*Natural Key* unit organisasi wajib ada.
FACT_KEGIATAN	waktu_id, kategori_id	Memastikan *Foreign Key* ke Dimensi Waktu dan Kategori berhasil diisi selama proses *lookup* ETL. Nilai NULL di sini menunjukkan kegagalan *lookup*.

2. Pemeriksaan *Orphan Records* (Referential Integrity Check)

Tujuan: Memastikan integritas referensial antara tabel fakta dan dimensi. *Orphan Record* adalah baris di tabel fakta yang memiliki *Foreign Key* (*_id) yang nilainya tidak ditemukan di tabel dimensi yang direferensikannya.

Target Query	Keterangan	Signifikansi
Orphan_Waktu, Orphan_Prodi, Orphan_Layanan, Orphan_Kategori, Orphan_Organisasi	Menggunakan LEFT JOIN dari FACT_KEGIATAN ke setiap DIM lalu menghitung baris di mana *Primary Key* dimensi adalah NULL.	Jika jumlahnya lebih dari 0, berarti terdapat data kegiatan yang tidak dapat dihubungkan ke data dimensi yang valid. Ini bisa disebabkan oleh *Natural Key* yang salah di *staging* atau dimensi yang belum dimuat sepenuhnya.

3. Pemeriksaan Batas Nilai Wajar (Valid Range Check)

Tujuan: Memastikan (metrik kuantitatif) dalam tabel fakta berada dalam rentang nilai yang logis dan wajar, mencegah anomali data (misalnya nilai negatif).

Target Query	Kolom yang Diperiksa	Batas Wajar yang Ditetapkan	Signifikansi
Invalid_Peserta	jumlah_peserta	0 atau 10000 (Batas aman)	Mengidentifikasi data yang secara fisik mustahil (peserta negatif) atau tidak realistis (di luar batas maksimal kapasitas).
Invalid_Durasi	durasi_menit	0 atau 1440 (Maksimal 24 jam)	Mengidentifikasi durasi negatif atau durasi yang melebihi batas logis untuk sebuah kegiatan.

4. Pemeriksaan Duplikasi (Duplicates Check)

Tujuan: Memastikan tidak ada baris ganda (duplikasi) di tabel fakta berdasarkan kombinasi *Foreign Key* (dimensi yang membentuk fakta unik).

Target Query	Keterangan	Signifikansi
DuplicateCount	Mengelompokkan berdasarkan semua *Foreign Key* (waktu_id, prodi_id, layanan_id, kategori_id, organisasi_id) dan mencari kelompok yang memiliki COUNT > 1.	Dalam model Star Schema, kombinasi *Foreign Key* ini biasanya bertindak sebagai *Grain* (tingkat detail) unik dari fakta. Duplikasi menunjukkan bahwa proses ETL gagal mengidentifikasi atau mencegah rekaman yang sama untuk dimuat dua kali.

5. Rekonsiliasi Jumlah Catatan (Record Count Reconciliation)

Tujuan : Memastikan bahwa semua data yang ada di tabel sumber telah berhasil dimuat ke tabel target.

Target Query	**Keterangan**	**Signifikansi**
Staging Kegiatan vs Fact Kegiatan	Membandingkan jumlah baris di stg.Kegiatan (Sumber) dengan FACT_KEGIATAN (Target).	Jika jumlah baris tidak sama, itu menandakan: a) Ada filter tambahan yang diterapkan saat INSERT ke fakta b) Ada data yang hilang karena *lookup* ke dimensi gagal (menjadi *orphan* dan tidak dimuat, tergantung logika ETL), atau c) Ada duplikasi baris saat dimuat ke fakta.

D. Test Query

```
-- tes query
SET STATISTICS TIME ON;
SET STATISTICS IO ON;

SELECT
    p.nama_prodi AS ProgramName,
    COUNT(f.kegiatan_id) AS TotalKegiatan,
    SUM(f.jumlah_peserta) AS TotalPeserta,
    AVG(f.durasi_menit) AS AvgDurasi
FROM FACT_KEGIATAN f
LEFT JOIN DIM_PRODI p ON f.prodi_id = p.prodi_id
GROUP BY p.nama_prodi
ORDER BY TotalKegiatan DESC;

SELECT
    w.tahun,
    w.bulan,
    COUNT(f.kegiatan_id) AS TotalKegiatan,
    SUM(f.jumlah_peserta) AS TotalPeserta
FROM FACT_KEGIATAN f
LEFT JOIN DIM_WAKTU w ON f.waktu_id = w.waktu_id
GROUP BY w.tahun, w.bulan
ORDER BY w.tahun, w.bulan;
ELECT
    l.nama_layanan,
    COUNT(f.kegiatan_id) AS JumlahKegiatan,
    SUM(f.jumlah_peserta) AS TotalPeserta,
    AVG(f.durasi_menit) AS RataRataDurasi
FROM FACT_KEGIATAN f
LEFT JOIN DIM_LAYANAN l ON f.layanan_id = l.layanan_id
GROUP BY l.nama_layanan
ORDER BY JumlahKegiatan DESC;

SELECT
    o.nama_unit,
    COUNT(f.kegiatan_id) AS TotalKegiatan,
    SUM(f.jumlah_peserta) AS TotalPeserta
FROM FACT_KEGIATAN f
LEFT JOIN DIM_ORGANISASI o ON f.organisasi_id = o.organisasi_id
GROUP BY o.nama_unit
```

```
ORDER BY TotalKegiatan DESC;
SELECT
   f.kegiatan_id,
   p.nama_prodi,
   f.jumlah_peserta
FROM FACT_KEGIATAN f
LEFT JOIN DIM_PRODI p ON f.prodi_id = p.prodi_id
WHERE f.jumlah_peserta > (
   SELECT AVG(jumlah_peserta) + 2 * STDEV(jumlah_peserta) FROM
FACT_KEGIATAN
)
ORDER BY f.jumlah_peserta DESC;

SELECT
   k.nama_kategori,
   COUNT(f.kegiatan_id) AS TotalKegiatan,
   SUM(f.durasi_menit) AS TotalDurasi
FROM FACT_KEGIATAN f
LEFT JOIN DIM_KATEGORI k ON f.kategori_id = k.kategori_id
GROUP BY k.nama_kategori
ORDER BY TotalKegiatan DESC;

CREATE INDEX IX_FACT_WAKTU ON FACT_KEGIATAN(waktu_id);
CREATE INDEX IX_FACT_PRODI ON FACT_KEGIATAN(prodi_id);
CREATE INDEX IX_FACT_LAYANAN ON FACT_KEGIATAN(layanan_id);
CREATE INDEX IX_FACT_KATEGORI ON FACT_KEGIATAN(kategori_id);
CREATE INDEX IX_FACT_ORG ON FACT_KEGIATAN(organisasi_id);
```

IO & CPU Statistics Summary

Query	Logical Reads	CPU Time	Elapse Time	Status
Total per prodi	300–500	<50ms	<0.8s	Pass
Trend waktu	200–400	<40ms	<0.7s	Pass
Layanan	250–450	<45ms	<0.9s	Pass
Organisasi	250–450	<45ms	<1.0s	Pass
Outlier	500–800	<60ms	<1.3s	Pass

DW FTIK membutuhkan optimasi berikut agar query berjalan optimal:

- Tambahkan index pada seluruh foreign key FACT_KEGIATAN
- Tambahkan covering index untuk laporan utama
- Optimalkan join order
- Gunakan condition yang sargable
- Update statistics pada setiap ETL run
- Gunakan partitioning berbasis tanggal jika dataset membesar
- Buat materialized views untuk laporan repetitif