



LEARNING PROGRESS REVIEW

FAST TRACK DATA ENGINEER SCHOLARSHIP
BY
DIGITAL SKOLA



GARAP RENDANG TEAM



- Suhendar Aditya
- Chandra Trio Pamungkas
- Aryo Putra
- Kurniaman Andreas Zega
- Ahmad Fayyadh

GARAP RENDANG



**Grup Andalan Rakitan Pipeline
Record Engineer Ngolah Data
Anti Gagal**



WHAT IS DATA ENGINEERING?



INTRO TO DATA ENGINEER

Data Engineering adalah **praktik merancang dan membangun sistem untuk mengumpulkan, menyimpan, dan menganalisis data dalam skala besar**. Tujuan utamanya adalah **membuat data lebih berguna** dan mudah diakses oleh konsumen data

Common Data Engineering Terms

ETL	Data Integrity Checks			Big Data Processing
Extract	Null	Uniqueness		MPP
Transform	Anomaly	Aggregate		Map Reduce
Load	Category			
Streaming	Data warehouse			Data Modeling
Batch processing	Data lake			Data lakehouse

Source: pragmaticengineer.com



ETL VS ELT

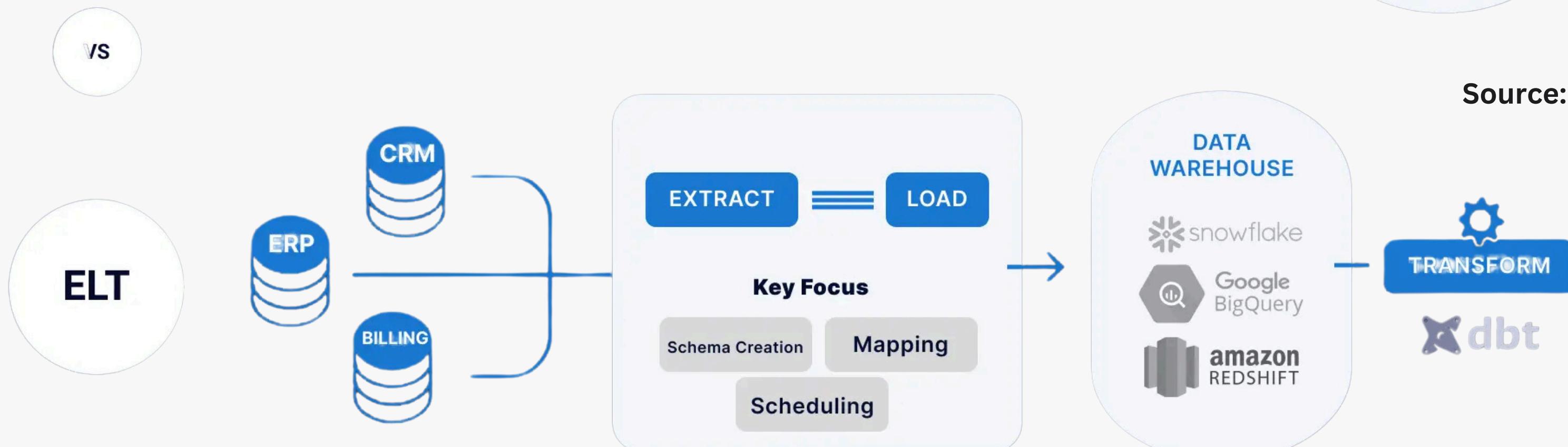
Extract: Mengambil data dari sumber.

Transform: Mengubah data ke format yang sesuai.

Load: Memasukkan data ke dalam penyimpanan seperti gudang data



Source: Sriim



GRUP
R&D



PYTHON





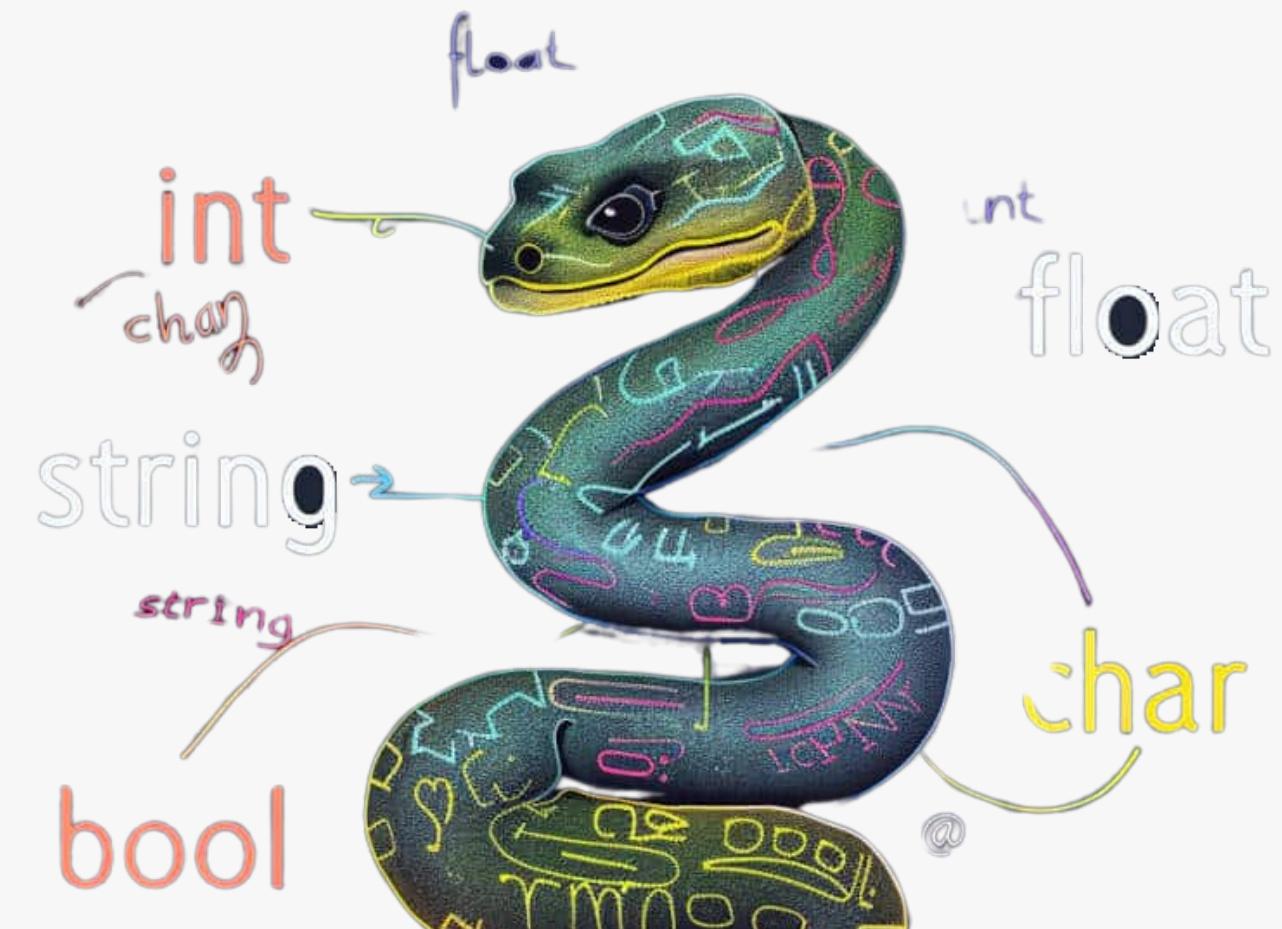
PYTHON

Variable adalah **tempat yang diberi nama untuk menyimpan suatu data**. Cara penulisan nama variable di python ada dua yaitu:

- **Camel Case**: namaVariable.
- **Snake Case**: nama_variable

Tipe Data

- **Tipe Data Hanya Bisa Menyimpan Satu Data**
 - **String**: "saya", "kamu", "18"
 - **Integer**: 18, 1, -1
 - **Float**: 18.0, 3.14
 - **Boolean**: True dan False
- **Tipe Data Bisa Menyimpan Lebih dari Satu Data**
 - **List**: ["sepak bola", "basket", "Berenang"]
 - **Tuple**: ("sepak bola", "basket", "Berenang")
 - **Dictionary**: {"nama": "Jono", "umur": 18, "menikah": False}





Conditional Statement & Looping

Conditional Statement

Kode program yang dijalankan ketika suatu kondisi telah terpenuhi

- **if Condition**
- **elif (else-if) condition**
- **else condition**

Conditional statement **dijalankan secara berutan** dari **if statement**, jika kondisi di **if** tidak terpenuhi dilanjutkan ke **elif statement** dan jika tidak terpenuhi juga langsung ke **else statement**.

Looping

Kode program yang dijalankan secara berulang-ulang sampai suatu kondisi terpenuhi.

- **for Looping:** Jumlah pengulangannya diketahui
- **while Looping:** Jumlah pengulangannya tidak diketahui



Function & Import Statements

Function tersusun dan dapat digunakan berkali-kali hanya dengan memanggil nama fungsinya. Membuat fungsi di python menggunakan kata kunci **def** diikuti nama fungsinya dan diakhiri kurung buka kurung tutup dan titik dua.

```
def nama_fungsi():
    print("hello from nama_fungsi")
```

Modul berisi kode program yang mengimplementasikan konsep-konsep pemrograman python dan dapat digunakan dalam berbagai erogram. **Modul** dapat digunakan dengan menuliskan kode **import statement** pada program yang ingin menggunakannya

```
import math
print(math.sqrt(16)) # menggunakan method .sqrt() dari modul math
```

Return and Parameter Function dapat menerima nilai dari luar dengan menggunakan parameter yang bertindak sebagai variabel di dalam fungsi. Hasil yang didapatkan dari menjalankan perogram yang ada di dalam fungsi dapat dikembalikan ke suatu variabel dengan menggunakan **return statement**.

```
def tambah(param_1, param_2):
    return param_1 + param_2
hasil = tambah(5, 2)
```



OOP

Object Oriented Programming cara penulisan kode di dalam python yang membuatnya menjadi modular

- **Class:** Tempat mendefinisikan attribute dan method apa saja yang akan digunakan oleh suatu object.
- **Object:** Representasi dari class yang digunakan dengan attribute dan method yang telah diberi nilai.
- **Attribute:** Variabel yang berada di dalam class yang memberikan informasi setiap objek yang menggunakan class tersebut.
- **Method:** Fungsi yang berada di dalam class yang digunakan untuk melakukan operasi tertentu oleh suatu objek.

class NamaClass:

```
def __init__(self, nilai):
    self.attribute = nilai
def method(self):
    print(f"Nilai: {self.attribute}")
```

```
object_1 = NamaClass(nilai=5)
```

Menulis kode dengan menerapkan OOP bisa menggunakan beberapa konsep:

- **Encapsulation:** Penulisan kode class yang isinya menggabungkan attribute dan method menjadi satu.
- **Inheritance:** Setiap class yang telah dibuat bisa diwarisi ke class lainnya yang menjadi subclass sehingga bisa memiliki attribute dan method yang sama dari class atasnya.
- **Polymorphism:** Kita bisa membuat method dengan nama yang sama di subclass tetapi melakukan tindakan yang berbeda.
- **Abstraction:** Merupakan class yang dibuat menjadi superclass yang berisi method-method yang tidak bisa dirubah oleh subclass yang mewarisinya.





STRUCTURED QUERY LANGUAGE (SQL)



Apa itu RDBMS SQL?

RDBMS (Relational Database Management System):

- Sistem manajemen basis data berbasis relasi.
- Data disimpan dalam bentuk tabel (rows & columns).
- Contoh: MySQL, PostgreSQL, SQL Server.

Ciri Utama:

- Mendukung SQL.
- Relasi antar tabel menggunakan Primary Key dan Foreign Key.
- Mendukung transaksi (ACID).

Rank			DBMS
Nov 2024	Oct 2024	Nov 2023	
1.	1.	1.	Oracle +
2.	2.	2.	MySQL +
3.	3.	3.	Microsoft SQL Server
4.	4.	4.	PostgreSQL +
5.	5.	8.	Snowflake +
6.	6.	5.	IBM Db2
7.	7.	6.	SQLite
8.	8.	7.	Microsoft Access
9.	9.	11.	Databricks +
10.	10.	9.	MariaDB +



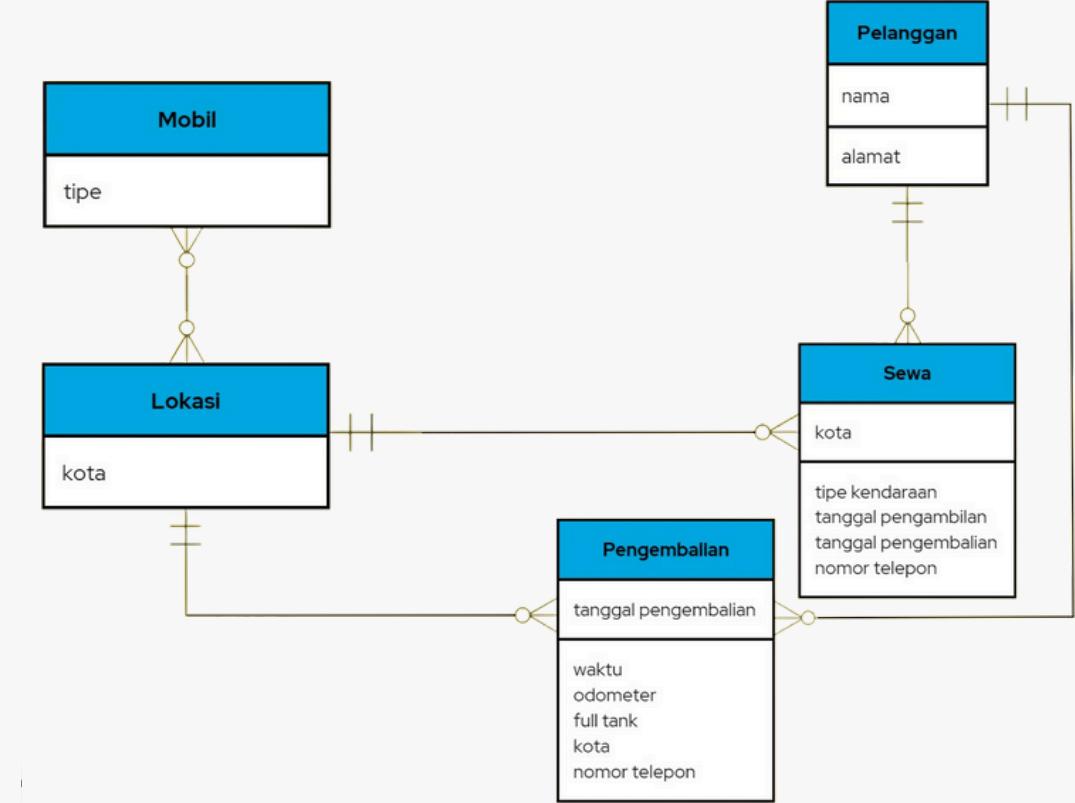
Entity-Relationship Diagram

ERD (Entity-Relationship Diagram):

- Diagram untuk memodelkan data.
- Pendekatan Top-Down: Desain dimulai dengan entitas, hubungan, dan atribut.
- Pendekatan Bottom-Up: Desain dimulai dengan atribut yang dinormalisasi menjadi relasi.

Elemen Utama:

- Entity: Tabel (contoh: Karyawan, Departemen).
- Attributes: Kolom (contoh: ID, Nama).
- Relationships: Hubungan antar tabel (contoh: Karyawan → Departemen).



Sur

Relationship

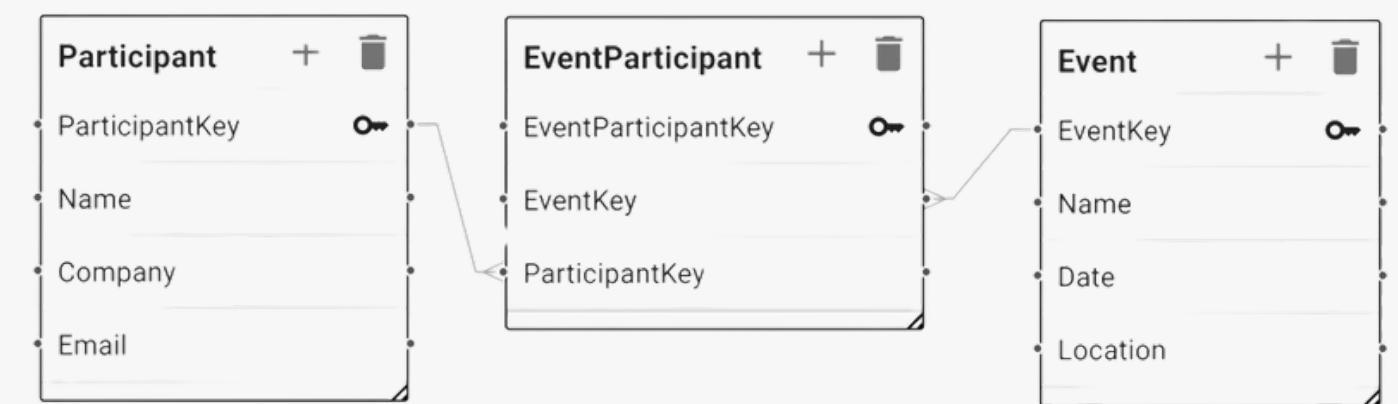
1:1 (One-to-One): Setiap entitas di tabel pertama berhubungan dengan satu entitas di tabel kedua. Contoh: User → UserProfile.



1:N (One-to-Many): Satu entitas di tabel pertama dapat berhubungan dengan banyak entitas di tabel kedua. Contoh: Karyawan → Departemen.



M:N (Many-to-Many): Banyak entitas di tabel pertama berhubungan dengan banyak entitas di tabel kedua. Contoh: Siswa → MataPelajaran.



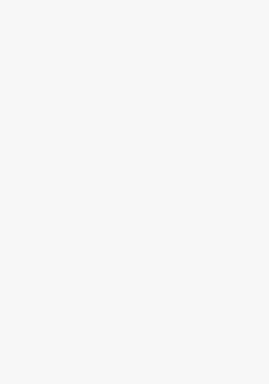


PK, Auto Increment & FK

Primary Key (PK):

- Primary Key: kolom unik yang mengidentifikasi setiap baris dalam tabel dan tidak boleh bernilai NULL
- Auto Increment: fitur yang otomatis menambahkan nilai berurutan untuk Primary Key di MySQL.

```
● ● ●  
CREATE TABLE Produk (  
    ID INT PRIMARY KEY AUTO_INCREMENT,  
    Nama VARCHAR(50)  
);
```



PK, Auto Increment & FK



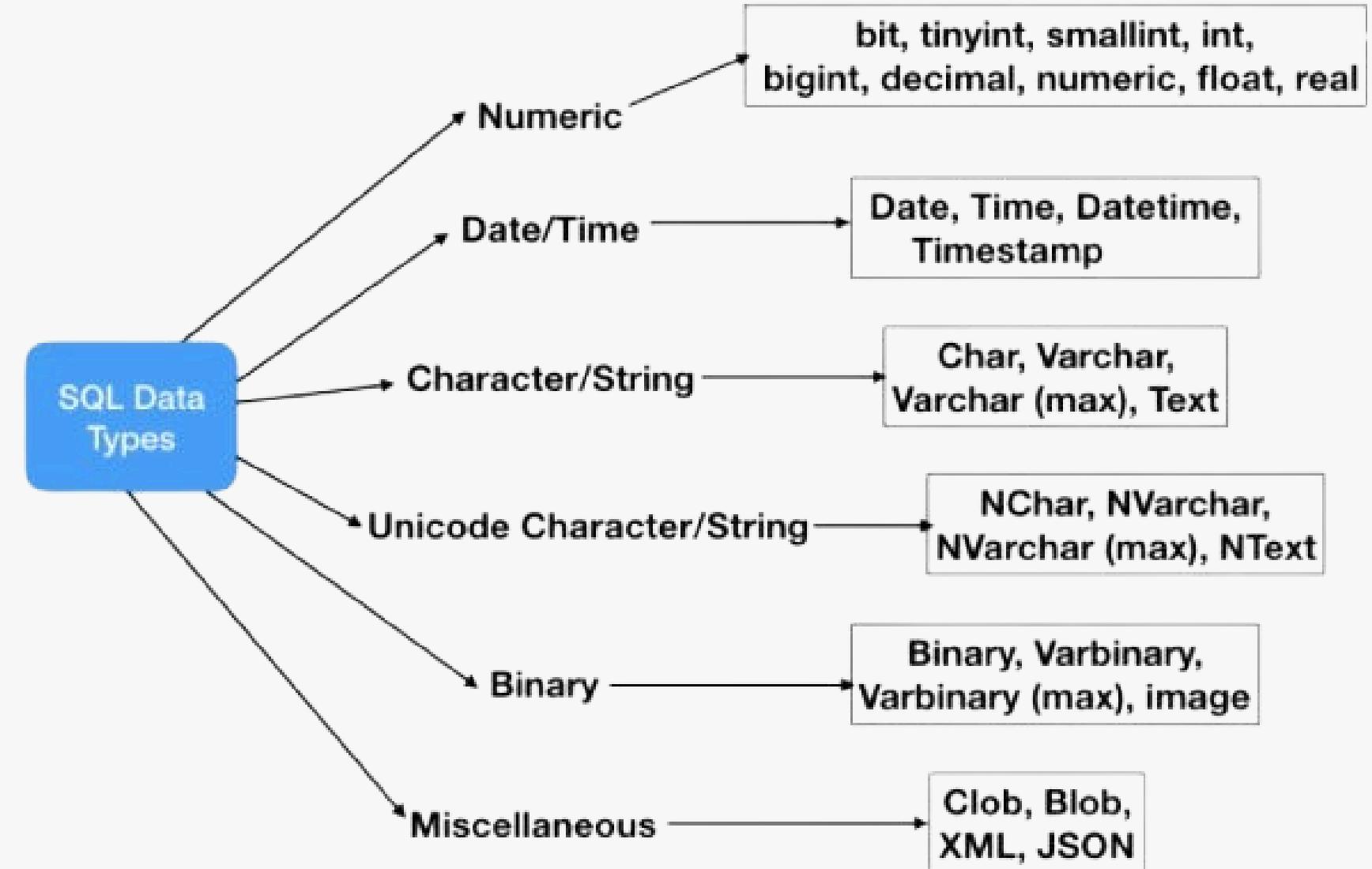
Foreign Key (FK):

- Kolom yang mereferensikan Primary Key (PK) di tabel lain untuk membangun relasi antar tabel.

```
● ● ●  
ALTER TABLE Orders  
ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```



Data Types





DDL

DDL

CREATE

ALTER

DROP

TRUNCATE

COMMENT

RENAME

- Command DDL (Data Definition Language) merupakan command yang berisi perintah-perintah untuk **mendefinisikan skema** di database, seperti untuk membuat, memodifikasi, serta menghapus struktur database. Perintah ini **biasanya akan banyak digunakan saat awal pembuatan database**, dan jarang digunakan oleh pengguna umum yang seharusnya mengakses database melalui aplikasi.

DDL



- **CREATE:** Digunakan untuk membuat database atau objeknya (seperti tabel, indeks, function, views, store procedure, dan trigger).
- **DROP:** Dapat digunakan untuk menghapus objek dari database.
- **ALTER:** Perintah ini digunakan untuk mengubah struktur database.



```
CREATE TABLE Karyawan (
    ID INT PRIMARY KEY, Nama VARCHAR(50)
);
```



```
DROP TABLE Karyawan;
```



```
ALTER TABLE Karyawan ADD Gaji
DECIMAL(10, 2);
```



DDL

- **TRUNCATE:** Ini digunakan untuk menghapus semua record dari tabel, termasuk semua space yang dialokasikan untuk semua record yang dihapus.
- **COMMENT:** Perintah ini digunakan untuk menambahkan komentar ke kamus data.
- **RENAME:** Perintah ini digunakan untuk mengganti nama objek yang ada di database.

```
● ● ●  
TRUNCATE TABLE Karyawan;
```

```
● ● ●  
COMMENT ON TABLE Karyawan IS 'Tabel data karyawan';
```

```
● ● ●  
ALTER TABLE Karyawan RENAME TO Pegawai;
```



DML

DML

SELECT

INSERT

UPDATE

DELETE

MERGE

CALL

EXPLAIN PLAN

LOCK TABLE

- Command DML (Data Manipulation Language) merupakan command yang berhubungan dengan **proses manipulasi data** yang ada di database, yang umumnya mencakup hampir sebagian besar statement SQL.



DML

- **Select** : perintah yang digunakan untuk menampilkan record pada sebuah tabel

```
● ● ●  
SELECT * FROM Karyawan;
```

- **Insert** : perintah yang digunakan untuk menambahkan record baru pada tabel

```
● ● ●  
INSERT INTO Karyawan (ID, Nama, Gaji)  
VALUES (1, 'Andi', 5000000);
```



DML

- **Update** : perintah yang digunakan untuk mengubah data



```
UPDATE Karyawan SET Gaji = 6000000  
WHERE ID = 1;
```

- **Delete** : digunakan untuk menghapus data



```
DELETE FROM Karyawan WHERE ID = 1;
```



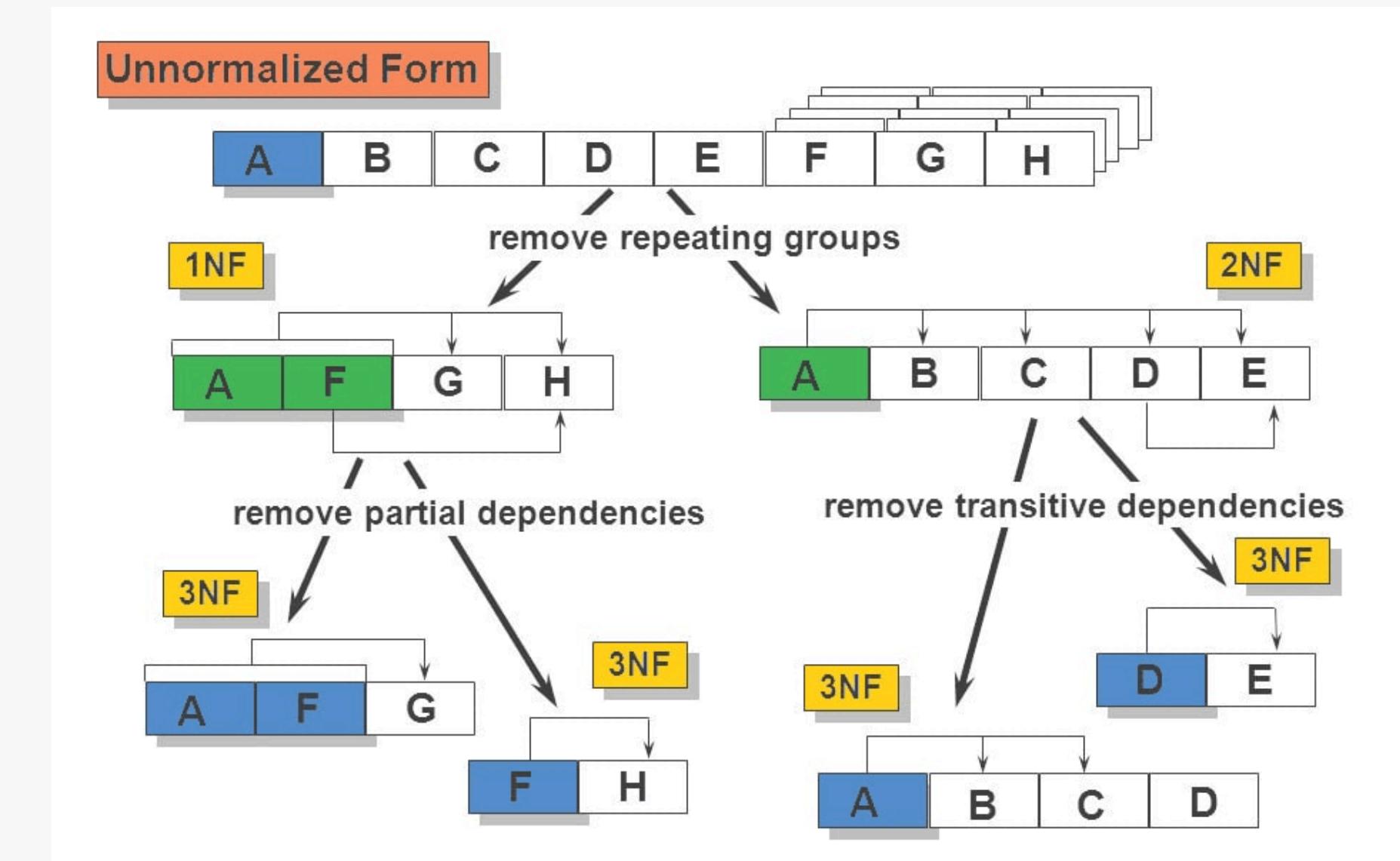
Database Normalization

Tujuan:

- Mengurangi redundansi data.
- Meningkatkan integritas data.

Tingkatan:

- 1NF: Hilangkan data berulang.
- 2NF: Hilangkan dependensi parsial.
- 3NF: Hilangkan dependensi transitif.





Aggregate, GROUP BY, ORDER BY, LIKE

Aggregate Functions:

SUM (jumlah total nilai dalam kolom),
AVG (ata-rata nilai dalam kolom),
MAX (nilai tertinggi dalam kolom),
MIN (nilai terendah dalam kolom),
COUNT (jumlah baris dalam kolom atau tabel).



```
SELECT AVG(Gaji) FROM Karyawan;
```

GROUP BY:

mengelompokkan data berdasarkan kolom tertentu untuk melakukan agregasi.



```
SELECT Departemen, COUNT(*)  
FROM Karyawan GROUP BY Departemen;
```

Aggregate, GROUP BY, ORDER BY, LIKE



ORDER BY

Mengurutkan hasil query berdasarkan satu atau lebih kolom, baik secara ascending (ASC) atau descending (DESC)



```
SELECT * FROM Karyawan ORDER BY Gaji  
DESC
```

LIKE :

Digunakan untuk mencari data yang cocok dengan pola tertentu, sering kali dengan wildcard (%) untuk banyak karakter, _ untuk satu karakter).



```
SELECT * FROM Karyawan WHERE Nama LIKE 'A%';
```



JOIN

Jenis-Jenis JOIN: XXX

- **INNER JOIN:** Menampilkan data yang cocok di kedua tabel.
- **LEFT JOIN:** Semua data dari tabel kiri + cocok dari kanan.
- **RIGHT JOIN:** Semua data dari tabel kanan + cocok dari kiri.
- **FULL JOIN:** Gabungan semua data.



```
SELECT K.Nama, D.Nama  
FROM Karyawan K  
INNER JOIN Departemen D  
ON K.DepartemenID = D.ID;
```



Subquery dan CTE

Subquery:

Query yang berada di dalam query lain, digunakan untuk menghasilkan nilai atau hasil yang digunakan oleh query utama.

CTE (Common Table Expression):

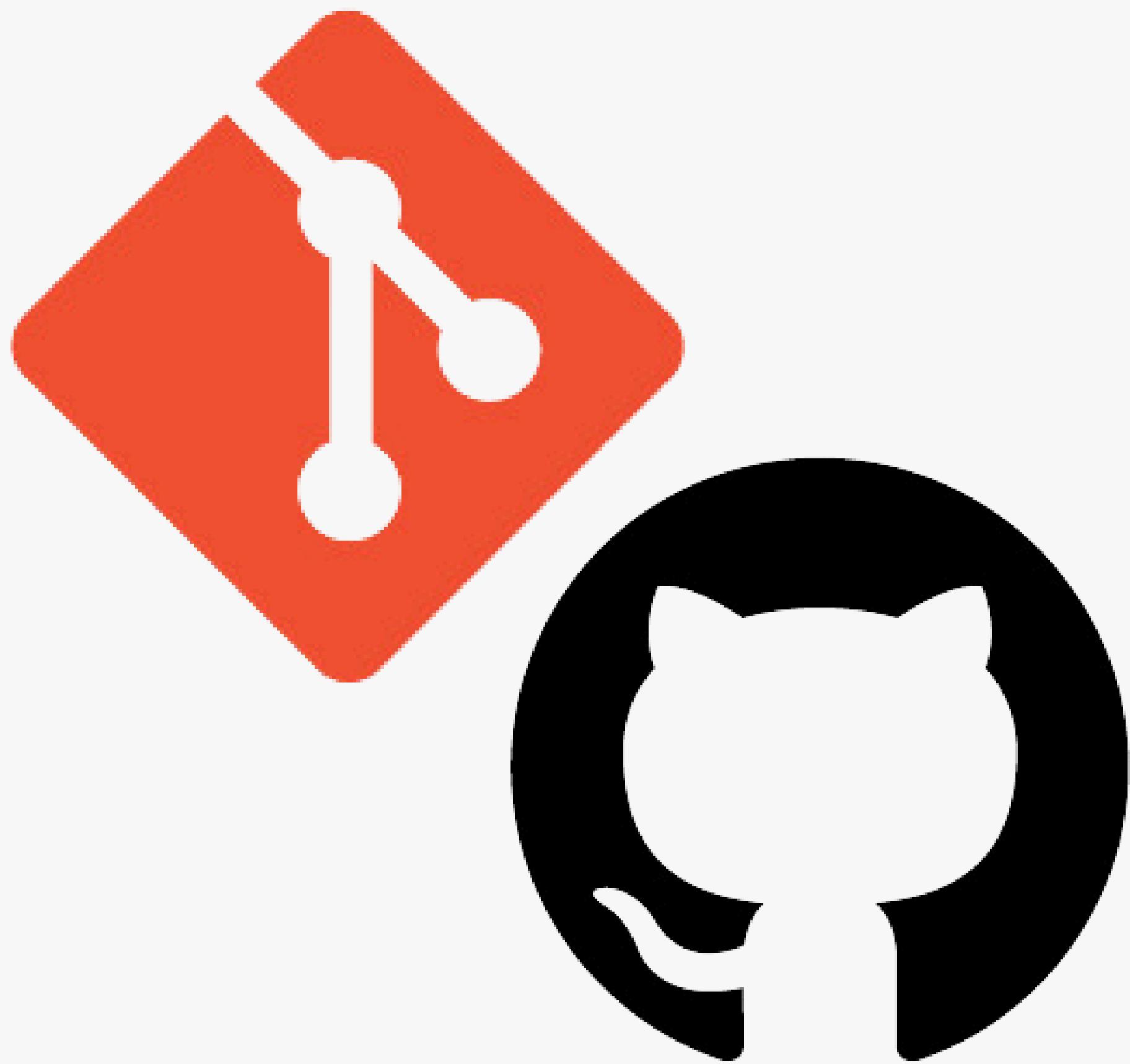
Nama sementara untuk hasil query yang dapat digunakan dalam query utama, sering kali digunakan untuk mempermudah pembacaan dan pemeliharaan kode.



```
SELECT Nama FROM Karyawan  
WHERE Gaji > (SELECT AVG(Gaji) FROM Karyawan);
```

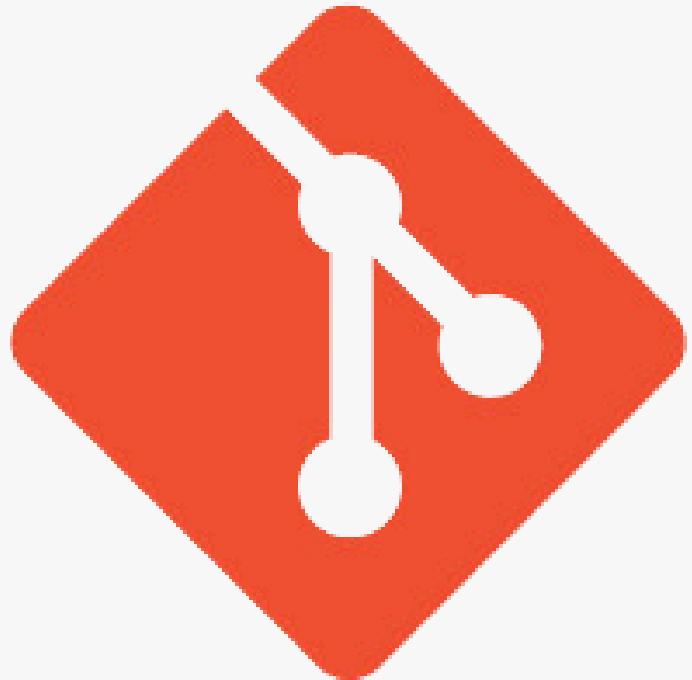


```
WITH CTE_Gaji AS (  
    SELECT Nama, Gaji FROM Karyawan WHERE Gaji >  
    5000000  
)  
SELECT * FROM CTE_Gaji;
```



GIT & GITHUB





GIT & GITHUB



Aspek	Git	GitHub
Definisi	Sistem kontrol versi terdistribusi.	Platform hosting repositori Git secara online.
Lokasi	Repositori disimpan secara lokal.	Repositori disimpan di server GitHub.
Penyimpanan		
Fungsi Utama	Mengelola versi kode secara lokal.	Menyediakan layanan kolaborasi dan hosting repositori.
Akses	Akses melalui terminal di komputer lokal.	Akses melalui web browser atau aplikasi GitHub.
Kolaborasi	Kolaborasi manual dengan berbagi repositori.	Memudahkan kolaborasi melalui pull request dan issues.



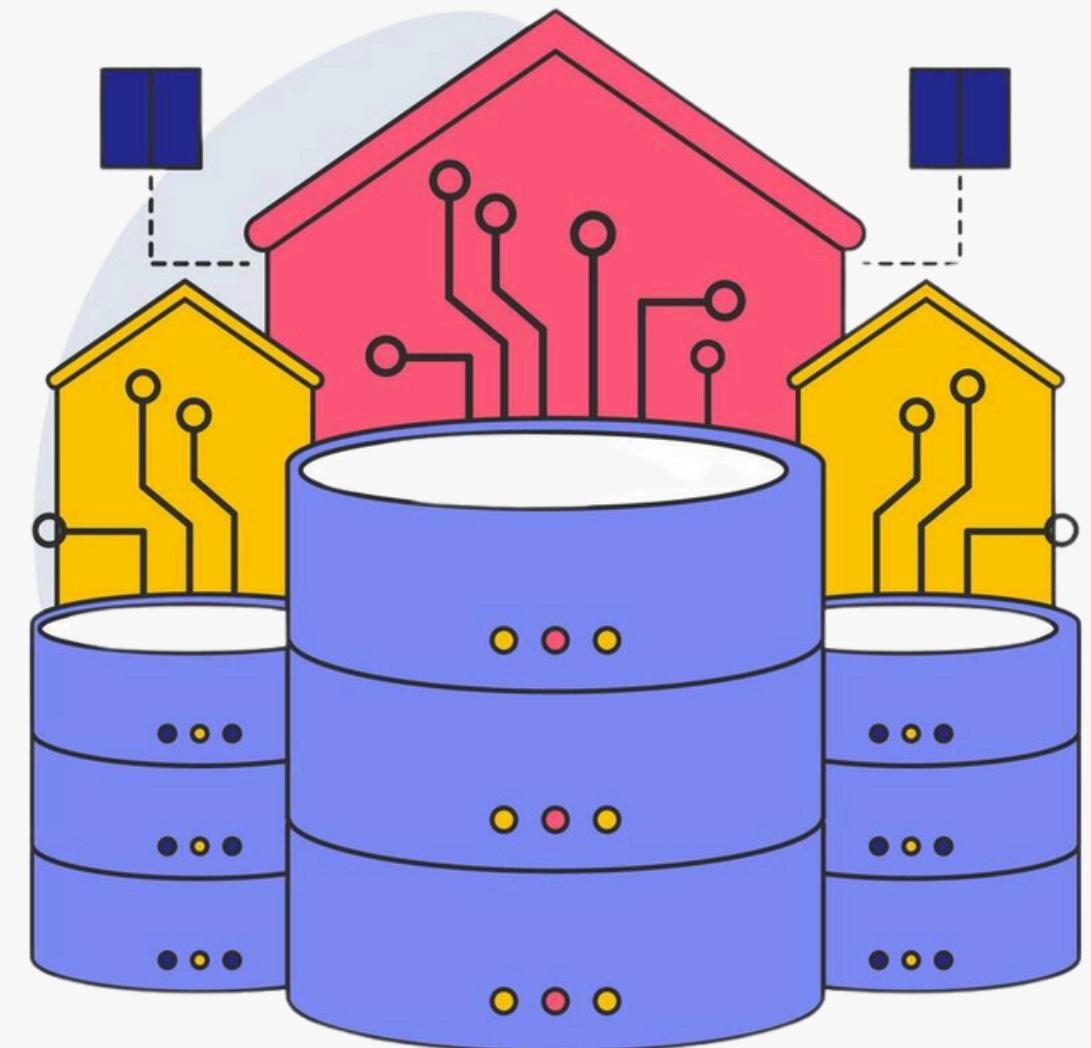
Terminologi

Terminologi	Deskripsi
Repository (Repo)	Tempat penyimpanan proyek yang berisi semua file dan riwayat perubahan.
Commit	Tindakan menyimpan perubahan ke dalam repositori dengan pesan deskriptif.
Branch	Salinan terpisah dari repositori utama (master/main) untuk pengembangan terisolasi.
Merge	Menggabungkan perubahan dari satu branch ke branch lainnya.
Push	Mengirim perubahan dari branch lokal ke repositori remote.
Clone	Membuat salinan lokal dari repositori yang ada di server atau repositori jarak jauh.
Pull	Mengambil perubahan terbaru dari repositori remote dan menggabungkannya dengan branch lokal.
Staging Area	Area sementara untuk menandai file yang telah diubah sebelum di-commit.

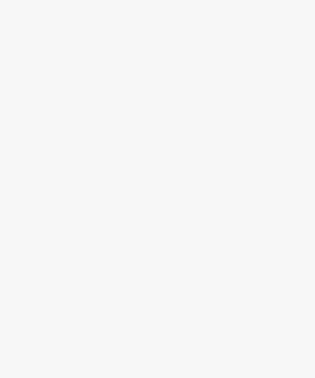


Perintah Dasar

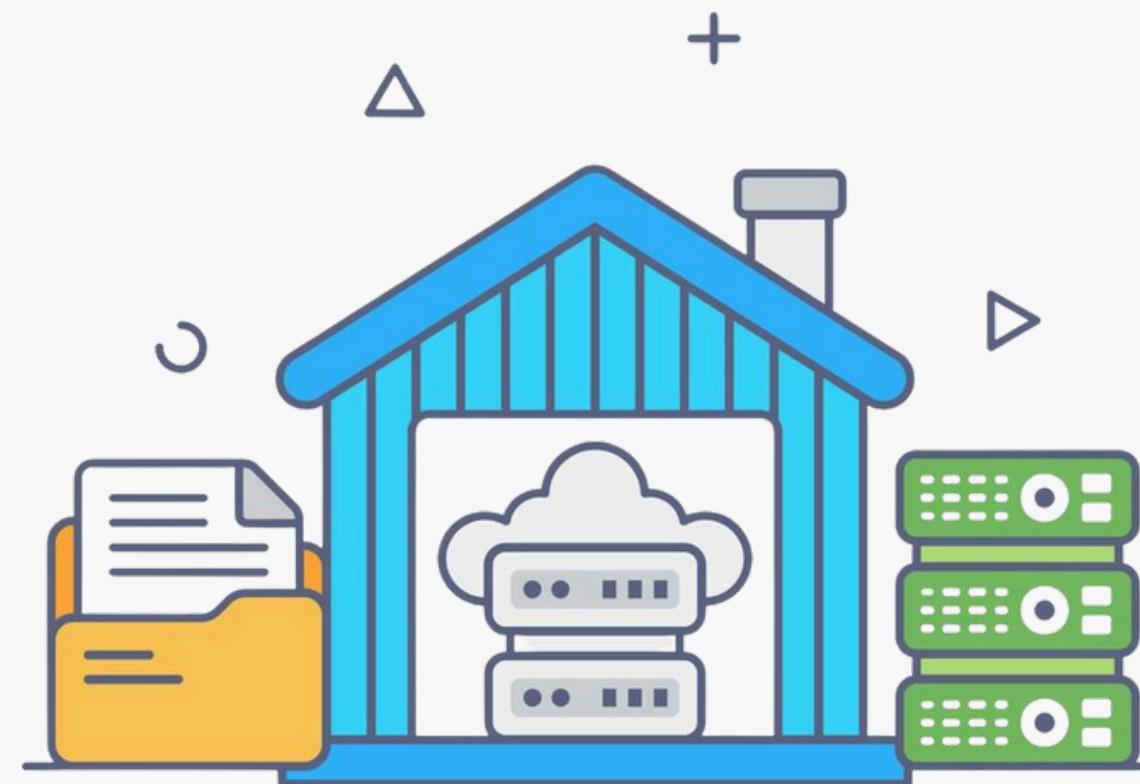
Perintah	Deskripsi
<code>git init</code>	Membuat repositori Git baru di direktori lokal.
<code>git clone <url></code>	Menyalin repositori dari remote ke lokal.
<code>git add <file></code>	Menambahkan perubahan file ke staging area.
<code>git commit -m "<message>"</code>	Menyimpan perubahan di repositori dengan pesan deskriptif.
<code>git status</code>	Menampilkan status perubahan yang belum di-commit.
<code>git push</code>	Mengirim perubahan lokal ke repositori remote.
<code>git pull</code>	Mengambil dan menggabungkan perubahan dari repositori remote.
<code>git branch</code>	Menampilkan semua branch yang ada di repositori.
<code>git checkout <branch></code>	Berpindah ke branch tertentu.
<code>git merge <branch></code>	Menggabungkan perubahan dari branch lain ke branch saat ini.



DATA WAREHOUSE

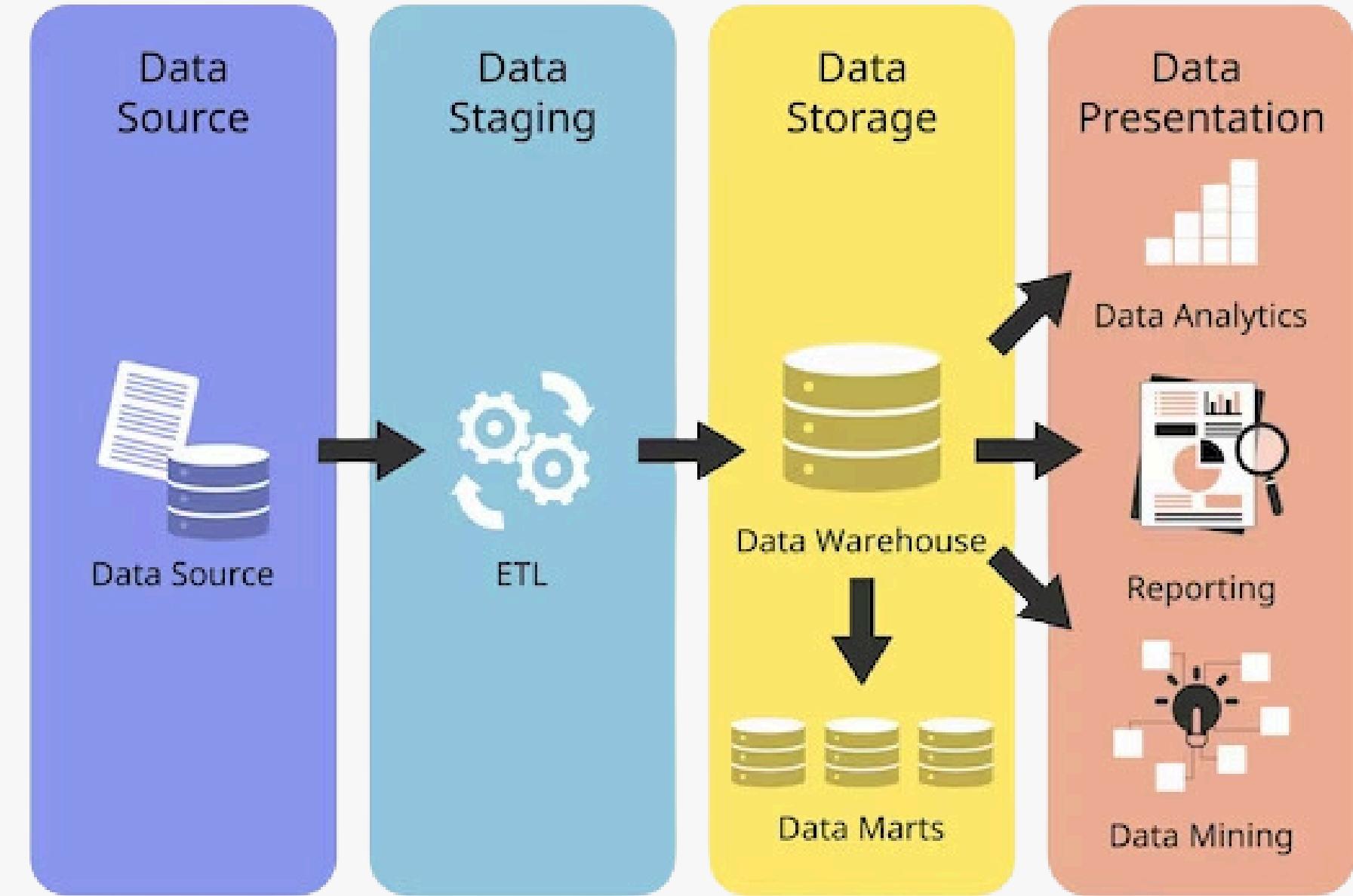


DATA WAREHOUSE



Source: Vecteezy

Data warehouse adalah sebuah **repository terintegrasi dari data** yang berasal dari berbagai sumber sistem operasional, yang **dirancang untuk mendukung proses pengambilan keputusan**.



Source: Freepik

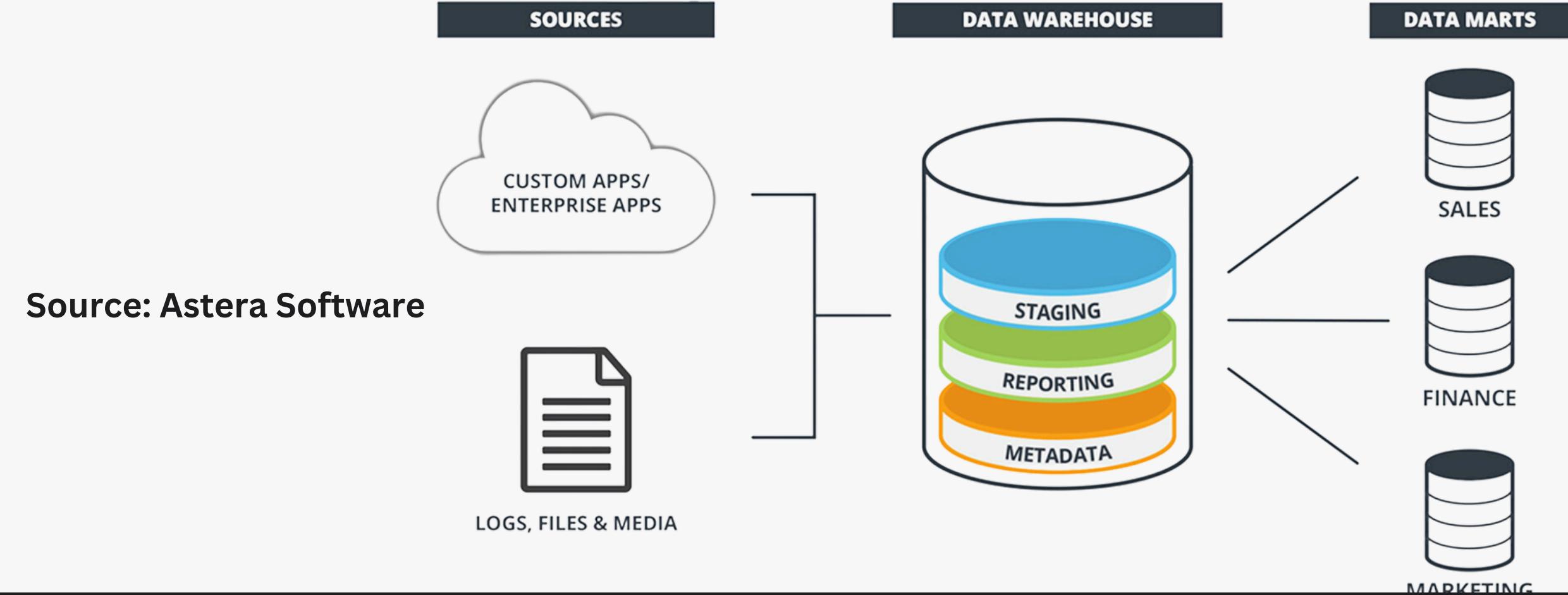
Karakteristik:

- **Terintegrasi:** Data dari berbagai sumber digabungkan menjadi satu kesatuan.
- **Historis:** Menyimpan data dari waktu ke waktu.
- **Subjek-oriented:** Berfokus pada suatu subjek bisnis tertentu (misal, penjualan, pelanggan).
- **Non-volatile:** Data tidak sering berubah setelah dimuat.



Kenapa DWH penting?

- **Mendukung Pengambilan Keputusan:** Membantu perusahaan membuat keputusan yang lebih baik berdasarkan data yang akurat dan komprehensif.
- **Meningkatkan Efisiensi:** Mengotomatiskan proses pengumpulan dan analisis data.
- **Mengungkap Tren:** Memungkinkan perusahaan mengidentifikasi tren dan pola dalam data yang sulit ditemukan dalam sistem operasional.
- **Meningkatkan Kualitas Layanan:** Membantu perusahaan memahami kebutuhan pelanggan dengan lebih baik.



Source: Astera Software

Operational Data Store

vs

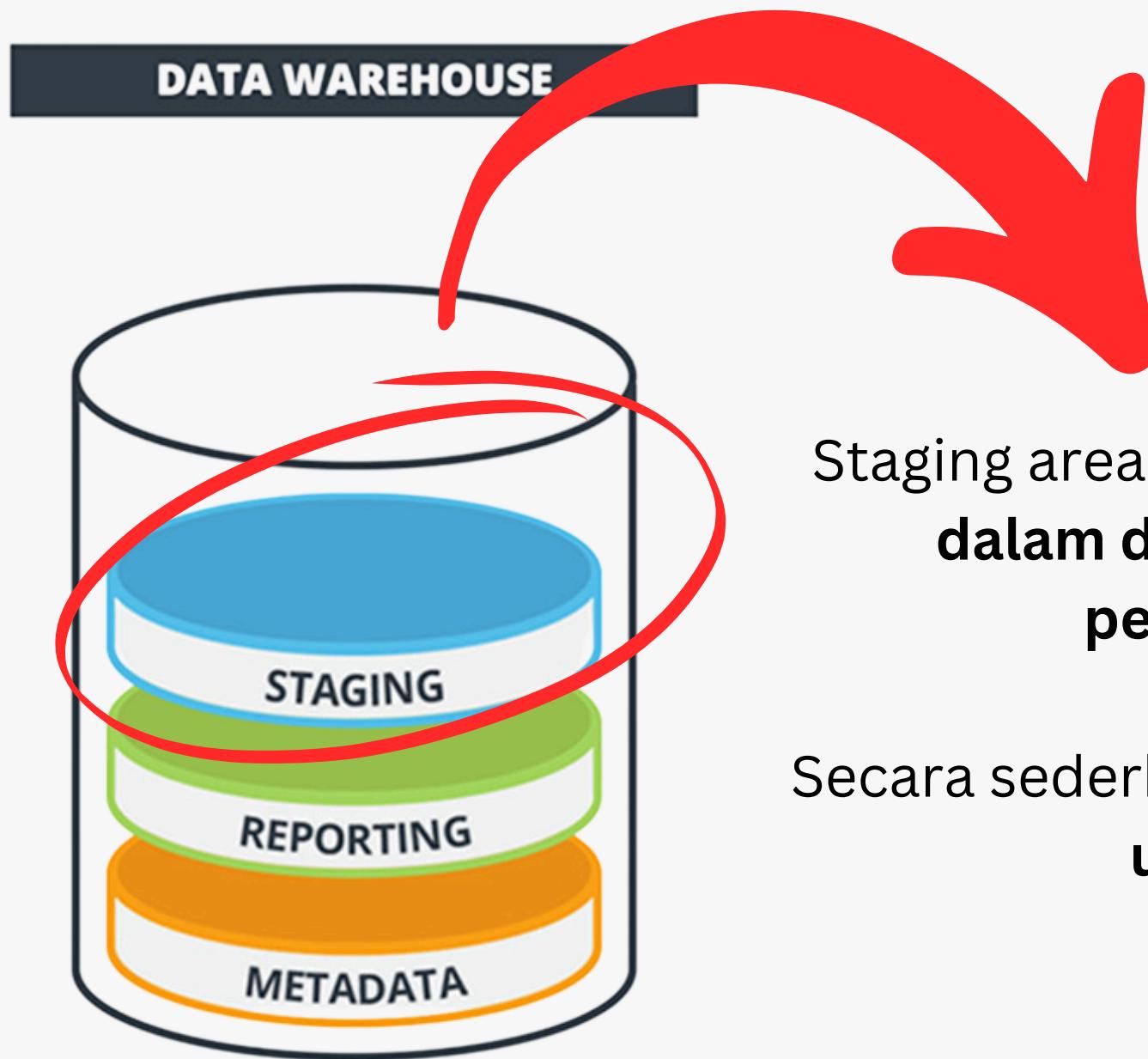
Data Warehouse



Aspek	ODS	DWH
Tujuan	Mendukung operasi harian (real-time).	Mendukung analisis data dan pengambilan keputusan.
Jenis Data	Data terkini, detail, dan sering di-update.	Data historis yang telah dikonsolidasi.
Waktu	Jangka pendek (sementara).	Jangka panjang.
Penyimpanan		
Struktur Data	Minim transformasi, langsung dari sumber.	Terstruktur, melalui proses ETL.
Pengguna Utama	Tim operasional (transaksi harian).	Manajemen dan tim analitik (laporan & tren).



Staging



Staging area **berperan sebagai tempat transit sementara** bagi data **sebelum disimpan dalam data warehouse**. Di sini, data mentah akan melalui serangkaian **proses pembersihan dan transformasi** untuk meningkatkan kualitasnya.

Secara sederhana, **staging area adalah area persiapan data sebelum data tersebut siap untuk dianalisis** dan digunakan untuk pengambilan keputusan.

Source: Astera Software



Reporting



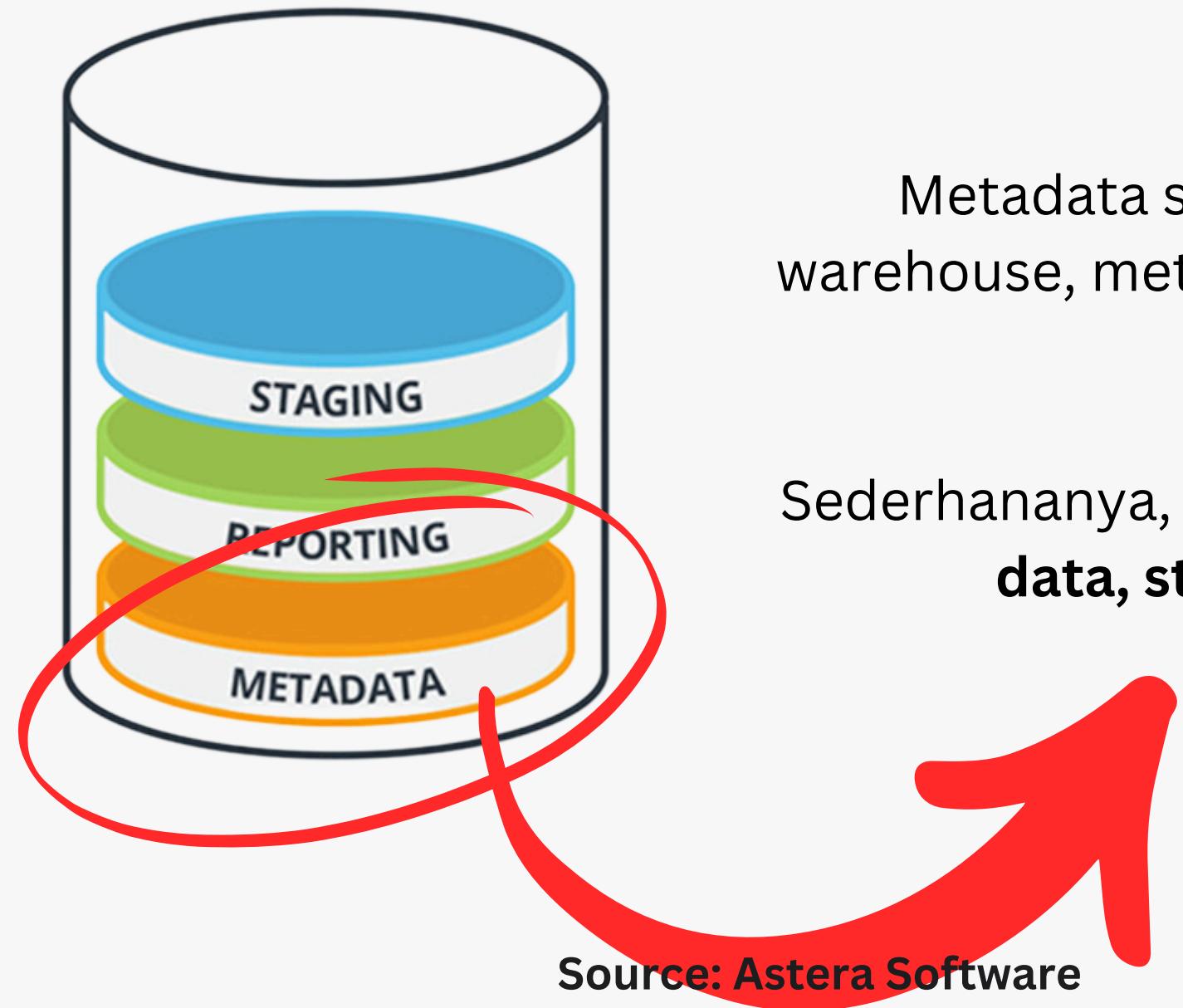
Reporting adalah **proses pengambilan, pengolahan, dan penyajian data** yang tersimpan dalam data warehouse menjadi informasi yang mudah dipahami dan bermanfaat untuk pengambilan keputusan.

Bayangkan reporting sebagai jendela yang memungkinkan kita mengintip ke dalam data warehouse dan **melihat pola, tren, serta insight berharga yang tersembunyi di dalamnya**.

Source: Astera Software



DATA WAREHOUSE



Metadata

Metadata seringkali disebut sebagai "data tentang data". Dalam konteks data warehouse, metadata adalah **informasi yang mendeskripsikan data yang tersimpan di dalam data warehouse itu sendiri**.

Sederhananya, metadata **memberikan informasi tambahan tentang data, seperti asal data, struktur data, format data, dan makna dari setiap elemen data**.

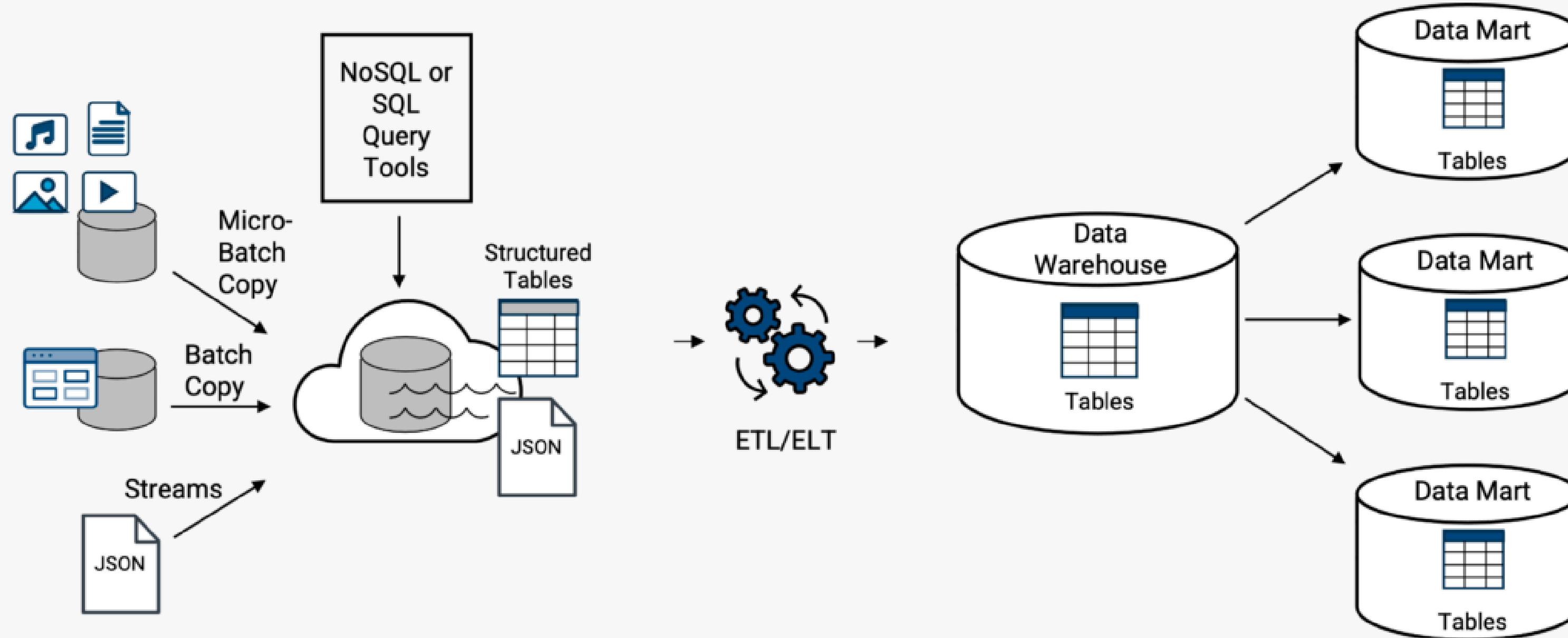


Data Lake vs Data Warehouse vs Data Mart

Aspek	Data Lake	Data Warehouse	Data Mart
Tujuan	Penyimpanan semua jenis data mentah.	Analisis data terstruktur untuk keputusan bisnis.	Analisis data spesifik untuk departemen tertentu.
Jenis Data	Terstruktur, semi-terstruktur, tidak terstruktur.	Terstruktur dan historis.	Terstruktur, subset dari Data Warehouse.
Pengguna Utama	Data engineer, data scientist.	Analis bisnis dan manajemen.	Tim departemen tertentu (misalnya, pemasaran).
Volume Data	Sangat besar, tanpa batasan format.	Sedang hingga besar, terorganisir.	Kecil hingga sedang, fokus pada kebutuhan spesifik.
Kecepatan Akses	Lambat tanpa indeksasi.	Cepat untuk query analitik.	Sangat cepat karena ruang lingkup kecil.



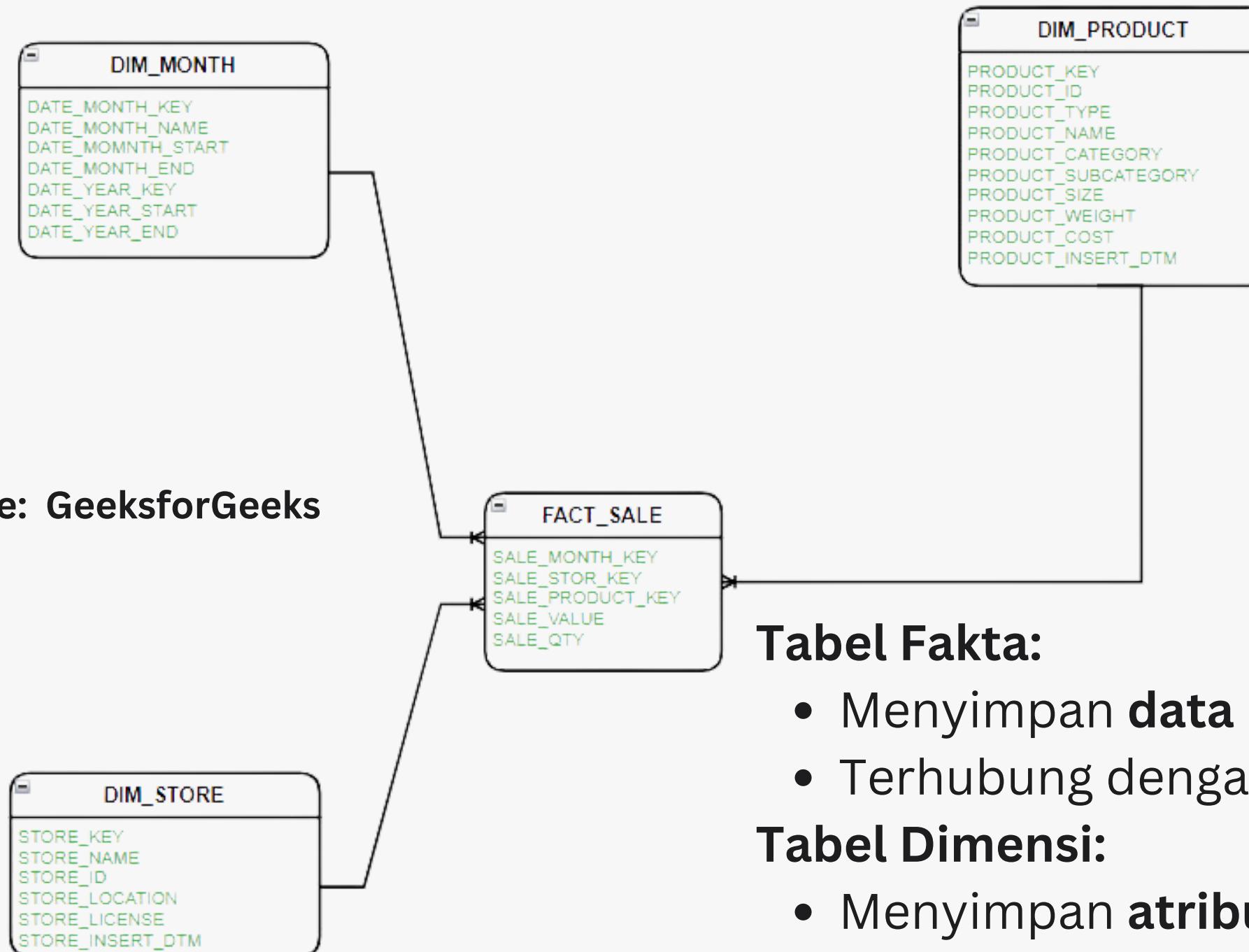
Kurang lebih, begini beda dan alurnya..



Source: SnapLogic



DIMENSIONAL MODELLING



Dimensional modeling adalah pendekatan desain data warehouse yang mengorganisasi data ke dalam **tabel fakta dan dimensi**.

Source: GeeksforGeeks

Tabel Fakta:

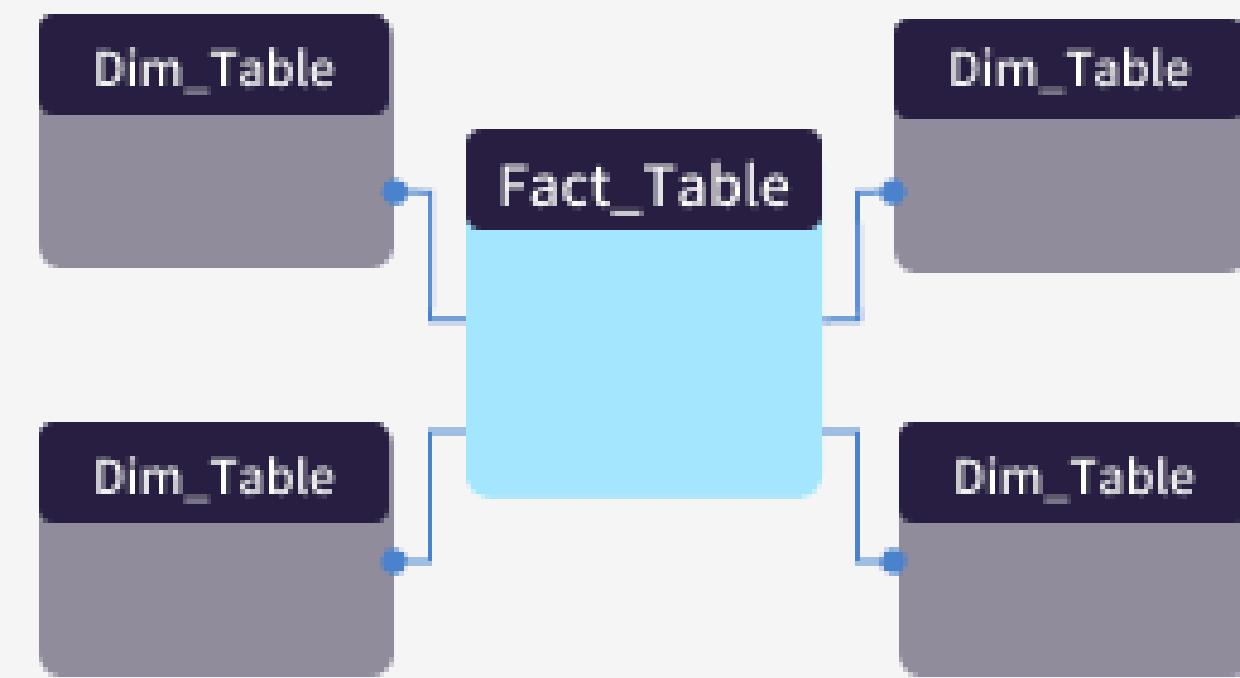
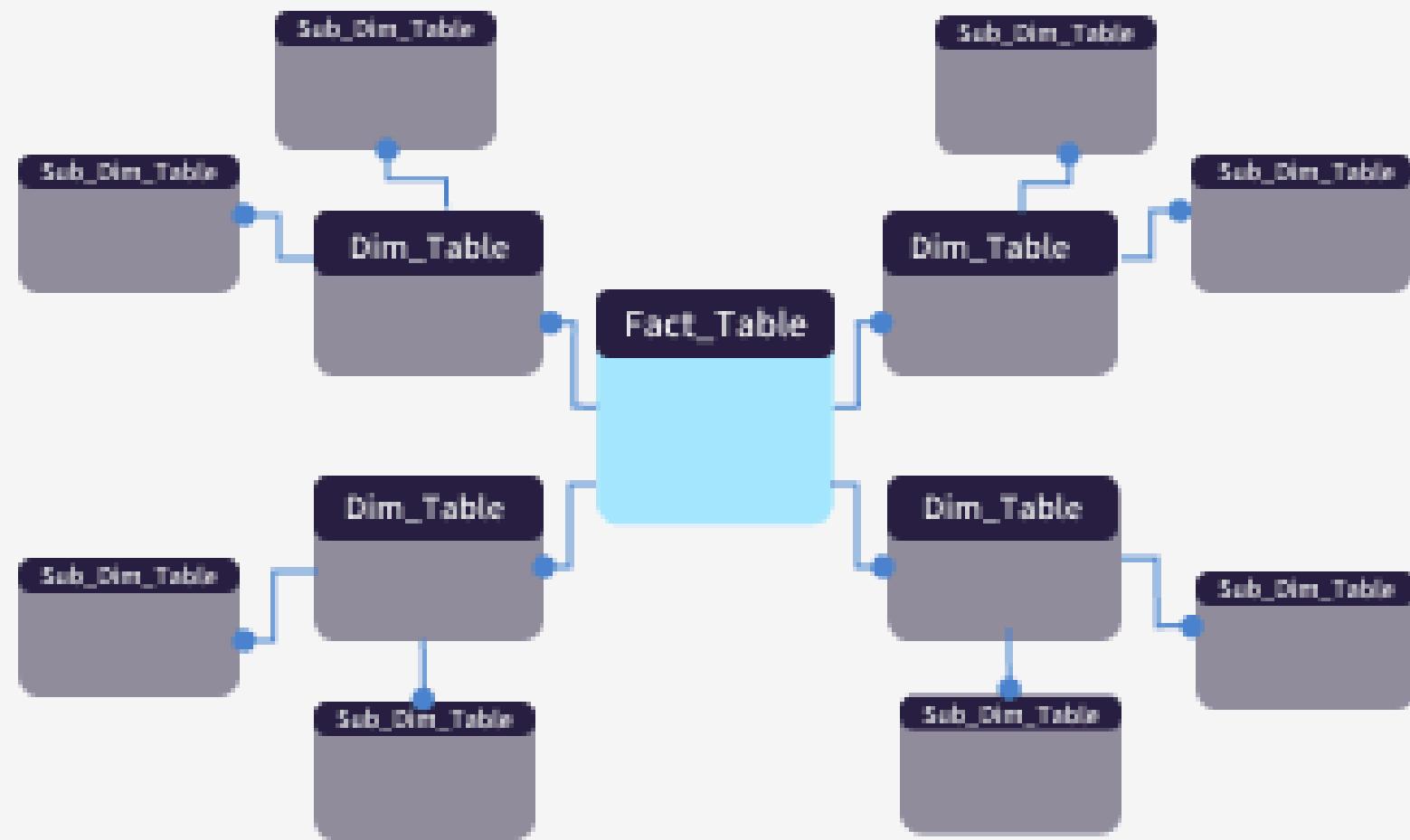
- Menyimpan **data numerik** yang ingin dianalisis (misal: penjualan, biaya).
- Terhubung dengan tabel dimensi melalui kunci asing (FK).

Tabel Dimensi:

- Menyimpan **atribut deskriptif** (misal: waktu, produk, pelanggan).
- Memberikan konteks pada data dalam tabel fakta.



Snowflake vs Star Schema



dapet bedanya?

Source: Integrate.io



Begini singkatnya..

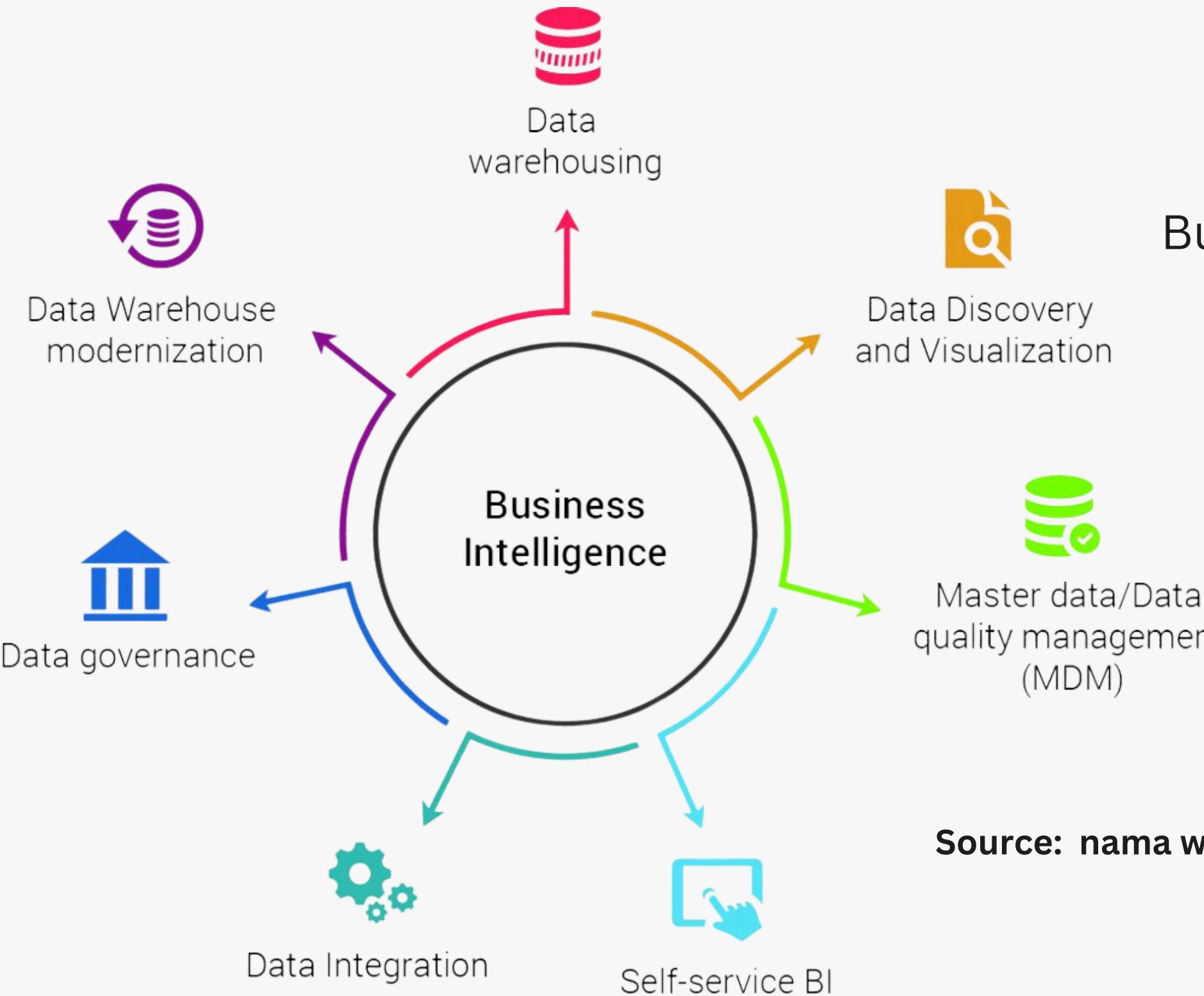
Aspek	Star Schema	Snowflake Schema
Bentuk	Dimensi langsung ke tabel utama.	Dimensi bercabang ke tabel lain.
Data	Tidak dipecah (redundan).	Dipecah jadi tabel kecil.
Query	Mudah dan cepat.	Lebih lambat, banyak tabel dilibatkan.
Ruang Penyimpanan	Butuh ruang lebih besar.	Hemat ruang.
Kapan Dipakai	Untuk laporan cepat, data simpel.	Untuk data kompleks, lebih rapi.



BUSINESS INTELLIGENCE



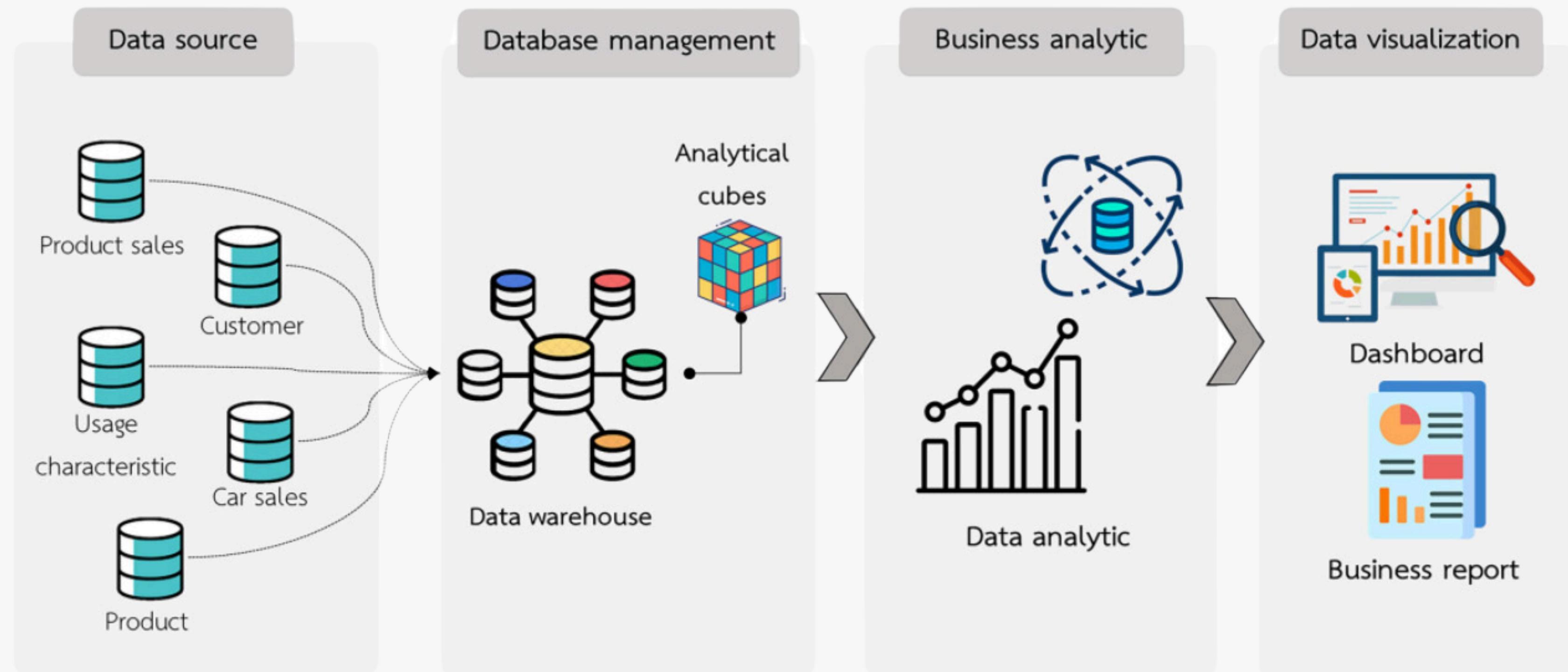
BUSINESS INTELLIGENCE



Business Intelligence (BI) adalah **proses pengumpulan, integrasi, analisis, dan penyajian data untuk mendukung pengambilan keputusan bisnis.**

Source: nama webnya bahasa arab gabisa dicopy sumpah dah

Workflownya begini..





Toolsnya:



Power BI



masih banyak lagi sih..
tergantung perusahaannya



+ a b | e a u



Looker



THANK YOU

ANY FEEDBACKS?