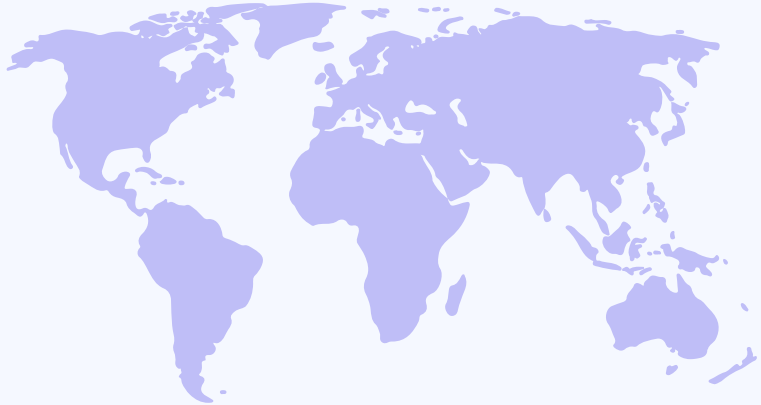
Learning Progress Review - Week II

4 – 9 Okt 2025

Group 5 | **Seru!**

Table of contents



01

Skills of a Data Scientist

Enrichment Session

1. Data Scientist Mindset

CRITICAL THINKING

The ability to **objectively question, analyze, and evaluate data, assumptions, and conclusions** before accepting them as truth.

LOGICAL REASONING

The ability to **connect evidence to conclusions through sound, structured argumentation and cause–effect thinking.**

The **APPLICATION** of critical thinking and logical reasoning in Data analysis

Together, they enable data professionals to transform raw data into meaningful insights and support sound decision-making grounded in factual evidence rather than intuition or bias.

Critical Thinking

Why it matters?

01 **What It Allows Us To Do:**

- Question what seems obvious
- Evaluate evidence, not opinions
- Detect hidden gaps or biases
- Decide with clarity and confidence

02 **When It's Missing:**

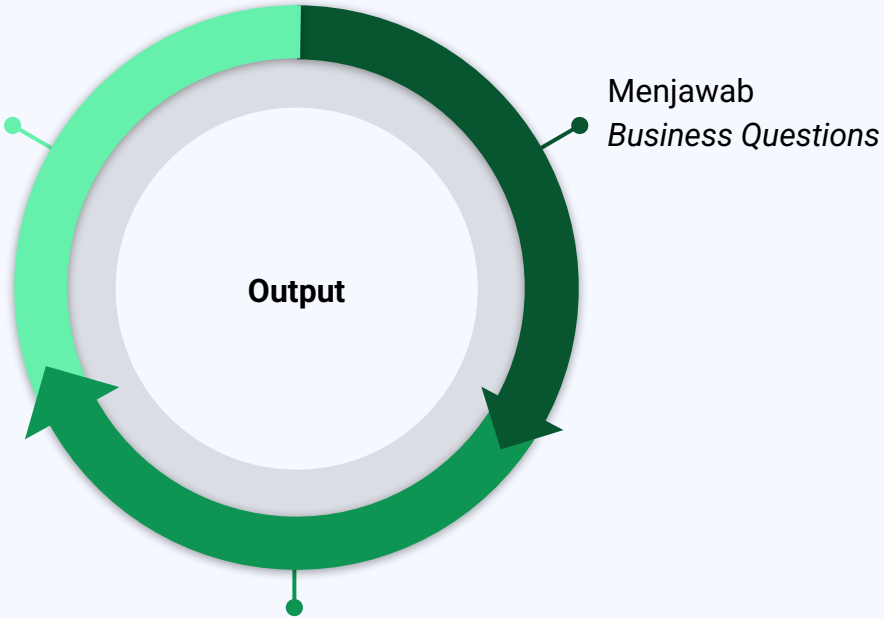
- Misinterpret patterns in the data
- Jump to conclusions based on incomplete evidence
- Be misled by biases or irrelevant information
- Deliver insights that are incorrect or misleading

03 **When It's Applied:**

- Clarify problems before diving into data
- Ask better questions
- Challenge initial assumptions
- Provide deeper and more accurate insights

Logical Reasoning

Its role in Data Analysis



Type of Logical Reasoning

Inductive

From general principle to specific conclusion

Concept:
Start from a known rule or theory, then apply it to a specific dataset or case.

Usage :
hypothesis testing, model validation, rule-based decision systems.

Example:

- Rule: “If sales increase after discount, then discount drives sales.”
- Observation: A 10% discount caused sales to rise by 15%.
- **Conclusion:** The discount likely influences customer buying behavior.

Deductive

From specific observations to general conclusion

Concept:
Observe data patterns and infer a general rule or trend.

Usage :
pattern recognition, exploratory data analysis, predictive modeling.

Example:

- Observation: Customers aged 18–25 buy more online during weekends.
- **Conclusion:** Younger customers tend to shop more on weekends.

Abductive

From incomplete data to the most plausible explanation

Concept:
Make the best possible inference when data is partial or uncertain.

Usage:
anomaly detection, diagnostic analysis, root cause analysis.

Example:

- Observation: Sudden drop in website traffic.
- Possible causes: server issue, SEO drop, or holiday period.
- **Conclusion:** The analyst hypothesizes it’s due to server downtime (most plausible).

Steps in applying logical reasoning

01	**Define the Problem Clearly**	• Tentukan pertanyaan bisnis atau masalah data yang ingin dijawab. • *Tujuan: fokus dan arah analisis menjadi jelas.*
02	**Gather Relevant Information**	• Kumpulkan data dan konteks yang benar-benar berkaitan dengan masalah. • *Tujuan: menghindari bias dan informasi yang tidak relevan.*
03	**Identify Patterns and Relationships**	• Analisis hubungan antar variabel, tren, atau pola dalam data. • *Tujuan: menemukan bukti logis dari observasi.*
04	**Formulate Logical Inferences**	• Gunakan penalaran deduktif dan induktif untuk menarik kesimpulan sementara. • *Tujuan: menguji apakah data mendukung hipotesis.*
05	**Evaluate Evidence Objectively**	• Periksa keandalan data dan argumentasi — hindari bias, asumsi, atau generalisasi. • *Tujuan: menjaga validitas dan objektivitas.*
06	**Communicate Reasoned Conclusions**	• Sampaikan hasil dengan argumen yang runtut, berbasis logika dan bukti. • *Tujuan: membantu pengambil keputusan memahami insight dengan jelas.*

Common Logical Fallacies to Avoid

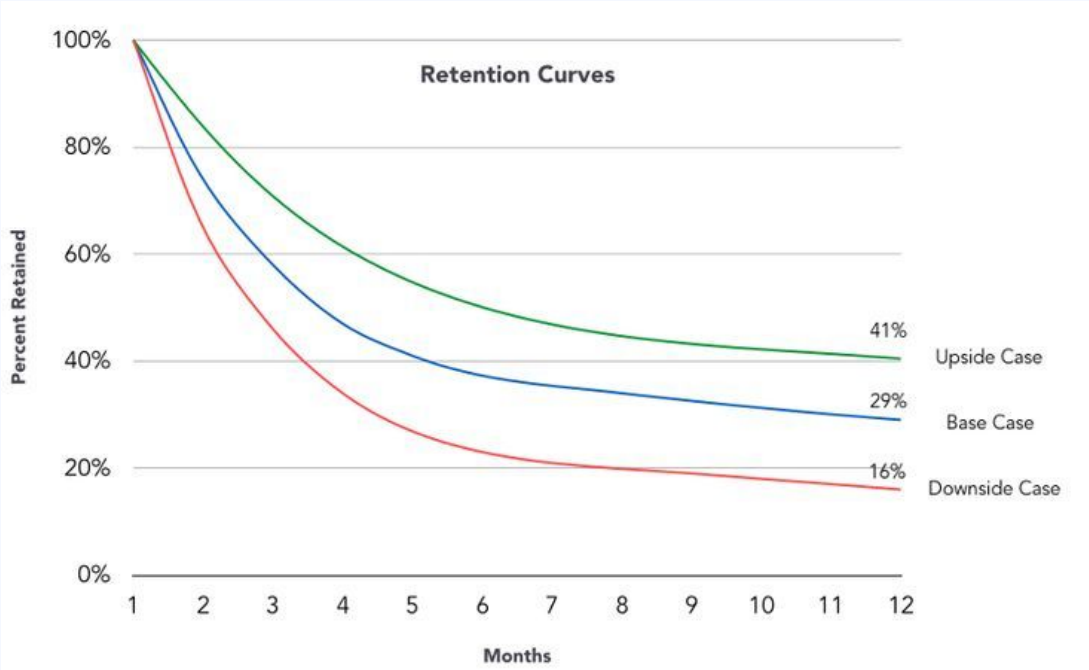
Type	Description
Hasty Generalization	Concluding an entire population based on small unrepresentative sample
Cherry Picking	Select data that supports hypothesis while ignoring contradictory data
Correlation vs Causation	Misinterpreting correlation as causation

Real case study

of applying Critical Thinking and Logical Reasoning

📉 Problem Statement:

Customer retention has decreased over the last quarter.
(Tingkat pelanggan yang kembali membeli menurun dalam tiga bulan terakhir.)



Real case study

of applying Critical Thinking and Logical Reasoning

🧠 **Instead of concluding such "customers are bored" or "competitors are cheaper.", The team questioning:**

- *When* did the drop start?
- *Which* customer segments are affected?
- *What* changed in our business recently?

📊 **Initial finding:**

Retention dropped **from 42% to 29%** — mostly among **repeat buyers aged 18–25** using the **mobile app**.

🎯 **Defined Problem:**

"Retention has dropped specifically among young mobile users in the last quarter."

🔶 Step 1. Identify the Problem

(*Critical Thinking:* Clarify and define the real issue, not just the symptom.)

Real case study

of applying Critical Thinking and Logical Reasoning

📂 Data Collected:

- *Transaction history (purchase frequency, product type).*
- *App performance (loading time, bugs).*
- *Customer feedback & reviews.*
- *Competitor promotions.*

📈 Insights:

- Mobile app reviews dropped from 4.5⭐ to 3.8⭐.
- Users complained about **slow loading** and **app crashes after update**.
- Competitors launched **loyalty programs** offering cashback.

🧩 The team filters out noise (like holiday season spikes) — focusing only on consistent, evidence-based indicators.

🔶 Step 2. Gather and Analyze Information

- (*Critical Thinking:* Collect relevant, accurate data.)
- (*Logical Reasoning:* Identify patterns and relationships.)

Real case study

of applying Critical Thinking and Logical Reasoning

🔶 **Step 3. Formulate Logical Hypotheses**

(*Logical Reasoning:* Develop explanations that can be tested.)
(*Critical Thinking:* Evaluate which hypotheses are most reasonable.)

The team uses **critical judgment** to rank these by likelihood and impact.

💡 **Possible hypotheses:**

- The **recent app update** caused user frustration → leading to churn.
- **Competitor loyalty programs** are attracting returning customers.
- **Customer service response time** is too slow, reducing satisfaction.

Real case study

of applying Critical Thinking and Logical Reasoning

🧪 Testing process:

- Compare retention of users **before and after** the app update.
- Survey customers about reasons for switching.
- Analyze app logs for crash frequency and duration.
- Review customer service response times.

📊 Results:

- 60% of churned users experienced **app errors** in the last month.
- Competitor analysis shows **10% higher cashback offers.**
- No significant change in customer service metrics.

🧠 *Logical reasoning:*

"If retention dropped *after* the update and users report app issues, the update likely contributed to the problem."

🔶 Step 4. Test and Evaluate Evidence

- (*Critical Thinking:* Question and test assumptions objectively.)
- (*Logical Reasoning:* Use causal analysis to confirm/refute each hypothesis.)

Real case study

of applying Critical Thinking and Logical Reasoning

💡 Conclusion:

- The **app update introduced performance issues** that reduced user satisfaction.
- Competitor promotions added **external pressure** that worsened the effect.
- **Customer service** is *not* a major cause.

🧩 Deductive reasoning:

If app issues frustrate users → satisfaction drops → retention decreases.

🧩 Inductive reasoning:

Based on surveys, most lost users reported app crashes → generalize that app quality drives retention.

🔶 Step 5. Draw Logical Conclusions

- (*Critical Thinking:* Assess validity of findings.)
- (*Logical Reasoning:* Connect cause and effect clearly.)

Real case study

of applying Critical Thinking and Logical Reasoning

✅ Actions Taken:

- **Fix app bugs** and release an improved version.
- Launch **retention campaign** with cashback for returning users.
- Set up **in-app satisfaction survey** to monitor ongoing user sentiment.

📈 Results After 2 Months:

- App rating increased from **3.8 ⭐ to 4.4⭐**.
- Retention rebounded from **29% → 38%**.
- Customer complaints decreased by **35%**.

🔶 Step 6. Make Decisions and Take Action

- (*Critical Thinking:* Choose the most effective, data-driven solution.)
- (*Logical Reasoning:* Implement actions that logically solve the root cause.)

2. Data Scientist Soft Skill

01

Data Storytelling

Kemampuan mengubah data menjadi informasi yang mudah dipahami oleh audiens.

02

Komunikasi

Dengan kemampuan komunikasi yang baik agar hasil analisis data dapat disampaikan dengan jelas dan mudah dipahami.

03

Berfikir Kritis

Berpikir kritis penting bagi Data Scientist dalam menjalankan proses pengumpulan, analisis, hingga penyajian hasil data secara efektif.

04

Kerjasama Tim

Data Scientist harus mampu bekerja sama dengan baik dalam tim agar proses kerja berjalan lancar dan efisien.

Tahapan Pengolahan Data

Menentukan Tujuan

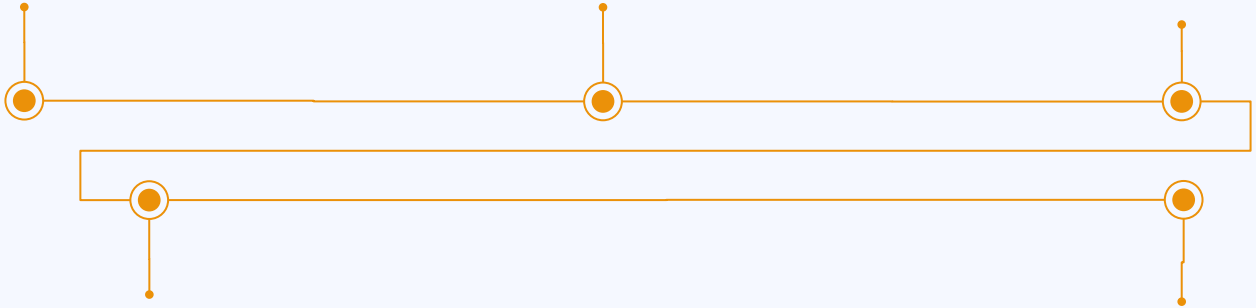
Define

Mengolah data

Processing

Mengumpulkan data

Collect



Integration

Menggabungkan data

Analysis

Menganalisis data

Membangun Insight

Menemukan insight menjadi tantangan bagi Data Scientist untuk mengungkap pola dan tren data yang membentuk hubungan serta aturan yang menghasilkan rekomendasi objektif dan menguntungkan berdasarkan bukti data.

Beberapa cara membangun insight:

1. Analisa informasi deskriptif
2. Analisa Hubungan Informasi Diagnostik
3. Teknik Deskriptif dan Uji Statistika

Membangun Algoritma Prediksi

Tantangan terbesar seorang Data Scientist adalah membangun algoritma prediksi berdasarkan keterkaitan antar informasi dalam data yang telah dikumpulkan dan digabungkan, untuk kemudian diimplementasikan pada aplikasi berbasis web, mobile, atau sistem ERP guna mendukung dan mengoptimalkan proses bisnis perusahaan.

Teori dan aturan konsep algoritma:

1. Classical Statistic
2. Machine Learning
3. Human Centered Modelling

Business Understanding

Data scientist perlu memahami secara menyeluruh terkait proses bisnis. Hal itu diperlukan agar insight dan algoritma prediksi yang tepat sehingga menghasilkan solusi yang tepat.

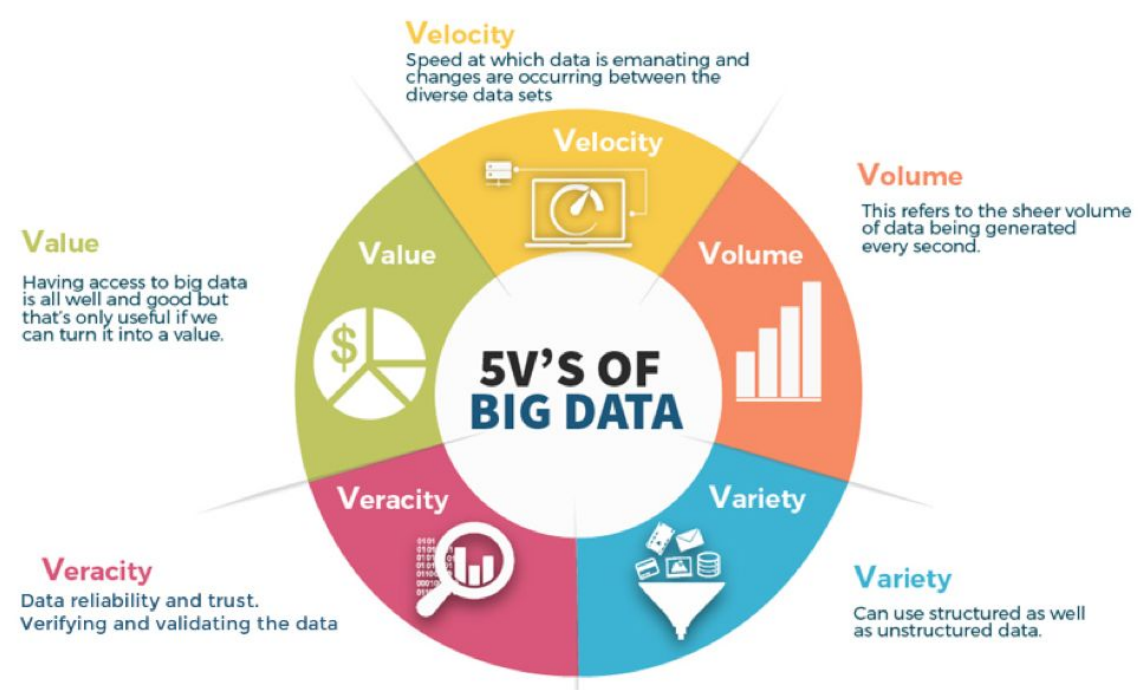
1. General and Specific Objective
2. Situational Analysis
3. Broad Goals of Data Mining Process
4. Project Planning

Analytics Approach

Analisis Deskriptif	Memperoleh informasi terkait perubahan nilai dan hubungan antar data
Analisis Diagnostik	Mengetahui penyebab dari perubahan nilai dan hubungan antar data
Analisis Prediktif	Mendapatkan tren atau pola dari suatu data
Analisis Preskriptif	Membuat rekomendasi dari hasil prediksi

3. Data Fundamentals

Big Data Characteristics:



4. Data Analysis Tools

Data Storage, Cleaning, and Preparation

- Menyimpan dan mengorganisir data
- Membersihkan dan menyiapkan data
- Mengambil data sesuai kebutuhan

Data Processing

- Mengolah dan memanipulasi data
- Mencari pola, tren, dan hubungan
- Menerapkan analisis/statistik

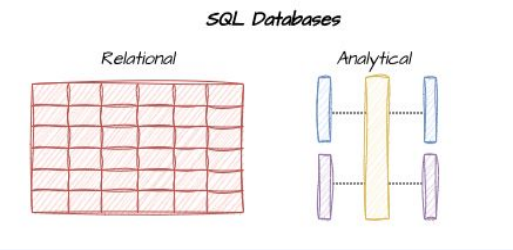
Data Presentation

- Menyajikan data dengan visualisasi
- Menyederhanakan informasi kompleks
- Mendukung pengambilan keputusan

5. SQL vs NoSQL

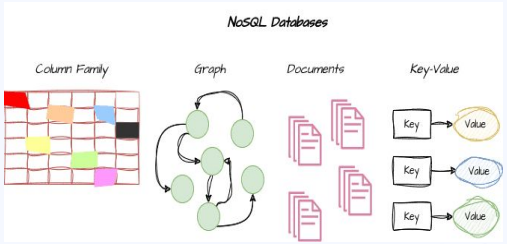
SQL ?

SQL, which stands for “Structured Query Language,” is the programming language that’s been widely used in managing data in relational database management systems (RDBMS) since the 1970s.



NoSQL ?

NoSQL, which stands for Not only SQL, is a database management system approach used to ingest, store, and retrieve unstructured data and semi-structured data within a database



source : https://www.mongodb.com/resources/basics/databases/nosql-explained/nosql-vs-sql

source : https://www.baeldung.com/cs/sql-vs-nosql#bd-sql-vs-nosql-databases

Perbedaan Utama Antara SQL dan NoSQL

Aspek	SQL (Relasional)	NoSQL (Non-relasional)
Model Data	Data disimpan dalam tabel yang terdiri dari baris dan kolom. Hubungan antar tabel ditentukan oleh *foreign key*.	Data disimpan dalam format yang lebih fleksibel, seperti dokumen (misalnya, JSON, BSON),Key-Value, Graph (Node dan Edge), Wide-Column.
Skema	Memiliki skema yang kaku dan sudah ditentukan sebelumnya. Setiap baris harus sesuai dengan struktur kolom yang ada.	Memiliki skema yang fleksibel atau "tanpa skema". Struktur data bisa berbeda antara satu entri dengan yang lain.
Skalabilitas	Menerapkan penskalaan vertikal, yaitu dengan meningkatkan kapasitas pada satu server (misalnya, menambah CPU atau RAM). Penskalaan horizontal dimungkinkan, tetapi lebih rumit.	Menerapkan penskalaan horizontal, yaitu dengan mendistribusikan data ke banyak server atau node. Lebih cocok untuk menangani volume data yang besar dan berkembang pesat.
Integritas Data	Memastikan integritas data melalui properti ACID (Atomicity, Consistency, Isolation, Durability), yang penting untuk transaksi yang andal.	Sebagian besar basis data NoSQL mengorbankan konsistensi ketat demi ketersediaan dan toleransi partisi, menggunakan model konsistensi akhir (BASE). Namun, beberapa database NoSQL modern juga mendukung transaksi ACID.
Bahasa Kueri	Menggunakan bahasa standar bernama SQL, yang kuat untuk melakukan kueri kompleks dan penggabungan (*join*) data antar tabel.	Tidak ada bahasa standar. Setiap jenis database NoSQL memiliki bahasa kueri atau API sendiri. Kueri cenderung lebih sederhana.
Uses Cases	Complex transaction, structured data	Unstructured dan semi-structured data
Contoh	MySQL, PostgreSQL, Oracle, SQL Server	MongoDB, Cassandra, Couchbase, Radis

source : https://lnk.dev/JwZjP

SQL : PostgreSQL

user_id	nama_lengkap	email	kota_as...	salary
1	Azra	azra@gmail.com	Jakarta	7000
2	Muhimah	muhimah@gmail.com	Depok	7500
3	Dhinar	dhinar@gmail.com	Jakarta	5500
4	Tyson	tyson@gmail.com	Bekasi	7000
5	Samuel	samuel@gmail.com	Bekasi	7000
6	Bayu	bayu@gmail.com	Depok	5500
7	Andi	andi@gmail.com	Depok	6500

NoSQ : MongoDB

Filter Type a query: { field: 'value' }

QUERY RESULTS: 1-2 OF 2

```
_id: ObjectId('678e66e8b1c0843b20f92a4f')
name : "Kalung | Necklace Untuk Anjing dan Kucing"
category : "Kalung"
quantity : 12
price : 25000
image : "kalung_biru.jpg"
```

```
_id: ObjectId('678e7e04b1c0843b20f92a55')
name : "Pet Cargo | Grey "
category : "Pet Bed"
quantity : 20
price : 115000
image : "cargo_abu.jpeg"
```

6. Data Drilling

Deep dive into data to uncover detailed information

Primary Reasons

1. Supporting root cause analysis by breaking down complex metrics.
2. Enhancing business intelligence by enabling deeper, more insightful questions.
3. Allowing for interactive reporting where users can explore data dynamically.

Example: Loan Portfolio Risk

- Problem: a 1% increase in loan default rate in the last quarter, in a **large national bank**.
- Investigate default rate changes > dive into loan type > dive into region > conclude
- Conclusion: that 1% default rate increase is from the high-risk segment of personal loans in some regions

Drill Down

Investigate from the summary to the detail view.

Drill Up

Aggregate detailed data back to a higher level of summary.

Hierarchical

Investigate hierarchical data, like time (year > quarter > month) or geography (country > state > city).

Interactive Dashboards

Interactive visualization like Tableau, Power BI, and Looker, where users can click on data points to drill down and get a more detailed view.

7. Data Diagnostic

Uncovers the root causes of trends, patterns, and anomalies in the data

Purpose

1. Identify data anomalies or inconsistencies within the data.
2. Uncover the relationships between variables.
3. Provide evidence to either support or challenge a hypothesis.
4. Guide corrective actions or necessary changes in strategy.

Key issues to look for

- Missing or inconsistent data entries.
- Unusual spikes or sudden drops in metrics.
- Duplicate data records.
- Unexpected data distributions.
- High correlations between predictor variables, also known as multicollinearity.

01 — EDA

Use statistical summaries and visualizations to explore the data, to identify outliers, skewness, and missing values.

02 — Correlation Analysis

Measure the linear relationships between variables to help identify potential causal factors.

03 — Segmentation

The data is broken down into different groups and then key metrics are compared to spot significant differences.

04 — Time Series

Analyze trends, seasonality, and change points over time using autoregressive model, MA, etc.

8. Data Protection

Definisi:

perlindungan data pribadi, sensitif, atau penting dari akses yang tidak sah, penyalahgunaan, kehilangan, atau kerusakan.

Kenapa perlindungan data dibutuhkan ?

- Melindungi kepercayaan pengguna dan reputasi perusahaan
- Mencegah pelanggaran data, penipuan, dan kerugian finansial
- Memastikan kepatuhan terhadap standar hukum dan industri

Data Protection - Core Principle

Confidentiality

Hanya orang yang berwenang yang dapat mengakses data

Integrity

Data harus akurat dan tidak diubah tanpa izin

Availability

Data harus dapat diakses oleh mereka yang membutuhkannya saat mereka membutuhkannya

Accountability

Organisasi harus menunjukkan penanganan data yang bertanggung jawab

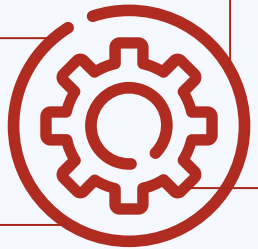
Data Protection - Legal Standard

GDPR

- General Data Protection Regulation - EU
- Berlaku untuk semua organisasi yang menangani data warga negara Uni Eropa

HIPAA

- Health Insurance Portability and Accountability Act - US
- Berlaku untuk data yang berhubungan dengan kesehatan
- Menekankan kerahasiaan dan keamanan



CCPA

- California Consumer Privacy Act
- Memberikan konsumen hak untuk mengetahui, menghapus dan memilih keluar dari pembagian data

ISO/IEC 27001

Standar Internasional untuk Information Security Management System

Data Protection - Examples

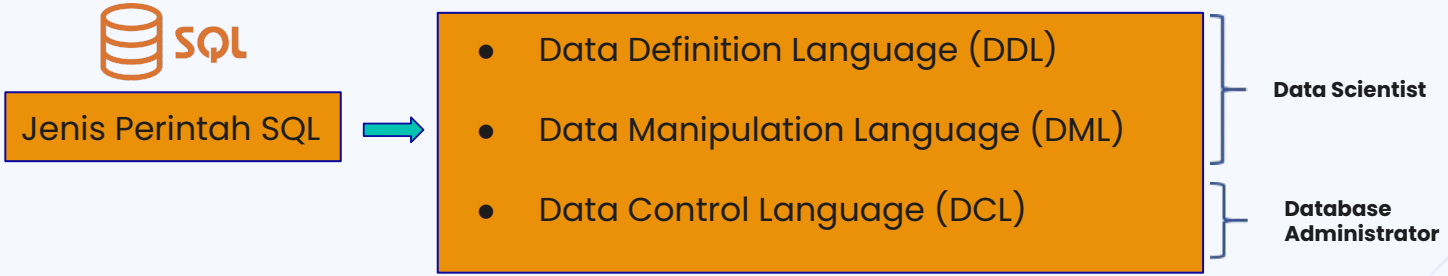
- Pengumpulan data pelanggan (nama, email, preferensi) di aplikasi pengiriman makanan online
- Enkripsikan data sebelum menyimpannya di database
- Gunakan role-based permission sehingga hanya agen layanan pelanggan yang dapat melihat profil pengguna
- Audit log secara berkala untuk pola akses yang tidak biasa
- Pastikan pengguna dapat meminta penghapusan data mereka (kepatuhan GDPR)

02

Basic SQL (Postgre)

Scope of Basic SQL

SQL atau Standard Query Language adalah **bahasa pemrograman** yang digunakan dalam **mengakses, mengubah, dan memanipulasi** data yang berbasis relasional berdasarkan standar American Nastional Standard Institute. Mencakup operasi mendasar yang diperlukan untuk berinteraksi dengan data dan mengelolanya dalam basis data relasional. Di dalamnya terdapat beberapa jenis perintah dasar yang memiliki fungsi berbeda-beda.

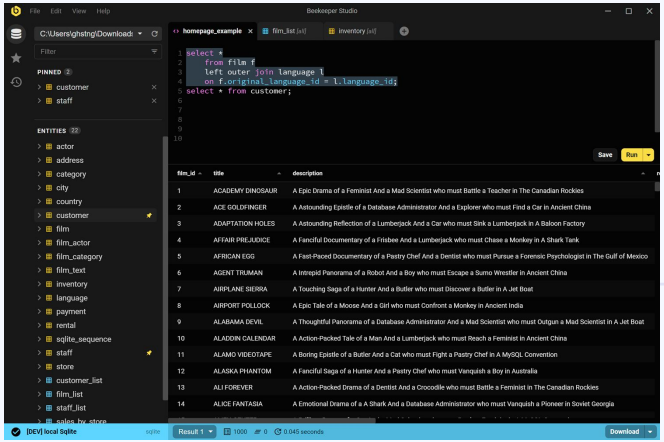


PostgreSQL dan Beekeeper Studio

PostgreSQL adalah salah satu database manajemen relasional open source yang digunakan untuk menyimpan, mengelola dan memproses data yang umumnya digunakan di program linux.

PostgreSQL

Beekeeper Studio adalah editor SQL dan aplikasi pengelola basis data yang bersifat open source.



Data Definition Language (DDL)

Perintah yang digunakan untuk mendefinisikan struktur data di dalam database, seperti membuat, mengubah, dan menghapus tabel dan database baru.

Create — Membuat objek baru (database, tabel, kolom, trigger, prosedur)

CREATE TABLE nama_tabel**;** Command membuat tabel baru

Alter — Memodifikasi struktur tabel yang sudah ada (termasuk perintah **RENAME**)

ALTER TABLE nama_tabel **ADD COLUMN** nama_kolom**;** Command menambahkan kolom baru

Drop — Menghapus komponen dalam database (**tidak bisa dikembalikan!**)

DROP TABLE nama_tabel**;** Command menghapus tabel

Show — Menampilkan list database, tabel, dan struktur lainnya

SHOW TABLES FROM nama_database; Command menampilkan list tabel yang ada di database tertentu

Data Control Language

Definisi:

Perintah yang digunakan khususnya untuk **mengelola** atau melakukan **kontrol** terhadap **privilege** atau **hak akses** yang dimiliki oleh **pengguna.**

Tujuan:

Menjaga **integritas** dan **keamanan database**

Contoh:

Perintah **GRANT** dan **REVOKE**

Data Control Language

GRANT

Digunakan untuk **memberikan wewenang/hak akses** tertentu kepada user untuk melakukan aksi tertentu pada object DB.

GRANT tipe_privilese
ON nama_object
TO nama_user

REVOKE

Digunakan untuk **menghapus** atau **mencabut wewenang/hak akses** tertentu kepada user, yang semula telah diberikan dengan perintah GRANT.

REVOKE tipe_privilese
ON nama_object
FROM nama_user

Data Manipulation Language

SQL prompts to manipulate data in a relational database

1) Insert

INSERT INTO pegawai (no, nama, hobi)
VALUES (1,"Billy","Baca");

No	Nama	Hobi

No	Nama	Hobi
1	Billy	Baca

2) Select

SELECT * **FROM** pegawai;

No	Nama	Hobi
1	Billy	Baca

SELECT no, nama **FROM** pegawai;

No	Nama
1	Billy

3) Update

UPDATE pegawai **SET** nama = Adit
WHERE No = 1;

No	Nama	Hobi
1	Billy	Baca

No	Nama	Hobi
1	Adit	Baca

4) Delete

DELETE FROM pegawai **WHERE** nama = "Adit";

No	Nama	Hobi
1	Adit	Baca

No	Nama	Hobi

03

Advanced SQL

Select Distinct

Fungsi

- Mengambil nilai unik dari suatu kolom dalam tabel.
- Menghapus record duplikat (nilai yang sama dianggap satu).

Syntax

- SELECT DISTINCT kolom1, kolom2, ...
- FROM nama_tabel;

Contoh Penggunaan

- SELECT DISTINCT “Location”
- FROM customers;

Where Condition

Fungsi :

- Digunakan untuk memfilter hasil **SELECT** dengan mengextract record yang memenuhi persyaratan tertentu
- Sering digunakan dalam proses data cleaning dan data transformation

Syntax :

SELECT column1,column2,...
FROM nama_table
WHERE condition

***syntax* di dalam SQL harus berurutan** seperti di atas, WHERE selalu berada setelah FROM dan SELECT selalu berada di atas FROM

Jenis-jenis Operator untuk WHERE :

Operator	Deskripsi
=	Sama dengan
>	Lebih besar dari
<	Kurang dari
>=	Lebih besar dari sama dengan
<=	Kurang dari sama dengan
<> atau !=	Tidak sama dengan
BETWEEN	Berada di antara jarak tertentu
LIKE	Melihat sebuah pola
IN	Berada didalam sebuah kolom

Source : https://www.revou.co/id/panduan-teknis/sql-where

String Function

SQL functions to manipulate String

1) LOWER

LOWER(“Grup Seru”)

Output : “grup seru”

2) UPPER

UPPER(“Grup Seru”)

Output : “GRUP SERU”

3) LENGTH

LENGTH(“Grup Seru”)

Output : 9

4) SUBSTRING

SUBSTRING(“Grup Seru”,3,2)

Output : “up”

5) CONCAT

CONCAT(“Grup Seru”, “ 99+”, “Aura”)

Output : “Grup Seru 99+Aura”

Aggregate Function

Pengertian

- Fungsi untuk menghitung atau meringkas data menjadi satu nilai tunggal.
- Digunakan dalam analisis seperti rata-rata, jumlah total, nilai tertinggi, dll.

Jenis Agregasi

- Avg (Menghitung rata-rata)
- COUNT (Menghitung jumlah record)
- MAX (Mengambil nilai terbesar)
- MIN (Mengambil nilai terkecil)
- SUM (Menghitung total nilai)

Aggregate Function

Penulisan Syntax

Count

```
SELECT
COUNT(nama_kolom)
FROM nama_tabel;
```

Average

```
SELECT
AVG(nama_kolom)
FROM nama_tabel;
```

Max

```
SELECT
MAX(nama_kolom)
FROM nama_tabel;
```

MIN

```
SELECT
MIN(nama_kolom)
FROM nama_tabel;
```

SUM

```
SELECT
SUM(nama_kolom)
FROM nama_tabel;
```

Subqueries/ Inner Query/ Nested Query

Definisi:

Subqueries adalah query di dalam query, yang digunakan di dalam sebuah query utama sebagai syarat untuk lebih membatasi / menspesifikkan data yang akan ditampilkan.

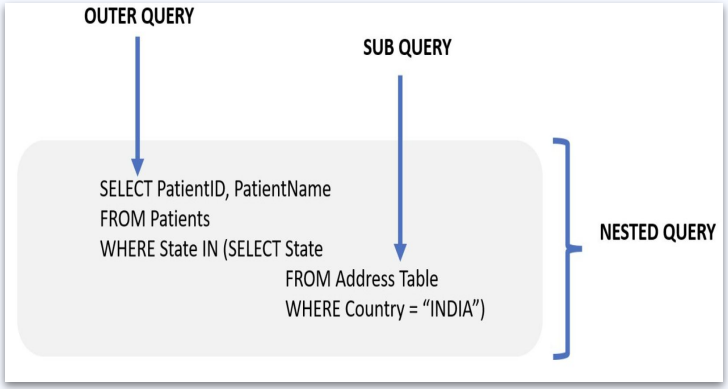
Beberapa tipe subqueries:

1. Subqueries IN FILTER
2. Subqueries IN TABLE
3. Common Table Expression (CTE)

Subqueries IN FILTER

Definisi:

Hasil data dari perintah SQL yang berhasil dijalankan sehingga dapat dipergunakan kembali untuk diproses oleh perintah SQL yang lainnya.



Source: https://www.shiksha.com/online-courses/articles/subqueries-in-sql

Note:

- Subqueries akan dijalankan terlebih dahulu
- Hanya dapat memilih 1 kolom pada subqueries yang dihasilkan
- Tidak ada batasan jumlah subqueries
- Semakin banyak subqueries akan mempengaruhi performansi SQL

Subqueries IN TABLE

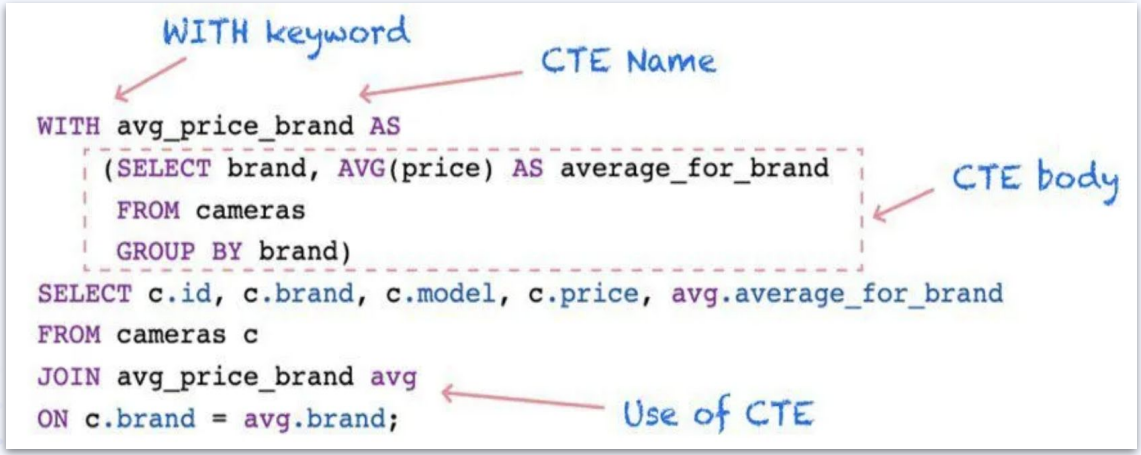
Definisi:

Subqueries tidak hanya dapat dilakukan pada bagian filter (WHERE), tetapi dapat juga dilakukan pada bagian table yang menjadi sumber.

```
SELECT AVG(billing_country_max) AS billing_country_max_avg
FROM (SELECT billing_country, MAX(total) AS billing_country_max
        FROM invoice
       GROUP BY billing_country);
```

Source: https://www.dataquest.io/tutorial/sql-subqueries-tutorial

Common Table Expression (CTE)



Source: https://programmers.io/blog/guide-to-common-table-expressions-ctes |

CTE digunakan untuk mendefinisikan temporary view yang akan sangat bermanfaat dalam menyederhanakan query yang kompleks dan juga membantu memudahkan dalam membaca dan memahami SQL Query.

Selain itu, implementasi CTE juga akan dapat meningkatkan performansi SQL

JOIN FUNCTION

Combining data from two or more tables based on common columns between them

INNER JOIN

Menampilkan data yang cocok (match) di kedua tabel.

```
SELECT pelanggan.nama_pelanggan, pesanan.produk
FROM pelanggan
INNER JOIN pesanan
ON pelanggan.id_pelanggan = pesanan.id_pelanggan;
```

➡️ **Hanya pelanggan yang punya pesanan** yang muncul.

LEFT JOIN

Menampilkan **semua** data dari **tabel kiri** (**LEFT**), dan data **yang cocok** dari **tabel kanan** (**RIGHT**).

```
SELECT pelanggan.nama_pelanggan, pesanan.produk
FROM pelanggan
LEFT JOIN pesanan
ON pelanggan.id_pelanggan = pesanan.id_pelanggan;
```

➡️ Semua pelanggan tetap ditampilkan, walau **belum pernah memesan apa pun**.

RIGHT JOIN

Menampilkan **semua** data dari **tabel kanan** (**RIGHT**), dan data **yang cocok** dari **tabel kiri** (**LEFT**).

```
SELECT pelanggan.nama_pelanggan, pesanan.produk
FROM pelanggan
RIGHT JOIN pesanan
ON pelanggan.id_pelanggan = pesanan.id_pelanggan;
```

➡️ Pesanan **tetap tampil** meski pelanggan belum terdaftar.

FULL JOIN

Menampilkan **semua data** dari **kedua tabel**, baik yang **cocok** maupun **tidak cocok.**

```
SELECT pelanggan.nama_pelanggan, pesanan.produk
FROM pelanggan
FULL JOIN pesanan
ON pelanggan.id_pelanggan = pesanan.id_pelanggan;
```

➡️ Semua pelanggan **dan** semua pesanan ditampilkan — termasuk pelanggan tanpa order dan pesanan tanpa pelanggan.

Thank You