



LEARNING PROGRESS REVIEW

Week 2

Entropy Team

A large, light gray semi-circle graphic located in the bottom right corner of the slide.

OUR TEAM

Entropy Team



Adhang Muntaha Muhammad

<https://www.linkedin.com/in/adhangmuntaha/>



Aziz Fauzi

<https://www.linkedin.com/in/aziz-fauzi-a6904711b/>



Iwan Wahyu

<https://www.linkedin.com/in/iwan-wahyu-setyawan-506809183>



Marcellina Alvita F

<https://www.linkedin.com/in/marcellina-alvita-faustina-63a284226>



Ramadhan Luthfan

<https://www.linkedin.com/in/luthfan-mahathir-91369b18b>

DAFTAR ISI

1.

Introduction to Data & Database

Pengenalan tentang
data dan *database*

2.

Basic SQL

Materi SQL dasar
(PostgreSQL)

3.

Intermediate SQL

Materi SQL menengah
(PostgreSQL)

01

INTRODUCTION TO DATA & DATABASE

Pengenalan tentang
data dan *database*

Data



- Data adalah suatu **informasi** yang dikumpulkan, baik dalam media **elektronik** maupun **non-elektronik**
- Data dapat **dianalisis** dan digunakan untuk membantu dalam **pembuatan keputusan**

Jenis Data

Categorical Data

Data memiliki jumlah **variasi** yang **tetap**

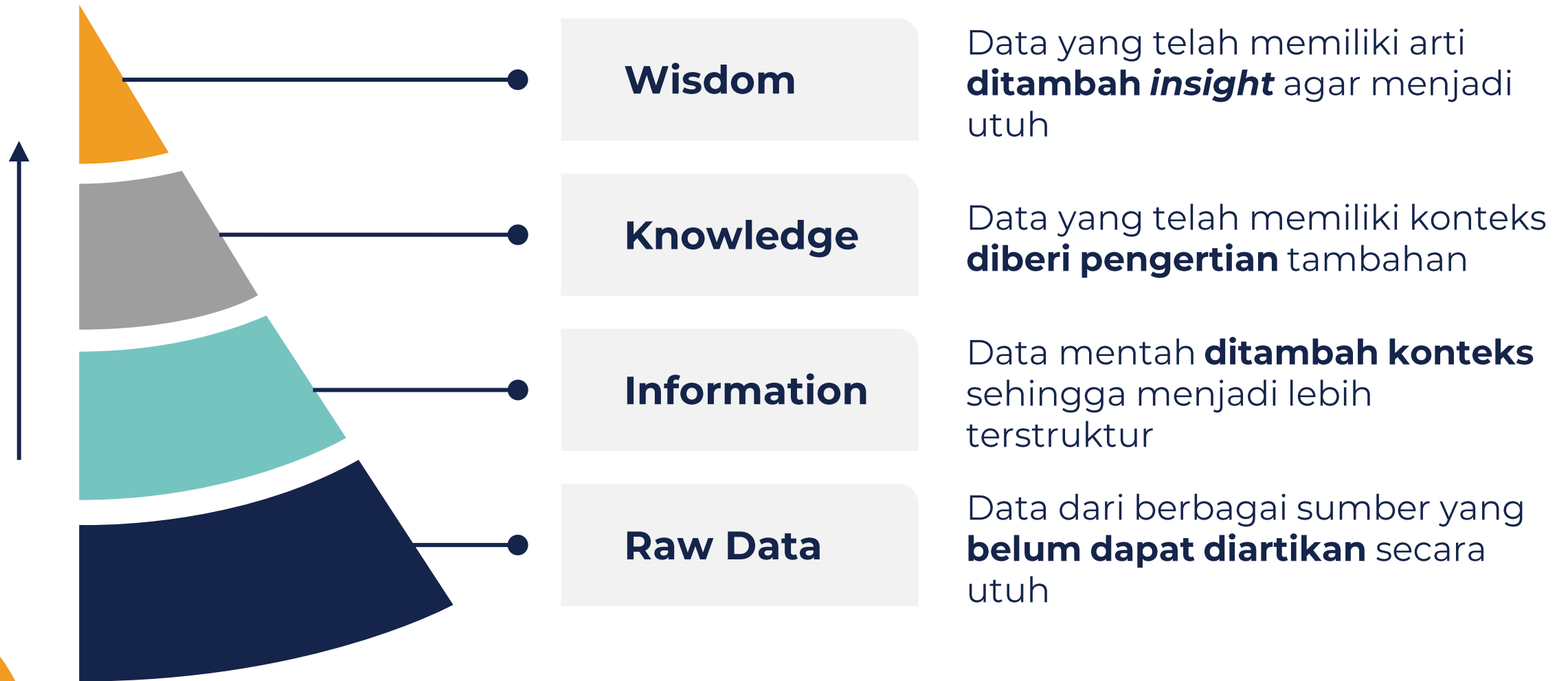
- **Nominal**
Kategori tidak memiliki bobot, misal jenis kelamin
- **Ordinal**
Kategori memiliki bobot, misal ranking

Numerical Data

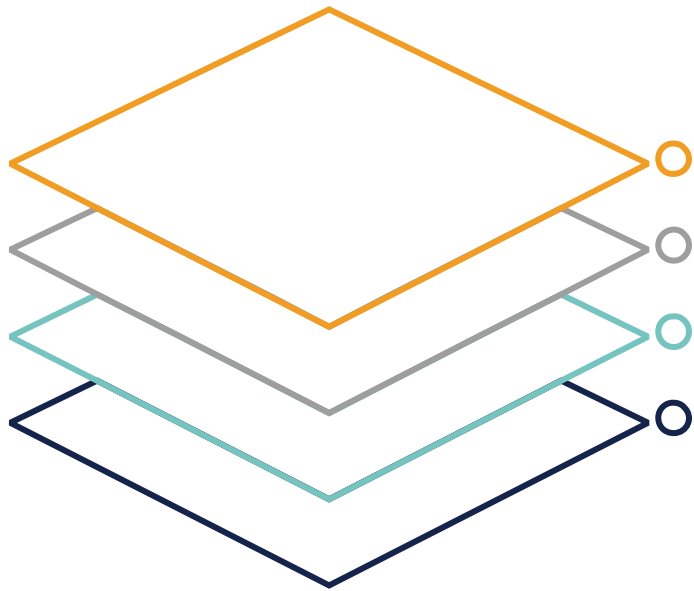
Data memiliki jumlah **variasi** yang **tidak tetap**

- **Discrete**
Data berbentuk bilangan bulat, misal jumlah barang
- **Continuous**
Data dapat berbentuk pecahan, misal harga barang

Hierarki Data



Database



- *Database* adalah suatu **kumpulan data** yang disimpan dan diakses secara **elektronik**

Database Management System



- Database Management System (DBMS) adalah **tools** yang dapat digunakan untuk **mengakses** dan **mengelola** data pada **database**

Relational DBMS



- **Relational DBMS** (RDBMS) adalah pengembangan DBMS untuk **database** yang memiliki **keterkaitan** (*relational database*)
- Contoh RDBMS: **PostgreSQL, MySQL, Oracle**
- **SQL** (Structured Query Language) adalah **bahasa standar** pada **RDBMS** untuk mengakses *database*

Jenis Database

Structured

Database memiliki **susunan** data yang **terstruktur**

- **Kelebihan**
Mudah diimplementasikan karena data bersifat tabular
- **Kekurangan**
Terbatas pada *relational database*
- **Tools**
PostgreSQL, MySQL, dll

Unstructured

Database memiliki **susunan** data yang **tidak terstruktur**

- **Kelebihan**
Menghemat *storage* karena format data tidak perlu diubah
- **Kekurangan**
Perlu *skill* dan *tools* tertentu untuk mengolah data
- **Tools**
MongoDB, Hadoop, dll

Tipe Data

Numeric

Menyimpan data
bilangan

Misal: int, float

Date & Time

Menyimpan data
waktu

Misal: date, timestamp

String

Menyimpan data
karakter atau **teks**

Misal: char, varchar

Unicode

Menyimpan data
karakter unicode

Misal: nchar, nvarchar

Binary

Menyimpan data
berformat biner

Misal: binary, varbinary

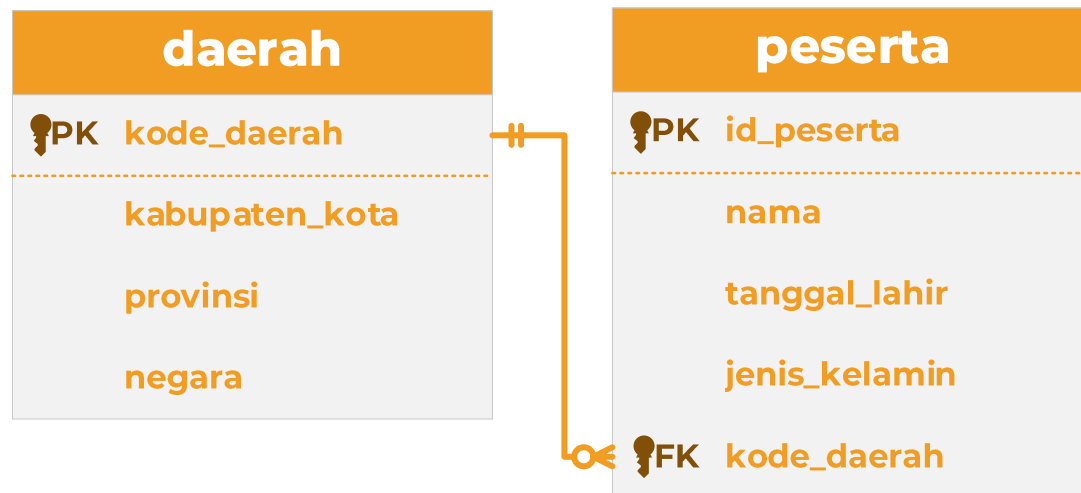
Lainnya

Menyimpan data
berformat tertentu

Misal: XML, JSON

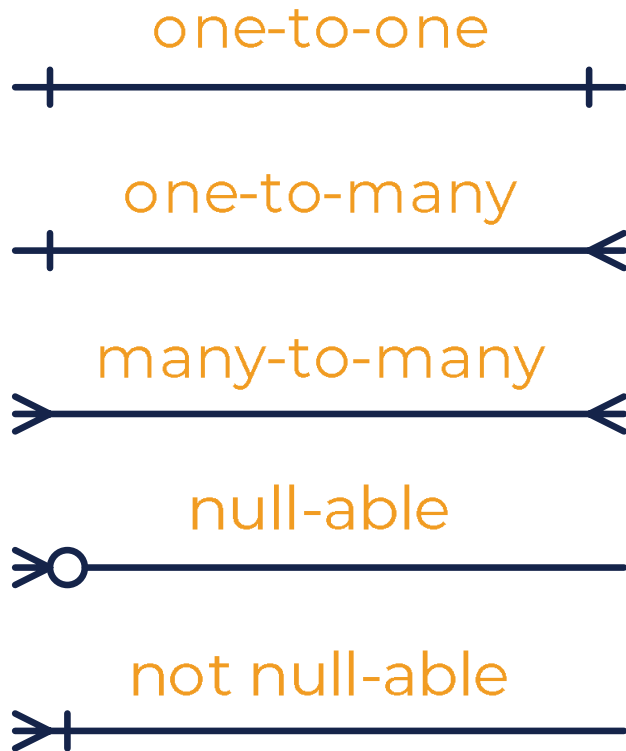
Disclaimer: tiap RDBMS bisa memiliki tipe data dan penamaan yang berbeda

Entity Relationship Diagram



- Entity Relationship Diagram (ERD) adalah sebuah **diagram** yang menggambarkan **hubungan antartabel**
- **Primary key** (PK) adalah kolom yang memiliki nilai **unique** (tidak ada data bernilai sama)
- **Foreign key** (FK) adalah kolom pada suatu tabel yang merupakan PK pada tabel lain

Cardinality



- Menggambarkan jumlah **minimal** dan **maksimal** kemunculan **entitas** pada suatu tabel dan entitas pada tabel lainnya
- Jumlah **maksimal**
 - One-to-one
 - One-to-many
 - Many-to-many
- Jumlah **minimal**
 - Null-able
 - Not null-able

Participant Constraint



Mandatory

- Paling tidak, terdapat 1 entitas yang terhubung dengan entitas lain
- *Cardinality* bertipe **not null-able**
- Contoh:
Setiap kelas harus diambil oleh 1 atau lebih peserta

Optional

- Entitas diperbolehkan tidak terhubung dengan entitas lain
- *Cardinality* bertipe **null-able**
- Contoh:
Tidak semua peserta harus mengambil kelas

Tipe Entitas

Strong

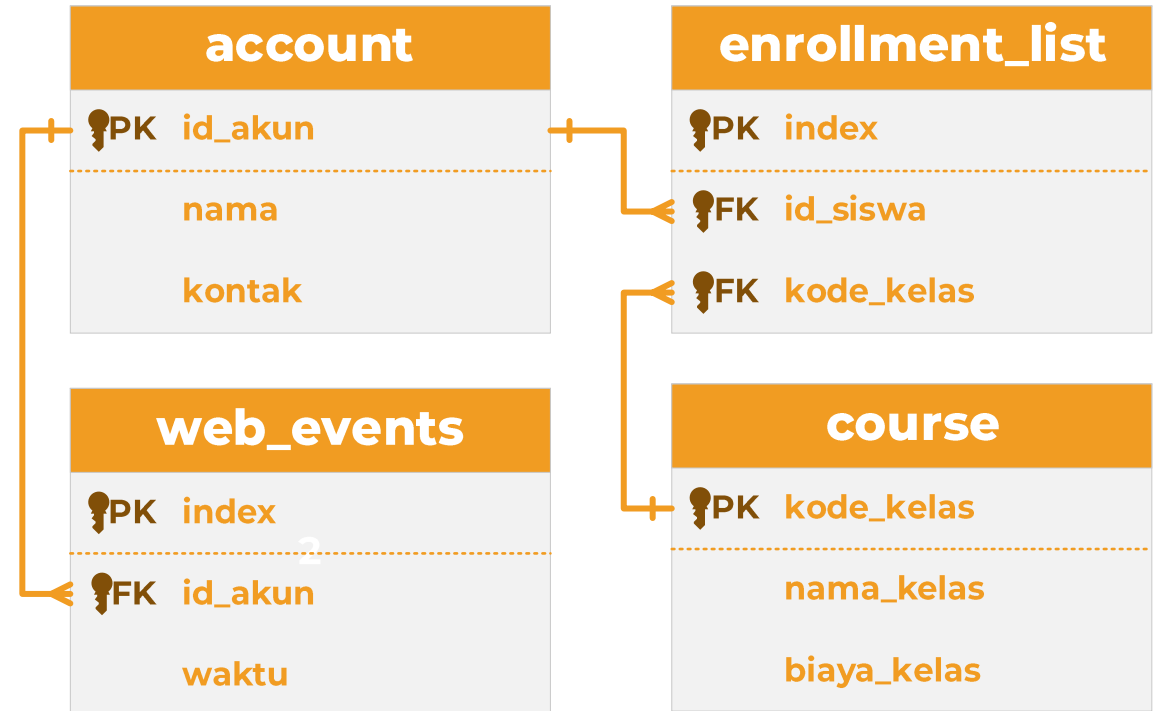
- Dapat berdiri sendiri
- Contoh:
Tabel *account* dan *course*

Weak

- Tidak dapat berdiri sendiri
- Contoh:
Tabel *web_events* (jika tanpa *index*)

Associative

- Entitas terbentuk karena entitas lain
- Contoh:
Tabel *enrollment_list* (jika tanpa *index*)



Note: tabel *web_events* dan *enrollment_list* sudah ditambahkan kolom *index* sebagai pengganti (*surrogate*) PK. Dengan adanya pengganti PK tersebut, maka tipe entitas menjadi *strong*

Data Lifecycle



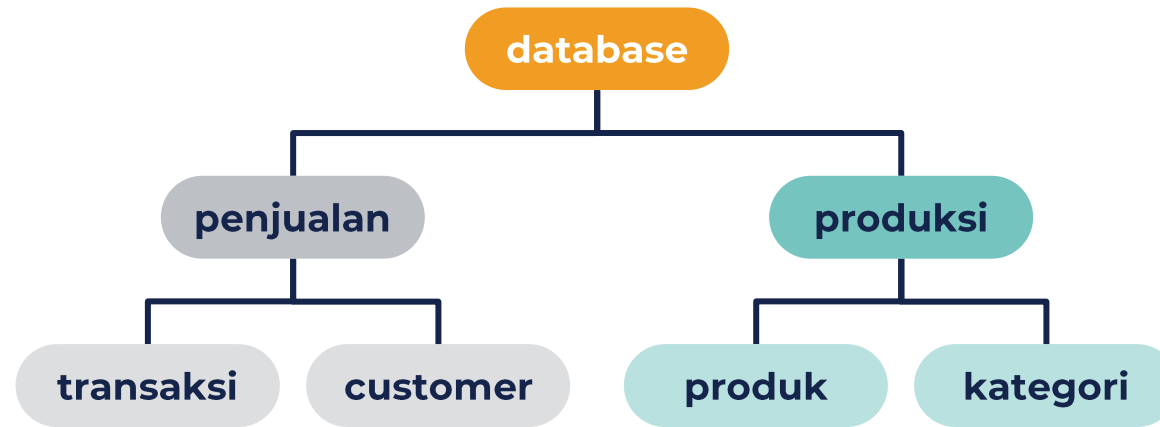
- *Data lifecycle* adalah **siklus** berulang yang terjadi pada **data**, mulai dari pembuatan data sampai interpretasi
- Urutan **secara umum**:
 - Pembuatan data
 - Pengumpulan data
 - Pemrosesan data
 - Penyimpanan data
 - Manajemen data
 - Analisis data
 - Visualisasi
 - Interpretasi

02

BASIC SQL

Materi SQL dasar
(PostgreSQL)

Schema & Table



Table

Tempat untuk **menyimpan** berbagai **data** secara **tabular**

Contoh
tabel transaksi,
customer, produk, dan
kategori

Schema

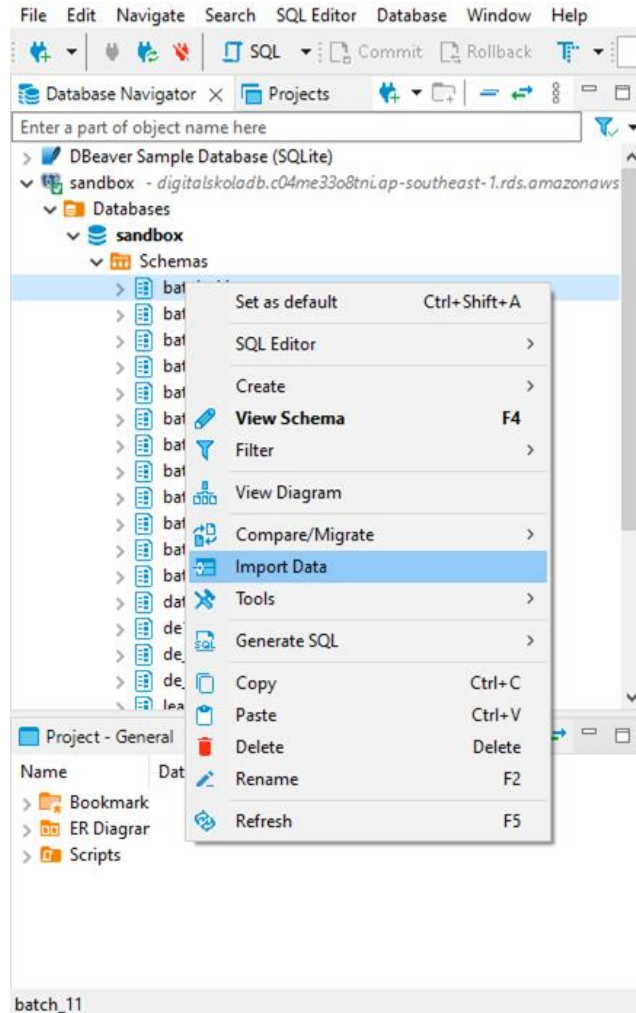
Tempat untuk **menyimpan** berbagai **tabel** yang berkaitan

Contoh
skema penjualan dan
produksi

Database

Tempat untuk **menyimpan** berbagai **skema** dari suatu perusahaan

Import Data



- DBeaver–PostgreSQL memiliki fitur *import data* dari **local directory** untuk file berformat **CSV**
- Data tersebut dapat digunakan sebagai **tabel baru** maupun mengisi **tabel lama**
- **Kesesuaian format** perlu diperhatikan, terutama bagian date-time dan nama kolom

Function – CREATE TABLE

- Digunakan untuk **membuat tabel** baru
- **Tipe data** dan **constraint** perlu diperhatikan

id	name	email

Syntax

```
CREATE TABLE IF NOT EXISTS table_name  
(  
    column1 data_type(length) column_constraint,  
    column2 data_type(length) column_constraint,  
    ...  
)
```

Contoh

```
CREATE TABLE IF NOT EXISTS customer  
(  
    id INT PRIMARY KEY,  
    name VARCHAR(50) NOT NULL,  
    email VARCHAR(255) UNIQUE NOT NULL  
)
```

Function – INSERT INTO

- Digunakan untuk **menambahkan data** pada tabel
- Penambahan data dapat secara **manual** ataupun dari **tabel lain**
- **Urutan** data perlu diperhatikan

id	name	email
1	Entropy	entropy@mymail.com
2	Team	team@mymail.com

Syntax

```
INSERT INTO table_name VALUES  
(data1, data2, data3, ...),  
(data1, data2, data3, ...)
```

```
INSERT INTO table_name (col1, col2, col3, ...) VALUES  
(data1, data2, data3, ...),  
(data1, data2, data3, ...)
```

Contoh

```
INSERT INTO customer VALUES  
(1, 'Entropy', 'entropy@mymail.com')
```

```
INSERT INTO customer (name, email, id) VALUES  
(Team, 'team@mymail.com', 2)
```

Function – UPDATE

- Digunakan untuk **mengubah data** pada tabel
- Kondisi perlu diperhatikan, karena tidak dapat di-*undo*

id	name	email
1	Entropy	entropy@mymail.com
2	Group	group@mymail.com

Syntax

```
INSERT table_name  
SET column1 = data1, column2 = data2, ...  
WHERE condition
```

Contoh

```
INSERT customer  
SET name = 'Group', column2 = 'group@mymail.com'  
WHERE id = 2
```

Function – ALTER TABLE

- Digunakan untuk **mengubah kolom**, seperti menambah atau menghapus kolom

id	name	email
1	Entropy	entropy@mymail.com
2	Group	group@mymail.com



id	name	email	Phone
1	Entropy	entropy@mymail.com	NULL
2	Group	group@mymail.com	NULL

Syntax

ALTER TABLE table_name **ADD** col1 datatype(length)

ALTER TABLE table_name **DROP COLUMN** col1

Contoh

ALTER TABLE customer **ADD** phone varchar(20)

Function – DELETE

- Digunakan untuk **menghapus sebagian baris** pada tabel

id	name	email
1	Entropy	entropy@mymail.com
2	Group	group@mymail.com



id	name	email
1	Entropy	entropy@mymail.com

Syntax

DELETE FROM table_name
WHERE condition

Contoh

DELETE FROM customer
WHERE id = 2

Function – TRUNCATE

- Digunakan untuk **menghapus semua baris** pada tabel

id	name	email
1	Entropy	entropy@mymail.com
2	Group	group@mymail.com



id	name	email

Syntax

TRUNCATE TABLE table_name

Contoh

TRUNCATE TABLE customer

Function – DROP

- Digunakan untuk **menghapus tabel** tanpa perlu kondisi

id	name	email
1	Entropy	entropy@mymail.com
2	Group	group@mymail.com



(tabel hilang)

Syntax

DROP TABLE table_name

Contoh

DROP TABLE customer

03

INTERMEDIATE SQL

Materi SQL menengah
(PostgreSQL)

Function Rule

- SQL memiliki **format** dan **urutan** yang harus diikuti
- Jika terdapat **kesalahan** format atau urutan, dapat menyebabkan **error**
- **Tidak semua** fungsi harus **digunakan** ketika melakukan *query*

Syntax

```
SELECT col1, col2, ...  
FROM table_name  
WHERE condition(s)  
GROUP BY field_name(s)  
HAVING condition(s)  
ORDER BY field_name(s)  
LIMIT number
```

Function – SELECT

- Digunakan untuk **memilih kolom** mana yang akan diambil datanya
- Simbol *asterisk* (*) digunakan untuk mengambil data dari semua kolom

id	name
1	Entropy
2	Group

Syntax

```
SELECT col1, col2, ...  
FROM table_name
```

```
SELECT *  
FROM table_name
```

Contoh

```
SELECT id, name  
FROM customer
```

Function – DISTINCT

- Digunakan untuk **menghapus** data **duplikat** pada suatu kolom

kelas		distinct
Data Science		Data Science
Data Science		
Data Analyst		Data Analyst
UI/ UX		UI/ UX
UI/ UX		
Web Dev		Web Dev
Data Analyst		

Syntax

```
SELECT DISTINCT col1  
FROM table_name
```

Contoh

```
SELECT DISTINCT kelas  
FROM enrollment
```

String Function

Fungsi	Syntax	Return	Contoh	Hasil
Menggabungkan teks	string1 string2	Teks	'En' 'tropy'	Entropy
	string num	Teks	'Nilai: ' 99	Nilai: 99
Menghitung panjang teks	char_length (string)	Angka	char_length('tea')	3
	character_length (string)	Angka	character_length('tea')	3
Mengubah teks menjadi <i>lower case</i>	lower (string)	Teks	lower('TeA')	tea
Mengubah teks menjadi <i>upper case</i>	upper (string)	Teks	upper('tEa')	TEA
Mengetahui lokasi suatu <i>substring</i>	position (substring in string)	Angka	position('ea' in 'tea')	2
Mengekstrak <i>substring</i>	substring (string from int for int)	Teks	substring('tea' from 2 for 2)	ea

Aggregation Function

Fungsi	Syntax
Menghitung rata-rata semua baris pada suatu kolom	AVG (column_name)
Menghitung jumlah baris pada tabel	COUNT (*)
Menghitung jumlah baris pada kolom tertentu	COUNT (column_name)
Mencari nilai terbesar pada suatu kolom	MAX (column_name)
Mencari nilai terkecil pada suatu kolom	MIN (column_name)
Menghitung jumlah kumulatif pada suatu kolom	SUM (column_name)

Function – CASE

- Digunakan untuk **mengevaluasi** suatu kondisi dan menentukan hasilnya

nilai	kelulusan
70	Tidak lulus
90	Lulus
80	Lulus
100	Lulus
60	Tidak lulus
0	Tidak lulus
90	Lulus

Syntax

```
CASE
  WHEN condition1 THEN result1
  WHEN condition2 THEN result2
  ...
  WHEN conditionN THEN resultN
  ELSE result
END AS column_alias_name
```

Contoh

```
CASE
  WHEN nilai > 75 THEN 'Lulus'
  ELSE 'Tidak lulus'
END AS kelulusan
```

Function – WHERE

- Digunakan untuk **memfilter** data berdasarkan 1 kondisi maupun beberapa kondisi

nilai		nilai
70		
90		
80	→	90
100		80
60		90
0		
90		

Syntax

```
SELECT col1, col2, ...  
FROM table_name  
WHERE condition(s)
```

Contoh

```
SELECT nilai  
FROM enrollment  
WHERE (nilai >= 80) AND (nilai != 100)
```

Function – GROUP BY

- Digunakan bersama **aggregation function** untuk **mengelompokkan** data berdasarkan suatu kolom
- Misal dari tabel penjualan, terdapat 2 jenis barang yang dijual

barang	total_barang
Meja	60
Kursi	40

Syntax

```
SELECT col1, col2, ...  
FROM table_name  
GROUP BY col1, col2, ...
```

Contoh

```
SELECT  
    barang,  
    SUM(jumlah_barang) AS total_barang  
FROM penjualan  
GROUP BY barang
```

Function – HAVING

- Digunakan untuk **memfilter data** yang telah **diagregasi** berdasarkan 1 kondisi maupun beberapa kondisi

barang	total_barang
Meja	60
Kursi	40



barang	total_barang
Meja	60

Syntax

```
SELECT col1, col2, ...  
FROM table_name  
GROUP BY col1, col2, ...  
HAVING condition(s)
```

Contoh

```
SELECT  
    barang,  
    SUM(jumlah_barang) AS total_barang  
FROM penjualan  
GROUP BY barang  
HAVING SUM(jumlah_barang) > 50
```

Function – ORDER BY

- Digunakan untuk **mengurutkan** data berdasarkan 1 kolom atau lebih

nilai		nilai
70		0
90		60
80		70
100	→	80
60		90
0		90
90		100

Syntax

```
SELECT col1, col2, ...  
FROM table_name  
ORDER BY col1, col2, ...
```


Contoh

```
SELECT nilai  
FROM enrollment  
ORDER BY nilai
```

Function – LIMIT

- Digunakan untuk **membatasi** jumlah data yang akan di-*query*

nilai
70
90
80
100
60
0
90



nilai
70
90
80

Syntax

```
SELECT col1, col2, ...  
FROM table_name  
LIMIT number
```

Contoh

```
SELECT nilai  
FROM enrollment  
LIMIT 3
```

Urutan Pemrosesan Query

Urutan Penulisan Fungsi

- SELECT** •
- FROM** •
- WHERE** •
- GROUP BY** •
- HAVING** •
- ORDER BY** •
- LIMIT** •



Urutan Pemrosesan Fungsi

- Mengambil data (**FROM, JOIN**)
- Filter baris (**WHERE**)
- Pengelompokan (**GROUP BY**)
- Filter grup (**HAVING**)
- Pemilihan data (**SELECT**)
- Pengururan dan pembatasan (**ORDER BY & LIMIT**)

THANKS

Entropy Team

CREDITS: This presentation template was originally created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**