LEARNING PROGRESS REVIEW

WEEK 2

Syntax Group

TABLE OF CONTENT

01 LEARNED SQL BASIC SKILLS

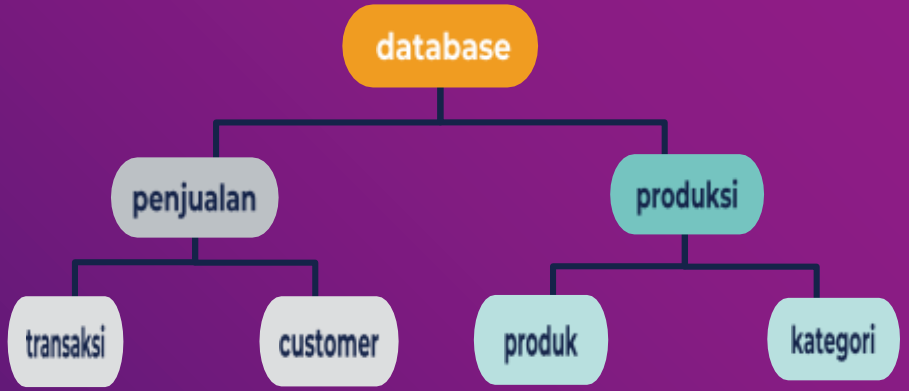
let's get started!

SCHEMA & TABLE

Table
Tempat **menyimpan** berbagai **data** secara **tabular contoh t**abel transaksi, customer, produk dan kategori

Schema
Tempat untuk **menyimpan** berbagai **tabel** yang berkaitan contoh skema penjulan dan produksi

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



SQL "STUCTURED QUERY LANGUAGE"

SQL bahasa pemrograman yang digunakan dalam mengakses, mengubah, dan memanipulasi data yang berbasis relasional berdasarkan Standar American National Standard Institute (ANSI)

Contoh database SQL :

- Postgre SQL
- My SQL
- Oracle
- ODBC

DDL

Data Definition Languange adalah perintah yang digunakan untufi mendefinisifian data seperti membuat tabel database baru, mengubah dataset, dan menghapus data. **Create, Alter, Rename, Drop, Show**

DML

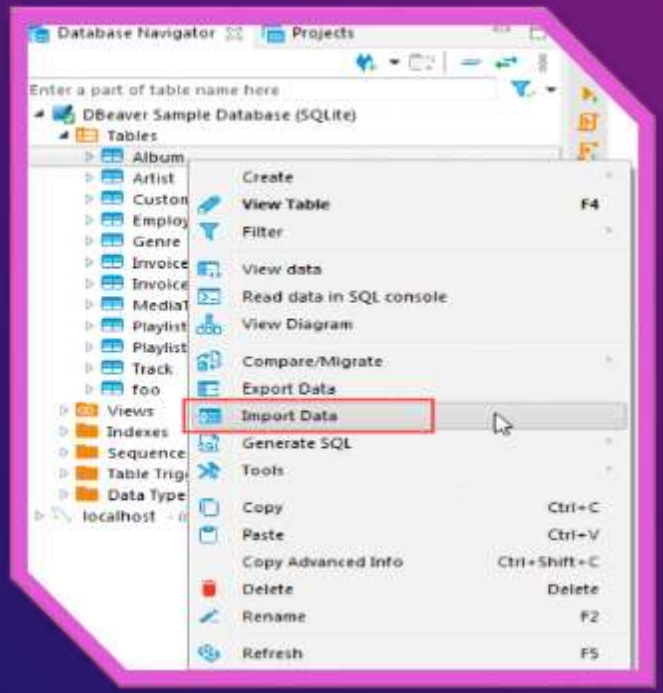
Data Manipulation Language merupakan perintah yang digunakan untuk memanipulasi data**. Insert, select, update, dan delete.**

DCL

Data Control Language berfungsi untuk melalui pegontrolan data. Pengontrolan yang dimaksud di sini meliputi pengontrolan terhadap hak user dan pengontrolan terhadap suatu transaksi, apakah akan disimpan secara permanen file dalam database atau akan dibatalkan**. Grant, Revoke**

JENIS PERINTAH SQL

IMPORT DATA



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

CREATE TABLE

```
CREATE TABLE table_name (
        Nama_Kolom_1 Tipe_Data_1,
        Nama_Kolom_2 Tipe_Data_2,
        Nama_Kolom_3 Tipe_Data_3
);
```

UPDATE TABLE

```
UPDATE table_name SET column1 = value1 WHERE condition;
```

INSERT TABLE

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

DELETE DATA

```
DELETE FROM table_name WHERE condition;
```

SHOW DATA

```
SELECT *
FROM pegawai;
```

```
SELECT nama_colum1, nama_column2, ...
FROM table_name;
```

PERINTAH DASAR

CREATE

CREATE TABLE *nama_tabel (kolom 1 tipedata 1, kolom 2 tipedata 2, ...)*

```
CREATE TABLE
sandbox.learning.users (
        id INT
        nama VARCHAR birthdate  DATE);
```

	123 id	ABC nama	birthdate

INSERT INTO

Mengisi seluruh kolom :

INSERT INTO *nama_tabel* VALUES *(data 1, data 2, ...), (data 1, data 2, ...)*

```
INSERT INTO
sandbox.learning.users
VALUES
(1, 'Udin', '1990-01-01'),
(2, 'Usep', '1991-01-01');
```

	123 id	ABC nama	birthdate
1	1	Udin	1990-01-01
2	2	Usep	1991-01-01

INSERT INTO

Mengisi kolom tertentu :

INSERT INTO *nama_tabel (column 1, column 2, ...)* VALUES *(data 1, data 2, ...), (data 1, data 2, ...)*

```
INSERT INTO
sandbox.learning.users  (id,
nama)
VALUES
(3, 'Udin'),
(4, 'Usep');
```

	123 id	ABC nama	birthdate
1	1	Udin	1990-01-01
2	2	Usep	1991-01-01
3	3	Alex	[NULL]
4	4	Asep	[NULL]

UPDATE

Mengedit tabel :

INSERT *nama_tabel*
SET column1=data1, column2=data2,..
WHERE *condition*

```
UPDATE
sandbox.learning.users
SELECT birthdate =
'1997-01-01'WHERE id = 3
```

123 id	ABC nama	birthdate
1	Udin	1990-01-01
2	Usep	1991-01-01
3	Alex	[NULL]
4	Asep	1997-01-01

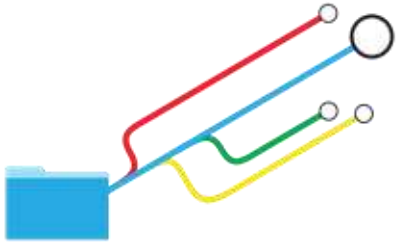
ALTER

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

Mengubah kolom:

```
ALTER TABLE nama_table ADD col1 datatype(length)
ALTER TABLE nama_table DROP COLUMN col1
```

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



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

DELETE

Menghapus baris dengan id 3 :

```
DELETE FROM sandbox.learning.users
WHERE id = 3
```

123 id	ABC nama	birthdate
1	Udin	1990-01-01
2	Usep	1991-01-01
4	Asep	1997-01-01

TRUNCATE

Digunakan untuk **menghapus semua baris** pada tabel

Menghapus baris:

```
TRUNCATE TABLE nama_table
```

```
TRUNCATE TABLE customer
```

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

id	name	email	Phone

DROP

Digunakan untuk **menghapus tabel** tanpa perlu kondisi

Menghapus tabel:

DROP TABLE *nama_table*

```
DROP TABLE customer
```

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

(tabel hilang)

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
```

FUNGSI LIMIT

OPERATOR SQL

Operator	Deskripsi
=	Sama dengan
>	Lebih dari
<	Kurang dari
>=	Lebih dari sama dengan
<=	Kurang dari sama dengan
<> or !=	Tidak sama dengan
BETWEEN	Di antara rentang nilai tertentu
LIKE	Mencari pola
IN	Mencari lebih dari satu nilai spesifik

02

LEARNED SQL
INTERMEDIATE SKILLS

let's get started!

DISTINC

Mengembalikan hanya nilai yang berbeda dari dalam sebuah tabel

```
SELECT DISTINCT Column1, column2, …
FROM table_name ;
```

```
1  SELECT DISTINCT customer_id
2  FROM customers;
```

Data Output | Messages | Notifications | Explain

	customer_id integer
1	1002
2	1004
3	1003
4	1001

WHERE

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

Memfilter data:

```
SELECT col1,col2,...
FROM nama_table WHERE condition
```

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

nilai
70
90
80
100
60
0
90

nilai
90
80
90

HAVING

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

Memfilter data:

```
SELECT col1,col2,...
FROM nama_table GROUP BY col1,col2,...
HAVING condition
```

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

barang	total_barang
Meja	60
Kursi	40

↓

barang	total_barang
Meja	60

FUNGSI STRING

SUBSTRING:	Mengambil char.
REVERSE:	Membalikkan kata.
REPLACE:	Mengganti nilai char.
LTRIM:	Menghilangkan spasi dari kiri.
RTRIM:	Menghilangkan spasi dari kanan.
LOWER:	Membuat semua huruf menjadi huruf kecil.
UPPER:	Membuat semua huruf menjadi huruf kapital.
LEN:	Mengembalikan banyak char.
DATALENGTH:	Mengembalikan banyak digit angka.
LEFT:	Mengambil char dari kiri, untuk index ke 1 - n.
RIGHT:	Mengambil char dari kanan, untuk index ke 1 - n.
CONCAT:	Menggabungkan string.

FUNGSI STRING (LANJUTAN)

ASCII:	Mengembalikan nilai ASCII.
CHAR:	Mengkonversi nilai integer menjadi nilai char.
CHARINDEX:	Mengembalikan posisi indeks suatu char.
PATINDEX:	Mengembalikan posisi indeks char berdasarkan pola.
REPLICATE:	Menggabungkan input string yang ditulis secara berulang sebanyak n.
SPACE:	Mengembalikan char spasi sebanyak n.
STUFF:	Mengganti beberapa bagian dari string.
UNICODE:	Mengembalikan nilai ASCII dari char index pertama.
NCHAR:	Mengkonversi nilai ASCII menjadi nilai char.

STRING FUNCTION

LOWER

Fungsi untuk **mengubah string menjadi huruf kecil**

Command string "Digital Skola"
LOWER ('Digital Skola')
Output:
'digital sfiola'

UPPER

Fungsi untuk **mengubah string menjadi huruf kapital**

Command string "Digital Skola"
UPPER ('Digital Skola')
Output:
'DIGITAL SKOLA"

LENGTH

Fungsi untuk **menghitung panjang karakter string**

Command string "Digital Skola"
LENGTH ('Digital Skola')
Output:
13

STRING FUNCTION

CONCAT

Fungsi untuk **menambahkan string** satu dengan yang lain

Command string
CONCAT ('Skola' , '-' , 'Batch26')
Output:
'Skola-Batch26'

SUBSTRING

Fungsi untuk **mengekstrak beberapa karakter** string

CONCAT (nama_Kolom, index_awal, jumlah Karakter)

Command string
CONCAT (DigitalSkola, 1, 7)
Output: **'Digital'**

Logical Operator digunakan untuk menguji kebenaran dari suatu kondisi. Berikut beberapa contoh yang sering digunakan:

AND:	Benar, jika kondisi yang dipisahkan oleh **AND** sesuai.	SELECT * FROM table_name WHERE condition1 AND condition2
LIKE:	Benar, jika kondisi record sesuai pola yang diberikan.	SELECT Name FROM Suppliers WHERE Name LIKE 'Ca%'
BETWEEN:	Benar, jika kondisi record berada pada range yang diberikan.	SELECT * FROM table_name WHERE column_name BETWEEN value1 AND value2
OR:	Benar, jika kondisi yang dipisahkan oleh **OR** sesuai.	SELECT * FROM table_name WHERE condition1 OR condition2
NOT:	Kebalikan dari kondisi yang diberikan.	SELECT * FROM table_name WHERE NOT condition1

Fungsi Agregasi salah satu fungsi terpenting di dalam SQL, fungsi ini memungkinkan kita melakukan beberapa proses statistik seperti:

COUNT:	Menghitung banyaknya data
AVG:	Menghitung rerata nilai dari data
SUM:	Mencari jumlah total nilai dari sebuah data.
MAX:	Mencari nilai terbesar dri sebuah data
MIN:	Mencari nilai terkecil dari sebuah data

Fungsi Agregasi biasanya digunakan dengan fungsi **GROUP BY** maupun fungsi **ORDER BY.**

GROUP BY	Fungsi ini membagi data menjadi beberapa group yang selanjutnya bisa dilakukan fungsi agregasi.
ORDER BY	Fungsi ini digunakan untuk mengurutkan data dari urutan tertinggi maupun urutan terendah.

FUNGSI AGREGASI, GROUP BY & ORDER BY

A. COUNT

```
SELECT COUNT(column_name)
FROM table_name
[GROUP BY column_name];
```

B. AVG

```
SELECT AVG(column_name)
FROM table_name
[GROUP BY column_name];
```

```
SELECT
    COUNT(city.lat) AS lat_count,
    SUM(city.lat) AS lat_sum,
    AVG(city.lat) AS lat_avg,
    MIN(city.lat) AS lat_min,
    MAX(city.lat) AS lat_max
FROM city;
```

100 %

Results | Messages

	lat_count	lat_sum	lat_avg	lat_min	lat_max
1	6	270.142498	45.023749	34.052235	52.520008

C. SUM

```
SELECT SUM(column_name)
FROM table_name
[GROUP BY column_name];
```

D. MAX

```
SELECT MAX(column_name)
FROM table_name
[GROUP BY column_name];
```

E. MIN

```
SELECT MIN(column_name)
FROM table_name
[GROUP BY column_name];
```

FUNGSI AGREGASI

GROUP BY

Misalkan 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
```

Syntax

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

Contoh

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

Seperti contoh dibawah **ORDER BY** digunakan untuk mengurutkan hasil dalam urutan menaik atau menurut

nilai
70
90
80
100
60
0
90

nilai
0
60
70
80
90
90
100

ORDER BY

Seperti contoh disamping **ORDER BY** digunakan untuk mengurutkan hasil dalam urutan menaik atau menurut serta dapat mengurutkan hasil yang diambil setelah menggunakan klausa **SELECT** dengan **GROUP BY**

```
SELECT SUM(emp_sal_paid) As "Total Salary", emp_name
FROM sto_emp_salary_paid
GROUP BY emp_name
ORDER BY emp_name;
```

125 %

Results | Messages

	emp_id	emp_name	emp_sal_paid	date_paid
1	1	Mike	4000.75	2016-07-01 00:00:00.000
2	1	Mike	4500.55	2017-02-01 00:00:00.000
3	1	Mike	5000.35	2017-03-01 00:00:00.000
4	2	Michale	4500.56	2017-06-01 00:00:00.000
5	2	Michale	4500.77	2017-07-01 00:00:00.000
6	3	Jimmy	3000.86	2017-06-01 00:00:00.000
7	3	Jimmy	3500.75	2017-07-01 00:00:00.000
8	4	Shaun	3500.45	2017-06-01 00:00:00.000
9	5	Ben	4500.50	2017-03-01 00:00:00.000

	Total Salary	emp_name
1	4500.50	Ben
2	6501.61	Jimmy
3	9001.33	Michale
4	13501.65	Mike
5	3500.45	Shaun

GROUP BY & ORDER BY

GROUP BY... ORDER BY

Urutan Penulisan Fungsi

SELECT

FROM

WHERE

GROUP BY

HAVING

ORDER BY

LIMIT

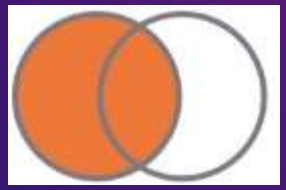
Urutan Pemrosesan Fungsi

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

URUTAN PEMROSESAN QUERY

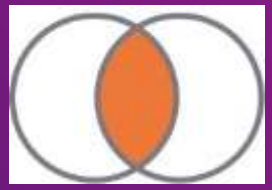
Fungsi **JOIN** digunakan untuk menggabungkan row dari dua maupun lebih tabel. Berikut beberapa kondisi pada fungsi **JOIN:**

LEFT JOIN: Menampilkan semua records pada tabel kiri digabungkan dengan semua records yang **sama** pada tabel kanan.



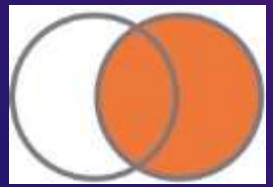
```
SELECT *
FROM left_table
LEFT JOIN right_table
ON left_table.key = right_table.key
```

INNER JOIN: Menampilkan hanya records yang sama pada tabel kiri dan tabel kanan.



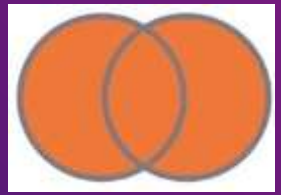
```
SELECT *
FROM left_table
INNER JOIN right_table
ON left_table.key = right_table.key
```

RIGHT JOIN: Menampilkan semua records pada tabel kanan digabungkan dengan semua records yang **sama** pada tabel kiri.



```
SELECT *
FROM left_table
RIGHT JOIN right_table
ON left_table.key = right_table.key
```

OUTER JOIN: Menampilkan semua records pada tabel kiri digabungkan dengan semua records pada tabel kanan.



```
SELECT *
FROM left_table
OUTER JOIN right_table
ON left_table.key = right_table.key
```

JOIN TABLE

01

Subqueries pada Filter (WHERE)

Sebuah subquery dalam klausa WHERE dapat digunakan untuk memenuhi syarat kolom terhadap satu set baris.
Misalnya, subquery berikut mengembalikan nomor departemen untuk departemen di lantai tiga. Kueri luar mengambil nama-nama karyawan yang bekerja di lantai tiga.

contoh sebagai berikut:

```
SELECT ename
FROM employee
WHERE dept IN
        (SELECT dno
        FROM dept
        WHERE floor = 3);
```

02

Subqueries pada TABEL

Melakukan subqueries tidak hanya dapat dilakukan pada filter namun, dapat juga dilakukan pada hasil query yang sudah ada.

contoh sebagai berikut:

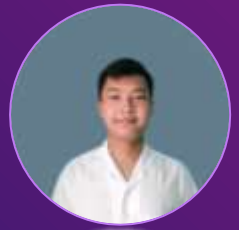
```
SELECT MAX(avg_amount_transaksi) AS max_avg_amount_transaksi
FROM ( SELECT hari_transaski,
AVG(amount_transksi) AS avg_amount_transaksi
FROM transaksi
GROUP BY hari_transaksi);
```

SYNTAX GROUP

Aldrich Alfatera Unpapar

Wa Ode Sukmasarny Musdigaswati

Patma Oktaviana



Otniel Sukma Priyambodo

Jason Tadeus

Thank You