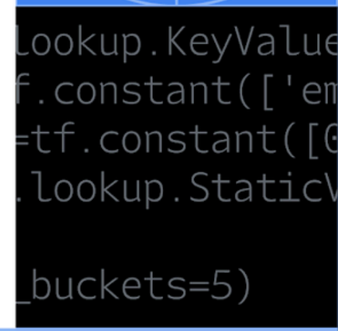
Google Developer Groups

Advanced Data Manipulation and Queries

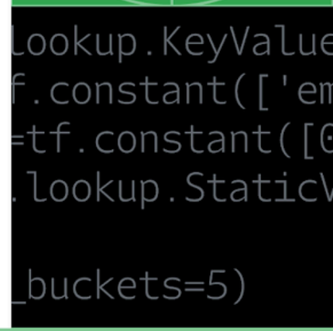


Overview

1. Filtering and Sorting Data
2. Data Aggregation and Grouping
3. Join and Subquery

Google Developer Groups

Filtering and Sorting Data



Operator BETWEEN

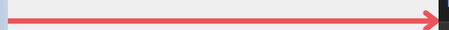
- **Syntax**

```
SELECT
    column_name(s)
FROM
    table_name
WHERE
    column_name BETWEEN value1 AND value2;
```

Operator BETWEEN digunakan untuk menyaring data numerik, memilih nilai dalam rentang tertentu.

Operator BETWEEN

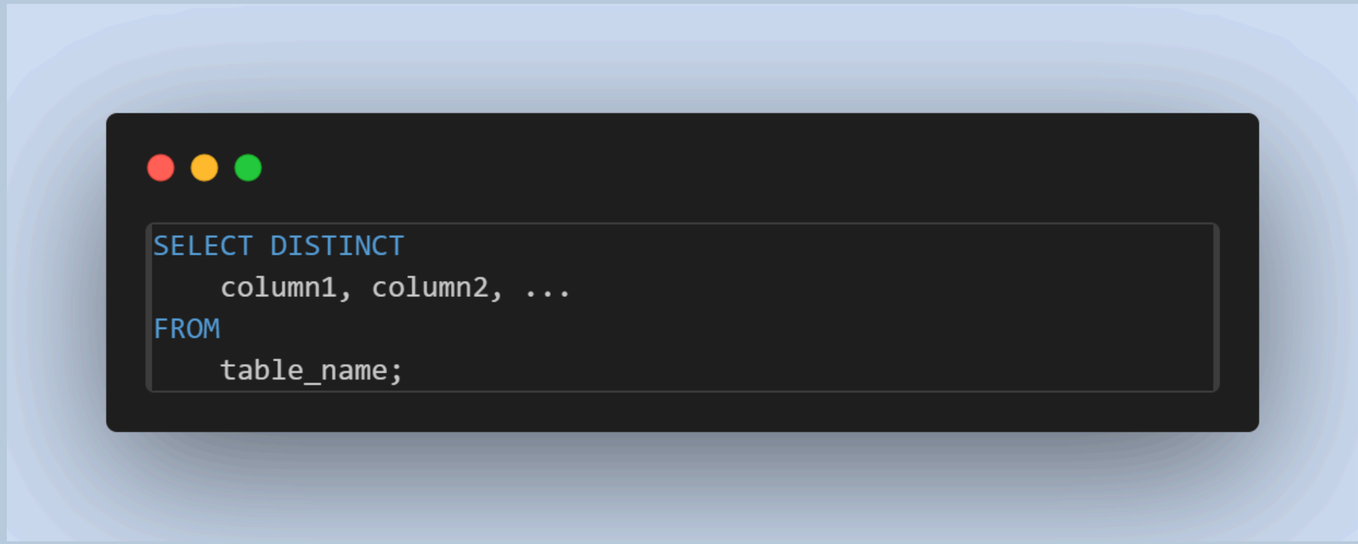
```sql
select
    *
from
    superstore
where
    sales between 100 and 400;
```



customer_id character varying	postal_code character varying	product_id character varying	sales double precision
CG-12520	42420	FUR-BO-10001798	261.96
BH-11710	90032	OFF-AP-10002892	114.9
ZD-21925	94109	TEC-PH-10001949	213.48
TB-21520	19140	FUR-FU-10004848	124.2
GH-14485	75080	FUR-FU-10003664	190.92
SN-20710	77041	OFF-EN-10002986	113.328
SN-20710	77041	FUR-CH-10004218	212.058
SN-20710	77041	TEC-PH-10000486	371.168
LC-16930	60540	TEC-PH-10004093	147.168

SELECT DISTINCT

- **Syntax**



```
SELECT DISTINCT
    column1, column2, ...
FROM
    table_name;
```

SELECT DISTINCT menghilangkan baris duplikat dari hasil.

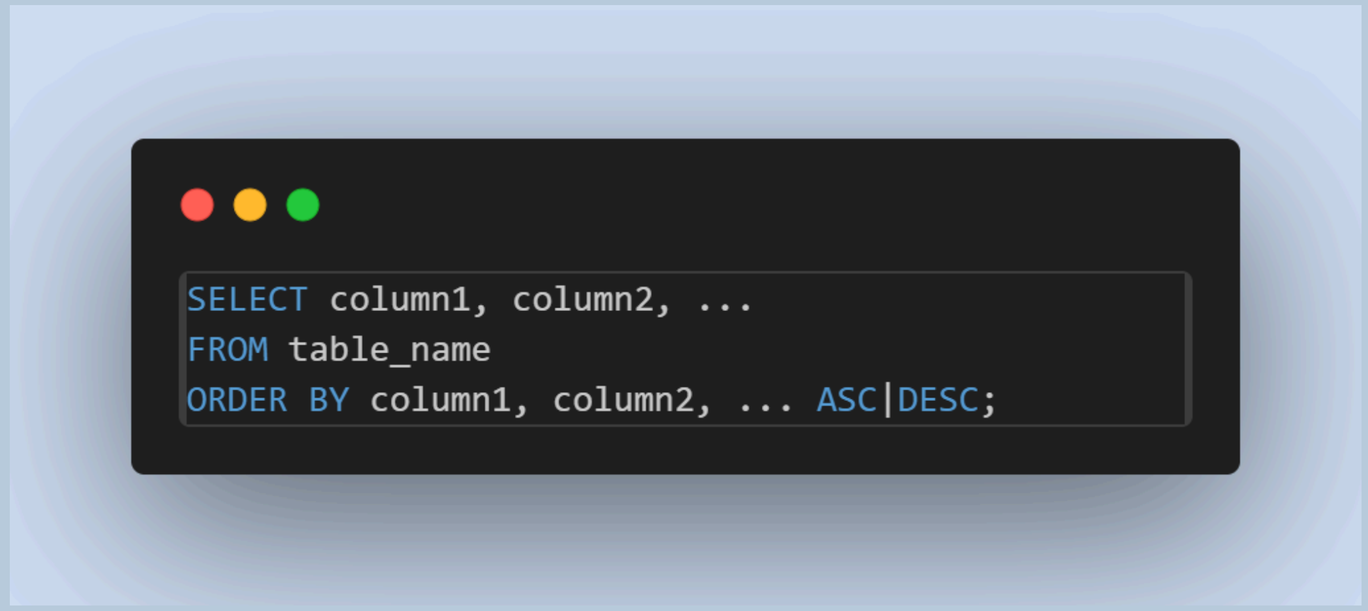
SELECT DISTINCT

```
select distinct
    state
from
    superstore;
```


	state character varying
1	Oklahoma
2	Colorado
3	North Carolina
4	Mississippi
5	Florida
6	Vermont
7	Delaware
8	Nevada
9	Louisiana
10	New York

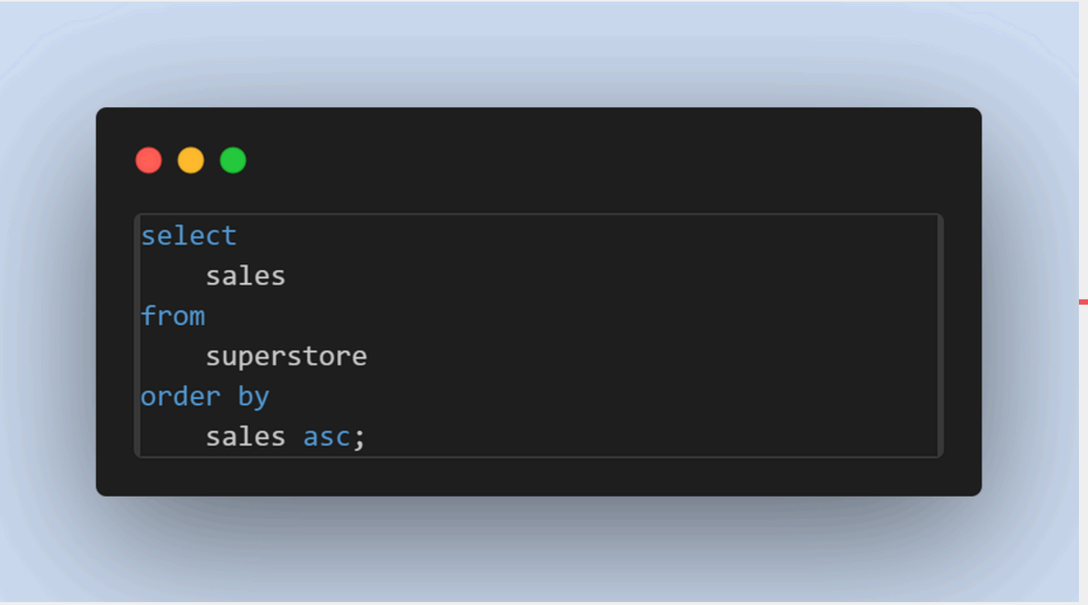
ORDER BY

- **Syntax**



ORDER BY digunakan untuk mengurutkan kumpulan hasil dalam urutan naik atau turun.

ORDER BY

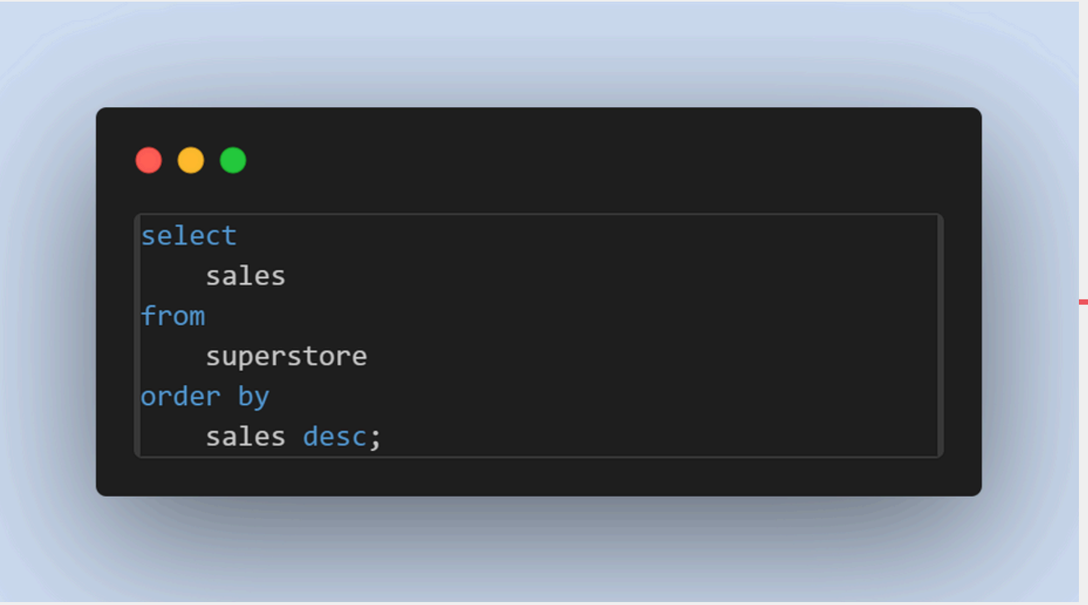


```
select
    sales
from
    superstore
order by
    sales asc;
```

Asc/Ascending: mengurutkan kolom dari nilai yang paling kecil atau mengurutkan secara naik.

	sales double precision
1	0.444
2	0.556
3	0.836
4	0.852
5	0.876
6	0.898
7	0.984
8	0.99
9	1.044
10	1.08

ORDER BY



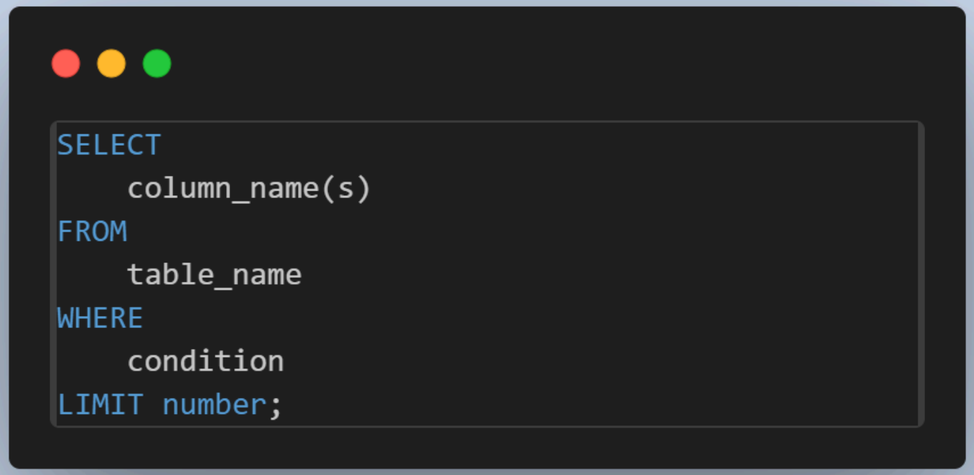
```
select
    sales
from
    superstore
order by
    sales desc;
```

Desc/Descending: mengurutkan kolom dari nilai yang paling besar atau mengurutkan secara turun.

	sales double precision
1	22638.48
2	17499.95
3	13999.96
4	11199.968
5	10499.97
6	9892.74
7	9449.95
8	9099.93
9	8749.95
10	8399.976

LIMIT

- **Syntax**

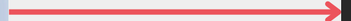


```
SELECT
    column_name(s)
FROM
    table_name
WHERE
    condition
LIMIT number;
```

LIMIT digunakan untuk membatasi jumlah maksimum row.

LIMIT

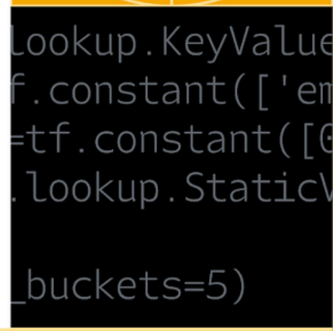
```
select
    *
from
    superstore
limit 5;
```



	order_id character varying	customer_id character varying	postal_code character varying
1	CA-2019-152156	CG-12520	42420
2	CA-2019-152156	CG-12520	42420
3	CA-2019-138688	DV-13045	90036
4	US-2018-108966	SO-20335	33311
5	US-2018-108966	SO-20335	33311

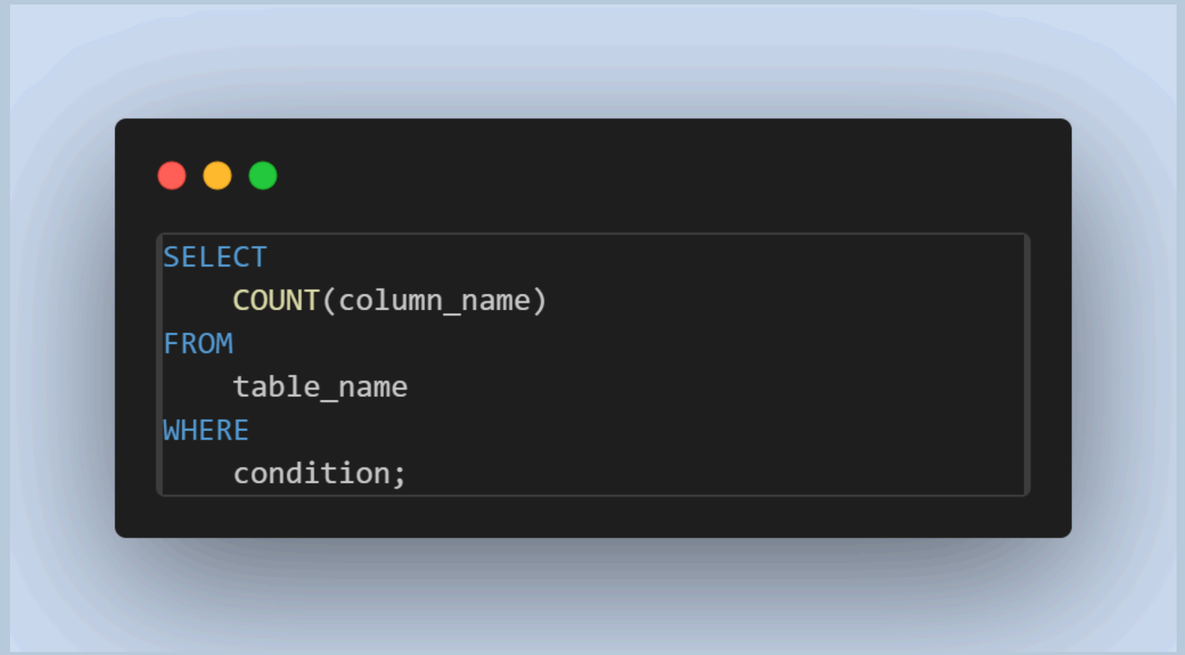
Google Developer Groups

Data Aggregation and Grouping



Aggregate Function: COUNT

- **Syntax**

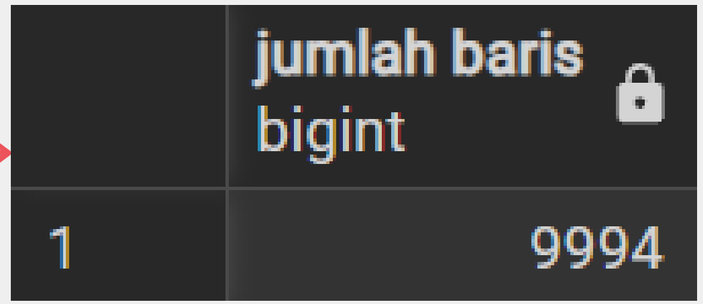


```
SELECT
    COUNT(column_name)
FROM
    table_name
WHERE
    condition;
```

COUNT adalah fungsi untuk menghitung jumlah baris dalam tabel.

Aggregate Function: COUNT

```
select
    count(*) as "jumlah baris"
from
    superstore;
```

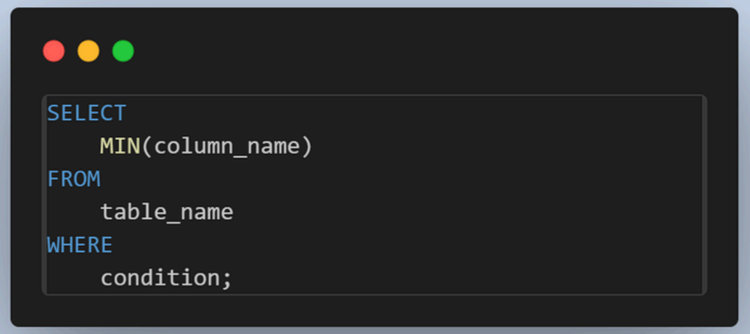


COUNT DISTINCT digunakan untuk menghitung jumlah baris yang berbeda

Aggregate Function: MIN, MAX

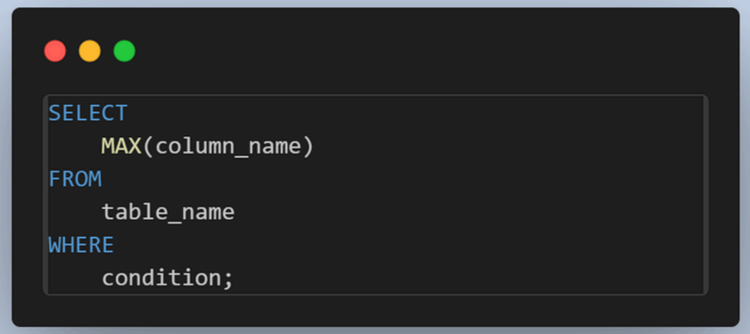
- **Syntax**

MIN



```sql
SELECT
    MIN(column_name)
FROM
    table_name
WHERE
    condition;
```

MAX

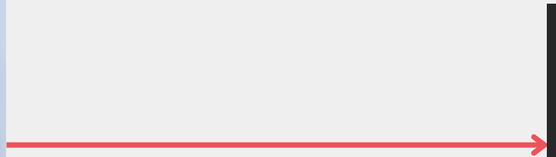


```sql
SELECT
    MAX(column_name)
FROM
    table_name
WHERE
    condition;
```

- MIN mengambil nilai minimum.
- MAX mengambil nilai minimum.

Aggregate Function: MIN

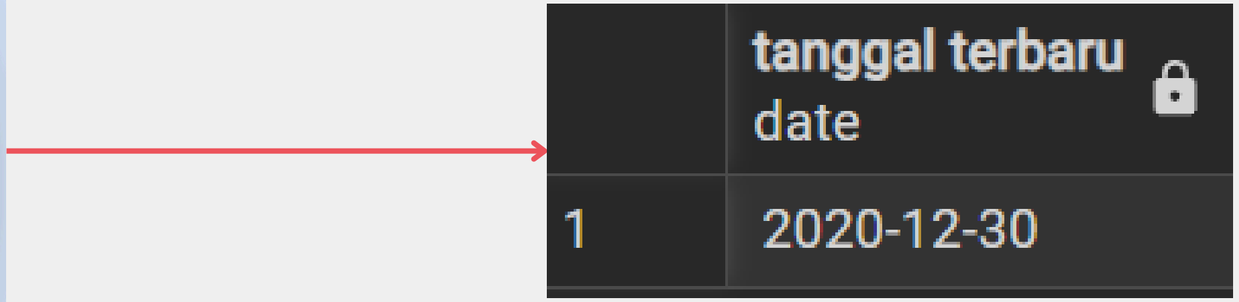
```sql
select
    min(order_date) as "tanggal terlama"
from
    superstore;
```



	tanggal terlama date
1	2017-01-03

Aggregate Function: MAX

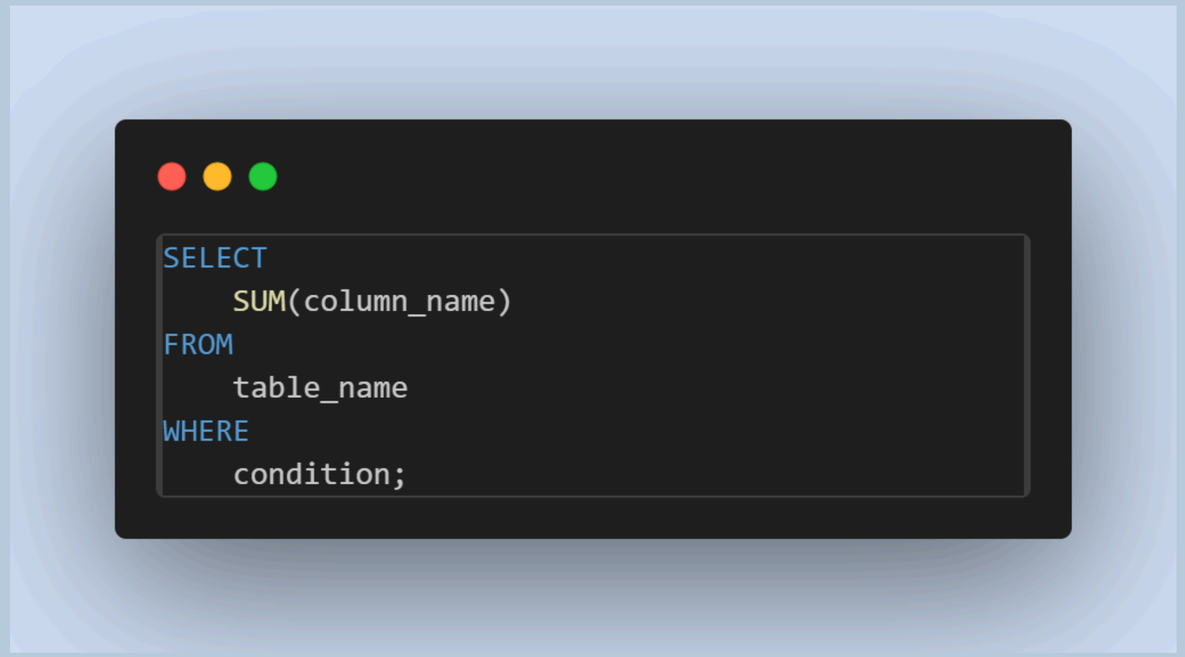
```sql
select
    max(order_date) as "tanggal terlama"
from
    superstore;
```



	tanggal terbaru date
1	2020-12-30

Aggregate Function: SUM

- **Syntax**



```
SELECT
    SUM(column_name)
FROM
    table_name
WHERE
    condition;
```

SUM menghitung jumlah total kolom numerik.

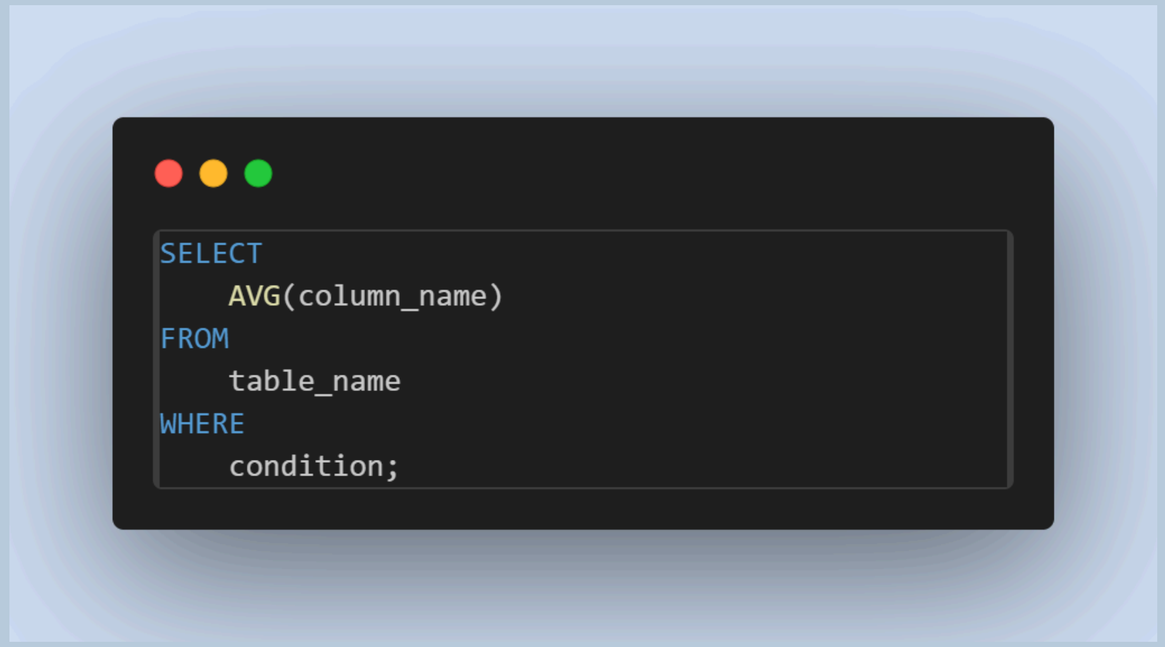
Aggregate Function: SUM

```
select
    sum(profit) as "total profit"
from
    superstore;
```

	total profit double precision
1	286397.0217000013

Aggregate Function: AVG

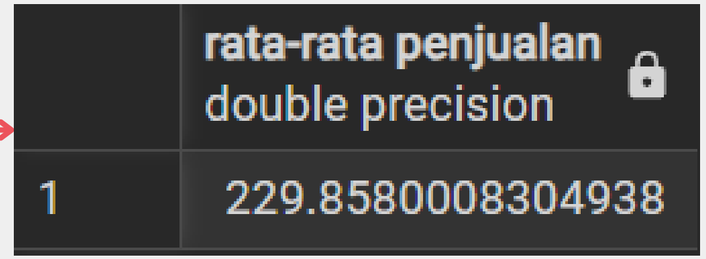
- **Syntax**



AVG menghitung rata-rata kolom numerik.

Aggregate Function: AVG

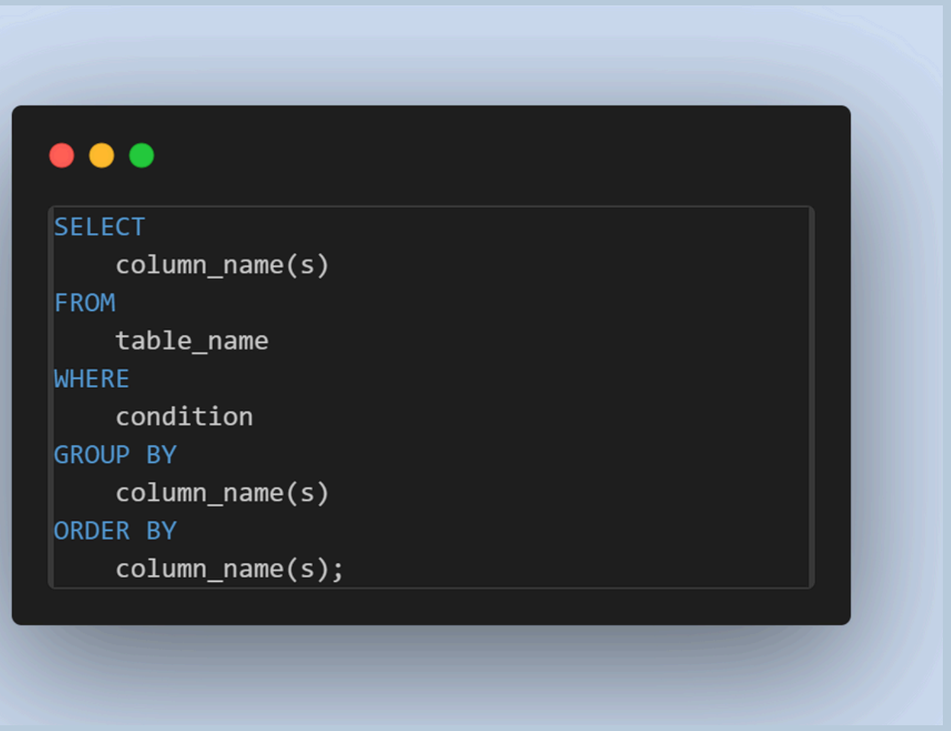
```sql
select
    avg(sales) as "rata-rata penjualan"
from
    superstore;
```



	rata-rata penjualan double precision
1	229.8580008304938

GROUP BY

- **Syntax**

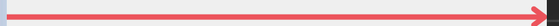


```
SELECT
    column_name(s)
FROM
    table_name
WHERE
    condition
GROUP BY
    column_name(s)
ORDER BY
    column_name(s);
```

GROUP BY untuk mengelompokkan baris berdasarkan nilai satu atau lebih kolom, biasanya digunakan dengan fungsi aggregat

GROUP BY

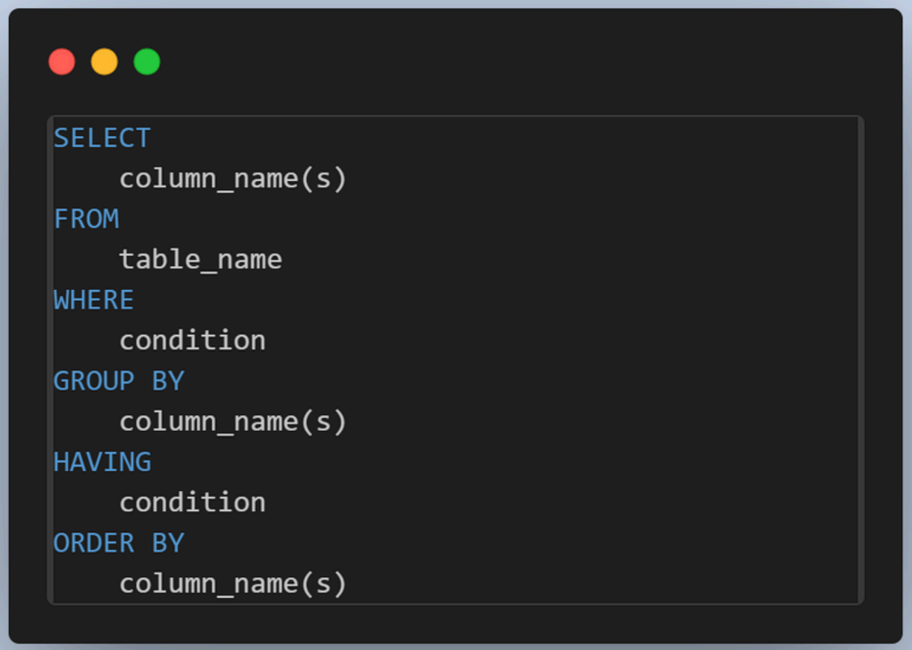
```sql
select
    customer_name, count(order_id) as
"banyak order"
from
    superstore
group by
    customer_name
order by
    customer_name;
```



	customer_name character varying	banyak order bigint
1	Aaron Bergman	6
2	Aaron Hawkins	11
3	Aaron Smayling	10
4	Adam Bellavance	18
5	Adam Hart	20
6	Adam Shillingsburg	25
7	Adrian Barton	20
8	Adrian Hane	16

HAVING

- **Syntax**

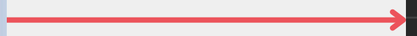


```
SELECT
    column_name(s)
FROM
    table_name
WHERE
    condition
GROUP BY
    column_name(s)
HAVING
    condition
ORDER BY
    column_name(s)
```

HAVING dan WHERE adalah fungsi yang sama, yang membedakannya adalah WHERE tidak dapat digunakan untuk fungsi aggregat sedangkan HAVING bisa digunakan.

HAVING

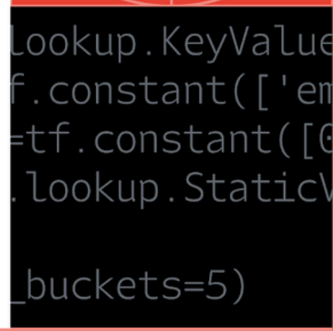
```sql
select
    customer_name, segment, count(order_id) as
"banyak order"
from
    superstore
where
    segment = 'Home Office'
group by
    customer_name, segment
having
    count(order_id) > 10
order by
    customer_name;
```



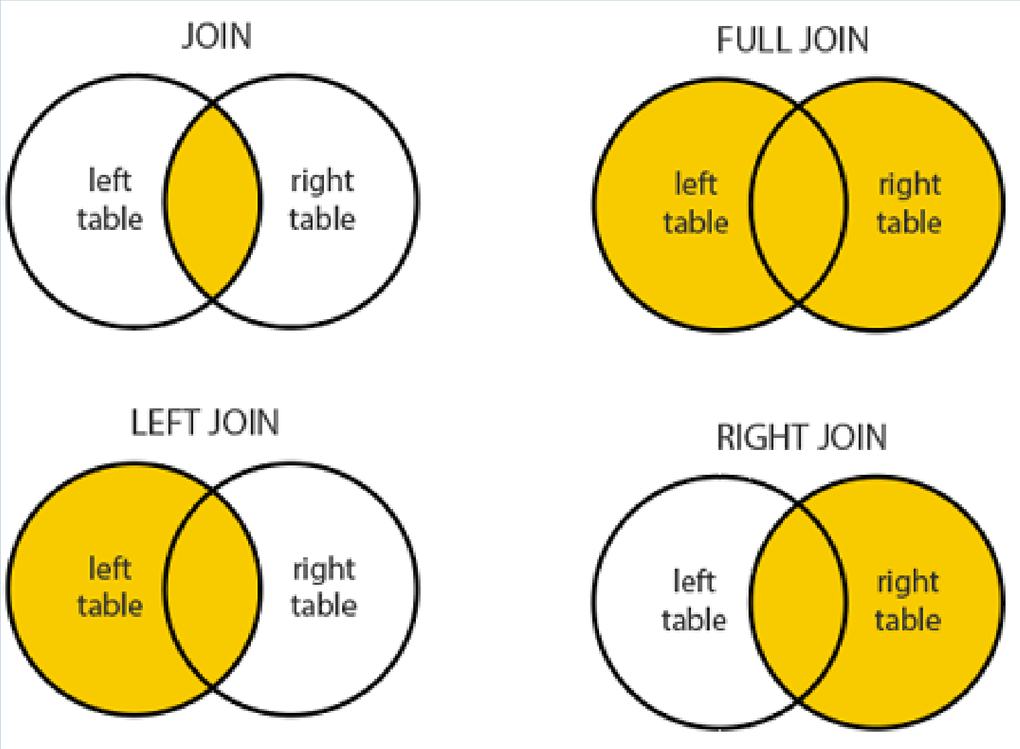
	customer_name character varying	segment character varying	banyak order bigint
1	Adam Bellavance	Home Office	18
2	Adrian Hane	Home Office	16
3	Alan Dominguez	Home Office	12
4	Alejandro Ballentine	Home Office	14
5	Ann Steele	Home Office	12
6	Anne Pryor	Home Office	19
7	Arthur Wiediger	Home Office	16
8	Becky Castell	Home Office	18

Google Developer Groups

Join and Subquery



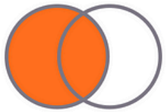
SQL JOIN (INNER, LEFT, RIGHT, FULL)



- **(INNER) JOIN:** Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN:** Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN:** Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN:** Returns all records when there is a match in either left or right table

SQL JOIN (INNER, LEFT, RIGHT, FULL)

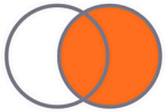
LEFT JOIN



Everything on the left
+
anything on the right that matches

```
SELECT *
FROM TABLE_1
LEFT JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
```

RIGHT JOIN



Everything on the right
+
anything on the left that matches

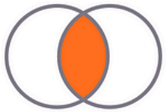
```
SELECT *
FROM TABLE_1
RIGHT JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
```

OUTER JOIN

Everything on the right
+
Everything on the left

```
SELECT *
FROM TABLE_1
OUTER JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
```

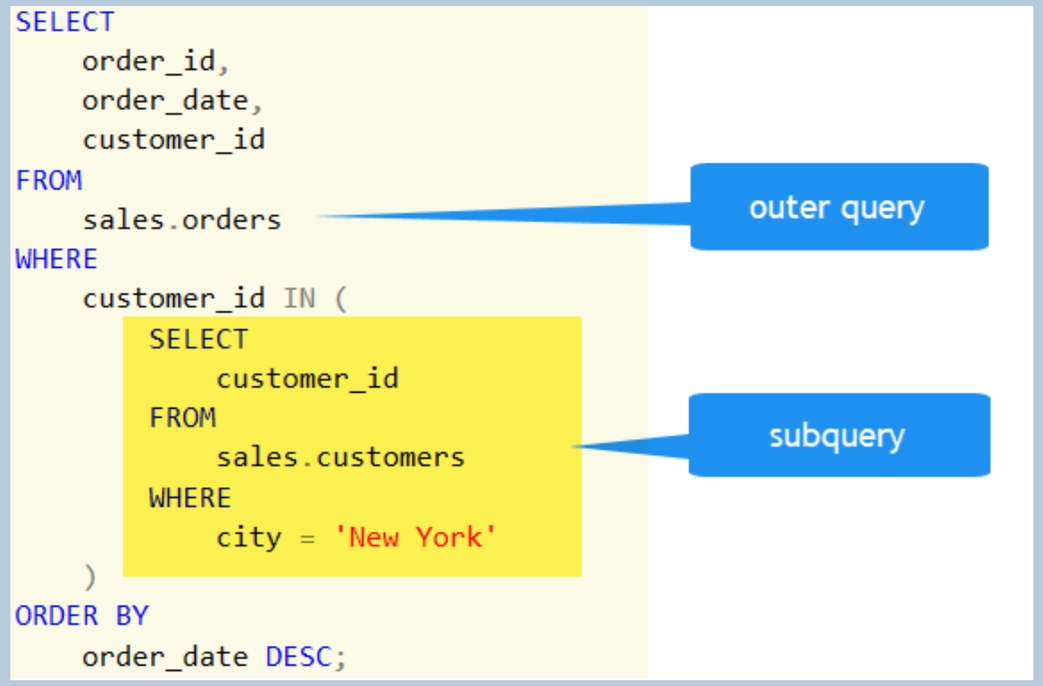
INNER JOIN



Only the things that match on the left AND the right

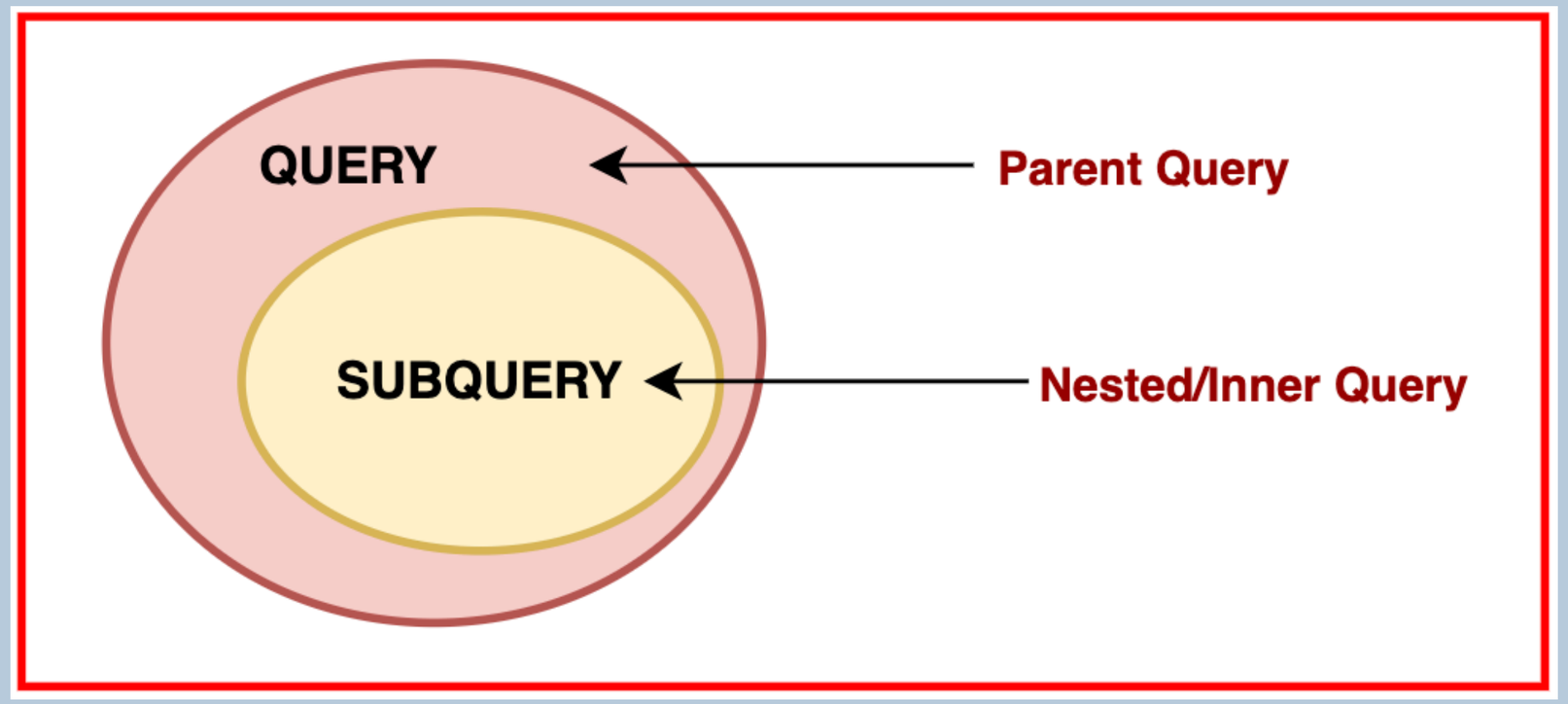
```
SELECT *
FROM TABLE_1
INNER JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
```

Subquery



Subquery adalah query yang ditumpuk pada query lain pada SQL. Dengan kata lain, kamu bisa menulis query baru di dalam query utama, dan dengan perintah dan klausa yang sama, seperti SELECT dan WHERE.
Subquery akan dieksekusi terlebih dahulu sebelum query utama, kemudian hasil keluaran dari subquery ini diteruskan ke query utama.

Subquery



Subquery di dalam WHERE/HAVING

Mencari pelanggan dalam segment 'Home Office' yang memiliki penjualan total di atas rata-rata pelanggan Home Office lainnya.

```sql
select
    sum(sales)/count(distinct customer_id)
from
    superstore
where
    segment = 'Home Office'
```

Untuk menghitung rata-rata sales pada setiap pelanggan di segment Home Office

```sql
select
    customer_name,
    sum(sales)
from
    superstore
where
    segment = 'Home Office'
group by
    customer_name
having
    sum(sales) >=
    (select
    sum(sales)/count(distinct customer_id)
from
    superstore
where
    segment = 'Home Office')
```

Subquery setelah digabungkan dengan outer query

Subquery di dalam FROM

Membuat tabel yang mencakup nama konsumen, segment, dan total penjualan, lalu menampilkan hanya konsumen yang termasuk dalam segment 'Home Office'.

```sql
select
    customer_name,
    segment,
    sum(sales)
from
    superstore
group by
    customer_name, segment
```

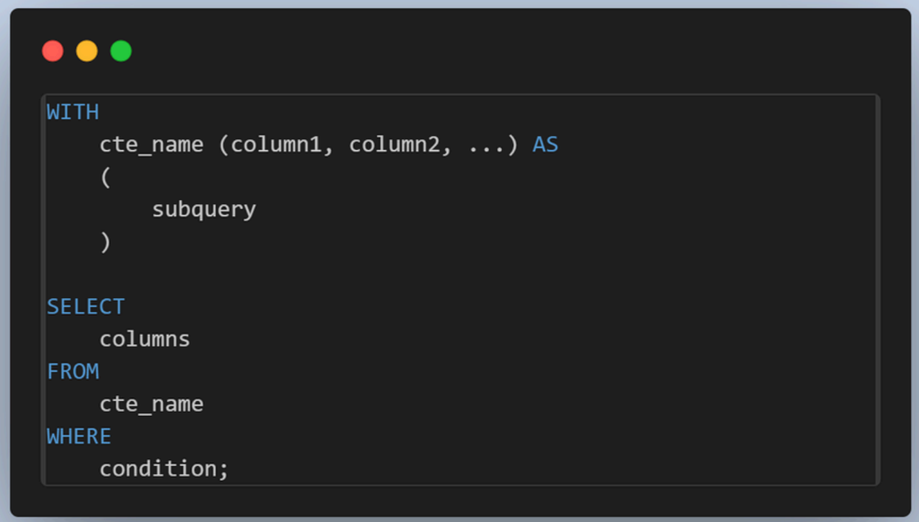
Membuat tabel sementara yang berisi nama konsumen, segment, dan total penjualan

```sql
select
    customer_name,
    value
from
    (select
    customer_name,
    segment,
    sum(sales) as value
from
    superstore
group by
    customer_name, segment) as new_table
where
    segment = 'Home Office'
```

Subquery setelah digabungkan dengan outer query

Subquery menggunakan WITH

- **Syntax**



```
WITH
    cte_name (column1, column2, ...) AS
    (
        subquery
    )

SELECT
    columns
FROM
    cte_name
WHERE
    condition;
```

SQL WITH, yang juga dikenal sebagai Common Table Expressions (CTE), adalah alat bantu yang bisa menyederhanakan query SQL yang kompleks, meningkatkan keterbacaan, dan meningkatkan kinerja dengan mendefinisikan kumpulan hasil sementara yang dapat digunakan kembali beberapa kali.

Subquery menggunakan WITH

```sql
with
    total_penjualan as
    (select
    customer_name,
    segment,
    sum(sales) as value
from
    superstore
group by
    customer_name, segment),

    HomeOffice_avg as
    (select
    sum(sales)/count(distinct customer_id) as "rata-rata"
from
    superstore
where
    segment = 'Home Office')
```

Terdapat dua subquery bernama total_penjualan untuk membuat tabel sementara, dan HomeOffice_avg untuk menghitung rata-rata penjualan konsumen berdasarkan segment Home Office.

```sql
select
    *
from
    total_penjualan
where
    segment = 'Home Office' and value >= (select "rata-rata"
from HomeOffice_avg)
```

Ini adalah query utama yang memanggil cte name di subquery.

Feedback Form