MoF-DAC
Ministry of Finance
Data Analytics Community

MoF-DAC Sharing Session#14:

Introduction to SQL

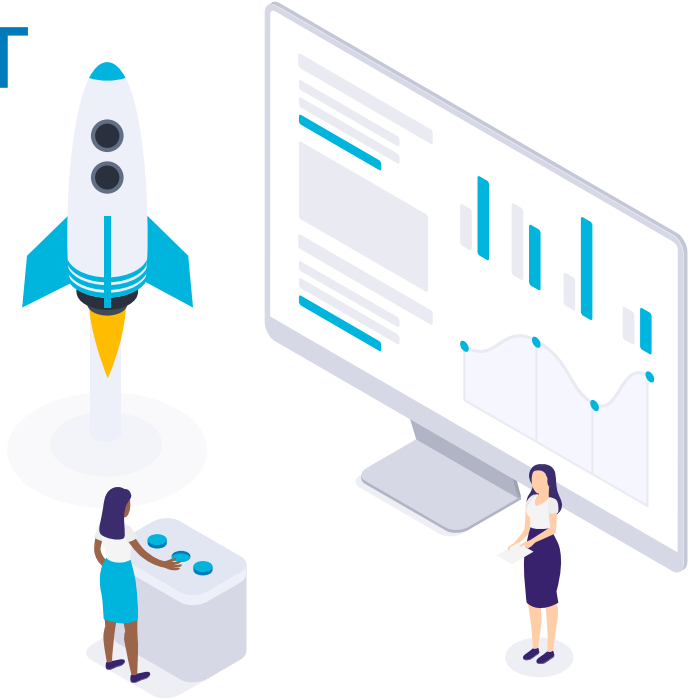
(Structured Query Language)

Reza R Pratama, CISA, CEH
@rezarzky

Why Data Analyst MUST Learn SQL?

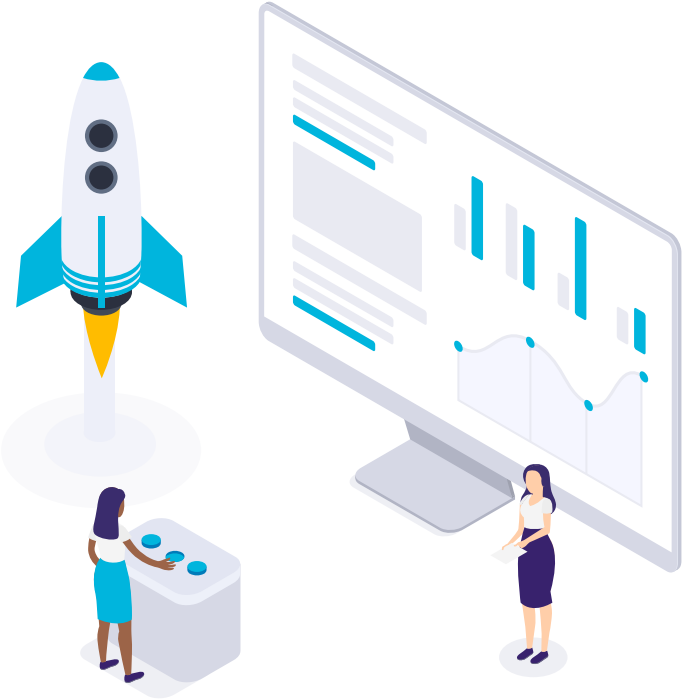
Not just Python/R, Tableau, Power BI, etc

SQL Database is Everywhere
SQL still the top language for data work in 2021*



*StackOverflow 2020 Developer Survey

What is Relational Database & SQL?

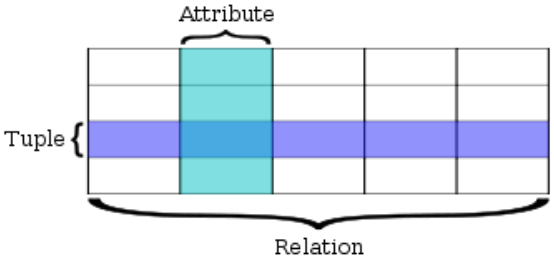


Relational Database

A **relational database** is a digital database based on the relational model of data, as proposed by E. F. Codd in 1970.[1] A software system used to maintain relational databases is a relational database management system (RDBMS). Many relational database systems have an option of using the SQL (Structured Query Language) for querying and maintaining the database.[2]

Terminology

SQL term	Relational database term	Description
Row	***Tuple*** or ***record***	A data set representing a single item
Column	***Attribute*** or ***field***	A labeled element of a tuple, e.g. "Address" or "Date of birth"
Table	***Relation*** or ***Base relvar***	A set of tuples sharing the same attributes; a set of columns and rows
View or ***result set***	***Derived relvar***	Any set of tuples; a data report from the RDBMS in response to a query

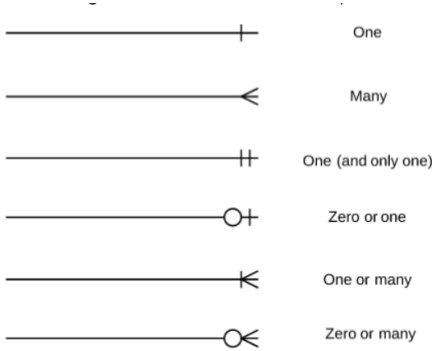


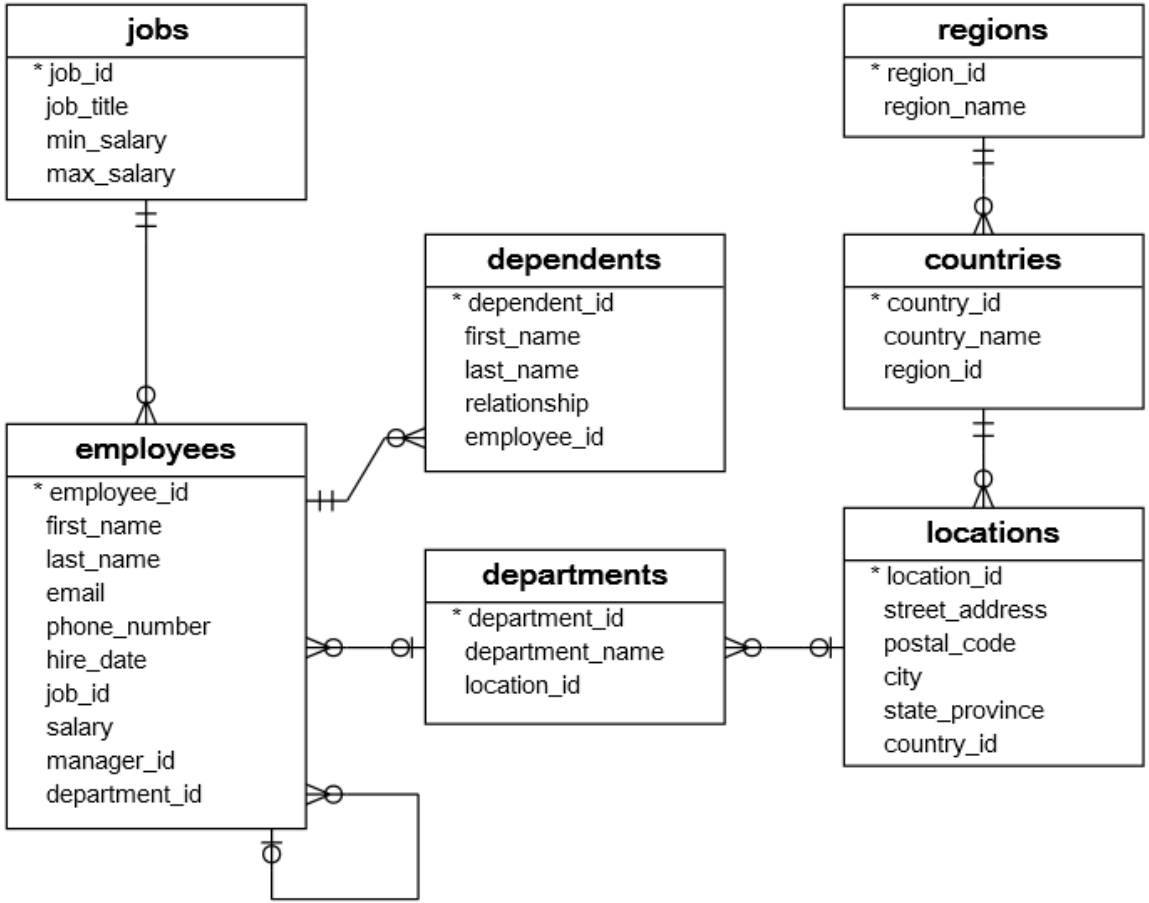
https://en.wikipedia.org/wiki/Relational_database

Basis Data Relasional

Entity Relationship Diagram (ERD)

Primary Key – unique and mandatory
Foreign Key – a cross- reference between tables because it references the primary key of another table



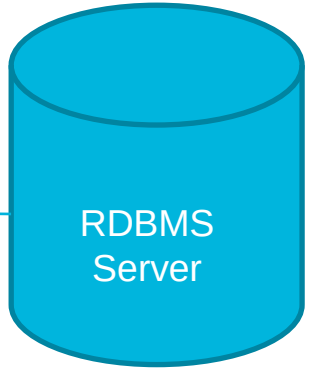


Structured Query Language (SQL) – *S-Q-L /sequel*

- Merupakan bahasa standar ANSI untuk mengoperasikan basis data relasional
- Efisien, mudah dipelajari, dan digunakan
- *Functionally complete* (dengan SQL, kita bisa mendefinisikan, mengambil, dan memanipulasi data dalam table)

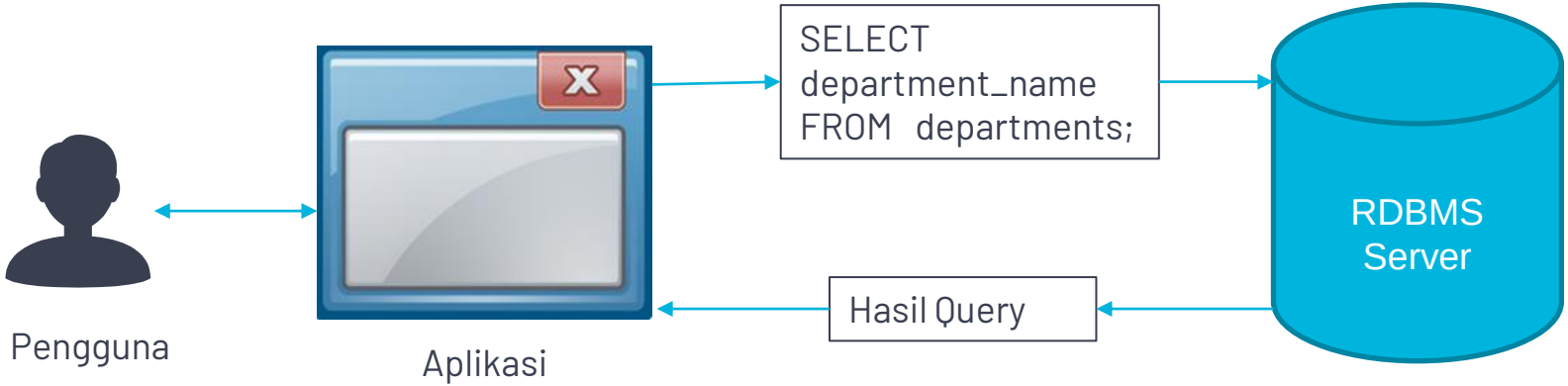
```
SELECT department_name FROM
departments;
```

DEPARTMENT_NAME
Administration
Marketing
Shipping
IT
Sales
Executive
Accounting
Contracting



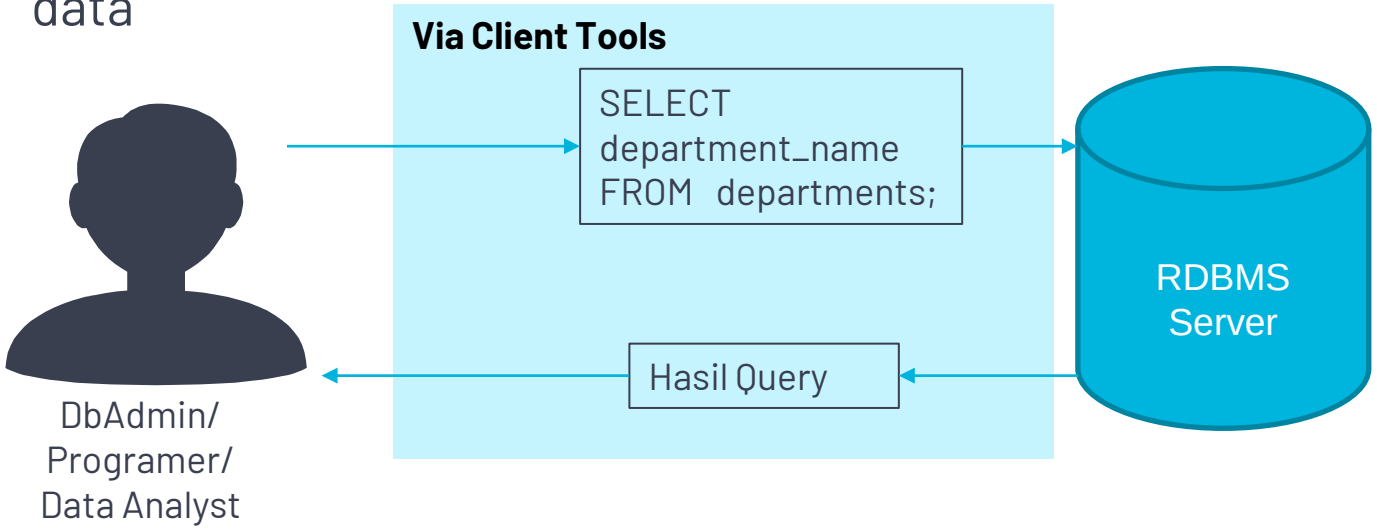
Bagaimana SQL bekerja dalam aplikasi/program

- Aplikasi atau program biasanya memungkinkan pengguna mengakses basis data tanpa menggunakan SQL secara langsung



RDBMS Client dan SQL Statement

- Adminstrator basis data/pengembang aplikasi menggunakan *client tools* untuk menjalankan/mengembangkan SQL statement ke basis data



RDBMS Client dan SQL Statement

- Perlu dipastikan bahwa client tools yang digunakan kompatibel dengan RDMBS yang akan diakses

RDBMS	ORACLE DATABASE	Microsoft SQL Server	MySQL	PostgreSQL
Client	SQL Plus SQL developer	SQL Server Management Studio	MySQL Workbench	PgAdmin

Tipe Data

https://www.w3schools.com/sql/sql_datatypes.asp

- String
  - char(n)
  - varchar(n)
  - nvarchar(max)
  - blob
- Numeric
  - boolean
  - Int, bigint
  - bit
  - float(n)
- Date and Time Data Types
  - Datetime
  - Timestamp
  - Date
  - time
- Other Data Types
  - sql_variant
  - xml
  - uniqueidentifier

SQL Statements

Data Definition Language (DDL)

- **Creating database structure.**
  - CREATE TABLE, ALTER TABLE, DROP TABLE

Data Manipulation Language (DML)

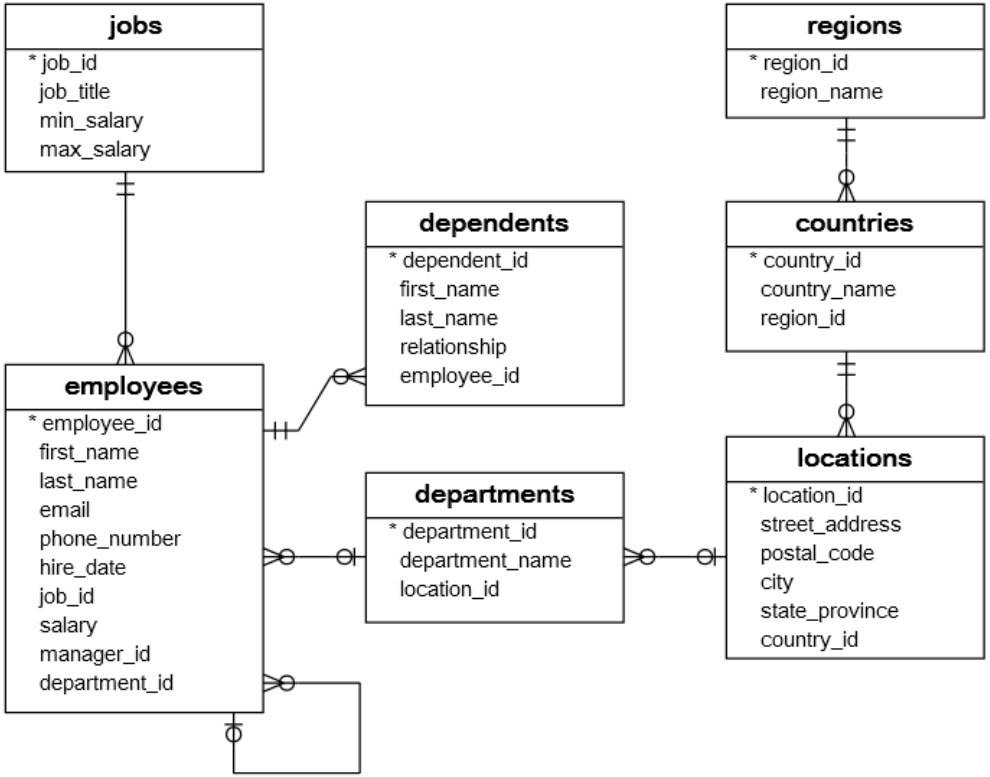
- **Adding and Manipulating database contents (rows).**
  - INSERT, UPDATE, DELETE
- **Retrieving data from database**
  - SELECT

Data Control Language (DCL)

  - GRANT, REVOKE

Writing SQL Statements

- SQL statements are not case-sensitive.
- SQL statements can be entered on one or more lines.
- Keywords cannot be abbreviated or split across lines.
- Clauses are usually placed on separate lines.
- Indents are used to enhance readability.
- SQL statements can optionally be terminated by a semicolon (;). Semicolons are required when you execute multiple SQL statements.



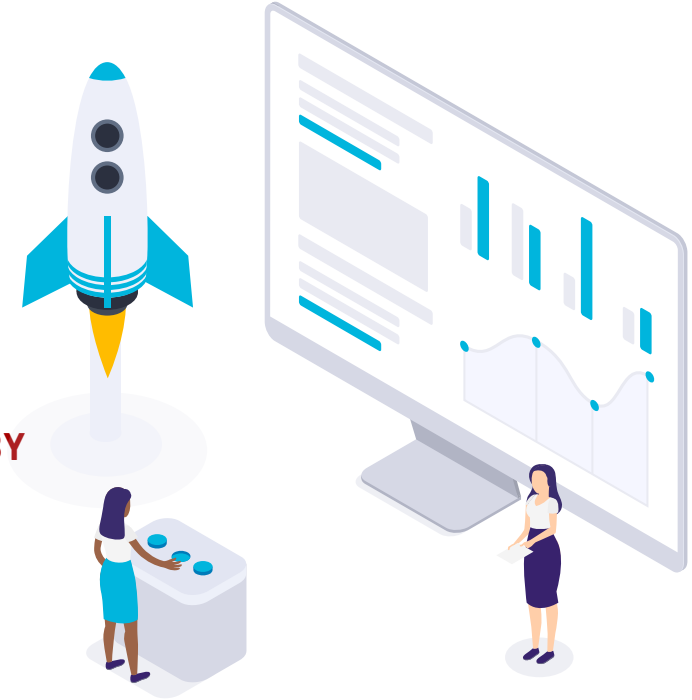
Untuk Latihan Tanpa Install Database

https://www.sqltutorial.org/seeit/

SQL Statement

BASIC QUERIES

SELECT – FROM – WHERE – GROUP BY – HAVING – ORDER BY



Basic Queries - SELECT

```
-- filter your columns
SELECT col1, col2, col3, ... FROM table1
-- filter the rows
WHERE col4 = 1 AND col5 = 2
-- aggregate the data
GROUP by ...
-- limit aggregated data
HAVING count(*) > 1
-- order of the results
ORDER BY col2
```

Employee

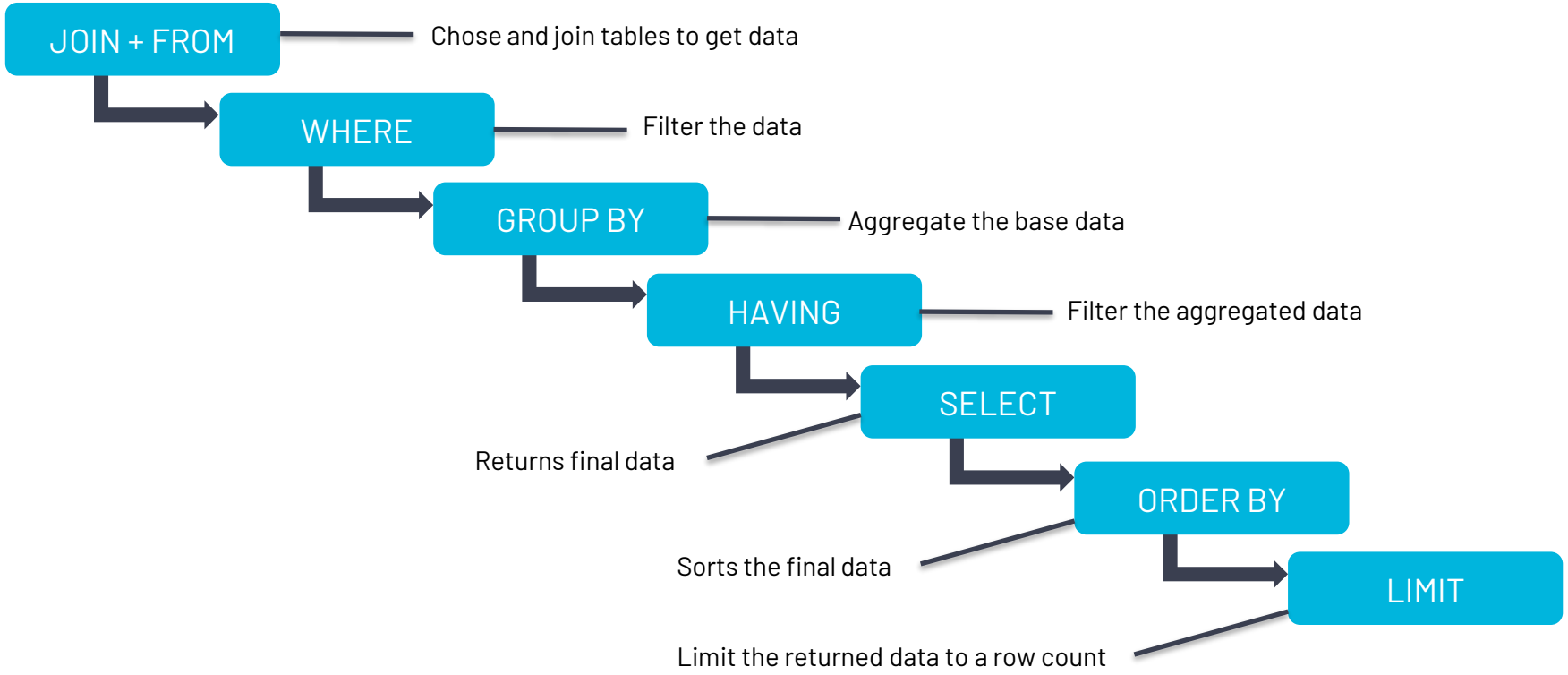
EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1002	Anna	1	3500
1003	James	1	2500
1004	David	2	5000
1005	Mark	2	3000
1006	Steve	3	4500
1007	Alice	3	3500

SELECT *DeptID, AVG(Salary)*
FROM *Employee*
GROUP BY *DeptID;*

GROUP BY Employee Table using DeptID

DeptID	AVG(Salary)
1	3000.00
2	4000.00
3	4250.00

SQL Query Order of Execution



SELECT:Anatomi SELECT Statement – Fungsi/Kegunaan

Kolom/field yang akan ditampilkan

Tabel yang digunakan

```
SELECT EMPLOYEE_ID, FIRST_NAME, HIRE_DATE, JOB_ID, SALARY
FROM EMPLOYEES
WHERE SALARY > 10000;
```

Row yang akan diambil adalah row dengan gaji di atas 10.000

Results | Messages

	EMPLOYEE_ID	FIRST_NAME	HIRE_DATE	JOB_ID	SALARY
1	100	Steven	2003-06-17 00:00:00.0000000	AD_PRES	24000
2	101	Neena	2005-09-21 00:00:00.0000000	AD_VP	17000
3	102	Lex	2001-01-13 00:00:00.0000000	AD_VP	17000
4	108	Nancy	2002-08-17 00:00:00.0000000	FI_MGR	12008
5	114	Den	2002-12-07 00:00:00.0000000	PU_MAN	11000
6	145	John	2004-10-01 00:00:00.0000000	SA_MAN	14000
7	146	Karen	2005-01-05 00:00:00.0000000	SA_MAN	13500
8	147	Alberto	2005-03-10 00:00:00.0000000	SA_MAN	12000
9	148	Gerald	2007-10-15 00:00:00.0000000	SA_MAN	11000
10	149	Eleni	2008-01-29 00:00:00.0000000	SA_MAN	10500
11	162	Clara	2005-11-11 00:00:00.0000000	SA_REP	10500
12	168	Lisa	2005-03-11 00:00:00.0000000	SA_REP	11500
13	174	Ellen	2004-05-11 00:00:00.0000000	SA_REP	11000
14	201	Michael	2004-02-17 00:00:00.0000000	MK_MAN	13000
15	205	Shelley	2002-06-07 00:00:00.0000000	AC_MGR	12008

SELECT

Limiting the Column that Are Selected

Menampilkan Semua Kolom:

- SELECT *****

Menampilkan Kolom yang Spesifik

- SELECT nama_kolom, nama_kolom2

Menampilkan Hanya Record yang Unique

- SELECT **DISTINCT** nama_kolom

Menggunakan Alias untuk Nama Kolom

- SELECT nama_kolom **as** alias_kolom
- SELECT nama_kolom alias_kolom

SELECT
Menggunakan Ekspresi Aritmatik

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

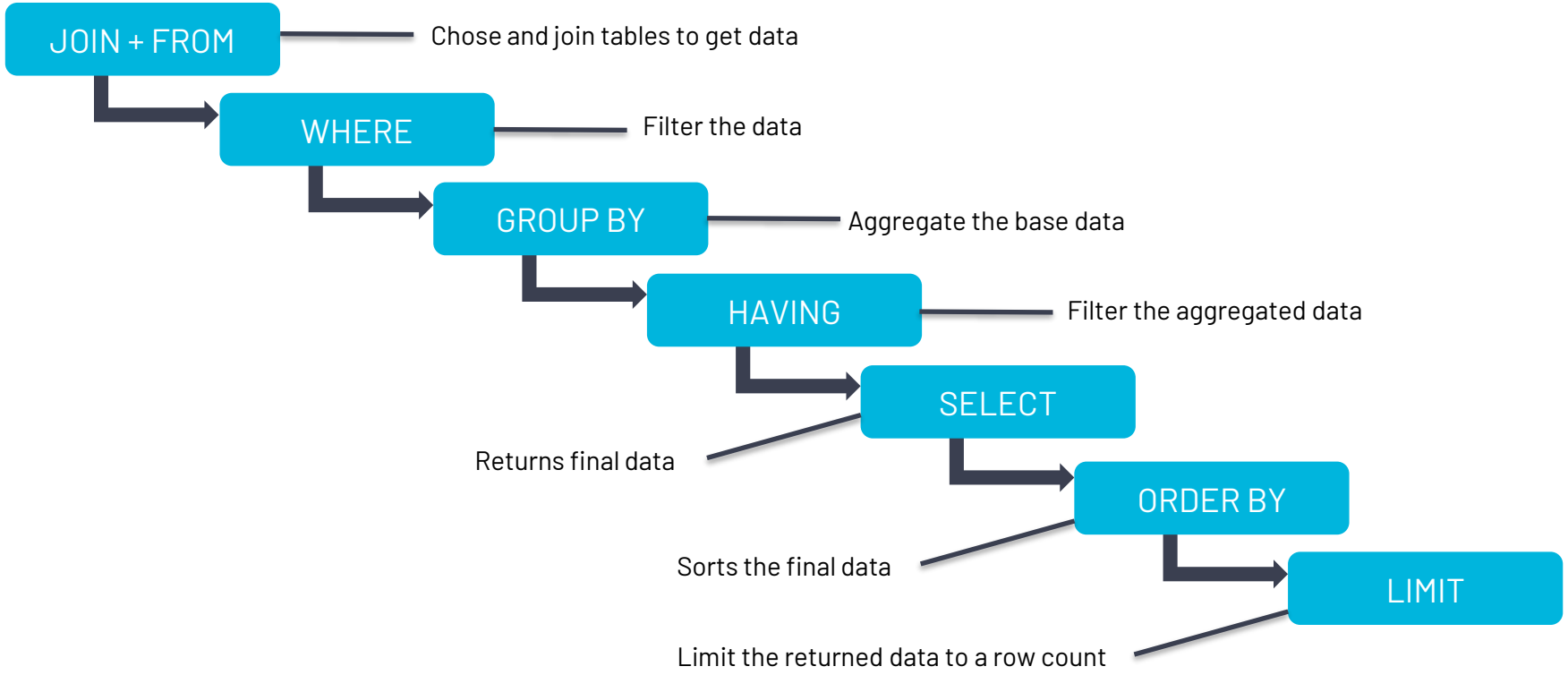
```
SELECT last_name, salary, 12*(salary+100)
FROM   employees;
```

	LAST_NAME	SALARY	12*(SALARY+100)
1	King	24000	289200
2	Kochhar	17000	205200
3	De Haan	17000	205200

Basic Queries - Restricting and Sorting Data

```
-- filter your columns
SELECT col1, col2, col3, ... FROM table1
-- filter the rows
WHERE col4 = 1 AND col5 = 2
-- aggregate the data
GROUP by ...
-- limit aggregated data
HAVING count(*) > 1
-- order of the results
ORDER BY col2
```

SQL Query Order of Execution



WHERE

Limiting the Rows that Are Selected

```
SELECT *|{[DISTINCT] column|expression [alias],...}
FROM   table
[WHERE condition(s)];
```

Comparison Operators

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
BETWEEN ...AND...	Between two values (inclusive)
IN(set)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

Example

```
SELECT employee_id, last_name, job_id, department_id
FROM   employees
WHERE  department_id = 90 ;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

WHERE

Limiting the Rows that Are Selected

Pattern Matching Using the LIKE Operator

- Use the LIKE operator to perform wildcard searches of valid search string values.
- Search conditions can contain either literal characters or numbers:
  - % denotes zero or many characters.
  - _ denotes one character.

```
SELECT        first_name
FROM          employees
WHERE         first_name LIKE 'S%' ;
```

Using the NULL Conditions

- The NULL conditions include the IS NULL condition and the IS NOT NULL condition.

```
SELECT last_name, manager_id
FROM   employees
WHERE  manager_id IS NULL ;
```

WHERE with more than one conditions: AND, OR, NOT

Operator	Meaning
AND	Returns TRUE if *both* component conditions are true
OR	Returns TRUE if *either* component condition is true
NOT	Returns TRUE if the condition is false

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
AND    job_id LIKE '%MAN%' ;
```

Quest!

- Siapakah pemilik Gaji Tertinggi di Perusahaan

a. Karen Colmenares
b. Steven King
c. Lex De Haan
d. Guy Himuro

ORDER BY

Sorting Data

Mengurutkan data menggunakan ORDER BY:

- **ASC:** Ascending order, default
- **DESC:** Descending order

Contoh:

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY hire_date ASC;
```

Mengurutkan berdasarkan kolom hire_date dengan urutan ascending

```
SELECT employee_id, last_name, salary*12 annsal
FROM   employees
ORDER BY annsal ;
```

Mengurutkan berdasarkan nama alias annsal

```
SELECT last_name, department_id, salary
FROM   employees
ORDER BY 2, salary DESC;
```

Mengurutkan menggunakan beberapa kolom, salah satunya menggunakan posisi dari kolom di SELECT

Quest 2!

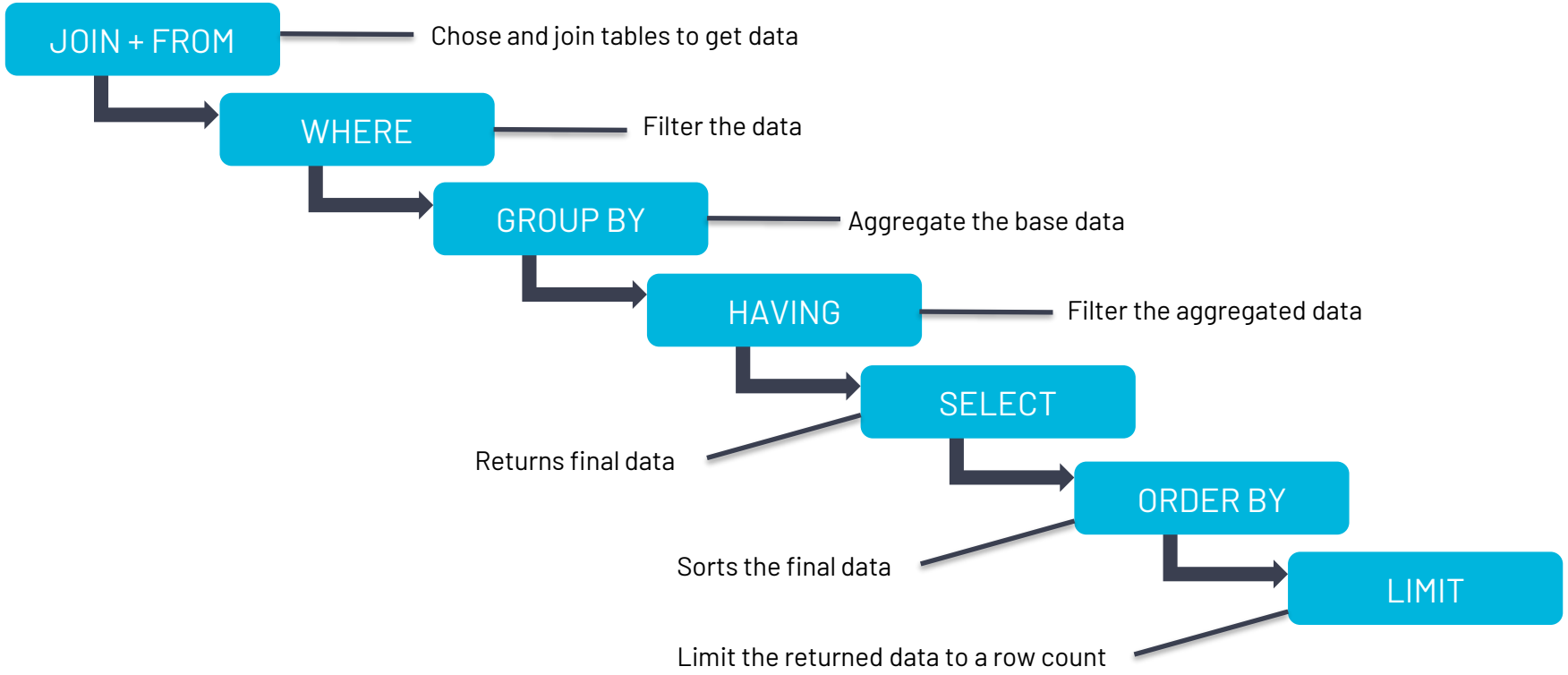
- Siapakah pegawai yang tidak mempunyai manager?

a. Karen Colmenares
b. Steven King
c. Lex De Haan
d. Guy Himuro

Basic Queries - Aggregating & Limit Aggregated Data

```
-- filter your columns
   SELECT col1, col2, col3, ... FROM table1
-- filter the rows
   WHERE col4 = 1 AND col5 = 2
-- aggregate the data
   GROUP by ...
-- limit aggregated data
   HAVING count(*) > 1
-- order of the results
   ORDER BY col2
```

SQL Query Order of Execution



Creating Groups of Data

EMPLOYEES

	DEPARTMENT_ID	SALARY	
1	10	4400	4400
2	20	13000	9500
3	20	6000	
4	50	5800	3500
5	50	2500	
6	50	2600	
7	50	3100	
8	50	3500	
9	60	4200	6400
10	60	6000	
11	60	9000	
12	80	11000	10033
13	80	10500	
14	80	8600	
...			
19	110	12000	
20	(null)	7000	

Average salary in EMPLOYEES table for each department

	DEPARTMENT_ID	AVG(SALARY)
1	10	4400
2	20	9500
3	50	3500
4	60	6400
5	80	10033.333333333333...
6	90	19333.333333333333...
7	110	10150
8	(null)	7000

Grouping by More than One Column

EMPLOYEES

	DEPARTMENT_ID	JOB_ID	SALARY
1	10	AD_ASST	4400
2	20	MK_MAN	13000
3	20	MK_REP	6000
4	50	ST_MAN	5800
5	50	ST_CLERK	2500
6	50	ST_CLERK	2600
7	50	ST_CLERK	3100
8	50	ST_CLERK	3500
9	60	IT_PROG	4200
10	60	IT_PROG	6000
11	60	IT_PROG	9000
12	80	SA_REP	11000
13	80	SA_MAN	10500
14	80	SA_REP	8600
...			
19	110	AC_MGR	12000
20	(null)	SA_REP	7000

Add the salaries in the EMPLOYEES table for each job, grouped by department.

	DEPARTMENT_ID	JOB_ID	SUM(SALARY)
1	10	AD_ASST	4400
2	20	MK_MAN	13000
3	20	MK_REP	6000
4	50	ST_CLERK	11700
5	50	ST_MAN	5800
6	60	IT_PROG	19200
7	80	SA_MAN	10500
8	80	SA_REP	19600
9	90	AD_PRES	24000
10	90	AD_VP	34000
11	110	AC_ACCOUNT	8300
12	110	AC_MGR	12000
13	(null)	SA_REP	7000

GROUP BY

Aggregating Data

```
SELECT     column, group_function(column)
FROM       table
[WHERE     condition]
[GROUP BY  group_by_expression]
[ORDER BY  column];
```

Group Function

- **AVG**
- **COUNT**
- **MAX**
- **MIN**
- **SUM**
- STDDEV
- VARIANCE

Example

```
SELECT    department_id dept_id, job_id, SUM(salary),
             COUNT(department_id)
FROM      employees
WHERE salary > 1000
GROUP BY department id, job id
ORDER BY department_id;
```

HAVING

Filtetring Aggregating Data

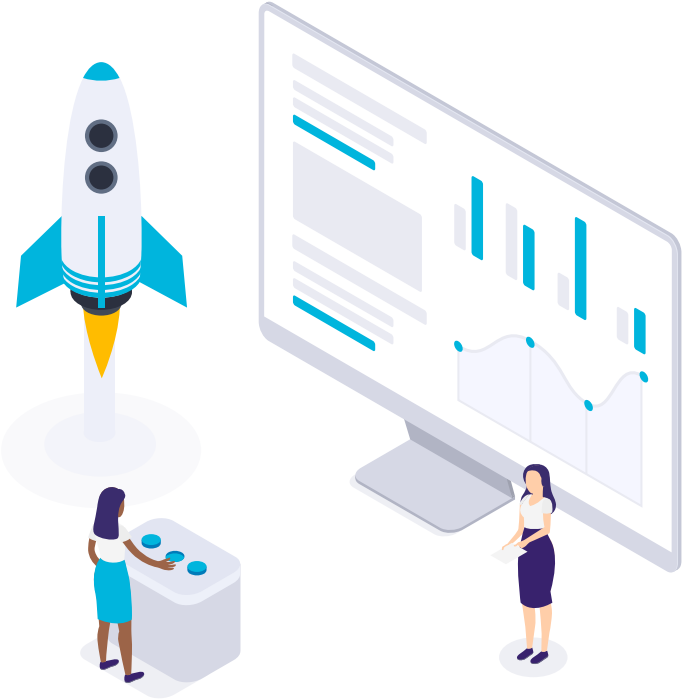
```
SELECT      column, group_function
FROM        table
[WHERE      condition]
[GROUP BY   group by expression]
[HAVING     group condition]
[ORDER BY   column];
```

- Filtering data yang telah di grouping
- Mirip dengan WHERE

SQL Statement

Queries on Multiple Table

JOIN – SET OPERATIONS - SUBQUERY



A. JOIN
Obtaining Data from Multiple Tables

EMPLOYEES

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	100	King	90
2	101	Kochhar	90
3	102	De Haan	90
...			
18	202	Fay	20
19	205	Higgins	110
20	206	Gietz	110

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700

	EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	200	10	Administration
2	201	20	Marketing
3	202	20	Marketing
4	124	50	Shipping
5	144	50	Shipping
...			
18	205	110	Accounting
19	206	110	Accounting

Joining Column Names

EMPLOYEES

EMPLOYEE_ID	DEPARTMENT_ID
100	90
101	90
102	90
103	60
104	60
107	60
124	50
141	50
142	50
143	50
144	50
149	80
174	80
176	80

...

Foreign key

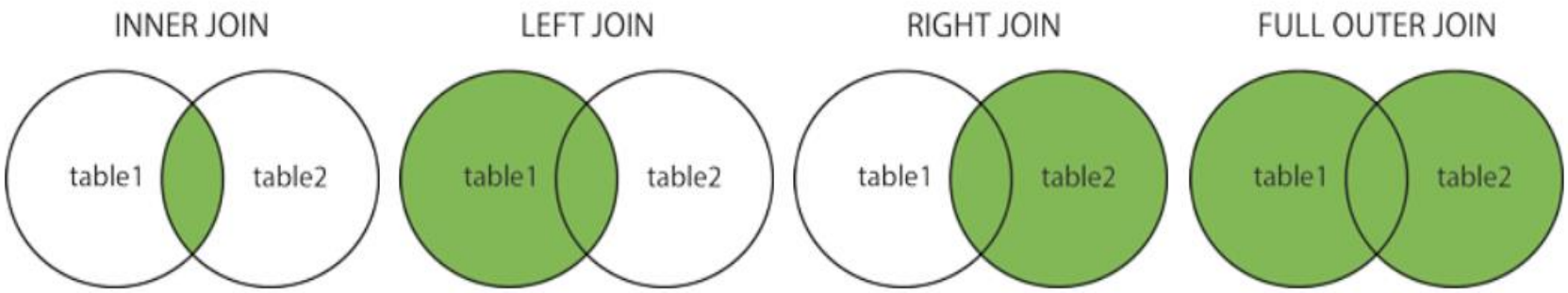
DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME
1	10	Administration
2	20	Marketing
3	50	Shipping
4	60	IT
5	80	Sales
6	90	Executive
7	110	Accounting
8	190	Contracting

Primary key

A. JOIN

mengambil data dari lebih dari satu table



Contoh:

```
SELECT * FROM table1
INNER JOIN table2 ON table2.CustomerID=table1.CustomerID;
```

INNER JOIN / JOIN

Student

ID	NAME
1	Alice
2	Bob
3	Chris

Mark

ID	SUBJECT	MARK
1	1004	95
2	1045	55
1	1045	90
4	1004	100

Natural Join gives no information for Chris and the student with ID 4

ID	NAME	ID_1	SUBJECT	MARK
1	Alice	1	1004	95
2	Bob	2	1045	55
1	Alice	1	1045	90

```
SELECT * FROM student s INNER JOIN mark m ON s.ID = m.ID
```

LEFT JOIN

Student

ID	NAME
1	Alice
2	Bob
3	Chris

Mark

ID	SUBJECT	MARK
1	1004	95
2	1045	55
1	1045	90
4	1004	100

```
SELECT *
FROM student s
LEFT JOIN mark m
ON s.ID = m.ID
```

Get (incomplete) information of only Chris

ID	NAME	ID_1	SUBJECT	MARK
1	Alice	1	1004	95
2	Bob	2	1045	55
1	Alice	1	1045	90
3	Chris	(null)	(null)	(null)

Coba:

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e LEFT JOIN departments d
ON   (e.department_id = d.department_id) ;
```

RIGHT JOIN

Student

ID	NAME
1	Alice
2	Bob
3	Chris

Mark

ID	SUBJECT	MARK
1	1004	95
2	1045	55
1	1045	90
4	1004	100

Get (incomplete) information of the student with ID 4

```
SELECT *
FROM student s
RIGHT OUTER JOIN mark m
ON s.ID = m.ID
```

ID	NAME	ID_1	SUBJECT	MARK
1	Alice	1	1045	90
1	Alice	1	1004	95
2	Bob	2	1045	55
(null)	(null)	4	1004	100

FULL OUTER JOIN

Student

ID	NAME
1	Alice
2	Bob
3	Chris

Mark

ID	SUBJECT	MARK
1	1004	95
2	1045	55
1	1045	90
4	1004	100

```
SELECT *
FROM student s
FULL OUTER JOIN mark m
ON s.ID = m.ID
```

Get (incomplete) information of both Chris and student with ID 4

ID	NAME	ID_1	SUBJECT	MARK
1	Alice	1	1004	95
2	Bob	2	1045	55
1	Alice	1	1045	90
(null)	(null)	4	1004	100
3	Chris	(null)	(null)	(null)

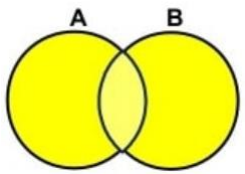
Join dengan lebih dari 2 table

```
SELECT employee_id, city, department_name
FROM   employees e
JOIN   departments d
ON     d.department_id = e.department_id
JOIN   locations l
ON     d.location_id = l.location_id;
```

	EMPLOYEE_ID	CITY	DEPARTMENT_NAME
1	100	Seattle	Executive
2	101	Seattle	Executive
3	102	Seattle	Executive
4	103	Southlake	IT
5	104	Southlake	IT
6	107	Southlake	IT
7	124	South San Francisco	Shipping
8	141	South San Francisco	Shipping

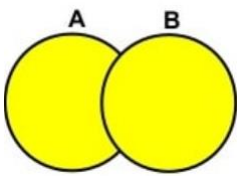
...

B. Relational Set Operations



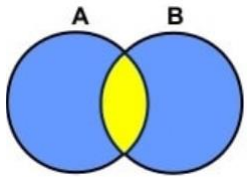
UNION ALL

- Semua row digabungkan. Walau ada row duplikat



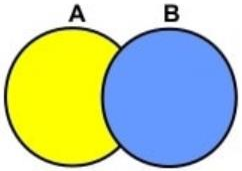
UNION

- Semua row digabungkan dan row yang duplikat di hilangkan (disisakan satu). Distinct dari UNION ALL



INTERSECT

- Ditampilkan row duplikat (ada di kedua sisi), namun hanya distinct row yang disisakan



EXCEPT

- Ditampilkan semua row dari sisi kiri, namus row-row yang juga ada pada sisi kanan dihilangkan

UNION COMPATIBLE

- Same number of attributes
- Similar datatypes

Contoh SET OPERATIONS:

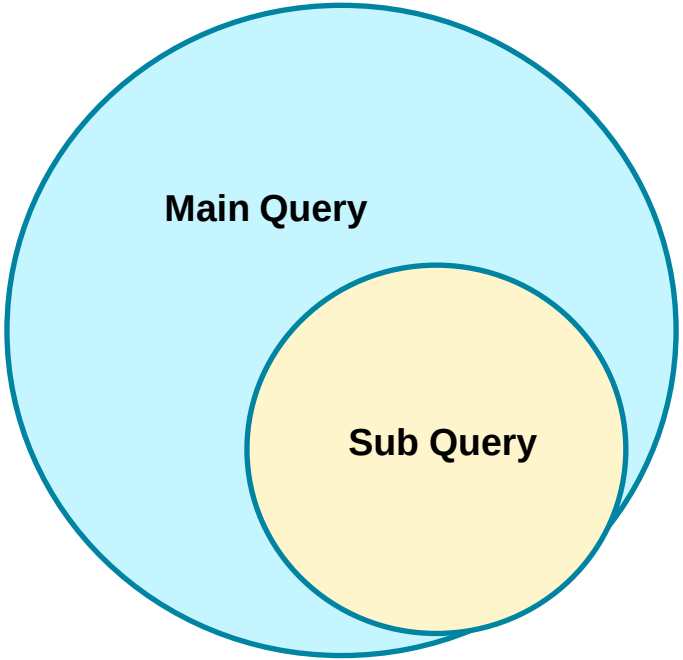
Using the UNION ALL Operator

```
SELECT employee_id, job_id, department_id
FROM   employees
UNION ALL
SELECT employee_id, job_id, department_id
FROM   job_history
ORDER BY  employee_id;
```

	EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	100	AD_PRES	90
...			
16	144	ST_CLERK	50
17	149	SA_MAN	80
18	174	SA_REP	80
19	176	SA_REP	80
20	176	SA_MAN	80
21	176	SA_REP	80
22	178	SA_REP	(null)
...			
30	206	AC_ACCOUNT	110

C. SUBBQUERY

Menjalankan Query di Dalam Query

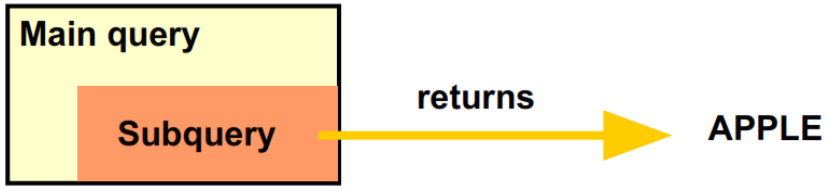



```
SELECT last_name, salary
FROM   employees
WHERE  salary >
               (SELECT salary
                FROM   employees
                WHERE  last_name = 'Abel');
```

	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Hartstein	13000
5	Higgins	12000

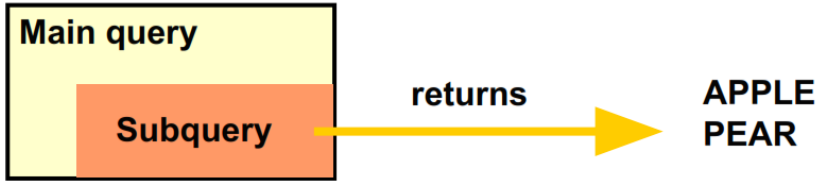
C. SUBBQUERY
Types of Subquery

Single Value



Multiple Value, many rows and columns

Multiple Value, many row but one column



Apa yang dihasilkan dari subquery ini

```
SELECT *
FROM employees
WHERE salary > (SELECT avg(salary)
                FROM employees
                GROUP BY department_id);
```

A. A value (a single column, single row).
B. A list of values.
C. Multiple columns, multiple rows.
D. None of the above.

Apa yang dihasilkan dari subquery ini

```
SELECT *
FROM EMPLOYEES
WHERE SALARY IN (SELECT MAX(SALARY)
                 FROM EMPLOYEES
                 GROUP BY
                   DEPARTMENT_ID)
```

A. A value (a single column, single row).
B. A list of values.
C. Multiple columns, multiple rows.
D. None of the above.

Single-Row Subquery

Return only one row
Use single-row comparison operators

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

Multi-Row Subquery

Return more than one row
Use multiple-row comparison operators

Operator	Meaning
`IN`	Equal to any member in the list
`ANY`	Must be preceded by =, !=, >, <, <=, >=. Compares a value to each value in a list or returned by a query. Evaluates to `FALSE` if the query returns no rows.
`ALL`	Must be preceded by =, !=, >, <, <=, >=. Compares a value to every value in a list or returned by a query. Evaluates to `TRUE` if the query returns no rows.

THANKS!

Reference

- https://www.w3schools.com/sql/
- https://en.wikipedia.org/wiki/Relational_database
- https://www.sqltutorial.org/sql-sample-database/