DQLab

# SELECT STATEMENT USING SQL & R

TANTUT WAHYU SETYOKO
**e-Commerce Data Analyst**

**Jakarta, 14 Oktober 2021**

# # OUTLINE

DATA PRACTITIONER in ROLE

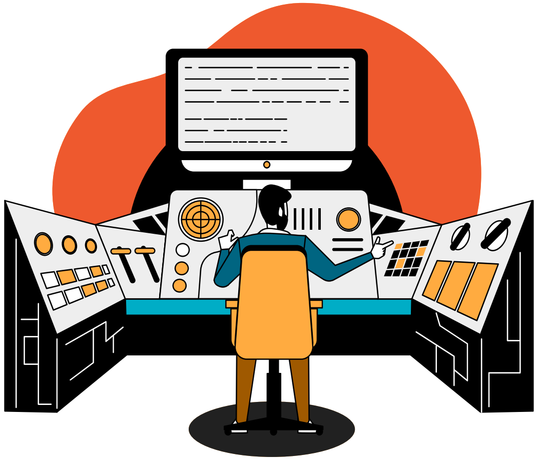
DATA SCIENCE LIFE CYCLE & PRACTITIONER TASK

SQL FUNDAMENTAL

SQL DEMO

SQL IN REAL CASE BUSINESS

# DATA IS THE NEW OIL (Clive Humby, 2006)



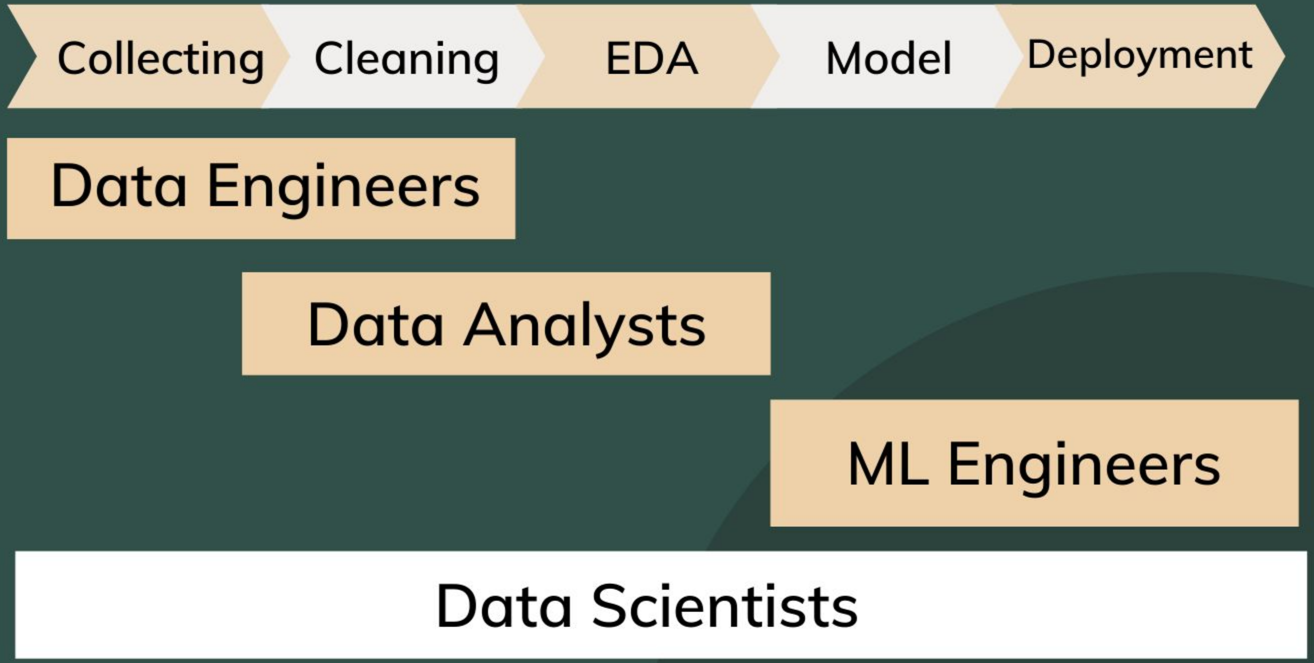
DATA ENGINEER

DATA ANALYST

MACHINE LEARNING ENGINEER

DATA SCIENTIST

# Data Science Life Cycle



Collecting

Cleaning

EDA

Model

Deployment

Data Engineers

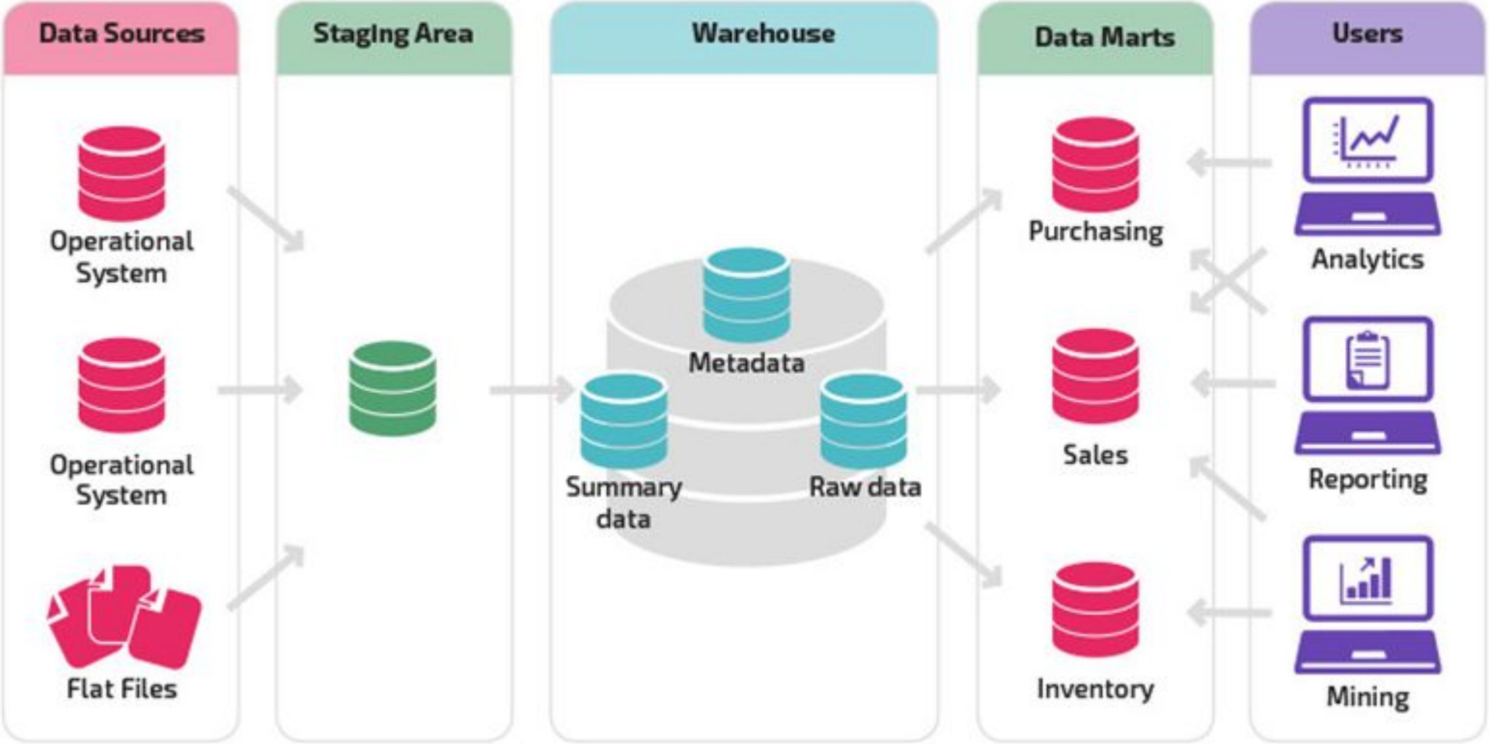
Data Analysts

ML Engineers

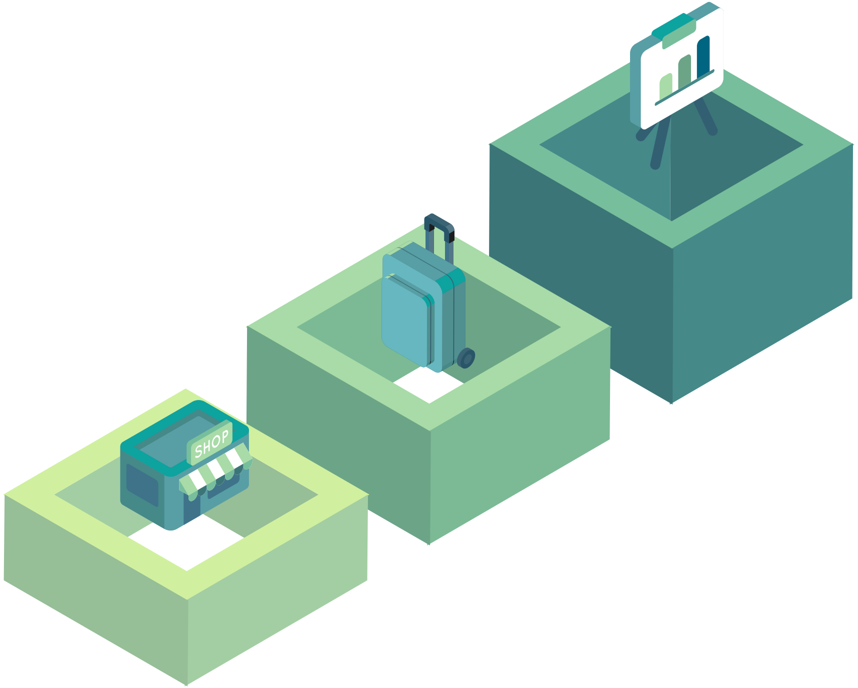
Data Scientists

Chanin Nantasenamat with Ken Jee

# WHAT IS THE
# DATA PRACTITIONER’S TASK ?

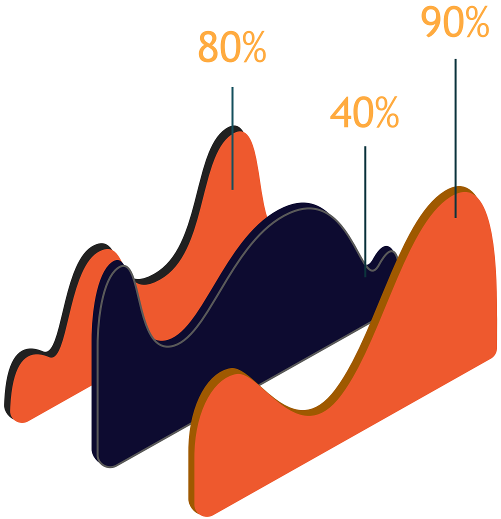


Panoply



# WHAT IS QUERYING ?

# “ WHAT IS STRUCTURED QUERY LANGUAGE ?”



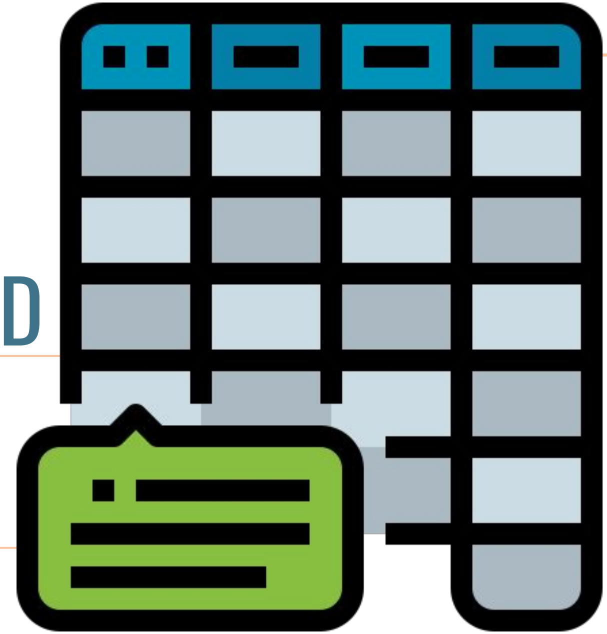
*wildcards*

```
SELECT *|{[DISTINCT] column|expression
[alias],...}

FROM table;
```

**SELECT identifies the columns to be displayed**

**FROM identifies the table containing those columns**



COLUMN NAME

RAW/RECORD

DATA

SQL IS A LANGUAGE
TO CREATE OUR QUERY

# “STRUCTURE SELECT STATEMENT”

```
SELECT

          NAMA_TABLE.NAMA_KOLOM AS NAMA_ALIAS

FROM NAMA_TABEL_SEMESTA
```

request data

```
JOIN NAMA_TABEL ON NAMA_KOLOM.NAMA_TABEL_PENGHUBUNG1=NAMA_KOLOM.NAMA_TABEL_PENGHUBUNG2
```

join condition

```
WHERE (CONDITION)
          AND(CONDITION)
          OR (CONDITION)
```

filter condition

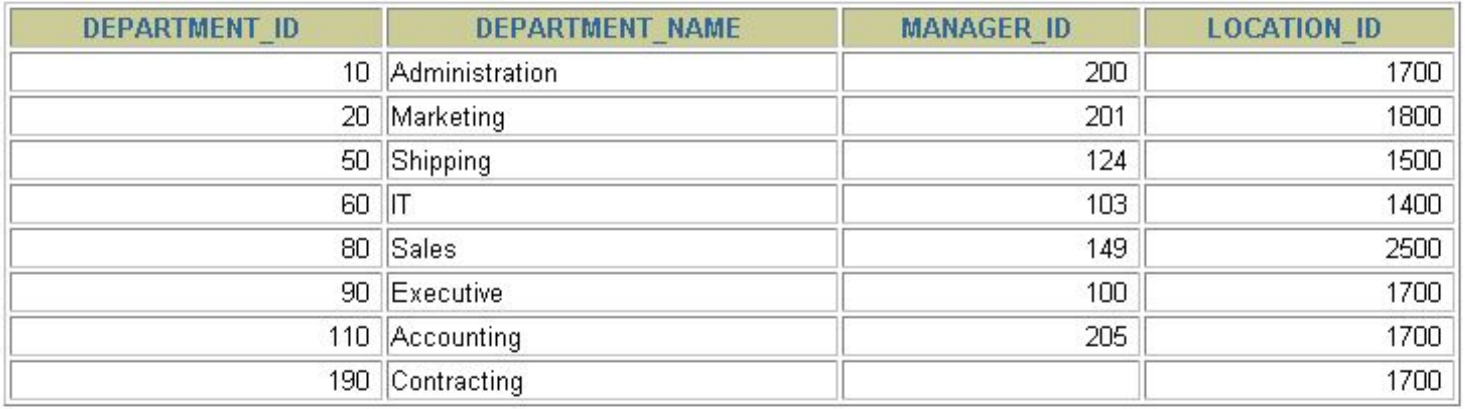
```
ORDER BY ( )
GROUP BY ( )
```

group and order condition

CLAUSES

# “EXAMPLE of DATASET”

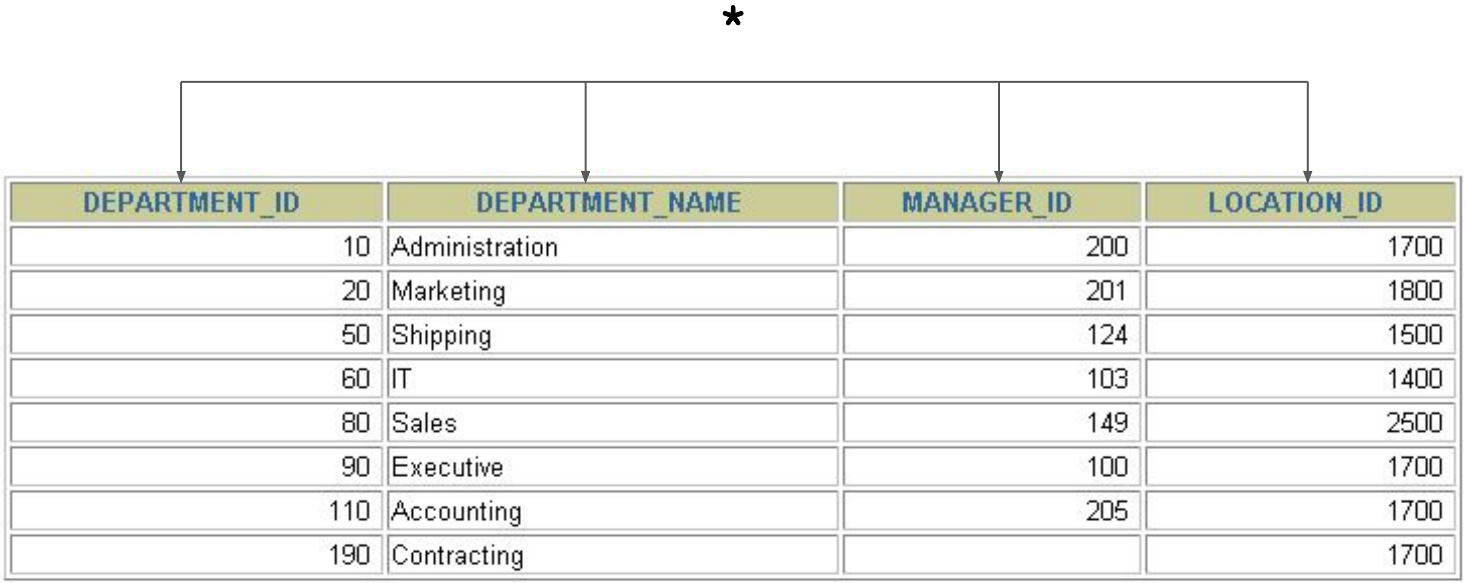
TABLE NAME : DEPARTMENTS
COLUMN NAME : DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID, LOCATION_ID



| DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|---:|---|---:|---:|
| 10 | Administration | 200 | 1700 |
| 20 | Marketing | 201 | 1800 |
| 50 | Shipping | 124 | 1500 |
| 60 | IT | 103 | 1400 |
| 80 | Sales | 149 | 2500 |
| 90 | Executive | 100 | 1700 |
| 110 | Accounting | 205 | 1700 |
| 190 | Contracting |  | 1700 |

# “SELECT STATEMENT”

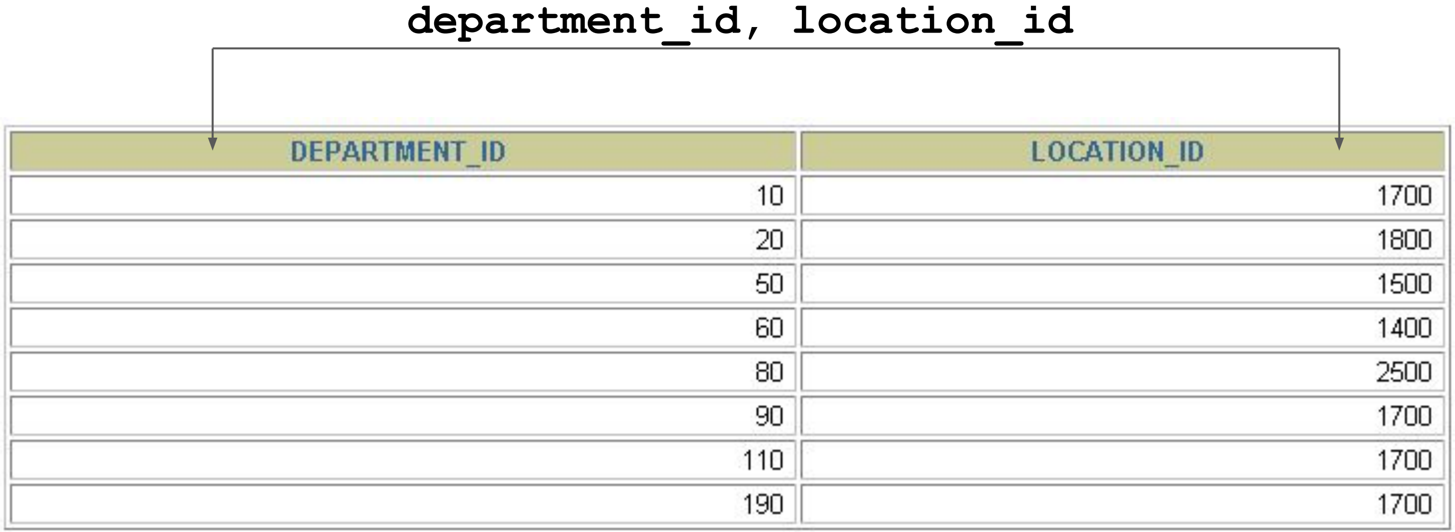
```
SELECT *
FROM   departments;
```



| DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|---|---|---|---|
| 10 | Administration | 200 | 1700 |
| 20 | Marketing | 201 | 1800 |
| 50 | Shipping | 124 | 1500 |
| 60 | IT | 103 | 1400 |
| 80 | Sales | 149 | 2500 |
| 90 | Executive | 100 | 1700 |
| 110 | Accounting | 205 | 1700 |
| 190 | Contracting | | 1700 |

# “SELECT STATEMENT”

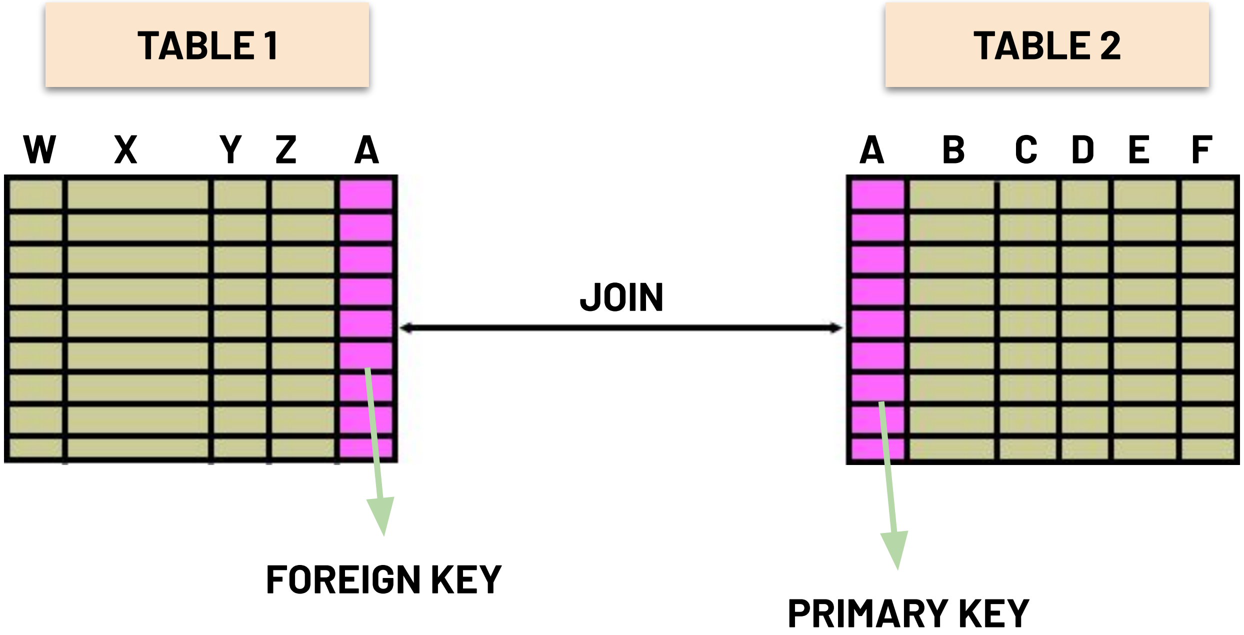
```
SELECT department_id, location_id
FROM   departments;
```



**department_id, location_id**

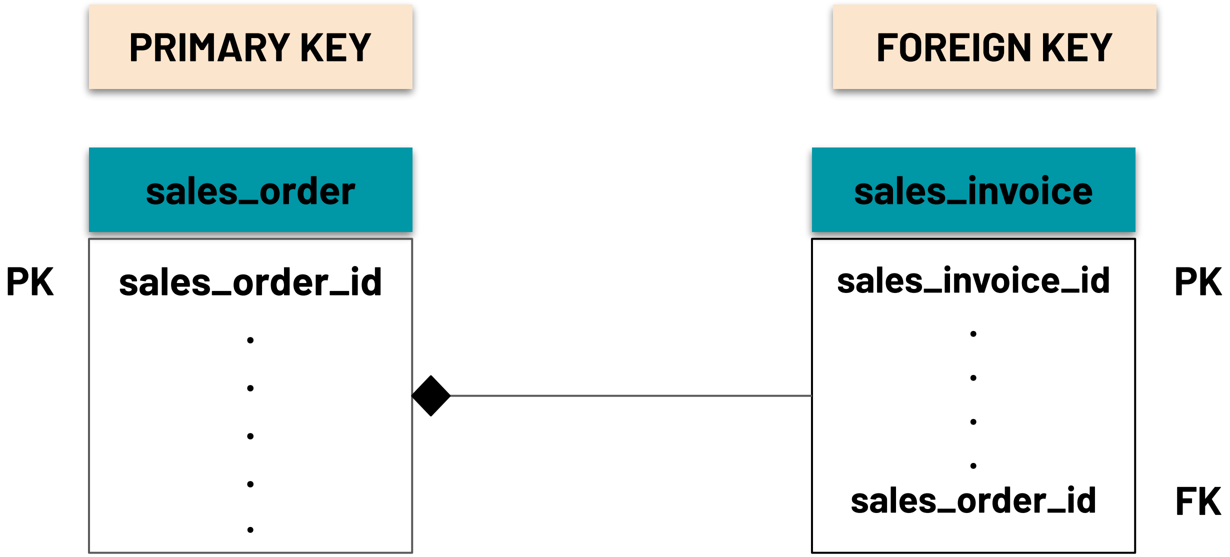
| DEPARTMENT_ID | LOCATION_ID |
|---:|---:|
| 10 | 1700 |
| 20 | 1800 |
| 50 | 1500 |
| 60 | 1400 |
| 80 | 2500 |
| 90 | 1700 |
| 110 | 1700 |
| 190 | 1700 |

# JOIN CONDITIONS



Pemisalan : TABLE 1 = sales_invoice
TABLE 2 = sales_order

# JOIN CONDITIONS (cont)



PRIMARY KEY

FOREIGN KEY

sales_order

PK sales_order_id

sales_invoice

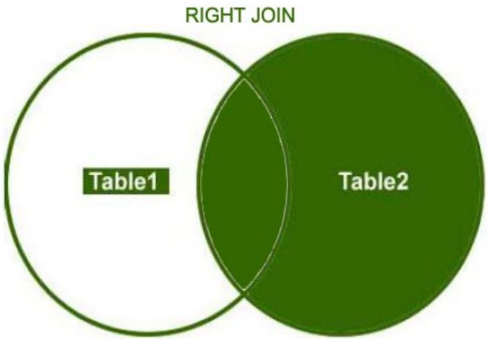
sales_invoice_id PK

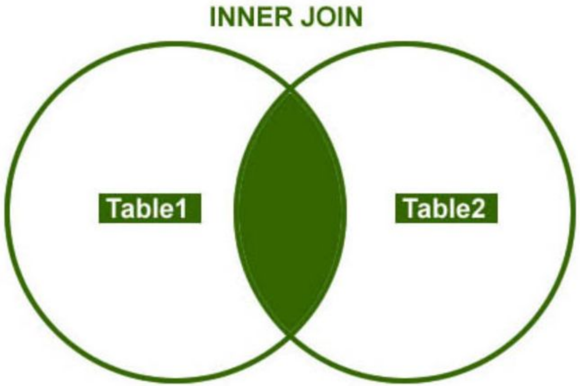
sales_order_id FK

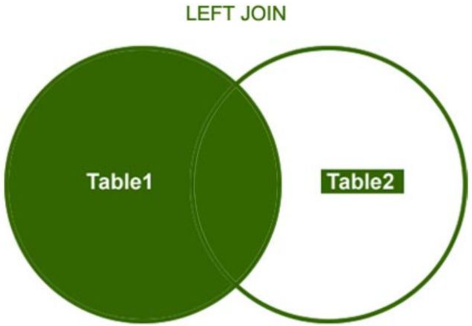
**sales_order_id adalah PK di kolom sales_order tetapi FK di kolom sales_invoice. Jika tidak terbentuk sales_order maka kolom sales_invoice tidak terbentuk**

**sales_invoice_id adalah PK di kolom sales_invoice**

# JOIN STATEMENT







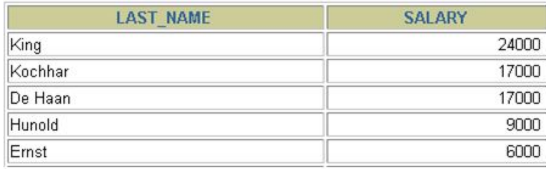
# “Arithmetic SQL”

Create expressions with number and date data by using arithmetic operators.

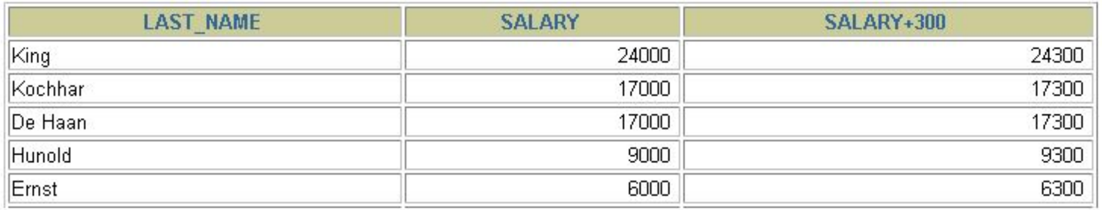
| Operator | Description |
|---|---|
| **+** | **Add** |
| **-** | **Subtract** |
| ***** | **Multiply** |
| **/** | **Divide** |

# “SELECT Statement using Arithmetic SQL”

```
SELECT last_name, salary, salary + 300
FROM   employees;
```



| LAST_NAME | SALARY |
|---|---|
| King | 24000 |
| Kochhar | 17000 |
| De Haan | 17000 |
| Hunold | 9000 |
| Ernst | 6000 |



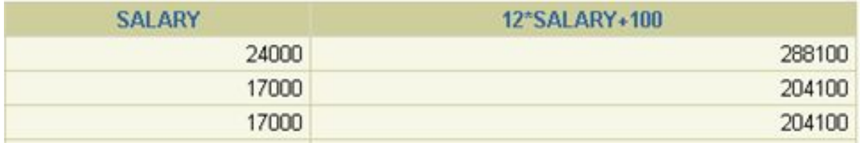
| LAST_NAME | SALARY | SALARY+300 |
|---|---|---|
| King | 24000 | 24300 |
| Kochhar | 17000 | 17300 |
| De Haan | 17000 | 17300 |
| Hunold | 9000 | 9300 |
| Ernst | 6000 | 6300 |

# “Operator Precedence”

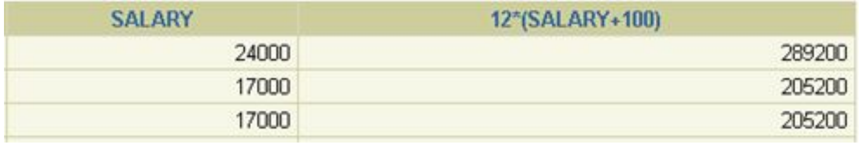
```
SELECT
last_name,
salary,
12*salary+100
FROM employees;
```

VS

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



| SALARY | 12*SALARY+100 |
|---|---|
| 24000 | 288100 |
| 17000 | 204100 |
| 17000 | 204100 |

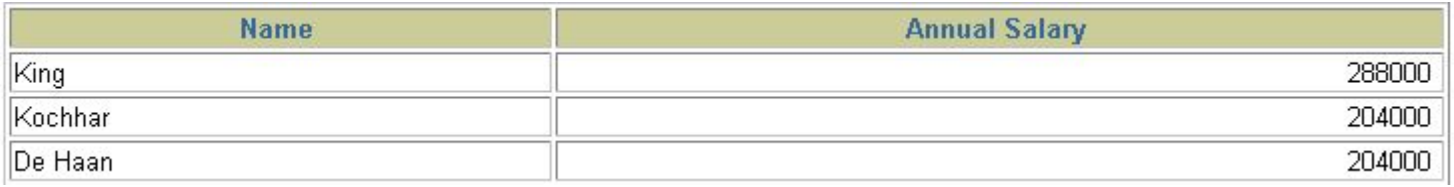


| SALARY | 12*(SALARY+100) |
|---|---|
| 24000 | 289200 |
| 17000 | 205200 |
| 17000 | 205200 |

# “ALIASES”

```
SELECT
last_name AS name,
salary*12 AS Annual Salary
FROM employees;
```

```
SELECT
last_name "Name",
salary*12 "Annual Salary"
FROM employees;
```



| Name | Annual Salary |
|---|---|
| King | 288000 |
| Kochhar | 204000 |
| De Haan | 204000 |

# “WHAT IS THE BENEFIT OF USING SQL ?”

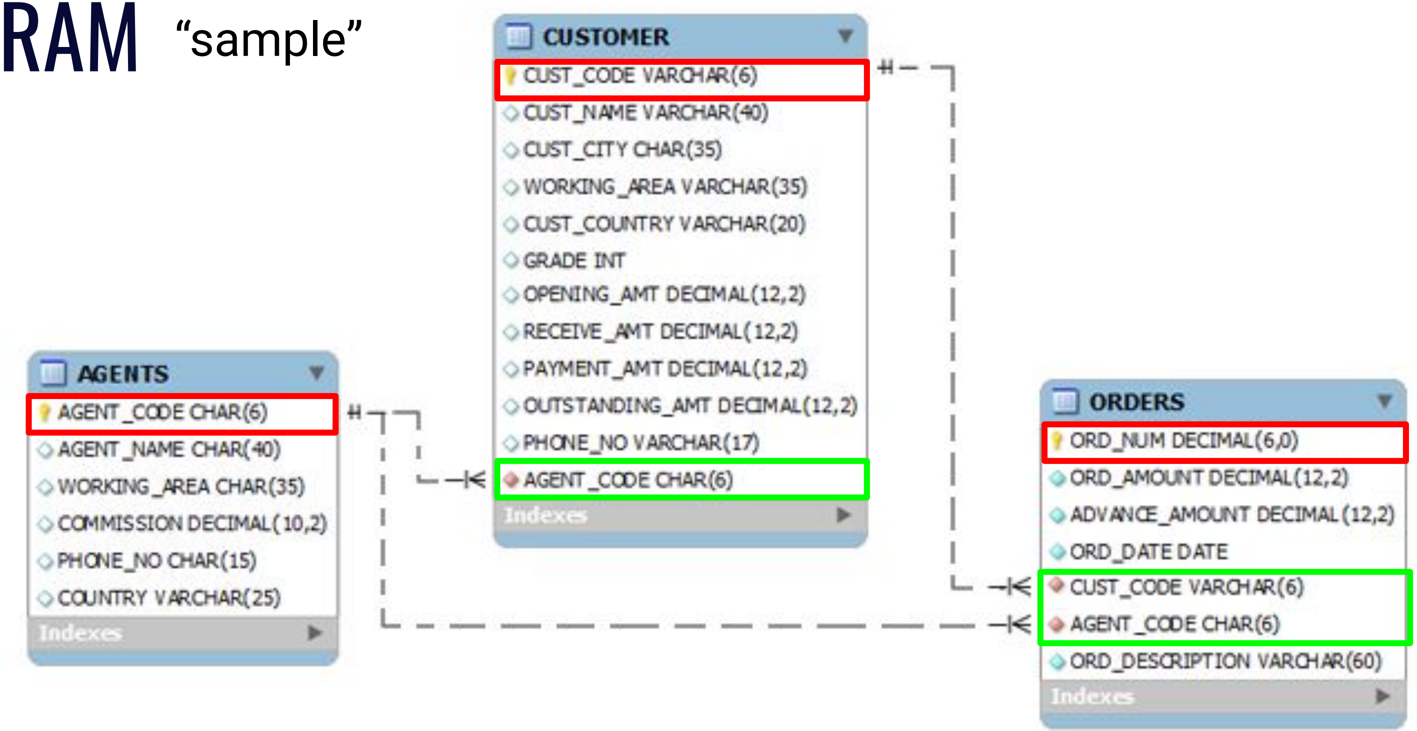
BIG DATA

R PYTHON

DATA PRACTITIONER

EASY

# ENTITY RELATIONSHIP DIAGRAM "sample"



CUSTOMER

CUST_CODE VARCHAR(6)
CUST_NAME VARCHAR(40)
CUST_CITY CHAR(35)
WORKING_AREA VARCHAR(35)
CUST_COUNTRY VARCHAR(20)
GRADE INT
OPENING_AMT DECIMAL(12,2)
RECEIVE_AMT DECIMAL(12,2)
PAYMENT_AMT DECIMAL(12,2)
OUTSTANDING_AMT DECIMAL(12,2)
PHONE_NO VARCHAR(17)
AGENT_CODE CHAR(6)
Indexes

AGENTS

AGENT_CODE CHAR(6)
AGENT_NAME CHAR(40)
WORKING_AREA CHAR(35)
COMMISSION DECIMAL(10,2)
PHONE_NO CHAR(15)
COUNTRY VARCHAR(25)
Indexes

ORDERS

ORD_NUM DECIMAL(6,0)
ORD_AMOUNT DECIMAL(12,2)
ADVANCE_AMOUNT DECIMAL(12,2)
ORD_DATE DATE
CUST_CODE VARCHAR(6)
AGENT_CODE CHAR(6)
ORD_DESCRIPTION VARCHAR(60)
Indexes

🟥 PRIMARY KEY

🟩 FOREIGN KEY

# “How to Write SQL STATEMENT”

1. **SQL statements are not case-sensitive;**
2. **SQL statements can be on one or more lines;**
3. **Keywords cannot be abbreviated or split across lines;**
4. **Clauses are usually placed on separate lines;**
5. **Indents are used to enhance readability;**
6. **SQL statements can optionally be terminated by a semicolon (;). Semicolons are required if you execute multiple SQL statements;**
7. **You are required to end each SQL statement with a semicolon (;).**

# THANKYOU