Course 5:
Database & SQL for Data Science with Python

Materi : tlkm.id/RxeqqI

AGENDA

01

Introduction to Database & SQL

Definition, Why, SQL in profesion, Tools

02

Learn SQL Syntax – Database & Table

Definition, Data Type, Modify Database & Table

03

Learn SQL Syntax –Viewing Data

Definition, Selecting table, Filtering, Ordering

04

Learn SQL Syntax – Agregating Data

Grouping Data, Agregation Function

05

Learn SQL Syntax – Merging Data

Joining Table, Union Table

06

Additional Knowledge

Case Statement, Common Function

07

Working with Python

SQL Magic

08

Study Case

Practical based on Business case

01

Introduction to Database & SQL

Definition, Why, SQL in profesion, Tools

DATABASE

A **database** is an organized collection of structured information, or data, typically stored electronically in a computer system.
A good database is crucial to any company or organization. This is because the database stores all the pertinent details about the company such as employee records, transactional records, salary details etc.

Why we need it ?

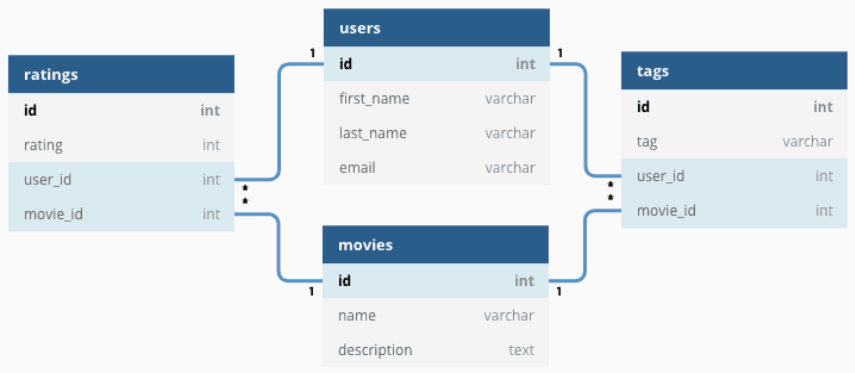
- Manages large amounts of data
- Accurate
- Easy to update data
- Security of data
- Data integrity
- Easy to research data

Database Management

A database is usually controlled by a database management system (DBMS).

RDBMS (Relational Database Management System)

A relational database refers to a database that stores data in a structured format, using rows and columns. This makes it easy to locate and access specific values within the database. It is "relational" because the values within each table are related to each other.



SQL in RDBMS

- Structured Query Language (SQL) is a programming language that is typically used in relational database management systems.
- We use SQL to be able to communicate with databases directly.
- It is capable to perform tasks such as creating, reading, updating, and deleting tables in a database.

Profesion & SQL

And the good news is, if you're skilled in sql, there are a lot of great jobs to choose from.

Here are the top 10:

- Business Analyst
- Senior Software Engineer
- Senior Sql Server Database Administrator
- Quality Assurance Tester
- Net Developer
- Systems Administrator
- Software Engineer
- Sql Server Developer
- Quality Assurance Analyst
- Senior Oracle Database Administrator

Source : zippia

Software

SQL Command

Data Definition Language (DDL)

Actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.

Ex: Create, Drop, Alter, Truncate

Data Manipulation Language (DML)

The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

Ex: Select, Insert, Delete, Update

Data Control Language (DCL)

which includes commands such as GRANT and mostly concerned with rights, permissions and other controls of the database system.

Ex: Grant, Revoke

TCL(transaction Control Language)

commands are used to manage transactions in the database

Ex: Commit, Rollback

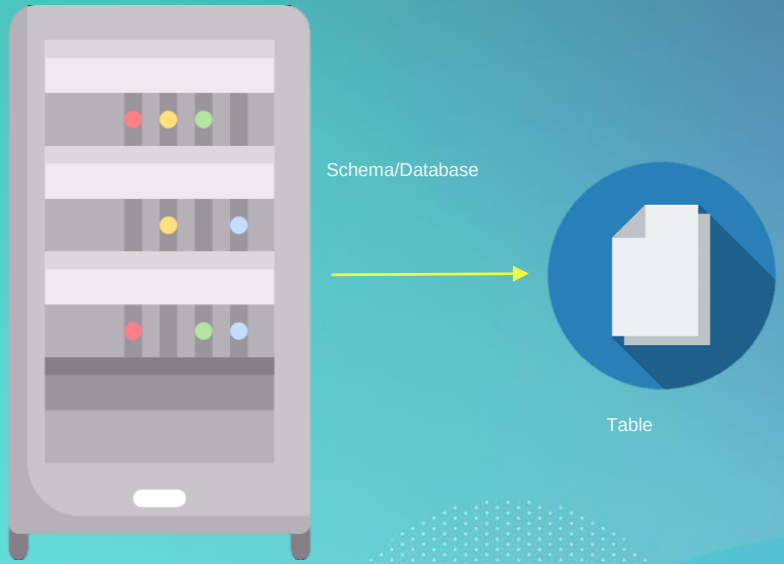
02

Learn SQL Syntax – Database & Table

Definition, Data Type, Modify Database & Table

Schema & Table

A table is a collection of related data held in a table format within a database.
A Schema/database consist of many tables.



ID	NAME	CLASS	MARK	SEX
1	John Deo	Four	75	female
2	Max Ruin	Three	85	male
3	Arnold	Three	55	male
4	Krish Star	Four	60	female
5	John Mike	Four	60	female
6	Alex John	Four	55	male
7	My John Rob	Fifth	78	male
8	Asruid	Five	85	male
9	Tes Qry	Six	78	male
10	Big John	Four	55	female

Command in Database

- **Creating Schema**
  used for creating a new database

```
CREATE SCHEMA schema_name;
```

Command SQL

Schema Name

- **Drop Schema**
  used for dropping Schema existing

```
DROP SCHEMA schema_name RESTRICT;
```

- **Using Schema**
  used for use schema

```
SET SCHEMA schema_name;
```

Command in Table

- Creating Table

  You can create a new table based on existing table or not

Based on your Criteria

```
CREATE TABLE table_name
(
  column1 datatype optional parameter,
  column2 datatype,
  column3 datatype,
  ....
);
```

Example

```
create table Customer (
                Id int  primary key not null,
                FirstName varchar(20) not null,
                LastName  varchar(20),
                City varchar(30),
                Country varchar(30),
                Phone varchar(20),
                Age int
);
```

Based on existing Table

```
CREATE TABLE new_table AS (
    SELECT *
    FROM old_table
)with data;
```

Example

```
create table Customer_backup
as (
select * from Customer
)with data;
```

Command in Table

- Inserting Table

  You can insert data into table with these commands

The first way specifies both the column names and the values to be inserted

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

Example

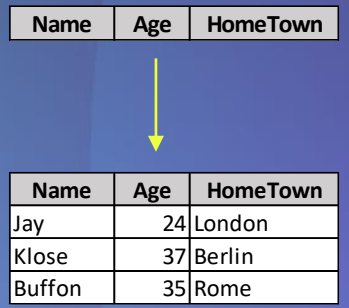
```
INSERT INTO Customer (Id,FirstName,LastName,City,Country,Phone,Age)
VALUES (1,'Maria','Anders','Berlin','Germany','030-0074321',31);
```

If you are adding values for all the columns of the table

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

Example

```
INSERT INTO Customer VALUES
(2,'UvuvwevwevweOnyetenvewveUgwemubwemOssas','Trujillo','México
D.F.','Mexico','(5) 555-4729',26);
```



Name	Age	HomeTown
Jay	24	London
Klose	37	Berlin
Buffon	35	Rome

Command in Table

- ALTER TABLE

  is used to rename table, rename column of table, add, delete, or modify columns in an existing table.

To change name of table

```
RENAME TABLE table_old TO table_new;
```

To change name of column

```
ALTER TABLE table_name RENAME COLUMN old_column TO new_column;
```

To add a column of table

```
ALTER TABLE table_name ADD COLUMN column_name datatype;
```

To Set datatype a column of table

```
ALTER TABLE table_name ALTER COLUMN column_name SET DATA TYPE datatype;
```

To drop a column of table

```
ALTER TABLE table_name DROP COLUMN column_name datatype;
```

Command in Table

- Updating Value of Table

  UPDATE statement is used to modify the existing records in a table.

```
UPDATE table_name
SET c1 = v1, c2 = v2, ... , cn = vn
WHERE condition;
```

Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!

Name	Age	HomeTown
Jay	24	London
Klose	37	Berlin
Buffon	35	Rome

Name	Age	HomeTown
Jay	24	Jakarta
Klose	37	Jepang
Buffon	35	Rome

Command in Table

- Deleting Rows Table

  used for delete specific rows of table

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

example

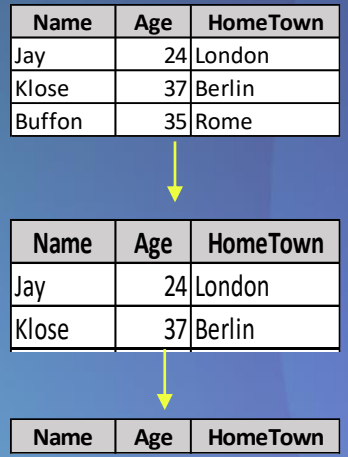
```
DELETE FROM dept WHERE LOC='ROME';
```

delete all rows of data in table

```
TRUNCATE TABLE table_name IMMEDIATE;
```

example

```
TRUNCATE TABLE dept IMMEDIATE;
```



Name	Age	HomeTown
Jay	24	London
Klose	37	Berlin
Buffon	35	Rome

Name	Age	HomeTown
Jay	24	London
Klose	37	Berlin

Name	Age	HomeTown

Command in Table

- Drop Table
  used for dropping table

```
DROP TABLE tablename;
```

```
DROP TABLE [IF EXIST] tablename;
```

Data Types

Type of Data used for stored table. Remember to be wise in defining data types.

DATA TYPE	DESCRIPTION
SMALLINT [16BIT]	Using this you can insert small int values into columns. Min Value -32768, Max Value 32767
INTEGER [32BIT]	Using this you can insert large int values into columns. Min Value -2147483648, Max Value 214748364
BIGINT [64BIT]	Using this you can insert larger int values into columns. Min Value -9223372036854775808, Max Value 9223372036854775807
CHAR (fixed length)	Fixed length of Character strings.
VARCHAR	Varying length character strings.
TIME	It represents the time of the day in hours, minutes and seconds.
TIMESTAMP	It represents seven values of the date and time in the form of year, month, day, hours, minutes, seconds and microseconds.
DATE	It represents date of the day in three parts in the form of year, month and day.
DECIMAL(p,s)	p adalah presisi yang merupakan jumlah maksimum digit desimal termasuk bagian utuh dan bagian pecahan. 12.345 memiliki presisi maksimum 5. s disebut skala yang merupakan jumlah angka desimal di bagian pecahan misalnya, untuk nomor 12.345, s adalah 3

Data	CHAR(5)	Ukuran Peyimpanan	VARCHAR(5)	Ukuran Peyimpanan
' '	' '	5 byte	' '	1 byte
'du'	'du '	5 byte	'du'	3 byte
'dunia'	'dunia'	5 byte	'dunia'	6 byte
'duniailkom'	'dunia'	5 byte	'dunia'	6 byte

03

Learn SQL Syntax –Viewing Data

Definition, Selecting table, Filtering, Ordering

Viewing Table

Definition

We can modify what we want to see based on specified condition such as specific columns / rows.

CUSTOMER						
Id	**FirstName**	**LastName**	**City**	**Country**	**Phone**	**Age**
1	Maria	Anders	Berlin	Germany	030-0074321	31
2	UvuvwevwevweOnyeten	Trujillo	México D.F.	Mexico	(5) 555-4729	26
3	Antonio	Moreno	México D.F.	Mexico	(5) 555-3932	28
4	Thomas	Hardy	London	UK	(171) 555-7788	30
5	Christina	Berglund	Luleå	Sweden	0921-12 34 65	
6	Hanna	Moos	Mannheim	Germany	0621-08460	35
7	Frédérique	Citeaux	Strasbourg	France	88.60.15.31	31
8	Martín	Sommer	Madrid	Spain	(91) 555 22 82	28
9	Laurence	Lebihan	Marseille	France		18
10	Elizabeth	Lincoln	Tsawassen	Canada	(604) 555-4729	26

Select command

If u want to view all column of table

```
SELECT * FROM table_name;
```

If u want to view specific columns of table

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

If u want to limit view of table

```
SELECT *
FROM table_name LIMIT numberofrows;
```

If u want to distinct view of table

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

Ordering Rows

If u want to view rows based on ascending order, just type

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC
```

If u want to view rows based on descending order, just type

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... DESC;
```

Filtering Table

WHERE

Viewing table based specific condition of columns.
WHERE clause is used to extract only those records that fulfill a specified condition.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

OPERATOR	DESCRIPTION	EXAMPLE of CONDITION
=	Equal	`where name='ANTON'; where age=17`
>	Greater than	`where age >17`
<	Less than	`where age <17`
>=	Greater than or equal	`where age >=17`
<=	Less than or equal	`where age <=17`
<>	Not equal. Note: In some versions of SQL this operator may be written as !=	`where name!='ANTON'; where age!=17`
BETWEEN	Between a certain range	`where age between 17 and 25`
LIKE	Search for a pattern	`where name like '%ANTON%'`
IN	To specify multiple possible values for a column	`where age in (17,18,19)`

Filtering Table

The AND and OR operators are used to filter records based on more than one condition:

- The AND operator displays a record if all the conditions separated by AND are TRUE.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

- The OR operator displays a record if any of the conditions separated by OR is TRUE.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

The NOT operator displays a record if the condition(s) is NOT TRUE.

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

04

Learn SQL Syntax –Agregating Data

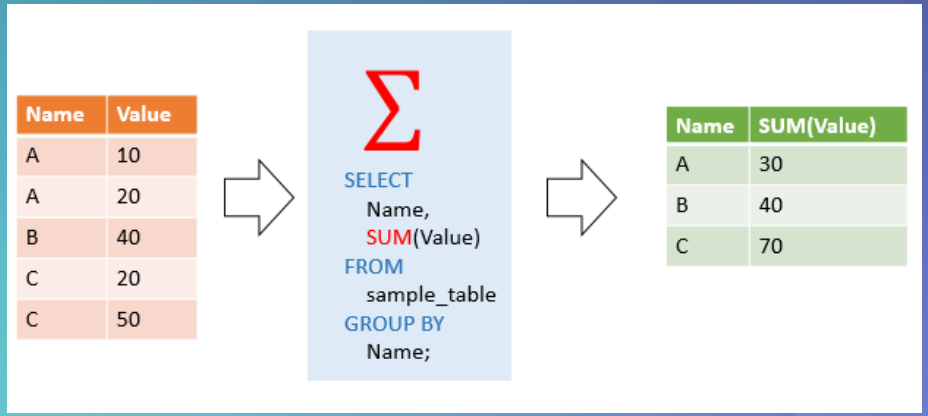
Grouping Data, Agregation Function

Grouping Data

GROUP BY

groups rows that have the same values into summary rows, like "find the number of customers in each country". This statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

```
SELECT column_name
FROM table_name
GROUP BY column_name
```



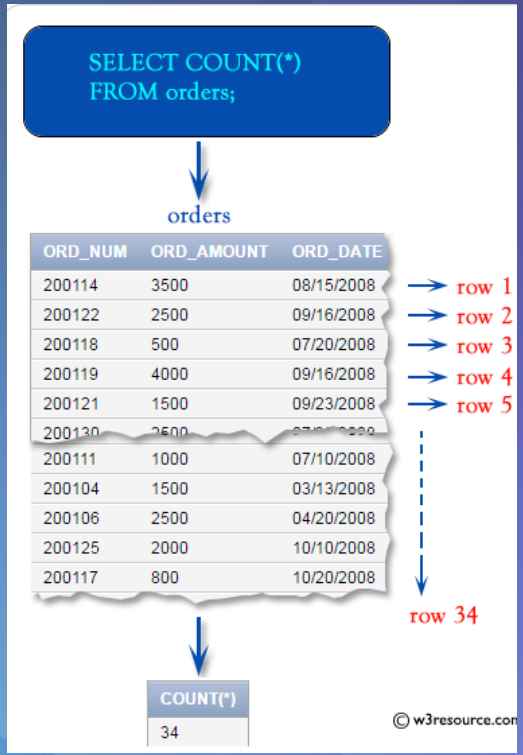
Agregate Function

COUNT()
function returns the number of rows that matches a specified criterion. It sets the number of rows or non NULL column values.
Basic command

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

Breakdown by another column

```
SELECT COUNT(column_name1), column_name2
FROM table_name
GROUP BY column_name2;
```



Agregate Function

SUM()

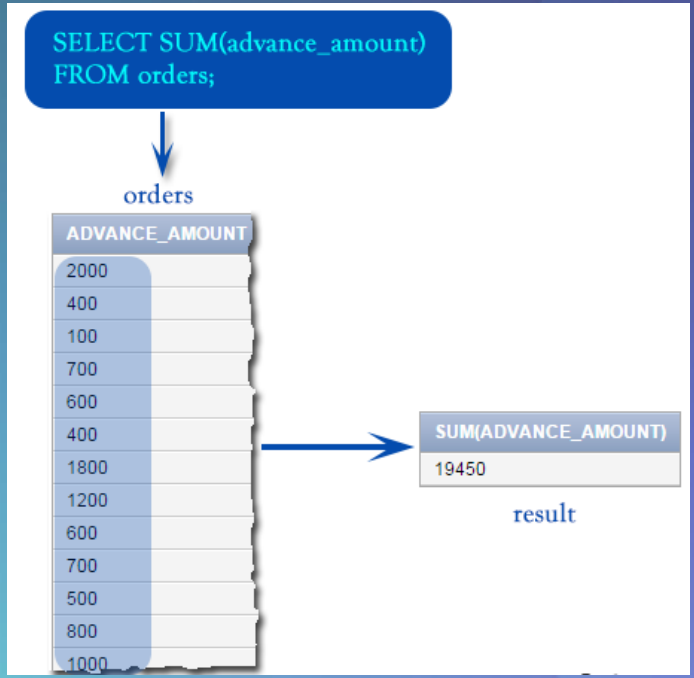
function returns the total sum of a numeric column.

Basic command

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

Breakdown by another column

```
SELECT SUM(column_name1), column_name2
FROM table_name
GROUP BY column_name2;
```



Agregate Function

AVG()

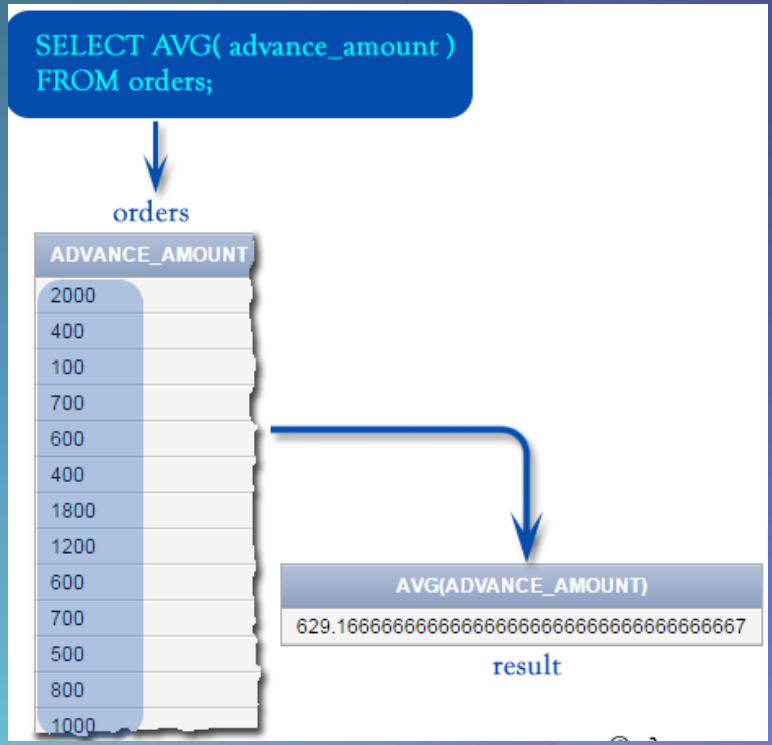
function returns the average of a numeric column.

Basic command

```
SELECT AVG(column_name)
FROM table_name;
```

Breakdown by another column

```
SELECT AVG(column_name1), column_name2
FROM table_name
GROUP BY column_name2;
```



Agregate Function

MIN()

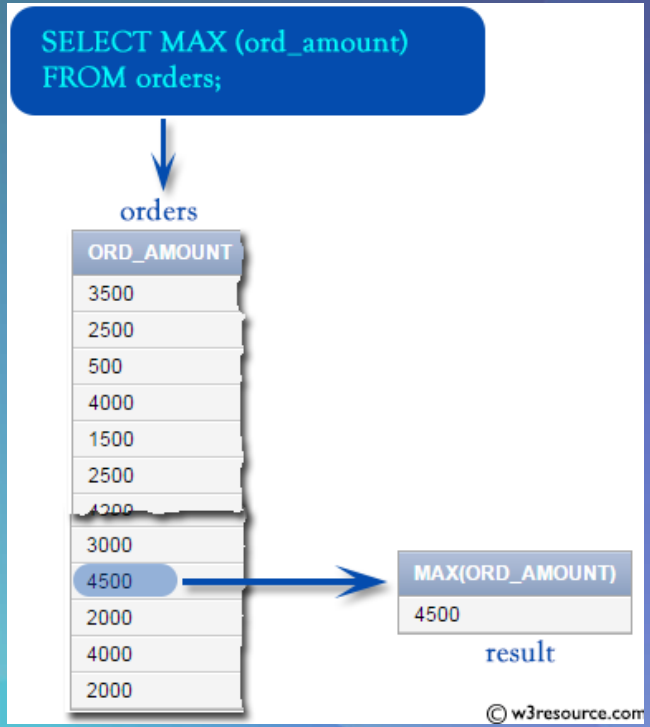
is used to find the minimum value or lowest value of a column or expression. This function is useful to determine the smallest of all selected values of a column.

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

MAX()

is used to find the maximum value or lowest value of a column or expression. This function is useful to determine the highest of all selected values of a column.

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



Agregate Function

HAVING CLAUS

specifies that an SQL SELECT statement must only return rows where aggregate values meet the specified conditions.

```
SELECT column_name
FROM table_name
GROUP BY column_name
HAVING condition;
```

05

Learn SQL Syntax – Multi Tables

Joining Table, Union Table

SUB QUERIES

A query inside another query.

- Sub Queries in list of columns

```
select id
      , FirstName
      , (select avg(age) from customer) as avg_age
from customer;
```

- Sub Queries to evaluate Aggregate Function

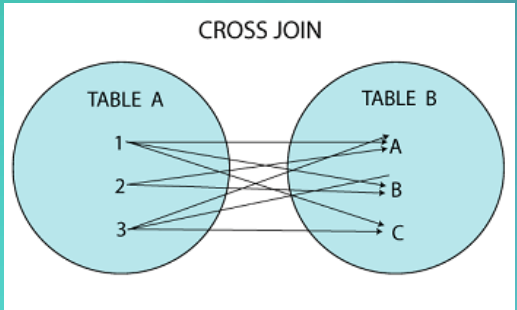
```
select id
      , FirstName
      , Age
from customer
where age < (select avg(age) from customer);
```

- Sub Queries in FROM clause

```
SELECT *
from (select *
      ,(select avg(age) from customer) as avg_age
      from customer)yaya;
```

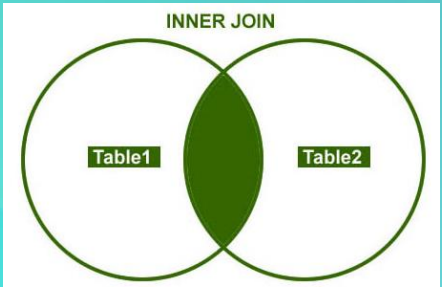
IMPLICIT JOIN

- The Result is Cartesian Join (Cross Join)




```
SELECT *
FROM table_name1, table_name2;
```

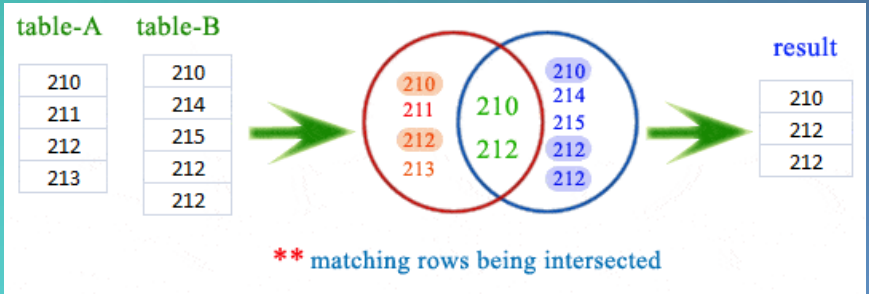
- The result is Inner Join




```
SELECT *
FROM table_name1, table_name2
WHERE
table_name1.column_name1=table_name2.column_name2=;
```

JOIN OPERATORS – INNER JOIN

The INNER JOIN selects all rows from both participating tables as long as there is a match between the columns.
An SQL INNER JOIN is same as JOIN clause, combining rows from two or more tables.

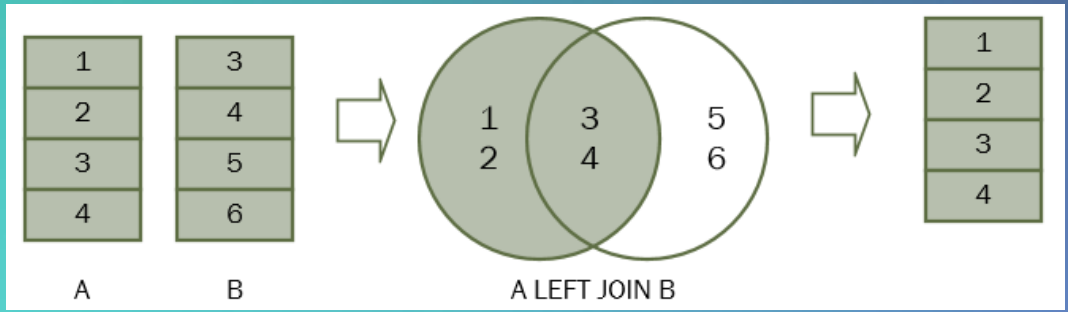



```
SELECT *
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

```
SELECT *
FROM table1
JOIN table2
ON table1.column_name = table2.column_name;
```

JOIN OPERATORS – LEFT JOIN

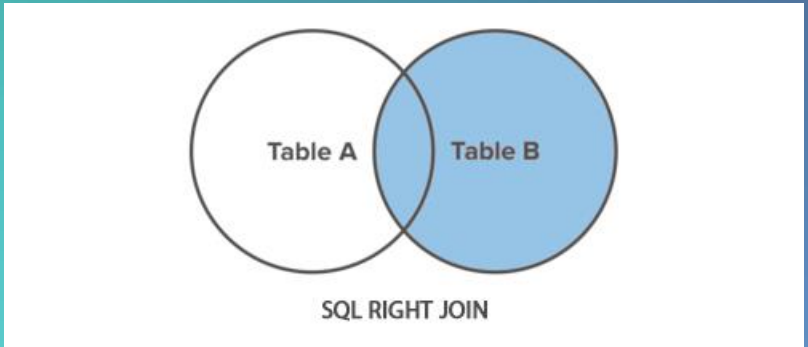
LEFT JOIN returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.




```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

JOIN OPERATORS – RIGHT JOIN

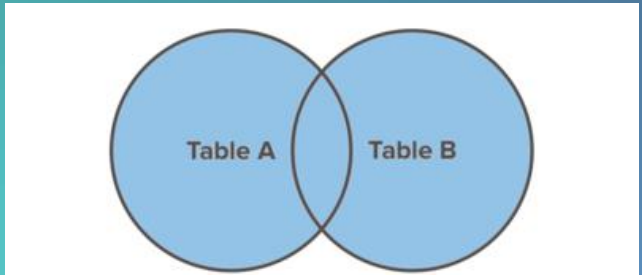
RIGHT JOIN returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.




```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

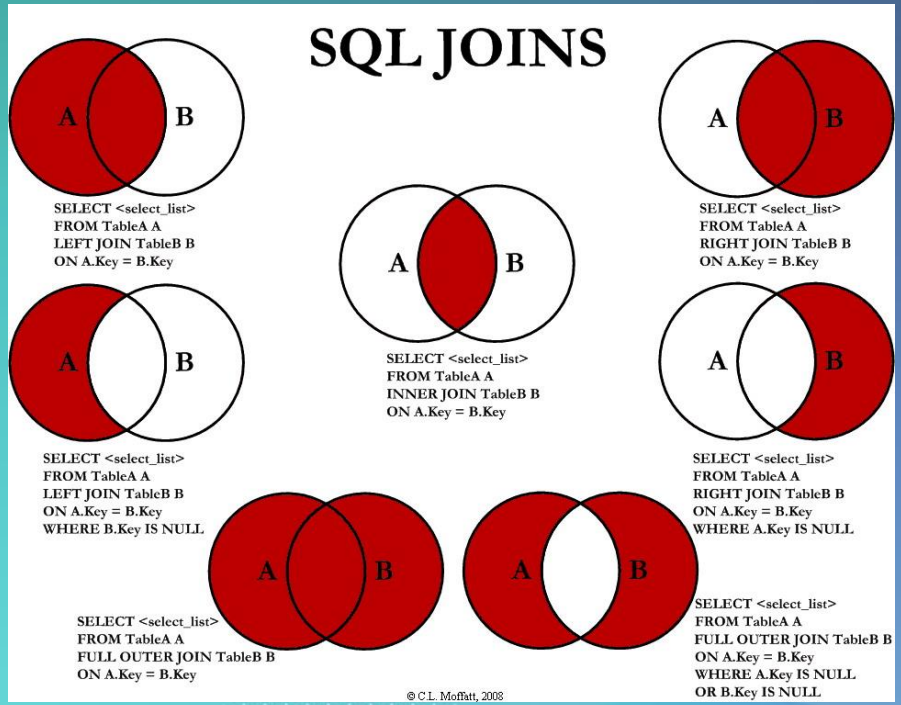
JOIN OPERATORS – FULL JOIN

FULL JOIN returns all records when there is a match in left (table1) or right (table2) table records.




```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name;
```

JOIN OPERATORS



UNION

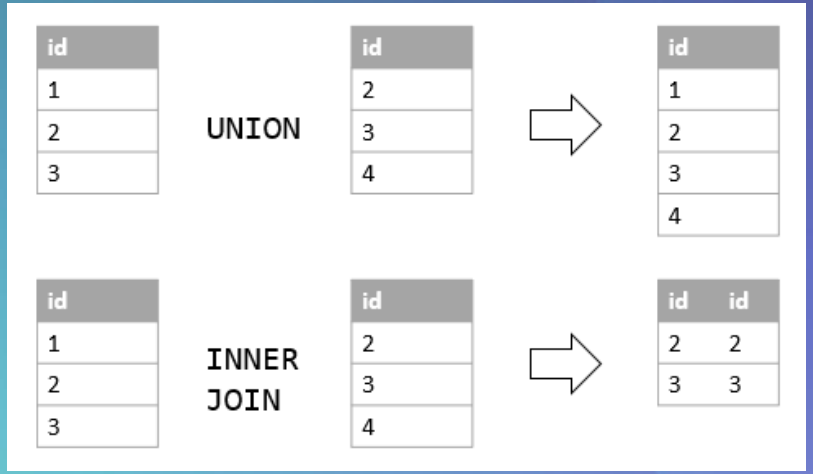
The UNION operator is used to combine the result-set of two or more SELECT statements.

- Each SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in each SELECT statement must also be in the same orde

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

To allow all duplicate values

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```



06

Additional Knowledge

Case Statement, Common Function

CASE STATEMENT

The CASE statement goes through conditions and returns a value when the first condition is met (like an if-then-else statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause.

```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    WHEN conditionN THEN resultN
    ELSE result
END;
```

```
SELECT OrderID, Quantity,
CASE
    WHEN Quantity > 30 THEN 'The quantity is greater than 30'
    WHEN Quantity = 30 THEN 'The quantity is 30'
    ELSE 'The quantity is under 30'
END AS QuantityText
FROM OrderDetails;
```

OrderID	Quantity	QuantityText
1	40	The quantity is greater 30
2	20	The quantity is under 30
3	35	The quantity is greater 30
4	30	The quantity is 30

Date and Time Functions

- To display the day of the month

```
EXTRACT(DAY FROM column_name)
```

```
DAY(column_name)
```

NOW_TIMESTAMP	EXTRACTION	DAY_FUNCTION
2021-07-21 03:59:33.688875	21	21

- To display the month

```
EXTRACT(MONTH FROM column_name)
```

```
MONTH(column_name)
```

NOW_TIMESTAMP	EXTRACTION	MONTH_FUNCTION
2021-07-21 03:54:24.733249	7	7

- To display the hour

```
EXTRACT(HOUR FROM column_name)
```

```
HOUR(column_name)
```

NOW_TIMESTAMP	EXTRACTION	HOUR_FUNCTION
2021-07-21 03:52:24.201184	3	3

- To display the minute

```
EXTRACT(MINUTE FROM column_name)
```

```
MINUTE(column_name)
```

NOW_TIMESTAMP	EXTRACTION	MINUTE_FUNCTION
2021-07-21 03:47:27.017292	47	47

Date and Time Functions

- To display days difference

```
DAYS(END_DATE)-DAYS(START_DATE)
```

- To display the month

Value	Interval
1	Microseconds
2	Seconds
4	Minutes
8	Hours
16	Days
32	Weeks
64	Months
128	Quarters
256	Years

```
TIMESTAMPDIFF(value,CHAR(column_timestamp1 - column_timestamp2))
```

Example:

```
TIMESTAMPDIFF(8,CHAR(TIMESTAMP1 - TIMESTAMP2)) AS HOURS_DIFFERENCE
```

```
TIMESTAMPDIFF(4,CHAR(TIMESTAMP1 - TIMESTAMP2)) AS MINUTES_DIFFERENCE
```

TIMESTAMP1 ↑↓	TIMESTAMP2	HOURS_DIFFERENCE	MINUTES_DIFFERENCE
2021-03-27 12:07:58.065497	2021-03-22 12:07:58.065497	24	1440

Scalar & String Function

- To returns the length of expression in the implicit or explicit string unit. `LENGTH(column_name)`
- To upper / lower case character `UPPER(column_name)` `LOWER(column_name)`

ANIMAL	LENGTH_COLUMN	UPPER_COLUMN	LOWER_COLUMN
Cat	3	CAT	cat
Dog	3	DOG	dog
Dog	3	DOG	dog
Parrot	6	PARROT	parrot

- To merge string columns

```
CONCAT(column1,column2)
```

```
Column1 || column2 || …… ||columnN
```

- To Replace value

```
REPLACE(columnName, '%', '0')
```

Scalar & String Function

- NULLIF function returns a null value if the arguments are equal, otherwise it returns the value of the first argument.

```
CASE WHEN expression1=expression2 THEN NULL
     ELSE expression1 END
```

→

```
NULLIF(expression1, expression1)
```

- COALESCE function returns a null value if the arguments are equal, otherwise it returns the value of the first argument.

```
CASE WHEN expression1 is NOT NULL THEN expression1
     ELSE expression2 END
```

→

```
COALESCE(expression1, expression1)
```

ANIMAL ↑↓	QUANTITY	NULLIF_FUNCTION	COALESCE_FUNCTION
Fish			2
Dog	2		2
Goldfish	24	24	24

```
,NULLIF(QUANTITY,2
,COALESCE(QUANTITY,2)
```

07

Working with Python

SQL Magic

SQL Magic

- Installing Package

To communicate with SQL Databases from within a JupyterLab notebook, we can use the SQL "magic" provided by the ipython-sql, and ibm_db_sa the driver.

```
!pip install sqlalchemy==1.3.9
!pip install ibm_db_sa
!pip install ipython-sql
```

- Load ipython-sql extension

```
%load_ext sql
```

- Set Connection

```
%sql ibm_db_sa://my-username:my-password@hostname:port/BLUDB?security=SSL
```

SQL Magic

- Using Python Variables in your SQL Statements

```
country = "Canada"
%sql select * from INTERNATIONAL_STUDENT_TEST_SCORES where country = :country
```

- Assigning the Results of Queries to Python Variables

```
variable = %sql SELECT * FROM TABLE_NAME;
```

- Converting Query Results to DataFrames

```
Import pandas as pd
dataframe = variable.DataFrame()
```

- Storing dataframe into table

```
%sql --persist dataframe
```

- Inserting dataframe into table

```
%sql --append dataframe
```

08

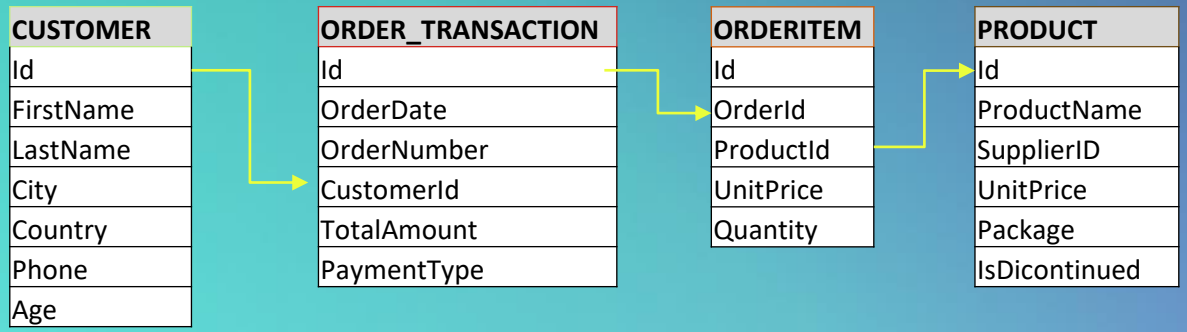
Study Case

Practice Makes Perfect

Study Case

Diketahui dalam sebuah minimarket deltamart , memiliki beberap data sebagai berikut:

- CUSTOMER, merupakan data profil customer minimarket
- ORDER_TRANSACTION, riwayat pembelanjaan customer
- ORDERITEM, detil pembelanjaan customer
- PRODUCT, Profil product yang dimiliki minimarket



TASK

Perusahaan tersebut ingin melakukan analisa pemodelan , dengan memprediksi apakah customer akan melakukan pembayaran secara cash atau tidak. Maka anda sebagai data scientist sebelum melakukan pemodelan akan mempersiapkan datamart terlebih dahulu.

COLUMN	DESCRIPTION	EXAMPLE
ID	Customer ID	1
Name	Gabungan FirstName & LastName	Anton Suhartono
Country	Asal Negara	Germany
flag_phone	Jika ada nomor telfonnya, tulis 1, jika tidak maka 0	1
Age	Umur, kalo ga ada datanya diasumsikan 25	17
LOS	Lama berlangganan dihitung semenjak Awal order barang, dalam satuan hari	365
TotalProduct	Total product seluruh yang pernah dibeli	50
TotalAmount	Total uang yang dikeluarkan dari seluruh barang yang dibeli	5000000
PaymentType	Type pembayaran, jika masih dalam bentuk cash maka tulis 1, lainnya 0	1

TASK

Perlu diperhatikan bahwasannya hasilnya berupa table berukuran 9 kolom 91 baris.

ID	Name	Country	flag_phone	Age	LOS	TotalProduct	TotalAmount	PaymentType
1	Anton Suhartono	Germany	1	17	365	50	5000000	1
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
91	Zbyszek Piestrzeniewicz	France	0	35	890	80	6780000	0

Silahkan dikerjakan sebagai PR

- Dikerjakan di notebook python
- Upload file notebook ke github masing-masing, diberi nama NIK_SQL_Data_Science
- Durasi pengerjaan 2 minggu
- Pastikan tampilkan hasil table finalnya dengan perintah select

Thank You.