Structured Query Language (SQL)

By: Mhd Shadi Hasan

What is SQL?

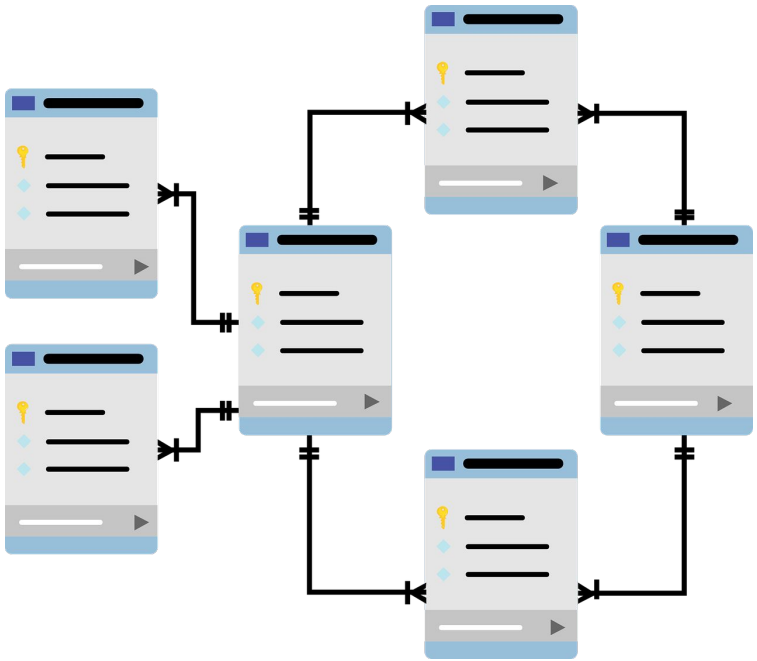
simi- structured data
ex : as Sensor monitor and analytic request from app as json

- A language used to manage and query data from databases.
- Particularly used with structured data in relational databases.
- A connection is established with a databases to perform SQL commands through a DBMS.
- You can use SQL with Python by utilizing special packages like SQLAlchemy.

DLL
DML
download in pc

Relational Database

module diagram



id	name	course
001	Ahmad	AI
002	Batool	CS
003	Firas	Business
004	Lama	CS

course_id	course_name	instructor
001	AI	Shadi
002	CS	Samar
003	Business	Shadi

SELECT

Used to query data from a database table. You can select specific columns or select all columns in a table.

```
SELECT column_1, column_2, column_3 FROM table_name;
```

```
SELECT * FROM table_name;
```

SELECT DISTINCT

This will return only the distinct values and ignore duplicate ones. You can bind that with the count function to return the number or distinct values.

```
SELECT DISTINCT column_1 FROM table_name;
```

```
SELECT COUNT(DISTINCT column_1) FROM table_name;
```

ORDER BY

This is used for sorting the query result. You can sort in ascending or descending order. By default, sorting is done in ascending order, for descending order you have to specify. You can sort alphabetically on a text column.

```
SELECT column_1, column_2 FROM table_name
ORDER BY column_1 DESC;
```

```
SELECT * FROM students ORDER BY name;
```

WHERE

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

Often you do not want to query all the records in a table.

Therefore, filtering is needed, or you want records that satisfy a condition.

```
SELECT column_1, column_2 FROM table_name WHERE condition;
```

```
SELECT * FROM students WHERE course='CS';
```

AND, OR, NOT

You can set multiple conditions in a WHERE statement. This is achieved via using AND, OR, NOT logic operators.

```
SELECT column_1, column_2 FROM table_name
WHERE condition_1 AND condition_2;
```

```
SELECT * FROM students
WHERE (course='CS' OR course='AI')
AND age>25;
```

```
SELECT * FROM students WHERE NOT course='Business';
```

INSERT INTO

You will often need to add new records to a table. This is achieved with the INSERT INTO command. If no value is added to a column the field will be considered NULL.

```
INSERT INTO table_name (column_1, column_2, column_3)
VALUES (value_1, value_2, value_3);
```

```
INSERT INTO table_name (column_1, column_2, column_3)
VALUES (value_1, value_2, value_3),
(value_1, value_2, value_3);
```

```
INSERT INTO table_name VALUES (value_1, value_2, value_3);
```

UPDATE

To update data in a specific record or group of records. BE CAREFUL! This is a bulk operation.

```
UPDATE table_name
SET column_1=value_1, column_2=value_2
WHERE condition;
```

```
UPDATE students
SET course = AI
WHERE id=002
```

DELETE

Delete records from a table following a specific condition. BE CAREFUL! This is a bulk operation.

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

```
DELETE FROM students
WHERE course='AI';
```

LIMIT

To select a specific number of records as a sample rather than return all records from a query. This is commonly used along with sorting. Note that LIMIT has alternatives in different databases.

```
SELECT column_1, column_2, column_3
FROM table_name
WHERE condition LIMIT n;
```

```
SELECT * from students
ORDER BY age DESC
LIMIT 10;
```

MIN and MAX

To return the smallest or largest value in a column use MIN and MAX functions.

```
SELECT MIN(column_1) FROM table_name
WHERE condition;
```

```
SELECT MAX(column_1) AS alias_name FROM table_name
WHERE condition
```

COUNT

To count the number of records that satisfy a condition.

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

```
SELECT COUNT(*) as alias_name FROM table_name
WHERE condition;
```

SUM and AVG

To return the total sum or the selected column use the SUM function. Similarly, AVG function returns the average.

```
SELECT SUM(column_1) FROM table_name
WHERE condition;
```

```
SELECT AVG(column_1) as alias_name FROM table_name
WHERE condition;
```

```
SELECT * FROM students
WHERE age > (SELECT AVG(age) from students);
```

GROUP BY

This is commonly used with aggregate functions like MIN, MAX, COUNT, SUM, and AVG. Resulting in the query result being performed on categories.

```
SELECT course, MIN(age) FROM students
GROUP BY course;
```

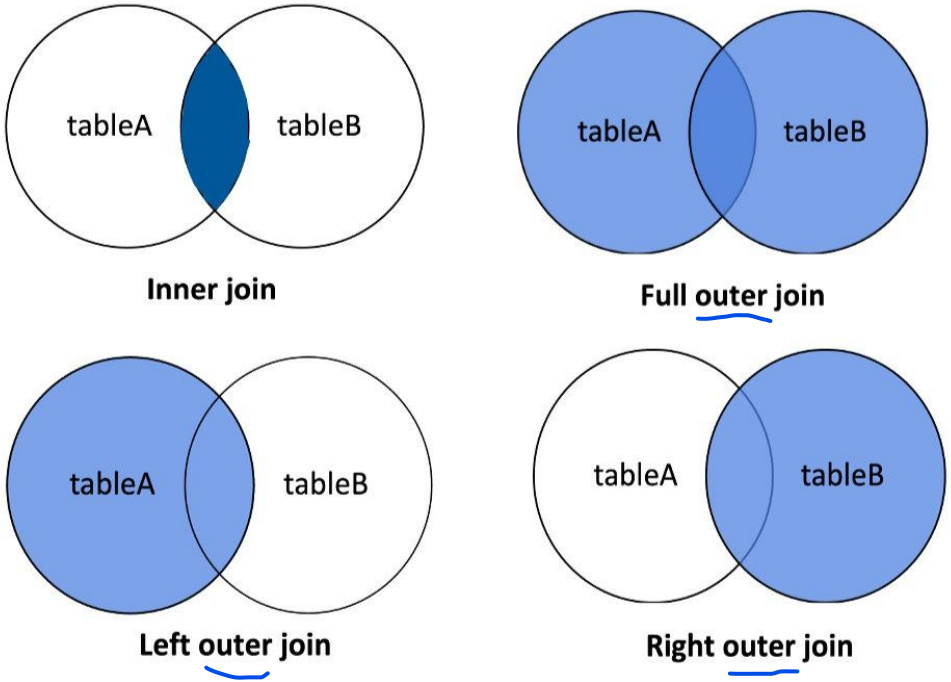
```
SELECT course, COUNT(*) as total_students FROM students
GROUP BY course;
```

```
SELECT course, AVG(age) as mean_age FROM students
GROUP BY course;
```

2 tables and more

JOIN

management db -> DDL -> create and relationships



Ven diagram

INNER JOIN

Return data from two tables where there is a match on selected records.

```
SELECT table_1.column_1, table_1.column_2, table_2.column_3
FROM table_1
INNER JOIN table_2
ON table_1.column_1 = table_2.column_3
```

LEFT JOIN

Return all data from the first table and data from the second table where there is a match.

```
SELECT table_1.column_1, table_1.column_2, table_2.column_3
FROM table_1
LEFT JOIN table_2
ON table_1.column_1 = table_2.column_3
```

HAVING

This is also used to filter returned data based on conditions. However, it is used with aggregate functions.

```
SELECT agg(column_1), column_2
FROM table_1
GROUP BY column_2
HAVING agg(column_1) condition
```

CASE

It is used to process data and create new columns. It functions in a way similar to if-else in Python. The results from CASE statement will be a new column in the returned data.

```
CASE
     WHEN condition_1 THEN value_1
     WHEN condition_2 THEN value_2
     WHEN condition_3 THEN value_3
     ELSE value_4
END;
```

CREATE TABLE

management db -> DDL -> create and relationships

Creating new tables in a database is important to save processed data, AI models results, or any resulting data from a data/ML pipeline.

The most important data types you will use are: INTEGER, FLOAT, VARCHAR, DATETIME.

format

YYYY-MM-DD HH:MM:SS

2024-01-03 16:33:40

```
CREATE TABLE table_name (
    column_1 data type,
    column_2 data type,
    column_3 data type,
    column_4 data type
);
```

DROP TABLE

To delete a table with all data in it.

```
DROP TABLE table_name;
```

ALTER TABLE

This statement updates a table in the database. It is different from UPDATE because this statement changes the table itself not the data in a set of records. It can be used to change and add columns in a table.

```
ALTER TABLE table_name ADD column_name data type;
```

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

```
ALTER TABLE table_name RENAME COLUMN old_name to new_name;
```

Specification on tables

When we create a table and add columns to it, we also add specifications to those columns. Such specifications include the data type as seen, but there are more specifications that can be added depending on the case.

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- DEFAULT
- AUTOINCREMENT