SQL is a standard database language used to access and manipulate data in databases. SQL stands for Structured Query Language.

By executing queries SQL can create, update, delete, and retrieve data in databases like MySQL, Oracle, PostgreSQL, etc.

Overall SQL is a query language that communicates with databases.

A database is the organized collection of structured data which is usually controlled by a database management system (DBMS).

DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema.

DDL is a set of SQL commands used to create, modify, and delete database structures but not data.

- **CREATE**: This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).
- **DROP**: This command is used to delete objects from the database.
- **ALTER:** This is used to alter the structure of the database.
- **TRUNCATE:** This is used to remove all records from a table, including all spaces allocated for the records are removed.
- **COMMENT**: This is used to add comments to the data dictionary.
- **RENAME:** This is used to rename an object existing in the database.

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

- **INSERT**: It is used to insert data into a table.
- **UPDATE:** It is used to update existing data within a table.
- **DELETE**: It is used to delete records from a database table.

DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system.

GRANT: This command gives users access privileges to the database.
REVOKE: This command withdraws the user's access privileges given by using the GRANT command.

Candidate Key: The minimal set of attributes that can uniquely identify a tuple is known as a candidate key.

- **Primary Key:** It is a unique key.

- It can identify only one tuple (a record) at a time

Foreign Key: It is a key it acts as a primary key in one table and it acts as secondary key in another table

---

select the first_name and last_name of all customers

who live in 'USA' and have the last name 'Doe'

SELECT first_name, last_name

FROM Customers

WHERE country = 'USA' AND last_name = 'Doe';

```
-- select first and last name of customers
-- who either live in the USA
-- or have the last name 'Doe'
SELECT first_name, last_name
FROM Customers
WHERE country = 'USA' OR last_name = 'Doe';

-- select customers who don't live in the USA
SELECT first_name, last_name
FROM Customers
WHERE NOT country = 'USA';

-- select customers who live in either USA or UK and whose age is less than 26
SELECT *
FROM Customers
WHERE (country = 'USA' OR country = 'UK') AND age < 26;
```

```
-- exclude customers who are from the USA and have 'Doe' as their last name
SELECT *
FROM customers
WHERE NOT country = 'USA' AND NOT last_name = 'Doe';

-- select the unique ages from the Customers table
SELECT DISTINCT age
FROM Customers;
```

The AS keyword is used to give columns or tables a temporary name that can be used to identify that column or table later

```
SELECT customer_id AS cid, first_name AS name
FROM Customers;
```

The LIMIT keyword in SQL allows you to specify the number of records to return in a query.

```
SELECT first_name, age
FROM Customers
LIMIT 2;

-- LIMIT 2 selects two results
-- OFFSET 3 excludes the first three results
SELECT first_name, last_name
FROM Customers
LIMIT 2 OFFSET 3;

-- select rows if the country is either USA or UK
SELECT first_name, country
FROM Customers
WHERE country IN ('USA', 'UK');
```

```
-- select rows with value 'USA' in the country column
SELECT first_name, country
FROM Customers
WHERE 'USA' IN (country);

-- select rows with value 'USA' in the country column
SELECT first_name, country
FROM Customers
WHERE 'USA' IN (country);

-- select only those customers who have placed an order
-- the subquery is enclosed within parentheses after the IN keyword
SELECT customer_id, first_name
FROM Customers
WHERE customer_id IN (
  SELECT customer_id
  FROM Orders
);

-- select rows where the amount is between 200 and 600
SELECT item, amount
FROM Orders
WHERE amount BETWEEN 200 AND 600;

-- exclude rows with amount between 300 and 500
SELECT item, amount
FROM Orders
WHERE amount NOT BETWEEN 300 AND 500;


-- select rows with NULL email values
```

```
SELECT *
FROM Employee
WHERE email IS NULL;

SELECT MAX(age)
FROM Customers;

SELECT MIN(age)
FROM Customers;

-- returns the number of rows in the Orders table
SELECT COUNT(*)
FROM Orders;

-- count of customers who live in the UK
SELECT COUNT(country) AS customers_in_UK
FROM Customers
WHERE country = 'UK';

--select the sum of amount from Orders table
SELECT SUM(amount) AS total_sales
FROM Orders;

-- get average age of customers
SELECT AVG(age) AS average_age
FROM Customers;

-- orders all rows from Customers in ascending order by country
SELECT *
FROM Customers
ORDER BY country;
```

```
-- select last_name and age of customers who don't live in the UK
-- and sort them by last_name in descending order
SELECT last_name, age
FROM Customers
WHERE NOT country = 'UK'
ORDER BY last_name DESC;

-- select the item column and the count of order ids from the Orders table
-- group them by the item column
SELECT COUNT(order_id), item
FROM Orders
GROUP BY item;

-- join the Customers and Orders tables
-- select customer_id and first_name from Customers table
-- also select the count of order ids from Orders table
-- group the result by customer_id
SELECT Customers.customer_id, Customers.first_name,
COUNT(Orders.order_id) AS order_count
FROM Customers
LEFT JOIN Orders
ON Customers.customer_id = Orders.customer_id
GROUP BY Customers.customer_id;

-- select customers who live in the UK
SELECT *
FROM Customers
WHERE country LIKE 'UK';
```

```sql
-- select customers whose last_name starts with R and ends with t
-- or customers whose last_name ends with e
SELECT *
FROM Customers
WHERE last_name LIKE 'R%t' OR last_name LIKE '%e';

-- select the union of name columns from two tables Teachers and Students
SELECT name
FROM Teachers
UNION
SELECT name
FROM Students;

-- add a new column named 'Priority' in the output
-- and store 'Huge Order' where amount is greater than or equal to 10000
SELECT order_id, item, amount,
CASE
  WHEN amount >= 10000 THEN 'Huge Order'
END AS Priority
FROM Orders;

-- select customers with the same first name based on their age count
SELECT COUNT(age) AS Count, first_name
FROM Customers
GROUP BY first_name
HAVING COUNT(age) > 1;

-- select customer id and first name of customers
-- whose order amount is less than 12000
SELECT customer_id, first_name
FROM Customers
```

```
WHERE EXISTS (
  SELECT order_id
  FROM Orders
  WHERE Orders.customer_id = Customers.customer_id AND amount < 12000
);
```

The SQL `JOIN` joins two tables based on a common column and selects records that have matching values in these columns.

```
-- join the Customers and Orders tables
-- based on the common values of their customer_id columns
SELECT Customers.customer_id, Customers.first_name, Orders.item
FROM Customers
JOIN Orders
ON Customers.customer_id = Orders.customer_id;


-- use alias C for Customers table
-- use alias O for Orders table
SELECT C.customer_id, C.first_name, O.amount
FROM Customers AS C
JOIN Orders AS O
ON C.customer_id = O.customer;
```

The SQL `INNER JOIN` command joins two tables based on a common column and selects rows that have matching values in these columns.

```
-- join Customers and Orders tables with their matching fields customer_id
SELECT Customers.customer_id, Orders.item
FROM Customers
INNER JOIN Orders
ON Customers.customer_id = Orders.customer;
```

The **LEFT JOIN keyword returns all records from the left table (table1),** and the **matching records from the right table (table2)**

-- left join Customers and Orders tables based on their shared customer_id columns

-- Customers is the left table

-- Orders is the right table

SELECT Customers.customer_id, Customers.first_name, Orders.item

FROM Customers

LEFT JOIN Orders

ON Customers.customer_id = Orders.customer_id;

The RIGHT JOIN keyword returns all records from the right table (table2), and thematching records from the left table (table1).

```
-- join Customers and Orders tables
-- based on their shared customer_id columns
-- Customers is the left table
-- Orders is the right table

SELECT Customers.customer_id, Customers.first_name, Orders.item
FROM Customers
RIGHT JOIN Orders
ON Customers.customer_id = Orders.customer_id;
```

```
CREATE DATABASE my_db;
```

-- create a table Companies with name, id, address, email, and phone number

CREATE TABLE Companies (

id int,

name varchar(50),

address text,

email varchar(50),

phone varchar(10)

```
);

-- insert a row in the Customers table
INSERT INTO Customers(customer_id, first_name, last_name, age, country)
VALUES
(7, 'Ron', 'Weasley', 31, 'UK');

-- update a single value in the given row
UPDATE Customers
SET age = 21
WHERE customer_id = 1;
```

The **`MERGE`** statement in SQL is used to perform an "upsert" operation, which is a combination of **`INSERT`**, **`UPDATE`**, and sometimes **`DELETE`** actions based on specified conditions

```
MERGE INTO employees AS target
USING employee_updates AS source
ON target.employee_id = source.employee_id
WHEN MATCHED THEN
   UPDATE SET target.first_name = source.new_first_name, target.last_name = source.new_last_name
WHEN NOT MATCHED THEN
   INSERT (employee_id, first_name, last_name)
   VALUES (source.employee_id, source.new_first_name, source.new_last_name);
```

- **`JOIN`** is used to combine rows from different tables based on related columns.
- **`UNION`** is used to combine the result sets of two or more **`SELECT`** queries, removing duplicates by default