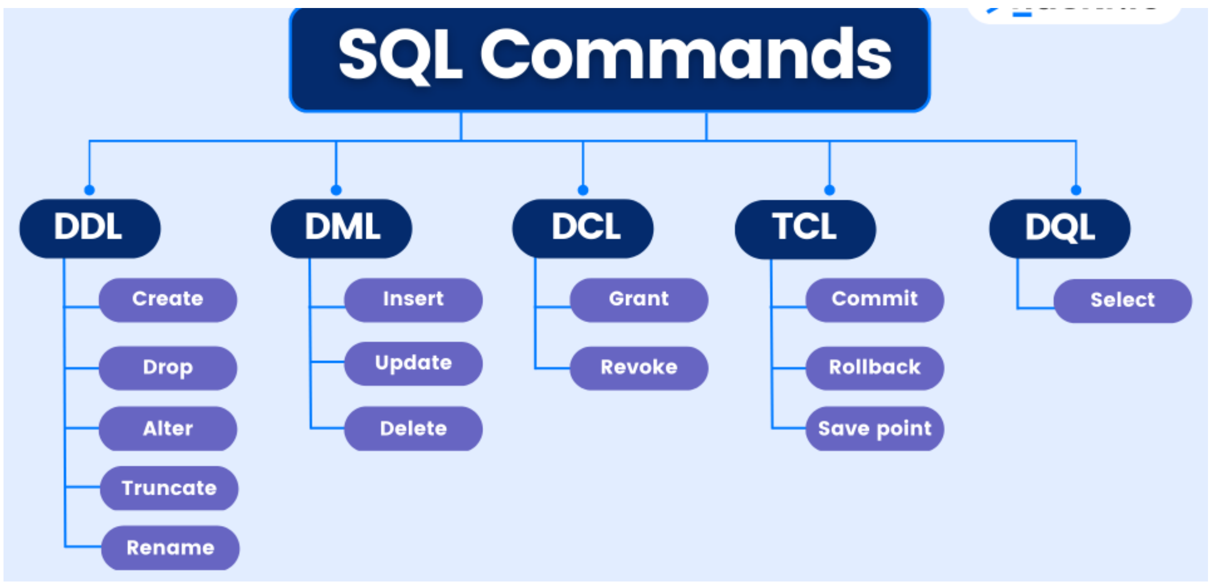
DBMS

Database

A Database Management System (DBMS) is a software system that is designed to manage and organize data in a structured manner. It allows users to create, modify, and query a database, as well as manage the security and access controls for that database.

Structure Query Language



DDL

CREATE DATABASE — creates a new database

CREATE DATABASE databasename;

CREATE DATABASE statements create a new SQL database.

CREATE TABLE — creates a new table

```
CREATE TABLE table_name (
column_1 datatype,
column_2 datatype,
column_3 datatype
);
```

CREATE TABLE statements create a new table in the database.

DROP TABLE — deletes a table

```
DROP TABLE table_name;
```

DROP TABLE statements drop an existing table in a database.

ALTER TABLE — modifies a table

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

ALTER TABLE statements add, delete, or modify columns in an existing table.

ALTER DATABASE — modifies a database

```
ALTER DATABASE database_name
[COLLATE collation_name ]
```

ALTER DATABASE statements change the characteristics of a database.

TRUNCATE

A truncate SQL statement is used to remove all rows (complete data) from a table. It is similar to the DELETE statement with no WHERE clause.

```
TRUNCATE TABLE table_name;
```

RENAME

```
RENAME old_table _name To new_table_name ;
```

CREATE INDEX — creates an index

```
CREATE INDEX index_name
ON table_name (column_name1, column_name2…);
```

Index statements create on existing tables to retrieve the rows quickly.

DROP INDEX — deletes an index

```
ALTER TABLE table_name
DROP INDEX index_name;
```

DROP INDEX statements delete an index in a table.

DML

INSERT INTO — inserts new data into a database

INSERT INTO table_name (column_1, column_2, column_3)

VALUES (value_1, 'value_2', value_3);

INSERT statements add a new row to a table

UPDATE — updates data in a database

UPDATE table_name

SET some_column = some_value

WHERE some_column = some_value;

UPDATE statements allow us to edit rows in a table.

DELETE — deletes data from a database

DELETE FROM table_name

WHERE some_column = some_value;

DELETE statements remove rows from a table.

DCL

GRANT: This command gives users access privileges to the database.

GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;

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

REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;

TCL

BEGIN: Opens a Transaction.

COMMIT: Commits a Transaction.

COMMIT;

ROLLBACK: Rollbacks a transaction in case of any error occurs.

ROLLBACK;

SAVEPOINT: Sets a save point within a transaction.

SAVEPOINT SAVEPOINT_NAME;

DRL

SELECT — extracts data from a database

SELECT column_name

FROM table_name;

SELECT statements fetch data from a database.

WHERE: This MySQL Query command filters the data for a specific value.

SELECT * FROM [TABLE NAME] WHERE [CONDITION];

Example:

SELECT * FROM EMPLOYEE WHERE EMP_ID=200;

AND: This condition filters the data based on conditions.

SELECT [COLUMN NAMES] FROM [TABLE NAME] WHERE [CONDITION] AND [CONDITON];

Example:

SELECT EMP_NAME, FROM EMPLOYEE WHERE EMP_ID=200 AND EMP_COUNTRY="INDIA";

OR: This MySQL Query Command combines the data from the table for the specific condition.

SELECT [COLUMN NAMES] FROM [TABLE NAME] WHERE TRUE OR FALSE

Example:

SELECT * FROM EMPLOYEE WHERE EMP_COUNTRY="INDIA" OR EMP_COUNTRY ="USA";

IN: This operator helps filter the data based on a value match.

SELECT COLUMN1, COLUMN2… FROM [TABLE NAME] WHERE [COLUMN NAME] IN ('val1','val2');

Example:

SELECT EMP_NAME, EMP_SALARY FROM EMPLOYEE WHERE EMP_COUNTRY IN ('INDIA','USA', 'NZ');

ORDER BY: It is used to sort the data in a particular order for a specific column in ascending or descending order.

SELECT COLUMN1, COLUMN2, FROM [TABLE NAME] ORDER BY Column1 desc, Column2 asc;

Example:

SELECT EMP_NAME, EMP_ID FROM EMPLOYEE ORDER BY EMP_NAME desc, EMP_ID asc;

GROUP BY: This is used to get the data for the particular value in the combined form.

SELECT Column1, Column2 FROM TABLE WHERE CONDITION Group by Col2;

HAVING

The HAVING clause is used instead of WHERE with aggregate functions. While the GROUP BY Clause groups rows that have the same values into summary rows. The having clause is used with the where clause in order to find rows with certain conditions.

Syntax

SELECT column1,column2,columnn

FROM table_name

GROUP BY column_name

HAVING aggregate_function(column_name) condition;

AGGREGATE FUNCTION

- **COUNT** counts how many rows are in a particular column.

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

- **SUM** adds together all the values in a particular column.

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

- **MIN** and **MAX** return the lowest and highest values in a particular column, respectively.

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

- **AVG** calculates the average of a group of selected values.

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

- **DISTINCT**

The SELECT DISTINCT statement is used to return only distinct (different) values.

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

Joins

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

- (INNER) JOIN: Returns records that have matching values in both tables

  ```
  SELECT Customers.customer_id, Orders.item
  FROM Customers
  INNER JOIN Orders
  ON Customers.customer_id = Orders.customer;
  ```

- LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table

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

- RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table

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

- FULL (OUTER) JOIN: Return all records when there is a match in either left or right table

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