SQL

- SQL is a short-form of the structured query language
- The basic use of SQL for data professionals and SQL users is to insert, update, and delete the data from the relational database.
- SQL allows the data professionals and users to retrieve the data from the relational database management systems.
- It also helps them to describe the structured data.
- It allows SQL users to create, drop, and manipulate the database and its tables.
- It also helps in creating the view, stored procedure, and functions in the relational database.
- It allows you to define the data and modify that stored data in the relational database.
- It also allows SQL users to set the permissions or constraints on table columns, views, and stored procedures.

Overview of DBMS:

- A Database Management System (DBMS) is software that facilitates the creation, organization, and management of databases.
- It serves as an interface between the database and the users or application programs, ensuring efficient and secure storage, retrieval, and manipulation of data.
- Here's a brief overview of key components and functionalities of a DBMS:
    - Data Definition Language (DDL)
    - Data Manipulation Language (DML)
    - Data Integrity and Constraints
    - Transaction Management
    - Concurrency Control
    - Security and Authorization
    - Backup and Recovery
    - Query Optimization

Overview of Workbench:

- MySQL Workbench is a visual tool designed to help users interact with MySQL databases.
- It provides a graphical interface where you can design, model, and administer databases.

- MySQL Workbench simplifies tasks such as creating and managing database structures, writing and executing SQL queries, and visualizing the relationships between different database components.
- It's especially useful for database administrators, developers, and data architects who prefer a visual approach to working with MySQL databases, offering a user-friendly environment to streamline database-related tasks.

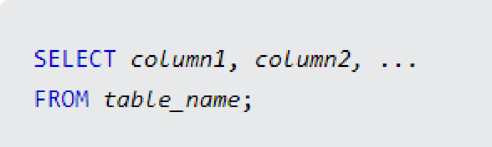
Database Design:

- Database design in MySQL involves planning and structuring a database to efficiently store and manage data.
- It includes defining tables, specifying relationships between tables, and establishing constraints to maintain data integrity.
- The process often starts with identifying the data requirements and organizing them into tables, deciding on appropriate data types, and establishing relationships between tables using primary and foreign keys.
- MySQL Workbench, a visual tool, can aid in designing and visualizing the database schema.
- Effective database design in MySQL contributes to better performance, data accuracy, and overall system efficiency.

Database Tables:

- A database most often contains one or more tables. Each table is identified by a name (e.g. "Customers" or "Orders"), and contains records (rows) with data.

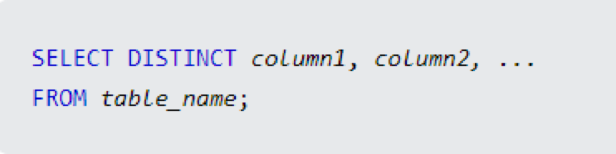
SQL SELECT:



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

- column1, column2, ... are the field names of the table you want to select data from.
- The table_name represents the name of the table you want to select data from.
- If you want to return all columns, without specifying every column name, you can use the SELECT * syntax

SELECT DISTINCT:

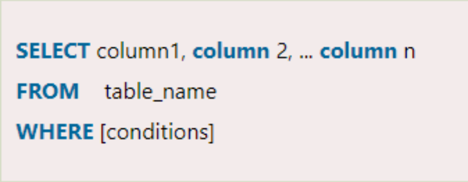


```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

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

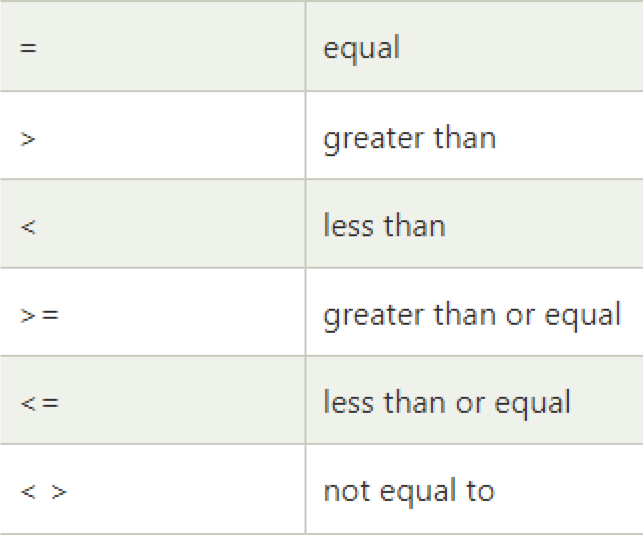
SQL WHERE Clause:

- WHERE clauses are not mandatory clauses of SQL DML statements. But it can be used to limit the number of rows affected by a SQL DML statement or returned by a query.
- It filters the records. It returns only those queries which fulfill the specific conditions.
- WHERE clause is used in SELECT, UPDATE, DELETE statement etc.



```
SELECT column1, column 2, ... column n
FROM    table_name
WHERE [conditions]
```

- WHERE clause uses some conditional selection

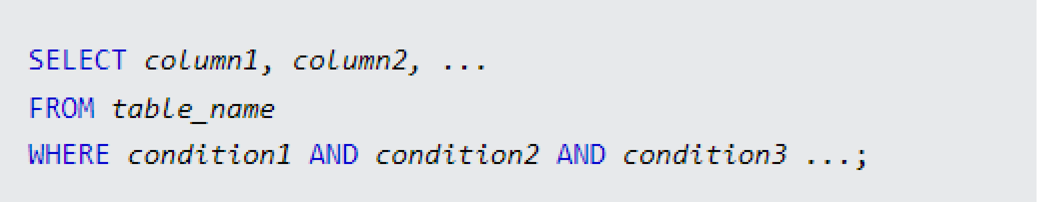


=	equal
>	greater than
<	less than
>=	greater than or equal
<=	less than or equal
< >	not equal to

SQL AND Operator:

- The SQL AND condition is used in SQL query to create two or more conditions to be met.
- It is used in SQL SELECT, INSERT, UPDATE and DELETE
- SELECT columns FROM tables WHERE condition 1 AND condition 2;

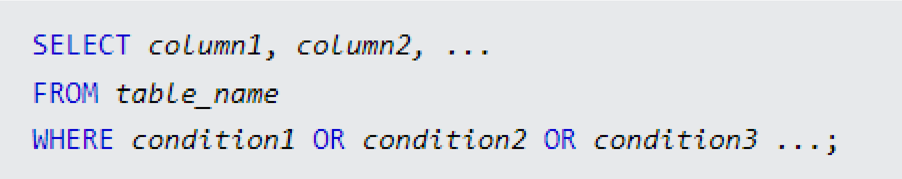
- The SQL AND condition require that both conditions should be met.



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

SQL OR Operator:

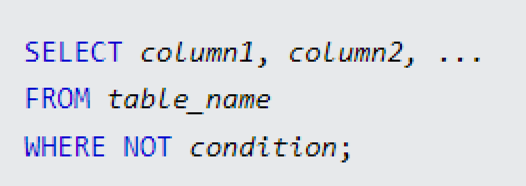
- The WHERE clause can contain one or more OR operators.
- The OR operator is used to filter records based on more than one condition.
- It can be used in a SELECT statement, INSERT statement, UPDATE statement or DELETE statement.



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

NOT Operator:

- The NOT operator is used in combination with other operators to give the opposite result, also called the negative result.

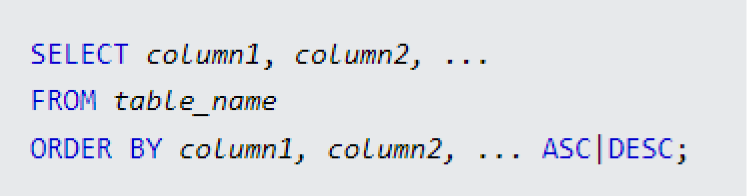


```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

- the NOT operator is used in combination with the = operator, but it can be used in combination with other comparison and/or logical operators.

SQL ORDER BY:

- The ORDER BY keyword is used to sort the result-set in ascending or descending order.

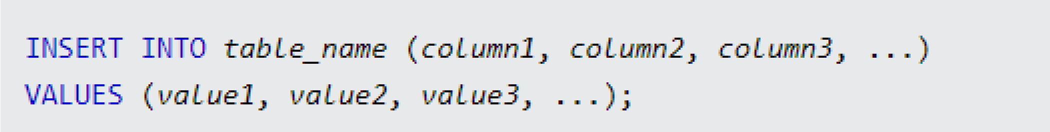


```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

- The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.
- For string values the ORDER BY keyword will order alphabetically.
- To sort the table reverse alphabetically, use the DESC keyword

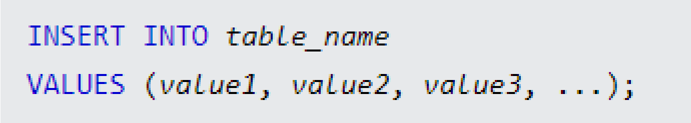
SQL INSERT INTO:

- The INSERT INTO statement is used to insert new records in a table.
- It is possible to write the INSERT INTO statement in two ways
    - Specify both the column names and the values to be inserted:



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

    - If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the INSERT INTO syntax would be as follows:



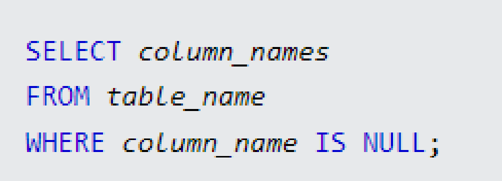
```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

- The ID column is an auto-increment field and will be generated automatically when a new record is inserted into the table.
- It is also possible to insert multiple rows in one statement.
- To insert multiple rows of data, we use the same INSERT INTO statement, but with multiple values.

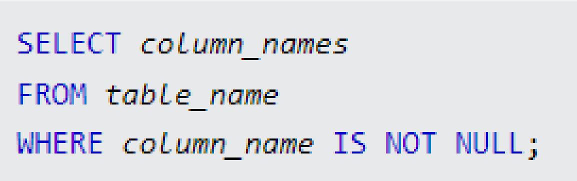
NULL Value:

- A field with a NULL value is a field with no value.
- If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.
- A NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation.
- It is not possible to test for NULL values with comparison operators, such as =, <, or <>.

- We will have to use the IS NULL and IS NOT NULL operators instead.



```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
```



```
SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL;
```

DDL

- DDL stands for Data Definition Language.
- DDL statements are used to define, manage, and manipulate the structure of database objects.

- These statements are responsible for creating, altering, and deleting database objects such as tables, indexes, views, and schemas. Common DDL statements include CREATE, ALTER, and DROP.

- **CREATE**:
  - Used to create new database objects, such as tables, indexes, views, or schemas.

    ```
    CREATE TABLE EMP01(
            P01F01 int comment "Id" primary key auto_increment,
            P01F02 varchar(50) comment "Name",
            P01F03 int comment "Age",
            P01F04 varchar(30) comment "Designation”
    );
    ```

- **ALTER**:
  - Used to modify the structure of existing database objects, such as adding or dropping columns in a table.

    ```
    ALTER TABLE
            EMP01
    ADD COLUMN
            P01F05 INT COMMENT "Salary";
    ```

- **DROP**:
  - Used to delete database objects, such as tables, views, or indexes.

    ```
    DROP TABLE
            EMP02;
    ```

- **TRUNCATE**:
  - Used to remove all records from a table but retain the table structure for future use.

    ```
    TRUNCATE TABLE
            EMP02;
    ```

- **RENAME**:
  - Used to rename an existing database object.

```
ALTER TABLE
        EMP02
RENAME TO
        EMP01;
```

DML:

- DML stands for Data Manipulation Language.
- DML statements are used to manipulate and interact with data stored in the database.
- DML statements deal with the actual data within those objects. Common DML statements include SELECT, INSERT, UPDATE, and DELETE.

- **SELECT**:
  - Used to retrieve data from one or more tables. It forms the basis for querying and retrieving information from the database

```
SELECT
        P01F01,
        P01F02,
        P01F03,
        P01F04,
        P01F05
FROM
        EMP01
WHERE
        P01F03 > 21;
```

- **INSERT**:
  - Used to add new records into a table.

```
INSERT INTO EMP01(
        P01F02,
        P01F03,
        P01F04,
        P01F05
) VALUES(
        "Kishan",
        26,
        "Engineer",
        50000
);
```

- **UPDATE**:
    - Used to modify existing records in a table.

      ```
      UPDATE
              EMP01
      SET
              P01F05 = 10000
      WHERE
              P01F01 = 1;
      ```

- **DELETE**:
    - Used to remove records from a table.

      ```
      DELETE FROM
              EMP01
      WHERE P01F01 = 6;
      ```

DCL:

- DCL stands for Data Control Language in SQL.
- DCL statements are used to control access to data within a database.
- These statements are primarily concerned with defining and managing user permissions and privileges.
- DCL statements are typically executed by database administrators or users with administrative privileges.
- There are two main DCL statements: GRANT and REVOKE.

- **GRANT**:
    - Used to give specific privileges to a user or role. Privileges may include the ability to SELECT, INSERT, UPDATE, DELETE, or execute other DDL and DML operations.

      ```
      GRANT
              SELECT,
              UPDATE,
      ```

```
        DELETE,
        INSERT
ON
        employee.*
TO
        'dev@localhost';
```

- **REVOKE**:
  - Used to take away specific privileges that have been previously granted to a user or role.

    ```
    REVOKE
            UPDATE,
            DELETE,
            INSERT
    ON
            employee.*
    FROM
            'dev@localhost';
    ```

TCL:

- TCL stands for Transaction Control Language in SQL.
- TCL statements are used to manage transactions in a database.
- Transactions are sequences of one or more SQL statements that are executed as a single unit of work.
- TCL statements help ensure the integrity of the database by allowing you to control the beginning and ending of transactions and manage the changes made during those transactions.
- The primary TCL statements are COMMIT, ROLLBACK, and SAVEPOINT.

- **COMMIT:**
  - Used to save all the changes made during the current transaction. Once a COMMIT statement is executed, the changes become permanent.

    ```
    INSERT INTO
            EMP03 (
            P03F01,
            P03F02
    ) VALUES (
            1,
    ```

```
   'Transaction Demo'
);

COMMIT;
```

- **ROLLBACK**:
  - Used to undo the changes made during the current transaction.
  - If there is an error or if the user decides to discard the changes, a ROLLBACK statement is executed to revert the database to its state before the transaction begins.

```
INSERT INTO
        EMP03 (
        P03F01,
        P03F02
) VALUES (
        3,
   'after Transaction Demo'
);

ROLLBACK;
```

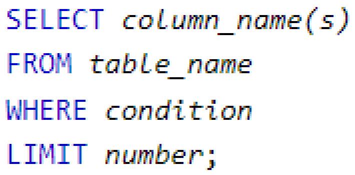
- **SAVEPOINT:**
  - Used to set a point within a transaction to which you can later roll back. It allows for partial rollback within a transaction.

```
BEGIN;
-- Inserting data into the table (after the transaction)
INSERT INTO
        EMP03 (
        P03F01,
        P03F02
) VALUES (
        4,
   'before save point Transaction Demo'
);

SAVEPOINT SVE_PT;

INSERT INTO
```

```
    EMP03 (
        P03F01,
        P03F02
) VALUES (
        5,
   'after save point Transaction Demo'
);

ROLLBACK TO SVE_PT;
```

SQL LIMIT Clause:

- The LIMIT clause is used to specify the number of records to return.
- The LIMIT clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.



```
SELECT column_name(s)
FROM table_name
WHERE condition
LIMIT number;
```

Aggregate functions:

- Aggregate functions in SQL perform a calculation on a set of values and return a single value.
- These functions are often used with the GROUP BY clause to perform operations on groups of rows.

- **COUNT**:
    - Counts the number of rows in a result set.

```
SELECT
        COUNT(DISTINCT P01F03) AS "Total Count"
FROM
        EMP01
WHERE
        P01F03 > 21;
```

- **SUM**:
    - Calculates the sum of values in a numeric column.

```
SELECT
```

```
        SUM(P01F05) AS "Sum"
FROM
        EMP01;

SELECT
        P01F04 AS "Designation",
        SUM(P01F05) AS "Total Salary"
FROM
        EMP01
GROUP BY
        P01F04
HAVING
        SUM(P01F05) > 35000;
```

- **AVG**:
  - Calculates the average of values in a numeric column.

```
SELECT
        AVG(P01F05)
AS "Average Salary"
FROM
        EMP01;
```

- **MIN**:
  - Finds the minimum value in a column.

```
SELECT
        MIN(P01F05)
AS "Minimum Salary"
FROM
        EMP01;
```

- **MAX**:
  - Finds the maximum value in a column.

```
SELECT
        MAX(P01F05)
AS "maximum Salary"
FROM
        EMP01;
```

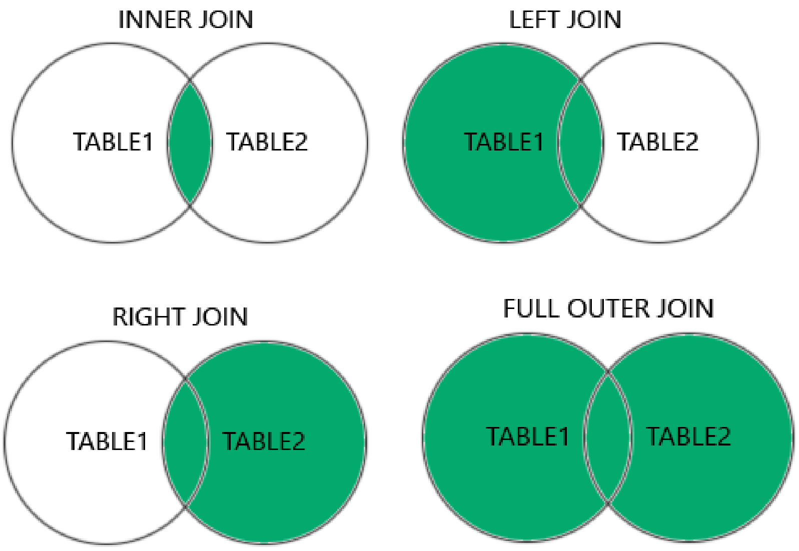
Sub Queries:

- Subqueries, also known as nested queries, are queries embedded within another query.
- They are enclosed within parentheses and can be used in various parts of a SQL statement, including the SELECT, FROM, WHERE, and HAVING clauses.
- Subqueries can be used to retrieve data that will be used by the main query, perform comparisons, or filter results based on the outcome of the subquery.
- Subqueries can be useful for breaking down complex queries into more manageable and modular components.
- They allow you to perform operations on a set of data before using that result in the main query.

```
SELECT
        P01F01,
        P01F02,
        P01F03,
        P01F04,
        P01F05
FROM
        EMP01
WHERE
        P01F05 >= (
SELECT AVG(
        P01F05
) FROM
        EMP01
);
```

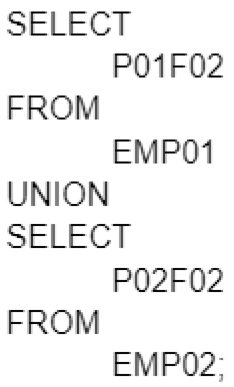
Joins:

- In SQL, joins are used to combine rows from two or more tables based on a related column between them.
- Joins allow you to retrieve data from multiple tables in a single query, facilitating the retrieval of related information.
- There are several types of joins in SQL, including INNER JOIN, LEFT JOIN (or LEFT OUTER JOIN), RIGHT JOIN (or RIGHT OUTER JOIN), and FULL JOIN (or FULL OUTER JOIN).

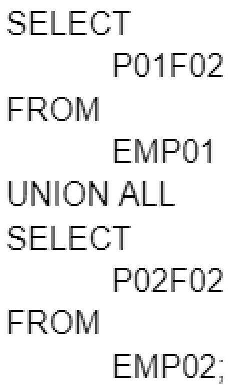


Union;

- The UNION operator is used to combine the results of two or more SELECT statements into a single result set.
- It removes duplicate rows from the combined result set, providing a distinct set of records.
- If you want to include duplicate rows in the result set, you can use the UNION ALL operator instead of UNION.



```
SELECT
        P01F02
FROM
        EMP01
UNION
SELECT
        P02F02
FROM
        EMP02;
```



```
SELECT
        P01F02
FROM
        EMP01
UNION ALL
SELECT
        P02F02
FROM
        EMP02;
```

INDEX:

- An index is a database object that provides a quick and efficient way to look up data in a table based on the values in one or more columns.

- Indexes improve the speed of data retrieval operations by allowing the database engine to locate and access the rows more efficiently.

```
CREATE INDEX
        idx_P01F04
ON
        EMP01(P01F04);


CREATE INDEX
        idx_P01_F01_F04
ON
        EMP01(P01F01,P01F04);
```

VIEW:

- A view is a virtual table based on the result of a SELECT query.
- Views allow you to encapsulate complex queries and present the result as if it were a table.
- They provide a way to simplify queries for users and can be used to control access to the underlying tables by exposing only specific columns or rows.

```
CREATE VIEW
    VIEW_EMP01
AS SELECT
    P01F01,
    P01F02,
    P01F04
FROM
    EMP01;


# view with joins
UPDATE
    VIEW_EMP01
SET
    P01F02 = CONCAT("SUPER ", P01F02)
WHERE
    P01F02 LIKE "A%";


# display with view with joins
CREATE OR REPLACE VIEW
    VIEW_EMP02_DEP01
AS SELECT
    P02F01,
    P02F02,
     P01F02
FROM
    EMP02
LEFT JOIN
    DEP01
ON
    EMP02.P02F03 = DEP01.P01F01;
```

Backup:

- The BACKUP statement is used to create backups of databases.

Restore:

- The RESTORE statement is used to restore a database from a backup.

Explain:

- The EXPLAIN statement helps analyze how the database engine plans to execute a query and is often used for query optimization.

```
EXPLAIN SELECT
        P01F01,
        P01F02,
        P01F03,
        P01F04,
         P01F05
FROM
        EMP01
WHERE
        P01F04 = "SDE";


EXPLAIN ANALYZE SELECT
        P01F01,
        P01F02,
        P01F03,
        P01F04,
         P01F05
FROM
        EMP01
WHERE
        P01F04 = "SDE";
```