# Introduction to SQL

Structured Query Language is a standard Database language that is used to create, maintain, and retrieve the relational database.

SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

SQL is case insensitive. But it is a recommended practice to use keywords (like SELECT, UPDATE, CREATE, etc.) in capital letters and use user-defined things (like table name, column name, etc.) in small letters.

SQL is the programming language for relational databases (explained below) like MySQL, Oracle, Sybase, SQL Server, Postgre, etc. Other non-relational databases (also called NOSQL) databases like MongoDB, DynamoDB, etc. do not use SQL.

## What is Relational Database?

- RDBMS stands for Relational Database Management System.
- RDBMS is a program used to maintain a relational database.
- RDBMS is the basis for all modern database systems such as MySQL, Microsoft SQL Server, Oracle, and Microsoft Access.
- RDBMS uses SQL queries to access the data in the database.

## What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases

- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database.

## SQL is a Standard - BUT....

Although SQL is an ANSI/ISO standard, there are different versions of the SQL language.

However, to be compliant with the ANSI standard, they all support at least the +

major commands (such as SELECT, UPDATE, DELETE, INSERT, WHERE) in a similar manner.

1) **Insert query:-**The INSERT INTO statement is used to insert new records in a table.

   **Syntax:-**

   INSERT INTO *table name* (*column1*, *column2*, *column3*, ...)
   VALUES (*value1*, *value2*, *value3*, ...);

   **For ex:-** INSERT INTO Students (stud_nm, stud_City, Country)
   VALUES ('Heer', 'Pune', 'India');

| Stud_id | Stud_nm | Stud_city | country |
|---|---|---|---|
| 1 | Heer | Pune | india |

   INSERT INTO *table name*
   VALUES (*value1*, *value2*, *value3*, ...);

   **For ex:-**

   INSERT INTO Customers VALUES ('Cardinal', 'Stavanger', 'Norway');

2) **Update query:-** The UPDATE statement is used to modify the existing records in a table.

   **Syntax:-**

```
UPDATE table name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

**For ex:-**

the following SQL statement updates the first customer (studentid =1 with a new stud_nm person *and* a new city.

```
UPDATE Student
SET stud_nm = 'helly', stud_City= 'pune'
WHERE  stud_id = 1;
```

| Stud_id | Stud_nm | Stud_city | country |
|---|---|---|---|
| 1 | Helly | pune | india |
| 2 | Charles | surat | sauth |
| 3 | xyz | norway | india |

**UPDATE Multiple Records:-**

It is the WHERE clause that determines how many records will be updated.

The following SQL statement will update the ContactName to "Juan" for all records where country is "Mexico":

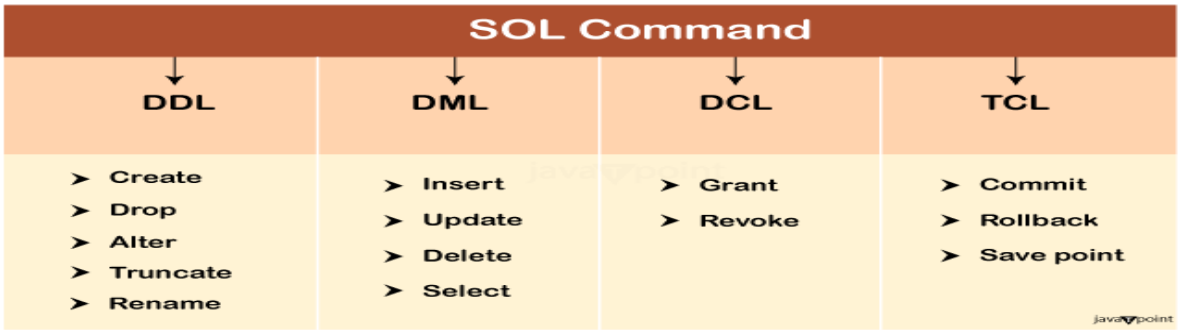
```
UPDATE Customers
SET stud_nm='heer'
WHERE Country='india';
```

| Stud_id | Stud_nm | Stud_city | country |
|---|---|---|---|
| 1 | Heer | pune | india |
| 2 | Charles | surat | sauth |
| 3 | Heer | norway | india |

3) **Select query:-** The `SELECT` statement is used to select data from a database.
**Syntax:-**

```
SELECT * FROM table_name;
```

# Sql commands:-




**DDL:-** **Data Definition Language**

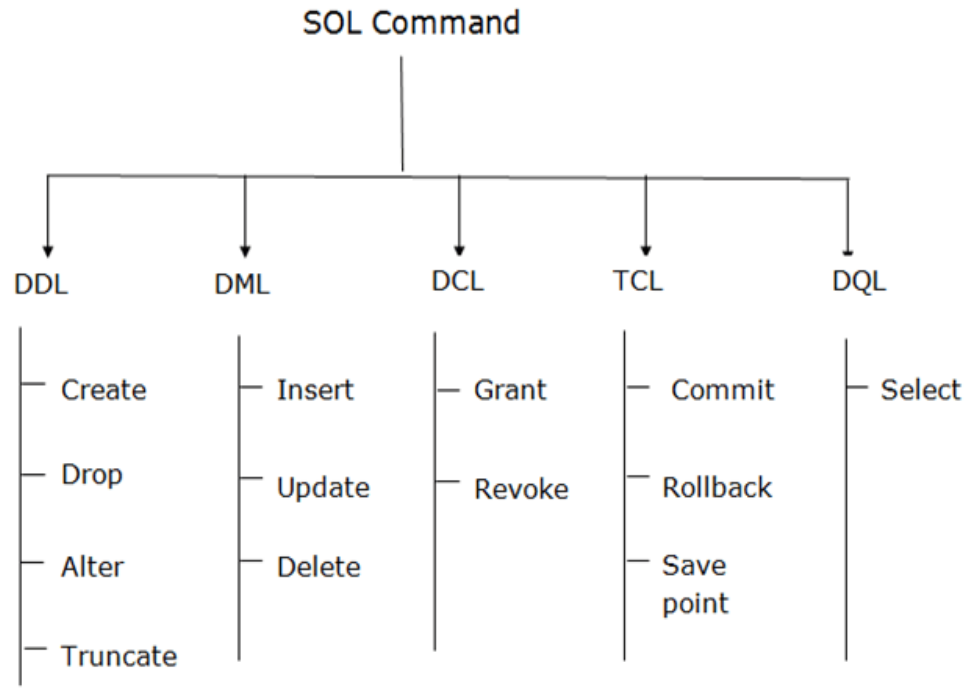
**DML:-** **Data Manipulation Language**

**DCL:-** **Data Control Language**

**TCL:-** **Transaction Control Language**

# SQL Commands And Data types

- SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.
- SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.
- **SQL commands** are very used to interact with the database.
- These commands allow users to perform various actions on a database. This article will teach us about **SQL commands** or **SQL sublanguage commands** like **DDL**, **DQL**, **DML**, **DCL,** and **TCL.**




## 1)Data Definition Language (DDL)

DDL Stand for **Data Definition Language**.

- DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
- All the command of DDL are auto-committed that means it permanently save all the changes in the database.

## 1) Create Command:-

CREATE is a DDL command used to create databases, tables, triggers and other database objects.

**Syntax to Create a Database:**

```
CREATE Database Database_Name;
```

**Ex:-**

```
Create Database Books;
```

**Syntax to create a new table:**

```
CREATE TABLE table_name
```

```
(
  column_Name1 data_type ( size of the column ) ,
  column_Name2 data_type ( size of the column) ,
  column_Name3 data_type ( size of the column) ,
    ...
  column_NameN data_type ( size of the column )
) ;
```

**Ex:-**

```
CREATE TABLE Student
    (
       Roll_No. Int ,
      First_Name Varchar (20) ,
      Last_Name Varchar (20) ,
      Age Int ,
     Marks Int ,
   ) ;
```

## 2) Drop Command:-

DROP is a DDL command used to delete/remove the database objects from the SQL database

We can easily remove the entire table, view, or index from the database using this DDL command.

### Syntax to remove a database:

```
DROP DATABASE Database_Name;
```

**Ex:-**

```
DROP DATABASE Books;
```

### Syntax to remove a table:

```
DROP TABLE Table_Name;
```

**Ex:**

```
DROP TABLE Student;
```

## 3) Alter Command:-

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

**ALTER TABLE - ADD Column**:-

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

Ex:-

```
ALTER TABLE Customers
ADD Email varchar(255);
```

**ALTER TABLE - DROP COLUMN**

```
ALTER TABLE name_of_table DROP Column_Name_1 , column_Name_2 ,
    ….., column_Name_N;
```

Ex:-

```
ALTER TABLE Student DROP Age, Marks;
```

## 4) Truncate Command:-

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.

Once the TRUNCATE command is used, you can't recover the data even using ROLLBACK. It is similar to the DELETE command in SQL without a WHERE clause.

**Syntax:- TRUNCATE TABLE table_name;**

```
Ex:-
CREATE TABLE Student(
   RollNo int PRIMARY KEY,
   Name VARCHAR(100) NOT NULL,
   Gender TEXT NOT NULL);

INSERT INTO Student VALUES (1, Vaibhav, M,);
INSERT INTO Student VALUES (2, Vishal, M);
INSERT INTO Student VALUES (3, Saumya, F);
```

```
SELECT * FROM Student;
```

| RollNo | Name | Gender |
|---|---|---|
| 1 | Vaibhav | M |
| 2 | Vishal | M |
| 3 | Saumya | F |

Use the TRUNCATE TABLE command in the above student table to remove all the records.

**Query :-**

```
TRUNCATE TABLE Student;
SELECT * FROM Student;
```

**Output:-**

No Data Found

**TRUNCATE TABLE command over a partition**

Remove the record of columns 1 and 3 TO 5.

**Query:-**

```
TRUNCATE TABLE Student
WITH PARTITIONS (1, 3);
```

**Output:-**

| RollNo | Name | Gender |
|---|---|---|
| NULL | Vaibhav | NULL |
| NULL | Vishal | NULL |
| NULL | Saumya | NULL |

# Different between truncate and delete:-

| DELETE | TRUNCATE |
|---|---|
| DELETE is a DML Command | TRUNCATE is a DDL Command |
| WHERE Clause can be used to remove few rows from the table. | Where clause cannot be used along with TRUNCATE command. |
| DELETE command maintains transaction Log | TRUNCATE Command will not store transaction log |
| DELETE Command uses more system resources compared to TRUNCATE While executing the command | DELETE Command uses more system resources compared to TRUNCATE While executing the command |
| DELETE Command removes data from table row by row. | TRUNCATE Command will remove data from the table at a single shot. |
| Triggers will work with DELETE Command | Triggers will not work when truncate command is used. |
| DELETED data can be restored back. | TRUNCATEd data cannot be retored back |

## 3) Data Control Language (DCL):-

DCL commands are used to grant and take back authority from any database user.

It is used to provide different users access to the stored data.

DCL, DDL, DML, DQL, and TCL commands form the SQL (Structured Query Language).

- DCL commands are primarily used to implement access control on the data stored in the database. It is implemented along the DML (Data Manipulation Language) and DDL (Data Definition Language) commands.
- It has a simple syntax and is easiest to implement in a database.
- The administrator can implement DCL commands to add or remove database permissions on a specific user that uses the database when required.
- DCL commands are implemented to grant, revoke and deny permission to retrieve or modify the data in the database.

## Types of DCL Commands in SQL

Two types of DCL commands can be used by the user in SQL. These commands are useful, especially when several users access the database. It enables the administrator to manage access control. The two types of DCL commands are as follows:

- GRANT
- REVOKE

## What is privilege ?

- Privilege is a permission given by database.
- To create a own user
- It provides right to execute a particular type of sql statement.
- It also gives right to connect the database and create a table in your schema.

**NOTE:-** sql DCL command deals with privileges

## Privileges can be divided into two parts:-

1) System privileges(DDL)
2) Object privileges(DQL,DML)

## Sql provides two-DCL command to manipulate the privileges.

## 1)GRANT:-

it helps to provide any kind of access to any user.

SQL Grant command is specifically used to provide privileges to database objects for a user.

This command also allows users to grant permissions to other users too.

**Syntax:**

*GRANT privileges_names ON object TO user;*

## Parameters Used:-

- **privileges name**: These are the access rights or privileges granted to the user.
- **object:** It is the name of the database object to which permissions are being granted. In the case of granting privileges on a table, this would be the table name.
- **user:** It is the name of the user to whom the privileges would be granted.

**Privileges**: The privileges that can be granted to the users are listed below along with the description:

| Privilege | Description |
|---|---|
| SELECT | *select statement on tables* |
| INSERT | *insert statement on the table* |
| DELETE | *delete statement on the table* |
| INDEX | *Create an index on an existing table* |
| CREATE | *Create table statements* |
| ALTER | *Ability to perform ALTER TABLE to change the table definition* |
| DROP | *Drop table statements* |
| ALL | *Grant all permissions except GRANT OPTION* |
| UPDATE | *Update statements on the table* |
| GRANT | *Allows to grant the privilege that* |

*Ex:-*

```
GRANT SELECT, INSERT, DELETE ON mydb TO
'user1'@'localhost';
```

```
Grant insert,
Select on accounts to Ram
```

**2)REVOKE**:- it is used to take back the access from users.

The REVOKE command is used to remove system, database, table, and view privileges from designated AuthID(s).
Revoke command withdraw user privileges on database objects if any granted.
it does operations opposite to the Grant command.

**Syntax:-**

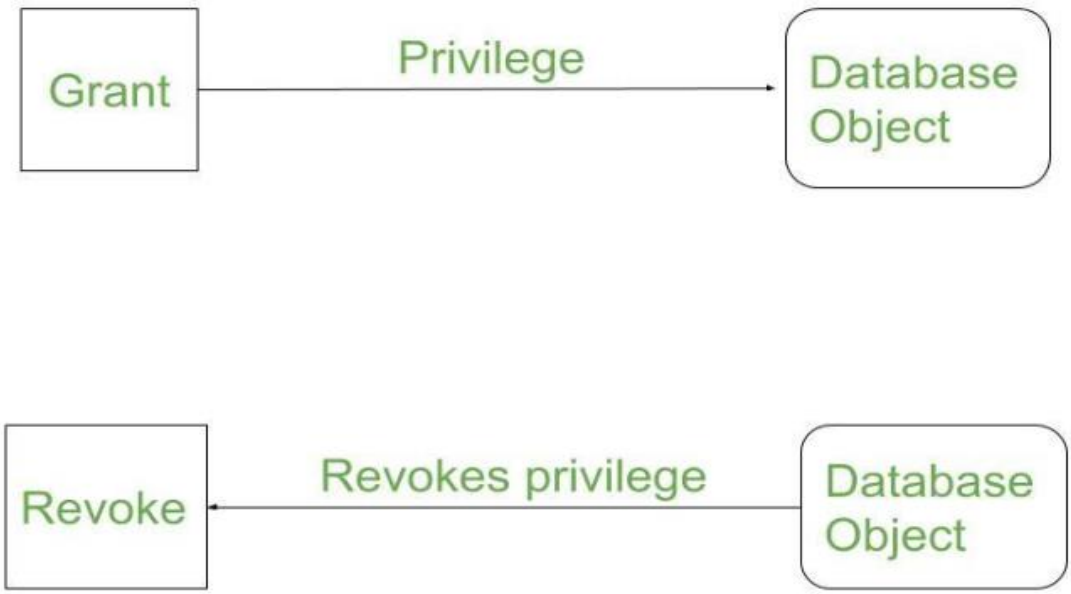
*REVOKE privileges_names ON object TO user;*

***Ex:-***

```
Grant insert,
Select on accounts to Ram
```

By the above command user ram has granted permissions on accounts database object like he can query or insert into accounts.

```
Revoke insert,
Select on accounts from Ram
```




Grant and Revoke Commands

| S.NO | Grant | Revoke |
|---|---|---|
| 1 | This DCL command grants permissions to the user on the database objects. | This DCL command removes permissions if any granted to the users on database objects. |
| 2 | It assigns access rights to users. | It revokes the useraccess rights of users. |
| 3 | For each user you need to specify the permissions. | If access for one user is removed; all the particular permissions provided by that users to others will be removed. |
| 4 | When the access is decentralized granting permissions will be easy. | If decentralized access removing the granted permissions is difficult. |

## 4)Transaction control  Language (TCL)

in SQL, TCL stands for **Transaction control language**.

TCL (Transaction Control Language) commands in SQL are used to manage transactions in a database.

Transactions are sequences of one or more SQL operations treated as a single unit.

A single unit of work in a database is formed after the sequential execution of commands is known as a transaction.

**COMMIT. ROLLBACK** and **SAVEPOINT** are the most commonly used TCL commands in SQL.

## 1. COMMIT

COMMIT command in SQL is used to save all the transaction-related changes permanently to the disk.

Whenever DDL commands such as INSERT, UPDATE and DELETE are used, the changes made by these commands are permanent only after closing the current session.

So before closing the session, one can easily roll back the changes made by the DDL commands. Hence, if we want the changes to be saved permanently to the disk without closing the session, we will use the commit command.

**Syntax**:-

```
COMMIT;
```

**Ex**:-

```
-- Start a transaction

START TRANSACTION;


-- Insert some data

INSERT INTO accounts (account_id, balance) VALUES (1, 1000);

INSERT INTO accounts (account_id, balance) VALUES (2, 2000);


-- Commit the transaction

COMMIT;
```

## 2)Rollback

ROLLBACK in SQL is a transactional control language that is used to undo the transactions that have not been saved in the database.

The command is only been used to undo changes since the last COMMIT.

### Syntax:-

```
ROLLBACK;
```

**Ex**:-

```
-- Start a transaction
   START TRANSACTION;

-- Update some records
UPDATE accounts SET balance = balance - 100 WHERE account_id = 1;
UPDATE accounts SET balance = balance + 100 WHERE account_id = 2;

-- Suppose an error occurs here
-- Rollback the transaction
ROLLBACK;

-- No changes will be saved
```

# Difference between COMMIT and ROLLBACK

| | COMMIT | ROLLBACK |
|---|---|---|
| **1.** | COMMIT permanently saves the changes made by the current transaction. | ROLLBACK undo the changes made by the current transaction. |
| **2.** | The transaction can not undo changes after COMMIT execution. | Transaction reaches its previous state after ROLLBACK. |
| **3.** | When the transaction is successful, COMMIT is applied. | When the transaction is aborted, incorrect execution, system failure ROLLBACK occurs. |
| **4.** | COMMIT statement permanently save the state, when all the statements are executed successfully without any error. | In ROLLBACK statement if any operations fail during the completion of a transaction, it cannot permanently save the change and we can undo them using this statement. |
| **5.** | **Syntax of COMMIT statement are:** COMMIT; | **Syntax of ROLLBACK statement are:** ROLLBACK; |

## 3) Savepoint :-

- Savepoint is a command in SQL that is used with the rollback command.
- It is a command in Transaction Control Language that is used to mark the transaction in a table.
- Consider you are making a very long table, and you want to roll back only to a certain position in a table then; this can be achieved using the savepoint.
- Savepoint is helpful when we want to roll back only a small part of a table and not the whole table. In simple words, we can say savepoint is a bookmark in SQL.

***For example***, we can save all the insert related queries with the savepoint named INS. To save all the insert related queries in one savepoint, we have to execute the SAVEPOINT query followed by the savepoint name after finishing the insert command execution.

**Syntax:**

SAVEPOINT savepoint_name;

# SQL Datatypes:-

## What is data type ?

A data type is a classification that specifies which type of value a variable can hold and what operations can be performed on it.

Data types specify the different sizes and values that can be stored in the variable. Examples include integers, floating-point numbers, characters, strings, and more complex types like arrays, lists, and objects.

Data types mainly classified into Four categories for every database.

- **Integer Data types**
- **Character String Data types**
- **Date and time Data types**
- **Numeric Data types**
- **Boolean Data type**
- **Binary Data Type**

## 1.Integer Data Types

- INT (Integer): Whole numbers, e.g., 1, 2, 3, etc.
- SMALLINT (Small Integer): Smaller range of whole numbers, e.g., -32,768 to 32,767.
- BIGINT (Big Integer): Larger range of whole numbers, e.g., -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.

## 2. Character Data Types

**- CHAR (Character):**

The char data type (short for "character") is used in many programming languages to represent a single character.

Fixed-length character string, e.g., 'hello' (5 characters).

This means that a CHAR column will always occupy the specified number of characters, padding with spaces if necessary.

**Characteristics :-**

- **Fixed-Length**: If the string is shorter than the specified length, it will be padded with spaces.
- **Storage**: It uses the same amount of storage regardless of the actual length of the string.

  **Ex**:-

  ```
  CREATE TABLE employees (
      employee_id INT PRIMARY KEY,
      first_name CHAR(20),
      last_name CHAR(20));
  ```

- **VARCHAR (Variable Character):**

In SQL, the VARCHAR (variable character) data type is used to store variable-length character strings.

Unlike CHAR, which always uses a fixed amount of storage, VARCHAR only uses as much storage as necessary to store the actual string length, plus a small amount of additional storage for length metadata.

Variable-length character string, e.g., 'hello' (5 characters) or 'hello world' (11 characters).

**Characteristics :-**

- **Variable-Length**: Stores only the actual string length plus a few bytes for length metadata.
- **Storage**: More efficient than CHAR for varying-length strings, as it does not pad with spaces.
- **Max Length**: Typically can be set to a maximum length up to 65,535 characters, depending on the database system.

**Ex**:-

```
CREATE TABLE customers (
    customer_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    email VARCHAR(100)
);
```

- **TEXT (Text):**

In SQL, the TEXT data type is used to store large amounts of text data.

Unlike CHAR and VARCHAR, which have size limitations, the TEXT data type can hold much larger strings, making it suitable for storing long descriptions, articles, or any other sizable text data.

Large text strings, e.g., a paragraph or article.

**Ex:-**

```
CREATE TABLE articles (
```

```
    article_id INT PRIMARY KEY,
    title VARCHAR(255),
    content TEXT,
    author VARCHAR(100),
    published_date DATE
);
```

## 3. Date and Time Data Types

**- DATE (Date):**

The DATE data type in SQL is used to store date values. It represents a specific calendar date, including the year, month, and day. The DATE data type does not store time information.

Stores date but not time.

Date values, e.g., '2022-07-30'.

**Ex:-**

INSERT INTO events (event_id, event_name, event_date) VALUES (1, 'Conference', '2024-08-15');

- **TIME (Time):**

The TIME data type in SQL is used to store time values, which represent the time of day without any date component. This includes hours, minutes, and seconds.

**Time Values**: Represents the time of day in the format HH:MM:SS.

Stores time but not date.

Time values, e.g., '14:30:00'.

**Ex:-**

INSERT INTO schedules (schedule_id, event_name, event_time) VALUES (1, 'Meeting', '09:30:00');

- **DATETIME (Date and Time):**

Represents both the date and time of day in the format YYYY-MM-DD HH:MM:SS.

Stores date and time both values.

Combination of date and time values, e.g., '2022-07-30 14:30:00'.

**Ex:-**

INSERT INTO appointments (appointment_id, client_name, appointment_datetime) VALUES (1, 'Alice', '2024-08-15 09:30:00');

- **TIMESTAMP (Timestamp):**

A timestamp is a data type used to store date and time information.
It can be represented in various ways depending on the programming language, database system, or application.
Date and time values with fractional seconds, e.g., '2022-07-30 14:30:00.123'.

**Ex:-**
INSERT INTO events (event_name, event_timestamp) VALUES ('Meeting', '2024-07-30 14:23:45');

## 4. Numeric Data Types

**- DECIMAL (Decimal):**
**Syntax:**
DECIMAL(precision, scale)
- **precision**: The maximum number of total digits.
- **scale**: The number of digits to the right of the decimal point

**Ex:-**
```
CREATE TABLE products
(
    product_id INT,
    product_name VARCHAR(100),
    price DECIMAL(10, 2)
);
```
Fixed-point numbers, e.g., 12.34.

**- FLOAT (Floating-point):**
The FLOAT data type in SQL is used to store approximate numeric values with floating-point precision.
Numbers that can have a fractional part and can be very large or very small.
Floating-point numbers, e.g., 12.3456.
**Ex:-**
INSERT INTO measurements (measurement_id, value) VALUES (1, 12345.6789);

**- DOUBLE(Double):**

The DOUBLE data type in SQL, also known as DOUBLE PRECISION, is used to store approximate numeric values with double the precision of the FLOAT data type.

Double-precision floating-point numbers, e.g., 12.3456789.

**Ex:-**

INSERT INTO measurements (measurement_id, precise_value) VALUES (1, 123456789.123456789);

## 5. Boolean Data Type

- **BOOLEAN (Boolean):**

Represents true or false. Some SQL databases also allow NULL as a possible value for a boolean column.

True or false values, e.g., TRUE or FALSE.

**Ex:-**

INSERT INTO users (user_id, username, is_active) VALUES (1, 'Alice', TRUE);
INSERT INTO users (user_id, username, is_active) VALUES (2, 'Bob', FALSE);

## 6. Binary Data Types

- BLOB (Binary Large Object): Large binary data, e.g., images or files.
- BINARY (Binary): Binary data, e.g., a small image or file

# Introduction to SQL *PLUS

**SQL*Plus** is a **command-line tool** for **Oracle Database** that allows users to interact with the database using **SQL** and **PL/SQL commands**.
SQL*Plus is an interactive and batch query tool that is included with Oracle Database.*
*It provides a command-line interface to Oracle Database, allowing users to execute SQL and PL/SQL commands, manage database objects, and perform various database administration tasks.*

# Getting Started with SQL*Plus

## *Installation and Setup*

1. **Installation**: SQL*Plus is typically installed as part of the Oracle Database software. Ensure that Oracle Database and Oracle Client software are installed on your system.
2. **Environment Variables**: Set up environment variables such as ORACLE_HOME and PATH to include the directory where SQL*Plus is installed.

## *Starting SQL*Plus*

To start SQL*Plus:

- **Windows**: Open Command Prompt and type sqlplus.
- **Linux/Unix**: Open Terminal and type sqlplus.

## *Logging In*

You will be prompted to enter your username and password. You can also provide the connection string directly:

```
sqlplus username/password@database
```

## *Basic Commands*

Here are some basic commands to get you started:

1. **Connecting to a Database**

   ```
   CONNECT username/password@database
   ```

2. **Executing SQL Statements**:

   ```
   SELECT * FROM employees;
   ```

3. **Running Scripts**: You can run SQL scripts using the @ command:

   @script_name.sql

4. **Exiting SQL*Plus**:
   EXIT

## ➢ Common SQL*Plus Commands:

1. CONNECT: Connect to a database as a specific user.
2. SQLPLUS: Start SQL*Plus and connect to a database.
3. EXIT: Exit SQL*Plus.
4. HELP: Display help for SQL*Plus commands.
5. DESCRIBE: Describe the structure of a database object, like a table or view.

## ➢ SQL*Plus Modes:

1. Command Mode: Execute SQL and PL/SQL commands.
2. Editor Mode: Edit SQL and PL/SQL commands in a buffer.
3. Browser Mode: Browse and execute SQL and PL/SQL commands from a file.

## ➢ Here are the advantages of SQL*Plus:-

- **Fast and Lightweight**: Starts quickly and uses minimal resources.
- **Flexible Use**: Can be used interactively or for running batch scripts.
- **Powerful Commands**: Offers many commands for managing and querying databases.
- **Customizable Output**: Format query results to meet your needs.
- **Output to Files**: Save query results and logs to files easily.
- **Handles Errors Well**: Manage what happens when errors occur.
- **Secure**: Works with Oracle's advanced security features.
- **Easy to Learn**: Straightforward commands similar to SQL.
- **Multiple Databases**: Connect to and manage different databases.
- **Supports Oracle Features**: Fully supports all Oracle Database features.
- **Cross-Platform**: Runs on Windows, Linux, and Unix.
- **Good Documentation**: Extensive support and guides available.

# SQL *PLUS formatting commands:-

## 1)SET Command

The SET command is used to set various environment variables that control the behavior of SQL*Plus.

*Example: Setting Page Size and Line Size*

```
SET PAGESIZE 50
SET LINESIZE 100
SELECT * FROM employees;
```

**Output:-**

| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | EMAIL | PHONE_NUMBER | HIRE_DATE | JOB_ID | SALARY |
|---|---|---|---|---|---|---|---|
| 100 | Steven | King | SKING | 515.123.4567 | 17-JUN | AD_PRES | 24000 |
| 101 | Neena | Kochhar | NKOCHHAR | 515.123.4568 | 21-SEP-05 | AD_VP | 17000 |
| 102 | Lex | De Haan | LDEHAAN | 515.123.4569 | 13-JAN-01 | AD_VP | 17000 |
| 103 | Alexander | Hunold | AHUNOLD | 590.423.4567 | 03-JAN-06 | IT_PROG | 9000 |
| 104 | Bruce | Ernst | BERNST | 590.423.4568 | 21-MAY-07 | IT_PROG | 6000 |

## 2)BREAK Command

The BREAK command is used to suppress the printing of duplicate values in specified columns.

It is used to break any job (task) of a particular transaction.

**Ex:-** break on job1;

We can on and off break command.

Break command is always used to with skip command.

**Ex:-** skip2

skip

*Example: Breaking on a Column*

BREAK ON department_id SKIP 1

## 3)COMPUTE Command

The COMPUTE command is used to calculate summary values for groups of rows defined by the BREAK command.

*Example: Summarizing Salaries by Department*

BREAK ON department_id SKIP 1

COMPUTE SUM OF salary ON department_id

**Output:-**

| DEPARTMENT_ID | EMPLOYEE_ID | FIRST_NAME | SALARY |
|---|---|---|---|
| 10 | 200 | Jennifer | 4400 |
| ********** | | | |
| sum | 4400 | | |
| 20 | 201 | Michael | 13000 |
| | 202 | Pat | 6000 |
| ********** | | | |
| sum | 19000 | | |

## 4)SPOOL Command

The SPOOL command is used to write the output of SQL commands to a file.

### Syntax:-

SPOOL [filename[.ext] | OFF | OUT]

*Example: Spooling Output to a File*

SPOOL output.txt

SELECT * FROM employees WHERE ROWNUM <= 5;

SPOOL OFF

**filename[.ext]**: The name of the file where the output will be written. If no extension is provided, .list is used by default.
**OFF**: Stops spooling and closes the spool file.
**OUT**: Stops spooling, closes the spool file, and sends the contents of the file to the default printer.

SQL> SPOOL output.txt;

SQL> SELECT * FROM departments;

**SQL> SPOOL OFF;**

SQL> EXIT

Disconnected from Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

## 5)SAVE Command:

This allows you to save buffer contents into a file.

## 6) DESCRIBE :

Describe command is used to view the structure of table.

**Syntax:-**

Desc<table name>;

**Ex:-**

Desc emp;

# SQL v/s SQL *PLUS

| SQL | SQL*PLUS |
|---|---|

| | |
|---|---|
| SQL is a language | SQL *Plus is an environment or tool. |
| SQL is created as per the ANSI standards | SQL *Plus is ORACLE proprietary |
| SQL keywords can not be abbreviated | SQL *Plus keywords can be abbreviated |
| SQL Statements manipulate data and table definitions in the database. | SQL *Plus commands do not allow manipulation of values in the database |
| SQL statements are passes to SQL Buffer | SQL *Plus commands are not passes to SQL Buffer |
| SQL is used for querying, manipulating, and defining data in databases. | SQL*Plus is used for executing SQL and PL/SQL commands, formatting query results, and performing database administration tasks. |
| SQL commands include SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, etc. | SQL*Plus commands include CONNECT, DISCONNECT, DESCRI |

# Operator and Expression:-

An operator is a reserved word or a character that is used to query our database in a SQL expression.

To query a database using operators, we use a WHERE clause. Operators are necessary to define a condition in SQL, as they act as a connector between two or more conditions.

Every database administrator and user uses SQL queries for manipulating and accessing the data of database tables and views.

The SQL reserved words and characters are called operators.

**SQL Operators** perform arithmetic, comparison, and logical operations to manipulate and retrieve data from databases.

Operators in SQL are symbols that help us to perform specific mathematical and logical computations on operands.
An operator can either be unary or binary.
The unary operator operates on one operand, and the binary operator operates on two operands.

**Types of Operators in SQL:-**

Different types of operators in SQL are:

- Arithmetic operator
- Comparison operator
- Logical operator
- Bitwise Operators
- Compound Operators

# 1) SQL Arithmetic Operators

**Arithmetic operators** in SQL are used to perform mathematical operations on numeric values in queries. Some common arithmetic operators are:

| Operator | Description |
|---|---|
| + | The addition is used to perform an addition operation on the data values. |
| – | This operator is used for the subtraction of the data values. |
| / | This operator works with the 'ALL' keyword and it calculates division operations. |
| * | This operator is used for multiplying data values. |
| % | Modulus is used to get the remainder when data is divided by another. |

## SQL Arithmetic Operators Example:-

***Addition (+) :***

It is used to perform **addition operation** on the data items, items include either single column or multiple columns.

**Implementation**:

```
SELECT employee_id, employee_name, salary, salary + 100
  AS "salary + 100" FROM addition;
```

Output:

| employee_id | employee_name | salary | salary+100 |
|---|---|---|---|
| 1 | alex | 25000 | 25100 |
| 2 | rr | 55000 | 55100 |
| 3 | jpm | 52000 | 52100 |
| 4 | ggshmr | 12312 | 12412 |

Here we have done addition of 100 to each Employee's salary i.e, addition operation on single column.

Let's perform **addition of 2 columns**:

```
SELECT employee_id, employee_name, salary, salary + employee_id
  AS "salary + employee_id" FROM addition;
```

Output:

| employee_id | employee_name | salary | salary+employee_id |
|---|---|---|---|
| 1 | alex | 25000 | 25001 |
| 2 | rr | 55000 | 55002 |
| 3 | jpm | 52000 | 52003 |
| 4 | ggshmr | 12312 | 12316 |

## 2) SQL Comparison/ Reletional Operators:-

**Comparison Operators** in SQL are used to compare one expression's value to other expressions. SQL supports different types of comparison operator, which are described below:

In **SQL Server**, **relational operators** are used to **compare** values and establish relationships between **data** stored in **tables**. These operators allow us to perform **logical comparisons** to filter data based on specific conditions.

Relational operators in SQL Server are used to compare values and determine the relationship between them.

These operators evaluate conditions and return a boolean (true or false) result based on whether the specified condition is met. Common relational operators include:

- **Equal to (=):** This operator checks if two values are the same. Generally, it compares values for equality.
- **Not equal to (<> or !=):** The not equal to operator checks if two values are different. It returns true if the values are not equal.
- **Greater than (>):** This operator checks if the left value is greater than the right value.
- **Less than (<):** Similar to the greater than operator, but it checks if the left operand is less than the right operand.
- **Greater than or equal to (>=):** This operator checks if the left operand is greater than or equal to the right operand.
- **Less than or equal to (<=):** Similar to the greater than or equal to operator, but it checks if the left operand is less than or equal to the right operand.

## Examples of Relational Operators :-

To understand **Relational Operators in SQL Server** we need a table on which we will perform various operations and queries. Here we will consider a table called **Persons** which contains data as shown below:

```
CREATE TABLE Persons (
    EmpID INT,
    Name NVARCHAR(50),
    Department NVARCHAR(50),
    Salary DECIMAL(10, 2),
    Age INT
);

INSERT INTO Persons(EmpID, Name, Department, Salary, Age)
VALUES
    (1, 'John Doe', 'Sales', 60000.00, 35),
    (2, 'Jane Smith', 'Marketing', 45000.00, 28),
    (3, 'Robert Johnson', 'HR', 55000.00, 42),
    (4, 'Lisa Brown', 'Sales', 70000.00, 45),
    (5, 'Michael Davis', 'IT', 65000.00, 30);
```

| PersonsID | Name | Department | Salary | Age |
|---|---|---|---|---|
| 1 | John Doe | Sales | 60000 | 35 |
| 2 | Jane Smith | Marketing | 45000 | 28 |
| 3 | Robert Johnson | HR | 55000 | 42 |
| 4 | Lisa Brown | Sales | 70000 | 45 |
| 5 | Michael Davis | IT | 65000 | 30 |

## Example 1: Equal to (=)

Select Persons whose department is ‘Sales’

```
-- Select Persons whose department is 'Sales'
SELECT * FROM Persons
WHERE Department = 'Sales';
```

| PersonsID | Name | Department | Salary | Age |
|---|---|---|---|---|
| 1 | John Doe | Sales | 60000 | 35 |
| 4 | Lisa Brown | Sales | 70000 | 45 |

In this example, we will retrieve all records from the “MATHS” table where the value in the “MARKS” column is equal to 50.

**Query:**

```
SELECT * FROM MATHS WHERE MARKS=50;
```

**Output:**

## Example 2: Not equal to (<>)

Select employees whose department is not ‘Sales’

```
-- Select Persons whose Salary is not equal to 100
SELECT * FROM Persons
WHERE Salary<> 45000;
```

| PersonsID | Name | Department | Salary | Age |
|---|---|---|---|---|
| 1 | John Doe | Sales | 60000 | 35 |
| 3 | Robert Johnson | HR | 55000 | 42 |
| 4 | Lisa Brown | Sales | 70000 | 45 |
| 5 | Michael Davis | IT | 65000 | 30 |

## Example 3: Greater than (>)

Select Persons whose Age is greater than 35

```
-- Select Person with a Age  greater than 35
SELECT * FROM Persons
WHERE Age> 35;
```

| PersonsID | Name | Department | Salary | Age |
|---|---|---|---|---|
| 3 | Robert Johnson | HR | 55000 | 42 |
| 4 | Lisa Brown | Sales | 70000 | 45 |

## Example 4: Less than (<)

Select Persons whose age is less than 35

```
-- Select Persons whose age is less than 30
SELECT * FROM Persons
WHERE Age < 35;
```

| PersonsID | Name | Department | Salary | Age |
|---|---|---|---|---|
| 2 | Jane Smith | Marketing | 45000 | 28 |
| 5 | Michael Davis | IT | 65000 | 30 |

## Example 5: Greater than or equal to (>=)

Select Persons whose salary is greater than or equal to 60000

```
-- Select Persons whose salary is greater than or equal to 60000
SELECT * FROM Persons
WHERE salary >= 60000;
```

| PersonsID | Name | Department | Salary | Age |
|---|---|---|---|---|
| 1 | John Doe | Sales | 60000 | 35 |
| 4 | Lisa Brown | Sales | 70000 | 45 |
| 5 | Michael Davis | IT | 65000 | 30 |

## Example 6: Less than or equal to (<=)

Select Persons whose Salary is less than or equal to 60000

```
-- Select products with a Salary less than or equal to 60000
SELECT * FROM Persons
WHERE Salary <= 60000;
```

| PersonsID | Name | Department | Salary | Age |
|---|---|---|---|---|
| 1 | John Doe | Sales | 60000 | 35 |
| 2 | Jane Smith | Marketing | 45000 | 28 |
| 3 | Robert Johnson | HR | 55000 | 42 |

# Example 7: Combining Relational Operators with Logical Operators

Relational operators can also be combined with logical operators (AND, OR, NOT) to create complex conditions in SQL queries:

```
-- Select products with a price between 50 and 100
SELECT * FROM Persons
WHERE PersonsID >= 2 AND PersonsID <= 4;
```

| PersonsID | Name | Department | Salary | Age |
|---|---|---|---|---|
| 2 | Jane Smith | Marketing | 45000 | 28 |

| PersonsID | Name | Department | Salary | Age |
|---|---|---|---|---|
| 3 | Robert Johnson | HR | 55000 | 42 |
| 4 | Lisa Brown | Sales | 70000 | 45 |

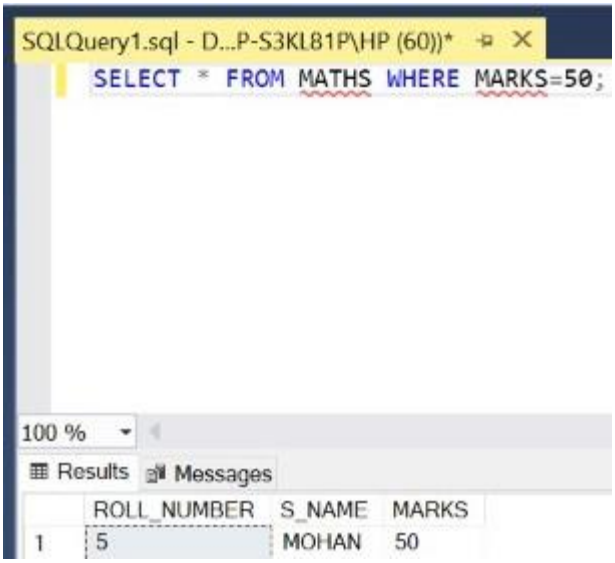
**SQL Comparison Operators Example :-**

In this example, we will retrieve all records from the “MATHS” table where the value in the “MARKS” column is equal to 50.

**Query:**

```
SELECT * FROM MATHS WHERE MARKS=50;
```

**Output:**




# 3)SQL – Logical Operators:-

SQL logical operators are used to test for the truth of the condition. A logical operator like the Comparison operator returns a Boolean value of **TRUE**, **FALSE**, or **UNKNOWN.** In this article, we will discuss different types of Logical Operators.

Logical operators are used to combine or manipulate the conditions given in a query to retrieve or manipulate data .there are some logical operators in SQL like OR, AND etc.

| Operator | Description |
|---|---|
| AND | Logical AND compares two Booleans as expressions and returns true when both expressions are true. |
| OR | Logical OR compares two Booleans as expressions and returns true when one of the expressions is true. |
| NOT | Not takes a single Boolean as an argument and change its value from false to true or from true to false. |

## Example of AND Operator :-

The AND operator is used to combines two or more conditions but if it is true when all the conditions are satisfied.

**Query :-**

```
SELECT * FROM employee WHERE emp_city = 'Allahabad' AND emp_country = 'India';
```

**Output :-**

| emp_id | emp_name | emp_city | emp_country |
|---|---|---|---|
| 104 | Utkarsh Singh | Allahabad | India |
| 105 | Sudhanshu Yadav | Allahabad | India |

## OR Operator:-

The OR operator is used to combines two or more conditions but if it is true when one of the conditions are satisfied.

**Query:-**

```
SELECT * FROM employee WHERE emp_city = 'Varanasi' OR emp_country = 'India';
```

**Output:-**

| emp_id | emp_name | emp_city | emp_country |
|---|---|---|---|
| 101 | Utkarsh Tripathi | Varanasi | India |
| 102 | Abhinav Singh | Varanasi | India |
| 103 | Utkarsh Raghuvanshi | Varanasi | India |
| 104 | Utkarsh Singh | Allahabad | India |
| 105 | Sudhanshu Yadav | Allahabad | India |
| 106 | Ashutosh Kumar | Patna | India |

## Not Operator:-

SQL NOT Operator is used to return the opposite result or negative result. It is a logical operator in SQL, that negates the boolean expression in the **WHERE** clause.
It is mostly used to specify what should not be included in the results table.

**NOT Syntax**

```
SELECT column1, colomn2, …
FROM table_name WHERE NOT condition;
```

## Ex:-

**SELECT * FROM Customers WHERE NOT Country='UK';**

| Customer ID | Customer Name | City | PostalCode | Country |
|---|---|---|---|---|
| 1 | John Wick | New York | 1248 | USA |
| 3 | Rohan | New Delhi | 100084 | India |

# 4)SQL – Bitwise Operators:-

Bitwise algorithms are algorithms that operate on individual bits of data rather than on larger data types like integers or floating-point numbers. These algorithms manipulate bits directly, typically using bitwise operators such as AND, OR, XOR, shift left, shift right, and complement.

## Common Bitwise Algorithms and Operations:

Here are some common bitwise algorithms and operations:

- **Bitwise AND (&):** Takes two numbers as input and performs a bitwise AND operation on their corresponding bits. It returns 1 only if both bits are 1; otherwise, it returns 0.
- **Bitwise OR (|):** Performs a bitwise OR operation on the corresponding bits of two numbers. It returns 1 if at least one of the bits is 1.
- **Bitwise XOR (^):** Performs a bitwise exclusive OR operation on the corresponding bits of two numbers. It returns 1 if the bits are different and 0 if they are the same.
- **Bitwise NOT (~):** Performs a bitwise NOT operation, which flips each bit of the input (1 becomes 0 and 0 becomes 1).
- **Left Shift (<<) and Right Shift (>>):** These operators shift the bits of a number to the left or right by a specified number of positions. Left shifting is equivalent to multiplying the number by 2, while right shifting is equivalent to dividing by 2.

## 5) SQL Compound Operators

Compound operator in SQL are used to perform an operation and assign the result to the original value in a single line. Some Compound operators are:

| Operators | Explanation |
|---|---|
| += | This operator can be used for incrementing the value of the variable on the left side of the operator by the operand or value specified on the right side of the operator. |
| -= | This operator can be used for decrementing the value of the variable on the left side of the operator by the operand or the value specified on the right side of the operator. |
| *= | This operator can be used for multiplying the variable on the left side of the operator with the operand or value on the right side of the operator and storing the updated value in the left-hand side variable. |
| /= | This operator can be used for updating the value of a variable by dividing the variable by the value specified on the right side of the operator. |
| %= | This operator stores the remainder when the variable is divided by a given value in the variable itself. |
| &= | This operator can be used for updating the value of a variable after performing a bitwise AND operation on the given operands, one operand is the variable itself. |
| \|= | This operator can be used for updating a variable with the output of the bitwise OR operation performed on the given operands, one operand being the variable itself. |

| | |
|---|---|
| ^= | This operator can be used for updating a variable with the output of the bitwise exclusive OR operation performed on the given operands, one operand being the variable itself. |