SQL

SQL ( Structured Query Language ) - SQL (Structured Query Language) is a standardized language used to interact with relational databases. It allows users to create, read, update, and delete data in a database.

Types Of SQL Commands : -

1. DDL Command(Data Definition Language)
2. DQL Command(Data Query Language)
3. DML Command (Data Manipulation Language)
4. DCL Command (Data Control Language)
5. TCL Command (Transaction Control Language)
6. DDL Command - DDL commands are used to define and manage database structures such as tables, schemas, and indexes.

Commands -

1. **Create** **Clause** - to create objects in the database .
2. Create Database : -

Syntax - **Create Database Database\_Name ;**

(Database\_Name : Name of the database)

1. Create Table : -

Syntax - **CREATE TABLE table\_name (**

**column1 datatype(size),**

**column2 datatype (size),**

**...**

**);**

table\_name - name of the table .

column1 - name of the first column .

datatype - Type of data . Ex - int for integer data .

B. **ALTER Clause -** alters the structure of the database . Modifies an existing database object . ALTER TABLE is used to add, delete/drop or modify columns in the existing table.

1. Adding a Column  **: -** To add a new column to an existing table:

Syntax : - **ALTER TABLE table\_name**

**ADD (column\_name1 datatype, column\_name2 datatype );**

2.Dropping a Column : - To remove an existing column from a table.

Syntax : - **ALTER TABLE table\_name**

**DROP COLUMN column\_name;**

3. Modifying a Column **: -**To change the data type or size of an existing column.

Syntax : - **ALTER TABLE table\_name**

**MODIFY column\_name new\_datatype;**

4. Renaming a Column **:-** To rename an existing column.

Syntax : - **ALTER TABLE table\_name**

**RENAME COLUMN old\_column\_name TO new\_column\_name;**

C. **Rename Clause : -** Renames the table’ s name.

Syntax : - **RENAME TABLE old\_table\_name TO new\_table\_name;**

UsingALTER Clause

Syntax : - **ALTER TABLE old\_table\_name RENAME TO new\_table\_name;**

D. **DROP Clause : -** The **DROP** command is used to delete entire database objects like tables, databases, indexes, or views. When we drop a table, all the data in the table and the table structure itself are removed permanently.

Syntax : - **DROP TABLE table\_name;** ( deletes the table )

**DROP DATABASE database\_name;** ( deletes the database )

E. **Truncate CLause** : - The **TRUNCATE** command is used to delete all rows from a table but keep the table structure for future use.

Syntax : - **TRUNCATE TABLE table\_name;**

**Data Types : -** SQL data types define the kind of data that can be stored in a table column.

1. **Numeric Data Types** : -
2. **INT : -**  used for integer values . Integer, a whole number.
3. **FLOAT**: Approximate numeric, floating-point number.Stores approximate values with floating-point precision

it provides about 6-9 significant decimal digits of precision.

1. **DOUBLE**: Double-precision floating-point number. (Higher precision than FLOAT) - which provides about 15 decimal digits of precision.
2. **DECIMAL** or **NUMERIC**: Exact numeric, fixed-point number, where we can specify the precision and scale.

2 **Character and String Data Types : -**

1. **CHAR(size)**: Fixed-length string.
2. **VARCHAR(size)**: Variable-length string.

3. **Date and Time Data Types**

1. **DATE**: Date in the format YYYY-MM-DD.
2. **TIME**: Time in the format HH:MM:SS.
3. **DATETIME**: Date and time in the format YYYY-MM-DD HH:MM:SS.
4. **TIMESTAMP**: Date and time with time zone.
5. **YEAR**: Year in the format YYYY.

**INSERT INTO Clause : -** The INSERT INTO statement of SQL is used to insert a new row in a table. There are two ways of using INSERT INTO statement for inserting rows:

1. **Only values:** First method is to specify only the value of data to be inserted without the column names.  
     
   **Syntax : - INSERT INTO Table\_name VALUES (Value1 , Value2 , Value3,.......)**

**Table\_name - name of the table**

**Value1 - value of first column**

**Value2 - value of second column**

1. **Values with Column Name:** In the second method we will specify both the columns which we want to fill and their corresponding values as shown below:  
   Syntax:**Syntax : - INSERT INTO Table\_name(Column1 , column2 ……) VALUES (Value1 , Value2 , Value3,.......)**

**Table\_name - name of the table**

**Value1 - value of first column**

**Value2 - value of second column**

**SQL CONSTRAINTS :-** SQL constraints are used to specify rules for data in a table.

1. **NOT NULL : -**

Syntax : - **CREATE TABLE table\_name (**

**column\_name1 data\_type NOT NULL,**

**column\_name2 data\_type**

**);**

**(**column can not have null values)

1. **UNIQUE : -**

Syntax : - **CREATE TABLE table\_name (**

**column\_name1 data\_type UNIQUE,**

**column\_name2 data\_type**

**);**

**(**all values in columns are different )

1. **Primary key : -**

Syntax : - **CREATE TABLE table\_name (**

**column\_name1 data\_type PRIMARY KEY,**

**column\_name2 data\_type**

**);**

**OR**

**CREATE TABLE table\_name (**

**column\_name1 data\_type,**

**column\_name2 data\_type,**

**PRIMARY KEY (Column\_Name)**

**);**

Makes a column unique and not null but used for only one .

1. **Foreign Key** : - prevent actions that would destroy links between tables .

Syntax : - **CREATE TABLE table\_name (**

**column\_name1 data\_type,**

**column\_name2 data\_type,**

**FOREIGN KEY (column\_name1) REFERENCES other\_table\_name(other\_column\_name)**

**);**

1. **DEFAULT : -** sets the default value of a column.

Syntax : - **CREATE TABLE table\_name (**

**column\_name1 data\_type DEFAULT default\_value,**

**column\_name2 data\_type**

**);**

1. **CHECK :-**  it can limit the values allowed in a column .

**CREATE TABLE table\_name (**

**column\_name1 data\_type CHECK (condition),**

**column\_name2 data\_type**

**);**

Example : - **CREATE TABLE fun (**

**Age INT CHECK (age>=18) );**

**DQL (Data Query Language) :** Data Query Language (DQL) is a subset of SQL (Structured Query Language) used for querying and retrieving data from a database. DQL primarily involves the SELECT statement, which allows users to fetch data from one or more tables. The focus of DQL is to retrieve the data based on the specified criteria.

**Key Components of DQL** : -

1. **SELECT Statement** : The SELECT statement is used to specify the columns of data we want to retrieve from a table.

Syntax : - **SELECT column\_name1 , column\_name2 from table\_name ;**

To fetch the entire table or all the fields in the table

Syntax : - **SELECT \* From table\_Name ;**

1. **WHERE Clause : -** The WHERE clause is used to filter rows based on specific conditions.
2. **ORDER BY Clause : -** The ORDER BY clause is used to sort the result set based on one or more columns.

By default ORDER BY sorts the data in **ascending order.**

We can use the keyword DESC to sort the data in descending order and the keyword ASC to sort in ascending order.

1. Sort according to one column **: -**

Syntax : - **SELECT \* From table\_name order by column\_name ASC|DESC ;**

1. Sort according to multiple columns **: -**

Syntax : - **SELECT \* From table\_name order by column\_name1 ASC|DESC , column\_name2 ASC|DESC .**

4. **DISTINCT Clause : -** The DISTINCT clause in SQL is used to remove duplicate rows from the result set of a SELECT statement. This ensures that the result set contains only unique rows based on the specified columns.

Syntax : - **SELECT DISTINCT column\_name1 ,column\_name2 From table\_Name ;**

**5. LIke Between IN clause : -**

**Between Clause : -** The BETWEEN operator is used to filter the result set within a certain range.

Syntax : - **SELECT \* From Table\_Name where Column\_Name Between Value1 AND Value2;**

**Ex : -** Query to get Student Records, Where Student's age is between 19 and 21.

**IN** **Clause : -** The IN clause in SQL is used to specify multiple values in a WHERE clause. It allows you to filter the result set to only include rows where the column value matches any value in a specified list.

Syntax : - **SELECT \* FROM table\_name**

**WHERE column\_name IN (value1, value2, ...);**

**Ex : -** Query to get Student Records, Where Student's age is in the set {18,20,21}.

**LIKE Clause : -** The LIKE clause in SQL is used to search for a specified pattern in a column. It is commonly used in a SELECT statement's WHERE clause to filter rows that match a pattern.

There are two kinds of wildcards used to filter out the results:

**Percent Sign (%)**: Represents zero, one, or multiple characters.

**Underscore (\_)**: Represents a single character.

Ex : - **SELECT \* FROM employees WHERE first\_name LIKE 'J%';**

This query retrieves all employees whose first name starts with 'J'.

**SELECT \* FROM employees WHERE first\_name LIKE 'J\_n';**

This query retrieves all employees whose first name is three characters long, starts with 'J', and ends with 'n'.

**SELECT \* FROM employees WHERE first\_name LIKE 'J%n';**

This query retrieves all employees whose first name starts with 'J' and ends with 'n'.

**Aggregate Function : -** Aggregate functions in SQL perform calculations on a set of values and return a single result.

**Common Aggregate Functions** : -

1. **COUNT(): Returns the number of rows.**

Syntax : -

**SELECT COUNT(\*) FROM Table\_Name;**

This query returns the total number of rows in the table.

**SELECT COUNT(Column\_Name) FROM Table\_Name;**

This query returns the count of non-null Coumn\_Name values in the table.

**SELECT COUNT(\*) FROM Table\_Name WHERE Condition;**

Counting Rows with a Condition .

**SELECT COUNT(DISTINCT Column\_Name) FROM Table\_Name;**

The DISTINCT keyword can be used within the COUNT() function to count the number of distinct (unique) values in a column.

1. **SUM() : -** The SUM() aggregate function in SQL is used to calculate the total sum of a numeric column.

Syntax : - **SELECT SUM(column\_name)**

**FROM table\_name**

**WHERE condition;**

**SELECT SUM( DISTINCT column\_name)**

**FROM table\_name**

**WHERE condition; // sum of all distinct column values .**

1. **AVG() : -** The AVG() aggregate function in SQL is used to calculate the average value of a numeric column.

Syntax : - **SELECT AVG(column\_name)**

**FROM table\_name**

**WHERE condition;**

**SELECT AVG(DISTINCT column\_name)**

**FROM table\_name**

**WHERE condition;**

1. **MIN() : -** The MIN() aggregate function in SQL is used to find the minimum value in a numeric column.

Syntax :-

**SELECT MIN(column\_name)**

**FROM table\_name**

**WHERE condition;**

1. **MAX() :-** The MAX() aggregate function in SQL is used to find the maximum value in a numeric column.

Syntax : -

**SELECT MAX(column\_name)**

**FROM table\_name**

**WHERE condition;**

**GROUP BY : -** The GROUP BY clause in SQL is used to arrange identical data into groups. This clause is often used in conjunction with aggregate functions (such as COUNT, SUM, AVG, MAX, MIN) to perform calculations on each group of data.

Syntax : - **SELECT column1, aggregate\_function(column2)**

**FROM table\_name**

**WHERE condition**

**GROUP BY column1, column2;**

**Having Clause : -** The HAVING clause in SQL is used to filter groups of rows created by the GROUP BY clause based on a specified condition.

Syntax : - **SELECT column1, aggregate\_function(column2)**

**FROM table\_name**

**WHERE condition**

**GROUP BY column1**

**HAVING condition;**

**Limit : -** The LIMIT clause is used to specify the maximum number of rows to return.

Syntax : -**SELECT column1, column2, ...**

**FROM table\_name**

**WHERE condition**

**LIMIT number\_of\_rows;**

Retrieve the first 5 rows from the employees table.

**SELECT id, name, salary, age**

**FROM employees**

**LIMIT 5;**

**OFFSET : -** The OFFSET clause is used to specify the number of rows to skip before starting to return rows.

**Using LIMIT along with OFFSET : -** LIMIT x OFFSET y simply means skip the first y entries and then return the next x entries.

Syntax : - **SELECT column1, column2, ...**

**FROM table\_name**

**WHERE condition**

**LIMIT number\_of\_rows**

**OFFSET offset\_value;**

**DML Command : -** DML (Data Manipulation Language) commands are used to manage and manipulate data within database tables.Here are the primary DML commands in SQL:

**INSERT**: Adds new rows of data to a table.

**UPDATE**: Modifies existing data within a table.

**DELETE**: Removes rows of data from a table.

**Update Clause** : - The UPDATE statement in SQL is used to update the data of an existing table in the database.

Syntax : - **UPDATE table\_name**

**SET column1 = value1, column2 = value2, ...**

**WHERE condition;**

**DELETE Clause : -** The DELETE Statement in SQL is used to delete existing records from a table. We can delete a single record or multiple records depending on the condition we specify in the WHERE clause.

Syntax : - **DELETE FROM table\_name**

**WHERE condition;**

We cannot directly delete a single column value in SQL. Instead, we can set the column value to NULL or to another value, such as 0, depending on our requirements.

Delete a Single Row

**DELETE FROM Table\_name**

**WHERE condition;**

Delete Multiple Rows

Using the IN Operator

**DELETE FROM Table\_name**

**WHERE column\_name IN (value1, value2 , ……);**

Using Multiple Conditions with OR

**DELETE FROM Table\_name**

**WHERE column1 = value1 OR column2= value2;**

Using a Range

**DELETE FROM Table\_name**

**WHERE column\_name BETWEEN a AND b; [a ,b] = Range**

**UNION : -** The UNION operator combines the result sets of two or more SELECT statements. It removes duplicate rows from the result set by default. To include duplicates, you can use UNION ALL.

Syntax :- **SELECT column1, column2, ...**

**FROM table1**

**UNION**

**SELECT column1, column2, ...**

**FROM table2;**

### **INTERSECT**

The INTERSECT operator returns the common rows from the result sets of two or more SELECT statements. Only rows that appear in both result sets are included.

Syntax : - **SELECT column1, column2, ...**

**FROM table1**

**INTERSECT**

**SELECT column1, column2, ...**

**FROM table2;**