## STRUCTURED QUERY

## LANGUAGE(SQL)

## DOCUMENT

## BY,

## DHARINISRI M M

## BATCH:2022-7688

## Enrollment Number:EBEON0722631828

## DATA BASE MANAGEMENT SYSTEM(DBMS)

Database management system is a software which is used to manage the database. For example: [MySQL](https://www.javatpoint.com/mysql-tutorial), [Oracle](https://www.javatpoint.com/oracle-tutorial), etc

**STRUCTURED QUERY LANGUAGE(SQL):**

Structured Query Language(SQL) as we all know is the database language by the use of which we can perform certain operations on the existing database and also we can use this language to create a database. [SQL](https://www.geeksforgeeks.org/structured-query-language/) uses certain commands like Create, Drop, Insert, etc. to carry out the required tasks.

These [SQL](https://www.geeksforgeeks.org/sql-concepts-and-queries/)commands are mainly categorized into four categories as:

1. DDL – Data Definition Language
2. DQl – Data Query Language/ DRL Data Retrieval Language
3. DML – Data Manipulation Language
4. DCL – Data Control Language

Though many resources claim there to be another category of SQL clauses **TCL – Transaction Control Language**. So we will see in detail about TCL as well. 

Diagram

Description automatically generated

**DDL (Data Definition Language):**

[DDL](https://www.geeksforgeeks.org/features-of-structured-query-language-sql/) or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database. DDL is a set of SQL commands used to create, modify, and delete database structures but not data. These commands are normally not used by a general user, who should be accessing the database via an application.

List of DDL commands:

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

**DQL (Data Query Language): Data Retrieval language.**

**DQL**statements are used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it. We can define DQL as follows it is a component of SQL statement that allows getting data from the database and imposing order upon it. It includes the SELECT statement. This command allows getting the data out of the database to perform operations with it. When a SELECT is fired against a table or tables the result is compiled into a further temporary table, which is displayed or perhaps received by the program i.e. a front-end.

List of DQL:

* [**SELECT**](https://www.geeksforgeeks.org/sql-select-clause/)**:**It is used to retrieve data from the database.

**DML(Data Manipulation Language):**

The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements. It is the component of the SQL statement that controls access to data and to the database. Basically, DCL statements are grouped with DML statements.

List of DML commands:

* [**INSERT**](https://www.geeksforgeeks.org/sql-insert-statement/) : It is used to insert data into a table.
* [**UPDATE**](https://www.geeksforgeeks.org/sql-update-statement/)**:** It is used to update existing data within a table.
* [**DELETE**](https://www.geeksforgeeks.org/sql-delete-statement/) : It is used to delete records from a database table.
* [**LOCK:**](https://www.geeksforgeeks.org/sql-lock-table/) Table control concurrency.
* **CALL:**Call a PL/SQL or JAVA subprogram.
* **EXPLAIN PLAN:** It describes the access path to data.

**DCL (Data Control Language):**

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

List of  DCL commands:

* [**GRANT:**](https://www.geeksforgeeks.org/mysql-grant-revoke-privileges/)This commandgives users access privileges to the database.
* [**REVOKE:**](https://www.geeksforgeeks.org/difference-between-grant-and-revoke/)This command withdraws the user’s access privileges given by using the GRANT command.

Though many resources claim there to be another category of SQL clauses **TCL – Transaction Control Language**.

So we will see in detail about TCL as well. TCL commands deal with the [transaction within the database](https://www.geeksforgeeks.org/sql-transactions/).

List of TCL commands:

* [**COMMIT**](https://www.geeksforgeeks.org/sql-transactions/)**:**Commits a Transaction.
* [**ROLLBACK**](https://www.geeksforgeeks.org/sql-transactions/)**:**Rollbacks a transaction in case of any error occurs.
* [**SAVEPOINT**](https://www.geeksforgeeks.org/sql-transactions/)**:**Sets a savepoint within a transaction.
* [**SET TRANSACTION:**](https://www.geeksforgeeks.org/sql-transactions/)Specify characteristics for the transaction.

<https://www.oracle.com/database/technologies/xe-prior-release-downloads.html>

<https://youtu.be/seFRL1GAzLY>

<https://www.testingdocs.com/download-install-mysql-on-windows-11/>

<https://dev.mysql.com/downloads/installer/>

<https://youtu.be/eq-e_n7lm2M>

<https://www.youtube.com/watch?v=WuBcTJnIuzo>

<https://www.youtube.com/watch?v=wEHWYuzP7VE>

**syntax with comment & Example:**

**1.DDL**

**a).CREATE TABLE**

The CREATE TABLE command creates a new table in the database.

**Example:**

CREATE TABLE Persons (  
    PersonID int,LastName varchar(255),  
    FirstName varchar(255),Address varchar(255),  
    City varchar(255));

**b).DROP:**

The DROP TABLE statement is used to drop an existing table in a database.

**Syntax:**

DROP TABLE table\_name;

**Example:**

DROP TABLE customers;

## c).ALTER

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

**Syntax:**

ALTER TABLE table\_name  
ADD column\_name datatype;

**Example:**

ALTER TABLE Customers ADD Email varchar(255);

**d).TRUNCATE TABLE**

The TRUNCATE TABLE command deletes the data inside a table, but not the table itself.

**Example:**

TRUNCATE TABLE customer;

**2.DML**

## a).INSERT INTO & VALUES

The INSERT INTO command is used to insert new rows in a table.

The VALUES command specifies the values of an INSERT INTO statement.

**Example:**

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)  
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

## b).UPDATE

The UPDATE command is used to update existing rows in a table.

**Example:**

UPDATE Customers  
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'  
WHERE CustomerID = 1;

## c).DELETE

The DELETE command is used to delete existing records in a table.

**Example:**

DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

DELETE FROM Customers

**3.DQL/DRL:**

**a).SELECT Statement**

The SELECT statement is used to select data from a database.

**syntax**

SELECT column1, column2, ...  
FROM table\_name;

**Example:**

SELECT \* FROM customer;

**4.TCL:**

**a).COMMIT** : to save the changes.

**Syntax:**

COMMIT

**Example:**

DELETE FROM CUSTOMERS WHERE AGE = 25;

COMMIT;

**b).ROLLBACK** − to roll back the changes.

**Syntax:**

ROLLBACK

**Example:**

DELETE FROM CUSTOMERS WHERE AGE = 25;

ROLLBACK;

**c).SAVEPOINT** − creates points within the groups of transactions in which to ROLLBACK.

**Syntax:**

SAVEPOINt SAVEPOINT NAME;

**Example:**

savepoint A;

insert into customers values(4,'Kawin','Erode',2700),(5,'Murugesan','Karur',2800);

savepoint B;

insert into customers values(6,'Hari','Erode',2900);

select \* from customers;

**d).SET TRANSACTION** − Places a name on a transaction.

**Syntax:**

SET TRANSACTION  [ READ WRITE | READ ONLY ]

**e).HAVING:**

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions

**Syntax:**

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)HAVING conditionORDER BY column\_name(s);

**EXAMPLE**

SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country  
HAVING COUNT(CustomerID) > 5;

**5.DCL**

### **a). GRANT Command**

It is employed to grant a privilege to a user. GRANT command allows specified users to perform specified tasks

**Syntax:**

**GRANT privilege name on o**bject**name to user;**

### b). **REVOKE Command**

It is employed to remove a privilege from a user. REVOKE helps the owner to cancel previously granted permissions.

**Syntax:**

REVOKE **privilege name on o**bject**name to user;**

# **1.SELECT FIRST**

The SQL first() function is used to return the first value of the selected column.

**syntax**

**SELECT** **FIRST**(column\_name) **FROM** table\_name;

**Example:**

**SELECT** **FIRST**(customer\_name) **AS** first\_customer **FROM** customers;

# **2.SELECT LAST**

The **LAST()** function in Structured Query Language shows the last value from the specified column of the table

**Syntax**

**SELECT** **LAST** (Field\_Name) **FROM** Table\_Name ;

**Example:**

**SELECT** **LAST** (CustomerName) **AS** Student **FROM** customer;

# **3.SELECT IN**

SQL IN is an operator used in a SQL query to help reduce the need to use multiple SQL "OR" conditions.

**Syntax**

Expression IN (value 1, value 2 ... value n);

**Example:**

1. **SELECT** \*
2. **FROM** customer
3. **WHERE** Customername IN ( Amit , Raghav, Rajeev)

# **4.SELECT SUM**

It is also known as SQL SUM() function. It is used in a SQL query to return summed value of an expression.

**Syntax**

1. **SELECT** SUM (expression)
2. **FROM** tables
3. **WHERE** conditions;

**Example:**

1. **SELECT** SUM (salary) **AS** "Total Salary"
2. **FROM** employees
3. **WHERE** salary > 20000;

**5.LOGICAL OPERATOR AnD:**

TRUE if all the conditions separated by AND is TRUE

**Example:**

SELECT \* FROM Customers

WHERE City = "London" AND Country = "UK";

**6.LOGICAL OPERATOR OR:**  
TRUE if any of the conditions separated by OR is TRUE

**Example:**

SELECT \* FROM Customers

WHERE City = "London" OR Country = "UK";

**7.LOGICAL OPERATOR** N**OT**:

Displays a record if the condition(s) is NOT TRUE

**Example:**

SELECT \* FROM Customers

WHERE City NOT LIKE 's%';

## 8.JOIN

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

**Different Types of SQL JOINs**

* (INNER) JOIN: Returns records that have matching values in both tables
* LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
* RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
* FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table



## 8.1 INNER JOIN

The INNER JOIN keyword selects records that have matching values in both tables.

**Syntax:**

SELECT column\_name(s)  
FROM table1  
INNER JOIN table2ON table1.column\_name = table2.column\_name;

**Example:**

SELECT Orders.OrderID, Customers.CustomerName  
FROM Orders  
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

**8.2 LEFT JOIN**

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

**Syntax:**

SELECT column\_name(s)  
FROM table1  
LEFT JOIN table2ON table1.column\_name = table2.column\_name;

**Example:**

SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID  
ORDER BY Customers.CustomerName;

## 8.3 RIGHT JOIN

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

**Syntax**

SELECT column\_name(s)  
FROM table1  
RIGHT JOIN table2ON table1.column\_name = table2.column\_name;

### **Example**

SELECT Orders.OrderID, Employees.LastName, Employees.FirstName  
FROM Orders  
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID  
ORDER BY Orders.OrderID;

**8.4 FULL OUTER JOIN**

The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

**Syntax**

SELECT column\_name(s)  
FROM table1  
FULL OUTER JOIN table2ON table1.column\_name = table2.column\_nameWHERE condition;

**Example:**

SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID  
ORDER BY Customers.CustomerName;

## 8.5 SELF JOIN

A self join is a regular join, but the table is joined with itself.

### **Syntax**

SELECT column\_name(s)  
FROM table1 T1, table1 T2  
WHERE condition;

**Example**

SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City  
FROM Customers A, Customers B  
WHERE A.CustomerID <> B.CustomerID  
AND A.City = B.City  
ORDER BY A.City

## 9.UNION

The UNION operator is used to combine the result-set of two or more SELECT statements.

### **Syntax**

SELECT column\_name(s) FROM table1  
UNION  
SELECT column\_name(s) FROM table2;

### **Example**

SELECT City FROM Customers  
UNION  
SELECT City FROM Suppliers  
ORDER BY City;

## 10.GROUP BY Statement

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

### **Syntax**

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)ORDER BY column\_name(s);

### **Example**

SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country;

**11.AND:**

The AND operator displays a record if all the conditions separated by AND are TRUE.

### **Syntax:**

SELECT column1, column2, ...  
FROM table\_name WHERE condition1 AND condition2 AND condition3 ...;

**Example:**

SELECT \* FROM Product

WHERE ProductID=3 AND price=40;

**12.OR:**

The OR operator displays a record if any of the conditions separated by OR is TRUE.

### **Syntax:**

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 OR condition2 OR condition3 ...;

**Example:**

SELECT \* FROM Product

WHERE ProductID=3 OR price=30;

**13.NOT:**

The NOT operator displays a record if the condition(s) is NOT TRUE.

### **Syntax:**

SELECT column1, column2, ...  
FROM table\_name  
WHERE NOT condition;

**Example:**

SELECT \* FROM Product

WHERE NOT ProductID=3;

**14.MINIMUM:**

The MIN() function returns the smallest value of the selected column.

### **Syntax:**

SELECT MIN(column\_name)  
FROM table\_name  
WHERE condition;

**Example:**

SELECT MIN(price) AS SmallestPrice

FROM Product;

**15.MAXIMUM:**

The MAX() function returns the largest value of the selected column.

### **Syntax:**

SELECT MAX(column\_name)  
FROM table\_name  
WHERE condition;

**Example:**

SELECT MAX(Price) AS LargestPrice

FROM Product;

**16.COUNT:**

The COUNT() function returns the number of rows that matches a specified criterion.

### **Syntax:**

SELECT COUNT(column\_name)  
FROM table\_name  
WHERE condition;

**Example:**

SELECT COUNT(ProductID)

FROM Product;

**17.AVERAGE:**

The AVG() function returns the average value of a numeric column.

### **Syntax:**

SELECT AVG(column\_name)  
FROM table\_name  
WHERE condition;

**Example:**

SELECT AVG(Price)

FROM Product;

**18.SUM:**

The SUM() function calculates the sum of a set of values.

**Syntax:**

SUM(*expression*)

**Example:**

SELECT SUM(ProductID)

FROM Product;

**19.UPPER:**

The UPPER() function converts a string to upper-case.

**Syntax:**

UPPER(text)

**Example:**

select upper('Shampoo');

**20.LOWER:**

The LOWER() function converts a string to lower-case.

## Syntax:

LOWER (text)

**Example:**

SELECT LOWER('PEN');

**21.TRIM:**

The TRIM() function removes the space character OR other specified characters from the start or end of a string.

**Syntax:**

TRIM([*characters*FROM]*string*)

**Example:**

SELECT TRIM(' SQL Tutorial ') AS TrimmedString;

**22.REVERSE:**

The REVERSE() function reverses a string and returns the result.

**Syntax:**

REVERSE(*string*)

**Example:**

SELECT REVERSE("SQL Tutorial");

**23.REPEAT:**

The REPEAT() function repeats a string as many times as specified.

**Syntax:**

REPEAT(*string*, *number*)

**Example:**

SELECT REPEAT("SQL Tutorial", 3);

**24.POSITION:**

The POSITION() function returns the position of the first occurrence of a substring in a string.

**Syntax:**

POSITION(substring IN string)

**Example:**

SELECT POSITION("3" IN "W3Schools.com") AS MatchPosition;

**25.MID:**

The MID() function extracts a substring from a string (starting at any position).

## Syntax:

MID(string, start, length)

**Example:**

SELECT MID(ProductName, 2, 5) AS ExtractString

FROM Product;

**26.LOCATE:**

The LOCATE() function returns the position of the first occurrence of a substring in a string.

## Syntax:

LOCATE(substring, string, start)

**EXAMPLE:**

SELECT LOCATE("3", "W3Schools.com") AS MatchPosition;

**27.LENGTH:**

The LENGTH() function returns the length of a string (in bytes).

**Syntax:**

LENGTH(*string)*

**Example:**

SELECT LENGTH("SQL Tutorial") AS LengthOfString;

**28.INSTR:**

The INSTR() function returns the position of the first occurrence of a string in another string.

**Syntax:**

INSTR(*string1*, *string2*)

**Example:**

SELECT INSTR("W3Schools.com", "3") AS MatchPosition;

**29.FORMAT:**

The FORMAT() function formats a value with the specified format.

## Syntax:

FORMAT(value, format, culture)

**Example:**

SELECT FORMAT(250500.5634, 3);

**30.CONCAT:**

The CONCAT() function adds two or more strings together.

**Syntax:**

CONCAT(string1, string2, ...., string\_n)

**Example:**

SELECT CONCAT("SQL ", "Tutorial ", "is ", "Learning ", "Platform!") AS ConcatenatedString;

**31.DESC and ASC :**

ASC: to sort the data in ascending order. DESC: to sort the data in descending order

**Example:**

SELECT \* FROM Product

ORDER BY Price DESC,ProductID ASC ;

**32.ASCII:**

The ASCII() function returns the ASCII value for the specific character.

**Syntax:**

ASCII(*character*)

**Example:**

SELECT Price, ASCII(Price) AS NumCodeOfFirstChar

FROM Product;

**33.POWER:**

The POWER() function returns the value of a number raised to the power of another number.

**Syntax:**

POWER(*a*, *b*)

**Example:**

SELECT POWER(4, 2);

**34.RADIAN:**

The RADIANS() function converts a degree value into radians.

**Syntax:**

RADIANS(*number*)

**Example:**

SELECT RADIANS(180);

**35.ROUND:**

The ROUND() function rounds a number to a specified number of decimal places.

## Syntax:

ROUND(number, decimals, operation)

**Example:**

SELECT ROUND(135.375, 2);

**36.SIGN:**

The SIGN() function returns the sign of a number.

**Syntax:**

SIGN(*number*)

**Example:**

SELECT SIGN(255.5);

**37.PI:**

The PI() function returns the value of PI.

## Syntax:

PI()

**Example:**

SELECT PI();

**38.MOD:**

The MOD() function returns the remainder of a number divided by another number.

**Syntax:**

MOD(*x*, *y*)

**Example:**

SELECT MOD(18, 4);

**39.LOG:**

The LOG() function returns the natural logarithm of a specified number, or the logarithm of the number to the specified base.

## Syntax:

LOG(number, base) -- Syntax for SQL Server

**Example:**

SELECT LOG(2);

**40.GREATEST:**

The GREATEST() function returns the greatest value of the list of arguments.

## Syntax:

GREATEST(arg1, arg2, arg3, ...)

**Example:**

SELECT GREATEST(3, 12, 34, 8, 25);

**41.FLOOR:**

The FLOOR() function returns the largest integer value that is smaller than or equal to a number.

## Syntax:

FLOOR(number)

**Example:**

SELECT FLOOR(25.75);

**42.Divide:**

The SQL divide ( / ) operator is used to divide one expressions or numbers by another.

**Example:**

SELECT 10 DIV 5;

**43.DEGREE:**

The DEGREES() function converts a value in radians to degrees.

## Syntax:

DEGREES(number)

**Example:**

SELECT DEGREES(1.5);

**44.CEILING :**

The CEILING() function returns the smallest integer value that is larger than or equal to a number.

## Syntax:

CEILING(number)

**Example:**

SELECT CEILING(25.75);

**45.CURRENT\_DATE:**

The CURRENT\_DATE() function returns the current date.

## Syntax:

CURRENT\_DATE()

**Example:**

SELECT CURRENT\_DATE();

**46.CURRENT\_TIME:**

The CURRENT\_TIME() function returns the current time.

## Syntax:

CURRENT\_TIME()

**Example:**

SELECT CURRENT\_TIME();

**47.CURRENT\_TIMESTAMP:**

The CURRENT\_TIMESTAMP() function returns the current date and time.

**Syntax:**

CURRENT\_TIMESTAMP()

**Example:**

SELECT CURRENT\_TIMESTAMP();

## 48.DAY:

The DAY() function returns the day of the month for a given date (a number from 1 to 31).

## Syntax:

DAY(date)

**Example:**

SELECT DAY("2017-06-15");

**49.DAYNAME:**

The DAYNAME() function returns the weekday name for a given date.

**Syntax:**

DAYNAME(*date*)

**Example:**

SELECT DAYNAME("2017-06-15");

**50.EXTRACT:**

The EXTRACT() function extracts a part from a given date.

**Syntax:**

EXTRACT(*part* FROM *date*)

**Example:**

SELECT EXTRACT(MONTH FROM "2017-06-15");

**51.HOUR:**

The HOUR() function returns the hour part for a given date (from 0 to 838).

**Syntax:**

HOUR(*datetime*)

**Example:**

SELECT HOUR("2017-06-20 09:34:00");

**52.BINARY:**

The BINARY function converts a value to a binary string.

## Syntax:

BINARY value

**Example:**

SELECT BINARY "W3Schools.com";

**53.LIKE:**

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

### **Syntax**

SELECT column1, column2, ...  
FROM table\_name  
WHERE columnN LIKE pattern;

**Example:**

SELECT \* FROM Product

WHERE ProductName LIKE 's%';

**54.Wildcard:**

A wildcard character is used to substitute one or more characters in a string.

**Example:**

SELECT \* FROM Customers  
WHERE City LIKE 'ber%';

## 55.Aliases:

SQL aliases are used to give a table, or a column in a table, a temporary name.

Aliases are often used to make column names more readable.

An alias only exists for the duration of that query.

An alias is created with the AS keyword.

### **Syntax:**

SELECT column\_name AS alias\_name  
FROM table\_name;

**Example:**

SELECT ProductID AS ID, ProductName AS Product  
FROM Product;