SQL(STRUCTURED QUERY LANGUAGE)

BY

SHERIN FIRUTHUBA.S

ENROLLMENT NUMBER:EBEONO722634429

BATCH NO :2022 - 7688

SQL :

i. SQL stands for Structured Query Language.
ii. SQL is used to perform operations on the records stored in the database such as updating records, deleting records, creating and modifying tables, views, etc.
iii. SQL is required to create new databases, tables and views and to insert ,update ,delete and retrieve data from a database.

DATABASE :

A database is an organized collection of related information.

It is a systematic collections of data and its supports storage and manipulation of data.

SQL SYNTAX & COMMANDS :

SQL is not case sensitive. Generally, SQL keywords are written in uppercase.

Some of the **IMPORTANT SQL COMMANDS**:

1. **SELECT**: it extracts data from a database.
2. **UPDATE**: it updates data in database.
3. **DELETE:** it deletes data from database.
4. **CREATE TABLE**: it creates a new table.
5. **ALTER TABLE**: it is used to modify the table.
6. **DROP TABLE**: it deletes a table.
7. **CREATE DATABASE**: it creates a new database.
8. **ALTER DATABASE**: It is used to modify a database
9. **INSERT INTO**: it inserts new data into a database.
10. **DROP INDEX**: it deletes an index.

SQL COMMANDS

SQL commands are mainly classified into five categories:

1) DDL
2) DML
3) DRL
4) DCL
5) TCL

1.DDL(Data Definition Language) :

DDL or Data Definition Language actually it create a table, add, rename consists of the SQL commands that can be used to define the database schema(outlet).

Examples of DDL commands:

i. CREATE
ii. DROP
iii. ALTER
iv. TRUNCATE
v. RENAME
vi. COMMENT

i. CREATE

Is used to create the database or its objects.There are two CREATE statements available in SQL:

1. CREATE DATABASE
2. CREATE TABLE

1. **CREATE DATABASE :**

A Database is defined as a structured set of data.

The CREATE DATABASE statement is used to create a new database in SQL.

SYNTAX:

CREATE DATABASE database_name;

database_name: name of the database.

EXAMPLE

```
CREATE DATABASE my_database;
```

ii. **CREATE TABLE**

The CREATE TABLE statement is used to create a table in SQL. SQL about the names of the columns, type of data to be stored in columns, size of the data etc. To use CREATE TABLE statement to create tables in SQL.

SYNTAX:

```
CREATE TABLE table_name
(
column1 data_type(size),
column2 data_type(size),
column3 data_type(size),
....
);
```

EXAMPLE:

```
CREATE TABLE Students
(
ROLL_NO int(3),
NAME varchar(20),
SUBJECT varchar(20),
);
```

2) **DROP**

It is used to delete objects from the database.

SYNTAX

DROP TABLE table_name;

EXAMPLES:

SQL> DROP DATABASE Customer;

3) ALTER

It is used to alter the structure of the database. ALTER TABLE is used to add, delete/drop or modify columns in the existing table. It is also used to add and drop various constraints on the existing table.

i.ALTER TABLE – ADD

ADD is used to add columns into the existing table.

SYNTAX:

```
ALTER TABLE table_name
ADD (Columnname_1 datatype,
Columnname_2 datatype,
…
Columnname_n datatype);
```

4) TRUNCATE

It is used to remove all records from a table, including all spaces allocated for the records are removed.

SYNTAX:

TRUNCATE TABLE table_name;

table_name: Name of the table to be truncated.

5) RENAME Command

It is used to rename an object existing in the database.

SYNTAX:

```
ALTER TABLE table_name
RENAME TO new_table_name;
```

2.DQL (Data Query Language) :

The SQL commands that deals with to get the result .It is also called as DRL(Data Retrival Language).

Examples of DQL:

1)SELECT – is used to retrieve data from the a database.

SYNTAX:

1. SELECT * FROM table_name ;
2. SELECT STU_ID ,AGE FROM table_name;
3. Select * FROM table_name where ID =14;(Condition)

3.DML(Data Manipulation Language) :

The SQL commands that deals with the manipulation of data present in database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

i. **INSERT** – It is used to insert data into a table.

There are two ways of using INSERT INTO statement for inserting rows:

SYNTAX:

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

EXAMPLE:

```
INSERT into Studentlist(reg_no int,name varchar(20),subject
varchar (10),marks int)VALUES(1,’Akash’,’Maths’,89));
```

ii. **UPDATE** – It is used to update existing data within a table.

SYNTAX

UPDATE table_name SET column1 = value1, column2 = value2,...

WHERE condition;

table_name: name of the table

column1: name of first , second, third column....

value1: new value for first, second, third column....

condition: condition

EXAMPLE

UPDATE STUDENT SET MARKS =100;

(OR)

UPDATE STUDENT SET MARKS =92 *

where fname ='shafiq';

iii. **DELETE** – is used to delete records from a database table.

SYNTAX:

DELETE FROM table_name WHERE some_condition;

table_name: name of the table

some_condition: condition to choose particular record.

4.DCL(Data Control Language) :

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

Examples of DCL commands:

1)GRANT-gives user's access privileges to database.

2)REVOKE-withdraw user’s access privileges given by using the GRANT command.

• **Allow a User to create session**

When we create a user in SQL, it is not even allowed to login and create a session until and unless proper permissions/priviliges are granted to the user.

Following command can be used to grant the session creating priviliges.

GRANT CREATE SESSION TO username;

5.TCL(Transaction Control Language) :

TCL commands deals with the transaction within the database.

Examples of TCL commands:

1)COMMIT– commits a Transaction.

When we use any DML command like INSERT, UPDATE or DELETE, the changes made by these commands are not permanent, until the current session is closed, the changes made by these commands can be rolled back.

To avoid that, we use the COMMIT command to mark the changes as permanent.

Following is commit command's **SYNTAX**,

COMMIT;

2)ROLLBACK– rollbacks a transaction in case of any error occurs.

This command restores the database to last commited state. It is also used with SAVEPOINT command to jump to a savepoint in an ongoing transaction.

If we have used the UPDATE command to make some changes into the database, and realise that those changes were not required, then we can use the ROLLBACK command to rollback those changes, if they were not commited using the COMMIT command.

Following is rollback command's **SYNTAX,**

ROLLBACK TO savepoint_name;

3)SAVEPOINT–sets a savepoint within a transaction.

SAVEPOINT command is used to temporarily save a transaction so that you can rollback to that point whenever required.

Following is savepoint command's **SYNTAX,**

SAVEPOINT savepoint

EXAMPLE:

```
INSERT INTO class VALUES(5, 'Rahul');
COMMIT;
UPDATE class SET name = 'Abhijit' WHERE id = '5';
SAVEPOINT A;
INSERT INTO class VALUES(6, 'Chris');
SAVEPOINT B;
INSERT INTO class VALUES(7, 'Bravo');
SAVEPOINT C;
SELECT * FROM class;
```

SQL JOIN:

Joins

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

Types of SQL JOINS :

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

 LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table.
 RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table.
 FULL (OUTER) JOIN: Return all records when there is a match in either left or right table .

INNER JOIN: The INNER JOIN keyword selects all rows from both the tables as long as the condition satisfies. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies .

SYNTAX

```
SELECT table1.column_name()
FROM table1
INNER JOIN table2
ON table1.matching_column = table2.column;
```

EXAMPLE

```
SELECT COLLEGECourse.COURSE_ID, Student.NAME,
Student.AGE FROM Student
INNER JOIN StudentCourse
ON Student.ROLL_NO = StudentCourse.ROLL_NO;
```

LEFT JOIN: This join returns all the rows of the table on the left side of the join and matching rows for the table on the right side of join. The rows for which there is no matching row on right side, the result-set will contain null. LEFT JOIN is also known as LEFT OUTER JOIN.

SYNTAX:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
LEFT JOIN table2
ON table1.matching_column = table2.matching_column;
```

EXAMPLE:

```
SELECT Student.NAME,StudentSubject.Student_ID
FROM Student
LEFT JOIN Student Subject
ON StudentSubject.ROLL_NO = Student.ROLL_NO;
```

RIGHT JOIN: RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of join.. RIGHT JOIN is also known as RIGHT OUTER JOIN.

SYNTAX:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
RIGHT JOIN table2
ON table1.matching_column = table2.matching_column;
```

EXAMPLE

```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student
RIGHT JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

FULL JOIN: FULL JOIN creates the result-set by combining result of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both the tables. The rows for which there is no matching, the result-set will contain NULL values.

SYNTAX:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
FULL JOIN table2
ON table1.matching_column = table2.matching_column;
```

EXAMPLE

```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student
FULL JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

SQL FUNCTION

1. CHAR() Function:

Return the character based on the number code

SYNTAX:

CHAR(*code*)

EXAMPLE:

```
SELECT CHAR(84) AS CodeToCharacter;
```

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

SYNTAX

CONCAT (string1,string2,….n)

EXAMPLE

SELECT CONCAT('SQL', ' is', ' fun!');

3.SELECT ALIASES: IT IS USED TO RENAME THE COLUMN TEMPORIRLY.

SYNTAX:

SELECT COLUMN1 AS NEW COLUMN1, COLUMN2 AS NEW COLUMN1 FROM TABLENAME;

EXAMPLE:

SELECT ENAME AS EMPLOYEENAME, EMAIL AS EMAILID FROM EMPLOYEE;

4.AND OPERATION : AND OPERATION IS USED TO BOTH CONDITION IS TRUE.

SYNTAX:

SELECT COLUMN1, COLUMN2, ... FROM TABLE_NAME

WHERE CONDITION1 AND CONDITION2 AND

CONDITION3 ...;

EXAMPLE:

SELECT * FROM EMPLOYEE

WHERE EMP_ID=23 AND SALARY >15000;

5.OR OPERATION : OR OPERATION IS USED TO ANY ONE CONDITION IS TRUE.

SYNTAX:

SELECT COLUMN1, COLUMN2, ... FROM TABLE_NAME
WHERE CONDITION1 OR CONDITION2 OR CONDITION3 ...;

EXAMPLE:

SELECT ENAME,SALARY FROM EMPLOYEE
WHERE AGE =24 OR SALARY >50000;

6. BETWEEN:

SYNTAX:

SELECT COLUMN_NAME(S) FROM TABLE_NAME WHERE COLUMN_NAME BETWEEN VALUE1 AND VALUE2;

EXAMPLE:

SELECT ENAME,AGE FROM EMPLOYEE
WHERE AGE BETWEEN 13 AND 25;

7. **COUNT** :it is used to give row count.

SYNTAX:

SELECT COUNT(SNO) FROM EMPLOYEE;

8 CONCAT

SYNTAX:

SELECT CONCAT('SQL IS INTERSTING')AS CONCAT

EXAMPLE:

SELECT CONCAT('SHERIN ', 'FIRUTHUBA') AS CONCAT;

9.DATA LENGTH:

EXAMPLE:

SELECT length('Have a nice day') AS LENGTH;

10.TRIMMED STRING

SYNTAX:

SELECT TRIM(“ ”)

EXAMPLE:

SELECT TRIM(' SOFTWARE TEST ENGINEER ! ') AS TrimmedString;

11. ASCENDING

SYNTAX:

SELECT * FROM TABLENAME
ORDER BY COLUMN ASC;

EXAMPLE:

SELECT * FROM EMPLOYEE
ORDER BY ENAME ASC;

12.DECENDING

SYNTAX:

SELECT * FROM TABLENAME
ORDER BY COLUMN DESC;

EXAMPLE:

```
SELECT * FROM EMPLOYEE
ORDER BY ENAME DESC;
```

13.MIN VALUE :

SYNTAX:

```
SELECT MIN(COLUMN) AS SMALLESTSALARY
FROM EMPLOYEE;
```

EXAMPLE:

```
SELECT MIN(SALARY) AS SMALLESTSALARY
FROM EMPLOYEE;
```

14.MAX VALUE

SYNTAX:

```
SELECT MAX(COLUMN) AS LARGESTSALARY
FROM EMPLOYEE;
```

EXAMPLE:

```
SELECT MAX(SALARY) AS LARGESALARY
FROM EMPLOYEE;
```

15. AVERAGE:

SYNTAX:

```
SELECT AVG(COLUMN_NAME) FROM TABLE_NAME
WHERE CONDITION;
```

EXAMPLE:

```
SELECT AVG(SALARY)
```

```
FROM EMPLOYEE
WHERE AGE <=22;
```

16.NULL FUNCTION:

SYNTAX:

-- WITHOUT IS NULL FUNCTION --

```
SELECT COLUMN1,COMN2 CONDITION
FROM TABLENAME
WHERE CONDITION;
```

EXAMPLE:

```
SELECT EName, AGE* (SNO )+ (SALARY)
FROM EMPLOYEE
WHERE AGE =23;
```

17. ISNULL FUNCTION:

SYNTAX:

```
SELECT COLUMN1,ISNULL COLUMN2 CONDITION
FROM TABLENAME
WHERE CONDITION;
```

EXAMPLE:

```
-- ISNULL FUNCTION -- (ADD 1)
SELECT EName, AGE* (SNO )+ ISNULL(SALARY)
FROM EMPLOYEE
WHERE AGE =23;
```