Structured Query Language

STUDY NOTES

- **Benefits of using a SQL are:** Redundancy can be controlled, Inconsistency can be avoided, Data can be shared, Security restrictions can be applied.
- **Create database:** create database statement is used to create a new database.
  Syntax:
  ```
  create database <database name>;
  ```
  Use: The USE Statement is used to select a database and perform SQL operations on that database.
  Syntax:
  ```
  use <database name>;
  ```
- **drop database:** Drops all tables in the database and deletes the database.
  Syntax:
  ```
  drop database <database name>;
  ```
- **select database():** To know which database is in use currently.
- **Create table:** The CREATE TABLE statement allows you to create a new table in a database.
  Syntax:
  ```
  CREATE TABLE table_name (
     column1 datatype,
     column2 datatype,
     column3 datatype,
     ....
  );
  ```
- The following example creates a table called "Student" that contains five columns: Rollno, Name, Age, Marks, and Gender.
- mysql>create table Student(Rollno int(3),Name char(20),Age int(2),Marks int(3),Gender char(1))
- Describe or Desc: DESCRIBE command is used to show the structure of table like column names and constraints on column name. The DESC is the short form of DESCRIBE command.
- **INSERT INTO:** The INSERT INTO statement is used to insert new records in a table. We can write insert into statement in two ways:
  1. **When insert data only in specified columns:** Specify the column's name in insert into statement when you want to insert values in specified columns in table.
     Syntax:
     ```
     insert into <table name>(<columns name>) values(<values>)
     ```
     for example if a student table having fields(Rollno,Name,Age,Marks) and the following insert into statement can be used if you want to insert a record with rollno and age only
     insert into student (rollno,age) values(5,18)
     NULL value will be stored automatically in rest of the fields i.e. Name,Marks
  2. **When adding values in all columns:** If you want to add values for all the columns of the table then no need to specify the column names in the SQL query. But make sure the order of the values must be in the

same order as the columns in the table. The syntax would be as follows:

```
insert into <table name> values(value1,value2,.....)
```

the following insert into statement adding new record in student table for all fields

```
insert into student values(6,'aman',17,89)
```

- **Select statement:** The SELECT statement is used to select data from one or more tables.

```
Select * from Student;
or
Select all from Student;
```

The above statement selects all the fields from the student table. To select specified field(s), give fields name instead of * or all.

- **Where clause:** The WHERE clause is used to extract only those records that fulfill a specified condition.

Syntax:

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

- **ORDER BY:** The SQL ORDER BY clause is used to sort the data in ascending or descending order according to one or more columns.

The following shows the syntax of the ORDER BY clause:

```
SELECT
   select_list
FROM
   table_name
ORDER BY
   sort_expression [ASC | DESC];
```

- **Delete:** The SQL DELETE statement is used to delete one or more records from a table.

Syntax:

```
delete from <tablename> [where condition]
```

where condition is optional and used to delete specific record(s), If no conditions are provided, all records in the table will be deleted.

- **Drop table:** The drop table command is used to delete a table (physically) as well as all records.

Syntax:

```
drop table <tablename>;
```

- **LIKE:** The SQL LIKE clause is used to search for a specified pattern using wildcard operators. The LIKE condition is used in the WHERE clause of a SELECT, INSERT, UPDATE or DELETE statement.
- **NULL:** Null or NULL is a special marker used to indicate that a data value does not exist in the database. A field with a NULL value is a field with no value. The IS NULL condition is used in SQL to test for a NULL value.
- **DISTINCT:** The SQL DISTINCT keyword is used to eliminate all the duplicate records and fetching only unique records.
- **BETWEEN:** BETWEEN condition is used to retrieve values within a range in a SELECT, INSERT, UPDATE, or DELETE statement. The values can be numbers, text, or dates. BETWEEN is a shorthand for >= AND <=. BETWEEN is inclusive, i.e. begin and end values are included.

Syntax:

```
expression BETWEEN begin value AND end value;
```

For example the following statement displays the records from students table those students have got marks between 30 and 70(both inclusive).

```
Select * from Student where Marks between 30 and 70;
or
Select * from Student where Marks>=30 and Marks<=70;
```

- **OR OPERATOR:** The Logical OR operator in MySQL compares two conditions and returns TRUE if either of the conditions is TRUE and returns FALSE when both are FALSE.
- **UPDATE():** The UPDATE command in SQL is used to modify or change the existing records in a table. If you want to update a particular value, then you can use where condition otherwise changes will take effect in all records.
  Syntax:
  UPDATE table_name SET col1=val1, col2=val2…[Where condition];
- **Alter table:** The MySQL ALTER TABLE statement is used to add, modify, rename or drop/delete columns in a table. The MySQL ALTER TABLE statement is also used to rename a table.
- **Drop column:** SQL has ALTER TABLE DROP COLUMN command for removing columns from an existing table.
  Syntax:
  ```
  ALTER TABLE table_name DROP COLUMN column_name;
  ```
  For example: The following command will remove Hobbies column from Student table
  Alter table Student drop Hobbies
  Syntax:
  ```
  ALTER TABLE "table_name" MODIFY "column_name" "New Data Type";
  ```
- **GROUP BY:** The Group By statement is used for organizing similar data into groups. The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

  The SELECT statement is used with the GROUP BY clause in the SQL query.

  WHERE clause is placed before the GROUP BY clause in SQL.

  ORDER BY clause is placed after the GROUP BY clause in SQL.
- **Having:** The SQL HAVING clause is used in combination with the GROUP BY clause to restrict the groups of returned rows to only those whose condition is TRUE.
- **JOIN on two tables:** A JOIN clause is used to combine rows from two or more tables, based on a related column between them. Usually, such an attribute is the primary key in one table and foreign key in another table Join joins table in such a way that it only shows those results which matches the condition that is given and hide others. The structure of Inner Join queries are:
  ```
  SELECT column_name(s) FROM table1 INNER JOIN table2 ON table1.column_name=table2.column_name;
  ```
  Inner Join and simple join both are same. You can also write your query like this:
  ```
  SELECT column_name(s) FROM table1, table2 where table1.column_name=table2.column_name;
  ```
- **Functions in MySQL:** A function is used to perform some particular task and it returns zero or more values as a result. SQL functions are categorised as Single row functions and Aggregate functions.
  - **(i) Single Row Functions:** These are also known as Scalar functions. Single row functions works on a single value/row and returns one value for each row. It accepts one or more arguments. Examples : round(x), concat(str1,str2,….)
  - **(ii) Aggregate Funcions:** These are also known as Multiple row functions. Aggregate functions works on multiple values in a single column and return one value. It accepts only one arguments. Examples : Max(), Min(), Avg(), Sum(), Count()
- **Single Row Functins:**
  - **(i) Numeric Functions:**
    - ❖ **power():** The POWER function returns m raised to the n^{th} power.
      Syntax: `POWER( m, n )`
      where
      m:The base used in the calculation. n:The exponent used in the calculation
      Example: `select pow(2,3) will return 8`
    - ❖ **mod():** The MySQL MOD function returns the remainder of n divided by m.
      The syntax for the MOD function in MySQL is: MOD(n, m)
      where

n:The value that will be divided by m. m:The value that will be divided into n
for example `mod(10,3)` will return 1.

(ii) String functions: String functions can perform various operations on alphanumeric data like change the case (uppercase to lowercase or vice versa), extract the left, right or substring, calculate the length of string etc.

❖ **mid():** is used to extract a substring from a string.
Syntax:
mid(string, start, length)
where
string: the string to extract start: starting position
length: number of characters to extract. If omitted number of characters to be extracted upto end of string.
Note: The MID() function equals the SUBSTR() and SUBSTRING() functions
for example `Select mid('HONESTY WINS',3,4)` will return NEST

❖ **right():** the right() function extracts a number of characters from a string from right.
Syntax:
`right(str,n)`
str: string to be extracted
n: length of character to be extracted from right
for example: `Select right("My School",2)` will return ol

❖ **left():** the left() function extracts a number of characters from a string from left.
Syntax:
`left(str,n)`
str: string to be extracted
n: length of character to be extracted from left
example:
(i) `Select left("My School",2)` (ii) `Select left("My School",5)`
output:
(i) My (ii) My Sc

❖ **length():** The MySQL LENGTH function returns the length of the specified string (in bytes).
Syntax:
`LENGTH( string )`
Parameters or Arguments
string: The string to return the length for.
for example: `select length("Amit Kumar")` will return 10

❖ **trim():** This function removes the blank spaces from the head and tail of the given string.
Syntax:
TRIM(str)
Parameter:
str: string to be trimmed

❖ **ltrim():** This function removes the blank spaces from the head of the given string.
Syntax:
`LTRIM(str)`
Parameter:
str: The string to remove leading spaces from

❖ **rtrim():** This function removes the blank spaces from the tail of the given string.
Syntax
`RTRIM(str)`
Parameter:
str: The string to remove trailing spaces from

❖ **Upper() or Ucase():** UPPER function converts all the letters in a string into uppercase
Syntax:
`upper(string)`
where
string: The string to convert in uppercase
for example: `select upper("computer")` will return COMPUTER

(iii) Date and Time Functions : These functions are used to display the current date, extracting each element of a date (day, month and year), displaying day of the week, etc.

❖ **NOW():** MySQL NOW() returns the value of current date and time in 'YYYY-MM-DD HH:MM:SS' format. There is no any parameter or argument for the NOW() function.
Syntax:
`Now()`
for example the statement SELECT NOW() will return current date and time i.e. 2022-04-22 15:08:15

❖ **Date():** The date() function is used to get the date part from given date/datetime.
Syntax:
`Date(expression)`
where expression is valid date/datetime value.

❖ **Dayname(date):** The DAYNAME function returns the name of the weekday (Sunday, Monday, Tuesday, etc.) for given date value.
Syntax:
`Dayname(date)`
where date is the date to extract the weekday name from
for example the statement `SELECT DAYNAME(DATE(NOW()))` will return 'Saturday', assuming current date is 2022-01-22.

❖ **day():** It returns the day part from the date.For example the statement `select day("2021-06-15")` will return 15

❖ **Aggregate Functions:**
MAX(): The MAX() function returns the maximum value in a set of values.
SUM(): The MySQL sum() function is used to return the total summed value of an expression.
AVG(): The AVG() function returns the average value of an expression.(NULL values are ignored)

❖ **COUNT() :**
(i) The COUNT(*) function returns the number of rows in a result including duplicate, non-NULL and NULL rows.
(ii) COUNT(expression):The COUNT(expression) returns the number of rows that do not contain NULL values as the result of the expression.