CHAPTER – 8 : INTRODUCTION TO STRUCTURED QUERY LANGUAGE(SQL)

MYSQL: MySQL is a Relational Database Management System. It was originally founded and developed in Sweden by David Axmark, Allan Larsson and Michael Widenius.

Characteristics of MySQL:

1. **Cost:** It is released under an open–source license and hence required no cost or payment for its usage.
2. **Speed:** It has superior speed. It is easy to use and is reliable.
3. **Ease of Use:** It is a simple database system. It can be easily managed from the command line. Several graphical interfaces are also available.
4. **Query Language Support:** Standard SQL commands are understandable to MySQL.
5. **Data Types:** It supports many data types to support different types of data. It also supports fixed–length and variable–length records.
6. **Security:** It offers privilege and password system that is very flexible and secure. Passwords are secure because all password traffic is encrypted at the time of connecting to a server.
7. **Scalability and Limits:** It can handle large databases. Some real life MySQL databases contain 50 million records, some have up to 60,000 tables and about 500,00,00,000 rows.
8. **Connectivity:** Clients can connect to MySQL server easily

CREATING AND USING A DATABASE:

To create a database the CREATE DATABASE command is used, as follows.

Syntax : CREATE DATABASE <Database_Name>;

Ex : CREATE DATABASE School;

The above statement creates a database with the name School. To work with this database this database should be opened using USE statement, as follows

Syntax : USE <Database_Name>;

Ex : USE School;

VIEWING TABLES OF A DATABASE:

To see the tables present in a database, the statement

SHOW TABLES;

can be used.

Data Types:

Commonly used data types available in MySQL are,

INT	A normal–sized integer that can be signed or unsigned. If signed, the allowable range is from –2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. Each INT value occupies 4 bytes of storage
DECIMAL(M, D)	Represents a floating point number. M is the total number of digits, and D is the number of digits after decimal point.
FLOAT	Holds numbers with decimal points. Each float value occupies 4 bytes
DATE	A date in YYYY–MM–DD format, between 1000–01–01 and 9999–12–31
CHAR(M)	A fixed–length string between 0 and 255 characters in length. M is the length. Providing M is optional and its default value is 1
VARCHAR(M)	A variable–length string between 0 and 65535 characters in length. Providing M is compulsory

Difference between CHAR and VARCHAR:

The difference between CHAR and VARCHAR is that CHAR is of fixed–length and VARCHAR is of variable length.

When a column is given data type as CHAR(n), then all values stored in that column have this length, i.e. n bytes. If a value is shorter than this length n then blanks are added, but the size of the value remains n bytes.

VARCHAR specifies a variable length string. When a column is given data type as VARCHAR(n), then the maximum size of a value is n bytes. Each value that is stored in this column stores exactly as given, no blanks are added if the length is shorter than maximum length. However, if the maximum length is exceeded than an error message will be displayed.

Constraints: Constraints are certain types of restrictions on the data values that an attribute can have. They are used to ensure the accuracy and reliability of data.

Constraint	Description
NOT NULL	Ensures that a column cannot have NULL values where NULL means missing/ unknown/not applicable value.
UNIQUE	Ensures that all the values in a column are distinct / unique.
DEFAULT	A default value specified for the column if no value is provided.
PRIMARY KEY	The column which can uniquely identify each row or record in a table.
FOREIGN KEY	The column which refers to value of an attribute defined as primary key in another table.

CREATING A TABLE:

A table in the database can be created using CREATE TABLE command, as follows

Syntax : CREATE TABLE <TableName> (<ColumnName1> <DataType1>, <ColumnName2> <DataType2>, ……., <ColumnNameN> <DataTypeN>);

Ex : CREATE TABLE STUDENT (RollNo INTEGER, Name VARCHAR(15), Class CHAR(3), DOB DATE);

- SQL commands are not case sensitive. For example, the command CREATE can also be provided as Create or create etc
- While giving name to a table it should be relevant to the data that it holds

INSERTING DATA INTO A TABLE:

INSERT INTO command can be used to insert rows of data into a table. Its usage is as follows

Syntax : INSERT INTO <TableName> (<ColumnName1>, <ColumnName2>, ….., <ColumnNameN>) VALUES (<Value1>, <Value2>, <Value3>, ……<ValueN>);

Ex : INSERT INTO STUDENT (RollNo, Name, Class, DOB) VALUES(154, ‘Rajesh’, ‘11c’, ‘1996–04–05’);

Ex : INSERT INTO STUDENT (RollNo, Name, Class) VALUES (189, ‘Kiran’, ‘12c’);

(In this case the value NULL will be assigned to the field DOB)

However while providing data for all the column elements the column names need not be provided.

Syntax : INSERT INTO <TableName> VALUES (<Value1>, <Value2>, <Value3>, ……<ValueN>);

Ex : INSERT INTO STUDENT VALUES(154, ‘Rajesh’, ‘11c’, ‘1996–04–05’);

RETRIEVING (DISPLAYING) DATA OF A TABLE:

The SELECT command is used to display data of a table. It is also possible to display the filtered data from the table.

To Display all the rows and columns of Table:

Syntax : SELECT * FROM <TableName>;

Ex : SELECT * FROM Student;

To Display selected Columns and all Rows:

Syntax : SELECT <ColumnName1>, <ColumnName2>, …. FROM <TableName>;

Ex : SELECT Name, DOB FROM Student;

To Display selected Columns and selected Rows:

Syntax : SELECT <ColumnName1>, <ColumnName2>, …. FROM <TableName> WHERE Condition:

Ex : SELECT Name, DOB FROM Student WHERE Class = ‘11C’;

Eliminating Redundant Data with DISTINCT Keyword:

Syntax : SELECT DISTINCT <ColumnName1>, <ColumnName2>,… FROM <TableName>;

Ex : SELECT DISTINCT Class FROM Student;

DISPLAYING CURRENT DATABASE:

To display the name of the present working database, the following statement can be used

SELECT DATABASE( );

CATEGORIES OF SQL COMMANDS:

1. ***Data Definition Language (DDL) Commands:*** The SQL commands used for creating a database, deleting a database, providing keys such as primary key, foreign key etc on tables are known as DDL commands. A few examples for DDL commands in SQL are:
   a. CREATE DATABASE – Creates a new database
   b. CREATE TABLE – Creates a new table
   c. ALTER TABLE – Modifies a table
   d. DROP TABLE – Deletes a table

   These commands will work with table structure not on table data directly (indirectly may act on table data)

2. ***Data Manipulation Language (DML) Commands:*** The Query and Update commands on tables are referred as DML commands. A few examples for DML commands in SQL are:
   a. SELECT – Extracts data from a table
   b. UPDATE – Updates data in a table
   c. DELETE – Deletes data from a table
   d. INSERT INTO – Inserts new data into a table

3. ***Data Control Language (DCL) / Transaction Control Language (TCL) Commands:*** The commands which are used to control the access to databases and tables are referred as Data Control Language or Transaction Control Language commands. A few examples for TCL commands in SQL are:
   a. GRANT – To give rights of transaction to a user
   b. REVOKE – To take back rights of transaction from a user
   c. COMMIT – To make changes permanent
   d. ROLLBACK – To undo changes of transaction

VIEWING STRUCTURE OF A TABLE:

To see the structure of a table that is already created, the DESCRIBE or DESC command can be used as follows

Syntax : DESCRIBE <TableName>; (Or) DESC <TableName>;

Ex : DESC Student;

CHANGING STRUCTURE OF A TABLE USING ALTER TABLE COMMAND:

An existing table's structure can be changed by using ALTER TABLE command.

Adding an attribute to a Table:

Syntax : ALTER TABLE <TableName> ADD <ColumnName> DataType;
Ex : ALTER TABLE Student ADD Grade CHAR(1);

Removing an attribute of a Table:

Syntax : ALTER TABLE <TableName> DROP <ColumnName>;
Ex : ALTER TABLE Student DROP DOB;

Modifying datatype of an attribute of a Table:

Syntax : ALTER TABLE <TableName> MODIFY <ColumnName> <New_Definition>;
Ex : ALTER TABLE Student Class VARCHAR(4);

Add Primary Key constraint to a Table:

Syntax : ALTER TABLE <TableName> ADD PRIMARY KEY(ColumnName);
Ex : ALTER TABLE Student ADD PRIMARY KEY(RollNo);

Removing Primary Key from a table:

Syntax : ALTER TABLE <TableName> DROP PRIMARY KEY;
Ex : ALTER TABLE Student DROP PRIMARY KEY;

Adding Foreign Key constraint to a Table:

Syntax : ALTER TABLE <TableName> ADD FOREIGN KEY(ColumnName) REFERENCES <Referenced_TableName>(ColumnName);
Ex : ALTER TABLE Student ADD FOREIGN KEY(RollNo) REFERENCES Marks(RollNo);

Adding UNIQUE constraint to a Table:

Syntax : ALTER TABLE <TableName> ADD UNIQUE(ColumnName);
Ex : ALTER TABLE Student ADD UNIQUE(RollNo);

Adding DEFAULT constraint to a Table:

Syntax : ALTER TABLE <TableName> MODIFY ColumnName DATATYPE DEFAULT Default_Value;
Ex : ALTER TABLE Student MODIFY Gender VARCHAR(10) DEFAULT 'Male';

Adding NOT NULL constraint to a Table:

Syntax : ALTER TABLE <TableName> MODIFY ColumnName DATATYPE NOT NULL;
Ex : ALTER TABLE Student MODIFY Gender VARCHAR(10) NOT NULL;

USING ARITHMETIC OPERATORS WITH SELECT:

Arithmetic operators perform mathematical calculations. In SQL the following arithmetic operators are used

Operator	Meaning
+	Addition
–	Subtraction
*	Multiplication
/	Division
%	Modulus Division (Remainder Division)

Ex : SELECT Marks1+5 FROM Student;
Ex : SELECT 5*4+2;

USING COLUMN ALIAS:

When arithmetic expressions are used in SELECT statement the Column name will be the expression. For example, for the query

SELECT Marks1+5 FROM Student;

the column name of the output is Marks1+5.

This column name can be changed (aliased) as follows

SELECT Marks1+5 AS "Modified Marks" FROM Student;

Here the keyword AS is optional

PUTTING TEXT IN QUERY OUTPUT:

The text can be put in a query by providing as a string constant

Ex : SELECT RollNo, Name, 'has secured marks', Marks1 FROM Student;

RELATIONAL OPERATORS:

The relational operators used in MySQL are as follows

Operator	Meaning
<	Less Than
<=	Less Than or Equal to
>	Greater Than
>=	Greater Than or Equal to
=	Equal to
!= (Or) <>	Not Equal to

To Display all Columns and selected Rows:

Syntax : SELECT * FROM <TableName> WHERE Condition;

Ex : SELECT * FROM Student WHERE Class='11c';

LOGICAL OPERATORS:

The logical operators available in MySQL are

Operator	Meaning
AND	If both conditions are true output results to true
OR	If either condition is true output results to true
NOT	Makes the input True as False and vice–versa

CONDITION BASED ON RANGE (USING BETWEEN):

The BETWEEN operator is used for range search. It is used along with WHERE clause.

Ex: SELECT RollNo, Name, Marks1 FROM Student WHERE Marks1 BETWEEN 70 AND 80;

The above query will displays the RollNo, Name, Marks1 from the Student table whose value of Marks1 value will be between 70 and 80, both inclusive.

The NOT BETWEEN can be used to negate the output of BETWEEN

Ex: SELECT RollNo, Name, Marks1 FROM Student WHERE Marks1 NOT BETWEEN 70 AND 80;

The above query will displays the RollNo, Name, Marks1 from the Student table whose value of Marks1 values are not be between 70 and 80, both exclusive

CONDITION BASED ON A LIST (USING IN):

The IN operator select values that match any value in the given list of values. To display data of Students whose marks are 68 or 76 or 78, we can use the IN operator like this:

Ex: SELECT RollNo, Name, Marks1 FROM Student WHERE Marks1 IN (68, 76, 78);

Ex: SELECT * FROM Employee WHERE State IN ('DELHI', 'MUMBAI', 'UP');

The NOT IN operator can be used to negate the output of IN operator

Ex: SELECT RollNo, Name, Marks1 FROM Student WHERE Marks1 NOT IN (68, 76, 78);

Ex: SELECT * FROM Employee WHERE State NOT IN ('DELHI', 'MUMBAI', 'UP');

PATTERN MATCHING (USING LIKE CLAUSE):

LIKE clause is used to fetch data which matches the specified pattern from a table. The LIKE clause tells the DBMS that we won't be doing a strict comparison like = or < or > but we will be using wildcards in our comparison.

Syntax: SELECT <column name>, [<column name>…] WHERE <column name> LIKE Pattern;

MySQL has % and _ wild card characters. The percent (%) symbol is used to represent any sequence of zero or more characters. The underscore (_) symbol is used to represent a single character.

Ex: SELECT * FROM Student WHERE Name LIKE '%Sen';

Ex: SELECT * FROM Student WHERE Name LIKE 'G%';

Ex: SELECT * FROM Student WHERE Name LIKE 'G%b';

Ex: SELECT * FROM Student WHERE Name LIKE '%Sen%';

Ex: SELECT * FROM Student WHERE Name LIKE 'A_ _ _ _ Ali';

A few more general examples are,

- 'Am%' matches any string starting with Am.
- '%Singh%' matches any string containing 'Singh'
- '%a' matches any string ending with 'a'
- '_ _ _' matches any string that is exactly 3 characters long.
- '_ _ %' matches any string that has at least 2 characters.
- '_ _ _ g' matches any string that is 4 characters long with any 3 characters in the beginning but 'g' as the 4th character.

The keyword NOT LIKE is used to select the rows that do not match the specified pattern. To display rows from the table Student that have names not starting with 'G', she enters:

Ex: SELECT * FROM Student WHERE Name NOT LIKE 'G%';

PRECEDENCE OF OPERATORS:

Precedence is the order in which different operators are evaluated in the same expression. When evaluating an expression containing multiple operators, operators with higher precedence are evaluated before evaluating those with lower precedence. **Operators with equal precedence are evaluated from left to right within the expression**. Parenthesis can be used to change the preference of an operator. Various operators in descending order of precedence (top to bottom) are listed below:

!

– (unary minus)

^

*, /, DIV, %, MOD

-, +

=, <=>, >=, >, <=, <, <>, !=, IS, LIKE, IN

BETWEEN,

NOT

&&, AND

||, OR

NULL:

NULL means a value that is unavailable, unassigned, unknown or inapplicable. NULL is not the same as zero or a space or any other character. In a table NULL is searched for using IS NULL keywords.

Ex: SELECT * FROM Student WHERE Name IS NULL;
Ex: SELECT * FROM Employee WHERE Commission IS NULL;

NOT NULL values in a table can be searched using IS NOT NULL.

Ex: SELECT * FROM Employee WHERE Commission IS NOT NULL;

SORTING THE RESULTS (USING ORDER BY):

The result obtained using SELECT statement is displayed in the order in which the rows were entered in the table using the INSERT INTO statement. The results of the SELECT statement can be displayed in the ascending or descending values of a single column or multiple columns using ORDER BY clause.

Syntax: SELECT <column name>, [<column name>…] [WHERE <Condition list>] ORDER BY <column name>;

Ex: SELECT * FROM Student ORDER BY Marks1;

Ex: SELECT * FROM Student ORDER BY Name;

To display data in descending order, DESC keyword is used in ORDER BY clause. However it is not necessary to specify ASC for ascending order as it is the default order.

Ex: SELECT * FROM Student ORDER BY Marks1 DESC;

SORTING ON COLUMN ALIAS:

If a Column alias is defined on a column, we can use it for displaying rows in an ascending or descending order using ORDER BY clause:

Ex: SELECT Name, Marks1 AS Total FROM Student ORDER BY Total;

UPDATE STATEMENT:

The UPDATE statement is used to update the data of the table. WHERE clause is also applicable to this statement.

Syntax: UPDATE <table_name> SET <column name> = <value>, [<column name> = <value>, …] [WHERE <condn>];

Ex: UPDATE Student SET Marks1 = 94;

The above statement sets the Column Marks1 value of all rows to 94 of the table Student. To apply this to specific rows, WHERE clause can be applied along with UPDATE statement

Ex: UPDATE Student SET Marks1 = 94 WHERE name = 'Monica Rana';

DELETE STATEMENT:

DELETE statement is used to delete rows from a table. DELETE removes the entire row, not the individual column values.

Syntax: DELETE FROM < tablename> [Where < condn>];

Ex: DELETE FROM Student WHERE Rollno = 14;

DELETE statement can be used to delete all rows of the table also. The following statement can be used to delete all the rows from Student table.

Ex: DELETE from Student;

CONSTRAINTS:

A constraint is a rule that is applied on the columns of tables to ensure data integrity and consistency.

1. ***NOT NULL Constraint:***

   When this constraint is set to a column, it ensures that the value 'NULL' cannot be entered in the specified column in any row.

   Ex: CREATE TABLE Student (RollNo INT(5) NOT NULL, Name VARCHAR(25));

2. ***UNIQUE Constraint:***

   When this constraint is set to a column, it ensures that the duplicate values cannot be entered in the specified column.

   Ex: CREATE TABLE Student (RollNo INT(5) UNIQUE, Name VARCHAR(25));

3. ***PRIMARY KEY Constraint:***

   This constraint sets a column or a group of columns as the Primary Key of a table. Therefore, NULLs and duplicate values in this column are not accepted.

   Ex: CREATE TABLE Student (RollNo INT(5) PRIMARY KEY, Name VARCHAR(25));

4. ***FOREIGN KEY Constraint:***

   Data will be accepted in this column, if same data value exists in a column in another related table. This other related table name and column name are specified while creating the foreign key constraint

   Ex: CREATE TABLE Result (RollNo INT(5), Marks INT(3), FOREIGN KEY(RollNo) REFERENCES Student(RollNo));

5. ***ENUM Constraint:***

   This constraint defines a set of string values as the column domain. So any value in this column will be from the specified values only.

   Ex: CREATE TABLE Person (Name VARCHAR(40), Gender ENUM('Male', 'Female'));

   Now, while inserting rows into the table Person, for the field Gender the value either 'Male' or 'Female' only is to be entered. Attempting entering value other than this will results in error

6. ***DEFAULT Constraint:***

   This constraint is used to assign a default value when no value is provided to attribute.

   Ex: CREATE TABLE Person (Name VARCHAR(40), Gender VARCHAR(2) DEFAULT 'Male');

   Now, while inserting data into the table if value for the gender is not provided then the value 'Male' will be assigned.