Introduction To SQL

Page No :

- SQL Stands for Structure Query Language.
- SQL is non-procedural in nature. The SQL Standard Specifies data definition, data manipulation and other associated facilities of a DBMS that supports the relational data model.
- SQL is a Comprehensive Language for Controlling and interacting with a database management System.
- This Language was named as the Structured English Query Language (SEQUEL) the name Later was shortened to Structure query Language (SQL).

Roles of SQL ⇒

- It is an interactive Language because the user gains access to the data by issuing desired query. This provides a Convenient, Easy to use to for ad-hoc database queries.
- It is a database programming Language because SQL Commands Can easily be Embedded into application programs to access the data from a database.
- It is a database administration Language because the database administrator uses SQL to define the database structure and Control

access to the stored data.

- It is a Client/Server Language because it can be used to access remote (over a network) database.

- It is a distributed database Language because SQL helps in distribution of data across many connected Computer System.

Advantages of SQL: ⇒

1) Standard Independent Language: ⇒ The universal rules of the SQL have been established by ANSI and ISO. Therefore, the SQL is an Open Language implying it is not owned or controlled by any single Company.

2) Speed ⇒ The intense Competition among database vendors has resulted in faster, more robust database management System that work at lower Cost per transaction.

3) Easy to Learn and use: ⇒ SQL Statements resemble Simple English Sentences, making SQL easy to learn and understand.

4) Less programming: ⇒ SQL allows Extration, manipulation and organisation of data with Less programming as compared to traditional methods.

MYSQL/SQL Data Types

Data types are used to represent the nature of the data that can be stored in the database table.

Data types mainly classified into three categories for every database

1) String Data types.
2) Numeric Data types.
3) Date and time Data types.

1) MYSQL String Data types ⇒

a) Char(size) ⇒ It is used to specify a fixed length string that can contain numbers, letters, and special characters. Its size can be 0 to 255 characters. Default is 1.

b) Varchar(size) ⇒ It is used to specify a variable length string that can contain numbers, letters and special characters. Its size can be from 0 to 65535 characters.

c) Text(size) ⇒ It hold a string that can contain a maximum length of 255 characters.

d) Binary(size) ⇒ It is equal to Char() but stores binary byte strings. Its

Size parameter Specifies the Column length in the byte. Default is 1.

e) Varbinary (size) ⇒ It is equal to varchar() but Stores binary byte strings. Its size parameter Specifies the maximum Column length in bytes.

f) Tinytext ⇒ It hold a string with maximum length of 255 characters.

g) mediumtext ⇒ It hold a string with maximum length of 16,777,215.

h) Longtext ⇒ It hold a string with a maximum length of 4,294,967,295 characters.

2) MySQL Numeric datatypes ⇒

a) Bit (size) ⇒ It is used for a bit-value type. The number of bits per value is specified in size. Its size can be 1 to 64. the default value is 1.

b) Int (size) ⇒ It is used for the integer value. It's signed range varies from -214748-3648 to 2147483647 and unsigned range varies from 0 to 4294967295. the size parameter specifies the max display width that is 255.

c) Integer(size) ⇒ It is Equal to int(size).

d) float (size, d) ⇒ It is used to Specify a floating point number. Its size parameter specifies the total number of digits. The number of digits after the decimal point is specified by d parameter.

e) float(p)

(f) Double (size, d)

g) Decimal (size, d)

h) Bool

3) MYSQL Date and time datatypes.

a) Date ⇒ It is used to specify date format YYYY-MM-DD. Its Supported range is from '1000-01-01' to '9999-12-31'

b) Datetime (fsp) ⇒ It is used to Specify date and time Combination. Its format is YYYY-MM-DD hh:mm:ss

c) Time (fsp) ⇒ It is used to Specify the time format. It format is hh:mm:ss.

d) Year ⇒ It is used to Specify a year in four-digit format. Values allowed in four digit format from 1901 to 2155, and 0000.

SQL Server Data types ⇒

1) SQL Server String Datatype ⇒

"Aman"

a) Char(n) ⇒ It is a fixed width character string datatype. Its size can be up to 8000 characters.

b) varchar(n) ⇒ It is a variable width character string datatype. Its size can be up to 8000 charcharter.

c) varChar(max) ⇒ It is a variable width character string data types. Its size can be up to 1,073,741,824 characters.

d) Text ⇒ It is a variable width character string datatype. Its size can be up to 2GB of text data.

e) nchar ⇒ It is a fixed width Unicode string datatype. Its size can be up to 4000 characters.

f) nvarchar ⇒ It is a variable width Unicode string data type. Its size can be up to 4000 characters.

g) ntext ⇒ It is a variable width unicode string data type. Its size can be up to 2GB of text data.

h) binary(n) ⇒ It is a fixed width Binary String data type. Its size can be up to 8000 bytes.

i) Varbinary ⇒ It is a variable width Binary String data type. Its size can be up to 8000 bytes.

j) image ⇒ It is also a variable width Binary String data type. It size can be up to 2GB.

2) SQL Server Number Data types ⇒

a) bit ⇒ It is an integer that can be 0, 1 or null.

b) tinyint ⇒ It allows whole number from 0 to 255.

c) Smallint ⇒ It allows whole number between -32,768 to 32767.

d) int ⇒ It allows whole number between -2,147,483,648 and 2,147,483,647.

e) bigint ⇒

f) float(n) ⇒ 12.56

g) real ⇒ It is a floating precision number data from -3.40E+38 to 3.40E+38.

3) SQL Server Date and time Data type ⇒

a) datetime ⇒ It is used to specify date and time combination. It supports range from January 1, 1753, to December 31, 9999 with an accuracy of 3.33 milliseconds.

b) datetime2 ⇒ It is used to specify data and time combination. It supports range from January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds.

c) date ⇒ It is used to store date only. It supports range from January 1, 001 to December 31, 9999.

d) time ⇒ It stores time only to an accuracy of 100 nanoseconds.

e) timestamp ⇒ It stores a unique number when a new row gets created or modified the time stamp value is based upon an internal clock and does not correspond to real time.
Each table may contain only one-time stamp variable.

SQL Statement

Page No :

SQL Statement tell the database what operation you want to perform on the structured data and what information you would like to access from the database.

Most important SQL Commands and Statement are.

1) Create Database Statement ⇒

Syntax ⇒

```
Create database database_name;
```

Ex ⇒

```
Create database Demo;
```

2) Create table Statement ⇒

Syntax ⇒

```
Create table tablename (columnname1 datatype, columnname2 datatype ... columnnamen datatype);
```

Ex ⇒

```
Create table Student ( name varchar(50), ID int Not Null primary key, address Char(50));
```

3) Insert into table ⇒ You can insert a row in the table by using SQL insert into command. It has two ways:

way 1) Syntax ⇒

insert into table_name (columnname 1, columnname 2, columnname 3 ---) values (value 1, value 2, value 3 ---);

Ex ⇒

insert into Student (name, id, address) values ("Kailash", 04, "Haridwar");

way 2) Syntax ⇒

insert into table_name values (value 1, value 2, value 3 --- value n);

Ex ⇒

insert into Student values ("Kailash", 04, "Haridwar");

SQL Command

4) Select Statement ⇒ It is used to access the records from one or more database tables and views.
There are four select statement.

1) Whole table selection ⇒
Syntax ⇒

```
Select * from tablename;
```

Ex ⇒

```
Select * from student;
```

2) Select/Access particular column;

Syntax ⇒

```
Select Columnname1, Columnname2 -- from tablename;
```

Ex ⇒

```
Select name, ID from Student;
```

3) select/Access particular Row;

Syntax ⇒

```
Select * from tablename where
        Condition;
```

Ex ⇒

```
Select * from student where ID = 04;
```

5) Update Statement ⇒ SQL update Statement is used to change the data of records held by table. Which rows is to be update, it is decided by a conditions. To specify condition, we use where clause.

Syntax ⇒

Update table_name set [Column_name1 = value1, Columnname2 = value2 --- Columnname3 = value3] [where Condition];

Ex ⇒
update ~~Student~~ set name = "Aman", address = "Dehradun" where id = 04;

6) Alter table Statement ⇒ • It is used to add, modify and delete columns of an existing table. (DDL)
• any user can change the name of table using this Statement.

Syntax ⇒

Alter table table_name Add Columnname datatype

Ex ⇒
Alter table student add email varchar(50);

SQL Command

7) Delete Statement ⇒ SQL Delete Statement is used to delete rows from a table.
Delete statement remove one or more records from a table.
Syntax ⇒

```
Delete from table_name where
    Condition;
```

Ex ⇒

```
Delete from Student where
    ID = 04;
```

in the given example Student table with ID 04 whole row is deleted.

Ex ⇒ Delete from table_name;

```
Delete from Student;
```

it deleted whole record from a student table but not free the space containing by the table.

8) Delete table ⇒ (Truncate Statement) ⇒

It is used to delete all the row from the table and free the containing space.

Page No :

Syntax ⇒ Truncate table tablename;

Ex ⇒ Truncate table Student.

8) Drop Command ⇒ Drop statement is deletes the table's row together with the table's definition so all the relationships of that table with other tables will no longer be valid.

When you drop a table.

- table Structure will be dropped
- Relationship will be dropped
- Integrity Constraints will be dropped
- Access privileges will also be dropped.

Syntax ⇒

Drop table table_name;

Ex ⇒

Drop table Student;

9) Drop Column ⇒

Syntax ⇒ alter table table_name drop Column Column_name;

Ex ⇒

alter table Student drop Column name;

SQL Operators

SQL operators are used for filtering the table's data by specific condition in the SQL statement.

There are six types of SQL operators.

1) SQL Arithmetic operators +, -, /, *, %.
2) SQL Comparison operators > < >= <=
3) SQL Logical Operators And, OR
4) SQL Set operators Union, inters, Main
5) SQL Bit-wise Operators
6) SQL unary Operators.

1) SQL Arithmetic Operators: The Arithmetic

Operators perform the mathematical operation on the numerical data of the SQL tables. These operators perform addition, Subtraction, multiplication and division operations on the numerical operands.

Arithmetic operators are :-

1) SQL Addition operator (+)
2) SQL Subtraction operator (-)
3) SQL multiplication operator (*)
4) SQL Division operator (/)
5) SQL Modulus operator (%)

Page No:

Ex⇒ We want to add 2000 to the Salary of each Employee in Employee table so the following query in the SQL.

Query. Select Emp_Salary + 2000 as Emp_New_Salary from Employee.

2) SQL Comparison Operators : The Comparison Operators in SQL Compare two different data of SQL table and check whether they are the Same, greater and Lesser.
The SQL Comparison operators are used with the WHERE Clause in the SQL queries.
Comparison operators are :-

1) SQL Equal operator (=)
2) SQL Not Equal operator (!=)
3) SQL Greater Than operator (>)
4) SQL Greater Than Equals to Operator (>=)
5) SQL Less Than Operator (<)
6) SQL Less Than Equals to operator (<=)

Ex⇒ To find all details of Employee having Salary Equal to 30,000.

Query⇒ Select * from Employee where Emp_Salary != 30,000;

Name	ID	Sal
		30,000
		30,000

SQL Operators

Page No :

3) SQL Logical Operators :⇒

The Logical operators in SQL perform the Boolean operations, which give two results True & false.

SQL Logical operators are :-

1) SQL ALL operator
2) SQL AND operator
3) SQL OR operator
4) SQL BETWEEN operator
5) SQL IN operator
6) SQL NOT operator
7) SQL ANY operator
8) SQL LIKE operator.

1) **SQL ALL Operator ⇒** The all operator in SQL Compares the Specified value to all the values of a Column from the Sub-query in the SQL database.

This operator is always used with the following statement.

1) Select
2) having
3) where.

Syntax ⇒

```
Select Column1, column2 from table_name
where Column Comparison_Operator
  ALL (Select Column from table2).
```

Page No :

Ex⇒ We want to access the Employee i'd and Employee names of those Employees from the Employee table whose Salaries are greater than the Salary of Employees who lives i'n Haridwar City, then we have to type the following query i'n SQL.

query⇒ Select Emp-i'd, Emp-Name from Employee where Emp-Salary > ALL (select Emp-Salary from Employee where City = "Haridwar");

2) SQL AND Operator ⇒

The AND operator i'n SQL would show the record from the database table if all the Conditions Separated by the AND operator Evaluated To True.

It i's also known as the Conjunctive operator and i's used with the where Clause.

Syntax ⇒

Select * from table_Name where Condition1 AND Condition2 AND Condition3 ...;

Ex⇒ We want to access all the records of those Employees from Employee table whose Salary i's 25000 and city i's Haridwar.

query⇒ Select * from Employee where Emp-Salary = 25000 and Emp-City = "Haridwar";

3) SQL OR Operator ⇒ The or operator in SQL shows the record from the table if any of the conditions separated by the OR operator evaluates to True.
It is also known as the conjunctive operator and is used with the where clause.

Syntax ⇒

```
Select Column1, Column2 -- Columnn from
table_Name where Condition1 OR Condition2
--- OR ConditionN;
```

4) SQL BETWEEN operator ⇒ The Between operator in SQL shows the record within the range mentioned in the SQL query.
This operator operates on the numbers, characters, and date/time operands.
If there is no value in the given range, then this operator shows NULL value

Syntax ⇒

```
Select Column1, column2 -- Columnn from
table_name where Columnname BETWEEN
Value1 and value 2;
```

Page No :

5) SQL IN operator ⇒ the In operator in SQL allows database users to specify two or more values in a where clause.
This Logical operator minimizes the requirement of multiple OR conditions.

Syntax ⇒

```
Select Column1, Column2 -- Columnn from table_Name
  where Column_name In (List of values);
```

6) SQL NOT operator ⇒ the Not operator in SQL shows the record from the table if the Condition Evaluates to false. It is always used with the where clause.

Syntax ⇒

```
Select Column1, Column2 --- ColumnN from
  table_Name where NOT Condition;
```

Ex⇒ Suppose, we want to show all the information of those Employees from the Employee table whose City not dehradun.

Query ⇒

```
Select * from Employee where Not
  Emp_City = 'dehradun';
```

7) SQL ANY Operator ⇒ The any operator in SQL shows the records when any of the values returned by the Sub-query meet the condition.

The Any Logical operator must match at least one record in the inner query and must be preceded by any SQL Comparison operator.

Syntax ⇒

```
Select Column1, Column2 -- ColumnN from
table_Name where Column_name Comparison_
operator ANY (select Column_name from
table_Name where Condition(s));
```

8) SQL Like Operator ⇒ The Like operator in SQL shows those records from the table which match with the given pattern specified in the sub-query.

The Like operator is used in a where clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the Like operator:

- The percent sign (%) represents zero, one or multiple characters.
- The underscore sign (_) represents one, single character.

The percent sign and the underscore can also be used in combinations.

Syntax ⇒

```
Select Column1, Colum2 -- ColumnN from
table_Name where Columnname Like pattern;
```

Ex ⇒ We want to show all the information of those Employees from the Employee table whose name starts with "K".

query ⇒

```
Select * from Employee where
Emp_Name Like 'K%';
```

query ⇒

```
Select * from Employee where
Emp_Name Like '%t';
```

query ⇒

```
Select * from Employee where
Emp_Name Like '_a%';
```

SQL Operators

4) **SQL Set Operators** :⇒ • the set operators in SQL Combine a similar type of data from two or more SQL database tables.

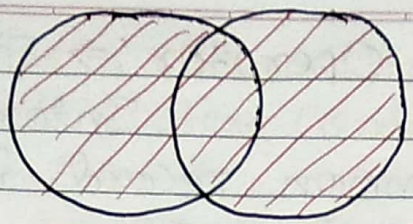
- It mixes the result, which is Extracted from two or more SQL queries, into a single result.
- Set operators Combine more than one select statement in a single query and return a specific result set.

SQL Set Operators are :-

1) SQL Union operator
2) SQL Union ALL operator
3) SQL Intersect Operator
4) SQL minus operator

1) **SQL Union Operator** :⇒ • the SQL Union operator Combines the result of two or more SELECT Statement and provide the single output.

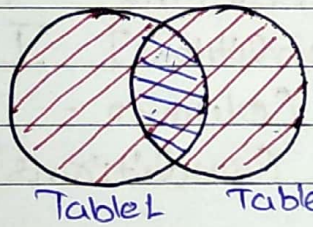
- It will Eliminate duplicate rows from its resultset
- The data type and the number of Columns must be the same for each Select Statement used with the Union operator.



Syntax :⇒

Select Column1, Column2 -- ColumnN from table1 [where Condition] Union Select Column1, Colum2 -- ColumnN from table2 [where Condition];

2) SQL Union ALL operator :⇒ The SQL Union ALL operator is the same as the Union operator. But it also shows the duplicate row.

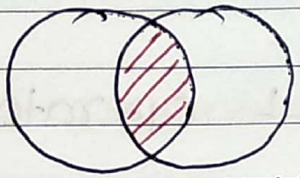


Syntax :⇒

Select Column1, Column2 -- ColumnN from table1 [where Condition] Union ALL Select Column1, Column2 -- ColumnN from table2 [where Condition];

3) SQL Intersect Operator :⇒ The SQL Interset Operator shows the Common record from two or more SELECT Statements.
The data type and the number of columns must be the same for each SELECT Statement used with the Intersect operator.

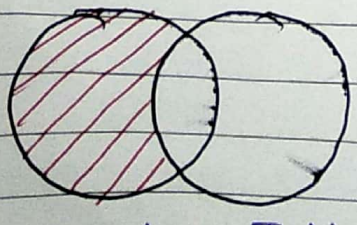
NOTE⇒ MySQL does not Support Intersect operator.



Syntax ⇒

```
Select Column1, Column2 -- ColumnN from
table1 [where Conditions] INTERSECT
Select Column1, Column2 -- ColumnN from
table2 [where Conditions];
```

4) SQL Minus Operator :⇒ SQL minus Operator Combines the result of two or more SELECT Statements and shows only the results from the first data set.



Page No :

Syntax ⇒ Select Column1, Column2 -- ColumnN from table1 [where Conditions] Minus Select Column1, Column2 -- ColumnN from table2 [where Conditions];

Difference between SQL and NoSQL

	SQL	No-SQL
1)	SQL is a relational database management System.	1) NO-SQL is a non-relational or distributed database management System.
2)	The query language used in this database System is a Structured query Language (SQL).	2) The query language used in the NO-SQL database System is a non-declarative query language (MongoDB)
3)	SQL Database are Vertically Scalable.	3) NO-SQL databases are horizontally Scalable.
4)	The database type of SQL is in the form of tables Suchas in the form of rows and Columns.	4) The database type of NO-SQL is in the form of documents, key-value and graphs.
5)	It follows ACID property (Atomicity, Consistency, Isolation, durability)	5) It follows CAP (Consistency, availability, partition tolerance).
6)	Complex queries are easily managed in the SQL database	6) NOSQL databases Cannot handled Complex queries.

Page No :

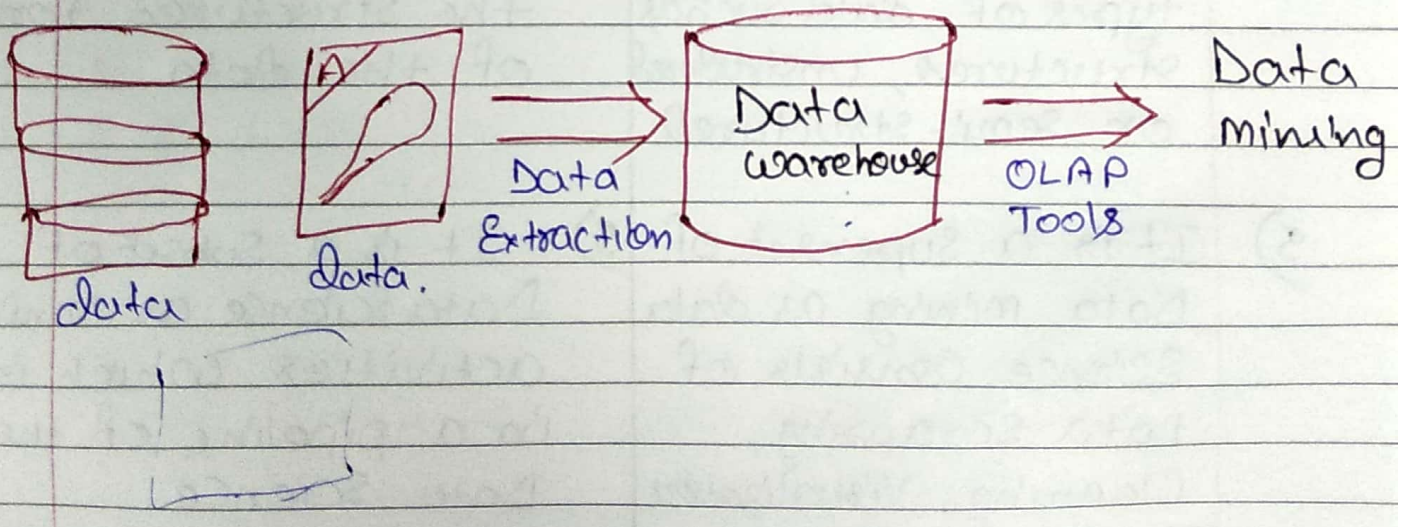
SQL	NO-SQL
7) SQL database is not the best choice for storing hierarchical database.	7) No-SQL database is a perfect option for storing hierarchical database.
8) All SQL database require object-relational mapping	8) many No-SQL database do not require object-relational mapping.
9) SQLite, MS-SQL, Oracle, PostgreSQL and MySQL are Example of SQL database Systems.	9) Redis, MogoDB, Hbase, Bigtable, CouchDB are Example of NoSQL database System.

Page No:

Difference between Data Mining and Data Warehousing :⇒

	Data mining	Data Warehousing
1)	Data mining is the process of determining data patterns.	1) A data warehouse is a database system designed for Analytics.
2)	Data mining is generally considered as the process of extracting useful data from a large set of data.	2) Data warehousing is ~~entirely carried out by the engi~~ the process of combining all the relevant data.
3)	Data mining, data is analyzed repeatedly.	3) Data warehousing, data is stored periodically.
4)	Data mining is carried out by business users with the help of Engineers.	4) Data warehousing is solely carried out by engineers.
5)	Data mining uses machine learning algorithms, databases, statistics and AI.	5) Data warehousing is integrated, subject-oriented, time-variant and non-volatile.
6)	Data mining uses pattern recognition techniques to identify patterns	6) Data warehousing is the process of extracting and storing data that allow easier reporting.

Data mining	Data warehousing
7) The data mining techniques are cost-efficient as compared to other statistical data applications	7) The responsibility of the data warehouse is to simplify every type of business data.
8) One of the most amazing data mining technique is the detection and identification of the unwanted errors that occur in the system.	8) One of the advantages of the data warehouse is its ability to update frequently. that is the reason why it is ideal for business entrepreneurs who want up to date with the latest stuff.



Difference Between Data Science and Data Mining

	Data Science	Data Mining
1)	Data Science is an Area.	1) Data mining is a Technique.
2)	It is about Collection, processing, analyzing and utilizing of data into various operations. It is more Conceptual.	2) It is about Extracting the vital and valuable information from the data.
3)	It is a field of study just like the Computer Science, Applied Statistics or Applied Mathematics.	3) It is a technique which is a part of the knowledge Discovery in Database processes (KDD).
4)	It deals with all types of data such as structured, unstructured or semi-structured.	4) It mainly deals with the structured forms of the data.
5)	It is a superset of Data Mining as data Science consists of Data Scrapping, Cleaning, Visualization, Statistics and many more techniques.	5) It is a subset of Data Science as mining activities which is in a pipeline of the Data Science.

Data Science	Data mining
6) It is mainly used for Scientific purposes.	6) It is mainly used for business purposes
7) It broadly focuses on the science of the data	7) It is more involved with the processes.

Difference Between DDL and DML

DDL	DML
1) It stands for Data Definition Language.	1) It stands for Data Manipulation Language.
2) It is used to Create, update delete, modify database structure but not data.	2) It is used to Create, update, delete, modify data but not database structure.
3) It basically defines the column (Attributes) of the table.	3) It add or update the row of the table. These rows are called as tuples.
4) DDL does not have further classification.	4) DML is further classified as procedural DML and non-procedural DML.
5) DDL Command is used to Create the database schema.	5) DML Command is used to popular and manipulate database

DDL	DML
6) DDL Statement Cannot be rolled back.	6) DML Statement Can be rolled back.
7) DDL does not use where clause in its Statement.	7) DML uses where Clause in its Statement.
8) DDL is declarative.	8) DML is imperative
9) DDL Statement effect the whole table.	9) DML effects one or more rows.
10) Basic Command present in DDL are Create, Drop, Rename, alter etc.	10) Basic Command present in DML are update, insert, merge etc.
Ex ⇒ Create table student (Name varchar(50), Branch varchar(20));	Ex ⇒ insert into student (Name, Branch) values ("Kamal", "IT");

Difference between Alter and Update Command in SQL.

Alter	Update
1) Alter Command is Data Definition Language (DDL)	1) Update Command is a Data Manipulation Language (DML)
2) Alter Command will perform the action on structure level and not on the data level	2) Update Command will perform on the data level.
3) Alter Command is used to add, delete, modify the attributes of the relations (tables) in the database.	3) Update Command is used to update Existing records in a database.
4) Alter Command by default initializes values of all the tuple as NULL.	4) Update Command sets Specified values in the Command to the tuples.
5) This Command make Changes with table Structure	5) This Command make Changes with data inside the table.

Alter	Update
6) It works on the attributes of a relation.	6) It works on the attribute of a particular tuple in a table.
7) Syntax with Example Syntax: Alter table tableName Drop Column ColumnName; Ex ⇒ Alter table Student Drop column address;	7) Syntax with Example Syntax: Update tableName set Column1 = Value, Column2 = Value, -- columnN = Value where Condition; Ex ⇒ Update Student Set Name = "kailash", City = " Haridwar" where ID = 10;

Difference between Delete, Drop and Truncate Command in SQL

DELETE Command	Drop Command	TRUNCATE Command
1) The Delete Command is Data manipulation Language (DML).	1) The Drop Command is Data Definition Language (DDL).	1) The truncate Command is a Data Definition Language (DDL).
2) The Delete Command deletes one or more existing records from the table in the database	2) The Drop Command Drops the Complete database table from the database.	2) The truncate Command deletes all the rows from the existing table, Leaving the row with the column names.
3) We can restore any deleted row or multiple rows from the database using the ROLLBACK Command.	3) We cannot get the Complete table deleted from the database using ROLLBACK Command.	3) We cannot restore all the deleted rows from the database using the ROLLBACK Command.
4) The Delete Command does not free the allocated Space of the table from memory.	4) The Drop Command removes the Space allocated for the table from memory	4) The truncate Command does not free the Space allocated for the table from memory.
5) The Integrity Constraints remain the same in the Delete Command.	5) The Integrity Constraints get removed from the Drop command.	5) The Integrity Constraints will not get removed from the Truncate Command.
6) Syntax :- Delete from table_name where Condition;	6) Syntax :- Drop table table_name;	6) Truncate table table_name;

SQL JOIN

⇒ Join means to Combine Something. In case of SQL, Join means "to Combine two or more tables".

⇒ SQL Join Clause takes records from two or more tables in a database and combines it together.

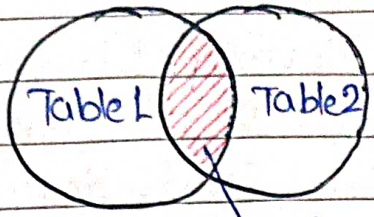
⇒ SQL Join statement is used to Combine data or rows from two or more tables based on a common field between them. ~~Differe~~

⇒ ANSI standard SQL defines five types of JOIN.

1) Inner join
2) Left outer join
3) right outer join
4) full outer join
5) Cross join

1) Inner Join ⇒ the inner Join keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both

the tables where the condition satisfies such as value of the common field will be the same.



Syntax ⇒

```
Select table1.Column1, table1.Column2, table2.
Column1, .... from table1 Inner join
table2 on table1.matching_Column = table2.matching_Column;
```

NOTE ⇒ We can also write Join instead of INNER JOIN. JOIN is same as INNER JOIN.

Ex ⇒ Consider the two tables

Student (Table1)

Roll_no	Name	Address	Age
1	Kailash	Dehradun	30
2	Kamal	Delhi	25
3	Karan	Rishikesh	18
4	Aman	haridwar	19
5	Ankit	Ramnagar	20

StudentCourse (Table2)

Course_ID	Roll_no
1	1
2	2
2	3
3	5
1	

Query ⇒ To Access the details (names and age) of Students Enrolled in Different Courses.

```
Select StudentCourse.Course_ID, Student.Name,
Student.age from Student inner Join
StudentCourse ON Student.Rollno = Student.
Course.Roll_no;
```

Output ⇒

Course_ID	Name	Age
1	Kailash	30
2	Kamal	25
2	Karan	18
3	Ankit	20

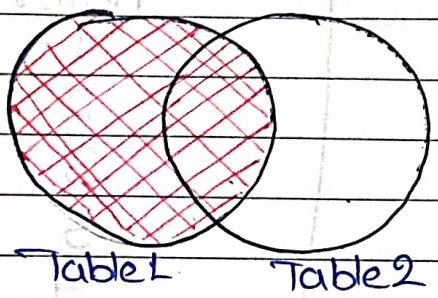
```
Select * from Student inner join StudentCourse
ON Student.Rollno = StudentCourse.Rollno;
```

SQL OUTER JOIN

⇒ In the SQL Outer JOIN all the Content of the both tables are integrated together either they are matched or not.

Outer Join of two types.

1) Left outer Join (also known as Left Join) ⇒ this join returns all the rows from Left table Combine with the matching rows of the right table. If you get no matching in the right table it returns NULL values.



Syntax ⇒

```
Select table1.Column1, table1.Column2,
table2.Column1 ---- from table1 LEFT
JOIN table2 On table1.matching_column =
table2.matching_column;
```

NOTE ⇒ We can also use Leftoutter join instead of Left join, both are same.

Ex ⇒ Consider the two tables.

Student (table1)

Rollno	Name	address	Age
1	Kailash	Dehradun	30
2	Kamal	Delhi	25
3	Karan	Rishikesh	18
4	Aman	haridwar	19
5	Ankit	Ramnagar	20

StudentCourse (Table 2)

Course_ID	Roll_no
1	1
2	2
2	3
3	5
1	9

Query ⇒ To Access the all Student Name field with given Course ID.

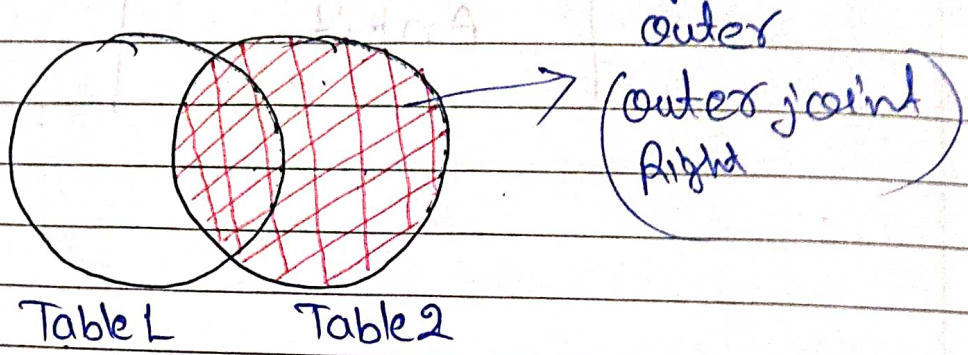
```
Select Student.Name, StudentCourse.
CourseID from Student Left Join
StudentCourse ON StudentCourse.
Roll_no = Student.Roll_no;
```

Output.

Name	Course_ID
Kailash	1
Kamal	2
Karan	2
Aman	NULL
Ankit	3

SQL RIGHT JOIN

⇒ The SQL right Join returns all the values from the rows of right table. It also includes the matched values from left table but if there is no maching in both tables, it returns NULL. It 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;
```

NOTE ⇒ We can also use Right outer Join instead of Right Join, both are the same.

Ex⇒ Consider the two tables

Student (Table 1)

Roll no	Name	Address	Age
1	Kailash	Dehradun	30
2	Kamal	Delhi	25
3	Karan	Rishikesh	18
4	Aman	haridwar	19
5	Ankit	Ramnagar	20

StudentCourse (Table 2)

Course_ID	Roll_no
1	1
2	2
2	3
3	5
1	9

query ⇒ To Access All Student ID with Name.

Command ⇒ Select Student.Name, StudentCourse.Course_I
from Student Right Join StudentCourse
ON StudentCourse.Rollno = Student.Roll_no;

output.

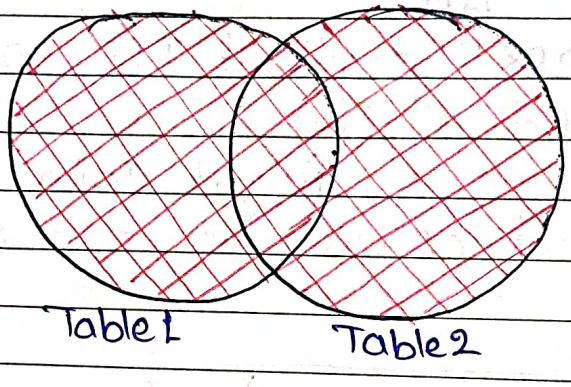
Name	Course_ID
Kailash	1
Kamal	2
Karan	2
Ankit	3
NULL	1

Select * from Student Right Join StudentCours
ON StudentCourse.Rollno = Student.Rollno;

SQL FULL JOIN / FULL OUTER JOIN

⇒ The SQL full join is the result of combination of both left and right outer join and the join tables have all the records from both tables. It puts NULL on the place of matches not found.

⇒ SQL full outer join and SQL join are same. generally it is known as SQL FULL JOIN.



Syntax ⇒

```
Select table1.Column1, table1.Column2,
table2.Column1 .... from table1 FULL
Join table2 ON table1.matching_Column =
table2.matching_Column;
```

Ex ⇒ Consider the two tables

Student (Table 1)

Roll_no	Name	Address	Age
1	Kailash	Dehradun	30
2	Kamal	Delhi	25
3	Karan	Rishikesh	18
4	Aman	haridwar	19
5	Ankit	Ramnagar	20

StudentCourse (Table 2)

Course_ID	Roll no
1	1
2	2
2	3
3	5
1	9

query ⇒ Select Student.Name, StudentCourse.Course_I
from Student FULL JOIN StudentCourse
ON StudentCourse.Roll_No = Student.Roll_No ;

Output

Name	Course_ID
Kailash	1
Kamal	2
Karan	2
Aman	NULL
Ankit	3
NULL	1

Select * from Student full joint StudentCourse
one StudentCourse.Rollno = Student.Rollno.

Rollno.	Name	address	Age	CourseID
1	Kailash	Dh	30	1
2	Kam	Del	25	2
3	Kar	Ru	18	2
4	Am	her	19	NULL
5	Ankit	R	20	3
9	NULL	NULL	NULL	1

SQL Cross Join

⇒ SQL Cross Join Combine two Different tables, ~~other~~ Then we will get the Cartesian product of the sets of rows from the joined table. when each row of the first table is Combined with each row from the second table. It is known as Cartesian join or Cross join.

⇒ After performing the Cross join operation, the total Number of rows present in the final table will be Equal to the product of the number of rows present in table1 and the number of rows present in table 2.

table1 (3)
table2 (3) ⇒ 3×3 = 9

Syntax ⇒

```
Select tableName1.Column1, table1.
Column2, table2 Column1 ... from table1
CROSS JOIN table2 ON Table1.matching_
Column = table2.matching_Column;
```

Ex ⇒ Consider the two table.

Student (table1)

Rollno	Name	address	Age
1	Kailash	Dehradun	30
2	Karan	haridwar	25
3	Kamal	Rishikesh	18

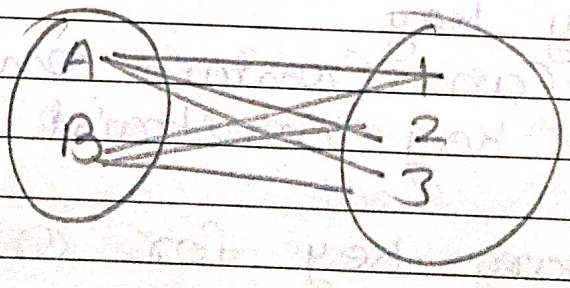
StudentCourse (table 2)

Course_ID	Rollno.
1	1
2	3
3	5

query ⇒ Select * from Student
Cross Join StudentCourse;

Rollno	Name	Address	Age	Curse_ID	Roll
1	Kailash	Dehradun	30	1	1
1	Kailash	Dehradun	30	2	3
1	Kailash	Dehradun	30	3	5
2	Karan	haridwar	25	1	1
2	Karan	haridwar	25	2	3
2	Karan	haridwar	25	3	5
3	Kamal	Rishikesh	18	1	1
3	Kamal	Rishikesh	18	2	3
3	Kamal	Rishikesh	18	3	5

3*3 = 9



SQL Primary Key

⇒ A Column or Columns is called Primary Key (PK) that uniquely identifies Each row in the table.

⇒ If you want to Create a primary key, you should define a Primary Key Constraint when you Create or modify a table.

⇒ When multiple Columns are used as a primary key, it is known as Composite primary key.

Important points for Primary Key:

- Primary key enforces the Entity integrity of the table.
- Primary key always has unique data.
- A primary length Cannot be Exceeded than 900 bytes.
- A Primary key Cannot have null value.
- There Can be no duplicate value for a Primary key.
- A table Can Contain only one Primary key Constraint.

SQL Primary key for one Column:

The given SQL Command Create a Primary key on the "S_ID" Column When the "Student" table is Created.

In MySQL query ⇒

```
Create table Student ( S_ID int Not NULL,
Name varchar (250), address varchar (250)
City varchar (100), Primary key (S_ID));
```

In SQL Server, oracle, MS Access query =

```
Create table Student ( S_ID int Not NULL
primary key, Name varchar (250), address
varchar (250), City varchar (100));
```

SQL primary key for multiple Columns:

MySQL, SQL Server, oracle, MS Access query

```
Create table Student ( S_ID int Not NULL,
Name varchar (250) Not NULL, address
Varchar (250), City varchar (100),
Constraint PK_StudentID Primary key
( S_ID, Name) );
```

Note :⇒ you should note that in the above example there is only one Primary key (PK_StudentID). It is made up of two Columns (S_ID and Name).

SQL primary key using ALTER Command ⇒

When table is already created and you want to create a PRIMARY KEY Constraint on the "S_ID" Column you should use the following SQL Statement:

Primary key on one Column:

```
ALTER TABLE Student ADD Primary
Key (S_ID);
```

Primary key on multiple Column:

```
ALTER TABLE Student ADD Constraint
pk_StudentID Primary key (S_ID, Name);
```

NOTE ⇒ when you use Alter table Statement to add a primary key, the primary key Columns must not Contain NULL values (when the table was already Created).

Drop a Primary key Constraint ⇒

if you want to Drop (remove) a primary key Constraint, you should perform the following syntax:

MYSQL ⇒

```
Alter table Student Drop primary
    key;
```

SQL Server, oracle, MS Access :

```
Alter table Student Drop
Constraint pk_StudentID;
```

SQL foreign key

⇒ In the relational databases, a foreign key is a field or a column that is used to establish a link between two tables.

⇒ Foreign key is one table used to point primary key in another table.

Ex ⇒

Here are two tables first one is Students table and second is orders table.

first table (Students)

P.K

S_ID	Name	Address	City
1	Kailash	Dehradun	Dehradun
2	Kamal	Haridwar	Haridwar
3	Ram	Rishikesh	Rishikesh

Second table (orders)

P.K F.K

O_ID	order_NO.	S_ID
1	123	2
2	324	2
3	567	3
4	354	1

Note ⇒ "S_ID" Column in the Order table points to the "S_ID" Column in Students table.

⇒ The "S_ID" Column in the Students table is the primary key in the Students table.

⇒ The S_ID Column in the Orders table is foreign key in the Orders table.

The foreign key Constraint is generally prevents action that destroy links between tables.

SQL foreign key Constraint ON Create table ⇒

To Create a foreign key on the "S_ID" Column when the "Orders" table is Created.

In MySQL ⇒

Create table orders (O_id int Not NULL, order_No int Not null, S_ID int Primary key (O_Id), Foreign key (s_id) References ~~Persons~~ Student (S_ID));

In SQL server / oracle / MS Access ⇒

Create table orders (O_id int Not NULL Primary key, order_No int Not NULL, S_ID int foreign key References ~~persons~~ Student (S_ID)

SQL foreign key Constraint for ALTER Table

If the order table is already Created and you want to Create a foreign key Constraint on the "S_ID" Column, you perform the following query ⇒

In MySQL / SQL server / oracle / MS Access

```
Alter table orders ADD Constraint fk_perorde
foreign key (s_id) References Students (s_id);
```

Drop Syntax for foreign key Constraint ⇒

If you want to drop a foreign key Constraint, use the following query.

In MySQL ⇒

```
Alter table orders Drop foreign key
fk_Perorders;
```

In SQL Server / oracle / MS Access ⇒

```
Alter table orders Drop Constraint
fk_Perorders;
```

Difference between primary key and foreign key in SQL ⇒

Primary key	foreign key
1) Primary key uniquely identify a record in the table. Ex ⇒ aadhar no.	1) foreign key is a field in the table that is primary key in another table. Ex ⇒ aadhar no. = AN
2) Primary key Can not accept Null values.	2) foreign key Can accept multiple null value.
3) By default, Primary key is clustered index and data in the database table is physically organized in the sequence of Clustered index.	3) foreign key do not automatically Create an index, Clustered or non-Clustered. You can manually Create an index on foreign key.
4) We Can have only one Primary key in a table.	4) We Can have more than one foreign key in a table.
5) Primary key is always unique	5) foreign key Can be duplicated.

SQL Composite key

⇒ A Composite key is a Combination of two or more Columns in a table that can be used to uniquely identify each row in the table. When the columns are combined uniqueness is guaranted, but when it taken individually it does not guarantee uniqueness.

⇒ Sometimes more than one attributes are needed to uniquely identify an Entity. A primary key that is made by the Combination of more than one attribute is known as a Composite key.

⇒ Composite key is a key which is the Combination of more than one field or Column of a given table. It may be a Candidate key or primary key.

⇒ Column that make up the Composite key Can be of different data types.

SQL Syntax for Composite key:

```
Create table Student ( Name varchar(50),
ID int, address varchar(100), primary key
(Name, address));
```

In all Cases Composite key Created Consist of Name and address.

mySQL / SQL Server / oracle Example ⇒

```
Create table tableName ( Column1 integer,
Column2 varchar(30), Column3 Varchar(50),
Primary key (column1, column2) );
```

Rollno. Name — key
(Composite key)

Unique key in SQL

⇒ A unique key is a set of one or more than one field/columns of a table that uniquely identify a record in a database table.

⇒ Unique key say that it is little like primary key but it can accept only one null value and it cannot have duplicate values.

⇒ The unique key and primary key both provide a guarantee for uniqueness for a column or a set of columns.

⇒ There is automatically defined unique key constraint within a primary key constraint.

⇒ There may be many unique key constraints for one table, but only one primary key constraint for one table.

SQL Unique key constraint on Create table ⇒

```
Create table Student ( S_id int Not NULL Unique, Name varchar(100) not NULL, address varchar(100), Phone_no int, Branch varchar(5));
```

In MySQL

```
Create table Student ( S-ID int not NULL,
Name varchar (100) not null, address
varchar, phone-no int, Branch varachar(5),
Unique (S-ID) );
```

SQL Unique Constraint on Alter table ⇒

if you want to Create a unique Constraint on S-ID Column when the table is already Created, you should use the following statement.

```
Alter table Student ADD Unique (S-ID);
```

SQL String functions

→ SQL String functions are the predefined functions that allow the database users for String manipulation. These functions only accept, process and give results of the string data type.

Some of the important string function are.

1) ASCII() ⇒ Return the ASCII Value for the Specific Character.

Ex ⇒

```
Select ASCII ("A");
```

output ⇒ 65

2) Char_Length() ⇒ Returns the Length of a String (in Characters).

Ex ⇒ Select Char_Legth ("Kailash Joshi");

output ⇒ 13

3) Concat() ⇒ Add two or more Expression together.

Ex ⇒ Select Concat ('Kailash', 'Joshi');
output ⇒ Kailashjoshi

4) Concat_ws() ⇒ Add two or more Expression together with a Seprator.

Ex ⇒ Select Concat_WS('_', 'Kailash', 'Joshi');

Output ⇒ Kailash_Joshi

5) field() ⇒ Return the index position of a value in a list of values.

Ex ⇒ Select field('a', 'x', 'y', 'a', 'b');

Output ⇒ 3

6) find_in_set() ⇒ Return the position of a String within a list of Strings.

Ex ⇒ Selet field_in_set("a", "x,a,p,q");

Output ⇒ 2

7) format() ⇒ formats a number to a format like "#,###,###.##", rounded to a specified number of decimal places.

Ex ⇒ Select format(2354.5634, 2);

Output ⇒ 2354.56

8) Insert() ⇒ Insert a String within a String at the specified position and for a certain number of characters.

Ex ⇒ Select insert("CSEngineering.com", 1, 13, "Kailash");

Output ⇒ Kailash.com.

9) LCase() ⇒ Converts a String to Lower-case.

Ex ⇒ Select LCase("KAILASH");

output ⇒ kailash

10) Lower() ⇒ Convert a String to Lower-case
Same as LCase().

11) Upper() ⇒ Convert a string to upper-case.

Ex ⇒ Select upper("kailash");
output ⇒ KAILASH

12) UCase() ⇒ Convert a String to upper-case
Same as Upper().

13) LTRIM() ⇒ Remove Leading Space from a String.

Ex ⇒ Select Ltrim(" kailash Joshi ");

output ⇒ kailash Joshi

14) RTRIM() ⇒ Remove trailing Space from a string.

15) TRIM() ⇒ Remove Leading and trailing Space from a String.

16) Reverse() ⇒ Reverse a string and return the result.

Ex ⇒ Select Reverse ("kailash");

output ⇒ hsailak.

17) Replace() ⇒ Replace all occurences of a substring within a string, with a new substring.

Ex ⇒ Select replace ("kailash Joshi", "kailash", "kamal");

output ⇒ kamal Joshi

18) SubStr() ⇒ Extracts a substring from a string (starting at any position)

Ex ⇒ Select SubStr ("kailash Joshi", 9, 3);

output ⇒ Jos

SQL/MYSQL Numeric function

SQL/MYSQL numeric functions are used primarily for numeric manipulation and/or mathematical calculations.

Some of the important String function are:-

1) ABS() ⇒ Returns the absolute value of numeric expression.

Ex ⇒ Select ABS(-235);

output ⇒ 235

2) ACOS() ⇒ Returns the arccosine of a numeric Expression. Returns NULL if the value is not in the range -1 to 1.

Ex ⇒ Select ACOS(0.8);

output ⇒ 0.6435011087932843

3) ASIN() ⇒ Returns the arcsine of numeric Expression. Returns NULL if value is not in the range -1 to 1.

Ex ⇒ Select ASIN(0.8);

output ⇒ 0.9272951800161 23

4) ATAN() ⇒ Returns the arctangent of numeric Expression.

Ex ⇒ ATAN(0.8)

Output ⇒ 0.6747409422235527

5) AVG() ⇒ Returns the average value of an Expression

Ex ⇒ Select Avg(marks) from student;

Output ⇒ 60

6) CEIL() ⇒ Returns the smallest integer value that is >= to a number.

or

7) CEILING() ⇒ Returns the Smallest integer value that is >= to a number.

Ex ⇒ Select CEIL(97.75);

or

Select CEILING(97.75);

Output ⇒ 98

8) FLOOR() ⇒ Returns the Largest integer value that is <= to a number.

Ex ⇒ Select floor(97.75);

Output ⇒ 97

9) Degree() ⇒ Converts a value in radians to degrees.

Ex ⇒ Select Degree(1.5)

output ⇒ 85.943669. { $\frac{1.5 \times 180}{\pi} = \frac{1.5 \times 180}{3.14159}$

$= 1.5 \times 57.2958$

$= 85.9436...$

10) DIV() ⇒ Used to Integer division.

Ex ⇒ Select 10 DIV 5;

output ⇒ 2

11) Greatest() ⇒ Returns the Largest integer value ~~that is~~ of the list of arguments.

Ex ⇒ Select greatest(10, 20, 3, 4, 18);

output ⇒ 20

12) Least() ⇒ Return the Smallest value of the list of arguments.

Ex ⇒ Select Least(10, 20, 3, 4, 18);

output ⇒ 3

13) LOG() ⇒ Return the natural logarithm of a number, or the Logarithm of a number to a specified base.

Ex⇒ Select Log(2);

output⇒ 0.693147..

14) LOG10() ⇒ Return the natural Logarithm of a number to base 10,

Ex⇒ Select Log10(2);

output⇒ 0.30102999--

15) LOG2() ⇒ Returns the natural logarithm of a number of base 2.

Ex ⇒ Select Log2(2); $\log_2 2 = 1$

output⇒ 1

16) SUM() ⇒ Calculates the sum of a set of values

Ex⇒ select Sum(marks) from students;

output ⇒ 250

17) SQRT() ⇒ Returns the square root of a number.

Ex ⇒ Select SQRT(64);

output ⇒ 8

18) MOD() ⇒ Returns the reminder of a number divided by another number.

Ex ⇒ Select MOD(18, 4)

Output ⇒ 2

$$4 \overline{)18} \quad 4 \quad 16 \quad 2$$

19) POW() or POWER() ⇒ Returns the value of a number raised to the power of another number.

Ex ⇒ Select POW(4, 2)

Output ⇒ 16

20) Radians() ⇒ Converts a degree value into radians.

Ex ⇒ Select Radians(180);

Output ⇒ 3.141592 -- $\frac{180 \times \pi}{180} = \pi = 3.141592$

21) ROUND() ⇒ Rounds a number to a specified number of decimal places.

Ex ⇒ Select Round(34.765, 2);

Output ⇒ 34.778 34.77

MYSQL Aggregate Functions

⇒ MYSQL's aggregate function is used to perform calculations on multiple values and return the result in a single value like the average of all values, the sum of all values, and maximum & minimum value among certain groups of values.

⇒ We mostly use the aggregate functions with Select statements in the data query Language.

⇒ Syntax ⇒

function_Name (DISTINCT / ALL expression)

⇒ first, we need to specify the Name of the aggregate function.

⇒ Second, we use the DISTINCT modifier when we want to calculate the result based on distinct values or ALL modifiers when we calculate the values, including duplicates. The default is ALL.

⇒ Third, we need to specify the Expression that involves Columns and arithmetic operators.

Some Aggregate functions are:⇒

1) Count() ⇒ It returns the Number of rows, including rows with NULL values in a group.

Syntax ⇒

Select Count(aggregate_Expression) from table_Name where Conditions;

Ex ⇒ Write the SQL Statement to find total number of students name available in Student table.

Soln ⇒ Select Count(s_name) from Student;

2) Sum() ⇒ It returns the total summed values (Non-NULL) in a set.

Syntax ⇒

Select Sum(aggregate_Expression) from table_Name where Conditions;

Ex ⇒ Write SQL Statement to find total marks of all students in student table.

Soln ⇒ Select Sum(marks) from student;

3) average() ⇒ It returns the average value of an expression.

Syntax ⇒
Select AVG (aggregate_Expression) from table_Name where Conditions;

Ex ⇒ Write SQL Statement to find average marks of all students in Student table.

Soln ⇒ Select AVG (marks) from Student;

4) min() ⇒ It returns the minimum (lowest) value in a set.

Syntax ⇒
Select min (DISTINCT aggregate_Expression) from table_Name where Conditions;

Ex ⇒ Write SQL Statement to find minimum marks of all students in Student table.

Soln ⇒ Select min (marks) from Student;

5) max() ⇒ It returns the maximum (highest) value in a set.

Syntax ⇒
Select max (DISTINCT aggregate_Expression) from table_name where Conditions;

Ex ⇒ write SQL Statement to find maximum marks of all Students in Student table.

Soln ⇒ Select max (marks) from Student;

6) Group_Concat() ⇒ This function is used to Concatenate string from multiple rows into a single string using various Clauses. If the group contains at least one non-null value, it always returns a string value. Otherwise, you will get a null value.

Syntax ⇒

Select *, Group_Concat(Distinct Expression) from table_name Group by Column_name;

Ex ⇒ Write SQL statement for Group_Concat in student table.

Query ⇒ Select *, Group_Concat (Subject) as Subject from student group by student_id;

7) first() ⇒ It returns the first value of an Expression.

Syntax ⇒

Select Column_name from table_name LIMIT 1;

Ex ⇒ Select * from Student LIMIT 1;

8) Last() ⇒ It returns the Last value of an Expression.

Syntax ⇒

Select Column_name from table_name order by Column_name DESC LIMIT 1;

Ex ⇒ Select * from Student Order By Stud_id DESC LIMIT 1;