

Introduction To SQL

Page No: 1

- 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 tool 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: $\Rightarrow$

- 1) Standard Independent Language: $\Rightarrow$ 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 $\Rightarrow$ 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: $\Rightarrow$ SQL statements resemble simple English sentences, making SQL easy to learn and understand.
- 4) Less programming: $\Rightarrow$ SQL allows extraction, manipulation and organisation of data with less programming as compared to traditional methods.

MYSQL/SQL Data Types

Page No. 3

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 $\Rightarrow$

a) Char(size) $\Rightarrow$ 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) $\Rightarrow$ 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) $\Rightarrow$ It holds a string that can contain a maximum length of 255 characters.

d) Binary(size) $\Rightarrow$ 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)** $\Rightarrow$ It is equal to Varchar() but Stores binary byte strings. Its size parameter specifies the maximum Column length in bytes.

f) **Tinytext** $\Rightarrow$ It hold a string with maximum length of 255 characters.

g) **mediumtext** $\Rightarrow$ It hold a string with maximum length of 16,777,215.

h) **Longtext** $\Rightarrow$ It hold a string with a maximum length of 4,294,967,295 characters.

2) **MySQL Numeric datatypes** $\Rightarrow$

a) **Bit (size)** $\Rightarrow$ 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)** $\Rightarrow$ It is used for the integer value. Its signed range varies from -2147483648 to 2147483647 and unsigned range varies from 0 to 4294967295. the size parameter specifies the max display width that is 255.

c) `Integer(size)` $\Rightarrow$ It's equal to `int(size)`.

d) `float(size, d)` $\Rightarrow$ 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` $\Rightarrow$ 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)` $\Rightarrow$ It is used to specify date and time combination. Its format is `YYYY-MM-DD hh:mm:ss`.

c) `Time(fsp)` $\Rightarrow$ It is used to specify the time format. Its format is `hh:mm:ss`.

d) `Year` $\Rightarrow$ 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 $\Rightarrow$

1) SQL Server String Datatype $\Rightarrow$

"Aman"

- a) **Char(n)** $\Rightarrow$ It is a fixed width character string datatype. Its size can be up to 8000 characters.
- b) **Varchar(n)** $\Rightarrow$ It is a variable width character string datatype. Its size can be up to 8000 character.
- c) **VarChar(max)** $\Rightarrow$ It is a variable width character string datatypes. Its size can be up to 1,073,741,824 characters.
- d) **Text** $\Rightarrow$ It is a variable width character string datatype. Its size can be up to 2GB of text data.
- e) **nchar** $\Rightarrow$ It is a fixed width Unicode string datatype. Its size can be up to 4000 Characters.
- f) **nvarchar** $\Rightarrow$ It is a variable width Unicode string datatype. Its size can be up to 4000 Characters.
- g) **ntext** $\Rightarrow$ It is a variable width unicode string datatype. Its size can be up to 2GB of text data.

b) **binary(n)** $\Rightarrow$ It is a fixed width Binary String data type. Its size can be up to 8000 bytes.

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

j) **image** $\Rightarrow$ It is also a variable width Binary String data type. Its size can be up to 2 GB.

2) **SQL Server Number Data types** $\Rightarrow$

a) **bit** $\Rightarrow$ It is an integer that can be 0, 1 or null.

b) **tinyint** $\Rightarrow$ It allows whole number from 0 to 255.

c) **Smallint** $\Rightarrow$ It allows whole number between -32,768 to 32,767.

d) **int** $\Rightarrow$ It allows whole number between -2,147,483,648 and 2,147,483,647.

e) **bigint** $\Rightarrow$

f) **float(n)** $\Rightarrow$ 12.56

g) **real** $\Rightarrow$ It is a floating precision number data from $-3.40E+38$ to $3.40E+38$.

3) SQL Server Date and time Data type $\Rightarrow$

- a) **datetime** $\Rightarrow$ 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** $\Rightarrow$ It is used to specify date and time combination. It supports range from January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds.
- c) **date** $\Rightarrow$ It is used to store date only. It supports range from January 1, 0001 to December 31, 9999.
- d) **time** $\Rightarrow$ It stores time only to an accuracy of 100 nanoseconds.
- e) **timestamp** $\Rightarrow$ 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 : 9

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 table name (columnname1 datatype, columnname2 datatype -- columnname n datatype);

Ex ⇒

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

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

way 1) Syntax $\Rightarrow$

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

Ex $\Rightarrow$

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

way 2) Syntax $\Rightarrow$

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

Ex $\Rightarrow$

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

SQL Command

Page No :

11

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

1) Whole table Selection $\Rightarrow$
Syntax $\Rightarrow$

Select * from tablename;

Ex $\Rightarrow$

Select * from Student;

2) Select/Access particular Column;

Syntax $\Rightarrow$

Select Columnname1, Columnname2 -- from tablename;

Ex $\Rightarrow$

Select name, ID from Student;

3) Select/Access particular Row;

Syntax $\Rightarrow$

Select * from tablename where Condition;

Ex $\Rightarrow$

Select * from Student where ID = 04;

5) **Update Statement** $\Rightarrow$ 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 $\Rightarrow$

Update table-name Set [Columnname1 = value1, Columnname2 = value2 ... Columnname3 = value3]
[where condition];

Ex $\Rightarrow$

Update ~~Student~~^{Set} name = "Aman", address = "Dehradun" where id = 04;

6) **Alter table Statement** $\Rightarrow$ 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 $\Rightarrow$

Alter table table-name Add Columnname datatype

Ex $\Rightarrow$

Alter table student add Email
Varchar(50);

SQL Command

13

Page No.

7) Delete Statement $\Rightarrow$ SQL Delete Statement is used to delete rows from a table.

Delete statement remove one or more records from a table.

Syntax $\Rightarrow$

Delete from table_name where Condition;

Ex $\Rightarrow$

Delete from Student where ID = 04;

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

Ex $\Rightarrow$ 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 $\Rightarrow$ (Truncate Statement) $\Rightarrow$

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

Syntax $\Rightarrow$ Truncate table tablename;

Ex $\Rightarrow$ Truncate table Student.

8) **Drop Command** $\Rightarrow$ 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 $\Rightarrow$
Drop table table-name;

Ex $\Rightarrow$
Drop table Student;

9) **Drop Column** $\Rightarrow$

Syntax $\Rightarrow$ alter table table-name
drop Column Column-name;

Ex $\Rightarrow$
alter table Student drop Column
name;

SQL Operators

15

Page No :

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, intersect, join
- 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 (%)

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

SOL Operators

19

Page No :

3) SOL Logical Operators ⇒

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

SOL Logical operators are:-

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

1) SOL ALL operator ⇒ The all operator in SOL compares the specified value to all the values of a column from the sub-query in the SOL 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).

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

Query⇒ `SELECT Emp-id, Emp-Name FROM Employee WHERE Emp-Salary > ALL (SELECT Emp-Salary FROM Employee WHERE City = "Haridwar");`

2) SQL AND Operator ⇒

The AND operator in SQL would show the record from the database table if all the conditions separated by the AND operator evaluated to True.

It is also known as the Conjunctive operator and is 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 is 25000 and city is Haridwar.

Query⇒ `SELECT * FROM Employee WHERE Emp-Salary = 25000 AND Emp-city = "Haridwar";`

3) **SQL OR Operator** $\Rightarrow$ 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 $\Rightarrow$

Select Column1, Column2 -- Column n from
table-name where Condition 1 OR Condition 2
--- OR Condition N;

4) **SQL BETWEEN operator** $\Rightarrow$ 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 $\Rightarrow$

Select Column1, Column2 -- Column n from
table-name where Columnname BETWEEN
Value1 and value 2;

5) **SQL IN operator** $\Rightarrow$ 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 $\Rightarrow$

Select Column1, Column2 -- Columnn from table-Name
where Column-name In (List of values);

6) **SQL NOT operator** $\Rightarrow$ 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 $\Rightarrow$

Select Column1, Column2 --- ColumnN from
table-Name where NOT Condition;

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

Query $\Rightarrow$ Select * from Employee where Not
Emp-city = 'dehradun';

7) SQL ANY Operator $\Rightarrow$ 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 $\Rightarrow$

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 $\Rightarrow$ 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 $\Rightarrow$

Select Column1, Column2 -- ColumnN from
table-Name where Columnname Like pattern;

Ex $\Rightarrow$ We want to show all the information of those employees from the Employee table whose name starts with "K".

query $\Rightarrow$ Select * from Employee where
Emp-Name Like 'K%';

query $\Rightarrow$ Select * from Employee where
Emp-Name Like '%t';

query $\Rightarrow$ Select * from Employee where
Emp-Name Like '_a%';

SQL Operators

23

Page No:

--	--	--

4) SQL Set Operators! $\Rightarrow$ • 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! $\Rightarrow$ • 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.

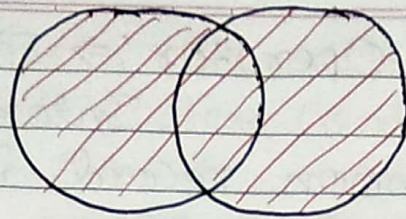


Table 1 Table 2

Syntax: $\Rightarrow$

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

- 2) SQL Union ALL operator: $\Rightarrow$ The SQL Union ALL operator is the same as the Union operator. But it also shows the duplicate row.

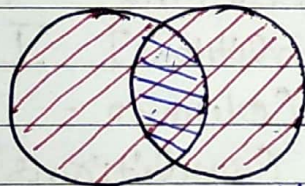


Table 1 Table 2

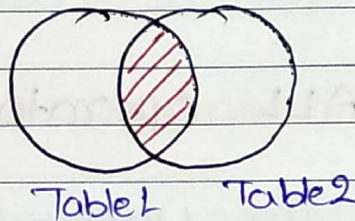
Syntax: $\Rightarrow$

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

3) **SQL Intersect Operator** $\Rightarrow$ The SQL Intersect 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.

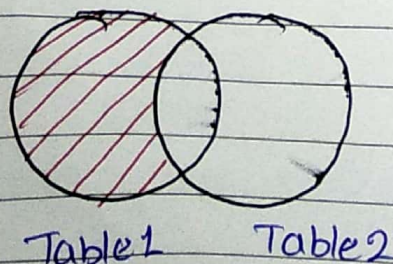
NOTE $\Rightarrow$ MySQL does not Support Intersect operator.



Syntax $\Rightarrow$

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

4) **SQL Minus Operator** $\Rightarrow$ SQL minus Operator Combines the result of two or more SELECT Statements and Shows only the results from the first data set.



Syntax $\Rightarrow$ $\text{Select Column1, Column2 -- ColumnN from table1 [where Conditions] Minus Select Column1, Column2 -- ColumnN from table2 [where Conditions];}$

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 such as 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 handle complex queries. |

SOL

- 7) SOL database is not the best choice for storing hierarchical database.
- 8) All SOL database require object-relational mapping.
- 9) SQLite, MS-SQL, Oracle, PostgreSQL and MySQL are Example of SOL database Systems.

No-SOL

- 7) No-SOL database is a perfect option for storing hierarchical database.
- 8) many No-SOL database do not require object-relational mapping.
- 9) Redis, MongoDB, Hbase, Bigtable, CouchDB are Example of NoSQL database System.

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

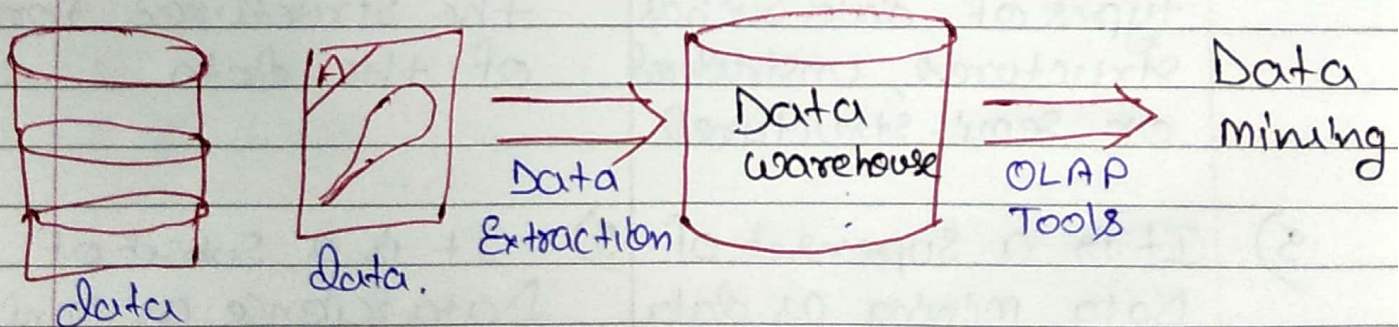
7) The data mining techniques are cost-efficient as compared to other statistical data applications.

8) One of the most amazing data mining technique is the detection and identification of the unwanted errors that occur in the system.

Data warehousing

7) The responsibility of the data warehouse is to simplify every type of business data.

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

- 1) Data Science is an Area.
- 2) It is about Collection, processing, analyzing and Utilizing of data into various operations. It is more Conceptual.
- 3) It is a field of study just like the Computer Science, Applied Statistics or Applied Mathematics.
- 4) It deals with all types of data such as structured, unstructured or semi-structured.
- 5) It is a Superset of Data Mining as data Science consists of Data Scrapping, Cleaning, Visualization, Statistics and many more techniques.

data mining

- 1) Data mining is a Technique.
- 2) It is about Extracting the vital and valuable information from the data.
- 3) It is a technique which is a part of the Knowledge Discovery in Database processes (KDD).
- 4) It mainly deals with the structured forms of the data.
- 5) It is a Subset of Data Science as mining activities which is in a pipeline of the Data Science.

Data Science

6) It is mainly used for Scientific purposes.

7) It broadly focuses on the Science of the data

Data mining

6) It is mainly used for business purposes

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 populate 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 affect 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 i'n 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) i'n the database.	3) Update Command is used to update existing records i'n a database.
4) Alter Command by default initializes values of all the tuple as NULL.	4) Update Command sets specified values i'n the Command to the tuples.
5) This Command make Changes with table Structure	5) This Command make Changes with data inside the table.

Alter

6) It works on the attributes of a relation.

7) Syntax with Example

Syntax:

Alter table tableName

Drop Column columnName

Ex ⇒

Alter table Student

Drop column address;

Update

6) It works on the attribute of a particular tuple in a table.

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 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

39

- ⇒ 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

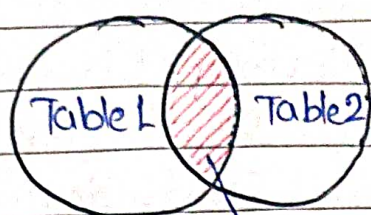
3) right outer join

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.



↓ inner join

Syntax $\Rightarrow$

Select table1.Column1, table1.Column2, table2.Column1, --- from table1 Inner join

table2 On table1.matching - Column = table2.matching Column;

NOTE $\Rightarrow$ We can also write Join instead of INNER JOIN. JOIN is same as INNER JOIN.

Ex $\Rightarrow$ Consider the two tables
Student (Table 1)

Roll-no	Name	Address	Age
1	Kailash	Behradun	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

Query $\Rightarrow$ 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 = StudentCourse, Rollno;

Output $\Rightarrow$

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

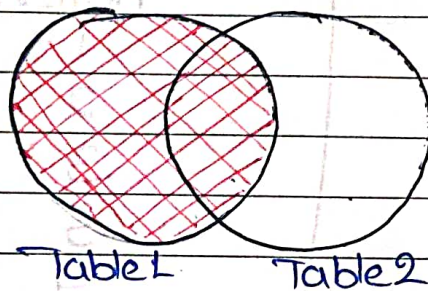
Page No.:

42

⇒ 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 table 1 LEFT  
JOIN table2 On table1.matching-column =  
table2.matching-column;
```

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

Ex $\Rightarrow$ Consider the two tables.

Student (Table 1)

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 $\Rightarrow$ To Access the all Student Name field with given Course ID.

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

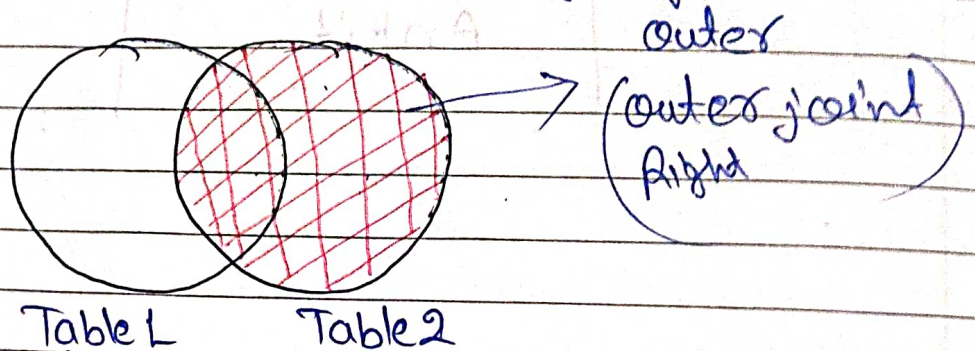
Output

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

SOL RIGHT JOIN

45

⇒ 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 matching in both tables, it returns NULL. It is also known as right 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	Behradun	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 $\Rightarrow$ To Access ^{All} Student ID with Name.

Command $\Rightarrow$ Select Student.Name, StudentCourse.Course-ID
from Student Right Join StudentCourse
ON StudentCourse.Rollno = Student.Rollno;

Output.

Name	Course-ID
Kailash	1
Kamal	2
Karan	2
Ankit	3
NULL	1

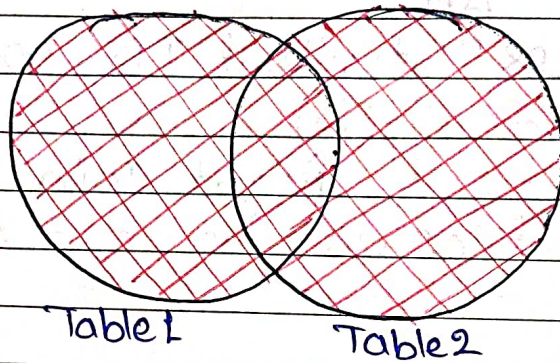
Select * from Student Right Join StudentCourse
ON StudentCourse.Rollno = Student.Rollno;

SQL FULL JOIN / FULL OUTER JOIN

(47)

⇒ 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	Behradun	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 $\Rightarrow$ Select Student, Name, StudentCourse, Course-ID
 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 join StudentCourse
 on StudentCourse.Rollno = Student.Rollno

Rollno.	Name	address	Age	CourseID
1	Kailash	Del	30	1
2	Kamal	Del	25	2
3	Karan	Rn	18	2
4	Aman	hwr	19	NULL
5	Ankit	n	20	3
9	NULL	NULL	NULL	1

SQL Cross Join

(49)

⇒ SQL Cross Join Combine two Different tables, ~~later~~ 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 table 1 and the number of rows present in table 2.

Syntax ⇒

$$\begin{matrix} \text{table 1 (3)} \\ \text{table 2 (3)} \end{matrix} \Rightarrow 3 \times 3 = 9$$

Select table Name 1. Column 1, table 1.
Column 2, table 2 Column 1 ... from table 1
CROSS JOIN table 2 ON Table 1. matching-
Column = table 2. matching Column;

Ex ⇒ Consider the two table.

Student (table 1)

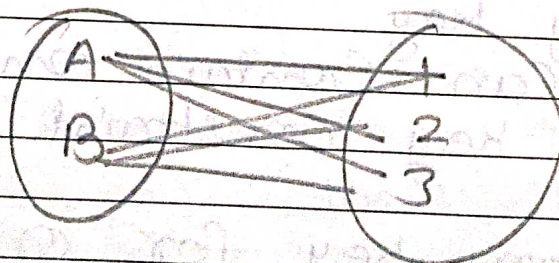
Roll no	Name	address	Age
1	Kailash	Dehradun	30
2	Karan	haridwar	25
3	Kamal	Rishikesh	18

StudentCourse (table 2)

Course-ID	Rollno.
1	1
2	3
3	5

Query $\Rightarrow$ Select * from Student
Cross Join StudentCourse;

Rollno	Name	Address	Age	Course ID	Rollno
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



SQL Primary Key

(51)

- ⇒ 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 $\Rightarrow$

```
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 $\Rightarrow$ 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:

MYSOL ⇒

```
Alter table Student Drop Primary  
key;
```


SQL Server, Oracle, MS Access :

Alter table Student Drop
Constraint pk_StudentID;

SQL foreign key

55

- ⇒ 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.

P.K first table (Students)

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

Second table (Orders)

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_PerOrders
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 $\Rightarrow$

Primary key

1) Primary key uniquely identify a record in the table.
Ex Aadhar no.

2) Primary key Can not Accept Null values.

3) By default, Primary key is clustered index and data in the database table is physically organized in the sequence of clustered index.

4) We can have only one Primary key in a table.

5) Primary key is always unique

foreign key

1) foreign key is a field in the table that is primary key in another table.
Ex | adhar no. - AC | ID (PK)

2) foreign key Can accept multiple null value.

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 more than one foreign key in a table.

5) foreign key Can be duplicated.

SQL Composite key



(59)

- ⇒ 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 guaranteed, but when it is 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 $\Rightarrow$

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

Rollno. Name — Key

(Composite Key)

Unique key in SQL

Page No. 61

- ⇒ 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's 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 varchar(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

63

⇒ 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 datatype.

Some of the important string functions 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-Length("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 separator.

Ex $\Rightarrow$ Select Concat_WS("-", 'Kailash', 'Joshi');

Output $\Rightarrow$ Kailash_Joshi

5) field() $\Rightarrow$ Return the index position of a value in a list of values.

Ex $\Rightarrow$ Select field('a', 'x', 'y', 'a', 'b');

Output $\Rightarrow$ 3

6) find-in-set() $\Rightarrow$ Return the position of a String within a list of Strings.

Ex $\Rightarrow$ Select field-in-set("a", "x,a,p,q");

Output $\Rightarrow$ 2

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

Ex $\Rightarrow$ Select format(2354.5634, 2);

Output $\Rightarrow$ 2354.56

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

Ex $\Rightarrow$ Select insert("CSEngineering.com", 1, 13, "kailash");

Output $\Rightarrow$ Kailash.Com

9) `LCASE()` $\Rightarrow$ Converts a String to Lower-case;

Ex $\Rightarrow$ Select `LCASE("KAILASH")`;

Output $\Rightarrow$ kailash

10) `LOWER()` $\Rightarrow$ Convert a String to Lower-case
Same as `LCASE()`.

11) `UPPER()` $\Rightarrow$ Convert a String to Upper-case

Ex $\Rightarrow$ Select `UPPER("kailash")`;

Output $\Rightarrow$ KAILASH

12) `UCASE()` $\Rightarrow$ Convert a String to Upper-case
Same as `UPPER()`.

13) `LTRIM()` $\Rightarrow$ Remove Leading Space from
a String.

Ex $\Rightarrow$ Select `LTRIM(" Kailash Joshi ")`;

Output $\Rightarrow$ kailash Joshi

14) `RTRIM()` $\Rightarrow$ Remove trailing Space from
a String.

15) `TRIM()` $\Rightarrow$ Remove Leading and trailing
Space from a String.

16) Reverse() $\Rightarrow$ Reverse a string and return the result.

Ex $\Rightarrow$ Select Reverse ("Kailash"),

output $\Rightarrow$ hsaliäk.

17) Replace() $\Rightarrow$ Replace all occurrences of a Substring within a string, with a new Substring.

Ex $\Rightarrow$ Select replace ("Kailash Joshi", "Kailash", "Kamal")

output $\Rightarrow$ Kamal Joshi.

18) Substr() $\Rightarrow$ Extracts a Substring from a string (starting at any position)

Ex $\Rightarrow$ Select Substr ("Kailash Joshi", 9, 3),

output $\Rightarrow$ Jos

SQL/MYSQL Numeric function

(67)

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

Some of the important String function are:-

1) **ABS()** $\Rightarrow$ Returns the absolute value of numeric expression.

Ex $\Rightarrow$ `Select ABS(-235);`

output $\Rightarrow$ 235

2) **ACOS()** $\Rightarrow$ Returns the arccosine of a numeric expression. Returns NULL if the value is not in the range -1 to 1.

Ex $\Rightarrow$ `Select ACOS(0.8);`

output $\Rightarrow$ 0.6435011087932843

3) **ASIN()** $\Rightarrow$ Returns the arcsine of numeric expression. Returns NULL if value is not in the range -1 to 1.

Ex $\Rightarrow$ `Select ASIN(0.8);`

output $\Rightarrow$ 0.927295180016123

4) $\text{ATAN}()$ $\Rightarrow$ Returns the arctangent of numeric expression.

Ex $\Rightarrow$ $\text{ATAN}(0.8)$

Output $\Rightarrow$ 0.6747409422235527

5) $\text{AVG}()$ $\Rightarrow$ Returns the average value of an expression.

Ex $\Rightarrow$ Select Avg(marks) from student;

Output $\Rightarrow$ 60

6) $\text{CEIL}()$ $\Rightarrow$ Returns the smallest integer value that is $\geq$ to a number.

7) $\text{CEILING}()$ $\Rightarrow$ Returns the Smallest integer value that is $\geq$ to a number.

Ex $\Rightarrow$ Select $\text{CEIL}(97.75)$;

or

Select $\text{CEILING}(97.75)$;

Output $\Rightarrow$ 98

8) $\text{FLOOR}()$ $\Rightarrow$ Returns the Largest integer value that is $\leq$ to a number.

Ex $\Rightarrow$ Select floor(97.75);

Output $\Rightarrow$ 97

9) Degree() $\Rightarrow$ Converts a value in radians to degrees.

Ex $\Rightarrow$ Select Degree (1.5)

$$\begin{aligned} \text{Output} \Rightarrow 85.943669 & \left\{ \begin{aligned} 1.5 \times \frac{180}{\pi} &= 1.5 \times \frac{180}{3.14159} \\ &= 1.5 \times 57.2958 \\ &= 85.9436 \end{aligned} \right. \end{aligned}$$

10) DIV() $\Rightarrow$ Used to Integer division.

Ex $\Rightarrow$ Select 10 DIV 5;

Output $\Rightarrow$ 2

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

Ex $\Rightarrow$ Select greatest (10, 20, 3, 4, 18);

Output $\Rightarrow$ 20

12) Least() $\Rightarrow$ Return the Smallest value of the list of arguments.

Ex $\Rightarrow$ Select Least (10, 20, 3, 4, 18);

Output $\Rightarrow$ 3

13) $\text{LOG}()$ $\Rightarrow$ Return the natural logarithm of a number, or the Logarithm of a number to a specified base.

Ex $\Rightarrow$ Select $\text{Log}(2),$

output $\Rightarrow 0.693147..$

14) $\text{LOG}_{10}()$ $\Rightarrow$ Return the natural Logarithm of a number to base 10,

Ex $\Rightarrow$ Select $\text{Log}_{10}(2),$

output $\Rightarrow 0.30102999..$

15) $\text{LOG}_2()$ $\Rightarrow$ Returns the natural logarithm of a number of base 2.

Ex $\Rightarrow$ Select $\text{Log}_2(2),$ $\log_2 2 = 1$

output $\Rightarrow 1$

16) $\text{SUM}()$ $\Rightarrow$ Calculates the sum of a set of values

Ex $\Rightarrow$ Select $\text{Sum}(\text{marks})$ from students,

output $\Rightarrow 250$

17) $\text{SQRT}()$ $\Rightarrow$ Returns the square root of a number.

Ex $\Rightarrow$ Select $\text{SQRT}(64),$

output $\Rightarrow 8$

18) MOD() $\Rightarrow$ Returns the remainder of a number divided by another number.

Ex $\Rightarrow$ Select MOD(18, 4)

Output $\Rightarrow$ 2

$$\begin{array}{r} 4 \\ 4 \overline{) 18} \\ \underline{16} \\ 2 \end{array}$$

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

Ex $\Rightarrow$ Select POW(4, 2)

Output $\Rightarrow$ 16

20) Radians() $\Rightarrow$ Converts a degree value into radians.

Ex $\Rightarrow$ Select Radians(180);

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

21) ROUND() $\Rightarrow$ Rounds a number to a specified number of decimal places.

Ex $\Rightarrow$ Select Round(34.765, 2);

Output $\Rightarrow$ 34.77

MYS@L Aggregate Functions

72

⇒ MYS@L'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: $\Rightarrow$

- 1) Count () $\Rightarrow$ It returns the Number of rows, including rows with NULL values in a group.

Syntax $\Rightarrow$

Select Count (aggregate Expression) from
table-Name where Conditions;

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

Solⁿ $\Rightarrow$ Select Count (s-name) from Student;

- 2) Sum () $\Rightarrow$ It returns the total Summed values (Non-NULL) in a set.

Syntax $\Rightarrow$

Select Sum (aggregate Expression) from
table-Name where Conditions;

Ex $\Rightarrow$ Write SQL Statement to find total marks of all Students in Student table.

Solⁿ $\Rightarrow$ Select Sum (marks) from Student;

3) average() $\Rightarrow$ It returns the average value of an expression.

Syntax $\Rightarrow$
 Select AVG (aggregate - Expression) from
 table - Name where Conditions;

Ex $\Rightarrow$ Write SQL Statement to find average marks of all students in Student table.

Solⁿ $\Rightarrow$ Select AVG (marks) from Student;

4) min() $\Rightarrow$ It returns the minimum (lowest) value in a set.

Syntax $\Rightarrow$
 Select min (DISTINCT aggregate - Expression)
 from table - Name where Conditions;

Ex $\Rightarrow$ Write SQL Statement to find minimum marks of all students in Student table.

Solⁿ $\Rightarrow$ Select min (marks) from Student;

5) max() $\Rightarrow$ It returns the maximum (highest) value in a set.

Syntax $\Rightarrow$
 Select max (DISTINCT aggregate - Expression)
 from table - name where Conditions;

Ex $\Rightarrow$ Write SQL Statement to find maximum marks of all students in Student table.

Solⁿ $\Rightarrow$ Select max (marks) from student;

- 6) Group-Concat() $\Rightarrow$ 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 $\Rightarrow$

Select *, Group-Concat (Distinct Expression)
from table-name Group by Column-name;

Ex $\Rightarrow$ Write SQL statement for Group-Concat in student table.

Query $\Rightarrow$ Select *, Group-Concat (Subject)
as Subject from student group by
student-id;

- 7) first() $\Rightarrow$ It returns the first value of an Expression.

Syntax $\Rightarrow$

Select Column-name from table-name LIMIT 1;

Ex $\Rightarrow$ Select * from student LIMIT 1;

- 8) Last() $\Rightarrow$ It returns the last value of an Expression.

Syntax $\Rightarrow$

Select Column-name from table-name Order By
Column-name DESC LIMIT 1;

Ex $\Rightarrow$ Select * from student Order By
studid DESC LIMIT 1;