**SQL**

**What is SQL:** SQL is a stand for “Structures quires language”. **Insert, Update, Modify, Delete.**

**Feature of SQL:**

* Popular and in demand technology**,**
* Manages and retrieves data from database,
* Structure and flexible query usage,
* Efficiency in handling high volume data.

**Type of SQL commands:** **Commands**

Data Definition Language (DDL) = **CREATE, ALTER, DROP, RENAME, TRUNCATE**

Data Manipulation Language (DML) = **INSERT**, **SELECT, UPDATE, DELETE**

Data Control Language (DLC) = **GRANT, REVOKE, CALL, RETURN**

Transaction Control Language(TCL) = **COMMIT, ROLLBACK, SAVEPOINT**

**Advantages of SQL:**

* High speed and fast query processing
* Allows multiple views
* Open source
* Efficient and Portable

**Disadvantages of SQL:**

* Complex Interface
* High Operating Cost
* High Maintenance

**Conclusion of SQL:**

* Language to operate databases.
* Simple and Easy to learn
* Minimal coding

**Applications:** **E**ducation, **H**ealthcare, **R**etail & **E-**commerce, **B**anking, **F**inance.

**DBMS**

**What is Database Management System (DBMS)?**

DBMS is a software to store **DATABASE** and **MANAGEMENT SYSTEM**.

**Components of DBMS:**

* Data
* Hardware
* Software
* User Access
* Procedure

**Type of DBMS Architecture:**

**1**-Tier Architecture: **END USER -> DIRECT ACCESS -> DATABSAE SYSTEM**

**2**-Tier Architecture: **USER -> APPLICATION -> SYSTEM**

**3**-Tier Architecture: **USER -> APPLICATION -> APPLICATION SEVER -> SYSTEM**

**RDBMS vs DBMS:**

**RDBMS**

* Store Data in **tables**
* Handles **Large** amount of Data
* Support **Multiple** Users
* Support Distributed Database
* **Ex:-** XML, Windows Registry etc.

**DBMS**

* Store Data in **Files**
* Handles **Small** Amount of Data
* Support **Single** Users at a Time
* **No** Support Distributed Database
* **Ex:-** MySQL, SQL, Server, Oracle, etc.

**Type of Keys in DBMS:**

* Primary Key (field is not null)
* Super Key
* Candidate Key (or minimal super key)
* Foreign Key – relationship between two tables

**Advantages of DBMS:**

* Reduced Data Redundancy
* Data Integrity
* Data Privacy & Security
* Data Consistency
* Easy of Sharing Data
* Backup and Recovery

**Disadvantages of DBMS:**

* High Cost and Maintenance
* Complexity
* Speed & Performance Issue
* Increase Vulnerability

**SQL Datatypes:** **Numeric** = INT, BIT, FLOAT, BOOLEAN

**String Data** = CHAR, VARCHAR, TEXT

**Date Time** = DATE(YYYY-MM-DD), DATETIME (YYYY-MM-DD HH**:**MM**:**SS), TIMESTAMP

**SQL Operators: A**rithmetic, **L**ogical, **C**omparison, **B**itwise, **S**et.

**Data Definition Language (DDL): CREATE, ALTER, DROP, TRUNCATE**

**Syntax: CREATE TABLE** *table\_name*(“col\_1”“datatype”,“col\_2”“datatype”,“col\_3”“datatype”…..);

Ex:- **CREATE TABLE** *Employee* ( Emp\_id **INT**, Name **VARCHAR**, Address **VARCHAR** ) ;

**Syntax: ALTER TABLE** *table\_name* **ADD** (col\_name1 datatype, col\_name2 datatype, col\_name3 datatype….);

Ex:- **ALTER TABLE** *Employee* **ADD** (date of birth **DATE**); (r DOB)

Ex2:- **ALTER TABLE** *Employee* **DROP COLUMN** date of birth; (header blank)

**Syntax: DROP TABLE** *table\_name;* (table employee does not exit)

Ex:- **DROP TABLE** *Employee*;

**Syntax: TRUNCATE TABLE** *table\_name;* (NULL in table)

Ex:- **TRUNCATE TABLE** *Employee*;

**Data Manipulation Language (DML): INSERT, SELECT, UPDATE, DELETE**

**Syntax-1: INSERT INTO** *table\_name* **VALUES** (value1, value2, value3……);

**Syntax-2: INSERT INTO** *table\_name* (column1, column2, column3….) **VALUES** (value1, value2, value3….);

Ex:- **CREATE TABLE** *Students*;

Ex:- **INSERT INTO** *Students* (roll\_no, name, age, city) **VALUES** (1, Rohan, 22, Hyderabad);

**Syntax-1: SELECT** column\_1, column\_2…column\_N *table\_name*;

**Syntax-2: SELECT \* FROM** *table\_name*;

**Syntax-3: SELECT \* FROM** *table\_name* **WHERE** C*ol\_name* = “location\_name”;

Ex:- **SELECT** Roll\_no, Name, Age, **FROM** *Student*;

Ex:- **SELECT \*** **FROM** *Student*;

Ex:- **SELECT \* FROM** *Employee* **WHERE** City= “*New Delhi*”;

**Syntax-1: UPDATE** *table\_name* **SET** col\_name = ‘*new\_value’* **WHERE** C*ol\_name* = “location\_name”;

Ex:- **UPDATE** Student**SET** City = ‘*Chennai’* **WHERE** Roll\_no = ‘3’;

**Syntax-1: DELETE FROM** *table\_name* ;

Ex:- **DELETE FROM** Student;

Ex2:- **DELETE FROM** Student

**WHERE** Roll\_no = ‘1’

**Data Control Language (DLC): GRANT, REVOKE**

**Syntax-1: GRAND <**privilege list**>**

**ON** <table\_name>

**TO** <user>;

**Ex:** **GRAND INSERT, SELECT ON** Employee **TO** Rahul;

**Syntax-1: REVOKE <**privilege list**>**

**ON** <table\_name>

**FROM** <user>**;**

**Ex:** **REVOKE INSERT ON** Employee **FROM** Rahul;

**SQL Select, Create, Use, Drop, Delete, Truncate, Alter, Database:**

|  |  |
| --- | --- |
| **SELECT** *col\_1, col\_2,…*  **FROM** *table\_name* **WHRE** *[EXPRESSION]*; | Display a table |
| **CREATE** **DATABASE** database\_name; | Create a database |
| **DROP** **DATABASE** database\_name; | Drop a database |
| **USE** database\_name; | Use a database |
| **CREATE** **TABLE** table\_name; | Create a Table |
| **DROP** **TABLE** table\_name; | Drop a Table |
| **INSERT** **INTO** table­\_name **VALUES** (value\_1, value\_2….) | Insert new row value |
| **INSERT** **INTO** table­\_name ( column1, column2, column3,…..)  **VALUES** (value1, value2, value3,….); | Insert new column & value |
| **TRUNCATE** **TABLE** table\_name; | NULL (delete all value of table) |
| **ALTER** **TABLE** table\_name **ADD** date\_birth **DATE**; | Create new column |
| **ALTER** **TABLE** table\_name **DROP** col\_name; | Delete column |
| **ALTER** **TABLE** table\_name **RENAME** **TO** new\_table\_Name; | Rename table |
| **CREATE** **TABLE** new\_table **SELECT** \* **FROM** old\_table; | Copy a table |
| **CREATE** **TABLE** new\_table **SELECT** \* **FROM** database\_name**.**old\_table; | Copy a table |
| **UPDATE** table\_name **SET** col\_name=‘new\_value’ ,… **WHERE** id=‘location’; | Update new value |
| **DELETE** **FROM** table\_name **WHERE** id = '102'; | Row delete |
| **SELECT** col\_name(s),... **FROM** table\_name  **WHERE** col\_name **IN** (value1, value2…); | Atleast one column, exact value you want |
| **SELECT** **COUNT**(**DISTINCT** city) **FROM** employee; | ( count no. of **Unique** city ) |
| SELECT \* FROM employee LIMIT 3; | (show **first 3** employee) |
| SELECT \* FROM employee ORDER BY salary DESC LIMIT 3; | **descending** order **last 3** employee |
| SELECT \* FROM employee ORDER BY **RAND()** LIMIT 4; | (**limit** all table 4) |
| SELECT \* FROM employee WHERE id IN (100, 099, 098); |  |
| SELECT \* FROM employee WHERE id BETWEEN '104' AND '108'; | (**from** 104 to 108) |
| SELECT **SUM**(salary) FROM employee WHERE city = 'mumbai'; | (total **each sum**) |
| SELECT \* FROM student WHERE DOB IS NULL; | (**NULL** value) |
| SELECT \* FROM student WHERE city = 'pune'; | (**Boolean** value) |
| SELECT \* FROM student WHERE total\_marks = '877'; |  |
| SELECT \* FROM student WHERE total\_marks / 2 > 480; | (**Numeric** operation) |
| SELECT **AVG** (total\_marks) FROM student; | **Average** |
| SELECT **SUM**(total\_marks) FROM student; | **Sum** |
|  |  |
| SELECT **current\_timestamp**(); | (**Date + Time**) |

**Addition, Subtraction, Multiplication, Division**

SELECT id, name, salary, salary **+** 10000 **as** new\_emp FROM employee;

SELECT id, name, salary, salary **-** 2000 **as** new\_emp FROM employee;

SELECT id, name, salary, salary **\*** 2 **as** new\_emp FROM employee;

SELECT id, name, salary, salary **/** 2 **as** new\_emp FROM employee;

**Equal to, NOT-Equal to, Greater than, Less than, Greater than Equal to, Less than Equal to**

SELECT \* FROM employee where salary **=** 40000;

SELECT \* FROM employee where salary **!=** 40000;

SELECT \* FROM employee where id **>** 104;

SELECT \* FROM employee where id **<** 105;

SELECT \* FROM employee where salary **>=** 40000;

SELECT \* FROM employee where salary **<=** 40000;

**AND, OR, BETWEEN, NOT condition operator**

SELECT \* FROM employee WHERE salary **>** 25000 **AND** city = 'hyderabad';

SELECT \* FROM employee WHERE salary **<** 30000 **OR** city = 'bangalore';

SELECT \* FROM employee WHERE salary **BETWEEN** 25000 **AND** 35000 ;

SELECT \* FROM employee WHERE **NOT** salary = 40000

**Types of SQL JOINS**

* **Inner joins:** Return records that have **matching** values in **both tables**.
* **Outer join:**
* **Left joins:** Return all records from **left table**, and the **matched** records from the right tables.
* **Right joins:** Return all records from **right table**, and the **matched** records from the left tables.
* **Cross joins:**

**Type of Aggregate Function:**

* COUNT() counts the number of non-null records in a field
* AVG() calculates the average of all non-null values belonging to a certain column of a table
* MIN() returns the minimum value from the entire list
* MAX() returns the maximum value from the entire list
* SUM() sums all the non-null values in a column

**Type of Scalar Function:**

* LCASE() convert string column value to **Lowercase**.
* UCASE() convert string column value to **Uppercase**.
* LENGTH() return the **length of text** values in the table.
* MID() Extract substring.(tab\_nam, st\_position, lenth)
* ROUND() rounds off **numeric** values.
* NOW() return the **current system** **date** and **time**.(DATETIME)
* FOMATE() **format** how a field must be displayed.

**Type of Scalar Function:**