Introduction to SQL

Introduction to SQL and Its Data Types.

SQL (Structured Query Language) is a standard programming language specifically for managing and manipulating relational databases. It is used to create, read, update, and delete data in a structured way. SQL data types help define the kind of data that can be stored in each column of a table. Here are

SQL data types:

1. Numeric Data Types

o INT: Integer numbers, e.g., 1, 100, -20.

o DECIMAL (p, s) or NUMERIC: Fixed precision numbers with specified digits after the decimal.

o FLOAT and REAL: For floating-point numbers (decimal numbers with variable precision).

2. Character Data Types

o CHAR(n): Fixed-length strings (e.g., CHAR(10) reserves 10 characters).

o VARCHAR(n): Variable-length strings (e.g., VARCHAR(50) allows up to 50 characters).

o TEXT: Large amounts of text.

3. Date and Time Data Types

o DATE: Stores date values (year, month, day).

o TIME: Stores time values (hours, minutes, seconds).

o DATETIME: Stores both date and time values.

o TIMESTAMP: Stores date and time with time zone info.

4. Boolean Data Types

o BOOLEAN: Stores true/false values.

5. Binary Data Types

o BLOB: Stores binary data, often used for images or files.

SQL Command Categories: DDL, DML, and DCL

SQL commands are organized into categories based on their purpose:

1. DDL (Data Definition Language)

o Used to define and manage database structures.

o Common DDL commands:

- CREATE: Creates a new database, table, or other objects.
- ALTER: Modifies an existing database object, such as adding a column.
- DROP: Deletes a database object like a table or view.
- TRUNCATE: Removes all rows from a table without logging individual row deletions.

2. DML (Data Manipulation Language)

o Used to interact with data within tables.

o Common DML commands:

- SELECT: Retrieves data from one or more tables.

- INSERT: Adds new rows to a table.
- UPDATE: Modifies existing data within a table.
- DELETE: Removes rows from a table.

3. DCL (Data Control Language)

o Used to manage permissions and control access to data.

o Common DCL commands:

- GRANT: Gives a user access privileges to a database or table.
- REVOKE: Removes access privileges from a user.

_

Experiment 1

Q1: Create the following tables

Student

Column_name	Data type	Size	Constraint
StudentId	Number	4	Primary Key
Student name	Varchar2	40	Not null
Address1	Varchar2	300	
Gender	Varchar2	15	
Course	Varchar2	8	

Course:

CourseID	Data type	Size	Constraint
DeptNo	Number	2	Primary Key
Dname	Varchar2	20	
Location	Varchar2	10	

Student Table

create table Student (StudentId integer (4) primary key,

Student name varchar(40) ,

Address1 varchar(300),

Gender varchar(15),

Course varchar(8));

Table name Course

CREATE TABLE Courses (

CourseID VARCHAR(12),

Dept_No INT,

D_name VARCHAR(20),

Location VARCHAR(100)

);

1. Insert five records for each table.

Student Table

Input

Run SQL

```
insert into Student values (1001, "Kiara", "Sector86", "Female", "BCA");
insert into Student values(1002, "Aryan Bhatia", "Sector7", "Male", "BCA");
insert into Student values (1003, "Puja", "jawahar colony", "Female", "BCA");
insert into Student values (1004, "Sivansh Singh", "NIT-1", "Male", "MCA");
insert into Student values (1005, "Rishi Kapoor", "Sec15-A", "Male", "BCA");
```

Student

Student

StudentId	StudentName	Address1	Gender	Course
1001	Kiara	Sector86	Female	BCA
1002	Aryan Bhatia	Sector7	Male	BCA
1003	Puja	jawahar colony	Female	BCA
1004	Sivansh Singh	NIT-1	Male	MCA
1005	Rishi Kapoor	Sec15-A	Male	BCA

Course table

Input

Run SQL

```
insert into Courses values ("1001A", 1, "SCA", "C block");
insert into CourseS values  ("1002A", 2, "SCA", "C block");
insert into Courses values ("1003A", 3, "SCA", "C block");
insert into Courses values ("1004A", 4, "SCA", "C block");
insert into Courses values ("1005A", 5, "SCA", "C block");
```

Courses

Output

CourseID	Dept_No	D_name	Location
1001A	1	SCA	C block
1002A	2	SCA	C block
1003A	3	SCA	C block
1004A	4	SCA	c block
1005A	5	SCA	C block

2. List all information about all students from student

table.

Input

Run SQL

```
select*from Student
```

Output

StudentId	Student	Address1	Gender	Course
1001	Kiara	Sector86	Female	BCA
1002	Aryan Bhatia	Sector7	Male	BCA
1003	Puja	jawahar colony	Female	BCA
1004	Sivansh Singh	NIT-1	Male	MCA
1005	Rishi Kapoor	Sec15-A	Male	BCA

3. List all student numbers along with their Courses.

Input

```
SELECT STUDENTid, course from Student
```

Output

StudentId	Course
1001	BCA
1002	BCA
1003	BCA
1004	MCA
1005	BCA

4. List Course names and locations from the Course table.

Input

```
SELECT D_name, Location from CourseS;
```

Output

D_name	Location
SCA	C block
SCA	C block
SCA	C block
SCA	c block
SCA	C block

5. List the details of the Students in MCA Course.

Input

```
SELECT*FROM STUDENT WHERE COURSE = 'MCA';
```

Output

StudentId	Student	Address1	Gender	Course
1004	Sivansh Singh	NIT-1	Male	MCA

6. List the students details in ascending order of course.

Input

```
select*from student order by Course ASC;
```

Output

StudentId	Student	Address1	Gender	Course
1001	Kiara	Sector86	Female	BCA
1002	Aryan Bhatia	Sector7	Male	BCA
1003	Puja	jawahar colony	Female	BCA
1005	Rishi Kapoor	Sec15-A	Male	BCA
1004	Sivansh Singh	NIT-1	Male	MCA

7. List the number of Students in BCA course.

Input

```
select COUNT (*) as BCA_Student from Student WHERE Course ='BCA'
```

Output

BCA_Student
4

8. List the number of students available in student table.

Input

```
select COUNT (*) as  TOTAL_Student from Student
```

Output

TOTAL_Student
5

9. Create a table with a primary key constraint.

Input

Run SQL

```
CREATE TABLE EMPLOYEE (
  EMPLY_ID INT(2) PRIMARY KEY,
  EMPLY_NAME
  );
```

EMPLOYEE

EMPLY_ID	EMPLY_NAME
empty	

10. Create a table with all column having not null constraints.

Input

```
CREATE TABLE BRAND(
BRAND_CODE INT (4) NOT NULL,
  BRAND_NAME VARCHAR (10) NOT NULL
  );
```

BRAND

BRAND_CODE	BRAND_NAME
empty	

11. Create a foreign key constraint in a table.

Input

Run SQL

```
CREATE TABLE genZyle(
BRAND_CODE INT (4) PRIMARY KEY,
  BRAND_NAME VARCHAR (10),
  FOREIGN KEY (BRAND_NAME) REFERENCES Student (BRAND_NAME)

  );
```

Available Tables

GenZyle

BRAND_CODE	BRAND_NAME
empty	

12. Create a Table with a unique key constraint.

Input

Run SQL

```
CREATE TABLE STYLO(
BRAND_CODE INT (4) PRIMARY KEY,
  BRAND_NAME VARCHAR (10),
  CUSTOMER_EMAIL VARCHAR (30) UNIQUE
);
```

STYLO

BRAND_CODE	BRAND_NAME	CUSTOMER_EMAIL
empty		

13. Display list of student ordered by course.

Input

Run SQL

```
select * from student order by course;
```

Output

StudentId	Student	Address1	Gender	Course
1001	Kiara	Sector86	Female	BCA
1002	Aryan Bhatia	Sector7	Male	BCA
1003	Puja	jawahar colony	Female	BCA
1005	Rishi Kapoor	Sec15-A	Male	BCA
1004	Sivansh Singh	NIT-1	Male	MCA

14. Display alphabetically sorted list of students.

Input

```
select * from student order by student ASC;
```

Output

StudentId	Student	Address1	Gender	Course
1002	Aryan Bhatia	Sector7	Male	BCA
1001	Kiara	Sector86	Female	BCA
1003	Puja	jawahar colony	Female	BCA
1005	Rishi Kapoor	Sec15-A	Male	BCA
1004	Sivansh Singh	NIT-1	Male	MCA

15. List the names of the employees whose employee numbers are 7369, 7777, 2233.

Input

```
SELECT EMPLY_name
FROM employee
WHERE emply_NUMBER IN (7369, 7777, 2233);
```

Output

EMPLY_NAME
KAJAL
VIRAT
ROHIT

16.List the employees whose names start with "S" (not "s").

Input

```
SELECT EMpLY_name
FROM employee
WHERE EMPLY_name LIKE 'S%';
```

Output

EMPLY_NAME
SHILPA

17. List the employees ending with name "s".

Input

```sql
SELECT empLY_name
FROM employee
WHERE empLY_name LIKE '%s';
```

Output

SQL query successfully executed. However, the result set is empty.

18. List the employee names having "k" as the second character.

```sql
SELECT empLY_name
FROM employee
WHERE empLY_name LIKE '_k%';
```

Output

SQL query successfully executed. However, the result set is empty.

Experiment 2

Q1: Create the following tables.

Table CUSTOMER

column name	Characteristic
SID	Primary Key
Last_Name	
First_Name	

Table ORDERS

column name	Characteristic
Order_ID	Primary Key
Order_Date	
Customer_SID	Foreign Key
Amount	Check > 20000

CUSTOMER TABLE

```
CREATE TABLE CUSTOMER (
    SID INT PRIMARY KEY,
    Last_Name VARCHAR(50),
```

```
    First_Name VARCHAR(50)
);
```

TABLE ORDER

```
CREATE TABLE ORDERSS (
    Order_ID INT PRIMARY KEY,
    Order_Date DATE,
    Customer_SID INT,
    Amount DECIMAL(10, 2),
    FOREIGN KEY (Customer_SID) REFERENCES CUSTOMER(SID),
    CHECK (Amount > 20000)
);
```

1. Insert five records for each table.

Coustmer Table

```sql
INSERT INTO CUSTOMER (SID, Last_Name, First_Name) VALUES
(1, 'Smith', 'John'),
(2, 'Jones', 'Sarah'),
(3, 'Taylor', 'Michael'),
(4, 'Williams', 'Emily'),
(5, 'Brown', 'James');
```

Customer

CUSTOMER

SID	Last_Name	First_Name
1	Smith	John
2	Jones	Sarah
3	Taylor	Michael
4	Williams	Emily
5	Brown	James

Orderss Table

Input

Run SQL

```sql
INSERT INTO ORDERSS (Order_ID, Order_Date, Customer_SID, Amount) VALUES
(101, "2024-01-15", 1, 25000.00),
(102, "2024-02-20", 2, 21000.00),
(103, "2024-03-12", 3, 22000.00),
(104, "2024-04-05", 4, 32000.00),
(105, "2024-05-10", 5, 27000.00);
```

Orderss

ORDERSS

Order_ID	Order_Date	Customer_SID	Amount
101	2024-01-15	1	25000
102	2024-02-20	2	21000
103	2024-03-12	3	22000
104	2024-04-05	4	32000
105	2024-05-10	5	27000

3. List the details of the customers along with the amount.

Input

Run SQL

```sql
SELECT C.SID, C.Last_Name, C.First_Name, O.Amount
FROM CUSTOMER C
JOIN ORDERSS O ON C.SID = O.Customer_SID;
```

Output

SID	Last_Name	First_Name	Amount
1	Smith	John	25000
2	Jones	Sarah	21000
3	Taylor	Michael	22000
4	Williams	Emily	32000
5	Brown	James	27000

4. List the customers whose names end with "s".

Input

Run SQL

```sql
SELECT *
FROM CUSTOMER
WHERE Last_Name LIKE '%s';
```

Output

SID	Last_Name	First_Name
2	Jones	Sarah
4	Williams	Emily

5. List the orders where the amount is between 21000 and 30000.

Input

Run SQL

```sql
SELECT *
FROM ORDERSS
WHERE Amount BETWEEN 21000 AND 30000;
```

Output

Order_ID	Order_Date	Customer_SID	Amount
101	2024-01-15	1	25000
102	2024-02-20	2	21000
103	2024-03-12	3	22000
105	2024-05-10	5	27000

6. List the orders where the amount is increased by 500 and replace with the name "new amount".

Input

Run SQL

```sql
SELECT Order_ID, Order_Date, Customer_SID, Amount + 500 AS "new amount"
FROM ORDERSS;
```

Output

Order_ID	Order_Date	Customer_SID	new amount
101	2024-01-15	1	25500
102	2024-02-20	2	21500
103	2024-03-12	3	22500
104	2024-04-05	4	32500
105	2024-05-10	5	27500

7. Display the Order_ID and total amount of orders.

Input

Run SQL

```sql
SELECT Order_ID, Amount
FROM ORDERS;
```

Output

order_id	amount
1	400
2	300
3	12000
4	400
5	250

8. Calculate the total amount of orders that have more than 15000.

Input

Run SQL

```sql
SELECT SUM(Amount) AS Total_Amount
FROM ORDERSS
WHERE Amount > 15000;
```

Output

Total_Amount
127000

9. Display all the contents of s4 and s5 using UNION clause.

```sql
SELECT * FROM s4
UNION
SELECT * FROM s5;
```

10. Find out the intersection of s4 and s5 tables.

```sql
-- Assuming s4 and s5 are table names
SELECT * FROM s4
INTERSECT
SELECT * FROM s5;
```

11. Display the names of s4 and s5 tables using left, right, inner, and full join.

Input

Run SQL

```sql
-- LEFT JOIN
SELECT * FROM s4
LEFT JOIN s5 ON s4.id = s5.id;

-- RIGHT JOIN
SELECT * FROM s4
RIGHT JOIN s5 ON s4.id = s5.id;

-- INNER JOIN
SELECT * FROM s4
INNER JOIN s5 ON s4.id = s5.id;

-- FULL OUTER JOIN
SELECT * FROM s4
FULL OUTER JOIN s5 ON s4.id = s5.id;
```

12. Display the first name of employees and their managers using self-join.

Input

```sql
SELECT e.First_Name AS Employee, m.First_Name AS Manager
FROM employees e
LEFT JOIN employees m ON e.manager_id = m.id;
```