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 some common 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 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
studentname	Varchar2	40	Null
Address1	Varchar2	300	
Address1	Varchar2	15	
Course	Varchar2	8	

Course:

Dept No	Number	2	constraint
Dname	Varchar	20	Primary key
Location	Varchar	10	

1. Insert five records for each table.

```
CREATE TABLE Student (StudentId NUMBER(4) PRIMARY KEY,StudentName VARCHAR2(40)
NULL,Address1 VARCHAR2(300),Gender VARCHAR2(15),Course VARCHAR2(8));
INSERT INTO Student (StudentId, StudentName, Address1, Gender, Course)
VALUES (1001, 'Shivam', '123 Elm St, Springfield, IL', 'Male', 'CS101');
INSERT INTO Student (StudentId, StudentName, Address1, Gender, Course)
VALUES (1002, 'Ansh', '456 Oak St, Springfield, IL', 'male', 'ENG102');

INSERT INTO Student (StudentId, StudentName, Address1, Gender, Course)
VALUES (1003, 'Harsh', '789 Pine St, Springfield, IL', 'Male', 'MATH203');

INSERT INTO Student (StudentId, StudentName, Address1, Gender, Course)
VALUES (1004, 'Gurbir', '321 Maple St, Springfield, IL', 'male', 'BIO301');

INSERT INTO Student (StudentId, StudentName, Address1, Gender, Course)
VALUES (1005, 'Piyush', '654 Birch St, Springfield, IL', 'Male', 'CHEM104');
```

Student

StudentId	StudentName	Address1	Gender	Course
1001	Shivam	123 Elm St, Springfield, IL	Male	CS101
1002	Ansh	456 Oak St, Springfield, IL	male	ENG10
1003	Harsh	789 Pine St, Springfield, IL	Male	MATH2
1004	Gurbir	321 Maple St, Springfield, IL	male	BIO301
1005	Piyush	654 Birch St, Springfield, IL	Male	CHEM1

```
CREATE TABLE Course (DeptNo NUMBER(2), Dname VARCHAR2(20) PRIMARY KEY, Location
VARCHAR2(10),CONSTRAINT chk_deptno CHECK (DeptNo > 0));
INSERT INTO Course (DeptNo, Dname, Location)
VALUES (10, 'Computer Science', 'Building A');
INSERT INTO Course (DeptNo, Dname, Location)
VALUES (20, 'English', 'Building B');
INSERT INTO Course (DeptNo, Dname, Location)
VALUES (30, 'Mathematics', 'Building C');
INSERT INTO Course (DeptNo, Dname, Location)
VALUES (40, 'Biology', 'Building D');

INSERT INTO Course (DeptNo, Dname, Location)
VALUES (50, 'Chemistry', 'Building E');
```

Course

DeptNo	Dname	Location
10	Computer Science	Building A
20	English	Building B
30	Mathematics	Building C
40	Biology	Building D
50	Chemistry	Building E

2.List all information about all students from student table

Input

```
SELECT * FROM Student;
```

Output

StudentId	StudentName	Address1	Gender	Course
1001	Shivam	123 Elm St, Springfield, IL	Male	CS101
1002	Ansh	456 Oak St, Springfield, IL	male	ENG102
1003	Harsh	789 Pine St, Springfield, IL	Male	MATH203
1004	Gurbir	321 Maple St, Springfield, IL	male	BIO301
1005	Piyush	654 Birch St, Springfield, IL	Male	CHEM104

3.List all student numbers along with their Courses

Input

```
SELECT StudentId, Course
FROM Student;
```

Course

DeptNo	Dname	Location
10	Computer Science	Building A
20	English	Building B
30	Mathematics	Building C
40	Biology	Building D
50	Chemistry	Building E

4.List Course names and locations from the Course table

Input

```
SELECT Dname, Location
FROM Course;
```

Output

Dname	Location
Computer Science	Building A
English	Building B
Mathematics	Building C
Biology	Building D
Chemistry	Building E

5. List the details of the Students in CS101 Course

Input

```
SELECT *
FROM Student
WHERE Course = 'CS101';
```

Output

StudentId	StudentName	Address1	Gender	Course
1001	Shivam	123 Elm St, Springfield, IL	Male	CS101

6. List the students details in ascending order of course

Input

```
SELECT *
FROM Student
ORDER BY Course ASC;
```

Output

StudentId	StudentName	Address1	Gender	Course
1004	Gurbir	321 Maple St, Springfield, IL	male	BIO301
1005	Piyush	654 Birch St, Springfield, IL	Male	CHEM104
1001	Shivam	123 Elm St, Springfield, IL	Male	CS101
1002	Ansh	456 Oak St, Springfield, IL	male	ENG102
1003	Harsh	789 Pine St, Springfield, IL	Male	MATH203

7.List the number of Students in CS101 course.

Input

```
SELECT COUNT(*) AS num_students_in_CS101
FROM Student
WHERE Course = 'CS101';
```

Output

num_students_in_CS101
1

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

Input

```
SELECT COUNT(*) AS total_students
FROM Student;
```

Output

total_students
5

9. Create a table with a primary key constraint.

Input

```
CREATE TABLE Course (
    DeptNo NUMBER(2) PRIMARY KEY,
    Dname VARCHAR2(20),
    Location VARCHAR2(10)
);
```

Course

DeptNo	Dname	Location
empty		

Experiment 2

Q1: Create the following tables

Customer

SID	**Primary key**
Last_Name	
First_Name	

Orders

Order_ID	**Primary key**
Order_Date	
Customer_sid	**Foreign key**
Amount	**Check > 20000**

1. Insert five records for each tabl

Input

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

INSERT INTO CUSTOMER (SID, Last_Name, First_Name) VALUES
(1, 'Smith', 'John'),
(2, 'Jones', 'Alex'),
(3, 'Roberts', 'Sarah'),
(4, 'Evans', 'James'),
(5, 'Stevens', 'Emma');
```

CUSTOMER

SID	Last_Name	First_Name
1	Smith	John
2	Jones	Alex
3	Roberts	Sarah
4	Evans	James
5	Stevens	Emma

Input

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

Customer

SID	Last_Name	First_Name
1	Shivam	Yadav
2	Ansh	Sharma
3	Harsh	Mondal
4	Gurbir	Singh
5	Piyush	Sharma

Input

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

INSERT INTO ORDERS (Order_ID, Order_Date, Customer_SID, Amount) VALUES
(101, '2023-01-10', 1, 25000),
(102, '2023-02-15', 2, 30000),
(103, '2023-03-20', 3, 27000),
(104, '2023-04-25', 4, 32000),
(105, '2023-05-30', 5, 29000);
```

ORDERS

Order_ID	Order_Date	Customer_SID	Amount
101	2023-01-10	1	25000
102	2023-02-15	2	30000
103	2023-03-20	3	27000
104	2023-04-25	4	32000
105	2023-05-30	5	29000

2. List Customer Details Along with the Order Amount

```
SELECT CUSTOMER.SID, CUSTOMER.Last_Name, CUSTOMER.First_Name, ORDERS.Amount
FROM CUSTOMER
JOIN ORDERS ON CUSTOMER.SID = ORDERS.Customer_SID;
```

SID	Last_Name	First_Name	Amount
1	Smith	John	25000
2	Jones	Alex	30000
3	Roberts	Sarah	27000
4	Evans	James	32000
5	Stevens	Emma	29000

3.List Customers Whose Names End with "s"

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

SID	Last_Name	First_Name
2	Jones	Alex
3	Roberts	Sarah
4	Evans	James
5	Stevens	Emma

4.List Orders Where Amount is Between 21000 and 30000

```
SELECT * FROM ORDERS
WHERE Amount BETWEEN 21000 AND 30000;
```

Order_ID	Order_Date	Customer_SID	Amount
101	2023-01-10	1	25000
102	2023-02-15	2	30000
103	2023-03-20	3	27000
105	2023-05-30	5	29000

5.List the orders where amount is increased by 500 and replace with name “new amount”.

```
SELECT Order_ID, Amount + 500 AS "New Amount"
FROM ORDERS;
```

Order_ID	New Amount
101	25500
102	30500
103	27500
104	32500
105	29500

6.Display the order_id and total amount of orders

```
SELECT Order_ID, Amount AS Total_Amount
FROM ORDERS;
```

Order_ID	Total_Amount
101	25000
102	30000
103	27000
104	32000
105	29000

7.Calculate the total amount of orders that has more than 15000

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

Total_Amount
143000

8. Display all the contents of s4 and s5 using union clause

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

9. Find out the intersection of s4 and s5 tables

```
SELECT * FROM s4
INTERSECT
SELECT * FROM s5;
```

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

```sql
SELECT s4.*, s5.*
FROM s4
LEFT JOIN s5 ON s4.ID = s5.ID;
```

```sql
SELECT s4.*, s5.*
FROM s4
INNER JOIN s5 ON s4.ID = s5.ID;
```

```sql
SELECT s4.*, s5.*
FROM s4
FULL OUTER JOIN s5 ON s4.ID = s5.ID;
```

11. Find out the names of s4 which are distinct

```sql
SELECT DISTINCT Name FROM s4;
```

12. Write a query to Grant access and modification rights to customer table to user

```sql
GRANT SELECT, INSERT, UPDATE, DELETE ON CUSTOMER TO user;
```

13. Write a query to revoke access rights to customer table to user

```sql
REVOKE SELECT, INSERT, UPDATE, DELETE ON CUSTOMER FROM user;
```

14. Write a query to take backup of a database

```sql
BACKUP DATABASE dbname TO DISK = 'path_to_backup_file';
```

15. Write a query to restore a database

```sql
RESTORE DATABASE dbname FROM DISK = 'path_to_backup_file';
```