# 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

	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	
urse:	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

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

#### Course table

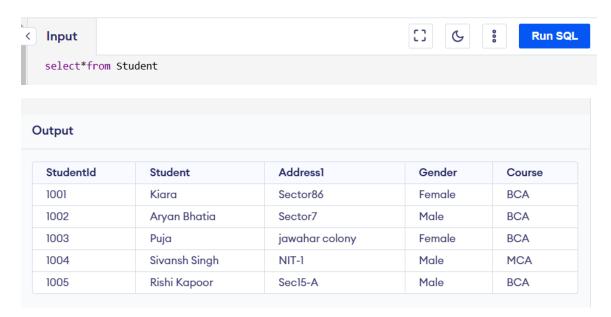
```
insert into Courses values ("1001A", 1, "SCA", "C block");
insert into Courses values ("1002A", 2, "SCA", "C block");
insert into Courses values ("1003A", 3, "SCA", "C block");
insert into Courses values ("1004A", 4, "SCA", "C block");
insert into Courses values ("1005A", 5, "SCA", "C block");
```

#### Courses

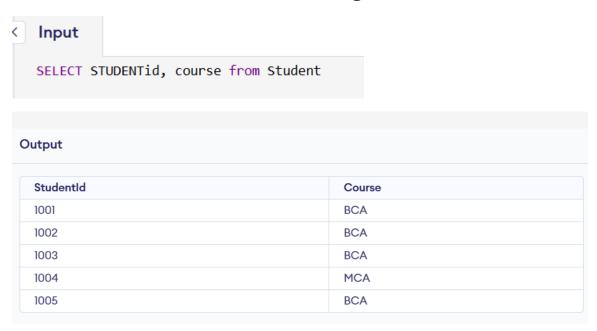
put			
CourseID	Dept_No	D_name	Location
001A	1	SCA	C block
002A	2	SCA	C block
003A	3	SCA	C block
004A	4	SCA	c block
005A	5	SCA	C block

2. List all information about all students from student

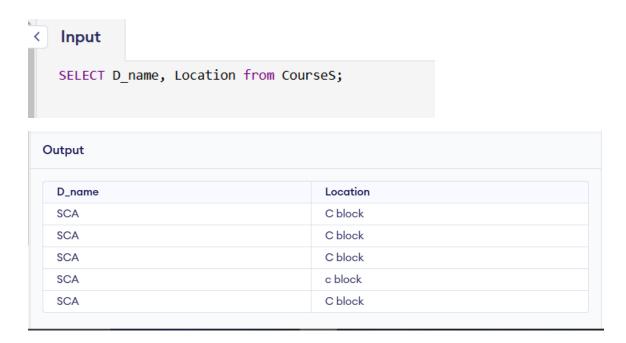
### table.



3. List all student numbers along with their Courses.



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



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



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

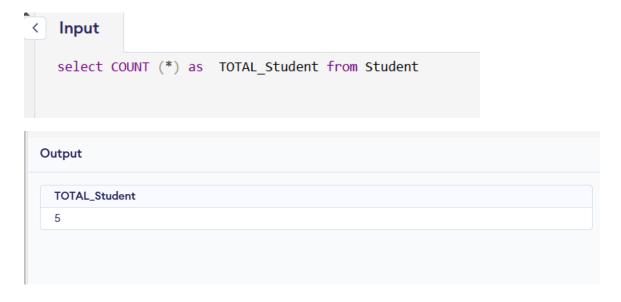
```
select*from student order by Course ASC;
```

StudentId	Student	Address1	Gender	0
Studentid	Student	Addressi	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

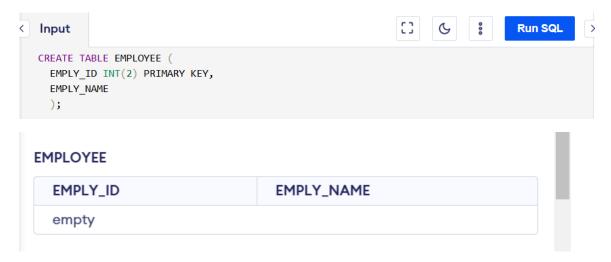
7. List the number of Students in BCA course.



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



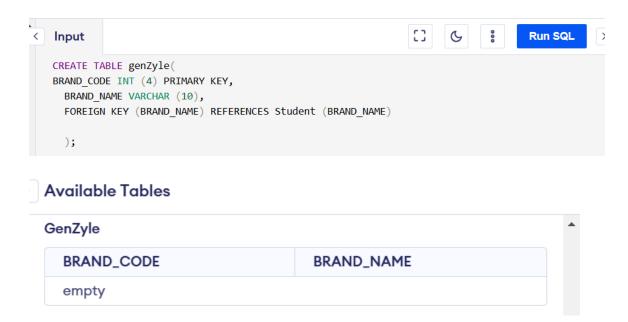
9. Create a table with a primary key constraint.



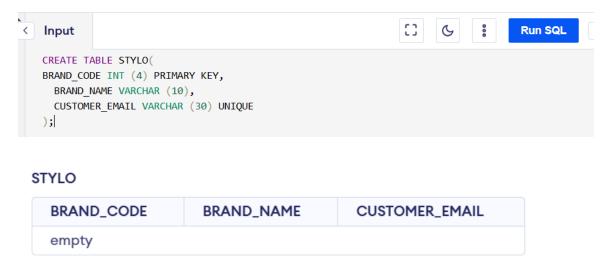
10. Create a table with all column having not null constraints.



11. Create a foreign key constraint in a table.



### 12. Create a Table with a unique key constraint.

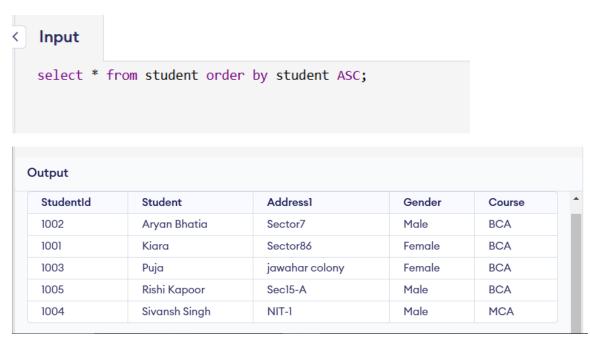


### 13. Display list of student ordered by course.



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.



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

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



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



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

```
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.

```
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	Prim	агу Кеу		
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**

```
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			

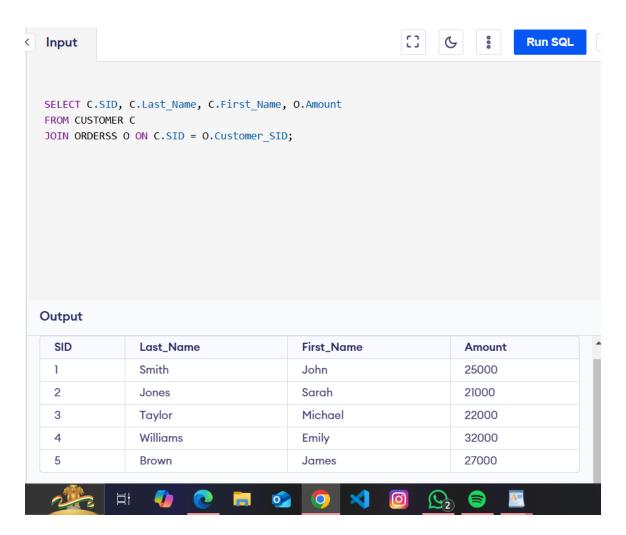
### Orderss Table

```
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);
```

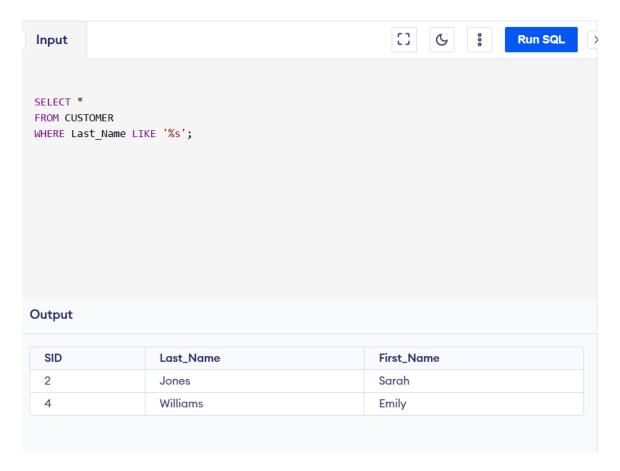
### **Orderss**

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

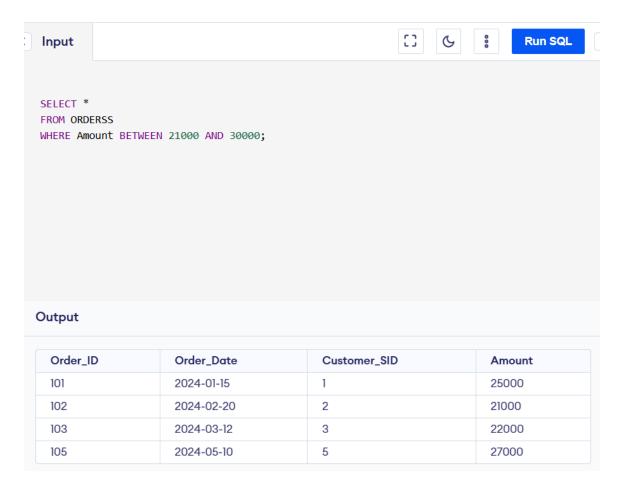
3. List the details of the customers along with the amount.



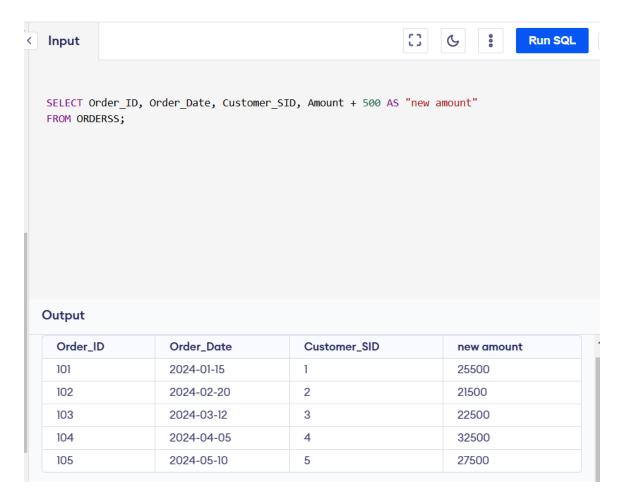
4. List the customers whose names end with "s".



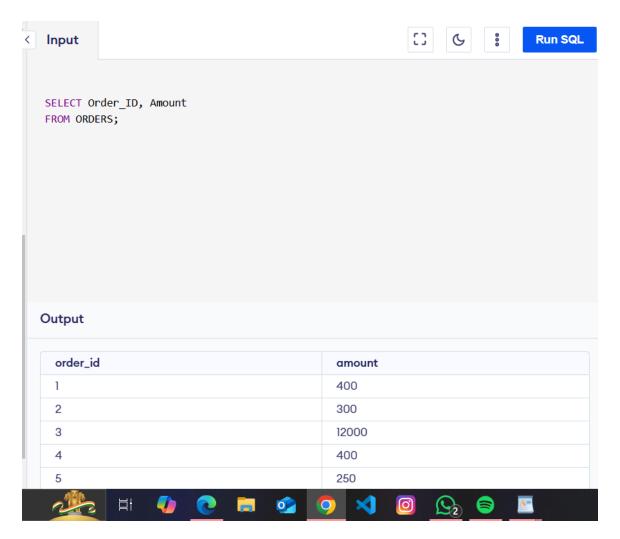
5. List the orders where the amount is between 21000 and 30000.



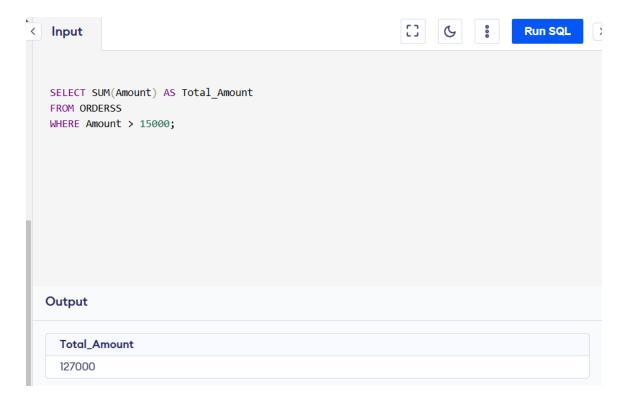
6. List the orders where the amount is increased by 500 and replace with the name "new amount".



7. Display the Order\_ID and total amount of orders.



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



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

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

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

```
-- 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.

```
|-- LEFT JOIN
|-- LEFT JOIN |
| SELECT * FROM $4
| LEFT JOIN $5 ON $4.id = $5.id;
| -- RIGHT JOIN |
| SELECT * FROM $4
| RIGHT JOIN $5 ON $4.id = $5.id;
| -- INNER JOIN |
| SELECT * FROM $4
| INNER JOIN $5 ON $4.id = $5.id;
| -- FULL OUTER JOIN |
| SELECT * FROM $4
| FULL OUTER JOIN $5 ON $4.id = $5.id;
```

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

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