

# **ORACLE LAB**

## **BCS-DS-552**

**Manav Rachna International Institute of Research and  
Studies  
School of Computer Application**

<b>Submitted By</b>	
<b>Student Name</b>	<b>Rakshit sharma</b>
<b>Roll No</b>	<b>22/FCA/BCA(DS&amp;BDA)019</b>
<b>Programme</b>	<b>Bachelor of Computer Applications</b>
<b>Semester</b>	<b>5<sup>th</sup> Semester</b>
<b>Section/Group</b>	<b>E</b>
<b>Department</b>	<b>School of Computer Applications</b>
<b>Batch</b>	<b>2022-25</b>
<b>Submitted To</b>	
<b>Faculty Name</b>	<b>Miss Iram fatima</b>

<u>S.No</u>	<u>Index</u>	<u>Signature</u>
1	Create the following tables	
2	Insert five records for each table	
3	Customer SID column in the ORDERS table is a foreign key pointing to the SID column in the CUSTOMER table.	
4	List the details of the customers along with the amount.	
5	List the customers whose names end with "s"	
6	List the orders where amount is between 21000 and 30000	
7	List the orders where amount is increased by 500 and replace with name "new amount".	
8	Display the order_id and total amount of orders	
9	Calculate the total amount of orders that has more than 15000.	
10	Create the following tables--( Student , Student 1)	
11	Display all the contents of student and student1 using union clause.	
12	Find out the intersection of student and student1 tables.	
13	Write a PL/SQL code to find the greatest of three numbers	
14	Write a PL/SQL code to print the numbers from 1 to n.	
15	Write a PL/SQL code to reverse a string using for loop	
16	Write a PL/SQL code to find the sum of n numbers.	
17		

# Experiment-1

Create the following tables

## **Customer**

<b><u>Column_name</u></b>	<b><u>Data type</u></b>	<b><u>Size</u></b>	<b><u>Constraint</u></b>
SID	Varchar2	4	Primary Key
First_Name	Char	20	
Last_name	Char	20	

## **Orders**

<b><u>Column_name</u></b>	<b><u>Data type</u></b>	<b><u>Size</u></b>	<b><u>Constraint</u></b>
Order_ID	Varchar2	4	Primary Key
Order_date	Char	20	
Customer_SID	Varchar2	20	Foreign Key
Amount	Number		Check > 20000

Here's the Command of SQL:

```
CREATE TABLE Customer( SID VARCHAR2(4) PRIMARY KEY,  
First_name VARCHAR2(20) NOT NULL,  
Last_name VARCHAR2(20)  
);  
CREATE TABLE Orders ( Order_ID VARCHAR2(4) PRIMARY KEY,  
Order_date VARCHAR2(20) NOT NULL,  
Customer_SID VARCHAR2(4),  
Amount NUMBER CHECK (Amount >= 20000),  
FOREIGN KEY (Customer_SID) REFERENCES Customer(SID)  
);
```

## SQL Worksheet

```
1 ✓ CREATE TABLE Customer(  
2     SID VARCHAR2(4) PRIMARY KEY,  
3     First_name VARCHAR2(20) NOT NULL,  
4     Last_name VARCHAR2(20)  
5 );  
6  
7 ✓ CREATE TABLE Orders (  
8     Order_ID VARCHAR2(4) PRIMARY KEY,  
9     Order_date VARCHAR2(20) NOT NULL,  
10    Customer_SID VARCHAR2(4),  
11    Amount NUMBER CHECK (Amount >= 20000),  
12    FOREIGN KEY (Customer_SID) REFERENCES Customer(SID)  
13 );  
14
```

Table created.

Table created.

This should create the tables with the necessary constraints and proper data types.

## Experiment-2

Insert five records for each table

SQL commands ;

-- Inserting records into the Customer table

```
INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C001', 'John', 'Doe');
```

```
INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C002', 'Jane', 'Smith');
```

```
INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C003', 'Alice', 'Johnson');
```

```
INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C004', 'Bob', 'Brown');
```

```
INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C005', 'Charlie', 'Davis');
```

-- Inserting records into the Orders table

```
INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount) VALUES ('O001', '2023-01-01', 'C001', 25000.00);
```

```
INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount) VALUES ('O002', '2023-01-02', 'C002', 30000.00);
```

```
INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount) VALUES ('O003', '2023-01-03', 'C003', 22000.00);
```

```
INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount) VALUES ('O004', '2023-01-04', 'C004', 21000.00);
```

```
INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount) VALUES ('O005', '2023-01-05', 'C005', 28000.00);
```

SQL Worksheet

Clear

Find

Actions

Save

Run

```
1 -- Inserting records into the Customer table
2 INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C001', 'John', 'Doe');
3 INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C002', 'Jane', 'Smith');
4 INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C003', 'Alice', 'Johnson');
5 INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C004', 'Bob', 'Brown');
6 INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C005', 'Charlie', 'Davis');
7
8 -- Inserting records into the Orders table
9 INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount) VALUES ('0001', '2023-01-01', 'C001', 25000.00);
10 INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount) VALUES ('0002', '2023-01-02', 'C002', 30000.00);
11 INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount) VALUES ('0003', '2023-01-03', 'C003', 22000.00);
12 INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount) VALUES ('0004', '2023-01-04', 'C004', 21000.00);
13 INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount) VALUES ('0005', '2023-01-05', 'C005', 28000.00);
14
```

1 row(s) inserted.  
1 row(s) inserted.  
1 row(s) inserted.  
1 row(s) inserted.  
1 row(s) inserted.  
1 row(s) inserted.

## Experiment-3

**Customer SID column in the ORDERS table is a foreign key pointing to the SID column in the CUSTOMER table.**

-- Creating Customer table

```
CREATE TABLE Customer(  
  SID VARCHAR2(4) PRIMARY KEY,  
  First_name VARCHAR2(20) NOT NULL,  
  Last_name VARCHAR2(20)  
);
```

-- Creating Orders table

```
CREATE TABLE Orders (  
  Order_ID VARCHAR2(4) PRIMARY KEY,  
  Order_date VARCHAR2(20) NOT NULL,  
  Customer_SID VARCHAR2(4),  
  Amount NUMBER(10, 2) CHECK (Amount >= 20000),  
  FOREIGN KEY (Customer_SID) REFERENCES Customer(SID)  
);
```

-- Inserting records into the Customer table

```
INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C001',  
'John', 'Doe');  
INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C002',  
'Jane', 'Smith');  
INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C003',  
'Alice', 'Johnson');  
INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C004',  
'Bob', 'Brown');  
INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C005',  
'Charlie', 'Davis');
```

-- Inserting records into the Orders table

```
INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount)  
VALUES ('O001', '2023-01-01', 'C001', 25000.00);  
INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount)  
VALUES ('O002', '2023-01-02', 'C002', 30000.00);  
INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount)  
VALUES ('O003', '2023-01-03', 'C003', 22000.00);  
INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount)  
VALUES ('O004', '2023-01-04', 'C004', 21000.00);
```

INSERT INTO Orders (Order\_ID, Order\_date, Customer\_SID, Amount)  
VALUES ('O005', '2023-01-05', 'C005', 28000.00);

```
SQL Worksheet

1  -- Creating Customer table
2  v CREATE TABLE Customer(
3      SID VARCHAR2(4) PRIMARY KEY,
4      First_name VARCHAR2(20) NOT NULL,
5      Last_name VARCHAR2(20)
6  );
7
8  -- Creating Orders table
9  v CREATE TABLE Orders (
10     Order_ID VARCHAR2(4) PRIMARY KEY,
11     Order_date VARCHAR2(20) NOT NULL,
12     Customer_SID VARCHAR2(4),
13     Amount NUMBER(10, 2) CHECK (Amount >= 20000),
14     FOREIGN KEY (Customer_SID) REFERENCES Customer(SID)
15 );
16
17 -- Inserting records into the Customer table
18 INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C001', 'John', 'Doe');
19 INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C002', 'Jane', 'Smith');
20 INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C003', 'Alice', 'Johnson');
21 INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C004', 'Bob', 'Brown');
22 INSERT INTO Customer (SID, First_name, Last_name) VALUES ('C005', 'Charlie', 'Davis');
23
24 -- Inserting records into the Orders table
25 INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount) VALUES ('O001', '2023-01-01', 'C001', 25000.00);
26 INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount) VALUES ('O002', '2023-01-02', 'C002', 30000.00);
27 INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount) VALUES ('O003', '2023-01-03', 'C003', 22000.00);
28 INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount) VALUES ('O004', '2023-01-04', 'C004', 21000.00);
29 INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount) VALUES ('O005', '2023-01-05', 'C005', 28000.00);
30
```

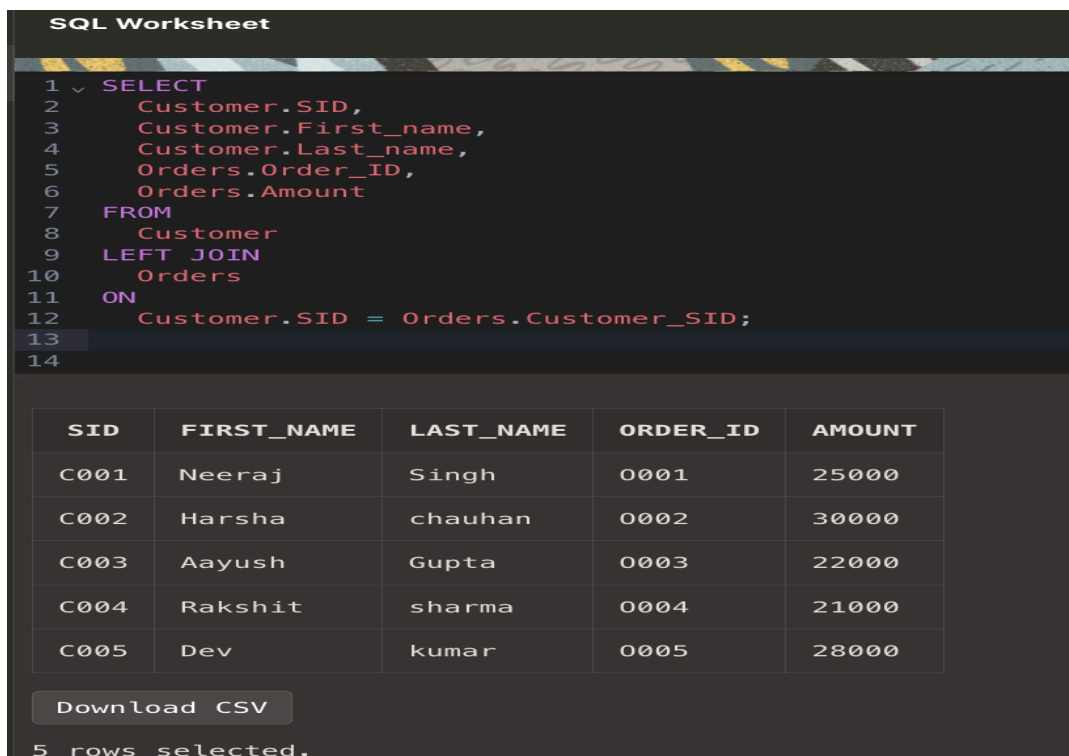


## Experiment-4

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

**--Sql commands :-**

```
SELECT
    Customer.SID,
    Customer.First_name,
    Customer.Last_name,
    Orders.Order_ID,
    Orders.Amount
FROM
    Customer
LEFT JOIN
    Orders
ON
    Customer.SID = Orders.Customer_SID;
```



The screenshot shows an SQL Worksheet interface. The top section contains the following SQL query:

```
1 SELECT
2     Customer.SID,
3     Customer.First_name,
4     Customer.Last_name,
5     Orders.Order_ID,
6     Orders.Amount
7 FROM
8     Customer
9 LEFT JOIN
10    Orders
11 ON
12     Customer.SID = Orders.Customer_SID;
```

Below the query, the results are displayed in a table with 5 rows and 5 columns:

SID	FIRST_NAME	LAST_NAME	ORDER_ID	AMOUNT
C001	Neeraj	Singh	0001	25000
C002	Harsha	chauhan	0002	30000
C003	Aayush	Gupta	0003	22000
C004	Rakshit	sharma	0004	21000
C005	Dev	kumar	0005	28000

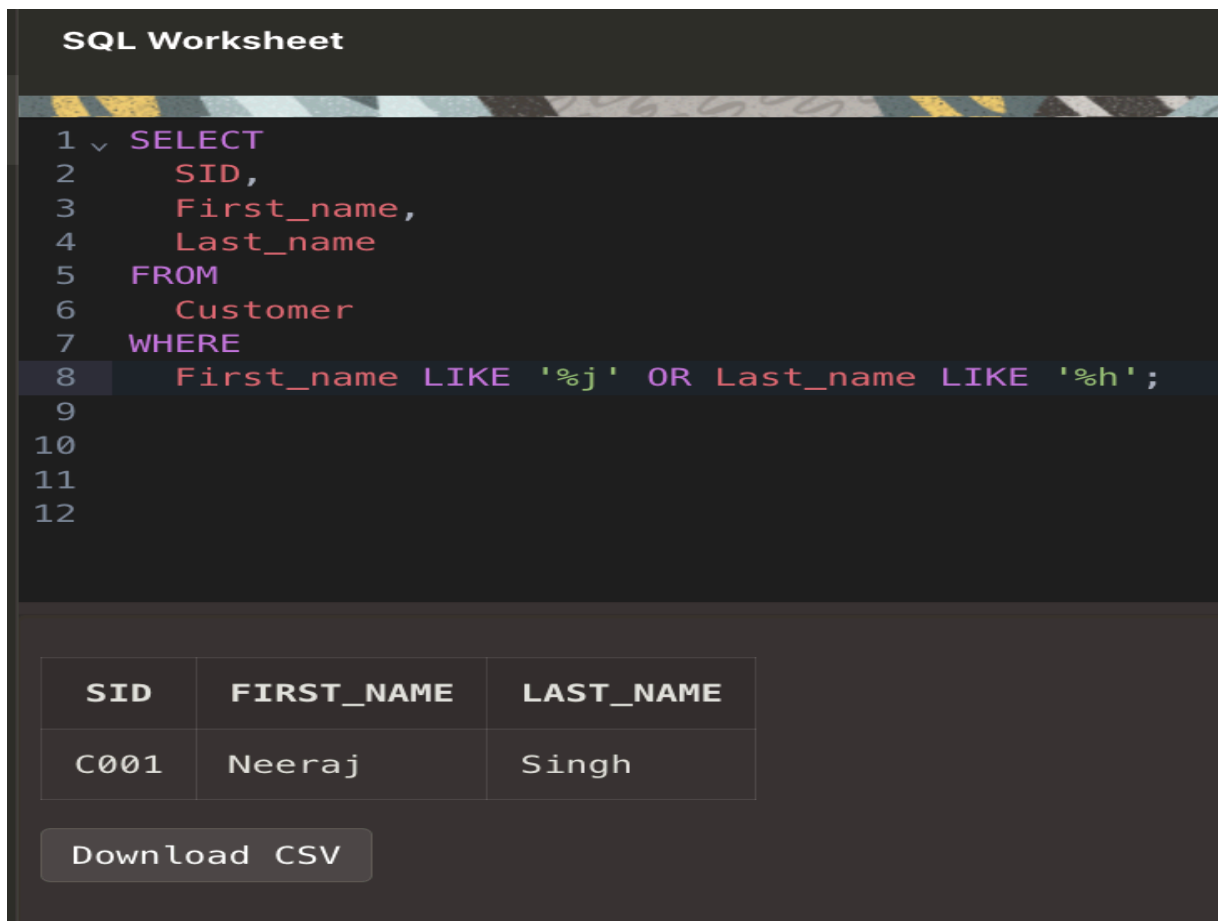
At the bottom of the results section, there is a button labeled "Download CSV" and a status message: "5 rows selected."

## Experiment-5

List the customers whose names end with “s”.

--Sql commands :-

```
SELECT
  SID,
  First_name,
  Last_name
FROM
  Customer
WHERE
  First_name LIKE '%j' OR Last_name LIKE '%h';
```



The screenshot shows an SQL Worksheet interface. At the top, the title "SQL Worksheet" is displayed. Below it, a query is entered in a text area, with line numbers 1 through 12 on the left. The query is: `SELECT SID, First_name, Last_name FROM Customer WHERE First_name LIKE '%j' OR Last_name LIKE '%h';`. Below the query area, a table displays the results of the query. The table has three columns: **SID**, **FIRST\_NAME**, and **LAST\_NAME**. The first row of data shows **C001**, **Neeraj**, and **Singh**. At the bottom of the interface, there is a button labeled "Download CSV".

SID	FIRST_NAME	LAST_NAME
C001	Neeraj	Singh

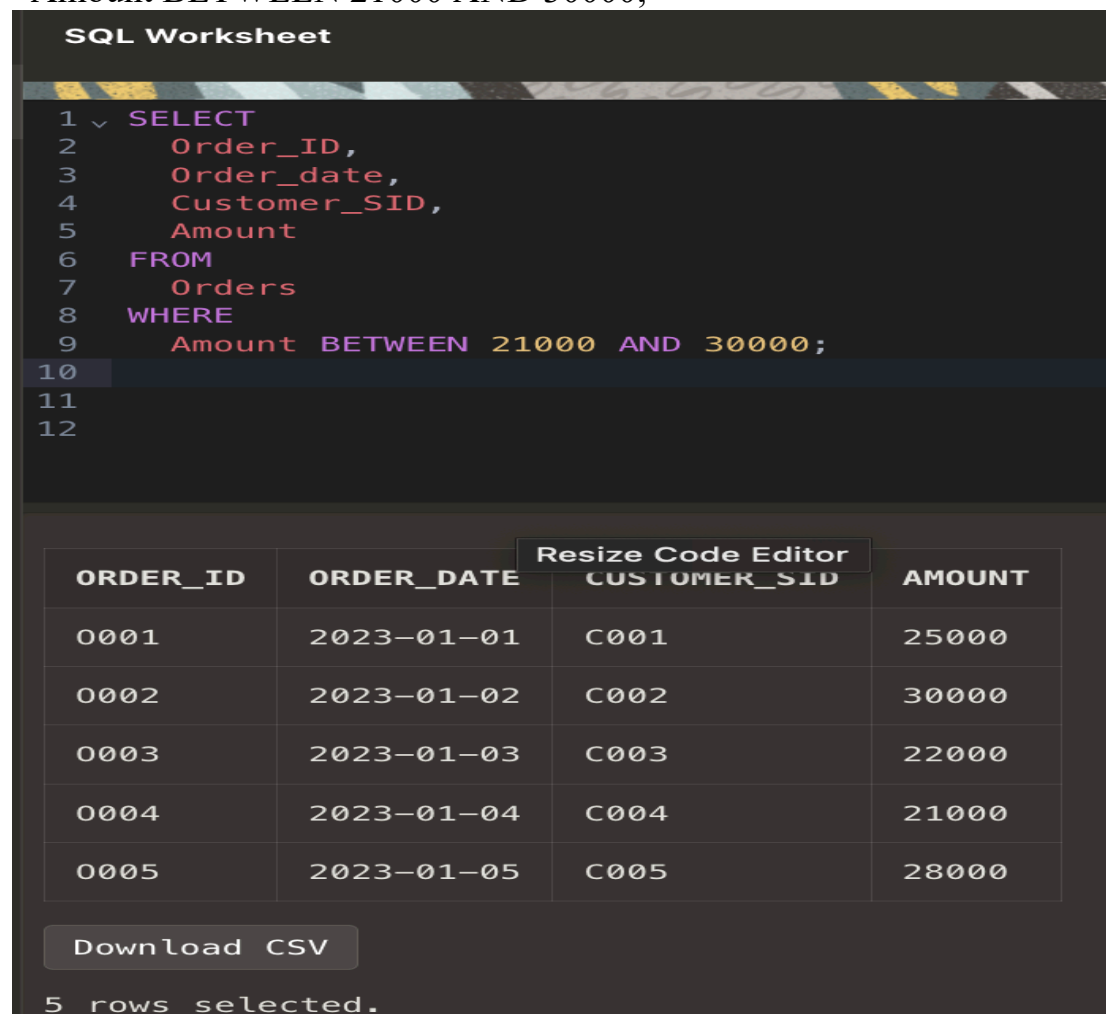
Download CSV

## Experiment-6

List the orders where amount is between 21000 and 30000

--Sql commands :-

```
SELECT
  Order_ID,
  Order_date,
  Customer_SID,
  Amount
FROM
  Orders
WHERE
  Amount BETWEEN 21000 AND 30000;
```



The screenshot shows an SQL Worksheet interface. At the top, the title "SQL Worksheet" is displayed. Below it, a code editor contains the following SQL query:

```
1 SELECT
2   Order_ID,
3   Order_date,
4   Customer_SID,
5   Amount
6 FROM
7   Orders
8 WHERE
9   Amount BETWEEN 21000 AND 30000;
```

Below the code editor, a table displays the results of the query. The table has four columns: ORDER\_ID, ORDER\_DATE, CUSTOMER\_SID, and AMOUNT. There are five rows of data. A tooltip "Resize Code Editor" is visible over the table header. At the bottom of the interface, there is a button labeled "Download CSV" and a status message "5 rows selected."

ORDER_ID	ORDER_DATE	CUSTOMER_SID	AMOUNT
0001	2023-01-01	C001	25000
0002	2023-01-02	C002	30000
0003	2023-01-03	C003	22000
0004	2023-01-04	C004	21000
0005	2023-01-05	C005	28000

Download CSV

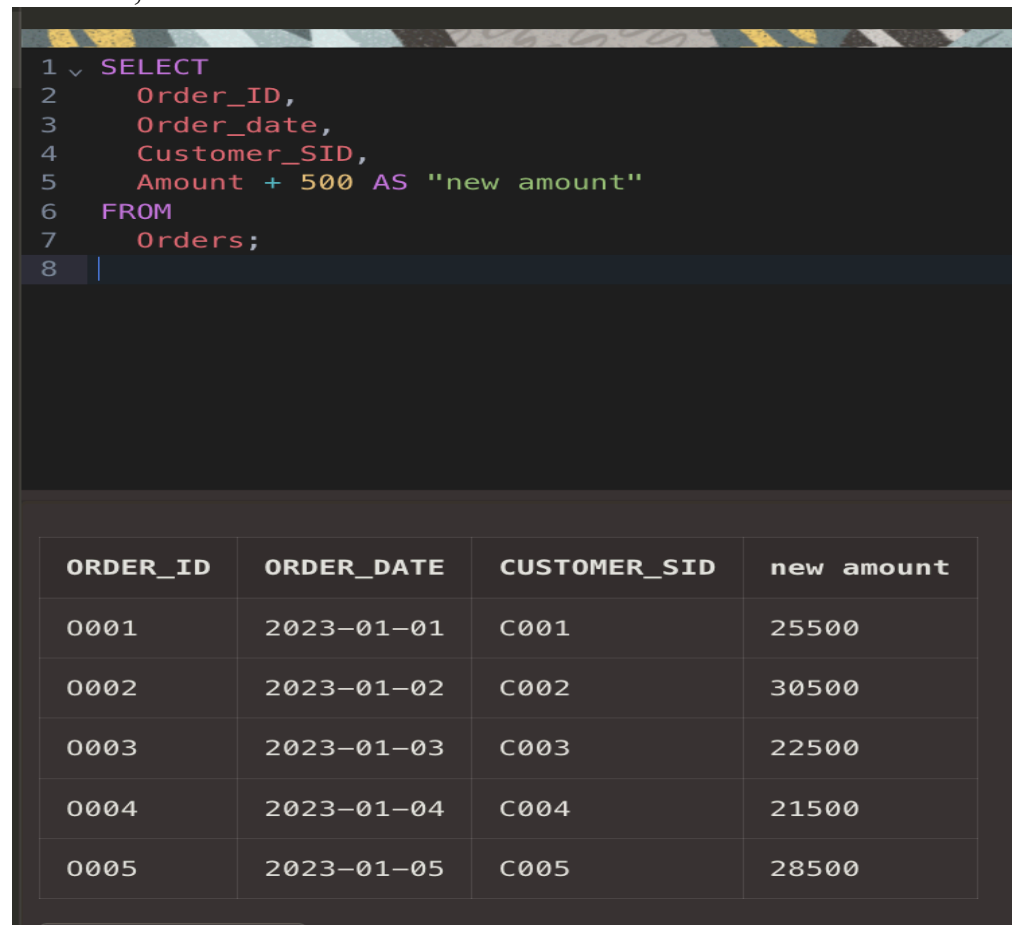
5 rows selected.

## Experiment-7

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

### --Sql commands :-

```
SELECT
  Order_ID,
  Order_date,
  Customer_SID,
  Amount + 500 AS "new amount"
FROM
  Orders;
```



```
1 SELECT
2   Order_ID,
3   Order_date,
4   Customer_SID,
5   Amount + 500 AS "new amount"
6 FROM
7   Orders;
```

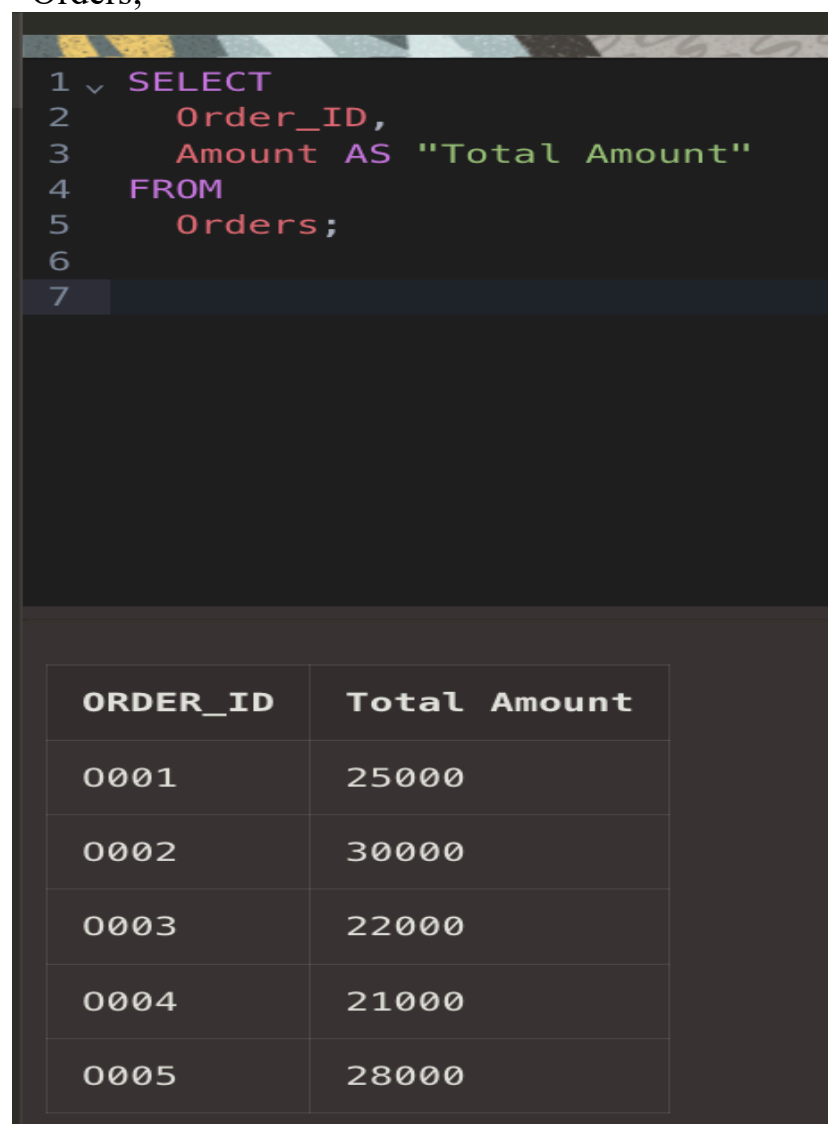
ORDER_ID	ORDER_DATE	CUSTOMER_SID	new amount
0001	2023-01-01	C001	25500
0002	2023-01-02	C002	30500
0003	2023-01-03	C003	22500
0004	2023-01-04	C004	21500
0005	2023-01-05	C005	28500

## Experiment-8

Display the order\_id and total amount of orders

--Sql commands :-

```
SELECT  
  Order_ID,  
  Amount AS "Total Amount"  
FROM  
  Orders;
```



```
1 SELECT  
2   Order_ID,  
3   Amount AS "Total Amount"  
4 FROM  
5   Orders;  
6  
7
```

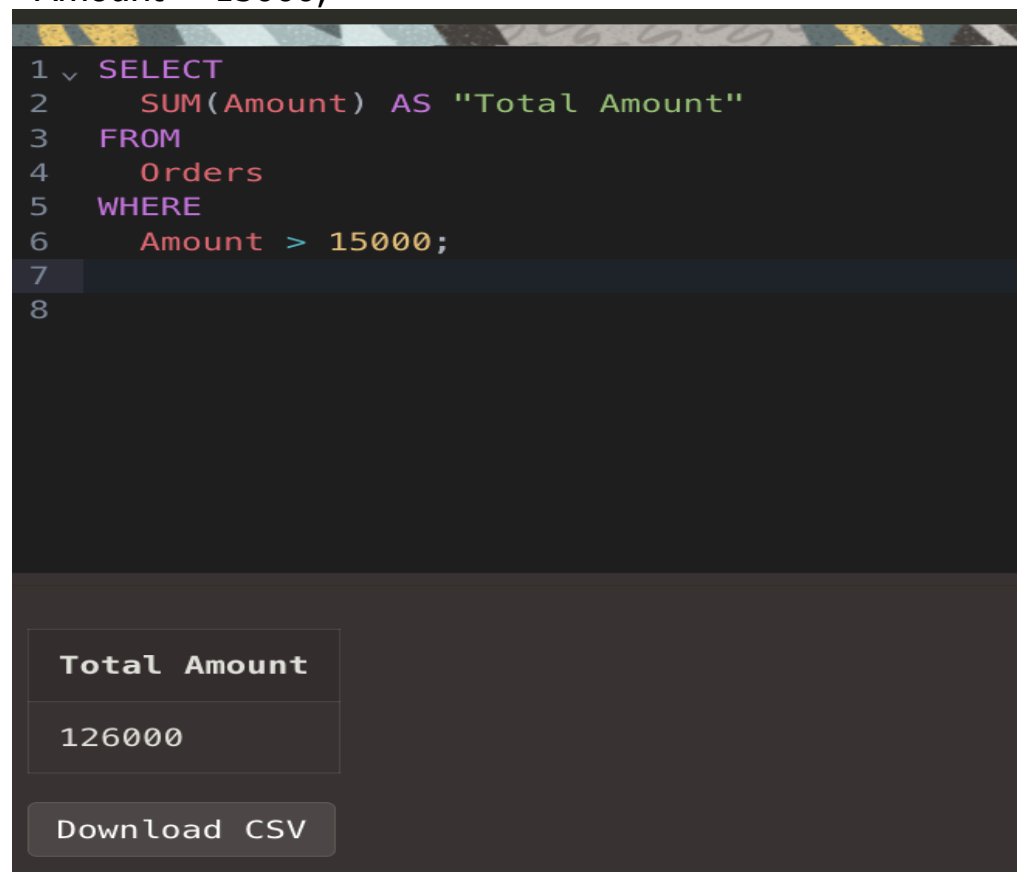
ORDER_ID	Total Amount
0001	25000
0002	30000
0003	22000
0004	21000
0005	28000

## Experiment-9

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

**--Sql commands :-**

```
SELECT
  SUM(Amount) AS "Total Amount"
FROM
  Orders
WHERE
  Amount > 15000;
```



The screenshot shows a SQL query execution interface. The query is displayed in a dark-themed editor with line numbers 1 through 8. The query is: `SELECT SUM(Amount) AS "Total Amount" FROM Orders WHERE Amount > 15000;`. Below the query editor, the result is shown in a table with one row and one column. The column header is "Total Amount" and the value is "126000". There is a "Download CSV" button below the result table.

Total Amount
126000

Download CSV

## Experiment-10

Create the following tables -- ( Student , Student 1 )

Create the following tables

### **Student**

<b><u>Column_name</u></b>	<b><u>Data type</u></b>	<b><u>Size</u></b>	<b><u>Constraint</u></b>
RollNo	Varchar2	20	Primary Key
Name	Char	20	
Class	Varchar2	20	
Marks	Number	6,2	

### **Student1**

<b><u>Column_name</u></b>	<b><u>Data type</u></b>	<b><u>Size</u></b>	<b><u>Constraint</u></b>
R_No	Varchar2	20	Primary Key
Name	Char	20	
Class	Varchar2	20	
Marks	Number	6,2	

### **Sql command :- Student**

```
CREATE TABLE Student (  
    Roll_no INT PRIMARY KEY,  
    Name VARCHAR(100),  
    Class VARCHAR(50),  
    Marks DECIMAL(5, 2)  
);
```

### **Sql command :- Student1**

```
CREATE TABLE Student1 (  
    R_No VARCHAR2 PRIMARY KEY,  
    Name VARCHAR(100),  
    Class VARCHAR(50),  
    Marks DECIMAL(5, 2)  
);
```

# Experiment-11

**Display all the contents of student and student1 using union clause.**

**--Sql commands :-**

```
SELECT Roll_no AS R_No, Name, Class, Marks FROM Student
UNION
SELECT R_No, Name, Class, Marks FROM Student1;
```

**SQL Worksheet**

1	✓	SELECT Roll_no AS R_No, Name, Class, Marks FROM Student
2		UNION
3		SELECT R_No, Name, Class, Marks FROM Student1;
4		
5		
6		

R_NO	NAME	CLASS	MARKS
1	Isha	11A	81.25
1	Neeraj	10A	85.5
2	Harsha	10A	78
2	Sumit	11A	74
3	Aman	11B	89.5
3	Rakshit	10B	92.75
4	Aayush	10B	67.4
4	Tisha	11B	65.75
5	Rahul	11C	93.6
5	Ranjeet	10C	88.9



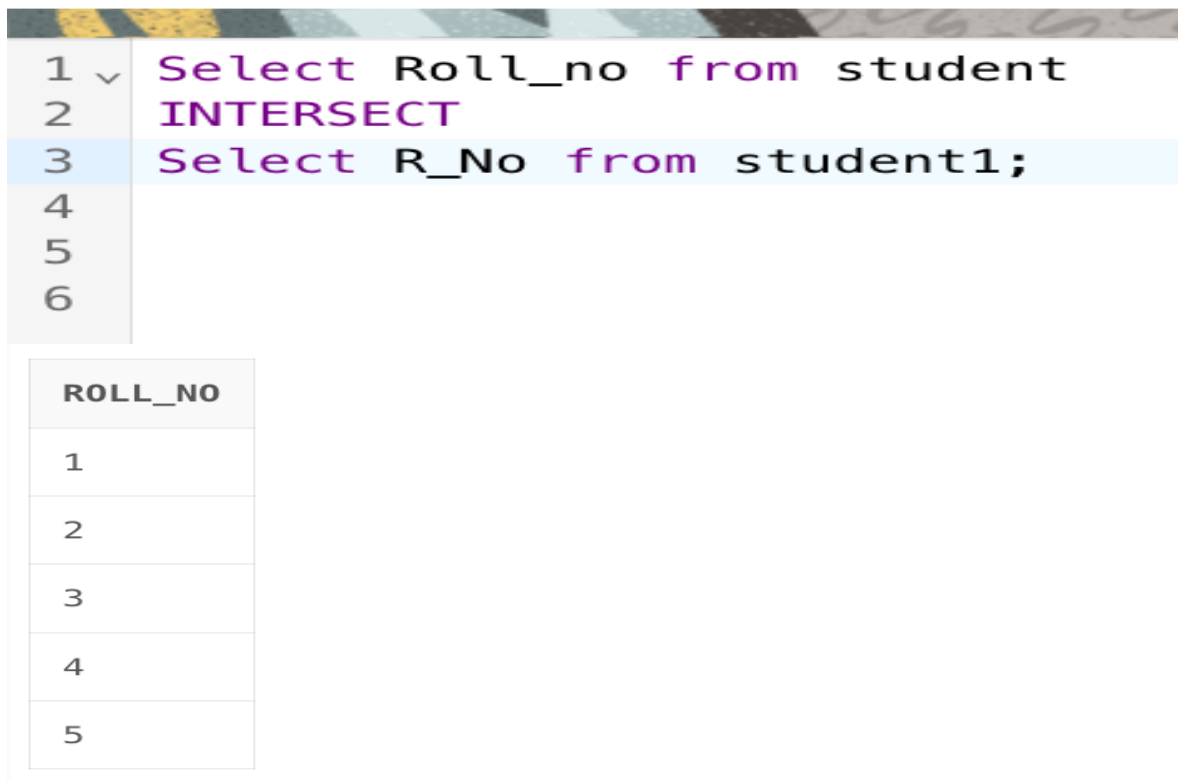
## Experiment-12

Find out the intersection of student and student1 tables.

### --Sql commands :-

```
Select Roll_no from student  
INTERSECT  
Select R_No from student1;
```

### SQL Worksheet



The screenshot shows an SQL worksheet with a query editor and a table view. The query editor contains the following SQL commands:

```
1 ✓ Select Roll_no from student  
2 INTERSECT  
3 Select R_No from student1;  
4  
5  
6
```

Below the query editor, there is a table with the following structure:

ROLL_NO
1
2
3
4
5

## Experiment-13

**Write a PL/SQL code to find the greatest of three numbers**

**--Sql commands :-**

```
DECLARE
    num1 NUMBER := 15;
    num2 NUMBER := 25;
    num3 NUMBER := 20;
    greatest_num NUMBER;
BEGIN
    IF num1 > num2 AND num1 > num3 THEN
        greatest_num := num1;
    ELSIF num2 > num1 AND num2 > num3 THEN
        greatest_num := num2;
    ELSE
        greatest_num := num3;
    END IF;
    DBMS_OUTPUT.PUT_LINE('The greatest number is: ' || greatest_num);
END;
```

```
DECLARE
    num1 NUMBER := 15;
    num2 NUMBER := 25;
    num3 NUMBER := 20;
    greatest_num NUMBER;
BEGIN
    IF num1 > num2 AND num1 > num3 THEN
        greatest_num := num1;
    ELSIF num2 > num1 AND num2 > num3 THEN
        greatest_num := num2;
    ELSE
        greatest_num := num3;
    END IF;
    DBMS_OUTPUT.PUT_LINE('The greatest number is: ' || greatest_num);
END;
/
```

STDIN

Output:

The greatest number is: 25

## Experiment-14

**Write a PL/SQL code to print the numbers from 1 to n.**

**--Sql commands :-**

```
DECLARE
    n NUMBER := 10;
    i NUMBER := 1;
BEGIN
    WHILE i <= n LOOP
        DBMS_OUTPUT.PUT_LINE(i);
        i := i + 1;
    END LOOP;
END;
```

/

```
DECLARE
    n NUMBER := 10;
    i NUMBER := 1;
BEGIN
    WHILE i <= n LOOP
        DBMS_OUTPUT.PUT_LINE(i);
        i := i + 1;
    END LOOP;
END;
```

/

STDIN

Output:

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

**n:** Set this to the maximum number you want to print.

**i:** Starts from 1 and increments by 1 in each iteration of the loop.

**DBMS\_OUTPUT.PUT\_LINE:** Prints the current value of i.

## Experiment-15

**Write a PL/SQL code to reverse a string using for loop.**

**--Sql commands :-**

```
DECLARE
original_str VARCHAR2(100) := 'Hello';
reversed_str VARCHAR2(100) := '';
str_length  NUMBER;
BEGIN
str_length := LENGTH(original_str);
FOR i IN REVERSE 1..str_length LOOP
    reversed_str := reversed_str || SUBSTR(original_str, i, 1);
END LOOP;
DBMS_OUTPUT.PUT_LINE('Original String: ' || original_str);
DBMS_OUTPUT.PUT_LINE('Reversed String: ' || reversed_str);
END;
```

```
DECLARE
    original_str VARCHAR2(100) := 'Hello';
    reversed_str VARCHAR2(100) := '';
    str_length  NUMBER;
BEGIN
    str_length := LENGTH(original_str);
    FOR i IN REVERSE 1..str_length LOOP
        reversed_str := reversed_str || SUBSTR(original_str, i, 1);
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Original String: ' || original_str);
    DBMS_OUTPUT.PUT_LINE('Reversed String: ' || reversed_str);
END;
```

STDIN

Output:

Original String: Hello  
Reversed String: olleH

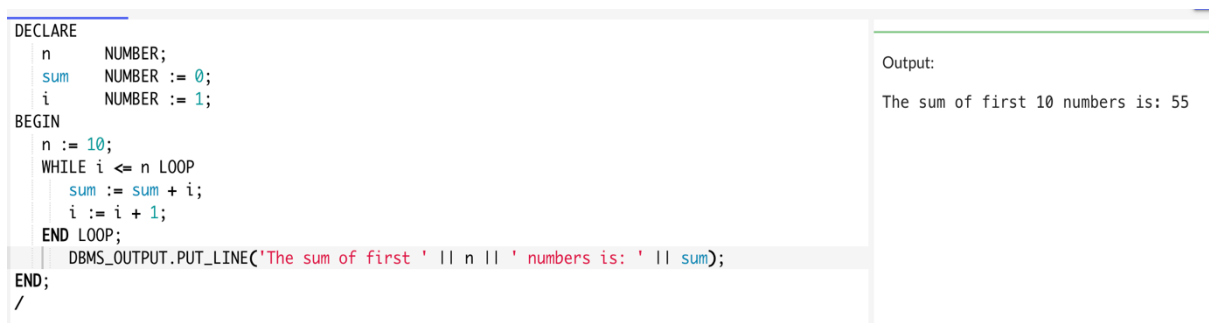
1. original\_str: Input string to be reversed.
2. str\_length: Stores the length of the input string.
3. FOR i IN REVERSE 1..str\_length: Loops from the last character to the first.
4. SUBSTR(original\_str, i, 1): Extracts the i-th character.
5. reversed\_str := reversed\_str || ...: Builds the reversed string.

## Experiment-16

**Write a PL/SQL code to find the sum of n numbers.**

**--Sql commands :-**

```
DECLARE
  n    NUMBER;
  sum  NUMBER := 0;
  i    NUMBER := 1;
BEGIN
  n := 10;
  WHILE i <= n LOOP
    sum := sum + i;
    i := i + 1;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('The sum of first ' || n || ' numbers is: ' || sum);
END;
/
```

A screenshot of a SQL IDE interface. On the left, a code editor displays the PL/SQL code from the previous block, with syntax highlighting. On the right, an 'Output' window shows the result of the execution: 'The sum of first 10 numbers is: 55'.

```
DECLARE
  n    NUMBER;
  sum  NUMBER := 0;
  i    NUMBER := 1;
BEGIN
  n := 10;
  WHILE i <= n LOOP
    sum := sum + i;
    i := i + 1;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('The sum of first ' || n || ' numbers is: ' || sum);
END;
/
```

Output:

The sum of first 10 numbers is: 55

**n:** Holds the value up to which the sum is calculated.

**sum:** Accumulates the sum of the first n numbers.

**i:** Counter used to iterate through numbers from 1 to n.

**WHILE LOOP:** Adds each number from 1 to n into sum.