**EXPERIMENT 1**

Title: Write query to create table Customer and order.

Objective: Create the following tables:

**Customer**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column\_name** | **Data type** | **Size** | **Constraint** |
| SID | Varchar2 | 4 | Primary Key |
| First\_Name | Char | 20 |  |
| Last\_name | Char | 20 |  |

**Orders**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column\_name** | **Data type** | **Size** | **Constraint** |
| Order\_ID | Varchar2 | 4 | Primary Key |
| Order\_date | Char | 20 |  |
| Customer\_SID | Varchar2 | 20 | Foreign Key |
| Amount | Number |  | Check > 20000 |

Pre-requisites:

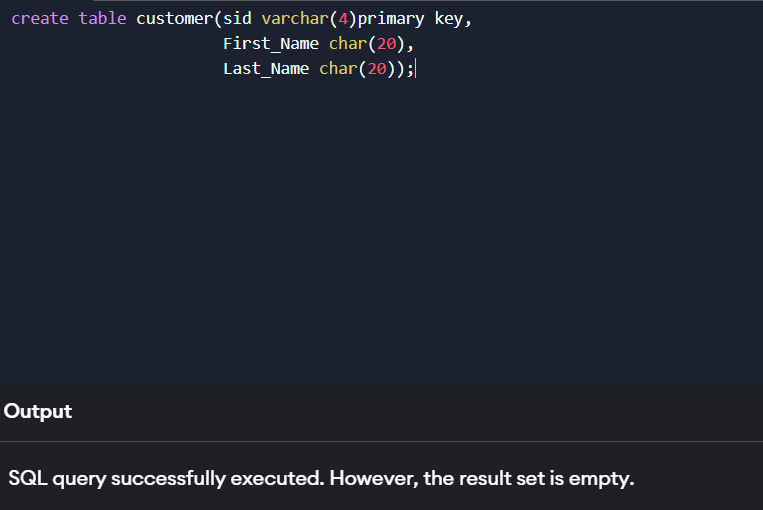
* Knowledge of RDBMS and DBMS
* Sql queries
* DDL query

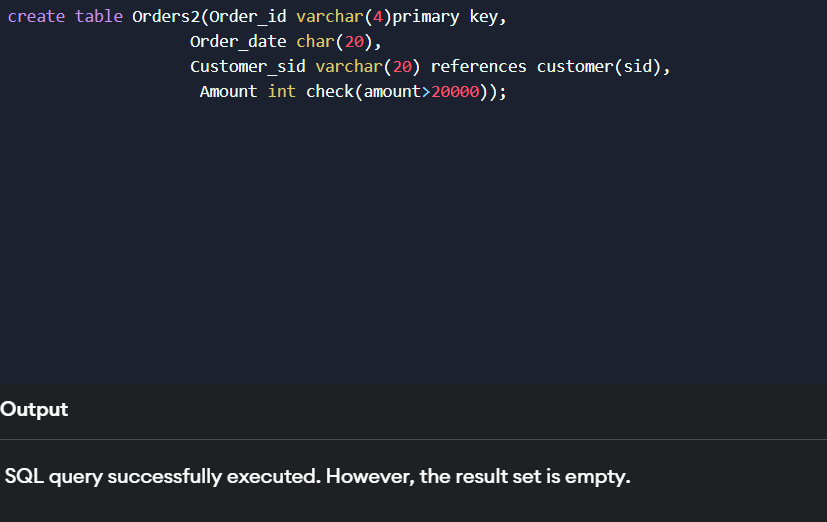
Query:

Create table Customer(sid varchar(4)primary key,First\_Name char(20),Last\_Name char(20));

Create table Orders2(Order\_id varchar(4) primary key, Order\_date char(20), Customer\_sid varchar(20) references customer(sid), Amount int check(amount>20000));

Output:

****

****

**EXPERIMENT 2**

Title:Write a query to insert records 5 records in Customer and Order table.

Objective: Insert five records for each table

Pre-requisites:

* Knowledge of SQL queries
* DDL query

Query:

**Customer values**

insert into customer values('1', 'Arun', 'Kumar');  
insert into customer values('2', 'Raja', 'Rogi');  
insert into customer values('3', 'Sumit', 'Kumar');  
insert into customer values('4', 'Jen', 'Joby');  
insert into customer values('5', 'Chinu', 'Gandhi');

**Order values**

insert into Orders2 values('101', '20-10-2012', '1', 25000);

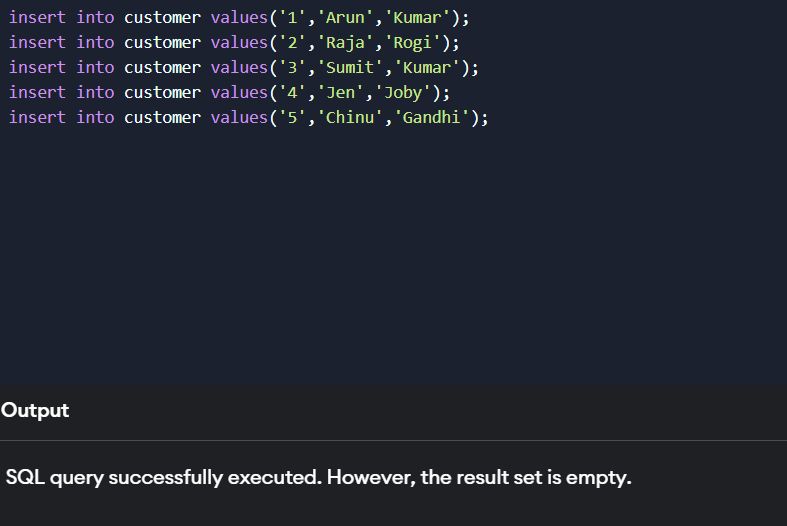
insert into Orders2 values('A12', '10-09-2024', '5',30000);

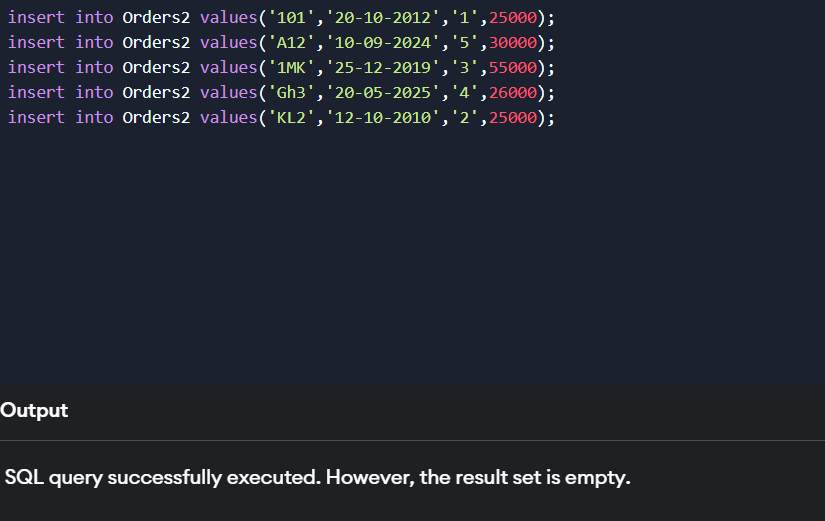
insert into Orders2 values('1MK', '25-12-2019','3',55000);

insert into Orders2 values('Gh3', '20-05-2025', '4', 26000);

insert into Orders2 values('KL2', '12-10-2010', '2',25000);

Output:

****

****

**EXPERIMENT 3**

Title: Write a query to show all records in table along with their amounts.

Objective: List the details of the customers along with the amount.

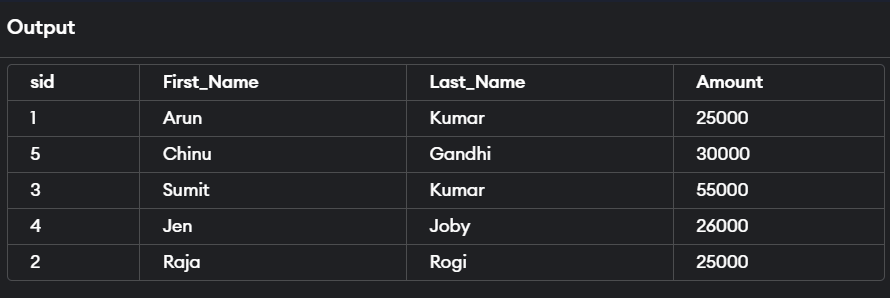
Pre-requisites:

* SQL queries
* DML commands

Query:

SELECT customer.sid,customer.First\_Name,customer.Last\_Name, Orders2.Amount FROM customer  
Inner join Orders2 on customer.sid = Orders2.customer\_sid;

Output:

****

**EXPERIMENT 4**

Title: Write a query to show records of customer name’s end with a

Objective: List the customers whose names end with “a”.

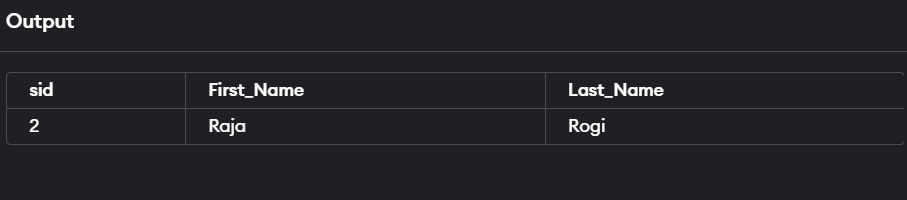
Pre-requisites:

* SQL queries
* DML queries

Query:

select \* from Customer where First\_Name like”%a”;

Output:

****

**EXPERIMENT 5**

Title: Write a query to show records of orders where amount is 21000 and 30000.

Objective: List the orders where amount is between 21000 and 30000

Pre-requisites:

* SQl queries
* DML queries

Query:

select \* from Orders2 where amount between “21000” and “30000”;

Output:

****

**EXPERIMENT 6**

Title: Write sql query to show records where amount is increased by 500.

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

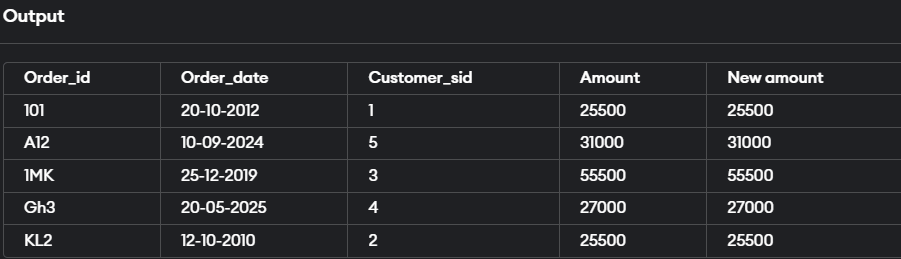
Pre-requisites:

* SQL queries
* DML queries

Query:

select \*,Amount as 'New amount' from Orders2;

Output:



**EXPERIMENT 7**

Title: Write sql query to show records with their order id and total amount of order done by that order id.

Objective: Display the order\_id and total amount of orders

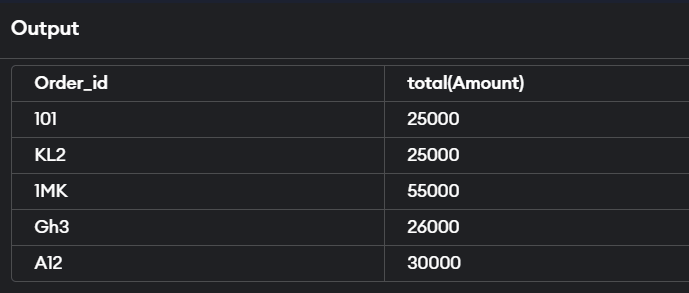
Pre-requisites:

* SQL query
* DML queries

Query:

select Order\_id,total(Amount) from Orders2 group by Customer\_sid;

Output:

****

**EXPERIMENT 8**

Title: Write a sql query to show records where amount is more than 15000.

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

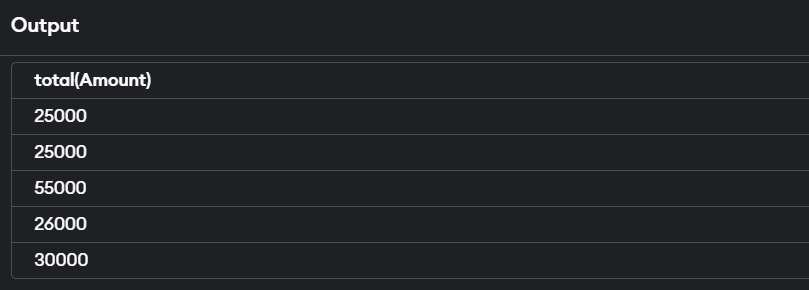
Pre-requisites:

* SQL query
* DML queries

Query:

Select total(Amount) from Orders2 where amount>15000 group by Customer\_sid;

Output:

****

**EXPERIMENT 9**

Title: Write query to create table Students and Student1.

Objective: Create the following tables

**Student**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column\_name** | **Data type** | **Size** | **Constraint** |
| RollNo | Varchar2 | 20 | Primary Key |
| Name | Char | 20 |  |
| Class | Varchar2 | 20 |  |
| Marks | Number | 6,2 |  |

**Student1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column\_name** | **Data type** | **Size** | **Constraint** |
| R\_No | Varchar2 | 20 | Primary Key |
| Name | Char | 20 |  |
| Class | Varchar2 | 20 |  |
| Marks | Number | 6,2 |  |

Pre-requisites:

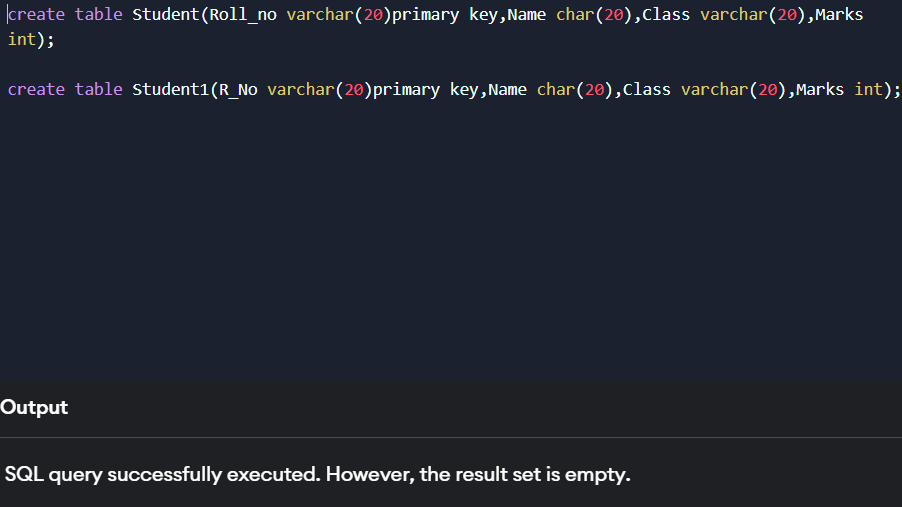
* SQL query
* DDL queries

Query:

create table Student(Roll\_no varchar(20)primary key,Name char(20),Class varchar(20),Marks int);

create table Student1(R\_No varchar(20)primary key,Name char(20),Class varchar(20),Marks int);

Output**:**

****

**EXPERIMENT 10**

Title: Write sql query to display records from student and student1 table.

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

Pre-requisites:

* SQL query
* DML queries

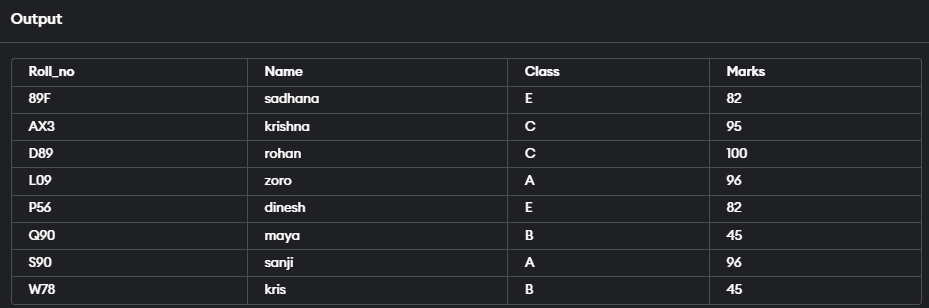
Query:

select \* from Student

UNION

select \* from Student1;

Output:



**EXPERIMENT 11**

Title: Write sql query to show records common in Student and Student1 table.

Objective: Find out the intersection of student and student1 tables.

Pre-requisites:

* SQL query
* DML queries

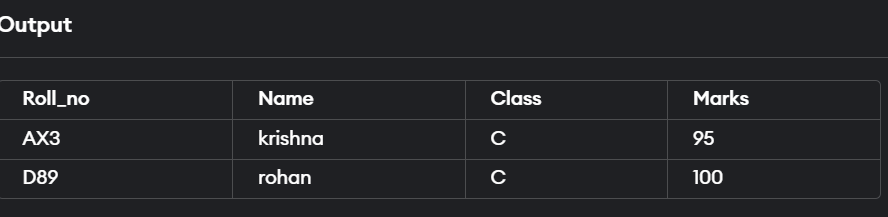
Query:

select \* from Student

INTERSECT

select \* from Student1;

Output:



**EXPERIMENT 12**

Title: write sql queries to show records using different joins.

Objective: Display the names of student and student1 tables using left and inner join.

Pre-requisites:

* SQL query
* DML queries
* JOINS

Query:

**LEFT JOIN**

SELECT Roll\_no FROM Student

LEFT JOIN Student1

ON Student.Roll\_no = Student1.R\_No;

**INNER JOIN**

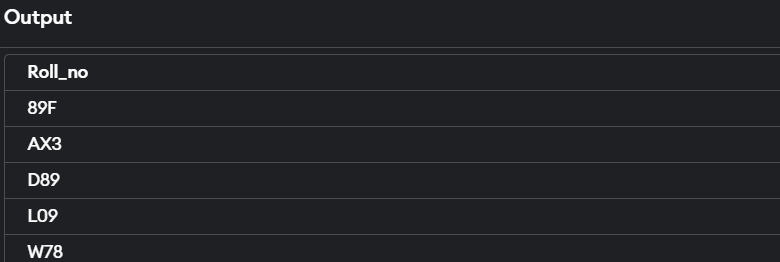
SELECT Roll\_no FROM Student

INNER JOIN Student1

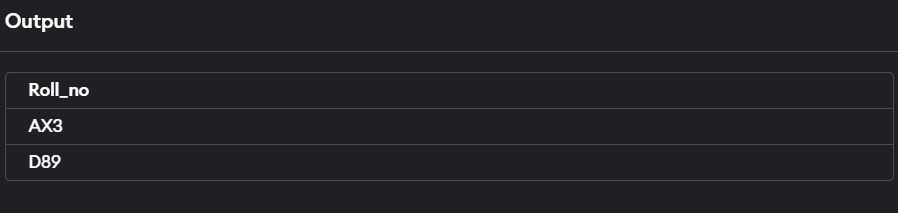
ON Student.Roll\_no = Student1.R\_No;

Output:

**LEFT JOIN**



**INNER JOIN**



**EXPERIMENT 13**

Title: PL/SQL queries to calculate

Objective: Write a PL/SQL block to calculate total salary of employee having employee number 100.

Pre-requisites:

* SQl query
* PL/SQL query

Query:

DECLARE

num1 NUMBER;

num2 NUMBER;

num3 NUMBER;

greatest NUMBER;

BEGIN

num1 := 15;

num2 := 25;

num3 := 10;  
 IF (num1 >= num2 AND num1 >= num3) THEN

greatest := num1;

ELSIF (num2 >= num1 AND num2 >= num3) THEN

greatest := num2;

ELSE

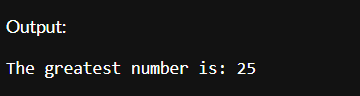
greatest := num3;

END IF;

DBMS\_OUTPUT.PUT\_LINE('The greatest number is: ' || greatest);

END;

Output:



**EXPERIMENT 14**

Title: Write pl/sql query to show number from 1 to n

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

Pre-requisites:

* SQL query
* PL/SQL queries

Query:

DECLARE

n NUMBER;

BEGIN

n := 10;

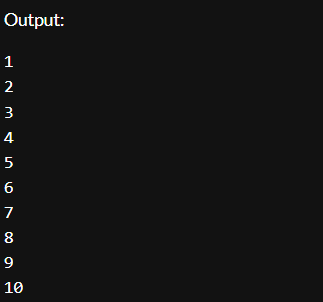
FOR i IN 1..n LOOP

DBMS\_OUTPUT.PUT\_LINE(i);

END LOOP;

END;

Output:



**EXPERIMENT 15**

Title: PL/SQL query to for reversing string.

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

Pre-requisites:

* SQL query
* PL/SQL queries

Query:

DECLARE

original\_string VARCHAR2(100) := 'Hello, World!';

reversed\_string VARCHAR2(100) := '';

string\_length INTEGER;

BEGIN

string\_length := LENGTH(original\_string);

FOR i IN REVERSE 1 .. string\_length LOOP

reversed\_string := reversed\_string || SUBSTR(original\_string, i, 1);

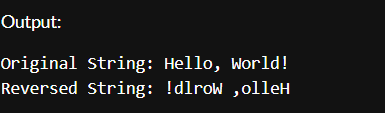
END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Original String: ' || original\_string);

DBMS\_OUTPUT.PUT\_LINE('Reversed String: ' || reversed\_string);

END;

Output:



**EXPERIMENT 16**

Title:PL/SQL command for finding factorial of number.

Objective:. Write a PL/SQL query to find factorial of a number.

Pre-requisites:

* SQL query
* PL/SQL query

PL/SQL queries

Query:

DECLARE

num INTEGER := 5;

factorial INTEGER := 1;

BEGIN

FOR i IN 1 .. num LOOP

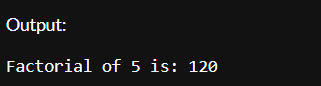
factorial := factorial \* i;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Factorial of ' || num || ' is: ' || factorial);

END;

Output:



**EXPERIMENT 17**

Title: PL/SQL command for finding power of number.

Objective: Write a PL/SQL query to find power of a number.

Pre-requisites:

* SQL query
* PL/SQL query

Query:

DECLARE

base NUMBER := 2;

exponent INTEGER := 3;

result NUMBER := 1;

BEGIN

FOR i IN 1 .. exponent LOOP

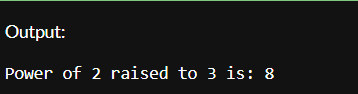
result := result \* base;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Power of ' || base || ' raised to ' || exponent || ' is: ' || result);

END;

Output:



**EXERCISE 18**

Title: PL/SQL command for finding reverse of string.

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

Pre-requisites:

* SQL query
* PL/SQL query

QUERY:

SET SERVEROUTPUT ON;

DECLARE

original\_string VARCHAR2(100) := 'Hello, World!'; -- Input string

reversed\_string VARCHAR2(100) := ''; -- Variable to hold the reversed string

string\_length INTEGER; -- Length of the original string

BEGIN

-- Get the length of the original string

string\_length := LENGTH(original\_string);

-- Loop through the original string in reverse order

FOR i IN REVERSE 1..string\_length LOOP

-- Concatenate each character to the reversed string

reversed\_string := reversed\_string || SUBSTR(original\_string, i, 1);

END LOOP;

-- Output the reversed string

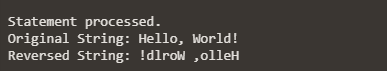
DBMS\_OUTPUT.PUT\_LINE('Original String: ' || original\_string);

DBMS\_OUTPUT.PUT\_LINE('Reversed String: ' || reversed\_string);

END;

/

OUTPUT:



**EXERCISE 19**

Title: PL/SQL command for finding sum of number.

Objective: Write a PL/SQL code to find suum of n numbers.

Pre-requisites:

* SQL query
* PL/SQL query

QUERY:

DECLARE

n NUMBER; -- Number of elements to sum

num NUMBER; -- Variable to hold each input number

total\_sum NUMBER := 0; -- Variable to hold the total sum

BEGIN

-- Prompt for the number of elements

DBMS\_OUTPUT.PUT\_LINE('Enter the number of elements to sum:');

-- Assume n is provided via some input mechanism, such as a substitution variable or input form

-- For this example, we can assign it directly, or you can modify this part to accept input

n := 5; -- Change this value as needed

FOR i IN 1..n LOOP

-- Prompt for each element - in real applications, you would capture input dynamically

-- Here, for demonstration, we can simulate input:

DBMS\_OUTPUT.PUT\_LINE('Enter number ' || i || ':');

-- Replace this with your input mechanism

-- For example purposes, we're just prompting and assuming fixed values:

num := i \* 10; -- This is just a placeholder, replace it with actual input capture

-- Add the number to the total sum

total\_sum := total\_sum + num;

END LOOP;

-- Output the result

DBMS\_OUTPUT.PUT\_LINE('The total sum of the ' || n || ' numbers is: ' || total\_sum);

EXCEPTION

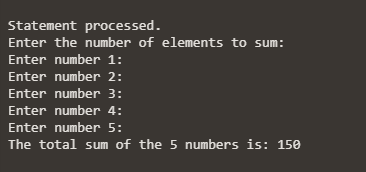
WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;

/

**OUTPUT:**

****

**EXERCISE 20**

Title: PL/SQL command for display the empno, ename, job of employees of department number 10

Objective: Write a PL/SQL code to consider a PL/SQL code to display the empno, ename, job of employees of department number 10

Pre-requisites:

* SQL query
* PL/SQL query

QUERY

SET SERVEROUTPUT ON;

DECLARE

CURSOR emp\_cursor IS

SELECT empno, ename, job

FROM employees

WHERE deptno = 10; -- Filter for department number 10

emp\_record emp\_cursor%ROWTYPE; -- Record type to hold cursor data

BEGIN

-- Open the cursor and fetch each employee record

OPEN emp\_cursor;

LOOP

FETCH emp\_cursor INTO emp\_record;

EXIT WHEN emp\_cursor%NOTFOUND; -- Exit loop when no more records

-- Display the employee details

DBMS\_OUTPUT.PUT\_LINE('Emp No: ' || emp\_record.empno ||

', Name: ' || emp\_record.ename ||

', Job: ' || emp\_record.job);

END LOOP;

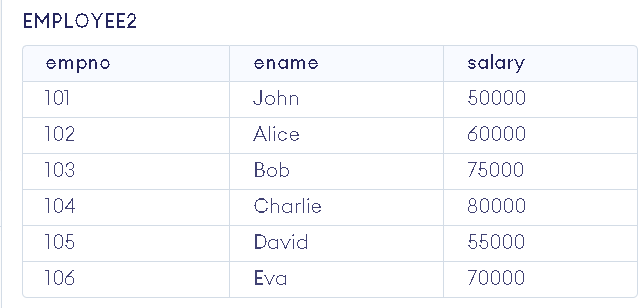
-- Close the cursor

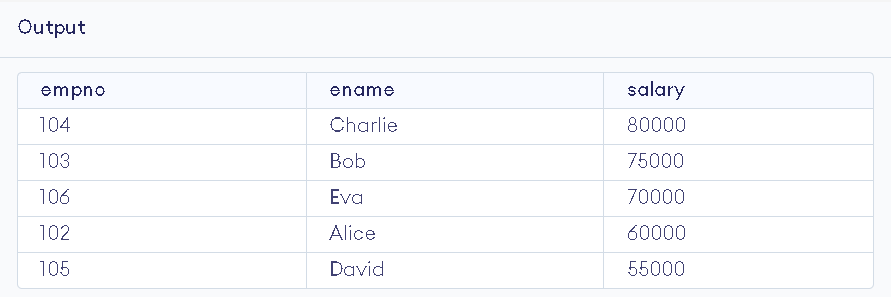
CLOSE emp\_cursor;

END;

/

Output:





**EXERCISE 21**

Title: To Consider a PL/SQL procedure that accepts 2 numbers & return addition,

subtraction, multiplication & division of two numbers using stored procedure AND local procedure.

Objective: To Consider a PL/SQL procedure that accepts 2 numbers & return addition,subtraction, multiplication & division of two numbers using stored procedure AND local procedure.

Pre-requisites:

* SQL query
* PL/SQL query

QUERY

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE calculate\_operations (

p\_num1 IN NUMBER,

p\_num2 IN NUMBER,

p\_add OUT NUMBER,

p\_sub OUT NUMBER,

p\_mul OUT NUMBER,

p\_div OUT NUMBER

) AS

-- Local procedure to perform calculations

PROCEDURE perform\_calculations (

num1 IN NUMBER,

num2 IN NUMBER,

add\_result OUT NUMBER,

sub\_result OUT NUMBER,

mul\_result OUT NUMBER,

div\_result OUT NUMBER

) IS

BEGIN

add\_result := num1 + num2;

sub\_result := num1 - num2;

mul\_result := num1 \* num2;

-- Check for division by zero

IF num2 != 0 THEN

div\_result := num1 / num2;

ELSE

div\_result := NULL; -- or you can raise an exception

END IF;

END perform\_calculations;

BEGIN

-- Call the local procedure to perform calculations

perform\_calculations(p\_num1, p\_num2, p\_add, p\_sub, p\_mul, p\_div);

END calculate\_operations;

/

Output:



**Exercise 22**

Title: To Write a PL/SQL block to show the use of NO\_DATA FOUND exception

Objective: To Write a PL/SQL block to show the use of NO\_DATA FOUND exception Pre-requisites:

* SQL query
* PL/SQL query

QUERY

DECLARE

v\_employee\_id NUMBER := 100; -- Assuming we are looking for an employee with ID 100

v\_first\_name VARCHAR2(50);

v\_last\_name VARCHAR2(50);

BEGIN

-- Attempt to fetch employee details

SELECT first\_name, last\_name

INTO v\_first\_name, v\_last\_name

FROM employees

WHERE employee\_id = v\_employee\_id;

-- Display the employee details if found

DBMS\_OUTPUT.PUT\_LINE('Employee Found: ' || v\_first\_name || ' ' || v\_last\_name);

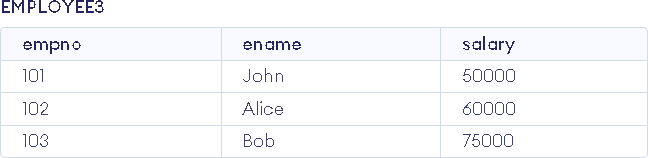
EXCEPTION

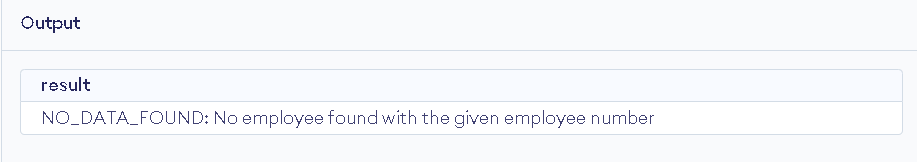
WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No employee found with ID ' || v\_employee\_id);

END;

Output





**Exercise 22**

Title: To Write a PL/SQL block to show use of local function

Objective: To Consider a PL/SQL code that accepts 2 numbers & return addition, subtraction, multiplication & division of two numbers using stored functions and local function.Pre-requisites:

* SQL query
* PL/SQL query

QUERY

CREATE OR REPLACE PACKAGE math\_operations AS

FUNCTION add\_numbers(num1 NUMBER, num2 NUMBER) RETURN NUMBER;

FUNCTION subtract\_numbers(num1 NUMBER, num2 NUMBER) RETURN NUMBER;

FUNCTION multiply\_numbers(num1 NUMBER, num2 NUMBER) RETURN NUMBER;

FUNCTION divide\_numbers(num1 NUMBER, num2 NUMBER) RETURN NUMBER;

END math\_operations;

/

CREATE OR REPLACE PACKAGE BODY math\_operations AS

FUNCTION add\_numbers(num1 NUMBER, num2 NUMBER) RETURN NUMBER IS

BEGIN

RETURN num1 + num2;

END add\_numbers;

FUNCTION subtract\_numbers(num1 NUMBER, num2 NUMBER) RETURN NUMBER IS

BEGIN

RETURN num1 - num2;

END subtract\_numbers;

FUNCTION multiply\_numbers(num1 NUMBER, num2 NUMBER) RETURN NUMBER IS

BEGIN

RETURN num1 \* num2;

END multiply\_numbers;

FUNCTION divide\_numbers(num1 NUMBER, num2 NUMBER) RETURN NUMBER IS

BEGIN

IF num2 = 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Division by zero is not allowed');

END IF;

RETURN num1 / num2;

END divide\_numbers;

END math\_operations;

/

DECLARE

num1 NUMBER;

num2 NUMBER;

result\_add NUMBER;

result\_subtract NUMBER;

result\_multiply NUMBER;

result\_divide NUMBER;

BEGIN

-- Accepting two numbers; these values can be taken from user input or hardcoded

num1 := 10; -- Example value

num2 := 5; -- Example value

-- Calling the stored functions from the package

result\_add := math\_operations.add\_numbers(num1, num2);

result\_subtract := math\_operations.subtract\_numbers(num1, num2);

result\_multiply := math\_operations.multiply\_numbers(num1, num2);

-- Handle division with exception

BEGIN

result\_divide := math\_operations.divide\_numbers(num1, num2);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

-- Display the results

DBMS\_OUTPUT.PUT\_LINE('Addition: ' || result\_add);

DBMS\_OUTPUT.PUT\_LINE('Subtraction: ' || result\_subtract);

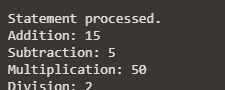
DBMS\_OUTPUT.PUT\_LINE('Multiplication: ' || result\_multiply);

DBMS\_OUTPUT.PUT\_LINE('Division: ' || result\_divide);

END;

/

Output



**Exercise 23**

Title: To Write a PL/SQL block to show use of TOO MANY ROWS

Objective: To Write a PL/SQL block to show the use of TOO\_MANY ROWS exception

Pre-requisites:

* SQL query
* PL/SQL query

Query

SET SERVEROUTPUT ON;

DECLARE

v\_deptno NUMBER := 10; -- Change this to a department number that has multiple employees

v\_ename VARCHAR2(100);

BEGIN

-- Attempt to select the employee name based on department number

SELECT ename INTO v\_ename

FROM employees

WHERE deptno = v\_deptno;

-- If the employee is found, display the name

DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || v\_ename);

EXCEPTION

WHEN TOO\_MANY\_ROWS THEN

-- Handle the exception when too many rows are found

DBMS\_OUTPUT.PUT\_LINE('Error: More than one employee found in department number: ' || v\_deptno);

WHEN NO\_DATA\_FOUND THEN

-- Handle the exception when no data is found

DBMS\_OUTPUT.PUT\_LINE('No employee found in department number: ' || v\_deptno);

WHEN OTHERS THEN

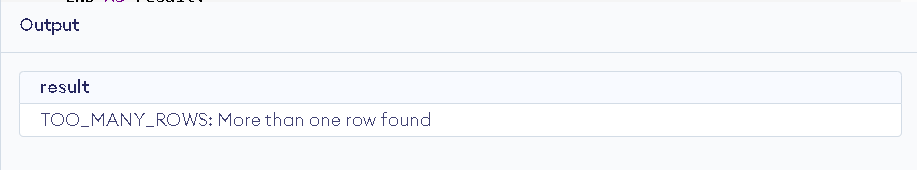
-- Handle any other exceptions

DBMS\_OUTPUT.PUT\_LINE('An unexpected error occurred: ' || SQLERRM);

END;

/

Output



**Exercise 24**

Title: To Write a PL/SQL block to show use of ZERO DIVIDE

Objective: Write a PL/SQL block to show the use of ZERO\_DIVIDE exception

Pre-requisites:

* SQL query
* PL/SQL query

Query

DECLARE

numerator NUMBER := 10;

denominator NUMBER := 0; -- Set this to 0 to trigger the ZERO\_DIVIDE exception

result NUMBER;

BEGIN

-- Attempt to perform the division

result := numerator / denominator;

DBMS\_OUTPUT.PUT\_LINE('Result: ' || result);

EXCEPTION

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Division by zero is not allowed.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An unexpected error occurred: ' || SQLERRM);

END;

Output



**Exercise 25**

Title: To Write a PL/SQL block to show audit of table.

Objective: To create a trigger on the emp table, which store the empno& operation in the table auditor for each operation i.e. Insert, Update & Delete.

Pre-requisites:

* SQL query
* PL/SQL query

Query

CREATE TABLE auditor (

audit\_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

empno NUMBER,

operation VARCHAR2(10),

operation\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE OR REPLACE TRIGGER trg\_audit\_emp

AFTER INSERT OR UPDATE OR DELETE ON emp

FOR EACH ROW

BEGIN

IF INSERTING THEN

INSERT INTO auditor (empno, operation)

VALUES (:NEW.empno, 'INSERT');

ELSIF UPDATING THEN

INSERT INTO auditor (empno, operation)

VALUES (:NEW.empno, 'UPDATE');

ELSIF DELETING THEN

INSERT INTO auditor (empno, operation)

VALUES (:OLD.empno, 'DELETE');

END IF;

END trg\_audit\_emp;

/

-- Insert a new employee

INSERT INTO emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)

VALUES (1, 'John Doe', 'Developer', NULL, SYSDATE, 50000, NULL, 10);

-- Update the employee

UPDATE emp

SET sal = 55000

WHERE empno = 1;

-- Delete the employee

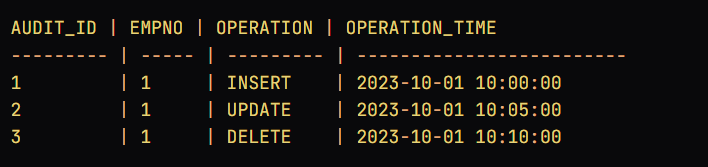
DELETE FROM emp

WHERE empno = 1;

-- Check the auditor table

SELECT \* FROM auditor;

Output



**Exercise 26**

Title: To Write a PL/SQL block to no operation code.

Objective: To create a trigger so that no operation can be performed on emp table

Pre-requisites:

* SQL query
* PL/SQL query

Query

CREATE OR REPLACE TRIGGER trg\_prevent\_emp\_operations

BEFORE INSERT OR UPDATE OR DELETE ON emp

BEGIN

RAISE\_APPLICATION\_ERROR(-20001, 'No operations are allowed on the emp table.');

END trg\_prevent\_emp\_operations;

/

-- Attempt to insert a new employee

INSERT INTO emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)

VALUES (1, 'John Doe', 'Developer', NULL, SYSDATE, 50000, NULL, 10);

