

EXPERIMENT 1

Title: Write query to create table Customer and order.

Objective: Create the following tables:

Customer

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

Orders

<u>Column name</u>	<u>Data type</u>	<u>Size</u>	<u>Constraint</u>
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:

```
create table customer(sid varchar(4)primary key,  
                      First_Name char(20),  
                      Last_Name char(20));|
```

Output

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

```
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

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

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:

```
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');
```

Output

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

```
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

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

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:

Output			
sid	First_Name	Last_Name	Amount
1	Arun	Kumar	25000
5	Chinu	Gandhi	30000
3	Sumit	Kumar	55000
4	Jen	Joby	26000
2	Raja	Rogi	25000

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:

Output		
sid	First_Name	Last_Name
2	Raja	Rogi

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:

Output			
Order_id	Order_date	Customer_sid	Amount
101	20-10-2012	1	25000
A12	10-09-2024	5	30000
Gh3	20-05-2025	4	26000
KL2	12-10-2010	2	25000

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:

Output				
Order_id	Order_date	Customer_sid	Amount	New amount
101	20-10-2012	1	25500	25500
A12	10-09-2024	5	31000	31000
1MK	25-12-2019	3	55500	55500
Gh3	20-05-2025	4	27000	27000
KL2	12-10-2010	2	25500	25500

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:

Output	
Order_id	total(Amount)
101	25000
KL2	25000
1MK	55000
Gh3	26000
A12	30000

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:

Output
total(Amount)
25000
25000
55000
26000
30000

EXPERIMENT 9

Title: Write query to create table Students and Student1.

Objective: Create the following tables

Student

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

Student1

<u>Column name</u>	<u>Data type</u>	<u>Size</u>	<u>Constraint</u>
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:

```
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

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

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:

Output			
Roll_no	Name	Class	Marks
89F	sadhana	E	82
AX3	krishna	C	95
D89	rohan	C	100
L09	zoro	A	96
P56	dinesh	E	82
Q90	maya	B	45
S90	sanji	A	96
W78	kris	B	45

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:

Output			
Roll_no	Name	Class	Marks
AX3	krishna	C	95
D89	rohan	C	100

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

Output	
Roll_no	
89F	
AX3	
D89	
L09	
W78	

INNER JOIN

Output

Roll_no
AX3
D89

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:

Output:

The greatest number is: 25

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;
```

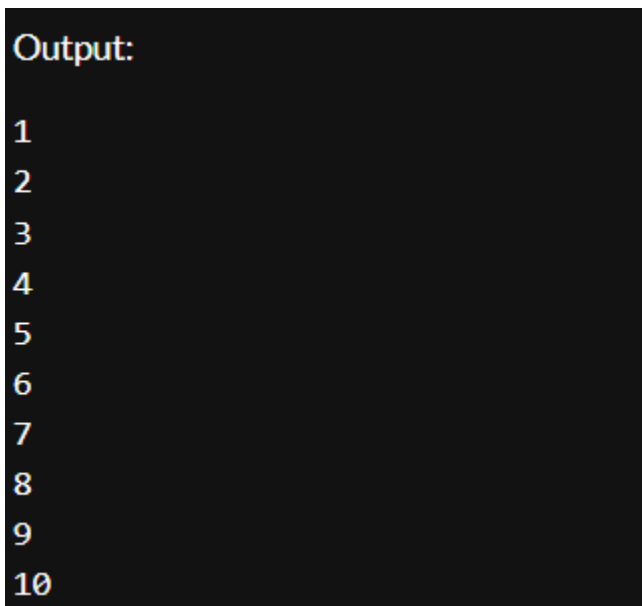
```
    FOR i IN 1..n LOOP
```

```
        DBMS_OUTPUT.PUT_LINE(i);
```

```
    END LOOP;
```

```
END;
```

Output:



The screenshot shows a dark-themed output window with the word "Output:" in orange at the top left. Below it, the numbers 1 through 10 are listed vertically in white text, representing the output of the PL/SQL program.

```
Output:
1
2
3
4
5
6
7
8
9
10
```

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:

Output:

Original String: Hello, World!

Reversed String: !dlrow ,olleH

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:

Output:

Factorial of 5 is: 120

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:

Output:

Power of 2 raised to 3 is: 8

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:

```
Statement processed.  
Original String: Hello, World!  
Reversed String: !dlrow ,olleH
```


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:

```
Statement processed.
Enter the number of elements to sum:
Enter number 1:
Enter number 2:
Enter number 3:
Enter number 4:
Enter number 5:
The total sum of the 5 numbers is: 150
```

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:

EMPLOYEE2

empno	ename	salary
101	John	50000
102	Alice	60000
103	Bob	75000
104	Charlie	80000
105	David	55000
106	Eva	70000

Output

empno	ename	salary
104	Charlie	80000
103	Bob	75000
106	Eva	70000
102	Alice	60000
105	David	55000

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:

```

Procedure created.

```

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

EMPLOYEE3

empno	ename	salary
101	John	50000
102	Alice	60000
103	Bob	75000

Output

result

NO_DATA_FOUND: No employee found with the given employee number

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

```
Statement processed.  
Addition: 15  
Subtraction: 5  
Multiplication: 50  
Division: 2
```

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

Output	
result	TOO_MANY_ROWS: More than one row found

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

```
Statement processed.  
Error: Division by zero is not allowed.
```


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

AUDIT_ID	EMPNO	OPERATION	OPERATION_TIME
-----	-----	-----	-----
1	1	INSERT	2023-10-01 10:00:00
2	1	UPDATE	2023-10-01 10:05:00
3	1	DELETE	2023-10-01 10:10:00

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);
```

```
ORA-20001: No operations are allowed on the emp table.  
ORA-06512: at "YOUR_SCHEMA.TRG_PREVENT_EMP_OPERATIONS", line 2  
ORA-04088: error during execution of trigger 'YOUR_SCHEMA.TRG_PREVEN'
```

