**Q1 ) Implementation of DDL commands of SQL with suitable examples**

**• Create table:-** CREATE TABLE student ( RollNo INT, sname VARCHAR(50), dateOfBirth DATE);

**• Alter table:-** ALTER TABLE student ADD address TEXT;

**• Drop Table:-** DROP TABLE student;

**Add new column:-** ALTER TABLE student

ADD phone INTEGER;

**Write a PL/SQL Program to Find Factorial of a Number**

SET SERVEROUTPUT ON;

DECLARE num NUMBER;

factorial NUMBER := 1;

num := &input\_number; IF num < 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Factorial is not defined for negative numbers.');

ELSE -- Calculate the factorial

FOR i IN 1..num LOOP

factorial := factorial \* i;

END LOOP;

-- Output the factorial

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

END IF; END;

**Implementation of different types of function with suitable examples**

**• By using Number function**

SELECT ABS(-10) AS AbsoluteValue;

-- Output: 10

**• By using Aggregate Function**

SELECT SUM(salary) AS TotalSalary FROM employees;

-- Output: TotalSalary = Sum of all salaries in the employees table

**• By using Character Function**

SELECT CONCAT('Hello', ' ', 'World') AS Greeting;

-- Output: Hello World

**• By using Conversion Function**

SELECT TO\_CHAR(12345) AS StringValue;

-- Output: '12345'

**• By using Date Function**

SELECT MONTH('2023-07-17') AS MonthValue;

-- Output: 7

**Q. 1. PL/SQL Program to Print Table of a Number**

SET SERVEROUTPUT ON;

DECLARE

num NUMBER;

multiplier NUMBER := 1;

product NUMBER;

BEGIN

-- Input the number for which you want to print the table

num := &input\_number;

-- Print the table of the given number

WHILE multiplier <= 10 LOOP

product := num \* multiplier;

DBMS\_OUTPUT.PUT\_LINE(num || ' x ' || multiplier || ' = ' || product);

multiplier := multiplier + 1;

END LOOP; END;

**Q. 2. Implementation of different types of Joins • Inner Join**

SELECT \* FROM Customer

INNER JOIN Order ON Customer.customer\_id = Order.customer\_id;

• **Outer Join**

SELECT \* FROM Customer

LEFT OUTER JOIN Cust\_order ON Customer.customer\_id = Cust\_order.customer\_id;

• **Natural Join**

SELECT \* FROM Customer

NATURAL JOIN Cust\_order;

**Q. 2. Write a program to implement SQL Cursors**.

SET SERVEROUTPUT ON;

DECLARE

CURSOR employee\_cursor IS

SELECT employee\_id, first\_name, last\_name

FROM employees;

emp\_id employees.employee\_id%TYPE;

emp\_first\_name employees.first\_name%TYPE;

emp\_last\_name employees.last\_name%TYPE;

BEGIN

OPEN employee\_cursor;

LOOP

FETCH employee\_cursor INTO emp\_id, emp\_first\_name, emp\_last\_name;

EXIT WHEN employee\_cursor%NOTFOUND; -- Process the data fetched from the cursor

DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || emp\_id);

DBMS\_OUTPUT.PUT\_LINE('First Name: ' || emp\_first\_name);

DBMS\_OUTPUT.PUT\_LINE('Last Name: ' || emp\_last\_name);

DBMS\_OUTPUT.PUT\_LINE('-------------------');

END LOOP;

CLOSE employee\_cursor;

END;

**Q. 1 Study & Implementation of SQL Triggers**

-- Create a sample table

CREATE TABLE employees (

employee\_id INT,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

salary INT

);

-- Create a trigger that updates the salary of an employee

-- if it exceeds a certain threshold

CREATE OR REPLACE TRIGGER update\_salary\_trigger

BEFORE INSERT OR UPDATE ON employees

FOR EACH ROW

BEGIN

IF :NEW.salary > 100000 THEN

:NEW.salary := 100000; -- Set the salary to the maximum allowed value

DBMS\_OUTPUT.PUT\_LINE('Salary exceeded the maximum limit. Setting to $100,000.');

END IF;

END;

/

INSERT INTO employees (employee\_id, first\_name, last\_name, salary)

VALUES (1, 'John', 'Doe', 120000);