**General Instructions:**

* **Follow the instructions given in each section.**
* **Make sure that you attempt the questions in order.**

**SECTION-A (10\*1 mark=10 marks)**

***(All questions are compulsory)***

1. Which type of procedure in DBMS returns a value?
   1. Stored Procedure
   2. Trigger
   3. **Function**
   4. Cursor
2. Which of the following statements is true about triggers in DBMS?
   1. **Triggers are automatically executed in response to certain database events**
   2. Triggers are used to query and retrieve data from the database
   3. Triggers are similar to stored procedures but cannot have input parameters
   4. Triggers are only used for database administration tasks
3. What is the purpose of a database trigger?
   1. To enforce referential integrity
   2. To perform calculations on database columns
   3. **To maintain audit logs of database changes**
   4. To optimize query performance
4. Which statement is true about package bodies in DBMS?
   1. Package bodies must be created before creating the package specification
   2. **Package bodies can exist without a corresponding package specification**
   3. Package bodies contain only public procedures and functions
   4. Package bodies are stored separately from the package specification
5. Which of the following is not an advantage of using packages in DBMS?
   1. Improved performance through the use of optimized code
   2. Encapsulation of related procedures and functions for better organization
   3. **Simplified data modeling and schema design**
   4. Reduced network traffic by reducing the number of round trips to the database server
6. The two-phase locking protocol in concurrency control ensures:
   1. Deadlock prevention
   2. Data consistency
   3. **Serializability**
   4. Atomicity
7. The technique used to detect and resolve conflicts among concurrent transactions is called:
   1. Two-phase locking
   2. Timestamp ordering
   3. **Deadlock detection**
   4. Deadlock prevention
8. Which of the following is a drawback of strict two-phase locking (S2PL)?
   1. Increased concurrency
   2. **Increased deadlock probability**
   3. Reduced consistency
   4. Reduced durability
9. Which of the following is a property of serializability in a DBMS?
   1. Atomicity
   2. Consistency
   3. **Isolation**
   4. Durability
10. Which of the following is used to enforce strict two-phase locking (S2PL)?
    1. Shared locks
    2. Exclusive locks
    3. **Intent locks**
    4. Latches

**SECTION-B (5\*2 mark=10 marks)**

***(All questions are compulsory)***

1. Which of the following is true concerning triggers?
   1. You do not create them with SQL.
   2. They execute against only some applications that access a database.
   3. **They have an event, condition, and action.**
   4. They cannot cascade (cause another trigger to fire).
2. What is the maximum number of triggers that can apply to a single table?
3. 14
4. 10
5. **12**
6. 16
7. A package will have which of these mandatory parts?
8. Package specification
9. Package body or definition
10. **Both A & B**
11. None of the above
12. In the PL/SQL block below, how many rows will be inserted in the messages table?

DECLARE

v\_start\_sales NUMBER := 2;

v\_end\_sales NUMBER := 100;

BEGIN

FOR i IN v\_start\_sales..v\_end\_sales LOOP

INSERT INTO messages(msgid)

VALUES v\_start\_sales;

END LOOP;

END;

1. 0
2. **99**
3. 1
4. 100
5. What is the maximum number of ELSE clauses that can be included in an IF clause that is not nested?
6. **1**
7. 0
8. 15
9. Any number

**SECTION-C(Coding Question) (4x5 marks=20 marks)**

1. Write a bodiless package to find the maximum and minimum values from a given list of numbers and display the result.

Solution:

**CREATE OR REPLACE PACKAGE stats\_pkg AS**

**END stats\_pkg;**

**/**

**CREATE OR REPLACE PACKAGE BODY stats\_pkg AS**

**FUNCTION find\_max(list\_arr NUMBER\_ARRAY) RETURN NUMBER IS**

**max\_val NUMBER := list\_arr(1);**

**BEGIN**

**FOR i IN 2..list\_arr.COUNT LOOP**

**IF list\_arr(i) > max\_val THEN**

**max\_val := list\_arr(i);**

**END IF;**

**END LOOP;**

**RETURN max\_val;**

**END find\_max;**

**FUNCTION find\_min(list\_arr NUMBER\_ARRAY) RETURN NUMBER IS**

**min\_val NUMBER := list\_arr(1);**

**BEGIN**

**FOR i IN 2..list\_arr.COUNT LOOP**

**IF list\_arr(i) < min\_val THEN**

**min\_val := list\_arr(i);**

**END IF;**

**END LOOP;**

**RETURN min\_val;**

**END find\_min;**

**PROCEDURE display\_stats(list\_arr NUMBER\_ARRAY) IS**

**max\_val NUMBER;**

**min\_val NUMBER;**

**BEGIN**

**max\_val := find\_max(list\_arr);**

**min\_val := find\_min(list\_arr);**

**DBMS\_OUTPUT.PUT\_LINE('Maximum value: ' || max\_val);**

**DBMS\_OUTPUT.PUT\_LINE('Minimum value: ' || min\_val);**

**END display\_stats;**

**END stats\_pkg;**

**/**

1. Write a PL/SQL program that display employees' information from a specific department.

Solution:

**-- Create the "employee" table**

**CREATE TABLE employee (**

**employee\_id NUMBER PRIMARY KEY,**

**first\_name VARCHAR2(50),**

**last\_name VARCHAR2(50),**

**department VARCHAR2(50),**

**salary NUMBER**

**);**

**-- Insert sample records into the "employee" table**

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

**VALUES (1, 'John', 'Doe', 'HR', 50000);**

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

**VALUES (2, 'Jane', 'Smith', 'Finance', 60000);**

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

**VALUES (3, 'Michael', 'Johnson', 'IT', 70000);**

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

**VALUES (4, 'Merry', 'Agarwal', 'IT', 50000);**

**CREATE OR REPLACE PROCEDURE display\_employees\_by\_department(p\_department VARCHAR2) AS**

**CURSOR c\_employees IS**

**SELECT \* FROM employee WHERE department = p\_department;**

**v\_employee employee%ROWTYPE;**

**BEGIN**

**OPEN c\_employees;**

**LOOP**

**FETCH c\_employees INTO v\_employee;**

**EXIT WHEN c\_employees%NOTFOUND;**

**DBMS\_OUTPUT.PUT\_LINE(**

**v\_employee.employee\_id || ', ' ||**

**v\_employee.first\_name || ' ' || v\_employee.last\_name || ', ' ||**

**v\_employee.department || ', ' || v\_employee.salary**

**);**

**END LOOP;**

**CLOSE c\_employees;**

**EXCEPTION**

**WHEN OTHERS THEN**

**DBMS\_OUTPUT.PUT\_LINE('An error occurred.');**

**END;**

**/**

1. Write a PL/SQL program that will calculate and display the average salary of all employees.

Solution:

**-- Create the "employee" table**

**CREATE TABLE employee (**

**employee\_id NUMBER PRIMARY KEY,**

**first\_name VARCHAR2(50),**

**last\_name VARCHAR2(50),**

**department VARCHAR2(50),**

**salary NUMBER**

**);**

**-- Insert sample records into the "employee" table**

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

**VALUES (1, 'John', 'Doe', 'HR', 50000);**

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

**VALUES (2, 'Jane', 'Smith', 'Finance', 60000);**

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

**VALUES (3, 'Michael', 'Johnson', 'IT', 70000);**

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

**VALUES (4, 'Merry', 'Agarwal', 'IT', 50000);**

**CREATE OR REPLACE PROCEDURE calculate\_average\_salary AS**

**total\_salary NUMBER := 0;**

**employee\_count NUMBER := 0;**

**CURSOR c\_employees IS**

**SELECT salary FROM employee;**

**v\_salary employee.salary%TYPE;**

**BEGIN**

**OPEN c\_employees;**

**LOOP**

**FETCH c\_employees INTO v\_salary;**

**EXIT WHEN c\_employees%NOTFOUND;**

**total\_salary := total\_salary + v\_salary;**

**employee\_count := employee\_count + 1;**

**END LOOP;**

**CLOSE c\_employees;**

**IF employee\_count > 0 THEN**

**DBMS\_OUTPUT.PUT\_LINE('Average Salary: ' || (total\_salary / employee\_count));**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('No employees found.');**

**END IF;**

**EXCEPTION**

**WHEN OTHERS THEN**

**DBMS\_OUTPUT.PUT\_LINE('An error occurred.');**

**END;**

**/**

1. Write a PL/SQL program that Create Trigger to enforce a minimum salary requirement while inserting record in employee table.

This trigger will prevent any new or updated records from being added to the "employee" table if the salary is less than 30,000.

It uses the RAISE\_APPLICATION\_ERROR procedure to raise a custom error when the condition is not met.

Solution:

**-- Create the "employee" table**

**CREATE TABLE employee (**

**employee\_id NUMBER PRIMARY KEY,**

**first\_name VARCHAR2(50),**

**last\_name VARCHAR2(50),**

**department VARCHAR2(50),**

**salary NUMBER**

**);**

**-- Insert sample records into the "employee" table**

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

**VALUES (1, 'John', 'Doe', 'HR', 50000);**

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

**VALUES (2, 'Jane', 'Smith', 'Finance', 60000);**

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

**VALUES (3, 'Michael', 'Johnson', 'IT', 70000);**

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

**VALUES (4, 'Merry', 'Agarwal', 'IT', 20000);**

**CREATE OR REPLACE TRIGGER enforce\_min\_salary**

**BEFORE INSERT OR UPDATE ON employee**

**FOR EACH ROW**

**BEGIN**

**IF :NEW.salary < 30000 THEN**

**RAISE\_APPLICATION\_ERROR(-20001, 'Salary must be at least 30,000.');**

**END IF;**

**END;**

**/**

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

**VALUES (5, 'Marvin', 'Miski', 'HR', 10000);**