**Roll No…………….. Total No. of Pages:……**

**ST-2 (SET-VI)**

**4th SEMESTER 2023-24**

**22CS007- Database Management System**

**Time allowed: 90 Minutes Max. Marks: 40**

**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 is used to process a set of rows returned by a query?
   1. Stored Procedure
   2. Trigger
   3. Function
   4. **Cursor**
2. An "instead of" trigger in DBMS is commonly used with which type of database object?
   1. Tables
   2. **Views**
   3. Indexes
   4. Constraints
3. Which of the following statements is true regarding triggers in DBMS?
   1. Triggers can only be written in SQL.
   2. Triggers are executed automatically and cannot be manually invoked.
   3. **Triggers can be used to enforce complex business rules and data validation.**
   4. Triggers are limited to performing basic data retrieval operations.
4. What is a package body in DBMS?
   1. The definition of the package's interface
   2. **The implementation of the package's procedures and functions**
   3. The documentation of the package's functionality
   4. The metadata associated with the package
5. In the context of recoverability, the term "redo" refers to:
   1. Repeating the operations of a transaction
   2. Rolling back a transaction
   3. **Writing modified data to disk**
   4. Locking data items for exclusive access
6. Which of the following is a technique used to ensure atomicity and durability of transactions?
   1. Checkpoints
   2. Locking
   3. **Logging**
   4. Caching
7. Which parameter mode is used when the procedure needs to receive data from the calling program?
   1. **IN**
   2. OUT
   3. INOUT
   4. REF
8. Which parameter mode is used when the procedure needs to send data back to the calling program?
   1. IN
   2. **OUT**
   3. INOUT
   4. REF
9. Any subprogram not in the package specification but coded in the package body is called a \_\_\_\_\_\_\_\_\_ object.
   1. protected
   2. **private**
   3. self
   4. public
10. The CREATE TRIGGER statement is used to create the trigger. THE \_\_\_\_\_ clause specifies the table name on which the trigger is to be attached. The \_\_\_\_\_\_ specifies that this is an AFTER INSERT trigger.
    1. for insert, on
    2. **On, for insert**
    3. For, insert
    4. None of the mentioned

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

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

1. A \_\_\_\_ statement cannot be used to change the trigger definition if the OR REPLACE parameter is specified.
   1. **DROP TRIGGER**
   2. ADD TRIGGER
   3. REPLACE TRIGGER
   4. None
2. Which of the following cannot be overwritten by procedure?
   1. **IN**
   2. OUT
   3. INOUT
   4. None of the above
3. What is the syntax of ENABLE Trigger?
4. **ALTER TRIGGER trigger\_name ENABLE;**
5. ALTER TRIGGER trigger\_name ALTER;
6. ENABLE TRIGGER trigger\_name ENABLE;
7. ENABLE TRIGGER trigger\_name ALTER;
8. What is the difference between PL/SQL Function and PL/SQL Procedure?
9. **PL/SQL Procedure may or may not return the value whereas PL/SQL Function must have to return the value.**
10. PL/SQL function may or may not return the value whereas PL/SQL Procedure must have to return the value.
11. PL/SQL Function may or may not return the function whereas PL/SQL Procedure must have to return the function.
12. None of the above
13. The correct syntax to declare PL/SQL variable is\_\_\_
14. **variable\_name [CONSTANT] datatype [NOT NULL] [:= | DEFAULT initial\_value]**
15. variable\_name [CONSTANT] datatype [NULL] [:= | DEFAULT initial\_value]
16. datatype [CONSTANT] variable\_name [NULL] [:= | DEFAULT initial\_value]
17. datatype [CONSTANT] variable\_name [NOT NULL] [:= | DEFAULT initial\_value]

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

1. Write a PL/SQL function calculates the differences between total areas of circles with a number of radii.

Solution:

**declare**

**v\_pi\_nr NUMBER:=3.14;**

**function f\_getDiff\_Nr(i\_rad1\_nr NUMBER,i\_rad2\_nr NUMBER)**

**return NUMBER is**

**v\_area1\_nr NUMBER;**

**v\_area2\_nr NUMBER;**

**v\_out\_nr NUMBER;**

**function f\_getArea\_Nr (i\_rad\_nr NUMBER)**

**return NUMBER**

**is**

**begin**

**return v\_pi\_nr\*(i\_rad\_nr\*\*2);**

**end;**

**begin**

**v\_area1\_nr := f\_getArea\_Nr (i\_rad1\_nr);**

**v\_area2\_nr := f\_getArea\_Nr (i\_rad2\_nr);**

**v\_out\_nr :=v\_area1\_nr-v\_area2\_nr;**

**return v\_out\_nr;**

**end;**

**begin**

**DBMS\_OUTPUT.put\_line('Diff between 3 and 4: '||f\_getDiff\_Nr(4,3));**

**DBMS\_OUTPUT.put\_line('Diff between 4 and 5: '||f\_getDiff\_Nr(5,4));**

**DBMS\_OUTPUT.put\_line('Diff between 5 and 6: '||f\_getDiff\_Nr(6,5));**

**end;**

**/**

1. Write a PL/SQL program that Create Trigger to automatically generate an employee ID using a sequence.

Solution:

**CREATE SEQUENCE seq\_employee\_id START WITH 1000 INCREMENT BY 1;**

**CREATE OR REPLACE TRIGGER trg\_generate\_employee\_id**

**BEFORE INSERT ON employee**

**FOR EACH ROW**

**BEGIN**

**IF :new.employee\_id IS NULL THEN**

**:new.employee\_id := seq\_employee\_id.NEXTVAL;**

**END IF;**

**END;**

**/**

1. Write a PL/SQL program to create Package with a function to get the total salary of all employees in 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', 20000);**

**CREATE OR REPLACE PACKAGE employee\_mgmt AS**

**FUNCTION get\_department\_salary(department\_name VARCHAR2) RETURN NUMBER;**

**END employee\_mgmt;**

**/**

**CREATE OR REPLACE PACKAGE BODY employee\_mgmt AS**

**FUNCTION get\_department\_salary(department\_name VARCHAR2) RETURN NUMBER IS**

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

**BEGIN**

**SELECT SUM(salary) INTO total\_salary**

**FROM employee**

**WHERE department = department\_name;**

**RETURN total\_salary;**

**END;**

**END employee\_mgmt;**

**/**

1. Write a PL/SQL program that delete an employee based on their employee\_id

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 delete\_employee\_by\_id(p\_employee\_id NUMBER) AS**

**BEGIN**

**DELETE FROM employee**

**WHERE employee\_id = p\_employee\_id;**

**IF SQL%ROWCOUNT > 0 THEN**

**DBMS\_OUTPUT.PUT\_LINE('Employee deleted successfully.');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('Employee ID not found. No employee deleted.');**

**END IF;**

**EXCEPTION**

**WHEN OTHERS THEN**

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

**END;**

**/**