**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. What is the key difference between stored procedures and functions in DBMS?
   1. **Stored procedures can return multiple values, while functions can only return a single value**
   2. Stored procedures can be executed by users, while functions can only be executed by the database administrator
   3. Stored procedures are used for data manipulation, while functions are used for data retrieval
   4. Stored procedures can be called from within other procedures, while functions cannot
2. Which of the following types of procedures in DBMS is used to maintain data integrity?
   1. Stored Procedure
   2. **Trigger**
   3. Function
   4. Cursor
3. In DBMS, an insert trigger is fired when:
   1. **A new record is inserted into a table**
   2. An existing record is updated
   3. A record is deleted from a table
   4. A query is executed
4. A delete trigger in DBMS is executed when:
   1. A new record is inserted into a table
   2. An existing record is updated
   3. **A record is deleted from a table**
   4. A query is executed
5. Packages in DBMS help in achieving which of the following?
   1. **Code reusability and modularity**
   2. Data normalization and indexing
   3. Database replication and synchronization
   4. Query optimization and performance tuning
6. Which of the following is a feature of packages in DBMS?
   1. Packages are not reusable across different database schemas
   2. Packages can only contain SQL statements and no procedural code
   3. **Packages allow private variables and procedures inaccessible from outside the package**
   4. Packages cannot be compiled or executed independently
7. Which statement is true about packages in DBMS?
   1. Packages cannot be used to encapsulate SQL statements
   2. Packages cannot be stored in the database
   3. **Packages provide a way to organize and encapsulate related database objects**
   4. Packages are only used for data modeling purposes
8. Which of the following is an advantage of using packages in DBMS?
   1. **Reduced code redundancy and improved maintainability**
   2. Faster query execution times
   3. Increased database security
   4. Enhanced data compression techniques
9. Which of the following isolation levels guarantees that a transaction sees only committed data and ignores uncommitted changes made by other transactions?
   1. Read Uncommitted
   2. **Read Committed**
   3. Repeatable Read
   4. Serializable
10. A transaction that reads data before any other transaction writes to it and holds a shared lock on the data is an example of:
    1. Read-Write conflict
    2. Write-Read conflict
    3. **Read-Read conflict**
    4. Write-Write conflict

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

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

1. Which of the following is mandatory part in procedure?
   1. Declarative Part
   2. Exception-handling
   3. **Executable Part**
   4. All of the above
2. Consider the following code snippet: what will be the output?

DECLARE

a number(2) ;

BEGIN

FOR a IN REVERSE 10 .. 20 LOOP

END LOOP;

dbms\_output.put\_line(a);

END;

1. 11
2. **10**
3. 29
4. 30
5. The pre-defined exception NO\_DATA\_FOUND is raised when
6. A null object is automatically assigned a value.
7. **A SELECT INTO statement returns no rows.**
8. PL/SQL has an internal problem.
9. PL/SQL ran out of memory or memory was corrupted.
10. Which of the following is true about PL/SQL package body?
11. **The package body has the codes for various methods declared in the package specification and other private declarations.**
12. It is created using the CREATE PACKAGE Statement.
13. The codes, methods and types declared in package body are not hidden from code outside the package.
14. All of the above.
15. Savepoints are set to
16. Help in splitting a long transaction into smaller units.
17. Help in rolling back to some checkpoint, within a long transaction.
18. To execute a COMMIT automatically.
19. **Answer a. and b.**

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

1. Write a PL/SQL program that creates a procedure which have parameters passed by reference.

Solution:

**create or replace procedure p\_validate(io\_string\_tx IN OUT NOCOPY VARCHAR2)**

**is**

**v\_invalid\_tx VARCHAR2(8):='test';**

**begin**

**io\_string\_tx:=replace (io\_string\_tx,v\_invalid\_tx);**

**if length(io\_string\_tx)>4000**

**then**

**io\_string\_tx:=substr(io\_string\_tx,1,3997)||'...';**

**end if;**

**end;**

**/**

1. Write a bodiless package to convert temperature from Celsius to Fahrenheit and vice versa.

Solution:

**CREATE OR REPLACE PACKAGE temperature\_conversion\_pkg AS**

**END temperature\_conversion\_pkg;**

**/**

**CREATE OR REPLACE PACKAGE BODY temperature\_conversion\_pkg AS**

**FUNCTION celsius\_to\_fahrenheit(celsius NUMBER) RETURN NUMBER IS**

**fahrenheit NUMBER := (celsius \* 9/5) + 32;**

**BEGIN**

**RETURN fahrenheit;**

**END celsius\_to\_fahrenheit;**

**FUNCTION fahrenheit\_to\_celsius(fahrenheit NUMBER) RETURN NUMBER IS**

**celsius NUMBER := (fahrenheit - 32) \* 5/9;**

**BEGIN**

**RETURN celsius;**

**END fahrenheit\_to\_celsius;**

**PROCEDURE display\_conversions(celsius NUMBER, fahrenheit NUMBER) IS**

**f\_temp NUMBER;**

**c\_temp NUMBER;**

**BEGIN**

**f\_temp := celsius\_to\_fahrenheit(celsius);**

**c\_temp := fahrenheit\_to\_celsius(fahrenheit);**

**DBMS\_OUTPUT.PUT\_LINE(celsius || ' degrees Celsius is ' || f\_temp || ' degrees Fahrenheit.');**

**DBMS\_OUTPUT.PUT\_LINE(fahrenheit || ' degrees Fahrenheit is ' || c\_temp || ' degrees Celsius.');**

**END display\_conversions;**

**END temperature\_conversion\_pkg;**

**/**

1. Write a PL/SQL program that Create Trigger to update the salary automatically when a new employee is inserted.

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 trg\_update\_salary**

**BEFORE INSERT ON employee**

**FOR EACH ROW**

**BEGIN**

**IF :new.salary IS NULL THEN**

**:new.salary := 50000; -- Default salary when not provided**

**END IF;**

**END;**

**/**

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

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

1. Write a PL/SQL program that update an employee's salary 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 update\_employee\_salary(**

**p\_employee\_id NUMBER,**

**p\_new\_salary NUMBER**

**) AS**

**BEGIN**

**UPDATE employee**

**SET salary = p\_new\_salary**

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

**IF SQL%ROWCOUNT > 0 THEN**

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

**ELSE**

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

**END IF;**

**EXCEPTION**

**WHEN OTHERS THEN**

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

**END;**

**/**