**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. In a serializable schedule, which of the following is true?
   1. **The schedule is conflict-serializable.**
   2. The schedule is deadlock-free.
   3. The schedule allows dirty reads.
   4. The schedule allows non-repeatable reads.
2. The ANSI/ISO standard SQL isolation level that provides the highest level of data consistency and isolation is:
   1. READ UNCOMMITTED
   2. READ COMMITTED
   3. REPEATABLE READ
   4. **SERIALIZABLE**
3. In a DBMS, a view can be used to:
   1. Create a backup of a table
   2. **Restrict access to certain columns of a table**
   3. Modify the structure of a table
   4. Rename a table
4. Which of the following is true about a "REPEAT" loop in programming?
   1. **The loop body is executed at least once, even if the condition is false initially**
   2. The loop body is executed only if the condition is true
   3. The loop body is executed until the condition becomes true
   4. The loop body is executed a fixed number of times
5. In a DBMS, a cursor can be used to navigate through:
   1. **Rows of a table**
   2. Columns of a table
   3. Indexes of a table
   4. Triggers of a table
6. The "DO-WHILE" loop in programming is an example of a:
   1. Pre-tested loop
   2. **Post-tested loop**
   3. Indefinite loop
   4. Nested loop
7. Which of the following is an example of a non-repeatable read?
   1. Transaction T1 reads a value X, then transaction T2 updates X, and T1 reads X again.
   2. Transaction T1 reads a value X, then transaction T2 reads X, and T1 reads X again.
   3. **Transaction T1 reads a value X, then transaction T2 updates X, and T1 reads a different value Y.**
   4. Transaction T1 reads a value X, then transaction T2 reads X, and T1 reads a different value Y.
8. The "write skew" anomaly is associated with which isolation level?
   1. Read Uncommitted
   2. Read Committed
   3. Repeatable Read
   4. **Serializable**
9. Which recovery technique requires the use of compensation transactions to undo the changes made by a failed transaction?
   1. **Undo logging**
   2. Redo logging
   3. Deferred update
   4. Immediate update
10. Which of the following recovery techniques is based on maintaining multiple copies of the database at different points in time?
    1. Undo logging
    2. Redo logging
    3. Deferred update
    4. **Replication**

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

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

1. What is the output of the following program?

DECLARE

a NUMBER := 10;

b NUMBER := 5;

BEGIN

IF a > b THEN

b := a;

ELSIF a < b THEN

a := b;

END IF;

DBMS\_OUTPUT.PUT\_LINE(a || ' ' || b);

END;

1. **10 5**
2. 5 5
3. 10 10
4. 5 10

12) What happens if a cursor is not explicitly closed after use in PL/SQL?

1. The cursor is automatically closed by PL/SQL.
2. **The cursor remains open and consumes memory resources until the session ends.**
3. The cursor is deallocated by the database server.
4. An error is raised if the cursor is not closed.

13) In PL/SQL, can you update data through a view if the view is based on multiple tables?

1. Yes, you can update data through a view based on multiple tables.
2. **No, views based on multiple tables are read-only and cannot be updated.**
3. Views based on multiple tables can only be updated with the INSTEAD OF trigger.
4. It depends on the complexity of the underlying tables; some views may allow updates.

14) Consider the following procedure declaration:

CREATE OR REPLACE PACKAGE my\_package AS

PROCEDURE proc1(p\_emp\_id NUMBER);

END;

/

Which keyword must be used to invoke the procedure 'proc1' from outside the package?

1. EXECUTE
2. **CALL**
3. BEGIN
4. INVOKER

15) CREATE OR REPLACE TRIGGER update\_inventory

AFTER INSERT OR UPDATE ON products

FOR EACH ROW

BEGIN

UPDATE inventory

SET quantity = quantity - :NEW.quantity

WHERE product\_id = :NEW.product\_id;

END;

What is the trigger's purpose?

1. **To update the quantity of products in the inventory table after they are inserted or updated.**
2. To insert new records into the inventory table after products are inserted or updated.
3. To delete records from the inventory table after products are inserted or updated.
4. To insert new records into the products table after an update or insert in the inventory table.

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

16) Write a PL/SQL program that takes a student's score as input and prints the corresponding grade according to the following grading system:

90 and above: A

80 to 89: B

70 to 79: C

60 to 69: D

Below 60: F

Solution:

**DECLARE**

**score NUMBER;**

**BEGIN**

**score := &score\_input;**

**IF score >= 90 THEN**

**DBMS\_OUTPUT.PUT\_LINE('Grade: A');**

**ELSIF score >= 80 THEN**

**DBMS\_OUTPUT.PUT\_LINE('Grade: B');**

**ELSIF score >= 70 THEN**

**DBMS\_OUTPUT.PUT\_LINE('Grade: C');**

**ELSIF score >= 60 THEN**

**DBMS\_OUTPUT.PUT\_LINE('Grade: D');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('Grade: F');**

**END IF;**

**END;**

17) Create a PL/SQL procedure that calculates the grade for a student based on the given marks.

Solution:

**CREATE OR REPLACE PROCEDURE calculate\_grade(marks IN NUMBER) IS**

**grade CHAR(1);**

**BEGIN**

**IF marks >= 90 THEN**

**grade := 'A';**

**ELSIF marks >= 80 THEN**

**grade := 'B';**

**ELSIF marks >= 70 THEN**

**grade := 'C';**

**ELSIF marks >= 60 THEN**

**grade := 'D';**

**ELSE**

**grade := 'F';**

**END IF;**

**DBMS\_OUTPUT.PUT\_LINE('Grade: ' || grade);**

**END;**

**/**

18) Create a PL/SQL Package to Manage Employee Information (consider employee table with proper attributes)

Solution:

**CREATE OR REPLACE PACKAGE Employee\_Info\_Pkg AS**

**PROCEDURE Add\_Employee(emp\_id NUMBER, emp\_name VARCHAR2, emp\_salary NUMBER);**

**FUNCTION Get\_Employee\_Count RETURN NUMBER;**

**END Employee\_Info\_Pkg;**

**/**

**CREATE OR REPLACE PACKAGE BODY Employee\_Info\_Pkg AS**

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

**PROCEDURE Add\_Employee(emp\_id NUMBER, emp\_name VARCHAR2, emp\_salary NUMBER) IS**

**BEGIN**

**INSERT INTO employees (employee\_id, employee\_name, salary)**

**VALUES (emp\_id, emp\_name, emp\_salary);**

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

**END;**

**FUNCTION Get\_Employee\_Count RETURN NUMBER IS**

**BEGIN**

**RETURN employee\_count;**

**END;**

**END Employee\_Info\_Pkg;**

**/**

19) Create a trigger that restricts updating the "salary" column of the "employees" table to a maximum of 10% increase from the current salary.

Solution:

**CREATE OR REPLACE TRIGGER tr\_limit\_salary\_increase**

**BEFORE UPDATE OF salary ON employees**

**FOR EACH ROW**

**BEGIN**

**IF :NEW.salary > (:OLD.salary \* 1.1) THEN**

**raise\_application\_error(-20001, 'Salary increase cannot exceed 10%.');**

**END IF;**

**END;**

**/**