**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. A view is actually a?
2. **composition of a table**
3. decomposition of a table
4. associated to a table
5. None of the above
6. What is true about view?
7. Database views are created using the CREATE VIEW statement.
8. To create a view, a user must have the appropriate system privilege according to the specific implementation.
9. **Both A and B are true**
10. Both A and B are false
11. Which of the following is true about recursive triggers?
    1. They are triggered by other triggers
    2. They can only be fired once per event
    3. **They can cause an infinite loop if not handled properly**
    4. They are not supported in DBMS
12. Which keyword is used to define an exception handler in PL/SQL?
    1. **EXCEPTION**
    2. CATCH
    3. TRY
    4. HANDLE
13. Which recovery technique is suitable for databases with large amounts of write-intensive operations?
    1. Undo logging
    2. **Redo logging**
    3. Deferred update
    4. Shadow paging
14. Which of the following recovery techniques provides the best performance for read-intensive workloads?
    1. Undo logging
    2. Redo logging
    3. Deferred update
    4. **Shadow paging**
15. In a two-phase locking protocol, when are locks released?
    1. After the transaction is committed
    2. After the transaction is aborted
    3. In the growing phase
    4. **In the shrinking phase**
16. Which of the following is NOT a benefit of using locking for concurrency control?
    1. Deadlock prevention
    2. Data consistency
    3. Serializable execution
    4. **High performance**
17. Which of the following is true about compound triggers?
    1. They can only be defined for tables, not views
    2. **They are fired once for each row affected by the triggering event**
    3. They cannot contain any SQL statements
    4. They are not supported in DBMS
18. What is the purpose of the SAVEPOINT statement in DBMS?
    1. To define the start of a transaction
    2. To create a temporary table
    3. **To define a point within a transaction to which you can roll back**
    4. To release a lock on a database object.

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

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

11) What is the output of the following program?

DECLARE

y NUMBER := 7;

BEGIN

CASE y

WHEN 1 THEN

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

WHEN 5 THEN

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

ELSE

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

END CASE;

END;

* 1. One
  2. Five
  3. **Other**
  4. No output

12) In PL/SQL, which of the following statements accurately describes a view?

1. **A view is a virtual table based on the result of a SELECT query.**
2. A view is a physical table that stores data permanently.
3. A view is a temporary table used for transactional purposes.
4. A view is a table used only for indexing purposes.

13) DECLARE

CURSOR emp\_cursor IS

SELECT emp\_name, emp\_salary FROM employees;

v\_emp\_rec emp\_cursor%ROWTYPE;

BEGIN

OPEN emp\_cursor;

FETCH emp\_cursor INTO v\_emp\_rec;

-- More code here

END;

What is the purpose of the v\_emp\_rec variable in the code above?

1. It stores the cursor fetch status.
2. It stores the current employee name.
3. It stores the current employee salary.
4. **It represents a record with the same structure as the cursor.**

14) How can you remove a package from the database in PL/SQL?

1. **By using the DROP PACKAGE statement**
2. By removing all the procedures and functions from the package body
3. By using the DELETE PACKAGE statement
4. By using the TRUNCATE PACKAGE statement

15) CREATE OR REPLACE TRIGGER log\_changes

AFTER UPDATE ON employees

FOR EACH ROW

BEGIN

IF :OLD.salary <> :NEW.salary THEN

INSERT INTO salary\_log (emp\_id, old\_salary, new\_salary)

VALUES (:OLD.emp\_id, :OLD.salary, :NEW.salary);

END IF;

END;

What is the trigger's purpose?

1. **To log changes made to the employees table when the salary is updated.**
2. To prevent updates to the employees table when the salary is changed.
3. To automatically increase the salary of employees after an update.
4. To update the salary of employees in the salary\_log table.

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

16) Write a PL/SQL program to check if a given number is a prime number or not.

Solution:

**DECLARE**

**num NUMBER;**

**is\_prime BOOLEAN := TRUE;**

**BEGIN**

**num := &num\_input;**

**IF num <= 1 THEN**

**is\_prime := FALSE;**

**ELSE**

**FOR i IN 2..TRUNC(SQRT(num)) LOOP**

**IF num MOD i = 0 THEN**

**is\_prime := FALSE;**

**EXIT;**

**END IF;**

**END LOOP;**

**END IF;**

**IF is\_prime THEN**

**DBMS\_OUTPUT.PUT\_LINE(num || ' is a prime number.');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE(num || ' is not a prime number.');**

**END IF;**

**END;**

17) Create a PL/SQL procedure that accepts a string as input and displays its reverse.

Solution:

**CREATE OR REPLACE PROCEDURE reverse\_string(input\_str IN VARCHAR2) IS**

**reversed\_str VARCHAR2(100);**

**BEGIN**

**FOR i IN REVERSE 1..LENGTH(input\_str) LOOP**

**reversed\_str := reversed\_str || SUBSTR(input\_str, i, 1);**

**END LOOP;**

**DBMS\_OUTPUT.PUT\_LINE('Original: ' || input\_str);**

**DBMS\_OUTPUT.PUT\_LINE('Reversed: ' || reversed\_str);**

**END;**

**/**

18) Create a PL/SQL Package to Manage Student Grades (Assume appropriate tables in database)

Solution:

**CREATE OR REPLACE PACKAGE Student\_Grades\_Pkg AS**

**TYPE grade\_rec IS RECORD (subject VARCHAR2(50), grade CHAR(1));**

**TYPE grade\_table IS TABLE OF grade\_rec;**

**FUNCTION Get\_Student\_Grades(student\_id NUMBER) RETURN grade\_table;**

**PROCEDURE Add\_Student\_Grade(student\_id NUMBER, subject VARCHAR2, grade CHAR);**

**END Student\_Grades\_Pkg;**

**/**

**CREATE OR REPLACE PACKAGE BODY Student\_Grades\_Pkg AS**

**student\_grades grade\_table;**

**FUNCTION Get\_Student\_Grades(student\_id NUMBER) RETURN grade\_table IS**

**grades grade\_table := grade\_table();**

**BEGIN**

**FOR i IN 1..student\_grades.COUNT LOOP**

**IF student\_grades(i).student\_id = student\_id THEN**

**grades.EXTEND;**

**grades(grades.LAST) := student\_grades(i);**

**END IF;**

**END LOOP;**

**RETURN grades;**

**END;**

**PROCEDURE Add\_Student\_Grade(student\_id NUMBER, subject VARCHAR2, grade CHAR) IS**

**rec grade\_rec;**

**BEGIN**

**rec.student\_id := student\_id;**

**rec.subject := subject;**

**rec.grade := grade;**

**student\_grades.EXTEND;**

**student\_grades(student\_grades.LAST) := rec;**

**END;**

**END Student\_Grades\_Pkg;**

**/**

19) Create a trigger that prevents the deletion of a department from the "departments" table if there are any employees belonging to that department.

Solution:

**CREATE OR REPLACE TRIGGER tr\_prevent\_dept\_deletion**

**BEFORE DELETE ON departments**

**FOR EACH ROW**

**DECLARE**

**dept\_count NUMBER;**

**BEGIN**

**SELECT COUNT(\*) INTO dept\_count**

**FROM employees**

**WHERE department\_id = :OLD.department\_id;**

**IF dept\_count > 0 THEN**

**raise\_application\_error(-20002, 'Cannot delete department with active employees.');**

**END IF;**

**END;**

**/**