**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 of the following statements is true about procedures in DBMS?
   1. **Procedures are used to perform complex calculations**
   2. Procedures are used to store data permanently
   3. Procedures are used to define database schema
   4. Procedures are used to create user accounts
2. Which of the following is not a type of procedure in DBMS?
   1. Stored Procedure
   2. Function
   3. Trigger
   4. **Index**
3. Which of the following best describes a database trigger?
   1. A program that executes a sequence of SQL statements
   2. A constraint that ensures data integrity
   3. **A mechanism to automatically perform actions in response to specific database events**
   4. A function that retrieves data from the database
4. Triggers in DBMS are associated with which of the following?
   1. **Tables**
   2. Views
   3. Indexes
   4. Constraints
5. What is a package in DBMS?
   1. A collection of related tables
   2. **A collection of related procedures and functions**
   3. A collection of related views
   4. A collection of related triggers
6. In a DBMS, a transaction is:
   1. A single SQL statement
   2. A sequence of SQL statements
   3. An individual user operation
   4. **A unit of work that is either executed completely or not at all**
7. ACID properties of a transaction include:
   1. **Atomicity, Consistency, Isolation, Durability**
   2. Attainability, Concurrency, Integration, Durability
   3. Association, Concurrency, Isolation, Durability
   4. Atomicity, Consistency, Isolation, Dependency
8. Which of the following is a purpose of locking in a DBMS?
   1. **Ensuring isolation**
   2. Ensuring durability
   3. Ensuring atomicity
   4. Ensuring consistency
9. In a DBMS, which type of lock allows exclusive access to a data item?
   1. Shared lock
   2. **Exclusive lock**
   3. Intent lock
   4. Deadlock
10. In a DBMS, a transaction is said to be serializable if:
    1. It can be divided into multiple sub-transactions
    2. It can execute in parallel with other transactions
    3. **Its execution is equivalent to some serial execution of the transactions**
    4. It guarantees the durability of data

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

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

1. What will be the output of the following code snippet?

DECLARE

a number(3) := 100;

b number(3) := 200;

BEGIN

IF( a = 100 ) THEN

IF( b <> 200 ) THEN

dbms\_output.put\_line(b);

END IF;

END IF;

dbms\_output.put\_line(a);

END;

* 1. It has syntax error, so there will not be any output.
  2. 200
  3. 200
  4. **100**

1. A standalone procedure is deleted with the \_\_\_\_\_\_\_\_ statement
2. REMOVE PROCEDURE
3. DELETE PROCEDURE
4. Both A and B
5. **DROP PROCEDURE**
6. Which of the following is the FIRING POINT: BEFORE?
7. BEFORE INSERT TRIGGER
8. BEFORE UPDATE TRIGGER
9. BEFORE DELETE TRIGGER
10. **All of the above**
11. How STANDARD Package Defines the PL/SQL Environment?
12. **The package specification declares public types, variables, subprograms**
13. The package specification declares private
14. A & B
15. None of the above
16. A \_\_\_\_\_\_\_\_\_\_ is a special kind of a store procedure that executes in response to certain actions.
    1. Procedures
    2. **Triggers**
    3. Functions
    4. None of the mentioned

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

1. Write a PL/SQL program that creates a stored procedure in the database.

Solution:

**declare**

**procedure p\_hello**

**is**

**my\_val varchar2(256):='Hello, World!';**

**begin**

**dbms\_output.put\_line(my\_val);**

**end;**

**begin**

**p\_hello;**

**end;**

1. Create a procedure that calculates and displays the total salary of employees in a given department. The department name is an optional parameter with a default value of 'HR'.

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 PROCEDURE total\_salary\_by\_department(p\_department\_name IN VARCHAR2 DEFAULT 'HR') AS**

**v\_total\_salary NUMBER;**

**BEGIN**

**SELECT SUM(salary) INTO v\_total\_salary**

**FROM employee**

**WHERE department = p\_department\_name;**

**DBMS\_OUTPUT.PUT\_LINE('Total Salary for Department ' || p\_department\_name || ': ' || v\_total\_salary);**

**EXCEPTION**

**WHEN OTHERS THEN**

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

**END;**

**/**

1. Write a bodiless package to generate the first n Fibonacci numbers and display them.

Solution:

**CREATE OR REPLACE TYPE NUMBER\_ARRAY AS TABLE OF NUMBER;**

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

**END fibonacci\_pkg;**

**/**

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

**FUNCTION generate\_fibonacci(n NUMBER) RETURN NUMBER\_ARRAY IS**

**fib\_list NUMBER\_ARRAY := NUMBER\_ARRAY();**

**BEGIN**

**IF n <= 0 THEN**

**RETURN fib\_list;**

**END IF;**

**fib\_list.EXTEND(n);**

**fib\_list(1) := 0;**

**IF n = 1 THEN**

**RETURN fib\_list;**

**END IF;**

**fib\_list(2) := 1;**

**IF n = 2 THEN**

**RETURN fib\_list;**

**END IF;**

**FOR i IN 3..n LOOP**

**fib\_list(i) := fib\_list(i - 1) + fib\_list(i - 2);**

**END LOOP;**

**RETURN fib\_list;**

**END generate\_fibonacci;**

**PROCEDURE display\_fibonacci(n NUMBER) IS**

**fib\_list NUMBER\_ARRAY;**

**BEGIN**

**fib\_list := generate\_fibonacci(n);**

**DBMS\_OUTPUT.PUT\_LINE('Fibonacci numbers:');**

**FOR i IN 1..fib\_list.COUNT LOOP**

**DBMS\_OUTPUT.PUT\_LINE(fib\_list(i));**

**END LOOP;**

**END display\_fibonacci;**

**END fibonacci\_pkg;**

**/**

1. Write a PL/SQL program that create Trigger to update the "salary" of an employee to 80000 if the "department" is changed to 'Management'.

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\_department\_update**

**BEFORE UPDATE OF department ON employee**

**FOR EACH ROW**

**BEGIN**

**IF :NEW.department = 'Management' THEN**

**:NEW.salary := 80000;**

**END IF;**

**END;**

**/**