1512N998

Candidate's Seat No:____

IMSC IT SD Sem.-9 Examination

SMD 902

Database Administrator-III

Time : 2.30 Hours] **December-2025** **[Max.Marks : 70**

Instructions:
1. **Attempt all questions.**
2. **Make suitable assumptions wherever necessary.**
3. **Figures to the right indicate full marks.**

1. Answer these questions in short.(Any 5) **(10 Marks)**

a. What SQL command creates a new database in PostgreSQL?
b. Name three common constraints for PostgreSQL tables.
c. What is the difference between UNION and INTERSECT set operations?
d. What does the ROLLBACK command do in a PostgreSQL transaction?
e. What are two types of indexes in PostgreSQL.
f. What is the primary purpose of the VACUUM command?
g. What Node.js module is commonly used to connect to PostgreSQL?

2. Answer these questions briefly.(Any 8) **(32 Marks)**

a. Describe the purpose of the FOREIGN KEY, UNIQUE, and NOT NULL constraints. Give one example used in a real-world table.
b. Describe RDBMS fundamentals and how PostgreSQL fits as a relational database management system.
c. Assume a users table with columns id, name, age, city. Write a SELECT query to fetch names of users aged > 25 from 'NYC', ordered by age DESC, limited to 5 results.
d. Describe the three types of table relationships (One-to-One, One-to-Many, Many-to-Many) with real-world examples.
e. Using employees (emp_id, dept_id, salary) table, write a query to find average salary per department using GROUP BY, filtering for avg > 50000 with HAVING.
f. Write a CTE (using WITH) to find employees earning more than their department average from employees (emp_id, dept_id, salary).
g. Using products (id, category, price) and orders (id, product_id, qty) tables, write a subquery to select product names with total qty > 10.
h. Describe control flow structures like IF-THEN-ELSE and LOOP in PL/pgSQL with syntax snippets.
i. Write an Express.js route for INSERT into posts table using parameterized query to add title and content.
j. Explain the use of CHECK constraints with an example..

P.T.O

IV998-2

3. Answer these questions in detail.(Any 2) **(28 Marks)**

a. Explain Aggregate Functions in PostgreSQL. Describe the following with syntax and examples:
COUNT()
SUM()
AVG()
MAX()
MIN()
STRING_AGG()
ARRAY_AGG()

b. CREATE TABLE products (product_id SERIAL PRIMARY KEY,product_name VARCHAR(100),stock INTEGER DEFAULT 0);

CREATE TABLE orders (order_id SERIAL PRIMARY KEY,product_id INTEGER REFERENCES products(product_id),quantity INTEGER,order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP);

CREATE TABLE inventory_log (log_id SERIAL PRIMARY KEY,product_id INTEGER,old_stock INTEGER,new_stock INTEGER,log_reason VARCHAR(50),log_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP);

```
-- FUNCTION: Place order and update inventory
CREATE OR REPLACE FUNCTION place_single_order(p_product_id
INTEGER, p_quantity INTEGER)
RETURNS VARCHAR AS $$
DECLARE
   v_current_stock INTEGER;
BEGIN
-- 1: Fetch current stock of the product into v_current_stock
```

-- 2: Check if the stock is less than p_quantity. If yes, return 'Insufficient stock'

-- 3: Insert the new order into the orders table

-- 4: Update product stock (reduce by p_quantity)

-- 5: Insert inventory log (old_stock, new_stock, reason 'ORDER PLACED')

N9983

```
    RETURN 'Order placed successfully';
END;
$$ LANGUAGE plpgsql;

-- 6: Create a trigger that fires AFTER UPDATE on products (only when
stock changes)
```

_____

c. Complete the code using Node js and PostgreSQL.

```
const express = require('express');
const { Pool } = require('pg');
const app = express();
// 1: Middleware to parse JSON request bodies
```

_____

```
// 2: Create PostgreSQL connection pool with environment variables
//  Use: host, user, database, password, port from process.env
const pool = new Pool({
```

_____

```
});
// GET /products - Read all products
app.get('/products', async (req, res) => {
    try {
        // 3: Query to select all products from products table
        const result = await pool.query(_____);
        res.json({success: true,data: result.rows});
    } catch (error) {
        console.error('Error fetching products:', error);
    }
});

// GET /products/:id - Read single product by ID
app.get('/products/:id', async (req, res) => {
    const id = req.params.id;

    try {
        // 4: Parameterized query to get product by ID
        const result = await pool.query(_____, [_____]);

        if (result.rows.length === 0) {
            return res.status(404).json({
                success: false,
                message: 'Product not found'
            });
        }
        res.json({success: true,data: result.rows[0]});
    } catch (error) {
        console.error('Error fetching product:', error);
    }
});
```