ADS POE

Create table Employee (FirstName, LastName, EmpId, Sallary) insert values for 10 employees and apply Range and List partitioning techniques.

```
-- Create the Employee table
CREATE TABLE Employee (
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  EmpId INT,
  Salary DECIMAL(10, 2)
);
-- Create range partitions
CREATE TABLE Employee_Range_Partitioned (
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  EmpId INT,
  Salary DECIMAL(10, 2)
) PARTITION BY RANGE (EmpId) (
  PARTITION p1 VALUES LESS THAN (100),
  PARTITION p2 VALUES LESS THAN (200),
  PARTITION p3 VALUES LESS THAN (300),
  PARTITION p4 VALUES LESS THAN (400),
  PARTITION p5 VALUES LESS THAN (500)
);
-- Create list partitions
CREATE TABLE Employee_List_Partitioned (
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  EmpId INT,
  Salary DECIMAL(10, 2)
) PARTITION BY LIST (LastName) (
  PARTITION pA VALUES IN ('Smith', 'Johnson'),
  PARTITION pB VALUES IN ('Williams', 'Brown'),
  PARTITION pC VALUES IN ('Jones', 'Miller'),
  PARTITION pD VALUES IN ('Davis', 'Garcia'),
  PARTITION pE VALUES IN ('Wilson', 'Martinez')
);


-- Insert values into the Employee table
INSERT INTO Employee (FirstName, LastName, EmpId, Salary)
VALUES
  ('John', 'Smith', 101, 5000),
  ('Jane', 'Johnson', 102, 6000),
  ('Robert', 'Williams', 201, 7000),
  ('Michael', 'Brown', 202, 8000),
  ('David', 'Jones', 301, 9000),
  ('Sarah', 'Miller', 302, 10000),
  ('Emily', 'Davis', 401, 11000),
  ('Christopher', 'Garcia', 402, 12000),
  ('Daniel', 'Wilson', 501, 13000),
  ('Jennifer', 'Martinez', 502, 14000);
```

Create table Student (Name, RollNo,City) insert values for 10 Students and apply List and Range partitioning techniques.

```
-- Create the Student table
CREATE TABLE Student (
  Name VARCHAR(50),
  RollNo INT,
  City VARCHAR(50)
);
-- Create list partitions
CREATE TABLE Student_List_Partitioned (
  Name VARCHAR(50),
  RollNo INT,
  City VARCHAR(50)
) PARTITION BY LIST (City) (
  PARTITION pA VALUES IN ('New York', 'Los Angeles'),
  PARTITION pB VALUES IN ('Chicago', 'Houston'),
  PARTITION pC VALUES IN ('San Francisco', 'Seattle'),
  PARTITION pD VALUES IN ('Boston', 'Atlanta'),
  PARTITION pE VALUES IN ('Miami', 'Dallas')
```

```
);
-- Create range partitions
CREATE TABLE Student_Range_Partitioned (
  Name VARCHAR(50),
  RollNo INT,
  City VARCHAR(50)
) PARTITION BY RANGE (RollNo) (
  PARTITION p1 VALUES LESS THAN (100),
  PARTITION p2 VALUES LESS THAN (200),
  PARTITION p3 VALUES LESS THAN (300),
  PARTITION p4 VALUES LESS THAN (400),
  PARTITION p5 VALUES LESS THAN (500)
);

-- Insert values into the Student table
INSERT INTO Student (Name, RollNo, City)
VALUES
  ('John Doe', 101, 'New York'),
  ('Jane Smith', 102, 'Los Angeles'),
  ('Robert Johnson', 201, 'Chicago'),
  ('Michael Brown', 202, 'Houston'),
  ('David Jones', 301, 'San Francisco'),
  ('Sarah Miller', 302, 'Seattle'),
  ('Emily Davis', 401, 'Boston'),
  ('Christopher Garcia', 402, 'Atlanta'),
  ('Daniel Wilson', 501, 'Miami'),
  ('Jennifer Martinez', 502, 'Dallas');
```

Create table Employee (FirstName, LastName, EmpId, Sallary) insert values for 10 employees and apply Hash partitioning technique.

```
-- Create the Employee table
CREATE TABLE Employee (
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  EmpId INT,
  Salary DECIMAL(10, 2)
);
-- Create hash partitions
CREATE TABLE Employee_Hash_Partitioned (
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  EmpId INT,
  Salary DECIMAL(10, 2)
) PARTITION BY HASH (EmpId) PARTITIONS 4;

-- Insert values into the Employee table
INSERT INTO Employee (FirstName, LastName, EmpId, Salary)
VALUES
  ('John', 'Smith', 101, 5000),
  ('Jane', 'Johnson', 102, 6000),
  ('Robert', 'Williams', 201, 7000),
  ('Michael', 'Brown', 202, 8000),
  ('David', 'Jones', 301, 9000),
  ('Sarah', 'Miller', 302, 10000),
  ('Emily', 'Davis', 401, 11000),
  ('Christopher', 'Garcia', 402, 12000),
  ('Daniel', 'Wilson', 501, 13000),
  ('Jennifer', 'Martinez', 502, 14000);

-- Insert values into the Employee_Hash_Partitioned table
INSERT INTO Employee_Hash_Partitioned (FirstName, LastName, EmpId, Salary)
SELECT FirstName, LastName, EmpId, Salary FROM Employee;
```

Create an “employee” table with necessary schema-eid, ename, esal, ecity on one computer. Insert at least 10 rows in it. Using Frontend to Backend connectivity and - server socket programming, perform horizontal fragmentation by giving appropriate query in SQL from client side.

```
-- Create the employee table
CREATE TABLE employee (
  eid INT,
  ename VARCHAR(50),
  esal DECIMAL(10, 2),
  ecity VARCHAR(50)
);
```

```
-- Insert values into the employee table
INSERT INTO employee (eid, ename, esal, ecity)
VALUES
  (1, 'John Doe', 5000.00, 'New York'),
  (2, 'Jane Smith', 6000.00, 'Los Angeles'),
  (3, 'Robert Johnson', 7000.00, 'Chicago'),
  (4, 'Michael Brown', 8000.00, 'Houston'),
  (5, 'David Jones', 9000.00, 'San Francisco'),
  (6, 'Sarah Miller', 10000.00, 'Seattle'),
  (7, 'Emily Davis', 11000.00, 'Boston'),
  (8, 'Christopher Garcia', 12000.00, 'Atlanta'),
  (9, 'Daniel Wilson', 13000.00, 'Miami'),
  (10, 'Jennifer Martinez', 14000.00, 'Dallas');
```

Server-side (Backend)

```
import socket
import sqlite3

# Establish a socket connection
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('localhost', 1234))
server_socket.listen(1)

# Establish a database connection
conn = sqlite3.connect('your_database.db')
cursor = conn.cursor()

# Wait for client connection and process queries
while True:
    client_socket, client_address = server_socket.accept()
    print('Client connected:', client_address)

    # Receive query from the client
    query = client_socket.recv(1024).decode()

    # Perform horizontal fragmentation query based on city
    sql_query = "SELECT * FROM employee WHERE ecity = ?"
    cursor.execute(sql_query, (query,))
    results = cursor.fetchall()

    # Send the query results back to the client
    client_socket.send(str(results).encode())

    # Close the client connection
    client_socket.close()
```

Client-side (Frontend)

```
import socket

# Establish a socket connection
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('localhost', 1234))

# Send the fragmentation query to the server
query = "New York"
client_socket.send(query.encode())

# Receive the query results from the server
results = client_socket.recv(1024).decode()
print(results)

# Close the socket connection
client_socket.close()
```

Create an “employee” table with necessary schema-eid, ename, esal, ecity on one computer. Insert at least 10 rows in it. Using Frontend to Backend connectivity and client- server socket programming, perform Vertical fragmentation by giving appropriate query in SQL from client side.

Server-side (Backend)

```
import socket
import sqlite3

# Establish a socket connection
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
server_socket.bind(('localhost', 1234))
server_socket.listen(1)

# Establish a database connection
conn = sqlite3.connect('your_database.db')
cursor = conn.cursor()

# Wait for client connection and process queries
while True:
    client_socket, client_address = server_socket.accept()
    print('Client connected:', client_address)

    # Receive query from the client
    query = client_socket.recv(1024).decode()

    # Perform vertical fragmentation query
    if query == 'fragment1':
        sql_query = "SELECT eid, ename FROM employee"
    elif query == 'fragment2':
        sql_query = "SELECT esal, ecity FROM employee"
    else:
        sql_query = ""

    cursor.execute(sql_query)
    results = cursor.fetchall()

    # Send the query results back to the client
    client_socket.send(str(results).encode())

    # Close the client connection
    client_socket.close()
```

Client-side (Frontend)

```
import socket

# Establish a socket connection
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('localhost', 1234))

# Send the fragmentation query to the server
query = "fragment1"
client_socket.send(query.encode())

# Receive the query results from the server
results = client_socket.recv(1024).decode()
print(results)

# Close the socket connection
Client_socket.close()
```

Write Java Program to save image into ORDBMS.

```
import java.io.File;
import java.io.FileInputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class SaveImageToORDBMS {
    public static void main(String[] args) {
        String url = "jdbc:postgresql://localhost:5432/your_database"; // Replace with your database URL
        String username = "your_username"; // Replace with your database username
        String password = "your_password"; // Replace with your database password

        String imagePath = "path_to_image.jpg"; // Replace with the path to your image file

        try {
            // Load the PostgreSQL JDBC driver
            Class.forName("org.postgresql.Driver");

            // Establish a connection to the database
            Connection connection = DriverManager.getConnection(url, username, password);

            // Prepare the SQL statement to insert the image
```

```
            String sql = "INSERT INTO images (image_data) VALUES (?)";
            PreparedStatement statement = connection.prepareStatement(sql);

            // Read the image file
            File imageFile = new File(imagePath);
            FileInputStream inputStream = new FileInputStream(imageFile);

            // Set the image as a parameter in the SQL statement
            statement.setBinaryStream(1, inputStream, (int) imageFile.length());

            // Execute the SQL statement to insert the image
            int rowsAffected = statement.executeUpdate();

            if (rowsAffected > 0) {
                System.out.println("Image saved successfully.");
            } else {
                System.out.println("Failed to save the image.");
            }

            // Close the resources
            statement.close();
            connection.close();
            inputStream.close();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

create table locations(locid varchar(5),city varchar(10),state varchar(5),country varchar(10));
create table sales(pid varchar(5),timeid varchar(5),locid varchar(5),sales varchar(5));
create table products(pid varchar(5), pname varchar(10), category varchar(15),price varchar(5));
create table time(timeid varchar(5),year varchar(5));

Perform OLAP operations CUBE and ROLLUP.

```
CREATE TABLE locations (
  locid VARCHAR(5),
  city VARCHAR(10),
  state VARCHAR(5),
  country VARCHAR(10)
);

INSERT INTO locations (locid, city, state, country) VALUES
  ('L1', 'New York', 'NY', 'USA'),
  ('L2', 'London', 'LDN', 'UK'),
  ('L3', 'Paris', 'PRS', 'France'),
  ('L4', 'Tokyo', 'TKY', 'Japan'),
  ('L5', 'Sydney', 'SYD', 'Australia'),
  ('L6', 'Toronto', 'TOR', 'Canada'),
  ('L7', 'Berlin', 'BER', 'Germany'),
  ('L8', 'Rome', 'ROM', 'Italy'),
  ('L9', 'Cairo', 'CAI', 'Egypt'),
  ('L10', 'Rio de Janeiro', 'RIO', 'Brazil');

CREATE TABLE sales (
  pid VARCHAR(5),
  timeid VARCHAR(5),
  locid VARCHAR(5),
  sales VARCHAR(5)
);

INSERT INTO sales (pid, timeid, locid, sales) VALUES
  ('P1', 'T1', 'L1', '100'),
  ('P2', 'T2', 'L2', '200'),
  ('P3', 'T3', 'L3', '150'),
  ('P1', 'T2', 'L1', '300'),
  ('P2', 'T1', 'L2', '250'),
  ('P3', 'T1', 'L3', '180'),
  ('P1', 'T3', 'L1', '120'),
```

```
  ('P2', 'T2', 'L2', '220'),
  ('P3', 'T2', 'L3', '160'),
  ('P1', 'T1', 'L1', '350');

CREATE TABLE products (
  pid VARCHAR(5),
  pname VARCHAR(10),
  category VARCHAR(15),
  price VARCHAR(5)
);

INSERT INTO products (pid, pname, category, price) VALUES
  ('P1', 'Product A', 'Category 1', '10.00'),
  ('P2', 'Product B', 'Category 2', '20.00'),
  ('P3', 'Product C', 'Category 1', '15.00'),
  ('P4', 'Product D', 'Category 2', '30.00'),
  ('P5', 'Product E', 'Category 1', '25.00'),
  ('P6', 'Product F', 'Category 2', '18.00'),
  ('P7', 'Product G', 'Category 1', '12.00'),
  ('P8', 'Product H', 'Category 2', '22.00'),
  ('P9', 'Product I', 'Category 1', '28.00'),
  ('P10', 'Product J', 'Category 2', '15.00');
CREATE TABLE time (
  timeid VARCHAR(5),
  year VARCHAR(5)
);

INSERT INTO time (timeid, year) VALUES
  ('T1', '2020'),
  ('T2', '2021'),
  ('T3', '2022'),
  ('T4', '2023'),
  ('T5', '2024'),
  ('T6', '2025'),
  ('T7', '2026'),
  ('T8', '2027'),
  ('T9', '2028'),
  ('T10', '2029');

CUBE Operation:
SELECT locid, timeid, pid, SUM(sales) AS total_sales
FROM sales
GROUP BY CUBE(locid, timeid, pid);

ROLLUP Operation:
SELECT locid, timeid, pid, SUM(sales) AS total_sales
FROM sales
GROUP BY ROLLUP(locid, timeid, pid);
```

Create structured data types of ORDBMS and perform operations. create table using structured data types, insert data and solve queries.
Create ADDRESS User Defined Type
Create PERSON UDT containing an embedded ADDRESS UDT

```
Create ADDRESS User-Defined Type:
CREATE TYPE ADDRESS AS (
  street VARCHAR(50),
  city VARCHAR(50),
  state VARCHAR(20),
  country VARCHAR(50)
);
Create PERSON UDT containing an embedded ADDRESS UDT:
CREATE TYPE PERSON AS (
  name VARCHAR(50),
  age INT,
  address ADDRESS
);
Create a table using the structured data types:
CREATE TABLE employees (
  employee_id INT,
  employee_info PERSON
);

Insert data into the table:
INSERT INTO employees (employee_id, employee_info)
```

```
VALUES (1, ROW('John Doe', 30, ROW('123 Main St', 'New York', 'NY', 'USA')));

SELECT * FROM employees;
SELECT employee_info.name, employee_info.address.city FROM employees;
SELECT *
FROM employees
WHERE employee_info.address.city = 'New York';
```