Kindly Note

The assignment has 3 questions. Please go through the questions and revert immediately with estimated time you would take and time of submission.

Question 1 : Database Assignment

Problem Statement: "E-commerce Analytics System"

You are tasked with writing complex SQL queries on a database designed for an e-commerce platform that tracks user activities, orders, and product inventory. The platform needs to provide detailed analytics and reports based on this data.

Queries

1. **Lifetime Value of a User**
    - Calculate the lifetime value (LTV) of each user. The LTV is defined as the total amount spent by the user on the platform.
    - Retrieve the top 10 users with the highest LTV.
2. **Product Popularity Over Time**
    - For each product, calculate the total quantity sold per month for the past year.
    - Return the product ID, product name, month, and total quantity sold, sorted by product ID and month.
3. **Customer Retention Rate**
    - Calculate the monthly retention rate for users. The retention rate for a given month is defined as the percentage of users who made a purchase in that month and also made a purchase in the previous month.
    - Return the month and the retention rate for each month in the past year.
4. **Average Order Value and Order Frequency by User Segment**
    - Segment users into three groups based on their total spend: low spenders (bottom 30%), medium spenders (middle 40%), and high spenders (top 30%).
    - For each segment, calculate the average order value (AOV) and the average number of orders per month.
    - Return the segment, AOV, and average number of orders per month.
5. **Conversion Rate of Marketing Campaigns**
    - Assume there is an additional table `MarketingCampaigns` with the columns: `campaign_id`, `campaign_name`, `user_id`, `campaign_start_date`, and `campaign_end_date`.
    - Calculate the conversion rate for each marketing campaign. The conversion rate is defined as the percentage of users who made a purchase during the campaign period.
    - Return the campaign ID, campaign name, and conversion rate.

Sample Tables and Schema

Here is a simplified schema to help you get started:

```
CREATE TABLE Users (
    user_id INT PRIMARY KEY,
    name VARCHAR(255),
    email VARCHAR(255),
    registration_date DATE
);

CREATE TABLE Products (
    product_id INT PRIMARY KEY,
    name VARCHAR(255),
    description TEXT,
    price DECIMAL(10, 2),
    stock_quantity INT
);

CREATE TABLE Orders (
    order_id INT PRIMARY KEY,
    user_id INT,
    order_date DATE,
    total_amount DECIMAL(10, 2),
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
```

```
);

CREATE TABLE OrderItems (
    order_item_id INT PRIMARY KEY,
    order_id INT,
    product_id INT,
    quantity INT,
    price DECIMAL(10, 2),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);

CREATE TABLE UserActivities (
    activity_id INT PRIMARY KEY,
    user_id INT,
    activity_type VARCHAR(255),
    product_id INT,
    activity_timestamp TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);

CREATE TABLE MarketingCampaigns (
```

```
    campaign_id INT PRIMARY KEY,
    campaign_name VARCHAR(255),
    user_id INT,
    campaign_start_date DATE,
    campaign_end_date DATE,
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
```

Submission

- **SQL Queries**: Submit the SQL script containing the queries for the specified tasks.
- **Documentation**: Include a document explaining your design choices, indexing strategies, and any assumptions made during the query process.

Question 2 : Operating Systems and Threading

Problem Statement: "Task Scheduler with Round-Robin Scheduling"

You are tasked with designing and implementing a task scheduler for a server that processes incoming tasks. The server should handle concurrent tasks using threading and manage task execution using a round-robin scheduling algorithm.

Requirements

1. **Task Definition**:
    - Each task has a unique ID, a priority level, and a processing time.
    - Tasks can arrive at any time and need to be processed by the server.
2. **Scheduler Design**:
    - Implement a task queue that stores incoming tasks.
    - Use a round-robin scheduling algorithm to manage the execution of tasks.
    - Ensure that high-priority tasks are executed before lower-priority tasks, but within each priority level, tasks are scheduled in a round-robin fashion.
3. **Threading**:
    - Use threads to simulate concurrent processing of tasks.
    - Ensure thread safety when accessing and modifying the task queue.
    - Implement mechanisms to handle task execution, suspension, and resumption.

4. **Queueing**:
    - Design a queueing system that can handle a high volume of incoming tasks.
    - Ensure that the queueing system is efficient and can handle tasks arriving at different rates.
5. **Task Management**:
    - Implement task management functionalities such as adding new tasks, updating task priorities, and removing completed tasks.
    - Provide real-time monitoring of task statuses and the current state of the task queue.

Detailed Tasks

1. **Task Scheduler Implementation**:
    - Implement a `TaskScheduler` class with methods to add tasks, execute tasks, and manage the task queue.
    - Use a priority queue to store tasks, ensuring that higher-priority tasks are scheduled before lower-priority tasks.
    - Implement round-robin scheduling within each priority level.
2. **Thread Management**:
    - Create a pool of worker threads that execute tasks from the task queue.
    - Ensure that the worker threads can handle task suspension and resumption based on the round-robin scheduling algorithm.
    - Implement thread synchronization to prevent race conditions and ensure data consistency.
3. **Task Queueing System**:
    - Implement a task queue that can efficiently handle a high volume of tasks.
    - Ensure that the queueing system can dynamically adjust to varying rates of incoming tasks.
4. **Monitoring and Reporting**:
    - Implement real-time monitoring of task statuses, including the number of tasks in the queue, tasks being executed, and completed tasks.
    - Provide a mechanism to report the current state of the task scheduler, including task execution times and queue lengths.

Example Scenario

- Tasks arrive at the server with varying priorities and processing times.
- The task scheduler adds tasks to the queue and uses a round-robin algorithm to manage task execution.
- Worker threads execute tasks concurrently, ensuring that high-priority tasks are processed first.
- The system monitors task statuses and provides real-time updates on the state of the task queue.

Guidelines

1. **Correctness**:
    - Ensure that your task scheduler correctly implements round-robin scheduling and handles tasks based on their priority.
2. **Efficiency**:
    - Optimize your queueing and threading mechanisms to handle a high volume of tasks efficiently.
3. **Thread Safety**:
    - Implement proper synchronization mechanisms to ensure thread safety when accessing shared resources.
4. **Real-Time Monitoring**:
    - Provide real-time monitoring and reporting of the task scheduler's status.

Submission

- **Code**: Submit your solution as a Python script or notebook file.
- **Documentation**: Include comments in your code to explain your logic and approach. Provide a brief document explaining your design choices, synchronization strategies, and any assumptions made during the implementation.

Question 3 : System Architecture Assignment

You are required to design a system that processes large volumes of data using GPU instances on Amazon EC2. The system should be able to scale up or down based on the computational load and requirements. The key components of the system include:

1. **Data Ingestion**: Collecting and preprocessing data before sending it to the GPU instances for processing.
2. **Processing Layer**: Utilizing EC2 GPU instances to perform computationally intensive tasks.
3. **Scaling Mechanism**: Automatically scaling the number of GPU instances based on the processing requirements.
4. **Result Storage and Retrieval**: Storing the processed results and making them accessible to the end-users.

Assignment

1. High-Level Architecture Design

Create a high-level architectural diagram that includes the following components:

- Data Ingestion Layer
- Processing Layer with EC2 GPU instances
- Scaling Mechanism
- Result Storage and Retrieval

2. Detailed Design Explanation

Provide a detailed explanation of your design, including:

- **Data Ingestion Layer**:
  - Describe how data will be collected and preprocessed.
  - Explain the technologies and tools you would use for data ingestion (e.g., AWS S3, AWS Kinesis, Kafka).
- **Processing Layer**:
  - Explain how the EC2 GPU instances will be utilized for processing.
  - Describe the type of EC2 GPU instances you would use and why.
  - Discuss how you would distribute the workload among multiple GPU instances.
- **Scaling Mechanism**:
  - Describe the parameters you would use to determine when to scale up or down (e.g., CPU/GPU utilization, job queue length).
  - Explain the implementation of the auto-scaling mechanism (e.g., AWS Auto Scaling, custom scaling logic).
  - Discuss how you would handle sudden spikes in demand and ensure high availability.
- **Result Storage and Retrieval**:
  - Explain how and where the processed results will be stored (e.g., AWS S3, databases).
  - Describe how the results will be retrieved and made accessible to the end-users.
  - Discuss any caching mechanisms or optimization strategies you would use to improve performance.

3. Implementation Details

Provide the following implementation details:

- **AWS Services and Tools**: List the specific AWS services and tools you would use in your design (e.g., AWS S3, AWS Lambda, EC2 Auto Scaling, AWS CloudWatch).
- **Configuration and Setup**: Describe the configuration and setup steps required to implement your design.
- **Monitoring and Logging**: Explain how you would monitor the system's performance and logs (e.g., AWS CloudWatch, ELK Stack).
- **Security Considerations**: Discuss the security measures you would implement to protect data and ensure secure communication between components (e.g., IAM roles, VPC, encryption).

4. Challenges and Solutions

Identify potential challenges in implementing your design and provide solutions for each. Consider aspects such as:

- Handling large data volumes
- Managing GPU resource allocation
- Ensuring seamless scaling and high availability
- Addressing latency and performance bottlenecks

5. Documentation and Presentation

- Prepare a comprehensive documentation of your design and implementation details.
- Create a presentation (e.g., slides) summarizing your architecture, design decisions, and implementation steps.
- Be prepared to present your design and answer questions related to your approach.

Submission Guidelines

- **Format**: Submit your high-level architectural diagram, detailed design explanation, implementation details, and documentation as a PDF document.