**Assignment 3: Exploring Replication in Distributed Databases with Cassandra**

**Objective:**

This assignment is designed to give you hands-on experience with **replication** in distributed databases using **Apache Cassandra**. You will learn how to configure replication settings, experiment with different replication strategies, and observe the impact of replication on data consistency and availability.

**Requirements:**

* **Apache Cassandra** installed locally.
* Knowledge of **CQL (Cassandra Query Language)** for database operations.

**Part 1: Setting Up a Cassandra Cluster with Replication**

1. **Cluster Initialization**:
   * Set up a Cassandra cluster with at least **three nodes**. You can use Docker, VMs, or configure multiple instances on your local machine.
   * Verify that all nodes are connected and part of the cluster.
2. **Create a Keyspace with Replication**:
   * Create a keyspace called university\_data using **SimpleStrategy** with a replication factor of **3**.
   * Write the CQL command used and explain the impact of this replication strategy on data distribution and fault tolerance.

**Part 2: Experimenting with Different Replication Strategies**

1. **SimpleStrategy vs. NetworkTopologyStrategy**:
   * **Step 1**: Create a new keyspace called global\_data using **NetworkTopologyStrategy**, assuming two datacenters: DC1 with a replication factor of **2** and DC2 with a replication factor of **1**.
   * **Step 2**: Write a brief comparison of SimpleStrategy and NetworkTopologyStrategy, focusing on how each strategy affects data availability and replication in different geographical locations.
2. **Switching Replication Factor**:
   * Alter the replication factor for the university\_data keyspace to **2**. Observe any changes and document how it impacts the **read and write availability** across nodes.

**Part 3: Working with Data and Observing Replication**

1. **Create a Table and Insert Data**:
   * In the university\_data keyspace, create a students table to store information about students, including **student\_id (PRIMARY KEY)**, **name**, **age**, and **department**.
   * Insert five sample records into the students table. Use UUID() to generate unique student\_id values.
2. **Verifying Data Replication**:
   * On each node in your cluster, execute a SELECT query to confirm that data is replicated correctly across nodes. Note any differences in query response times or errors if any nodes are temporarily taken offline.
3. **Simulating Node Failures**:
   * Shut down one of the nodes and perform a read and write operation on the remaining nodes.
   * Document how Cassandra handles these operations, specifically observing **consistency** and **availability**.

**Part 4: Consistency Levels and Testing**

1. **Consistency Level Testing**:
   * Set different **consistency levels** (e.g., ONE, QUORUM, ALL) for read and write operations on the students table.
   * Run read and write operations at each consistency level and note:
     + Latency or performance differences.
     + Success or failure of each operation (especially when nodes are offline).
2. **Analysis**:
   * Summarize the impact of using different consistency levels on **data consistency, latency, and fault tolerance**.
   * Explain scenarios where you would choose lower or higher consistency levels based on application needs.

**Part 5: Write a Summary and Reflection**

1. **Summary of Findings**:
   * Write a 1-2 page report summarizing the following:
     + Differences observed between **SimpleStrategy** and **NetworkTopologyStrategy**.
     + Effects of changing the **replication factor**.
     + How Cassandra’s replication strategies affected **availability and performance** under node failures.
     + Impact of varying **consistency levels** on data access.

**Submission**

Submit the following:

* **Code snippets and screenshots** of key CQL commands and results.
* **Summary report** with findings and observations.

**Grading Criteria**

* **Completeness**: All parts of the assignment are completed with detailed answers.
* **Accuracy**: Commands and configuration steps are correct, and explanations are relevant.
* **Depth of Analysis**: Thoughtful comparisons, analysis of replication strategies, and consistency levels.
* **Clarity**: Report is well-organized, clearly written, and easy to follow.