REPUBLIC OF THE PHILIPPINES

BICOL UNIVERSITY

BICOL UNIVERSITY POLANGUI

MEMBERS:

BORBE, RECHELLE B.

RICO, SUSAINE R

SAN JUAN, SHAINE S.

REODIQUE, CURT JUSTIN N.

VALEZA, MHELARRY O.

SAÑADO, FAITH ANN N.

Optimization Report

The dataset generation script is designed to create and insert large amounts of data into a MySQL database, with 100,000 records for categories, discounts, suppliers, and users. While the script works as intended, it faced several performance challenges due to the high volume of data being processed. These challenges included slow query execution, server timeouts, and potential connection issues. For example, inserting all 100,000 records at once for each table overwhelmed the database, causing errors like Error Code: 2013 (lost connection to the server). Additionally, the lack of proper indexing on key columns, such as product_id, category_id, and username, made queries slower, especially when performing joins or lookups.

To address these issues, several improvements were made. First, the script was updated to use batch inserts, breaking the data into smaller chunks (e.g., 1,000 records at a time). This reduced the load on the database and prevented timeouts. Indexes were also added to frequently queried columns, such as Categories(category_name), Discounts(product_id, is_active), and Users(username, email). These indexes significantly improved the speed of queries by making it faster for the database to find and join related data. Additionally, the script was enhanced to ensure unique values for critical fields like username and email, reducing the risk of duplicate data and improving overall data quality.

The impact of these changes was noticeable. Query execution times Improved significantly. For example, Query 2, which initially took 1.797 seconds, now runs in under 0.5 seconds. Similarly, Query 3, which took 3.610 seconds, now completes in under 1 second. These optimizations also reduced the strain on the database server, making it more stable and capable of handling large datasets without crashing. The use of the EXPLAIN command to analyze query performance helped identify bottlenecks and guided the implementation of these improvements.

In addition to performance gains, the script became more reliable and scalable. Retry logic was added to handle temporary connection issues, ensuring that the script could recover and continue processing data. The use of the Faker library to generate realistic sample data added value by creating meaningful test data for development purposes. However, further improvements could still be made, such as implementing multi-threading to process data in parallel or normalizing the database schema to reduce redundancy.

In conclusion, the optimizations applied to the script addressed key performance issues, making it faster, more efficient, and more reliable. These changes not only improved the current functionality but also prepared the script for handling even larger datasets in the future. By focusing on batch processing, indexing, and query analysis, the script is now better equipped to meet the demands of large-scale data operations.