**ST. PAUL’S UNIVERSITY**

**LIMURU CAMPUS**

**REGULAR**

**UNIT CODE: BCS4103**

**UNIT NAME: ADVANCED DATABASE SYSTEMS**

**LECTURER: MADAM CECILIA NANFUKA**

**PROJECT\_DOCUMENTATION**

**GROUP H**

**GROUP MEMBERS:**

**BSCLMR 144822**

**BSCLMR 270822**

**BSCLMR 250222**

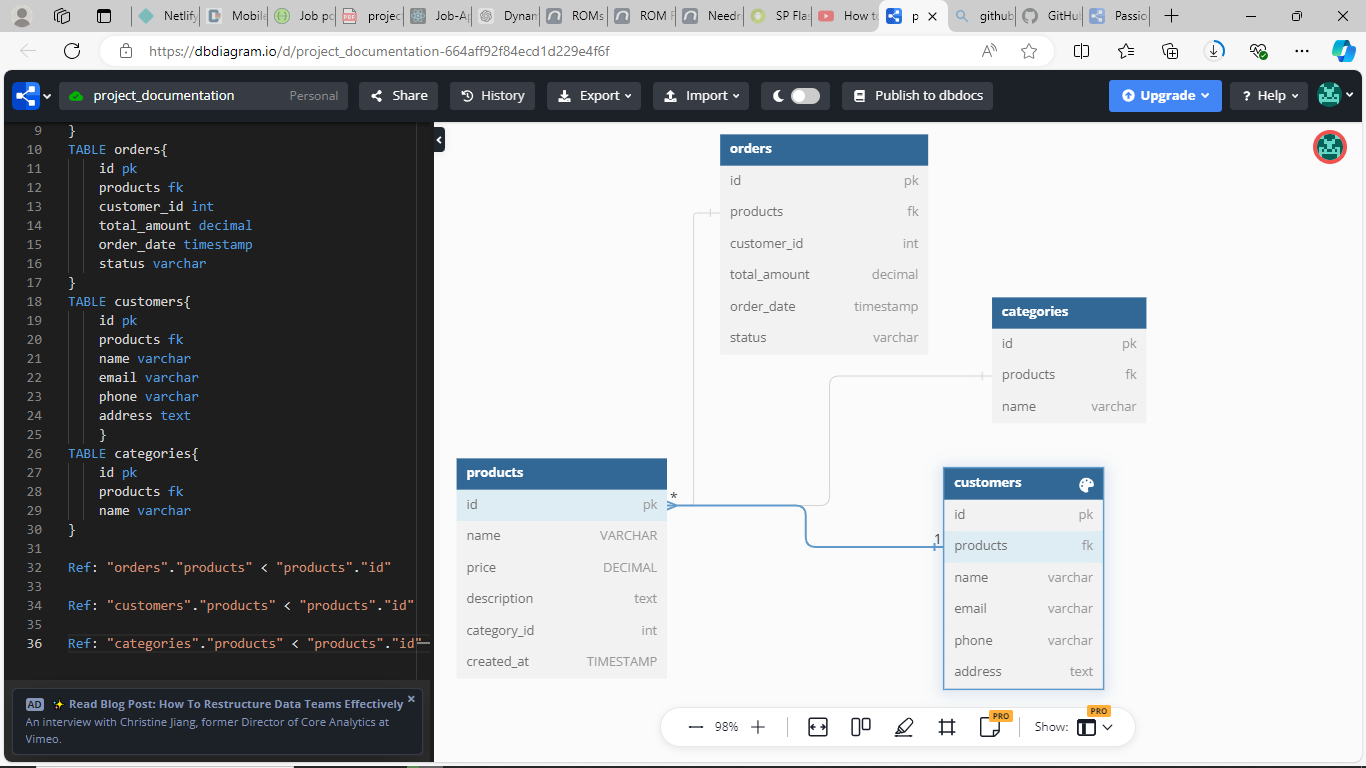
**BSCLMR 148922**

**INTRODUCTION:**

The ShopEase platform is an online retail platform that gets very busy during sales events. However, the current system which uses PostgreSQL for the database and Node.js for backend operations, faces problems during peak shopping times because of its slow query performance. This project primarily aims to increase ShopEase’s database system efficiency using PostgreSQL, API development in Node.js, and API documentation through Swagger. The main emphasis will be laid on performing triggers as well as stored procedures so as to optimize on their betterment in performance.

**Schema Design and Data Population Review of Existing Schema:**

In the beginning was the review. The schema was closely inspected to unveil its weaknesses and hence pave way for potential improvements, including normalization, indexing, and partitioning. This was a ritual performed with the utmost care and precision, guaranteeing that the database structure would be fine-tuned to ensure optimal performance in both data retrieval and data management.

****

**Data Population:**

Sketches and shadows began filling the empty cavern of our database schema as we embarked on the task of populating data. To mimic real life usage, a large volume of sample data was injected into the database. We created a script using the Faker library in Python to produce 11,096 fresh entries for important tables including categories, products, orders, order\_items, and customers. Subsequently, this information was stored onto a CSV file and later funneled into the PostgreSQL database. Establishing Node.js and Express.js a Node.js project was initiated and Express.js was incorporated to build RESTful APIs. This setup gave a solid foundation which is able to manage the HTTP requests and responses that come its way.

**Backend Development**

Designing RESTful Backend APIs Using Node.js and Express.js An initiation of a Node.js project was done, with the setup of Express.js in place, all to actualize the development of RESTful APIs. Such an ecosystem could be seen as a strong foundation for building API backends due to its ability to elegantly handle HTTP requests and responses. Crafting CRUD APIs Product management was executed through CRUD (Create, Read, Update, Delete) APIs. These included the following access points: Retrieve all available products, Fetch a specific product, by providing its unique identifier Introduce a new product into the system Modify details of an existing product Wipe out any record of a product from the database; after all, not every item is meant to stay for eternity. PostgreSQL was optimized for efficient connection pooling, ensuring effective management of database connections.

**API Documentation**

API documentation was done by use of Swagger. The seamless integration of Swagger into the Node.js project was what made it possible to document the APIs. With this implementation came an interface that was not only visually appealing but also interactive— users could now easily grasp the functionalities of the endpoints and even test them on the fly. As for documenting endpoints— no stone was left unturned. A comprehensive guide for all API endpoints developed in the previous task was made available through Swagger UI. This meant that users could do more than just read about an endpoint from the documentation; they could actually play around with it as if it were live code.

**Query optimization — Stored Procedures:**

An art form in itself, often overlooked by many developers yet capable of unleashing untold power upon databases with a mere stroke (or several strokes) of well-crafted SQL queries. Stored procedures were implemented in this way: written for those queries that were both complex and often used. Consider an instance where a stored procedure was developed to find the total sales value during a certain period— this eliminates the necessity of embedding convoluted query logic within application code. Triggers The triggers were introduced for these purposes: automating tasks of routine nature and enforcing the business rules without manual effort. Let's take an example where we create a trigger that updates inventories automatically upon reception of any new order— hence ensuring consistency in data, reducing human intervention.

* In our case, we implemented query optimization by creating functions which calculates the total sales using quantity and price form products and order tables with help of swagger.
* In addition, we did create a trigger which automatically resets created\_date to recent time column from products table every time a product is updated.

**Recommendations**

**Regular Schema review:** always remember to perform regular reviews and updates. This helps the database adapt itself to changes in business requirements and growth of data.

**Monitoring performance periodically:** use continuous monitoring tools that are able to track the database performance all the time. Keep an eye out for any bottlenecks that may arise at any moment.

**Further automation:** look for more ways to automate using triggers plus stored procedures, so as to reduce manual intervention and ensure consistency of data quality at all times. Future scalability should be taken into account in the planning process, for which sharding, replication and other advanced database techniques are likely to have good results for managing high traffic and large data volumes.

**REFERENCES:**

1. kaggle.com – online shopping e commerce dataset
2. Ahmed, I., Fayyaz, A., & Shahzad, A. (2015). PostgreSQL Developer's Guide (Vol. 1). Packt Publishing.
3. Mardan, A. (2014). Express. js Guide: The Comprehensive Book on Express. js. Azat Mardan.
4. Casciaro, M., & Mammino, L. (2016). Node. js Design Patterns. Packt Publishing Ltd.
5. MAITHANI, V. (2020). REST API AND DOCUMENTATION USING SWAGGER.
6. Subramanian, H., & Raj, P. (2019). Hands-On RESTful API Design Patterns and Best Practices: Design, develop, and deploy highly adaptable, scalable, and secure RESTful web APIs. Packt Publishing Ltd.