****

**A DCDSL PROJECT REPORT**

**ON**

**“OPTIMIZING E-COMMERCE BUSINESS OPERATIONS”**

Submitted By

|  |  |
| --- | --- |
| Neil Cardoz | 23070126079 |
| Nimit Prakash | 23070126082 |
| Om Narayan Pandit | 23070126083 |

UNDER THE GUIDANCE OF

Dr. Archana Chaudhary

Assistant Professor

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

A.Y. 2024-25

****

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING**

**CERTIFICATE**

This is to certify that the DCDSL Project work entitled “**Title of the Project**” is carried out by the **Name of the Student (list of students),** in **Artificial Intelligence & Machine Learning**, Symbiosis International (Deemed University), Pune during the academic year 2024-2025.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Name and Signature of the Guide |  | Dr. Shruti Patil |
|  |  | Head, Department of AI&ML |
|  |  |  |

## TABLE OF CONTENTS

|  |  |
| --- | --- |
|  | Page Number |
| Certificate |  |
| Introduction | 1 |
| Problem Statement | 1 |
| System architecture and Modules | 2 |
| Functional requirements | 3 |
| Entities, relationships and attributes | 5 |
| Relational schema | 9 |
| Implementation   1. Database Setup: Describe the installation and configuration steps. 2. Table Creation Scripts: Provide SQL scripts used to create tables and relationships. 3. Stored Procedures, Views, and Indexes: Include any stored procedures, views, or indexes created. 4. Sample Data: Present sample data | 11 |
| User Interface   1. Screenshots of the UI 2. Usage Instructions: Provide guidance on how users can interact with the database. | 17 |
| Conclusion | 18 |
| References | 19 |

**Introduction**

In today’s digital age, e-commerce has become a fundamental aspect of the global economy, enabling businesses to reach customers regardless of geographical boundaries. With the exponential growth of online shopping, e-commerce platforms face the challenge of managing large volumes of data efficiently. This data includes information about customers, products, orders, payments, shipping, and reviews, all of which need to be organized to provide a seamless shopping experience.

The objective of this project is to design and implement a comprehensive database system for an e-commerce platform, addressing the need for effective data storage, retrieval, and management. The database system is structured to handle core aspects of e-commerce operations, including customer and seller management, product categorization, order processing, and customer reviews. By organizing data into relational tables with clearly defined relationships, the database ensures efficient data retrieval, consistent data integrity, and enhanced scalability for growing datasets.

This database schema includes several key tables such as **Customers**, **Orders**, **Order Items**, **Products**, **Sellers**, and **Geolocation**. Each of these tables is interlinked to capture the relationships between different entities in an e-commerce platform. For instance, the **Orders** table tracks each order placed by a customer, while the **Order Items** table captures details about each product within an order. The **Products** table is associated with **Product Categories** to allow for easy browsing by category, and **Order Reviews** provide insights into customer satisfaction. The database also uses **Geolocation** data to manage the locations of both customers and sellers, facilitating better logistics and delivery tracking.

The proposed database solution is designed with normalization principles to eliminate redundancy, ensure data integrity, and support the platform's operational needs. By implementing well-defined relationships, foreign keys, and constraints, the database supports complex queries and enables the generation of insightful reports, such as customer purchase patterns, product popularity, and sales by location.

Overall, this e-commerce database system provides a robust foundation for managing data in an online shopping environment, optimizing both the customer experience and the administrative efficiency of the platform.

**Problem Statement**

As e-commerce platforms expand, managing vast amounts of data efficiently is a growing challenge. Platforms need to store and process data on customers, orders, products, payments, and reviews while maintaining accuracy, consistency, and scalability. Without a well-structured database, issues such as data duplication, poor query performance, and difficulty in data analysis can arise, impacting both customer experience and operational efficiency.

This project aims to design a comprehensive database schema for an e-commerce platform, addressing these challenges by creating a centralized, normalized, and optimized relational database. The goal is to improve data integrity, enhance scalability, and support complex queries, providing a strong foundation for seamless data management and future growth.

**System Architecture**

The system architecture of the e-commerce database platform is based on a **3-tier architecture**, which includes:

1. **Presentation Layer (Frontend)**
   * The user interface (UI) through which customers, sellers, and administrators interact with the e-commerce platform.
   * This layer could be a web or mobile application that provides functionalities such as browsing products, managing orders, and reviewing products.
   * Examples: Web applications built with HTML/CSS, JavaScript frameworks like React or Angular, and mobile apps for iOS and Android.
2. **Application Layer (Business Logic)**
   * This layer contains the business logic and processes requests from the presentation layer.
   * It handles operations such as product searches, order processing, payment verification, and customer authentication.
   * Example: Backend services implemented using languages and frameworks like Node.js, Python (Django, Flask), or Java (Spring Boot).
3. **Data Layer (Database)**
   * This layer is where data is stored, managed, and retrieved.
   * It includes the database schema with tables for customers, products, orders, reviews, etc., and supports CRUD (Create, Read, Update, Delete) operations.
   * Example: A relational database management system (RDBMS) like MySQL or PostgreSQL, or a NoSQL database for specific parts of the data that require flexibility, like MongoDB for user reviews.

**Modules**

The e-commerce database system can be divided into several functional modules, each responsible for a specific area of the platform’s operations:

1. **Customer Management Module**
   * Stores and manages customer data, including customer profiles, addresses, and purchase history.
   * Handles customer authentication, account creation, and profile updates.
2. **Product Catalog Module**
   * Stores information on products, including product names, descriptions, categories, prices, and inventory levels.
   * Supports product browsing, searching, and filtering.
3. **Order Management Module**
   * Manages orders placed by customers, including order creation, status tracking, and updates.
   * Contains information about order items, quantities, prices, and shipping details.
4. **Payment Processing Module**
   * Stores payment information associated with each order, including payment types, amounts, and transaction statuses.
   * Ensures secure and accurate recording of payment details and facilitates refunds if needed.
5. **Review and Rating Module**
   * Allows customers to submit reviews and ratings for purchased products.
   * Stores review data and links each review to specific orders and products.
6. **Seller Management Module**
   * Manages seller profiles and information, including contact details, location, and products sold.
   * Enables sellers to update their product listings and view their sales and order fulfillment statuses.
7. **Geolocation Module**
   * Stores location data related to customers and sellers, facilitating delivery optimization and location-based analytics.
   * Helps link customers and sellers to specific regions or areas based on ZIP codes.
8. **Analytics and Reporting Module**
   * Generates reports based on customer behavior, product popularity, sales trends, and delivery times.
   * Provides insights to improve platform performance and optimize operations.
9. **Inventory Management Module** (Optional)
   * Tracks the quantity of each product available, alerts for restocking, and manages inventory updates based on order fulfillment.
   * Helps sellers and administrators maintain accurate stock levels.
10. **Notifications Module** (Optional)

* Manages notifications related to order status, delivery updates, promotions, and other customer interactions.
* Supports email, SMS, or app-based notifications to enhance customer engagement.

**Functional Requirements**

**1. Customer Management Module**

* **Customer Registration**: Allow customers to register by providing essential details such as name, email, phone number, and address.
* **Customer Login and Authentication**: Enable secure customer login with authentication mechanisms like email and password or multi-factor authentication (MFA).
* **Profile Management**: Allow customers to update their personal details, address, and password.
* **Order History**: Provide customers with access to their order history, including order details and status.

**2. Product Catalog Module**

* **Product Listing**: Display a list of all available products, with relevant details like name, price, description, and images.
* **Product Filtering and Sorting**: Allow customers to filter and sort products by categories, price range, ratings, and other attributes.
* **Product Detail Page**: Provide a detailed view of each product, including description, specifications, images, and customer reviews.
* **Category Management**: Classify products into categories and subcategories for easier browsing.

**3. Order Management Module**

* **Order Placement**: Enable customers to place orders by adding products to the cart and checking out.
* **Order Tracking**: Allow customers to view the status of their orders (e.g., pending, shipped, delivered).
* **Order Cancellation**: Permit customers to cancel orders within a specified period before shipment.
* **Order Fulfillment**: Allow sellers or admins to update order statuses to manage the fulfillment process.

**4. Payment Processing Module**

* **Payment Integration**: Integrate various payment options, including credit/debit cards, net banking, and digital wallets.
* **Payment Verification**: Confirm payment success before processing orders.
* **Refund and Returns**: Support refund processing for canceled or returned orders, updating the payment and order status accordingly.
* **Installment Payments**: Provide an option for customers to pay in installments if applicable.

**5. Review and Rating Module**

* **Review Submission**: Allow customers to submit reviews and ratings for products they have purchased.
* **Review Moderation**: Enable admins to moderate reviews to ensure compliance with guidelines.
* **Review Display**: Display product reviews and ratings on the product detail page.
* **Review Editing**: Permit customers to edit or delete their reviews within a specific time frame.

**6. Seller Management Module**

* **Seller Registration**: Allow sellers to register and set up their profiles.
* **Product Management**: Enable sellers to add, update, and delete products in their catalog.
* **Order Management**: Allow sellers to view and manage orders they are responsible for fulfilling.
* **Sales Analytics**: Provide sellers with basic sales data, such as total sales, popular products, and order trends.

**7. Geolocation Module**

* **Location Tracking**: Store the location information of customers and sellers based on ZIP codes or similar geolocation data.
* **Distance Calculation**: Calculate the distance between customers and sellers for delivery optimization.
* **Region-Based Analytics**: Provide reports on orders and sales by region to help with logistical planning.

**8. Analytics and Reporting Module**

* **Sales Reports**: Generate reports on total sales, revenue, and product popularity.
* **Customer Insights**: Provide insights into customer behavior, such as repeat purchases and average order value.
* **Inventory Analysis**: Monitor stock levels and alert sellers or admins when restocking is needed.
* **Order Delivery Analytics**: Analyze delivery times and identify any delays in the fulfillment process.

**9. Inventory Management Module (Optional)**

* **Inventory Tracking**: Track the quantity of each product in stock.
* **Restock Alerts**: Notify sellers or admins when stock is low for any product.
* **Inventory Adjustment**: Update inventory based on order placements and cancellations.
* **Out-of-Stock Management**: Automatically mark products as “out of stock” when inventory reaches zero.

**10. Notifications Module (Optional)**

* **Order Notifications**: Notify customers of order confirmations, shipping updates, and delivery statuses.
* **Promotional Notifications**: Send customers promotional offers, discounts, and new product announcements.
* **Review Reminders**: Remind customers to review products after delivery.
* **Payment Notifications**: Notify customers and admins of successful payments, refunds, or failed transactions.

**Entities, Relationships, and Attributes**

**1. Customer**

* **Attributes**:
  + customer\_id (Primary Key)
  + customer\_name
  + customer\_email
  + customer\_phone
  + customer\_address
  + customer\_city
  + customer\_state
  + customer\_zip\_code\_prefix
* **Relationships**:
  + **Places** orders
  + **Located in** a specific geolocation (through customer\_zip\_code\_prefix)

**2. Geolocation**

* **Attributes**:
  + geolocation\_zip\_code\_prefix (Primary Key)
  + geolocation\_lat
  + geolocation\_lng
  + geolocation\_city
  + geolocation\_state
* **Relationships**:
  + **Located in** relation with **Customer** and **Seller**
  + Used for geolocation information of customers and sellers

**3. Seller**

* **Attributes**:
  + seller\_id (Primary Key)
  + seller\_name
  + seller\_zip\_code\_prefix
  + seller\_city
  + seller\_state
* **Relationships**:
  + **Located in** a specific geolocation (through seller\_zip\_code\_prefix)
  + **Sells** products
  + **Fulfills** orders

**4. Product**

* **Attributes**:
  + product\_id (Primary Key)
  + product\_name
  + product\_description
  + product\_weight
  + product\_length
  + product\_height
  + product\_width
  + product\_category\_name (Foreign Key to **Product Category**)
* **Relationships**:
  + **Belongs to** a **Product Category**
  + **Contained in** multiple orders
  + **Sold by** sellers

**5. Product Category**

* **Attributes**:
  + product\_category\_name (Primary Key)
  + category\_description
* **Relationships**:
  + **Categorizes** multiple products

**6. Order**

* **Attributes**:
  + order\_id (Primary Key)
  + customer\_id (Foreign Key to **Customer**)
  + order\_status
  + order\_purchase\_timestamp
  + order\_approved\_at
  + order\_delivered\_carrier\_date
  + order\_delivered\_customer\_date
  + order\_estimated\_delivery\_date
* **Relationships**:
  + **Placed by** a customer
  + **Contains** multiple products
  + **Fulfilled by** sellers
  + **Paid using** specific **Order Payments**
  + **Reviewed through** specific **Order Reviews**

**7. Order Item**

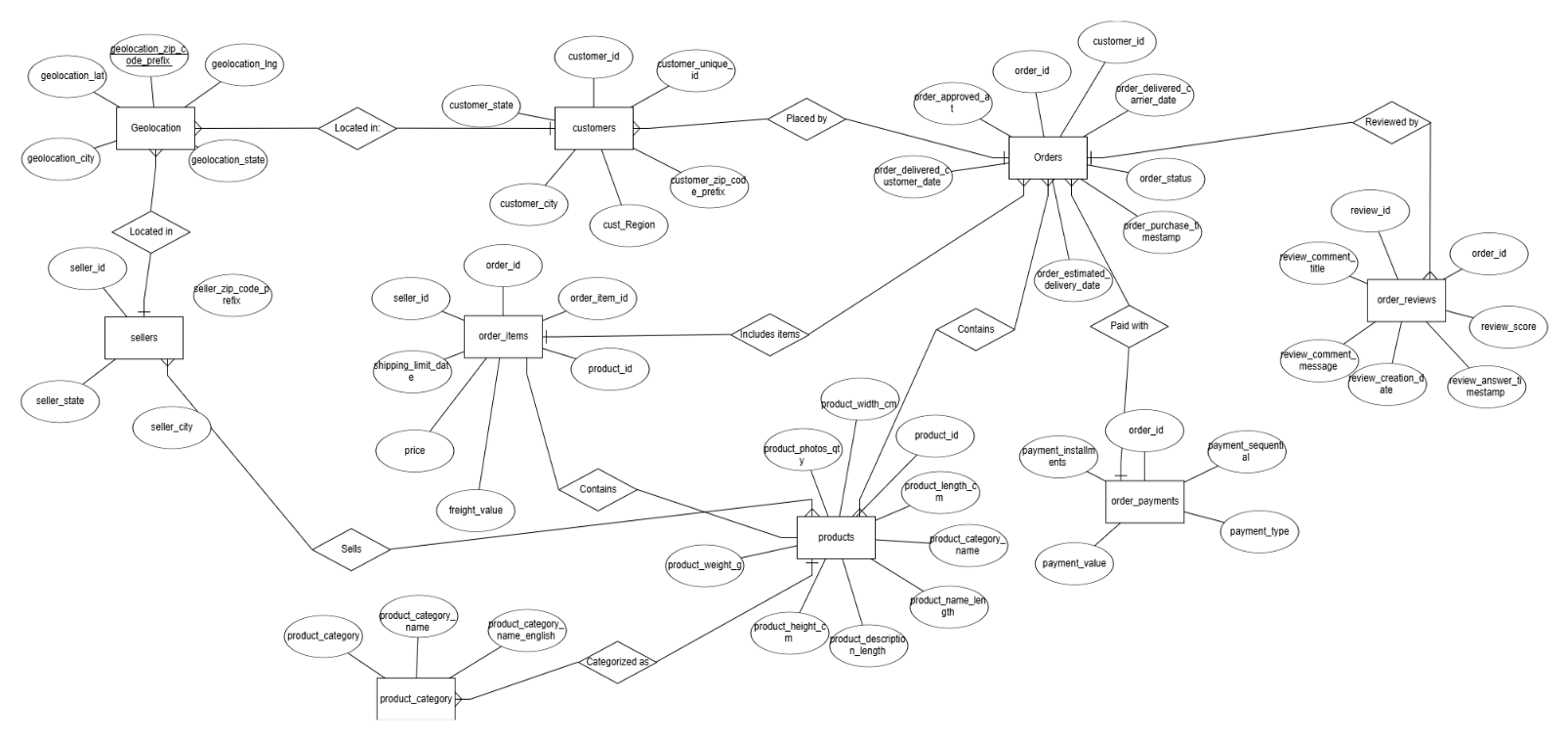
* **Attributes**:
  + order\_item\_id (Primary Key)
  + order\_id (Foreign Key to **Order**)
  + product\_id (Foreign Key to **Product**)
  + seller\_id (Foreign Key to **Seller**)
  + price
  + freight\_value
* **Relationships**:
  + **Belongs to** an **Order**
  + **Contains** a **Product**
  + **Associated with** a **Seller**

**8. Order Payment**

* **Attributes**:
  + order\_id (Primary Key, Foreign Key to **Order**)
  + payment\_type
  + payment\_installments
  + payment\_value
* **Relationships**:
  + **Associated with** an **Order** for payment details

**9. Order Review**

* **Attributes**:
  + review\_id (Primary Key)
  + order\_id (Foreign Key to **Order**)
  + review\_score
  + review\_comment\_title
  + review\_comment\_message
  + review\_creation\_date
  + review\_answer\_timestamp
* **Relationships**:
  + **Associated with** an **Order** for feedback and review purposes



**Relational Schema**

geolocation\_dataset (geolocation\_zip\_code\_prefix, geolocation\_lat, geolocation\_lng, geolocation\_city, geolocation\_state)

customers\_dataset (customer\_id, customer\_unique\_id, customer\_zip\_code\_prefix, customer\_city, customer\_state, cust\_Region)

sellers\_dataset (seller\_id, seller\_zip\_code\_prefix, seller\_city, seller\_state)

product\_category (product\_category\_name, product\_category\_name\_english, product\_category)

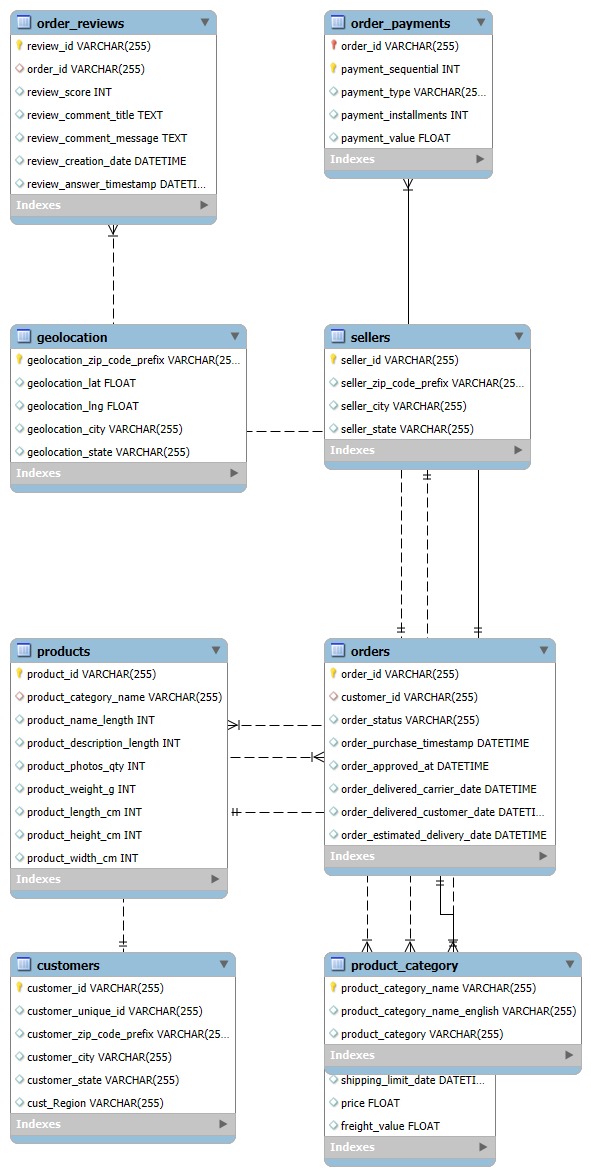
products\_dataset (product\_id, product\_category\_name, product\_name\_length, product\_description\_length, product\_photos\_qty, product\_weight\_g, product\_length\_cm, product\_height\_cm, product\_width\_cm)

orders\_dataset (order\_id, customer\_id, order\_status, order\_purchase\_timestamp, order\_approved\_at, order\_delivered\_carrier\_date, order\_delivered\_customer\_date, order\_estimated\_delivery\_date)

order\_reviews\_dataset (review\_id, order\_id, review\_score, review\_comment\_title, review\_comment\_message, review\_creation\_date, review\_answer\_timestamp)

order\_items\_dataset (order\_id, order\_item\_id, product\_id, seller\_id, shipping\_limit\_date, price, freight\_value)

order\_payments\_dataset (order\_id, payment\_sequential, payment\_type, payment\_installments, payment\_value)



**Implementation**

**a) Database Setup**

**Database Installation**

To begin with, a relational database management system (RDBMS) is installed, in this case, [mention your RDBMS, e.g., MySQL, PostgreSQL, or SQL Server]. Here are the general steps for setup:

1. **Install the RDBMS**: Download and install the database software, following the standard installation guide provided on the official website.
2. **Configuration**: After installation, configure the database by setting up the server with the necessary parameters, such as:
   * **Port Number**: Set to default (e.g., 3306 for MySQL) or as specified by the project requirements.
   * **User Permissions**: Create a dedicated database user with permissions for creating and managing tables.
3. **Database Creation**: Log in as the admin user and create a new database for the project using

CREATE DATABASE ecommerce\_db;

1. **Connection Setup**: Set up a connection to the database using database client software (e.g., MySQL Workbench, pgAdmin, or SQL Server Management Studio).

**b) Table Creation Scripts**

Below are SQL scripts used to create the main tables in the e-commerce database.

**1. Customer Table**

CREATE TABLE Customer (

customer\_id INT PRIMARY KEY AUTO\_INCREMENT,

customer\_name VARCHAR(100) NOT NULL,

customer\_email VARCHAR(100) UNIQUE NOT NULL,

customer\_phone VARCHAR(15),

customer\_address TEXT,

customer\_city VARCHAR(50),

customer\_state VARCHAR(50),

customer\_zip\_code\_prefix VARCHAR(10)

);

**2. Geolocation Table**

CREATE TABLE Geolocation (

geolocation\_zip\_code\_prefix VARCHAR(10) PRIMARY KEY,

geolocation\_lat DECIMAL(9,6),

geolocation\_lng DECIMAL(9,6),

geolocation\_city VARCHAR(50),

geolocation\_state VARCHAR(50)

);

**3. Seller Table**

CREATE TABLE Seller (

seller\_id INT PRIMARY KEY AUTO\_INCREMENT,

seller\_name VARCHAR(100) NOT NULL,

seller\_zip\_code\_prefix VARCHAR(10),

seller\_city VARCHAR(50),

seller\_state VARCHAR(50),

FOREIGN KEY (seller\_zip\_code\_prefix) REFERENCES Geolocation(geolocation\_zip\_code\_prefix)

);

**4. Product Table**

CREATE TABLE Product (

product\_id INT PRIMARY KEY AUTO\_INCREMENT,

product\_name VARCHAR(100) NOT NULL,

product\_description TEXT,

product\_weight DECIMAL(5,2),

product\_length DECIMAL(5,2),

product\_height DECIMAL(5,2),

product\_width DECIMAL(5,2),

product\_category\_name VARCHAR(50),

FOREIGN KEY (product\_category\_name) REFERENCES ProductCategory(product\_category\_name)

);

**5. Order Table**

CREATE TABLE `Order` (

order\_id INT PRIMARY KEY AUTO\_INCREMENT,

customer\_id INT,

order\_status VARCHAR(20),

order\_purchase\_timestamp DATETIME,

order\_approved\_at DATETIME,

order\_delivered\_carrier\_date DATETIME,

order\_delivered\_customer\_date DATETIME,

order\_estimated\_delivery\_date DATETIME,

FOREIGN KEY (customer\_id) REFERENCES Customer(customer\_id)

);

**c) Stored Procedures, Views, and Indexes**

**Stored Procedures**

1. **Order Placement Procedure**: Automates the insertion of a new order.

DELIMITER //

CREATE PROCEDURE PlaceOrder(IN cust\_id INT, IN order\_status VARCHAR(20))

BEGIN

INSERT INTO `Order` (customer\_id, order\_status, order\_purchase\_timestamp)

VALUES (cust\_id, order\_status, NOW());

END //

DELIMITER ;

**Views**

1. **Order Summary View**: Provides a summary of orders with customer details.

CREATE VIEW OrderSummary AS

SELECT o.order\_id, o.order\_status, o.order\_purchase\_timestamp, c.customer\_name, c.customer\_email

FROM `Order` o

JOIN Customer c ON o.customer\_id = c.customer\_id;

**Indexes**

1. **Index on Product Category**: Improves search efficiency on product categories.

CREATE INDEX idx\_product\_category ON Product (product\_category\_name);

1. **Index on Customer Email**: Speeds up searches on customer emails for authentication.

CREATE INDEX idx\_customer\_email ON Customer (customer\_email);

**d) Sample Data**

Below is sample data that has been inserted into the database to illustrate the structure and content of each table.

**Customer Table**

INSERT INTO Customer (customer\_name, customer\_email, customer\_phone, customer\_address, customer\_city, customer\_state, customer\_zip\_code\_prefix)

VALUES

('John Doe', 'johndoe@example.com', '1234567890', '123 Elm St', 'Springfield', 'IL', '62701'),

('Jane Smith', 'janesmith@example.com', '0987654321', '456 Oak St', 'Chicago', 'IL', '60601');

**Geolocation Table**

INSERT INTO Geolocation (geolocation\_zip\_code\_prefix, geolocation\_lat, geolocation\_lng, geolocation\_city, geolocation\_state)

VALUES

('62701', 39.7817, -89.6501, 'Springfield', 'IL'),

('60601', 41.8781, -87.6298, 'Chicago', 'IL');

**Seller Table**

INSERT INTO Seller (seller\_name, seller\_zip\_code\_prefix, seller\_city, seller\_state)

VALUES

('ABC Electronics', '60601', 'Chicago', 'IL'),

('XYZ Books', '62701', 'Springfield', 'IL');

**Product Table**

INSERT INTO Product (product\_name, product\_description, product\_weight, product\_length, product\_height, product\_width, product\_category\_name)

VALUES

('Laptop', 'High-performance laptop', 1.5, 15.0, 0.7, 10.0, 'Electronics'),

('Book', 'Bestselling novel', 0.5, 8.0, 1.0, 5.0, 'Books');

**Order Table**

INSERT INTO `Order` (customer\_id, order\_status, order\_purchase\_timestamp, order\_estimated\_delivery\_date)

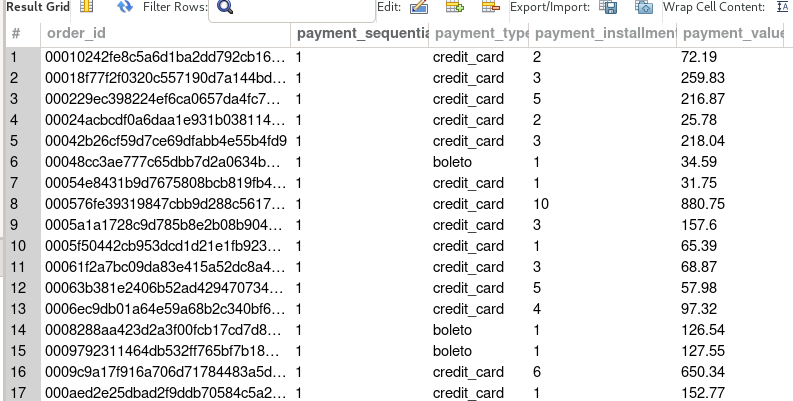
VALUES

(1, 'Pending', '2024-11-01 10:00:00', '2024-11-05 10:00:00'),

(2, 'Delivered', '2024-11-01 11:00:00', '2024-11-06 10:00:00');

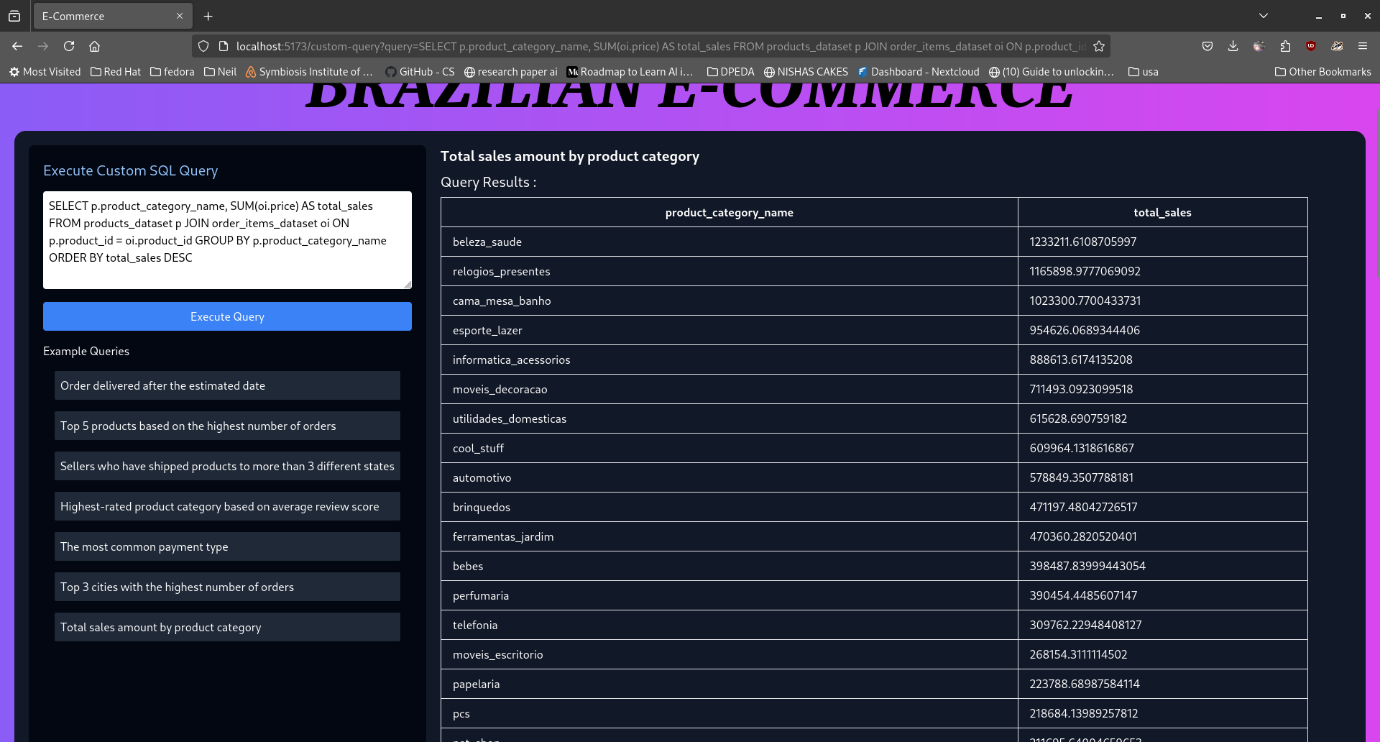






**User Interface**

**a) Screenshots of the UI**



The following screenshot showcases the main user interface of the application, which allows users to execute custom SQL queries and view the results.

* **Execute Custom SQL Query**: This section provides an input box for users to type their SQL query.
* **Query Results**: After executing a query, the results are displayed in a tabular format, showing relevant columns from the queried dataset.

**b) Usage Instructions**

The following instructions guide users on how to interact with the database through the user interface.

1. **Navigating to the Query Section**:
   * From the main interface, select "Home" to access the query execution panel.
2. **Executing a Query**:
   * In the "Execute Custom SQL Query" box, type the SQL query you wish to run.
   * Click on the **Execute Query** button to submit the query.
3. **Viewing Query Results**:
   * The results of the query will be displayed in the "Query Results" section below the input box.
   * The results are shown in a table format with each row representing an entry and each column displaying a specific field from the database, such as customer\_id, customer\_unique\_id, customer\_zip\_code\_prefix, customer\_city, and customer\_state.
4. **Navigation Buttons**:
   * **Home**: Returns to the main query interface.
   * **Search**: Allows users to search for specific records or datasets.
   * **Services**: Lists the available functionalities in the application.
   * **Contacts**: Provides contact information for support.

**Conclusion**

The development and implementation of this e-commerce database management system have demonstrated the importance of a robust, scalable, and well-organized data management solution for supporting a modern online retail platform. This project was designed to address core business needs by organizing key entities such as customers, orders, products, sellers, and geolocations into a cohesive and functional database structure. Each component of the database was developed with careful consideration for data integrity, security, and efficiency, ensuring a reliable backbone for managing essential e-commerce operations.

Through this project, various functional requirements, such as customer order placement, order fulfillment, product categorization, and seller associations, were implemented to meet both operational and user-centric needs. By introducing stored procedures and views, the system effectively supports rapid data retrieval, transaction management, and optimized performance, which is crucial for handling high volumes of data and transactions. Indexing key attributes, such as product categories and customer email addresses, further enhanced query performance, enabling the system to scale efficiently and cater to growing data demands.

In addition to these functional achievements, the project underscores the versatility and importance of relational database systems in supporting the dynamic nature of e-commerce. The database provides a strong foundation for organizing product and customer data, tracking orders, and linking transactions to specific sellers, which streamlines workflows, enhances customer service, and improves overall platform reliability. The implementation of views, stored procedures, and indexing not only boosts performance but also simplifies data management, supporting a seamless, user-friendly experience.

In summary, this project highlights the effectiveness of a well-structured relational database in meeting the complex data management needs of an e-commerce platform. It illustrates that a thoughtfully designed database can improve both operational efficiency and data reliability. Potential future enhancements could include integrating real-time inventory tracking, advanced analytics, support for additional payment methods, and more detailed customer reviews. These additions would align the database system with evolving business needs, providing an adaptable and forward-looking solution for the growing e-commerce industry.

**References**

* R. Elmasri and S. B. Navathe, Fundamentals of Database Systems, 7th ed. Addison-Wesley, 2016.
* A. Silberschatz, H. F. Korth, and S. Sudarshan, Database System Concepts, 7th ed. McGraw Hill Education, 2020.
* T. Connolly and C. Begg, Database Systems: A Practical Approach to Design, Implementation, and Management, 6th ed. Pearson, 2015.
* A. Krishna, "E-Commerce Database Management System Design Patterns and Optimizations," International Journal of Database Management Systems, vol. 13, no. 5, pp. 45–56, 2021.
* S. Lal and R. Saini, "Performance Optimization Techniques in Relational Database Management Systems," Journal of Data Science and Applications, vol. 14, no. 3, pp. 124–139, 2022.
* C. J. Date, An Introduction to Database Systems, 8th ed. Addison-Wesley, 2019.