Inventory Management and Online Order Tracking

Eva Pradhan
University at Buffalo
50414973
evapradh@buffalo.edu

Harshitha Damineni
University at Buffalo
50471119
hdaminen@buffalo.edu

Venkata Datta Viswanadha Vishnubhotla
University at Buffalo
50465091
vvishub@buffalo.edu

I. Problem Statement

A. Description

Most businesses need an inventory management database system to keep track of their products and stock levels, as well as an online order tracking database system to manage customer orders and shipments. If a system is not in place to keep track it can lead to overselling or stockouts, resulting in lost sales, missed sales opportunities and dissatisfied customers. Furthermore, without a system for tracking online orders, it can be challenging to monitor the status of orders, such as whether they have been fulfilled or shipped, causing delays in delivery times and customer dissatisfaction. Additionally, they require a user review system to allow customers to leave feedback on their purchases. It can be used for various analyses such as sales forecasting, customer segmentation, and product recommendation systems. On other hand, manual inventory tracking can be time-consuming and prone to errors, leading to inaccurate data and delays in decision making. Therefore, the need arises for an inventory management database system with an online order tracking database system that can efficiently manage and track small to large inventory levels, orders, and shipments. This system must be capable of monitoring real-time stock levels, generating alerts for low stock levels, managing purchase orders, and updating inventory levels automatically. Additionally, the system should be able to track online orders from the time they are placed to the point of delivery and provide customers with real-time updates on the status of their orders. With such a system in place, businesses can efficiently manage inventory levels, reduce the risk of stockouts, improve customer satisfaction, and streamline their overall operations. The system should be user-friendly, efficient, and scalable to accommodate the company's growth. The company also needs a robust reporting and analytics feature to monitor inventory levels, track inventory movements, and identify trends and patterns to make informed decisions. Finally, the system should have strong security features to protect sensitive inventory data from unauthorized access and cyber threats. The system should also allow customers to receive updates on their orders via email or SMS notifications. The system should be able to handle order information such as order status, delivery date, and shipping information. This will improve customer satisfaction by providing a seamless and transparent order tracking experience, which in turn will improve customer loyalty and increase sales for the business.

B. Database vs Excel Files

- In terms of data organization, consistency, security, scalability, and sharing/collaboration, databases outperform Excel files.
- While databases are intended to handle complicated queries and data interactions more effectively than Excel, they can provide greater performance when working with massive datasets. They also make it simple to create reports and visualizations from the data, enabling more insightful analysis and wiser decision-making.
- Moreover, databases give data a centralized location, lowering the possibility of data loss or corruption that can happen when Excel files are kept in many locations. This can reduce the time and effort required to retrieve lost or damaged data.
- Also, databases allow for more complex data conversions and calculations than Excel can, giving them greater versatility in terms of data manipulation. Businesses who need to run advanced analytics or machine learning on their data may find this to be extremely helpful.
- Overall, databases offer a more robust and scalable solution for enterprises with bigger volumes of data that require secure, consistent, and efficient management, whereas Excel can be effective for small datasets and straightforward analytics.

II. Target User

Depending on the size and nature of the company, as well as the particular needs of the system, a variety of workforce may be the target users for an inventory management system and online order tracking. Here are a few examples of the intended users:

- E-commerce business owners: These people are in charge of overseeing the overall operations of the e-commerce business. They can use the management system to track sales, inventory levels, order management, and customer behavior.
- Customer service representatives: These individuals have the responsibility of responding to consumer questions and addressing any problems with orders. They can view order histories, access customer information, and handle returns or refunds via the management system.
- Logistics and shipping personnel: These people are responsible for organizing the shipping and delivery of

items to clients. Users could create shipping labels, track orders, and manage inventories using the management system.
- Inventory managers and supervisors: These users are in charge of monitoring inventory levels and ensuring that products are available when needed. An inventory management system can assist them in tracking inventory levels, monitoring sales, and making informed decisions about purchasing and stocking products.
- Purchasing and procurement staff: These users are in charge of ordering new inventory and ensuring that the company has the materials and products it needs to run efficiently. An inventory management system can help them track orders, manage suppliers, and ensure that the right products are available at the right time.
- Accounting and finance personnel: These people are in charge of overseeing the business's finances, which includes keeping track of inventory expenses and sales revenue.
- Sales representatives: These people are in charge of processing customer orders and making sure that orders are delivered promptly and accurately. They may create reports, manage inventories, and track orders using the management system.
- Data analysts: In order to find trends and patterns in customer behavior, product sales, and money generated by the e-commerce platform, these people may use the database management system to extract data from the system and conduct analysis.

III. Real Life Scenario

A retail company that sells things both online and in-person might be a real-world example of how an inventory management system and online order monitoring are used. The company maintains a sizable inventory of goods that are kept in one or more warehouses. The products' quantities, locations, and any status changes would all be tracked in real-time by the inventory management system (such as being shipped or received).

The online order tracking system enables customers to follow their orders when they are placed online and up until they are delivered. The user would be able to view the pick-up, packing, and shipment dates of the purchase as well as any tracking details offered by the shipping carrier. As orders are filled, the inventory management system would automatically update the inventory levels.

A product's availability on the website will be automatically updated by the inventory management system whenever it runs out of stock. The technology would also send alerts when inventory levels reached a predetermined level, urging the company to place new orders for goods and replenish the warehouse.

The company can guarantee that it always has enough inventory to fulfill customer demand by putting in place an efficient inventory management system and an online order tracking system, as well as by giving customers the ability to track their orders in real-time.

Here are some potential real-life scenarios where this dataset could be used:

- Market Analysis: The dataset can be used to assess market trends, including popular product categories, customer behavior, and seller performance, by an online retailer interested in breaking into the market. The corporation can use this research to make well-informed decisions regarding the products it offers and the way it markets those products.
- Customer Segmentation: An online store can use the dataset to classify its customers based on their demographics, buying habits, and other traits. The business may boost consumer engagement and loyalty by tailoring its marketing messaging and promotions to certain customer groups.
- Supply Chain Optimization: A logistics business might use the dataset to assess how successfully different suppliers manage orders and adhere to delivery dates. This analysis can help the company optimize its supply chain by identifying reliable providers and accelerating delivery.
- Fraud Detection: The dataset can be used by an online retailer to spot phony orders and sales. The business can create methods and algorithms to stop fraud in the future by studying the patterns of fraudulent activity.
- Product Recommendation: An online store can use the dataset to develop algorithms that recommend products to customers based on their prior purchases and browsing patterns. As a result, customers might be more inclined to make additional purchases, which will improve their whole shopping experience.

IV. Data Description

The e-commerce market's dataset contains data on orders, consumers, sellers, products, and reviews. The e-commerce platform Olist, which offers marketplace services to small and medium-sized firms, is the source of the dataset.

The dataset is composed of multiple tables, including:

- Seller dataset: This file includes details about the marketplace vendors, including seller ID, seller name, and seller zip code.
- Customers dataset: This file includes details about the customers who placed the orders, including their name, city, state, and zip code, as well as a special customer ID.
- Order items dataset: Each order's items are listed in this file together with details such as the order ID, product ID, seller ID, price, freight value, and quantity.
- Geolocation dataset: The latitude and longitude of the buyers' and sellers' locations are included in this file along with other geolocation data.
- Order payments dataset: This file includes details on payments made for each order, including the value, kind, and installments of payments.

- Orders dataset: The order ID, customer ID, order status, purchase timestamp, and anticipated delivery date are just a few of the details regarding the orders that are included in this file.
- Products dataset: The product ID, product category name, product name, and product description are among the details about the goods offered on the marketplace that are included in this file.
- Order reviews dataset: The order ID, review score, review comment title, and review comment message are among the details regarding the reviews that consumers have written for the orders that are included in this file.
- Order items dataset: Each item of an order is described in detail in the order Items dataset. For each item, the dataset specifically offers details such as order id, order item id, product id, seller id, shipping limit date, etc.
- Product Seller dataset: This table maps the product id to corresponding seller id.

In conclusion, the E-Commerce Public Dataset by Olist offers a wealth of information for studying the e-commerce business, including consumer behavior, market trends, and seller performance.

V. ER Diagram

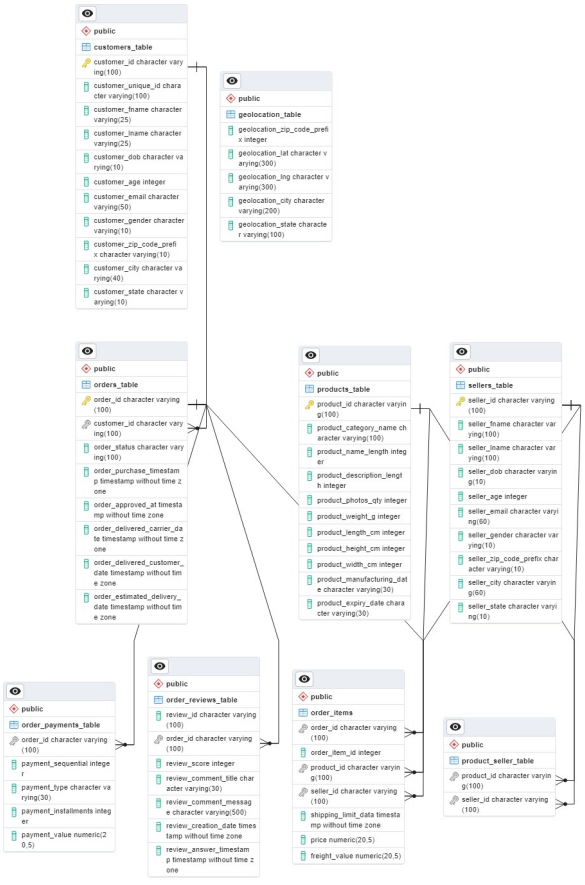


Fig. 1. ER diagram

VI. Database Schemas

- SELLERS_TABLE (seller_id VARCHAR primary key, seller_zip_code_prefix INT, seller_city VARCHAR, seller_state VARCHAR, seller_fname VARCHAR, seller_lname VARCHAR, seller_email VARCHAR, seller_gender VARCHAR, seller_dob VARCHAR, age INT)
- PRODUCTS_TABLE (product_id VARCHAR primary key, product_category_name VARCHAR, product_name_lenght INT, product_description_lenght INT, product_photos_qty INT, product_weight_g INT, product_length_cm INT, product_height_cm INT, product_width_cm INT, product_manufacturing_date VARCHAR, product_expiry_date VARCHAR)
- CUSTOMERS_TABLE (customer_id VARCHAR, customer_unique_id VARCHAR, customer_fname VARCHAR, customer_lname VARCHAR, customer_DOB VARCHAR, customer_age INT, customer_email VARCHAR, customer_zip_code_prefix VARCHAR, customer_gender VARCHAR, customer_city VARCHAR, customer_state VARCHAR)
- ORDERS_TABLE (order_id VARCHAR, customer_id, order_status VARCHAR, order_purchase_timestamp TIMESTAMP, order_approved_at TIMESTAMP, order_delivered_carrier_date TIMESTAMP, order_delivered_customer_date TIMESTAMP, order_estimated_delivery_date TIMESTAMP)
- ORDER_ITEMS_TABLE (order_item_id VARCHAR, order_id VARCHAR, product_id, seller_id, shipping_limit_date TIMESTAMP, price NUMERIC, freight_value NUMERIC)
- ORDER_PAYMENTS_TABLE (order_id VARCHAR, payment_sequential INT, payment_type VARCHAR, payment_installments INT, payment_value NUMERIC)
- ORDER_REVIEWS_TABLE (review_id VARCHAR, order_id VARCHAR, review_score INT, review_comment_title VARCHAR, review_comment_message VARCHAR, review_creation_date TIMESTAMP, review_answer_timestamp TIMESTAMP))
- GEOLOCATION_TABLE (geolocation_zip_code_prefix VARCHAR primary key, geolocation_lat VARCHAR, geolocation_lng VARCHAR, geolocation_city VARCHAR, geolocation_state VARCHAR)
- PRODUCT_SELLER_TABLE (product_id VARCHAR primary key, seller_id VARCHAR)

VII. Descriptions of Relations and Their Attributes:

The following is a description of each relation and its attributes of the dataset we used for inventory management and online order tracking:

- CUSTOMERS_TABLE: The customers who place orders on the e-commerce platform are represented by this relation. This table's attributes are as follows:

  - customer_id: Each customer's unique identifier.
  - customer_unique_id: An identifier that is unique to each customer and remains the same across all orders.
  - customer_fname: The first name of the customer.
  - customer_lname: The last name of the customer.
  - customer_zip_code_prefix: The customer's location's zip code prefix.
  - customer_city: The city in which the customer resides.
  - customer_state: The state in which the customer resides.
  - customer_email: The email of the customer.
  - customer_DOB: The date of birth of the customer.
  - customer_age: The age of the customer.
  - customer_gender: The gender of the customer.
- ORDERS_TABLE: This relation represents customer orders placed on the e-commerce platform. This table's attributes are as follows:
  - order_id: The order's unique identifier.
  - customer_id: The unique identifier for the customer who made the purchase.
  - order_purchase_timestamp: The date and time of the order.
  - order_approved_at: The date and time that the e-commerce platform approved the order.
  - order_delivered_carrier_date: The date and time when the order was delivered to the carrier.
  - order_delivered_customer_date: The time and date the order was delivered to the customer.
  - order_estimated_delivery_date: The order's expected delivery date.
- ORDER_ITEMS_TABLE: This relation represents the items in every customer order. This table's attributes are:
  - order_item_id: The order item's unique identifier.
  - order_id: The order to which the item belongs.
  - product_id: A unique identifier for each product.
  - seller_id: The unique identifier for the seller who sold the item.
  - shipping_limit_date: The deadline for the seller to ship the product.
  - price: The price of the product.
  - freight_value: The shipping cost of the product.
- PRODUCTS_TABLE: This relation represents the e-commerce platform's products. This table's attributes are as follows:
  - product_id: Each product's unique identifier.
  - product_category_name: The category to which the product belongs.
  - product_name_length: The length of the product name.
  - product_description_length: The length of the product description.
  - product_manufacturing_date: The manufacturing date of the product.
  - product_expiry_date: The expiry date of the product.
  - product_photos_qty: The number of product photos available.
  - product_weight_g: The product's weight in grams.
  - product_length_cm: The product's length in centimeters.
  - product_height_cm: The product's height in centimeters.
  - product_width_cm: The product's width in centimeters.
- SELLERS_TABLE: This relation represents the sellers who sell products on the e-commerce platform. This relation has the following attributes:
  - seller_id: A unique identifier for each seller.
  - seller_fname: The first name of the seller.
  - seller_lname: The last name of the seller.
  - seller_DOB: The date of birth of the seller.
  - seller_age: The age of the seller.
  - seller_zip_code: The zip code of the seller.
  - seller_zip_code_prefix: The seller's zip code prefix.
  - seller_city: The city in which the seller is located.
  - seller_state: The seller's home state.
  - seller_gender: The gender of the seller.
  - seller_email: The email of the seller.
- ORDER_PAYMENTS_TABLE: This relation represents the payments made for each order. This relation has the following attributes:
  - order_id: The identifier for the order for which payment was made.
  - payment_sequential: The order's payment sequence number.
  - payment_type: The type of payment used.
  - payment_installments: The number of installments in which the payment was made.
  - payment_value: The payment's monetary value.
- ORDER_REVIEWS_TABLE: This table contains information about customer reviews for orders they placed, as well as any responses from sellers. This table has the following attributes:
  - review_id: A unique identifier for the review
  - order_id: A unique identifier for the review order
  - review_score: The rating given by the customer in the review, ranging from 1 (worst) to 5 (best)
  - review_comment_title: The title of the review, which summarizes the customer's experience.
  - review_comment_message: The text of the review that describes the customer's experience.
  - review_creation_date: The date and time when the customer created the review
  - review_answer_timestamp: The date and time when the seller responded to the review, if applicable.
- GEOLOCATION_TABLE: This table includes geographical information for each zip code prefix in an area, such as latitude and longitude coordinates, as well as the city and state. This table has the following attributes:

  - geolocation_zip_code_prefix: The first five digits of the zip code that identify the region in which it is located.
  - geolocation_lat: The latitude of the zip code's associated location.
  - geolocation_lng: The longitude of the location associated with the zip code.
  - geolocation_city: The name of the city associated with the zip code.
  - geolocation_state: Abbreviation for the state associated with the zip code.

- PRODUCT_SELLER_TABLE: This table maps the product id to corresponding seller id
  - product_id: product id of products
  - seller_id: seller id of corresponding sellers

VIII. RELATION BETWEEN TABLES

The relations between tables that exist in the dataset used for this system are:

- The ORDERS table has a one-to-many relationship with the ORDER_ITEMS table, as each order can have multiple items.
- The ORDER_ITEMS table has a many-to-one relationship with the ORDERS table, as each item can only belong to one order.
- The ORDER_ITEMS table has a many-to-one relationship with the PRODUCTS table, as each item is associated with one product.
- The ORDER_ITEMS table has a many-to-one relationship with the SELLERS table, as each item is sold by one seller.
- The SELLERS table has a one-to-many relationship with the ORDERS table, as each seller can have multiple orders.
- The CUSTOMERS table has a one-to-many relationship with the ORDERS table, as each customer can have multiple orders.
- The ORDER_REVIEWS table has a many-to-one relationship with the ORDERS table, as each review is associated with one order.
- The GEOLOCATION table is not directly related to any other table, but it can be used to obtain geographic information for orders and customers based on their zip codes.
- The PRODUCT_SELLER_TABLE has many to one mapping to the SELLERS table, as multiple products can belong to each seller

IX. TABLES AND KEYS

Tables	Keys
CUSTOMERS_TABLE	Primary key: **customer_id** Foreign key: customer_zip_code_prefix Foreign key references to GEOLOCATION_TABLE Not NULL: customer_unique_id, customer_fname, customer_DOB, customer_email, customer_gender
ORDERS_TABLE	Primary key: **order_id** Foreign key: customer_id Foreign key references to CUSTOMERS_TABLE Not NULL: order_status, order_purchase_timestamp
PRODUCTS_TABLE	Primary key: **product_id**
SELLERS_TABLE	Primary key: **seller_id** Foreign key: seller_zip_code_prefix Foreign key references to GEOLOCATION_TABLE Not NULL: seller_fname, seller_email, seller_gender, seller_dob
ORDER_PAYMENTS_TABLE	Foreign key: order_id Foreign key references to ORDERS_TABLE Not NULL: payment_sequential, payment_type, payment_installments, payment_value
ORDER_ITEMS_TABLE	Primary key: **order_item_id** Foreign key: order_id, product_id, seller_id Foreign key references to ORDERS_TABLE Foreign key references to PRODUCTS_TABLE Foreign key references to SELLERS_TABLE Not NULL: price
ORDER_REVIEWS_TABLE	Foreign key: order_id Foreign key references to ORDERS_TABLE Not NULL: review_score, review_creation_date
GEOLOCATION_TABLE	Primary key: **geolocation_zip_code_prefix**
PRODUCT_SELLER_TABLE	Foreign key: product_id Foreign key: seller_id

X. FUNCTIONAL DEPENDENCIES

The functional dependencies for each table in the database schema are:

- SELLERS TABLE:
  seller_id → seller_fname, seller_lname, email, gender, date_of_birth, age, seller_zip_code_prefix, seller_city, seller_state
- PRODUCTS_TABLE:
  product_id → product_category_name, product_name_length, product_description_length, product_manufacturing_date, product_expiry_date, product_photos_qty, product_weight_g, product_length_cm, product_height_cm, product_width_cm, manufacturing_date, expiry_date
- CUSTOMERS_TABLE:
  customer_id → customer_unique_id, customer_fname, customer_lname, customer_DOB, customer_email, customer_zip_code_prefix, customer_gender, customer_age, customer_zip_code_prefix, customer_city, customer_state
- ORDERS_TABLE:
  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_ITEMS_TABLE:
  order_id → order_item_id, product_id, seller_id, shipping_limit_date, price, freight_value

- ORDER_PAYMENTS_TABLE:
  order_id → payment_sequential, payment_type, payment_installments, payment_value
- ORDER_REVIEWS_TABLE:
  review_id → order_id, review_score, review_comment_title, review_comment_message, review_creation_date, review_answer_timestamp
- GEOLOCATION_TABLE:
  geolocation_zip_code_prefix → geolocation_lat, geolocation_lng, geolocation_city, geolocation_state
- PRODUCT_SELLER_TABLE:
  product_id → seller_id

The Functional Dependencies listed above for our relatives are all non-trivial FDs. Additionally, the super keys of that table are on the left side of each of these non-trivial functional dependencies. Therefore, we can state that BCNF is the relational schema used by all of the tables in our database.

XI. Database Implementation

A. Creating Tables

```
create table sellers_table(seller_id VARCHAR(100) primary key,
                           seller_fname VARCHAR(100),
                           seller_lname VARCHAR(100),
                           seller_dob VARCHAR(10),
                           seller_age INT,
                           seller_email VARCHAR(60),
                           seller_gender VARCHAR(10),
                           seller_zip_code_prefix VARCHAR(10),
                           seller_city VARCHAR(60),
                           seller_state VARCHAR(10))

select count(*) from sellers_table
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 58 msec.
```

Fig. 2. Creation of Sellers table

```
create table products_table(product_id VARCHAR(100) primary key,
                            product_category_name VARCHAR(100),
                            product_name_length INT,
                            product_description_length INT,
                            product_photos_qty INT,
                            product_weight_g INT,
                            product_length_cm INT,
                            product_height_cm INT,
                            product_width_cm INT,
                            product_manufacturing_date VARCHAR(30),
                            product_expiry_date VARCHAR(30))
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 66 msec.
```

Fig. 3. Creation of products table

```
create table customers_table(customer_id VARCHAR(100) primary key,
                             customer_unique_id VARCHAR(100),
                             customer_fname VARCHAR(25),
                             customer_lname VARCHAR(25),
                             customer_DOB VARCHAR(10),
                             customer_age INT,
                             customer_email VARCHAR(50),
                             customer_gender VARCHAR(10),
                             customer_zip_code_prefix VARCHAR(10),
                             customer_city VARCHAR(40),
                             customer_state VARCHAR(10))
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 101 msec.
```

Fig. 4. Creation of customers table

```
create table orders_table(order_id VARCHAR(100) primary key,
                          customer_id VARCHAR(100),
                          order_status VARCHAR(100),
                          order_purchase_timestamp TIMESTAMP,
                          order_approved_at TIMESTAMP,
                          order_delivered_carrier_date TIMESTAMP,
                          order_delivered_customer_date TIMESTAMP,
                          order_estimated_delivery_date TIMESTAMP,
                          FOREIGN KEY (customer_id) REFERENCES customers_table(customer_id))
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 191 msec.
```

Fig. 5. Creation of orders table

```
create table order_items(order_id VARCHAR(100),
                         order_item_id INT,
                         product_id VARCHAR(100),
                         seller_id VARCHAR(100),
                         shipping_limit_data TIMESTAMP,
                         price NUMERIC(20,5),
                         freight_value NUMERIC(20,5),
                         FOREIGN KEY (order_id) REFERENCES orders_table(order_id),
                         FOREIGN KEY (seller_id) REFERENCES sellers_table(seller_id),
                         FOREIGN KEY (product_id) REFERENCES products_table(product_id))
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 73 msec.
```

Fig. 6. Creation of order_items table

```
create table order_reviews_table(review_id VARCHAR(100),
                                 order_id VARCHAR(100),
                                 review_score INT,
                                 review_comment_title VARCHAR(30),
                                 review_comment_message VARCHAR(500),
                                 review_creation_date TIMESTAMP,
                                 review_answer_timestamp TIMESTAMP,
                                 FOREIGN KEY (order_id) REFERENCES orders_table(order_id))
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 77 msec.
```

Fig. 7. Creation of order_reviews table

```
create table order_payments_table(order_id VARCHAR(100),
                                  payment_sequential INT,
                                  payment_type VARCHAR(30),
                                  payment_installments INT,
                                  payment_value NUMERIC(20,5),
                                  FOREIGN KEY (order_id) REFERENCES orders_table(order_id))
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 49 msec.
```

Fig. 8. Creation of order_payments table

```
create table geolocation_table(geolocation_zip_code_prefix INT,
                               geolocation_lat VARCHAR(300),
                               geolocation_lng VARCHAR(300),
                               geolocation_city VARCHAR(200),
                               geolocation_state VARCHAR(100))
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 54 msec.
```

Fig. 9. Creation of geolocation table

```
create table product_seller_table(product_id VARCHAR(100) NOT NULL,
                                  seller_id VARCHAR(100) NOT NULL,
                                  FOREIGN KEY (seller_id) REFERENCES sellers_table(seller_id),
                                  FOREIGN KEY (product_id) REFERENCES products_table(product_id))
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 53 msec.
```

Fig. 10. Creation of product_sellers table

B. Loading data to PgAdmin4

For loading from csv files to the postgreSQL for each dataset we have created table for a well-defined structure to store the CSV file data and imported the CSV file of that dataset containing data into PostgreSQL.
For creating the table structure for each table we have executed the above mentioned create queries for the tables.
For importing the csv files to postreSQL for all the tables we have imported as shown in the figure below.

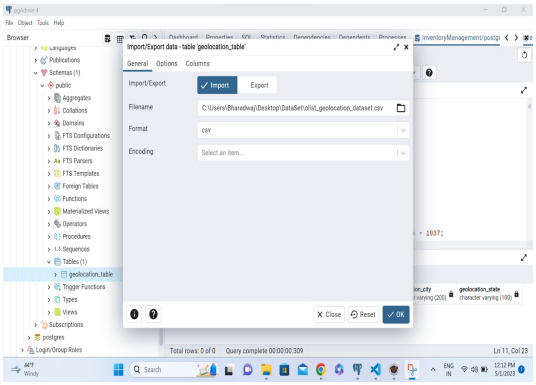


Fig. 11. Importing a table's csv file to postgreSQL

C. Insertion

```
INSERT INTO sellers_table(seller_id,
                          seller_fname,
                          seller_lname,
                          seller_dob,
                          seller_age,
                          seller_email,
                          seller_gender,
                          seller_zip_code_prefix,
                          seller_city,
                          seller_state)
VALUES (35436, 'johny', 'depp', '04/21/96', '89', 'johny@gmail.com', 'male', '9999', 'MUMBAI', 'MAHARASTRA')

SELECT * FROM sellers_table where seller_city = 'MUMBAI';
```

Data Output | Messages | Notifications

	seller_id [PK] character varying (100)	seller_fname character varying (100)	seller_lname character varying (100)	seller_dob character varying (10)	seller_age integer	seller_email character varying (60)	seller_gender character varying (10)	seller_zip_cod character varyi
1	35436	johny	depp	04/21/96	89	johny@gmail.com	male	9999

Fig. 12. Inserting a record into Sellers table

Fig. 13. Inserting a record into products table

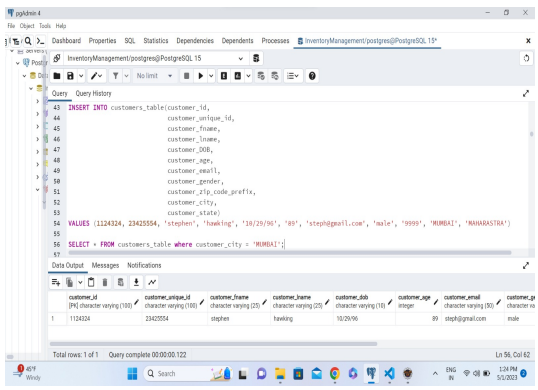


Fig. 14. Inserting a record into Customers table

Fig. 15. Inserting a record into Orders table

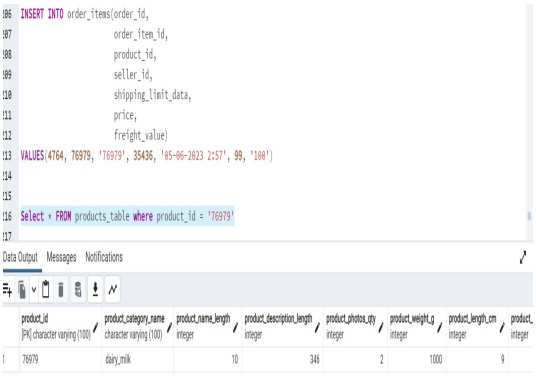


Fig. 16. Inserting a record into order_items table

Fig. 17. Inserting a record into order_reviews table

Fig. 18. Inserting a record into order_payments table

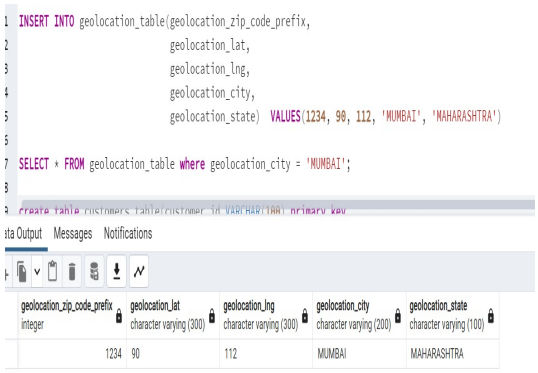


Fig. 19. Inserting a record into geolocation table

Fig. 20. Inserting a record into product_seller table

D. Updating tables

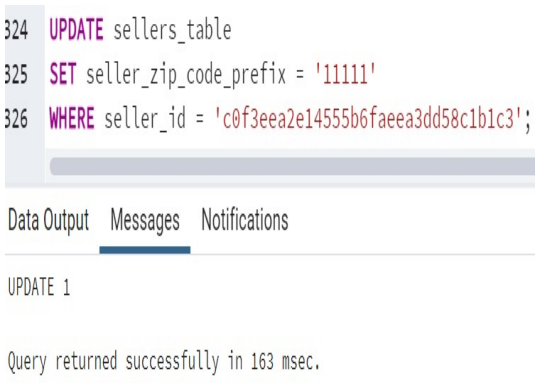


Fig. 21. Update command on sellers table

Fig. 22. Update command on products table

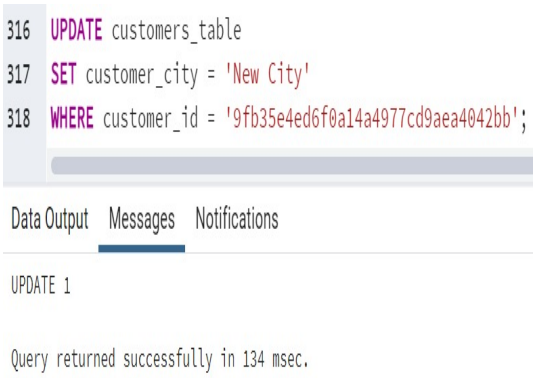


Fig. 23. Update command on customers table

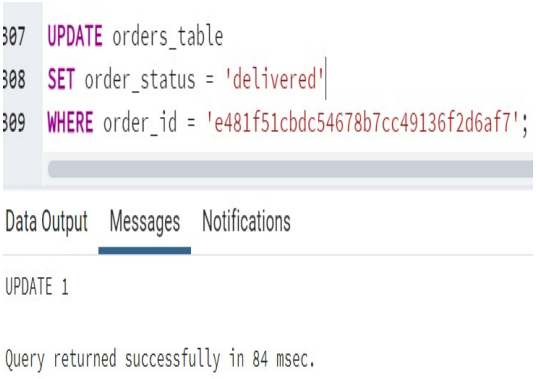


Fig. 24. Update command on orders table

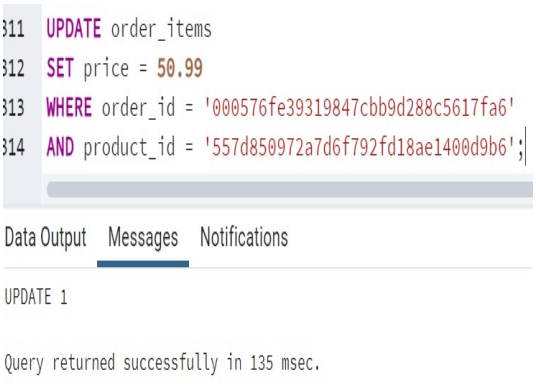


Fig. 25. Update command on order_items table

E. Deletion

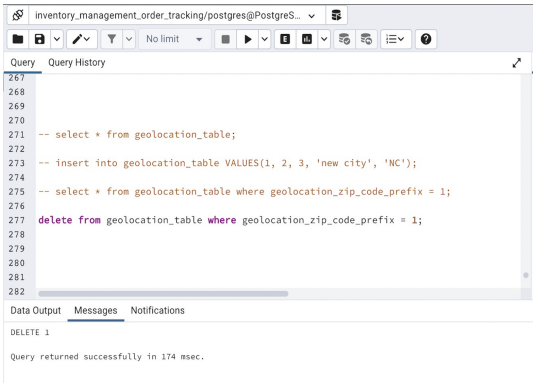


Fig. 26. Delete command on table

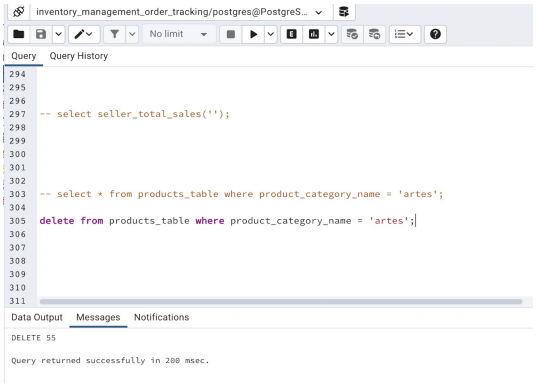


Fig. 27. Delete command on table

F. SQL Queries

1) Query to find the top 10 customers who have made the highest number of purchases:
2) Query to find the average rating for each product category
3) Query to find the number of unique customers who have made a purchase in each month of 2018

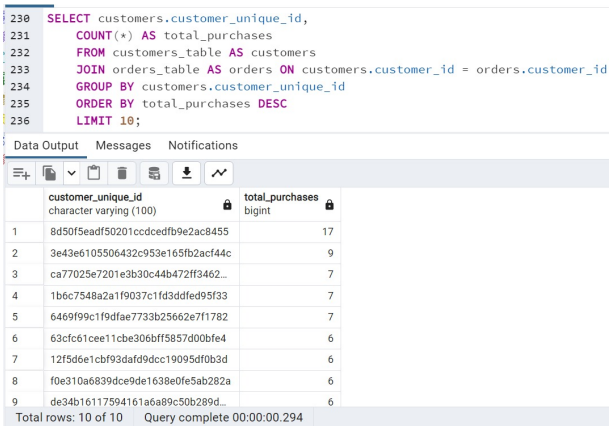


Fig. 28. Select query 1

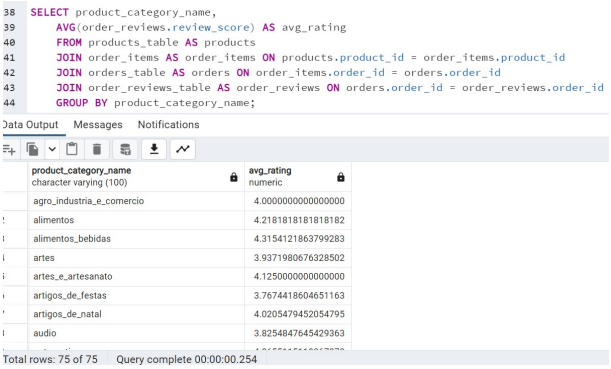


Fig. 29. Select query 2

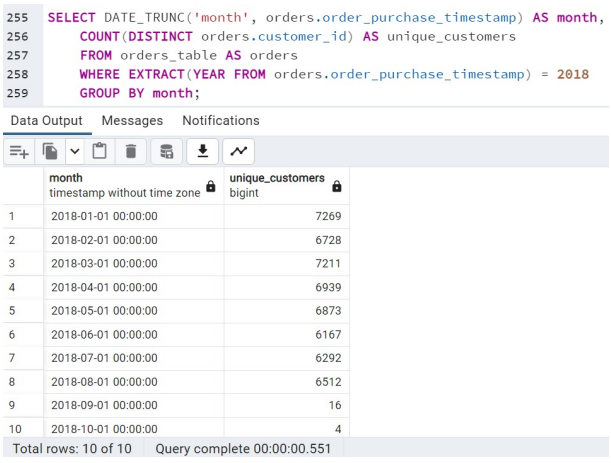


Fig. 30. Select query 3

```
SELECT customers.customer_city,
    AVG(order_items.price + order_items.freight_value) AS avg_order_value
    FROM customers_table AS customers
    JOIN orders_table AS orders ON customers.customer_id = orders.customer_id
    JOIN order_items AS order_items ON orders.order_id = order_items.order_id
    GROUP BY customers.customer_city
    ORDER BY avg_order_value DESC
    LIMIT 10;
```

Data Output | Messages | Notifications

customer_city character varying (40)	avg_order_value numeric
pianco	2324.9900000000000000
nova esperanca do piria	2252.6600000000000000
engenheiro navarro	2106.5500000000000000
agrestina	2066.3400000000000000
mariental	1867.8500000000000000
loreto	1643.6400000000000000
ibitita	1534.5800000000000000
pirpirituba	1372.2500000000000000

Total rows: 10 of 10 | Query complete 00:00:00.260

Fig. 31. Select query 4

4) Query to find the top 10 cities with the highest average order value
5) Query to find the top 5 products that have been returned the most:

```
SELECT order_items.product_id, products.product_category_name,
    COUNT(*) AS return_count
    FROM order_items AS order_items
    JOIN orders_table AS orders ON order_items.order_id = orders.order_id
    JOIN order_reviews_table AS order_reviews ON orders.order_id = order_reviews.order_id
    JOIN products_table AS products ON order_items.product_id = products.product_id
    WHERE order_reviews.review_comment_title LIKE '%devolução%'
    GROUP BY order_items.product_id, products.product_category_name
    ORDER BY return_count DESC
    LIMIT 5;
```

Data Output | Messages | Notifications

product_id character varying (100)	product_category_name character varying (100)	return_count bigint
89b121bee266dcd25688a1ba72eefb61	informatica_acessorios	2
ea3681fd0335adf45b71f1145ac562c7	informatica_acessorios	2
0798196b28c04c8e543322896c8829...	informatica_acessorios	1
2b4042dfabc3fe9d3f16951a8e1a3b70	utilidades_domesticas	1
c1e8014cae93306629b1de89dfa5a1bb	relogios_presentes	1

Total rows: 5 of 5 | Query complete 00:00:00.125

Fig. 32. Select query 5

6) Query to find the average time it takes for a customer to receive their order after it has been shipped

```
SELECT AVG(EXTRACT(EPOCH FROM
    (orders.order_delivered_customer_date - orders.order_delivered_carrier_date))) / 86400 AS avg_delivery_time_days
    FROM orders_table AS orders
    WHERE
    orders.order_status = 'delivered' AND
    orders.order_delivered_customer_date IS NOT NULL AND
    orders.order_delivered_carrier_date IS NOT NULL
```

Data Output | Messages | Notifications

avg_delivery_time_days numeric
9.3302970579831249

Fig. 33. Select query 6

7) Query to find the top 10 categories that have the highest number of orders:
8) Query to find top 5 sellers who have the highest percentage of orders delivered on time (i.e., the order was delivered on or before the estimated delivery date)
9) Query to find the top 10 customers who have the highest average rating for products they have reviewed, along with the number of reviews they have written.

```
SELECT products.product_category_name,
    COUNT(DISTINCT orders.order_id) AS order_count
    FROM orders_table AS orders
    JOIN order_items AS order_items ON orders.order_id = order_items.order_id
    JOIN products_table AS products ON order_items.product_id = products.product_id
    GROUP BY products.product_category_name
    ORDER BY order_count DESC
    LIMIT 10;
```

Data Output | Messages | Notifications

product_category_name character varying (100)	order_count bigint
cama_mesa_banho	9417
beleza_saude	8836
esporte_lazer	7720
informatica_acessorios	6689
moveis_decoracao	6449
utilidades_domesticas	5884
relogios_presentes	5624
telefonia	4199

Total rows: 10 of 10 | Query complete 00:00:01.433

Fig. 34. Select query 7

```
SELECT  s.seller_id,
    COUNT(CASE WHEN o.order_delivered_customer_date <= o.order_estimated_delivery_date THEN 1 END) /
    COUNT(*) AS on_time_delivery_rate
    FROM sellers_table AS s
    JOIN order_items AS oi ON s.seller_id = oi.seller_id
    JOIN orders_table AS o ON oi.order_id = o.order_id
    GROUP BY s.seller_id
    ORDER BY on_time_delivery_rate DESC
    LIMIT 5;
```

Output | Messages | Notifications

seller_id [PK] character varying (100)	on_time_delivery_rate bigint
fc38b5dceee1a730fad8853453437fbd	1
c53bcd3be457a342a97e39e5a9f0be22	1
6d2f2e3b539480db1e0842b3a4e32e...	1
a61cc0479330839 5a8408071043651...	1
5def4c3732941a971cba8fdee992ede1	1

Fig. 35. Select query 8

```
SELECT r.order_id,
    COUNT(r.review_id) AS num_reviews,
    AVG(r.review_score) AS avg_rating
    FROM order_reviews_table AS r
    GROUP BY r.order_id
    ORDER BY avg_rating DESC
    LIMIT 10;
```

ata Output | Messages | Notifications

order_id character varying (100)	num_reviews bigint	avg_rating numeric
0006ec9db01a64e59a68b2c340bf65a7	1	5.0000000000000000
0009792311464db532ff765bf7b182ae	1	5.0000000000000000
00061f2a7bc09da83e415a52dc8a4af1	1	5.0000000000000000
00063b381e2406b52ad429470734ebd5	1	5.0000000000000000
000229ec398224ef6ca0657da4fc703e	1	5.0000000000000000
0008288aa423d2a3f00fcb17cd7d8719	1	5.0000000000000000
000576fe39319847cbb9d288c5617fa6	1	5.0000000000000000
00010242fe8c5a6d1ba2dd792cb16214	1	5.0000000000000000
00042b26cf59d7ce69dfabb4e55b4fd9	1	5.0000000000000000

otal rows: 10 of 10 | Query complete 00:00:01.297

Fig. 36. Select query 9

XII. QUERY EXECUTION ANALYSES

A. *TRIGGER COMMAND*

In some cases when one table in the database is updated with insert, update or delete command we also need to update a related table. In such cases trigger command with function implementation helps carry out the changes in related tables. In our case, when a successful payment is made, the order_payments_table has a record inserted, which upon insertion triggers and update on the order_status column of orders_table

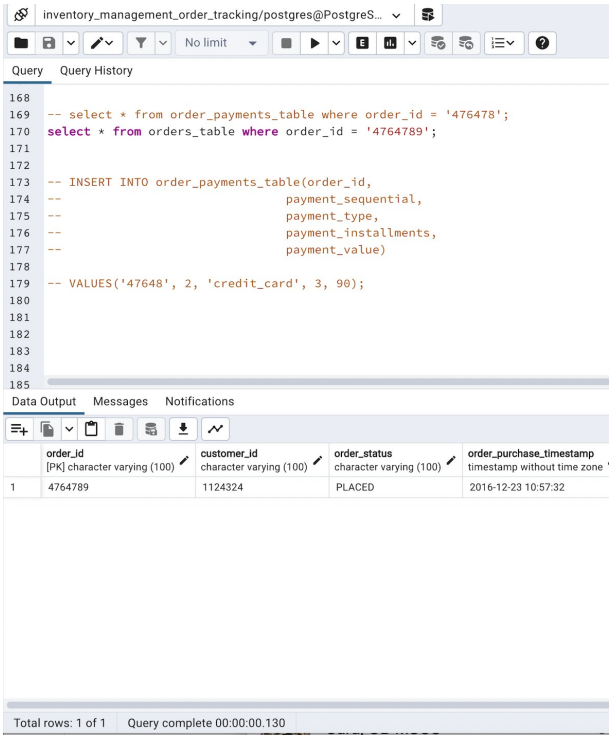


Fig. 37. Trigger Command

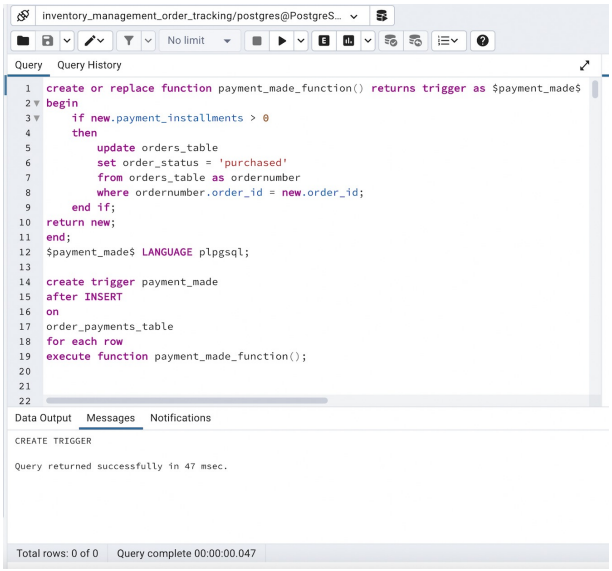


Fig. 38. Trigger Command

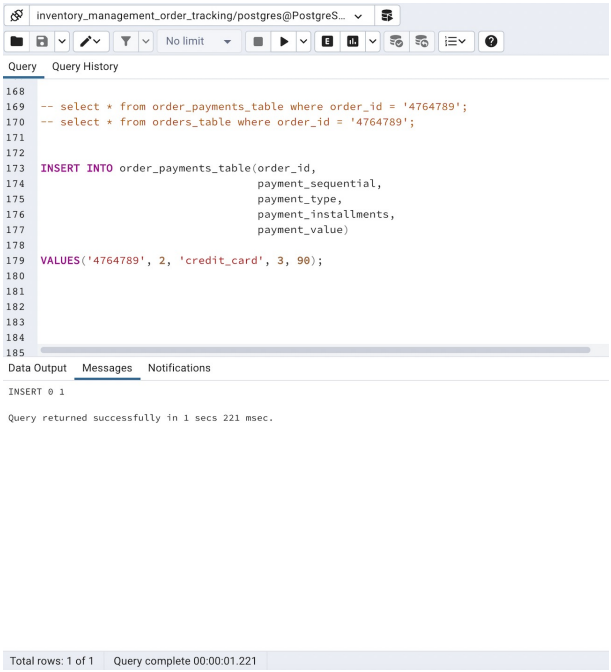


Fig. 39. Trigger Command

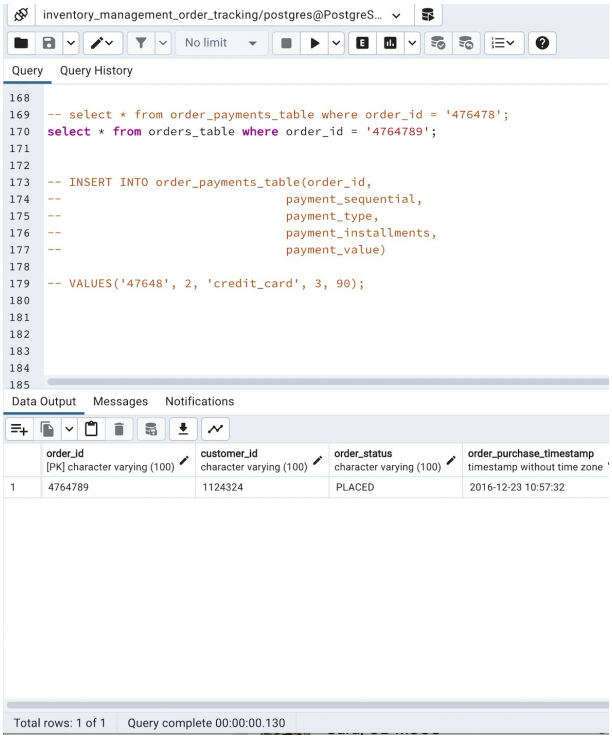


Fig. 40. Trigger Command

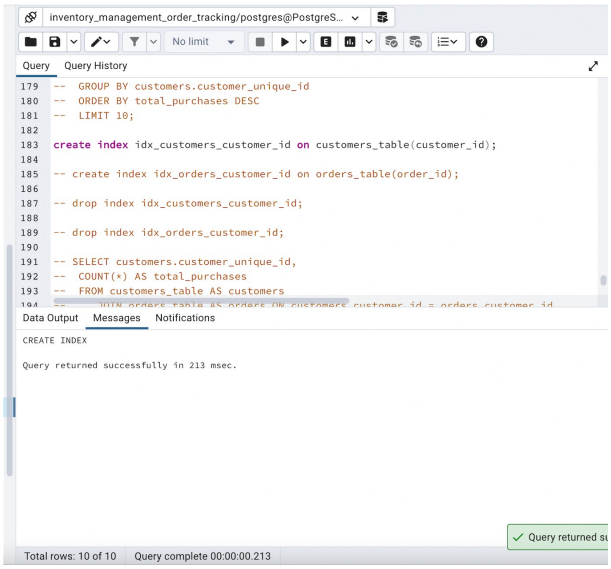


Fig. 42. Indexing

B. *INDEXING*

While running some complex select queries involving one or more databases with huge amount of data, we noticed a larger amount of time for query execution. In the example given, we run a select query involving a join operation on orders_table and customers_table. Upon adding an index to order_id in orders_table we see a reduction in the query execution time. Upon adding an index to customer_id in customers_table also we see a further reduction in the query execution time.

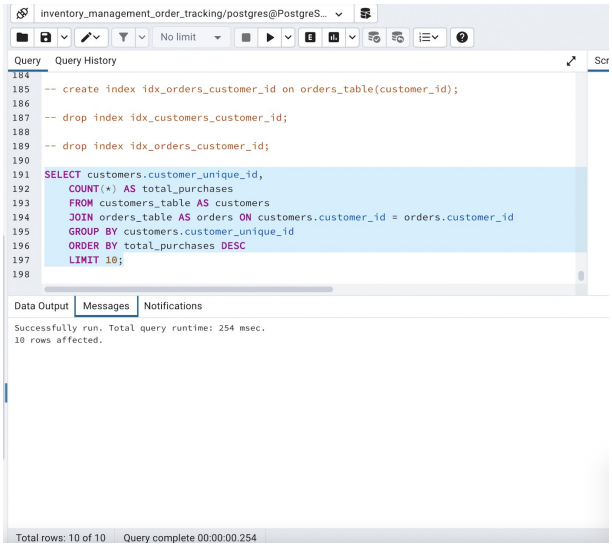


Fig. 41. Indexing

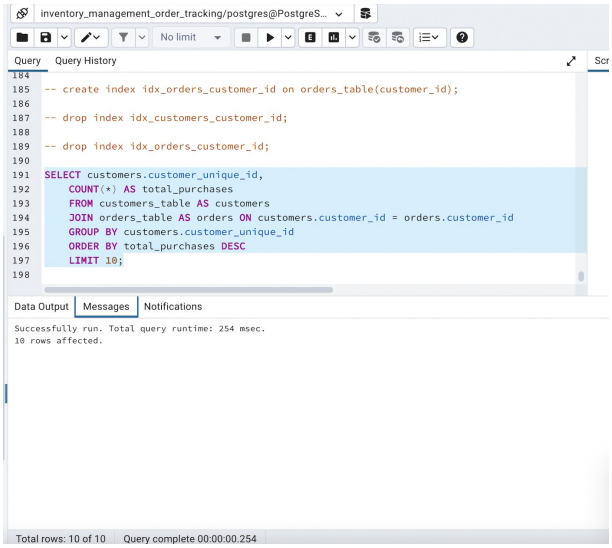


Fig. 43. Indexing

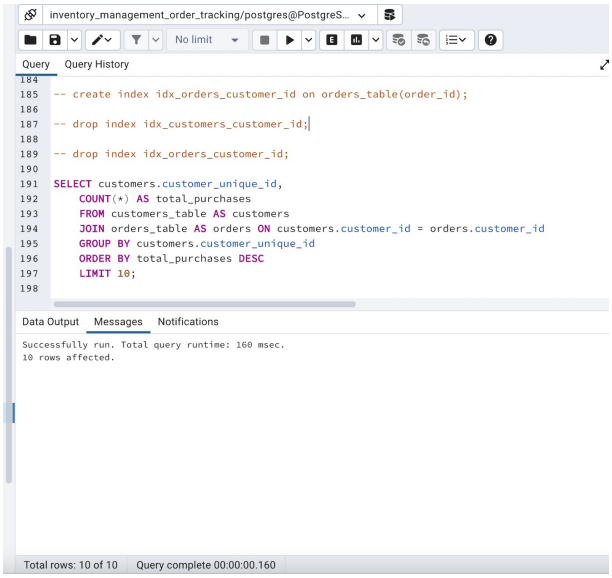


Fig. 44. Indexing

C. ADDING FUNCTIONS

In certain tables, certain calculations help in data analysis. We can add functions to carry out such operations that are required repeatedly for different values. In the example, we have shown a function which calculates the most up to date total sales amount of a particular seller.

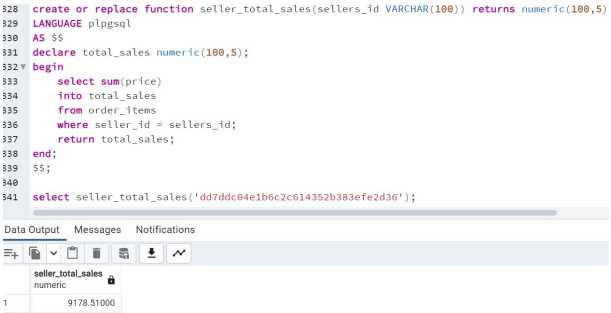


Fig. 45. Function Implementation

XIII. GUI

This application has been developed using JSP(Jakarta Server Pages) for the frontend and Java for the backend using postgreSQL.

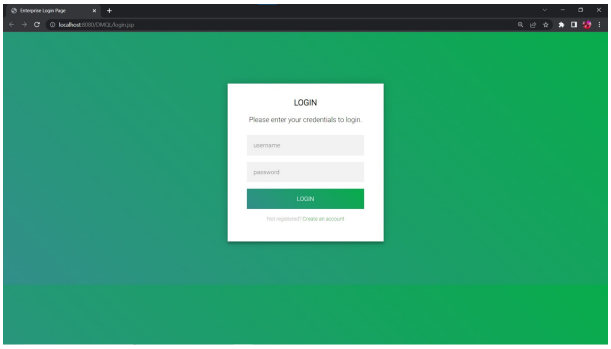


Fig. 46. Existing customer login

Fig. 47. New customer Registration

Fig. 48. Actions that can be performed

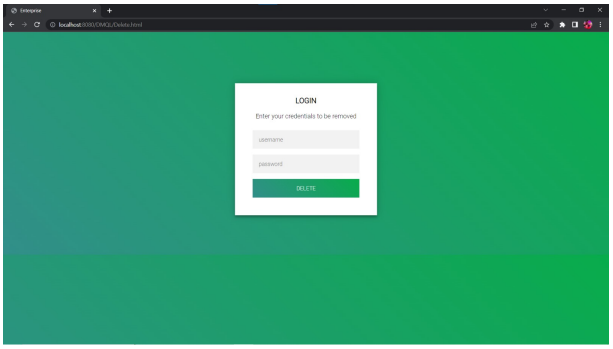
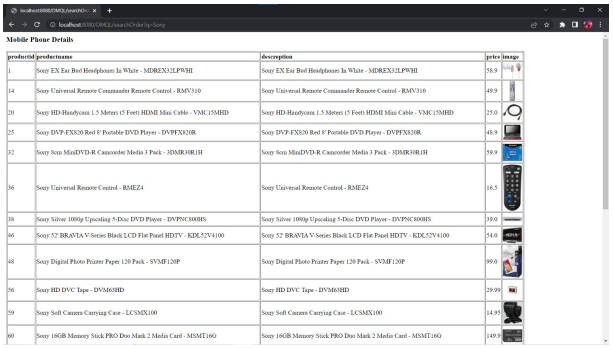


Fig. 49. Deleting user from the database

Fig. 50. Customers searching for the products to purchase



Mobile Phone Details

productid	productname	description	price	image
1	Sony EX Ear Bud Headphones In White - MDREX32LPWH	Sony EX Ear Bud Headphones In White - MDREX32LPWH	58.9	
14	Sony Universal Remote Commander Remote Control - RMV310	Sony Universal Remote Commander Remote Control - RMV310	49.9	
20	Sony HD-Handycom 1.5 Meters (5 Feet) HDMI Mini Cable - VMC15MHD	Sony HD-Handycom 1.5 Meters (5 Feet) HDMI Mini Cable - VMC15MHD	25.0	
25	Sony DVP-FX820 Red 8' Portable DVD Player - DVPFX820R	Sony DVP-FX820 Red 8' Portable DVD Player - DVPFX820R	48.9	
32	Sony 8cm MiniDVD-R Camcorder Media 3 Pack - 3DMR30R1H	Sony 8cm MiniDVD-R Camcorder Media 3 Pack - 3DMR30R1H	59.9	
36	Sony Universal Remote Control - RMEZ4	Sony Universal Remote Control - RMEZ4	16.5	
38	Sony Silver 1080p Upscaling 5-Disc DVD Player - DVPNC800HS	Sony Silver 1080p Upscaling 5-Disc DVD Player - DVPNC800HS	39.0	
46	Sony 52' BRAVIA V-Series Black LCD Flat Panel HDTV - KDL52V4100	Sony 52' BRAVIA V-Series Black LCD Flat Panel HDTV - KDL52V4100	54.0	
48	Sony Digital Photo Printer Paper 120 Pack - SVMF120P	Sony Digital Photo Printer Paper 120 Pack - SVMF120P	99.0	
56	Sony HD DVC Tape - DVM63HD	Sony HD DVC Tape - DVM63HD	29.99	
59	Sony Soft Camera Carrying Case - LCSMX100	Sony Soft Camera Carrying Case - LCSMX100	14.95	
60	Sony 16GB Memory Stick PRO Duo Mark 2 Media Card - MSMT16G	Sony 16GB Memory Stick PRO Duo Mark 2 Media Card - MSMT16G	149.9	

Fig. 51. Search images for the word 'Sony'

REFERENCES

1) https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce
2) https://www.w3schools.com/sql/
3) https://www.enterprisedb.com/downloads/postgres-postgresql-downloads
4) https://extendsclass.com/csv-generator.html
5) https://www.eclipse.org/downloads/packages/release/neon/3/eclipse-ide-java-ee-developers
6) https://tomcat.apache.org/download-90.cgi
7) https://www.postgresqltutorial.com/postgresql-jdbc/connecting-to-postgresql-database/