E-Commerce Database Management

1st Jahnavi Kollu
Dept. of Computer Science and Engg
University at Buffalo, NY
Buffalo, New York, USA
UBID: 50591633
UBIDname: jkollu
jkollu@buffalo.edu

2nd Likhitha Kodali
Dept. of Computer Science and Engg
University at Buffalo, NY
Buffalo, New York, USA
UBID: 50600456
UBIDname: lkodali
lkodali@buffalo.edu

3rd Sanjay Kumar Mandru
Dept. of Computer Science and Engg
University at Buffalo, NY
Buffalo, New York, USA
UBID: 50598178
UBIDname: smandru
smandru@buffalo.edu

***Abstract*—This project focuses on designing a relational SQL database for optimizing and organizing the Brazilian E-Commerce Public Dataset by Olist, which contains over 100,000 orders from multiple Brazilian marketplaces. The dataset includes key entities such as customers, orders, products, sellers, reviews, and payments, all linked together to provide valuable insights into e-commerce business operations. By employing entity-relationship (ER) modeling and schema normalization, the database ensures efficient data retrieval and storage. Additionally, indexing strategies and optimized queries are utilized to improve query performance, while stored procedures and triggers automate data validation and enforce business rules. This system enables real-time order monitoring, customer behavior analysis, and seller performance management, thereby facilitating more effective decision-making for e-commerce platforms.**

***Index Terms*—E- Commerce, Databases , Tables**

I. Need of database

The E Commerce dataset with its millions of records about orders and other e-commerce elements requires a specific framework and efficient retrieval system because standard spreadsheet tools such as Excel are insufficient for this scale of data storage requirements. A relational SQL database proves to be the best system for managing this type of data based on following reasons:

1. Scalability and Performance A lack of handling capacity for large datasets makes Excel operate poorly and results in system breakdowns while using datasets with millions of rows. The indexing together with query optimization techniques in SQL databases deliver high performance to handle the efficient storage and retrieval of millions of records.

2. Data Integrity and Consistency Excel has limitations because the system depends on human error for data input and validation yet this approach endangers the system's data integrity through accidental entries of duplicates or missing or inconsistent records. A relational database makes use of primary keys together with foreign keys and constraints to enforce data integrity so users achieve both accuracy and consistency across multiple tables.

3. Handling Relationships Between Data (Normalization) A flat database structure in Excel prevents smooth maintenance of relationships between distinct data entities such as customers, orders and products or sellers. SQL databases implement relational models to handle efficient relationships between data entities between one and many entities as well as numerous entities thereby organizing data effectively.

4. Efficient Querying and Data Retrieval Excel has a major limitation in its manual operation of data search through filtering functions because this process becomes exceptionally slow when handling large databases. The SELECT query with JOIN constructs and filtering operations in SQL databases lets users get complex data retrieval in only seconds.

5. Multi-User Access and Security The lack of multi-user design in Excel causes file systems to become vulnerable to data conflicts as well as accidental overwrites and security threats. SQL databases deliver secure data access regulation through role-based access control (RBAC) while simultaneously managing several users who work in tandem.

6. Automation with Stored Procedures and Triggers The Excel system features limited automated processing through basic macros that must be operated by users manually. Database systems through SQL databases organize procedures and triggers which perform automatic functions for order status updates and sales computations in addition to data quality checking and verification processes free of human input.

7. Real-World Business Applications The efficiency of online platform transactions, order management and payment details tracking together with customer data evaluation requires database systems that utilize relational structures. An SQL database stands above Excel for large-scale data management because it delivers scalable performance alongside security and it handles extensive datasets.

Moving data from Excel to an SQL-based database represents an essential requirement for creating an efficient and reliable system to handle e-commerce data storage.

II. Background of the Problem and Objectives

Complexity of Managing E-Commerce Data: The online platform Olist connects thousands of users daily through transactions that happen among multiple sellers and customers with various products. This database presents the real-world details

essential for managing e-commerce through its Brazilian E-Commerce Public Dataset by Olist that features dynamic information about orders and payments along with product descriptions and customer data and seller performance and logistics details. Efficient operation of such vast volumes of data depends on a purpose-built system which should both organize and retrieve data smoothly with high analysis capability.

The main obstacle for e-commerce operations involves managing various integrated data structures. The entire order process includes multiple specific entities that participate in the transaction.

Customers (who place orders)
Sellers (who fulfill orders)
Products (sold across multiple categories)
Payments (with different methods and installment options)
Consumer reviews of orders serve to influence seller standings on review platforms.
Geolocation information serves both for delivery logistics and delivery tracking purposes.
Businesses experience severe difficulties with data consistency and retrieval efficiency as well as integrity challenges because they lack a relational database structure.

Challenges in Data Organization and Processing The unprocessed dataset exists in several separate CSV files which demands human assistance for extracting valuable information. The key challenges include:

Handling Many-to-Many Relationships: A customer order brings together multiple products that sellers from various sources provide. A single order can link to various installment payment methods which collectively charge a single customer. The correct linking process must establish relationships between reviews with their associated orders and customers. The complex data relationships need proper management through a structured SQL database system.

Data Redundancy and Consistency Issues: Normalizing seller details storage prevents data inconsistencies that result from maintaining seller details. Multiple duplicate records in different CSV files create issues with data integrity because they become unintegrated with other sources. Relational databases protect referential integrity through foreign key constraints as well as prevent duplicate entries from occurring.

Efficient Data Retrieval for Business Insights:

Businesses must evaluate order patterns together with customer activities and product effectiveness levels. The absence of optimized query mechanisms causes the process of retrieving dataset insights to run slowly and poorly. The ability of SQL databases to index and query data enables rapid performance during information retrieval.

Scalability and Performance Optimization:

The process of handling growing datasets becomes less efficient along with becoming harder to control through traditional storage methods. SQL handles scaling requirement by delivering three performance elements that maintain scalability along with indexing features as well as partitioning functions with query optimization capabilities.

E Commerce Business Requirements:

Business success depends heavily on providing real-time visibility into order statuses together with payment transactions and customer reviews to customers. Data accuracy stays stable thanks to automated validation solutions implemented through SQL constraints with triggers in place. A database requires optimal design to perform queries on customer purchasing data as well as refunds and seller analytics tracking functions.

Significance The efficient management of e-commerce data remains essential for three key functions within an organization: story reports and delivery timelines as well as shipment details need to be precisely monitored for order fulfillment operations.

Customer Satisfaction Retention: Ensuring quick resolution of complaints and analyzing customer feedback trends. The analysis of seller performance reveals high-performing sellers to the company to enhance both seller-buyer exchanges. Organizational decision-making and business intelligence functions enhance with quicker access to analytics about payments along with trends and sales reports alongside customer behavioral information.

III. Potential of project

1. Enhancing Data Organization Integrity

Ensuring proper data structure and keeping information intact stands as the main obstacle when managing e-commerce data. Various dependent components including orders, customers, products, payments, sellers and reviews make up the Brazilian E-Commerce Public Dataset by Olist. A clear management system for these interconnected entities prevents duplicate data together with inconsistency and decreases efficiency.

The project addresses this problem through: Structural design of relational database systems includes enabling data integrity by implementing primary keys and foreign keys as well as normalization rules. The database design ensures elimination of data duplication by separating customer, seller and product information into different tables that stay related to each other. The system upholds referential integrity standards to stop data errors which produce unconnected orders and payment records.

The method ensures data reliability while providing high-quality information needed by any efficient e-commerce platform.

2. Enabling Faster More Efficient Data Retrieval

The system enables efficient retrieval of data through its faster operation. Online companies depend on instantaneous data access for their strategic decision-making procedures. It takes excessive time to conduct queries on large Excel and CSV datasets due to their slow data retrieval speeds.

This project achieves data retrieval efficiency through these measures: The system implements indexing methods together with query optimization tools which allow businesses to get customer orders data as well as seller performance records and payment trend analytics immediately. Having established

SQL joins together with aggregation tools enables efficient extraction of complicated data connections such as purchase records from customers or product performance standings. What database administrators accomplish is the development of stored procedures and views because these items enable quick data access for common business inquiries which shortens the time needed to generate business insights.

The improved speed of business operations results from this advancement which enables stakeholders to receive their needed information instantly.

3. Improving E-Commerce Business Insights Decision Making The integration of a structured SQL database enhances decision-making and insight development for e-commerce businesses. A properly designed SQL database foundation enables e-commerce organizations to retrieve important business intelligence leading to the following outcomes: Market order data analysis enables companies to forecast sales patterns which helps them set inventory targets appropriately. Best-selling products together with categories should be identified for enhanced marketing strategy optimization. A complete overview of performance metrics should include seller delivery times along with ratings from customers and return rate information. Customer purchasing patterns enable the generation of customized product suggestions for individual buyers and shoppers.

Organizations obtain these business insights to drive data-based choices which optimize customer satisfaction alongside better seller alliances and stronger sales productivity.

4. Higher Order and Payment Processing Speed Requires Improvement Online platforms which handle massive order and payment operations need to maintain secure precise transaction management systems. This project contributes by: The necessary arrangement of order data should establish transparent connections between customers and sellers through proper mapping of order-party-payment data elements. A tracking system for installment payments which exists commonly in Brazilian markets enables the proper association of orders. Transactions must pass triggers to guarantee that payments avoid mistakes and duplications.

This addition avoids mistakes in order completion and stops financial inconsistencies that allow customers to experience a smooth transaction pathway.

5. Optimizing Logistics Delivery Tracking

Geolocation information within the dataset contains zip codes together with latitude-longitude mappings to enable these functions: The tracking system identifies customer delivery points to help businesses optimize their shipping network design for faster delivery times. Strategic business expansion occurs through the analysis of high-demand areas which lets companies choose their growth locations based on order frequency. The delivery success rates from specific locations help business teams resolve customer service problems promptly through location-based analysis.

Companies that utilize location information enable lower transportation expenses and better productivity and superior customer satisfaction.

Why This Contribution is Crucial

The database development represents more than collection and organization of data since it redefines e-commerce data management systems and their utilization practices. The structured database will: The system design should achieve scalability features that will accommodate more orders and customers. The storage infrastructure offers secure and dependable platform needed to safeguard customer information together with financial records. Businesses need real-time data for remaining competitive because the e-commerce industry operates at high speeds. The system should automate core operations while it minimizes two key issues in data management: manual work and human errors.

IV. Target User

Who Will Use the Database?

The SQL database created to handle e-commerce data acts as a vital information system for multiple participants in the online retail structure. The key users include:

1. E-Commerce Business Managers Business managers depend on this database to access information regarding sales patterns together with order processing performance level and customer behavior along with seller outcomes.

How They Use It: Users need access to reports which show the highest-selling products and most successful sellers. Foresight of order movement patterns helps business managers design stock management systems alongside marketing enterprise strategies. By using automated processes the system evaluates customer satisfaction ratings through both review data and order feedback systems. Business analysts generate reports using database queries which identify sellers with superior ratings and speediest delivery times to help the company advance its service quality.

2. Operations and Logistics Teams The company uses this platform to deliver efficient order processing and distribution services.

How They Use It: The management of pending, shipped and delivered orders allows operations teams to achieve timely order fulfillment. The delivery routes can be optimized and logistics costs can decrease through analysis of geolocation data. The company needs to locate both products with excessive returns and delivery locations that consistently generate delivery difficulties. Operational data shows the operations manager which specific sellers regularly deliver orders after deadlines so logistics teams work together to address this problem.

3. Customer Support Representatives Customer Support Representatives fulfill their duty by responding to inquiries from clients while monitoring delivery problems and dispute resolution tasks.

How They Use It: The information about payments and previous orders aids representatives during refund and exchange request resolution. Check seller response times and review history for complaint handling. To handle order-related questions support representatives need to check shipment locations and delivery forecasts. During the call the support

agent helps the customer who wants to know the status of their refund request. Through the database query the agent obtains current information about the processed refund status.

4. Sellers and Vendors Purpose: Tracking product sales, customer feedback, and payment settlements.

How They Use It: The system provides access to viewing product sales numbers together with revenue distribution information. The evaluation of customer reviews enables the enhancement of product quality alongside service operations. The system helps users track both unpaid transactions as well as successfully finalized payments. A seller consults the database to identify products generating the most returns thus enabling better product quality assessments. **5. Finance and Payment Processing Teams** Purpose: Ensuring secure payment transactions and financial accuracy.

How They Use It: The system must record every payment transaction alongside payment claims and disputes. Commercial teams should evaluate the popularity of different payment options while studying patterns of installment purchases made by customers. The platform should maintain payment verification between the sellers and the network system. The financial analyst uses the system's database to retrieve information about total credit card revenue from the recent quarter.

Who Will Administer the Database?

The Database Administrators (DBAs) along with IT teams will serve as the managers who keep the database operational. The key administrators include:

1. Database Administrator (DBA) Role: Ensuring database performance, security, and integrity.

Responsibilities: A backup system together with recovery protocols safeguards the database from data destruction. User access control allows authorized personnel to modify database data. The method used to optimize queries for achieving better results with big data. Within the real-life scenario the DBA prepares user roles coupled with permissions to enable financial analysts to view payment data without enabling them to adjust order records.

2. IT Infrastructure Team The IT Infrastructure Team handles server management responsibilities for the systems that support database hosting.

Responsibilities: The IT team must accomplish maximum database accessibility and scalability measures. IT staff should measure server consumption levels to improve operational efficiency.

3. Data Analysts and Developers Staff members in this team develop application-specific reports as well as business intelligence dashboards.

Responsibilities: Applicants must develop advanced SQL syntax to perform business analytics. The front-end applications will interface with the database through API-driven connectivity.

V. REAL LIFE SCENARIO

Users operating on e-commerce platforms that utilize SQL relational databases execute their orders for purchase. Three different sellers will send the three products included in the order which consists of a smartwatch and wireless earbuds and a laptop stand. The platform shows delivery estimates of five days as the user selects credit card payment through three installments.

The system creates and saves an order with a special order ID which connects the order to the user account after their purchase. The system connects each product to a particular seller for delivery support while storing payment details which detail the installment arrangements. The system updates the shipping table with the delivery time table information. The smartwatch seller experiences a stock issue which results in delayed shipment while other sellers handle their orders instantly.

When the scheduled delivery date arrives the user obtains only two items but they need to reach out to Customer Support. An employee of the support team accesses the SQL database system to verify order progress. The order status database indicates the smartwatch order remains in the processing phase although the other products successfully shipped to the recipient. After contacting the seller the agent learns there is goods availability but understands the seller will not fulfill the order. The agent explains to the user two possible choices namely refund or replacement of the items.

The finance staff reviews user payment records since the transaction proceeded through three installments. The purchasing system registered the initial payment for the order after which the pending second and third payments will be handled. The processing system cancels any outstanding installments through the system which prevents the user from being charged for the non-deliverable product. The advance payment sent by users gets returned to their credit card account first and a confirmation email is immediately sent.

The system automatically flags the seller for failure to fulfill the order. The seller's cancellation rate increases, affecting their marketplace ranking. The product is temporarily removed from listings due to stock issues, and the seller receives a penalty notification, impacting their reputation score. Meanwhile, the operations team analyzes seller performance trends, customer refund patterns, and top-selling products to improve overall marketplace efficiency.

This SQL-based database is essential in streamlining order tracking, automating refunds, maintaining seller accountability, and providing actionable business insights. It ensures faster dispute resolution, improved seller transparency, and enhanced user satisfaction, all of which are crucial for a growing e-commerce business.

VI. ER DIAGRAM:

[Entity Relationship Diagram:]

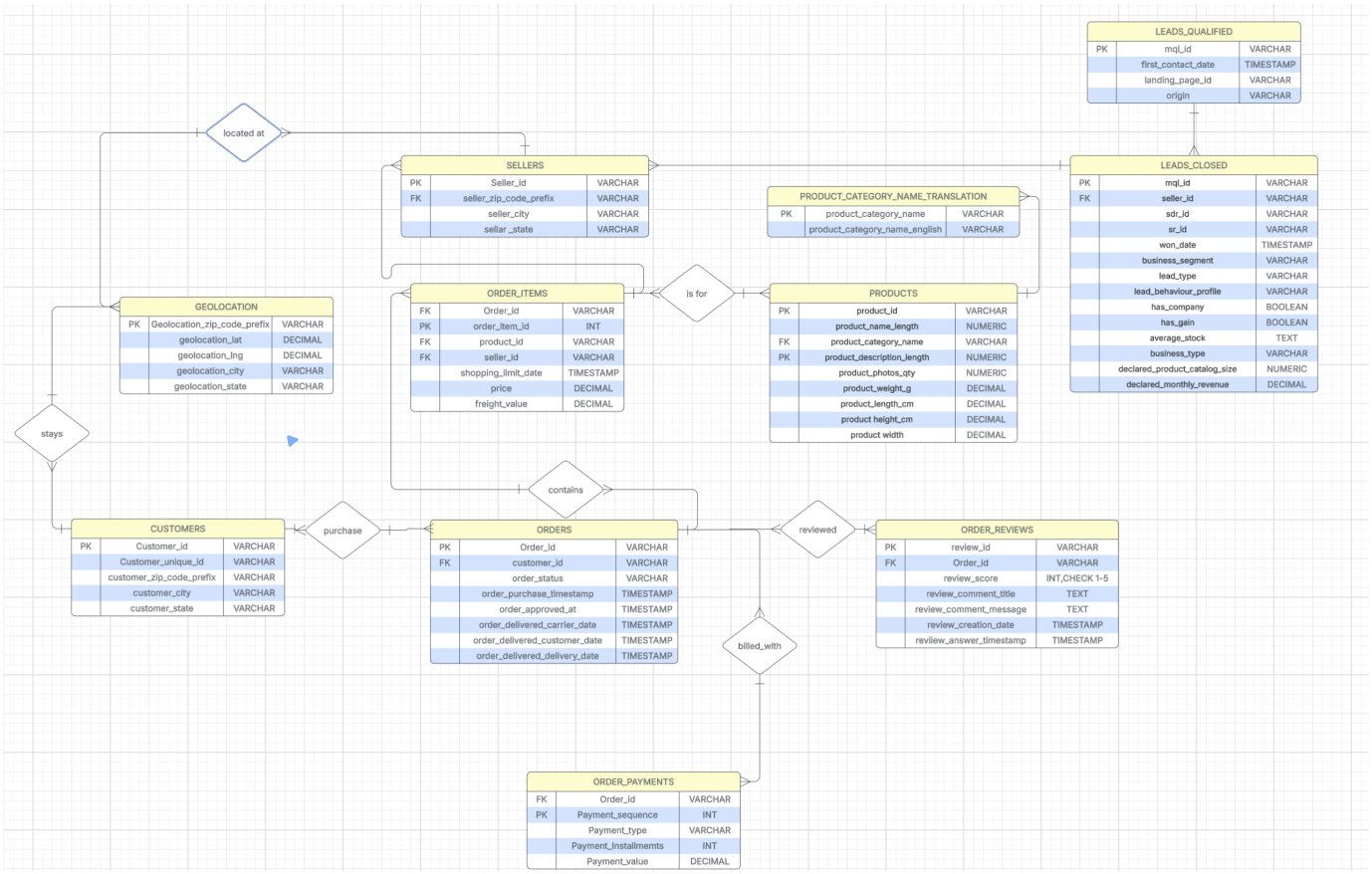


Fig. 1. ER Diagram : ER Diagram Link for more clarity visible

A. *CUSTOMERS Table*

1) Primary Key:

- The primary key for **CUSTOMERS** table is `customer_id` (`VARCHAR`).
- It will ensure that each customer is uniquely identified. The `customer_id` serves as the point of convergence between orders and other customer-related data.

2) Attributes and Descriptions:

- The **CUSTOMERS** table has the following properties:
- **customer_id** (**VARCHAR**): The primary identifier for this table. This column cannot be NULL. It relates customers to their payment records and orders.
- **customer_unique_id** (**VARCHAR**): A globally unique customer identifier made available to customers. This facilitates tracking returning customers and for analytical purposes. This field cannot be NULL.
- **customer_zip_code_prefix** (**VARCHAR**): The initial segment of the customer address ZIP code.Used for geolocation analysis. It cannot be NULL.
- **customer_city** (**VARCHAR**): City of the customer lives. This assists in territorial research and shipping facilitation. This column may be NULL.
- **customer_state** (**VARCHAR**): The state where the customer is available. Handy for taxation, shipping rules, and demographic data. This column may be NULL.

3) Foreign Key Relationships:

- The **CUSTOMERS** table is missing any foreign key relationships. The **ORDERS** table, however, references the `customer_id` column.
- When removing a customer, all corresponding orders in the **ORDERS** table are removed since due to the `CASCADE` action, preventing orphaned orders in the database.
- All of the customer's essential information is stored in the **CUSTOMERS** table.
- The database relates customers through `customer_id`, thus making it a useful component in customer relationships. The company utilizes customer orders to perform

behavioural analysis and market research based on its database technology.

B. SELLERS Table

1) Primary Key:

- The primary key of the **SELLERS** table is `seller_id` (`VARCHAR`).
- It uniquely identifies every seller within the system, ensuring that all seller information occur just once. The `seller_id` occurs in more than one database table, primarily linking sellers to the **ORDER ITEMS** and **LEADS CLOSED** tables to record transactions.

2) Attributes and Descriptions:

- The **SELLERS** table possesses the following characteristics:
- **seller_id** (**VARCHAR**): An identifier assigned for every seller. It is an area for observing seller activity, processing orders, and performance measurement. This field cannot be NULL.
- **seller_zip_code_prefix** (**VARCHAR**): Shows the zip seller's location code prefix. This field assists in identifying each seller's location geographically for purposes of logistics and delivery estimates. NULL cannot be set.
- **seller_city** (**VARCHAR**): Contains the name of the city where the vendor is located. It assists to examine the dissemination trend of sellers across different geographical areas. This field can be NULL.
- **seller_state** (**VARCHAR**): Holds the state where the seller runs business. It is beneficial to tax law, business policies, and regional analysis. This column may be NULL.

3) Foreign Key Relationships:

- The **SELLERS** table has three foreign keys:
- `seller_zip_code_prefix` field is linked to the **GEOLOCATION** table to which every seller can be linked to a geographical region.
- **ORDER ITEMS** table utilizes the `seller_id` field to establish connections between suppliers and their respective electronic transactions.
- The **LEADS CLOSED** table cross-references the `seller_id` column.
- **Cascade Deletion Rules:** When a seller is deleted from the database:
- All corresponding records in the **ORDER ITEMS** table will be deleted due to the `CASCADE` action.
- All corresponding records in the **LEADS CLOSED** table will be eliminated as well due to the `CASCADE` action.

This framework ensures that no orphaned records remain in the system when a seller is de-listed. The **SELLERS** table is required to track marketplace vendors, handle seller logistics, and offer smooth order processing.

C. PRODUCTS Table

1) Primary Key:

- The primary key for the **PRODUCTS** table is `product_id` (`VARCHAR`). This ensures that each product has a unique system identifier. The `product_id` field is referenced by the **ORDER ITEMS** table to link products with orders.

2) Attributes and Descriptions:

- The **PRODUCTS** table contains the following attributes:
- **product_id** (**VARCHAR**): The primary key for the table, uniquely identifying each product. This field is used to track product specifications, customer orders, and associated categories. This field cannot be NULL.
- **product_name_length** (**NUMERIC**): Represents the character count of the product name. It is used for text analysis and product structuring in the catalog. This field cannot be NULL.
- **product_category_name** (**VARCHAR**): Links the product to a specific category. This field references the **PRODUCT CATEGORY NAME TRANSLATION** table. This field cannot be NULL.
- **product_description_length** (**NUMERIC**): Stores the length of the product description in characters. It is used to assess product information completeness. This field cannot be NULL.
- **product_photo_count** (**NUMERIC**): Represents the number of photos associated with a product. It helps determine product display in the marketplace. This field cannot be NULL.
- **product_weight_g** (**DECIMAL**): Represents the weight of the product in grams. It is useful for shipping calculations and logistics. This field cannot be NULL.
- **product_length_cm** (**DECIMAL**): Stores the length of the product in centimeters. It helps with packaging and warehouse storage management. This field cannot be NULL.
- **product_height_cm** (**DECIMAL**): Represents the height of the product in centimeters. It is essential for volume-based shipping cost estimation. This field cannot be NULL.
- **product_width_cm** (**DECIMAL**): Stores the width of the product in centimeters. This attribute aids in packaging optimization and storage. This field cannot be NULL.

3) Foreign Key Relations:

- The **PRODUCTS** table contains three foreign keys:
- The **product_category_name** field links to the **PRODUCT CATEGORY NAME TRANSLATION** table, ensuring that each product is assigned a valid category.
- The **ORDER ITEMS** table references the `product_id` field to associate products with orders.
- **Cascade Deletion Rules:** If a product is deleted from the database:
- All corresponding records in the **ORDER ITEMS** table will be removed due to the `CASCADE` action.

This mechanism ensures data integrity by preventing orphaned records. The **PRODUCTS** table is crucial for e-commerce operations, as it maintains inventory management,

links inventory items to product categories, and supports order tracking functionalities.

D. ORDERS Table

1) Primary Key:

- The primary key of the **ORDERS** table is `order_id` (`VARCHAR`). This identifies each customer order uniquely. The `order_id` is a multi-table reference key. **ORDER ITEMS**, **ORDER PAYMENTS**, and **ORDER REVIEWS** tables are dependent on the `order_id` primary key contained within the **ORDERS** table. Each order maintains its relationship through referencing links to corresponding records in these tables.

2) Attributes and Descriptions:

- The **ORDERS** table contains the following attributes:
- **order_id** (**VARCHAR**): The primary key of the table, which uniquely identifies each order. The database refers to every action an order has an impact on using this field. This field is not NULL.
- **customer_id** (**VARCHAR**): Refers to the primary key in the **CUSTOMERS** table. Each order points to a recurring customer. This field is not NULL.
- **order_status** (**VARCHAR**): The current status of an order. Possible values are `"pending,"` `"shipped,"` `"delivered,"` or `"canceled."` Cannot be NULL and has a default value of `"pending."`
- **order_purchase_timestamp** (**TIMESTAMP**): Stores the order purchase timestamp. This field is utilized for order tracking and analysis. An automatic default value of the current timestamp is given.
- **order_approved_at** (**TIMESTAMP**): Stores timestamp representing order payment approval. If NULL, the order remains in a pending approval status.
- **order_delivered_carrier_date** (**TIMESTAMP**): Specifies the exact date when the order is handed over to the shipping carrier. If NULL, the order is not yet shipped.
- **order_delivered_customer_date** (**TIMESTAMP**): Specifies the date the order is actually delivered to the customer. If NULL, the order has not yet been delivered.
- **order_estimated_delivery_date** (**TIMESTAMP**): Records the order's estimated delivery date. If NULL, no estimated delivery date exists.

3) Foreign Key Constraints:

- **ORDERS** has a single foreign key:
- `customer_id`: References the **CUSTOMERS** table to point to enrolled customers.
- **ORDER ITEMS**, **ORDER PAYMENTS**, and **ORDER REVIEWS** `order_id` field refers to this table. Deletion of an order also removes all corresponding payment records, items, and review documents due to the `CASCADE` action. It ensures consistency across transactions and prevents orphan data records.

The **ORDERS** table maintains all customer purchase information, which is essential to business operations. It records customer orders with payment and delivery information. This table stores e-commerce transaction history in the database.

E. ORDER ITEMS Table

1) Primary Key:

- The primary key for the **ORDER ITEMS** table is `order_item_id` (`INT`). This allows each item within an order to be uniquely distinguishable when multiple items exist in one order. The `order_item_id` is auto-incremented for uniqueness.

2) Attributes and Descriptions:

- The **ORDER ITEMS** table contains the following attributes:
- **order_id** (**VARCHAR**): This attribute is a reference to the **ORDERS** table so that every order item has a valid order. This field cannot be NULL.
- **order_item_id** (**INT**): This is a unique identifier for every item of an order. It distinguishes multiple items in the same order. This column should not be NULL.
- **product_id** (**VARCHAR**): This column references the **PRODUCTS** table, linking each order item to a valid product. This column should not be NULL.
- **seller_id** (**VARCHAR**): This column references the **SELLERS** table, linking each order item to a valid seller. This column should not be NULL.
- **shopping_limit_date** (**TIMESTAMP**): This column stores the shipping time limit for the product. It is used for order item tracking. In the event NULL, no shipping due date is entered.
- **price** (**DECIMAL**): The selling price of the product in this particular order item. This helps in computing revenue and customers' payments. It has a default value of `0.00`.
- **freight_value** (**DECIMAL**): This is the cost of shipping of the order item. This contributes to the computation of total cost for the customer. Its default value is `0.00`.

3) Foreign Key Actions:

- The **ORDER ITEMS** table contains foreign keys:
- The `order_id` field references the **ORDERS** table, so that each item belongs to a valid order.
- The `product_id` field references the **PRODUCTS** table, linking each item to a valid product.
- The `seller_id` field references the **SELLERS** table, associating items with the seller who will ship them.
- Since `order_id`, `product_id`, and `seller_id` are referenced:
- If an order is deleted, the related order items will be deleted by the `CASCADE` action as well.
- If a product is deleted, related order items will also be deleted in order not to have orphaned records.
- If a seller is deleted, all referenced order items will also be deleted.

The **ORDER ITEMS** table plays a very significant role in tracking particular products per order, assigning them to sellers, and allocating correct pricing and freight costs.

F. ORDER PAYMENTS Table

1) Primary Key:

- The primary key for the **ORDER PAYMENTS** table is `payment_sequence` (`INT`). This uniquely identifies each payment transaction associated with an order. Since orders may involve multiple payments (installments), each payment within the same order is assigned a unique sequence number.

2) Attributes and Descriptions:

- The **ORDER PAYMENTS** table contains the following attributes:
- **order_id** (**VARCHAR**): This attribute references the **ORDERS** table, ensuring that every payment belongs to a valid order. This field cannot be NULL.
- **payment_sequence** (**INT**): This is a unique identifier for each payment installment related to an order. It ensures proper tracking of multiple payments per order. This field cannot be NULL.
- **payment_type** (**VARCHAR**): This field specifies the method of payment, such as "credit card," "debit card," "boleto," or "voucher." This field cannot be NULL and defaults to "unknown."
- **payment_installments** (**INT**): This attribute indicates the number of installments used for the payment. If NULL, it means the payment was made in full. The default value is 1.
- **payment_value** (**DECIMAL**): This field stores the total amount of the payment installment. It helps track financial transactions and revenue. The default value is 0.00.

3) Foreign Key Actions:

- The **ORDER PAYMENTS** table contains one foreign key:
- The `order_id` attribute references the **ORDERS** table, ensuring that each payment is linked to a valid order.

Since `order_id` is referenced in **ORDER PAYMENTS**:

- If an order is deleted, all corresponding payment records will also be deleted due to the `CASCADE` action.
- This ensures data integrity by preventing orphaned payment records.

The **ORDER PAYMENTS** table is crucial for tracking payment transactions, handling installment-based payments, and ensuring proper financial records for each order.

G. ORDER REVIEWS Table

1) Primary Key:

- Primary key for the **ORDER REVIEWS** table is `review_id` (`VARCHAR`). This specifically sets all the reviews that a customer has written. It guarantees that there are no repeated reviews and enables tracking of individual reviews for examination and customer comment.

2) Attributes and Descriptions:

- The **ORDER REVIEWS** table attributes are as follows:
- **review_id** (**VARCHAR**): It is the primary key for each customer review. It gives accurate monitoring of feedback for orders. This column cannot be NULL.
- **order_id** (**VARCHAR**): This column is the **ORDERS** table reference so that each review is linked with a valid order. This field must not be NULL.
- **review_score** (**INT**, CHECK 1–5): This column represents the rating provided by the customer out of 1 to 5. It helps in assessing customer satisfaction. This field cannot be NULL.
- **review_comment_title** (**TEXT**): This property contains a short title that summarizes the review. It provides a brief insight into the customer's feedback. This field may be NULL.
- **review_comment_message** (**TEXT**): This field has the entire review message provided by the customer. It assists in assessing in-depth customer opinions. This column may be NULL.
- **review_creation_date** (**TIMESTAMP**): This property contains the date and time of posting of the review. It helps to understand trends in feedback over time. The default is the present timestamp.
- **review_answer_timestamp** (**TIMESTAMP**): This attribute documents the date and time when a response was allocated to the review. If NULL, then the review has yet to be answered.

3) Foreign Key Actions:

- The **ORDER REVIEWS** table has a foreign key:
- The `order_id` attribute references the **ORDERS** table to ensure reviews are only created for real orders.

Since `order_id` is referenced in **ORDER REVIEWS**:

- When a user order is cancelled, all its associated reviews are also deleted by the `CASCADE` action.
- This avoids reviews from being orphaned.

The **ORDER REVIEWS** table is required for noting customer feedback, service quality, and improved user experience based on sentiment analysis and satisfaction ratings.

H. PRODUCT CATEGORY NAME TRANSLATION Table

1) Primary Key:

- The primary key for the **PRODUCT CATEGORY NAME TRANSLATION** table is `product_category_name` (`VARCHAR`). This ensures that each product category name is uniquely identified and mapped to its English equivalent. Since product categories may be stored in different languages, this table provides a reference for standardized names.

2) Attributes and Descriptions:

- The **PRODUCT CATEGORY NAME TRANSLATION** table contains the following attributes:
- **product_category_name** (**VARCHAR**): This is the unique identifier for each product category. It represents the category name used in the original dataset, which may be in a different language. This field cannot be NULL.
- **product_category_name_english** (**VARCHAR**): This attribute provides the English translation of the product category name. It helps in standardizing category names for analysis and reporting. This field cannot be NULL.

3) Foreign Key Actions:

- The **PRODUCT CATEGORY NAME TRANSLATION** table does not contain any foreign keys, as it serves as a reference table for category names.
- The **PRODUCT CATEGORY NAME TRANSLATION** table is essential for maintaining a consistent naming convention across different languages, enabling better data analysis, reporting, and category-based filtering.

I. GEOLOCATION Table

1) Primary Key:

- The primary key for the **GEOLOCATION** table is `geolocation_zip_code_prefix` (`VARCHAR`). This ensures that each geographical location entry is uniquely identified based on the zip code prefix. The zip code prefix serves as a regional identifier for mapping locations.

2) Attributes and Descriptions:

- The **GEOLOCATION** table contains the following attributes:
- **geolocation_zip_code_prefix** (**VARCHAR**): This is the unique identifier for a geographic location. It represents the zip code prefix associated with a particular region. This field cannot be NULL.
- **geolocation_lat** (**DECIMAL(9,6)**): This attribute stores the latitude coordinate of the location. It allows for accurate geospatial mapping and distance calculations. This field cannot be NULL.
- **geolocation_lng** (**DECIMAL(9,6)**): This attribute stores the longitude coordinate of the location. It is used in conjunction with latitude for geographical positioning. This field cannot be NULL.
- **geolocation_city** (**VARCHAR**): This field stores the name of the city corresponding to the zip code prefix. It helps in regional analysis and logistics. This field can be NULL.
- **geolocation_state** (**VARCHAR**): This attribute represents the state where the location is situated. It is useful for shipping logistics, taxation, and regional analytics. This field can be NULL.

3) Foreign Key Actions:

- The **GEOLOCATION** table does not contain any foreign keys. However, it serves as a reference for other tables such as **CUSTOMERS** and **SELLERS** to link zip codes with location details.
- The **GEOLOCATION** table is crucial for mapping customer and seller locations, enabling logistics management, delivery tracking, and distance-based optimizations for e-commerce operations.

J. LEADS QUALIFIED Table

1) Primary Key: The primary key for the **LEADS QUALIFIED** table is `mql_id` (`VARCHAR`). This uniquely identifies each marketing qualified lead (MQL). The primary key ensures that no duplicate leads exist and allows proper tracking of lead conversions.

2) Attributes and Descriptions:

- The **LEADS QUALIFIED** table contains the following attributes:
- **mql_id** (**VARCHAR**): This is the unique identifier for a marketing qualified lead. It allows tracking of potential customers who have shown interest in products or services. This field cannot be NULL.
- **first_contact_date** (**TIMESTAMP**): This attribute stores the date and time when the lead first interacted with the platform. It is useful for analyzing the lead conversion timeline. The default value is the current timestamp.
- **landing_page_id** (**VARCHAR**): This field identifies the landing page that the lead visited before conversion. It helps track the effectiveness of marketing campaigns. This field can be NULL.
- **origin** (**VARCHAR**): This attribute represents the source through which the lead was acquired, such as social media, email campaigns, or advertisements. It is useful for marketing analysis. This field can be NULL.

3) Foreign Key Actions:

- The **LEADS QUALIFIED** table does not contain any foreign keys. However, it serves as a reference for analyzing customer acquisition strategies and marketing funnel efficiency.
- The **LEADS QUALIFIED** table is crucial for monitoring lead conversions, optimizing marketing campaigns, and understanding customer acquisition trends.

K. LEADS CLOSED Table

1) Primary Key:

- The primary key for the **LEADS CLOSED** table is `mql_id` (`VARCHAR`). This clearly qualifies each closed marketing qualified lead (MQL). It prevents the duplication of leads and enables tracking of leads converted successfully.

2) Attributes and Descriptions:

- The **LEADS CLOSED** table has the following characteristics:
- **mql_id** (**VARCHAR**): It is the primary key for a marketing qualified lead that has closed. It makes sure precise monitoring of lead conversion. This is not NULL.
- **seller_id** (**VARCHAR**): This is the **SELLERS** table reference, linking every closed lead to a specific seller. It must not be NULL.
- **sdr_id** (**VARCHAR**): This field defines the sales development representative (SDR) responsible for engaging with the lead. This column may be NULL.
- **sr_id** (**VARCHAR**): The column contains the senior representative (SR) who oversaw the process of lead conversion. This column may be NULL.
- **won_date** (**TIMESTAMP**): The date is stored in this field when the lead was successfully converted into a customer. It helps to measure lead conversion efficiency. This field can be NULL.

- **business_segment** (**VARCHAR**): This field specifies the industry of business related to the lead, i.e., retail or technology. This column may be NULL.
- **lead_type** (**VARCHAR**): This column defines the type of lead, i.e., inbound or outbound. This field can be NULL.
- **lead_behaviour_profile** (**VARCHAR**): This field provides an indication of the lead's profile of behavior based on interactions. This column is nullable.
- **has_company** (**BOOLEAN**, **DEFAULT FALSE**): This attribute indicates whether the lead is associated with a company. The default is FALSE.
- **has_gain** (**BOOLEAN**, **DEFAULT FALSE**): This field suggests whether the lead is prospective to support revenue growth. The default is FALSE.
- **average_stock** (**VARCHAR**): The column holds information regarding the lead's average product inventory. This field can be NULL.
- **business_type** (**VARCHAR**): This column is the nature of business the lead is engaged in, i.e., B2B or B2C. This column may be NULL.
- **declared_product_catalog_size** (**DECIMAL(10,2)**): This property captures the approximate size of the product catalog kept by the lead. It gives details on business potential. This column may be NULL.
- **Monthly_declared_revenue** (**DECIMAL(10,2)**): This field is the rough monthly income reported by the lead. It aids in business forecasting. This can be NULL.

3) Foreign Key Actions:

- The **LEADS CLOSED** table has a foreign key:
- **seller_id** attribute is a reference to the **SELLERS** table, which guarantees that all closed leads are linked to a valid seller.

Since **seller_id** is highlighted in **LEADS CLOSED**:

- When a seller is removed, all associated closed leads will also be deleted due to the `ON DELETE CASCADE` action.
- This ensures data integrity by preventing orphaned lead records.

LEADS CLOSED table is employed to note successfully converted leads, monitoring seller performance, and researching business expansion patterns.

L. Cardinality of the Relations in given ER diagram

CUSTOMERS → ORDERS (One-to-Many) [1:N] Customers have the ability to create numerous orders through the system. One customer can possess a single order. Each customer creates numerous orders which always relate back to an individual customer.

ORDERS → ORDER_ITEMS (One-to-Many) [1:N] The same order contains an unlimited number of order items. All order items exclusively link to a single order in the system. The cardinality shows that one order contains multiple items even though every order item connects with a single order.

PRODUCTS → ORDER_ITEMS (One-to-Many) [1:N] A single product can exist in several order items when placed for sale. There is just one product related to each order item. A product appears in multiple order items while each order item contains one and only one product.

SELLERS → ORDER_ITEMS (One-to-Many) [1:N] Multiple products can be sold through each seller and thus appear in different order items. All order items directly link with one individual seller. Each seller maintains at least one order item though every order item relates to a single seller.

ORDERS → ORDER_PAYMENTS (One-to-Many) [1:N] An order contains several payment installments depending on the number of payments involved. The payment installment extends only to a single order. Each order contains many payments although each payment belongs to a single order.

ORDERS → ORDER_REVIEWS (One-to-One) [1:1] The review section of an order contains maximum one entry. One order contains only a single active review. Each order only receives one review since a single review links to exactly one order.

GEOLOCATION → CUSTOMERS (One-to-Many) [1:N] One geolocation zip code prefix may link up with an unlimited number of customers. Customers have association with a single geolocation prefix throughout the database system. The relationship operates under cardinality rules which state that one single zip code matches several customers while every individual client belongs to only one such zone.

GEOLOCATION → SELLERS (One-to-Many) [1:N] One geolocation zip code prefix establishes associations with various sellers in the system. A seller connects to only one single geographic area prefix. Each zip code serves as a home for various sellers yet all sellers match exactly one unique geographical code.

PRODUCT_CATEGORY_NAME_TRANSLATION → PRODUCTS (One-to-Many) [1:N] A translated product category name maintains relationships with several products. Every product receives its classification from just one category name. A single category name connects to different products though every product participates in just one category.

LEADS_QUALIFIED → LEADS_CLOSED (One-to-One) [1:1] Marketers consider marketing qualified leads as closeable once. The precise relationship exists between each single closed lead with a single unique MQL. An MQL qualifies as one complete lead that transforms into a closed deal.

SELLERS → LEADS_CLOSED (One-to-Many) [1:N] A single seller can work on different closed leads despite their association with multiple sellers. Each final lead can only be connected to one specific seller. Each lead requires single closure through a particular seller even though one seller can handle various leads during the process.

VII. TASK 3: LOAD THE LARGE DATASET

In data they are some Nan values and redudancies we removed them and loaded them into SQL using python script because the data is large so we used script to insert the data into tables

below figure is python script :

Fig. 2. Python script

Creating Tables

we have created the tables using primary key constraint

Fig. 3. Create Tables 1

Fig. 4. Create Tables 2

Alter Tables

we added foreign key constraint using alter tables commands:

Fig. 5. Create Tables 3

Fig. 6. Alter Table 1

Now we are verifying whether the tables created and the data is loaded properly

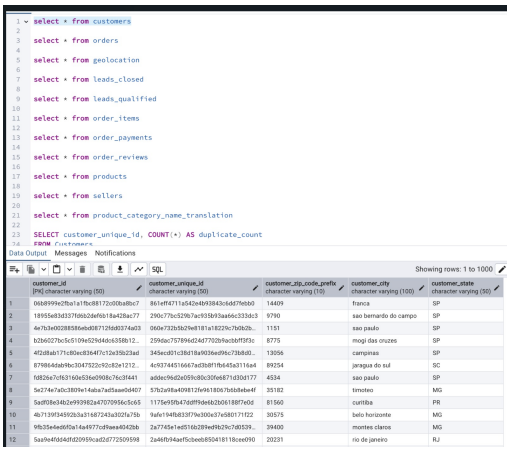


Fig. 7. Verifying Table 1

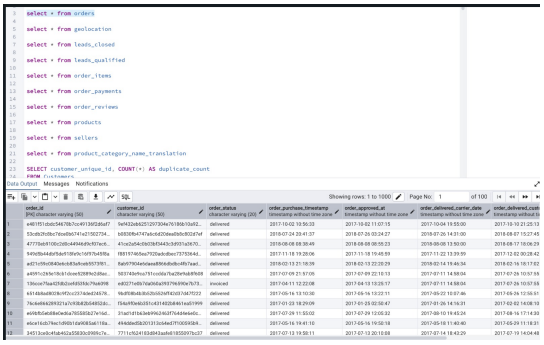


Fig. 8. Verifying Table 2

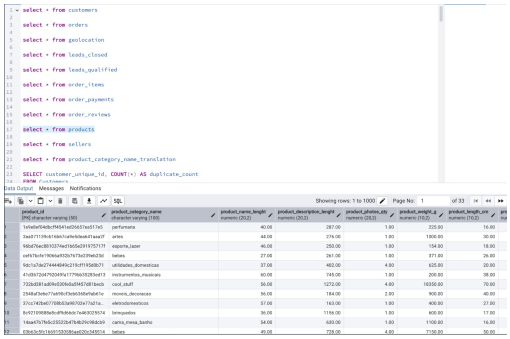


Fig. 9. Verifying Table 3

QUERIES

We are validating some conditions using some queries attached below:

1 Find Duplicate Customers

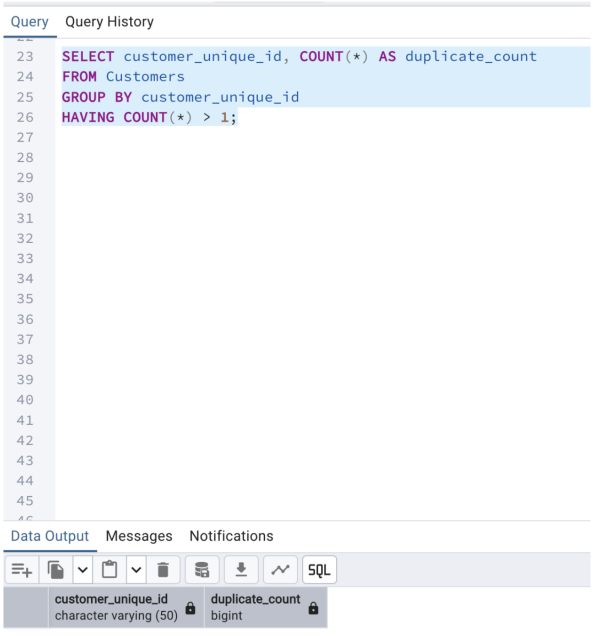


Fig. 10. Query 1

By executing this query, we're checking whether the customer_unique_id column is actually one unique ID per customer within the table. Although the same person may have multiple customer_id records possibly due to having multiple orders or accounts—the customer_unique_id is supposed to be the same to signify that one person.

2 Find Orders Without Customers

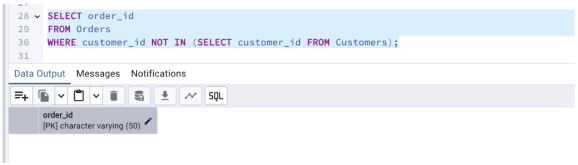


Fig. 11. Query 2

3 Find Orphaned Products Without a Category

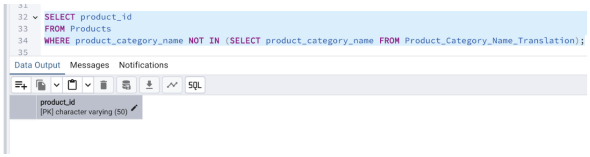


Fig. 12. Query 3

4 Find Orders with Multiple Sellers

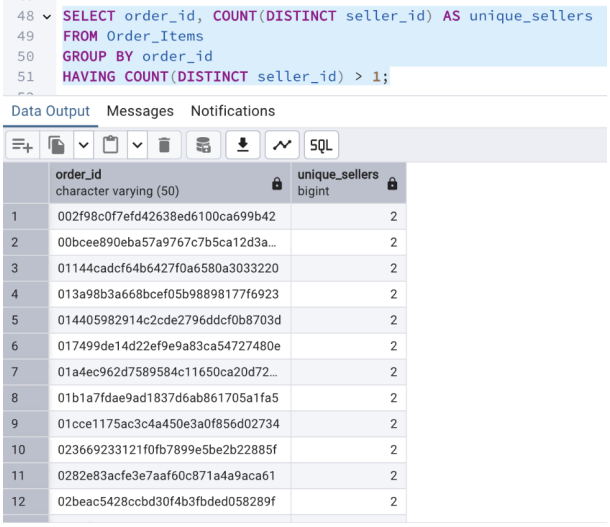
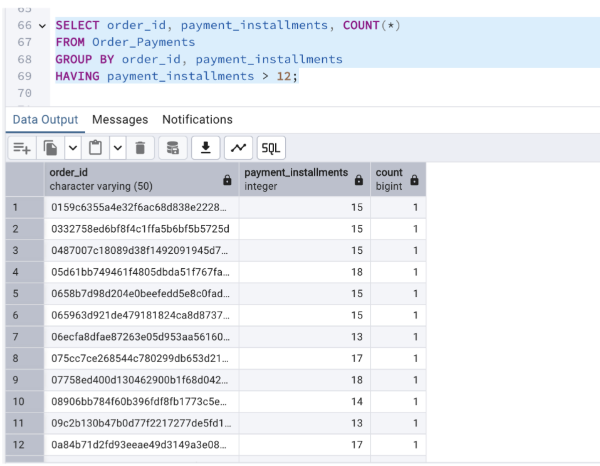


Fig. 13. Query 4

5 Find Orders with Unusual Payment Installments



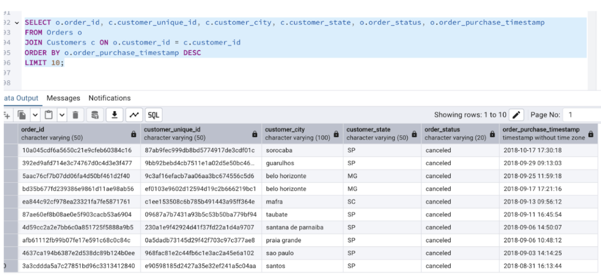
```sql
SELECT order_id, payment_installments, COUNT(*)
FROM Order_Payments
GROUP BY order_id, payment_installments
HAVING payment_installments > 12;
```

Data Output | Messages | Notifications

	order_id character varying (50)	payment_installments integer	count bigint
1	0159c6355a4e32f6ac68d838e2228...	15	1
2	0332758ed6bf8f4c1ffa5b6bf5b5725d	15	1
3	0487007c18089d38f1492091945d7...	15	1
4	05d61bb749461f4805dbda51f767fa...	18	1
5	0658b7d98d204e0beefedd5e8c0fad...	15	1
6	065963d921de479181824ca8d8737...	15	1
7	06ecfa8dfae87263e05d953aa56160...	13	1
8	075cc7ce268544c780299db653d21...	17	1
9	07758ed400d130462900b1f68d042...	18	1
10	08906bb784f60b396fdf8fb1773c5e...	14	1
11	09c2b130b47b0d77f2217277de5fd1...	13	1
12	0a84b71d2fd93eeae49d3149a3e08...	17	1

Fig. 14. Query 5

6 Find Orders with Customer Details



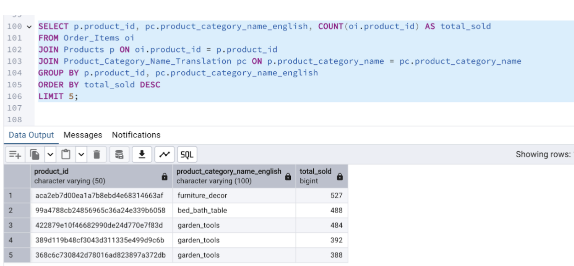
```sql
SELECT o.order_id, c.customer_unique_id, c.customer_city, c.customer_state, o.order_status, o.order_purchase_timestamp
FROM Orders o
JOIN Customers c ON o.customer_id = c.customer_id
ORDER BY o.order_purchase_timestamp DESC
LIMIT 10;
```

Data Output | Messages | Notifications — Showing rows: 1 to 10 — Page No: 1

order_id character varying (50)	customer_unique_id character varying (50)	customer_city character varying (100)	customer_state character varying (50)	order_status character varying (20)	order_purchase_timestamp timestamp without time zone
10a045cdf6a5650c21e9cfeb60384c16	87ab9fec999db8bd5774917de3cdf01c	sorocaba	SP	canceled	2018-10-17 17:30:18
392ed9afd714e3c74767d0c4d3e3f477	9bb92bebd4cb7511e1a02d5e50bc46...	guarulhos	SP	canceled	2018-09-29 09:13:03
5aac76cf7b07dd06fa4d50bf461d2f40	9c3af16efacb7aa06aa3bc674556c5d6	belo horizonte	MG	canceled	2018-09-25 11:59:18
bd35b677fd239386e9861d11ae98ab56	ef0103e9602d12594d19c2b666219bc1	belo horizonte	MG	canceled	2018-09-17 17:21:16
ea844c92cf978ea23321fa7fe5871761	c1ee153508c6b785b491443a95ff364e	mafra	SC	canceled	2018-09-13 09:56:12
87ae60ef8b08ae0e5f903cacb53a6904	09687a7b7431a93b5c53b50ba779bf94	taubate	SP	canceled	2018-09-11 16:45:54
4d59cc2a2e7bb6c0a851725f5888a9b5	230a1e9f42924d41f37fd22a1d4a9707	santana de parnaiba	SP	canceled	2018-09-06 14:50:07
afb61112fb99b07fe17e591c68c0c84c	0a5dadb73145d29f42f703c97c377ae8	praia grande	SP	canceled	2018-09-06 10:48:12
4637ca194b6387e2d538dc89b124b0ee	968fac81e2c44fb6c1e3ac2a45e6a102	sao paulo	SP	canceled	2018-09-03 14:14:25
3a3cddda5a7c27851bd96c3313412840	e90598185d2427a35e32ef241a5c04aa	santos	SP	canceled	2018-08-31 16:13:44

Fig. 15. Query 6

7 Find Top 5 Best-Selling Products with Category



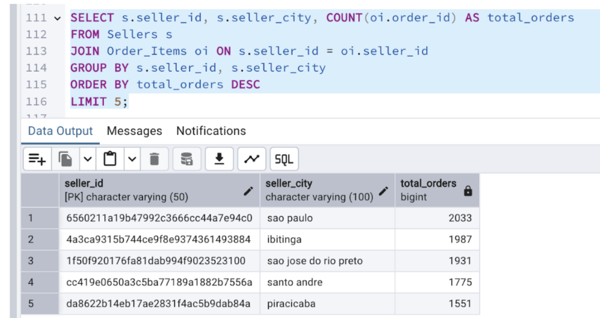
```sql
SELECT p.product_id, pc.product_category_name_english, COUNT(oi.product_id) AS total_sold
FROM Order_Items oi
JOIN Products p ON oi.product_id = p.product_id
JOIN Product_Category_Name_Translation pc ON p.product_category_name = pc.product_category_name
GROUP BY p.product_id, pc.product_category_name_english
ORDER BY total_sold DESC
LIMIT 5;
```

Data Output | Messages | Notifications — Showing rows:

	product_id character varying (50)	product_category_name_english character varying (100)	total_sold bigint
1	aca2eb7d00ea1a7b8ebd4e68314663af	furniture_decor	527
2	99a4788cb24856965c36a24e339b6058	bed_bath_table	488
3	422879e10f46682990de24d770e7f83d	garden_tools	484
4	389d119b48cf3043d311335e499d9c6b	garden_tools	392
5	368c6c730842d78016ad823897a372db	garden_tools	388

Fig. 16. Query 7

8 Find Sellers with the Highest Number of Orders



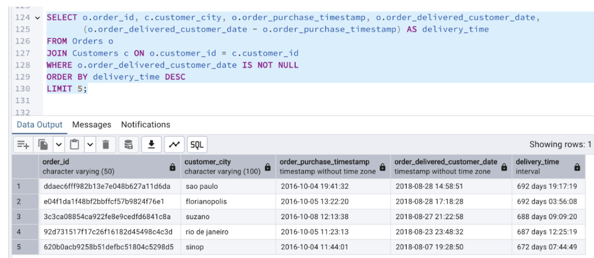
```sql
SELECT s.seller_id, s.seller_city, COUNT(oi.order_id) AS total_orders
FROM Sellers s
JOIN Order_Items oi ON s.seller_id = oi.seller_id
GROUP BY s.seller_id, s.seller_city
ORDER BY total_orders DESC
LIMIT 5;
```

Data Output | Messages | Notifications

	seller_id [PK] character varying (50)	seller_city character varying (100)	total_orders bigint
1	6560211a19b47992c3666cc44a7e94c0	sao paulo	2033
2	4a3ca9315b744ce9f8e9374361493884	ibitinga	1987
3	1f50f920176fa81dab994f9023523100	sao jose do rio preto	1931
4	cc419e0650a3c5ba77189a1882b7556a	santo andre	1775
5	da8622b14eb17ae2831f4ac5b9dab84a	piracicaba	1551

Fig. 17. Query 8

9 Find Orders That Took the Longest to Deliver



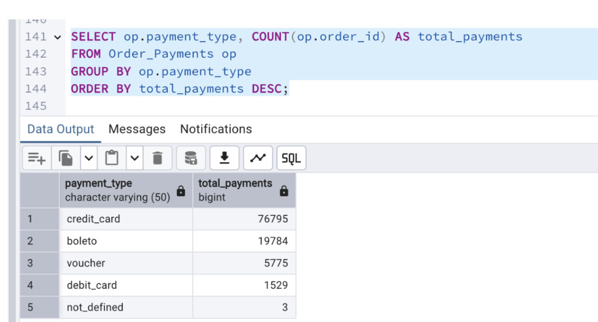
```sql
SELECT o.order_id, c.customer_city, o.order_purchase_timestamp, o.order_delivered_customer_date,
       (o.order_delivered_customer_date - o.order_purchase_timestamp) AS delivery_time
FROM Orders o
JOIN Customers c ON o.customer_id = c.customer_id
WHERE o.order_delivered_customer_date IS NOT NULL
ORDER BY delivery_time DESC
LIMIT 5;
```

Data Output | Messages | Notifications — Showing rows: 1 to

	order_id character varying (50)	customer_city character varying (100)	order_purchase_timestamp timestamp without time zone	order_delivered_customer_date timestamp without time zone	delivery_time interval
1	ddaec6fff982b13e7e048b627a11d6da	sao paulo	2016-10-04 19:41:32	2018-08-28 14:58:51	692 days 19:17:19
2	e04f1da1f48bf2bbffcf57b9824f76e1	florianopolis	2016-10-05 13:22:20	2018-08-28 17:18:28	692 days 03:56:08
3	3c3ca08854ca922fe8e9cedfd6841c8a	suzano	2016-10-08 12:13:38	2018-08-27 21:22:58	688 days 09:09:20
4	92d731517f17c26f16182d45498c4c3d	rio de janeiro	2016-10-05 11:23:13	2018-08-23 23:48:32	687 days 12:25:19
5	620b0acb9258b51defbc51804c5298d5	sinop	2016-10-04 11:44:01	2018-08-07 19:28:50	672 days 07:44:49

Fig. 18. Query 9

10 Find Most Popular Payment Methods



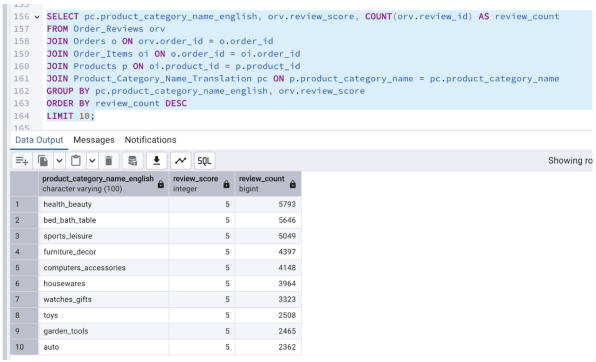
```sql
SELECT op.payment_type, COUNT(op.order_id) AS total_payments
FROM Order_Payments op
GROUP BY op.payment_type
ORDER BY total_payments DESC;
```

Data Output | Messages | Notifications

	payment_type character varying (50)	total_payments bigint
1	credit_card	76795
2	boleto	19784
3	voucher	5775
4	debit_card	1529
5	not_defined	3

Fig. 19. Query 10

11 Find the Most Common Review Score by Product Category



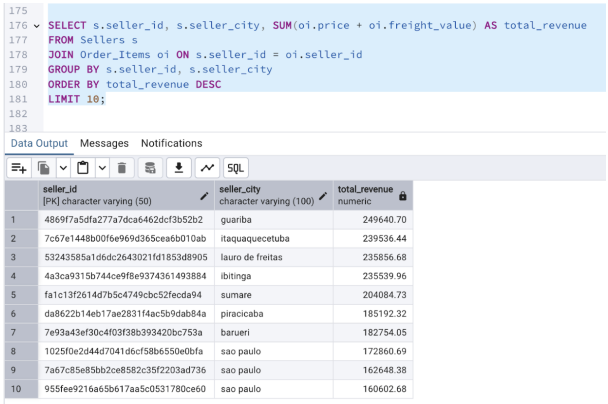
```
SELECT pc.product_category_name_english, orv.review_score, COUNT(orv.review_id) AS review_count
FROM Order_Reviews orv
JOIN Orders o ON orv.order_id = o.order_id
JOIN Order_Items oi ON o.order_id = oi.order_id
JOIN Products p ON oi.product_id = p.product_id
JOIN Product_Category_Name_Translation pc ON p.product_category_name = pc.product_category_name
GROUP BY pc.product_category_name_english, orv.review_score
ORDER BY review_count DESC
LIMIT 10;
```

Data Output | Messages | Notifications

	product_category_name_english character varying (100)	review_score integer	review_count bigint
1	health_beauty	5	5793
2	bed_bath_table	5	5646
3	sports_leisure	5	5049
4	furniture_decor	5	4397
5	computers_accessories	5	4148
6	housewares	5	3964
7	watches_gifts	5	3323
8	toys	5	2508
9	garden_tools	5	2465
10	auto	5	2362

Fig. 20. Query 11

12 Find Total Revenue by Seller



```
SELECT s.seller_id, s.seller_city, SUM(oi.price + oi.freight_value) AS total_revenue
FROM Sellers s
JOIN Order_Items oi ON s.seller_id = oi.seller_id
GROUP BY s.seller_id, s.seller_city
ORDER BY total_revenue DESC
LIMIT 10;
```

Data Output | Messages | Notifications

	seller_id [PK] character varying (50)	seller_city character varying (100)	total_revenue numeric
1	4869f7a5dfa277a7dca6462dcf3b52b2	guariba	249640.70
2	7c67e1448b00f6e969d365cea6b010ab	itaquaquecetuba	239536.44
3	53243585a1d6dc2643021fd1853d8905	lauro de freitas	235856.68
4	4a3ca9315b744ce9f8e9374361493884	ibitinga	235539.96
5	fa1c13f2614d7b5c4749cbc52fecda94	sumare	204084.73
6	da8622b14eb17ae2831f4ac5b9dab84a	piracicaba	185192.32
7	7e93a43ef30c4f03f38b393420bc753a	barueri	182754.05
8	1025f0e2d44d7041d6cf58b6550e0bfa	sao paulo	172860.69
9	7a67c85e85bb2ce8582c35f2203ad736	sao paulo	162648.38
10	955fee9216a65b617aa5c0531780ce60	sao paulo	160602.68

Fig. 21. Query 12

VIII. Task 4: BCNF Relations

Checking the BCNF of all realtions:

A. Customers Table (BCNF)

Attributes:

- customer_id (PK)
- customer_unique_id
- customer_zip_code_prefix
- customer_city
- customer_state

Functional Dependencies (FDs):

- customer_id → customer_unique_id, customer_zip_code_prefix, customer_city, customer_state
- customer_zip_code_prefix → (customer_city, customer_state)

BCNF Check: This table is already in BCNF.

B. Orders Table (BCNF)

Attributes:

- order_id (PK)
- customer_id (FK → Customers)
- order_status
- order_purchase_timestamp
- order_approved_at
- order_delivered_carrier_date
- order_delivered_customer_date
- order_estimated_delivery_date

Functional Dependencies (FDs):

- order_id → customer_id, order_status, order_purchase_timestamp, order_approved_at, order_delivered_carrier_date, order_delivered_customer_date, order_estimated_delivery_date

BCNF Check: This table is already in BCNF.

C. Product_Category_Name_Translation Table (BCNF)

Attributes:

- product_category_name (PK)
- product_category_name_english

Functional Dependencies (FDs):

- product_category_name → product_category_name_english

BCNF Check: This table is already in BCNF.

D. Products Table (BCNF)

Attributes:

- product_id (PK)
- product_category_name (FK → Product_Category_Name_Translation)
- product_name_length
- product_description_length
- product_photos_qty
- product_weight_g
- product_length_cm
- product_height_cm
- product_width_cm

Functional Dependencies (FDs):

- 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

BCNF Check: This table is already in BCNF.

E. Geolocation Table (BCNF)

Attributes:

- geolocation_zip_code_prefix (PK)
- geolocation_lat
- geolocation_lng
- geolocation_city
- geolocation_state

Functional Dependencies (FDs):

- geolocation_zip_code_prefix → geolocation_lat, geolocation_lng, geolocation_city, geolocation_state

BCNF Check: This table is already in BCNF.

F. Sellers Table (BCNF)

Attributes:

- seller_id (PK)
- seller_zip_code_prefix (FK → Geolocation)
- seller_city
- seller_state

Functional Dependencies (FDs):

- seller_id → seller_zip_code_prefix, seller_city, seller_state

- seller_zip_code_prefix → geolocation_city, geolocation_state from Geolocation.

BCNF Check: The primary key is seller_id, and all attributes are functionally dependent on it. This table is already in BCNF.

G. Order_Items Table (BCNF)

Attributes:

- order_id (PK, FK → Orders)
- order_item_id (PK)
- product_id (FK → Products)
- seller_id (FK → Sellers)
- shipping_limit_date
- price
- freight_value

Functional Dependencies (FDs):

- order_id,order_item_id → product_id, _id, shipping_limit_date, price, freight_value

- product_id → product_category_name
  seller_id → seller_zip_code_prefix

BCNF Check: This table is already in BCNF.

H. Order_Payments Table (BCNF)

Attributes:

- order_id (PK, FK → Orders)
- payment_sequential (PK)
- payment_type
- payment_installments
- payment_value

Functional Dependencies (FDs):

- order_id, payment_sequential → payment_type, payment_installments payment_value

- order_id → customer_id (from Orders)

BCNF Check: This table is already in BCNF.

I. Order_Reviews Table (BCNF)

Attributes:

- review_id (PK)
- order_id (FK → Orders)
- review_score
- review_comment_title
- review_comment_message
- review_creation_date
- review_answer_timestamp

Functional Dependencies (FDs):

- order_id,order_item_id → product_id, seller_id, shipping_limit_date, price, freight_value

- product_id → product_category_name
  seller_id → seller_zip_code_prefix
- review_id → order_id, review_score, review_comment_title, review_comment_message, review_creation_date, review_answer_timestamp

- order_id → customer_id (from Orders)

BCNF Check: This table is already in BCNF.

10. Leads_Qualified Table (BCNF)

Attributes:

- mql_id (PK)
- first_contact_date
- landing_page_id
- origin

Functional Dependencies (FDs):

- mql_id → first_contact_date, landing_page_id, origin

BCNF Check: The primary key is mql_id, and all attributes are functionally dependent on it. This table is already in BCNF.

11. Leads_Closed Table (BCNF)

Attributes:

- mql_id (PK, FK → Leads_Qualified)
- seller_id (PK, FK → Sellers)
- sdr_id
- sr_id
- won_date
- business_segment
- lead_type
- lead_behaviour_profile
- has_company
- has_gtin
- average_stock
- business_type
- declared_product_catalog_size
- declared_monthly_revenue

Functional Dependencies (FDs):

- mql_id, seller_id → sdr_id, sr_id, won_date, business_segment, lead_type, lead_behaviour_profile, has_company, has_gtin, average_stock, business_type, declared_product_catalog_size, declared_monthly_revenue

BCNF Check: The composite primary key is (mql_id, seller_id), and all attributes are functionally dependent on it. This table is already in BCNF.

TASK 5 QUERY EXECUTION AND PROCEDURE DESIGN

This task focuses on executing practical operations on the designed SQL database using DML statements like INSERT, UPDATE, DELETE, diverse SELECT queries, and creating stored procedures/functions for reusable operations. All operations were validated with SELECT queries post execution to ensure correctness.

Insert Query 1 – New Customer

Insert Query to insert record into Customer table

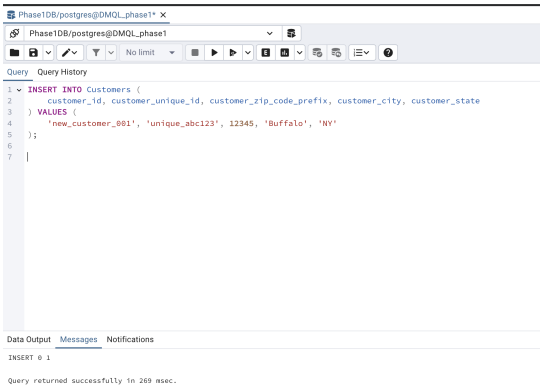


Fig. 1. Inserting into customer table

Validating the insertion of record

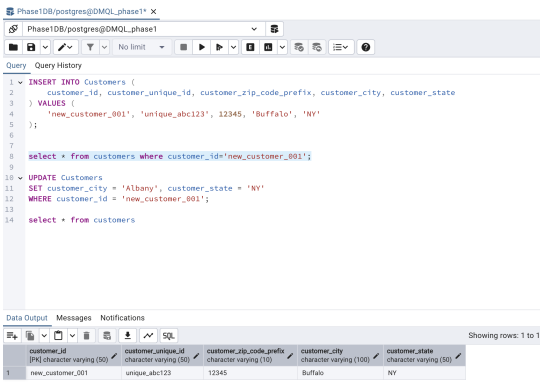


Fig. 2. Validation

Update Query 1 – Customer

Updating record in customer table

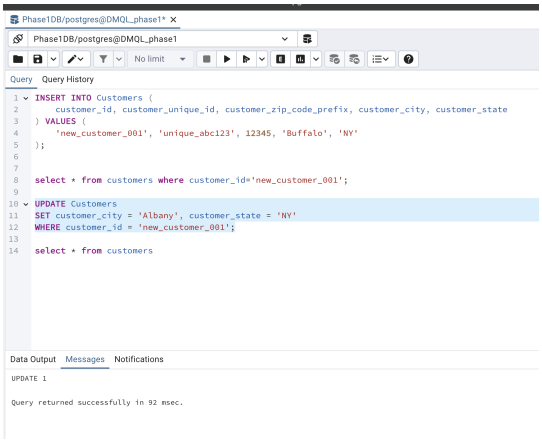


Fig. 3. Updating Customer records

Validating the update query

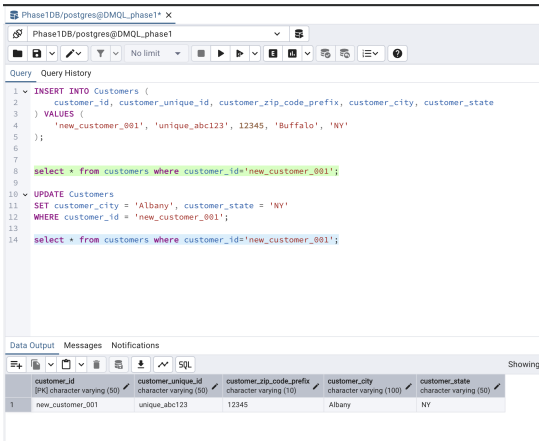


Fig. 4. Validation

Delete Query 1 – Reviews

Delete record from review table

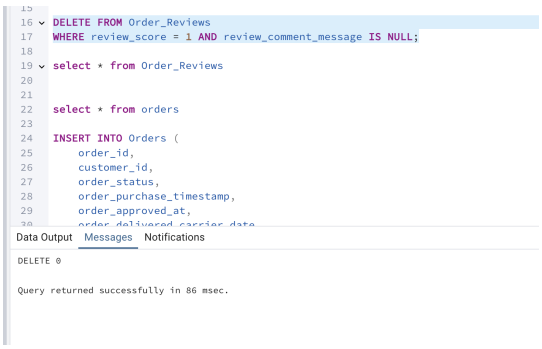


Fig. 5. Deleting Reviews records

Validating the delete query

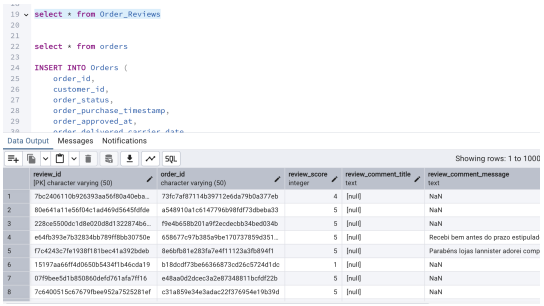


Fig. 6. Validation

Insert Query 2 – Orders

Inserting record into Orders table

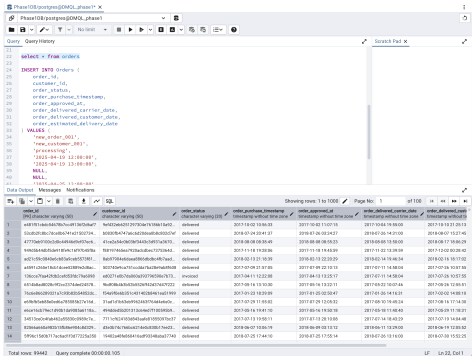


Fig. 7. Inserting Order records

Validating the insertion of record in Order table

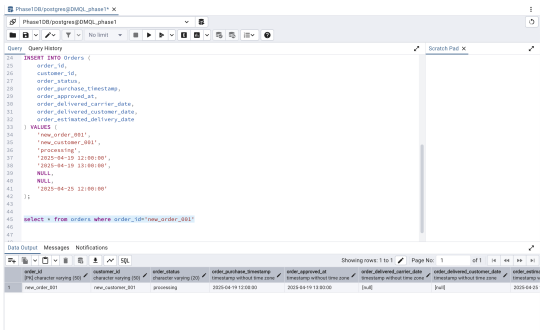


Fig. 8. Validation

Insert Query 3 – Product category name translation

Inserting record in Product category name translation table

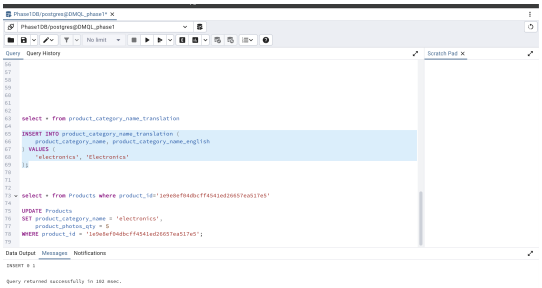
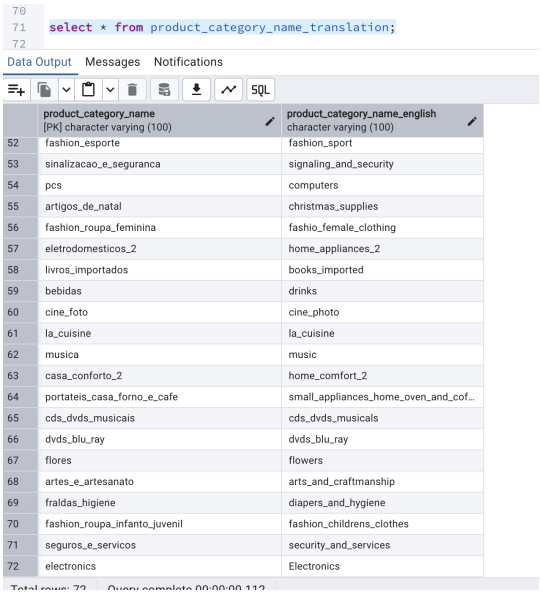


Fig. 9. Inserting Record

Validating the insertion



	product_category_name [PK] character varying (100)	product_category_name_english character varying (100)
52	fashion_esporte	fashion_sport
53	sinalizacao_e_seguranca	signaling_and_security
54	pcs	computers
55	artigos_de_natal	christmas_supplies
56	fashion_roupa_feminina	fashio_female_clothing
57	eletrodomesticos_2	home_appliances_2
58	livros_importados	books_imported
59	bebidas	drinks
60	cine_foto	cine_photo
61	la_cuisine	la_cuisine
62	musica	music
63	casa_conforto_2	home_comfort_2
64	portateis_casa_forno_e_cafe	small_appliances_home_oven_and_cof...
65	cds_dvds_musicais	cds_dvds_musicals
66	dvds_blu_ray	dvds_blu_ray
67	flores	flowers
68	artes_e_artesanato	arts_and_craftmanship
69	fraldas_higiene	diapers_and_hygiene
70	fashion_roupa_infanto_juvenil	fashion_childrens_clothes
71	seguros_e_servicos	security_and_services
72	electronics	Electronics

Fig. 10. Validation

Update Query 2 – Products

Updating record in Product table

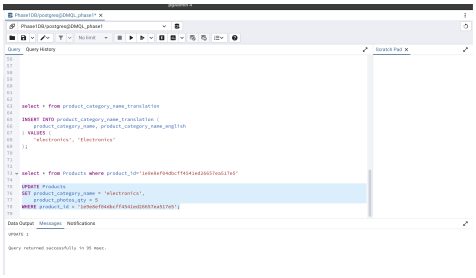


Fig. 11. Updating Product records

Validating the update query

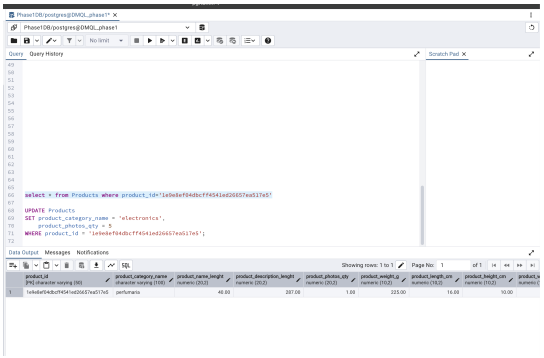


Fig. 12. Validation

Validating the update query

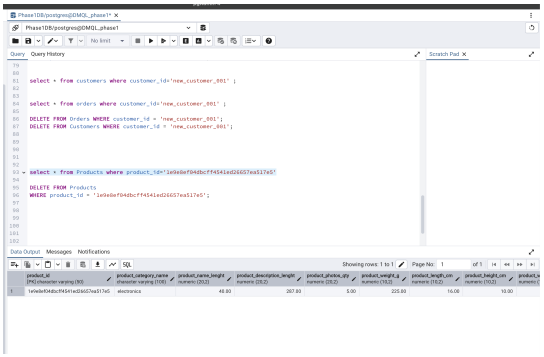


Fig. 13. Validation

Delete Query 2 – Orders and Customers

Delete queries on Orders and Customer table

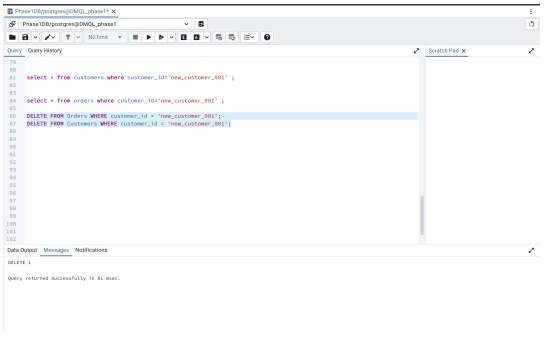


Fig. 14. Deleting order review records

Delete Query 3 – Products and order Items

Deleting records in Product and Order Items

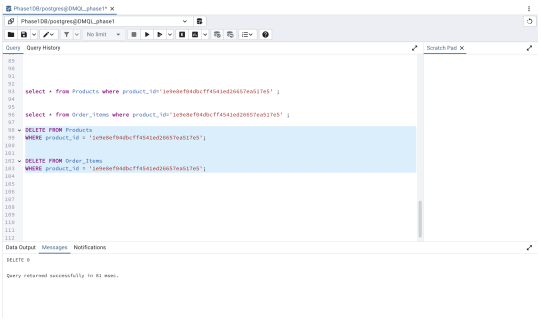


Fig. 15. Validation

SELECT Queries using Groupby, Orderby, Joins etc

We have demonstrated by using JOIN and ORDERBY

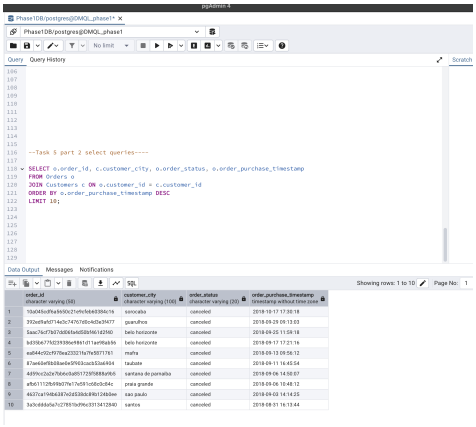


Fig. 16. JOIN and ORDERBY

JOIN clause demonstration

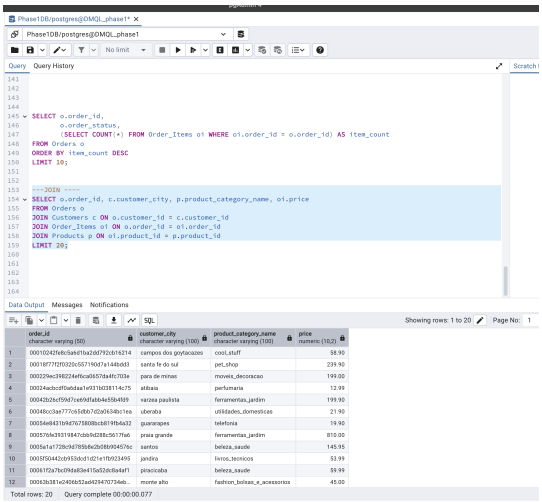


Fig. 17. JOIN

JOIN ,GROUPBY,ORDERBY Demonstration

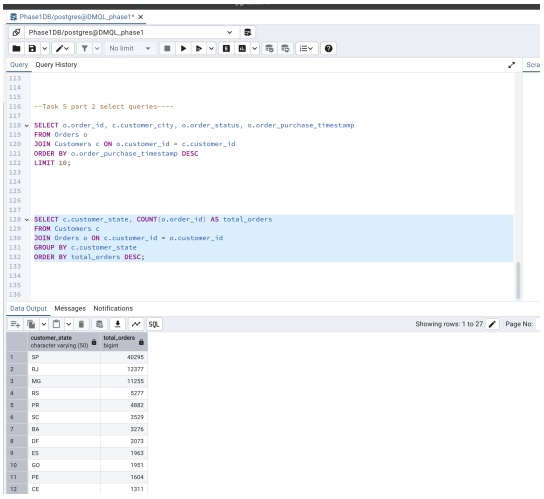


Fig. 18. JOIN, GROUPBY and ORDERBY

ORDERBY clause ,SUBQUERY,LIMIT Demonstration

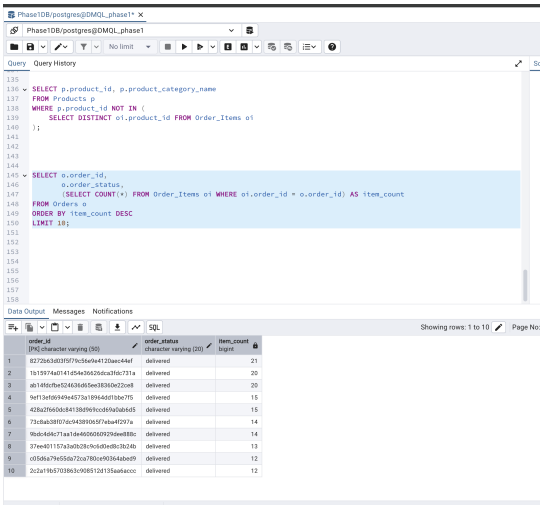


Fig. 19. ORDERBY, SUBQUERY,LIMIT

SUBQUERY Demonstration

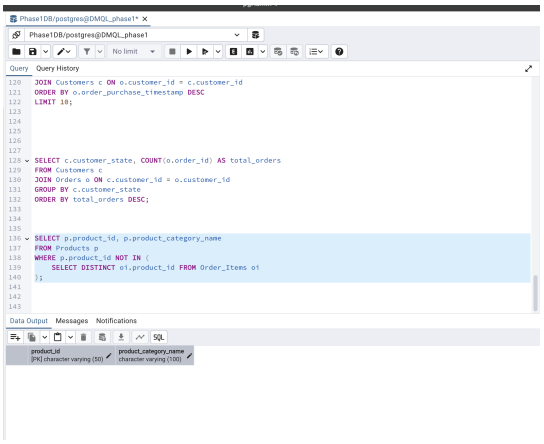


Fig. 20. SUBQUERY

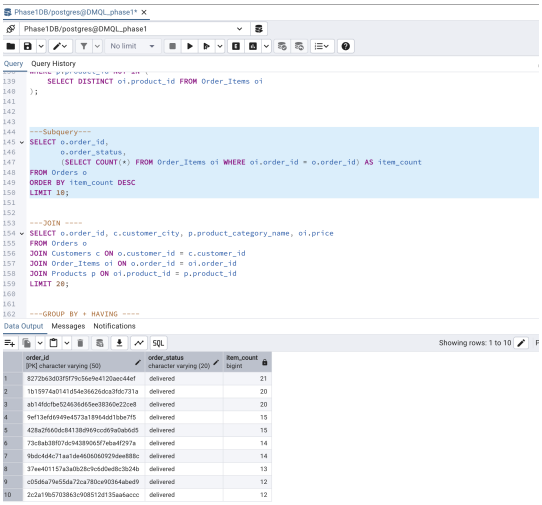


Fig. 21. SUBQUERY

WHERE Clause and SUBQUERY Demonstration

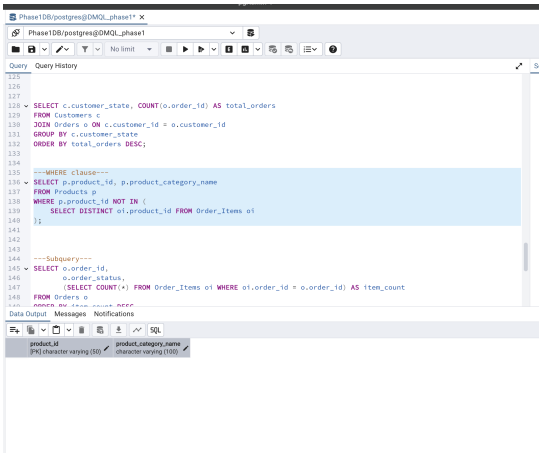


Fig. 22. WHERE and SUBQUERY

Procedures and Functions

The procedures help encapsulate operations like inserting, updating, or deleting records safely especially when foreign key constraints are involved.

Update Procedure

This function first inserts a new product into the Products table,then updates the same product to modify certain fields, such as product photos qty or category. we created this because this is useful when a product might need updates immediately after being created like correcting initial insertion errors or auto setting default fields.

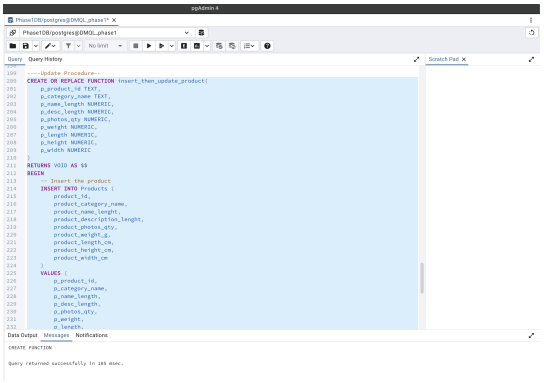


Fig. 23. Update Procedure

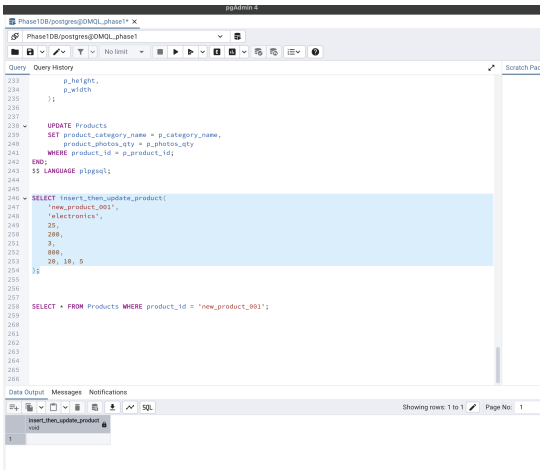


Fig. 24. Update Procedure

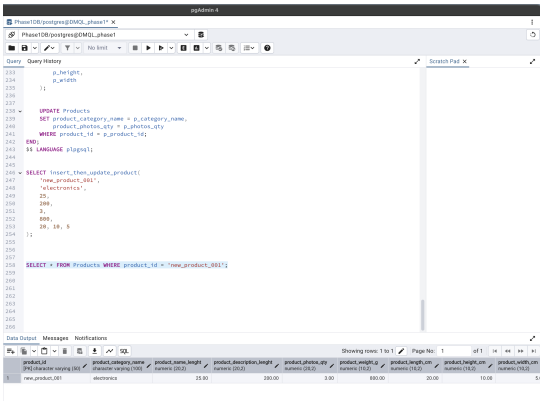


Fig. 25. Validation

Insert Procedure

This function inserts a new customer record into the Customers table in a reusable and safer way.we created this so that any new customer can be easily added using a function call without manually writing long insert queries every time.

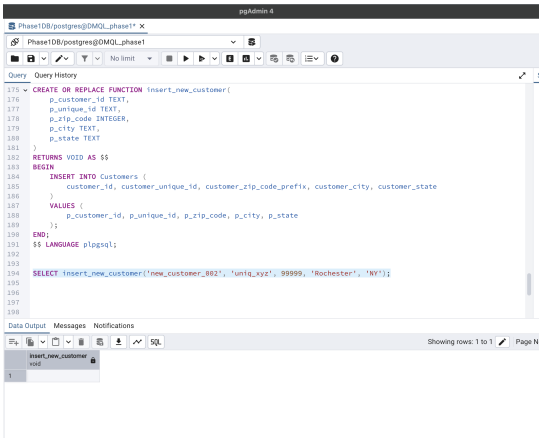


Fig. 26. Insertion procedure

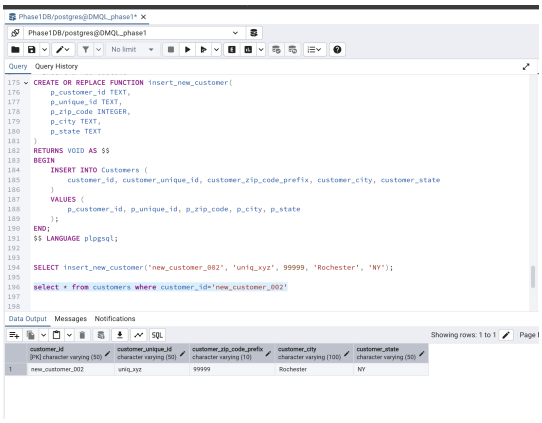


Fig. 27. Validation

Delete Procedure

Before deleting a product, this function checks whether that product is referenced in Order Items which is a foreign key.If it is referenced, it blocks the deletion and returns a warning message.If it is not referenced, it safely deletes the product.

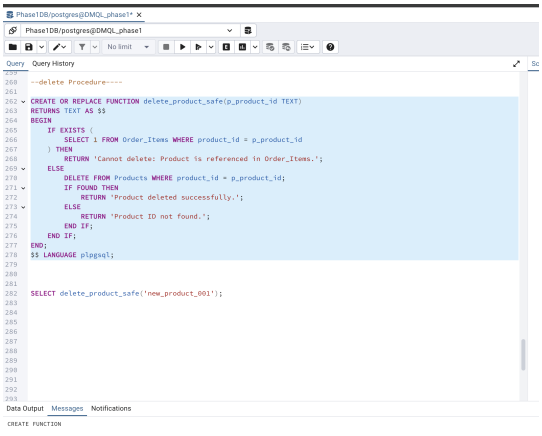


Fig. 28. Delete Procedure

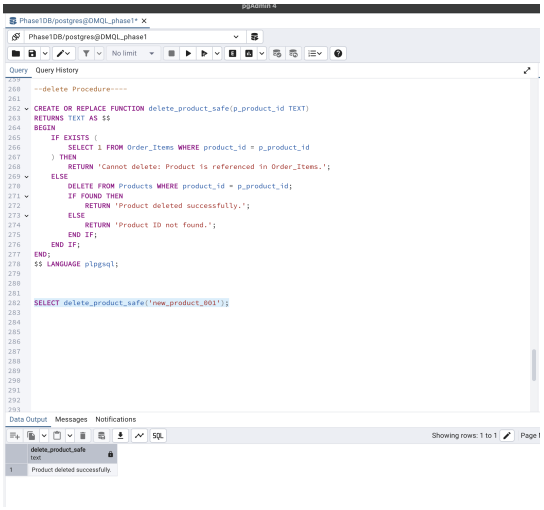


Fig. 29. Validation

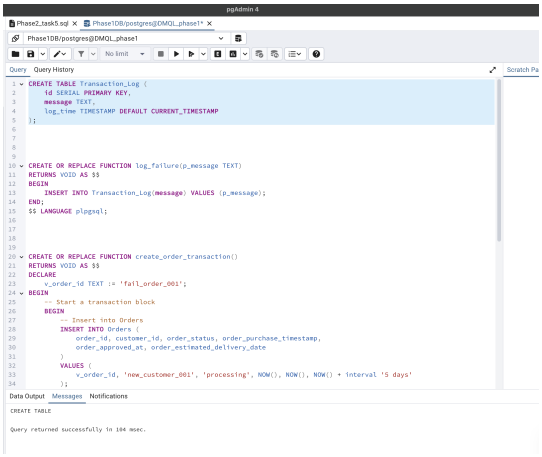


Fig. 31. Transaction log

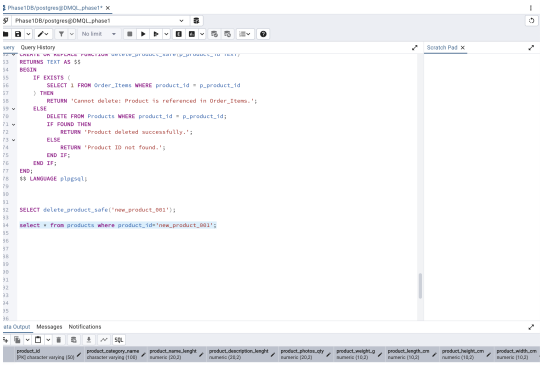


Fig. 30. Validation

Fig. 32. Transaction

TASK 6 TRANSACTION AND TRIGGERS

The goal of this task is to implement a transaction combining multiple related database operations and handle any transaction failures gracefully using a trigger mechanism. This ensures that database consistency and atomicity are preserved even under error conditions.

We implemented a transaction to insert an order into the Orders table and its corresponding items into the Order Items table. Here we have created a Transaction Block:If any of the INSERT operations fails due to foreign key constraint violation or invalid data, the transaction is automatically rolled back.

Failure Handling: A trigger-based failure handling mechanism was created using:a Transaction Log table to record failure events and a log failure function which inserts failure messages into the Transaction Log table.

To test the robustness an intentional failure is simulated by inserting an order item referencing a non-existent product ID.Now No partial records were inserted. The entire transaction was rolled back as expected and corresponding failure entry appeared in the Transaction Log table with the appropriate error message.

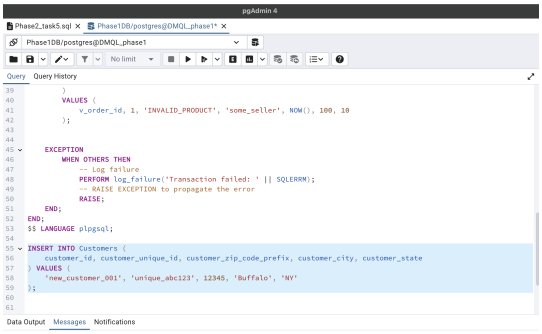


Fig. 33. Insertion

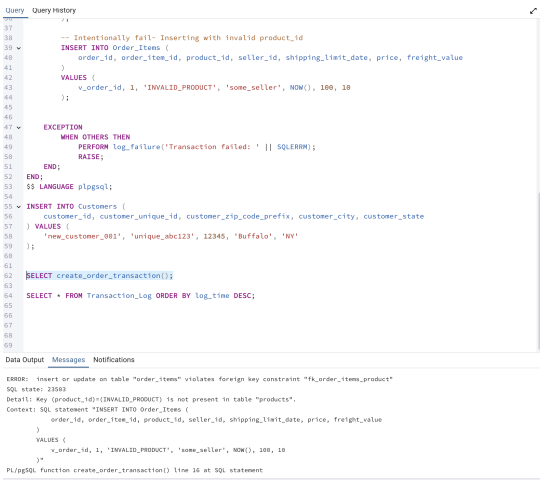


Fig. 34. Validation

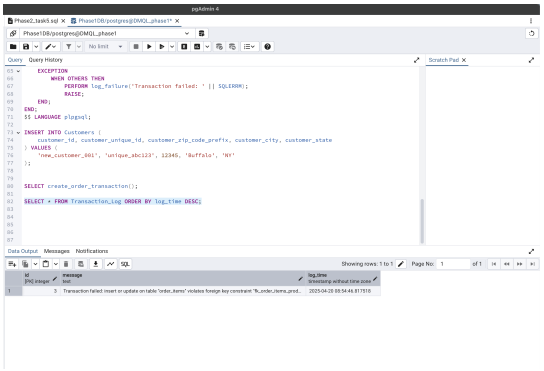


Fig. 35. Error logs

Justification: When the transaction encounters an error when inserting an invalid foreign key into the Order Items table, the database automatically aborts the entire transaction and None of the partially inserted records are saved.The database maintains consistency by performing a rollback.The log failure trigger captures the error and records it in the Transaction Log table for auditing purposes. This aligns with the principle of transaction atomicity, ensuring that either all operations succeed together or none are applied.Since our design constitutes all of the above points our design and safe and reliable,preserves data integrity

TASK 7: INDEXING AND QUERY EXECUTION ANALYSIS

The objective of this task was to identify expensive queries through query cost analysis, and to propose indexing strategies, to implement indexes, and verify performance improvements through EXPLAIN plans.

Problematic Query 1 – Frequent Lookups on product id ,Order Items Table

Before Creating Index the explain plan gives us the below result:

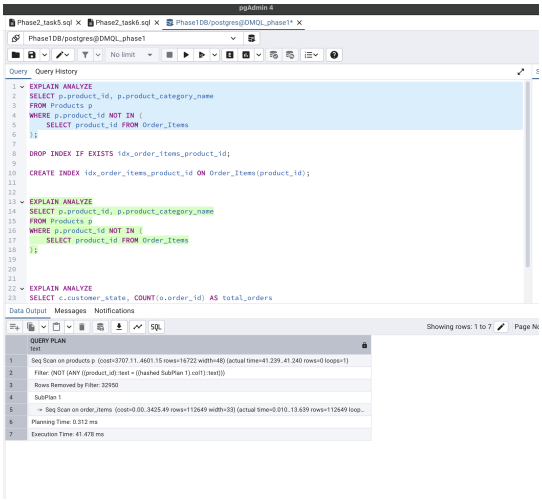


Fig. 36. Indexing on Order Items table

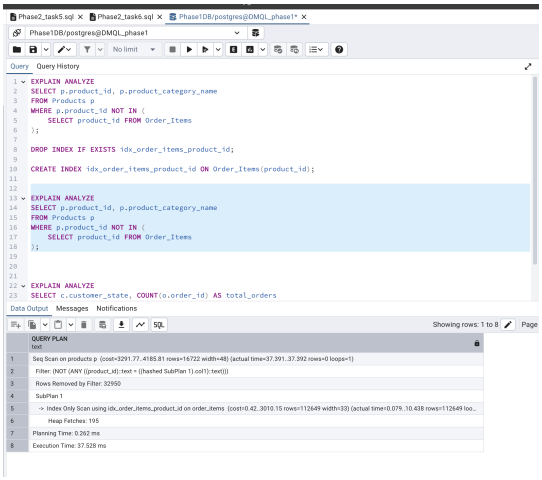


Fig. 37. Result

After Indexing the EXPLAIN Plan Summary shows us that index scan used on product id Since product id is frequently used in WHERE clauses for lookups, indexing reduced the search time from a full scan to a fast indexed retrieval the cost intially is around 3700 and after indexing it reduced to around 3200.

Problematic Query 2 – Customer Based Lookups

Before Creating Index the explain plan gives us the below result:

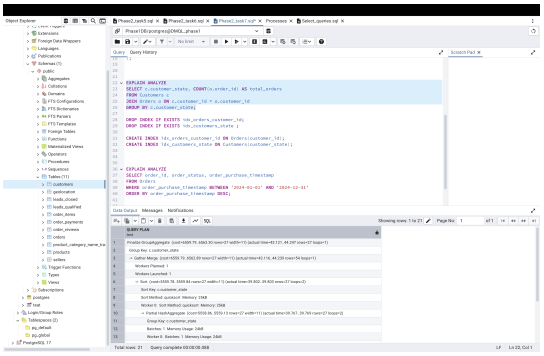


Fig. 38. Indexing on Order table

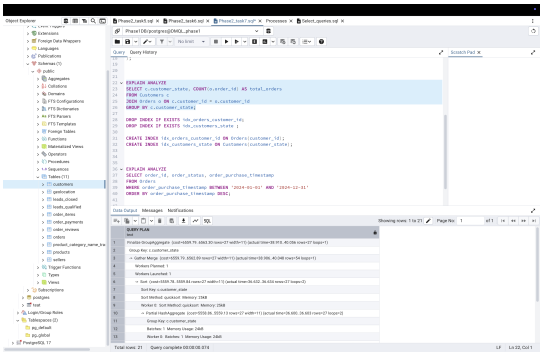


Fig. 39. Result

Customers often retrieve their orders by customer ID. Creating an index on customer id enables quick retrieval instead of scanning the entire Orders table. After Indexing the EXPLAIN Plan Summary shows us that index scan used on customer id.Since customer id is frequently used in WHERE clauses for lookups, indexing reduced the search time from a full scan to a fast indexed retrieval.

Problematic Query 3 – Filtering and Sorting by order purchase timestamp

Before Creating Index the explain plan gives us the below result:

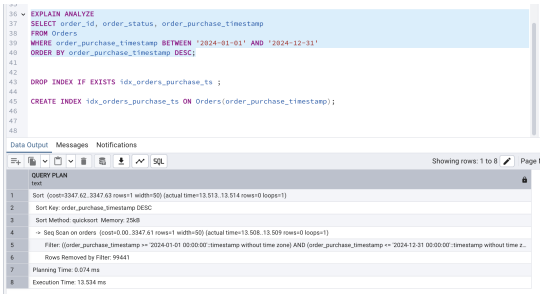


Fig. 40. Indexing on Order table

Fig. 41. Result

Before Indexing the EXPLAIN plan summary shows that it did a full sequential scan through all orders and sorting is performed after filtering. After Indexing the EXPLAIN plan summary shows that it did index only scan on order purchase timestamp and sorting became faster. Adding an index on order purchase timestamp improved performance for time-range queries and accelerated ORDER BY operations because the data is already partially sorted in the index.

I. Visualisations using Tableau

Orders by State

SQL Query:

```
SELECT customer_state,
COUNT(DISTINCT order_id)
AS total_orders
FROM customers
JOIN orders ON
customers.customer_id = orders.customer_id
GROUP BY customer_state
```

This visualization highlights the total number of orders placed from each state. Sao Paulo (SP) has the highest number of orders, followed by Rio de Janeiro (RJ) and Minas Gerais (MG). This regional concentration suggests that marketing efforts can be focused on Southeast Brazil for higher customer engagement.

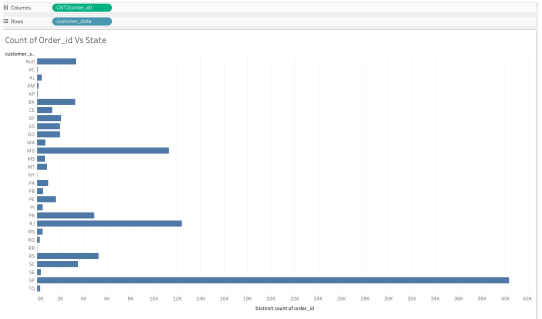


Fig. 42. Visualisation 1

Orders Over Time

```
SELECT YEAR(order_purchase_timestamp)
AS year, COUNT(DISTINCT order_id)
AS total_orders
FROM orders
```

```
GROUP BY year
ORDER BY year;
```

This line chart shows the growth of orders over time, with a noticeable increase in 2017. This indicates the platform's rapid growth and adoption by consumers during this period. Understanding these trends can help predict future growth and resource planning.

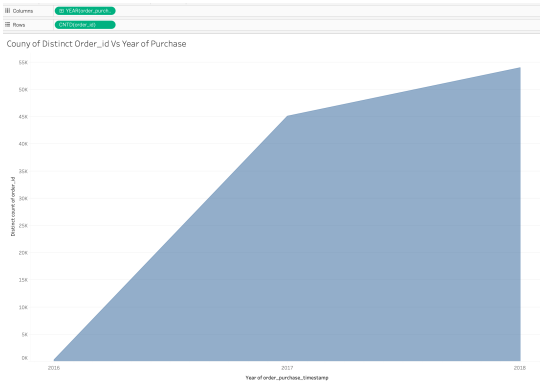


Fig. 43. Visualisation 2

Top Product Categories

```
SELECT product_category_name,
COUNT(*) AS total_orders
FROM order_items
GROUP BY product_category_name
ORDER BY total_orders DESC;
```

Product categories like beleza saude and informatica acessorios lead in order volume, indicating consumer preference for health-related and electronics products. These insights can guide inventory management and marketing strategies.

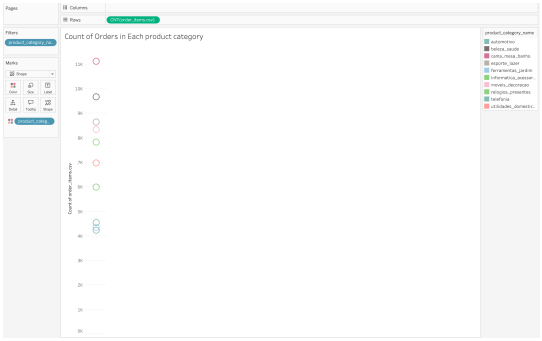


Fig. 44. Visualisation 3

Discounts by Category

```
SELECT product_category_name,
COUNT(DISTINCT price)
AS unique_price_points
FROM order_items
GROUP BY product_category_name
ORDER BY unique_price_points DESC;
```

The number of discounts for different product categories is represented in a Tableau bubble chart that derives its visual data from unique price values in order items. The visual presentation includes bubbles which represent individual categories of products and their sizes demonstrate the number of discounted items within each group. The product categories containing beleza saude, informatica acessorios and utilidades domesticas possess large bubble sizes which demonstrates their greater number of distinct discounted products. The couple of product categories brinquedos and ferramentas jardim present smaller bubble dimensions which indicates either a lack of many distinct prices or sparse discount offers. The analysis found 9179 different prices among recorded data throughout all categories.

Fig. 45. Visualisation 4

Orders Sold by Sellers

The visualization depicts seller order counts through this Tableau heatmap which displays the data using seller id 1 as identifier. Order item id sums up to show the complete item quantity sold by each seller. Each bar represents a seller who has its color scheme indicating their order volume numbers through intensity changes. Between them, one seller achieves 2,898 orders while multiple others reach between 250 and 1,200 orders. The graphic illustration indicates that seller performance levels are uneven because many transactions are handled by merely a handful of sellers. A total of 60,795 order items have been reported by the top 100 sellers.

```
SELECT seller_id_1, SUM(order_item_id)
AS total_orders_sold
FROM order_items
GROUP BY seller_id_1
ORDER BY total_orders_sold DESC;
```

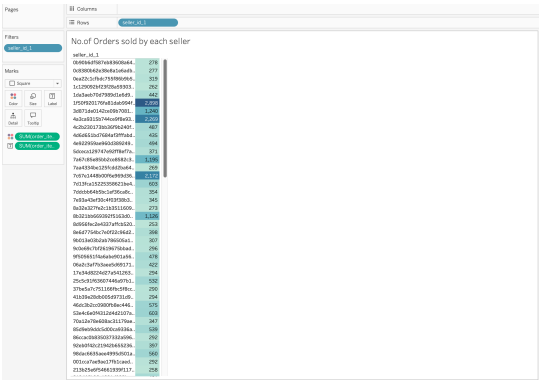


Fig. 46. Visualisation 5

Average Price of items

Using order items data the Tableau tool shows average prices of top 10 products through a stacked bar chart format. The vertical sizes of colorful bar segments stand for different product categories while showing their average pricing information. Items found in pcs and climatizacao together with construcao ferramente possess elevated average costs comparing to lower-priced products within telefonia fixa and relogios presentes. The cumulative average value amounts to 3,924.4 dollars through the combined analysis of multiple categories. The graphical representation lets companies notice which product segments exist in the premium category by reflecting their pricing structure.

```
SELECT product_category_name,
AVG(price) AS avg_price
FROM order_items
GROUP BY product_category_name
ORDER BY avg_price DESC
LIMIT 10;
```

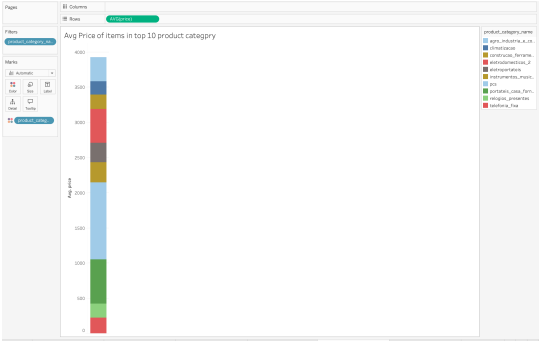


Fig. 47. Visualisation 6

Revenue generated in top categories

The revenue picture depicted by Tableau treemap demonstrates how the top 20 product categories generate revenue through item price summations in the order items database. The visualization displays categories through rectangles that present their total revenue worth through size measurements. The three product groups beleza saude, relogios presentes and esporte lazer rule the market with the largest amount of revenue. The highest manufacturing revenue comes from three main groups: informatica acessorios, cama mesa banho, and utilidades domesticas. The pcs pet shop and eletroportateis rectangles illustrate categories with a lower level of revenue. The leading 20 retail product categories together produce approximately 11,418,780 dollars in total revenue.

```
SELECT product_category_name,
SUM(price) AS total_revenue
FROM order_items
GROUP BY product_category_name
ORDER BY total_revenue DESC
LIMIT 20;
```

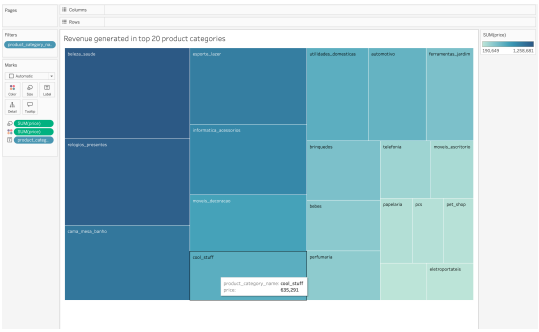


Fig. 48. Visualisation 7

Avg no of orders in top product categories

The average order counts for each product category are displayed in this bubble chart using AVG(order item id) as the metric. The visual representation shows each blue circle as a product category while the circle size illustrates the average order statistics for these categories. Among the categories, the average number of placed orders stand highest in telefonia fixa and moveis sala and climatizacao while the sizes of perfumaria and beleza saude along with eletronicos fall at intermediate levels. The displayed graphic presents an even distribution of demand levels among different product categories. As displayed category data adds up to 62.67, the order intensity varies widely.

```
SELECT product_category_name,
AVG(order_item_id) AS avg_orders
FROM order_items
GROUP BY product_category_name;
```

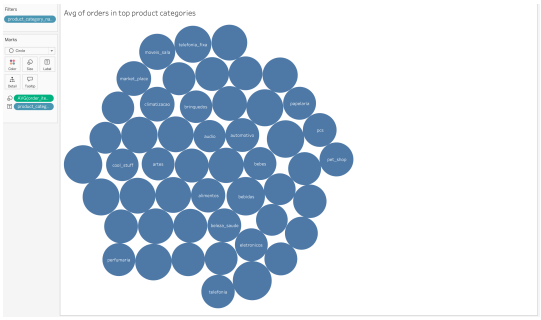


Fig. 49. Visualisation 8

The below are the link for the dashboard visualisations
1)View Dashboard 1
2)View Dashboard 2
3)View Dashboard 3

CONCLUSION

This Brazilian E-Commerce Database Design is an organized relational schema that can model much of an online market database is able to manage large-scale transactional data, seller and buyer activity, and marketing analysis with data integrity, scalability, and performance.

Through a careful analysis of relationships between entities, 11 tables were designed, which include the customers, orders, products, payments, reviews, sellers, geolocation information, and marketing leads. all tables were designed with care using primary keys foreign keys and referential integrity constraints so that database consistency could be maintained.the assertion continued: e ntity relationship modeling to decide cardinality and create relationships among tables.

Specify attribute data like purpose, type, and constraints for every column. foreign key constraints and cascade operations for primary data integrity. Normalization rules for data redundancy reduction and query optimization.Optimal best practice SQL schema script deployment for optimal table structure. the proposed database schema is ideal for efficient retrieval, analysis, and business intelligence to decision makers like sellers, buyers, and business managers. The structured solution addresses spreadsheet data management issues with secure, stable, and query-optimized transaction processing.

Through analysis and real-world implementation, we were able to effectively illustrate key aspects of relational database management in Phase 2 of the project. Through validation procedures following each operation, we ensured referential integrity while performing data manipulation operations like insertion, updating, and deletion across several related tables.

To extract valuable information from the dataset, sophisticated SQL queries were created and run using `JOIN`, `GROUP BY`, `HAVING`, `ORDER BY`, and subqueries. Additionally, to improve modularity and code maintainability, reusable stored procedures and functions were created to encapsulate common database operations.

A composite transaction with exception handling was used for transaction management. In order to enforce atomicity and consistency in the database even in the event of a failure, a specific trigger function was developed to log transaction failures into an audit table.

Furthermore, indexing techniques were used to optimize expensive queries, which greatly shortened execution times, as confirmed by `EXPLAIN ANALYZE` plans. The significance of appropriate indexing in improving database performance was highlighted by these optimizations.

Lastly, Tableau Public was used to create an interactive business intelligence dashboard. Key trends and patterns in the dataset were displayed on the dashboard, offering useful information for making important choices.

ACKNOWLEDGMENT

We would like to express our heartfelt gratitude to our professor,Dr. Shamsad Parvin, for teaching us various tools and techniques essential for this project. Your insights into the importance of conducting this type of research have been invaluable in shaping our understanding and approach. Thank you for your guidance and support throughout this journey.

CONTRIBUTION

Everyone has equal contribution in this project phase 1

Member	Percentage
jkollu	33.33%
lkodali	33.33%
smandru	33.33%

REFERENCES

[1] https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce
[2] https://www.postgresql.org/docs/
[3] Database System Concepts 7th Edition — Abraham Silberschatz, Henry F. Korth, S. Sudarshan
[4] Lecture Slides