OLIST SQL PROJECT:

E-COMMERCE DATA ANALYSIS

INDEX

1.INTRODUCTION

This project focuses on performing SQL-based analysis on an e-commerce platform's historical data. The primary goal is to gain deep insights into customer behavior, product trends, seller performance, and delivery efficiency using relational data analysis.

The dataset includes multiple interconnected tables, such as orders, products, customers, sellers, and payments, which represent different aspects of the business operations. Using SQL, we perform structured analysis across these datasets to solve real business problems.

Purpose of the Project

The objective of this project is to:

- **Understand customer behavior** and purchasing patterns across different regions.
- **Analyze order fulfilments** by evaluating delivery rates and average delivery times.
- **Identify high-performing sellers** and top-selling product categories.
- **Evaluate payment trends** to understand customer preferences in transaction methods.
- **Clean and prepare** raw datasets to ensure data integrity for accurate analysis.

The outcome of this analysis will support data-informed strategies for marketing, operations, inventory management, logistics, and customer service optimization.

Scope of Work

This project is divided into the following major phases:

1. **Exploratory Data Review:** Understanding the structure, content, and relationships of multiple datasets (orders, payments, products, customers, etc.).
2. **Data Cleaning & Transformation:** Creating filtered and standardized versions of original tables to remove anomalies and ensure consistency.
3. **SQL Analysis:** Writing and executing queries to answer real business questions, such as:
    - Which products generate the most revenue?
    - Which sellers drive the highest sales?
    - What is the most used payment method?
    - How efficient is the delivery process?
4. **Insight Documentation:** Presenting results visually and descriptively with interpretations and recommendations.

Technology Used

- **SQL (PostgreSQL flavor):** For data manipulation and querying.
- **DB Management Tool (e.g., pgAdmin):** For executing and visualizing queries.
- **ERD Tool / Diagram Editor:** For designing the entity relationship model.

Challenges Faced

Working with real-world e-commerce data presented several challenges:

1. **Data Inconsistencies:**
   - Variations in text formatting (e.g., state abbreviations in mixed case).
   - Presence of null values in critical fields like order_id, product_id.
2. **Missing or Redundant Records:**
   - Duplicates and empty records required careful filtering and deduplication using DISTINCT and WHERE clauses.
3. **Multi-table Joins and Relationships:**
   - Understanding the complex relationships between customers, orders, products, and sellers required a well-planned ERD.
4. **Volume and Complexity:**
   - The size and interconnectedness of tables demanded efficient querying strategies to ensure performance and accuracy.
5. **Time-Based Calculations:**
   - Calculating delivery time accurately involved timestamp conversions and interval calculations.

Expected Outcomes

By the end of this project, we expect to have:

- A clear overview of sales trends and customer distribution.
- Key metrics on order processing and delivery efficiency.
- Identification of top-performing sellers and categories.
- Recommendations for business improvements based on data-backed insights.

2. ENTITY RELATIONSHIP DIAGRAM (ERD)

The ERD (Entity Relationship Diagram) visually represents how different tables in the database are connected through primary and foreign keys. It helps in understanding the data structure and planning queries effectively.

Entities & Relationships:

- **Customers** place **Orders**
- Each **Order** has **Order Items**
- **Order Items** reference **Products**
- **Orders** are linked to **Payments**
- **Products** belong to **Categories**
- **Sellers** fulfil **Order Items**
- **Geolocation** links to zip codes of both customers and sellers

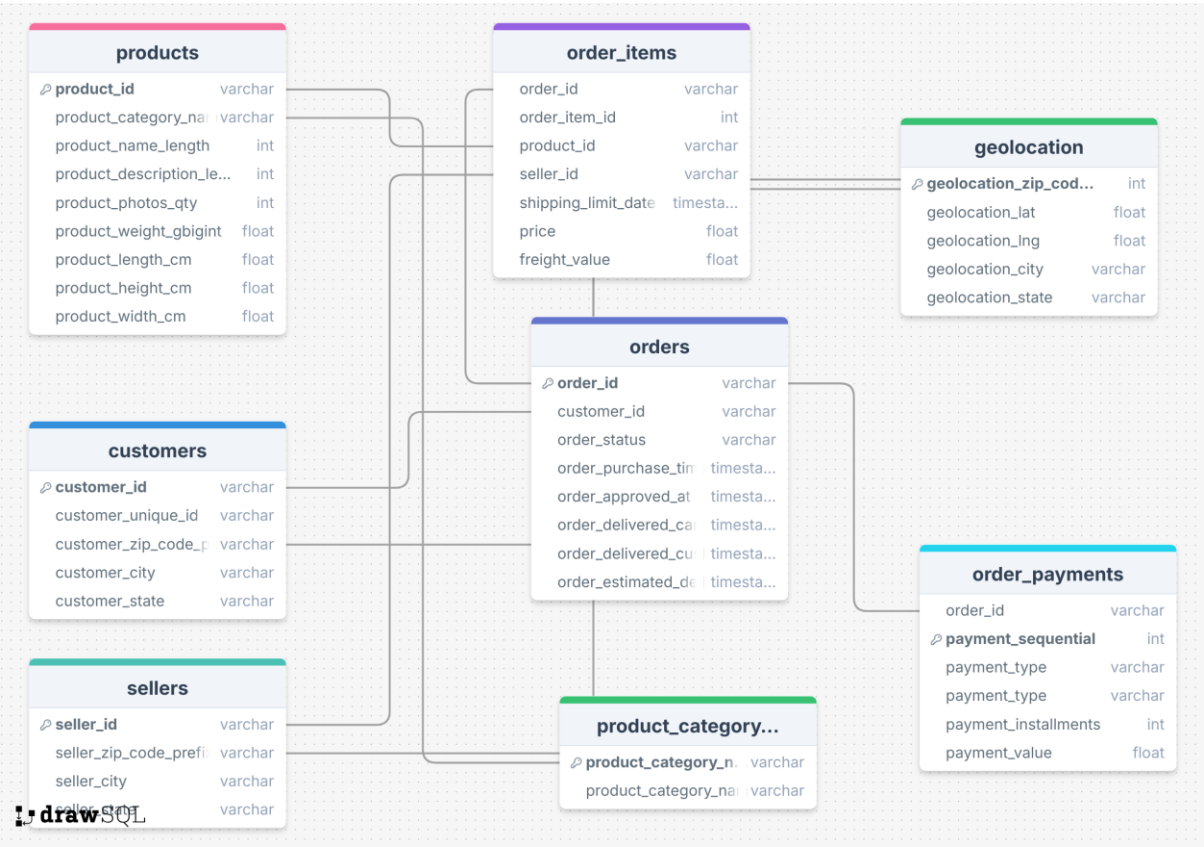

Fig 2.1 Entity Relationship Diagram

3. DATA CLEANING & TRANSFORMATION

Raw data often contains inconsistencies such as:

- Null values
- Duplicates
- Irregular formatting (e.g., mixed-case states or cities)
- Invalid entries (like zero prices or weights)

Before diving into analysis, it's essential to create **clean versions** of the tables by applying filters, removing nulls, standardizing formats, and ensuring logical integrity. This ensures all insights generated are **reliable** and **accurate**.

Cleaning Steps:

1. **clean_customers**
    - Removed nulls
    - Standardized city to lowercase and state to uppercase

```sql
CREATE TABLE clean_customers AS
SELECT DISTINCT
    customer_id,
    customer_unique_id,
    TRIM(LOWER(customer_city)) AS customer_city,
    TRIM(UPPER(customer_state)) AS customer_state,
    customer_zip_code_prefix
FROM customers
WHERE customer_id IS NOT NULL;
```

2. **clean_orders**
    - Filtered out orders with missing IDs, customer IDs, or status

```sql
CREATE TABLE clean_orders AS
SELECT DISTINCT *
FROM orders
WHERE customer_id IS NOT NULL
  AND order_status IS NOT NULL
  AND order_id IS NOT NULL;
```

3. **clean_order_items**
    - Ensured non-null order/product IDs
    - Kept only items with positive prices

```sql
CREATE TABLE clean_order_items AS
SELECT DISTINCT *
FROM order_items
WHERE order_id IS NOT NULL
  AND product_id IS NOT NULL
  AND price > 0;
```

4. **clean_order_payments**
    - Removed nulls
    - Excluded zero-value transactions

```sql
CREATE TABLE clean_order_payments AS
SELECT DISTINCT *
FROM order_payments
WHERE payment_value > 0
  AND payment_type IS NOT NULL
  AND order_id IS NOT NULL;
```

5. **clean_products**
   - Removed products with missing IDs or weights
   - Ensured valid product category names

```sql
CREATE TABLE clean_products AS
SELECT DISTINCT *
FROM products
WHERE product_id IS NOT NULL
  AND product_weight_g > 0
  AND product_category_name IS NOT NULL;
```

6. **clean_product_category_translation**
   - Ensured valid translations (both original and English names)

```sql
CREATE TABLE clean_product_category_translation AS
SELECT DISTINCT *
FROM product_category_translation
WHERE product_category_name IS NOT NULL
  AND product_category_name_english IS NOT NULL;
```

7. **clean_sellers**
   - Cleaned and standardized seller city/state like customers

```sql
CREATE TABLE clean_sellers AS
SELECT DISTINCT
    seller_id,
    TRIM(LOWER(seller_city)) AS seller_city,
    TRIM(UPPER(seller_state)) AS seller_state,
    seller_zip_code_prefix
FROM sellers
WHERE seller_id IS NOT NULL;
```

8. **clean_geolocation**
   - Filtered for valid lat/long and zip code prefixes

```sql
CREATE TABLE clean_geolocation AS
SELECT DISTINCT *
FROM geolocation
WHERE geolocation_zip_code_prefix IS NOT NULL
  AND geolocation_lat IS NOT NULL
  AND geolocation_lng IS NOT NULL;
```

ANALYSIS 4: ORDER INSIGHTS

4.1 List of Available Payment Methods

ProblemStatement:
Identify all distinct payment methods available in the dataset to understand the range of customer payment preferences.

Approach:

- Query the payment table.
- Use DISTINCT to eliminate duplicate records and list unique values in the payment_type column.

Analysis:
The query helps in understanding the types of transactions customers are most comfortable using. This insight can guide future integrations or promotions.

Recommendation:
Ensure all popular payment methods remain supported, and consider introducing newer digital wallets to enhance flexibility.

Query Query History

```sql
---Which Payment method is most used---
SELECT payment_type , COUNT(*) AS usage_count
FROM clean_order_payments
GROUP BY payment_type
ORDER BY usage_count desc
```

Data Output Messages Explain Notifications

Showing rows: 1 to 4 Page No: 1 of 1

	payment_type character varying	usage_count bigint
1	credit_card	76795
2	boleto	19784
3	voucher	5769
4	debit_card	1529

4.2 Order Status Distribution

ProblemStatement:
Determine how many orders fall into each status (e.g., delivered, shipped, canceled).

Approach:

- Group records from the orders table using order_status.
- Use COUNT() to get the frequency of each status.

Analysis:
This identifies operational efficiency and highlights problem areas like cancellations or processing delays.

Recommendation:
Track status changes in real-time and implement interventions for delayed or canceled orders to improve fulfilments rates.

Query Query History

```sql
---Total number of orders per order status---
SELECT order_status, count(*)
FROM clean_orders
GROUP BY order_status
```

Data Output Messages Explain Notifications

Showing rows: 1 to 8 Page No: 1 of 1

	order_status character varying	count bigint
1	shipped	1107
2	unavailable	609
3	invoiced	314
4	created	5
5	approved	2
6	processing	301
7	delivered	96478
8	canceled	625

4.3 Total Number of Orders

ProblemStatement:
Find the total number of orders to understand the dataset's transaction volume.

Approach:

- Count the number of records in the orders table using COUNT(*).

Analysis:
Serves as a benchmark for further calculations such as delivery rate and customer engagement.

Recommendation:
Use total orders as a base metric for performance KPIs and planning for scalability.

Query Query History

```
---Total Number of orders---
SELECT COUNT(*)
FROM clean_orders;
```

Data Output Messages Explain Notifications

Showing rows: 1 to 1 Page No: 1 of 1

	count bigint
1	99441

4.4 Total Number of Delivered Orders

ProblemStatement:
Count how many orders have been successfully delivered to measure order fulfilment success.

Approach:

- Filter the orders table for order_status = 'delivered'.
- Use COUNT() to calculate the total.

Analysis:
Provides insights into delivery performance. A high ratio of delivered orders indicates a smooth logistics process.

Recommendation:
Maintain or improve delivery standards by monitoring courier performance and warehouse processes.

5. ANALYSIS 2: PAYMENT AND DELIVERY ANALYSIS

5.1 Most Frequently Used Payment Method

ProblemStatement:
Identify the most commonly used payment method by customers.

Approach:

- Group the order_payments data by payment_type.
- Use COUNT() to find frequency.
- Order by count descending.

Analysis:
Shows customer preference trends and trust in specific payment types (e.g., credit card dominance).

Recommendation:
Promote frequently used payment methods with cashback or loyalty offers and ensure a smooth checkout process for them.

Query Query History

```
---list of all distinct payment methods---
SELECT DISTINCT payment_type
FROM clean_order_payments
```

Data Output Messages Explain Notifications

Showing rows: 1 to 4 Page No: 1 of 1

	payment_type character varying
1	boleto
2	debit_card
3	voucher
4	credit_card

5.2 Average Delivery Time

ProblemStatement:
Calculate the average time it takes from order placement to delivery.

Approach:

- Use DATEDIFF or time subtraction between order_purchase_timestamp and order_delivered_customer_date.
- Apply AVG() to get the mean duration.

Analysis:
This metric reflects logistics performance and affects customer satisfaction.

Recommendation:
If average delivery time exceeds industry standards, invest in faster shipping partners or regional warehouses.

Query Query History

```
---What is the average delivery---
--time for delivered orders--
SELECT AVG(order_delivered_customer_date - order_purchase_timestamp) as order_status
FROM clean_orders
WHERE order_status = 'delivered'
```

Data Output Messages Explain Notifications

Showing rows: 1 to 1 Page No: 1 of 1

	order_status interval
1	12 days 13:23:49.957272

5.3 Top Product Categories by Revenue

ProblemStatement:
Identify which product categories bring in the most revenue.

Approach:

- Join order_items with products and product_category_translation.
- Use SUM(price * quantity) and group by category.

Analysis:
Reveals best-selling product segments. Useful for inventory and promotion planning.

Recommendation:
Allocate more marketing budget to high-performing categories and optimize stock levels accordingly.

Query Query History

```sql
---Which Product categories generate
---the highest revenue
SELECT pct.product_category_name_english,
       SUM(oi.price) AS total_revenue
FROM clean_order_items oi
JOIN clean_products p ON oi.product_id = p.product_id
JOIN clean_product_category_translation pct ON p.product_category_name = pct.product_category_name
GROUP BY pct.product_category_name_english
ORDER BY total_revenue DESC
LIMIT 5;
```

Data Output Messages Explain × Notifications

Showing rows: 1 to 5 Page No: 1 of 1

	product_category_name_english character varying	total_revenue double precision
1	health_beauty	1258681.340000017
2	watches_gifts	1205005.680000013
3	bed_bath_table	1036039.1800000409
4	sports_leisure	988048.9700000199
5	computers_accessories	911954.3200000154

6. ANALYSIS 3: SELLER & CUSTOMER PATTERNS

6.1 Top Revenue-Generating Sellers

ProblemStatement:
Determine which sellers have generated the most revenue.

Approach:

- Group by seller_id in the order_items table.
- Sum total revenue using price * quantity.

Analysis:
Highlights seller performance and can help identify reliable vendors for partnerships or promotions.

Recommendation:
Provide high-performing sellers with visibility perks and consider onboarding similar profiles

Query Query History

```sql
---Who are the top 5 sellers by revenue
SELECT seller_id,sum(price) as total_sales
FROM clean_order_items
GROUP BY seller_id
ORDER BY total_sales desc
limit 5
```

Data Output Messages Explain × Notifications

Showing rows: 1 to 5 Page No: 1 of 1

	seller_id character varying	total_sales double precision
1	4869f7a5dfa277a7dca6462dcf3b52b2	229472.6299999981
2	53243585a1d6dc2643021fd1853d89...	222776.04999999952
3	4a3ca9315b744ce9f8e9374361493884	200472.9199999949
4	fa1c13f2614d7b5c4749cbc52fecda94	194042.02999999846
5	7c67e1448b00f6e969d365cea6b010ab	187923.8899999995

6.2 Customer Distribution by State

ProblemStatement:
Understand the geographic distribution of customers across states.

Approach:

- Group customers by customer_state.
- Count the number of customers in each group.

Analysis:
This shows where most of the customer base is concentrated, aiding in targeted marketing and regional logistics planning.

Recommendation:
Focus outreach in states with high customer density and explore growth opportunities in underrepresented areas.

Query Query History

```sql
---Which states have the most customers---
SELECT customer_state, COUNT(*) AS customer_count
FROM clean_customers
GROUP BY customer_state
ORDER BY customer_count DESC
LIMIT 5
```

Data Output Messages Explain Notifications

Showing rows: 1 to 5 Page No: 1 of 1

	customer_state text	customer_count bigint
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045