

E-Commerce Sales Analysis using SQL & Python

by
Boga Vivek

Overview

This project analyzes e-commerce transactions using **SQL and Python** to derive insights into customer behavior, product performance, and payment trends. The dataset includes customer details, order records, payments, and product information. SQL is used for data extraction, while Python (Pandas, Matplotlib, and Seaborn) is used for analysis and visualization.

Table of Contents

Overview.....	1
Dataset Description.....	2
1. customers.csv.....	2
2. geolocation.csv.....	3
3. orders.csv.....	3
4. order_items.csv.....	3
5. payments.csv.....	3
6. products.csv.....	4
7. sellers.csv.....	4
Technologies Used.....	4
Project Workflow.....	5
SQL Queries Used.....	5
Basic Queries.....	6
1. List all unique cities where customers are located.....	6
2. Count the number of orders placed in 2017.....	6
3. Find the total sales per category.....	6
4. Calculate the percentage of orders that were paid in installments.....	6
5. Count the number of customers from each state.....	6
Intermediate Queries.....	6
1. Calculate the number of orders per month in 2018.....	6
2. Find the average number of products per order, grouped by customer city.....	6
3. Calculate the percentage of total revenue contributed by each product category.....	7
4. Identify the correlation between product price and the number of times a product has been purchased.....	7
5. Calculate the total revenue generated by each seller, and rank them by revenue.....	7
Advanced Queries.....	7
1. Calculate the moving average of order values for each customer over their order history.....	7
2. Calculate the cumulative sales per month for each year.....	8
3. Calculate the year-over-year growth rate of total sales.....	8
4. Calculate the retention rate of customers, defined as the percentage of customers who make another purchase within 6 months of their first purchase....	8
5. Identify the top 3 customers who spent the most money each year.....	9
Results & Key Findings.....	9
Conclusion.....	12
Future Improvements.....	13
GitHub Link.....	13

Dataset Description

The dataset consists of multiple CSV files containing structured e-commerce transaction data. Below is a description of each dataset:

1. customers.csv

Contains unique customer IDs and their location details. Helps in analyzing customer distribution and order patterns.

Key Columns:

- **customer_id** – Unique identifier for each customer.
- **customer_unique_id** – Maps multiple orders to the same customer.
- **customer_city** – Customer's city.
- **customer_state** – Customer's state.

2. geolocation.csv

Provides latitude and longitude data for locations. Useful for mapping customer and seller distributions.

Key Columns:

- **geolocation_zip_code_prefix** – ZIP code of the location.
- **geolocation_lat** – Latitude coordinate.
- **geolocation_lng** – Longitude coordinate.

3. orders.csv

Contains order-level information, including timestamps and status. Helps analyze order trends over time.

Key Columns:

- **order_id** – Unique order identifier.
- **customer_id** – Links order to a customer.
- **order_status** – Status of the order (delivered, canceled, etc.).
- **order_purchase_timestamp** – When the order was placed.

4. order_items.csv

Details of products within each order. Helps analyze top-selling products and order values.

Key Columns:

- **order_id** – Links to the orders table.
- **product_id** – Identifies which product was purchased.
- **seller_id** – Identifies the seller of the product.
- **price** – Price of the product.
- **freight_value** – Shipping cost.

5. payments.csv

Records payment transactions for orders. Helps analyze preferred payment methods.

Key Columns:

- **order_id** – Links to the orders table.
- **payment_sequential** – Indicates multiple transactions per order.
- **payment_type** – Payment method (credit card, boleto, etc.).
- **payment_installments** – Number of installments chosen by the customer.
- **payment_value** – Total amount paid.

6. products.csv

Contains product descriptions and category information. Helps analyze product popularity and pricing.

Key Columns:

- **product_id** – Unique identifier for each product.
- **product_category_name** – Category of the product.
- **product_weight_g** – Weight of the product.
- **product_length_cm** – Length of the product.

7. sellers.csv

Contains seller details and their location information. Helps analyze seller distribution and performance.

Key Columns:

- **seller_id** – Unique identifier for each seller.
- **seller_zip_code_prefix** – Seller's ZIP code.
- **seller_city** – Seller's city.
- **seller_state** – Seller's state.

Technologies Used

This project leverages the following technologies for data extraction, analysis, and visualization:

- **SQL** – Used for querying structured datasets and extracting meaningful insights.
- **MySQL** – Stores and manages structured e-commerce data.
- **Python** – Employed for data processing and analysis.
 - **Pandas** – For data cleaning, manipulation, and transformation.
 - **Matplotlib** – For creating visual representations of sales trends and patterns.
 - **Seaborn** – Used for advanced statistical data visualization and trend analysis.

These technologies collectively help in extracting, processing, and visualizing e-commerce data efficiently.

Project Workflow

1. Data Exploration

- Understanding the dataset structure, relationships, and missing values.
- Exploring the different tables and their attributes.

2. SQL Queries

- Extracting insights using SQL queries.
- Filtering, grouping, and aggregating data to analyze sales trends, customer activity, and payment methods.

3. Python Analysis

- Cleaning and transforming data using **Pandas**.
- Visualizing data using **Matplotlib** and **Seaborn** to identify patterns.
- Summarizing key findings for better decision-making.

4. Key Insights

- **Sales Trends:** Identifying seasonal demand fluctuations.
- **Customer Behavior:** Analyzing order frequency and geographic distribution.
- **Payment Method Analysis:** Understanding customer preferences in payment types.
- **Product Performance:** Identifying top-selling products and underperforming categories.

SQL Queries

This project involved executing various SQL queries to extract meaningful insights from the e-commerce dataset. The queries are categorized into **Basic, Intermediate, and Advanced levels** based on complexity.

Basic Queries

1. List all unique cities where customers are located.

SQL Query: `select distinct customer_city from customers`

2. Count the number of orders placed in 2017.

SQL Query: `select count(order_id) from orders
where year(order_purchase_timestamp) = 2017`

3. Find the total sales per category.

SQL Query: `select upper(products.product_category) category,
round(sum(payments.payment_value),2) sales
from products join order_items
on products.product_id = order_items.product_id
join payments
on payments.order_id = order_items.order_id
group by category`

4. Calculate the percentage of orders that were paid in installments.

SQL Query: `select (sum(case when payment_installments >=1 then 1
else 0 end))/count(*)*100 from payments`

5. Count the number of customers from each state.

SQL Query: `select customer_state, count(customer_id)
from customers group by customer_state`

Intermediate Queries

1. Calculate the number of orders per month in 2018.

SQL Query: `select monthname(order_purchase_timestamp) months,
count(order_id) order_count
from orders where year(order_purchase_timestamp) = 2018
group by months`

2. Find the average number of products per order, grouped by customer city.

SQL Query: `with count_per_order as
(select orders.order_id, orders.customer_id, count(order_items.order_id) as oc
from orders join order_items
on orders.order_id = order_items.order_id
group by orders.order_id, orders.customer_id)

select customers.customer_city, round(avg(count_per_order.oc),2) average_orders
from customers join count_per_order
on customers.customer_id = count_per_order.customer_id
group by customers.customer_city order by average_orders desc`

3. Calculate the percentage of total revenue contributed by each product category.

SQL Query: `select upper(products.product_category) category,
round((sum(payments.payment_value)/(select sum(payment_value) from
payments))*100,2) sales_percentage
from products join order_items
on products.product_id = order_items.product_id
join payments
on payments.order_id = order_items.order_id
group by category order by sales_percentage desc`

4. Identify the correlation between product price and the number of times a product has been purchased.

SQL Query: `select products.product_category,
count(order_items.product_id),
round(avg(order_items.price),2)
from products join order_items
on products.product_id = order_items.product_id
group by products.product_category`

5. Calculate the total revenue generated by each seller, and rank them by revenue.

SQL Query: `select *, dense_rank() over(order by revenue desc) as rn from
(select order_items.seller_id, sum(payments.payment_value)
revenue from order_items join payments
on order_items.order_id = payments.order_id
group by order_items.seller_id) as a`

Advanced Queries

1. Calculate the moving average of order values for each customer over their order history.

SQL Query: `select customer_id, order_purchase_timestamp, payment,
avg(payment) over(partition by customer_id order by order_purchase_timestamp
rows between 2 preceding and current row) as mov_avg
from
(select orders.customer_id, orders.order_purchase_timestamp,
payments.payment_value as payment from payments
join orders
on payments.order_id = orders.order_id) as a`

2. Calculate the cumulative sales per month for each year.

SQL Query: `select years, months, payment, sum(payment)
over(order by years, months) cumulative_sales from
(select year(orders.order_purchase_timestamp) as years,
month(orders.order_purchase_timestamp) as months,
round(sum(payments.payment_value),2) as payment from orders join payments
on orders.order_id = payments.order_id
group by years, months order by years, months) as a`

3. Calculate the year-over-year growth rate of total sales.

SQL Query: `with a as(select year(orders.order_purchase_timestamp) as years,
round(sum(payments.payment_value),2) as payment from orders join payments
on orders.order_id = payments.order_id
group by years order by years)

select years, ((payment - lag(payment, 1) over(order by years))/
lag(payment, 1) over(order by years))*100 from a`

4. Calculate the retention rate of customers, defined as the percentage of customers who make another purchase within 6 months of their first purchase.

SQL Query: `with a as (select customers.customer_id,
min(orders.order_purchase_timestamp) first_order
from customers join orders
on customers.customer_id = orders.customer_id
group by customers.customer_id),`


```

b as (select a.customer_id, count(distinct orders.order_purchase_timestamp)
next_order
from a join orders
on orders.customer_id = a.customer_id
and orders.order_purchase_timestamp > first_order
and orders.order_purchase_timestamp < date_add(first_order, interval 6 month)
group by a.customer_id)





select 100*(count(distinct a.customer_id)/count(distinct b.customer_id))
from a left join b
on a.customer_id = b.customer_id

```

5. Identify the top 3 customers who spent the most money each year.

SQL Query: `select years, customer_id, payment, d_rank from`
`(select year(orders.order_purchase_timestamp) years, orders.customer_id,`
`sum(payments.payment_value) payment,`
`dense_rank() over(partition by year(orders.order_purchase_timestamp)`
`order by sum(payments.payment_value) desc) d_rank`
`from orders join payments`
`on payments.order_id = orders.order_id`
`group by year(orders.order_purchase_timestamp), orders.customer_id) as a`
`where d_rank <= 3`

These queries were used to explore various aspects of **customer behavior, sales trends, product performance, and revenue analysis** in the e-commerce dataset. The insights derived from these SQL queries helped in:

- Understanding **sales trends over time** 
- Identifying **customer purchase behavior** 
- Finding **top-selling products and sellers** 
- Analyzing **payment preferences and installment trends** 

These findings, along with **data visualizations in Python**, provide a comprehensive view of business performance and customer engagement.

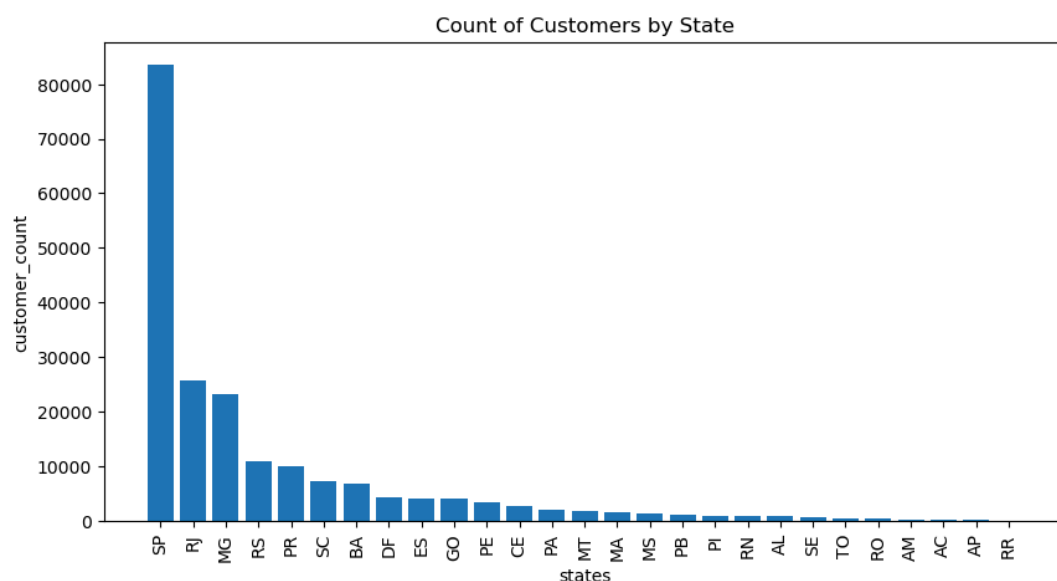
Results & Key Findings

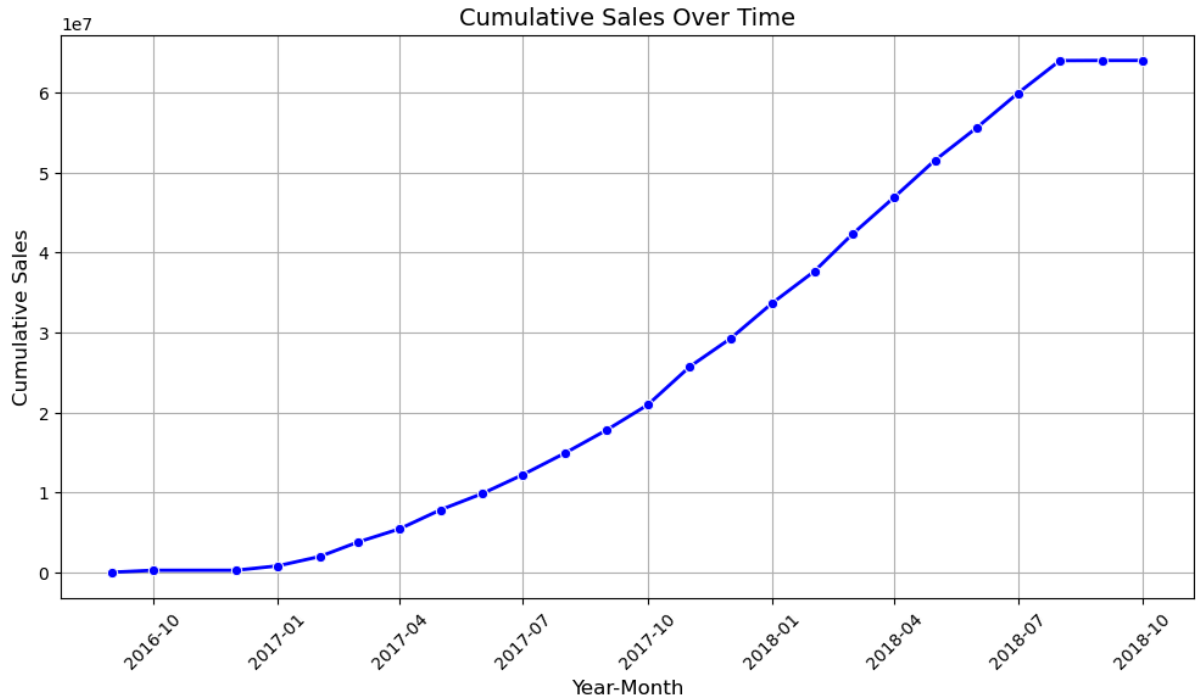
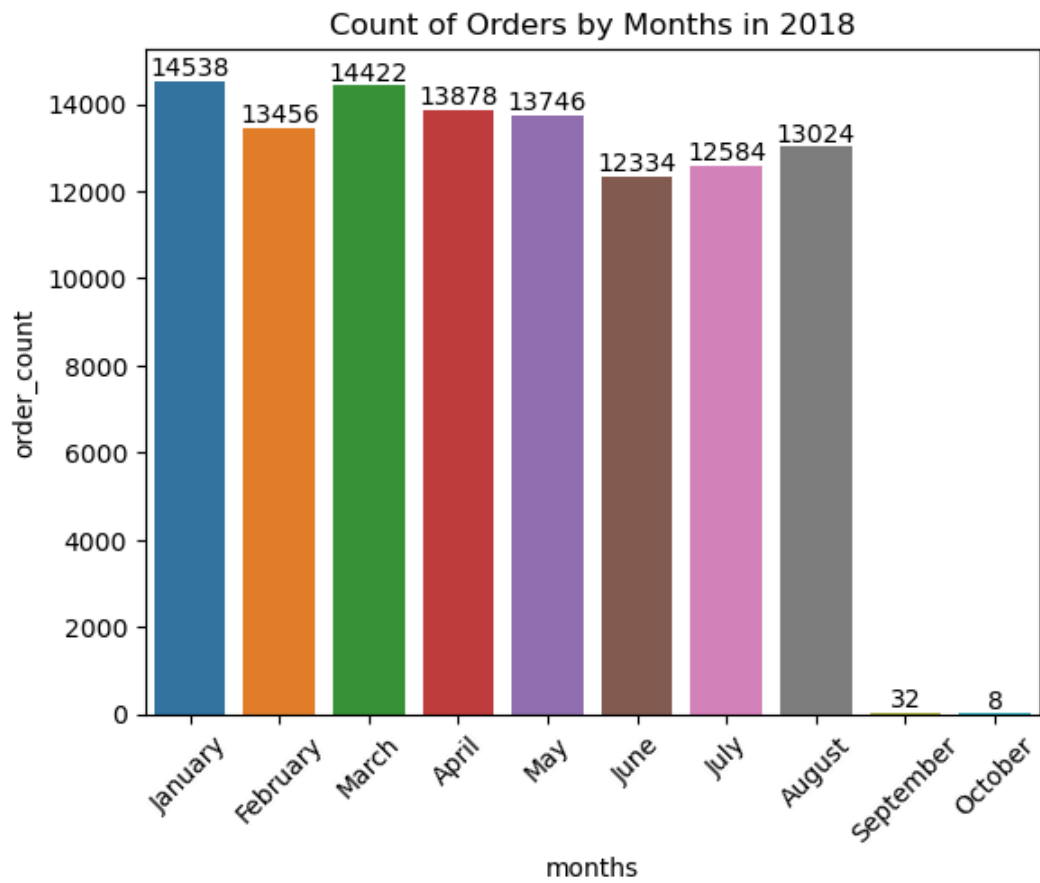
The analysis of e-commerce sales data revealed key insights through various visualizations, highlighting trends in sales, customer behavior, and payment preferences. The time-series analysis of monthly sales indicated seasonal spikes, especially during November and December, suggesting a need for targeted marketing strategies during peak shopping periods. Further analysis using **Seaborn's heatmaps** could help visualize correlations between different time periods and sales performance.

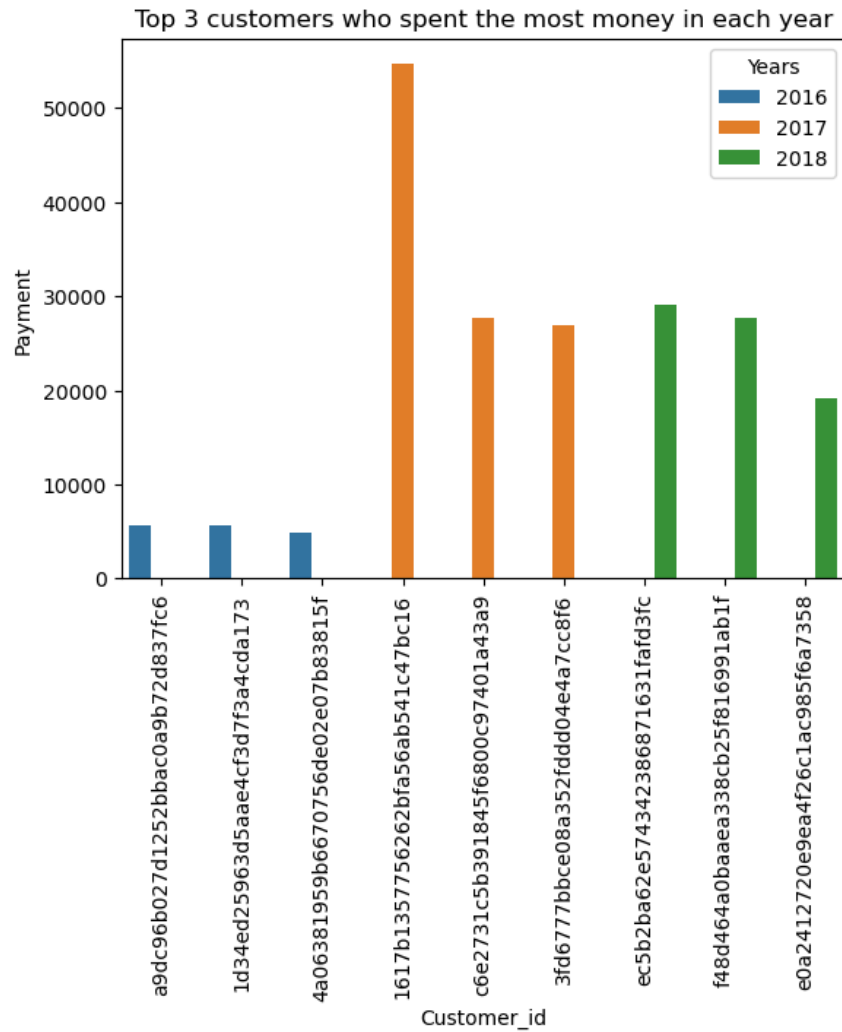
The bar charts displaying top-selling product categories and seller performance showed that a **few categories and sellers contribute to the majority of revenue**. This insight suggests that focusing on high-performing categories while improving the visibility of underperforming ones could enhance overall sales. Additional **box plots and violin plots** can be used to analyze variations in product pricing and customer spending habits.

Customer demographic distribution, represented in a geographic bar chart, showed that most buyers are from **major cities like São Paulo and Rio de Janeiro**. Expanding this visualization into a geographic heatmap with **Matplotlib's Basemap or Seaborn's choropleth maps** could provide a clearer regional sales distribution. Similarly, analyzing delivery times with histograms and KDE plots could help identify logistics bottlenecks.

Payment method analysis confirmed that **credit cards are the dominant payment mode**, followed by bank slips (boleto). A deeper exploration using **stacked bar charts or swarm plots** could reveal the relationship between payment preferences and order values. By incorporating additional advanced visualizations, we can gain a more detailed understanding of customer purchasing patterns and business opportunities.







Conclusion

The analysis of e-commerce sales data provided valuable insights into **sales trends, customer behavior, and payment preferences**. Seasonal peaks in sales highlight the importance of strategic promotions, while the dominance of a few product categories and sellers suggests opportunities for product diversification. Customer distribution data emphasizes the concentration of buyers in major cities, indicating potential for expansion in underserved regions. Additionally, the preference for **credit card payments** suggests a need to optimize payment options for a broader customer base.

Future Improvements

1. **Predictive Analytics** – Implementing **machine learning models** to forecast future sales trends and customer purchasing behavior based on historical data.
2. **Customer Segmentation** – Using **clustering techniques (K-Means, DBSCAN)** to classify customers based on spending patterns, purchase frequency, and preferences.
3. **Real-Time Dashboards** – Integrating tools like **Power BI or Tableau** to create interactive dashboards for dynamic data visualization and real-time decision-making.
4. **Logistics Optimization** – Analyzing **delivery times and shipping delays** using advanced statistical models to improve supply chain efficiency and customer satisfaction.
5. **Payment Behavior Analysis** – Exploring correlations between **payment methods and order values** to enhance financial strategies and introduce flexible payment options.
6. **Product Recommendation System** – Implementing **collaborative filtering or content-based filtering** to suggest relevant products to customers and increase sales conversions.
7. **Geospatial Sales Insights** – Using **heatmaps and geospatial analysis** to identify untapped markets and optimize marketing efforts in different regions.

GitHub Link

<https://github.com/VSSS19/E-Commerce-Sales-Analysis-using-SQL-Python>

Thankyou