Data Analysis on Target E-commerce Sales Data

A Comprehensive SQL and Python Analysis

BY UBEID TAMBOLI

Ecommerce Data Analysis

TABLE OF CONTENTS

BY UBEID TAMBOLI

Ecommerce Data Analysis

SQL + PYTHON PROJECT

PROJECT OVERVIEW:

- This project analyzes Target company's e-commerce data to derive actionable insights. Using Jupyter Notebook, I transformed and analyzed over 100,000 orders (from 2016–2018) with SQL and Python.
- Since SQL is limited in its visualization capabilities, I leveraged Python's data visualization libraries (like Matplotlib and Seaborn) to graphically represent insights, making patterns more interpretable and visually impactful.
- The dataset, encompassing customer demographics, order details, and payment history, was first structured by loading large CSV files into SQL databases through Python, enabling efficient querying.
- While SQL powered data extraction, Python was used to visualize key metrics like sales trends, customer distribution, and payment methods, offering a deeper understanding of purchasing behavior.

BY UBEID TAMBOLI

Ecommerce Data Analysis

SQL + PYTHON PROJECT

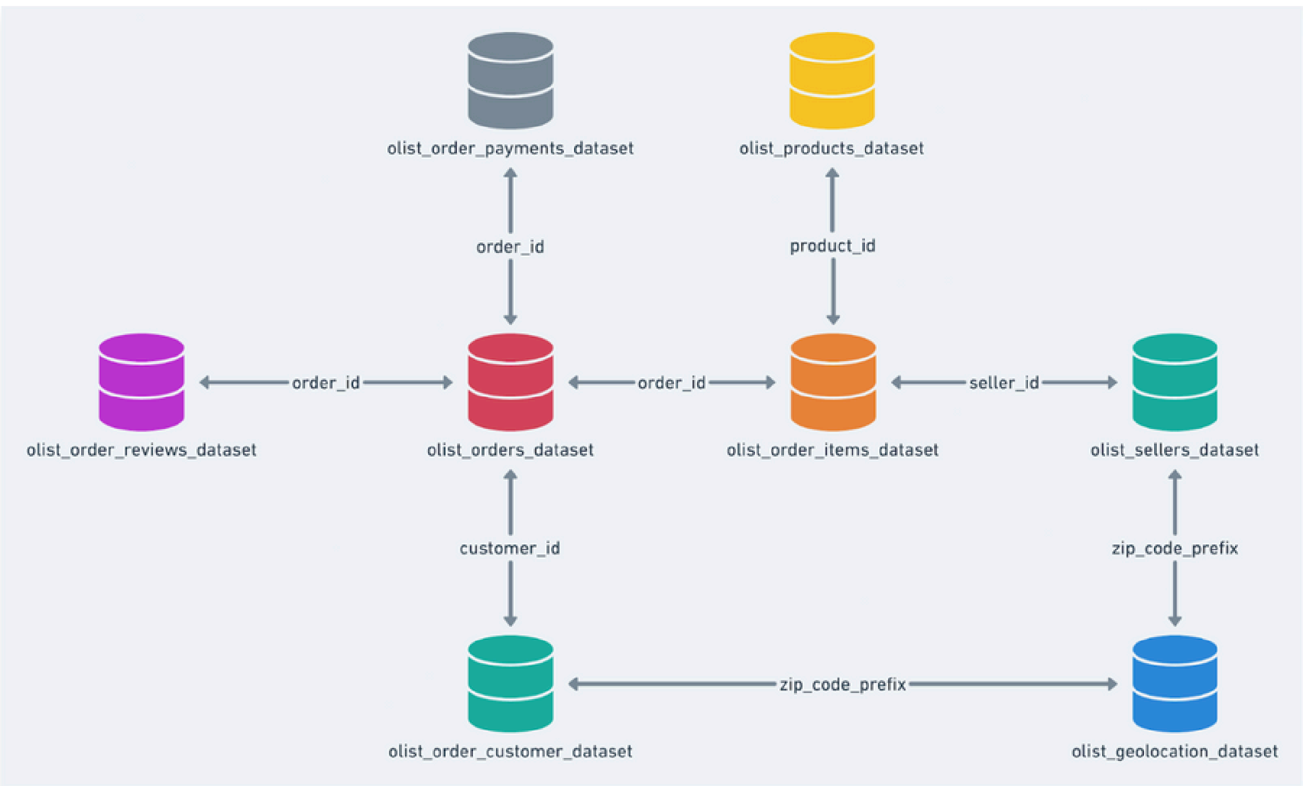
OBJECTIVE :

- The primary objective of this project is to conduct a comprehensive data analysis of an e-commerce dataset from Brazil, focusing on various key performance metrics.
- This includes evaluating customer demographics, order trends, and payment patterns to derive actionable insights that can enhance business strategies.
- Through the use of SQL and Python for data manipulation and visualization, the project aims to identify sales patterns, customer behavior, and revenue contributions by product categories. Ultimately, the analysis seeks to provide recommendations for optimizing sales strategies, improving customer retention, and maximizing revenue generation.

QUESTIONS MODE :

- **Easy Questions:** These queries primarily focus on basic data retrieval and aggregation techniques, employing fundamental SQL commands.
- **Intermediate Questions:** This level incorporates more complex queries that involve joining multiple tables and using aggregate functions for deeper insights.
- **Advanced Questions:** These queries utilize sophisticated SQL techniques, including Common Table Expressions (CTEs) and subqueries, to derive intricate insights from the dataset.

DATASET SCHEMA :



BY UBEID TAMBOLI

Ecommerce Data Analysis

SQL + PYTHON PROJECT

IMPORT CSV FILE TO SQL :

```
import pandas as pd
import mysql.connector
import os

# List of CSV files and their corresponding table names
csv_files = [
    ('customers.csv', 'customers'),
    ('orders.csv', 'orders'),
    ('sellers.csv', 'sellers'),
    ('products.csv', 'products'),
    ('geolocation.csv', 'geolocation'),
    ('payments.csv', 'payments'),
    ('order_items.csv', 'order_items')] # Added payments.csv for specific handling

# Connect to the MySQL database
try:
    conn = mysql.connector.connect(
        host='localhost',
        user='root',
        password='Ubeid@123',
        database='ecommerce'
    )
    print("Connected to database.")
except mysql.connector.Error as err:
    print(f"Error: {err}")
    exit(1)

cursor = conn.cursor()
```

BY UBEID TAMBOLI

Ecommerce Data Analysis

BASIC QUESTIONS

List all unique cities where customers are located.

In [149…

```
# SQL query to select distinct customer cities
query = """SELECT DISTINCT customer_city FROM customers"""

# Execute the SQL query
cur.execute(query)

# Fetch all unique customer cities from the executed query
data = cur.fetchall()

# Display the fetched data
df = pd.DataFrame(data, columns = ["cities"])
df.head()
```

Out[149…

	cities
0	franca
1	sao bernardo do campo
2	sao paulo
3	mogi das cruzes
4	campinas

INSIGHTS :

- This query provides a list of all unique customer cities, helping to understand the geographic spread of the customer base.
- Useful for demographic analysis and targeted marketing strategies.

Count the number of orders placed in 2017.

In [57]:

```
# SQL query to count the number of orders placed in 2017
query = """SELECT COUNT(*) AS NumberOfOrders FROM orders WHERE YEAR(order_purchase_timestamp) = 2017"""

# Execute the SQL query
cur.execute(query)

# Fetch the count of orders placed in 2017
data = cur.fetchall()

# Print the total number of orders placed in 2017
print("Total orders placed in 2017 are:", data[0][0])
```

```
Total orders placed in 2017 are: 45101
```

INSIGHTS :

- This query provides the total number of orders placed in 2017, which is crucial for year-over-year comparison and trend analysis.
- Useful for business performance reviews and strategic planning.

Find the total sales per category.

In [43]:
```
# SQL query to get total sales per product category
query = """SELECT products.product_category AS category, ROUND(SUM(payments.payment_value), 2) AS sales
FROM products
INNER JOIN order_items ON products.product_id = order_items.product_id
INNER JOIN payments ON payments.order_id = order_items.order_id
GROUP BY category"""

# Execute the SQL query
cur.execute(query)

# Fetch the results from the executed query
data = cur.fetchall()

# Create a DataFrame from the fetched data with specified column names
df = pd.DataFrame(data, columns=["Category", "Sales"])

# Display the DataFrame
df
```

Out[43]:

	Category	Sales
0	perfumery	506738.66
1	Furniture Decoration	1430176.39
2	telephony	486882.05
3	bed table bath	1712553.67
4	automotive	852294.33
...	...	...
69	cds music dvds	1199.43
70	La Cuisine	2913.53
71	Fashion Children's Clothing	785.67
72	PC Gamer	2174.43
73	insurance and services	324.51

74 rows × 2 columns

INSIGHTS :

- Of 74 categories, 'bed table bath' and 'Furniture Decoration' lead in sales with 1.7M and 1.4M, respectively, indicating high demand in home essentials. In contrast, 'cds music dvds' and 'PC Gamer' have minimal sales, reflecting lower demand in media categories.

Ecommerce Data Analysis

BASIC QUESTIONS

Calculate the percentage of orders that were paid in installments.

In [48]:
```
# SQL query to calculate the percentage of orders paid in installments
query = """SELECT SUM(payment_installments >= 1) / COUNT(*) * 100 AS Installments_percentage FROM payments """

cur.execute(query)

data = cur.fetchall()
# Print the percentage of orders paid in installments
print("The percentage of orders where paid in installments is:", data[0][0])
```

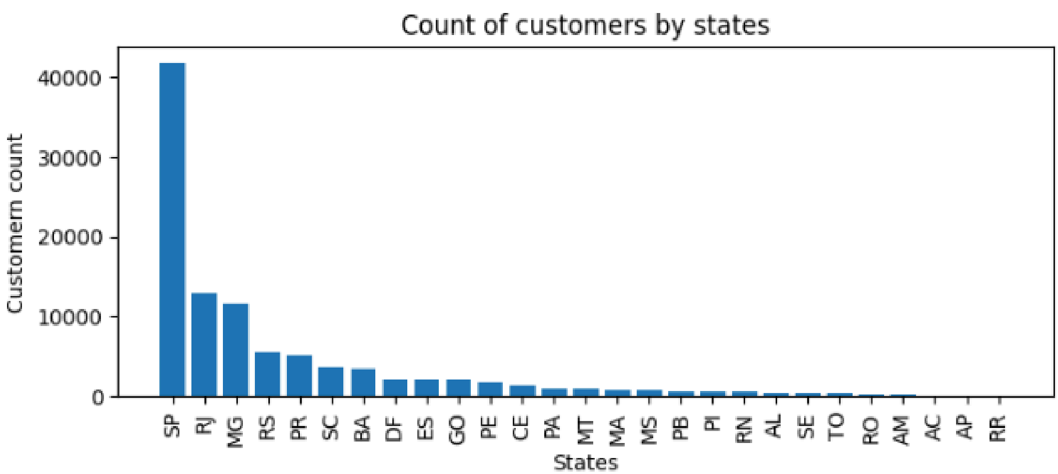
```
The percentage of orders where paid in installments is: 99.9981
```

INSIGHTS :

- Nearly all orders (99.9981%) were paid in installments, indicating a strong preference among customers for installment payments over one-time payments.

Count the number of customers from each state.

In [63]:
```
# SQL query to count customers per state and group by customer_state
query = """SELECT customer_state, COUNT(customer_id) AS customer_count FROM customers GROUP BY customer_state"""

# Execute the SQL query
cur.execute(query)

# Fetch the results from the executed query
data = cur.fetchall()

# Create DataFrame with specified column names
df = pd.DataFrame(data, columns=["State", "Customer_count"])
df = df.sort_values(by="Customer_count", ascending=False)

# Plotting the data
plt.figure(figsize=(8, 3))  # Set the size of the figure
plt.bar(df["State"], df["Customer_count"])  # Create a bar plot with states and customer counts
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability
plt.xlabel("States")
plt.ylabel("Customern count")
plt.title("Count of customers by states")
plt.show()  # Display the plot
```



INSIGHTS :

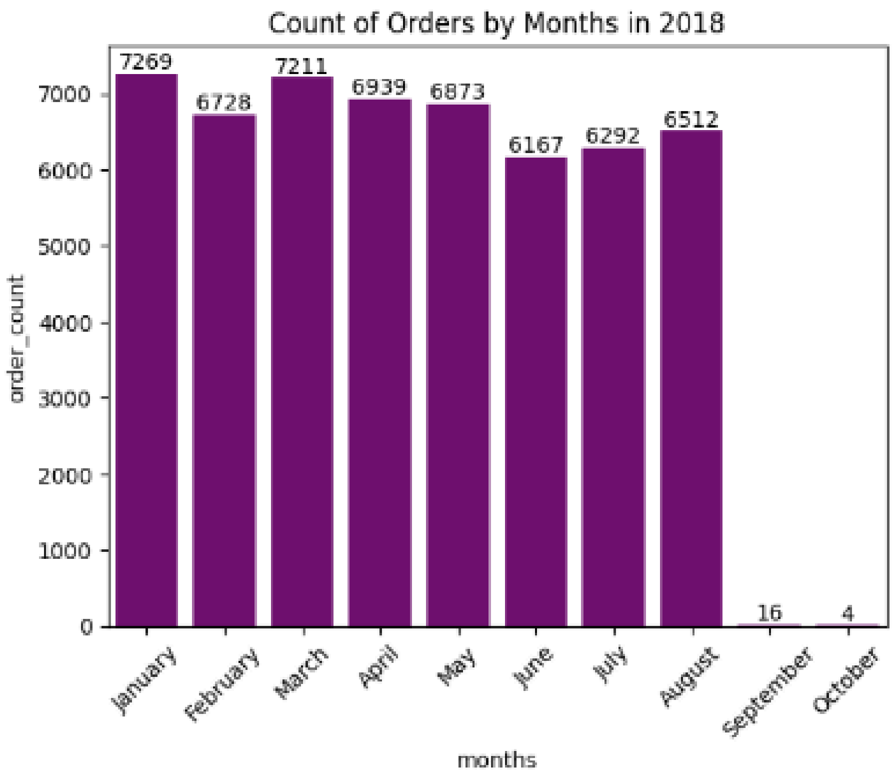
- Most customers are concentrated in the state of São Paulo (SP), followed by Rio de Janeiro (RJ) and Minas Gerais (MG), indicating these regions as key markets.

Ecommerce Data Analysis

INTERMEDIATE QUESTIONS

Calculate the number of orders per month in 2018.

In [81]:
```
query = """SELECT monthname(order_purchase_timestamp) AS months, COUNT(order_id) AS customer_count FROM orders WHERE YEAR(o

cur.execute(query)
data = cur.fetchall()

# Create a DataFrame from the fetched data
df = pd.DataFrame(data, columns = ["months", "order_count"])

# Define the order of months for the bar plot
o = ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October"]

# Create the bar plot using seaborn
ax = sns.barplot(x = df["months"], y = df["order_count"], data = df, order = o, color = "purple")

# Rotate the x-axis labels for better readability
plt.xticks(rotation = 45)

# Add labels to the bars
ax.bar_label(ax.containers[0])

# Add a title to the plot
plt.title("Count of Orders by Months in 2018")

# Display the plot
plt.show()
```



INSIGHTS :

- Based on the sales data for 2018, January recorded the highest number of orders (7,269), closely followed by March (7,211) and April (6,939).
- Sales are notably low in October and September, with only 4 and 16 orders respectively, indicating a potential seasonal decline in these months.

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

In [86]:

```
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) AS average_orders
FROM customers JOIN count_per_order ON customers.customer_id = count_per_order.customer_id
GROUP BY customers.customer_city; """

cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ["Customer_city", "average_products/orders"])
df.head(10) #shows top 10
```

Out[14]:

	City	Order
0	padre carvalho	7.00
1	celso ramos	6.50
2	candido godoi	6.00
3	datas	6.00
4	matias olimpio	5.00
...	...	...
4105	sao joao evangelista	1.00
4106	araponga	1.00
4107	arraias	1.00
4108	zacarias	1.00
4109	cedro de sao joao	1.00

4110 rows × 2 columns

INSIGHTS :

- The analysis reveals notable disparities in average products per order by city. Padre Carvalho (7.00) and Celso Ramos (6.50) show strong purchasing behavior, while cities like Sao Joao Evangelista and Araponga average only 1.00 product per order, indicating potential market limitations.
- These insights can inform targeted marketing and inventory strategies to enhance sales and customer engagement.

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

In [88]:

```
query = """SELECT UPPER(products.product_category) AS category,
ROUND(SUM(payments.payment_value)/(SELECT SUM(payment_value) FROM payments)*100,2) As sales_percentage
FROM products INNER JOIN order_items ON products.product_id = order_items.product_id
INNER JOIN payments ON payments.order_id = order_items.order_id
GROUP BY category ORDER BY sales_percentage DESC"""

cur.execute(query)

data = cur.fetchall()
df = pd.DataFrame(data, columns = ["Products Category", "Percentage distribution"])
df
```

Out[88]:

	Products Category	percentage
0	BED TABLE BATH	10.70
1	HEALTH BEAUTY	10.35
2	COMPUTER ACCESSORIES	9.90
3	FURNITURE DECORATION	8.93
4	WATCHES PRESENT	8.93
...	...	...
69	HOUSE COMFORT 2	0.01
70	CDS MUSIC DVDS	0.01
71	PC GAMER	0.01
72	FASHION CHILDREN'S CLOTHING	0.00
73	INSURANCE AND SERVICES	0.00

74 rows × 2 columns

INSIGHTS :

- Bed Table Bath (10.70%), Health Beauty (10.35%), and Computer Accessories (9.90%) are the top revenue contributors.
- In contrast, categories like House Comfort 2 and CDs Music DVDs contribute minimally, indicating potential growth opportunities through focused marketing and product strategies.

Ecommerce Data Analysis

INTERMEDIATE QUESTIONS

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

In [98]:
```
query = """SELECT products.product_category, COUNT(products.product_id) AS product_count,
ROUND(AVG(order_items.price),2) AS average_item_price FROM order_items JOIN products
ON order_items.product_id = products.product_id GROUP BY products.product_category;"""

cur.execute(query)

data = cur.fetchall()
df = pd.DataFrame(data, columns = ["Counr", "order_count", "price"])

arr1 = df["order_count"]
arr2 = df["price"]

a = np.corrcoef([arr1,arr2])
print("the correlation is:", a[0][-1])
```

```
the correlation is: -0.10631514167157562
```

INSIGHTS :

- The analysis reveals a weak negative correlation (-0.1063) between product price and purchase frequency, indicating that higher prices slightly correspond to fewer purchases, but the effect is minimal.
- Therefore, price alone does not significantly influence purchase frequency, suggesting that other factors (e.g., product appeal, quality, marketing) likely play a larger role in driving purchases. This insight highlights limited impact of price on demand, making it valuable to consider additional factors when developing sales and marketing strategies.

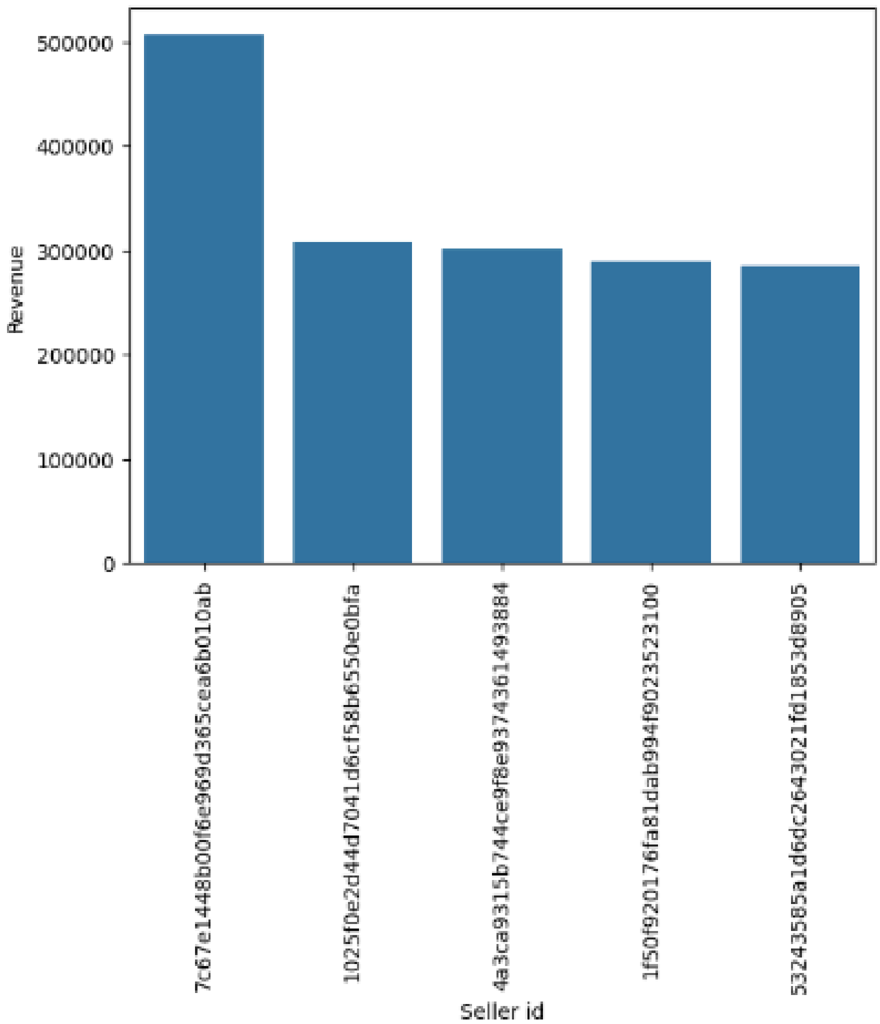
Ecommerce Data Analysis

INTERMEDIATE QUESTIONS

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

In [110…

```
query = """WITH SellerRevenue AS (
    SELECT order_items.seller_id, ROUND(SUM(payments.payment_value), 2) AS revenue
    FROM order_items JOIN payments ON order_items.order_id = payments.order_id
    GROUP BY order_items.seller_id)
SELECT seller_id, revenue, DENSE_RANK() OVER (ORDER BY revenue DESC) AS ranks
FROM SellerRevenue;"""

cur.execute(query)

data = cur.fetchall()
df = pd.DataFrame(data, columns = ["Seller id", "Revenue", "Ranks"])
df = df.head()

sns.barplot (x = "Seller id", y = "Revenue", data = df)
plt.xticks(rotation = 90)
plt.show()
```



INSIGHTS :

- The top 5 sellers contribute significantly to revenue, with the leading seller generating $507,166.91, far surpassing others. The second highest is $308,222.04, showing a substantial revenue gap.
- This highlights the dominance of a few sellers and suggests potential for strategic partnerships with high-performing sellers to maximize revenue.

Ecommerce Data Analysis

ADVANCED QUESTIONS

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

In [119…

```
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 move_avg
FROM(
SELECT orders.customer_id, orders.order_purchase_timestamp, payments.payment_value AS payment
FROM orders JOIN payments ON orders.order_id = payments.order_id
) AS a"""

cur.execute(query)

data = cur.fetchall()
df = pd.DataFrame(data, columns = ["customer_id", "purchase_timestamp", "price", "moving_ average"])
df = df.head()
df
```

Out[119…

	customer_id	purchase_timestamp	price	moving_ average
0	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.74	114.739998
1	000161a058600d5901f007fab4c27140	2017-07-16 09:40:32	67.41	67.410004
2	0001fd6190edaaf884bcaf3d49edf079	2017-02-28 11:06:43	195.42	195.419998
3	0002414f95344307404f0ace7a26f1d5	2017-08-16 13:09:20	179.35	179.350006
4	000379cdec625522490c315e70c7a9fb	2018-04-02 13:42:17	107.01	107.010002

INSIGHTS :

- The moving average of order values per customer over their order history gives insight into spending patterns.
- It smooths out fluctuations in order values, showing a trend of each customer's average spending over time.
- For example, as new purchases are made, each customer's moving average adjusts, reflecting recent spending behavior, which can be useful for targeting repeat customers with tailored marketing or personalized offers.

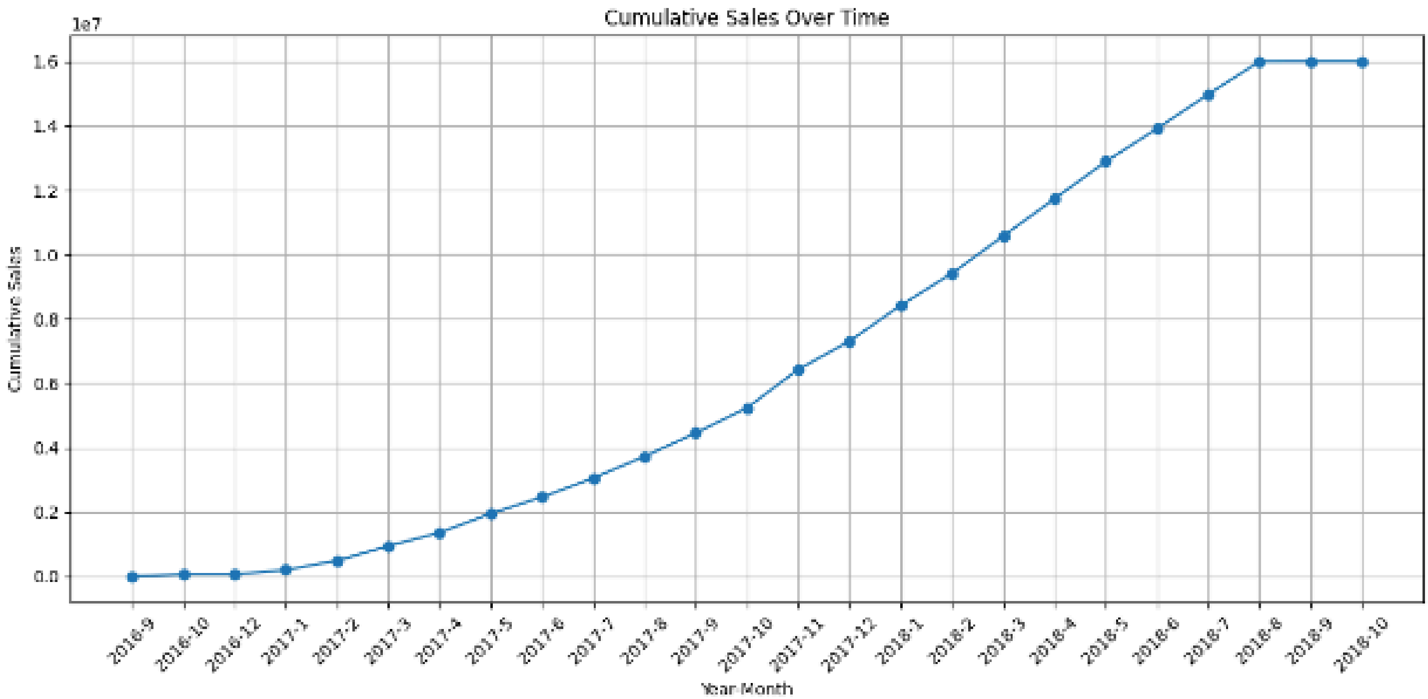
Ecommerce Data Analysis

ADVANCED QUESTIONS

Calculate the cumulative sales per month for each year.

In [124…

```
query = """SELECT years, months, payment, SUM(payment)
OVER(ORDER BY years, months) AS 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"""

cur.execute(query)

data = cur.fetchall()
df = pd.DataFrame(data, columns = ["years", "months", "payments", "cumulative sales"])

# Plotting the data
plt.figure(figsize=(12, 6))
#This creates labels like 2018-1, 2018-2
plt.plot(df["years"].astype(str) + '-' + df["months"].astype(str), df["cumulative sales"], marker='o')
plt.xlabel('Year-Month')
plt.ylabel('Cumulative Sales')
plt.title('Cumulative Sales Over Time')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



INSIGHTS :

- The cumulative sales data shows strong growth from early to late 2017, likely driven by increased demand or promotional efforts, before stabilizing towards the end of the year.
- This trend highlights peak periods for maximizing sales and signals potential saturation, suggesting a need for renewed strategies as growth plateaus.

Ecommerce Data Analysis

ADVANCED QUESTIONS

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

In [137…

```
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"""

cur.execute(query)

data = cur.fetchall()
df = pd.DataFrame(data, columns = ["years", "yoy % growth"])
df
```

Out[137…

	years	yoy % growth
0	2016	NaN
1	2017	12112.703761
2	2018	20.000924

INSIGHTS :

- The data shows an extraordinary 12112.7% growth from 2016 to 2017, likely due to a low sales base in 2016, followed by a minimal 20% increase in 2018.

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

In [138…

```
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 ;"""

cur.execute(query)
data = cur.fetchall()

data #SInce none of our customer is repeated thats why our value is none
```

Out[138… [(None,)]

INSIGHTS :

- The result shows that none of the customers made a repeat purchase within six months of their first purchase, resulting in a retention rate of zero.
- This suggests either a low customer loyalty rate, possibly due to product type or satisfaction issues, or a need for enhanced retention strategies, such as follow-up promotions or loyalty programs to encourage repeat purchases.

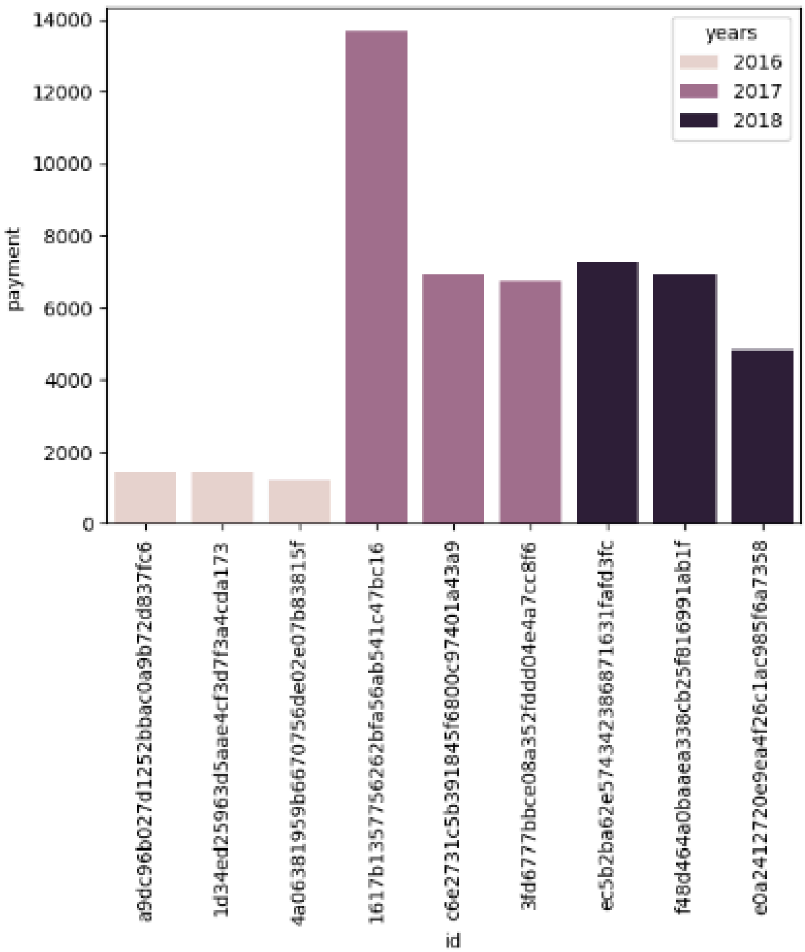
Ecommerce Data Analysis

ADVANCED QUESTIONS

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

In [147…

```
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 ;"""

cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ["years","id","payment","rank"])
sns.barplot(x = "id", y = "payment", data = df, hue = "years")
plt.xticks(rotation = 90)
plt.show()
```



INSIGHTS :

- The analysis reveals that the top 3 customers varied in spending each year, with a peak in 2017 where one customer spent nearly 14,000. This indicates a potential outlier in high-value transactions that year, while 2018 saw a decline in top-customer spending.
- The presence of consistent high spenders across years suggests opportunities for loyalty programs to retain and maximize value from these customers.