E-COMMERCE BUSINESS ANALYSIS USING SQL & PYTHON (POSTGRESQL)

📊 END-TO-END DATA ANALYSIS PROJECT

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import psycopg2
import numpy as np

conn = psycopg2.connect(
    host='localhost',
    port=5432,
    dbname='ecommerce',
    user='postgres',
    password='Postgres@123')
cur = conn.cursor()
```

✓ [132] Notebook is debugging…

🔹 PostgreSQL | 🔹 Python | 🔹 Pandas | 🔹 Matplotlib | 🔹 NumPy | 🔹 Seaborn

- **PostgreSQL – data storage & relational querying**
- **SQL – joins, aggregation, business metrics**
- **Python – data extraction & analysis**
- **Pandas & NumPy – data cleaning & correlation**
- **Matplotlib – data visualization**
- **Seaborn – advanced & statistical visualizations**

KEY BUSINESS QUESTIONS

1. **List all unique cities where customers are located.**
2. **Count the number of orders placed in 2017.**
3. **Find the total sales per category.**
4. **Count the number of customers from each state.**
5. **Calculate the number of orders per month in 2018.**
6. **Find the average number of products per order, grouped by customer city.**
7. **Calculate the percentage of total revenue contributed by each product category.**
8. **Identify the correlation between product price and the number of times a product has been purchased.**
9. **Calculate the total revenue generated by each seller, and rank them by revenue.**

PROBLEM STATEMENT

Analyze e-commerce transactional data to understand customer purchasing behavior,
pricing trends,
and payment patterns using SQL and Python.

WHAT I DID

- Converted raw CSV files into PostgreSQL tables
- Designed relational schema and applied joins
- Wrote optimized SQL queries for analysis
- Connected PostgreSQL with Python using psycopg2
- Performed correlation and trend analysis
- Visualized insights for decision-making

INSIGHTS

- Orders with lower average prices tend to have higher demand.
- Installment payments are widely used across categories .
- Price shows measurable correlation with order volume.

List all unique cities where customers are located.

```
query= """ select distinct customer_city from customers """

cur.execute(query)

data = cur.fetchall()

df=pd.DataFrame(data, columns=['customer_city'])
df
```

✓ [126] Notebook is debugging…

	customer_city
0	bom jardim de minas
1	alto rio doce
2	alvorada do gurgueia
3	batatais
4	capao da porteira
...	...
4114	carbonita
4115	concordia do para
4116	independencia
4117	governador valadares
4118	balsa nova

Count the number of orders placed in 2017

Notebook is debugging…

```
query = """ SELECT COUNT(order_id)
FROM orders
WHERE order_purchase_timestamp >= '2017-01-01'
  AND order_purchase_timestamp < '2018-01-01' """

cur.execute(query)

data = cur.fetchall()

"total orders placed in 2017 are", data[0][0]
```

✓ [27] Notebook is debugging…

```
('total orders placed in 2017 are', 45101)
```

Count the number of customers from each state

Notebook is debugging…

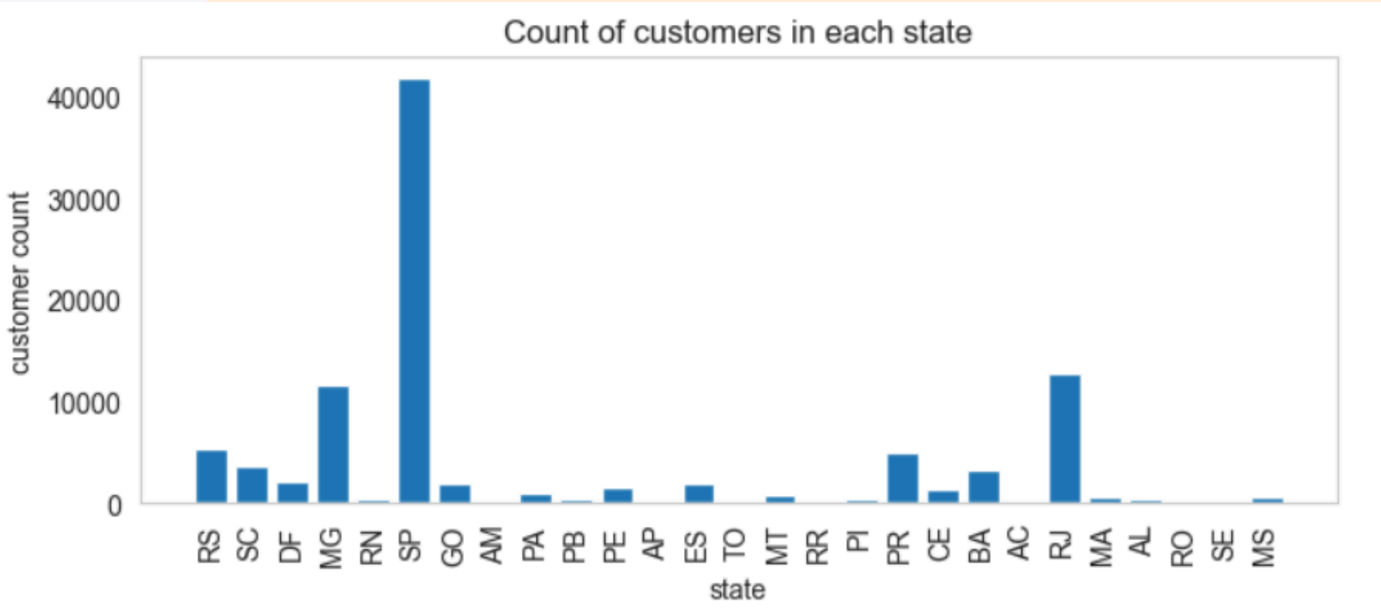
```python
query = """ select customer_state, count(customer_id) from customers
group by customer_state """

cur.execute(query)

data = cur.fetchall()

data
df= pd.DataFrame(data, columns=['state', 'customer_count'])
plt.figure(figsize=(8,3))
plt.bar(df['state'], df['customer_count'])
plt.xticks(rotation=90)
plt.xlabel('state')
plt.ylabel('customer count')
plt.title("Count of customers in each state")
plt.grid(False)
plt.show()
```

✓ [109] Notebook is debugging…



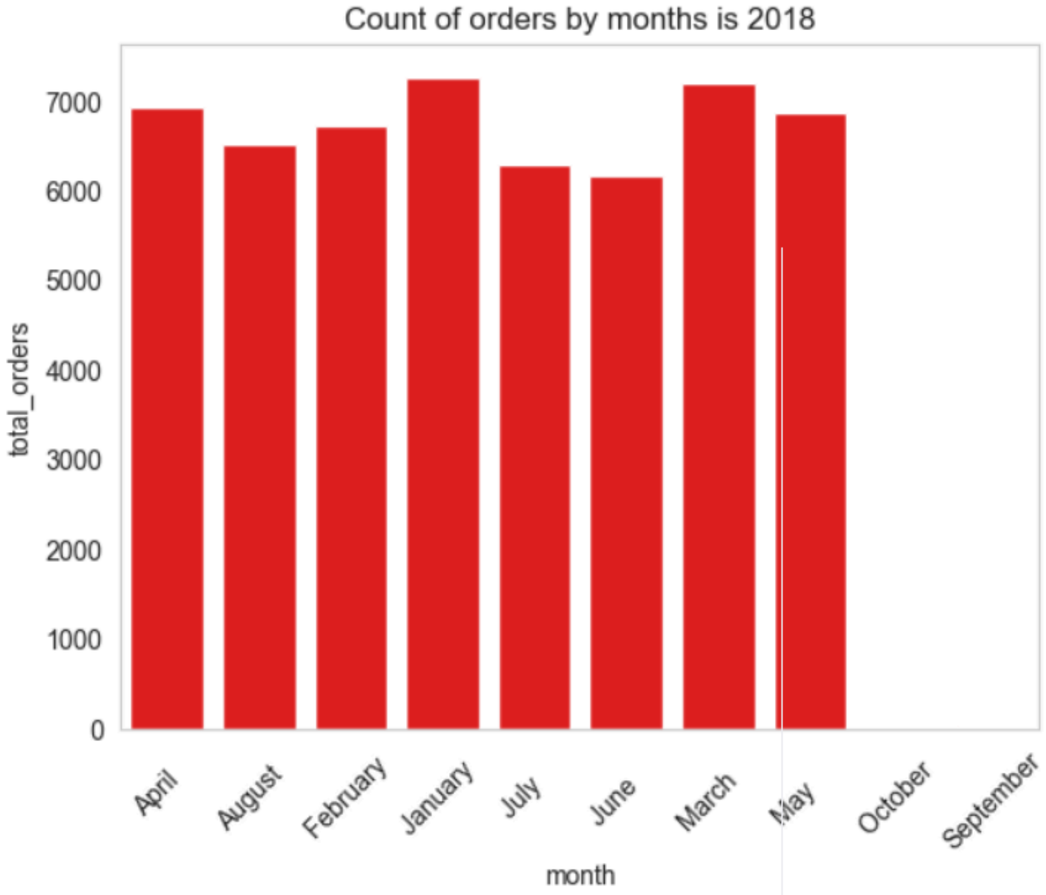
Calculate the number of orders per month in 2018

Notebook is debugging…

```
query = """ SELECT
    TO_CHAR(order_purchase_timestamp::timestamp, 'Month') AS month,
    COUNT(order_id) AS total_orders
FROM orders
WHERE order_purchase_timestamp::timestamp >= '2018-01-01'
  AND order_purchase_timestamp::timestamp < '2019-01-01'
GROUP BY month
ORDER BY month """

cur.execute(query)

data = cur.fetchall()
df=pd.DataFrame(data, columns=['month', 'total_orders'])
o=["January", "February", "March", "April","May","June","July",
   "August","September","October","November","December"]
ax= sns.barplot(x=df['month'], y=df["total_orders"],data=df, color="red")
plt.xticks(rotation=45)
plt.title("Count of orders by months is 2018")
plt.grid(False)
plt.show()
```



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

Notebook is debugging…

```
query = """ with count_per_order as
(select o.order_id, o.customer_id, count(oi.order_id) as oc from orders o
join order_items oi on o.order_id = oi.order_id
group by o.order_id, o.customer_id)

select c.customer_city, round(avg(cpo.oc),2) as avg_products_per_order from customers c
join count_per_order cpo on c.customer_id= cpo.customer_id
group by c.customer_city order by avg_products_per_order desc """

cur.execute(query)

data = cur.fetchall()
df=pd.DataFrame(data, columns=['customer_city', 'avg_products_per_order'])
df.head(5)
```

✓ [134] Notebook is debugging…

	customer_city	avg_products_per_order
0	padre carvalho	7.00
1	celso ramos	6.50
2	candido godoi	6.00
3	datas	6.00

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

Notebook is debugging…

```
query = """
select p.product_category, (sum(payment_value)/(select sum(payment_value)
from payments))*100 as sales_perentage from products p
join order_items oi on p.product_id= oi.product_id
join payments ps on oi.order_id= ps.order_id
group by p.product_category order by sales_perentage desc """

cur.execute(query)

data = cur.fetchall()
df=pd.DataFrame(data, columns=['Category', 'Percentage Distribution'])
df.head(5)
```

✓ [122] Notebook is debugging…

	Category	Percentage Distribution
0	bed table bath	10.697529
1	HEALTH BEAUTY	10.352841
2	computer accessories	9.902824
3	Furniture Decoration	8.933649
4	Watches present	8.927654

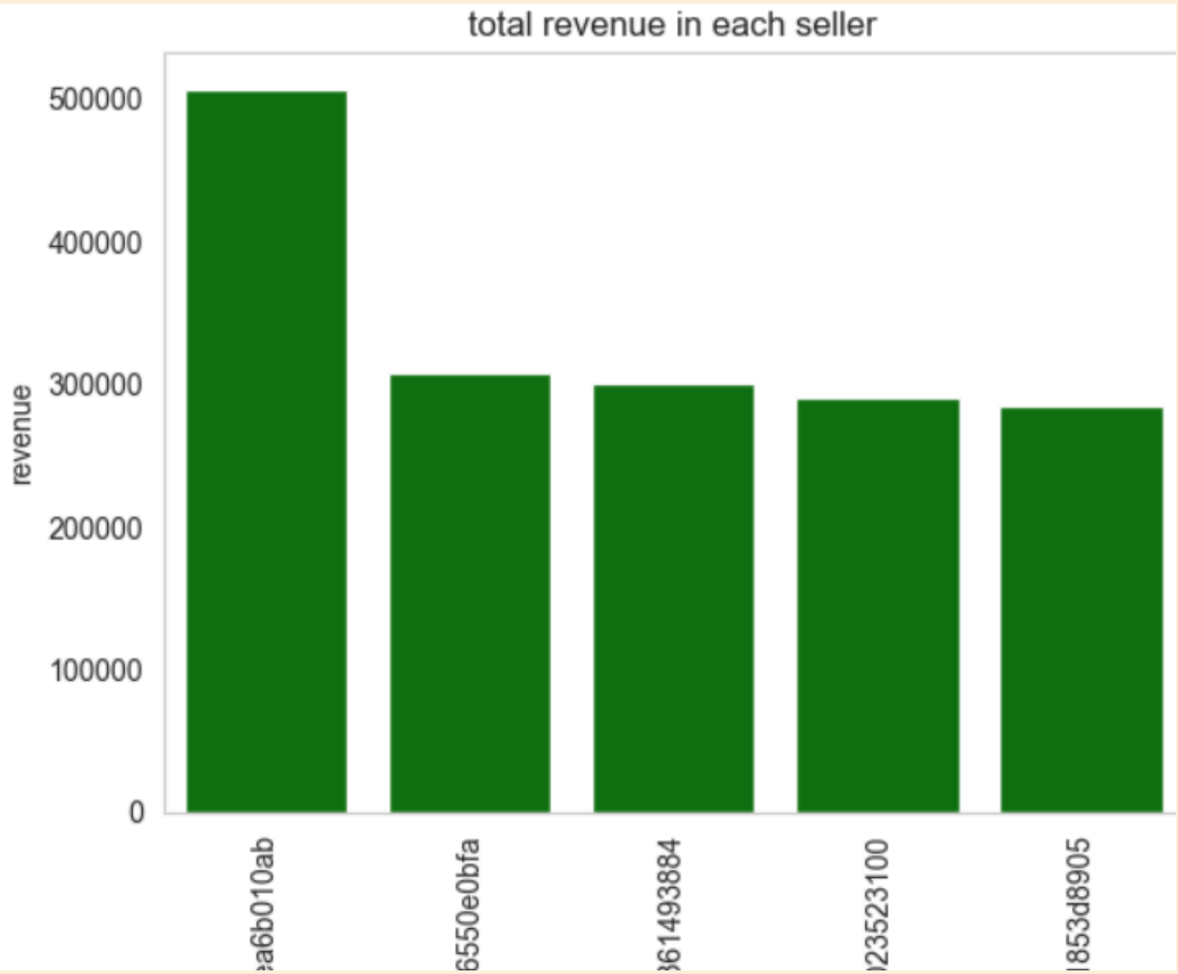
Calculate the total revenue generated by each seller, and rank them by revenue.

Notebook is debugging…

```
query = """
select *, dense_rank() over(order by revenue desc) as rn from
(select oi.seller_id, sum(p.payment_value) revenue from order_items oi
join payments p on oi.order_id=p.order_id
group by oi.seller_id)as a
 """

cur.execute(query)

data = cur.fetchall()
df=pd.DataFrame(data, columns=['seller_id', 'revenue', 'rank'])
df= df.head()
sns.barplot(x='seller_id', y='revenue', data=df, color="green")
plt.xticks(rotation=90)
plt.xlabel('seller_id')
plt.ylabel('revenue')
plt.grid(False)
plt.title("total revenue in each seller")
plt.show()
```



Find the total sales per category.

Notebook is debugging…

```
query = """ select p.product_category,sum(payment_value) from products p
join order_items oi on p.product_id= oi.product_id
join payments ps on oi.order_id= ps.order_id
group by p.product_category; """

cur.execute(query)

data = cur.fetchall()

data[0][0]
df= pd.DataFrame(data, columns=['category', 'sales'])
df.head(5)
```

✓ [124] Notebook is debugging…

	category	sales
0	Agro Industria e Comercio	118730.61
1	Art	30992.93
2	Arts and Crafts	2326.17
3	audio	60324.62
4	automotive	852294.33

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

Notebook is debugging…

```
query = """ select pr.product_category, count(oi.product_id),
ROUND(avg(oi.price)::numeric,2) from products pr
join order_items oi on pr.product_id= oi.product_id
group by pr.product_category """

cur.execute(query)

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

df=df.dropna()
arr1=df["order_count"].astype(float)
arr2=df["price"].astype(float)
correlation= np.corrcoef(arr1,arr2)
print("the correlation between price and number of orders is ",
      correlation[0],[-1])
```

✓ [160] Notebook is debugging…

```
the correlation between price and number of orders is  [ 1.         -0.10625495] [-1]
```

THANK YOU