ECOMMERCE DATA ANLAYSIS

TOOLS USED-MYSQL, PYTHON LIBRARIES PANDAS, NUMPY, MATPLOTLIB, SEABORN, ETC.).

BY -RAMAN GULHANE

WHAT IS THE PROJECT ABOUT?

Objective:To understand customer behavior, identify key product trends, and optimize sales strategies and Briefly describe the e-commerce dataset and its source.

Project workflow:

Data Acquisition, Data Cleaning/Preprocessing, SQL Analysis, Python Analysis, Visualization, Insights

Connecting Python to E-commerce Data

I began by importing the necessary Python libraries: Pandas for data handling, Matplotlib and Seaborn for visualizations, and the MySQL Connector to interface with our database. Then, I established a connection to the 'ecommerce' database using my local host and credentials. This connection, and the creation of a cursor, 'cur', enabled me to execute SQL queries directly from Python to pull data for analysis and visualization.

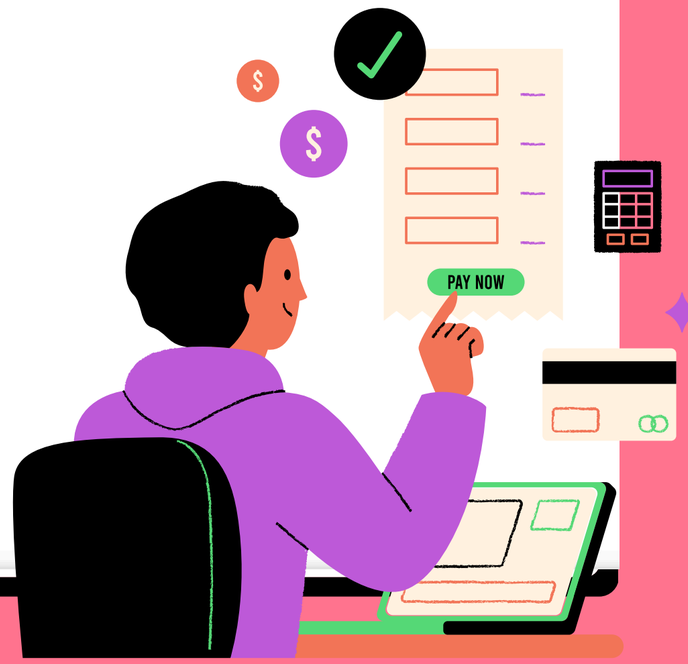
```
[1]: import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import mysql.connector
     db=mysql.connector.connect(
         host='127.0.0.1',
         user='root',
         password='raman@1234',
         database='ecommerce')
     cur=db.cursor()
```

CUSTOMER CITIES: A FIRST LOOK

To understand where our customers are located, I used this simple SQL query to find all the unique cities in our customer data. The result, shown here, is a list of these cities. This gives us a basic idea of our customer's geographical spread.

```
[28]: query=""" select distinct(customer_city) from customers """
      cur.execute(query)
      data=cur.fetchall()
      data
```

```
[28]: [('franca',),
       ('sao bernardo do campo',),
       ('sao paulo',),
       ('mogi das cruzes',),
       ('campinas',),
       ('jaragua do sul',),
       ('timoteo',),
       ('curitiba',),
       ('belo horizonte',),
       ('montes claros',),
       ('rio de janeiro',),
       ('lencois paulista',),
       ('caxias do sul',),
       ('piracicaba',),
       ('guarulhos',),
       ('pacaja',),
       ('florianopolis',),
       ('anapolis de goiania',)
```



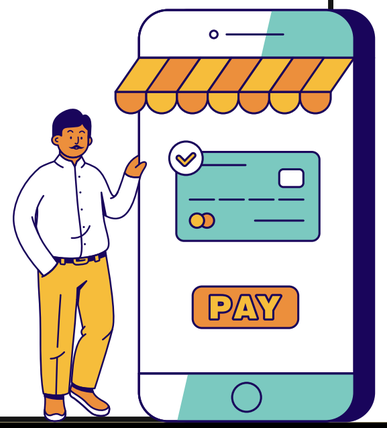
IDENTIFYING HIGH & LOW REVENUE CATEGORIES

Find the total sales per category

```
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
"""
cur.execute(query)
data=cur.fetchall()
df=pd.DataFrame(data,columns=["category","Sales"])
df
```

	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



WE'RE ANALYZING PRODUCT CATEGORY SALES HERE, REVEALING WHICH CATEGORIES DRIVE THE MOST REVENUE AND WHICH LAG BEHIND

- SQL FOR DATA: WE USED A SQL QUERY TO COMBINE DATA AND CALCULATE TOTAL SALES PER CATEGORY.
- KEY PERFORMERS: 'BED TABLE BATH' AND 'FURNITURE DECORATION' ARE OUR TOP REVENUE GENERATORS.
- LOW PERFORMERS: CONVERSELY, 'INSURANCE AND SERVICES' AND 'FASHION CHILDREN'S CLOTHING' SHOW SIGNIFICANTLY LOWER SALES.
- INSIGHTS: THIS VARIANCE POINTS TO POTENTIAL AREAS FOR STRATEGIC ADJUSTMENTS, LIKE TARGETED MARKETING OR PRODUCT OPTIMIZATION.

KEY PAYMENT & ORDER INSIGHTS

```
query = """
SELECT (sum(case when payment_installments >= 1 then 1 else 0 end) * 100.0) / count(*)
as percentage_of_installments
from payments
"""

cur.execute(query)  # Execute the SQL query
data = cur.fetchall()  # Fetch the result
print(f"installments percentage:{data[0][0]}")  # Extract the percentage value from the tuple
```

```
installments percentage:99.99807
```

PAYMENT INSTALLMENTS: FIRST, WE FOUND THAT NEARLY ALL PAYMENTS – 99.99% – INVOLVED INSTALLMENTS. THIS SHOWS HOW POPULAR INSTALLMENT PAYMENTS ARE WITH OUR CUSTOMERS.

2017 ORDERS: SECOND, WE LOOKED AT ORDERS PLACED IN 2017. WE FOUND THAT THERE WERE 45,101 ORDERS IN THAT YEAR.

```
query=""" select count(order_id) from orders where year (order_purchase_timestamp)=201
cur.execute(query)
data=cur.fetchall()
"total orders placed in 2017 are " ,data[0][0]
```

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

CUSTOMER DISTRIBUTION BY STATE (TOP 10)

To understand where our customers are located in Brazil, we used a simple SQL query to count customers per state. This slide shows the top 10 states by customer count. The result is clear: São Paulo (SP) has far more customers than any other state, with over 41,000. States like Rio de Janeiro and Minas Gerais follow, but with much lower counts.

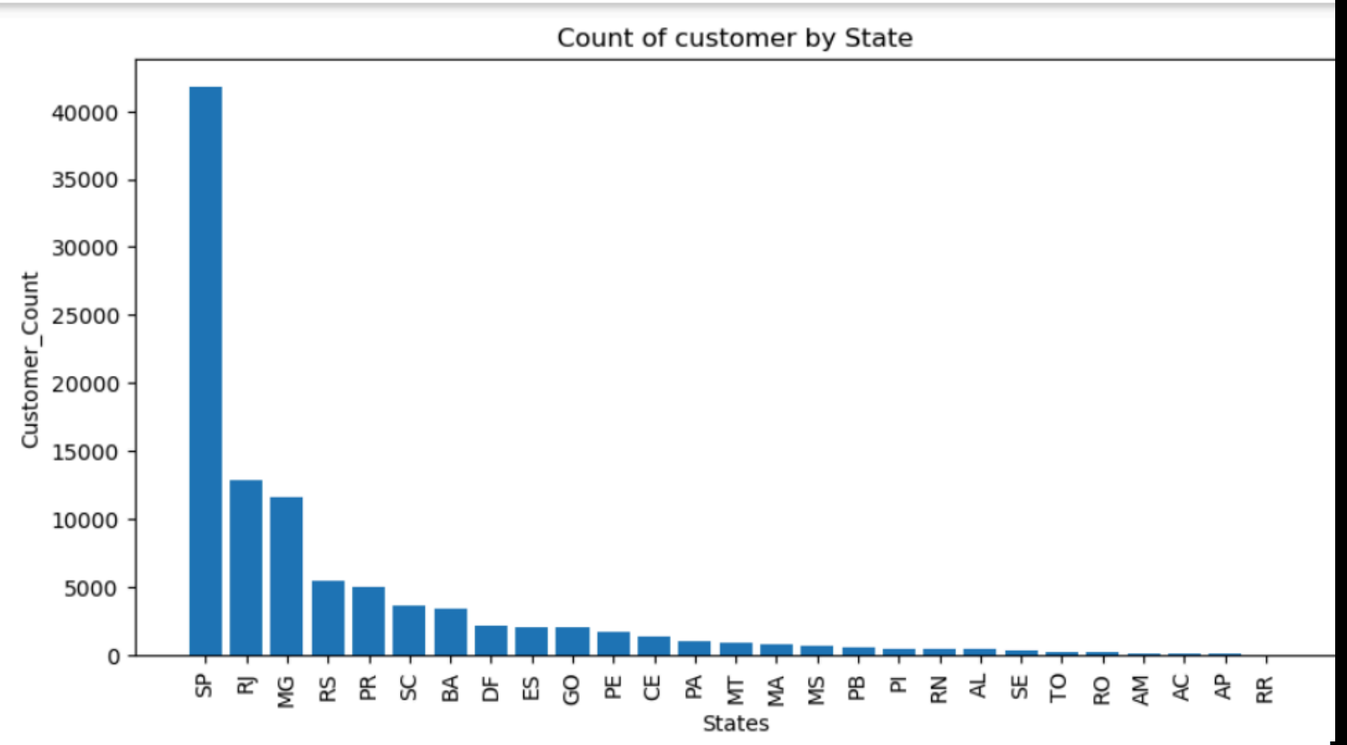
[29]:
```
query=""" select customer_state,count(customer_id)
from customers group by customer_state"""
cur.execute(query)
data=cur.fetchall()
df=pd.DataFrame(data,columns=["State","Customer_Count"])
df.head(10)
```

[29]:

	State	Customer_Count
0	SP	41746
1	SC	3637
2	MG	11635
3	PR	5045
4	RJ	12852
5	RS	5466
6	PA	975
7	GO	2020
8	ES	2033

Graphical Representation of Top 10 States in Customer Distribution

```
query=""" select customer_state,count(customer_id)
from customers group by customer_state"""
cur.execute(query)
data=cur.fetchall()
df=pd.DataFrame(data,columns=["State","Customer_Count"])
df=df.sort_values(by="Customer_Count",ascending=False)
plt.figure(figsize=(10,5))
plt.bar(df["State"],df["Customer_Count"])
plt.xticks(rotation=90)
plt.xlabel("States")
plt.ylabel("Customer_Count")
plt.title("Count of customer by State")
plt.show()
```

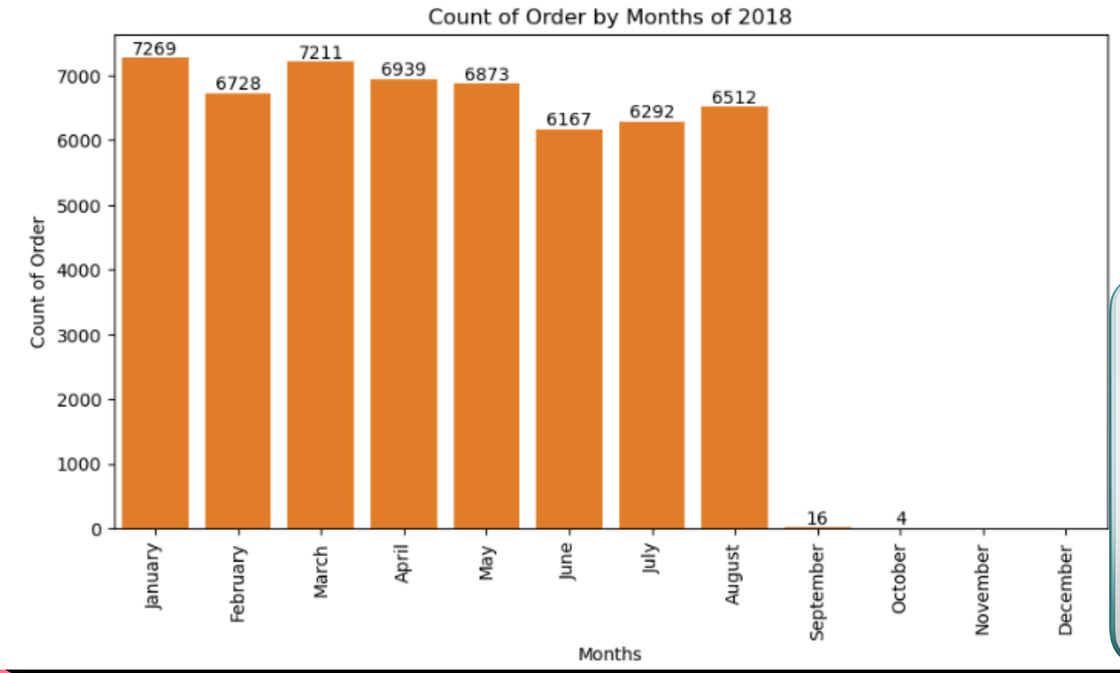


- The chart visually represents the customer distribution across Brazilian states.
- The height of each bar corresponds to the number of customers from that state.
- The state of São Paulo (SP) has a significantly taller bar, indicating a much larger customer base compared to other states.
- The chart effectively highlights the concentration of customers in São Paulo and the relative sparsity in other states.
- This visual aid makes it very easy to understand the geographical distribution of the customer base.

2018 Order Trends: Monthly Breakdown

```
query = """
select
    monthname(order_purchase_timestamp) as month_name,
    count(order_id) as order_count
from orders
where year(order_purchase_timestamp) = 2018
group by month (order_purchase_timestamp), monthname(order_purchase_timestamp)
order by month(order_purchase_timestamp);
"""

cur.execute(query)  # Execute query
data = cur.fetchall()  # Fetch results
df=pd.DataFrame(data,columns=["Months","Orders"])
plt.figure(figsize=(10,5))
plt.bar(df["Months"],df["Orders"])
plt.xticks(rotation=90)
month_order = ["January", "February", "March", "April", "May", "June",
               "July", "August", "September", "October", "November", "December"]
ax = sns.barplot(x="Months", y="Orders", data=df, order=month_order)
ax.bar_label(ax.containers[0])
plt.xlabel("Months")
plt.ylabel("Count of Order")
plt.title("Count of Order by Months of 2018")
plt.show()
```



- "Let's explore the monthly order trends we observed in 2018. This chart visualizes the order volume for each month.
- "SQL Query: We asked the database to count the orders for each month of 2018.
- Data Table: We then put this data into a Python table (DataFrame) for easier analysis.
- Bar Chart: Finally, we visualized this data using a bar chart, showing the order count for each month.

Key Observations:

- As you can see, the first half of the year, from January to July, saw a relatively consistent and high number of orders.
- However, starting in August, we see a dramatic drop in orders, with September through December having very low order counts.

Percent of Total Sales by Product Category

```
query="""SELECT
    UPPER(products.product_category) AS category,
    ROUND(SUM(payments.payment_value / (SELECT SUM(payment_value) FROM payments)) * 100, 2) AS sal
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 DESC; -- Changed from sales_percentage to sales

"""
cur.execute(query)

data = cur.fetchall()

df = pd.DataFrame(data, columns=['Category', 'Percentage Distribution'])
df.head(10)
```

	Category	Percentage Distribution
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
5	SPORT LEISURE	8.70
6	HOUSEWARES	6.84
7	AUTOMOTIVE	5.32
8	GARDEN TOOLS	5.24
9	COOL STUFF	4.87

This slide shows the percentage of our total revenue contributed by each product category. We used SQL to calculate the sales for each category as a percentage of our overall sales.

- SQL Logic: We joined the 'products', 'order_items', and 'payments' tables to link categories with their payment values. Then, we calculated each category's sales as a percentage of the total sales.
- Top Performers: The table shows the top 10 categories. 'BED TABLE BATH' contributes the most, at 10.70% of our total revenue, followed closely by 'HEALTH BEAUTY' and 'COMPUTER ACCESSORIES'.

This data helps us understand which product categories are driving the most revenue, and where we should focus our efforts.

Customer Spending Trends: Moving Average Analysis

```
query="""
select subquery.customer_id, subquery.order_purchase_timestamp, subquery.payment,
       AVG(subquery.payment) OVER (PARTITION BY subquery.customer_id ORDER BY subquery.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 subquery;
"""
cur.execute(query)

data = cur.fetchall()

df = pd.DataFrame(data,columns=['customer_id','timestamp','price','moving_average'])
df
```

	customer_id	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
...	...	...	...	...
103881	fffecc9f79fd8c764f843e9951b11341	2018-03-29 16:59:26	71.23	27.120001
103882	fffeda5b6d849fbd39689bb92087f431	2018-05-22 13:36:02	63.13	63.130001
103883	ffff42319e9b2d713724ae527742af25	2018-06-13 16:57:05	214.13	214.130005
103884	ffffa3172527f765de70084a7e53aae8	2017-09-02 11:53:32	45.50	45.500000
103885	ffffe8b65bbe3087b653a978c870db99	2017-09-29 14:07:03	18.37	18.370001

103886 rows × 4 columns

This slide examines customer spending habits over time by calculating a moving average of their purchase amounts. We used a SQL query to join order and payment data, then calculated the average of each customer's current order value with their two preceding orders. This 'moving average' helps smooth out fluctuations and reveal underlying spending trends for individual customers."

"The resulting table displays the customer ID, order timestamp, order price, and the calculated moving average. By observing the moving average, we can identify patterns in customer spending, such as increasing or decreasing purchase values over time. This information is valuable for targeted marketing, customer segmentation, and predicting future spending behaviors.


Thank You!!
Jumpers