# ECOMMERCE DATA ANALYSIS PROJECT

07/08/2024

Created By
*Vijay Kumar Shah*

# PROJECT OVERVIEW

This is the material point that will be deliverThis project aims to provide insightful analysis of e-commerce data by leveraging SQL and Python. It involves extracting data from various sources, cleaning and transforming it, and loading it into a MySQL database. We utilized libraries such as pandas, seaborn, and matplotlib for data manipulation and visualization.ed in the presentation.

# TECHNOLOGY STACK

Tools :

- Mysql Server
- Jupyter Notebook

Programming Languages :

- Python
- MySql

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import mysql.connector
import os

import warnings
warnings.filterwarnings('ignore')
```

Libraries:

- Numpy
- Pandas
- Matplotlib
- Seaborn
- mysql.connector

# Import Libraries:

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import mysql.connector
import os

import warnings
warnings.filterwarnings('ignore')
```

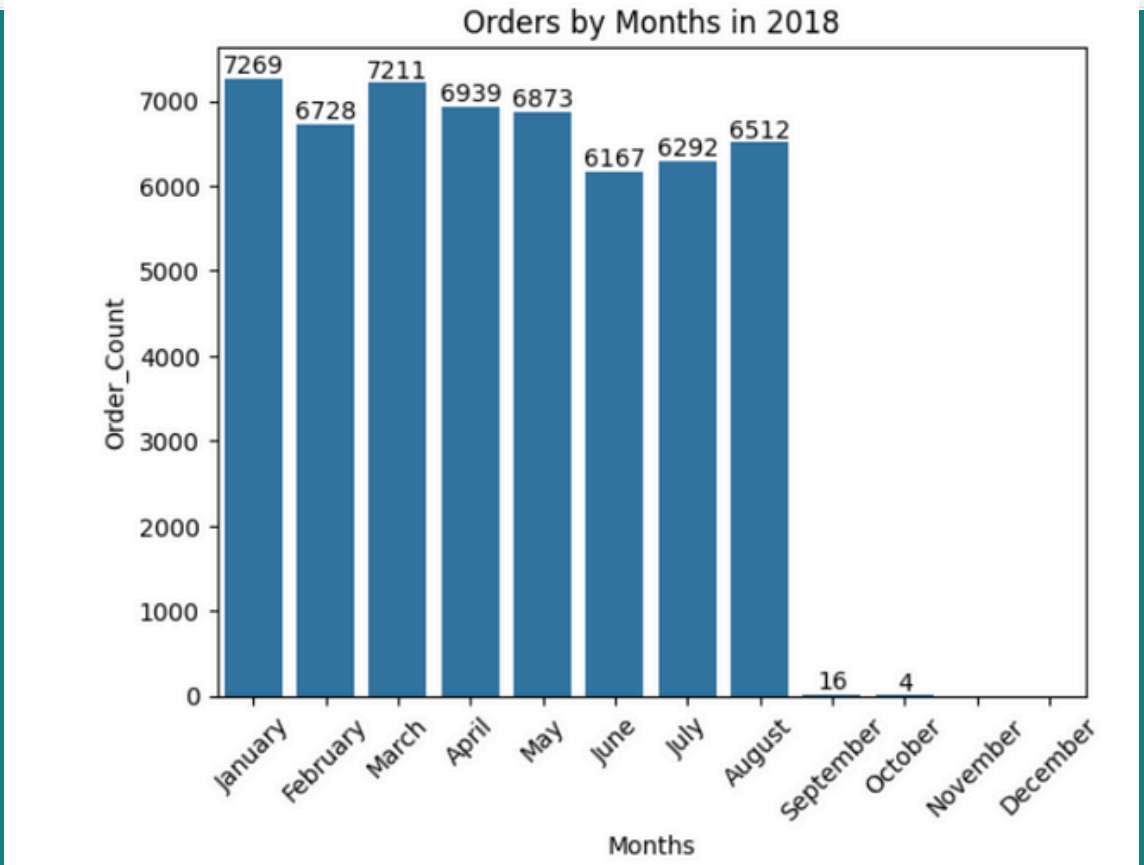
# Connect Mysql with Python

```
db = mysql.connector.connect(host = "localhost",
                             username = "root",
                             password = "Mysql@2023",
                             database = "ecommerce")

cur = db.cursor()
```

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

```
query = """select monthname(order_purchase_timestamp) months ,count(order_id) order_count
from orders where year(order_purchase_timestamp)=2018
group by months
"""
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data,columns = ["Months","Order_Count"])
o = ["January","February","March","April","May","June","July","August","September","October","November","December"]
ax = sns.barplot(x='Months',y = 'Order_Count',data=df,order=o)
plt.xticks(rotation=45)
ax.bar_label(ax.containers[0])
plt.title("Orders by Months in 2018")
plt.show()
```



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

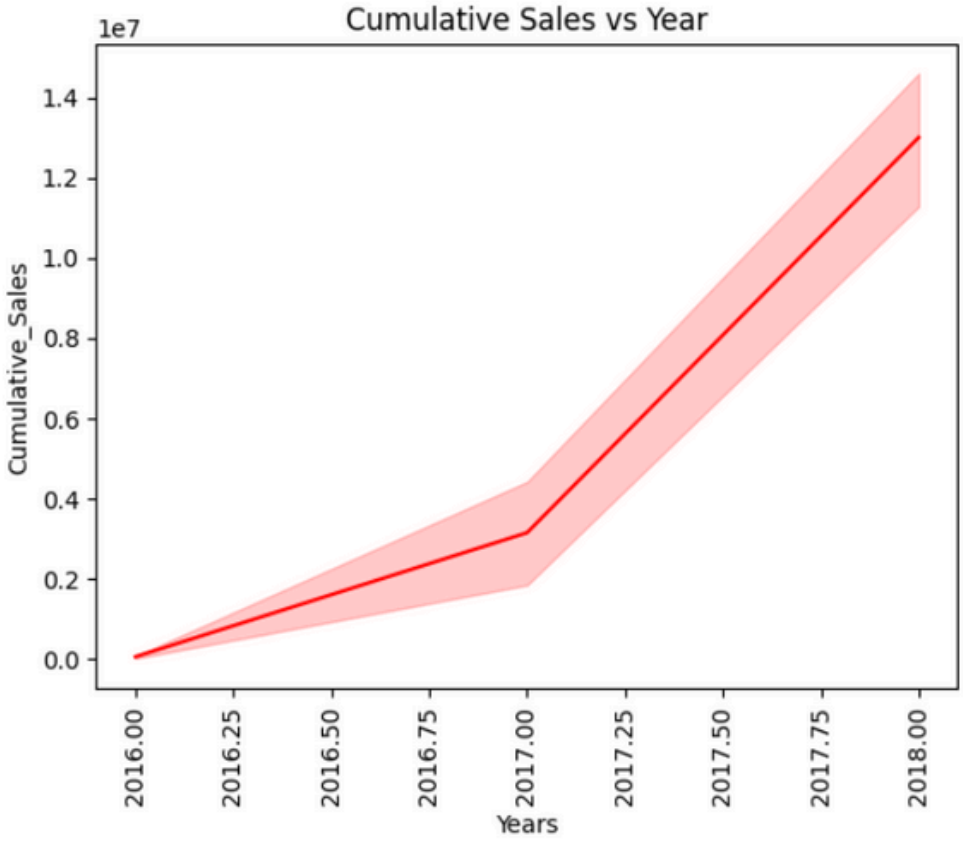
```
query = """ select years,months,payment,sum(payment)
over(order by years ,months) cumulative_sales
from
(select year(orders.order_purchase_timestamp) as years,
month(orders.order_purchase_timestamp) as months,
round(sum(payments.payment_value),2) as payment from orders join payments
on orders.order_id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

"""
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ['Years','Months','Sales','Cumulative_Sales'])
df.head(5)
```

| | Years | Months | Sales | Cumulative_Sales |
|---|---|---|---|---|
| **0** | 2016 | 9 | 252.24 | 252.24 |
| **1** | 2016 | 10 | 59090.48 | 59342.72 |
| **2** | 2016 | 12 | 19.62 | 59362.34 |
| **3** | 2017 | 1 | 138488.04 | 197850.38 |
| **4** | 2017 | 2 | 291908.01 | 489758.39 |

```
sns.lineplot(data=df,x='Years',y = 'Cumulative_Sales',color = 'red')
plt.xticks(rotation = 90)
plt.title("Cumulative Sales vs Year")
```

```
Text(0.5, 1.0, 'Cumulative Sales vs Year')
```



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

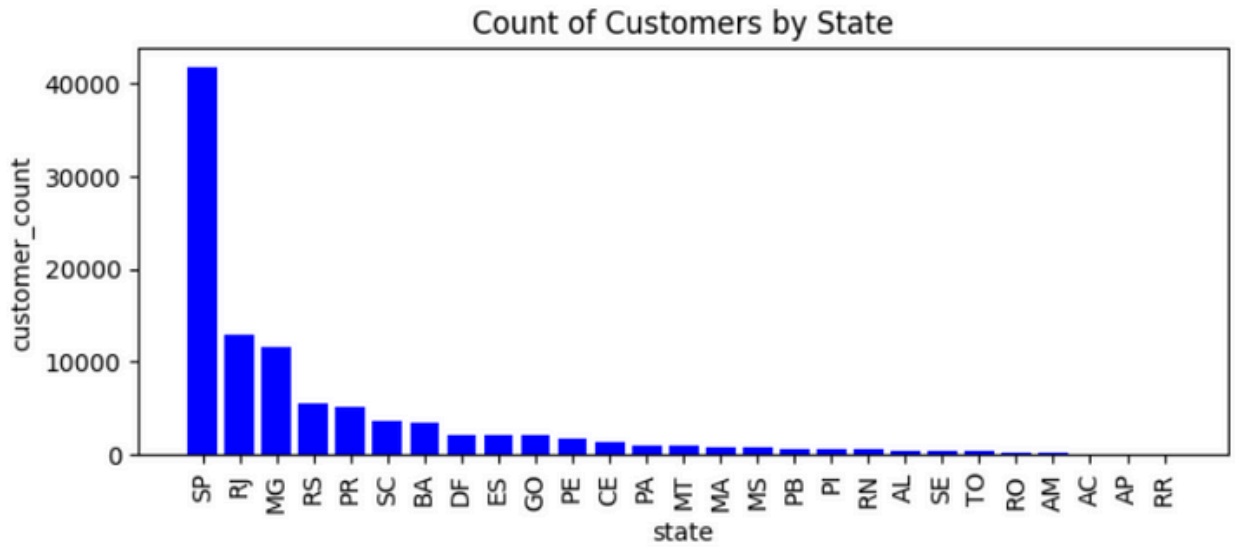
```
query = """with a as( select year(orders.order_purchase_timestamp) as years,
round(sum(payments.payment_value),2) as payment from orders join payments
on orders.order_id = payments.order_id
group by years order by years)

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
```

| | Years | YoY_%_Growth |
|---|---|---|
| 0 | 2016 | NaN |
| 1 | 2017 | 12112.703761 |
| 2 | 2018 | 20.000924 |

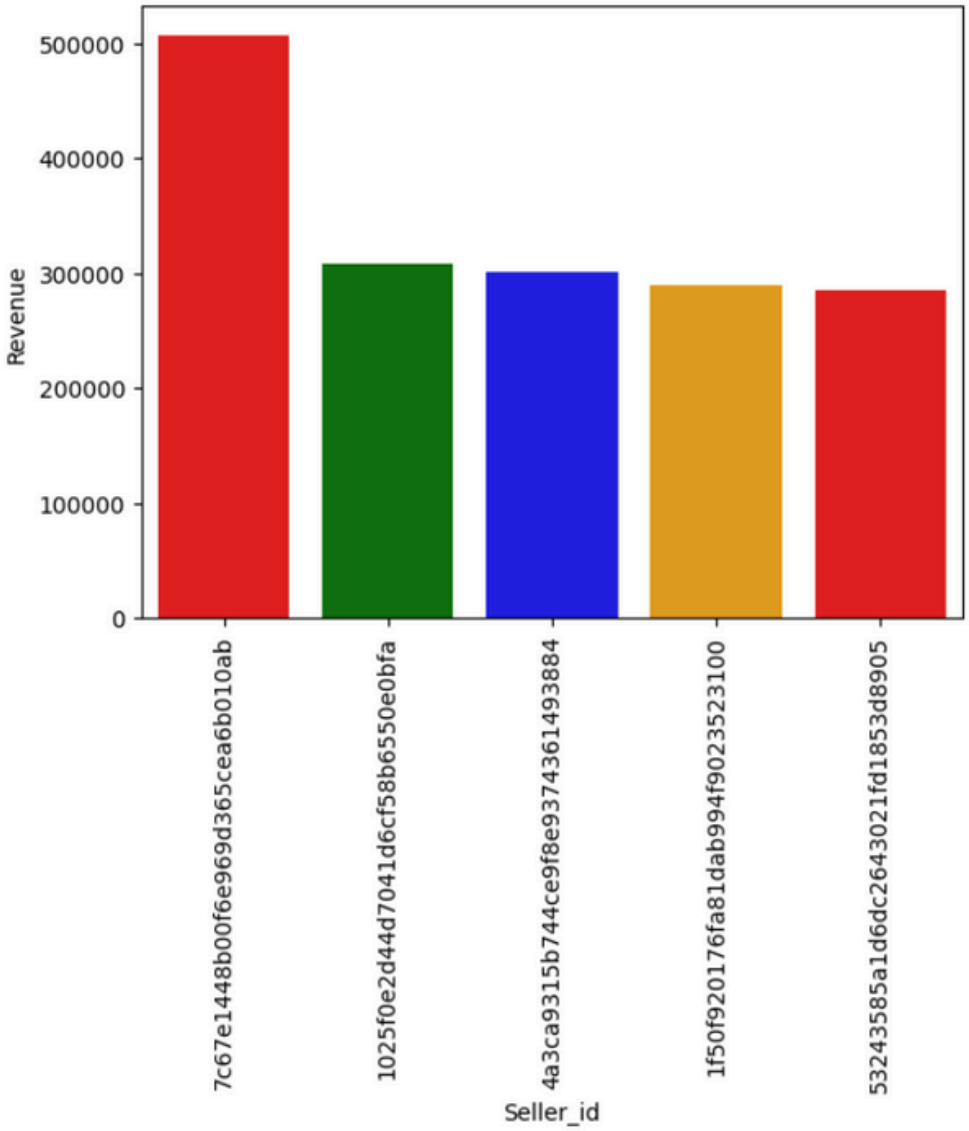
# 5. Count the number of customers from each state.

```
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 = (8,3))
plt.bar(df["state"],df["customer_count"],color= 'blue')
plt.xlabel("state")
plt.ylabel("customer_count")
plt.title('Count of Customers by State' )
plt.xticks(rotation=90)
plt.show()
```



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

```
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
"""
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data,columns = ['Seller_id','Revenue','Rank'])
df= df.head()
custom_colors = ['red', 'green', 'blue', 'orange']
sns.barplot(x = 'Seller_id',y = 'Revenue',data= df,palette = custom_colors)
plt.xticks(rotation = 90)
plt.show()
```




# CALCULATE THE PERCENTAGE OF ORDERS THAT WERE PAID IN INSTALLMENTS.

```
query = """select (sum(case when payment_installments >= 1 then 1
else 0 end))/ count(*)*100 from payments
"""
cur.execute(query)
data = cur.fetchall()
"the percentage of orders that were paid in installments is:", data[0][0]
```

```
('the percentage of orders that were paid in installments is:',
 Decimal('99.9981'))
```

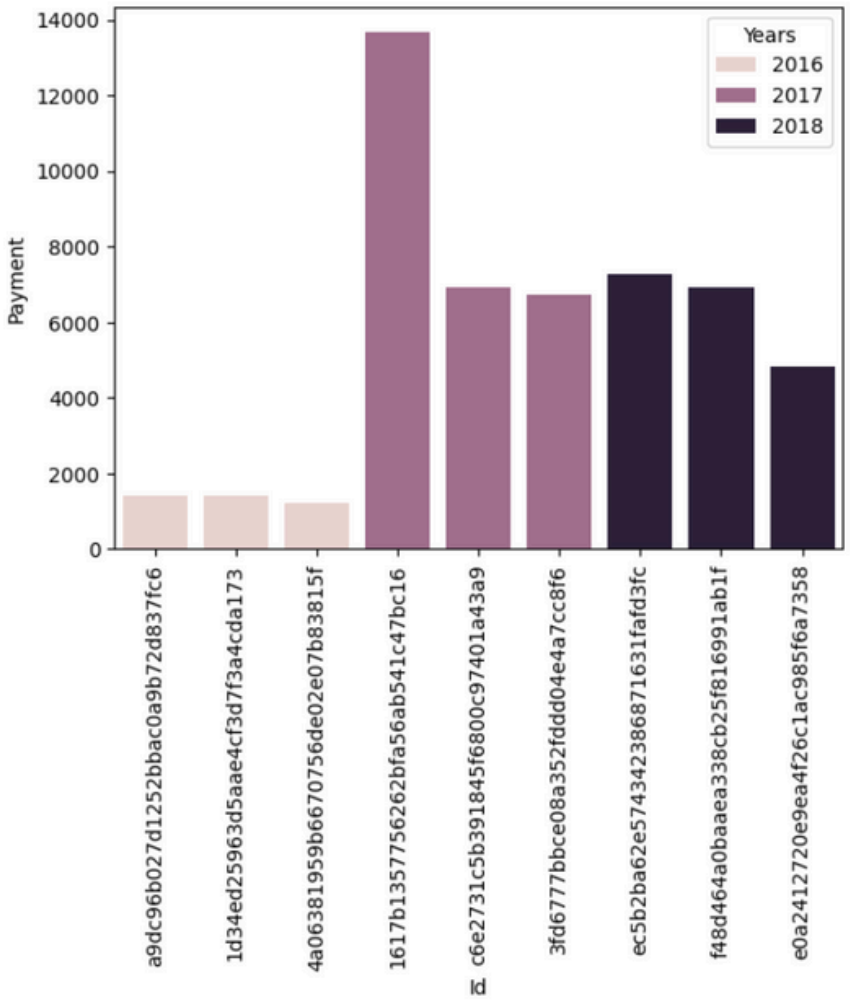
# IDENTIFY THE TOP 3 CUSTOMERS WHO SPENT THE MOST MONEY IN EACH YEAR.

```
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" ])
df.head()
```

| | Years | Id | Payment | Rank |
|---|---|---|---|---|
| 0 | 2016 | a9dc96b027d1252bbac0a9b72d837fc6 | 1423.550049 | 1 |
| 1 | 2016 | 1d34ed25963d5aae4cf3d7f3a4cda173 | 1400.739990 | 2 |
| 2 | 2016 | 4a06381959b6670756de02e07b83815f | 1227.780029 | 3 |
| 3 | 2017 | 1617b1357756262bfa56ab541c47bc16 | 13664.080078 | 1 |
| 4 | 2017 | c6e2731c5b391845f6800c97401a43a9 | 6929.310059 | 2 |

```
sns.barplot(x = 'Id',y = 'Payment',data = df,hue = 'Years')
plt.xticks(rotation = 90)
plt.show()
```



## IDENTIFY THE TOP 3 CUSTOMERS WHO SPENT THE MOST MONEY IN EACH YEAR.

```
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" ])
df.head()
```

| | Years | Id | Payment | Rank |
|---|---|---|---|---|
| 0 | 2016 | a9dc96b027d1252bbac0a9b72d837fc6 | 1423.550049 | 1 |
| 1 | 2016 | 1d34ed25963d5aae4cf3d7f3a4cda173 | 1400.739990 | 2 |
| 2 | 2016 | 4a06381959b6670756de02e07b83815f | 1227.780029 | 3 |
| 3 | 2017 | 1617b1357756262bfa56ab541c47bc16 | 13664.080078 | 1 |
| 4 | 2017 | c6e2731c5b391845f6800c97401a43a9 | 6929.310059 | 2 |

## CALCULATE THE MOVING AVERAGE OF ORDER VALUES FOR EACH CUSTOMER OVER THEIR ORDER HISTORY.

```
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

"""
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data,columns = ['Order_id','Timestamp','Avg_Order_Value','Moving_Avg_Order_Value'])
df.head()
```

| | Order_id | Timestamp | Avg_Order_Value | Moving_Avg_Order_Value |
|---|---|---|---|---|
| 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 |

## CONTACT INFORMATION

For further discussions or inquiries, please feel free to reach out:


- Name: Vijay Kumar Shah
- Email: vijaykumarshah1942@gmail.com
- LinkedIn: linkedin.com/in/vijay-kumar-shah-a47b0920a
- Github: GitHub.com/VIJAY626404


# THANK YOU