Comprehensive E-commerce Sales Analysis using SQL

Introduction

This project involves analyzing an e-commerce dataset using SQL to extract valuable insights into customer behavior, sales trends, and product performance. The analysis includes basic to advanced queries, such as calculating total sales per category, identifying customer retention rates, and determining the top customers by spending, offering a thorough understanding of the business dynamics.

Customers

customer_id	customer_unique_id	customer_zip_code_prefix	customer_city	customer_state
06b8999e2fba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0	14409	franca	SP
18955e83d337fd6b2def6b18a428ac77	290c77bc529b7ac935b93aa66c333dc3	9790	sao bernardo do campo	SP
4e7b3e00288586ebd08712fdd0374a03	060e732b5b29e8181a18229c7b0b2b5e	1151	sao paulo	SP
b2b6027bc5c5109e529d4dc6358b12c3	259dac757896d24d7702b9acbbff3f3c	8775	mogi das cruzes	SP
4f2d8ab171c80ec8364f7c12e35b23ad	345ecd01c38d18a9036ed96c73b8d066	13056	campinas	SP
879864dab9bc3047522c92c82e1212b8	4c93744516667ad3b8f1fb645a3116a4	89254	jaragua do sul	SC

Geolocation

geolocation_zip_code_prefix	geolocation_lat	geolocation_lng	geolocation_city	geolocation_state
1037	-23.54562128	-46.63929205	sao paulo	SP
1046	-23.54608113	-46.6448203	sao paulo	SP
1046	-23.54612897	-46.64295148	sao paulo	SP
1041	-23.54439216	-46.63949931	sao paulo	SP
1035	-23.54157796	-46.64160722	sao paulo	SP
1012	-23.5477623	-46.63536054	são paulo	SP

Order_items

order_id	order_item_id	product_id	seller_id	shipping_limit_date	price	freight_value
00010242fe8c5a6d1ba2dd792cb16214	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	9/19/2017 9:45	58.9	13.29
00018f77f2f0320c557190d7a144bdd3	1	e5f2d52b802189ee658865ca93d83a8f	dd7ddc04e1b6c2c614352b383efe2d36	5/3/2017 11:05	239.9	19.93
000229ec398224ef6ca0657da4fc703e	1	c777355d18b72b67abbeef9df44fd0fd	5b51032eddd242adc84c38acab88f23d	1/18/2018 14:48	199	17.87
00024acbcdf0a6daa1e931b038114c75	1	7634da152a4610f1595efa32f14722fc	9d7a1d34a5052409006425275ba1c2b4	8/15/2018 10:10	12.99	12.79
00042b26cf59d7ce69dfabb4e55b4fd9	1	ac6c3623068f30de03045865e4e10089	df560393f3a51e74553ab94004ba5c87	2/13/2017 13:57	199.9	18.14
00048cc3ae777c65dbb7d2a0634bc1ea	1	ef92defde845ab8450f9d70c526ef70f	6426d21aca402a131fc0a5d0960a3c90	5/23/2017 3:55	21.9	12.69

orders

order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at	order_delivered_carrier_date	order_delivered_customer_date	order_estimated_delivery_date
e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	10/2/2017 10:56	10/2/2017 11:07	10/4/2017 19:55	10/10/2017 21:25	10/18/2017 0:00
53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	7/24/2018 20:41	7/26/2018 3:24	7/26/2018 14:31	8/7/2018 15:27	8/13/2018 0:00
47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	8/8/2018 8:38	8/8/2018 8:55	8/8/2018 13:50	8/17/2018 18:06	9/4/2018 0:00
949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	11/18/2017 19:28	11/18/2017 19:45	11/22/2017 13:39	12/2/2017 0:28	12/15/2017 0:00
ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2/13/2018 21:18	2/13/2018 22:20	2/14/2018 19:46	2/16/2018 18:17	2/26/2018 0:00
a4591c265e18cb1dcee52889e2d8acc3	503740e9ca751ccdda7ba28e9ab8f608	delivered	7/9/2017 21:57	7/9/2017 22:10	7/11/2017 14:58	7/26/2017 10:57	8/1/2017 0:00

payment

order_id	payment_sequential	payment_type	payment_installments	payment_value
b81ef226f3fe1789b1e8b2acac839d17	1	credit_card	8	99.33
a9810da82917af2d9aefd1278f1dcfa0	1	credit_card	1	24.39
25e8ea4e93396b6fa0d3dd708e76c1bd	1	credit_card	1	65.71
ba78997921bbcdc1373bb41e913ab953	1	credit_card	8	107.78
42fdf880ba16b47b59251dd489d4441a	1	credit_card	2	128.45
298fcdf1f73eb413e4d26d01b25bc1cd	1	credit_card	2	96.12

Product

product_id	product category	product_name_length	product_description_length	product_photos_qty	product_weight_g	product_length_cm	product_height_cm	product_width_cm
1e9e8ef04dbcff4541ed26657ea517e5	perfumery	40	287	1	225	16	10	14
3aa071139cb16b67ca9e5dea641aaa2f	Art	44	276	1	1000	30	18	20
96bd76ec8810374ed1b65e291975717f	sport leisure	46	250	1	154	18	9	15
cef67bcfe19066a932b7673e239eb23d	babies	27	261	1	371	26	4	26
9dc1a7de274444849c219cff195d0b71	housewares	37	402	4	625	20	17	13
41d3672d4792049fa1779bb35283ed13	musical instruments	60	745	1	200	38	5	11

Sellere

order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at	order_delivered_carrier_date	order_delivered_customer_date	order_estimated_delivery_date
e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	10/2/2017 10:56	10/2/2017 11:07	10/4/2017 19:55	10/10/2017 21:25	10/18/2017 0:00
53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	7/24/2018 20:41	7/26/2018 3:24	7/26/2018 14:31	8/7/2018 15:27	8/13/2018 0:00
47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	8/8/2018 8:38	8/8/2018 8:55	8/8/2018 13:50	8/17/2018 18:06	9/4/2018 0:00
949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	11/18/2017 19:28	11/18/2017 19:45	11/22/2017 13:39	12/2/2017 0:28	12/15/2017 0:00
ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2/13/2018 21:18	2/13/2018 22:20	2/14/2018 19:46	2/16/2018 18:17	2/26/2018 0:00
a4591c265e18cb1dcee52889e2d8acc3	503740e9ca751ccdda7ba28e9ab8f608	delivered	7/9/2017 21:57	7/9/2017 22:10	7/11/2017 14:58	7/26/2017 10:57	8/1/2017 0:00

PYTHON 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
conn = mysql.connector.connect(
    host='localhost',
    user='root',
    password='Ali242.@com',
    database='E_Commerce_Sales_Dataset'
)
cursor = conn.cursor()

# Folder containing the CSV files
folder_path = 'C:/Users/hst/Desktop/Data analytics/sql+python project'
```

```
def get_sql_type(dtype):
    if pd.api.types.is_integer_dtype(dtype):
        return 'INT'
    elif pd.api.types.is_float_dtype(dtype):
        return 'FLOAT'
    elif pd.api.types.is_bool_dtype(dtype):
        return 'BOOLEAN'
    elif pd.api.types.is_datetime64_any_dtype(dtype):
        return 'DATETIME'
    else:
        return 'TEXT'

for csv_file, table_name in csv_files:
    file_path = os.path.join(folder_path, csv_file)

    # Read the CSV file into a pandas DataFrame
    df = pd.read_csv(file_path)

    # Replace NaN with None to handle SQL NULL
    df = df.where(pd.notnull(df), None)

    # Debugging: Check for NaN values
    print(f"Processing {csv_file}")
    print(f"NaN values before replacement:\n{df.isnull().sum()}\n")

    # Clean column names
    df.columns = [col.replace(' ', '_').replace('-', '_').replace('.', '_') for col in df.columns]
```

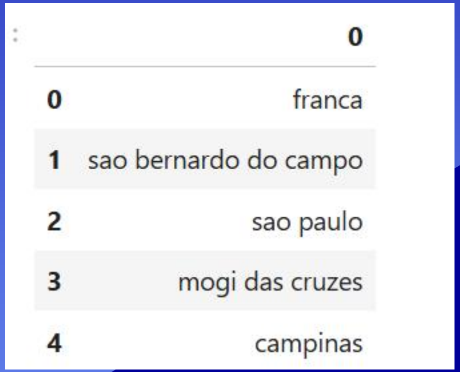
```
# Generate the CREATE TABLE statement with appropriate data types
    columns = ', '.join([f'`{col}` {get_sql_type(df[col].dtype)}' for col in df.columns])
    create_table_query = f'CREATE TABLE IF NOT EXISTS `{table_name}` ({columns})'
    cursor.execute(create_table_query)

    # Insert DataFrame data into the MySQL table
    for _, row in df.iterrows():
        # Convert row to tuple and handle NaN/None explicitly
        values = tuple(None if pd.isna(x) else x for x in row)
        sql = f"INSERT INTO `{table_name}` ({', '.join(['`' + col + '`' for col in df.columns])}) VALUES ({', '.join(['%s'] * len(row))})"
        cursor.execute(sql, values)

    # Commit the transaction for the current CSV file
    conn.commit()

# Close the connection
conn.close()

pip install pandas
pip install mysql-connector-python
pip install matplotlib
pip install seaborn
cursor = conn.cursor()
```

Lets Solve Queries

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

# Connect to the MySQL database
conn = mysql.connector.connect(
    host='localhost',
    user='root',
    password='Ali242.@com',
    database='E_Commerce_Sales_Dataset'
)
# activate cursor in database
cursor = conn.cursor()
```

1. List all unique cities where customers are located.

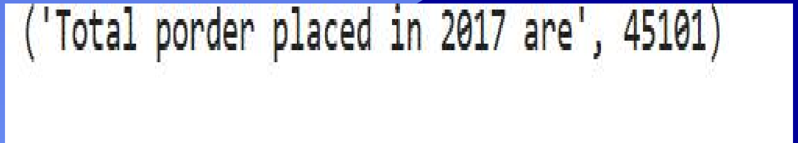
```
# query= sql code
query = """ select distinct customer_city from customers """
#execute the query
cursor.execute(query)
#to show dat here from sql
data = cursor.fetchall()

df = pd.DataFrame(data)
df.head()
```



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

2. Count the number of orders placed in 2017.

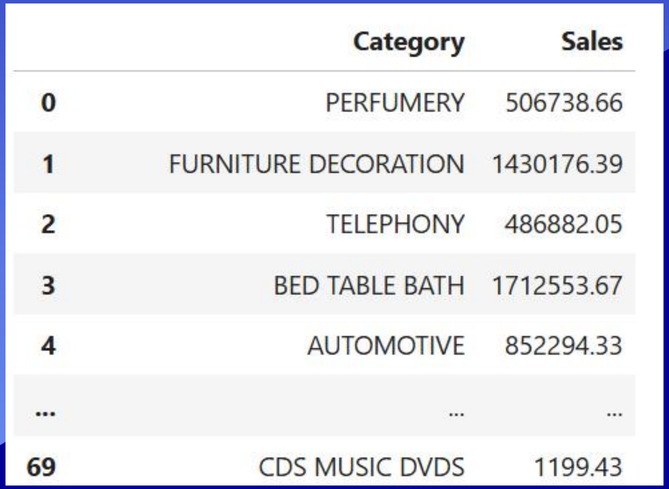
```
# query= sql code
query = """ select count(order_id) from orders where
year(order_purchase_timestamp)=2017 """
#execute the query
cursor.execute(query)
#to show dat here from sql
data = cursor.fetchall()

data       #[(45101,)]
data[0]    #(45101,)
data[0][0] #45101
"Total porder placed in 2017 are", data[0][0]
```



```
('Total porder placed in 2017 are', 45101)
```

3. Find the total sales per category.

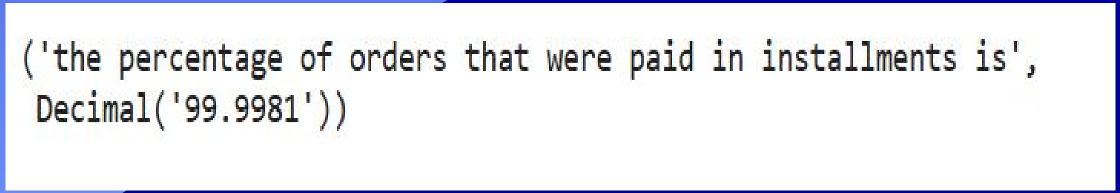
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import mysql.connector

# Connect to the MySQL database
conn = mysql.connector.connect(
    host='localhost',
    user='root',
    password='Ali242.@com',
    database='E_Commerce_Sales_Dataset'
)
# activate cursor in database
cursor = conn.cursor()
```



	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

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

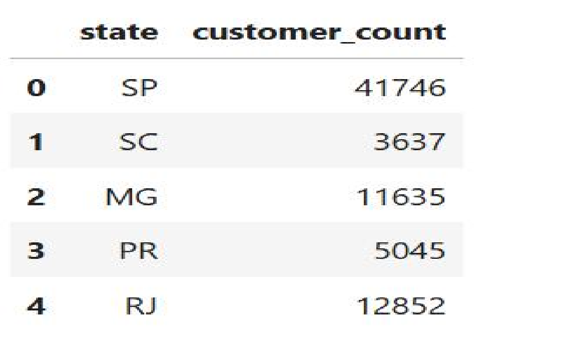
```
query = """ select ((sum(case when payment_installments >= 1 then
1
else 0 end))/count(*))*100
from payments
"""


cursor.execute(query)


data = cursor.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'))
```

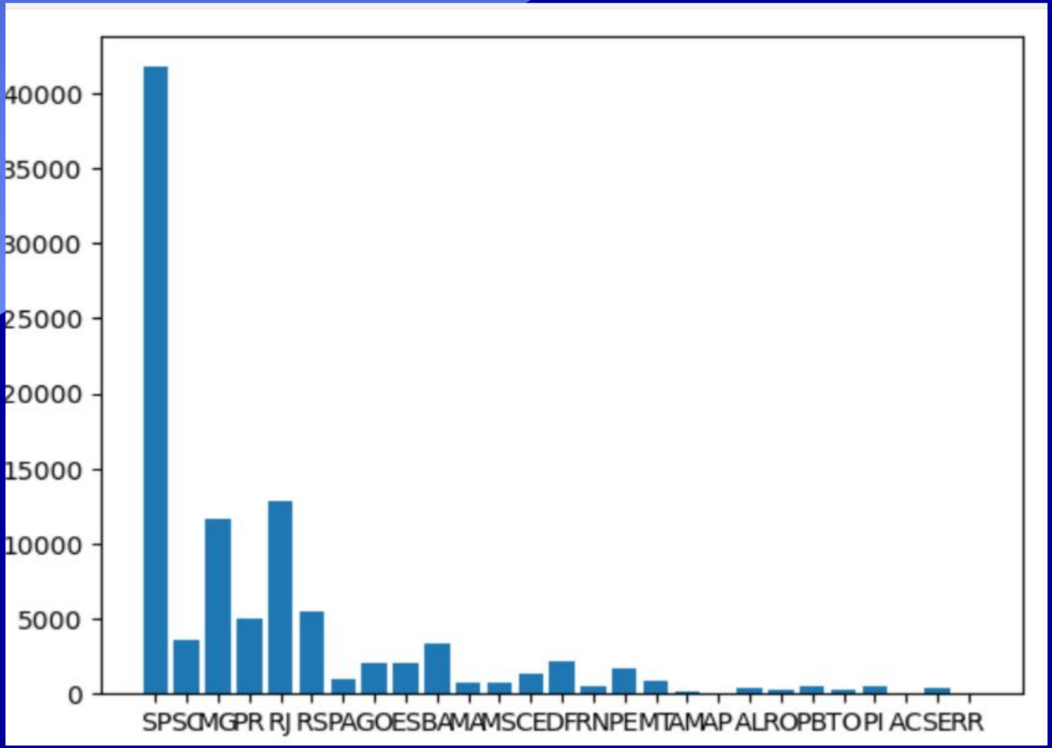
5. Count the number of customers from each state.

```
query = """ select customer_state ,count(customer_id)
from customers group by customer_state
"""
cursor.execute(query)
data = cursor.fetchall()
df = pd.DataFrame(data, columns = ["state", "customer_count" ])
df.head()
```



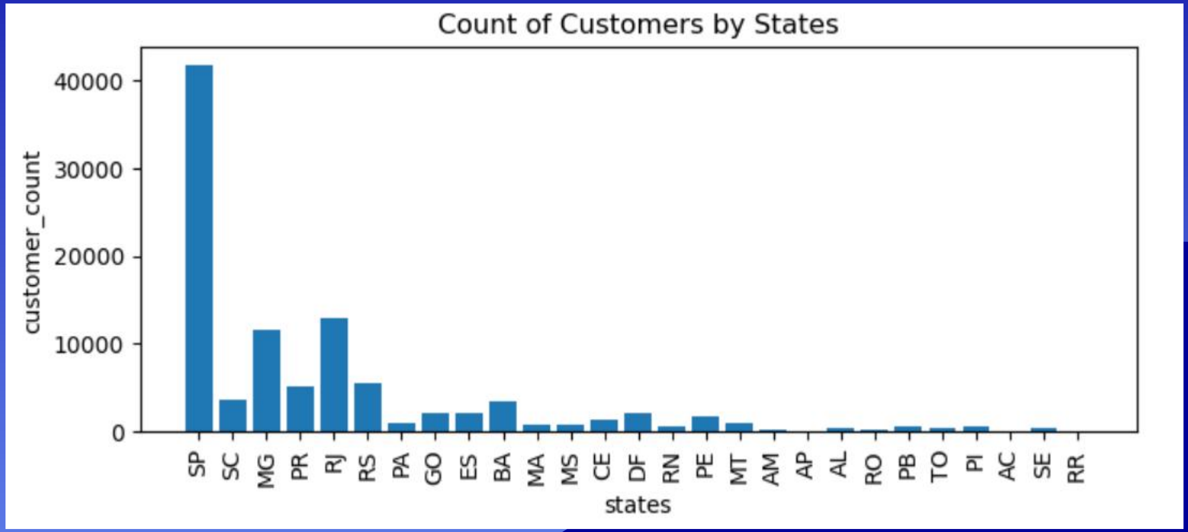
	state	customer_count
0	SP	41746
1	SC	3637
2	MG	11635
3	PR	5045
4	RJ	12852

```
#bar plot
plt.bar(df["state"], df["customer_count"])
plt.show()
```

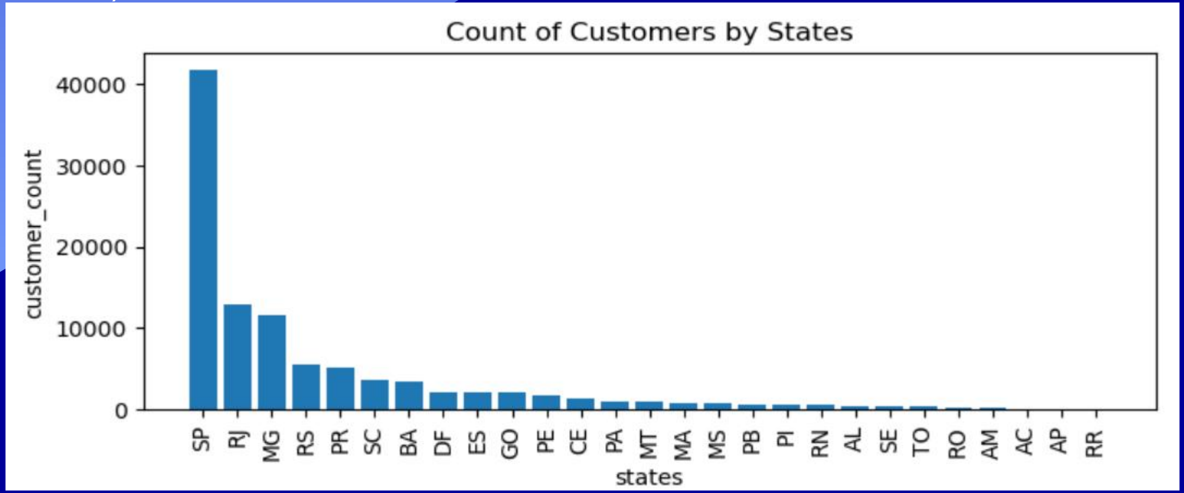


Adjust the BAR plot

```
plt.figure(figsize = (8,3))
plt.bar(df["state"], df["customer_count"])
plt.xticks(rotation = 90)
plt.xlabel("states")
plt.ylabel("customer_count")
plt.title("Count of Customers by States")
plt.show()
```

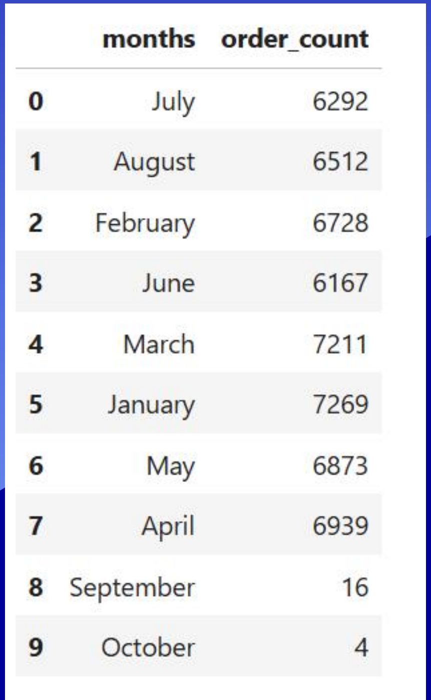


```
df = df.sort_values(by = "customer_count", ascending= False)
plt.figure(figsize = (8,3))
plt.bar(df["state"], df["customer_count"])
plt.xticks(rotation = 90)
plt.xlabel("states")
plt.ylabel("customer_count")
plt.title("Count of Customers by States")
plt.show()
```



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

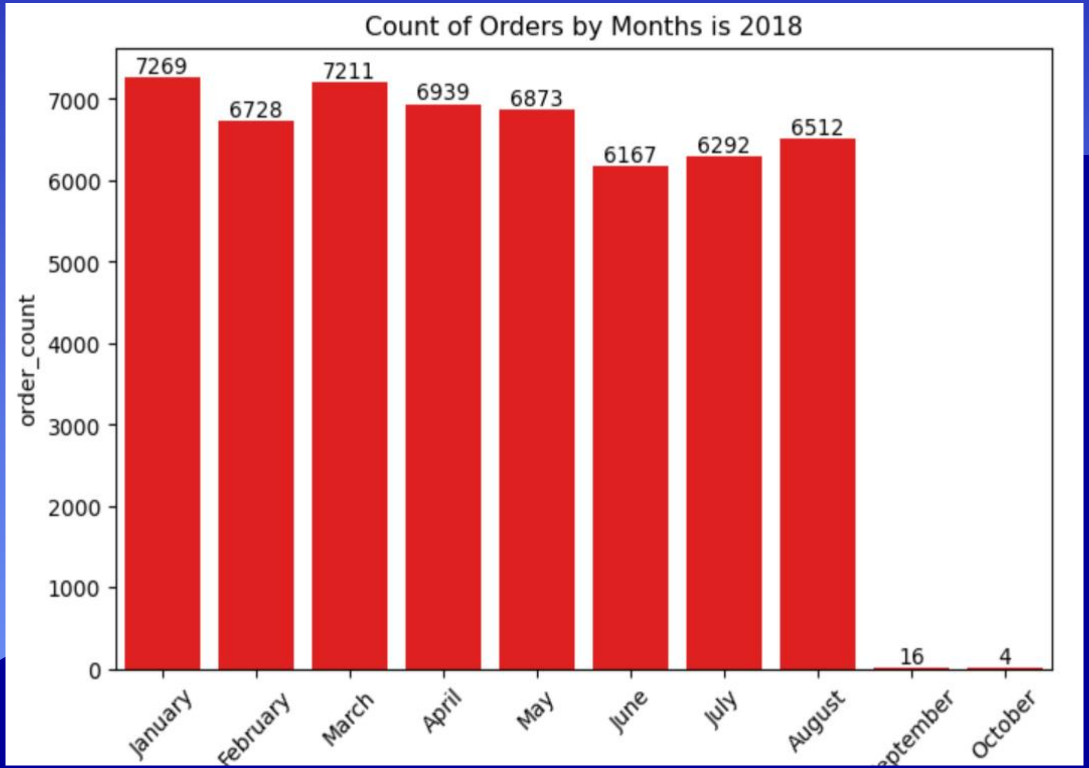
```
query = """ select monthname(order_purchase_timestamp) months,
count(order_id) order_count
from orders where year(order_purchase_timestamp) = 2018
group by months
"""

cursor.execute(query)

data = cursor.fetchall()
df = pd.DataFrame(data, columns = ["months", "order_count"])
df
```



	months	order_count
0	July	6292
1	August	6512
2	February	6728
3	June	6167
4	March	7211
5	January	7269
6	May	6873
7	April	6939
8	September	16
9	October	4

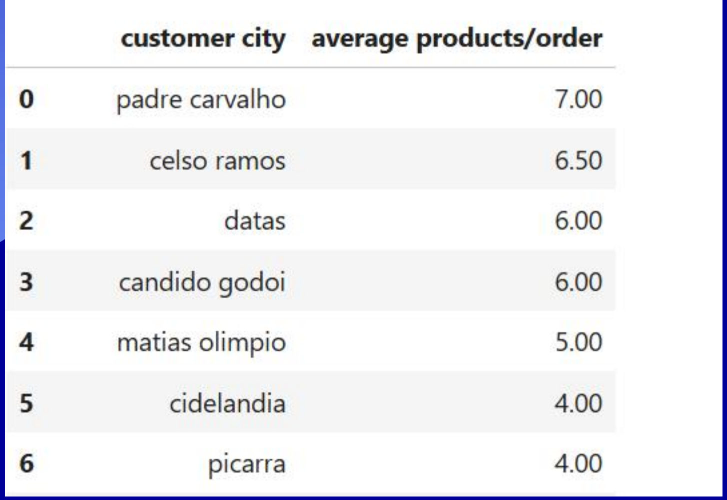
SNS PLOT

```
o = ["January",
"February","March","April","May","June","July","August","September",
"October"]

#by seaborn
plt.figure(figsize = (8,5))
ax = sns.barplot(x = df["months"],y =  df["order_count"], data = df,
order = o, color = "red")
plt.xticks(rotation = 45)
ax.bar_label(ax.containers[0])
plt.title("Count of Orders by Months is 2018")
plt.show()
```



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

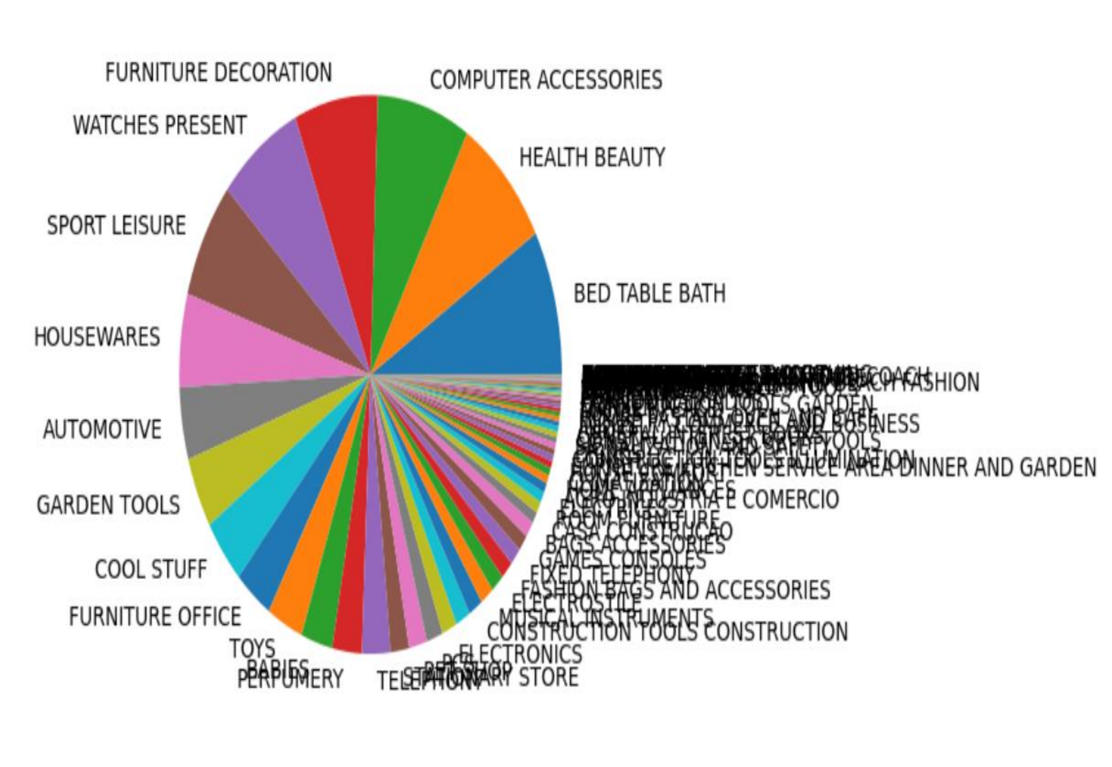
```
query = """with count_per_order as
(select orders.order_id, orders.customer_id, count(order_items.order_id) as oc
from orders join order_items
on orders.order_id = order_items.order_id
group by orders.order_id, orders.customer_id)

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
"""


cursor.execute(query)

data = cursor.fetchall()
df = pd.DataFrame(data,columns = ["customer city", "average products/order"])
df.head(10)
```



	customer city	average products/order
0	padre carvalho	7.00
1	celso ramos	6.50
2	datas	6.00
3	candido godoi	6.00
4	matias olimpio	5.00
5	cidelandia	4.00
6	picarra	4.00

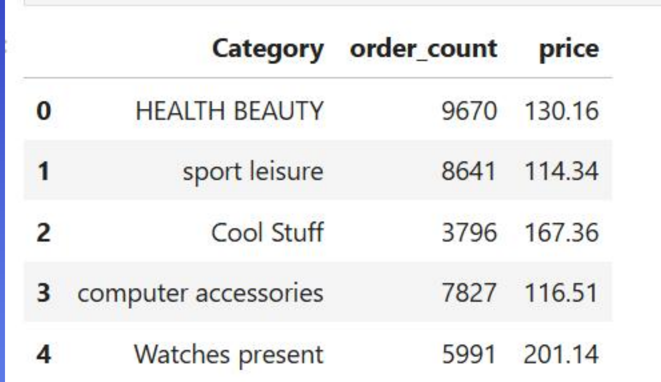
8. Calculate the percentage of total revenue contributed by each product category.

```
query = """select upper(products.product_category) category,
(sum(payments.payment_value)/(select sum(payment_value)
from payments))*100 Per_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
order by Per_Sales desc;
"""


cursor.execute(query)


data = cursor.fetchall()
df = pd.DataFrame(data,columns = ["category", "Per_Sales"])
df.head(5)
#pie chart
plt.pie(df["Per_Sales"],labels=df["category"])
plt.show
```

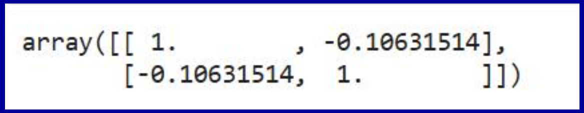


9. Identify the correlation between product price and the number of times a product has been purchased.

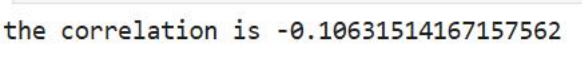
```
import numpy as np
query = """select products.product_category, count(order_items.product_id),
round(avg(order_items.price),2)
from products join order_items
on products.product_id = order_items.product_id
group by products.product_category"""
cursor.execute(query)
data = cursor.fetchall()
df = pd.DataFrame(data,columns = ["Category", "order_count","price"])
df.head(5)

# coorelation
arr1 = df["order_count"]
arr2 = df["price"]
np.corrcoef([arr1,arr2])


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



	Category	order_count	price
0	HEALTH BEAUTY	9670	130.16
1	sport leisure	8641	114.34
2	Cool Stuff	3796	167.36
3	computer accessories	7827	116.51
4	Watches present	5991	201.14



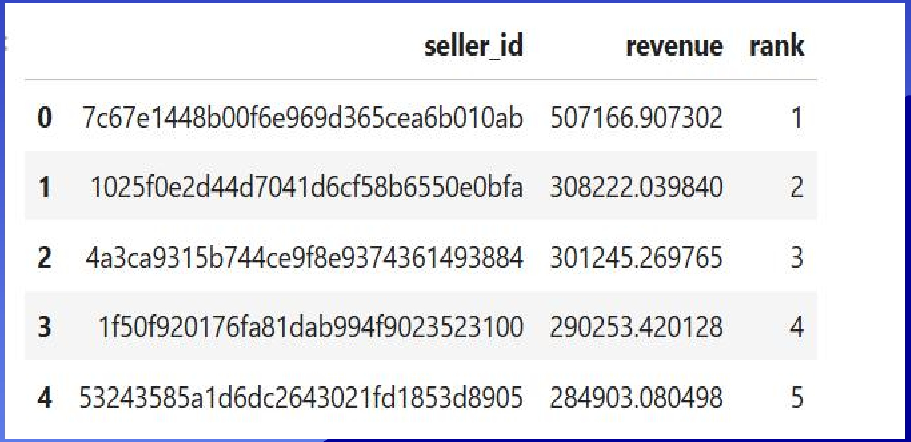
```
array([[ 1.        , -0.10631514],
       [-0.10631514,  1.        ]])
```



```
the correlation is -0.10631514167157562
```

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

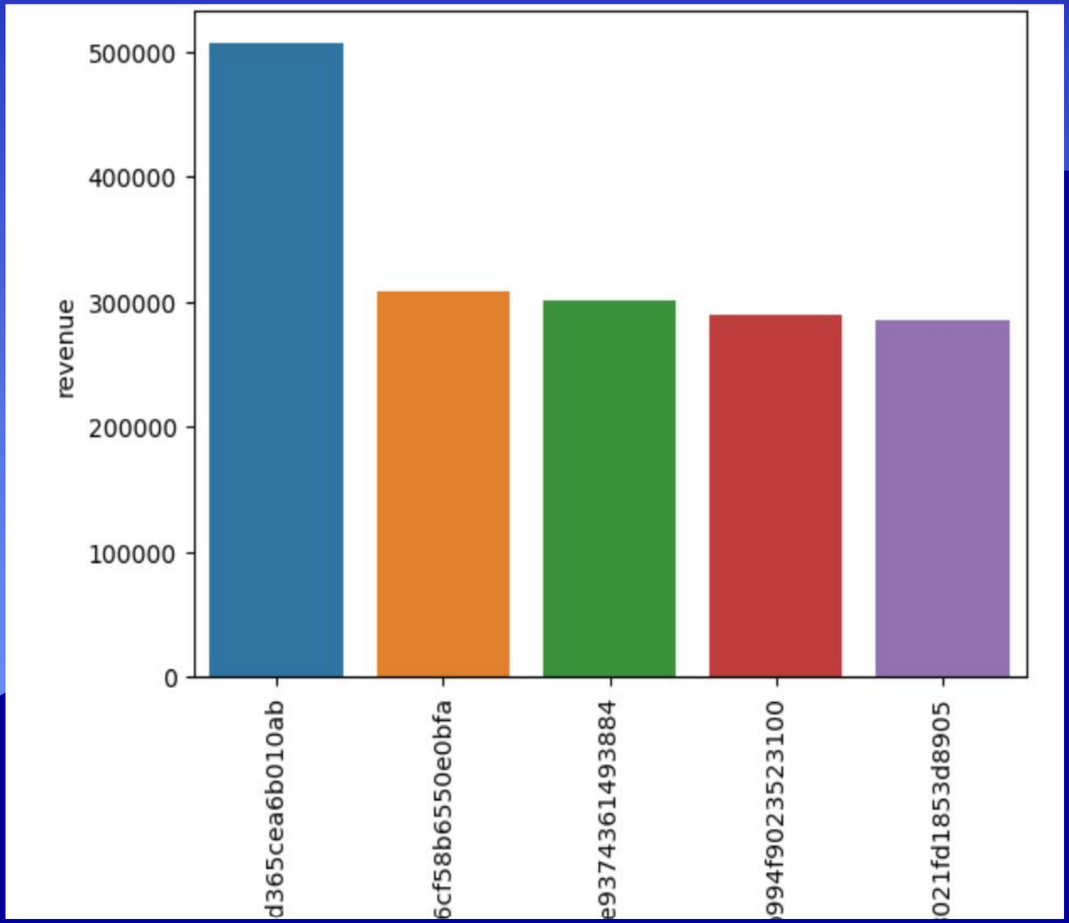
```
query = """ select *,
dense_rank() over(order by revenue desc) as rn
from
(select order_items.seller_id, sum(payments.payment_value)
revenue from order_items join payments
on order_items.order_id = payments.order_id
group by order_items.seller_id) as a """

cursor.execute(query)
data = cursor.fetchall()
df = pd.DataFrame(data, columns = ["seller_id", "revenue", "rank"])
df.head(5)
```



	seller_id	revenue	rank
0	7c67e1448b00f6e969d365cea6b010ab	507166.907302	1
1	1025f0e2d44d7041d6cf58b6550e0bfa	308222.039840	2
2	4a3ca9315b744ce9f8e9374361493884	301245.269765	3
3	1f50f920176fa81dab994f9023523100	290253.420128	4
4	53243585a1d6dc2643021fd1853d8905	284903.080498	5

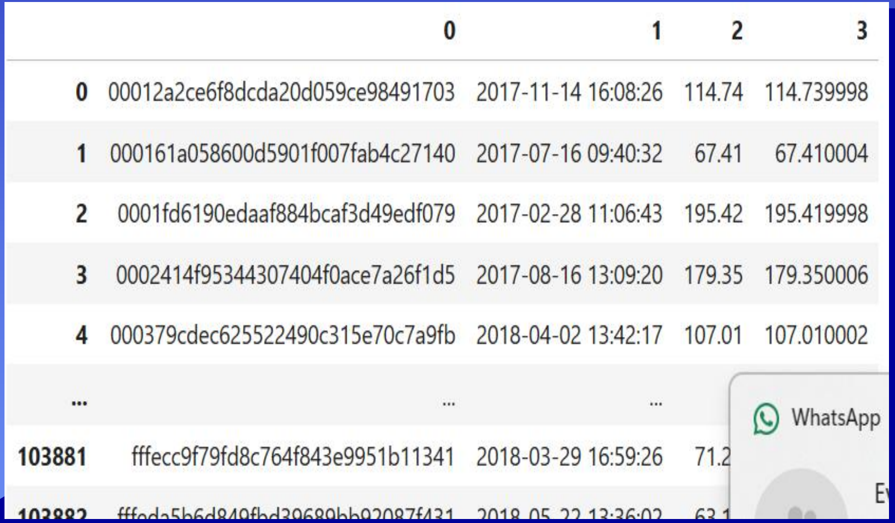
BAR PLOT

```
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 """

cursor.execute(query)
data = cursor.fetchall()
df = pd.DataFrame(data, columns = ["seller_id", "revenue", "rank"])
df = df.head()
sns.barplot(x = "seller_id", y = "revenue", data = df)
plt.xticks(rotation = 90)
plt.show()
```



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

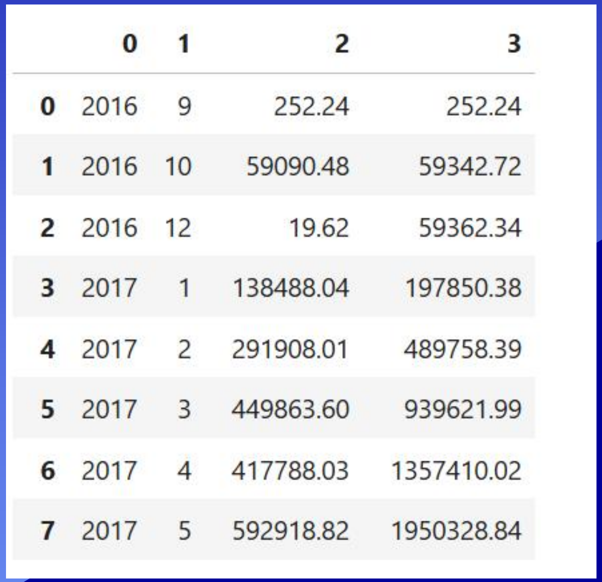
```
query = """ select *,
dense_rank() over(order by revenue desc) as rn
from
(select order_items.seller_id, sum(payments.payment_value)
revenue from order_items join payments
on order_items.order_id = payments.order_id
group by order_items.seller_id) as a """

cursor.execute(query)
data = cursor.fetchall()
df = pd.DataFrame(data, columns = ["seller_id", "revenue", "rank"])
df.head(5)
```



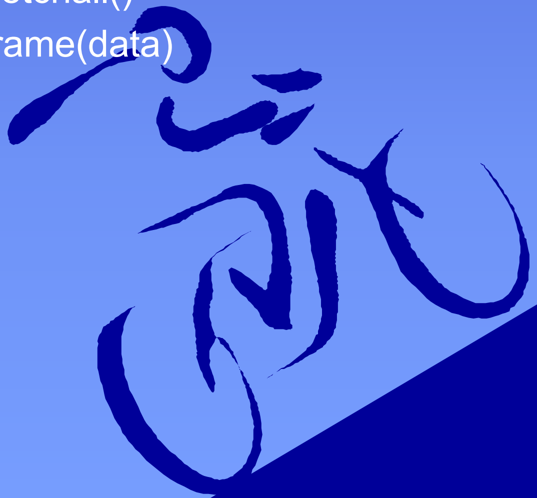
	0	1	2	3
0	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.74	114.739998
1	000161a058600d5901f007fab4c27140	2017-07-16 09:40:32	67.41	67.410004
2	0001fd6190edaaf884bcaf3d49edf079	2017-02-28 11:06:43	195.42	195.419998
3	0002414f95344307404f0ace7a26f1d5	2017-08-16 13:09:20	179.35	179.350006
4	000379cdec625522490c315e70c7a9fb	2018-04-02 13:42:17	107.01	107.010002
...	...	...		
103881	fffecc9f79fd8c764f843e9951b11341	2018-03-29 16:59:26	71.2	
103882	fffeda5b6d849fbd39689bb92087f431	2018-05-22 13:36:02	63.1	

12. 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
"""
cursor.execute(query)
data = cursor.fetchall()
df = pd.DataFrame(data)
df
```

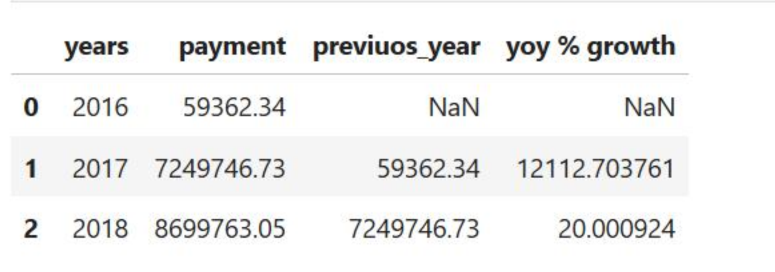


	0	1	2	3
0	2016	9	252.24	252.24
1	2016	10	59090.48	59342.72
2	2016	12	19.62	59362.34
3	2017	1	138488.04	197850.38
4	2017	2	291908.01	489758.39
5	2017	3	449863.60	939621.99
6	2017	4	417788.03	1357410.02
7	2017	5	592918.82	1950328.84



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

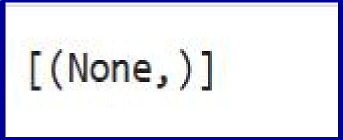
```
query = """with a as(select year(orders.order_purchase_timestamp)
as years,
round(sum(payments.payment_value),2) as payment from orders
join payments
on orders.order_id = payments.order_id
group by years order by years)

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

cursor.execute(query)
data = cursor.fetchall()
df = pd.DataFrame(data, columns =
["years","payment","previuos_year", "yoy % growth"])
df
```



	years	payment	previuos_year	yoy % growth
0	2016	59362.34	NaN	NaN
1	2017	7249746.73	59362.34	12112.703761
2	2018	8699763.05	7249746.73	20.000924

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

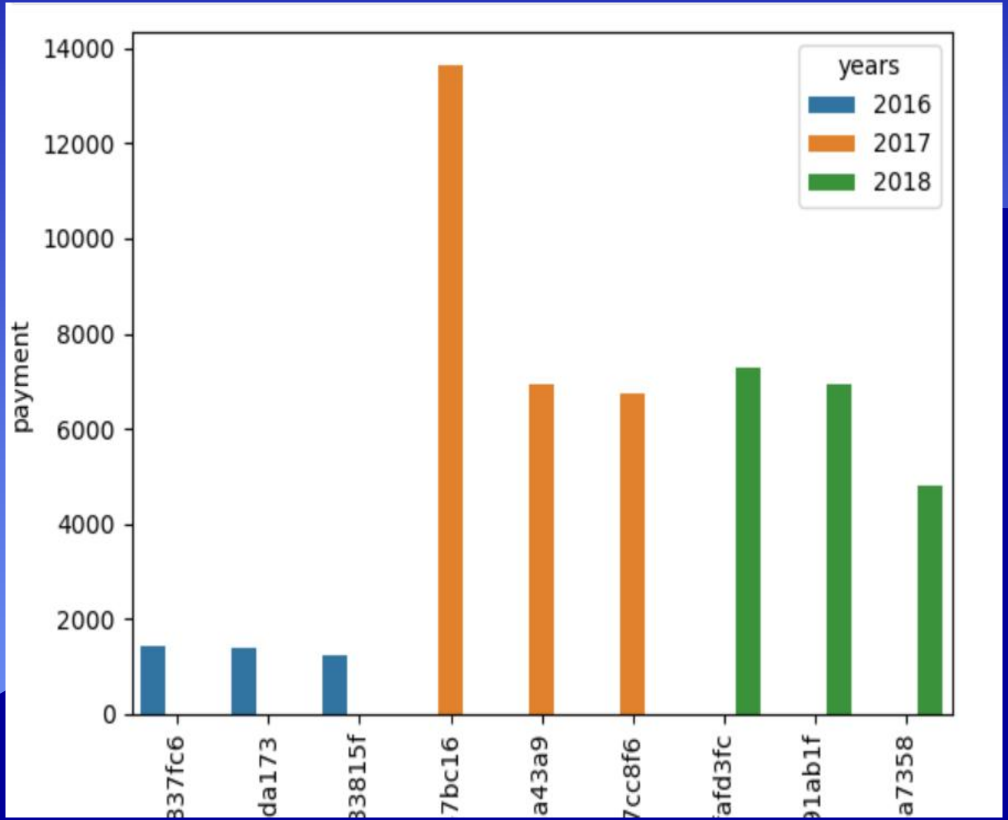
```
query = """with a as (select customers.customer_id,
min(orders.order_purchase_timestamp) first_order
from customers join orders
on customers.customer_id = orders.customer_id
group by customers.customer_id),

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 ;"""
cursor.execute(query)
data = cursor.fetchall()
data
```



```
[(None,)]
```

15. 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 ;"""

cursor.execute(query)
data = cursor.fetchall()
df = pd.DataFrame(data, columns = ["years","id","payment","rank"])
df
df['years'] = df['years'].astype(str)

sns.barplot(x = "id", y = "payment", data = df, hue = "years")
plt.xticks(rotation = 90)
plt.show()
```


years
payment
337fc6
da173
33815f
7bc16
a43a9
7cc8f6
afd3fc
91ab1f
a7358