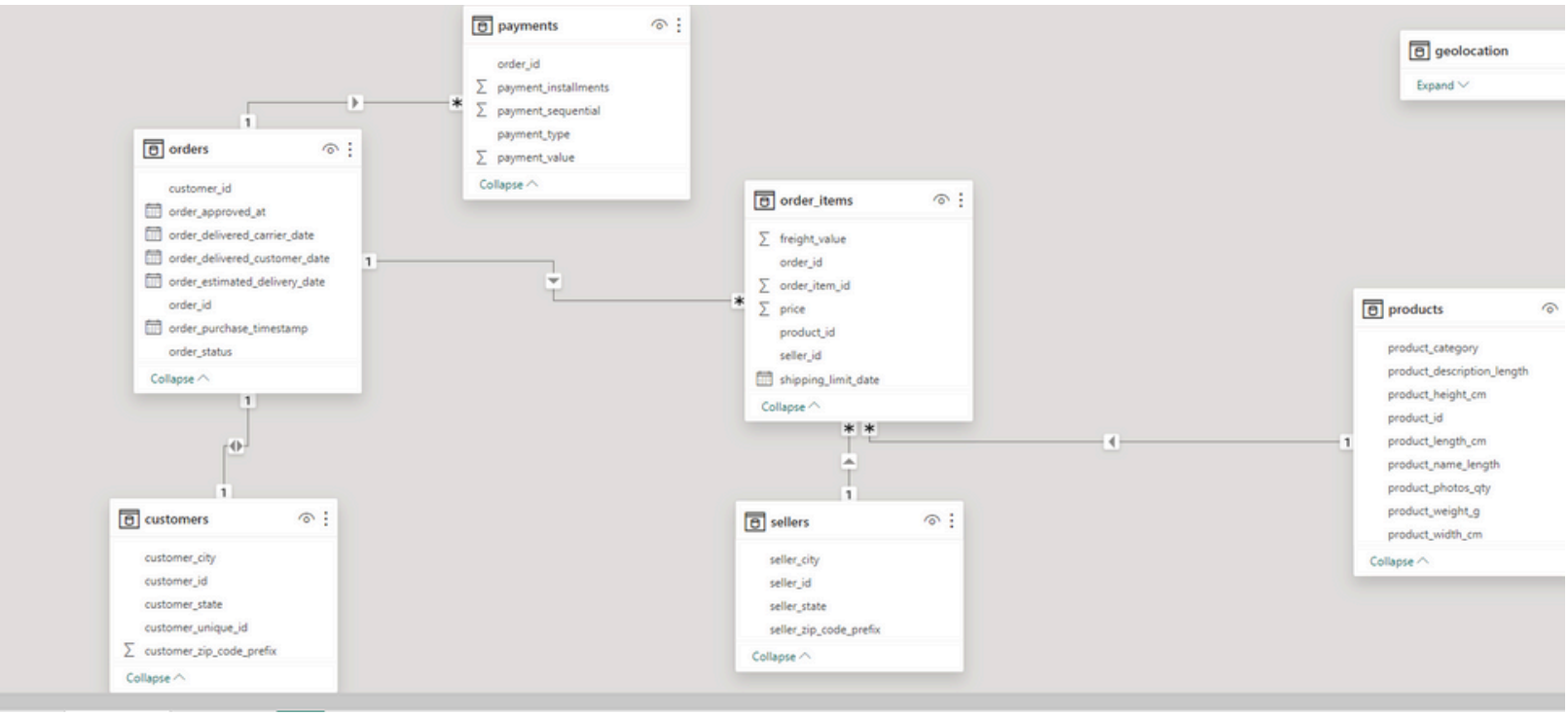
SQL_python_connection

```
 import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import mysql.connector
import numpy as np
db= mysql.connector.connect(host = "localhost",
                            username = "root",
                            password = "Mysql@12337",
                            database = "e_commerce")

cur = db.cursor()
```

Schema



##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)
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 = """Select count(order_id) from orders
where year(order_purchase_timestamp) = 2017"""


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

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

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

##Find the total sales per category.

```
query = """select products.product_category as category,
round(sum(payments.payment_value),2) as 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.head(10)
```

##Find the total sales per category.

```
query = """select products.product_category as category,
round(sum(payments.payment_value),2) as 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.head(10)
```

	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
5	computer accessories	1585330.45
6	housewares	1094758.13
7	babies	539845.66
8	toys	619037.69
9	Furniture office	646826.49

##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()
data
```

```
[(Decimal('99.9981'),)]
```

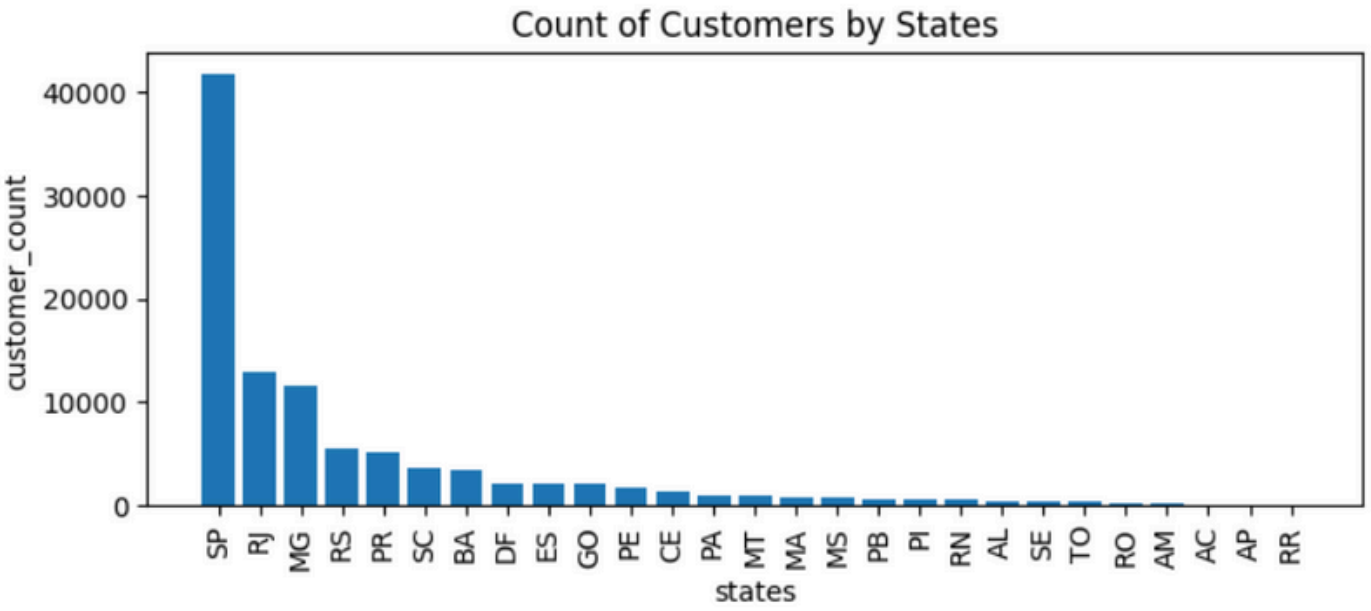
##Count the number of customers from each state.

```
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"])
plt.xticks(rotation = 90)
plt.xlabel("states")
plt.ylabel("customer_count")
plt.title("Count of Customers by States")
plt.show()
```

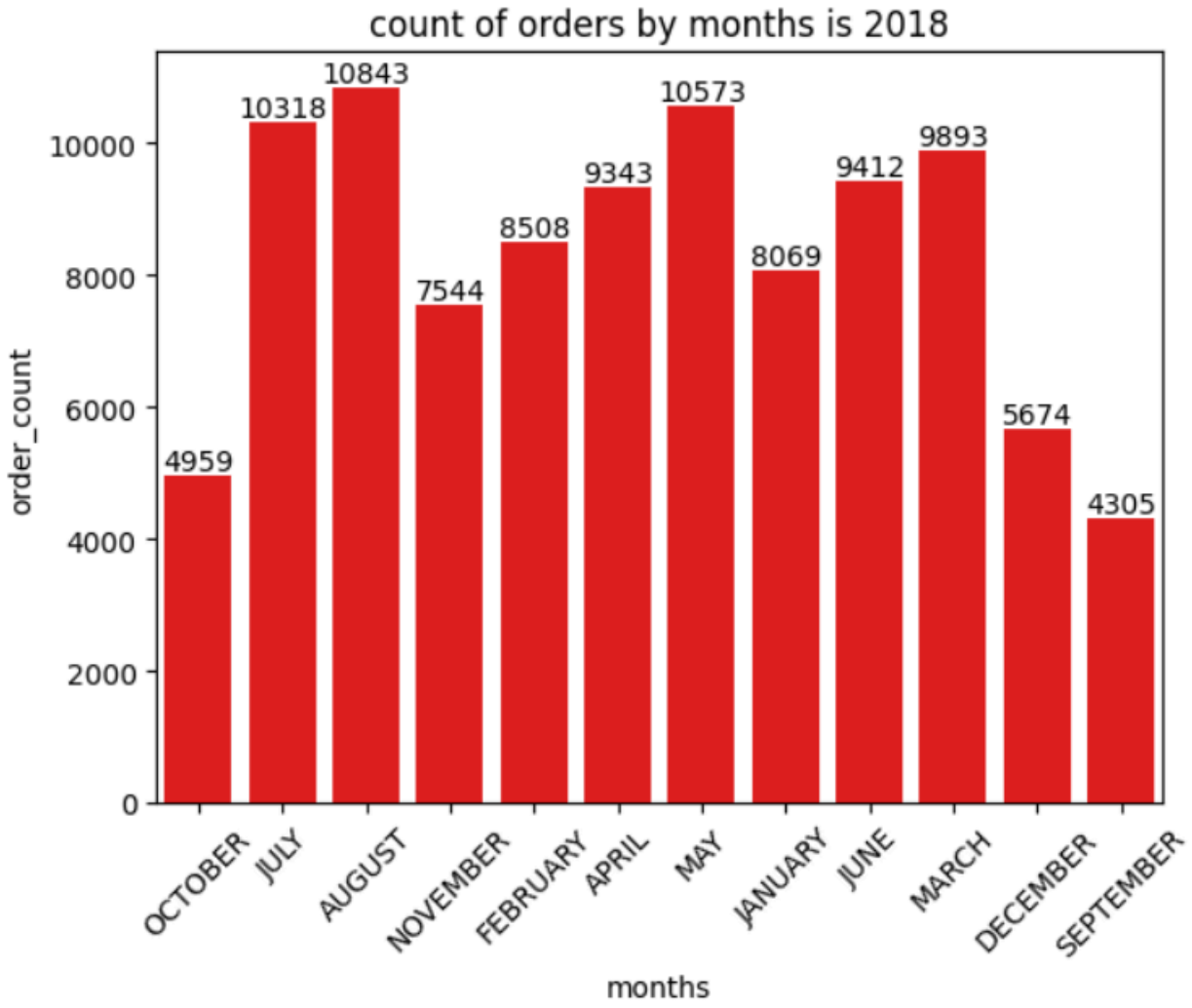


##Calculate the number of orders per month in 2018.

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

cur.execute(query)

data = cur.fetchall()
df = pd.DataFrame(data, columns = ["months", "order_count"])
ax =sns.barplot(x = df["months"], y= df["order_count"], data = df, color = "red")
plt.xticks(rotation = 45)
ax.bar_label(ax.containers[0])
plt.title("count of orders by months is 2018")
plt.show()
```



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

```
query = """with order_per_count 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 upper(customers.customer_city) as customer_city, round(avg(order_per_count.oc),2) as average_ordered
from customers join order_per_count on customers.customer_id = order_per_count.customer_id
group by customer_city"""

cur.execute(query)

data = cur.fetchall()
df = pd.DataFrame(data, columns = ["customer_city", "average_ordered_product"])
df
```

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

```
query = """select  upper(products.product_category) as category,
round((sum(payments.payment_value)/(select sum(payment_value) from payments))*100,2) as sales_precentage
from e_commerce.products join e_commerce.order_items on products.product_id = order_items.product_id
join e_commerce.payments on payments.order_id = order_items.order_id
group by category order by  sales_precentage desc;"""

cur.execute(query)

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

	customer_city	average_ordered_product
0	SAO PAULO	1.16
1	SAO JOSE DOS CAMPOS	1.14
2	PORTO ALEGRE	1.17
3	INDAIAL	1.12
4	TREZE TILIAS	1.27
...	...	...
4105	JAPARATUBA	1.00
4106	SEBASTIAO LEAL	1.00
4107	BURITI	3.00
4108	MORRO AGUDO DE GOIAS	1.00
4109	PADRE PARAISO	1.00

4110 rows × 2 columns

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

```
query = """select  upper(products.product_category) as category,
round((sum(payments.payment_value)/(select sum(payment_value) from payments))*100,2) as sales_precentage
from e_commerce.products join e_commerce.order_items on products.product_id = order_items.product_id
join e_commerce.payments on payments.order_id = order_items.order_id
group by category order by  sales_precentage desc;"""

cur.execute(query)

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

:

	category	sales_precentage
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

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

```
query = """select upper(products.product_category),
count(order_items.product_id),
round(avg(order_items.price),2)
from e_commerce.products join e_commerce.order_items
on products.product_id = order_items.product_id
group by products.product_category;"""



cur.execute(query)

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

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

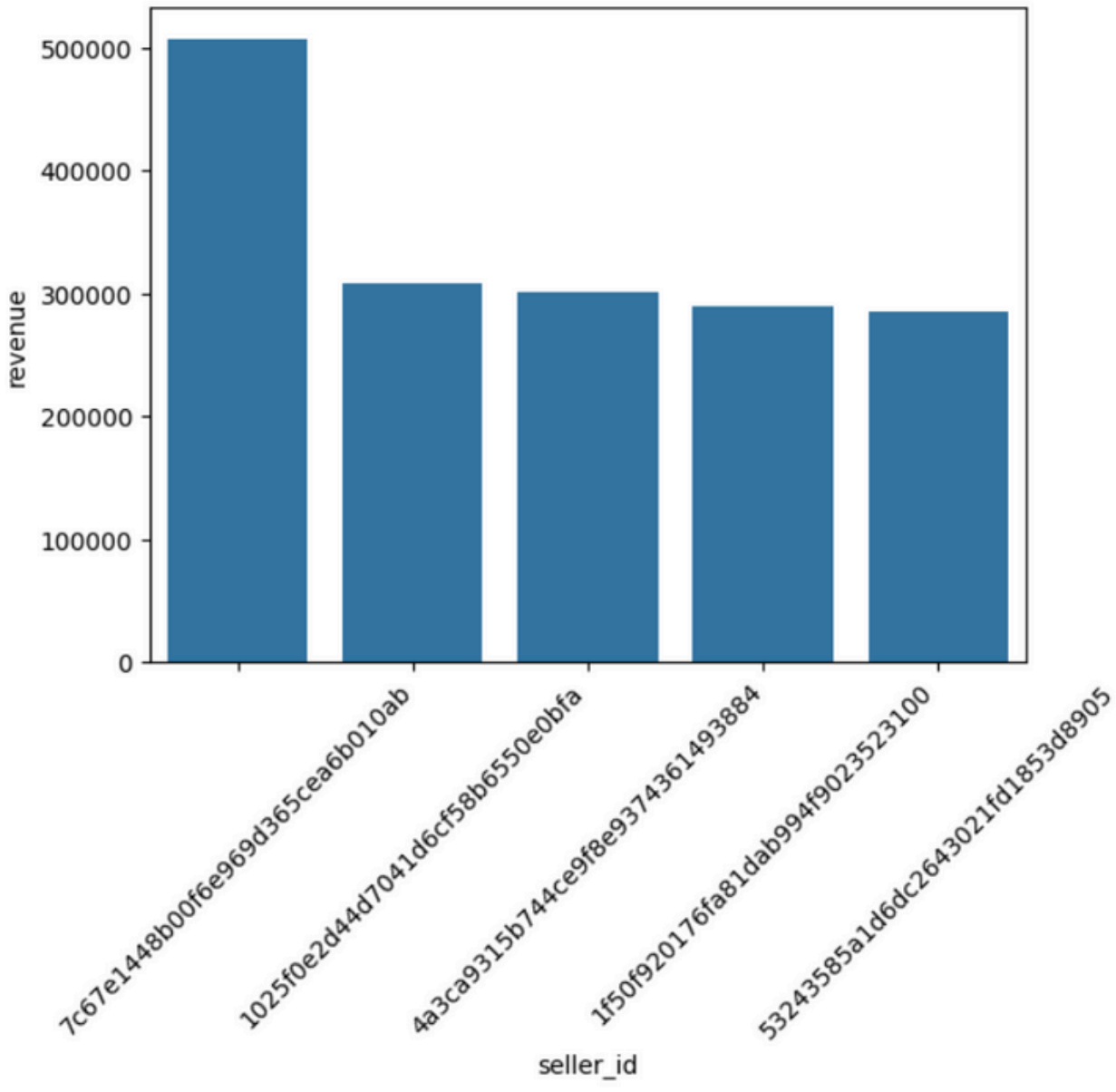
```
The correlation is -0.10631514167157562
```

##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) as revenue
from e_commerce.order_items join e_commerce.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()
sns.barplot(x = "seller_id", y = "revenue", data = df)
plt.xticks(rotation = 90)
plt.show()
```



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

```
query = """select years, months, payment, sum(payment) over(order by years, months) from
(select month(orders.order_purchase_timestamp) as months,
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 months,years order by months, years) as a"""


cur.execute(query)

data = cur.fetchall()
df = pd .DataFrame(data)
df.head(11)
```

]:

	years	months	payment	total_payment
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
8	2017	6	511276.38	2461605.22

##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