In [ ]:
```python
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 = "#Akm5408",
                            database = "ecommerce")

cur = db.cursor()
```

List all unique cities where customers are located.

In [ ]:
```python
query = """ select distinct customer_city from customers """

cur.execute(query)

data = cur.fetchall()

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

Out[ ]:

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

Count the number of orders placed in 2017.

In [ ]:
```python
query = """ select count(order_id) from orders where year(order_purchase_timestamp)

cur.execute(query)

data = cur.fetchall()

'total orders placed in 2017 were', data[0][0]
```

Out[ ]:
```
('total orders placed in 2017 were', 45101)
```

Find the total sales per category.

In [ ]:
```
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
```

Out[ ]:

	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

Calculate the percentage of orders that were paid in more than one installments.

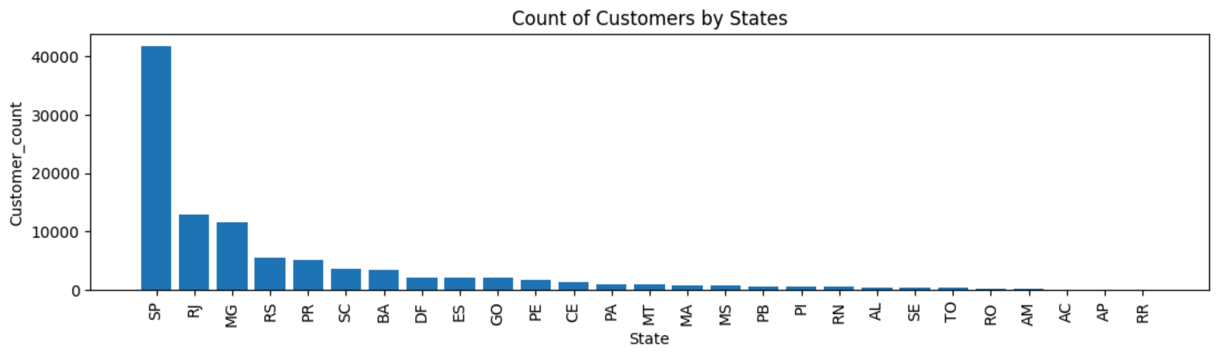
In [ ]:
```
query = """ select ((sum(case when payment_installments > 1 then 1
else 0 end))/count(*))*100 from payments
"""

cur.execute(query)
```

```
data = cur.fetchall()

"the percentage of orders that were paid in installments is", data[0]
```

```
Out[ ]: ('the percentage of orders that were paid in installments is',
         (Decimal('49.4176'),))
```

Count the number of customers from each state.

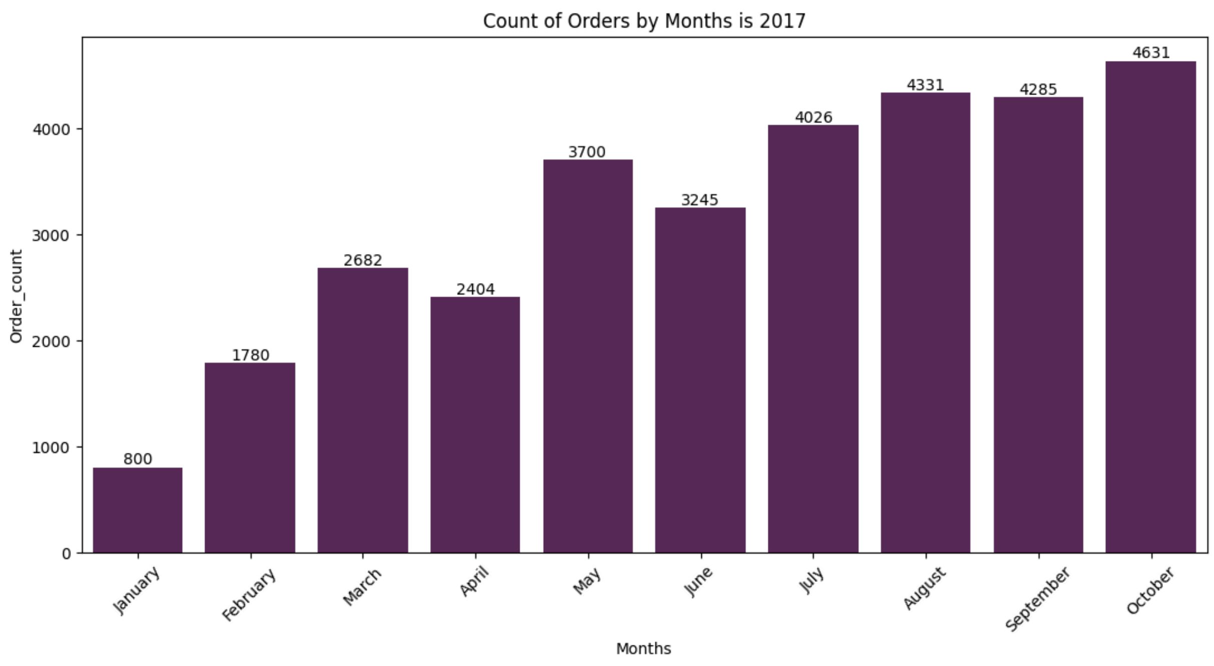
```
In [ ]: query = """ select distinct customer_state ,count(customer_id)
        from customers group by customer_state
        """

        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 = (13,3))
        plt.bar(df["state"], df["customer_count"])
        plt.xticks(rotation = 90)
        plt.xlabel("State")
        plt.ylabel("Customer_count")
        plt.title("Count of Customers by States")
        plt.show()
```



Calculate the number of orders per month in 2017.

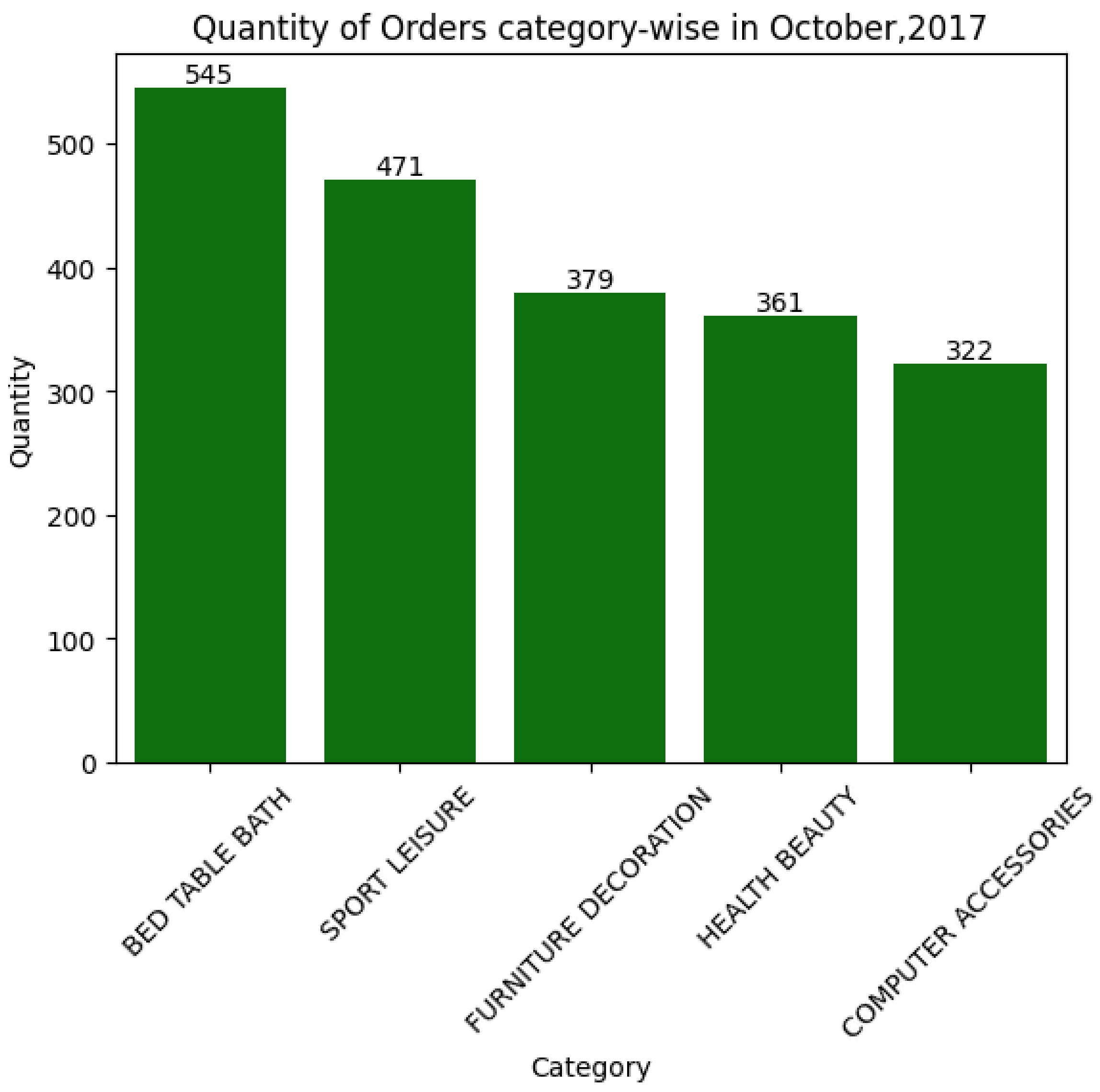
```
In [ ]: query = """ select monthname(order_purchase_timestamp) months, count(order_id) orde
        from orders where year(order_purchase_timestamp) = 2017
        group by months
        """

        cur.execute(query)
```

```
data = cur.fetchall()
df = pd.DataFrame(data, columns = ["Months", "Order_count"])
o = ["January", "February","March","April","May","June","July","August","September"
plt.figure(figsize=(13, 6))
ax = sns.barplot(x = df["Months"],y =  df["Order_count"], data = df, order = o, col

plt.xticks(rotation = 45)
ax.bar_label(ax.containers[0])
plt.title("Count of Orders by Months is 2017")

plt.show()
```



Calculate category wise orders in the month of February in 2018

In [ ]:
```
query="""select upper(products.product_category) category, count(orders.order_id) q
from products join order_items on products.product_id = order_items.product_id join
"""
cur.execute(query)

data = cur.fetchall()

df = pd.DataFrame(data, columns = ["Category", "Quantity"])

o = df['Category']

ax = sns.barplot(x = df["Category"],y =  df["Quantity"], data = df, order = o , col
plt.xticks(rotation = 45)
ax.bar_label(ax.containers[0])
plt.title("Quantity of Orders category-wise in October,2017")

plt.show()
```



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

```
In [ ]: 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
        """

        cur.execute(query)

        data = cur.fetchall()
```

```
df = pd.DataFrame(data,columns = ["Customer city", "Average products per order"])
df.head(10)
```

Out[ ]:

	Customer city	Average products per 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
7	morro de sao paulo	4.00
8	teixeira soares	4.00
9	curralinho	4.00

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

In [ ]:

```
query = """select upper(products.product_category) category,
round((sum(payments.payment_value)/(select sum(payment_value) from payments))*100,2
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_percentage desc"""


cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data,columns = ["Category", "Percentage distribution"])
df.head()
```

Out[ ]:

	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

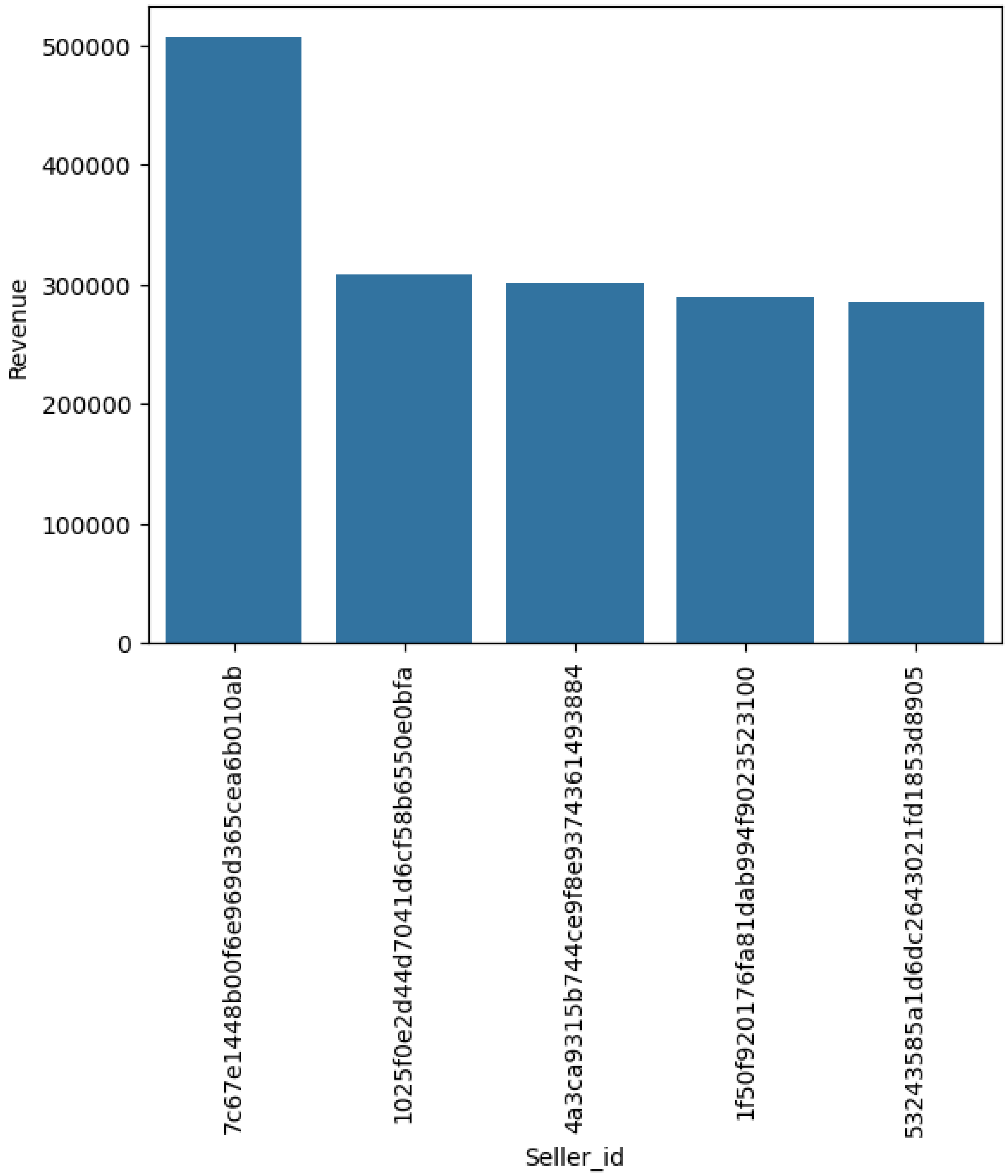
Identify the correlation between product price and the number of times a product has been purchased.

```
In [ ]: cur = db.cursor()
        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"""

        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.

```
In [ ]: 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()
        sns.barplot(x = "Seller_id", y = "Revenue", data = df)
        plt.xticks(rotation = 90)
        plt.show()
```



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

```
In [ ]: 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)
df
```

Out[ ]:

	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	71.23	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

Calculate the cumulative sales per month for each year.

In [ ]:

```
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)
df
```

Out[ ]:

	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
8	2017	6	511276.38	2461605.22
9	2017	7	592382.92	3053988.14
10	2017	8	674396.32	3728384.46
11	2017	9	727762.45	4456146.91
12	2017	10	779677.88	5235824.79
13	2017	11	1194882.80	6430707.59
14	2017	12	878401.48	7309109.07
15	2018	1	1115004.18	8424113.25
16	2018	2	992463.34	9416576.59
17	2018	3	1159652.12	10576228.71
18	2018	4	1160785.48	11737014.19
19	2018	5	1153982.15	12890996.34
20	2018	6	1023880.50	13914876.84
21	2018	7	1066540.75	14981417.59
22	2018	8	1022425.32	16003842.91
23	2018	9	4439.54	16008282.45
24	2018	10	589.67	16008872.12

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

In [ ]:

```
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
```

Out[ ]:

	years	yoy % growth
0	2016	NaN
1	2017	12112.703761
2	2018	20.000924

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

In [ ]:

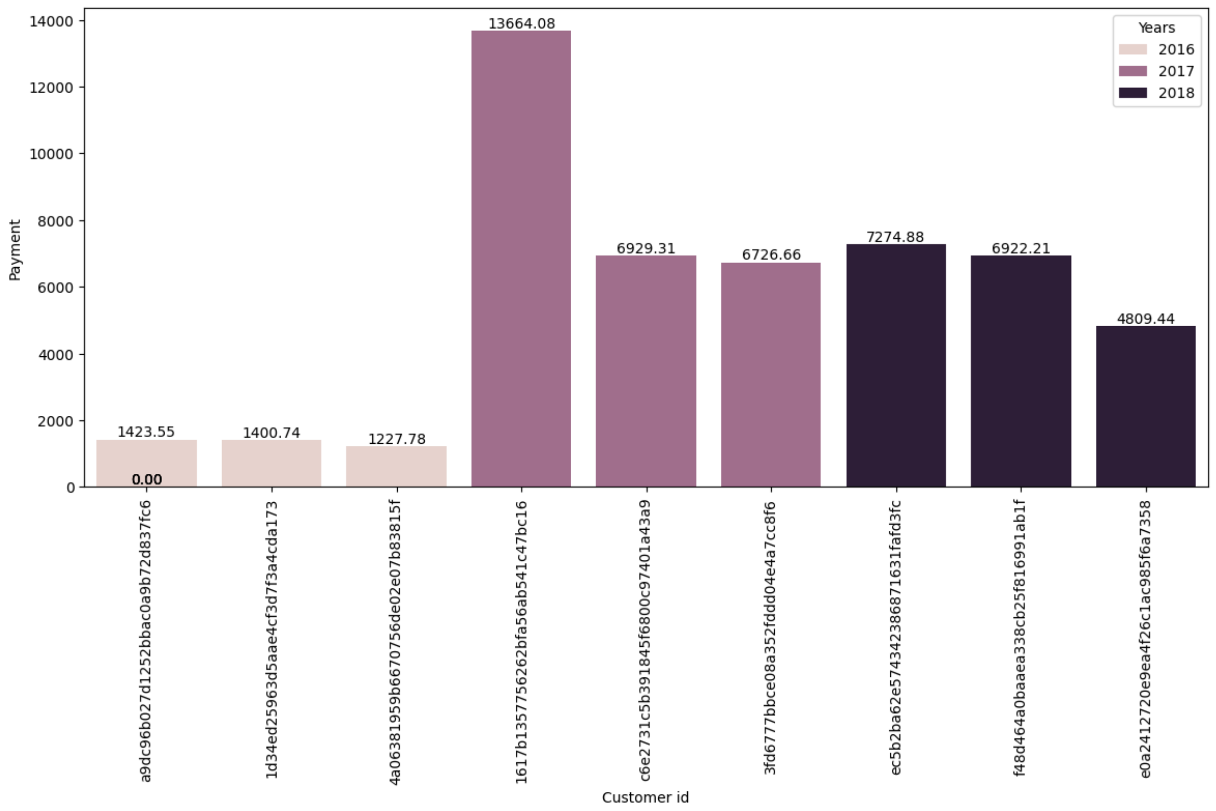
```
query = """with a as (select customers.customer_id,
min(orders.order_purchase_timestamp) first_order
from customers join orders
on customers.customer_id = orders.customer_id
group by customers.customer_id),

b as (select a.customer_id, count(distinct orders.order_purchase_timestamp) next_or
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 ;"""

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

data
```

Out[ ]:

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

Identify the top 3 customers who spent the most money in each year.

```
In [ ]: query = """select years, customer_id, payment, d_rank
from
(select year(orders.order_purchase_timestamp) years,
orders.customer_id,
round(sum(payments.payment_value),2) 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","Customer id","Payment","Rank"])
plt.figure(figsize=(14, 6))
ax = sns.barplot(x = "Customer id", y = "Payment", data = df, hue = "Years")

for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 5),  # 5 points vertical offset
                textcoords='offset points')
plt.xticks(rotation = 90)
plt.show()
```


Years
2016
2017
2018
Payment
14000
12000
10000
8000
6000
4000
2000
0
13664.08
6929.31
6726.66
7274.88
6922.21
4809.44
1423.55
1400.74
1227.78
0.00
a9dc96b027d1252bbac0a9b72d837fc6
1d34ed25963d5aae4cf3d7f3a4cda173
4a06381959b6670756de02e07b83815f
1617b1357756262bfa56ab541c47bc16
c6e2731c5b391845f6800c97401a43a9
3fd6777bbce08a352fddd04e4a7cc8f6
ec5b2ba62e574342386871631fafd3fc
f48d464a0baaea338cb25f816991ab1f
e0a2412720e9ea4f26c1ac985f6a7358
Customer id

```
In [ ]:
```