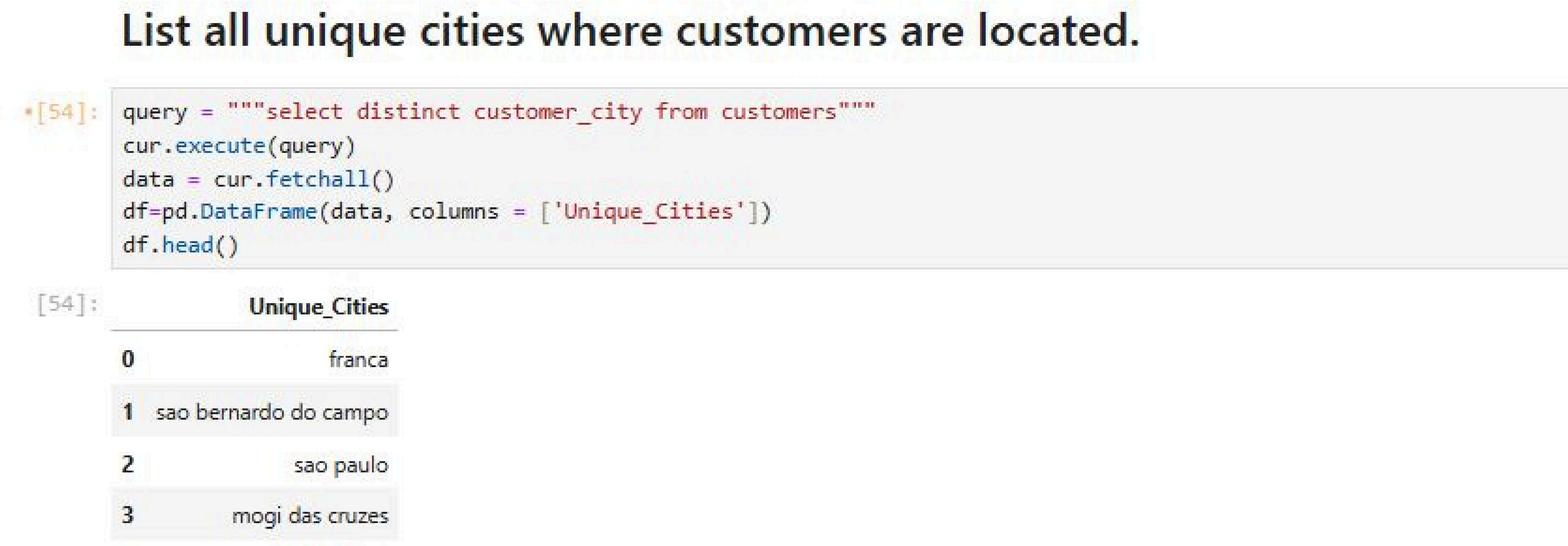
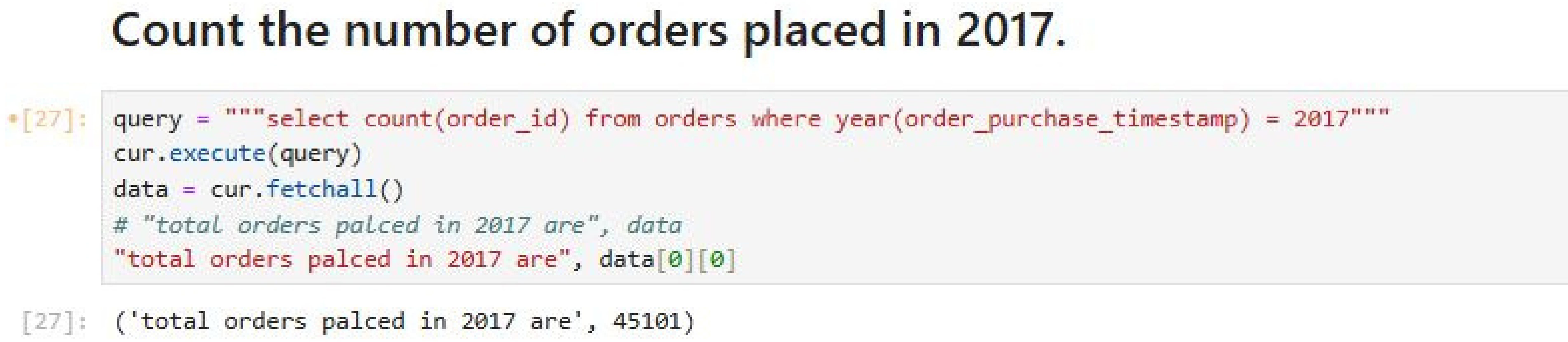


List all unique cities where customers are located.

```
•[54]:
query = """select distinct customer_city from customers"""
cur.execute(query)
data = cur.fetchall()
df=pd.DataFrame(data, columns = ['Unique_Cities'])
df.head()
```

[54]:

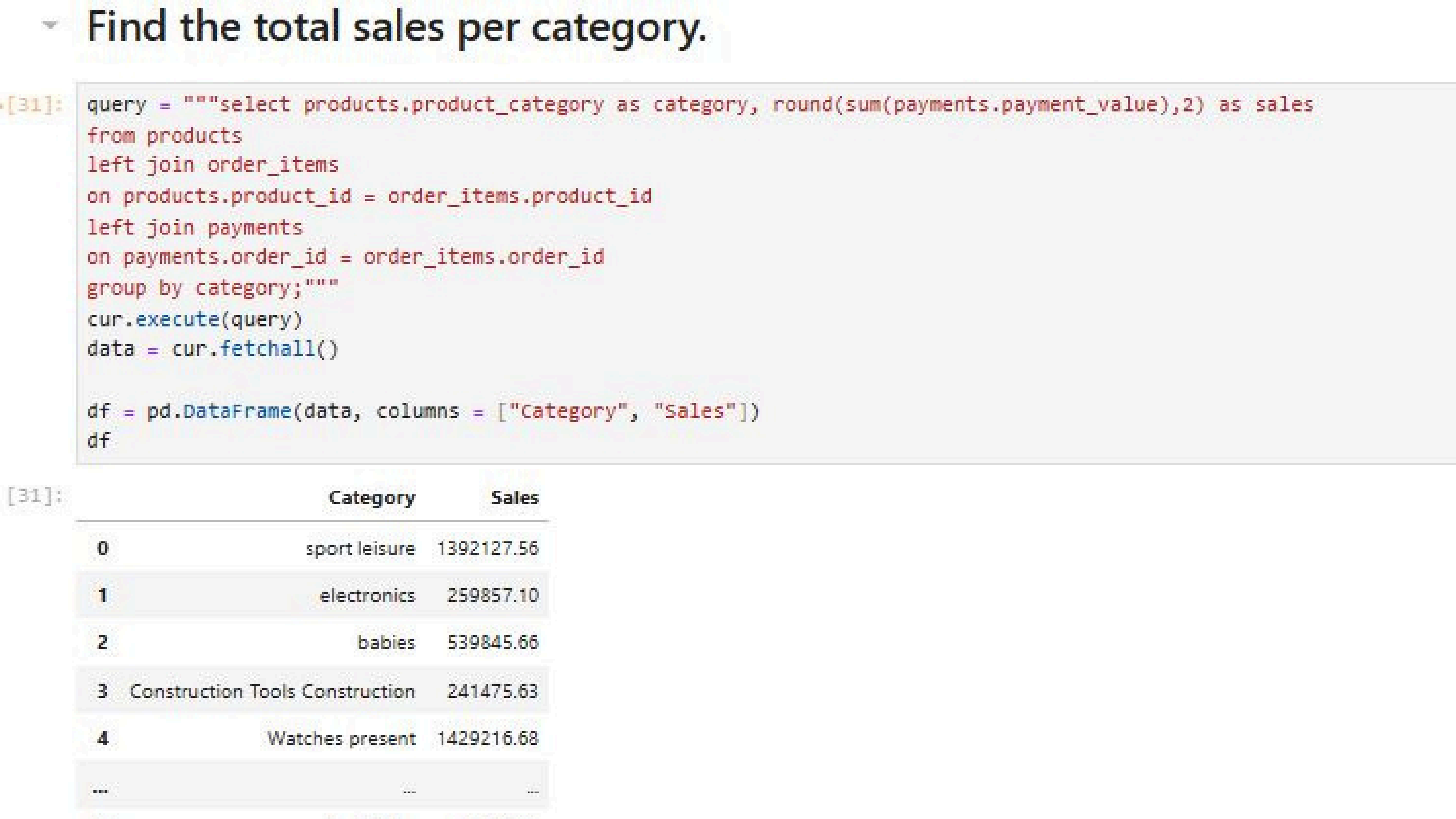
	Unique_Cities
0	franca
1	sao bernardo do campo
2	sao paulo
3	mogi das cruzes



Count the number of orders placed in 2017.

```
•[27]: query = """select count(order_id) from orders where year(order_purchase_timestamp) = 2017"""
       cur.execute(query)
       data = cur.fetchall()
       # "total orders palced in 2017 are", data
       "total orders palced in 2017 are", data[0][0]
```

```
[27]: ('total orders palced in 2017 are', 45101)
```



Find the total sales per category.

[31]:
```python
query = """select products.product_category as category, round(sum(payments.payment_value),2) as sales
from products
left join order_items
on products.product_id = order_items.product_id
left 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
```

[31]:
```
                             Category       Sales
0                        sport leisure  1392127.56
1                          electronics   259857.10
2                               babies   539845.66
3    Construction Tools Construction    241475.63
4                      Watches present  1429216.68
...                                ...         ...
```

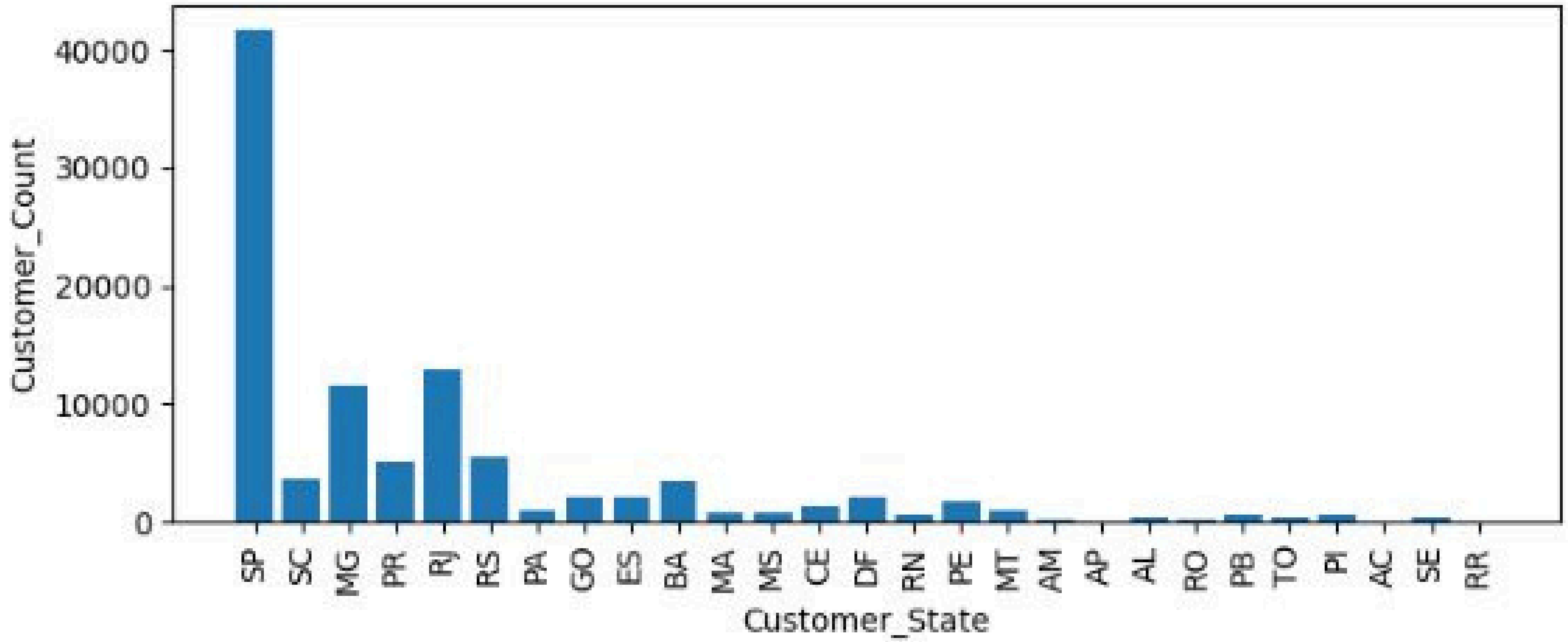
Calculate the percentage of orders that were paid in installments.

```
[36]: 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]
```

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

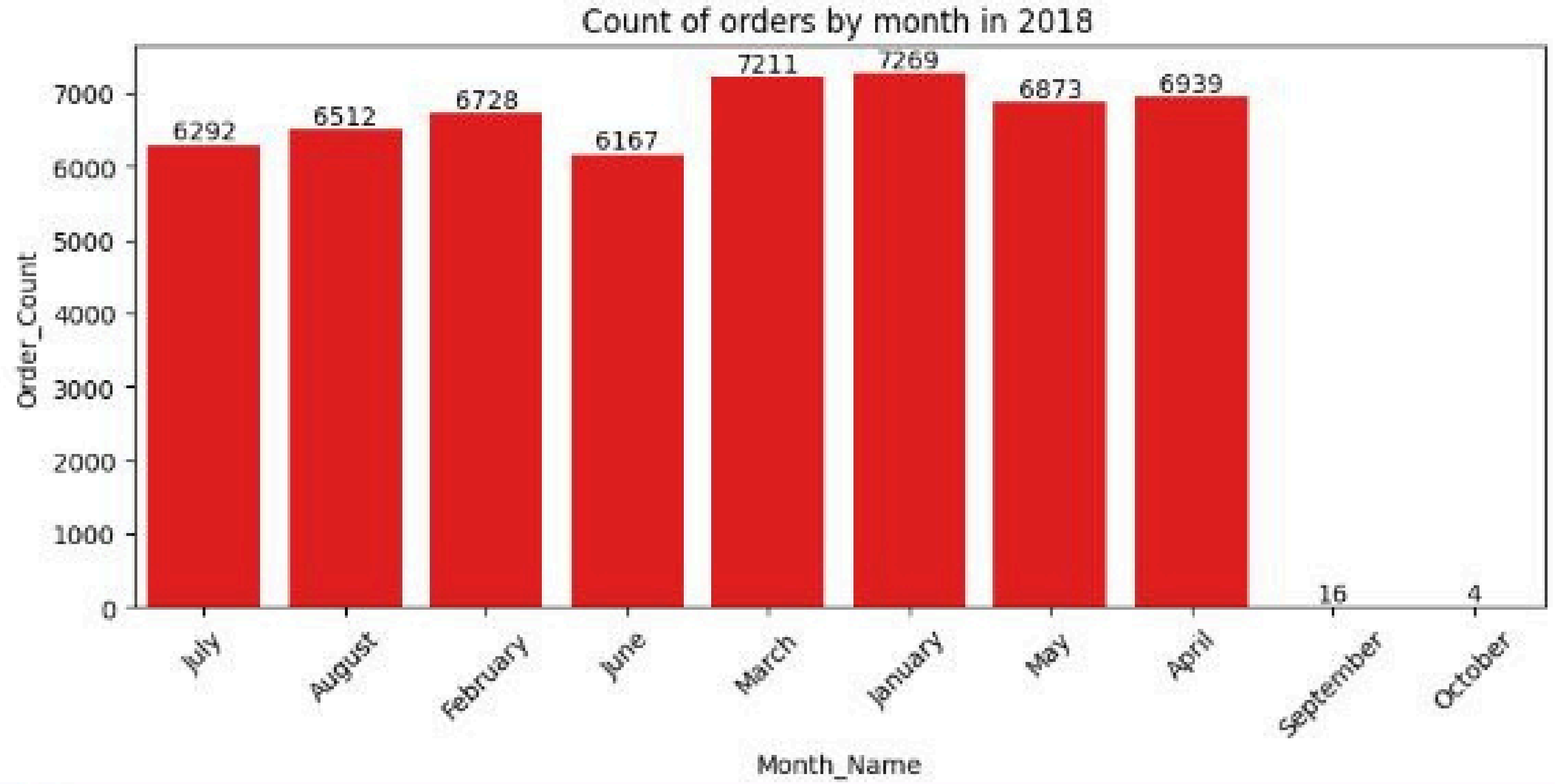
Count the number of customers from each state.

```
[50]: query = """select customer_state, count(customer_state) from customers group by customer_state;"""
cur.execute(query)
data = cur.fetchall()
df=pd.DataFrame(data, columns = ['Customer_State', 'Customer_Count'])
plt.figure(figsize=(8,3))
plt.bar(df['Customer_State'], df['Customer_Count'])
plt.xlabel('Customer_State')
plt.ylabel('Customer_Count')
plt.xticks(rotation=90)
plt.show()
```



Calculate the number of orders per month in 2018.

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

cur.execute(query)
data=cur.fetchall()
df=pd.DataFrame(data, columns = ['Month_Name', 'Order_Count'])

plt.figure(figsize=(10,4))
ax = sns.barplot(x = df['Month_Name'], y = df['Order_Count'], data = df, color = 'red')
plt.xticks(rotation=45)
plt.title('Count of orders by month in 2018')
plt.xlabel('Month_Name')
plt.ylabel('Order_Count')
ax.bar_label(ax.containers[0])
plt.show()
```



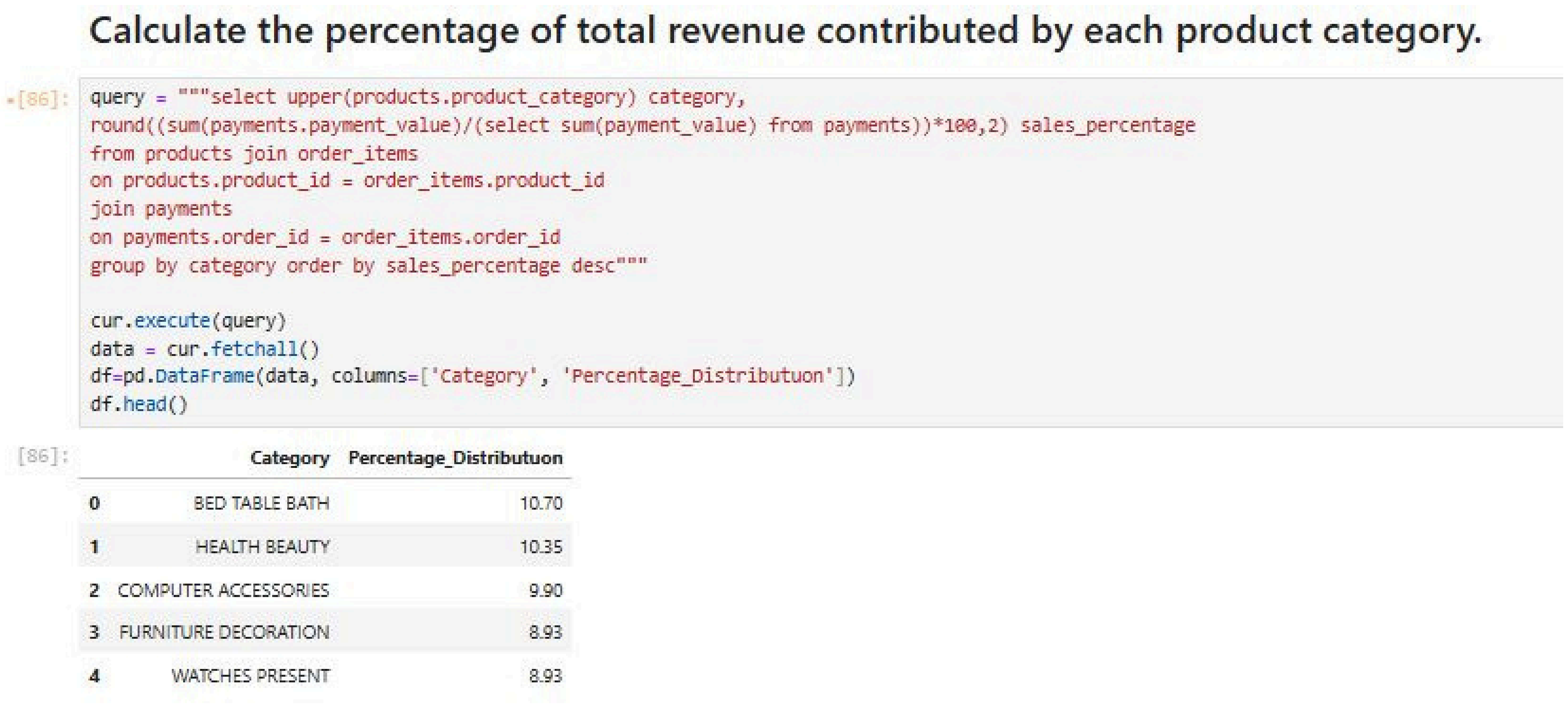
Find the average number of products per order, grouped by customer city.

[82]:

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


cur.execute(query)
data = cur.fetchall()
df=pd.DataFrame(data, columns = ['customer_city', 'average_product_per_order'])
df.head()
```

[82]:

	customer_city	average_product_per_order
0	treze tilias	1.27
1	indaial	1.12
2	sao jose dos campos	1.13
3	sao paulo	1.15
4	porto alegre	1.17

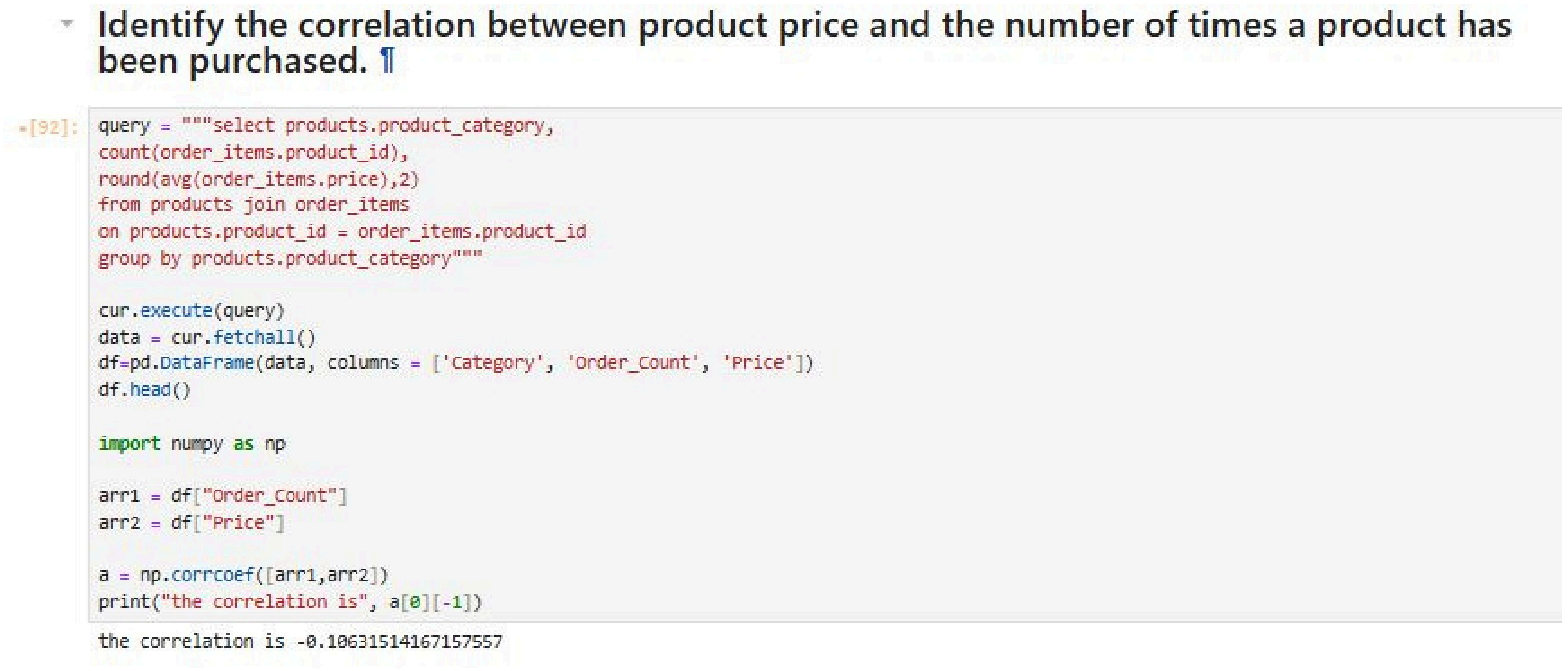


Calculate the percentage of total revenue contributed by each product category.

```
[86]: query = """select upper(products.product_category) category,
      round((sum(payments.payment_value)/(select sum(payment_value) from payments))*100,2) sales_percentage
      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_Distributuon'])
      df.head()
```

[86]:

	Category	Percentage_Distributuon
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. ¶

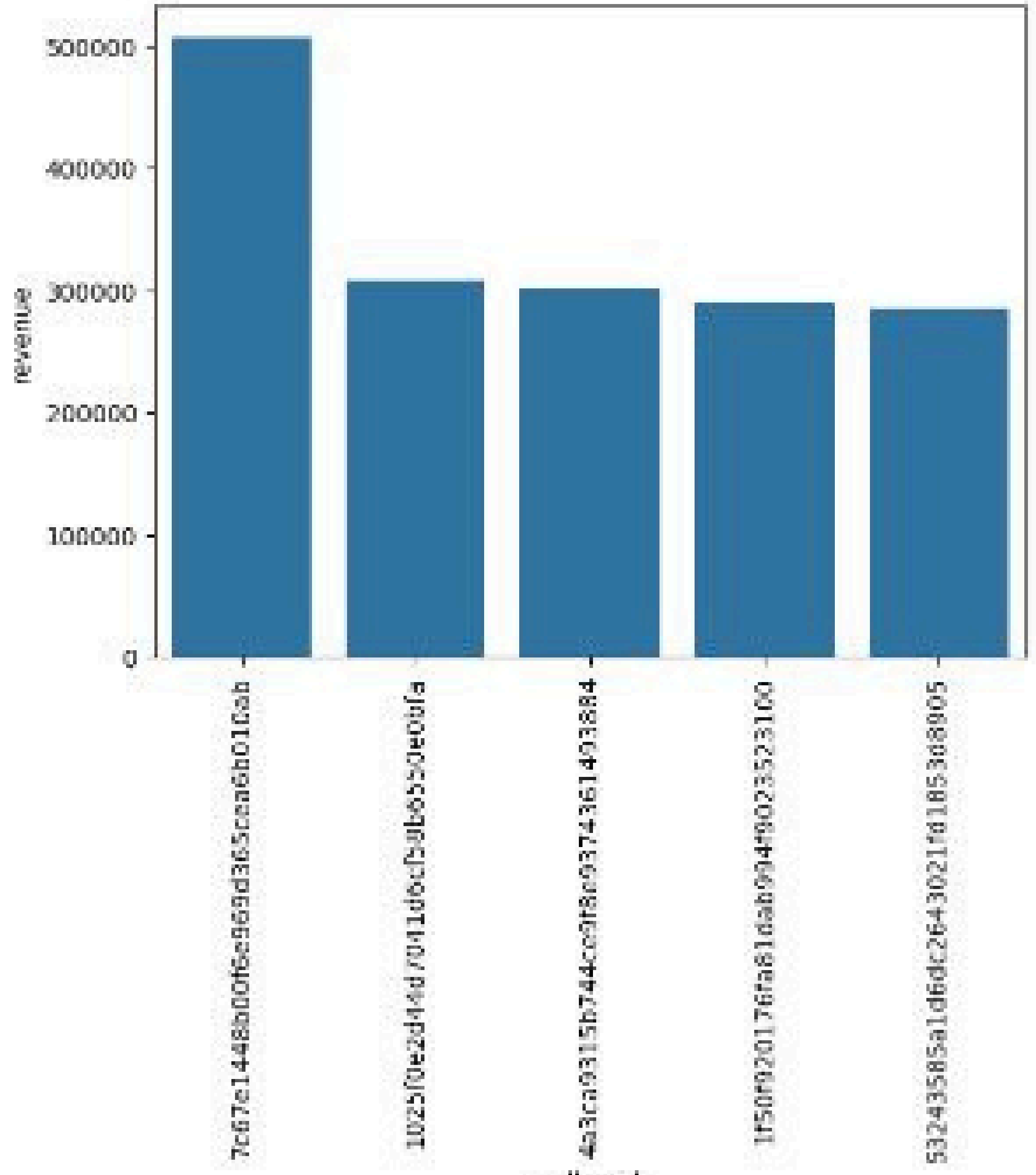
[92]:

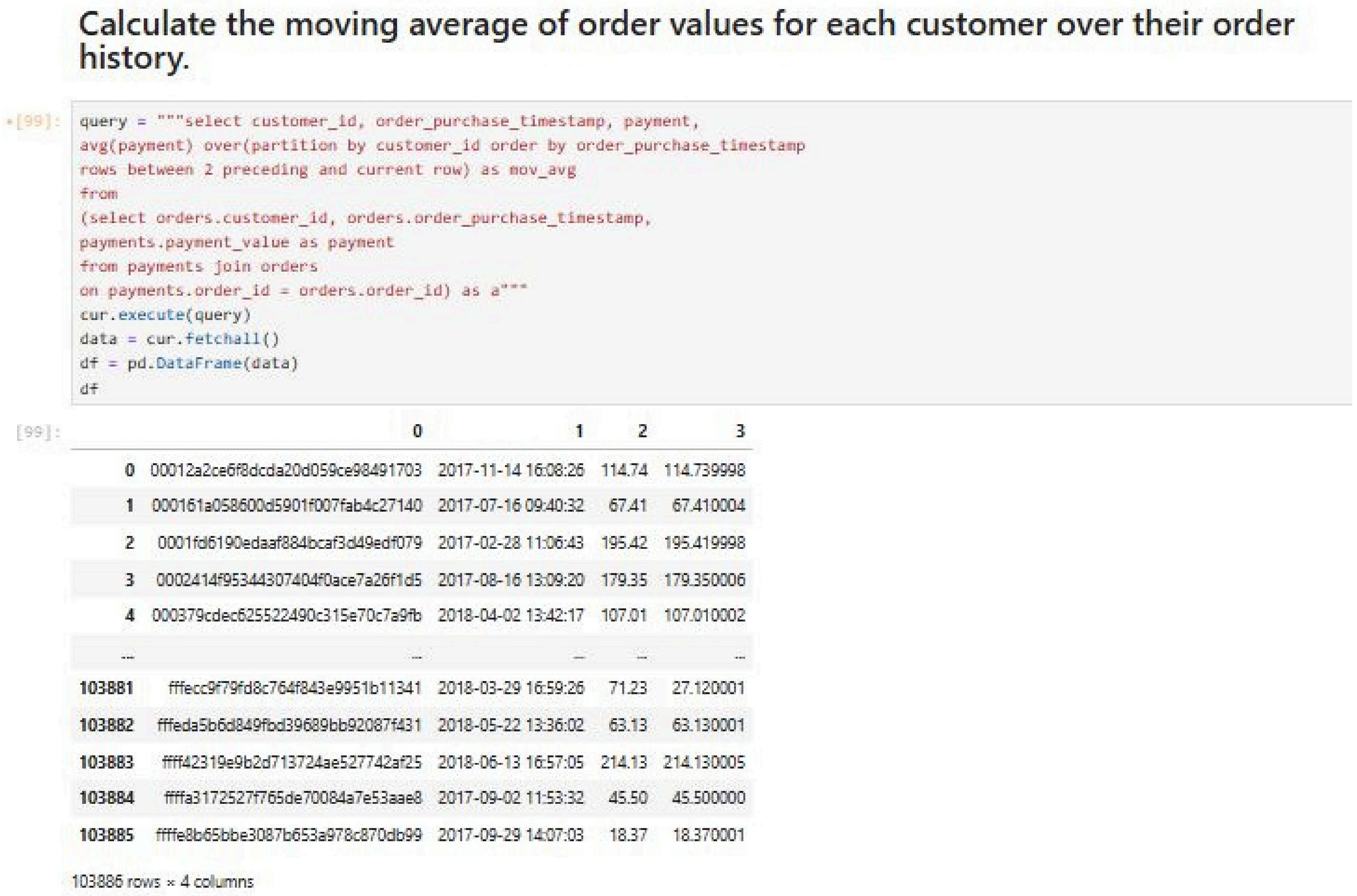
```python
query = """select products.product_category,
count(order_items.product_id),
round(avg(order_items.price),2)
from products join order_items
on products.product_id = order_items.product_id
group by products.product_category"""

cur.execute(query)
data = cur.fetchall()
df=pd.DataFrame(data, columns = ['Category', 'Order_Count', 'Price'])
df.head()

import numpy as np

arr1 = df["Order_Count"]
arr2 = df["Price"]

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

```
the correlation is -0.10631514167157557
```

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()
sns.barplot(x='seller_id', y='revenue', data=df)
plt.xticks(rotation=90)
plt.show()
```





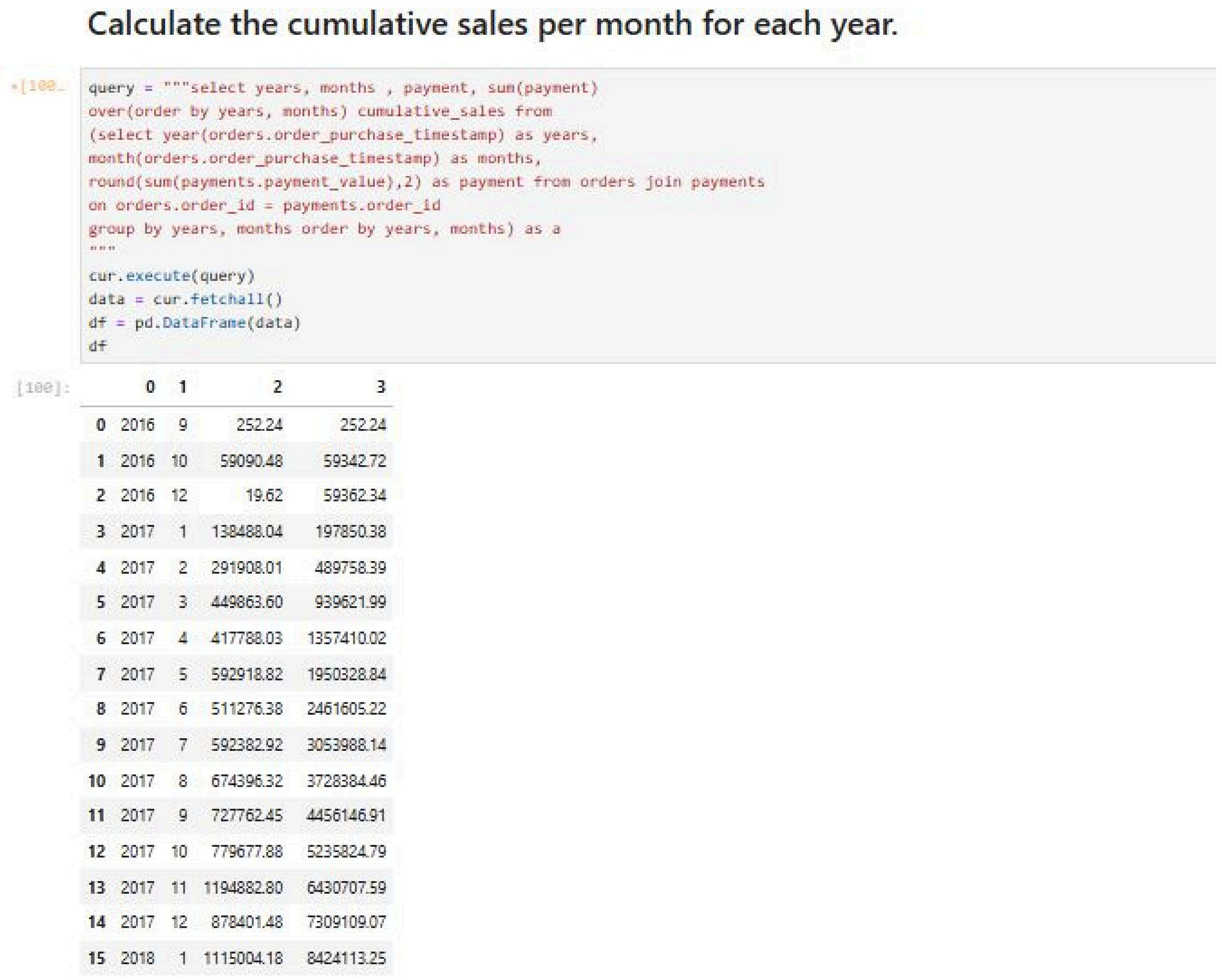
Calculate the moving average of order values for each customer over their order history.

```
•[99]: 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
```

[99]:

	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.

```
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
```

[100]:

	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

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

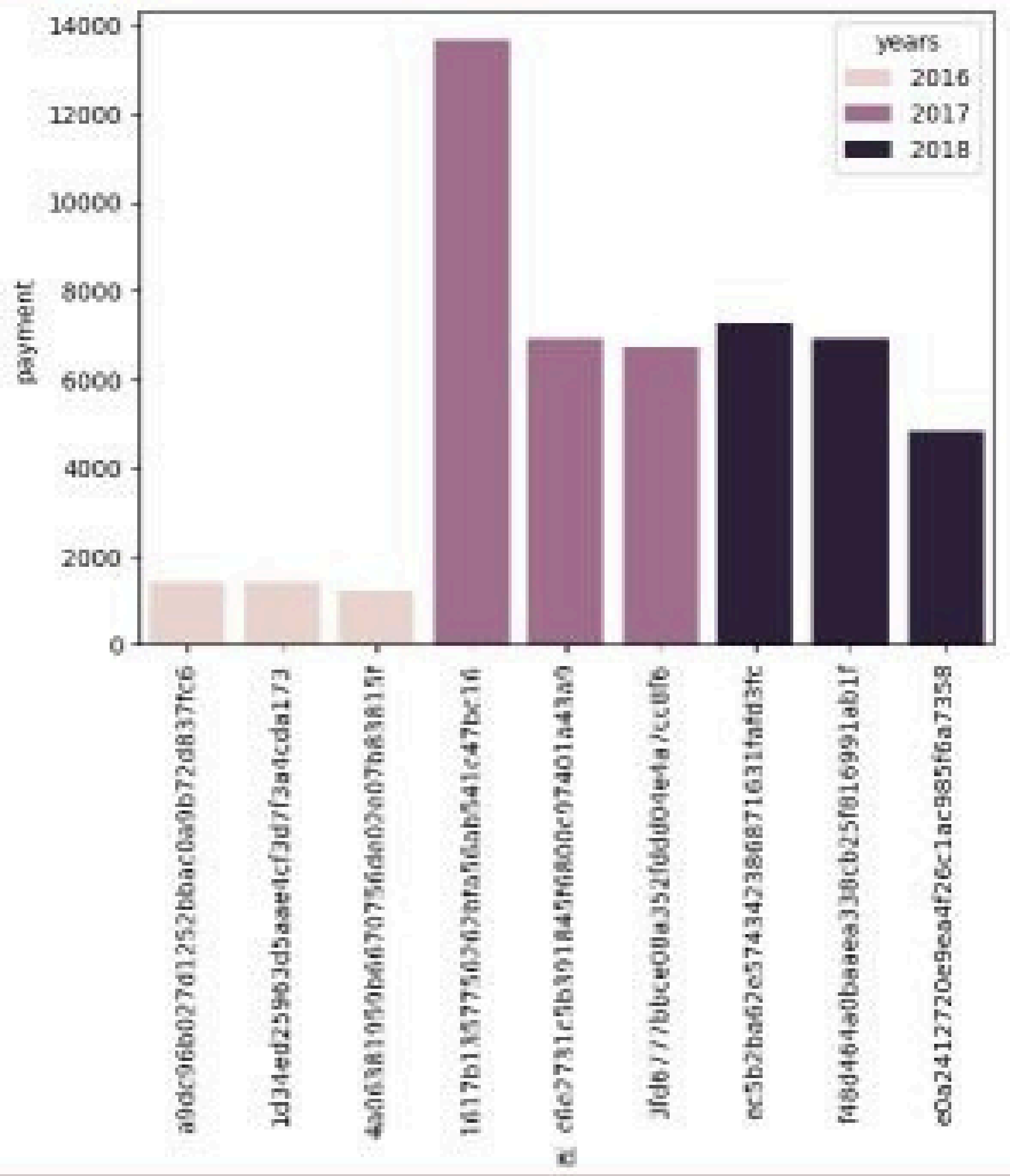
[101]:
```
query = """with a as(select year(orders.order_purchase_timestamp) as years,
round(sum(payments.payment_value),2) as payment from orders join payments
on orders.order_id = payments.order_id
group by years order by years)

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
```

[101]:

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

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

```
[102…
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"])
sns.barplot(x = "id", y = "payment", data = df, hue = "years")
plt.xticks(rotation = 90)
plt.show()
```


years
payment
id
a0dc96b027d12521bac0a9b72d837fc6
1d34ed25963d5aae4cf3d7f3a4cda173
4a06381959b6670756de02e07b83815f
1617b1357756262bfa56ab541c47bc16
c6e2731c5b391845f6800c97401a43a9
3fd6777bbce08a352fddd04e4a7cc8f6
ec5b2ba62e574342386871631fafd3fc
f48d464a0baaea338cb25f816991ab1f
e0a2412720e9ea4f26c1ac985f6a7358