SQL+PYTHON

Ecommerce Project

2024

Ecommerce Project

Target Sales Dataset

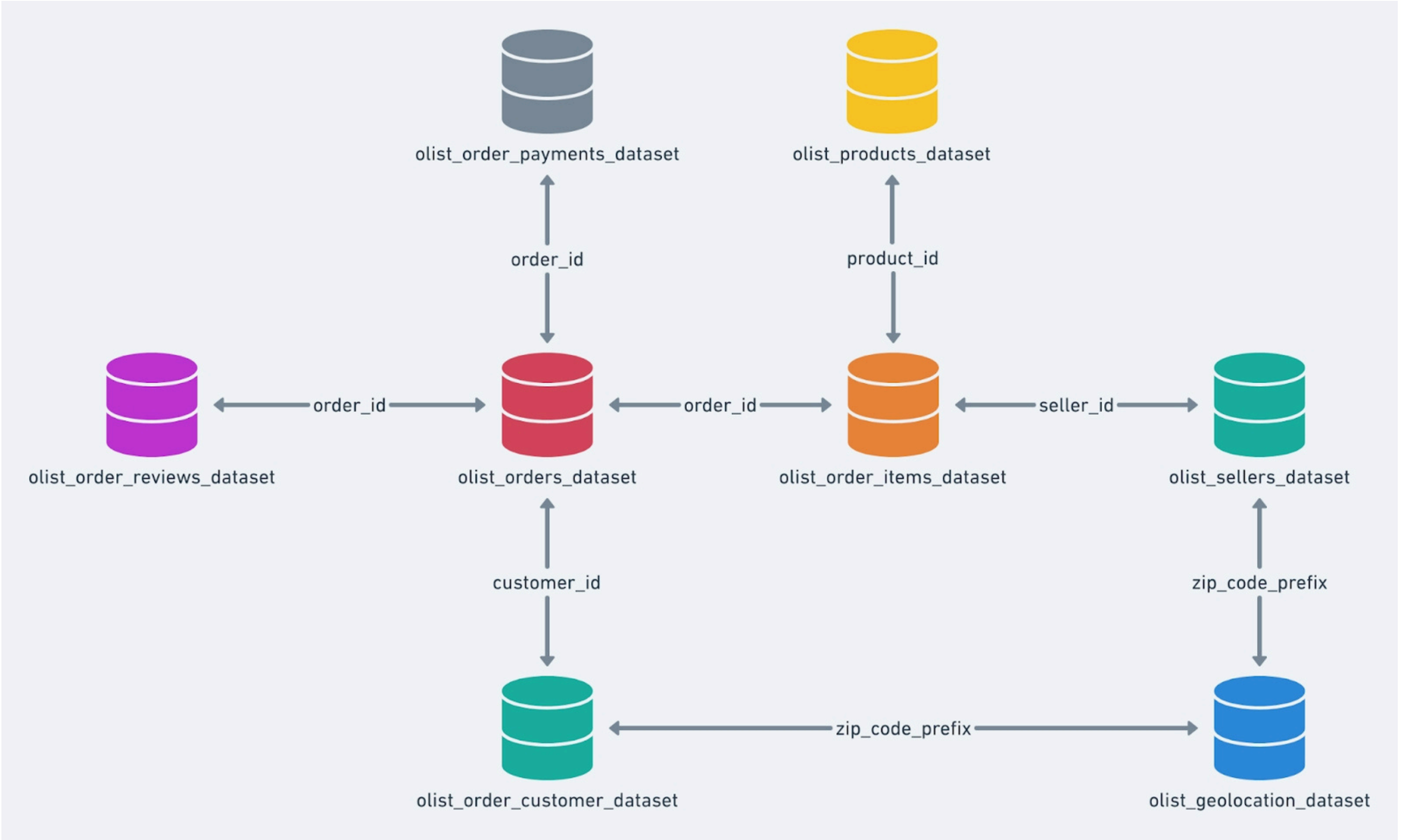
Target is a globally recognized brand and a leading retailer in the United States, known for offering exceptional value, inspiration, innovation, and a unique shopping experience.

This dataset focuses on Target's operations in Brazil, covering 100,000 orders placed between 2016 and 2018. It includes detailed information on order status, pricing, payment and shipping performance, customer locations, product attributes, and customer reviews.

Features:-
The data is available in 8 csv files:

- customers.csv
- sellers.csv
- order_items.csv
- geolocation.csv
- payments.csv
- orders.csv
- product.csv

Dataset Schema



In [12]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import mysql.connector

db=mysql.connector.connect(host = "localhost",
                           username = "root",
                           password = "212",
                           database = "sql_ecommerce")
cur = db.cursor()
```

List all unique cities where customers are located.

In [2]:
```
query = """ select distinct customer_city from customers """

cur.execute(query)

data = cur.fetchall()

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

Out[2]:

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

Count the number of orders placed in 2017.

[4]:
```
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]
```

t[4]:
```
('total orders placed in 2017 are', 45101)
```

Find the total sales per category.

[10]:
```
query = """ select upper(products.product_category) category,
round(sum(payments.payment_value),2) sales
from products join order_item
on products.product_id = order_item.product_id
join payments
on payments.order_id = order_item.order_id
group by category
"""

cur.execute(query)

data = cur.fetchall()

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

0	PERFUMERY	506738.66
1	FURNITURE DECORATION	1430176.39
2	TELEPHONY	486882.05
3	FASHION BAGS AND ACCESSORIES	218158.28
4	BED TABLE BATH	1712553.67
...	...	...
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

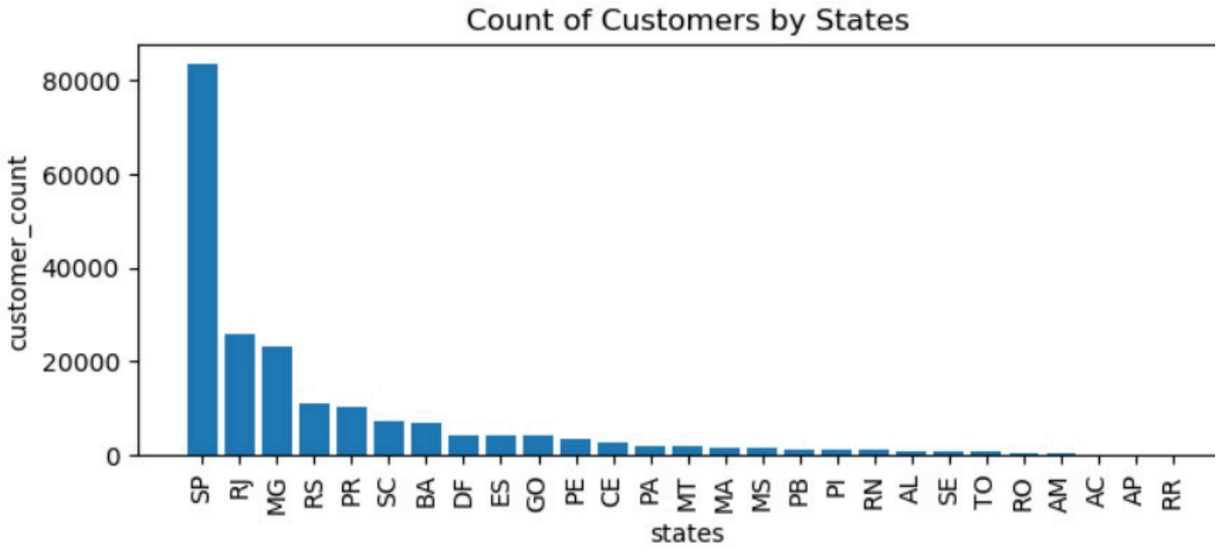
Calculate the percentage of orders that were paid in installments.

In [6]:
```
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]
```

Out[6]:
```
('the percentage of orders that were paid in installments is',
 Decimal('99.9981'))
```

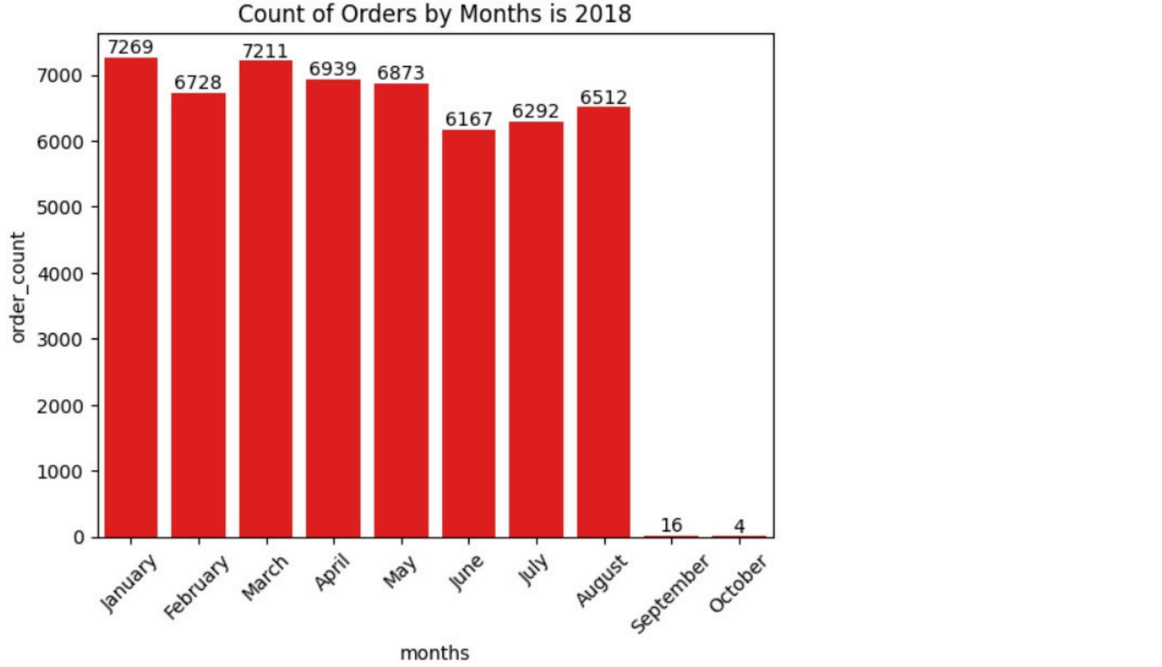
Count the number of customers from each state.

In [7]:
```
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.

In [7]:
```
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"]

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()
```



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

In [11]:
```
query = """with count_per_order as
(select orders.order_id, orders.customer_id, count(order_item.order_id) as oc
from orders join order_item
on orders.order_id = order_item.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/order"])
df.head(10)
```

Out[11]:

	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	curralinho	4.00
7	picarra	4.00

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

[12]:
```
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_item
on products.product_id = order_item.product_id
join payments
on payments.order_id = order_item.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()
```

[12]:

	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

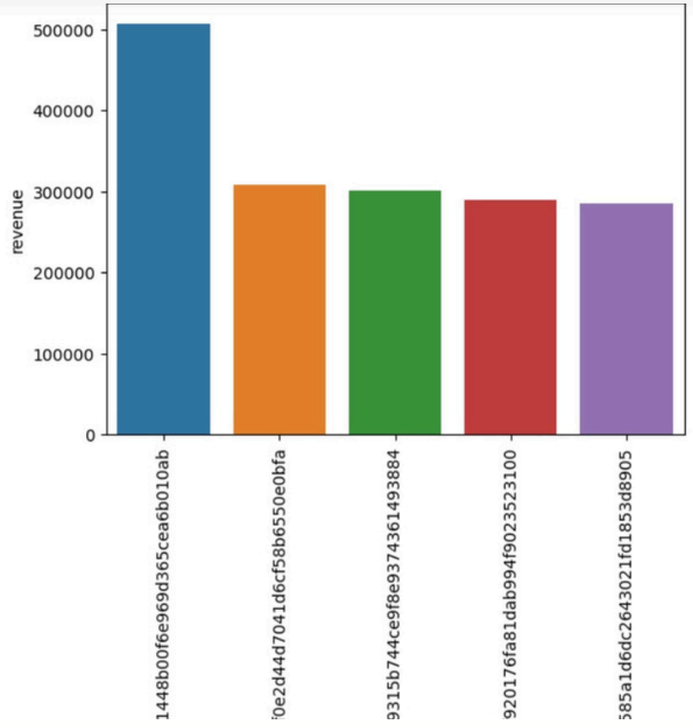
Identify the correlation between product price and the number of times a product has been purchased.

In [8]:
```
import numpy as np
cur = db.cursor()
query = """select products.product_category,
count(order_item.product_id),
round(avg(order_item.price),2)
from products join order_item
on products.product_id = order_item.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 [15]:
```
query = """ select *, dense_rank() over(order by revenue desc) as rn from
(select order_item.seller_id, sum(payments.payment_value)
revenue from order_item join payments
on order_item.order_id = payments.order_id
group by order_item.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 [9]:
```
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[9]:

	0	1	2	3
0	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.74	114.739998
1	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.74	114.739998
2	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.74	114.739998
3	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.74	114.739998
4	000161a058600d5901f007fab4c27140	2017-07-16 09:40:32	67.41	67.410004
...	...	...	...	...
415539	ffffa3172527f765de70084a7e53aae8	2017-09-02 11:53:32	45.50	45.500000
415540	fffffe8b65bbe3087b653a978c870db99	2017-09-29 14:07:03	18.37	18.370001
415541	fffffe8b65bbe3087b653a978c870db99	2017-09-29 14:07:03	18.37	18.370001
415542	fffffe8b65bbe3087b653a978c870db99	2017-09-29 14:07:03	18.37	18.370001

Calculate the cumulative sales per month for each year.

In [10]:
```
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[10]:

	0	1	2	3
0	2016	9	1008.96	1008.96
1	2016	10	236361.92	237370.88
2	2016	12	78.48	237449.36
3	2017	1	553952.16	791401.52
4	2017	2	1167632.04	1959033.56
5	2017	3	1799454.40	3758487.96
6	2017	4	1671152.12	5429640.08
7	2017	5	2371675.28	7801315.36
8	2017	6	2045105.52	9846420.88
9	2017	7	2369531.68	12215952.56
10	2017	8	2697585.28	14913537.84

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

In [18]:
```
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[18]:

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

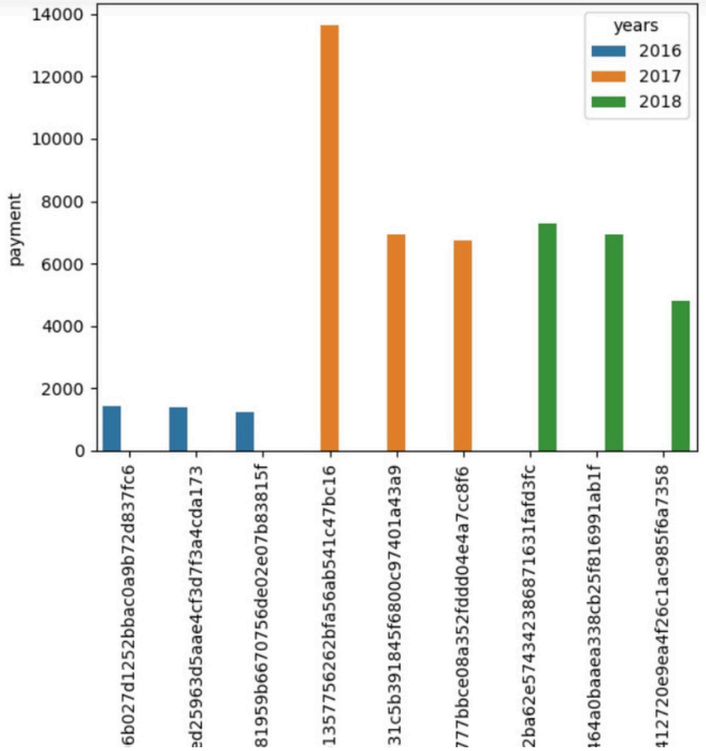
Calculate the retention rate of customers, defined as the percentage of customers who make another purchase within 6 months of their first purchase.

[20]:
```
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 ;"""

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

data
```

[20]:
```
[(None,)]
```

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

In [21]:
```
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()
```



Thank you