
E-COMMERCE PROJECT USING SQL AND PYTHON(END TO END)

PROBLEMS

Tuhin Chanda

Basic Queries

1. List all unique cities where customers are located.
2. Count the number of orders placed in 2017.
3. Find the total sales per category.
4. Calculate the percentage of orders that were paid in installments.
5. Count the number of customers from each state.

Intermediate Queries

1. Calculate the number of orders per month in 2018.
2. Find the average number of products per order, grouped by customer city.
3. Calculate the percentage of total revenue contributed by each product category.
4. Identify the correlation between product price and the number of times a product has been purchased.
5. Calculate the total revenue generated by each seller, and rank them by revenue.

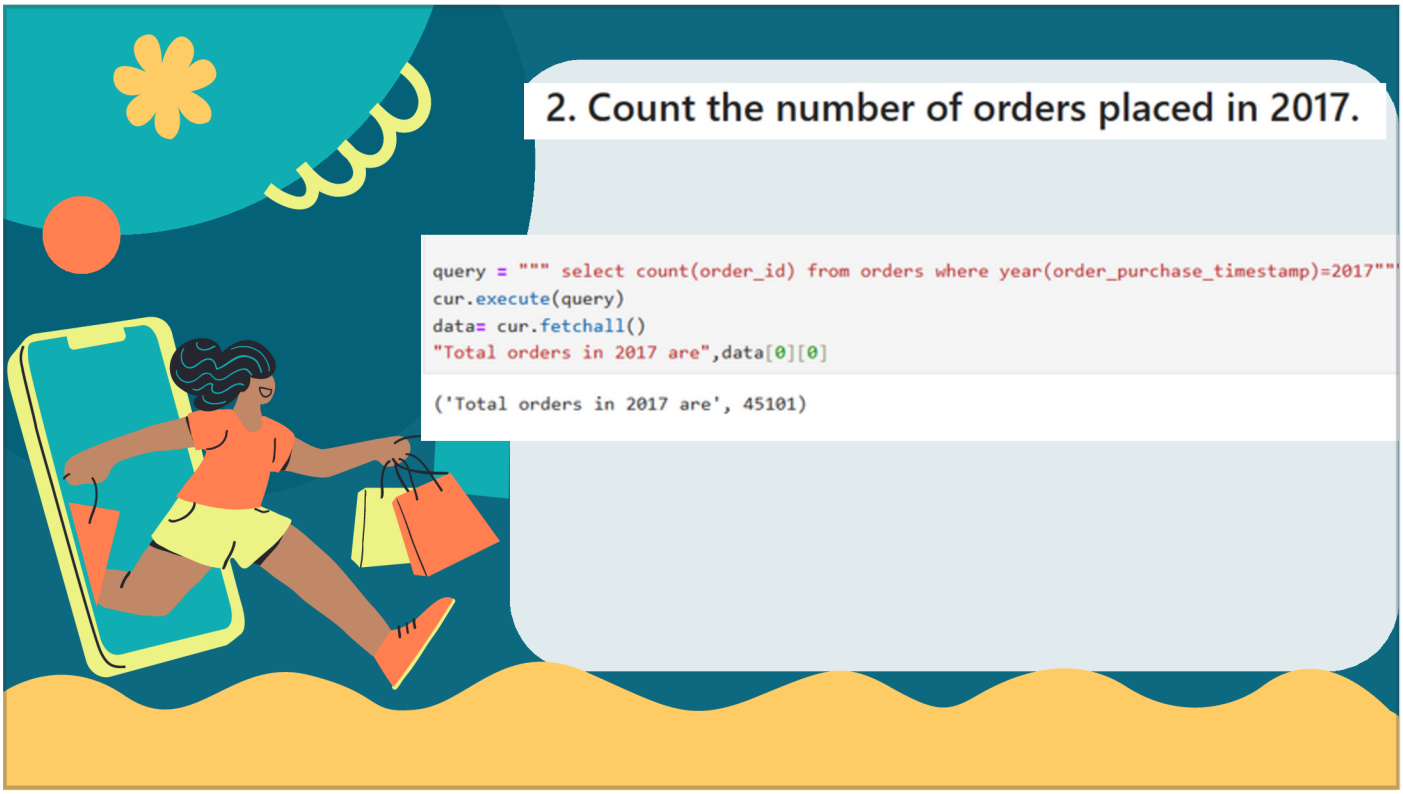
Advanced Queries

1. Calculate the moving average of order values for each customer over their order history.
2. Calculate the cumulative sales per month for each year.
3. Calculate the year-over-year growth rate of total sales.
4. Calculate the retention rate of customers, defined as the percentage of customers who make another purchase within 6 months of their first purchase.
5. Identify the top 3 customers who spent the most money in each year.

1. List all unique cities where customers are located. question of above code

[48]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import mysql.connector

db = mysql.connector.connect (host='localhost',
    user='root',
    password='Tuhin@2002',
    database='ECOMMERCE'
)

cur = db.cursor()
query = """ select distinct customer_city from customers"""
cur.execute(query)
data= cur.fetchall()
data
```

[48]:
```
[('franca',),
 ('sao bernardo do campo',),
 ('sao paulo',),
 ('mogi das cruzes',),
 ('campinas',),
 ('jaragua do sul',),
 ('timoteo',),
 ('curitiba',),
 ('belo horizonte',),
 ('montes claros',),
 ('rio de janeiro',),
 ('lencois paulista',),
 ('caxias do sul',),
```


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 in 2017 are",data[0][0]
('Total orders in 2017 are', 45101)

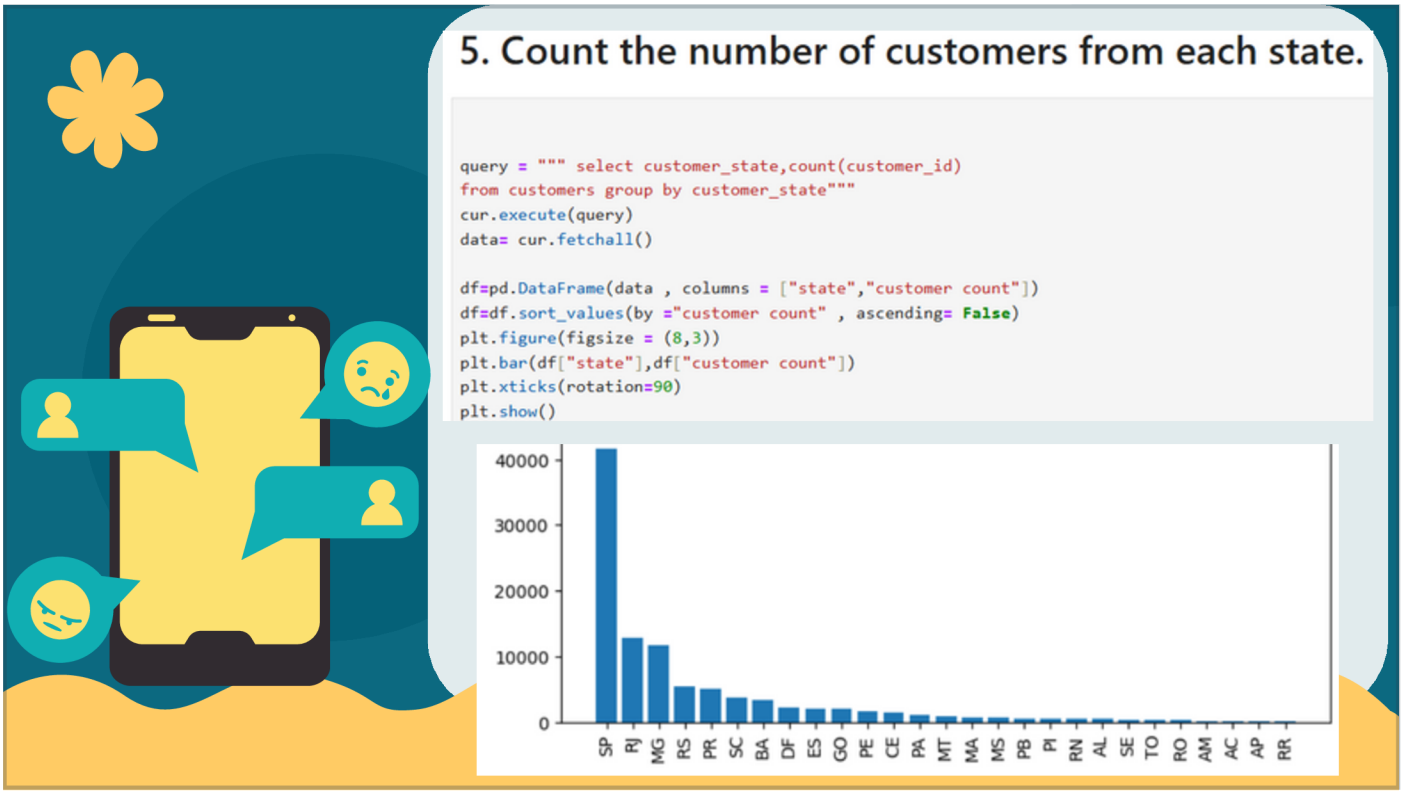
3. Find the total sales per category.

```
""" select products.product_category category ,
round(sum(payments.payment_value),2)sales
frpm products join orders_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.dataframde(data,columns = ["Category","Sales"])
df
```

	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

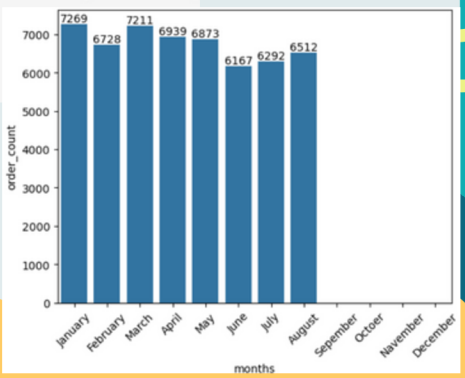

. Calculate the percentage of orders that were paid in installments.
query = """ select (sum(case when payment_installments >=1 then 1 else 0 end))/count(*)*100 from payments"""
cur.execute(query)
data= cur.fetchall()
data
[(Decimal('99.9981'),)]


5. 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.show()
40000
30000
20000
10000
0
SP
RJ
MG
RS
PR
SC
BA
DF
ES
GO
PE
CE
PA
MT
MA
MS
PB
PI
RN
AL
SE
TO
RO
AM
AC
AP
RR

6 Calculate the number of orders per month in 2018.

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

```
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","Sepember","Octoer","Navember","December"]
ax=sns.barplot(x=df["months"],y=df["order_count"],data=df,order= o)
plt.xticks(rotation=45)
ax.bar_label(ax.containers[0])
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) AS 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=["city","avg products/ orders"])
df.head(10)
```

	city	avg products/ orders
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

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

```
query = """select upper( products.product_category) category ,
round((sum(payments.payment_value)/(select sum(payment_value) from payments))*100,2)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 order by sales desc
 """
cur.execute(query)
data= cur.fetchall()
df=pd.DataFrame(data, columns=["category","percentage distribution"])
df.head(10)
```

	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
5	SPORT LEISURE	8.70
6	HOUSEWARES	6.84
7	AUTOMOTIVE	5.32
8	GARDEN TOOLS	5.24
9	COOL STUFF	4.87

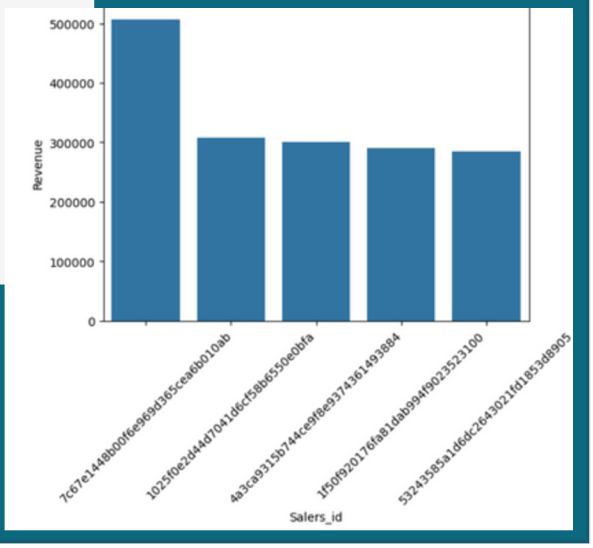
9. Identify the correlation between product price and the number of times a product has been purchased.

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import mysql.connector



db = mysql.connector.connect (host='localhost',
    user='root',
    password='Tuhin@2002',
    database='ECOMMERCE'
)

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"])
df.head(10)
arr1=df["order_count"]
arr2=df["price"]
np.corrcoef([arr1,arr2])
```

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

10. 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=["Salers_id","Revenue","Ranking"])
df=df.head()

sns.barplot(x = "Salers_id" ,y= "Revenue", data= df)
plt.xticks(rotation=45)
plt.show()
```



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

```
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
```

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

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"""
cur.execute(query)
data= cur.fetchall()
df=pd.DataFrame(data, columns=["year","months","payment","sum of payment"]
df
```

	year	months	payment	sum of 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
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

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))/
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

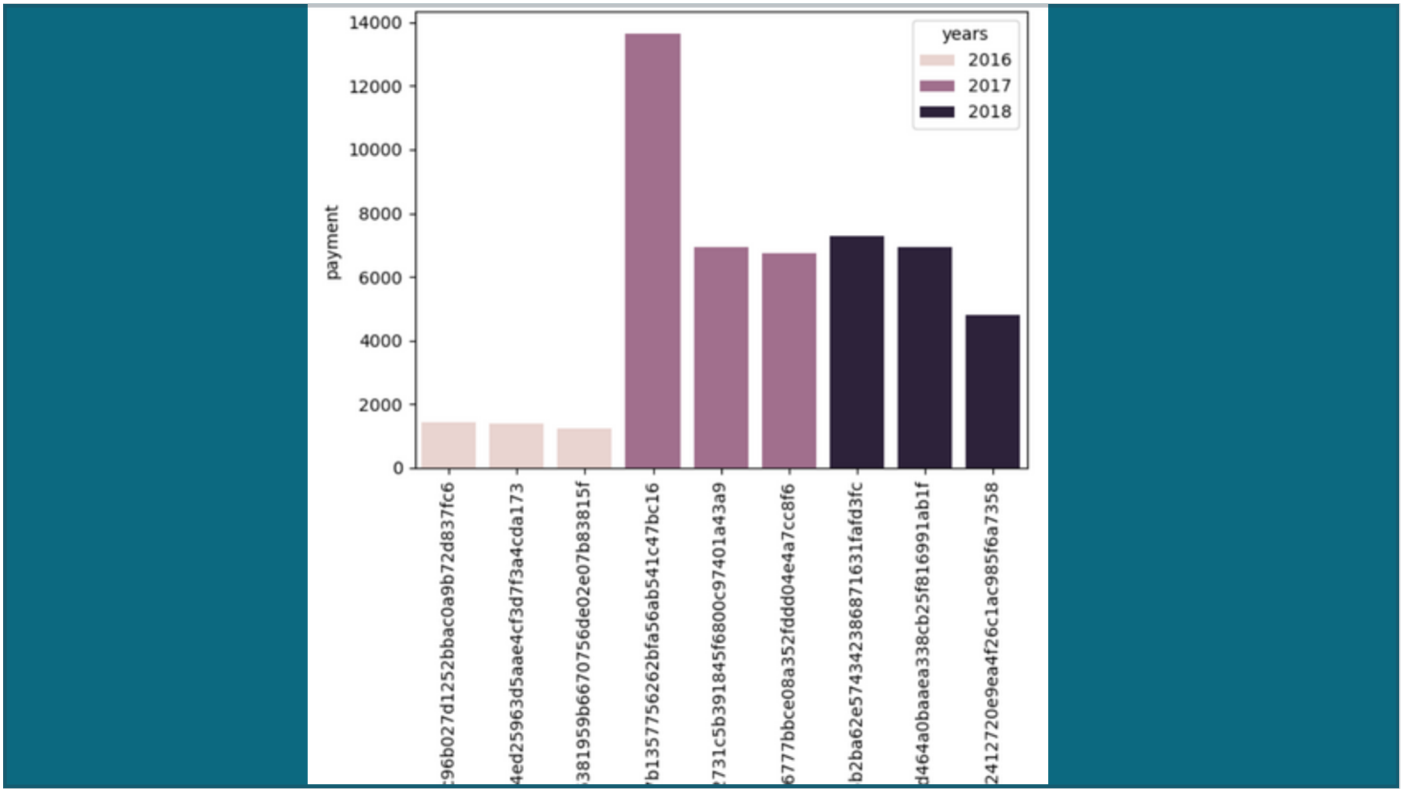
14. Calculate the retention rate of customers, defined as the percentage of customers who make another purchase within 6 months of their first purchase.

```
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 ;"""
cur.execute(query)
data = cur.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"""

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
2016
2017
2018
payment
14000
12000
10000
8000
6000
4000
2000
0


THANK YOU
FOR JOINING ME TODAY TO EXPLORE THE EXCITING TRENDS SHAPING THE FUTURE OF E-COMMERCE.