

```
In [61]: 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='127.0.0.1',
    user='root',
    password='Bittu_N@098',
    database='ecommerce')

cur = db.cursor()
```

1. List all unique cities where customers are located.

```
In [36]: query = """select distinct customer_city from customers"""

cur.execute(query)

data = cur.fetchall()

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

```
Out[36]:
```

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

2. Count the number of orders placed in 2017.

```
In [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]
```

```
Out[4]: ('total orders placed in 2017 are', 45101)
```

3. Find the total sales per category.

```
In [21]: query = """select products.product_category category,
round(sum(payments.payment_value),2) sales
from products join delivery
on products.product_id = delivery.product_id
join payments
on payments.order_id = delivery.order_id
group by category
"""

cur.execute(query)

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

```
Out[21]:
```

	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

4. Calculate the percentage of orders that were paid in installments.

```
In [25]: 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[25]: ('the percentage of orders that were paid in installments is',
Decimal('99.9981'))
```

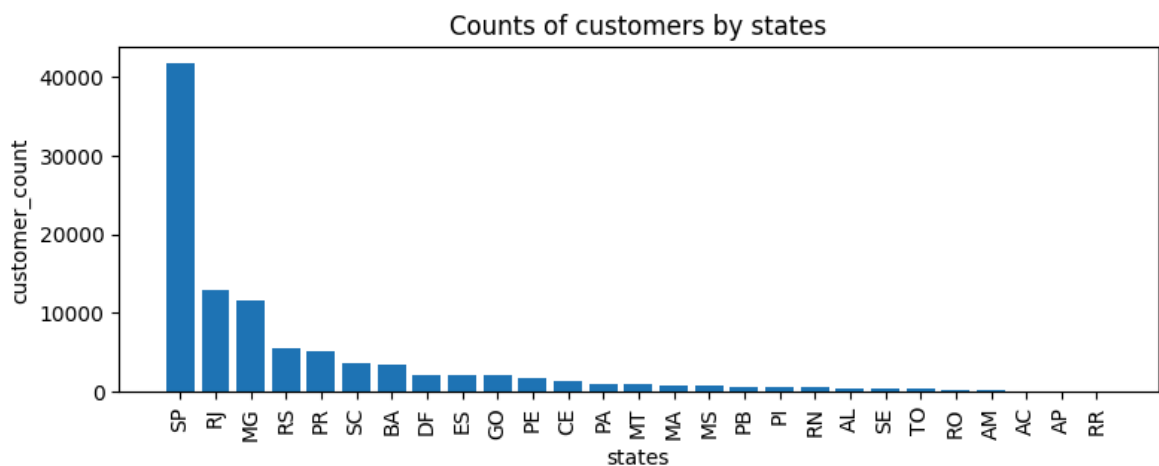
5. Count the number of customers from each state.

```
In [48]: 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 = (9,3))
plt.bar(df["state"], df["customer_count"])
plt.xticks(rotation=90)
plt.xlabel("states")
plt.ylabel("customer_count")
plt.title("Counts of customers by states")
plt.show()
```



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

```
In [51]: query = """select monthname(order_purchase_timestamp) months, count(order_id) or
from orders where year(order_purchase_timestamp) = 2018

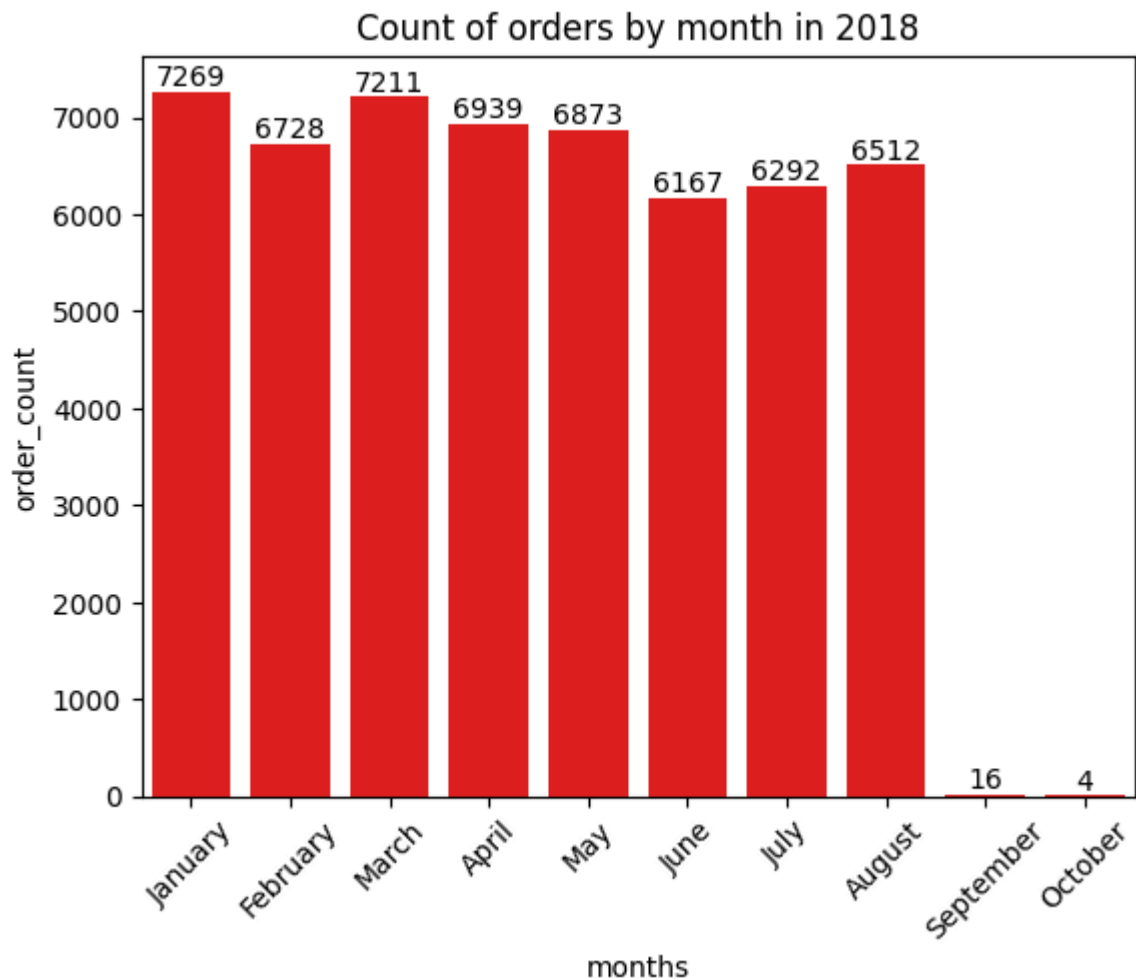
group by months
"""

cur.execute(query)

data = cur.fetchall()
df = pd.DataFrame(data, columns = ["months", "order_count"])
o = ["January", "February", "March", "April", "May", "June", "July", "August", "

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.title("Count of orders by month in 2018")
plt.show()
```



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

```
In [57]: query = """with count_per_order as
(select orders.order_id, orders.customer_id, count(delivery.order_id) as oc
from orders join delivery
on orders.order_id = delivery.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", "Avg orders/order"])
df.head(10)
```

Out[57]:

	customer city	Avg orders/order
0	padre carvalho	7.00
1	celso ramos	6.50
2	datas	6.00
3	candido godoi	6.00
4	matias olimpico	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

```
In [58]: query = """select products.product_category category,
round((sum(payments.payment_value)/(select sum(payment_value) from payments))*10
from products join delivery
on products.product_id = delivery.product_id
join payments
on payments.order_id = delivery.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
```

Out[58]:

	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
...
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

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

In [64]:

```
query = """select products.product_category,
count(delivery.product_id),
round( avg (delivery.price), 2)
from products join delivery
on products.product_id = delivery.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 btw price and the number of times a product has been purchased.
```

the correlation btw price and the number of times a product has been purchased. -
0.10631514167157562

10. Calculate the total revenue generated by each seller, and rank them by revenue.

In [72]:

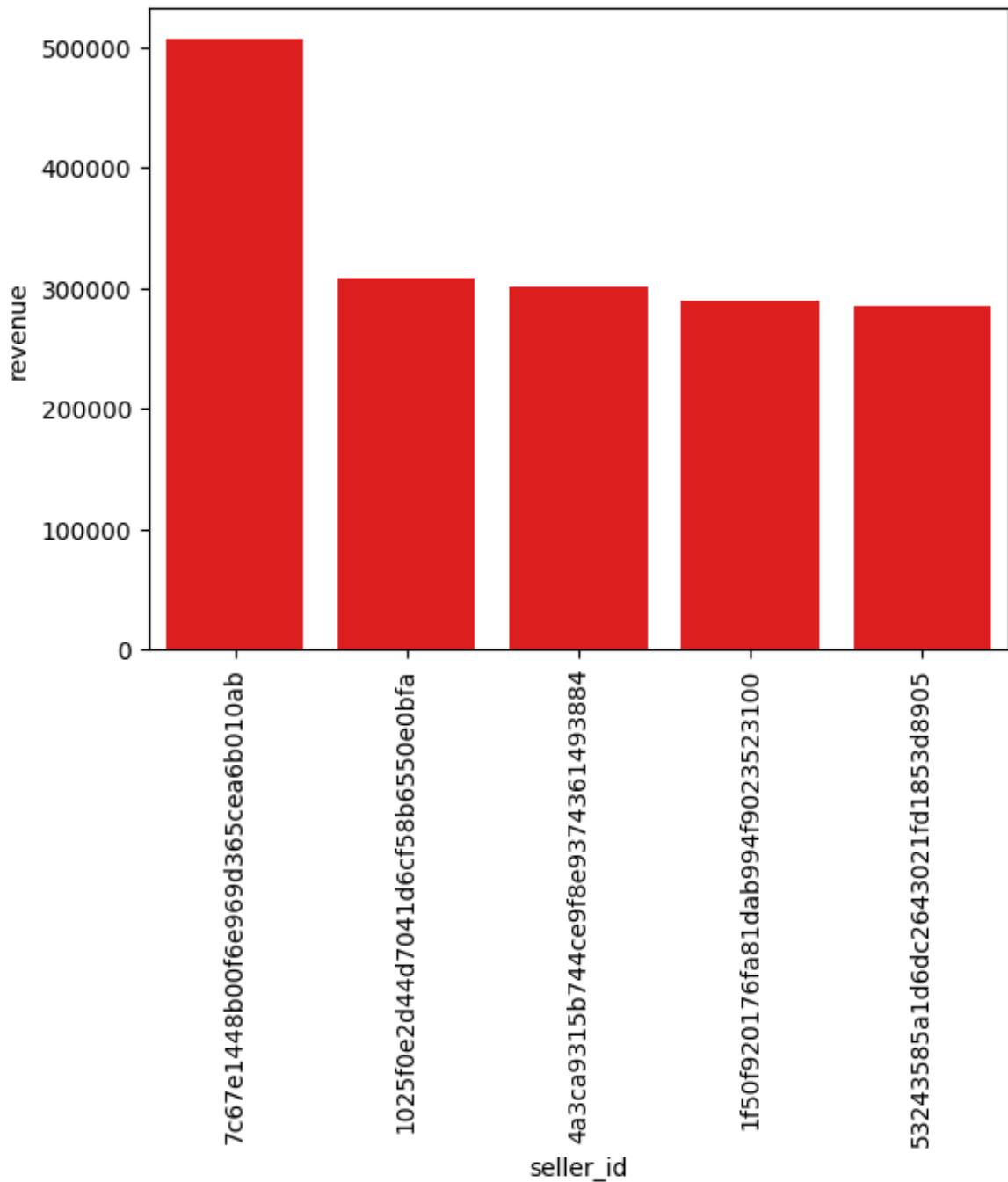
```
query = """select *, dense_rank() over(order by revenue desc) as rn from
(select delivery.seller_id, sum(payments.payment_value) revenue
```

```

from delivery
join payments
on delivery.order_id = payments.order_id
group by delivery.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, color = "Red")
plt.xticks(rotation = 90)
plt.show()

```



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

history.

```
In [75]: 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, columns = ["customer_id", "order_purchase_timestamp", "p
df
```

```
Out[75]:
```

	customer_id	order_purchase_timestamp	payment	movin
0	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.74	114.74
1	000161a058600d5901f007fab4c27140	2017-07-16 09:40:32	67.41	67.41
2	0001fd6190edaaf884bcaf3d49edf079	2017-02-28 11:06:43	195.42	195.42
3	0002414f95344307404f0ace7a26f1d5	2017-08-16 13:09:20	179.35	179.35
4	000379cdec625522490c315e70c7a9fb	2018-04-02 13:42:17	107.01	107.01
...
103881	fffecc9f79fd8c764f843e9951b11341	2018-03-29 16:59:26	71.23	27.12
103882	fffed5b6d849fbd39689bb92087f431	2018-05-22 13:36:02	63.13	63.13
103883	ffff42319e9b2d713724ae527742af25	2018-06-13 16:57:05	214.13	214.13
103884	ffffa3172527f765de70084a7e53aae8	2017-09-02 11:53:32	45.50	45.50
103885	ffffe8b65bbe3087b653a978c870db99	2017-09-29 14:07:03	18.37	18.37

103886 rows × 4 columns



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

```
In [77]: 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 = ["years", "months", "payment", "cumulative_sal
df
```

Out[77]:

	years	months	payment	cumulative_sales
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

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

```
In [82]: 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[82]:

	years	YoY % growth
0	2016	NaN
1	2017	0.205732
2	2018	0.000017

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

```

In [85]: 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
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()
print("none of the customers have repeat purchase within 6 months of there first
data

```

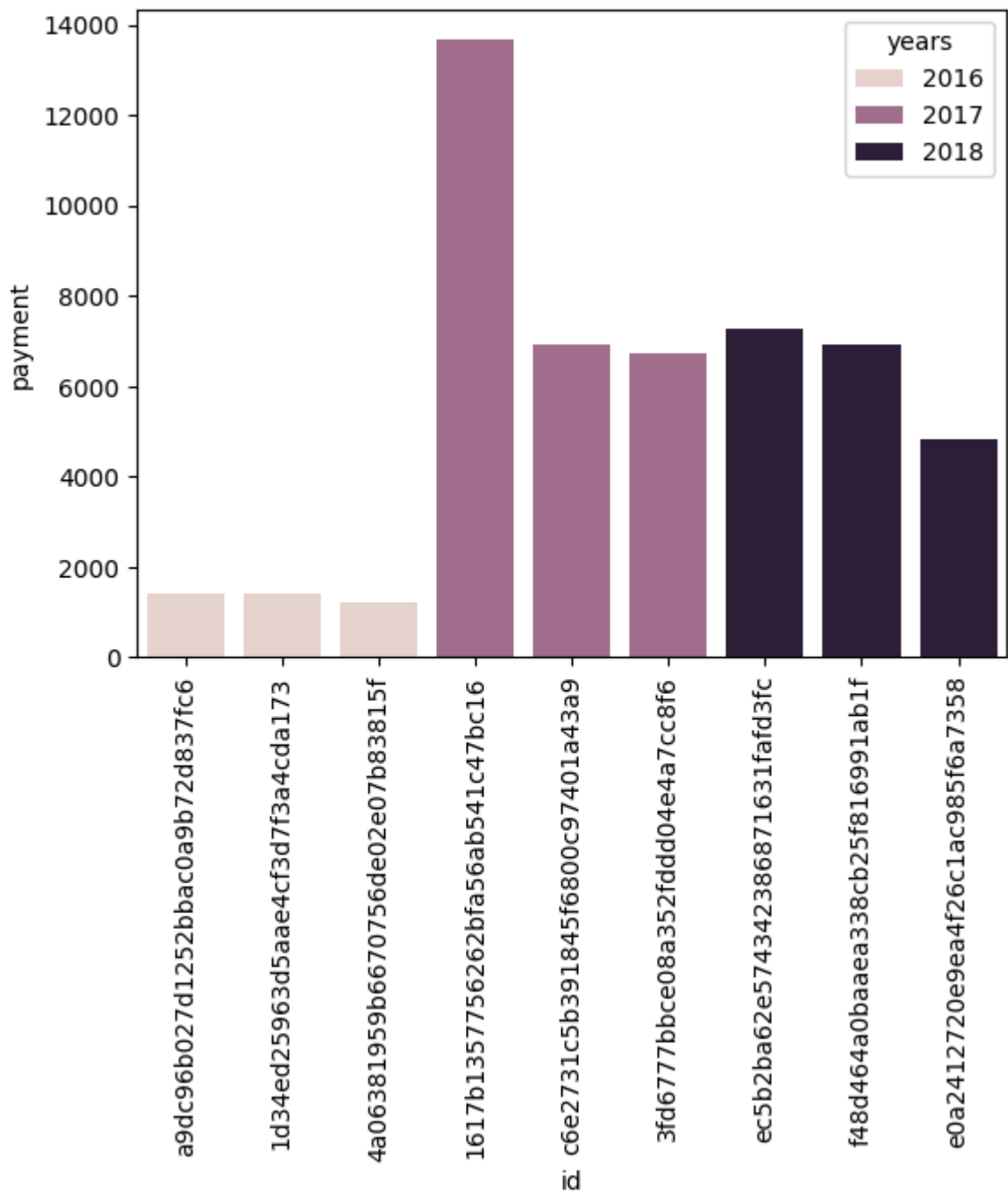
none of the customers have repeat purchase within 6 months of there first purchas
e, means no repeat customer.

Out[85]: [(None,)]

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

```
In [87]: 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()
```



In []: