python+sql_ecommerce

November 17, 2024

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import mysql.connector


db = mysql.connector.connect(host = "localhost",
                             username = "root",
                             password = "",
                             database = "ecommerce")

cur = db.cursor()
```

1 List all unique cities where customers are located.

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

cur.execute(query)

data = cur.fetchall()

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

[2]:
```
                        0
0                  franca
1   sao bernardo do campo
2               sao paulo
3         mogi das cruzes
4                campinas
```

2 Count the number of orders placed in 2017.

[3]:
```
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]
```

[3]:
```
('total orders placed in 2017 are', 135303)
```

3 Find the total sales per category.

[4]:
```
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
```

[4]:
```
                      Category       Sales
0                    PERFUMERY  2026954.64
1         FURNITURE DECORATION  5720705.57
2                    TELEPHONY  1947528.20
3               BED TABLE BATH  6850214.68
4                   AUTOMOTIVE  3409177.32
..                         …         …
69              CDS MUSIC DVDS     4797.72
70                  LA CUISINE    11654.12
71  FASHION CHILDREN'S CLOTHING     3142.68
72                    PC GAMER     8697.72
73       INSURANCE AND SERVICES     1298.04

[74 rows x 2 columns]
```

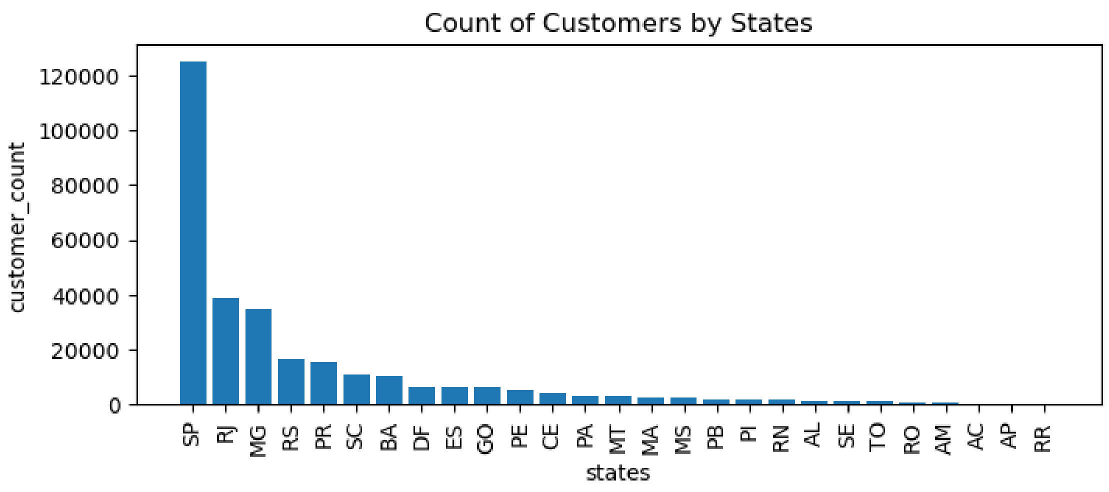
4 Calculate the percentage of orders that were paid in installments.

```
[5]: 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]
```

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

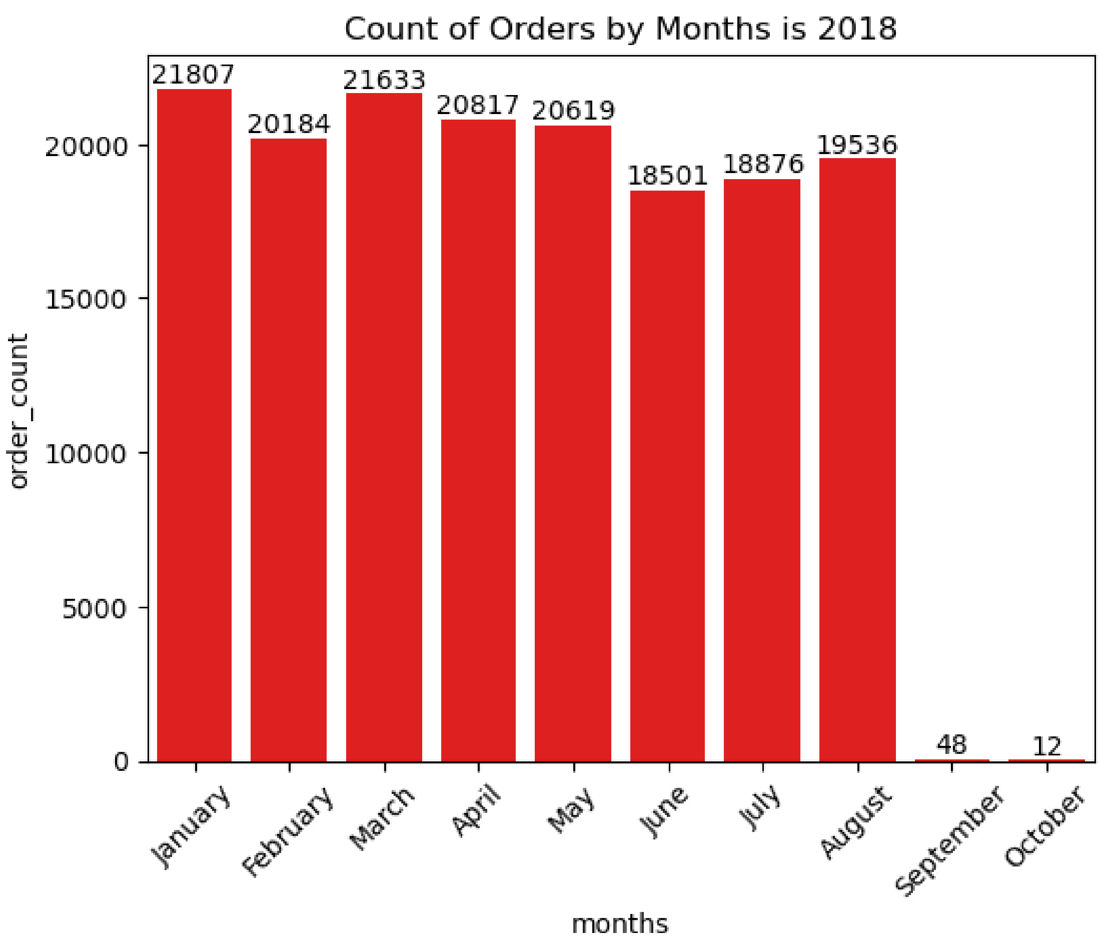
5 Count the number of customers from each state.

```
[6]: 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()
```



6 Calculate the number of orders per month in 2018.

[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()
```



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

[8]:
```
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/order"])
df.head(10)
```

```
[8]:        customer city average products/order
     0      padre carvalho                  21.00
     1         celso ramos                  19.50
     2               datas                  18.00
     3       candido godoi                  18.00
     4      matias olimpio                  15.00
     5          cidelandia                  12.00
     6          curralinho                  12.00
     7             picarra                  12.00
     8  morro de sao paulo                  12.00
     9     teixeira soares                  12.00
```

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

```
[9]: 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 distribution"])
     df.head()
```

```
[9]:              Category  percentage distribution
     0        BED TABLE BATH                    21.40
     1         HEALTH BEAUTY                    20.71
     2  COMPUTER ACCESSORIES                    19.81
     3  FURNITURE DECORATION                    17.87
     4       WATCHES PRESENT                    17.86
```

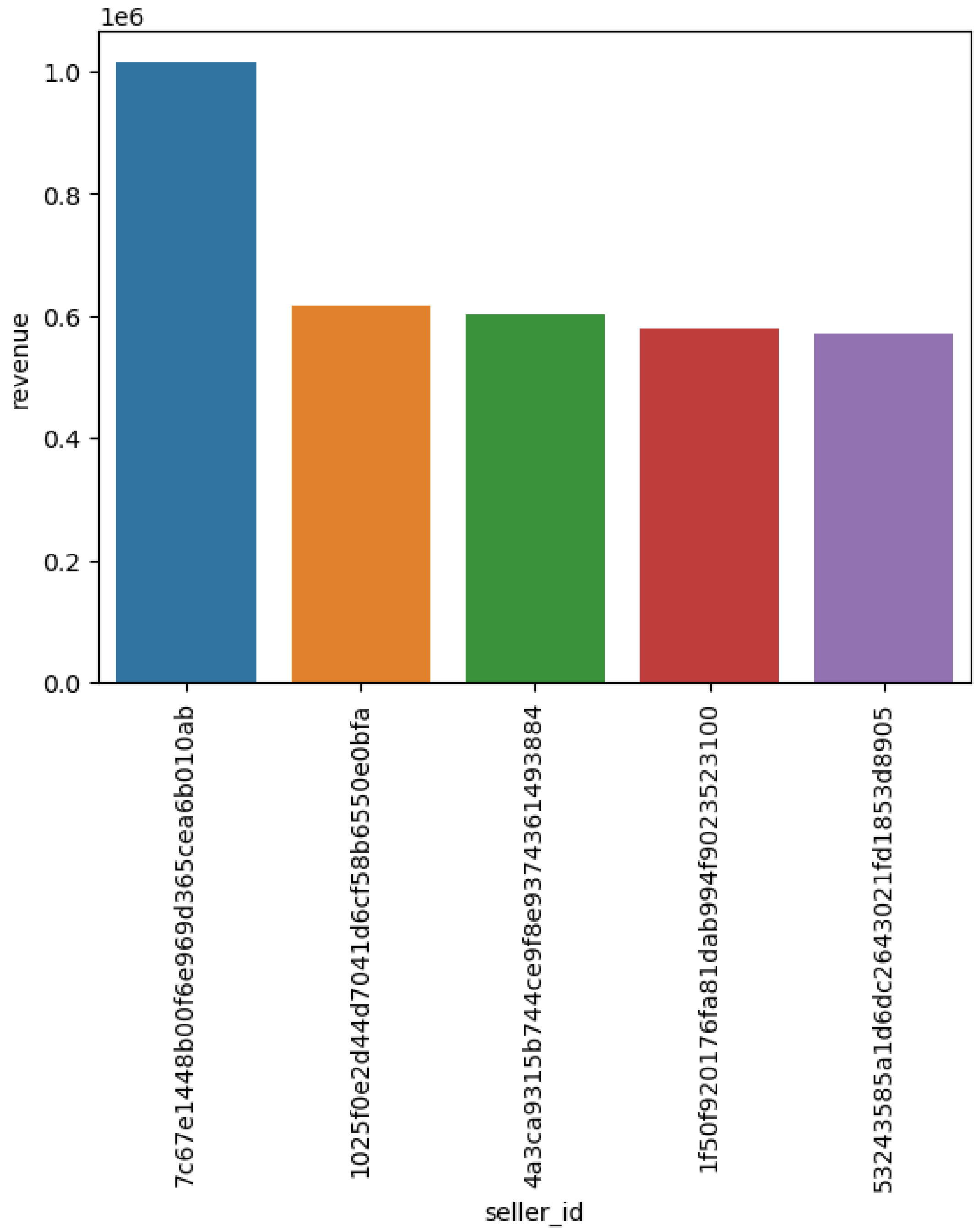
9 Identify the correlation between product price and the number of times a product has been purchased.

```
[10]: 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
```

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

```
[11]: 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()
```


1e6
1.0
0.8
0.6
0.4
0.2
0.0
revenue
7c67e1448b00f6e969d365cea6b010ab
1025f0e2d44d7041d6cf58b6550e0bfa
4a3ca9315b744ce9f8e9374361493884
1f50f920176fa81dab994f9023523100
53243585a1d6dc2643021fd1853d8905
seller_id

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

[12]:
```
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
```

[12]:
```
                                       0                    1       2  \
0       00012a2ce6f8dcda20d059ce98491703  2017-11-14 16:08:26  114.74
1       00012a2ce6f8dcda20d059ce98491703  2017-11-14 16:08:26  114.74
2       00012a2ce6f8dcda20d059ce98491703  2017-11-14 16:08:26  114.74
3       00012a2ce6f8dcda20d059ce98491703  2017-11-14 16:08:26  114.74
4       00012a2ce6f8dcda20d059ce98491703  2017-11-14 16:08:26  114.74
…                                      …                    …       …
623311  ffffe8b65bbe3087b653a978c870db99  2017-09-29 14:07:03   18.37
623312  ffffe8b65bbe3087b653a978c870db99  2017-09-29 14:07:03   18.37
623313  ffffe8b65bbe3087b653a978c870db99  2017-09-29 14:07:03   18.37
623314  ffffe8b65bbe3087b653a978c870db99  2017-09-29 14:07:03   18.37
623315  ffffe8b65bbe3087b653a978c870db99  2017-09-29 14:07:03   18.37

                 3
0       114.739998
1       114.739998
2       114.739998
3       114.739998
4       114.739998
…              …
623311   18.370001
623312   18.370001
623313   18.370001
623314   18.370001
623315   18.370001

[623316 rows x 4 columns]
```

12 Calculate the cumulative sales per month for each year.

[13]:
```
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
```

[13]:
```
       0   1           2            3
0   2016   9     1513.44      1513.44
1   2016  10   354542.88    356056.32
2   2016  12      117.72    356174.04
3   2017   1   830928.24   1187102.28
4   2017   2  1751448.06   2938550.34
5   2017   3  2699181.60   5637731.94
6   2017   4  2506728.18   8144460.12
7   2017   5  3557512.92  11701973.04
8   2017   6  3067658.28  14769631.32
9   2017   7  3554297.52  18323928.84
10  2017   8  4046377.92  22370306.76
11  2017   9  4366574.70  26736881.46
12  2017  10  4678067.28  31414948.74
13  2017  11  7169296.80  38584245.54
14  2017  12  5270408.88  43854654.42
15  2018   1  6690025.07  50544679.49
16  2018   2  5954780.04  56499459.53
17  2018   3  6957912.72  63457372.25
18  2018   4  6964712.88  70422085.13
19  2018   5  6923892.91  77345978.04
20  2018   6  6143283.00  83489261.04
21  2018   7  6399244.49  89888505.53
22  2018   8  6134551.93  96023057.46
23  2018   9    26637.24  96049694.70
24  2018  10     3538.02  96053232.72
```

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

[14]:
```
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
```

[14]:
```
   years  yoy % growth
0   2016           NaN
1   2017  12112.703759
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.

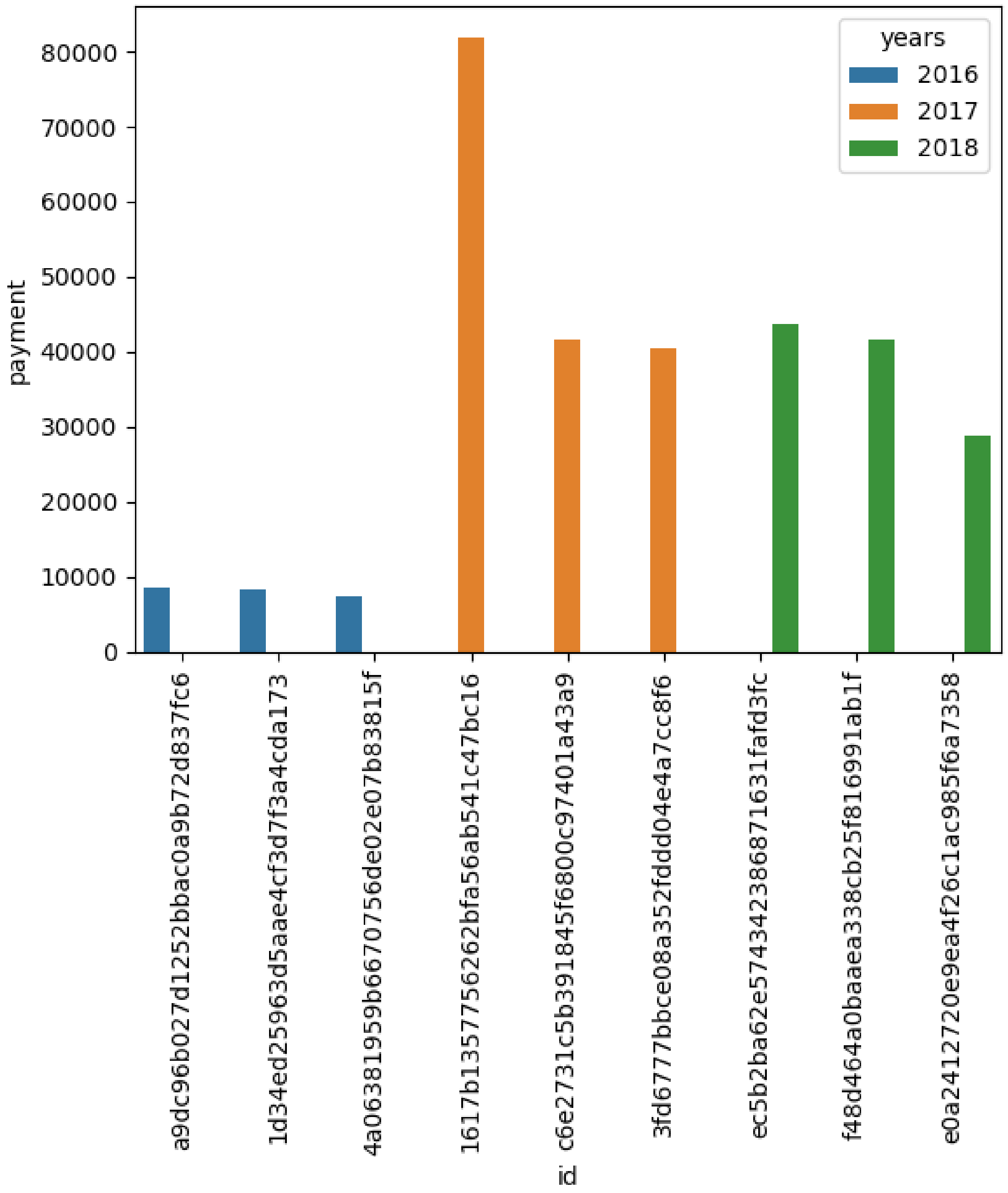
[15]:
```
query = """with a as (select customers.customer_id,
min(orders.order_purchase_timestamp) first_order
from customers join orders
on customers.customer_id = orders.customer_id
group by customers.customer_id),

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
```

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

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

```
[16]: 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
80000
70000
60000
50000
40000
30000
20000
10000
0
a9dc96b027d1252bbac0a9b72d837fc6
1d34ed25963d5aae4cf3d7f3a4cda173
4a06381959b6670756de02e07b83815f
1617b1357756262bfa56ab541c47bc16
c6e2731c5b391845f6800c97401a43a9
3fd6777bbce08a352fddd04e4a7cc8f6
ec5b2ba62e574342386871631fafd3fc
f48d464a0baaea338cb25f816991ab1f
e0a2412720e9ea4f26c1ac985f6a7358
id