Importing Necessary Libraries to Data Analysis

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

Set Connection to SQL Database with Notebook

```
db= mysql.connector.connect( host='127.0.0.1',
    user='root',
    password='12345678910',
    database='ecomers'
)

cur=db.cursor()
```

Basic Queries

List all unique cities where customers are located.

```
 query= """ select distinct customer_city
 from customers"""

cur.execute(query)
data = cur.fetchall()
print( " Top 5 cities is franca, sao bernardo do campo, sao paulo,
mogi das cruzes, campinas")
```

```
 Top 5 cities is franca, sao bernardo do campo, sao paulo, mogi das
cruzes, campinas
```

Count the number of orders placed in 2017.

```
 query= """ select count(order_id) from orders where
order_purchase_timestamp = 2017 """

cur.execute(query)
data = cur.fetchall()
data
print( f" Total unique order id from 2017 is : {data}")
```

```
 Total unique order id from 2017 is : [(45101,)]
```

Find the total sales per category.

```
query= """ select product_category category,
round(sum(payments.payment_value),2) sale
```

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

dff
```

	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 x 2 columns]

Calculate the percentage of orders that were paid in installments.

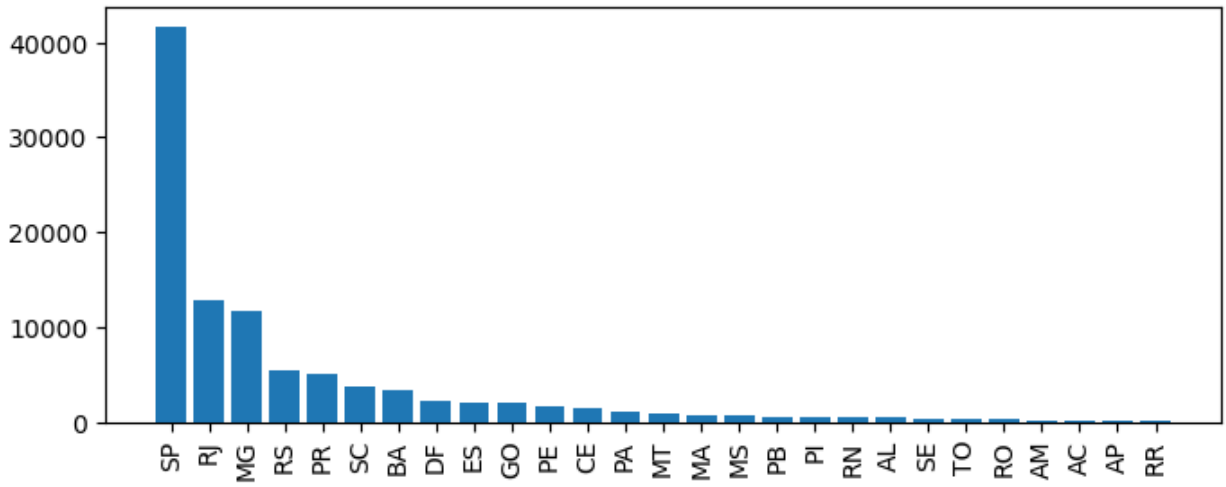
```
 query= """ select (sum(case when payment_installments >=1
 then 1 else 0 end))/count(*)*100 from payments """

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

```
[(Decimal('99.9981'),)]
```

Count the number of customers from each state.

```
 query= """ select customer_state, count(customer_unique_id)
 from customers
 group by customer_state"""

cur.execute(query)
data = cur.fetchall()
data
dff=pd.DataFrame(data, columns=["State", "Count_of_customers"])
dff=dff.sort_values(by="Count_of_customers", ascending=False)
```

```
plt.figure(figsize=(8,3))
plt.bar(dff["State"], dff["Count_of_customers"])
plt.xticks(rotation=90)
plt.show()
```



Intermediate Queries

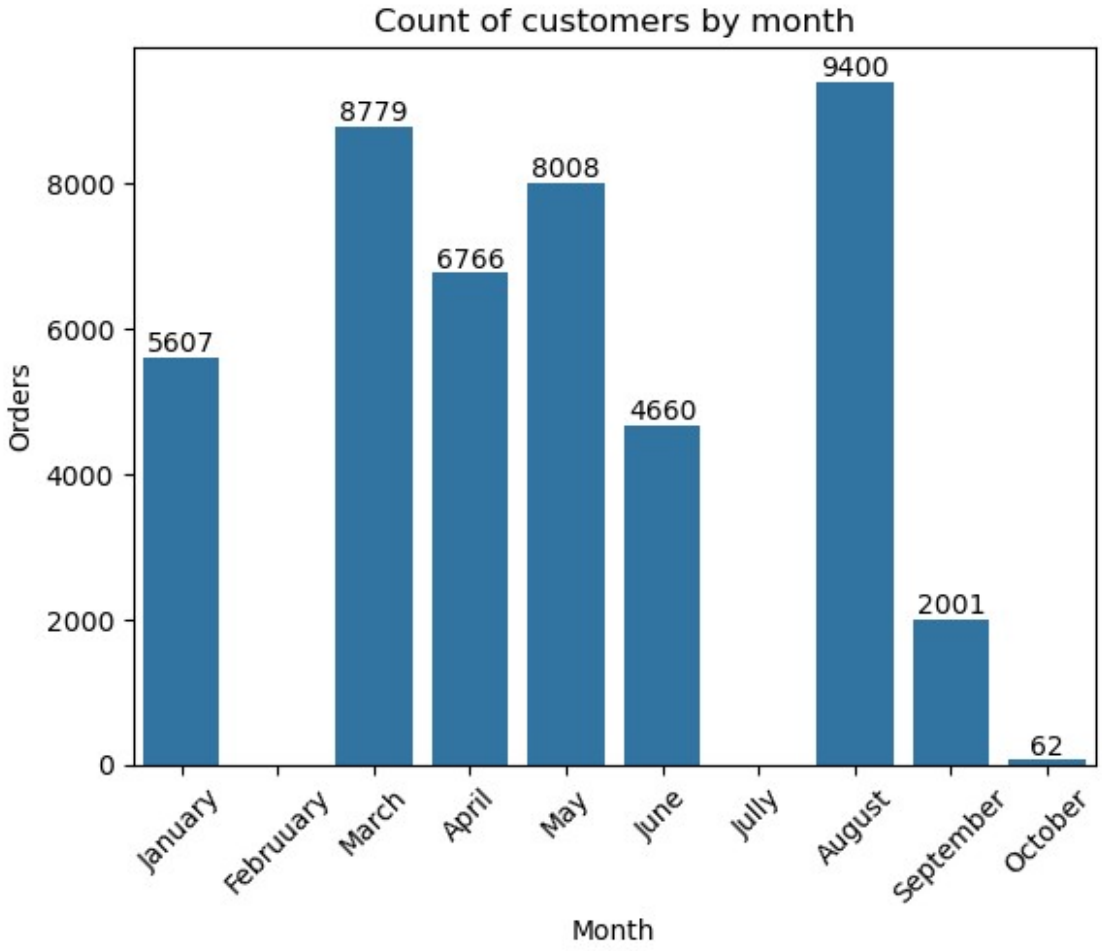
Calculate the number of orders per month in 2018.

```
 query= """ select count(order_purchase_timestamp),
monthname(order_estimated_delivery_date) month_year
from orders
where order_estimated_delivery_date = 2018
group by month_year;
 """

cur.execute(query)
data = cur.fetchall()
dff=pd.DataFrame(data, columns=["Orders", "Month"])
dff=dff.sort_values(by="Orders", ascending=False)
o=["January", "Februuary", "March", "April", "May",
"June","Jully","August","September","October"]

dff
```

	Orders	Month
0	9400	August
4	8779	March
6	8008	May
3	7210	July
7	6766	April
2	6304	February
5	5607	January
8	4660	June

```
1     2001  September
9       62    October
10       1   November
```

```
ax=sns.barplot(x=dff["Month"], y=dff["Orders"], data=dff, order=o)
plt.xticks(rotation=45)
ax.bar_label(ax.containers[0])
plt.title("Count of customers by month")
plt.show()
```




```
 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
"""

cur.execute(query)
data = cur.fetchall()
df=pd.DataFrame(data, columns=["Customer_city","Average_per_ordr"])
df.head(10)
```

	Customer_city	Average_per_ordr
0	sao paulo	1.16
1	sao jose dos campos	1.14
2	porto alegre	1.17
3	indaial	1.12
4	treze tilias	1.27
5	rio de janeiro	1.15
6	mario campos	1.33
7	guariba	1.00
8	cuiaba	1.20
9	franca	1.25

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

```
query= """select product_category category,
round((sum(payments.payment_value)/
(select sum(payment_value) from payments sale))*100,2) sale
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 sale desc"""

cur.execute(query)
data = cur.fetchall()
df=pd.DataFrame(data, columns=["Category","Sale_in_percent"])
df.head(5)
```

	Category	Sale_in_percent
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.

```
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","avg_price"])
df.head(5)
```

	Category	Order_count	avg_price
0	HEALTH BEAUTY	9670	130.16
1	sport leisure	8641	114.34
2	Cool Stuff	3796	167.36
3	computer accessories	7827	116.51
4	Watches present	5991	201.14

```
arr1=df["Order_count"]
arr2=df["avg_price"]

a=np.corrcoef([arr1,arr2])
print("The correlation between price and number of product has been
purchesed is",a[0],[1])
```

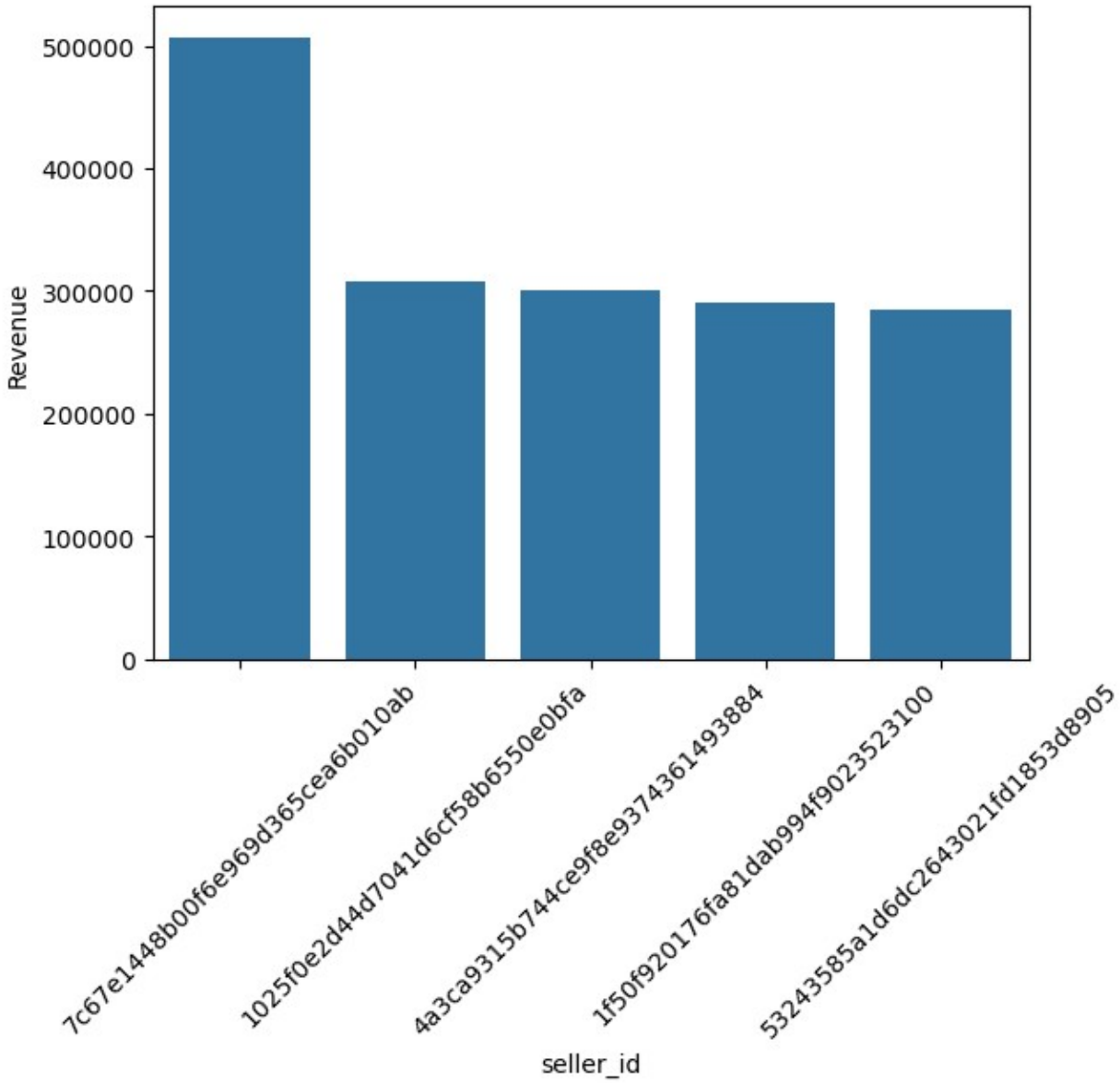
```
The correlation between price and number of product has been purchesed
is [ 1.         -0.10631514] [1]
```

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","Renk"])
df=df.head(5)
df
```

	seller_id	Revenue	Renk
0	7c67e1448b00f6e969d365cea6b010ab	507166.907302	1
1	1025f0e2d44d7041d6cf58b6550e0bfa	308222.039840	2
2	4a3ca9315b744ce9f8e9374361493884	301245.269765	3
3	1f50f920176fa81dab994f9023523100	290253.420128	4
4	53243585a1d6dc2643021fd1853d8905	284903.080498	5

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



Advanced Queries

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, columns=["Customer_id","Order purchase
time","Mov_avg", "Price"])
df=df.head(5)
df
```

```
                        Customer_id  Order purchase time  Mov_avg
Price
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
```

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(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","Payments", "Cumulative
sales"])
# df=df.head(5)
df
```

```
    Year  Months    Payments  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

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

```
query= """ with a as(select year(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=["Year","Year-over-year-growth"])
# df=df.head(5)
df
```

	Year	Year-over-year-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.

```
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 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()
df=pd.DataFrame(data)#, columns=["Year","Year-over-year-growth"])
# df=df.head(5)
df
```

```
      0
0  None
```

```
# 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 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;
```

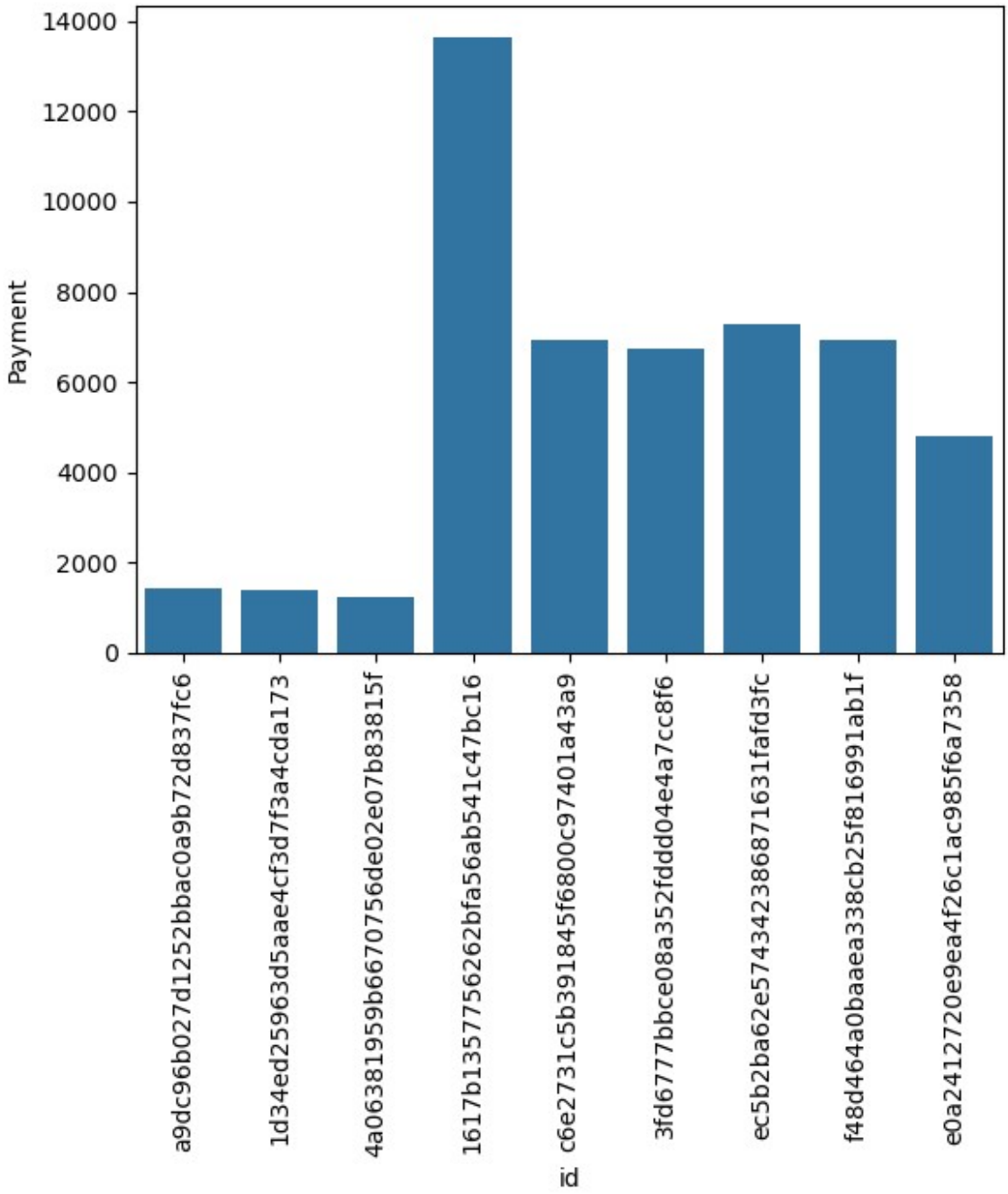
Identify the top 3 customers who spent the most money in each year.

```
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=["Year","id", "Payment","Rank"])
# df=df.head(5)
df
```

	Year	id	Payment	Rank
0	2016	a9dc96b027d1252bbac0a9b72d837fc6	1423.550049	1
1	2016	1d34ed25963d5aae4cf3d7f3a4cda173	1400.739990	2
2	2016	4a06381959b6670756de02e07b83815f	1227.780029	3
3	2017	1617b1357756262bfa56ab541c47bc16	13664.080078	1
4	2017	c6e2731c5b391845f6800c97401a43a9	6929.310059	2
5	2017	3fd6777bbce08a352fddd04e4a7cc8f6	6726.660156	3
6	2018	ec5b2ba62e574342386871631fafd3fc	7274.879883	1
7	2018	f48d464a0baaea338cb25f816991ab1f	6922.209961	2
8	2018	e0a2412720e9ea4f26c1ac985f6a7358	4809.439941	3

```
# 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;

sns.barplot(x = "id", y="Payment", data= df)
plt.xticks(rotation=90)
plt.show()
```


14000
12000
10000
8000
6000
4000
2000
0
Payment
a9dc96b027d1252bbac0a9b72d837fc6
1d34ed25963d5aae4cf3d7f3a4cda173
4a06381959b6670756de02e07b83815f
1617b1357756262bfa56ab541c47bc16
c6e2731c5b391845f6800c97401a43a9
3fd6777bbce08a352fddd04e4a7cc8f6
ec5b2ba62e574342386871631fafd3fc
f48d464a0baaea338cb25f816991ab1f
e0a2412720e9ea4f26c1ac985f6a7358
id