```python
import pandas as pd
import pymysql
import os

# List of CSV files and their corresponding table names
csv_files = [
    ('customers.csv', 'customers'),
    ('orders.csv', 'orders'),
    ('sellers.csv', 'sellers'),
    ('products.csv', 'products'),
    ('geolocation.csv', 'geolocation'),
    ('payments.csv', 'payments'),
    ('order_items.csv','order_items')   # Added payments.csv for
specific handling
]

# Connect to the MySQL database
conn = pymysql.connect(
    host='localhost',
    port=3306,
    user='root',
    password='100128',
    database='ecommerce'
)
cursor = conn.cursor()

# Folder containing the CSV files

folder_path = 'D:\SQL Project\e-commerce'
file_name = "customers.csv"
file_path = os.path.join(folder_path, file_name)


def get_sql_type(dtype):
    if pd.api.types.is_integer_dtype(dtype):
        return 'INT'
    elif pd.api.types.is_float_dtype(dtype):
        return 'FLOAT'
    elif pd.api.types.is_bool_dtype(dtype):
        return 'BOOLEAN'
    elif pd.api.types.is_datetime64_any_dtype(dtype):
        return 'DATETIME'
    else:
        return 'TEXT'

for csv_file, table_name in csv_files:
    file_path = os.path.join(folder_path, csv_file)
    # Read the CSV file into a pandas DataFrame
    df = pd.read_csv(file_path)

    # Replace NaN with None to handle SQL NULL
    df = df.where(pd.notnull(df), None)

    # Debugging: Check for NaN values
    print(f"Processing {csv_file}")
    print(f"NaN values before replacement:\n{df.isnull().sum()}\n")

    # Clean column names
    df.columns = [col.replace(' ', '_').replace('-', '_').replace('.', 
'_') for col in df.columns]

    # Generate the CREATE TABLE statement with appropriate data types
    columns = ', '.join([f'`{col}` {get_sql_type(df[col].dtype)}' for 
col in df.columns])
    create_table_query = f'CREATE TABLE IF NOT EXISTS `{table_name}` 
({columns})'
    cursor.execute(create_table_query)

    # Insert DataFrame data into the MySQL table
    for _, row in df.iterrows():
        # Convert row to tuple and handle NaN/None explicitly
        values = tuple(None if pd.isna(x) else x for x in row)
        sql = f"INSERT INTO `{table_name}` ({', '.join(['`' + col + 
'`' for col in df.columns])}) VALUES ({', '.join(['%s'] * len(row))})"
        cursor.execute(sql, values)

    # Commit the transaction for the current CSV file
    conn.commit()

# Close the connection
conn.close()
```

```
<>:28: SyntaxWarning: invalid escape sequence '\S'
<>:28: SyntaxWarning: invalid escape sequence '\S'
C:\Users\HP\AppData\Local\Temp\ipykernel_10064\1599385218.py:28: 
SyntaxWarning: invalid escape sequence '\S'
  folder_path = 'D:\SQL Project\e-commerce'

Processing customers.csv
NaN values before replacement:
customer_id                 0
customer_unique_id          0
customer_zip_code_prefix    0
customer_city               0
customer_state              0
dtype: int64
```

```
Processing orders.csv
NaN values before replacement:
order_id                            0
customer_id                         0
order_status                        0
order_purchase_timestamp            0
order_approved_at                 160
order_delivered_carrier_date     1783
order_delivered_customer_date    2965
order_estimated_delivery_date       0
dtype: int64

Processing sellers.csv
NaN values before replacement:
seller_id                 0
seller_zip_code_prefix    0
seller_city               0
seller_state              0
dtype: int64

Processing products.csv
NaN values before replacement:
product_id                      0
product category              610
product_name_length           610
product_description_length    610
product_photos_qty            610
product_weight_g                2
product_length_cm               2
product_height_cm               2
product_width_cm                2
dtype: int64

Processing geolocation.csv
NaN values before replacement:
geolocation_zip_code_prefix    0
geolocation_lat                0
geolocation_lng                0
geolocation_city               0
geolocation_state              0
dtype: int64

Processing payments.csv
NaN values before replacement:
order_id                0
payment_sequential      0
payment_type            0
payment_installments    0
payment_value           0
dtype: int64
```

```
Processing order_items.csv
NaN values before replacement:
order_id                0
order_item_id           0
product_id              0
seller_id               0
shipping_limit_date     0
price                   0
freight_value           0
dtype: int64
```

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

conn = pymysql.connect(
    host='localhost',
    port=3306,
    user='root',
    password='100128',
    database='ecommerce'
)
cursor = conn.cursor()
```

List all unique cities where customers are located.

```
cursor = conn.cursor()
query = """ select distinct (customer_city) from customers """
cursor.execute(query)

data = cursor.fetchall()

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

```
                       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.

```
cursor = conn.cursor()
query = """  select count(order_id) from orders where
year(order_purchase_timestamp) = 2017 """
cursor.execute(query)

data = cursor.fetchall()

"total orders placed in 2017 are ", data [0][0]
```

```
('total orders placed in 2017 are ', 315707)
```

Find the total sales per category.

```
cursor = conn.cursor()

query = """select  upper(products.product_category),
round(sum(payments.payment_value),2) as 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 products.product_category """

cursor.execute(query)

data = cursor.fetchall()

data

df = pd.DataFrame (data, columns= ["category","sales"])
df
```

```
                        category        sales
0                            ART    247943.44
1                     COOL STUFF   6237584.00
2                 GAMES CONSOLES   1563843.03
3                      TELEPHONY   3895056.41
4                 SPORT LEISURE  11137020.47
..                           ...          ...
69                CDS MUSIC DVDS      9595.44
70                    LA CUISINE     23308.24
71  FASHION CHILDREN'S CLOTHING       6285.36
72                      PC GAMER     17395.44
73        INSURANCE AND SERVICES      2596.08

[74 rows x 2 columns]
```

Calculate the percentage of orders that were paid in installments.

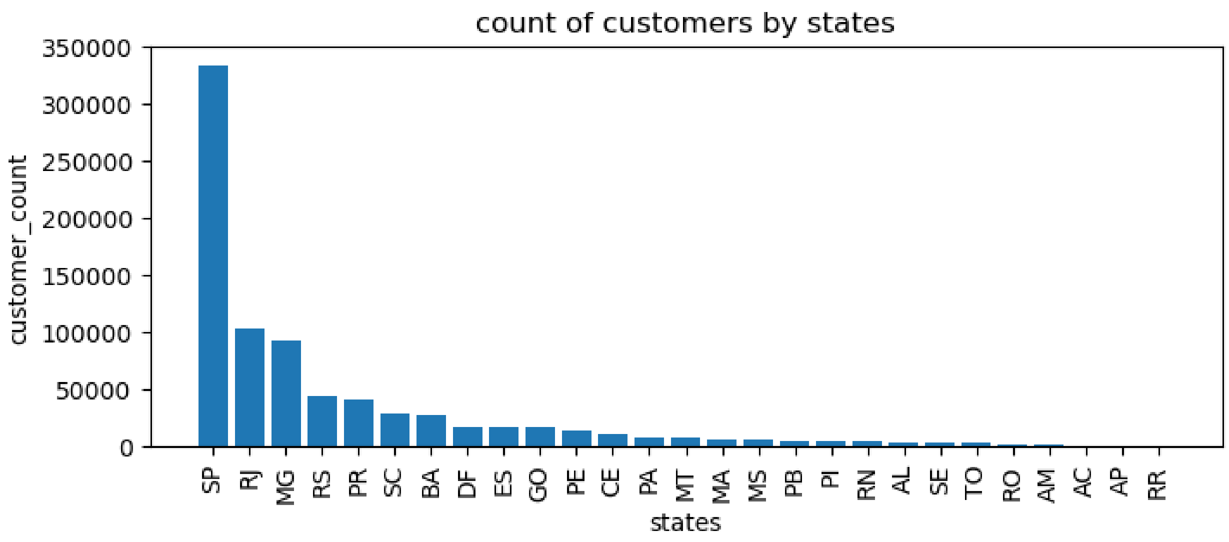
```
cursor = conn.cursor()

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

cursor.execute(query)

data = cursor.fetchall()

"the percentage of orders that were paid in installments is ",data[0]
[0]
```

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

Count the number of customers from each state.

```
cursor = conn.cursor()

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

cursor.execute(query)

data = cursor.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
```

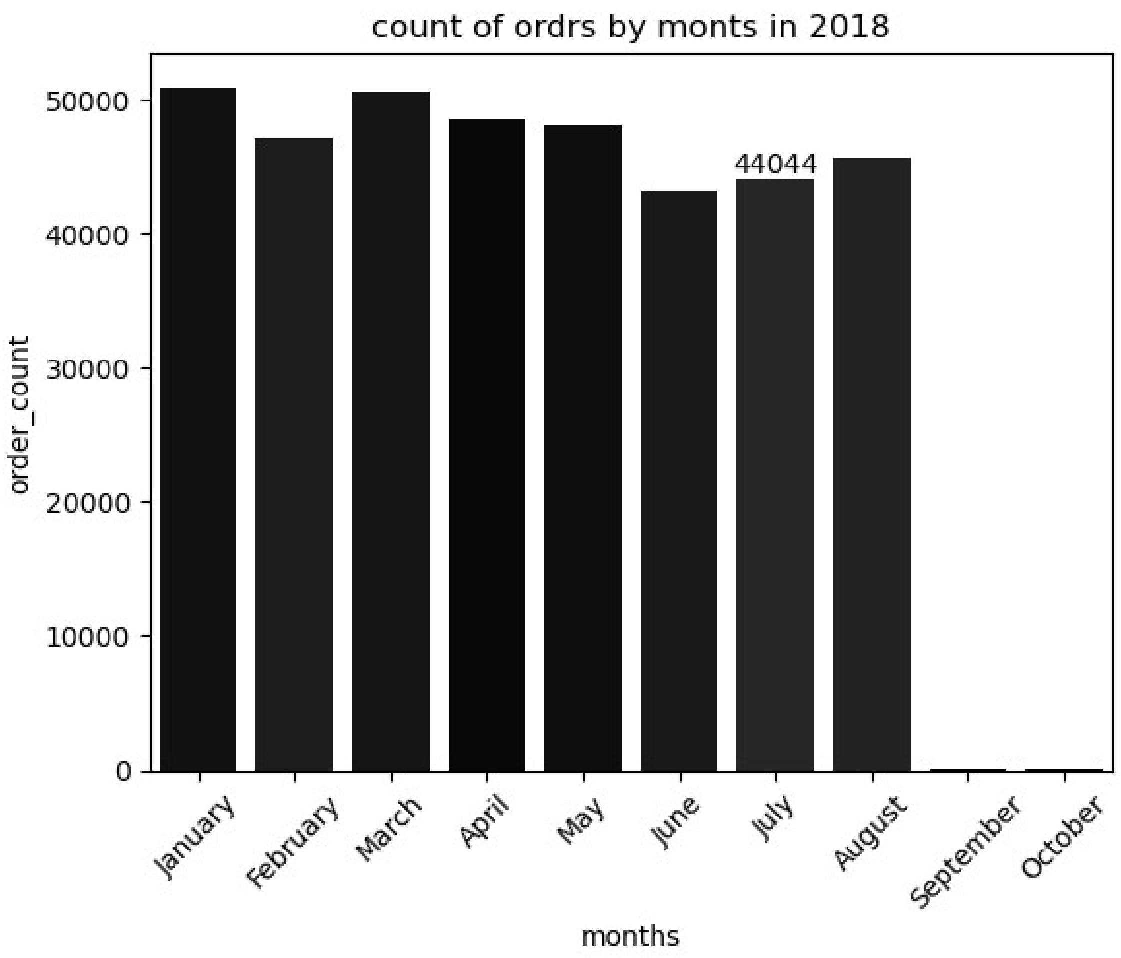
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Calculate the number of orders per month in 2018.

```
cursor = conn.cursor()

query = """ select monthname(order_purchase_timestamp) months,
count(order_id) order_count
from orders where year(order_purchase_timestamp) = 2018
group by months
"""

cursor.execute(query)

data = cursor.fetchall()
data
df = pd.DataFrame(data,columns = ["months", "order_count"])
o =
("January","February","March","April","May","June","July","August","Se
ptember","October")
fig,ax = plt.subplots()
sns.barplot(x= df["months"], y = df["order_count"], data = df, order =
o, ax=ax, hue=df["months"] ,color="black")
plt.xticks(rotation = 45)

ax.bar_label (ax.containers[0])
plt.title ("count of ordrs by monts in 2018")

plt.show()

C:\Users\HP\AppData\Local\Temp\ipykernel_10064\3433035414.py:15:
FutureWarning:
```

```
Setting a gradient palette using color= is deprecated and will be
removed in v0.14.0. Set `palette='dark:black'` for the same effect.

  sns.barplot(x= df["months"], y = df["order_count"], data = df, order
= o, ax=ax, hue=df["months"] ,color="black")
```



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

```
import pandas as pd
import pymysql
import os

# List of CSV files and their corresponding table names
csv_files = [
    ('customers.csv', 'customers'),
    ('orders.csv', 'orders'),
    ('sellers.csv', 'sellers'),
    ('products.csv', 'products'),
    ('geolocation.csv', 'geolocation'),
    ('payments.csv', 'payments'),
    ('order_items.csv','order_items')   # Added payments.csv for
specific handling
]

# Connect to the MySQL database
conn = pymysql.connect(
    host='localhost',
    port=3306,
    user='root',
    password='100128',
    database='ecommerce'
)

cursor = conn.cursor()

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
"""

cursor.execute(query)

data = cursor.fetchall()
data
df = pd.DataFrame(data,columns = ["customer city", "average order"])
df.head()
```

```
         customer city  average order
0         treze tilias           8.91
1              indaial           7.81
2  sao jose dos campos           7.97
3            sao paulo           8.09
4         porto alegre           8.22
```

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

```
import pandas as pd
import pymysql
import os



# List of CSV files and their corresponding table names
csv_files = [
    ('customers.csv', 'customers'),
    ('orders.csv', 'orders'),
    ('sellers.csv', 'sellers'),
    ('products.csv', 'products'),
    ('geolocation.csv', 'geolocation'),
    ('payments.csv', 'payments'),
    ('order_items.csv','order_items')   # Added payments.csv for
specific handling
]

# Connect to the MySQL database
conn = pymysql.connect(
    host='localhost',
    port=3306,
    user='root',
    password='100128',
    database='ecommerce'
)
cursor = conn.cursor()

query = """select upper(products.product_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 products.product_category order by sales_percentage desc"""
cursor.execute(query)

data = cursor.fetchall()
data
df = pd.DataFrame(data,columns = ["category", "percentage
distribution"])
df.head()
```

```
               category  percentage distribution
0        BED TABLE BATH                    42.79
1         HEALTH BEAUTY                    41.41
```

```
2  COMPUTER ACCESSORIES                        39.61
3  FURNITURE DECORATION                        35.73
4       WATCHES PRESENT                        35.71
```

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

```
import pandas as pd
import pymysql
import os
import numpy as np
# List of CSV files and their corresponding table names
csv_files = [
    ('customers.csv', 'customers'),
    ('orders.csv', 'orders'),
    ('sellers.csv', 'sellers'),
    ('products.csv', 'products'),
    ('geolocation.csv', 'geolocation'),
    ('payments.csv', 'payments'),
    ('order_items.csv','order_items')   # Added payments.csv for
specific handling
]

# Connect to the MySQL database
conn = pymysql.connect(
    host='localhost',
    port=3306,
    user='root',
    password='100128',
    database='ecommerce'
)
cursor = conn.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"""
cursor.execute(query)

data = cursor.fetchall()
data
df = pd.DataFrame(data,columns = ["category", "order_count","price"])
```

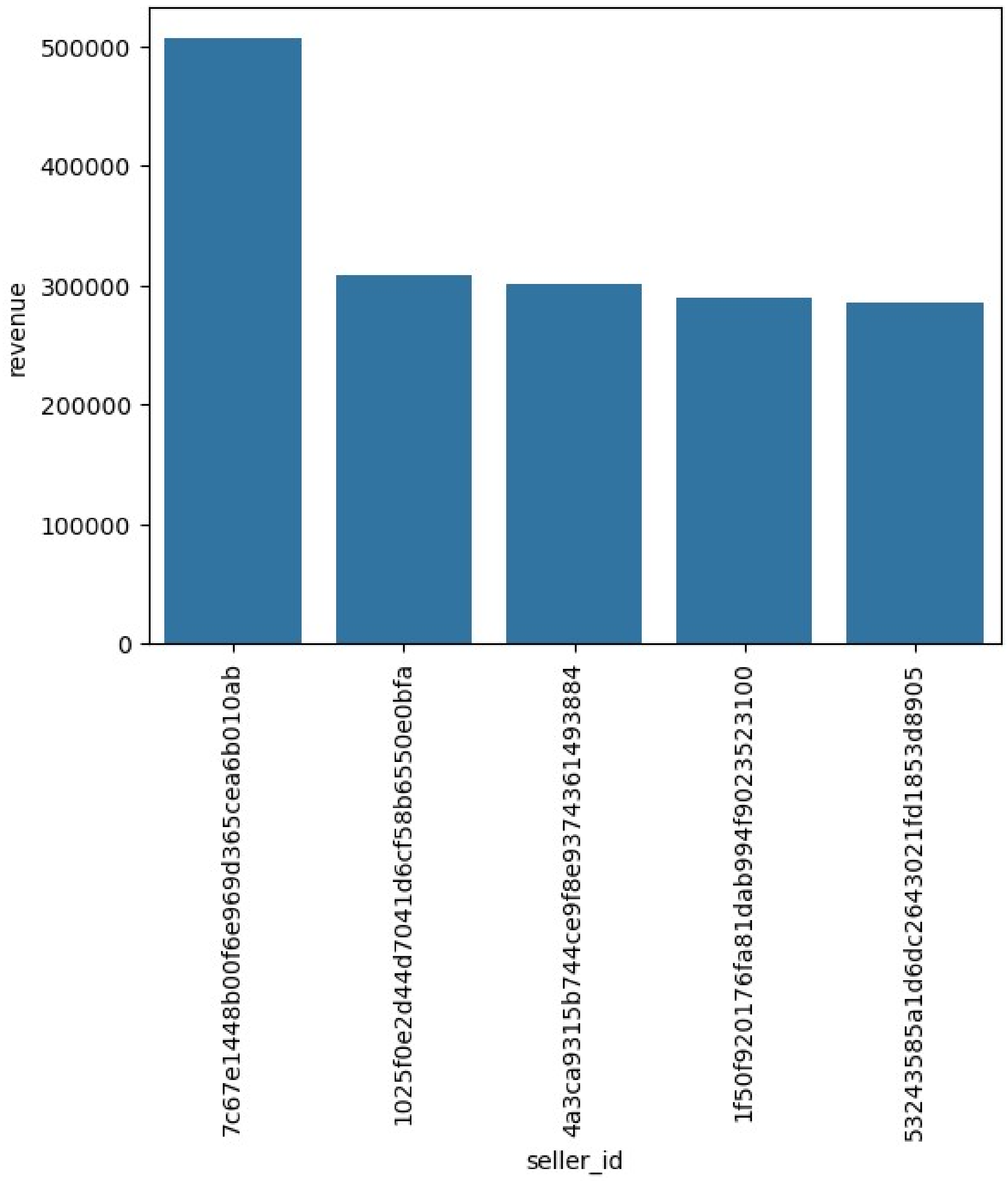
```
arr1 = df["order_count"]
arr2 = df["price"]

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

```
the correlation between product price and the number of times a
product has been purchased is  -0.10631514167157562
```

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

```
import pandas as pd
import pymysql
import os
import  seaborn as sns
# List of CSV files and their corresponding table names
csv_files = [
    ('customers.csv', 'customers'),
    ('orders.csv', 'orders'),
    ('sellers.csv', 'sellers'),
    ('products.csv', 'products'),
    ('geolocation.csv', 'geolocation'),
    ('payments.csv', 'payments'),
    ('order_items.csv','order_items')   # Added payments.csv for
specific handling
]

# Connect to the MySQL database
conn = pymysql.connect(
    host='localhost',
    port=3306,
    user='root',
    password='100128',
    database='ecommerce'
)

cursor = conn.cursor()

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
```

```
"""
cursor.execute(query)

data = cursor.fetchall()
data
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.

```python
import pandas as pd
import pymysql
import os
import  seaborn as sns
```

```
# List of CSV files and their corresponding table names
csv_files = [
    ('customers.csv', 'customers'),
    ('orders.csv', 'orders'),
    ('sellers.csv', 'sellers'),
    ('products.csv', 'products'),
    ('geolocation.csv', 'geolocation'),
    ('payments.csv', 'payments'),
    ('order_items.csv','order_items')   # Added payments.csv for
specific handling
]

# Connect to the MySQL database
conn = pymysql.connect(
    host='localhost',
    port=3306,
    user='root',
    password='100128',
    database='ecommerce'
)

cursor = conn.cursor()

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

"""
cursor.execute(query)

data = cursor.fetchall()
data
df = pd.DataFrame (data)
df.head()

                                  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  00012a2ce6f8dcda20d059ce98491703  2017-11-14 16:08:26  114.74
114.739998
```

Calculate the cumulative sales per month for each year.

```
 cursor = conn.cursor()

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

"""
cursor.execute(query)

data = cursor.fetchall()
data
df = pd.DataFrame (data)
df.head()

      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
```

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

```
import pandas as pd
import pymysql
import os
import  seaborn as sns
# List of CSV files and their corresponding table names
csv_files = [
    ('customers.csv', 'customers'),
    ('orders.csv', 'orders'),
    ('sellers.csv', 'sellers'),
    ('products.csv', 'products'),
    ('geolocation.csv', 'geolocation'),
    ('payments.csv', 'payments'),
    ('order_items.csv','order_items')   # Added payments.csv for
specific handling
]

# Connect to the MySQL database
conn = pymysql.connect(
    host='localhost',
    port=3306,
    user='root',
    password='100128',
    database='ecommerce'
)
cursor = conn.cursor()

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"""

cursor.execute(query)

data = cursor.fetchall()

df = pd.DataFrame(data,columns =["years", "yoy % growth"])
df
```

```
   years  yoy % growth
0   2016           NaN
1   2017  12112.703759
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.

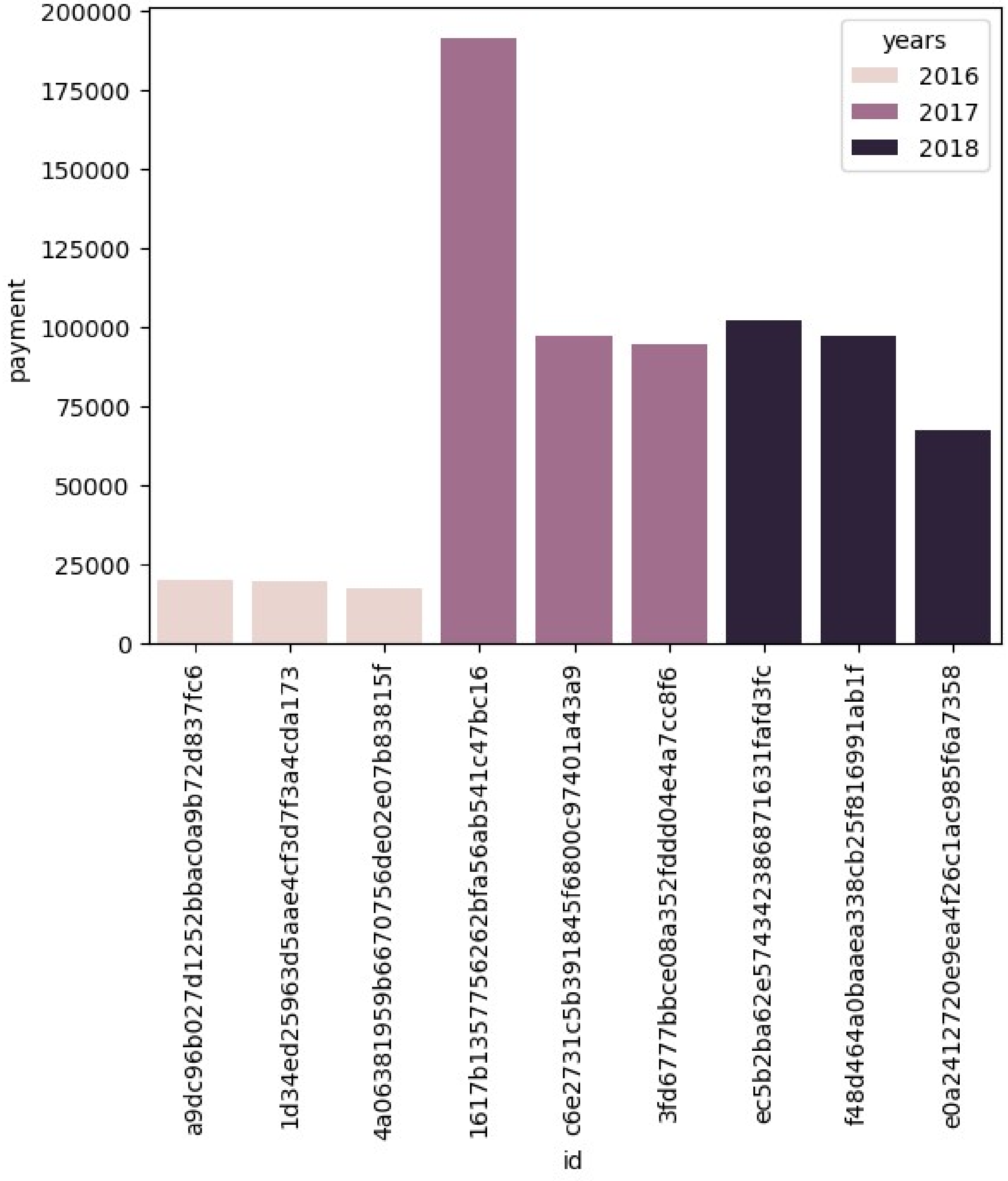
```
import pandas as pd
import pymysql
import os
import  seaborn as sns
# List of CSV files and their corresponding table names
csv_files = [
    ('customers.csv', 'customers'),
    ('orders.csv', 'orders'),
    ('sellers.csv', 'sellers'),
    ('products.csv', 'products'),
    ('geolocation.csv', 'geolocation'),
    ('payments.csv', 'payments'),
    ('order_items.csv','order_items')   # Added payments.csv for
specific handling
]

# Connect to the MySQL database
conn = pymysql.connect(
    host='localhost',
    port=3306,
    user='root',
    password='100128',
    database='ecommerce'
)
cursor = conn.cursor()

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 """

cursor.execute(query)

data = cursor.fetchall()
data
```

```
((None,),)
```

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

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

# List of CSV files and their corresponding table names
csv_files = [
    ('customers.csv', 'customers'),
    ('orders.csv', 'orders'),
    ('sellers.csv', 'sellers'),
    ('products.csv', 'products'),
    ('geolocation.csv', 'geolocation'),
    ('payments.csv', 'payments'),
    ('order_items.csv','order_items')   # Added payments.csv for
specific handling
]

# Connect to the MySQL database
conn = pymysql.connect(
    host='localhost',
    port=3306,
    user='root',
    password='100128',
    database='ecommerce'
)
cursor = conn.cursor()

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 """

cursor.execute(query)

data = cursor.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()
```


payment
200000
175000
150000
125000
100000
75000
50000
25000
0
years
2016
2017
2018
a9dc96b027d1252bbac0a9b72d837fc6
1d34ed25963d5aae4cf3d7f3a4cda173
4a06381959b6670756de02e07b83815f
1617b1357756262bfa56ab541c47bc16
c6e2731c5b391845f6800c97401a43a9
3fd6777bbce08a352fddd04e4a7cc8f6
ec5b2ba62e574342386871631fafd3fc
f48d464a0baaea338cb25f816991ab1f
e0a2412720e9ea4f26c1ac985f6a7358
id