In [1]:
```
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 = "localhost",
                             username = "root",
                             password = "sani_0362",
                             database = "ecommerce")

cur = db.cursor()
```

List all unique cities where customers are located.

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

cur.execute(query)

data = cur.fetchall()

data
df = pd.DataFrame(data, columns = ["Customer_City"])
df.head()
```

Out[2]:

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

Count the number of orders placed in 2017.

```
In [3]: query = """ select count(order_id) from orders where year(order_purchase_timestamp) = 2017 """

        cur.execute(query)

        data = cur.fetchall()

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

```
Out[3]: ('total order placed in 2017 are', 225505)
```

Find the total sales per category.

```
In [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()

        data

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

Out[4]:

	Category	Sales
0	ART	24080.28
1	COOL STUFF	463317.96
2	GAMES CONSOLES	214927.56
3	TELEPHONY	506025.24
4	SPORT LEISURE	1102358.88
...	...	...
65	ARTS AND CRAFTS	4084.92
66	CDS MUSIC DVDS	899.28
67	CITTE AND UPHACK FURNITURE	5981.88
68	KITCHEN PORTABLE AND FOOD COACH	1244.16
69	FASHION SPORT	982.68

70 rows × 2 columns

Calculate the percentage of orders that were paid in installments.

In [5]:

```
query = """ select (sum(case when payments.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[5]:

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

Count the number of customers from each state.

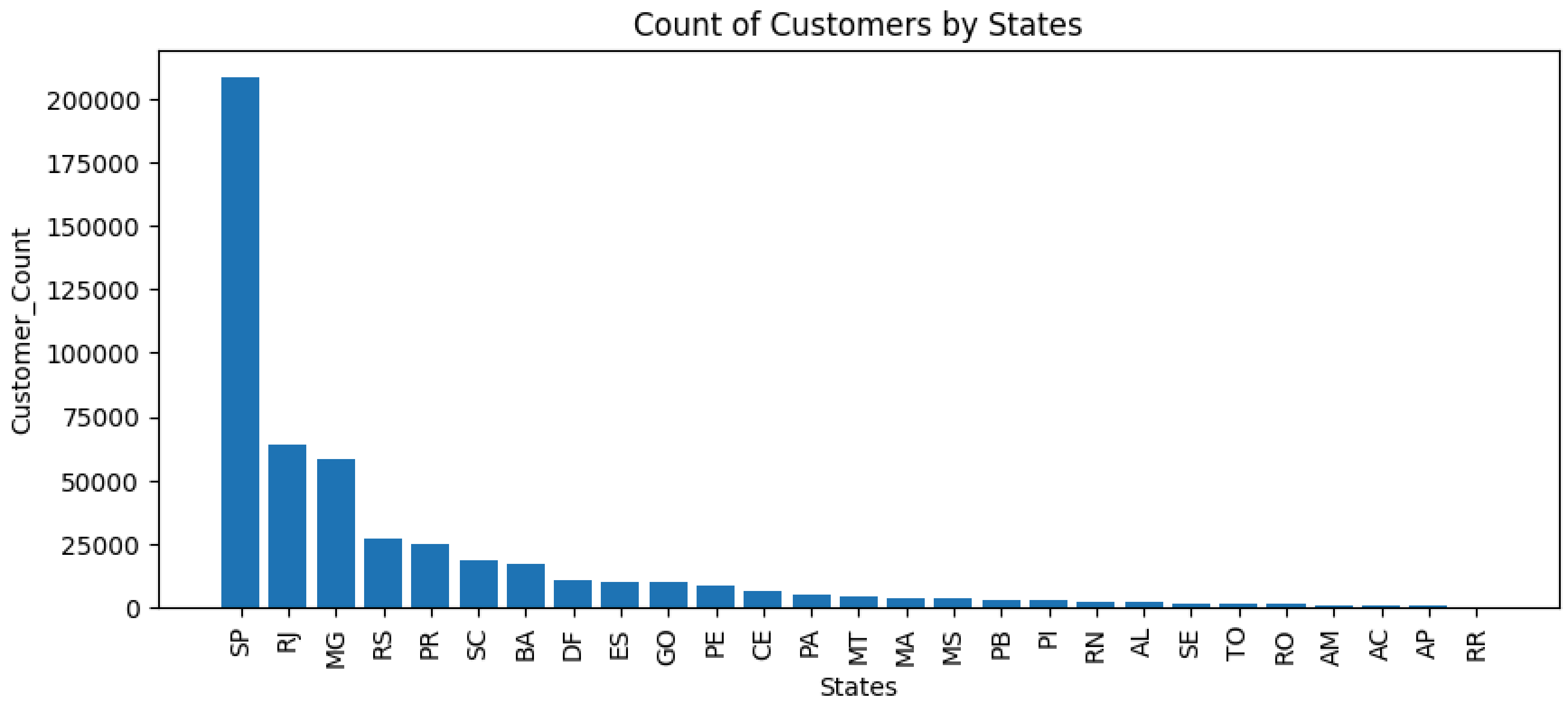
```
In [6]: query = """ select customer_state, count(customer_id) from customers
        group by customer_state"""

        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= (10,4))
        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()
```



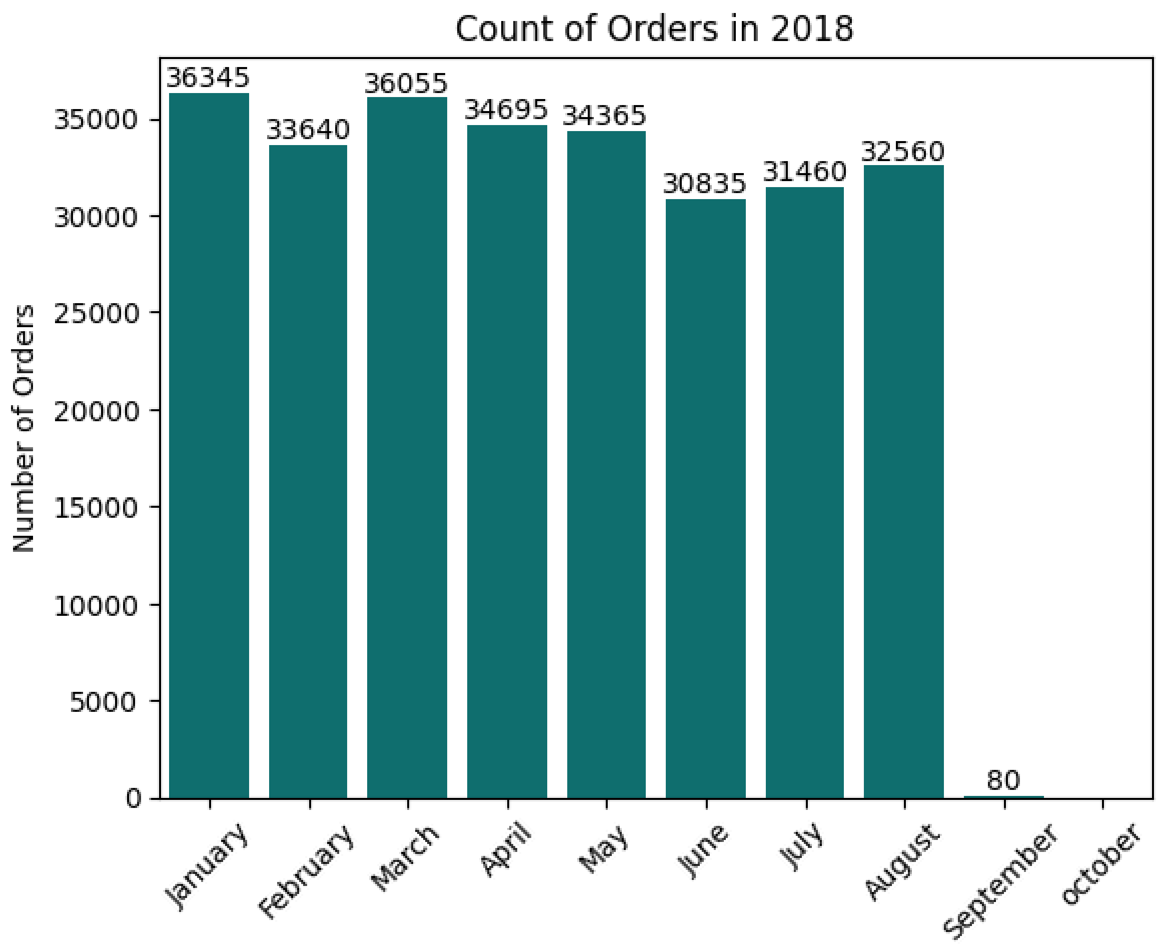
Calculate the number of orders per month in 2018.

```
In [7]: query = """ select monthname(order_purchase_timestamp) months,count(order_id) from orders
where year(order_purchase_timestamp)= 2018
group by months"""

cur.execute(query)

data = cur.fetchall()
data
df = pd.DataFrame(data, columns = ["Months","Number of Orders"])
df
o = ["January","February","March","April","May","June","July","August","September","october"]
plt.xticks(rotation = 45)
```

```
ax = sns.barplot(x = df["Months"], y =df["Number of Orders"] , order= o, color = "teal")
ax.bar_label(ax.containers[0])
plt.title("Count of Orders in 2018")
plt.show()
```



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

```
In [8]: query = """ with Count_Per_Order as
        (select orders.order_id, orders.customer_id, count(order_items.order_id) as order_count
        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.order_count),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()
        data
        df = pd.DataFrame(data, columns = ["customer_city","avg_products_per_orders"])
        df.head(10)
```

Out[8]:

	customer_city	avg_products_per_orders
0	vilhena	20.00
1	tiradentes	20.00
2	ronda alta	15.00
3	imbe	15.00
4	santana	15.00
5	vacaria	15.00
6	sao raimundo nonato	15.00
7	pires do rio	15.00
8	sao joao nepomuceno	15.00
9	passo fundo	12.50

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

In [9]:
```
query = """ select 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()

data
df = pd.DataFrame(data, columns = ["Category","Percentage_Distribution"])
df.head()
```

Out[9]:

	Category	Percentage_Distribution
0	bed table bath	2.98
1	fixed telephony	2.82
2	computer accessories	2.66
3	HEALTH BEAUTY	2.50
4	sport leisure	2.30

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

In [10]:
```
query = """ select products.product_category, count(order_items.product_id) as count_of_products,
round(avg(order_items.price),2) as avg_price from products
join order_items
on products.product_id = order_items.product_id
group by products.product_category, order_items.price order by count_of_products desc """

cur.execute(query)

data = cur.fetchall()

data
df = pd.DataFrame(data, columns= ["Category","Order_Count","Price"])
df

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", a[0][1])
```

```
The correlation between product price and the number of times a product has been purchased -0.08773617051577573
```

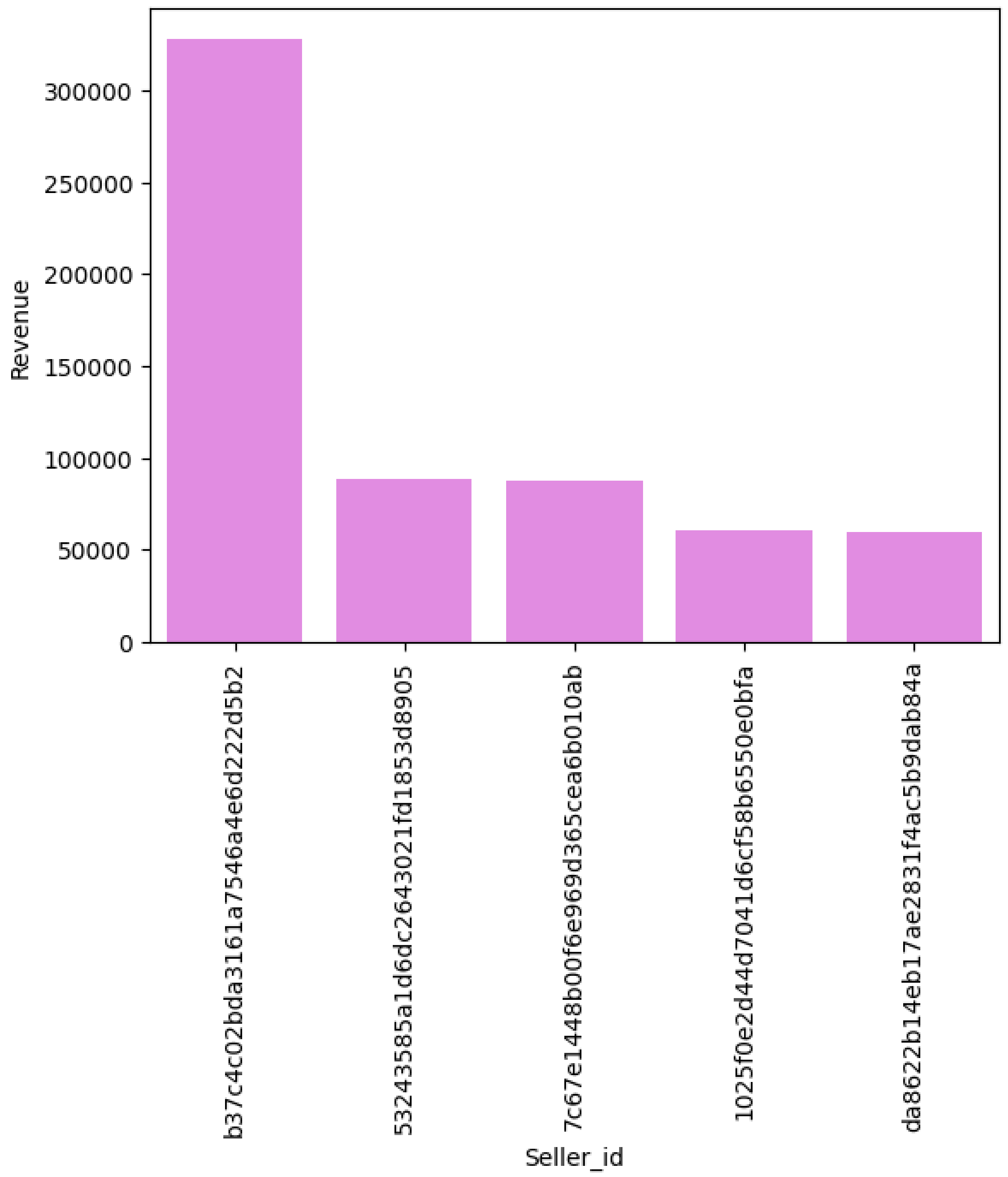
Calculate the total revenue generated by each seller, and rank them by revenue.

In [11]:
```
query = """ select * , dense_rank() over(order by revenue desc) as ranks from
(select order_items.seller_id, round(sum(payments.payment_value),2) as 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()
data
df = pd.DataFrame(data, columns = ["Seller_id","Revenue","Rank"])
sns.barplot(x= "Seller_id", y ="Revenue", data= df.head(), color = "violet")
```

```
plt.xticks(rotation =90)
plt.show()
```



Revenue
300000
250000
200000
150000
100000
50000
0
b37c4c02bda3161a7546a4e6d222d5b2
53243585a1d6dc2643021fd1853d8905
7c67e1448b00f6e969d365cea6b010ab
1025f0e2d44d7041d6cf58b6550e0bfa
da8622b14eb17ae2831f4ac5b9dab84a
Seller_id

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

```
In [12]: query = """ select customer_id, order_purchase_timestamp,
avg(payment) over (partition by customer_id order by order_purchase_timestamp
rows between 2 preceding and current row) as moving_average
from
(select orders.customer_id, orders.order_purchase_timestamp, payments.payment_value as payment
from orders join payments
on orders.order_id = payments.order_id) as a"""

cur.execute(query)

data = cur.fetchall()
data
df = pd.DataFrame(data, columns = ["Customer_id","Order_History","Moving_Average"])
df.head(10)
```

Out[12]:

	Customer_id	Order_History	Moving_Average
0	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.739998
1	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.739998
2	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.739998
3	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.739998
4	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.739998
5	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.739998
6	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.739998
7	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.739998
8	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.739998
9	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.739998

Calculate the cumulative sales per month for each year.

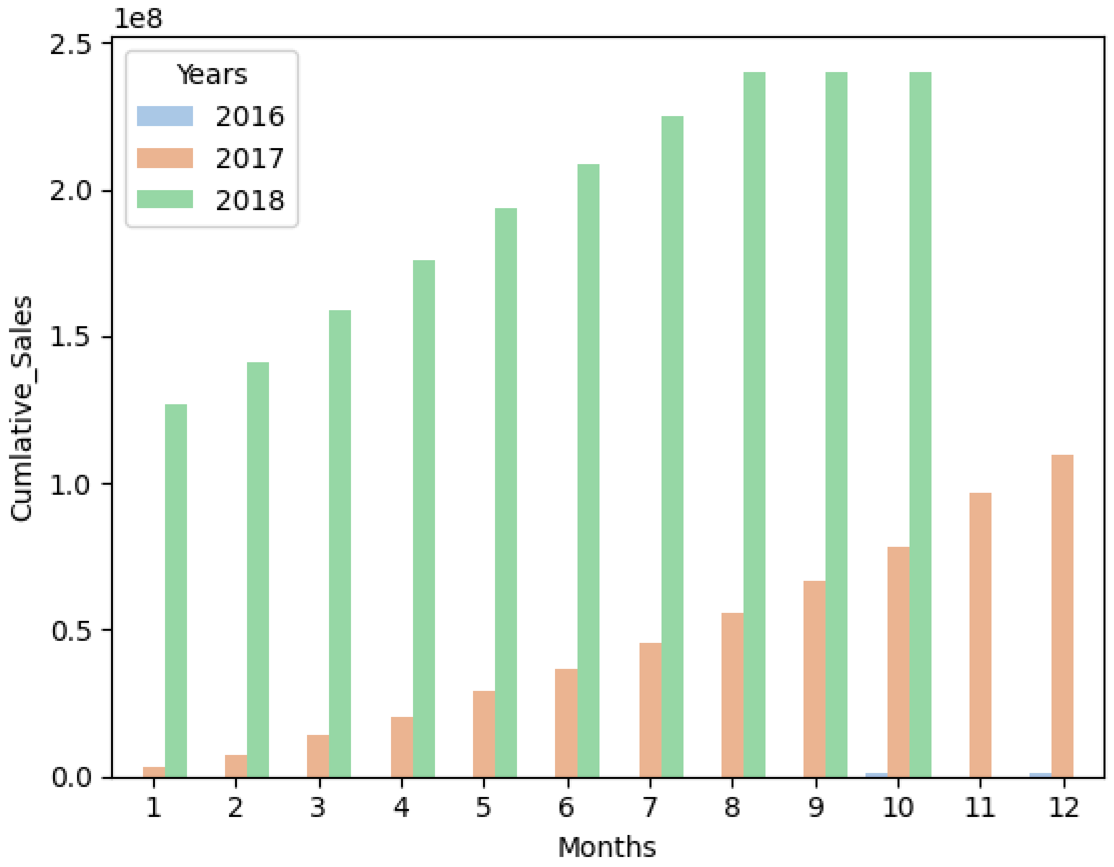
In [13]:
```
query = """ select months, years, payment, sum(payment)
over(order by years, months) as cumlative_sales
from
(select month(orders.order_purchase_timestamp) as months,
year(orders.order_purchase_timestamp) as years,
round(sum(payments.payment_value)) as payment
from payments join orders
on orders.order_id = payments.order_id
group by months, years order by years) as a; """

cur.execute(query)

data = cur.fetchall()
data
```

```
df = pd.DataFrame(data, columns = ["Months","Years","Payments","Cumlative_Sales"])
df

sns.barplot(data = df, x = df["Months"], y= df["Cumlative_Sales"], hue="Years", palette = "pastel")
plt.show()
```



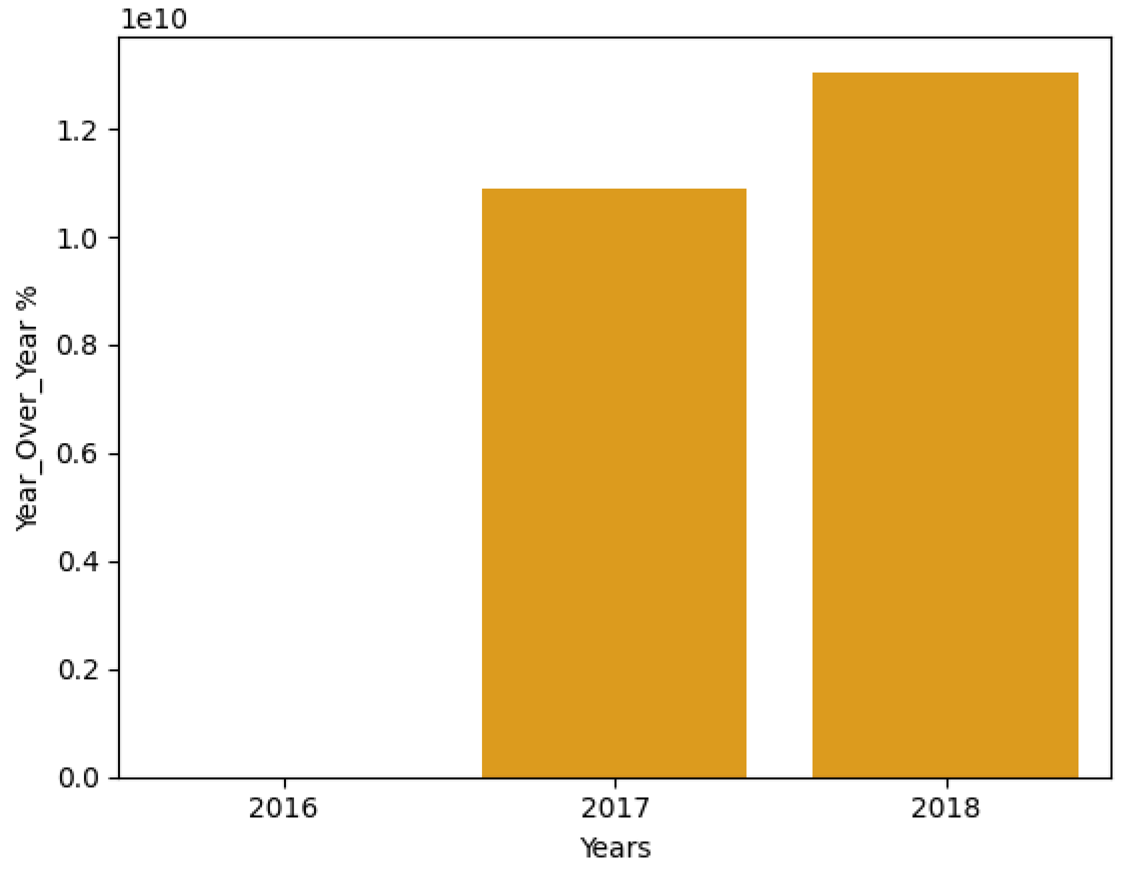
Calculate the year-over-year growth rate of total sales.

In [14]:
```
query = """select years, payment,
((payment -lag(payment, 1) over (order by years)
/lag(payment,1) over (order by years))* 100) as year_over_year
```

```
from
(select year(orders.order_purchase_timestamp) as years,
round(sum(payments.payment_value),2) as payment from payments join orders
on orders.order_id = payments.order_id
group by years order by years) as a"""
cur.execute(query)

data = cur.fetchall()
data

df = pd.DataFrame(data, columns = ["Years","Payments","Year_Over_Year %"])
df
sns.barplot(data = df, x = df["Years"], y = df["Year_Over_Year %"], color ="orange")
plt.show()
```



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

In [15]:
```
query = """ with a as (select customers.customer_id, min(orders.order_purchase_timestamp) as 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) as next_order
from a join orders
on a.customer_id = orders.customer_id
and orders.order_purchase_timestamp > first_order
and orders.order_purchase_timestamp < date_add(first_order, interval 24 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
```

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

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

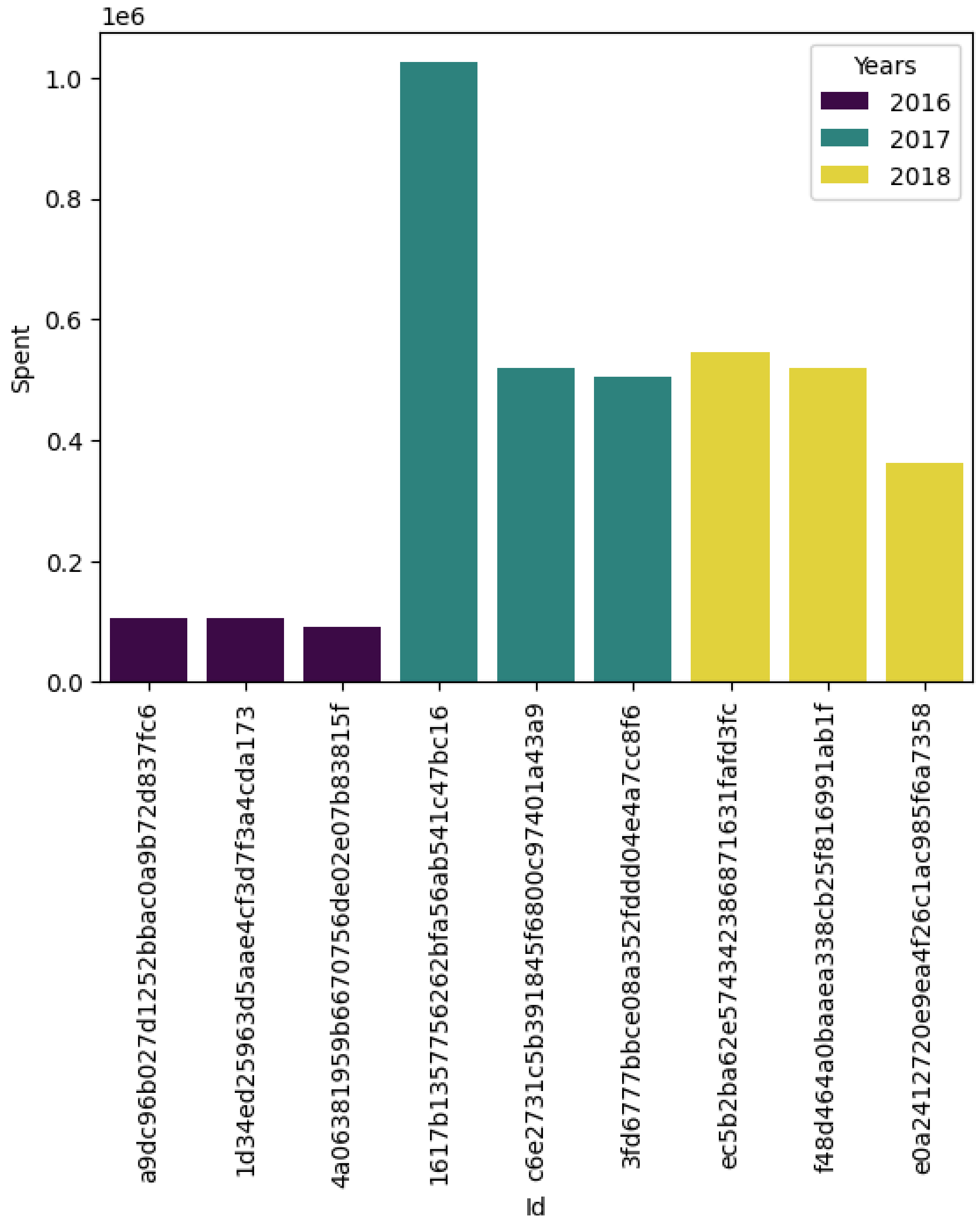
if customer_id is taken from customers table

In [16]:
```
query =""" select a.id, a.years, a.spent, a.ranks
from
(select customers.customer_id as id, year(orders.order_purchase_timestamp) as years,
sum(payments.payment_value) as spent,
dense_rank() over (partition by year(orders.order_purchase_timestamp) order by sum(payments.payment_value) desc) as ranks
from customers join orders
on customers.customer_id = orders.customer_id
join payments
on payments.order_id = orders.order_id
group by customers.customer_id, years) as a
where ranks<=3 """

cur.execute(query)

data = cur.fetchall()
```

```
data
df = pd.DataFrame(data, columns = ["Id","Years","Spent","Ranks"])
df
ax = sns.barplot(data = df, x=df["Id"], y = df["Spent"], hue = "Years", palette = "viridis")
plt.xticks(rotation = 90)
plt.show()
```

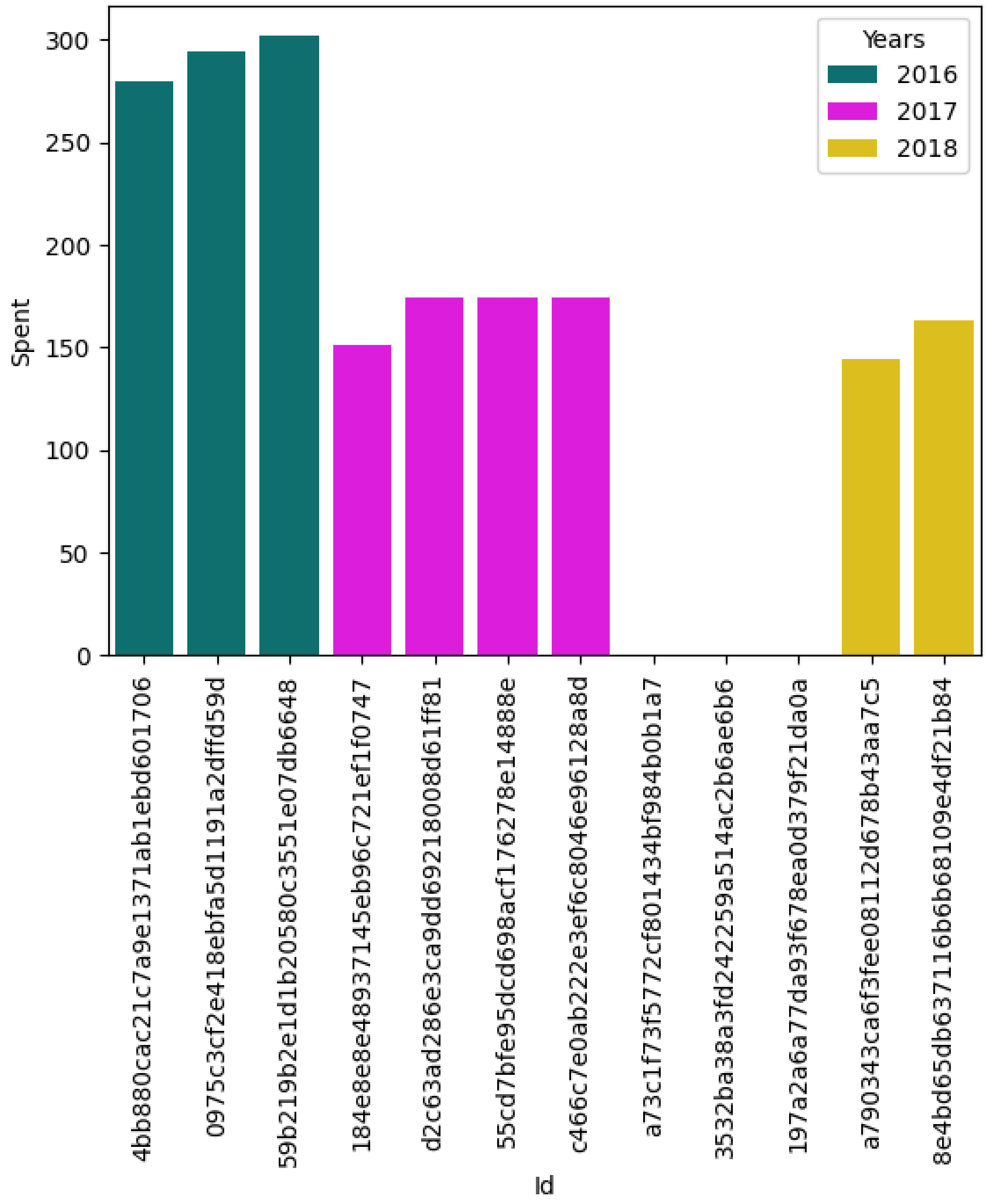

1e6
1.0
0.8
0.6
0.4
0.2
0.0
Spent
Years
2016
2017
2018
a9dc96b027d1252bbac0a9b72d837fc6
1d34ed25963d5aae4cf3d7f3a4cda173
4a06381959b6670756de02e07b83815f
1617b1357756262bfa56ab541c47bc16
c6e2731c5b391845f6800c97401a43a9
3fd6777bbce08a352fddd04e4a7cc8f6
ec5b2ba62e574342386871631fafd3fc
f48d464a0baaea338cb25f816991ab1f
e0a2412720e9ea4f26c1ac985f6a7358
Id

if customer_id is taken from orders table

```
In [17]: query =""" select a.id, a.years, a.spent, a.ranks
from
(select orders.customer_id as id, year(orders.order_purchase_timestamp) as years,
sum(payments.payment_value) as spent,
dense_rank() over (partition by year(orders.order_purchase_timestamp) order by sum(payments.payment_value)) as ranks
from orders join payments
on orders.order_id = payments.order_id
group by orders.customer_id, years) as a
where ranks <=3; """

cur.execute(query)

data = cur.fetchall()
data
df = pd.DataFrame(data, columns = ["Id","Years","Spent","Ranks"])
df
ax = sns.barplot(data = df, x=df["Id"], y = df["Spent"], hue = "Years", palette = ["teal","magenta","gold"])
plt.xticks(rotation = 90)
plt.show()
```


Years
2016
2017
2018
Spent
300
250
200
150
100
50
0
Id
4bb880cac21c7a9e1371ab1ebd601706
0975c3cf2e418ebfa5d1191a2dffd59d
59b219b2e1d1b20580c3551e07db6648
184e8e8e48937145eb96c721ef1f0747
d2c63ad286e3ca9dd69218008d61ff81
55cd7bfe95dcd698acf176278e14888e
c466c7e0ab222e3ef6c8046e96128a8d
a73c1f73f5772cf801434bf984b0b1a7
3532ba38a3fd242259a514ac2b6ae6b6
197a2a6a77da93f678ea0d379f21da0a
a790343ca6f3fee08112d678b43aa7c5
8e4bd65db637116b6b68109e4df21b84