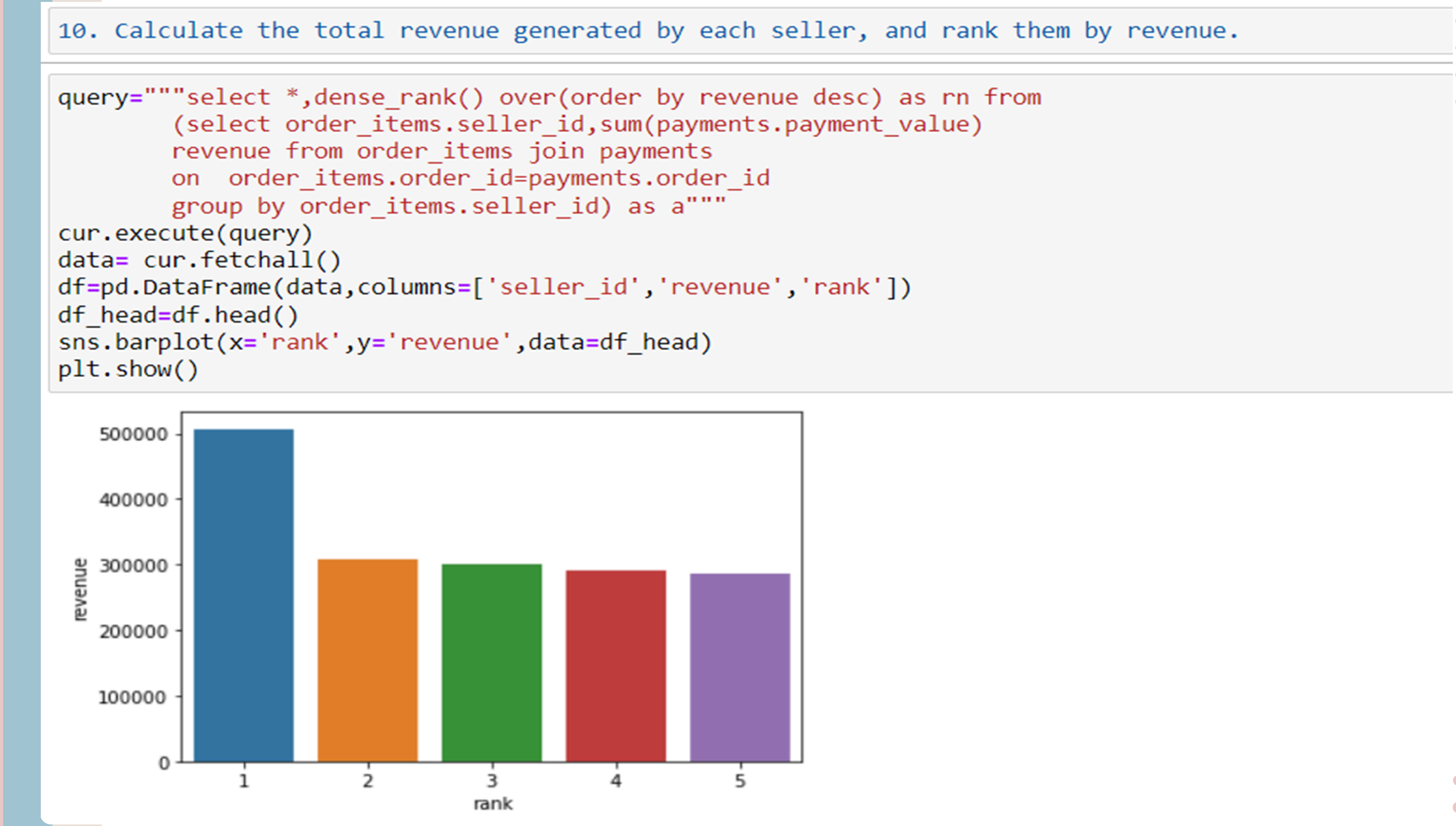
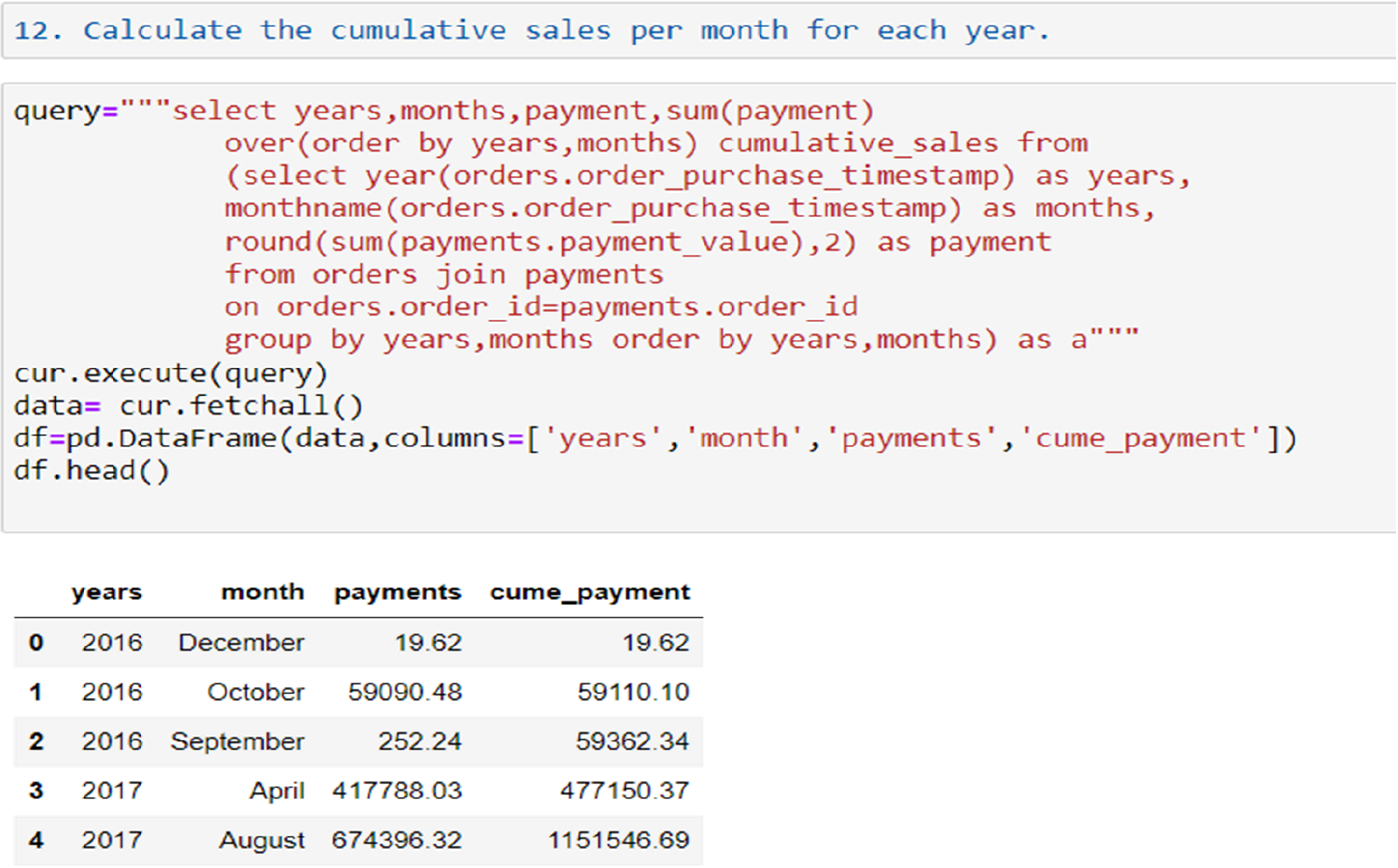
Total revenue generated by top 5 Seller



10. 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','rank'])
df_head=df.head()
sns.barplot(x='rank',y='revenue',data=df_head)
plt.show()
```

Cummelative sum of sales on monthly basis

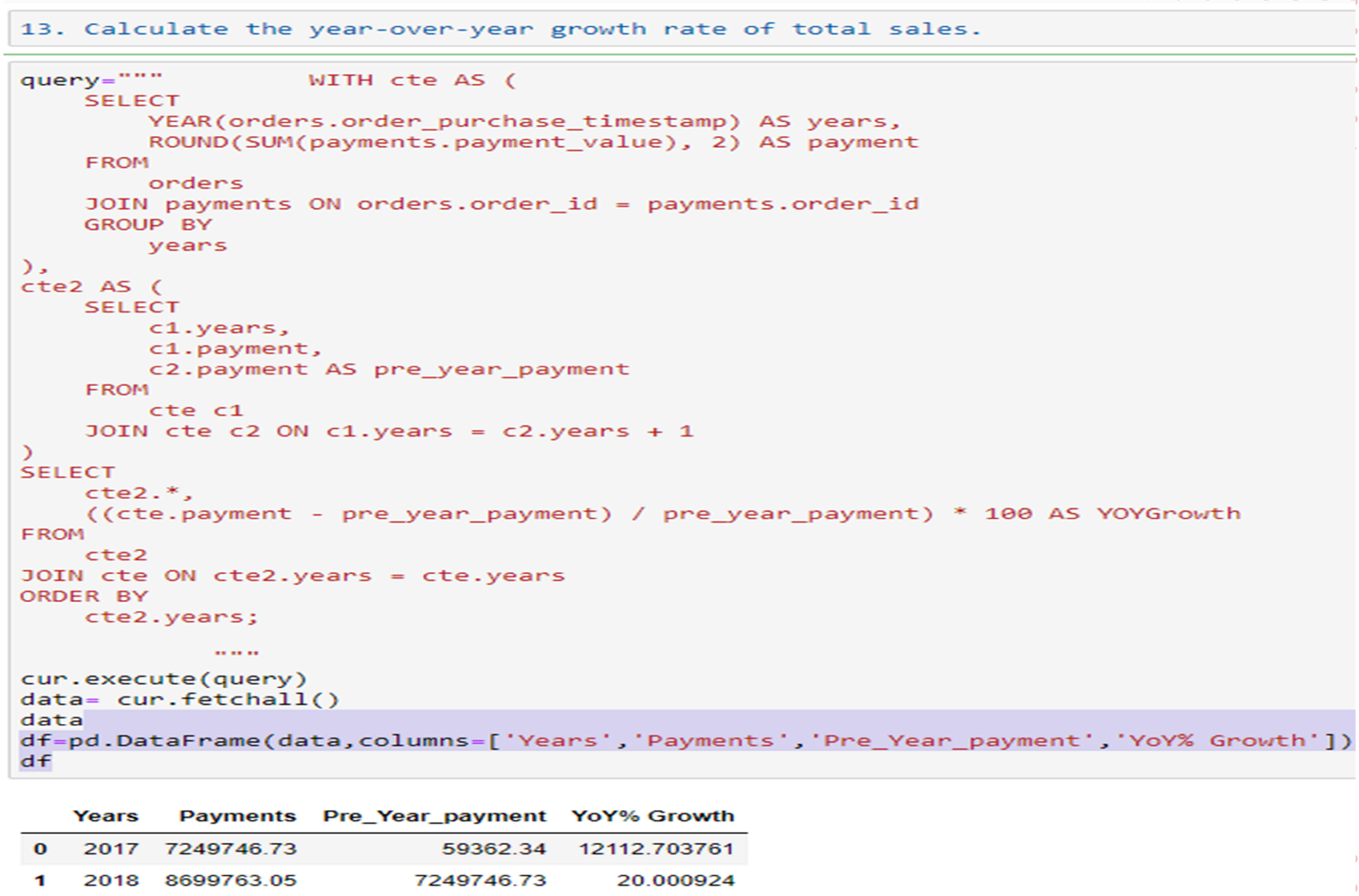


12. Calculate the cumulative sales per month for each year.

```
query="""select years,months,payment,sum(payment)
            over(order by years,months) cumulative_sales from
            (select year(orders.order_purchase_timestamp) as years,
            monthname(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','month','payments','cume_payment'])
df.head()
```

	years	month	payments	cume_payment
0	2016	December	19.62	19.62
1	2016	October	59090.48	59110.10
2	2016	September	252.24	59362.34
3	2017	April	417788.03	477150.37
4	2017	August	674396.32	1151546.69

Year on Year Sales Growth



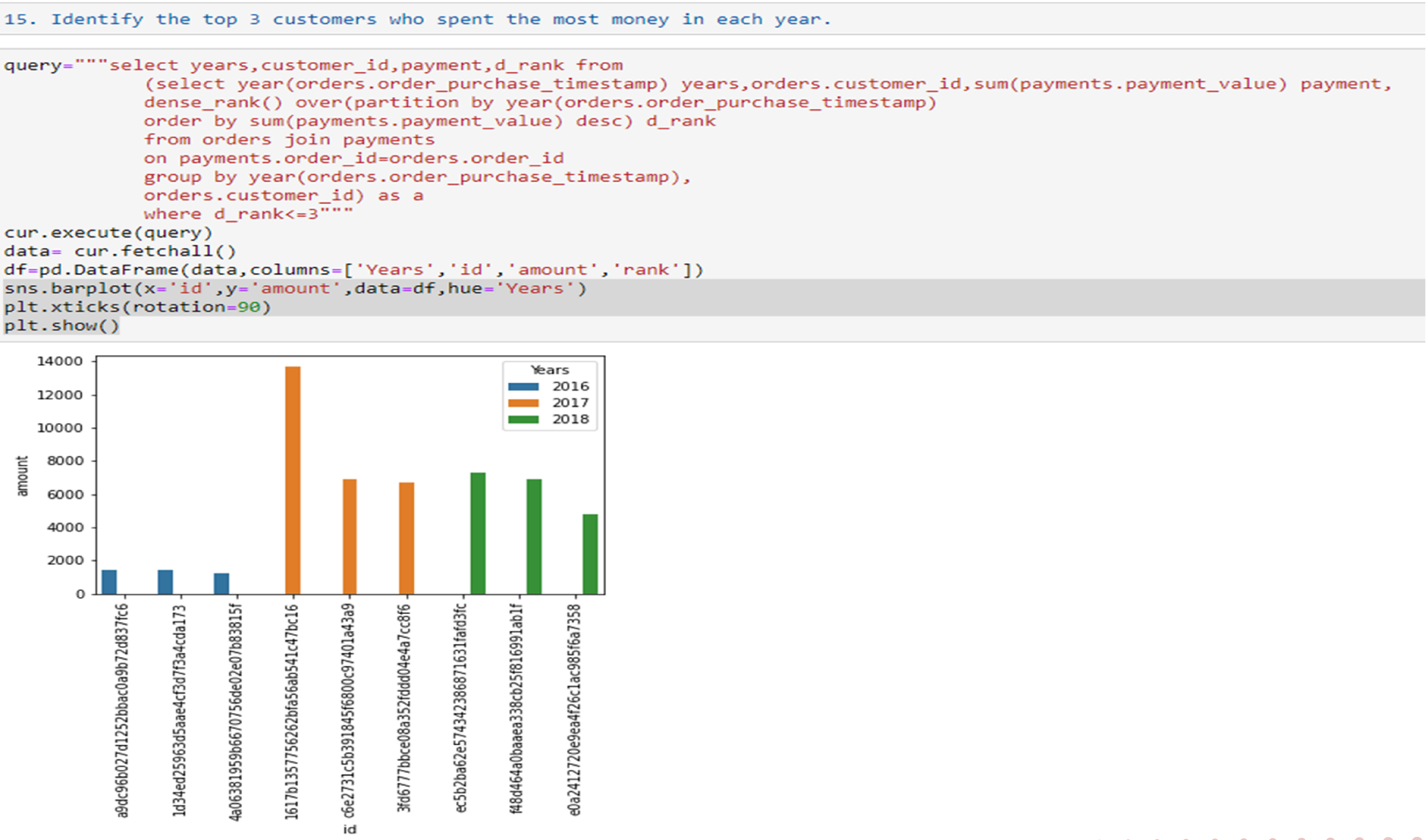
13. Calculate the year-over-year growth rate of total sales.

```
query="""           WITH cte 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
),
cte2 AS (
    SELECT
        c1.years,
        c1.payment,
        c2.payment AS pre_year_payment
    FROM
        cte c1
    JOIN cte c2 ON c1.years = c2.years + 1
)
SELECT
    cte2.*,
    ((cte.payment - pre_year_payment) / pre_year_payment) * 100 AS YOYGrowth
FROM
    cte2
JOIN cte ON cte2.years = cte.years
ORDER BY
    cte2.years;

            """
cur.execute(query)
data= cur.fetchall()
data
df=pd.DataFrame(data,columns=['Years','Payments','Pre_Year_payment','YoY% Growth'])
df
```

	Years	Payments	Pre_Year_payment	YoY% Growth
0	2017	7249746.73	59362.34	12112.703761
1	2018	8699763.05	7249746.73	20.000924

Top 3 Customer on Yearly Basis



15. 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=['Years','id','amount','rank'])
sns.barplot(x='id',y='amount',data=df,hue='Years')
plt.xticks(rotation=90)
plt.show()
```

Years
2016
2017
2018
amount
14000
12000
10000
8000
6000
4000
2000
0
a9dc96b027d1252bbac0a9b72d837fc6
1d34ed25963d5aae4cf3d7f3a4cda173
4a06381959b6670756de02e07b83815f
1617b1357756262bfa56ab541c47bc16
c6e2731c5b391845f6800c97401a43a9
3fd6777bbce08a352fddd04e4a7cc8f6
ec5b2ba62e574342386871631fafd3fc
f48d464a0baaea338cb25f816991ab1f
e0a2412720e9ea4f26c1ac985f6a7358
id