In [51]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as mtn
import seaborn as sns
import psycopg2
```

In [52]:
```python
db = psycopg2.connect(
    host='localhost',
    user='postgres',
    password='SQUAL#QUE',
    database='E-commerce'
)

cur = db.cursor()
```

1. List all unique cities where customers are located.

In [41]:
```python
query = """ select distinct(customer_city) from customers """
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ["Customer_cities"])
df
```

Out[41]:

	Customer_cities
0	bom jardim de minas
1	alto rio doce
2	alvorada do gurgueia
3	batatais
4	capao da porteira
...	...
4114	carbonita
4115	concordia do para
4116	independencia
4117	governador valadares
4118	balsa nova

4119 rows × 1 columns

2. Count the number of orders placed in 2017.

In [42]:
```python
query = """ select count(extract(year from cast(order_booked as TIMESTAMP))) AS years from orders
where extract(year from cast(order_booked as TIMESTAMP)) = 2017 """
cur.execute(query)
data = cur.fetchall()
('total order placed in 2017 is', data[0][0])
```

Out[42]: ('total order placed in 2017 is', 90202)

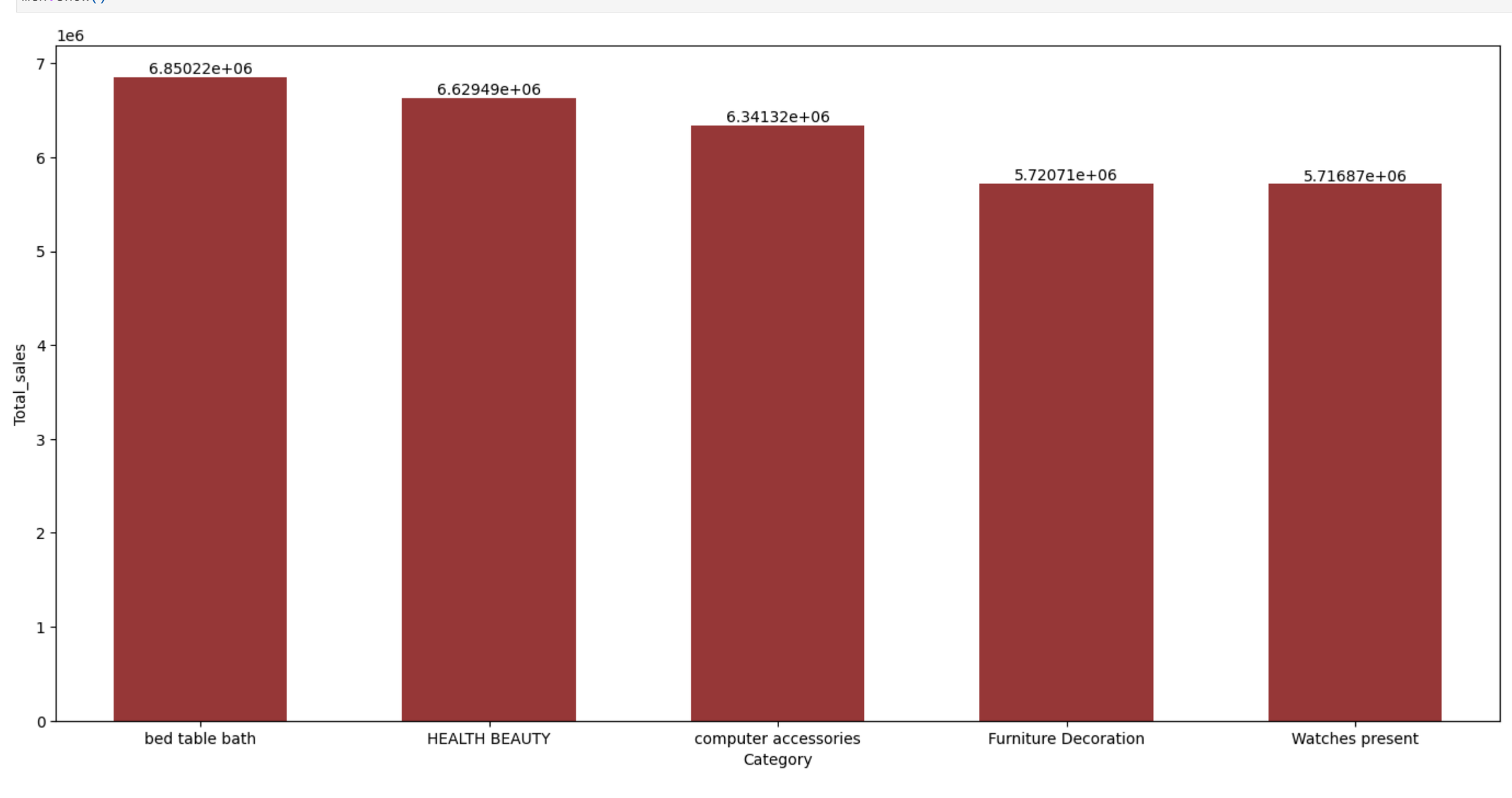
3. Find the Top 5 Category by Sales.

In [43]:
```python
query = """ select p.product_category, cast(sum(ps.payment_value) as Integer) as Total_sales from products AS p
join order_items as o
ON p.product_id = o.product_id
join payments as ps
ON o.order_id = ps.order_id
group by 1
order by 2 desc
limit 5"""
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ["Category", "Total_sales"])
df
```

Out[43]:

	Category	Total_sales
0	bed table bath	6850215
1	HEALTH BEAUTY	6629492
2	computer accessories	6341322
3	Furniture Decoration	5720706
4	Watches present	5716867

In [44]:
```python
mtn.figure(figsize = (17,8))
ax = sns.barplot(x = "Category", y = "Total_sales", data = df, width = 0.6, color = "Brown")
ax.bar_label(ax.containers[0])
mtn.show()
```



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

In [48]:
```python
query = """ select round((sum(case when payment_installments >= 1 then 1 else 0 end*1.0)/Count(*))*100,3) as intallments
from payments """
cur.execute(query)
data = cur.fetchall()
"The Percentage Of Orders That were paid in installments is ", data[0][0]
```

Out[48]: ('The Percentage Of Orders That were paid in installments is ',
 Decimal('99.998'))

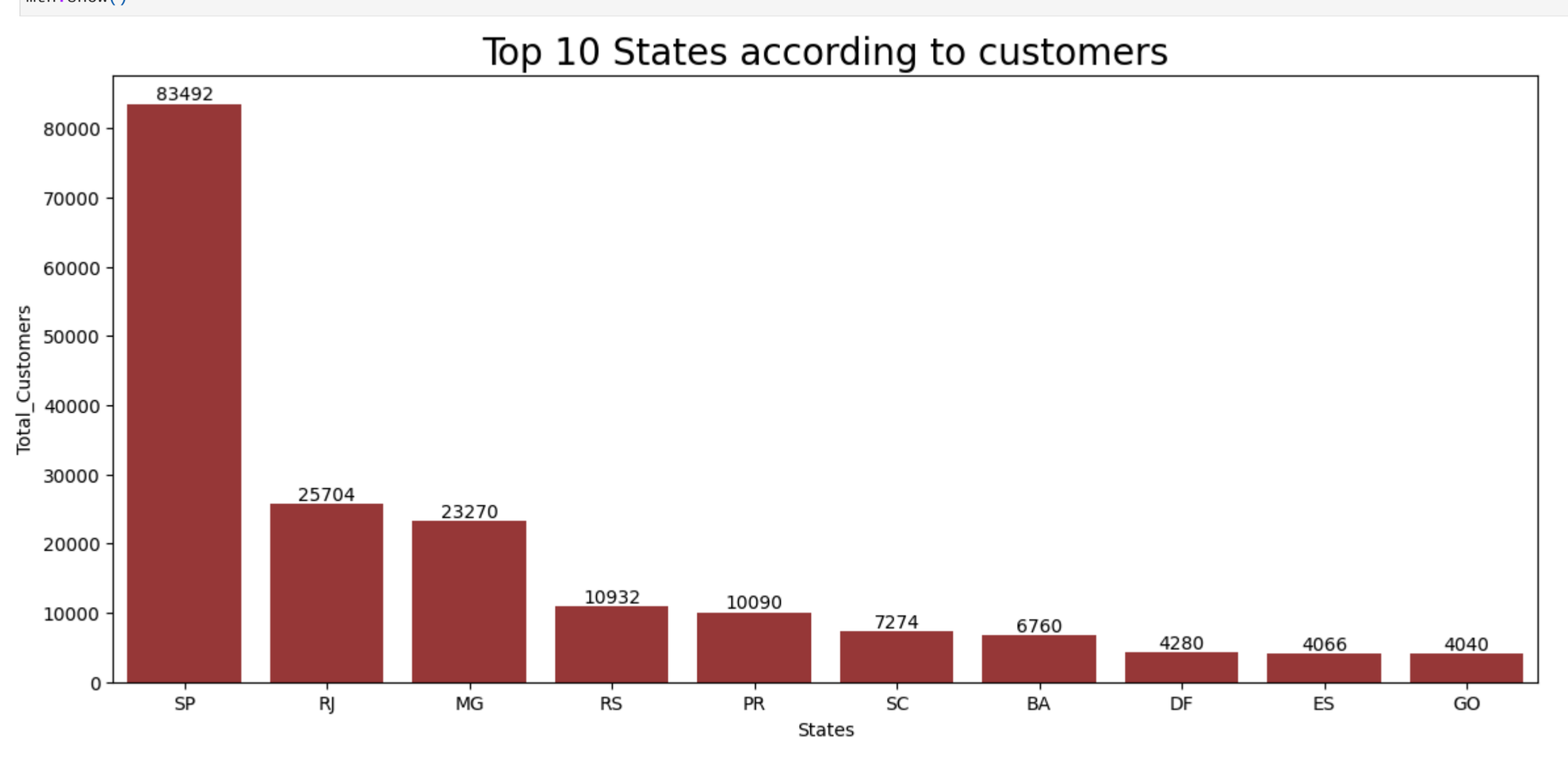
5. Top 10 States on the behalf of Customers

In [37]:
```python
query =  """ select customer_state, count(*) AS Total_customers from customers
group by 1
order by 2 desc
limit 10 """
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ["States", "Total_Customers"])
df
```

Out[37]:

	States	Total_Customers
0	SP	83492
1	RJ	25704
2	MG	23270
3	RS	10932
4	PR	10090
5	SC	7274
6	BA	6760
7	DF	4280
8	ES	4066
9	GO	4040

In [40]:
```python
mtn.figure(figsize = (14,6))
ax = sns.barplot(x = "States", y = "Total_Customers", data = df, color = "Brown")
ax.bar_label(ax.containers[0])
mtn.title("Top 10 States according to customers", fontsize = 20,)
mtn.show()
```



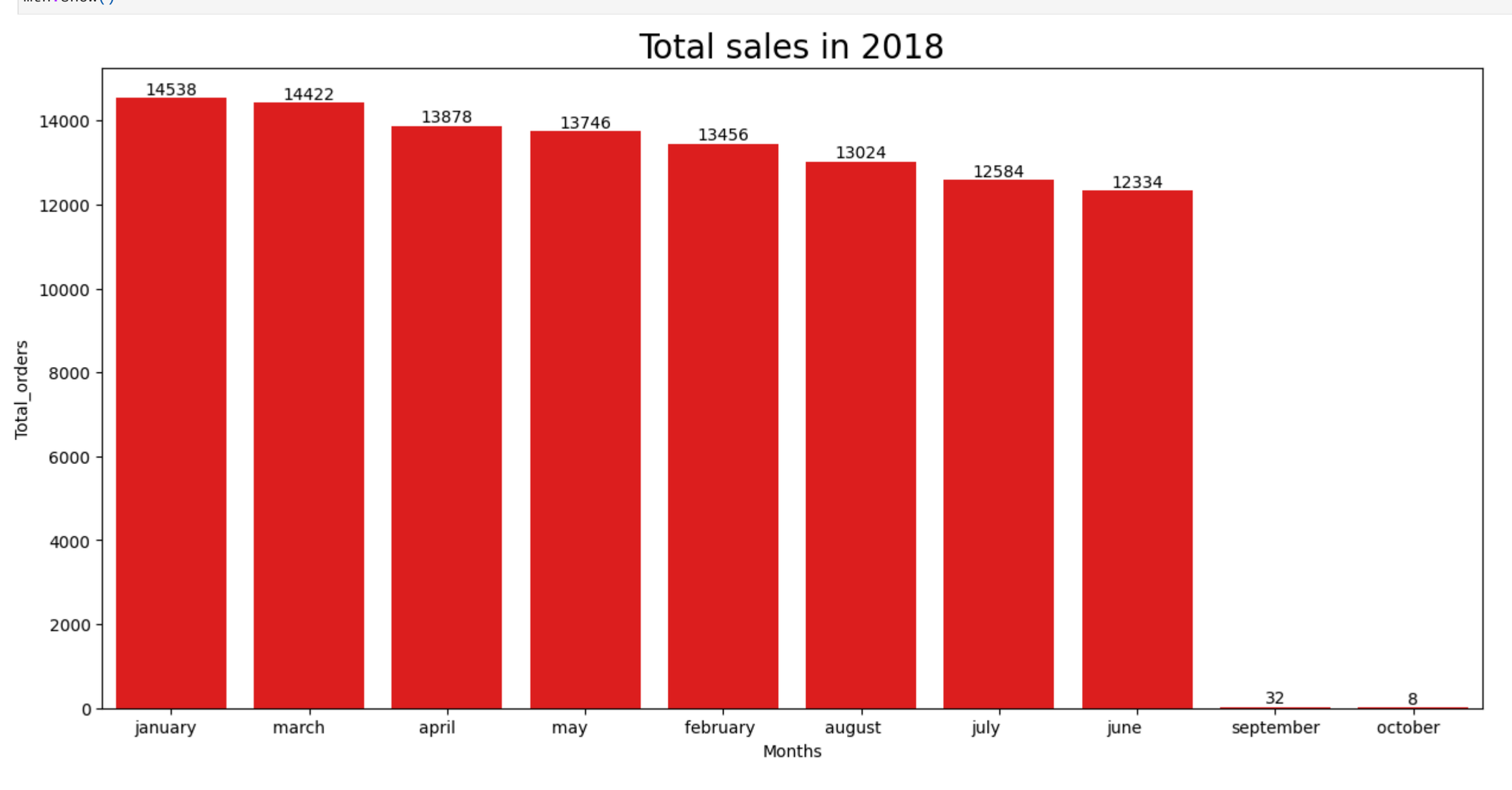
6. Calculate the number of orders per month in 2018.

In [42]:
```python
query = """ select extract(year from cast(order_booked as TIMESTAMP)) AS years, To_char(cast(order_booked as TIMESTAMP), 'month') AS MOnths,
count(*) as TOTAL_orders from orders
where extract(year from cast(order_booked AS TIMESTAMP)) = 2018
group by 1,2
order by 3  desc"""
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ["Year","Months", "Total_orders"])
df
```

Out[42]:

	Year	Months	Total_orders
0	2018	january	14538
1	2018	march	14422
2	2018	april	13878
3	2018	may	13746
4	2018	february	13456
5	2018	august	13024
6	2018	july	12584
7	2018	june	12334
8	2018	september	32
9	2018	october	8

In [45]:
```python
mtn.figure(figsize = (15,7))
ax = sns.barplot(x = "Months", y = "Total_orders", data = df, color = "r")
ax.bar_label(ax.containers[0])
mtn.title("Total sales in 2018", fontsize = 20)
mtn.show()
```



January 2018 got the highest number of Orders

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

In [72]:
```python
query = """ with one as (select o.order_id, o.customer_id, Count(og.order_id) as Total_orders from orders as o
join order_items as og
ON o.order_id = og.order_id
group by 1,2
order by 3)
select c.customer_city, round(avg(one.total_orders),2) AS avg_per_order from customers as c
join one
ON c.customer_id = one.customer_id
group by 1 """
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ["Customer_city", "Avg_Orders"])
df
```

Out[72]:

	Customer_city	Avg_Orders
0	bom jardim de minas	2.75
1	alto rio doce	2.00
2	alvorada do gurgueia	2.00
3	batatais	2.21
4	capao da porteira	2.00
...	...	...
4105	concordia do para	2.00
4106	carbonita	2.00
4107	independencia	2.00
4108	governador valadares	2.19
4109	balsa nova	2.00

4110 rows × 2 columns

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

In [74]:
```python
query = """ select product_category, round(cast((total_revenue/(select sum(payment_value) from payments))*100 AS Decimal),2) as Total_percn from
(select p.product_category, CAST(sum(pm.payment_value) AS integer) as Total_revenue from products AS p
join order_items as o
ON p.product_id = o.product_id
join payments as pm
ON o.order_id = pm.order_id
group by 1
order by 2 desc)as k"""
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ["Category", "Sales%"])
df
```

Out[74]:

	Category	Sales%
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
...	...	...
69	PC Gamer	0.03
70	House Comfort 2	0.02
71	cds music dvds	0.01
72	Fashion Children's Clothing	0.01
73	insurance and services	0.00

74 rows × 2 columns

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

In [77]:
```python
query = """ select p.product_category, round(cast(avg(og.price) AS decimal),2) as avg_price, count(og.order_id) as Total_orders from products as p
join order_items as og
ON og.product_id = p.product_id
group by 1
order by 3 desc """
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ["Category", "Price","Total_orders"])
df
```

Out[77]:

	Category	Price	Total_orders
0	bed table bath	93.30	22230
1	HEALTH BEAUTY	130.16	19340
2	sport leisure	114.34	17282
3	Furniture Decoration	87.56	16668
4	computer accessories	116.51	15654
...	...	...	...
69	cds music dvds	52.14	28
70	La Cuisine	146.79	28
71	PC Gamer	171.77	18
72	Fashion Children's Clothing	71.23	16
73	insurance and services	141.65	4

74 rows × 3 columns

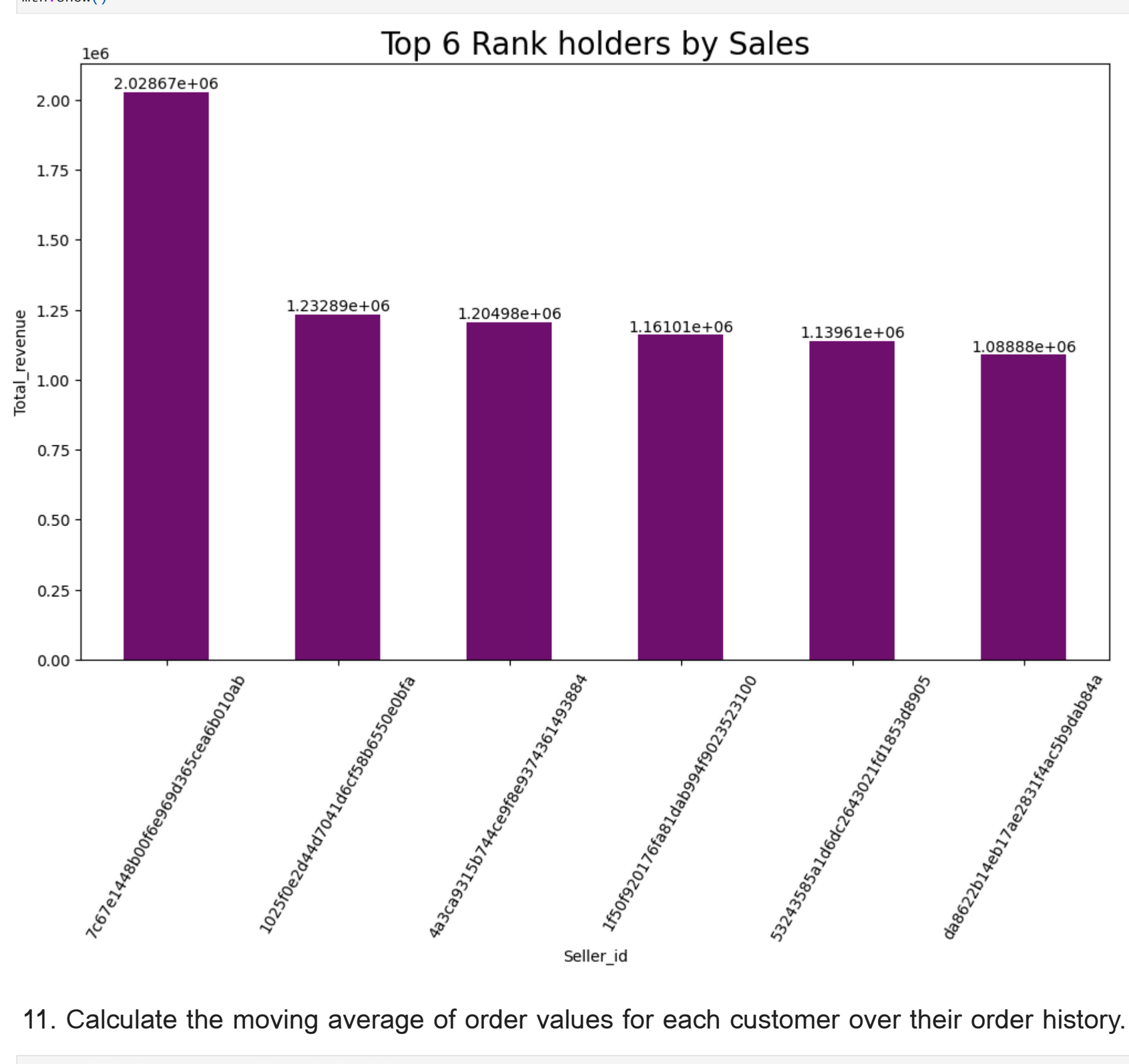
10. Identify the Top 6 Rank holders by total sales

In [81]:
```python
query = """ select *, rank()over(order by total_revenue desc) AS rank from
(select s.seller_id, CAST(sum(p.payment_value) as INTEGER) AS Total_revenue from sellers as s
join order_items as o
ON s.seller_id = o.seller_id
join payments as p
ON o.order_id = p.order_id
group by 1)as l
limit 6"""
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ["Seller_id", "Total_revenue","Rank"])
df
```

Out[81]:

	Seller_id	Total_revenue	Rank
0	7c67e1448b00f6e969d365cea6b010ab	2028668	1
1	1025f0e2d44d7041d6cf58b6550e0bfa	1232888	2
2	4a3ca9315b744ce9f8e9374361493884	1204981	3
3	1f50f920176fa81dab994f9023523100	1161014	4
4	53243585a1d6dc2643021fd1853d8905	1139612	5
5	da8622b14eb17ae2831f4ac5b9dab84a	1088877	6

In [95]:
```python
mtn.figure(figsize = (12,7))
ax = sns.barplot(x = "Seller_id", y = "Total_revenue", data = df, width = 0.5, color = "#800080")
ax.bar_label(ax.containers[0])
mtn.title("Top 6 Rank holders by Sales", fontsize = 20)
mtn.xticks(rotation = 60)
mtn.show()
```



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

In [47]:
```python
query = """select customer_id, order_booked, payment,
avg(payment)over(partition by customer_id order by order_booked rows between 2 preceding and current row) from
(select o.customer_id, o.order_booked, p.payment_value as payment from orders as o
join payments as p
ON o.order_id = p.order_id)as l"""
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ["Customer_id", "Date","Payment","Moving_avg"])
df
```

Out[47]:

	Customer_id	Date	Payment	Moving_avg
0	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.74	114.74
1	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.74	114.74
2	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.74	114.74
3	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.74	114.74
4	000161a058600d5901f007fab4c27140	2017-07-16 09:40:32	67.41	67.41
...	...	...	...	...
415539	ffffa3172527f765de70084a7e53aae8	2017-09-02 11:53:32	45.50	45.50
415540	ffffe8b65bbe3087b653a978c870db99	2017-09-29 14:07:03	18.37	18.37
415541	ffffe8b65bbe3087b653a978c870db99	2017-09-29 14:07:03	18.37	18.37
415542	ffffe8b65bbe3087b653a978c870db99	2017-09-29 14:07:03	18.37	18.37
415543	ffffe8b65bbe3087b653a978c870db99	2017-09-29 14:07:03	18.37	18.37

415544 rows × 4 columns

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

In [4]:
```python
query = """select *, sum(total_revenue)over(partition by years order by months) as cumulative_revenue from
(select extract(year from cast(o.order_booked AS TIMESTAMP)) AS years,
extract(month from cast(o.order_booked AS TIMESTAMP)) as months, round(cast(sum(p.payment_value) as decimal),2) as Total_revenue
from orders as o
join payments as p
ON o.order_id = p.order_id
group by 1,2)as k"""
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ["Years", "Months","Total_sales","Cumulative_sales"])
df
```

Out[4]:

	Years	Months	Total_sales	Cumulative_sales
0	2016	9	1008.96	1008.96
1	2016	10	236361.92	237370.88
2	2016	12	78.48	237449.36
3	2017	1	553952.16	553952.16
4	2017	2	1167632.04	1721584.20
5	2017	3	1799454.40	3521038.60
6	2017	4	1671152.12	5192190.72
7	2017	5	2371675.28	7563866.00
8	2017	6	2045105.52	9608971.52
9	2017	7	2369531.68	11978503.20
10	2017	8	2697585.28	14676088.48
11	2017	9	2911049.80	17587138.28
12	2017	10	3118711.52	20705849.80
13	2017	11	4779531.20	25485381.00
14	2017	12	3513605.92	28998986.92
15	2018	1	4460016.72	4460016.72
16	2018	2	3969853.36	8429870.08
17	2018	3	4638608.48	13068478.56
18	2018	4	4643141.92	17711620.48
19	2018	5	4615928.60	22327549.08
20	2018	6	4095522.00	26423071.08
21	2018	7	4266163.00	30689234.08
22	2018	8	4089701.28	34778935.36
23	2018	9	17758.16	34796693.52
24	2018	10	2358.68	34799052.20

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

In [53]:
```python
query = """ with one as (select extract(year from cast(o.order_booked AS TIMESTAMP)) AS years, round(cast(sum(p.payment_value)as DECIMAL),2) as Total_revenue from orders as o
join payments as p
ON o.order_id = p.order_id
group by 1)
select years,
total_revenue,
round(((total_revenue - lag(total_revenue,1)over(order by years))/lag(total_revenue,1)over(order by years))*100,2) as YoY_Growth
from one"""
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ["Years", "Total_sales", "YoY Growth"])
df
```

Out[53]:

	Years	Total_sales	YoY Growth
0	2016	237449.36	None
1	2017	28998986.92	12112.70
2	2018	34799052.20	20.00

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

In [54]:
```python
query = """ with a as (select c.customer_id, min(CAST(o.order_booked AS TIMESTAMP)) as first_order from customers c
join orders as o
ON c.customer_id = o.customer_id
group by 1)
select a.customer_id, count(distinct CAST(o.order_booked AS TIMESTAMP)) AS order from a
join orders as o
ON a.customer_id = o.customer_id
and cast(o.order_booked AS TIMESTAMP) > first_order
and cast(o.order_booked AS TIMESTAMP) < first_order + interval '6 months'
group by 1"""
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data)
df
```

Out[54]: —

No customers had purchased the order within 6 month of their 1st order

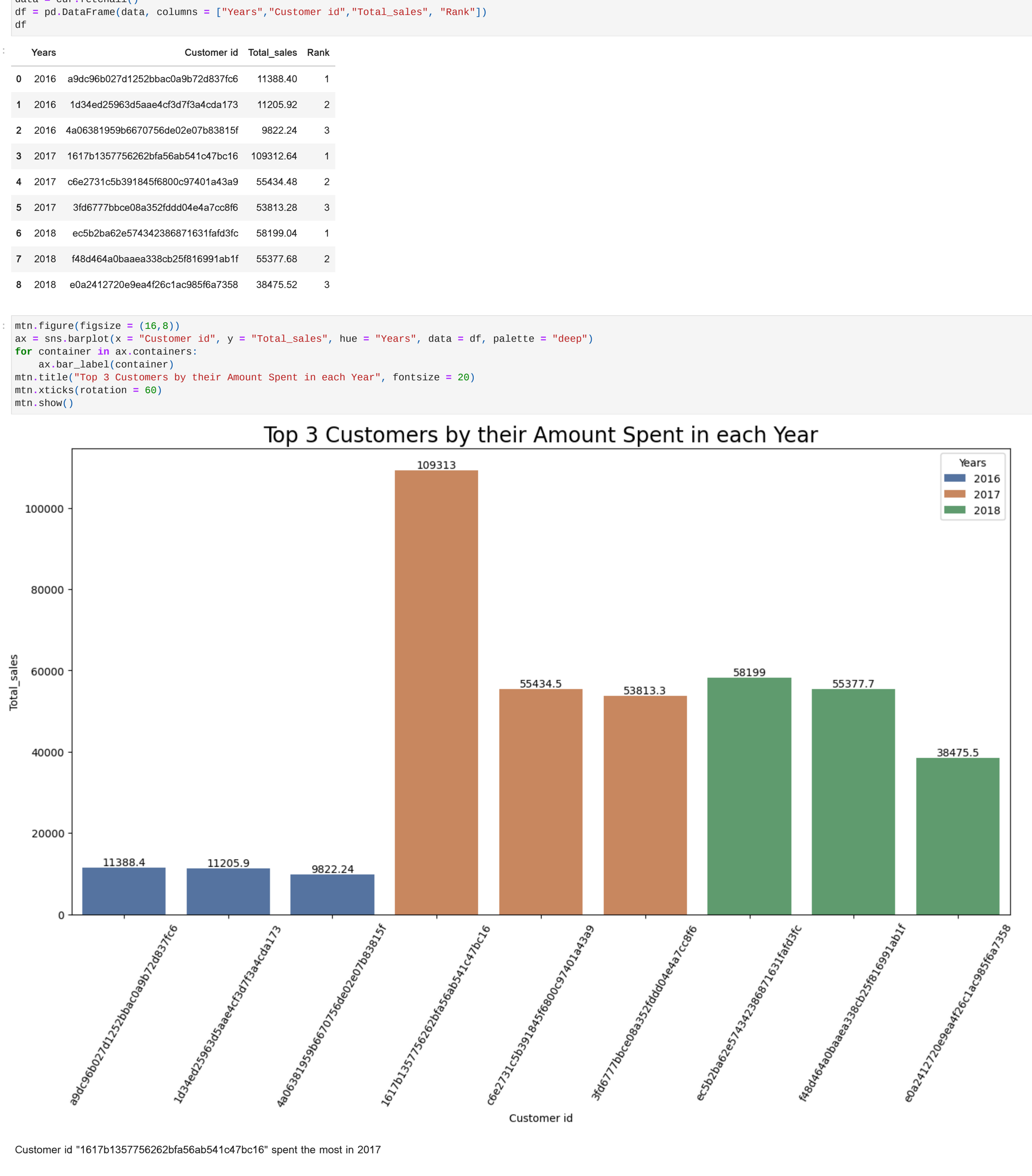
15. Identify the Top 3 customers who spent the most money in each year.

In [32]:
```python
query = """select * from
(select *, rank()over(partition by years order by total_revenue desc) as rank from
(select extract(year from cast(order_booked AS TIMESTAMP)) AS years, c.customer_id, round(cast(sum(p.payment_value) as decimal),2) as TOtal_revenue from customers AS c
join orders as o
ON o.customer_id = c.customer_id
join payments as p
ON o.order_id = p.order_id
group by 1,2)as l)as k
where rank <= 3"""
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ["Years","Customer id","Total_sales", "Rank"])
df
```

Out[32]:

	Years	Customer id	Total_sales	Rank
0	2016	a9dc96b027d1252bbac0a9b72d837fc6	11388.40	1
1	2016	1d34ed25963d5aae4cf3d7f3a4cda173	11205.92	2
2	2016	4a06381959b6670756de02e07b83815f	9822.24	3
3	2017	1617b1357756262bfa56ab541c47bc16	109312.64	1
4	2017	c6e2731c5b391845f6800c97401a43a9	55434.48	2
5	2017	3fd6777bbce08a352fddd04e4a7cc8f6	53813.28	3
6	2018	ec5b2ba62e574342386871631fafd3fc	58199.04	1
7	2018	f48d464a0baaea338cb25f816991ab1f	55377.68	2
8	2018	e0a2412720e9ea4f26c1ac985f6a7358	38475.52	3

In [34]:
```python
mtn.figure(figsize = (16,8))
ax = sns.barplot(x = "Customer id", y = "Total_sales", hue = "Years", data = df, palette = "deep")
for container in ax.containers:
    ax.bar_label(container)
mtn.title("Top 3 Customers by their Amount Spent in each Year", fontsize = 20)
mtn.xticks(rotation = 60)
mtn.show()
```



Customer id "1617b1357756262bfa56ab541c47bc16" spent the most in 2017