E-COMMERCE SALES ANALYSIS

A Data Science Project Using SQL, Python & Jupyter Notebook

Presented by Rakhi Kumari

SQL | Python | Data Analytics | Visualization

1. List all unique cities where customers are located

```
query = '''SELECT distinct(customer_city) FROM e_commerce.customers'''

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



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

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

2. Count the number of order placed in 2017

```
query = '''SELECT
      count(order_id)
      FROM e_commerce.orders
      where year(order_purchase_timestamp) = 2017'''

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



df = pd.DataFrame(data)
"Total orders places in 2017 are", df[0][0]
```

```
('Total orders places in 2017 are', np.int64(45101))
```

3. Calculate the percentage of orders that were paid in installements

```
query = ''' SELECT
    (sum(case
        when payment_installments >=1 then 1
        else 0
        end))/count(*)*100  as total_installment_payment_pct
FROM e_commerce.payments
'''


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


df = pd.DataFrame(data)
"total payment on installment pct" , df[0][0]
```

```
('total payment on installment pct', Decimal('99.9981'))
```

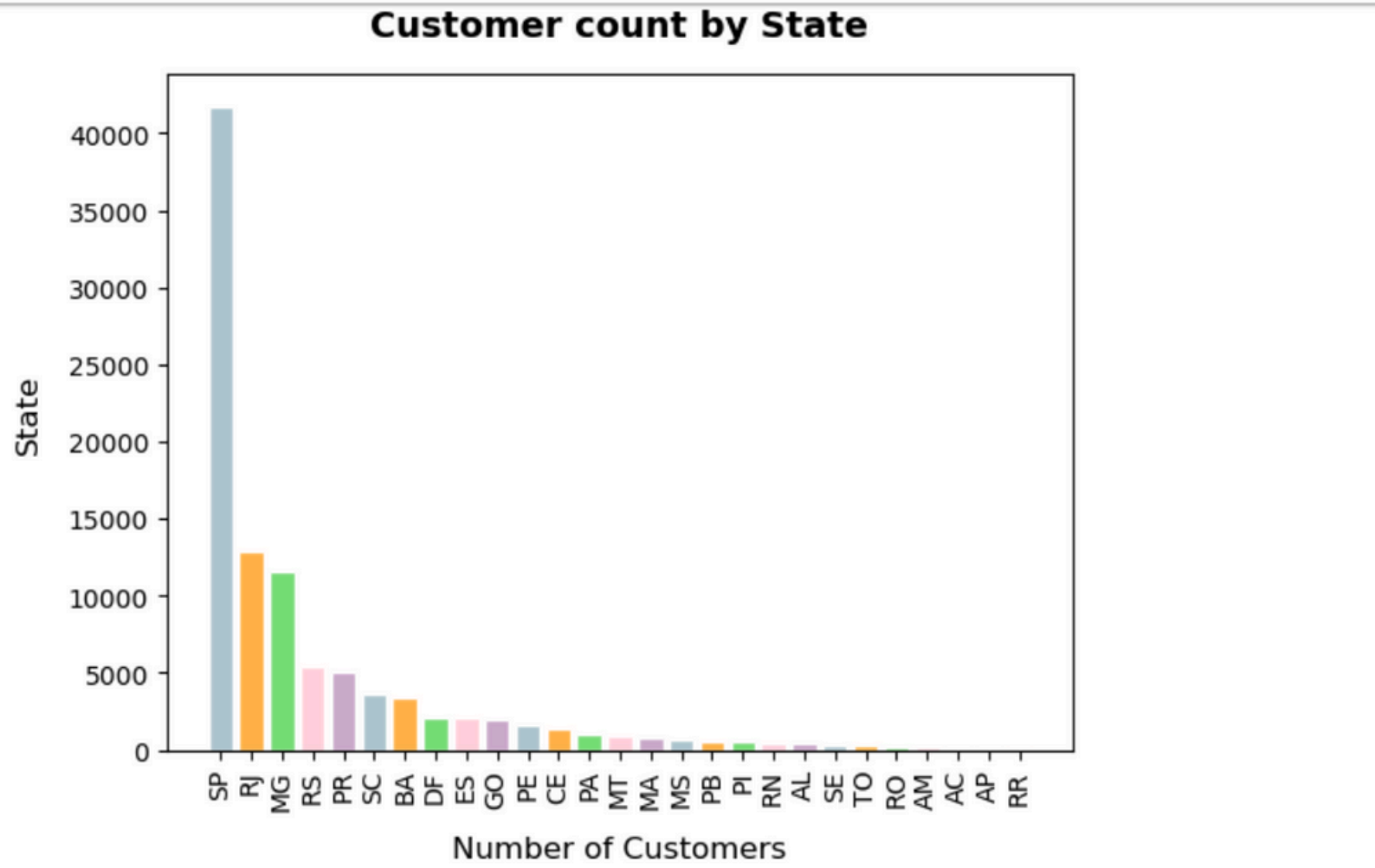
4. Find the total sales per category

```
query = ''' SELECT
       t.product_category, round(sum(p.payment_value),2) as total_value
 FROM order_items o
   join payments p
 on p.order_id = o.order_id
    join products t
 on t.product_id = o.product_id
    group by product_category'''


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


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

	category	sales
0	perfumery	506738.66
1	Furniture Decoration	1430176.39
2	telephony	486882.05
3	bed table bath	1712553.67

5. Count the number of customers in each state

```
query = '''SELECT
          count(customer_id) as customers,
                 customer_state
     FROM e_commerce.customers
     group by customer_state
'''
cur.execute(query)
data = cur.fetchall()

df = pd.DataFrame(data , columns=["customer_count" ,"state"])
df = df.sort_values(by = "customer_count" , ascending = False )

## plot          ]
palette = ['#AEC6CF', '#FFB347', '#77DD77', '#FFD1DC', '#CBAACB']
plt.bar(df["state"], df["customer_count"], color=palette[:len(df)], edgecolor = 'white')
plt.xlabel("Number of Customers", fontsize=12, labelpad=10)
plt.ylabel("State", fontsize=12, labelpad=10)
plt.title("Customer count by State", fontsize=14, weight="bold", pad=15)
plt.xticks(rotation=90)
plt.show()
```



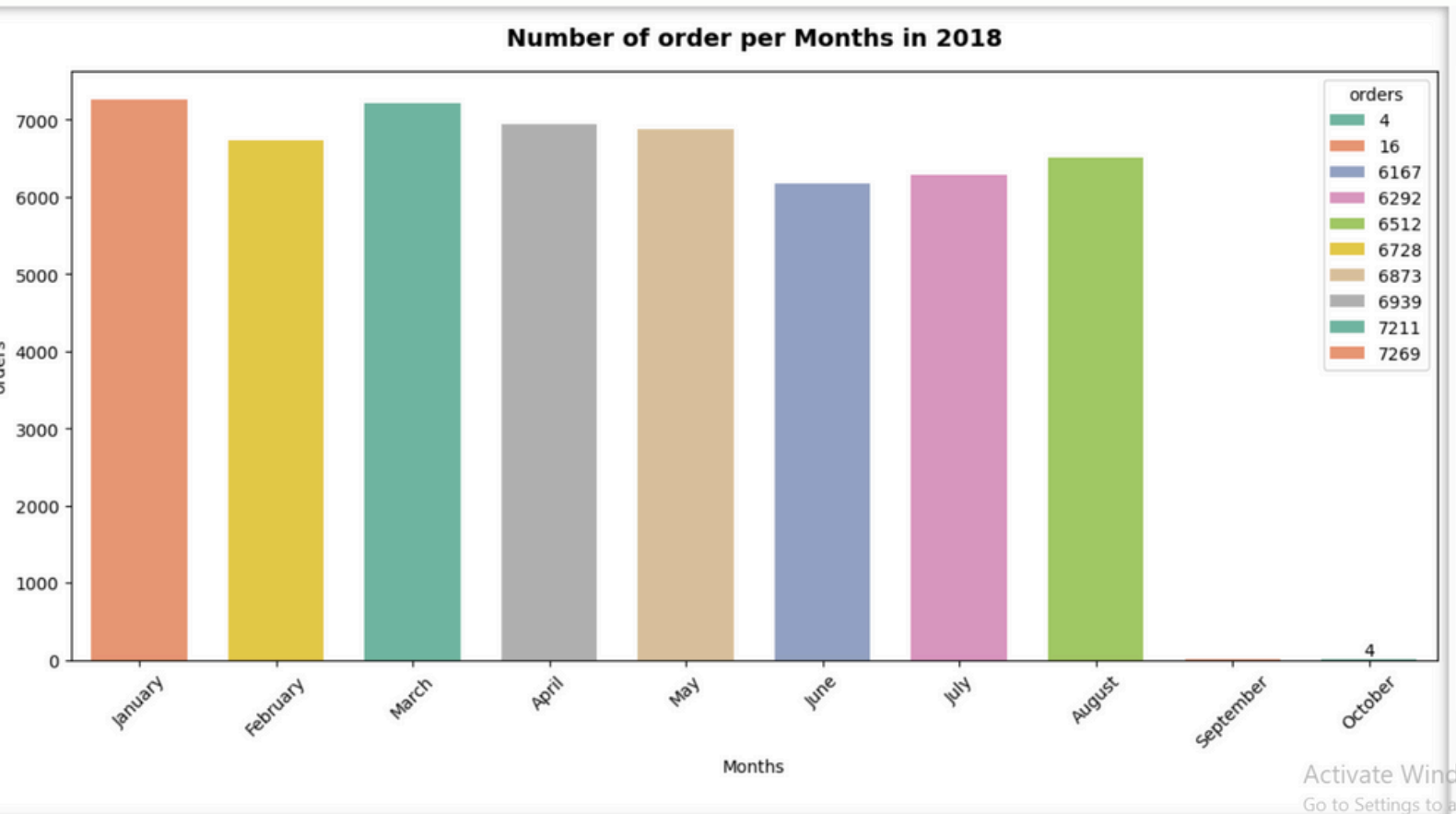
6. Calculate the percentage of total revenue contributed by each product category

```
query = ''' SELECT
      upper(t.product_category) as category,
 round((sum(p.payment_value)/
 (select sum(payment_value) from payments))*100,2)
 as total_revenue_percentage
   FROM order_items o
   JOIN payments p
ON p.order_id = o.order_id
  JOIN products t
ON t.product_id = o.product_id
  GROUP BY product_category
  ORDER BY total_revenue_percentage desc

'''

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


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

	category	percentage
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

7. Calculate the number of orders per month in 2018

```
query = ''' SELECT
           monthname(order_purchase_timestamp) as months ,
           count(order_id) as orders
    FROM orders
where year(order_purchase_timestamp) = 2018
   group by months
   order by months asc
'''
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data , columns=["Months" ,"orders"])
### plot
o = ["January", "February", "March", "April", "May", "June"
     , "July", "August", "September", "October"]
plt.figure(figsize=(12,6))
ax = sns.barplot(x = df["Months"] , y =df["orders"],
        data = df, hue = df["orders"], order = o, palette = "Set2" , width=0.7)
plt.xticks(rotation = 45)
plt.tight_layout()
plt.title("Number of order per Months in 2018", fontsize=14, weight="bold", pad=15)
ax.bar_label(ax.containers[0])
plt.show()
```



8. Find the number of product per orders, group by customer city

```
query = ''' with count_per_order as (
SELECT
      o.order_id,
      o.customer_id,
       count(i.order_item_id) as order_count
  FROM orders o
  join order_items i
 on i.order_id = o.order_id
 group by o.order_id, o.customer_id)
SELECT
     c.customer_city,
     Round(avg(cpo.order_count),2) as average_order
  from customers c
  join count_per_order cpo
 ` on c.customer_id = cpo.customer_id
 group by c.customer_city
 order by average_order desc
'''
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns=["customer_city", "average_orders_per_product"
df.head()
```

	customer_city	average_orders_per_product
0	padre carvalho	7.00
1	celso ramos	6.50
2	datas	6.00
3	candido godoi	6.00
4	matias olimpio	5.00

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

```
query = ''' SELECT
        p.product_category,
        COUNT( ord.product_id),
        round(avg(ord.price),2)
    from products p
    join order_items ord
   on ord.product_id = p.product_id
  group by p.product_category;


'''

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


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

arr1 = df["order_count"]
arr2 = df["price"]
a =np.corrcoef([arr1,arr2])
print("The corelation between product price and product sales is" ,a[0][-1])
```

```
The corelation between product price and product sales is -0.10631514167157562
```

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
    od.seller_id,
    sum(pt.payment_value) revenue
 FROM order_items od
 join payments pt
 on pt.order_id = od.order_id
 group by od.seller_id
 order by revenue desc) as a;
'''

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


df = pd.DataFrame(data, columns= ["seller_id", "revenue", "rank"])
df.head()
```

	seller_id	revenue	rank
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

11. Calculate the moving average of order values

for each customer over their order history.

```
query = ''' SELECT
           *,
 avg(payment)
 over(partition by customer_id order by order_purchase_timestamp
 rows between 2 preceding and current row) AS mov_avg
    FROM
 (SELECT
    ord.customer_id, ord.order_purchase_timestamp,
    pt.payment_value AS payment
 FROM orders ord
 JOIN payments pt
 ON pt.order_id = ord.order_id) as a
'''

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


df = pd.DataFrame(data, columns= ["customer_id", "order_purchase_timestamp", "payment", "moving_avg"])
df
```

	customer_id	order_purchase_timestamp	payment	moving_avg
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
...	...	...	...	...
103881	fffecc9f79fd8c764f843e9951b11341	2018-03-29 16:59:26	71.23	27.120001
103882	fffeda5b6d849fbd39689bb92087f431	2018-05-22 13:36:02	63.13	63.130001
103883	ffff42319e9b2d713724ae527742af25	2018-06-13 16:57:05	214.13	214.130005

12. Calculate the cumulative sales per month for each yea

```
query = ''' select
     year, month, payments,
    sum(payments) over(order by year, month) as cumulative_sales
 FROM
 (SELECT
  YEAR(ord.order_purchase_timestamp) as year,
   month(ord.order_purchase_timestamp) as month,
round(sum(pt.payment_value),2) as payments
   FROM orders ord
   JOIN payments pt
   on pt.order_id = ord.order_id
   group by year , month
   order by year, month) AS a
'''

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


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

	Years	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

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

```
query = ''' with a as (SELECT
  YEAR(ord.order_purchase_timestamp) as year,
round(sum(pt.payment_value),2) as payments
   FROM orders ord
   JOIN payments pt
   on pt.order_id = ord.order_id
   group by year
   order by year)

SELECT year, round((payments - lag(payments,1) over(order by year))/
lag(payments,1) over(order by year),2) * 100
AS previous_year from a
'''

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


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

	years	yoy growth_rate
0	2016	NaN
1	2017	12113.0
2	2018	20.0

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

```
query =''' with a as
     (SELECT
     ct.customer_id,
     min(ord.order_purchase_timestamp) as first_order
 FROM customers ct
 join orders ord
 on ord.customer_id = ct.customer_id
 group by ct.customer_id),

b as
     (SELECT
        a.customer_id , count(distinct ord.order_purchase_timestamp) as next_order
   FROM a
 join orders ord
 on ord.customer_id = a.customer_id and
 ord.order_purchase_timestamp > first_order and
 ord.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
 '''

cur.execute(query)
data = cur.fetchall()
print ("since none of our customer repeated , thats why value is", data)
```

```
since none of our customer repeated , thats why value is [(None,)]
```

15. Calculate the top 3 customers

who spent the most money in each year

```
query = '''SELECT *
 FROM
 (SELECT
     year(ord.order_purchase_timestamp) as year,
     ord.customer_id,
     sum(pt.payment_value) as payment,
    dense_rank() over(partition by year(ord.order_purchase_timestamp)
    order by  sum(pt.payment_value)desc) as d_rank
     from orders ord
     join payments pt
     on pt.order_id = ord.order_id
     group by year(order_purchase_timestamp), ord.customer_id) as a
WHERE d_rank <= 3
'''

cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data , columns = ["years" , "customer_id", "payment", "rank"])

# plot
palette = ["#000000", "#1C1C1C","#2F2F2F"
          ]
sns.barplot(x= "customer_id", y= "payment" , data= df , hue= "years", palette = "Set2")
plt.title("Top 3 customers by anual spend", fontsize=14, weight="bold", pad=15 )
plt.xticks(rotation= 45)
plt.show()
```