**Business Case: Target SQL**

**Context:** Target is a globally renowned brand and a prominent retailer in the United States. Target makes itself a preferred shopping destination by offering outstanding value, inspiration, innovation and an exceptional guest experience that no other retailer can deliver.

**Problem statement:** To analyze data collected between 2016 and 2018 for the Brazil region, extract meaningful insights, and provide actionable recommendations to support data-driven decision-making and strategic business growth.

**Analysis:**

**Q1:** **Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:**

1. **Data type of all columns in the "customers" table.**

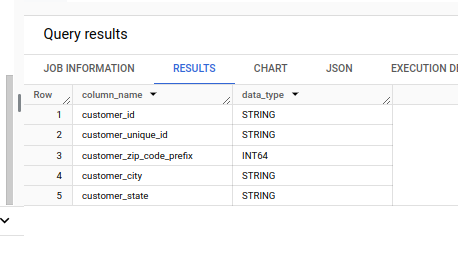
**Query:**

select column\_name,data\_type

from sunm-442402.target.INFORMATION\_SCHEMA.COLUMNS

where table\_name = 'customers';

**Query result screenshot:**



**Insights:**

From the above query result, we can see that all the columns in customers table are of type string. Even though zip-code is stored as an integer, it is more suited for a categorical data type and not a numerical data type as numerical operations on zip code does not make sense

1. **Get the time range between which the orders were placed.**

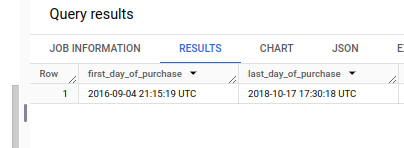
**Query:**

select min(order\_purchase\_timestamp) as first\_day\_of\_purchase,

max(order\_purchase\_timestamp) as last\_day\_of\_purchase

from `target.orders`;

**Query result screenshot:**



**Insights:**

The data given captures order transactions from the first recorded purchase on 2016-09-04 21:15:19 UTC to the most recent on 2018-10-17 17:30:18 UTC, providing a comprehensive timeline for analyzing customer behavior and sales trends during this period.

1. **Count the Cities & States of customers who ordered during the given period.**

**Query:**

select count(distinct c.customer\_city) as city\_count,

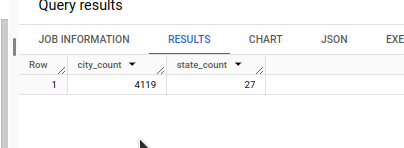
count(distinct c.customer\_state) as state\_count

from `target.orders` o

inner join `target.customers` c

using (customer\_id);

**Query result screenshot:**



**Insights:**

From the above we can see that there are **27** **distinct states** and **4119 distinct cities** where customers have placed orders, highlighting the geographical diversity of our customer base which can be used to identify regions with the highest engagement or market coverage.

**Q2: In-depth Exploration:**

1. **Is there a growing trend in the no. of orders placed over the past years?**

**Query:**

select extract(year from o.order\_purchase\_timestamp) as order\_year,

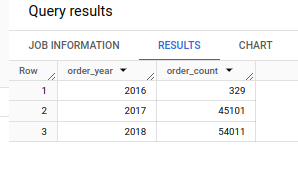
count(o.order\_id) as order\_count

from `target.orders` o

group by order\_year

order by order\_year;

**Query result screenshot:**



**Insights:**

From the above we can see that initially the orders were very less in the first year (order count 329 in 2016), but this increased to a significantly higher level (45101 in 2017 and 54011 in 2018). Even by considering that we have the data for only 3 months in 2016 we can see that there is an upward trend in the orders received by Target.

1. **Can we see some kind of monthly seasonality in terms of the no. of orders being placed?**

**Query:**

select

extract(year from o.order\_purchase\_timestamp) as order\_year,

extract(month from o.order\_purchase\_timestamp) as order\_month,

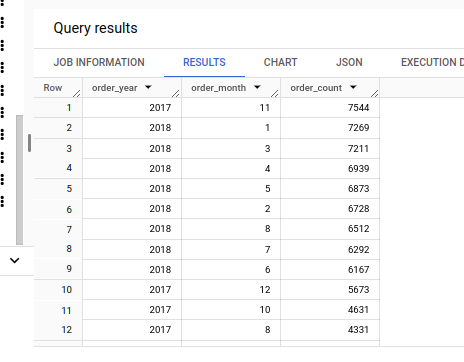
count(o.order\_id) as order\_count

from `target.orders` o

group by order\_year, order\_month

order by order\_count desc;

**Query result screenshot:**



**Insights:**

From the above results we can see that the order follows a certain pattern. The number of orders rises up just before December and dips down during December and again rises in the beginning of the new year till almost the mid of the year post which it starts to dip down again.

1. **During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)**
   1. **0-6 hrs : Dawn**
   2. **7-12 hrs : Mornings**
   3. **13-18 hrs : Afternoon**
   4. **19-23 hrs : Night**

**Query:**

select

case

when extract(hour from o.order\_purchase\_timestamp) between 0 and 6 then 'Dawn'

when extract(hour from o.order\_purchase\_timestamp) between 7 and 12 then 'Morning'

when extract(hour from o.order\_purchase\_timestamp) between 13 and 18 then 'Afternoon'

when extract(hour from o.order\_purchase\_timestamp) between 19 and 23 then 'Night'

end as time\_of\_day,

count(o.order\_id) as order\_count

from `target.orders` o

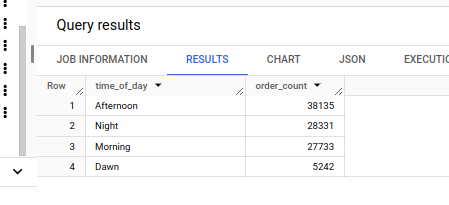
inner join `target.customers` c

using (customer\_id)

group by time\_of\_day

order by order\_count desc;

**Query result screenshot:**



**Insights:**

From the above query results we can see that the Brazilians place their orders mostly during the Afternoon.

**Q3: Evolution of E-commerce orders in the Brazil region:**

1. **Get the month on month no. of orders placed in each state.**

**Query:**

select

c.customer\_state,

extract(year from o.order\_purchase\_timestamp) as order\_year,

extract(month from o.order\_purchase\_timestamp) as order\_month,

count(o.order\_id) as order\_count

from `target.orders` o

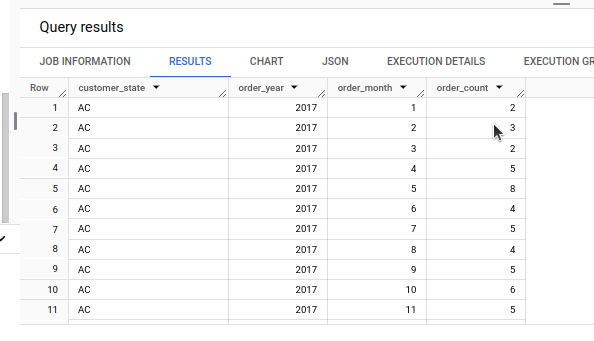
inner join `target.customers` c

using (customer\_id)

group by c.customer\_state, order\_year, order\_month

order by c.customer\_state, order\_year, order\_month;

**Query result screenshot:**



**Insights:**

We can derive multiple insights from the above results and below are a few of them.

* State SP is the state with the highest number of orders and most of the orders were in the year 2018
* Orders in state AC is more in the beginning of 2018 and after it has a few downs and ups.

1. **How are the customers distributed across all the states?**

**Query:**

select

c.customer\_state,

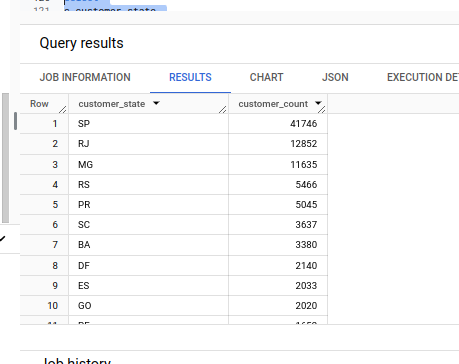
count(distinct c.customer\_id) as customer\_count

from `target.customers` c

group by c.customer\_state

order by customer\_count desc;

**Query result screenshot:**



**Insights:**

From the results we can see that the majority of the customers are located in SP, i.e. more than thrice the customers present in the state RJ, which has the second highest customer count.

**Q4: Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.**

1. **Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).**

**Query:**

with yearly\_payment as (

select

extract(year from o.order\_purchase\_timestamp) as order\_year,

sum(p.payment\_value) as total\_payment

from `target.orders` o

inner join `target.payments` p

using (order\_id)

where extract(month from o.order\_purchase\_timestamp) between 1 and 8

group by order\_year

having order\_year in (2017, 2018)

)

select y2017,y2018,

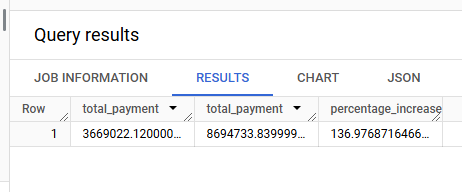
(y2018.total\_payment - y2017.total\_payment) / y2017.total\_payment \* 100 as percentage\_increase

from

(select total\_payment from yearly\_payment where order\_year = 2017) y2017,

(select total\_payment from yearly\_payment where order\_year = 2018) y2018;

**Query result screenshot:**



**Insights:**

The results show that the payment value has increased by almost 137% from the year 2017 to 2018 for the months between January to August.

1. **Calculate the Total & Average value of the order price for each state.**

**Query:**

select

c.customer\_state,

round(sum(p.payment\_value),2) as total\_order\_value,

round(avg(p.payment\_value),2) as average\_order\_value

from `target.orders` o

inner join `target.payments` p

using (order\_id)

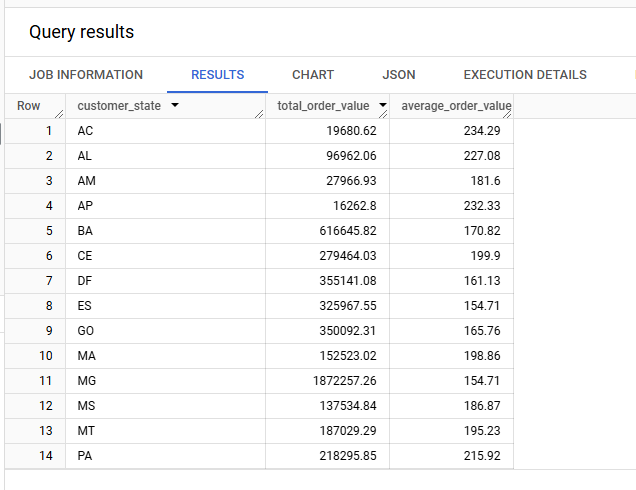
inner join `target.customers` c

using (customer\_id)

group by c.customer\_state

order by c.customer\_state;

**Query result screenshot:**

****

**Insights:**

Multiple insights can be derived from the above results. Below are a few of them

* State SP shows a significant contribution to the total order value, but its average order value is comparatively lower, suggesting that a high volume of smaller orders drives the revenue here. This state can be targeted with campaigns promoting bundled deals or bulk discounts to increase the average order size.
* State PB has a moderate total order value but a relatively high average order value, indicating a preference for premium or higher-priced products. This state can be targeted with exclusive product launches or premium membership programs to further capitalize on customer spending habits.
* States with lower total order values, such as RR, AP, and AC may represent untapped potential. Focused marketing campaigns or region-specific discounts could help boost sales in these regions.

1. **Calculate the Total & Average value of order freight for each state.**

**Query:**

select

c.customer\_state,

round(sum(oi.freight\_value), 2) as total\_freight\_value,

round(avg(oi.freight\_value), 2) as average\_freight\_value

from `target.orders` o

inner join `target.order\_items` oi

using (order\_id)

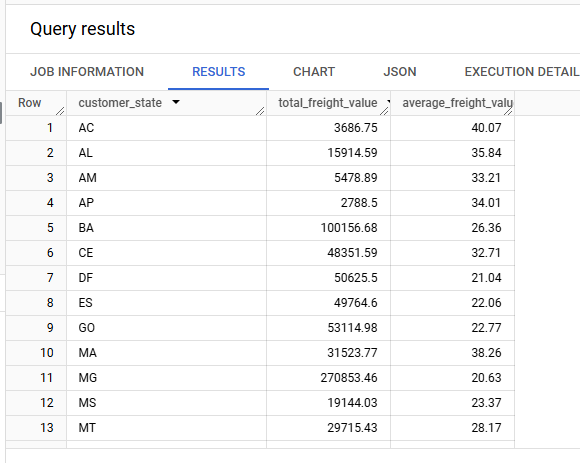
inner join `target.customers` c

using (customer\_id)

group by c.customer\_state

order by c.customer\_state;

**Query result screenshot:**

****

**Insights:**

Below are few insights based on the above results

* State SP has a high total transport cost and a low average transport cost, indicating that this state has a huge order count with a low order value for each order.
* State RR has a high average transport cost and a comparatively low total transport cost, indicating that this has fewer orders, but each order is of high value.

**Q5: Analysis based on sales, freight, and delivery time**

1. **Find the no. of days taken to deliver each order from the order’s purchase date as delivery time.  
   Also, calculate the difference (in days) between the estimated & actual delivery date of an order.  
   Do this in a single query.**

**Query:**

select

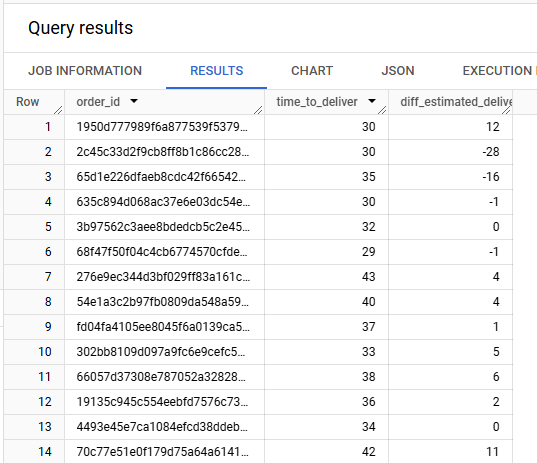
o.order\_id,

date\_diff(o.order\_delivered\_customer\_date, o.order\_purchase\_timestamp, day) as time\_to\_deliver,

date\_diff(o.order\_delivered\_customer\_date, o.order\_estimated\_delivery\_date, day) as diff\_estimated\_delivery

from `target.orders` o;

**Query result screenshot:**

****

**Insights:**

The time to deliver column represents the time taken for the order to reach the customer from the date of ordering. This can be improved wherever the time is too high.

The difference in estimated delivery is the difference between the delivered date and the estimated delivery date. If this is low or negative, it means the order was delivered very fast and in case of negative value it means the order got delivered faster than expected. Such deliveries can be analyzed to find out what helped with reducing the time and the same can be implemented to other orders wherever possible.

1. **Find out the top 5 states with the highest & lowest average freight value.**

**Query:**

with highest\_avg\_freight as (

select

c.customer\_state,

round(avg(oi.freight\_value), 2) as average\_freight\_value,

row\_number() over (order by avg(oi.freight\_value) desc) as row\_num

from `target.order\_items` oi

inner join `target.orders` o

using (order\_id)

inner join `target.customers` c

using (customer\_id)

group by c.customer\_state

),

lowest\_avg\_freight as (

select

c.customer\_state,

round(avg(oi.freight\_value), 2) as average\_freight\_value,

row\_number() over (order by avg(oi.freight\_value) asc) as row\_num

from `target.order\_items` oi

inner join `target.orders` o

using (order\_id)

inner join `target.customers` c

using (customer\_id)

group by c.customer\_state

)

select

hf.customer\_state as high\_freight\_state,

hf.average\_freight\_value as high\_freight\_value,

lf.customer\_state as low\_freight\_state,

lf.average\_freight\_value as low\_freight\_value

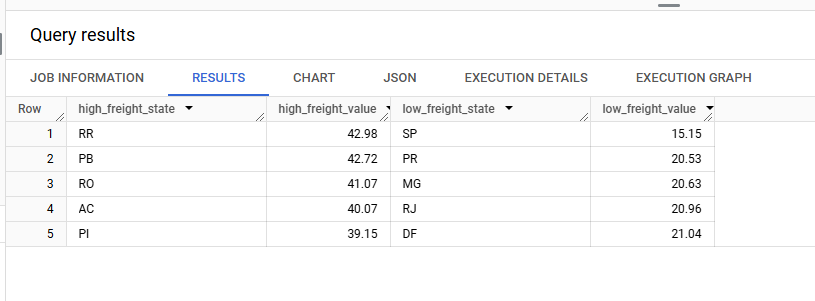
from highest\_avg\_freight hf

join lowest\_avg\_freight lf

on hf.row\_num = lf.row\_num

where hf.row\_num <= 5 and lf.row\_num <= 5;

**Query result screenshot:**

****

**Insights:**

The first two columns display the top 5 states with the highest average freight values, while the next two columns show the top 5 states with the lowest average freight values. These averages can help estimate the likely transport costs for each order. Additionally, this data suggests that states with lower freight values may be ordering goods that are easier to transport, while states with higher freight values are likely to order goods that are more challenging to transport.

1. **Find out the top 5 states with the highest & lowest average delivery time.**

**Query:**

with delivery\_times as (

select

c.customer\_state,

date\_diff(o.order\_delivered\_customer\_date, o.order\_purchase\_timestamp, day) as delivery\_time

from `target.orders` o

inner join `target.customers` c

on o.customer\_id = c.customer\_id

where o.order\_status = 'delivered'

),

ranked\_delivery\_times as (

select

customer\_state as state,

avg(delivery\_time) as avg\_delivery\_time,

rank() over (order by avg(delivery\_time) desc) as high\_rank,

rank() over (order by avg(delivery\_time) asc) as low\_rank

from delivery\_times

group by customer\_state

)

select

high\_states.state as high\_state,

round(high\_states.avg\_delivery\_time,2) as high\_avg\_delivery\_time,

low\_states.state as low\_state,

round(low\_states.avg\_delivery\_time,2) as low\_avg\_delivery\_time

from ranked\_delivery\_times high\_states

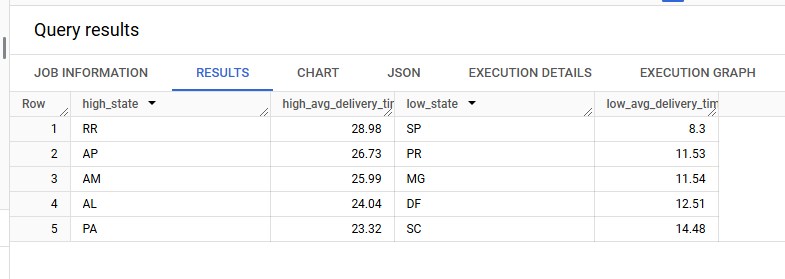
join ranked\_delivery\_times low\_states

on high\_states.high\_rank = low\_states.low\_rank

where high\_states.high\_rank <= 5 and low\_states.low\_rank <= 5

order by high\_states.high\_rank;

**Query result screenshot:**

****

**Insights:**

The results show the top 5 states with the highest average delivery times and the top 5 states with the lowest average delivery times. Analyzing the fast delivery states can provide valuable insights into the factors contributing to quicker deliveries. These factors can potentially be adopted or replicated by the states with slower deliveries to improve their performance.

1. **Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.**

**Query:**

select customer\_state as state,

round(avg(date\_diff(o.order\_delivered\_customer\_date, o.order\_estimated\_delivery\_date, day)),2) as avg\_speed\_delivery

from `target.customers` as c

join `target.orders` as o on c.customer\_id = o.customer\_id

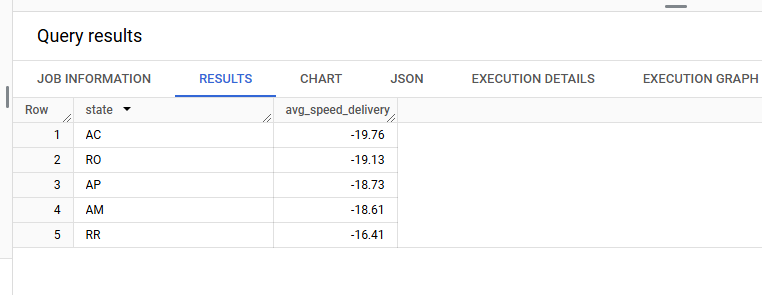
where o.order\_status = 'delivered'

group by state

order by avg\_speed\_delivery

limit 5;

**Query result screenshot:**

****

**Insights:**

The results highlight the top 5 states with the fastest deliveries, where the negative values in the average indicate that, on average, orders were delivered well ahead of the estimated delivery date. By analyzing the factors contributing to this efficiency in these top states, we can explore opportunities to replicate these practices in other states to enhance overall delivery performance.

**Q6: Analysis based on the payments**

1. **Find the month on month no. of orders placed using different payment types.**

**Query:**

select

extract(year from o.order\_purchase\_timestamp) as year,

extract(month from o.order\_purchase\_timestamp) as month,

p.payment\_type,

count(o.order\_id) as number\_of\_orders

from `target.orders` o

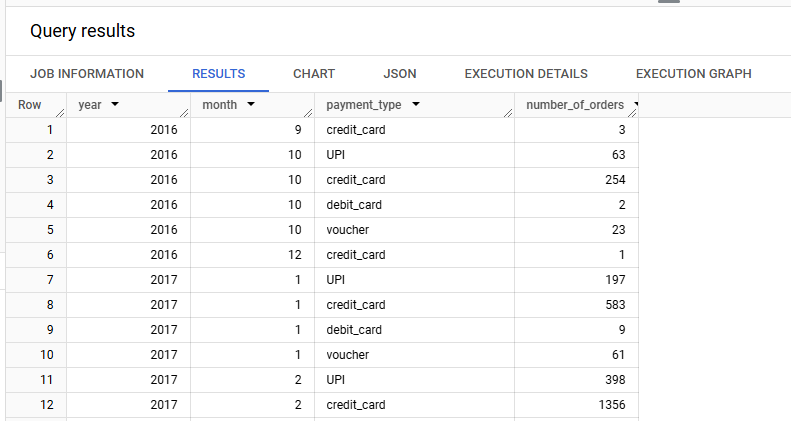
join `target.payments` p

on o.order\_id = p.order\_id

group by year, month, p.payment\_type

order by year, month, p.payment\_type;

**Query result screenshot:**

****

**Insights:**

The number\_of\_orders column shows the count of orders placed for each payment method. This data allows us to identify which payment type is most preferred by customers, providing insights into customer behavior and helping tailor payment options to improve user experience.

1. **Find the no. of orders placed on the basis of the payment installments that have been paid.**

**Query:**

select

payment\_installments,

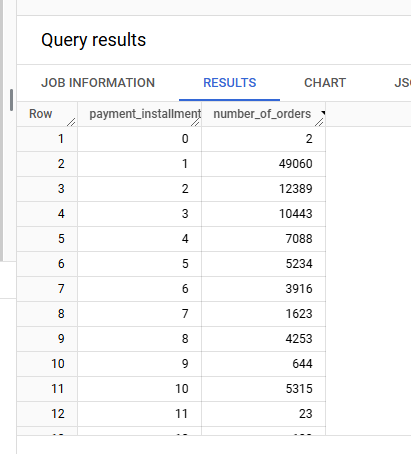
count(distinct p.order\_id) as number\_of\_orders

from `target.payments` p

group by payment\_installments

order by payment\_installments;

**Query result screenshot:**

****

**Insights:**

The results provide the number of orders for each installment option. This data helps us analyze customer preferences, such as how many customers prefer to pay in full (no installments) versus opting for payment plans like 3-month, 6-month, or longer installment options. By understanding these preferences, we can gain valuable insights into how customers value payment flexibility.

For example, if a significant number of customers choose installment plans, it might indicate a demand for more flexible payment options. On the other hand, if most customers prefer paying in full, it could suggest they prioritize simplicity or may be influenced by discounts for upfront payments.

These insights can be used to tailor offers and promotions. For instance, you could incentivize installment plans by offering low or zero interest rates, or encourage full payments by providing discounts for upfront transactions. Ultimately, this analysis helps businesses align their payment strategies with customer preferences, improving satisfaction and driving sales.

**Segmenting customers using RFM**

First I am extracting the columns that are needed for this.

select c.customer\_id,

o.order\_purchase\_timestamp,

p.payment\_value

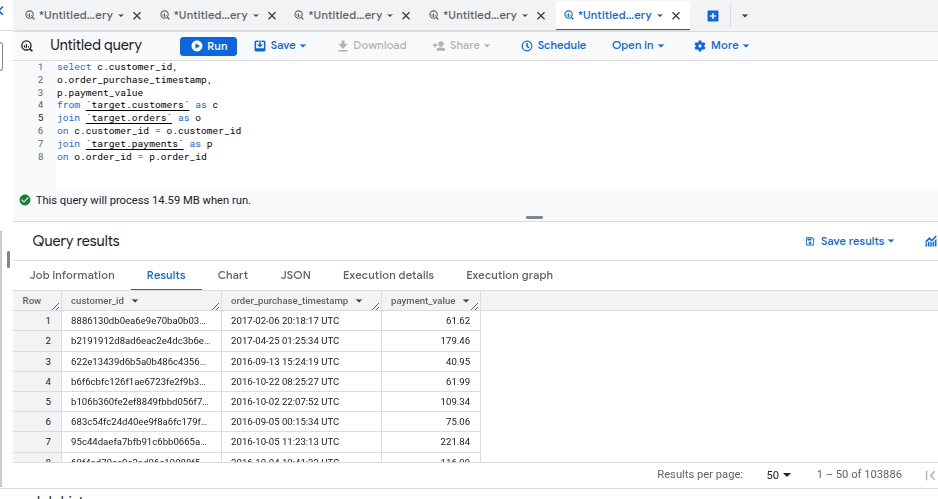
from `target.customers` as c

join `target.orders` as o

on c.customer\_id = o.customer\_id

join `target.payments` as p

on o.order\_id = p.order\_id



I am also exporting this data as a csv file so that I can use that later for comparison with other clustering algos.

Now, below I have the draft columns with RFM values:

with cust\_cte as (

select c.customer\_id,

o.order\_purchase\_timestamp,

p.payment\_value,

max(o.order\_purchase\_timestamp) over() as reference\_date,

from `target.customers` as c

join `target.orders` as o

on c.customer\_id = o.customer\_id

join `target.payments` as p

on o.order\_id = p.order\_id

)

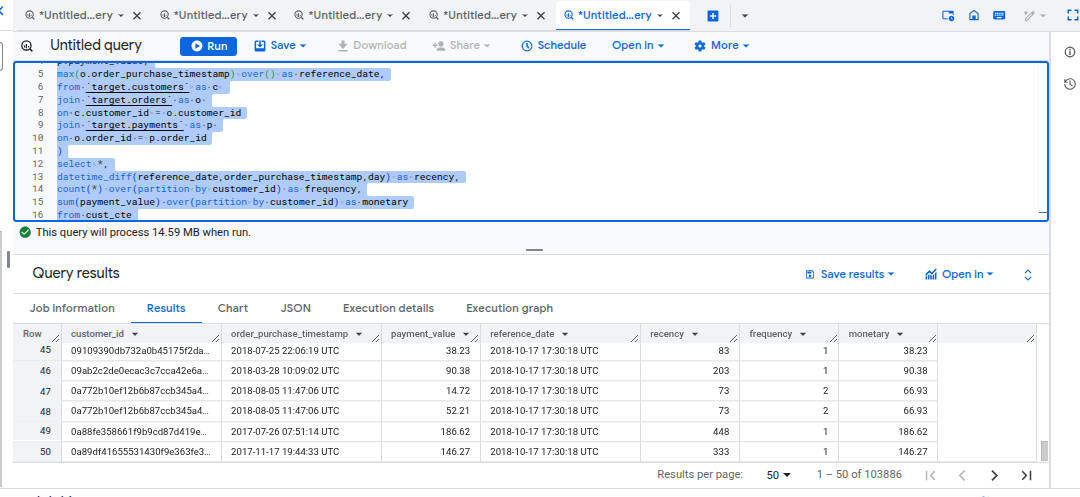
select \*,

datetime\_diff(reference\_date,order\_purchase\_timestamp,day) as recency,

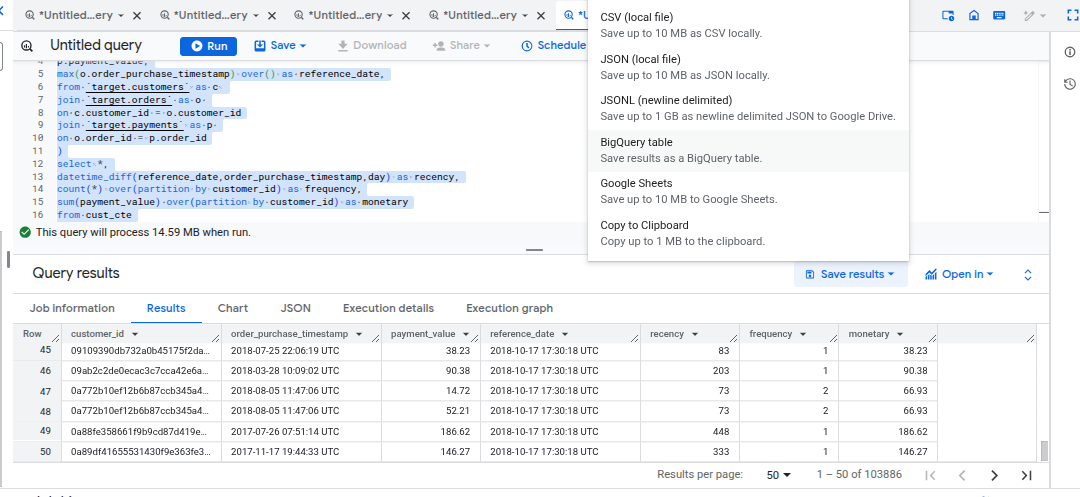
count(\*) over(partition by customer\_id) as frequency,

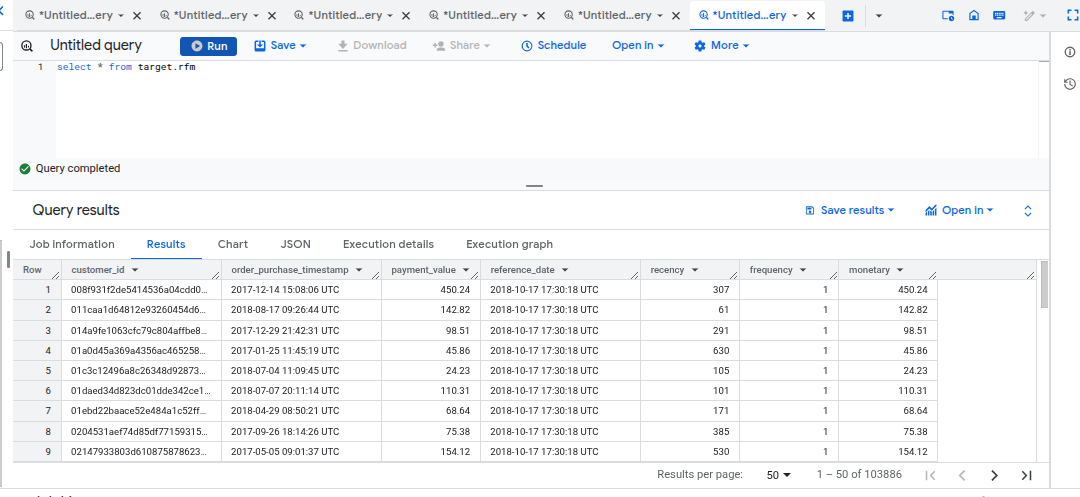
sum(payment\_value) over(partition by customer\_id) as monetary

from cust\_cte



I am saving this table so that I can query using a simple query instead of this complex one.



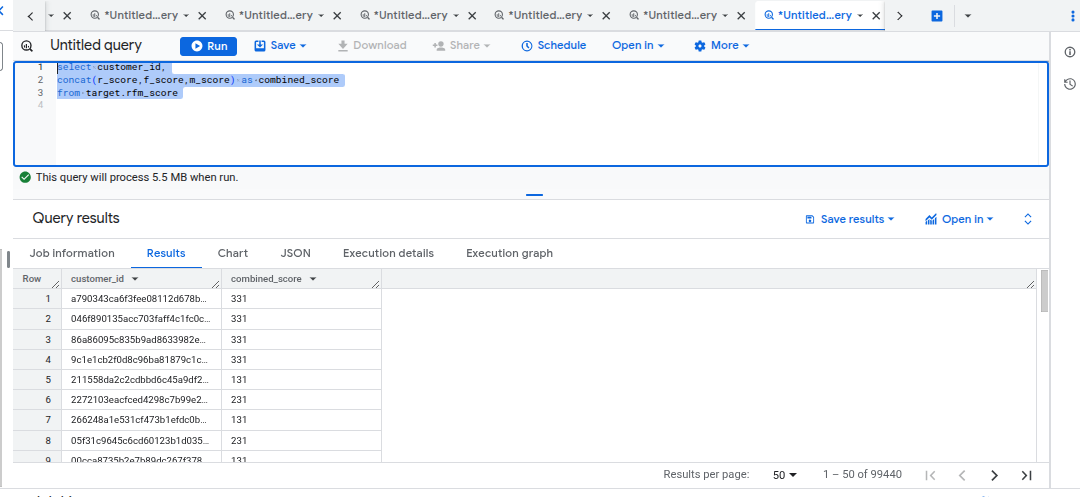


Final segregation of customers based on rfm

select customer\_id,

concat(r\_score,f\_score,m\_score) as combined\_score

from target.rfm\_score



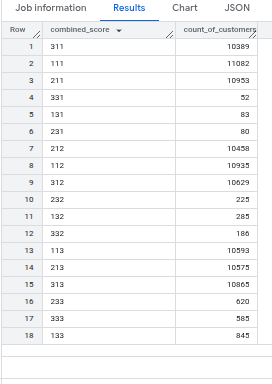
select

concat(r\_score,f\_score,m\_score) as combined\_score,

count(customer\_id) as count\_of\_customers

from target.rfm\_score

group by 1



With the above we are able to see the number of customers segregated based on their RFM scores.

Now for curiosity sake I am checking the clustering using k-means clustering.

Just to see how that clusters the customers based on the same data.

import pandas as pd

from datetime import datetime

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

df = pd.read\_csv('cust.csv')

df['order\_purchase\_timestamp'] = pd.to\_datetime(df['order\_purchase\_timestamp'])

max\_date = df['order\_purchase\_timestamp'].max()

df['recency\_days'] = (max\_date - df['order\_purchase\_timestamp']).dt.days

df = df[['customer\_id', 'recency\_days', 'payment\_value']]

df = df.groupby('customer\_id').agg({'recency\_days': 'mean', 'payment\_value': 'mean'}).reset\_index()

scaler = StandardScaler()

scaled = scaler.fit\_transform(df[['recency\_days', 'payment\_value']])

kmeans = KMeans(n\_clusters=4)

df['cluster'] = kmeans.fit\_predict(scaled)

print(df)

import matplotlib.pyplot as plt

import seaborn as sns

sns.set(style="whitegrid")

plt.figure(figsize=(10, 6))

sns.scatterplot(

x='recency\_days',

y='payment\_value',

hue='cluster',

palette='Set1',

data=df,

s=100,

edgecolor='black'

)

plt.title('K-Means Clusters of Customers')

plt.xlabel('Recency (days)')

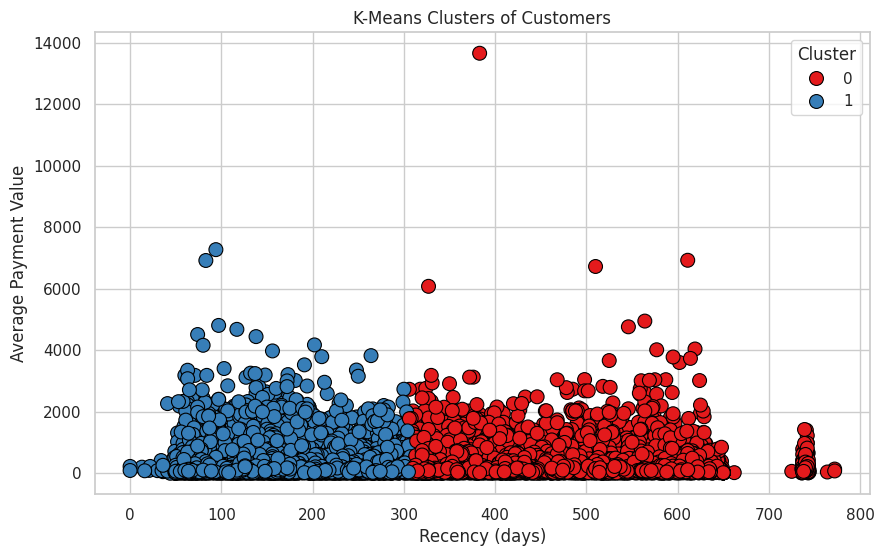
plt.ylabel('Average Payment Value')

plt.legend(title='Cluster')

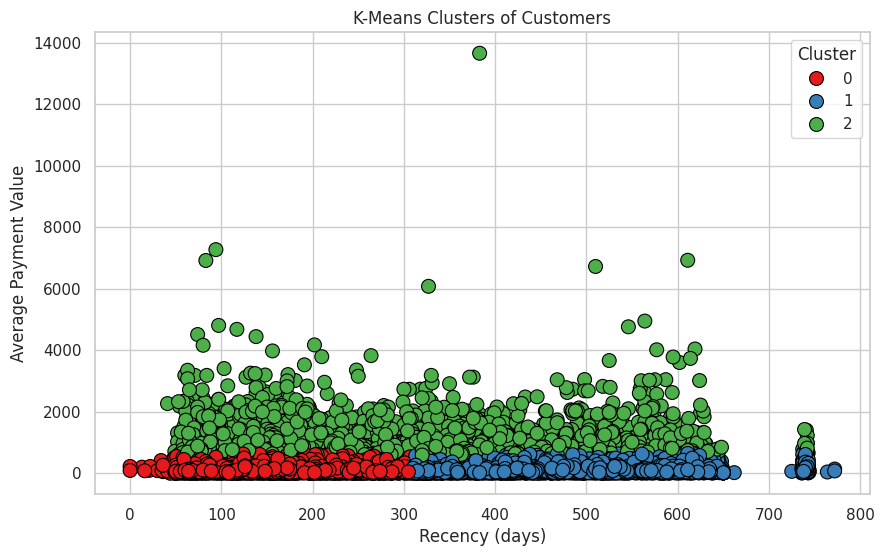
plt.show()

With the above code, and by tweaking the n\_clusters, we get the below.

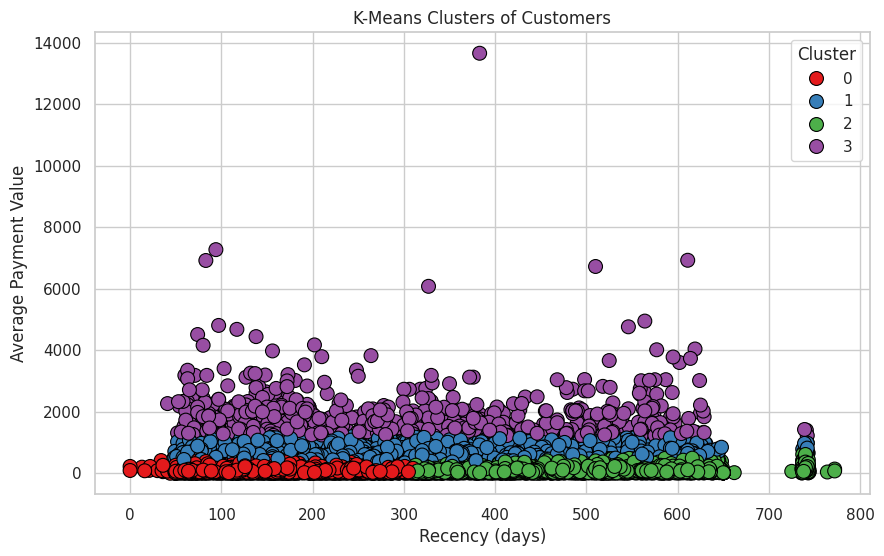
n\_clusters=2



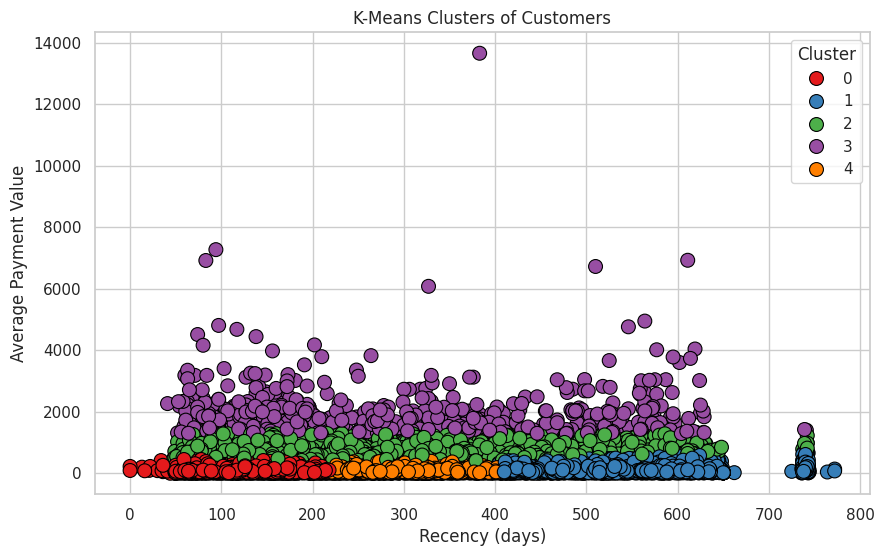
n\_clusters=3

****

n\_clusters=4

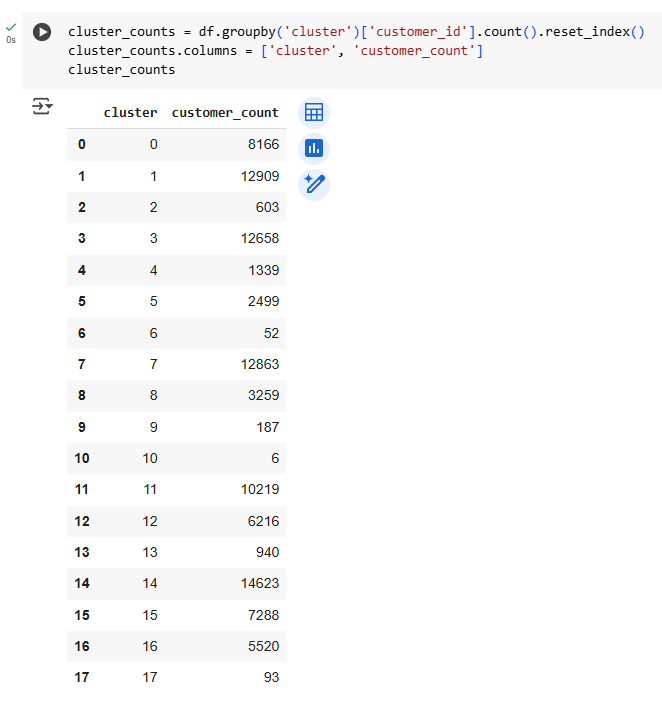


n\_clusters=5



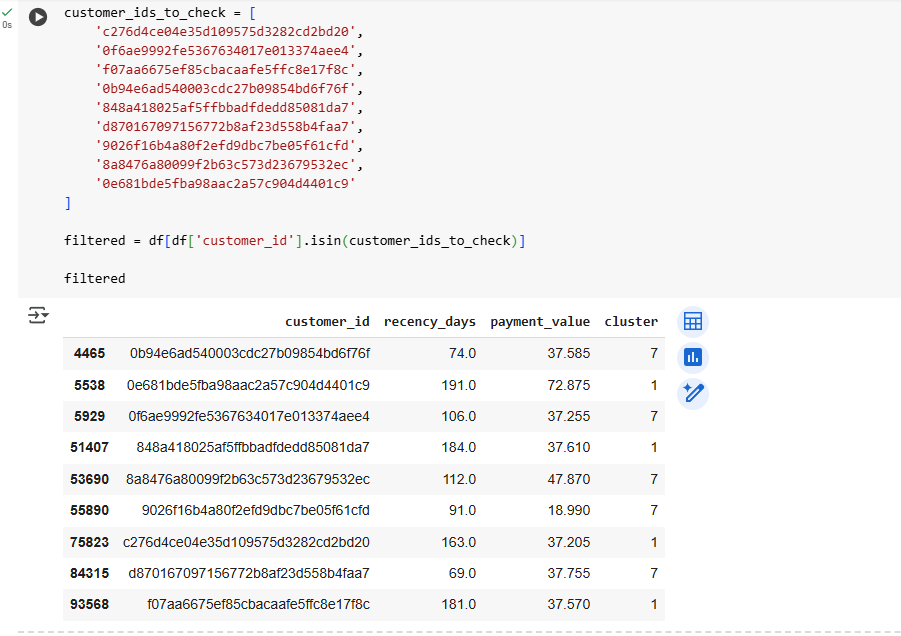
We can see here that with increase in clusters we see splits in the closely packed data. But we can see the main split between high spenders and low spending customers even across recency.

Here we see that the clusters have a different count of customers compared to RFM.



However, when I am checking for the customer\_ids that belong to the same cluster in RFM, I see that the IDs are grouped within 2 or 3 clusters.

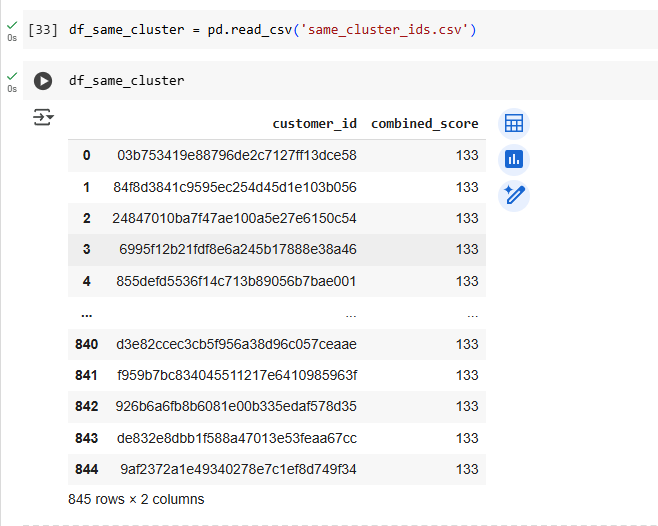
Below all these IDs belong to the same cluster in RFM, and K-means has them in either cluster 7 or 1.



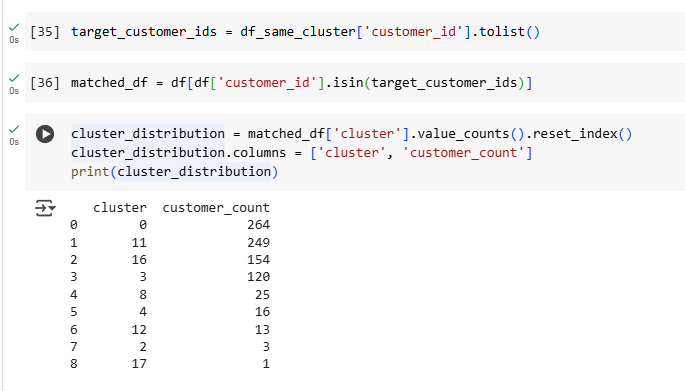
customer\_ids below are in the same group in RFM, and k-means has divided them into 3 clusters.



Now I am checking for all the ids that belong to a single RFM cluster by exporting the data into a csv file and importing as a pandas dataFrame.

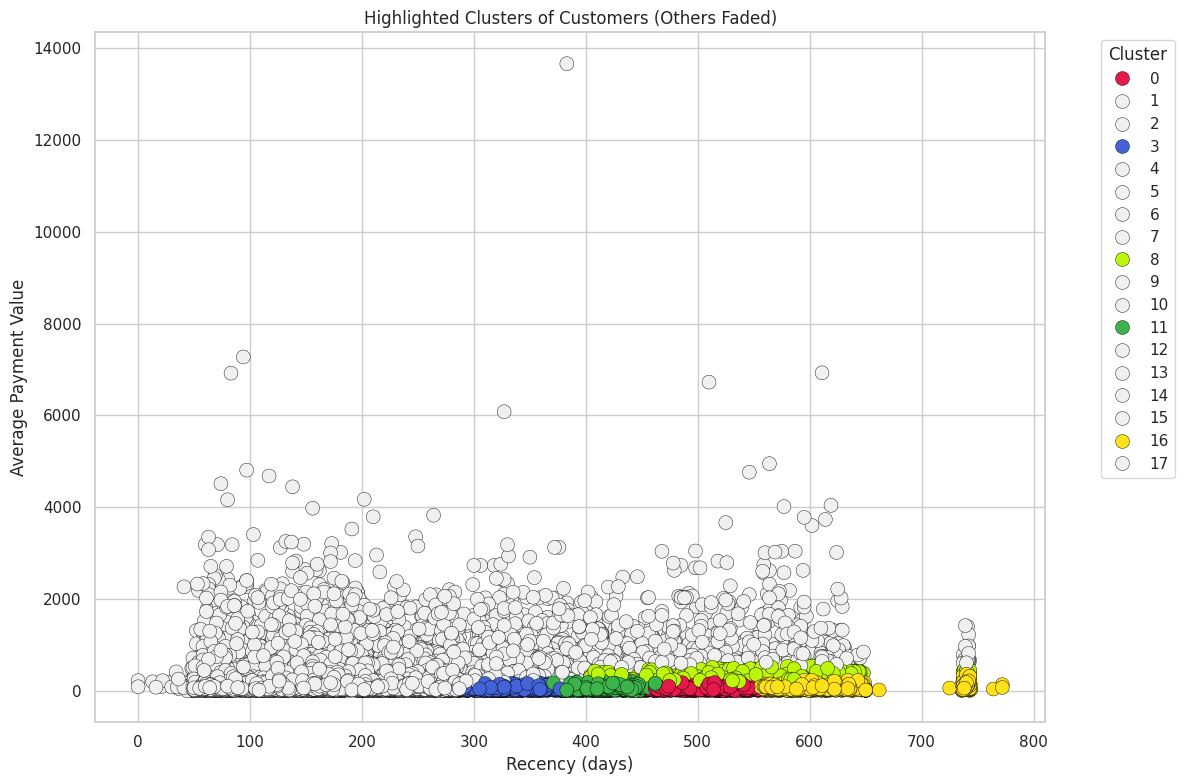


All of these belong to the same cluster as per RFM. Below is how k-means has segregated this.



It seems like too many clusters. But cluster 8,4,12,2,17 have too few datapoints

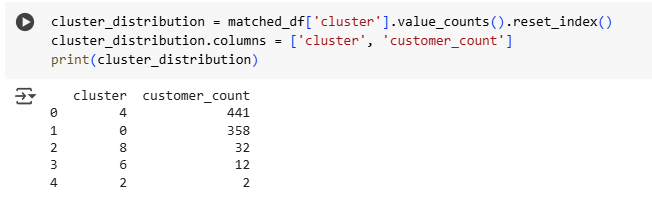
Let me plot this visually and see if the clusters are nearby.



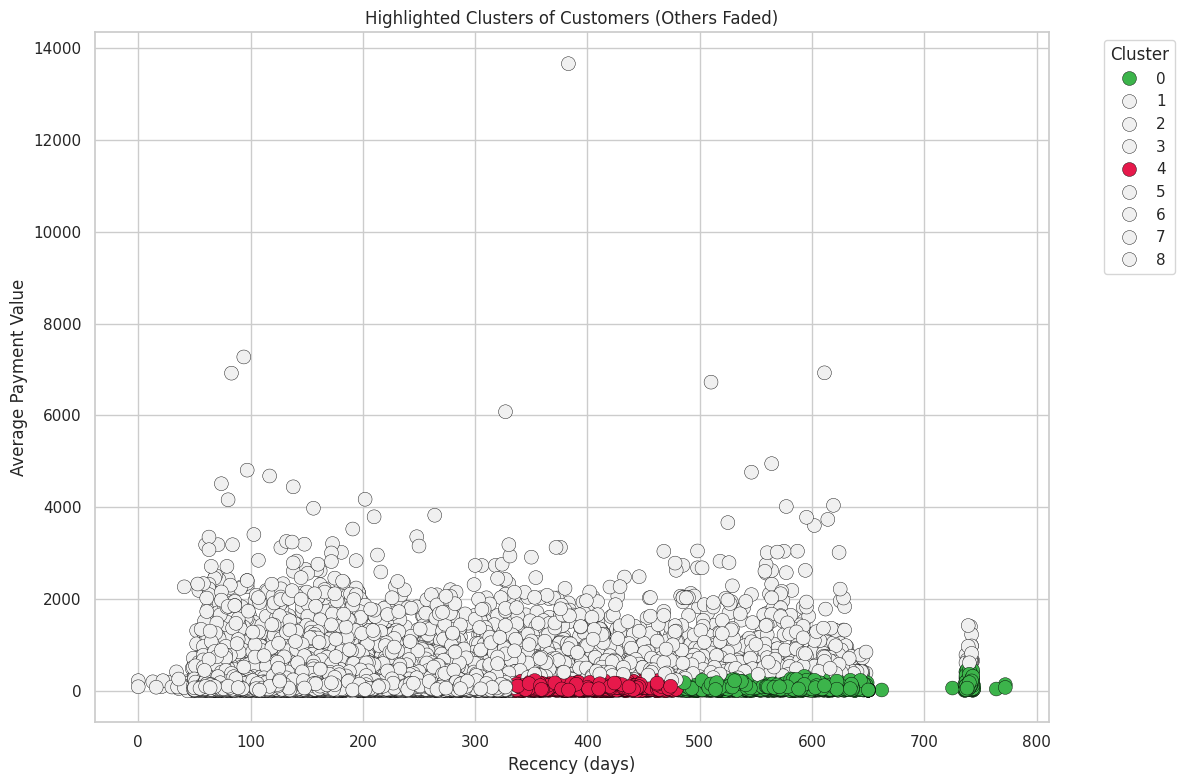
As we can see from the plot, all these are closer to each other. Also, the recency days are stretched. In RFM we have grouped into 3 parts, first 33%, next 33% and the remaining. But here it is more granular and shows the stretch of days for different groups.

Let me reduce the cluster count and see if that makes a difference.

With reduced cluster count below is what we get



Ignoring cluster 8,6,2, as these have few records, we get the below plot.



Here we see that these IDs are more together.

This demonstrates the power of machine learning. For RFM, I had to invest a considerable amount of time, but using the k-means clustering, I was able to get the clusters together more easily and in a way more accurately considering the granular way in which the clusters get formed.