TASK3

September 15, 2025

[1]:
```
import pandas as pd
import numpy as np
from datetime import timedelta
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
df = pd.read_csv('onlineretail.csv', encoding='ISO-8859-1')
df.head()
```

[2]:
```
  Invoice StockCode                          Description  Quantity  \
0  489434     85048  15CM CHRISTMAS GLASS BALL 20 LIGHTS      12.0
1  489434    79323P                   PINK CHERRY LIGHTS      12.0
2  489434    79323W                  WHITE CHERRY LIGHTS      12.0
3  489434     22041         RECORD FRAME 7" SINGLE SIZE       48.0
4  489434     21232       STRAWBERRY CERAMIC TRINKET BOX      24.0

           InvoiceDate  Price  Customer ID         Country
0  2009-12-01 07:45:00   6.95      13085.0  United Kingdom
1  2009-12-01 07:45:00   6.75      13085.0  United Kingdom
2  2009-12-01 07:45:00   6.75      13085.0  United Kingdom
3  2009-12-01 07:45:00   2.10      13085.0  United Kingdom
4  2009-12-01 07:45:00   1.25      13085.0  United Kingdom
```

[5]:
```
print("columns:", df.columns.tolist())

print([repr(c) for c in df.columns])

display(df.head())
df.info()
```

```
columns: ['Invoice', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
'Price', 'Customer ID', 'Country']
["'Invoice'", "'StockCode'", "'Description'", "'Quantity'", "'InvoiceDate'",
"'Price'", "'Customer ID'", "'Country'"]
```

```
  Invoice StockCode                          Description  Quantity  \
0  489434     85048  15CM CHRISTMAS GLASS BALL 20 LIGHTS      12.0
1  489434    79323P                   PINK CHERRY LIGHTS      12.0
```

```
2  489434    79323W                  WHITE CHERRY LIGHTS      12.0
3  489434     22041         RECORD FRAME 7" SINGLE SIZE       48.0
4  489434     21232       STRAWBERRY CERAMIC TRINKET BOX      24.0

           InvoiceDate  Price  Customer ID         Country
0  2009-12-01 07:45:00   6.95      13085.0  United Kingdom
1  2009-12-01 07:45:00   6.75      13085.0  United Kingdom
2  2009-12-01 07:45:00   6.75      13085.0  United Kingdom
3  2009-12-01 07:45:00   2.10      13085.0  United Kingdom
4  2009-12-01 07:45:00   1.25      13085.0  United Kingdom
<class 'pandas.core.frame.DataFrame'>
Index: 226510 entries, 0 to 235810
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Invoice      226510 non-null  object
 1   StockCode    226510 non-null  object
 2   Description  225863 non-null  object
 3   Quantity     226510 non-null  float64
 4   InvoiceDate  226510 non-null  object
 5   Price        226510 non-null  float64
 6   Customer ID  176162 non-null  float64
 7   Country      226510 non-null  object
dtypes: float64(3), object(5)
memory usage: 15.6+ MB
```

[7]:
```
df.rename(columns={"Customer ID": "CustomerID"}, inplace=True)
```

[8]:
```
df = df.drop_duplicates()
```

[9]:
```
df = df[df["Quantity"] > 0]
```

[10]:
```
df = df[df["CustomerID"].notnull()]
```

[11]:
```
df["InvoiceDate"] = pd.to_datetime(df["InvoiceDate"])
```

[12]:
```
df["TotalPrice"] = df["Quantity"] * df["Price"]
```

[13]:
```
snapshot_date = df["InvoiceDate"].max() + pd.Timedelta(days=1)
print("Snapshot date:", snapshot_date)
```

```
Snapshot date: 2010-06-15 13:03:00
```

[14]:
```
rfm = df.groupby("CustomerID").agg(
    Recency=("InvoiceDate", lambda x: (snapshot_date - x.max()).days),
    Frequency=("Invoice", "nunique"),
    Monetary=("TotalPrice", "sum")
).reset_index()

rfm.head()
```

[14]:
```
   CustomerID  Recency  Frequency  Monetary
0     12346.0      104         10    230.55
1     12349.0       28          2   1268.52
2     12355.0       25          1    488.21
3     12358.0        8          2   1697.93
4     12359.0       84          4   1522.23
```

[15]:
```
rfm["R_Score"] = pd.qcut(rfm["Recency"], 5, labels=[5,4,3,2,1]).astype(int)
rfm["F_Score"] = pd.qcut(rfm["Frequency"].rank(method="first"), 5,␣
 ↪labels=[1,2,3,4,5]).astype(int)
rfm["M_Score"] = pd.qcut(rfm["Monetary"].rank(method="first"), 5,␣
 ↪labels=[1,2,3,4,5]).astype(int)


rfm["RFM_Score"] = (
    rfm["R_Score"].astype(str) +
    rfm["F_Score"].astype(str) +
    rfm["M_Score"].astype(str)
)

rfm.head()
```

[15]:
```
   CustomerID  Recency  Frequency  Monetary  R_Score  F_Score  M_Score  \
0     12346.0      104         10    230.55        2        5        2
1     12349.0       28          2   1268.52        4        3        4
2     12355.0       25          1    488.21        4        1        3
3     12358.0        8          2   1697.93        5        3        5
4     12359.0       84          4   1522.23        2        4        5

  RFM_Score
0       252
1       434
2       413
3       535
4       245
```

[16]:
```
def segment(row):
    if row["R_Score"] >= 4 and row["F_Score"] >= 4 and row["M_Score"] >= 4:
        return "Champions"
    if row["R_Score"] >= 3 and row["F_Score"] >= 3:
        return "Loyal_Customers"
    if row["R_Score"] >= 4 and row["F_Score"] <= 2:
        return "New_Customers"
    if row["R_Score"] <= 2 and row["F_Score"] >= 3:
        return "At_Risk"
    if row["R_Score"] == 1:
        return "Lost"
    return "Others"

rfm["Segment"] = rfm.apply(segment, axis=1)


print(rfm["Segment"].value_counts())
```

```
Segment
Loyal_Customers    701
Champions          567
Others             480
At_Risk            435
Lost               403
New_Customers      252
Name: count, dtype: int64
```

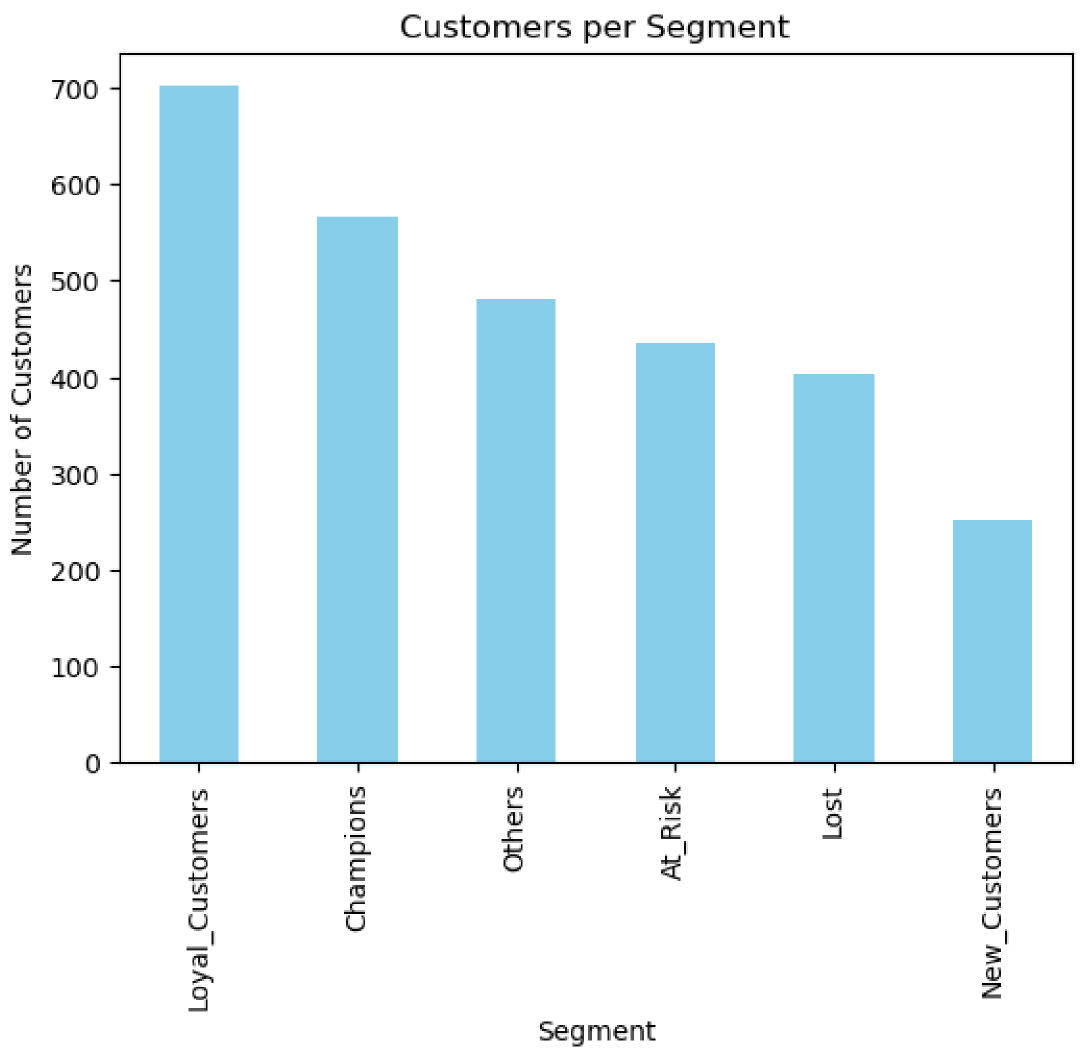
[17]:
```
seg_summary = rfm.groupby("Segment").agg({
    "CustomerID": "count",
    "Monetary": "mean"
}).rename(columns={"CustomerID": "Count", "Monetary": "Avg_Monetary"}).
 ↪reset_index()

print(seg_summary)
```

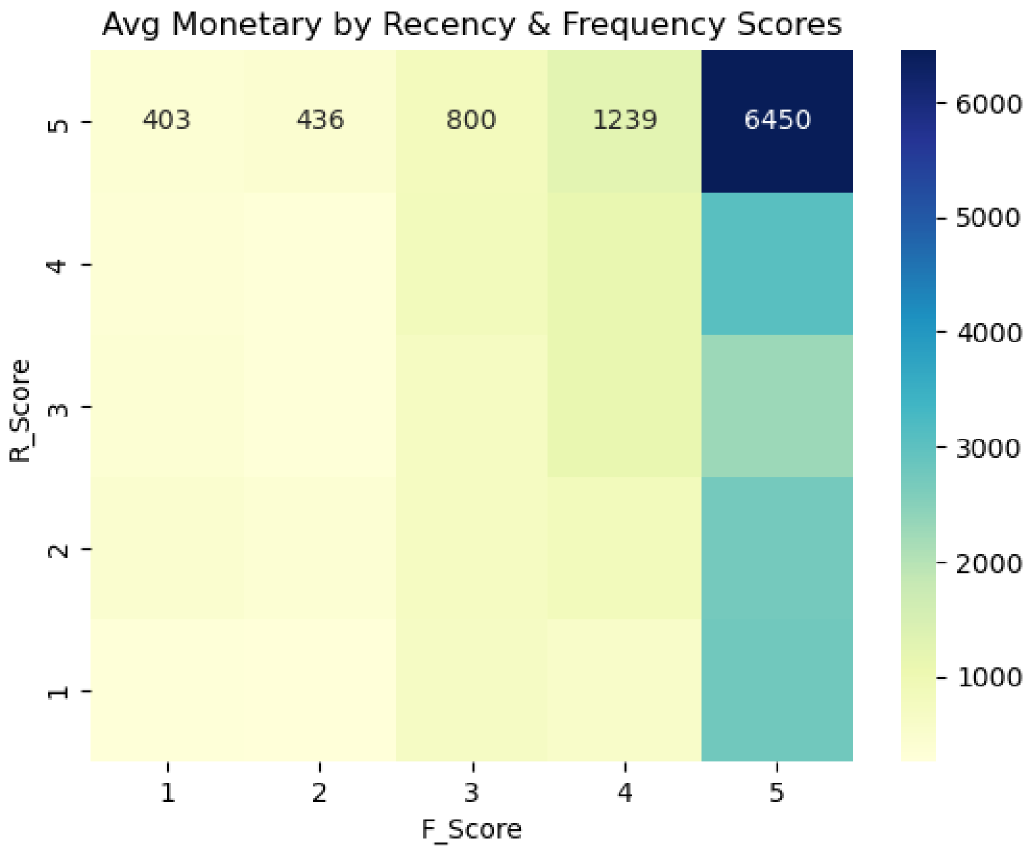
```
           Segment  Count  Avg_Monetary
0          At_Risk    435    966.167218
1        Champions    567   4256.631067
2             Lost    403    278.095660
3  Loyal_Customers    701    995.708234
4    New_Customers    252    361.420357
5           Others    480    406.099798
```

[20]:
```
# Bonus
rfm["Segment"].value_counts().plot(kind="bar", title="Customers per Segment",␣
 ↪color="skyblue")
plt.ylabel("Number of Customers")
plt.show()

pivot = rfm.pivot_table(index="R_Score", columns="F_Score", values="Monetary",␣
 ↪aggfunc="mean")
pivot = pivot.sort_index(ascending=False)
sns.heatmap(pivot, annot=True, fmt=".0f", cmap="YlGnBu")
plt.title("Avg Monetary by Recency & Frequency Scores")
plt.show()
```



Customers per Segment
Number of Customers
700
600
500
400
300
200
100
0
Loyal_Customers
Champions
Others
At_Risk
Lost
New_Customers
Segment



```
[19]: rfm.to_csv("rfm_results.csv", index=False)
      print("RFM results saved to rfm_results.csv")
```

```
RFM results saved to rfm_results.csv
```

```
[ ]:
```