```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_auc_score, roc_curve


from sklearn.cluster import KMeans
from mlxtend.frequent_patterns import apriori, association_rules


import warnings
warnings.filterwarnings('ignore', category=DeprecationWarning)
```

```
churn_df = pd.read_csv('/content/WA_Fn-UseC_-Telco-Customer-Churn.csv')
retail_df = pd.read_excel('/content/online_retail_II.xlsx')


print("Churn Data Sample:")
display(churn_df.head())

print("\nOnline Retail Sample:")
display(retail_df.head())
```

Churn Data Sample:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...

5 rows × 21 columns

Online Retail Sample:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45:00	2.10	13085.0	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom

Question 1 – Customer Churn Prediction

Data Preprocessing

```
churn_df['TotalCharges'] = churn_df['TotalCharges'].replace(' ', np.nan)
churn_df.dropna(subset=['TotalCharges'], inplace=True)
churn_df['TotalCharges'] = churn_df['TotalCharges'].astyp  oat)
```

```
for col in churn_df.columns:
    if churn_df[col].dtype == 'object':
        le = LabelEncoder()
        churn_df[col] = le.fit_transform(churn_df[col])


X = churn_df.drop('Churn', axis=1)
y = churn_df['Churn']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

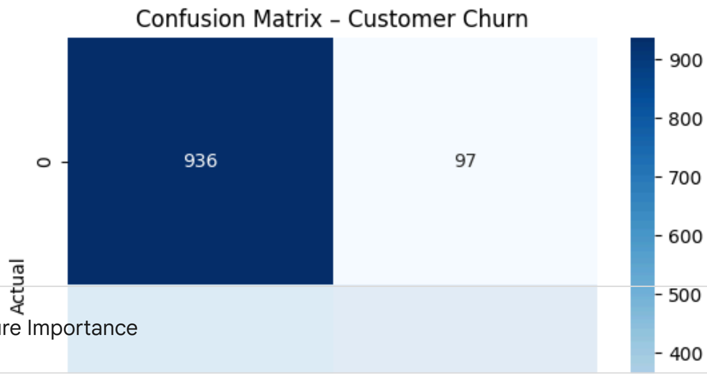
Model Training (Random Forest)

```
rf_model = RandomForestClassifier(n_estimators=200, random_state=42)
rf_model.fit(X_train, y_train)
y_pred = rf_model.predict(X_test)
```

Evaluation + Visuals

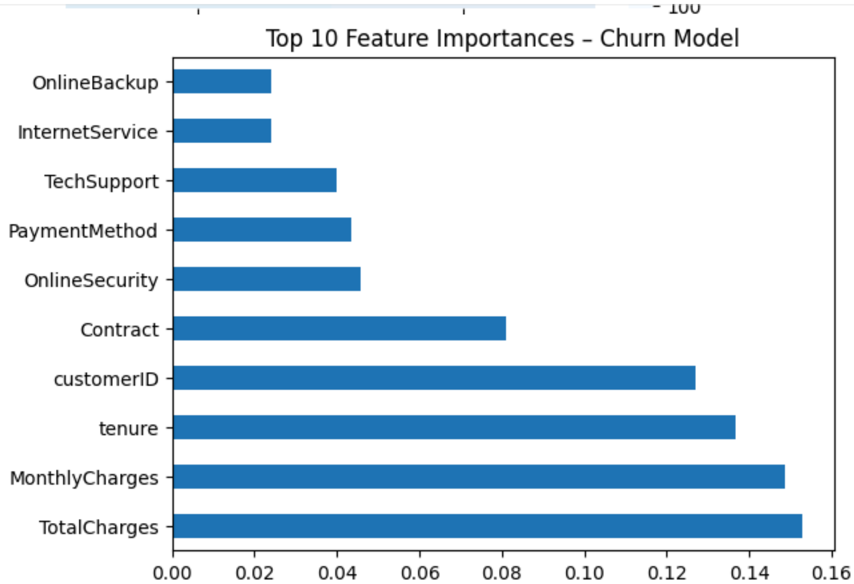
```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))


cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix – Customer Churn')
plt.xlabel('Predicted'); plt.ylabel('Actual'); plt.show()


y_pred_proba = rf_model.predict_proba(X_test)[::,1]
fpr, tpr, _ = roc_curve(y_test,  y_pred_proba)
plt.plot(fpr,tpr,label="AUC="+str(round(roc_auc_score(y_test, y_pred_proba),2)))
plt.legend(); plt.title('ROC Curve – Churn Prediction'); plt.show()
```

```
Accuracy: 0.7917555081734187

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.91      0.86      1033
           1       0.65      0.48      0.55       374

    accuracy                           0.79      1407
   macro avg       0.74      0.69      0.71      1407
weighted avg       0.78      0.79      0.78      1407
```



Feature Importance

```
importances = pd.Series(rf_model.feature_importances_, index=X.columns).sort_values(ascending=False)[:10]
importances.plot(kind='barh')
plt.title('Top 10 Feature Importances – Churn Model')
plt.show()
```



Question 2 – Customer Segmentation (Online Retail)

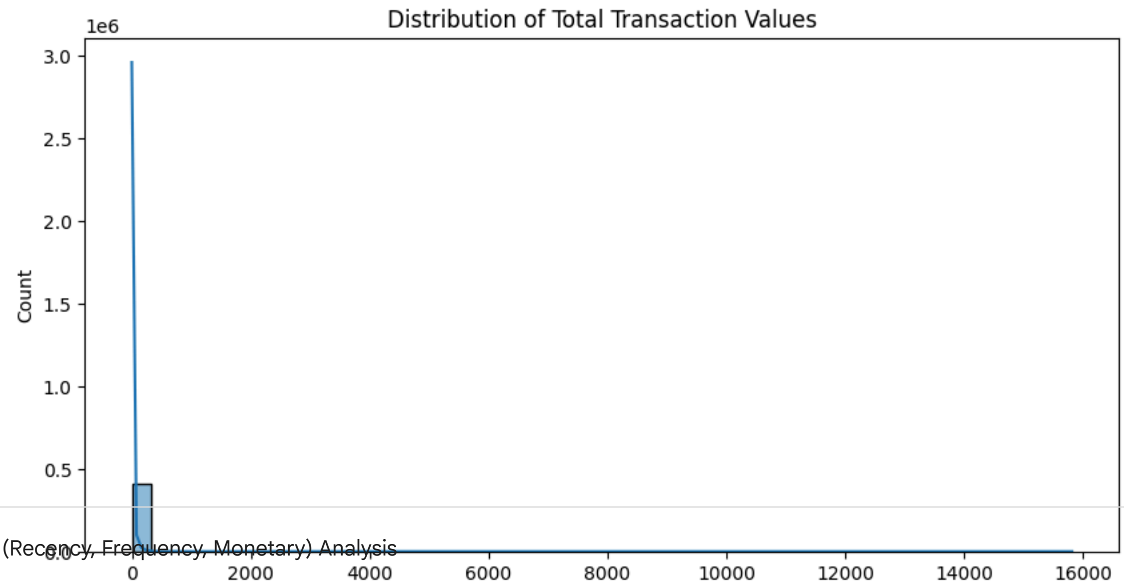
Data Cleaning

```
print(retail_df.info())
print(retail_df.head())


retail_df = retail_df.dropna(subset=['Customer ID', 'Description'])


retail_df['Customer ID'] = retail_df['Customer ID'].astype(int)


retail_df = retail_df[retail_df['Quantity'] > 0]


retail_df['totalprice'] = retail_df['Quantity'] * retail_df['Price']
```

```
retail_df['totalprice'] = retail_df['Quantity'] * retail_df['Price']


print("After cleaning:")
print(retail_df.describe(include='all'))
print(retail_df.isnull().sum())
```

```
1  489434    79323P                  PINK CHERRY LIGHTS        12
2  489434    79323W                 WHITE CHERRY LIGHTS        12
3  489434     22041         RECORD FRAME 7" SINGLE SIZE        48
4  489434     21232       STRAWBERRY CERAMIC TRINKET BOX       24

          InvoiceDate  Price  Customer ID         Country  totalprice
0 2009-12-01 07:45:00   6.95        13085  United Kingdom        83.4
1 2009-12-01 07:45:00   6.75        13085  United Kingdom        81.0
2 2009-12-01 07:45:00   6.75        13085  United Kingdom        81.0
3 2009-12-01 07:45:00   2.10        13085  United Kingdom       100.8
4 2009-12-01 07:45:00   1.25        13085  United Kingdom        30.0
After cleaning:
         Invoice StockCode                         Description       Quantity  \
count   407695.0    407695                              407695  407695.000000
unique   19215.0      4017                                4444            NaN
top     500356.0    85123A  WHITE HANGING HEART T-LIGHT HOLDER            NaN
freq       270.0      3153                                3153            NaN
mean         NaN       NaN                                 NaN      13.586686
min          NaN       NaN                                 NaN       1.000000
25%          NaN       NaN                                 NaN       2.000000
50%          NaN       NaN                                 NaN       5.000000
75%          NaN       NaN                                 NaN      12.000000
max          NaN       NaN                                 NaN   19152.000000
std          NaN       NaN                                 NaN      96.842229

                          InvoiceDate          Price    Customer ID  \
count                          407695  407695.000000  407695.000000
unique                            NaN            NaN            NaN
top                               NaN            NaN            NaN
freq                              NaN            NaN            NaN
mean    2010-07-01 10:10:10.782177792       3.294188   15368.504107
min               2009-12-01 07:45:00       0.000000   12346.000000
25%               2010-03-26 14:01:00       1.250000   13997.000000
50%               2010-07-09 15:46:00       1.950000   15321.000000
75%               2010-10-14 17:09:00       3.750000   16812.000000
max               2010-12-09 20:01:00   10953.500000   18287.000000
std                               NaN      34.756655    1679.795700

               Country     totalprice
count           407695  407695.000000
unique              37            NaN
top     United Kingdom            NaN
freq            370951            NaN
mean               NaN      21.663261
min                NaN       0.000000
25%                NaN       4.950000
50%                NaN      11.900000
75%                NaN      19.500000
max                NaN   15818.400000
std                NaN      77.147356
Invoice        0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
Price          0
Customer ID    0
Country        0
totalprice     0
dtype: int64
```

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10,5))
sns.histplot(retail_df['totalprice'], bins=50, kde=True)
plt.title('Distribution of Total Transaction Values')
plt.xlabel('Total Price per Invoice Line')
plt.ylabel('Count')
plt.show()
```



RFM (Recency, Frequency, Monetary) Analysis

```
rfm = retail_df.groupby('customerid').agg({
    'invoicedate': lambda x: (retail_df['invoicedate'].max() - x.max()).days,
    'invoice': 'count',
    'totalprice': 'sum'
}).rename(columns={'invoicedate': 'recency', 'invoice': 'frequency', 'totalprice': 'monetary'})

rfm.reset_index(inplace=True)
rfm.head()
```

	Customer ID	recency	frequency	monetary
0	12346	164	33	372.86
1	12347	2	71	1323.32
2	12348	73	20	222.16
3	12349	42	102	2671.14
4	12351	10	21	300.93

Next steps: Generate code with rfm New interactive sheet

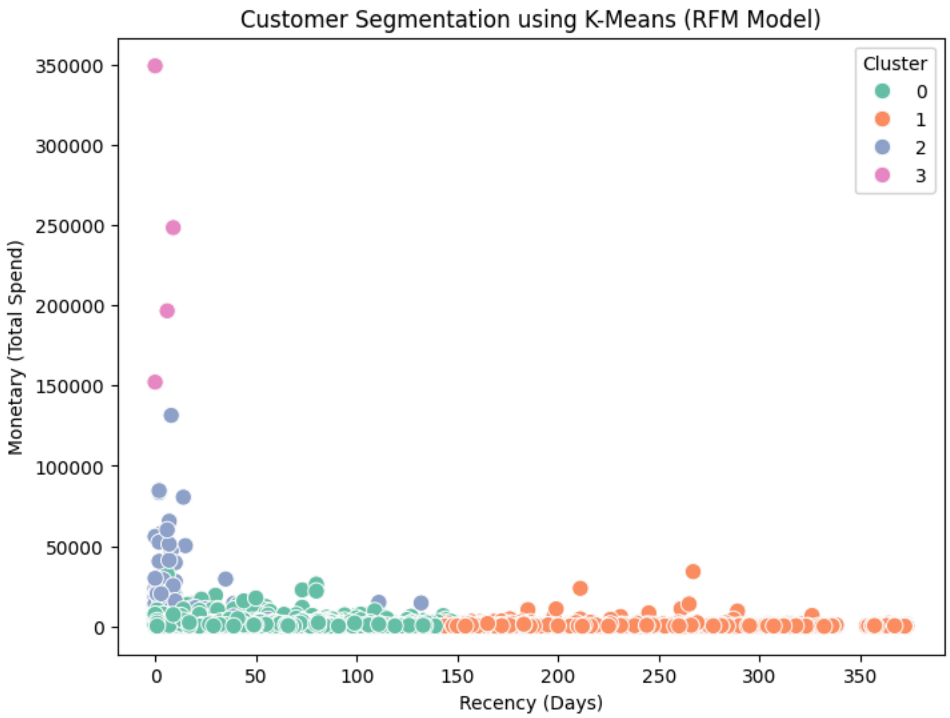
: Scale & Cluster (K-Means)

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans


scaler = StandardScaler()
rfm_scaled = scaler.fit_transform(rfm[['recency', 'frequency', 'monetary']])


kmeans = KMeans(n_clusters=4, random_state=42)
rfm['cluster'] = kmeans.fit_predict(rfm_scaled)


print(rfm.groupby('cluster').mean())
```

```
         Customer ID     recency    frequency       monetary
cluster
0       15334.840728   43.057772    84.812959    1592.729466
1       15403.817308  243.004808    29.875962     614.363956
2       15249.846715   13.000000   732.014599   16472.760460
3       15453.750000    3.750000  2654.750000  236568.790000
```

: Visualize Clusters

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))
sns.scatterplot(data=rfm, x='recency', y='monetary', hue='cluster', palette='Set2', s=80)
plt.title('Customer Segmentation using K-Means (RFM Model)')
plt.xlabel('Recency (Days)')
plt.ylabel('Monetary (Total Spend)')
plt.legend(title='Cluster')
plt.show()
```



Question 3 – Cross-Selling Opportunities

Prepare Basket Data

Gemini

```
basket = (retail_df
          .groupby(['Invoice', 'description'])['quantity']
          .groupby(['Invoice', 'Description'])['Quantity']
          .sum().unstack().fillna(0))


basket = basket.applymap(lambda x: 1 if x > 0 else 0)


print("Basket data shape:", basket.shape)
basket.head()
```

```
/tmp/ipython-input-2011280971.py:6: FutureWarning: DataFrame.applymap has been deprecated. Use DataFrame.map instead.
  basket = basket.applymap(lambda x: 1 if x > 0 else 0)
Basket data shape: (19215, 4444)
```

Description	DOORMAT UNION JACK GUNS AND ROSES	3 STRIPEY MICE FELTCRAFT	4 PURPLE FLOCK DINNER CANDLES	ANIMAL STICKERS	BLACK PIRATE TREASURE CHEST	BROWN PIRATE TREASURE CHEST	Bank Charges	CAMPHOR WOOD PORTOBELLO MUSHROOM	CHERRY BLOSSOM DECORATIVE FLASK	FAIRY CAKE CANDLES	...	ZINC HEART LATTICE CHARGER LARGE	Z HE LATT CHAR SM
Invoice													
489437	0	0	0	0	0	0	0	0	0	0	...	0	
489438	0	0	0	0	0	0	0	0	0	0		0	

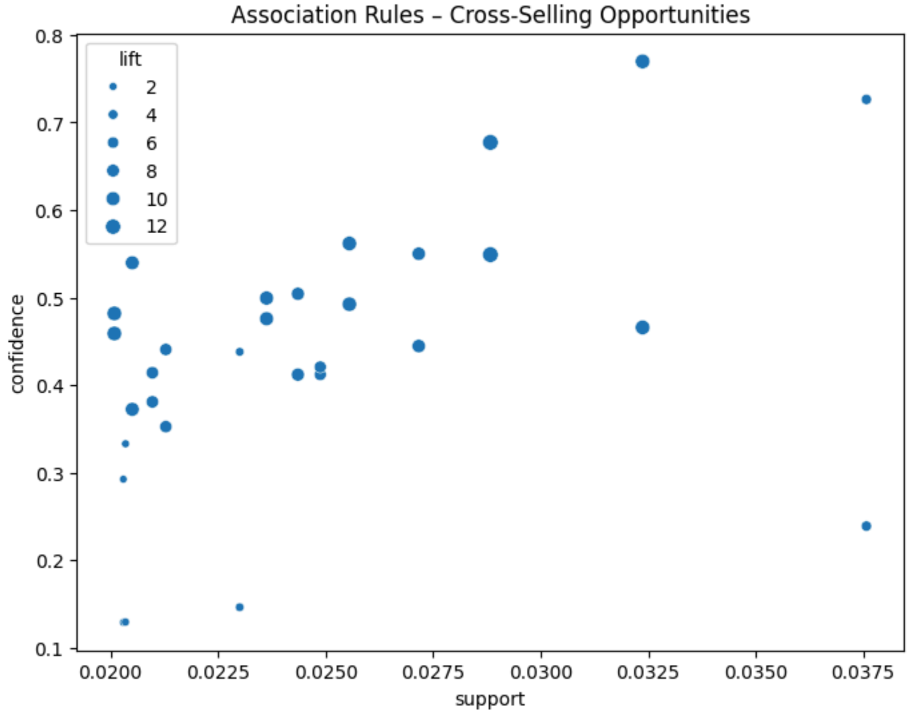
Association Rule Mining

```
frequent_items = apriori(basket, min_support=0.02, use_colnames=True)
rules = association_rules(frequent_items, metric="lift", min_threshold=1)
rules.sort_values('lift', ascending=False).head(10)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	representativity	leverage	conviction	z
28	(WOODEN FRAME ANTIQUE WHITE)	(WOODEN PICTURE FRAME WHITE FINISH)	0.052511	0.042571	0.028832	0.549058	12.897504	1.0	0.026596	2.123178	
29	(WOODEN PICTURE FRAME WHITE FINISH)	(WOODEN FRAME ANTIQUE WHITE)	0.042571	0.052511	0.028832	0.677262	12.897504	1.0	0.026596	2.935780	
25	(SWEETHEART CERAMIC TRINKET BOX)	(STRAWBERRY CERAMIC TRINKET BOX)	0.042050	0.069477	0.032371	0.769802	11.079959	1.0	0.029449	4.042272	
24	(STRAWBERRY CERAMIC TRINKET BOX)	(SWEETHEART CERAMIC TRINKET BOX)	0.069477	0.042050	0.032371	0.465918	11.079959	1.0	0.029449	1.793636	
7	(CHOCOLATE HOT WATER BOTTLE)	(HOT WATER BOTTLE TEA AND SYMPATHY)	0.041686	0.043768	0.020088	0.481898	11.010301	1.0	0.018264	1.845643	
6	(HOT WATER BOTTLE TEA AND SYMPATHY)	(CHOCOLATE HOT WATER BOTTLE)	0.043768	0.041686	0.020088	0.458977	11.010301	1.0	0.018264	1.771301	
	(HEART OF	(HEART OF									

Visualization

```
plt.figure(figsize=(8,6))
sns.scatterplot(x='support', y='confidence', size='lift', data=rules)
plt.title('Association Rules - Cross-Selling Opportunities')
plt.show()
```



Combined Summary Output

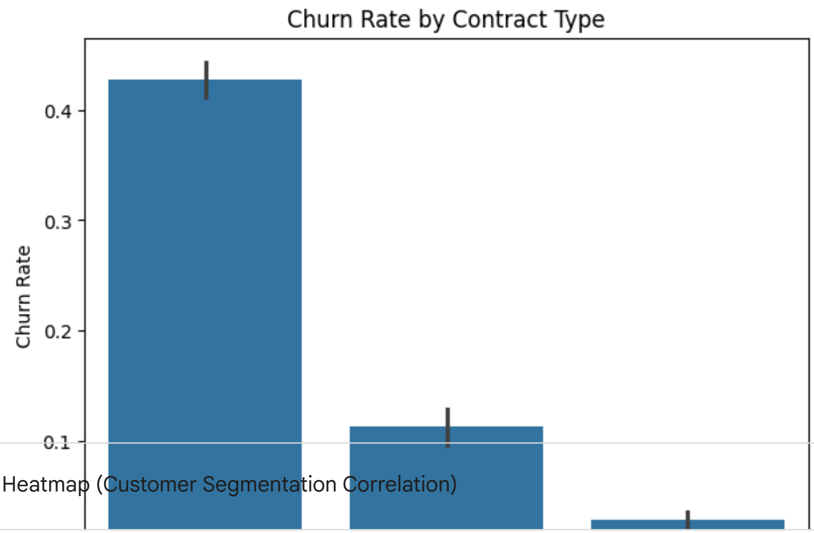
```
summary = pd.DataFrame({
    'Question': ['Churn Prediction', 'Customer Segmentation', 'Cross-Selling'],
    'Algorithm': ['Random Forest', 'K-Means', 'Apriori'],
    'Goal': ['Predict churn likelihood', 'Group customers by RFM behavior', 'Find product associations'],
    'Key Metric': ['Accuracy/AUC', 'Cluster Means', 'Lift/Support']
})
display(summary)
```

	Question	Algorithm	Goal	Key Metric
0	Churn Prediction	Random Forest	Predict churn likelihood	Accuracy/AUC
1	Customer Segmentation	K-Means	Group customers by RFM behavior	Cluster Means
2	Cross-Selling	Apriori	Find product associations	Lift/Support

Next steps: Generate code with summary | New interactive sheet

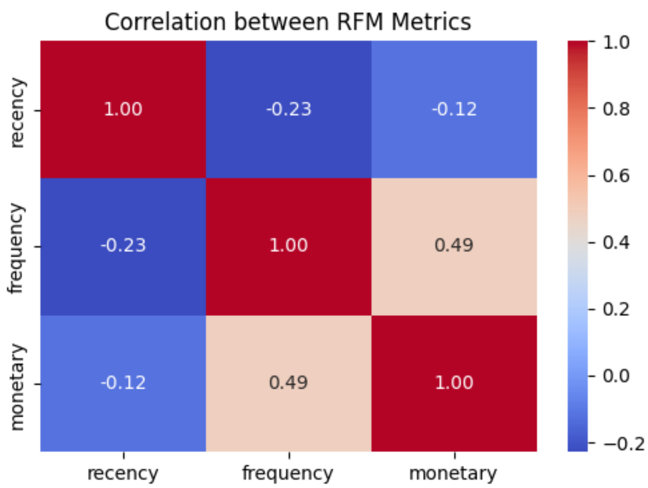
Churn Rate by Contract Type

```
import seaborn as sns
import matplotlib.pyplot as plt


churn_raw = pd.read_csv('/content/WA_Fn-UseC_-Telco-Customer-Churn.csv')

plt.figure(figsize=(7,5))
sns.barplot(data=churn_raw, x='Contract', y=churn_raw['Churn'].apply(lambda x: 1 if x=='Yes' else 0))
plt.title('Churn Rate by Contract Type')
plt.ylabel('Churn Rate')
plt.xlabel('Contract Type')
plt.show()
```



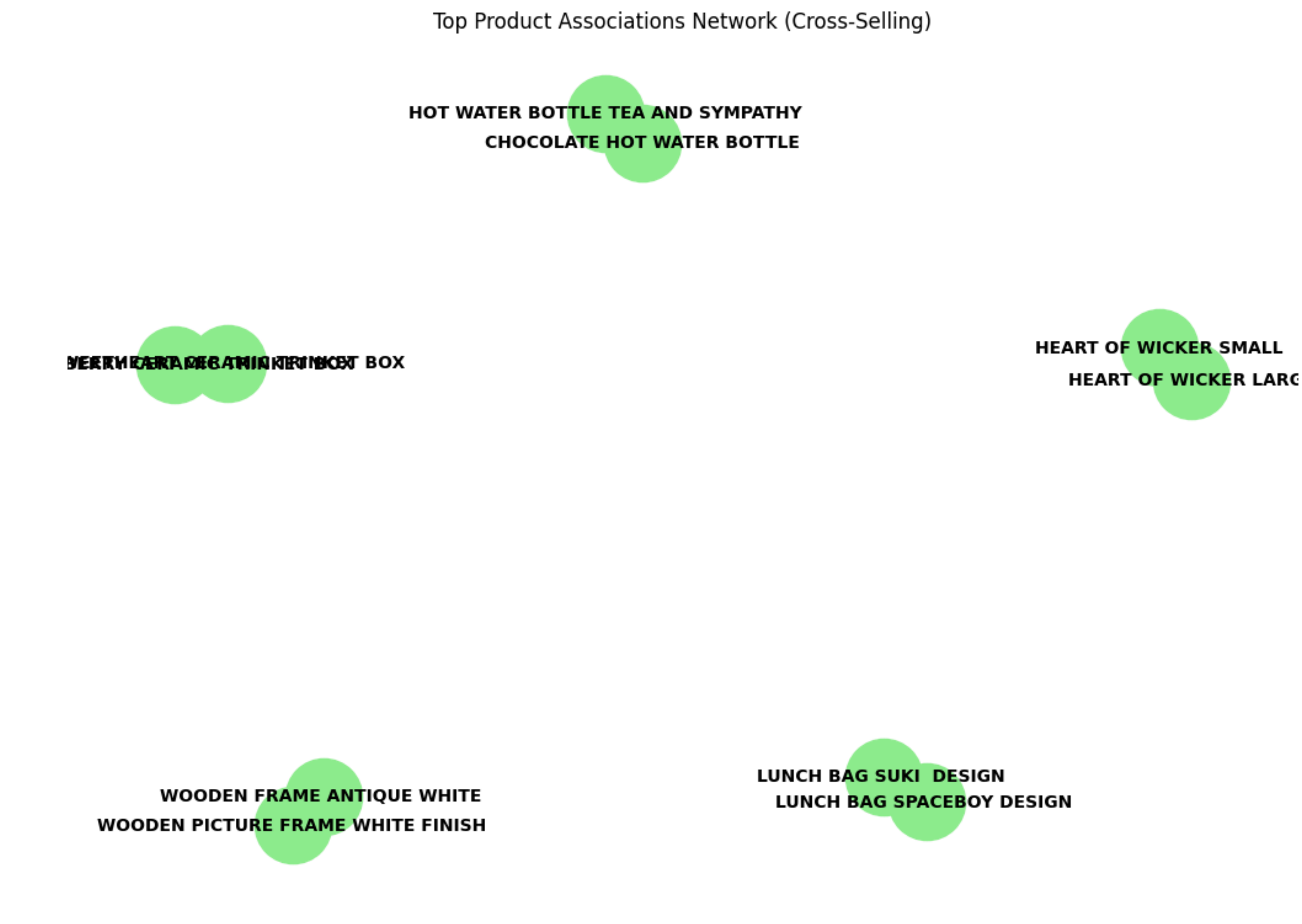
RFM Heatmap (Customer Segmentation Correlation)

```
plt.figure(figsize=(6,4))
sns.heatmap(rfm[['recency','frequency','monetary']].corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation between RFM Metrics')
plt.show()
```



Top Product Co-Occurrences Network (Cross-Selling)

```
import networkx as nx


top_rules = rules.sort_values('lift', ascending=False).head(10)


G = nx.Graph()
for _, row in top_rules.iterrows():
    for a in row['antecedents']:
        for b in row['consequents']:
            G.add_edge(a, b, weight=row['lift'])


plt.figure(figsize=(10,7))
pos = nx.spring_layout(G, k=0.5)
nx.draw(G, pos,
        with_labels=True,
        node_size=2000,
        node_color='lightgreen',
        font_size=10,
        font_weight='bold',
        edge_color='gray')
plt.title('Top Product Associations Network (Cross-Selling)')
plt.show()
```


Top Product Associations Network (Cross-Selling)
HOT WATER BOTTLE TEA AND SYMPATHY
CHOCOLATE HOT WATER BOTTLE
HEART OF WICKER SMALL
HEART OF WICKER LARG
WOODEN FRAME ANTIQUE WHITE
WOODEN PICTURE FRAME WHITE FINISH
LUNCH BAG SUKI DESIGN
LUNCH BAG SPACEBOY DESIGN