#import libraries

import pandas as pd

import numpy as np

import matplotlib as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

#load dataset

df = pd.read\_excel('/content/drive/MyDrive/dataset1/Online Retail.xlsx')

df

df.shape

df.dtypes

# 1a. Missing Data Treatment:

# Check for missing data

missing\_data = df.isnull().sum()

# percentage of missing data

percentage\_missing = (missing\_data / len(df)) \* 100

# Create a dataframe for analysis

missing\_info = pd.DataFrame({

'Missing Values': missing\_data,

'Percentage': percentage\_missing

}).sort\_values(by='Percentage', ascending=False)

missing\_info.head()

# 1b. Removing Duplicate Data:

import seaborn as sns

import matplotlib.pyplot as plt

# Create a heatmap to visualize null data

plt.figure(figsize=(12, 8))

sns.heatmap(df.isnull(), cbar=False, cmap='viridis')

plt.title('Heatmap de Dados Nulos')

plt.show()

# Check and remove duplicate data

duplicated\_data = df[df.duplicated()]

# Check and remove duplicate data

df = df.drop\_duplicates()

# Check again for duplicates after removal

duplicated\_data\_after\_removal = df[df.duplicated()]

# Display the number of duplicates removed

len(duplicated\_data)

I decided to fill in null 'CustomerID' values with N/A because I wanted to keep them in the dataset but differentiate them from those that have 'CustomerID' valid.and I did two types, removing and not removing the null data, there was no difference.

I decided to fill in the null data in the 'Description' column with 'NO SOURCE' because there was little and it didn't make much difference in the data analysis

df.dropna(subset=['CustomerID'], inplace=True)

df['Description'].fillna('NO SOURCE', inplace=True)

df['CustomerID'].fillna('N/A', inplace=True)

df['Description'].fillna('NO SOURCE', inplace=True)

# 1c. Descriptive Analysis:

# Perform descriptive analysis

descriptive\_stats = df.describe(include='all')

# Display descriptive statistics

descriptive\_stats

#'################################################'

# STEP 1 . finished

#'################################################'

2. Cohort Analysis: A cohort is a group of subjects who share a defining characteristic. We can observe how a cohort behaves across time and compare it to other cohorts.

# 2a. a. Create month cohorts and analyse active customers for each cohort.

# Create a Cohort Column:

df['InvoiceMonth'] = df['InvoiceDate'].dt.to\_period('M')

df['CohortMonth'] = df.groupby('CustomerID')['InvoiceDate'].transform('min').dt.to\_period('M')

# Calculate the Period since Acquisition:

df['CohortIndex'] = ((df['InvoiceDate'] - df['CohortMonth'].dt.to\_timestamp()).dt.days // 30)

# Create a retention table

retention\_table = df.pivot\_table(index='CohortMonth', columns='CohortIndex', values='CustomerID', aggfunc=pd.Series.nunique)

# Calculate retention rate

cohort\_size = retention\_table.iloc[:, 0]

retention\_matrix = retention\_table.divide(cohort\_size, axis=0)

# Retention table

print(retention\_matrix)

# Heatmap using seaborn

plt.figure(figsize=(12, 8))

sns.heatmap(retention\_table / 1000, annot=True, fmt=".0%", cmap="YlGnBu", vmin=0.0, vmax=1.0)

plt.title('Retention Table')

plt.xlabel('Period since Acquisition')

plt.ylabel('Cohort')

plt.show()

# 2b. Also Analyse the retention rate of customers. Comment.

# Calculate the average retention rate per period:

retention\_rate = retention\_table.mean(axis=0)

# View retention rate over time:

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

plt.plot(retention\_rate.index, retention\_rate.values, marker='o')

plt.title('Average Customer Retention Rate Over Time')

plt.xlabel('Cohort Index')

plt.ylabel('Retention Rate')

plt.grid(True)

plt.show()

#\* Strong startand then a strong decay: Since at the beginning all customers are retained, it starts high, but soon drops, which could mean a few things like:

Unmet expectations: Customers may have initial expectations that are not met after the first interaction with the product or service.

Quality problems: If there are quality problems with the products or services, customers may be disappointed and choose not to continue.

Inadequate communication: If communication about the product or service is unclear, customers may become confused or dissatisfied.

Competition: If customers find more attractive offers from competitors, they may choose to switch quickly.

#\* Recovery: As retention strategies, special offers, or other initiatives are implemented, the retention rate may begin to recover

#\* Stability: Over time, the retention rate may stabilize at a more sustainable level.

#'################################################'

# STEP 2 . finished

#'################################################'

3. Build a RFM model – Recency Frequency and Monetary based on their behaviour.

Recency is about when was the last order of a customer. It means the number of days since a customer made the last purchase. If it’s a case for a website or an app, this could be interpreted as the last visit day or the last login time

# 1. Recency

max\_date = df['InvoiceDate'].max()

df['Recency'] = (max\_date - df['InvoiceDate']).dt.days

# 2. Frequency

frequency\_df = df.groupby('CustomerID')['InvoiceNo'].nunique().reset\_index()

frequency\_df.columns = ['CustomerID', 'Frequency']

df = pd.merge(df, frequency\_df, on='CustomerID', how='left')

# 3. Monetary

monetary\_df = df.groupby('CustomerID')['UnitPrice'].sum().reset\_index()

monetary\_df.columns = ['CustomerID', 'Monetary']

df = pd.merge(df, monetary\_df, on='CustomerID', how='left')

# 4. Segmentation (using k-means, for example)

from sklearn.cluster import KMeans

features = df[['Recency', 'Frequency', 'Monetary']]

kmeans = KMeans(n\_clusters=3, random\_state=42)

df['RFM\_Segment'] = kmeans.fit\_predict(features)

# Results

print(df[['CustomerID', 'Recency', 'Frequency', 'Monetary', 'RFM\_Segment']])

df

# dataset to see frequency for each user

# Calculate the Frequency for each customer:

frequency\_df = df.groupby('CustomerID')['InvoiceNo'].nunique().reset\_index()

frequency\_df.columns = ['CustomerID', 'Frequency']

frequency\_df

#'################################################'

# End of the project thank you very much

#'################################################'

from google.colab import files

files.download('meu\_dataset\_limpo.csv')