import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

import plotly.express as px

import warnings

warnings.filterwarnings('ignore')

import os

for dirname, \_, filenames in os.walk('/kaggle/input'):

for filename in filenames:

print(os.path.join(dirname, filename))

df=pd.read\_csv('/kaggle/input/market-basket-analysis/Assignment-1\_Data.csv',delimiter=';')

df.head()

df.info()

df.isnull().sum()

df.loc[df['Quantity']<=0][:5]

df=df.loc[df['Quantity']>0]

df.loc[df['Price']<='0'][:5

]

df=df.loc[df['Price']>'0']

df.loc[(df['Itemname']=='POSTAGE')|(df['Itemname']=='DOTCOM POSTAGE')|(df['Itemname']=='Adjust bad debt')|(df['Itemname']=='Manual')].head()

df.isnull().sum()

df=df.fillna('-')

df.isnull().sum()

df['Year']=df['Date'].apply(lambda x:x.split('.')[2])

df['Year']=df['Year'].apply(lambda x:x.split(' ')[0])

df['Month']=df['Date'].apply(lambda x:x.split('.')[1])

df.head()

df['Price']=df['Price'].str.replace(',','.').astype('float64')

df['Total price']=df.Quantity\*df.Price

df.head()

df.groupby(['Year','Month'])['Total price'].sum()

df=df.loc[df['Year']!='2010']

sales=df.groupby(['Year','Month'])['Total price','Quantity'].sum()

sales.to\_csv('sales.csv')

sales=pd.read\_csv('sales.csv')

sales=sales.pivot\_table(sales,index=['Year','Month'],aggfunc=np.sum,fill\_value=0)

sales.plot(kind='bar',cmap='Set1')

plt.show()

sales\_country=df.groupby(['Year','Month','Country'])['Total price'].sum()

sales\_country.to\_csv('sales\_country.csv')

sales\_country=pd.read\_csv('sales\_country.csv')

fig=px.bar(sales\_country,x=['Month'],y='Total price',color='Country',title='Monthly sales amount in each country in 2021')

fig.update\_layout(xaxis\_title='Month',yaxis\_title='Sales amount')

fig.show()

country=df.groupby('Country')['Total price'].sum()

country.to\_csv('country.csv')

country=pd.read\_csv('country.csv')

fig=px.bar(country,x='Country',y='Total price',title='Sales amount in each country in 2021')

fig.update\_layout(xaxis={'categoryorder':'total descending'},yaxis\_title='Sales amount')

fig.show()

cm=sns.light\_palette("green",as\_cmap=True)

item\_sales=df.groupby('Itemname')['Price'].sum().sort\_values(ascending=False)[:10]

item\_sales.to\_csv('item\_sales.csv')

item\_sales=pd.read\_csv('item\_sales.csv')

item\_sales.style.background\_gradient(cmap=cm).set\_precision(2)

df[['Itemname','Quantity']].sort\_values(by='Quantity',ascending=False)[:10].style.background\_gradient(cmap=cm).set\_precision(2)

color=plt.cm.rainbow(np.linspace(0,1,30))

df['Itemname'].value\_counts().head(10).plot.bar(color=color,figsize=(6,3))

plt.xticks(rotation=90,fontsize=8)

plt.grid()

plt.show()

from mlxtend.frequent\_patterns import apriori

from mlxtend.frequent\_patterns import association\_rules

df['Itemname']=df['Itemname'].str.strip()

df['BillNo']=df['BillNo'].astype('str')

basket=(df[df['Country']=='United Kingdom']

.groupby(['BillNo','Itemname'])['Quantity']

.sum().unstack().reset\_index().fillna(0)

.set\_index('BillNo'))

basket.head(3)

def encode\_units(x):

if x<=0:

return 0

if x>=1:

return 1

basket\_sets=basket.applymap(encode\_units)

frequent\_itemsets=apriori(basket\_sets,min\_support=0.03,use\_colnames=True)

rules=round(association\_rules(frequent\_itemsets,metric='lift',min\_threshold=1),2)

rules.head(5)

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

plt.subplot(221)

sns.scatterplot(x="support",y="confidence",data=rules,hue="lift",palette="viridis")

plt.subplot(222)

sns.scatterplot(x="support",y="lift",data=rules,hue="confidence",palette="viridis")

plt.subplot(223)

sns.scatterplot(x="confidence",y="lift",data=rules,hue='support',palette="viridis")

plt.subplot(224)

sns.scatterplot(x="antecedent support",y="consequent support",data=rules,hue='confidence',palette="viridis")

plt.tight\_layout()

plt.show()

rules[['antecedents','consequents','support']].sort\_values('support',ascending=False)[:5].style.background\_gradient(cmap=cm).set\_precision(2)

rules[['antecedents','consequents','confidence']].sort\_values('confidence',ascending=False)[:5].style.background\_gradient(cmap=cm).set\_precision(2)

rules[['antecedents','consequents','lift']].sort\_values('lift',ascending=False)[:5].style.background\_gradient(cmap=cm).set\_precision(2)

rules[(rules['lift']>=13)&(rules['confidence']>=0.7)].sort\_values('lift',ascending=False).style.background\_gradient(cmap=cm).set\_precision(2)