```
import pandas as pd
import numpy as np
import random as rn
import sklearn as sk
```

```
data=np.array([[ 1001 , 33 , "Male" , "Married" , 65000 , 18 , "Electronics"],
[1002 , 28 , "Female" , "Single" , 45000 , 15 , "Appliances"] ,
[1003 , 42 , "Male" , "Single" , 55000 , 20 , "Electronics" ],
[1004 , 51 , "Female" , "Married" , 80000 , 12 , "Electronics"]
, [1005 , 37 , "Male" , "Divorced" , 58000 , 10 , "Appliances" ]])
columns=[ 'CustomerID' , 'Age' , 'Gender' , 'MaritalStatus' , 'AnnualIncome (USD)' , 'TotalPurchases' , 'PreferredCategory' ]
Data = pd.DataFrame(data,columns=columns)
```

```
#Now we will perform encoding first
from sklearn.preprocessing import LabelEncoder
Encoder1=LabelEncoder()
Encoder2=LabelEncoder()
Encoder3=LabelEncoder()
Data['Gender']=Encoder1.fit_transform(Data['Gender'])
Data['MaritalStatus']=Encoder2.fit_transform(Data['MaritalStatus'])
Data['PreferredCategory']=Encoder3.fit_transform(Data['PreferredCategory'])
```

```
Data['MaritalStatus'].max(),Data['PreferredCategory'].max(),Data['Gender'].max(),Data['TotalPurchases'].min(),Data['Age'].min()
```

```
(2, 1, 1, '10', '28')
```

```
list1=np.random.randint(low=20,high=65,size=(495))
list2=np.random.choice(Data['Gender'],size=(495))
list3=np.random.choice(Data['MaritalStatus'],size=(495))
list4=np.random.randint(low=Data['AnnualIncome (USD)'].min(),high=Data['AnnualIncome (USD)'].max(),size=(495))
list5=np.random.randint(low=5,high=50,size=(495))
list6=np.random.choice(Data['PreferredCategory'],size=(495))
```

```
List1=pd.DataFrame(list1,columns=['Age'])
List2=pd.DataFrame(list2,columns=['Gender'])
List3=pd.DataFrame(list3,columns=['MaritalStatus'])
List4=pd.DataFrame(list4,columns=['AnnualIncome (USD)'])
List5=pd.DataFrame(list5,columns=['TotalPurchases'])
List6=pd.DataFrame(list6,columns=['PreferredCategory'])
```

```
#As Customer ID do not help in doing any analysis
Data=Data.iloc[:,1:]
Data
```

```
#To concat we must get index start from 5
ind_list=list(range(5,500))
List1.index=ind_list
List2.index=ind_list
List3.index=ind_list
List4.index=ind_list
List5.index=ind_list
List6.index=ind_list
```

```
result = pd.concat([List1,List2,List3,List4,List5,List6], axis=1, join='inner')
```

```
Data=pd.concat([Data,result])
```

```
Data
```

```
Data['Gender']=Encoder1.inverse_transform(Data['Gender'])
Data['MaritalStatus']=Encoder2.inverse_transform(Data['MaritalStatus'])
Data['PreferredCategory']=Encoder3.inverse_transform(Data['PreferredCategory'])
```

```
Data
```

```
#Now we will do plottings
from matplotlib import pyplot as plt
```

```
import seaborn as sns
#First to see ratio of male and females wrt to thier status
sns.countplot(x='Gender',hue='MaritalStatus',data=Data)
plt.show()
```

```
#First to see ratio of male and females wrt to choice of appliances
sns.countplot(x='Gender',hue='PreferredCategory',data=Data)
plt.show()
```

```
Dat=Data.loc[:,["Age","TotalPurchases"]].values
#Now first we do K means clusstering for knowing No. of purchases wrt Annual budget
from sklearn.cluster import KMeans
#First we will find elbow point through plottign between wcss and K clusters
Wcss=[]
for k in range(1,11):
    kmeans=KMeans(n_clusters=k,init="k-means++")
    kmeans.fit(Dat)
    Wcss.append(kmeans.inertia_)
plt.figure(figsize=(12,6))
plt.grid()
plt.plot(range(1,11),Wcss,linewidth=2,color="red",marker="8")
plt.xlabel("K_cluster")
plt.ylabel("Wcss")
plt.show()
```

```
#So we see our elbow point is 5
km=KMeans(n_clusters=5,init="k-means++")
km.fit(Dat)
plt.scatter(Dat[:,0].astype(float),Dat[:,1].astype(float),c=km.labels_,cmap='rainbow')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='black')
plt.xlabel("Age")
plt.ylabel("Purchases")
plt.show()
```

```
#Now for creating interactive dashboard
import dash
from dash import Dash, dcc, html, Input, Output, callback
import dash_bootstrap_components as dbc
from flask import Flask
import dash_bootstrap_components as dbc
import pandas as pd
import plotly.express as px
import plotly.graph_objects as grp


# Build the Components
Header_component1 = html.H1("Tech Electro Incorporates")
Header_component2 = html.H2("Product Priorities wrt Gender")
Header_component3 = html.H3("Purchases wrt Marital Status")
Header_component4 = html.H4("No of Clusters")


# Initiate the App
server=Flask (__name__)
app = dash.Dash (__name__, server = server, external_stylesheets=[dbc.themes. BOOTSTRAP, dbc.icons.BOOTSTRAP])


# Design the app layout
variant1 = Data['Gender'].unique()
variant2 = Data['MaritalStatus'].unique()
variant3=list(range(1,6))
app.layout = html.Div([
dbc.Row([Header_component1]),
dbc.Row([Header_component2,
dcc.Dropdown(
id='type-dropdown1',
options=[{'label': v, 'value': v} for v in variant1],
value=variant1[0]
),
dcc.Graph(id='graph-with-slider1')
]),
dbc.Row([Header_component3,
dcc.Dropdown(
id='type-dropdown2',
options=[{'label': v, 'value': v} for v in variant2],
value=variant2[0]
),
dcc.Graph(id='graph-with-slider2')
]),dbc.Row([Header_component4,
           dcc.Dropdown(
id='type-dropdown3',
options=[{'label': v, 'value': v} for v in variant3],
value=variant3[0]
           ),
           dcc.Graph(id='graph-with-slider3')
           ])
])
@callback(
Output('graph-with-slider1', 'figure'),
Input('type-dropdown1', 'value'))
def DashUpdater_1(Choice1):
    New_Data = Data[Data['Gender'] == Choice1]
    New_count = New_Data['PreferredCategory'].value_counts()
    figur = px.bar(New_count, x=New_count.index, y=New_count.values)
    return figur

@callback(
Output('graph-with-slider2', 'figure'),
Input('type-dropdown2', 'value'))
def DashUpdater_1(Choice2):
    New_Data2 = Data[Data['MaritalStatus'] == Choice2]
    New_count2 = New_Data2['TotalPurchases'].value_counts()
    figur = px.bar(New_count2, x=New_count2.index, y=New_count2.values)
    return figur
@callback(
Output('graph-with-slider3', 'figure'),
Input('type-dropdown3', 'value'))
def DashUpdater_1(Choice3):
    km=KMeans(n_clusters=Choice3,init="k-means++")
    km.fit(Dat)
    figur=px.scatter(Dat,x=Dat[:,0].astype(float),y=Dat[:,1].astype(float),color=km.labels_)
    return figur



# Run the App
app.run_server (debug=True)
```

Colab paid products - Cancel contracts here