Exercise 2.2: Graph Analysis with Matplotlib

#1: Introduction of the Dataset

The Dataset opted for this exercise is "Marketing Campaign" dataset from Kaggle. A response model can provide a significant boost to the efficiency of a marketing campaign by increasing responses or reducing expenses. The objective is to predict who will respond to an offer for a product or service via a detailed analysis of this dataset which might provide good insights for the business to make an informed decision. The dataset has a total of 29 attributes covering customers info, products, promotions and sales.

#2: Question - Who are the potential customers to respond to an offer ?

In [19]:
```
# Import required libraries

import warnings
import numpy as np
import pandas as pd
from datetime import date
import plotly as py
import seaborn as sns
import plotly.express as px
import plotly.graph_objs as go
warnings.filterwarnings("ignore")
pd.set_option('display.max_columns', None)
import matplotlib.lines as lines
import matplotlib.pyplot as plt

#Load the "Marketing Campaign" dataset from the downloaded file (Source: Kaggle
customers = pd.read_csv("/Users/anjanibonda/Data-Science/DSC550/Week2_Graph_Ana
```

Examine and understand the Data

In [2]:
```
customers.head()
```

Out[2]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	I
0	5524	1957	Graduation	Single	58138.0	0	0	2012-09-04	
1	2174	1954	Graduation	Single	46344.0	1	1	2014-03-08	
2	4141	1965	Graduation	Together	71613.0	0	0	2013-08-21	
3	6182	1984	Graduation	Together	26646.0	1	0	2014-02-10	
4	5324	1981	PhD	Married	58293.0	1	0	2014-01-19	

In [3]:
```
customers.tail()
```

Out[3]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Custom
2235	10870	1967	Graduation	Married	61223.0	0	1	2013-06-
2236	4001	1946	PhD	Together	64014.0	2	1	2014-06-
2237	7270	1981	Graduation	Divorced	56981.0	0	0	2014-01-
2238	8235	1956	Master	Together	69245.0	0	1	2014-01-
2239	9405	1954	PhD	Married	52869.0	1	1	2012-10-

In [4]:
```
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   ID                   2240 non-null   int64  
 1   Year_Birth           2240 non-null   int64  
 2   Education            2240 non-null   object 
 3   Marital_Status       2240 non-null   object 
 4   Income               2216 non-null   float64
 5   Kidhome              2240 non-null   int64  
 6   Teenhome             2240 non-null   int64  
 7   Dt_Customer          2240 non-null   object 
 8   Recency              2240 non-null   int64  
 9   MntWines             2240 non-null   int64  
 10  MntFruits            2240 non-null   int64  
 11  MntMeatProducts      2240 non-null   int64  
 12  MntFishProducts      2240 non-null   int64  
 13  MntSweetProducts     2240 non-null   int64  
 14  MntGoldProds         2240 non-null   int64  
 15  NumDealsPurchases    2240 non-null   int64  
 16  NumWebPurchases      2240 non-null   int64  
 17  NumCatalogPurchases  2240 non-null   int64  
 18  NumStorePurchases    2240 non-null   int64  
 19  NumWebVisitsMonth    2240 non-null   int64  
 20  AcceptedCmp3         2240 non-null   int64  
 21  AcceptedCmp4         2240 non-null   int64  
 22  AcceptedCmp5         2240 non-null   int64  
 23  AcceptedCmp1         2240 non-null   int64  
 24  AcceptedCmp2         2240 non-null   int64  
 25  Complain             2240 non-null   int64  
 26  Z_CostContact        2240 non-null   int64  
 27  Z_Revenue            2240 non-null   int64  
 28  Response             2240 non-null   int64  
dtypes: float64(1), int64(25), object(3)
memory usage: 507.6+ KB
```

Examine the marital status

In [20]:
```
customers.Marital_Status.unique()
```

Out[20]:
```
array(['Single', 'Together', 'Married', 'Divorced', 'Widow', 'Alone',
       'Absurd', 'YOLO'], dtype=object)
```

Lets merge 'Kidhome' and 'Teenhome' into one column 'Kids' for simplicity

In [21]:
```
customers['Kids'] = customers['Kidhome'] + customers['Teenhome']
```

Lets simplify further and categorize the data into 2 categories - "In Relation" and "Not in Relation"

In [22]:
```
map_status = {
    'Single' : 'Not in Relation',
    'Together': 'In Relation',
    'Married' : 'In Relation',
    'Divorced' : 'Not in Relation',
    'Widow' : 'Not in Relation',
    'Alone' : 'Not in Relation',
    'Absurd' : 'Not in Relation',
    'YOLO' : 'Not in Relation'
}

customers.Marital_Status = customers.Marital_Status.map(map_status)
```

Based on Marital Status, adjust the Family_Size for every row in the dataset.

In [23]:
```
customers['Family_Size'] = 0
for i in range(len(customers)):
    if customers['Marital_Status'][i] == 'In Relation':
        customers['Family_Size'][i] = customers['Kids'][i]  + 2 # Add 2 parents
    else:
        customers['Family_Size'][i] = customers['Kids'][i] + 1 # Add single par
```

In [24]:
```
# Examine the Data availability (End Date)

pd.to_datetime(customers['Dt_Customer']).unique().max()
```

Out[24]:
```
numpy.datetime64('2014-06-29T00:00:00.000000000')
```

In [25]:
```
# Calculate Customer's Age based on end date of dataset from above and Calculat

customers['Age'] = 2014 - customers['Year_Birth']
customers['Total_purchases_sum'] = customers['MntWines'] + customers['MntFruits
+ customers['MntMeatProducts'] + customers['MntFishProducts']
+ customers['MntSweetProducts']+ customers['MntGoldProds']

customers['Total_purchases_amount'] = customers['NumCatalogPurchases']
+customers['NumStorePurchases'] + customers['NumWebPurchases']

customers['TotalCmp'] = customers['AcceptedCmp1'] + customers['AcceptedCmp2']
customers['AcceptedCmp3'] + customers['AcceptedCmp4'] +customers['AcceptedCmp5'
```

Out[25]:
```
0       1
1       0
2       0
3       0
4       0
       ..
2235    0
2236    0
2237    1
2238    0
2239    1
Length: 2240, dtype: int64
```

In [26]:
```
# Drop redundant columns

customers.drop(columns=['ID', 'MntWines', 'MntFruits',
                        'MntMeatProducts', 'MntFishProducts',
                        'MntSweetProducts', 'MntGoldProds',
                        'NumCatalogPurchases', 'NumStorePurchases', 'NumWebPurc
                        'Kidhome', 'Teenhome', 'Marital_Status',
                        'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5',
                        'AcceptedCmp1','AcceptedCmp2'], inplace=True)
```

In [27]:
```
# Examine missing/null values or in other words, find outliers

customers.isnull().sum()
```

Out[27]:
```
Year_Birth                0
Education                 0
Income                   24
Dt_Customer               0
Recency                   0
NumDealsPurchases         0
NumWebVisitsMonth         0
Complain                  0
Z_CostContact             0
Z_Revenue                 0
Response                  0
Kids                      0
Family_Size               0
Age                       0
Total_purchases_sum       0
Total_purchases_amount    0
TotalCmp                  0
dtype: int64
```

In [14]:
```
# Replace missing values for income with mean

customers['Income'].fillna(customers['Income'].mean(), inplace=True)
```

In [15]:
```
customers.head()
```

Out[15]:

	Year_Birth	Education	Income	Dt_Customer	Recency	NumDealsPurchases	NumWebVisits
0	1957	Graduation	58138.0	2012-09-04	58	3	
1	1954	Graduation	46344.0	2014-03-08	38	2	
2	1965	Graduation	71613.0	2013-08-21	26	1	
3	1984	Graduation	26646.0	2014-02-10	26	2	
4	1981	PhD	58293.0	2014-01-19	94	5	

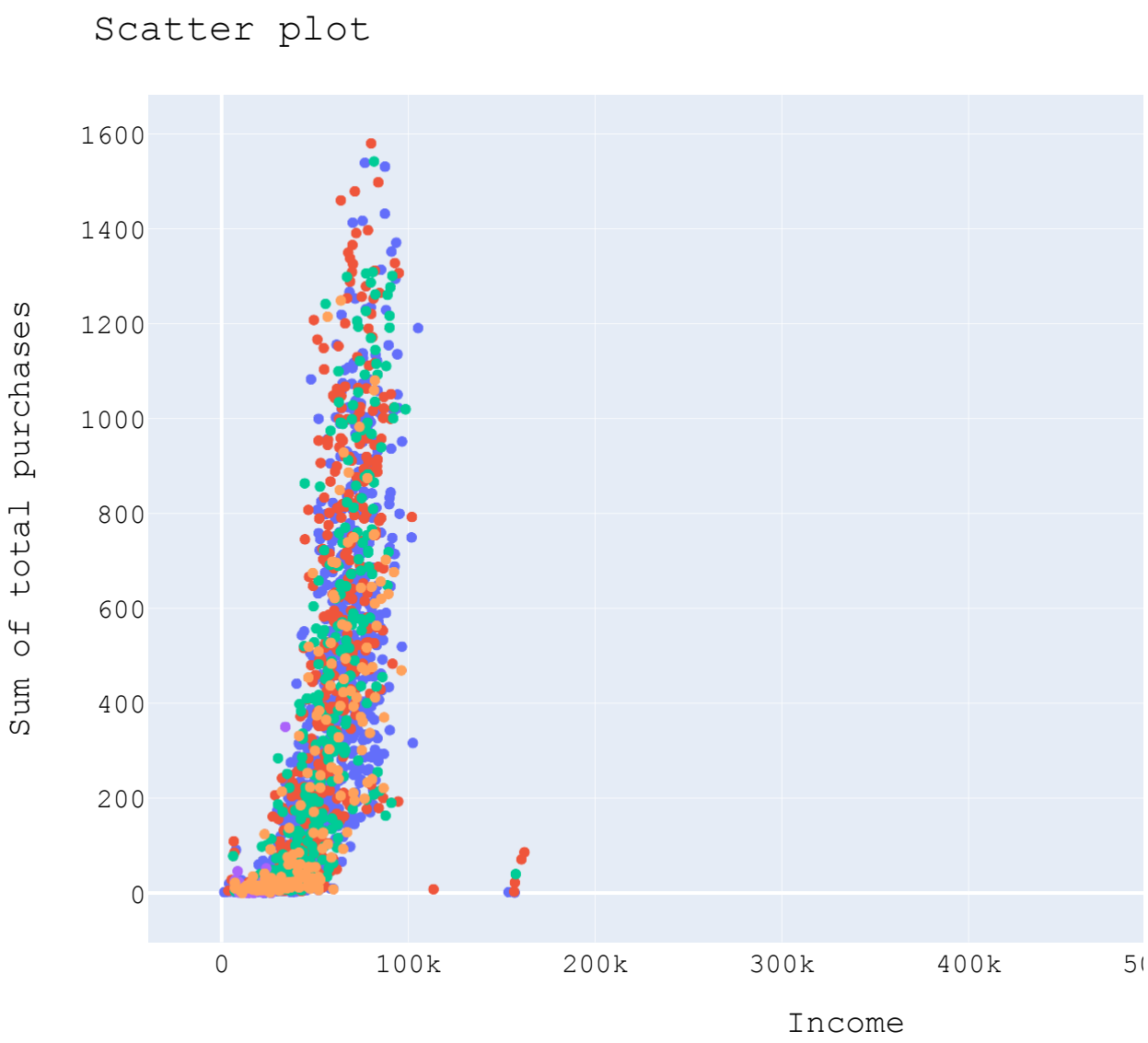
#3: Graph1 - Plot a Scatter plot between 'Income' and 'Total_purchases_sum' (Amount spent).

In [16]:

```
fig = px.scatter(customers, x='Income', y='Total_purchases_sum',
                 color = 'Education')

fig.update_layout(height=600, width=1000, title_text='Scatter plot',
                  yaxis_title="Sum of total purchases",font=dict(
        family="Courier New, monospace",
        size=15,
        color="black"
    ))

fig.show()
```



#4: Graph1 Observations:

Above graph suggests that there's a clear pattern between income and total purchases which are directly propotional (i.e., As one increases, other increases and vice-versa). However, same is not evident with respect to Eduation and Income.

#3: Graph2 - Plot a Bar Graph using Family_Size, Education and Total Purchases

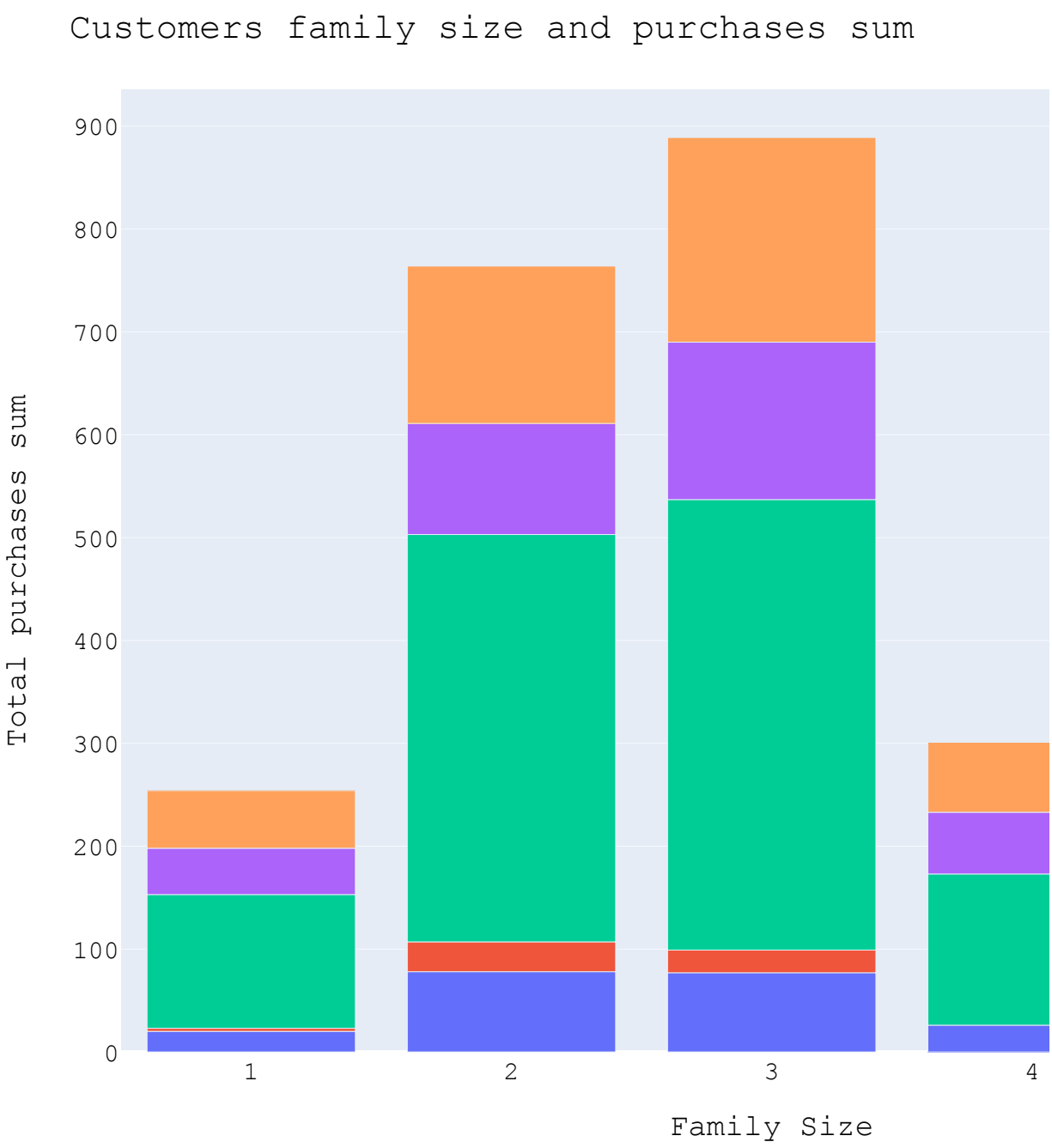
In [17]:
```
groupby_family_size = customers.groupby(['Family_Size', 'Education']).count()['

fig = px.bar(x = groupby_family_size.index.get_level_values(0) ,y = groupby_fam
             color = groupby_family_size.index.get_level_values(1),width = 100
             labels=dict(x="Family Size", y="Sum of total purchases", color="Ed
)

fig.update_layout(title_text='Customers family size and purchases sum',
```

```
                    xaxis_title='Family Size',
                    yaxis_title='Total purchases sum',
                    legend_title = 'Education',
                  font=dict(
        family="Courier New, monospace",
        size=15,
        color="black")
)

fig.show()
```



#4: Graph2 Observations:

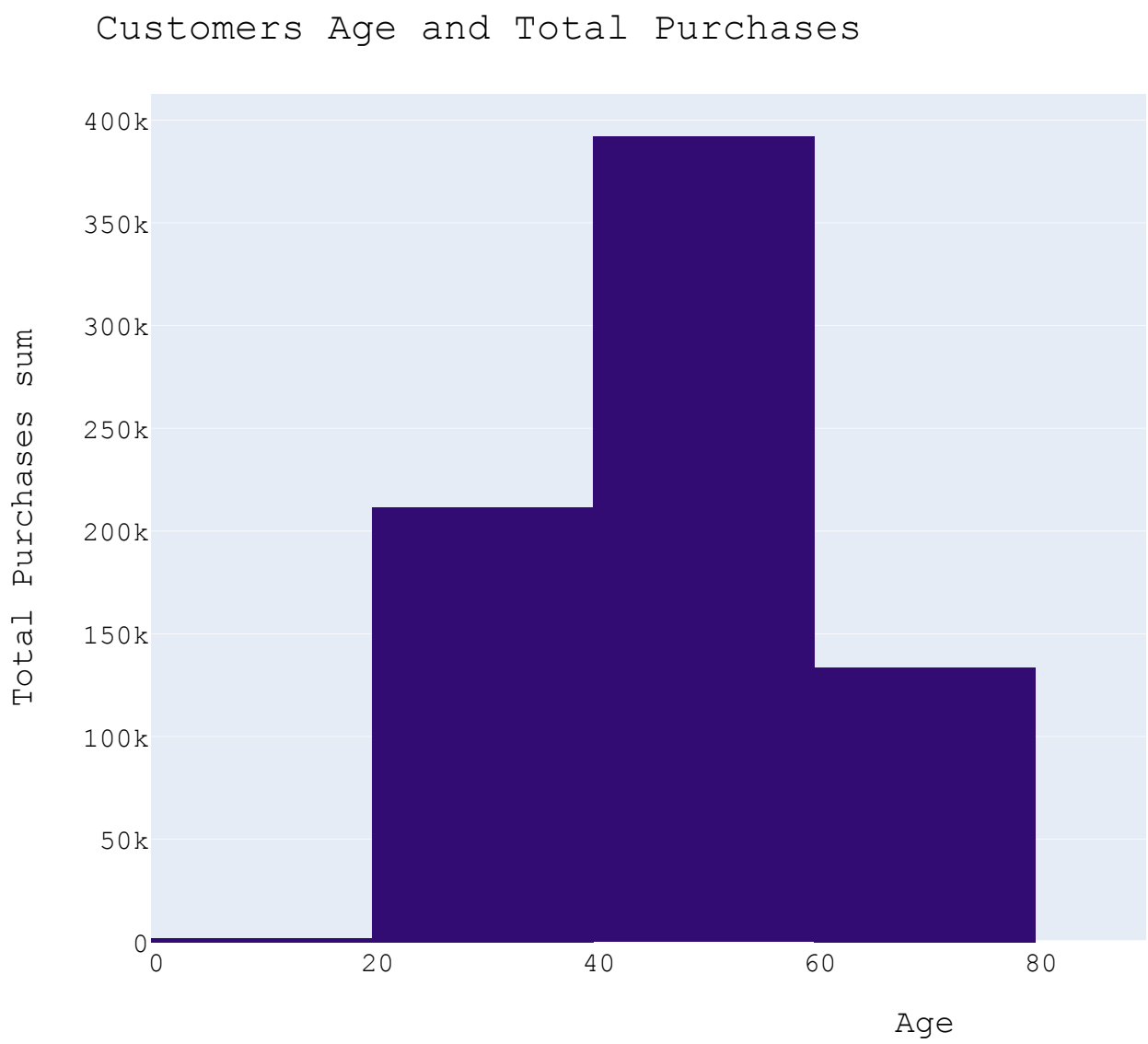
Above graph suggests that the potential customers are most likely the couples with either no kids or a single kid at max and most of them usually have a graduation degree.

#3: Graph3 - Plot a Bar Graph using Age and Total Purchases

In [17]:
```
fig = px.histogram(data_frame=customers, x='Age', y='Total_purchases_sum',heigh
                   color_discrete_sequence=['#330C73'])

fig.update_layout(title_text='Customers Age and Total Purchases',
                    xaxis_title='Age',
                    yaxis_title='Total Purchases sum',
                  font=dict(
        family="Courier New, monospace",
        size=15,
        color="black")
)

fig.show()
```



#4: Graph3 Observations:

Above graph suggests that the customes of Age 20-80 are the ones with good potential.

#5: Conclusion

Based on above 3 graphs, it is safe to conclude that the potential customers who are more likely to respond to the campaigns/offers are the ones between Age 20-80 with no kids or atmost 1 kid and usually have a graduation degree.

In [ ]: