In [1]:

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

In [2]:

```
df = pd.read_csv('Black_Friday_sale.csv')
df
```

Out[2]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	
...	...	...	...	...	...	...	
550063	1006033	P00372445	M	51-55	13	B	
550064	1006035	P00375436	F	26-35	1	C	
550065	1006036	P00375436	F	26-35	15	B	
550066	1006038	P00375436	F	55+	1	C	
550067	1006039	P00371644	F	46-50	0	B	

550068 rows × 12 columns

In [3]:

```
df.describe
```

Out[3]:

```
<bound method NDFrame.describe of         User_ID Product_ID Gender    Age
Occupation City_Category  \
0       1000001  P00069042      F   0-17          10              A
1       1000001  P00248942      F   0-17          10              A
2       1000001  P00087842      F   0-17          10              A
3       1000001  P00085442      F   0-17          10              A
4       1000002  P00285442      M    55+          16              C
...         ...        ...    ...    ...         ...            ...
550063  1006033  P00372445      M  51-55          13              B
550064  1006035  P00375436      F  26-35           1              C
550065  1006036  P00375436      F  26-35          15              B
550066  1006038  P00375436      F    55+           1              C
550067  1006039  P00371644      F  46-50           0              B

       Stay_In_Current_City_Years  Marital_Status  Product_Category_1  \
0                               2               0                   3
1                               2               0                   1
2                               2               0                  12
3                               2               0                  12
4                              4+               0                   8
...                           ...             ...                 ...
550063                          1               1                  20
550064                          3               0                  20
550065                         4+               1                  20
550066                          2               0                  20
550067                         4+               1                  20

        Product_Category_2  Product_Category_3  Purchase
0                      NaN                 NaN      8370
1                      6.0                14.0     15200
2                      NaN                 NaN      1422
3                     14.0                 NaN      1057
4                      NaN                 NaN      7969
...                    ...                 ...       ...
550063                 NaN                 NaN       368
550064                 NaN                 NaN       371
550065                 NaN                 NaN       137
550066                 NaN                 NaN       365
550067                 NaN                 NaN       490

[550068 rows x 12 columns]>
```

In [4]:

```
df.head()
```

Out[4]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
0	1000001	P00069042	F	0-17	10	A	2
1	1000001	P00248942	F	0-17	10	A	2
2	1000001	P00087842	F	0-17	10	A	2
3	1000001	P00085442	F	0-17	10	A	2
4	1000002	P00285442	M	55+	16	C	4+

In [5]:

```
df.columns
```

Out[5]:

```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Categ
ory',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category_
1',
       'Product_Category_2', 'Product_Category_3', 'Purchase'],
      dtype='object')
```

In [6]:

```
df.nunique()
```

Out[6]:

```
User_ID                        5891
Product_ID                     3631
Gender                            2
Age                               7
Occupation                       21
City_Category                     3
Stay_In_Current_City_Years        5
Marital_Status                    2
Product_Category_1               20
Product_Category_2               17
Product_Category_3               15
Purchase                      18105
dtype: int64
```

In [7]:

```
df.isna().sum()
```

Out[7]:

```
User_ID                            0
Product_ID                         0
Gender                             0
Age                                0
Occupation                         0
City_Category                      0
Stay_In_Current_City_Years         0
Marital_Status                     0
Product_Category_1                 0
Product_Category_2            173638
Product_Category_3            383247
Purchase                           0
dtype: int64
```

In [8]:

```
df.drop('Product_Category_2',axis=1,inplace=True)
df.drop('Product_Category_3',axis=1,inplace=True)
```

In [9]:

```
from sklearn.preprocessing import LabelEncoder

labenc=LabelEncoder()
```

In [10]:

```
df['Gender']=labenc.fit_transform(df['Gender'])
df['Gender']
```

Out[10]:

```
0         0
1         0
2         0
3         0
4         1
         ..
550063    1
550064    0
550065    0
550066    0
550067    0
Name: Gender, Length: 550068, dtype: int32
```

In [11]:

```
df['Age']=labenc.fit_transform(df['Age'])
df['Age']
```

Out[11]:

```
0         0
1         0
2         0
3         0
4         6
         ..
550063    5
550064    2
550065    2
550066    6
550067    4
Name: Age, Length: 550068, dtype: int32
```

In [12]:

```
selected_col=['Age','Purchase']
```

In [13]:

```
from sklearn.preprocessing import StandardScaler

scaler=StandardScaler()
scaleddf=scaler.fit_transform(df[selected_col])
scaleddf
```

Out[13]:

```
array([[-1.84424754, -0.1779729 ],
       [-1.84424754,  1.1817558 ],
       [-1.84424754, -1.56119326],
       ...,
       [-0.36673935, -1.81701338],
       [ 2.58827703, -1.77162273],
       [ 1.11076884, -1.7467375 ]])
```

In [14]:

```
from sklearn.cluster import KMeans
inertia_value=[]
k_values=range(1,11)

for i in k_values:
    kmeans=KMeans(n_clusters=i,random_state=0)
    kmeans.fit(scaleddf)
    inertia_value.append(kmeans.inertia_)

    # Plot the Elbow method to choose the optimal number of clusters
plt.plot(k_values,inertia_value,marker='o')
plt.title('Elbow Method for Optimal Number of Clusters')
plt.xlabel('No: of clusters(k)')
plt.ylabel('Inertia')
plt.show()
```

```
t modify any behavior and is safe. If you specifically wanted the numpy
scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.o
rg/devdocs/release/1.20.0-notes.html#deprecations (https://numpy.org/dev
docs/release/1.20.0-notes.html#deprecations)
  dtype = np.float
C:\Users\hp\Anaconda3\lib\site-packages\sklearn\metrics\pairwise.py:56:
DeprecationWarning: `np.float` is a deprecated alias for the builtin `fl
oat`. To silence this warning, use `float` by itself. Doing this will no
t modify any behavior and is safe. If you specifically wanted the numpy
scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.o
rg/devdocs/release/1.20.0-notes.html#deprecations (https://numpy.org/dev
docs/release/1.20.0-notes.html#deprecations)
  dtype = np.float
C:\Users\hp\Anaconda3\lib\site-packages\sklearn\metrics\pairwise.py:56:
DeprecationWarning: `np.float` is a deprecated alias for the builtin `fl
oat`. To silence this warning, use `float` by itself. Doing this will no
t modify any behavior and is safe. If you specifically wanted the numpy
scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.o
```

In [16]:

```
k=5
kmeans=KMeans(n_clusters=5,random_state=0)
df['cluster']=kmeans.fit_predict(df[selected_col])
df.head()
```

```
rg/devdocs/release/1.20.0-notes.html#deprecations (https://numpy.org/dev
docs/release/1.20.0-notes.html#deprecations)
  dtype = np.float
C:\Users\hp\Anaconda3\lib\site-packages\sklearn\metrics\pairwise.py:56:
DeprecationWarning: `np.float` is a deprecated alias for the builtin `fl
oat`. To silence this warning, use `float` by itself. Doing this will no
t modify any behavior and is safe. If you specifically wanted the numpy
scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.o
rg/devdocs/release/1.20.0-notes.html#deprecations (https://numpy.org/dev
docs/release/1.20.0-notes.html#deprecations)
  dtype = np.float
C:\Users\hp\Anaconda3\lib\site-packages\sklearn\metrics\pairwise.py:56:
DeprecationWarning: `np.float` is a deprecated alias for the builtin `fl
oat`. To silence this warning, use `float` by itself. Doing this will no
t modify any behavior and is safe. If you specifically wanted the numpy
scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.o
rg/devdocs/release/1.20.0-notes.html#deprecations (https://numpy.org/dev
```

In [17]:

```
k=5
cluster_characteristics = []
for cluster_id in range(k):
    cluster_data = df[df['cluster']==cluster_id]

    dominant_age_group=cluster_data['Age'].mode()[0]
    product_of_interest=cluster_data['Product_ID'].mode()[0]

    cluster_characteristics.append({
        'Cluster': cluster_id,
        'Dominant_Age_Group': dominant_age_group,
        'Product_Of_Interest': product_of_interest
    })

cluster_characteristics_df=pd.DataFrame(cluster_characteristics)
cluster_characteristics_df
```

Out[17]:

	Cluster	Dominant_Age_Group	Product_Of_Interest
0	0	2	P00145042
1	1	2	P00265242
2	2	2	P00051442
3	3	2	P00025442
4	4	2	P00102642

In [18]:

```
df.groupby('Age')['Product_ID'].agg(pd.Series.mode).to_frame()
```

Out[18]:

	Product_ID
Age	
0	P00255842
1	P00265242
2	P00265242
3	P00025442
4	P00265242
5	P00265242
6	P00265242

In [ ]:

```
df.head()
```

In [21]:

```
max_sales_category=df.groupby('Product_Category_1')['Purchase'].sum().idxmax()
max_sales_value=df.groupby('Product_Category_1')['Purchase'].sum().max()

print(" Category of Product with Highest Sales: Product Category", max_sales_category)
print("Highest Sales Value:", max_sales_value)
```

```
 Category of Product with Highest Sales: Product Category 1
Highest Sales Value: 1910013754
```

In [20]:

```
max_sales_product=df.groupby('Product_ID')['Purchase'].sum().idxmax()
max_sales_value=df.groupby('Product_ID')['Purchase'].sum().max()

print(" ID of Highest Sales Product:", max_sales_product)
print(" Sales Value:", max_sales_value)
```

```
Product ID with Highest Sales: P00025442
Highest Sales Value: 27995166
```

In [ ]:

```

```