```
#OIBSIP Data Analytics
#level 1 task no:-2- Customer Segmentation
#Intern name:- Sanjana Gidwani
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

```
df=pd.read_csv('/content/ifood_df.csv')
df.head()
```

	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds	.
0	58138.0	0	0	58	635	88	546	172	88	88	
1	46344.0	1	1	38	11	1	6	2	1	6	
2	71613.0	0	0	26	426	49	127	111	21	42	
3	26646.0	1	0	26	11	4	20	10	3	5	
4	58293.0	1	0	94	173	43	118	46	27	15	

5 rows × 39 columns

```
df.info()
df.isnull().sum()
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2205 entries, 0 to 2204
Data columns (total 39 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Income               2205 non-null   float64
 1   Kidhome              2205 non-null   int64
 2   Teenhome             2205 non-null   int64
 3   Recency              2205 non-null   int64
 4   MntWines             2205 non-null   int64
 5   MntFruits            2205 non-null   int64
 6   MntMeatProducts      2205 non-null   int64
 7   MntFishProducts      2205 non-null   int64
 8   MntSweetProducts     2205 non-null   int64
 9   MntGoldProds         2205 non-null   int64
 10  NumDealsPurchases    2205 non-null   int64
```

```
columns_to_drop= ['Z_CostContact', 'Z_Revenue']
df=df.drop([col for col in columns_to_drop if col in df.columns], axis=1,inplace=False)
df.ffill(inplace=True)
df['TotalPurchases']=df['NumDealsPurchases']+df['NumWebPurchases']+df['NumCatalogPurchases']+df['NumStorePurchases']
df['TotalSpent']=df['MntWines']+df['MntFruits']+df['MntMeatProducts']+df['MntFishProducts']+df['MntSweetProducts']+df['MntGol
```

```
average_purchase_value=df['TotalSpent'].mean()
purchase_frequency=df['TotalPurchases'].mean()
print(average_purchase_value)
print(purchase_frequency)
df[['Income','Age','TotalSpent','TotalPurchases']].describe()
```

```
 26  marital_Divorced     2205 non-null   int64
 27  marital_Married      2205 non-null   int64
 28  marital_Single       2205 non-null   int64
 29  marital_Together     2205 non-null   int64
 32  education_Basic      2205 non-null   int64
 33  education_Graduation 2205 non-null   int64
 36  MntTotal             2205 non-null   int64
 37  MntRegularProds      2205 non-null   int64
dtypes: float64(1), int64(38)
memory usage: 672.0 KB
```

```
606.821768707483
14.88798185941043
```

	Income	Age	TotalSpent	TotalPurchases
count	2205.000000	2205.000000	2205.000000	2205.000000
mean	51622.094785	51.095692	606.821769	14.887982
std	20713.063826	11.705801	601.675284	7.615277
min	1730.000000	24.000000	5.000000	0.000000
25%	35196.000000	43.000000	69.000000	8.000000
50%	51287.000000	50.000000	397.000000	15.000000
75%	68281.000000	61.000000	1047.000000	21.000000
max	113734.000000	80.000000	2525.000000	43.000000

					MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetP
count	2205.000000	2205.000000	2205.000000	2205.000000	2205.000000	2205.000000	2205.000000	2205.000000	220
75%	68281.000000	1.000000	1.000000	74.000000	507.000000	33.000000	232.000000	50.000000	3
max	113734.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.000000	259.000000	26

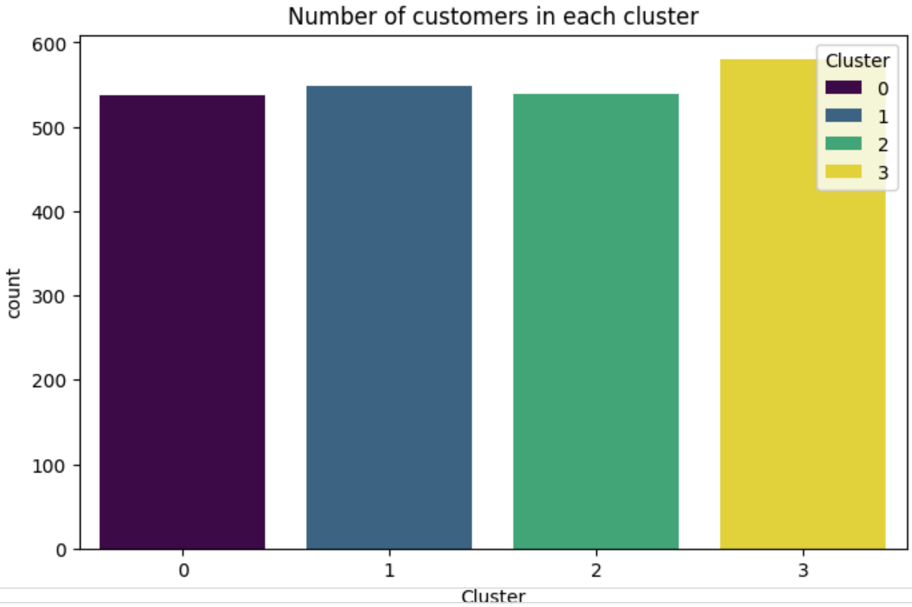
8 rows × 39 columns

```
#k-means clustering
features = df[['Income', 'Age', 'TotalSpent', 'TotalPurchases', 'Recency', 'NumWebVisitsMonth']]
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)
kmeans = KMeans(n_clusters=4, random_state=42)
df['Cluster'] = kmeans.fit_predict(scaled_features)
df['Cluster'].value_counts()
```

Cluster	count
3	580
1	549
2	539
0	537

dtype: int64

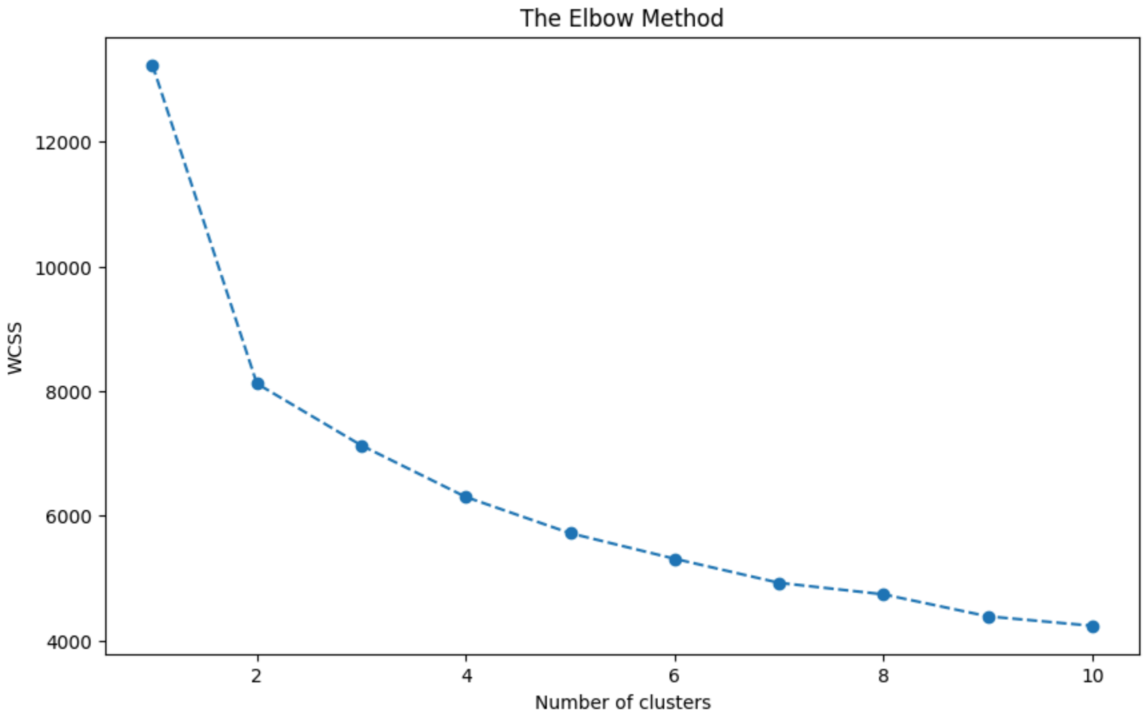
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Income', y='TotalSpent', hue='Cluster', data=df , palette='viridis')
plt.title('Income vs Total Spent by Cluster')
plt.show()

plt.figure(figsize=(8,5))
sns.countplot(x='Cluster', data=df, hue='Cluster', palette='viridis')
plt.title('Number of customers in each cluster')
plt.show()
```




```
cluster_analysis= df.groupby('Cluster').mean()[['Income','Age','TotalSpent','TotalPurchases','Recency','NumWebVisitsMonth']]
print(cluster_analysis)
```

```
               Income        Age   TotalSpent  TotalPurchases     Recency  \
Cluster
0        76679.921788  50.301676  1353.085661       20.206704  50.148976
1        35905.271403  49.510018   130.650273        8.892532  75.344262
2        60685.836735  57.515770   864.784787       22.348794  48.224490
3        34875.760345  47.365517   126.877586        8.705172  23.755172

         NumWebVisitsMonth
Cluster
0                 2.415270
1                 6.466302
2                 5.829314
3                 6.515517
```

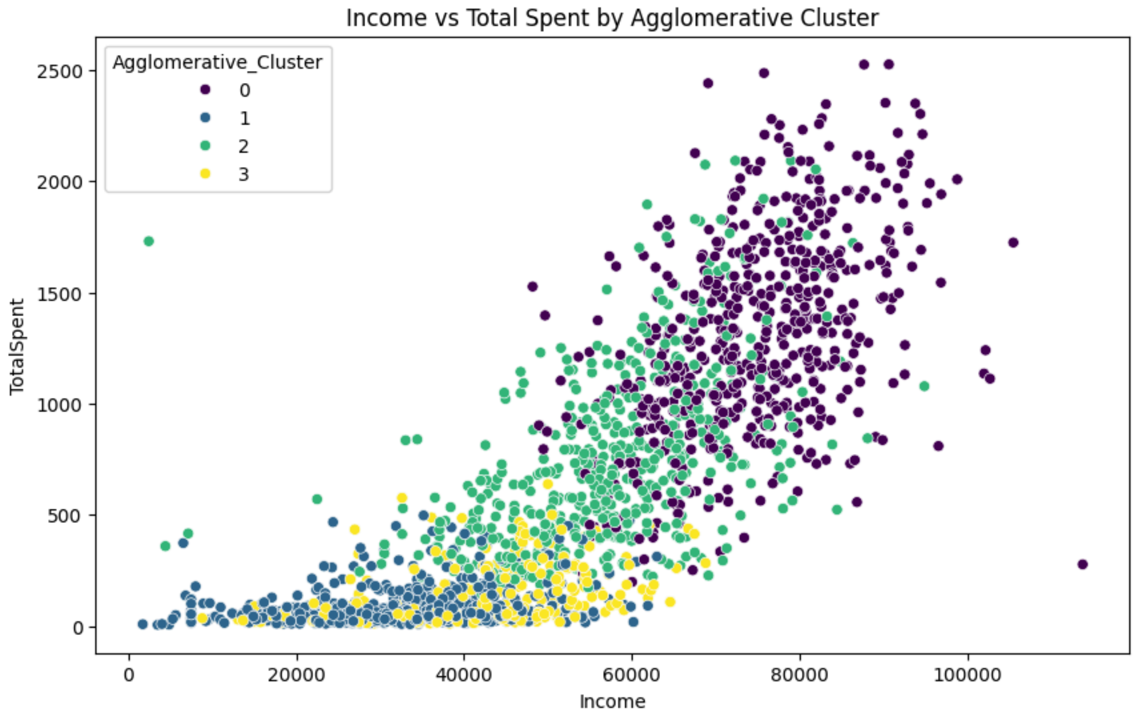
```
#elbow method
WCSS=[]
for i in range(1,11):
  Kmeans=KMeans(n_clusters=i,random_state=42)
  Kmeans.fit(scaled_features)
  WCSS.append(Kmeans.inertia_)
plt.figure(figsize=(10,6))
plt.plot(range(1,11),WCSS,marker='o',linestyle='--')
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```




```
#silhouette score
from sklearn.metrics import silhouette_score
for n_clusters in range(2,6):
  kmeans=KMeans(n_clusters=n_clusters,random_state=42)
  cluster_labels=kmeans.fit_predict(scaled_features)
  silhouette_avg=silhouette_score(scaled_features,cluster_labels)
  print(f"For n_clusters={n_clusters}, the silhouette score is {silhouette_avg}")
```

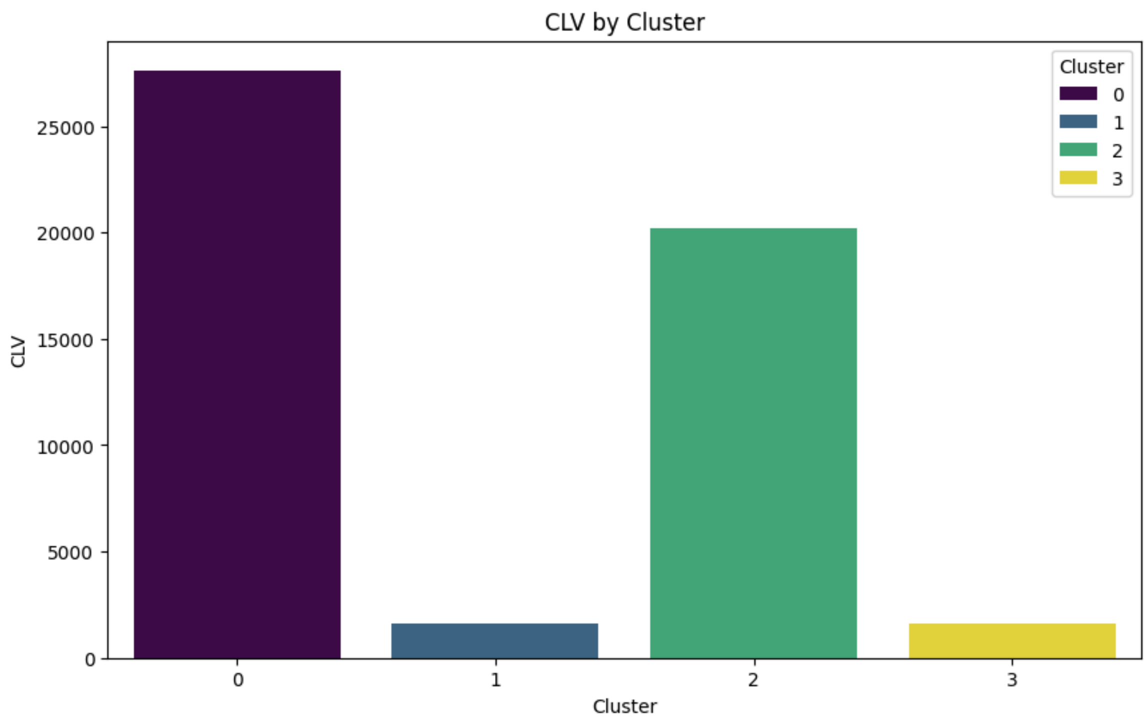
```
For n_clusters=2, the silhouette score is 0.3382877874027988
For n_clusters=3, the silhouette score is 0.23929563147291963
For n_clusters=4, the silhouette score is 0.20115382621298702
For n_clusters=5, the silhouette score is 0.20469588596966065
```

```
#agglomerative clustering
from sklearn.cluster import AgglomerativeClustering
agg_cluster=AgglomerativeClustering(n_clusters=4)
df['Agglomerative_Cluster']=agg_cluster.fit_predict(scaled_features)

plt.figure(figsize=(10,6))
sns.scatterplot(x='Income',y='TotalSpent',hue='Agglomerative_Cluster',data=df,palette='viridis')
plt.title('Income vs Total Spent by Agglomerative Cluster')
plt.show()
```




```
#clv analysis
df['CLv']=df['TotalSpent']* df['TotalPurchases']
clv_by_cluster=df.groupby('Cluster')['CLv'].mean()
print(clv_by_cluster)

plt.figure(figsize=(10,6))
sns.barplot(x=clv_by_cluster.index,y=clv_by_cluster.values,hue=clv_by_cluster.index, palette='viridis')
plt.title('CLV by Cluster')
plt.xlabel('Cluster')
plt.ylabel('CLV')
plt.show()
```

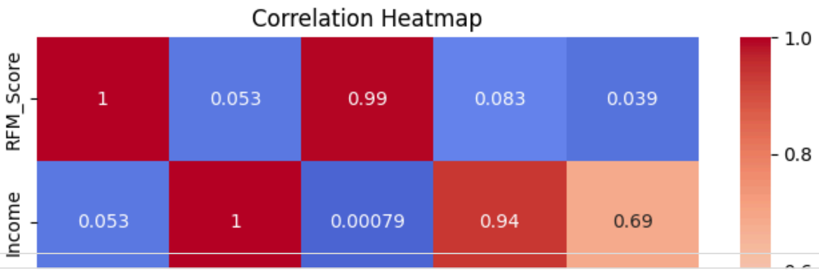
```
Cluster
0    27631.482309
1     1626.786885
2    20227.894249
3     1587.567241
Name: CLv, dtype: float64
```



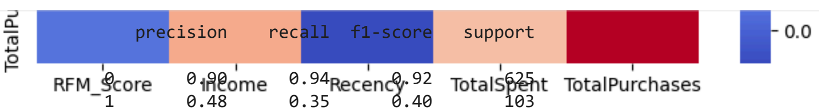

```
#rfm analysis
df['RecencyScore']=pd.qcut(df['Recency'],4,labels=False)
df['FrequencyScore']=pd.qcut(df['TotalPurchases'],4,labels=False)
```

```
df['MonetaryScore']=pd.qcut(df['TotalSpent'],4,labels=False)

df['RFM_Score']=df['RecencyScore'].astype(str)+df['FrequencyScore'].astype(str)+df['MonetaryScore'].astype(str)
rfm_analysis=df.groupby('RFM_Score').agg({'Income':'mean','Recency':'mean','TotalSpent':'mean','TotalPurchases':'mean'}).rese
print(rfm_analysis)
plt.figure(figsize=(8,6))
sns.heatmap(rfm_analysis.corr(),annot=True,cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

```
   RFM_Score        Income    Recency   TotalSpent  TotalPurchases
0        000  31347.188976  12.307087    36.842520        5.346457
1        001  35222.722222   8.722222   104.722222        6.916667
[illegible]
3        011  41061.304348  12.760870   191.456522       11.336957
4        012  59962.727273  14.090909   654.818182       13.909091
5        013  78804.800000   7.800000  1592.500000       14.000000
6        021  45260.888889   8.777778   307.388889       16.888889
7        022  58846.080645  10.790323   587.758065       18.580645
8        023  75780.225352  10.267606  1475.436620       18.718310
9        031  59081.000000   4.500000   318.000000       26.000000
10       032  64004.882353  12.941176   817.220588       25.014706
11       033  74221.183673  12.102041  1431.551020       26.081633
12       100  30645.777778  37.037037    38.274074        5.651852
13       101  38908.785714  35.321429    98.357143        7.178571
14       110  23131.250000  41.300000    52.750000        9.850000
15       111  42032.000000  37.256098   204.097561       11.402439
16       112  61729.200000  37.300000   697.500000       13.700000
17       113  75308.714286  37.571429  1509.142857       14.428571
18       121  49371.666667  35.533333   336.666667       16.866667
19       122  58787.672131  35.442623   592.491803       18.475410
20       123  75694.433333  35.016667  1494.316667       18.816667
21       132  61157.203390  37.711864   813.745763       24.661017
22       133  72376.140845  37.295775  1523.746479       26.014085
23       200  30976.514563  61.349515    36.922330        5.310680
24       201  37209.441860  60.627907   104.000000        7.209302
25       210  22915.785714  58.928571    50.500000       10.500000
26       211  41968.577465  62.957746   203.507042       11.507042
27       212  67656.736842  63.000000   792.210526       14.052632
28       213  75161.625000  66.250000  1697.750000       12.750000
29       221  45344.083333  59.833333   310.666667       17.000000
30       222  58080.658228  62.772152   650.569620       18.582278
31       223  75659.657534  62.547945  1488.643836       18.726027
32       232  60512.265625  60.828125   793.781250       24.171875
33       233  73578.750000  61.859375  1435.468750       26.093750
34       300  32322.920000  86.176000    39.760000        5.592000
35       301  37934.846154  86.205128   101.692308        7.205128
36       310  22317.700000  84.550000    53.850000        9.900000
37       311  41765.418605  87.116279   196.790698       11.581395
38       312  62762.533333  86.866667   815.333333       14.000000
39       313  83400.600000  90.200000  1449.200000       13.800000
40       321  40792.333333  89.916667   300.750000       16.666667
41       322  59292.115385  88.365385   627.846154       18.692308
42       323  76487.017857  86.178571  1487.910714       18.535714
43       332  60326.900000  86.960000   793.000000       24.320000
44       333  73437.285714  87.545455  1517.883117       25.428571
```




```
#predictive modeling
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report,accuracy_score
X=df[['Income',  'TotalSpent', 'TotalPurchases', 'Recency']]
y=df['Response']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
clf=RandomForestClassifier(n_estimators=100,random_state=42)
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred))
```




```
              precision    recall  f1-score   support

           0       0.90      0.94      0.92       625
           1       0.48      0.35      0.40       103

    accuracy                           0.85       728
```

```
   macro avg       0.69      0.64      0.66       728
weighted avg       0.84      0.85      0.84       728

0.8543956043956044
```

```
#pairplot
sns.pairplot(df,hue='Cluster', vars=['Income','Age','TotalSpent','TotalPurchases','Recency','NumWebVisitsMonth'])
plt.show()
plt.figure(figsize=(12,10))
correlation_matrix=df.corr()
mask=np.triu(np.ones_like(correlation_matrix,dtype=bool))
sns.heatmap(correlation_matrix,annot=True,mask=mask,fmt='.2f', cmap='coolwarm', linewidths=.5, square=True, cbar_kws={"shrink
plt.title('Correlation Matrix')
plt.xticks(rotation=45, ha='right',fontsize=12)
plt.yticks(fontsize=12)
plt.show()

from mpl_toolkits.mplot3d import Axes3D
fig=plt.figure(figsize=(10,7))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df['Income'], df['Age'], df['TotalSpent'], c=df['Cluster'], cmap='viridis')
ax.set_xlabel('Income')
ax.set_ylabel('Age')
ax.set_zlabel('TotalSpent')
plt.title('3D Scatter Plot of Income, Age, and TotalSpent')
plt.show()
```