```
from google.colab import files
uploaded = files.upload()
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving ifood_df.csv to ifood_df.csv
```

```
import pandas as pd

print ("Data Cleaning")
file_path = "ifood_df.csv"

df = pd.read_csv(file_path)

df.head()
```

```
Data Cleaning
```

	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	Mnt
0	58138.0	0	0	58	635	88	546	
1	46344.0	1	1	38	11	1	6	
2	71613.0	0	0	26	426	49	127	
3	26646.0	1	0	26	11	4	20	
4	58293.0	1	0	94	173	43	118	

5 rows × 39 columns

```
print("\nDescriptive Statistics for Numerical Columns:")
print(df.describe())

categorical_columns = ['Kidhome', 'Teenhome', 'Complain', 'Response']
for col in categorical_columns:
    print(f"\nUnique values in {col}:")
    print(df[col].value_counts())
```

```
Descriptive Statistics for Numerical Columns:
                Income      Kidhome     Teenhome      Recency     MntWines  \
count    2205.000000  2205.000000  2205.000000  2205.000000  2205.000000
mean    51622.094785     0.442177     0.506576    49.009070   306.164626
std     20713.063826     0.537132     0.544380    28.932111   337.493839
min      1730.000000     0.000000     0.000000     0.000000     0.000000
25%     35196.000000     0.000000     0.000000    24.000000    24.000000
50%     51287.000000     0.000000     0.000000    49.000000   178.000000
75%     68281.000000     1.000000     1.000000    74.000000   507.000000
max    113734.000000     2.000000     2.000000    99.000000  1493.000000

          MntFruits  MntMeatProducts  MntFishProducts  MntSweetProducts  \
```

```
count  2205.000000      2205.000000      2205.000000       2205.000000
mean     26.403175       165.312018        37.756463         27.128345
std      39.784484       217.784507        54.824635         41.130468
min       0.000000         0.000000         0.000000          0.000000
25%       2.000000        16.000000         3.000000          1.000000
50%       8.000000        68.000000        12.000000          8.000000
75%      33.000000       232.000000        50.000000         34.000000
max     199.000000      1725.000000       259.000000        262.000000

       MntGoldProds  ...  marital_Together  marital_Widow  education_2n Cycle
count   2205.000000  ...       2205.000000    2205.000000         2205.000000
mean      44.057143  ...          0.257596       0.034467            0.089796
std       51.736211  ...          0.437410       0.182467            0.285954
min        0.000000  ...          0.000000       0.000000            0.000000
25%        9.000000  ...          0.000000       0.000000            0.000000
50%       25.000000  ...          0.000000       0.000000            0.000000
75%       56.000000  ...          1.000000       0.000000            0.000000
max      321.000000  ...          1.000000       1.000000            1.000000

       education_Basic  education_Graduation  education_Master  education_PhD
count      2205.000000           2205.000000       2205.000000    2205.000000
mean          0.024490              0.504762          0.165079       0.215873
std           0.154599              0.500091          0.371336       0.411520
min           0.000000              0.000000          0.000000       0.000000
25%           0.000000              0.000000          0.000000       0.000000
50%           0.000000              1.000000          0.000000       0.000000
75%           0.000000              1.000000          0.000000       0.000000
max           1.000000              1.000000          1.000000       1.000000

          MntTotal  MntRegularProds  AcceptedCmpOverall
count  2205.000000      2205.000000          2205.00000
mean    562.764626       518.707483             0.29932
std     575.936911       553.847248             0.68044
min       4.000000      -283.000000             0.00000
25%      56.000000        42.000000             0.00000
50%     343.000000       288.000000             0.00000
75%     964.000000       884.000000             0.00000
max    2491.000000      2458.000000             4.00000

[8 rows x 39 columns]

Unique values in Kidhome:
Kidhome
0    1276
1     883
```

```
print ("1 Feature Selection & Engineering")
selected_features = ['Income', 'Recency', 'MntTotal', 'Age',
                     'NumWebPurchases', 'NumCatalogPurchases',
                     'NumStorePurchases', 'NumWebVisitsMonth']

df_selected = df[selected_features]
print("\nSelected Features:\n", df_selected.head())
```

```
1 Feature Selection & Engineering

Selected Features:
     Income  Recency  MntTotal  Age  NumWebPurchases  NumCatalogPurchases  \
0  58138.0       58      1529   63                8                   10
1  46344.0       38        21   66                1                    1
```

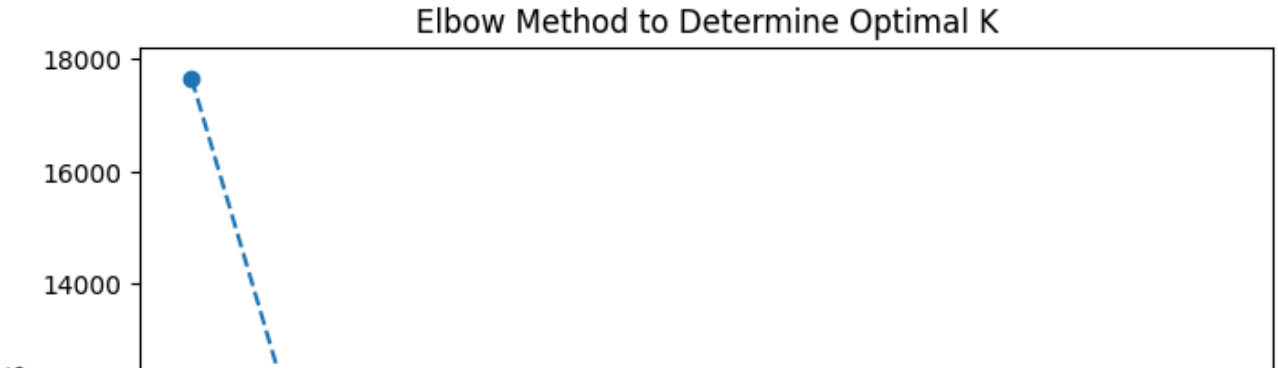
```
2  71613.0       26       734   55                8                     2
3  26646.0       26        48   36                2                     0
4  58293.0       94       407   39                5                     3

   NumStorePurchases  NumWebVisitsMonth
0                  4                  7
1                  2                  5
2                 10                  4
3                  4                  6
4                  6                  5
```

```
print ("2 Feature Selection & Engineering")
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_selected)

df_scaled = pd.DataFrame(df_scaled, columns=selected_features)

print("\nScaled Data Sample:\n", df_scaled.head())
```

```
2 Feature Selection & Engineering

Scaled Data Sample:
      Income   Recency  MntTotal       Age  NumWebPurchases  \
0  0.314651  0.310830  1.678056  1.017189         1.424772
1 -0.254877 -0.380600 -0.940880  1.273530        -1.132957
2  0.965354 -0.795458  0.297384  0.333612         1.424772
3 -1.206087 -0.795458 -0.893989 -1.289883        -0.767567
4  0.322136  1.555404 -0.270516 -1.033542         0.328602

   NumCatalogPurchases  NumStorePurchases  NumWebVisitsMonth
0             2.628526          -0.562650           0.689203
1            -0.588043          -1.179732          -0.139645
2            -0.230646           1.288596          -0.554069
3            -0.945440          -0.562650           0.274779
4             0.126750           0.054432          -0.139645
```

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
print ("Clustering K-means")
wcss = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(df_scaled)
    wcss.append(kmeans.inertia_)  # Inertia = Sum of squared distances of samples to the

plt.figure(figsize=(8, 5))
plt.plot(range(1, 11), wcss, marker='o', linestyle='--')
plt.xlabel("Number of Clusters (K)")
plt.ylabel("WCSS")
plt.title("Elbow Method to Determine Optimal K")
plt.show()
```

Clustering K-means



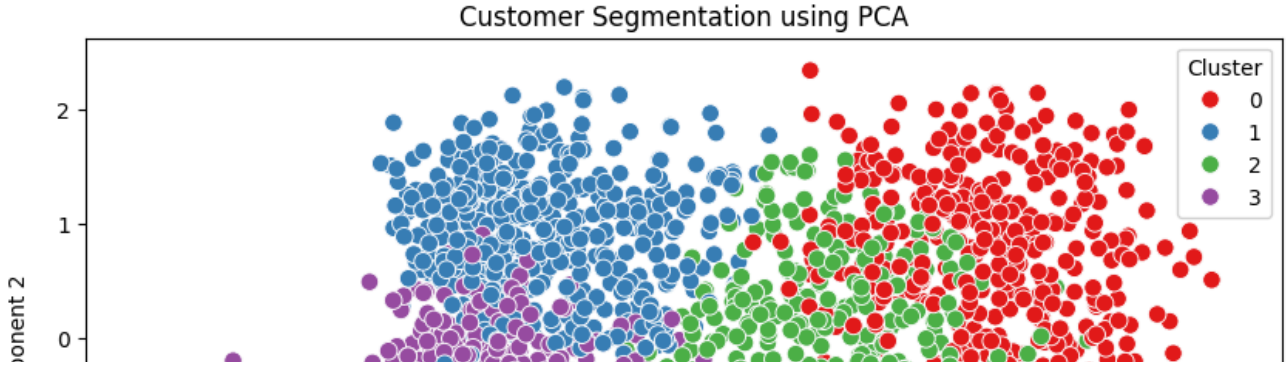

```
print ("K-means Clustering")
optimal_k = 4

kmeans = KMeans(n_clusters=optimal_k, random_state=42, n_init=10)
df_scaled['Cluster'] = kmeans.fit_predict(df_scaled)

df['Cluster'] = df_scaled['Cluster']
print("\nCluster Counts:\n", df['Cluster'].value_counts())
```

K-means Clustering

```
Cluster Counts:
 Cluster
2    581
3    560
1    535
0    529
Name: count, dtype: int64
```

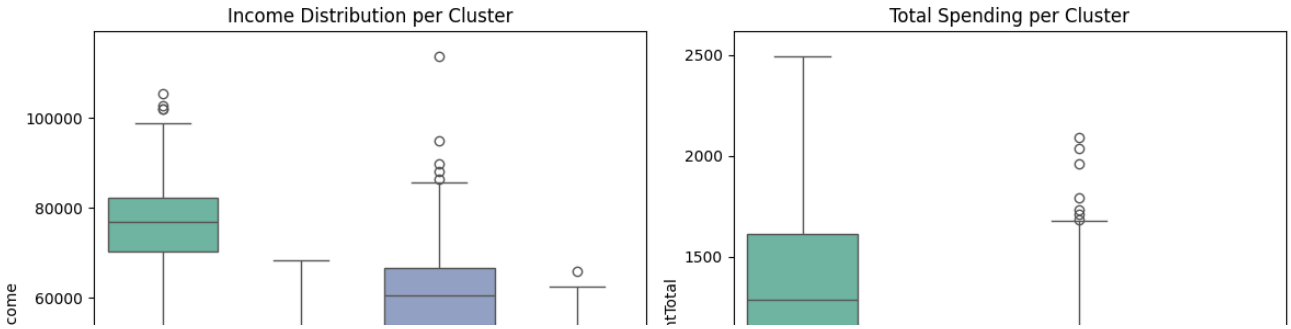
```
from sklearn.decomposition import PCA
import seaborn as sns
import matplotlib.pyplot as plt
print ("Cluster Visualization")

pca = PCA(n_components=2)
df_pca = pca.fit_transform(df_scaled.drop(columns=['Cluster']))

df_pca = pd.DataFrame(df_pca, columns=['PC1', 'PC2'])
df_pca['Cluster'] = df_scaled['Cluster']

plt.figure(figsize=(10, 6))
sns.scatterplot(x=df_pca['PC1'], y=df_pca['PC2'], hue=df_pca['Cluster'], palette='Set1',
plt.title("Customer Segmentation using PCA")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend(title="Cluster")
plt.show()
```

Cluster Visualization




```
print ("Spending Behaviour Analysis Of Cluster Visualization")
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
sns.boxplot(x=df['Cluster'], y=df['Income'], palette='Set2')
plt.title("Income Distribution per Cluster")

plt.subplot(1, 2, 2)
sns.boxplot(x=df['Cluster'], y=df['MntTotal'], palette='Set2')
plt.title("Total Spending per Cluster")

plt.tight_layout()
plt.show()
```

```
Spending Behaviour Analysis Of Cluster Visualization
<ipython-input-34-85549867e827>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed ir

  sns.boxplot(x=df['Cluster'], y=df['Income'], palette='Set2')
<ipython-input-34-85549867e827>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed ir

  sns.boxplot(x=df['Cluster'], y=df['MntTotal'], palette='Set2')
```




```
!pip install gtts
```

```
Collecting gtts
  Downloading gTTS-2.5.4-py3-none-any.whl.metadata (4.1 kB)
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.11/
Requirement already satisfied: click<8.2,>=7.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/pyth
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11
Downloading gTTS-2.5.4-py3-none-any.whl (29 kB)
Installing collected packages: gtts
```

```
    Successfully installed gtts-2.5.4
```

```
from IPython.display import display, Markdown
from gtts import gTTS
import os

recommendations = [
    "**1. Calories:** Some items go up to **1880 kcal**. Pick lower-calorie options like
    "**2. Fat:** Some meals have **1000+ kcal from fat**. Avoid high-fat foods and pick o
    "**3. Saturated & Trans Fat:** Bad for the heart. Choose meals with **less than 5g sa
    "**4. Carbs & Sugar:** Some items have **140g carbs** and **128g sugar**. Avoid high
    "**5. Protein:** More than **20g protein** is good."
]

display(Markdown("### **Recommendations**"))
for rec in recommendations:
    display(Markdown(rec))  # Bold text display
    tts = gTTS(text=rec, lang='en')  # Convert text to speech
    tts.save("rec.mp3")  # Save as MP3
    os.system("mpg321 rec.mp3")  # Play in Colab (Linux)

report = "\n".join(recommendations)

with open("recommendations_report.txt", "w") as file:
    file.write(report)

print("Report saved as 'recommendations_report.txt'")
```

Recommendations

1. Calories: Some items go up to **1880 kcal**. Pick lower-calorie options like **Egg White Delight (250 kcal)** if watching weight.

2. Fat: Some meals have **1000+ kcal from fat**. Avoid high-fat foods and pick ones with **less than 20g fat**.

3. Saturated & Trans Fat: Bad for the heart. Choose meals with **less than 5g saturated fat** and **zero trans fat**.

4. Carbs & Sugar: Some items have **140g carbs** and **128g sugar**. Avoid high sugar foods, aim for **below 30g sugar**.

5. Protein: More than **20g protein** is good.

```
Report saved as 'recommendations_report.txt'
```