Market Segmentation Analysis

Step 1: Deciding (not) to Segment

Market segmentation is a strategic approach where an organization divides its broad consumer or business market into sub-groups of consumers based on shared characteristics. While widely adopted, evaluating whether to pursue this strategy is crucial as it involves significant long-term commitments and investments. Based on the "Market Segmentation Analysis" PDF, here is the detailed evaluation for deciding whether to proceed with segmentation for McDonald's:

1. Data Collection and Preparation
   Gather Relevant Data: Collect data on customer demographics, purchase behavior, preferences, and any other relevant variables.
   Data Cleaning: Ensure the data is clean by handling missing values, removing duplicates, and correcting inconsistencies.

2. Exploratory Data Analysis (EDA)
   Descriptive Statistics: Compute measures like mean, median, mode, and standard deviation for different variables.
   Visualization: Use visualizations (e.g., histograms, bar charts, box plots) to understand the distribution of data and identify patterns or anomalies.
   Correlation Analysis: Check for correlations between variables to understand relationships within the data.

3. Segmentation Analysis
   Clustering Techniques: Apply clustering algorithms (e.g., K-means, hierarchical clustering) to see if distinct groups emerge within the data.
   Evaluation of Clusters: Assess the quality of the clusters using metrics like silhouette score, within-cluster sum of squares (WCSS), and between-cluster variation.
   Segmentation Criteria: Determine if the clusters have practical and meaningful differences in terms of demographics, behavior, or preferences.

4. Decision Making
   Assess Homogeneity: If the customer base shows significant homogeneity with little variation, segmentation may not be necessary.
   Assess Business Goals: Align the segmentation findings with business objectives. If distinct segments align with targeted marketing strategies or product offerings, segmentation can be beneficial.
   Cost-Benefit Analysis: Consider the costs and benefits of implementing segmentation. If the benefits outweigh the costs, segmentation is justified.

5. Summary
   Data Collection and Preparation: Gather and clean the data.

Exploratory Data Analysis (EDA): Understand data distribution and relationships.
Segmentation Analysis: Apply and evaluate clustering methods to identify potential segments.
Decision Making: Assess the need for segmentation based on homogeneity, business goals, and cost-benefit analysis.

Step 2: Specifying the Ideal Target Segment:

Specifying the ideal target segment for McDonald's involves identifying the group of customers who are most likely to be profitable, have the highest growth potential, or align best with the company's strategic goals. Here are the steps to specify the ideal target segment:

1. Identify Segmentation Variables
Demographic: Age, gender, income, education, occupation, family size.
Geographic: Region, city size, urban/rural.
Psychographic: Lifestyle, personality, values, interests.
Behavioral: Purchase frequency, brand loyalty, usage rate, benefits sought.
2. Segment the Market
Use clustering algorithms (e.g., K-means, hierarchical clustering) on the chosen variables to segment the market.
Ensure each segment is distinct, measurable, accessible, substantial, and actionable.
3. Analyze and Profile Each Segment
Demographic Profile: Describe the age, gender, income, etc., of each segment.
Geographic Profile: Describe where the segment is located.
Psychographic Profile: Describe lifestyle, personality, values, etc.
Behavioral Profile: Describe purchasing behavior, brand loyalty, usage rate, etc.
4. Evaluate Segment Attractiveness
Market Size and Growth: Assess the size and growth potential of each segment.
Profitability: Estimate the potential revenue and profitability.
Competitive Landscape: Analyze the level of competition within each segment.
Strategic Fit: Ensure alignment with McDonald's brand values, mission, and strategic goals.
5. Select the Ideal Target Segment
Based on the evaluation, select the segment(s) that are most attractive in terms of profitability, growth potential, and strategic alignment.
Example of an Ideal Target Segment for McDonald's
Let's create a hypothetical example based on the above steps:

Segmentation Variables:

Demographic: Young adults (ages 18-34), middle-income, urban dwellers.
Geographic: Major metropolitan areas.
Psychographic: Health-conscious, tech-savvy, socially active.
Behavioral: High frequency of eating out, preference for quick service, interest in

innovative menu items.
Segment the Market:

Apply K-means clustering and identify a segment that fits the above criteria.
Profile the Segment:

Demographic Profile: Young adults, college students or early-career professionals, with an average income range of $30,000-$60,000.
Geographic Profile: Reside in large cities like New York, Los Angeles, Chicago.
Psychographic Profile: Value convenience and speed, have an interest in healthy eating options and sustainability.
Behavioral Profile: Frequent fast-food diners (3-5 times a week), show high engagement with digital ordering apps, respond well to loyalty programs and promotional offers.
Evaluate Segment Attractiveness:

Market Size and Growth: Large and growing population in urban areas.
Profitability: High disposable income, willingness to spend on convenient and healthy options.
Competitive Landscape: Competitive but manageable with unique value propositions.
Strategic Fit: Aligns with McDonald's goals of increasing digital engagement and offering healthier menu options.
Select the Ideal Target Segment:

Young urban adults (ages 18-34) who are health-conscious and tech-savvy, as they provide a significant growth opportunity, align with strategic goals and can be effectively targeted through digital marketing and innovative menu offerings.

Step 3: Collecting Data:

To collect and prepare data for segmenting the McDonald's market, you would typically follow these steps using Python. Below is a basic outline of how you might collect, clean, and prepare the data:

Step 1: Collecting Data
Assuming you have access to a CSV file containing customer data, here's how you might load and explore this data using Python.

In [55]:
```
# Step 1: Collecting Data
# Loading data from a CSV file
file_path = 'mcdonalds.csv'  # replace with your file path
data = pd.read_csv(file_path)
# Display the first few rows of the dataset
print(data.info())
print(data.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1453 entries, 0 to 1452
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   yummy           1453 non-null   object
 1   convenient      1453 non-null   object
 2   spicy           1453 non-null   object
 3   fattening       1453 non-null   object
 4   greasy          1453 non-null   object
 5   fast            1453 non-null   object
 6   cheap           1453 non-null   object
 7   tasty           1453 non-null   object
 8   expensive       1453 non-null   object
 9   healthy         1453 non-null   object
 10  disgusting      1453 non-null   object
 11  Like            1453 non-null   object
 12  Age             1453 non-null   int64
 13  VisitFrequency  1453 non-null   object
 14  Gender          1453 non-null   object
dtypes: int64(1), object(14)
memory usage: 170.4+ KB
None
               Age
count  1453.000000
mean     44.604955
std      14.221178
min      18.000000
25%      33.000000
50%      45.000000
75%      57.000000
max      71.000000
```

Assigned Task

Step 4: Exploring Data

exploring data generally means performing exploratory data analysis (EDA) to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods. EDA helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions. It can also help in understanding data structures, identifying patterns and relationships, detecting anomalies and outliers, testing assumptions, and handling missing values.

In [14]:
```
# Step 3: Exploratory Data Analysis (EDA)

print(data.describe())
```

```
               Age
count  1431.000000
mean     44.656184
std      14.199400
min      18.000000
25%      33.000000
50%      45.000000
75%      57.000000
max      71.000000
```

In [71]:
```
print(data.shape)  # number of rows and columns
```

```
(1453, 15)
```

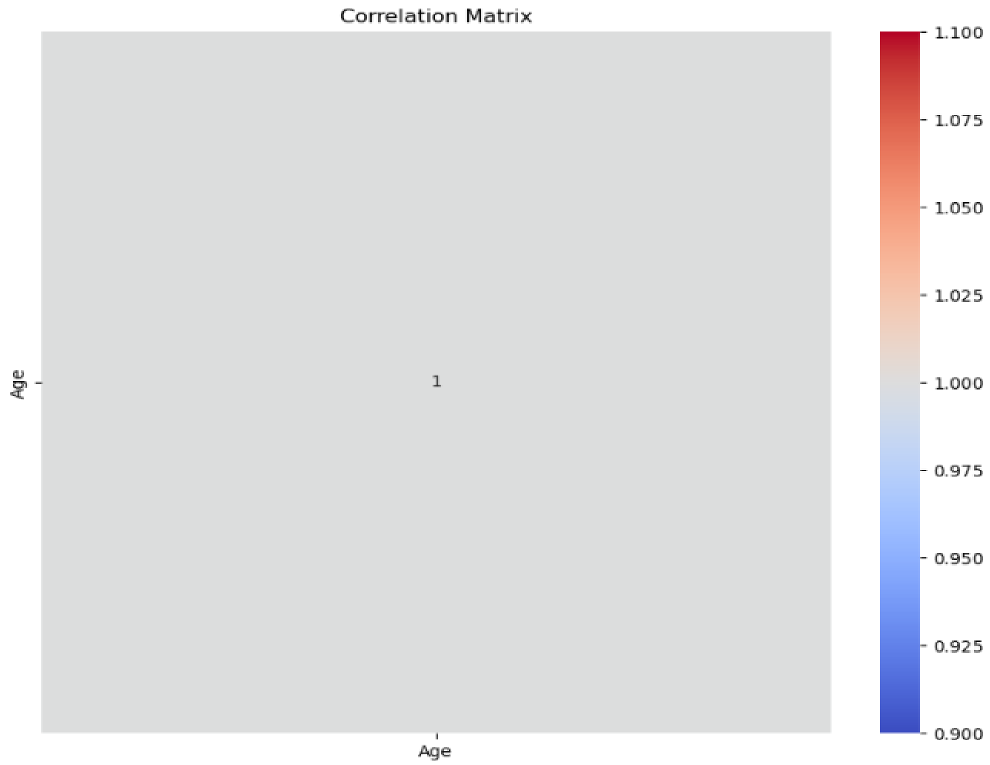
In [69]:
```
print(data.info())  # summary of the dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1453 entries, 0 to 1452
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   yummy           1453 non-null   object
 1   convenient      1453 non-null   object
 2   spicy           1453 non-null   object
 3   fattening       1453 non-null   object
 4   greasy          1453 non-null   object
 5   fast            1453 non-null   object
 6   cheap           1453 non-null   object
 7   tasty           1453 non-null   object
 8   expensive       1453 non-null   object
 9   healthy         1453 non-null   object
 10  disgusting      1453 non-null   object
 11  Like            1453 non-null   object
 12  Age             1453 non-null   int64 
 13  VisitFrequency  1453 non-null   object
 14  Gender          1453 non-null   object
dtypes: int64(1), object(14)
memory usage: 170.4+ KB
None
```

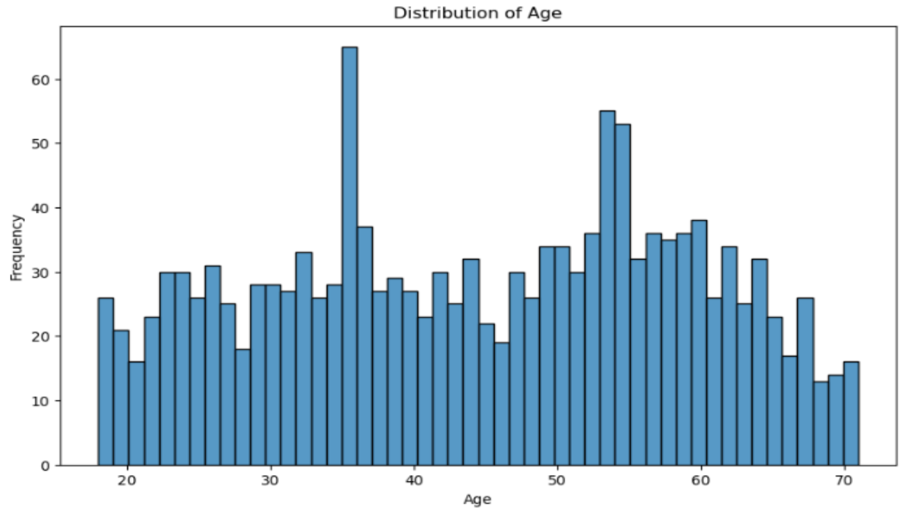
In [70]:
```
print(data.dtypes)  # data types of each column
```

```
yummy             object
convenient        object
spicy             object
fattening         object
greasy            object
fast              object
cheap             object
tasty             object
expensive         object
healthy           object
disgusting        object
Like              object
Age                int64
VisitFrequency    object
Gender            object
dtype: object
```

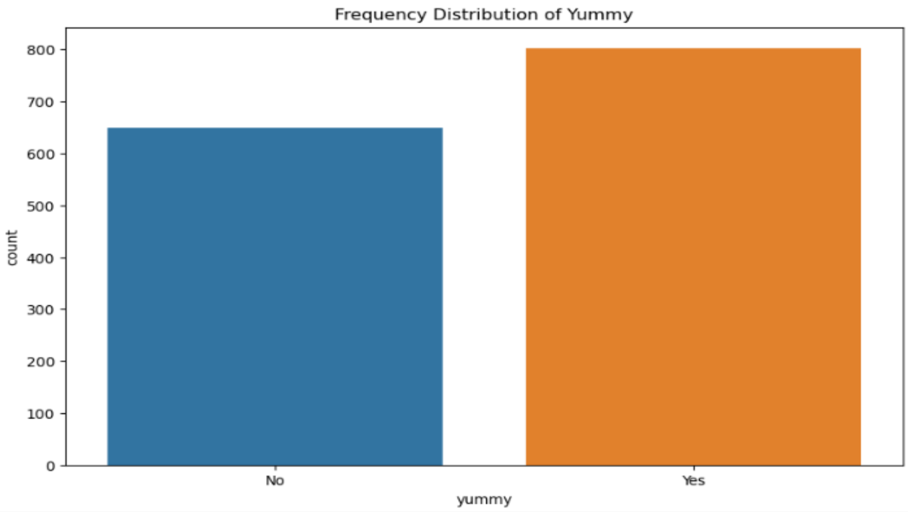
In [78]:
```
# Heatmap of correlation matrix
corr_matrix =data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', square=True)
plt.title('Correlation Matrix')
plt.show()
```



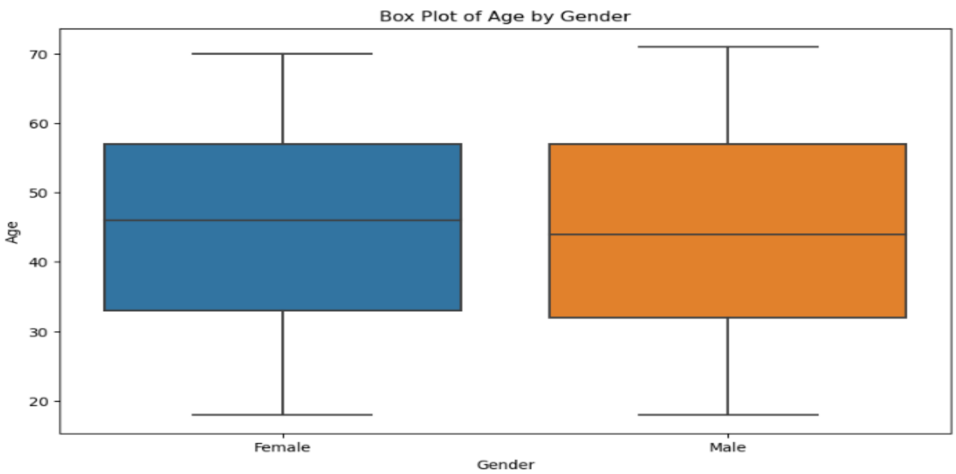
In [75]:
```python
# Histogram of Age
plt.figure(figsize=(10, 6))
sns.histplot(data['Age'], bins=50)
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



In [77]:
```python
# Bar chart of frequency distribution of categorical variables
plt.figure(figsize=(10, 6))
sns.countplot(x='yummy', data=data)
plt.title('Frequency Distribution of Yummy')
plt.show()
```



In [80]:
```python
# Box plot of Age by Gender
plt.figure(figsize=(10, 6))
sns.boxplot(x='Gender', y='Age', data=data)
plt.title('Box Plot of Age by Gender')
plt.xlabel('Gender')
plt.ylabel('Age')
plt.show()
```



Data cleaning, also known as data preprocessing, is an essential step in the data science workflow. Here we perform data cleaning:

```
In [6]: #step 02 cleaning
        # Check for missing values
        print(data.isnull().sum())
```

```
yummy             0
convenient        0
spicy             0
fattening         0
greasy            0
fast              0
cheap             0
tasty             0
expensive         0
healthy           0
disgusting        0
Like              0
Age               0
VisitFrequency    0
Gender            0
dtype: int64
```

```
In [7]: # Fill or drop missing values as appropriate
        data = data.fillna(method='ffill')
```

```
In [8]: # Check for duplicates
        print(data.duplicated().sum())
```

```
22
```

```
In [9]: # Remove duplicates
        data = data.drop_duplicates()
```


```
In [8]: # Check for duplicates
        print(data.duplicated().sum())
```

```
22
```

```
In [9]: # Remove duplicates
        data = data.drop_duplicates()
```


```
In [10]: # Display the cleaned data
         print(data.head())
```

```
  yummy convenient spicy fattening greasy fast cheap tasty expensive healthy  \
0    No        Yes    No       Yes     No  Yes   Yes    No       Yes      No
1   Yes        Yes    No       Yes    Yes  Yes   Yes   Yes       Yes      No
2    No        Yes   Yes       Yes    Yes  Yes    No   Yes       Yes     Yes
3   Yes        Yes    No       Yes    Yes  Yes   Yes   Yes        No      No
4    No        Yes    No       Yes    Yes  Yes   Yes    No        No     Yes

  disgusting Like  Age      VisitFrequency  Gender
0         No   -3   61  Every three months  Female
1         No   +2   51  Every three months  Female
2         No   +1   62  Every three months  Female
3        Yes   +4   69         Once a week  Female
4         No   +2   49        Once a month    Male
```

```
In [11]: # Convert the first 11 columns to a matrix
         D_x = data.iloc[:, 0:11].to_numpy()
```

```
In [12]: # Convert "Yes" values to 1 and "No" values to 0
         D_x = (D_x == "Yes") + 0
```

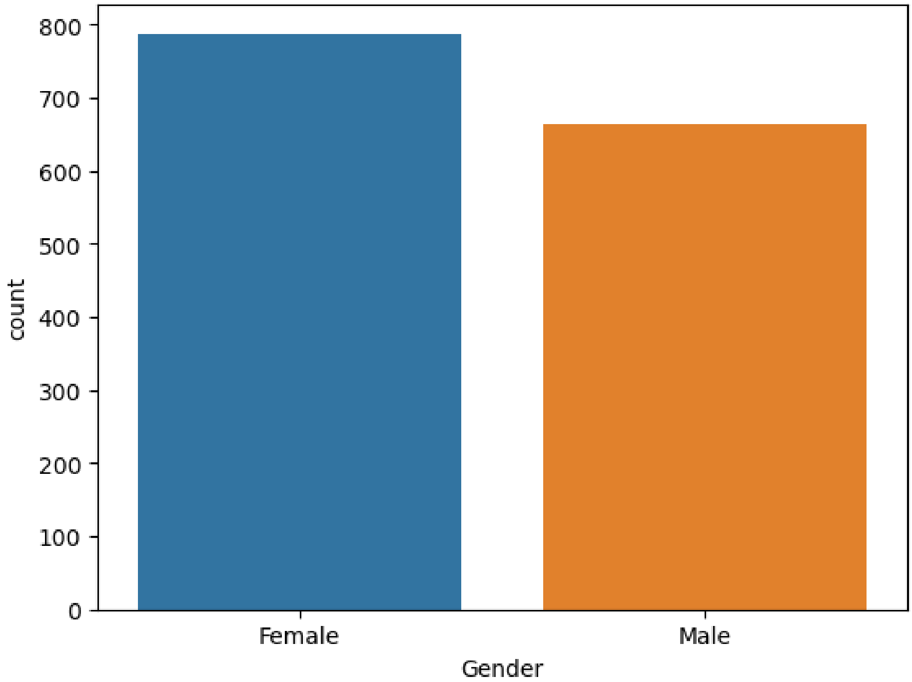
```
In [13]: # Calculate the column means and round to 2 decimal places
         col_means = np.round(np.mean(D_x, axis=0), 2)

         print(col_means)
```

```
[0.55 0.91 0.1  0.87 0.53 0.9  0.6  0.64 0.36 0.2  0.24]
```

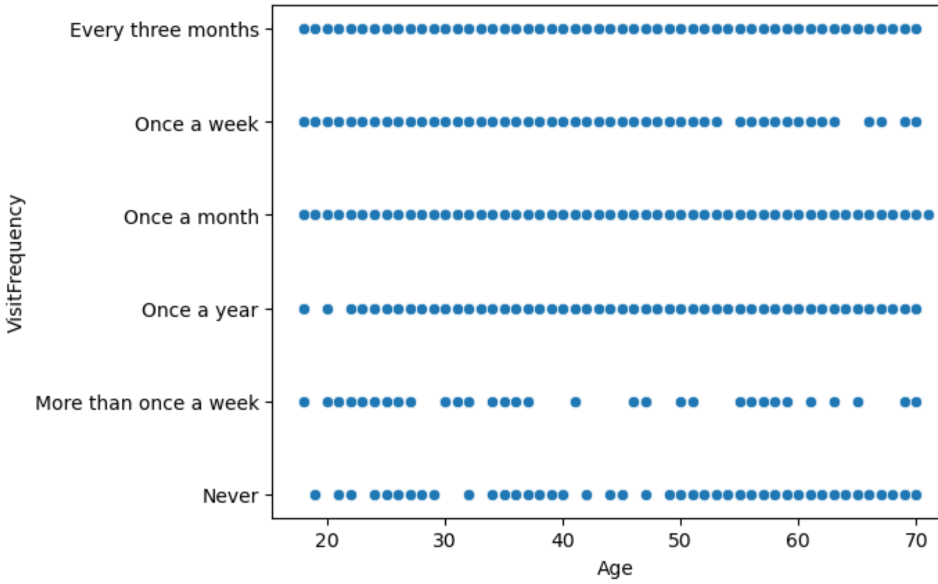
Descriptive Analysis: descriptive analysis is a fundamental tool for transforming raw data into clear and concise information, making it easier to draw meaningful conclusions.

```
In [85]: #Descriptive Analysis on data
         # Frequency distribution
         print(data['Gender'].value_counts())
         sns.countplot(x='Gender', data=data)
         plt.show()
```

```
Female    788
Male      665
Name: Gender, dtype: int64
```




```
In [87]: # Data visualization
         sns.scatterplot(x='Age', y='VisitFrequency', data=data)
         plt.show()
```



Pre-Processing : Two pre-processing procedures are often used for categorical variables. One is merging levels of categorical variables before further analysis, the other one is converting categorical variables to numeric ones, if it makes sense to do so.

In [95]:
```python
# Pre-Processing
# sort the frequency table of 'VisitFrequency ' column
# Get the categorical columns
categorical_cols = mcdonalds.select_dtypes(include=['object']).columns

# Apply one-hot encoding to each categorical column
mcdonalds_ohe = pd.get_dummies(mcdonalds, columns=categorical_cols)
```

In [94]:
```python
print(mcdonalds_ohe.head())
```

```
   Age  yummy_No  yummy_Yes  convenient_No  convenient_Yes  spicy_No  \
0   61         1          0              0               1         1
1   51         0          1              0               1         1
2   62         1          0              0               1         0
3   69         0          1              0               1         1
4   49         1          0              0               1         1

   spicy_Yes  fattening_No  fattening_Yes  greasy_No  ...  Like_I hate it!-5  \
0          0             0              1          1  ...                  0
1          0             0              1          0  ...                  0
2          1             0              1          0  ...                  0
3          0             0              1          0  ...                  0
4          0             0              1          0  ...                  0

   Like_I love it!+5  VisitFrequency_Every three months  \
0                  0                                  1
1                  0                                  1
2                  0                                  1
3                  0                                  0
4                  0                                  0

   VisitFrequency_More than once a week  VisitFrequency_Never  \
0                                     0                     0
1                                     0                     0
2                                     0                     0
3                                     0                     0
4                                     0                     0

   VisitFrequency_Once a month  VisitFrequency_Once a week  \
0                            0                           0
1                            0                           0
2                            0                           0
3                            0                           1
4                            1                           0

   VisitFrequency_Once a year  Gender_Female  Gender_Male
0                           0              1            0
1                           0              1            0
2                           0              1            0

1                          0             1           0
2                          0             1           0
3                          0             1           0
4                          0             0           1

[5 rows x 42 columns]
```

In [97]:
```python
# Select the categorical columns (yummy to disgusting)
categorical_cols = mcdonalds[['yummy', 'convenient', 'fattening', 'greasy', 'fast', 'cheap', 'tasty', 'expensive', 'healthy', 'di

# Convert the categorical columns to binary indicators (0 or 1)
vacmot = (categorical_cols == 'yes').astype(int)

print(vacmot.head())
```

```
   yummy  convenient  fattening  greasy  fast  cheap  tasty  expensive  \
0      0           0          0       0     0      0      0          0
1      0           0          0       0     0      0      0          0
2      0           0          0       0     0      0      0          0
3      0           0          0       0     0      0      0          0
4      0           0          0       0     0      0      0          0

   healthy  disgusting
0        0           0
1        0           0
2        0           0
3        0           0
4        0           0
```

```
In [100]: #pre-processing on numeric value
          data['Age'].fillna(data['Age'].mean(), inplace=True)
```

```
In [101]: # Data Normalization
          from sklearn.preprocessing import MinMaxScaler
          scaler = MinMaxScaler()
          data['Age'] = scaler.fit_transform(data[['Age']])
```

```
In [102]: #Outlier Detection and Handling
          from scipy import stats

          z_scores = np.abs(stats.zscore(data['Age']))
          data = data[(z_scores < 3)]
```

```
In [104]: #Binning
          from sklearn.preprocessing import KBinsDiscretizer

          discretizer = KBinsDiscretizer(n_bins=5, encode='ordinal')
          data['Age'] = discretizer.fit_transform(data[['Age']])
```

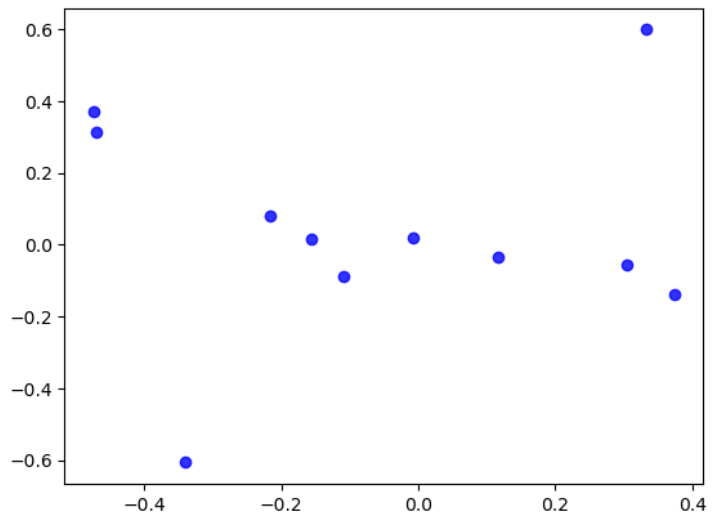
PCA (Principal component analysis):

```
In [22]: # Fit PCA to the data
         D_pca = PCA().fit(D_x)

         # Print the summary of the PCA results
         print("PCA Summary:")
         print("-------------")
         print("Proportion of variance explained by each PC:")
         print(D_pca.explained_variance_ratio_)
         print("\nCumulative proportion of variance explained:")
         print(np.cumsum(D_pca.explained_variance_ratio_))
```

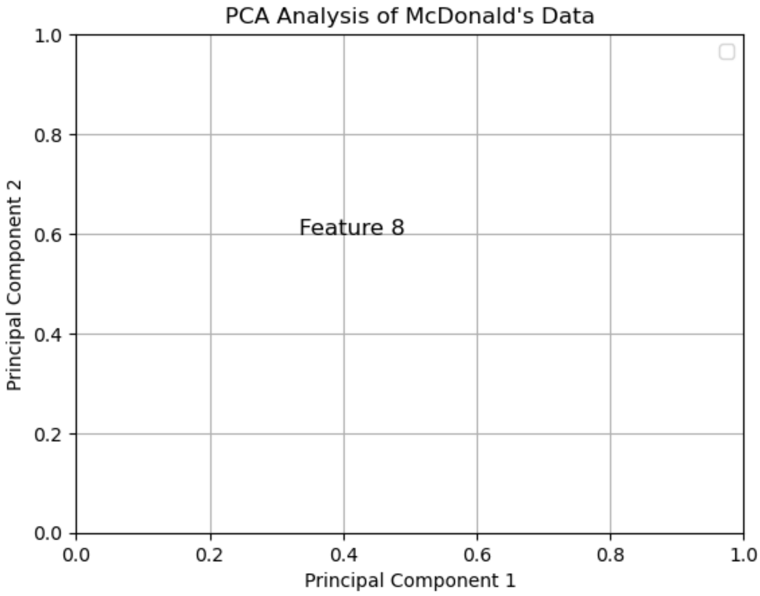
```
PCA Summary:
-------------
Proportion of variance explained by each PC:
[0.29899056 0.19156392 0.13267983 0.08290307 0.05969759 0.05069322
 0.04429957 0.03985029 0.03715547 0.03260161 0.02956487]

Cumulative proportion of variance explained:
[0.29899056 0.49055448 0.62323431 0.70613738 0.76583497 0.81652819
 0.86082776 0.90067805 0.93783352 0.97043513 1.        ]
```

```
In [23]: # Scatter plot of the first two principal components
         plt.scatter(D_pca.components_[0], D_pca.components_[1], alpha=0.8, c='blue', marker='o', label='PC1 vs PC2')
```

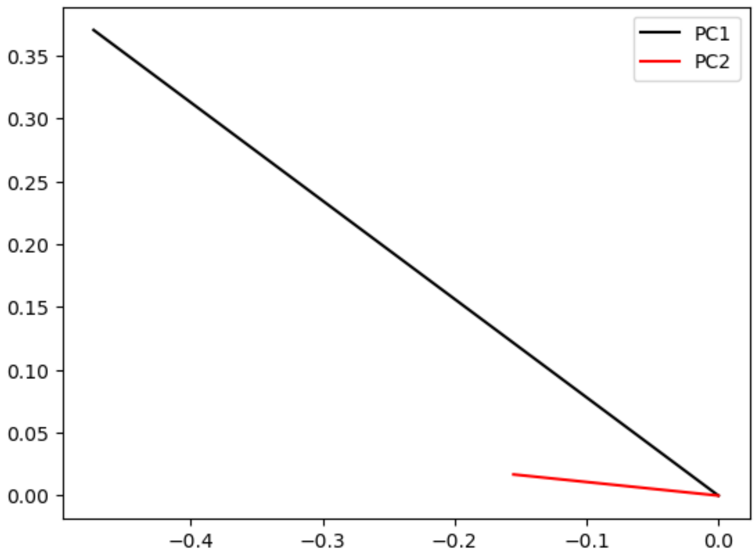
```
Out[23]: <matplotlib.collections.PathCollection at 0x20308d0e510>
```



```
In [24]: # Annotation for each feature
for i, feature in enumerate(range(11)):  # Replace range(11) with your actual feature names or indices
    plt.annotate(f'Feature {feature}', (D_pca.components_[0, i], D_pca.components_[1, i]), fontsize=12)
# Labels and title
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA Analysis of McDonald\'s Data')
# Add a legend with a single entry (dummy entry)
plt.legend(['PC1 vs PC2'], loc='upper right')
# Show plot
plt.grid()
plt.show()
```




```
In [25]: # plot the projection axes of the first two principal components
plt.plot([0, D_pca.components_[0, 0]], [0, D_pca.components_[1, 0]], 'k-', label='PC1')
plt.plot([0, D_pca.components_[0, 1]], [0, D_pca.components_[1, 1]], 'r-', label='PC2')
plt.legend()
plt.show()
```



https://github.com/prachikale2004/Market_Segmentation_Analysis-FeynnLab.git