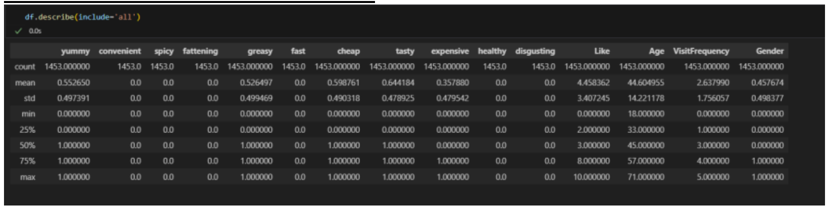
# Data Collection and Preprocessing Phase

| | |
|---|---|
| Date | 15 July 2024 |
| Team ID | 739844 |
| Project Title | MARKET SEGMENTATION ANALYSIS |
| Maximum Marks | 6 Marks |

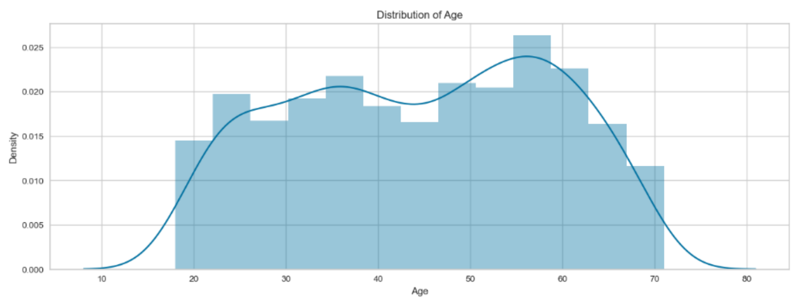
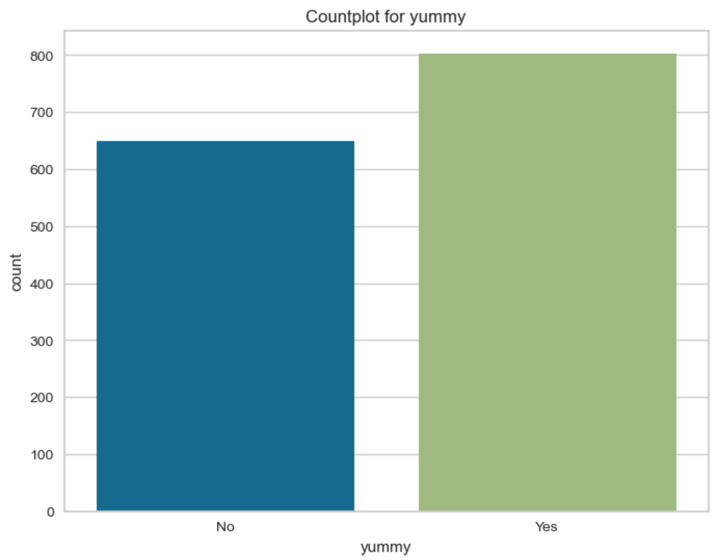
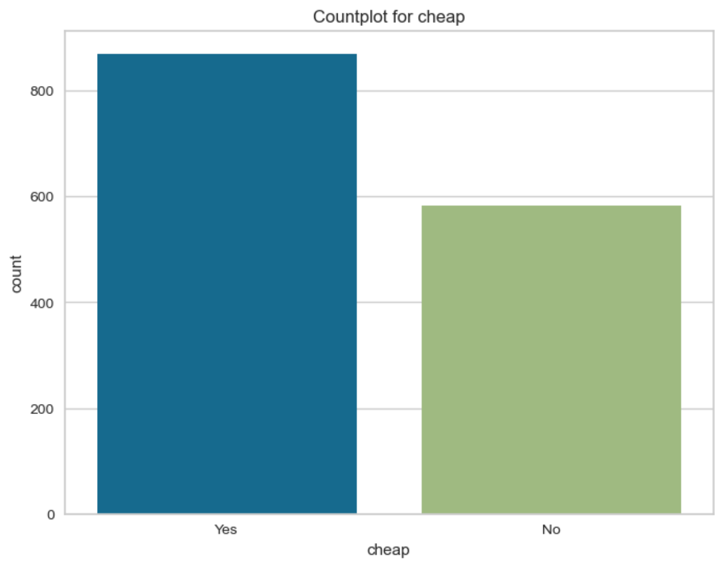
## Data Exploration and Preprocessing Template

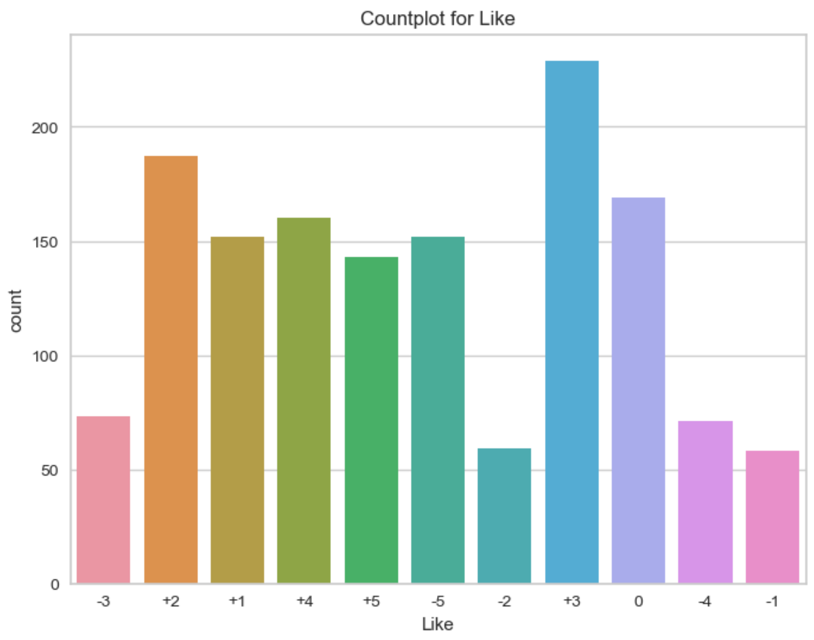
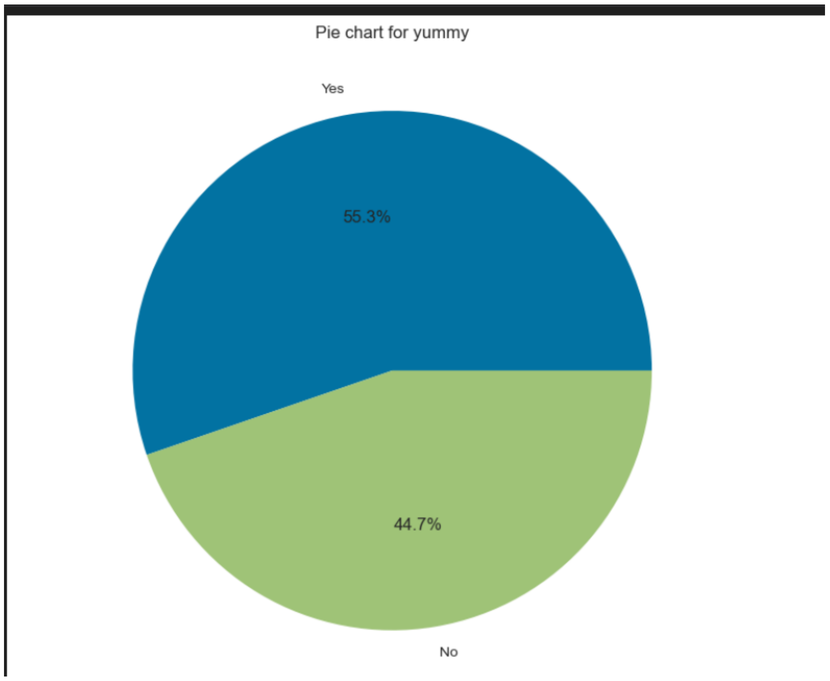
Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

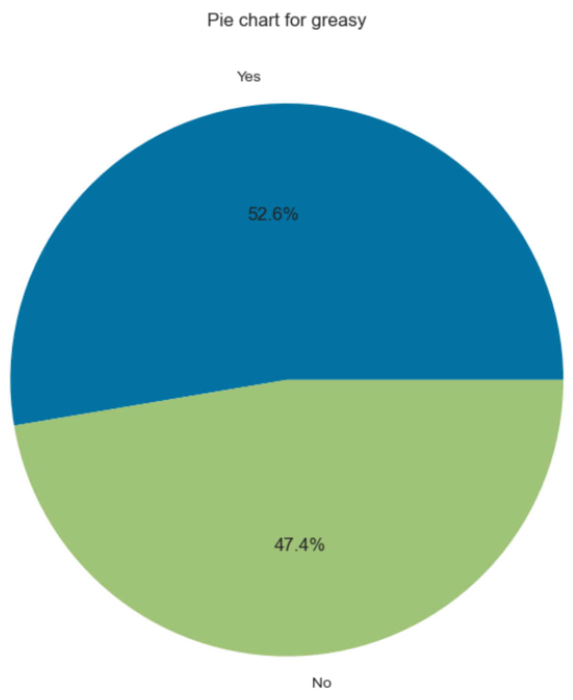
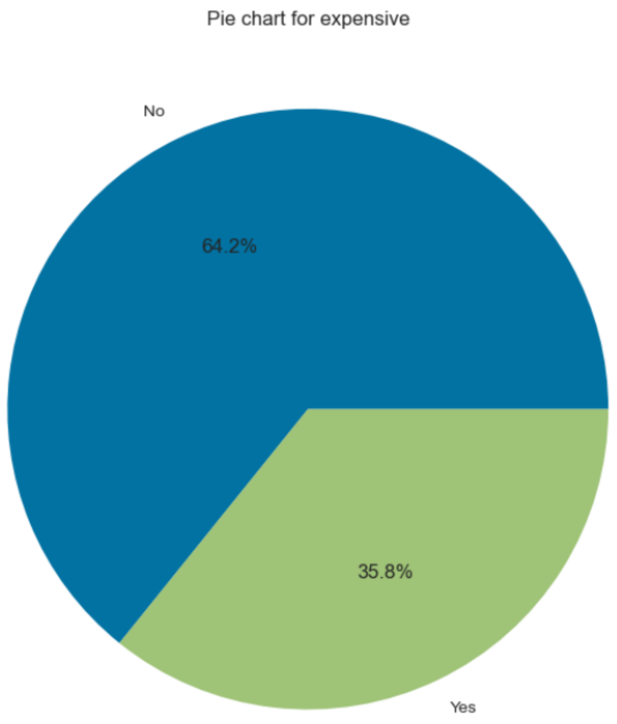
| **Section** | **Description** |
|---|---|
| Data Overview | DIMENSIONS=1453 ROWS X 15COLUMNS<br>DESCRIPTIVE ANALYSIS:<br> |

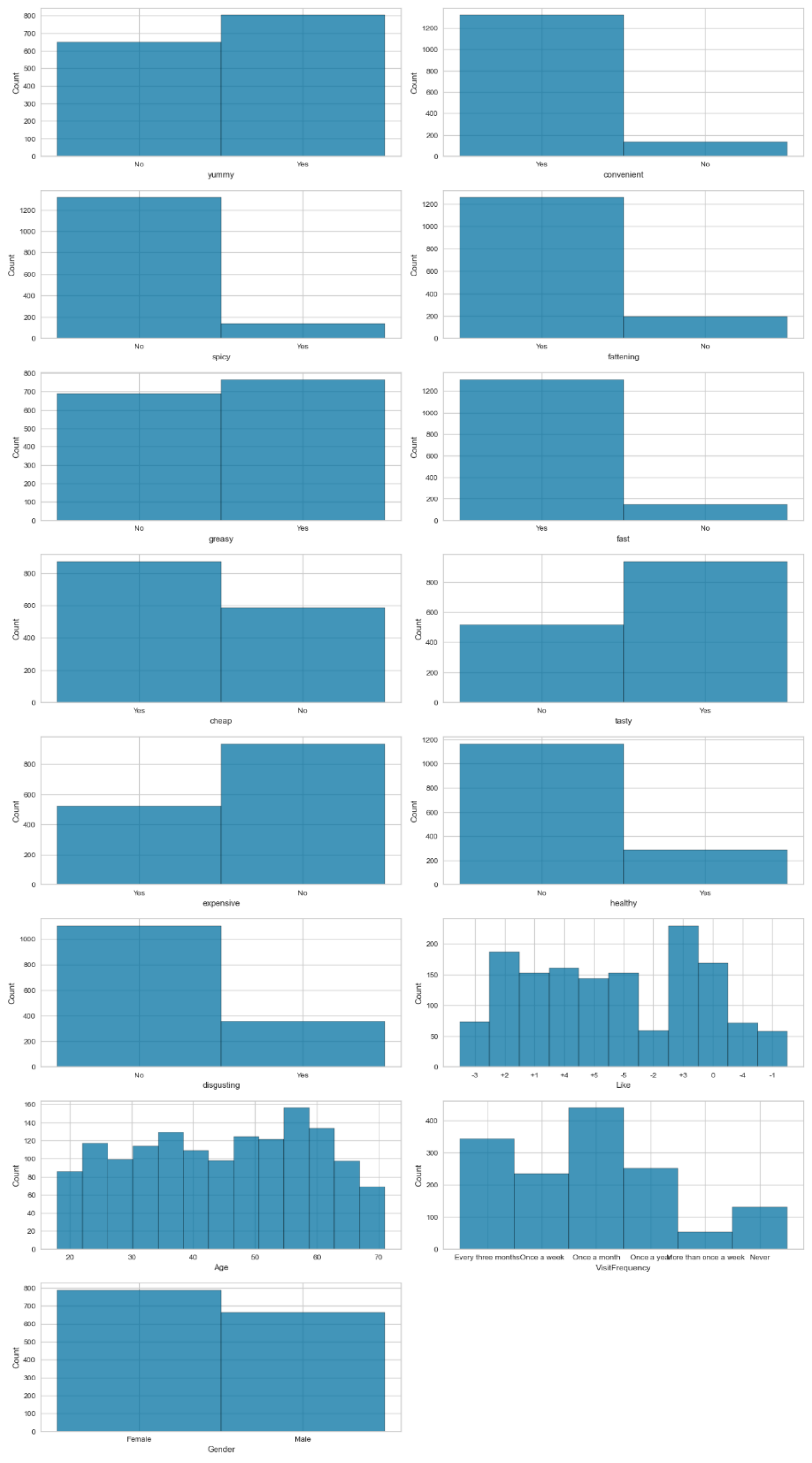
```
df.describe(include='all')
✓ 0.0s
```

```
            yummy  convenient   spicy  fattening       greasy    fast        cheap        tasty    expensive  healthy  disgusting         Like          Age  VisitFrequency       Gender
count  1453.000000     1453.0  1453.0     1453.0  1453.000000  1453.0  1453.000000  1453.000000  1453.000000   1453.0      1453.0  1453.000000  1453.000000     1453.000000  1453.000000
mean      0.552650        0.0     0.0        0.0     0.526497     0.0     0.598761     0.644184     0.357880      0.0         0.0     4.458362    44.604955        2.637990     0.457674
std       0.497391        0.0     0.0        0.0     0.499469     0.0     0.490318     0.478925     0.479542      0.0         0.0     3.407245    14.221178        1.756057     0.498377
min       0.000000        0.0     0.0        0.0     0.000000     0.0     0.000000     0.000000     0.000000      0.0         0.0     0.000000    18.000000        0.000000     0.000000
25%       0.000000        0.0     0.0        0.0     0.000000     0.0     0.000000     0.000000     0.000000      0.0         0.0     2.000000    33.000000        1.000000     0.000000
50%       1.000000        0.0     0.0        0.0     1.000000     0.0     1.000000     1.000000     0.000000      0.0         0.0     3.000000    45.000000        3.000000     0.000000
75%       1.000000        0.0     0.0        0.0     1.000000     0.0     1.000000     1.000000     1.000000      0.0         0.0     8.000000    57.000000        4.000000     1.000000
max       1.000000        0.0     0.0        0.0     1.000000     0.0     1.000000     1.000000     1.000000      0.0         0.0    10.000000    71.000000        5.000000     1.000000
```

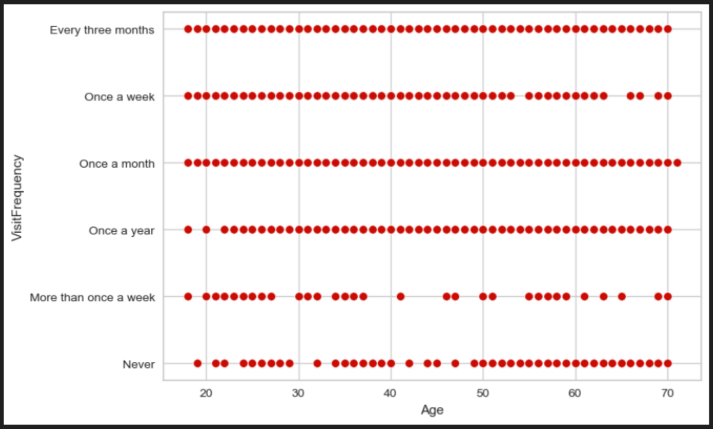
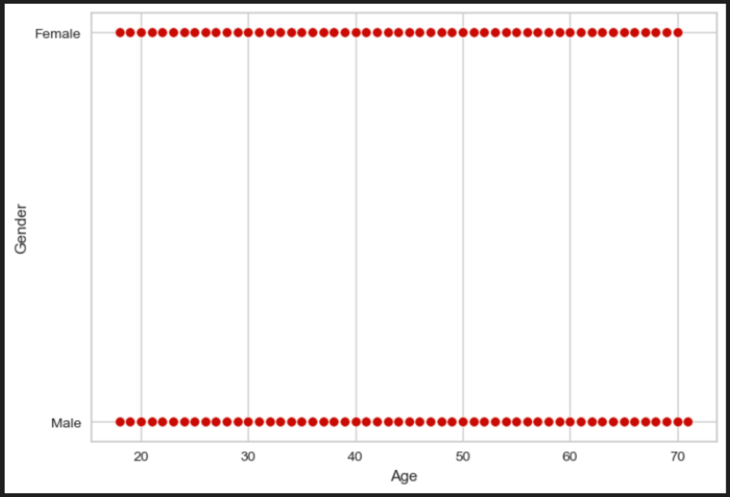
| | |
|---|---|
| Univariate Analysis |    |

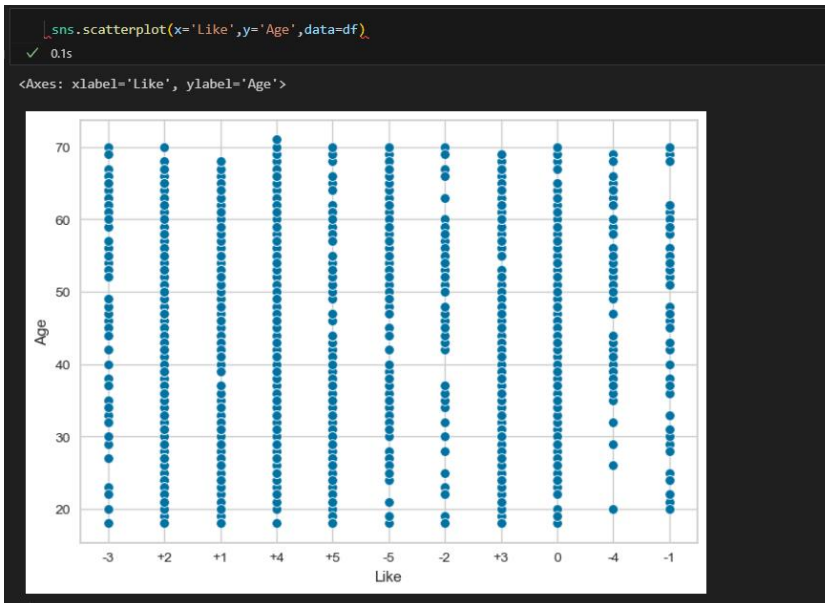
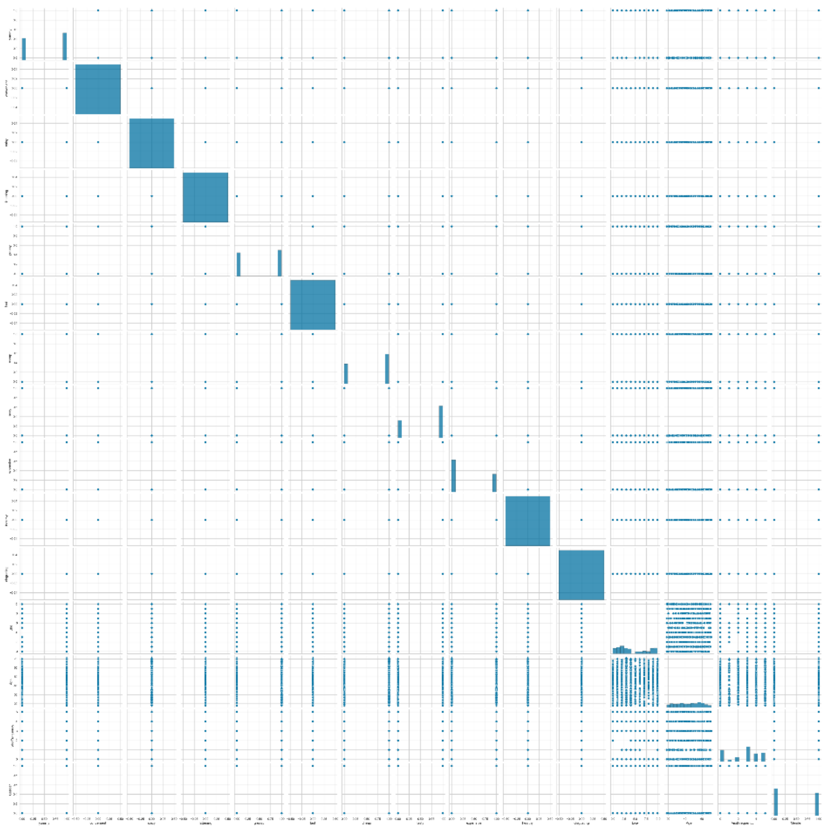
| | |
|---|---|
| |  Countplot for Like<br>count<br>200<br>150<br>100<br>50<br>0<br>-3<br>+2<br>+1<br>+4<br>+5<br>-5<br>-2<br>+3<br>0<br>-4<br>-1<br>Like  Pie chart for yummy<br>Yes<br>55.3%<br>44.7%<br>No |

|  |  |
|---|---|
|  | Pie chart for greasy<br>Yes<br>52.6%<br>47.4%<br>No<br>Pie chart for expensive<br>No<br>64.2%<br>35.8%<br>Yes |

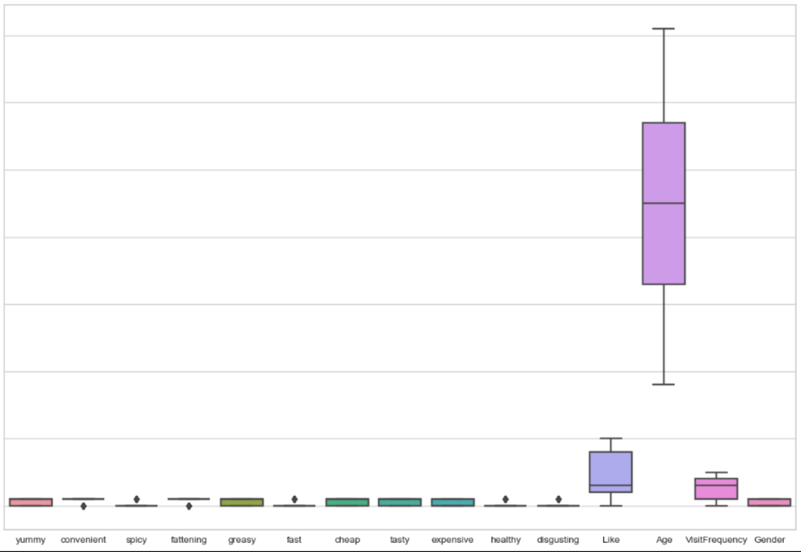
| | |
|---|---|
| |  |

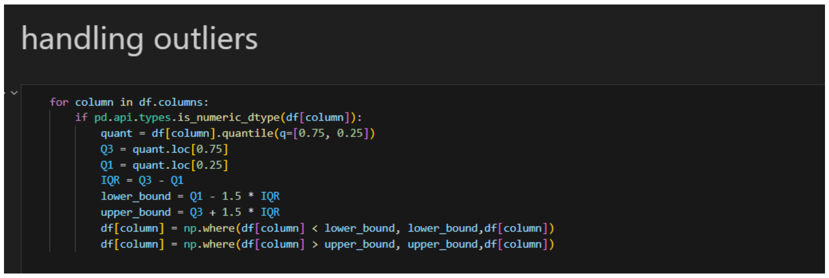
Count
yummy
No
Yes
convenient
spicy
fattening
greasy
fast
cheap
tasty
expensive
healthy
disgusting
Like
-3
+2
+1
+4
+5
-5
-2
+3
0
-4
-1
Age
VisitFrequency
Every three months
Once a week
Once a month
Once a year
More than once a week
Never
Gender
Female
Male

| | |
|---|---|
| Bivariate Analysis | bivariate analysis<br><br>scatter plots<br><br>`for i,col in enumerate(df.columns):`<br>`    sns.scatterplot(x='Age',y='VisitFrequency',data=df)`<br>✓ 0.4s<br> <br>`for i,col in enumerate(df.columns):`<br>`    sns.scatterplot(x='Age',y='Gender',data=df)`<br>✓ 0.7s<br> |

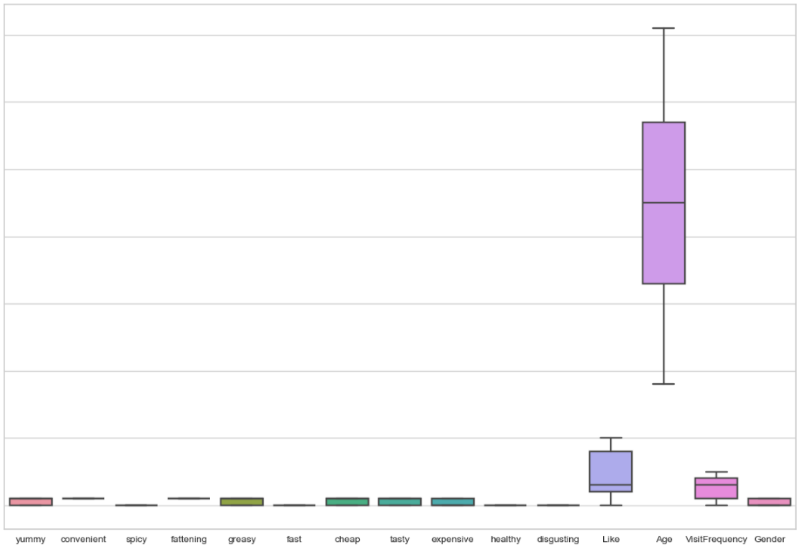
| | |
|---|---|
| |  ```sns.scatterplot(x='Like',y='Age',data=df)``` ✓ 0.1s ```<Axes: xlabel='Like', ylabel='Age'>
```

Age<br>70<br>60<br>50<br>40<br>30<br>20<br>-3<br>+2<br>+1<br>+4<br>+5<br>-5<br>-2<br>+3<br>0<br>-4<br>-1<br>Like

 |
| Multivariate Analysis |  • |
```

Outliers and Anomalies



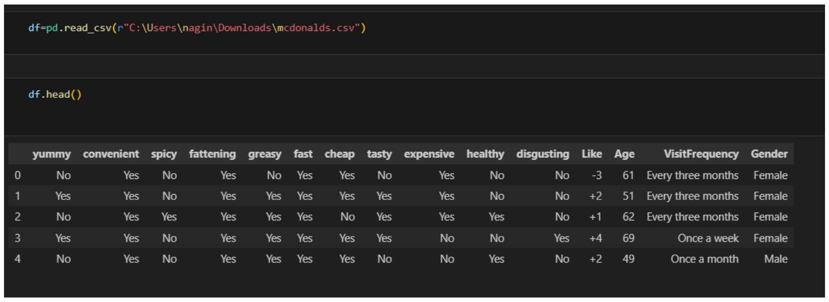

handling outliers

```
for column in df.columns:
    if pd.api.types.is_numeric_dtype(df[column]):
        quant = df[column].quantile(q=[0.75, 0.25])
        Q3 = quant.loc[0.75]
        Q1 = quant.loc[0.25]
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df[column] = np.where(df[column] < lower_bound, lower_bound,df[column])
        df[column] = np.where(df[column] > upper_bound, upper_bound,df[column])
```



**Data Preprocessing Code Screenshots**

Loading Data



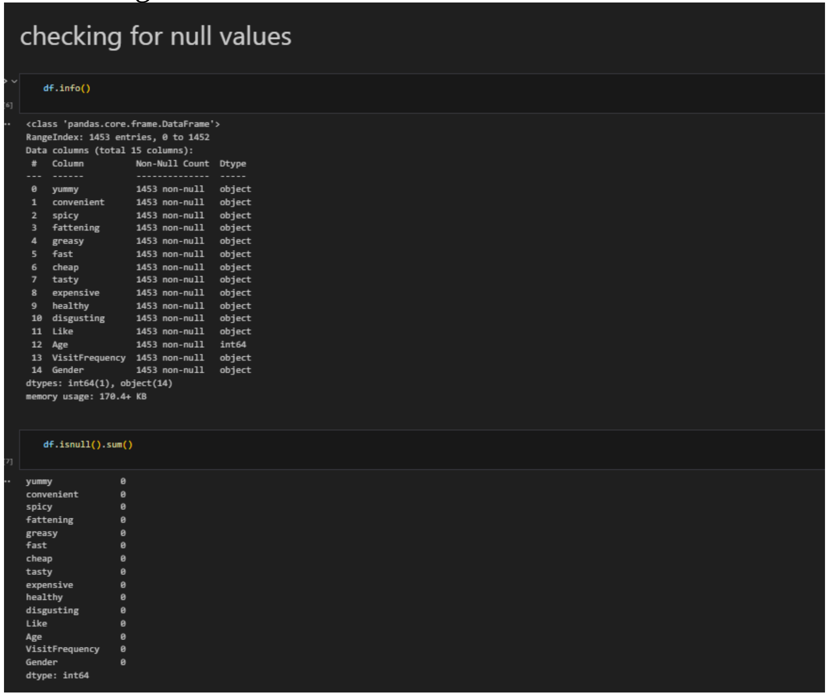
```
df=pd.read_csv(r"C:\Users\nagin\Downloads\mcdonalds.csv")
```

```
df.head()
```

|   | yummy | convenient | spicy | fattening | greasy | fast | cheap | tasty | expensive | healthy | disgusting | Like | Age | VisitFrequency | Gender |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | No | Yes | No | Yes | No | Yes | Yes | No | Yes | No | No | -3 | 61 | Every three months | Female |
| 1 | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes | No | No | +2 | 51 | Every three months | Female |
| 2 | No | Yes | Yes | Yes | Yes | Yes | No | Yes | Yes | Yes | No | +1 | 62 | Every three months | Female |
| 3 | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | No | No | Yes | +4 | 69 | Once a week | Female |
| 4 | No | Yes | No | Yes | Yes | Yes | Yes | No | No | Yes | No | +2 | 49 | Once a month | Male |

Handling Missing Data

No missing values found



checking for null values

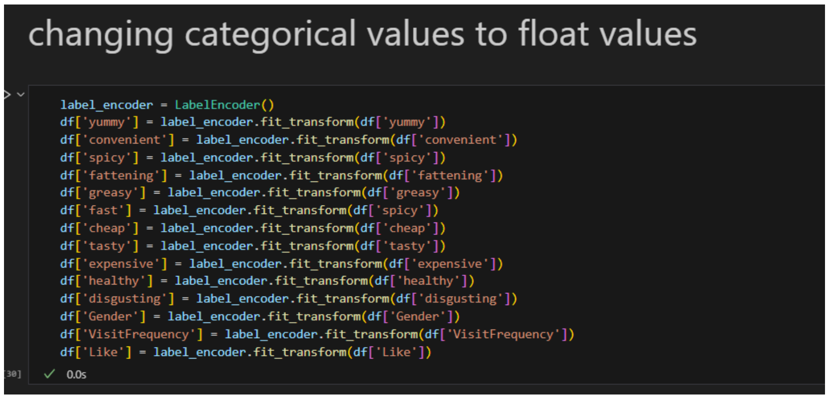
```
df.info()
```

```
<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
```

```
df.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
```

Data Transformation



changing categorical values to float values

```
label_encoder = LabelEncoder()
df['yummy'] = label_encoder.fit_transform(df['yummy'])
df['convenient'] = label_encoder.fit_transform(df['convenient'])
df['spicy'] = label_encoder.fit_transform(df['spicy'])
df['fattening'] = label_encoder.fit_transform(df['fattening'])
df['greasy'] = label_encoder.fit_transform(df['greasy'])
df['fast'] = label_encoder.fit_transform(df['spicy'])
df['cheap'] = label_encoder.fit_transform(df['cheap'])
df['tasty'] = label_encoder.fit_transform(df['tasty'])
df['expensive'] = label_encoder.fit_transform(df['expensive'])
df['healthy'] = label_encoder.fit_transform(df['healthy'])
df['disgusting'] = label_encoder.fit_transform(df['disgusting'])
df['Gender'] = label_encoder.fit_transform(df['Gender'])
df['VisitFrequency'] = label_encoder.fit_transform(df['VisitFrequency'])
df['Like'] = label_encoder.fit_transform(df['Like'])
```

[30] ✓ 0.0s

Feature Engineering

Attached the codes in final documentation

| | |
|---|---|
| Save Processed Data | - |