D212T2 Final

December 3, 2024

[1]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

[2]:
```
df = pd.read_csv('churn_clean.csv')
```

[3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   CaseOrder             10000 non-null  int64
 1   Customer_id           10000 non-null  object
 2   Interaction           10000 non-null  object
 3   UID                   10000 non-null  object
 4   City                  10000 non-null  object
 5   State                 10000 non-null  object
 6   County                10000 non-null  object
 7   Zip                   10000 non-null  int64
 8   Lat                   10000 non-null  float64
 9   Lng                   10000 non-null  float64
 10  Population            10000 non-null  int64
 11  Area                  10000 non-null  object
 12  TimeZone              10000 non-null  object
 13  Job                   10000 non-null  object
 14  Children              10000 non-null  int64
 15  Age                   10000 non-null  int64
 16  Income                10000 non-null  float64
 17  Marital               10000 non-null  object
 18  Gender                10000 non-null  object
 19  Churn                 10000 non-null  object
 20  Outage_sec_perweek    10000 non-null  float64
 21  Email                 10000 non-null  int64
```

```
 22  Contacts               10000 non-null  int64
 23  Yearly_equip_failure   10000 non-null  int64
 24  Techie                 10000 non-null  object
 25  Contract               10000 non-null  object
 26  Port_modem             10000 non-null  object
 27  Tablet                 10000 non-null  object
 28  InternetService        10000 non-null  object
 29  Phone                  10000 non-null  object
 30  Multiple               10000 non-null  object
 31  OnlineSecurity         10000 non-null  object
 32  OnlineBackup           10000 non-null  object
 33  DeviceProtection       10000 non-null  object
 34  TechSupport            10000 non-null  object
 35  StreamingTV            10000 non-null  object
 36  StreamingMovies        10000 non-null  object
 37  PaperlessBilling       10000 non-null  object
 38  PaymentMethod          10000 non-null  object
 39  Tenure                 10000 non-null  float64
 40  MonthlyCharge          10000 non-null  float64
 41  Bandwidth_GB_Year      10000 non-null  float64
 42  Item1                  10000 non-null  int64
 43  Item2                  10000 non-null  int64
 44  Item3                  10000 non-null  int64
 45  Item4                  10000 non-null  int64
 46  Item5                  10000 non-null  int64
 47  Item6                  10000 non-null  int64
 48  Item7                  10000 non-null  int64
 49  Item8                  10000 non-null  int64
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

[4]:
```
df.isnull().sum()
```

[4]:
```
CaseOrder             0
Customer_id           0
Interaction           0
UID                   0
City                  0
State                 0
County                0
Zip                   0
Lat                   0
Lng                   0
Population            0
Area                  0
TimeZone              0
Job                   0
```

```
Children                0
Age                     0
Income                  0
Marital                 0
Gender                  0
Churn                   0
Outage_sec_perweek      0
Email                   0
Contacts                0
Yearly_equip_failure    0
Techie                  0
Contract                0
Port_modem              0
Tablet                  0
InternetService         0
Phone                   0
Multiple                0
OnlineSecurity          0
OnlineBackup            0
DeviceProtection        0
TechSupport             0
StreamingTV             0
StreamingMovies         0
PaperlessBilling        0
PaymentMethod           0
Tenure                  0
MonthlyCharge           0
Bandwidth_GB_Year       0
Item1                   0
Item2                   0
Item3                   0
Item4                   0
Item5                   0
Item6                   0
Item7                   0
Item8                   0
dtype: int64
```

[5]:
```
df.duplicated()
```

[5]:
```
0       False
1       False
2       False
3       False
4       False
        …
9995    False
```

```
9996    False
9997    False
9998    False
9999    False
Length: 10000, dtype: bool
```

[6]:
```
df1=df.select_dtypes(exclude='object')
df1.columns
```

[6]:
```
Index(['CaseOrder', 'Zip', 'Lat', 'Lng', 'Population', 'Children', 'Age',
       'Income', 'Outage_sec_perweek', 'Email', 'Contacts',
       'Yearly_equip_failure', 'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year',
       'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8'],
      dtype='object')
```

[7]:
```
df1.shape
```

[7]:
```
(10000, 23)
```

[8]:
```
df2=df.select_dtypes(exclude='number')
df2.columns
```

[8]:
```
Index(['Customer_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Area',
       'TimeZone', 'Job', 'Marital', 'Gender', 'Churn', 'Techie', 'Contract',
       'Port_modem', 'Tablet', 'InternetService', 'Phone', 'Multiple',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'PaperlessBilling', 'PaymentMethod'],
      dtype='object')
```

[9]:
```
cont_df=df1.drop(['CaseOrder','Zip','Children',],axis=1)
cont_df.shape
```

[9]:
```
(10000, 20)
```

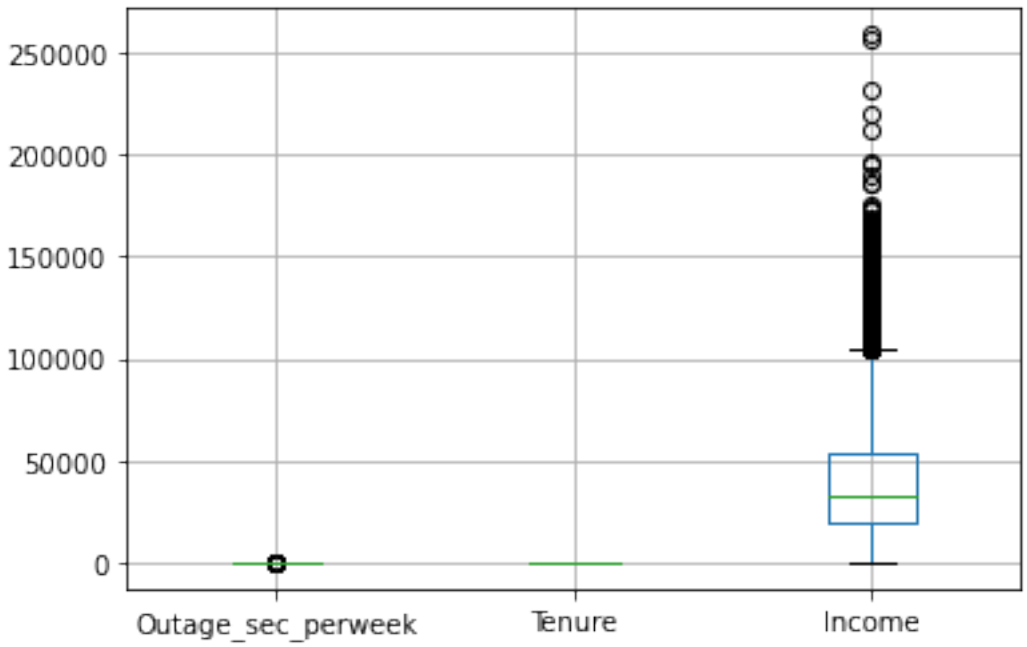
[10]:
```
columns_to_keep = ['Population', 'Age', 'Income', 'Tenure', 'MonthlyCharge',␣
↪'Bandwidth_GB_Year', 'Children', 'Outage_sec_perweek']
```

[11]:
```
df = df[columns_to_keep]
```

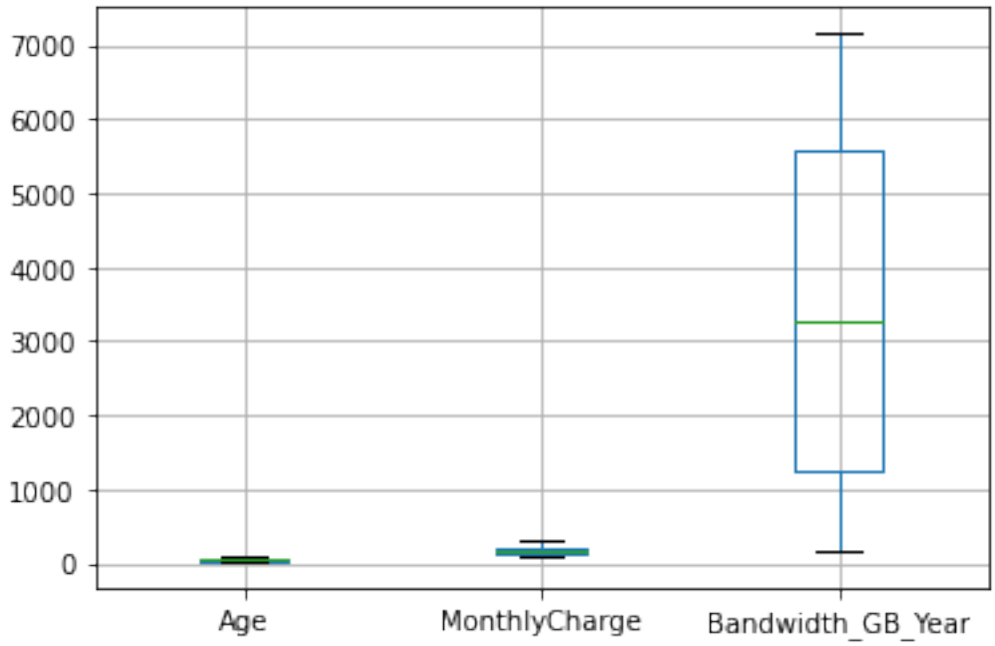
[12]:
```
print(df.columns)
```

```
Index(['Population', 'Age', 'Income', 'Tenure', 'MonthlyCharge',
       'Bandwidth_GB_Year', 'Children', 'Outage_sec_perweek'],
      dtype='object')
```

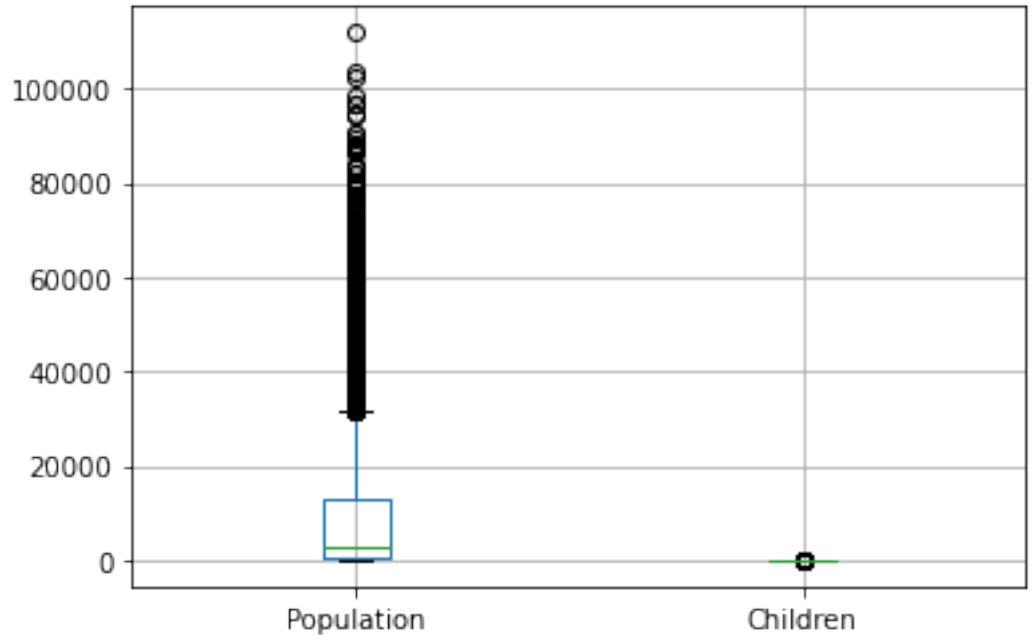
[13]:
```
df.boxplot(column=['Outage_sec_perweek', 'Tenure', 'Income'])
plt.show()
```



[14]:
```
df.boxplot(column=['Age','MonthlyCharge','Bandwidth_GB_Year'])
plt.show()
```



[15]:
```
df.boxplot(column=['Population','Children'])
plt.show()
```



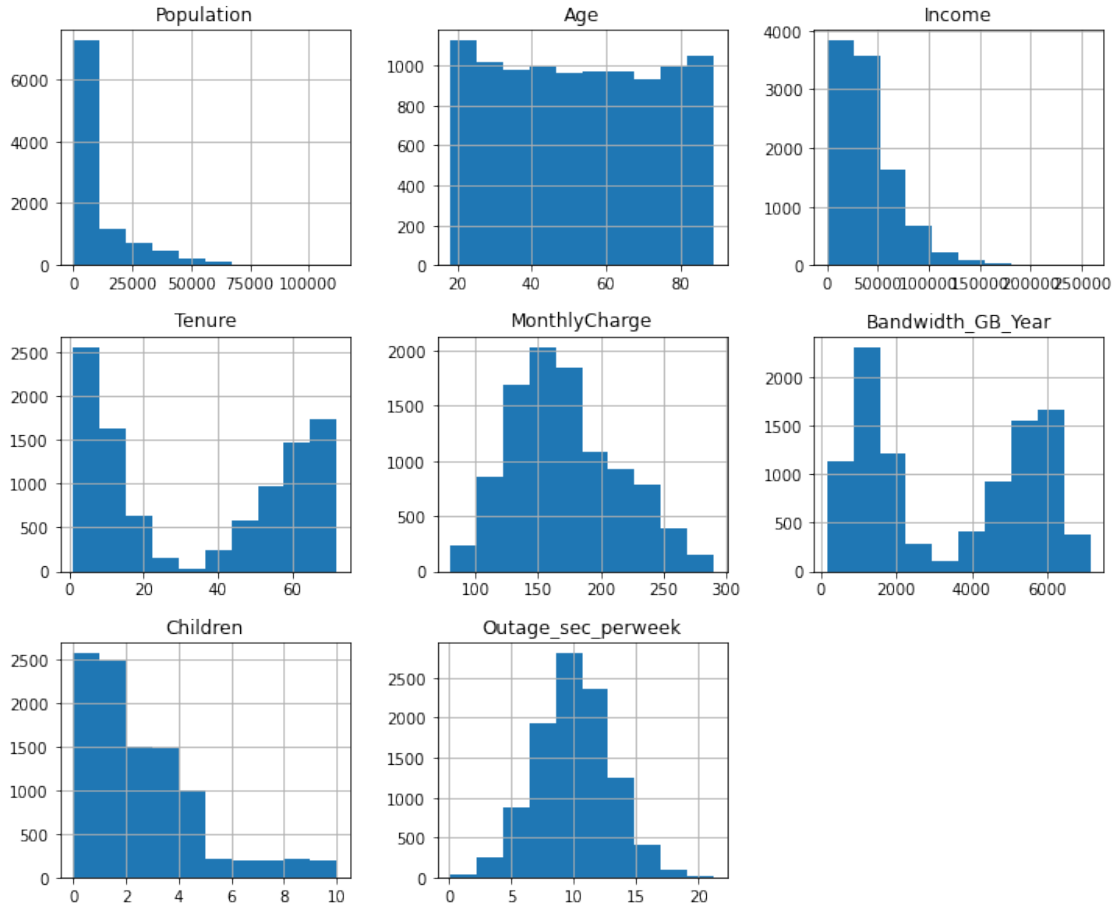
[16]:
```
df.describe()
```

[16]:

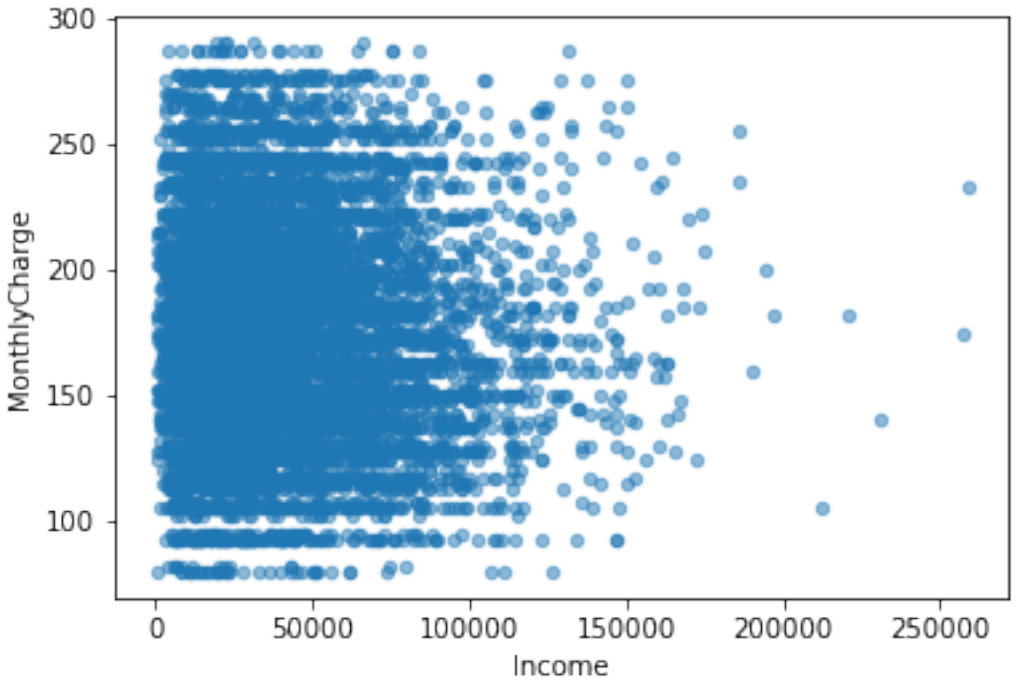
	Population	Age	Income	Tenure
count	10000.000000	10000.000000	10000.000000	10000.000000
mean	9756.562400	53.078400	39806.926771	34.526188
std	14432.698671	20.698882	28199.916702	26.443063
min	0.000000	18.000000	348.670000	1.000259
25%	738.000000	35.000000	19224.717500	7.917694
50%	2910.500000	53.000000	33170.605000	35.430507
75%	13168.000000	71.000000	53246.170000	61.479795
max	111850.000000	89.000000	258900.700000	71.999280

	MonthlyCharge	Bandwidth_GB_Year	Children	Outage_sec_perweek
count	10000.000000	10000.000000	10000.0000	10000.000000
mean	172.624816	3392.341550	2.0877	10.001848
std	42.943094	2185.294852	2.1472	2.976019
min	79.978860	155.506715	0.0000	0.099747
25%	139.979239	1236.470827	0.0000	8.018214
50%	167.484700	3279.536903	1.0000	10.018560
75%	200.734725	5586.141370	3.0000	11.969485
max	290.160419	7158.981530	10.0000	21.207230

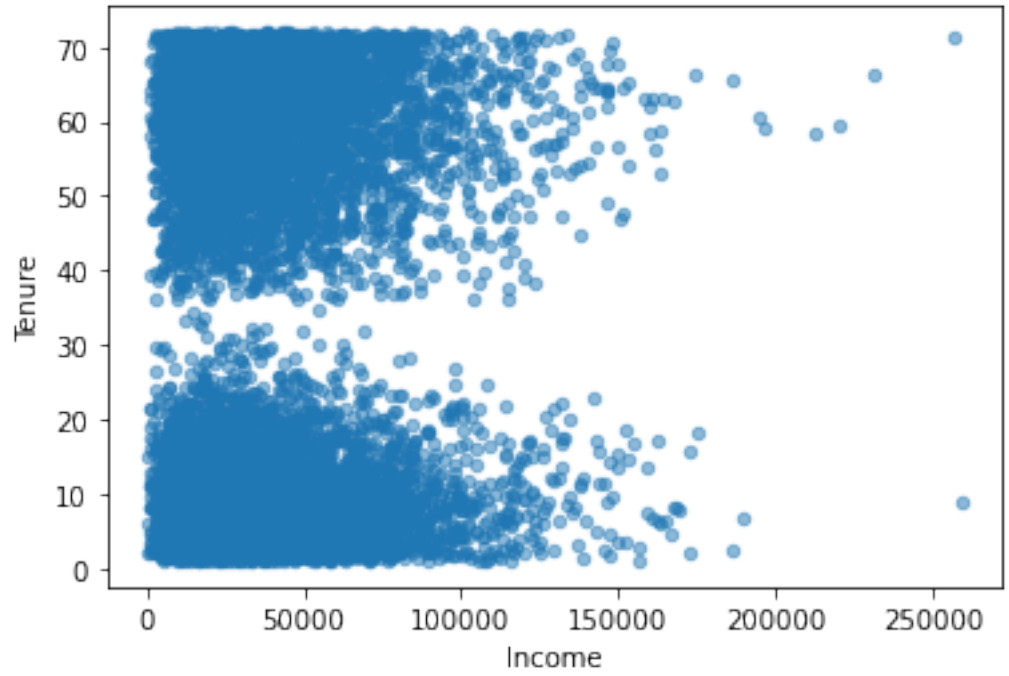
[17]:
```
df.hist(bins=10, figsize=(12, 10))
plt.show()
```



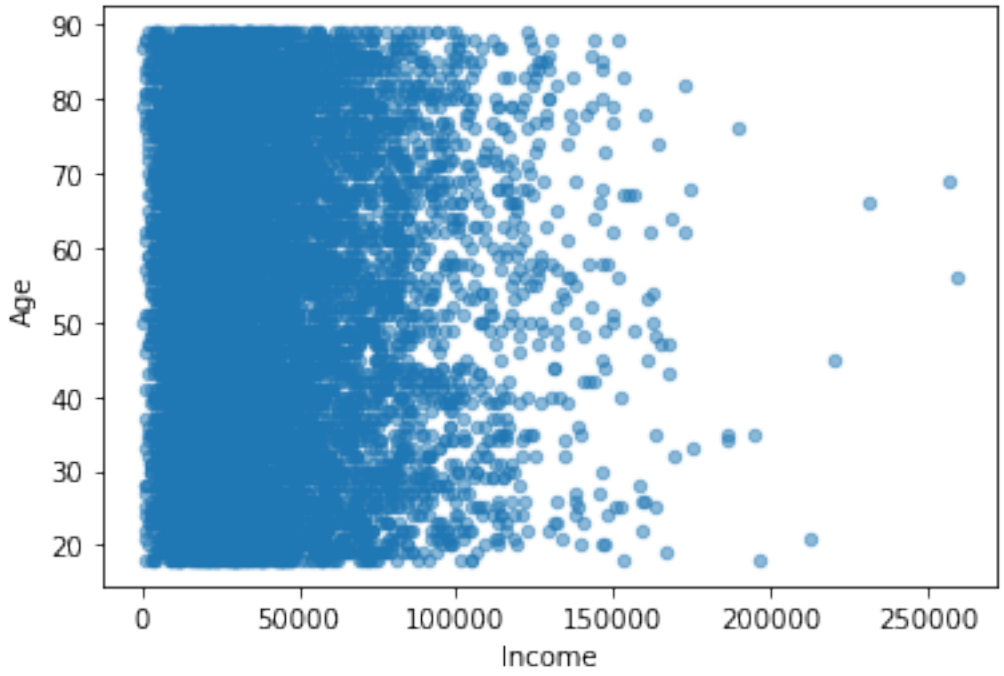
[18]:
```
df.plot(kind='scatter', x='Income', y='MonthlyCharge', alpha=0.5)
plt.show()
```



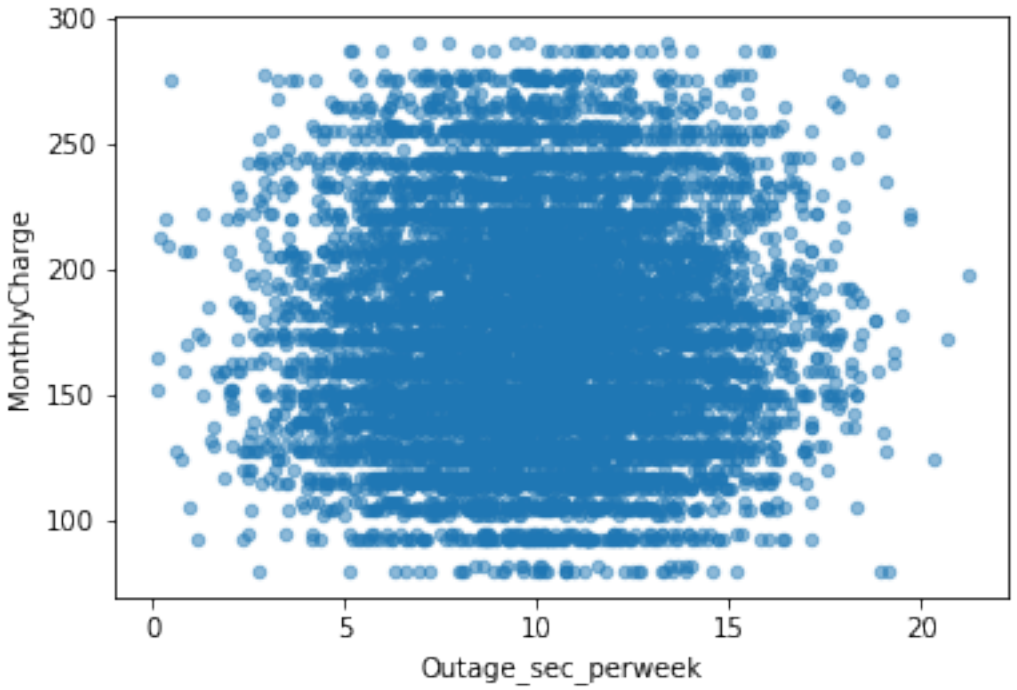
[19]:
```
df.plot(kind='scatter', x='Income', y='Tenure', alpha=0.5)
plt.show()
```



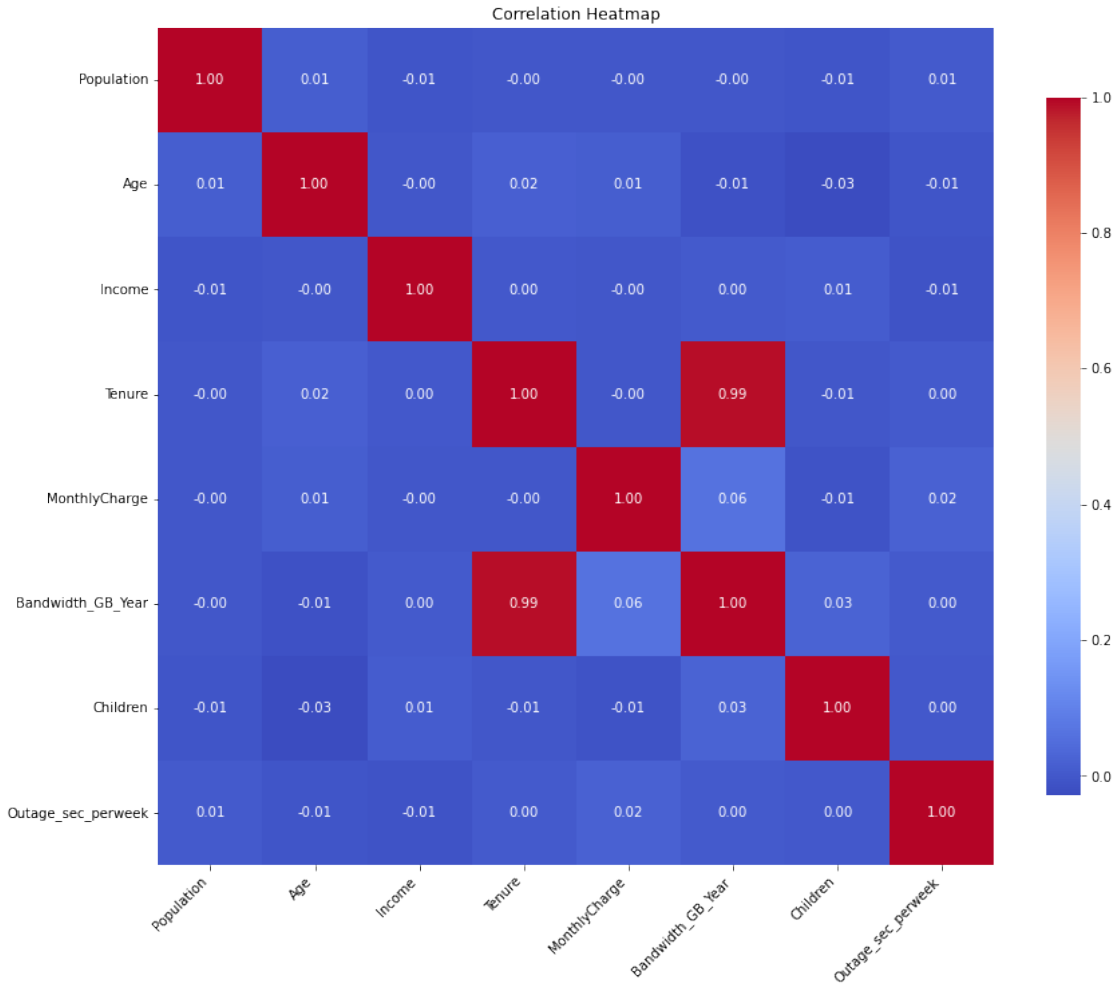
[20]:
```
df.plot(kind='scatter', x='Income', y='Age', alpha=0.5)
plt.show()
```



[21]:
```
df.plot(kind='scatter', x='Outage_sec_perweek', y='MonthlyCharge', alpha=0.5)
plt.show()
```



[22]:
```
correlation_matrix = df.corr()

plt.figure(figsize=(14, 12))  # Increase the size of the figure
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm',␣
 ↪square=True, cbar_kws={"shrink": .8})

plt.title('Correlation Heatmap')
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.show()
```



[23]:
```
from sklearn.preprocessing import StandardScaler

#  List of continuous variables to standardize
columns_to_standardize = ['Population', 'Age',
                          'Income', 'Tenure', 'MonthlyCharge',
                           'Bandwidth_GB_Year', 'Children',
                          'Outage_sec_perweek']

# Fit and transform the data
scaler = StandardScaler()
standardized_data = scaler.fit_transform(df[columns_to_standardize])

# Convert the standardized data back to a DataFrame
standardized_df = pd.DataFrame(standardized_data,␣
 ↪columns=columns_to_standardize)
```

[24]:
```
from sklearn.preprocessing import MinMaxScaler

# Normalize the standardized data
min_max_scaler = MinMaxScaler()
norm_data = min_max_scaler.fit_transform(standardized_df)

# Convert the normalized data back to a DataFrame
scaled_df = pd.DataFrame(norm_data, columns=standardized_df.columns)

# Save to CSV if needed and check the first few rows
scaled_df.to_csv('Scaled_df.csv', index=False)
scaled_df.head()
```

[24]:

	Population	Age	Income	Tenure	MonthlyCharge	Bandwidth_GB_Year	\
0	0.000340	0.704225	0.109120	0.081624	0.439985	0.106951	
1	0.093393	0.126761	0.082599	0.002203	0.773872	0.092165	
2	0.033393	0.450704	0.035818	0.207804	0.380474	0.271180	
3	0.123943	0.422535	0.071848	0.226580	0.190207	0.286868	
4	0.101493	0.915493	0.153646	0.009447	0.332900	0.016561	

	Children	Outage_sec_perweek
0	0.0	0.373260
1	0.1	0.549537
2	0.4	0.504705
3	0.1	0.701827
4	0.0	0.381271

[25]:
```
print(scaled_df.columns)
```

```
Index(['Population', 'Age', 'Income', 'Tenure', 'MonthlyCharge',
       'Bandwidth_GB_Year', 'Children', 'Outage_sec_perweek'],
      dtype='object')
```

[26]:
```
print(scaled_df.round(2))
```

	Population	Age	Income	Tenure	MonthlyCharge	Bandwidth_GB_Year	\
0	0.00	0.70	0.11	0.08	0.44	0.11	
1	0.09	0.13	0.08	0.00	0.77	0.09	
2	0.03	0.45	0.04	0.21	0.38	0.27	
3	0.12	0.42	0.07	0.23	0.19	0.29	
4	0.10	0.92	0.15	0.01	0.33	0.02	
…	…	…	…	…	…	…	
9995	0.01	0.07	0.21	0.95	0.38	0.91	
9996	0.69	0.42	0.13	0.85	0.61	0.79	
9997	0.00	0.42	0.18	0.65	0.43	0.57	
9998	0.32	0.30	0.06	0.99	0.82	0.90	
9999	0.11	0.14	0.03	0.88	0.65	0.81	

```
      Children  Outage_sec_perweek
0          0.0                0.37
1          0.1                0.55
2          0.4                0.50
3          0.1                0.70
4          0.0                0.38
…          …                  …
9995       0.3                0.44
9996       0.4                0.31
9997       0.1                0.31
9998       0.1                0.57
9999       0.1                0.55

[10000 rows x 8 columns]
```

[27]:
```
scaled_df.shape
```

[27]:
```
(10000, 8)
```

[28]:
```
covariance_matrix = np.cov(standardized_data.T)
```

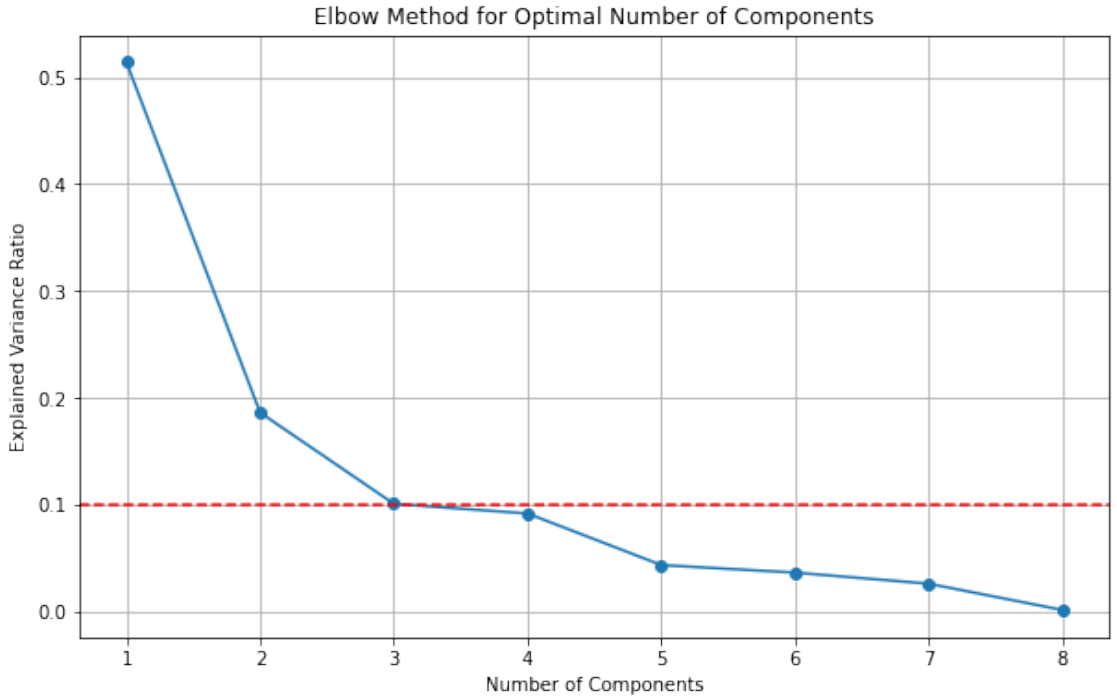
[29]:
```
covariance_matrix_df = pd.DataFrame(
    covariance_matrix,
    columns=columns_to_standardize,
    index=columns_to_standardize
)

print("Covariance Matrix:")
print(covariance_matrix_df)
```

```
Covariance Matrix:
                   Population       Age    Income    Tenure  MonthlyCharge  \
Population           1.000100  0.010539 -0.008639 -0.003560      -0.004779
Age                  0.010539  1.000100 -0.004091  0.016981       0.010730
Income              -0.008639 -0.004091  1.000100  0.002115      -0.003014
Tenure              -0.003560  0.016981  0.002115  1.000100      -0.003337
MonthlyCharge       -0.004779  0.010730 -0.003014 -0.003337       1.000100
Bandwidth_GB_Year   -0.003902 -0.014725  0.003674  0.991594       0.060412
Children            -0.005877 -0.029735  0.009943 -0.005092      -0.009782
Outage_sec_perweek   0.005484 -0.008048 -0.010012  0.002932       0.020498

                   Bandwidth_GB_Year  Children  Outage_sec_perweek
Population                -0.003902 -0.005877            0.005484
Age                       -0.014725 -0.029735           -0.008048
Income                     0.003674  0.009943           -0.010012
Tenure                     0.991594 -0.005092            0.002932
MonthlyCharge              0.060412 -0.009782            0.020498
Bandwidth_GB_Year          1.000100  0.025587            0.004176
```

```
Children                  0.025587  1.000100          0.001889
Outage_sec_perweek        0.004176  0.001889          1.000100
```

[30]:
```
pca = PCA()
pca.fit(scaled_df[columns_to_standardize])
explained_variance = pca.explained_variance_ratio_
```

[31]:
```
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(explained_variance) + 1), explained_variance, marker='o')
plt.title('Elbow Method for Optimal Number of Components')
plt.xlabel('Number of Components')
plt.ylabel('Explained Variance Ratio')
plt.grid()
plt.axhline(y=0.1, color='r', linestyle='--')  # Optional: Add a horizontal␣
 ↪line for reference
plt.show()
```



[32]:
```
# Fit PCA with the chosen number of components
optimal_n_components = 2 # Based on the elbow method
pca = PCA(n_components=optimal_n_components)
pca_result = pca.fit_transform(df[columns_to_standardize])
```

[33]:
```
PC_df = pd.DataFrame(data=pca_result, columns=['PC1', 'PC2'])
```

[34]:
```
loading_matrix = pd.DataFrame(pca.components_, columns=columns_to_standardize,␣
↪index=['PC1', 'PC2'])
```

[35]:
```
PC_df.head()
```

[35]:
```
             PC1          PC2
0 -11187.234677 -9784.257845
1 -18106.705877   582.537571
2 -30161.128539 -6201.542204
3 -20906.271624  3982.012057
4    256.804941  1598.877038
```

[36]:
```
loading_matrix
```

[36]:
```
     Population       Age    Income    Tenure  MonthlyCharge  \
PC1   -0.005990 -0.000003  0.999982  0.000002      -0.000005
PC2    0.999982  0.000015  0.005990 -0.000007      -0.000014

     Bandwidth_GB_Year      Children  Outage_sec_perweek
PC1           0.000287  7.584130e-07           -0.000001
PC2          -0.000598 -8.574206e-07            0.000001
```

[46]:
```
print(pca.explained_variance_ratio_)
```

```
[0.78869716 0.2065639 ]
```

[47]:
```
# Calculate total variance
total_variance = sum(eigenvalues)
print("Total Variance:", total_variance)
```

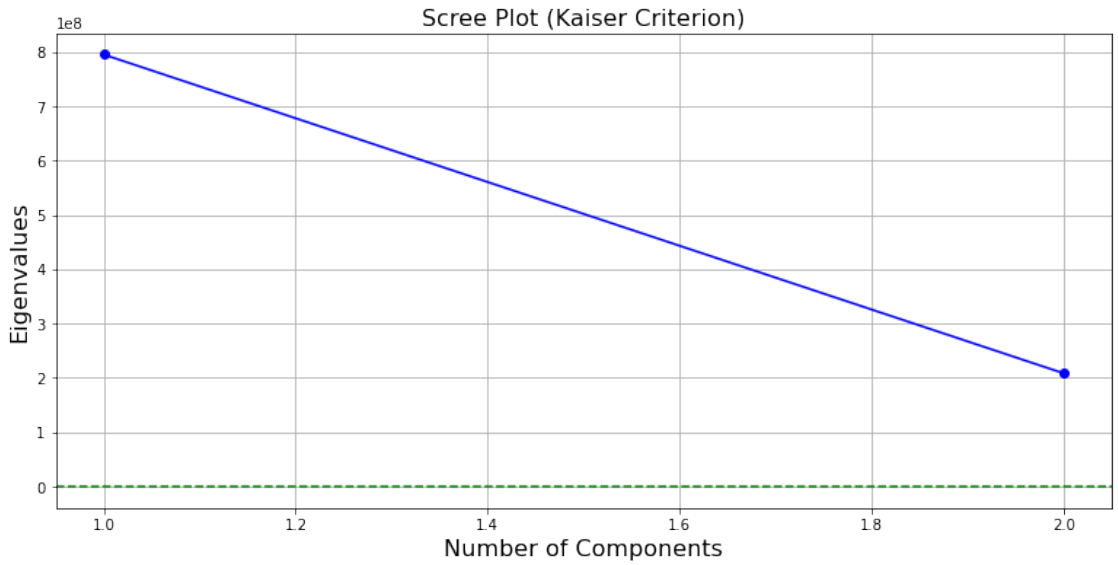
```
Total Variance: 1003538231.0681603
```

[48]:
```
# Retrieve eigenvalues
eigenvalues = pca.explained_variance_

# Apply the Kaiser criterion
kaiser_criterion = eigenvalues > 1
number_of_components_to_keep = np.sum(kaiser_criterion)

print("Eigenvalues:", eigenvalues)
print("Components to retain (Kaiser Criterion):", number_of_components_to_keep)
```

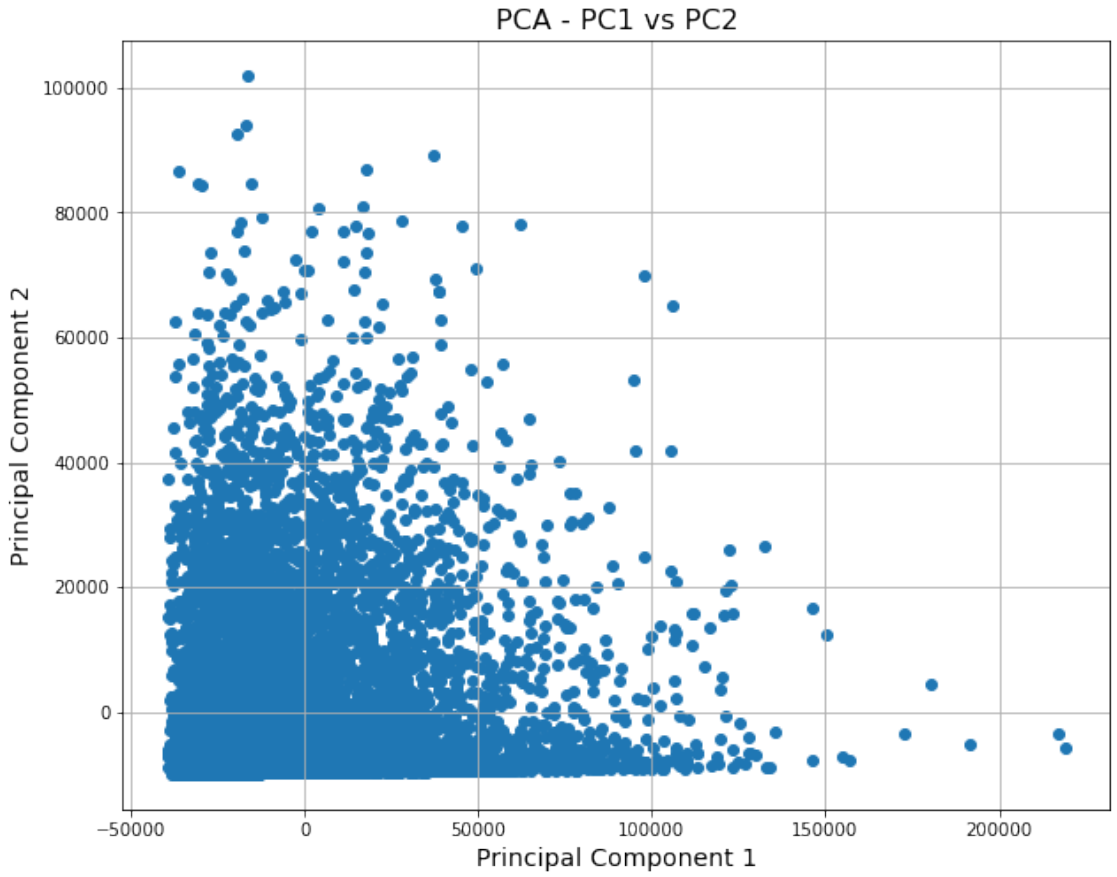
```
Eigenvalues: [7.95256428e+08 2.08281803e+08]
Components to retain (Kaiser Criterion): 2
```

[49]:
```
eigenvalues = pca.explained_variance_

pcomp = np.arange(1, len(eigenvalues) + 1)
```

```
# Plotting the scree plot
plt.figure(figsize=(13, 6))
plt.plot(pcomp, eigenvalues, 'b-', marker='o')  # Blue line with markers
plt.title('Scree Plot (Kaiser Criterion)', fontsize=16)
plt.xlabel('Number of Components', fontsize=16)
plt.ylabel('Eigenvalues', fontsize=16)
plt.axhline(y=1, color='g', linestyle='--')  # Green dashed line at y=1
plt.grid()
plt.show()
```



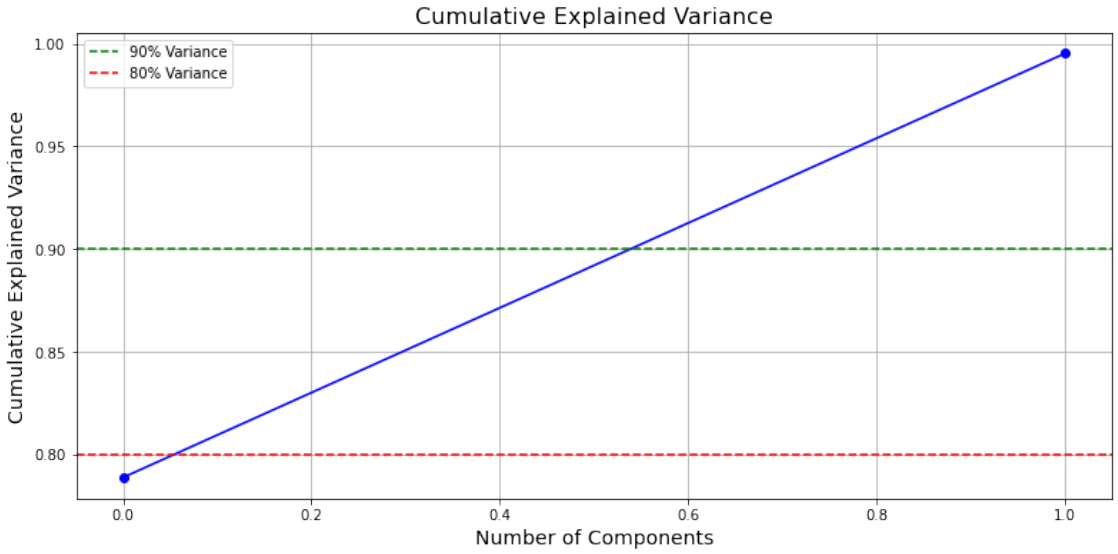
[50]:
```
# Visualize the first two principal components
plt.figure(figsize=(10, 8))
plt.scatter(PC_df['PC1'], PC_df['PC2'])
plt.title('PCA - PC1 vs PC2', fontsize=16)
plt.xlabel('Principal Component 1', fontsize=14)
plt.ylabel('Principal Component 2', fontsize=14)
plt.grid()
plt.show()
```



[51]:
```
# Calculate cumulative explained variance
cumulative_variance = np.cumsum(pca.explained_variance_ratio_)
print("Cumulative Explained Variance:", cumulative_variance)
```

```
Cumulative Explained Variance: [0.78869716 0.99526105]
```

[52]:
```
# Calculate cumulative variance
cumulative_variance = np.cumsum(pca.explained_variance_ratio_)

# Create a figure for cumulative variance
plt.figure(figsize=(13, 6))
plt.plot(cumulative_variance, marker='o', color='b')
plt.title('Cumulative Explained Variance', fontsize=16)
plt.xlabel('Number of Components', fontsize=14)
plt.ylabel('Cumulative Explained Variance', fontsize=14)
plt.axhline(y=0.90, color='g', linestyle='--', label='90% Variance')
plt.axhline(y=0.80, color='r', linestyle='--', label='80% Variance')
plt.grid()
plt.legend()
plt.show()
```


Cumulative Explained Variance
90% Variance
80% Variance
Cumulative Explained Variance
1.00
0.95
0.90
0.85
0.80
0.0
0.2
0.4
0.6
0.8
1.0
Number of Components

[ ]: