In [1]:

```
# Install necessary packages
!pip install pandas
!pip install numpy
!pip install scipy
!pip install scikit-learn
!pip install matplotlib
!pip install seaborn
```

```
Requirement already satisfied: pandas in c:\users\josue\anaconda3\envs\d206performan
ceassesment\lib\site-packages (2.2.1)
Requirement already satisfied: numpy<2,>=1.23.2 in c:\users\josue\anaconda3\envs\d20
6performanceassesment\lib\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\josue\anaconda3\en
vs\d206performanceassesment\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\josue\anaconda3\envs\d206per
formanceassesment\lib\site-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\josue\anaconda3\envs\d206p
erformanceassesment\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: six>=1.5 in c:\users\josue\anaconda3\envs\d206perform
anceassesment\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: numpy in c:\users\josue\anaconda3\envs\d206performanc
eassesment\lib\site-packages (1.26.4)
Requirement already satisfied: scipy in c:\users\josue\anaconda3\envs\d206performanc
eassesment\lib\site-packages (1.12.0)
Requirement already satisfied: numpy<1.29.0,>=1.22.4 in c:\users\josue\anaconda3\env
s\d206performanceassesment\lib\site-packages (from scipy) (1.26.4)
Requirement already satisfied: scikit-learn in c:\users\josue\anaconda3\envs\d206per
formanceassesment\lib\site-packages (1.4.1.post1)
Requirement already satisfied: numpy<2.0,>=1.19.5 in c:\users\josue\anaconda3\envs\d
206performanceassesment\lib\site-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in c:\users\josue\anaconda3\envs\d206per
formanceassesment\lib\site-packages (from scikit-learn) (1.12.0)
Requirement already satisfied: joblib>=1.2.0 in c:\users\josue\anaconda3\envs\d206pe
rformanceassesment\lib\site-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\josue\anaconda3\envs
\d206performanceassesment\lib\site-packages (from scikit-learn) (3.3.0)
Requirement already satisfied: matplotlib in c:\users\josue\anaconda3\envs\d206perfo
rmanceassesment\lib\site-packages (3.8.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\josue\anaconda3\envs\d20
6performanceassesment\lib\site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\josue\anaconda3\envs\d206per
formanceassesment\lib\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\josue\anaconda3\envs\d2
06performanceassesment\lib\site-packages (from matplotlib) (4.49.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\josue\anaconda3\envs\d2
06performanceassesment\lib\site-packages (from matplotlib) (1.4.5)
Requirement already satisfied: numpy<2,>=1.21 in c:\users\josue\anaconda3\envs\d206p
erformanceassesment\lib\site-packages (from matplotlib) (1.26.4)
Requirement already satisfied: packaging>=20.0 in c:\users\josue\anaconda3\envs\d206
performanceassesment\lib\site-packages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=8 in c:\users\josue\anaconda3\envs\d206perfor
manceassesment\lib\site-packages (from matplotlib) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\josue\anaconda3\envs\d20
6performanceassesment\lib\site-packages (from matplotlib) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\josue\anaconda3\envs
\d206performanceassesment\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\josue\anaconda3\envs\d206perform
anceassesment\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Requirement already satisfied: seaborn in c:\users\josue\anaconda3\envs\d206performa
nceassesment\lib\site-packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\josue\anaconda3\envs
\d206performanceassesment\lib\site-packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=1.2 in c:\users\josue\anaconda3\envs\d206perf
ormanceassesment\lib\site-packages (from seaborn) (2.2.1)
```

```
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in c:\users\josue\anaconda3\envs\d206performanceassesment\lib\site-packages (from seaborn) (3.8.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\josue\anaconda3\envs\d206performanceassesment\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\josue\anaconda3\envs\d206performanceassesment\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\josue\anaconda3\envs\d206performanceassesment\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.49.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\josue\anaconda3\envs\d206performanceassesment\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\users\josue\anaconda3\envs\d206performanceassesment\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (23.1)
Requirement already satisfied: pillow>=8 in c:\users\josue\anaconda3\envs\d206performanceassesment\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\josue\anaconda3\envs\d206performanceassesment\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\josue\anaconda3\envs\d206performanceassesment\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\josue\anaconda3\envs\d206performanceassesment\lib\site-packages (from pandas>=1.2->seaborn) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\josue\anaconda3\envs\d206performanceassesment\lib\site-packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: six>=1.5 in c:\users\josue\anaconda3\envs\d206performanceassesment\lib\site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
```

In [2]:
```python
# Standard imports
import numpy as np
import pandas as pd
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:
```python
# Load data set into Pandas dataframe
churn_df = pd.read_csv('C:\\Users\\josue\\Desktop\\WGU\\D206\\churn_raw_data.csv')
```

In [4]:
```python
#display churn dataframe to know it actually loaded up.
#A. Research Question how many people over the age of 62 have an active service and
churn_df
```

Out[4]:

	Unnamed: 0	CaseOrder	Customer_id	Interaction	City
0	1	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	Point Baker
1	2	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	West Branch
2	3	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	Yamhill
3	4	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	Del Mar
4	5	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	Needville
...	...	...	...	...	...
9995	9996	9996	M324793	45deb5a2-ae04-4518-bf0b-c82db8dbe4a4	Mount Holly
9996	9997	9997	D861732	6e96b921-0c09-4993-bbda-a1ac6411061a	Clarksville
9997	9998	9998	I243405	e8307ddf-9a01-4fff-bc59-4742e03fd24f	Mobeetie
9998	9999	9999	I641617	3775ccfc-0052-4107-81ae-9657f81ecdf3	Carrollton
9999	10000	10000	T38070	9de5fb6e-bd33-4995-aec8-f01d0172a499	Clarkesville

10000 rows × 52 columns

In [5]:

```
#list of dataframe columns
df = churn_df.columns
print(df)
```

```
Index(['Unnamed: 0', 'CaseOrder', 'Customer_id', 'Interaction', 'City',
       'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area',
       'Timezone', 'Job', 'Children', 'Age', 'Education', 'Employment',
       'Income', 'Marital', 'Gender', 'Churn', 'Outage_sec_perweek', 'Email',
       'Contacts', 'Yearly_equip_failure', 'Techie', 'Contract', 'Port_modem',
       'Tablet', 'InternetService', 'Phone', 'Multiple', 'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
       'StreamingMovies', 'PaperlessBilling', 'PaymentMethod', 'Tenure',
       'MonthlyCharge', 'Bandwidth_GB_Year', 'item1', 'item2', 'item3',
       'item4', 'item5', 'item6', 'item7', 'item8'],
      dtype='object')
```

In [6]:

```
#remove unnamed column and display first five records
df= churn_df.drop(churn_df.columns[0], axis =1)
df.head()
```

Out[6]:

	CaseOrder	Customer_id	Interaction	City	State	County	Zip	
0	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	Point Baker	AK	Prince of Wales-Hyder	99927	56
1	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	West Branch	MI	Ogemaw	48661	44
2	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	Yamhill	OR	Yamhill	97148	45
3	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	Del Mar	CA	San Diego	92014	32
4	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	Needville	TX	Fort Bend	77461	29

5 rows × 51 columns

In [7]:

```
#rename last 8 survey columns to more descriptive names rather than item
df.rename(columns = { 'item1':'Responses',
                      'item2':'Solutions',
                      'item3':'Replacements',
                      'item4':'Reliability',
                      'item5':'Options',
                      'item6':'Respectfulness',
                      'item7':'Courteous',
                      'item8':'Listening'},
          inplace=True)
```

In [8]:

```
#amount of records and columns of the dataset
df.shape
```

Out[8]: (10000, 51)

In [9]:

```
#describe Churn dataset columns and rows
df.describe()
```

Out[9]:

	CaseOrder	Zip	Lat	Lng	Population	Children
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	7505.000000
mean	5000.50000	49153.319600	38.757567	-90.782536	9756.562400	2.095936
std	2886.89568	27532.196108	5.437389	15.156142	14432.698671	2.154758
min	1.00000	601.000000	17.966120	-171.688150	0.000000	0.000000
25%	2500.75000	26292.500000	35.341828	-97.082812	738.000000	0.000000
50%	5000.50000	48869.500000	39.395800	-87.918800	2910.500000	1.000000
75%	7500.25000	71866.500000	42.106908	-80.088745	13168.000000	3.000000
max	10000.00000	99929.000000	70.640660	-65.667850	111850.000000	10.000000

8 rows × 23 columns

In [10]:

```
#remove variables from the dataset that do not help answer the question then displa
df_stats = df.drop(columns=['CaseOrder', 'Zip', 'Lat', 'Lng'])
df_stats.describe()
```

Out[10]:

	Population	Children	Age	Income	Outage_sec_perweek	
count	10000.000000	7505.000000	7525.000000	7510.000000	10000.000000	1000C
mean	9756.562400	2.095936	53.275748	39936.762226	11.452955	12
std	14432.698671	2.154758	20.753928	28358.469482	7.025921	3
min	0.000000	0.000000	18.000000	740.660000	-1.348571	1
25%	738.000000	0.000000	35.000000	19285.522500	8.054362	1C
50%	2910.500000	1.000000	53.000000	33186.785000	10.202896	12
75%	13168.000000	3.000000	71.000000	53472.395000	12.487644	14
max	111850.000000	10.000000	89.000000	258900.700000	47.049280	23

In [11]:

```
#calculate churn rate
df.Churn.value_counts() / len(df)
```

Out[11]:

```
Churn
No     0.735
Yes    0.265
Name: count, dtype: float64
```

In [12]:

```
#review data types ( numerical => "int64", "float64", while categorical data is =>
df.dtypes
```

Out[12]:
```
CaseOrder                 int64
Customer_id              object
Interaction              object
City                     object
State                    object
County                   object
Zip                       int64
Lat                     float64
Lng                     float64
Population                int64
Area                     object
Timezone                 object
Job                      object
Children                float64
Age                     float64
Education                object
Employment               object
Income                  float64
Marital                  object
Gender                   object
Churn                    object
Outage_sec_perweek      float64
Email                     int64
Contacts                  int64
Yearly_equip_failure      int64
Techie                   object
Contract                 object
Port_modem               object
Tablet                   object
InternetService          object
Phone                    object
Multiple                 object
OnlineSecurity           object
OnlineBackup             object
DeviceProtection         object
TechSupport              object
StreamingTV              object
StreamingMovies          object
PaperlessBilling         object
PaymentMethod            object
Tenure                  float64
MonthlyCharge           float64
Bandwidth_GB_Year       float64
Responses                 int64
Solutions                 int64
Replacements              int64
Reliability               int64
Options                   int64
Respectfulness            int64
Courteous                 int64
Listening                 int64
dtype: object
```

In [13]:
```
# Re-validate column data types and missing values
df.columns.to_series().groupby(df.dtypes).groups
```

```
Out[13]: {int64: ['CaseOrder', 'Zip', 'Population', 'Email', 'Contacts', 'Yearly_equip_failure', 'Responses', 'Solutions', 'Replacements', 'Reliability', 'Options', 'Respectfulness', 'Courteous', 'Listening'], float64: ['Lat', 'Lng', 'Children', 'Age', 'Income', 'Outage_sec_perweek', 'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year'], object: ['Customer_id', 'Interaction', 'City', 'State', 'County', 'Area', 'Timezone', 'Job', 'Education', 'Employment', 'Marital', 'Gender', 'Churn', 'Techie', 'Contract', 'Port_modem', 'Tablet', 'InternetService', 'Phone', 'Multiple', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'PaperlessBilling', 'PaymentMethod']}
```

```
In [14]: #find missing values
df.isnull()
```

Out[14]:

	CaseOrder	Customer_id	Interaction	City	State	County	Zip	Lat	Lng	Popula
0	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	
9995	False	False	False	False	False	False	False	False	False	
9996	False	False	False	False	False	False	False	False	False	
9997	False	False	False	False	False	False	False	False	False	
9998	False	False	False	False	False	False	False	False	False	
9999	False	False	False	False	False	False	False	False	False	

10000 rows × 51 columns

```
In [15]: #access only rows from dataframe containing missing values
df.isnull().any(axis=1)
```

```
Out[15]: 0        True
1       False
2        True
3       False
4       False
        ...
9995     True
9996     True
9997     True
9998    False
9999     True
Length: 10000, dtype: bool
```

```
In [16]: #display specific columns with no value i.e NA
df.isna().any()
```

```
Out[16]: CaseOrder               False
         Customer_id             False
         Interaction             False
         City                    False
         State                   False
         County                  False
         Zip                     False
         Lat                     False
         Lng                     False
         Population              False
         Area                    False
         Timezone                False
         Job                     False
         Children                 True
         Age                      True
         Education               False
         Employment              False
         Income                   True
         Marital                 False
         Gender                  False
         Churn                   False
         Outage_sec_perweek      False
         Email                   False
         Contacts                False
         Yearly_equip_failure    False
         Techie                   True
         Contract                False
         Port_modem              False
         Tablet                  False
         InternetService          True
         Phone                    True
         Multiple                False
         OnlineSecurity          False
         OnlineBackup            False
         DeviceProtection        False
         TechSupport              True
         StreamingTV             False
         StreamingMovies         False
         PaperlessBilling        False
         PaymentMethod           False
         Tenure                   True
         MonthlyCharge           False
         Bandwidth_GB_Year        True
         Responses               False
         Solutions               False
         Replacements            False
         Reliability             False
         Options                 False
         Respectfulness          False
         Courteous               False
         Listening               False
         dtype: bool
```

In [17]:
```python
# store null values in a variable to get total counts
d_nulls = df.isnull().sum()
print(d_nulls)
```

```
CaseOrder                 0
Customer_id               0
Interaction               0
City                      0
State                     0
County                    0
Zip                       0
Lat                       0
Lng                       0
Population                0
Area                      0
Timezone                  0
Job                       0
Children               2495
Age                    2475
Education                 0
Employment                0
Income                 2490
Marital                   0
Gender                    0
Churn                     0
Outage_sec_perweek        0
Email                     0
Contacts                  0
Yearly_equip_failure      0
Techie                 2477
Contract                  0
Port_modem                0
Tablet                    0
InternetService        2129
Phone                  1026
Multiple                  0
OnlineSecurity            0
OnlineBackup              0
DeviceProtection          0
TechSupport             991
StreamingTV               0
StreamingMovies           0
PaperlessBilling          0
PaymentMethod             0
Tenure                  931
MonthlyCharge             0
Bandwidth_GB_Year      1021
Responses                 0
Solutions                 0
Replacements              0
Reliability               0
Options                   0
Respectfulness            0
Courteous                 0
Listening                 0
dtype: int64
```

In [18]:
```
# Store rows with missing values in a new variable
rows_with_missing_values = df.isnull().any(axis=1)
df[rows_with_missing_values]
```

Out[18]:

	CaseOrder	Customer_id	Interaction	City	State	Count
0	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	Point Baker	AK	Prince c Wales-Hyde
2	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	Yamhill	OR	Yamhi
5	6	W303516	2b451d12-6c2b-4cea-a295-ba1d6bced078	Fort Valley	GA	Peac
6	7	U335188	6630d501-838c-4be4-a59c-6f58c814ed6a	Pioneer	TN	Sco
7	8	V538685	70ddaa89-b726-49dc-9022-2d655e4c7936	Oklahoma City	OK	Oklahom
...	...	...	...	...	...	
9994	9995	P175475	c60df12b-a50b-4397-ae57-98381a0d3960	West Kill	NY	Green
9995	9996	M324793	45deb5a2-ae04-4518-bf0b-c82db8dbe4a4	Mount Holly	VT	Rutlan
9996	9997	D861732	6e96b921-0c09-4993-bbda-a1ac6411061a	Clarksville	TN	Montgomer
9997	9998	I243405	e8307ddf-9a01-4fff-bc59-4742e03fd24f	Mobeetie	TX	Wheele
9999	10000	T38070	9de5fb6e-bd33-4995-aec8-f01d0172a499	Clarkesville	GA	Habershar

8316 rows × 51 columns

In [19]:
```
#Examine columns for miss spelled words in categorical variables using unquie() met
df['Employment'].unique()
```

Out[19]:
```
array(['Part Time', 'Retired', 'Student', 'Full Time', 'Unemployed'],
      dtype=object)
```

In [20]:
```
df['Area'].unique()
```

Out[20]:
```
array(['Urban', 'Suburban', 'Rural'], dtype=object)
```

In [21]:
```
df['Timezone'].unique()
```

Out[21]:
```
array(['America/Sitka', 'America/Detroit', 'America/Los_Angeles',
       'America/Chicago', 'America/New_York', 'America/Puerto_Rico',
       'America/Denver', 'America/Menominee', 'America/Phoenix',
       'America/Indiana/Indianapolis', 'America/Boise',
       'America/Kentucky/Louisville', 'Pacific/Honolulu',
       'America/Indiana/Petersburg', 'America/Nome', 'America/Anchorage',
       'America/Indiana/Knox', 'America/Juneau', 'America/Toronto',
       'America/Indiana/Winamac', 'America/Indiana/Vincennes',
       'America/North_Dakota/New_Salem', 'America/Indiana/Tell_City',
       'America/Indiana/Marengo', 'America/Ojinaga'], dtype=object)
```

In [22]:
```
#get total count of how many unquie items there are
len(df['Job'].unique())
```

Out[22]:
```
639
```

In [23]:
```
df['Children'].unique()
```

Out[23]:
```
array([nan,  1.,  4.,  0.,  3.,  2.,  7.,  5.,  9.,  6., 10.,  8.])
```

In [24]:
```
#find the unique amount of ages  then get a count for how many differnet age's ther

df['Age'].unique()
```

Out[24]:
```
array([68., 27., 50., 48., 83., nan, 49., 86., 23., 56., 30., 39., 63.,
       60., 61., 52., 75., 77., 47., 70., 69., 45., 40., 82., 26., 25.,
       66., 72., 41., 44., 43., 84., 59., 31., 51., 58., 73., 33., 42.,
       81., 87., 54., 67., 46., 24., 20., 71., 32., 29., 80., 53., 79.,
       65., 35., 34., 74., 55., 76., 57., 38., 78., 19., 36., 88., 62.,
       37., 28., 22., 85., 89., 18., 21., 64.])
```

In [25]:
```
age_range = df['Age'].unique()
print(sorted(age_range))
```

```
[23.0, 25.0, 26.0, 27.0, 30.0, 31.0, 39.0, 40.0, 41.0, 43.0, 44.0, 45.0, 47.0, 48.0,
49.0, 50.0, 51.0, 52.0, 59.0, 61.0, 68.0, 83.0, nan, 18.0, 19.0, 20.0, 21.0, 22.0, 2
4.0, 28.0, 29.0, 32.0, 33.0, 34.0, 35.0, 36.0, 37.0, 38.0, 42.0, 46.0, 53.0, 54.0, 5
5.0, 56.0, 57.0, 58.0, 60.0, 62.0, 63.0, 64.0, 65.0, 66.0, 67.0, 69.0, 70.0, 71.0, 7
2.0, 73.0, 74.0, 75.0, 76.0, 77.0, 78.0, 79.0, 80.0, 81.0, 82.0, 84.0, 85.0, 86.0, 8
7.0, 88.0, 89.0]
```

In [26]:
```
#finding more unquie values in other columns
df['Education'].unique()
```

Out[26]:
```
array(["Master's Degree", 'Regular High School Diploma',
       'Doctorate Degree', 'No Schooling Completed', "Associate's Degree",
       "Bachelor's Degree", 'Some College, Less than 1 Year',
       'GED or Alternative Credential',
       'Some College, 1 or More Years, No Degree',
       '9th Grade to 12th Grade, No Diploma',
       'Nursery School to 8th Grade', 'Professional School Degree'],
      dtype=object)
```

In [27]:
```
df['Employment'].unique()
```

Out[27]:
```
array(['Part Time', 'Retired', 'Student', 'Full Time', 'Unemployed'],
      dtype=object)
```

In [28]:
```
df['Marital'].unique()
```

Out[28]:
```
array(['Widowed', 'Married', 'Separated', 'Never Married', 'Divorced'],
      dtype=object)
```

In [29]:
```
df['Gender'].unique()
```

Out[29]:
```
array(['Male', 'Female', 'Prefer not to answer'], dtype=object)
```

In [30]:
```
df['Contract'].unique()
```

Out[30]:
```
array(['One year', 'Month-to-month', 'Two Year'], dtype=object)
```

In [31]:
```
df['PaymentMethod'].unique()
```

Out[31]:
```
array(['Credit Card (automatic)', 'Bank Transfer(automatic)',
       'Mailed Check', 'Electronic Check'], dtype=object)
```

In [32]:
```
# Display any duplicate rows in the dataframe.
data_duplicates = df.loc[df.duplicated()]
print(data_duplicates)
```

```
Empty DataFrame
Columns: [CaseOrder, Customer_id, Interaction, City, State, County, Zip, Lat, Lng, P
opulation, Area, Timezone, Job, Children, Age, Education, Employment, Income, Marita
l, Gender, Churn, Outage_sec_perweek, Email, Contacts, Yearly_equip_failure, Techie,
Contract, Port_modem, Tablet, InternetService, Phone, Multiple, OnlineSecurity, Onli
neBackup, DeviceProtection, TechSupport, StreamingTV, StreamingMovies, PaperlessBill
ing, PaymentMethod, Tenure, MonthlyCharge, Bandwidth_GB_Year, Responses, Solutions,
Replacements, Reliability, Options, Respectfulness, Courteous, Listening]
Index: []

[0 rows x 51 columns]
```

In [33]:
```
data_nulls = df_stats.isnull().sum()
print(data_nulls)
```

```
Customer_id                0
Interaction                0
City                       0
State                      0
County                     0
Population                 0
Area                       0
Timezone                   0
Job                        0
Children                2495
Age                     2475
Education                  0
Employment                 0
Income                  2490
Marital                    0
Gender                     0
Churn                      0
Outage_sec_perweek         0
Email                      0
Contacts                   0
Yearly_equip_failure       0
Techie                  2477
Contract                   0
Port_modem                 0
Tablet                     0
InternetService         2129
Phone                   1026
Multiple                   0
OnlineSecurity             0
OnlineBackup               0
DeviceProtection           0
TechSupport              991
StreamingTV                0
StreamingMovies            0
PaperlessBilling           0
PaymentMethod              0
Tenure                   931
MonthlyCharge              0
Bandwidth_GB_Year       1021
Responses                  0
Solutions                  0
Replacements               0
Reliability                0
Options                    0
Respectfulness             0
Courteous                  0
Listening                  0
dtype: int64
```

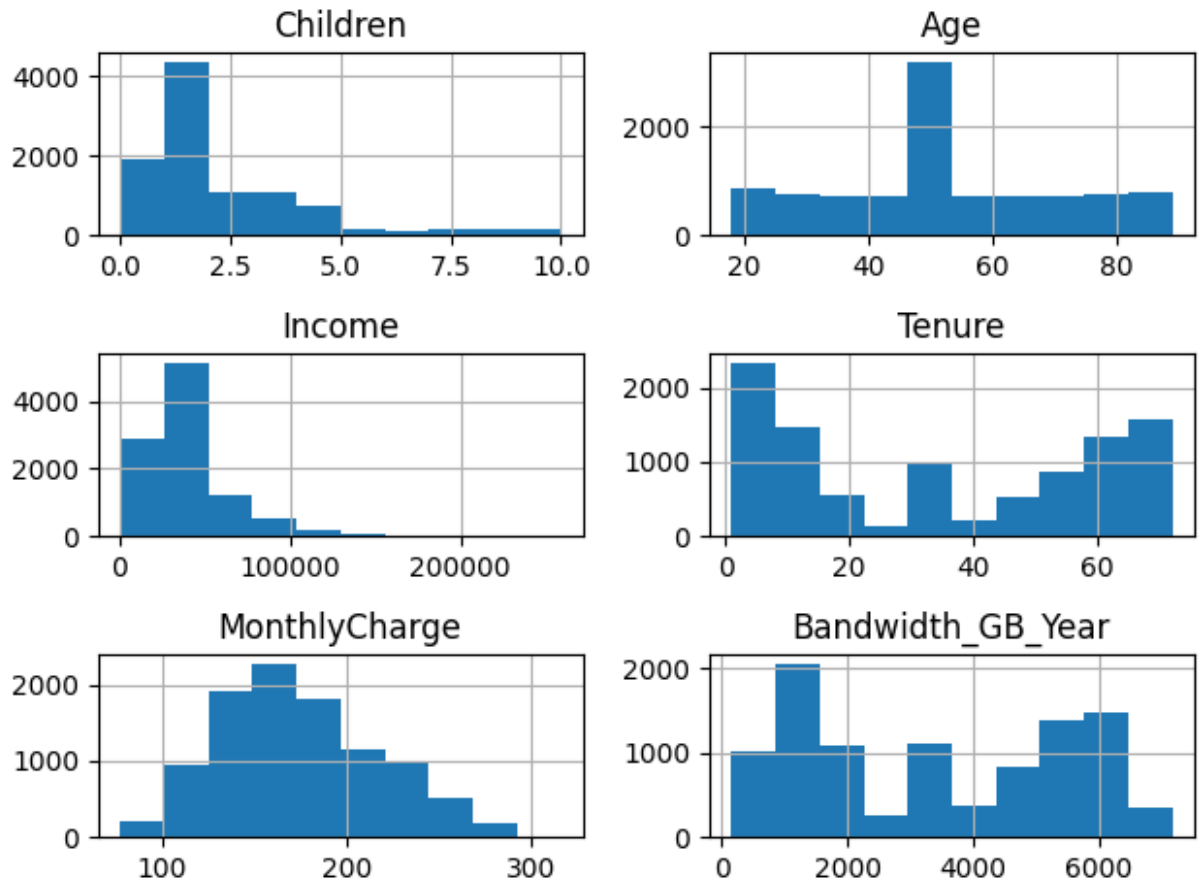
In [34]:

```
#impute missing fields for variables with median or mean
df_stats['Children'] = df['Children'].fillna(df['Children'].median())
df_stats['Age'] = df['Age'].fillna(df['Age'].median())
df_stats['Income'] = df['Income'].fillna(df['Income'].median())
df_stats['Tenure'] = df['Tenure'].fillna(df['Tenure'].median())
df_stats['Bandwidth_GB_Year'] = df['Bandwidth_GB_Year'].fillna(df['Bandwidth_GB_Yea
```

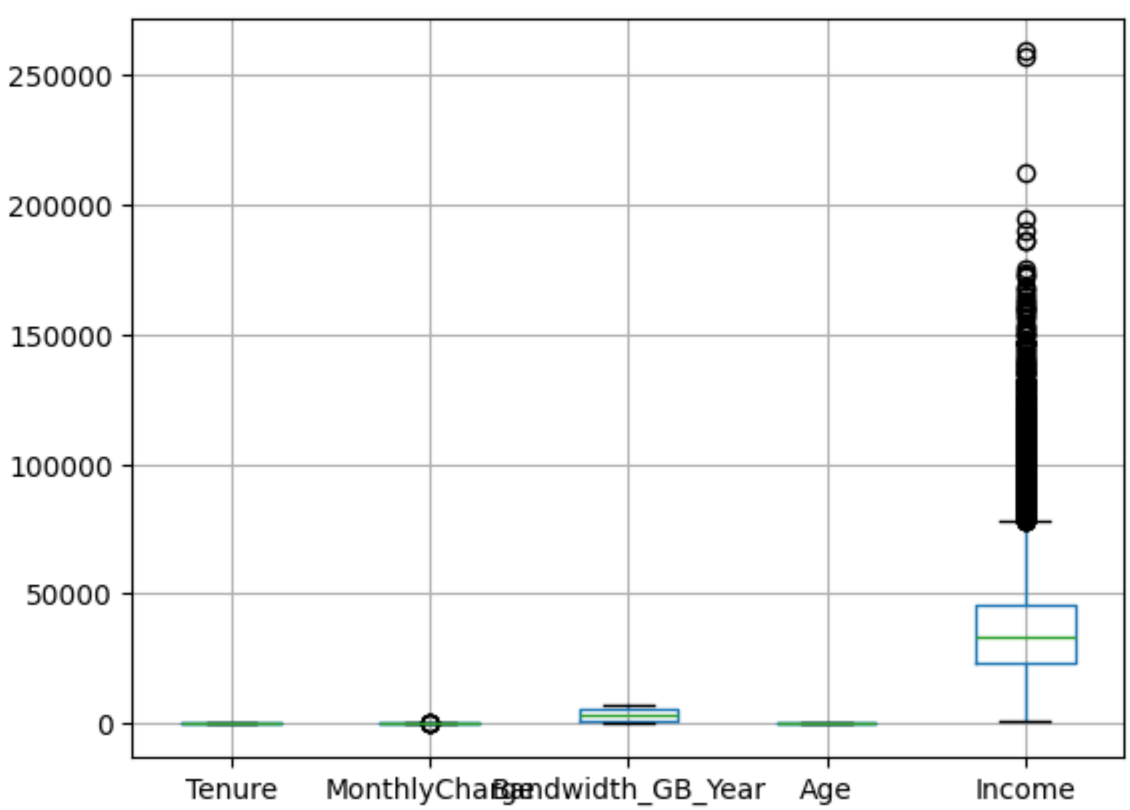
In [35]:
```python
data_nulls = df_stats.isnull().sum()
print(data_nulls)
```

```
Customer_id               0
Interaction               0
City                      0
State                     0
County                    0
Population                0
Area                      0
Timezone                  0
Job                       0
Children                  0
Age                       0
Education                 0
Employment                0
Income                    0
Marital                   0
Gender                    0
Churn                     0
Outage_sec_perweek        0
Email                     0
Contacts                  0
Yearly_equip_failure      0
Techie                 2477
Contract                  0
Port_modem                0
Tablet                    0
InternetService        2129
Phone                  1026
Multiple                  0
OnlineSecurity            0
OnlineBackup              0
DeviceProtection          0
TechSupport             991
StreamingTV               0
StreamingMovies           0
PaperlessBilling          0
PaymentMethod             0
Tenure                    0
MonthlyCharge             0
Bandwidth_GB_Year         0
Responses                 0
Solutions                 0
Replacements              0
Reliability               0
Options                   0
Respectfulness            0
Courteous                 0
Listening                 0
dtype: int64
```

In [36]:
```python
# Create histograms of important variables
df_stats[['Children', 'Age', 'Income', 'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Yea
plt.savefig('churn_pyplot.jpg')
plt.tight_layout()
```

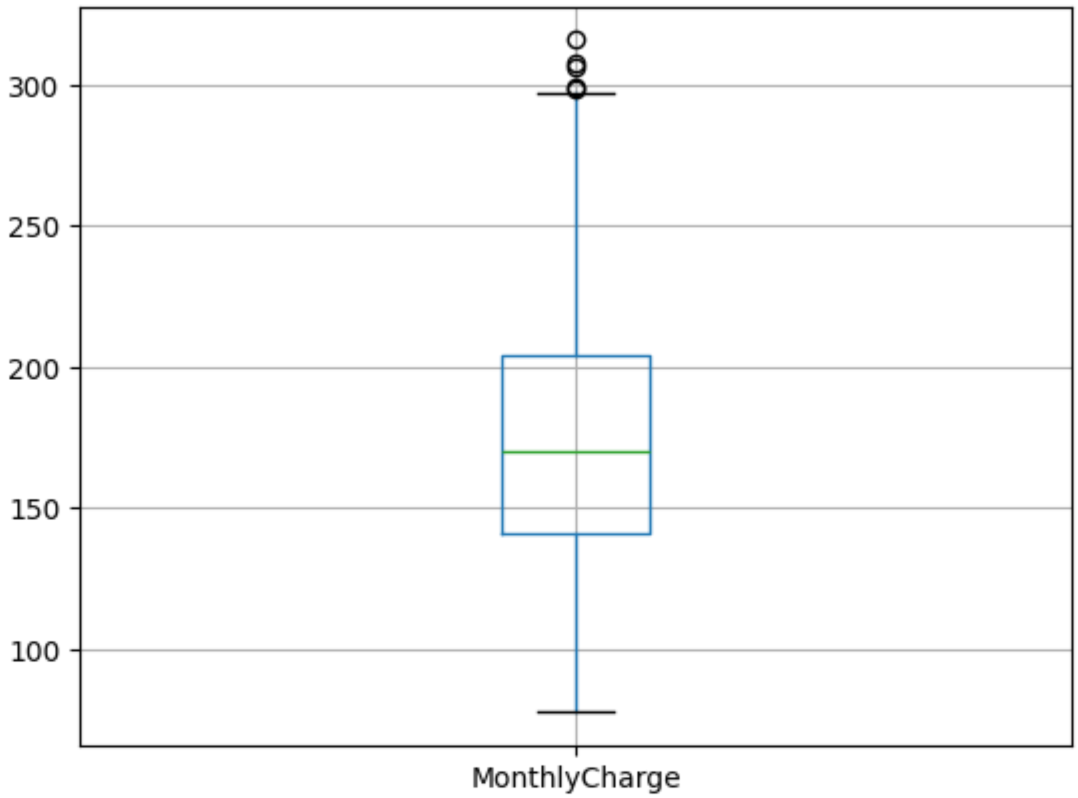


In [77]:
```
# Create a boxplot of user duration, payment, useage,income & age variables
df_stats.boxplot(['Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year','Age','Income'])
plt.savefig('churn_boxplots.jpg')
```



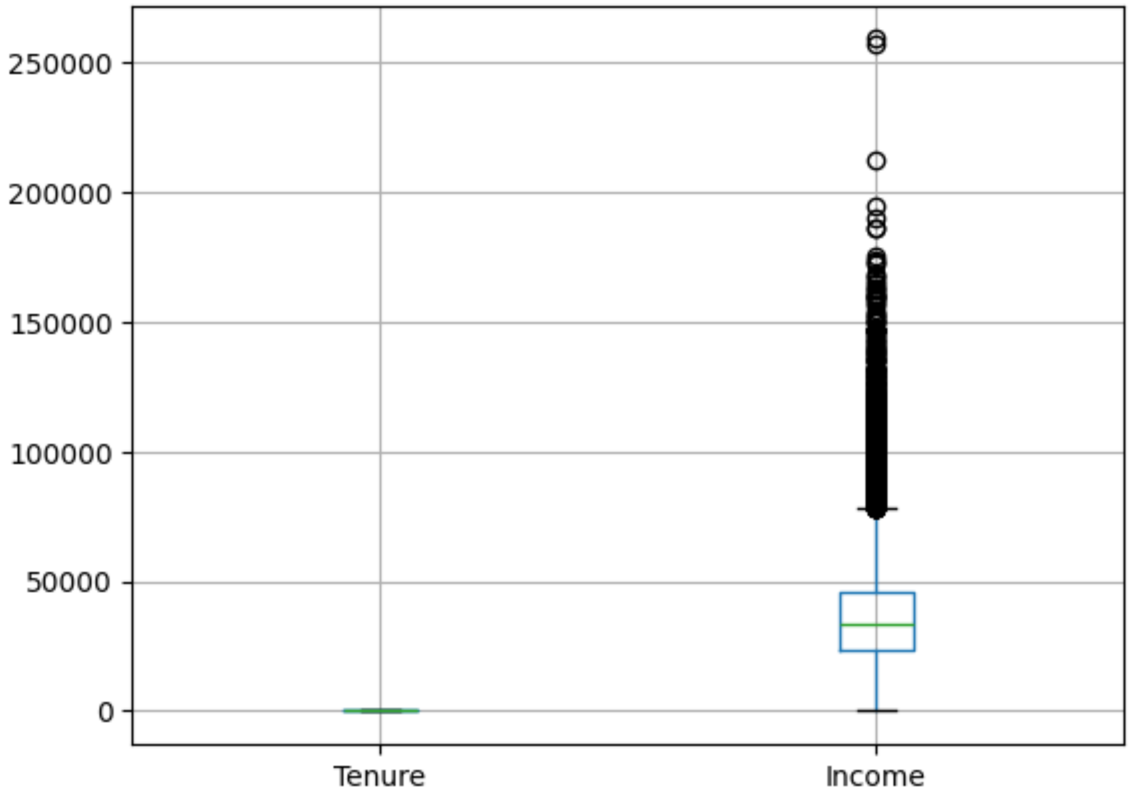
In [38]:
```
#isolate monlthy charge  where we can take a better look at the outliers
df_stats.boxplot(['MonthlyCharge'])
```

Out[38]: <Axes: >



In [55]:
```
# lets take a closer look at income to see the outliers here
df_stats.boxplot(['Tenure','Income'])
```

Out[55]: <Axes: >



In [56]:
```
#extract Clean dataset
df_stats.to_csv('churn_cleaned.csv')
```

In [59]:
```
#reload cleaned data & remove all variable except user services payment info and su
churn_user = pd.read_csv('churn_cleaned.csv')
```

In [60]:
```
#remove all but the last 11 service related variables for the PCA
data = churn_user.loc[:, 'Tenure':'Listening']
data.head()
```

Out[60]:

	Tenure	MonthlyCharge	Bandwidth_GB_Year	Responses	Solutions	Replacements	Re
0	6.795513	171.449762	904.536110	5	5	5	
1	1.156681	242.948015	800.982766	3	4	3	
2	15.754144	159.440398	2054.706961	4	4	2	
3	17.087227	120.249493	2164.579412	4	4	4	
4	1.670972	150.761216	271.493436	4	4	4	

In [61]:
```
#Import Scikit Learn PCA application
from sklearn.decomposition import PCA
```

In [62]:
```
#Normalize the data
churn_normalized = (data - data.mean()) / data.std()
```

In [63]:
```
#Select number of components to extract
pca = PCA(n_components = data.shape[1])
```

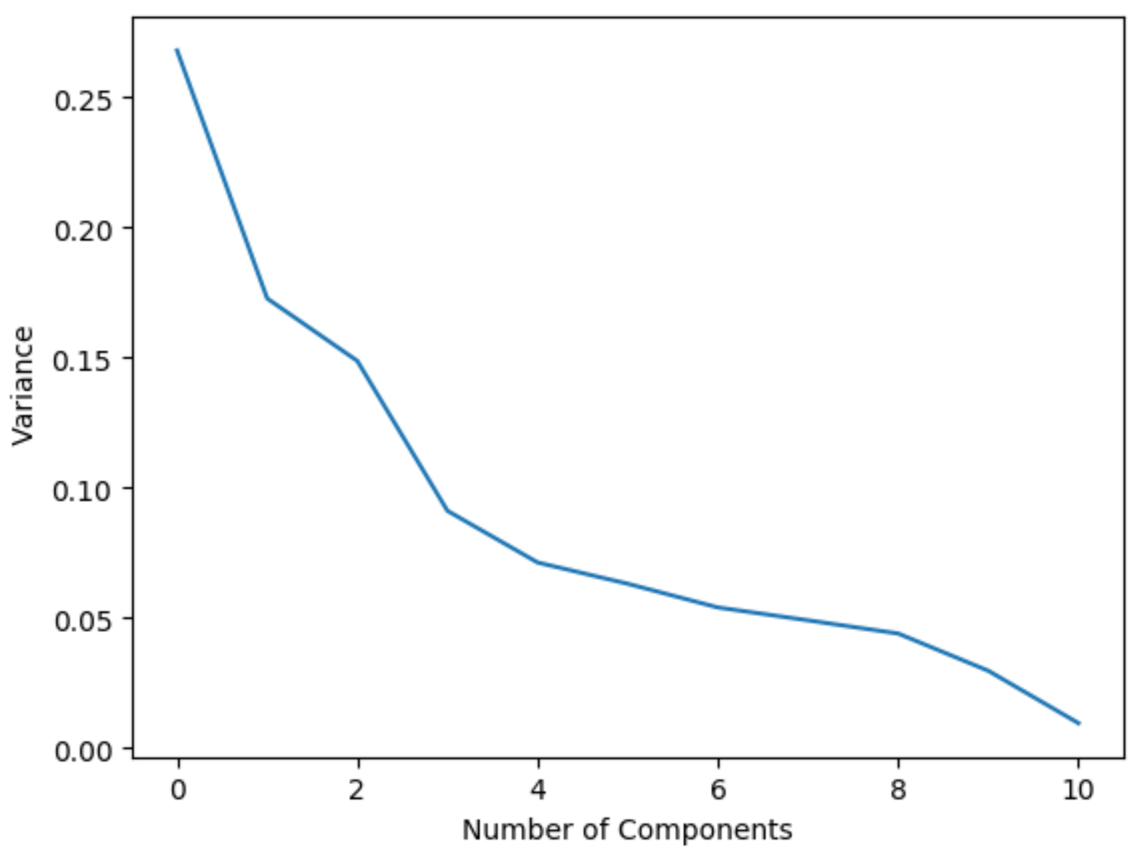
In [65]:
```
#Create a list of PCA names
churn_numeric = data[['Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year', 'Responses',
                      'Solutions', 'Replacements', 'Reliability', 'Options',
                      'Respectfulness', 'Courteous', 'Listening']]
pcs_names = []
for i, col in enumerate(churn_numeric.columns):
    pcs_names.append('PC' + str(i + 1))
print(pcs_names)
```

```
['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10', 'PC11']
```

In [66]:
```
#Call PCA application & convert the dataset of 11 variables into a dataset of 11 co
pca.fit(churn_normalized)
churn_pca = pd.DataFrame(pca.transform(churn_normalized),
                        columns = pcs_names)
```

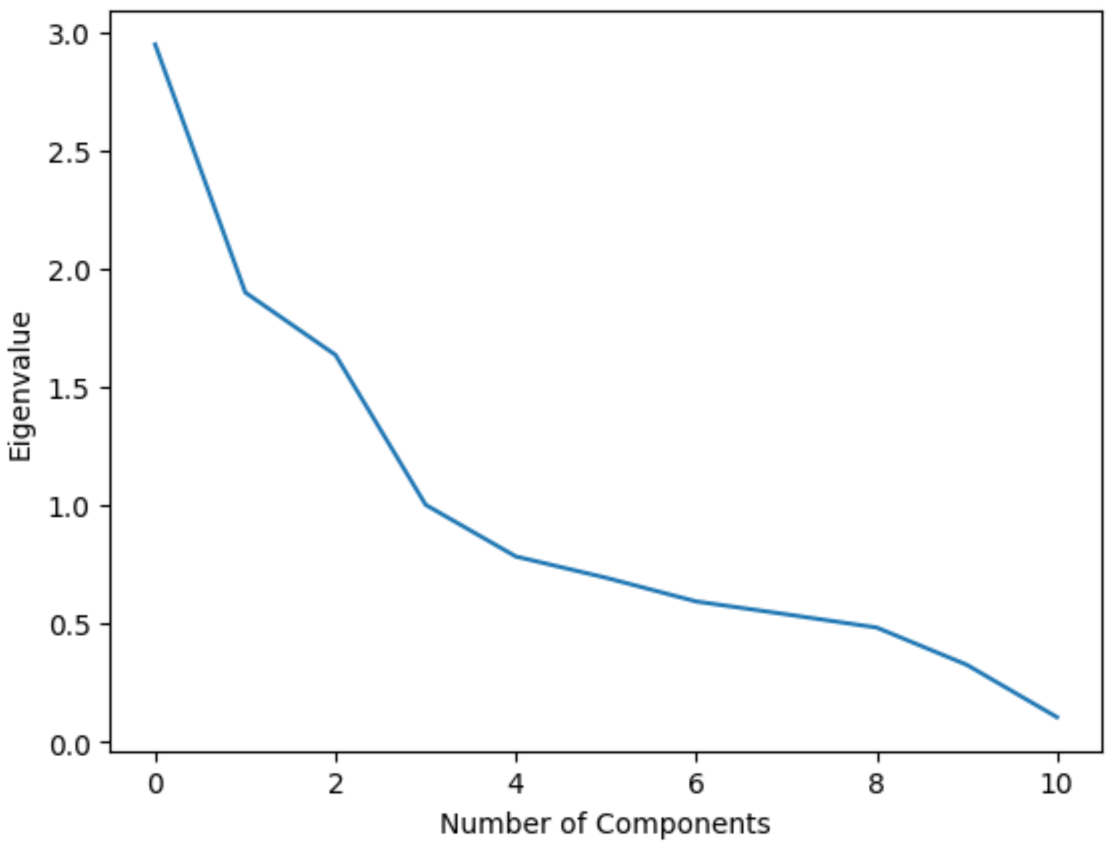
In [67]:
```
#For a scree plot import matplotlib & seaborn libraries
import matplotlib.pyplot as plt
import seaborn as sns
```

In [69]:
```
#Run the scree plot
plt.plot(pca.explained_variance_ratio_)
plt.xlabel('Number of Components')
plt.ylabel(' Variance')
plt.show();
```



In [70]:
```
#Extract the eigenvalues
cov_matrix = np.dot(churn_normalized.T, churn_normalized) / data.shape[0]
eigenvalues = [np.dot(eigenvector.T, np.dot(cov_matrix, eigenvector)) for eigenvect
```

In [71]:
```
#Plot the eigenvalues
plt.plot(eigenvalues)
plt.xlabel('Number of Components')
plt.ylabel('Eigenvalue')
plt.show();
```



In [72]:
```
for pc, var in zip(pcs_names, np.cumsum(pca.explained_variance_ratio_)):
    print(pc, var)
```

```
PC1 0.2679266062396318
PC2 0.44053732880229535
PC3 0.589144452224861
PC4 0.6801579842425111
PC5 0.751321705437251
PC6 0.814311709410132
PC7 0.8681970531672304
PC8 0.9171387362588809
PC9 0.9610122820002168
PC10 0.9905736363144829
PC11 0.9999999999999998
```

In [73]:
```
# we see that 86% of variance is explained by 7 components
# Create a rotation
rotation = pd.DataFrame(pca.components_.T, columns = pcs_names, index = churn_numer
print(rotation)
```

```
                      PC1       PC2       PC3       PC4       PC5       PC6  \
Tenure          -0.010403  0.701838 -0.072209 -0.063594  0.005683 -0.011155
MonthlyCharge    0.000317  0.041147 -0.014151  0.996995 -0.022136  0.015231
Bandwidth_GB_Year -0.012166  0.703079 -0.074222  0.004399  0.009590  0.003466
Responses        0.458932  0.031325  0.281154  0.018568 -0.070233 -0.119149
Solutions        0.434134  0.042559  0.282404  0.007508 -0.106632 -0.169752
Replacements     0.400639  0.034665  0.281118 -0.019631 -0.173742 -0.255336
Reliability      0.145799 -0.050367 -0.567815 -0.010310 -0.171334 -0.483328
Options         -0.175633  0.066334  0.587335 -0.000047  0.135949  0.060124
Respectfulness   0.405207 -0.012680 -0.183447  0.004596 -0.062342  0.064609
Courteous        0.358342 -0.003886 -0.181697 -0.027959 -0.182406  0.806166
Listening        0.308925 -0.017396 -0.131173  0.015574  0.931612 -0.011133

                      PC7       PC8       PC9      PC10      PC11
Tenure           0.007419 -0.011527  0.006935  0.003286 -0.705445
MonthlyCharge   -0.018038 -0.004316  0.023690 -0.013785 -0.047865
Bandwidth_GB_Year  0.003701 -0.002364 -0.008068  0.008529  0.706925
Responses       -0.045963  0.025431 -0.240574  0.793237 -0.004306
Solutions       -0.065414  0.074400 -0.592131 -0.573832 -0.002217
Replacements    -0.146887 -0.396333  0.673088 -0.177665  0.014933
Reliability     -0.443353  0.431528  0.087207  0.018301  0.002283
Options         -0.209767  0.693861  0.265474 -0.042012 -0.002514
Respectfulness   0.757954  0.402835  0.230319 -0.063972  0.001604
Courteous       -0.379136  0.067889  0.067293 -0.040946 -0.006875
Listening       -0.113297 -0.045132  0.046107 -0.042251 -0.002357
```

In [74]:

```python
# Output loadings for components
loadings = pd.DataFrame(pca.components_.T,
                        columns = pcs_names,
                        index = data.columns)
loadings
```

Out[74]:

	PC1	PC2	PC3	PC4	PC5	PC6	P
Tenure	-0.010403	0.701838	-0.072209	-0.063594	0.005683	-0.011155	0.007
MonthlyCharge	0.000317	0.041147	-0.014151	0.996995	-0.022136	0.015231	-0.018
Bandwidth_GB_Year	-0.012166	0.703079	-0.074222	0.004399	0.009590	0.003466	0.003
Responses	0.458932	0.031325	0.281154	0.018568	-0.070233	-0.119149	-0.045
Solutions	0.434134	0.042559	0.282404	0.007508	-0.106632	-0.169752	-0.065
Replacements	0.400639	0.034665	0.281118	-0.019631	-0.173742	-0.255336	-0.146
Reliability	0.145799	-0.050367	-0.567815	-0.010310	-0.171334	-0.483328	-0.443
Options	-0.175633	0.066334	0.587335	-0.000047	0.135949	0.060124	-0.209
Respectfulness	0.405207	-0.012680	-0.183447	0.004596	-0.062342	0.064609	0.757
Courteous	0.358342	-0.003886	-0.181697	-0.027959	-0.182406	0.806166	-0.379
Listening	0.308925	-0.017396	-0.131173	0.015574	0.931612	-0.011133	-0.113

In [75]:

```python
#reduced dataset & print 3 components
```

```python
churn_reduced = churn_pca.iloc[ : , 0:3]
print(churn_reduced)
```

```
           PC1       PC2       PC3
0     1.923875 -1.421955  1.903125
1    -0.199798 -1.706801  0.538766
2    -0.667923 -0.985940  0.227390
3     0.046465 -0.730628  2.282040
4     1.326741 -1.924880  0.825729
...        ...       ...       ...
9995 -2.097964  1.961837  0.104147
9996  1.917485  1.645946  0.611009
9997  1.431918  0.323573  0.028288
9998  2.011460  2.187756 -0.079864
9999 -2.266364  1.591986 -0.819973

[10000 rows x 3 columns]
```

In [ ]: