In [1]:
```
#Install packages
!pip install pandas
!pip install numpy
!pip install scipy
!pip install sklearn
!pip install matplotlib
```

```
Requirement already satisfied: pandas in c:\users\coope\anaconda3\lib\site-packages (1.
2.4)
Requirement already satisfied: numpy>=1.16.5 in c:\users\coope\anaconda3\lib\site-packag
es (from pandas) (1.20.1)
Requirement already satisfied: pytz>=2017.3 in c:\users\coope\anaconda3\lib\site-package
s (from pandas) (2021.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\coope\anaconda3\lib\si
te-packages (from pandas) (2.8.1)
Requirement already satisfied: six>=1.5 in c:\users\coope\anaconda3\lib\site-packages (f
rom python-dateutil>=2.7.3->pandas) (1.15.0)
Requirement already satisfied: numpy in c:\users\coope\anaconda3\lib\site-packages (1.2
0.1)
Requirement already satisfied: scipy in c:\users\coope\anaconda3\lib\site-packages (1.6.
2)
Requirement already satisfied: numpy<1.23.0,>=1.16.5 in c:\users\coope\anaconda3\lib\sit
e-packages (from scipy) (1.20.1)
Requirement already satisfied: sklearn in c:\users\coope\anaconda3\lib\site-packages (0.
0)
Requirement already satisfied: scikit-learn in c:\users\coope\anaconda3\lib\site-package
s (from sklearn) (0.24.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\coope\anaconda3\lib\site
-packages (from scikit-learn->sklearn) (2.1.0)
Requirement already satisfied: scipy>=0.19.1 in c:\users\coope\anaconda3\lib\site-packag
es (from scikit-learn->sklearn) (1.6.2)
Requirement already satisfied: joblib>=0.11 in c:\users\coope\anaconda3\lib\site-package
s (from scikit-learn->sklearn) (1.0.1)
Requirement already satisfied: numpy>=1.13.3 in c:\users\coope\anaconda3\lib\site-packag
es (from scikit-learn->sklearn) (1.20.1)
Requirement already satisfied: matplotlib in c:\users\coope\anaconda3\lib\site-packages
(3.3.4)
Requirement already satisfied: pillow>=6.2.0 in c:\users\coope\anaconda3\lib\site-packag
es (from matplotlib) (8.2.0)
Requirement already satisfied: numpy>=1.15 in c:\users\coope\anaconda3\lib\site-packages
(from matplotlib) (1.20.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\coop
e\anaconda3\lib\site-packages (from matplotlib) (2.4.7)
Requirement already satisfied: cycler>=0.10 in c:\users\coope\anaconda3\lib\site-package
s (from matplotlib) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\coope\anaconda3\lib\site-pa
ckages (from matplotlib) (1.3.1)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\coope\anaconda3\lib\site
-packages (from matplotlib) (2.8.1)
Requirement already satisfied: six in c:\users\coope\anaconda3\lib\site-packages (from c
ycler>=0.10->matplotlib) (1.15.0)
```

In [2]:
```
# 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]:
```
#Load data into Pandas
churn_cd = pd.read_csv('churn_raw_data.csv')
```

In [4]:
```
# Display Data
churn_cd
```

Out[4]:

	Unnamed: 0	CaseOrder	Customer_id	Interaction	City	State	County	Zip	I
0	1	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	Point Baker	AK	Prince of Wales-Hyder	99927	56.251
1	2	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	West Branch	MI	Ogemaw	48661	44.328
2	3	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	Yamhill	OR	Yamhill	97148	45.355
3	4	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	Del Mar	CA	San Diego	92014	32.966
4	5	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	Needville	TX	Fort Bend	77461	29.380
...	...	...	...	...	...	...	...	...	
9995	9996	9996	M324793	45deb5a2-ae04-4518-bf0b-c82db8dbe4a4	Mount Holly	VT	Rutland	5758	43.433
9996	9997	9997	D861732	6e96b921-0c09-4993-bbda-a1ac6411061a	Clarksville	TN	Montgomery	37042	36.569
9997	9998	9998	I243405	e8307ddf-9a01-4fff-bc59-4742e03fd24f	Mobeetie	TX	Wheeler	79061	35.520
9998	9999	9999	I641617	3775ccfc-0052-4107-81ae-9657f81ecdf3	Carrollton	GA	Carroll	30117	33.580
9999	10000	10000	T38070	9de5fb6e-bd33-4995-aec8-f01d0172a499	Clarkesville	GA	Habersham	30523	34.707

10000 rows × 52 columns

In [5]:
```
#Columns
cd = churn_cd.columns
print(cd)
```

```
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 head
cd = churn_cd.drop(churn_cd.columns[0], axis = 1)
cd.head()
```

Out[6]:

	CaseOrder	Customer_id	Interaction	City	State	County	Zip	Lat	Lng	Popu
0	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	-133.37571	
1	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	West Branch	MI	Ogemaw	48661	44.32893	-84.24080	
2	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	Yamhill	OR	Yamhill	97148	45.35589	-123.24657	
3	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	Del Mar	CA	San Diego	92014	32.96687	-117.24798	
4	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	Needville	TX	Fort Bend	77461	29.38012	-95.80673	

5 rows × 51 columns

In [7]:
```
#Rename columns labeled as "Item" to responses/variable names
cd.rename(columns = {'item1':'Response',
                     'item2':'Fixes',
                     'item3':'Replacement',
                     'item4':'Reliability',
```

```
                    'item5':'Options',
                    'item6':'Respectful',
                    'item7':'Courteous',
                    'item8':'Active_Listening'}, inplace=True)
```

In [8]:
```
#Display head to see new column names
cd.head()
```

Out[8]:

	CaseOrder	Customer_id	Interaction	City	State	County	Zip	Lat	Lng	Popu
0	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	-133.37571	
1	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	West Branch	MI	Ogemaw	48661	44.32893	-84.24080	
2	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	Yamhill	OR	Yamhill	97148	45.35589	-123.24657	
3	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	Del Mar	CA	San Diego	92014	32.96687	-117.24798	
4	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	Needville	TX	Fort Bend	77461	29.38012	-95.80673	

5 rows × 51 columns

In [9]:
```
#Identify number of rows and columns
cd.shape
```

Out[9]:
```
(10000, 51)
```

In [10]:
```
#Describe dataset
cd.describe()
```

Out[10]:

	CaseOrder	Zip	Lat	Lng	Population	Children	Age
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	7505.000000	7525.000000
mean	5000.50000	49153.319600	38.757567	-90.782536	9756.562400	2.095936	53.275748
std	2886.89568	27532.196108	5.437389	15.156142	14432.698671	2.154758	20.753928
min	1.00000	601.000000	17.966120	-171.688150	0.000000	0.000000	18.000000
25%	2500.75000	26292.500000	35.341828	-97.082812	738.000000	0.000000	35.000000

	CaseOrder	Zip	Lat	Lng	Population	Children	Age
50%	5000.50000	48869.500000	39.395800	-87.918800	2910.500000	1.000000	53.000000
75%	7500.25000	71866.500000	42.106908	-80.088745	13168.000000	3.000000	71.000000
max	10000.00000	99929.000000	70.640660	-65.667850	111850.000000	10.000000	89.000000

8 rows × 23 columns

In [11]:
```
#Remove columns whos variables should not be included in dataset
cd_stats = cd.drop(columns=['CaseOrder', 'Zip', 'Lat', 'Lng'])
cd_stats.describe()
```

Out[11]:

	Population	Children	Age	Income	Outage_sec_perweek	Email	
count	10000.000000	7505.000000	7525.000000	7510.000000	10000.000000	10000.000000	100
mean	9756.562400	2.095936	53.275748	39936.762226	11.452955	12.016000	
std	14432.698671	2.154758	20.753928	28358.469482	7.025921	3.025898	
min	0.000000	0.000000	18.000000	740.660000	-1.348571	1.000000	
25%	738.000000	0.000000	35.000000	19285.522500	8.054362	10.000000	
50%	2910.500000	1.000000	53.000000	33186.785000	10.202896	12.000000	
75%	13168.000000	3.000000	71.000000	53472.395000	12.487644	14.000000	
max	111850.000000	10.000000	89.000000	258900.700000	47.049280	23.000000	

In [12]:
```
#Calculate the churn rate for the dataset
cd.Churn.value_counts() / len(cd)
```

Out[12]:
```
No     0.735
Yes    0.265
Name: Churn, dtype: float64
```

In [13]:
```
#Locate missing values
cd.isnull()
```

Out[13]:

	CaseOrder	Customer_id	Interaction	City	State	County	Zip	Lat	Lng	Population	...	Mo
0	False	False	False	False	False	False	False	False	False	False	...	
1	False	False	False	False	False	False	False	False	False	False	...	
2	False	False	False	False	False	False	False	False	False	False	...	
3	False	False	False	False	False	False	False	False	False	False	...	
4	False	False	False	False	False	False	False	False	False	False	...	
...	...	...	...	...	...	...	...	...	...	...	...	
9995	False	False	False	False	False	False	False	False	False	False	...	

	CaseOrder	Customer_id	Interaction	City	State	County	Zip	Lat	Lng	Population	...	Mo
9996	False	False	False	False	False	False	False	False	False	False	...	
9997	False	False	False	False	False	False	False	False	False	False	...	
9998	False	False	False	False	False	False	False	False	False	False	...	
9999	False	False	False	False	False	False	False	False	False	False	...	

10000 rows × 51 columns

In [14]:
```
#Generate only rows that have missing data
cd.isnull().any(axis=1)
```

Out[14]:
```
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 [15]:
```
#Show columns with NA values
cd.isna().any()
```

Out[15]:
```
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         False
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
Response                False
Fixes                   False
Replacement             False
Reliability             False
Options                 False
Respectful              False
Courteous               False
Active_Listening        False
dtype: bool
```

In [16]:

```
#Show how much null values per columns
data_null = cd.isnull().sum()
print(data_null)
```

```
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          0
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
Response                  0
Fixes                     0
Replacement               0
Reliability               0
Options                   0
Respectful                0
Courteous                 0
Active_Listening          0
dtype: int64
```

In [17]:

```
#Assign missing values to a new variable
rows_missing_data = cd.isnull().any(axis=1)
cd[rows_missing_data]
```

Out[17]:

	CaseOrder	Customer_id	Interaction	City	State	County	Zip	Lat	L
0	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	-133.375
2	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	Yamhill	OR	Yamhill	97148	45.35589	-123.246
5	6	W303516	2b451d12-6c2b-4cea-a295-ba1d6bced078	Fort Valley	GA	Peach	31030	32.57032	-83.890
6	7	U335188	6630d501-838c-4be4-a59c-6f58c814ed6a	Pioneer	TN	Scott	37847	36.43420	-84.278
7	8	V538685	70ddaa89-b726-49dc-9022-2d655e4c7936	Oklahoma City	OK	Oklahoma	73109	35.43313	-97.524
...	...	...	...	...	...	...	...	...	
9994	9995	P175475	c60df12b-a50b-4397-ae57-98381a0d3960	West Kill	NY	Greene	12492	42.18491	-74.335
9995	9996	M324793	45deb5a2-ae04-4518-bf0b-c82db8dbe4a4	Mount Holly	VT	Rutland	5758	43.43391	-72.787

	CaseOrder	Customer_id	Interaction	City	State	County	Zip	Lat	L
9996	9997	D861732	6e96b921-0c09-4993-bbda-a1ac6411061a	Clarksville	TN	Montgomery	37042	36.56907	-87.41
9997	9998	I243405	e8307ddf-9a01-4fff-bc59-4742e03fd24f	Mobeetie	TX	Wheeler	79061	35.52039	-100.44
9999	10000	T38070	9de5fb6e-bd33-4995-aec8-f01d0172a499	Clarkesville	GA	Habersham	30523	34.70783	-83.53

7867 rows × 51 columns

In [18]:
```
#Identify data types for each column
cd.dtypes
```

Out[18]:
```
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
Response                  int64
Fixes                     int64
Replacement               int64
Reliability               int64
Options                   int64
Respectful                int64
Courteous                 int64
Active_Listening          int64
dtype: object
```

In [19]:
```
#Find unique values. Tried to do multiple columns which resulted with more confusion.Go
cd['Age'].unique()
```

Out[19]:
```
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 [20]:
```
#Assign and sort Age values in ascending order
age_range = cd['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, 24.0, 2
8.0, 29.0, 32.0, 33.0, 34.0, 35.0, 36.0, 37.0, 38.0, 42.0, 46.0, 53.0, 54.0, 55.0, 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, 72.0, 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, 87.0, 88.0, 89.0]
```

In [21]:
```
cd['Education'].unique()
```

Out[21]:
```
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 [22]:
```
cd['Contract'].unique()
```

Out[22]:
```
array(['One year', 'Month-to-month', 'Two Year'], dtype=object)
```

In [23]:
```
cd['Marital'].unique()
```

Out[23]:
```
array(['Widowed', 'Married', 'Separated', 'Never Married', 'Divorced'],
      dtype=object)
```

In [24]:
```
cd['PaymentMethod'].unique()
```

Out[24]:
```
array(['Credit Card (automatic)', 'Bank Transfer(automatic)',
       'Mailed Check', 'Electronic Check'], dtype=object)
```

In [25]:
```
cd['Response'].unique()
```

Out[25]:
```
array([5, 3, 4, 6, 2, 1, 7], dtype=int64)
```

In [26]:
```
cd['Response'].unique()
```

Out[26]:
```
array([5, 3, 4, 6, 2, 1, 7], dtype=int64)
```

In [27]:
```
#Duplicate rows
duplicate_data = cd.loc[cd.duplicated()]
print(duplicate_data)
```

```
Empty DataFrame
Columns: [CaseOrder, Customer_id, Interaction, City, State, County, Zip, Lat, Lng, Popul
ation, Area, Timezone, Job, Children, Age, Education, Employment, Income, Marital, Gende
r, Churn, Outage_sec_perweek, Email, Contacts, Yearly_equip_failure, Techie, Contract, P
ort_modem, Tablet, InternetService, Phone, Multiple, OnlineSecurity, OnlineBackup, Devic
eProtection, TechSupport, StreamingTV, StreamingMovies, PaperlessBilling, PaymentMethod,
Tenure, MonthlyCharge, Bandwidth_GB_Year, Response, Fixes, Replacement, Reliability, Opt
ions, Respectful, Courteous, Active_Listening]
Index: []

[0 rows x 51 columns]
```

In [28]:
```
#Find standard deviation for each quantitative column
cd_std = cd_stats.std()
print(cd_std)
```

```
Population             14432.698671
Children                   2.154758
Age                       20.753928
Income                 28358.469482
Outage_sec_perweek         7.025921
Email                      3.025898
Contacts                   0.988466
Yearly_equip_failure       0.635953
Tenure                    26.438904
MonthlyCharge             43.335473
Bandwidth_GB_Year       2187.396807
Response                   1.037797
Fixes                      1.034641
Replacement                1.027977
Reliability                1.025816
Options                    1.024819
Respectful                 1.033586
Courteous                  1.028502
Active_Listening           1.028633
dtype: float64
```

In [29]:
```
#Finding the nulls of each column
null_data = cd_stats.isnull().sum()
print(null_data)
```

```
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          0
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
Response                 0
Fixes                    0
Replacement              0
Reliability              0
Options                  0
Respectful               0
Courteous                0
Active_Listening         0
dtype: int64
```

In [30]:

```
#Use median to enter into missing fields
cd_stats['Income'] = cd['Income'].fillna(cd['Income'].median())
cd_stats['Children'] = cd['Children'].fillna(cd['Children'].median())
cd_stats['Bandwidth_GB_Year'] = cd['Bandwidth_GB_Year'].fillna(cd['Bandwidth_GB_Year'].
cd_stats['Age'] = cd['Age'].fillna(cd['Age'].median())
cd_stats['Tenure'] = cd['Tenure'].fillna(cd['Tenure'].median())
```

In [31]:

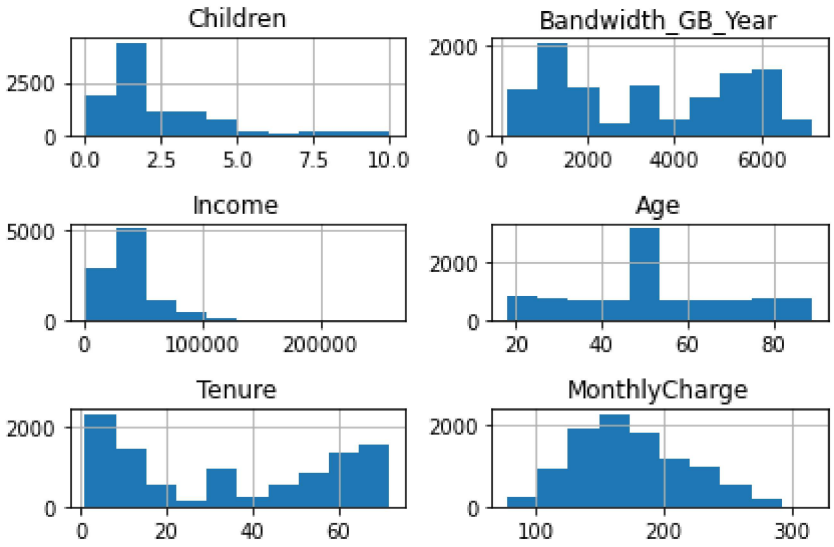
```
null_data = cd_stats.isnull().sum()
print(null_data)
```

```
Customer_id           0
Interaction           0
City                  0
State                 0
County                0
```

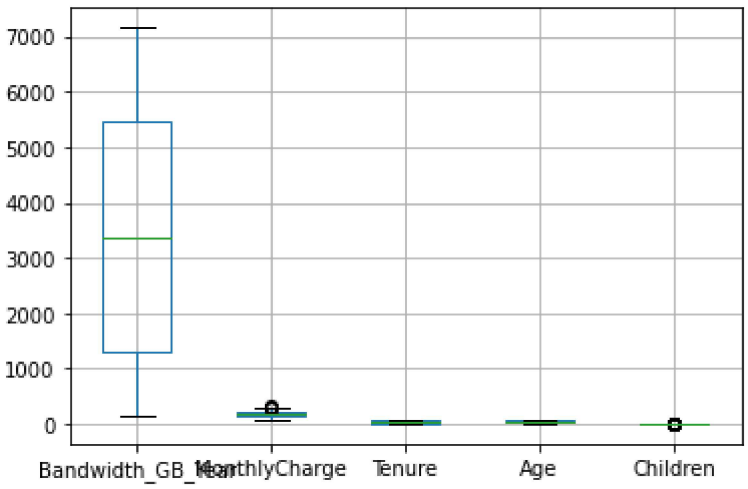
```
Population                0
Area                      0
Timezone                  0
Job                       0
Children                  0
Age                       0
Education                 0
Employment                0
Income                    0
Marital                   0
Gender                    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           0
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
Response                  0
Fixes                     0
Replacement               0
Reliability               0
Options                   0
Respectful                0
Courteous                 0
Active_Listening          0
dtype: int64
```

In [32]:

```
#Histogram creation and display
cd_stats[['Children', 'Bandwidth_GB_Year', 'Income', 'Age', 'Tenure', 'MonthlyCharge']]
plt.savefig('churn_pyplot.jpg')
plt.tight_layout()
```

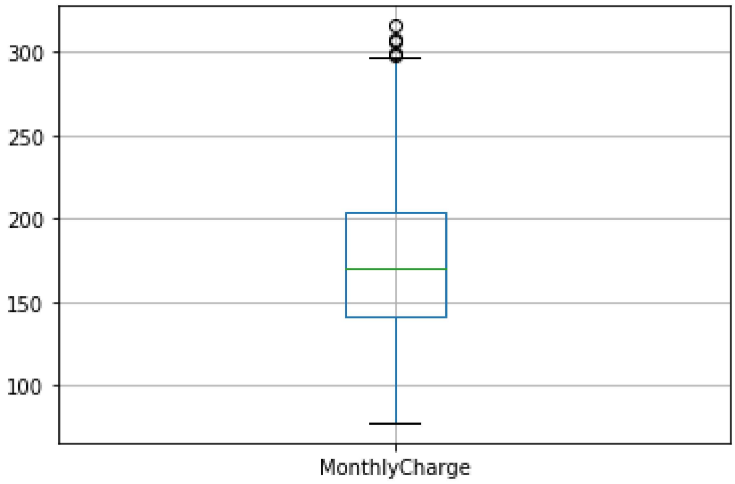


In [33]:
```
#Finding some skewed and abnormally high data on certain columns within each dataset
#Boxplot creation to help identify outliers
cd_stats.boxplot(['Bandwidth_GB_Year', 'MonthlyCharge', 'Tenure', 'Age', 'Children'])
plt.savefig('churn_boxplots.jpg')
```



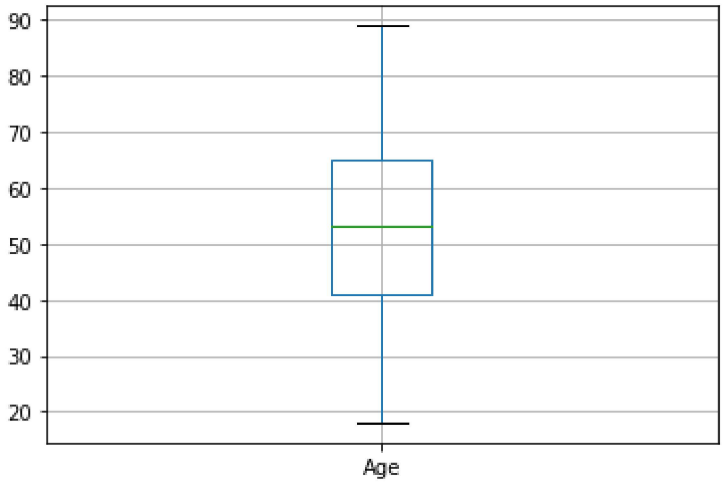
In [34]:
```
#Look closer at individual boxplots to examine outliers
cd_stats.boxplot(['MonthlyCharge'])
#definitely have outliers
```

Out[34]: <AxesSubplot:>



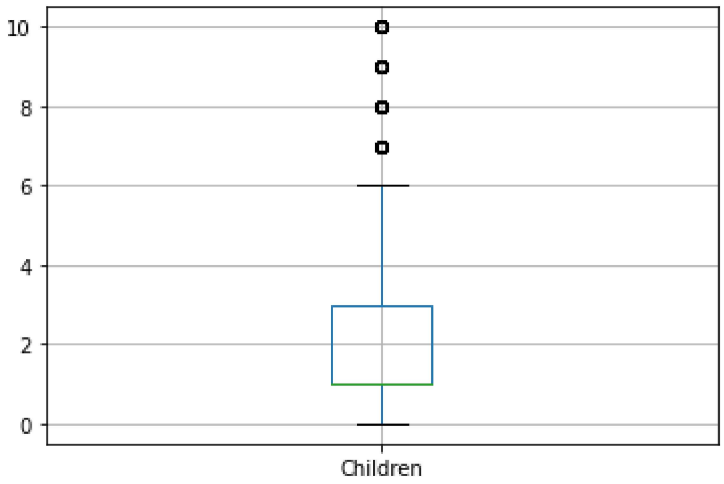
In [35]:
```
cd_stats.boxplot(['Age'])
```

Out[35]: <AxesSubplot:>



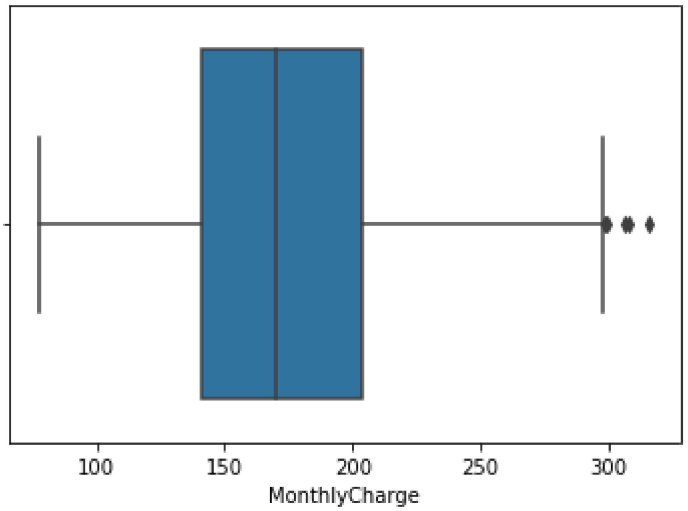
In [36]:
```
cd_stats.boxplot(['Children'])
#outliers present
```

Out[36]: <AxesSubplot:>



In [37]:
```
#Seaborn boxplot
sns.boxplot('MonthlyCharge', data = cd_stats)
plt.show()
```

```
C:\Users\coope\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```



In [38]:
```
#Extract cleaned dataset
cd_stats.to_csv('churn_clean.csv')
```

In [39]:
```
#Reload cleaned data
churn_user = pd.read_csv('churn_clean.csv')
```

In [40]:
```
#Removing columns I don't need and displaying first 5 rows for reference
data = churn_user.loc[:, 'Tenure':'Active_Listening']
data.head()
```

Out[40]:

	Tenure	MonthlyCharge	Bandwidth_GB_Year	Response	Fixes	Replacement	Reliability	Options
0	6.795513	171.449762	904.536110	5	5	5	3	4
1	1.156681	242.948015	800.982766	3	4	3	3	4
2	15.754144	159.440398	2054.706961	4	4	2	4	4
3	17.087227	120.249493	2164.579412	4	4	4	2	5
4	1.670972	150.761216	271.493436	4	4	4	3	4

In [41]:
```
#PCA ned to import sklearn for PCA analysis
from sklearn.decomposition import PCA
```

In [42]:
```
#normalizing
churn_normalized = (data - data.mean()) / data.std()
```

In [43]:
```
#creating name to hold number of variables for extraction
pca = PCA(n_components = data.shape[1])
```

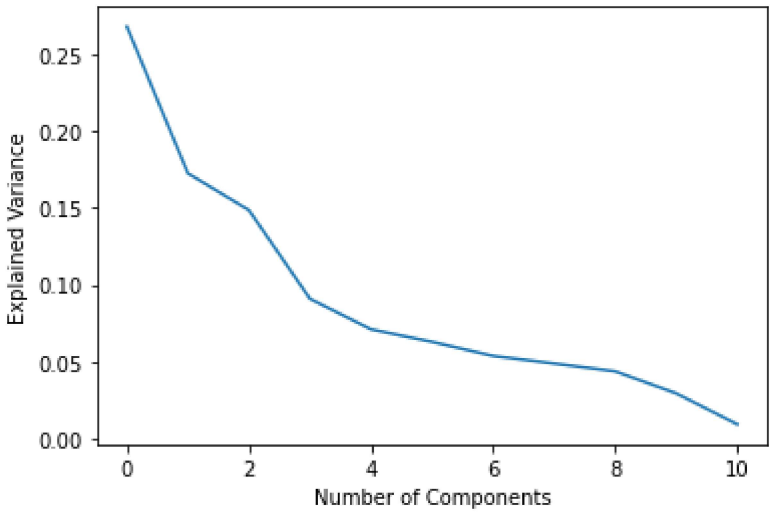
In [44]:
```
#List of PCA names in order of table column names
churn_numeric = data[['Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year', 'Response', 'Fixe
pcs_names = []
for i, col in enumerate(churn_numeric.columns):
    pcs_names.append('PC' + str(i + 1))
print(pcs_names)
```

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

In [45]:
```
#Convert 11 variables into components
pca.fit(churn_normalized)
pca_churn = pd.DataFrame(pca.transform(churn_normalized), columns = pcs_names)
```

In [46]:
```
#Need matplotlib for scree plot
import matplotlib.pyplot as plt
import seaborn as sns
```

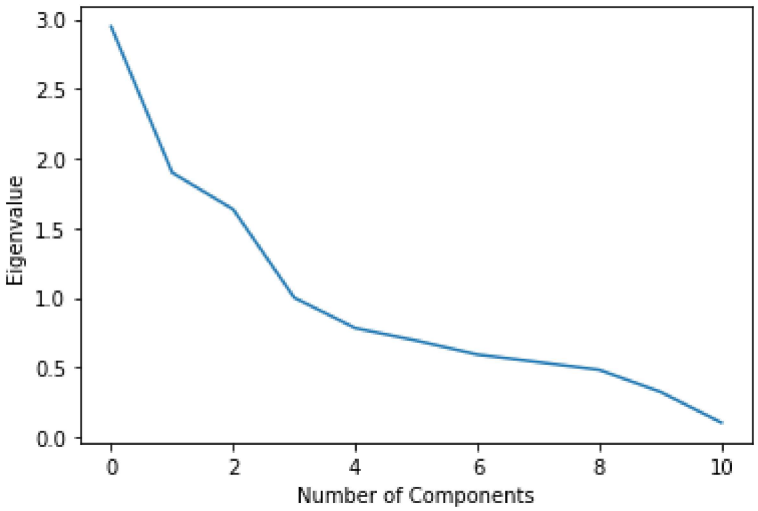
In [53]:
```
#Create scree plot
plt.plot(pca.explained_variance_ratio_)
plt.xlabel('Number of Components')
plt.ylabel('Explained Variance')
plt.show()
```



In [56]:
```
#Get eigenvalues
cov_matrix = np.dot(churn_normalized.T, churn_normalized) / data.shape[0]
eigenvalues = [np.dot(eigenvector.T, np.dot(cov_matrix, eigenvector))
               for eigenvector in pca.components_]
```

In [57]:
```
#Plot those values
plt.plot(eigenvalues)
plt.xlabel('Number of Components')
```

```
plt.ylabel('Eigenvalue')
plt.show()
```



In [58]:

```
#PC value
for pc, var in zip(pcs_names, np.cumsum(pca.explained_variance_ratio_)):
    print(pc, var)
```

```
PC1 0.26792660623963077
PC2 0.4405373288022937
PC3 0.5891444522248596
PC4 0.6801579842425085
PC5 0.75132170543725
PC6 0.814311709410132
PC7 0.8681970531672317
PC8 0.9171387362588829
PC9 0.9610122820002187
PC10 0.9905736363144831
PC11 1.0
```

In [61]:

```
#Rotate - better visualize components
rotation = pd.DataFrame(pca.components_.T, columns = pcs_names, index = churn_numeric.c
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
Response           0.458932  0.031325  0.281154  0.018568 -0.070233 -0.119149
Fixes              0.434134  0.042559  0.282404  0.007508 -0.106632 -0.169752
Replacement        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
Respectful         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
Active_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
Response         -0.045963  0.025431 -0.240574  0.793237 -0.004306
Fixes            -0.065414  0.074400 -0.592131 -0.573832 -0.002217
Replacement      -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
Respectful         0.757954  0.402835  0.230319 -0.063972  0.001604
Courteous         -0.379136  0.067889  0.067293 -0.040946 -0.006875
Active_Listening  -0.113297 -0.045132  0.046107 -0.042251 -0.002357
```

In [62]:
```
#component loading
loadings = pd.DataFrame(pca.components_.T,
                        columns = pcs_names,
                        index = data.columns)
```

In [63]:
```
loadings
```

Out[63]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
Tenure	-0.010403	0.701838	-0.072209	-0.063594	0.005683	-0.011155	0.007419	-0.011527
MonthlyCharge	0.000317	0.041147	-0.014151	0.996995	-0.022136	0.015231	-0.018038	-0.004316
Bandwidth_GB_Year	-0.012166	0.703079	-0.074222	0.004399	0.009590	0.003466	0.003701	-0.002364
Response	0.458932	0.031325	0.281154	0.018568	-0.070233	-0.119149	-0.045963	0.025431
Fixes	0.434134	0.042559	0.282404	0.007508	-0.106632	-0.169752	-0.065414	0.074400
Replacement	0.400639	0.034665	0.281118	-0.019631	-0.173742	-0.255336	-0.146887	-0.396333
Reliability	0.145799	-0.050367	-0.567815	-0.010310	-0.171334	-0.483328	-0.443353	0.431528
Options	-0.175633	0.066334	0.587335	-0.000047	0.135949	0.060124	-0.209767	0.693861
Respectful	0.405207	-0.012680	-0.183447	0.004596	-0.062342	0.064609	0.757954	0.402835
Courteous	0.358342	-0.003886	-0.181697	-0.027959	-0.182406	0.806166	-0.379136	0.067889
Active_Listening	0.308925	-0.017396	-0.131173	0.015574	0.931612	-0.011133	-0.113297	-0.045132

In [65]:
```
#Extract dataset
#Show 3 PCAs
churn_reduced = pca_churn.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 [ ]: