Cleaning data, performing PCA and printing Scree Plot

In [1]:
```
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
import missingno as msno
from sklearn.decomposition import PCA
```

In [7]:
```
df2 = pd.read_csv('churn_raw_data.csv')
```

In [8]:
```
df = df2.drop('Unnamed: 0', axis=1)
```

In [9]:
```
df.info()
```

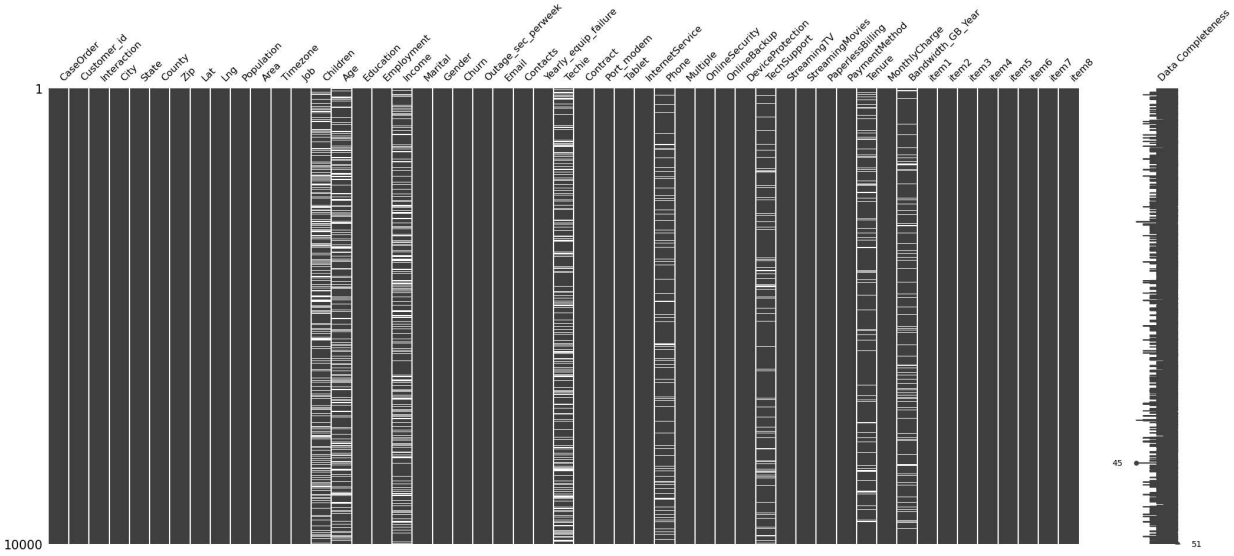
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 51 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   CaseOrder             10000 non-null  int64
 1   Customer_id           10000 non-null  object
 2   Interaction           10000 non-null  object
 3   City                  10000 non-null  object
 4   State                 10000 non-null  object
 5   County                10000 non-null  object
 6   Zip                   10000 non-null  int64
 7   Lat                   10000 non-null  float64
 8   Lng                   10000 non-null  float64
 9   Population            10000 non-null  int64
 10  Area                  10000 non-null  object
 11  Timezone              10000 non-null  object
 12  Job                   10000 non-null  object
 13  Children              7505 non-null   float64
 14  Age                   7525 non-null   float64
 15  Education             10000 non-null  object
 16  Employment            10000 non-null  object
 17  Income                7510 non-null   float64
 18  Marital               10000 non-null  object
 19  Gender                10000 non-null  object
 20  Churn                 10000 non-null  object
 21  Outage_sec_perweek    10000 non-null  float64
 22  Email                 10000 non-null  int64
 23  Contacts              10000 non-null  int64
 24  Yearly_equip_failure  10000 non-null  int64
 25  Techie                7523 non-null   object
 26  Contract              10000 non-null  object
 27  Port_modem            10000 non-null  object
 28  Tablet                10000 non-null  object
 29  InternetService       10000 non-null  object
 30  Phone                 8974 non-null   object
 31  Multiple              10000 non-null  object
 32  OnlineSecurity        10000 non-null  object
 33  OnlineBackup          10000 non-null  object
 34  DeviceProtection      10000 non-null  object
 35  TechSupport           9009 non-null   object
 36  StreamingTV           10000 non-null  object
 37  StreamingMovies       10000 non-null  object
 38  PaperlessBilling      10000 non-null  object
 39  PaymentMethod         10000 non-null  object
 40  Tenure                9069 non-null   float64
 41  MonthlyCharge         10000 non-null  float64
 42  Bandwidth_GB_Year     8979 non-null   float64
 43  item1                 10000 non-null  int64
 44  item2                 10000 non-null  int64
 45  item3                 10000 non-null  int64
 46  item4                 10000 non-null  int64
 47  item5                 10000 non-null  int64
 48  item6                 10000 non-null  int64
 49  item7                 10000 non-null  int64
 50  item8                 10000 non-null  int64
dtypes: float64(9), int64(14), object(28)
memory usage: 3.9+ MB
```

In [10]:
```
print(df.duplicated().value_counts())
```

```
False    10000
dtype: int64
```

In [11]:
```
#Missingno to visualize missing data
msno.matrix(df, fontsize = 12, labels=True)
```

Out[11]:
```
<AxesSubplot:>
```



In [12]:
```
#get missing values
df.isna().sum()
```

Out[12]:
```
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
item1                    0
item2                    0
item3                    0
item4                    0
item5                    0
item6                    0
item7                    0
item8                    0
dtype: int64
```

In [13]:
```
df['Children'].value_counts()
```

Out[13]:
```
0.0     1919
1.0     1874
2.0     1100
3.0     1096
4.0      769
5.0      161
8.0      158
7.0      149
6.0      135
10.0      74
9.0       70
Name: Children, dtype: int64
```

In [14]:
```
df['Techie'].value_counts()
```

Out[14]:
```
No     6266
Yes    1257
Name: Techie, dtype: int64
```

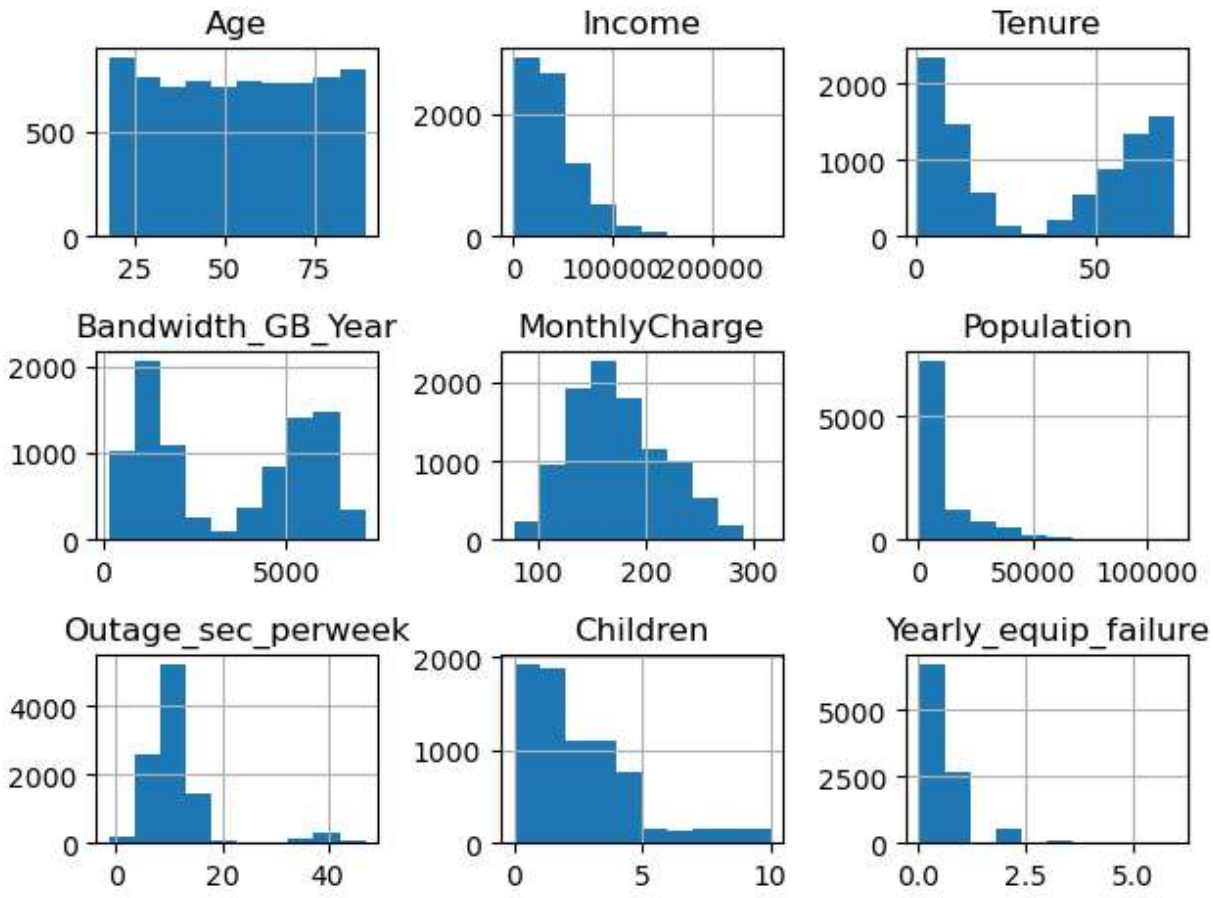
In [15]:
```
df['TechSupport'].value_counts()
```

Out[15]:
```
No     5635
Yes    3374
Name: TechSupport, dtype: int64
```

In [16]:
```
df['Churn'].value_counts()
```

Out[16]:
```
No     7350
Yes    2650
Name: Churn, dtype: int64
```

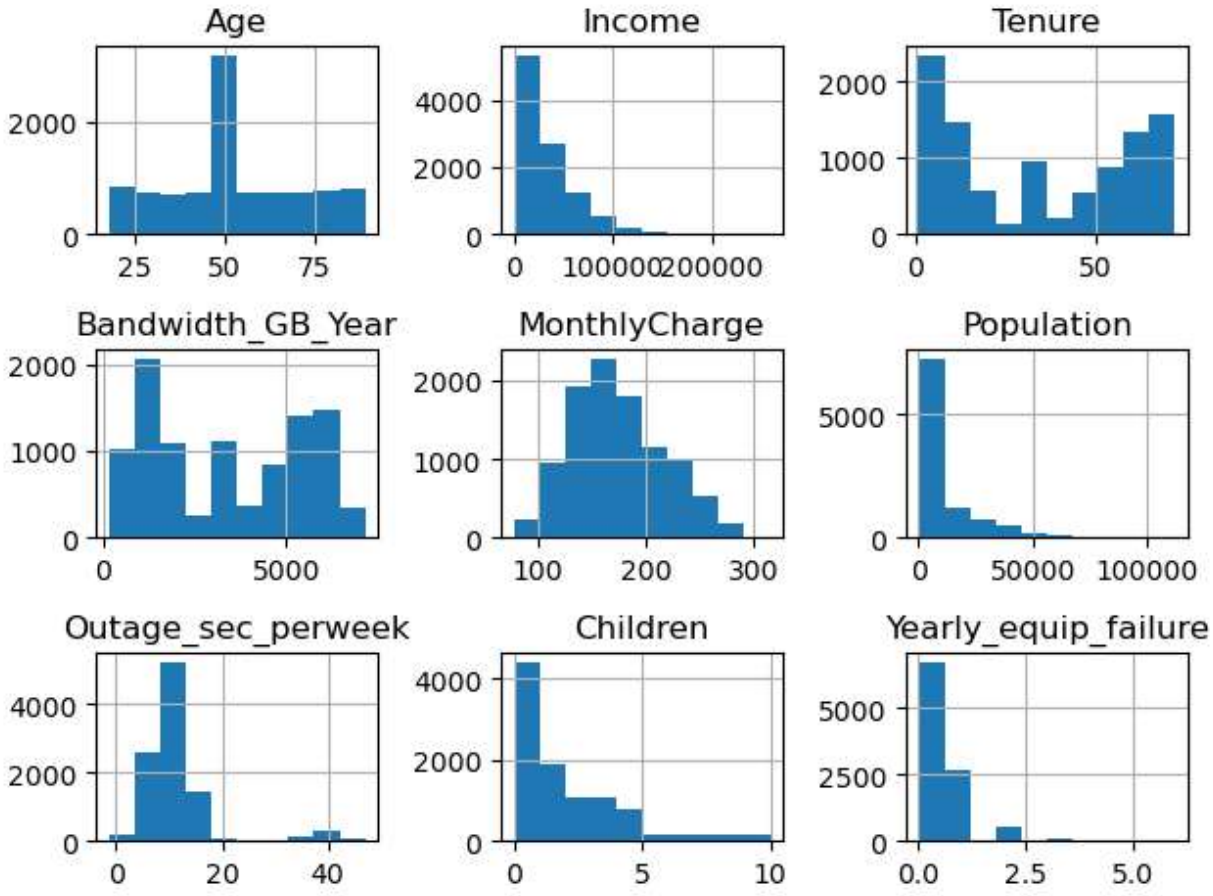
In [17]:
```
df[['Age', 'Income', 'Tenure', 'Bandwidth_GB_Year', 'MonthlyCharge', 'Population', 'Ou
plt.tight_layout()
```



In [18]:
```
df['Age'].fillna(df['Age'].mean(), inplace = True)
df['Bandwidth_GB_Year'].fillna(df['Bandwidth_GB_Year'].mean(), inplace = True)
df["Techie"].fillna("No", inplace = True)
df["Phone"].fillna("No", inplace = True)
df["Income"].fillna("0", inplace = True)
df["TechSupport"].fillna("No", inplace = True)
df["Children"].fillna("0", inplace = True)
df['Tenure'].fillna(df['Tenure'].mean(), inplace = True)
```

In [19]:
```
#changing 'Age', 'Children', and 'Income' datatype to Integer before filling in missir
df = df.astype({'Age':'int'})
df = df.astype({'Children':'int'})
df = df.astype({'Income':'int'})
```

In [20]:
```
df[['Age', 'Income', 'Tenure', 'Bandwidth_GB_Year', 'MonthlyCharge', 'Population', 'Ou
plt.tight_layout()
```

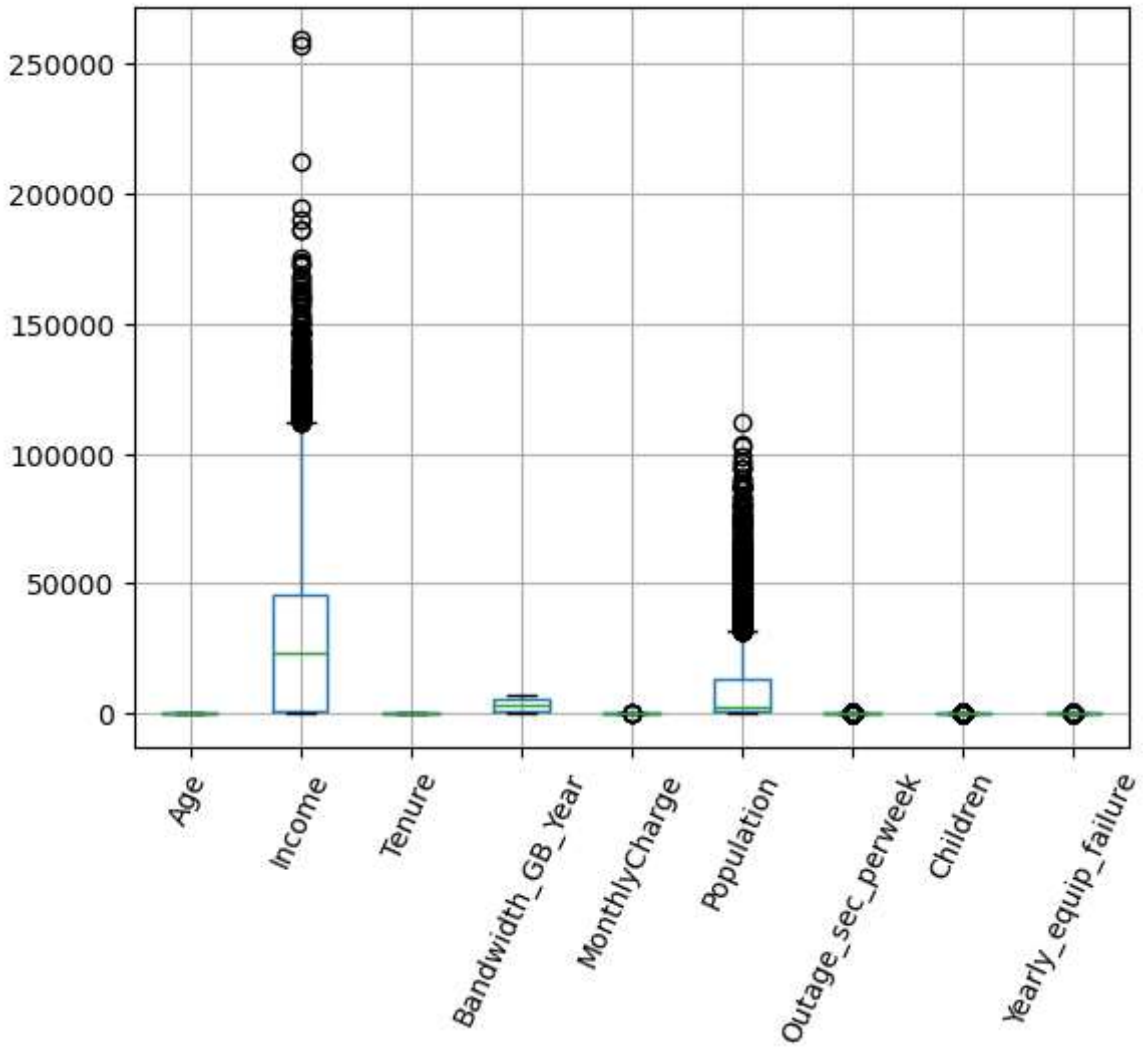


In [21]:
```
df.std(numeric_only=True)
```

Out[21]:
```
CaseOrder                2886.895680
Zip                     27532.196108
Lat                         5.437389
Lng                        15.156142
Population              14432.698671
Children                    2.075356
Age                        18.003457
Income                  30036.848168
Outage_sec_perweek          7.025921
Email                       3.025898
Contacts                    0.988466
Yearly_equip_failure        0.635953
Tenure                     25.177982
MonthlyCharge              43.335473
Bandwidth_GB_Year        2072.712613
item1                       1.037797
item2                       1.034641
item3                       1.027977
item4                       1.025816
item5                       1.024819
item6                       1.033586
item7                       1.028502
item8                       1.028633
dtype: float64
```

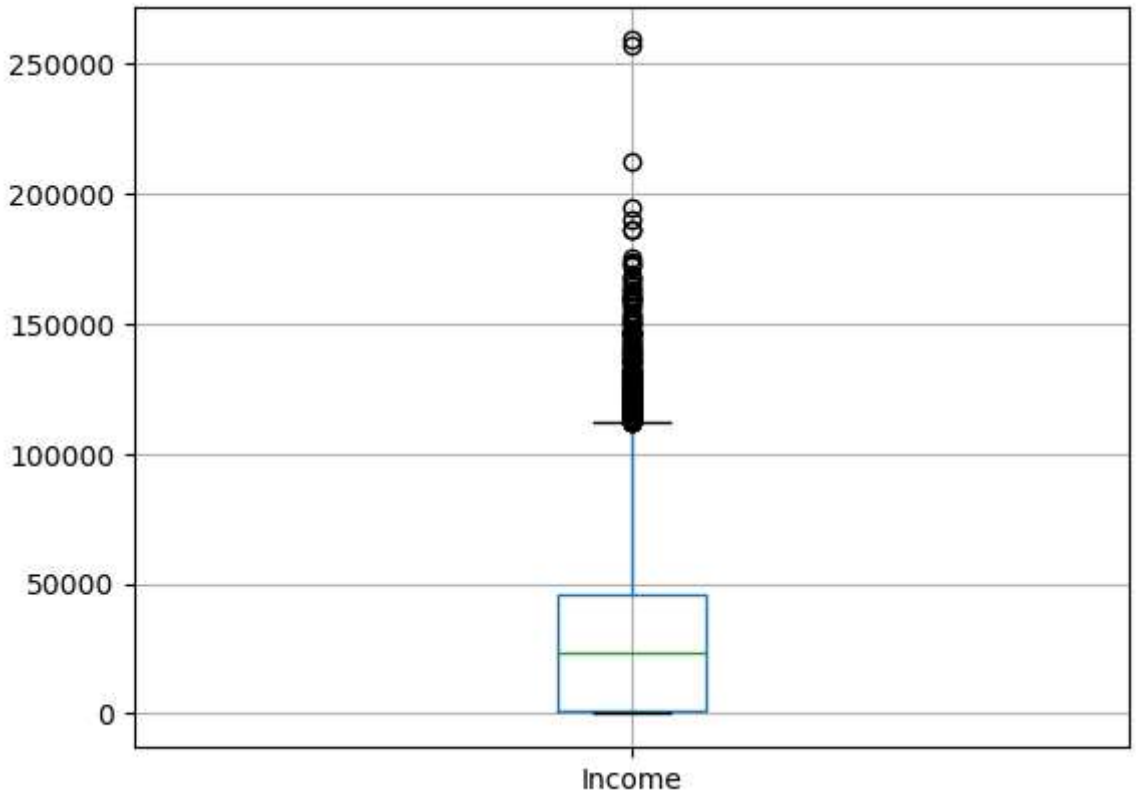
In [22]:
```
df.boxplot(column=['Age', 'Income', 'Tenure', 'Bandwidth_GB_Year', 'MonthlyCharge', 'P
plt.xticks(rotation=65)
```

Out[22]:
```
(array([1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(1, 0, 'Age'),
  Text(2, 0, 'Income'),
  Text(3, 0, 'Tenure'),
  Text(4, 0, 'Bandwidth_GB_Year'),
  Text(5, 0, 'MonthlyCharge'),
  Text(6, 0, 'Population'),
  Text(7, 0, 'Outage_sec_perweek'),
  Text(8, 0, 'Children'),
  Text(9, 0, 'Yearly_equip_failure')])
```



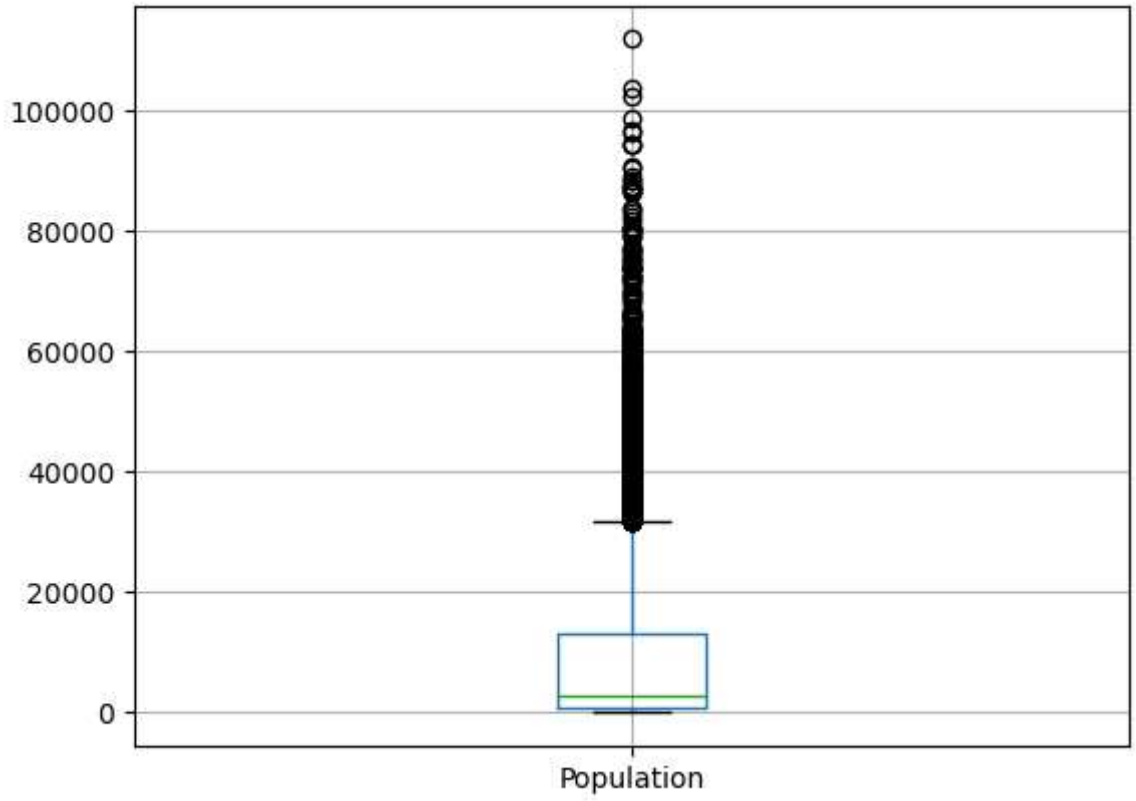
In [23]:
```
df.boxplot(['Income'])
```

Out[23]:
```
<AxesSubplot:>
```



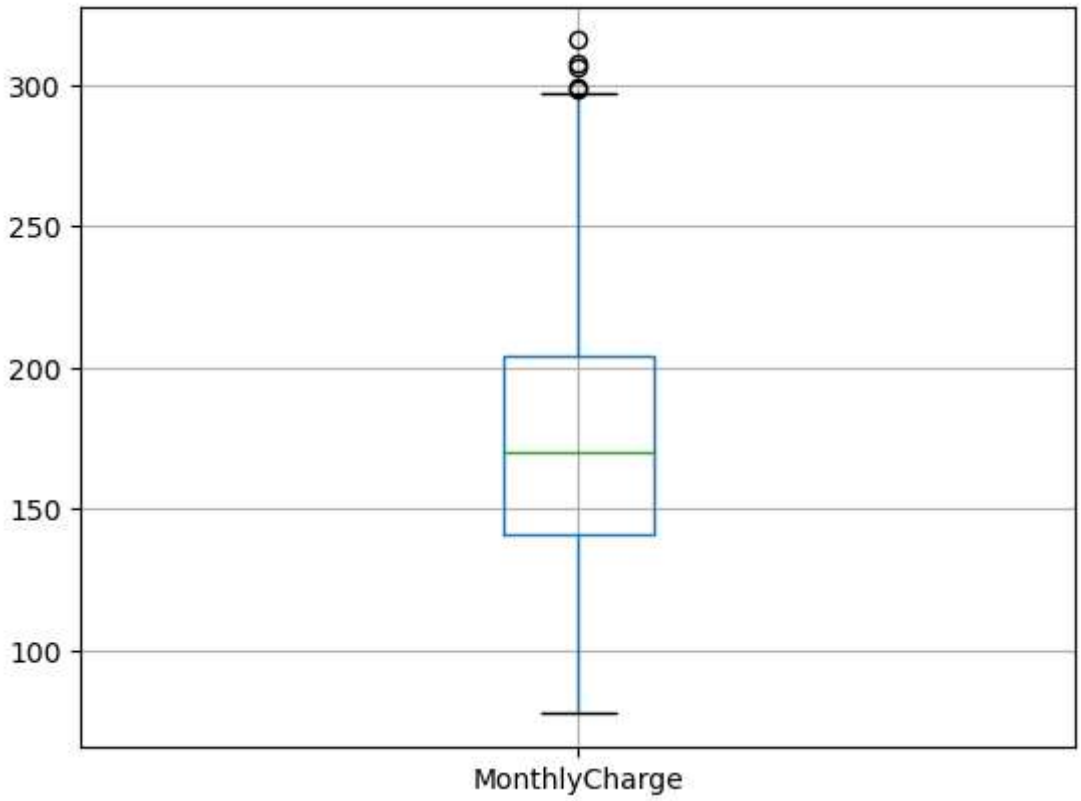
In [24]:
```
df.boxplot(['Population'])
```

Out[24]: <AxesSubplot:>



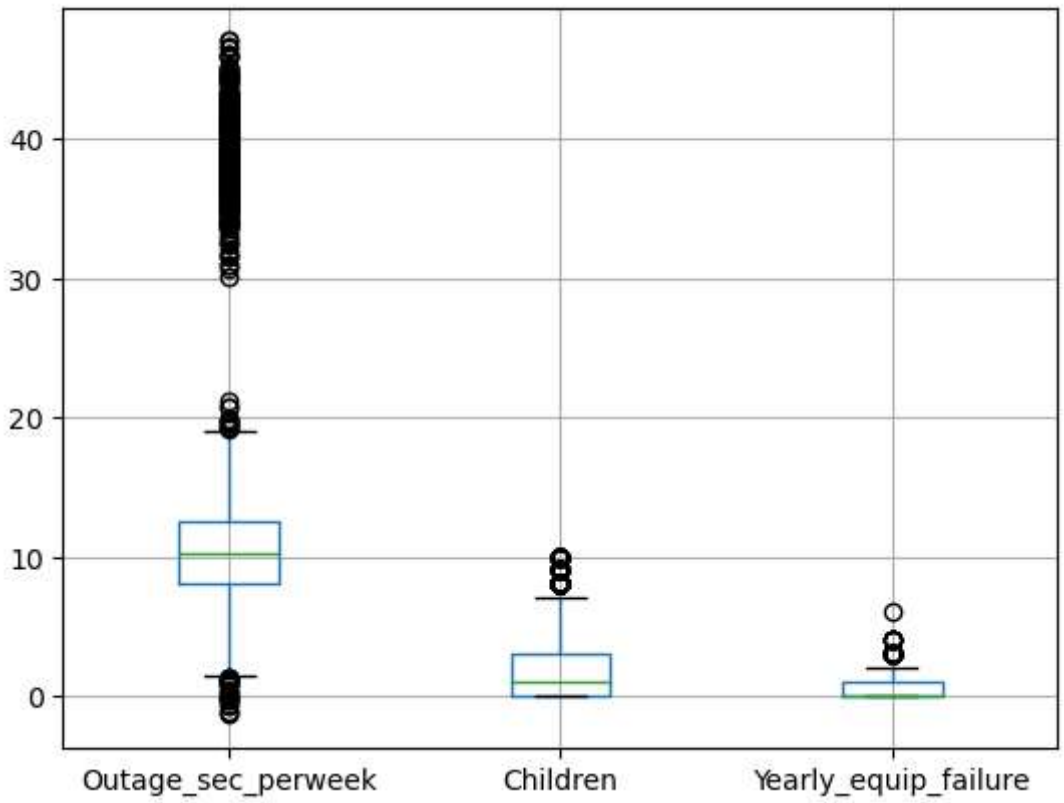
In [25]:
```
df.boxplot(['MonthlyCharge'])
```

Out[25]: <AxesSubplot:>



In [26]:
```
df.boxplot(column=['Outage_sec_perweek', 'Children', 'Yearly_equip_failure'])
```

Out[26]: <AxesSubplot:>



In [27]:
```
#if wanted we can rename survey response for clarity
#df.rename(columns = {'item1':'Survey_Responses', 'item2':'Survey_Fixes', 'item3': 'Su
```

In [28]:
```
df.to_csv('clean_churn_raw_data.csv', index=False)
```

PCA - Princpal component analysis

In [29]:
```
dfpca = pd.read_csv('clean_churn_raw_data.csv')
```

In [30]:
```
dfpca = df[['Outage_sec_perweek','Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year', 'iter
```

In [31]:
```
dfpcanormalized=(dfpca-dfpca.mean())/dfpca.std()
```

In [32]:
```
pca = PCA(n_components=dfpca.shape[1])
```

In [33]:
```
pca.fit(dfpcanormalized)
```

Out[33]:
```
PCA(n_components=12)
```

In [34]:
```
dfpca2 = pd.DataFrame(pca.transform(dfpcanormalized),columns=['PC1','PC2','PC3','PC4',
```

In [35]:
```
loadings=pd.DataFrame(pca.components_.T,
columns=['PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC10','PC11', 'PC12'],
index=dfpcanormalized.columns)
loadings
```

Out[35]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	
Outage_sec_perweek	-0.013095	0.018143	-0.048093	0.702378	0.701621	-0.104609	0.008284	-0.013
Tenure	-0.010671	0.701678	-0.069509	-0.058171	0.035723	0.002902	-0.011113	0.007
MonthlyCharge	-0.000572	0.044005	-0.023888	0.705336	-0.703684	0.040144	0.011706	-0.013
Bandwidth_GB_Year	-0.012420	0.703095	-0.072218	-0.010223	-0.011483	0.011527	0.003565	0.003
item1	0.458937	0.031085	0.280191	0.030630	-0.003708	-0.071382	-0.118892	-0.046
item2	0.434162	0.042284	0.281710	0.019594	-0.002491	-0.107904	-0.169426	-0.066
item3	0.400745	0.034350	0.281102	-0.011675	-0.008347	-0.172882	-0.255269	-0.146
item4	0.145648	-0.049151	-0.567237	-0.031686	-0.022497	-0.170017	-0.483180	-0.443
item5	-0.175485	0.065189	0.586767	0.027868	0.034003	0.132328	0.060178	-0.210
item6	0.405094	-0.011886	-0.183877	0.004373	0.000198	-0.064035	0.065042	0.757
item7	0.358318	-0.003543	-0.181223	-0.034142	-0.014704	-0.179606	0.806339	-0.379
item8	0.308775	-0.016918	-0.132204	0.035415	0.095812	0.926230	-0.012562	-0.112

In [36]:
```
# covariance matrix is a square matrix giving the covariance between each pair of elem
cov_matrix = np.dot(dfpcanormalized.T, dfpcanormalized) / dfpca.shape[0]
```

In [37]:
```
#scree plot, elbow is cutoff point for significant variables
eigenvalues = [np.dot(eigenvector.T, np.dot(cov_matrix, eigenvector)) for eigenvector
```

In [38]:
```
plt.plot(eigenvalues)
plt.xlabel('number of components')
plt.ylabel('eigenvalues')
plt.axhline(y=1, color='red')
plt.show()
```