***Import certain python libraries***

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

***Code to import the csv***

In [2]:
```
df = pd.read_csv(r'C:\Users\richa\OneDrive\Desktop\d206\churn_raw_data.csv')
print(df)
```

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

              City State                 County    Zip       Lat        Lng  \
0      Point Baker    AK  Prince of Wales-Hyder  99927  56.25100 -133.37571
1      West Branch    MI                Ogemaw  48661  44.32893  -84.24080
2          Yamhill    OR               Yamhill  97148  45.35589 -123.24657
3          Del Mar    CA             San Diego  92014  32.96687 -117.24798
4        Needville    TX             Fort Bend  77461  29.38012  -95.80673
...            ...   ...                   ...    ...       ...        ...
9995   Mount Holly    VT               Rutland   5758  43.43391  -72.78734
9996   Clarksville    TN            Montgomery  37042  36.56907  -87.41694
9997      Mobeetie    TX               Wheeler  79061  35.52039 -100.44180
9998    Carrollton    GA               Carroll  30117  33.58016  -85.13241
9999  Clarkesville    GA              Habersham  30523  34.70783  -83.53648

      ...  MonthlyCharge Bandwidth_GB_Year item1 item2  item3  item4 item5  \
0     ...     171.449762        904.536110     5     5      5      3     4
1     ...     242.948015        800.982766     3     4      3      3     4
2     ...     159.440398       2054.706961     4     4      2      4     4
3     ...     120.249493       2164.579412     4     4      4      2     5
4     ...     150.761216        271.493436     4     4      4      3     4
...   ...            ...               ...   ...   ...    ...    ...   ...
9995  ...     159.828800       6511.253000     3     2      3      3     4
9996  ...     208.856400       5695.952000     4     5      5      4     4
9997  ...     168.220900       4159.306000     4     4      4      4     4
9998  ...     252.628600       6468.457000     4     4      6      4     3
9999  ...     218.371000       5857.586000     2     2      3      3     3

     item6  item7 item8
0        4      3     4
1        3      4     4
2        3      3     3
3        4      3     3
4        4      4     5
...    ...    ...   ...
9995     3      2     3
9996     5      2     5
9997     4      4     5
9998     3      5     4
9999     3      4     1

[10000 rows x 52 columns]
```

***indexing CaseOrder column***

In [3]:
```
df_index = pd.read_csv(r'C:\Users\richa\OneDrive\Desktop\d206\churn_raw_data.csv', usecols = ['CaseOrder
'Customer_id'])
print(df_index)
print(df_index["CaseOrder"])
type(df["CaseOrder"])
```

```
      CaseOrder Customer_id
0             1     K409198
1             2     S120509
2             3     K191035
3             4      D90850
4             5     K662701
...         ...         ...
9995       9996     M324793
9996       9997     D861732
9997       9998     I243405
9998       9999     I641617
9999      10000      T38070

[10000 rows x 2 columns]
0           1
1           2
2           3
3           4
4           5
         ...
9995     9996
9996     9997
9997     9998
9998     9999
9999    10000
Name: CaseOrder, Length: 10000, dtype: int64
```

Out[3]: pandas.core.series.Series

########## Step 2: Rename columns for better description of the variables

In [4]:
```
df.rename(columns = {'item1':'Timely_response', 'item2':'Timely_fixes',
                     'item3':'Timely_replacements', 'item4':'Reliability',
                      'item5':'options', 'item6':'Respectful_response',
                     'item7':'Courteous_exchange', 'item8':'Active_listening'}, inplace=True)
print(df)
```

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

              City State                County    Zip       Lat        Lng  \
0      Point Baker    AK  Prince of Wales-Hyder  99927  56.25100 -133.37571
1      West Branch    MI                Ogemaw  48661  44.32893  -84.24080
2          Yamhill    OR              Yamhill  97148  45.35589 -123.24657
3          Del Mar    CA            San Diego  92014  32.96687 -117.24798
4        Needville    TX            Fort Bend  77461  29.38012  -95.80673
...            ...   ...                  ...    ...       ...        ...
9995   Mount Holly    VT              Rutland   5758  43.43391  -72.78734
9996   Clarksville    TN           Montgomery  37042  36.56907  -87.41694
9997      Mobeetie    TX              Wheeler  79061  35.52039 -100.44180
9998    Carrollton    GA              Carroll  30117  33.58016  -85.13241
9999  Clarkesville    GA            Habersham  30523  34.70783  -83.53648

      ...  MonthlyCharge Bandwidth_GB_Year Timely_response Timely_fixes  \
0     ...     171.449762        904.536110               5            5
1     ...     242.948015        800.982766               3            4
2     ...     159.440398       2054.706961               4            4
3     ...     120.249493       2164.579412               4            4
4     ...     150.761216        271.493436               4            4
...   ...            ...               ...             ...          ...
9995  ...     159.828800       6511.253000               3            2
9996  ...     208.856400       5695.952000               4            5
9997  ...     168.220900       4159.306000               4            4
9998  ...     252.628600       6468.457000               4            4
9999  ...     218.371000       5857.586000               2            2

      Timely_replacements  Reliability options Respectful_response  \
0                       5            3       4                   4
1                       3            3       4                   3
2                       2            4       4                   3
3                       4            2       5                   4
4                       4            3       4                   4
...                   ...          ...     ...                 ...
9995                    3            3       4                   3
9996                    5            4       4                   5
9997                    4            4       4                   4
9998                    6            4       3                   3
9999                    3            3       3                   3

      Courteous_exchange Active_listening
0                      3                4
1                      4                4
2                      3                3
3                      3                3
4                      4                5
...                  ...              ...
9995                   2                3
9996                   2                5
9997                   4                5
9998                   5                4
9999                   4                1

[10000 rows x 52 columns]
```

####### Display the list of Dataframe Columns

In [5]:
```
df.columns
```

Out[5]:
```
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', 'Timely_response', 'Timely_fixes',
       'Timely_replacements', 'Reliability', 'options', 'Respectful_response',
       'Courteous_exchange', 'Active_listening'],
      dtype='object')
```

########## Describe Churn dataset statistics

In [6]:
```
df.describe()
```

Out[6]:

| | Unnamed: 0 | CaseOrder | Zip | Lat | Lng | Population | Children | Age | Inco |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 10000.00000 | 10000.00000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 7505.000000 | 7525.000000 | 7510.000 |
| **mean** | 5000.50000 | 5000.50000 | 49153.319600 | 38.757567 | -90.782536 | 9756.562400 | 2.095936 | 53.275748 | 39936.762 |
| **std** | 2886.89568 | 2886.89568 | 27532.196108 | 5.437389 | 15.156142 | 14432.698671 | 2.154758 | 20.753928 | 28358.469 |
| **min** | 1.00000 | 1.00000 | 601.000000 | 17.966120 | -171.688150 | 0.000000 | 0.000000 | 18.000000 | 740.660 |
| **25%** | 2500.75000 | 2500.75000 | 26292.500000 | 35.341828 | -97.082812 | 738.000000 | 0.000000 | 35.000000 | 19285.522 |
| **50%** | 5000.50000 | 5000.50000 | 48869.500000 | 39.395800 | -87.918800 | 2910.500000 | 1.000000 | 53.000000 | 33186.785 |
| **75%** | 7500.25000 | 7500.25000 | 71866.500000 | 42.106908 | -80.088745 | 13168.000000 | 3.000000 | 71.000000 | 53472.395 |
| **max** | 10000.00000 | 10000.00000 | 99929.000000 | 70.640660 | -65.667850 | 111850.000000 | 10.000000 | 89.000000 | 258900.700 |

8 rows × 24 columns

########## Calculate Churn Rate

In [7]:
```
df.Churn.value_counts() / len(df)
```

Out[7]:
```
No     0.735
Yes    0.265
Name: Churn, dtype: float64
```

####### Identify the standard deviation of every numeric column in the data set

In [8]:
```
df.std()
```

```
C:\Users\richa\AppData\Local\Temp\ipykernel_182928\3390915376.py:1: FutureWarning: The default value of
numeric_only in DataFrame.std is deprecated. In a future version, it will default to False. In additio
n, specifying 'numeric_only=None' is deprecated. Select only valid columns or specify the value of nume
ric_only to silence this warning.
  df.std()
```

Out[8]:
```
Unnamed: 0               2886.895680
CaseOrder                2886.895680
Zip                     27532.196108
Lat                         5.437389
Lng                        15.156142
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
Timely_response             1.037797
Timely_fixes                1.034641
Timely_replacements         1.027977
Reliability                 1.025816
options                     1.024819
Respectful_response         1.033586
Courteous_exchange          1.028502
Active_listening            1.028633
dtype: float64
```

########## Review data type

In [9]:
```
df.dtypes
```

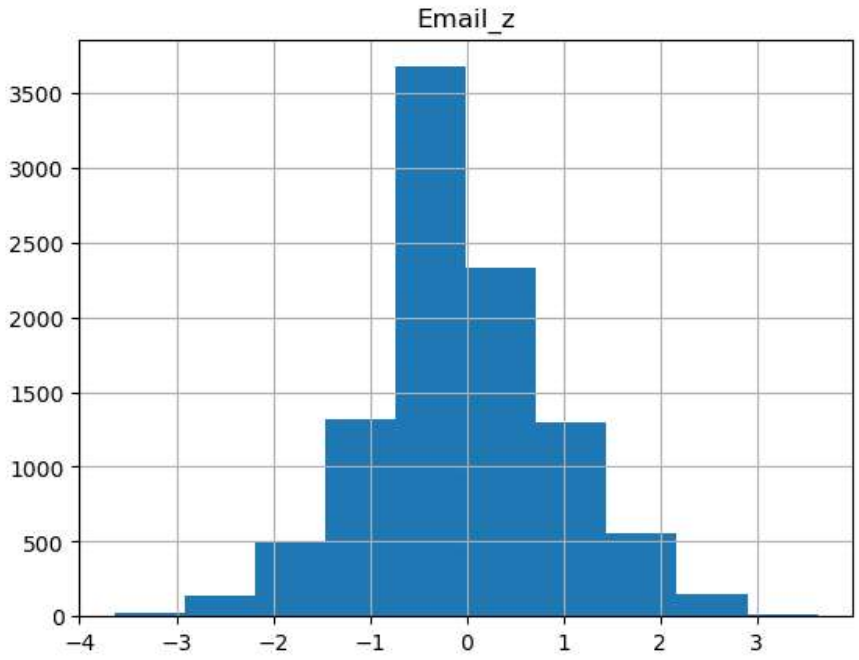
Out[9]:
```
Unnamed: 0              int64
CaseOrder               int64
Customer_id            object
Interaction            object
City                   object
State                  object
County                 object
Zip                     int64
Lat                   float64
Lng                   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
Timely_response         int64
Timely_fixes            int64
Timely_replacements     int64
Reliability             int64
options                 int64
Respectful_response     int64
Courteous_exchange      int64
Active_listening        int64
dtype: object
```

######### Step 4: Identify outliers in certain columns

######### Detecting outliers in Email column code

In [10]:
```
data_frame = pd.read_csv(r'C:\Users\richa\OneDrive\Desktop\d206\churn_raw_data.csv')

data_frame['Email_z'] = stats.zscore(data_frame['Email'])
graph = pd.DataFrame({"Email_z":data_frame.loc[:,"Email_z"]})

graph.hist()
```

Out[10]:
```
array([[<Axes: title={'center': 'Email_z'}>]], dtype=object)
```



In [11]:

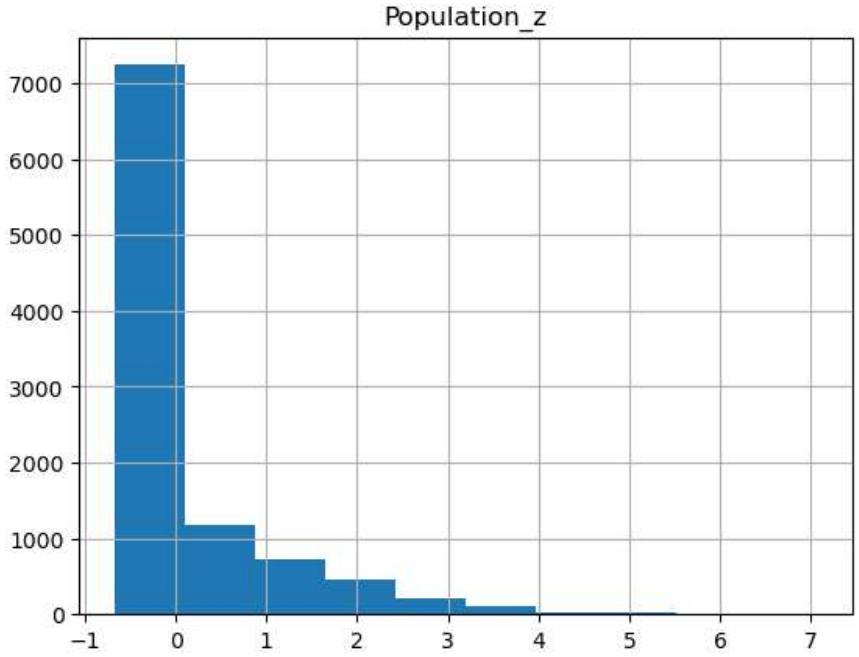
```
Email_z = (df['Email'] - df['Email'].mean()) /df['Email'].std()
data = df.loc[(Email_z > 3) | (Email_z < -3)]
print(data)
```

```
      Unnamed: 0  CaseOrder Customer_id                          Interaction  \
795          796        796     V397454  61317680-6fcb-46e3-a3ae-6dfb5d74652d
1152        1153       1153     B328912  0e6351f8-e91b-44c4-b4d1-1876ec047b91
1381        1382       1382     A626819  9f969c90-52dd-4575-8a73-b8f3a5d6baea
1399        1400       1400     P897691  97c7cf1b-1c78-4a7d-8c2c-cd528f65218d
1473        1474       1474      N49031  6d4104a2-bb45-4995-b0a5-680df77c70a8
1746        1747       1747     Z756585  3d6df3c6-daef-4920-90aa-50aa240d0125
6320        6321       6321     R392539  4a979f3a-2bcd-4d61-9eb7-d8730e06b03d
7408        7409       7409     X451865  ba32dc79-a135-4360-b13d-ecbd6d989181
8365        8366       8366     Q926339  f58d5572-c1f9-4092-b363-f07fbd2446c1
8948        8949       8949     I301776  ebafa68f-e3b4-4dfc-96e8-46630f8ec00d
9248        9249       9249     N508508  afe055dd-0435-4554-be03-45d6fbc5c914
9475        9476       9476     U102287  b454d911-062c-483f-a4ea-35e84fd6c45e

             City State       County    Zip       Lat        Lng  ...  \
795       Readlyn    IA       Bremer  50668  42.69069  -92.22652  ...
1152       Galena    MO        Stone  65656  36.79505  -93.46672  ...
1381         Anna    TX       Collin  75409  33.35197  -96.52735  ...
1399    Arlington    IL       Bureau  61312  41.43573  -89.23293  ...
1473      Houston    TX       Harris  77098  29.73487  -95.41529  ...
1746     Carthage    TX       Panola  75633  32.12895  -94.27705  ...
6320  Noblesville    IN     Hamilton  46062  40.06175  -86.05543  ...
7408      Trinway    OH    Muskingum  43842  40.13683  -82.01275  ...
8365       Powell    TX      Navarro  75153  32.15027  -96.32857  ...
8948     Glendale    CA  Los Angeles  91204  34.13636 -118.26100  ...
9248    Estherwood    LA       Acadia  70534  30.19491  -92.44206  ...
9475         Belva    WV     Nicholas  26656  38.26455  -81.16352  ...

      MonthlyCharge Bandwidth_GB_Year Timely_response Timely_fixes  \
795      173.077144        829.270710               3            4
1152     153.442884       2021.782055               1            3
1381     130.540061       3192.755309               3            2
1399     185.970819        627.396266               4            4
1473     192.986294       1257.976073               5            4
1746     189.523622       1606.847228               3            3
6320     149.470600       4598.720000               3            4
7408     174.094100       4835.664000               5            5
8365     168.357900       6000.122000               4            3
8948     257.679500       7015.440000               4            5
9248     118.479000       5970.386000               3            4
9475     209.037500               NaN               2            2

      Timely_replacements  Reliability options Respectful_response  \
795                     3            4       3                   2
1152                    1            3       4                   3
1381                    2            3       3                   3
1399                    4            2       3                   4
1473                    5            3       3                   5
1746                    4            5       3                   3
6320                    3            3       3                   2
7408                    4            4       3                   5
8365                    4            2       5                   3
8948                    3            3       2                   5
9248                    2            3       2                   3
9475                    3            4       4                   2

      Courteous_exchange Active_listening
795                    3                1
1152                   2                2
1381                   3                3
1399                   4                4
1473                   5                3
1746                   3                4
6320                   3                3
7408                   7                5
8365                   4                4
8948                   4                4
9248                   3                4
9475                   2                3

[12 rows x 52 columns]
```

########## Detecting outliers in Population column code

In [12]:
```
data_frame['Population_z'] = stats.zscore(data_frame['Population'])
graph = pd.DataFrame({"Population_z":data_frame.loc[:,"Population_z"]})
graph.hist()
```

Out[12]:
```
array([[<Axes: title={'center': 'Population_z'}>]], dtype=object)
```



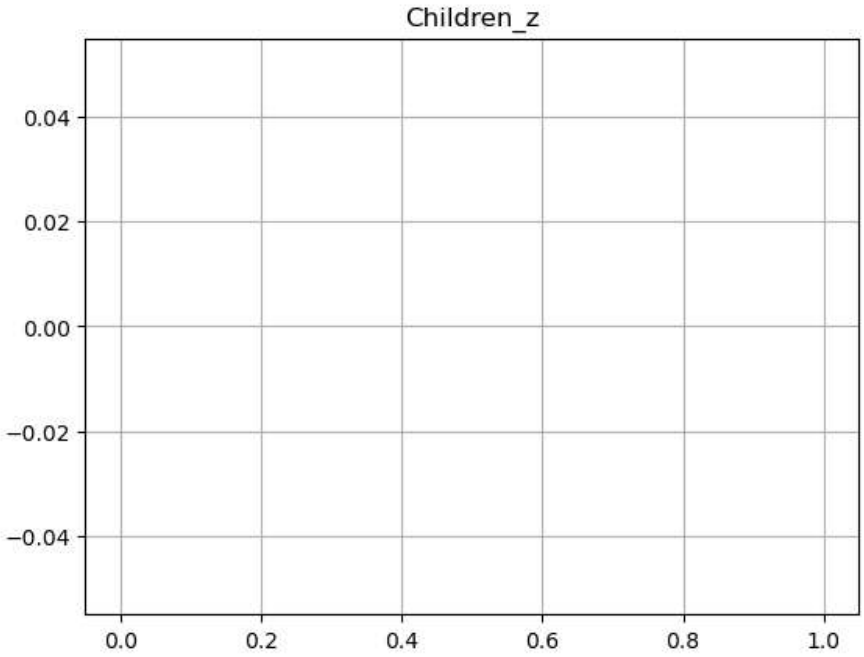

In [13]:
```
Population_z = (df['Population'] - df['Population'].mean()) /df['Population'].std()
data = df.loc[(Population_z > 3) | (Population_z < -3)]
print(data)
```

```
9728                    5             2       5                    3
9905                    3             3       5                    3
9987                    3             5       3                    3
9996                    5             4       4                    5

      Courteous_exchange Active_listening
57                     4                5
90                     3                3
100                    4                3
157                    3                5
203                    4                3
...                  ...              ...
9647                   3                3
9728                   4                4
9905                   3                3
9987                   3                3
9996                   2                5

[219 rows x 52 columns]
```

########## Detecting outliers in Children column code

In [14]:
```
data_frame['Children_z'] = stats.zscore(data_frame['Children'])
graph = pd.DataFrame({"Children_z":data_frame.loc[:,"Children_z"]})
graph.hist()
```

Out[14]:
```
array([[<Axes: title={'center': 'Children_z'}>]], dtype=object)
```



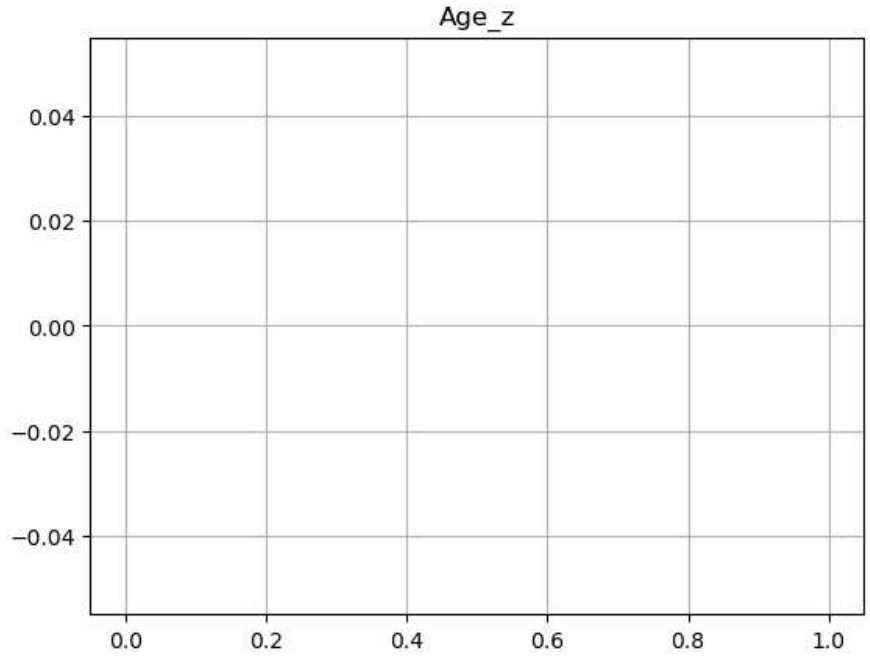
In [15]:
```
Children_z = (df['Children'] - df['Children'].mean()) /df['Children'].std()
data = df.loc[(Children_z > 3) | (Children_z < -3)]
print(data)
```

```
      Unnamed: 0  CaseOrder Customer_id                            Interaction  \
30            31         31     L357432  79b827eb-46b9-4737-8484-0b670171bc4b
97            98         98     N417817  832e769e-3b90-47cc-8f70-12e77fbd52a2
144          145        145     Q541158  f67c47f8-93da-4b90-babe-4cd9b9deebf6
329          330        330     T369132  bad7853a-c59b-4775-ad6d-05ff1d14c4b4
334          335        335     O402306  c8ffb29f-7ee4-4883-8698-f6936607cd04
...          ...        ...         ...                                   ...
9599        9600       9600     J345464  b1e94ff4-ff0c-49b8-9a9d-8065ddb10cc8
9623        9624       9624     Q831134  5b4ad141-b338-4638-9b10-1d0e2ce826b5
9790        9791       9791     U741276  5c3198d1-4138-4ba8-b0b9-652748f0c127
9871        9872       9872     F573675  71c2c800-8cee-434b-b60a-efa9ecbacf38
9901        9902       9902     O774002  569131f4-71f4-4e59-9176-e4f8f7b7bbe1

                  City State       County    Zip       Lat        Lng  ...  \
30          Whitesboro    NY       Oneida  13492  43.11988  -75.32875  ...
97         Beaverville    IL     Iroquois  60912  40.97003  -87.59935  ...
144         Saint Paul    MN       Ramsey  55107  44.93099  -93.07957  ...
329          Maryville    TN       Blount  37801  35.66815  -84.08767  ...
334             Salmon    ID        Lemhi  83467  45.00975 -113.89910  ...
...                ...   ...          ...    ...       ...        ...  ...
9599          Lovejoy    IL    St. Clair  62059  38.65652  -90.16503  ...
9623    Granada Hills    CA  Los Angeles  91344  34.29392 -118.50750  ...
9790  Colorado Springs    CO      El Paso  80930  38.81433 -104.50240  ...
9871           Holton    IN       Ripley  47023  39.07935  -85.38150  ...
9901           Kaaawa    HI     Honolulu  96730  21.54614 -157.85110  ...

      MonthlyCharge Bandwidth_GB_Year Timely_response Timely_fixes  \
30       230.964891       1795.465729              3            2
97       188.611913       1956.733700              3            2
144      136.870564        819.419527              4            4
329      221.913604       1973.661316              4            3
334      127.617949       1451.201108              3            2
...             ...               ...            ...          ...
9599     130.854100       4968.246000              4            4
9623     173.170700       6432.294000              4            5
9790     243.035600       6900.075000              3            4
9871     144.145200               NaN              3            3
9901     223.945600       6680.275000              4            4

      Timely_replacements  Reliability options Respectful_response  \
30                      1            5       3                   3
97                      2            4       3                   2
144                     3            3       2                   3
329                     4            2       5                   3
334                     2            3       4                   4
...                   ...          ...     ...                 ...
9599                    4            5       2                   6
9623                    5            3       5                   3
9790                    4            3       4                   4
9871                    3            3       4                   3
9901                    4            4       3                   4

      Courteous_exchange Active_listening
30                     3                2
97                     3                2
144                    4                2
329                    4                3
334                    4                5
...                  ...              ...
9599                   4                4
9623                   4                2
9790                   4                2
9871                   4                4
9901                   3                4

[144 rows x 52 columns]
```

############Detecting outliers in Age column code

In [16]:
```
data_frame['Age_z'] = stats.zscore(data_frame['Age'])
graph = pd.DataFrame({"Age_z":data_frame.loc[:,"Age_z"]})
graph.hist()
```

Out[16]:
```
array([[<Axes: title={'center': 'Age_z'}>]], dtype=object)
```




In [17]:
```
Age_z = (df['Age'] - df['Age'].mean()) /df['Age'].std()
data = df.loc[(Age_z > 3) | (Age_z < -3)]
print(data)
```

```
Empty DataFrame
Columns: [Unnamed: 0, CaseOrder, Customer_id, Interaction, City, State, County, Zip, Lat, Lng, Populati
on, Area, Timezone, Job, Children, Age, Education, Employment, Income, Marital, Gender, Churn, Outage_s
ec_perweek, Email, Contacts, Yearly_equip_failure, Techie, Contract, Port_modem, Tablet, InternetServic
e, Phone, Multiple, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, Streaming
Movies, PaperlessBilling, PaymentMethod, Tenure, MonthlyCharge, Bandwidth_GB_Year, Timely_response, Tim
ely_fixes, Timely_replacements, Reliability, options, Respectful_response, Courteous_exchange, Active_l
istening]
Index: []

[0 rows x 52 columns]
```

######### Detecting outliers in Income column code

In [18]:
```
data_frame['Income_z'] = stats.zscore(data_frame['Income'])
graph = pd.DataFrame({"Income_z":data_frame.loc[:,"Income_z"]})
graph.hist()
```

Out[18]:
```
array([[<Axes: title={'center': 'Income_z'}>]], dtype=object)
```

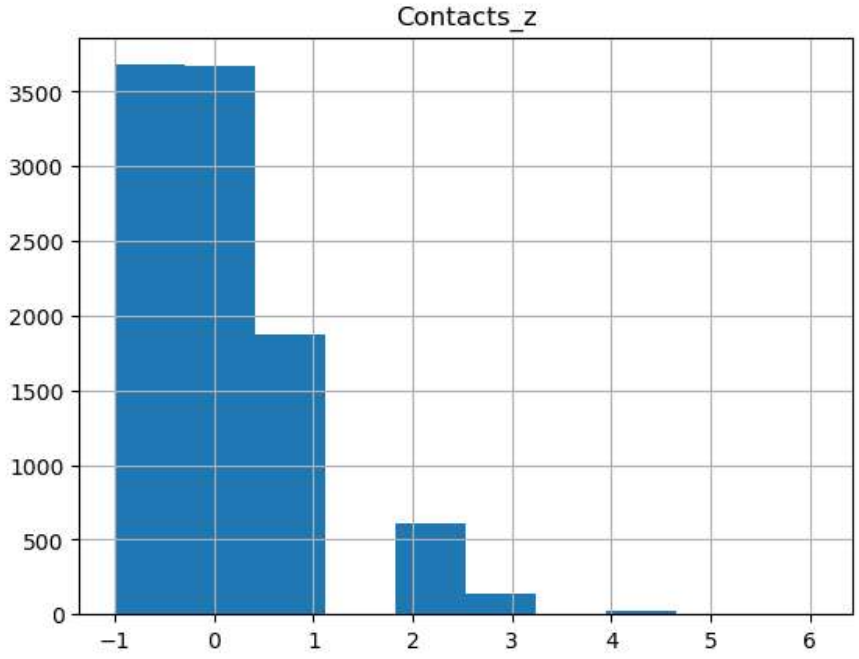
In [19]:
```
Income_z = (df['Income'] - df['Income'].mean()) /df['Income'].std()
data = df.loc[(Income_z > 3) | (Income_z < -3)]
print(data)
```

```
      Unnamed: 0  CaseOrder Customer_id                            Interaction  \
46            47         47     B609739  bb4a7c2d-6524-41d6-b2ed-f7511509ff5b
130          131        131     X645702  cfc79a87-f608-485d-86c9-cfce7db37a29
186          187        187     F501848  b47490b5-a967-4f46-a76a-600552c4d141
470          471        471     Q585011  e018df7d-ad42-41c7-9971-75ddf0a3f9ee
511          512        512     X286988  9fb14b9f-a39c-48bc-9cdf-3cad65435793
...          ...        ...         ...                                   ...
9615        9616       9616     X447899  6fb07a81-0fa0-4484-ae02-81a76506696c
9639        9640       9640     K483384  663eaa8d-1ae1-4305-814e-0abbed5c8260
9656        9657       9657     T234307  8ce025a0-86b1-4e33-9c97-f9c8408e7689
9849        9850       9850     C406267  5fae818b-cdcb-452e-bb5a-c9dc04ed9383
9876        9877       9877     T156561  34f05b13-dc84-4732-a9ac-2e2c68a2d625

             City State        County    Zip       Lat        Lng  ...  \
46         Peoria    IL        Peoria  61606  40.69980  -89.61143  ...
130      Dennison    OH    Tuscarawas  44621  40.42752  -81.29587  ...
186     Beckville    TX        Panola  75631  32.24926  -94.45456  ...
470   Saint Marie    MT        Valley  59231  48.39806 -106.54796  ...
511       Ardmore    TN         Giles  38449  35.04820  -86.82587  ...
...           ...   ...           ...    ...       ...        ...  ...
9615   Queen City    TX          Cass  75572  33.23153  -94.13216  ...
9639     Moorhead    MS     Sunflower  38761  33.43204  -90.48489  ...
9656     Guernsey    IA     Poweshiek  52221  41.64197  -92.33228  ...
9849      Laporte    CO       Larimer  80535  40.73342 -105.18350  ...
9876   Manchester    NH  Hillsborough   3104  43.00934  -71.44146  ...

      MonthlyCharge Bandwidth_GB_Year Timely_response Timely_fixes  \
46       185.071940       2384.885909               4            4
130      168.916463       1162.332926               3            3
186      107.276898       1468.357421               3            3
470      192.216418        499.717065               3            4
511       91.947750       1216.325478               3            3
...             ...               ...             ...          ...
9615     221.875200       5702.684000               3            4
9639     217.543700       6140.576000               2            2
9656     202.963500       6603.944000               3            4
9849     143.253600       4516.883000               4            4
9876     147.456100       5343.897000               3            4

      Timely_replacements  Reliability options Respectful_response  \
46                      5            2       2                   4
130                     4            2       4                   4
186                     3            3       5                   2
470                     4            3       4                   3
511                     3            3       4                   4
...                   ...          ...     ...                 ...
9615                    2            2       5                   3
9639                    2            4       3                   3
9656                    4            2       3                   4
9849                    4            2       5                   3
9876                    4            4       4                   4

      Courteous_exchange Active_listening
46                     3                3
130                    3                2
186                    4                3
470                    2                3
511                    3                4
...                  ...              ...
9615                   4                3
9639                   3                5
9656                   4                4
9849                   3                4
9876                   5                3

[110 rows x 52 columns]
```

############## Detecting outliers in Contacts column code

In [20]:
```
data_frame['Contacts_z'] = stats.zscore(data_frame['Contacts'])
graph = pd.DataFrame({"Contacts_z":data_frame.loc[:,"Contacts_z"]})
graph.hist()
```

Out[20]:
```
array([[<Axes: title={'center': 'Contacts_z'}>]], dtype=object)
```



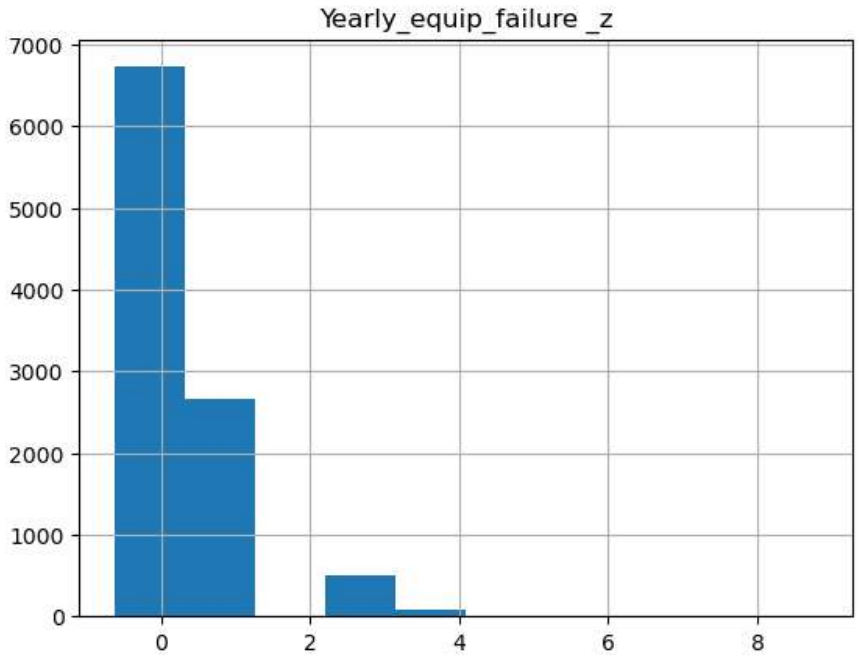
In [21]:
```
Contacts_z = (df['Contacts'] - df['Contacts'].mean()) /df['Contacts'].std()
data = df.loc[(Contacts_z > 3) | (Contacts_z < -3)]
print(data)
```

```
      Unnamed: 0  CaseOrder Customer_id                            Interaction  \
88            89         89      C73741  1ca2e70f-6207-4514-a8ce-e97244a56251
129          130        130      S985225  f3878f9d-fe73-4db7-9d66-de25fab169c1
187          188        188      Q971369  6ec3ca40-a3db-4a9d-b0c8-012950c949bd
205          206        206      Q279060  33a0892c-cd0b-4177-865c-b836af7a9a1c
345          346        346      O511762  3a0cf6f8-3aa9-4262-b5a0-e9c7788cbac5
...          ...        ...          ...                                   ...
9799        9800       9800      D32957  774a53fc-547a-4402-ad49-72abe33914b1
9805        9806       9806      G824974  a0de52e5-421f-4cd1-b98e-9d1e39945f06
9828        9829       9829      W814496  e576dc49-9d95-4f8e-9d77-734e901e7424
9923        9924       9924      I173343  0deb3f09-647c-467b-94c0-a621350323c8
9972        9973       9973      F952565  559568ad-0d48-4ebb-9bc8-f1a50230a1c2

              City State        County    Zip       Lat        Lng  ...  \
88       Waterbury    CT     New Haven   6704  41.58650  -73.03305  ...
129        Momence    IL      Kankakee  60954  41.15338  -87.62902  ...
187    Barnesville    PA    Schuylkill  18214  40.79879  -76.08148  ...
205         Cosmos    MN        Meeker  56228  44.93688  -94.67707  ...
345   Arnoldsville    GA    Oglethorpe  30619  33.85751  -83.24105  ...
...            ...   ...           ...    ...       ...        ...  ...
9799        Athens    AL     Limestone  35614  34.86542  -87.08385  ...
9805       Hyampom    CA       Trinity  96046  40.59969 -123.42900  ...
9828       Donegal    PA  Westmoreland  15628  40.10054  -79.37343  ...
9923      Caliente    NV       Lincoln  89008  37.27637 -114.54630  ...
9972      Danville    CA  Contra Costa  94506  37.80812 -121.90630  ...

      MonthlyCharge Bandwidth_GB_Year Timely_response Timely_fixes  \
88       187.621573       1393.202392               4            4
129      219.277619               NaN               2            1
187      152.404442        590.776580               1            3
205      240.472505       2059.200998               2            3
345      232.889666       1718.600163               6            6
...             ...               ...             ...          ...
9799     154.112200       5253.832000               3            3
9805     207.797200       5773.173000               3            4
9828     159.899700       4486.583000               3            3
9923     132.042700       5582.271000               4            3
9972     142.823500       5706.972000               4            4

      Timely_replacements  Reliability options Respectful_response  \
88                      4            3       3                   3
129                     2            4       3                   5
187                     2            3       4                   3
205                     3            4       3                   4
345                     5            3       5                   4
...                   ...          ...     ...                 ...
9799                    2            5       3                   3
9805                    2            2       3                   3
9828                    3            4       4                   2
9923                    4            2       4                   4
9972                    4            5       2                   5

      Courteous_exchange Active_listening
88                     3                5
129                    3                1
187                    3                3
205                    5                4
345                    4                4
...                  ...              ...
9799                   4                2
9805                   4                3
9828                   2                3
9923                   4                2
9972                   4                5

[165 rows x 52 columns]
```

################## Detecting outliers in Yearly_equip_failure column code

In [22]:
```
data_frame['Yearly_equip_failure _z'] = stats.zscore(data_frame['Yearly_equip_failure'])
graph = pd.DataFrame({"Yearly_equip_failure _z":data_frame.loc[:,"Yearly_equip_failure _z"]})
graph.hist()
```

Out[22]:
```
array([[<Axes: title={'center': 'Yearly_equip_failure _z'}>]],
      dtype=object)
```



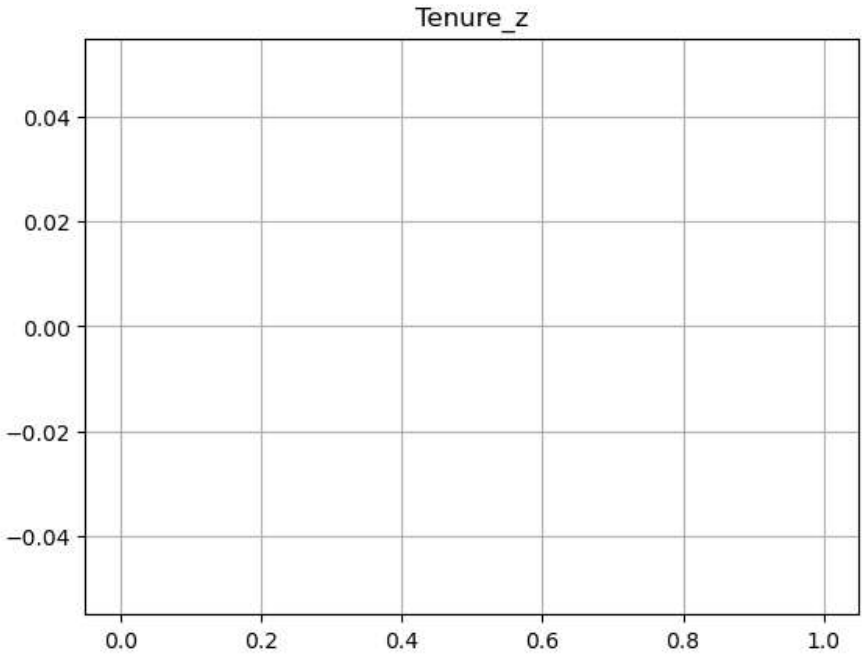
In [23]:
```
Yearly_equip_failure_z = (df['Yearly_equip_failure'] - df['Yearly_equip_failure'].mean()) /df['Yearly_eq
data = df.loc[(Yearly_equip_failure_z > 3) | (Yearly_equip_failure_z < -3)]
print(data)
```

```
      Unnamed: 0  CaseOrder Customer_id                            Interaction  \
8              9          9     M716771  05a49ee3-8fd5-453a-a5f3-82b6cd986856
20            21         21     X325271  ebf7fbd7-9f65-48d8-8f82-b7b7fd4f3412
171          172        172     P954849  7ec593ca-82bd-4a03-8359-ab20212e3ce7
592          593        593     X987521  c09f56b6-1b05-46ea-b33c-8d9fed4dea5c
621          622        622     S636055  bba47549-f3bb-46d7-b4f3-1c4effa75b3b
...          ...        ...         ...                                   ...
9623        9624       9624     Q831134  5b4ad141-b338-4638-9b10-1d0e2ce826b5
9674        9675       9675     H712105  0a489672-b274-4fb8-9e82-9705e4bd9933
9763        9764       9764     A472924  68f06d83-dca3-473e-9543-3246da5c941d
9769        9770       9770     W133136  24a71b97-d47a-4ab6-b0a0-3c3cf30257dd
9967        9968       9968     Q597995  17ead91e-fee8-4b36-bec3-60610ced2732

               City State       County    Zip       Lat        Lng  ...  \
8       Saint Cloud    FL      Osceola  34771  28.27646  -81.16273  ...
20        Kaneville    IL         Kane  60144  41.83594  -88.52060  ...
171   Valley Stream    NY       Nassau  11580  40.67495  -73.70356  ...
592     Westerville    OH     Franklin  43081  40.11064  -82.89134  ...
621         Marlton    NJ   Burlington   8053  39.86049  -74.89466  ...
...             ...   ...          ...    ...       ...        ...  ...
9623  Granada Hills    CA  Los Angeles  91344  34.29392 -118.50750  ...
9674        Jackson    MI      Jackson  49203  42.22151  -84.40110  ...
9763       Snelling    CA       Merced  95369  37.53346 -120.43170  ...
9769        Buffalo    NY         Erie  14207  42.95167  -78.89778  ...
9967      Lynnfield    MA        Essex   1940  42.53456  -71.03757  ...

      MonthlyCharge Bandwidth_GB_Year Timely_response Timely_fixes  \
8        118.366844       1312.874964               5            4
20       185.096264       2330.319383               2            3
171      161.083510       1522.289512               2            3
592      206.930518       1795.270512               2            3
621      163.417357               NaN               3            2
...             ...               ...             ...          ...
9623     173.170700       6432.294000               4            5
9674     194.028400       5222.665000               6            3
9763     172.004700       4849.092000               3            3
9769     275.342100       5890.998000               2            2
9967     209.167500       6524.466000               1            1

      Timely_replacements  Reliability options Respectful_response  \
8                       4            3       4                   3
20                      3            2       4                   2
171                     3            5       4                   3
592                     2            3       4                   2
621                     3            3       4                   3
...                   ...          ...     ...                 ...
9623                    5            3       5                   3
9674                    4            4       2                   3
9763                    3            2       5                   5
9769                    2            5       2                   4
9967                    2            2       5                   3

      Courteous_exchange Active_listening
8                      4                4
20                     3                3
171                    2                3
592                    1                3
621                    1                4
...                  ...              ...
9623                   4                2
9674                   5                3
9763                   4                4
9769                   3                3
9967                   1                3

[94 rows x 52 columns]
```

################## Detecting outliers in Tenure column code

In [24]:
```
data_frame['Tenure_z'] = stats.zscore(data_frame['Tenure'])
graph = pd.DataFrame({"Tenure_z":data_frame.loc[:,"Tenure_z"]})
graph.hist()
```

Out[24]:
```
array([[<Axes: title={'center': 'Tenure_z'}>]], dtype=object)
```



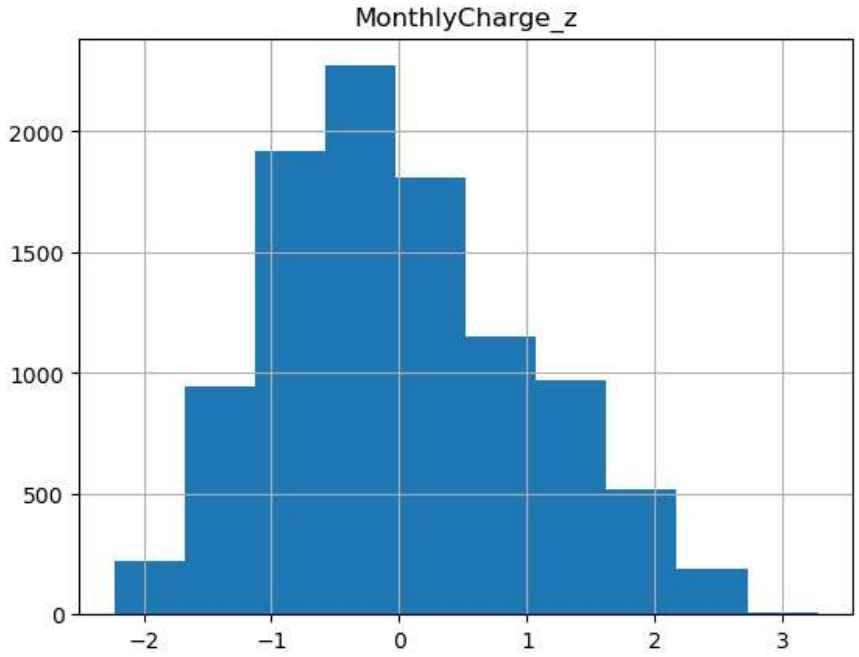

In [25]:
```
Tenure_z = (df['Tenure'] - df['Tenure'].mean()) /df['Tenure'].std()
data = df.loc[(Tenure_z > 3) | (Tenure_z < -3)]
print(data)
```

```
Empty DataFrame
Columns: [Unnamed: 0, CaseOrder, Customer_id, Interaction, City, State, County, Zip, Lat, Lng, Populati
on, Area, Timezone, Job, Children, Age, Education, Employment, Income, Marital, Gender, Churn, Outage_s
ec_perweek, Email, Contacts, Yearly_equip_failure, Techie, Contract, Port_modem, Tablet, InternetServic
e, Phone, Multiple, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, Streaming
Movies, PaperlessBilling, PaymentMethod, Tenure, MonthlyCharge, Bandwidth_GB_Year, Timely_response, Tim
ely_fixes, Timely_replacements, Reliability, options, Respectful_response, Courteous_exchange, Active_l
istening]
Index: []

[0 rows x 52 columns]
```

###################### Detecting outliers in MonthlyCharge column code

In [26]:
```
data_frame['MonthlyCharge_z'] = stats.zscore(data_frame['MonthlyCharge'])
graph = pd.DataFrame({"MonthlyCharge_z":data_frame.loc[:,"MonthlyCharge_z"]})
graph.hist()
```

Out[26]:
```
array([[<Axes: title={'center': 'MonthlyCharge_z'}>]], dtype=object)
```



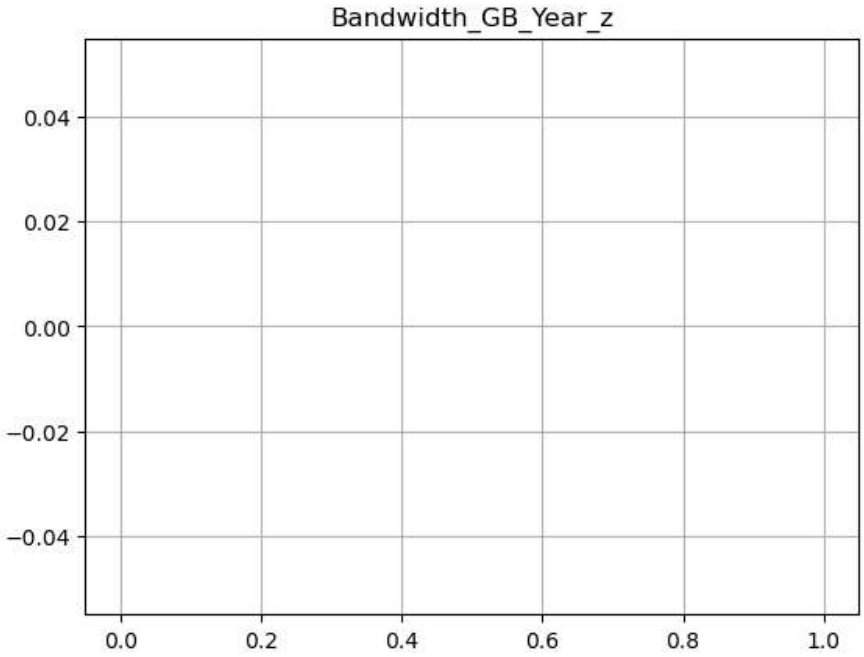

In [27]:
```
MonthlyCharge_z = (df['MonthlyCharge'] - df['MonthlyCharge'].mean()) /df['MonthlyCharge'].std()
data = df.loc[(MonthlyCharge_z > 3) | (MonthlyCharge_z < -3)]
print(data)
```

```
      Unnamed: 0  CaseOrder Customer_id                            Interaction  \
927          928        928     O479736  fa6578cd-243a-498d-9a20-8fd5a133c982
3746        3747       3747     Q642279  869dcfbe-d65e-45f7-844c-736ae3426677
4700        4701       4701     I250439  c9a55402-8498-4b7f-97ad-7f2bee5eece3

             City State                 County    Zip       Lat        Lng  \
927        Mendon    UT                  Cache  84325  41.72579 -111.99279
3746  Point Baker    AK  Prince of Wales-Hyder  99927  56.25100 -133.37570
4700        Inkom    ID                Bannock  83245  42.81817 -112.22040

      ...  MonthlyCharge Bandwidth_GB_Year Timely_response Timely_fixes  \
927   ...     307.528124       1482.558282               4            4
3746  ...     315.878600       1288.595000               4            3
4700  ...     306.268000       1383.764000               2            3

      Timely_replacements  Reliability options Respectful_response  \
927                     4            5       2                   4
3746                    5            3       4                   4
4700                    3            3       4                   1

      Courteous_exchange Active_listening
927                    2                2
3746                   4                4
4700                   2                3

[3 rows x 52 columns]
```

############ Detecting outliers in Bandwidth_GB_Year outlier column code

In [28]:
```
data_frame['Bandwidth_GB_Year_z'] = stats.zscore(data_frame['Bandwidth_GB_Year'])
graph = pd.DataFrame({"Bandwidth_GB_Year_z":data_frame.loc[:,"Bandwidth_GB_Year_z"]})
graph.hist()
```

Out[28]:
```
array([[<Axes: title={'center': 'Bandwidth_GB_Year_z'}>]], dtype=object)
```

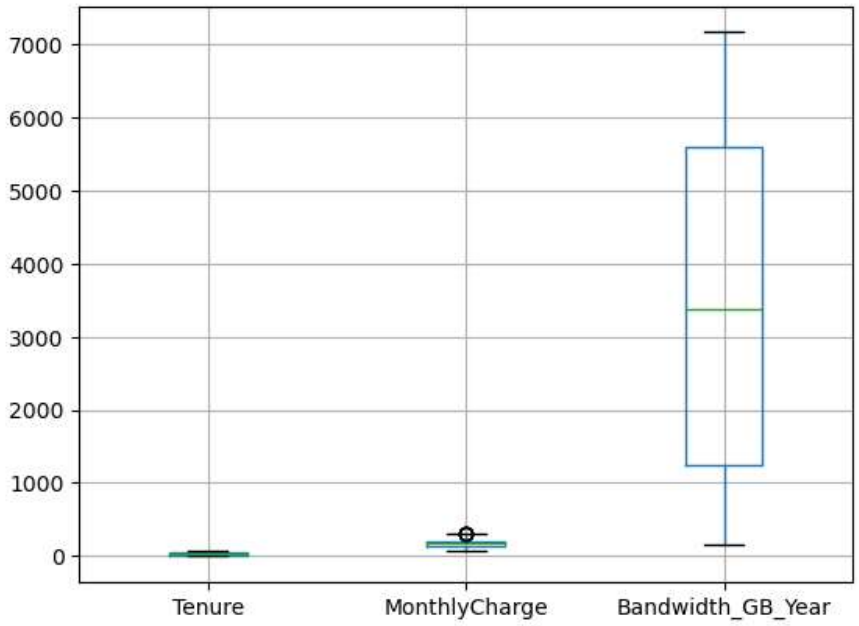



In [29]:
```
Bandwidth_GB_Year_z = (df['Bandwidth_GB_Year'] - df['Bandwidth_GB_Year'].mean()) /df['Bandwidth_GB_Year'
data = df.loc[(Bandwidth_GB_Year_z > 3) | (Bandwidth_GB_Year_z < -3)]
print(data)
```

```
Empty DataFrame
Columns: [Unnamed: 0, CaseOrder, Customer_id, Interaction, City, State, County, Zip, Lat, Lng, Populati
on, Area, Timezone, Job, Children, Age, Education, Employment, Income, Marital, Gender, Churn, Outage_s
ec_perweek, Email, Contacts, Yearly_equip_failure, Techie, Contract, Port_modem, Tablet, InternetServic
e, Phone, Multiple, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, Streaming
Movies, PaperlessBilling, PaymentMethod, Tenure, MonthlyCharge, Bandwidth_GB_Year, Timely_response, Tim
ely_fixes, Timely_replacements, Reliability, options, Respectful_response, Courteous_exchange, Active_l
istening]
Index: []

[0 rows x 52 columns]
```

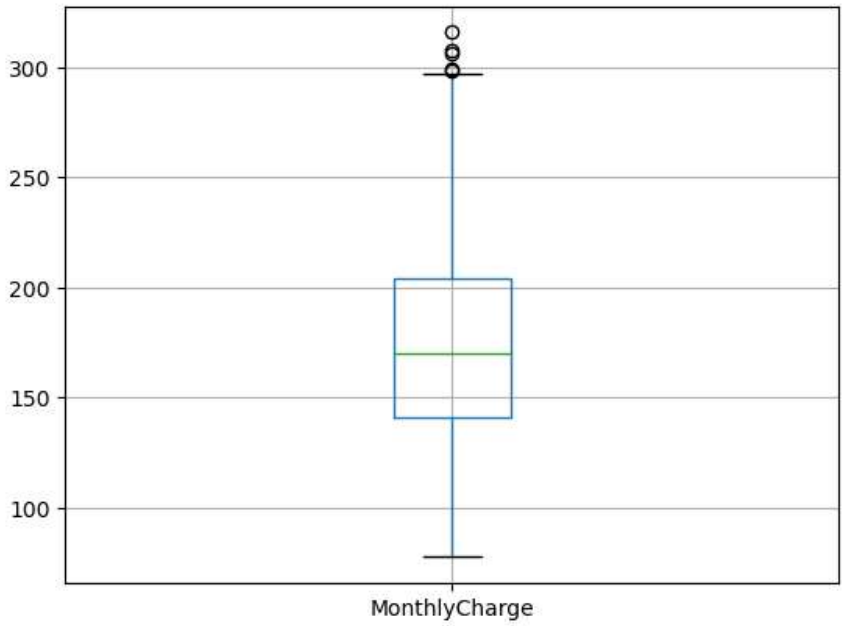
###############let us generate box plots for outliers analysis for Tenure, MonthlyCharge, Bandwidth_GB_Year columns

In [30]:
```
df.boxplot(['Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year'])
plt.savefig('df_boxplots.jpg')
```




################## Let us box plot the MonthlyCharge seperately

In [31]:
```
df.boxplot(['MonthlyCharge'])
plt.savefig('df_boxplots.jpg')
```




Step 5: ############## Locate missing values in the data set and impute these missing values

Find missing values

In [32]:
```
df.isnull()
```

Out[32]:

| | Unnamed: 0 | CaseOrder | Customer_id | Interaction | City | State | County | Zip | Lat | Lng | ... | MonthlyCharge | Bandwidth_G |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | False | False | False | False | False | False | False | False | False | False | ... | False | |
| **1** | False | False | False | False | False | False | False | False | False | False | ... | False | |
| **2** | False | False | False | False | False | False | False | False | False | False | ... | False | |
| **3** | False | False | False | False | False | False | False | False | False | False | ... | False | |
| **4** | False | False | False | False | False | False | False | False | False | False | ... | False | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **9995** | False | False | False | False | False | False | False | False | False | False | ... | False | |
| **9996** | False | False | False | False | False | False | False | False | False | False | ... | False | |
| **9997** | False | False | False | False | False | False | False | False | False | False | ... | False | |
| **9998** | False | False | False | False | False | False | False | False | False | False | ... | False | |
| **9999** | False | False | False | False | False | False | False | False | False | False | ... | False | |

10000 rows × 52 columns

Locate rows that contain missing values

In [33]:
```
df.isnull().any(axis=1)
```

Out[33]:
```
0        True
1       False
2        True
3       False
4       False
        ...
9995     True
9996     True
9997     True
9998    False
9999     True
Length: 10000, dtype: bool
```

Display the columns with missing values

In [34]:
```
df.isna().any()
```

Out[34]:
```
Unnamed: 0             False
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
Timely_response        False
Timely_fixes           False
Timely_replacements    False
Reliability            False
options                False
Respectful_response    False
Courteous_exchange     False
Active_listening       False
dtype: bool
```

################ Count the number of missing values in columns

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

```
Unnamed: 0                0
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
Timely_response           0
Timely_fixes              0
Timely_replacements       0
Reliability               0
options                   0
Respectful_response       0
Courteous_exchange        0
Active_listening          0
dtype: int64
```

################ Impute the missing values for the variables Children, Age, Income, Tenure and Bandwidth_GB_Year with the median and the variables Techie, Phone, TechSupport with "No"

In [36]:

```
df['Children'].fillna(0, inplace=True)
df['Age'].fillna(df['Age'].median(), inplace=True)
df['Income'].fillna(df['Income'].median(), inplace=True)
df['Bandwidth_GB_Year'].fillna(df['Bandwidth_GB_Year'].median(), inplace=True)
df['Tenure'].fillna(df['Tenure'].median(), inplace=True)
df['Techie'].fillna('No', inplace=True)
df['Phone'].fillna('No', inplace=True)
df['TechSupport'].fillna('No', inplace=True)
print(df)
```

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

              City State                 County    Zip       Lat        Lng  \
0      Point Baker    AK  Prince of Wales-Hyder  99927  56.25100 -133.37571
1      West Branch    MI                Ogemaw  48661  44.32893  -84.24080
2          Yamhill    OR               Yamhill  97148  45.35589 -123.24657
3          Del Mar    CA             San Diego  92014  32.96687 -117.24798
4        Needville    TX             Fort Bend  77461  29.38012  -95.80673
...            ...   ...                   ...    ...       ...        ...
9995   Mount Holly    VT               Rutland   5758  43.43391  -72.78734
9996   Clarksville    TN            Montgomery  37042  36.56907  -87.41694
9997      Mobeetie    TX               Wheeler  79061  35.52039 -100.44180
9998    Carrollton    GA               Carroll  30117  33.58016  -85.13241
9999  Clarkesville    GA              Habersham  30523  34.70783  -83.53648

      ...  MonthlyCharge Bandwidth_GB_Year Timely_response Timely_fixes  \
0     ...     171.449762        904.536110               5            5
1     ...     242.948015        800.982766               3            4
2     ...     159.440398       2054.706961               4            4
3     ...     120.249493       2164.579412               4            4
4     ...     150.761216        271.493436               4            4
...   ...            ...               ...             ...          ...
9995  ...     159.828800       6511.253000               3            2
9996  ...     208.856400       5695.952000               4            5
9997  ...     168.220900       4159.306000               4            4
9998  ...     252.628600       6468.457000               4            4
9999  ...     218.371000       5857.586000               2            2

      Timely_replacements  Reliability options Respectful_response  \
0                       5            3       4                   4
1                       3            3       4                   3
2                       2            4       4                   3
3                       4            2       5                   4
4                       4            3       4                   4
...                   ...          ...     ...                 ...
9995                    3            3       4                   3
9996                    5            4       4                   5
9997                    4            4       4                   4
9998                    6            4       3                   3
9999                    3            3       3                   3

      Courteous_exchange Active_listening
0                      3                4
1                      4                4
2                      3                3
3                      3                3
4                      4                5
...                  ...              ...
9995                   2                3
9996                   2                5
9997                   4                5
9998                   5                4
9999                   4                1

[10000 rows x 52 columns]
```

In [37]:
```
data_nulls = df.isnull().sum()
print(data_nulls)
```

```
Unnamed: 0              0
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                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                  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
Timely_response         0
Timely_fixes            0
Timely_replacements     0
Reliability             0
options                 0
Respectful_response     0
Courteous_exchange      0
Active_listening        0
dtype: int64
```

Step 6: Detecting and removing duplicate values in rows

In [38]:
```
data_duplicates = df.loc[df.duplicated()]
print(data_duplicates)
```

```
Empty DataFrame
Columns: [Unnamed: 0, CaseOrder, Customer_id, Interaction, City, State, County, Zip, Lat, Lng, Populati
on, Area, Timezone, Job, Children, Age, Education, Employment, Income, Marital, Gender, Churn, Outage_s
ec_perweek, Email, Contacts, Yearly_equip_failure, Techie, Contract, Port_modem, Tablet, InternetServic
e, Phone, Multiple, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, Streaming
Movies, PaperlessBilling, PaymentMethod, Tenure, MonthlyCharge, Bandwidth_GB_Year, Timely_response, Tim
ely_fixes, Timely_replacements, Reliability, options, Respectful_response, Courteous_exchange, Active_l
istening]
Index: []

[0 rows x 52 columns]
```

############### Step 7: Export the cleaned data

In [39]:
```
df.to_csv(r'C:\Users\richa\OneDrive\Desktop\d206\cleaned_churn_data.csv', index=False, header=True)
print(df)
```

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

              City State                 County    Zip       Lat        Lng  \
0      Point Baker    AK  Prince of Wales-Hyder  99927  56.25100 -133.37571
1      West Branch    MI                 Ogemaw  48661  44.32893  -84.24080
2          Yamhill    OR               Yamhill  97148  45.35589 -123.24657
3          Del Mar    CA             San Diego  92014  32.96687 -117.24798
4        Needville    TX             Fort Bend  77461  29.38012  -95.80673
...            ...   ...                   ...    ...       ...        ...
9995   Mount Holly    VT               Rutland   5758  43.43391  -72.78734
9996   Clarksville    TN            Montgomery  37042  36.56907  -87.41694
9997      Mobeetie    TX               Wheeler  79061  35.52039 -100.44180
9998    Carrollton    GA               Carroll  30117  33.58016  -85.13241
9999  Clarkesville    GA             Habersham  30523  34.70783  -83.53648

      ...  MonthlyCharge Bandwidth_GB_Year Timely_response Timely_fixes  \
0     ...     171.449762        904.536110               5            5
1     ...     242.948015        800.982766               3            4
2     ...     159.440398       2054.706961               4            4
3     ...     120.249493       2164.579412               4            4
4     ...     150.761216        271.493436               4            4
...   ...            ...               ...             ...          ...
9995  ...     159.828800       6511.253000               3            2
9996  ...     208.856400       5695.952000               4            5
9997  ...     168.220900       4159.306000               4            4
9998  ...     252.628600       6468.457000               4            4
9999  ...     218.371000       5857.586000               2            2

      Timely_replacements  Reliability options Respectful_response  \
0                       5            3       4                   4
1                       3            3       4                   3
2                       2            4       4                   3
3                       4            2       5                   4
4                       4            3       4                   4
...                   ...          ...     ...                 ...
9995                    3            3       4                   3
9996                    5            4       4                   5
9997                    4            4       4                   4
9998                    6            4       3                   3
9999                    3            3       3                   3

      Courteous_exchange Active_listening
0                      3                4
1                      4                4
2                      3                3
3                      3                3
4                      4                5
...                  ...              ...
9995                   2                3
9996                   2                5
9997                   4                5
9998                   5                4
9999                   4                1

[10000 rows x 52 columns]
```

############E- Principal Component Analysis

############## We reload the clean data set

In [40]:
```
df = pd.read_csv(r'C:\Users\richa\OneDrive\Desktop\d206\cleaned_churn_data.csv')
print(df)
```

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

              City State                 County    Zip       Lat        Lng  \
0      Point Baker    AK  Prince of Wales-Hyder  99927  56.25100 -133.37571
1      West Branch    MI                 Ogemaw  48661  44.32893  -84.24080
2          Yamhill    OR               Yamhill  97148  45.35589 -123.24657
3          Del Mar    CA             San Diego  92014  32.96687 -117.24798
4        Needville    TX             Fort Bend  77461  29.38012  -95.80673
...            ...   ...                   ...    ...       ...        ...
9995   Mount Holly    VT               Rutland   5758  43.43391  -72.78734
9996   Clarksville    TN            Montgomery  37042  36.56907  -87.41694
9997      Mobeetie    TX               Wheeler  79061  35.52039 -100.44180
9998    Carrollton    GA               Carroll  30117  33.58016  -85.13241
9999  Clarkesville    GA              Habersham  30523  34.70783  -83.53648

      ...  MonthlyCharge Bandwidth_GB_Year Timely_response Timely_fixes  \
0     ...     171.449762        904.536110               5            5
1     ...     242.948015        800.982766               3            4
2     ...     159.440398       2054.706961               4            4
3     ...     120.249493       2164.579412               4            4
4     ...     150.761216        271.493436               4            4
...   ...            ...               ...             ...          ...
9995  ...     159.828800       6511.253000               3            2
9996  ...     208.856400       5695.952000               4            5
9997  ...     168.220900       4159.306000               4            4
9998  ...     252.628600       6468.457000               4            4
9999  ...     218.371000       5857.586000               2            2

      Timely_replacements  Reliability options Respectful_response  \
0                       5            3       4                   4
1                       3            3       4                   3
2                       2            4       4                   3
3                       4            2       5                   4
4                       4            3       4                   4
...                   ...          ...     ...                 ...
9995                    3            3       4                   3
9996                    5            4       4                   5
9997                    4            4       4                   4
9998                    6            4       3                   3
9999                    3            3       3                   3

      Courteous_exchange Active_listening
0                      3                4
1                      4                4
2                      3                3
3                      3                3
4                      4                5
...                  ...              ...
9995                   2                3
9996                   2                5
9997                   4                5
9998                   5                4
9999                   4                1

[10000 rows x 52 columns]
```

# We selected the last eleven service related columns

In [41]:
```
data = df.loc[:, 'Tenure':'Active_listening']

print(data)
```

```
         Tenure  MonthlyCharge  Bandwidth_GB_Year  Timely_response  \
0      6.795513     171.449762         904.536110                5
1      1.156681     242.948015         800.982766                3
2     15.754144     159.440398        2054.706961                4
3     17.087227     120.249493        2164.579412                4
4      1.670972     150.761216         271.493436                4
...         ...            ...                ...              ...
9995  68.197130     159.828800        6511.253000                3
9996  61.040370     208.856400        5695.952000                4
9997  36.196030     168.220900        4159.306000                4
9998  71.095600     252.628600        6468.457000                4
9999  63.350860     218.371000        5857.586000                2

      Timely_fixes  Timely_replacements  Reliability  options  \
0                5                    5            3        4
1                4                    3            3        4
2                4                    2            4        4
3                4                    4            2        5
4                4                    4            3        4
...            ...                  ...          ...      ...
9995             2                    3            3        4
9996             5                    5            4        4
9997             4                    4            4        4
9998             4                    6            4        3
9999             2                    3            3        3

      Respectful_response  Courteous_exchange  Active_listening
0                       4                   3                 4
1                       3                   4                 4
2                       3                   3                 3
3                       4                   3                 3
4                       4                   4                 5
...                   ...                 ...               ...
9995                    3                   2                 3
9996                    5                   2                 5
9997                    4                   4                 5
9998                    3                   5                 4
9999                    3                   4                 1

[10000 rows x 11 columns]
```

############ We Normalize the data

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

########### We select number of components to extract

In [43]:
```
pca = PCA(n_components = data.shape[1])
```

####### We create a list of PCA names

In [44]:
```
churn_numeric = data[['Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year', 'Timely_response',
                      'Timely_fixes', 'Timely_replacements', 'Reliability', 'options',
                      'Respectful_response', 'Courteous_exchange', 'Active_listening']]
pcs_names = []
for i, col in enumerate(churn_numeric.columns):
    pcs_names.append('PC' + str(i + 1))
print(pcs_names)
```

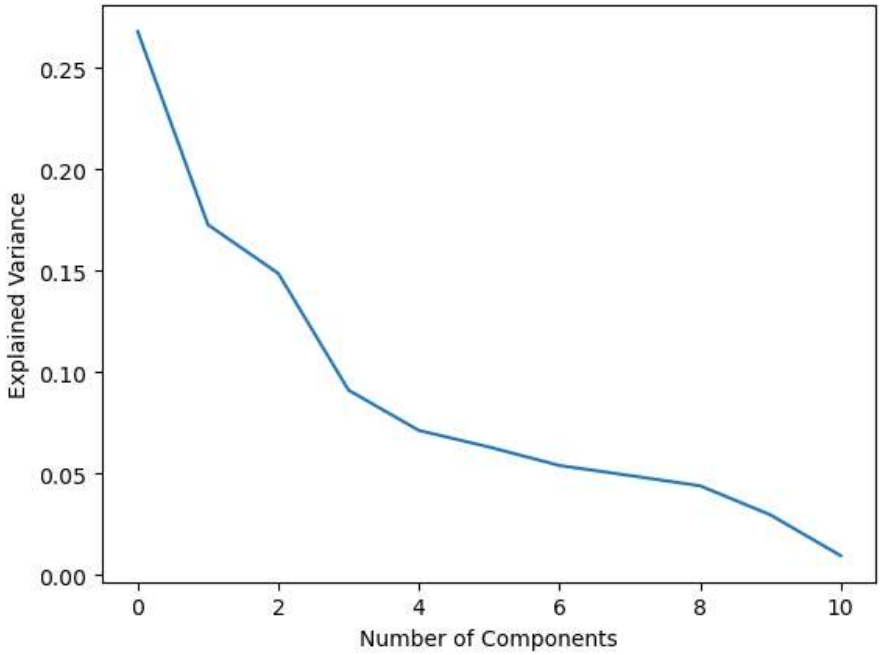
```
['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10', 'PC11']
```

***We Call PCA application and convert the dataset of 11 variables into a dataset of 11 components***

In [45]:
```
pca.fit(churn_normalized)
churn_pca = pd.DataFrame(pca.transform(churn_normalized),
columns = pcs_names)
```

######### We run the scree plot

In [46]:
```
plt.plot(pca.explained_variance_ratio_)
plt.xlabel('Number of Components')
plt.ylabel('Explained Variance')
plt.show();
```

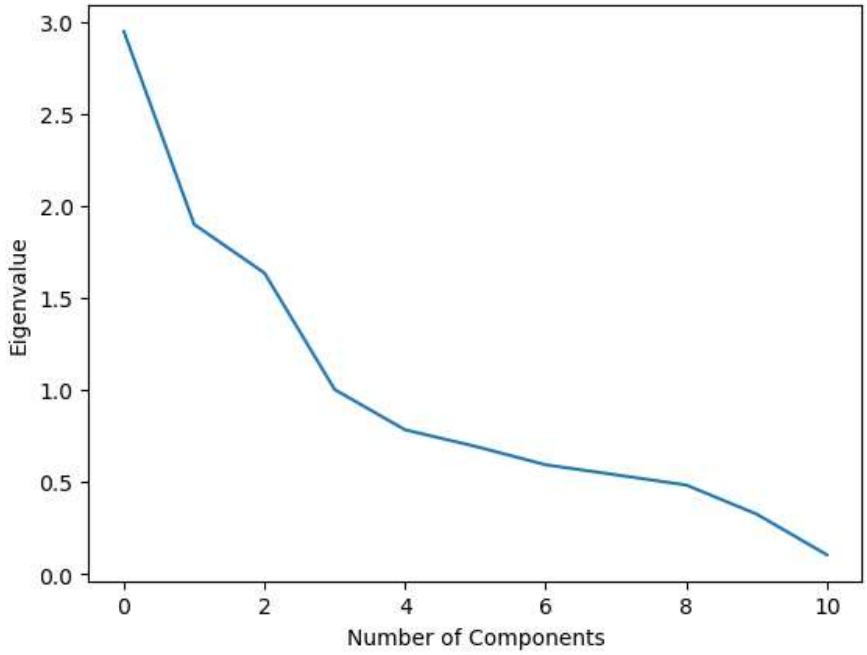



# Extract the eigenvalues

In [47]:
```
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_
```

# Plot the eigenvalues

In [48]:
```
plt.plot(eigenvalues)
plt.xlabel('Number of Components')
plt.ylabel('Eigenvalue')
plt.show();
```




## Select the fewest components

In [49]:
```
for pc, var in zip(pcs_names, np.cumsum(pca.explained_variance_ratio_)):
    print(pc, var)
```

```
PC1 0.2679266062396307
PC2 0.4405373288022937
PC3 0.5891444522248597
PC4 0.6801579842425086
PC5 0.75132170543725
PC6 0.814311709410132
PC7 0.8681970531672318
PC8 0.9171387362588831
PC9 0.9610122820002189
PC10 0.9905736363144834
PC11 1.0000000000000002
```

## Create a rotation

In [50]:
```
rotation = pd.DataFrame(pca.components_.T, columns = pcs_names, index = churn_numeric.columns)
print(rotation)
```

```
                          PC1       PC2       PC3       PC4       PC5  \
Tenure              -0.010403  0.701838 -0.072209 -0.063594  0.005683
MonthlyCharge        0.000317  0.041147 -0.014151  0.996995 -0.022136
Bandwidth_GB_Year   -0.012166  0.703079 -0.074222  0.004399  0.009590
Timely_response      0.458932  0.031325  0.281154  0.018568 -0.070233
Timely_fixes         0.434134  0.042559  0.282404  0.007508 -0.106632
Timely_replacements  0.400639  0.034665  0.281118 -0.019631 -0.173742
Reliability          0.145799 -0.050367 -0.567815 -0.010310 -0.171334
options             -0.175633  0.066334  0.587335 -0.000047  0.135949
Respectful_response  0.405207 -0.012680 -0.183447  0.004596 -0.062342
Courteous_exchange   0.358342 -0.003886 -0.181697 -0.027959 -0.182406
Active_listening     0.308925 -0.017396 -0.131173  0.015574  0.931612

                          PC6       PC7       PC8       PC9      PC10  \
Tenure              -0.011155  0.007419 -0.011527  0.006935  0.003286
MonthlyCharge        0.015231 -0.018038 -0.004316  0.023690 -0.013785
Bandwidth_GB_Year    0.003466  0.003701 -0.002364 -0.008068  0.008529
Timely_response     -0.119149 -0.045963  0.025431 -0.240574  0.793237
Timely_fixes        -0.169752 -0.065414  0.074400 -0.592131 -0.573832
Timely_replacements -0.255336 -0.146887 -0.396333  0.673088 -0.177665
Reliability         -0.483328 -0.443353  0.431528  0.087207  0.018301
options              0.060124 -0.209767  0.693861  0.265474 -0.042012
Respectful_response  0.064609  0.757954  0.402835  0.230319 -0.063972
Courteous_exchange   0.806166 -0.379136  0.067889  0.067293 -0.040946
Active_listening    -0.011133 -0.113297 -0.045132  0.046107 -0.042251

                         PC11
Tenure              -0.705445
MonthlyCharge       -0.047865
Bandwidth_GB_Year    0.706925
Timely_response     -0.004306
Timely_fixes        -0.002217
Timely_replacements  0.014933
Reliability          0.002283
options             -0.002514
Respectful_response  0.001604
Courteous_exchange  -0.006875
Active_listening    -0.002357
```

######### We generated output loadings for components

In [51]:
```
loadings = pd.DataFrame(pca.components_.T,
                        columns = pcs_names,
                        index = data.columns)
loadings
```

Out[51]:

| | PC1 | PC2 | PC3 | PC4 | PC5 | PC6 | PC7 | PC8 | PC9 | PC10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Tenure** | -0.010403 | 0.701838 | -0.072209 | -0.063594 | 0.005683 | -0.011155 | 0.007419 | -0.011527 | 0.006935 | 0.003286 |
| **MonthlyCharge** | 0.000317 | 0.041147 | -0.014151 | 0.996995 | -0.022136 | 0.015231 | -0.018038 | -0.004316 | 0.023690 | -0.013785 |
| **Bandwidth_GB_Year** | -0.012166 | 0.703079 | -0.074222 | 0.004399 | 0.009590 | 0.003466 | 0.003701 | -0.002364 | -0.008068 | 0.008529 |
| **Timely_response** | 0.458932 | 0.031325 | 0.281154 | 0.018568 | -0.070233 | -0.119149 | -0.045963 | 0.025431 | -0.240574 | 0.793237 |
| **Timely_fixes** | 0.434134 | 0.042559 | 0.282404 | 0.007508 | -0.106632 | -0.169752 | -0.065414 | 0.074400 | -0.592131 | -0.573832 |
| **Timely_replacements** | 0.400639 | 0.034665 | 0.281118 | -0.019631 | -0.173742 | -0.255336 | -0.146887 | -0.396333 | 0.673088 | -0.177665 |
| **Reliability** | 0.145799 | -0.050367 | -0.567815 | -0.010310 | -0.171334 | -0.483328 | -0.443353 | 0.431528 | 0.087207 | 0.018301 |
| **options** | -0.175633 | 0.066334 | 0.587335 | -0.000047 | 0.135949 | 0.060124 | -0.209767 | 0.693861 | 0.265474 | -0.042012 |
| **Respectful_response** | 0.405207 | -0.012680 | -0.183447 | 0.004596 | -0.062342 | 0.064609 | 0.757954 | 0.402835 | 0.230319 | -0.063972 |
| **Courteous_exchange** | 0.358342 | -0.003886 | -0.181697 | -0.027959 | -0.182406 | 0.806166 | -0.379136 | 0.067889 | 0.067293 | -0.040946 |
| **Active_listening** | 0.308925 | -0.017396 | -0.131173 | 0.015574 | 0.931612 | -0.011133 | -0.113297 | -0.045132 | 0.046107 | -0.042251 |

# Extract reduced dataset & print 3 components

In [52]:
```
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 [ ]:
```

```