# D206 Performance Assessment

**Student:** Chris Fischer **ID:** 011933891 **Dataset:** Churn ---

## ## Initialization

### Import libraries

#import needed libraries  
import numpy as np  
import pandas as pd  
import missingno as msno  
import matplotlib.pyplot as plt  
from sklearn.decomposition import PCA

### Set Pandas options

#allow pd.describe to show all columns of the data frame [In-Text Citation: (Ray, 2020)]  
pd.options.display.max\_columns = None

### Initialize some static arrays

numerical\_variables = ['Age','Income','Children','Tenure','Bandwidth\_GB\_Year','Outage\_sec\_perweek','Email','Contacts','Yearly\_equip\_failure','MonthlyCharge']  
categorical\_variables = ['Education','Employment','Marital','Gender','InternetService','PaymentMethod']  
yes\_no\_variables = ['Churn','Techie','Port\_modem','Tablet','Phone','Multiple','OnlineSecurity','OnlineBackup','DeviceProtection','TechSupport','StreamingTV','StreamingMovies','PaperlessBilling']  
survey\_variables = ['item1','item2','item3','item4','item5','item6','item7','item8',]  
survey\_answers = [1,2,3,4,5,6,7,8]

### Load raw data from CSV

#load church\_missing\_data.csv  
raw\_data = pd.read\_csv('churn\_raw\_data.csv')

#Examine the data frame [In-Text Citation: (Patanam, 2015)]  
raw\_data.info()  
raw\_data.describe(include='all')

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 10000 entries, 0 to 9999  
Data columns (total 52 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Unnamed: 0 10000 non-null int64   
 1 CaseOrder 10000 non-null int64   
 2 Customer\_id 10000 non-null object   
 3 Interaction 10000 non-null object   
 4 City 10000 non-null object   
 5 State 10000 non-null object   
 6 County 10000 non-null object   
 7 Zip 10000 non-null int64   
 8 Lat 10000 non-null float64  
 9 Lng 10000 non-null float64  
 10 Population 10000 non-null int64   
 11 Area 10000 non-null object   
 12 Timezone 10000 non-null object   
 13 Job 10000 non-null object   
 14 Children 7505 non-null float64  
 15 Age 7525 non-null float64  
 16 Education 10000 non-null object   
 17 Employment 10000 non-null object   
 18 Income 7510 non-null float64  
 19 Marital 10000 non-null object   
 20 Gender 10000 non-null object   
 21 Churn 10000 non-null object   
 22 Outage\_sec\_perweek 10000 non-null float64  
 23 Email 10000 non-null int64   
 24 Contacts 10000 non-null int64   
 25 Yearly\_equip\_failure 10000 non-null int64   
 26 Techie 7523 non-null object   
 27 Contract 10000 non-null object   
 28 Port\_modem 10000 non-null object   
 29 Tablet 10000 non-null object   
 30 InternetService 7871 non-null object   
 31 Phone 8974 non-null object   
 32 Multiple 10000 non-null object   
 33 OnlineSecurity 10000 non-null object   
 34 OnlineBackup 10000 non-null object   
 35 DeviceProtection 10000 non-null object   
 36 TechSupport 9009 non-null object   
 37 StreamingTV 10000 non-null object   
 38 StreamingMovies 10000 non-null object   
 39 PaperlessBilling 10000 non-null object   
 40 PaymentMethod 10000 non-null object   
 41 Tenure 9069 non-null float64  
 42 MonthlyCharge 10000 non-null float64  
 43 Bandwidth\_GB\_Year 8979 non-null float64  
 44 item1 10000 non-null int64   
 45 item2 10000 non-null int64   
 46 item3 10000 non-null int64   
 47 item4 10000 non-null int64   
 48 item5 10000 non-null int64   
 49 item6 10000 non-null int64   
 50 item7 10000 non-null int64   
 51 item8 10000 non-null int64   
dtypes: float64(9), int64(15), object(28)  
memory usage: 4.0+ MB

Unnamed: 0 CaseOrder Customer\_id \  
count 10000.00000 10000.00000 10000   
unique NaN NaN 10000   
top NaN NaN K409198   
freq NaN NaN 1   
mean 5000.50000 5000.50000 NaN   
std 2886.89568 2886.89568 NaN   
min 1.00000 1.00000 NaN   
25% 2500.75000 2500.75000 NaN   
50% 5000.50000 5000.50000 NaN   
75% 7500.25000 7500.25000 NaN   
max 10000.00000 10000.00000 NaN   
  
 Interaction City State County \  
count 10000 10000 10000 10000   
unique 10000 6058 52 1620   
top aa90260b-4141-4a24-8e36-b04ce1f4f77b Houston TX Washington   
freq 1 34 603 111   
mean NaN NaN NaN NaN   
std NaN NaN NaN NaN   
min NaN NaN NaN NaN   
25% NaN NaN NaN NaN   
50% NaN NaN NaN NaN   
75% NaN NaN NaN NaN   
max NaN NaN NaN NaN   
  
 Zip Lat Lng Population Area \  
count 10000.000000 10000.000000 10000.000000 10000.000000 10000   
unique NaN NaN NaN NaN 3   
top NaN NaN NaN NaN Suburban   
freq NaN NaN NaN NaN 3346   
mean 49153.319600 38.757567 -90.782536 9756.562400 NaN   
std 27532.196108 5.437389 15.156142 14432.698671 NaN   
min 601.000000 17.966120 -171.688150 0.000000 NaN   
25% 26292.500000 35.341828 -97.082812 738.000000 NaN   
50% 48869.500000 39.395800 -87.918800 2910.500000 NaN   
75% 71866.500000 42.106908 -80.088745 13168.000000 NaN   
max 99929.000000 70.640660 -65.667850 111850.000000 NaN   
  
 Timezone Job Children Age \  
count 10000 10000 7505.000000 7525.000000   
unique 25 639 NaN NaN   
top America/New\_York Occupational psychologist NaN NaN   
freq 4072 30 NaN NaN   
mean NaN NaN 2.095936 53.275748   
std NaN NaN 2.154758 20.753928   
min NaN NaN 0.000000 18.000000   
25% NaN NaN 0.000000 35.000000   
50% NaN NaN 1.000000 53.000000   
75% NaN NaN 3.000000 71.000000   
max NaN NaN 10.000000 89.000000   
  
 Education Employment Income Marital \  
count 10000 10000 7510.000000 10000   
unique 12 5 NaN 5   
top Regular High School Diploma Full Time NaN Divorced   
freq 2421 5992 NaN 2092   
mean NaN NaN 39936.762226 NaN   
std NaN NaN 28358.469482 NaN   
min NaN NaN 740.660000 NaN   
25% NaN NaN 19285.522500 NaN   
50% NaN NaN 33186.785000 NaN   
75% NaN NaN 53472.395000 NaN   
max NaN NaN 258900.700000 NaN   
  
 Gender Churn Outage\_sec\_perweek Email Contacts \  
count 10000 10000 10000.000000 10000.000000 10000.000000   
unique 3 2 NaN NaN NaN   
top Female No NaN NaN NaN   
freq 5025 7350 NaN NaN NaN   
mean NaN NaN 11.452955 12.016000 0.994200   
std NaN NaN 7.025921 3.025898 0.988466   
min NaN NaN -1.348571 1.000000 0.000000   
25% NaN NaN 8.054362 10.000000 0.000000   
50% NaN NaN 10.202896 12.000000 1.000000   
75% NaN NaN 12.487644 14.000000 2.000000   
max NaN NaN 47.049280 23.000000 7.000000   
  
 Yearly\_equip\_failure Techie Contract Port\_modem Tablet \  
count 10000.000000 7523 10000 10000 10000   
unique NaN 2 3 2 2   
top NaN No Month-to-month No No   
freq NaN 6266 5456 5166 7009   
mean 0.398000 NaN NaN NaN NaN   
std 0.635953 NaN NaN NaN NaN   
min 0.000000 NaN NaN NaN NaN   
25% 0.000000 NaN NaN NaN NaN   
50% 0.000000 NaN NaN NaN NaN   
75% 1.000000 NaN NaN NaN NaN   
max 6.000000 NaN NaN NaN NaN   
  
 InternetService Phone Multiple OnlineSecurity OnlineBackup \  
count 7871 8974 10000 10000 10000   
unique 2 2 2 2 2   
top Fiber Optic Yes No No No   
freq 4408 8128 5392 6424 5494   
mean NaN NaN NaN NaN NaN   
std NaN NaN NaN NaN NaN   
min NaN NaN NaN NaN NaN   
25% NaN NaN NaN NaN NaN   
50% NaN NaN NaN NaN NaN   
75% NaN NaN NaN NaN NaN   
max NaN NaN NaN NaN NaN   
  
 DeviceProtection TechSupport StreamingTV StreamingMovies \  
count 10000 9009 10000 10000   
unique 2 2 2 2   
top No No No No   
freq 5614 5635 5071 5110   
mean NaN NaN NaN NaN   
std NaN NaN NaN NaN   
min NaN NaN NaN NaN   
25% NaN NaN NaN NaN   
50% NaN NaN NaN NaN   
75% NaN NaN NaN NaN   
max NaN NaN NaN NaN   
  
 PaperlessBilling PaymentMethod Tenure MonthlyCharge \  
count 10000 10000 9069.000000 10000.000000   
unique 2 4 NaN NaN   
top Yes Electronic Check NaN NaN   
freq 5882 3398 NaN NaN   
mean NaN NaN 34.498858 174.076305   
std NaN NaN 26.438904 43.335473   
min NaN NaN 1.000259 77.505230   
25% NaN NaN 7.890442 141.071078   
50% NaN NaN 36.196030 169.915400   
75% NaN NaN 61.426670 203.777441   
max NaN NaN 71.999280 315.878600   
  
 Bandwidth\_GB\_Year item1 item2 item3 \  
count 8979.000000 10000.000000 10000.000000 10000.000000   
unique NaN NaN NaN NaN   
top NaN NaN NaN NaN   
freq NaN NaN NaN NaN   
mean 3398.842752 3.490800 3.505100 3.487000   
std 2187.396807 1.037797 1.034641 1.027977   
min 155.506715 1.000000 1.000000 1.000000   
25% 1234.110529 3.000000 3.000000 3.000000   
50% 3382.424000 3.000000 4.000000 3.000000   
75% 5587.096500 4.000000 4.000000 4.000000   
max 7158.982000 7.000000 7.000000 8.000000   
  
 item4 item5 item6 item7 item8   
count 10000.000000 10000.000000 10000.000000 10000.000000 10000.000000   
unique NaN NaN NaN NaN NaN   
top NaN NaN NaN NaN NaN   
freq NaN NaN NaN NaN NaN   
mean 3.497500 3.492900 3.497300 3.509500 3.495600   
std 1.025816 1.024819 1.033586 1.028502 1.028633   
min 1.000000 1.000000 1.000000 1.000000 1.000000   
25% 3.000000 3.000000 3.000000 3.000000 3.000000   
50% 3.000000 3.000000 3.000000 4.000000 3.000000   
75% 4.000000 4.000000 4.000000 4.000000 4.000000   
max 7.000000 7.000000 8.000000 7.000000 8.000000

## ## Detection (C4)

### Detect any duplicate variable names

#Find the number of duplicated variable names  
#Formatting code [In-Text Citation:(Sanchhaya Education, Pvt Ltd, 2023)]  
print('Number of duplicated variables : {}'.format(raw\_data.columns.duplicated().sum()))

Number of duplicated variables : 0

### Detect duplicate rows and ID values

#Count duplicated rows [In-Text Citiation: (Zach, 2022)]  
#Formatting code [In-Text Citation:(Sanchhaya Education, Pvt Ltd, 2023)]  
print('Duplicate rows : {}'.format(len(raw\_data)-len(raw\_data.drop\_duplicates())))

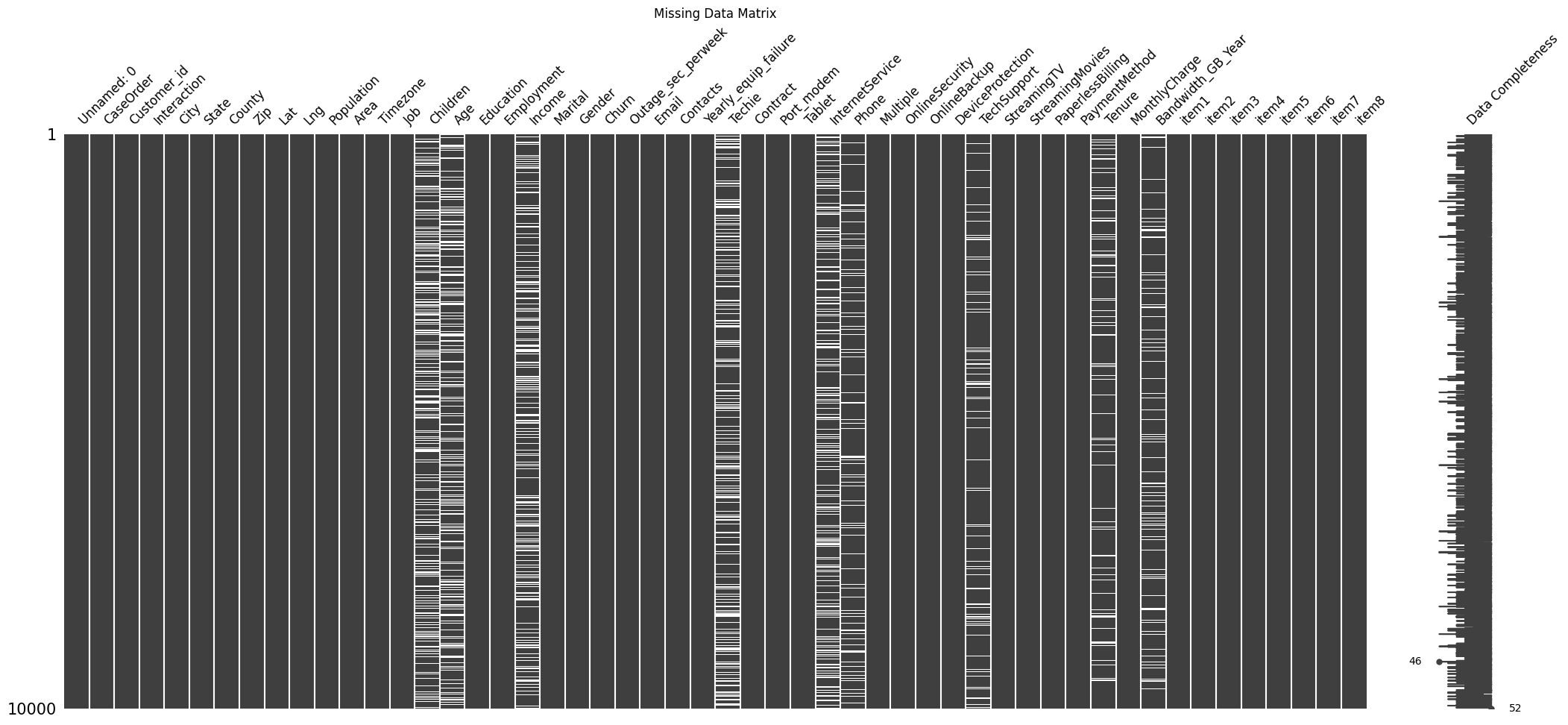
Duplicate rows : 0

#Count duplicate Customer\_id and Interaction values [In-Text Citiation: (Zach, 2022)]  
#Formatting code [In-Text Citation:(Sanchhaya Education, Pvt Ltd, 2023)]  
print('Duplicate Customer\_ids : {}'.format(len(raw\_data['Customer\_id'])-len(raw\_data['Customer\_id'].drop\_duplicates())))   
print('Duplicate Interactions : {}'.format(len(raw\_data['Interaction'])-len(raw\_data['Interaction'].drop\_duplicates())))

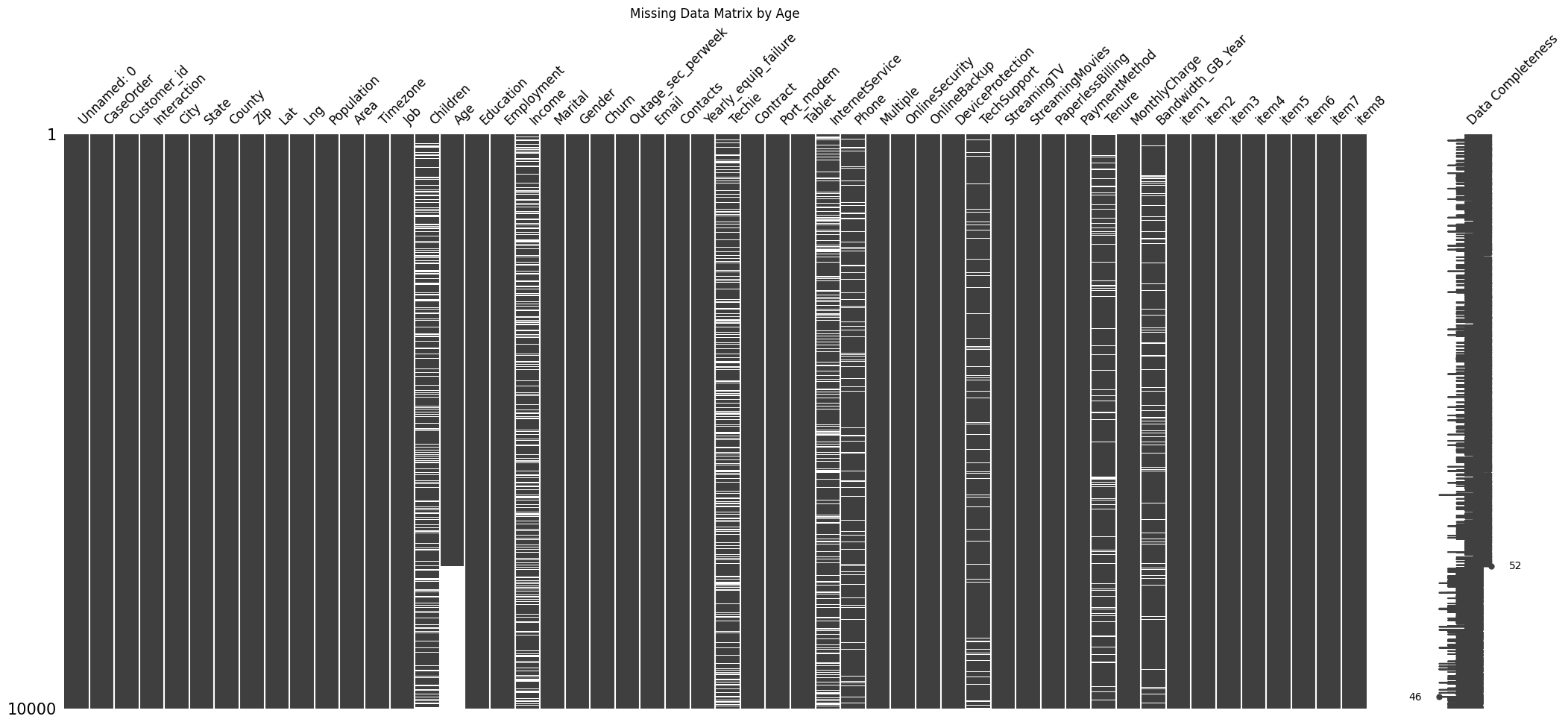
Duplicate Customer\_ids : 0  
Duplicate Interactions : 0

### Detect and evaluate missing data

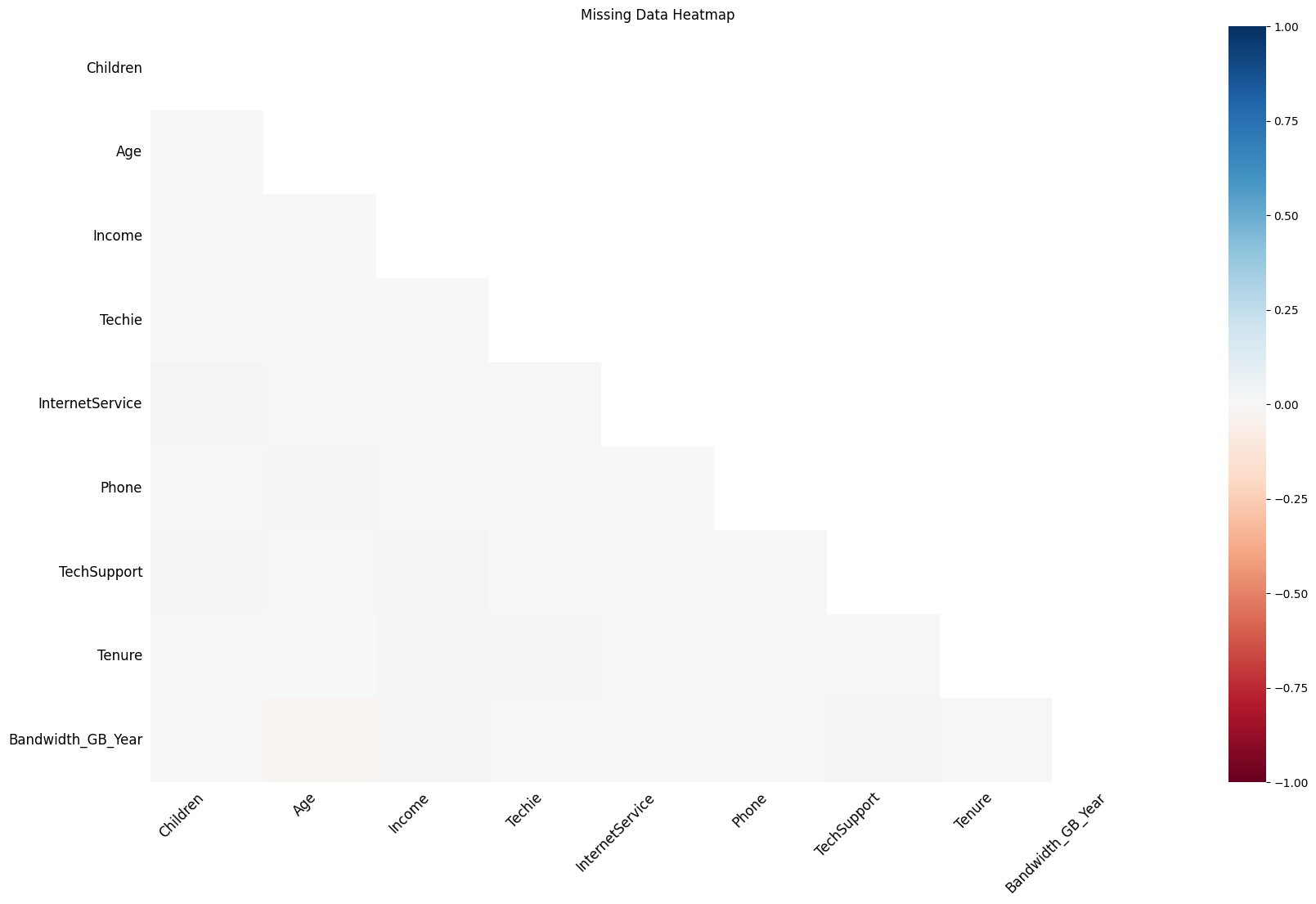
#Print the missing data matrix [In-Text Citation:(Middleton, Getting Started with Detecting and Treating Missing Values, 2023)]  
msno.matrix(raw\_data, fontsize = 12, labels=True)  
plt.title('Missing Data Matrix')  
plt.show()



#Sort the data by Age to see if there is a correlation of missingness  
#between Age, Children, Income [In-Text Citation:(Middleton, Getting Started with Detecting and Treating Missing Values, 2023)]  
msno.matrix(raw\_data.sort\_values(by='Age'), fontsize = 12, labels=True)  
plt.title('Missing Data Matrix by Age')  
plt.show()



# Display a heatmap to show any correlation between missing columns  
msno.heatmap(raw\_data, fontsize = 12, labels=True)  
plt.title('Missing Data Heatmap')  
plt.show()



#List any features with missing values [In-Text Citation: (Uzunov, 2016)]  
missing\_data = raw\_data[raw\_data.columns[raw\_data.isnull().any()]].copy()  
print(len(missing\_data.columns)) # How many variables have missing data?  
print(missing\_data.isnull().sum()) # How many missing values in each variable?

9  
Children 2495  
Age 2475  
Income 2490  
Techie 2477  
InternetService 2129  
Phone 1026  
TechSupport 991  
Tenure 931  
Bandwidth\_GB\_Year 1021  
dtype: int64

### Detect value errors in survey questions

#Check values in survey questions [In-Text Citation: (Guar, 2019)]  
print(raw\_data[survey\_variables][~raw\_data[survey\_variables].isin(survey\_answers)].count())

item1 0  
item2 0  
item3 0  
item4 0  
item5 0  
item6 0  
item7 0  
item8 0  
dtype: int64

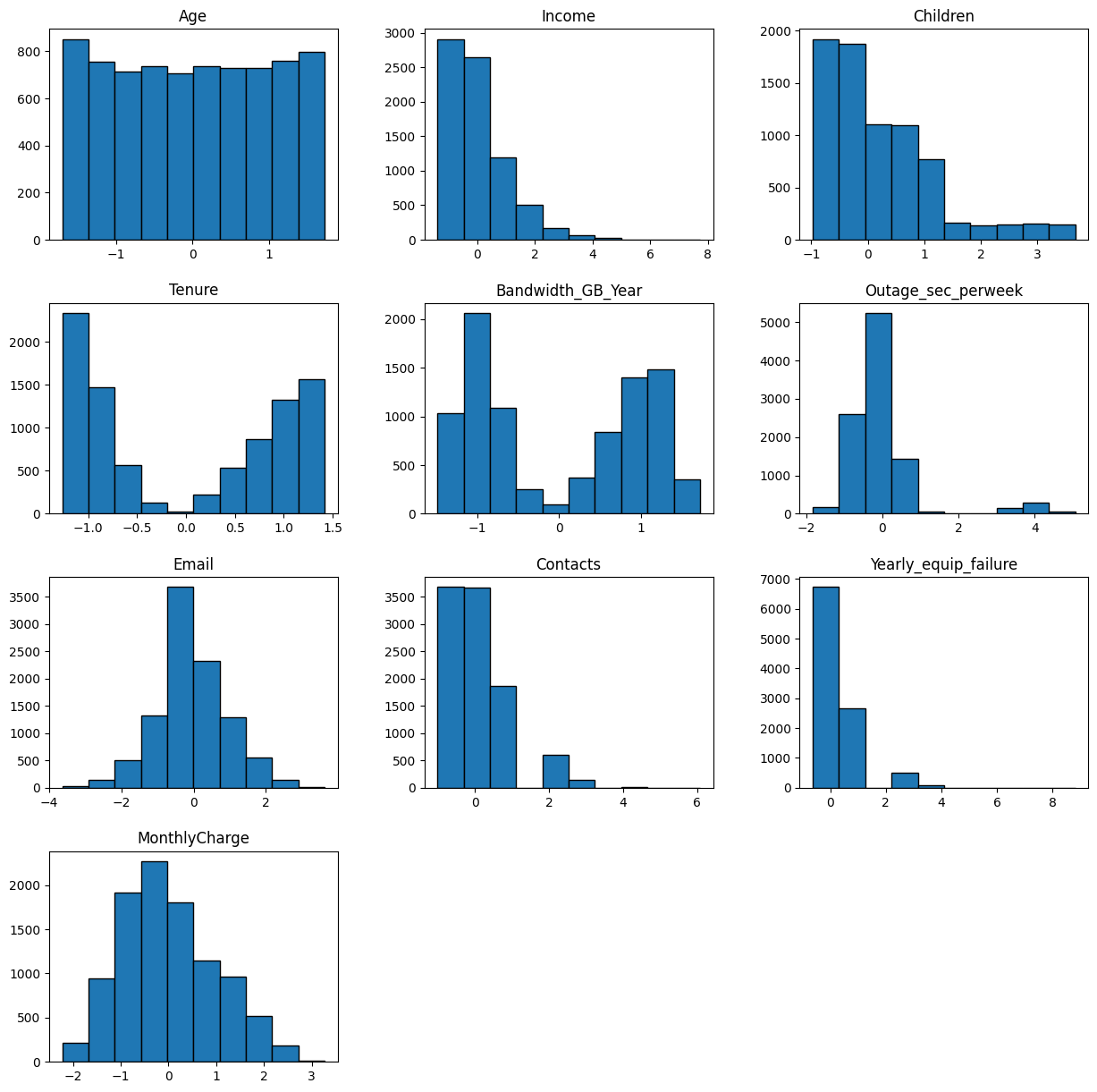
### Detect value errors in yes/no variables

#Check values in yes/no variables [In-Text Citation: (Guar, 2019)]  
print(raw\_data[yes\_no\_variables][~raw\_data[yes\_no\_variables].isin(['Yes','No'])].count())

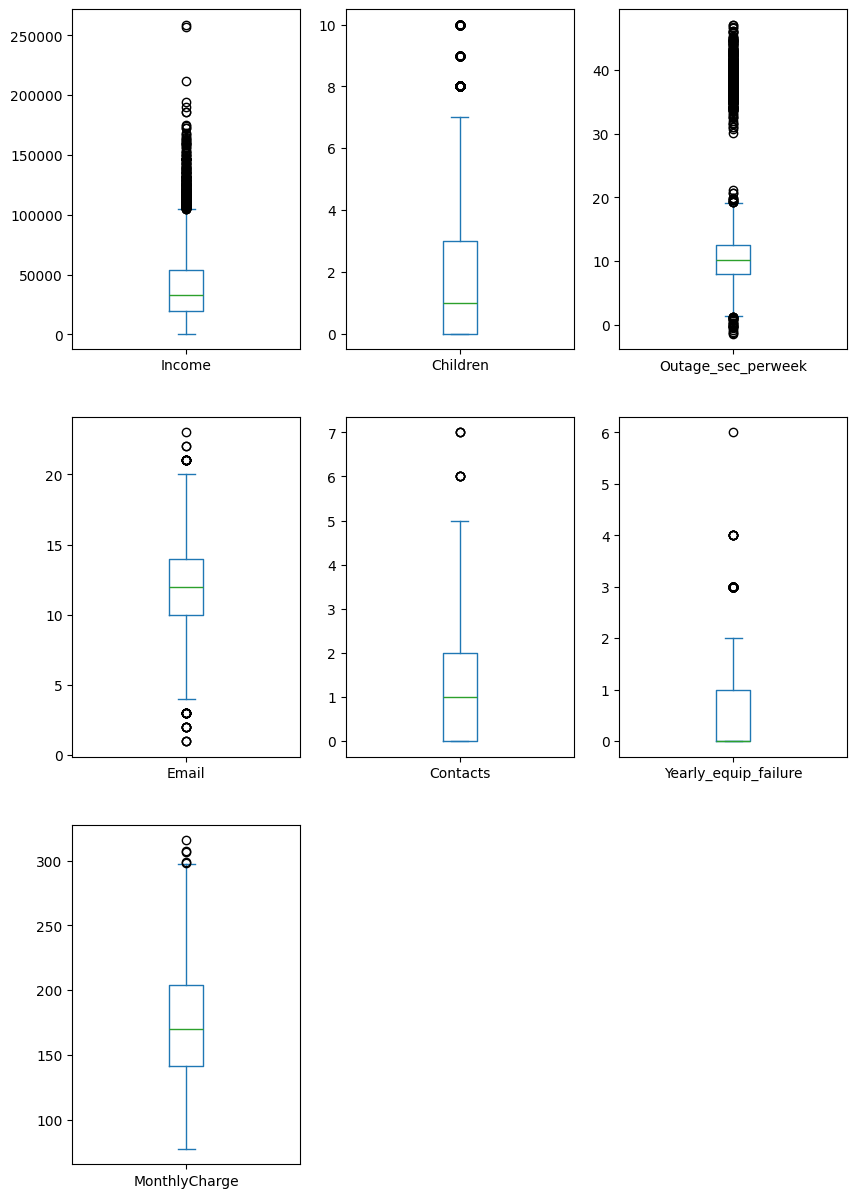
Churn 0  
Techie 0  
Port\_modem 0  
Tablet 0  
Phone 0  
Multiple 0  
OnlineSecurity 0  
OnlineBackup 0  
DeviceProtection 0  
TechSupport 0  
StreamingTV 0  
StreamingMovies 0  
PaperlessBilling 0  
dtype: int64

### Detect possible outliers

#Compute zscore for all numerical variables (other than demographics) [In-Text Citation:(Bathelt, 2017)]  
raw\_data\_z = (raw\_data[numerical\_variables] - raw\_data[numerical\_variables].mean())/raw\_data[numerical\_variables].std(ddof=0)   
raw\_data\_z.hist(edgecolor='black', grid=False, figsize=(15,15))  
plt.show()



#Examine boxplots of variables with any values with zscore > 3 [In-Text Citation:(Varun, 2023)]  
outlier\_cols = raw\_data\_z.loc[: , (raw\_data\_z > 3.0).any()].columns  
raw\_data[outlier\_cols].plot(kind='box', subplots=True, layout=(3,3), sharex=False, sharey=False, figsize=(10, 15))  
plt.show()



#Count, min, max of outliers  
#Formatting code [In-Text Citation:(Sanchhaya Education, Pvt Ltd, 2023)]  
for col in outlier\_cols :  
 cnt = len(raw\_data\_z[raw\_data\_z[col]>3])  
 min, max = raw\_data[col].min(), raw\_data[col].max()  
 print('Likely outlier for {0:<20}\t Count: {1:7d} ({2:5.2%} of observations)\tMin: {3:>9.2f}\tMax: {4:>9.2f}'.format(col,cnt,cnt/10000,min,max))

Likely outlier for Income Count: 110 (1.10% of observations) Min: 740.66 Max: 258900.70  
Likely outlier for Children Count: 144 (1.44% of observations) Min: 0.00 Max: 10.00  
Likely outlier for Outage\_sec\_perweek Count: 491 (4.91% of observations) Min: -1.35 Max: 47.05  
Likely outlier for Email Count: 3 (0.03% of observations) Min: 1.00 Max: 23.00  
Likely outlier for Contacts Count: 165 (1.65% of observations) Min: 0.00 Max: 7.00  
Likely outlier for Yearly\_equip\_failure Count: 94 (0.94% of observations) Min: 0.00 Max: 6.00  
Likely outlier for MonthlyCharge Count: 3 (0.03% of observations) Min: 77.51 Max: 315.88

### Evaluate categorical variables for re-expression

#Evaluate categorical variables for re-expression  
for column in categorical\_variables:  
 print(raw\_data[column].value\_counts())

Education  
Regular High School Diploma 2421  
Bachelor's Degree 1703  
Some College, 1 or More Years, No Degree 1562  
9th Grade to 12th Grade, No Diploma 870  
Master's Degree 764  
Associate's Degree 760  
Some College, Less than 1 Year 652  
Nursery School to 8th Grade 449  
GED or Alternative Credential 387  
Professional School Degree 198  
No Schooling Completed 118  
Doctorate Degree 116  
Name: count, dtype: int64  
Employment  
Full Time 5992  
Part Time 1042  
Retired 1011  
Unemployed 991  
Student 964  
Name: count, dtype: int64  
Marital  
Divorced 2092  
Widowed 2027  
Separated 2014  
Never Married 1956  
Married 1911  
Name: count, dtype: int64  
Gender  
Female 5025  
Male 4744  
Prefer not to answer 231  
Name: count, dtype: int64  
InternetService  
Fiber Optic 4408  
DSL 3463  
Name: count, dtype: int64  
PaymentMethod  
Electronic Check 3398  
Mailed Check 2290  
Bank Transfer(automatic) 2229  
Credit Card (automatic) 2083  
Name: count, dtype: int64

## ## Treatment (D4)

### Drop unneeded column

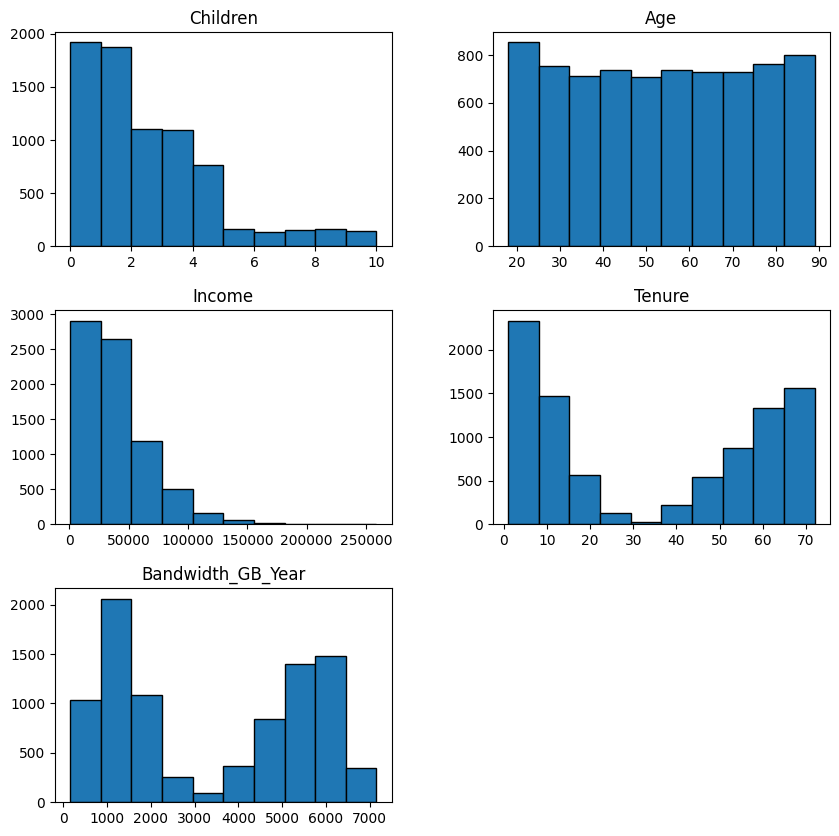
#Drop column zero as it is a copy of column 1 [In-Text Citation:(Chen, 2019)]  
raw\_data.drop(raw\_data.filter(regex="Unnamed"),axis=1, inplace=True)

### Rename vague variables

#Copy the survey response variables and rename for better clarity; leave the original columns intact  
raw\_data[['Survey\_TimelyResponses', \  
 'Survey\_TimelyFixes', \  
 'Survey\_TimelyReplacements', \  
 'Survey\_Reliability', \  
 'Survey\_Options', \  
 'Survey\_Respectful', \  
 'Survey\_Courteous', \  
 'Survey\_ActiveListening']] = \  
 raw\_data[['item1', \  
 'item2', \  
 'item3', \  
 'item4', \  
 'item5', \  
 'item6', \  
 'item7', \  
 'item8']]

### Treat missing data

#Plot a histogram of all numeric columns with missing values  
missing\_data.hist(edgecolor='black', grid=False, figsize=(10,10))  
plt.show()



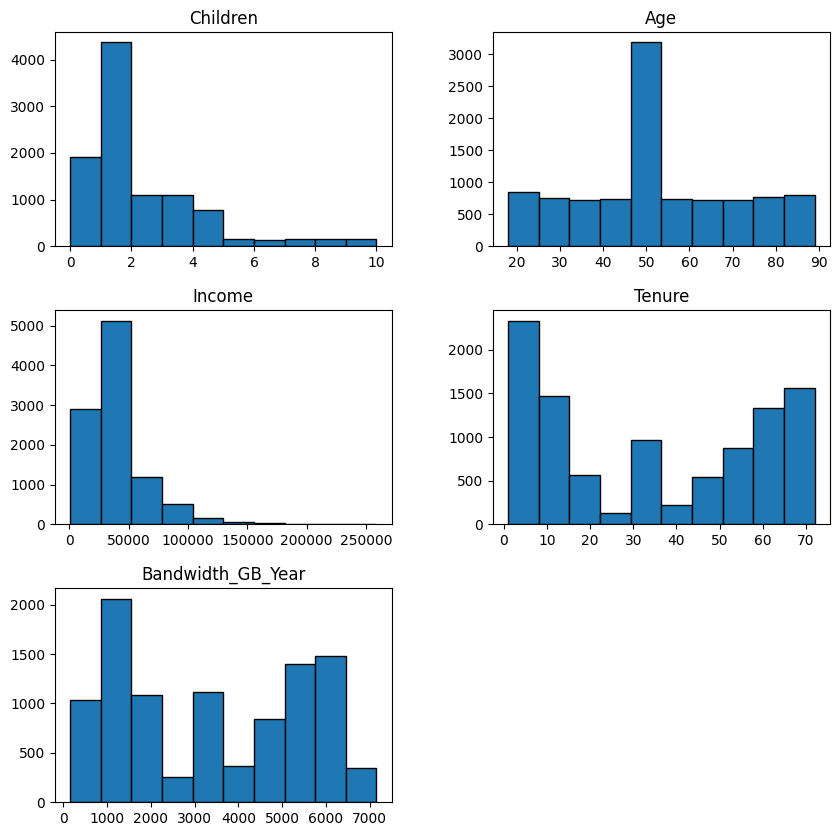
#Create a copy of untreated data for later comparison  
missing\_data\_treated = missing\_data.copy()

#Treat the missing values  
missing\_data\_treated['Age'] = missing\_data['Age'].fillna(missing\_data['Age'].mean().round())  
missing\_data\_treated['Income'] = missing\_data['Income'].fillna(missing\_data['Income'].median())  
missing\_data\_treated['Children'] = missing\_data['Children'].fillna(missing\_data['Children'].median())  
missing\_data\_treated['Tenure'] = missing\_data['Tenure'].fillna(missing\_data['Tenure'].median())  
missing\_data\_treated['Bandwidth\_GB\_Year'] = missing\_data['Bandwidth\_GB\_Year'].fillna(missing\_data['Bandwidth\_GB\_Year'].median())  
missing\_data\_treated['Phone'] = missing\_data['Phone'].fillna(missing\_data['Phone'].mode()[0])  
missing\_data\_treated['TechSupport'] = missing\_data['TechSupport'].fillna(missing\_data['TechSupport'].mode()[0])  
missing\_data\_treated['Techie'] = missing\_data['Techie'].fillna(missing\_data['Techie'].mode()[0])  
missing\_data\_treated['InternetService'] = missing\_data['InternetService'].fillna(missing\_data['InternetService'].mode()[0])

#Check mean and median of before and after treatment  
for column in ['Age','Income','Children','Tenure','Bandwidth\_GB\_Year']:  
 mean\_before = missing\_data[column].mean()  
 mean\_after = missing\_data\_treated[column].mean()  
 mean\_diff = (mean\_before - mean\_after) / mean\_before  
  
 median\_before = missing\_data[column].median()  
 median\_after = missing\_data\_treated[column].median()  
 median\_diff = (median\_before - median\_after) / median\_before  
  
 #Formatting code [In-Text Citation:(Sanchhaya Education, Pvt Ltd, 2023)]  
 print('\nVariable: {}'.format(column))  
 print('\tMean before: {0:7.2f}'.format(mean\_before))  
 print('\tMean after: {0:7.2f}'.format(mean\_after))  
 print('\tMean diff: {0:7.2%}'.format(mean\_diff))  
 print('\n\tMedian before: {0:7.2f}'.format(median\_before))  
 print('\tMedian after: {0:7.2f}'.format(median\_after))  
 print('\tMedian diff: {0:7.2%}'.format(median\_diff))

Variable: Age  
 Mean before: 53.28  
 Mean after: 53.21  
 Mean diff: 0.13%  
  
 Median before: 53.00  
 Median after: 53.00  
 Median diff: 0.00%  
  
Variable: Income  
 Mean before: 39936.76  
 Mean after: 38256.02  
 Mean diff: 4.21%  
  
 Median before: 33186.79  
 Median after: 33186.79  
 Median diff: 0.00%  
  
Variable: Children  
 Mean before: 2.10  
 Mean after: 1.82  
 Mean diff: 13.05%  
  
 Median before: 1.00  
 Median after: 1.00  
 Median diff: 0.00%  
  
Variable: Tenure  
 Mean before: 34.50  
 Mean after: 34.66  
 Mean diff: -0.46%  
  
 Median before: 36.20  
 Median after: 36.20  
 Median diff: 0.00%  
  
Variable: Bandwidth\_GB\_Year  
 Mean before: 3398.84  
 Mean after: 3397.17  
 Mean diff: 0.05%  
  
 Median before: 3382.42  
 Median after: 3382.42  
 Median diff: 0.00%

#Plot a histogram of all treated columns  
missing\_data\_treated.hist(edgecolor='black', grid=False, figsize=(10,10))  
plt.show()



#Replace missing value columns in original df with treated columns  
for column in missing\_data\_treated.columns:  
 raw\_data[column] = missing\_data\_treated[column]

### Treat outliers

#Replace any Outage\_sec\_perweek < 0 with 0  
raw\_data['Outage\_sec\_perweek'] = np.where(raw\_data['Outage\_sec\_perweek'] < 0, 0, raw\_data['Outage\_sec\_perweek'])

### Re-express all yes/no variables as integer

#Reexpress yes/no columns as numbers [In-Text Citation:(Eiler, 2017)]  
yesno\_dict = {'No': 0, 'Yes': 1}  
yes\_no\_int\_variables = []  
for column in yes\_no\_variables:  
 new\_col = column + '\_int'  
 yes\_no\_int\_variables.append(new\_col)  
 raw\_data[new\_col] = raw\_data[column].map(yesno\_dict)  
 print(raw\_data[new\_col].value\_counts())

Churn\_int  
0 7350  
1 2650  
Name: count, dtype: int64  
Techie\_int  
0 8743  
1 1257  
Name: count, dtype: int64  
Port\_modem\_int  
0 5166  
1 4834  
Name: count, dtype: int64  
Tablet\_int  
0 7009  
1 2991  
Name: count, dtype: int64  
Phone\_int  
1 9154  
0 846  
Name: count, dtype: int64  
Multiple\_int  
0 5392  
1 4608  
Name: count, dtype: int64  
OnlineSecurity\_int  
0 6424  
1 3576  
Name: count, dtype: int64  
OnlineBackup\_int  
0 5494  
1 4506  
Name: count, dtype: int64  
DeviceProtection\_int  
0 5614  
1 4386  
Name: count, dtype: int64  
TechSupport\_int  
0 6626  
1 3374  
Name: count, dtype: int64  
StreamingTV\_int  
0 5071  
1 4929  
Name: count, dtype: int64  
StreamingMovies\_int  
0 5110  
1 4890  
Name: count, dtype: int64  
PaperlessBilling\_int  
1 5882  
0 4118  
Name: count, dtype: int64

### Re-express Education variable as integer with fewer values

#Re-express education into fewer numeric ordinal values [In-Text Citation:(Eiler, 2017)]  
edu\_dict = {"Regular High School Diploma" : 12, \  
 "Bachelor's Degree" : 16, \  
 "Some College, 1 or More Years, No Degree" : 13, \  
 "9th Grade to 12th Grade, No Diploma" : 9, \  
 "Master's Degree" : 18, \  
 "Associate's Degree" : 14, \  
 "Some College, Less than 1 Year" : 13, \  
 "Nursery School to 8th Grade" : 8, \  
 "GED or Alternative Credential" : 12, \  
 "Professional School Degree" : 20, \  
 "No Schooling Completed" : 0, \  
 "Doctorate Degree" : 20}  
raw\_data['Education\_int'] = raw\_data['Education'].map(edu\_dict)  
print(raw\_data['Education\_int'].value\_counts())

Education\_int  
12 2808  
13 2214  
16 1703  
9 870  
18 764  
14 760  
8 449  
20 314  
0 118  
Name: count, dtype: int64

### Re-express Employment variable as integer with fewer values

#Re-express employment into fewer numeric values [In-Text Citation:(Eiler, 2017)]  
emp\_dict = {"Full Time" : 2, \  
 "Part Time" : 1, \  
 "Retired" : 0, \  
 "Unemployed" : 0, \  
 "Student" : 0}  
raw\_data['Employment\_int'] = raw\_data['Employment'].map(emp\_dict)  
print(raw\_data['Employment\_int'].value\_counts())

Employment\_int  
2 5992  
0 2966  
1 1042  
Name: count, dtype: int64

## Results (D3)

### Verify redundant column has been removed

raw\_data.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 10000 entries, 0 to 9999  
Data columns (total 74 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 10000 non-null float64  
 14 Age 10000 non-null float64  
 15 Education 10000 non-null object   
 16 Employment 10000 non-null object   
 17 Income 10000 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 10000 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 10000 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 10000 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 10000 non-null float64  
 41 MonthlyCharge 10000 non-null float64  
 42 Bandwidth\_GB\_Year 10000 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   
 51 Survey\_TimelyResponses 10000 non-null int64   
 52 Survey\_TimelyFixes 10000 non-null int64   
 53 Survey\_TimelyReplacements 10000 non-null int64   
 54 Survey\_Reliability 10000 non-null int64   
 55 Survey\_Options 10000 non-null int64   
 56 Survey\_Respectful 10000 non-null int64   
 57 Survey\_Courteous 10000 non-null int64   
 58 Survey\_ActiveListening 10000 non-null int64   
 59 Churn\_int 10000 non-null int64   
 60 Techie\_int 10000 non-null int64   
 61 Port\_modem\_int 10000 non-null int64   
 62 Tablet\_int 10000 non-null int64   
 63 Phone\_int 10000 non-null int64   
 64 Multiple\_int 10000 non-null int64   
 65 OnlineSecurity\_int 10000 non-null int64   
 66 OnlineBackup\_int 10000 non-null int64   
 67 DeviceProtection\_int 10000 non-null int64   
 68 TechSupport\_int 10000 non-null int64   
 69 StreamingTV\_int 10000 non-null int64   
 70 StreamingMovies\_int 10000 non-null int64   
 71 PaperlessBilling\_int 10000 non-null int64   
 72 Education\_int 10000 non-null int64   
 73 Employment\_int 10000 non-null int64   
dtypes: float64(9), int64(37), object(28)  
memory usage: 5.6+ MB

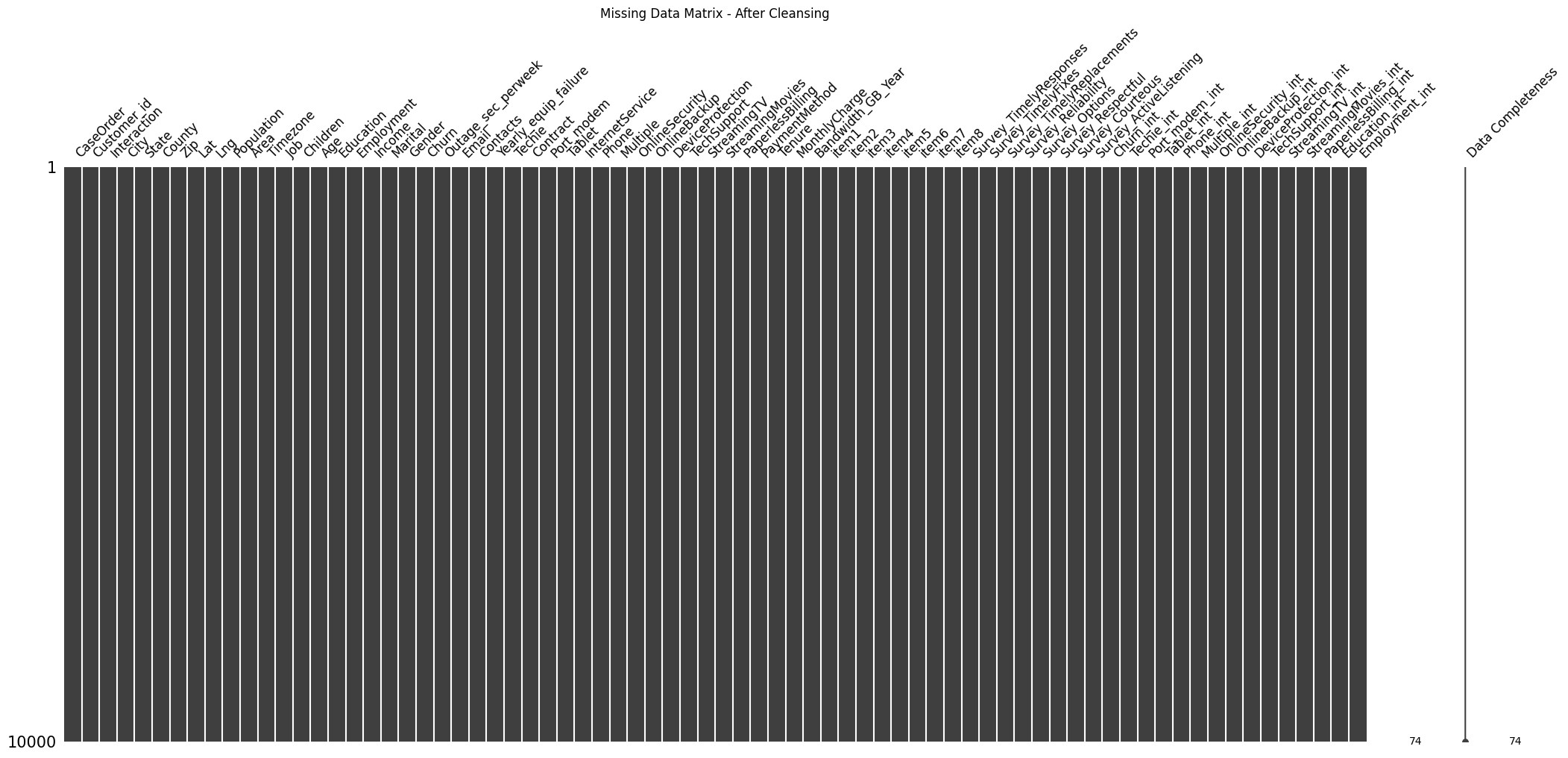
### Verify no missing data

#List any features with missing values [In-Text Citation: (Uzunov, 2016)]  
missing\_data = raw\_data[raw\_data.columns[raw\_data.isnull().any()]].copy()  
print(len(missing\_data.columns)) # How many variables have missing data?  
print(missing\_data.isnull().sum()) # How many missing values in each variable?

0  
Series([], dtype: float64)

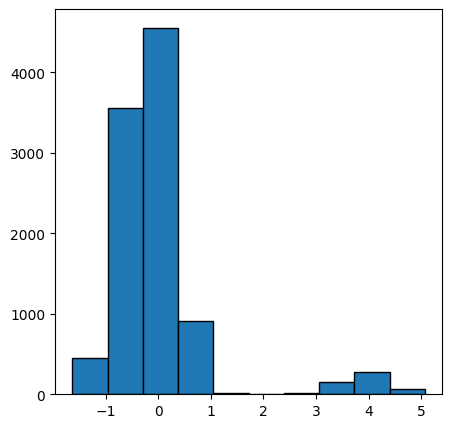
#Check once again that no data are missing [In-Text Citation: (Uzunov, 2016)]  
print(raw\_data[raw\_data.columns[raw\_data.isnull().any()]].isnull().sum())  
msno.matrix(raw\_data, fontsize = 12, labels=True)  
plt.title('Missing Data Matrix - After Cleansing')  
plt.show()

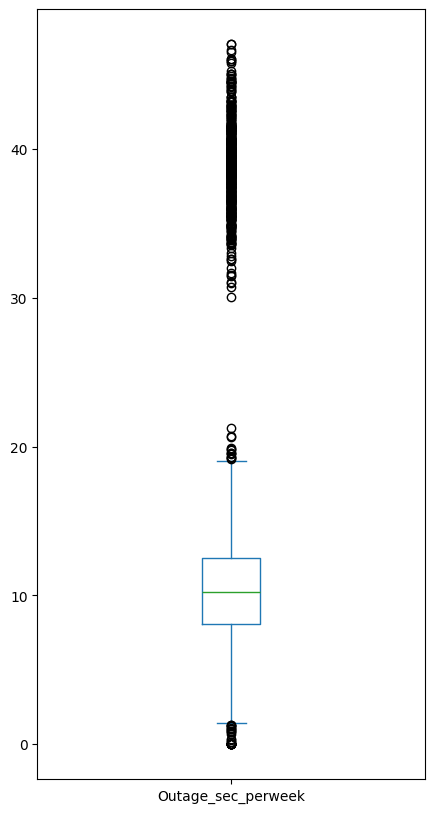
Series([], dtype: float64)



### Evaluate outliers treated

#Update zscore for Outage\_sec\_perweek and plot [In-Text Citation:(Bathelt, 2017)]  
raw\_data\_z['Outage\_sec\_perweek'] = (raw\_data['Outage\_sec\_perweek'] - raw\_data['Outage\_sec\_perweek'].mean())/raw\_data['Outage\_sec\_perweek'].std()  
raw\_data\_z['Outage\_sec\_perweek'].hist(edgecolor='black', grid=False, figsize=(5,5))  
plt.show()  
#Check boxplot of Outage\_sec\_perweek after treatment  
raw\_data['Outage\_sec\_perweek'].plot(kind='box', figsize=(5, 10))  
plt.show()





### Verify re-expression of Yes/No variables

raw\_data[yes\_no\_int\_variables].info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 10000 entries, 0 to 9999  
Data columns (total 13 columns):  
 # Column Non-Null Count Dtype  
--- ------ -------------- -----  
 0 Churn\_int 10000 non-null int64  
 1 Techie\_int 10000 non-null int64  
 2 Port\_modem\_int 10000 non-null int64  
 3 Tablet\_int 10000 non-null int64  
 4 Phone\_int 10000 non-null int64  
 5 Multiple\_int 10000 non-null int64  
 6 OnlineSecurity\_int 10000 non-null int64  
 7 OnlineBackup\_int 10000 non-null int64  
 8 DeviceProtection\_int 10000 non-null int64  
 9 TechSupport\_int 10000 non-null int64  
 10 StreamingTV\_int 10000 non-null int64  
 11 StreamingMovies\_int 10000 non-null int64  
 12 PaperlessBilling\_int 10000 non-null int64  
dtypes: int64(13)  
memory usage: 1015.8 KB

## Export Clean Data (D5)

#Write cleaned data to CSV  
raw\_data.to\_csv('churn\_clean\_data.csv')

## Principal Component Analysis

#Standardize numeric variables in preparation for PCA [In-Text Citation:(Middleton, Getting Started with Principal Component Analysis (PCA), 2023)]  
normalized\_data = (raw\_data[numerical\_variables] - raw\_data[numerical\_variables].mean()) / raw\_data[numerical\_variables].std()

#Fit the model [In-Text Citation:(Middleton, Getting Started with Principal Component Analysis (PCA), 2023)]  
pca = PCA(n\_components=normalized\_data.shape[1])  
pca.fit(normalized\_data)  
col\_names = []  
for i in range(normalized\_data.shape[1]):  
 col\_names.append('PC' + str(i+1))  
pca\_data = pd.DataFrame(pca.transform(normalized\_data),columns=col\_names)

#Examine the loadings [In-Text Citation:(Middleton, Getting Started with Principal Component Analysis (PCA), 2023)]  
loadings\_data = pd.DataFrame(pca.components\_.T, columns=col\_names, index=raw\_data[numerical\_variables].columns)  
print(loadings\_data)

PC1 PC2 PC3 PC4 PC5 \  
Age -0.012380 -0.047346 -0.462197 0.441230 -0.092627   
Income 0.006196 -0.006591 0.232681 0.306368 0.790237   
Children -0.001879 0.017635 0.587184 -0.072966 0.057459   
Tenure 0.704917 -0.058011 -0.018841 -0.011045 -0.003719   
Bandwidth\_GB\_Year 0.706839 -0.009236 0.004624 -0.017922 0.001164   
Outage\_sec\_perweek 0.022535 0.707244 0.042371 0.026858 0.018518   
Email -0.021286 0.065941 -0.234674 -0.592281 -0.069137   
Contacts 0.004536 -0.004906 -0.495040 0.231550 0.251028   
Yearly\_equip\_failure 0.015832 0.054784 0.277052 0.547830 -0.542672   
MonthlyCharge 0.045209 0.697414 -0.091558 0.020897 0.031586   
  
 PC6 PC7 PC8 PC9 PC10   
Age 0.429082 -0.504067 -0.356839 0.121335 0.021552   
Income 0.211687 -0.192125 0.375348 -0.069506 0.001167   
Children 0.498470 0.227574 -0.587861 0.009695 -0.018397   
Tenure 0.015694 -0.016664 0.007748 0.038156 -0.705122   
Bandwidth\_GB\_Year 0.008043 0.004107 -0.009499 -0.012675 0.706836   
Outage\_sec\_perweek -0.011445 0.021988 0.072282 0.700428 0.000614   
Email 0.658569 -0.020440 0.383825 -0.055374 0.005569   
Contacts 0.132169 0.783465 -0.083286 0.005402 -0.002982   
Yearly\_equip\_failure 0.261623 0.192000 0.451426 -0.127582 -0.002459   
MonthlyCharge -0.050345 -0.073057 -0.150933 -0.684664 -0.048329

#Calculate and plot eigenvalues [In-Text Citation:(Middleton, Getting Started with Principal Component Analysis (PCA), 2023)]  
cov\_matrix = np.dot(normalized\_data.T, normalized\_data) / raw\_data[numerical\_variables].shape[0]  
eigenvalues = [np.dot(eigenvector.T, np.dot(cov\_matrix, eigenvector)) for eigenvector in pca.components\_]  
fig = plt.figure(figsize=(10,5))  
plt.plot(np.arange(1,len(eigenvalues)+1),eigenvalues, marker='o')  
plt.xlabel('PC number')  
plt.ylabel('eigenvalue')  
plt.axhline(y=1, color="red")  
plt.show()

