Task 1 Clustering Techniques

Data Preparation

In [1]:
```python
#import packages and libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
```

In [2]:
```python
# load data set

df = pd.read_csv('churn_clean.csv')
```

In [3]:
```python
# look at size of data set
df.shape
```

Out[3]: (10000, 50)

In [4]:
```python
# look at first 5 rows
df.head()
```

Out[4]:

	CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	L
0	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	e885b299883d4f9fb18e39c75155d990	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	-133.375
1	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	f2de8bef964785f41a2959829830fb8a	West Branch	MI	Ogemaw	48661	44.32893	-84.240
2	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	f1784cfa9f6d92ae816197eb175d3c71	Yamhill	OR	Yamhill	97148	45.35589	-123.246
3	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	dc8a365077241bb5cd5ccd305136b05e	Del Mar	CA	San Diego	92014	32.96687	-117.247
4	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	aabb64a116e83fdc4befc1fbab1663f9	Needville	TX	Fort Bend	77461	29.38012	-95.806

5 rows × 50 columns

In [5]:
```python
# list all column names and data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 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   UID                   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              10000 non-null  int64
 15  Age                   10000 non-null  int64
 16  Income                10000 non-null  float64
 17  Marital               10000 non-null  object
 18  Gender                10000 non-null  object
 19  Churn                 10000 non-null  object
 20  Outage_sec_perweek    10000 non-null  float64
 21  Email                 10000 non-null  int64
 22  Contacts              10000 non-null  int64
 23  Yearly_equip_failure  10000 non-null  int64
 24  Techie                10000 non-null  object
 25  Contract              10000 non-null  object
 26  Port_modem            10000 non-null  object
 27  Tablet                10000 non-null  object
 28  InternetService       10000 non-null  object
 29  Phone                 10000 non-null  object
 30  Multiple              10000 non-null  object
 31  OnlineSecurity        10000 non-null  object
 32  OnlineBackup          10000 non-null  object
 33  DeviceProtection      10000 non-null  object
 34  TechSupport           10000 non-null  object
 35  StreamingTV           10000 non-null  object
 36  StreamingMovies       10000 non-null  object
 37  PaperlessBilling      10000 non-null  object
 38  PaymentMethod         10000 non-null  object
 39  Tenure                10000 non-null  float64
 40  MonthlyCharge         10000 non-null  float64
 41  Bandwidth_GB_Year     10000 non-null  float64
 42  Item1                 10000 non-null  int64
 43  Item2                 10000 non-null  int64
 44  Item3                 10000 non-null  int64
 45  Item4                 10000 non-null  int64
 46  Item5                 10000 non-null  int64
 47  Item6                 10000 non-null  int64
 48  Item7                 10000 non-null  int64
 49  Item8                 10000 non-null  int64
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

In [6]:
```python
# statistic summary
df.describe()
```

Out[6]:

	CaseOrder	Zip	Lat	Lng	Population	Children	Age	Income	Outage_s
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.0000	10000.000000	10000.000000	10
mean	5000.50000	49153.319600	38.757567	-90.782536	9756.562400	2.0877	53.078400	39806.926771	
std	2886.89568	27532.196108	5.437389	15.156142	14432.698671	2.1472	20.698882	28199.916702	
min	1.00000	601.000000	17.966120	-171.688150	0.000000	0.0000	18.000000	348.670000	
25%	2500.75000	26292.500000	35.341828	-97.082813	738.000000	0.0000	35.000000	19224.717500	
50%	5000.50000	48869.500000	39.395800	-87.918800	2910.500000	1.0000	53.000000	33170.605000	
75%	7500.25000	71866.500000	42.106908	-80.088745	13168.000000	3.0000	71.000000	53246.170000	
max	10000.00000	99929.000000	70.640660	-65.667850	111850.000000	10.0000	89.000000	258900.700000	

8 rows × 23 columns

In [7]:
```python
# drop catagorical and other columns not needed for this analysis

df2 = df.drop(['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State', \
              'County', 'Zip', 'Lat', 'Lng', 'TimeZone', 'Job', 'Area', 'Marital',\
             'Gender', 'Churn', 'Techie', 'Contract', 'Port_modem', 'Tablet',\
             'InternetService', 'Phone', 'Multiple','OnlineSecurity', 'OnlineBackup',\
             'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',\
             'PaperlessBilling', 'PaymentMethod', 'Item1', 'Item2', 'Item3', 'Item4',\
             'Item5', 'Item6', 'Item7', 'Item8'], axis = 1)
```

In [8]:
```python
# inspect new data frame
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Population            10000 non-null  int64
 1   Children              10000 non-null  int64
 2   Age                   10000 non-null  int64
 3   Income                10000 non-null  float64
 4   Outage_sec_perweek    10000 non-null  float64
 5   Email                 10000 non-null  int64
 6   Contacts              10000 non-null  int64
 7   Yearly_equip_failure  10000 non-null  int64
 8   Tenure                10000 non-null  float64
 9   MonthlyCharge         10000 non-null  float64
 10  Bandwidth_GB_Year     10000 non-null  float64
dtypes: float64(5), int64(6)
memory usage: 859.5 KB
```

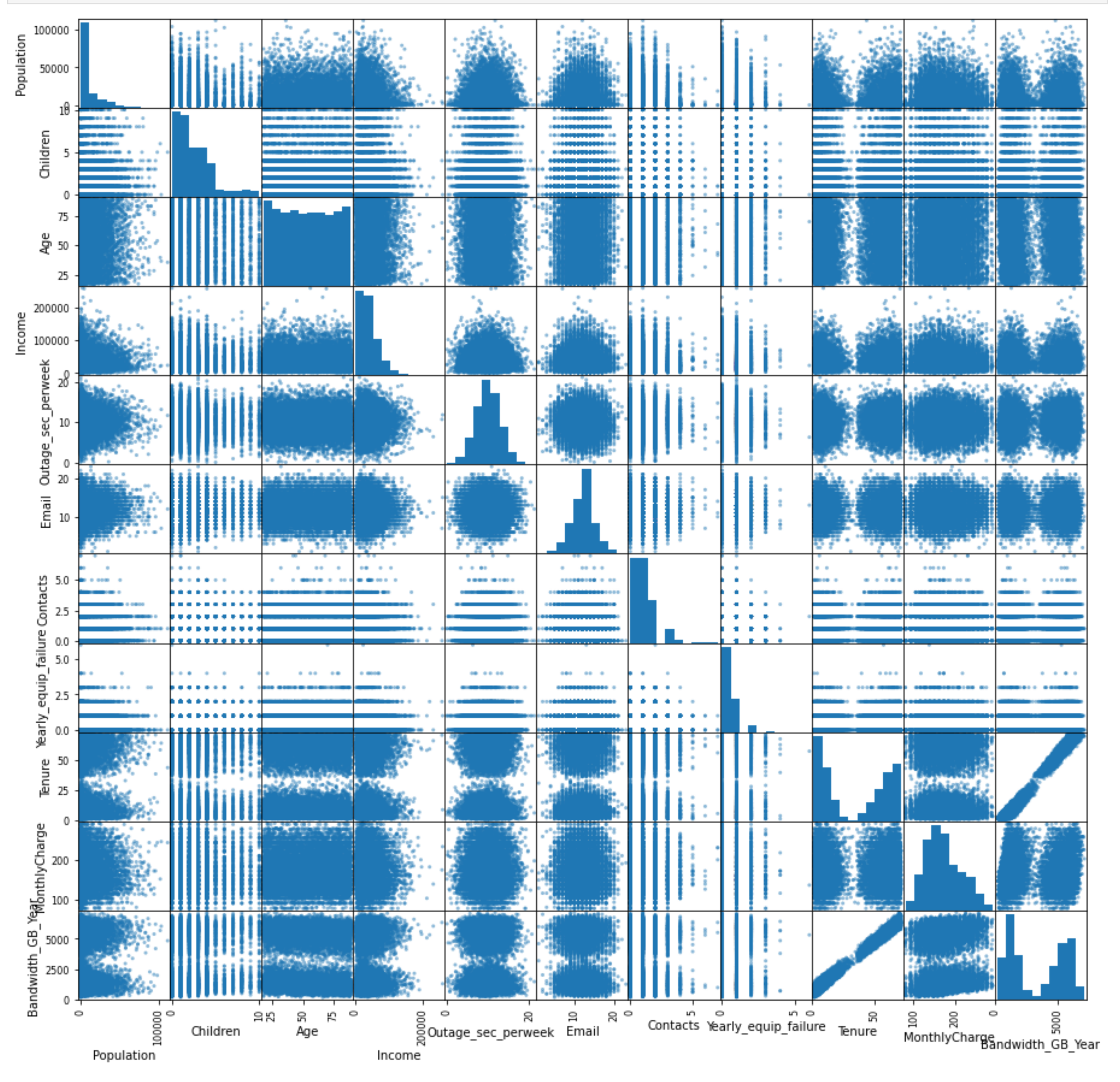
In [9]:
```python
# check for null values in data set
df2.isnull().sum()
```

Out[9]:
```
Population              0
Children                0
Age                     0
Income                  0
Outage_sec_perweek      0
Email                   0
Contacts                0
Yearly_equip_failure    0
Tenure                  0
MonthlyCharge           0
Bandwidth_GB_Year       0
dtype: int64
```

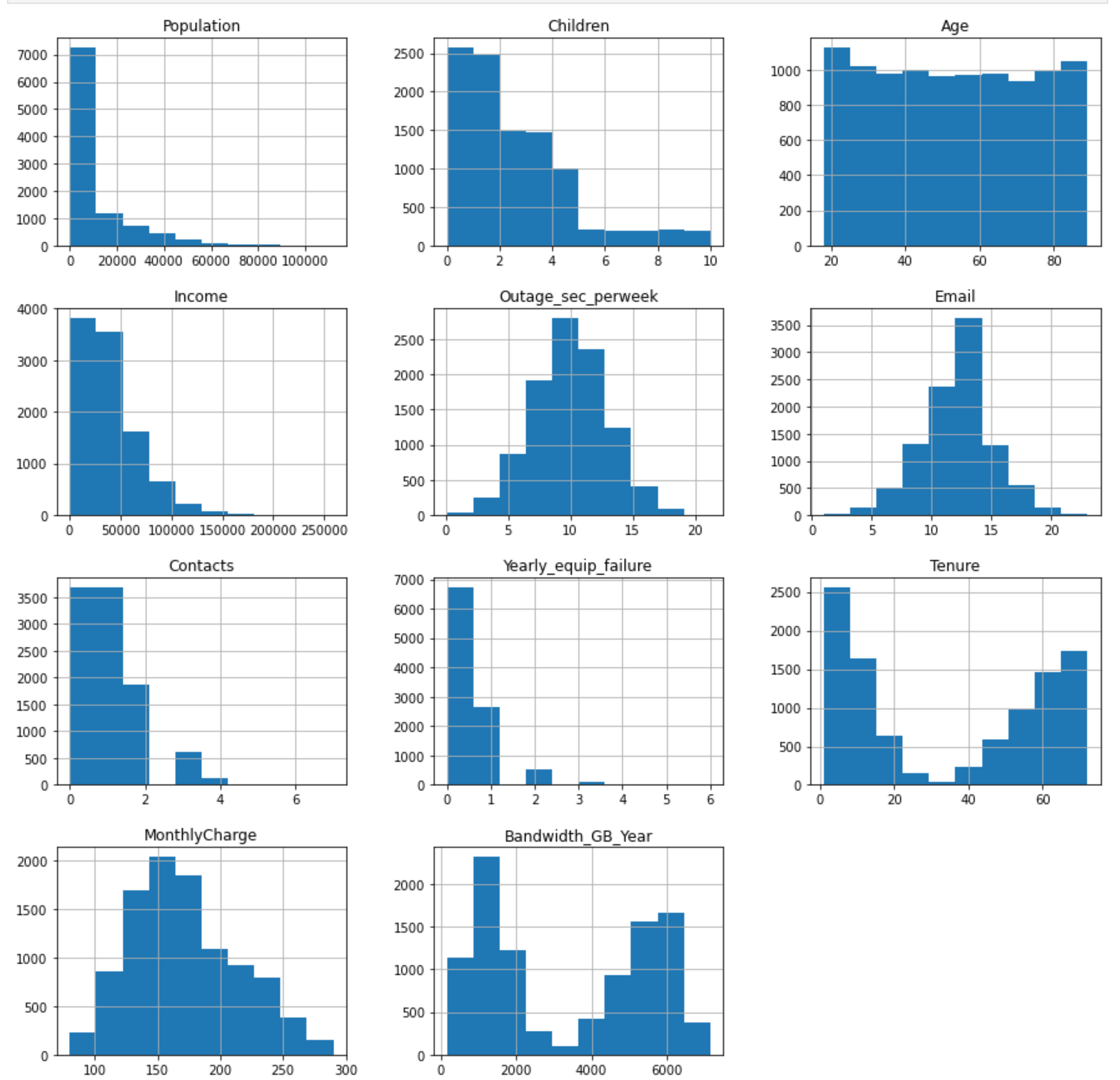
In [10]:
```python
# Create scatter matrix to look for potential relationships

matrix_features = df2[['Population', 'Children', 'Age', 'Income', 'Outage_sec_perweek', \
                       'Email', 'Contacts', 'Yearly_equip_failure', 'Tenure', \
                        'MonthlyCharge', 'Bandwidth_GB_Year']]

scatter_matrix = pd.plotting.scatter_matrix(matrix_features, figsize = [15,15]);
```



In [11]:
```python
# Create histograms to look at distribution of continuous features
df2.hist(figsize =[15,15]);
```



In [12]:
```python
df2.to_csv('Clustering_Prepared_Data.csv')
```

Analysis with KMeans

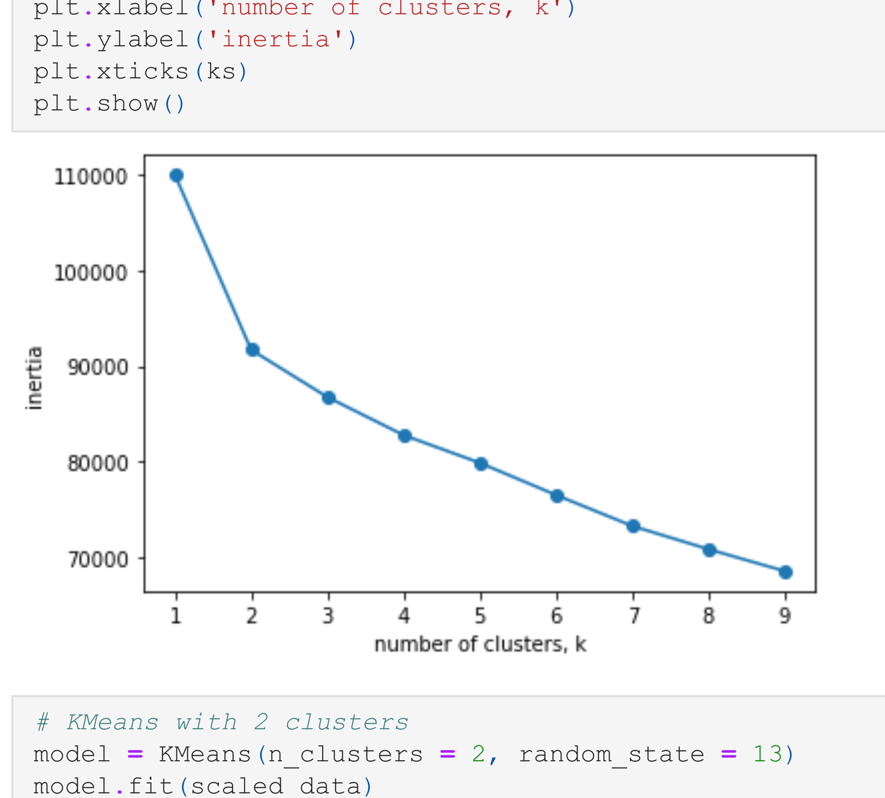
In [13]:
```python
# Use StandardScaler and check results
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df2)
scaled_data = pd.DataFrame(scaled_data, columns  = df2.columns)
scaled_data.head()
```

Out[13]:

	Population	Children	Age	Income	Outage_sec_perweek	Email	Contacts	Yearly_equip_failure	Tenure	MonthlyC
0	-0.673405	-0.972338	0.720925	-0.398778	-0.679978	-0.666282	-1.005852	0.946658	-1.048746	-0.0
1	0.047772	-0.506592	-1.259957	-0.641954	0.570331	-0.005288	-1.005852	0.946658	-1.262001	1.6
2	-0.417238	0.890646	-0.148730	-1.070885	0.252347	-0.996779	-1.005852	0.946658	-0.709940	-0.2
3	0.284537	-0.506592	-0.245359	-0.740525	1.650506	0.986203	1.017588	-0.625864	-0.659524	-1.2
4	0.110549	-0.972338	1.445638	0.009478	-0.623156	1.316700	1.017588	0.946658	-1.242551	-0.5

In [14]:
```python
# Build inertia plot to find number of clusters to use
ks = range(1,10)
inertias = []
for k in ks:
    model = KMeans(n_clusters = k)
    model.fit(scaled_data)
    inertias.append(model.inertia_)
```

In [15]:
```python
# Make elbow polt of ks vs inertias
plt.plot(ks, inertias, '-o')
plt.xlabel('number of clusters, k')
plt.ylabel('inertia')
plt.xticks(ks)
plt.show()
```



In [16]:
```python
# KMeans with 2 clusters
model = KMeans(n_clusters = 2, random_state = 13)
model.fit(scaled_data)
labels = model.predict(scaled_data)
print(labels)
```

```
[0 0 0 ... 1 1 1]
```

In [17]:
```python
# Create a new column in original data set for Kmeans labels
df['labels'] = labels
df.head()
```

Out[17]:

	CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	L
0	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	e885b299883d4f9fb18e39c75155d990	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	-133.375
1	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	f2de8bef964785f41a2959829830fb8a	West Branch	MI	Ogemaw	48661	44.32893	-84.240
2	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	f1784cfa9f6d92ae816197eb175d3c71	Yamhill	OR	Yamhill	97148	45.35589	-123.246
3	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	dc8a365077241bb5cd5ccd305136b05e	Del Mar	CA	San Diego	92014	32.96687	-117.247
4	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	aabb64a116e83fdc4befc1fbab1663f9	Needville	TX	Fort Bend	77461	29.38012	-95.806

5 rows × 51 columns

In [18]:
```python
# Count number of customers who churned
churned = df.query('Churn == "Yes"').Customer_id.count()
print(churned)
```

```
2650
```

In [19]:
```python
# Count number of customers who churned and are in cluster 1
clusterchurn = df.query('labels == 0 and Churn == "Yes"').Customer_id.count()
print(clusterchurn)
```

```
2366
```

In [20]:
```python
# Create crosstabulation table to inspect label to churn
pd.crosstab(df.labels, df.Churn)
```

Out[20]:

Churn	No	Yes
labels		
0	2635	2366
1	4715	284

In [21]:
```python
# Overall Accuracy score
cluster_not_churn = df.query('labels == 1 and Churn == "No"').Customer_id.count()
(cluster_not_churn + clusterchurn)/ df.Customer_id.count()
```

Out[21]: 0.7081

In [22]:
```python
# Churn prediction accuracy
predacc = clusterchurn/churned
print(predacc)
```

```
0.8928301886792452
```

In [31]:
```python
# code from Python Data Science Handbook
# (https://jakevdp.github.io/PythonDataScienceHandbook/05.11-k-means.html)
plt.scatter(scaled_data.iloc[:,0], scaled_data.iloc[:,1], c = labels, cmap = 'viridis')

centers = model.cluster_centers_
plt.scatter(centers[:,0], centers[:,1], c = 'black', s = 200)
```

Out[31]: <matplotlib.collections.PathCollection at 0x7fee710a8d00>