William Stults - D208 Task 1

May 14, 2022

---

1 Part I: Research Question

1.1 Research Question

My dataset for this predictive modeling exercise includes data on an internet service provider's current and former subscribers, with an emphasis on customer churn (whether customers are maintaining or discontinuing their subscription to the ISP's service). Data analysis performed on the dataset will be aimed with this research question in mind: is there a relationship between customer lifestyle, or "social" factors, and customer churn? Lifestyle and social factors might include variables such as age, income, and marital status, among others.

1.2 Objectives and Goals

Conclusions gleaned from analysis of this data can benefit stakeholders by revealing information on which customer populations may be more likely to "churn", or to terminate their service contract with the ISP. Such information may be used to fuel targeted advertising campaigns, special promotional offers, and other strategies related to customer retention.

---

2 Part II: Method Justification

2.1 Assumptions of a multiple regression model

The assumptions of a multiple regression model are as follows:

- There exists a linear relationship between each predictor variable and the response variable
- None of the predictor variables are highly correlated with each other
- The observations are independent
- The residuals have constant variance at every point in the linear model
- The residuals of the model are normally distributed

For each of these assumptions that is violated, the potential reliability of the multiple regression model decreases. Adherance to these assumptions can be measured via tests such as Durbin-Watson, Q-Q plot, and VIF (Zach, 2021).

2.2 Tool Selection

All code execution was carried out via Jupyter Lab, using Python 3. I used Python as my selected programming language due to prior familiarity and broader applications when considering programming in general. R is a very strong and robust language tool for data analysis and statistics but finds itself somewhat limited to that niche role (Insights for Professionals, 2019). I utilized the NumPy, Pandas, and Matplotlib libraries to perform many of my data analysis tasks, as they are among the most popular Python libraries employed for this purpose and see widespread use. Seaborn is included primarily for its better-looking boxplots, seen later in this document (Parra, 2021).

Beyond these libraries, I relied upon the Statsmodels library; in particular its main API, formula API, and the variance_inflation_factor from its outliers_influence module. Statsmodels is one of several Python libraries that support linear regression. I am most familiar with it due to the course material's heavy reliance upon it.

[3]:
```
# Imports and housekeeping
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.outliers_influence import variance_inflation_factor
sns.set_theme(style="darkgrid")
```

2.3 Why Multiple Regression?

Simple linear regression allows us to determine whether a relationship exists between a dependent variable and a single independent variable. This type of model does have its uses and proper applications, but results in a more simple predictive model without taking into account how other variables may relate to both the independent and dependent variable. In both the real world and business world it may be rare to encounter data collections with only 2 variables, and relying heavily on simple linear regression models can create a situation where predictions are somewhat unreliable. Utilizing multiple independent variables in a predictive model can make our predictions stronger and allows higher conviction in the reliance on those models for decision making.

---

3 Part III: Data Preparation

3.1 Data Preparation Goals and Data Manipulations

I would like my data to include only variables relevant to my research question, and to be clean and free of missing values and duplicate rows. It will also be important to re-express any categorical variable types with numeric values. My first steps will be to import the complete data set and execute functions that will give me information on its size, the data types of its variables, and a peek at the data in table form. I will then narrow the data set to a new dataframe containing only the variables I am concerned with, and then utilizing functions to determine if any null values or duplicate rows exist.

[4]:
```
# Import the main dataset
df = pd.read_csv('churn_clean.csv',dtype={'locationid':np.int64})
```

[5]:
```
# Display dataset info
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
```

[6]:
```
# Display dataset top 5 rows
df.head()
```

[6]:
```
   CaseOrder Customer_id                           Interaction  \
0          1     K409198  aa90260b-4141-4a24-8e36-b04ce1f4f77b
1          2     S120509  fb76459f-c047-4a9d-8af9-e0f7d4ac2524
2          3     K191035  344d114c-3736-4be5-98f7-c72c281e2d35
3          4      D90850  abfa2b40-2d43-4994-b15a-989b8c79e311
4          5     K662701  68a861fd-0d20-4e51-a587-8a90407ee574

                                UID         City State                 County  \
0  e885b299883d4f9fb18e39c75155d990  Point Baker    AK  Prince of Wales-Hyder
1  f2de8bef964785f41a2959829830fb8a  West Branch    MI                 Ogemaw
2  f1784cfa9f6d92ae816197eb175d3c71      Yamhill    OR                Yamhill
3  dc8a365077241bb5cd5ccd305136b05e      Del Mar    CA              San Diego
4  aabb64a116e83fdc4befc1fbab1663f9   Needville    TX              Fort Bend

     Zip       Lat        Lng  …  MonthlyCharge Bandwidth_GB_Year Item1  \
0  99927  56.25100 -133.37571  …     172.455519        904.536110     5
1  48661  44.32893  -84.24080  …     242.632554        800.982766     3
2  97148  45.35589 -123.24657  …     159.947583       2054.706961     4
3  92014  32.96687 -117.24798  …     119.956840       2164.579412     4
4  77461  29.38012  -95.80673  …     149.948316        271.493436     4

  Item2  Item3  Item4  Item5 Item6 Item7 Item8
0     5      5      3      4     4     3     4
1     4      3      3      4     3     4     4
2     4      2      4      4     3     3     3
```

```
3    4     4     2     5    4    3    3
4    4     4     3     4    4    4    5

[5 rows x 50 columns]
```

[7]:
```
# Trim dataset to variables relevant to research question
columns = ['Area', 'Children', 'Age', 'Income', 'Marital', 'Gender', 'Churn',␣
↪'Outage_sec_perweek',
           'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',␣
↪'Bandwidth_GB_Year']
df_data = pd.DataFrame(df[columns])
```

[8]:
```
# Check data for null or missing values
df_data.isna().any()
```

[8]:
```
Area                    False
Children                False
Age                     False
Income                  False
Marital                 False
Gender                  False
Churn                   False
Outage_sec_perweek      False
Yearly_equip_failure    False
Tenure                  False
MonthlyCharge           False
Bandwidth_GB_Year       False
dtype: bool
```

[9]:
```
# Check data for duplicated rows
df_data.duplicated().sum()
```

[9]: 0

3.2 Summary Statistics

I can use the describe() function to display the summary statistics for the entire dataframe, as well as each variable I'll be evaluating for inclusion in the model. I have selected the Bandwidth_GB_Year variable as my dependent variable.

I will also utilize histogram plots to illustrate the distribution of each numeric variable in the dataframe, and countplots for the categorical variables.

[10]:
```
# Display summary statistics for entire dataset - continuous variables
df_data.describe()
```

[10]:
```
          Children           Age          Income  Outage_sec_perweek  \
count   10000.0000  10000.000000    10000.000000        10000.000000
mean        2.0877     53.078400    39806.926771           10.001848
std         2.1472     20.698882    28199.916702            2.976019
min         0.0000     18.000000      348.670000            0.099747
25%         0.0000     35.000000    19224.717500            8.018214
50%         1.0000     53.000000    33170.605000           10.018560
75%         3.0000     71.000000    53246.170000           11.969485
max        10.0000     89.000000   258900.700000           21.207230

       Yearly_equip_failure        Tenure  MonthlyCharge  Bandwidth_GB_Year
count          10000.000000  10000.000000   10000.000000       10000.000000
mean               0.398000     34.526188     172.624816        3392.341550
std                0.635953     26.443063      42.943094        2185.294852
min                0.000000      1.000259      79.978860         155.506715
25%                0.000000      7.917694     139.979239        1236.470827
50%                0.000000     35.430507     167.484700        3279.536903
75%                1.000000     61.479795     200.734725        5586.141370
max                6.000000     71.999280     290.160419        7158.981530
```

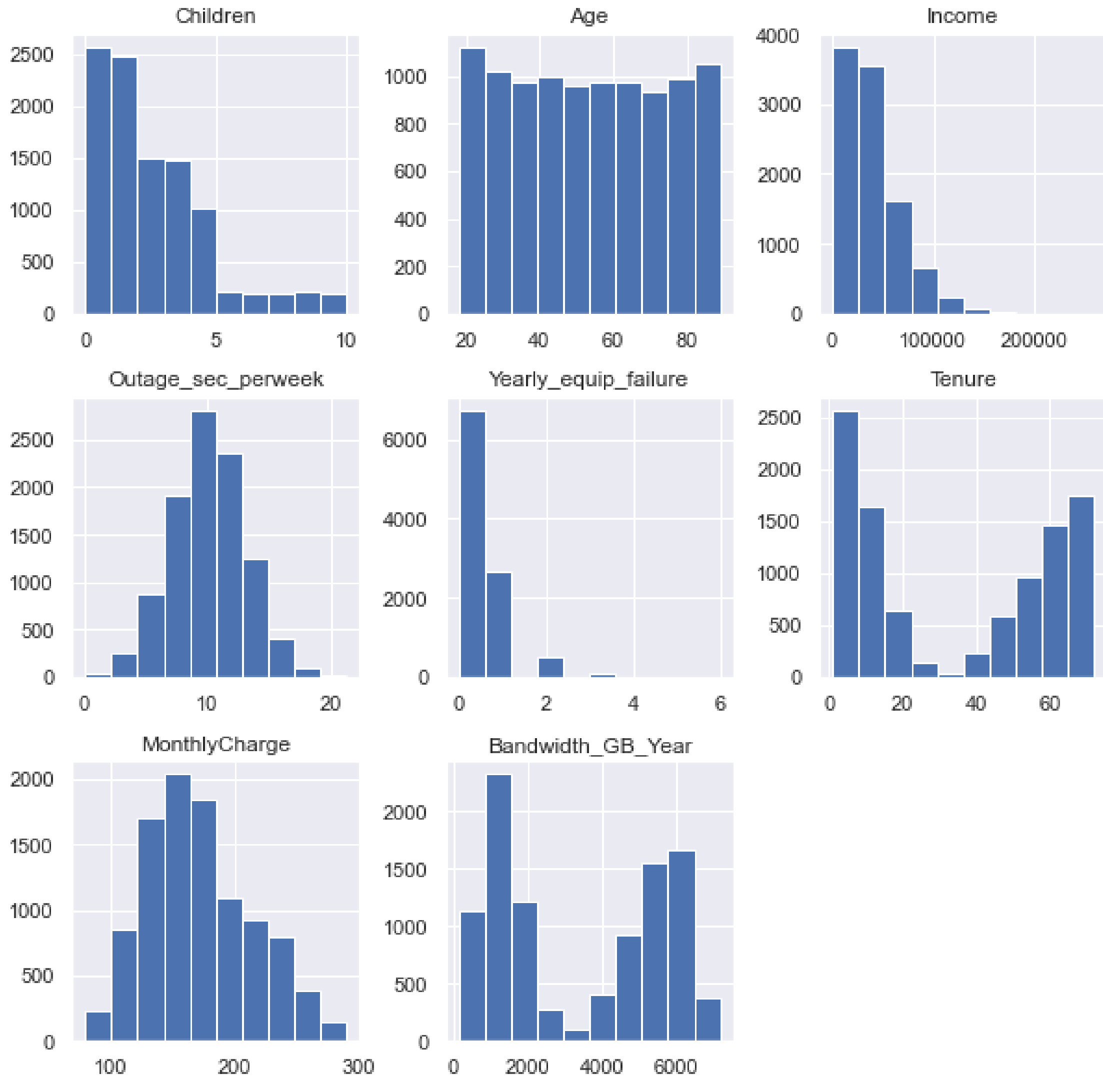
[11]:
```python
# Display summary statistics for entire dataset - categorical variables
df_data.describe(include = object)
```

[11]:
```
            Area   Marital  Gender  Churn
count      10000     10000   10000  10000
unique         3         5       3      2
top     Suburban  Divorced  Female     No
freq        3346      2092    5025   7350
```

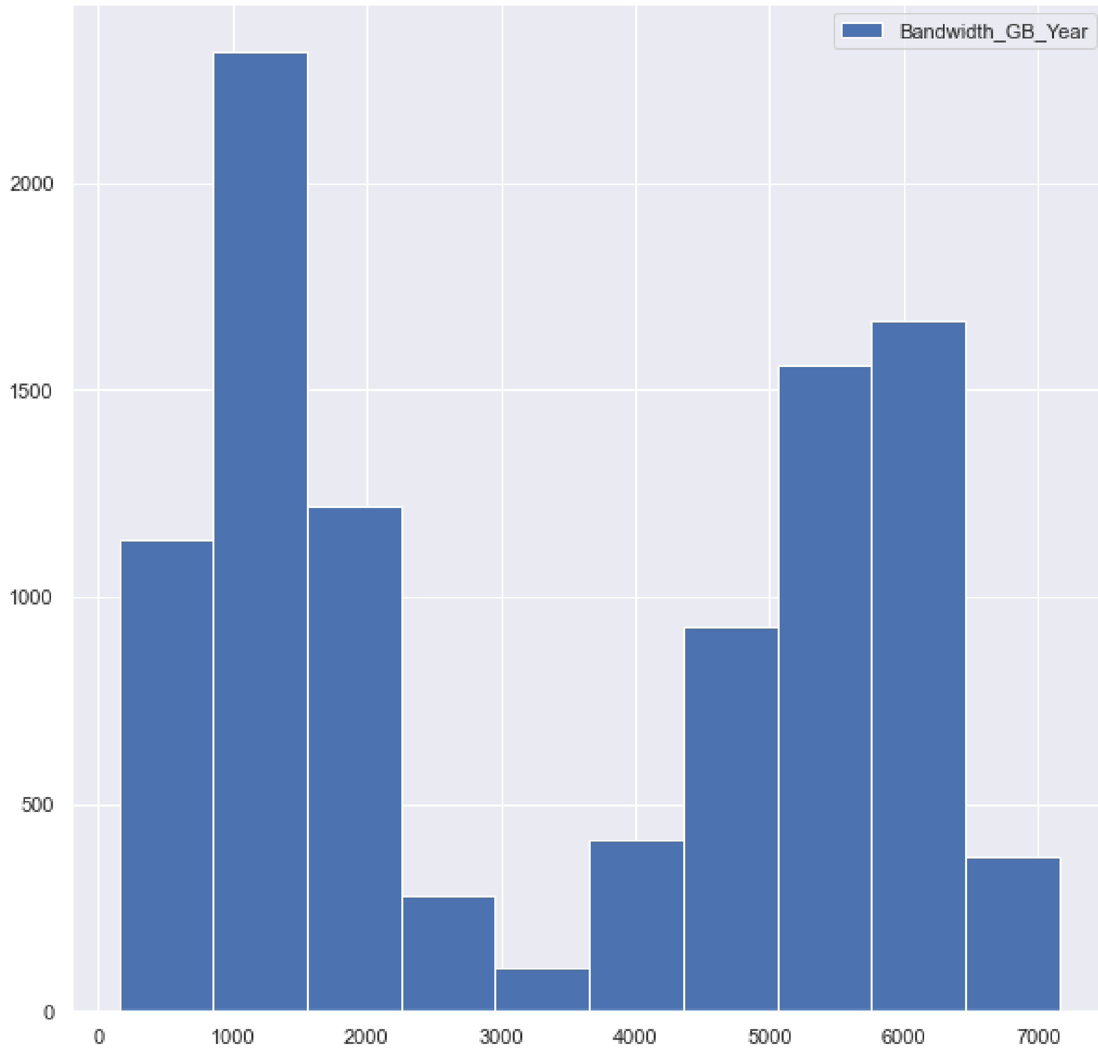
[12]:
```python
# Initialize figure size settings
plt.rcParams['figure.figsize'] = [10, 10]
```

[13]:
```python
# Display histogram plots for distribution of continuous variables
df_data.hist()
```

[13]:
```
array([[<AxesSubplot:title={'center':'Children'}>,
        <AxesSubplot:title={'center':'Age'}>,
        <AxesSubplot:title={'center':'Income'}>],
       [<AxesSubplot:title={'center':'Outage_sec_perweek'}>,
        <AxesSubplot:title={'center':'Yearly_equip_failure'}>,
        <AxesSubplot:title={'center':'Tenure'}>],
       [<AxesSubplot:title={'center':'MonthlyCharge'}>,
        <AxesSubplot:title={'center':'Bandwidth_GB_Year'}>,
        <AxesSubplot:>]], dtype=object)
```

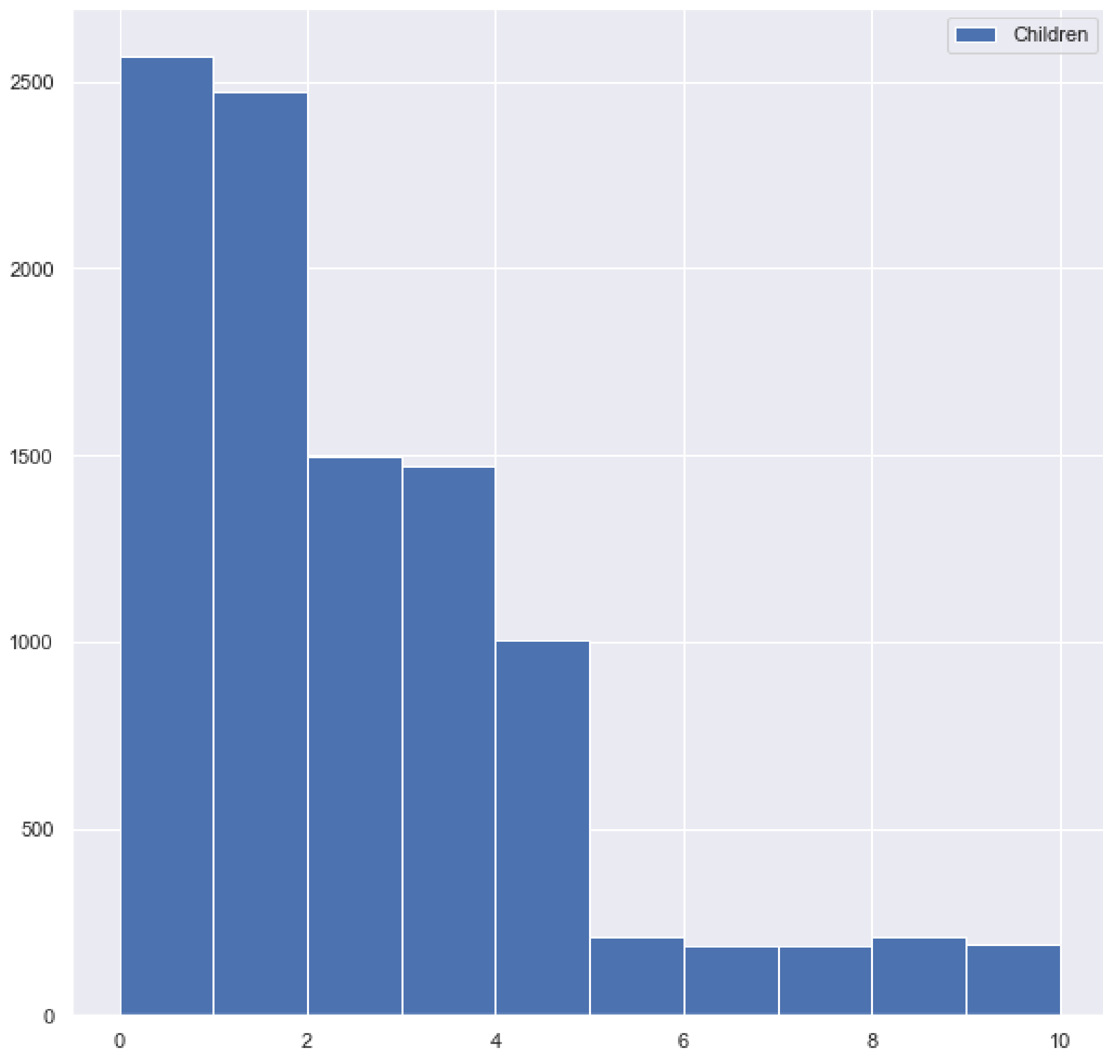


[14]:
```
# Display histogram plot and summary statistics for Bandwidth_GB_Year
df_data['Bandwidth_GB_Year'].hist(legend = True)
plt.show()
df_data['Bandwidth_GB_Year'].describe()
```



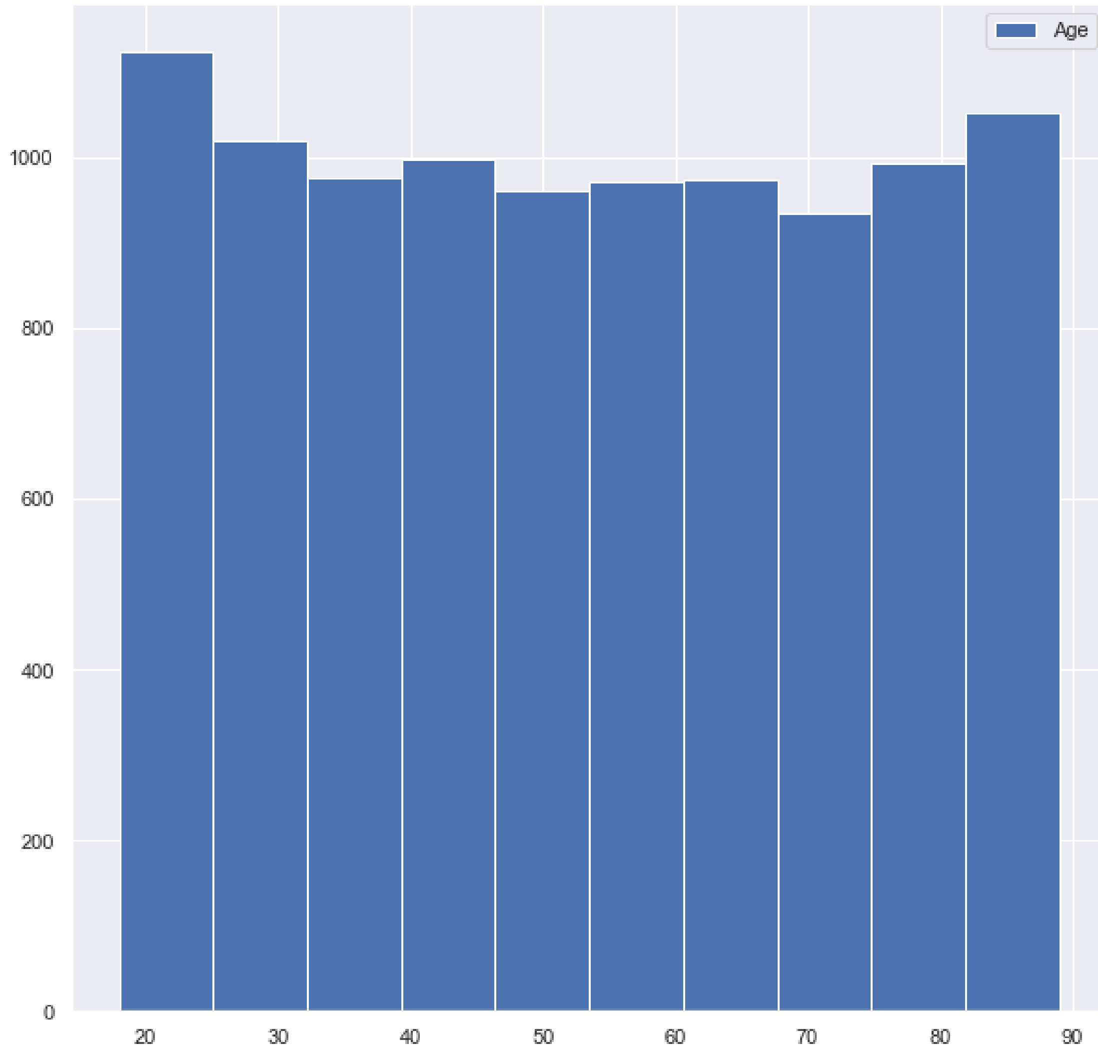
[14]:
```
count    10000.000000
mean      3392.341550
std       2185.294852
min        155.506715
25%       1236.470827
50%       3279.536903
75%       5586.141370
max       7158.981530
Name: Bandwidth_GB_Year, dtype: float64
```

[15]:
```
# Display histogram plot and summary statistics for Children
df_data['Children'].hist(legend = True)
plt.show()
df_data['Children'].describe()
```



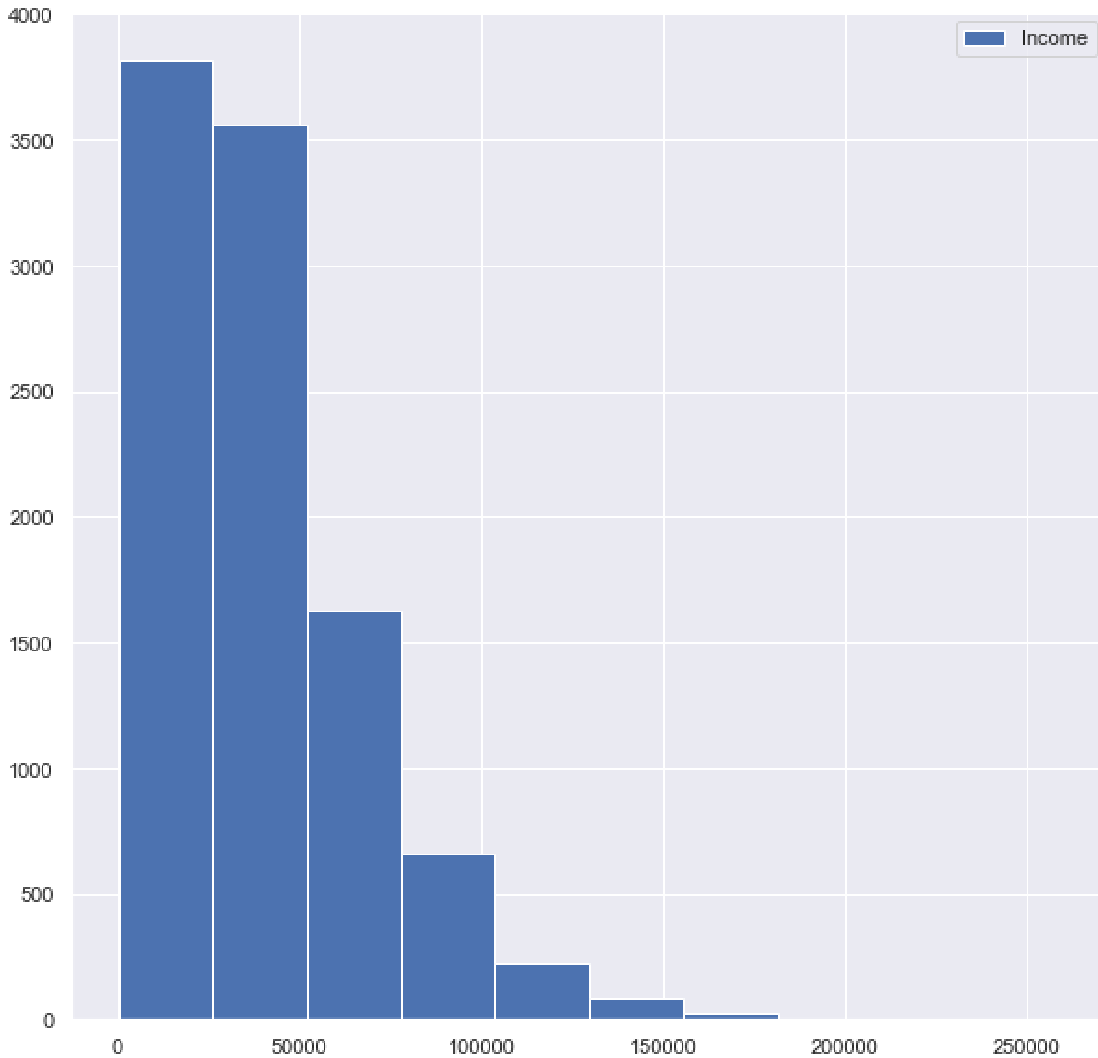

```
[15]: count    10000.0000
      mean         2.0877
      std          2.1472
      min          0.0000
      25%          0.0000
      50%          1.0000
      75%          3.0000
      max         10.0000
      Name: Children, dtype: float64
```

```
[16]: # Display histogram plot and summary statistics for Age
      df_data['Age'].hist(legend = True)
      plt.show()
      df_data['Age'].describe()
```



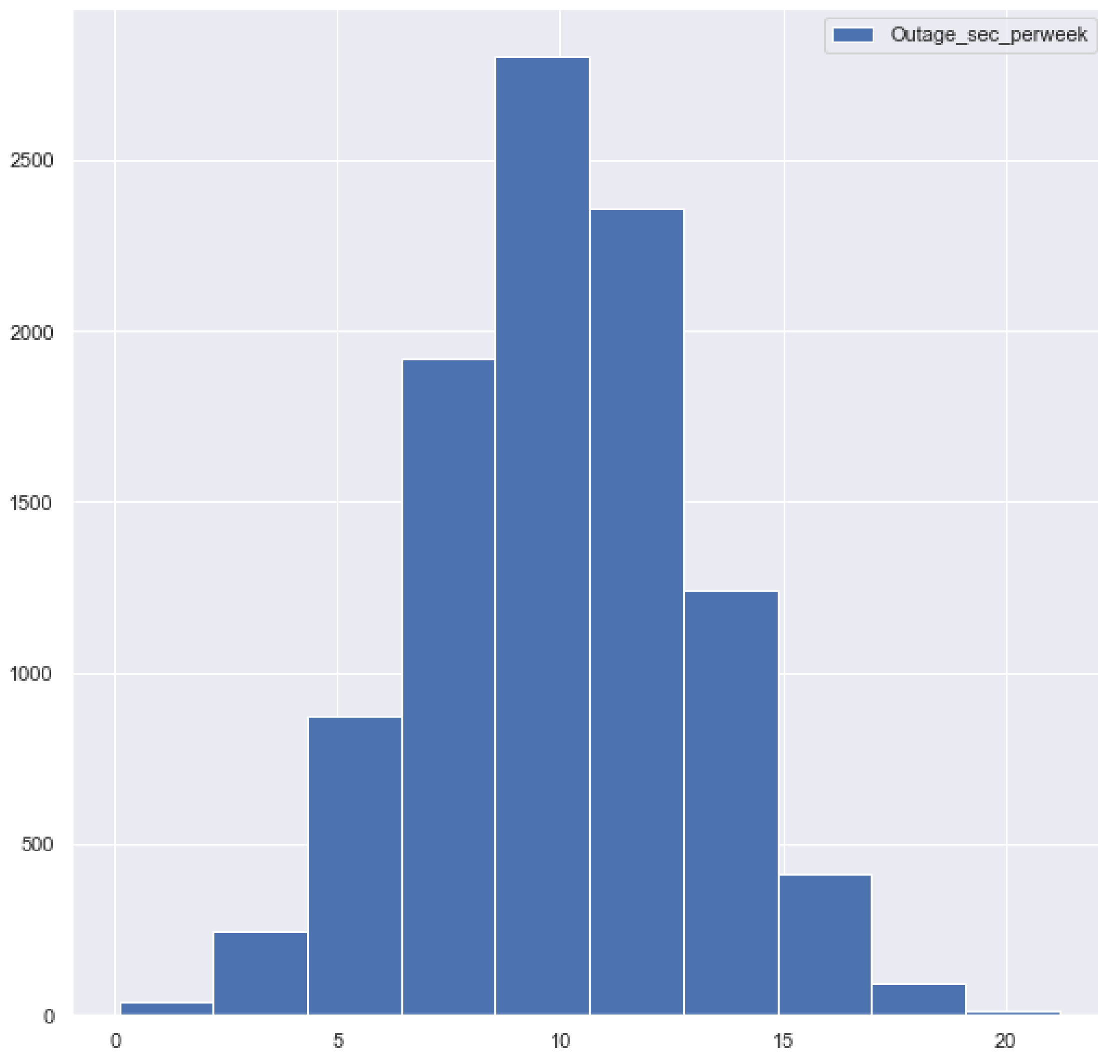
[16]:
```
count    10000.000000
mean        53.078400
std         20.698882
min         18.000000
25%         35.000000
50%         53.000000
75%         71.000000
max         89.000000
Name: Age, dtype: float64
```

[17]:
```
# Display histogram plot and summary statistics for Income
df_data['Income'].hist(legend = True)
plt.show()
df_data['Income'].describe()
```



[17]:
```
count     10000.000000
mean      39806.926771
std       28199.916702
min         348.670000
25%       19224.717500
50%       33170.605000
75%       53246.170000
max      258900.700000
Name: Income, dtype: float64
```

[18]:
```
# Display histogram plot and summary statistics for Outage_sec_perweek
df_data['Outage_sec_perweek'].hist(legend = True)
plt.show()
df_data['Outage_sec_perweek'].describe()
```

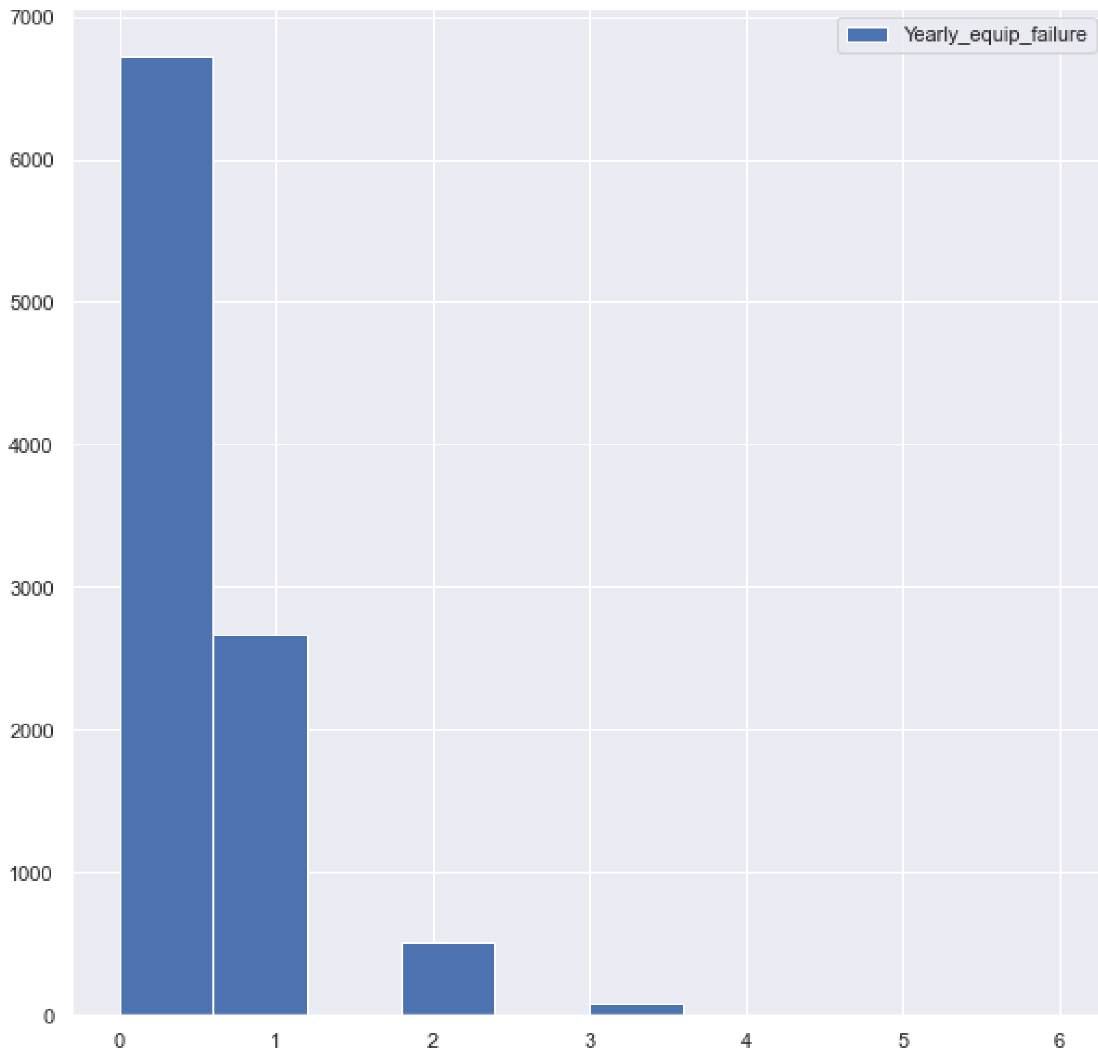


[18]:

```
count    10000.000000
mean        10.001848
std          2.976019
min          0.099747
25%          8.018214
50%         10.018560
75%         11.969485
max         21.207230
Name: Outage_sec_perweek, dtype: float64
```

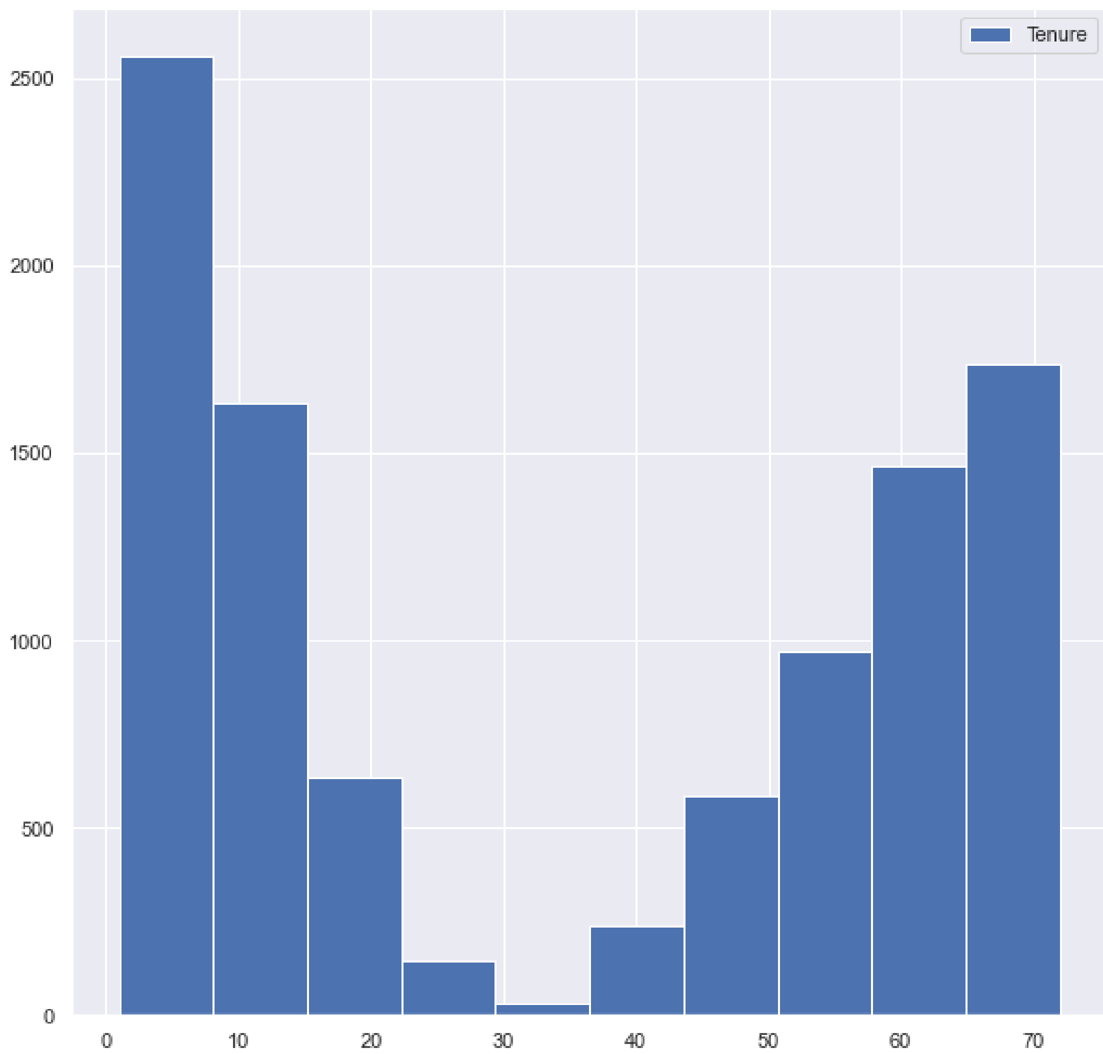
[19]:

```
# Display histogram plot and summary statistics for Yearly_equip_failure
df_data['Yearly_equip_failure'].hist(legend = True)
plt.show()
df_data['Yearly_equip_failure'].describe()
```



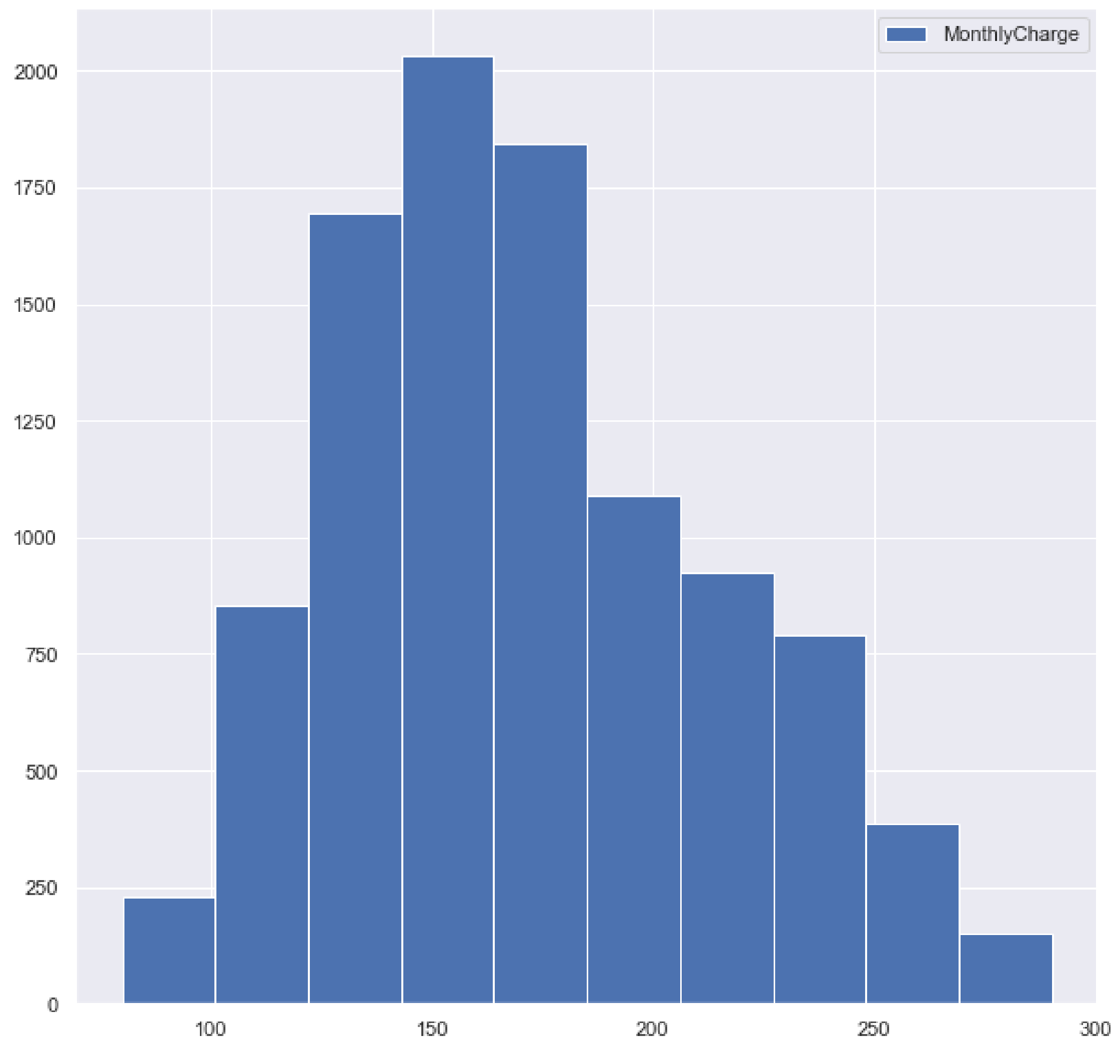
[19]:
```
count    10000.000000
mean         0.398000
std          0.635953
min          0.000000
25%          0.000000
50%          0.000000
75%          1.000000
max          6.000000
Name: Yearly_equip_failure, dtype: float64
```

[20]:
```
# Display histogram plot and summary statistics for Tenure
df_data['Tenure'].hist(legend = True)
plt.show()
df_data['Tenure'].describe()
```



[20]:
```
count    10000.000000
mean        34.526188
std         26.443063
min          1.000259
25%          7.917694
50%         35.430507
75%         61.479795
max         71.999280
Name: Tenure, dtype: float64
```

[21]:
```
# Display histogram plot and summary statistics for MonthlyCharge
df_data['MonthlyCharge'].hist(legend = True)
plt.show()
df_data['MonthlyCharge'].describe()
```

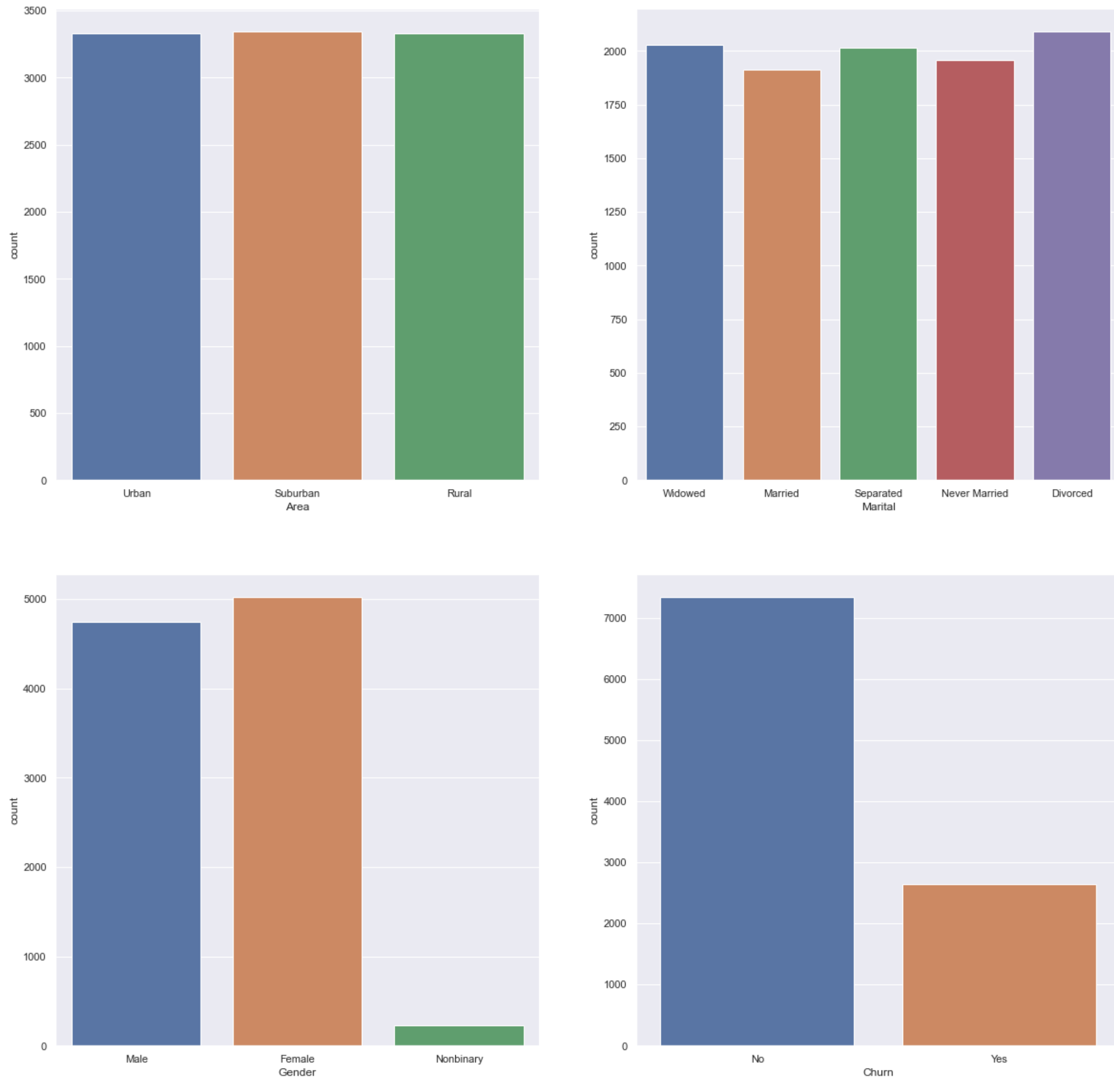


[21]:
```
count    10000.000000
mean       172.624816
std         42.943094
min         79.978860
25%        139.979239
50%        167.484700
75%        200.734725
max        290.160419
Name: MonthlyCharge, dtype: float64
```

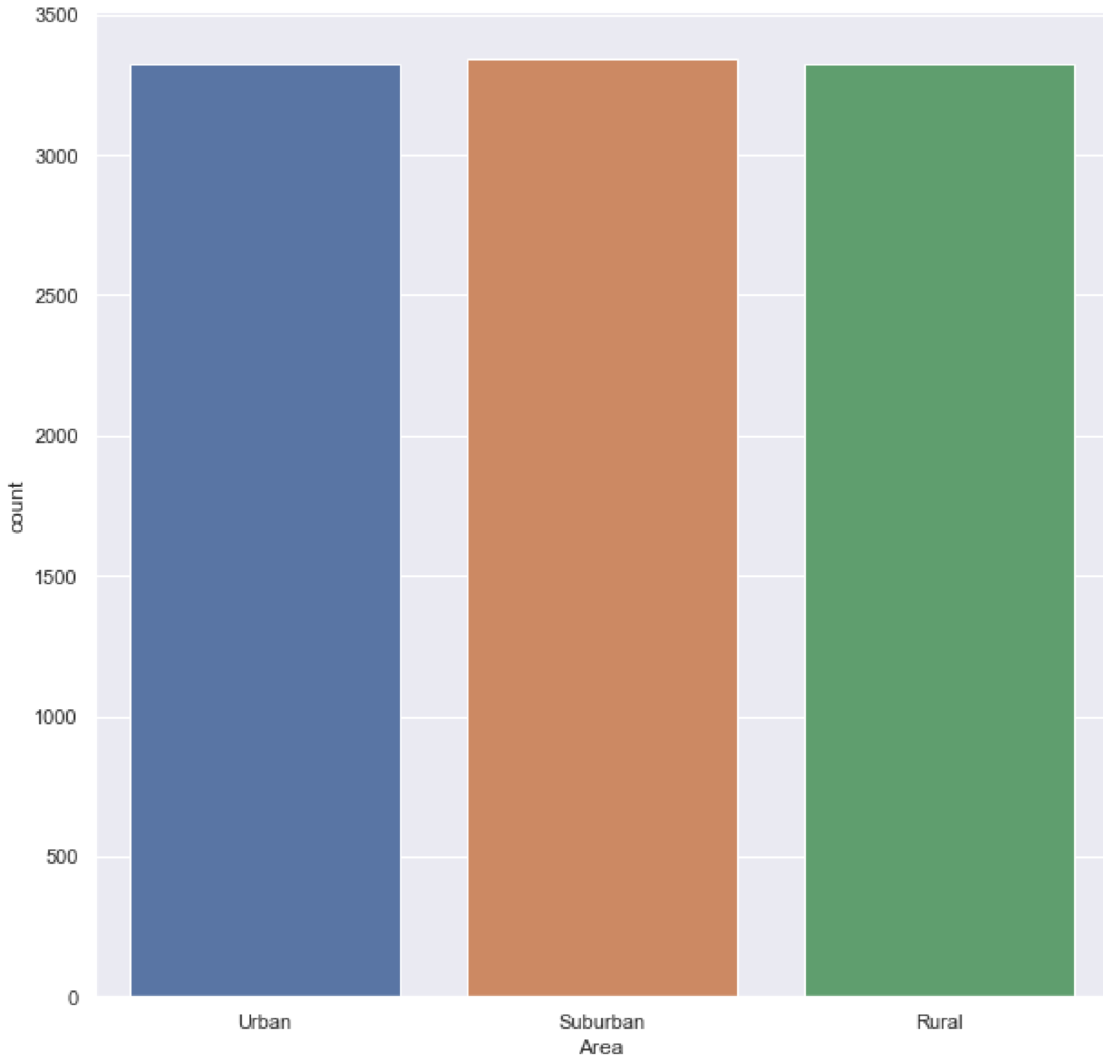
[22]:
```
# Display countplots for distribution of categorical variables
fig, ax = plt.subplots(figsize = (20,20), ncols = 2, nrows = 2)
sns.countplot(x='Area', data=df_data, ax = ax[0][0])
sns.countplot(x='Marital', data=df_data, ax = ax[0][1])
sns.countplot(x='Gender', data=df_data, ax = ax[1][0])
```

```
sns.countplot(x='Churn', data=df_data, ax = ax[1][1])
```

[22]: <AxesSubplot:xlabel='Churn', ylabel='count'>

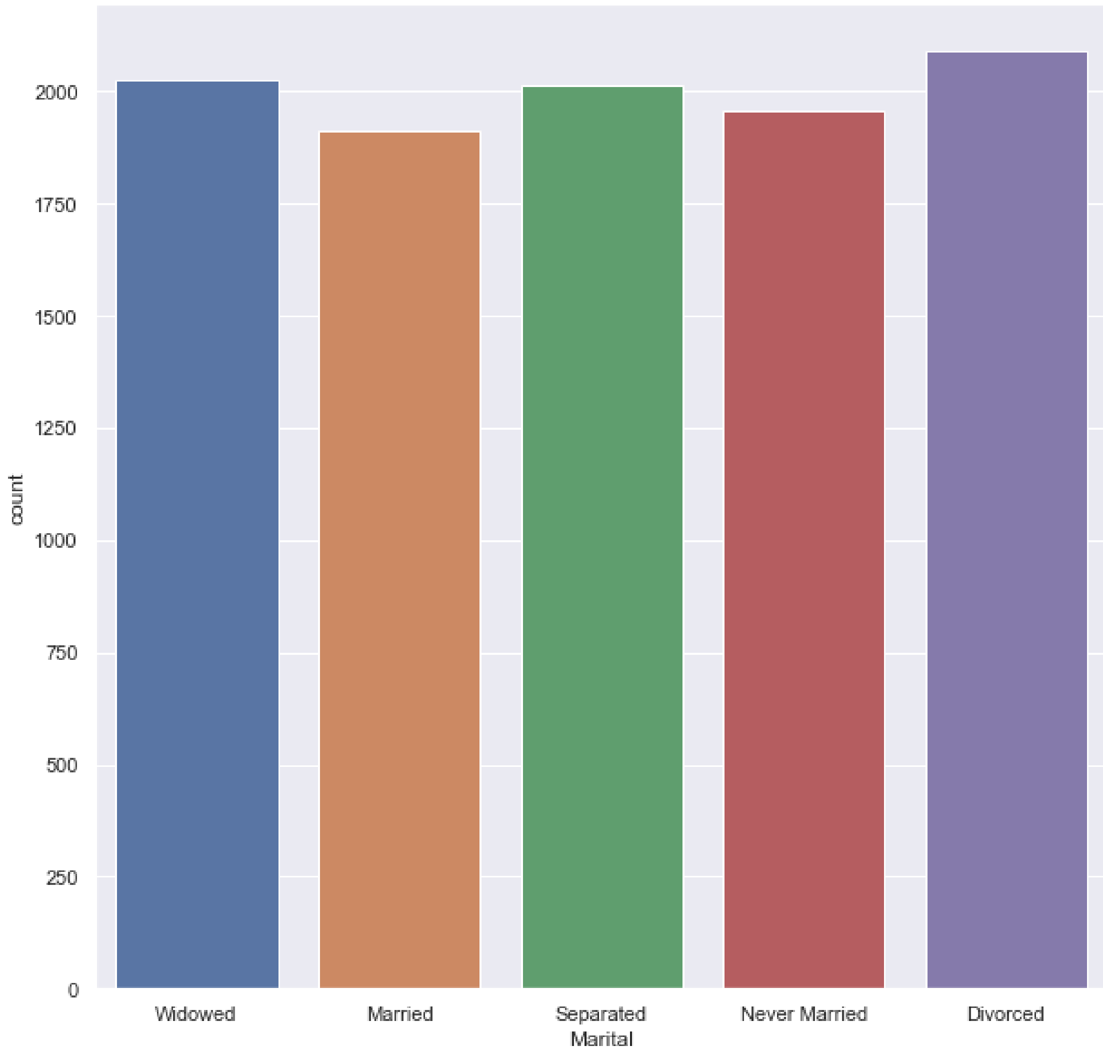


[23]:
```
# Display countplot and summary statistics for Area
sns.countplot(x='Area', data=df_data)
plt.show()
df_data['Area'].describe()
```



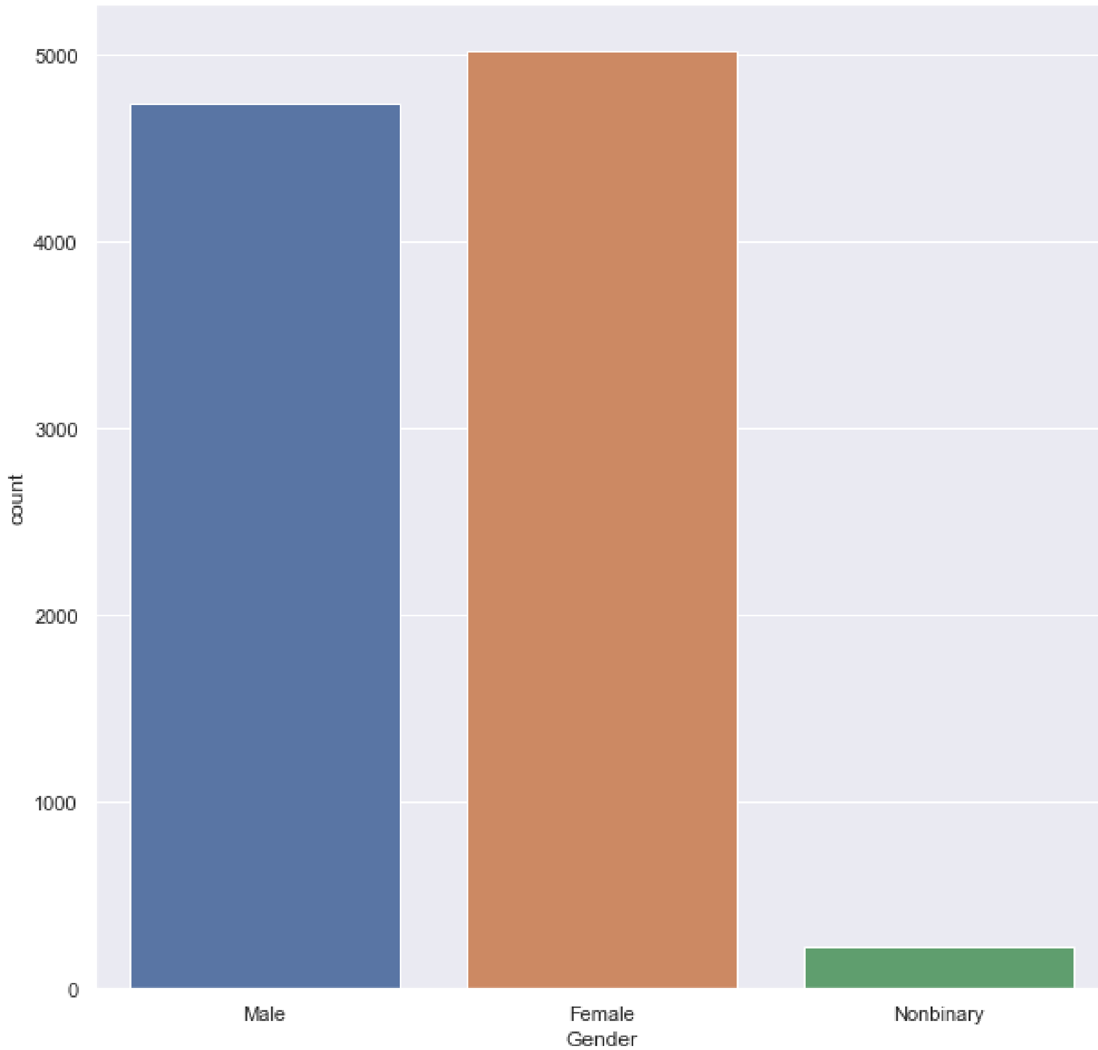
[23]:
```
count         10000
unique            3
top        Suburban
freq           3346
Name: Area, dtype: object
```

[24]:
```
# Display countplot and summary statistics for Marital
sns.countplot(x='Marital', data=df_data)
plt.show()
df_data['Marital'].describe()
```



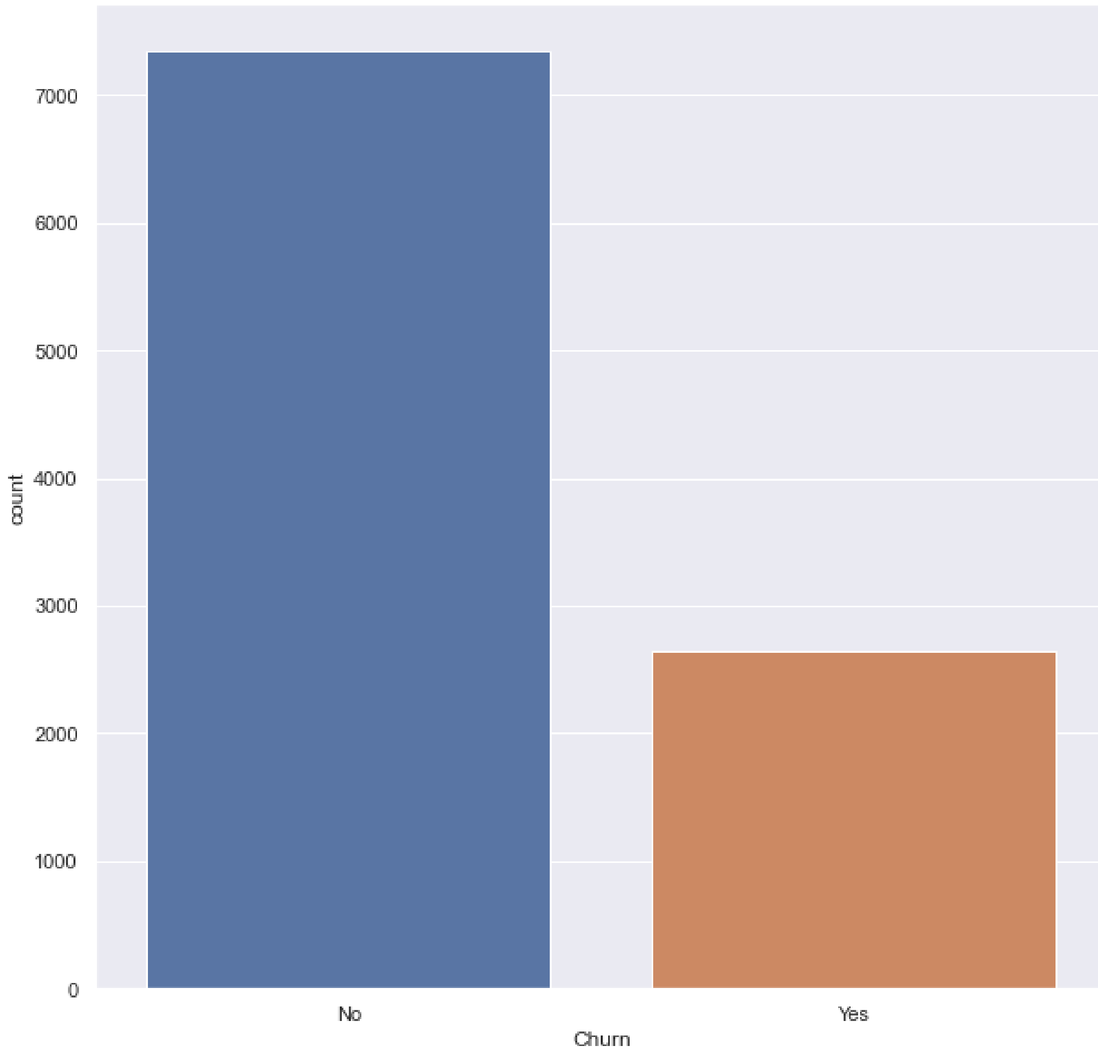
[24]:
```
count          10000
unique             5
top         Divorced
freq            2092
Name: Marital, dtype: object
```

[25]:
```
# Display countplot and summary statistics for Gender
sns.countplot(x='Gender', data=df_data)
plt.show()
df_data['Gender'].describe()
```



[25]:
```
count       10000
unique          3
top        Female
freq         5025
Name: Gender, dtype: object
```

[26]:
```
# Display countplot and summary statistics for Churn
sns.countplot(x='Churn', data=df_data)
plt.show()
df_data['Churn'].describe()
```




```
[26]: count      10000
      unique         2
      top           No
      freq        7350
      Name: Churn, dtype: object
```

3.3 Further Preparation Steps

I will make some adjustments to my data types to make my variables easier to work with. Conversion of "object" types as "category" in particular will lend itself to more efficient conversion of categorical variables to numeric.

[27]:
```
# Reassign data types
for col in df_data:
    if df_data[col].dtypes == 'object':
        df_data[col] = df_data[col].astype('category')
    if df_data[col].dtypes == 'int64':
        df_data[col] = df_data[col].astype(int)
    if df_data[col].dtypes == 'float64':
        df_data[col] = df_data[col].astype(float)
```

[28]:
```
# Display dataset info and observe data type changes
df_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Area                  10000 non-null  category
 1   Children              10000 non-null  int32
 2   Age                   10000 non-null  int32
 3   Income                10000 non-null  float64
 4   Marital               10000 non-null  category
 5   Gender                10000 non-null  category
 6   Churn                 10000 non-null  category
 7   Outage_sec_perweek    10000 non-null  float64
 8   Yearly_equip_failure  10000 non-null  int32
 9   Tenure                10000 non-null  float64
 10  MonthlyCharge         10000 non-null  float64
 11  Bandwidth_GB_Year     10000 non-null  float64
dtypes: category(4), float64(5), int32(3)
memory usage: 547.6 KB
```

Here I will use the cat.codes accessor to perform label encoding on three of my categorical variables.

[29]:
```
# Use cat.codes for label encoding of 4 categorical variables
df_data['Area_cat'] = df_data['Area'].cat.codes
df_data['Marital_cat'] = df_data['Marital'].cat.codes
df_data['Gender_cat'] = df_data['Gender'].cat.codes
df_data['Churn_cat'] = df_data['Churn'].cat.codes
```

[30]:
```
# Display dataset top 5 rows from label encoded variables
df_data[['Area', 'Marital', 'Gender', 'Churn', 'Area_cat', 'Marital_cat',␣
 ↪'Gender_cat', 'Churn_cat']].head()
```

[30]:
```
       Area   Marital  Gender Churn  Area_cat  Marital_cat  Gender_cat  \
0     Urban   Widowed    Male    No         2            4           1
```

```
1     Urban    Married  Female   Yes          2            1            0
2     Urban    Widowed  Female    No          2            4            0
3  Suburban    Married    Male    No          1            1            1
4  Suburban  Separated    Male   Yes          1            3            1

   Churn_cat
0          0
1          1
2          0
3          0
4          1
```

3.4 Univariate and Bivariate Visualizations

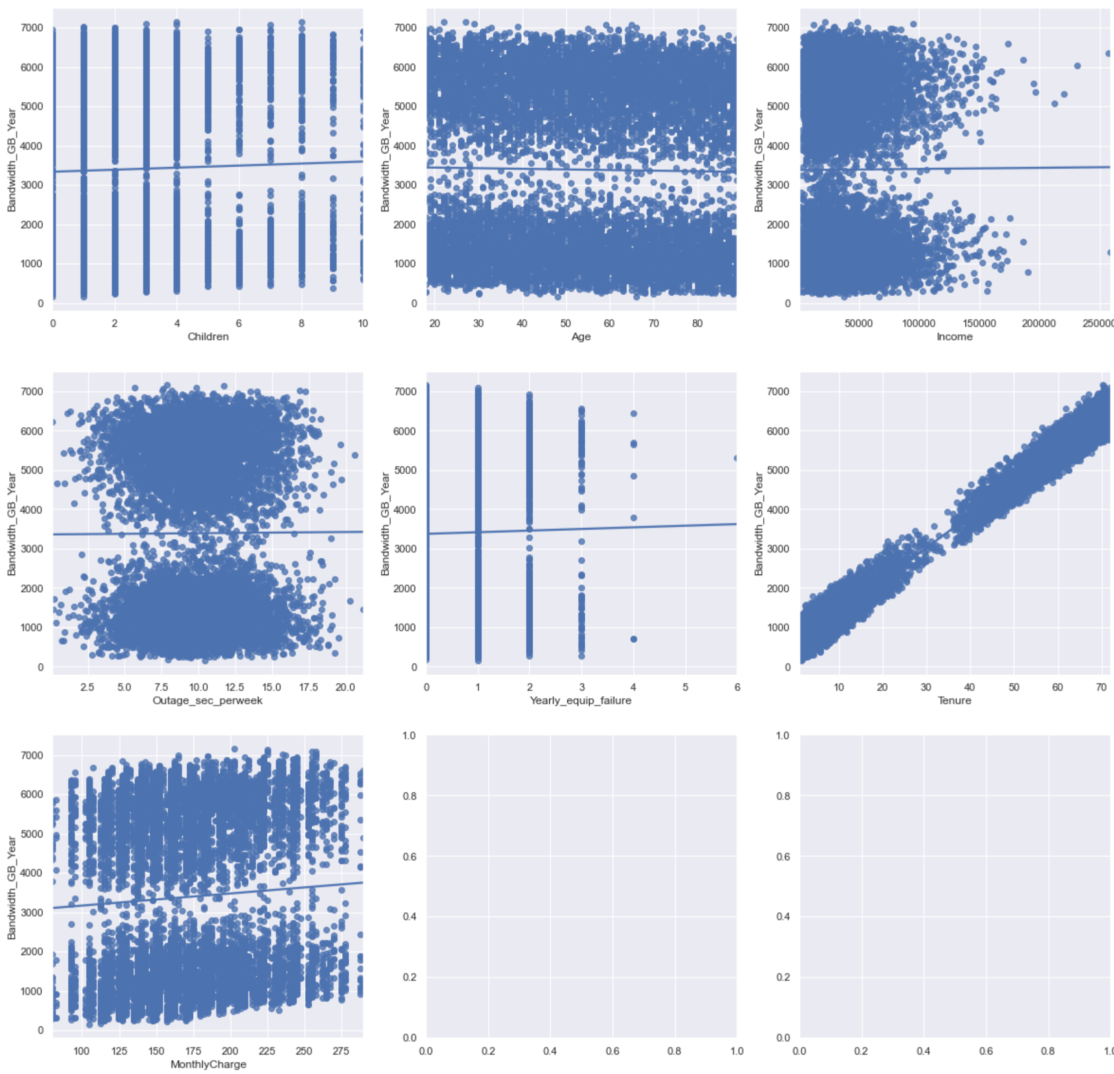
Univariate analysis of each variable can be seen above in section 2 of part III, "Data Preparation". I will make use of Seaborn's regplot() function for bivariate analysis of continuous variables, and Seaborn's boxplot() function for the categorical variables. Each independent variable is paired against my dependent variable, "Bandwidth_GB_Year".

[31]:

```
# Display regplots for bivariate statistical analysis of continuous variables -␣
↪dependent variable = Bandwidth_GB_Year
fig, ax = plt.subplots(figsize = (20,20), ncols = 3, nrows = 3)
sns.regplot(x="Children",
            y="Bandwidth_GB_Year",
            data=df_data,
            ax = ax[0][0],
            ci=None)
sns.regplot(x="Age",
            y="Bandwidth_GB_Year",
            data=df_data,
            ax = ax[0][1],
            ci=None)
sns.regplot(x="Income",
            y="Bandwidth_GB_Year",
            data=df_data,
            ax = ax[0][2],
            ci=None)
sns.regplot(x="Outage_sec_perweek",
            y="Bandwidth_GB_Year",
            data=df_data,
            ax = ax[1][0],
            ci=None)
sns.regplot(x="Yearly_equip_failure",
            y="Bandwidth_GB_Year",
            data=df_data,
            ax = ax[1][1],
```

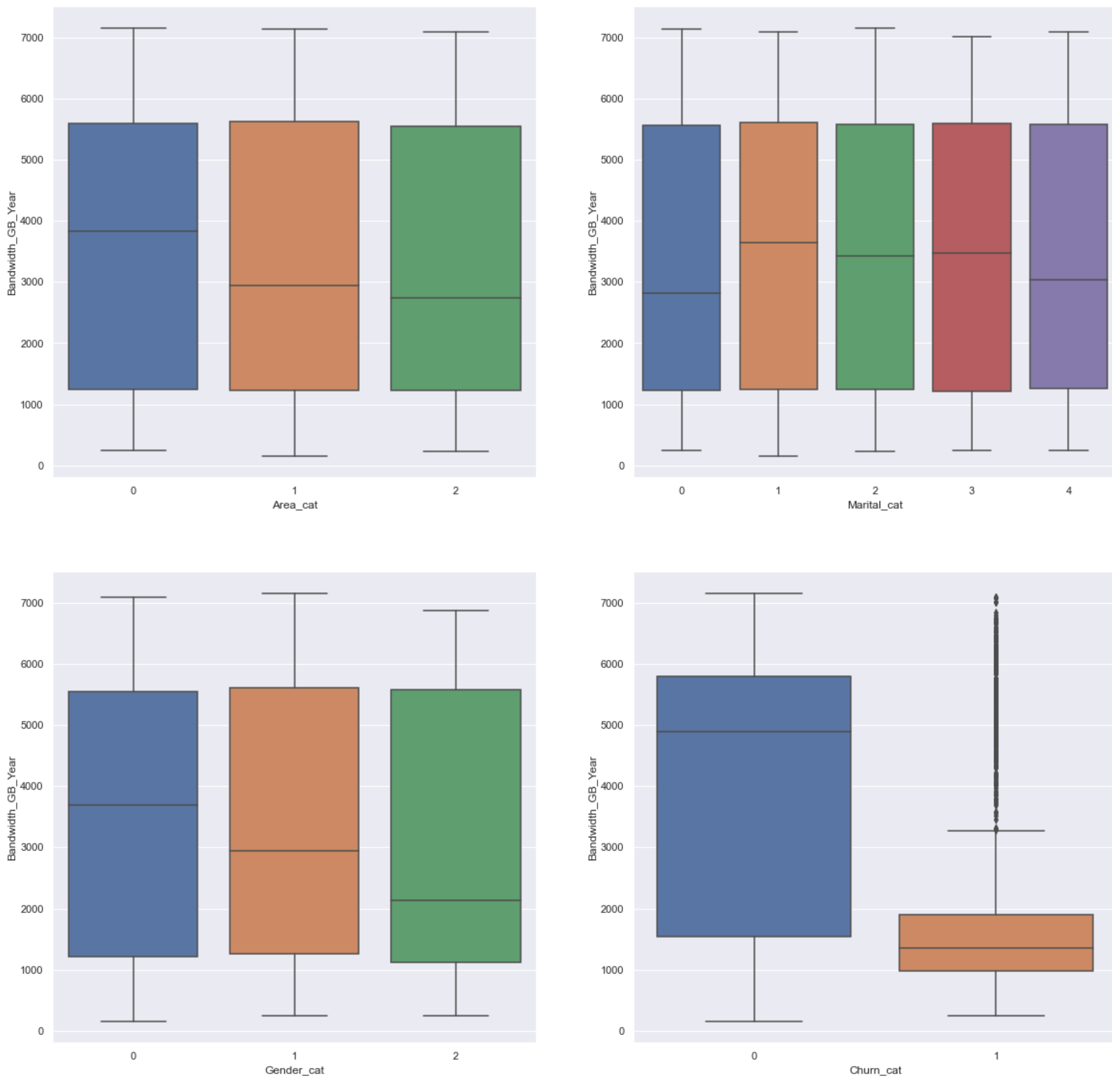
```
              ci=None)
sns.regplot(x="Tenure",
              y="Bandwidth_GB_Year",
              data=df_data,
              ax = ax[1][2],
              ci=None)
sns.regplot(x="MonthlyCharge",
              y="Bandwidth_GB_Year",
              data=df_data,
              ax = ax[2][0],
              ci=None)
```

[31]: <AxesSubplot:xlabel='MonthlyCharge', ylabel='Bandwidth_GB_Year'>



[32]:
```python
# Display boxplots for bivariate analysis of categorical variables - dependent␣
↪variable = Bandwidth_GB_Year
fig, ax = plt.subplots(figsize = (20, 20), ncols = 2, nrows = 2)
sns.boxplot(x = 'Area_cat', y = 'Bandwidth_GB_Year', data = df_data, ax =␣
↪ax[0][0])
sns.boxplot(x = 'Marital_cat', y = 'Bandwidth_GB_Year', data = df_data, ax =␣
↪ax[0][1])
sns.boxplot(x = 'Gender_cat', y = 'Bandwidth_GB_Year', data = df_data, ax =␣
↪ax[1][0])
sns.boxplot(x = 'Churn_cat', y = 'Bandwidth_GB_Year', data = df_data, ax =␣
↪ax[1][1])
```

[32]: <AxesSubplot:xlabel='Churn_cat', ylabel='Bandwidth_GB_Year'>



3.5 Copy of Prepared Data Set

Below is the code used to export the prepared data set to csv format.

[33]:
```
# Export prepared dataframe to csv
df_data.to_csv(r'C:\Users\wstul\d208\churn_clean_perpared.csv')
```

4 Part IV: Model Comparison and Analysis

4.1 Initial Multiple Regression Model

Below I will create an initial multiple regression model and display its summary info.

[34]:
```
# Create initial model and display summary
mdl_bandwidth_vs_all = ols("Bandwidth_GB_Year ~ Area_cat + Children + Age +␣
↪Income + Marital_cat + Gender_cat + Churn_cat + \
                        Outage_sec_perweek + Yearly_equip_failure +␣
↪MonthlyCharge + Tenure", data=df_data).fit()
print(mdl_bandwidth_vs_all.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Bandwidth_GB_Year   R-squared:                       0.990
Model:                            OLS   Adj. R-squared:                  0.990
Method:                 Least Squares   F-statistic:                 8.719e+04
Date:                Sat, 14 May 2022   Prob (F-statistic):               0.00
Time:                        20:55:15   Log-Likelihood:                -68209.
No. Observations:               10000   AIC:                         1.364e+05
Df Residuals:                    9988   BIC:                         1.365e+05
Df Model:                          11
Covariance Type:            nonrobust
================================================================================
========
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
Intercept               99.7584     14.780      6.749      0.000      70.786
128.731
Area_cat                 0.3384      2.722      0.124      0.901      -4.998
5.675
Children                30.8539      1.035     29.824      0.000      28.826
32.882
Age                     -3.3329      0.107    -31.047      0.000      -3.543
-3.122
```

```
Income                    0.0001   7.88e-05      1.334      0.182   -4.93e-05
0.000
Marital_cat              -3.3111      1.555     -2.129      0.033      -6.359
-0.263
Gender_cat               52.9976      4.085     12.975      0.000      44.991
61.004
Churn_cat               128.6483      6.358     20.233      0.000     116.184
141.112
Outage_sec_perweek       -0.2474      0.746     -0.332      0.740      -1.710
1.216
Yearly_equip_failure      0.6557      3.492      0.188      0.851      -6.189
7.501
MonthlyCharge             2.7781      0.057     48.635      0.000       2.666
2.890
Tenure                   83.0711      0.098    843.690      0.000      82.878
83.264
==============================================================================
Omnibus:                     7610.424   Durbin-Watson:                   1.987
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              918.783
Skew:                           0.451   Prob(JB):                    3.08e-200
Kurtosis:                       1.820   Cond. No.                     3.26e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.26e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

4.2 Reducing the Initial Model

The initial model has a high r-squared score as is, but this is typically the case when a high number of independent variables are included. I will aim to reduce the model by eliminating variables not suitable for this multiple regression model, using statistical analysis in my selection process.

To begin I will look at some metrics for the current model.

[35]:
```
# Display MSE, RSE, Rsquared and Adjusted Rsquared for initial model
mse_all = mdl_bandwidth_vs_all.mse_resid
print('MSE of original model: ', mse_all)
rse_all = np.sqrt(mse_all)
print('RSE of original model: ', rse_all)
print('Rsquared of original model: ', mdl_bandwidth_vs_all.rsquared)
print('Rsquared Adjusted of original model: ', mdl_bandwidth_vs_all.
 ↪rsquared_adj)
```

```
MSE of original model:  49276.41635896182
```

```
RSE of original model:  221.9829190702785
Rsquared of original model:  0.989692793051259
Rsquared Adjusted of original model:  0.989681441501756
```

As I proceed through my reduction process, my aim will be to keep r-squared and residual standard error scores close to the initial model's performance. Higher r-squared scores are considered better, while lower RSE scores are preferable.

First I will perform a variance inflation factor analysis for all features currently in the model.

[36]:
```
# Perform variance inflation factor analysis for initial feature set
X = df_data[['Area_cat', 'Children', 'Age', 'Income', 'Marital_cat',␣
 ↪'Gender_cat', 'Churn_cat', 'Outage_sec_perweek',
            'Yearly_equip_failure', 'Tenure', 'MonthlyCharge']]
vif_data = pd.DataFrame()
vif_data['IndVar'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(len(X.
 ↪columns))]
print(vif_data)
```

```
                  IndVar        VIF
0               Area_cat   2.425228
1               Children   1.897584
2                    Age   6.461163
3                 Income   2.847813
4            Marital_cat   2.810940
5             Gender_cat   1.879759
6              Churn_cat   2.166807
7     Outage_sec_perweek   9.250534
8   Yearly_equip_failure   1.382208
9                 Tenure   3.653279
10         MonthlyCharge  14.379482
```

Right away, I can see very high VIF scores for two variables, MonthlyCharge and Outage_sec_perweek. High VIF values (usually greater than 5) indicate a high degree of multicollinearity with other variables in the model. This reduces the model accuracy, so I will drop these two variables from the set and repeat my VIF analysis.

[37]:
```
# Drop 2 high VIF variables
X = X.drop(['Outage_sec_perweek', 'MonthlyCharge'], axis = 1)
```

[38]:
```
# Perform variance inflation factor analysis for trimmed feature set
vif_data = pd.DataFrame()
vif_data['IndVar'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(len(X.
 ↪columns))]
```

```python
print(vif_data)
```

```
                IndVar       VIF
0             Area_cat  2.315820
1             Children  1.834000
2                  Age  4.959615
3               Income  2.675762
4          Marital_cat  2.652144
5           Gender_cat  1.820088
6            Churn_cat  1.616385
7  Yearly_equip_failure  1.367955
8               Tenure  2.954382
```

Though the Age variable is higher than the other variables after my second VIF analysis, it is below 5 and so merits inclusion as I continue.

Next I will create a dataframe based on my remaining variables and generate a correlation table to check for other signs of collinearity.

[39]:
```python
# Create temporary dataframe of trimmed feature set + dependent variable and␣
↪display correlation table
df_data_reduced = pd.DataFrame(df_data[['Area_cat', 'Children', 'Age',␣
↪'Income', 'Marital_cat', 'Gender_cat', 'Churn_cat', 'Yearly_equip_failure',␣
↪'Tenure', 'Bandwidth_GB_Year']])
df_data_reduced.corr()
```

[39]:

	Area_cat	Children	Age	Income	Marital_cat
Area_cat	1.000000	-0.007879	0.011745	0.002557	0.013733
Children	-0.007879	1.000000	-0.029732	0.009942	0.000045
Age	0.011745	-0.029732	1.000000	-0.004091	-0.009721
Income	0.002557	0.009942	-0.004091	1.000000	-0.005045
Marital_cat	0.013733	0.000045	-0.009721	-0.005045	1.000000
Gender_cat	0.004057	0.006032	-0.005660	-0.018436	-0.008360
Churn_cat	0.014166	-0.004264	0.005630	0.005937	0.012716
Yearly_equip_failure	-0.006554	0.007321	0.008577	0.005423	0.001183
Tenure	-0.016615	-0.005091	0.016979	0.002114	0.003241
Bandwidth_GB_Year	-0.016575	0.025585	-0.014724	0.003674	0.001499

	Gender_cat	Churn_cat	Yearly_equip_failure	Tenure
Area_cat	0.004057	0.014166	-0.006554	-0.016615
Children	0.006032	-0.004264	0.007321	-0.005091
Age	-0.005660	0.005630	0.008577	0.016979
Income	-0.018436	0.005937	0.005423	0.002114
Marital_cat	-0.008360	0.012716	0.001183	0.003241
Gender_cat	1.000000	0.023919	0.014750	-0.016051
Churn_cat	0.023919	1.000000	-0.015927	-0.485475
Yearly_equip_failure	0.014750	-0.015927	1.000000	0.012435

```
Tenure                -0.016051  -0.485475              0.012435  1.000000
Bandwidth_GB_Year     -0.001469  -0.441669              0.012034  0.991495

                      Bandwidth_GB_Year
Area_cat                      -0.016575
Children                       0.025585
Age                           -0.014724
Income                         0.003674
Marital_cat                    0.001499
Gender_cat                    -0.001469
Churn_cat                     -0.441669
Yearly_equip_failure           0.012034
Tenure                         0.991495
Bandwidth_GB_Year              1.000000
```

In my correlation table, "Tenure" shares a high collinearity with "Bandwidth_GB_Year", but this is acceptable as "Bandwidth_GB_Year" is our dependent variable. No other instances of high collinearity appear.

I will create a reduced model based on my remaining variables to see how our statistics look.

[40]:
```
# Create first reduced model and display summary
mdl_bandwidth_vs_reduced = ols("Bandwidth_GB_Year ~ Area_cat + Children + Age +␣
↪Income + Marital_cat + Gender_cat + Churn_cat + \
                        Yearly_equip_failure + Tenure", data=df_data).fit()
print(mdl_bandwidth_vs_reduced.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Bandwidth_GB_Year   R-squared:                       0.987
Model:                            OLS   Adj. R-squared:                  0.987
Method:                 Least Squares   F-statistic:                 8.596e+04
Date:                Sat, 14 May 2022   Prob (F-statistic):               0.00
Time:                        20:55:28   Log-Likelihood:                -69272.
No. Observations:               10000   AIC:                         1.386e+05
Df Residuals:                    9990   BIC:                         1.386e+05
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
========
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
Intercept              507.8800     10.720     47.377      0.000     486.867
528.893
Area_cat                 0.4784      3.027      0.158      0.874      -5.456
```

```
6.413
Children                 30.5015      1.150     26.514      0.000      28.246
32.756
Age                      -3.3109      0.119    -27.736      0.000      -3.545
-3.077
Income                 7.853e-05   8.76e-05      0.897      0.370   -9.31e-05
0.000
Marital_cat              -4.0706      1.729     -2.354      0.019      -7.459
-0.682
Gender_cat               53.2392      4.542     11.722      0.000      44.336
62.142
Churn_cat               259.9755      6.402     40.611      0.000     247.427
272.524
Yearly_equip_failure      0.2252      3.883      0.058      0.954      -7.387
7.837
Tenure                   84.1201      0.107    787.392      0.000      83.911
84.330
==============================================================================
Omnibus:                      480.226   Durbin-Watson:                   1.982
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              332.027
Skew:                           0.337   Prob(JB):                     7.97e-73
Kurtosis:                       2.414   Cond. No.                     2.19e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.19e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[41]:
```python
# Display MSE, RSE for first reduced model
mse_reduced = mdl_bandwidth_vs_reduced.mse_resid
print('MSE of reduced model: ', mse_reduced)
rse_reduced = np.sqrt(mse_reduced)
print('RSE of reduced model: ', rse_reduced)
```

```
MSE of reduced model:  60936.347257474525
RSE of reduced model:  246.8528858601303
```

According to this summary, several variables exhibit high p-values, indicating no relationship between that variable and the dependent variable, "Bandwidth_GB_Year". A value greater than .05 is considered high. I will remove these variables from the model and once again evaluate the resulting summary and statistics.

[42]:
```python
# Create second reduced model and display summary
mdl_bandwidth_vs_features = ols("Bandwidth_GB_Year ~ Children + Age +␣
 ↪Marital_cat + Gender_cat + Churn_cat + Tenure", data=df_data).fit()
```

```python
print(mdl_bandwidth_vs_features.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Bandwidth_GB_Year   R-squared:                       0.987
Model:                            OLS   Adj. R-squared:                  0.987
Method:                 Least Squares   F-statistic:                 1.290e+05
Date:                Sat, 14 May 2022   Prob (F-statistic):               0.00
Time:                        20:55:32   Log-Likelihood:                -69273.
No. Observations:               10000   AIC:                         1.386e+05
Df Residuals:                    9993   BIC:                         1.386e+05
Df Model:                           6
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept     511.5803      9.586     53.369      0.000     492.790     530.370
Children       30.5109      1.150     26.528      0.000      28.256      32.765
Age            -3.3111      0.119    -27.744      0.000      -3.545      -3.077
Marital_cat    -4.0750      1.728     -2.358      0.018      -7.463      -0.687
Gender_cat     53.1699      4.540     11.711      0.000      44.270      62.070
Churn_cat     260.0266      6.400     40.628      0.000     247.481     272.572
Tenure         84.1205      0.107    787.559      0.000      83.911      84.330
==============================================================================
Omnibus:                      481.158   Durbin-Watson:                   1.982
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              332.204
Skew:                           0.336   Prob(JB):                     7.29e-73
Kurtosis:                       2.413   Cond. No.                         275.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[43]:
```python
# Display MSE, RSE, Rsquared and Adjusted Rsquared for second reduced model
mse_features = mdl_bandwidth_vs_features.mse_resid
print('MSE of reduced model: ', mse_features)
rse_features = np.sqrt(mse_features)
print('RSE of reduced model: ', rse_features)
print('Rsquared of reduced model: ', mdl_bandwidth_vs_features.rsquared)
print('Rsquared Adjusted of reduced model: ', mdl_bandwidth_vs_features.
 ↪rsquared_adj)
```

```
MSE of reduced model:  60923.13862410184
RSE of reduced model:  246.82613035110737
Rsquared of reduced model:  0.9872502548864599
Rsquared Adjusted of reduced model:  0.9872425996807478
```

4.3 Final Reduced Multiple Regression Model

At this point, I have eliminated any sources of multicollinearity and collinearity as well as variables exhibiting p-values that exceed .05. I will finalize the reduced model and check to see how it compares to my initial model which included all variables in the set.

[44]:
```
# Create final reduced model and display summary
mdl_bandwidth_vs_features_final = ols("Bandwidth_GB_Year ~ Children + Age +␣
↪Marital_cat + Gender_cat + Churn_cat + Tenure", data=df_data).fit()
print(mdl_bandwidth_vs_features_final.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Bandwidth_GB_Year   R-squared:                       0.987
Model:                            OLS   Adj. R-squared:                  0.987
Method:                 Least Squares   F-statistic:                 1.290e+05
Date:                Sat, 14 May 2022   Prob (F-statistic):               0.00
Time:                        20:55:36   Log-Likelihood:                -69273.
No. Observations:               10000   AIC:                         1.386e+05
Df Residuals:                    9993   BIC:                         1.386e+05
Df Model:                           6
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept     511.5803      9.586     53.369      0.000     492.790     530.370
Children       30.5109      1.150     26.528      0.000      28.256      32.765
Age            -3.3111      0.119    -27.744      0.000      -3.545      -3.077
Marital_cat    -4.0750      1.728     -2.358      0.018      -7.463      -0.687
Gender_cat     53.1699      4.540     11.711      0.000      44.270      62.070
Churn_cat     260.0266      6.400     40.628      0.000     247.481     272.572
Tenure         84.1205      0.107    787.559      0.000      83.911      84.330
==============================================================================
Omnibus:                      481.158   Durbin-Watson:                   1.982
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              332.204
Skew:                           0.336   Prob(JB):                     7.29e-73
Kurtosis:                       2.413   Cond. No.                         275.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[45]:
```
# Display MSE, RSE, Rsquared and Adjusted Rsquared for initial model and final␣
↪reduced model for comparison
print('MSE of original model: ', mse_all)
```

```
print('RSE of original model: ', rse_all)
print('Rsquared of original model: ', mdl_bandwidth_vs_all.rsquared)
print('Rsquared Adjusted of original model: ', mdl_bandwidth_vs_all.
 ↪rsquared_adj)
mse_final = mdl_bandwidth_vs_features_final.mse_resid
print('MSE of final model: ', mse_final)
rse_final = np.sqrt(mse_final)
print('RSE of final model: ', rse_final)
print('Rsquared of final model: ', mdl_bandwidth_vs_features_final.rsquared)
print('Rsquared Adjusted of final model: ', mdl_bandwidth_vs_features_final.
 ↪rsquared_adj)
```

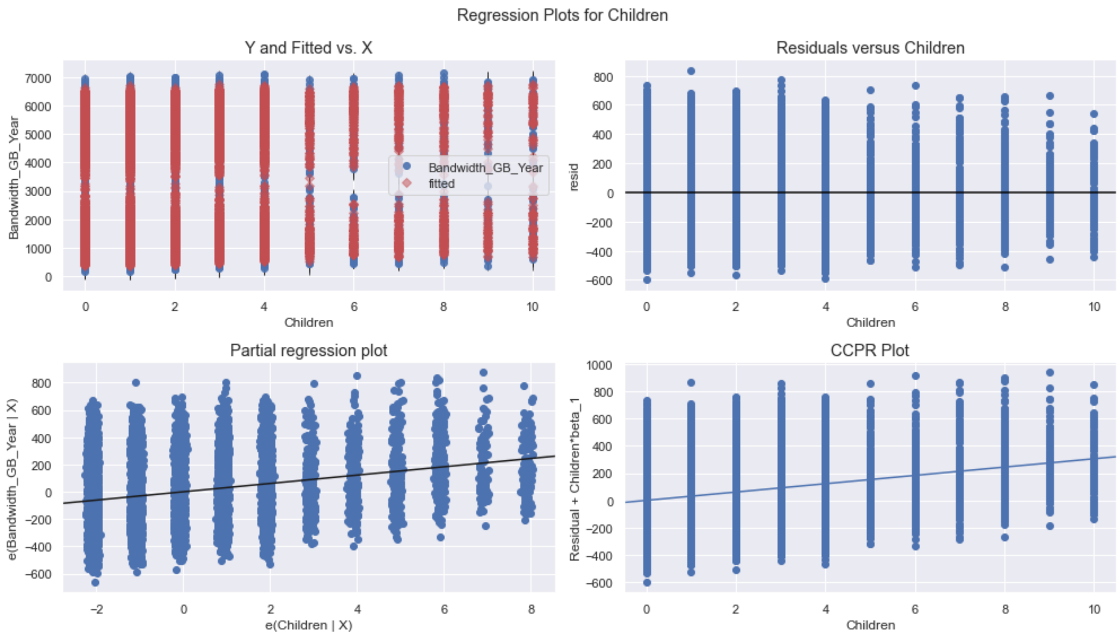
```
MSE of original model:  49276.41635896182
RSE of original model:  221.9829190702785
Rsquared of original model:  0.989692793051259
Rsquared Adjusted of original model:  0.989681441501756
MSE of final model:  60923.13862410184
RSE of final model:  246.82613035110737
Rsquared of final model:  0.9872502548864599
Rsquared Adjusted of final model:  0.9872425996807478
```

4.4 Data Analysis Process

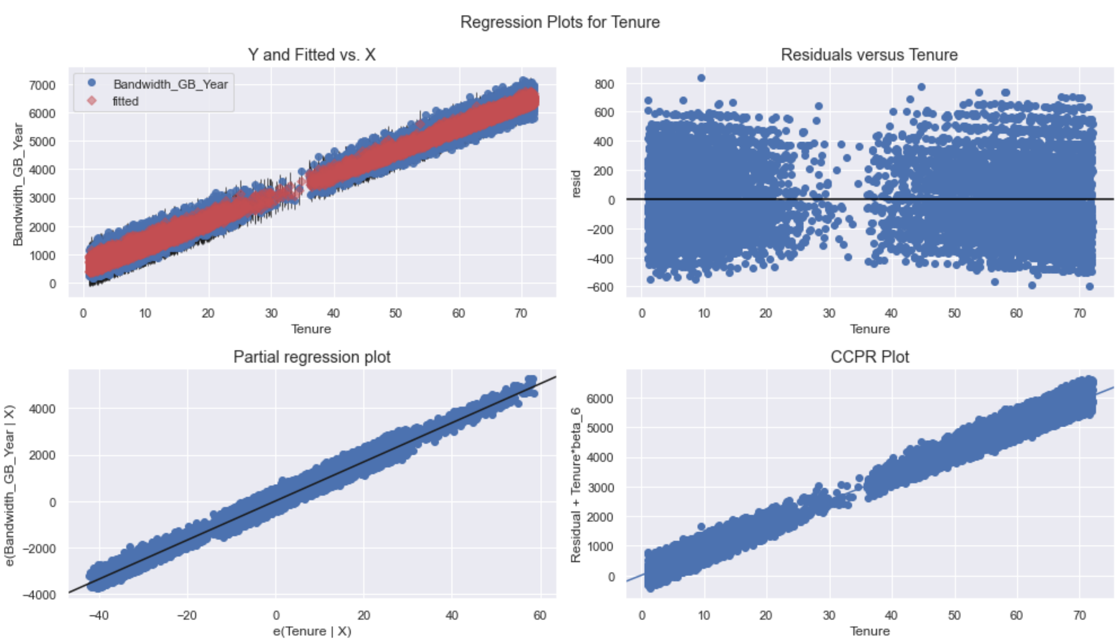
During my variable selection process I relied upon trusted methods for identifying variables unsuitable for the model, such as VIF, a correlation table, and p-values. I measured each model's performance by its r-squared and adjusted r-squared scores, as well as the residual standard error.

Residual plots for each remaining variable, as well as a Q-Q plot for the entire model are shown below.

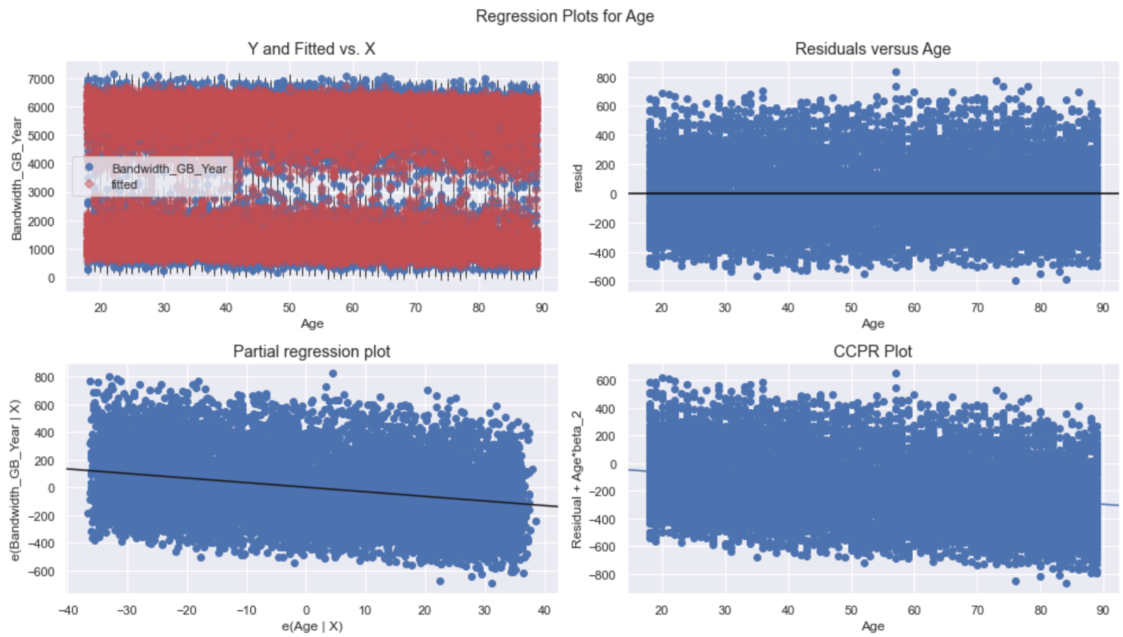
[46]:
```
# Plots for independent variable Children
fig = plt.figure(figsize=(14, 8))
fig = sm.graphics.plot_regress_exog(mdl_bandwidth_vs_features_final,␣
 ↪'Children', fig=fig)
```



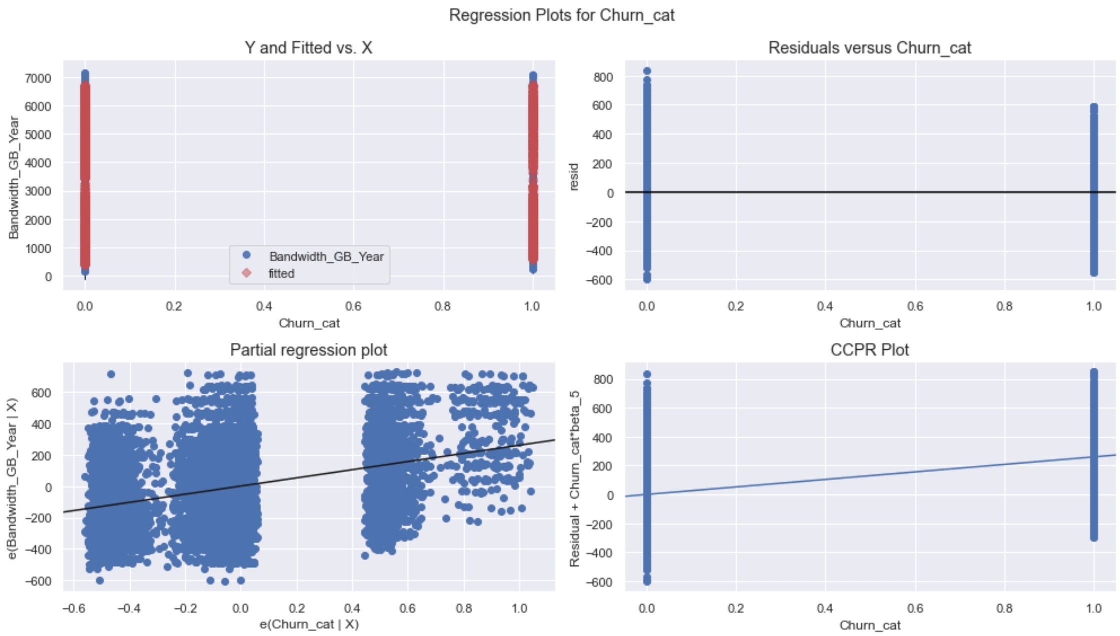
[47]:
```python
# Plots for independent variable Tenure
fig = plt.figure(figsize=(14, 8))
fig = sm.graphics.plot_regress_exog(mdl_bandwidth_vs_features_final, 'Tenure',␣
↪fig=fig)
```



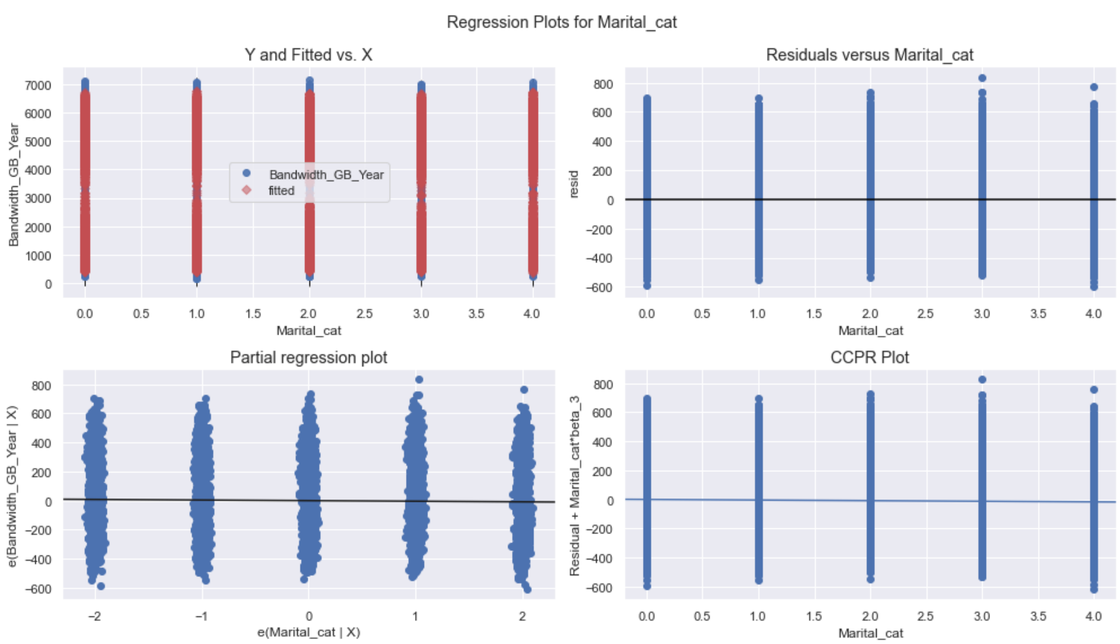
[48]:
```python
# Plots for independent variable Age
fig = plt.figure(figsize=(14, 8))
fig = sm.graphics.plot_regress_exog(mdl_bandwidth_vs_features_final, 'Age',
 fig=fig)
```



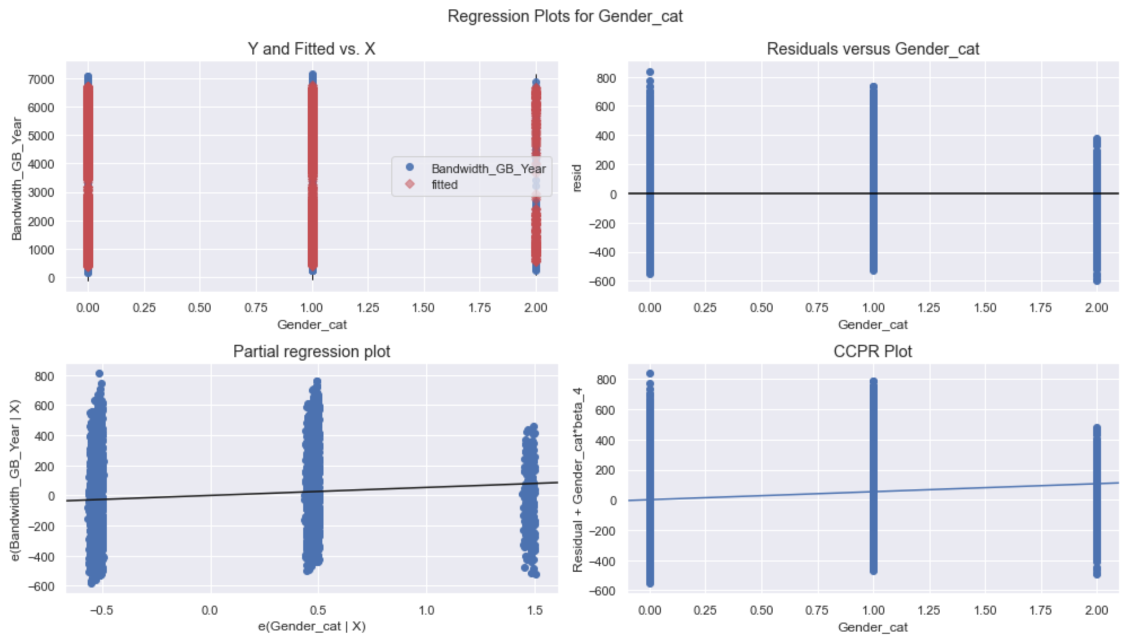
[49]:
```python
# Plots for independent variable Churn_cat
fig = plt.figure(figsize=(14, 8))
fig = sm.graphics.plot_regress_exog(mdl_bandwidth_vs_features_final,
 'Churn_cat', fig=fig)
```



[50]:
```
# Plots for independent variable Marital_cat
fig = plt.figure(figsize=(14, 8))
fig = sm.graphics.plot_regress_exog(mdl_bandwidth_vs_features_final,
 ↪'Marital_cat', fig=fig)
```

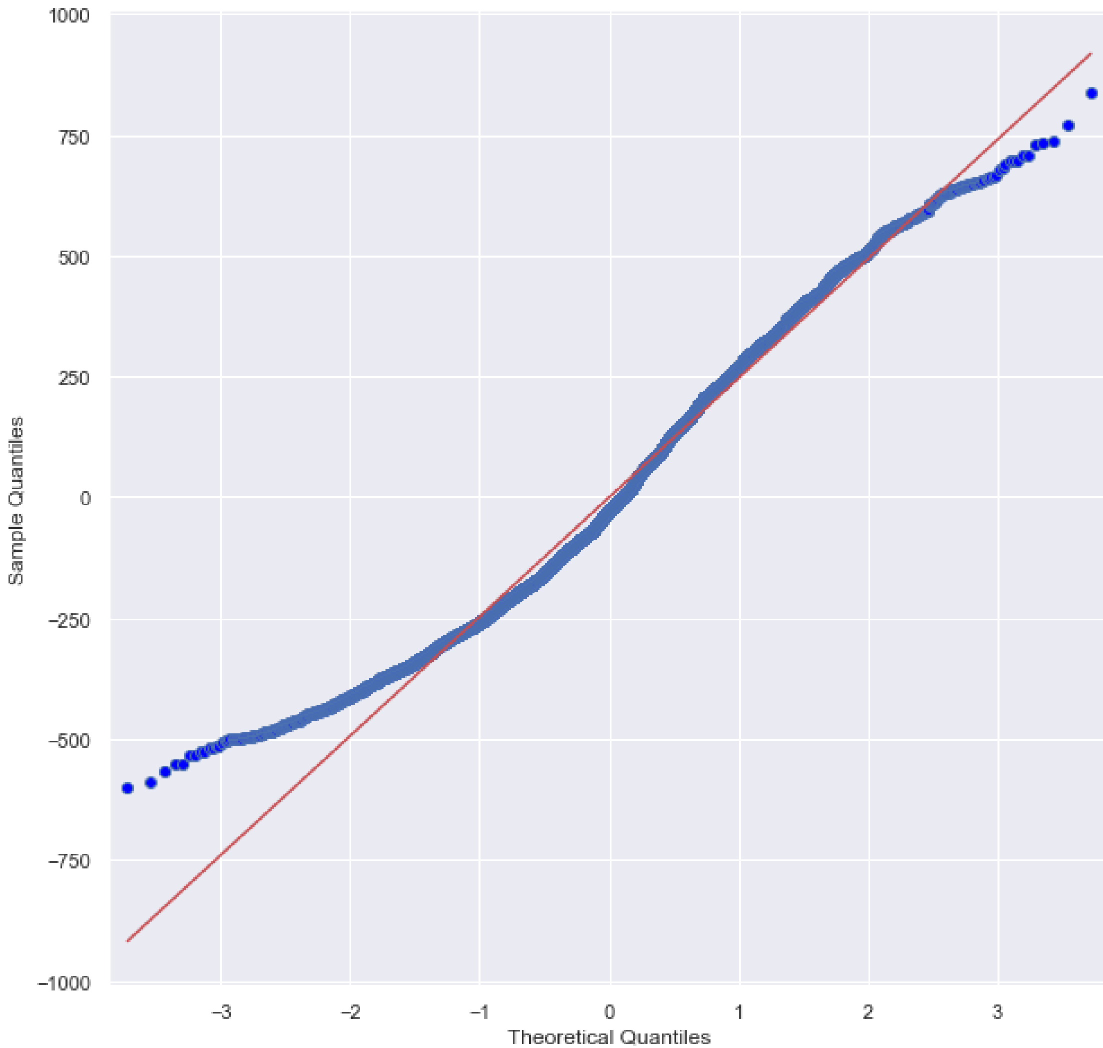


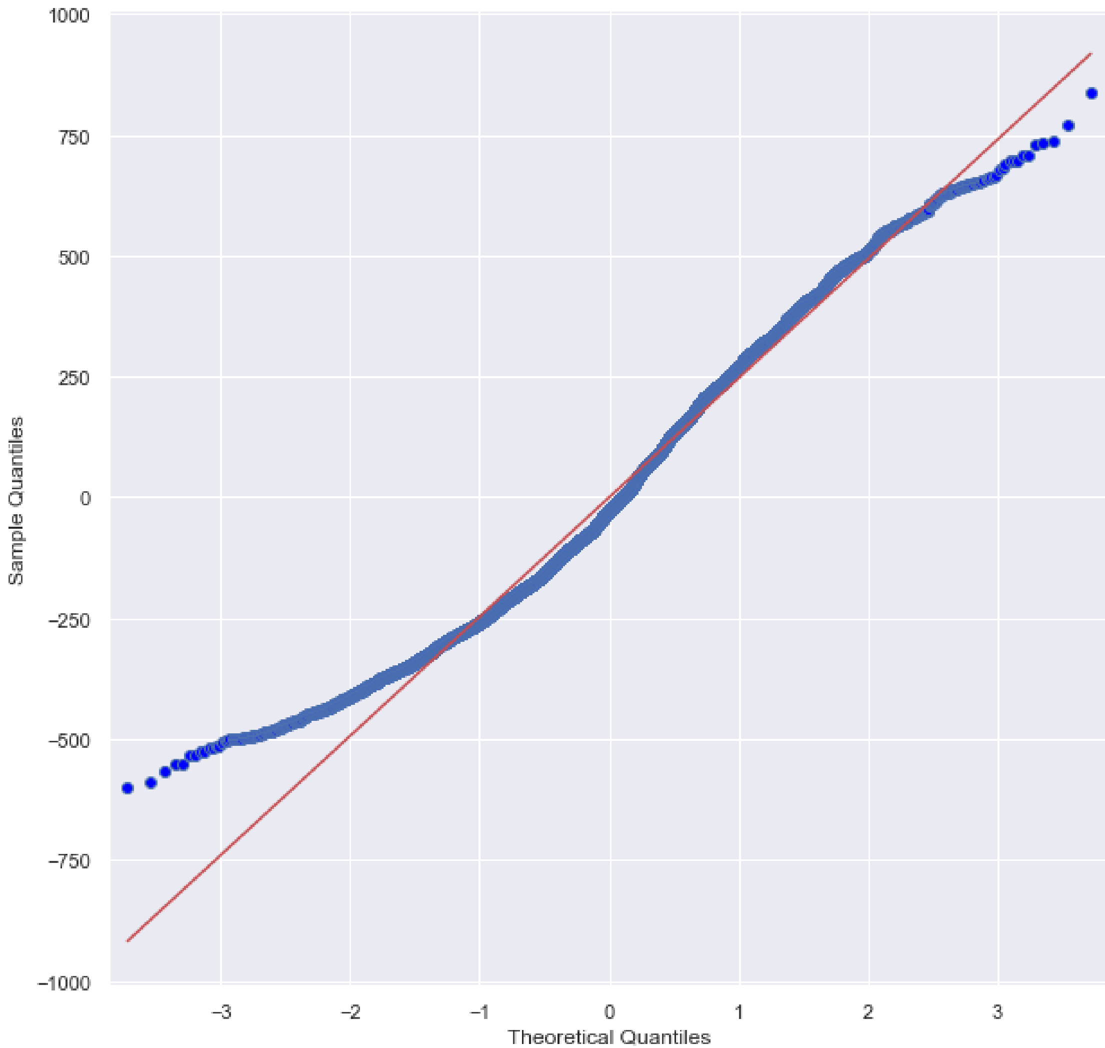
```
[51]: # Plots for independent variable Gender_cat
fig = plt.figure(figsize=(14, 8))
fig = sm.graphics.plot_regress_exog(mdl_bandwidth_vs_features_final,␣
 ↪'Gender_cat', fig=fig)
```




```
[52]: # Q-Q plot for final model
sm.qqplot(mdl_bandwidth_vs_features_final.resid, line='s')
```

[52]:


Sample Quantiles
Theoretical Quantiles
1000
750
500
250
0
−250
−500
−750
−1000
−3
−2
−1
0
1
2
3



---

5 Part V: Data Summary and Implications

5.1 Summary of Findings

The regression equation for the final reduced model is as follows:

Bandwidth_GB_Year ~ Children + Age + Marital_cat + Gender_cat + Churn_cat + Tenure

The coefficients for each variable included:

Children 30.5109

Age -3.3111

Marital_cat -4.0750

Gender_cat 53.1699

Churn_cat 260.0266

Tenure 84.1205

We can use these coefficients to determine the effect each variable will have on the amount of bandwidth use by the customer per year, in GB. For example, for each additional child in the household, bandwidth will increase by 30.5109 GB.

The model can provide significant data when evaluating customer retention from a practical perspective, as customers who use the service more (and use more bandwidth) may be more likely to find the service has value and continue using it. The limitations of using multiple regression models for pratical purposes are always present, however. Data is based on a sample size, and therefore may not reflect general population as accurately as we would like. It also assumes correlation is causation, or that when one thing is true it causes another to be true as well.

5.2 Recommended Course of Action

There are a few key takeaways based on the analysis of this model. Customers who have been with the service provider for a long use more bandwidth and are likely happy with their service. They may have even purchased addons such as streaming video or phone. Based on this, newer customers may be an opportunistic target for special promotions or proactive support initiatives. At the same time, as a customer ages or has a change in marital status, their usage decreases. New products or services that cater to those populations may enhance their experience and generate value for them as users of the service.

---

6 Part VI: Demonstration

Panopto Video Recording

A link for the Panopto video has been provided separately. The demonstration includes the following:

- Demonstration of the functionality of the code used for the analysis
- Identification of the version of the programming environment
- Comparison of the two multiple regression models you used in your analysis
- Interpretation of the coefficients

---

7 Web Sources

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.astype.html

https://www.geeksforgeeks.org/how-to-create-a-residual-plot-in-python/

https://www.sfu.ca/~mjbrydon/tutorials/BAinPy/09_regression.html

https://pbpython.com/categorical-encoding.html

---

8 References

Insights for Professionals. (2019, February 26). *5 Niche Programming Languages (And Why They're Underrated).* https://www.insightsforprofessionals.com/it/software/niche-programming-languages

Parra, H. (2021, April 20). *The Data Science Trilogy.* Towards Data Science. https://towardsdatascience.com/the-data-science-trilogy-numpy-pandas-and-matplotlib-basics-42192b89e26

Zach. (2021, November 16). *The Five Assumptions of Multiple Linear Regression.* Statology. https://www.statology.org/multiple-linear-regression-assumptions/