In [1]:
```
import pandas as pd;
import numpy as np;
import scipy.stats as stats;
import matplotlib.pyplot as plt;
import seaborn as sns;
import sklearn;
from sklearn import linear_model;
from sklearn.metrics import mean_squared_error;
from statsmodels.formula.api import ols;
import statsmodels.api as sm;
from statsmodels.stats.outliers_influence import variance_inflation_factor;
from statsmodels.tools.tools import add_constant;
from platform import python_version
```

In [2]:
```
df = pd.read_csv('churn_clean.csv')
```

In [3]:
```
df["Churn"] = df['Churn'].map({'No':0, 'Yes':1})
df['Churn'].value_counts()
```

Out[3]:
```
0    7350
1    2650
Name: Churn, dtype: int64
```

In [4]:
```
dfmlr = df[['MonthlyCharge', 'Churn', 'Children', 'Income', 'Bandwidth_GB_Year', 'Tenu
```

In [5]:
```
#C2 summary statistics
dfmlr.describe()
```

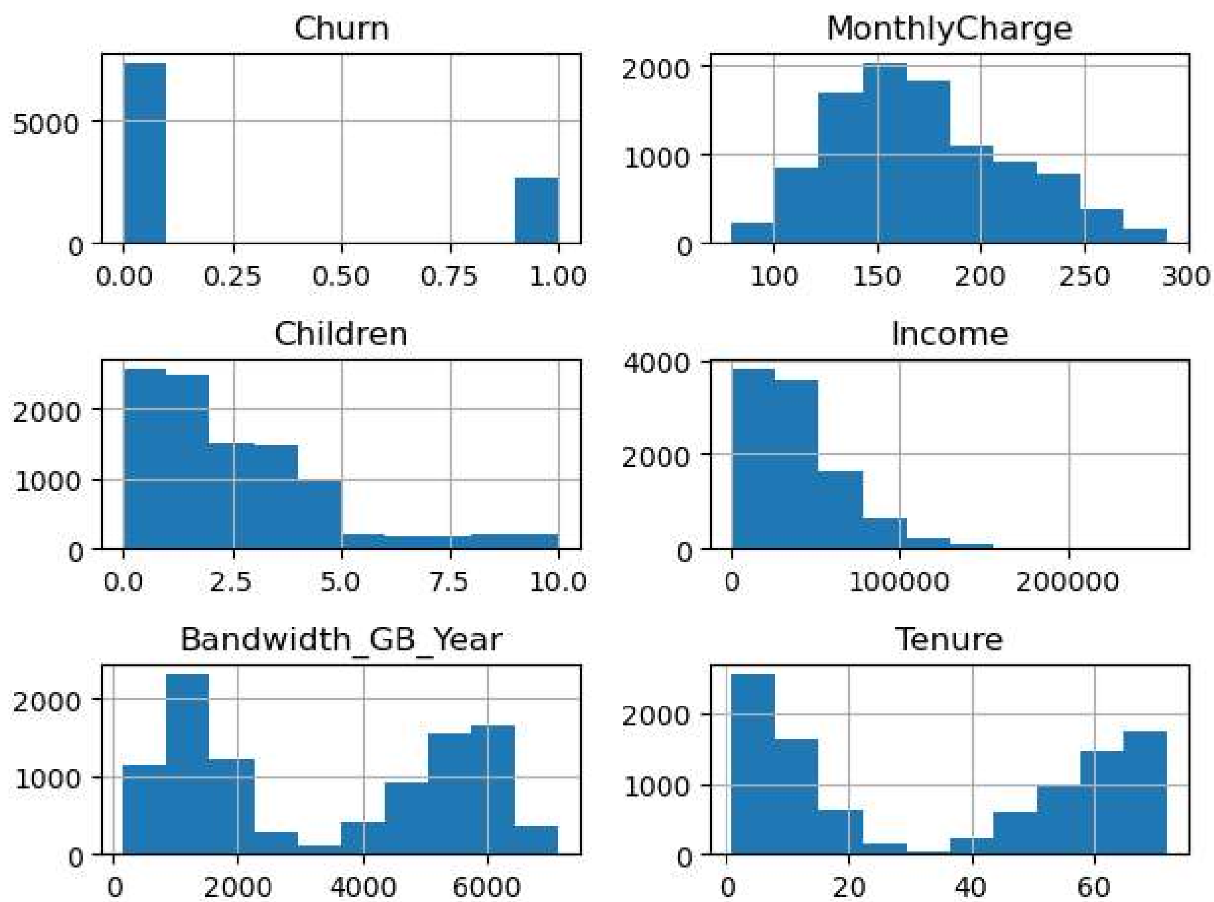
Out[5]:

	MonthlyCharge	Churn	Children	Income	Bandwidth_GB_Year	Tenure
count	10000.000000	10000.000000	10000.0000	10000.000000	10000.000000	10000.000000
mean	172.624816	0.265000	2.0877	39806.926771	3392.341550	34.526188
std	42.943094	0.441355	2.1472	28199.916702	2185.294852	26.443063
min	79.978860	0.000000	0.0000	348.670000	155.506715	1.000259
25%	139.979239	0.000000	0.0000	19224.717500	1236.470827	7.917694
50%	167.484700	0.000000	1.0000	33170.605000	3279.536903	35.430507
75%	200.734725	1.000000	3.0000	53246.170000	5586.141370	61.479795
max	290.160419	1.000000	10.0000	258900.700000	7158.981530	71.999280

In [6]:
```
#get missing values
dfmlr.isna().sum()
```

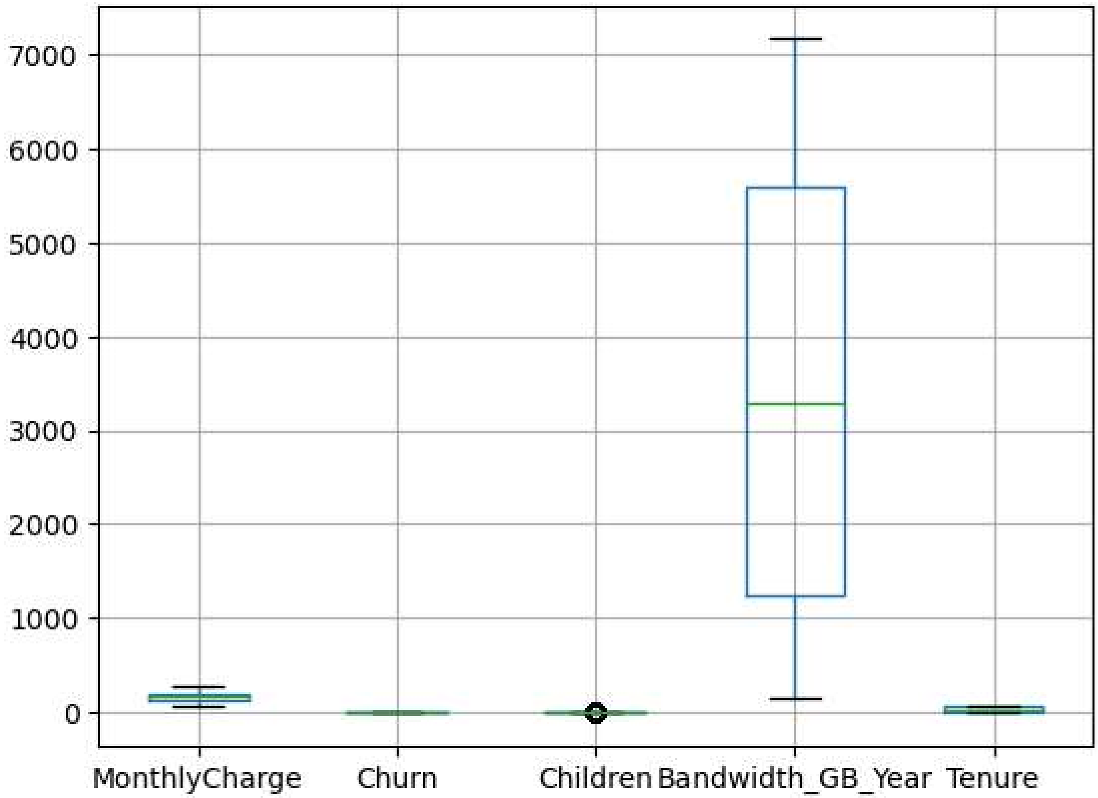
Out[6]:
```
MonthlyCharge        0
Churn                0
Children             0
Income               0
Bandwidth_GB_Year    0
Tenure               0
dtype: int64
```

In [7]:
```
df[['Churn', 'MonthlyCharge', 'Children', 'Income', 'Bandwidth_GB_Year', 'Tenure']].hi
plt.tight_layout()
```



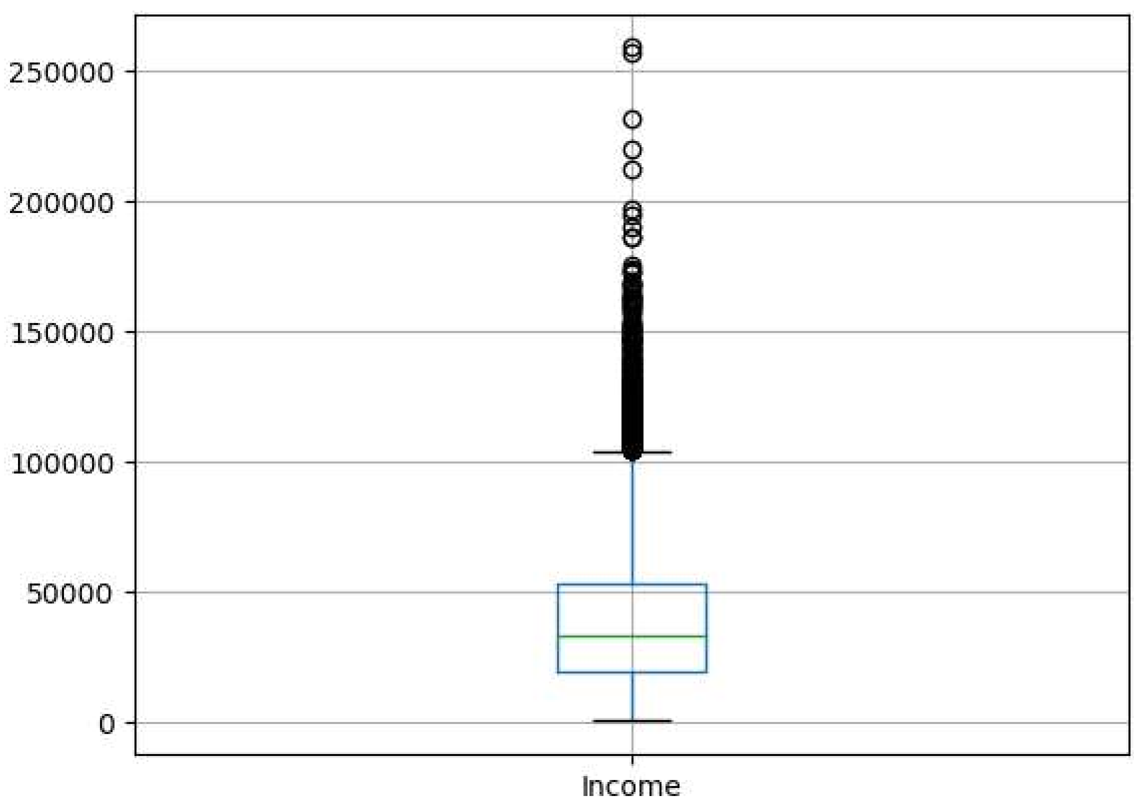
In [8]:
```
dfmlr.boxplot(column=['MonthlyCharge', 'Churn', 'Children', 'Bandwidth_GB_Year', 'Tenu
```

Out[8]:
```
<AxesSubplot:>
```



In [9]:
```
dfmlr.boxplot(column=['Income'])
```

Out[9]: <AxesSubplot:>



In [10]:
```
dfmlr.to_excel('MLR_Churn_clean.xlsx')
dfmlr.head()
```

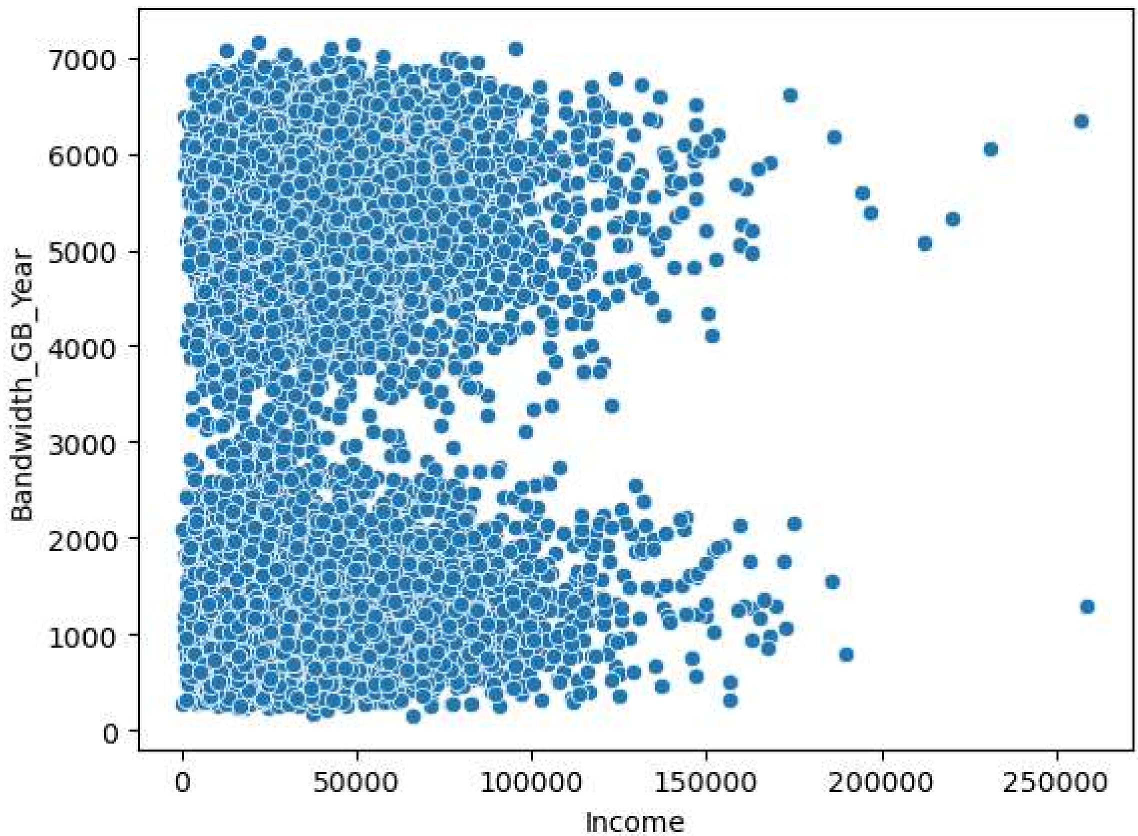
Out[10]:

	MonthlyCharge	Churn	Children	Income	Bandwidth_GB_Year	Tenure
0	172.455519	0	0	28561.99	904.536110	6.795513
1	242.632554	1	1	21704.77	800.982766	1.156681
2	159.947583	0	4	9609.57	2054.706961	15.754144
3	119.956840	0	1	18925.23	2164.579412	17.087227
4	149.948316	1	0	40074.19	271.493436	1.670972

In [11]:

```
sns.scatterplot(x='Income',
                y='Bandwidth_GB_Year', data=dfmlr)
```

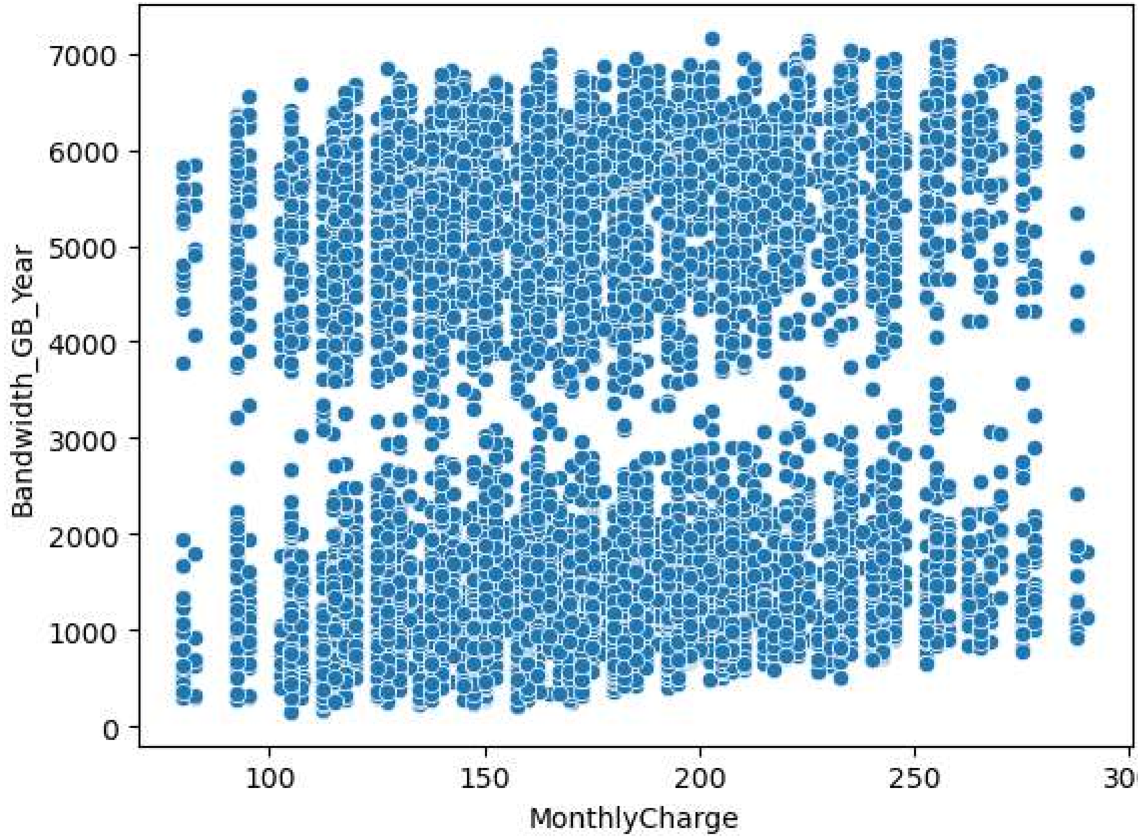
Out[11]: <AxesSubplot:xlabel='Income', ylabel='Bandwidth_GB_Year'>



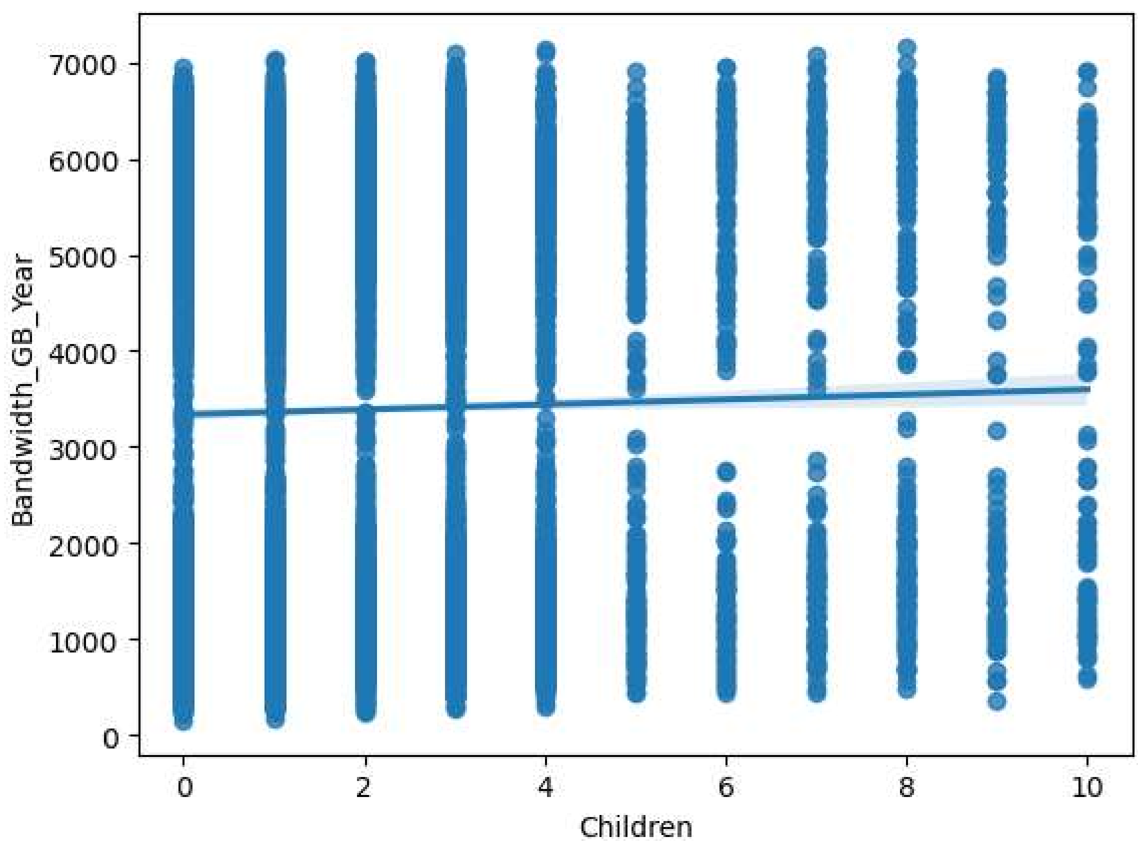
In [12]:

```
sns.scatterplot(x='MonthlyCharge',
                y='Bandwidth_GB_Year', data=dfmlr)
```

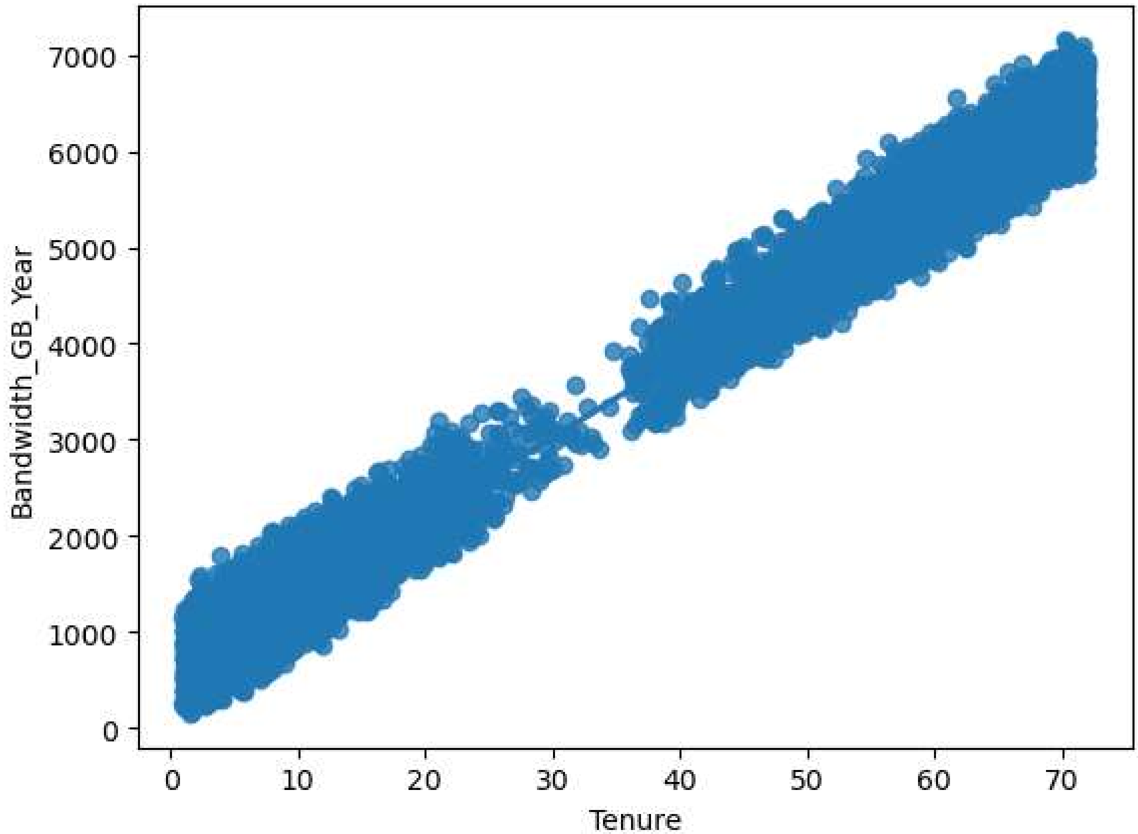
Out[12]: <AxesSubplot:xlabel='MonthlyCharge', ylabel='Bandwidth_GB_Year'>



In [13]:
```
sns.regplot(x="Children",
            y="Bandwidth_GB_Year",
            data=dfmlr);
```

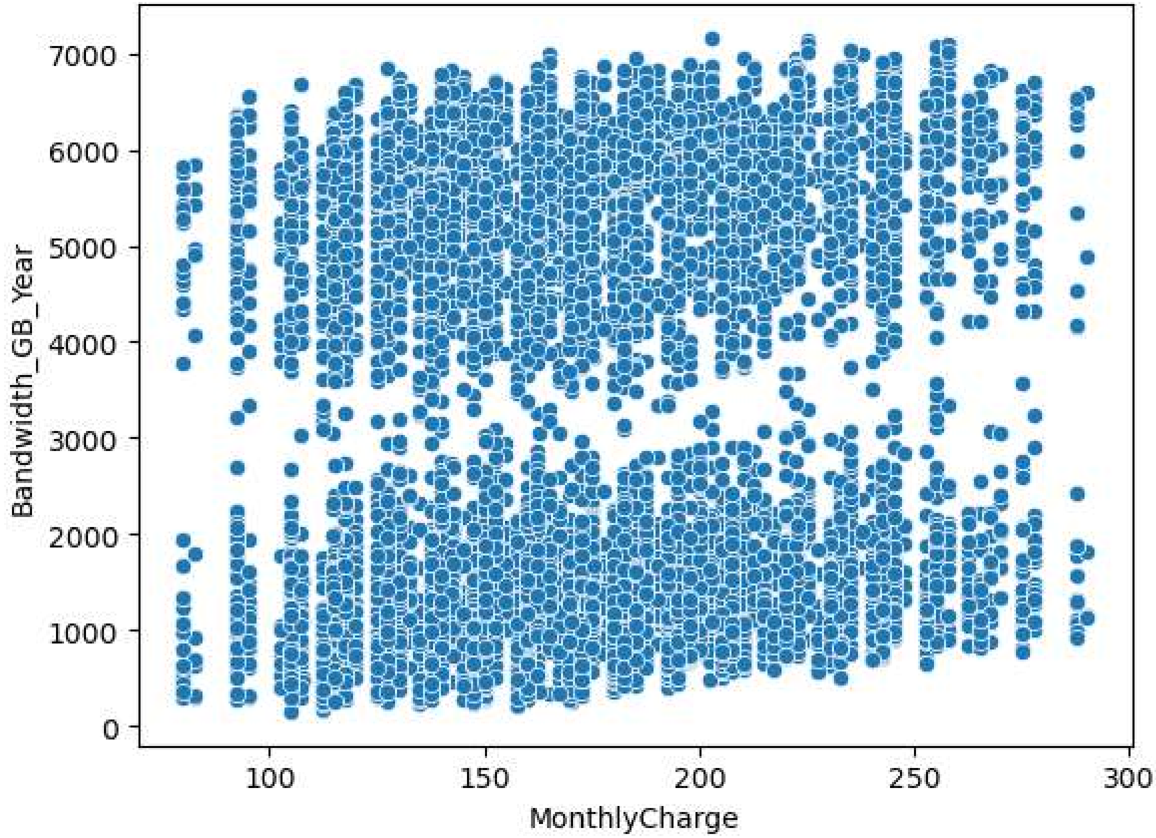


In [14]:
```
sns.regplot(x="Tenure",
            y="Bandwidth_GB_Year",
            data=dfmlr);
```



In [15]:
```
sns.scatterplot(x='MonthlyCharge',
                y='Bandwidth_GB_Year', data=dfmlr)
```

Out[15]: <AxesSubplot:xlabel='MonthlyCharge', ylabel='Bandwidth_GB_Year'>



In [16]:

```
#C1 VIF - Checking for multicollinarity

X = dfmlr[['Churn', 'MonthlyCharge', 'Children', 'Income', 'Tenure']]
vif = pd.DataFrame()
vif['feature'] = X.columns
vif["VIF"] = [variance_inflation_factor(X.values, i)
                          for i in range(len(X.columns))]
print(vif)
```

```
         feature       VIF
0          Churn  2.160821
1  MonthlyCharge  7.223742
2       Children  1.857087
3         Income  2.721804
4         Tenure  3.565599
```

In [17]:

```
#MLR using statsmodels
X = dfmlr[['Churn', 'MonthlyCharge', 'Children', 'Income', 'Tenure']]
y = dfmlr['Bandwidth_GB_Year']
X = sm.add_constant(X)
modl = sm.OLS(y, X).fit()

print(modl.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Bandwidth_GB_Year   R-squared:                       0.989
Model:                            OLS   Adj. R-squared:                  0.989
Method:                 Least Squares   F-statistic:                 1.720e+05
Date:                Thu, 05 Jan 2023   Prob (F-statistic):               0.00
Time:                        12:56:43   Log-Likelihood:                -68750.
No. Observations:               10000   AIC:                         1.375e+05
Df Residuals:                    9994   BIC:                         1.376e+05
Df Model:                           5
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const           -56.0022     10.998     -5.092      0.000     -77.561     -34.444
Churn           127.2987      6.706     18.981      0.000     114.153     140.445
MonthlyCharge     2.7724      0.060     46.007      0.000       2.654       2.890
Children         31.8840      1.091     29.221      0.000      29.745      34.023
Income         9.688e-05   8.31e-05      1.166      0.244    -6.6e-05       0.000
Tenure           82.9982      0.104    799.137      0.000      82.795      83.202
==============================================================================
Omnibus:                     2216.683   Durbin-Watson:                   1.990
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              639.489
Skew:                           0.383   Prob(JB):                    1.37e-139
Kurtosis:                       2.026   Cond. No.                     2.29e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
ified.
[2] The condition number is large, 2.29e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [18]:

```python
#reduced MLR model
X = dfmlr[['MonthlyCharge', 'Children', 'Tenure']]
y = dfmlr['Bandwidth_GB_Year']
X = sm.add_constant(X)
modl = sm.OLS(y, X).fit()

print(modl.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Bandwidth_GB_Year   R-squared:                       0.988
Model:                            OLS   Adj. R-squared:                  0.988
Method:                 Least Squares   F-statistic:                 2.767e+05
Date:                Thu, 05 Jan 2023   Prob (F-statistic):               0.00
Time:                        12:56:43   Log-Likelihood:                -68928.
No. Observations:               10000   AIC:                         1.379e+05
Df Residuals:                    9996   BIC:                         1.379e+05
Df Model:                           3
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const           -66.5773     10.646     -6.254      0.000     -87.446     -45.708
MonthlyCharge     3.2580      0.056     58.673      0.000       3.149       3.367
Children         31.8156      1.111     28.649      0.000      29.639      33.992
Tenure           81.9695      0.090    909.025      0.000      81.793      82.146
==============================================================================
Omnibus:                     3510.726   Durbin-Watson:                   1.988
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              723.629
Skew:                           0.389   Prob(JB):                    7.35e-158
Kurtosis:                       1.936   Cond. No.                         809.
==============================================================================

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

In [19]:

```python
residuals = modl.resid
print(residuals)
```

```
0      -147.763598
1       -49.554067
2       181.560974
3       407.896116
4      -287.422847
           ...
9995    371.094440
9996    -44.144476
9997   -246.438124
9998   -147.488322
9999     -9.042070
Length: 10000, dtype: float64
```

In [20]:

```python
y_pred=modl.predict(X)
y_pred
```

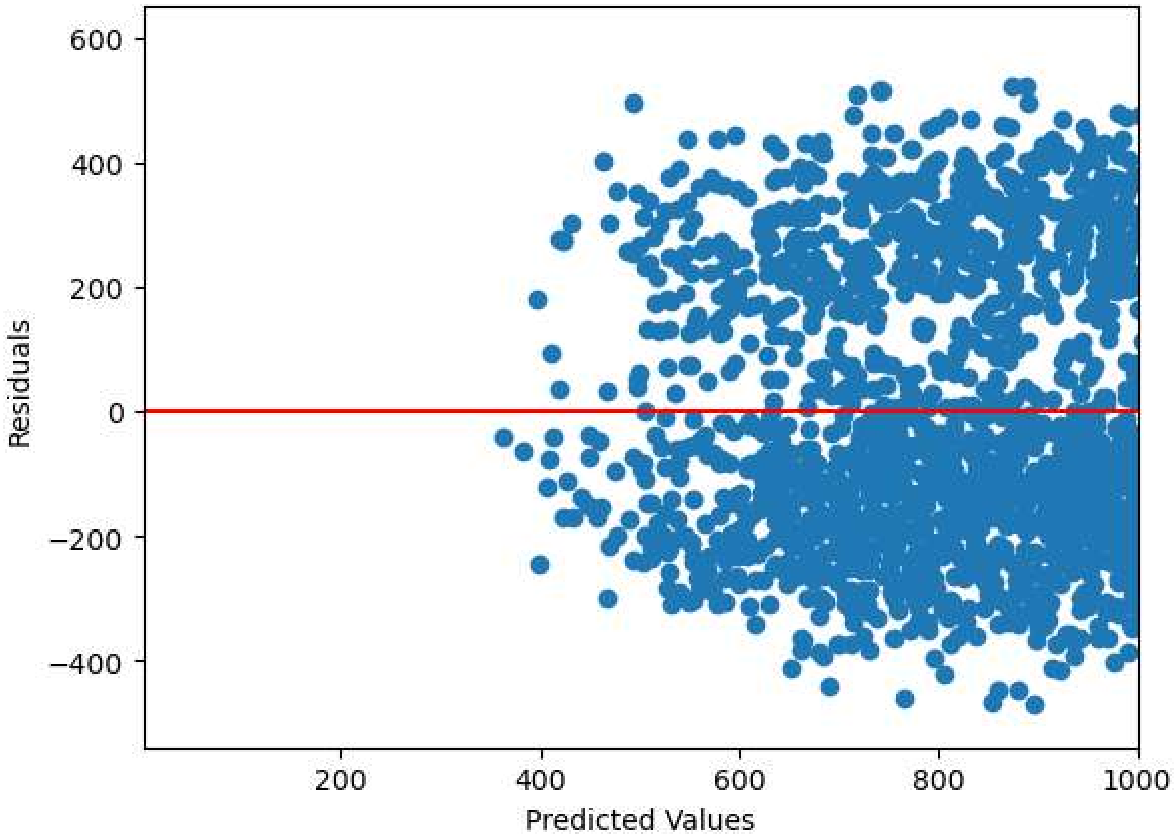
Out[20]:

```
0       1052.299708
1        850.536833
2       1873.145987
3       1756.683296
4        558.916283
           ...
9995    6140.158161
9996    5740.096286
9997    4405.743923
9998    6615.945074
9999    5866.628237
Length: 10000, dtype: float64
```

In [21]:
```
mean_squared_error(y,y_pred)
```

Out[21]:
```
56825.44473679924
```

In [22]:
```
plt.scatter(modl.predict(), residuals);
plt.axhline(0, color='red')
plt.xlabel('Predicted Values');
plt.ylabel('Residuals');
plt.xlim([1,1000]);
```



In [23]:
```
python_version()
```

Out[23]:
```
'3.9.13'
```

In [24]:
```
!jupyter --version
```

```
Selected Jupyter core packages...
IPython          : 7.31.1
ipykernel        : 6.15.2
ipywidgets       : 7.6.5
jupyter_client   : 7.3.4
jupyter_core     : 4.11.1
jupyter_server   : 1.18.1
jupyterlab       : 3.4.4
nbclient         : 0.5.13
nbconvert        : 6.4.4
nbformat         : 5.5.0
notebook         : 6.4.12
qtconsole        : 5.2.2
traitlets        : 5.1.1
```