In [1]:
```
# Import packages
import numpy as np
import pandas as pd
from pandas import Series, DataFrame
```

In [2]:
```
#Visual imports
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:
```
#Stat packages import
import pylab
from pylab import rcParams
import statsmodels.api as sm
import statistics
from scipy import stats
```

In [4]:
```
#skikit
import sklearn
from sklearn import preprocessing
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report
```

In [5]:
```
from scipy.stats import chisquare
from scipy.stats import chi2_contingency
```

In [6]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [7]:
```
#Load dataset
churn_df = pd.read_csv('churn_clean.csv')
```

In [8]:
```
#rename survey items
churn_df.rename(columns = {'Item1' : 'TimelyResponse',
                           'Item2' : 'Fixes',
                           'Item3' : 'Replacement',
                           'Item4' : 'Reliability',
                           'Item5' : 'Options',
                           'Item6' : 'Respectful',
                           'Item7' : 'Courteous',
                           'Item8' : 'Listening'},
                inplace = True)
```

In [9]:
```
churn_df
```

Out[9]:

	CaseOrder	Customer_id	Interaction	UID	City	S
0	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	e885b299883d4f9fb18e39c75155d990	Point Baker	
1	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	f2de8bef964785f41a2959829830fb8a	West Branch	
2	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	f1784cfa9f6d92ae816197eb175d3c71	Yamhill	
3	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	dc8a365077241bb5cd5ccd305136b05e	Del Mar	
4	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	aabb64a116e83fdc4befc1fbab1663f9	Needville	
...	...	...	...	...	...	
9995	9996	M324793	45deb5a2-ae04-4518-bf0b-c82db8dbe4a4	9499fb4de537af195d16d046b79fd20a	Mount Holly	
9996	9997	D861732	6e96b921-0c09-4993-bbda-a1ac6411061a	c09a841117fa81b5c8e19afec2760104	Clarksville	
9997	9998	I243405	e8307ddf-9a01-4fff-bc59-4742e03fd24f	9c41f212d1e04dca84445019bbc9b41c	Mobeetie	
9998	9999	I641617	3775ccfc-0052-4107-81ae-9657f81ecdf3	3e1f269b40c235a1038863ecf6b7a0df	Carrollton	
9999	10000	T38070	9de5fb6e-bd33-4995-aec8-f01d0172a499	0ea683a03a3cd544aefe8388aab16176	Clarkesville	

10000 rows × 50 columns

In [10]:
```
#DF columns
df = churn_df.columns
print(df)
```

```
Index(['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State',
       'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Jo
b',
       'Children', 'Age', 'Income', 'Marital', 'Gender', 'Churn',
       'Outage_sec_perweek', 'Email', 'Contacts', 'Yearly_equip_failure',
       'Techie', 'Contract', 'Port_modem', 'Tablet', 'InternetService',
       'Phone', 'Multiple', 'OnlineSecurity', 'OnlineBackup',
       'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
       'PaperlessBilling', 'PaymentMethod', 'Tenure', 'MonthlyCharge',
       'Bandwidth_GB_Year', 'TimelyResponse', 'Fixes', 'Replacement',
       'Reliability', 'Options', 'Respectful', 'Courteous', 'Listening'],
      dtype='object')
```

In [11]:
```
churn_df.shape
```

Out[11]:
```
(10000, 50)
```

In [12]:
```
churn_df.describe()
```

Out[12]:

	CaseOrder	Zip	Lat	Lng	Population	Children	
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.0000	10000.
mean	5000.50000	49153.319600	38.757567	-90.782536	9756.562400	2.0877	53.
std	2886.89568	27532.196108	5.437389	15.156142	14432.698671	2.1472	20.
min	1.00000	601.000000	17.966120	-171.688150	0.000000	0.0000	18.
25%	2500.75000	26292.500000	35.341828	-97.082813	738.000000	0.0000	35.
50%	5000.50000	48869.500000	39.395800	-87.918800	2910.500000	1.0000	53.
75%	7500.25000	71866.500000	42.106908	-80.088745	13168.000000	3.0000	71.
max	10000.00000	99929.000000	70.640660	-65.667850	111850.000000	10.0000	89.

8 rows × 23 columns

In [13]:
```
#Drop demographics columns
churn_df = churn_df.drop(columns = ['CaseOrder',
                                    'Customer_id',
                                    'Interaction',
                                    'UID',
                                    'City',
                                    'State',
                                    'County',
                                    'Zip',
                                    'Lat',
                                     'Lng',
                                     'Population',
                                     'Area',
                                     'TimeZone',
                                     'Job',
                                    'Marital',
                                    'PaymentMethod'])
churn_df.describe()
```

Out[13]:

	Children	Age	Income	Outage_sec_perweek	Email	Contact
count	10000.0000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000
mean	2.0877	53.078400	39806.926771	10.001848	12.016000	0.99420
std	2.1472	20.698882	28199.916702	2.976019	3.025898	0.98846
min	0.0000	18.000000	348.670000	0.099747	1.000000	0.00000
25%	0.0000	35.000000	19224.717500	8.018214	10.000000	0.00000
50%	1.0000	53.000000	33170.605000	10.018560	12.000000	1.00000
75%	3.0000	71.000000	53246.170000	11.969485	14.000000	2.00000
max	10.0000	89.000000	258900.700000	21.207230	23.000000	7.00000

In [14]:
```
#look for missing data
data_null = churn_df.isnull().sum()
print(data_null)
```

```
Children                0
Age                     0
Income                  0
Gender                  0
Churn                   0
Outage_sec_perweek      0
Email                   0
Contacts                0
Yearly_equip_failure    0
Techie                  0
Contract                0
Port_modem              0
Tablet                  0
InternetService         0
Phone                   0
Multiple                0
OnlineSecurity          0
OnlineBackup            0
DeviceProtection        0
TechSupport             0
StreamingTV             0
StreamingMovies         0
PaperlessBilling        0
Tenure                  0
MonthlyCharge           0
Bandwidth_GB_Year       0
TimelyResponse          0
Fixes                   0
Replacement             0
Reliability             0
Options                 0
Respectful              0
Courteous               0
Listening               0
dtype: int64
```

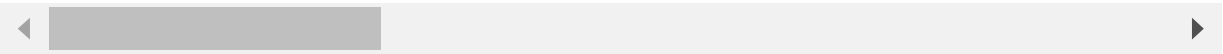
In [15]:
```
#Add dummy variables
churn_df['DummyGender'] = [1 if v == 'Male' else 0 for v in churn_df['Gender'
]]
churn_df['DummyChurn'] = [1 if v == 'Yes' else 0 for v in churn_df['Churn']]
churn_df['DummyTechie'] = [1 if v == 'Yes' else 0 for v in churn_df['Techie']]
churn_df['DummyContract'] = [1 if v == 'Two Year' else 0 for v in churn_df['Co
ntract']]
churn_df['DummyPort_modem'] = [1 if v == 'Yes' else 0 for v in churn_df['Port_
modem']]
churn_df['DummyTablet'] = [1 if v == 'Yes' else 0 for v in churn_df['Tablet']]
churn_df['DummyInternetService'] = [1 if v == 'Fiber Optic' else 0 for v in ch
urn_df['InternetService']]
churn_df['DummyPhone'] = [1 if v == 'Yes' else 0 for v in churn_df['Phone']]
churn_df['DummyMultiple'] = [1 if v == 'Yes' else 0 for v in churn_df['Multipl
e']]
churn_df['DummyOnlineSecurity'] = [1 if v == 'Yes' else 0 for v in churn_df['O
nlineSecurity']]
churn_df['DummyOnlineBackup'] = [1 if v == 'Yes' else 0 for v in churn_df['Onl
ineBackup']]
churn_df['DummyDeviceProtection'] = [1 if v == 'Yes' else 0 for v in churn_df[
'DeviceProtection']]
churn_df['DummyTechSupport'] = [1 if v == 'Yes' else 0 for v in churn_df['Tech
Support']]
churn_df['DummyStreamingTV'] = [1 if v == 'Yes' else 0 for v in churn_df['Stre
amingTV']]
churn_df['DummyStreamingMovies'] = [1 if v == 'Yes' else 0 for v in churn_df[
'StreamingMovies']]
churn_df['DummyPaperlessBilling'] = [1 if v == 'Yes' else 0 for v in churn_df[
'PaperlessBilling']]
churn_df['Churn'] = [1 if v == 'Yes' else 0 for v in churn_df['Churn']]
```

In [16]:
```
churn_df.head()
```

Out[16]:

	Children	Age	Income	Gender	Churn	Outage_sec_perweek	Email	Contacts	Yearly_equip
0	0	68	28561.99	Male	0	7.978323	10	0	
1	1	27	21704.77	Female	1	11.699080	12	0	
2	4	50	9609.57	Female	0	10.752800	9	0	
3	1	48	18925.23	Male	0	14.913540	15	2	
4	0	83	40074.19	Male	1	8.147417	16	2	

5 rows × 50 columns



In [17]:
```
churn_df = churn_df.drop(columns=['Gender',
                                  'Gender',
                                  'Techie',
                                  'Contract',
                                  'Port_modem',
                                  'Tablet',
                                  'InternetService',
                                  'Phone',
                                  'Multiple',
                                  'OnlineSecurity',
                                  'OnlineBackup',
                                  'DeviceProtection',
                                  'TechSupport',
                                  'StreamingTV',
                                  'StreamingMovies',
                                  'PaperlessBilling'])
churn_df.describe()
```

Out[17]:

	Children	Age	Income	Churn	Outage_sec_perweek	Ema
count	10000.0000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000
mean	2.0877	53.078400	39806.926771	0.265000	10.001848	12.01600
std	2.1472	20.698882	28199.916702	0.441355	2.976019	3.02589
min	0.0000	18.000000	348.670000	0.000000	0.099747	1.00000
25%	0.0000	35.000000	19224.717500	0.000000	8.018214	10.00000
50%	1.0000	53.000000	33170.605000	0.000000	10.018560	12.00000
75%	3.0000	71.000000	53246.170000	1.000000	11.969485	14.00000
max	10.0000	89.000000	258900.700000	1.000000	21.207230	23.00000

8 rows × 35 columns

In [18]:
```
df = churn_df.columns
print(df)
```

```
Index(['Children', 'Age', 'Income', 'Churn', 'Outage_sec_perweek', 'Email',
       'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',
       'Bandwidth_GB_Year', 'TimelyResponse', 'Fixes', 'Replacement',
       'Reliability', 'Options', 'Respectful', 'Courteous', 'Listening',
       'DummyGender', 'DummyChurn', 'DummyTechie', 'DummyContract',
       'DummyPort_modem', 'DummyTablet', 'DummyInternetService', 'DummyPhon
e',
       'DummyMultiple', 'DummyOnlineSecurity', 'DummyOnlineBackup',
       'DummyDeviceProtection', 'DummyTechSupport', 'DummyStreamingTV',
       'DummyStreamingMovies', 'DummyPaperlessBilling'],
      dtype='object')
```

In [19]:
```
#move dummy churn to the end
churn_df = churn_df[['Children', 'Age', 'Income', 'Churn', 'Outage_sec_perwee
k', 'Email',
       'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',
       'Bandwidth_GB_Year', 'TimelyResponse', 'Fixes', 'Replacement',
       'Reliability', 'Options', 'Respectful', 'Courteous', 'Listening',
       'DummyGender', 'DummyTechie', 'DummyContract',
       'DummyPort_modem', 'DummyTablet', 'DummyInternetService', 'DummyPhone',
       'DummyMultiple', 'DummyOnlineSecurity', 'DummyOnlineBackup',
       'DummyDeviceProtection', 'DummyTechSupport', 'DummyStreamingTV',
       'DummyStreamingMovies', 'DummyPaperlessBilling', 'DummyChurn']]
```

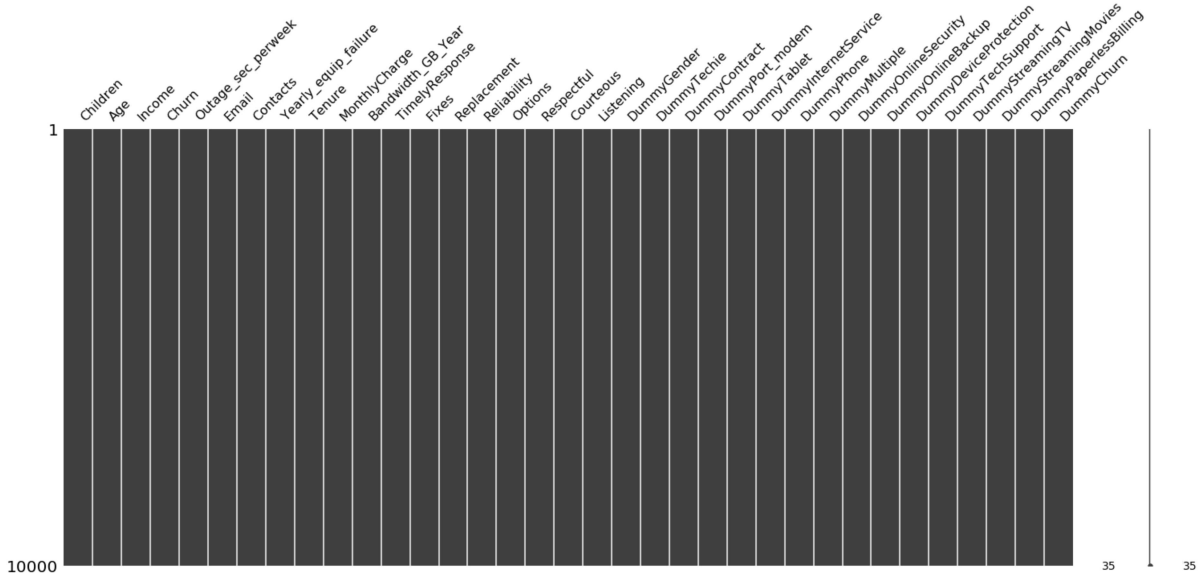
In [20]:
```
df = churn_df.columns
print(df)
```

```
Index(['Children', 'Age', 'Income', 'Churn', 'Outage_sec_perweek', 'Email',
       'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',
       'Bandwidth_GB_Year', 'TimelyResponse', 'Fixes', 'Replacement',
       'Reliability', 'Options', 'Respectful', 'Courteous', 'Listening',
       'DummyGender', 'DummyTechie', 'DummyContract', 'DummyPort_modem',
       'DummyTablet', 'DummyInternetService', 'DummyPhone', 'DummyMultiple',
       'DummyOnlineSecurity', 'DummyOnlineBackup', 'DummyDeviceProtection',
       'DummyTechSupport', 'DummyStreamingTV', 'DummyStreamingMovies',
       'DummyPaperlessBilling', 'DummyChurn'],
      dtype='object')
```

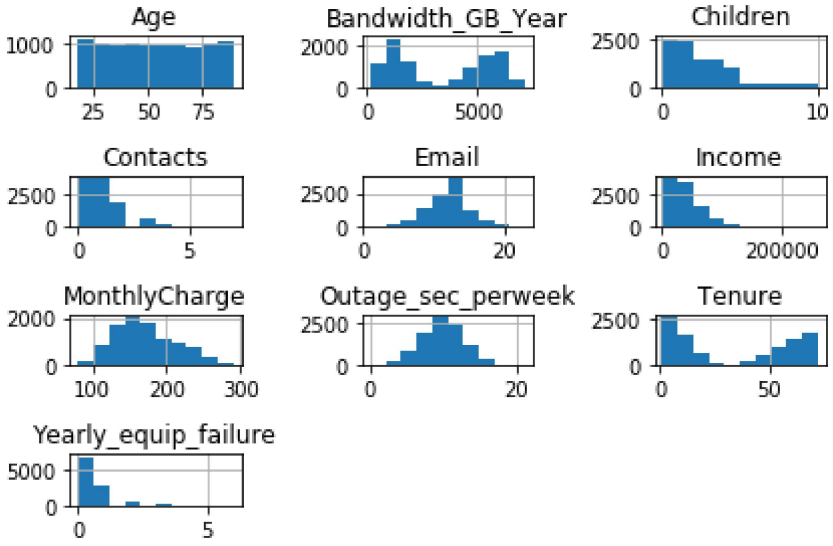
In [21]:
```
#Visualization imports
!pip install missingno
import missingno as msno
msno.matrix(churn_df)
```

```
Requirement already satisfied: missingno in c:\users\coope\anaconda3\lib\site
-packages (0.5.0)
Requirement already satisfied: scipy in c:\users\coope\anaconda3\lib\site-pac
kages (from missingno) (1.3.1)
Requirement already satisfied: numpy in c:\users\coope\anaconda3\lib\site-pac
kages (from missingno) (1.16.5)
Requirement already satisfied: seaborn in c:\users\coope\anaconda3\lib\site-p
ackages (from missingno) (0.9.0)
Requirement already satisfied: matplotlib in c:\users\coope\anaconda3\lib\sit
e-packages (from missingno) (3.1.1)
Requirement already satisfied: pandas>=0.15.2 in c:\users\coope\anaconda3\lib
\site-packages (from seaborn->missingno) (0.25.1)
Requirement already satisfied: cycler>=0.10 in c:\users\coope\anaconda3\lib\s
ite-packages (from matplotlib->missingno) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\coope\anaconda3
\lib\site-packages (from matplotlib->missingno) (1.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
c:\users\coope\anaconda3\lib\site-packages (from matplotlib->missingno) (2.4.
2)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\coope\anacond
a3\lib\site-packages (from matplotlib->missingno) (2.8.0)
Requirement already satisfied: pytz>=2017.2 in c:\users\coope\anaconda3\lib\s
ite-packages (from pandas>=0.15.2->seaborn->missingno) (2019.3)
Requirement already satisfied: six in c:\users\coope\anaconda3\lib\site-packa
ges (from cycler>=0.10->matplotlib->missingno) (1.12.0)
Requirement already satisfied: setuptools in c:\users\coope\anaconda3\lib\sit
e-packages (from kiwisolver>=1.0.1->matplotlib->missingno) (41.4.0)
```

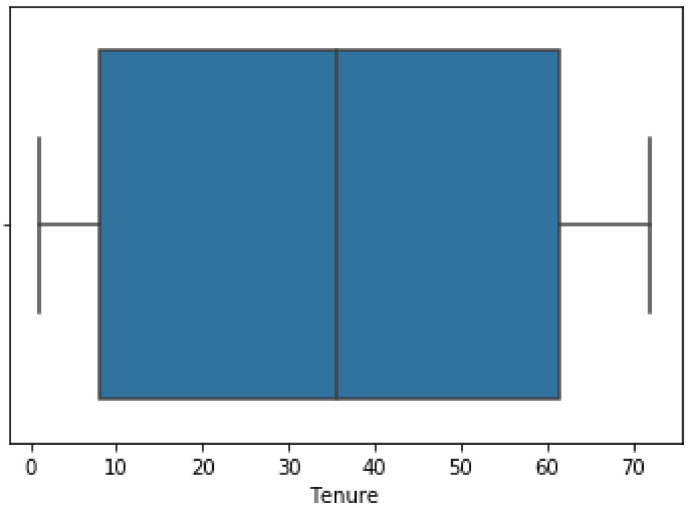
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x1f5dfe82488>



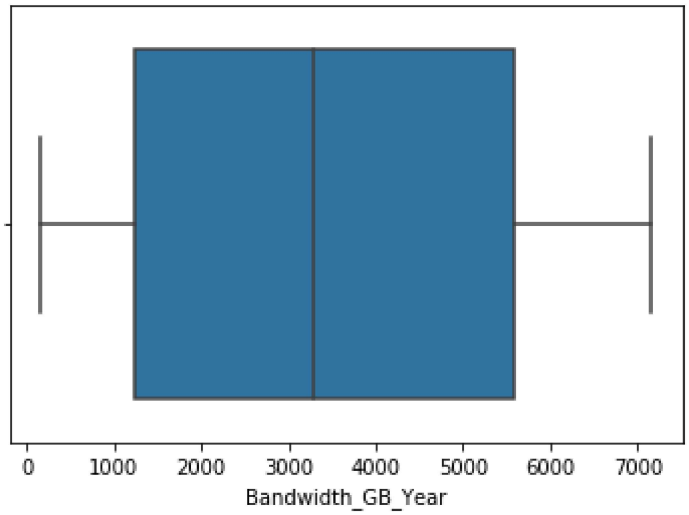
In [22]:
```
#Create histograms
churn_df[['Children',
          'Age',
          'Income',
          'Outage_sec_perweek',
          'Email',
         'Contacts',
         'Yearly_equip_failure',
         'Tenure',
         'MonthlyCharge',
         'Bandwidth_GB_Year']].hist()
plt.savefig('churn_pyplot.jpg')
plt.tight_layout()
```



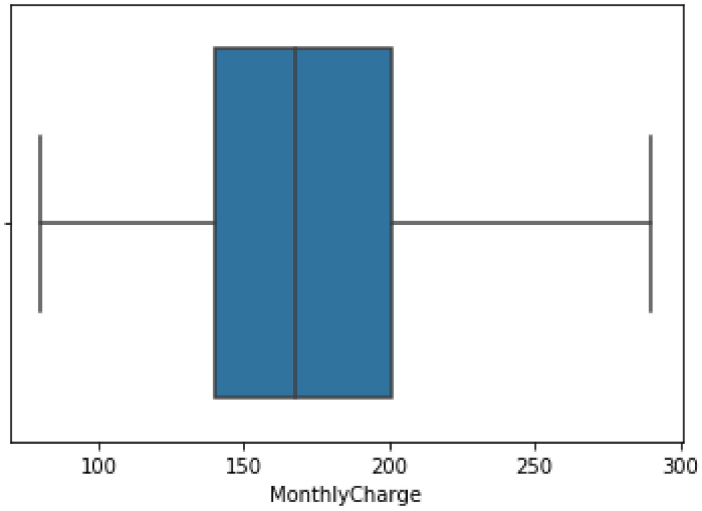
In [23]:
```
#Create seaborn boxplots
sns.boxplot('Tenure', data = churn_df)
plt.show()
```



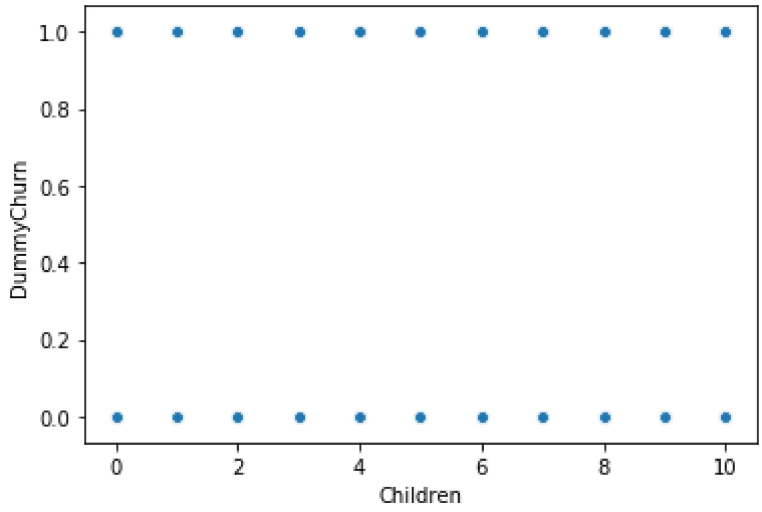
In [24]:
```
sns.boxplot('Bandwidth_GB_Year', data = churn_df)
plt.show()
```



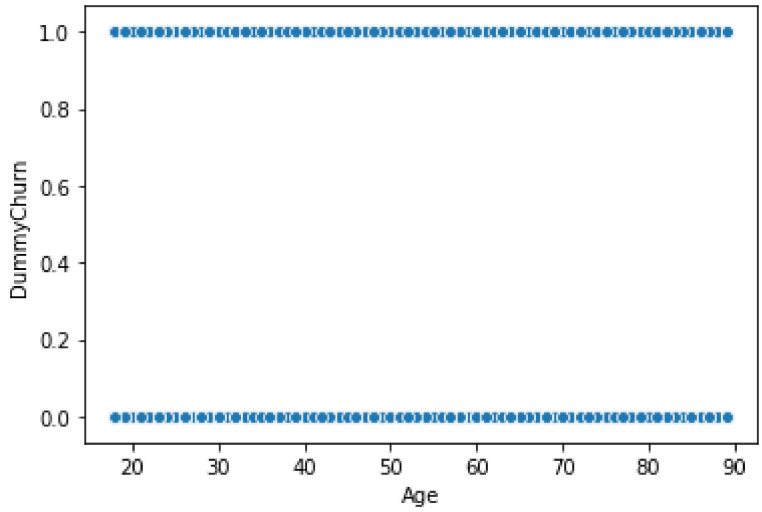
In [25]:
```
sns.boxplot('MonthlyCharge', data = churn_df)
plt.show()
```



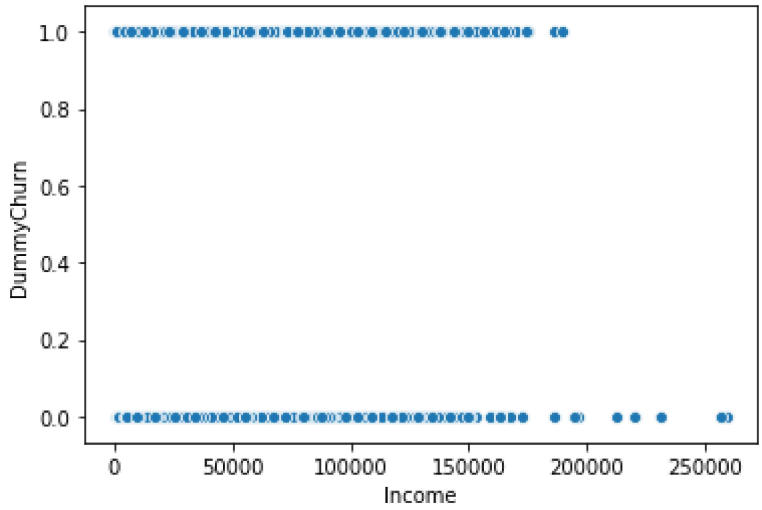
In [26]:
```
#Scatterplot creation
sns.scatterplot(x=churn_df['Children'], y=churn_df['DummyChurn'])
plt.show()
```



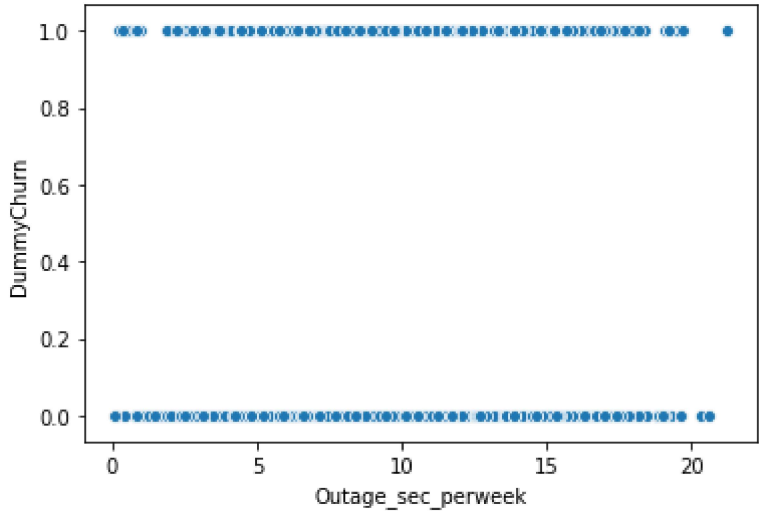
In [27]:
```
sns.scatterplot(x=churn_df['Age'], y=churn_df['DummyChurn'])
plt.show()
```



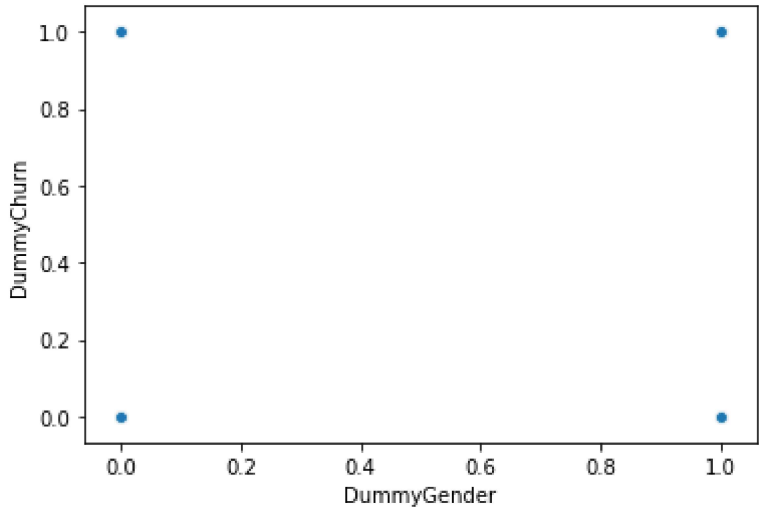
In [28]:
```
sns.scatterplot(x=churn_df['Income'], y=churn_df['DummyChurn'])
plt.show()
```



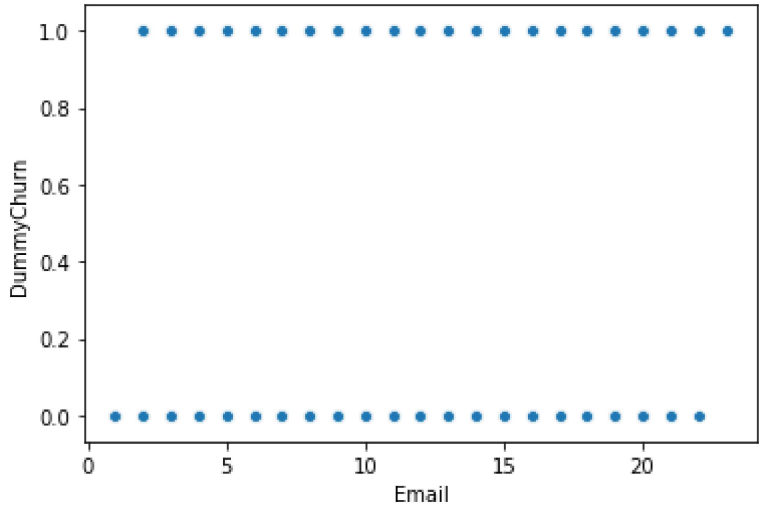
In [29]:
```
sns.scatterplot(x=churn_df['Outage_sec_perweek'], y=churn_df['DummyChurn'])
plt.show()
```



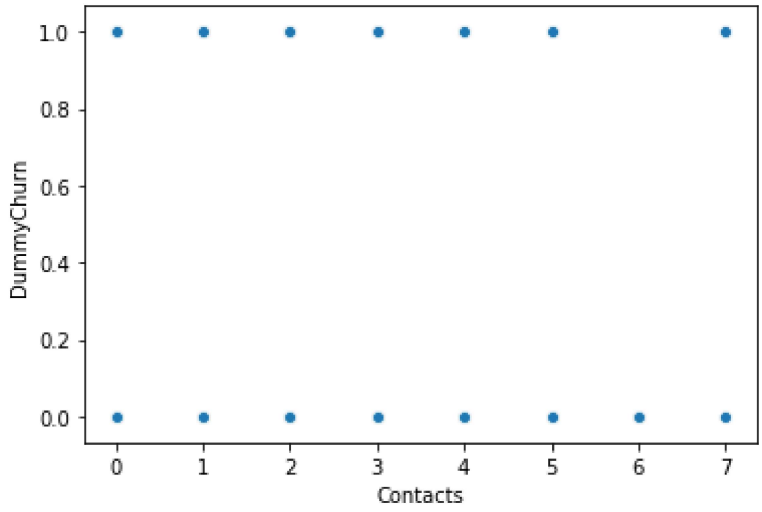
In [30]:
```
sns.scatterplot(x=churn_df['DummyGender'], y=churn_df['DummyChurn'])
plt.show()
```



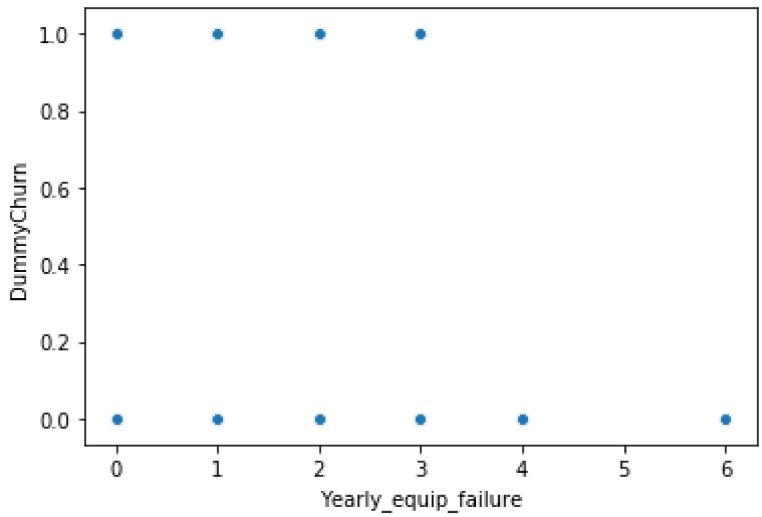
In [31]:
```
sns.scatterplot(x=churn_df['Email'], y=churn_df['DummyChurn'])
plt.show()
```



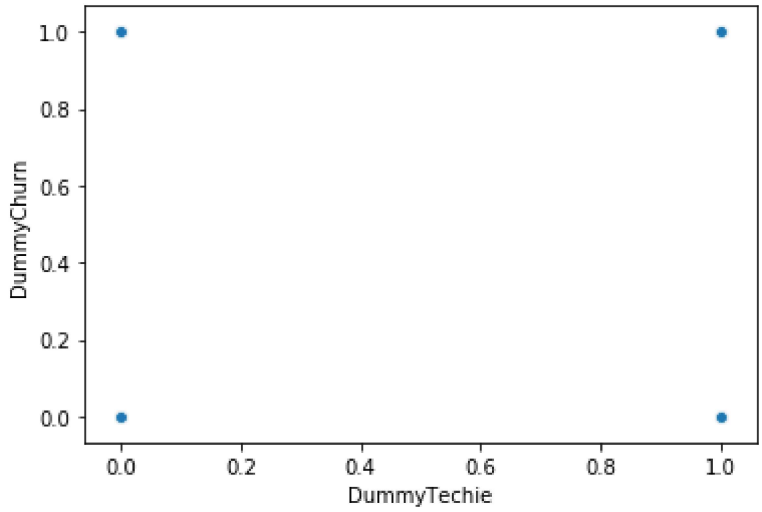
In [32]:
```
sns.scatterplot(x=churn_df['Contacts'], y=churn_df['DummyChurn'])
plt.show()
```



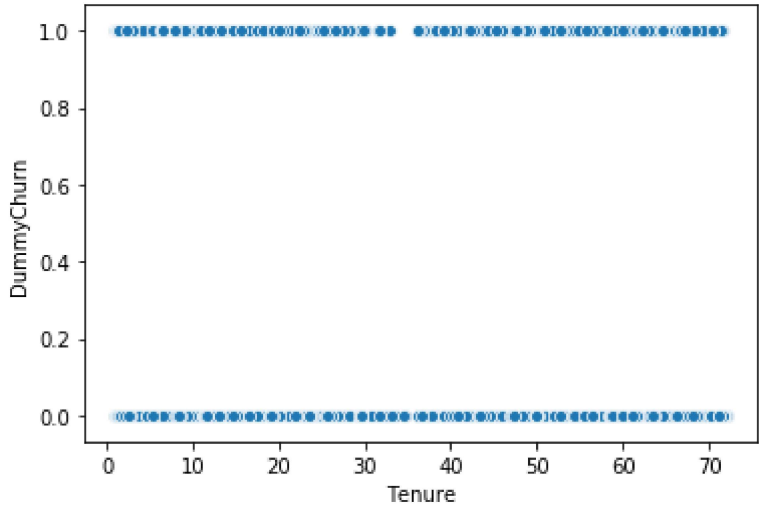
In [33]:
```
sns.scatterplot(x=churn_df['Yearly_equip_failure'], y=churn_df['DummyChurn'])
plt.show()
```



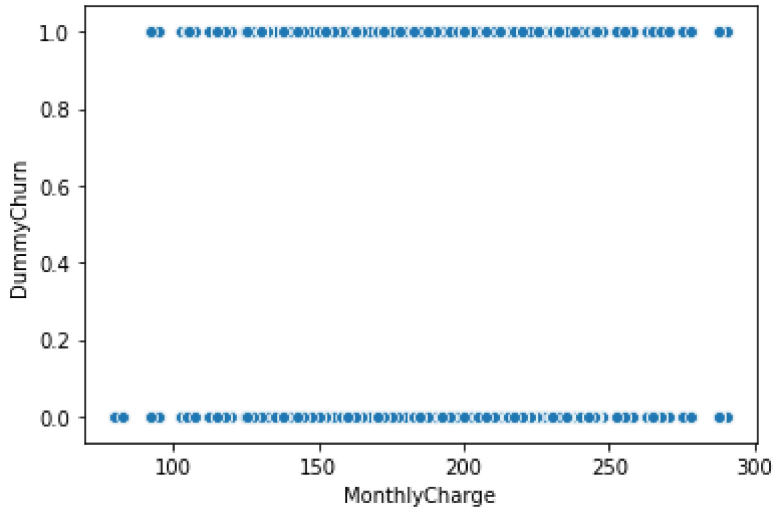
In [34]:
```
sns.scatterplot(x=churn_df['DummyTechie'], y=churn_df['DummyChurn'])
plt.show()
```



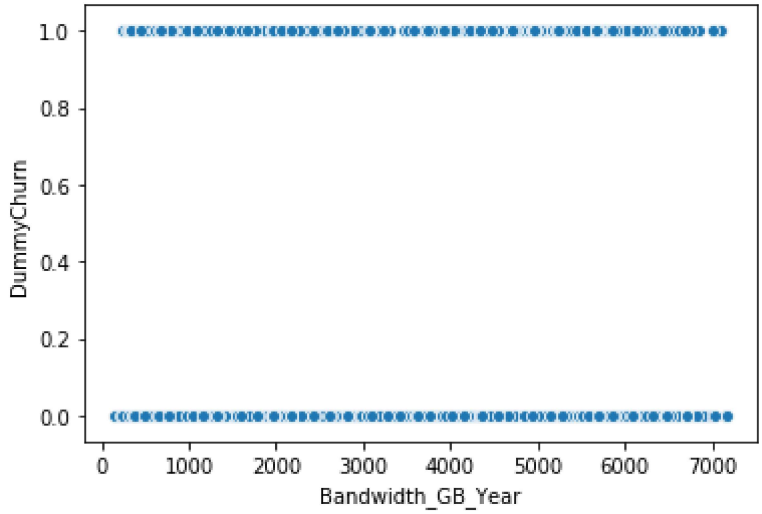
In [35]:
```
sns.scatterplot(x=churn_df['Tenure'], y=churn_df['DummyChurn'])
plt.show()
```



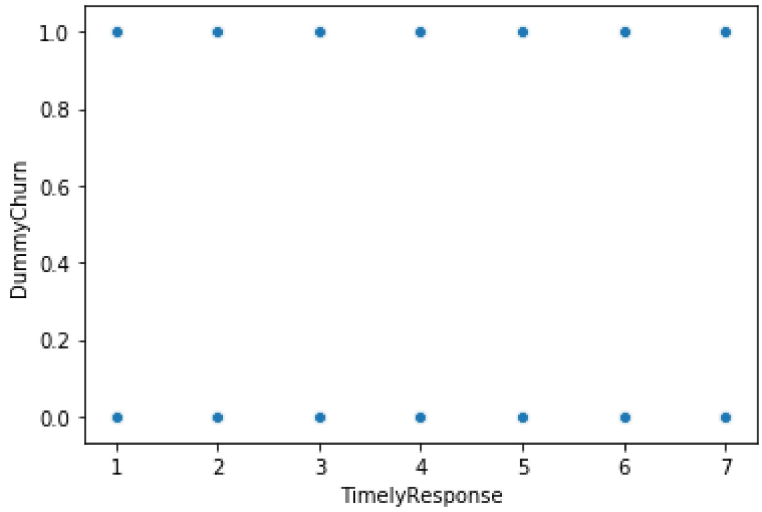
In [36]:
```
sns.scatterplot(x=churn_df['MonthlyCharge'], y=churn_df['DummyChurn'])
plt.show()
```



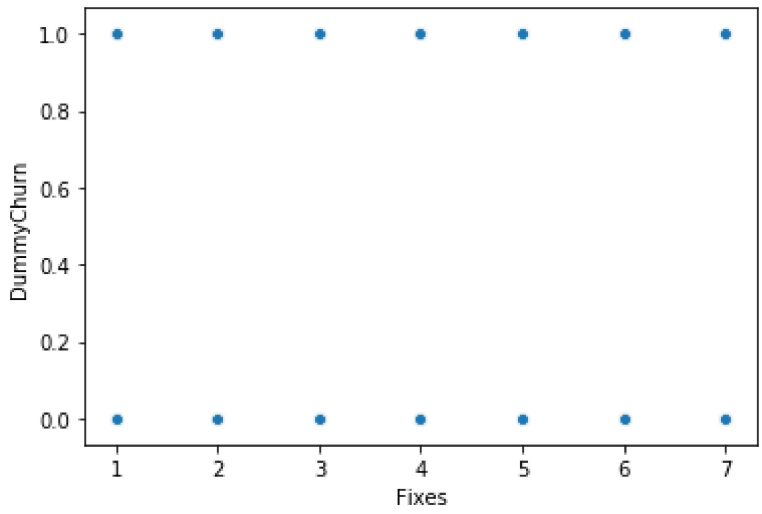
In [37]:
```
sns.scatterplot(x=churn_df['Bandwidth_GB_Year'], y=churn_df['DummyChurn'])
plt.show()
```



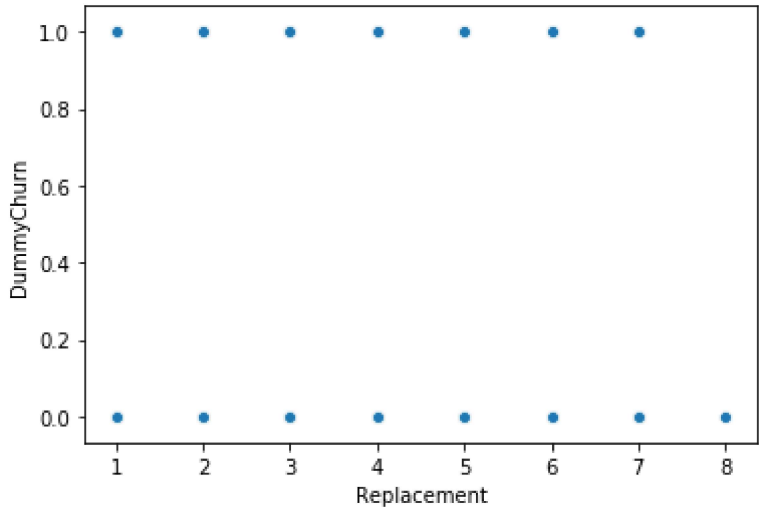
In [38]:
```
sns.scatterplot(x=churn_df['TimelyResponse'], y=churn_df['DummyChurn'])
plt.show()
```



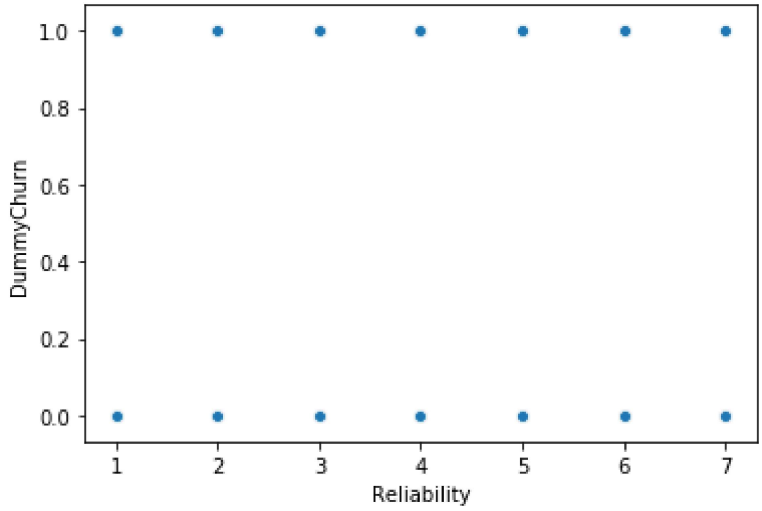
In [39]:
```
sns.scatterplot(x=churn_df['Fixes'], y=churn_df['DummyChurn'])
plt.show()
```



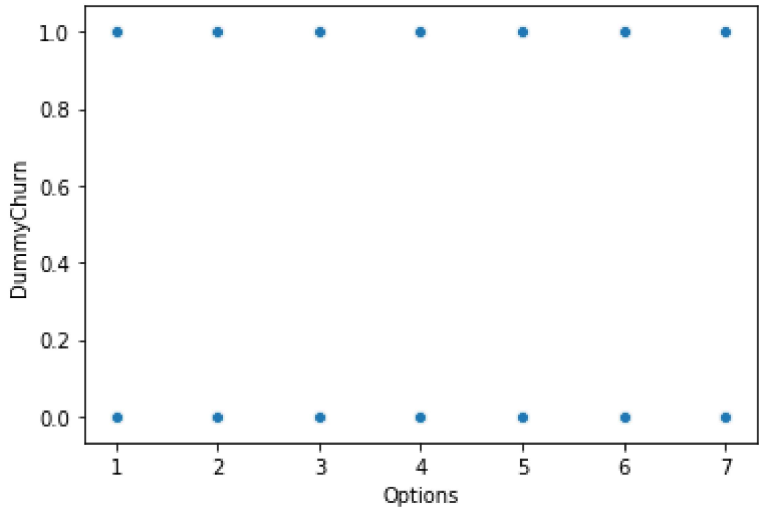
In [40]:
```
sns.scatterplot(x=churn_df['Replacement'], y=churn_df['DummyChurn'])
plt.show()
```



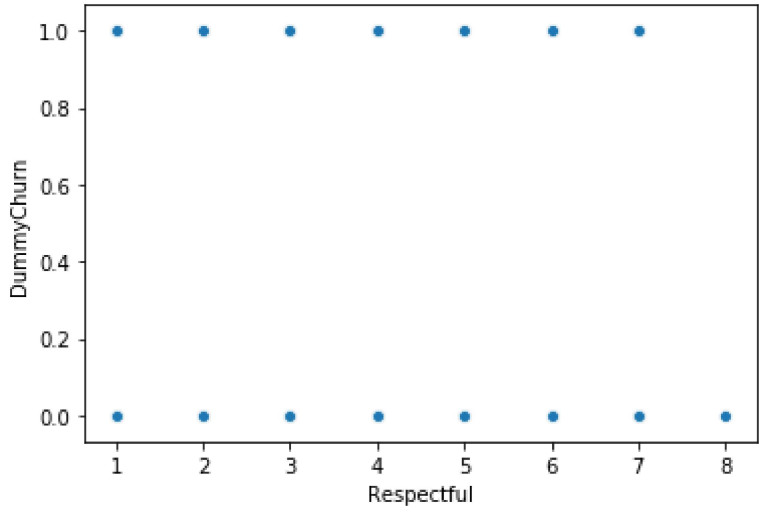
In [41]:
```
sns.scatterplot(x=churn_df['Reliability'], y=churn_df['DummyChurn'])
plt.show()
```



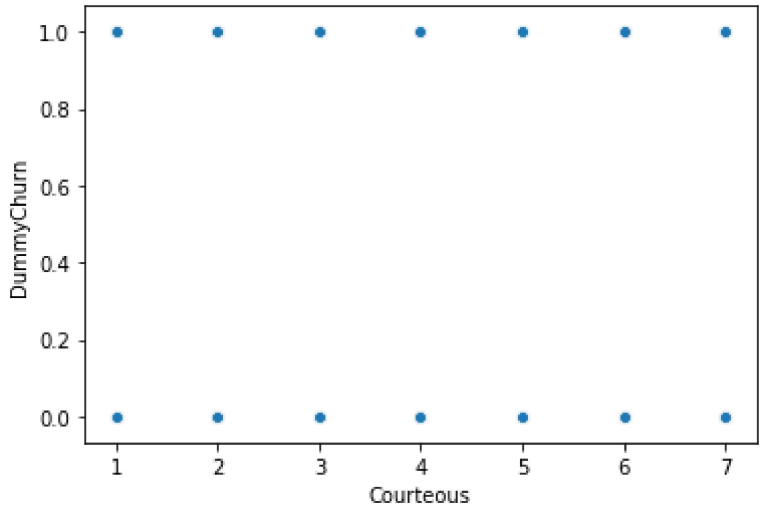
In [42]:
```
sns.scatterplot(x=churn_df['Options'], y=churn_df['DummyChurn'])
plt.show()
```



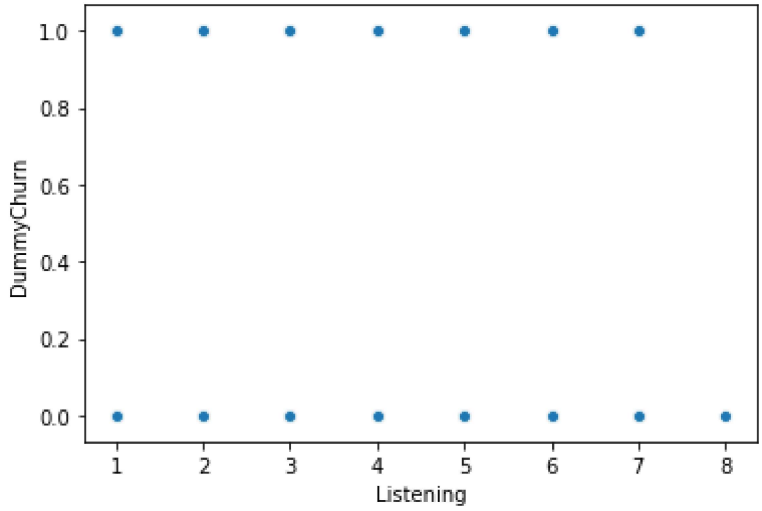
In [43]:
```
sns.scatterplot(x=churn_df['Respectful'], y=churn_df['DummyChurn'])
plt.show()
```



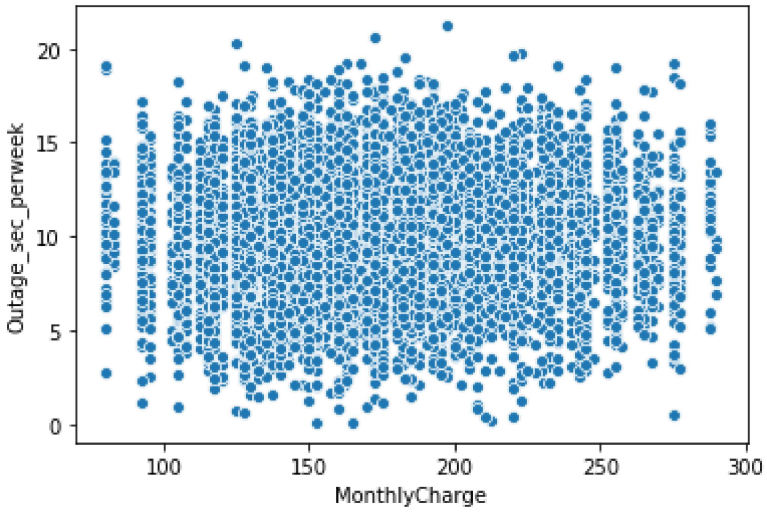
In [44]:
```
sns.scatterplot(x=churn_df['Courteous'], y=churn_df['DummyChurn'])
plt.show()
```



In [45]:
```
sns.scatterplot(x=churn_df['Listening'], y=churn_df['DummyChurn'])
plt.show()
```



In [46]:
```
sns.scatterplot(x=churn_df['MonthlyCharge'], y=churn_df['Outage_sec_perweek'])
plt.show()
```



In [57]:
```
#Extract cleaned dataset
churn_df.to_csv('churn_prepared_Dataset.csv')
```

In [48]:
```
#Initial logistics regression model construct
churn_df = pd.read_csv('churn_prepared_Dataset.csv')
churn_df['intercept'] = 1
churn_df = pd.get_dummies(churn_df, drop_first = True)
churn_logit_model = sm.Logit(churn_df['DummyChurn'],
                             churn_df[['Children',
                                       'Age',
                                       'Income',
                                       'Outage_sec_perweek',
                                       'Email',
                                       'Contacts',
                                       'Yearly_equip_failure',
                                       'Tenure',
                                       'MonthlyCharge',
                                       'Bandwidth_GB_Year',
                                       'TimelyResponse',
                                       'Fixes',
                                       'Replacement',
                                       'Reliability',
                                       'Options',
                                       'Respectful',
                                       'Courteous',
                                       'Listening',
                                       'intercept']]).fit()
print(churn_logit_model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.319573
         Iterations 8
                           Logit Regression Results
==============================================================================
=
Dep. Variable:             DummyChurn   No. Observations:                1000
0
Model:                          Logit   Df Residuals:                     998
1
Method:                           MLE   Df Model:                           1
8
Date:                Wed, 04 May 2022   Pseudo R-squ.:                  0.447
3
Time:                        11:42:47   Log-Likelihood:                -3195.
7
converged:                       True   LL-Null:                       -5782.
2
Covariance Type:            nonrobust   LLR p-value:                     0.00
0
==============================================================================
===========
                             coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
-----------
Children                -0.0980      0.016     -6.318      0.000      -0.128
-0.068
Age                      0.0114      0.002      7.130      0.000       0.008
0.015
Income                5.015e-07   1.12e-06      0.450      0.653   -1.68e-06
2.69e-06
Outage_sec_perweek      -0.0009      0.011     -0.087      0.931      -0.022
0.020
Email                    0.0018      0.010      0.169      0.866      -0.019
0.022
Contacts                 0.0243      0.032      0.764      0.445      -0.038
0.087
Yearly_equip_failure    -0.0267      0.050     -0.539      0.590      -0.124
0.071
Tenure                  -0.3156      0.012    -25.482      0.000      -0.340
-0.291
MonthlyCharge            0.0262      0.001     27.344      0.000       0.024
0.028
Bandwidth_GB_Year        0.0029      0.000     20.156      0.000       0.003
0.003
TimelyResponse          -0.0201      0.045     -0.447      0.655      -0.108
0.068
Fixes                   -0.0162      0.042     -0.384      0.701      -0.099
0.067
Replacement             -0.0053      0.039     -0.138      0.890      -0.081
0.070
Reliability             -0.0376      0.034     -1.096      0.273      -0.105
0.030
Options                 -0.0439      0.036     -1.223      0.221      -0.114
0.026
Respectful              -0.0044      0.037     -0.119      0.906      -0.076
```

```
0.068
Courteous                -0.0203      0.035     -0.580      0.562      -0.089
0.048
Listening                -0.0024      0.033     -0.071      0.943      -0.067
0.062
intercept                -5.4290      0.369    -14.709      0.000      -6.152
-4.706
==============================================================================
```

In [49]:
```
#Model with dummy variables
churn_df = pd.read_csv('churn_prepared_Dataset.csv')
churn_df['intercept'] = 1
churn_df = pd.get_dummies(churn_df, drop_first = True)
churn_logit_model_2 = sm.Logit(churn_df['DummyChurn'],
                               churn_df[['Children',
                                         'Age',
                                         'Income',
                                         'Outage_sec_perweek',
                                         'Email',
                                         'Contacts',
                                         'Yearly_equip_failure',
                                         'Tenure',
                                         'MonthlyCharge',
                                         'Bandwidth_GB_Year',
                                         'TimelyResponse',
                                         'Fixes',
                                         'Replacement',
                                         'Reliability',
                                         'Options',
                                         'Respectful',
                                         'Courteous',
                                         'Listening',
                                          'DummyGender',
                                          'DummyTechie',
                                          'DummyContract',
                                          'DummyPort_modem',
                                           'DummyTablet',
                                          'DummyInternetService',
                                          'DummyPhone',
                                          'DummyMultiple',
                                          'DummyOnlineSecurity',
                                          'DummyOnlineBackup',
                                          'DummyDeviceProtection',
                                          'DummyTechSupport',
                                          'DummyStreamingTV',
                                          'DummyStreamingMovies',
                                          'DummyPaperlessBilling',
                                          'intercept']]).fit()
print(churn_logit_model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.270966
         Iterations 8
                           Logit Regression Results
==============================================================================
=
Dep. Variable:             DummyChurn   No. Observations:                1000
0
Model:                          Logit   Df Residuals:                     998
1
Method:                           MLE   Df Model:                           1
8
Date:                Wed, 04 May 2022   Pseudo R-squ.:                  0.447
3
Time:                        11:42:47   Log-Likelihood:                -3195.
7
converged:                       True   LL-Null:                       -5782.
2
Covariance Type:            nonrobust   LLR p-value:                     0.00
0
==============================================================================
===========
                             coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
-----------
Children                -0.0980      0.016     -6.318      0.000      -0.128
-0.068
Age                      0.0114      0.002      7.130      0.000       0.008
0.015
Income                5.015e-07   1.12e-06      0.450      0.653   -1.68e-06
2.69e-06
Outage_sec_perweek      -0.0009      0.011     -0.087      0.931      -0.022
0.020
Email                    0.0018      0.010      0.169      0.866      -0.019
0.022
Contacts                 0.0243      0.032      0.764      0.445      -0.038
0.087
Yearly_equip_failure    -0.0267      0.050     -0.539      0.590      -0.124
0.071
Tenure                  -0.3156      0.012    -25.482      0.000      -0.340
-0.291
MonthlyCharge            0.0262      0.001     27.344      0.000       0.024
0.028
Bandwidth_GB_Year        0.0029      0.000     20.156      0.000       0.003
0.003
TimelyResponse          -0.0201      0.045     -0.447      0.655      -0.108
0.068
Fixes                   -0.0162      0.042     -0.384      0.701      -0.099
0.067
Replacement             -0.0053      0.039     -0.138      0.890      -0.081
0.070
Reliability             -0.0376      0.034     -1.096      0.273      -0.105
0.030
Options                 -0.0439      0.036     -1.223      0.221      -0.114
0.026
Respectful              -0.0044      0.037     -0.119      0.906      -0.076
```

```
0.068
Courteous                -0.0203      0.035     -0.580      0.562      -0.089
0.048
Listening                -0.0024      0.033     -0.071      0.943      -0.067
0.062
intercept                -5.4290      0.369    -14.709      0.000      -6.152
-4.706
==============================================================================
```

In [50]:
```
#Reduced ols regression
churn_df['intercept'] = 1
churn_logit_reduced = sm.Logit(churn_df['DummyChurn'],
                               churn_df[['Children',
                                         'Age',
                                         'DummyTechie',
                                         'DummyContract',
                                         'DummyPort_modem',
                                         'DummyInternetService',
                                         'DummyPhone',
                                         'DummyMultiple',
                                         'DummyOnlineSecurity',
                                         'DummyOnlineBackup',
                                         'DummyDeviceProtection',
                                         'DummyTechSupport',
                                         'Tenure',
                                        'MonthlyCharge',
                                        'Bandwidth_GB_Year',
                                         'intercept']]).fit()
print(churn_logit_reduced.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.272362
         Iterations 8
                           Logit Regression Results
==============================================================================
=
Dep. Variable:             DummyChurn   No. Observations:                1000
0
Model:                          Logit   Df Residuals:                     998
4
Method:                           MLE   Df Model:                           1
5
Date:                Wed, 04 May 2022   Pseudo R-squ.:                  0.529
0
Time:                        11:42:47   Log-Likelihood:                -2723.
6
converged:                       True   LL-Null:                       -5782.
2
Covariance Type:            nonrobust   LLR p-value:                     0.00
0
==============================================================================
============
                               coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
------------
Children                    -0.0391      0.018     -2.221      0.026      -0.074
-0.005
Age                          0.0070      0.002      3.735      0.000       0.003
0.011
DummyTechie                  0.7970      0.089      8.996      0.000       0.623
0.971
DummyContract               -2.2895      0.103    -22.136      0.000      -2.492
-2.087
DummyPort_modem              0.1598      0.068      2.339      0.019       0.026
0.294
DummyInternetService        -1.4240      0.125    -11.359      0.000      -1.670
-1.178
DummyPhone                  -0.3193      0.116     -2.749      0.006      -0.547
-0.092
DummyMultiple               -0.2964      0.077     -3.857      0.000      -0.447
-0.146
DummyOnlineSecurity         -0.3303      0.073     -4.497      0.000      -0.474
-0.186
DummyOnlineBackup           -0.5146      0.072     -7.125      0.000      -0.656
-0.373
DummyDeviceProtection       -0.4075      0.070     -5.790      0.000      -0.545
-0.270
DummyTechSupport            -0.3555      0.073     -4.892      0.000      -0.498
-0.213
Tenure                      -0.2049      0.021     -9.770      0.000      -0.246
-0.164
MonthlyCharge                0.0463      0.002     25.620      0.000       0.043
0.050
Bandwidth_GB_Year            0.0013      0.000      5.279      0.000       0.001
0.002
intercept                   -6.1973      0.236    -26.280      0.000      -6.659
```

```
-5.735
================================================================================
============
```

In [51]:
```
#Confusion Matrix
#Import dataset
dataset = pd.read_csv('churn_prepared_Dataset.csv')
X = dataset.iloc[:, 1:-1].values
y = dataset.iloc[:, -1].values
```

In [52]:
```
#split into test and training set
from sklearn.model_selection import train_test_split
```

In [53]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, ran
dom_state = 0)
```

In [54]:
```
#train logistic model on training set
from sklearn.linear_model import LogisticRegression
```

In [55]:
```
classifier = LogisticRegression(random_state = 0)
```

In [56]:
```
classifier.fit(X_train, y_train)
```

Out[56]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=0, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [58]:
```
#predict test results
y_pred = classifier.predict(X_test)
```

In [59]:
```
#Confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

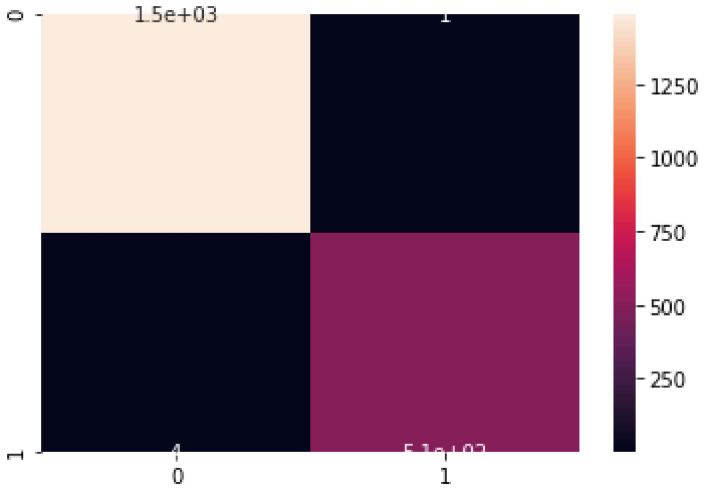
```
[[1485    1]
 [   4  510]]
```

In [60]:
```
#accuracy with k fold
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train,
cv = 10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 97.37 %
Standard Deviation: 2.86 %
```

In [62]:
```
y_predict_test = classifier.predict(X_test)
cm2 = confusion_matrix(y_test, y_predict_test)
sns.heatmap(cm2, annot=True)
```

Out[62]: <matplotlib.axes._subplots.AxesSubplot at 0x1f5e06d3648>



In [63]:
```
#Classification report
from sklearn.metrics import classification_report
print(classification_report(y_test, y_predict_test))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1486
           1       1.00      0.99      1.00       514

    accuracy                           1.00      2000
   macro avg       1.00      1.00      1.00      2000
weighted avg       1.00      1.00      1.00      2000
```

In [ ]:
```

```