Multiple Linear Regression for Predictive Modeling

Data Preparation

In [49]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
import statsmodels.api as sm
import statsmodels.formula.api as smf
from sklearn import metrics
import warnings
warnings.filterwarnings('ignore')
```

In [50]:
```
df = pd.read_csv('/Users/ebeth/Desktop/Churn Data/churn_clean.csv')
```

Check the data set loaded and see the first 5 rows.

In [51]:
```
df.head()
```

Out[51]:

	CaseOrder	Customer_id	Interaction	UID	City	Sta
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	

5 rows × 50 columns

Check to see the number of columns and rows

In [52]:
```
df.shape
```

Out[52]:
```
(10000, 50)
```

Look at the names and data types of each columns.

In [53]:
```
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
```

Drop columns that are not being used.

In [54]:
```
df2 = df.drop(['CaseOrder','Customer_id','Interaction', 'UID','Lat','Lng','Ti
```

Take a closer look at state and zip. I am not sure if they should be included so I will look at how many distinct values are present. Having too many will make the final regression equation unwieldy.

In [55]:
```
df2.nunique(axis = 0)
```

Out[55]:
```
State                      52
Zip                      8583
Population               5933
Area                        3
Children                   11
Age                        72
Income                   9993
Marital                     5
Gender                      3
Churn                       2
Outage_sec_perweek       9986
Email                      23
Contacts                    8
Yearly_equip_failure        6
Techie                      2
Contract                    3
Port_modem                  2
Tablet                      2
InternetService             3
Phone                       2
Multiple                    2
OnlineSecurity              2
OnlineBackup                2
DeviceProtection            2
TechSupport                 2
StreamingTV                 2
StreamingMovies             2
PaperlessBilling            2
PaymentMethod               4
Tenure                   9996
MonthlyCharge             750
Bandwidth_GB_Year       10000
Item1                       7
Item2                       7
Item3                       8
Item4                       7
Item5                       7
Item6                       8
Item7                       7
Item8                       8
dtype: int64
```

There are 52 states and 8583 unique zip codes present. For this overall look at tunure length having this many variables would not be helpful. Later, going back and exploring maybe state by state would be benifical. State and Zip will be dropped from the data set.

In [56]:
```
df3 = df2.drop(['State', 'Zip'], axis = 1)
df3.head()
```

Out[56]:

	Population	Area	Children	Age	Income	Marital	Gender	Churn	Outage_sec_perv
0	38	Urban	0	68	28561.99	Widowed	Male	No	7.978
1	10446	Urban	1	27	21704.77	Married	Female	Yes	11.699
2	3735	Urban	4	50	9609.57	Widowed	Female	No	10.752
3	13863	Suburban	1	48	18925.23	Married	Male	No	14.913
4	11352	Suburban	0	83	40074.19	Separated	Male	Yes	8.14

5 rows × 38 columns

Look for missing values, duplicates, and outliers.

In [57]:

```
df3.isnull().count()
```

Out[57]:

```
Population              10000
Area                    10000
Children                10000
Age                     10000
Income                  10000
Marital                 10000
Gender                  10000
Churn                   10000
Outage_sec_perweek      10000
Email                   10000
Contacts                10000
Yearly_equip_failure    10000
Techie                  10000
Contract                10000
Port_modem              10000
Tablet                  10000
InternetService         10000
Phone                   10000
Multiple                10000
OnlineSecurity          10000
OnlineBackup            10000
DeviceProtection        10000
TechSupport             10000
StreamingTV             10000
StreamingMovies         10000
PaperlessBilling        10000
PaymentMethod           10000
Tenure                  10000
MonthlyCharge           10000
Bandwidth_GB_Year       10000
Item1                   10000
Item2                   10000
Item3                   10000
Item4                   10000
Item5                   10000
Item6                   10000
Item7                   10000
Item8                   10000
dtype: int64
```

In [58]:

```
df3.duplicated()
```

```
Out[58]: 0       False
         1       False
         2       False
         3       False
         4       False
                 ...
         9995    False
         9996    False
         9997    False
         9998    False
         9999    False
         Length: 10000, dtype: bool
```

```
In [59]: df3.describe(include=[np.number])
```

Out[59]:

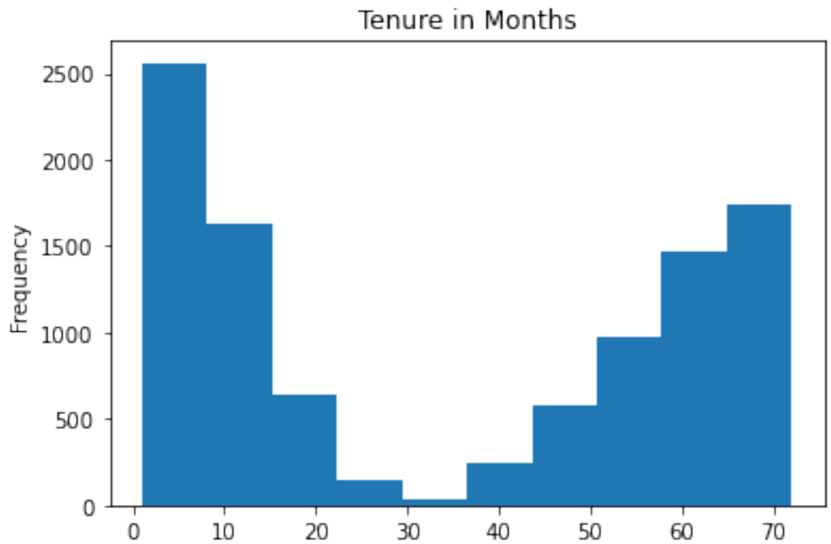
	Population	Children	Age	Income	Outage_sec_perweek	
count	10000.000000	10000.0000	10000.000000	10000.000000	10000.000000	10000.
mean	9756.562400	2.0877	53.078400	39806.926771	10.001848	12
std	14432.698671	2.1472	20.698882	28199.916702	2.976019	3.
min	0.000000	0.0000	18.000000	348.670000	0.099747	1.
25%	738.000000	0.0000	35.000000	19224.717500	8.018214	10.
50%	2910.500000	1.0000	53.000000	33170.605000	10.018560	12.
75%	13168.000000	3.0000	71.000000	53246.170000	11.969485	14.
max	111850.000000	10.0000	89.000000	258900.700000	21.207230	23.

There are no Nans or duplicates. From a glance, no minimum or maximum value seems unrealistic.

Univariate Visualizations

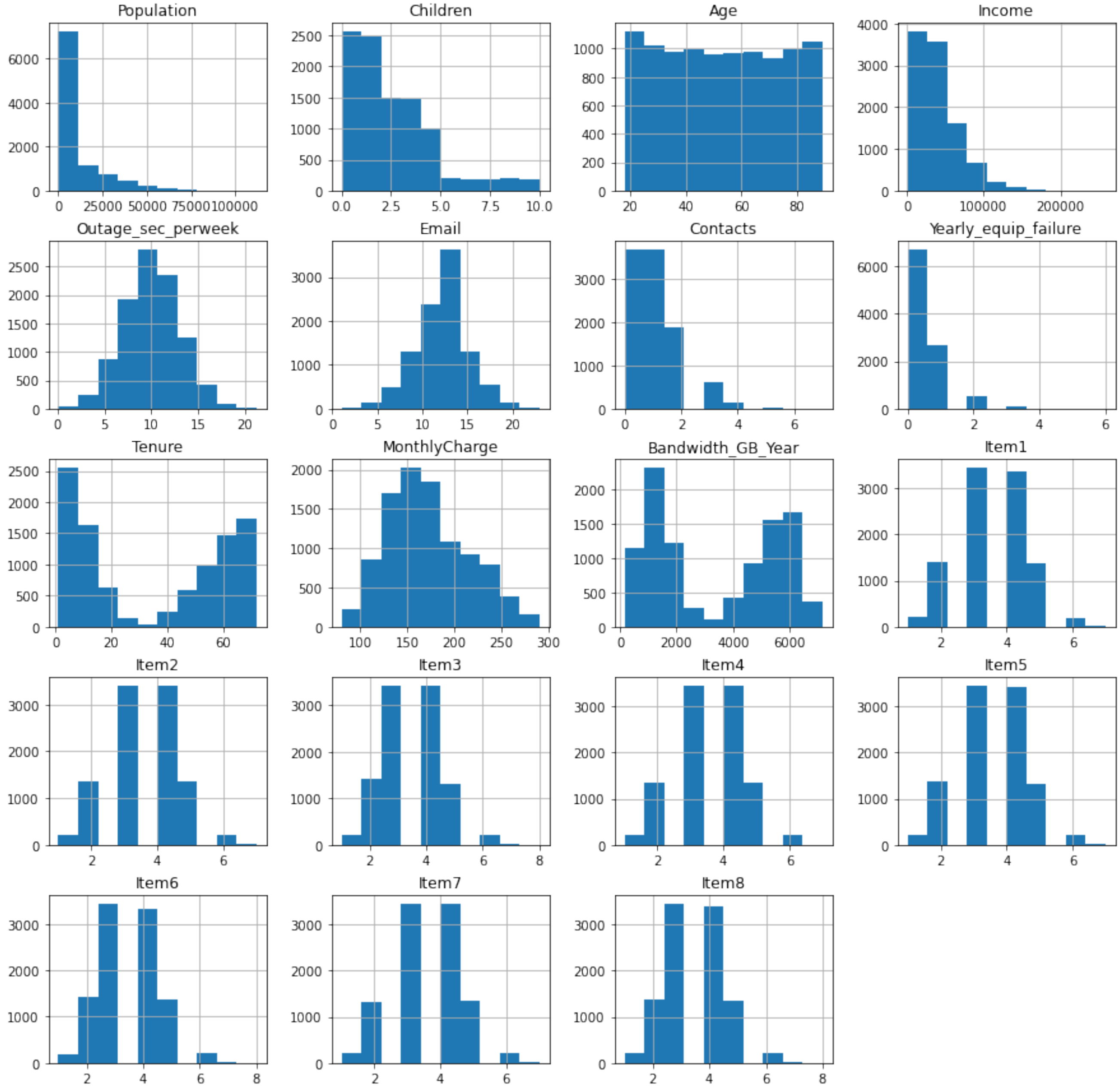
Target Variable Tenure

```
In [60]: df3['Tenure'].plot.hist(title = 'Tenure in Months');
```

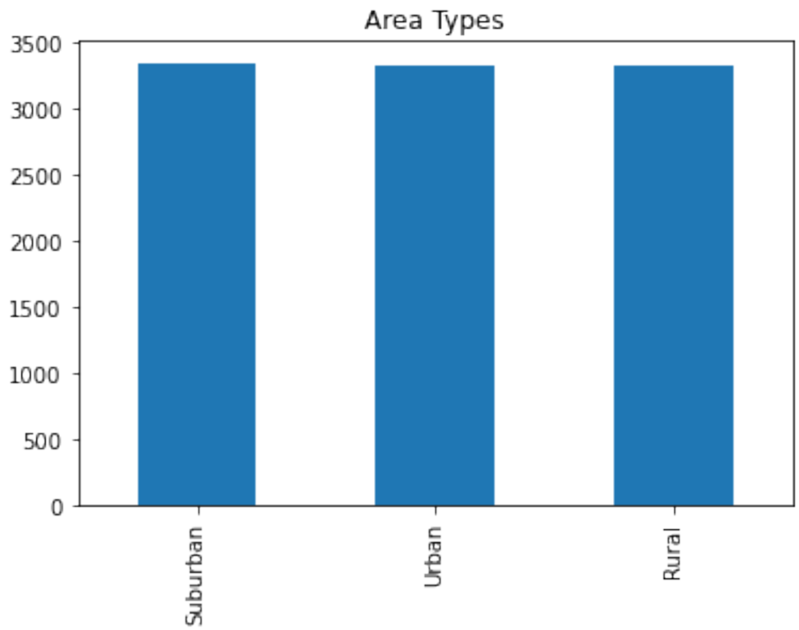


Predictor Variables

In [61]:
```
df3.hist(figsize = (15,15));
```

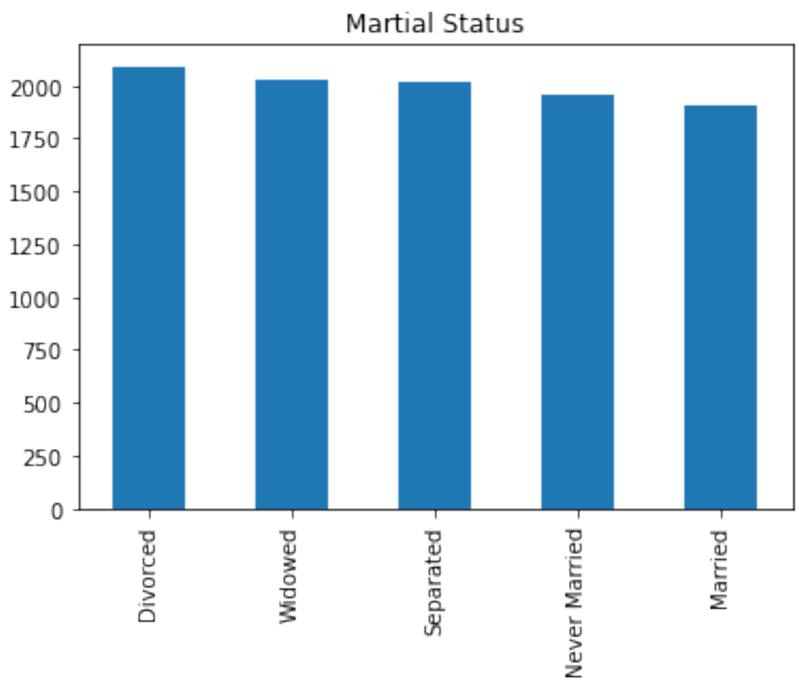


In [62]:
```
df3['Area'].value_counts().plot.bar(title = 'Area Types');
```



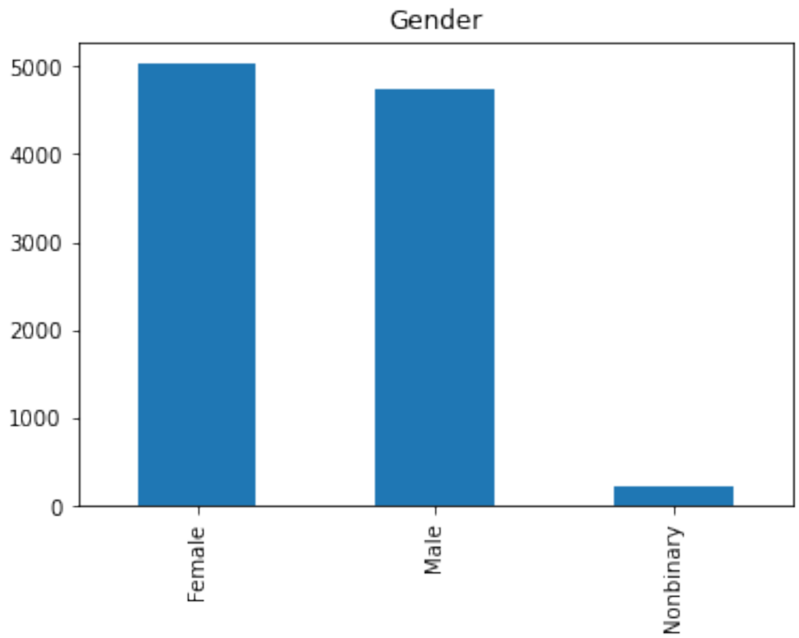
In [63]:

```
df3['Marital'].value_counts().plot.bar(title = 'Martial Status');
```

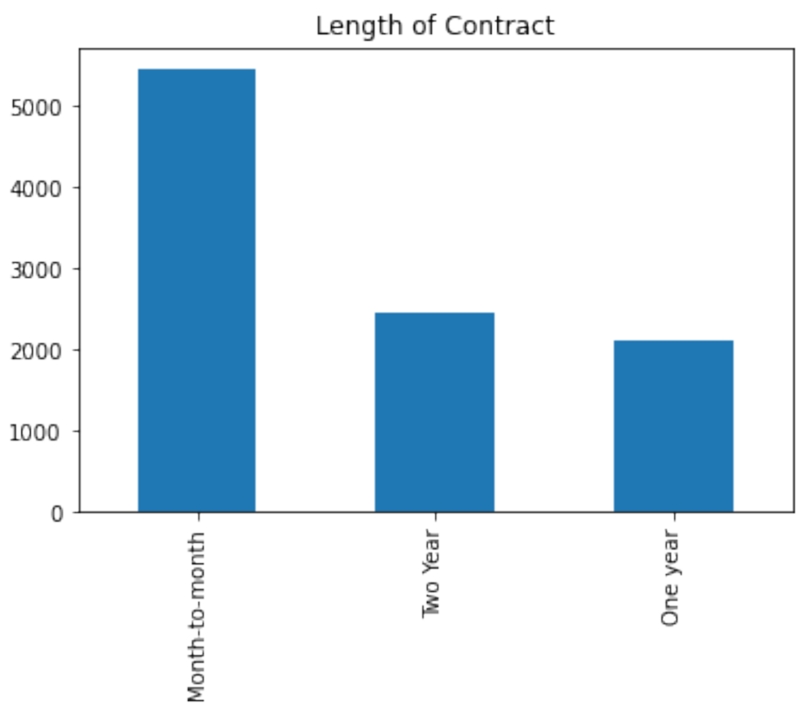


In [64]:

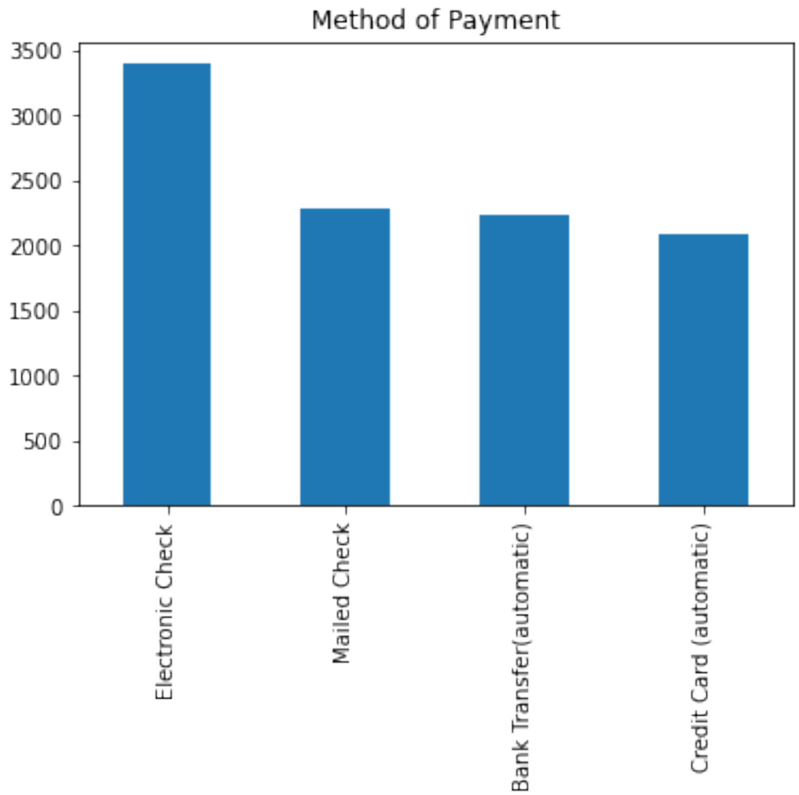
```
df3['Gender'].value_counts().plot.bar(title = 'Gender');
```



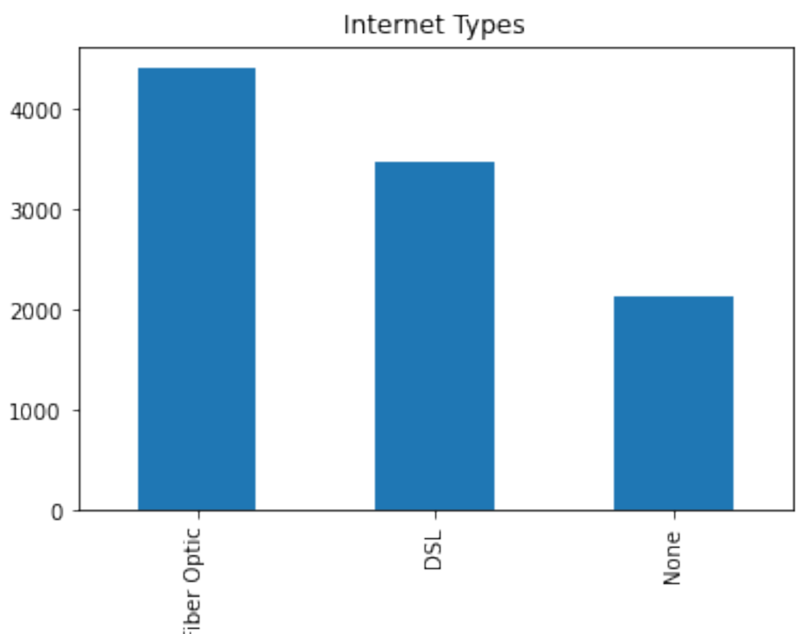
In [65]:
```
df3['Contract'].value_counts().plot.bar(title = 'Length of Contract');
```



In [66]:
```
df3['PaymentMethod'].value_counts().plot.bar(title = 'Method of Payment');
```

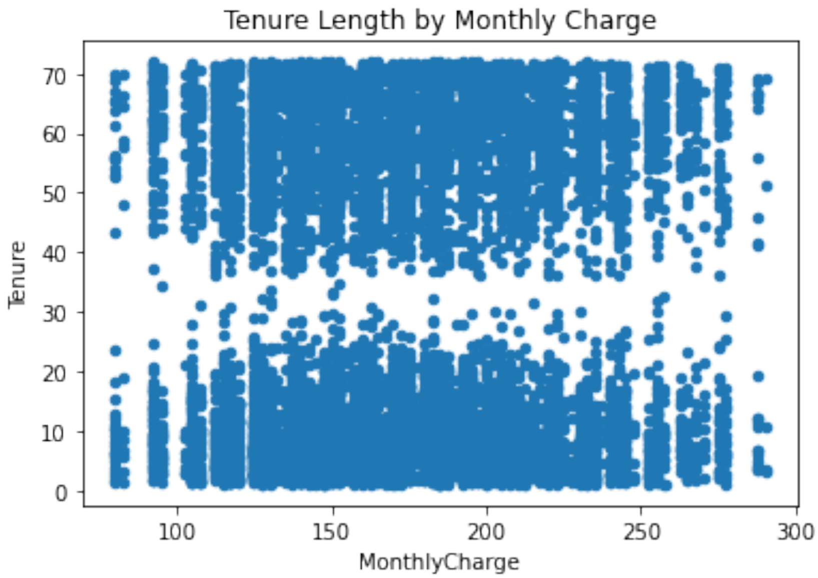


In [67]:
```
df3['InternetService'].value_counts().plot.bar(title = 'Internet Types');
```

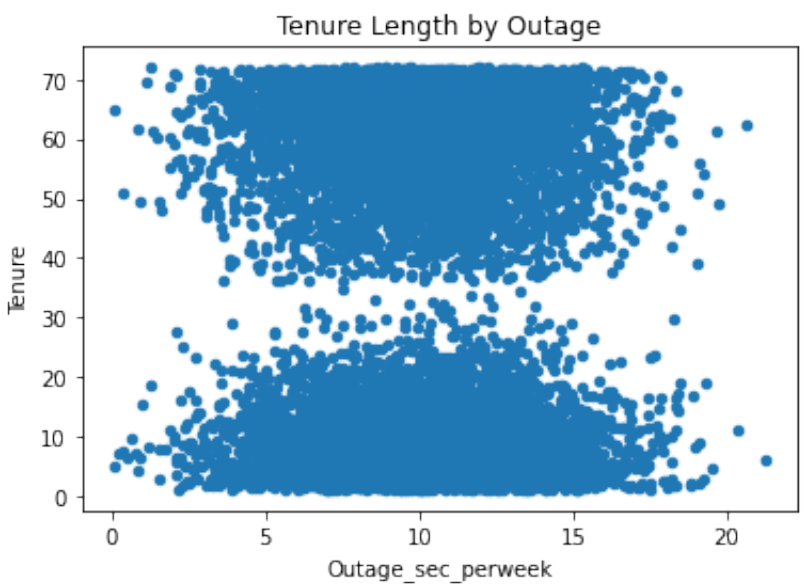


Bivariate visualizations

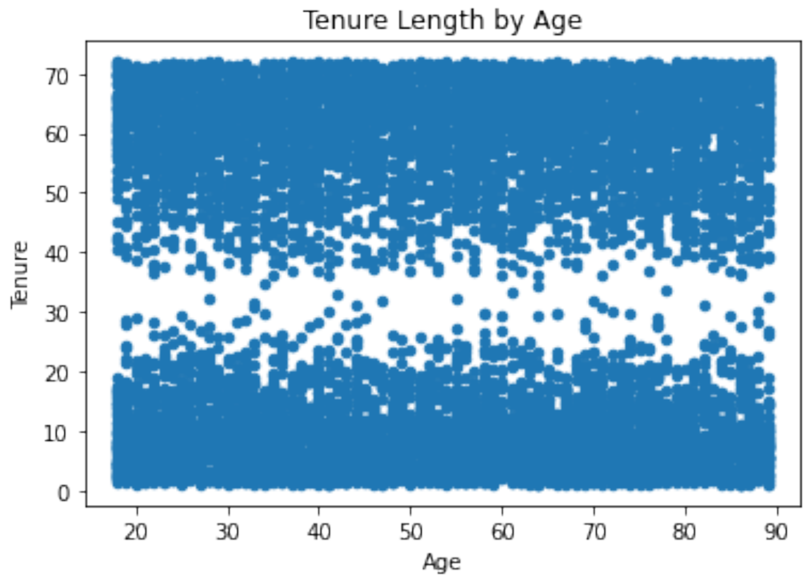
In [68]:
```
df3.plot.scatter('MonthlyCharge', 'Tenure', title = 'Tenure Length by Monthly
```



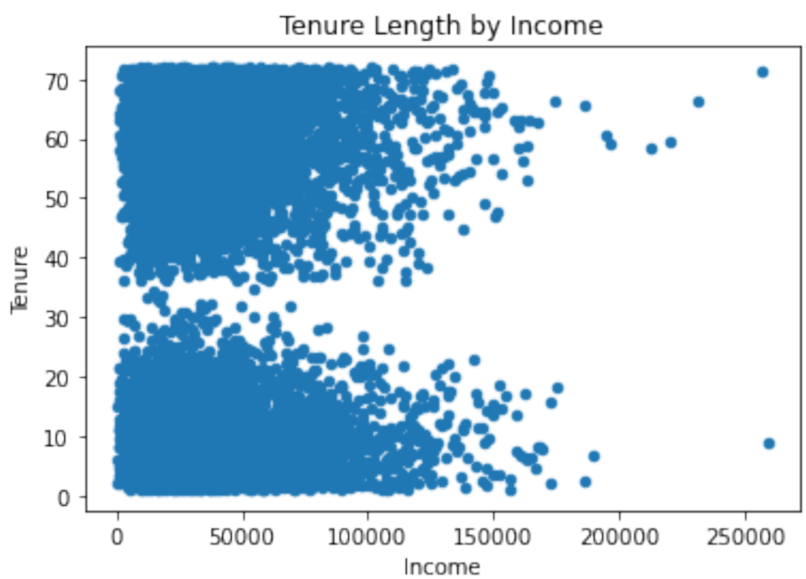
In [69]:
```
df3.plot.scatter('Outage_sec_perweek', 'Tenure', title = 'Tenure Length by Ou
```



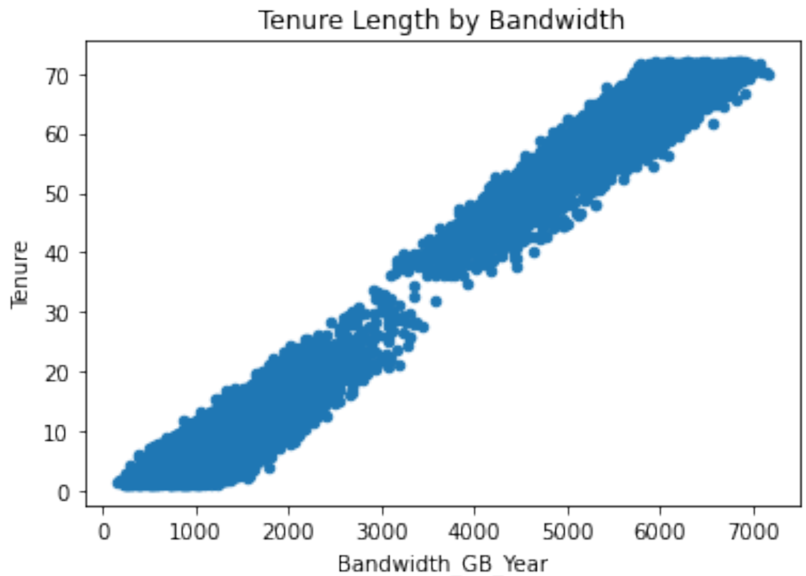
In [70]:
```
df3.plot.scatter('Age', 'Tenure', title = 'Tenure Length by Age');
```



In [71]:
```
df3.plot.scatter('Income', 'Tenure', title = 'Tenure Length by Income');
```

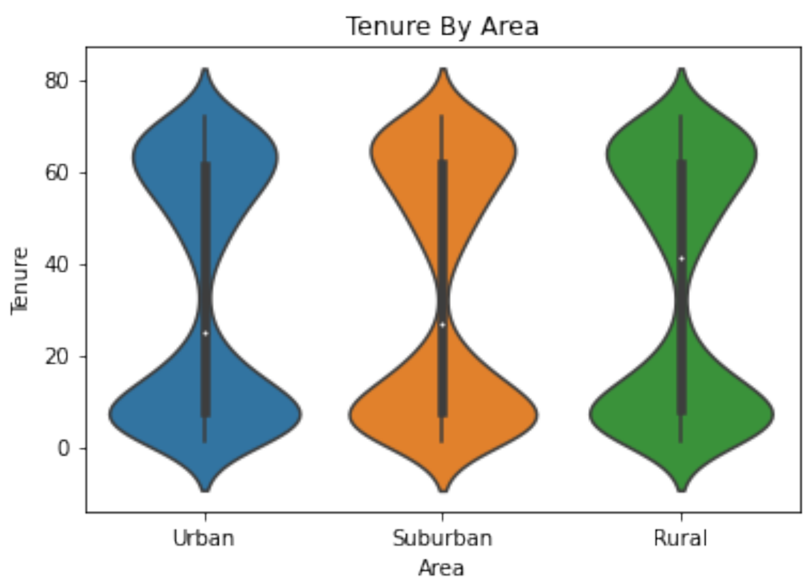


In [72]:
```
df3.plot.scatter('Bandwidth_GB_Year', 'Tenure', title = 'Tenure Length by Ban
```

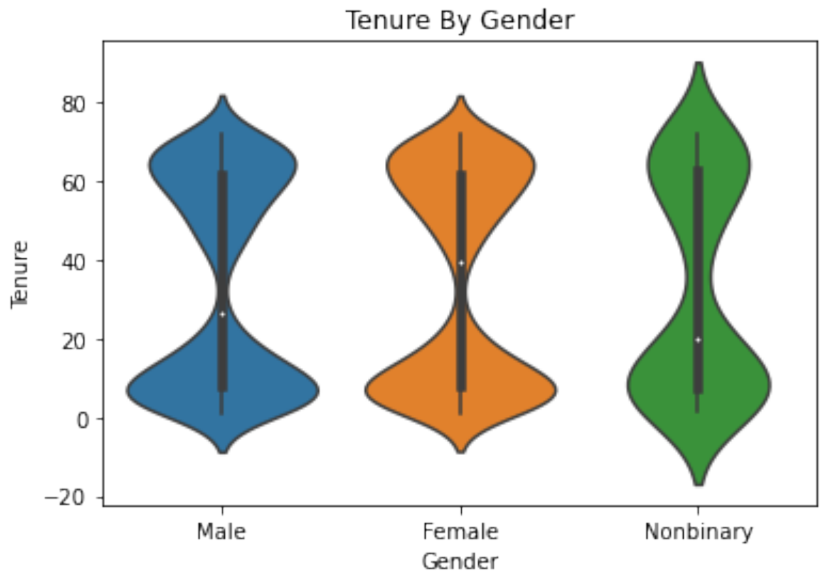


In [ ]:

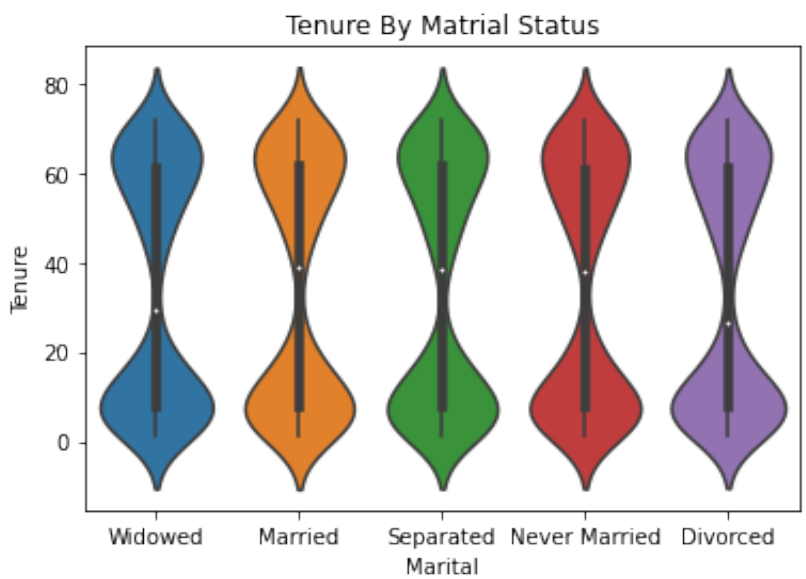
In [73]:
```
sn.violinplot(df3['Area'], df3['Tenure']).set_title('Tenure By Area');
```



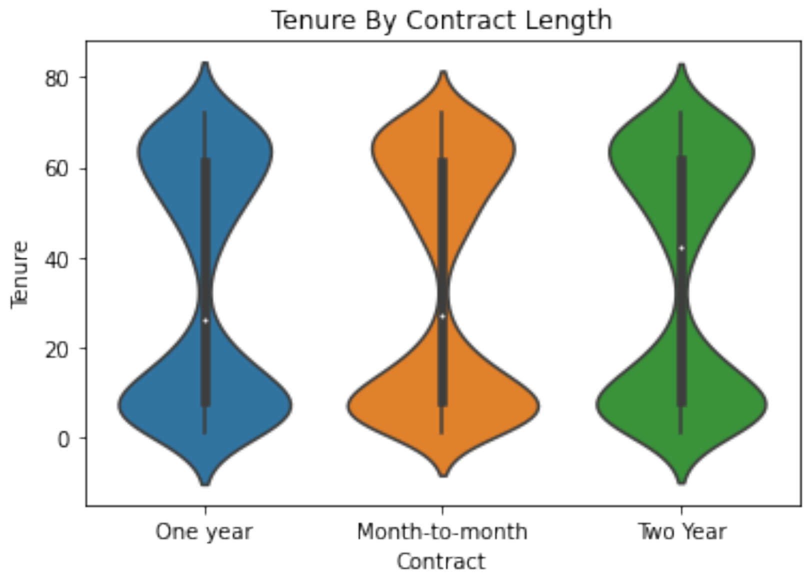
In [74]:
```
sn.violinplot(df3['Gender'], df3['Tenure']).set_title('Tenure By Gender');
```



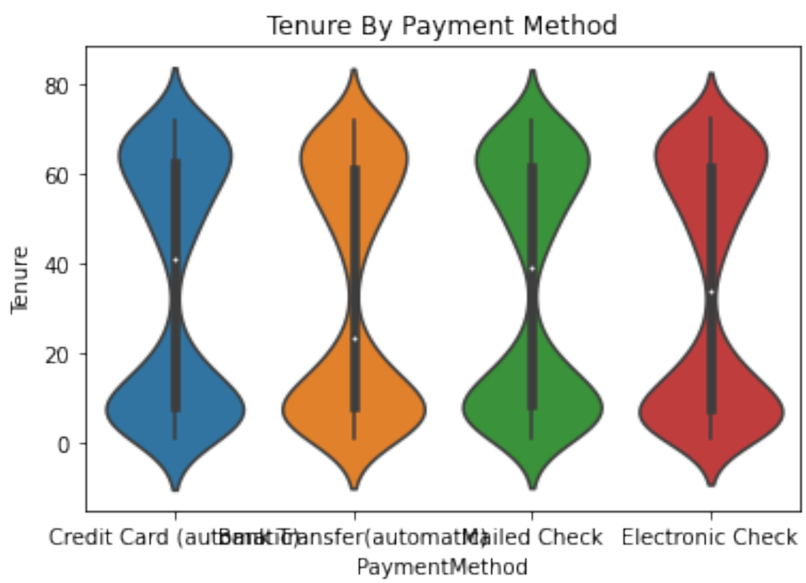
In [75]:
```
sn.violinplot(df3['Marital'], df3['Tenure']).set_title('Tenure By Matrial Sta
```



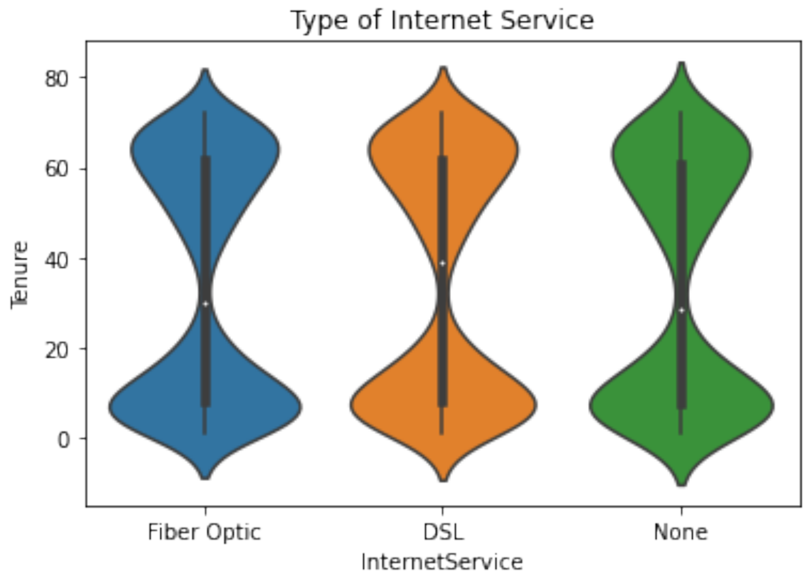
In [76]:
```
sn.violinplot(df3['Contract'], df3['Tenure']).set_title('Tenure By Contract L
```



In [77]:
```
sn.violinplot(df3['PaymentMethod'], df3['Tenure']).set_title('Tenure By Payme
```



In [78]:
```
sn.violinplot(df3['InternetService'], df3['Tenure']).set_title('Type of Inter
```



Create Dummy variables for all categorical columns and drop unneeded columns. (code used from: https://towardsdatascience.com/the-dummys-guide-to-creating-dummy-variables-f21faddb1d40)

In [79]:
```
dummy1 = pd.get_dummies(df3.Area, prefix = 'Area', drop_first = True)
dummy2 = pd.get_dummies(df3.Marital, prefix = 'Marital', drop_first = True)
dummy3 = pd.get_dummies(df3.Gender, prefix = 'Gender', drop_first = True)
dummy4 = pd.get_dummies(df3.Churn, prefix = 'Churn', drop_first = True)
dummy5 = pd.get_dummies(df3.Techie, prefix = 'Techie', drop_first = True)
dummy6 = pd.get_dummies(df3.Contract, prefix = 'Contract', drop_first = True)
dummy7 = pd.get_dummies(df3.Port_modem, prefix = 'Port_modem', drop_first = T
dummy8 = pd.get_dummies(df3.Tablet, prefix = 'Tablet', drop_first = True)
dummy9 = pd.get_dummies(df3.InternetService, prefix = 'InternetService', drop_
dummy10 = pd.get_dummies(df3.Phone, prefix = 'Phone', drop_first = True)
dummy11 = pd.get_dummies(df3.Multiple, prefix = 'Multiple', drop_first = True
dummy12 = pd.get_dummies(df3.OnlineSecurity, prefix = 'OnlineSecurity', drop_
dummy13 = pd.get_dummies(df3.OnlineBackup, prefix = 'OnlineBackup', drop_firs
dummy14 = pd.get_dummies(df3.DeviceProtection, prefix = 'DeviceProtection', d
dummy15 = pd.get_dummies(df3.TechSupport, prefix = 'TechSupport', drop_first 
dummy16 = pd.get_dummies(df3.StreamingTV, prefix = 'StreamingTV', drop_first 
dummy17 = pd.get_dummies(df3.StreamingMovies, prefix = 'StreamingMovies', dro
dummy18 = pd.get_dummies(df3.PaperlessBilling, prefix = 'PaperlessBilling', d
dummy19 = pd.get_dummies(df3.PaymentMethod, prefix = 'PaymentMethod', drop_fi

df3 = df3.drop(columns = 'Area').merge(dummy1, left_index = True, right_index
df3 = df3.drop(columns = 'Marital').merge(dummy2, left_index = True, right_in
df3 = df3.drop(columns = 'Gender').merge(dummy3, left_index = True, right_ind
df3 = df3.drop(columns = 'Churn').merge(dummy4, left_index = True, right_inde
df3 = df3.drop(columns = 'Techie').merge(dummy5, left_index = True, right_ind
df3 = df3.drop(columns = 'Contract').merge(dummy6, left_index = True, right_i
df3 = df3.drop(columns = 'Port_modem').merge(dummy7, left_index = True, right_
df3 = df3.drop(columns = 'Tablet').merge(dummy8, left_index = True, right_ind
df3 = df3.drop(columns = 'InternetService').merge(dummy9, left_index = True, 
df3 = df3.drop(columns = 'Phone').merge(dummy10, left_index = True, right_ind
df3 = df3.drop(columns = 'Multiple').merge(dummy11, left_index = True, right_
df3 = df3.drop(columns = 'OnlineSecurity').merge(dummy12, left_index = True, 
df3 = df3.drop(columns = 'OnlineBackup').merge(dummy13, left_index = True, ri
df3 = df3.drop(columns = 'DeviceProtection').merge(dummy14, left_index = True
df3 = df3.drop(columns = 'TechSupport').merge(dummy15, left_index = True, rig
df3 = df3.drop(columns = 'StreamingTV').merge(dummy16, left_index = True, rig
df3 = df3.drop(columns = 'StreamingMovies').merge(dummy17, left_index = True,
df3 = df3.drop(columns = 'PaperlessBilling').merge(dummy18, left_index = True
df3 = df3.drop(columns = 'PaymentMethod').merge(dummy19, left_index = True, r
```

In [80]:
```
df3.head()
```

Out[80]:

	Population	Children	Age	Income	Outage_sec_perweek	Email	Contacts	Yearly_equip_fa
0	38	0	68	28561.99	7.978323	10	0	
1	10446	1	27	21704.77	11.699080	12	0	
2	3735	4	50	9609.57	10.752800	9	0	
3	13863	1	48	18925.23	14.913540	15	2	
4	11352	0	83	40074.19	8.147417	16	2	

5 rows × 47 columns

Convert all columns to float type.

In [81]:

```
df4 = df3.astype(float)
df4
```

Out[81]:

	Population	Children	Age	Income	Outage_sec_perweek	Email	Contacts	Yearly_equ
0	38.0	0.0	68.0	28561.99	7.978323	10.0	0.0	
1	10446.0	1.0	27.0	21704.77	11.699080	12.0	0.0	
2	3735.0	4.0	50.0	9609.57	10.752800	9.0	0.0	
3	13863.0	1.0	48.0	18925.23	14.913540	15.0	2.0	
4	11352.0	0.0	83.0	40074.19	8.147417	16.0	2.0	
...	...	...	...	...	...	...	...	
9995	640.0	3.0	23.0	55723.74	9.415935	12.0	2.0	
9996	77168.0	4.0	48.0	34129.34	6.740547	15.0	2.0	
9997	406.0	1.0	48.0	45983.43	6.590911	10.0	0.0	
9998	35575.0	1.0	39.0	16667.58	12.071910	14.0	1.0	
9999	12230.0	1.0	28.0	9020.92	11.754720	17.0	1.0	

10000 rows × 47 columns

In [82]:

```
df4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 47 columns):
 #   Column                                 Non-Null Count  Dtype  
---  ------                                 --------------  -----  
 0   Population                             10000 non-null  float64
 1   Children                               10000 non-null  float64
 2   Age                                    10000 non-null  float64
 3   Income                                 10000 non-null  float64
 4   Outage_sec_perweek                     10000 non-null  float64
 5   Email                                  10000 non-null  float64
 6   Contacts                               10000 non-null  float64
 7   Yearly_equip_failure                   10000 non-null  float64
 8   Tenure                                 10000 non-null  float64
 9   MonthlyCharge                          10000 non-null  float64
 10  Bandwidth_GB_Year                      10000 non-null  float64
 11  Item1                                  10000 non-null  float64
 12  Item2                                  10000 non-null  float64
 13  Item3                                  10000 non-null  float64
 14  Item4                                  10000 non-null  float64
 15  Item5                                  10000 non-null  float64
 16  Item6                                  10000 non-null  float64
 17  Item7                                  10000 non-null  float64
 18  Item8                                  10000 non-null  float64
 19  Area_Suburban                          10000 non-null  float64
 20  Area_Urban                             10000 non-null  float64
 21  Marital_Married                        10000 non-null  float64
 22  Marital_Never Married                  10000 non-null  float64
 23  Marital_Separated                      10000 non-null  float64
 24  Marital_Widowed                        10000 non-null  float64
 25  Gender_Male                            10000 non-null  float64
 26  Gender_Nonbinary                       10000 non-null  float64
 27  Churn_Yes                              10000 non-null  float64
 28  Techie_Yes                             10000 non-null  float64
 29  Contract_One year                      10000 non-null  float64
 30  Contract_Two Year                      10000 non-null  float64
 31  Port_modem_Yes                         10000 non-null  float64
 32  Tablet_Yes                             10000 non-null  float64
 33  InternetService_Fiber Optic            10000 non-null  float64
 34  InternetService_None                   10000 non-null  float64
 35  Phone_Yes                              10000 non-null  float64
 36  Multiple_Yes                           10000 non-null  float64
 37  OnlineSecurity_Yes                     10000 non-null  float64
 38  OnlineBackup_Yes                       10000 non-null  float64
 39  DeviceProtection_Yes                   10000 non-null  float64
 40  TechSupport_Yes                        10000 non-null  float64
 41  StreamingTV_Yes                        10000 non-null  float64
 42  StreamingMovies_Yes                    10000 non-null  float64
 43  PaperlessBilling_Yes                   10000 non-null  float64
 44  PaymentMethod_Credit Card (automatic)  10000 non-null  float64
 45  PaymentMethod_Electronic Check         10000 non-null  float64
 46  PaymentMethod_Mailed Check             10000 non-null  float64
dtypes: float64(47)
memory usage: 3.6 MB
```

In [83]:

```
df4.describe()
```

Out[83]:

	Population	Children	Age	Income	Outage_sec_perweek	
count	10000.000000	10000.0000	10000.000000	10000.000000	10000.000000	10000.
mean	9756.562400	2.0877	53.078400	39806.926771	10.001848	12
std	14432.698671	2.1472	20.698882	28199.916702	2.976019	3.
min	0.000000	0.0000	18.000000	348.670000	0.099747	1.
25%	738.000000	0.0000	35.000000	19224.717500	8.018214	10.
50%	2910.500000	1.0000	53.000000	33170.605000	10.018560	12.
75%	13168.000000	3.0000	71.000000	53246.170000	11.969485	14.
max	111850.000000	10.0000	89.000000	258900.700000	21.207230	23.

8 rows × 47 columns

Save copy of prepared data

In [84]:
```
df4.to_csv('multiple_prepared_churn.csv')
```

Construct Initial Multiple Regression Model

define y and X variables. Some code borrowed from https://www.geeksforgeeks.org/select-all-columns-except-one-given-column-in-a-pandas-dataframe/.

In [85]:
```
y = df4['Tenure']
X = df4.loc[:, df4.columns != 'Tenure']
X = sm.add_constant(X)
```

In [86]:
```
regression = sm.OLS(y,X)
```

In [87]:
```
allmodel = regression.fit()
```

In [88]:
```
print(allmodel.params)
```

```
const                                         -3.843090e+00
Population                                    -7.835527e-08
Children                                      -3.755418e-01
Age                                            3.998674e-02
Income                                         1.563977e-08
Outage_sec_perweek                             2.927557e-04
Email                                         -6.710452e-05
Contacts                                      -5.939200e-04
Yearly_equip_failure                           1.920471e-04
MonthlyCharge                                 -3.521020e-02
Bandwidth_GB_Year                              1.220489e-02
Item1                                          2.318052e-03
Item2                                         -7.001376e-04
Item3                                          1.005443e-03
Item4                                          4.468566e-04
Item5                                         -5.943479e-04
Item6                                         -1.275849e-03
Item7                                         -1.076200e-03
Item8                                          7.958727e-05
Area_Suburban                                 -6.828673e-03
Area_Urban                                    -3.302096e-03
Marital_Married                               -5.666849e-04
Marital_Never Married                         -1.113461e-03
Marital_Separated                              2.638689e-03
Marital_Widowed                                3.450319e-05
Gender_Male                                   -7.923645e-01
Gender_Nonbinary                               2.618276e-01
Churn_Yes                                      1.951267e-03
Techie_Yes                                    -1.103595e-05
Contract_One year                              1.119808e-03
Contract_Two Year                              2.346317e-03
Port_modem_Yes                                 2.686231e-03
Tablet_Yes                                     6.531866e-04
InternetService_Fiber Optic                    5.754226e+00
InternetService_None                           4.600568e+00
Phone_Yes                                      1.692035e-03
Multiple_Yes                                   2.684380e-01
OnlineSecurity_Yes                            -8.312509e-01
OnlineBackup_Yes                              -3.551574e-01
DeviceProtection_Yes                          -5.970609e-01
TechSupport_Yes                                3.850405e-01
StreamingTV_Yes                               -1.298429e+00
StreamingMovies_Yes                           -7.226721e-01
PaperlessBilling_Yes                          -3.665722e-03
PaymentMethod_Credit Card (automatic)          2.058826e-03
PaymentMethod_Electronic Check                 2.865085e-03
PaymentMethod_Mailed Check                     7.316969e-03
dtype: float64
```

In [89]:
```
allmodel.summary()
```

Out[89]:

OLS Regression Results

Dep. Variable:	Tenure	**R-squared:**	1.000
Model:	OLS	**Adj. R-squared:**	1.000
Method:	Least Squares	**F-statistic:**	1.316e+07
Date:	Mon, 24 May 2021	**Prob (F-statistic):**	0.00
Time:	15:37:41	**Log-Likelihood:**	8140.4

No. Observations:	10000	**AIC:**	-1.619e+04
Df Residuals:	9953	**BIC:**	-1.585e+04
Df Model:	46		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-3.8431	0.017	-231.335	0.000	-3.876	-3.811
Population	-7.836e-08	7.46e-08	-1.051	0.293	-2.25e-07	6.78e-08
Children	-0.3755	0.001	-748.384	0.000	-0.377	-0.375
Age	0.0400	5.2e-05	768.285	0.000	0.040	0.040
Income	1.564e-08	3.82e-08	0.410	0.682	-5.92e-08	9.05e-08
Outage_sec_perweek	0.0003	0.000	0.809	0.419	-0.000	0.001
Email	-6.71e-05	0.000	-0.189	0.850	-0.001	0.001
Contacts	-0.0006	0.001	-0.545	0.586	-0.003	0.002
Yearly_equip_failure	0.0002	0.002	0.113	0.910	-0.003	0.004
MonthlyCharge	-0.0352	0.000	-284.931	0.000	-0.035	-0.035
Bandwidth_GB_Year	0.0122	5.99e-07	2.04e+04	0.000	0.012	0.012
Item1	0.0023	0.002	1.503	0.133	-0.001	0.005
Item2	-0.0007	0.001	-0.484	0.628	-0.004	0.002
Item3	0.0010	0.001	0.758	0.448	-0.002	0.004
Item4	0.0004	0.001	0.377	0.706	-0.002	0.003
Item5	-0.0006	0.001	-0.483	0.629	-0.003	0.002
Item6	-0.0013	0.001	-1.007	0.314	-0.004	0.001
Item7	-0.0011	0.001	-0.898	0.369	-0.003	0.001
Item8	7.959e-05	0.001	0.070	0.944	-0.002	0.002
Area_Suburban	-0.0068	0.003	-2.593	0.010	-0.012	-0.002
Area_Urban	-0.0033	0.003	-1.251	0.211	-0.008	0.002
Marital_Married	-0.0006	0.003	-0.166	0.868	-0.007	0.006
Marital_Never Married	-0.0011	0.003	-0.329	0.742	-0.008	0.006
Marital_Separated	0.0026	0.003	0.785	0.432	-0.004	0.009
Marital_Widowed	3.45e-05	0.003	0.010	0.992	-0.007	0.007
Gender_Male	-0.7924	0.002	-362.915	0.000	-0.797	-0.788

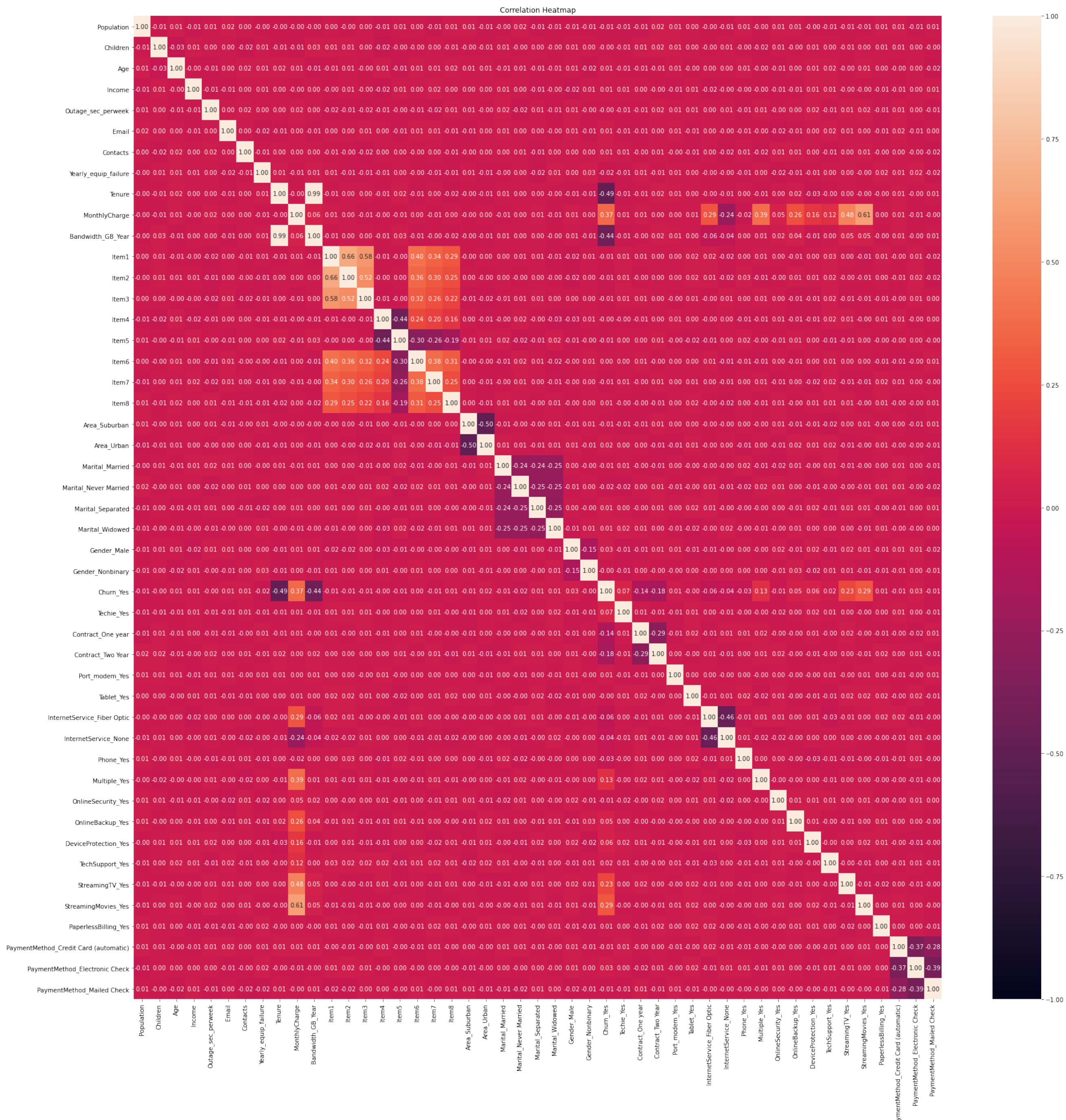
Gender_Nonbinary	0.2618	0.007	36.143	0.000	0.248	0.276
Churn_Yes	0.0020	0.003	0.571	0.568	-0.005	0.009
Techie_Yes	-1.104e-05	0.003	-0.004	0.997	-0.006	0.006
Contract_One year	0.0011	0.003	0.389	0.697	-0.005	0.007
Contract_Two Year	0.0023	0.003	0.855	0.393	-0.003	0.008
Port_modem_Yes	0.0027	0.002	1.248	0.212	-0.002	0.007
Tablet_Yes	0.0007	0.002	0.277	0.781	-0.004	0.005
InternetService_Fiber Optic	5.7542	0.004	1623.665	0.000	5.747	5.761
InternetService_None	4.6006	0.003	1363.478	0.000	4.594	4.607
Phone_Yes	0.0017	0.004	0.457	0.648	-0.006	0.009
Multiple_Yes	0.2684	0.005	59.120	0.000	0.260	0.277
OnlineSecurity_Yes	-0.8313	0.002	-365.801	0.000	-0.836	-0.827
OnlineBackup_Yes	-0.3552	0.004	-101.115	0.000	-0.362	-0.348
DeviceProtection_Yes	-0.5971	0.003	-224.678	0.000	-0.602	-0.592
TechSupport_Yes	0.3850	0.003	142.188	0.000	0.380	0.390
StreamingTV_Yes	-1.2984	0.006	-232.008	0.000	-1.309	-1.287
StreamingMovies_Yes	-0.7227	0.007	-106.938	0.000	-0.736	-0.709
PaperlessBilling_Yes	-0.0037	0.002	-1.675	0.094	-0.008	0.001
PaymentMethod_Credit Card (automatic)	0.0021	0.003	0.628	0.530	-0.004	0.008
PaymentMethod_Electronic Check	0.0029	0.003	0.975	0.329	-0.003	0.009
PaymentMethod_Mailed Check	0.0073	0.003	2.283	0.022	0.001	0.014

Omnibus:	34822.579	**Durbin-Watson:**	2.002
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1633.265
Skew:	-0.034	**Prob(JB):**	0.00
Kurtosis:	1.021	**Cond. No.**	8.31e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.31e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

The r-squared value is 1. The model is a perfect fit and explains all the variance. There are several predictors that have a p-value of more than 0.05. These warrent closer inspection to decide which to remove from the model. The condition number test is well above 30. Create a heatmap to look for High correlation between variables.

In [90]:
```
plt.figure(figsize = (30,30))
heat_map = sn.heatmap(df4.corr(),vmin = -1, vmax = 1, annot = True, fmt = '.2
heat_map.set_title('Correlation Heatmap');
```



In [91]:
```
pred_y = allmodel.predict()
```

In [92]:
```
metrics.mean_absolute_error(y, pred_y)
```

Out[92]: 0.10692196984791037

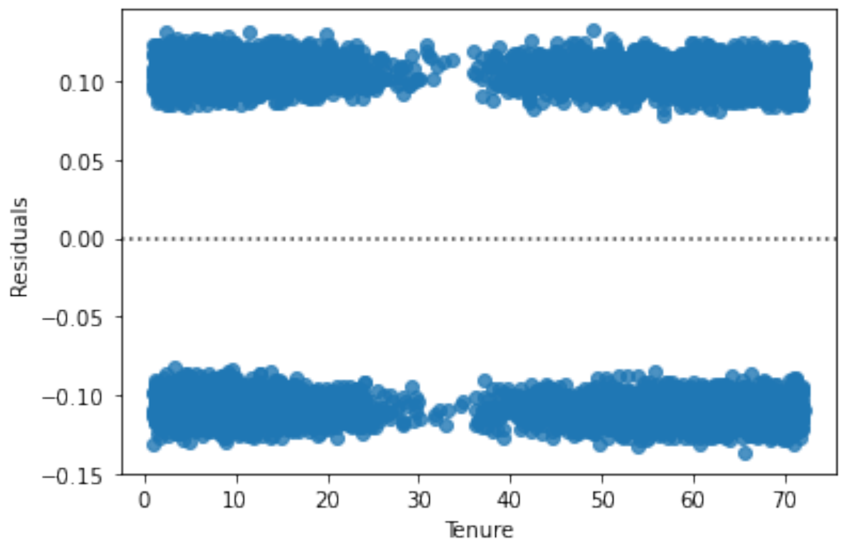
In [93]:
```
metrics.mean_squared_error(y, pred_y)
```

Out[93]: 0.011493740504174298

In [94]:
```
np.sqrt(metrics.mean_squared_error(y, pred_y))
```

Out[94]: 0.10720886392539704

In [95]:
```
residuals = y - pred_y
ax = sn.residplot(x = y, y = residuals)
ax.set(xlabel = 'Tenure', ylabel = 'Residuals');
```



Dimension Reduction

In [96]:
```
from sklearn.linear_model import Lasso
lasso = Lasso(alpha = 0.01, normalize = True)
```

In [97]:
```
lasso.fit(X, y)
```

Out[97]: Lasso(alpha=0.01, normalize=True)

In [98]:
```
lasso.coef_
```

Out[98]:
```
array([ 0.        , -0.        , -0.        ,  0.        , -0.        ,
       -0.        , -0.        , -0.        ,  0.        , -0.01338274,
        0.01146681,  0.        , -0.        ,  0.        ,  0.        ,
        0.        , -0.        ,  0.        , -0.        , -0.        ,
       -0.        ,  0.        ,  0.        , -0.        ,  0.        ,
       -0.        ,  0.        , -1.18126788,  0.        , -0.        ,
       -0.        ,  0.        , -0.        ,  1.1795769 ,  0.        ,
        0.        , -0.        , -0.        , -0.        , -0.        ,
        0.        , -0.        , -0.        ,  0.        ,  0.        ,
        0.        , -0.        ])
```

https://stackoverflow.com/questions/34649969/how-to-find-the-features-names-of-the-coefficients-using-scikit-linear-regressio

In [99]:
```
print(list(zip(lasso.coef_, X)))
```

```
[(0.0, 'const'), (-0.0, 'Population'), (-0.0, 'Children'), (0.0, 'Age'), (-0.0
, 'Income'), (-0.0, 'Outage_sec_perweek'), (-0.0, 'Email'), (-0.0, 'Contacts')
, (0.0, 'Yearly_equip_failure'), (-0.013382742738548973, 'MonthlyCharge'), (0.
011466812203708 66, 'Bandwidth_GB_Year'), (0.0, 'Item1'), (-0.0, 'Item2'), (0.0
, 'Item3'), (0.0, 'Item4'), (0.0, 'Item5'), (-0.0, 'Item6'), (0.0, 'Item7'), (
-0.0, 'Item8'), (-0.0, 'Area_Suburban'), (-0.0, 'Area_Urban'), (0.0, 'Marital_
Married'), (0.0, 'Marital_Never Married'), (-0.0, 'Marital_Separated'), (0.0,
'Marital_Widowed'), (-0.0, 'Gender_Male'), (0.0, 'Gender_Nonbinary'), (-1.1812
67875545039, 'Churn_Yes'), (0.0, 'Techie_Yes'), (-0.0, 'Contract_One year'), (
-0.0, 'Contract_Two Year'), (0.0, 'Port_modem_Yes'), (-0.0, 'Tablet_Yes'), (1.
1795768973975804, 'InternetService_Fiber Optic'), (0.0, 'InternetService_None'
), (0.0, 'Phone_Yes'), (-0.0, 'Multiple_Yes'), (-0.0, 'OnlineSecurity_Yes'), (
-0.0, 'OnlineBackup_Yes'), (-0.0, 'DeviceProtection_Yes'), (0.0, 'TechSupport_
Yes'), (-0.0, 'StreamingTV_Yes'), (-0.0, 'StreamingMovies_Yes'), (0.0, 'Paperl
essBilling_Yes'), (0.0, 'PaymentMethod_Credit Card (automatic)'), (0.0, 'Payme
ntMethod_Electronic Check'), (-0.0, 'PaymentMethod_Mailed Check')]
```

Reduced Model

In [100…
```
df5 = df4[['Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year', 'Churn_Yes', 'Inte
df5
```

Out[100…

	Tenure	MonthlyCharge	Bandwidth_GB_Year	Churn_Yes	InternetService_Fiber Optic
0	6.795513	172.455519	904.536110	0.0	1.0
1	1.156681	242.632554	800.982766	1.0	1.0
2	15.754144	159.947583	2054.706961	0.0	0.0
3	17.087227	119.956840	2164.579412	0.0	0.0
4	1.670972	149.948316	271.493436	1.0	1.0
...	...	...	...	...	...
9995	68.197130	159.979400	6511.252601	0.0	0.0
9996	61.040370	207.481100	5695.951810	0.0	1.0
9997	47.416890	169.974100	4159.305799	0.0	1.0
9998	71.095600	252.624000	6468.456752	0.0	1.0
9999	63.350860	217.484000	5857.586167	0.0	1.0

10000 rows × 5 columns

In [101…
```
y = df5['Tenure']
X = df5.loc[:, df5.columns != 'Tenure']
X = sm.add_constant(X)
```

In [102…
```
regression = sm.OLS(y,X)
reducemodel = regression.fit()
reducemodel.summary()
```

Out[102…

OLS Regression Results

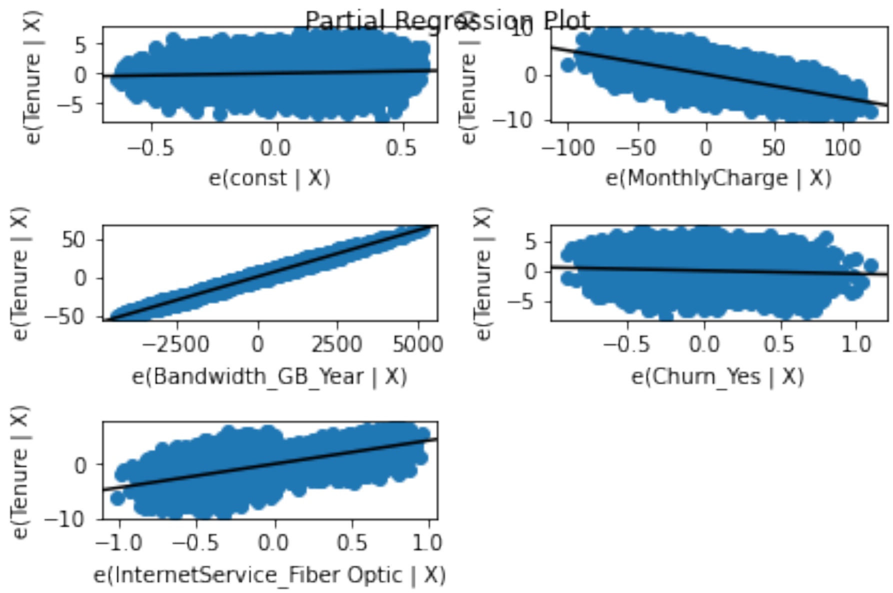
Dep. Variable:	Tenure	**R-squared:**	0.993
Model:	OLS	**Adj. R-squared:**	0.993
Method:	Least Squares	**F-statistic:**	3.799e+05
Date:	Mon, 24 May 2021	**Prob (F-statistic):**	0.00
Time:	15:37:50	**Log-Likelihood:**	-21786.
No. Observations:	10000	**AIC:**	4.358e+04
Df Residuals:	9995	**BIC:**	4.362e+04
Df Model:	4		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	0.6918	0.093	7.406	0.000	0.509	0.875
MonthlyCharge	-0.0515	0.001	-85.638	0.000	-0.053	-0.050
Bandwidth_GB_Year	0.0121	1.16e-05	1042.589	0.000	0.012	0.012
Churn_Yes	-0.5324	0.063	-8.510	0.000	-0.655	-0.410
InternetService_Fiber Optic	4.3373	0.047	92.751	0.000	4.246	4.429

Omnibus:	52.378	**Durbin-Watson:**	1.982
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	51.770
Skew:	0.162	**Prob(JB):**	5.73e-12
Kurtosis:	2.859	**Cond. No.**	1.77e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.77e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [103…
```
fig2 = sm.graphics.plot_partregress_grid(reducemodel)
```



In [104...
```
y_pred = reducemodel.predict()
```

In [105...
```
metrics.mean_absolute_error(y, y_pred)
```

Out[105... 1.7063725726823755

In [106...
```
metrics.mean_squared_error(y, y_pred)
```

Out[106... 4.5691606765843575

In [107...
```
np.sqrt(metrics.mean_squared_error(y, y_pred))
```

Out[107... 2.13755951416197

In [109...
```
residuals = y - y_pred
ax = sn.residplot(x = y, y = residuals)
ax.set(xlabel = 'Tenure', ylabel = 'Residuals');
```