In [129]:
```
#Import libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
from scipy import stats

#Import dataset
df = pd.read_csv("C:/Users/hkeim/OneDrive/Documents/School/D208/churn_clean.csv")
```

In [130]:
```
# Change object type to category codes
df['State'] = df['State'].astype('category')
df['State'] = df['State'].cat.codes
```

In [131]:
```
df['Area'] = df['Area'].astype('category')
df['Area'] = df['Area'].cat.codes
```

In [132]:
```
df['TimeZone'] = df['TimeZone'].astype('category')
df['TimeZone'] = df['TimeZone'].cat.codes
```

In [133]:
```
df['Marital'] = df['Marital'].astype('category')
df['Marital'] = df['Marital'].cat.codes
```

In [134]:
```
df['Gender'] = df['Gender'].astype('category')
df['Gender'] = df['Gender'].cat.codes
```

In [135]:
```
df['Contract'] = df['Contract'].astype('category')
df['Contract'] = df['Contract'].cat.codes
```

In [136]:
```
df['PaymentMethod'] = df['PaymentMethod'].astype('category')
df['PaymentMethod'] = df['PaymentMethod'].cat.codes
```

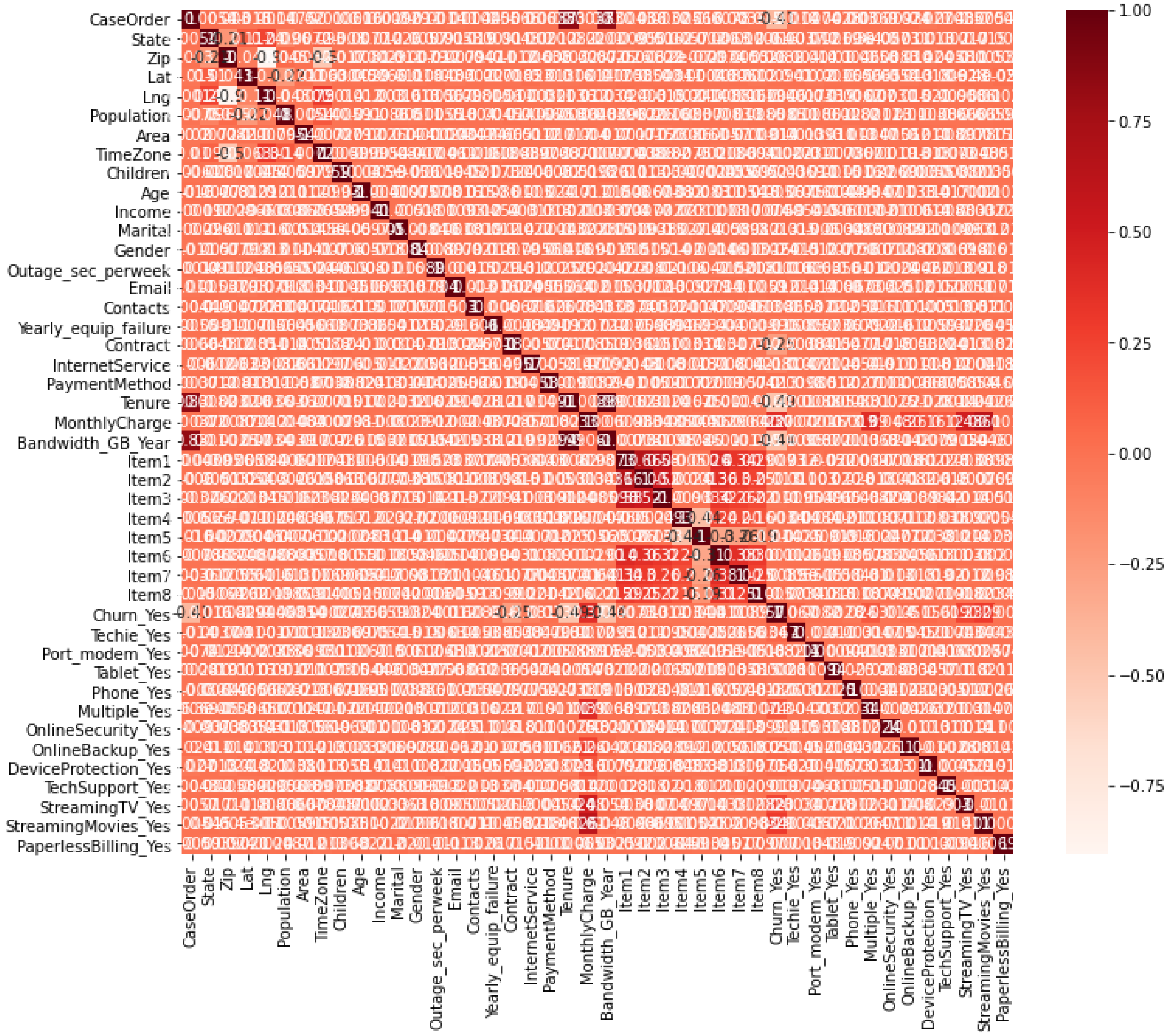
In [137]:
```
df['InternetService'] = df['InternetService'].astype('category')
df['InternetService'] = df['InternetService'].cat.codes
```

In [138]:
```
# Create dummies for bianry objects
df=pd.get_dummies(df, columns=['Churn', 'Techie', 'Port_modem', 'Tablet', 'Phone', 'Multiple'
                               'OnlineSecurity', 'OnlineBackup', 'DeviceProte
```

In [139]:
```
# Drop 'No' dummies
df=df.drop(['Churn_No', 'Techie_No', 'Port_modem_No', 'Tablet_No', 'Phone_No', 'Multiple_No',
           'StreamingTV_No', 'StreamingMovies_No', 'PaperlessBilling_No'], axis=1)
```

In [140]:
```
# Use Pearson Correlation to choose initial model variables
```

```
plt.figure(figsize=(12,10))
cor = df.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()
```



In [141]:

```
# Create dataframe with initial variables
df=df.filter(items=['CaseOrder', 'Tenure', 'Bandwidth_GB_Year', 'Churn_Yes'])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   CaseOrder          10000 non-null  int64  
 1   Tenure             10000 non-null  float64
 2   Bandwidth_GB_Year  10000 non-null  float64
 3   Churn_Yes          10000 non-null  uint8  
dtypes: float64(2), int64(1), uint8(1)
memory usage: 244.3 KB
```

In [142]:

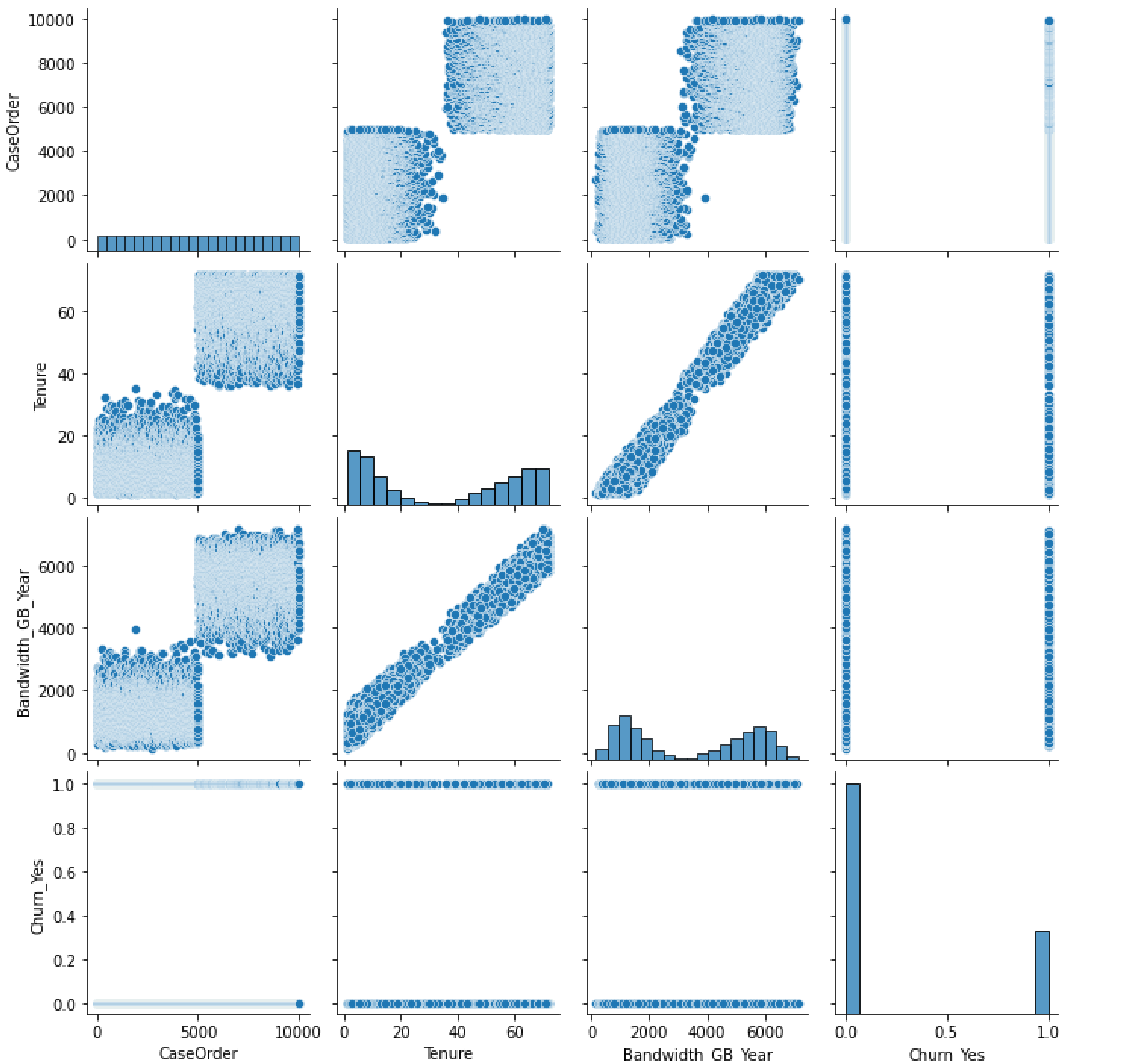
```
# Summary statistics
print(df.describe())
```

```
       CaseOrder       Tenure  Bandwidth_GB_Year     Churn_Yes
count  10000.00000  10000.000000       10000.000000  10000.000000
mean    5000.50000     34.526188        3392.341550      0.265000
std     2886.89568     26.443063        2185.294852      0.441355
min        1.00000      1.000259         155.506715      0.000000
25%     2500.75000      7.917694        1236.470827      0.000000
50%     5000.50000     35.430507        3279.536903      0.000000
75%     7500.25000     61.479795        5586.141370      1.000000
max    10000.00000     71.999280        7158.981530      1.000000
```

In [143]:

```
# Univariate and bivariate visualizations
sns.pairplot(df)
```

Out[143]: <seaborn.axisgrid.PairGrid at 0x1c8cd2a94c0>



In [144]:

```
# Defining the dependent Variable
y=df['Bandwidth_GB_Year']
```

In [145]:

```
# Defining the independent variables
```

```
X=df[['CaseOrder', 'Tenure', 'Churn_Yes']]
```

In [146]:
```
# Initial linear Regression Model
model=linear_model.LinearRegression()
```

In [147]:
```
# Initial model training
model.fit(X,y)
linear_model.LinearRegression(copy_X=True, fit_intercept=True,
                 n_jobs=None, normalize= False)
```

Out[147]:
```
LinearRegression()
```

In [148]:
```
# Initial model coefficients ('CaseOrder', 'Tenure','Churn_Yes')
model.coef_
```

Out[148]:
```
array([8.84048998e-04, 8.39413803e+01, 2.57074782e+02])
```

In [149]:
```
# Initial model Y intercept
model.intercept_
```

Out[149]:
```
421.6201622108065
```

In [150]:
```
# Initial model predict target
y_pred = model.predict(X)
y_pred
```

Out[150]:
```
array([ 992.04578262,  775.79011136, 1744.04741328, ..., 4410.69807842,
       6398.32256269, 5748.21928137])
```

In [151]:
```
# Initial model MSE
mean_squared_error(y,y_pred)
```

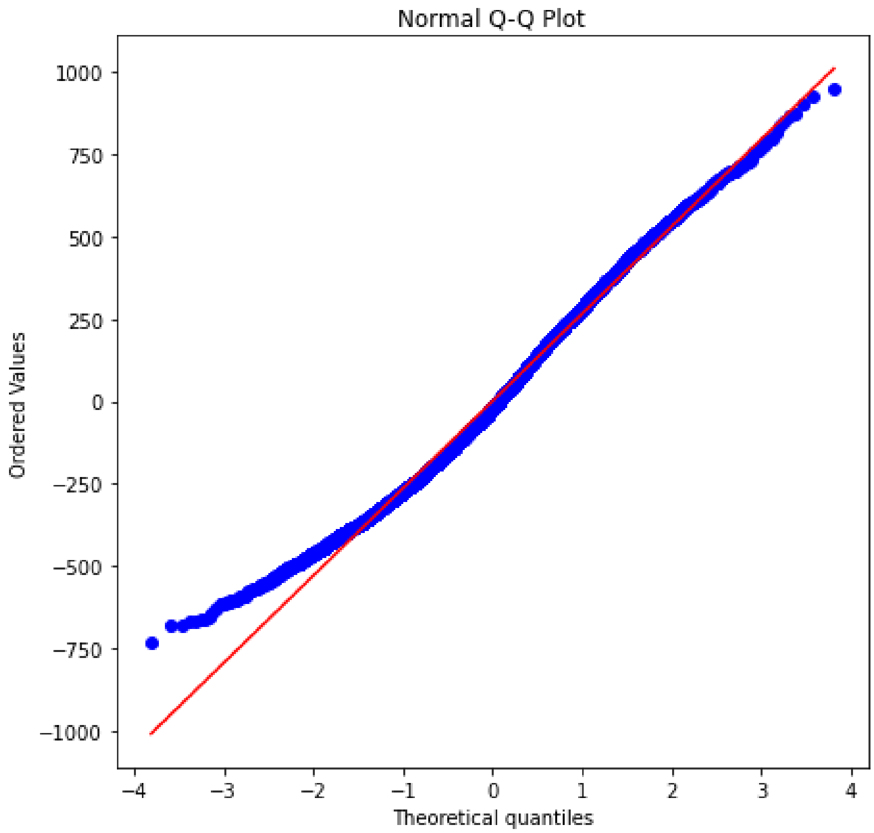
Out[151]:
```
71038.72808569088
```

In [152]:
```
# Initial model coefficient of determination
r2_score(y,y_pred)
```

Out[152]:
```
0.9851228917478336
```

In [153]:
```
# Initial model residual plot
residuals=(y-y_pred)
plt.figure(figsize=(7,7))
stats.probplot(residuals, dist="norm", plot=plt)
plt.title("Normal Q-Q Plot")
```

Out[153]:
```
Text(0.5, 1.0, 'Normal Q-Q Plot')
```

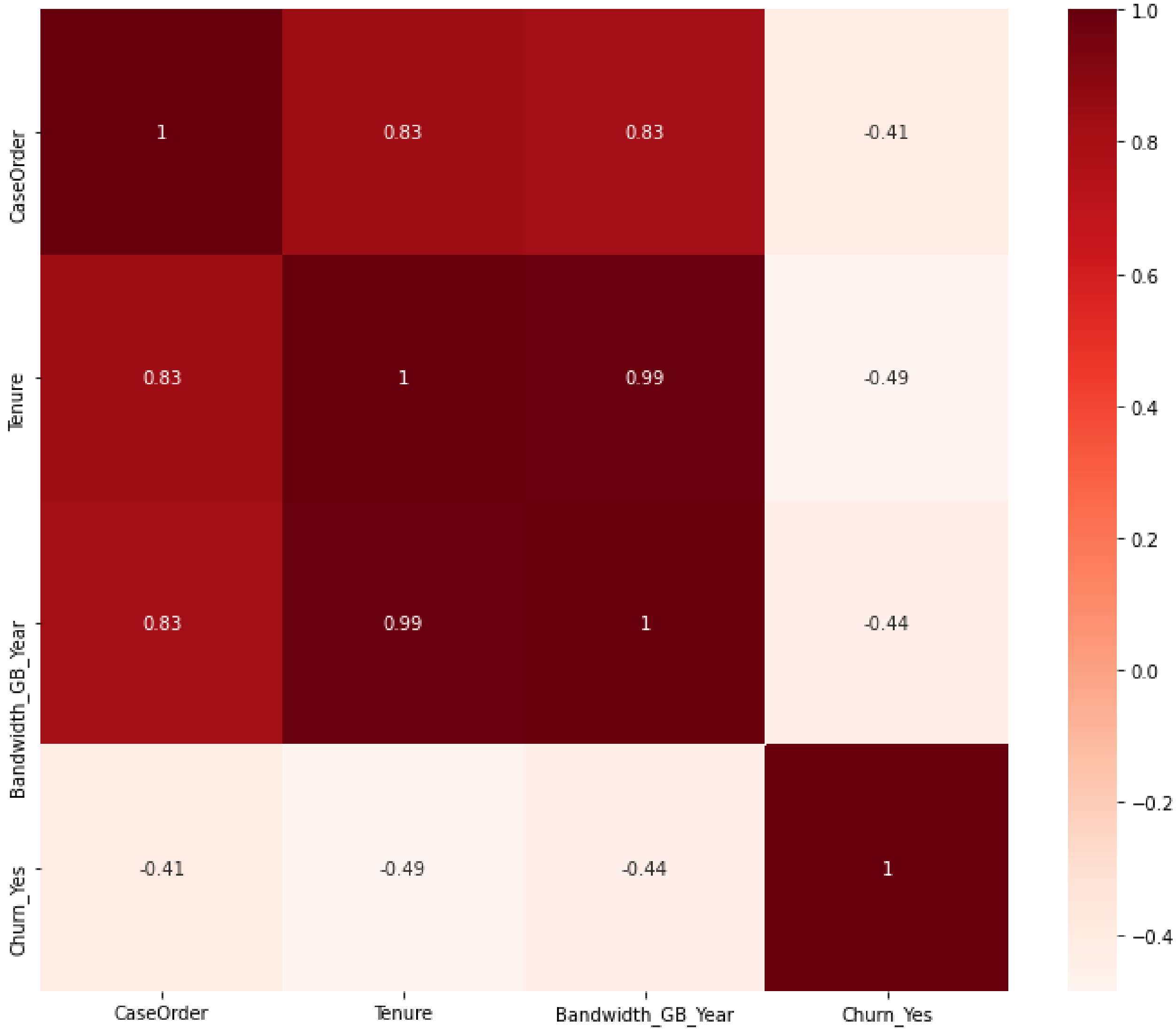


In [154]:
```
# summary of the initial model, p values

X2 = sm.add_constant(X)
est = sm.OLS(y, X2)
est2 = est.fit()
print(est2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Bandwidth_GB_Year   R-squared:                       0.985
Model:                            OLS   Adj. R-squared:                  0.985
Method:                 Least Squares   F-statistic:                 2.206e+05
Date:                Tue, 12 Oct 2021   Prob (F-statistic):               0.00
Time:                        21:12:49   Log-Likelihood:                -70044.
No. Observations:               10000   AIC:                         1.401e+05
Df Residuals:                    9996   BIC:                         1.401e+05
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        421.6202      6.579     64.088      0.000     408.725     434.516
CaseOrder      0.0009      0.002      0.530      0.596      -0.002       0.004
Tenure        83.9414      0.190    441.909      0.000      83.569      84.314
Churn_Yes    257.0748      6.910     37.205      0.000     243.530     270.619
==============================================================================
Omnibus:                      262.092   Durbin-Watson:                   1.986
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              210.306
Skew:                           0.277   Prob(JB):                     2.15e-46
Kurtosis:                       2.556   Cond. No.                     1.84e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.84e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [155]:

```
#Using Pearson Correlation to choose reduced model variables
plt.figure(figsize=(12,10))
cor = df.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()
```



In [156]:

```
# Defining the reduced independent variables
Xr=df[['Churn_Yes', 'Tenure']]
```

In [157]:

```
# Reduced linear Regression Model
modelr=linear_model.LinearRegression()
```

In [158]:

```
# Reduced model training
modelr.fit(Xr,y)
linear_model.LinearRegression(copy_X=True, fit_intercept=True,
                 n_jobs=None, normalize= False)
```

Out[158]: LinearRegression()

In [159]:

```
# Reduced model coefficients ('Churn_Yes', 'Tenure')
modelr.coef_
```

Out[159]:
```
array([257.03592381,  84.02141907])
```

In [160]:
```
# Reduced model Y intercept
modelr.intercept_
```

Out[160]:
```
423.28771144043594
```

In [161]:
```
# Predict bandwidth with reduced model
y_predr = modelr.predict(Xr)
y_predr
```

Out[161]:
```
array([ 994.25635259,  777.50961404, 1746.97325335, ..., 4407.32209736,
       6396.84091342, 5746.11686826])
```

In [162]:
```
# How close to the reduced model are the datapoints
mean_squared_error(y,y_predr)
```

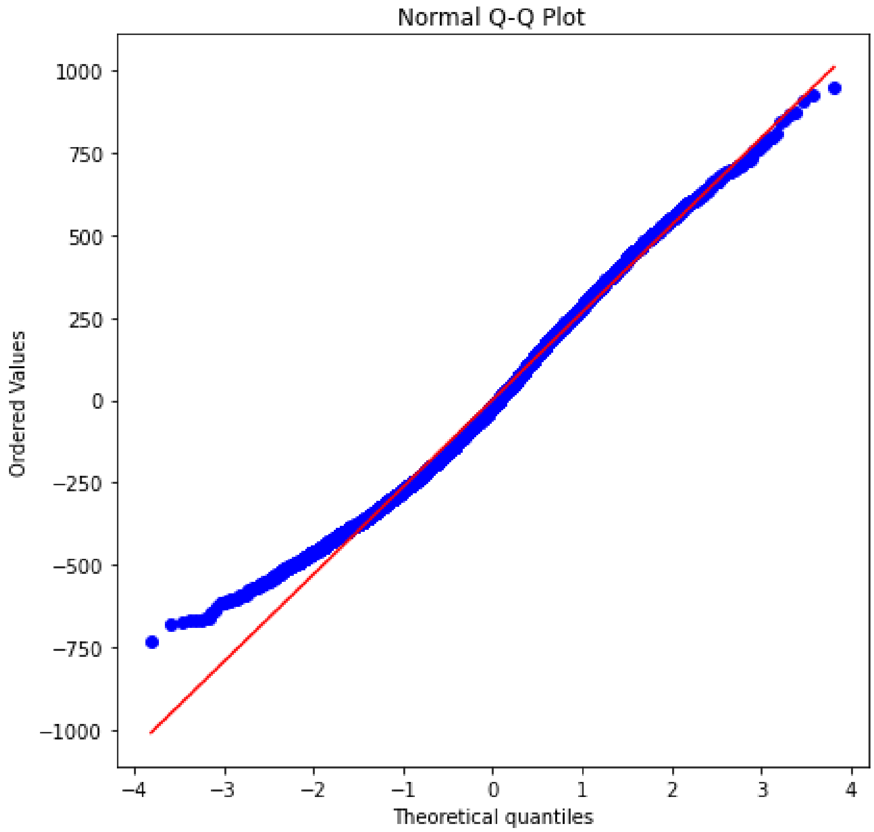
Out[162]:
```
71040.726404036
```

In [163]:
```
# The coefficient of determination for reduced model
r2_score(y,y_predr)
```

Out[163]:
```
0.9851224732550038
```

In [164]:
```
# Reduced model residual plot
residualsr=(y-y_predr)
plt.figure(figsize=(7,7))
stats.probplot(residualsr, dist="norm", plot=plt)
plt.title("Normal Q-Q Plot")
```

Out[164]:
```
Text(0.5, 1.0, 'Normal Q-Q Plot')
```



In [165]:
```
# Reduced model summary Cond. No.

X2 = sm.add_constant(Xr)
est = sm.OLS(y, X2)
est2 = est.fit()
print(est2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       Bandwidth_GB_Year   R-squared:                       0.985
Model:                             OLS   Adj. R-squared:                  0.985
Method:                  Least Squares   F-statistic:                 3.310e+05
Date:                 Tue, 12 Oct 2021   Prob (F-statistic):               0.00
Time:                         21:14:03   Log-Likelihood:                -70044.
No. Observations:                10000   AIC:                         1.401e+05
Df Residuals:                     9997   BIC:                         1.401e+05
Df Model:                            2
Covariance Type:             nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        423.2877      5.778     73.255      0.000     411.961     434.614
Churn_Yes    257.0359      6.909     37.203      0.000     243.493     270.579
Tenure        84.0214      0.115    728.613      0.000      83.795      84.247
==============================================================================
Omnibus:                      261.693   Durbin-Watson:                   1.986
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              210.043
Skew:                           0.277   Prob(JB):                     2.45e-46
Kurtosis:                       2.556   Cond. No.                         134.
==============================================================================

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

In [166]:
```
# Export regression data set
```

```
df.to_csv("C:/Users/hkeim/OneDrive/Documents/School/D208/churn_mlr.csv", index=False, header=
```

In [ ]:

```
```