# A.

## 1.

How can the organization best allocate resources to direct sales, improve service provision, and or client facing services in order to minimize 'Churn' ?

## 2.

The goals of this data analysis are to indentify correlations and relationships in the data set that are actionable and have a positive correlation with 'Churn'.

# B.

## 1.

Observations are independent:

Logistic regression assumes the observations to be independent of each other and independent of repetitive measurement. Any individual should not be measured more than once and neither should it be taken in for the model.

No Multicollinearity: It is essential that the independent variables are not too highly correlated with each other, a condition known as multicollinearity. This can be checked using: Correlation matrices, where correlation coefficients should ideally be below 0.80.

No extreme outliers: Logistic regression assumes that there are no extreme outliers or any external observations that influence the data that goes into the model. Cook's distance is an effective way to rule out the outliers and external observations from a dataset. You can choose to eradicate those from the data or decide to replace them with a mean or median. You can also let the outliers be, but remember to report those in the regression results.

Sample size: The logistic regression assumes that the sample size from which the observations are drawn is large enough to give reliable conclusions for the regression model.

https://www.voxco.com/blog/logistic-regression-assumptions/

## 2.

One benefit of python is that it is an interpreted language. There is no compile time, so it is much quicker for iterative processes such as the backward elimination process when we are reducing the regression model and reducing independent variables.

Another benefit of python language is that it has many libraries and packages that can automate the regression model creation process and simplify it to just a few lines of code. When it is time to compare the reduced model, the python packages can help us quickly compare the models by showing us important regression model metrics such as the p values of coefficients

### 3 .

Logistic regression is an appropriate technique to use for analyzing the research question in part 1 because the question we are answering involves predicting a binary categorical variable 'Churn'. Another reason Logistic regression is an appropriate technique is because part of the question involves identifying correlations between multiple predictor variables and one categorical dependent variable.

## C.

### 1.

My data cleaning goals are as follows:

Identify any duplicate rows and remove them. I will do this by comparing rows by 'CaseOrder'. If there are any duplicates I will drop one of the duplicate rows.

Identify any missing values. I will use the df.isna() function to list columns with missing values. I will impute the values with different techniques depending on the data type and context of each column.

Identify any outliers. I will use z-scores, IQR tests and the describe() method to identify outliers. I will first use the describe() function to get an overview, and if further analysis is needed I can use z-scores and IQR tests to further identify outliers. If a value is clearly an outlier, it can be imputed from other values or the row dropped.

# See cells below for further explanation of each step and annotated code.

In [1]:
```
#import libraries and read in the data from file.
import pandas as pd
from scipy.stats import zscore
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
# Assuming your CSV file is named 'data.csv', adjust the file path as needed
file_path = '/home/dj/skewl/d208/churn_clean.csv'
pd.set_option('display.max_columns', None)
```

```
# Read the data from the CSV file into a DataFrame
df = pd.read_csv(file_path)
#drop index column
df = df.loc[:, ~df.columns.str.contains('Unnamed')]
```

In [2]:
```
# helper functions

#function to plot histogram univariate
def plot_hist(col_name, num_bins, do_rotate=False):
    plt.hist(df[col_name], bins=num_bins)
    plt.xlabel(col_name)
    plt.ylabel('Frequency')
    plt.title(f'Histogram of {col_name}')
    if do_rotate:
        plt.xticks(rotation=90)
    plt.show()

def line_plot(indep):
    # hexbin plot for continuous variables
    plt.hexbin(df[indep], df['MonthlyCharge'], gridsize=10)
    plt.colorbar()
    plt.title('Hexbin Plot')
    plt.xlabel(indep)
    plt.ylabel('Churn')
    plt.show()

def cross_tab(col2):
    # Create a cross-tabulation of the two categorical variables
    cross_tab = pd.crosstab(df['Churn'], df[col2])
    # Plot the heatmap
    sns.heatmap(cross_tab, annot=True, cmap='Blues')
    plt.title(f'Heatmap of Churn and {col2}')
    plt.xlabel(col2)
    plt.ylabel('Churn')
    plt.show()

def box_plot(indep):
    # Box plot for categorical predictor and continuous outcome variable
    df.boxplot(column=indep, by='Churn')
    plt.title('Box Plot',y=.5)
    plt.xlabel("Churn")
    plt.ylabel(indep)
    plt.show()

def stacked_tab(col1):
    # Create a cross-tabulation of the two categorical variables
    cross_tab = pd.crosstab(df['Churn'], df[col1])
    # Plot the stacked bar plot
    cross_tab.plot.bar(stacked=True)
    plt.figure(figsize=(10, 6))
    plt.title(f'Stacked Bar Plot of Chrun and {col1}')
    plt.xlabel('Churn')
    plt.ylabel('Frequency')
    plt.show()
```

## identify duplicate rows by 'CaseOrder'

In [3]:
```
# Find duplicate rows
duplicate_rows = df.duplicated(["CaseOrder"]).sum()

# Print duplicate rows   # found NO duplicate rows here!
print(duplicate_rows)
```

```
0
```

## identify missing values

In [4]:
```
# Identify missing values using isna() method
missing_values = df.isna().sum()
# Print DataFrame with True for missing values and False for non-missing val
print(missing_values)

# no missing values here!
```

```
CaseOrder              0
Customer_id            0
Interaction            0
UID                    0
City                   0
State                  0
County                 0
Zip                    0
Lat                    0
Lng                    0
Population             0
Area                   0
TimeZone               0
Job                    0
Children               0
Age                    0
Income                 0
Marital                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
PaymentMethod          0
Tenure                 0
MonthlyCharge          0
Bandwidth_GB_Year      0
Item1                  0
Item2                  0
Item3                  0
Item4                  0
Item5                  0
Item6                  0
Item7                  0
Item8                  0
dtype: int64
```

## Check for outliers

In [5]:
```
# check for outliers. Doesn't seem to be any outliers.
df.describe()
```

Out[5]:

| | CaseOrder | Zip | Lat | Lng | Population | Child |
|---|---|---|---|---|---|---|
| **count** | 10000.00000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.0( |
| **mean** | 5000.50000 | 49153.319600 | 38.757567 | -90.782536 | 9756.562400 | 2.0: |
| **std** | 2886.89568 | 27532.196108 | 5.437389 | 15.156142 | 14432.698671 | 2.1 |
| **min** | 1.00000 | 601.000000 | 17.966120 | -171.688150 | 0.000000 | 0.0( |
| **25%** | 2500.75000 | 26292.500000 | 35.341828 | -97.082812 | 738.000000 | 0.0( |
| **50%** | 5000.50000 | 48869.500000 | 39.395800 | -87.918800 | 2910.500000 | 1.0( |
| **75%** | 7500.25000 | 71866.500000 | 42.106908 | -80.088745 | 13168.000000 | 3.0( |
| **max** | 10000.00000 | 99929.000000 | 70.640660 | -65.667850 | 111850.000000 | 10.0( |

# 2. Describe dependent and independent variables

In [6]:
```
## dependent variable

df['Churn'].describe()
```

Out[6]:
```
count     10000
unique        2
top          No
freq       7350
Name: Churn, dtype: object
```

In [7]:
```
# independent variable
df['Gender'].describe()
```

Out[7]:
```
count      10000
unique         3
top       Female
freq        5025
Name: Gender, dtype: object
```

In [8]:
```
df['Area'].describe()
```

Out[8]:
```
count        10000
unique           3
top       Suburban
freq          3346
Name: Area, dtype: object
```

In [9]:
```
df['Age'].describe()
```

Out[9]:
```
count    10000.000000
mean        53.078400
std         20.698882
min         18.000000
25%         35.000000
50%         53.000000
75%         71.000000
max         89.000000
Name: Age, dtype: float64
```

In [10]:
```
df['Income'].describe()
```

Out[10]:
```
count     10000.000000
mean      39806.926771
std       28199.916702
min         348.670000
25%       19224.717500
50%       33170.605000
75%       53246.170000
max      258900.700000
Name: Income, dtype: float64
```

In [11]:
```
df['Outage_sec_perweek'].describe()
```

Out[11]:
```
count    10000.000000
mean        10.001848
std          2.976019
min          0.099747
25%          8.018214
50%         10.018560
75%         11.969485
max         21.207230
Name: Outage_sec_perweek, dtype: float64
```

In [12]:
```
df['InternetService'].describe()
```

Out[12]:
```
count           10000
unique              3
top       Fiber Optic
freq             4408
Name: InternetService, dtype: object
```

In [13]:
```
df['Phone'].describe()
```

Out[13]:
```
count     10000
unique        2
top         Yes
freq       9067
Name: Phone, dtype: object
```

In [14]:
```
df['OnlineSecurity'].describe()
```

Out[14]:
```
count     10000
unique        2
top          No
freq       6424
Name: OnlineSecurity, dtype: object
```

In [15]:
```
df['DeviceProtection'].describe()
```

Out[15]:
```
count     10000
unique        2
top          No
freq       5614
Name: DeviceProtection, dtype: object
```

In [16]:
```
df['StreamingMovies'].describe()
```

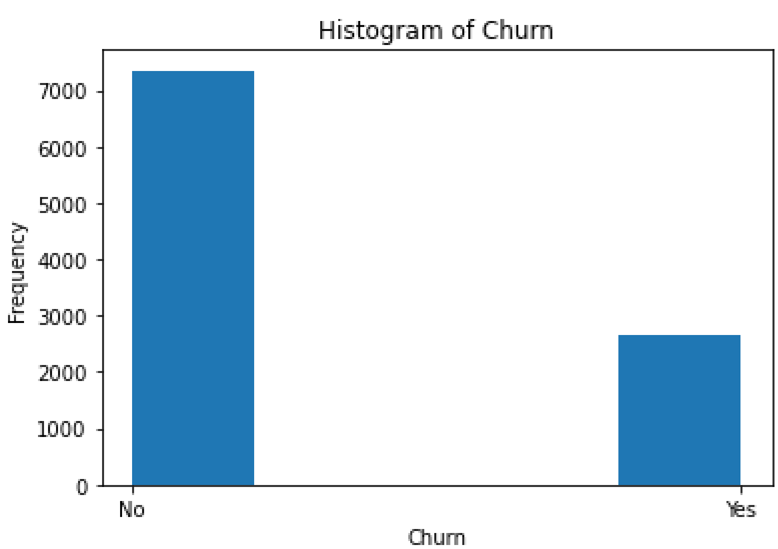
Out[16]:
```
count     10000
unique        2
top          No
freq       5110
Name: StreamingMovies, dtype: object
```

In [17]:
```
df['OnlineBackup'].describe()
```

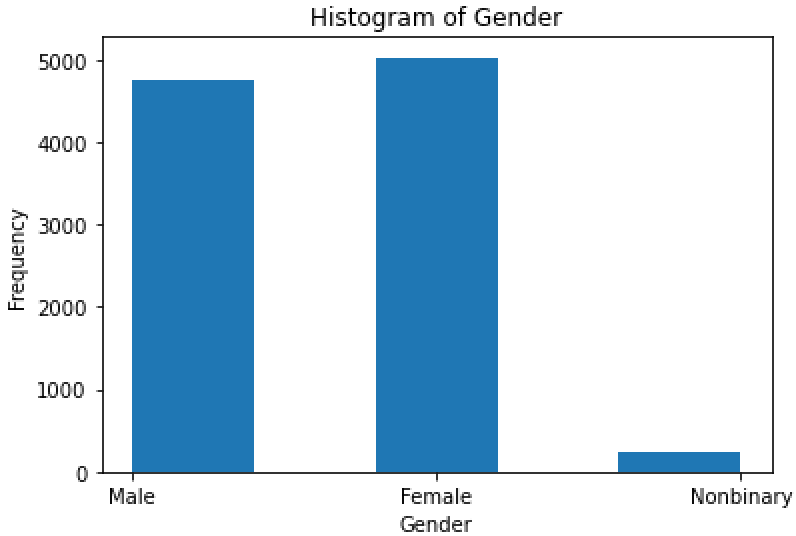
Out[17]:
```
count     10000
unique        2
top          No
freq       5494
Name: OnlineBackup, dtype: object
```

# 3. Generate univariate and bivariate visualizations of the distributions of the dependent and independent variables, including the dependent variable in your bivariate visualizations.

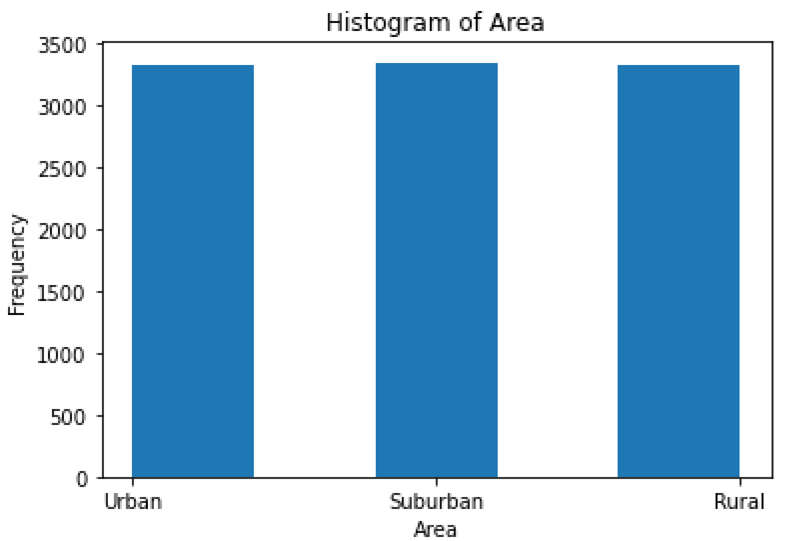
In [18]:
```
plot_hist('Churn',5)
```



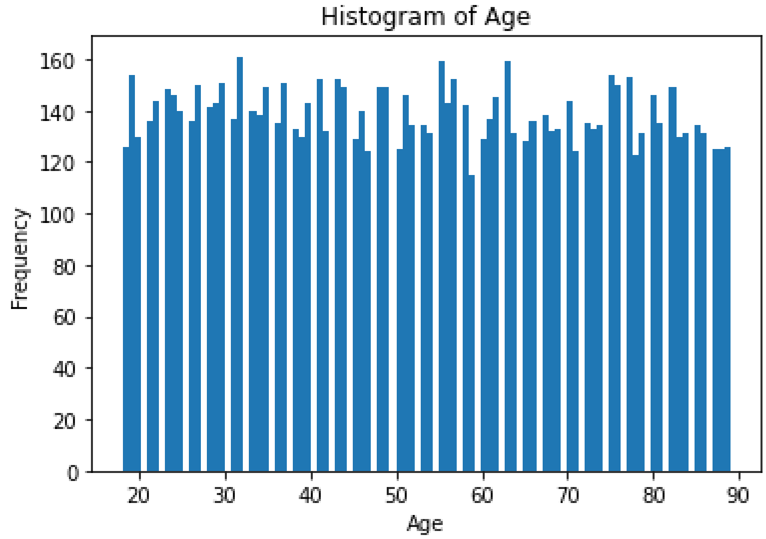
In [19]:
```
plot_hist('Gender',5)
```



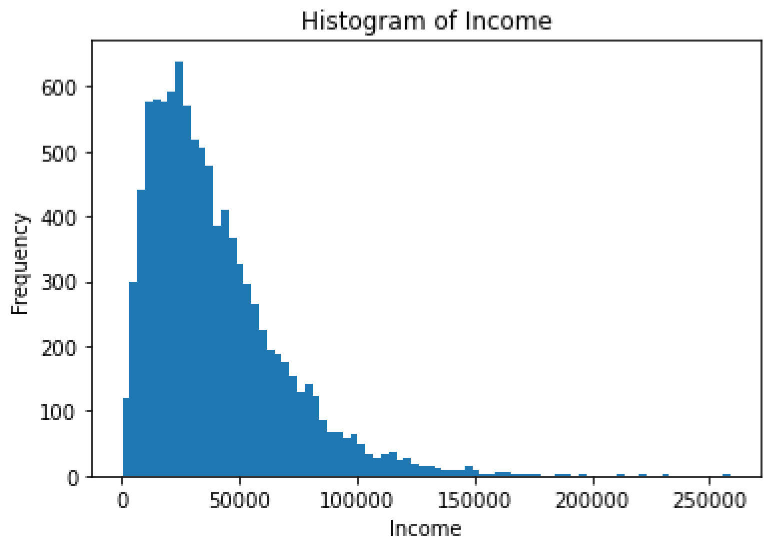

In [20]:
```
plot_hist('Area',5)
```



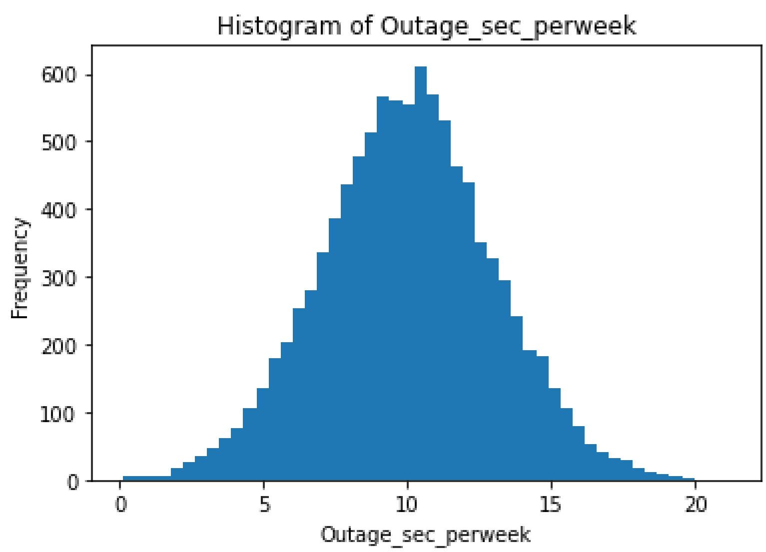

In [21]:
```
plot_hist('Age',100)
```



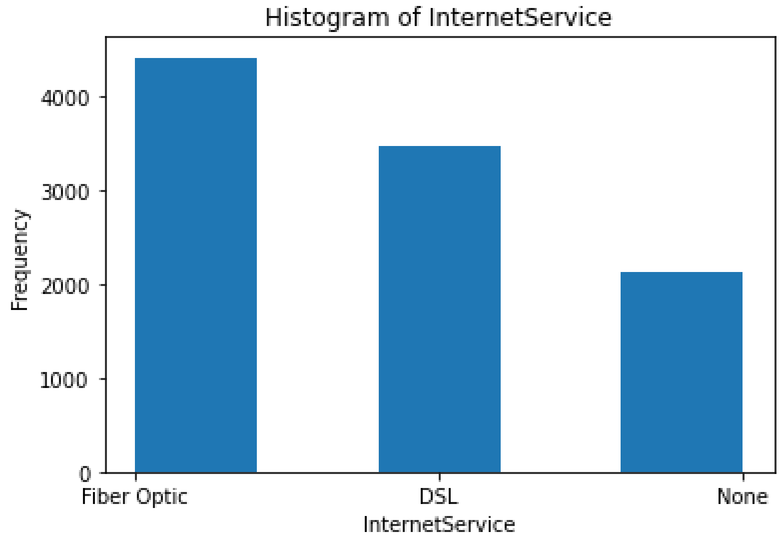

In [22]:
```
plot_hist('Income',80)
```



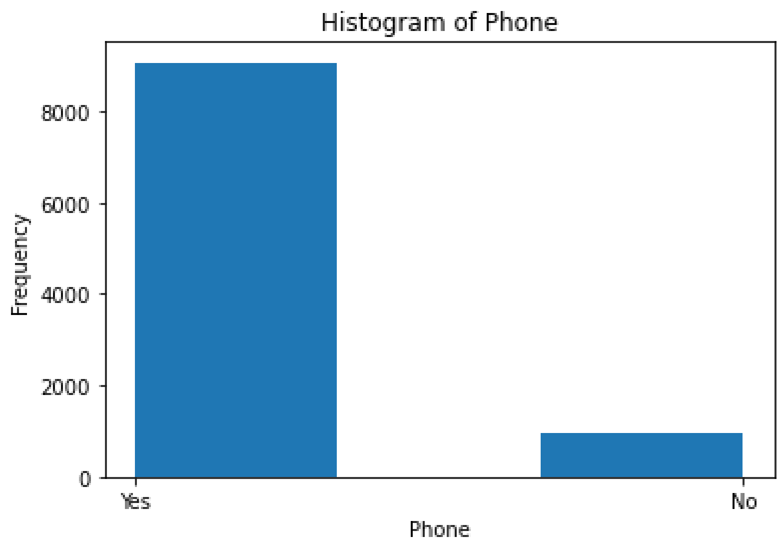

In [23]:
```
plot_hist('Outage_sec_perweek',50)
```



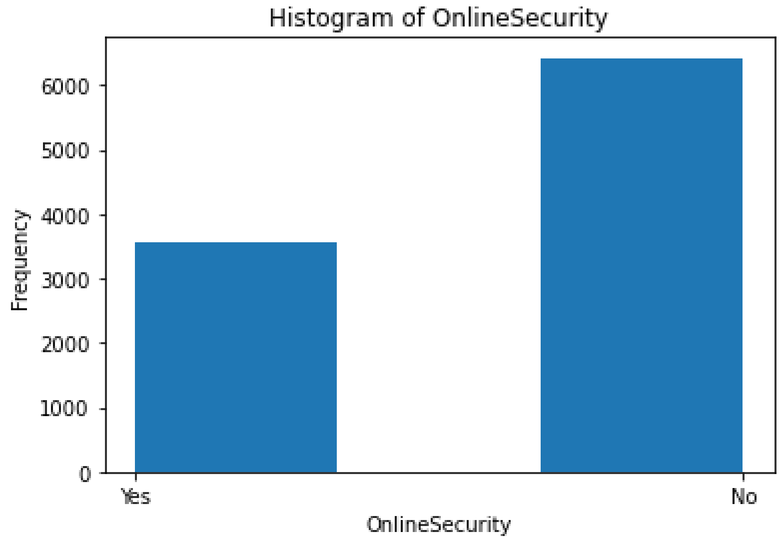
In [24]:
```
plot_hist('InternetService',5)
```



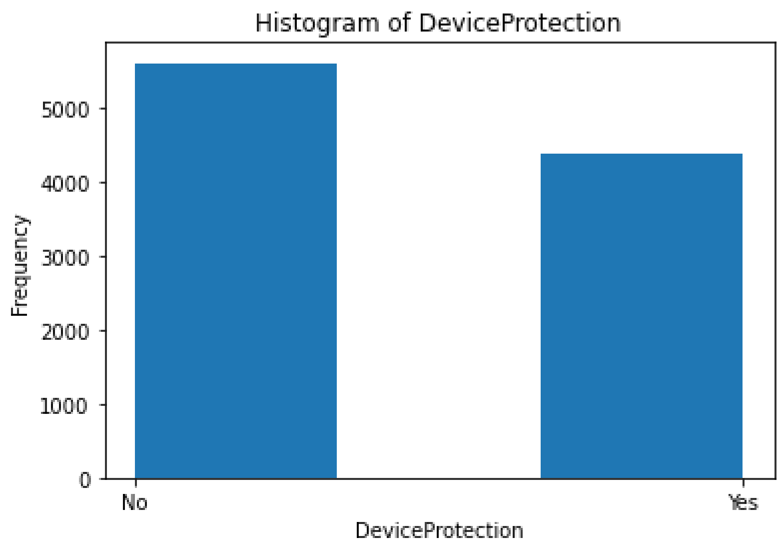

In [25]:
```
plot_hist('Phone',3)
```



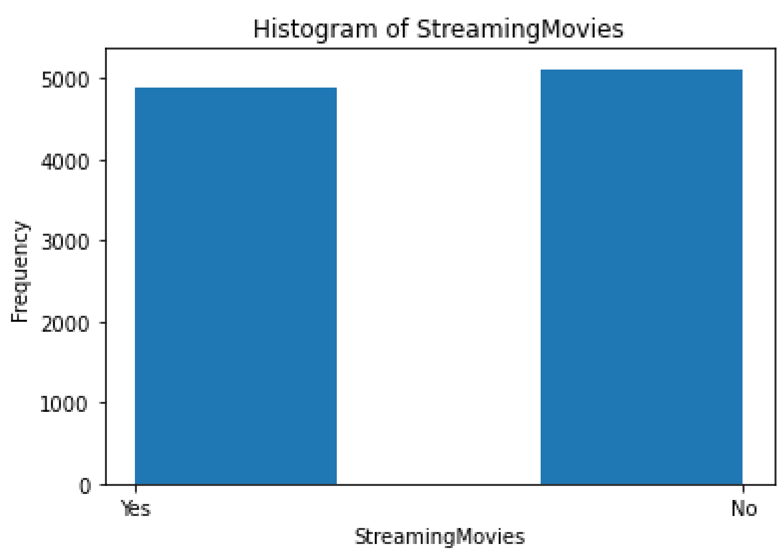

In [26]:
```
plot_hist('OnlineSecurity',3)
```



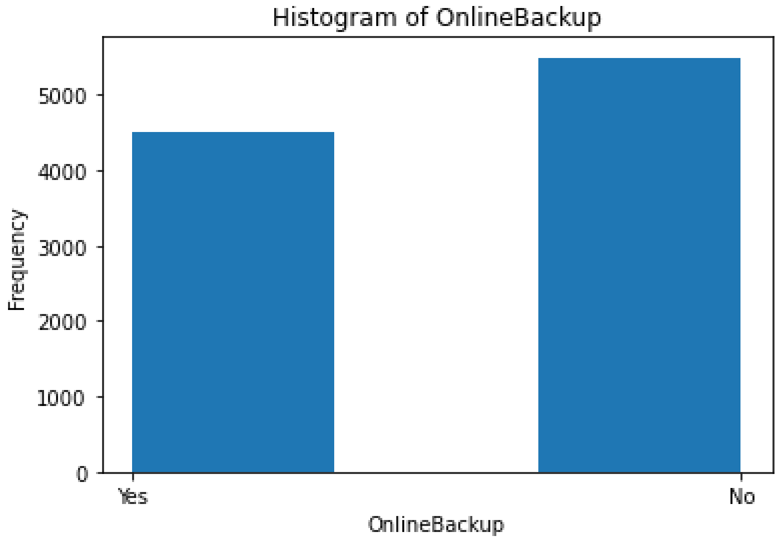

In [27]:
```
plot_hist('DeviceProtection',3)
```




In [28]:
```
plot_hist('StreamingMovies',3)
```

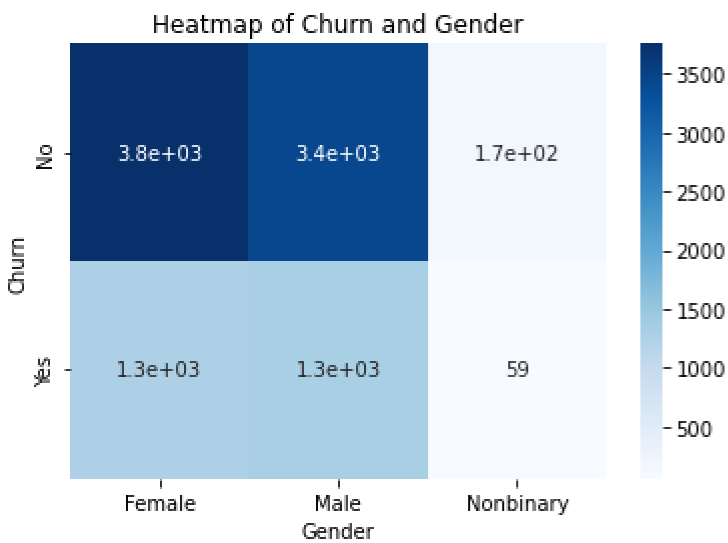


In [29]:
```
plot_hist('OnlineBackup',3)
```

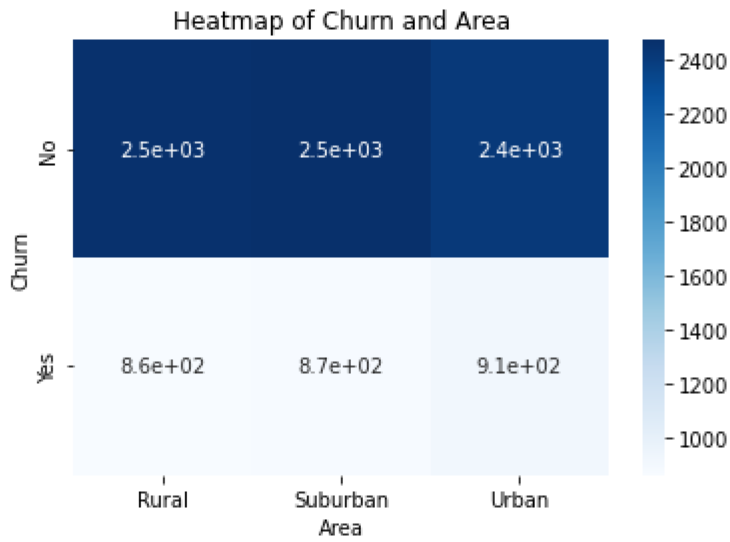



# bivariate - graphing against the dependent variable

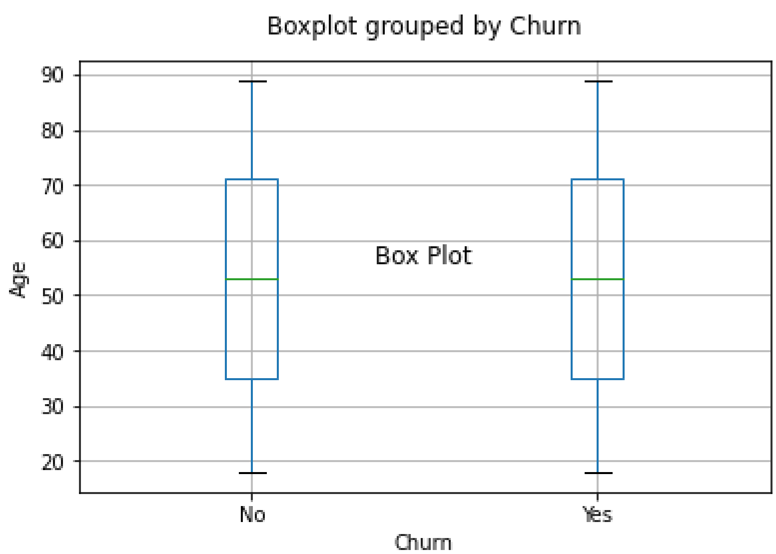
In [30]:
```
cross_tab('Gender')
```



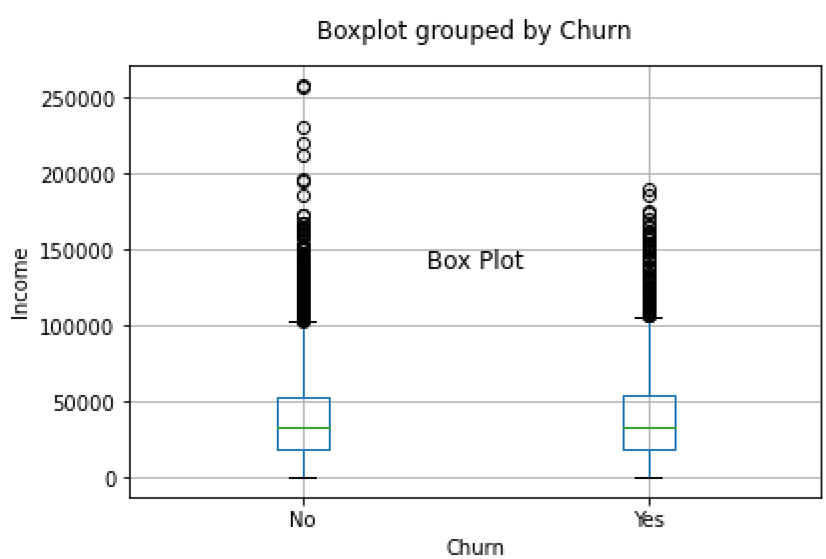

In [31]:
```
cross_tab('Area')
```



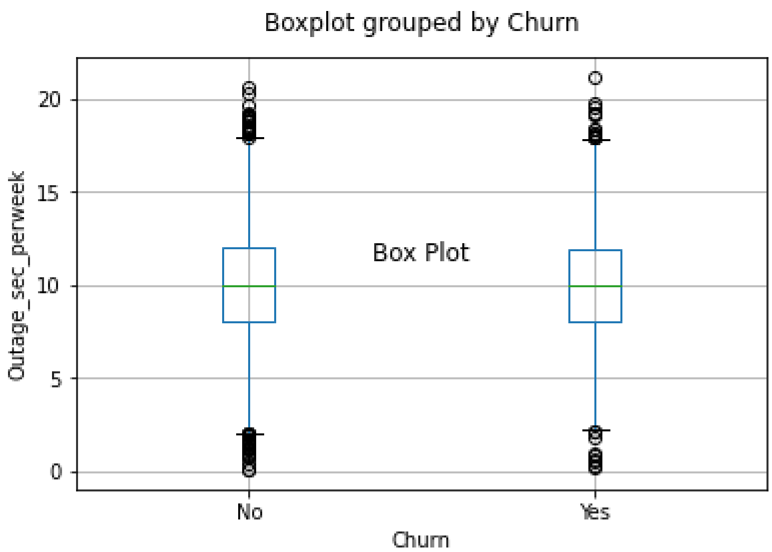

In [32]:
```
box_plot('Age')
```



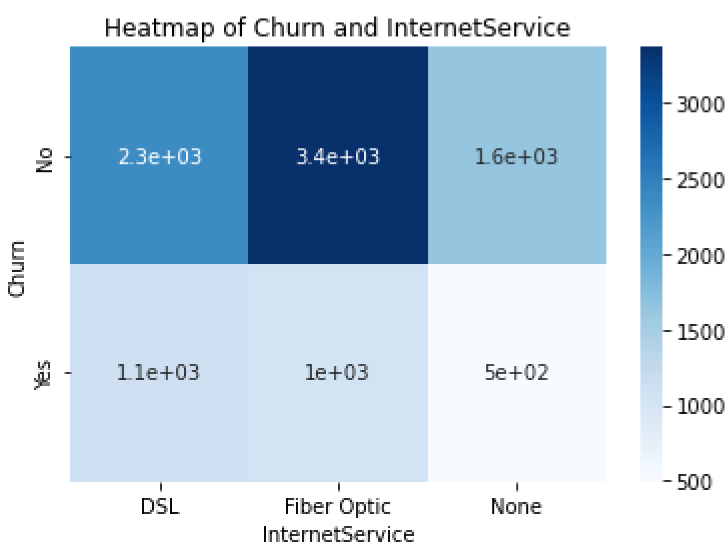

In [33]:
```
box_plot('Income')
```



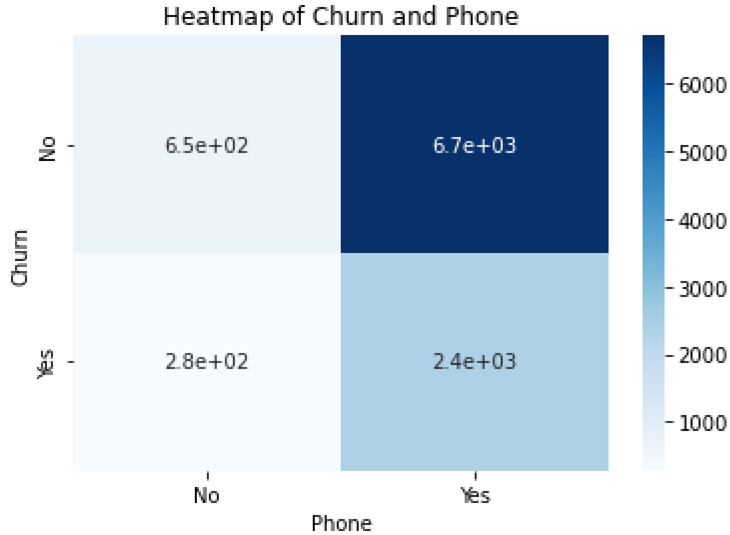
In [34]:
```
box_plot('Outage_sec_perweek')
```



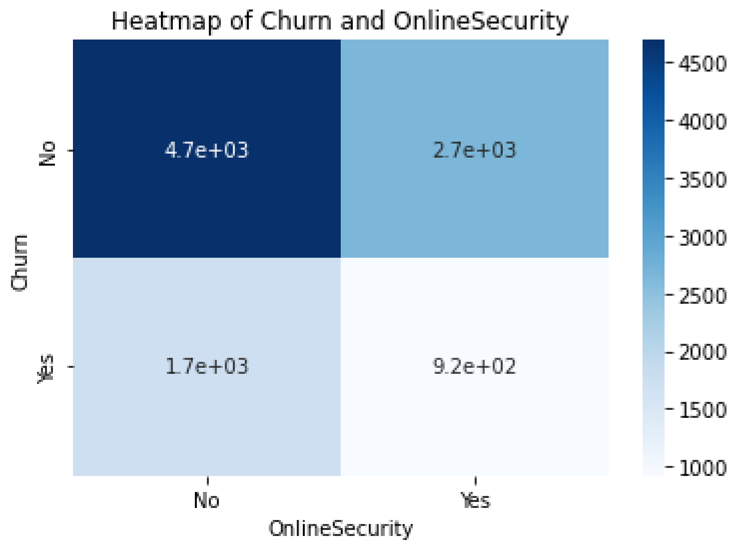

In [35]:
```
cross_tab('InternetService')
```



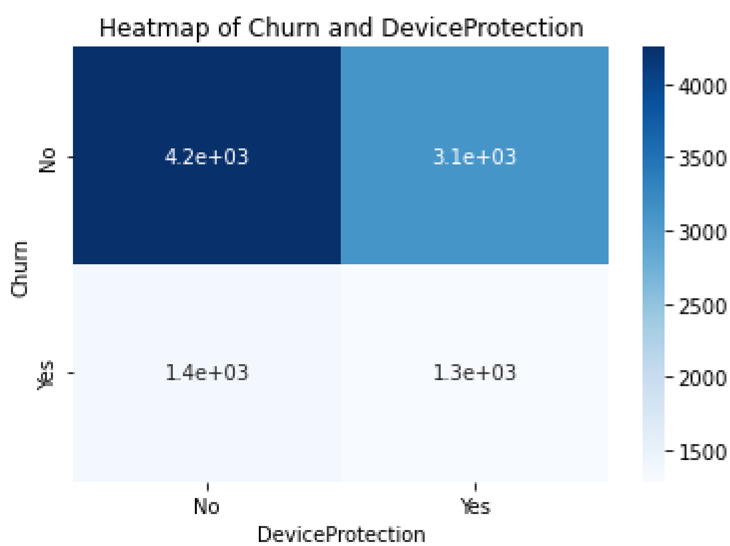

In [36]:
```
cross_tab('Phone')
```



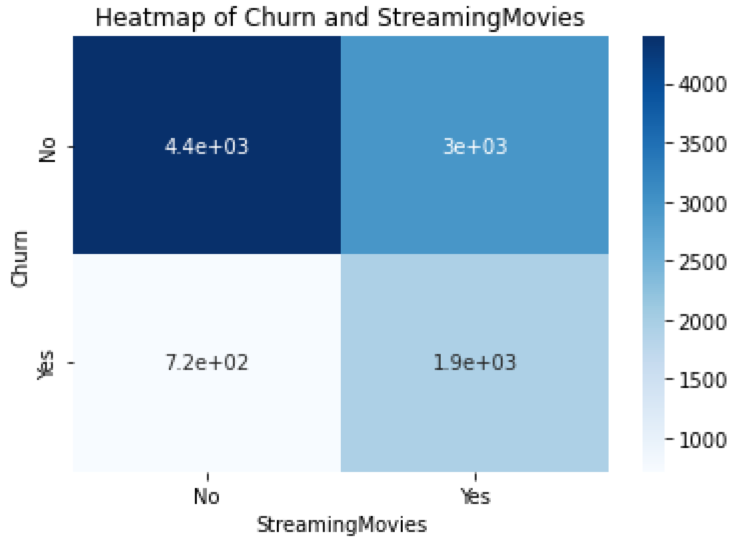

In [37]:
```
cross_tab('OnlineSecurity')
```



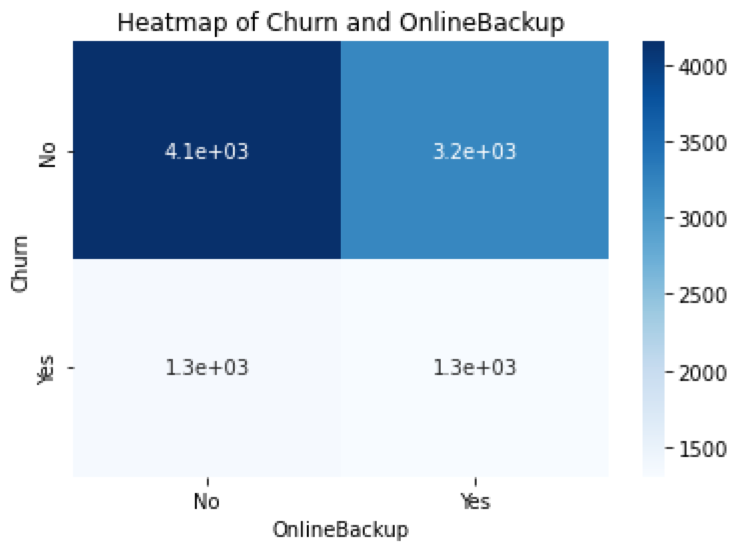

In [38]:
```
cross_tab('DeviceProtection')
```



In [39]:
```
cross_tab('StreamingMovies')
```




In [40]:
```
cross_tab('OnlineBackup')
```




# 4)

My goals for data transformation are to one-hot encode the categorical variables and then normalize all values. The dependent variable 'churn' will be mapped to binary values as well. After this we will split the data into training and test sets.

In [41]:
```
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
#split continuous and categorical variables into separate dataframes
dfcon = df[['Age','Income','Outage_sec_perweek']]
dfcat = df[['Gender','Area','InternetService','Phone','OnlineSecurity','Devi
#one-hot encode categorical data and drop first level of each
dfcat_encoded = pd.get_dummies(dfcat,drop_first=True)
#concatenate the columns
```

```
data = pd.concat([dfcon, dfcat_encoded], axis=1)
#normalize the data

scaler = MinMaxScaler()
df_normalized = pd.DataFrame(scaler.fit_transform(data), columns=data.columr
#write the prepared data to .csv file
df_normalized.to_csv('prepared-data2.csv', index=False)
# Convert categorical dependent variable to binary (0/1)
churn_binary = df['Churn'].map({'No': 0, 'Yes': 1})
independent_vars = sm.add_constant(df_normalized)
x_train, x_test, y_train, y_test = train_test_split(independent_vars, churn_
```

# D. Compare an initial and a reduced linear regression model

## 1. Construct an initial multiple linear regression model from all independent variables that were identified in part C2.

In [42]:
```
#Initial Model
model = sm.Logit(y_train,  x_train).fit()
print(model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.526410
         Iterations 6
                           Logit Regression Results
============================================================================
==
Dep. Variable:                  Churn   No. Observations:                 80
00
Model:                          Logit   Df Residuals:                     79
85
Method:                           MLE   Df Model:
14
Date:                Fri, 12 Apr 2024   Pseudo R-squ.:                 0.086
77
Time:                        05:42:27   Log-Likelihood:                -421
1.3
converged:                       True   LL-Null:                       -461
1.4
Covariance Type:            nonrobust   LLR p-value:                9.510e-1
62
============================================================================
===================
                                 coef    std err          z      P>|z|
[0.025      0.975]
----------------------------------------------------------------------------
-------------------
const                         -1.5842      0.158    -10.050      0.000
-1.893      -1.275
Age                           -0.0241      0.092     -0.263      0.793
-0.204       0.156
Income                         0.2137      0.241      0.886      0.376
-0.259       0.686
Outage_sec_perweek            -0.0862      0.189     -0.455      0.649
-0.457       0.285
Gender_Male                    0.1173      0.054      2.167      0.030
0.011       0.223
Gender_Nonbinary              -0.0104      0.186     -0.056      0.955
-0.375       0.354
Area_Suburban                  0.0115      0.065      0.176      0.861
-0.117       0.140
Area_Urban                     0.0260      0.066      0.396      0.692
-0.103       0.155
InternetService_Fiber Optic   -0.4363      0.060     -7.275      0.000
-0.554      -0.319
InternetService_None          -0.4694      0.074     -6.364      0.000
-0.614      -0.325
Phone_Yes                     -0.2390      0.088     -2.703      0.007
-0.412      -0.066
OnlineSecurity_Yes            -0.0422      0.056     -0.754      0.451
-0.152       0.067
DeviceProtection_Yes           0.2359      0.054      4.394      0.000
0.131       0.341
StreamingMovies_Yes            1.4019      0.056     24.895      0.000
1.292       1.512
OnlineBackup_Yes               0.2425      0.054      4.519      0.000
0.137       0.348
```

```
============================================================================
===================
```

# 2. Justify a statistically based feature selection procedure or a model evaluation metric to reduce the initial model in a way that aligns with the research question.

I have chosen to use backward elimination of predictor variables as my feature selection procedure. This is so I can iteratively choose which predictor variables I want to keep based on p values. This is an effective way to reduce the model because I may choose to keep some predictor variables that may not necessarily meet standard thresholds of $p < .05$. This will enable me to more precisely answer the research question by identifying the effect of these predictor variables on the outcome variable even though they may not meet the $p > .05$ criteria. So even though the predictors may have a slightly larger p value we can still answer questions about how a variable correlates to 'Churn'. We not only want to predict future values of 'Churn' but also know how a given predictor variable will correlate with things like the magnitude and sign of the coefficient so it may be wise to include them in the model. Also a predictor variable may have a good p value but won't be practically significant. With this feature selection method I have more control to actually get meaningful information about what the correlations are to 'Churn'.

I have chosen to use the log likelihood for a model evaluation metric because it measures the 'goodness of fit' of the model, as well as the likelihood that the model will predict the correct outcome

# 3. Provide a reduced linear regression model that follows the feature selection or model evaluation process in part D2, including a screenshot of the output for each model.

In [43]:
```python
#original model

model = sm.Logit(y_train, x_train).fit()
print(model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.526410
         Iterations 6
                           Logit Regression Results
============================================================================
==
Dep. Variable:                  Churn   No. Observations:                 80
00
Model:                          Logit   Df Residuals:                     79
85
Method:                           MLE   Df Model:
14
Date:                Fri, 12 Apr 2024   Pseudo R-squ.:                 0.086
77
Time:                        05:42:27   Log-Likelihood:                -421
1.3
converged:                       True   LL-Null:                       -461
1.4
Covariance Type:            nonrobust   LLR p-value:                9.510e-1
62
============================================================================
===================
                                 coef    std err          z      P>|z|
[0.025      0.975]
----------------------------------------------------------------------------
-------------------
const                         -1.5842      0.158    -10.050      0.000
-1.893      -1.275
Age                           -0.0241      0.092     -0.263      0.793
-0.204       0.156
Income                         0.2137      0.241      0.886      0.376
-0.259       0.686
Outage_sec_perweek            -0.0862      0.189     -0.455      0.649
-0.457       0.285
Gender_Male                    0.1173      0.054      2.167      0.030
0.011       0.223
Gender_Nonbinary              -0.0104      0.186     -0.056      0.955
-0.375       0.354
Area_Suburban                  0.0115      0.065      0.176      0.861
-0.117       0.140
Area_Urban                     0.0260      0.066      0.396      0.692
-0.103       0.155
InternetService_Fiber Optic   -0.4363      0.060     -7.275      0.000
-0.554      -0.319
InternetService_None          -0.4694      0.074     -6.364      0.000
-0.614      -0.325
Phone_Yes                     -0.2390      0.088     -2.703      0.007
-0.412      -0.066
OnlineSecurity_Yes            -0.0422      0.056     -0.754      0.451
-0.152       0.067
DeviceProtection_Yes           0.2359      0.054      4.394      0.000
0.131       0.341
StreamingMovies_Yes            1.4019      0.056     24.895      0.000
1.292       1.512
OnlineBackup_Yes               0.2425      0.054      4.519      0.000
0.137       0.348
```

=============================================================================
===================

# Reduced model

In [44]:
```
#Reduced model
df_normalized = pd.DataFrame(scaler.fit_transform(data), columns=data.columr
del df_normalized['Area_Urban']
del df_normalized['Area_Suburban']
del df_normalized['Age']
del df_normalized['Outage_sec_perweek']
del df_normalized['Gender_Nonbinary']
del df_normalized['Income']
#del df_normalized['InternetService_None']
independent_vars = sm.add_constant(df_normalized)
x_train, x_test, y_train, y_test = train_test_split(independent_vars, churn_
model = sm.Logit(y_train, x_train).fit()
print(model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.526486
         Iterations 6
                           Logit Regression Results
============================================================================
==
Dep. Variable:                  Churn   No. Observations:                 80
00
Model:                          Logit   Df Residuals:                     79
91
Method:                           MLE   Df Model:
8
Date:                Fri, 12 Apr 2024   Pseudo R-squ.:                 0.086
64
Time:                        05:42:27   Log-Likelihood:                -421
1.9
converged:                       True   LL-Null:                       -461
1.4
Covariance Type:            nonrobust   LLR p-value:                3.247e-1
67
============================================================================
===================
                                 coef    std err          z      P>|z|
[0.025      0.975]
----------------------------------------------------------------------------
-------------------
const                         -1.5888      0.109    -14.643      0.000
-1.801      -1.376
Gender_Male                    0.1164      0.053      2.177      0.030
0.012       0.221
InternetService_Fiber Optic   -0.4375      0.060     -7.298      0.000
-0.555      -0.320
InternetService_None          -0.4695      0.074     -6.368      0.000
-0.614      -0.325
Phone_Yes                     -0.2398      0.088     -2.713      0.007
-0.413      -0.067
OnlineSecurity_Yes            -0.0422      0.056     -0.755      0.450
-0.152       0.067
DeviceProtection_Yes           0.2361      0.054      4.401      0.000
0.131       0.341
StreamingMovies_Yes            1.4012      0.056     24.891      0.000
1.291       1.512
OnlineBackup_Yes               0.2411      0.054      4.497      0.000
0.136       0.346
============================================================================
===================
```

# E.

## 1.Explain your data analysis process by comparing the initial logistic regression model and reduced logistic regression model

My model evaluation metrics are log-likelihood and pusedo r squared. Since I had predictor variables that had large coefficients the psuedo R squared value was about the same in both models. This is because the predictor variables with the largest coefficients and smallest P values were not removed.

Original log-likelihood = -4211.3

Reduced model log-likelihood = -4211.9

Original psuedo R squared = .08667

Reduced model psuedo R squared = .08664

The reduced model is just slightly worse in terms of log-likelihood and psuedo R squared metric. The difference is very small and the model has much fewer independent variables, so this may be a worthwhile trade off.

## 2. Provide the output and all calculations of the analysis you performed, including the following elements for your reduced logistic regression model

In [45]:
```
#calculations to reduce original model
df_normalized = pd.DataFrame(scaler.fit_transform(data), columns=data.columr
del df_normalized['Area_Urban']
del df_normalized['Area_Suburban']
del df_normalized['Age']
del df_normalized['Outage_sec_perweek']
del df_normalized['Gender_Male']
del df_normalized['Gender_Nonbinary']
del df_normalized['Income']
del df_normalized['Phone_Yes']
independent_vars = sm.add_constant(df_normalized)
x_train, x_test, y_train, y_test = train_test_split(independent_vars, churn_
model = sm.Logit(y_train, x_train).fit()
print(model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.527236
         Iterations 6
                           Logit Regression Results
=============================================================================
==
Dep. Variable:                  Churn   No. Observations:                  80
00
Model:                          Logit   Df Residuals:                      79
93
Method:                           MLE   Df Model:
6
Date:                Fri, 12 Apr 2024   Pseudo R-squ.:                  0.085
34
Time:                        05:42:27   Log-Likelihood:                 -421
7.9
converged:                       True   LL-Null:                        -461
1.4
Covariance Type:            nonrobust   LLR p-value:                 9.465e-1
67
=============================================================================
===================
                                  coef    std err          z      P>|z|
[0.025      0.975]
-----------------------------------------------------------------------------
-------------------
const                          -1.7528      0.068    -25.676      0.000
-1.887      -1.619
InternetService_Fiber Optic    -0.4358      0.060     -7.277      0.000
-0.553      -0.318
InternetService_None           -0.4681      0.074     -6.354      0.000
-0.612      -0.324
OnlineSecurity_Yes             -0.0393      0.056     -0.703      0.482
-0.149       0.070
DeviceProtection_Yes            0.2418      0.054      4.513      0.000
0.137       0.347
StreamingMovies_Yes             1.4010      0.056     24.909      0.000
1.291       1.511
OnlineBackup_Yes                0.2402      0.054      4.486      0.000
0.135       0.345
=============================================================================
===================
```

# confusion matrix and accuracy calculation

In [46]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
#get predictions
y_pred = model.predict(x_test)

# Compute confusion matrix
y_pred = (y_pred >= .5).astype(int)
conf_matrix = confusion_matrix(y_test, y_pred)
```

```
print("Confusion Matrix:")
print(conf_matrix)
print('======================')
print(f"accuracy=  {accuracy_score(y_test, y_pred, normalize=True)}")
```

```
Confusion Matrix:
[[1422   34]
 [ 506   38]]
======================
accuracy=  0.73
```

# 3. code will be submitted with assignment.

# F.

## 1. Discuss the results of your data analysis

### regression equation :

log(p/1-p) = -1.7528 + (-0.4358)(X) + (-0.4681)(X) + (-0.0393)(x) + 0.2418(x) + 1.4010(X) + 0.2402(X)

### Interpretation of coefficients:

```
The coefficient itself is the magnitude which represents the
strength of the relationship.

 The sign tells us if the relationship is negative or
positive to the log odds of the dependent variable.

 All these coefficients have p values < .05 so they are
statistically significant.


                                 coef    std err          z
P>|z|      [0.025      0.975]
-------------------------------------------------------------
----------------------------------
const                         -1.7528      0.068    -25.676
0.000      -1.887      -1.619
InternetService_Fiber Optic   -0.4358      0.060     -7.277
0.000      -0.553      -0.318
InternetService_None          -0.4681      0.074     -6.354
0.000      -0.612      -0.324
OnlineSecurity_Yes            -0.0393      0.056     -0.703
0.482      -0.149       0.070
DeviceProtection_Yes           0.2418      0.054      4.513
0.000       0.137       0.347
StreamingMovies_Yes            1.4010      0.056     24.909
0.000       1.291       1.511
```

```
OnlineBackup_Yes               0.2402      0.054      4.486
0.000       0.135       0.345
```

All other predictors must be constant for these rules to work.

```
For continuous predictors:
```

A one-unit increase in the coefficient is associated with a change in the log odds of the outcome equal to the coefficient value.

```
For categorical predictors (dummy variables):
```

The coefficient represents the difference in log odds between the reference category (usually the category with the value of 0) and the category represented by the dummy variable.

## significance

I think that the practical significance of this reduced model is moderate. By reading the coefficients in the regression equation we can identify some factors that correlate to a higher probability of churn. We can also see a few factors that correlate to a lower rate of churn. The accuracy is .73 so the model is predicting outcomes correctly in the test set.

The statistical significance here appears good by looking at psuedo R squared and log-likelihood. However I don't think this is a very good model because if you look at the confusion matrix, you can see that it almost had as many false positives as it predicted true positives. I think this may have been caused by the data set it self being skewed. The 'Churn' variable in this data set is mostly false. This looks like it is causing the model to have difficulty predicting 'churn'. I think this could be done better maybe with more data or some different variables. I really don't like the confusion matrix. The accuracy is .73 but it is misleading because those are all true negatives as because most of the 'churn' variable is set to false.

### Limitations.

Some of the limitations of this analysis are that the model is only predicting true negatives with with any accuracy. The ability to predict true positives is almost less than half the predictions. Also I think there is not a clear linear relationship between the predictors and the outcome. That makes the model have lower accuracy. I think that is reflected in the confusion matrix.

## 2.

My recommendations based on this analysis are that the organization should allocate resources to the sales team to upsell more fiber optic Internet, Online security and focus less on device protection and streaming movies. This is based on the coefficients of the logistic regression equation.

## Citations

Assumptions of multiple linear regression (2024) Statistics Solutions. Available at: https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/assumptions-of-multiple-linear-regression/ (Accessed: 11 April 2024).

Dansbecker (2018) Using categorical data with one hot encoding, Kaggle. Available at: https://www.kaggle.com/code/dansbecker/using-categorical-data-with-one-hot-encoding (Accessed: 11 April 2024).

Sklearn.metrics.confusion_matrix (no date) scikit. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html#sklearn.metri (Accessed: 12 April 2024).

Explore the core of logistic regression assumptions (2023) Voxco. Available at: https://www.voxco.com/blog/logistic-regression-assumptions/ (Accessed: 11 April 2024).

How to replace column values in a pandas DataFrame (2023) Saturn Cloud Blog. Available at: https://saturncloud.io/blog/how-to-replace-column-values-in-a-pandas-dataframe/ (Accessed: 06 April 2024).

In [47]:
```
import sys
print(sys.version)
```

```
3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]
```

In [ ]: