D207 Mattinson (2nd Submission)

```
########################################
        STUDENT INFORMATION
########################################
       Student | Mike Mattinson
    Student ID | 001980761
         Class | D207 Exploratory Data Analysis
       Dataset | Churn
    Submission | 2nd Submission
        School | WGU
    Instructor | Dr. Sewell
Today is July 31, 2021
```

This is what a code snippet looks like:

```
# imports
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import datetime
from mm_lib import *
```

This is what terminal output looks like:

```
##################################
        DATAFRAME (DF)
##################################
  Customer_id         City Churn  MonthlyCharge     Tenure
0     K409198  Point Baker    No     172.455519   6.795513
1     S120509  West Branch   Yes     242.632554   1.156681
2     K191035      Yamhill    No     159.947583  15.754144
3      D90850      Del Mar    No     119.956840  17.087227
(10000, 50)
```

Note. Changes from 1st submission are identified by vertical line in left margin and dark blue colored font.

Table of Contents

List of Tables

List of Figures

Part A Research Question & Benefits

Part A1. Is the predictor variable 'MonthlyCharge' dependent of the target variable 'Churn'? Is there confidence (at the 99% significance level) that 'MonthlyCharge' can be used to predict 'Churn'?

Part A2. Stakeholders will benefit from knowing the answer to this question. Future market research can make changes and recommendations regarding 'MonthlyCharge' where customers retain services longer.

Part A3. The company provided a dataset of 10,000 customer records. 'Churn' is the categorical variable to indicate ('Yes' or 'No') whether the customer has terminated service within the last month. This will be our target variable. In addition, there are a number of other categorical and numerical variables that can be analyzed as predictor variables. This analysis will consider the following:

Tenure – **Numerical** – Number of months the customer has stayed with the provider.

MonthlyCharge – **Numerical** (our primary predictor for part A1 and the following hypothesis testing to follow. It is the amount charged to the customer monthly.

Bandwidth_GB_Year – **Numerical** – Average amount of data used, in GB, in a year by the customer.

Outage_sec_perweek – Numerical - Average number of seconds per week of system outage s in the customer's neighborhood

Income – Numerical - : Annual income of cu stomer as reported at time of sign up

Gender – Categorical - Customer self identification as male, female, or nonbinary

InternetService – Categorical - Customer's internet service provider (DSL, fiber optic, None)

Remove Unwanted Data

Some of the original data was removed:

```
# remove unwanted columns
unwanted_columns = ["UID", "Interaction", "Lat", "Lng"]

for uc in unwanted_columns:
    if uc in df.columns:
        df.drop(columns=uc, inplace=True)
        print("[{}] column removed.".format(uc))

# show remaining columns
print("Remaining columns: \n{}".format(df.columns))
```

```
###########################################
        REMOVE UNWANTED COLUMNS
###########################################
[UID] column removed.
[Interaction] column removed.
[Lat] column removed.
[Lng] column removed.
Remaining columns:
Index(['CaseOrder', 'Customer_id', 'City', 'State', 'County',
'Zip',
       'Population', 'Area', 'TimeZone', 'Job', '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', 'Item1',
'Item2', 'Item3',
       'Item4', 'Item5', 'Item6', 'Item7', 'Item8'],
      dtype='object')
```

Numerical Data

Here is the list of available continuous data:

```
print(heading_toString("CONTINUOUS DATA"))
print(df.select_dtypes(include="float").info())
```

```
###################################
        CONTINUOUS DATA
###################################
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 5 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Income               10000 non-null  float64
 1   Outage_sec_perweek   10000 non-null  float64
 2   Tenure               10000 non-null  float64
 3   MonthlyCharge        10000 non-null  float64
 4   Bandwidth_GB_Year    10000 non-null  float64
dtypes: float64(5)
memory usage: 390.8 KB
None
```

Here is a list of available integer data:

```
print(heading_toString("INTEGER DATA"))
print(df.select_dtypes(include="integer").info())
```

```
################################
        INTEGER DATA
################################
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   CaseOrder              10000 non-null  int64
 1   Zip                    10000 non-null  int64
 2   Population             10000 non-null  int64
 3   Children               10000 non-null  int64
 4   Age                    10000 non-null  int64
 5   Email                  10000 non-null  int64
 6   Contacts               10000 non-null  int64
 7   Yearly_equip_failure   10000 non-null  int64
 8   Item1                  10000 non-null  int64
 9   Item2                  10000 non-null  int64
 10  Item3                  10000 non-null  int64
 11  Item4                  10000 non-null  int64
 12  Item5                  10000 non-null  int64
 13  Item6                  10000 non-null  int64
 14  Item7                  10000 non-null  int64
 15  Item8                  10000 non-null  int64
dtypes: int64(16)
memory usage: 1.2 MB
None
```

Categorical Data

Here is a list of available categorical data:

```
print(heading_toString("CATEGORICAL DATA"))
print(df.select_dtypes(include="object").info())
```

```
####################################
        CATEGORICAL DATA
####################################
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Customer_id       10000 non-null  object
 1   City              10000 non-null  object
 2   State             10000 non-null  object
 3   County            10000 non-null  object
 4   Area              10000 non-null  object
 5   TimeZone          10000 non-null  object
 6   Job               10000 non-null  object
 7   Marital           10000 non-null  object
 8   Gender            10000 non-null  object
 9   Churn             10000 non-null  object
 10  Techie            10000 non-null  object
 11  Contract          10000 non-null  object
 12  Port_modem        10000 non-null  object
 13  Tablet            10000 non-null  object
 14  InternetService   10000 non-null  object
 15  Phone             10000 non-null  object
 16  Multiple          10000 non-null  object
 17  OnlineSecurity    10000 non-null  object
 18  OnlineBackup      10000 non-null  object
 19  DeviceProtection  10000 non-null  object
 20  TechSupport       10000 non-null  object
 21  StreamingTV       10000 non-null  object
 22  StreamingMovies   10000 non-null  object
 23  PaperlessBilling  10000 non-null  object
 24  PaymentMethod     10000 non-null  object
dtypes: object(25)
memory usage: 1.9+ MB
None
```

Part B Analysis, code and output

Part B1. All the analysis is conducted using Python programming language using Microsoft's Visual Studio Code IDE.

Part B2. The complete .PY file is attached. In addition, selected 'Code Snippets' and 'Terminal Output' will be included and described in this document

Part B3. This analysis will use the '**Chi-square, Independence Test'** The test is suited to the analysis of two categorical variables.

Here is the procedure I used to conduct the Chi-square **independence** test. I wrote this as a procedure so that I can reused and standardize all my testing. The assessment asks to complete one analysis test, but because it was a method, I was able to quickly run and look at other predictor variables as well. All the procedures I used were consolidated into a library file, mm_lib.py, which is attached. Here is the function that calculates and outputs the chi-square test:

```
def chi_square_analysis(target,predictor,bins,prob,df):
    ''' Calculate and display results of chi-square independence test

    Arg:
       target - target variable
       predictor - predictor variable
       bins - number of bins, if numerical, 0 if categorical
       prob - probability
       df - dataframe

    Return: outputs analysis to terminal.

    '''
    import pandas as pd
    from scipy.stats import chi2_contingency
    from scipy.stats import chi2
    if bins == 0:
        # variable is categorical
        contingency= pd.crosstab(df[target], df[predictor])
    else:
        # variable is numerical
        mc_groups =pd.cut(df[predictor], bins=bins)
        contingency= pd.crosstab(df[target], mc_groups)

    # print out the contingency table
    print(contingency.T)

    stat, p, dof, expected = chi2_contingency(contingency)
    alpha = 1 - prob # significance level
    critical = chi2.ppf(prob, dof)

    # interpret test-statistic
    if abs(stat) >= critical:
        print('At an alpha level of {}, the critical value is {}.'.fo
rmat(round(alpha,3),round(critical,3)))
        print('chi^2 = {} > {}, therefore, variables are dependent (r
eject H0).'.format(round(stat,3),round(critical,3)))
        #print('Also, we will need to perform additional analysis to
determine extent of the relationship.')
    else:
        print('chi^2 = {} <= {}, therefore, variables are independent
 (fail to reject H0).'.format(round(stat,3),round(critical,3)))

    print(' p-value: {:>8.3f}'.format(p))
    print('     dof: {:>8d}'.format(dof))
    print('   alpha: {:>8.3f}'.format(alpha))
    print('    stat: {:>8.3f}'.format(stat))
    print('critical: {:>8.3f}'.format(critical))
```

Here is the snippet where the procedure was called:

```
# CHI-SQUARE TEST
target = 'Churn'
prob = 0.999
predictor = 'MonthlyCharge' # numerical
print(heading_toString("PART B - CHI-
SQUARE INDEPENDENCE TEST - " + target + " vs. " + predictor))
bins = 6

# use library function to perform test
chi_square_analysis(target,predictor,bins,prob,df)

# use library function to print out distribution table
print(heading_toString("CHI-SQUARE DISTRIBUTION TABLE"))
print(chi_square_dist_table(0,0))
```

The null hypothesis is that 'Churn' and 'MonthlyCharge' are independent of each other, the null hypothesis is defined as HO.

Null Hypothesis (H0) – variables are independent

Alternate Hypothesis (H1) – variables are not independent

Here is the terminal output for the test:

```
############################################################
######################
        PART B - CHI-SQUARE INDEPENDENCE TEST - Churn vs.
MonthlyCharge
############################################################
######################
Churn                     No   Yes
MonthlyCharge
(79.769, 115.009]        755    35
(115.009, 150.039]      2477   338
(150.039, 185.07]       2356   693
(185.07, 220.1]         1009   608
(220.1, 255.13]          579   715
(255.13, 290.16]         174   261
At an alpha level of 0.001, the critical value is 20.515.
chi^2 = 1425.642 > 20.515, therefore, variables are dependent
(reject H0).
 p-value:     0.000
     dof:         5
   alpha:     0.001
    stat:  1425.642
critical:    20.515
```

A couple of notes for this test. 'MonthlyCharge' is a numerical variable, so I used **bins=6** to break the numerical values into groups in order to conduct the test. The p-value is extremely low, lower than the alpha of 0.001, and the calculated statistic was extremely high compared to the critical value. Both indicate strong support that the two variables are dependent. The analysis indicates to '**reject null hypothesis'**. Also, further testing and research will be needed to determine the full extent of the relationship.

The test critical value, X^2, was determined as a function of degree of freedom (dof) and probability (prob). I used the chi2 method in the schipy.stats package to calculate the value, but I also wanted to see the table for reference, so I made a library function to create a distribution table, here is the Chi-square lower-tail distribution table from terminal output using my function:

Table 1 CHI-SQUARE DISTRIBUTION TABLE

```
##############################################
        CHI-SQUARE DISTRIBUTION TABLE
##############################################
Lower-tail critical values of chi-Square distribution:
 dof   0.100   0.050   0.025   0.010   0.001
----------------------------------------------
   1    2.71    3.84    5.02    6.63   10.83
   2    4.61    5.99    7.38    9.21   13.82
   3    6.25    7.81    9.35   11.34   16.27
   4    7.78    9.49   11.14   13.28   18.47
   5    9.24   11.07   12.83   15.09   20.52
   6   10.64   12.59   14.45   16.81   22.46
   7   12.02   14.07   16.01   18.48   24.32
   8   13.36   15.51   17.53   20.09   26.12
   9   14.68   16.92   19.02   21.67   27.88
  10   15.99   18.31   20.48   23.21   29.59
  11   17.28   19.68   21.92   24.72   31.26
  12   18.55   21.03   23.34   26.22   32.91
  13   19.81   22.36   24.74   27.69   34.53
  14   21.06   23.68   26.12   29.14   36.12
  15   22.31   25.00   27.49   30.58   37.70
  16   23.54   26.30   28.85   32.00   39.25
  17   24.77   27.59   30.19   33.41   40.79
  18   25.99   28.87   31.53   34.81   42.31
  19   27.20   30.14   32.85   36.19   43.82
```

Table was generated using the following code:

```
def chi_square_dist_table(dof,prob):
    '''create chi square distribution table

    Args:
        dof - degree of freedom
        prob - significance level (alpha)

    Returns:
        A string representation of the chi-square
        lower-left distribution table.

    '''
    from scipy.stats import chi2
    prob_range = [ .10, 0.05, 0.025, 0.01, 0.001]
    dof_range = range(1,20)
    output = 'Lower-tail critical values of chi-
Square distribution:\n'.format()

    # header row
    output += '{:>4}'.format('dof') # blank space for first col
    for p in prob_range:
        output += '{:>8.3f}'.format(p)
```

```
    output += '\n' +'-' * 47

    # each row
    for d in dof_range:
        output += '\n{:4d}'.format(d)
        for p in prob_range:
            temp = chi2.isf(p, d)
            if d == dof and p == prob:
                output += '{:>8.2f}'.format(temp)
            else:
                output += '{:>8.2f}'.format(temp)

    return output
```

Part C Univariate Statistics

Part C. Identify the distribution of two continuous variables and two categorical variables using univariate statistics from your cleaned and prepared data. The plots were created and then saved as .PNG files.

Descriptive Statistics

To begin, it is helpful to show traditional descriptive statistics for the numerical data, this includes the center of the data and the range. Here is the terminal output showing the descriptive statistics of the numerical (float) variables:

```
#####################################
        DESCRIPTIVE STATS
#####################################
          Income  Outage_sec_perweek    Tenure  MonthlyCharge
Bandwidth_GB_Year
count   10000.00            10000.00  10000.00       10000.00
10000.00
mean    39806.93               10.00     34.53         172.62
3392.34
std     28199.92                2.98     26.44          42.94
2185.29
min       348.67                0.10      1.00          79.98
155.51
25%     19224.72                8.02      7.92         139.98
1236.47
50%     33170.60               10.02     35.43         167.48
3279.54
75%     53246.17               11.97     61.48         200.73
5586.14
max    258900.70               21.21     72.00         290.16
7158.98
```

```
print(df.select_dtypes(include="float").describe().round(2))
```

The next two plots are countplots of univariant categorical variables:

- Fig_1_Churn_categorical_countplot.png
- Fig_2_InternetService_categorical_countplot.png

Here is the code:

```
# visualization of selected categorical data
univariate_categorical = {
  "1": "Churn",
  "2": "InternetService",
}

for key in univariate_categorical:
    fig, ax = plt.subplots()
    c = univariate_categorical[key]
    sns.countplot(df[c])
    plt.xlabel(c)
    plt.ylabel('Count')
    count_figures = key
    #title = 'Hello world'
    sub_title = '{}_{}_{}'.format(c,'categorical','countplot')
    title = '{}'.format(c)
    fname = 'Fig_{}_{}.{}'.format(str(count_figures), sub_title, "png
")
    plt.title(title, fontsize=14, fontweight="bold")
    fig.suptitle(sub_title, fontsize=14)
    plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14, bbox={"fa
cecolor":"orange", "alpha":0.5, "pad":5})
    fig.tight_layout()

    # save file in the course figure's folder
    plt.savefig(os.path.join(figure_folder, fname))
    plt.close()
    print('figure saved at: [{}]'.format(fname))
```

Count Plot – Churn (Cat)

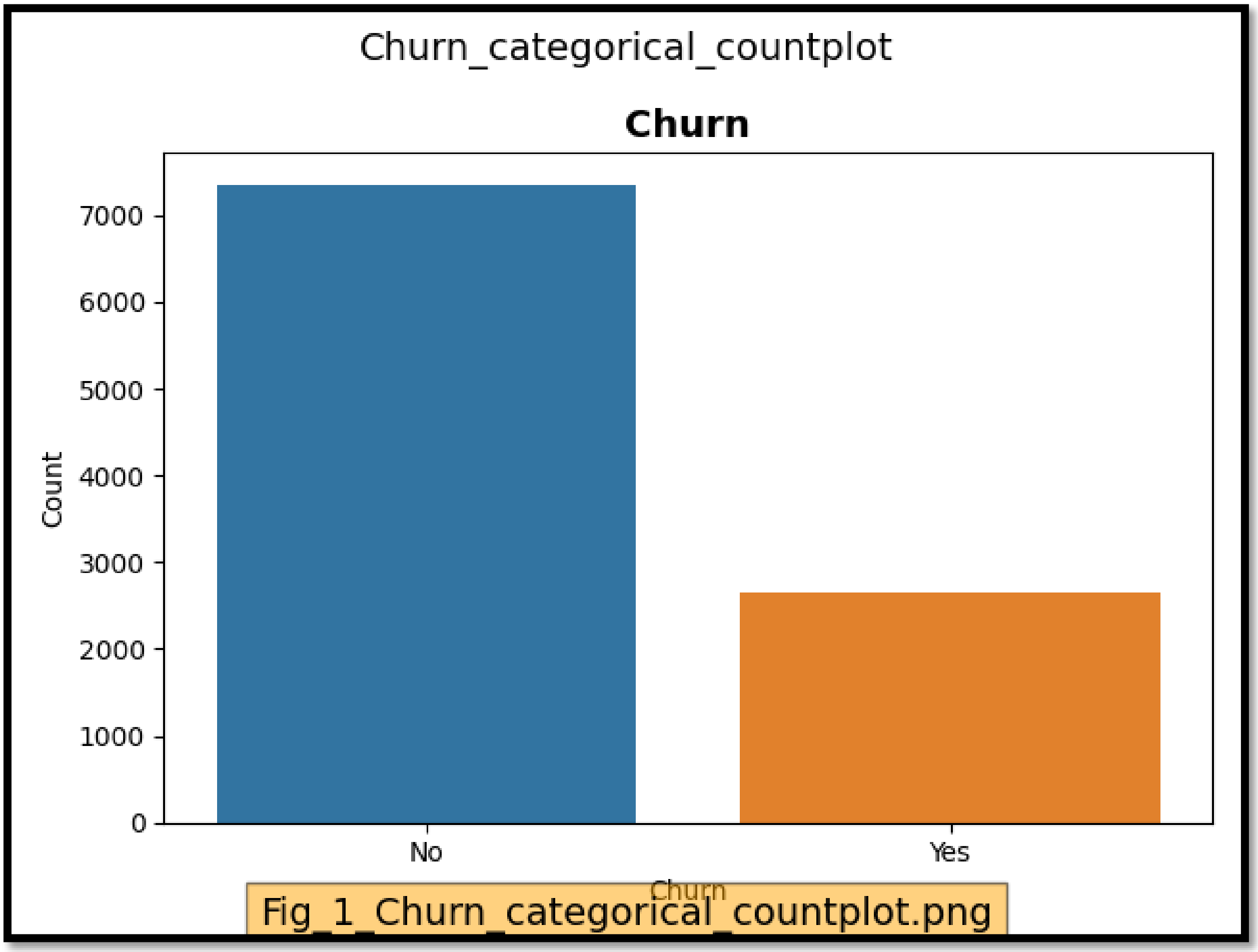


Figure 1 Churn Cat_countplot (.PNG)

Count Plot – InternetService (Cat)

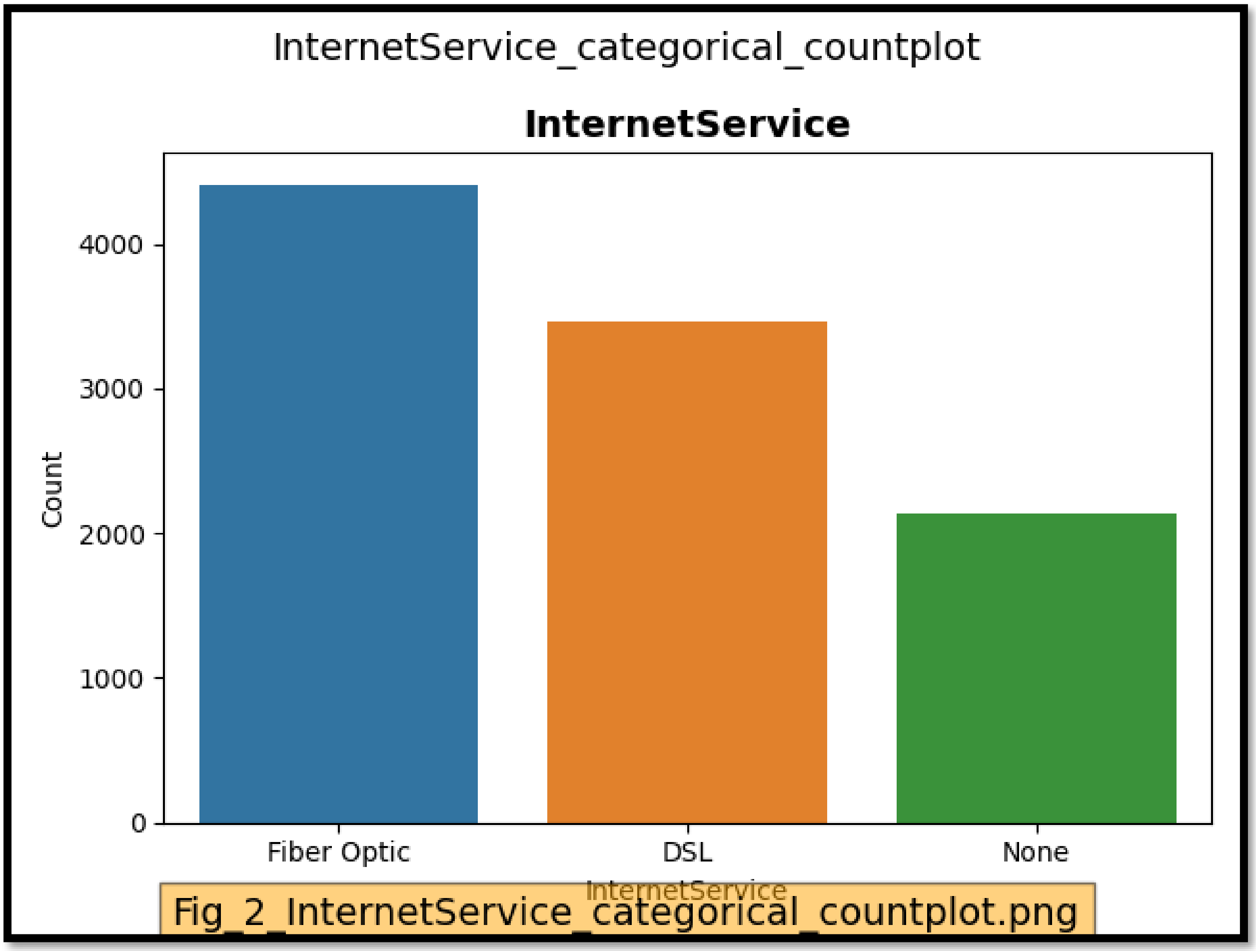


Figure 2 InternetService Cat_countplot (.PNG)

The next two plots are univariate continuous:

- Fig_3_MonthlyCharge_continuous_scatterplot.png
- Fig_4_Income_continuous_scatterplot.png

Here is the code:

```
# visualization of selected univariate continuous data
univariate_continuous = {
  "3": "MonthlyCharge",
  "4": "Income",
}

for key in univariate_continuous:
    fig, ax = plt.subplots()
    c = univariate_continuous[key]
    plt.scatter(df.index,df[c])
    plt.xlabel(c)
    plt.ylabel('Count')
    count_figures = key
    #title = 'Hello world'
    sub_title = '{}_{}_{}'.format(c,'continuous','scatterplot')
    title = '{}'.format(c)
    fname = 'Fig_{}_{}.{}'.format(str(count_figures), sub_title, "png
")
    plt.title(title, fontsize=14, fontweight="bold")
    fig.suptitle(sub_title, fontsize=14)
    plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14, bbox={"fa
cecolor":"orange", "alpha":0.5, "pad":5})
    fig.tight_layout()

    # save file in the course figure's folder
    plt.savefig(os.path.join(figure_folder, fname))
    plt.close()
    print('figure saved at: [{}]'.format(fname))
```

Scatter Plot – MonthlyCharge (Cont)

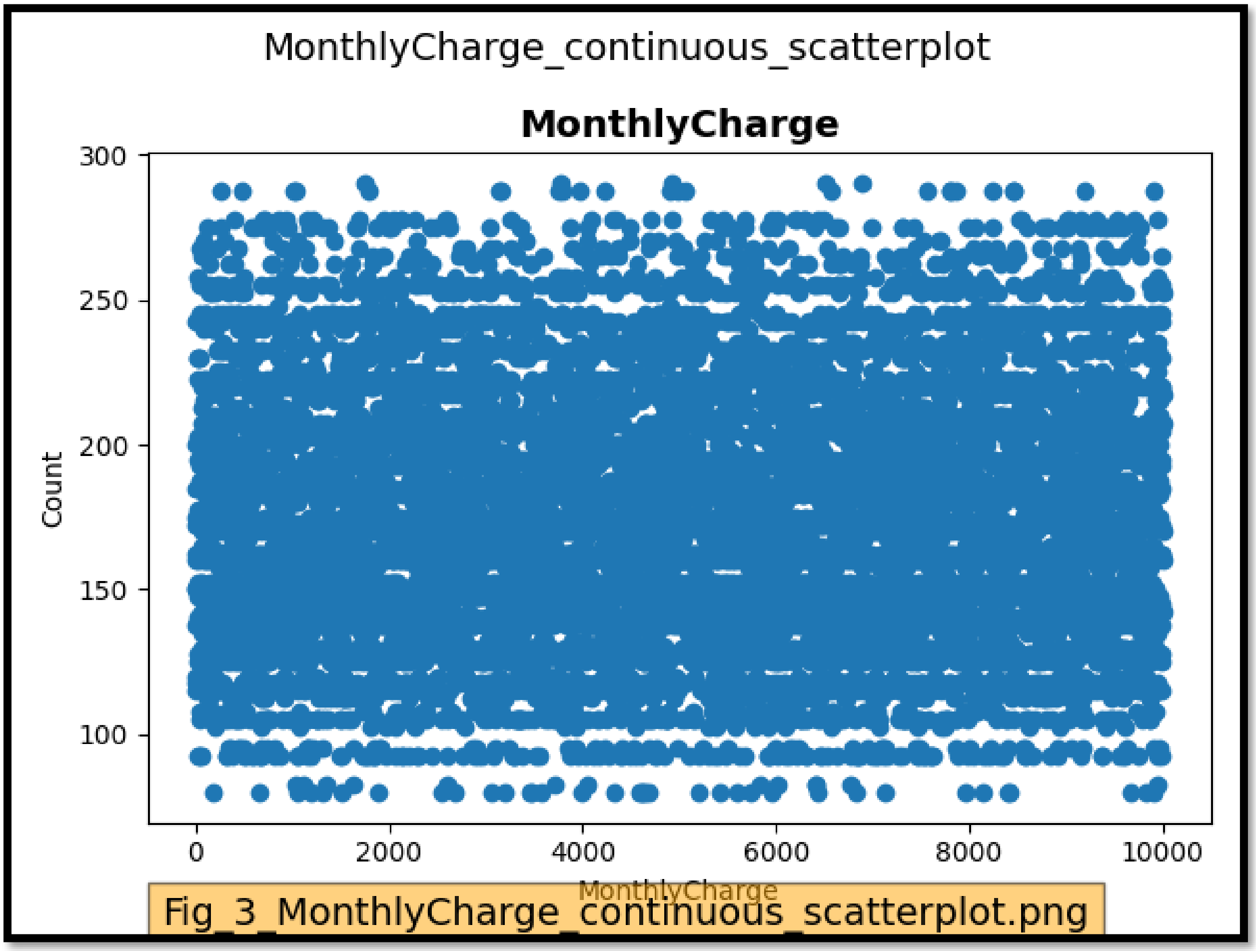


Figure 3 MonthlyCharge Cont_scatterplot (.PNG)

Scatter Plot – Income (Cont)

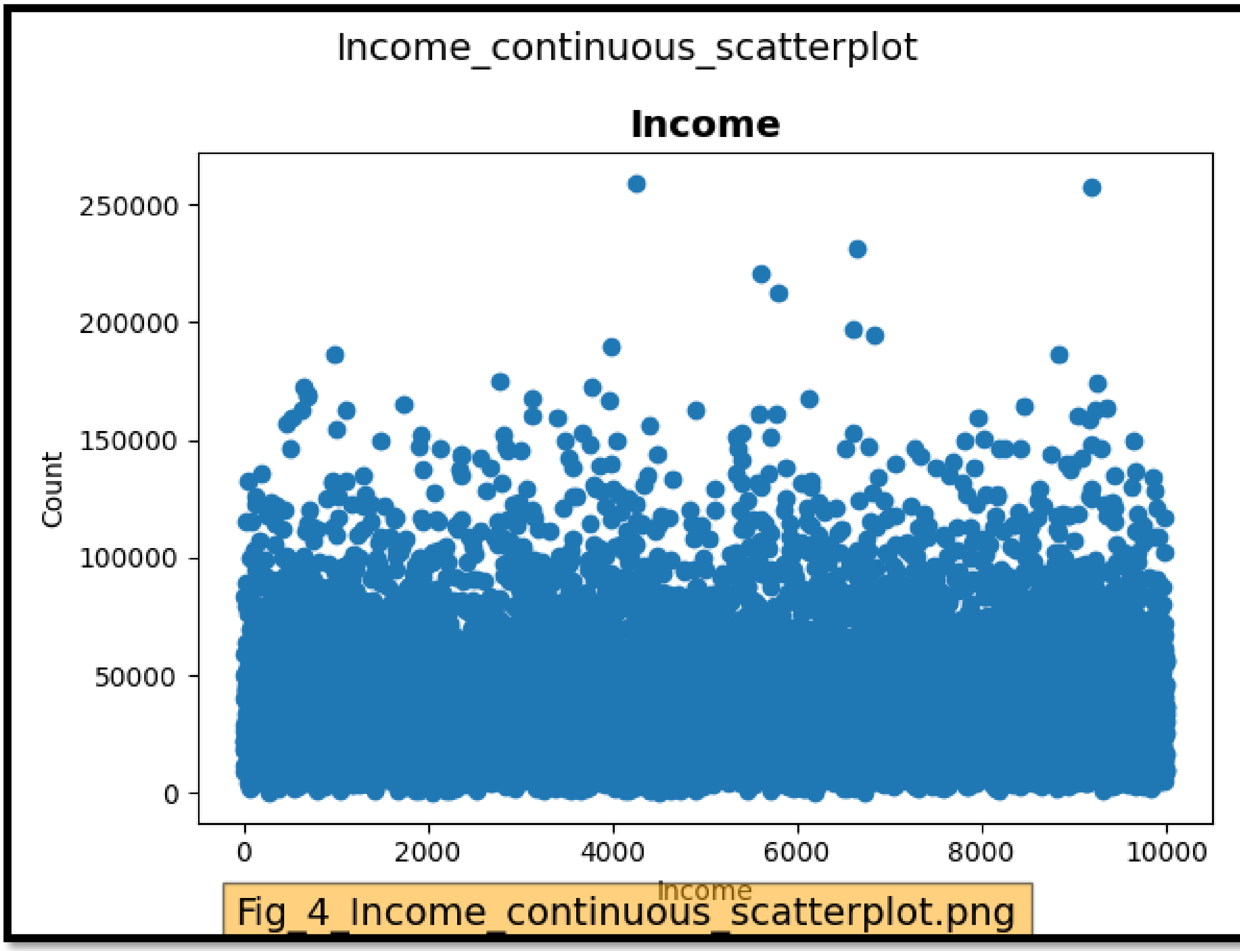


Figure 4 Income Cont_scatterplot (.PNG)

Box Plot – Normalized Problem Data (Cat)

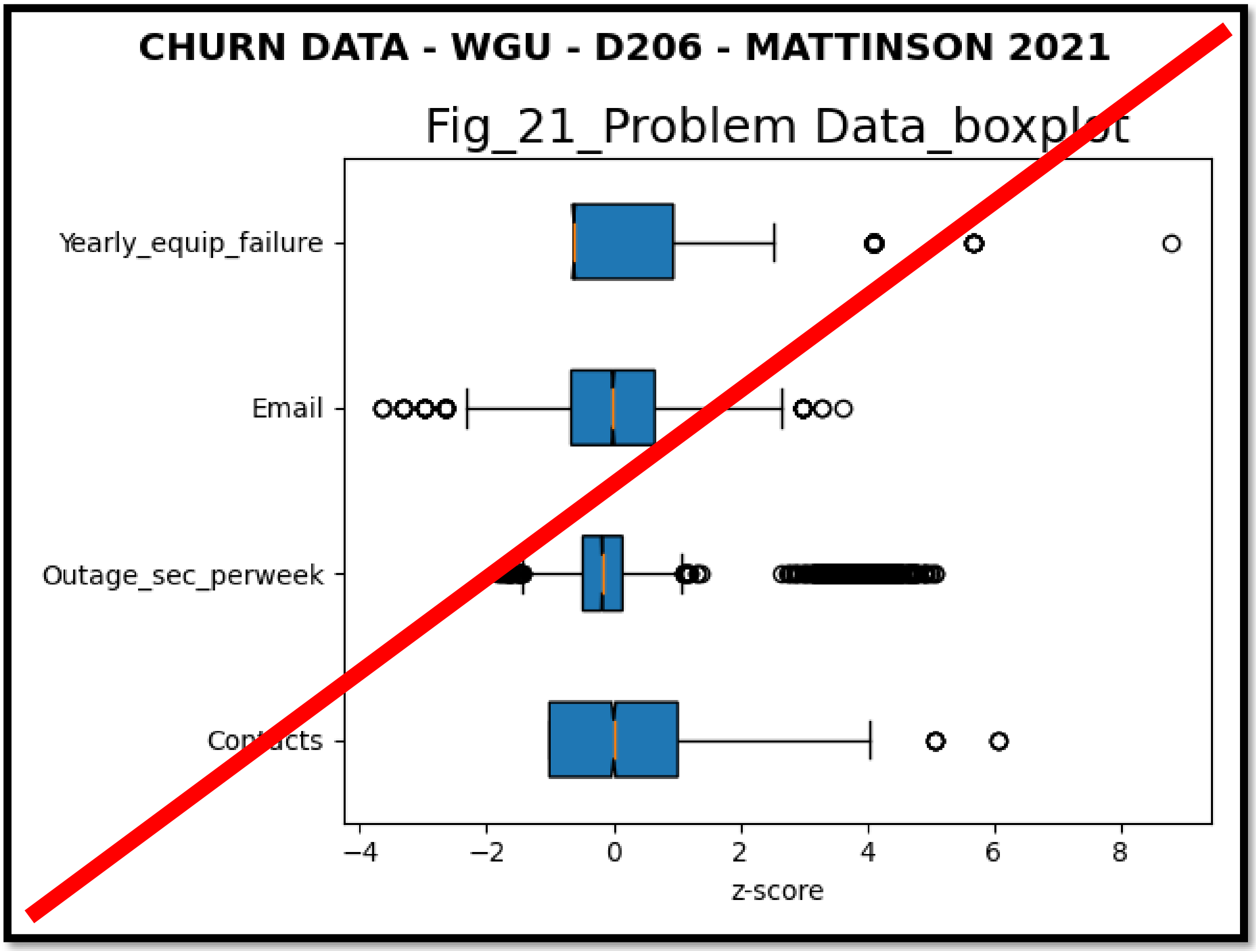


Figure 5 Problem Data_boxplot (.PNG) - not used

Figure notes: This was created using a list of related, problem, variables:

1. Yearly_equip_failure (Cont)
2. Email (Integer)
3. Outage_sec_perweek (Integer)
4. Contacts (Integer)

Box Plot – Normalized Demographic Data (Integer, Cont)

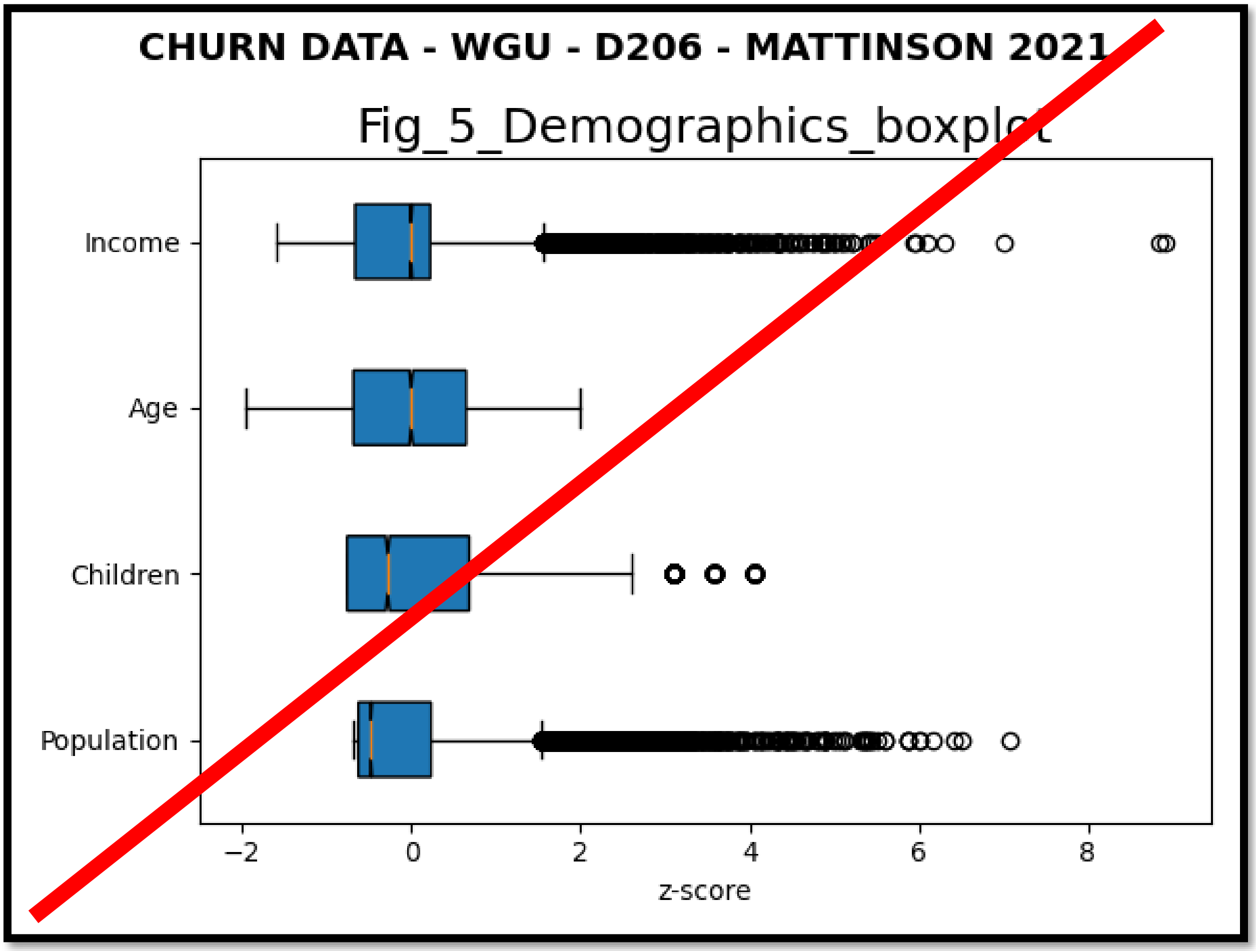
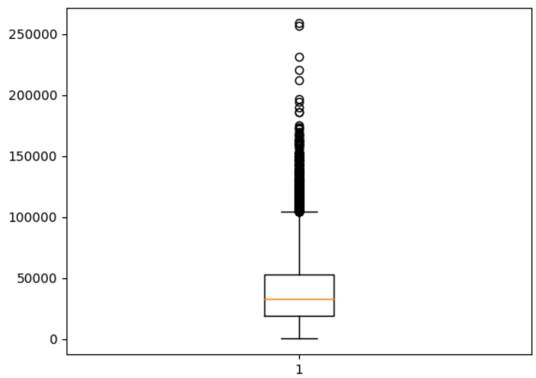
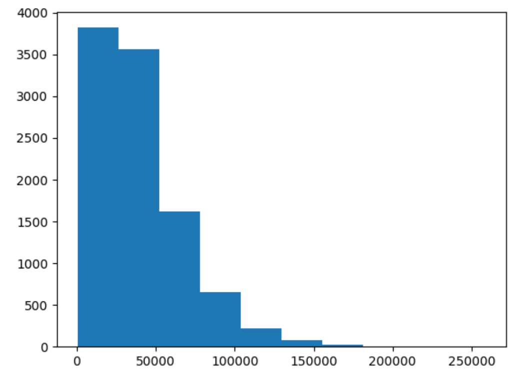
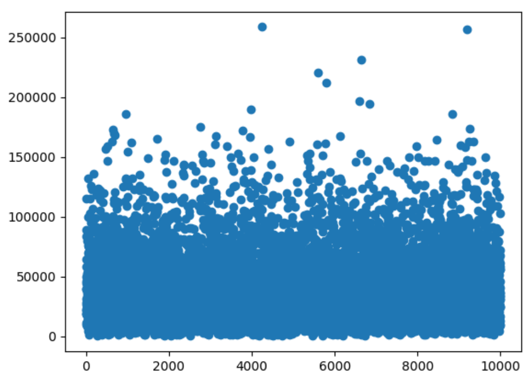


Figure 6 Demographics_boxplot (.PNG) - not used

Figure notes: This was created using a list of related, demographic, variables:

5. Income (Cont)
6. Age (Integer)
7. Children (Integer)
8. Population (Cont)

Sample of different plots – Income (Cont)

 *Figure 7 Boxplot - Income (Cont)*	 *Figure 8 Histogram plot - Income (Cont)*
BOX PLOT	HISTOGRAM
 *Figure 9 Scatter plot - Income (Cont)*	
SCATTER PLOT	

```
plt.scatter(df.index,df['Income'])
plt.hist(df['Income'])
plt.boxplot(df['Income'])
```

Part D Bivariate Statistics

Part D. Identify the distribution of two continuous variables and two categorical variables using bivariate statistics from your cleaned and prepared data. **Here is a list of all bivariate plots created:**

- Fig_10_Bivariate Income vs MonthlyCharge_Jointplot
- Fig_11_InternetService_bivariate_countplot.png
- Fig_12_TechSupport_bivariate_countplot.png
- Fig_13_PaymentMethod_bivariate_countplot.png
- Fig_14_Tablet_bivariate_countplot.png
- Fig_15_Gender_bivariate_countplot.png
- Fig_16_Port_modem_bivariate_countplot.png
- Fig_17_Techie_bivariate_countplot.png
- Fig_18_MonthlyCharge_bivariate_stacked histogram.png
- Fig_19_Bandwidth_GB_Year_bivariate_stacked histogram.png
- Fig_20_Tenure_bivariate_stacked histogram.png
- Fig_21_Outage_sec_perweek_bivariate_stacked histogram.png
- Fig_22_Income_bivariate_stacked histogram.png
- Fig_23_Gender_bivariate_stacked bar.png
- Fig_24_InternetService_bivariate_stacked bar.png

Joint Plot – Income (Cont) vs MonthlyCharge (Cont)

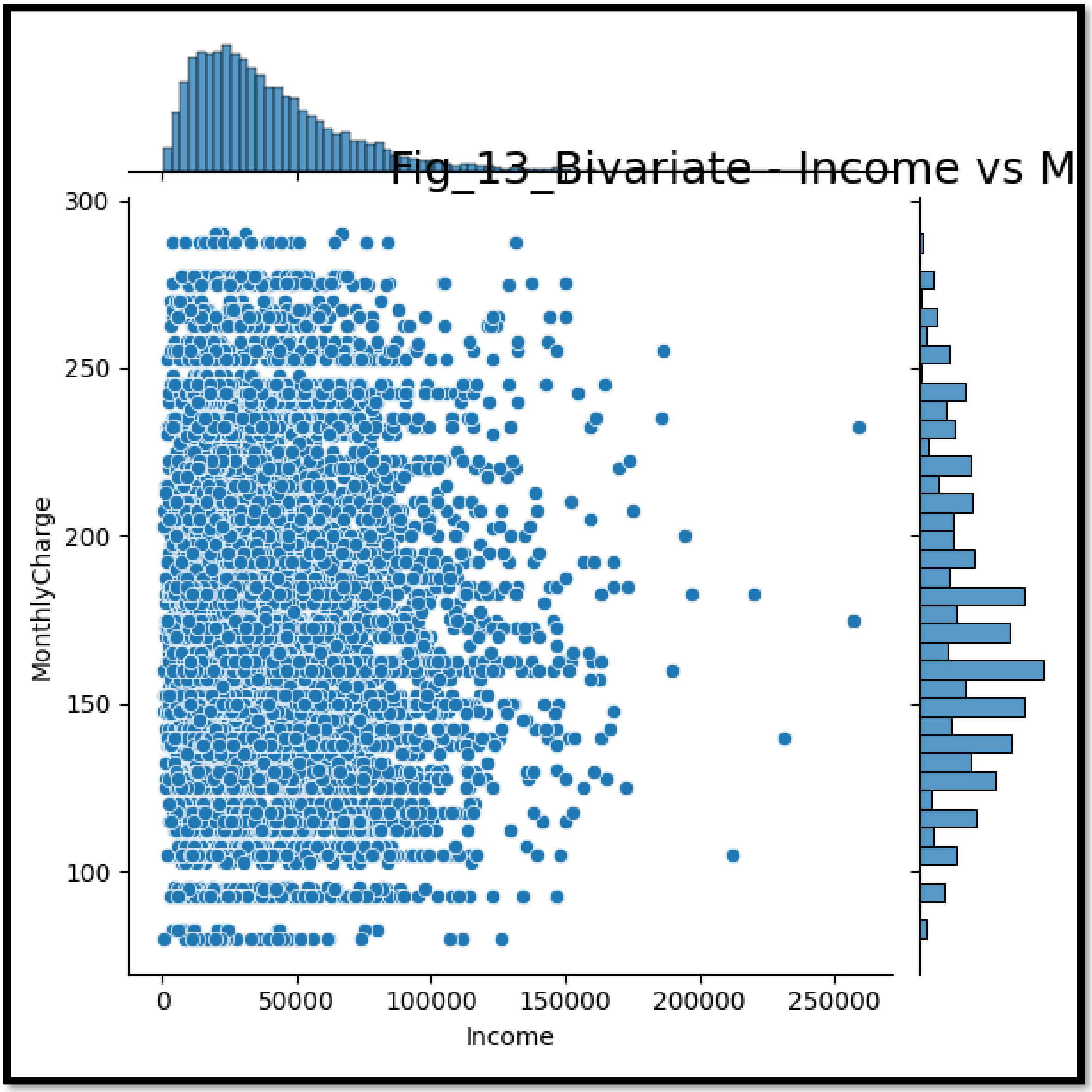


Figure 10 Birvariate - Income vs. MonthlyCharge (.PNG)

Figure notes. This plot allows you to see both variables as scatter plot, but also the individual distribution above or to the side of the data.

Ref: https://mlcourse.ai/articles/topic2-visual-data-analysis-in-python/#2.-Univariate-visualization

The next bivariate plots were countplots of two categorical variables:

- Fig_11_InternetService_bivariate_countplot.png
- Fig_12_TechSupport_bivariate_countplot.png
- Fig_13_PaymentMethod_bivariate_countplot.png
- Fig_14_Tablet_bivariate_countplot.png
- Fig_15_Gender_bivariate_countplot.png
- Fig_16_Port_modem_bivariate_countplot.png
- Fig_17_Techie_bivariate_countplot.png

Here is the code snippet that created the plots:

```
# visualization of selected bivariate data
bivariate = {
  '11': 'InternetService',
  '12': 'TechSupport',
  '13': 'PaymentMethod',
  '14': 'Tablet',
  '15': 'Gender',
  '16': 'Port_modem',
  '17': 'Techie',

}

for key in bivariate:
    fig, ax = plt.subplots()
    c = bivariate[key]
    sns.countplot(x='Churn', hue=c, data=df, palette='hls')
    plt.xlabel('{} vs. {}'.format(c,'Churn'))
    plt.ylabel('Count')
    count_figures = key
    sub_title = '{}_{}_{}'.format(c,'bivariate','countplot')
    title = '{} vs. {}'.format(c,'Churn')
    fname = 'Fig_{}_{}.{}'.format(str(count_figures), sub_title, "png
")
    plt.title(title, fontsize=14, fontweight="bold")
    fig.suptitle(sub_title, fontsize=14)
    plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14, bbox={"fa
cecolor":"orange", "alpha":0.5, "pad":5})
    fig.tight_layout()

    # save file in the course figure's folder
    plt.savefig(os.path.join(figure_folder, fname))
    plt.close()
    print('figure saved at: [{}]'.format(fname))
```

InternetService_bivariate_countplot

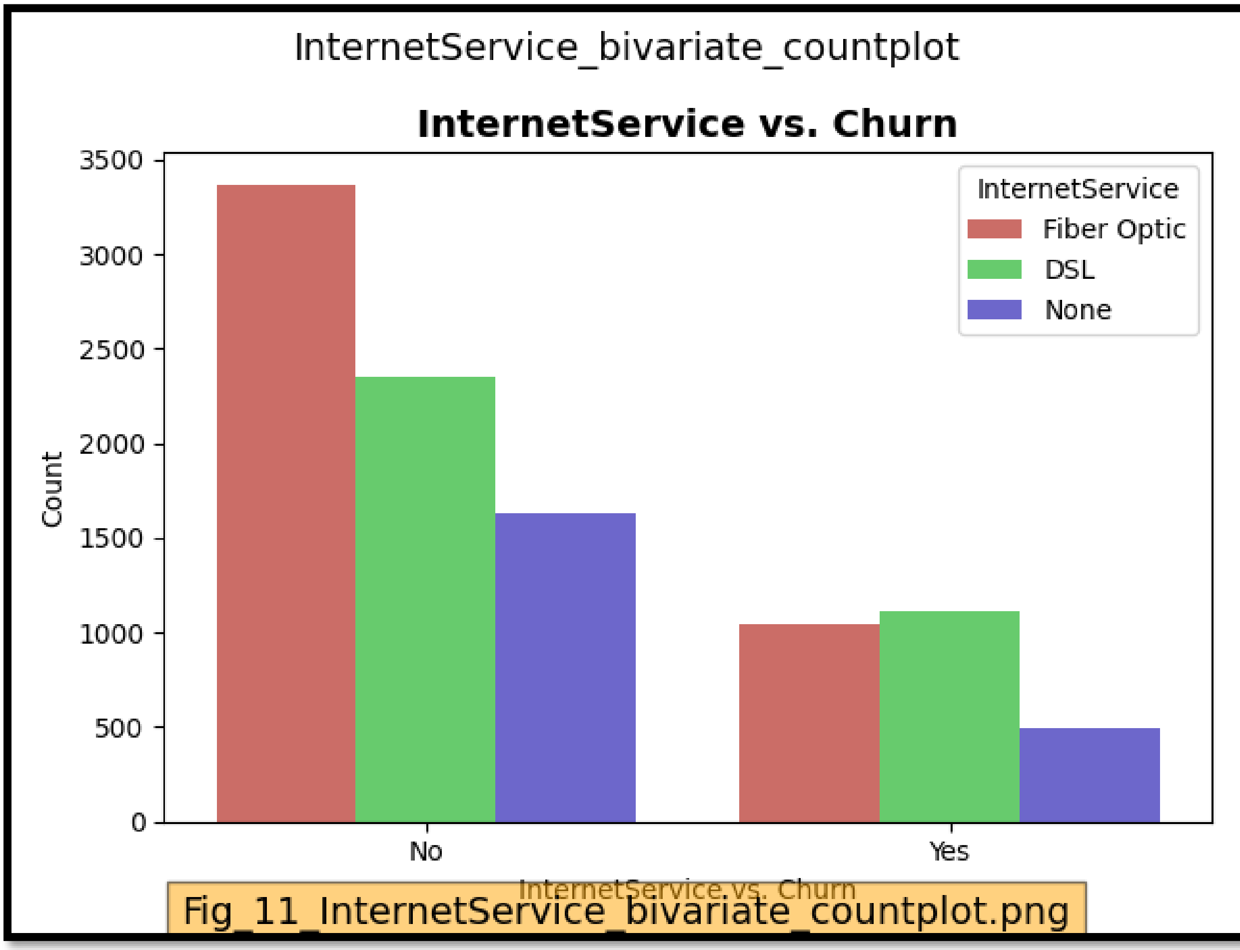


Figure 11 InternetService_bivariate_countplot

TechSupport_bivariate_countplot

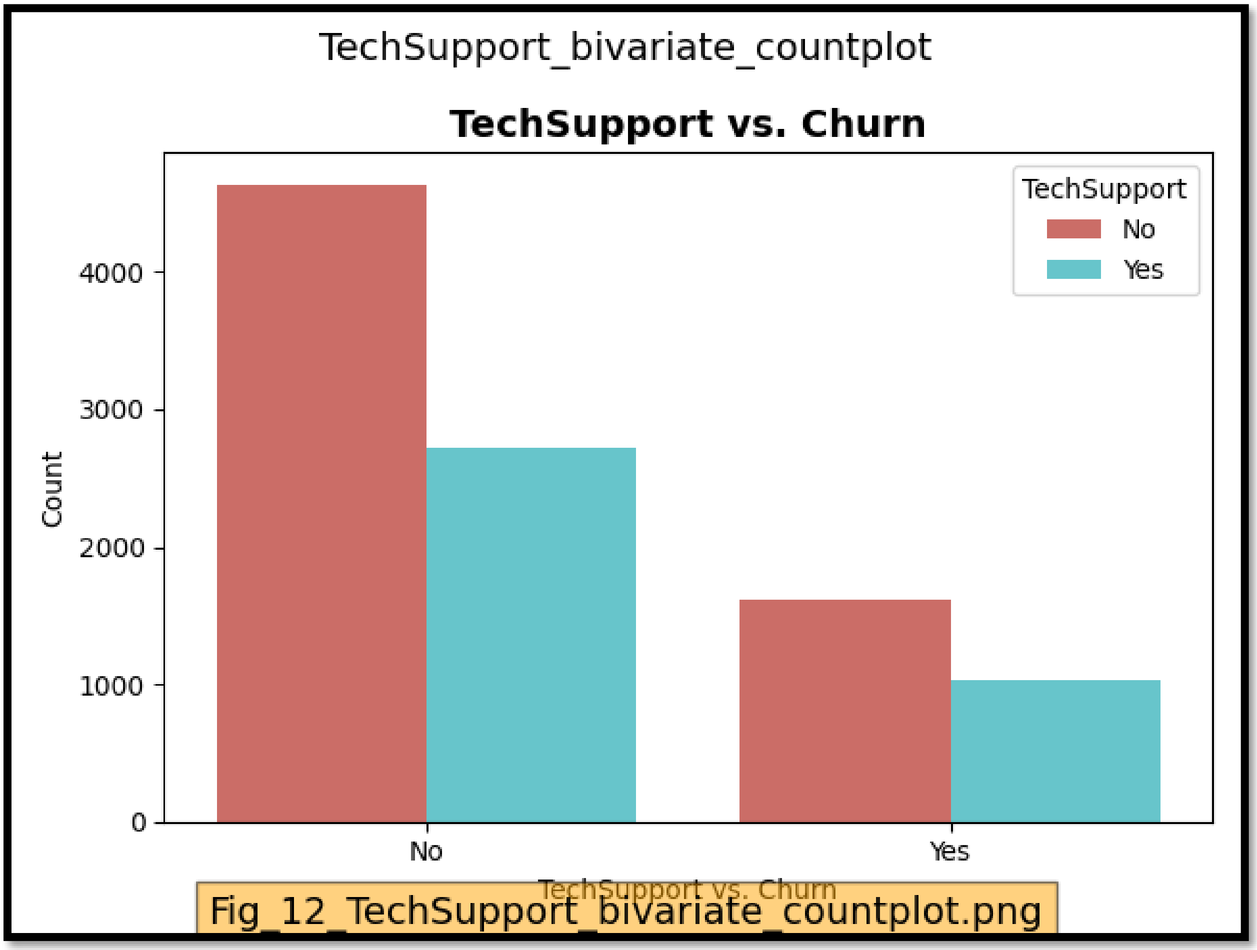


Figure 12 TechSupport_bivariate_countplot

PaymentMethod_bivariate_countplot

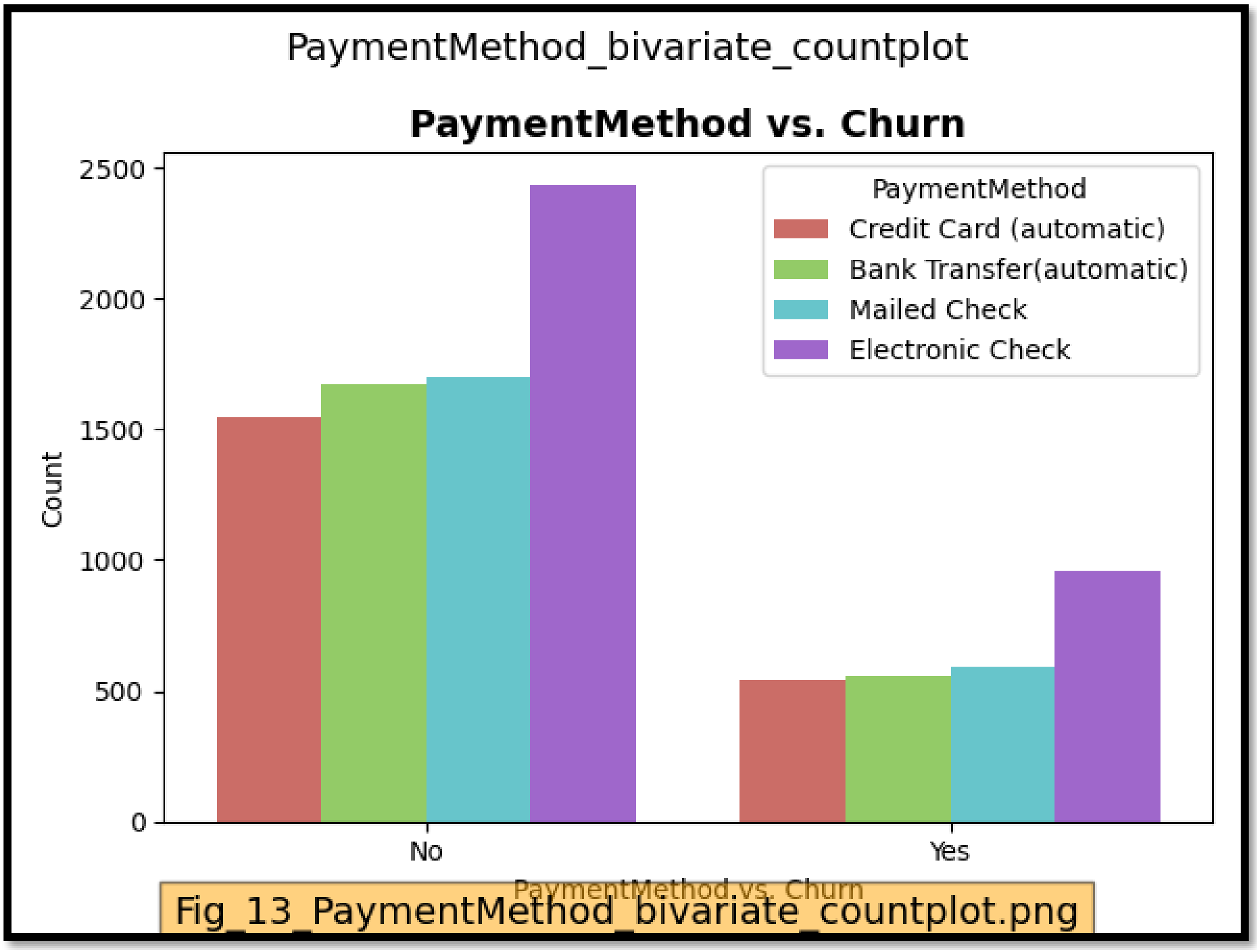


Figure 13 PaymentMethod_bivariate_countplot

Tablet_bivariate_countplot

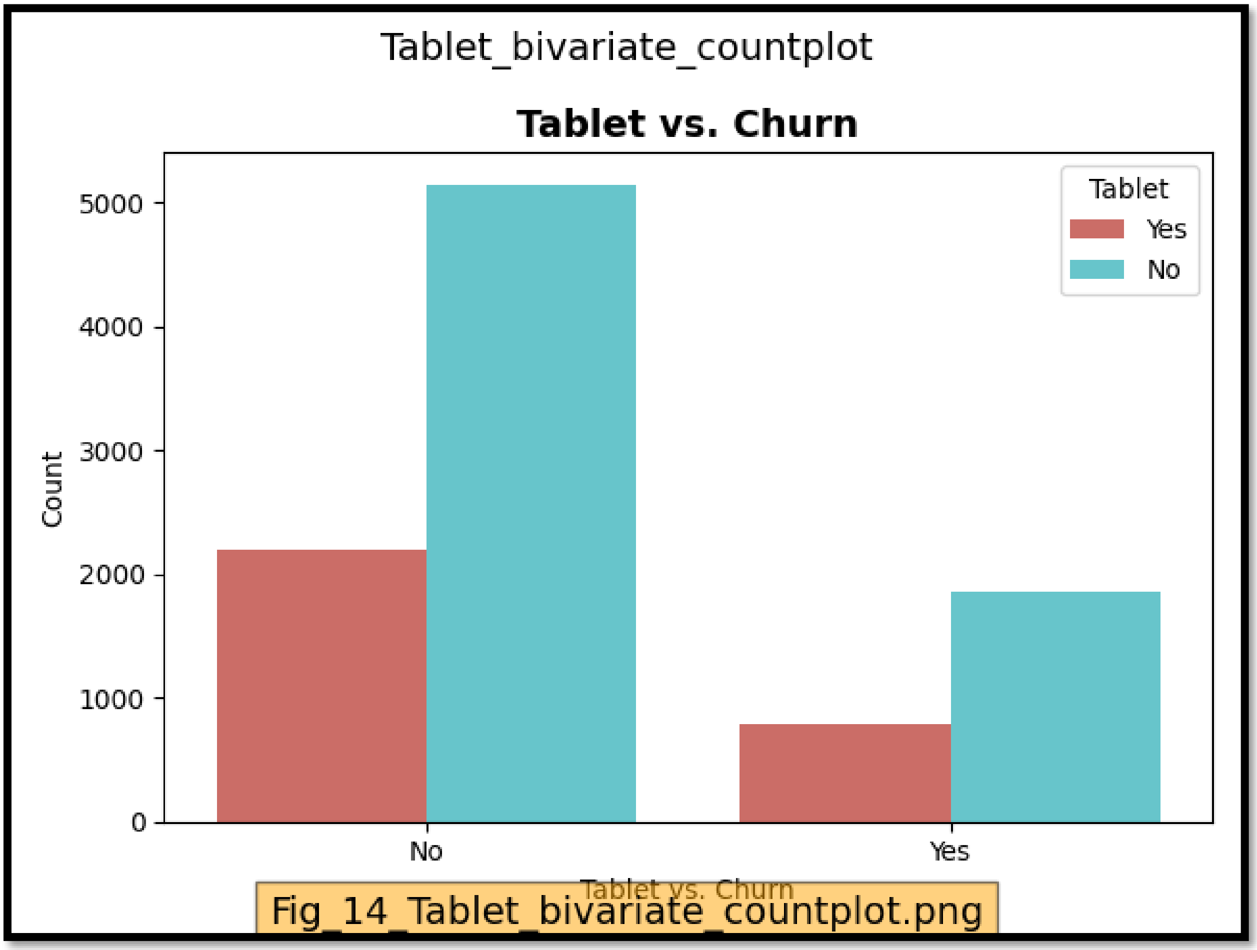


Figure 14 Tablet_bivariate_countplot

Gender_bivariate_countplot

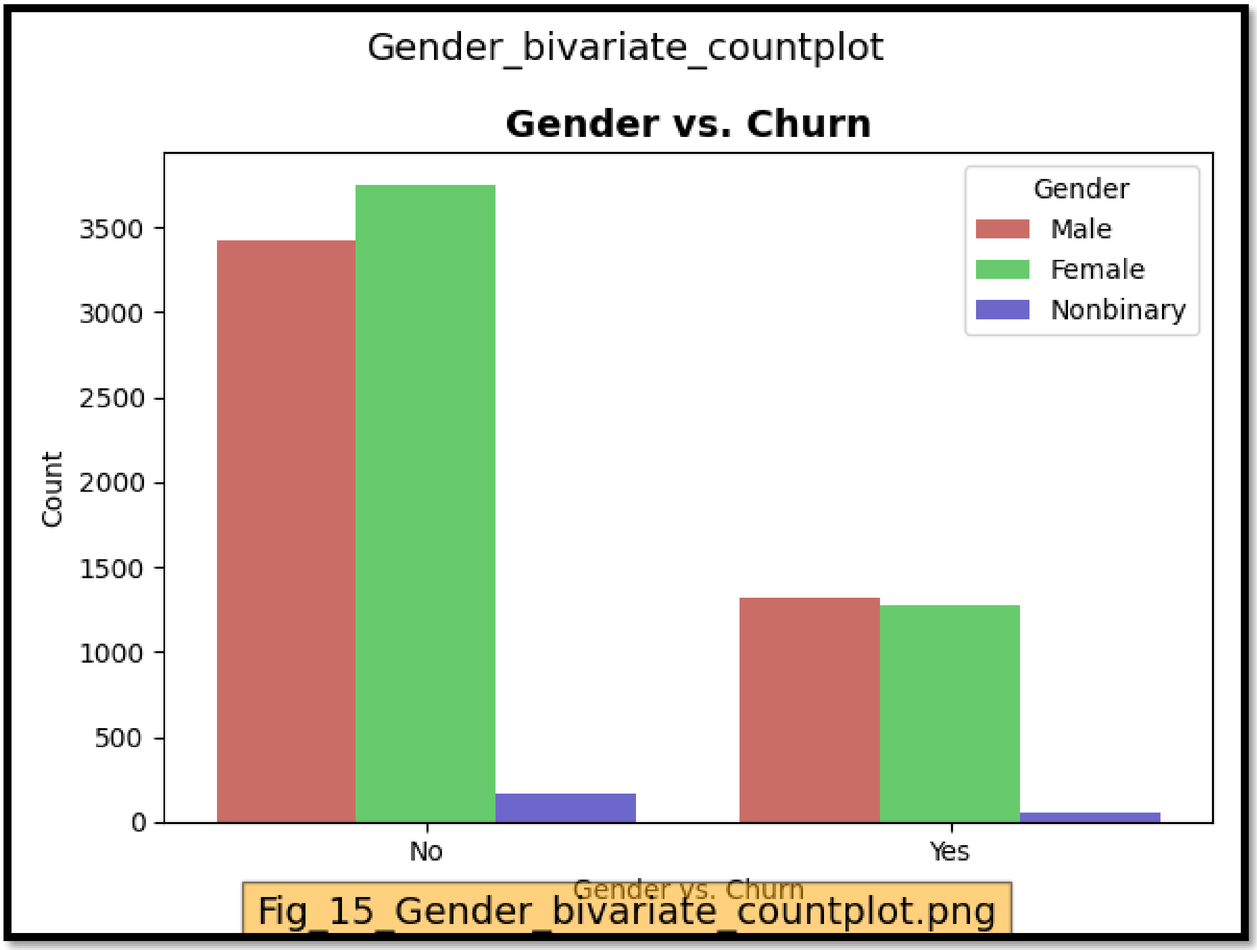


Figure 15 Gender_bivariate_countplot

Port_modem_bivariate_countplot

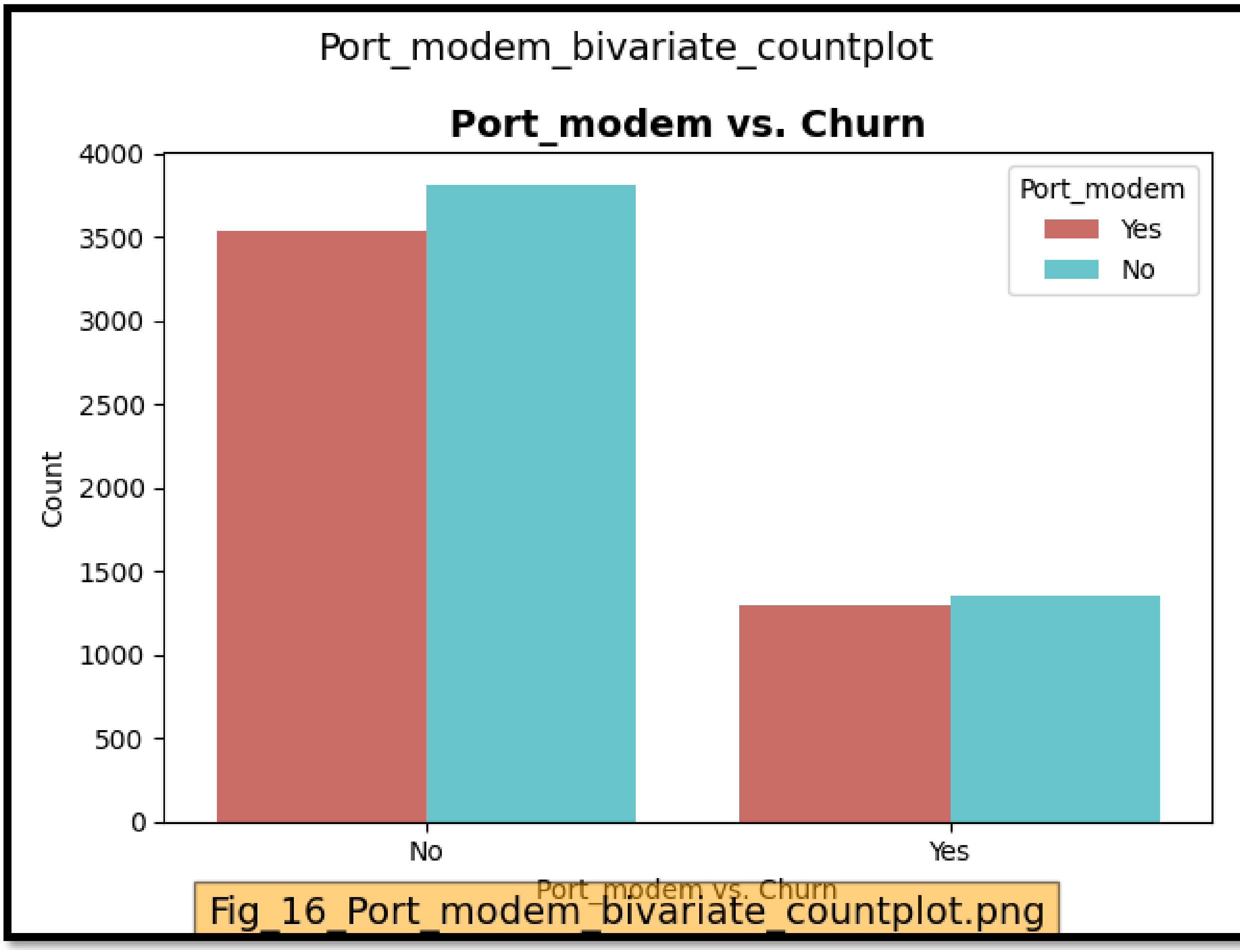


Figure 16 Port_modem_bivariate_countplot

Techie_bivariate_countplot

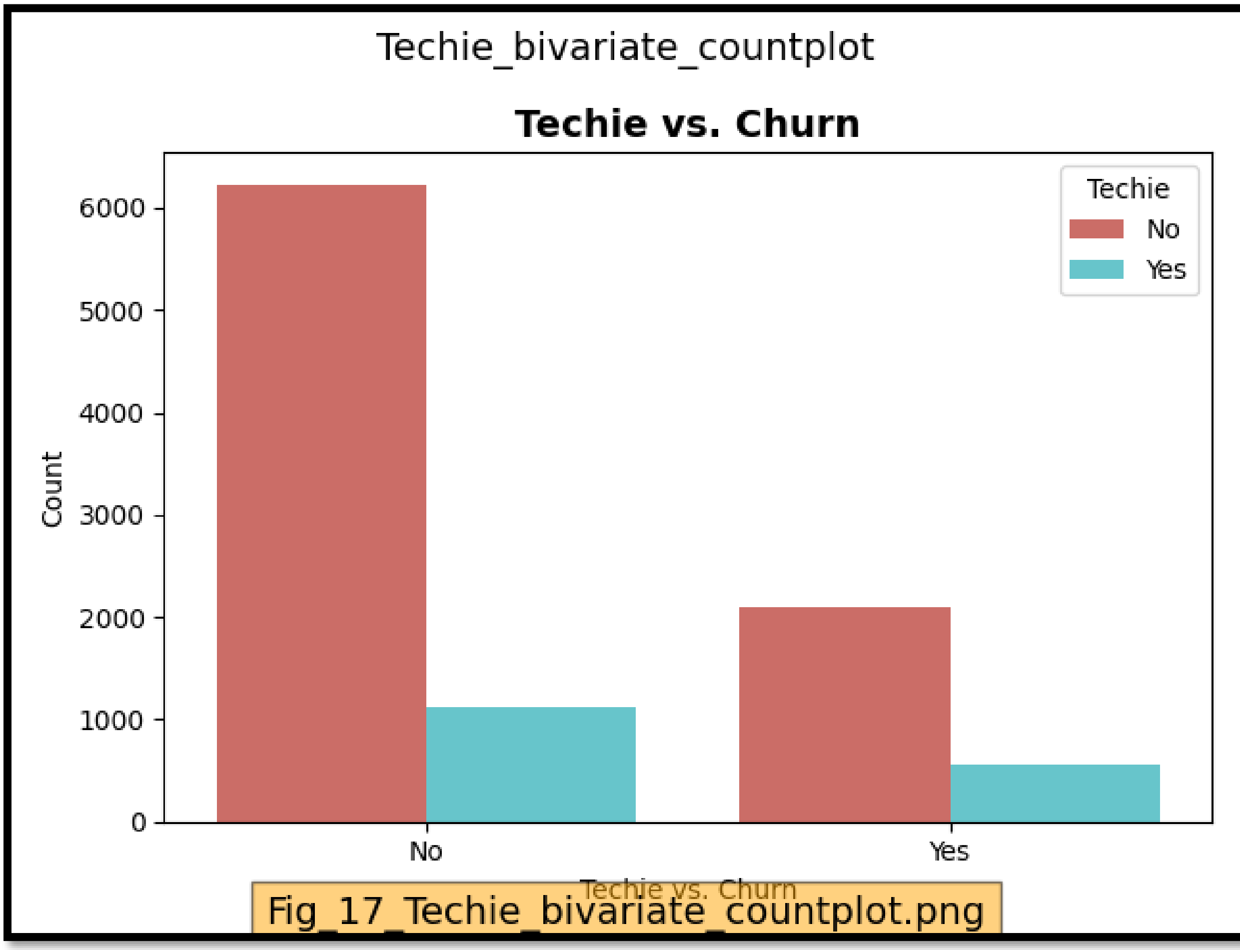


Figure 17 Techie_bivariate_countplot

The next plots are bivariate plots of continuous variables overlayed with the target 'Churn' (categorical) variable:

- Fig_18_MonthlyCharge_bivariate_stacked histogram.png
- Fig_19_Bandwidth_GB_Year_bivariate_stacked histogram.png

Here is the code:

```
# visualization of selected continous data
bivariate = {
  '18': 'MonthlyCharge',
  '19': 'Bandwidth_GB_Year',
  '20': 'Tenure',
  '21': 'Outage_sec_perweek',
  '22': 'Income',
}

for key in bivariate:
    c = bivariate[key]
    target = 'Churn'

    # print crosstab table associated with figure
    print(heading_toString(c + " crosstab"))
    print(pd.crosstab(pd.cut(df[c], bins = 6), df[target], margins=Tr
ue))

    df_yes = df[df.Churn == "Yes"][c]
    df_no = df[df.Churn == "No"][c]
    yes_mean = df_yes.mean()
    no_mean = df_no.mean()
    fig, ax = plt.subplots()
    _, bins, _ = ax.hist([df_yes, df_no], bins=6, stacked=True)
    ax.legend(["Churn - Yes", "Churn - No"])
    ymin, ymax = ax.get_ylim()
    xmin, xmax = ax.get_xlim()
    ax.axvline(yes_mean, color="blue", lw=2)  # yes mean
    ax.axvline(no_mean, color="orangered", lw=2)  # no mean
    ax.text(
        (xmax - xmin) / 2,
        (ymax - ymin) / 2,
        "Delta:\n" + str(round(abs(yes_mean - no_mean), 2)),
        bbox={"facecolor": "white"},
    )
    plt.xlabel('{} vs. {}'.format(c,'Churn'))
    plt.ylabel('Count')
    count_figures = key
    sub_title = '{}_{}_{}'.format(c,'bivariate','stacked histogram')
    title = '{} vs. {}'.format(c,'Churn')
    fname = 'Fig_{}_{}.{}'.format(str(count_figures), sub_title, "png
")
    plt.title(title, fontsize=14, fontweight="bold")
    fig.suptitle(sub_title, fontsize=14)
    plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14, bbox={"fa
cecolor":"orange", "alpha":0.5, "pad":5})
    fig.tight_layout()

    # save file in the course figure's folder
    plt.savefig(os.path.join(figure_folder, fname))
    plt.close()
    print('figure saved at: [{}]'.format(fname))
```

MonthlyCharge_bivariate_stacked histogram

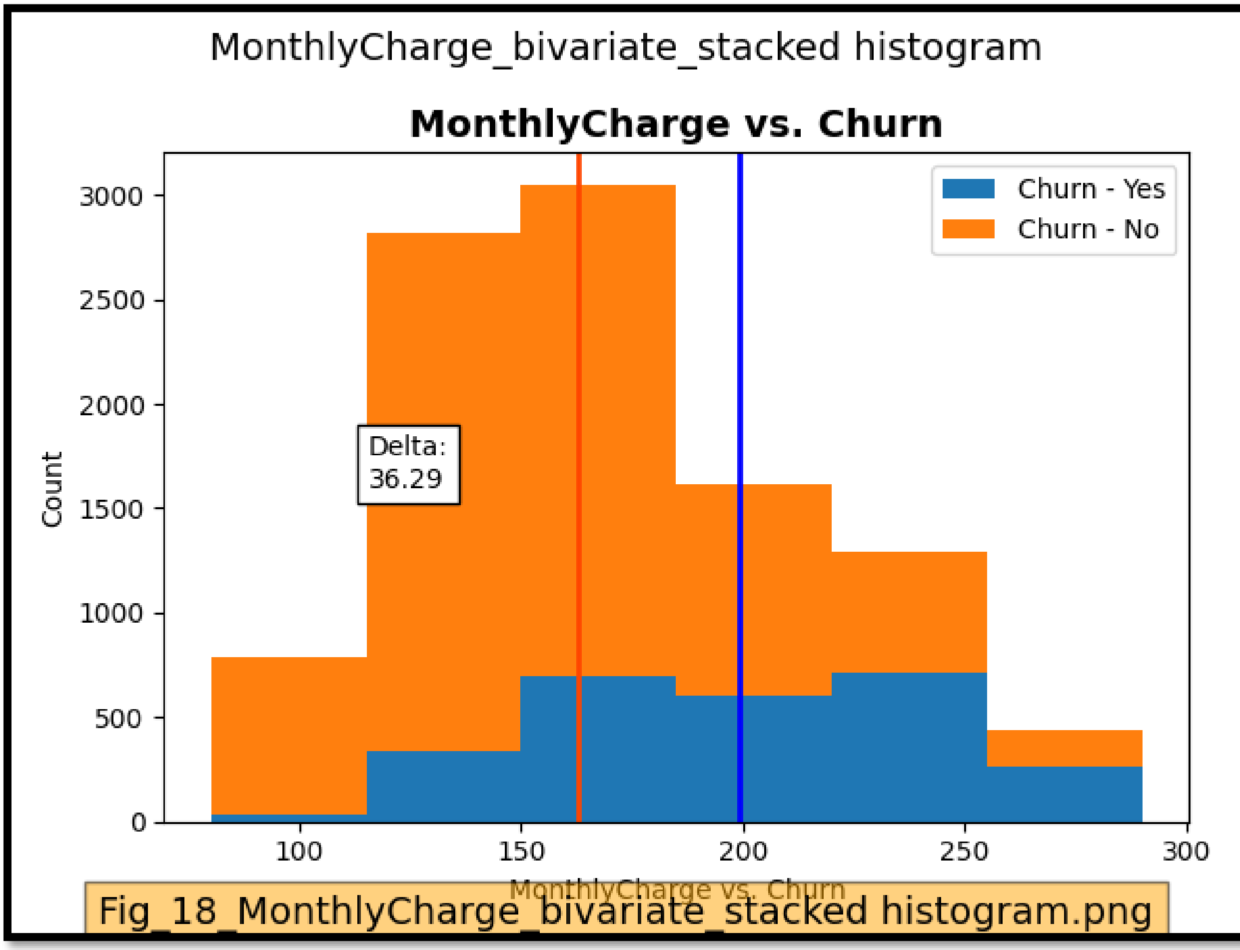


Figure 18 MonthlyCharge_bivariate_stacked histogram

Figure notes: the figure shows a blue line (Median of 'Churn' Yes) and an orange line (Median of 'Churn' No) to show a visualization of the difference between the two sub-populations. The difference between the two means is the Delta.

Here is the data that goes with this plot:

```
############################################
        MonthlyCharge crosstab
############################################
Churn                      No    Yes     All
MonthlyCharge
(79.769, 115.009]         755     35     790
(115.009, 150.039]       2477    338    2815
(150.039, 185.07]        2356    693    3049
(185.07, 220.1]          1009    608    1617
(220.1, 255.13]           579    715    1294
(255.13, 290.16]          174    261     435
All                      7350   2650   10000
```

Bandwidth_GB_Year_bivariate_stacked histogram

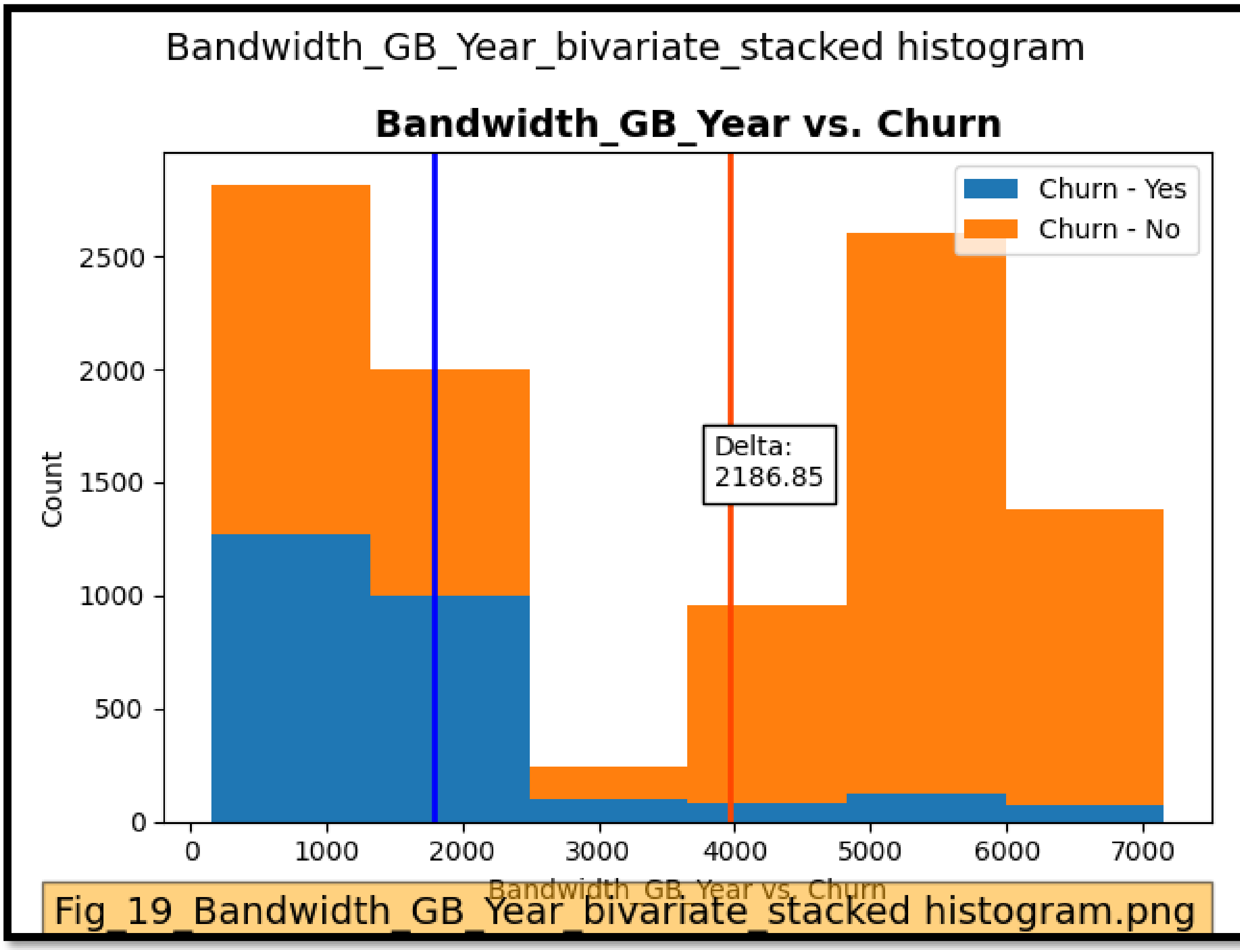


Figure 19 Bandwidth_GB_Year_bivariate_stacked histogram

Figure notes: the figure shows a blue line (Median of 'Churn' Yes) and an orange line (Median of 'Churn' No) to show a visualization of the difference between the two sub-populations. The difference between the two means is the Delta.

Here is the data that goes with this plot:

```
##############################################
        Bandwidth_GB_Year crosstab
##############################################
Churn                       No   Yes    All
Bandwidth_GB_Year
(148.503, 1322.753]       1544  1273   2817
(1322.753, 2489.998]      1000   999   1999
(2489.998, 3657.244]       147    97    244
(3657.244, 4824.49]        874    82    956
(4824.49, 5991.736]       2476   125   2601
(5991.736, 7158.982]      1309    74   1383
All                       7350  2650  10000
```

The next plots are stacked bar plots of two categorical values:

- Fig_23_Gender_bivariate_stacked bar.png
- Fig_23_Gender_bivariate_stacked bar.png

Here is the code:

```
# visualization of selected bivariate data
bivariate = {
    "23": "Gender",
    "24": "InternetService",
}

for key in bivariate:
    c = bivariate[key]
    target = 'Churn'

    # print crosstab table associated with figure
    print(heading_toString(c + " crosstab"))
    print(pd.crosstab(df[c], df[target], margins=True))

    y = DataFrame({"count": df.groupby([c, target]).size()}).reset_in
dex()

    """ each dataframe will look like this:
    Phone Churn  count
0    No    No   1351
1    No   Yes    521
2   Yes    No   5999
3   Yes   Yes   2129
    """

    x = y[c].unique()

    fig, ax = plt.subplots()
    no = y[y[target] == "No"]
    yes = y[y[target] == "Yes"]

    ax.barh(
        x,
        yes["count"],
        height=0.75,
        color="lightgreen",
        label="Churn Yes",
        left=no["count"],
    )
    ax.barh(x, no["count"], height=0.75, color="darkgreen", label="Ch
urn No")
    ax.legend(["Churn - Yes", "Churn - No"])
    plt.xlabel('{} vs. {}'.format(c,'Churn'))
    plt.ylabel('Count')
    count_figures = key
    sub_title = '{}_{}_{}'.format(c,'bivariate','stacked bar')
    title = '{} vs. {}'.format(c,'Churn')
    fname = 'Fig_{}_{}.{}'.format(str(count_figures), sub_title, "png
")
    plt.title(title, fontsize=14, fontweight="bold")
    fig.suptitle(sub_title, fontsize=14)
    plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14, bbox={"fa
cecolor":"orange", "alpha":0.5, "pad":5})
    fig.tight_layout()
```

```
    # save file in the course figure's folder
    plt.savefig(os.path.join(figure_folder, fname))
    plt.close()
    print('figure saved at: [{}]'.format(fname))
```

Gender_bivariate_stacked bar

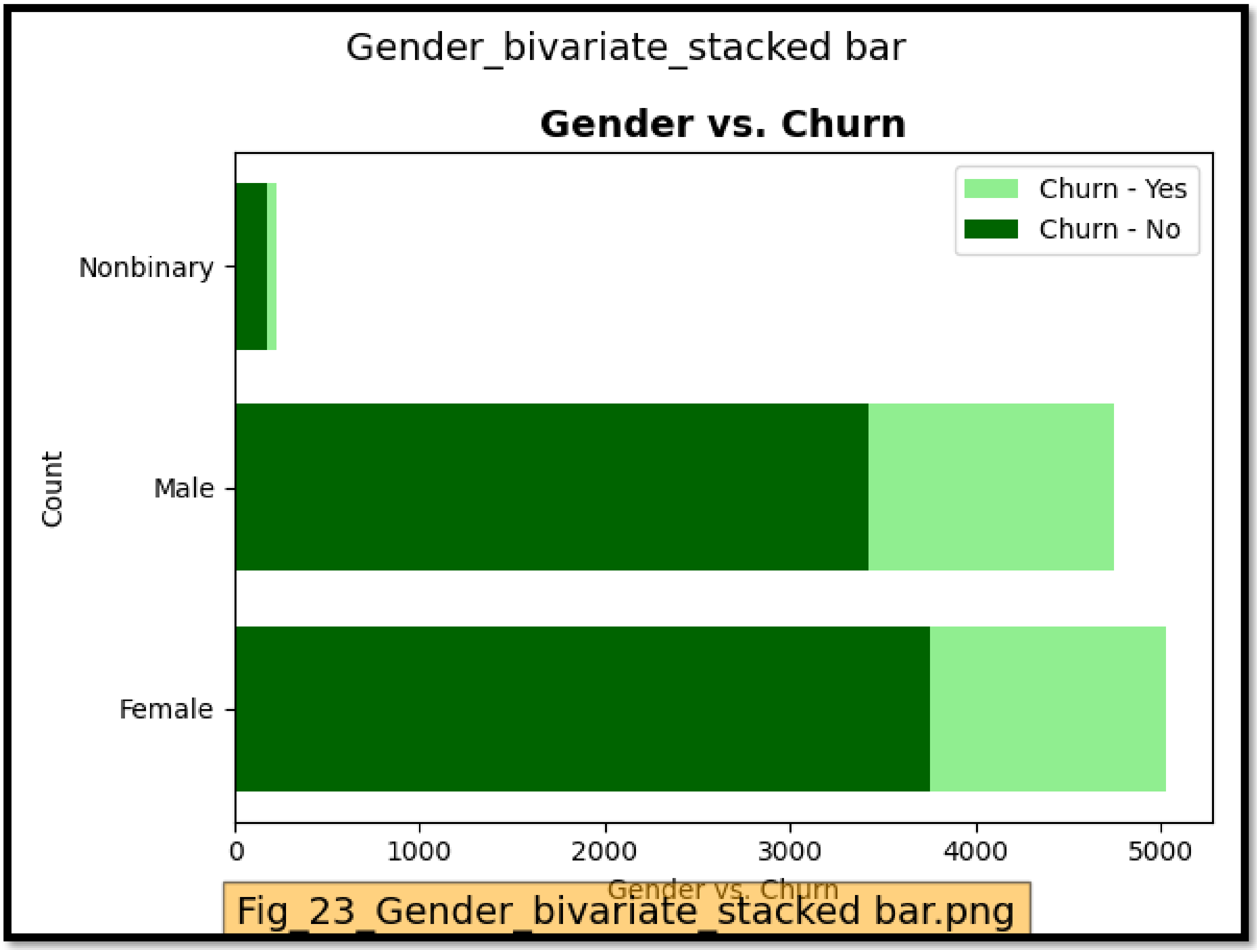


Figure 20 Gender_bivariate_stacked bar.png

Here is the data table for this plot:

```
####################################
        Gender crosstab
####################################
Churn          No   Yes    All
Gender
Female       3753  1272   5025
Male         3425  1319   4744
Nonbinary     172    59    231
All          7350  2650  10000
```

InternetService_bivariate_stacked bar

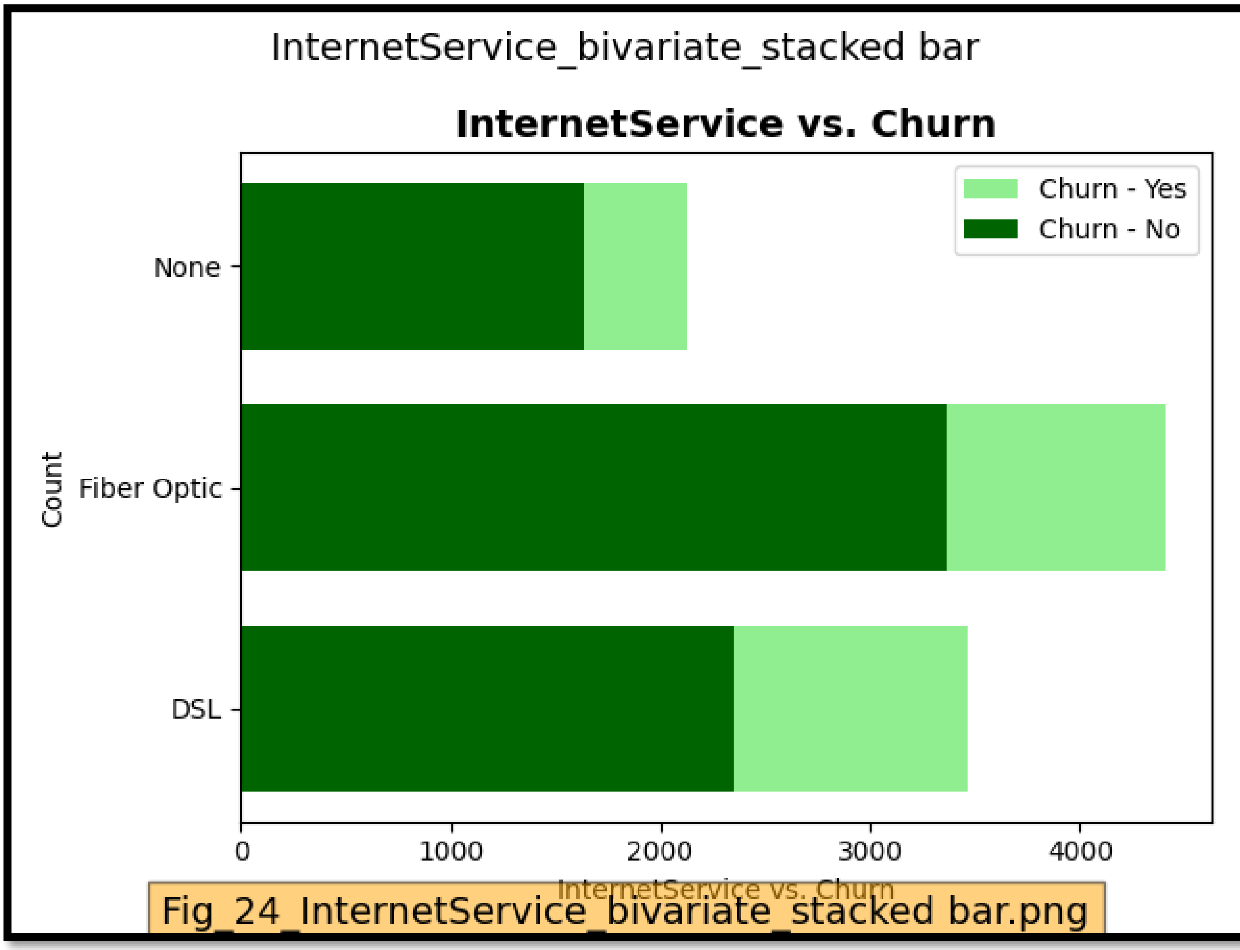


Figure 21 InternetService_bivariate_stacked bar.png

Here is the data table for this plot:

```
############################################
        InternetService crosstab
############################################
Churn                  No   Yes    All
InternetService
DSL                  2349  1114   3463
Fiber Optic          3368  1040   4408
None                 1633   496   2129
All                  7350  2650  10000
```

Part E Summary

Part E1. Discuss the results of the test

The Chi-square hypothesis test was a very strong evidence of a relationship between 'Churn' and 'MonthlyCharge'.

Part E2. Discuss the limitations of your data analysis

- **The test results do not explain the relationship, but only indicate possibility of relationship or association, further research is required.**

Part E3. Recommend a course of action based on your results.

Recommend continued research to find extent of the relationship. Recommend organization consider modifying monthly charge rates for customers.

Part F Video

Part F – Submit video presentation. Video created and .MP4 file is attached to submission.

Part G References

Part G – References

Reference the web sources used to acquire segments of third-party code to support the analysis.

Brown, J. (2018, June 15). *A Gentle Introduction to the Chi-Squared Test for Machine Learning*. Retrieved from Machine Learning Mastery.com: https://machinelearningmastery.com/chi-squared-test-for-machine-learning/

Bruce, P., Bruce, A., & Gedeck, P. (2020). *Practical Statistics for Data Scientists.* O'Reilly Media Inc.

Chi Squared Table. (2021, July 31). Retrieved from Statistics How To: https://www.statisticshowto.com/tables/chi-squared-table-right-tail/

Contingency Tables. (2021, July 31). Retrieved from statsmodels.org: https://www.statsmodels.org/stable/contingency_tables.html

Griffiths, D. (2009). *Head First Statistics.* O'Reilly Media Inc.

Hamel, G. (2021, July 31). *Python for Data 25: Chi-Squared Tests*. Retrieved from kaggle.com: https://www.kaggle.com/hamelg/python-for-data-25-chi-squared-tests

Larose, C. D., & Larose, D. T. (2019). *Data Science Using Python and R.* Wiley.

scipy.stats.chisquare. (2021, July 31). Retrieved from Scipy.org: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.chisquare.html

Sullivan, L. (2021, July 31). *Hypothesis Testing - Chi Squared Test*. Retrieved from sphweb.bumc.be.edu: https://sphweb.bumc.bu.edu/otlt/MPH-Modules/BS/BS704_HypothesisTesting-ChiSquare/BS704_HypothesisTesting-ChiSquare_print.html

(MAT21) https://matplotlib.org/

(MAT21) https://math.meta.stackexchange.com/questions/21841/how-to-type-greater-than-orequal-to-symbols?noredirect=1

(NUM21) https://numpy.org/

(PAN21) https://pandas.pydata.org/docs/pandas.pdf

(PCA21) https://sebastianraschka.com/Articles/2014_pca_step_by_step.html

(PCA21) https://pub.towardsai.net/principal-component-analysis-pca-with-python-examplestutorial-

67a917bae9aa

(PCA21) https://www.districtdatalabs.com/principal-component-analysis-with-python

(PCA21) https://www.edureka.co/blog/principal-component-analysis/

(PCA21) https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-componentanalysis.html

(PCA21) https://www.geeksforgeeks.org/principal-component-analysis-with-python/

(PCA21) https://stackoverflow.com/questions/13224362/principal-component-analysis-pca-inpython12

(PCA21) https://www.machinelearningplus.com/machine-learning/principal-componentsanalysis-pca-better-explained/

(SCI21) https://www.scipy.org/

(STO21) Stoltenber, S. (2021, April 26). Styling a Jupyter Notebook. Retrieved from https://skelouse.github.io/styling_a_jupyter_notebook

(TUK77) Tukey, J. W. (1977). Exploratory Data Analysis. Addison-Wesley Publishing Company. Retrieved from http://www.ru.ac.bd/wp-content/uploads/sites/25/2019/03/102_05_01_Tukey-Exploratory-Data-Analysis-1977.pdf

(WIL19) Wilke, C. O. (2019). Fundamental of Data Visualization: A Primer on Making Informative and Compelling Figures. O'Reilly Media Inc.

(WGU21) WGU.edu (2021, Jun 1). Data Files and Associated Dictionary Files. Retrieved from https://access.wgu.edu/ASP3/aap/content/kgj47f8gj49f8du49d3k.html

Part H Acknowledge Sources

Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

Part I Code

The following is a complete list of all Python code:

```
'''
########################################
    IMPORT REQUIRED PACKAGES
######################################## '''

# imports
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from pandas.core.frame import DataFrame
import scipy.stats as stats
import seaborn as sns
from sklearn import preprocessing
from sklearn.decomposition import PCA
from scipy.stats import chi2_contingency
from scipy.stats import chi2
from IPython.display import Latex
import sys
import os
import datetime
from mm_lib import *

# remove the ellipsis from terminal pandas tables
pd.set_option("display.width", None)
np.set_printoptions(threshold=np.inf)

# reconfigure sys.stdout to utf-8
sys.stdout.reconfigure(encoding="utf-8")

'''
########################################
    STUDENT INFORMATION
######################################## '''

# print student information
print(heading_toString("STUDENT INFORMATION"))
stud_info = {
  "Student": "Mike Mattinson",
  "Student ID": "001980761",
  "Class": "D207 Exploratory Data Analysis",
  "Dataset": "Churn",
  "Submission": "2nd Submission",
  "School": "WGU",
  "Instructor": "Dr. Sewell"
}
for key in stud_info:
    print('{:>14} | {}'.format(key,stud_info[key]))

print("Today is {}".format(datetime.date.today().strftime("%B %d, %Y")))

# print python enivronment
print(heading_toString("PYTHON ENVIRONMENT"))
print("Version: {} located at {}".format(sys.version, sys.executable))

# global variables
count_tables = 0  # keep track of the number of tables generated
```

```
    count_figures = 0  # keep track of the number of figures generated
    course = "d207"
    fig_title_fontsize = 18
    target = "Churn"
    title_str = "CHURN DATA - WGU - D207 - MATTINSON 2021"

    # check and create folder to hold all figures
    figure_folder='figures\\' + course + '\\'
    if not os.path.exists(figure_folder):
            os.makedirs(figure_folder)

    '''
    ########################################
        SETUP DATAFRAME
    ######################################## '''

    # create dataframe
    print(heading_toString("DATAFRAME (DF)"))
    df = pd.read_csv("data\churn_clean.csv")
    print(df[["Customer_id", "City", "Churn", "MonthlyCharge", "Tenure"]].head(4))
    print(df.shape)

    # remove unwanted columns
    print(heading_toString("REMOVE UNWANTED COLUMNS"))

    # remove unwanted columns
    unwanted_columns = ["UID", "Interaction", "Lat", "Lng"]

    for uc in unwanted_columns:
        if uc in df.columns:
            df.drop(columns=uc, inplace=True)
            print("[{}] column removed.".format(uc))

    # show remaining columns
    print("Remaining columns: \n{}".format(df.columns))

    print(heading_toString("CONTINUOUS DATA"))
    print(df.select_dtypes(include="float").info())

    print(heading_toString("INTEGER DATA"))
    print(df.select_dtypes(include="integer").info())

    print(heading_toString("CATEGORICAL DATA"))
    print(df.select_dtypes(include="object").info())

    # output dataframe as .csv table
    print(heading_toString("SAVE CLEAN DATAFRAME"))
    count_tables += 1
    table_title = "churn_clean_data"
    fname = (
        "tables\\" + course + "\\" + "Tab_" + str(count_tables) + "_" + table_title + ".csv"
    )
    print("table saved at: {}".format(fname))
    df.to_csv(fname, index=False, header=True)

    '''
    ########################################
        PART B - CHI-SQUARE TEST
    ######################################## '''

    # CHI-SQUARE TEST
```

```
target = 'Churn'
prob = 0.999
predictor = 'MonthlyCharge' # numerical
print(heading_toString("PART B - CHI-SQUARE INDEPENDENCE TEST - " + target + " vs. " + predictor))
bins = 6

# use library function to perform test
chi_square_analysis(target,predictor,bins,prob,df)

# use library function to print out distribution table
print(heading_toString("CHI-SQUARE DISTRIBUTION TABLE"))
print(chi_square_dist_table(0,0))

'''
########################################
    PART C - UNIVARIATE CATEOGRICAL
######################################## '''

print(heading_toString("PLOT UNIVARIATE CATEGORICAL"))
# box plot of Churn (cat)
#sns.countplot(df['InternetService'])
#plt.show()

# visualization of selected categorical data
univariate_categorical = {
  "1": "Churn",
  "2": "InternetService",
}

for key in univariate_categorical:
    fig, ax = plt.subplots()
    c = univariate_categorical[key]
    sns.countplot(df[c])
    plt.xlabel(c)
    plt.ylabel('Count')
    count_figures = key
    #title = 'Hello world'
    sub_title = '{}_{}_{}'.format(c,'categorical','countplot')
    title = '{}'.format(c)
    fname = 'Fig_{}_{}.{}'.format(str(count_figures), sub_title, "png")
    plt.title(title, fontsize=14, fontweight="bold")
    fig.suptitle(sub_title, fontsize=14)
    plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14, bbox={"facecolor":"orange", "alpha":0.5, "pad":5})
    fig.tight_layout()

    # save file in the course figure's folder
    plt.savefig(os.path.join(figure_folder, fname))
    plt.close()
    print('figure saved at: [{}]'.format(fname))

'''
########################################
    PART C - UNIVARIATE CONTINUOUS
######################################## '''

print(heading_toString("DESCRIPTIVE STATS"))
# print descriptive stats for the dataframe
print(df.select_dtypes(include="float").describe().round(2))
```

```
print(heading_toString("PLOT UNIVARIATE CONTINUOUS"))

# visualization of selected univariate continuous data
univariate_continuous = {
  "3": "MonthlyCharge",
  "4": "Income",
}

for key in univariate_continuous:
    fig, ax = plt.subplots()
    c = univariate_continuous[key]
    plt.scatter(df.index,df[c])
    plt.xlabel(c)
    plt.ylabel('Count')
    count_figures = key
    #title = 'Hello world'
    sub_title = '{}_{}_{}'.format(c,'continuous','scatterplot')
    title = '{}'.format(c)
    fname = 'Fig_{}_{}.{}'.format(str(count_figures), sub_title, "png")
    plt.title(title, fontsize=14, fontweight="bold")
    fig.suptitle(sub_title, fontsize=14)
    plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14, bbox={"facecolor":"orange", "alpha":0.5, "pad":5})
    fig.tight_layout()

    # save file in the course figure's folder
    plt.savefig(os.path.join(figure_folder, fname))
    plt.close()
    print('figure saved at: [{}]'.format(fname))


'''
########################################
    PART D - BIVARIATE
######################################## '''
print(heading_toString("PLOT BIVARIATE COUNTPLOT"))
# visualization of selected bivariate data
bivariate = {
  '11': 'InternetService',
  '12': 'TechSupport',
  '13': 'PaymentMethod',
  '14': 'Tablet',
  '15': 'Gender',
  '16': 'Port_modem',
  '17': 'Techie',
}

for key in bivariate:
    fig, ax = plt.subplots()
    c = bivariate[key]
    sns.countplot(x='Churn', hue=c, data=df, palette='hls')
    plt.xlabel('{} vs. {}'.format(c,'Churn'))
    plt.ylabel('Count')
    count_figures = key
    sub_title = '{}_{}_{}'.format(c,'bivariate','countplot')
    title = '{} vs. {}'.format(c,'Churn')
    fname = 'Fig_{}_{}.{}'.format(str(count_figures), sub_title, "png")
    plt.title(title, fontsize=14, fontweight="bold")
```

```
    fig.suptitle(sub_title, fontsize=14)
    plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14, bbox={"facecolor":"orange", "alpha":0.5, "pad":5})
    fig.tight_layout()

    # save file in the course figure's folder
    plt.savefig(os.path.join(figure_folder, fname))
    plt.close()
    print('figure saved at: [{}]'.format(fname))

print(heading_toString("PLOT BIVARIATE STACKED HISTOGRAM"))
# visualization of selected continous data
bivariate = {
  '18': 'MonthlyCharge',
  '19': 'Bandwidth_GB_Year',
  '20': 'Tenure',
  '21': 'Outage_sec_perweek',
  '22': 'Income',
}

for key in bivariate:
    c = bivariate[key]
    target = 'Churn'

    # print crosstab table associated with figure
    print(heading_toString(c + " crosstab"))
    print(pd.crosstab(pd.cut(df[c], bins = 6), df[target], margins=True))

    df_yes = df[df.Churn == "Yes"][c]
    df_no = df[df.Churn == "No"][c]
    yes_mean = df_yes.mean()
    no_mean = df_no.mean()
    fig, ax = plt.subplots()
    _, bins, _ = ax.hist([df_yes, df_no], bins=6, stacked=True)
    ax.legend(["Churn - Yes", "Churn - No"])
    ymin, ymax = ax.get_ylim()
    xmin, xmax = ax.get_xlim()
    ax.axvline(yes_mean, color="blue", lw=2)  # yes mean
    ax.axvline(no_mean, color="orangered", lw=2)  # no mean
    ax.text(
        (xmax - xmin) / 2,
        (ymax - ymin) / 2,
        "Delta:\n" + str(round(abs(yes_mean - no_mean), 2)),
        bbox={"facecolor": "white"},
    )
    plt.xlabel('{} vs. {}'.format(c,'Churn'))
    plt.ylabel('Count')
    count_figures = key
    sub_title = '{}_{}_{}'.format(c,'bivariate','stacked histogram')
    title = '{} vs. {}'.format(c,'Churn')
    fname = 'Fig_{}_{}.{}'.format(str(count_figures), sub_title, "png")
    plt.title(title, fontsize=14, fontweight="bold")
    fig.suptitle(sub_title, fontsize=14)
    plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14, bbox={"facecolor":"orange", "alpha":0.5, "pad":5})
    fig.tight_layout()

```

```python
    # save file in the course figure's folder
    plt.savefig(os.path.join(figure_folder, fname))
    plt.close()
    print('figure saved at: [{}]'.format(fname))

print(heading_toString("PLOT BIVARIATE STACKED BAR"))
# visualization of selected bivariate data
bivariate = {
    "23": "Gender",
    "24": "InternetService",
}

for key in bivariate:
    c = bivariate[key]
    target = 'Churn'

    # print crosstab table associated with figure
    print(heading_toString(c + " crosstab"))
    print(pd.crosstab(df[c], df[target], margins=True))

    y = DataFrame({"count": df.groupby([c, target]).size()}).reset_index()

    """ each dataframe will look like this:
    Phone Churn  count
0    No     No    1351
1    No    Yes     521
2   Yes     No    5999
3   Yes    Yes    2129
    """

    x = y[c].unique()

    fig, ax = plt.subplots()
    no = y[y[target] == "No"]
    yes = y[y[target] == "Yes"]

    ax.barh(
        x,
        yes["count"],
        height=0.75,
        color="lightgreen",
        label="Churn Yes",
        left=no["count"],
    )
    ax.barh(x, no["count"], height=0.75, color="darkgreen", label="Churn No")
    ax.legend(["Churn - Yes", "Churn - No"])
    plt.xlabel('{} vs. {}'.format(c,'Churn'))
    plt.ylabel('Count')
    count_figures = key
    sub_title = '{}_{}_{}'.format(c,'bivariate','stacked bar')
    title = '{} vs. {}'.format(c,'Churn')
    fname = 'Fig_{}_{}.{}'.format(str(count_figures), sub_title, "png")
    plt.title(title, fontsize=14, fontweight="bold")
    fig.suptitle(sub_title, fontsize=14)
    plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14, bbox={"facecolor":"orange", "alpha":0.5, "pad":5})
    fig.tight_layout()

    # save file in the course figure's folder
```

```
        plt.savefig(os.path.join(figure_folder, fname))
        plt.close()
        print('figure saved at: [{}]'.format(fname))
```

Here is the mm_lib.py file:

```
# consolidate all of my data analytics functions
# to be reused in the diff courses.
# Author: Mike Mattinson (MM)
# Date: June 29, 2021
#
# To use, place following import in each course file:
# from mm import *

def chi_square_analysis(target,predictor,bins,prob,df):
    ''' Calculate and display results of chi-square independence test

    Arg:
       target - target variable
       predictor - predictor variable
       bins - number of bins, if numerical, 0 if categorical
       prob - probability
       df - dataframe

    Return: outputs analysis to terminal.

    '''
    import pandas as pd
    from scipy.stats import chi2_contingency
    from scipy.stats import chi2
    if bins == 0:
        # variable is categorical
        contingency= pd.crosstab(df[target], df[predictor])
    else:
        # variable is numerical
        mc_groups =pd.cut(df[predictor], bins=bins)
        contingency= pd.crosstab(df[target], mc_groups)

    # print out the contingency table
    print(contingency.T)

    stat, p, dof, expected = chi2_contingency(contingency)
    alpha = 1 - prob # significance level
    critical = chi2.ppf(prob, dof)

    # interpret test-statistic
    if abs(stat) >= critical:
        print('At an alpha level of {}, the critical value is {}.'.format(round(alpha,3),round(critical,3)))
        print('chi^2 = {} > {}, therefore, variables are dependent (reject H0).'.format(round(stat,3),round(critical,3)))
        #print('Also, we will need to perform additional analysis to determine extent of the relationship.')
    else:
        print('chi^2 = {} <= {}, therefore, variables are independent (fail to reject H0).'.format(round(stat,3),round(critical,3)))

```

```
        print(' p-value: {:>8.3f}'.format(p))
        print('     dof: {:>8d}'.format(dof))
        print('   alpha: {:>8.3f}'.format(alpha))
        print('    stat: {:>8.3f}'.format(stat))
        print('critical: {:>8.3f}'.format(critical))

    def chi_square_dist_table(dof,prob):
        '''create chi square distribution table

        Args:
            dof - degree of freedom
            prob - significance level (alpha)

        Returns:
            A string representation of the chi-square
            lower-left distribution table.

        '''
        from scipy.stats import chi2
        prob_range = [ .10, 0.05, 0.025, 0.01, 0.001]
        dof_range = range(1,20)
        output = 'Lower-tail critical values of chi-Square distribution:\n'.format()

        # header row
        output += '{:>4}'.format('dof') # blank space for first col
        for p in prob_range:
            output += '{:>8.3f}'.format(p)
        output += '\n' +'-' * 47

        # each row
        for d in dof_range:
            output += '\n{:4d}'.format(d)
            for p in prob_range:
                temp = chi2.isf(p, d)
                if d == dof and p == prob:
                    output += '{:>8.2f}'.format(temp)
                else:
                    output += '{:>8.2f}'.format(temp)

        return output

    # heading_to_str
    def heading_toString(heading):
        """Create a heading string.

        Args:
            heading: heading title.

        Returns:
            The formatted heading as string.

        """
        how_wide = len(heading) + 20
        underline_str = "#" * how_wide

        # unicode support
        # must use sys.stdout.reconfigure(encoding="utf-8")
        overline_char = "#" # "\u203e"

        overline_str = overline_char * how_wide
        temp = "\n\n{}".format(underline_str)
```

```
    temp += "\n\t{}".format(heading)
    temp += "\n{}".format(overline_str)
    return temp

# mitigate missing data
def mitigate_missing_data(source, column, new_value, inplace=True):
    """Replace missing data with new_value and print message.

    Args:
        source: source dataframe
        column: column to be adjusted.
        new_value: value to be inserted
        inplace (Default: True): change dataframe inplace

    Returns:
        The success/failure message.

    """
    try:
        source[column].fillna(new_value, inplace=inplace)
        print("[{}] missing data set to [{}] successfully.".format(column, new_value))

    except:
        print("[{}] unable changes.".format(column))

def custom_lineplot(ax, x, y, error, xlims, ylims, color="red"):
    """Customized line plot with error bars."""

    ax.errorbar(
        x,
        y,
        yerr=error,
        color=color,
        ls="--",
        marker="o",
        capsize=5,
        capthick=1,
        ecolor="black",
    )

    ax.set_xlim(xlims)
    ax.set_ylim(ylims)

    return ax

def custom_scatterplot(ax, x, y, error, xlims, ylims, color="green", markerscale=100):
    """Customized scatter plot where marker size is proportional to error measure."""

    markersize = error * markerscale

    ax.scatter(x, y, color=color, marker="o", s=markersize, alpha=0.5)

    ax.set_xlim(xlims)
    ax.set_ylim(ylims)

    return ax

```

```
import numpy as np

def custom_barchart(
    ax, x, y, error, xlims, ylims, error_kw, color="lightblue", width=0.75
):
    """Customized bar chart with positive error bars only."""

    error = [np.zeros(len(error)), error]

    ax.bar(
        x, y, color=color, width=width, yerr=error, error_kw=error_kw, align="center"
    )

    ax.set_xlim(xlims)
    ax.set_ylim(ylims)

    return ax

def custom_boxplot(ax, x, y, xlims, ylims, mediancolor="magenta"):
    """Customized boxplot with solid black lines for box, whiskers, caps, and outliers."""

    medianprops = {"color": mediancolor, "linewidth": 2}
    boxprops = {"color": "black", "linestyle": "-"}
    whiskerprops = {"color": "black", "linestyle": "-"}
    capprops = {"color": "black", "linestyle": "-"}
    flierprops = {"color": "black", "marker": "x"}

    ax.boxplot(
        y,
        positions=x,
        medianprops=medianprops,
        boxprops=boxprops,
        whiskerprops=whiskerprops,
        capprops=capprops,
        flierprops=flierprops,
    )

    ax.set_xlim(xlims)
    ax.set_ylim(ylims)

    return ax

def fig_title_toString(title, fig_count, fig_type):
    return "Fig_" + str(fig_count) + "_" + title + "_" + fig_type

def table_title_toString(title, table_count):
    return "Tab_" + str(table_count) + "_" + title

def fig_fname_toString(folder, title, filetype):
    return folder + title + "." + filetype

def table_fname_toString(course, title, filetype):
    return "tables\\" + course + "\\" + title + "." + filetype

def numeric_data_toBoxplot():
    return True
```

Part J Output

The following is a complete list of all terminal output:

```
#######################################
        STUDENT INFORMATION
#######################################
       Student | Mike Mattinson
    Student ID | 001980761
         Class | D207 Exploratory Data Analysis
       Dataset | Churn
    Submission | 2nd Submission
        School | WGU
    Instructor | Dr. Sewell
Today is August 03, 2021


######################################
        PYTHON ENVIRONMENT
######################################
Version: 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21)
[MSC v.1929 64 bit (AMD64)] located at P:\workspace-
wgu\wgu_local_py\wgu_venu\Scripts\python.exe


##################################
        DATAFRAME (DF)
##################################
  Customer_id         City Churn MonthlyCharge     Tenure
0     K409198  Point Baker    No    172.455519   6.795513
1     S120509  West Branch   Yes    242.632554   1.156681
2     K191035      Yamhill    No    159.947583  15.754144
3      D90850      Del Mar    No    119.956840  17.087227
(10000, 50)


#########################################
        REMOVE UNWANTED COLUMNS
#########################################
[UID] column removed.
[Interaction] column removed.
[Lat] column removed.
[Lng] column removed.
Remaining columns:
Index(['CaseOrder', 'Customer_id', 'City', 'State', 'County',
'Zip',
       'Population', 'Area', 'TimeZone', 'Job', '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', 'Item1',
'Item2', 'Item3',
       'Item4', 'Item5', 'Item6', 'Item7', 'Item8'],
      dtype='object')
```

```
###################################
        CONTINUOUS DATA
###################################
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Income              10000 non-null  float64
 1   Outage_sec_perweek  10000 non-null  float64
 2   Tenure              10000 non-null  float64
 3   MonthlyCharge       10000 non-null  float64
 4   Bandwidth_GB_Year   10000 non-null  float64
dtypes: float64(5)
memory usage: 390.8 KB
None


################################
        INTEGER DATA
################################
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   CaseOrder             10000 non-null  int64
 1   Zip                   10000 non-null  int64
 2   Population            10000 non-null  int64
 3   Children              10000 non-null  int64
 4   Age                   10000 non-null  int64
 5   Email                 10000 non-null  int64
 6   Contacts              10000 non-null  int64
 7   Yearly_equip_failure  10000 non-null  int64
 8   Item1                 10000 non-null  int64
 9   Item2                 10000 non-null  int64
 10  Item3                 10000 non-null  int64
 11  Item4                 10000 non-null  int64
 12  Item5                 10000 non-null  int64
 13  Item6                 10000 non-null  int64
 14  Item7                 10000 non-null  int64
 15  Item8                 10000 non-null  int64
dtypes: int64(16)
memory usage: 1.2 MB
None


####################################
        CATEGORICAL DATA
####################################
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Customer_id       10000 non-null  object
 1   City              10000 non-null  object
 2   State             10000 non-null  object
 3   County            10000 non-null  object
 4   Area              10000 non-null  object
```

```
 5   TimeZone          10000 non-null  object
 6   Job               10000 non-null  object
 7   Marital           10000 non-null  object
 8   Gender            10000 non-null  object
 9   Churn             10000 non-null  object
 10  Techie            10000 non-null  object
 11  Contract          10000 non-null  object
 12  Port_modem        10000 non-null  object
 13  Tablet            10000 non-null  object
 14  InternetService   10000 non-null  object
 15  Phone             10000 non-null  object
 16  Multiple          10000 non-null  object
 17  OnlineSecurity    10000 non-null  object
 18  OnlineBackup      10000 non-null  object
 19  DeviceProtection  10000 non-null  object
 20  TechSupport       10000 non-null  object
 21  StreamingTV       10000 non-null  object
 22  StreamingMovies   10000 non-null  object
 23  PaperlessBilling  10000 non-null  object
 24  PaymentMethod     10000 non-null  object
dtypes: object(25)
memory usage: 1.9+ MB
None


########################################
        SAVE CLEAN DATAFRAME
########################################
table saved at: tables\d207\Tab_1_churn_clean_data.csv


##############################################################
######################
        PART B - CHI-SQUARE INDEPENDENCE TEST - Churn vs.
MonthlyCharge
##############################################################
######################
Churn                 No  Yes
MonthlyCharge
(79.769, 115.009]    755   35
(115.009, 150.039]  2477  338
(150.039, 185.07]   2356  693
(185.07, 220.1]     1009  608
(220.1, 255.13]      579  715
(255.13, 290.16]     174  261
At an alpha level of 0.001, the critical value is 20.515.
chi^2 = 1425.642 > 20.515, therefore, variables are dependent
(reject H0).
 p-value:    0.000
     dof:        5
   alpha:    0.001
    stat: 1425.642
critical:   20.515


#################################################
        CHI-SQUARE DISTRIBUTION TABLE
#################################################
Lower-tail critical values of chi-Square distribution:
 dof   0.100   0.050   0.025   0.010   0.001
-----------------------------------------------
   1    2.71    3.84    5.02    6.63   10.83
```

```
   2    4.61     5.99     7.38     9.21    13.82
   3    6.25     7.81     9.35    11.34    16.27
   4    7.78     9.49    11.14    13.28    18.47
   5    9.24    11.07    12.83    15.09    20.52
   6   10.64    12.59    14.45    16.81    22.46
   7   12.02    14.07    16.01    18.48    24.32
   8   13.36    15.51    17.53    20.09    26.12
   9   14.68    16.92    19.02    21.67    27.88
  10   15.99    18.31    20.48    23.21    29.59
  11   17.28    19.68    21.92    24.72    31.26
  12   18.55    21.03    23.34    26.22    32.91
  13   19.81    22.36    24.74    27.69    34.53
  14   21.06    23.68    26.12    29.14    36.12
  15   22.31    25.00    27.49    30.58    37.70
  16   23.54    26.30    28.85    32.00    39.25
  17   24.77    27.59    30.19    33.41    40.79
  18   25.99    28.87    31.53    34.81    42.31
  19   27.20    30.14    32.85    36.19    43.82


###############################################
        PLOT UNIVARIATE CATEGORICAL
###############################################
P:\workspace-wgu\wgu_local_py\wgu_venu\lib\site-
packages\seaborn\_decorators.py:36: FutureWarning: Pass the
following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
figure saved at: [Fig_1_Churn_categorical_countplot.png]
P:\workspace-wgu\wgu_local_py\wgu_venu\lib\site-
packages\seaborn\_decorators.py:36: FutureWarning: Pass the
following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
figure saved at:
[Fig_2_InternetService_categorical_countplot.png]


#####################################
        DESCRIPTIVE STATS
#####################################
          Income  Outage_sec_perweek    Tenure  MonthlyCharge
Bandwidth_GB_Year
count   10000.00            10000.00  10000.00       10000.00
10000.00
mean    39806.93               10.00     34.53         172.62
3392.34
std     28199.92                2.98     26.44          42.94
2185.29
min       348.67                0.10      1.00          79.98
155.51
25%     19224.72                8.02      7.92         139.98
1236.47
50%     33170.60               10.02     35.43         167.48
3279.54
75%     53246.17               11.97     61.48         200.73
5586.14
```

```
max    258900.70                 21.21     72.00          290.16
7158.98


#############################################
        PLOT UNIVARIATE CONTINUOUS
#############################################
figure saved at:
[Fig_3_MonthlyCharge_continuous_scatterplot.png]
figure saved at: [Fig_4_Income_continuous_scatterplot.png]


###########################################
        PLOT BIVARIATE COUNTPLOT
###########################################
figure saved at:
[Fig_11_InternetService_bivariate_countplot.png]
figure saved at: [Fig_12_TechSupport_bivariate_countplot.png]
figure saved at:
[Fig_13_PaymentMethod_bivariate_countplot.png]
figure saved at: [Fig_14_Tablet_bivariate_countplot.png]
figure saved at: [Fig_15_Gender_bivariate_countplot.png]
figure saved at: [Fig_16_Port_modem_bivariate_countplot.png]
figure saved at: [Fig_17_Techie_bivariate_countplot.png]


###################################################
        PLOT BIVARIATE STACKED HISTOGRAM
###################################################


#########################################
        MonthlyCharge crosstab
#########################################
Churn                    No   Yes    All
MonthlyCharge
(79.769, 115.009]       755    35    790
(115.009, 150.039]     2477   338   2815
(150.039, 185.07]      2356   693   3049
(185.07, 220.1]        1009   608   1617
(220.1, 255.13]         579   715   1294
(255.13, 290.16]        174   261    435
All                    7350  2650  10000
figure saved at: [Fig_18_MonthlyCharge_bivariate_stacked
histogram.png]


#############################################
        Bandwidth_GB_Year crosstab
#############################################
Churn                      No   Yes    All
Bandwidth_GB_Year
(148.503, 1322.753]      1544  1273   2817
(1322.753, 2489.998]     1000   999   1999
(2489.998, 3657.244]      147    97    244
(3657.244, 4824.49]       874    82    956
(4824.49, 5991.736]      2476   125   2601
(5991.736, 7158.982]     1309    74   1383
All                      7350  2650  10000
figure saved at: [Fig_19_Bandwidth_GB_Year_bivariate_stacked
histogram.png]
```

```
###################################
        Tenure crosstab
###################################
Churn                  No   Yes    All
Tenure
(0.929, 12.833]      1876  1924   3800
(12.833, 24.667]      692   418   1110
(24.667, 36.5]         70    28     98
(36.5, 48.333]        511    83    594
(48.333, 60.166]     1526   105   1631
(60.166, 71.999]     2675    92   2767
All                  7350  2650  10000
figure saved at: [Fig_20_Tenure_bivariate_stacked
histogram.png]


###############################################
        Outage_sec_perweek crosstab
###############################################
Churn                    No   Yes    All
Outage_sec_perweek
(0.0786, 3.618]         132    43    175
(3.618, 7.136]         1091   397   1488
(7.136, 10.653]        3086  1137   4223
(10.653, 14.171]       2447   855   3302
(14.171, 17.689]        557   200    757
(17.689, 21.207]         37    18     55
All                    7350  2650  10000
figure saved at: [Fig_21_Outage_sec_perweek_bivariate_stacked
histogram.png]


###################################
        Income crosstab
###################################
Churn                         No   Yes    All
Income
(90.118, 43440.675]         4739  1682   6421
(43440.675, 86532.68]       2121   792   2913
(86532.68, 129624.685]       411   138    549
(129624.685, 172716.69]       70    34    104
(172716.69, 215808.695]        5     4      9
(215808.695, 258900.7]         4     0      4
All                         7350  2650  10000
figure saved at: [Fig_22_Income_bivariate_stacked
histogram.png]


##############################################
        PLOT BIVARIATE STACKED BAR
##############################################


###################################
        Gender crosstab
###################################
Churn         No   Yes    All
Gender
Female      3753  1272   5025
Male        3425  1319   4744
Nonbinary    172    59    231
```

```
All          7350  2650  10000
figure saved at: [Fig_23_Gender_bivariate_stacked bar.png]


############################################
        InternetService crosstab
############################################
Churn              No   Yes    All
InternetService
DSL              2349  1114   3463
Fiber Optic      3368  1040   4408
None             1633   496   2129
All              7350  2650  10000
figure saved at: [Fig_24_InternetService_bivariate_stacked
bar.png]
```

Part K Video Script

Hello, my name is Mike Mattinson.

This is a video walk-through of my 2nd submission of the D207 final assessment.

The D207 class is “Exploratory Data Analysis”, WGU.

My instructor is Dr. Sewell.

I selected the ‘Churn’ dataset.

All of the code was written in Python and executed in Visual Studio Code.

The version of Python is 3.9.6.

The primary list of Python packages used:

- Pandas
- Numpy
- Matplotlib

I also created some helper functions located in the mm_lib.py file.

Lines 64-65: Used to create a folder in the working directory to contain the plot figures generated by the code. Some of the data was removed (Lines 79-91)

Lines 75: Used pandas to create dataframe from the provided .csv data file.

Lines 93-97: Created a list of numerical data.

Lines 99-100: Created a list of categorical data.

Lines 113-129: Part B – Chi-Square Independence test. The analysis code is located in the mm_lib.py file. The code performs the independence test and then uses logic to print out the results of the test.

Lines 132-166: Part C – The first section is the code to generate the univariate categorical plots.

Lines 169-204: Part C – The second section is the code to generate the univariate continuous plots.

Lines 207-348: Part D – The bivariate plots are executed by creating three different types of plots.

All code is executed without error.

Open file explore and show the plot files.

This concludes my presentation.