In [1]:

```
import numpy as np
import pandas as pd
import sklearn
import seaborn as sns
%matplotlib inline
from IPython.display import Image
import matplotlib as mlp
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
from matplotlib import rcParams

import warnings
import pickle
pd.set_option('display.max_columns', None)
pd.set_option("max_rows", 1000)
pd.options.display.float_format = lambda x : '{:.0f}'.format(x) if x> 2 else '{:,.2f}'
df = pd.read_csv("./data/WA_Fn-UseC_-Telco-Customer-Churn.csv")
df_Seg_Metadata = pd.read_excel("./data/CustomerSegmentMetadata.xlsx")
```

## Table of Contents



# Functions To Create Visuals

In [2]:

```
# Graph 1 - Loyal vs. Churn Summary

def ShowChurnVsNoChurnSummary_Counts():
    '''Creates Two Summary Graphs, Uses Data Created in Exploratory Part of Notebook'
    df_churnCounts = df_wD["Churn"].value_counts()
    df_churnpercents = df_wD["Churn"].value_counts(normalize=True)

    df_churnCounts = df_churnCounts.rename(index={0: 'Loyal', 1:'Churn'})
    df_churnpercents = df_churnpercents.rename(index={0: 'Loyal', 1:'Churn'})

    LoyalCnt = int(df_churnCounts[:2][0])
    ChurnCnt = int(df_churnCounts[:2][1])

    Loyalpercent = round(df_churnpercents[:2][0],2)
    Churnpercent = round(df_churnpercents[:2][1],2)

    fig, ax = plt.subplots(1, 2, figsize=(15, 4))
    sns.set(font_scale=2)
    sns.set_style("whitegrid")
    sns.barplot(x =df_churnCounts.index, y = df_churnCounts, ax=ax[0]);
    ax[0].annotate('{:.0f}'.format(ChurnCnt), (1,500));
    ax[0].annotate('{:.0f}'.format(LoyalCnt), (0,2900));
    ax[0].axes.xaxis.set_visible(True)
    ax[0].set_ylabel("Count", size=15)

    for tick in ax[0].axes.xaxis.get_major_ticks():
        tick.label.set_fontsize(20)

    for tick in ax[0].axes.yaxis.get_major_ticks():
        tick.label.set_fontsize(20)

    sns.barplot(x =df_churnCounts.index, y = df_churnCounts, ax=ax[1]);
    ax[1].annotate('{:.0f}%'.format(100* Churnpercent), (1,500));
    ax[1].annotate('{:.0f}%'.format(100* Loyalpercent), (0,2900));
    ax[1].axes.xaxis.set_visible(True)
    ax[1].axes.yaxis.set_visible(False)
    ax[1].set_ylabel("Percent", size=15, color="black")
    for tick in ax[1].axes.xaxis.get_major_ticks():
        tick.label.set_fontsize(20)

    for tick in ax[1].axes.yaxis.get_major_ticks():
        tick.label.set_fontsize(20)

    plt.suptitle('Loyal vs. Churn Counts', fontsize = 30)

    plt.tight_layout(pad=4.0)
    plt.show();

#Graph 2 Review Counts by features
def reviewCategoricalCounts ():
    '''Creates Count Plots of all Features within the Dataset, Takes Data from Frames
    warnings.filterwarnings("ignore")
    fig, ax = plt.subplots(figsize=(13,50))

    sns.set(rc={'axes.facecolor':'white', 'figure.facecolor':'white'})
    total = float(len(cats_w_Churn))
    for index, column in enumerate(cats_w_Churn):
        if column != 'Churn':
            ax = plt.subplot(20, 3, index+1, facecolor="white")
            ax = sns.countplot(x=column, hue="Churn", data=cats_w_Churn)
            GraphTitle = df_Seg_Metadata[df_Seg_Metadata["group"]==column]["Title"].va
```

```
            ax.set_title(GraphTitle, size=20)

            ax.set_xlabel('')
            ax.set_ylabel('')
            ax.get_legend().set_visible(False)
            plt.xticks(rotation = 45)
            ax.axes.yaxis.set_visible(False)

            for tick in ax.axes.yaxis.get_major_ticks():
                tick.label.set_fontsize(25)

            for tick in ax.axes.xaxis.get_major_ticks():
                tick.label.set_fontsize(20)

            for p in ax.patches:
                percentage = '{:.0f}%'.format(100 * p.get_height()/total)
                x = p.get_x() + p.get_width()
                y = p.get_height()
                ax.annotate(percentage, (x, y),ha='center', size=15)

    fig.tight_layout()
    plt.show();
#No Churn vs. Churn By Counts Above ____________________________________________

#No Churn vs. Churn By Revenue Below ____________________________________________


# Graphs 3 & 4 - Summary No Churn vs. Churn By Revenue
def showPercentSummary(CountofGroupsToBeGraphed,plot_df,suptitle):
    '''Creates Summary Graphs, Takes in # of graphs to be created, Supporting Data nee

    sns.set(rc={"font.style":"normal",
            "axes.facecolor":"white",
            "figure.facecolor":"white"})

    #Caluculate number of columns and rows for the sub plots
    if CountofGroupsToBeGraphed == 1:
        nRows = 1
        nCols = 1
        figsize=(5,5)
    fig, axes = plt.subplots(nrows=nRows, ncols=nCols, figsize=figsize)
    fig.subplots_adjust(top=8)

    if CountofGroupsToBeGraphed == 1:
        i = axes
        g = plot_df
        ax = axes
        custom_stacked_Summary(g, ax)
        ax.legend().set_visible(True)
        L=plt.legend(loc=(.9,0.5), prop={'size':30})
        L.get_texts()[0].set_text('Loyal')
        L.get_texts()[1].set_text('Churn')
        plt.setp(plt.gca().get_legend().get_texts(), fontsize='20') #legend 'list' for
        ax.set_xticklabels([])

    fig.suptitle(suptitle, fontsize = 30, y=1.05)
    rotation='vertical'
    plt.tight_layout()
    plt.show()

# Graphs 3 & 4 - Called within the above showPercentSummary() function
def custom_stacked_Summary(cross_tab_df, ax):
```

```
    '''Supporting Function Called by showPercentSummary() to create plots, takes in da

    plot_df = cross_tab_df
    p = plot_df.plot(kind="bar", stacked=False, ax = ax)
    p.yaxis.set_major_formatter(mtick.PercentFormatter(1))
    ax.tick_params(labelrotation=0)
    ax.xaxis.label.set_visible(False)
    ax.set_xlabel('')

    rcParams['axes.titlepad'] = 20

    for tick in ax.axes.yaxis.get_major_ticks():
        tick.label.set_fontsize(15)

    for pa in ax.patches:
        percentage = '{:.0f}%'.format(100 * pa.get_height())
        x = pa.get_x() + pa.get_width()
        y = pa.get_height()
        ax.annotate(percentage, (x, y),ha='center',size=25)

    return p

def showPercentRevenueByFeature():
    '''Creates subplots, uses data created a few steps prior to calling this function
    g_dfs = df_all_Percents.groupby("group")
    sns.set(rc={"font.style":"normal",
            "axes.facecolor":"white",
            "figure.facecolor":"white"})

    fig, axes = plt.subplots(nrows=6, ncols=int(len(g_dfs)/7)+1, figsize=(15,15))
    fig.subplots_adjust(top=8)

    # BUILD PLOTS ACROSS LEVELS
    for ax, (i,g) in zip(axes.ravel(), sorted(g_dfs)):
        custom_stacked_barplotByFeature(i, g, ax)

    fig.suptitle('% Revenue By:', fontsize = 30, y=1.03)
    rotation='vertical'
    plt.tight_layout()
    plt.show()



def custom_stacked_barplotByFeature(t, sub_df, ax):
    '''Supporting function, called by showPercentRevenueByFeature only, takes in data
    plot_df = pd.crosstab(index=sub_df["shortDesc"], columns=sub_df['Churn'],
                          values=sub_df['MnthlyRev%'], aggfunc=sum)

    ttle = sub_df["Title"][0:1].to_list()[0]
    p = plot_df.plot(kind="bar", stacked=False, ax = ax, title=ttle)
    p.yaxis.set_major_formatter(mtick.PercentFormatter(1))
    ax.tick_params(labelrotation=0)
    ax.xaxis.label.set_visible(False)
    ax.legend().set_visible(False)

    rcParams['axes.titlepad'] = 20

    for tick in ax.axes.yaxis.get_major_ticks():
        tick.label.set_fontsize(15)

    for pa in ax.patches:
        percentage = '{:.0f}%'.format(100 * pa.get_height())
```

```
        x = pa.get_x() + pa.get_width()
        y = pa.get_height()
        ax.annotate(percentage, (x, y),ha='center')

    return p

#No Churn vs. Churn By Revenue Ends ____________________________________________

#No Churn vs. Churn By Continous ____________________________________________

def ShowChurnNoChurnByContinousFeatures ():
    '''Creates Box Plots of the Continous Features Contained within the Dataset, Data
    #Get List of Categorical and Continueous features
    cont_features = [col for col in df.columns if df[col].dtype in [np.float64, np.int
    cat_features = [col for col in df.columns if df[col].dtype in [np.object]]

#     # Show/ look at numerical columns by churn and non churn
    df_IQR = pd.DataFrame(columns=["25%","75%"])
    warnings.filterwarnings("ignore")

    df_review_featuresByChurnNotChurn = df[cont_features]
    df_review_featuresByChurnNotChurn["Churn"] = df["Churn"]
    plt.figure(figsize=(15,7), facecolor="white")
#     sns.set(font_scale=4)
    sns.set_style("whitegrid")


    df_review_featuresByChurnNotChurn['Churnstr'] = df_review_featuresByChurnNotChurn|
    df_review_featuresByChurnNotChurn['Churnstr'] = df_review_featuresByChurnNotChurn|
    df_review_featuresByChurnNotChurn['Churnstr'].replace(to_replace="1", value='Churn
    df_review_featuresByChurnNotChurn['Churnstr'].replace(to_replace="0", value='Loya

    for index, column in enumerate(df_review_featuresByChurnNotChurn):

        if column != "Churnstr" and column != 'Churn':
            ax = plt.subplot(2, 3, index+1)
            medians = round(df_review_featuresByChurnNotChurn.groupby(['Churn'])[colu
            Q1 = df_review_featuresByChurnNotChurn.groupby(['Churnstr'])[column].quant
            Q3 = df_review_featuresByChurnNotChurn.groupby(['Churnstr'])[column].quant
            ttl = pd.concat([Q1, Q3], axis=1)
            ttl.columns = ["25%","75%"]
            ttl.index = [f'{column} - {Q1.index[0]}', f'{column}- {Q1.index[1]}']
            ttl.reset_index(inplace=True)

            df_IQR = df_IQR.append(ttl)


            vertical_offset = df_review_featuresByChurnNotChurn[column].median() * 0.0
            box_plot = sns.boxplot(x="Churnstr", y=column, data=df_review_featuresByCh
            ax.set_xlabel('')

            for tick in ax.axes.xaxis.get_major_ticks():
                tick.label.set_fontsize(20)

            for tick in ax.axes.yaxis.get_major_ticks():
                tick.label.set_fontsize(20)

            for xtick in box_plot.get_xticks():
                medlabel = '{:.0f}'.format(medians[xtick])
                box_plot.text(xtick,medians[xtick] + vertical_offset,medlabel, horizon
                              ,size='medium',color='w',weight='bold', bbox=dict(facec
                              va='center')
```

```
248
249
250
251     plt.subplots_adjust(top=0.2)
252
253     plt.tight_layout(pad=4.0)
254     plt.show();
255     warnings.filterwarnings('default')
256
257     from IPython.display import display, HTML
258     display(HTML(df_IQR.to_html(index=False)))
259
260 #No Churn vs. Churn By Continous Ends ____________________________________________
261
262 #No Churn vs. Churn Index Graphs ____________________________________________
263
264 def ShowChurnNoIndex():
265     '''Creates Heatmaps, Uses Index, Churn vs. Non Churn Data Created in Earlier Steps
266     #Create Customer Profile for Churn vs non churn
267     df_Chrn_Cust_T = df[df["Churn"] == 1].describe().T
268     df_Nchrn_Cust_T = df[df["Churn"] == 0].describe().T
269
270     df_wD_all_gpby = df_wD_all.groupby(['Churn']).agg('sum')
271     df_wD_all_gpby = df_wD_all_gpby.reset_index().rename_axis(None, axis=1)
272     df_wD_all_gpby_T = df_wD_all_gpby.T
273     df_wD_all_gpby_T.columns = ['LoyalCust', 'ChurnCust']
274
275
276     df_wD_all_gpby_T["Ttls"] = df_wD_all_gpby_T["LoyalCust"] + df_wD_all_gpby_T["Churn
277     df_wD_all_gpby_T["Loyal_I"] = df_wD_all_gpby_T["LoyalCust"]/df_wD_all_gpby_T["Ttls
278     df_wD_all_gpby_T["Churn_I"] =  df_wD_all_gpby_T["ChurnCust"]/df_wD_all_gpby_T["Ttl
279     LoyalIndex = df_wD_all_gpby_T.loc[ ['Cnt'] ,['Loyal_I'] ].values[0][0]
280     ChurnIndex = df_wD_all_gpby_T.loc[ ['Cnt'] ,['Churn_I'] ].values[0][0]
281     df_wD_all_gpby_T["Loyal"] = df_wD_all_gpby_T["Loyal_I"]/LoyalIndex
282     df_wD_all_gpby_T["Churn"] =  df_wD_all_gpby_T["Churn_I"]/ChurnIndex
283
284     df_wD_all_gpby_T = df_wD_all_gpby_T.drop(['Churn', 'Cnt'])
285     df_wD_all_gpby_T.sort_values(by='Churn', ascending=True, inplace=True)
286
287
288     df_Cust_Profile_heatmap = df_wD_all_gpby_T.drop(columns=['LoyalCust', 'ChurnCust',
289                                                     axis=1)
290
291     colormap=sns.diverging_palette(220,10,as_cmap=True)
292     therows = list(df_Cust_Profile_heatmap.index)
293     thecols = list(df_Cust_Profile_heatmap.columns)
294
295     fig, ax = plt.subplots(figsize=(30,7))
296     im = ax.imshow(df_Cust_Profile_heatmap.T, cmap=colormap)
297
298     colormap=sns.diverging_palette(220,10,as_cmap=True)
299
300     fig.colorbar(im, orientation="horizontal", pad=0.5)
301
302     ax = sns.heatmap(df_Cust_Profile_heatmap.T, linewidth=0.5, cmap=colormap, cbar=Fal
303                      annot_kws={"size": 20,"style": "italic", "weight": "bold"},center
304
305     ax.set_yticks(np.arange(len(thecols)))
306     ax.set_xticks(np.arange(len(therows)))
307     ax.set_yticklabels(thecols,fontsize = 18)
308     ax.set_xticklabels(therows, fontsize = 18)
309
```

```
    # Rotate the tick labels and set their alignment.
    plt.setp(ax.get_xticklabels(), rotation=45, ha="right", rotation_mode="anchor")

    #ax.set_title("Index of Churn vs. No Churn vs. Average")
    plt.subplots_adjust(top=0.2)
    plt.suptitle('Feature Usage Index', fontsize = 50)
    plt.tight_layout(pad=4.0)
    plt.show()

def showCorrelationWithChurn():
    '''Creates Heatmap, Uses Data Created in Earlier Steps, plots Correlation Between
    # what features correlate with churn
    df_ChurnvsNoChurnCorrelation = df_wD_all.corr().filter(regex='Churn')
    df_ChurnvsNoChurnCorrelation = df_ChurnvsNoChurnCorrelation[df_ChurnvsNoChurnCorre
    df_ChurnvsNoChurnCorrelation.sort_values(by="Churn", inplace=True)

    colormap=sns.diverging_palette(220,10,as_cmap=True)
    therows = list(df_ChurnvsNoChurnCorrelation.index)
    thecols = list(df_ChurnvsNoChurnCorrelation.columns)

    fig, ax = plt.subplots(figsize=(30,7))
    im = ax.imshow(df_ChurnvsNoChurnCorrelation.T, cmap=colormap)

    fig.colorbar(im, orientation="horizontal", pad=0.5)

    ax = sns.heatmap(df_ChurnvsNoChurnCorrelation.T, linewidth=0.5, cmap=colormap, cba
                     annot=True, annot_kws={"size": 20,"style": "italic", "weight": "b


    ax.set_yticks(np.arange(len(thecols)))
    ax.set_xticks(np.arange(len(therows)))
    ax.set_yticklabels(thecols)
    ax.set_xticklabels(therows, fontsize =25)

    # Rotate the tick labels and set their alignment.
    plt.setp(ax.get_xticklabels(), rotation=45, ha="right", rotation_mode="anchor")

    plt.subplots_adjust(top=0.2)
    ax.set_title("Features Correlated with Churn", fontsize = 60)
    fig.tight_layout()
    plt.show()

```

In [3]:
```
print(ShowChurnVsNoChurnSummary_Counts.__doc__)
```

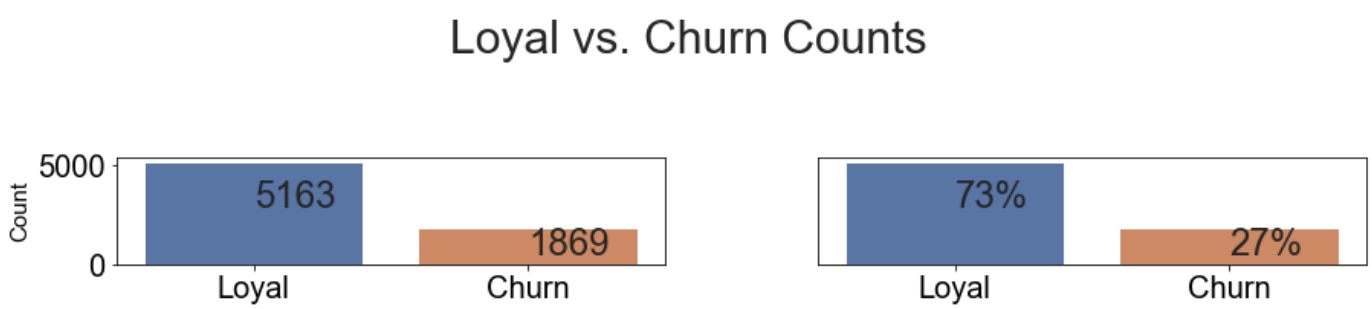
```
Creates Two Summary Graphs, Uses Data Created in Exploratory Part of Notebook
```

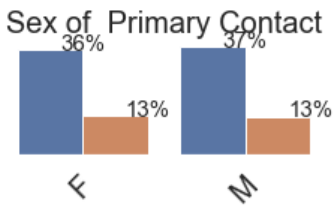
# Exploratory Analysis, and Preprocessing Data [...]

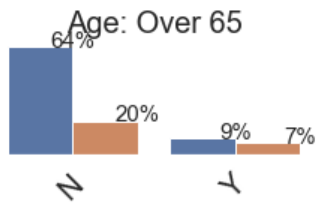
# Visuals - No Churn vs. Churn By Count

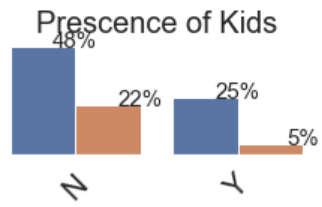
In [13]:

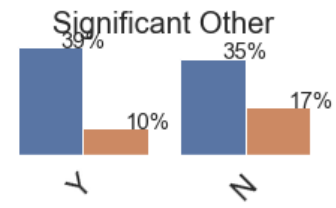
```
1 warnings.filterwarnings("ignore")
2 ShowChurnVsNoChurnSummary_Counts()
3 reviewCategoricalCounts()
```

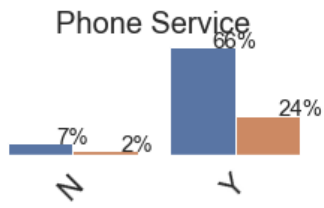


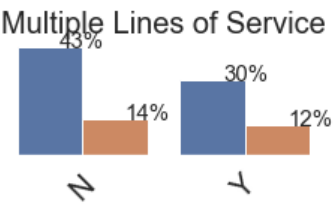

Sex of Primary Contact
36%
13%
37%
13%
F
M

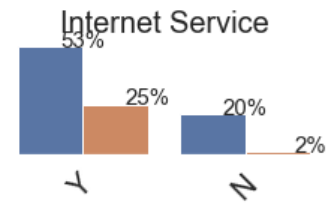

Age: Over 65
64%
20%
9%
7%
N
Y

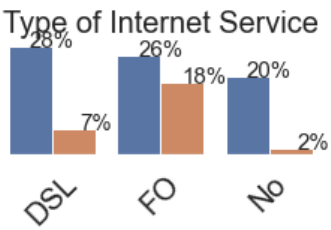

Prescence of Kids
48%
22%
25%
5%
N
Y

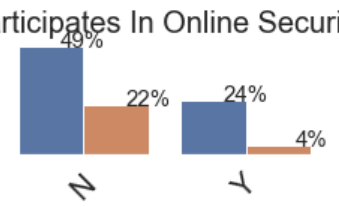

Significant Other
39%
10%
35%
17%
Y
N

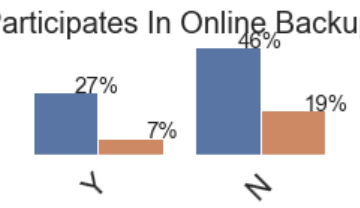

Phone Service
7%
2%
66%
24%
N
Y

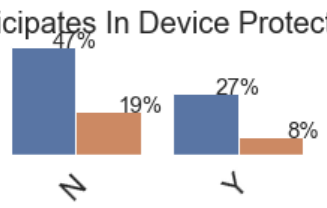

Multiple Lines of Service
43%
14%
30%
12%
N
Y

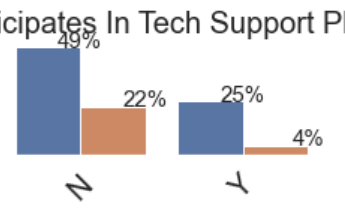

Internet Service
53%
25%
20%
2%
Y
N

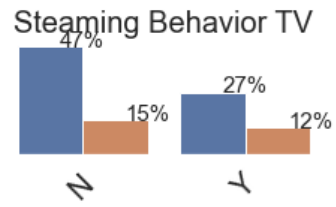

Type of Internet Service
28%
7%
26%
18%
20%
2%
DSL
FO
No

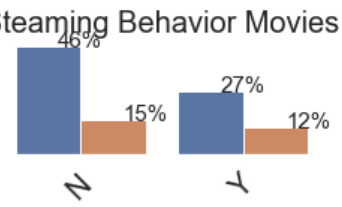

Participates In Online Security
49%
22%
24%
4%
N
Y

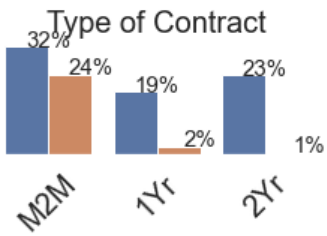

Participates In Online Backup
27%
7%
46%
19%
Y
N

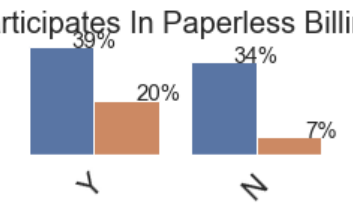

Participates In Device Protection
47%
19%
27%
8%
N
Y

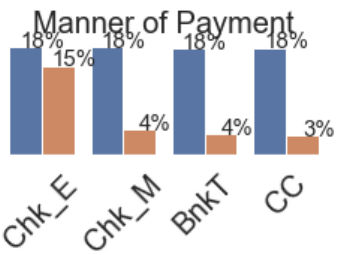

Participates In Tech Support Plan
49%
22%
25%
4%
N
Y

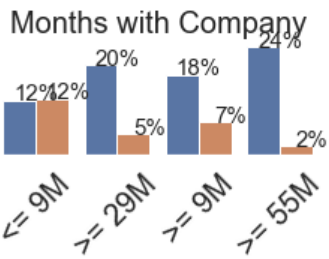

Steaming Behavior TV
47%
15%
27%
12%
N
Y

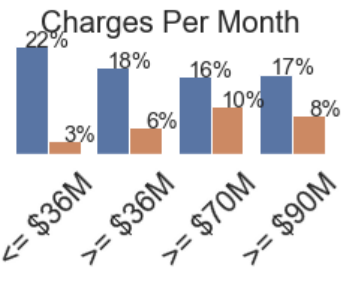

Steaming Behavior Movies
46%
15%
27%
12%
N
Y

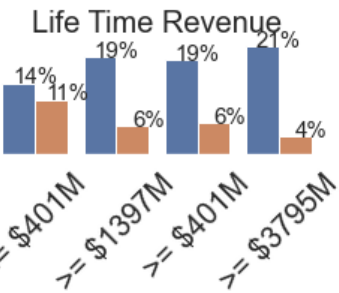

Type of Contract
32%
24%
19%
2%
23%
1%
M2M
1Yr
2Yr

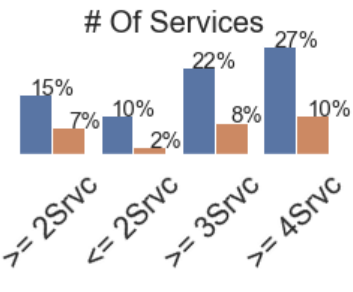

Participates In Paperless Billing
39%
20%
34%
7%
Y
N


Manner of Payment
18%
15%
18%
4%
18%
4%
18%
3%
Chk_E
Chk_M
BnkT
CC


Months with Company
12%
12%
20%
5%
18%
7%
24%
2%
<= 9M
>= 29M
>= 9M
>= 55M


Charges Per Month
22%
3%
18%
6%
16%
10%
17%
8%
<= $36M
>= $36M
>= $70M
>= $90M


Life Time Revenue
14%
11%
19%
6%
19%
6%
21%
4%
<= $401M
>= $1397M
>= $401M
>= $3795M


# Of Services
15%
7%
10%
2%
22%
8%
27%
10%
>= 2Srvc
<= 2Srvc
>= 3Srvc
>= 4Srvc

OBSERVATIONS:

Graphs - Loyal vs. Churn Summary:

- Dealing with imbalanced dataset, will employ "Weight" or SMOTE as neccessary.
- Type Of Contract - 89% of Churners In Month-To-Month Contracts
- Type of Internet Service - 66% Churners Part of Fiber Optics
- Count of Services - 66% Churners Part of 3 or more Services
- Monthly Charges - 66% of churners paying >= $70 per month, less churn with lower monthly spend

# Visuals - No Churn vs. Churn By Monthly Revenue

## Create Additional Dataframes Needed to Graph Revenue Breakdown

[...]

## Create Additional Dataframse to Calculate % Revenue Monthly and Total By Features

[...]

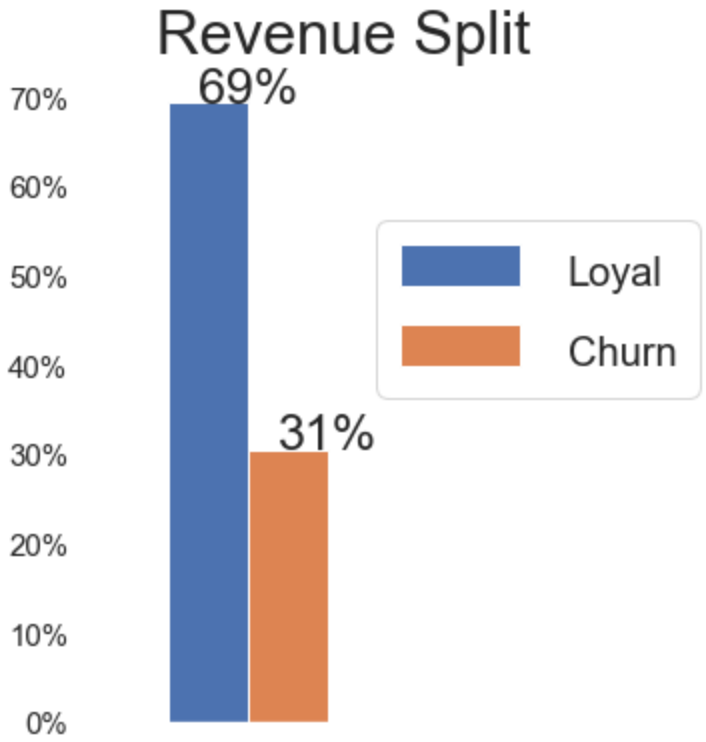
## Call Graph Functions

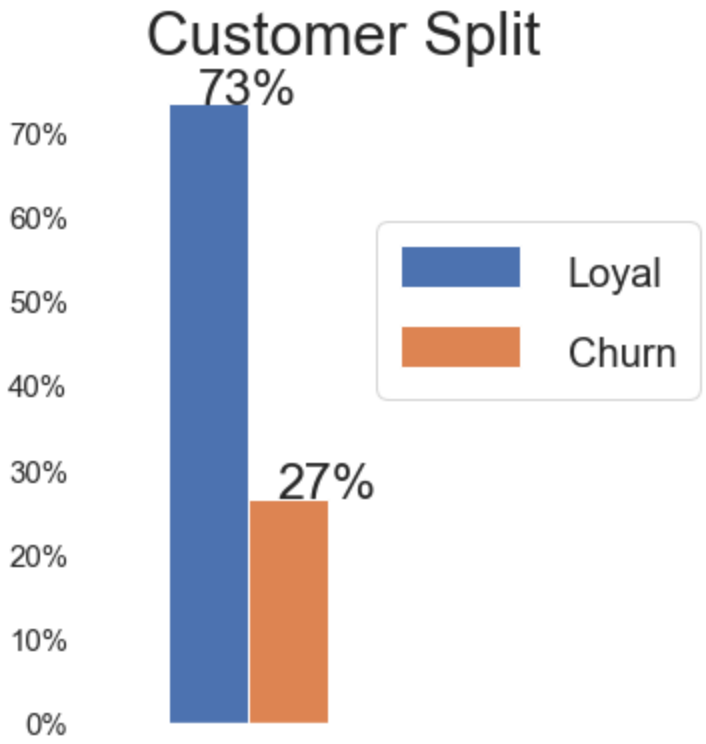
In [16]:

```
# Call graph function
warnings.filterwarnings("ignore")

suptitle = 'Revenue Split'
showPercentSummary(CountofGroupsToBeGraphed,plot_df,suptitle)

CountofGroupsToBeGraphed = df_churnpercents["group"].nunique()
plot_df1 = pd.crosstab(index=[df_churnpercents['group'],df_churnpercents['Title']], co
                       values=df_churnpercents['%Customers'], aggfunc=sum)
suptitle = 'Customer Split'
showPercentSummary(CountofGroupsToBeGraphed,plot_df1,suptitle)
```

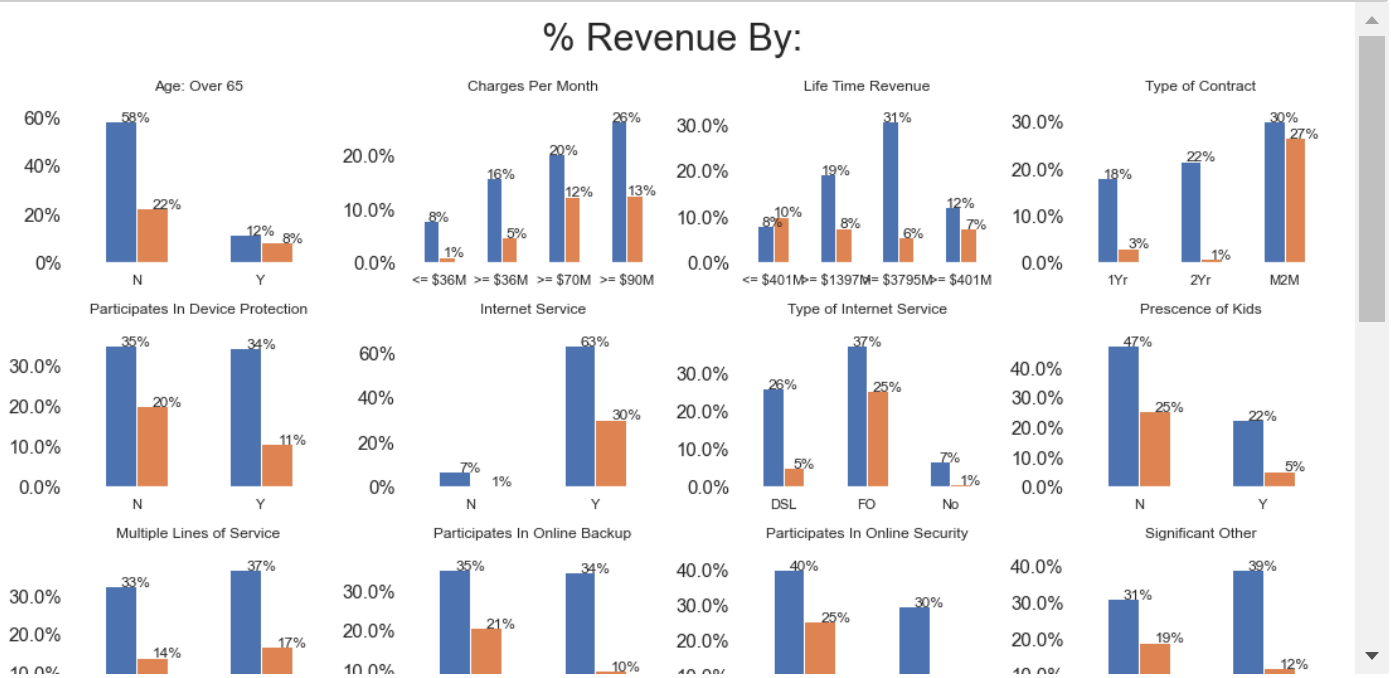



Revenue Summary By Feature

In [17]:

```
# Call graphs to view Montly Revenue by Feature
warnings.filterwarnings("ignore")
showPercentRevenueByFeature()
```




OBSERVATIONS:

Graphs - Loyal vs. Churn Summary:

- Building on Features Noted Above

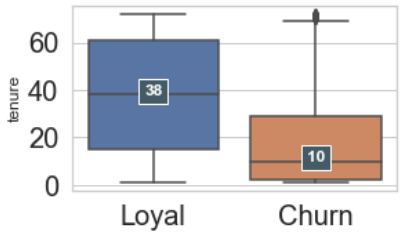
Graphs - Loyal vs. Churn Summary:

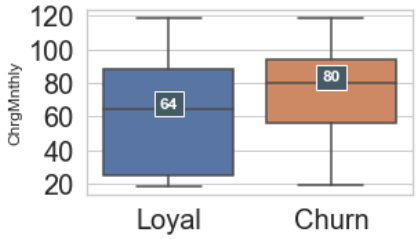
- Churners over index on spend, they make up 27% count, but account for 31% revenue
- Type Of Contract - 27% of Revenue at risk with Month-To-Month Contracts
- Type of Payment - Majority of Churners Pay with Electronic Pay
- Monthsly with Company - Majority of Churners Part of 3 or more Services
- Monthly Charges - 66% of churners paying >= $70 per month, less churn with lower monthly spend

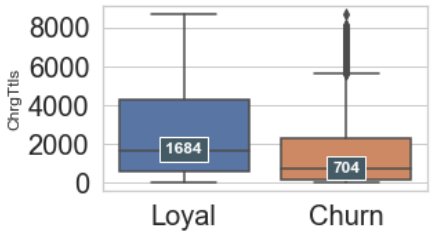
# Visuals - No Churn vs. Churn Misc

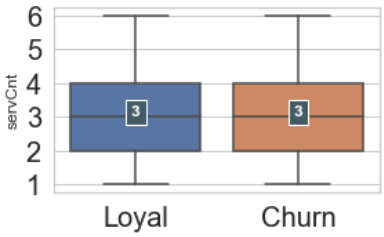
In [18]:

```
# Show Range of Countinous Features
ShowChurnNoChurnByContinousFeatures()
```



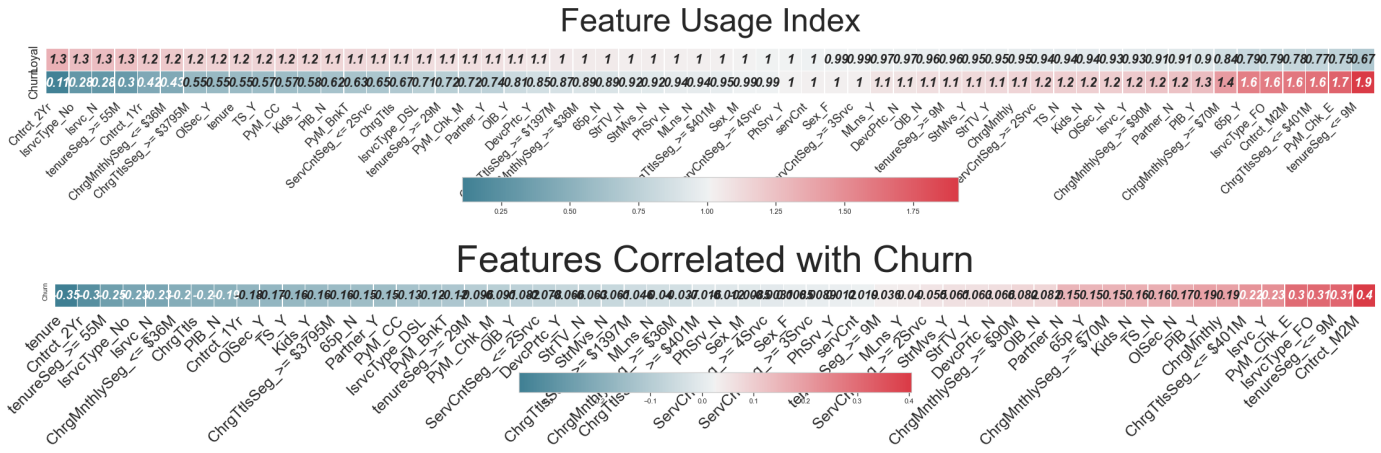




| 25% | 75% | index |
|---|---|---|
| 2.00 | 29 | tenure - Churn |
| 15 | 61 | tenure- Loyal |
| 56 | 94 | ChrgMnthly - Churn |
| 25 | 88 | ChrgMnthly- Loyal |
| 134 | 2331 | ChrgTtls - Churn |
| 578 | 4264 | ChrgTtls- Loyal |
| 2.00 | 4 | servCnt - Churn |
| 2.00 | 4 | servCnt- Loyal |

Churn Correlation By Feature

In [19]:
```
# Correlations between features and CHurn
ShowChurnNoIndex()
showCorrelationWithChurn()
```




OBSERVATIONS:

Graphs - Loyal vs. Churn Summary:

- Tenure - 75% of Churners leave between 2 - 29 months of usage, a few longer term customers also are showing churn.
- Charges Monthly - Churners seem to have a hire monthly spend and/or are charged more on a monthly bases vs Loyal customers.
- Charges Totals - charges obviously greater for Loyal customers vs. Churners.

Feature Usage Index:

- Churners over index on Paying with Check Electronically, pay Month to Month vs. Contract, participate in Fiber Optics Internet Services, skew a little older than younger, and favor Paperless Billing.
- Loyal customers over index on Contract usage (1&2Yr), Not using Internet Services, and using Online Security.

Features Correlated with Churn:

- Correlation figures support points shown in Index Heatmap.

# Conclusions

CONCLUSIONS:

- Building on Features Noted Above

Graphs - Loyal vs. Churn Summary:

- Churners over index on spend, they make up 27% count, but account for 31% revenue
- Type Of Contract - 27% of Revenue at risk with Month-To-Month Contracts
- Type of Payment - Majority of Churners Pay with Electronic Pay
- Monthsly with Company - Majority of Churners Part of 3 or more Services
- Monthly Charges - 66% of churners paying >= $70 per month, less churn with lower monthly spend

Graphs - Loyal vs. Churn Summary:

- Dealing with imbalanced dataset, will employ "Weight" or SMOTE as neccessary.
- Type Of Contract - 89% of Churners In Month-To-Month Contracts, 27% of Revenue at risk with Month-To-Month Contracts
- Tenure - 75% of Churners leave between 2 - 29 months of usage, a few longer term customers also are showing churn.
- Type of Internet Service - 66% Churners Part of Fiber Optics
- Count of Services - 66% Churners Part of 3 or more Services
- Monthly Charges - 66% of churners paying >= $70 per month, less churn with lower monthly spend

Feature Usage Index:

- Churners over index on Paying with Check Electronically, paying Month to Month, participate in Fiber Optics Internet Services, skew a little older than younger, and favor Paperless Billing.
- Loyal customers over index on Contract usage (1&2Yr), Not using Internet Services, and using Online Security.

# Create Pickle Files for Modeling

In [20]:
```
#Create Files for Modeling
with open('./data/df_wD_all.pickle', 'wb') as f:
    pickle.dump(df_wD_all, f)

with open('./data/df_wD.pickle', 'wb') as f:
    pickle.dump(df_wD, f)
```

In [ ]:
```

```