In [6]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns # For creating plots
import matplotlib.ticker as mtick # For specifying the axes tick format
import matplotlib.pyplot as plt

sns.set(style = 'white')

# Input data files are available in the "../input/" directory.

import os
print(os.listdir('../A-BAHA'))
```

```
['.ipynb_checkpoints', 'telecom_customer_churn.csv', 'telecom_data_dictionar
y.csv', 'telecom_zipcode_population.csv', 'Untitled.ipynb', 'WA_Fn-UseC_-Telc
o-Customer-Churn.csv']
```

In [9]:

```
telecom_cust = pd.read_csv('../A-BAHA/WA_Fn-UseC_-Telco-Customer-Churn.csv')
telecom_cust
```

Out[9]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLi
0	7590-VHVEG	Female	0	Yes	No	1	No	No ph serv
1	5575-GNVDE	Male	0	No	No	34	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	
3	7795-CFOCW	Male	0	No	No	45	No	No ph serv
4	9237-HQITU	Female	0	No	No	2	Yes	
5	9305-CDSKC	Female	0	No	No	8	Yes	
6	1452-KIOVK	Male	0	No	Yes	22	Yes	

In [10]:

```
telecom_cust.columns.values
```

Out[10]:

```
array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

In [8]:

```
telecom_cust.head()
```

Out[8]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service
1	5575-GNVDE	Male	0	No	No	34	Yes	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No
3	7795-CFOCW	Male	0	No	No	45	No	No phone service
4	9237-HQITU	Female	0	No	No	2	Yes	No

5 rows × 21 columns

In [11]:

```
# Checking the data types of all the columns
telecom_cust.dtypes
```

Out[11]:

```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

In [12]:

```
# Converting Total Charges to a numerical data type.
telecom_cust.TotalCharges = pd.to_numeric(telecom_cust.TotalCharges, errors='co
telecom_cust.isnull().sum()
```

Out[12]:

```
customerID           0
gender               0
SeniorCitizen        0
Partner              0
Dependents           0
tenure               0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
MonthlyCharges       0
TotalCharges        11
Churn                0
dtype: int64
```

In [15]:

```
#Removing missing values
telecom_cust.dropna(inplace = True)
#Remove customer IDs from the data set
df2 = telecom_cust.iloc[:,1:]
#Convertin the predictor variable in a binary numeric variable
df2['Churn'].replace(to_replace='Yes', value=1, inplace=True)
df2['Churn'].replace(to_replace='No',  value=0, inplace=True)

#Let's convert all the categorical variables into dummy variables
df_dummies = pd.get_dummies(df2)
df_dummies.head()
```

Out[15]:

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male	Pa
0	0	1	29.85	29.85	0	1	0	
1	0	34	56.95	1889.50	0	0	1	
2	0	2	53.85	108.15	1	0	1	
3	0	45	42.30	1840.75	0	0	1	
4	0	2	70.70	151.65	1	1	0	

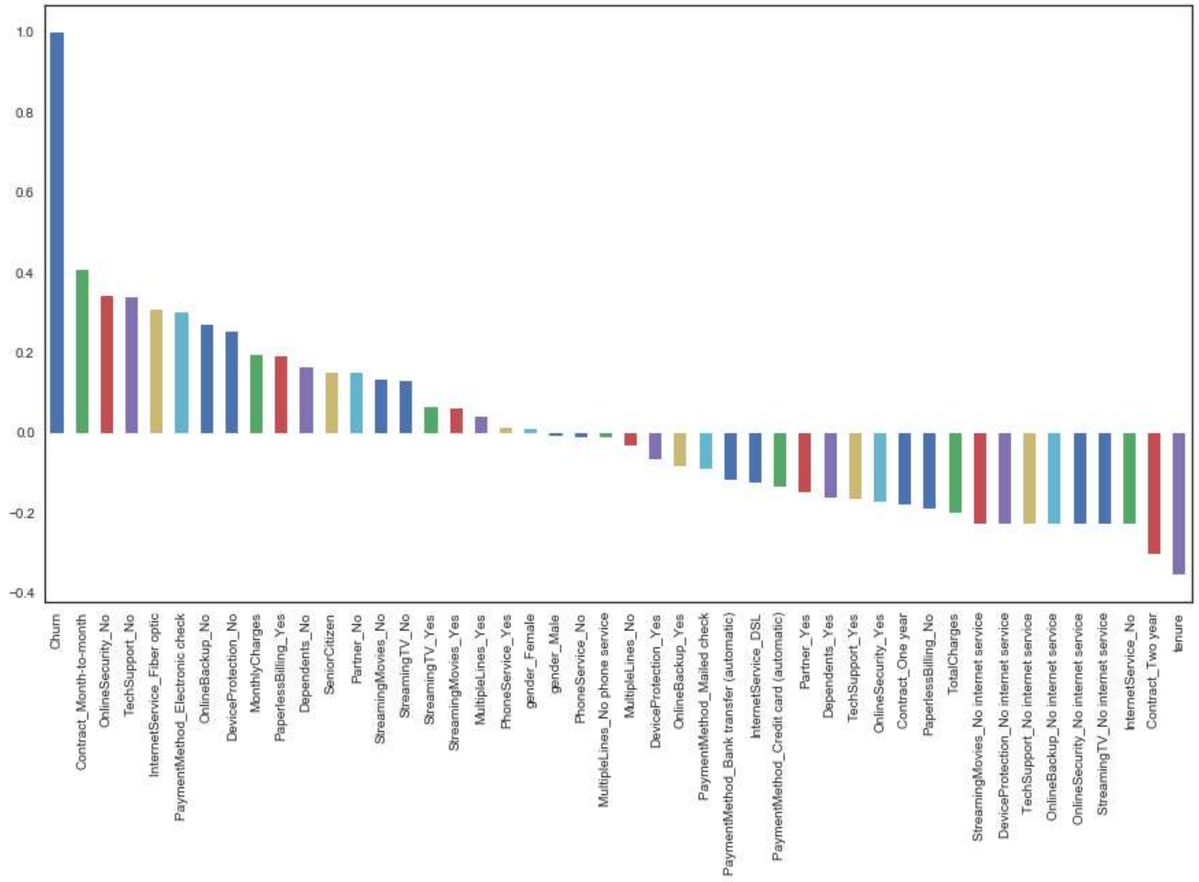
5 rows × 46 columns

In [14]:

```
#Get Correlation of "Churn" with other variables:
plt.figure(figsize=(15,8))
df_dummies.corr()['Churn'].sort_values(ascending = False).plot(kind='bar')
```

Out[14]:

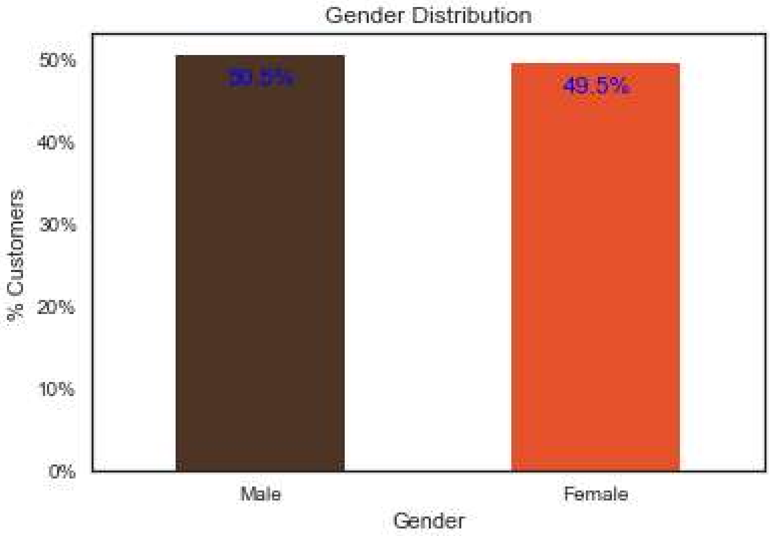
```
<matplotlib.axes._subplots.AxesSubplot at 0x247eef5e358>
```



gender distribution

In [21]:

```
colors = ['#4D3425','#E4512B']
ax = (telecom_cust['gender'].value_counts()*100.0 /len(telecom_cust)).plot(kind
                                                                            stac
                                                                           rot =
                                                                           color

ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.set_ylabel('% Customers')
ax.set_xlabel('Gender')
ax.set_ylabel('% Customers')
ax.set_title('Gender Distribution')

# create a list to collect the plt.patches data
totals = []

# find the values and append to list
for i in ax.patches:
    totals.append(i.get_width())

# set individual bar lables using above list
total = sum(totals)

for i in ax.patches:
    # get_width pulls left or right; get_y pushes up or down
    ax.text(i.get_x()+.15, i.get_height()-3.5, \
            str(round((i.get_height()/total), 1))+'%',
            fontsize=12,
            color='blue',
           weight = 'light')
```



% Senior Citizens

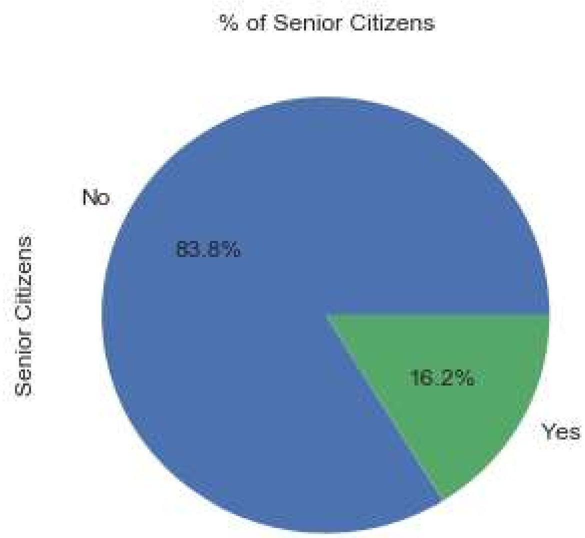
- There are only 16% of the customers who are senior citizens. Thus most of our customers in the data are younger people.

In [22]:

```
ax = (telecom_cust['SeniorCitizen'].value_counts()*100.0 /len(telecom_cust))\
.plot.pie(autopct='%.1f%%', labels = ['No', 'Yes'],figsize =(5,5), fontsize = 1
ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.set_ylabel('Senior Citizens',fontsize = 12)
ax.set_title('% of Senior Citizens', fontsize = 12)
```

Out[22]:

```
Text(0.5,1,'% of Senior Citizens')
```



Partner and dependent status -

About 50% of the customers have a partner, while only 30% of the total customers have dependents.

In [ ]:

```
df2 = pd.melt(telecom_cust, id_vars=['customerID'], value_vars=['Dependents','P
df3 = df2.groupby(['variable','value']).count().unstack()
df3 = df3*100/len(telecom_cust)
colors = ['#4D3425','#E4512B']
ax = df3.loc[:,'customerID'].plot.bar(stacked=True, color=colors,
                                      figsize=(8,6),rot = 0,
                                     width = 0.2)

ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.set_ylabel('% Customers',size = 14)
ax.set_xlabel('')
ax.set_title('% Customers with dependents and partners',size = 14)
ax.legend(loc = 'center',prop={'size':14})

for p in ax.patches:
    width, height = p.get_width(), p.get_height()
    x, y = p.get_xy()
    ax.annotate('{:.0f}%'.format(height), (p.get_x()+.25*width, p.get_y()+.4*he
                color = 'white', weight = 'bold',
               size = 14)
```

Customer Account Information

Tenure: After looking at the below histogram we can see that a lot of customers have been with the telecom company for just a month, while quite a many are there for about 72 months. This could be potentially because different customers have different contracts. Thus based on the contract they are into it could be more/less easier for the customers to stay/leave the telecom company.

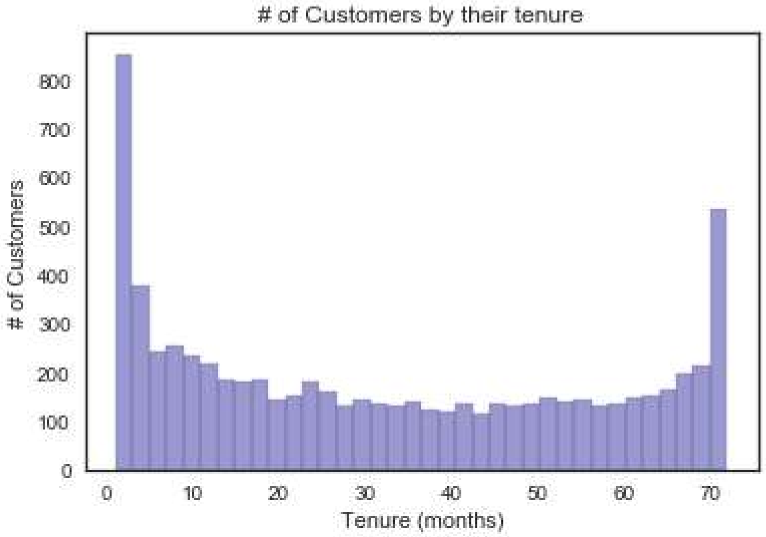
In [24]:

```
ax = sns.distplot(telecom_cust['tenure'], hist=True, kde=False,
             bins=int(180/5), color = 'darkblue',
             hist_kws={'edgecolor':'black'},
             kde_kws={'linewidth': 4})
ax.set_ylabel('# of Customers')
ax.set_xlabel('Tenure (months)')
ax.set_title('# of Customers by their tenure')
```

```
C:\Users\John Doe\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced by the
'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

Out[24]:

```
Text(0.5,1,'# of Customers by their tenure')
```



Contracts:

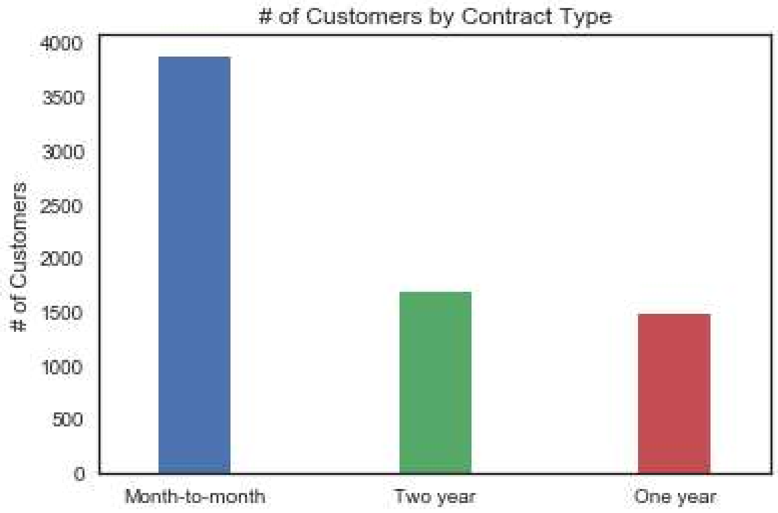
To understand the above graph, lets first look at the # of customers by different contracts.

In [25]:

```
ax = telecom_cust['Contract'].value_counts().plot(kind = 'bar',rot = 0, width =
ax.set_ylabel('# of Customers')
ax.set_title('# of Customers by Contract Type')
```

Out[25]:

```
Text(0.5,1,'# of Customers by Contract Type')
```



As we can see from this graph most of the customers are in the month to month contract. While there are equal number of customers in the 1 year and 2 year contracts.

Below we will understand the tenure of customers based on their contract type.

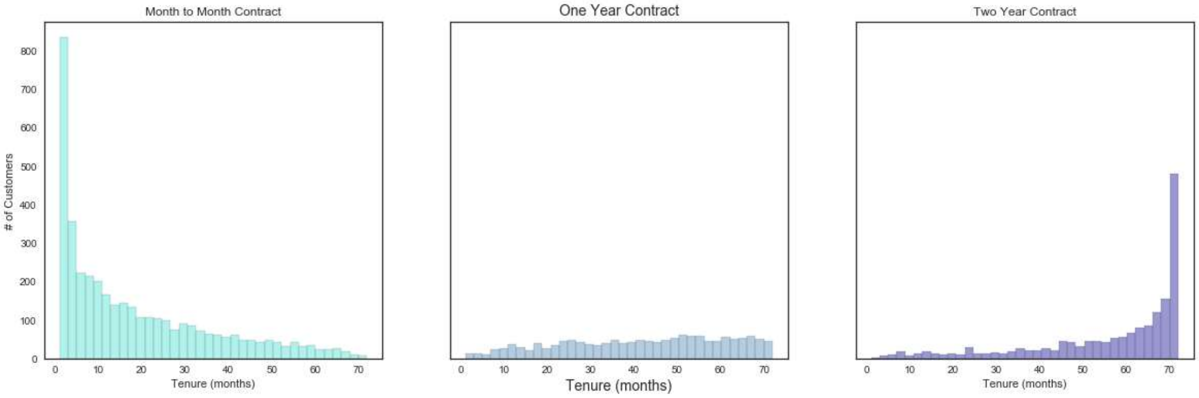
In [26]:

```
fig, (ax1,ax2,ax3) = plt.subplots(nrows=1, ncols=3, sharey = True, figsize = (2

ax = sns.distplot(telecom_cust[telecom_cust['Contract']=='Month-to-month']['ten
                   hist=True, kde=False,
                   bins=int(180/5), color = 'turquoise',
                   hist_kws={'edgecolor':'black'},
                   kde_kws={'linewidth': 4},
                 ax=ax1)
ax.set_ylabel('# of Customers')
ax.set_xlabel('Tenure (months)')
ax.set_title('Month to Month Contract')

ax = sns.distplot(telecom_cust[telecom_cust['Contract']=='One year']['tenure'],
                   hist=True, kde=False,
                   bins=int(180/5), color = 'steelblue',hist_kws={'edgecolor':'
                   kde_kws={'linewidth': 4},
                 ax=ax2)
ax.set_xlabel('Tenure (months)',size = 14)
ax.set_title('One Year Contract',size = 14)

ax = sns.distplot(telecom_cust[telecom_cust['Contract']=='Two year']['tenure'],
                   hist=True, kde=False,
                   bins=int(180/5), color = 'darkblue',
                   hist_kws={'edgecolor':'black'},
                   kde_kws={'linewidth': 4},
                 ax=ax3)

ax.set_xlabel('Tenure (months)')
ax.set_title('Two Year Contract')
```

```
C:\Users\John Doe\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced by the
'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
C:\Users\John Doe\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced by the
'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
C:\Users\John Doe\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced by the
'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

Out[26]:

```
Text(0.5,1,'Two Year Contract')
```

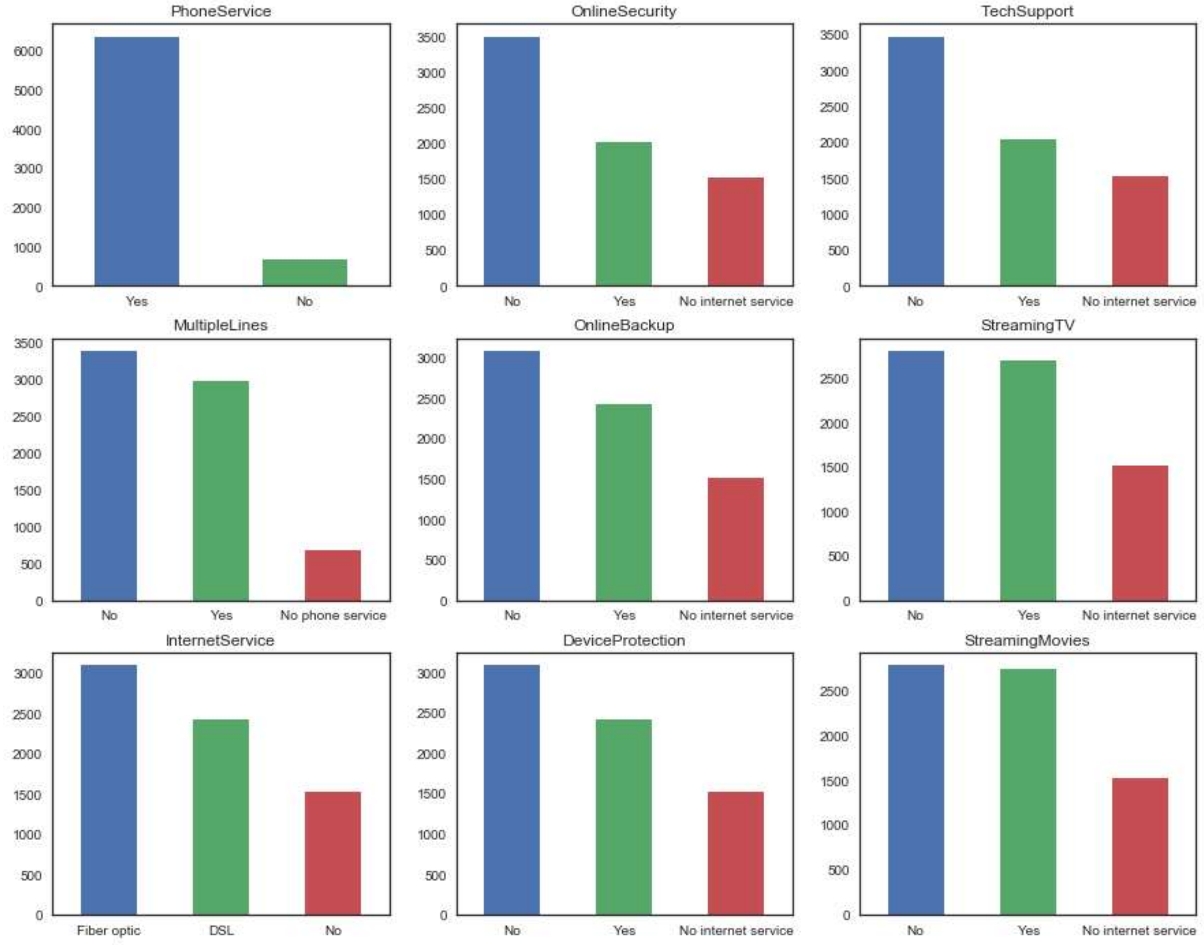


Let us now look at the distribution of various services used by customers

In [27]:
```
telecom_cust.columns.values
```

Out[27]:
```
array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

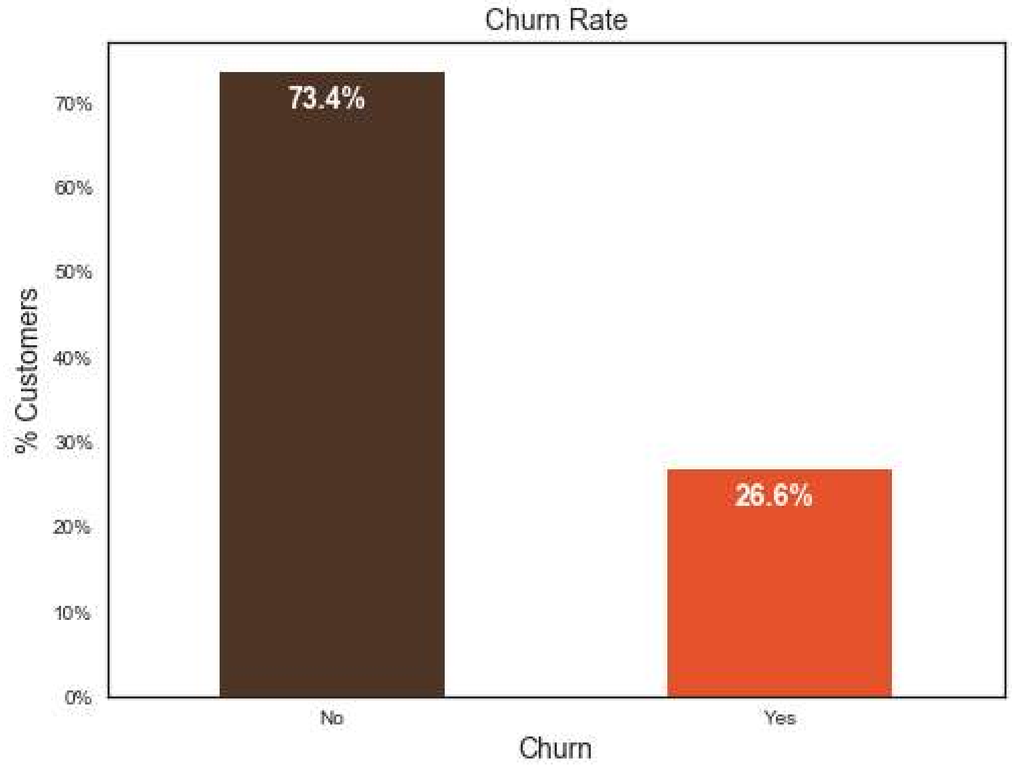
In [29]:
```
services = ['PhoneService','MultipleLines','InternetService','OnlineSecurity',
            'OnlineBackup','DeviceProtection','TechSupport','StreamingTV','Strea

fig, axes = plt.subplots(nrows = 3,ncols = 3,figsize = (15,12))
for i, item in enumerate(services):
    if i < 3:
        ax = telecom_cust[item].value_counts().plot(kind = 'bar',ax=axes[i,0],r

    elif i >=3 and i < 6:
        ax = telecom_cust[item].value_counts().plot(kind = 'bar',ax=axes[i-3,1]
    elif i < 9:
        ax = telecom_cust[item].value_counts().plot(kind = 'bar',ax=axes[i-6,2]
    ax.set_title(item)
```



Finally, let's take a look at out predictor variable (Churn) and understand its interaction with other important variables as was found out in the correlation plot.

In [32]:

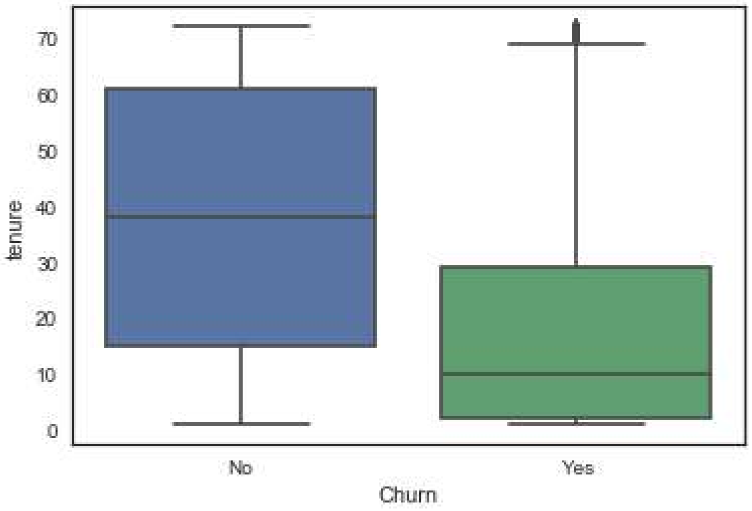
```
#Lets first look at the churn rate in our data
colors = ['#4D3425','#E4512B']
ax = (telecom_cust['Churn'].value_counts()*100.0 /len(telecom_cust)).plot(kind=
                                                                           stac
                                                                          rot =
                                                                          color
                                                                         figsiz

ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.set_ylabel('% Customers',size = 14)
ax.set_xlabel('Churn',size = 14)
ax.set_title('Churn Rate', size = 14)
# create a list to collect the plt.patches data
totals = []

# find the values and append to list
for i in ax.patches:
    totals.append(i.get_width())

# set individual bar lables using above list
total = sum(totals)

for i in ax.patches:
    # get_width pulls left or right; get_y pushes up or down
    ax.text(i.get_x()+.15, i.get_height()-4.0, \
            str(round((i.get_height()/total), 1))+'%',
            fontsize=12,
            color='white',
           weight = 'bold',
           size = 14)
```



Churn vs Tenure:

As we can see form the below plot, the customers who do not churn, they tend to stay for a longer tenure with the telecom company.

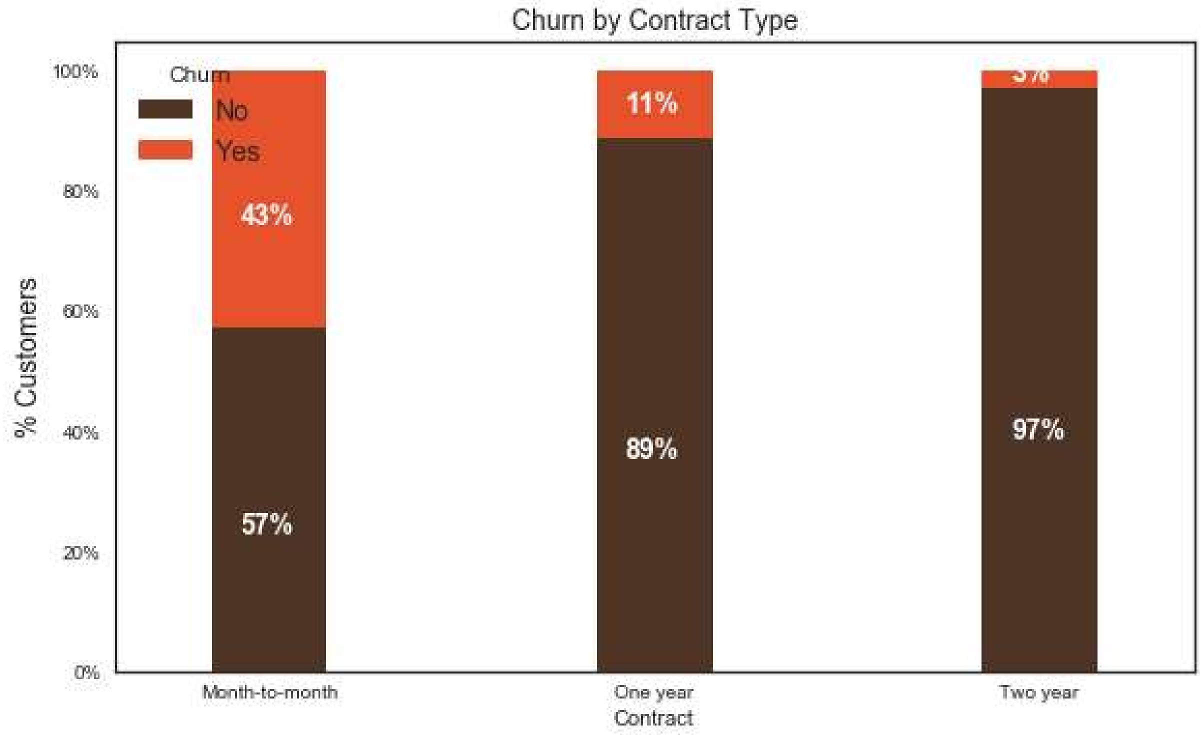
```
In [33]: sns.boxplot(x = telecom_cust.Churn, y = telecom_cust.tenure)

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x247f13e1470>
```



Churn by Contract Type:

Similar to what we saw in the correlation plot, the customers who have a month to month contract have a very high churn rate.

```
In [34]: colors = ['#4D3425','#E4512B']
         contract_churn = telecom_cust.groupby(['Contract','Churn']).size().unstack()

         ax = (contract_churn.T*100.0 / contract_churn.T.sum()).T.plot(kind='bar',
                                                                         width = 0.3,
                                                                         stacked = True,
                                                                         rot = 0,
                                                                         figsize = (10,6
                                                                         color = colors)
         ax.yaxis.set_major_formatter(mtick.PercentFormatter())
         ax.legend(loc='best',prop={'size':14},title = 'Churn')
         ax.set_ylabel('% Customers',size = 14)
         ax.set_title('Churn by Contract Type',size = 14)
         # Code to add the data labels on the stacked bar chart
         for p in ax.patches:
             width, height = p.get_width(), p.get_height()
             x, y = p.get_xy()
             ax.annotate('{:.0f}%'.format(height), (p.get_x()+.25*width, p.get_y()+.4*he
                             color = 'white',
                            weight = 'bold',
                            size = 14)
```

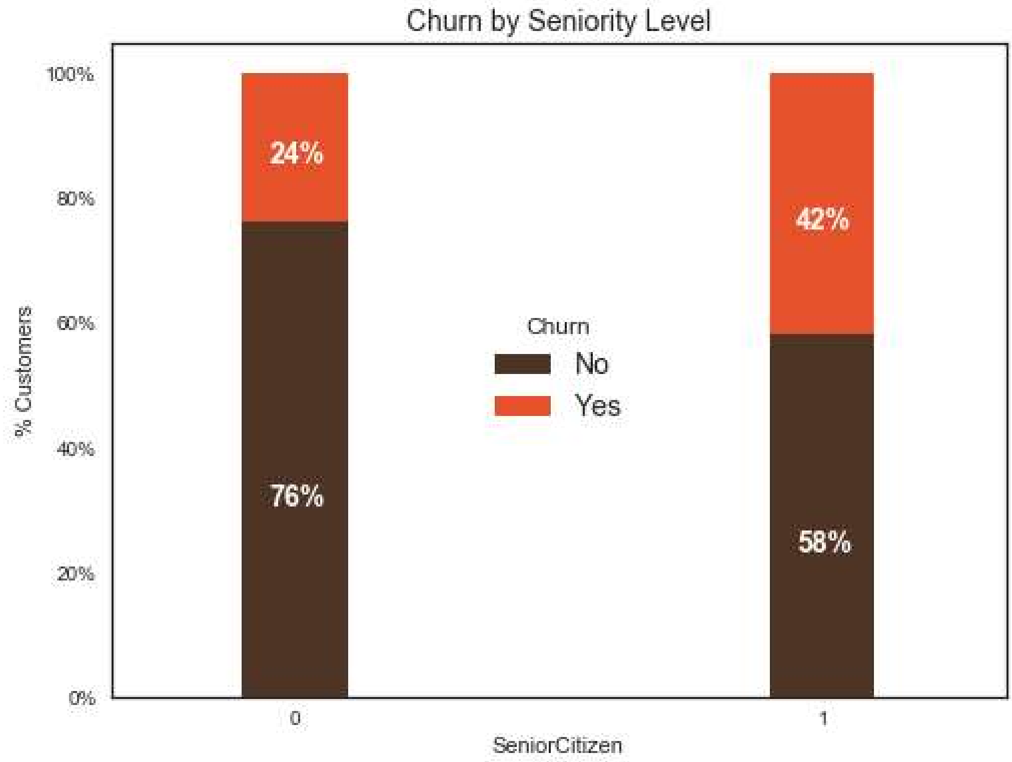


Churn by Seniority:

Senior Citizens have almost double the churn rate than younger population.

In [35]:

```
colors = ['#4D3425','#E4512B']
seniority_churn = telecom_cust.groupby(['SeniorCitizen','Churn']).size().unstac

ax = (seniority_churn.T*100.0 / seniority_churn.T.sum()).T.plot(kind='bar',
                                                                width = 0.2,
                                                                stacked = True,
                                                                rot = 0,
                                                                figsize = (8,6)
                                                                color = colors)
ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.legend(loc='center',prop={'size':14},title = 'Churn')
ax.set_ylabel('% Customers')
ax.set_title('Churn by Seniority Level',size = 14)

# Code to add the data labels on the stacked bar chart
for p in ax.patches:
    width, height = p.get_width(), p.get_height()
    x, y = p.get_xy()
    ax.annotate('{:.0f}%'.format(height), (p.get_x()+.25*width, p.get_y()+.4*he
                color = 'white',
               weight = 'bold',size =14)
```



Churn by Monthly Charges:

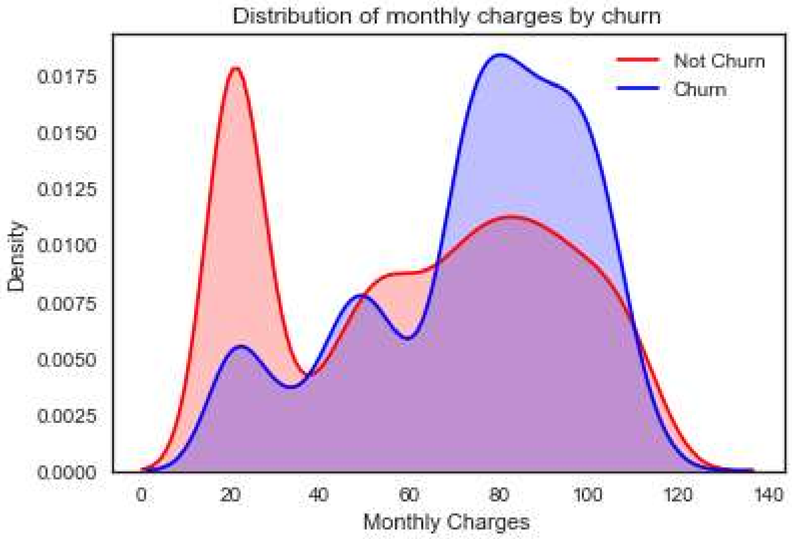
Higher % of customers churn when the monthly charges are high.

In [36]:

```
ax = sns.kdeplot(telecom_cust.MonthlyCharges[(telecom_cust["Churn"] == 'No') ],
                color="Red", shade = True)
ax = sns.kdeplot(telecom_cust.MonthlyCharges[(telecom_cust["Churn"] == 'Yes') ]
                ax =ax, color="Blue", shade= True)
ax.legend(["Not Churn","Churn"],loc='upper right')
ax.set_ylabel('Density')
ax.set_xlabel('Monthly Charges')
ax.set_title('Distribution of monthly charges by churn')
```

Out[36]:

```
Text(0.5,1,'Distribution of monthly charges by churn')
```



Churn by Total Charges:

It seems that there is higer churn when the total charges are lower.

After going through the above EDA we will develop some predictive models and compare them.

In [ ]:

```
Logistic Regression
```

```
In [38]: # We will use the data frame where we had created dummy variables
         y = df_dummies['Churn'].values
         X = df_dummies.drop(columns = ['Churn'])

         # Scaling all the variables to a range of 0 to 1
         from sklearn.preprocessing import MinMaxScaler
         features = X.columns.values
         scaler = MinMaxScaler(feature_range = (0,1))
         scaler.fit(X)
         X = pd.DataFrame(scaler.transform(X))
         X.columns = features
```

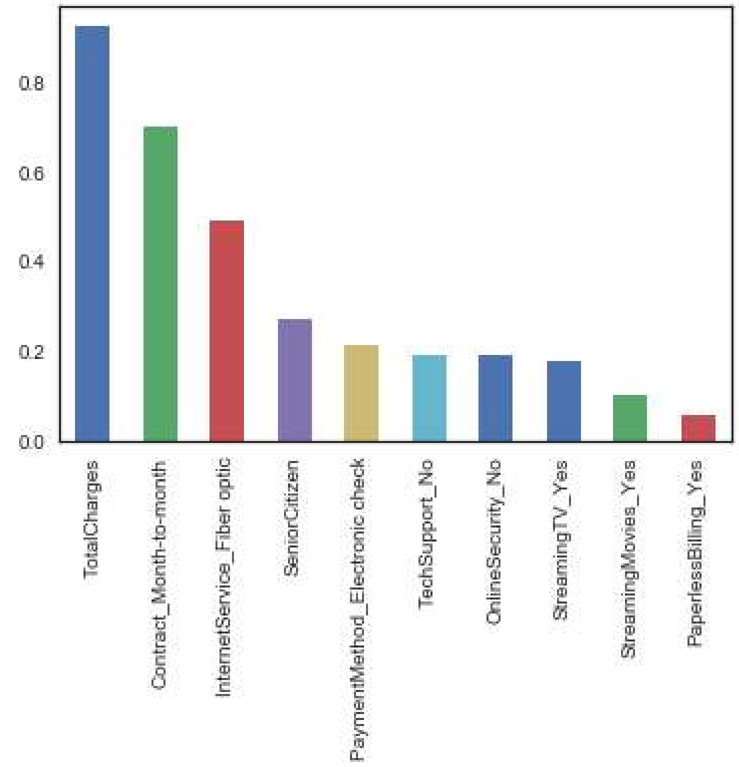
```
In [39]: # Create Train & Test Data
         from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random
```

```
In [40]: # Running logistic regression model
         from sklearn.linear_model import LogisticRegression
         model = LogisticRegression()
         result = model.fit(X_train, y_train)
```

```
In [41]: from sklearn import metrics
         prediction_test = model.predict(X_test)
         # Print the prediction accuracy
         print (metrics.accuracy_score(y_test, prediction_test))

         0.8075829383886256
```

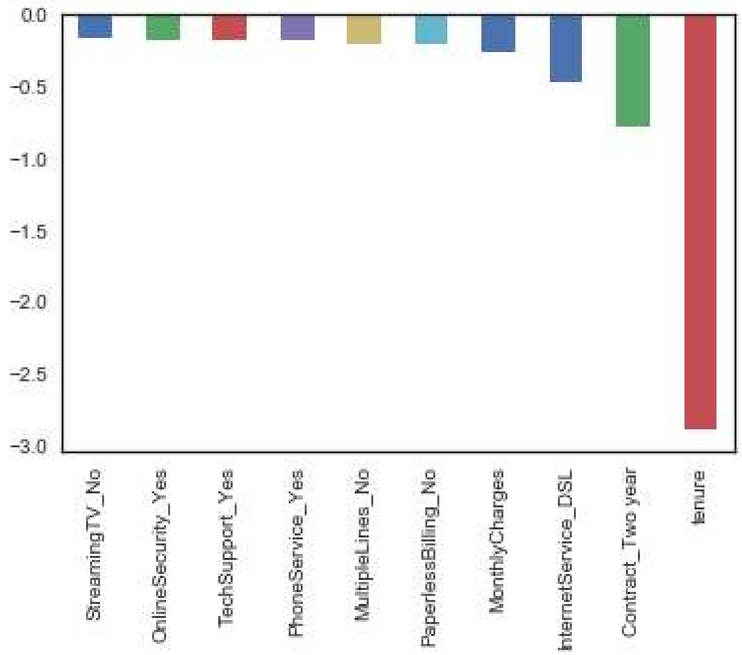
```
In [42]: # To get the weights of all the variables
         weights = pd.Series(model.coef_[0],
                          index=X.columns.values)
         print (weights.sort_values(ascending = False)[:10].plot(kind='bar'))
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```



```
In [43]: print(weights.sort_values(ascending = False)[-10:].plot(kind='bar'))
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```


StreamingTV_No
OnlineSecurity_Yes
TechSupport_Yes
PhoneService_Yes
MultipleLines_No
PaperlessBilling_No
MonthlyCharges
InternetService_DSL
Contract_Two year
tenure