In [1]:
```
import pandas as pd
import numpy as np
import warnings

from scipy import special
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import seaborn as sns
import math
from IPython.display import Markdown, display ,HTML


from sklearn.model_selection import train_test_split


sns.set(style="whitegrid")
pd.set_option('display.max_columns', 100)
pd.set_option('display.max_colwidth', -1) # make sure data and colu
pd.options.display.float_format = '{:20,.2f}'.format # display floa


warnings.filterwarnings('ignore')
```

```
<ipython-input-1-bcb97af57e2e>:18: FutureWarning: Passing a negati
ve integer is deprecated in version 1.0 and will not be supported
in future version. Instead, use None to not limit the column widt
h.
  pd.set_option('display.max_colwidth', -1) # make sure data and c
olumns are displayed correctly withput purge
```

In [2]:
```
telecom_df = pd.read_csv('telecom_churn_data.csv')
telecom_df.head(3)
```

Out[2]:

	mobile_number	circle_id	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	last_date_c
0	7000842753	109	0.00	0.00	0.00	
1	7001865778	109	0.00	0.00	0.00	
2	7001625959	109	0.00	0.00	0.00	

3 rows × 226 columns

#Derive New features

In [3]:
```
amt_recharge_columns =  telecom_df.columns[telecom_df.columns.str.c
print(amt_recharge_columns)
```

```
Index(['total_rech_amt_6', 'total_rech_amt_7', 'total_rech_amt_8',
       'total_rech_amt_9', 'max_rech_amt_6', 'max_rech_amt_7',
       'max_rech_amt_8', 'max_rech_amt_9', 'date_of_last_rech_data
 6'
```

In [4]:
```
telecom_df['total_rech_data_6'] = telecom_df['total_rech_data_6'].r
telecom_df['total_rech_data_7'] = telecom_df['total_rech_data_7'].r
telecom_df['total_rech_data_8'] = telecom_df['total_rech_data_8'].r
```

In [5]:
```
telecom_df['av_rech_amt_data_6'] = telecom_df['av_rech_amt_data_6']
telecom_df['av_rech_amt_data_7'] = telecom_df['av_rech_amt_data_7']
telecom_df['av_rech_amt_data_8'] = telecom_df['av_rech_amt_data_8']
```

In [6]:
```
telecom_df['total_rech_amt_data_6'] = telecom_df.av_rech_amt_data_6
telecom_df['total_rech_amt_data_7'] = telecom_df.av_rech_amt_data_7
telecom_df['total_rech_amt_data_8'] = telecom_df.av_rech_amt_data_8
```

In [7]:
```
telecom_df['total_avg_rech_amnt_6_7_GPhase'] = (telecom_df.total_re
                                                + telecom_df.total_r
```

In [8]:
```
high_value_filter = telecom_df.total_avg_rech_amnt_6_7_GPhase.quant

print('70 percentile of 6th and 7th months avg recharge amount: '+s

telecom_df_high_val_cust = telecom_df[telecom_df.total_avg_rech_amn
print('Dataframe Shape after Filtering High Value Customers: ' + st
```

```
70 percentile of 6th and 7th months avg recharge amount: 478.0
Dataframe Shape after Filtering High Value Customers: (29953, 230)
```

In [9]:
```
high_val_cust_9 = ['total_ic_mou_9', 'total_og_mou_9', 'vol_2g_mb_9
```

In [10]:
```
telecom_df_high_val_cust['churn']= 0
```

In [11]:
```
is_churned = (telecom_df_high_val_cust.total_ic_mou_9 == 0) & \
             (telecom_df_high_val_cust.total_og_mou_9 == 0) & \
             (telecom_df_high_val_cust.vol_2g_mb_9 ==0) & \
             (telecom_df_high_val_cust.vol_3g_mb_9 ==0)
```

In [12]:
```
telecom_df_high_val_cust.loc[is_churned,'churn']=1
```

In [13]:
```
100*telecom_df_high_val_cust.churn.sum()/len(telecom_df_high_val_cu
```

Out[13]:
```
8.122725603445398
```

In [14]:
```
churn_month_columns = telecom_df_high_val_cust.columns[telecom_df_h
```

In [15]:
```
telecom_df_high_val_cust.drop(churn_month_columns,axis=1,inplace=Tr
```

#EDA

In [16]:
```
#list_potential_categorical_type(get_meta_data(telecom_df_high_val
```

In [17]:
```
drop_col_with_unique_col =['circle_id', 'loc_og_t2o_mou', 'std_og_t2
                           'last_date_of_month_6', 'last_date_of_mon
                           'std_og_t2c_mou_6', 'std_og_t2c_mou_7',
                           'std_og_t2c_mou_8', 'std_ic_t2o_mou_6',
                           'std_ic_t2o_mou_7', 'std_ic_t2o_mou_8']
```

In [18]:
```
telecom_df_high_val_cust.shape
```

Out[18]:
```
(29953, 177)
```

In [19]:
```
telecom_df_high_val_cust.drop(drop_col_with_unique_col,axis=1,inpla
telecom_df_high_val_cust.shape
```

Out[19]:
```
(29953, 164)
```

In [20]:
```
def plot_box_chart(attribute):
    plt.figure(figsize=(20,16))
    df = telecom_df_high_val_cust
    plt.subplot(2,3,1)
    sns.boxplot(data=df, y=attribute+"_6",x="churn",hue="churn",
                showfliers=False,palette=("plasma"))
    plt.subplot(2,3,2)
    sns.boxplot(data=df, y=attribute+"_7",x="churn",hue="churn",
                showfliers=False,palette=("plasma"))
    plt.subplot(2,3,3)
    sns.boxplot(data=df, y=attribute+"_8",x="churn",hue="churn",
                showfliers=False,palette=("plasma"))
    plt.show()
```

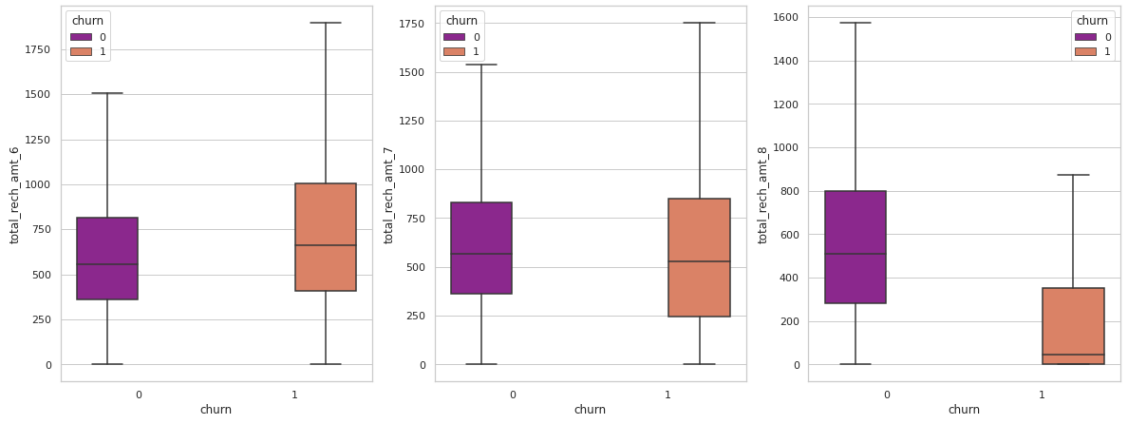
In [21]:
```
def plot_mean_bar_chart(df,columns_list):
    df_0 = df[df.churn==0].filter(columns_list)
    df_1 = df[df.churn==1].filter(columns_list)

    mean_df_0 = pd.DataFrame([df_0.mean()],index={'Non Churn'})
    mean_df_1 = pd.DataFrame([df_1.mean()],index={'Churn'})

    frames = [mean_df_0, mean_df_1]
    mean_bar = pd.concat(frames)

    mean_bar.T.plot.bar(figsize=(10,5),rot=0)
    plt.show()

    return mean_bar
```

Recharge Related Analysis

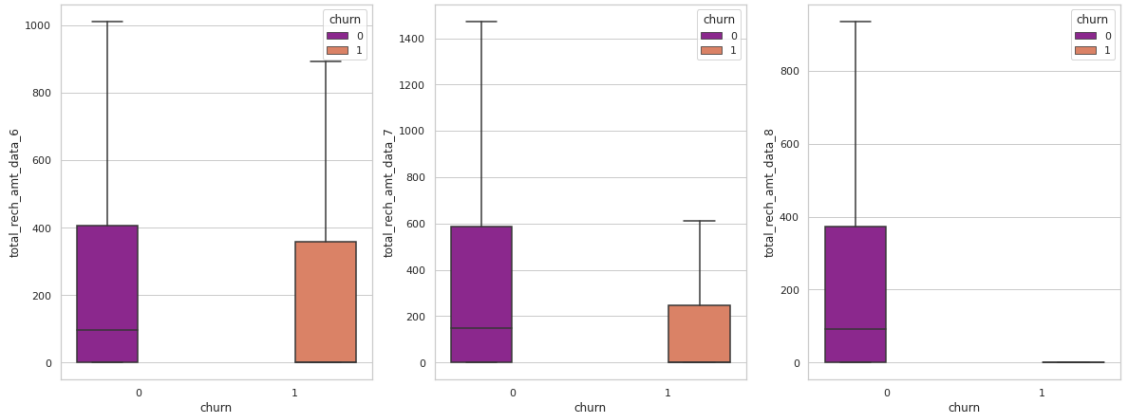
In [22]:
```
recharge_amnt_columns =  telecom_df_high_val_cust.columns[telecom_d
recharge_amnt_columns.tolist()
```

Out[22]:
```
['total_rech_amt_6',
 'total_rech_amt_7',
 'total_rech_amt_8',
 'max_rech_amt_6',
 'max_rech_amt_7',
 'max_rech_amt_8',
 'av_rech_amt_data_6',
 'av_rech_amt_data_7',
 'av_rech_amt_data_8',
 'total_rech_amt_data_6',
 'total_rech_amt_data_7',
 'total_rech_amt_data_8']
```

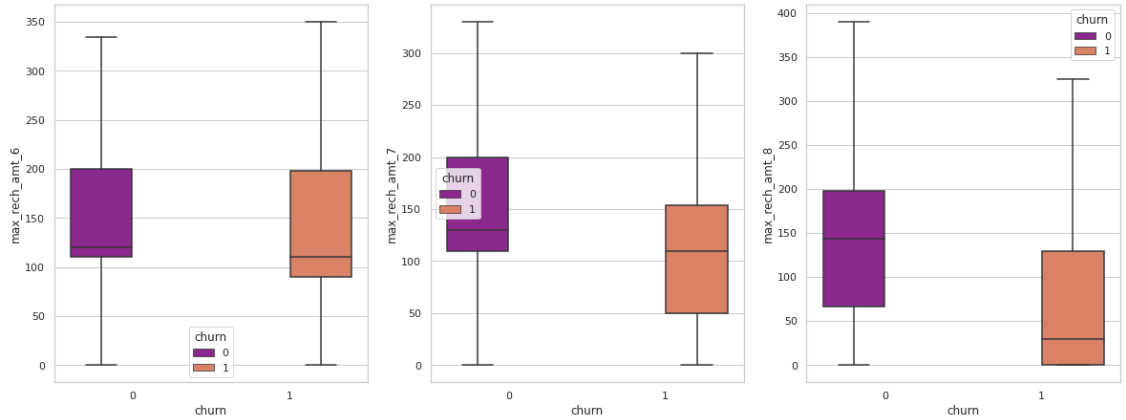
In [23]:
```
# Ploting for total recharge amount:
plot_box_chart('total_rech_amt')
```



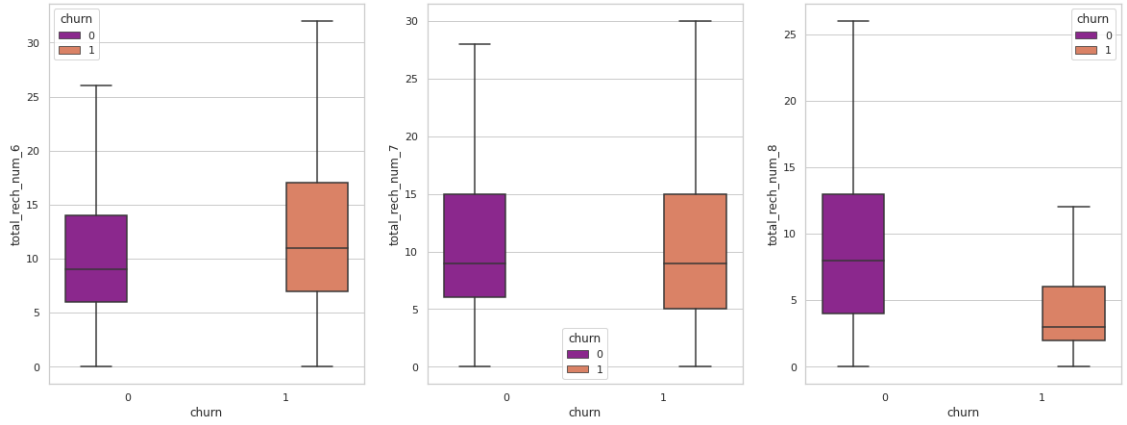
In [24]:
```
# Ploting for total recharge amount for data:
plot_box_chart('total_rech_amt_data')
```



In [25]:
```
# Ploting for maximum recharge amount for data:
plot_box_chart('max_rech_amt')
```

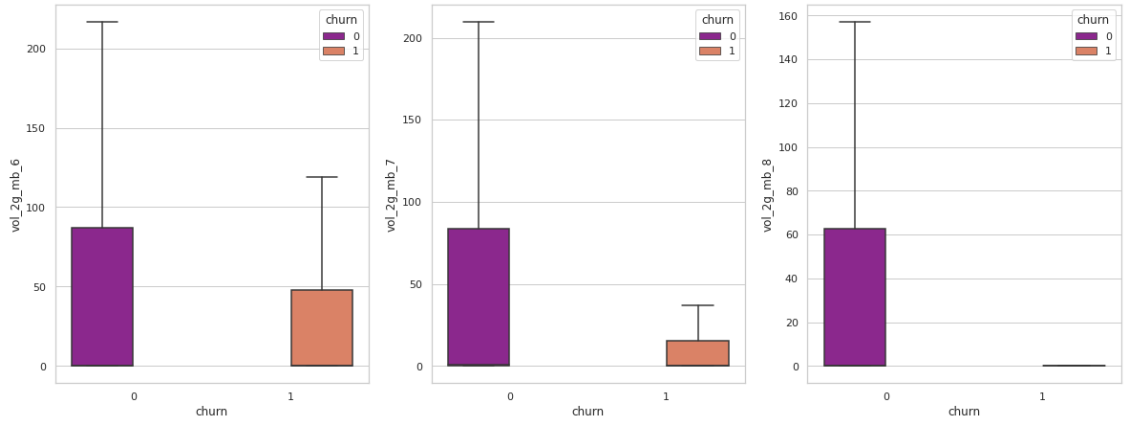


In [26]:
```
# Ploting for Total recharge for Number:
plot_box_chart('total_rech_num')
```

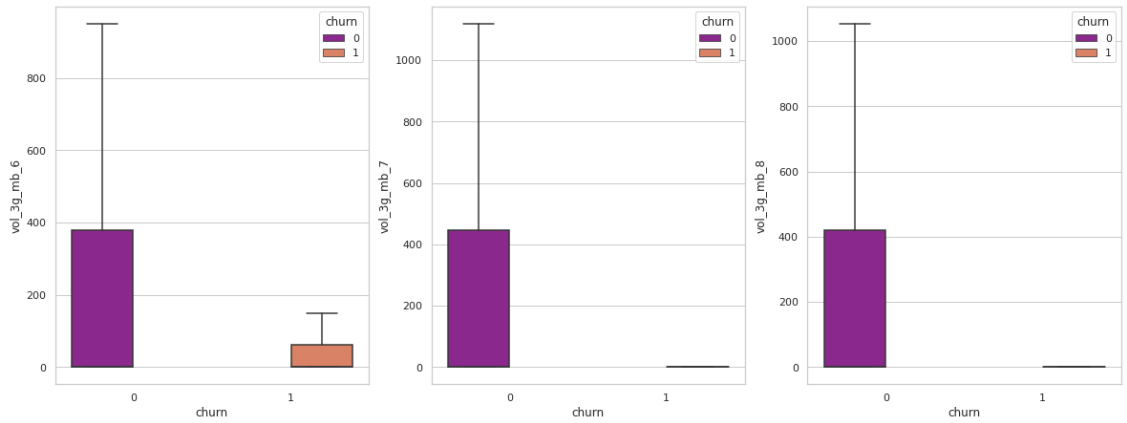


2G and 3G usage related analysis

In [27]:
```
# Ploting for volume of 2G and 3G usage columns:
plot_box_chart('vol_2g_mb')
```



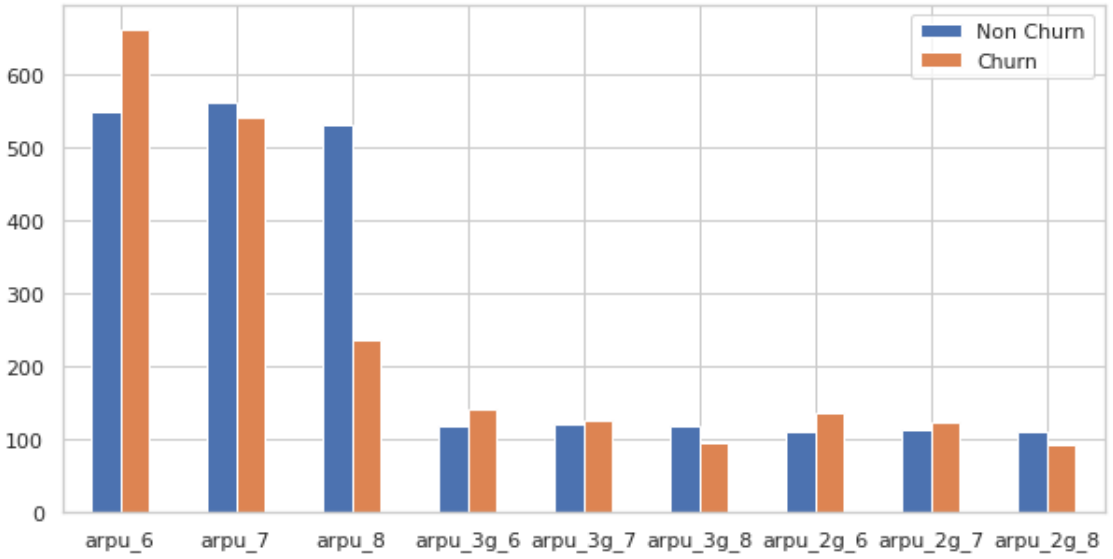
In [28]:
```
plot_box_chart('vol_3g_mb')
```



2G/3G usage is higher for non-churned customers indicating that churned customers might be from areas where 2G/3G service is not properly available.

Average Revenue Per User

In [29]:
```
arpu_cols = telecom_df_high_val_cust.columns[telecom_df_high_val_cus

plot_mean_bar_chart(telecom_df_high_val_cust, arpu_cols)
```

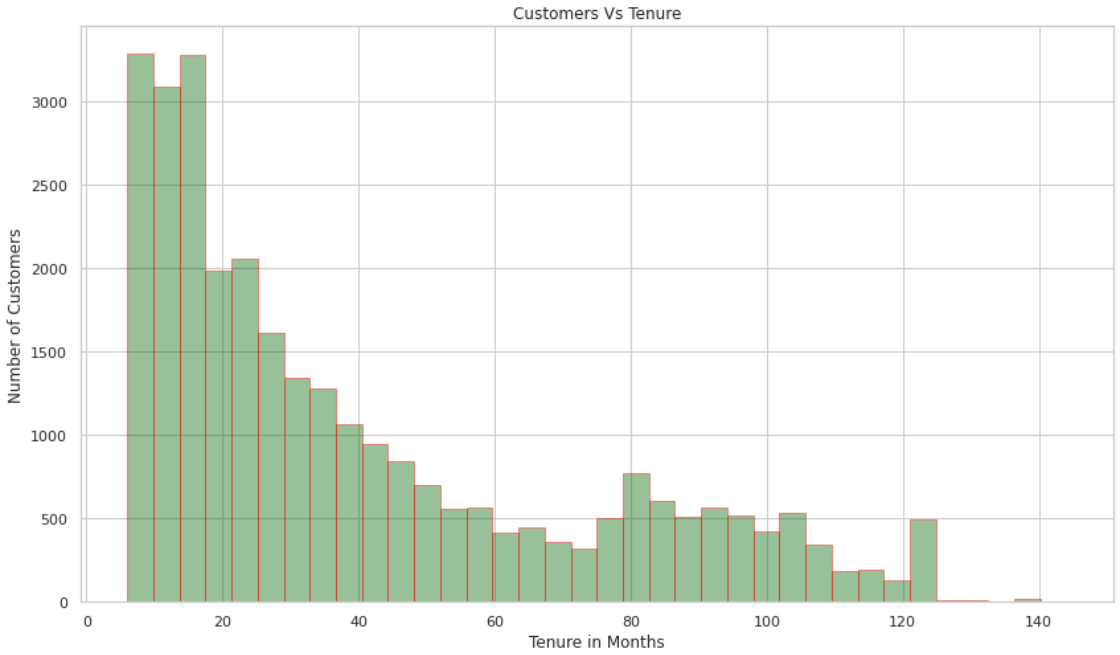


Out[29]:

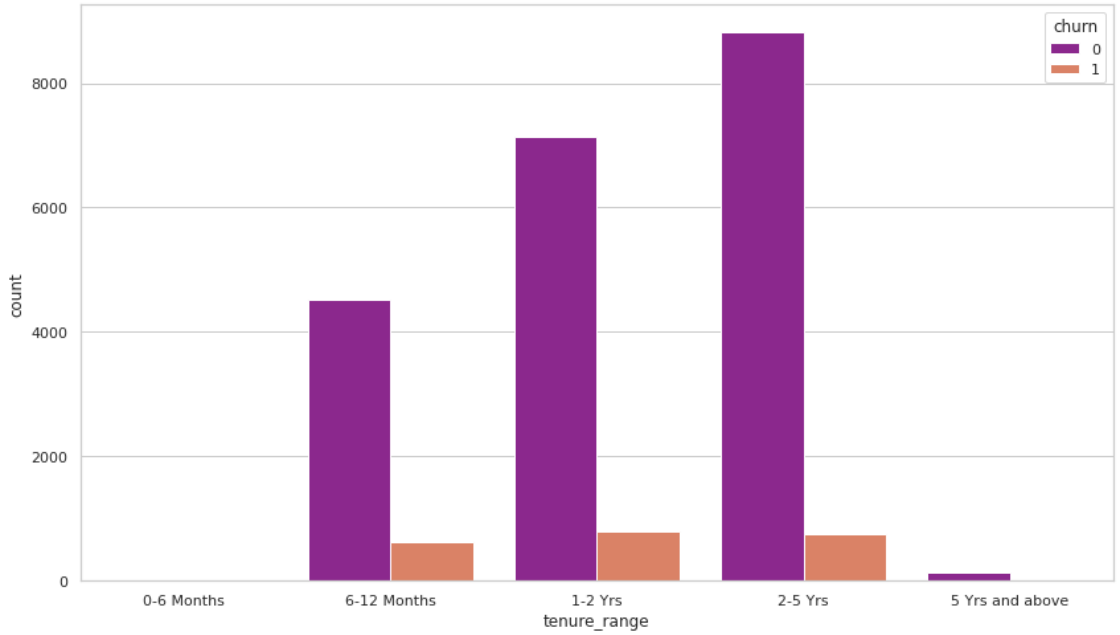
	arpu_6	arpu_7	arpu_8	arpu_3g_6	arpu_3g_7	arpu_3g_8	arpu_2g_6	arpu_2g_7
Non Churn	549.55	562.93	532.87	118.50	120.44	118.76	112.00	113.29
Churn	663.71	541.15	237.66	141.47	127.14	96.09	136.68	124.36

Tenure Analysis for Customers

In [30]:
```
tenure_data = telecom_df_high_val_cust.copy()
plt.figure(figsize=(14,8))
# aon --> Age on network - number of days the customer is using the
tenure_data['tenure'] = tenure_data['aon']/30
tenure_data['tenure'].head()
ax = sns.distplot(tenure_data['tenure'], hist=True, kde=False,
             bins=int(180/5), color = 'darkgreen',
             hist_kws={'edgecolor':'red'},
             kde_kws={'linewidth': 4})
ax.set_ylabel('Number of Customers')
ax.set_xlabel('Tenure in Months')
ax.set_title('Customers Vs Tenure')
plt.show()
```



In [31]:
```
tn_range = [0, 6, 12, 24, 60, 61]
tn_label = [ '0-6 Months', '6-12 Months', '1-2 Yrs', '2-5 Yrs', '5 Y
tenure_data['tenure_range'] = pd.cut(tenure_data['tenure'], tn_range

plt.figure(figsize=(14,8))
sns.countplot(x = 'tenure_range', hue = 'churn',data = tenure_data,p
plt.show()
```



In [ ]: