In...
```
from IPython.display import HTML

HTML('''<script>
code_show=true;
function code_toggle() {
 if (code_show){
 $('div.input').hide();
 } else {
 $('div.input').show();
 }
 code_show = !code_show
}
$( document ).ready(code_toggle);
</script>
<form action="javascript:code_toggle()"><input type="submit" value="Click here to tc
```

Out[1]: Click here to toggle on/off the raw code.

In [2]:
```
import pandas as pd
import numpy as np
import os
import warnings
import pickle, time
from datetime import datetime
import matplotlib.pyplot as plt
import seaborn as sns
```

Read Data and EDA

In [8]:
```
file = 'DS_MiniProject_ANON.csv'
path = r'D:\z7z8\DS_MiniProject_ANON'
filename = os.path.join(path,file)
df = pd.read_csv(filename)
wide_view(df.head(4))
```

	DATE_FOR	RTD_ST_CD	CustomerSegment	Tenure	Age	MART_STATUS	GENDER	CHANNEL1_6M
0	5/19/2014	ST_S0	1	16.175222	78.403833	MS_S0	F	0.0
1	5/17/2014	ST_S0	1	15.931554	70.989733	MS_S1	F	0.0
2	5/15/2014	ST_S0	1	15.937029	87.578371	MS_S2	M	0.0
3	5/16/2014	ST_S1	1	15.934292	68.438056	MS_S2	M	0.0

In [9]:
```
### data types
print(df.dtypes,end ='')
```

```
DATE_FOR                 object
RTD_ST_CD                object
CustomerSegment          object
```

In [10]:
```
### missing values
print(df.isna().sum(),end ='')
```

```
DATE_FOR                  0
RTD_ST_CD                 0
CustomerSegment           0
Tenure                    0
Age                       0
MART_STATUS               0
GENDER                    0
CHANNEL1_6M             809
CHANNEL2_6M             809
CHANNEL3_6M             809
CHANNEL4_6M             809
CHANNEL5_6M             809
METHOD1_6M              809
RECENT_PAYMENT          809
PAYMENTS_6M             809
CHANNEL1_3M               0
CHANNEL2_3M               0
CHANNEL3_3M               0
CHANNEL4_3M               0
CHANNEL5_3M               0
METHOD1_3M                0
PAYMENTS_3M               0
NOT_DI_3M                 0
NOT_DI_6M                 0
EVENT1_30_FLAG            0
EVENT2_90_SUM             0
LOGINS                    0
POLICYPURCHASECHANNEL     0
Call_Flag                 0
dtype: int64
```

In [1...
```
str_features = df.select_dtypes(exclude = [np.number]).columns.tolist()
num_features = df.select_dtypes(include = [np.number]).columns.tolist()
cat_features = str_features + [col for col in num_features if (df[col].nunique())
print('String    Features  : {}'.format(str_features))
print('Numerical Features  : {}'.format(num_features))
print('Categorical Features: {}'.format(cat_features))
```

```
String    Features  : ['DATE_FOR', 'RTD_ST_CD', 'CustomerSegment', 'MART_STATUS', 'GEND
ER']
Numerical Features  : ['Tenure', 'Age', 'CHANNEL1_6M', 'CHANNEL2_6M', 'CHANNEL3_6M', 'C
HANNEL4_6M', 'CHANNEL5_6M', 'METHOD1_6M', 'RECENT_PAYMENT', 'PAYMENTS_6M', 'CHANNEL1_3
M', 'CHANNEL2_3M', 'CHANNEL3_3M', 'CHANNEL4_3M', 'CHANNEL5_3M', 'METHOD1_3M', 'PAYMENTS
_3M', 'NOT_DI_3M', 'NOT_DI_6M', 'EVENT1_30_FLAG', 'EVENT2_90_SUM', 'LOGINS', 'POLICYPUR
CHASECHANNEL', 'Call_Flag']
Categorical Features: ['DATE_FOR', 'RTD_ST_CD', 'CustomerSegment', 'MART_STATUS', 'GEND
ER', 'RECENT_PAYMENT', 'NOT_DI_3M', 'NOT_DI_6M', 'EVENT1_30_FLAG', 'POLICYPURCHASECHANN
EL', 'Call_Flag']
```

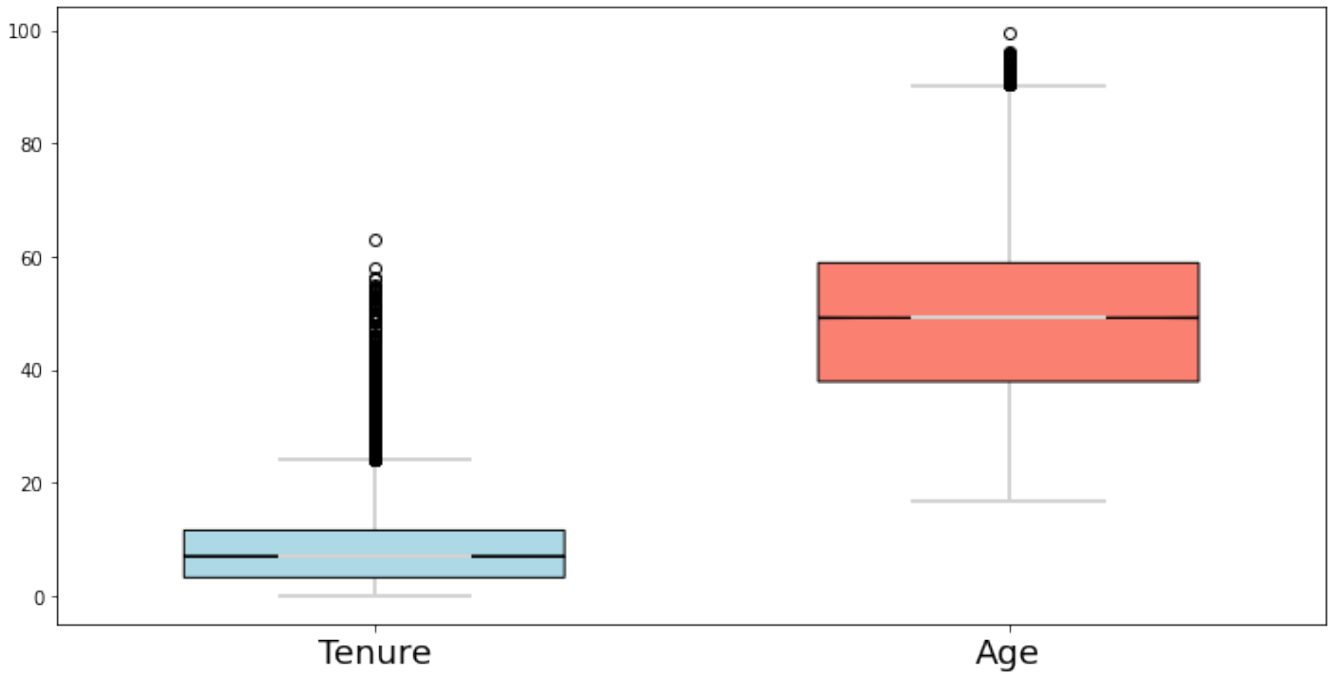
Check and Visualize outliers for Numerical features

In [12]:

	Tenure	Age	CHANNEL1_6M	CHANNEL2_6M	CHANNEL3_6M	CHANNEL4_6M	CHANN
count	130086.000000	130086.000000	129277.000000	129277.000000	129277.000000	129277.000000	12927
mean	8.583048	48.842201	0.261810	0.948467	0.811652	0.401216	(
std	7.297112	14.038089	1.092346	1.880123	1.824188	1.065413	1
min	0.032854	16.689938	0.000000	0.000000	0.000000	0.000000	(
25%	3.249829	37.878166	0.000000	0.000000	0.000000	0.000000	(
50%	7.006160	49.147159	0.000000	0.000000	0.000000	0.000000	(
75%	11.610540	58.858316	0.000000	1.000000	0.000000	0.000000	(
max	63.091034	99.348392	12.000000	53.000000	26.000000	18.000000	29

In [1...

```
# df_temp = df[[x for x in num_features if x not in cat_features]]
# df_temp.columns
plot_box(df[['Tenure', 'Age']])
plot_box(df[['METHOD1_3M', 'PAYMENTS_3M', 'EVENT2_90_SUM', 'LOGINS']],colors =['s
plot_box(df[['CHANNEL1_6M', 'CHANNEL2_6M', 'CHANNEL3_6M', 'CHANNEL4_6M',
       'CHANNEL5_6M','METHOD1_6M','PAYMENTS_6M']].fillna(-9),colors =['salmon']*7
fig = plot_box(df[['CHANNEL1_3M', 'CHANNEL2_3M', 'CHANNEL3_3M', 'CHANNEL4_3M',
       'CHANNEL5_3M']],colors =['salmon']*5,rot = 60)
```



Check and Visualize Colinearity for Numerical Features

In [15]:
```
df['CHANNEL1_6M'].value_counts(normalize =True)
```

Out[15]:
```
0.0     0.928974
1.0     0.012833
2.0     0.012562
6.0     0.011704
5.0     0.009917
3.0     0.009777
4.0     0.009453
7.0     0.002777
8.0     0.001091
9.0     0.000541
10.0    0.000255
11.0    0.000077
12.0    0.000039
Name: CHANNEL1_6M, dtype: float64
```

In...
```
df_temp =df[['Tenure', 'Age',  'CHANNEL1_3M', 'CHANNEL2_3M', 'CHANNEL3_3M', 'CHANNEL
corr_matrix = pd.DataFrame(np.corrcoef(df_temp.values, rowvar=False).astype('float32
                                    columns=df_temp.columns, index=df_temp.columns).a
corr_matrix = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.b
collinear = {k: corr_matrix.loc[(corr_matrix[k] > 0.95)&(corr_matrix[k] < 1), k].ind
```

In [17]:
```
fig = plot_heatmap(corr_matrix,cmap = 'Blues')
```

Bar Charts for Categorical Variables

In [...

```
df['Weekday'] = df['DATE_FOR'].apply(lambda x: pd.to_datetime(x.zfill(10),format =

display_sides(df.groupby('GENDER')[['Call_Flag']].agg([np.mean,np.size]),
                df.groupby('CustomerSegment')[['Call_Flag']].agg([np.mean,np.size])
              df.groupby('RTD_ST_CD')[['Call_Flag']].agg([np.mean,np.size]).iloc[:
              df.groupby('MART_STATUS')[['Call_Flag']].agg([np.mean,np.size]),
              df.groupby('Weekday')[['Call_Flag']].agg([np.mean,np.size]))
```

	Call_Flag	
	mean	size
GENDER		
F	0.034864	60177
M	0.038135	69909

	Call_Flag	
	mean	size
CustomerSegment		
1	0.033655	95885
2	0.046670	25434
3	0.048963	5065
NONE	0.027553	3702

	Call_Flag	
	mean	size
RTD_ST_CD		
ST_S0	0.036184	8512
ST_S1	0.047785	1151
ST_S10	0.022337	1164
ST_S11	0.042978	4258
ST_S12	0.028194	2270

	Call_Flag	
	mean	size
MART_STATUS		
MS_S0	0.047805	4853
MS_S1	0.039154	45717
MS_S2	0.031927	58164
MS_S3	0.040012	13446
MS_S4	0.043891	7906

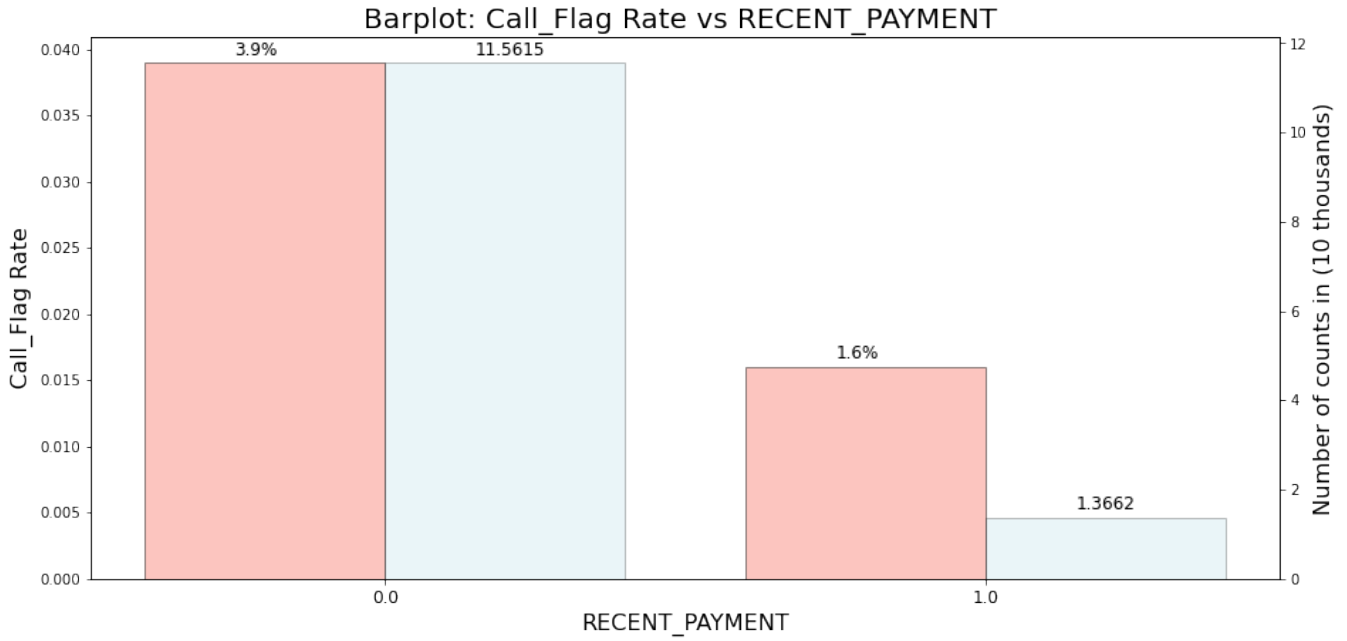
	Call_Flag	
	mean	size
Weekday		
0	0.031909	16704
1	0.033819	30249
2	0.038380	16884
3	0.038378	16624
4	0.040133	16570
5	0.038747	16053
6	0.037349	17002

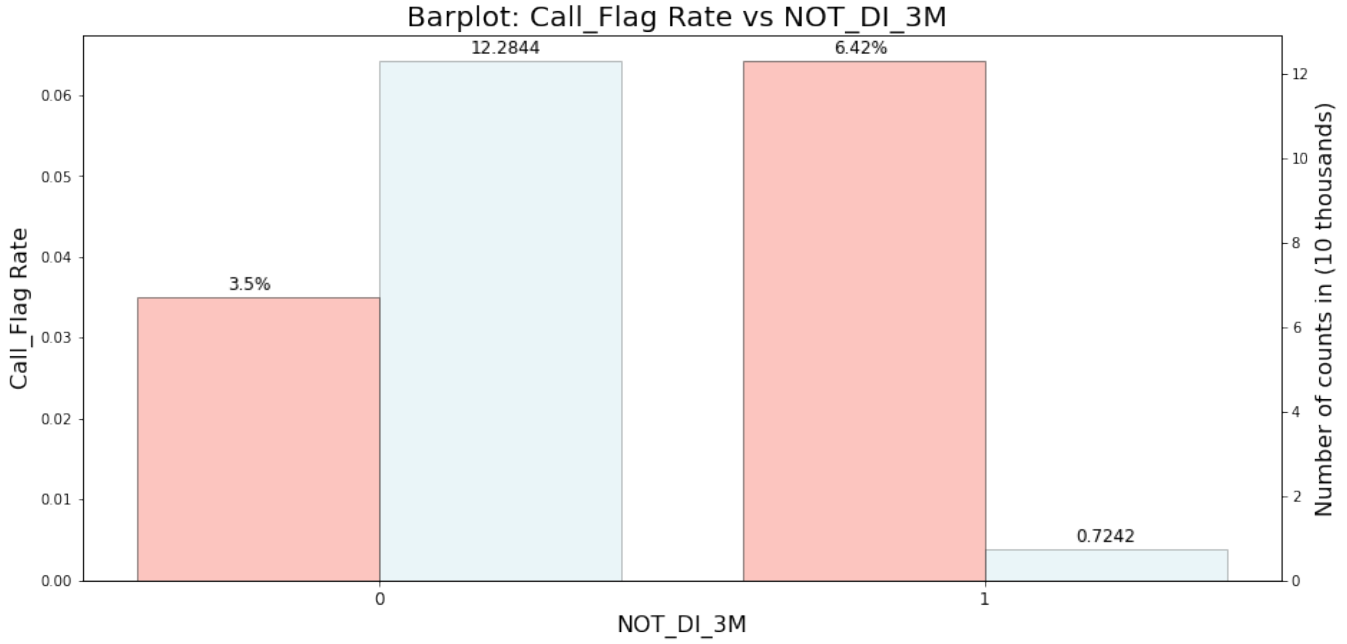
In...

```
plot_bar(df.groupby('CustomerSegment')[['Call_Flag']].agg([np.mean,np.size]),feature
plot_bar(df.groupby('Weekday')[['Call_Flag']].agg([np.mean,np.size]),feature = 'Week
plot_bar(df.groupby('GENDER')[['Call_Flag']].agg([np.mean,np.size]),feature = 'GENDE
plot_bar(df.groupby('MART_STATUS')[['Call_Flag']].agg([np.mean,np.size]),feature = '
plot_bar(df.groupby('RTD_ST_CD')[['Call_Flag']].agg([np.mean,np.size]),feature = 'RT
```

In...

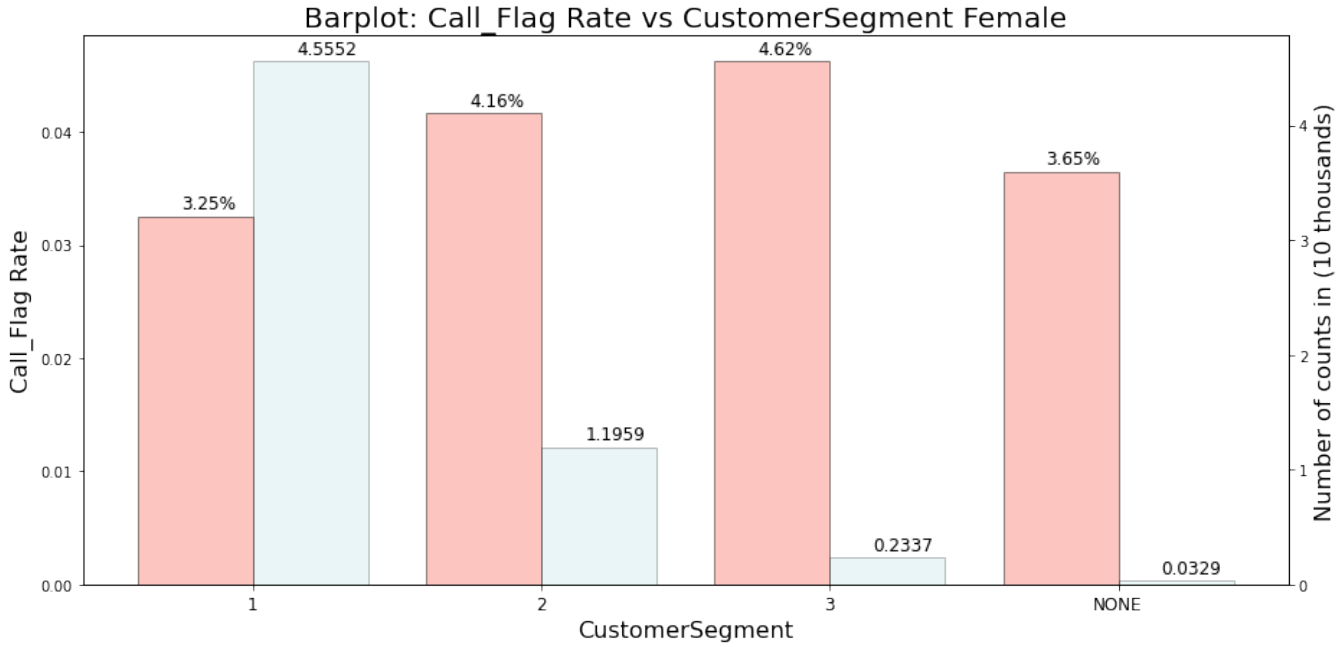
```
plot_bar(df.groupby('RECENT_PAYMENT')[['Call_Flag']].agg([np.mean,np.size]),feature
plot_bar(df.groupby('NOT_DI_3M')[['Call_Flag']].agg([np.mean,np.size]),feature = 'N
plot_bar(df.groupby('NOT_DI_6M')[['Call_Flag']].agg([np.mean,np.size]),feature = 'N
plot_bar(df.groupby('EVENT1_30_FLAG')[['Call_Flag']].agg([np.mean,np.size]),feature

plot_bar(df.groupby('POLICYPURCHASECHANNEL')[['Call_Flag']].agg([np.mean,np.size]),
```

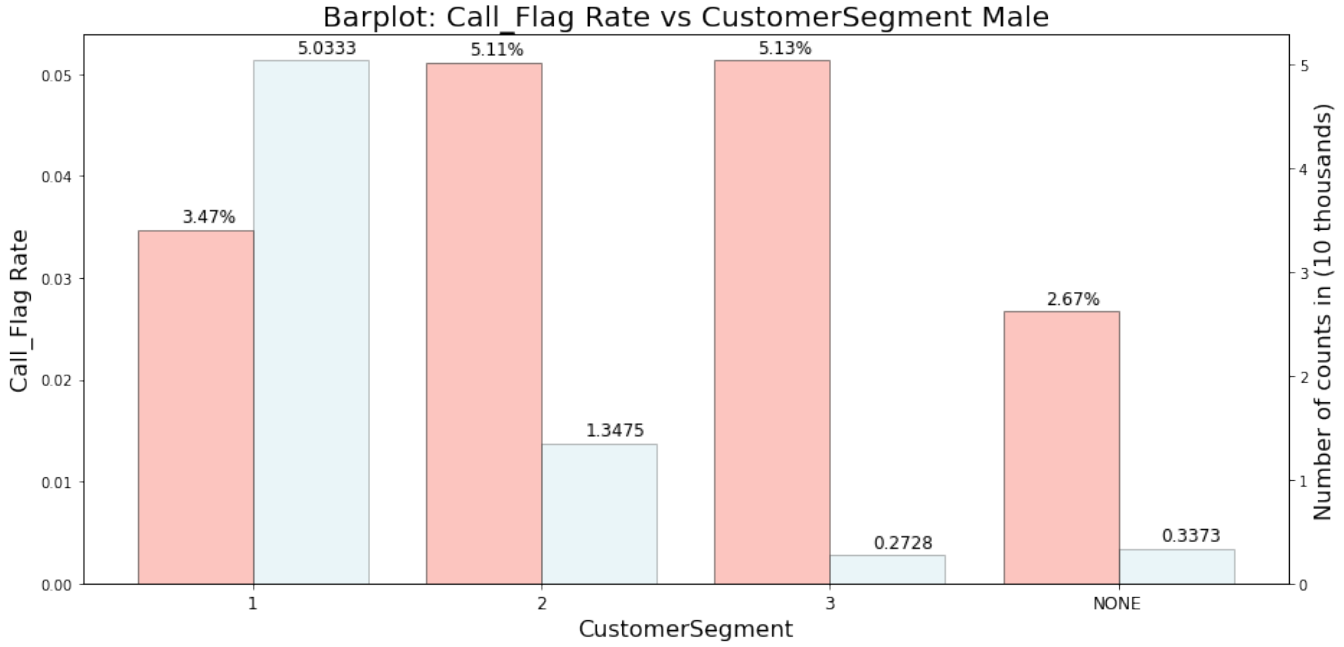




Bar Charts for Interaction effects

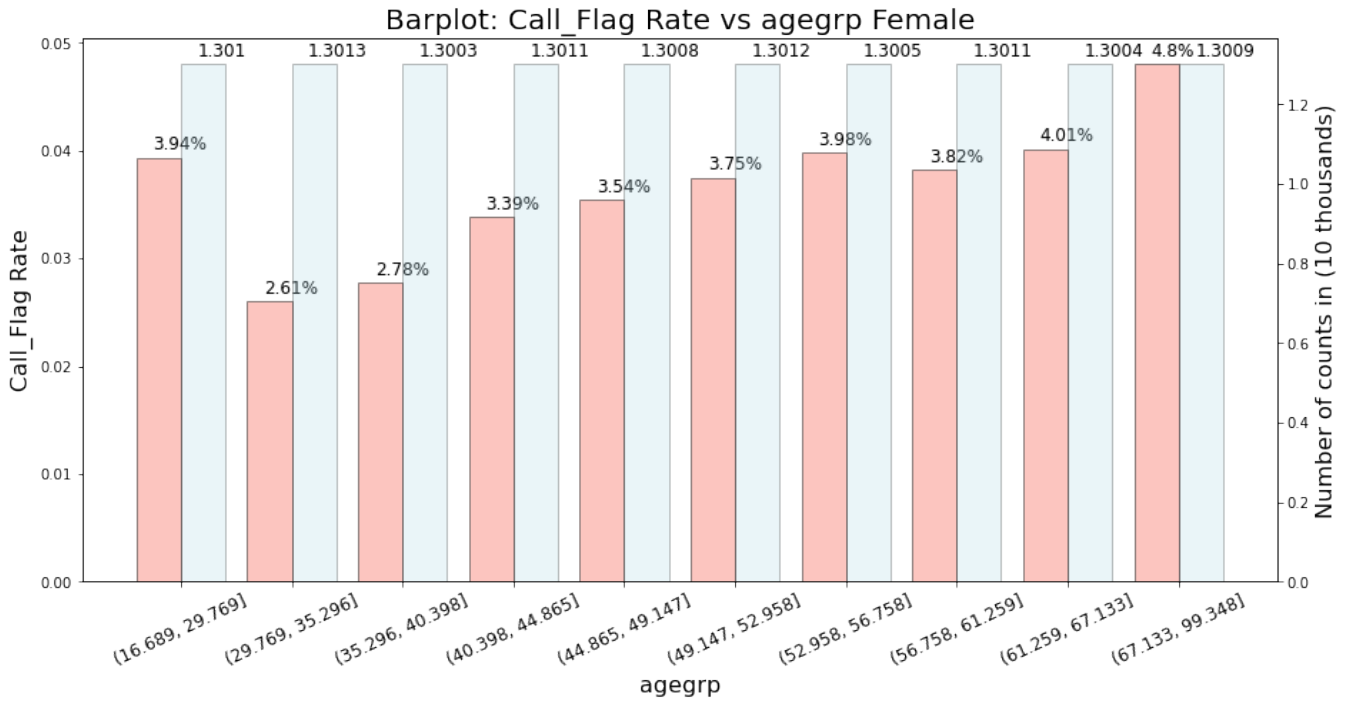
In [...]:
```
plot_bar(df[df.GENDER =='F'].groupby('CustomerSegment')[['Call_Flag']].agg([np.mean,
plot_bar(df[df.GENDER =='M'].groupby('CustomerSegment')[['Call_Flag']].agg([np.mean,
```

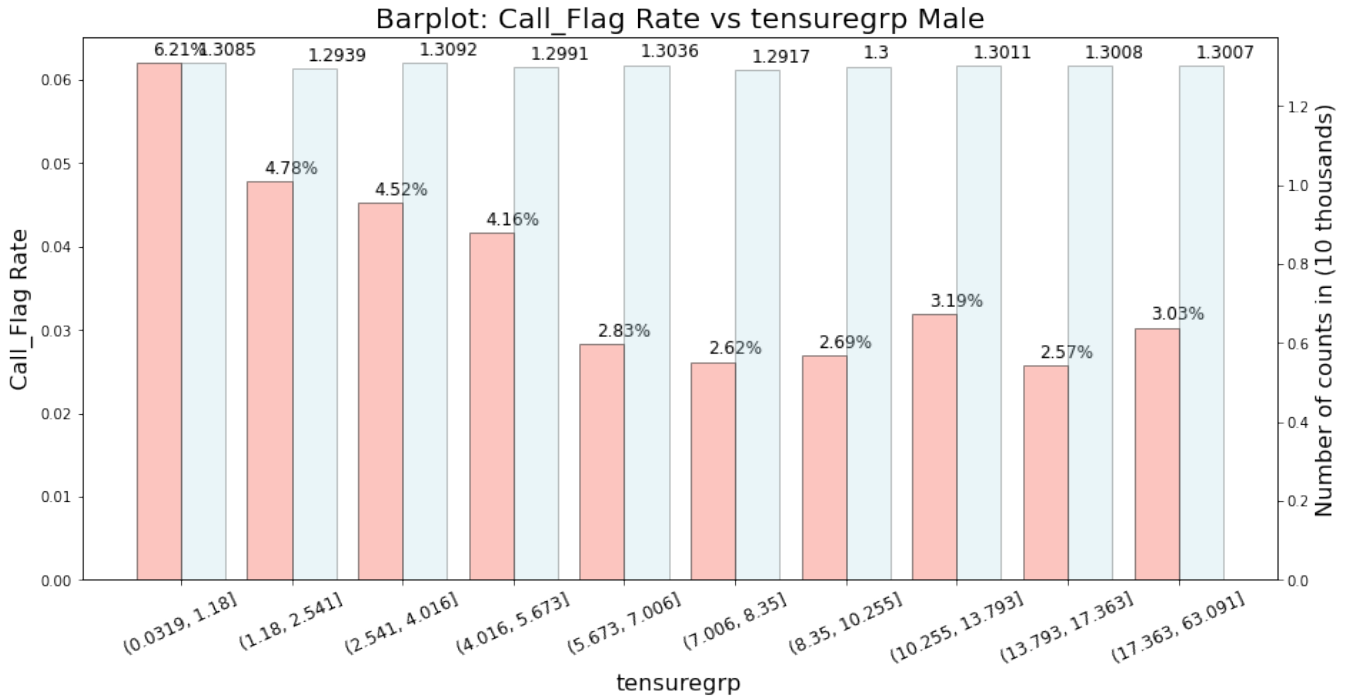




Out[...]:
```
(<Figure size 1080x504 with 2 Axes>,
 <AxesSubplot:title={'center':'Barplot: Call_Flag Rate vs CustomerSegment Male'}, x
label='CustomerSegment', ylabel='Call_Flag Rate'>)
```

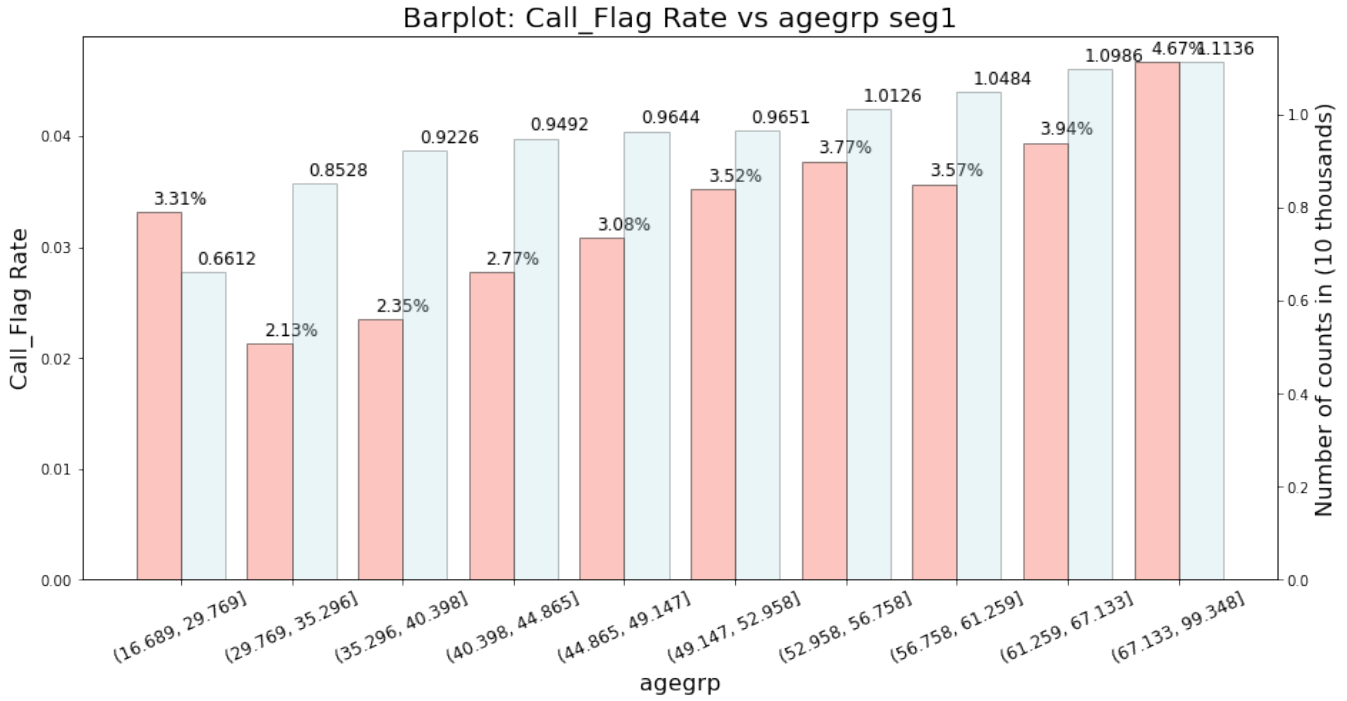
In [...]:
```
df['agegrp'] = pd.qcut(df['Age'],q =10,duplicates = 'drop')
df['tensuregrp'] = pd.qcut(df['Tenure'],q =10,duplicates = 'drop')

plot_bar(df.groupby('agegrp')[['Call_Flag']].agg([np.mean,np.size]),feature = 'agegr
plot_bar(df.groupby('tensuregrp')[['Call_Flag']].agg([np.mean,np.size]),feature = 't
```

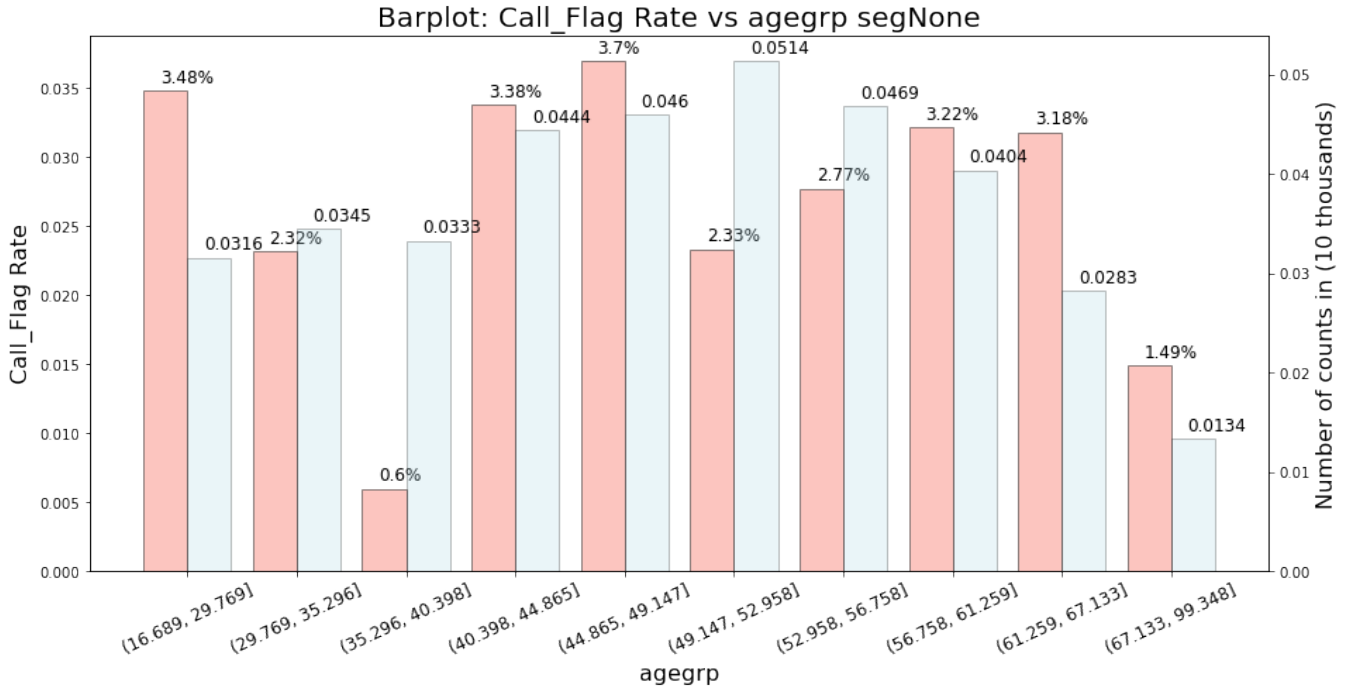

Barplot: Call_Flag Rate vs agegrp Female
Call_Flag Rate
Number of counts in (10 thousands)
agegrp
3.94%
2.61%
2.78%
3.39%
3.54%
3.75%
3.98%
3.82%
4.01%
4.8%
1.301
1.3013
1.3003
1.3011
1.3008
1.3012
1.3005
1.3011
1.3004
1.3009
(16.689, 29.769]
(29.769, 35.296]
(35.296, 40.398]
(40.398, 44.865]
(44.865, 49.147]
(49.147, 52.958]
(52.958, 56.758]
(56.758, 61.259]
(61.259, 67.133]
(67.133, 99.348]


Barplot: Call_Flag Rate vs tensuregrp Male
Call_Flag Rate
Number of counts in (10 thousands)
tensuregrp
6.21%
4.78%
4.52%
4.16%
2.83%
2.62%
2.69%
3.19%
2.57%
3.03%
1.3085
1.2939
1.3092
1.2991
1.3036
1.2917
1.3
1.3011
1.3008
1.3007
(0.0319, 1.18]
(1.18, 2.541]
(2.541, 4.016]
(4.016, 5.673]
(5.673, 7.006]
(7.006, 8.35]
(8.35, 10.255]
(10.255, 13.793]
(13.793, 17.363]
(17.363, 63.091]

I...

```
plot_bar(df[df.CustomerSegment =='1'].groupby('agegrp')[['Call_Flag']].agg([np.mean,
plot_bar(df[df.CustomerSegment =='NONE'].groupby('agegrp')[['Call_Flag']].agg([np.me
plot_bar(df[df.CustomerSegment.isin(['2','3'])].groupby('agegrp')[['Call_Flag']].agg

plot_bar(df[df.CustomerSegment =='1'].groupby('tensuregrp')[['Call_Flag']].agg([np.m
plot_bar(df[df.CustomerSegment =='NONE'].groupby('tensuregrp')[['Call_Flag']].agg([n
plot_bar(df[df.CustomerSegment.isin(['2','3'])].groupby('tensuregrp')[['Call_Flag']]
```





In [111]:
```
# df.to_csv(r'D:\z7z8\DS_MiniProject_ANON\processed_data.csv',index = False)
```

Functions

In [...
```
def wide_view(df):
    from IPython.core.display import HTML
    display(HTML(df.to_html()))

def display_sides(*args):
    from IPython.core.display import display_html
    html_str = ''
    for arg in args:
        html_str += arg.to_html()
    display_html(html_str.replace('table','table style = display:inline'),raw = Tru
```

In...
```
def plot_bar(df,feature,target = 'Call_Flag',show_counts = False,show_annotate = Tru
    fig,ax = plt.subplots(figsize = (15,7))
    for x,h in zip([x - 0.4 for x in list(np.arange(1,df.shape[0]+1))],df[(target,
        ax.bar(x,h,color = 'salmon',alpha = 0.45,width = 0.4,align = 'edge',edgecolo
        if show_annotate:
            ax.annotate(str(round(100*h,2))+'%',(x+0.15,h),
                        textcoords="offset points",
                        xytext =(0,9),ha = 'left',va = 'center',size = 12)
    ax.set_xticks(list(np.arange(1,df.shape[0]+1)))
    ax.set_xticklabels(list(df.index),size =12,rotation = rotation)
    ax.set_title('Barplot: {} Rate vs {} '.format(target,feature)+title_extra,size =
#     ax.set_ylim([0,0.66])
    ax.set_ylabel('{} Rate'.format(target), size = 16)
    ax.set_xlabel('%s' % feature, size = 16)
    if show_counts:
        ax2 = ax.twinx()
        for x,z in zip([x for x in list(np.arange(1,df.shape[0]+1))],df[(target, 's
            ax2.bar(x,z,color = 'lightblue',alpha = 0.25,width = 0.4,align = 'edge',
            if show_annotate:
                ax2.annotate(str(z),(x+0.15,z),
                             textcoords="offset points",
                             xytext =(0,9),ha = 'left',va = 'center',size = 12)
    #     ax2.set_ylim([0,350])
        ax2.set_ylabel('Number of counts in (10 thousands)',size = 16)
    plt.show()
    return fig,ax
```

In [...
```
def plot_box(df,colors = ['lightblue', 'salmon'],rot = 0):
    fig,ax = plt.subplots(figsize =(12,6))
    bp = ax.boxplot([df[col] for col in df.columns],labels =df.columns.tolist(),pat
    ax.set_xticklabels(df.columns.tolist(),size =18,rotation = rot)

    for patch, color in zip(bp['boxes'], colors):
        patch.set_facecolor(color)
    for whisker in bp['whiskers']:
```

In [ ]:

In [ ]: