Customers Retention Analysis Project

Credit Card Retention

As a financial analyst at Xbank working with the marketing department to help optimize the marketing campaigns and maximize return on investment.

- Stakeholder: Marketing team and my manager
- Question: Perform analysis on the existing data to better understand how to increase customer retention.
- Research:
  - What are KPIs for credit cards?
  - What are the top credit cards on the market and why are they successful?
- Some definitions:
  - credit limit is the maximum amount of the money a creditor allows a credit card user to spend.
  - Basically, it is a cap on the card for which the customer cannot go beyond.
  - Attrition: Employees or customers lost and not replaced over a period of time.
  - Utilization: the amount of money a person owes divided by their credit limit.
  - Open to buy: credit limit minus the present balance in the account.
- Customer churn: Customer churn is the percentage of customers that stopped using your company's product or service during a certain time frame. calculate churn rate by dividing the number of customers you lost during that time period -- say a quarter -- by the number of customers you had at the beginning of that time period.

Internal meetings with people in the marketing team about what they think the cause of customers not holding credit cards for a longer time is.

- Problem statement:
  - What marketing retention campaigns could we implement to help reduce customer churn?
- Audience: Marketing department
- Delivery:presentation
- Time: 02 weeks

By: Alexis Mekueko

```
Data Scientist/Analyst
```

Data Summary

This dataset consists of 10127 customers mentioning their age, salary, marital_status, credit card limit, credit card category, etc. There are nearly 23 features. We have only 16.07% of customers who have churned. Thus, it's a bit difficult to train our model to predict churning customers.

Data Dictionary

Features Names ---------------------------> Description

CLIENTNUM -------------------> Client number. Unique identifier for the customer holding the account

Attrition_Flag -------------------> Internal event (customer activity) variable - if the account is closed then 1 else 0

Customer_Age -------------------> Demographic variable - Customer's Age in Years

Gender -------------------> Demographic variable - M=Male, F=Female

Dependent_count -------------------> Demographic variable - Number of dependents

Education_Level -------------------> Demographic variable - Educational Qualification of the account holder (example: high school, college graduate, etc.)

Marital_Status -------------------> Demographic variable - Married, Single, Divorced, Unknown

Income_Category -------------------> Demographic variable - Annual Income Category of the account holder (< $40K$,40K - 60K, $60K-$80K, $80K-$120K, >

Card_Category -------------------> Product Variable - Type of Card (Blue, Silver, Gold, Platinum)

Months_on_book -------------------> Period of relationship with bank

Total_Relationship_count -------------------> Total no. of products held by the customer

Months_Inactive_12_mon -------------------> No. of months inactive in the last 12 months

Contacts_Count_12_mon -------------------> No. of Contacts in the last 12 months

Credit_Limit -------------------> Credit Limit on the Credit Card

Total_Revolving_Bal -------------------> Total Revolving Balance on the Credit Card

Avg_Open_To_Buy -------------------> Open to Buy Credit Line (Average of last 12 months)

Total_Amt_Chng_Q4_Q1 -------------------> Change in Transaction Amount (Q4 over Q1)

Total_Trans_Amt -------------------> Total Transaction Amount (Last 12 months)

Total_Trans_Ct -------------------> Total Transaction Count (Last 12 months)

Total_Ct_Chng_Q4_Q1 -------------------> Change in Transaction Count (Q4 over Q1)

Avg_Utilization_Ratio -------------------> Average Card Utilization Ratio

Naive_Bayes_Classifier_attribution -------------------> Naive Bayes

In [316…

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
#import plotly.plotly as py
from tabulate import tabulate
import plotly.graph_objs as go
from plotly.offline import iplot
sns.set()
pd.options.display.max_columns = 999
```

Data Cleaning

In [137…

```
df = pd.read_csv("credit card cutomers.csv")
dataset = df
dataset.shape # (10127 records or customer, 23 features or coloumns)
dataset.head(5) # checking top rows
#dataset.tail(5) # checking end rows
#dataset.columns # checking columns
#dataset.loc[dataset['column'] = 'value of the row to capture'] #print(dataset.loc[4])
```

Out[137]:

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Sta
0	768805383	Existing Customer	45	M	3	High School	Marı
1	818770008	Existing Customer	49	F	5	Graduate	Sir
2	713982108	Existing Customer	51	M	3	Graduate	Marı
3	769911858	Existing Customer	40	F	4	High School	Unknc
4	709106358	Existing Customer	40	M	3	Uneducated	Marı

In [30]:

```
#Looks like column "CLIENTNUM" is a unique identifier...
#Let's check for any duplicated or missing information
dataset['CLIENTNUM'].nunique()
dataset.drop_duplicates(inplace=True) #dataset["col_name"].drop_duplicates(keep="first
#dataset.drop_duplicates("col_name", keep="first", inplace=False, ignore_index=False)
#import pandas as pd; pd.__version__ # checking pandas version
dataset.shape #checking shape again to see if nothing has changed
```

Out[30]: (10127, 23)

In [155…

```
#dataset.dtypes # check data type for all column
#dataset.dtypes[dataset.dtypes == 'int64'] # check all columns that have a specific da
#dataset.dtypes[dataset.dtypes == 'object']
#dataset.dtypes[dataset.dtypes == 'float64']

#dataset.column_name.dtype #check data type of one specific column

#print(tabulate(dataset.info(verbose=True)))
```

In [124…

```
#dataset.isnull().sum() #checking for missing value
#dataset.isnull().values.any()
#dataset.isna().sum()
#dataset[dataset["Marital_Status"].isna()]
findRows = dataset[(dataset == "Unknown").any(axis=1)]
findRows
#(pd.DataFrame(findRows)).shape # there are 3046 rows with values of 'Unknown'
#(pd.DataFrame(findRows)).shape[0] # there are 3046 rows with values of 'Unknown'
#len(findRows)
#len(findRows.index)
findRows.count()
#(pd.DataFrame(findRows)).shape[:]
#len(pd.DataFrame(findRows).columns) #count number of columns of df
#dataset[(dataset == "Unknown").columns]
#dataset.head(1)
#dataset[dataset.columns == "Unknown"]
#dataset[dataset == 'Unknown'].dropna(how='all') #print all rows with only value "Unkn
#dataset[dataset.isin(['Unknown'])] # print all rows with value "Unknown"
#dataset.columns.values #shows columns names
dataset[dataset=='Unknown'].any() #print coumns with specific value "Unknows", False,
#dataset['CLIENTNUM'].value_counts()["Unknown"] ##count occurrences of the value chara
#dataset['CLIENTNUM'].value_counts()[8] ##count occurrences of the value numeric 8 in
#dataset['CLIENTNUM'].value_counts() ##count occurrences of every unique value in the
#dataset['CLIENTNUM'].value_counts(dropna=False)
#dataset.groupby('CLIENTNUM').count()
```

Out[124]:

CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Stat

In [129…

```
dataset[(dataset == " ").any(axis=1)]
#findRows
dataset[(dataset == "NaN").any(axis=1)]
```

Out[129]:

CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Stat

In [140…

```
#dataset[dataset=='NaN'].any()
#dataset[(dataset == "NaN").columns]
#pd.isna(dataset)
#pd.isna(dataset).any()
#pd.isna(dataset).sum()
#dataset['Marital_Status'] = dataset['Marital_Status'].fillna("Unknown")
```

In [143…

```
### Data Transformation
```

Out[143]:

```
46.32596030413745
```

In [154…

```
#dataset['Customer_Age'].min()  #minimum age 26

#dataset['Customer_Age'].max()   # eldest client 73 years old

#dataset['Customer_Age'].mean()   # average client has 46 years old
# Let's set up bin for age range
bin1 = [25, 30, 40, 50, 60, 70, 80]
label1 = ['20s', '30s', '40s', '50s','60s','70s']
dataset['Customer_Age_Range'] = pd.cut(dataset['Customer_Age'], bins = bin1, labels =
dataset[dataset['Customer_Age']== 40].head(5)
```

Out[154]:

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_S
3	769911858	Existing Customer	40	F	4	High School	Unk
4	709106358	Existing Customer	40	M	3	Uneducated	M
150	711009708	Existing Customer	40	M	3	High School	
259	779656908	Existing Customer	40	M	1	Graduate	M
271	717806133	Existing Customer	40	M	2	Uneducated	Div

Data Summary

In [159…

```
#churn rate
dataset['Attrition_Flag'].value_counts() #Existing Customer    8500, Attrited Customer
dataset['Attrition_Flag'].value_counts()['Attrited Customer']/ dataset.shape[0]
#np.append(value) #to add value to a list
dataset.describe()
```

Out[159]:

	CLIENTNUM	Customer_Age	Dependent_count	Months_on_book	Total_Relationship_Count	M
count	1.012700e+04	10127.000000	10127.000000	10127.000000	10127.000000	
mean	7.391776e+08	46.325960	2.346203	35.928409	3.812580	
std	3.690378e+07	8.016814	1.298908	7.986416	1.554408	
min	7.080821e+08	26.000000	0.000000	13.000000	1.000000	
25%	7.130368e+08	41.000000	1.000000	31.000000	3.000000	
50%	7.179264e+08	46.000000	2.000000	36.000000	4.000000	
75%	7.731435e+08	52.000000	3.000000	40.000000	5.000000	
max	8.283431e+08	73.000000	5.000000	56.000000	6.000000	

In [171…

```
# looking mean and median to find if there any outlier
#round(np.mean(dataset['Credit_Limit']), 2) ## mean = 8631.95
#round(np.median(dataset['Credit_Limit']), 2) ## median = 4549.0
```

```
round(np.mean(dataset['Total_Relationship_Count']), 2) # meadian = 3.81
m =round(np.median(dataset['Total_Relationship_Count']), 2) # median = 4.0
#print('The median credit limit is',m,'$')
```

```
The median credit limit is 4.0 $
```

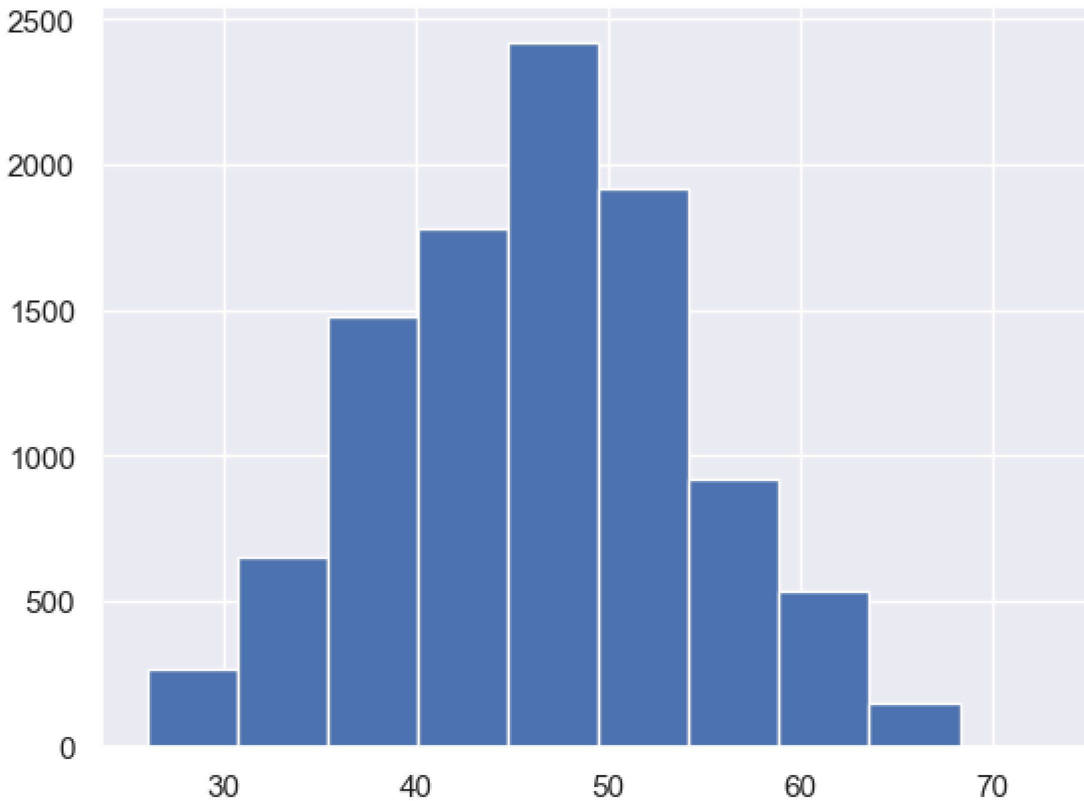
Data Distributions

In [172…

```
plt.hist(dataset["Customer_Age"])
```

Out[172]:

```
(array([2.650e+02, 6.540e+02, 1.478e+03, 1.778e+03, 2.422e+03, 1.920e+03,
        9.210e+02, 5.350e+02, 1.520e+02, 2.000e+00]),
 array([26. , 30.7, 35.4, 40.1, 44.8, 49.5, 54.2, 58.9, 63.6, 68.3, 73. ]),
 <BarContainer object of 10 artists>)
```

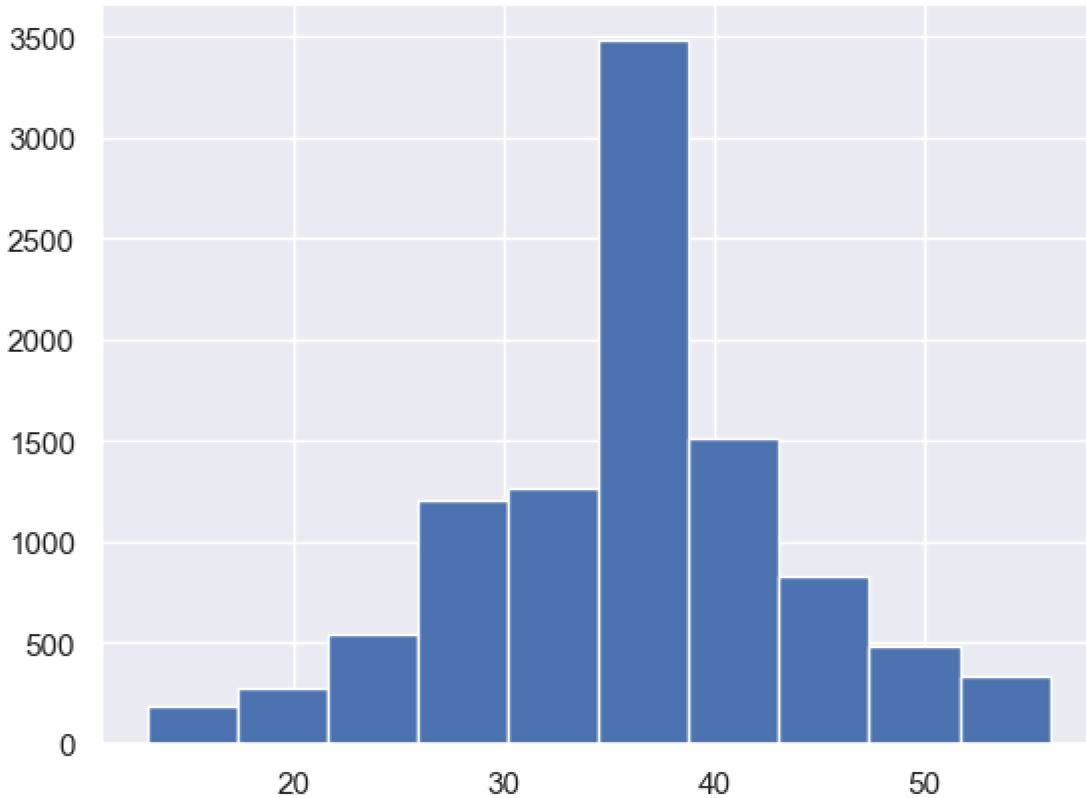


In [174…

```
plt.hist(dataset["Months_on_book"])
```

Out[174]:

```
(array([ 188.,  278.,  546., 1208., 1265., 3485., 1515.,  825.,  479.,
         338.]),
 array([13. , 17.3, 21.6, 25.9, 30.2, 34.5, 38.8, 43.1, 47.4, 51.7, 56. ]),
 <BarContainer object of 10 artists>)
```

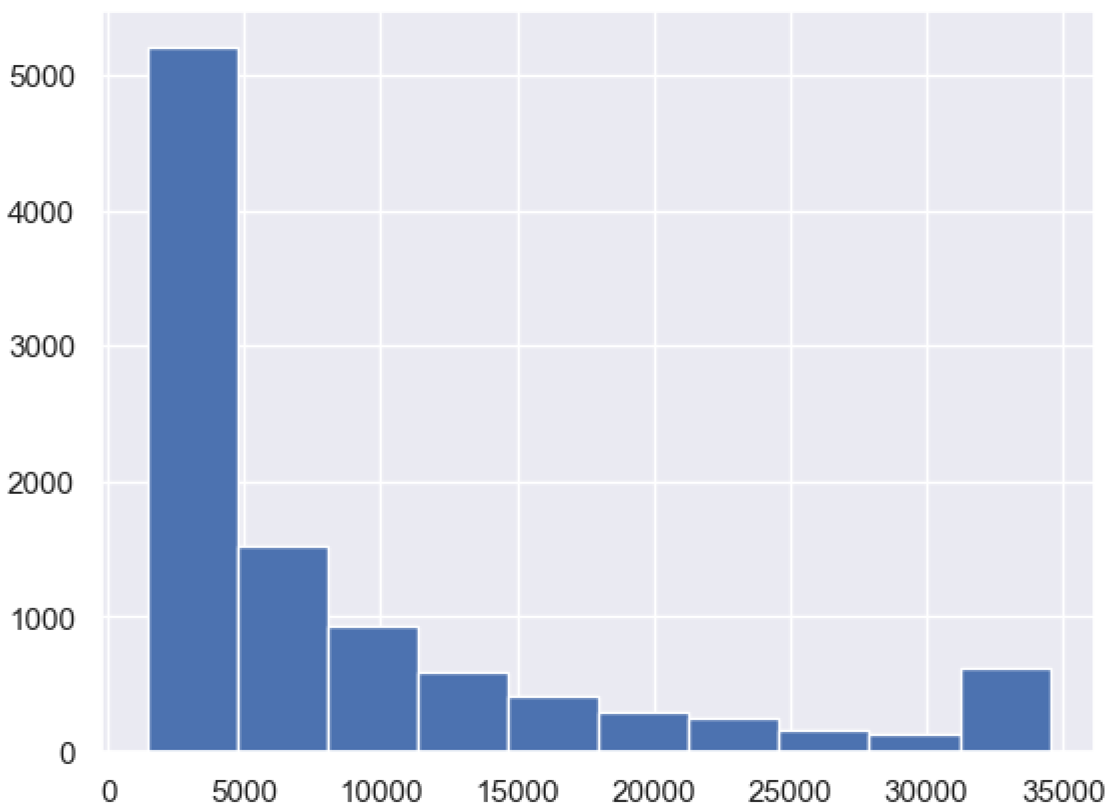


In [179…

```
plt.hist(dataset['Credit_Limit']) # right/positive skewed distribution
```

Out[179]:

```
(array([5211., 1524.,  931.,  589.,  407.,  299.,  252.,  161.,  128.,
         625.]),
 array([ 1438.3 ,  4746.07,  8053.84, 11361.61, 14669.38, 17977.15,
        21284.92, 24592.69, 27900.46, 31208.23, 34516.  ]),
 <BarContainer object of 10 artists>)
```

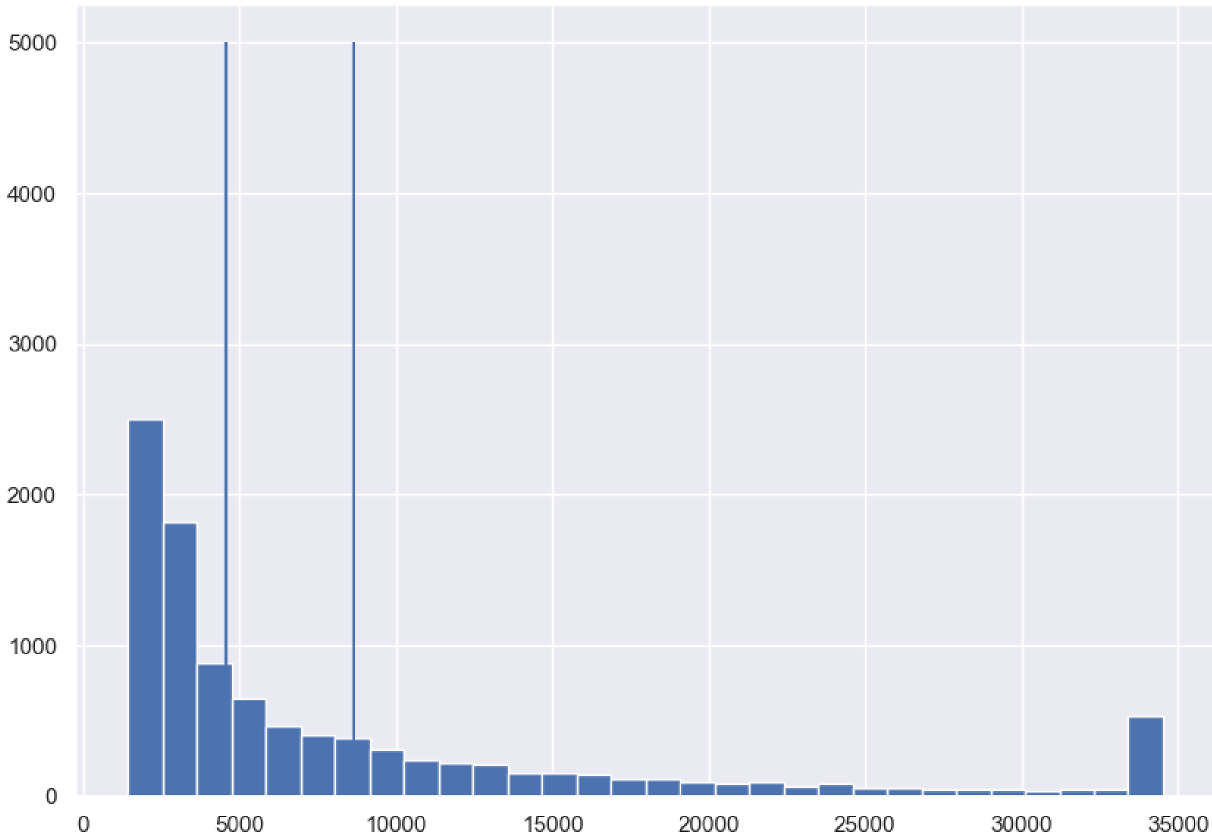


In [184…

```
plt.figure(figsize = (10,7)) # increasing figure size
plt.hist(dataset["Credit_Limit"], bins = 30) # histogram of the variable/feature "Cred
plt.vlines(dataset['Credit_Limit'].mean(), 0, 5000) # Adding mean on the histogram dis

plt.vlines(dataset['Credit_Limit'].median(), 0, 5000) # Adding median on the histogram
```

Out[184]:

```
<matplotlib.collections.LineCollection at 0x1fabc288b50>
```

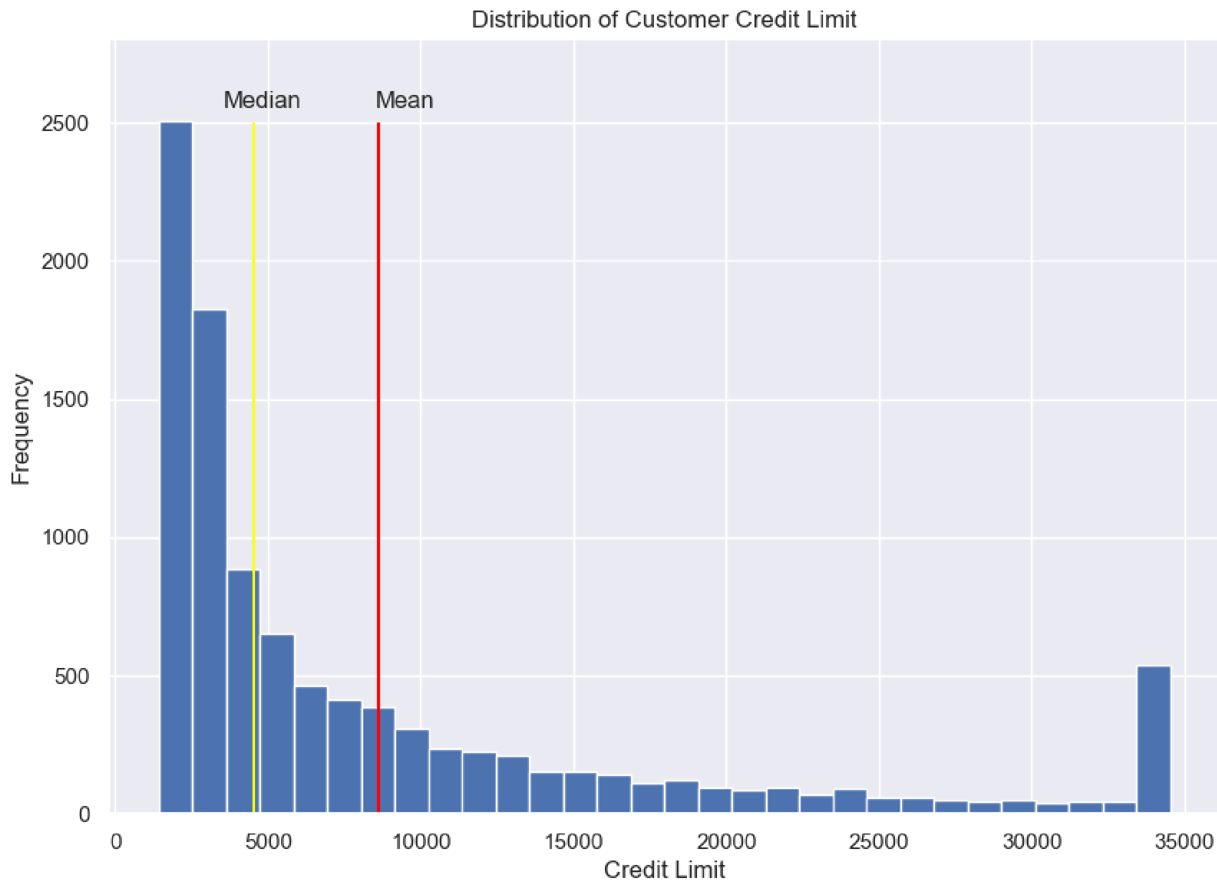


In [188…

```
plt.figure(figsize = (10,7)) # increasing figure size
plt.hist(dataset["Credit_Limit"], bins = 30) # histogram of the variable/feature "Cred
plt.vlines(dataset['Credit_Limit'].mean(), 0, 2500, colors = 'orange') # Adding mean o
plt.vlines(dataset['Credit_Limit'].median(), 0, 2500, colors = 'yellow') # Adding medi
plt.text(dataset['Credit_Limit'].mean()-100, 2500+50, 'Mean')
plt.text(dataset['Credit_Limit'].median()-1000, 2500+50, "Median")
plt.ylim(0,2800)
plt.title('Distribution of Customer Credit Limit')
plt.ylabel('Frequency')
plt.xlabel('Credit Limit')
```

Out[188]:

```
Text(0.5, 0, 'Credit Limit')
```

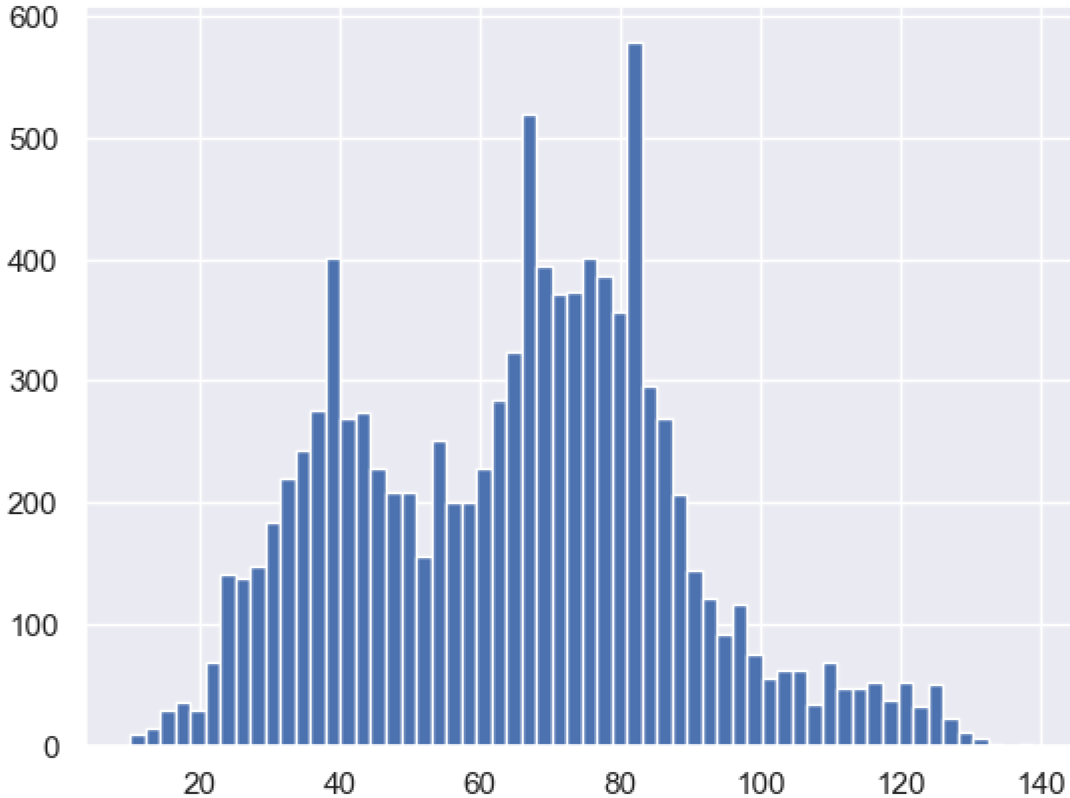


In [192…

```python
plt.hist(dataset['Total_Trans_Ct'], bins = 60)
```

Out[192]:

```
(array([ 10.,  14.,  29.,  36.,  30.,  68., 141., 138., 148., 184., 220.,
        243., 276., 401., 270., 274., 229., 208., 209., 156., 252., 200.,
        200., 229., 284., 324., 520., 395., 371., 373., 401., 387., 357.,
        579., 295., 270., 207., 145., 121.,  91., 117.,  76.,  55.,  62.,
         63.,  35.,  69.,  47.,  48.,  53.,  38.,  53.,  33.,  50.,  22.,
         11.,   7.,   1.,   0.,   2.]),
 array([ 10.  ,  12.15,  14.3 ,  16.45,  18.6 ,  20.75,  22.9 ,  25.05,
         27.2 ,  29.35,  31.5 ,  33.65,  35.8 ,  37.95,  40.1 ,  42.25,
         44.4 ,  46.55,  48.7 ,  50.85,  53.  ,  55.15,  57.3 ,  59.45,
         61.6 ,  63.75,  65.9 ,  68.05,  70.2 ,  72.35,  74.5 ,  76.65,
         78.8 ,  80.95,  83.1 ,  85.25,  87.4 ,  89.55,  91.7 ,  93.85,
         96.  ,  98.15, 100.3 , 102.45, 104.6 , 106.75, 108.9 , 111.05,
        113.2 , 115.35, 117.5 , 119.65, 121.8 , 123.95, 126.1 , 128.25,
        130.4 , 132.55, 134.7 , 136.85, 139.  ]),
 <BarContainer object of 60 artists>)
```



Data Transformation

In [194…

```
# Log transformation helps make data less skewed
# normalization is like a min-max scaler
def normalize(column):
    upper = column.max()
    lower = column.min()
    y = (column - lower)/(upper - lower)
    return y
normalize(dataset['Credit_Limit'])
```

Out[194]:

```
0        0.340190
1        0.206112
2        0.059850
3        0.056676
4        0.099091
           ...
10122    0.077536
10123    0.085819
10124    0.120042
10125    0.116172
10126    0.270566
Name: Credit_Limit, Length: 10127, dtype: float64
```

In [198…

```
dataset['Credit_Limit_Normalized'] = normalize(dataset['Credit_Limit'])
dataset['Credit_Limit_Log_Transformed'] = np.log(dataset['Credit_Limit'])

fig, axes = plt.subplots(2, 2, figsize = (15,10))
fig.suptitle('Before and After Transformation')
#AxesSubplot:xlabel='Credit_Limit_Log_Transformed', ylabel='Count'

#create boxplot in each subplot
```

```
sns.histplot(dataset, x = 'Credit_Limit', ax=axes[0,0])
sns.histplot(dataset, x = 'Credit_Limit_Normalized', ax=axes[0,1])
sns.histplot(dataset, x = 'Credit_Limit', ax=axes[1,0])
sns.histplot(dataset, x = 'Credit_Limit_Log_Transformed', ax=axes[1,1])
```

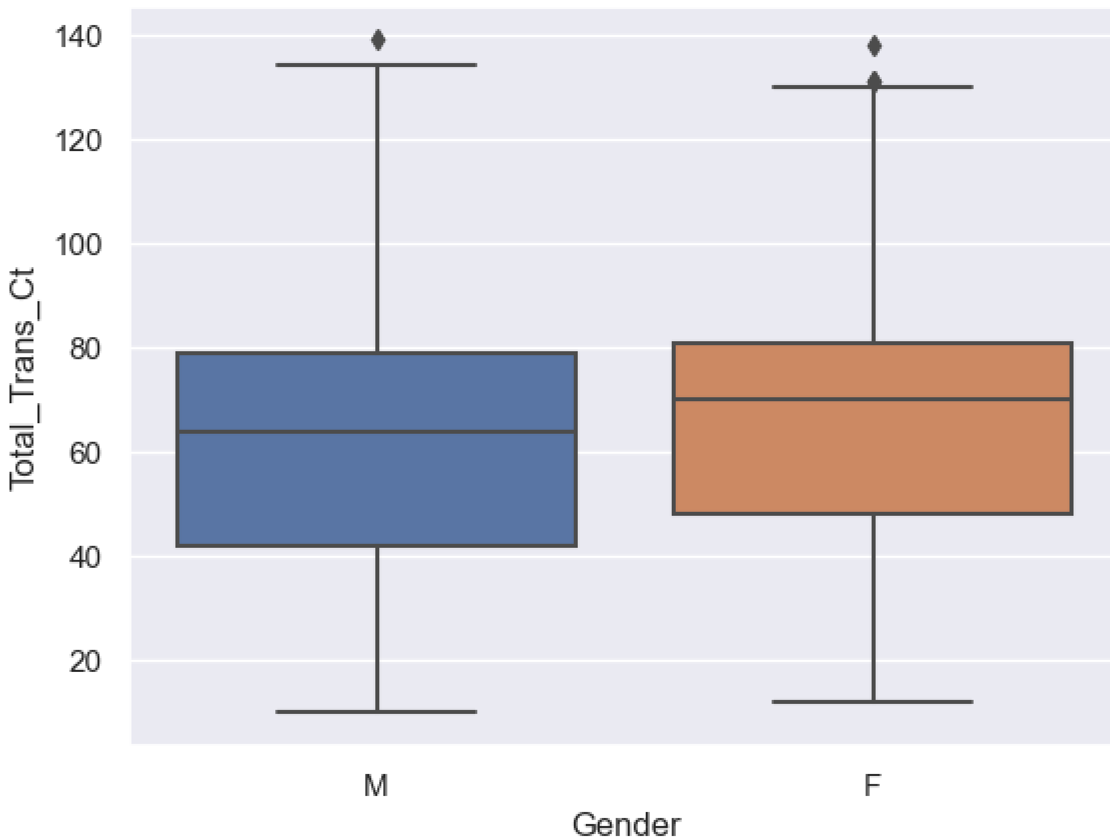
```
  File "C:\Users\user\AppData\Local\Temp\ipykernel_3212\2188215937.py", line 6
    AxesSubplot:xlabel='Credit_Limit_Log_Transformed', ylabel='Count'
                                                                  ^
SyntaxError: invalid syntax
```

In [199…

```
#Box and Wisker Plot
# to visualize median, quartiles and any outliers
sns.boxplot(x = dataset['Gender'], y = dataset['Total_Trans_Ct'])
```

Out[199]:

```
<AxesSubplot:xlabel='Gender', ylabel='Total_Trans_Ct'>
```



In [203…

```
#### Visualize gender and age gap
pyramid = dataset.groupby(['Gender', 'Customer_Age_range'])['CLIENTNUM'].nunique().res
pyramid
```

Out[203]:

	Gender	Customer_Age_range	CLIENTNUM
0	F	20s	93
1	F	30s	956
2	F	40s	2410
3	F	50s	1619
4	F	60s	280
5	F	70s	0
6	M	20s	102
7	M	30s	885
8	M	40s	2151
9	M	50s	1379
10	M	60s	250
11	M	70s	2

In [208…

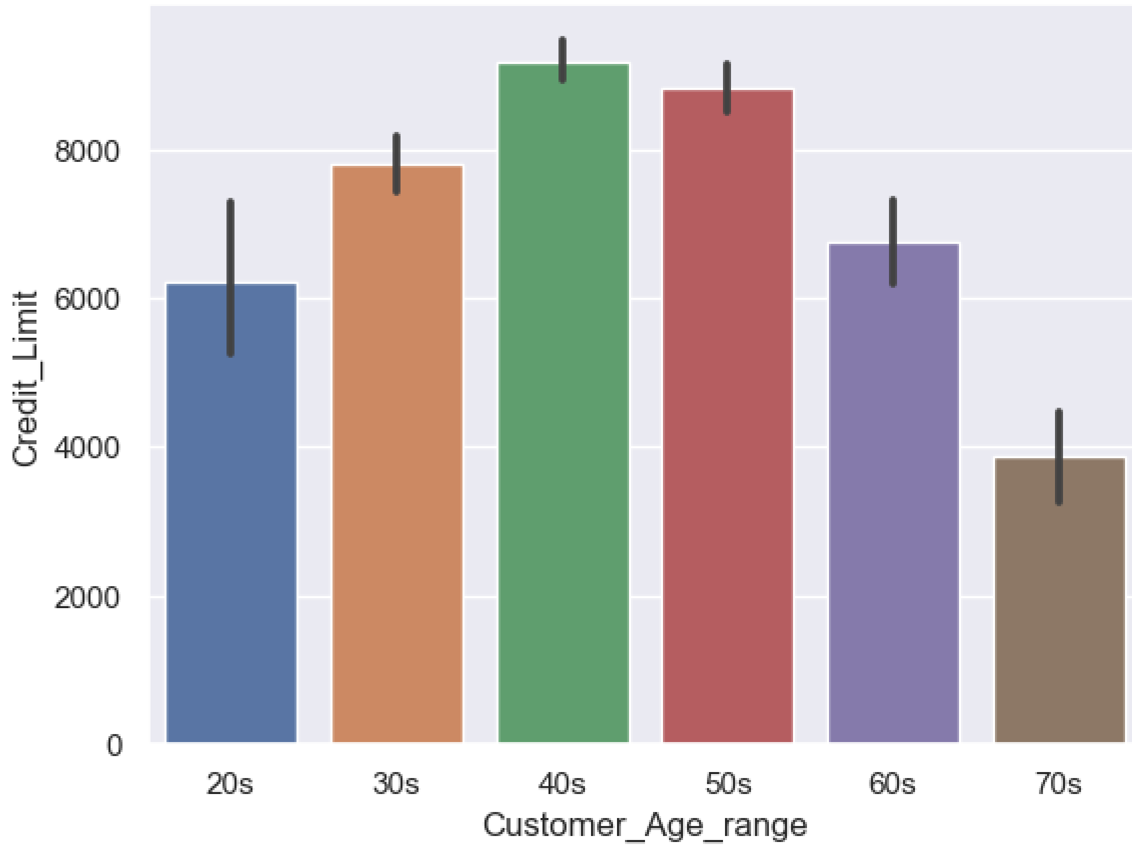
```
men_bins = np.array(pyramid[pyramid['Gender'] == 'M']['CLIENTNUM'])
women_bins = np.array(-1*pyramid[pyramid['Gender'] == 'F']['CLIENTNUM'])
y = list(range(25, 100, 10))

layout = go.Layout(yaxis=go.layout.YAxis(title='Age'),
                   xaxis=go.layout.XAxis(
                       range=[-1200, 1200],
                       tickvals=[-1000, -700, -300, 0, 300, 700, 1000],
                       ticktext=[1000, 700, 300, 0, 300, 700, 1000],
                       title='Number'),
                   barmode='overlay',
                   bargap=0.1)

pyramid_data = [go.Bar(y=y,
               x=men_bins,
               orientation='h',
               name='Men',
               hoverinfo='x',
               marker=dict(color='powderblue')
               ),
        go.Bar(y=y,
               x=women_bins,
               orientation='h',
               name='Women',
               text=-1 * women_bins.astype('int'),
               hoverinfo='text',
               marker=dict(color='seagreen')
               )]

iplot(dict(data=pyramid_data, layout=layout), filename='EXAMPLES/bar_pyramid')
```

In [222…

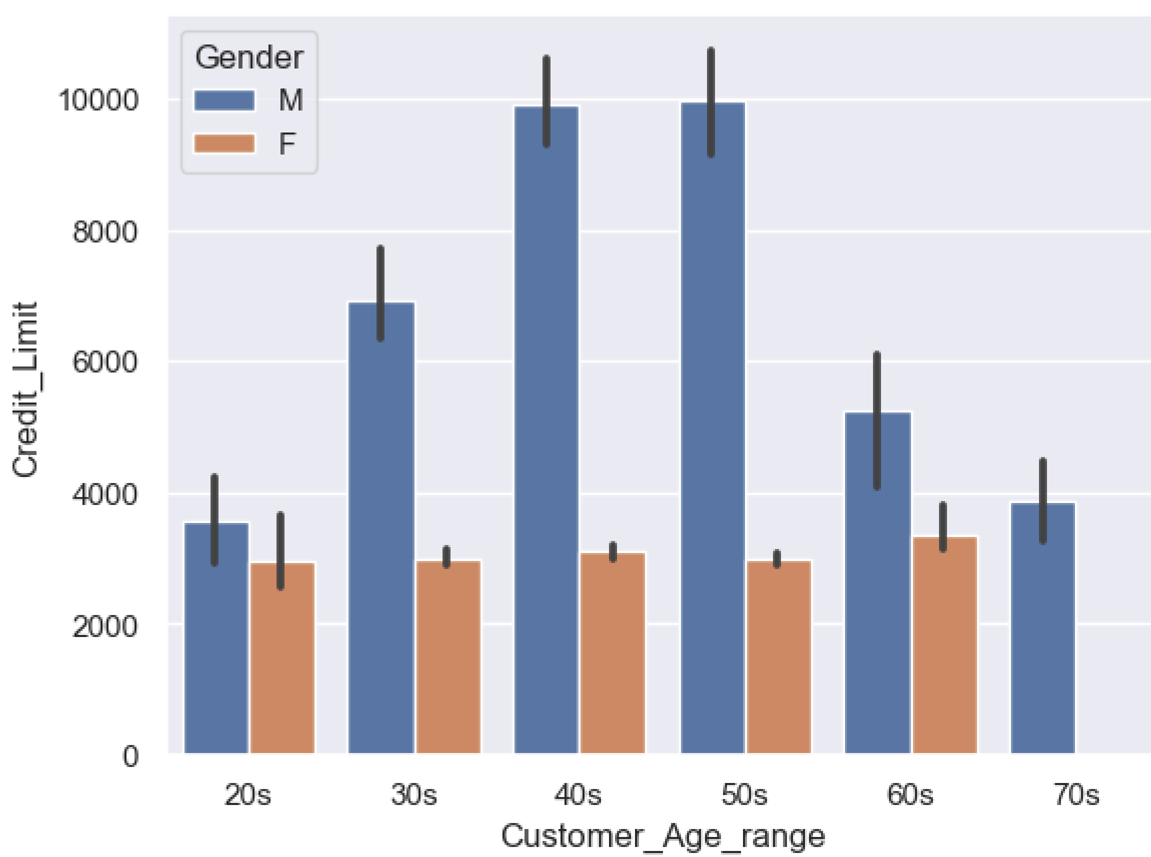
```
#### Comparing Categories

sns.barplot(x='Customer_Age_range', y ='Credit_Limit', data = dataset, estimator=np.me
```

Out[222]:

```
<AxesSubplot:xlabel='Customer_Age_range', ylabel='Credit_Limit'>
```



In [223…
```
# adding gender
sns.barplot(x='Customer_Age_range', y ='Credit_Limit', hue = 'Gender', data = dataset,
```
Out[223]:
```
<AxesSubplot:xlabel='Customer_Age_range', ylabel='Credit_Limit'>
```



In [227…

```
barplot = dataset.groupby(['Customer_Age_range', 'Gender'])['Credit_Limit'].mean().res
barplot
```

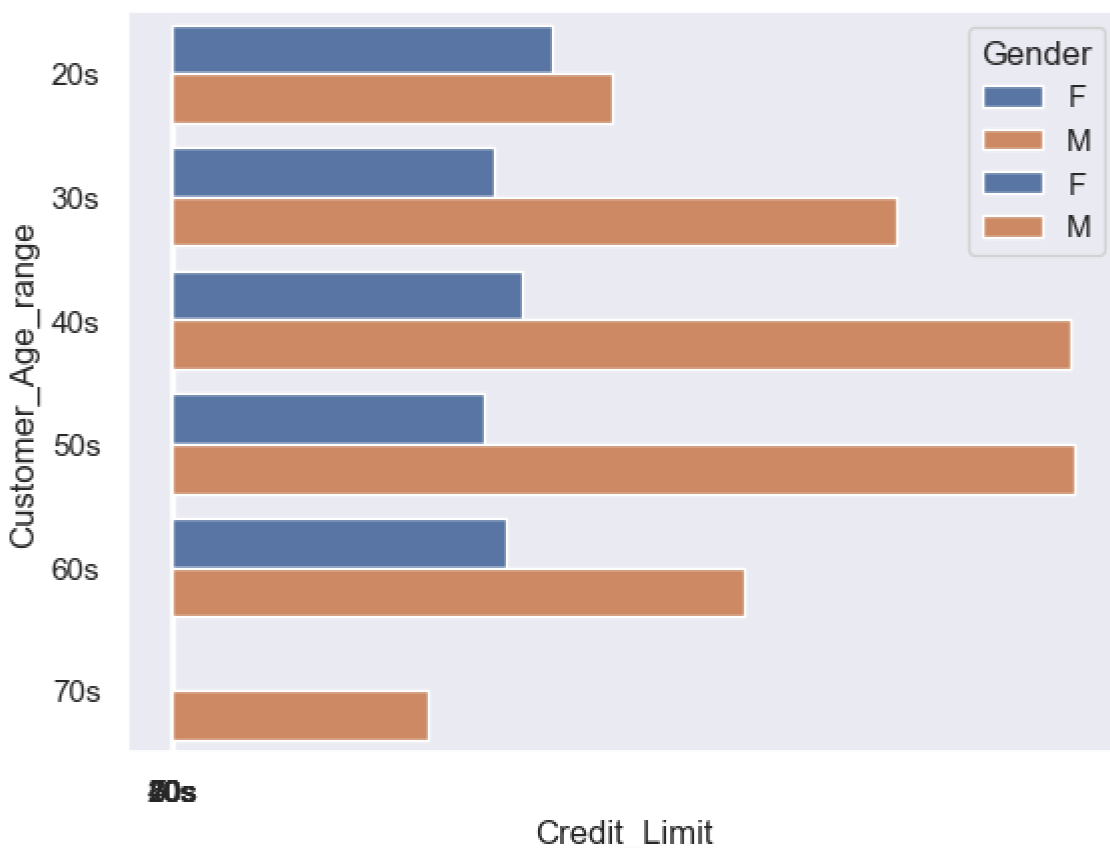
Out[227]:

	Customer_Age_range	Gender	Credit_Limit
0	20s	F	5731.101075
1	20s	M	6649.367647
2	30s	F	4867.775314
3	30s	M	10948.605311
4	40s	F	5270.821784
5	40s	M	13557.484844
6	50s	F	4702.649475
7	50s	M	13635.717041
8	60s	F	5053.412857
9	60s	M	8626.832000
10	70s	F	NaN
11	70s	M	3860.500000

In [231…

```
sns.barplot(x = 'Customer_Age_range', y = 'Credit_Limit', hue = 'Gender', data = barpl
#sns.barplot(x = 'Credit_Limit', y = 'Customer_Age_range', hue = 'Gender', data = barp
```

Out[231]: <AxesSubplot:xlabel='Credit_Limit', ylabel='Customer_Age_range'>



In [236…

```
### Lolipop chart
lolipop = dataset.groupby(['Customer_Age_range'])['Credit_Limit'].mean().reset_index()
lolipop
```

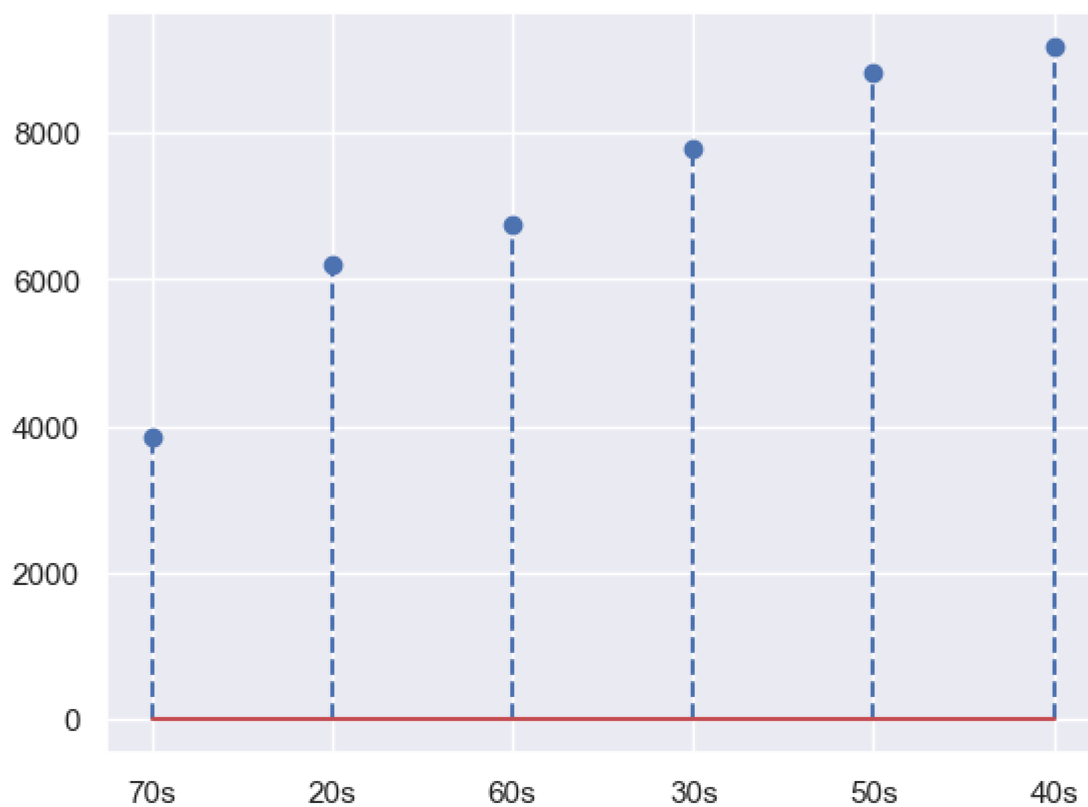
Out[236]:

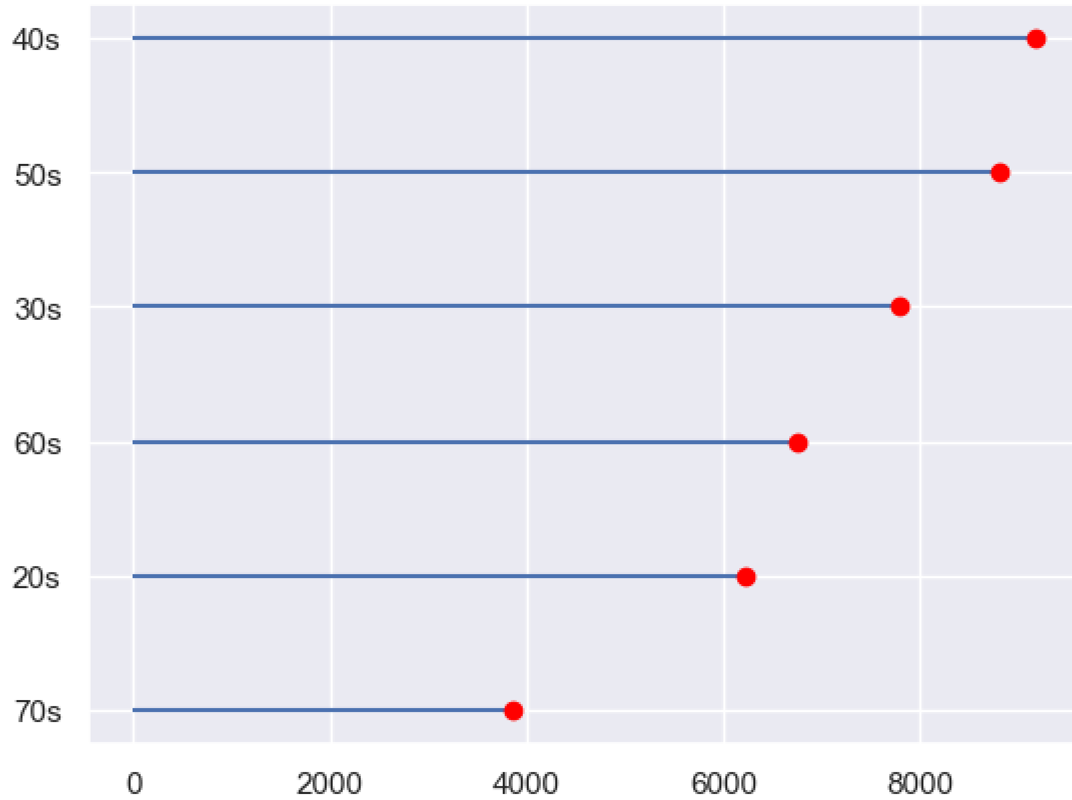
	Customer_Age_range	Credit_Limit
5	70s	3860.500000
0	20s	6211.425128
4	60s	6738.987925
1	30s	7790.933677
3	50s	8811.622181
2	40s	9178.870949

In [238…

```
plt.stem(lolipop['Customer_Age_range'], lolipop['Credit_Limit'], linefmt = '--')
#fig, ax = plt.subplots()
#ax.hlines(lolipop['Customer_Age_range'], xmin = 0, xmax = lolipop['Credit_Limit'])
#ax.plot(lolipop['Credit_Limit'], lolipop['Customer_Age_range'], 'o', color= 'red')
```

Out[238]: [<matplotlib.lines.Line2D at 0x1fac4b01bb0>]





Data Visualization

In [ ]:
```
#dataset.columns
#dataset._get_numeric_data()
dataset._get_numeric_data().columns
```

In [244…
```
## Why customers are leaving the bank or the credit card service.
#Looking at customer behavior with aggregated table
dataset.groupby(['Attrition_Flag']).agg({'CLIENTNUM':'nunique',
                                        'Customer_Age':'mean',
                                         #'Gender':'mean',
                                         'Dependent_count':'mean',
                                         #'Education_Level':'mean',
                                         #'Marital_Status':'mean',
                                         #'Income_Category':'mean',
                                         #'Card_Category':'mean',
                                         'Months_on_book':'mean',
                                         'Total_Relationship_Count':'mean',
                                         'Months_Inactive_12_mon':'mean',
                                         'Contacts_Count_12_mon':'mean',
                                         'Credit_Limit':'mean',
                                         'Total_Revolving_Bal':'mean',
                                         'Avg_Open_To_Buy':'mean',
                                         'Total_Amt_Chng_Q4_Q1':'mean',
                                         'Total_Trans_Amt':'mean',
                                         'Total_Trans_Ct':'mean',
                                         'Total_Ct_Chng_Q4_Q1':'mean',
                                         'Avg_Utilization_Ratio':'mean'})
```

Out[244]:

Attrition_Flag	CLIENTNUM	Customer_Age	Dependent_count	Months_on_book	Total_Relationship_Cou
Attrited Customer	1627	46.659496	2.402581	36.178242	3.2796
Existing Customer	8500	46.262118	2.335412	35.880588	3.9145

In [246...

```
pivot_table = dataset.groupby(['Attrition_Flag']).agg({'CLIENTNUM':'nunique',
                                        'Customer_Age':'mean',
                                         #'Gender':'mean',
                                         'Dependent_count':'mean',
                                         #'Education_Level':'mean',
                                         #'Marital_Status':'mean',
                                         #'Income_Category':'mean',
                                         #'Card_Category':'mean',
                                         'Months_on_book':'mean',
                                         'Total_Relationship_Count':'mean',
                                         'Months_Inactive_12_mon':'mean',
                                         'Contacts_Count_12_mon':'mean',
                                         'Credit_Limit':'mean',
                                         'Total_Revolving_Bal':'mean',
                                         'Avg_Open_To_Buy':'mean',
                                         'Total_Amt_Chng_Q4_Q1':'mean',
                                         'Total_Trans_Amt':'mean',
                                         'Total_Trans_Ct':'mean',
                                         'Total_Ct_Chng_Q4_Q1':'mean',
                                         'Avg_Utilization_Ratio':'mean'}).T
pivot_table
```

Out[246]:

Attrition_Flag	Attrited Customer	Existing Customer
CLIENTNUM	1627.000000	8500.000000
Customer_Age	46.659496	46.262118
Dependent_count	2.402581	2.335412
Months_on_book	36.178242	35.880588
Total_Relationship_Count	3.279656	3.914588
Months_Inactive_12_mon	2.693301	2.273765
Contacts_Count_12_mon	2.972342	2.356353
Credit_Limit	8136.039459	8726.877518
Total_Revolving_Bal	672.822987	1256.604118
Avg_Open_To_Buy	7463.216472	7470.273400
Total_Amt_Chng_Q4_Q1	0.694277	0.772510
Total_Trans_Amt	3095.025814	4654.655882
Total_Trans_Ct	44.933620	68.672588
Total_Ct_Chng_Q4_Q1	0.554386	0.742434
Avg_Utilization_Ratio	0.162475	0.296412

In [251…

```
## Let's see the difference between attrited customer and existing customer
pivot_table['difference'] = pivot_table['Attrited Customer']/pivot_table['Existing Cus
pivot_table.sort_values('difference')

#customers leaving have the following things in common...below existing customers
#Total_Revolving_Bal     672.822987      1256.604118     -0.464570
#Avg_Utilization_Ratio  0.162475        0.296412        -0.451860
#Total_Trans_Ct 44.933620       68.672588       -0.345683
```

Out[251]:

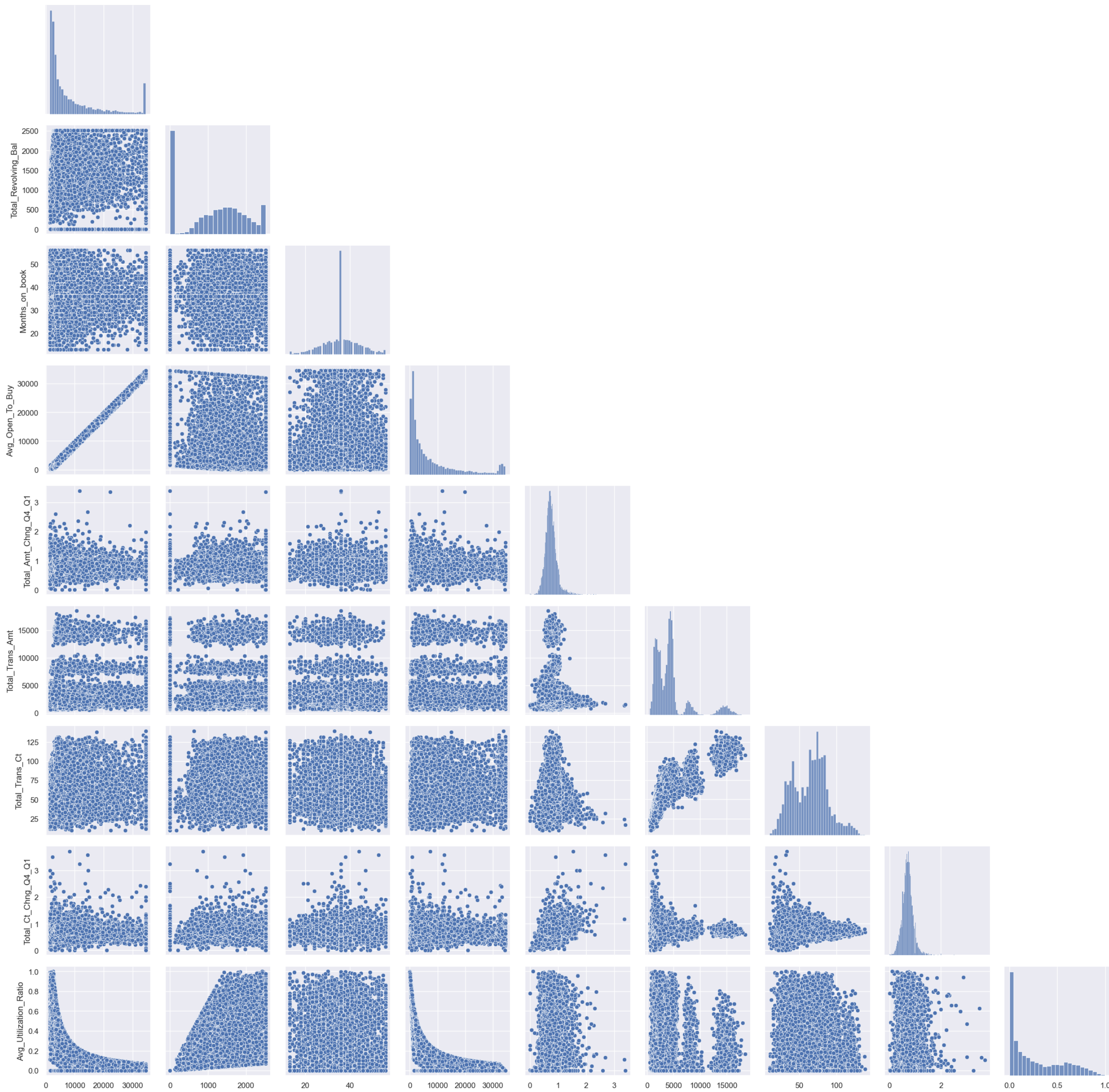
Attrition_Flag	Attrited Customer	Existing Customer	difference
CLIENTNUM	1627.000000	8500.000000	-0.808588
Total_Revolving_Bal	672.822987	1256.604118	-0.464570
Avg_Utilization_Ratio	0.162475	0.296412	-0.451860
Total_Trans_Ct	44.933620	68.672588	-0.345683
Total_Trans_Amt	3095.025814	4654.655882	-0.335069
Total_Ct_Chng_Q4_Q1	0.554386	0.742434	-0.253286
Total_Relationship_Count	3.279656	3.914588	-0.162196
Total_Amt_Chng_Q4_Q1	0.694277	0.772510	-0.101271
Credit_Limit	8136.039459	8726.877518	-0.067703
Avg_Open_To_Buy	7463.216472	7470.273400	-0.000945
Months_on_book	36.178242	35.880588	0.008296
Customer_Age	46.659496	46.262118	0.008590
Dependent_count	2.402581	2.335412	0.028761
Months_Inactive_12_mon	2.693301	2.273765	0.184512
Contacts_Count_12_mon	2.972342	2.356353	0.261416

In [255…

```
### Looking at relationship xy
dataset2 = dataset._get_numeric_data()
dataset2 = dataset[[ 'Credit_Limit', 'Total_Revolving_Bal','Months_on_book',
       'Avg_Open_To_Buy', 'Total_Amt_Chng_Q4_Q1', 'Total_Trans_Amt',
       'Total_Trans_Ct', 'Total_Ct_Chng_Q4_Q1', 'Avg_Utilization_Ratio','Attrition_Fla
g = sns.PairGrid(dataset2, diag_sharey=False, corner=True)
g.map_lower(sns.scatterplot)
g.map_diag(sns.histplot)
```

Out[255]:

```
<seaborn.axisgrid.PairGrid at 0x1fac0fa6640>
```

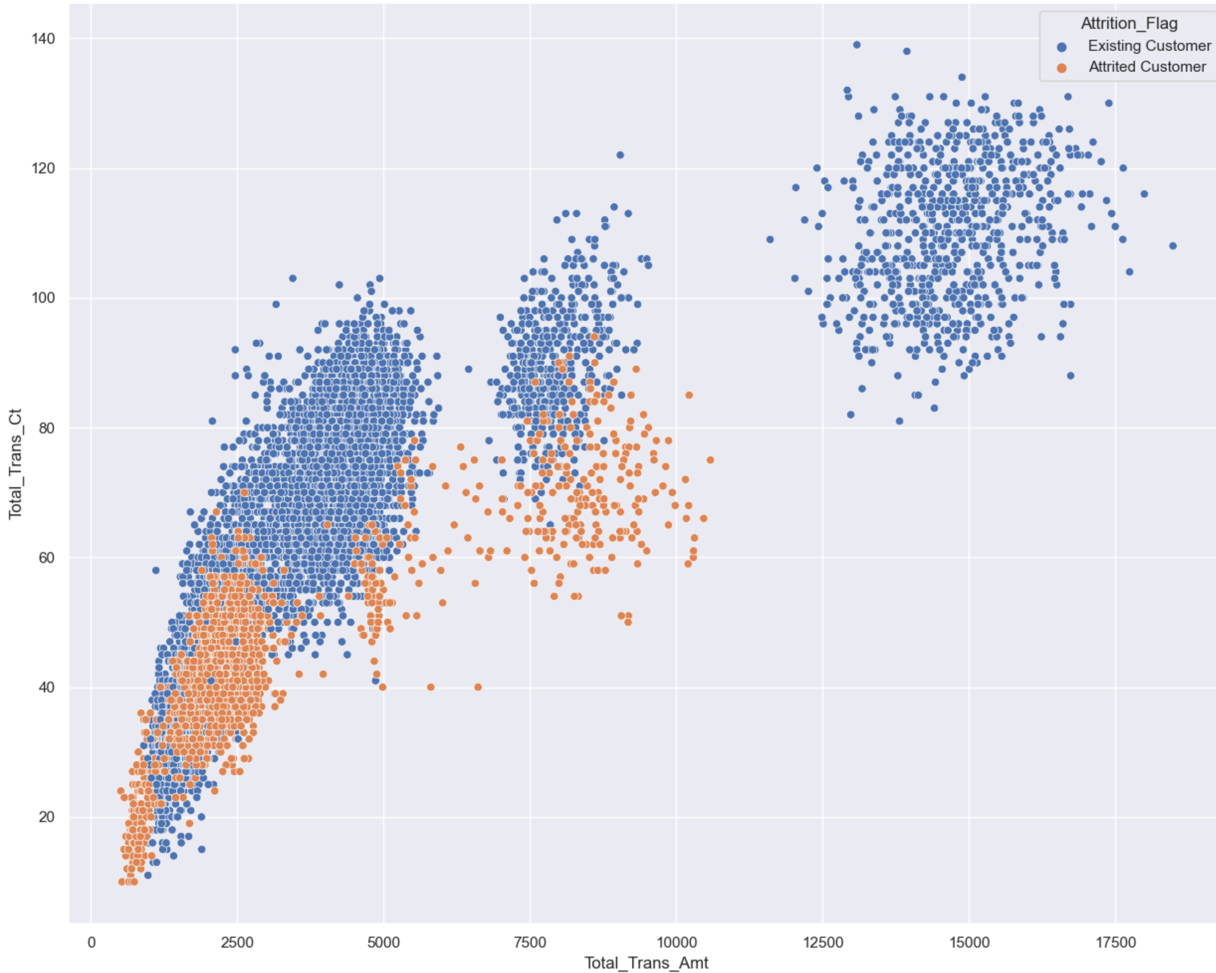


In [263…

```
plt.figure(figsize = (15,12))
#sns.scatterplot(x='Credit_Limit', y = 'Avg_Open_To_Buy', data = dataset)
sns.scatterplot(x='Total_Trans_Amt', y = 'Total_Trans_Ct', hue = 'Attrition_Flag', dat
# Based on the plots, no attrited customer (the ones that left) has spent more above $
```

Out[263]:

```
<AxesSubplot:xlabel='Total_Trans_Amt', ylabel='Total_Trans_Ct'>
```

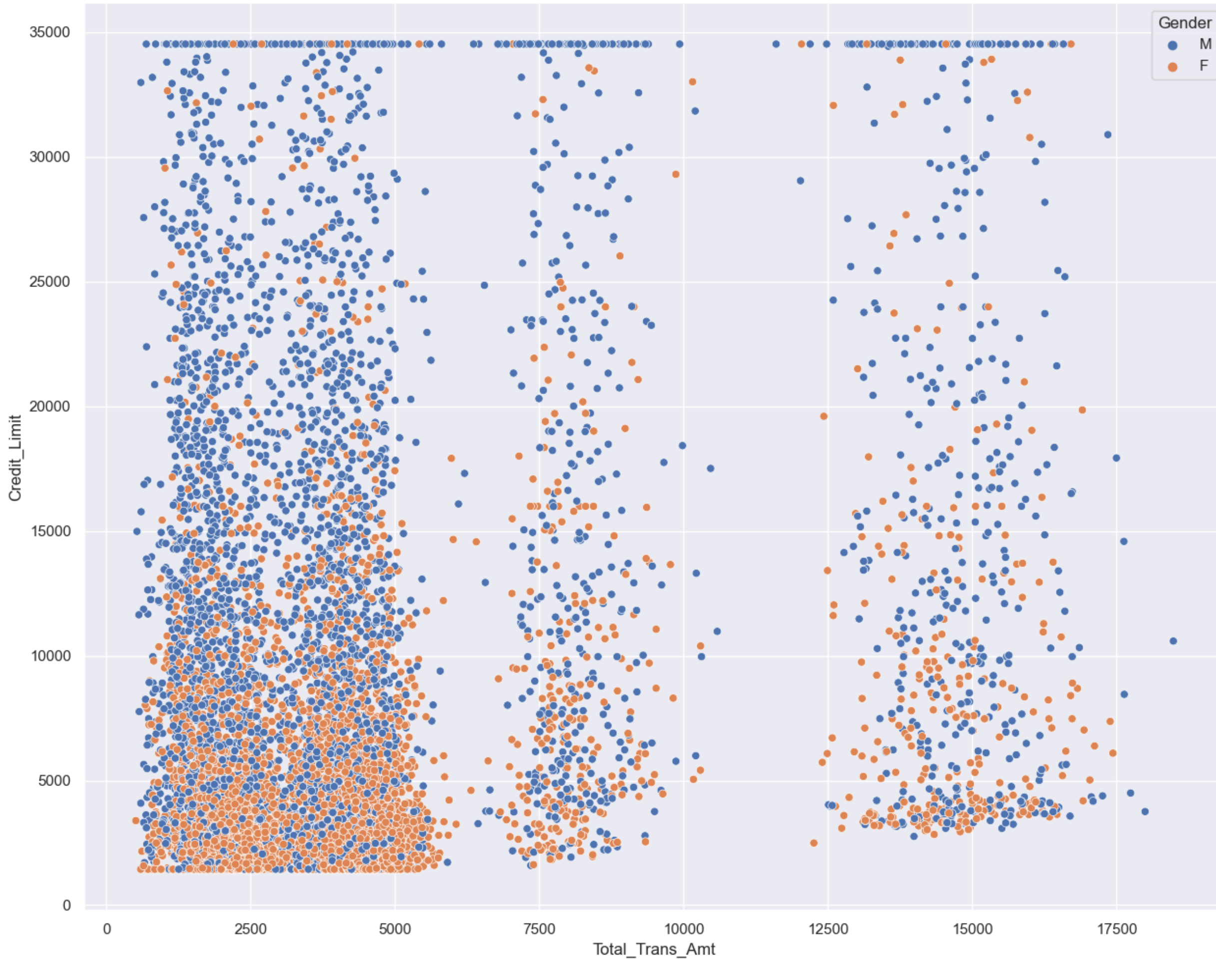


In [271…

```
plt.figure(figsize = (15,12))
#sns.scatterplot(x='Total_Trans_Amt', y = 'Credit_Limit', hue = 'Attrition_Flag', data
sns.scatterplot(x='Total_Trans_Amt', y = 'Credit_Limit', hue = 'Gender', data = datase

#dataset['Credit_Limit'].min() # max = 34516.0$, min = 1438.3$
#dataset['Total_Trans_Amt'].max() # min = 510 , max = 18484$
# It looks like some customers with highest credit limit (close to $35k) still spent l
# we could compare the number of existing customers with spending greater than $6k and
# attrited customer with the same range.

# scatter plot is telling us there about the same (if not more than) existing customer
# under $11k total transaction amount spent and credit limit no more than $35k
# Thus, we have a reason to believe that credit limit is not a factor or influence cus
# it would be interesting to zoom in (total_trans_Amt < 3000 and credit limit <15000)
# from this groupB...this groupB seems to have the most attrited customer..and attrite
# Are there additional factors to explain this groupB, could be: education, customer c
# How do you see attrition flag and gender on the same scatter plot x = total_trans_am
# count...we want to look at the attrited customer to see if gender has impact...could
# sns.barplot(x='Customer_Age_range', y ='Credit_Limit', hue = 'Gender', data = datase
# or
# create a new feature/column called attrition_gender (AM = attited male, AF = attrite
```

Out[271]:

```
<AxesSubplot:xlabel='Total_Trans_Amt', ylabel='Credit_Limit'>
```



In [275…

```
## Ridge Plot
# let's dig into variable showing differences between Churned and Existing customer
bins = [ 0, 11000, 900000]
labels = ["Group A", "Group B"]
dataset["Total_Trans_Amt_bin"] = pd.cut(dataset['Total_Trans_Amt'], bins = bins, label
                                        include_lowest = True, right = False)
dataset.groupby(['Total_Trans_Amt_bin' ,'Attrition_Flag']).agg({'CLIENTNUM':'nunique',
                                        'Customer_Age':'mean',
                                         #'Gender':'mean',
                                         'Dependent_count':'mean',
                                         #'Education_Level':'mean',
                                         #'Marital_Status':'mean',
                                         #'Income_Category':'mean',
                                         #'Card_Category':'mean',
                                         'Months_on_book':'mean',
                                         'Total_Relationship_Count':'mean',
                                         'Months_Inactive_12_mon':'mean',
                                         'Contacts_Count_12_mon':'mean',
                                         'Credit_Limit':'mean',
                                         'Total_Revolving_Bal':'mean',
                                         'Avg_Open_To_Buy':'mean',
                                         'Total_Amt_Chng_Q4_Q1':'mean',
                                         'Total_Trans_Amt':'mean',
                                         'Total_Trans_Ct':'mean',
                                         'Total_Ct_Chng_Q4_Q1':'mean',
                                         'Avg_Utilization_Ratio':'mean'}).T


# Total_Revolving_Bal    672.822987(Attrited Customer)    1245.908165 (Existing Customer
```

Out[275]:

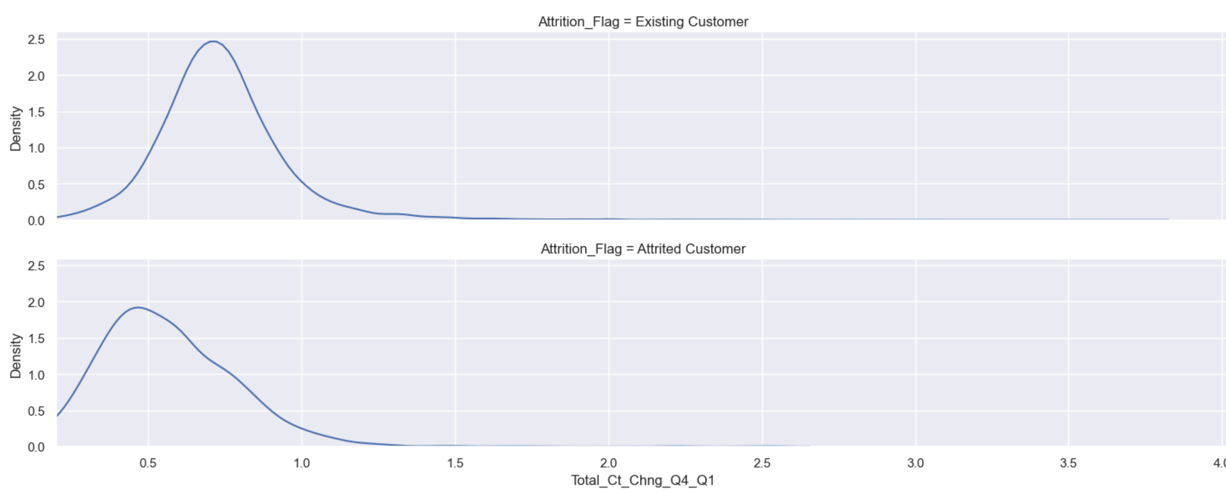
Total_Trans_Amt_bin	Group A		Group B	
Attrition_Flag	Attrited Customer	Existing Customer	Attrited Customer	Existing Customer
CLIENTNUM	1627.000000	7753.000000	0.0	747.000000
Customer_Age	46.659496	46.373920	NaN	45.101740
Dependent_count	2.402581	2.341545	NaN	2.271754
Months_on_book	36.178242	35.964272	NaN	35.012048
Total_Relationship_Count	3.279656	4.064620	NaN	2.357430
Months_Inactive_12_mon	2.693301	2.279376	NaN	2.215529
Contacts_Count_12_mon	2.972342	2.369018	NaN	2.224900
Credit_Limit	8136.039459	8213.629808	NaN	14053.797858
Total_Revolving_Bal	672.822987	1245.908165	NaN	1367.615797
Avg_Open_To_Buy	7463.216472	6967.721643	NaN	12686.182062
Total_Amt_Chng_Q4_Q1	0.694277	0.772248	NaN	0.775229
Total_Trans_Amt	3095.025814	3686.943506	NaN	14698.396252
Total_Trans_Ct	44.933620	64.658326	NaN	110.336011
Total_Ct_Chng_Q4_Q1	0.554386	0.741687	NaN	0.750190
Avg_Utilization_Ratio	0.162475	0.307600	NaN	0.180288

In [279…

```
g = sns.FacetGrid(dataset, row ='Attrition_Flag', aspect=5, height=3)
g.map_dataframe(sns.kdeplot, x = "Total_Ct_Chng_Q4_Q1")
plt.xlim(0.2)
```

Out[279]:

```
(0.2, 4.025070335121794)
```



In [308…

```
#Analytics path to value
#data ..>information...> insight...>data story...>decision...> action...> value
#Changing the background color to white (this is to remover gridlines)
#Remove border
sns.set_theme(style="white")
palette = sns.color_palette("Set2", 12)
```

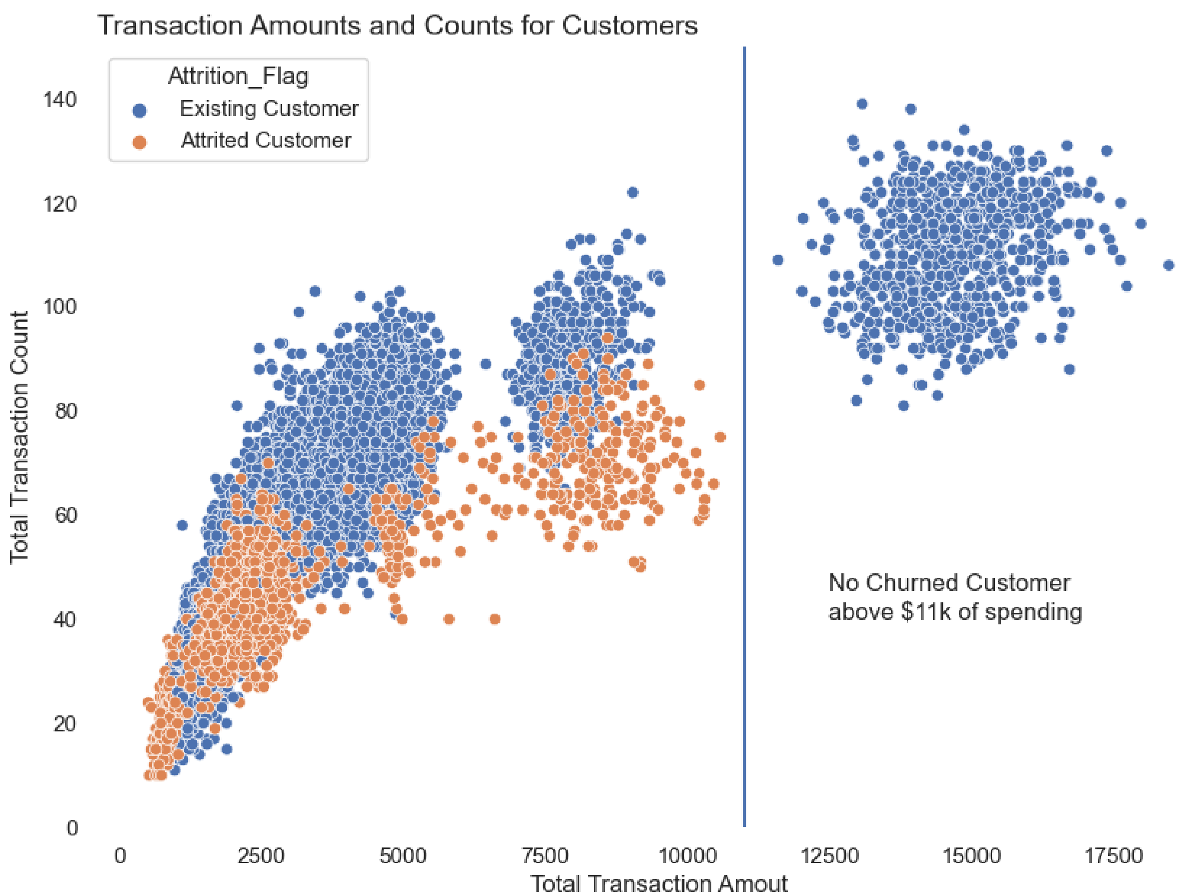
Explanatory Analysis

In [309…

```
#def chart1 ():
#    return
plt.figure(figsize=(10,7))
sns.scatterplot(x='Total_Trans_Amt', y = 'Total_Trans_Ct', hue = 'Attrition_Flag', dat
sns.despine(bottom=True, left=True) # Removes the border
plt.ylim(0,150) #changes the limits of the yaxis
plt.xlabel('Total Transaction Amout') #axis labels
plt.ylabel('Total Transaction Count') # y axis labels
plt.title("Transaction Amounts and Counts for Customers", loc = 'left', size=14)
plt.vlines(11000, 0 , 150) # adding vertical line at the $11k spending
plt.text(12500, 40, "No Churned Customer \nabove $11k of spending")
```

Out[309]:

```
Text(12500, 40, 'No Churned Customer \nabove $11k of spending')
```

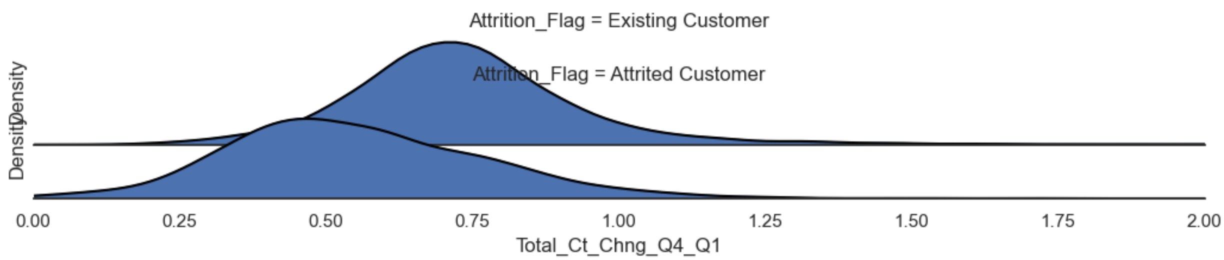


In [303…

```
## Explanatory Analysis
sns.set_theme(style='white', rc={'axes.facecolor': (0, 0, 0 , 0)})
g = sns.FacetGrid(dataset, row = "Attrition_Flag", aspect = 9, height = 1.2)
g.map_dataframe(sns.kdeplot, x="Total_Ct_Chng_Q4_Q1", fill=True, alpha=1)
g.map_dataframe(sns.kdeplot, x='Total_Ct_Chng_Q4_Q1', color='black')
g.fig.subplots_adjust(hspace=-.5)
```

```
g.set(yticks=[])
g.despine(left=True)
plt.xlim(0,2)
```

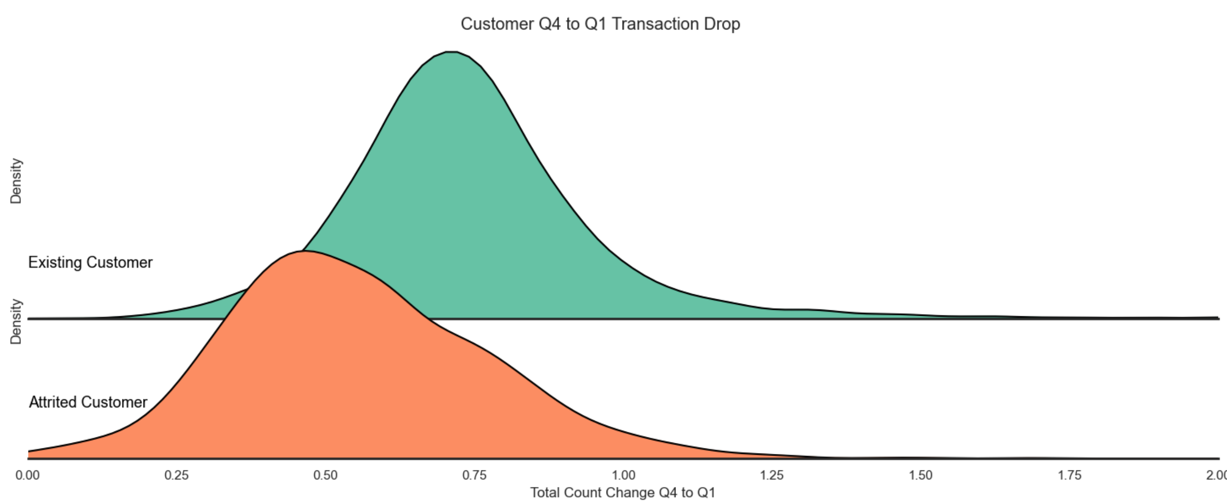
Out[303]: (0.0, 2.0)



In [292…

```
## Explanatory Analysis
sns.set_theme(style='white', rc={'axes.facecolor': (0, 0, 0 , 0), 'axes.linewidth':2})
palette = sns.color_palette("Set2", 12)
g = sns.FacetGrid(dataset, palette = palette, row = "Attrition_Flag", hue="Attrition_F
g.map_dataframe(sns.kdeplot, x="Total_Ct_Chng_Q4_Q1", fill=True, alpha=1)
g.map_dataframe(sns.kdeplot, x='Total_Ct_Chng_Q4_Q1', color='black')
def label(x, color, label):
    ax = plt.gca()
    ax.text(0, .2, label, color ='black', fontsize = 13,
            ha = "left", va='center', transform=ax.transAxes)
g.map(label, "Attrition_Flag")
g.fig.subplots_adjust(hspace=-.5)
g.set_titles("")
g.set(yticks=[], xlabel="Total Count Change Q4 to Q1")
g.despine(left=True)
plt.suptitle('Customer Q4 to Q1 Transaction Drop', y=0.98)
plt.xlim(0,2)
```

Out[292]: (0.0, 2.0)



Analysis

```
+ Total of 10k customers in the portfolio
+ We have experience a 16% churn rate
+ Our average customer has been around for 36 months
+ Average age of our customer is 46 with credit limit of 8600 dollars
+ Customers with highest credit limit (close to 35k dollars) still
```

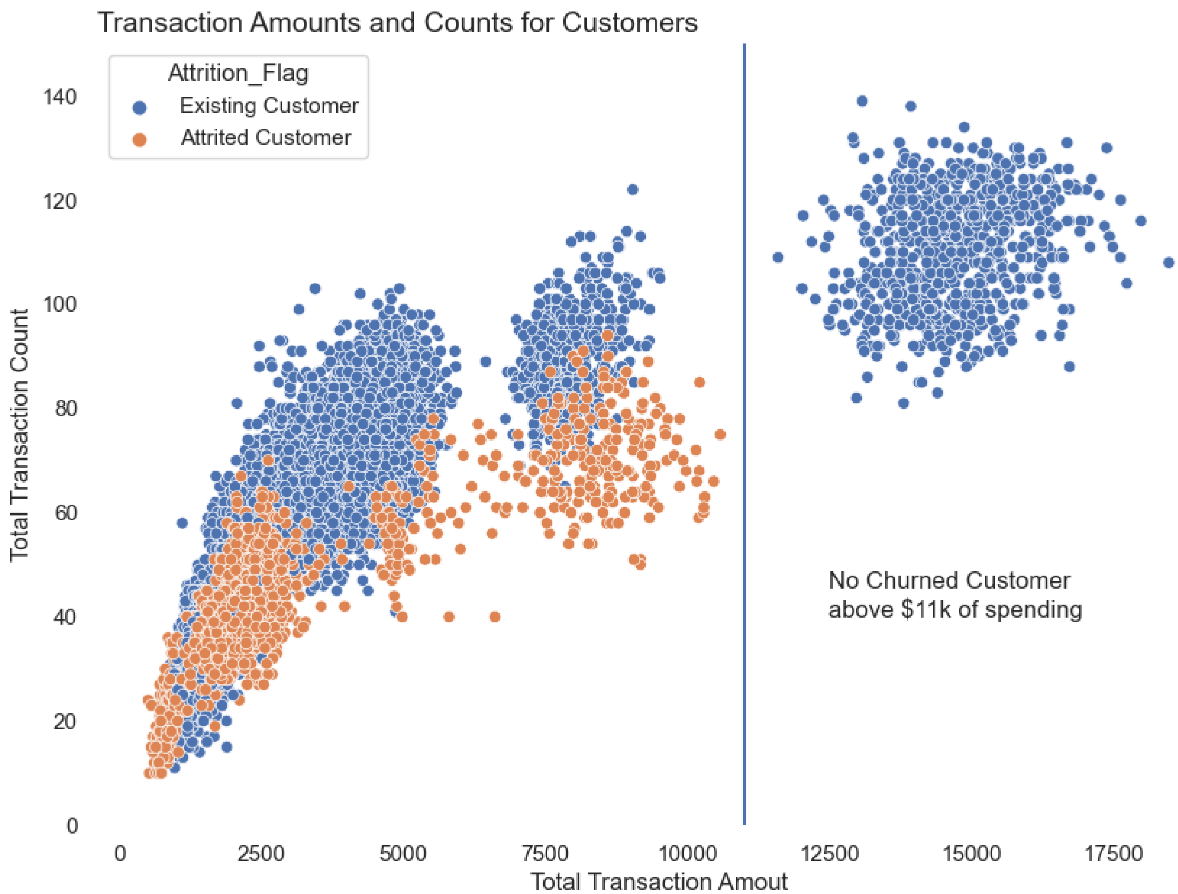
spent less than 10k dollars and close the account
+ There about the same existing than attrited customers under 11k total transaction amount spent and credit limit >35k
+ Credit limit does not influeence customer churn
+ Gender does not influence customer churn

In [310…

```
### No Churned Customers beyond $11k Spending
plt.figure(figsize=(10,7))
sns.scatterplot(x='Total_Trans_Amt', y = 'Total_Trans_Ct', hue = 'Attrition_Flag', dat
sns.despine(bottom=True, left=True) # Removes the border
plt.ylim(0,150) #changes the limits of the yaxis
plt.xlabel('Total Transaction Amout') #axis labels
plt.ylabel('Total Transaction Count') # y axis labels
plt.title("Transaction Amounts and Counts for Customers", loc = 'left', size=14)
plt.vlines(11000, 0 , 150) # adding vertical line at the $11k spending
plt.text(12500, 40, "No Churned Customer \nabove $11k of spending")
```

Out[310]:

```
Text(12500, 40, 'No Churned Customer \nabove $11k of spending')
```



How can we get more customers above the $11k threshold?

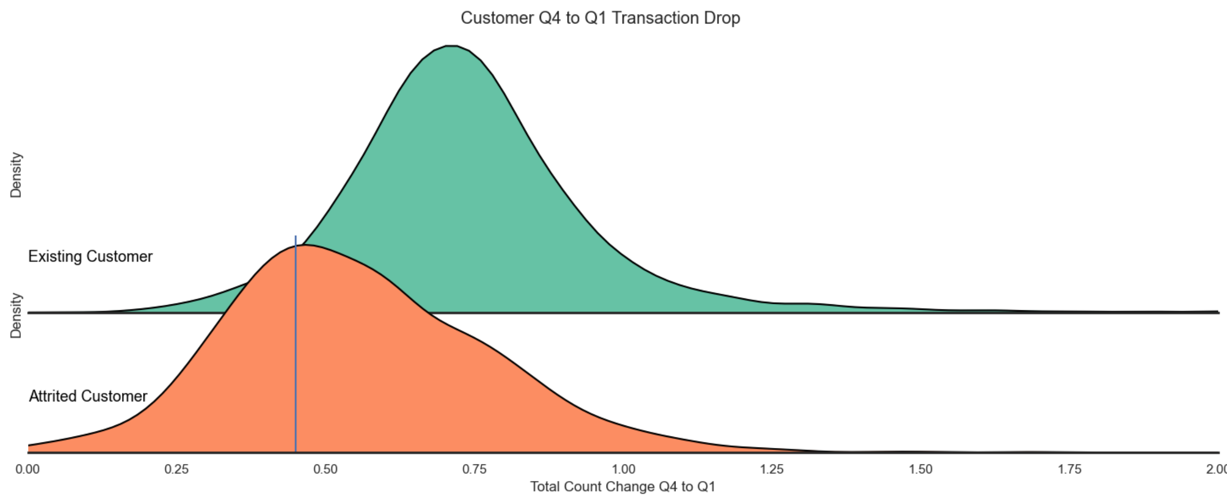
In [314…

```
## Can We Influence the Q4 to Q1 Dip?

sns.set_theme(style='white', rc={'axes.facecolor': (0, 0, 0 , 0), 'axes.linewidth':2})
palette = sns.color_palette("Set2", 12)
g = sns.FacetGrid(dataset, palette = palette, row = "Attrition_Flag", hue="Attrition_F
g.map_dataframe(sns.kdeplot, x="Total_Ct_Chng_Q4_Q1", fill=True, alpha=1)
g.map_dataframe(sns.kdeplot, x='Total_Ct_Chng_Q4_Q1', color='black')
def label(x, color, label):
```

```
    ax = plt.gca()
    ax.text(0, .2, label, color ='black', fontsize = 13,
            ha = "left", va='center', transform=ax.transAxes)
g.map(label, "Attrition_Flag")
g.fig.subplots_adjust(hspace=-.5)
g.set_titles("")
g.set(yticks=[], xlabel="Total Count Change Q4 to Q1")
g.despine(left=True)
plt.suptitle('Customer Q4 to Q1 Transaction Drop', y=0.98)
plt.vlines(0.45, 0 , 2) # adding vertical line at the $11k spending
plt.xlim(0,2)
```

Out[314]:

```
(0.0, 2.0)
```



Recommendations

- Can we implement a marketing re-engagement campaign to "prevent the cliff" for those who have seen this in the past?
- How big is this group and what is the potential opportunity?
- Customers surveys
- Offer loyalty points, cash back, etc.
- Offer more loyality points to customers reaching total spending greater than 11k dollars
- Next Steps: we can look at any historical marketing campaigns to learn from what worked and didn't work.

In [ ]: