#Credit Card Consumers Churn Rate

Data Gathering

I used the website "Kaggle" to search for datasets that can be used. I have chosen three datasets that can be potentially used for the project. In the end, we have chosen the Credit Card customers dataset and downloaded the "BankChurners.csv" file and used this csv file as our main dataset.

Cleansing Data

In order to scrub the data I decided to check for NaN values using the code old_bank.isnull().sum().sum() and then I decided to check if there are any missing values using the code old_bank.isna().sum().sum() in order to filter out and have a clean dataset that can be used. I also renamed some of the column heading.

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import io

# importing libraries
```

In [2]:
```
old_bank = pd.read_csv(r"..\Group 2 - CS170_BM1\old_bank.csv")

# importing dataset of old_bank
```

In [3]:
```
print(old_bank.shape)

# to display number of rows and columns
```

```
(10127, 23)
```

In [4]:
```
old_bank.isnull().sum().sum()

# to check if there are any NaN values in the dataset
```

Out[4]:
```
0
```

In [5]:
```
old_bank.isna().sum().sum()

# to check if there are any missing values in the dataset
```

Out[5]:
```
0
```

In [6]:
```
old_bank = old_bank.rename(columns = {'Attrition_Flag' : 'Customer_Attrition' , '

# to rename some of the column heading
```

In [7]:
```
old_bank.head(10)

# to show the first 10 records
```

Out[7]:

	CLIENTNUM	Customer_Attrition	Customer_Age	Gender	Dependent_count	Education_Level	Ma
0	768805383	Existing Customer	45	M	3	High School	
1	818770008	Existing Customer	49	F	5	Graduate	
2	713982108	Existing Customer	51	M	3	Graduate	
3	769911858	Existing Customer	40	F	4	High School	
4	709106358	Existing Customer	40	M	3	Uneducated	
5	713061558	Existing Customer	44	M	2	Graduate	
6	810347208	Existing Customer	51	M	4	Unknown	
7	818906208	Existing Customer	32	M	0	High School	
8	710930508	Existing Customer	37	M	3	Uneducated	
9	719661558	Existing Customer	48	M	2	Graduate	

10 rows × 23 columns

Exploratory Data Analysis

General Problem Statement: To predict which customers will not be using the amenities of the bank in the future.

Data Science Questions:

1. Does gender and marital status influence the churn rate of the bank?
2. Is there a correlation between the attrited customers and the number of inactive months?
3. Is there a correlation between the total transaction amount and total transaction count?

Data Inspection: Upon inspecting the data, we have confirmed that there are two types of data namely, numerical and nominal data. However, we change the data type of Customer_Attrition to Numerical Data because we needed it for the correlation

Nominal Data: Gender, Education_Level, Marital_Status, Customer_Income, Customer_Card_Category

Numerical Data: Customer_Attrition, CLIENTNUM, Customer_Age, Dependent_count, Months_on_book, Total_Relationship_Count, Months_Inactive, Contacts_Count_Month, Credit_Limit, Total_Revolving_Balance, Total_Credit_used, Total_Amount_Change, Total_Transaction_Amount, Total_Transaction_Count, Total_Count_Change, Average_Utiliation_Ratio,
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_coun
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_coun

In [8]:
```
old_bank['Customer_Attrition'].replace({'Existing Customer':0, 'Attrited Customer

# to change the nominal data "Existing Customer" and "Attrited Customer" to numer
```

In [9]:
```
old_bank.dtypes

# to inspect the properties of the updated datset
```

Out[9]:
```
CLIENTNUM
int64
Customer_Attrition
int64
Customer_Age
int64
Gender
object
Dependent_count
int64
Education_Level
object
Marital_Status
object
Customer_Income
object
Customer_Card_Category
object
Months_on_book
int64
Total_Relationship_Count
int64
Months_Inactive
int64
Contacts_Count_Month
int64
Credit_Limit
float64
Total_Revolving_Balance
int64
Total_Credit_used
float64
Total_Amount_Change
float64
Total_Transaction_Amount
int64
Total_Transaction_Count
int64
Total_Count_Change
float64
Average_Utilization_Ratio
float64
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Depen
dent_count_Education_Level_Months_Inactive_12_mon_1     float64
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Depen
dent_count_Education_Level_Months_Inactive_12_mon_2     float64
dtype: object
```

Variable Correlation: To show the correlation of the following variables:

1. Customer_Attrition with Gender

2. Customer Attrition with Marital_Status
3. Customer_Attrition with both Gender and Marital_status
4. Customer_Attrition and Inactive_Months
5. Total_Transaction_Amount and Total_Transaction_Count with Customer_Attrition

In [10]:
```
old_bank['Customer_Attrition'].value_counts(normalize=True)

# to show the percentage of current existing customer and attrited customers
# there is a 16% churn rate of customers
```

Out[10]:
```
0    0.83934
1    0.16066
Name: Customer_Attrition, dtype: float64
```

In [11]:
```
old_bank[['Customer_Attrition','Gender']].\
groupby(['Gender']).agg(['mean','count']).round(2)

# to show the correlation of Customer_Attrition with Gender
# the mean is the churn rate of customers
# the count shows the number of Female and Male customers
```

Out[11]:

	Customer_Attrition	
	mean	**count**
Gender		
F	0.17	5358
M	0.15	4769

In [12]:
```
old_bank[['Customer_Attrition','Marital_Status']].\
groupby(['Marital_Status']).agg(['mean','count']).round(2)

# to show the correlation of Customer_Attrition with Marital_Status
# the mean is the churn rate of customers
# the count shows the number of Divorced, Married, Single, and Unknown customers
```

Out[12]:

	Customer_Attrition	
	mean	**count**
Marital_Status		
Divorced	0.16	748
Married	0.15	4687
Single	0.17	3943
Unknown	0.17	749

In [13]:

```
old_bank[['Customer_Attrition','Gender','Marital_Status']].\
groupby(['Gender','Marital_Status']).agg(['mean','count']).round(2)

# to show the correlation of Customer_Attrition with both Gender and Marital_Stat
# the mean is the churn rate of customers
# for the Gender F and M, the count shows the number of Divorced, Married, Single
```

Out[13]:

		Customer_Attrition	
		mean	count
Gender	**Marital_Status**		
F	**Divorced**	0.17	402
	Married	0.17	2451
	Single	0.18	2125
	Unknown	0.18	380
M	**Divorced**	0.15	346
	Married	0.13	2236
	Single	0.16	1818
	Unknown	0.16	369

In [14]:

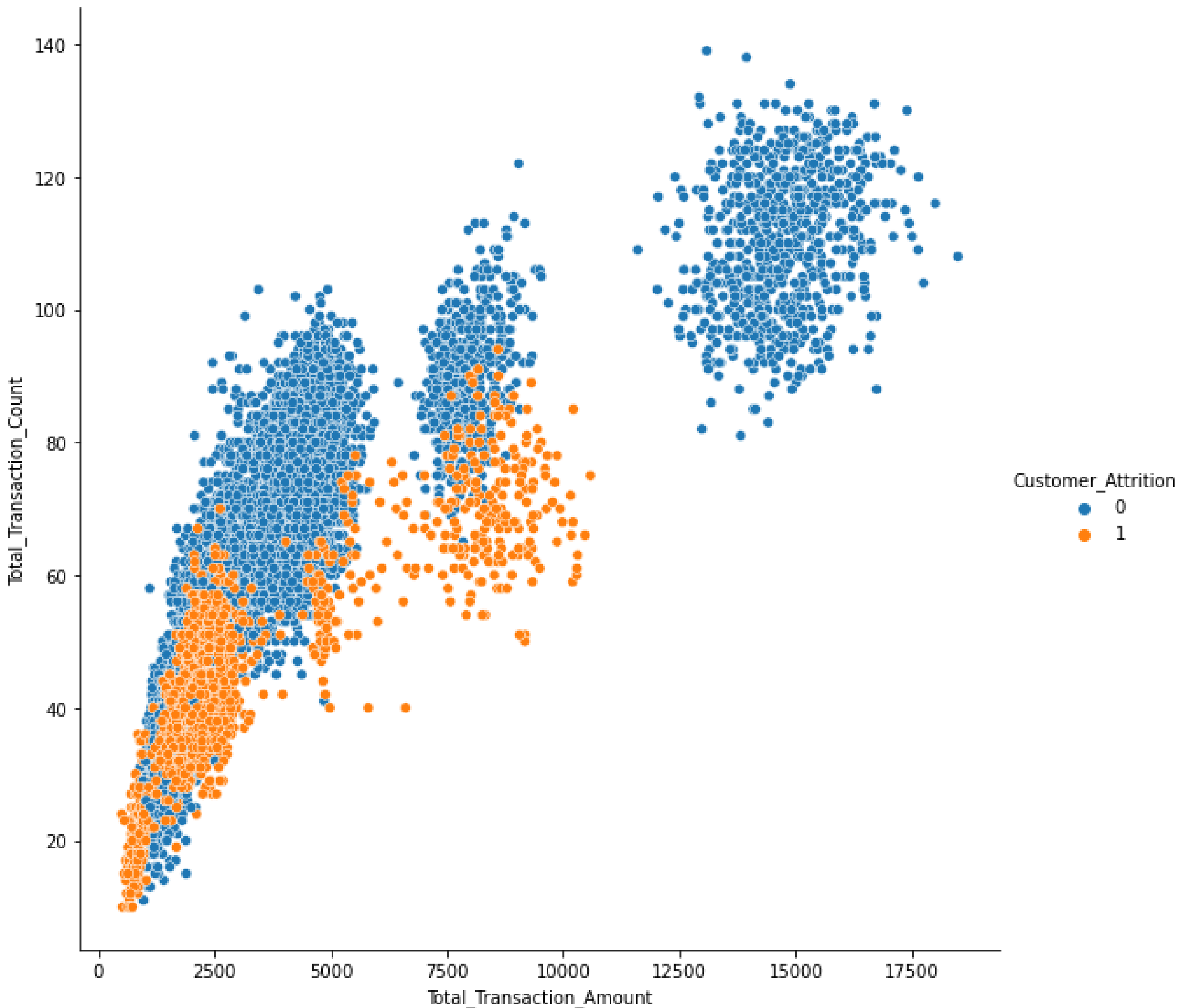
```
old_bank[['Customer_Attrition','Months_Inactive']].\
groupby(['Months_Inactive']).agg(['mean','count']).round(2)

# to show the correlation of Customer_Attrition with Months_Inactive
# the mean is the churn rate of customers
# the count shows the number of customers that have been inactive based on the nu
```

Out[14]:

	Customer_Attrition	
	mean	count
Months_Inactive		
0	0.52	29
1	0.04	2233
2	0.15	3282
3	0.21	3846
4	0.30	435
5	0.18	178
6	0.15	124

In [15]:
```
sns.relplot(data=old_bank, kind='scatter', x='Total_Transaction_Amount', y='Total

# to show the correlation between Total_Transaction_Amount and Total_Transaction_
# the '0' is the existing customer
# the '1' is the attrited customer
```

Out[15]: <seaborn.axisgrid.FacetGrid at 0x1e5e1c38610>

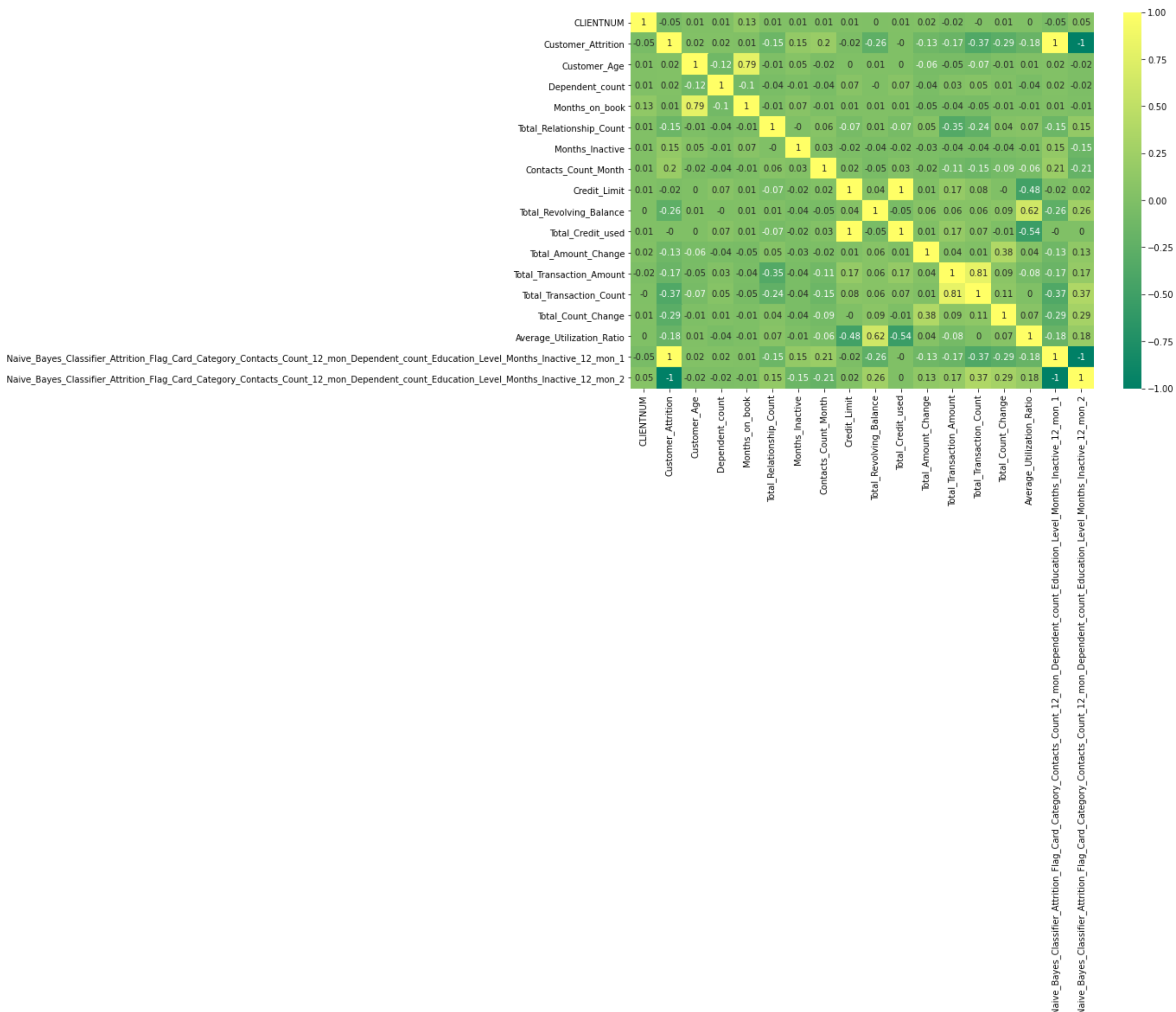


Feature Extraction: We filtered the dataset using correlation matrix by using Pearson correlation which is under the Filter Method. We can see the correlation of different variables with each other by plotting using the Pearson correlation heatmap

In [16]:
```
hmap = old_bank.corr().round(2)
plt.figure(figsize=(12,8))
sns.heatmap(hmap, annot=True, cmap="summer")

# to display the correlations of the variables with other variables
# the 1st row is highly needed because it shows the correlation of Customer_Attri
# the closer to the value of 1, the stronger the positive correlation
# the closer to the value of -1, the stronger the negative correlation
# the closer to the value of 0, the weaker the correlation or if 0, there is no c
```

Out[16]:

```
<AxesSubplot:>
```

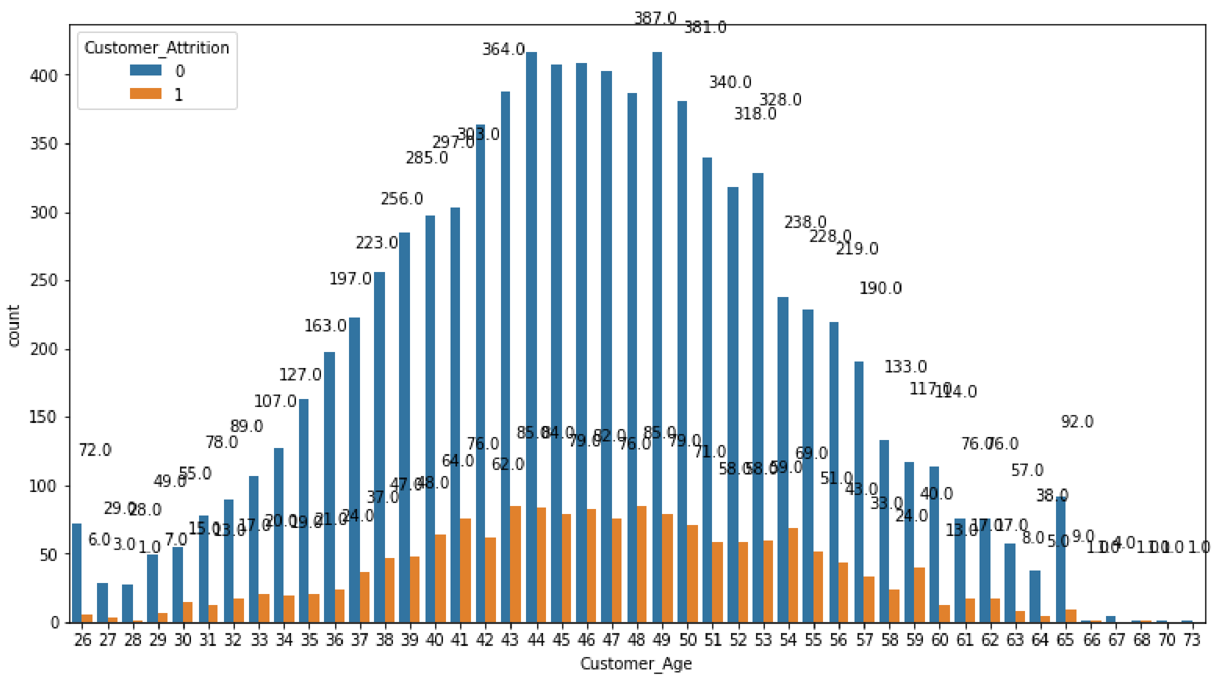


In [17]:
```
hmap_target = abs(hmap['Customer_Attrition']).sort_values(ascending=False)
hmap_target

# to show the correlation of Customer_Attrition with the other variables
# all the values have been changed to absolute value because all of the correlati
```

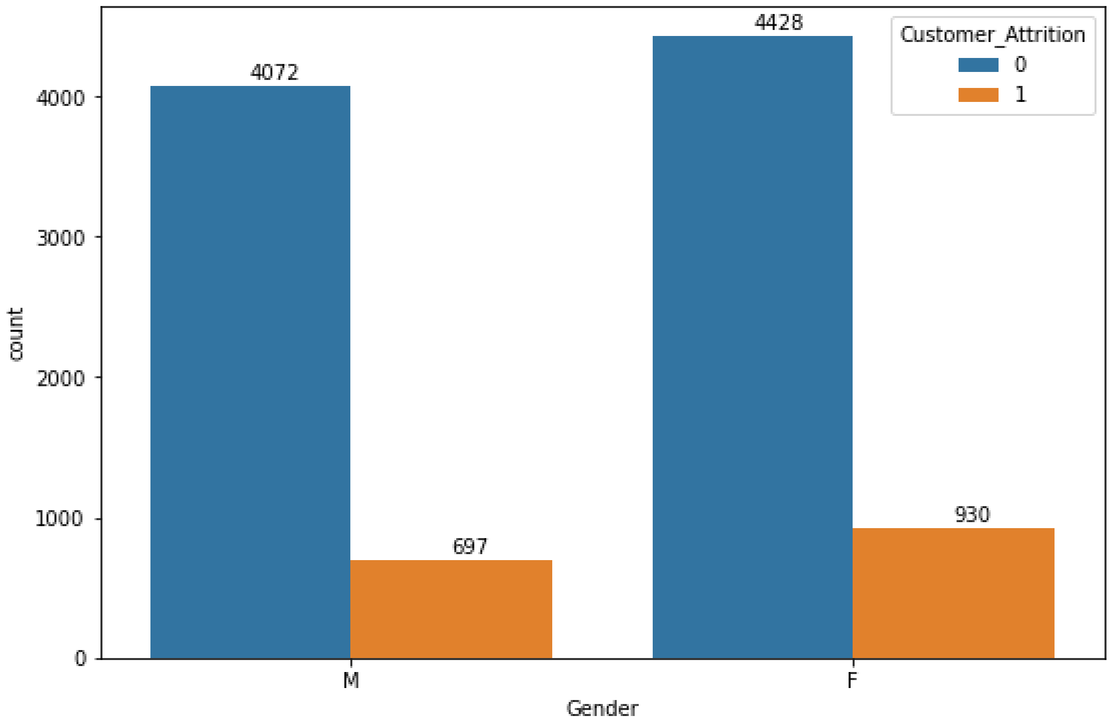
Out[17]:
```
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Depen
dent_count_Education_Level_Months_Inactive_12_mon_2    1.00
Customer_Attrition
1.00
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Depen
dent_count_Education_Level_Months_Inactive_12_mon_1    1.00
Total_Transaction_Count
0.37
Total_Count_Change
0.29
Total_Revolving_Balance
0.26
Contacts_Count_Month
0.20
Average_Utilization_Ratio
0.18
Total_Transaction_Amount
0.17
Total_Relationship_Count
0.15
Months_Inactive
0.15
Total_Amount_Change
0.13
CLIENTNUM
0.05
Dependent_count
0.02
Customer_Age
0.02
Credit_Limit
0.02
Months_on_book
0.01
Total_Credit_used
0.00
Name: Customer_Attrition, dtype: float64
```

Data Visualization: We wanted to show the data in forms of simple charts such as bar charts and scatter plot chart to get a better understanding of the data

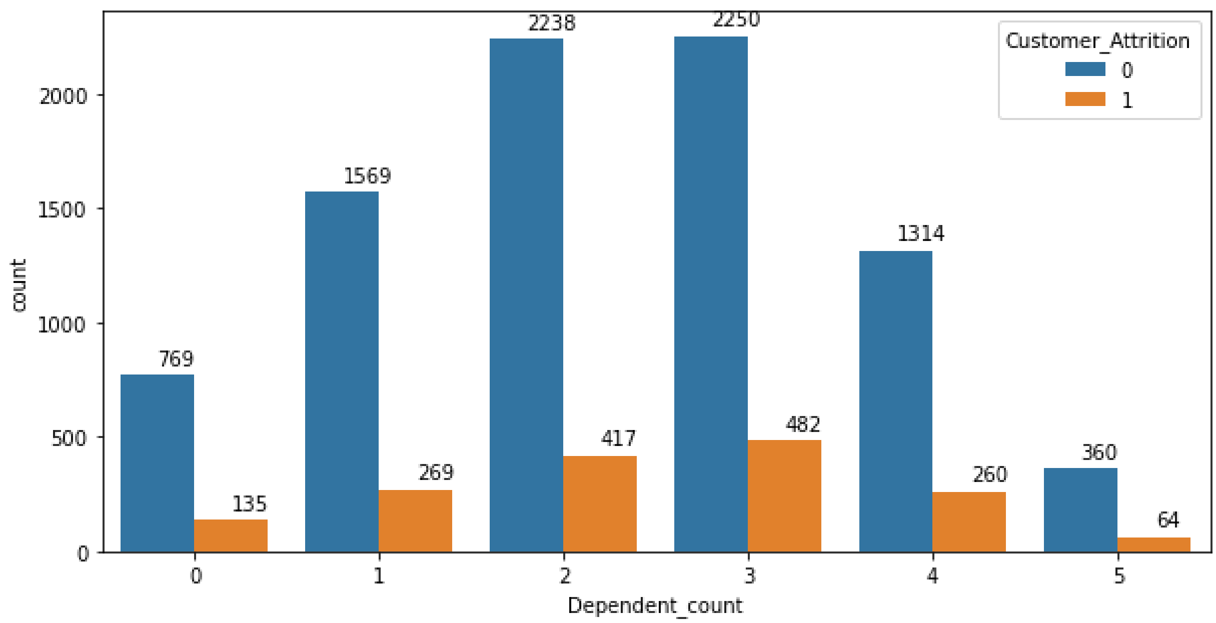
In [18]:
```
plt.figure(figsize=(13,7))
plot=sns.countplot(x=old_bank.Customer_Age,hue=old_bank.Customer_Attrition)
for p in plot.patches:
    plot.annotate(p.get_height(),(p.get_x()+p.get_width()/2,p.get_height()+50))
#plt.xticks(rotation=90)
plt.show()

# to show the count of Customer_Attrition in the Customer_Age: 26, 27, 28, 29, 30
# wherein '0' is the existing customer and '1' is the attrited customer
```



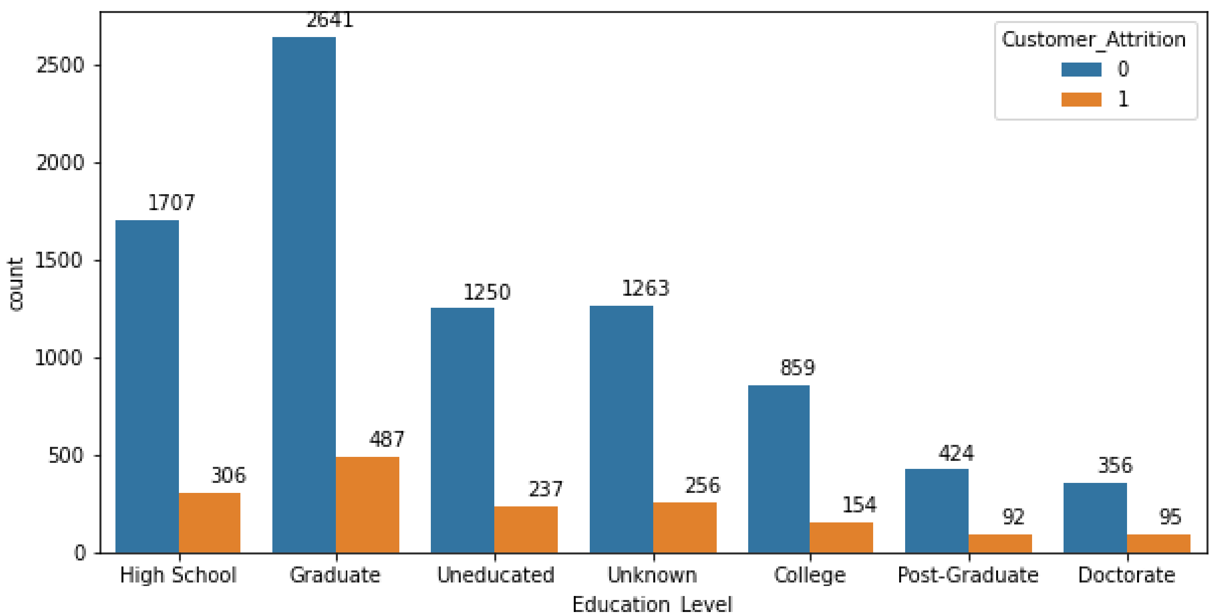
In [19]:
```
plt.figure(figsize=(9,6))
plot=sns.countplot(x=old_bank.Gender,hue=old_bank.Customer_Attrition)
for p in plot.patches:
  plot.annotate(p.get_height(),(p.get_x()+p.get_width()/2,p.get_height()+50))
plt.show()

# to show the count of Customer_Attrition in the Gender: M and F
# wherein '0' is the existing customer and '1' is the attrited customer
```



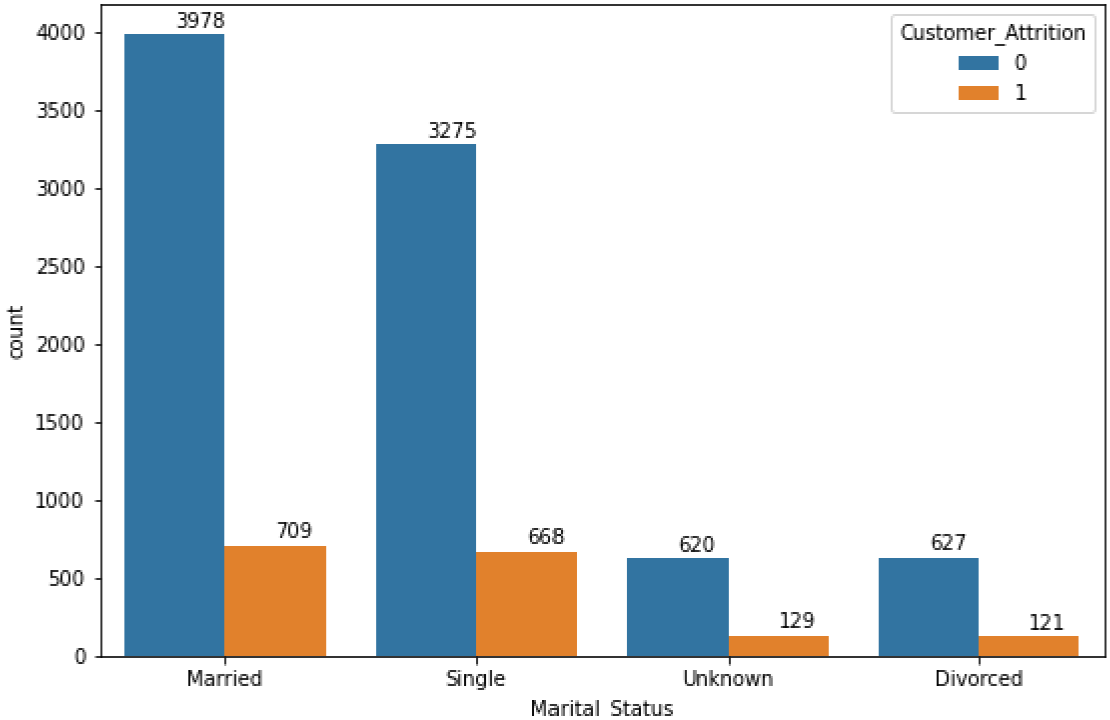
In [20]:
```
plt.figure(figsize=(10,5))
plot=sns.countplot(x=old_bank.Dependent_count,hue=old_bank.Customer_Attrition)
for p in plot.patches:
    plot.annotate(p.get_height(),(p.get_x()+p.get_width()/2,p.get_height()+50))
plt.show()

# to show the count of Customer_Attrition in the Dependent_count: 0, 1, 2, 3, 4,
# wherein '0' is the existing customer and '1' is the attrited customer
```



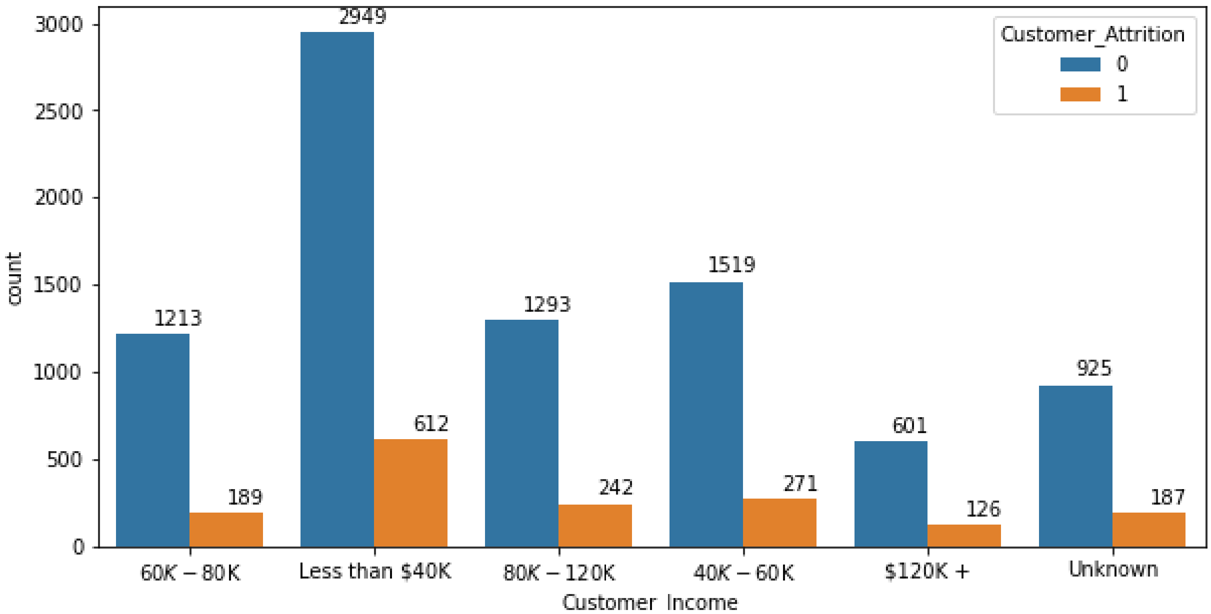
In [21]:
```
plt.figure(figsize=(10,5))
plot=sns.countplot(x=old_bank.Education_Level,hue=old_bank.Customer_Attrition)
for p in plot.patches:
    plot.annotate(p.get_height(),(p.get_x()+p.get_width()/2,p.get_height()+50))#p
plt.show()

# to show the count of Customer_Attrion in the Education_Level: High School, Grad
# wherein '0' is the existing customer and '1' is the attrited customer
```



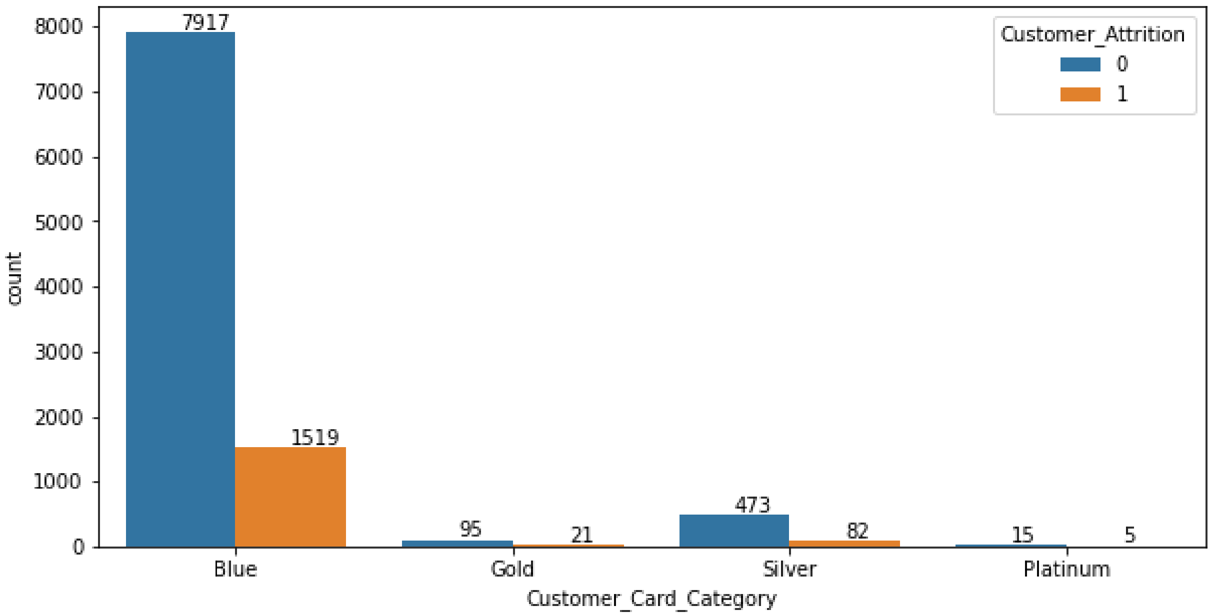
In [22]:
```
plt.figure(figsize=(9,6))
plot=sns.countplot(x=old_bank.Marital_Status,hue=old_bank.Customer_Attrition)
for p in plot.patches:
  plot.annotate(p.get_height(),(p.get_x()+p.get_width()/2,p.get_height()+50))
plt.show()

# to show the count of Customer_Attrion in the Marital_Status: Married, Single, U
# wherein '0' is the existing customer and '1' is the attrited customer
```



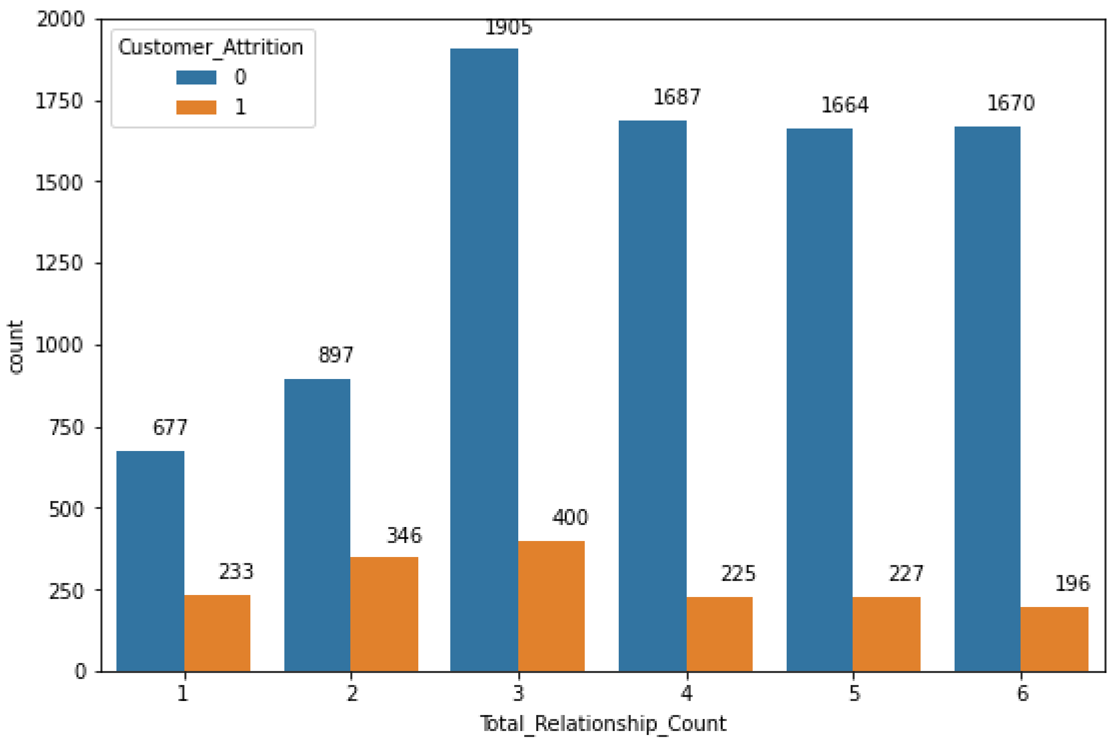
In [23]:
```
plt.figure(figsize=(10,5))
plot=sns.countplot(x=old_bank.Customer_Income,hue=old_bank.Customer_Attrition)
for p in plot.patches:
    plot.annotate(p.get_height(),(p.get_x()+p.get_width()/2,p.get_height()+50))
plt.show()

# to show the count of Customer_Attrion in the Customer_Income: 60k-80k, Less tha
# wherein '0' is the existing customer and '1' is the attrited customer
```



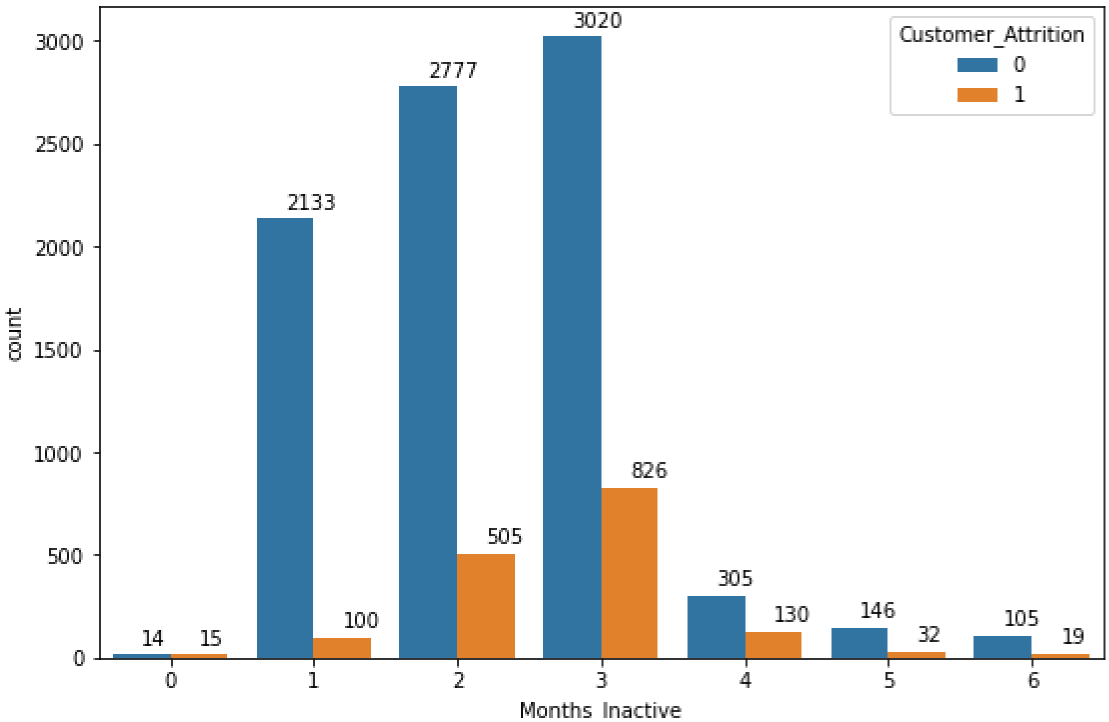
In [24]:
```
plt.figure(figsize=(10,5))
plot=sns.countplot(x=old_bank.Customer_Card_Category,hue=old_bank.Customer_Attrit
for p in plot.patches:
    plot.annotate(p.get_height(),(p.get_x()+p.get_width()/2,p.get_height()+50))
plt.show()

# to show the count of Customer_Attrion in the Customer_Card_Category: Blue, Gold
# wherein '0' is the existing customer and '1' is the attrited customer
```



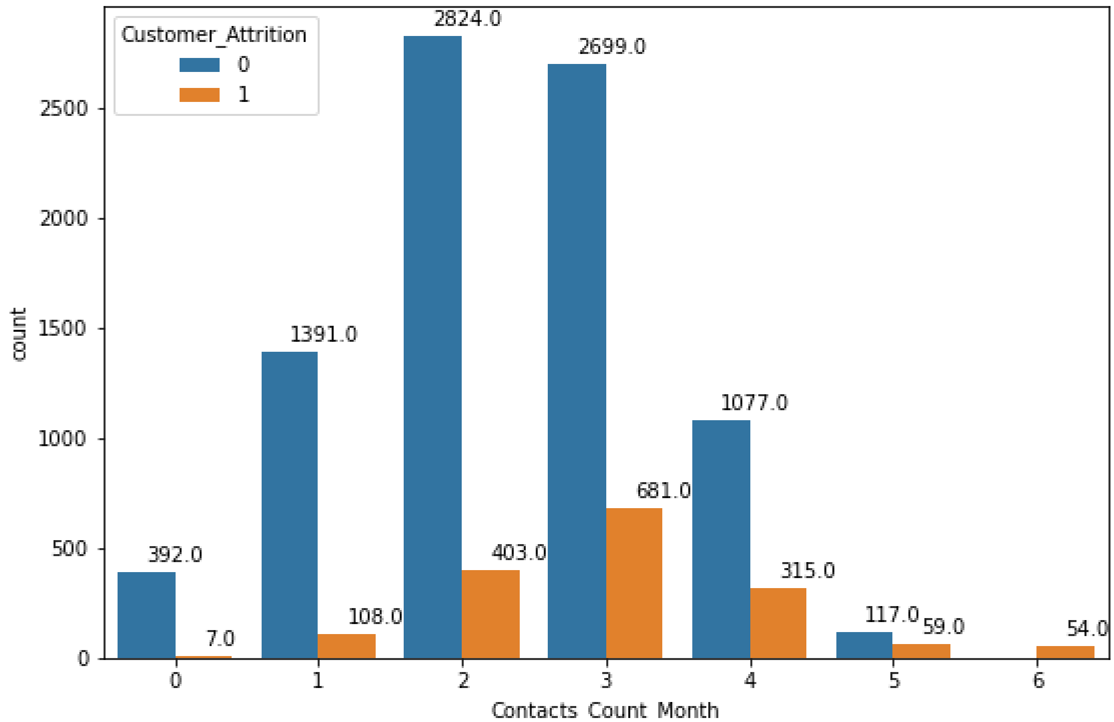
In [25]:
```
plt.figure(figsize=(9,6))
plot=sns.countplot(x=old_bank.Total_Relationship_Count,hue=old_bank.Customer_Attr
for p in plot.patches:
  plot.annotate(p.get_height(),(p.get_x()+p.get_width()/2,p.get_height()+50))
plt.show()

# to show the count of Customer_Attrion in the Total_Relationshop_Count: 1, 2, 3,
# wherein '0' is the existing customer and '1' is the attrited customer
```



In [26]:
```
plt.figure(figsize=(9,6))
plot=sns.countplot(x=old_bank.Months_Inactive,hue=old_bank.Customer_Attrition)
for p in plot.patches:
  plot.annotate(p.get_height(),(p.get_x()+p.get_width()/2,p.get_height()+50))
plt.show()

# to show the count of Customer_Attrion in the Months_Inactive: 0, 1, 2, 3, 4, 5,
# wherein '0' is the existing customer and '1' is the attrited customer
```

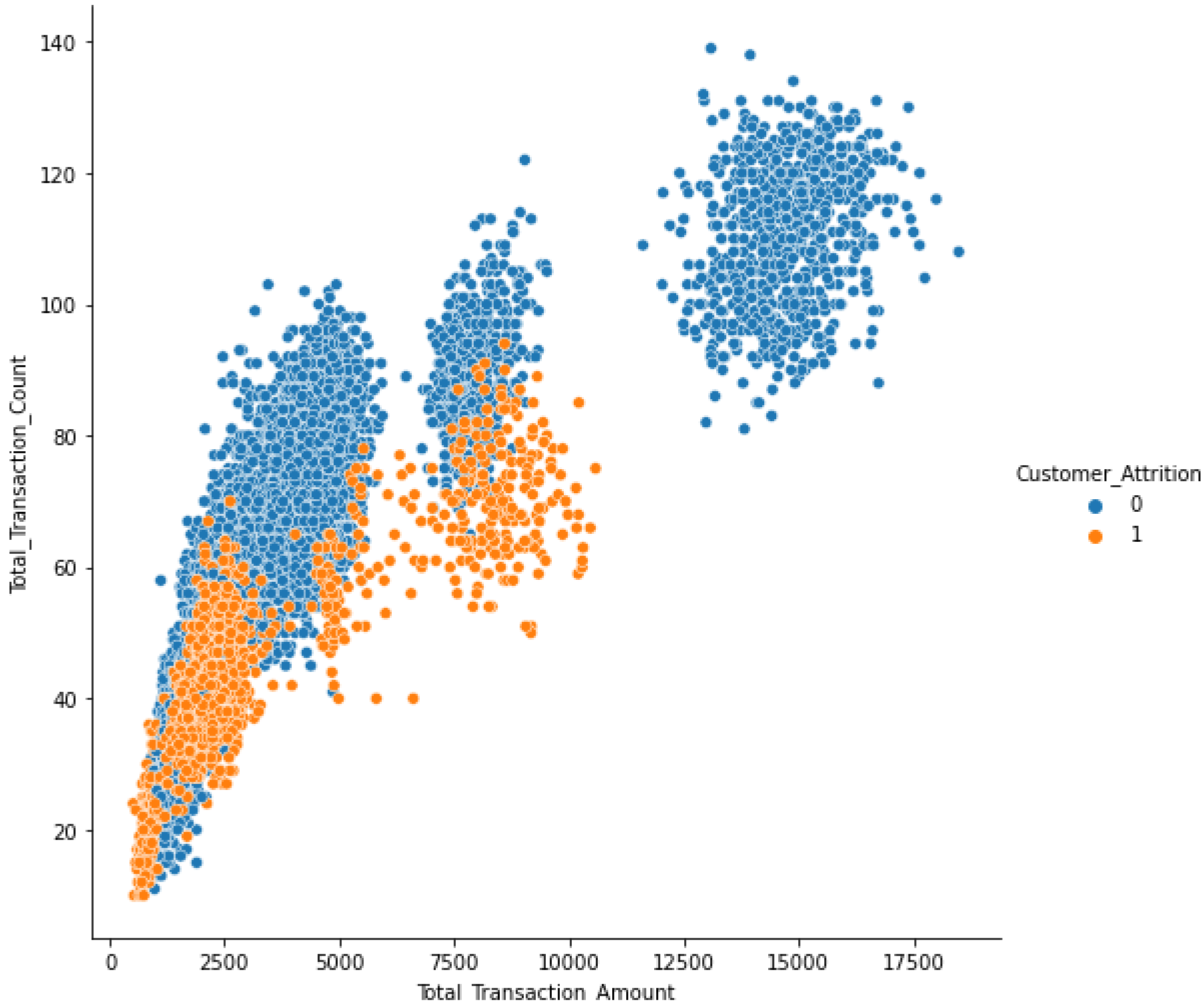


In [27]:
```
plt.figure(figsize=(9,6))
plot=sns.countplot(x=old_bank.Contacts_Count_Month,hue=old_bank.Customer_Attritic
for p in plot.patches:
  plot.annotate(p.get_height(),(p.get_x()+p.get_width()/2,p.get_height()+50))
plt.show()

# to show the count of Customer_Attrion in the Contacts_Count_Month: 0, 1, 2, 3,
# wherein '0' is the existing customer and '1' is the attrited customer
```



In [28]:
```
sns.relplot(data=old_bank, kind='scatter', x='Total_Transaction_Amount', y='Total

# to show the count and compare the Total_Transaction_Amount with the Total_Trans
# wherein '0' is the existing customer and '1' is the attrited customer
```

Out[28]: <seaborn.axisgrid.FacetGrid at 0x1e5e369a940>



Modelling

Features: The columns that I have selected are the following: Customer_Attrition, Gender, Marital_Status, Months_Inactive, Total_Transaction_Amount, and Total_Transaction_Acount. I only selected this columns because they will help me answer my data science questions.

In [29]:
```
bank = pd.read_csv(r"..\Group 2 - CS170_BM1\bank.csv")

# importing libraries and datasets
```

In [30]:
```
print(bank.shape)

# to display number of rows and columns
```

```
(10127, 6)
```

In [31]:
```
bank.isna().sum().sum()

# to check if there are any missing values in the dataset
```

Out[31]:
```
0
```

In [32]:
```
bank.isnull().sum().sum()

# to check if there are any missing values in the dataset
```

Out[32]:
```
0
```

In [33]:
```
bank.head(10)

# to show the first 10 records
```

Out[33]:

	Customer_Attrition	Gender	Marital_Status	Months_Inactive	Total_Transaction_Amount	Total_Tra
0	Existing Customer	M	Married	1	1144	
1	Existing Customer	F	Single	1	1291	
2	Existing Customer	M	Married	1	1887	
3	Existing Customer	F	Unknown	4	1171	
4	Existing Customer	M	Married	1	816	
5	Existing Customer	M	Married	1	1088	
6	Existing Customer	M	Married	1	1330	
7	Existing Customer	M	Unknown	2	1538	
8	Existing Customer	M	Single	2	1350	
9	Existing Customer	M	Single	3	1441	

Train and Test the Model:

In [34]:
```
bank['Customer_Attrition'].replace({'Existing Customer':0, 'Attrited Customer':1]

# to change the nominal data "Existing Customer" and "Attrited Customer" to numer
```

In [35]:
```
bank.dtypes

# to show the properties of the dataset
```

Out[35]:
```
Customer_Attrition           int64
Gender                      object
Marital_Status              object
Months_Inactive              int64
Total_Transaction_Amount     int64
Total_Transaction_Count      int64
dtype: object
```

In [36]:
```
Total_Transaction_Amount = bank.iloc[:,4].values.reshape(-1,1)
# to gather all the rows of the 4th column which is the Total_Transaction_Amount


Total_Transaction_Count = bank.iloc[:,5].values
# to gather all the rows in the 5th column which is the Total_Transaction_Count
# to assign Total_Transaction_Amount and Total_Transaction_Count to the specifiec
```

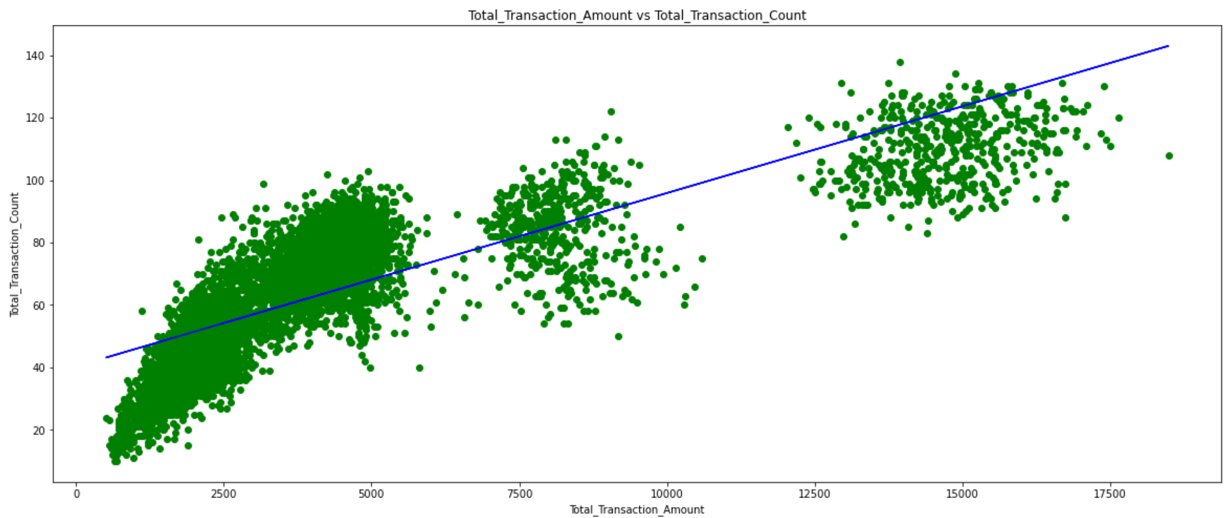
In [37]:
```
from sklearn.model_selection import train_test_split
# to import libraries

unit_train, unit_test, gross_train, gross_test = train_test_split(Total_Transacti
# to split the dataset into a train and test set
```

In [38]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(unit_train, gross_train)
# to train the simple linear regression model on the train set
```

Out[38]:
```
LinearRegression()
```

In [39]:
```
plt.figure(figsize=(20,8))
plt.scatter(unit_train, gross_train, color = 'green')
plt.plot(unit_train, regressor.predict(unit_train), color = 'blue')
plt.title('Total_Transaction_Amount vs Total_Transaction_Count')
plt.xlabel('Total_Transaction_Amount')
plt.ylabel('Total_Transaction_Count')

# to show the correlation of Total_Transaction_Amount with Total_Transaction_Cour
```

Out[39]:
```
Text(0, 0.5, 'Total_Transaction_Count')
```



Libraries Needed

1. import pandas as pd
2. import numpy as np
3. import seaborn as sns
4. import matplotlib.pyplot as plt
5. import io
6. from sklearn.model_selection import train_test_split
7. from sklearn.linear_model import LinearRegression
8. import statsmodels.tools.tools as stattools
9. from sklearn.tree import DecisionTreeClassifier

Validation and Evaluation Measurement

In [40]:
```
import statsmodels.tools.tools as stattools
from sklearn.tree import DecisionTreeClassifier

# importing libraries
```

In [41]:
```
y = bank['Customer_Attrition']
X = bank[['Total_Transaction_Amount', 'Total_Transaction_Count']]

# to specify the names of the combined matrix and the target variable
```

In [42]:
```
c50_01 = DecisionTreeClassifier(criterion="entropy", min_samples_split=75, max_le

# to identify the best split in the data
```

In [43]:
```
c50_01_predict = c50_01.predict(bank[['Total_Transaction_Amount', 'Total_Transact

# to obtain classifications
```

In [44]:
```
bar = pd.crosstab(bank['Customer_Attrition'], c50_01_predict)
bar

# to show the different data in Customer_Attrition
```

Out[44]:

col_0	0	1
Customer_Attrition		
0	7731	769
1	548	1079

In [45]:
```
def model_eval(matrix,model_name):
        tn = matrix.iloc[0,0]
        tp = matrix.iloc[1,1]
        fn = matrix.iloc[1,0]
        fp = matrix.iloc[0,1]
        tap = fn+tp
        tan = tn+fp
        tpn = tn+fn
        tpp = fp+tp
        precision = tp/tpp
        recall = tp/tap
        total = tn+tp+fn+fp
        data = [
                round((tp+tn)/total,4),
                round(1-((tp+tn)/total),4),
                round(tp/tap,4),
                round(tn/tan,4),
                round(precision,4),
                round(2 * (precision * recall) / (precision + recall),4),
                round(5 * (precision * recall) / ((4 * precision) + recall),4),
                round(1.25 * (precision * recall) / ((.25 * precision) + recall),
                ]
        return(
                pd.DataFrame(data, columns=[model_name],
                             index=['Accuracy','Error Rate','Sensitivity','Speci
                )

model_bank = model_eval(bar, model_name = 'model_bank')

# to compute for the evaluation measure
```

In [46]:
```
model_bank

# to show the evaluation measure
```

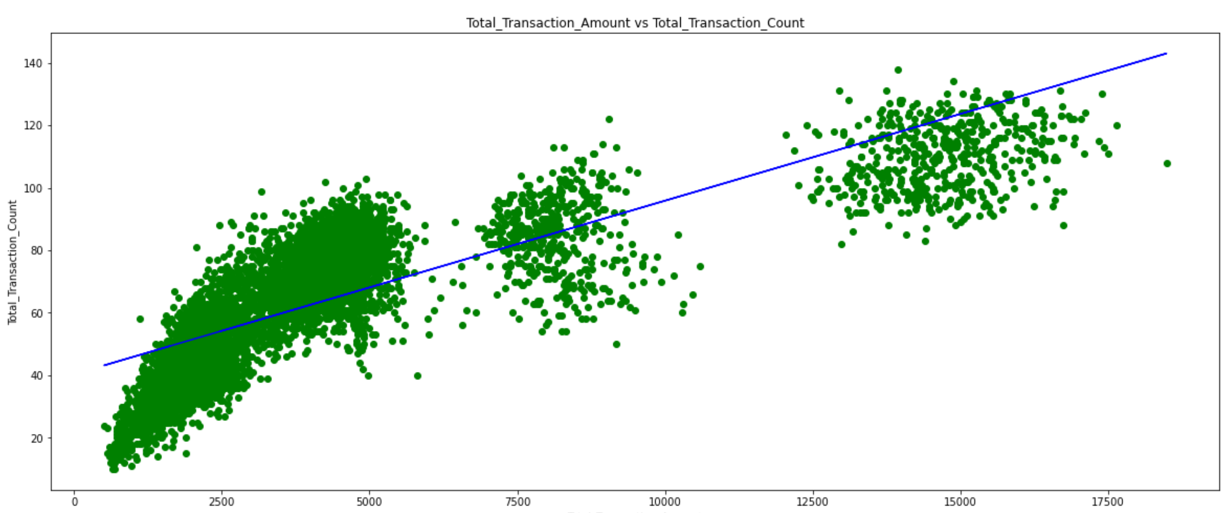
Out[46]:

	model_bank
Accuracy	0.8700
Error Rate	0.1300
Sensitivity	0.6632
Specificity	0.9095
Precision	0.5839
F1	0.6210
F2	0.6456
F0.5	0.5982

Evaluation (Interpreting Results)

In [47]:
```
plt.figure(figsize=(20,8))
plt.scatter(unit_train, gross_train, color = 'green')
plt.plot(unit_train, regressor.predict(unit_train), color = 'blue')
plt.title('Total_Transaction_Amount vs Total_Transaction_Count')
plt.xlabel('Total_Transaction_Amount')
plt.ylabel('Total_Transaction_Count')

# to show the correlation of Total_Transaction_Amount with Total_Transaction_Cour
```

Out[47]:
```
Text(0, 0.5, 'Total_Transaction_Count')
```



I have discovered that the Total_Transaction_Amount and the Total_Transaction_Count is highly correlated with each other. As the Total_Transaction_Amount goes up the Total_Transaction_Count also goes up.

In [48]:
```
model_bank

# to show the evaluation measure
```

Out[48]:

	model_bank
Accuracy	0.8700
Error Rate	0.1300
Sensitivity	0.6632
Specificity	0.9095
Precision	0.5839
F1	0.6210
F2	0.6456
F0.5	0.5982

Results of the Evaluation Measurement To calibrate the accuracy of model_bank, the all negative model was used as a baseline. The following information was then interpreted

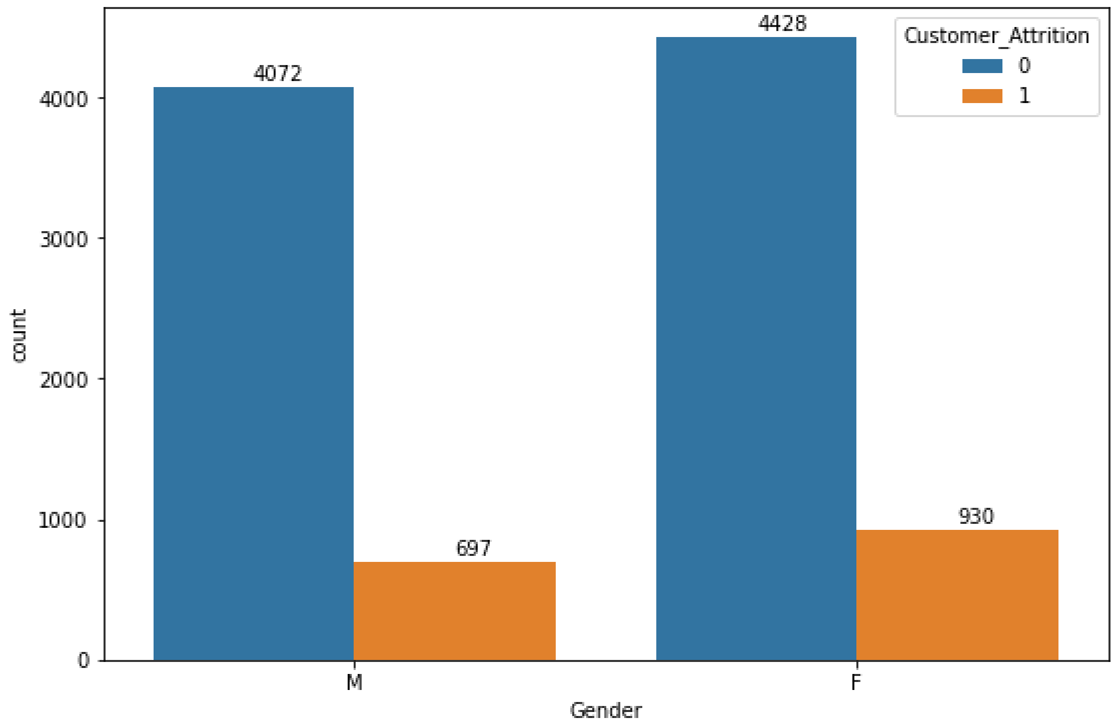
1. The model has an Accuracy of 87.00%
2. The model has an Error rate of 13.00%
3. A Specificity of 0.9095 means the model correctly classifies 90.95% of the actual negative records as negative.
4. A Sensitivity of 0.6632 means that 66.32% of the actual positive records were classified as positive.
5. A Precision of 0.5839 means that 58.39% of the data would respond positively
6. F1 has a value of 0.6210, F2 has a value of 0.6456 which is close to Sensitivity, F0.5 has a value of 0.5982 which is close to the precision.
7. The Fscore are compared directly to choose the best model outcome

Useful Findings

General Problem Explanation: In order to solve the general problem statement, we decided to create and answer 3 data science questions that would prove our findings. We used different graphs and correlations of variables to predict which customers will not be using the amenities of the bank in the future.

First Data Science Question: Does gender and marital status influence the churn rate of the bank?

In [49]:
```
plt.figure(figsize=(9,6))
plot=sns.countplot(x=bank.Gender,hue=bank.Customer_Attrition)
for p in plot.patches:
  plot.annotate(p.get_height(),(p.get_x()+p.get_width()/2,p.get_height()+50))
plt.show()

# to show the count of Customer_Attrition in the Gender: M and F
# wherein '0' is the existing customer and '1' is the attrited customer
```



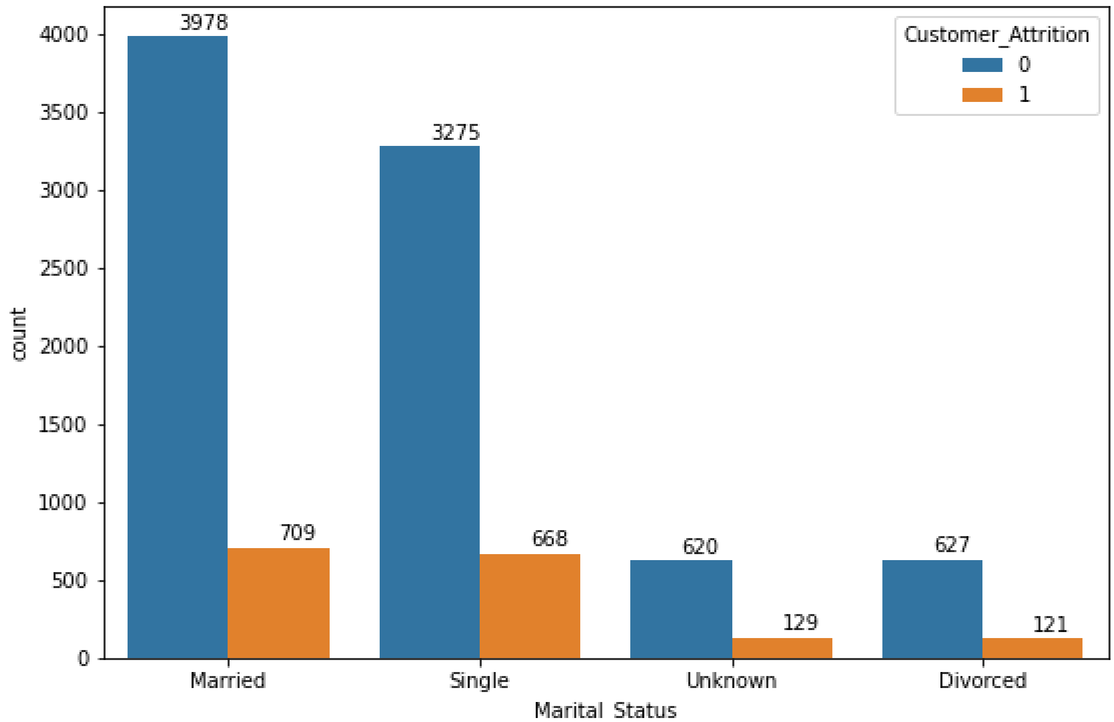
In [50]:
```
bank[['Customer_Attrition','Gender']].\
groupby(['Gender']).agg(['mean','count']).round(2)

# to show the correlation of Customer_Attrition with Gender
# the mean is the churn rate of customers
# the count shows the number of Female and Male customers
```

Out[50]:

	Customer_Attrition	
	mean	count
Gender		
F	0.17	5358
M	0.15	4769

The female customers have a 17% churn rate. The male customers have a 15% churn rate.

In [51]:
```
plt.figure(figsize=(9,6))
plot=sns.countplot(x=bank.Marital_Status,hue=bank.Customer_Attrition)
for p in plot.patches:
  plot.annotate(p.get_height(),(p.get_x()+p.get_width()/2,p.get_height()+50))
plt.show()

# to show the count of Customer_Attrion in the Marital_Status: Married, Single, U
# wherein '0' is the existing customer and '1' is the attrited customer
```



In [52]:
```
bank[['Customer_Attrition','Marital_Status']].\
groupby(['Marital_Status']).agg(['mean','count']).round(2)

# to show the correlation of Customer_Attrition with Marital_Status
# the mean is the churn rate of customers
# the count shows the number of Divorced, Married, Single, and Unknown customers
```

Out[52]:

	Customer_Attrition	
	mean	**count**
Marital_Status		
Divorced	0.16	748
Married	0.15	4687
Single	0.17	3943
Unknown	0.17	749

The divorced customers have a 16% churn rate. The married customers have a 15% churn rate. The single customers have a 17% churn rate. The unknown customers have a 17% churn rate.

In [53]:
```
bank[['Customer_Attrition','Gender','Marital_Status']].\
groupby(['Gender','Marital_Status']).agg(['mean','count']).round(2)

# to show the correlation of Customer_Attrition with both Gender and Marital_Stat
# the mean is the churn rate of customers
# for the Gender F and M, the count shows the number of Divorced, Married, Single
```

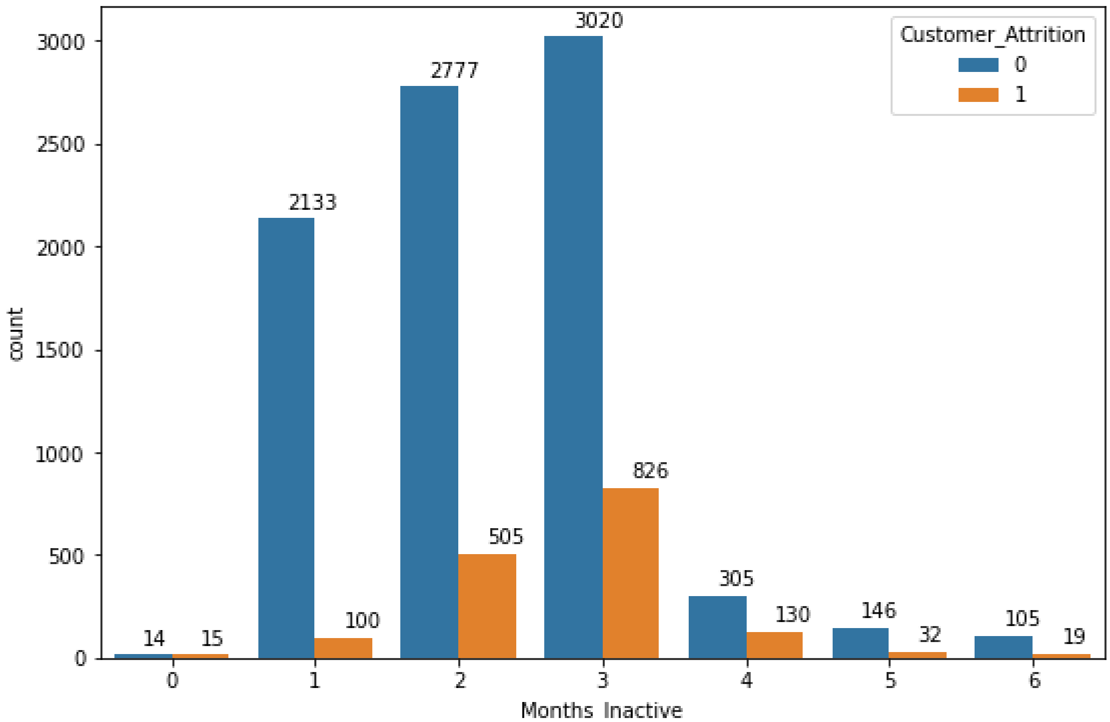
Out[53]:

		Customer_Attrition	
		mean	**count**
Gender	**Marital_Status**		
F	**Divorced**	0.17	402
	Married	0.17	2451
	Single	0.18	2125
	Unknown	0.18	380
M	**Divorced**	0.15	346
	Married	0.13	2236
	Single	0.16	1818
	Unknown	0.16	369

First Data Science Question Answer: The female customers have a much higher churn rate than the male customers. Which means that the female customers are more likely to churn than the male customers. An example of a scenario could be that a bank should better improve its amenities to avoid having a high churn rate of its customers.

Second Data Science Question: Is there a correlation between the attrited customers and the number of inactive months?

In [54]:
```
plt.figure(figsize=(9,6))
plot=sns.countplot(x=old_bank.Months_Inactive,hue=old_bank.Customer_Attrition)
for p in plot.patches:
  plot.annotate(p.get_height(),(p.get_x()+p.get_width()/2,p.get_height()+50))
plt.show()

# to show the count of Customer_Attrion in the Months_Inactive: 0, 1, 2, 3, 4, 5,
# wherein '0' is the existing customer and '1' is the attrited customer
```



In [55]:
```
old_bank[['Customer_Attrition','Months_Inactive']].\
groupby(['Months_Inactive']).agg(['mean','count']).round(2)

# to show the correlation of Customer_Attrition with Months_Inactive
# the mean is the churn rate of customers
# the count shows the number of customers that have been inactive based on the nu
```

Out[55]:

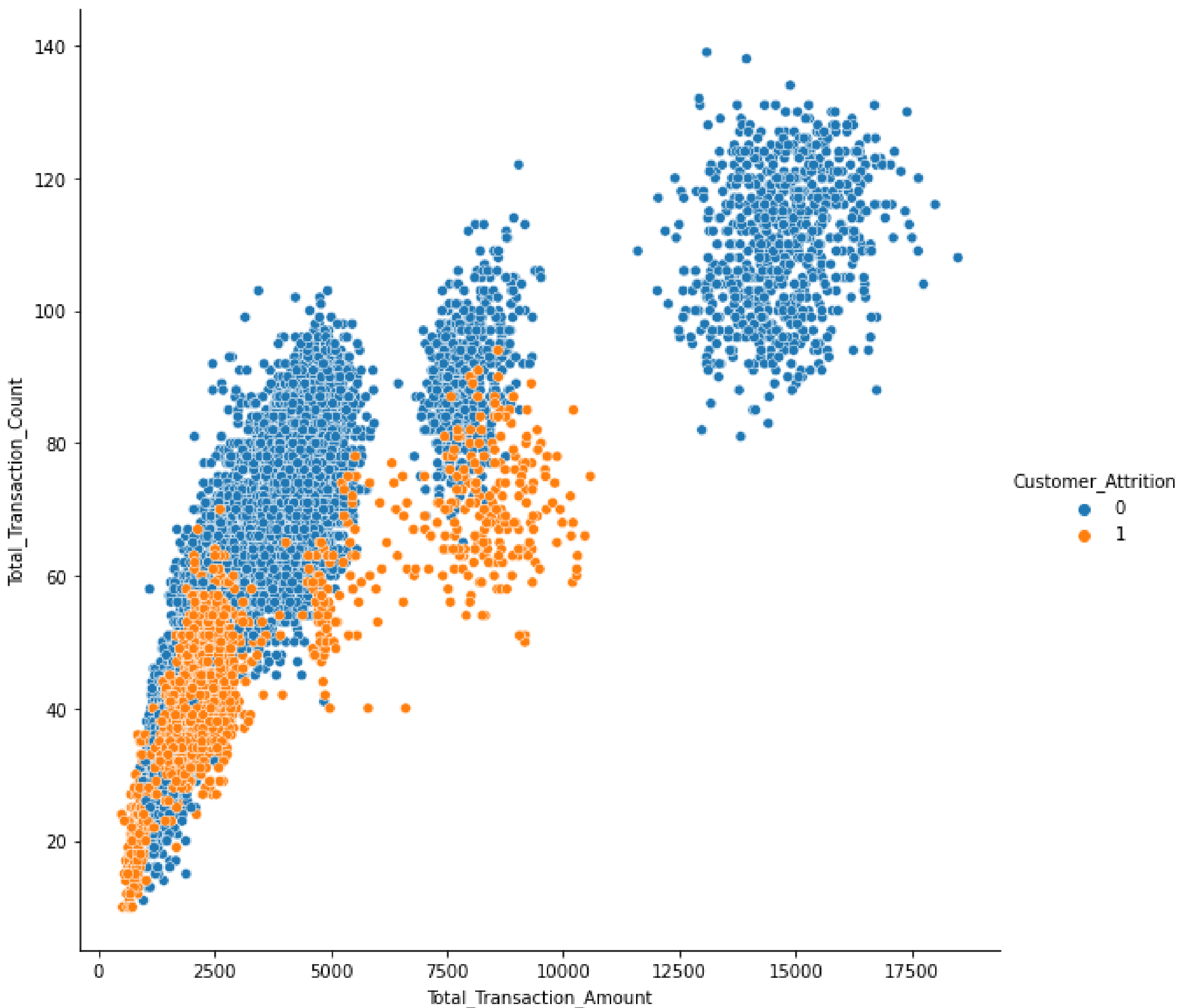
	Customer_Attrition	
	mean	count
Months_Inactive		
0	0.52	29
1	0.04	2233
2	0.15	3282
3	0.21	3846
4	0.30	435
5	0.18	178
6	0.15	124

Second Data Science Question Answer: There is a correlation betwen the attrited customers and the number of inactive months because the churn rate of the customers increases as the number of inactive months increases, excluding the months when there were only a few customers. An example of a scenario could be that the bank kasi anticipate the churn rate of customers depending on the months they are inactive and they could prevent some of the churn rate of the customers.

Third Data Science Question: Is there a correlation between the total transaction amount and total transaction count?

In [56]:
```
sns.relplot(data=old_bank, kind='scatter', x='Total_Transaction_Amount', y='Total

# to show the correlation between Total_Transaction_Amount and Total_Transaction_
# the '0' is the existing customer
# the '1' is the attrited customer
```

Out[56]: <seaborn.axisgrid.FacetGrid at 0x1e5e57e9ca0>



Third Data Science Question Answer: There is a correlation between the total transaction amount and the total transaction count because as seen from the scatter plot, there is almost no churn for the customers that have done atleast 90 total transactions. An example of a scenario could be that the bank can predict that if a customer have done more than 90 transactions, that customer will keep using the amenities of the bank and will not churn.

Final learnings

I have learned how to use some of the common functions and techniques of the data science exploratory data analaysis. I now have a subtle understanding of how to approach this types of dataset.

Recommendations

I recommend to the future users of this dataset to correlate the variables that have not been used yet to get a better understanding of the correlation between variables and if they will affect the churn rate of the customers.

References

https://towardsdatascience.com/practical-data-analysis-with-pandas-and-seaborn-8fec3cb9cd16 (https://towardsdatascience.com/practical-data-analysis-with-pandas-and-seaborn-8fec3cb9cd16)

https://www.kaggle.com/amanpatyal/exploratory-analysis-bankchurners-csv (https://www.kaggle.com/amanpatyal/exploratory-analysis-bankchurners-csv)

https://www.youtube.com/watch?v=LIGLh7H6lFk&ab_channel=JPTomas (https://www.youtube.com/watch?v=LIGLh7H6lFk&ab_channel=JPTomas)

https://www.youtube.com/watch?v=eSA3v5wmRJo&ab_channel=JPTomas (https://www.youtube.com/watch?v=eSA3v5wmRJo&ab_channel=JPTomas)

https://stackabuse.com/linear-regression-in-python-with-scikit-learn/ (https://stackabuse.com/linear-regression-in-python-with-scikit-learn/)

In [ ]: