Lead Scoring Case Study

1: Importing libraries and suppressing warnings

In [ ]:
```
# importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
```

In [ ]:
```
# supressing warnings
import warnings
warnings.filterwarnings("ignore")
```

2: Importing and inspecting the data

NOTE:

1. Make sure you have this python notebook, the data set (Leads.csv), and the "Leads Data Dictionary.xlsx" file in the same folder while running the cells from this point onwards.

2. Inferences are mentioned in **BLUE**

3. Important notes are mentioned in **RED**

4. Insights are mentioned in **GREEN**

In [ ]:
```
#Setting up the display environment
pd.set_option("display.max_columns",None)
pd.set_option("display.max_rows",None)
pd.set_option('display.width',None)
```

In [ ]:
```
# Reading the dataset
df_leads = pd.read_csv("Leads.csv")
```

In [ ]:
```
df_leads.head()
```

Out[ ]:

	Prospect ID	Lead Number	Lead Origin	Lead Source	Do Not Email	Do Not Call	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Last Activity	Country	Specialization	How did you hear about X Education	o
0	7927b2df-8bba-4d29-b9a2-b6e0beafe620	660737	API	Olark Chat	No	No	0	0.0	0	0.0	Page Visited on Website	NaN	Select	Select	Ur
1	2a272436-5132-4136-86fa-dcc88c88f482	660728	API	Organic Search	No	No	0	5.0	674	2.5	Email Opened	India	Select	Select	Ur
2	8cc8c611-a219-4f35-ad23-fdfd2656bd8a	660727	Landing Page Submission	Direct Traffic	No	No	1	2.0	1532	2.0	Email Opened	India	Business Administration	Select	
3	0cc2df48-7cf4-4e39-9de9-19797f9b38cc	660719	Landing Page Submission	Direct Traffic	No	No	0	1.0	305	1.0	Unreachable	India	Media and Advertising	Word Of Mouth	Ur
4	3256f628-e534-4826-9d63-4a8b88782852	660681	Landing Page Submission	Google	No	No	1	2.0	1428	1.0	Converted to Lead	India	Select	Other	Ur

In [ ]:
```
df_leads.shape
```

Out[ ]:
```
(9240, 37)
```

In [ ]:
```
# Getting an overview of all numerical columns
df_leads.describe()
```

Out[ ]:

	Lead Number	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Asymmetrique Activity Score	Asymmetrique Profile Score
count	9240.000000	9240.000000	9103.000000	9240.000000	9103.000000	5022.000000	5022.000000
mean	617188.435606	0.385390	3.445238	487.698268	2.362820	14.306252	16.344883
std	23405.995698	0.486714	4.854853	548.021466	2.161418	1.386694	1.811395
min	579533.000000	0.000000	0.000000	0.000000	0.000000	7.000000	11.000000
25%	596484.500000	0.000000	1.000000	12.000000	1.000000	14.000000	15.000000
50%	615479.000000	0.000000	3.000000	248.000000	2.000000	14.000000	16.000000
75%	637387.250000	1.000000	5.000000	936.000000	3.000000	15.000000	18.000000
max	660737.000000	1.000000	251.000000	2272.000000	55.000000	18.000000	20.000000

In [ ]:

```
# checking number of unique values in each column
df_leads.nunique()
```

Out[ ]:
```
Prospect ID                                      9240
Lead Number                                      9240
Lead Origin                                         5
Lead Source                                        21
Do Not Email                                        2
Do Not Call                                         2
Converted                                           2
TotalVisits                                        41
Total Time Spent on Website                      1731
Page Views Per Visit                              114
Last Activity                                      17
Country                                            38
Specialization                                     19
How did you hear about X Education                 10
What is your current occupation                     6
What matters most to you in choosing a course       3
Search                                              2
Magazine                                            1
Newspaper Article                                   2
X Education Forums                                  2
Newspaper                                           2
Digital Advertisement                               2
Through Recommendations                             2
Receive More Updates About Our Courses              1
Tags                                               26
Lead Quality                                        5
Update me on Supply Chain Content                   1
Get updates on DM Content                           1
Lead Profile                                        6
City                                                7
Asymmetrique Activity Index                         3
Asymmetrique Profile Index                          3
Asymmetrique Activity Score                        12
Asymmetrique Profile Score                         10
I agree to pay the amount through cheque            1
A free copy of Mastering The Interview              2
Last Notable Activity                              16
dtype: int64
```

In [ ]:
```
# Checking for null values
df_leads.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9240 entries, 0 to 9239
Data columns (total 37 columns):
 #   Column                                         Non-Null Count  Dtype  
---  ------                                         --------------  -----  
 0   Prospect ID                                    9240 non-null   object 
 1   Lead Number                                    9240 non-null   int64  
 2   Lead Origin                                    9240 non-null   object 
 3   Lead Source                                    9204 non-null   object 
 4   Do Not Email                                   9240 non-null   object 
 5   Do Not Call                                    9240 non-null   object 
 6   Converted                                      9240 non-null   int64  
 7   TotalVisits                                    9103 non-null   float64
 8   Total Time Spent on Website                    9240 non-null   int64  
 9   Page Views Per Visit                           9103 non-null   float64
 10  Last Activity                                  9137 non-null   object 
 11  Country                                        6779 non-null   object 
 12  Specialization                                 7802 non-null   object 
 13  How did you hear about X Education             7033 non-null   object 
 14  What is your current occupation                6550 non-null   object 
 15  What matters most to you in choosing a course  6531 non-null   object 
 16  Search                                         9240 non-null   object 
 17  Magazine                                       9240 non-null   object 
 18  Newspaper Article                              9240 non-null   object 
 19  X Education Forums                             9240 non-null   object 
 20  Newspaper                                      9240 non-null   object 
 21  Digital Advertisement                          9240 non-null   object 
 22  Through Recommendations                        9240 non-null   object 
 23  Receive More Updates About Our Courses         9240 non-null   object 
 24  Tags                                           5887 non-null   object 
 25  Lead Quality                                   4473 non-null   object 
 26  Update me on Supply Chain Content              9240 non-null   object 
 27  Get updates on DM Content                      9240 non-null   object 
 28  Lead Profile                                   6531 non-null   object 
 29  City                                           7820 non-null   object 
 30  Asymmetrique Activity Index                    5022 non-null   object 
 31  Asymmetrique Profile Index                     5022 non-null   object 
 32  Asymmetrique Activity Score                    5022 non-null   float64
 33  Asymmetrique Profile Score                     5022 non-null   float64
 34  I agree to pay the amount through cheque       9240 non-null   object 
 35  A free copy of Mastering The Interview         9240 non-null   object 
 36  Last Notable Activity                          9240 non-null   object 
dtypes: float64(4), int64(3), object(30)
memory usage: 2.6+ MB
```

Inference: There are some NULL values in this dataframe.

In [ ]:
```
#Checking for count of missing values in each column
df_leads.isnull().sum()
```

Out[ ]:
```
Prospect ID                                       0
Lead Number                                       0
Lead Origin                                       0
Lead Source                                      36
Do Not Email                                      0
Do Not Call                                       0
Converted                                         0
TotalVisits                                     137
Total Time Spent on Website                       0
Page Views Per Visit                            137
Last Activity                                   103
Country                                        2461
Specialization                                 1438
How did you hear about X Education             2207
What is your current occupation                2690
What matters most to you in choosing a course  2709
Search                                            0
Magazine                                          0
Newspaper Article                                 0
X Education Forums                                0
Newspaper                                         0
Digital Advertisement                             0
Through Recommendations                           0
Receive More Updates About Our Courses            0
Tags                                           3353
Lead Quality                                   4767
Update me on Supply Chain Content                 0
Get updates on DM Content                         0
Lead Profile                                   2709
City                                           1420
Asymmetrique Activity Index                    4218
Asymmetrique Profile Index                     4218
Asymmetrique Activity Score                    4218
Asymmetrique Profile Score                     4218
I agree to pay the amount through cheque          0
A free copy of Mastering The Interview            0
Last Notable Activity                             0
dtype: int64
```

Inference: Some columns have a high number of missing/null values in the dataframe. We shall address them in the next step, i.e., Data Cleaning.

In [ ]:
```
# Checking for duplicate rows
print(df_leads.duplicated().sum())
```

```
0
```

Inference: There are no duplicates in the dataframe df_leads.

3: Data Cleaning

3.1 Handling NULL and "Select" values

It's mentioned in the problem statement that many categorical variables have an option called "Select". This means that the customer did not select any option for those variables, and the default value remains "Select". Hence, such values should be treated as "NULL" values.

In [ ]:
```
# Listing columns having 'Select' as an option

cols_with_select_value = [col for col in df_leads.columns if len(df_leads[col].isin(['Select']).unique())>1]
print(cols_with_select_value)
```

```
['Specialization', 'How did you hear about X Education', 'Lead Profile', 'City']
```

In [ ]:
```
# Converting 'Select' values to NaN.
df_leads = df_leads.replace('Select', np.nan)
```

In [ ]:
```
# Checking if all 'Select' values have been handled in the columns
cols_with_select_value = [col for col in df_leads.columns if len(df_leads[col].isin(['Select']).unique())>1]
print(cols_with_select_value)
```

```
[]
```

Inference: Now, there are no "Select" values in the dataframe df_leads.

3.2 Handling Missing Values

In [ ]:
```
# Calculating Missing Values Percentage
100*(df_leads.isna().mean()).sort_values(ascending=False)
```

Out[ ]:
```
How did you hear about X Education              78.463203
Lead Profile                                    74.188312
Lead Quality                                    51.590909
Asymmetrique Profile Score                      45.649351
Asymmetrique Activity Score                     45.649351
Asymmetrique Activity Index                     45.649351
Asymmetrique Profile Index                      45.649351
City                                            39.707792
Specialization                                  36.580087
Tags                                            36.287879
What matters most to you in choosing a course   29.318182
What is your current occupation                 29.112554
Country                                         26.634199
Page Views Per Visit                             1.482684
TotalVisits                                      1.482684
Last Activity                                    1.114719
Lead Source                                      0.389610
Receive More Updates About Our Courses           0.000000
I agree to pay the amount through cheque         0.000000
Get updates on DM Content                        0.000000
Update me on Supply Chain Content                0.000000
A free copy of Mastering The Interview           0.000000
Prospect ID                                      0.000000
Newspaper Article                                0.000000
Through Recommendations                          0.000000
Digital Advertisement                            0.000000
Newspaper                                        0.000000
X Education Forums                               0.000000
Lead Number                                      0.000000
Magazine                                         0.000000
Search                                           0.000000
Total Time Spent on Website                      0.000000
Converted                                        0.000000
Do Not Call                                      0.000000
Do Not Email                                     0.000000
Lead Origin                                      0.000000
Last Notable Activity                            0.000000
dtype: float64
```

Dropping Columns with more than 40% Null Values

In [ ]:
```
# user defined function to drop columns and print its shape before and after dropping
```

```
def dropNullColumns(data ,percentage=40):

    missing_perc = 100*(data.isna().mean()).sort_values(ascending=False)
    col_to_drop = missing_perc[missing_perc>=percentage].index.to_list()
    print("Total columns dropped: ",len(col_to_drop),"\n")
    print("List of columns dropped : " , col_to_drop,"\n")
    print("Shape before dropping columns: ",data.shape)

    data.drop(labels=col_to_drop,axis=1, inplace=True)

    print("Shape after dropping columns: ",data.shape)
```

In [ ]:
```
# dropping columns using the User-defined function
dropNullColumns(df_leads)
```

```
Total columns dropped:  7

List of columns dropped :  ['How did you hear about X Education', 'Lead Profile', 'Lead Quality', 'Asymmetrique Profile Score',
'Asymmetrique Activity Score', 'Asymmetrique Activity Index', 'Asymmetrique Profile Index']

Shape before dropping columns:  (9240, 37)
Shape after dropping columns:  (9240, 30)
```

In [ ]:
```
# Checking the percentage of null values in the remaining columns

100*(df_leads.isna().mean()).sort_values(ascending=False)
```

Out[ ]:
```
City                                            39.707792
Specialization                                  36.580087
Tags                                            36.287879
What matters most to you in choosing a course   29.318182
What is your current occupation                 29.112554
Country                                         26.634199
Page Views Per Visit                             1.482684
TotalVisits                                      1.482684
Last Activity                                    1.114719
Lead Source                                      0.389610
Through Recommendations                          0.000000
Receive More Updates About Our Courses           0.000000
Prospect ID                                      0.000000
Newspaper                                        0.000000
Update me on Supply Chain Content                0.000000
Get updates on DM Content                        0.000000
I agree to pay the amount through cheque         0.000000
A free copy of Mastering The Interview           0.000000
Digital Advertisement                            0.000000
Search                                           0.000000
X Education Forums                               0.000000
Newspaper Article                                0.000000
Magazine                                         0.000000
Lead Number                                      0.000000
Total Time Spent on Website                      0.000000
Converted                                        0.000000
Do Not Call                                      0.000000
Do Not Email                                     0.000000
Lead Origin                                      0.000000
Last Notable Activity                            0.000000
dtype: float64
```

3.3 Analyzing columns with Categorical Data

In [ ]:
```
# Selecting the columns with non-numeric data type
categorical_cols = df_leads.select_dtypes(include=['category', 'object']).columns.tolist()

# Printing the selected columns
print(categorical_cols)
```

```
['Prospect ID', 'Lead Origin', 'Lead Source', 'Do Not Email', 'Do Not Call', 'Last Activity', 'Country', 'Specialization', 'What
is your current occupation', 'What matters most to you in choosing a course', 'Search', 'Magazine', 'Newspaper Article', 'X Educ
ation Forums', 'Newspaper', 'Digital Advertisement', 'Through Recommendations', 'Receive More Updates About Our Courses', 'Tag
s', 'Update me on Supply Chain Content', 'Get updates on DM Content', 'City', 'I agree to pay the amount through cheque', 'A fre
e copy of Mastering The Interview', 'Last Notable Activity']
```

Inference: The best approach is to check the percentage of values in each categorical variable and treat the missing values individually for each of them.

In [ ]:
```
columnsList= ["City","Specialization","Tags",'What matters most to you in choosing a course',
              'What is your current occupation','Country','Last Activity','Lead Source']

for i in columnsList:
        perc=100*df_leads[i].value_counts(normalize=True)
        print("value_counts % for :",i,"\n")
        print(perc,"\n")
        print("___"*40,"\n")
```

value_counts % for : City

```
Mumbai                          57.835218
Thane & Outskirts               13.498474
Other Cities                    12.313768
Other Cities of Maharashtra      8.203195
Other Metro Cities               6.821038
Tier II Cities                   1.328307
Name: City, dtype: float64
```

---

value_counts % for : Specialization

```
Finance Management                   16.655290
Human Resource Management            14.470990
Marketing Management                 14.300341
Operations Management                 8.583618
Business Administration               6.877133
IT Projects Management                6.245734
Supply Chain Management               5.955631
Banking, Investment And Insurance     5.767918
Travel and Tourism                    3.464164
Media and Advertising                 3.464164
International Business                3.037543
Healthcare Management                 2.713311
Hospitality Management                1.945392
E-COMMERCE                            1.911263
Retail Management                     1.706485
Rural and Agribusiness                1.245734
E-Business                            0.972696
Services Excellence                   0.682594
Name: Specialization, dtype: float64
```

---

value_counts % for : Tags

```
Will revert after reading the email                35.196195
Ringing                                            20.434856
Interested in other courses                         8.714116
Already a student                                   7.898760
Closed by Horizzon                                  6.081196
switched off                                        4.076779
```

```
Busy                                                3.159504
Lost to EINS                                        2.972652
Not doing further education                         2.463054
Interested  in full time MBA                        1.987430
Graduation in progress                              1.885510
invalid number                                      1.409886
Diploma holder (Not Eligible)                       1.070155
wrong number given                                  0.798369
opp hangup                                          0.560557
number not provided                                 0.458638
in touch with EINS                                  0.203839
Lost to Others                                      0.118906
Still Thinking                                      0.101919
Want to take admission but has financial problems   0.101919
In confusion whether part time or DLP               0.084933
Interested in Next batch                            0.084933
Lateral student                                     0.050960
Shall take in the next coming month                 0.033973
University not recognized                           0.033973
Recognition issue (DEC approval)                    0.016987
Name: Tags, dtype: float64
```

```
value_counts % for : What matters most to you in choosing a course

Better Career Prospects      99.954065
Flexibility & Convenience     0.030623
Other                         0.015312
Name: What matters most to you in choosing a course, dtype: float64
```

```
value_counts % for : What is your current occupation

Unemployed              85.496183
Working Professional    10.778626
Student                  3.206107
Other                    0.244275
Housewife                0.152672
Businessman              0.122137
Name: What is your current occupation, dtype: float64
```

```
value_counts % for : Country

India                   95.766337
United States            1.017849
United Arab Emirates     0.781826
Singapore                0.354035
Saudi Arabia             0.309780
United Kingdom           0.221272
Australia                0.191769
Qatar                    0.147514
Hong Kong                0.103260
Bahrain                  0.103260
Oman                     0.088509
France                   0.088509
unknown                  0.073757
South Africa             0.059006
Nigeria                  0.059006
Germany                  0.059006
Kuwait                   0.059006
Canada                   0.059006
Sweden                   0.044254
China                    0.029503
Asia/Pacific Region      0.029503
Uganda                   0.029503
Bangladesh               0.029503
Italy                    0.029503
Belgium                  0.029503
Netherlands              0.029503
Ghana                    0.029503
Philippines              0.029503
Russia                   0.014751
Switzerland              0.014751
Vietnam                  0.014751
Denmark                  0.014751
Tanzania                 0.014751
Liberia                  0.014751
Malaysia                 0.014751
Kenya                    0.014751
Sri Lanka                0.014751
Indonesia                0.014751
Name: Country, dtype: float64
```

```
value_counts % for : Last Activity

Email Opened                    37.616285
SMS Sent                        30.042684
Olark Chat Conversation         10.649010
Page Visited on Website          7.004487
Converted to Lead                4.684251
Email Bounced                    3.567911
Email Link Clicked               2.922185
Form Submitted on Website        1.269563
Unreachable                      1.017840
Unsubscribed                     0.667615
Had a Phone Conversation         0.328335
Approached upfront               0.098501
View in browser link Clicked     0.065667
Email Received                   0.021889
Email Marked Spam                0.021889
Visited Booth in Tradeshow       0.010945
Resubscribed to emails           0.010945
Name: Last Activity, dtype: float64
```

```
value_counts % for : Lead Source

Google               31.160365
Direct Traffic       27.629292
Olark Chat           19.067797
Organic Search       12.538027
Reference             5.801825
Welingak Website      1.542807
Referral Sites        1.358105
Facebook              0.597566
bing                  0.065189
google                0.054324
Click2call            0.043459
Press_Release         0.021730
Social Media          0.021730
Live Chat             0.021730
youtubechannel        0.010865
testone               0.010865
Pay per Click Ads     0.010865
welearnblog_Home      0.010865
```

```
WeLearn               0.010865
blog                  0.010865
NC_EDM                0.010865
Name: Lead Source, dtype: float64
```

Insights:

- **City:** City has 39.71 % missing values. Imputing these values with Mumbai will skew the data and cause bias in the model. Hence, this column can be dropped.
- **Specialization:** Specialization has 36.58 % missing values and are evenly distributed. Hence, it cannot be imputed or dropped. We need to create an additional category called "Others".
- **Tags:** Tags has 36.29 % missing values. Tags are assigned to customers indicating their current status. Since this is their current status, this column won't be useful for modeling, and can be dropped.
- **What matters most to you in choosing a course:** This variable has 29.32 % missing values. 99.95% customers have selected 'better career prospects'. This data is massively skewed and won't provide any insight.
- **What is your current occupation:** We can impute the missing values with `'Unemployed'` as it has the most values. This seems to be a important variable from business context, since X Education sells online courses and unemployed people might take this course to increase their chances of getting employed.
- **Country:** Since 96% of the customers are from India, it does not make sense to impute missing values with India. Hence, this column can be dropped.
- **Last Activity:** "Email Opened" has the highest number of values and the overall missing values in this column is only 1.11%. Hence, we'll impute the missing values with label 'Email Opened'.
- **Lead Source:** "Google" has the highest number of occurences and overall nulls in this column is just 0.39%. Hence, we will impute the missing values with label 'Google'

Columns to be dropped

- 'City'
- 'Tags'
- 'What matters most to you in choosing a course'
- 'Country'

In [ ]:
```
# Dropping the Columns
print("Before Drop",df_leads.shape)
df_leads.drop(['City','Tags','Country','What matters most to you in choosing a course'],axis=1,inplace=True)
print("After Drop",df_leads.shape)
```

```
Before Drop (9240, 30)
After Drop (9240, 26)
```

Columns to be imputed

- 'Specialization'
- 'Lead Source'
- 'Last Activity'
- 'What is your current occupation'

In [ ]:
```
# Imputing values as per the above insights

missing_values={'Specialization':'Others','Lead Source':'Google','Last Activity':'Email Opened',
                'What is your current occupation':'Unemployed'}
df_leads=df_leads.fillna(value=missing_values)
```

In [ ]:
```
# Re Checking the percentage of null values in remaining columns

round(((df_leads.isnull().sum()/df_leads.shape[0])*100),2).sort_values(ascending=False)
```

Out[ ]:
```
TotalVisits                                    1.48
Page Views Per Visit                           1.48
Prospect ID                                    0.00
Magazine                                       0.00
A free copy of Mastering The Interview         0.00
I agree to pay the amount through cheque       0.00
Get updates on DM Content                      0.00
Update me on Supply Chain Content              0.00
Receive More Updates About Our Courses         0.00
Through Recommendations                        0.00
Digital Advertisement                          0.00
Newspaper                                      0.00
X Education Forums                             0.00
Newspaper Article                              0.00
Search                                         0.00
Lead Number                                    0.00
What is your current occupation                0.00
Specialization                                 0.00
Last Activity                                  0.00
Total Time Spent on Website                    0.00
Converted                                      0.00
Do Not Call                                    0.00
Do Not Email                                   0.00
Lead Source                                    0.00
Lead Origin                                    0.00
Last Notable Activity                          0.00
dtype: float64
```

3.4 Columns with Numerical Data

In [ ]:
```
# TotalVisits
print("TotalVisits - Value Counts")
print("----------------------------------------")
df_leads.TotalVisits.value_counts().head(10)
```

```
TotalVisits - Value Counts
----------------------------------------
```

Out[ ]:
```
0.0    2189
2.0    1680
3.0    1306
4.0    1120
5.0     783
6.0     466
1.0     395
7.0     309
8.0     224
9.0     164
Name: TotalVisits, dtype: int64
```

Inference: Missing values in 'TotalVisits' can be imputed with "mode".

In [ ]:
```
# TotalVisits missing values to be imputed with mode
df_leads['TotalVisits'].fillna(df_leads['TotalVisits'].mode()[0], inplace=True)
```

In [ ]:
```
# Page Views Per Visit
print("Page Views Per Visit - Value Counts")
print("---------------------------------------")
df_leads.TotalVisits.value_counts().head(10)
```

```
Page Views Per Visit - Value Counts
---------------------------------------
```

Out[ ]:
```
0.0    2326
2.0    1680
3.0    1306
4.0    1120
5.0     783
6.0     466
1.0     395
7.0     309
8.0     224
9.0     164
Name: TotalVisits, dtype: int64
```

Inference: Missing values in 'Page Views Per Visit' can be imputed with "mode".

In [ ]:
```
# Page Views Per Visit missing values to be imputed with mode

df_leads['Page Views Per Visit'].fillna(df_leads['Page Views Per Visit'].mode()[0], inplace=True)
```

Re-checking the null values for columns

In [ ]:
```
# Re Checking the percentage of null values after handling categorical and numerical columns

round(((df_leads.isnull().sum()/df_leads.shape[0])*100),2).sort_values(ascending=False)
```

Out[ ]:
```
Prospect ID                                    0.0
Lead Number                                    0.0
A free copy of Mastering The Interview         0.0
I agree to pay the amount through cheque       0.0
Get updates on DM Content                      0.0
Update me on Supply Chain Content              0.0
Receive More Updates About Our Courses         0.0
Through Recommendations                        0.0
Digital Advertisement                          0.0
Newspaper                                      0.0
X Education Forums                             0.0
Newspaper Article                              0.0
Magazine                                       0.0
Search                                         0.0
What is your current occupation                0.0
Specialization                                 0.0
Last Activity                                  0.0
Page Views Per Visit                           0.0
Total Time Spent on Website                    0.0
TotalVisits                                    0.0
Converted                                      0.0
Do Not Call                                    0.0
Do Not Email                                   0.0
Lead Source                                    0.0
Lead Origin                                    0.0
Last Notable Activity                          0.0
dtype: float64
```

3.5 Removing Unwanted Columns

In [ ]:
```
# Last Notable Activity
print("Last Notable Activity")
print("---------------------------------------")
100*df_leads['Last Notable Activity'].value_counts(normalize=True)
```

Out[ ]:
```
Last Notable Activity
--------------------------------------
Modified                        36.872294
Email Opened                    30.595238
SMS Sent                        23.506494
Page Visited on Website          3.441558
Olark Chat Conversation          1.980519
Email Link Clicked               1.872294
Email Bounced                    0.649351
Unsubscribed                     0.508658
Unreachable                      0.346320
Had a Phone Conversation         0.151515
Email Marked Spam                0.021645
Approached upfront               0.010823
Resubscribed to emails           0.010823
View in browser link Clicked     0.010823
Form Submitted on Website        0.010823
Email Received                   0.010823
Name: Last Notable Activity, dtype: float64
```

Handling columns with only one unique value

In [ ]:
```
#checking for columns with one unique value

df_leads.describe(include = 'object')
```

Out[ ]:

	Prospect ID	Lead Origin	Lead Source	Do Not Email	Do Not Call	Last Activity	Specialization	What is your current occupation	Search	Magazine	Newspaper Article	X Education Forums	Newspaper	Ad
count	9240	9240	9240	9240	9240	9240	9240	9240	9240	9240	9240	9240	9240	
unique	9240	5	21	2	2	17	19	6	2	1	2	2	2	
top	7927b2df-8bba-4d29-b9a2-b6e0beafe620	Landing Page Submission	Google	No	No	Email Opened	Others	Unemployed	No	No	No	No	No	
freq	1	4886	2904	8506	9238	3540	3380	8290	9226	9240	9238	9239	9239	

Inference: Following columns have only "one unique value":

- 'I agree to pay the amount through cheque',
- 'Get updates on DM Content',
- 'Update me on Supply Chain Content',
- 'Receive More Updates About Our Courses',
- 'Magazine'

These columns are of no use as they have only one category of response from customers and can be dropped.

In [ ]:
```
# List and dropping columns with one unique value whose count and frequency are same
cols_to_drop = ['Magazine','Receive More Updates About Our Courses',
                    'Update me on Supply Chain Content',
                    'Get updates on DM Content',
                    'I agree to pay the amount through cheque']

print("Before Dropping Columns",df_leads.shape)
df_leads.drop(cols_to_drop, axis = 1, inplace = True)
print("After Dropping Columns",df_leads.shape)
```

```
Before Dropping Columns (9240, 26)
After Dropping Columns (9240, 21)
```

Dropping columns not used for modeling

NOTE: The following columns do not add much value to the model. Hence we shall drop them to reduce the noise in the data:

- 'Prospect ID',
- 'Lead Number',
- 'Last Notable Activity'

In [ ]:
```
# Dropping Columns
print("Before Dropping Columns",df_leads.shape)
df_leads.drop(['Prospect ID','Lead Number','Last Notable Activity'],axis=1,inplace=True)
print("After Dropping Columns",df_leads.shape)
```

```
Before Dropping Columns (9240, 21)
After Dropping Columns (9240, 18)
```

In [ ]:
```
# getting the percentage of missing values in each row, output is listed in descending order
100*(df_leads.isna().mean(axis=1)).sort_values(ascending=False).head(10)
```

Out[ ]:
```
0       0.0
6064    0.0
6156    0.0
6157    0.0
6158    0.0
6159    0.0
6160    0.0
6161    0.0
6162    0.0
6163    0.0
dtype: float64
```

Inference: There are no missing values in rows

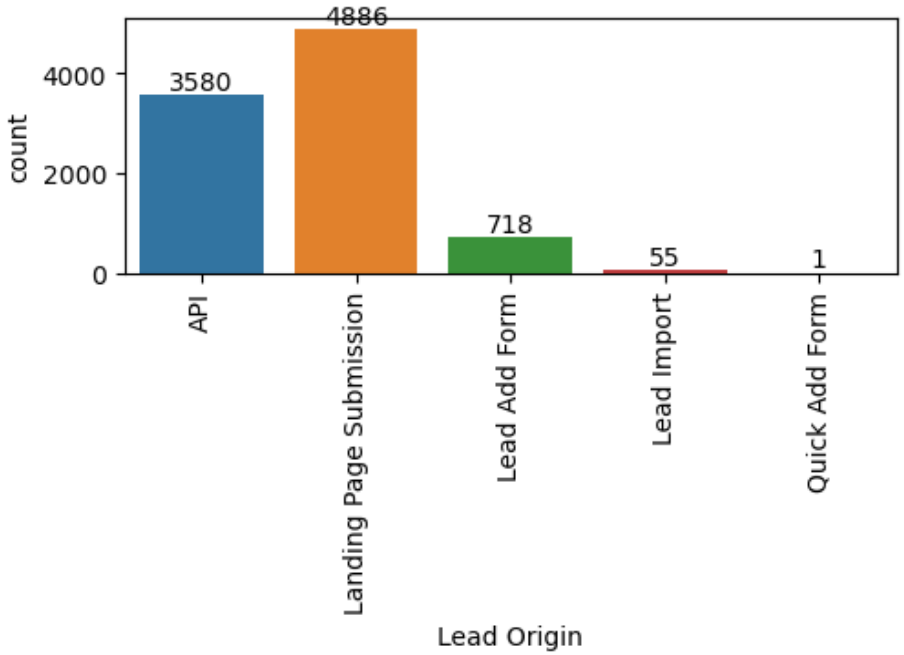
3.6 Checking & Dropping Category Columns that are Skewed

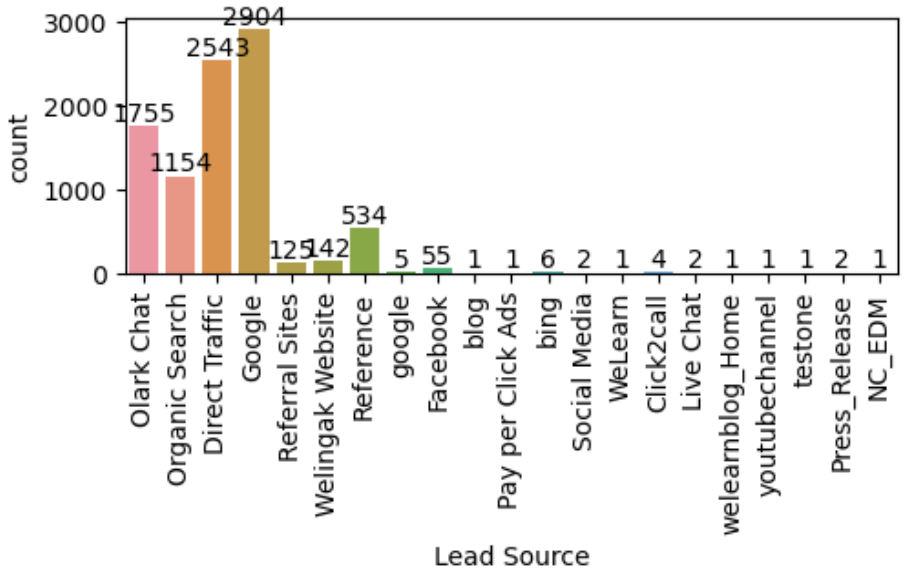
NOTE: This is similar to the handling of unique values in numeric columns.

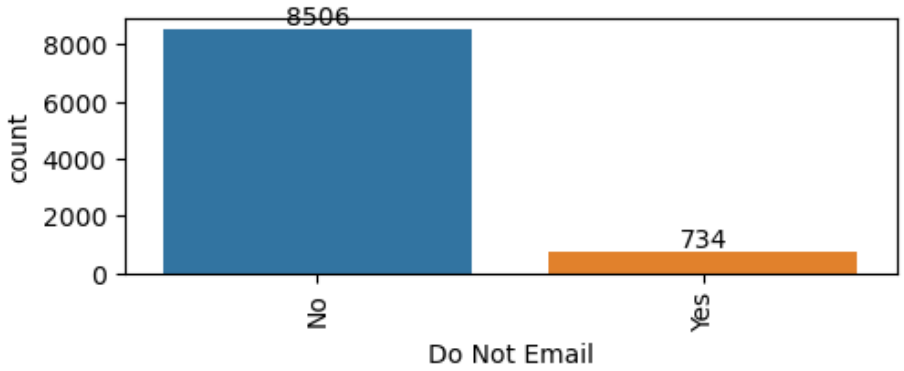
Checking skewness in categorical columns

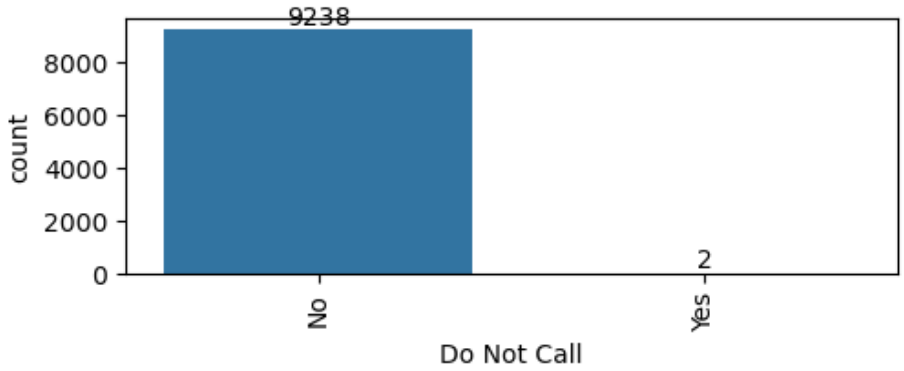
In [ ]:

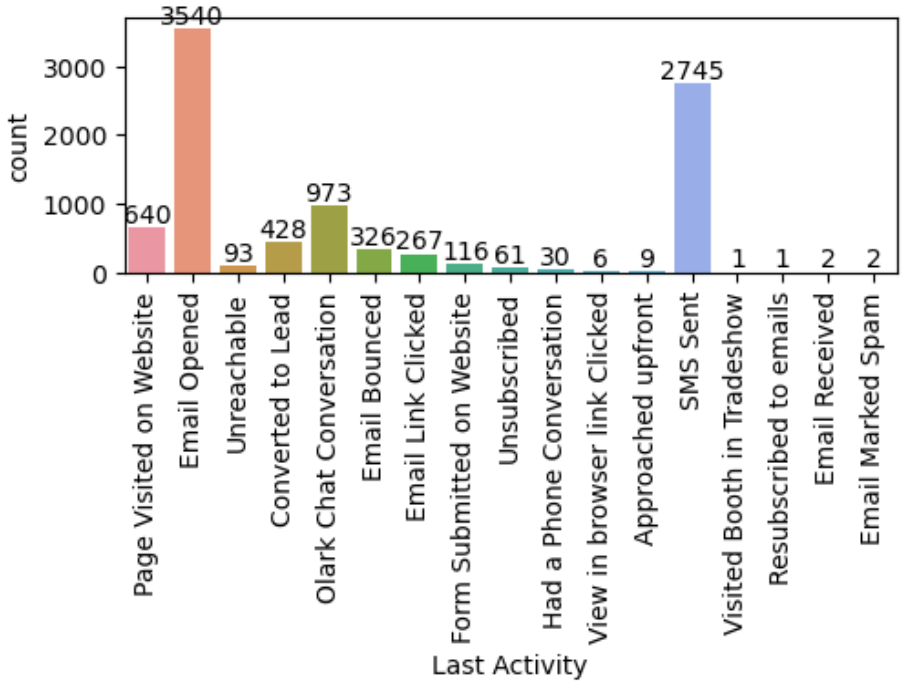
```
# plotting countplot for object dtype and histogram for number to get data distribution
categorical_col = df_leads.select_dtypes(include=['category', 'object']).columns.tolist()
plt.figure(figsize=(12,40))

plt.subplots_adjust(wspace=.2,hspace=2)
for i in enumerate(categorical_col):
    plt.subplot(8,2, i[0]+1)
    ax=sns.countplot(x=i[1],data=df_leads)
    plt.xticks(rotation=90)

    for p in ax.patches:
        ax.annotate('{:.0f}'.format(p.get_height()), (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha = 'center', va = 'center', xytext = (0, 5), textcoords = 'offset points')


plt.show()
```

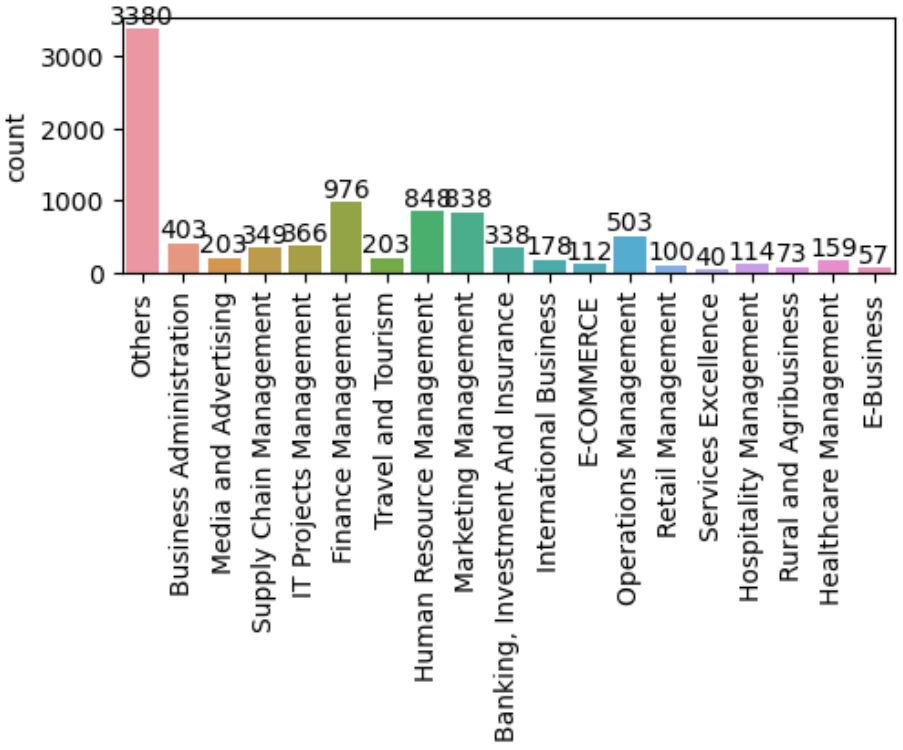

count
4000
2000
0
3580
4886
718
55
1
API
Landing Page Submission
Lead Add Form
Lead Import
Quick Add Form
Lead Origin

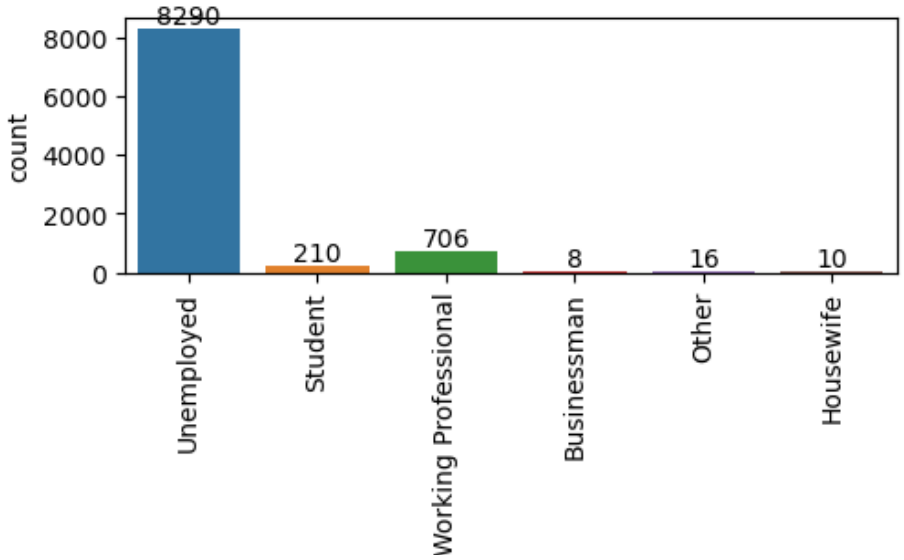

count
3000
2000
1000
0
1755
1154
2543
2904
125
142
534
5
55
1
1
6
2
1
4
2
1
1
1
2
1
Olark Chat
Organic Search
Direct Traffic
Google
Referral Sites
Welingak Website
Reference
google
Facebook
blog
Pay per Click Ads
bing
Social Media
WeLearn
Click2call
Live Chat
welearnblog_Home
youtubechannel
testone
Press_Release
NC_EDM
Lead Source

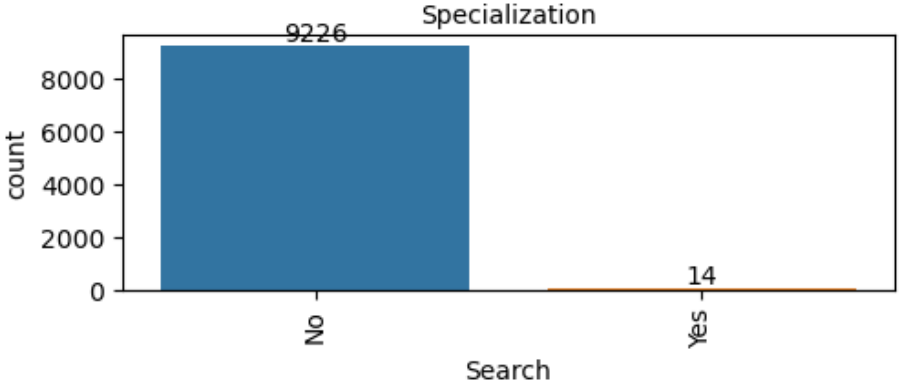

count
8000
6000
4000
2000
0
8506
734
No
Yes
Do Not Email


count
8000
6000
4000
2000
0
9238
2
No
Yes
Do Not Call

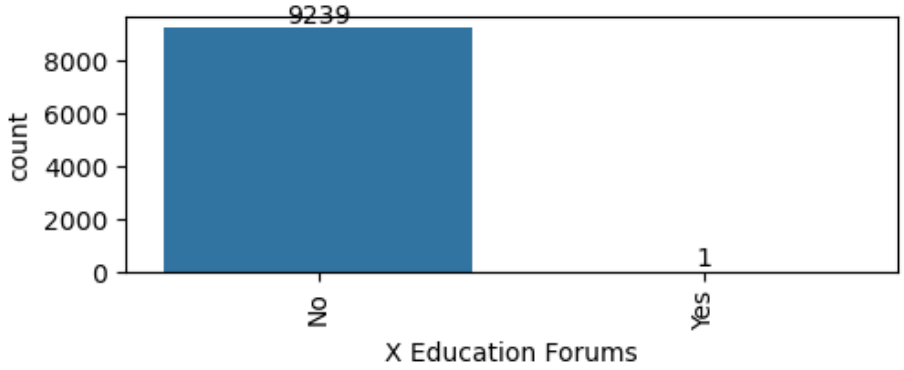

count
Last Activity
Page Visited on Website
Email Opened
Unreachable
Converted to Lead
Olark Chat Conversation
Email Bounced
Email Link Clicked
Form Submitted on Website
Unsubscribed
Had a Phone Conversation
View in browser link Clicked
Approached upfront
SMS Sent
Visited Booth in Tradeshow
Resubscribed to emails
Email Received
Email Marked Spam
640
3540
93
428
973
326
267
116
61
30
6
9
2745
1
1
2
2

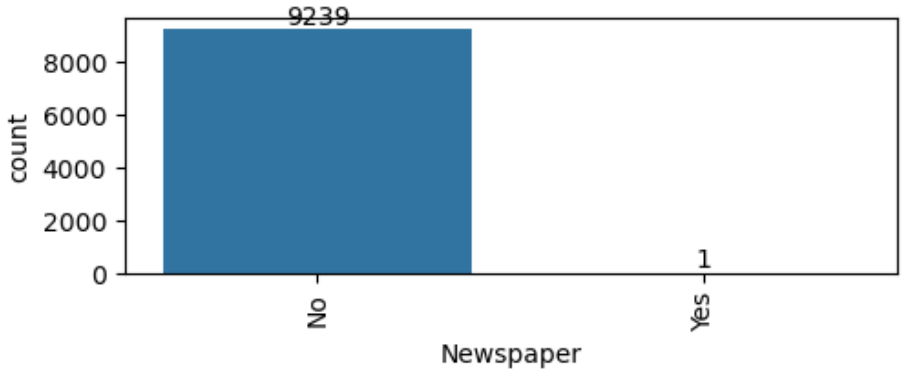

count
Specialization
Others
Business Administration
Media and Advertising
Supply Chain Management
IT Projects Management
Finance Management
Travel and Tourism
Human Resource Management
Marketing Management
Banking, Investment And Insurance
International Business
E-COMMERCE
Operations Management
Retail Management
Services Excellence
Hospitality Management
Rural and Agribusiness
Healthcare Management
E-Business
3380
403
203
349
366
976
203
848
838
338
178
112
503
100
40
114
73
159
57

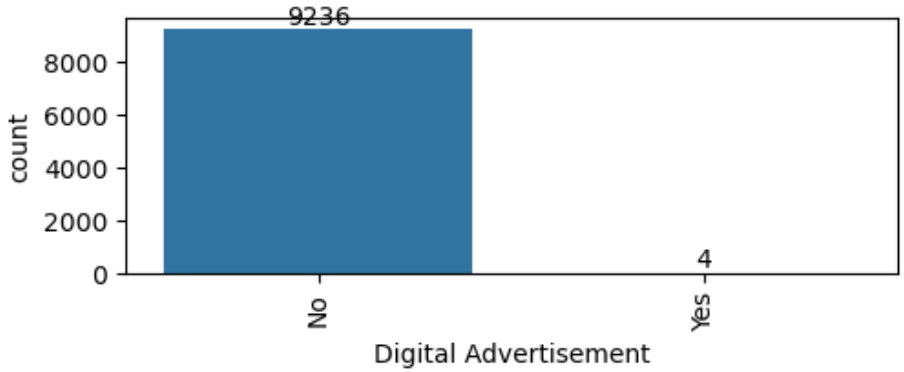

count
Unemployed
Student
Working Professional
Businessman
Other
Housewife
8290
210
706
8
16
10

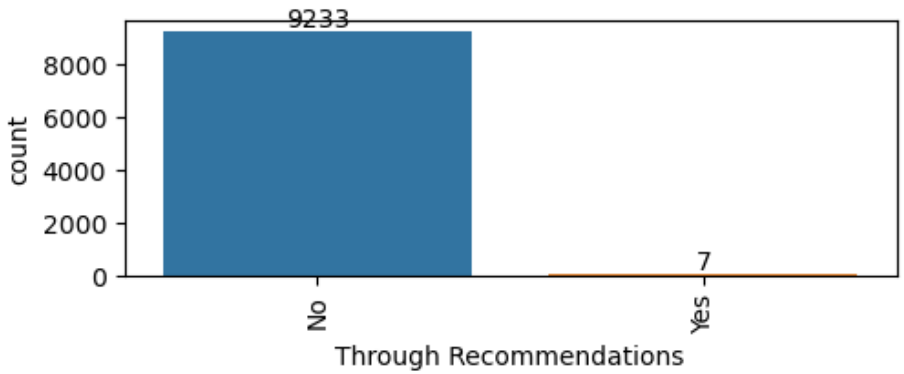

count
Search
No
Yes
9226
14

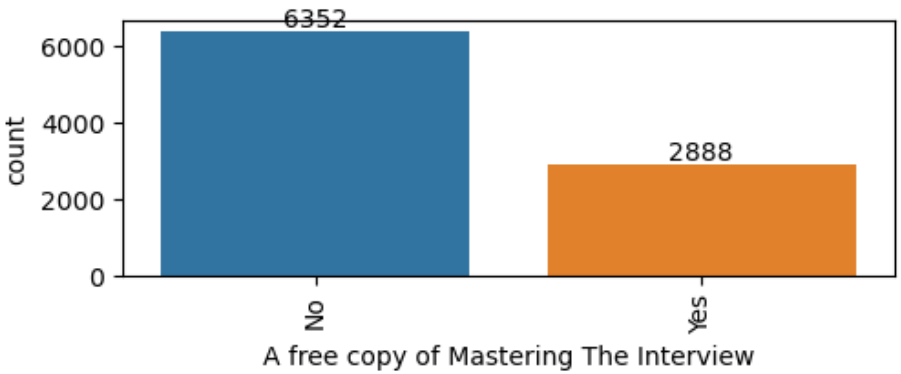
What is your current occupation











Inference: The following columns have highly skewed data and won't add much value to the model. Hence they can be dropped:

- 'Do Not Call'
- 'Search'
- 'Newspaper Article'
- 'X Education Forums'
- 'Newspaper'
- 'Digital Advertisement'
- 'Through Recommendations'

In [ ]:
```
# Dropping categorical columns with highly skewed data

print("Before Drop: ",df_leads.shape)
df_leads.drop(['Do Not Call','Search','Newspaper Article','X Education Forums','Newspaper','Digital Advertisement','Through Recor
print("After Drop: ",df_leads.shape)
```

```
Before Drop:  (9240, 18)
After Drop:  (9240, 11)
```

In [ ]:
```
df_leads.head()
```

Out[ ]:

	Lead Origin	Lead Source	Do Not Email	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Last Activity	Specialization	What is your current occupation	A free copy of Mastering The Interview
0	API	Olark Chat	No	0	0.0	0	0.0	Page Visited on Website	Others	Unemployed	No
1	API	Organic Search	No	0	5.0	674	2.5	Email Opened	Others	Unemployed	No
2	Landing Page Submission	Direct Traffic	No	1	2.0	1532	2.0	Email Opened	Business Administration	Student	Yes
3	Landing Page Submission	Direct Traffic	No	0	1.0	305	1.0	Unreachable	Media and Advertising	Unemployed	No
4	Landing Page Submission	Google	No	1	2.0	1428	1.0	Converted to Lead	Others	Unemployed	No

3.7 Outlier Analysis

For Numerical Columns

In [ ]:
```
# User-defined function (UDF) for checking outliers
def Check_Outliers(data,columnList):

    plt.figure(figsize=[22,11])
    plt.subplots_adjust(wspace=0.4,hspace=0.5)

    for i,j in enumerate(columnList):
        plt.subplot(2,2,i+1)

        sns.boxplot(y=data[j])     # y = df_leads[j] to make plot verticle

        plt.suptitle("\nChecking Outliers using Boxplot",fontsize=20,color="green")
        plt.ylabel(None)
        plt.title(j,fontsize=15,color='brown')
```

In [ ]:
```
# Checking outliers for numerical variables other than target variable using UDF
num_cols = ["TotalVisits","Page Views Per Visit","Total Time Spent on Website"]
Check_Outliers(df_leads,num_cols)
```

Inference: The variables "TotalVisits", and "Page Views Per Visit" contain outliers and need to be treated.

Treating outliers using capping method

In [ ]:
```
# Before capping treatment
df_leads.describe(percentiles=[.10,.25,.50,.75,.95])
```

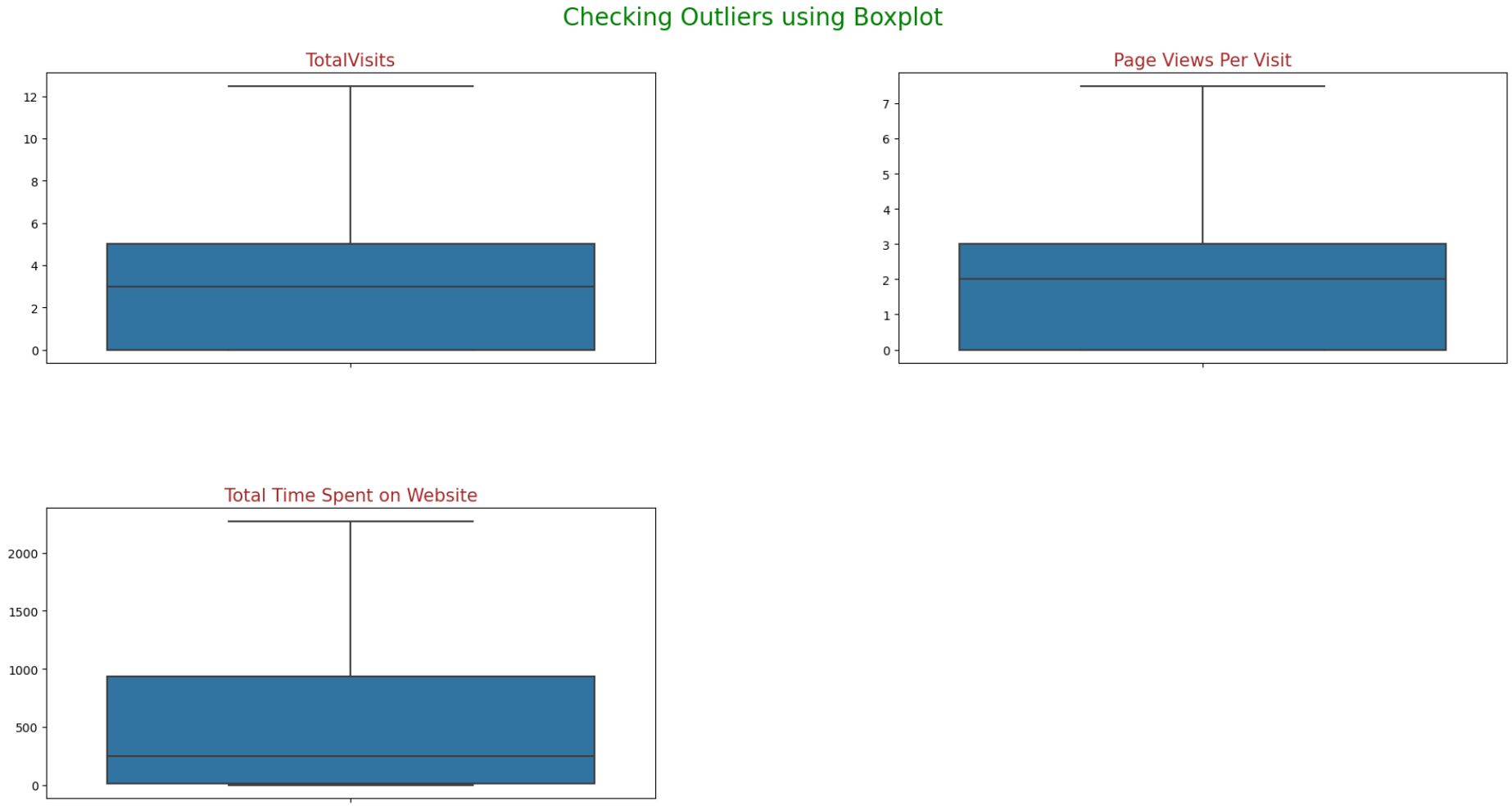
Out[ ]:

	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit
count	9240.000000	9240.000000	9240.000000	9240.000000
mean	0.385390	3.394156	487.698268	2.327787
std	0.486714	4.836682	548.021466	2.164258
min	0.000000	0.000000	0.000000	0.000000
10%	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	12.000000	0.000000
50%	0.000000	3.000000	248.000000	2.000000
75%	1.000000	5.000000	936.000000	3.000000
95%	1.000000	10.000000	1562.000000	6.000000
max	1.000000	251.000000	2272.000000	55.000000

In [ ]:
```
# User-defined function to treat outliers using capping and flooring

def Outlier_treatment(df,columnList):
    for i in columnList:
        q1 = df[i].describe()["25%"]
        q3 = df[i].describe()["75%"]
        IQR = q3 - q1

        upper_bound = q3 + 1.5*IQR
        lower_bound = q1 - 1.5*IQR

        # capping upper_bound
        df[i] = np.where(df[i] > upper_bound, upper_bound,df[i])

        # flooring lower_bound
        df[i] = np.where(df[i] < lower_bound, lower_bound,df[i])
```

In [ ]:
```
# Checking outliers for numerical variables other than target variable
capping_cols = ["TotalVisits","Page Views Per Visit"]

# UDF
Outlier_treatment(df_leads,capping_cols)
```

In [ ]:
```
# Boxplot after capping Treatment

num_cols = ["TotalVisits","Page Views Per Visit","Total Time Spent on Website"]
Check_Outliers(df_leads,num_cols)
```



In [ ]:
```
# Detailed percentlie values after treatment
df_leads.describe(percentiles=[.10,.25,.50,.75,.95])
```

Out[ ]:

	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit
count	9240.000000	9240.000000	9240.000000	9240.000000
mean	0.385390	3.213853	487.698268	2.274987
std	0.486714	3.005136	548.021466	1.917776
min	0.000000	0.000000	0.000000	0.000000
10%	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	12.000000	0.000000
50%	0.000000	3.000000	248.000000	2.000000
75%	1.000000	5.000000	936.000000	3.000000
95%	1.000000	10.000000	1562.000000	6.000000
max	1.000000	12.500000	2272.000000	7.500000

3.8 Fixing Invalid values & Standardising Data in the columns

In [ ]:
```
df_leads.head()
```

Out[ ]:

	Lead Origin	Lead Source	Do Not Email	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Last Activity	Specialization	What is your current occupation	A free copy of Mastering The Interview
0	API	Olark Chat	No	0	0.0	0	0.0	Page Visited on Website	Others	Unemployed	No
1	API	Organic Search	No	0	5.0	674	2.5	Email Opened	Others	Unemployed	No
2	Landing Page Submission	Direct Traffic	No	1	2.0	1532	2.0	Email Opened	Business Administration	Student	Yes
3	Landing Page Submission	Direct Traffic	No	0	1.0	305	1.0	Unreachable	Media and Advertising	Unemployed	No
4	Landing Page Submission	Google	No	1	2.0	1428	1.0	Converted to Lead	Others	Unemployed	No

In [ ]:

```
## Categorical Variables

columnsList_cat = ["Lead Origin","Lead Source","Do Not Email","Last Activity","Specialization",
                   "What is your current occupation","A free copy of Mastering The Interview"]

for i in columnsList_cat:
        perc=100*df_leads[i].value_counts(normalize=True)
        print("value_counts % for :",i,"\n")
        print(perc,"\n")
        print("_^_"*40,"\n")
```

```
value_counts % for : Lead Origin

Landing Page Submission    52.878788
API                        38.744589
Lead Add Form               7.770563
Lead Import                 0.595238
Quick Add Form              0.010823
Name: Lead Origin, dtype: float64

_^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^_

value_counts % for : Lead Source

Google               31.428571
Direct Traffic       27.521645
Olark Chat           18.993506
Organic Search       12.489177
Reference             5.779221
Welingak Website      1.536797
Referral Sites        1.352814
Facebook              0.595238
bing                  0.064935
google                0.054113
Click2call            0.043290
Press_Release         0.021645
Social Media          0.021645
Live Chat             0.021645
youtubechannel        0.010823
testone               0.010823
Pay per Click Ads     0.010823
welearnblog_Home      0.010823
WeLearn               0.010823
blog                  0.010823
NC_EDM                0.010823
Name: Lead Source, dtype: float64

_^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^_

value_counts % for : Do Not Email

No     92.056277
Yes     7.943723
Name: Do Not Email, dtype: float64
```

```
_^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^_

value_counts % for : Last Activity

Email Opened                    38.311688
SMS Sent                        29.707792
Olark Chat Conversation         10.530303
Page Visited on Website          6.926407
Converted to Lead                4.632035
Email Bounced                    3.528139
Email Link Clicked               2.889610
Form Submitted on Website        1.255411
Unreachable                      1.006494
Unsubscribed                     0.660173
Had a Phone Conversation         0.324675
Approached upfront               0.097403
View in browser link Clicked     0.064935
Email Received                   0.021645
Email Marked Spam                0.021645
Visited Booth in Tradeshow       0.010823
Resubscribed to emails           0.010823
Name: Last Activity, dtype: float64

_^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^_

value_counts % for : Specialization

Others                             36.580087
Finance Management                 10.562771
Human Resource Management           9.177489
Marketing Management                9.069264
Operations Management               5.443723
Business Administration             4.361472
IT Projects Management              3.961039
Supply Chain Management             3.777056
Banking, Investment And Insurance   3.658009
Travel and Tourism                  2.196970
Media and Advertising               2.196970
International Business              1.926407
Healthcare Management               1.720779
Hospitality Management              1.233766
E-COMMERCE                          1.212121
Retail Management                   1.082251
Rural and Agribusiness              0.790043
```

```
E-Business                            0.616883
Services Excellence                   0.432900
Name: Specialization, dtype: float64

_^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^_

value_counts % for : What is your current occupation

Unemployed              89.718615
Working Professional     7.640693
Student                  2.272727
Other                    0.173160
Housewife                0.108225
Businessman              0.086580
Name: What is your current occupation, dtype: float64

_^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^_

value_counts % for : A free copy of Mastering The Interview

No     68.744589
Yes    31.255411
Name: A free copy of Mastering The Interview, dtype: float64

_^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^__^_
```

Inference:

- Some of the categories/levels in the "Lead Score" and "Last Activity" columns have very few records. To prevent ambiguent columns when creating dummy variables, we'll group these categories together under "Others", to keep the dataframe as clean as possible.
- "Google" & "google" are same in "Lead Source", so we will standardise the case.

Grouping Low frequency values

In [ ]:
```
# Grouping low frequency value levels to Others
df_leads['Lead Source'] = df_leads['Lead Source'].replace(["bing","Click2call","Press_Release",
                                                           "Social Media","Live Chat","youtubechannel",
                                                           "testone","Pay per Click Ads","welearnblog_Home",
                                                           "WeLearn","blog","NC_EDM"],"Others")
```

```
# Changing google to Google
df_leads['Lead Source'] = df_leads['Lead Source'].replace("google","Google")
```

In [ ]:
```
# value_counts percentage after replace
df_leads["Lead Source"].value_counts(normalize=True)*100
```

Out[ ]:
```
Google              31.482684
Direct Traffic      27.521645
Olark Chat          18.993506
Organic Search      12.489177
Reference            5.779221
Welingak Website     1.536797
Referral Sites       1.352814
Facebook             0.595238
Others               0.248918
Name: Lead Source, dtype: float64
```

In [ ]:
```
# Grouping low frequency value levels to Others
df_leads['Last Activity'] = df_leads['Last Activity'].replace(['Unreachable','Unsubscribed',
                                                               'Had a Phone Conversation',
                                                               'Approached upfront',
                                                               'View in browser link Clicked',
                                                               'Email Marked Spam',
                                                               'Email Received','Visited Booth in Tradeshow',
                                                               'Resubscribed to emails'],'Others')
```

In [ ]:
```
# value_counts percentage after replace
df_leads['Last Activity'].value_counts(normalize=True)*100
```

Out[ ]:
```
Email Opened                 38.311688
SMS Sent                     29.707792
Olark Chat Conversation      10.530303
Page Visited on Website       6.926407
Converted to Lead             4.632035
Email Bounced                 3.528139
Email Link Clicked            2.889610
Others                        2.218615
Form Submitted on Website     1.255411
Name: Last Activity, dtype: float64
```

In [ ]:
```
# Renaming column name to "Free_copy" from "A free copy of Mastering The Interview"
df_leads.rename(columns={'A free copy of Mastering The Interview': 'Free_copy'}, inplace=True)
```

```
# Renaming column name to "Current_occupation" from "What is your current occupation"
df_leads.rename(columns={'What is your current occupation': 'Current_occupation'}, inplace=True)
```

Inference: "Do Not Email" & "Free_copy" are both binary categorical columns, so we'll map yes/no to 1/0.

Mapping Binary categorical variables

In [ ]:
```
# Mapping binary categorical variables (Yes/No to 1/0)
df_leads['Do Not Email'] = df_leads['Do Not Email'].apply(lambda x: 1 if x =='Yes' else 0)

df_leads['Free_copy'] = df_leads['Free_copy'].apply(lambda x: 1 if x =='Yes' else 0)
```

Checking Data-types of variables

In [ ]:
```
df_leads.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9240 entries, 0 to 9239
Data columns (total 11 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Lead Origin                   9240 non-null   object 
 1   Lead Source                   9240 non-null   object 
 2   Do Not Email                  9240 non-null   int64  
 3   Converted                     9240 non-null   int64  
 4   TotalVisits                   9240 non-null   float64
 5   Total Time Spent on Website   9240 non-null   int64  
 6   Page Views Per Visit          9240 non-null   float64
 7   Last Activity                 9240 non-null   object 
 8   Specialization                9240 non-null   object 
 9   Current_occupation            9240 non-null   object 
 10  Free_copy                     9240 non-null   int64  
dtypes: float64(2), int64(4), object(5)
memory usage: 794.2+ KB
```

Inference: All the data types are now suitable to conduct Exploratory Data Analysis (EDA).

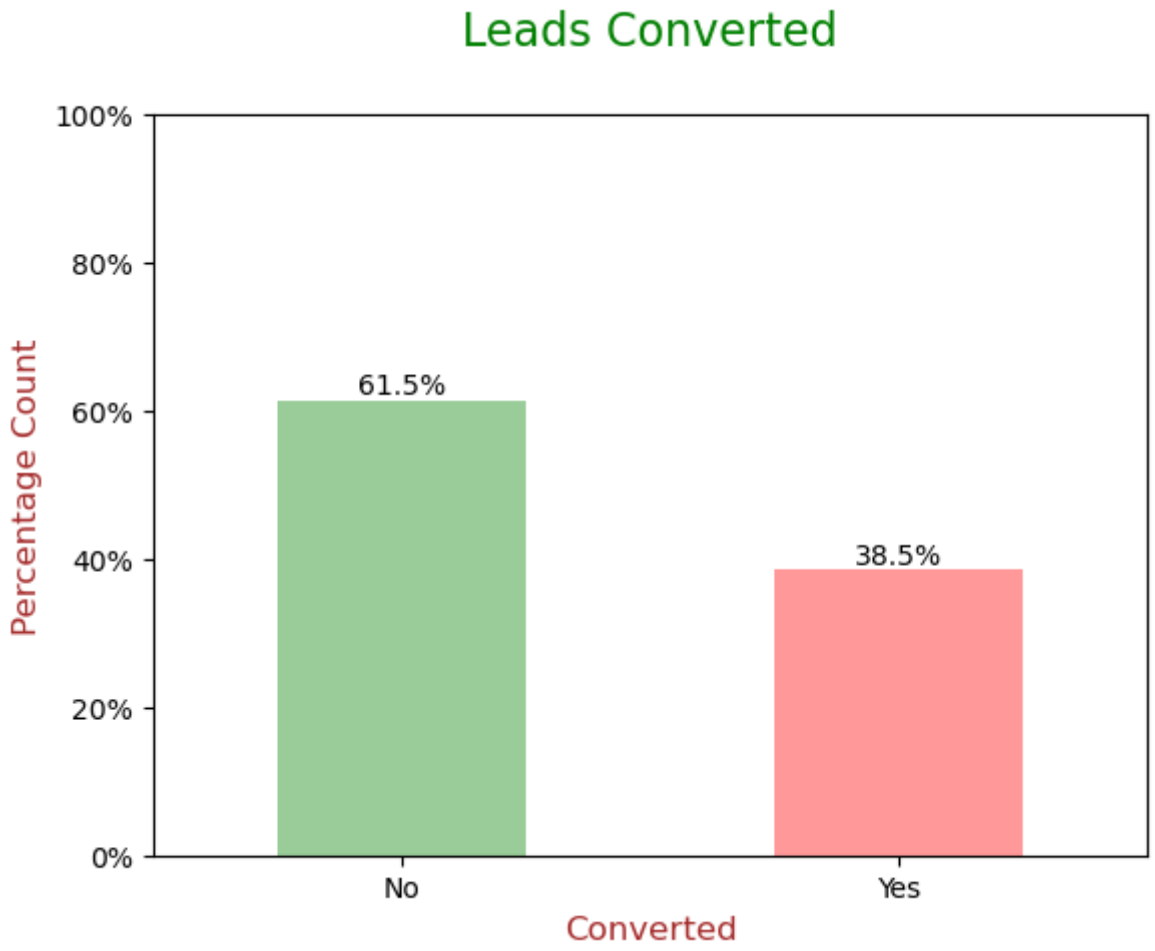
4: Data Analysis (EDA)

4.1 Checking if Data is Imbalanced or not

Inference:

- The data is imbalanced when there is uneven distribution of observations in the dataset. In other words, one value is in majority and other is in minority.
- Data imbalance is in the context of Target variable only
- Our target variable is 'Converted' which tells whether a past lead was converted or not, and denoted by 1 and zero respectively.

In [ ]:
```
## ploting the results on a bar plot

ax=(100*df_leads["Converted"].value_counts(normalize=True)).plot.bar(color=["Green","Red"],alpha=0.4)

# Adding and formatting title
plt.title("Leads Converted\n", fontdict={'fontsize': 16, 'fontweight' : 12, 'color' : 'Green'})


# Labeling Axes
plt.xlabel('Converted', fontdict={'fontsize': 12, 'fontweight' : 20, 'color' : 'Brown'})
plt.ylabel("Percentage Count", fontdict={'fontsize': 12, 'fontweight' : 20, 'color' : 'Brown'})

# modification ticks y axis
ticks=np.arange(0,101,20)
labels=["{:.0f}%".format(i) for i in ticks]
plt.yticks(ticks,labels)

#xticks
plt.xticks([0,1],["No","Yes"])
plt.xticks(rotation=0)

for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()), (p.get_x() + p.get_width() / 2., p.get_height()),
                 ha = 'center', va = 'center', xytext = (0, 5), textcoords = 'offset points')

plt.show()
```



Insights:

- **Rate of conversion is 38.5%**: Only 38.5% of the leads have converted from cold to hot.
- Remaining 61.5% of the people didnt convert to leads.

In [ ]:
```
### Ratio of Data Imbalance
ratio=(df_leads["Converted"].value_counts(normalize=True).loc[0])/(df_leads["Converted"].value_counts(normalize=True).loc[1])

print("Data Imbalance Ratio : {:.2f} : {}".format(ratio,1))
```

```
Data Imbalance Ratio : 1.59 : 1
```

4.2 Univariate Analysis

In [ ]:
```
df_leads.head()
```

Out[ ]:

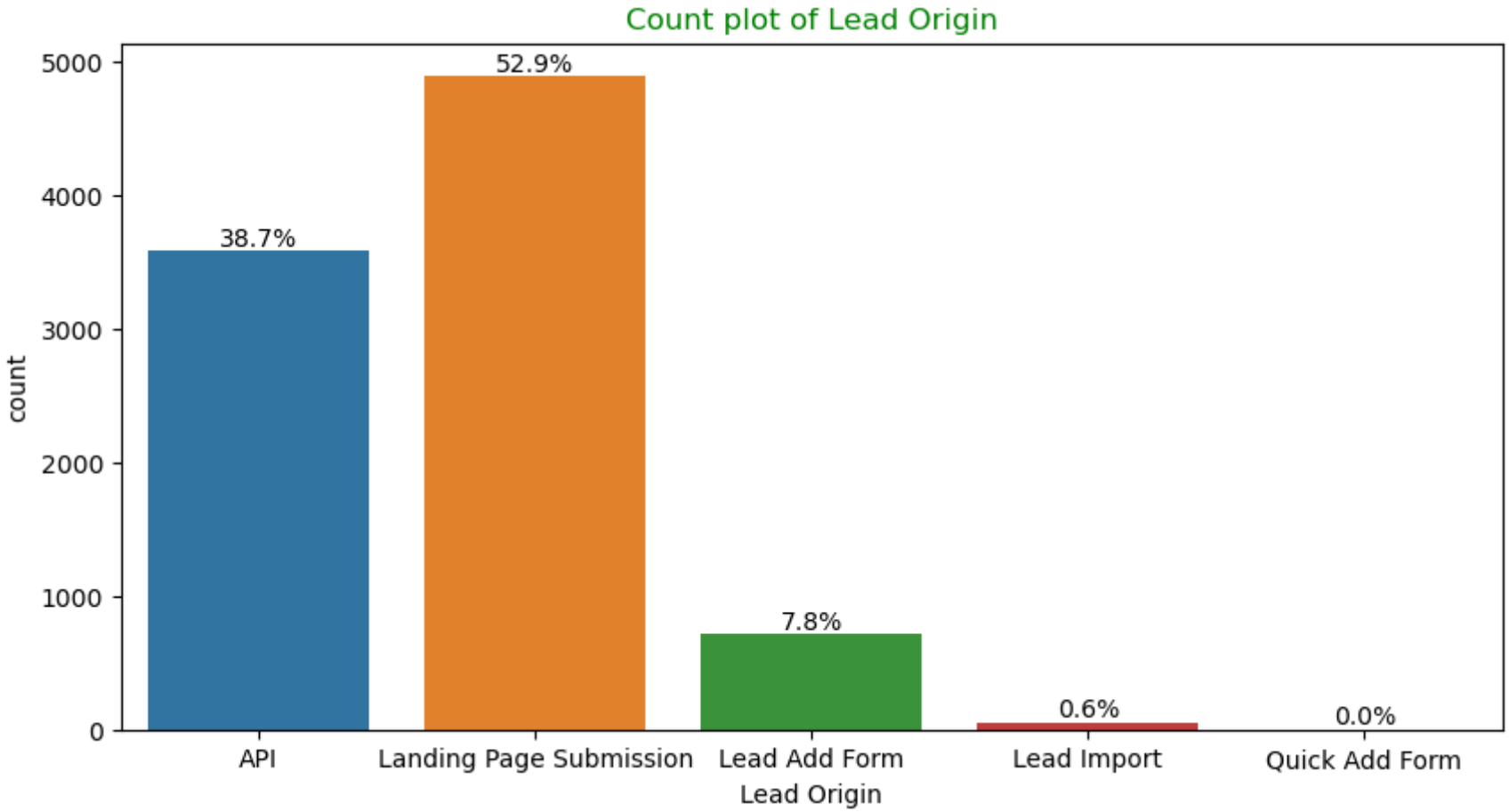
	Lead Origin	Lead Source	Do Not Email	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Last Activity	Specialization	Current_occupation	Free_copy
0	API	Olark Chat	0	0	0.0	0	0.0	Page Visited on Website	Others	Unemployed	0
1	API	Organic Search	0	0	5.0	674	2.5	Email Opened	Others	Unemployed	0
2	Landing Page Submission	Direct Traffic	0	1	2.0	1532	2.0	Email Opened	Business Administration	Student	1
3	Landing Page Submission	Direct Traffic	0	0	1.0	305	1.0	Others	Media and Advertising	Unemployed	0
4	Landing Page Submission	Google	0	1	2.0	1428	1.0	Converted to Lead	Others	Unemployed	0

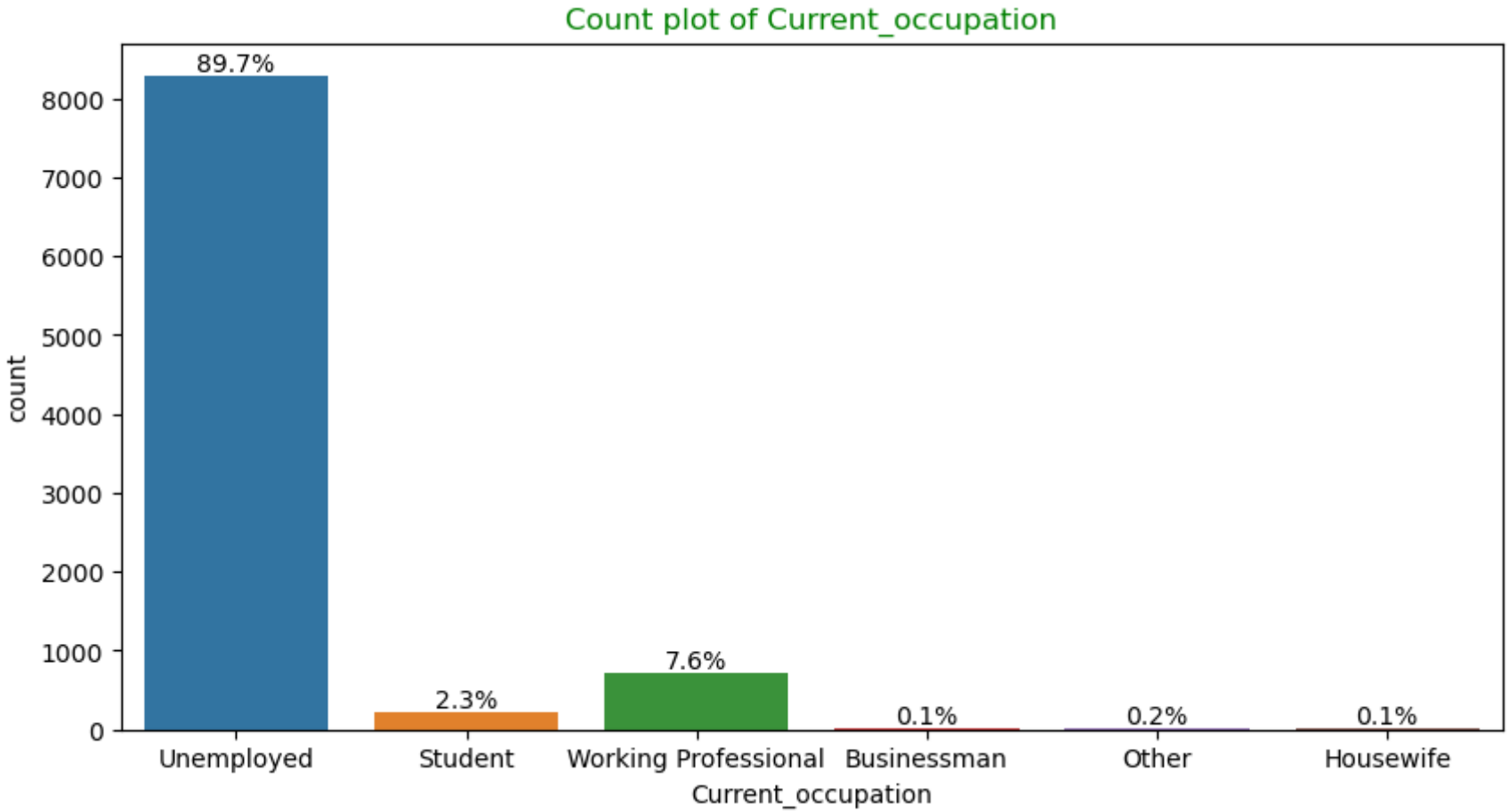
Categorical Variables

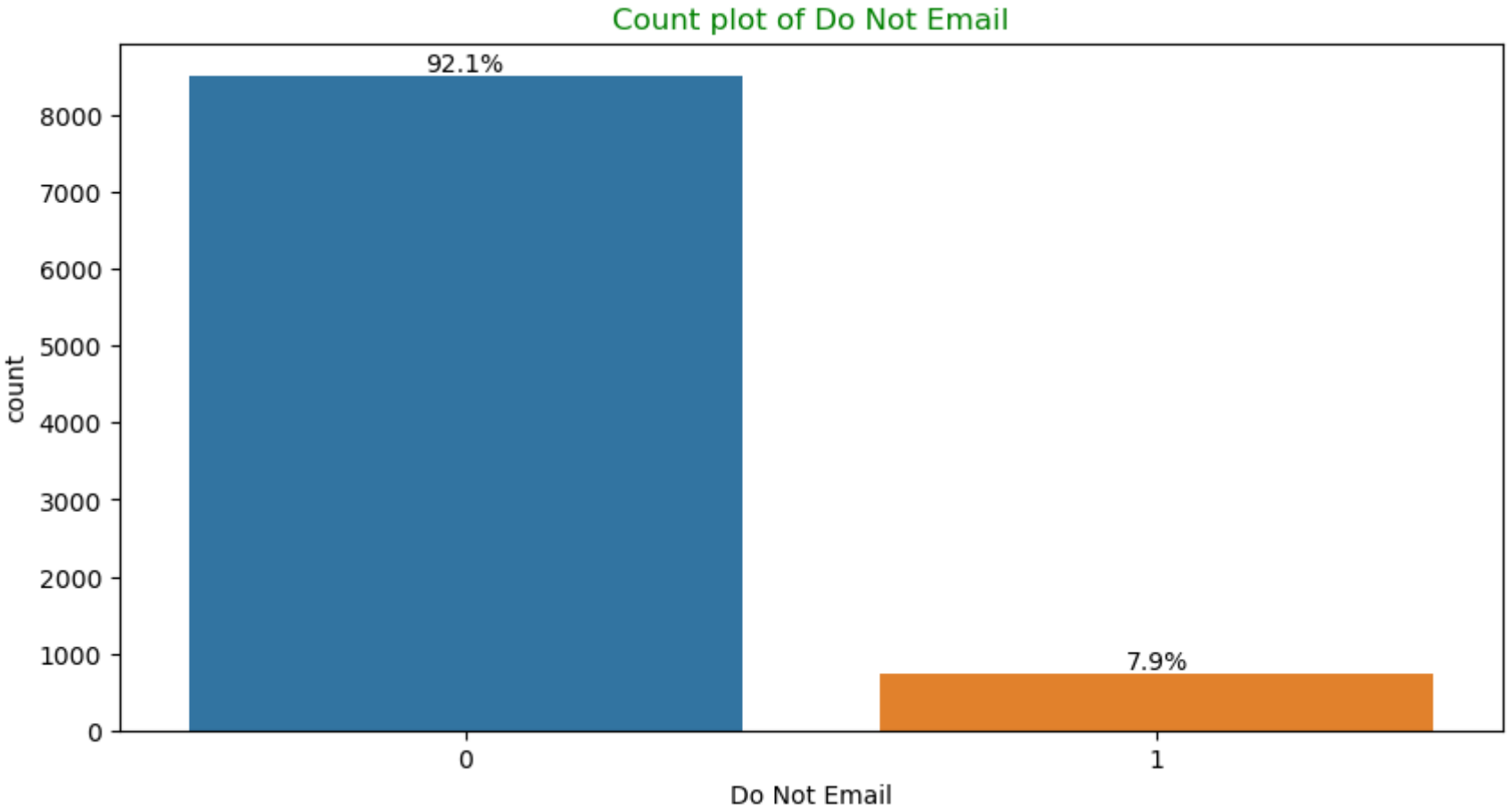
In [ ]:
```
#List of categorical columns
cat_cols = ["Lead Origin","Current_occupation","Do Not Email",
            "Free_copy","Lead Source","Last Activity","Specialization"]
```

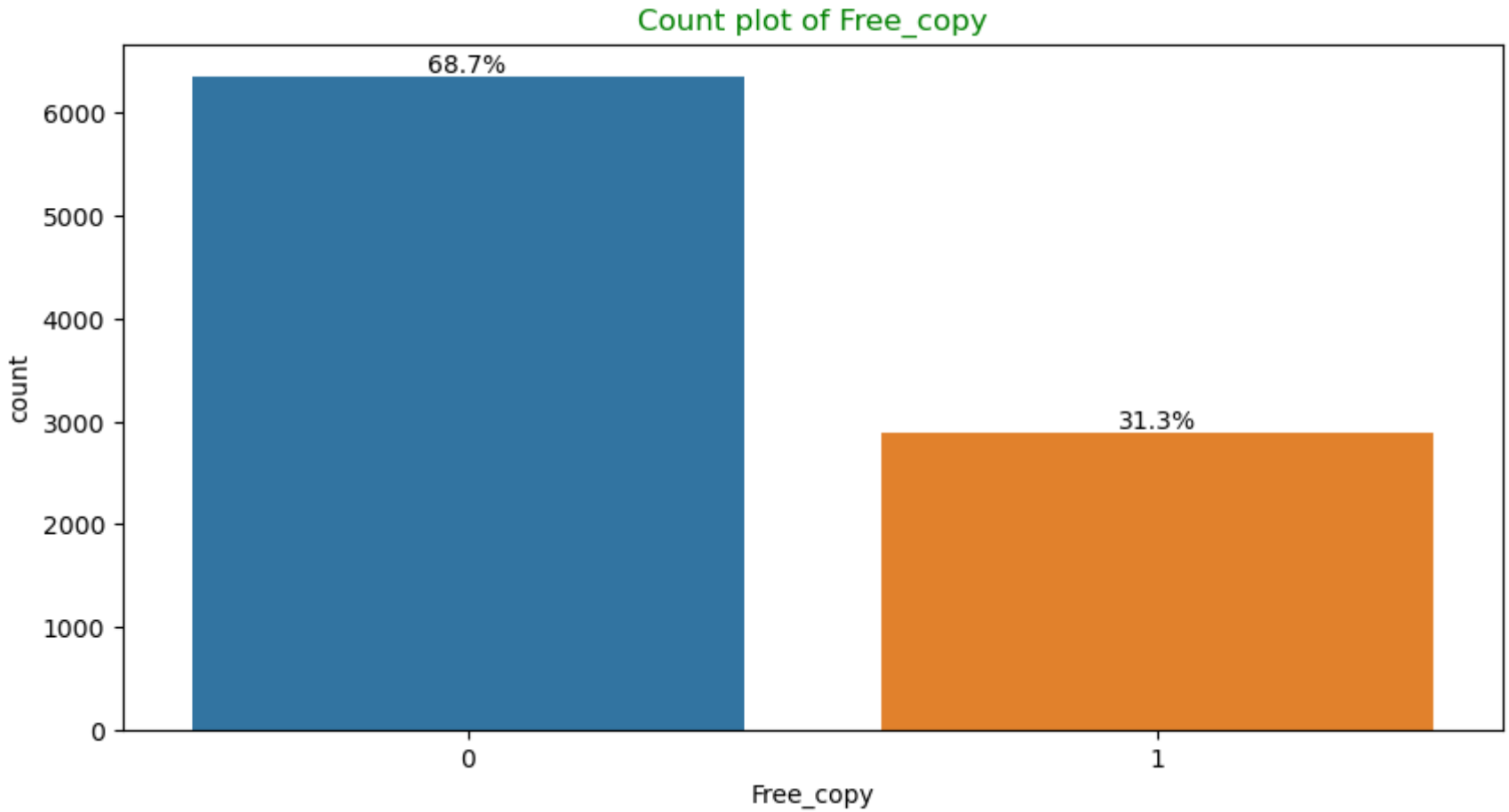
In [ ]:
```
# countplot of columns with its value_counts percentage as annotation
for i in cat_cols[:4]:

    plt.figure(figsize=[10,5])
    plt.title("Count plot of {}".format(i),color="green")
    ax=sns.countplot(x=i,data=df_leads)
    total=len(df_leads[i])
    plt.xticks(rotation=0)

    for p in ax.patches:
        text = '{:.1f}%'.format(100*p.get_height()/total)
```

```
        x = p.get_x() + p.get_width() / 2.
        y = p.get_height()

        ax.annotate(text, (x,y), ha = 'center', va = 'center', xytext = (0, 5), textcoords = 'offset points')
```



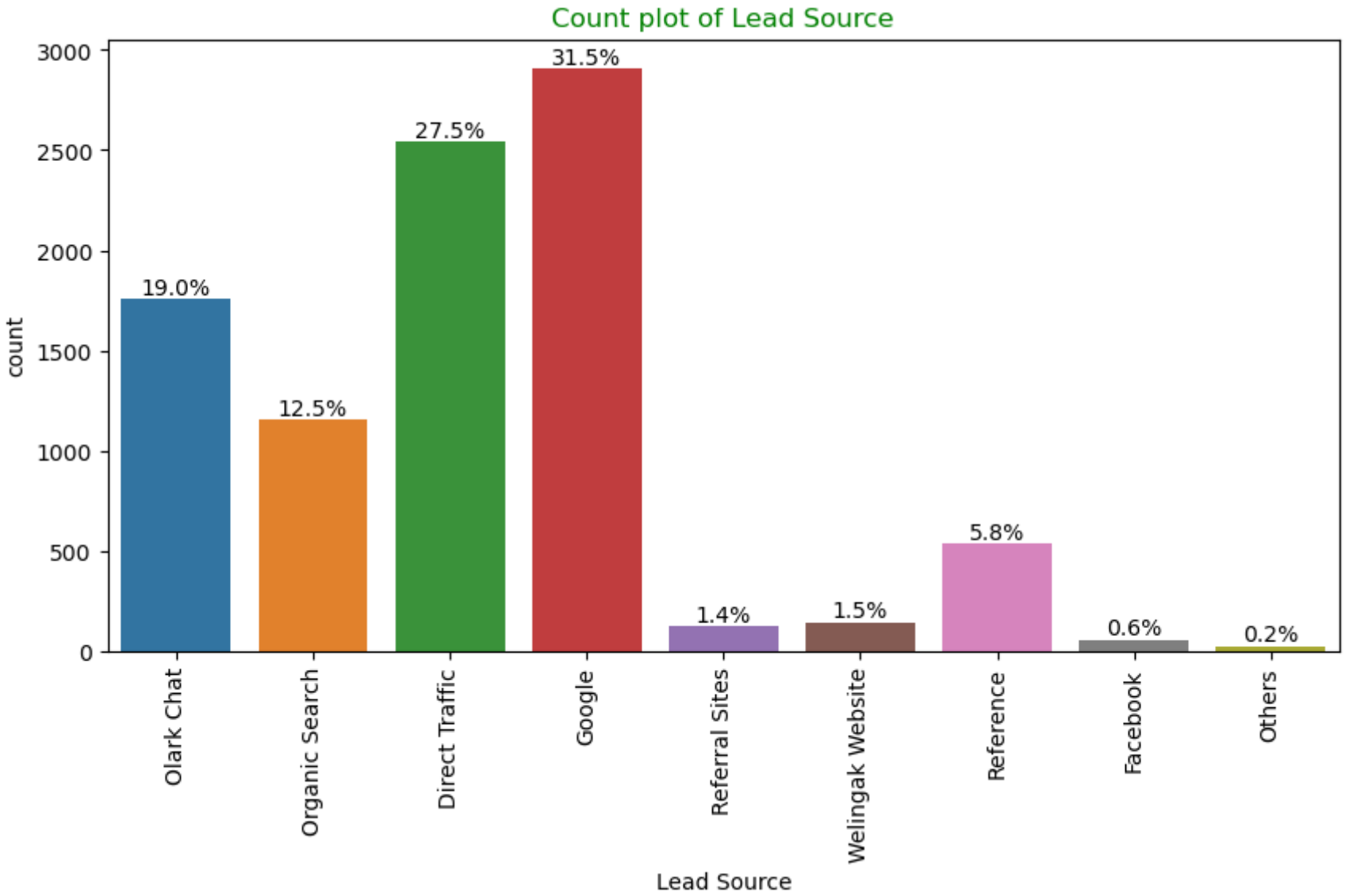

Count plot of Current_occupation
89.7%
2.3%
7.6%
0.1%
0.2%
0.1%
8000
7000
6000
5000
4000
3000
2000
1000
0
count
Unemployed
Student
Working Professional
Businessman
Other
Housewife
Current_occupation

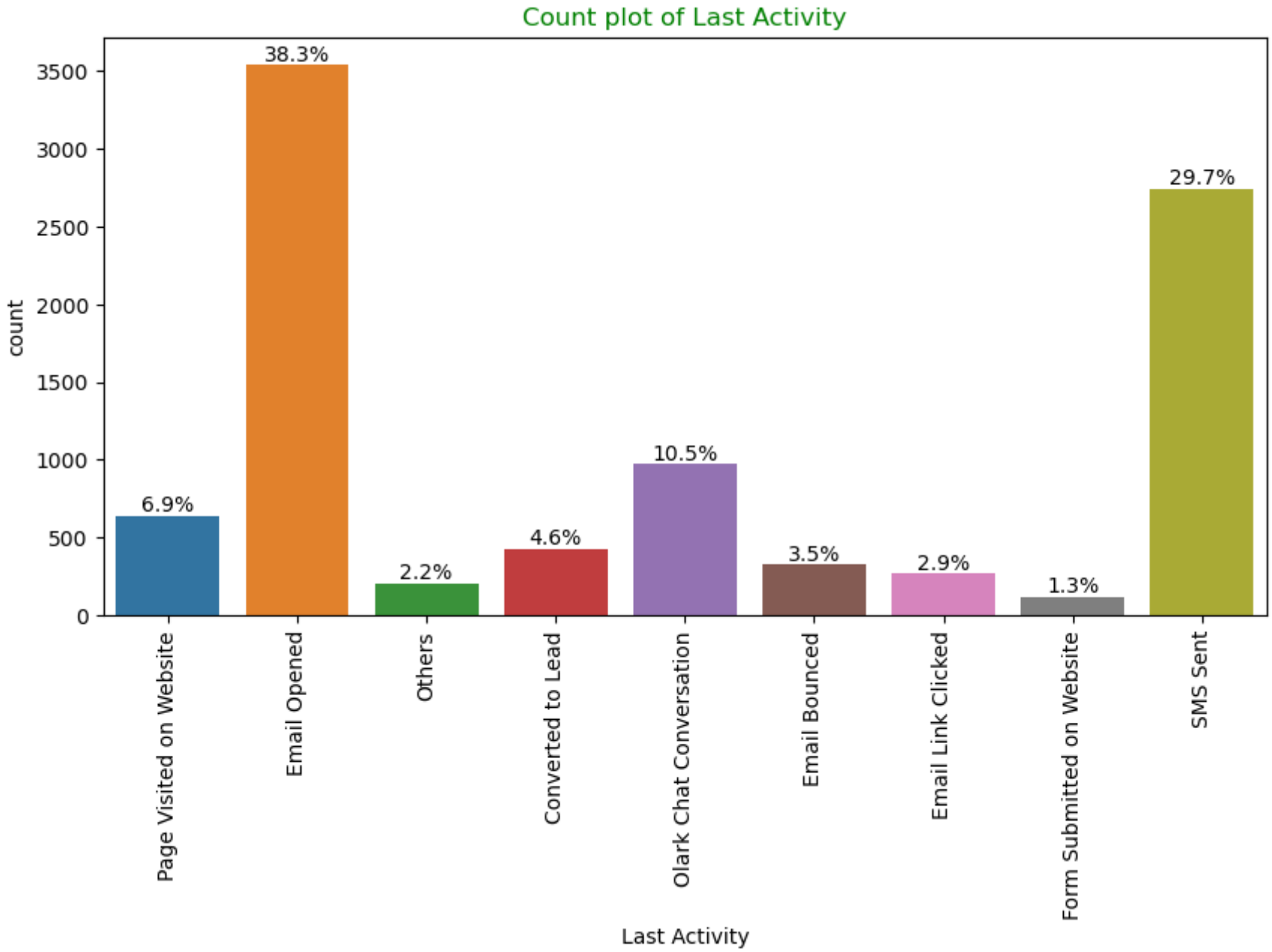

Count plot of Do Not Email
92.1%
7.9%
8000
7000
6000
5000
4000
3000
2000
1000
0
count
0
1
Do Not Email

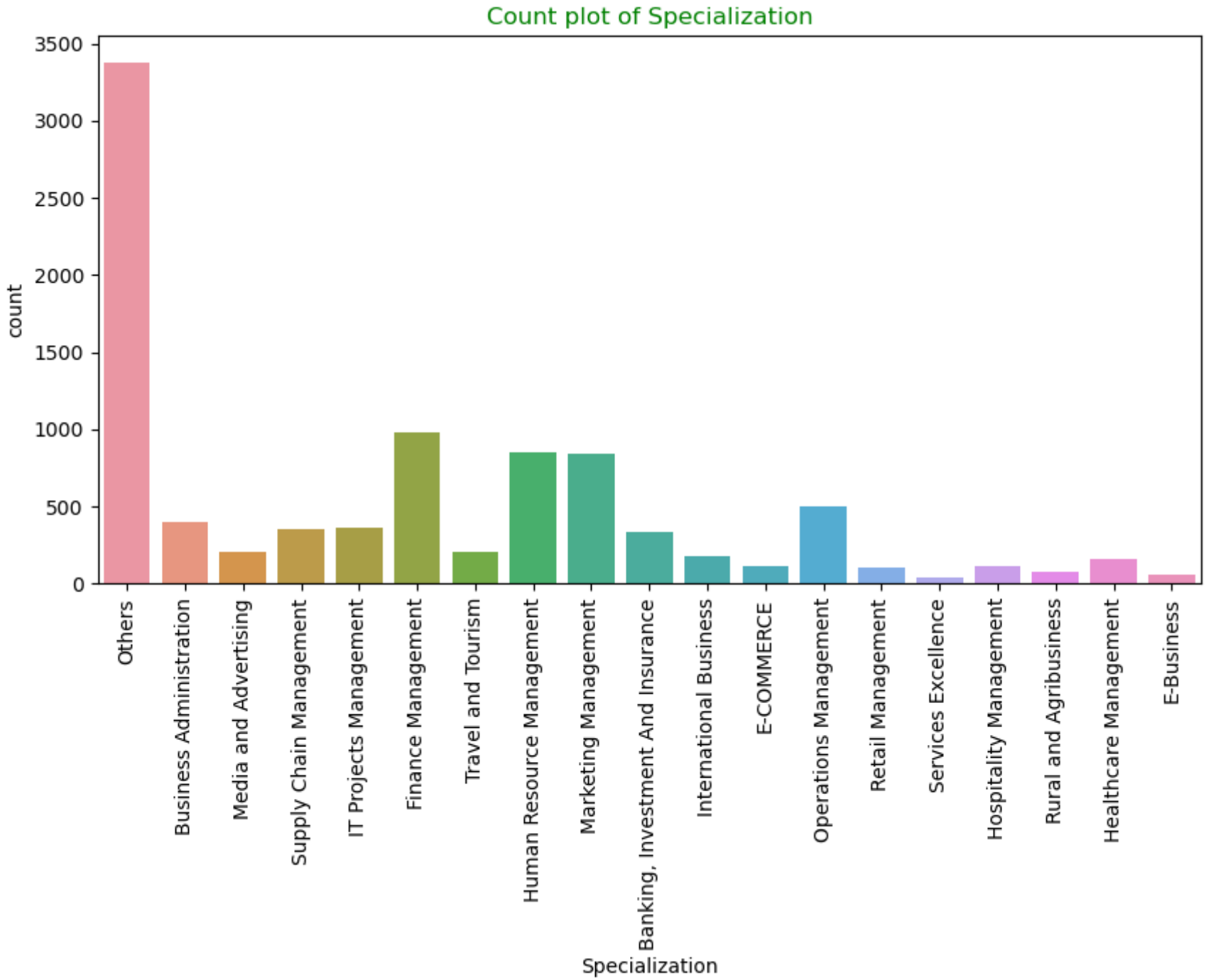


In [ ]:
```
# Barplot for remaining columns from cat_cols (Did separate to rotate xticks 90* so labels doesnt become messy)
for i in cat_cols[4:]:

    plt.figure(figsize=[10,5])
    plt.title("Count plot of {}".format(i),color="green")
    ax=sns.countplot(x=i,data=df_leads)
    total=len(df_leads[i])
    plt.xticks(rotation=90)


    if i!="Specialization":        # (not doing for Specialization xtick labels will be messy)
        for p in ax.patches:
            text = '{:.1f}%'.format(100*p.get_height()/total)
            x = p.get_x() + p.get_width() / 2.
```

```
            y = p.get_height()

            ax.annotate(text, (x,y), ha = 'center', va = 'center', xytext = (0, 5), textcoords = 'offset points')
    else:
        pass
```




Count plot of Last Activity
count
Last Activity
0
500
1000
1500
2000
2500
3000
3500
6.9%
38.3%
2.2%
4.6%
10.5%
3.5%
2.9%
1.3%
29.7%
Page Visited on Website
Email Opened
Others
Converted to Lead
Olark Chat Conversation
Email Bounced
Email Link Clicked
Form Submitted on Website
SMS Sent


Count plot of Specialization
count
3500
3000
2500
2000
1500
1000
500
0
Others
Business Administration
Media and Advertising
Supply Chain Management
IT Projects Management
Finance Management
Travel and Tourism
Human Resource Management
Marketing Management
Banking, Investment And Insurance
International Business
E-COMMERCE
Operations Management
Retail Management
Services Excellence
Hospitality Management
Rural and Agribusiness
Healthcare Management
E-Business
Specialization

Inferences:

- In Categorical Univariate Analysis we get to know the value counts percentage in each variable. I.e., the distribution of values in each column.
- With this we get an understanding of which variables can be used in **Bivariate analysis**.

Insights:

List of features from variables present in majority (Including non-converted)

1. **Lead Origin:**
    - "Landing Page Submission: 53%
    - "API": 39%
2. **Current_occupation:**
    - Unemployed: 90%
3. **Do Not Email:** 92%
4. **Lead Source:**
    - Google & Direct Traffic: 58%
5. **Last Activity:**
    - SMS sent and Email opened: 68%

NOTE: These insights will be helpful in further Bivariate Analysis.

4.3 Bivariate Analysis

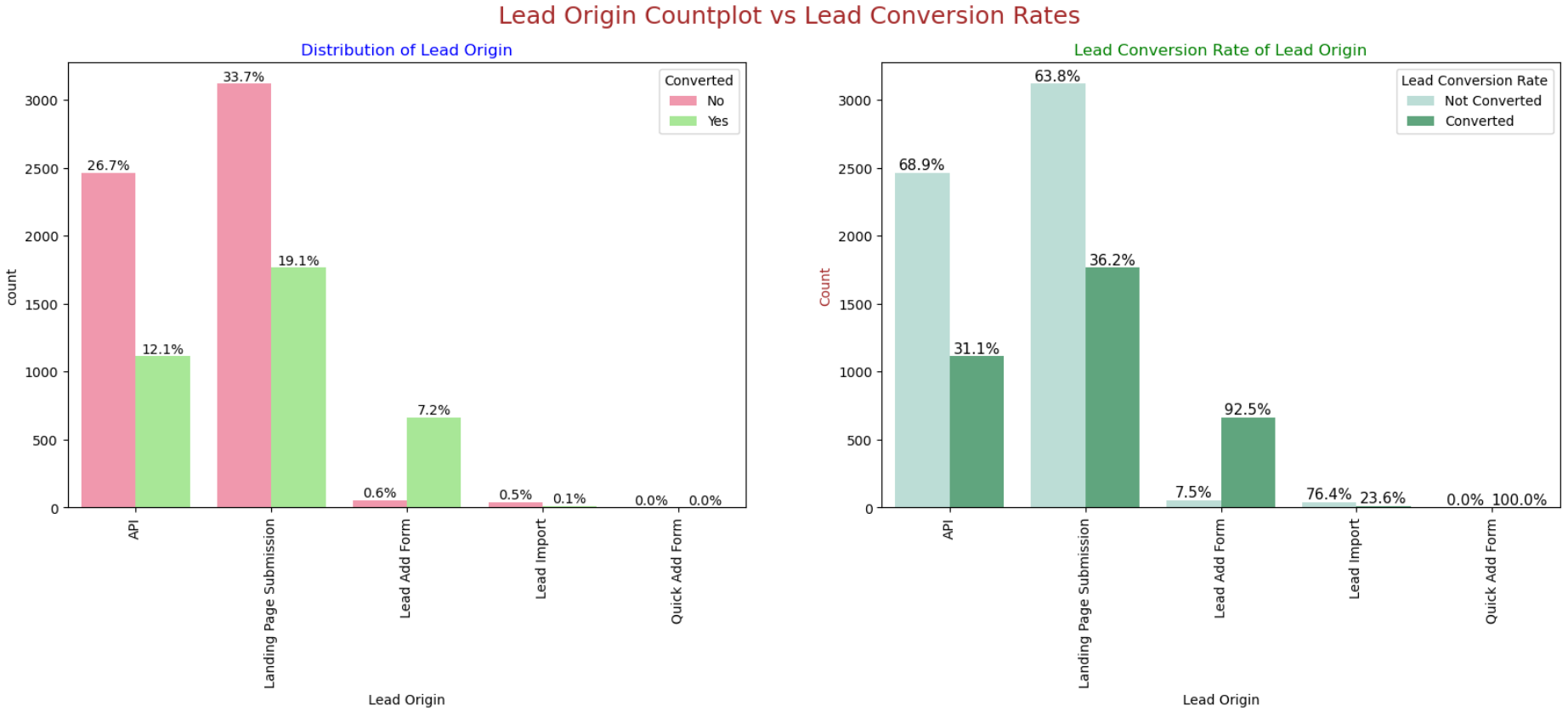
Categorical Variables

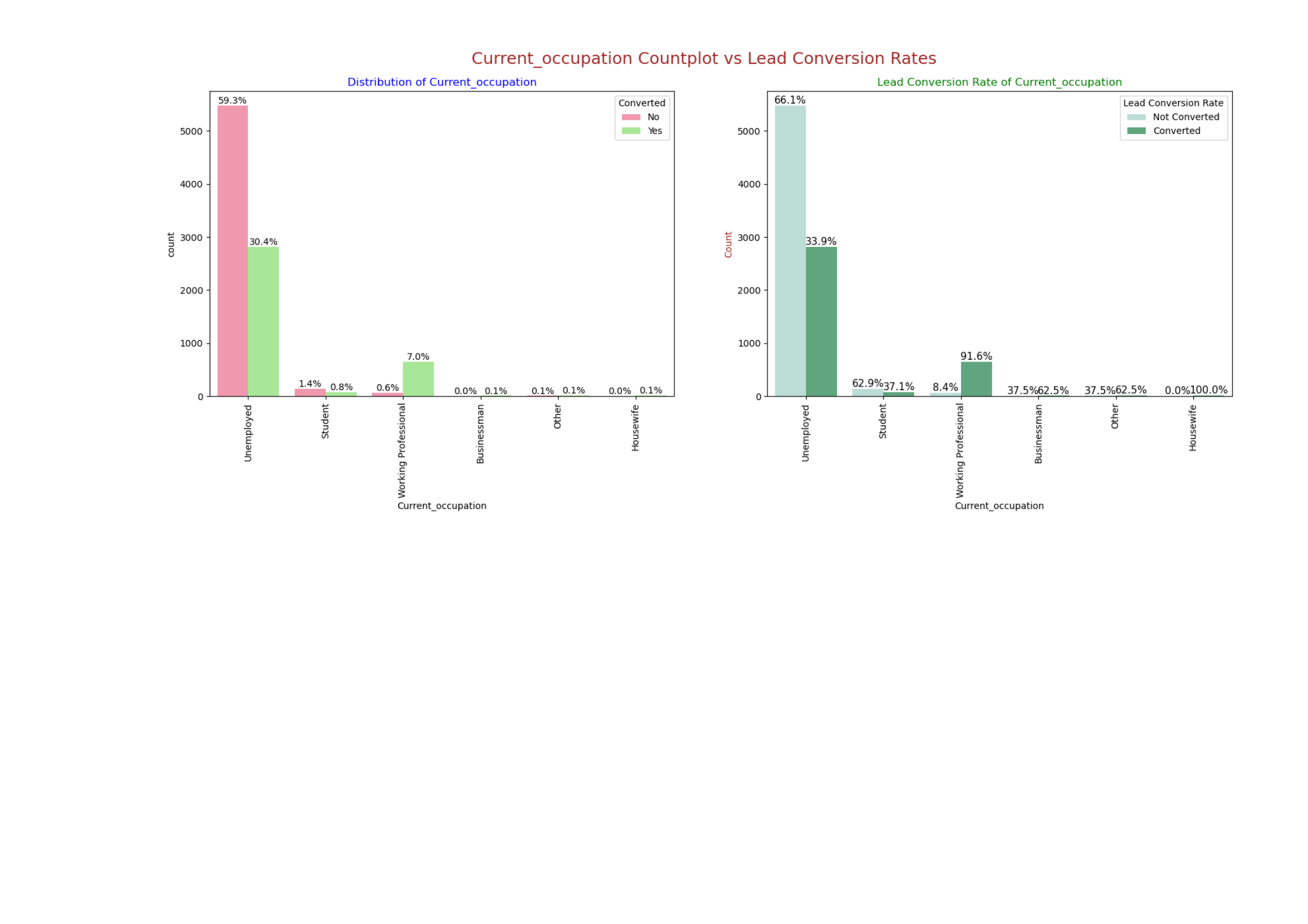
In [ ]:

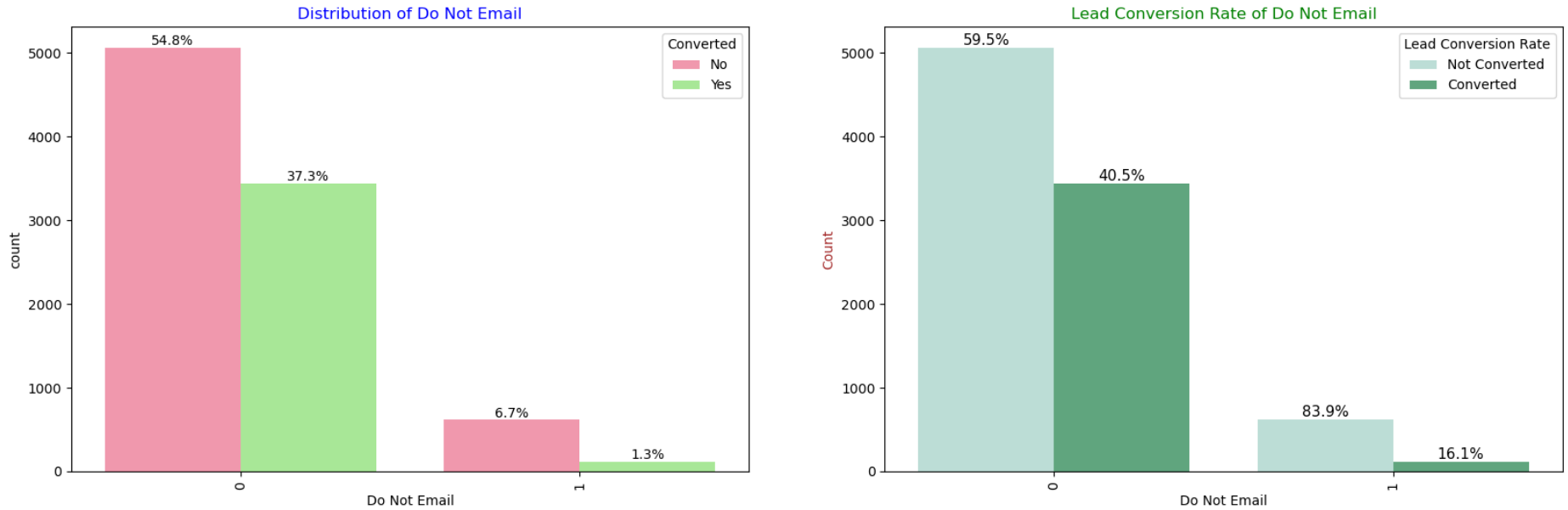
```
# We create a UDF that compares the actual distribution with the raw data
# The 2nd graph gives the conversion rate in percentage

def Bivariate_cat(df,variable_name,Target="Converted"):
    plt.figure(figsize=(20,6))
    plt.suptitle("{} Countplot vs Lead Conversion Rates".format(variable_name),color="Brown", fontsize=18)

    # 1st plot in subplot
    plt.subplot(1,2,1)
    plt.title("Distribution of {}".format(variable_name),color="blue")
    ax=sns.countplot(x=variable_name,hue=Target,data=df_leads,palette="prism_r",alpha=0.46)

    total=len(df_leads[variable_name])
    plt.xticks(rotation=90)
    plt.legend(["No","Yes"],title = "Converted")

    # Annotation for 1st plot
    for p in ax.patches:
        text = '{:.1f}%'.format(100*p.get_height()/total)
        x = p.get_x() + p.get_width() / 2.
        y = p.get_height()

        ax.annotate(text, (x,y), ha = 'center', va = 'center', xytext = (0, 5), textcoords = 'offset points')

    # 2nd plot
    plt.subplot(1,2,2)
    plt.title("Lead Conversion Rate of {}".format(variable_name),color="green",fontsize=12)
    ax=sns.countplot(x=variable_name,hue=Target,data=df,palette="BuGn",alpha=0.85)   #ax1 is for annotation

    # Modifications
    plt.xticks(rotation=90)
    plt.ylabel("Count",color='brown')
    plt.xlabel("{}".format(variable_name))
    plt.legend(labels=["Not Converted","Converted"],title = "Lead Conversion Rate")

    # Annotation for 2nd plot
    # Calculating conversion rates in countplot
```

```
    all_heights = [[p.get_height() for p in bars] for bars in ax.containers]
    for bars in ax.containers:
        for i, p in enumerate(bars):
            total = sum(xgroup[i] for xgroup in all_heights)
            percentage = f'{(100 * p.get_height() / total) :.1f}%'
            ax.annotate(percentage, (p.get_x() + p.get_width() / 2, p.get_height()), size=11, ha='center', va='bottom')
```

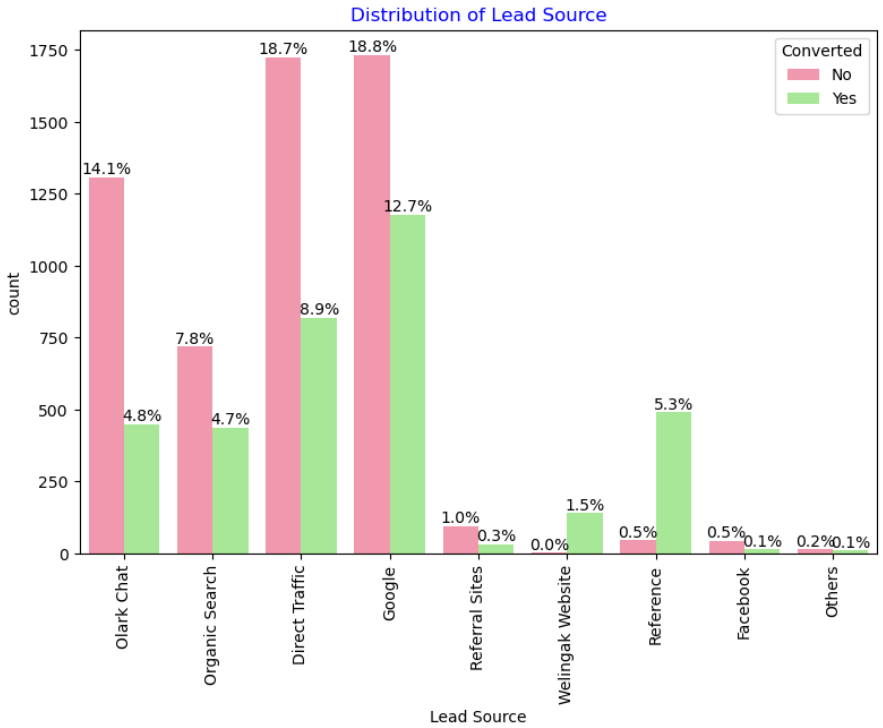
In [ ]:
```
# Bivariate Analysis for all these variables using loop and UDF
cat_cols = ["Lead Origin","Current_occupation","Do Not Email",
            "Lead Source","Last Activity","Specialization","Free_copy"]

for i in cat_cols:
    Bivariate_cat(df_leads,variable_name=i)
```

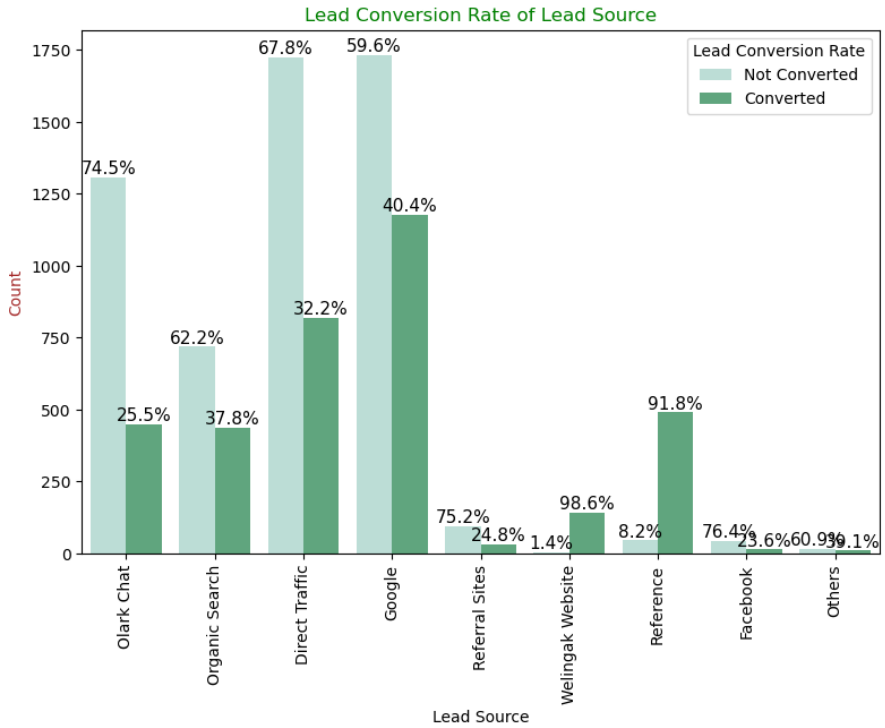


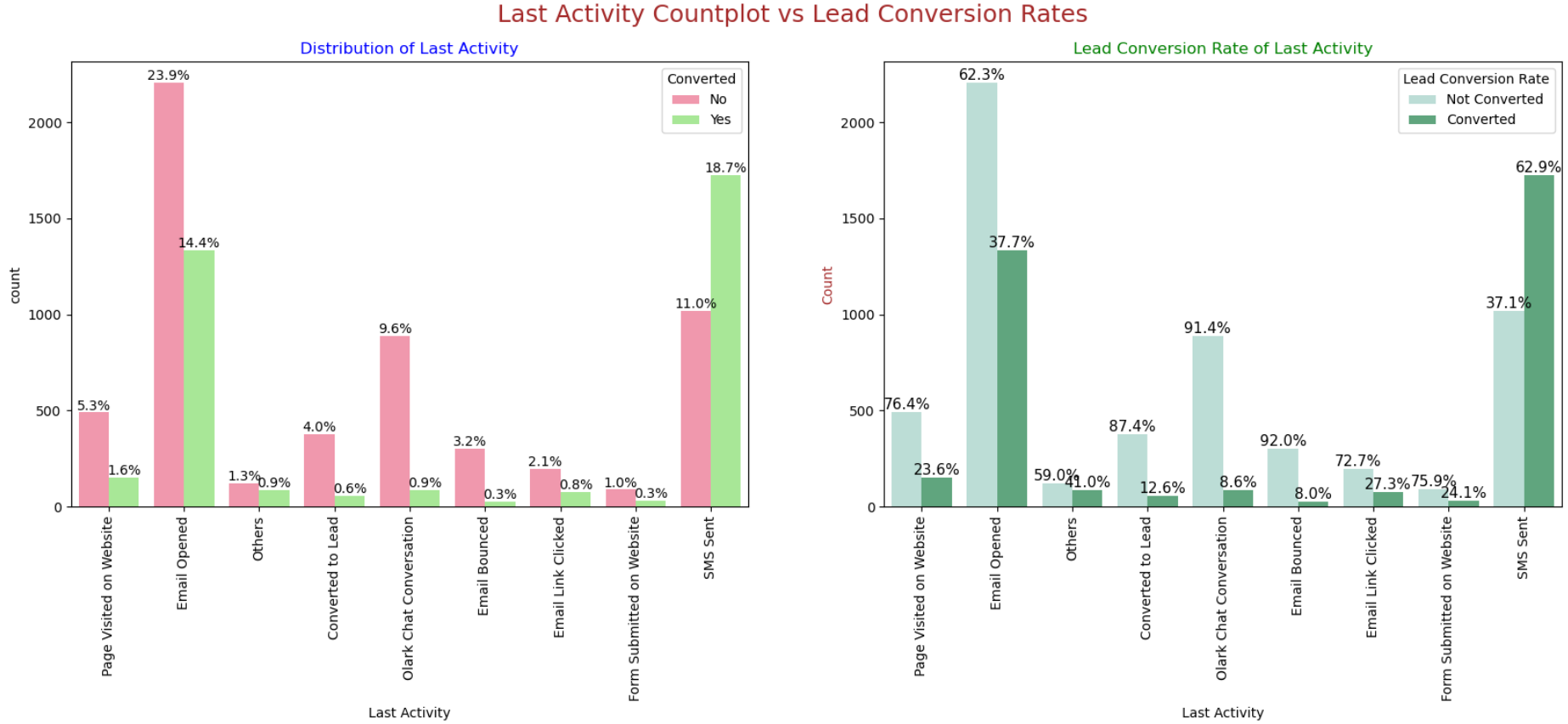

Current_occupation Countplot vs Lead Conversion Rates
Distribution of Current_occupation
Converted
No
Yes
count
59.3%
30.4%
1.4%
0.8%
0.6%
7.0%
0.0%
0.1%
0.1%
0.1%
0.0%
0.1%
Unemployed
Student
Working Professional
Businessman
Other
Housewife
Current_occupation
Lead Conversion Rate of Current_occupation
Lead Conversion Rate
Not Converted
Converted
Count
66.1%
33.9%
62.9%
37.1%
8.4%
91.6%
37.5%
62.5%
37.5%
62.5%
0.0%
100.0%
Unemployed
Student
Working Professional
Businessman
Other
Housewife
Current_occupation


Do Not Email Countplot vs Lead Conversion Rates
Distribution of Do Not Email
Converted
No
Yes
54.8%
37.3%
6.7%
1.3%
count
Do Not Email
Lead Conversion Rate of Do Not Email
Lead Conversion Rate
Not Converted
Converted
59.5%
40.5%
83.9%
16.1%
Count
Do Not Email

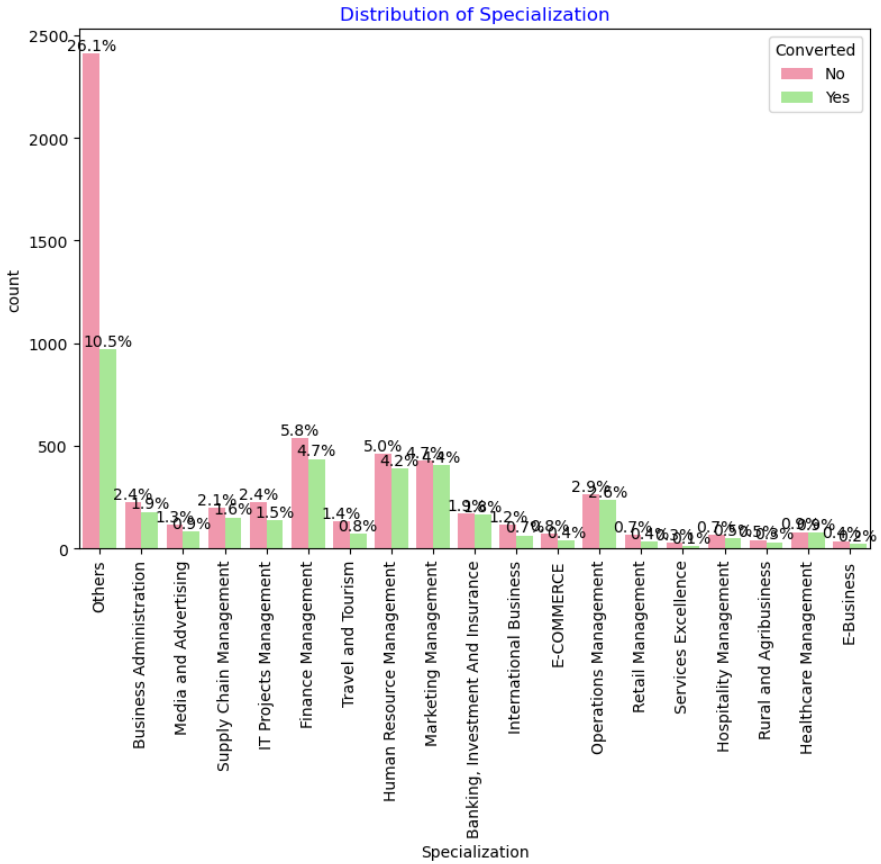
Lead Source Countplot vs Lead Conversion Rates

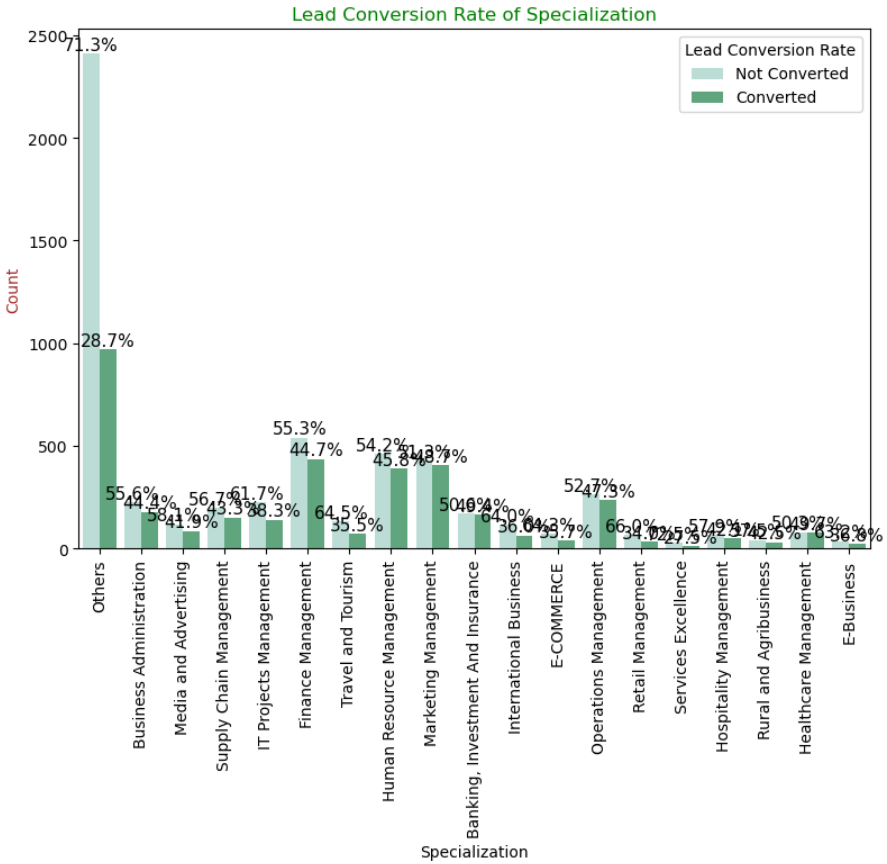


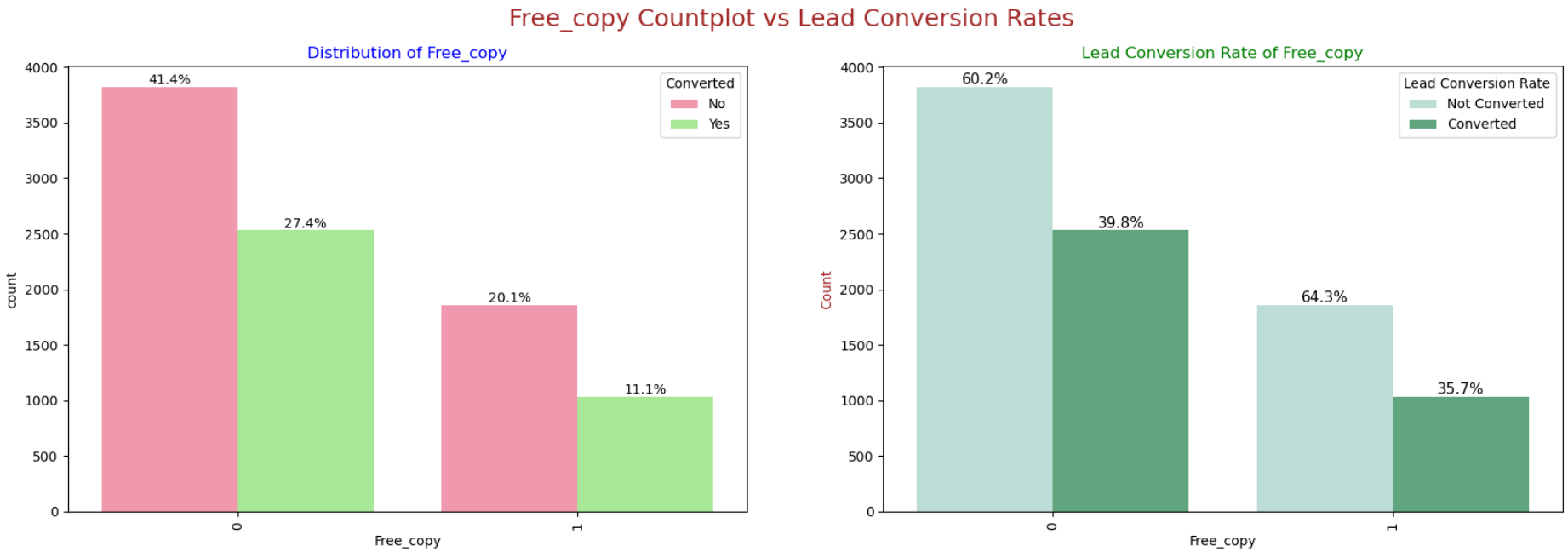



Last Activity Countplot vs Lead Conversion Rates
Distribution of Last Activity
Converted
No
Yes
count
Last Activity
Page Visited on Website
Email Opened
Others
Converted to Lead
Olark Chat Conversation
Email Bounced
Email Link Clicked
Form Submitted on Website
SMS Sent
5.3%
1.6%
23.9%
14.4%
1.3%
0.9%
4.0%
0.6%
9.6%
0.9%
3.2%
0.3%
2.1%
0.8%
1.0%
0.3%
11.0%
18.7%
Lead Conversion Rate of Last Activity
Lead Conversion Rate
Not Converted
Converted
Count
Last Activity
76.4%
23.6%
62.3%
37.7%
59.0%
41.0%
87.4%
12.6%
91.4%
8.6%
92.0%
8.0%
72.7%
27.3%
75.9%
24.1%
37.1%
62.9%

Specialization Countplot vs Lead Conversion Rates







Insights:

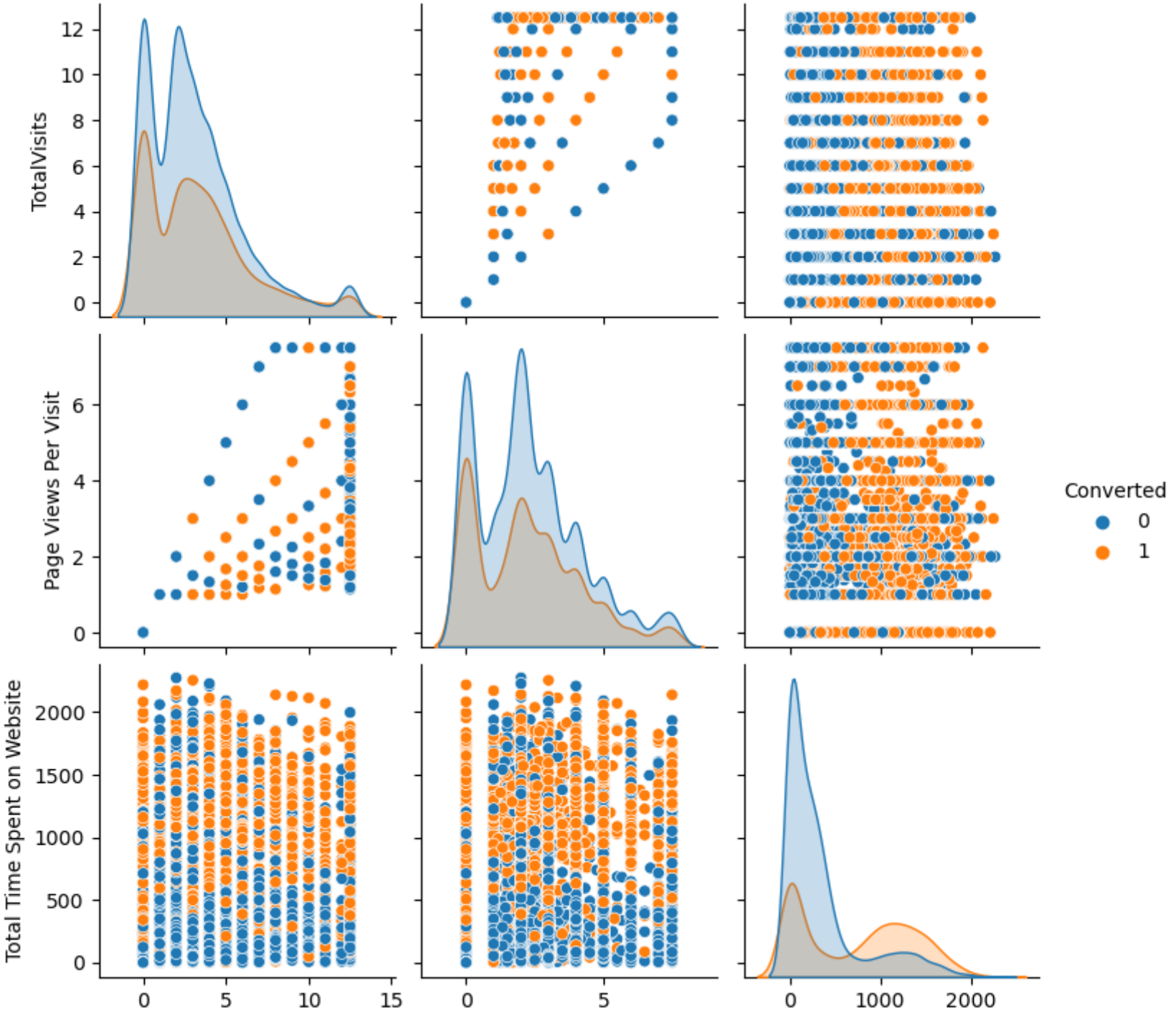
- **Lead Origin:** Around 52% of all leads originated from "*Landing Page Submission*" with a **lead conversion rate (LCR) of 36%**.The "*API*" identified approximately 39% of customers with a **lead conversion rate (LCR) of 31%**.
- **Current_occupation:** Around 90% of the customers are *Unemployed* with **lead conversion rate (LCR) of 34%**. While *Working Professional* contribute only 7.6% of total customers with almost **92% lead conversion rate (LCR)**.
- **Do Not Email:** 92% of the people has opted that they dont want to be emailed about the course.
- **Lead Source:** *Google* has **LCR of 40%** out of 31% customers , *Direct Traffic* contributes **32% LCR** with 27% customers which is lower than Google,*Organic Search* also gives **37.8% of LCR** but the contribution is by only 12.5% of customers ,*Reference* has **LCR of 91%** but there are only around 6% of customers through this Lead Source.
- **Last Activity:** '*SMS Sent*' has **high lead conversion rate of 63%** with 30% contribution from last activities, '*Email Opened*' activity contributed 38% of last activities performed by the customers with 37% lead conversion rate.
- **Specialization:** Marketing Managemt,HR Management,Finance Management shows good contribution.

NOTE: We have assumed **LCR** as **Lead Conversion Rate** in short form.

Bivariate Analysis for Numerical Variables

In [ ]:
```
plt.figure(figsize=(16, 4))
sns.pairplot(data=df_leads,vars=num_cols,hue="Converted")
plt.show()
```

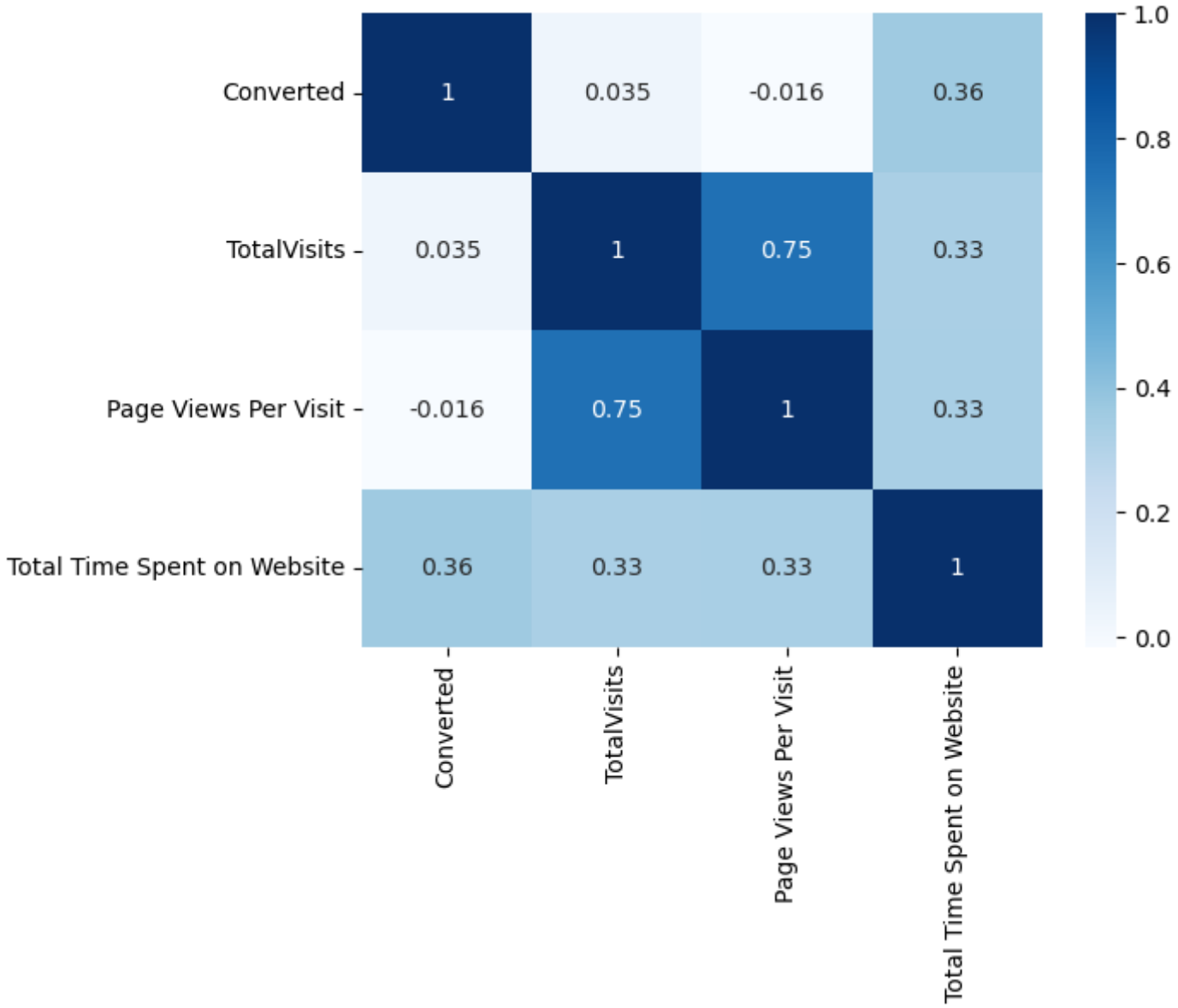
```
<Figure size 1600x400 with 0 Axes>
```


TotalVisits
Page Views Per Visit
Total Time Spent on Website
Converted
0
1

TotalVisits Page Views Per Visit Total Time Spent on Website

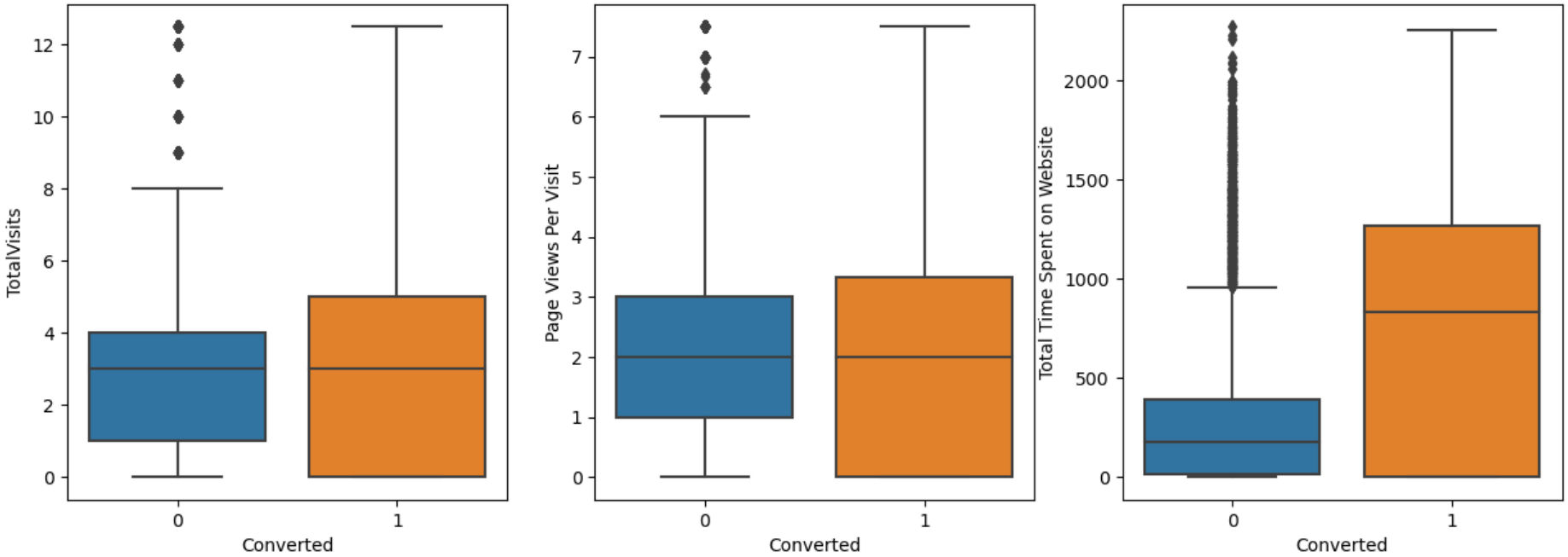
In [ ]:
```
num_cols =["Converted",'TotalVisits', 'Page Views Per Visit', 'Total Time Spent on Website']
```

In [ ]:
```
# Heatmap to show correlation between numerical variables
sns.heatmap(data=df_leads[num_cols].corr(),cmap="Blues",annot=True)
plt.show()
```



In [ ]:
```
# Boxplot with Converted as hue

plt.figure(figsize=(15, 5))
plt.subplot(1,3,1)
sns.boxplot(y = 'TotalVisits', x = 'Converted', data = df_leads)
```

```
plt.subplot(1,3,2)
sns.boxplot(y = 'Page Views Per Visit', x = 'Converted', data = df_leads)
plt.subplot(1,3,3)
sns.boxplot(y = 'Total Time Spent on Website', x = 'Converted', data = df_leads)
plt.show()
```



Inference:

- Past Leads who spends more time on Website are successfully converted than those who spends less as seen in the boxplot

5: Data Preparation

5.1 Dummy Variables

In [ ]:
```
df_leads.head()
```

Out[ ]:

	Lead Origin	Lead Source	Do Not Email	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Last Activity	Specialization	Current_occupation	Free_copy
0	API	Olark Chat	0	0	0.0	0	0.0	Page Visited on Website	Others	Unemployed	0
1	API	Organic Search	0	0	5.0	674	2.5	Email Opened	Others	Unemployed	0
2	Landing Page Submission	Direct Traffic	0	1	2.0	1532	2.0	Email Opened	Business Administration	Student	1
3	Landing Page Submission	Direct Traffic	0	0	1.0	305	1.0	Others	Media and Advertising	Unemployed	0
4	Landing Page Submission	Google	0	1	2.0	1428	1.0	Converted to Lead	Others	Unemployed	0

In [ ]:
```
# Creating dummy variables for some of the categorical variables and dropping the first ones
dummy = pd.get_dummies(df_leads[["Lead Origin","Lead Source","Last Activity","Specialization","Current_occupation"]], drop_first=

# Adding the results to the master dataframe
df_leads = pd.concat([df_leads, dummy], axis=1)
```

In [ ]:
```
df_leads.head()
```

Out[ ]:

	Lead Origin	Lead Source	Do Not Email	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Last Activity	Specialization	Current_occupation	Free_copy	Lead Origin_Landing Page Submission	I Origin_I Add F
0	API	Olark Chat	0	0	0.0	0	0.0	Page Visited on Website	Others	Unemployed	0	0	
1	API	Organic Search	0	0	5.0	674	2.5	Email Opened	Others	Unemployed	0	0	
2	Landing Page Submission	Direct Traffic	0	1	2.0	1532	2.0	Email Opened	Business Administration	Student	1	1	
3	Landing Page Submission	Direct Traffic	0	0	1.0	305	1.0	Others	Media and Advertising	Unemployed	0	1	
4	Landing Page Submission	Google	0	1	2.0	1428	1.0	Converted to Lead	Others	Unemployed	0	1	

In [ ]:
```
# We have created dummies for the below variables, so we can drop them

df_leads = df_leads.drop(["Lead Origin","Lead Source","Last Activity","Specialization","Current_occupation"],1)
```

In [ ]:
```
df_leads.shape
```

Out[ ]:
```
(9240, 49)
```

In [ ]:
```
df_leads.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9240 entries, 0 to 9239
Data columns (total 49 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   Do Not Email                                  9240 non-null   int64
 1   Converted                                     9240 non-null   int64
 2   TotalVisits                                   9240 non-null   float64
 3   Total Time Spent on Website                   9240 non-null   int64
 4   Page Views Per Visit                          9240 non-null   float64
 5   Free_copy                                     9240 non-null   int64
 6   Lead Origin_Landing Page Submission           9240 non-null   uint8
 7   Lead Origin_Lead Add Form                     9240 non-null   uint8
 8   Lead Origin_Lead Import                       9240 non-null   uint8
 9   Lead Origin_Quick Add Form                    9240 non-null   uint8
 10  Lead Source_Facebook                          9240 non-null   uint8
 11  Lead Source_Google                            9240 non-null   uint8
 12  Lead Source_Olark Chat                        9240 non-null   uint8
 13  Lead Source_Organic Search                    9240 non-null   uint8
 14  Lead Source_Others                            9240 non-null   uint8
 15  Lead Source_Reference                         9240 non-null   uint8
 16  Lead Source_Referral Sites                    9240 non-null   uint8
 17  Lead Source_Welingak Website                  9240 non-null   uint8
 18  Last Activity_Email Bounced                   9240 non-null   uint8
 19  Last Activity_Email Link Clicked              9240 non-null   uint8
 20  Last Activity_Email Opened                    9240 non-null   uint8
 21  Last Activity_Form Submitted on Website       9240 non-null   uint8
 22  Last Activity_Olark Chat Conversation         9240 non-null   uint8
 23  Last Activity_Others                          9240 non-null   uint8
 24  Last Activity_Page Visited on Website         9240 non-null   uint8
 25  Last Activity_SMS Sent                        9240 non-null   uint8
 26  Specialization_Business Administration        9240 non-null   uint8
 27  Specialization_E-Business                     9240 non-null   uint8
 28  Specialization_E-COMMERCE                     9240 non-null   uint8
 29  Specialization_Finance Management             9240 non-null   uint8
 30  Specialization_Healthcare Management          9240 non-null   uint8
 31  Specialization_Hospitality Management         9240 non-null   uint8
 32  Specialization_Human Resource Management      9240 non-null   uint8
 33  Specialization_IT Projects Management         9240 non-null   uint8
 34  Specialization_International Business         9240 non-null   uint8
 35  Specialization_Marketing Management           9240 non-null   uint8
 36  Specialization_Media and Advertising          9240 non-null   uint8
 37  Specialization_Operations Management          9240 non-null   uint8
 38  Specialization_Others                         9240 non-null   uint8
```

```
 39  Specialization_Retail Management          9240 non-null   uint8
 40  Specialization_Rural and Agribusiness     9240 non-null   uint8
 41  Specialization_Services Excellence        9240 non-null   uint8
 42  Specialization_Supply Chain Management    9240 non-null   uint8
 43  Specialization_Travel and Tourism         9240 non-null   uint8
 44  Current_occupation_Housewife              9240 non-null   uint8
 45  Current_occupation_Other                  9240 non-null   uint8
 46  Current_occupation_Student                9240 non-null   uint8
 47  Current_occupation_Unemployed             9240 non-null   uint8
 48  Current_occupation_Working Professional   9240 non-null   uint8
dtypes: float64(2), int64(4), uint8(43)
memory usage: 821.3 KB
```

6: Splitting Test-Train

In [ ]:
```
# Putting predictor variables to X
X = df_leads.drop('Converted', axis=1)

# Putting Target variables to y
y = df_leads["Converted"]
```

In [ ]:
```
# Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, test_size=0.3, random_state=100)
```

In [ ]:
```
print("X_train:", X_train.shape,"\ny_train:",y_train.shape)
```

```
X_train: (6468, 48)
y_train: (6468,)
```

In [ ]:
```
print("X_test:", X_test.shape,"\ny_test:",y_test.shape)
```

```
X_test: (2772, 48)
y_test: (2772,)
```

7: Feature Scaling

In [ ]:
```
# using standard scaler for scaling the features
scaler = StandardScaler()

# fetching int64 and float64 dtype columns from dataframe for scaling
```

```
num_cols=X_train.select_dtypes(include=['int64','float64']).columns

X_train[num_cols] = scaler.fit_transform(X_train[num_cols])
```

In [ ]:
```
# X-train dataframe after standard scaling
X_train.head()
```

Out[ ]:

	Do Not Email	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Free_copy	Lead Origin_Landing Page Submission	Lead Origin_Lead Add Form	Lead Origin_Lead Import	Lead Origin_Quick Add Form	Lead Source_Facebook	Lead Source_Google
1871	-0.291638	-1.064974	-0.885371	-1.184892	-0.673169	0	0	0	0	0	0
6795	-0.291638	0.262370	0.005716	-0.488713	1.485511	1	0	0	0	0	0
3516	-0.291638	0.594206	-0.691418	0.123715	-0.673169	0	0	0	0	0	0
8105	-0.291638	0.594206	1.365219	1.432322	-0.673169	1	0	0	0	0	1
3934	-0.291638	-1.064974	-0.885371	-1.184892	-0.673169	0	0	0	0	0	0

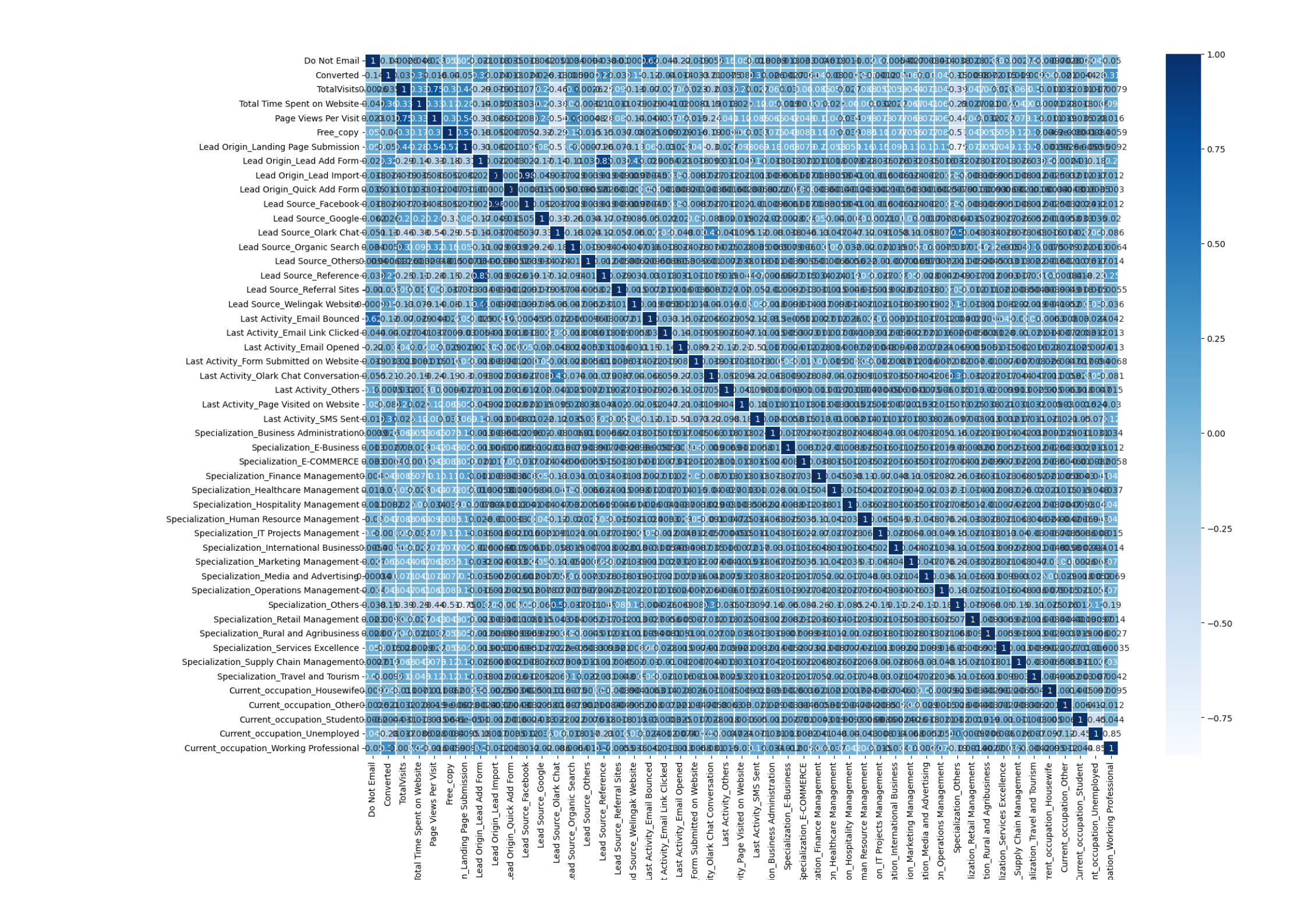
In [ ]:
```
# Checking Lead Conversion Rate (LCR) - The target variable
LCR = (sum(df_leads['Converted'])/len(df_leads['Converted'].index))*100
LCR
```

Out[ ]:
```
38.53896103896104
```

Inference: We have 38.5% Conversion Rate

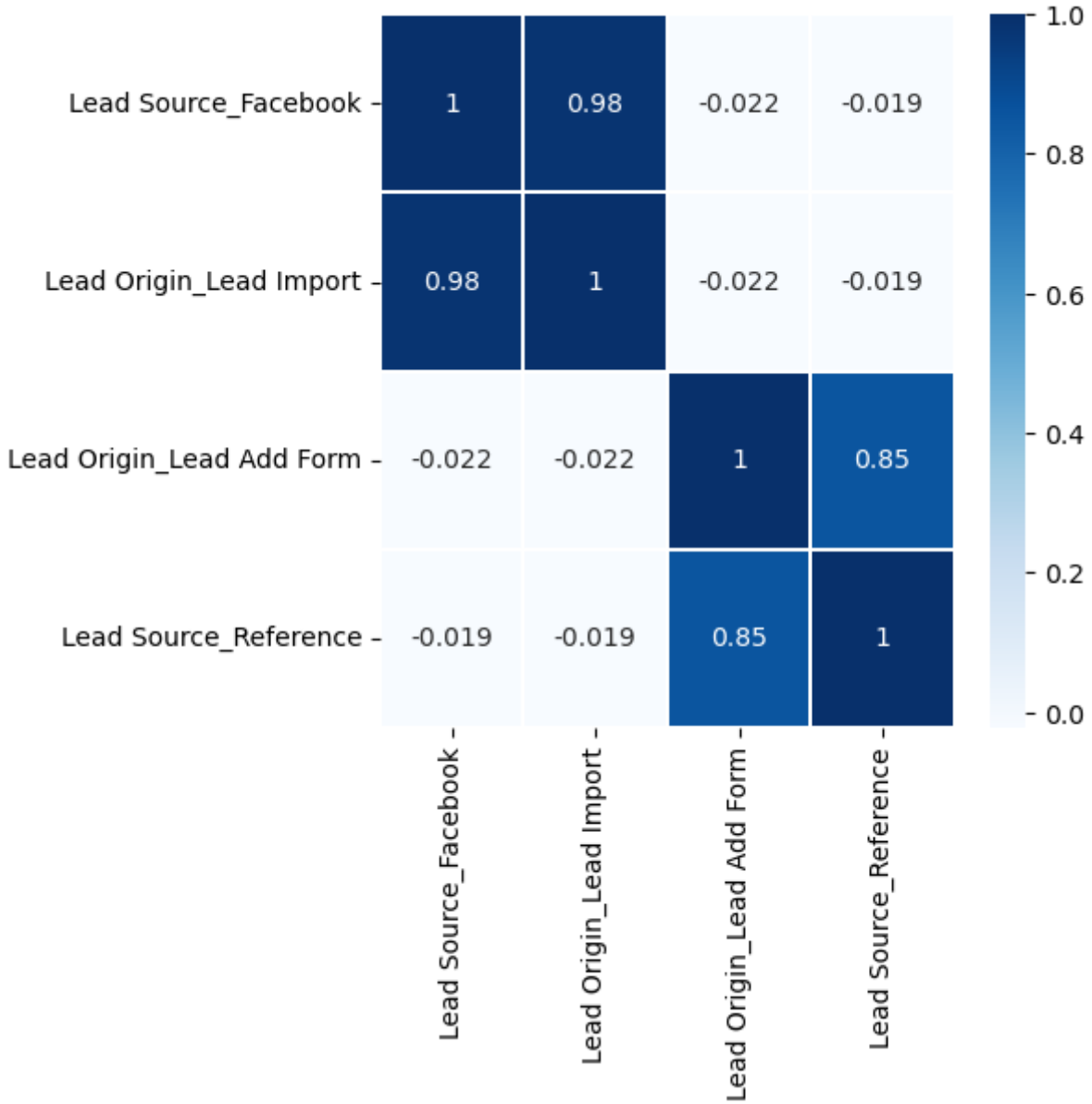
7.1 : Looking at Correlations

In [ ]:
```
# analyzing the correlation matrix
plt.figure(figsize = (20,15))
sns.heatmap(df_leads.corr(),linewidths=0.01,cmap="Blues",annot=True)
plt.show()
```


Do Not Email
Converted
TotalVisits
Total Time Spent on Website
Page Views Per Visit
Free_copy
Lead Origin_Landing Page Submission
Lead Origin_Lead Add Form
Lead Origin_Lead Import
Lead Origin_Quick Add Form
Lead Source_Facebook
Lead Source_Google
Lead Source_Olark Chat
Lead Source_Organic Search
Lead Source_Others
Lead Source_Reference
Lead Source_Referral Sites
Lead Source_Welingak Website
Last Activity_Email Bounced
Last Activity_Email Link Clicked
Last Activity_Email Opened
Last Activity_Form Submitted on Website
Last Activity_Olark Chat Conversation
Last Activity_Others
Last Activity_Page Visited on Website
Last Activity_SMS Sent
Specialization_Business Administration
Specialization_E-Business
Specialization_E-COMMERCE
Specialization_Finance Management
Specialization_Healthcare Management
Specialization_Hospitality Management
Specialization_Human Resource Management
Specialization_IT Projects Management
Specialization_International Business
Specialization_Marketing Management
Specialization_Media and Advertising
Specialization_Operations Management
Specialization_Others
Specialization_Retail Management
Specialization_Rural and Agribusiness
Specialization_Services Excellence
Specialization_Supply Chain Management
Specialization_Travel and Tourism
Current_occupation_Housewife
Current_occupation_Other
Current_occupation_Student
Current_occupation_Unemployed
Current_occupation_Working Professional
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

In [ ]:

```
# The above heatmap has too many columns and is not easy to infer information from. Hence, we'll breakdown suspected variables wh
plt.figure(figsize = (5,5))
sns.heatmap(df_leads[["Lead Source_Facebook","Lead Origin_Lead Import","Lead Origin_Lead Add Form","Lead Source_Reference"]].corr
plt.show()
```



Inference: These predictor variables above are very highly correlated with each other near diagonal with (0.98 and 0.85), it is better that we drop one of these variables from each pair as they won't add much value to the model. So , we can drop any of them, lets drop `'Lead Origin_Lead Import'` and `'Lead Origin_Lead Add Form'`.

In [ ]:
```
X_test = X_test.drop(['Lead Origin_Lead Import','Lead Origin_Lead Add Form'],1)

X_train = X_train.drop(['Lead Origin_Lead Import','Lead Origin_Lead Add Form'],1)
```

8: Model Building

8.1 Feature Selection Using RFE (Recursive Feature Elimination)

In [ ]:
```
# Using RFE to reduce variables
logreg = LogisticRegression()
rfe = RFE(logreg, n_features_to_select=15)
rfe = rfe.fit(X_train, y_train)
```

In [ ]:
```
#checking the output of RFE
list(zip(X_train.columns, rfe.support_, rfe.ranking_))
```

Out[ ]:
```
[('Do Not Email', False, 11),
 ('TotalVisits', False, 9),
 ('Total Time Spent on Website', True, 1),
 ('Page Views Per Visit', False, 10),
 ('Free_copy', False, 18),
 ('Lead Origin_Landing Page Submission', True, 1),
 ('Lead Origin_Quick Add Form', False, 19),
 ('Lead Source_Facebook', True, 1),
 ('Lead Source_Google', False, 22),
 ('Lead Source_Olark Chat', True, 1),
 ('Lead Source_Organic Search', False, 28),
 ('Lead Source_Others', True, 1),
 ('Lead Source_Reference', True, 1),
 ('Lead Source_Referral Sites', False, 24),
 ('Lead Source_Welingak Website', True, 1),
 ('Last Activity_Email Bounced', False, 14),
 ('Last Activity_Email Link Clicked', False, 6),
 ('Last Activity_Email Opened', True, 1),
 ('Last Activity_Form Submitted on Website', False, 29),
 ('Last Activity_Olark Chat Conversation', True, 1),
 ('Last Activity_Others', True, 1),
 ('Last Activity_Page Visited on Website', False, 12),
 ('Last Activity_SMS Sent', True, 1),
 ('Specialization_Business Administration', False, 13),
 ('Specialization_E-Business', False, 31),
 ('Specialization_E-COMMERCE', False, 23),
 ('Specialization_Finance Management', False, 27),
 ('Specialization_Healthcare Management', False, 26),
 ('Specialization_Hospitality Management', True, 1),
 ('Specialization_Human Resource Management', False, 16),
 ('Specialization_IT Projects Management', False, 25),
 ('Specialization_International Business', False, 7),
 ('Specialization_Marketing Management', False, 17),
 ('Specialization_Media and Advertising', False, 30),
 ('Specialization_Operations Management', False, 15),
 ('Specialization_Others', True, 1),
 ('Specialization_Retail Management', False, 8),
 ('Specialization_Rural and Agribusiness', False, 5),
 ('Specialization_Services Excellence', False, 20),
 ('Specialization_Supply Chain Management', False, 21),
 ('Specialization_Travel and Tourism', False, 32),
 ('Current_occupation_Housewife', True, 1),
 ('Current_occupation_Other', False, 3),
 ('Current_occupation_Student', False, 4),
```

```
('Current_occupation_Unemployed', False, 2),
('Current_occupation_Working Professional', True, 1)]
```

In [ ]:

```
# Checknig the top 15 features chosen by RFE
top15=pd.DataFrame()
top15['features']=X_train.columns
top15['Feature Chosen'] = rfe.support_
top15['Ranking']=rfe.ranking_
top15.sort_values(by='Ranking')
```

Out[ ]:

	features	Feature Chosen	Ranking
22	Last Activity_SMS Sent	True	1
41	Current_occupation_Housewife	True	1
35	Specialization_Others	True	1
28	Specialization_Hospitality Management	True	1
20	Last Activity_Others	True	1
19	Last Activity_Olark Chat Conversation	True	1
17	Last Activity_Email Opened	True	1
14	Lead Source_Welingak Website	True	1
12	Lead Source_Reference	True	1
11	Lead Source_Others	True	1
45	Current_occupation_Working Professional	True	1
9	Lead Source_Olark Chat	True	1
7	Lead Source_Facebook	True	1
5	Lead Origin_Landing Page Submission	True	1
2	Total Time Spent on Website	True	1
44	Current_occupation_Unemployed	False	2
42	Current_occupation_Other	False	3
43	Current_occupation_Student	False	4
37	Specialization_Rural and Agribusiness	False	5
16	Last Activity_Email Link Clicked	False	6
31	Specialization_International Business	False	7
36	Specialization_Retail Management	False	8
1	TotalVisits	False	9
3	Page Views Per Visit	False	10

	features	Feature Chosen	Ranking
0	Do Not Email	False	11
21	Last Activity_Page Visited on Website	False	12
23	Specialization_Business Administration	False	13
15	Last Activity_Email Bounced	False	14
34	Specialization_Operations Management	False	15
29	Specialization_Human Resource Management	False	16
32	Specialization_Marketing Management	False	17
4	Free_copy	False	18
6	Lead Origin_Quick Add Form	False	19
38	Specialization_Services Excellence	False	20
39	Specialization_Supply Chain Management	False	21
8	Lead Source_Google	False	22
25	Specialization_E-COMMERCE	False	23
13	Lead Source_Referral Sites	False	24
30	Specialization_IT Projects Management	False	25
27	Specialization_Healthcare Management	False	26
26	Specialization_Finance Management	False	27
10	Lead Source_Organic Search	False	28
18	Last Activity_Form Submitted on Website	False	29
33	Specialization_Media and Advertising	False	30
24	Specialization_E-Business	False	31
40	Specialization_Travel and Tourism	False	32

In [ ]:
```
# columns selected by RFE
rfe_col = X_train.columns[rfe.support_]
```

```
rfe_col
```

Out[ ]:
```
Index(['Total Time Spent on Website', 'Lead Origin_Landing Page Submission',
       'Lead Source_Facebook', 'Lead Source_Olark Chat', 'Lead Source_Others',
       'Lead Source_Reference', 'Lead Source_Welingak Website',
       'Last Activity_Email Opened', 'Last Activity_Olark Chat Conversation',
       'Last Activity_Others', 'Last Activity_SMS Sent',
       'Specialization_Hospitality Management', 'Specialization_Others',
       'Current_occupation_Housewife',
       'Current_occupation_Working Professional'],
      dtype='object')
```

In [ ]:
```
# columns not selected by RFE
X_train.columns[~rfe.support_]
```

Out[ ]:
```
Index(['Do Not Email', 'TotalVisits', 'Page Views Per Visit', 'Free_copy',
       'Lead Origin_Quick Add Form', 'Lead Source_Google',
       'Lead Source_Organic Search', 'Lead Source_Referral Sites',
       'Last Activity_Email Bounced', 'Last Activity_Email Link Clicked',
       'Last Activity_Form Submitted on Website',
       'Last Activity_Page Visited on Website',
       'Specialization_Business Administration', 'Specialization_E-Business',
       'Specialization_E-COMMERCE', 'Specialization_Finance Management',
       'Specialization_Healthcare Management',
       'Specialization_Human Resource Management',
       'Specialization_IT Projects Management',
       'Specialization_International Business',
       'Specialization_Marketing Management',
       'Specialization_Media and Advertising',
       'Specialization_Operations Management',
       'Specialization_Retail Management',
       'Specialization_Rural and Agribusiness',
       'Specialization_Services Excellence',
       'Specialization_Supply Chain Management',
       'Specialization_Travel and Tourism', 'Current_occupation_Other',
       'Current_occupation_Student', 'Current_occupation_Unemployed'],
      dtype='object')
```

In [ ]:
```
# User defined function for calculating VIFs for variables
def get_vif(model_df):
    X = pd.DataFrame()
    X['Features'] = model_df.columns
    X['VIF'] = [variance_inflation_factor(model_df.values, i) for i in range(model_df.shape[1])]
    X['VIF'] = round(X['VIF'], 2)
```

```
    X = X.sort_values(by='VIF', ascending=False)
    X = X.reset_index(drop=True)
    return X
```

Model 1

In [ ]:
```
# Building a model using statsmodels

# Selecting columns for this model
rfe_col=X_train.columns[rfe.support_]

# Creating X_train dataframe with those variables
X_train_rfe = X_train[rfe_col]

# Adding a constant variable
X_train_sm1 = sm.add_constant(X_train_rfe)

# Create a fitted model
logm1 = sm.GLM(y_train,X_train_sm1,family = sm.families.Binomial()).fit()

logm1.params
```

Out[ ]:
```
const                                   -1.033284
Total Time Spent on Website              1.050544
Lead Origin_Landing Page Submission     -1.272090
Lead Source_Facebook                    -0.696059
Lead Source_Olark Chat                   0.900083
Lead Source_Others                       0.980708
Lead Source_Reference                    2.897685
Lead Source_Welingak Website             5.380227
Last Activity_Email Opened               0.950623
Last Activity_Olark Chat Conversation   -0.553411
Last Activity_Others                     1.258012
Last Activity_SMS Sent                   2.068763
Specialization_Hospitality Management   -1.072037
Specialization_Others                   -1.193681
Current_occupation_Housewife            23.022209
Current_occupation_Working Professional  2.685466
dtype: float64
```

In [ ]:
```
print(logm1.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:              Converted   No. Observations:                 6468
Model:                            GLM   Df Residuals:                     6452
Model Family:                Binomial   Df Model:                           15
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -2732.8
Date:                Mon, 12 Jun 2023   Deviance:                       5465.5
Time:                        00:05:18   Pearson chi2:                 8.09e+03
No. Iterations:                    21   Pseudo R-squ. (CS):             0.3839
Covariance Type:            nonrobust
=============================================================================================
                                        coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------------------
const                                -1.0333      0.144     -7.155      0.000      -1.316      -0.750
Total Time Spent on Website           1.0505      0.039     27.169      0.000       0.975       1.126
Lead Origin_Landing Page Submission  -1.2721      0.126    -10.059      0.000      -1.520      -1.024
Lead Source_Facebook                 -0.6961      0.529     -1.316      0.188      -1.733       0.340
Lead Source_Olark Chat                0.9001      0.119      7.585      0.000       0.668       1.133
Lead Source_Others                    0.9807      0.512      1.915      0.056      -0.023       1.985
Lead Source_Reference                 2.8977      0.216     13.434      0.000       2.475       3.320
Lead Source_Welingak Website          5.3802      0.729      7.384      0.000       3.952       6.808
Last Activity_Email Opened            0.9506      0.105      9.061      0.000       0.745       1.156
Last Activity_Olark Chat Conversation -0.5534      0.187     -2.956      0.003      -0.920      -0.186
Last Activity_Others                  1.2580      0.238      5.276      0.000       0.791       1.725
Last Activity_SMS Sent                2.0688      0.108     19.188      0.000       1.857       2.280
Specialization_Hospitality Management -1.0720      0.324     -3.310      0.001      -1.707      -0.437
Specialization_Others                 -1.1937      0.121     -9.841      0.000      -1.431      -0.956
Current_occupation_Housewife          23.0222   1.33e+04      0.002      0.999    -2.6e+04     2.6e+04
Current_occupation_Working Professional  2.6855      0.190     14.104      0.000       2.312       3.059
=============================================================================================
```

NOTE: "Current_occupation_Housewife" column will be removed from the model due to a high p-value of 0.999, which is above the accepted threshold of 0.05 for statistical significance.

Model 2

In [ ]:
```
# Dropping 'Current_occupation_Housewife' column
rfe_col=rfe_col.drop("Current_occupation_Housewife")
```

In [ ]:
```
# Creating X_train dataframe with variables selected by RFE
X_train_rfe = X_train[rfe_col]

# Adding a constant variable
X_train_sm2 = sm.add_constant(X_train_rfe)

# Create a fitted model
logm2 = sm.GLM(y_train,X_train_sm2,family = sm.families.Binomial()).fit()

logm2.params
```

Out[ ]:
```
const                                  -1.025075
Total Time Spent on Website             1.049364
Lead Origin_Landing Page Submission    -1.267369
Lead Source_Facebook                   -0.696913
Lead Source_Olark Chat                  0.899051
Lead Source_Others                      0.973897
Lead Source_Reference                   2.917123
Lead Source_Welingak Website            5.379144
Last Activity_Email Opened              0.949036
Last Activity_Olark Chat Conversation  -0.558345
Last Activity_Others                    1.248172
Last Activity_SMS Sent                  2.058828
Specialization_Hospitality Management  -1.079528
Specialization_Others                  -1.197801
Current_occupation_Working Professional 2.677350
dtype: float64
```

In [ ]:
```
print(logm2.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:              Converted   No. Observations:                 6468
Model:                            GLM   Df Residuals:                     6453
Model Family:                Binomial   Df Model:                           14
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -2740.3
Date:                Mon, 12 Jun 2023   Deviance:                       5480.7
Time:                        00:05:18   Pearson chi2:                 8.12e+03
No. Iterations:                     7   Pseudo R-squ. (CS):             0.3825
Covariance Type:            nonrobust
==============================================================================================
                                         coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------------
const                                 -1.0251      0.144     -7.111      0.000      -1.308      -0.743
Total Time Spent on Website            1.0494      0.039     27.177      0.000       0.974       1.125
Lead Origin_Landing Page Submission   -1.2674      0.126    -10.030      0.000      -1.515      -1.020
Lead Source_Facebook                  -0.6969      0.529     -1.318      0.187      -1.733       0.339
Lead Source_Olark Chat                 0.8991      0.119      7.580      0.000       0.667       1.132
Lead Source_Others                     0.9739      0.512      1.902      0.057      -0.030       1.977
Lead Source_Reference                  2.9171      0.215     13.538      0.000       2.495       3.339
Lead Source_Welingak Website           5.3791      0.729      7.384      0.000       3.951       6.807
Last Activity_Email Opened             0.9490      0.105      9.077      0.000       0.744       1.154
Last Activity_Olark Chat Conversation -0.5583      0.187     -2.985      0.003      -0.925      -0.192
Last Activity_Others                   1.2482      0.238      5.238      0.000       0.781       1.715
Last Activity_SMS Sent                 2.0588      0.108     19.151      0.000       1.848       2.270
Specialization_Hospitality Management -1.0795      0.324     -3.334      0.001      -1.714      -0.445
Specialization_Others                 -1.1978      0.121     -9.881      0.000      -1.435      -0.960
Current_occupation_Working Professional 2.6773     0.190     14.068      0.000       2.304       3.050
==============================================================================================
```

NOTE: "Lead Source_Facebook" column will be removed from model due to high p-value of 0.187, which is above the accepted threshold of 0.05 for statistical significance.

Model 3

In [ ]:
```
# Dropping 'Lead Source_Facebook' column
rfe_col=rfe_col.drop("Lead Source_Facebook")
```

In [ ]:
```
# Creating X_train dataframe with variables selected by RFE
X_train_rfe = X_train[rfe_col]
```

```
# Adding a constant variable
X_train_sm3 = sm.add_constant(X_train_rfe)

# Create a fitted model
logm3 = sm.GLM(y_train,X_train_sm3,family = sm.families.Binomial()).fit()

logm3.params
```

Out[ ]:
```
const                                  -1.040542
Total Time Spent on Website             1.051824
Lead Origin_Landing Page Submission    -1.249311
Lead Source_Olark Chat                  0.916940
Lead Source_Others                      0.981395
Lead Source_Reference                   2.937179
Lead Source_Welingak Website            5.396676
Last Activity_Email Opened              0.943420
Last Activity_Olark Chat Conversation  -0.558250
Last Activity_Others                    1.248700
Last Activity_SMS Sent                  2.055550
Specialization_Hospitality Management  -1.090368
Specialization_Others                  -1.194330
Current_occupation_Working Professional 2.675678
dtype: float64
```

In [ ]:
```
print(logm3.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:              Converted   No. Observations:                 6468
Model:                            GLM   Df Residuals:                     6454
Model Family:                Binomial   Df Model:                           13
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -2741.3
Date:                Mon, 12 Jun 2023   Deviance:                       5482.6
Time:                        00:05:18   Pearson chi2:                 8.12e+03
No. Iterations:                     7   Pseudo R-squ. (CS):             0.3823
Covariance Type:            nonrobust
==============================================================================================
                                         coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------------
const                                 -1.0405      0.144     -7.245      0.000      -1.322      -0.759
Total Time Spent on Website            1.0518      0.039     27.262      0.000       0.976       1.127
Lead Origin_Landing Page Submission   -1.2493      0.125     -9.958      0.000      -1.495      -1.003
Lead Source_Olark Chat                 0.9169      0.118      7.773      0.000       0.686       1.148
Lead Source_Others                     0.9814      0.512      1.917      0.055      -0.022       1.985
Lead Source_Reference                  2.9372      0.215     13.661      0.000       2.516       3.359
Lead Source_Welingak Website           5.3967      0.728      7.409      0.000       3.969       6.824
Last Activity_Email Opened             0.9434      0.105      9.028      0.000       0.739       1.148
Last Activity_Olark Chat Conversation -0.5582      0.187     -2.984      0.003      -0.925      -0.192
Last Activity_Others                   1.2487      0.238      5.239      0.000       0.782       1.716
Last Activity_SMS Sent                 2.0555      0.107     19.124      0.000       1.845       2.266
Specialization_Hospitality Management -1.0904      0.323     -3.377      0.001      -1.723      -0.458
Specialization_Others                 -1.1943      0.121     -9.873      0.000      -1.431      -0.957
Current_occupation_Working Professional  2.6757      0.190     14.063      0.000       2.303       3.049
==============================================================================================
```

NOTE: "Lead Source_Others" column will be removed from model due to high p-value of 0.055, which is above the accepted threshold of 0.05 for statistical significance.

Model 4

In [ ]:
```
# Dropping 'Lead Source_Facebook' column
rfe_col=rfe_col.drop("Lead Source_Others")
```

In [ ]:
```
# Creating X_train dataframe with variables selected by RFE
X_train_rfe = X_train[rfe_col]
```

```
# Adding a constant variable
X_train_sm4 = sm.add_constant(X_train_rfe)

# Create a fitted model
logm4 = sm.GLM(y_train,X_train_sm4,family = sm.families.Binomial()).fit()

logm4.params
```

Out[ ]:
```
const                                  -1.023594
Total Time Spent on Website             1.049789
Lead Origin_Landing Page Submission    -1.258954
Lead Source_Olark Chat                  0.907184
Lead Source_Reference                   2.925326
Lead Source_Welingak Website            5.388662
Last Activity_Email Opened              0.942099
Last Activity_Olark Chat Conversation  -0.555605
Last Activity_Others                    1.253061
Last Activity_SMS Sent                  2.051879
Specialization_Hospitality Management  -1.094445
Specialization_Others                  -1.203333
Current_occupation_Working Professional 2.669665
dtype: float64
```

In [ ]:
```
print(logm4.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:              Converted   No. Observations:                 6468
Model:                            GLM   Df Residuals:                     6455
Model Family:                Binomial   Df Model:                           12
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -2743.1
Date:                Mon, 12 Jun 2023   Deviance:                       5486.1
Time:                        00:05:18   Pearson chi2:                 8.11e+03
No. Iterations:                     7   Pseudo R-squ. (CS):             0.3819
Covariance Type:            nonrobust
=============================================================================================
                                        coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------------------
const                                -1.0236      0.143     -7.145      0.000      -1.304      -0.743
Total Time Spent on Website           1.0498      0.039     27.234      0.000       0.974       1.125
Lead Origin_Landing Page Submission  -1.2590      0.125    -10.037      0.000      -1.505      -1.013
Lead Source_Olark Chat                0.9072      0.118      7.701      0.000       0.676       1.138
Lead Source_Reference                 2.9253      0.215     13.615      0.000       2.504       3.346
Lead Source_Welingak Website          5.3887      0.728      7.399      0.000       3.961       6.816
Last Activity_Email Opened            0.9421      0.104      9.022      0.000       0.737       1.147
Last Activity_Olark Chat Conversation -0.5556      0.187     -2.974      0.003      -0.922      -0.189
Last Activity_Others                  1.2531      0.238      5.259      0.000       0.786       1.720
Last Activity_SMS Sent                2.0519      0.107     19.106      0.000       1.841       2.262
Specialization_Hospitality Management -1.0944      0.323     -3.391      0.001      -1.727      -0.462
Specialization_Others                -1.2033      0.121     -9.950      0.000      -1.440      -0.966
Current_occupation_Working Professional  2.6697      0.190     14.034      0.000       2.297       3.042
=============================================================================================
```

Inference: Model 4 is stable and has significant p-values within the threshold (p-values < 0.05), so we will use it for further analysis.

In [ ]:
```
# Checking VIFs for all variables in the Model 4
get_vif(X_train_rfe)
```

Out[ ]:

	Features	VIF
0	Specialization_Others	2.47
1	Lead Origin_Landing Page Submission	2.45
2	Last Activity_Email Opened	2.36
3	Last Activity_SMS Sent	2.20
4	Lead Source_Olark Chat	2.14
5	Last Activity_Olark Chat Conversation	1.72
6	Lead Source_Reference	1.31
7	Total Time Spent on Website	1.24
8	Current_occupation_Working Professional	1.21
9	Lead Source_Welingak Website	1.08
10	Last Activity_Others	1.08
11	Specialization_Hospitality Management	1.02

Inference: No variable needs to be dropped as they all have good VIF values less than 5.

- p-values for all variables is less than 0.05
- This model is acceptable since both p-values & VIFs seem okay.
- Hence, we shall select Model 4 for `Model Evaluation`.

9: Model Evaluation

In [ ]:
```
# Getting the predicted values on the train set
y_train_pred = logm4.predict(X_train_sm4)          # giving prob. of getting 1

y_train_pred[:10]
```

Out[ ]:
```
1871    0.474082
6795    0.073252
3516    0.249087
8105    0.768973
3934    0.212973
4844    0.987807
3297    0.108454
8071    0.996128
987     0.169259
7423    0.869641
dtype: float64
```

In [ ]:
```
y_train_pred = y_train_pred.values.reshape(-1)
y_train_pred[:10]
```

Out[ ]:
```
array([0.47408215, 0.07325161, 0.24908697, 0.7689732 , 0.2129729 ,
       0.9878074 , 0.1084541 , 0.99612765, 0.16925902, 0.86964073])
```

In [ ]:
```
# Creating a dataframe with the actual churn flag and the predicted probabilities

y_train_pred_final = pd.DataFrame({'Converted':y_train.values, 'Converted_Prob':y_train_pred})
y_train_pred_final['Prospect ID'] = y_train.index
y_train_pred_final.head()
```

Out[ ]:

	Converted	Converted_Prob	Prospect ID
0	0	0.474082	1871
1	0	0.073252	6795
2	0	0.249087	3516
3	0	0.768973	8105
4	0	0.212973	3934

NOTE: Now we have to find the optimal cutoff Threshold value of Probability.

In [ ]:
```
y_train_pred_final['Predicted'] = y_train_pred_final["Converted_Prob"].map(lambda x: 1 if x > 0.5 else 0)

# checking head
y_train_pred_final.head()
```

Out[ ]:

	Converted	Converted_Prob	Prospect ID	Predicted
0	0	0.474082	1871	0
1	0	0.073252	6795	0
2	0	0.249087	3516	0
3	0	0.768973	8105	1
4	0	0.212973	3934	0

9.1 Confusion Matrix

In [ ]:
```
# Confusion matrix  (Actual / predicted)

confusion = metrics.confusion_matrix(y_train_pred_final["Converted"], y_train_pred_final["Predicted"])
print(confusion)
```

```
[[3588  414]
 [ 846 1620]]
```

In [ ]:
```
# Predicted        not_converted  |  converted
# Actual                          |
# ------------------------------------------------
# not_converted        3588       |   414
# converted            846        |   1620


# Above is the confusion matrix when we use threshold of probability as 0.5
```

9.2 Accuracy

In [ ]:
```
# Checking the overall accuracy.
print(metrics.accuracy_score(y_train_pred_final["Converted"], y_train_pred_final["Predicted"]))
```

```
0.8051948051948052
```

9.3 Metrics beyond accuracy

- Sensitivity and Specificity

- When we have Predicted at threshold 0.5 probability

In [ ]:
```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

In [ ]:
```
# Sensitivity of the logistic regression model
print("Sensitivity :",TP / float(TP+FN))
```

```
Sensitivity : 0.656934306569343
```

In [ ]:
```
# Calculating the specificity
print("Specificity :",TN / float(TN+FP))
```

```
Specificity : 0.896551724137931
```

In [ ]:
```
# Calculating false postive conversion rate
print(FP/ float(TN+FP))
```

```
0.10344827586206896
```

In [ ]:
```
# positive predictive value
print (TP / float(TP+FP))
```

```
0.7964601769911505
```

In [ ]:
```
# Negative predictive value
print (TN / float(TN+ FN))
```

```
0.8092016238159675
```

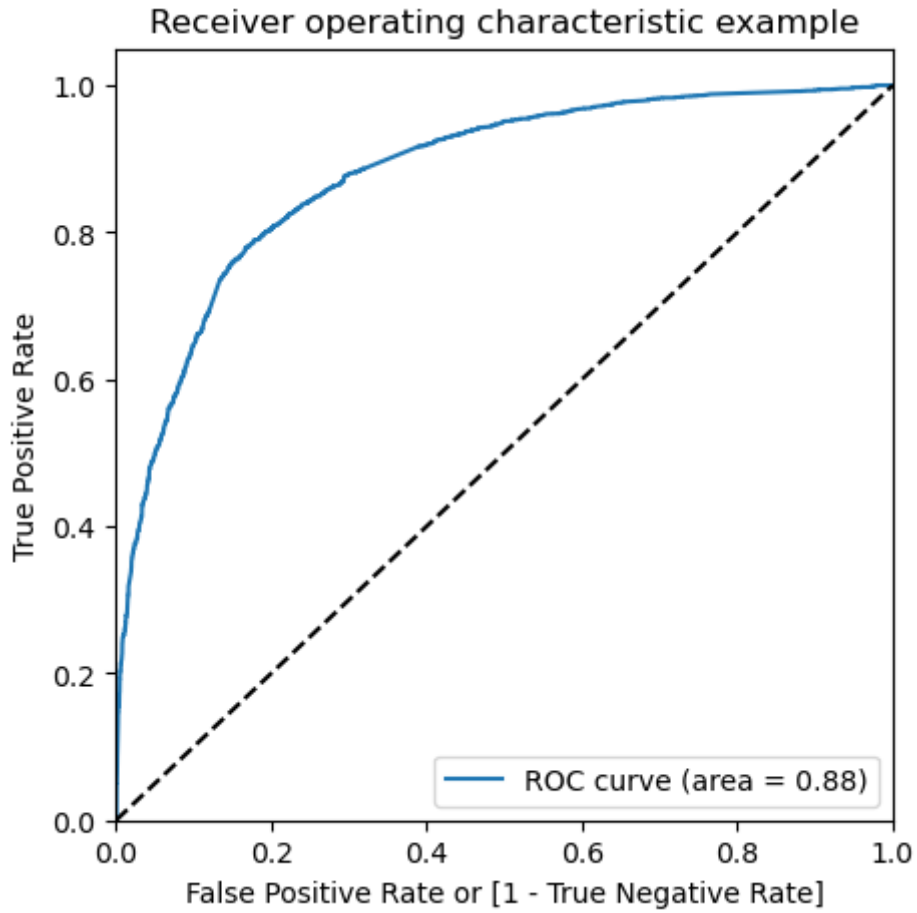
9.4 Plotting the ROC Curve

In [ ]:
```
# UDF to draw ROC curve
def draw_roc( actual, probs ):
    fpr, tpr, thresholds = metrics.roc_curve( actual, probs,
                                              drop_intermediate = False )
    auc_score = metrics.roc_auc_score( actual, probs )
    plt.figure(figsize=(5, 5))
    plt.plot( fpr, tpr, label='ROC curve (area = %0.2f)' % auc_score )
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
```

```
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()

    return None
```

In [ ]:
```
fpr, tpr, thresholds = metrics.roc_curve(y_train_pred_final["Converted"], y_train_pred_final["Converted_Prob"], drop_intermediat
```

In [ ]:
```
# Drawing ROC curve for Train Set
draw_roc(y_train_pred_final["Converted"], y_train_pred_final["Converted_Prob"])
```



Inference: Area under ROC curve is 0.88 out of 1 which indicates a good predictive model

Finding Optimal Cutoff Point/ Probability

It is that probability where we get `balanced sensitivity and specificity`

In [ ]:
```
# Creating columns with different probability cutoffs
numbers = [float(x)/10 for x in range(10)]
for i in numbers:
    y_train_pred_final[i]= y_train_pred_final['Converted_Prob'].map(lambda x: 1 if x > i else 0)
y_train_pred_final.head()
```

Out[ ]:

	Converted	Converted_Prob	Prospect ID	Predicted	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0	0	0.474082	1871	0	1	1	1	1	1	0	0	0	0	0
1	0	0.073252	6795	0	1	0	0	0	0	0	0	0	0	0
2	0	0.249087	3516	0	1	1	1	0	0	0	0	0	0	0
3	0	0.768973	8105	1	1	1	1	1	1	1	1	1	0	0
4	0	0.212973	3934	0	1	1	1	0	0	0	0	0	0	0

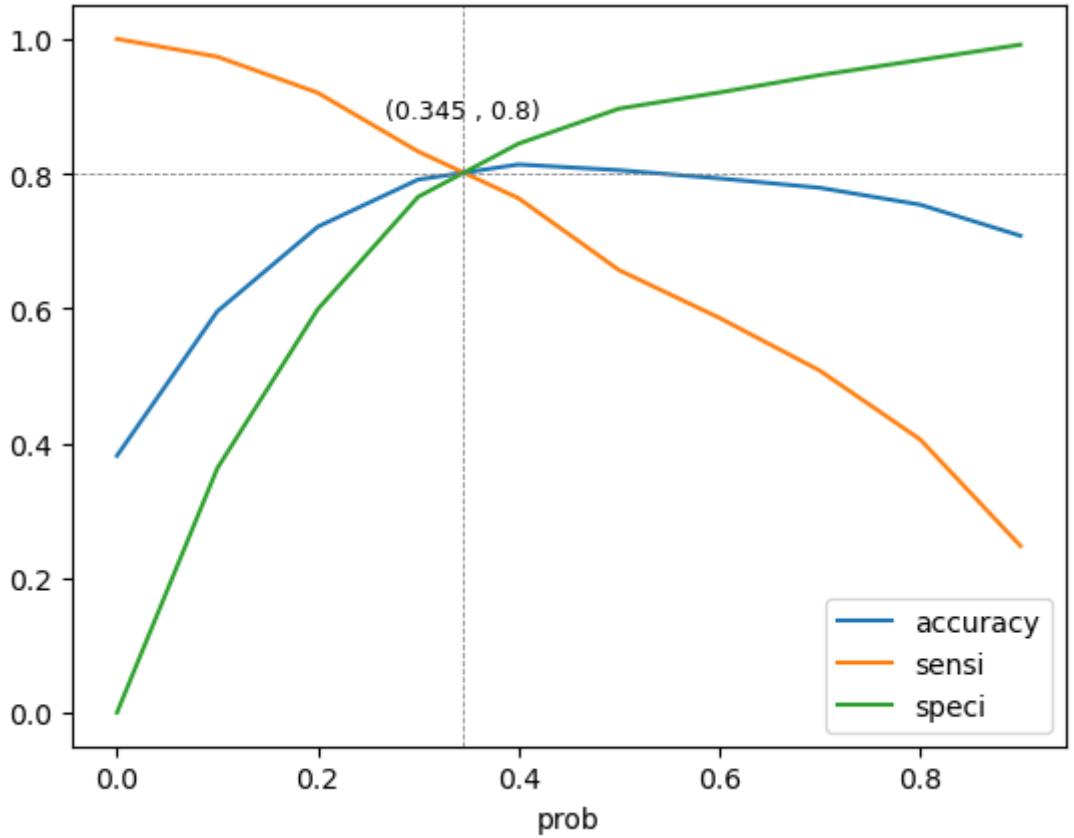
In [ ]:
```
# Calculating accuracy sensitivity and specificity for various probability cutoffs.
cutoff_df = pd.DataFrame( columns = ['prob','accuracy','sensi','speci'])
num = [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
for i in num:
    cm1 = metrics.confusion_matrix(y_train_pred_final["Converted"], y_train_pred_final[i] )
    total1=sum(sum(cm1))
    accuracy = (cm1[0,0]+cm1[1,1])/total1

    speci = cm1[0,0]/(cm1[0,0]+cm1[0,1])
    sensi = cm1[1,1]/(cm1[1,0]+cm1[1,1])
    cutoff_df.loc[i] =[ i ,accuracy,sensi,speci]
print(cutoff_df)
```

	prob	accuracy	sensi	speci
0.0	0.0	0.381262	1.000000	0.000000
0.1	0.1	0.595702	0.973642	0.362819
0.2	0.2	0.721243	0.920114	0.598701
0.3	0.3	0.791280	0.832928	0.765617
0.4	0.4	0.813698	0.763585	0.844578
0.5	0.5	0.805195	0.656934	0.896552
0.6	0.6	0.792981	0.585969	0.920540
0.7	0.7	0.779066	0.507705	0.946277
0.8	0.8	0.754020	0.405515	0.968766
0.9	0.9	0.707792	0.247364	0.991504

In [ ]:

```
# Plotting accuracy sensitivity and specificity for various probabilities.

from scipy.interpolate import interp1d
from scipy.optimize import fsolve

# Finding the intersection points of the sensitivity and accuracy curves
sensi_interp = interp1d(cutoff_df['prob'], cutoff_df['sensi'], kind='linear')
acc_interp = interp1d(cutoff_df['prob'], cutoff_df['accuracy'], kind='linear')
intersection_1 = np.round(float(fsolve(lambda x : sensi_interp(x) - acc_interp(x), 0.5)), 3)

# Finding the intersection points of the specificity and accuracy curves
speci_interp = interp1d(cutoff_df['prob'], cutoff_df['speci'], kind='linear')
intersection_2 = np.round(float(fsolve(lambda x : speci_interp(x) - acc_interp(x), 0.5)), 3)

# Calculating the average of the two intersection points
intersection_x = (intersection_1 + intersection_2) / 2

# Interpolating the accuracy, sensitivity, and specificity at the intersection point
accuracy_at_intersection = np.round(float(acc_interp(intersection_x)), 2)
sensitivity_at_intersection = np.round(float(sensi_interp(intersection_x)), 2)
specificity_at_intersection = np.round(float(speci_interp(intersection_x)), 2)

# Plotting the three curves and add vertical and horizontal lines at intersection point
cutoff_df.plot.line(x='prob', y=['accuracy', 'sensi', 'speci'])
plt.axvline(x=intersection_x, color='grey',linewidth=0.55, linestyle='--')
plt.axhline(y=accuracy_at_intersection, color='grey',linewidth=0.55, linestyle='--')

# Adding annotation to display the (x,y) intersection point coordinates
plt.annotate(f'({intersection_x} , {accuracy_at_intersection})',
             xy=(intersection_x, accuracy_at_intersection),
             xytext=(0,20),
             textcoords='offset points',
```

```
            ha='center',
            fontsize=9)

# Displaying the plot
plt.show()
```



Inference: 0.345 is the approx. point where all the curves meet, so 0.345 shall be our `Optimal cutoff point` for probability threshold.

Mapping again using optimal cutoff point

In [ ]:
```
y_train_pred_final['final_predicted'] = y_train_pred_final['Converted_Prob'].map( lambda x: 1 if x > 0.345 else 0)

# deleting the unwanted columns from dataframe
y_train_pred_final.drop([0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,"Predicted"],axis = 1, inplace = True)
y_train_pred_final.head()
```

Out[ ]:

	Converted	Converted_Prob	Prospect ID	final_predicted
0	0	0.474082	1871	1
1	0	0.073252	6795	0
2	0	0.249087	3516	0
3	0	0.768973	8105	1
4	0	0.212973	3934	0

9.5 Calculating all metrics using confusion matrix for Train

In [ ]:
```
# Checking the overall accuracy.
print(metrics.accuracy_score(y_train_pred_final["Converted"], y_train_pred_final["final_predicted"]))
```

```
0.8045763760049475
```

In [ ]:
```
# UDF for all Logistic Regression Metrics
def logreg_all_metrics(confusion_matrix):
    TN =confusion_matrix[0,0]
    TP =confusion_matrix[1,1]
    FP =confusion_matrix[0,1]
    FN =confusion_matrix[1,0]

    accuracy = (TN+TP)/(TN+TP+FN+FP)
    sensi = TP/(TP+FN)
    speci = TN/(TN+FP)
    precision = TP/(TP+FP)
    recall = TP/(TP+FN)
    TPR = TP/(TP + FN)
    TNR = TN/(TN + FP)

    #Calculating false postive conversion rate
    FPR = FP/(FP + TN)
    FNR = FN/(FN +TP)

    print ("True Negative                    : ", TN)
    print ("True Positive                    : ", TP)
    print ("False Negative                   : ", FN)
    print ("False Positve                    : ", FP)
```

```
    print ("Model Accuracy                  : ", round(accuracy,4))
    print ("Model Sensitivity               : ", round(sensi,4))
    print ("Model Specificity               : ", round(speci,4))
    print ("Model Precision                 : ", round(precision,4))
    print ("Model Recall                    : ", round(recall,4))
    print ("Model True Positive Rate (TPR)  : ", round(TPR,4))
    print ("Model False Positive Rate (FPR) : ", round(FPR,4))
```

In [ ]:

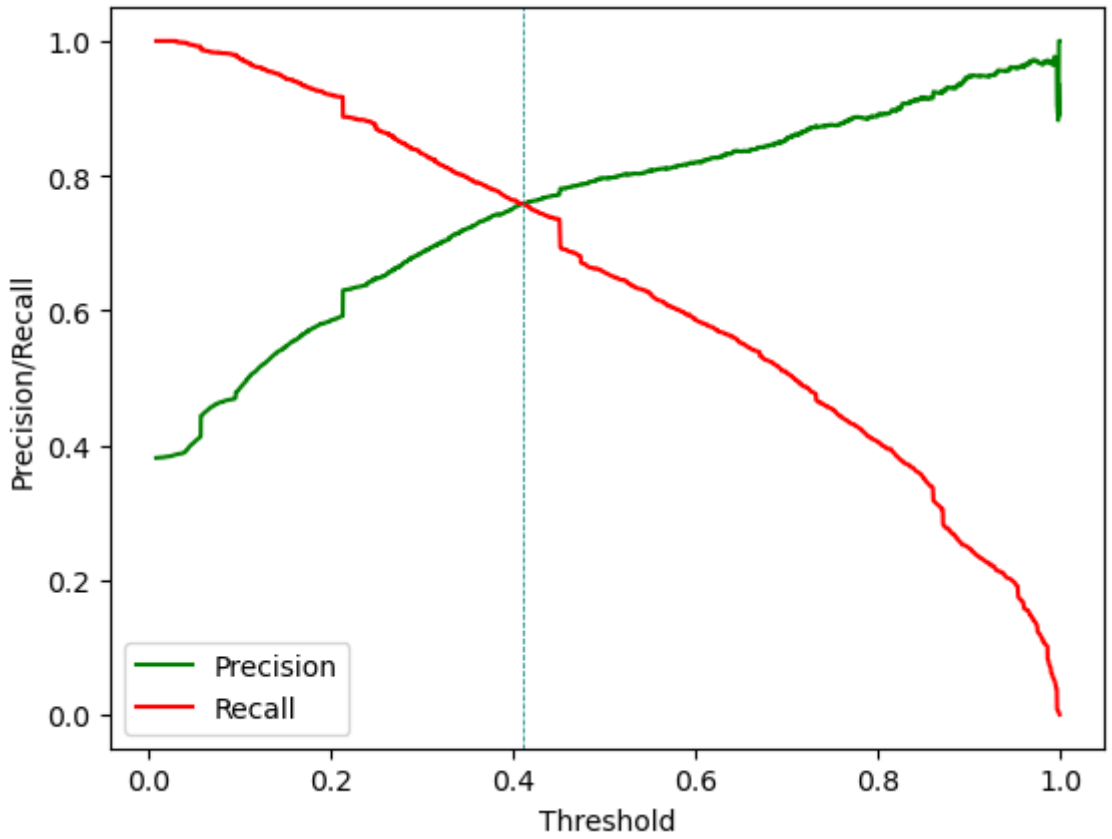
```
# Finding Confusion metrics for 'y_train_pred_final' df
confusion_matrix = metrics.confusion_matrix(y_train_pred_final['Converted'], y_train_pred_final['final_predicted'])
print("*"*50,"\n")

#
print("Confusion Matrix")
print(confusion_matrix,"\n")

print("*"*50,"\n")

# Using UDF to calculate all metrices of logistic regression
logreg_all_metrics(confusion_matrix)

print("\n")
print("*"*50,"\n")
```

```
************************************************

Confusion Matrix
[[3230  772]
 [ 492 1974]]

************************************************

True Negative                    :  3230
True Positive                    :  1974
False Negative                   :  492
False Positve                    :  772
Model Accuracy                   :  0.8046
Model Sensitivity                :  0.8005
Model Specificity                :  0.8071
Model Precision                  :  0.7189
Model Recall                     :  0.8005
Model True Positive Rate (TPR)   :  0.8005
Model False Positive Rate (FPR)  :  0.1929


************************************************
```

9.6 Precision and recall tradeoff

In [ ]:
```
# Creating precision-recall tradeoff curve
y_train_pred_final['Converted'], y_train_pred_final['final_predicted']
p, r, thresholds = precision_recall_curve(y_train_pred_final['Converted'], y_train_pred_final['Converted_Prob'])
```

In [ ]:
```
# plot precision-recall tradeoff curve
plt.plot(thresholds, p[:-1], "g-", label="Precision")
plt.plot(thresholds, r[:-1], "r-", label="Recall")

# adding legend and axis labels

plt.axvline(x=0.41, color='teal',linewidth = 0.55, linestyle='--')
plt.legend(loc='lower left')
plt.xlabel('Threshold')
plt.ylabel('Precision/Recall')

plt.show()
```



NOTE: The intersection point of the curve is the threshold value where the model achieves a balance between precision and recall. Here our probability threshold is 0.41 aprrox from above curve.

In [ ]:
```
# copying df to test model evaluation with precision recall threshold of 0.41
y_train_precision_recall = y_train_pred_final.copy()
```

In [ ]:
```
# assigning a feature for 0.41 cutoff from precision recall curve for comparison (sensi-speci or precision-recall)
y_train_precision_recall['precision_recall_prediction'] = y_train_precision_recall['Converted_Prob'].map( lambda x: 1 if x > 0.4
y_train_precision_recall.head()
```

Out[ ]:

	Converted	Converted_Prob	Prospect ID	final_predicted	precision_recall_prediction
0	0	0.474082	1871	1	1
1	0	0.073252	6795	0	0
2	0	0.249087	3516	0	0
3	0	0.768973	8105	1	1
4	0	0.212973	3934	0	0

In [ ]:

```
# Finding Confusion metrics for 'y_train_precision_recall' df
confusion_matrix = metrics.confusion_matrix(y_train_precision_recall['Converted'], y_train_precision_recall['precision_recall_pre
print("*"*50,"\n")

#
print("Confusion Matrix")
print(confusion_matrix,"\n")

print("*"*50,"\n")

# Using UDF to calculate all metrices of logistic regression
logreg_all_metrics(confusion_matrix)

print("\n")
print("*"*50,"\n")
```

```
************************************************

Confusion Matrix
[[3406  596]
 [ 596 1870]]

************************************************

True Negative                   :  3406
True Positive                   :  1870
False Negative                  :  596
False Positve                   :  596
Model Accuracy                  :  0.8157
Model Sensitivity               :  0.7583
Model Specificity               :  0.8511
Model Precision                 :  0.7583
Model Recall                    :  0.7583
Model True Positive Rate (TPR)  :  0.7583
Model False Positive Rate (FPR) :  0.1489


************************************************
```

Inference:

- As seen in the above metrics, when we use precision-recall threshold cut-off of 0.41 the values of True Positive Rate, Sensitivity, and Recall have dropped to 75%, but the business requirement is 80%.

Adding "Lead Score" Feature to Training dataframe

- A higher score would mean that the lead is hot, i.e. is most likely to convert
- A lower score would mean that the lead is cold and likely won't convert.

In [ ]:
```
# Adding the Lead Score
y_train_pred_final['Lead_Score'] = y_train_pred_final['Converted_Prob'].map( lambda x: round(x*100))
y_train_pred_final.head()
```

Out[ ]:

	Converted	Converted_Prob	Prospect ID	final_predicted	Lead_Score
0	0	0.474082	1871	1	47
1	0	0.073252	6795	0	7
2	0	0.249087	3516	0	25
3	0	0.768973	8105	1	77
4	0	0.212973	3934	0	21

10: Making Predictions on test set

10.1 Scaling Test dataset

In [ ]:
```
X_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2772 entries, 4269 to 2960
Data columns (total 46 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Do Not Email                            2772 non-null   int64
 1   TotalVisits                             2772 non-null   float64
 2   Total Time Spent on Website             2772 non-null   int64
 3   Page Views Per Visit                    2772 non-null   float64
 4   Free_copy                               2772 non-null   int64
 5   Lead Origin_Landing Page Submission     2772 non-null   uint8
 6   Lead Origin_Quick Add Form              2772 non-null   uint8
 7   Lead Source_Facebook                    2772 non-null   uint8
 8   Lead Source_Google                      2772 non-null   uint8
 9   Lead Source_Olark Chat                  2772 non-null   uint8
 10  Lead Source_Organic Search              2772 non-null   uint8
 11  Lead Source_Others                      2772 non-null   uint8
 12  Lead Source_Reference                   2772 non-null   uint8
 13  Lead Source_Referral Sites              2772 non-null   uint8
 14  Lead Source_Welingak Website            2772 non-null   uint8
 15  Last Activity_Email Bounced             2772 non-null   uint8
 16  Last Activity_Email Link Clicked        2772 non-null   uint8
 17  Last Activity_Email Opened              2772 non-null   uint8
 18  Last Activity_Form Submitted on Website 2772 non-null   uint8
 19  Last Activity_Olark Chat Conversation   2772 non-null   uint8
 20  Last Activity_Others                    2772 non-null   uint8
 21  Last Activity_Page Visited on Website   2772 non-null   uint8
 22  Last Activity_SMS Sent                  2772 non-null   uint8
 23  Specialization_Business Administration  2772 non-null   uint8
 24  Specialization_E-Business               2772 non-null   uint8
 25  Specialization_E-COMMERCE               2772 non-null   uint8
 26  Specialization_Finance Management       2772 non-null   uint8
 27  Specialization_Healthcare Management    2772 non-null   uint8
 28  Specialization_Hospitality Management   2772 non-null   uint8
 29  Specialization_Human Resource Management 2772 non-null  uint8
 30  Specialization_IT Projects Management   2772 non-null   uint8
 31  Specialization_International Business   2772 non-null   uint8
 32  Specialization_Marketing Management     2772 non-null   uint8
 33  Specialization_Media and Advertising    2772 non-null   uint8
 34  Specialization_Operations Management    2772 non-null   uint8
 35  Specialization_Others                   2772 non-null   uint8
 36  Specialization_Retail Management        2772 non-null   uint8
 37  Specialization_Rural and Agribusiness   2772 non-null   uint8
 38  Specialization_Services Excellence      2772 non-null   uint8
```

```
 39  Specialization_Supply Chain Management    2772 non-null   uint8
 40  Specialization_Travel and Tourism         2772 non-null   uint8
 41  Current_occupation_Housewife              2772 non-null   uint8
 42  Current_occupation_Other                  2772 non-null   uint8
 43  Current_occupation_Student                2772 non-null   uint8
 44  Current_occupation_Unemployed             2772 non-null   uint8
 45  Current_occupation_Working Professional   2772 non-null   uint8
dtypes: float64(2), int64(3), uint8(41)
memory usage: 240.9 KB
```

In [ ]:
```
# fetching int64 and float64 dtype columns from dataframe for scaling
num_cols=X_test.select_dtypes(include=['int64','float64']).columns

# scaling columns
X_test[num_cols] = scaler.transform(X_test[num_cols])

X_test = X_test[rfe_col]
X_test.head()
```

Out[ ]:

	Total Time Spent on Website	Lead Origin_Landing Page Submission	Lead Source_Olark Chat	Lead Source_Reference	Lead Source_Welingak Website	Last Activity_Email Opened	Last Activity_Olark Chat Conversation	Last Activity_Others	Last Activity_SMS Sent	Specializ
4269	0.964504	0	0	0	0	0	0	0	1	
2376	-0.885371	0	0	1	0	0	0	0	1	
7766	-0.777416	0	0	0	0	0	0	1	0	
9199	-0.885371	0	1	0	0	0	1	0	0	
4359	-0.885371	0	0	1	0	1	0	0	0	

10.2 Prediction on Test Dataset using final model

In [ ]:
```
# Adding contant value
X_test_sm = sm.add_constant(X_test)
X_test_sm.shape
```

Out[ ]:
```
(2772, 13)
```

In [ ]:
```
# making prediction using model 4 (final model)
y_test_pred = logm4.predict(X_test_sm)
```

In [ ]:
```
# top 10 columns
y_test_pred[:10]
```

Out[ ]:
```
4269    0.697934
2376    0.860665
7766    0.889241
9199    0.057065
4359    0.871510
9186    0.503859
1631    0.419681
8963    0.154531
8007    0.072344
5324    0.298849
dtype: float64
```

In [ ]:
```
# Changing to dataframe of predicted probability
y_test_pred = pd.DataFrame(y_test_pred)
y_test_pred.head()
```

Out[ ]:

	0
4269	0.697934
2376	0.860665
7766	0.889241
9199	0.057065
4359	0.871510

In [ ]:
```
# Converting y_test to dataframe
y_test_df = pd.DataFrame(y_test)
y_test_df.head()
```

Out[ ]:

	Converted
4269	1
2376	1
7766	1
9199	0
4359	1

In [ ]:
```
# Putting Prospect ID to index
y_test_df['Prospect ID'] = y_test_df.index

# Removing index for both dataframes to append them side by side
y_test_pred.reset_index(drop=True, inplace=True)
y_test_df.reset_index(drop=True, inplace=True)

# Appending y_test_df and y_test_pred
y_pred_final = pd.concat([y_test_df, y_test_pred],axis=1)
y_pred_final.head()
```

Out[ ]:

	Converted	Prospect ID	0
0	1	4269	0.697934
1	1	2376	0.860665
2	1	7766	0.889241
3	0	9199	0.057065
4	1	4359	0.871510

In [ ]:
```
# Renaming the column
y_pred_final= y_pred_final.rename(columns={ 0 : 'Converted_Prob'})

# Rearranging the columns
y_pred_final = y_pred_final.reindex(['Prospect ID','Converted','Converted_Prob'], axis=1)

y_pred_final.head()
```

Out[ ]:

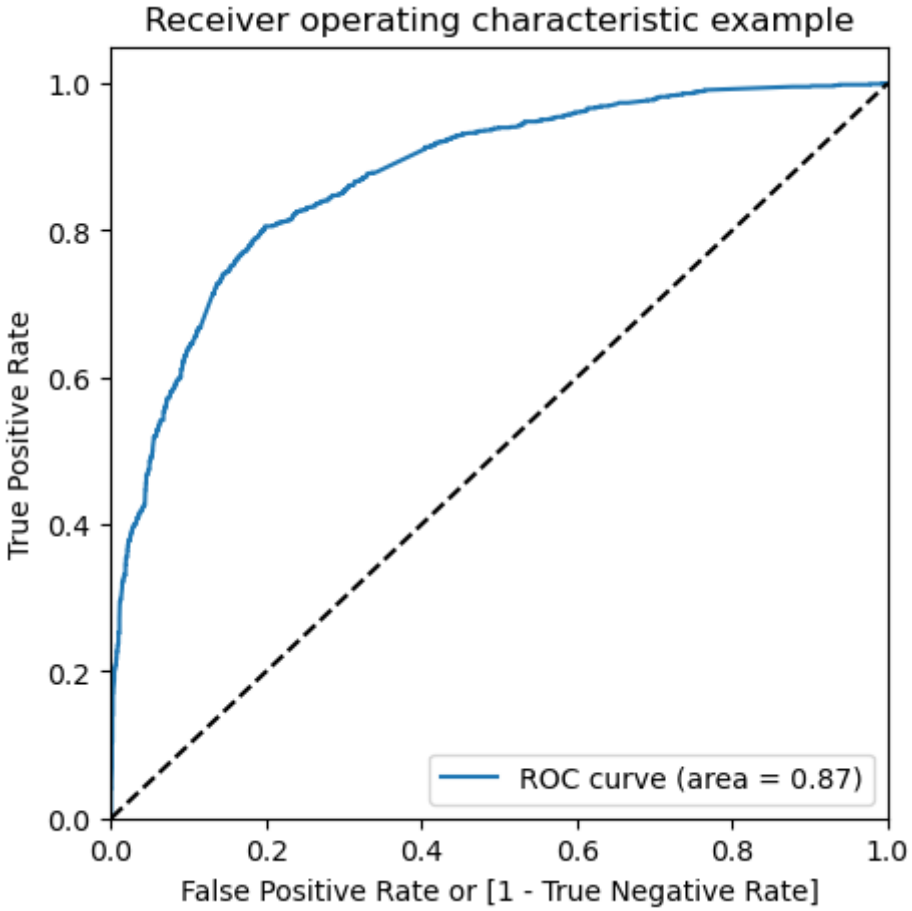
	Prospect ID	Converted	Converted_Prob
0	4269	1	0.697934
1	2376	1	0.860665
2	7766	1	0.889241
3	9199	0	0.057065
4	4359	1	0.871510

In [ ]:
```
# taking sensitivity-specificity method at 0.345 probability cutoff during training
y_pred_final['final_predicted'] = y_pred_final['Converted_Prob'].map(lambda x: 1 if x > 0.345 else 0)
y_pred_final.head()
```

Out[ ]:

	Prospect ID	Converted	Converted_Prob	final_predicted
0	4269	1	0.697934	1
1	2376	1	0.860665	1
2	7766	1	0.889241	1
3	9199	0	0.057065	0
4	4359	1	0.871510	1

In [ ]:
```
# Drawing ROC curve for Test Set
fpr, tpr, thresholds = metrics.roc_curve(y_pred_final["Converted"], y_pred_final["Converted_Prob"], drop_intermediate = False )

draw_roc(y_pred_final["Converted"], y_pred_final["Converted_Prob"])
```



Inference: Area under ROC curve is 0.87 out of 1 which indicates a good predictive model

10.3 Test set Model Evaluation

- Calculating all metrics using confusion matrix for Test set

In [ ]:
```
# Finding Confusion metrics for 'y_train_pred_final' df
confusion_matrix = metrics.confusion_matrix(y_pred_final['Converted'], y_pred_final['final_predicted'])
print("*"*50,"\n")

#
print("Confusion Matrix")
```

```
print(confusion_matrix,"\n")

print("*"*50,"\n")

# Using UDF to calculate all metrices of logistic regression
logreg_all_metrics(confusion_matrix)

print("\n")
print("*"*50,"\n")
```

```
**************************************************

Confusion Matrix
[[1353  324]
 [ 221  874]]

**************************************************

True Negative                   :  1353
True Positive                   :  874
False Negative                  :  221
False Positve                   :  324
Model Accuracy                  :  0.8034
Model Sensitivity               :  0.7982
Model Specificity               :  0.8068
Model Precision                 :  0.7295
Model Recall                    :  0.7982
Model True Positive Rate (TPR)  :  0.7982
Model False Positive Rate (FPR) :  0.1932


**************************************************
```

Inference: The evaluation matrics are pretty close to each other, indicating that the model is performing consistently across different evaluation metrics in both test and train dataset.

For Test set

- Accuracy = 80.34%
- Sensitivity = 79.82% ≈ 80%
- Specificity = 80.68%

These matrics are very close to train set, so out final model logm4 is performing with good consistency on both Train & Test set

In [ ]:
```
# features and their coefficicent from final model
parameters=logm4.params.sort_values(ascending=False)
parameters
```

Out[ ]:
```
Lead Source_Welingak Website               5.388662
Lead Source_Reference                      2.925326
Current_occupation_Working Professional    2.669665
Last Activity_SMS Sent                     2.051879
Last Activity_Others                       1.253061
Total Time Spent on Website                1.049789
Last Activity_Email Opened                 0.942099
Lead Source_Olark Chat                     0.907184
Last Activity_Olark Chat Conversation     -0.555605
const                                     -1.023594
Specialization_Hospitality Management     -1.094445
Specialization_Others                     -1.203333
Lead Origin_Landing Page Submission       -1.258954
dtype: float64
```

Insights: A high positive coefficient indicates that a variable has a stronger influence on predicting the probability of leads converting to take up X-Education's course.

Adding "Lead Score" Feature to Test dataframe

- A higher score means that the lead is hot, i.e. is most likely to convert
- A lower score means that the lead is cold and won't likely convert.

In [ ]:
```
# Adding the Lead Score

y_pred_final['Lead_Score'] = y_pred_final['Converted_Prob'].map( lambda x: round(x*100))
y_pred_final.head()
```

Out[ ]:

	Prospect ID	Converted	Converted_Prob	final_predicted	Lead_Score
0	4269	1	0.697934	1	70
1	2376	1	0.860665	1	86
2	7766	1	0.889241	1	89
3	9199	0	0.057065	0	6
4	4359	1	0.871510	1	87

Conclusions

Train - Test

Train Data Set:

- **Accuracy:** 80.46%
- **Sensitivity:** 80.05%
- **Specificity:** 80.71%

Test Data Set:

- **Accuracy:** 80.34%
- **Sensitivity:** 79.82% ≈ 80%

- **Specificity:** 80.68%

- The model achieved a sensitivity of 80.05% in the train set and 79.82% in the test set, using a cut-off value of 0.345.
- Sensitivity in this case indicates the accuracy of the model for each customer.
- The model also achieved an accuracy of 80.46%, which is in line with the study's objectives.

NOTE: The Optimal cutoff probability point is 0.345. Converted probability greater than 0.345 will be predicted as Converted lead (Hot lead) & probability smaller than 0.345 will be predicted as not Converted lead (Cold lead).

Recommendations

To increase Lead Conversion Rates:

- Allocate more budget for advertisement expenditure on Welingak website.
- Come up with a referral program for existing customers and incentivize them
- Optimize the product to suit the requirements of working professionals. Engage them with tailored messaging.
- Optimize communication channels such as SMS and email.
- Make sure customers spend more time on the website. This can be done by optimizing the UI/UX.
- Leverage email marketing.

To identify areas of improvement:

- Optimize content for specializations since not many leads are converting in specific specialization programs.

- Use Olark Chat only as the initial touch point. After that, shift the primary communication channel to SMS or WhatsApp (since it is easier to send multimedia messages, and make customer see the company logo regularly).