Facebook Campaign marketing performance:

Author: Arpit Khandekar

Problem Statement

Q1. What is the cost per lead, cost per appointment, and ROAS by week, month, and all time?

```
# Author: Arpit Khandekar
```

```
# installing plotly and cufflinks libraries using pip command.
import sys
!{sys.executable} -m pip install cufflinks
!{sys.executable} -m pip install Plotly
```

```
# importing packages
from pandas import DataFrame, read_csv
import matplotlib.pyplot as plt
import seaborn as sns
from plotly import __version__
%matplotlib inline
import pandas as pd
# Plotly helps in building interactive visualization.
#Cufflinks directly binds plotly to Pandas dataframe.
import cufflinks as cf
from plotly.offline import download_plotlyjs,init_notebook_mode,plot,iplot
# It saves the data visualization even if it crashes.
init_notebook_mode(connected=True)
cf.go_offline()
```

Importing data to find the required values

We have Facebook campaign marketing performance data.

```
#importing the Facebook marketing data
# reading the csv file
def import_data():
    file = r'F:\Arpit\Arpit_Stuff1\Full_Time_Prep\Sundae DA position\sundaecasestudy\sundae_data_challenge_531.csv'
    Fbdata_df = pd.read_csv(file)
    return Fbdata_df
```

```
Fbmarket_df = import_data()
```

```
Fbmarket_df.head()
```

	Unnamed: 0	date	spend	clicks	impressions	ctr	market	age	creative_type	creative_name	prospecting_or_retargeting	leads	campaign_id
0	1	5/30/2019 0:00	0.834390	0	2	0.000000	Inland Empire	18-65+	Video	Skip The Repairs- Pink Bathroom Video	Retargeting	0	1
1	2	5/30/2019 0:00	0.252471	0	2	0.000000	NaN	NaN	NaN	NaN	NaN	0	2
2	3	5/30/2019 0:00	0.136869	0	2	0.000000	NaN	NaN	NaN	NaN	NaN	0	3
3	4	5/30/2019 0:00	0.038948	0	1	0.000000	San Diego	18-65+	Video	Skip The Repairs- Pink Bathroom Video	Retargeting	0	4
4	5	5/29/2019 0:00	89.279997	34	1918	0.017727	San Diego	55-65+	Static Image	Worn Wooden Stairs-How Much\n	Prospecting	2	5

```
#  to find the number of rows and columns
Fbmarket_df.shape
```

```
(2550, 19)
```

Data preprocessing

#We are filtering the data and dropping the unnessecary coloumns.
As it is given that blank values we don't have access to data hence dropping those data is the right choice for our analysis.
Aslo dropping the unnamed coloumns which are not useful for our analysis.
Making sure that proper data type is used for every coloumn.

Check Null Values in the data:

```
# to check null values
Fbmarket_df.isnull().sum()
```

```
Unnamed: 0                    0
date                          0
spend                         0
clicks                        0
impressions                   0
ctr                           0
market                      155
age                         155
creative_type               201
creative_name               201
prospecting_or_retargeting  201
leads                         0
campaign_id                   0
appointments                  0
contracts                     0
revenue                       0
ad_id                         0
Unnamed: 17                   0
Unnamed: 18                2549
dtype: int64
```

Dropping Unneccessary coloumns

```
# To find and dropping Unnecessary columns
Fbmarket_df.drop(["market","age","creative_type","creative_name","prospecting_or_retargeting","clicks","impressions","ctr",
                  "contracts","ad_id"],axis=1,inplace=True)
```

```
Fbmarket_df.drop(Fbmarket_df.columns[Fbmarket_df.columns.str.contains('unnamed',case = False)],axis = 1, inplace = True)
```

```
Fbmarket_df.head()
```

	date	spend	leads	campaign_id	appointments	revenue
0	5/30/2019 0:00	0.834390	0	1	0	0.0
1	5/30/2019 0:00	0.252471	0	2	0	0.0
2	5/30/2019 0:00	0.136869	0	3	0	0.0
3	5/30/2019 0:00	0.038948	0	4	0	0.0
4	5/29/2019 0:00	89.279997	2	5	0	0.0

Deriving New Fields for our analysis:

Using given formula, we derived new field
Cost per lead: spend / number of leads

```
#Defining new coloumn 'Cost per lead' can be calculated using formula spend divided by No. of leads
Fbmarket_df['Cost_per_lead'] =  Fbmarket_df['spend'] / Fbmarket_df['leads']
Fbmarket_df.head()
```

	date	spend	leads	campaign_id	appointments	revenue	Cost_per_lead
0	5/30/2019 0:00	0.834390	0	1	0	0.0	inf
1	5/30/2019 0:00	0.252471	0	2	0	0.0	inf
2	5/30/2019 0:00	0.136869	0	3	0	0.0	inf
3	5/30/2019 0:00	0.038948	0	4	0	0.0	inf
4	5/29/2019 0:00	89.279997	2	5	0	0.0	44.639999

Using given formula, we derived new field
Cost per appointment: spend / number of appointments

```
#Defining new coloumn 'Cost_per_appointment' can be calculated using formula spend divided by No. of appointments
Fbmarket_df['Cost_per_appointment'] =  Fbmarket_df['spend'] /Fbmarket_df['appointments']
Fbmarket_df.head()
```

	date	spend	leads	campaign_id	appointments	revenue	Cost_per_lead	Cost_per_appointment
0	5/30/2019 0:00	0.834390	0	1	0	0.0	inf	inf
1	5/30/2019 0:00	0.252471	0	2	0	0.0	inf	inf
2	5/30/2019 0:00	0.136869	0	3	0	0.0	inf	inf
3	5/30/2019 0:00	0.038948	0	4	0	0.0	inf	inf
4	5/29/2019 0:00	89.279997	2	5	0	0.0	44.639999	inf

Using given formula, we derived new field
Return on ad spend (ROAS): revenue / spend

```
#Defining new coloumn 'ROAS'(return on ad spend) can be calculated using formula revenue divided by spend
Fbmarket_df['ROAS'] = Fbmarket_df['revenue'] / Fbmarket_df['spend']
Fbmarket_df.head()
```

	date	spend	leads	campaign_id	appointments	revenue	Cost_per_lead	Cost_per_appointment	ROAS
0	5/30/2019 0:00	0.834390	0	1	0	0.0	inf	inf	0.0
1	5/30/2019 0:00	0.252471	0	2	0	0.0	inf	inf	0.0
2	5/30/2019 0:00	0.136869	0	3	0	0.0	inf	inf	0.0
3	5/30/2019 0:00	0.038948	0	4	0	0.0	inf	inf	0.0
4	5/29/2019 0:00	89.279997	2	5	0	0.0	44.639999	inf	0.0

Now as found that we have infinite values in our data after calculation field.
We need to preprocess this data to get Cost per lead, Cost per appointment weekly and monthly.

```
# making sure the right data type is used for date coloumn
Fbmarket_df['date'] = pd.to_datetime(Fbmarket_df['date'])
```

```
# setting data frame index
Fbmarket_df = Fbmarket_df.set_index(Fbmarket_df.date)
```

```
# To check the elements
Fbmarket_df.head()
```

	date	spend	leads	campaign_id	appointments	revenue	Cost_per_lead	Cost_per_appointment	ROAS
date									
2019-05-30	2019-05-30	0.834390	0	1	0	0.0	inf	inf	0.0
2019-05-30	2019-05-30	0.252471	0	2	0	0.0	inf	inf	0.0
2019-05-30	2019-05-30	0.136869	0	3	0	0.0	inf	inf	0.0
2019-05-30	2019-05-30	0.038948	0	4	0	0.0	inf	inf	0.0
2019-05-29	2019-05-29	89.279997	2	5	0	0.0	44.639999	inf	0.0

Now we will convert inf value to zero as, we cannot use inf values for summation to find Cost per lead and Cost per appointment weekly and monthly. We will use numpy for this

```
new_Fbmarket_df=Fbmarket_df.replace(np.inf, '0')
```

```
new_Fbmarket_df.head()
```

	date	spend	leads	campaign_id	appointments	revenue	Cost_per_lead	Cost_per_appointment	ROAS
date									
2019-05-30	2019-05-30	0.834390	0	1	0	0.0	0	0	0.0
2019-05-30	2019-05-30	0.252471	0	2	0	0.0	0	0	0.0
2019-05-30	2019-05-30	0.136869	0	3	0	0.0	0	0	0.0
2019-05-30	2019-05-30	0.038948	0	4	0	0.0	0	0	0.0
2019-05-29	2019-05-29	89.279997	2	5	0	0.0	44.64	0	0.0

Putting Cost per lead values in one dataframe

```
new_Fbmarket_df['date'] = pd.to_datetime(new_Fbmarket_df['date'])
```

```
new_Fbmarket_df = new_Fbmarket_df.set_index(new_Fbmarket_df.date)
```

```
CPL_Fbmarket_df= new_Fbmarket_df['Cost_per_lead'].astype('str')
```

```
#CPL_Fbmarket_df = new_Fbmarket_df
```

```
CPL_Fbmarket_df.columns = ['Cost_per_lead']
```

```
CPL_Fbmarket_df
```

```
date
2019-05-30                    0
2019-05-30                    0
2019-05-30                    0
2019-05-30                    0
2019-05-29           44.63999852
2019-05-29           28.81916749
2019-05-29                    0
2019-05-28    56.959459200000005
2019-05-28                    0
2019-05-28                    0
2019-05-28            5.11326081
2019-05-27     58.21303204999999
2019-05-27                    0
2019-05-27                    0
2019-05-27                    0
2019-05-26    52.547048950000004
2019-05-26                    0
2019-05-26                    0
```

Calculating Cost per lead on weekly basis:

```
# Deriving Cost per lead on weekly basis.
CPLweekly=CPL_Fbmarket_df.resample("W").sum()
```

```
date
2019-04-14    17.6400
2019-04-21    51.8768
2019-04-28    36.5839
2019-05-05    13.6317
2019-05-12    92.4235
2019-05-19    65.4765
2019-05-26    54.9855
2019-06-02    58.2130
Freq: W-SUN,
```

Calculating Cost per Appointment on weekly basis:

```
CPA_Fbmarket_df= new_Fbmarket_df['Cost_per_appointment'].astype('str')
```

```
CPA_Fbmarket_df.columns = ['Cost_per_appointment']
```

```
CPA_Fbmarket_df
```

```
date
2019-05-30                   0
2019-05-30                   0
2019-05-30                   0
2019-05-30                   0
2019-05-29                   0
2019-05-29                   0
2019-05-29                   0
2019-05-28    113.91891840000001
2019-05-28                   0
2019-05-28                   0
2019-05-28            5.11326081
2019-05-27    232.85212819999995
2019-05-27                   0
2019-05-27                   0
2019-05-27                   0
2019-05-26    105.09409790000001
2019-05-26                   0
2019-05-26                   0
```

```
CPAweekly=CPA_Fbmarket_df.resample("W").sum()
```

```
print(CPAweekly)
```

```
date
2019-04-14    69.7278
2019-04-21    62.3860
2019-04-28    73.1679
2019-05-05    27.2635
2019-05-12   184.8471
2019-05-19    66.4925
2019-05-26   109.9710
2019-06-02   232.8521
Freq: W-SUN,
```

Calculating Return on ad spend on weekly basis:

```
ROASweekly=Fbmarket_df['ROAS'].resample("W").sum()
```

```
print(ROASweekly)
```

```
date
2019-04-14    3572.092575
2019-04-21     657.088007
2019-04-28    5323.928088
2019-05-05    6029.500461
2019-05-12    9677.926611
2019-05-19    4372.343985
2019-05-26    8356.561236
2019-06-02     783.819231
Freq: W-SUN, Name: ROAS, dtype: float64
```

Calculating Cost per lead on Monthly basis:

```
CPLmonly=CPL_Fbmarket_df.resample("M").sum()
print(CPLmonly)
```

```
date
2019-04-30    17.6400
2019-05-31    83.3459
Freq: M, Name: Cost_per_lead,
```

Calculating Cost per Appointment on Monthly basis:

```
CPAmonly=CPA_Fbmarket_df.resample("M").sum()
print(CPAmonly)
```

```
date
2019-04-30    69.7278
2019-05-31    250.0379
Freq: M, Name: Cost_per_lead,
```

Calculating Return on ad spend on Monthly basis:

```
ROASmonly=Fbmarket_df['ROAS'].resample("M").sum()
```

```
print(ROASmonly)
```

```
date
2019-04-30    12478.929559
2019-05-31    26294.330635
Freq: M, Name: ROAS, dtype: float64
```

Calculating Cost per lead on All time basis:

```
CPLalltym=Fbmarket_df['Cost_per_lead'].sum()
```

```
print(CPLalltym)
```

```
54.189493607579024
```

Calculating Cost per Appointment on All time basis:

```
CPAalltym=Fbmarket_df['Cost_per_appointment'].sum()
```

```
print(CPAalltym)
```

```
17.328460120551362
```

Calculating Return on ad spend on All time basis:

```
ROASalltym=Fbmarket_df['ROAS'].sum()
```

```
print(ROASalltym)
```

```
38773.26019453754
```

Q2. Plot the weekly spend, weekly cost per lead, and weekly CTR for the 5 campaigns with the highest total spend.

Solution:

Importing packages:

In [41]:
```
# importing packages
from pandas import DataFrame, read_csv
import matplotlib.pyplot as plt
import seaborn as sns
from plotly import __version__
%matplotlib inline
import pandas as pd
import numpy as np
# Plotly helps in building interactive visualization.
#Cufflinks directly binds plotly to Pandas dataframe.
import cufflinks as cf
from plotly.offline import download_plotlyjs,init_notebook_mode,plot,iplot
# It saves the data visualization even if it crashes.
init_notebook_mode(connected=True)
cf.go_offline()
```

:
```
#importing the Facebook marketing data
# reading the csv file
def import_data():
    file = r'F:\Arpit\Arpit_Stuff1\Full_Time_Prep\Sundae DA position\sundaecasestudy\sundae_data_challenge_531.csv'
    Fbdata_df = pd.read_csv(file)
    return Fbdata_df
```

```
Fbmarket_df = import_data()
```

```
Fbmarket_df.head()
```

	date	spend	ctr	leads	campaign_id	appointments	revenue
date							
2019-05-30	2019-05-30	0.834390	0.000000	0	1	0	0.0
2019-05-30	2019-05-30	0.252471	0.000000	0	2	0	0.0
2019-05-30	2019-05-30	0.136869	0.000000	0	3	0	0.0
2019-05-30	2019-05-30	0.038948	0.000000	0	4	0	0.0
2019-05-29	2019-05-29	89.279997	0.017727	2	5	0	0.0

Again, dropping the unnecessary columns from the dataset.

```
# To find and dropping Unnecessary columns
Fbmarket_df.drop(["market","age","creative_type","creative_name","prospecting_or_retargeting","clicks","impressions",
                  "contracts","ad_id"],axis=1,inplace=True)
```

```
Fbmarket_df.drop(Fbmarket_df.columns[Fbmarket_df.columns.str.contains('unnamed',case = False)],axis = 1, inplace = True)
```

```
# to check null values
Fbmarket_df.isnull().sum()
```

```
date            0
spend           0
ctr             0
leads           0
campaign_id     0
appointments    0
revenue         0
dtype: int64
```

Using the correct Datatype

```
# making sure the right data type is used for date coloumn
Fbmarket_df['date'] = pd.to_datetime(Fbmarket_df['date'])
```

```
# To check what columns are present in the dataset
df=Fbmarket_df
df.columns
```

```
Index(['date', 'spend', 'ctr', 'leads', 'campaign_id', 'appointments',
       'revenue'],
      dtype='object')
```

Setting an Index

```
# setting data frame index
Fbmarket_df = Fbmarket_df.set_index(Fbmarket_df.date)
```

```
df=Fbmarket_df
```

```
# To check the values
df.head()
```

	date	spend	ctr	leads	campaign_id	appointments	revenue
date							
2019-05-30	2019-05-30	0.834390	0.000000	0	1	0	0.0
2019-05-30	2019-05-30	0.252471	0.000000	0	2	0	0.0
2019-05-30	2019-05-30	0.136869	0.000000	0	3	0	0.0
2019-05-30	2019-05-30	0.038948	0.000000	0	4	0	0.0
2019-05-29	2019-05-29	89.279997	0.017727	2	5	0	0.0

Defining the function which will help us in resample the data on weekly basis.

```
# Taking data weekly using resample function and defining the function for it.
campaing_list=list(Fbmarket_df['campaign_id'].unique())
def prepareWeeklyData():
    dataset=[]
    for camp in campaing_list:
        df_cp_id=df.loc[df['campaign_id']==camp]
        df2=df_cp_id.resample('W').sum()
        df2.drop(['campaign_id'],axis=1,inplace=True)
        df2['campaign_id']=camp
        #df2.columns=['']*len(df2.columns)
        #print(df2)

        dataset.extend(df2.values.tolist())
    return dataset
```

Writing the data to csv file

```
# Writing the output data to csv file.
to_file=prepareWeeklyData()
to_df=pd.DataFrame(to_file)
to_df.to_csv("Datatest.csv",encoding='utf-8',index=False)
```

Importing the data from csv file to new data frame which is groupby Campaign_id

```
# defining the function to import the data
def import_newdata():
    file = r'F:\Arpit\Arpit_Stuff1\Full_Time_Prep\Sundae DA position\sundaecasestudy\Datatest.csv'
    newdata_df = pd.read_csv(file)
    return newdata_df
```

```
# transforming the imported data to dataframe
TopCamp_df = import_newdata()
```

```
# To check elements
TopCamp_df.head()
```

	spend	ctr	leads	appointments	revenue	campaign_id
0	39.197241	0.091230	1	1	0.0	1
1	255.529100	0.530257	4	1	0.0	1
2	213.315592	0.106433	3	0	0.0	1
3	192.807072	0.266703	8	2	0.0	1
4	417.136609	0.186483	6	1	0.0	1

Defining the function which will help in calculating the total spend based for each campaign

```
# Defined Funnction which will help in calculating total spend based for each campaign
def compareToCampaingId(Freq='W',target='spend'):
    global campain_totalSpend
    campain_totalSpend={}
    for camp in campaing_list:

        totalspend=TopCamp_df.loc[TopCamp_df['campaign_id']==camp][target].sum()
        #print(Fbmrkt_wkly.loc[Fbmrkt_wkly['campaign_id']==camp][target])
        campain_totalSpend[camp]=totalspend
    return campain_totalSpend
```

Showing the sum of spends based on each campaign.

```
# function which shows summation of total spend based on each campaign id
compareToCampaingId()
```

```
.
{1: 1580.79180262,
 2: 1071.6250653,
 3: 481.70258550000005,
 4: 1089.7617327799999,
 5: 5142.6571091999995,
 6: 3927.964277,
 7: 4416.3978621,
 8: 3516.1849899000003,
 9: 2892.11908287,
 10: 1116.0224581500001,
 11: 2293.8667688,
 12: 5717.6203289000005,
 13: 3007.6579322000002,
 14: 2292.4559905999995,
 15: 1584.59406055,
 16: 6954.89323,
 17: 4001.1205969000002,
 18: 2730.3160705200003,
 19: 8717.8692398,
 20: 2357.6640589000003,
 21: 2013.1615281599998,
 22: 4635.888537,
 23: 12285.774126,
 24: 2510.0193476000004,
 25: 12257.344328,
 26: 1561.97160855,
 27: 8275.8374454,
 28: 625.3776723999999,
 29: 362.15060439000007,
 30: 0.14353712799999999,
 31: 631.3371963,
 32: 2248.9751883400004,
 33: 0.590824879,
 34: 1020.9818218,
 35: 2620.1447157999996,
```

```
36: 3504.2818380000003,
37: 168.55953276999998,
38: 895.83253825,
39: 1836.2848020000001,
40: 1184.884518,
41: 1468.351801}
```

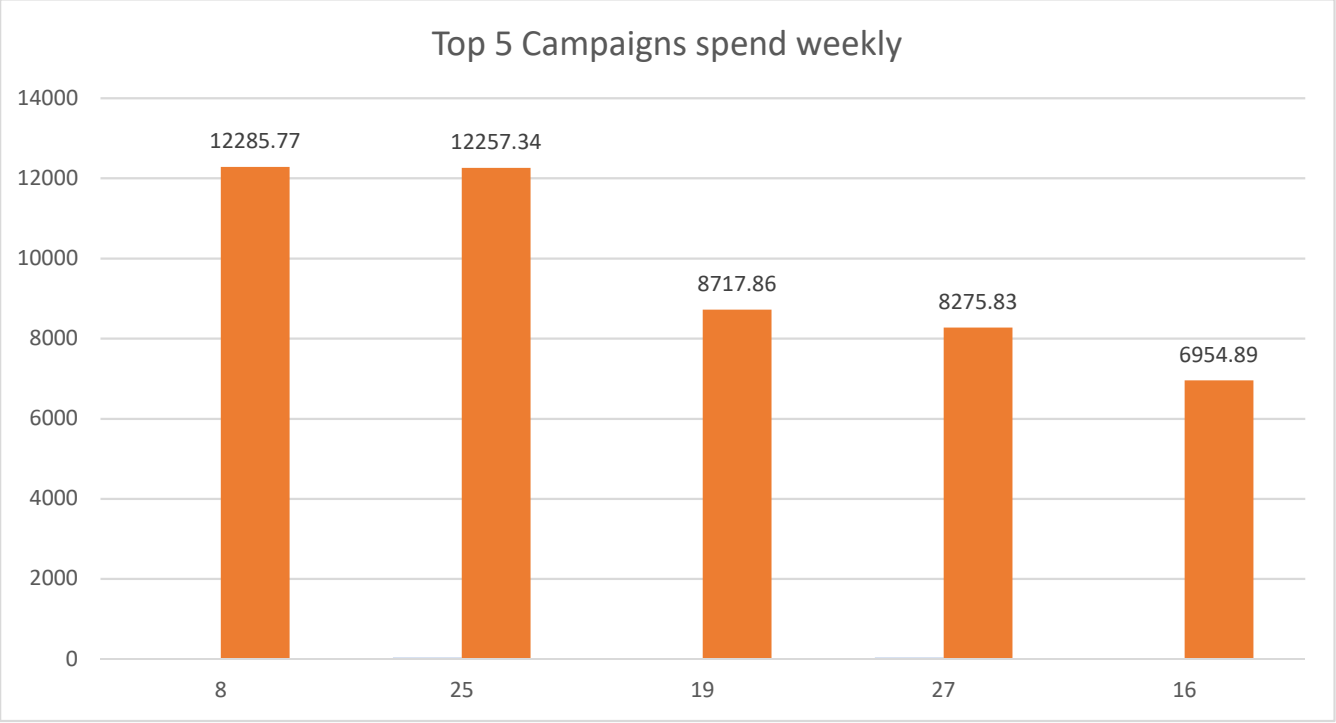
Defining Lambda function to derive top 5 campaign for weekly spend.

```
# To find top 5 campaign with highest total spend for weekly spend
Top5camp_rev=sorted(campain_totalSpend.items(),key=lambda v:v[1],reverse=True)[:5]
```

```
# display Total spend for top 5 campaign for weekly spend.
print (Top5camp_rev[:])
```

```
[(23, 12285.774126), (25, 12257.344328), (19, 8717.8692398), (27, 8275.8374454), (16, 6954.89323)]
```

Plotting Top 5 Campaign on the basis of Spend weekly



Now, we will find top 5 campaign on the basis on weekly CTR using the same function, just pass target value as CTR.

```
# To find top 5 campaign for weekly CTR having top 5 campaign using the same function with different arguments
compareToCampaingId(target='ctr')
```

```
{1: 1.530466248,
 2: 0.093957079,
 3: 0.067916763,
 4: 1.6279540890000002,
 5: 1.6846821459999999,
 6: 0.259376955,
 7: 1.811721331,
 8: 3.215581037,
 9: 0.956793757,
 10: 0.420877106,
 11: 0.23147011899999997,
 12: 1.583342681,
 13: 0.270753376,
 14: 0.376836603,
 15: 1.198769504,
 16: 2.077529763,
 17: 0.590321761,
 18: 1.01225103,
 19: 0.9392684169999999,
 20: 2.269281121,
 21: 0.9696697160000001,
 22: 0.633797416,
 23: 0.9450646549999999,
 24: 0.47656921899999993,
 25: 1.102917317,
 26: 1.178369545,
 27: 0.9880285520000001,
 28: 0.23093813700000002,
 29: 0.344982879,
 30: 0.5,
 31: 0.288208389,
 32: 0.333493174,
 33: 0.0,
 34: 0.262335983,
 35: 0.14248981700000002,
 36: 0.101095188,
 37: 0.099369245,
 38: 0.19206347,
 39: 0.45435677799999996,
 40: 0.388416799,
41: 0.05248407700000004}
```

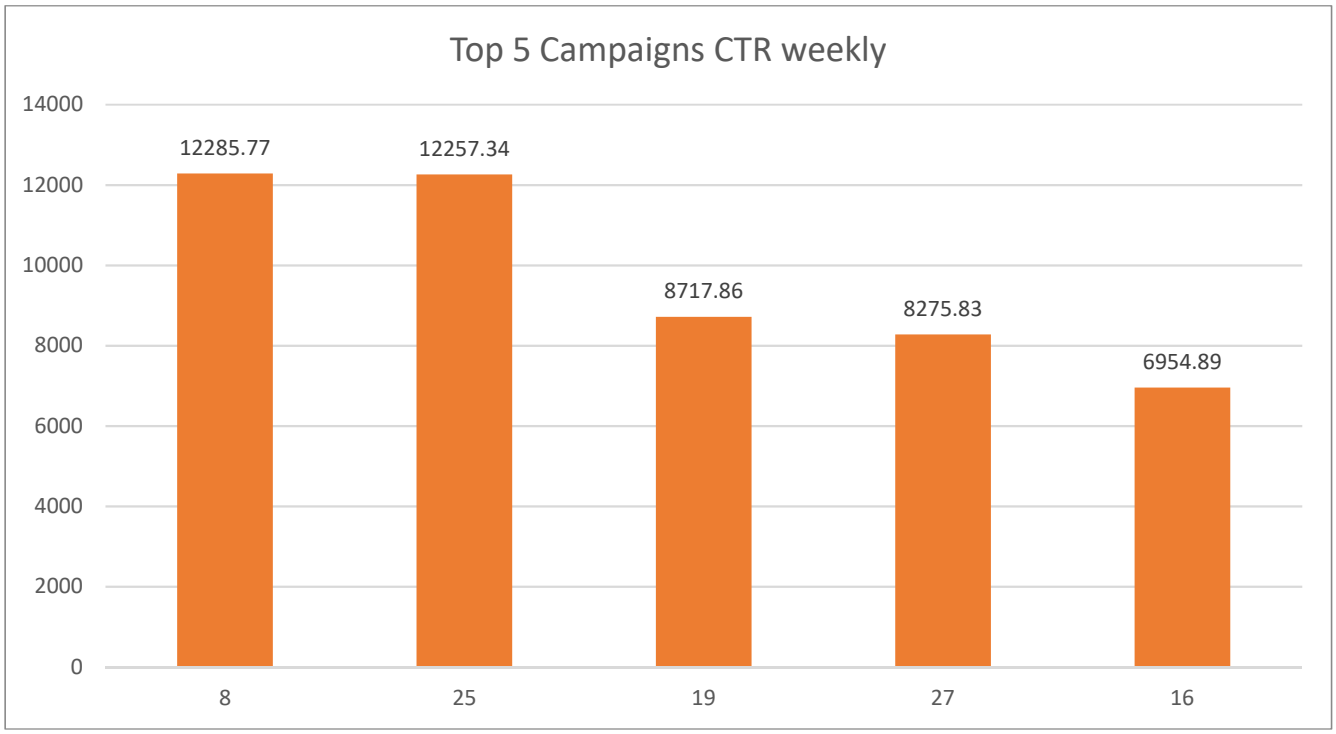
Now, we will find top 5 campaign CTR weekly.

```
# To find top 5 campaign on the basis of weekly ctr
Top5camp_ctr=sorted(campain_totalSpend.items(),key=lambda v:v[1],reverse=True)[:5]
```

```
# display values for top 5 campaign on the basis of weekly ctr
print (Top5camp_ctr[:])
```

```
[(8, 3.215581037), (20, 2.269281121), (16, 2.077529763), (7, 1.811721331), (5, 1.6846821459999999)]
```

Plotting Top 5 Campaign on the basis of CTR weekly



Q3. What recommendations would you provide the marketing team in order to optimize spend?

Solution:

First, we will find the correlation between variable using correlation matrix. Correlation values equal to 1 means having strong positive correlation whereas Correlation values equal to -1 means having strong negative correlation values.
In addition, correlation value having zero means there is not correlation between variables.

```
# Copying the raw data from csv file using import_data function
predict_df = import_data()
```

```
# Checking the null values
predict_df.isnull().sum()
```

```
Unnamed: 0                    0
date                          0
spend                         0
clicks                        0
impressions                   0
ctr                           0
market                      155
age                         155
creative_type               201
creative_name               201
prospecting_or_retargeting  201
leads                         0
campaign_id                   0
appointments                  0
contracts                     0
revenue                       0
ad_id                         0
Unnamed: 17                   0
Unnamed: 18                2549
dtype: int64
```

```
# Making sure we are not taking null values
predict_df = predict_df[predict_df['market'].notnull() & predict_df['age'].notnull() &
                        predict_df['prospecting_or_retargeting'].notnull()  & predict_df['creative_type'].notnull()]
```

```
# To check non-null elements
predict_df
```

	date	spend	clicks	impressions	ctr	market	age	creative_type	creative_name	prospecting_or_retargeting	leads	campaign_id	appo
0	5/30/2019 0:00	0.834390	0	2	0.000000	Inland Empire	18-65+	Video	Skip The Repairs- Pink Bathroom Video	Retargeting	0	1	
3	5/30/2019 0:00	0.038948	0	1	0.000000	San Diego	18-65+	Video	Skip The Repairs- Pink Bathroom Video	Retargeting	0	4	
4	5/29/2019 0:00	89.279997	34	1918	0.017727	San Diego	55-65+	Static Image	Worn Wooden Stairs-How Much\n	Prospecting	2	5	
5	5/29/2019 0:00	28.819167	5	572	0.008741	Inland Empire	25-65+	Video	Adjective - With Photos Video	Prospecting	1	6	
6	5/29/2019 0:00	8.775672	6	96	0.062500	Inland Empire	18-65+	Video	Skip The Repairs- Pink Bathroom Video	Retargeting	0	7	

```
# To drop unname coloumns
predict_df.drop(predict_df.columns[predict_df.columns.str.contains('unnamed',case = False)],axis = 1, inplace = True)
```

```
# To check null present in the dataframe
predict_df.isnull().sum()
```

```
date                        0
spend                       0
clicks                      0
impressions                 0
ctr                         0
market                      0
age                         0
creative_type               0
creative_name               0
prospecting_or_retargeting  0
leads                       0
campaign_id                 0
appointments                0
contracts                   0
revenue                     0
ad_id                       0
dtype: int64
```

```
# To get the summary of data
predict_df.head()
```

	date	spend	clicks	impressions	ctr	market	age	creative_type	creative_name	prospecting_or_retargeting	leads	campaign_id	appointmen
0	5/30/2019 0:00	0.834390	0	2	0.000000	Inland Empire	18-65+	Video	Skip The Repairs- Pink Bathroom Video	Retargeting	0	1	
3	5/30/2019 0:00	0.038948	0	1	0.000000	San Diego	18-65+	Video	Skip The Repairs- Pink Bathroom Video	Retargeting	0	4	
4	5/29/2019 0:00	89.279997	34	1918	0.017727	San Diego	55-65+	Static Image	Worn Wooden Stairs-How Much\n	Prospecting	2	5	
5	5/29/2019 0:00	28.819167	5	572	0.008741	Inland Empire	25-65+	Video	Adjective - With Photos Video	Prospecting	1	6	
6	5/29/2019 0:00	8.775672	6	96	0.062500	Inland Empire	18-65+	Video	Skip The Repairs- Pink Bathroom Video	Retargeting	0	7	

```
# To find and dropping Unnecessary columns

predict_df.drop(["age","creative_name"],axis=1,inplace=True)
```

```
predict_df.head()
```

	date	spend	clicks	impressions	ctr	market	creative_type	prospecting_or_retargeting	leads	campaign_id	appointments	contracts	revenu
0	5/30/2019 0:00	0.834390	0	2	0.000000	Inland Empire	Video	Retargeting	0	1	0	0	0.
3	5/30/2019 0:00	0.038948	0	1	0.000000	San Diego	Video	Retargeting	0	4	0	0	0.
4	5/29/2019 0:00	89.279997	34	1918	0.017727	San Diego	Static Image	Prospecting	2	5	0	0	0.
5	5/29/2019 0:00	28.819167	5	572	0.008741	Inland Empire	Video	Prospecting	1	6	0	0	0.
6	5/29/2019 0:00	8.775672	6	96	0.062500	Inland Empire	Video	Retargeting	0	7	0	0	0.

```
# Making sure the right datatype is used for analysis purpose
predict_df['market']=predict_df['market'].astype('category').copy()
```

```
predict_df['market'] = predict_df['market'].cat.codes
```

```
predict_df.head()
```

	date	spend	clicks	impressions	ctr	market	creative_type	prospecting_or_retargeting	leads	campaign_id	appointments	contracts	revenu
0	5/30/2019 0:00	0.834390	0	2	0.000000	0	Video	Retargeting	0	1	0	0	0.
3	5/30/2019 0:00	0.038948	0	1	0.000000	1	Video	Retargeting	0	4	0	0	0.
4	5/29/2019 0:00	89.279997	34	1918	0.017727	1	Static Image	Prospecting	2	5	0	0	0.
5	5/29/2019 0:00	28.819167	5	572	0.008741	0	Video	Prospecting	1	6	0	0	0.
6	5/29/2019 0:00	8.775672	6	96	0.062500	0	Video	Retargeting	0	7	0	0	0.

```
# Making sure the right datatype is used for analysis purpose
predict_df['creative_type']=predict_df['creative_type'].astype('category').copy()
```

```
predict_df['creative_type'] = predict_df['creative_type'].cat.codes
```

```
predict_df.head()
```

	date	spend	clicks	impressions	ctr	market	creative_type	prospecting_or_retargeting	leads	campaign_id	appointments	contracts	revenu
0	5/30/2019 0:00	0.834390	0	2	0.000000	0	1	Retargeting	0	1	0	0	0.
3	5/30/2019 0:00	0.038948	0	1	0.000000	1	1	Retargeting	0	4	0	0	0.
4	5/29/2019 0:00	89.279997	34	1918	0.017727	1	0	Prospecting	2	5	0	0	0.
5	5/29/2019 0:00	28.819167	5	572	0.008741	0	1	Prospecting	1	6	0	0	0.
6	5/29/2019 0:00	8.775672	6	96	0.062500	0	1	Retargeting	0	7	0	0	0.

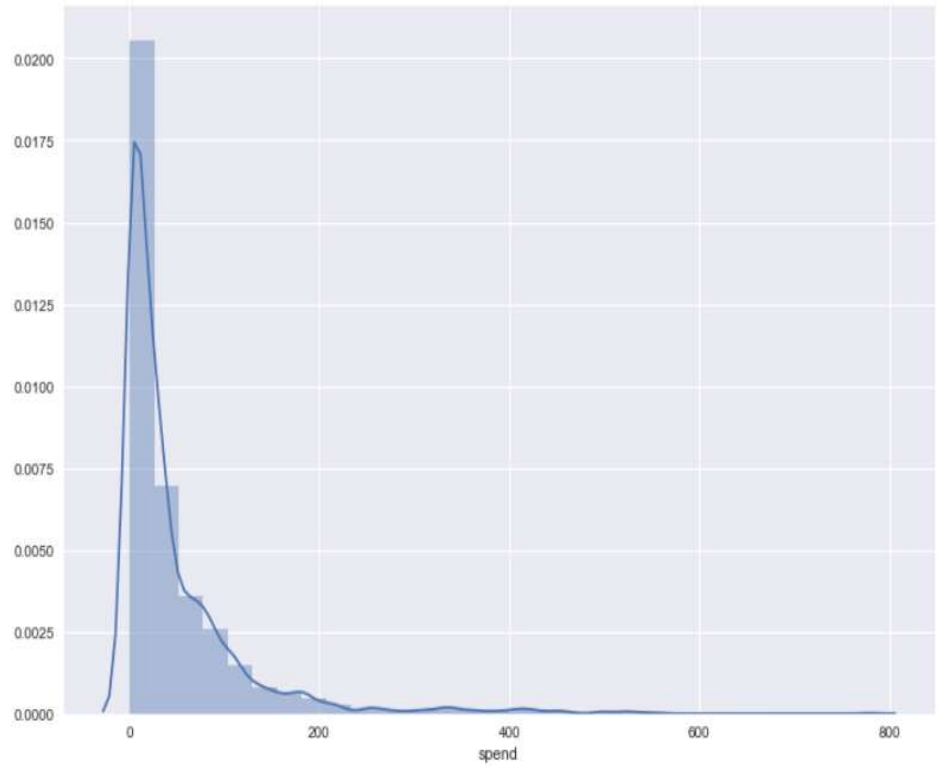
```
# for analysis purpose transforming the categorical values to numeric one.
predict_df['prospecting_or_retargeting']=predict_df['prospecting_or_retargeting'].astype('category').copy()
```

```
predict_df['prospecting_or_retargeting'] = predict_df['prospecting_or_retargeting'].cat.codes
```

```
predict_df.head()
```

	date	spend	clicks	impressions	ctr	market	creative_type	prospecting_or_retargeting	leads	campaign_id	appointments	contracts	revenu
0	5/30/2019 0:00	0.834390	0	2	0.000000	0	1	1	0	1	0	0	0.
3	5/30/2019 0:00	0.038948	0	1	0.000000	1	1	1	0	4	0	0	0.
4	5/29/2019 0:00	89.279997	34	1918	0.017727	1	0	0	2	5	0	0	0.
5	5/29/2019 0:00	28.819167	5	572	0.008741	0	1	0	1	6	0	0	0.
6	5/29/2019 0:00	8.775672	6	96	0.062500	0	1	1	0	7	0	0	0.

```
#first plot the distribution of the target variable spend. We will use the distplot function from the seaborn library.
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.distplot(predict_df['spend'], bins=30)
plt.show()
```



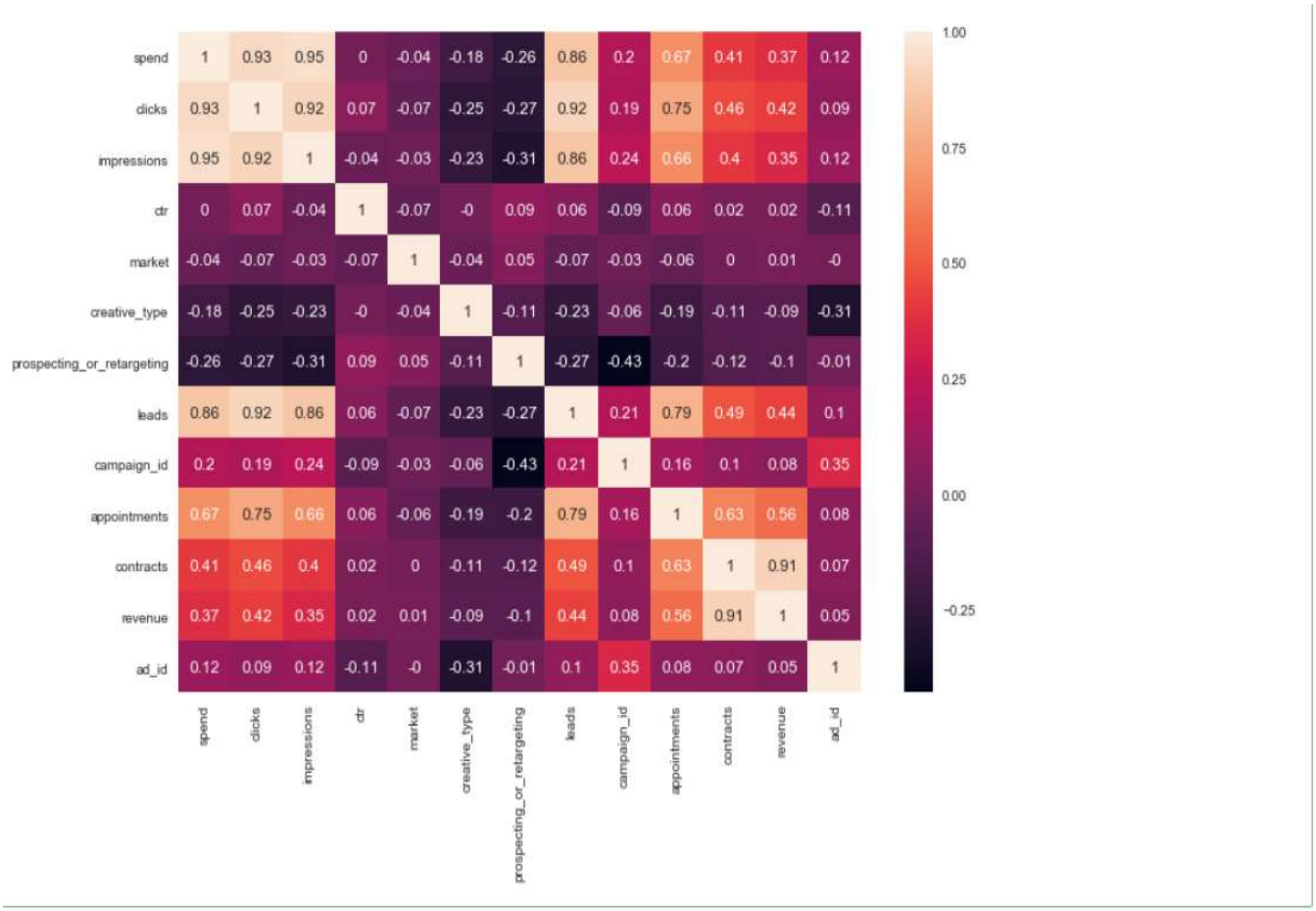

```
# Checking the correlation value between spend and prospecting_or_retargeting
predict_df['spend'].corr(predict_df['prospecting_or_retargeting'])
```

```
-0.2564725893891854
```

```
# THe values of Spend is normally distrubuted
```

```
#we will create correlation matrix by using corr funciton from pandas library
# heatmap function from seaborn library to plot correlation matrix.
correlation_matrix = predict_df.corr().round(2)
# annot = True to print the values inside the square
sns.heatmap(data=correlation_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x22bd3d259e8>
```



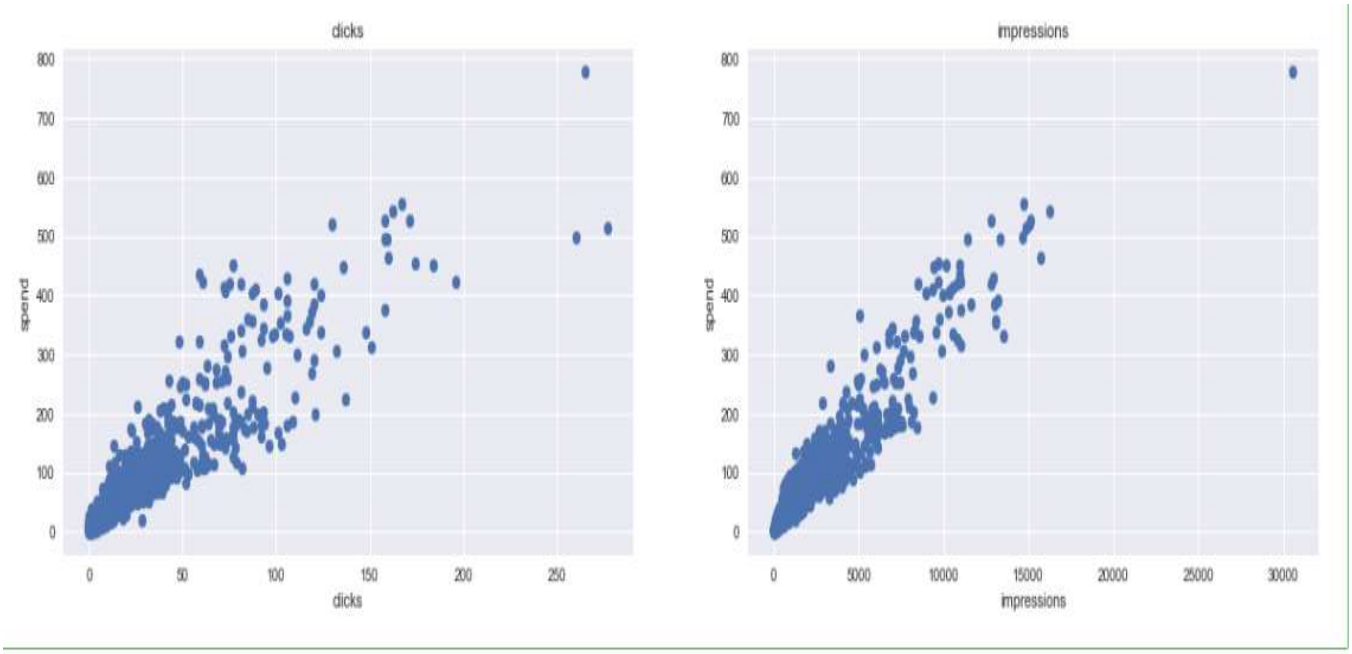

```
# hence we found that impressions and clicks are having highest correleation value close to 1 i.e. .95 and .93
# using scatter plot lets imagine how the feature vary with spend.
```

```
plt.figure(figsize=(20, 5))

features = ['clicks', 'impressions']
target = predict_df['spend']

for i, col in enumerate(features):
    plt.subplot(1, len(features) , i+1)
    x = predict_df[col]
    y = target
    plt.scatter(x, y, marker='o')
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('spend')
```

Above Correlation matrix shows us that highest correlation value for Spend is with clicks and impressions having .95 and .93 values respectively.



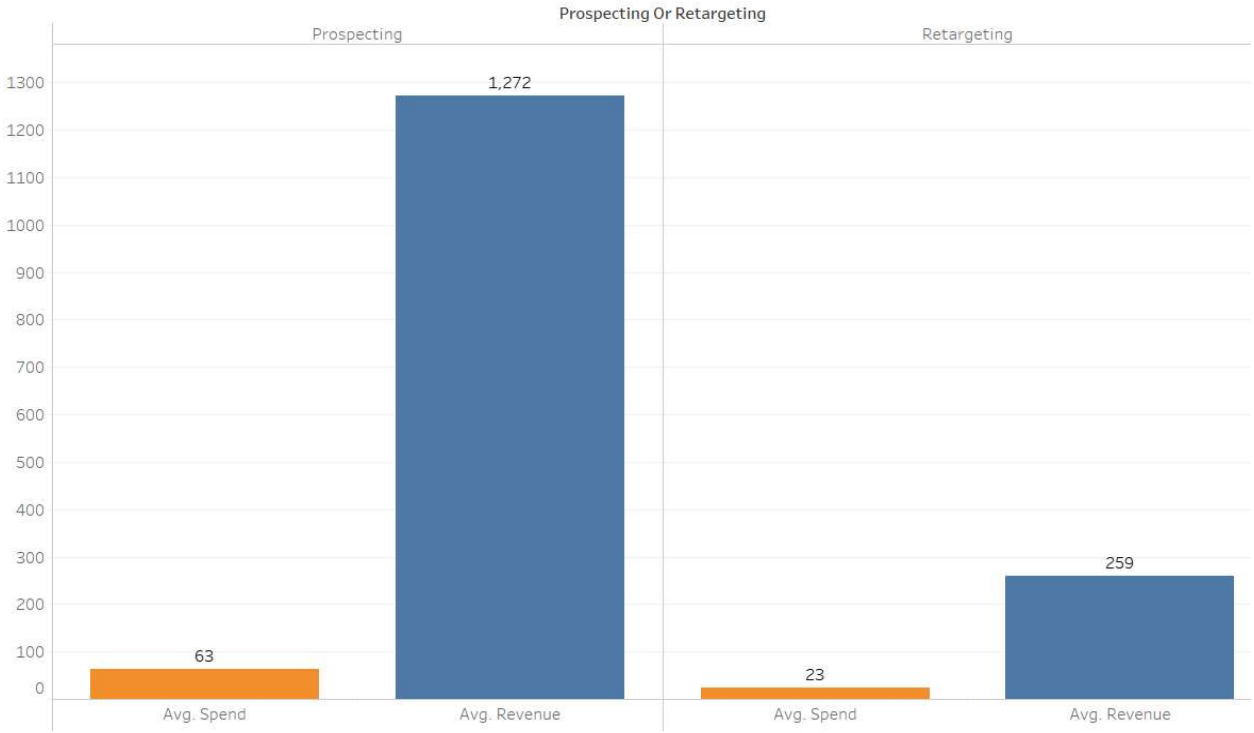

```
# By graph we found that Spend depends on two factors i.e. clicks and impressions. hence it is derived from our analysis that
# to optmize the spend we need to spend less money in advertasing.
```

Above solution tells us that Spend is directly proportional to Clicks and Impressions.

To optimize Spend, following are the recommendations which can be provided to the marketing team are as follow:

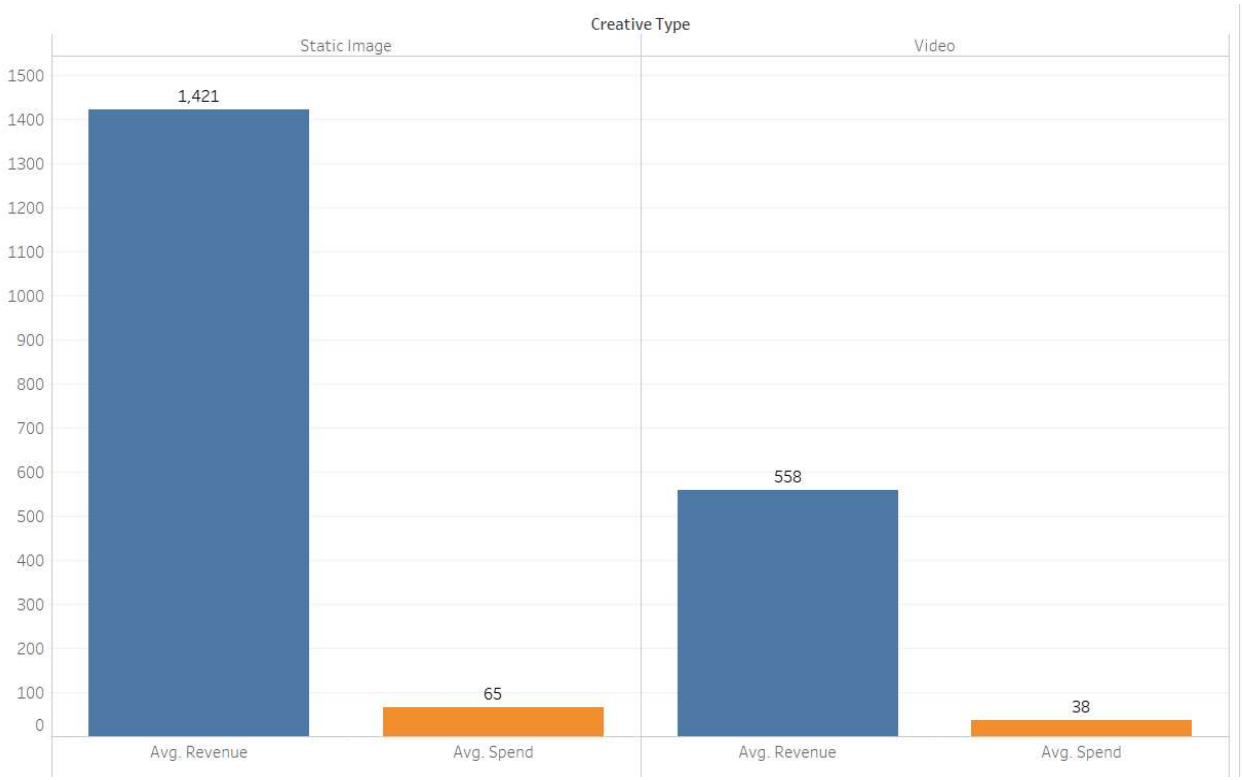
By our analysis we found that whenever Spend is less we will have more ROAS(Return on ad spend)

a) Based on the above observation, we found that for average ROAS is less for Retargeting people, which can be calculated by using below formula:
Average ROAS = Avg Revenue / Avg Spend
Below graph shows the values for Avg Spend and Avg Revenue for both Retargeting and Prospect people.



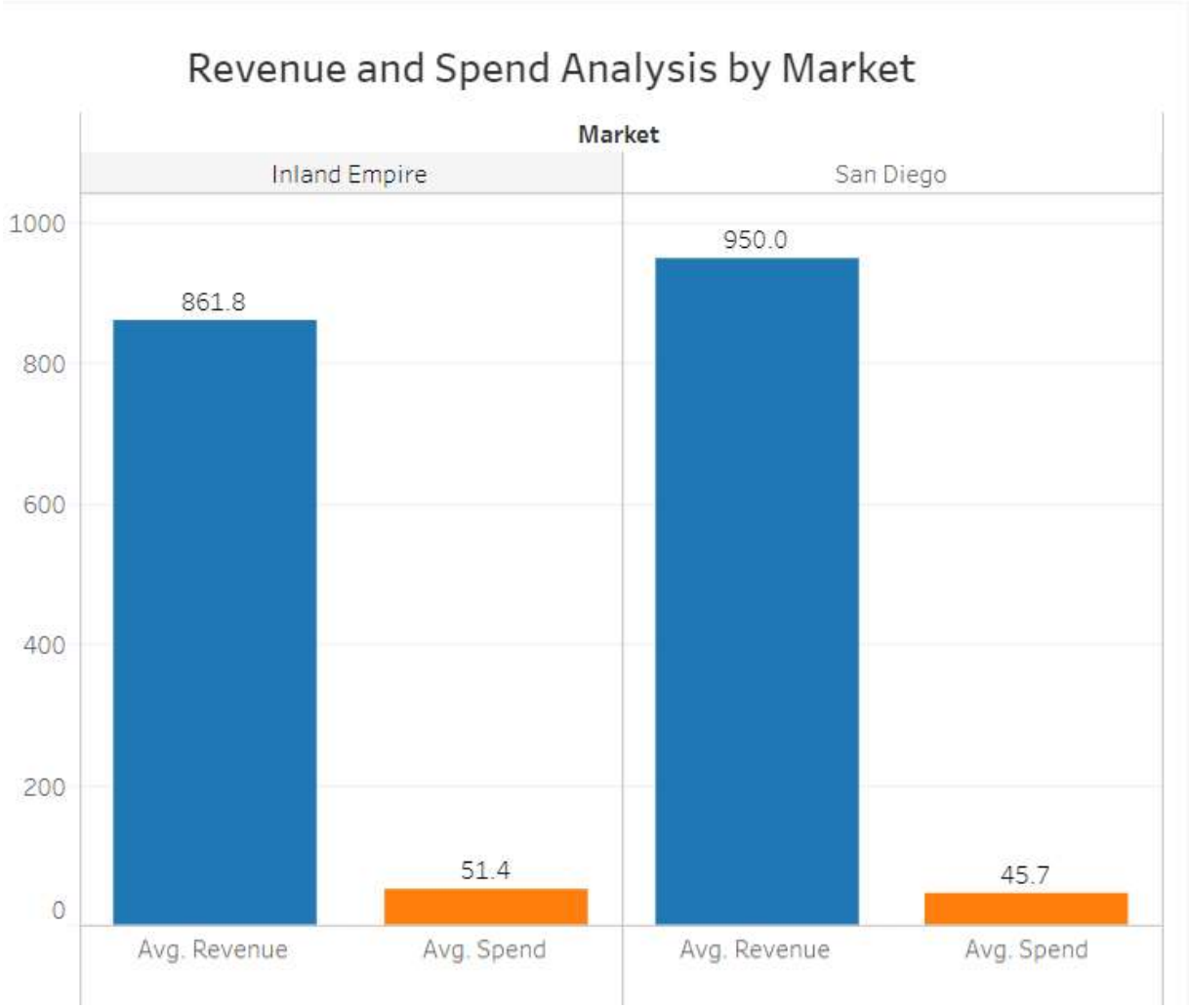
Hence for Retargeting people Avg ROAS is 11.14 and for Prospect people is 20.19, thus we can say to optimize Spend we should put less advertisement for Retargeting people, because they are already our customer, by putting less amount on Spend for retargeting people, we can also increase our Avg ROAS value. In conclusion we can conclude that we should target people who have not been on our website before.

b) In Our second we compare the values of Spend and Revenue for both Video and Static image advertisement. We found that Avg ROAS for Static Image advertisement is less than Avg ROAS for Video advertisement.



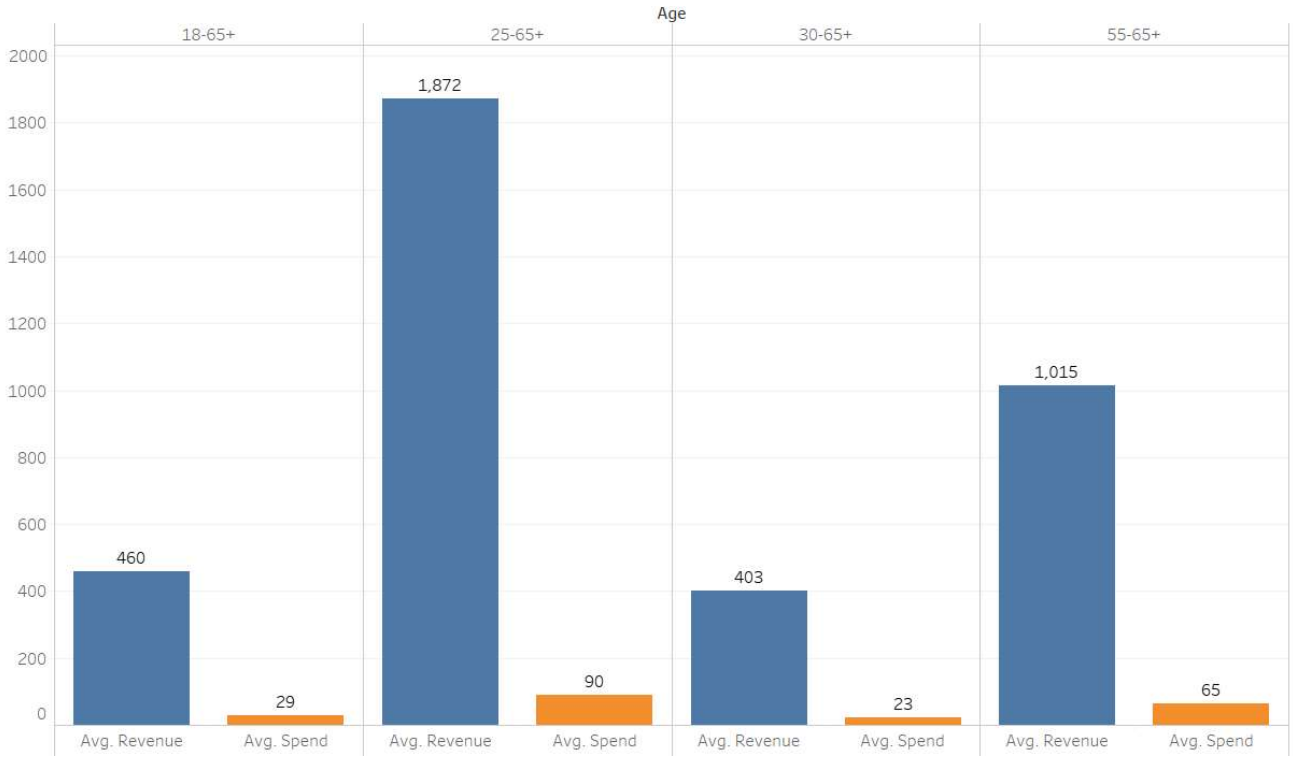
We found that Avg ROAS for Static Image is 14.68 compare to Avg ROAS for Video which is 21.86, so with this observation we can say that if we decrease our spend for Static image, we can increase our ROAS for Static Image. Hence, we could suggest our Marketing team to spend less on Static Image Advertisement as we are getting less ROAS for this. People are more attracted towards advertisement which in form of videos.

c) Our third observation is with respect to both the market San Diego and Inland Empire. We compare ROAS values for both the market.



We found that ROAS value for Inland Empire which is 16.7 less compared to the ROAS value of San Diego which is 20.21, hence with this observation we can say that we optimize the Spend for InLand Empire we can increase our Return on Ad Value. In conclusion we can say that San Doego Market is good our marketing compared to Inland Empire which is having high ROAS value.

d) Our fourth observation is with respect to both the Age groups provided in the data set. We compare ROAS values for every age group.

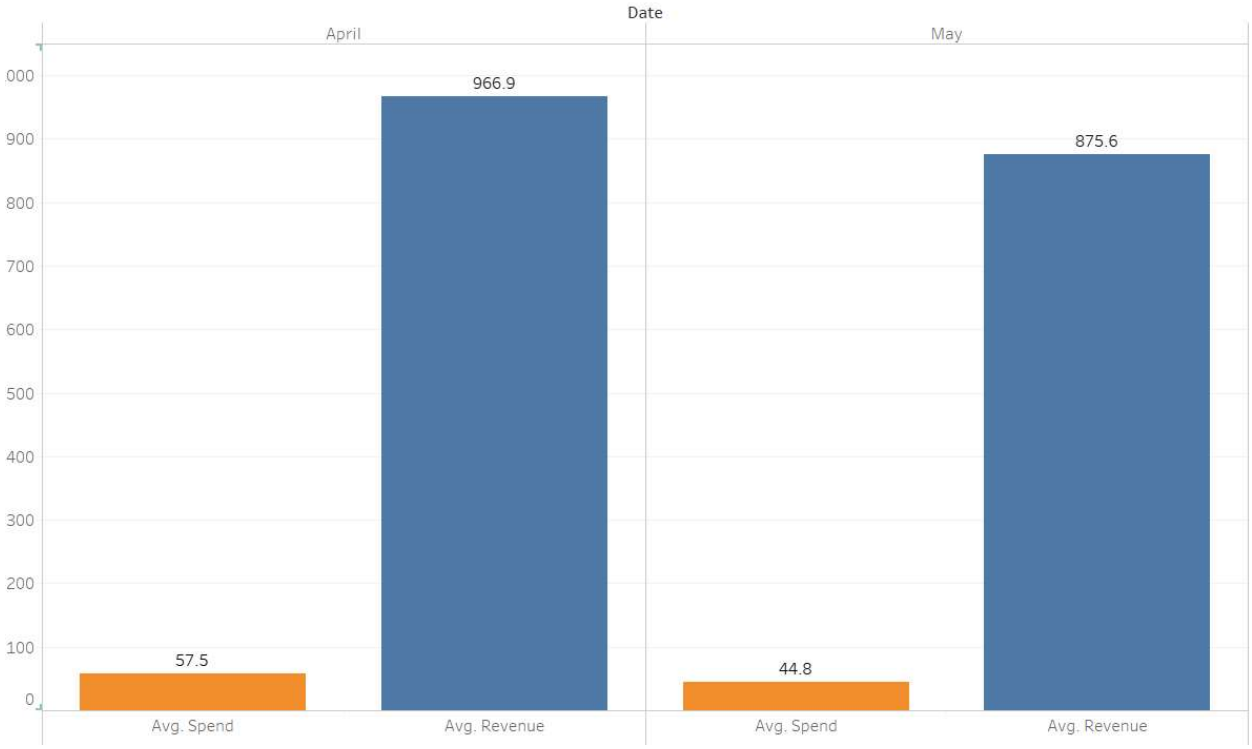


We found that ROAS value for Age group 55-65+ which is 15.61 least among all the age groups. hence with this observation we can say that we optimize the Spend for Old people age group, as our Return on Ad Value is lowest for this age group. In conclusion we can say Advertisement should be made to target young generation with innovative ideas.

Q4. Are there any particularly interesting insights you found in this data?

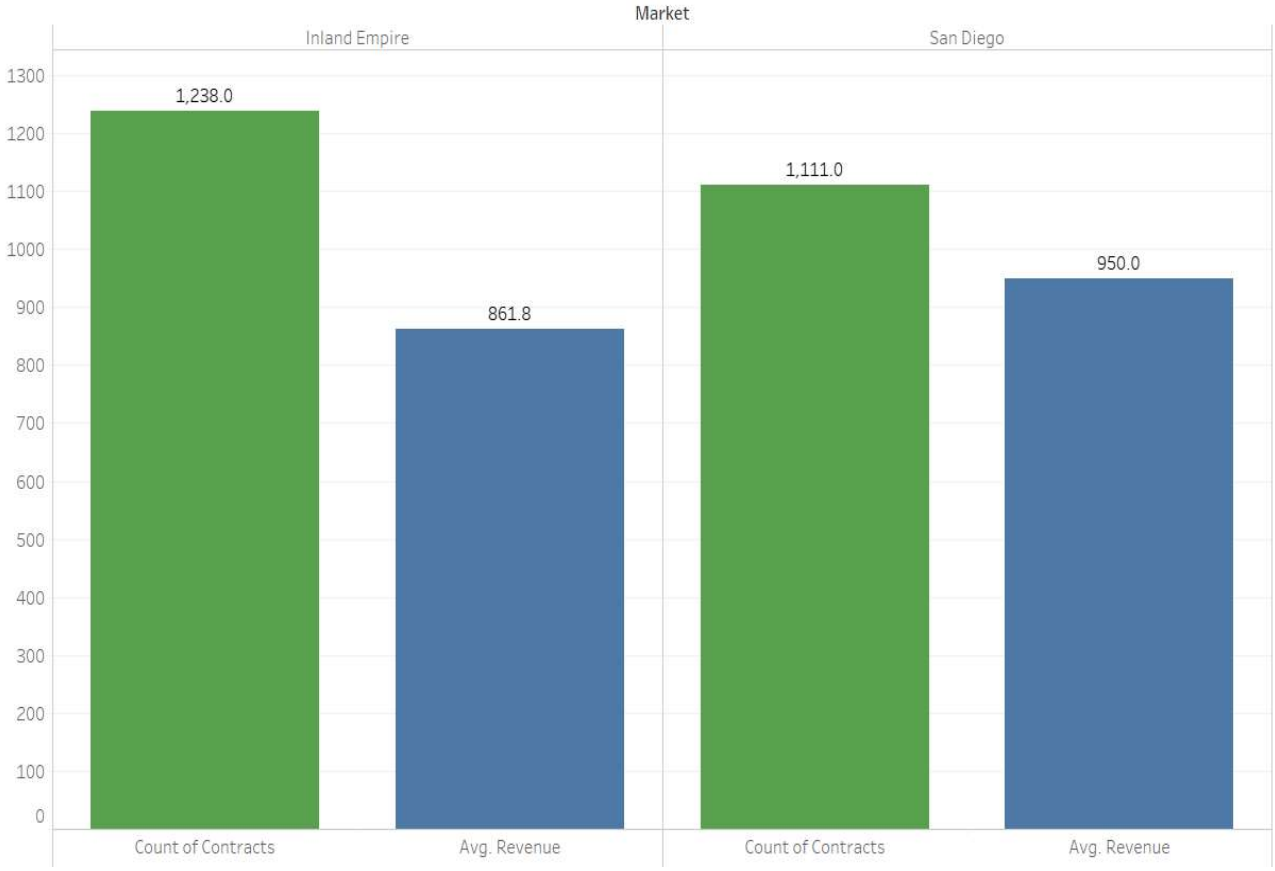
I found following interesting insights in this data. They are as follows:

a) Based on the analysis work, we found that ROAS value is getting better month by month. We found that for May month we found higher ROAS value that is 19.5 compared to April month ROAS value which is 16.81



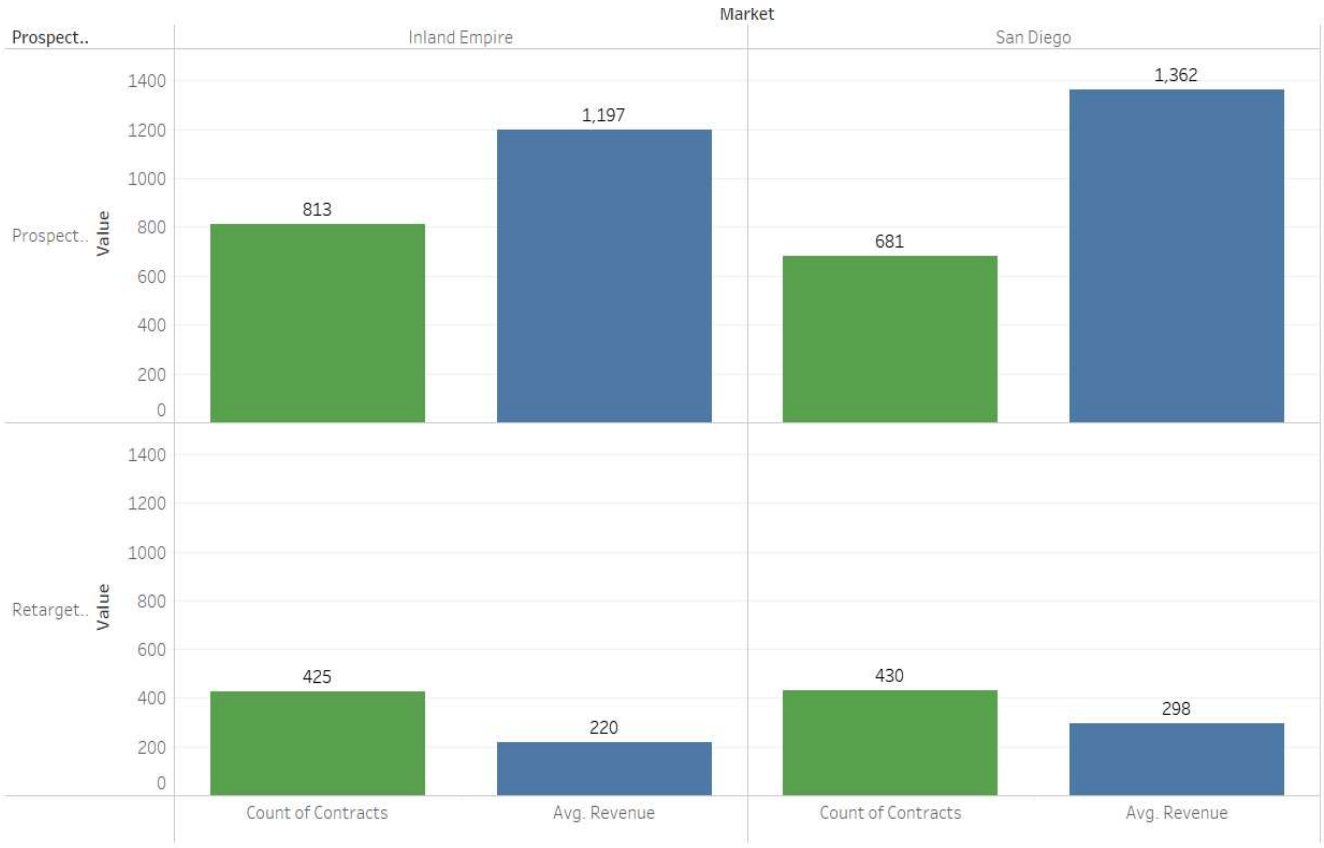
With this observation we can say that the ROAS value is increasing as the month increases but there is signification decrease in the total average Revenue from April to May month. Marketing team should be concerned of this situation and should plan their campaign accordingly.

b) My second observation with the data is comparison between different Market type and different type of people (Retargeting and Prospect)



With this observation we can say that the Number of Contracts for Inland Empire is more compared to the number of Contracts in San Diego region. However, Average revenue for San Diego market is higher 950 compared to average revenue for Inland Empire having Avg Revenue as 861.8

Also, we add different types of people in this graph we can find some interesting observation with that as well:

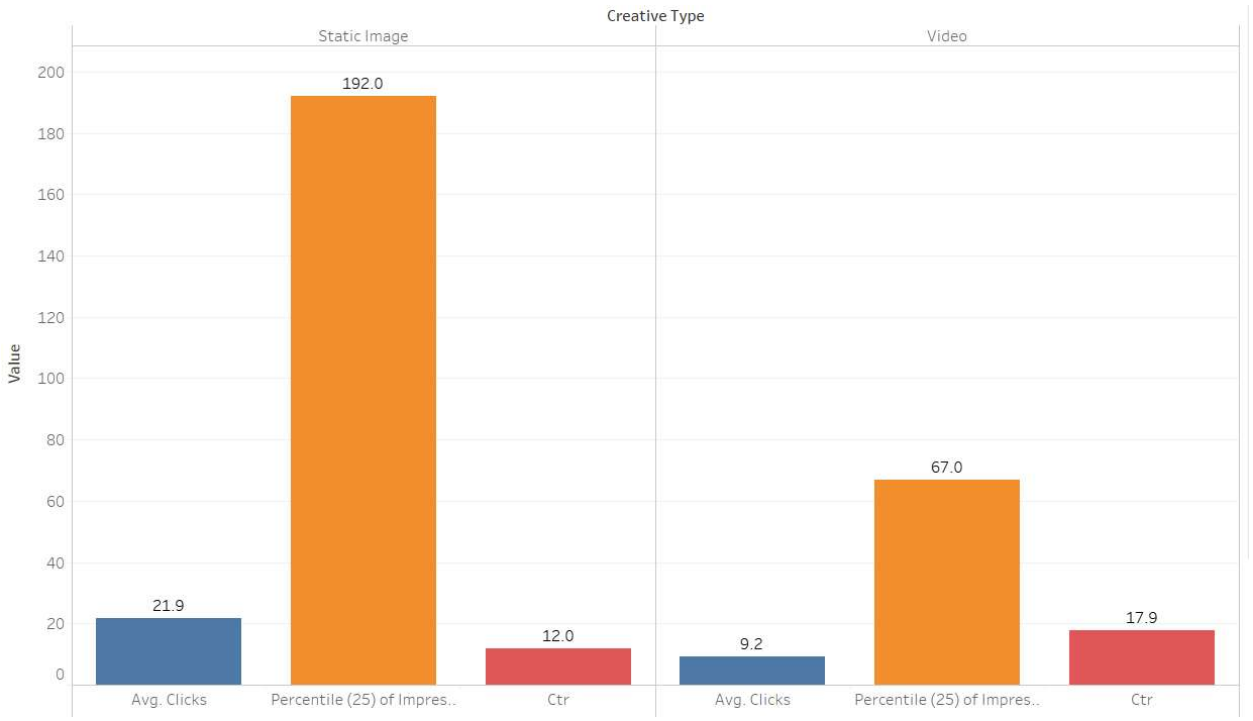


For Prospecting people, the number of contracts is higher for Inland market compared to San Diego market but if we compared the Avg Revenue it is higher in San Diego market compared to Inland Empire.

For Retargeting people, the number of contracts almost same for both the markets Inland market and San Diego. But if we compared the Avg Revenue it is higher in San Diego market compared to Inland Empire.

Marketing team should more focus on Prospecting people who have not been on the website before.

c) My third observation with the data is comparison between CTR, Clicks and Impression for both creative type of ad (Static Image and Video)



With this observation we can say that CTR is better for Video advertisement compared to Static Image. By this we can conclude that market team should focus on Video Advertisement for their campaign which helps in attracting more people to see the ads.

d) Interactive Dashboard made to get a proper insight knowledge of the data:

Link:
https://public.tableau.com/profile/arpit.khandekar#!/vizhome/Market_Cmpaign_Analysis/Market_Campaign_Analysis