Importing the necessary libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading the Dataset

```
df = pd.read_csv('D:\\Instagram data.csv' , encoding='latin1')

df.head()
```

```
   Impressions  From Home  From Hashtags  From Explore  From Other
Saves  \
0         3920       2586           1028           619          56
98
1         5394       2727           1838          1174          78
194
2         4021       2085           1188             0         533
41
3         4528       2700            621           932          73
172
4         2518       1704            255           279          37
96

   Comments  Shares  Likes  Profile Visits  Follows  \
0         9       5    162              35        2
1         7      14    224              48       10
2        11       1    131              62       12
3        10       7    213              23        8
4         5       4    123               8        0

                                             Caption  \
0  Here are some of the most important data visua...
1  Here are some of the best data science project...
2  Learn how to train a machine learning model an...
3  Here□s how you can write a Python program to d...
4  Plotting annotations while visualizing your da...

                                            Hashtags
0  #finance #money #business #investing #investme...
1  #healthcare #health #covid #data #datascience ...
2  #data #datascience #dataanalysis #dataanalytic...
3  #python #pythonprogramming #pythonprojects #py...
4  #datavisualization #datascience #data #dataana...
```

```
df.shape
```

```
(119, 13)
```

```
df.columns
```

```
Index(['Impressions', 'From Home', 'From Hashtags', 'From Explore',
       'From Other', 'Saves', 'Comments', 'Shares', 'Likes', 'Profile
Visits',
       'Follows', 'Caption', 'Hashtags'],
      dtype='object')
```

Checking Null Values

```
df.isnull().sum()
```

```
Impressions       0
From Home         0
From Hashtags     0
From Explore      0
From Other        0
Saves             0
Comments          0
Shares            0
Likes             0
Profile Visits    0
Follows           0
Caption           0
Hashtags          0
dtype: int64
```

Descriptive statistics

```
df.describe()
```

```
        Impressions     From Home  From Hashtags  From Explore   From
Other  \
count    119.000000    119.000000     119.000000    119.000000
119.000000
mean    5703.991597   2475.789916    1887.512605   1078.100840
171.092437
std     4843.780105   1489.386348    1884.361443   2613.026132
289.431031
min     1941.000000   1133.000000     116.000000      0.000000
9.000000
25%     3467.000000   1945.000000     726.000000    157.500000
38.000000
50%     4289.000000   2207.000000    1278.000000    326.000000
74.000000
75%     6138.000000   2602.500000    2363.500000    689.500000
196.000000
max    36919.000000  13473.000000   11817.000000  17414.000000
2547.000000
```

```
              Saves    Comments      Shares       Likes  Profile Visits
\
count   119.000000  119.000000  119.000000  119.000000      119.000000

mean    153.310924    6.663866    9.361345  173.781513       50.621849

std     156.317731    3.544576   10.089205   82.378947       87.088402

min      22.000000    0.000000    0.000000   72.000000        4.000000

25%      65.000000    4.000000    3.000000  121.500000       15.000000

50%     109.000000    6.000000    6.000000  151.000000       23.000000

75%     169.000000    8.000000   13.500000  204.000000       42.000000

max    1095.000000   19.000000   75.000000  549.000000      611.000000


           Follows
count  119.000000
mean    20.756303
std     40.921580
min      0.000000
25%      4.000000
50%      8.000000
75%     18.000000
max    260.000000

df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119 entries, 0 to 118
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Impressions     119 non-null    int64
 1   From Home       119 non-null    int64
 2   From Hashtags   119 non-null    int64
 3   From Explore    119 non-null    int64
 4   From Other      119 non-null    int64
 5   Saves           119 non-null    int64
 6   Comments        119 non-null    int64
 7   Shares          119 non-null    int64
 8   Likes           119 non-null    int64
 9   Profile Visits  119 non-null    int64
 10  Follows         119 non-null    int64
 11  Caption         119 non-null    object
 12  Hashtags        119 non-null    object
```

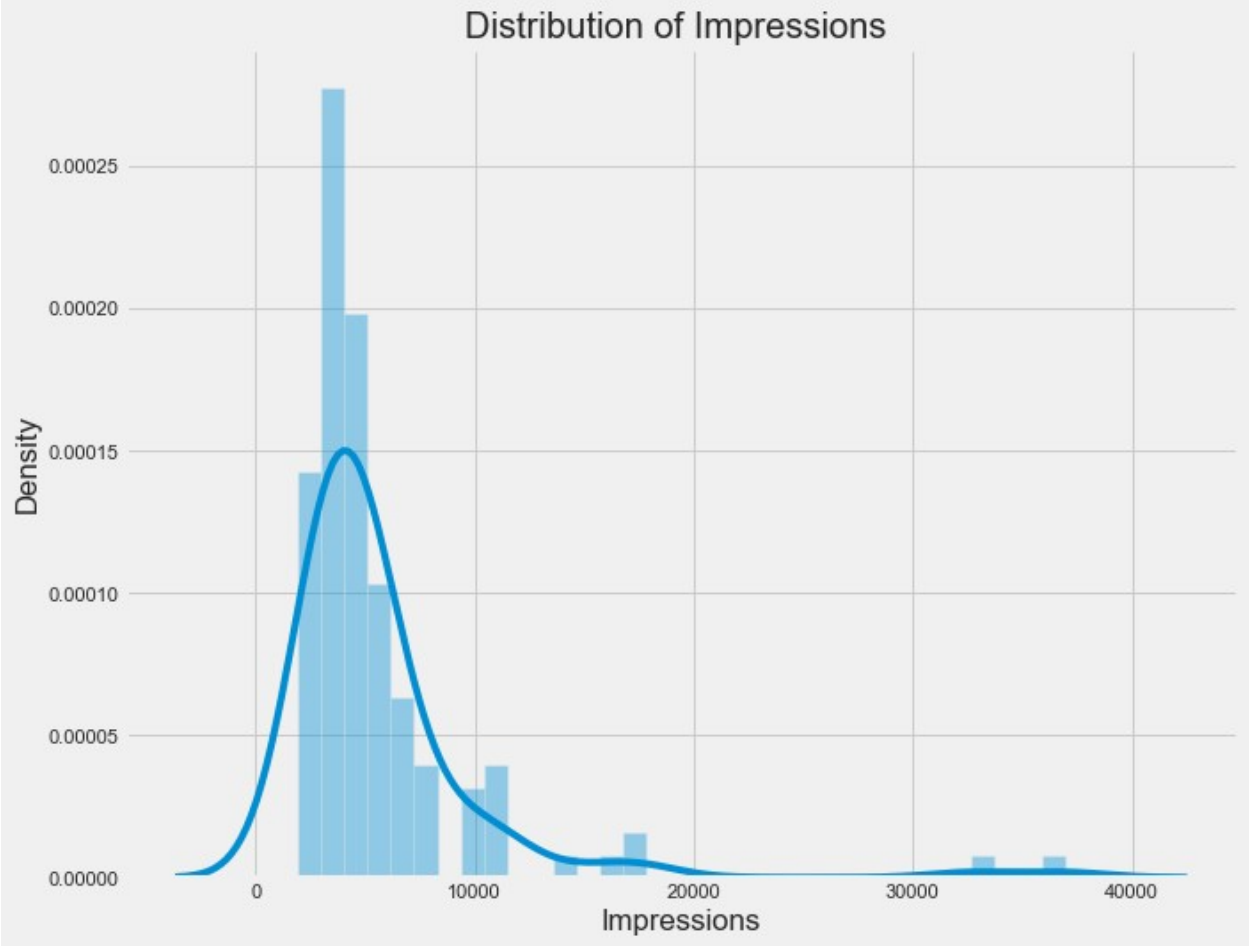
```
dtypes: int64(11), object(2)
memory usage: 12.2+ KB
```

Data Visualization & Explanatory Data Analysis

```
plt.figure(figsize=(10,8))
plt.style.use("fivethirtyeight")
plt.title("Distribution of Impressions ")
sns.distplot(df['Impressions'], kde= True)
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_7276\2045788956.py:4:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Impressions'], kde= True)
```

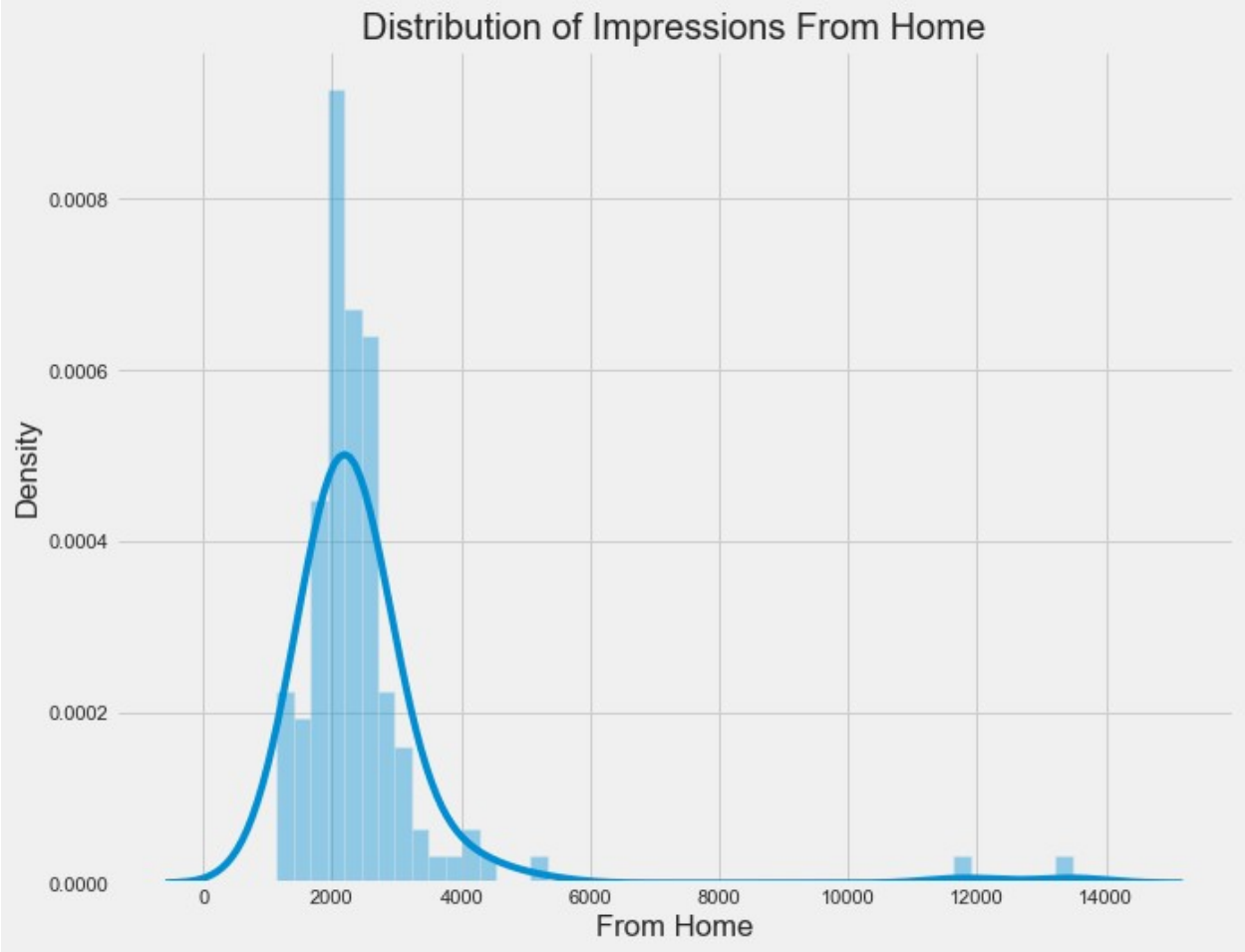
```
<Axes: title={'center': 'Distribution of Impressions '},
xlabel='Impressions', ylabel='Density'>
```




```
plt.figure(figsize=(10,8))
plt.style.use("fivethirtyeight")
plt.title("Distribution of Impressions From Home")
sns.distplot(df['From Home'], kde= True)
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_7276\513589272.py:4:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['From Home'], kde= True)
```

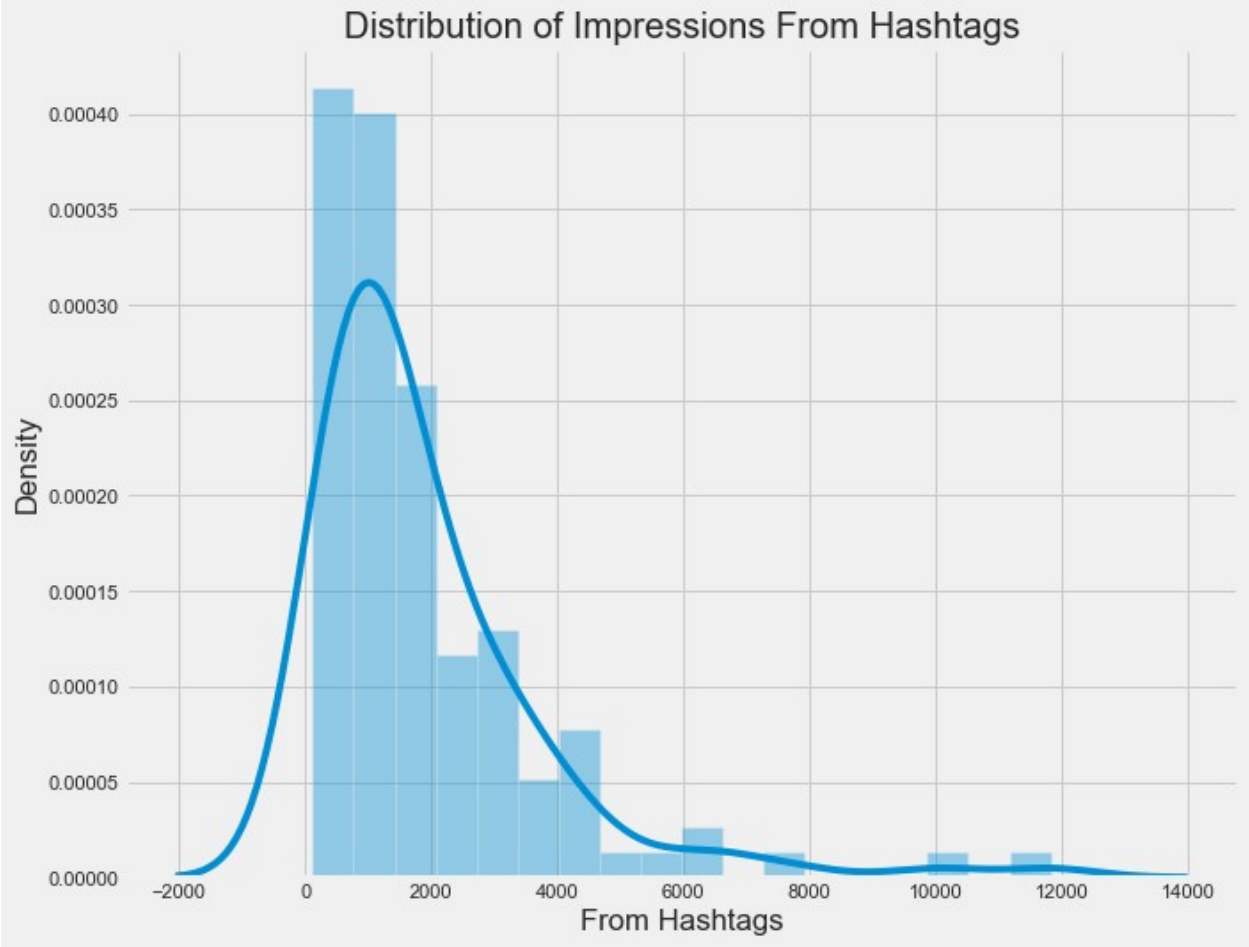
```
<Axes: title={'center': 'Distribution of Impressions From Home'},
xlabel='From Home', ylabel='Density'>
```




```
plt.figure(figsize=(10,8))
plt.style.use("fivethirtyeight")
plt.title("Distribution of Impressions From Hashtags")
sns.distplot(df['From Hashtags'], kde= True)
```

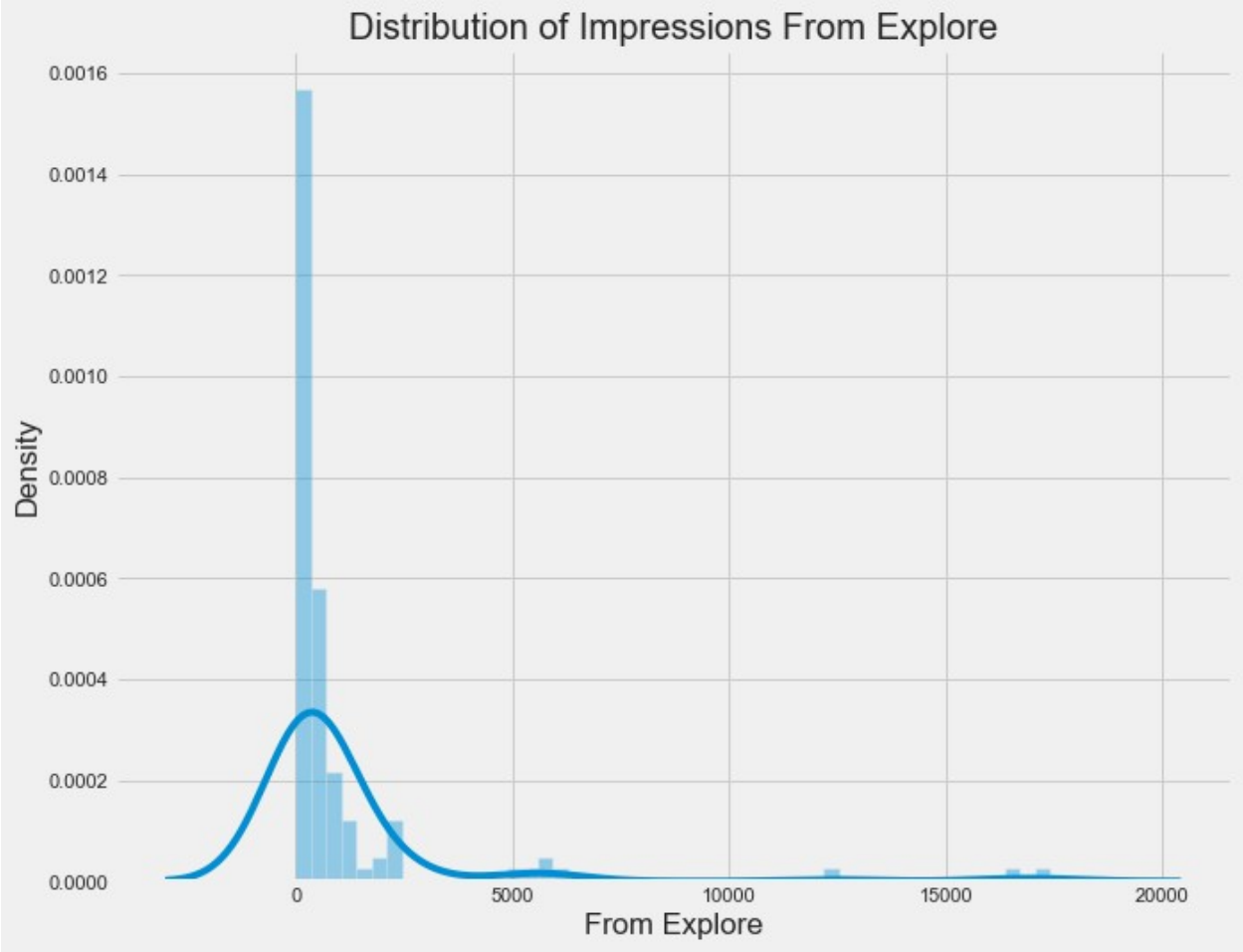
```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_7276\634880641.py:4:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
```

```
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['From Hashtags'], kde= True)

<Axes: title={'center': 'Distribution of Impressions From Hashtags'},
xlabel='From Hashtags', ylabel='Density'>
```



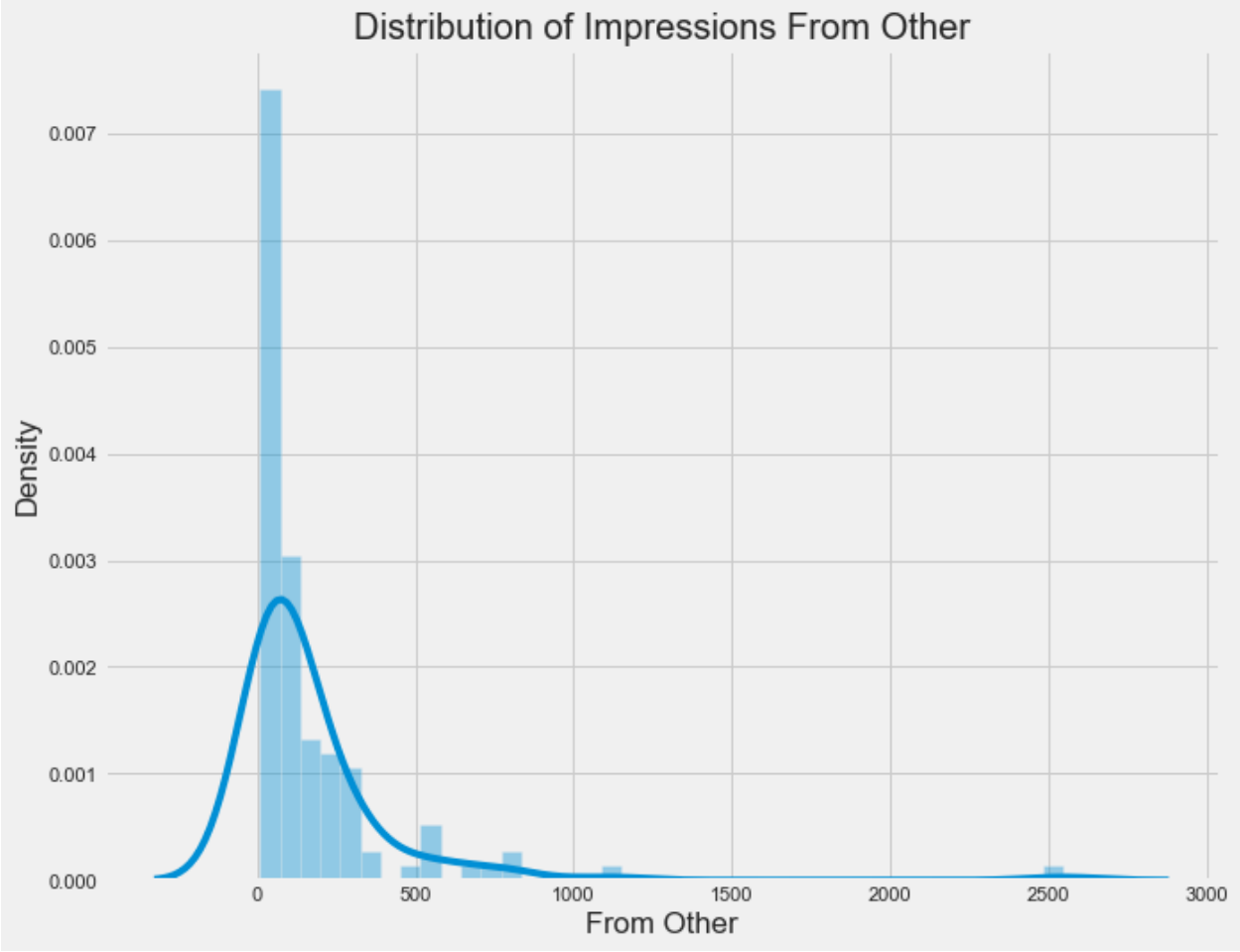

```
plt.figure(figsize=(10,8))
plt.style.use("fivethirtyeight")
plt.title("Distribution of Impressions From Explore")
sns.distplot(df['From Explore'], kde= True)

C:\Users\LENOVO\AppData\Local\Temp\ipykernel_7276\1504332497.py:4:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
```

```
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['From Explore'], kde= True)

<Axes: title={'center': 'Distribution of Impressions From Explore'},
xlabel='From Explore', ylabel='Density'>
```



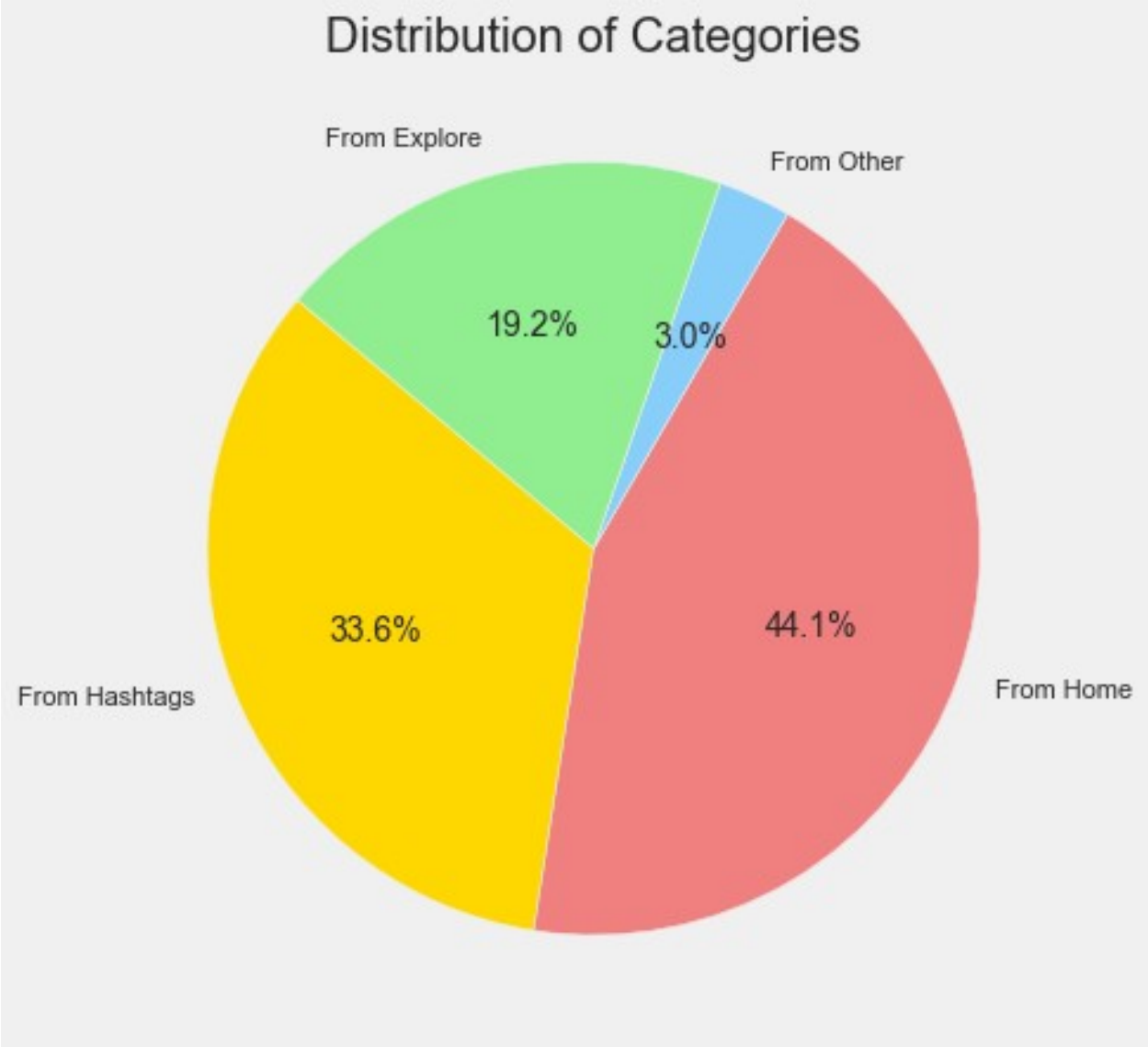

```
plt.figure(figsize=(10,8))
plt.style.use("fivethirtyeight")
plt.title("Distribution of Impressions From Other")
sns.distplot(df['From Other'], kde= True)

C:\Users\LENOVO\AppData\Local\Temp\ipykernel_7276\873347331.py:4:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.
```

```
Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['From Other'], kde= True)

<Axes: title={'center': 'Distribution of Impressions From Other'},
xlabel='From Other', ylabel='Density'>
```

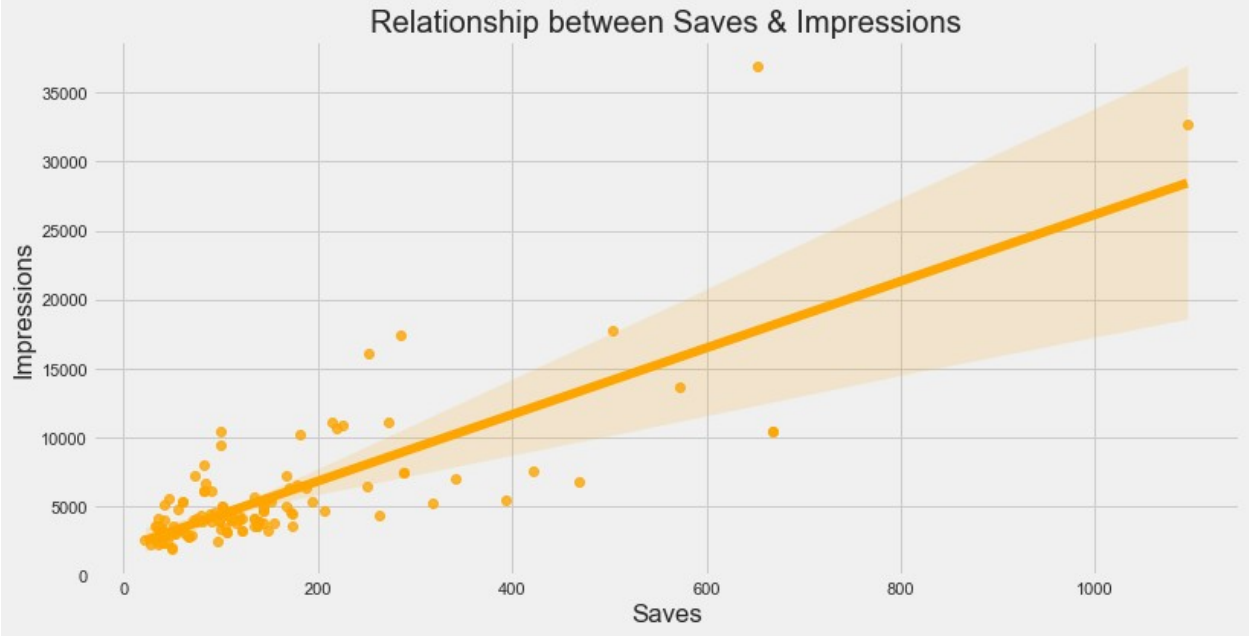



```
plt.figure(figsize=(7, 7))

# Calculate the means
sum = [df["From Hashtags"].sum(), df["From Home"].sum(), df["From
Other"].sum(), df["From Explore"].sum()]

# Define colors for each category
colors = ['gold', 'lightcoral', 'lightskyblue', 'lightgreen']
```

```
# Plot the pie chart with specified colors
plt.pie(sum, labels=['From Hashtags', 'From Home', 'From Other', 'From
Explore'], colors=colors, autopct='%1.1f%%', startangle=140)

plt.title('Distribution of Categories')
plt.show()
```

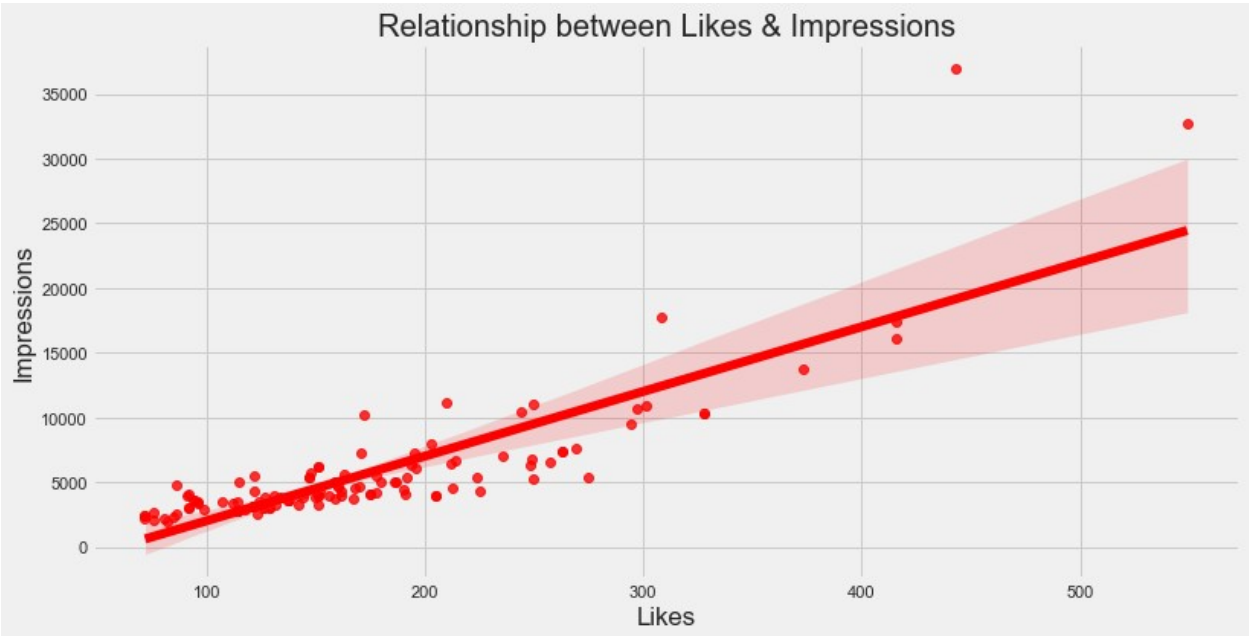



```
plt.figure(figsize=(12, 6))
plt.title("Relationship between Saves & Impressions")
sns.regplot( data=df , x=df['Saves'], y=df['Impressions'] , color =
"orange")

<Axes: title={'center': 'Relationship between Saves & Impressions'},
xlabel='Saves', ylabel='Impressions'>
```




```
plt.figure(figsize=(12, 6))
plt.title("Relationship between Likes & Impressions")
sns.regplot( data=df , x=df['Likes'], y=df['Impressions'] , color =
"red")
```

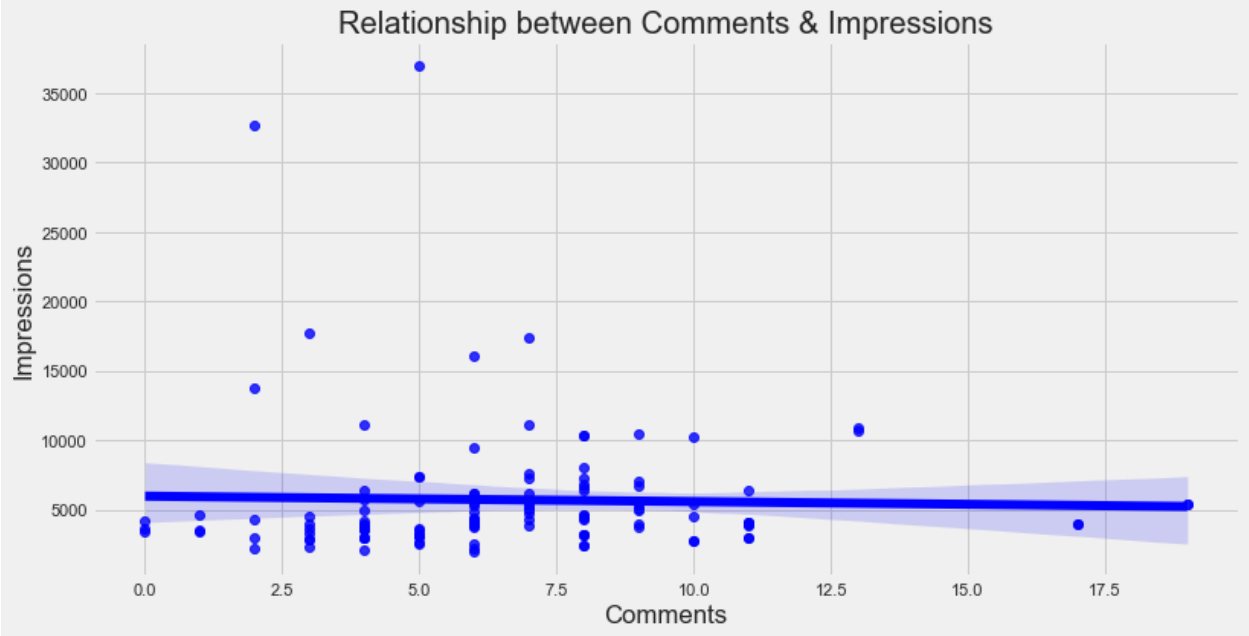
```
<Axes: title={'center': 'Relationship between Likes & Impressions'},
xlabel='Likes', ylabel='Impressions'>
```



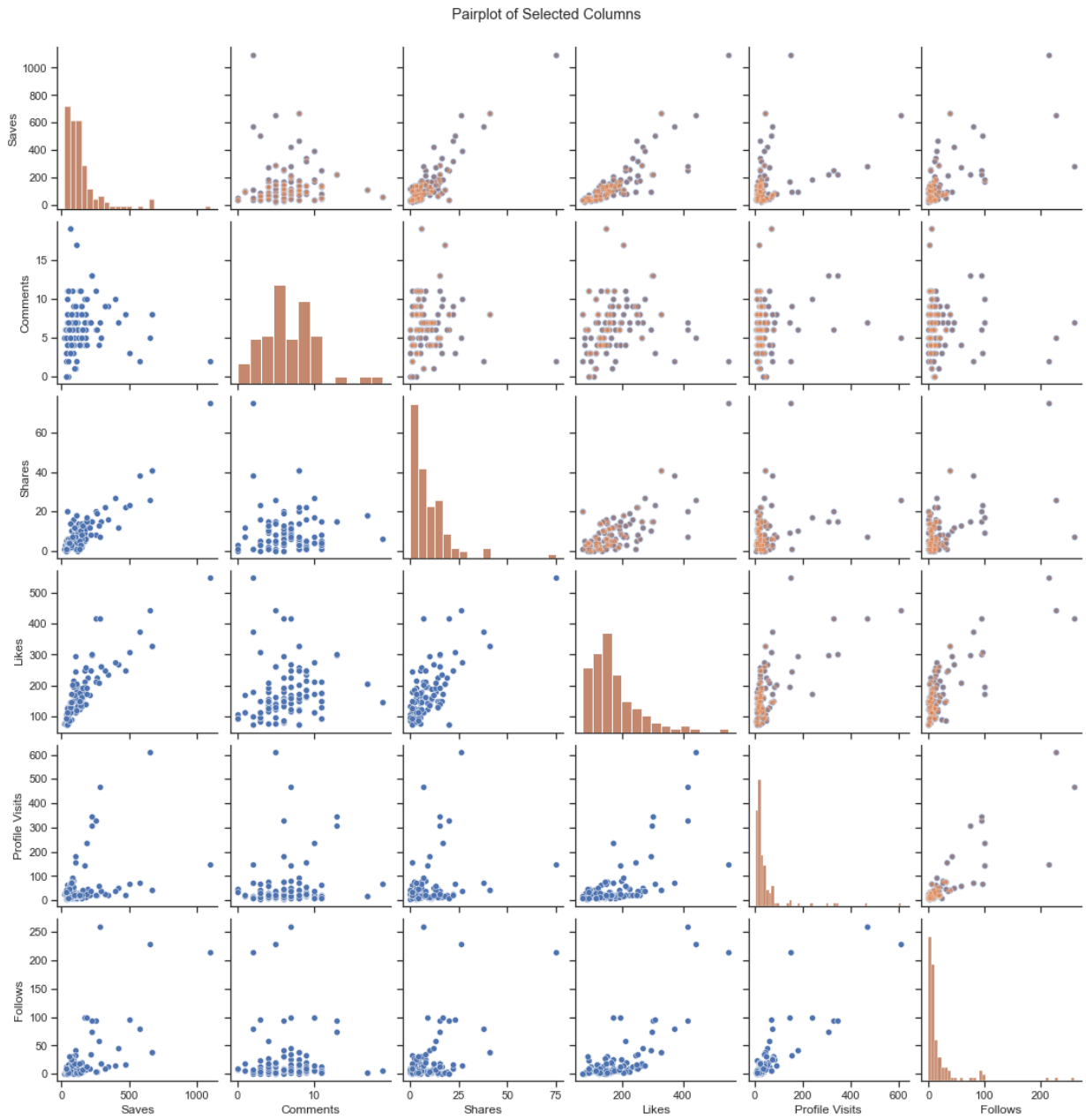

```
plt.figure(figsize=(12, 6))
plt.title("Relationship between Comments & Impressions")
```

```
sns.regplot( data=df , x=df['Comments'], y=df['Impressions'] , color = "blue")
```

```
<Axes: title={'center': 'Relationship between Comments & Impressions'}, xlabel='Comments', ylabel='Impressions'>
```




```
selected_columns = ['Saves', 'Comments', 'Shares', 'Likes', 'Profile Visits', 'Follows']


subset_df = df[selected_columns]

sns.set(style="ticks")
pair_plot = sns.pairplot(subset_df, markers="o")

pair_plot.fig.suptitle("Pairplot of Selected Columns", y=1.02)
pair_plot.map_upper(sns.scatterplot, s=20, alpha=0.5)
pair_plot.map_diag(sns.histplot, kde_kws={'color': 'k'})


pair_plot.add_legend()

plt.show()
```



Correlation

```
df1 = df.select_dtypes(include=['number'])
df1.corr()
```

	Impressions	From Home	From Hashtags	From Explore	\
Impressions	1.000000	0.844698	0.560760	0.893607	
From Home	0.844698	1.000000	0.177516	0.800573	
From Hashtags	0.560760	0.177516	1.000000	0.190453	
From Explore	0.893607	0.800573	0.190453	1.000000	
From Other	0.592960	0.555666	0.229623	0.495685	
Saves	0.779231	0.768817	0.305929	0.747803	

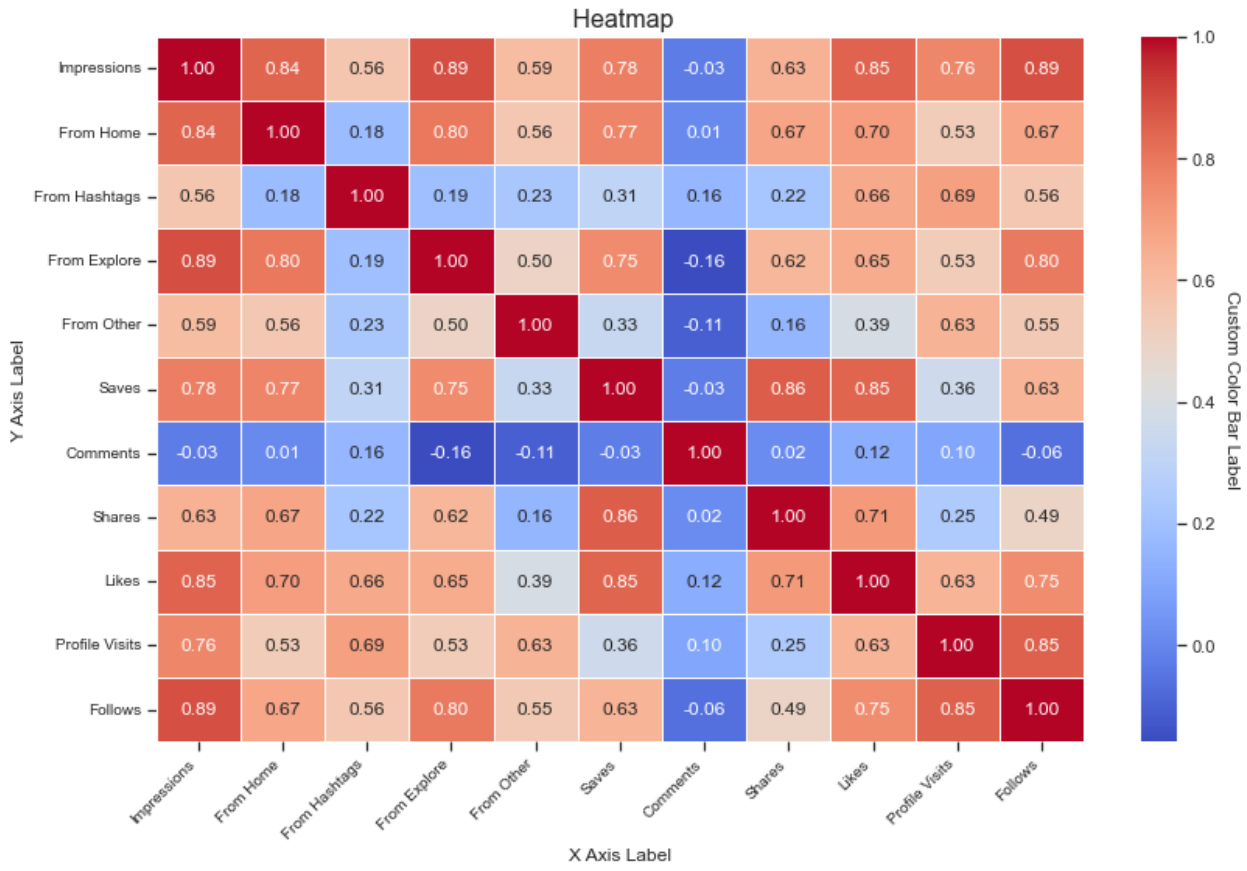
```
Comments           -0.028524   0.012716       0.161439     -0.158565
Shares              0.634675   0.674985       0.219511      0.615731
Likes               0.849835   0.698330       0.662124      0.653699
Profile Visits      0.760981   0.531076       0.691345      0.531850
Follows             0.889363   0.672675       0.555485      0.796019
```

```
                 From Other     Saves  Comments    Shares     Likes  \
Impressions        0.592960  0.779231 -0.028524  0.634675  0.849835
From Home          0.555666  0.768817  0.012716  0.674985  0.698330
From Hashtags      0.229623  0.305929  0.161439  0.219511  0.662124
From Explore       0.495685  0.747803 -0.158565  0.615731  0.653699
From Other         1.000000  0.331907 -0.108703  0.156834  0.393510
Saves              0.331907  1.000000 -0.026912  0.860324  0.845643
Comments          -0.108703 -0.026912  1.000000  0.016933  0.123586
Shares             0.156834  0.860324  0.016933  1.000000  0.707794
Likes              0.393510  0.845643  0.123586  0.707794  1.000000
Profile Visits     0.633080  0.360628  0.096714  0.245361  0.626107
Follows            0.546737  0.628461 -0.060631  0.493070  0.746333
```

```
                 Profile Visits   Follows
Impressions            0.760981  0.889363
From Home              0.531076  0.672675
From Hashtags          0.691345  0.555485
From Explore           0.531850  0.796019
From Other             0.633080  0.546737
Saves                  0.360628  0.628461
Comments               0.096714 -0.060631
Shares                 0.245361  0.493070
Likes                  0.626107  0.746333
Profile Visits         1.000000  0.853152
Follows                0.853152  1.000000
```

```
plt.figure(figsize=(12, 8))

heatmap = sns.heatmap(df1.corr(), cmap="coolwarm", annot=True,
fmt=".2f", linewidths=.5)

cbar = heatmap.collections[0].colorbar
cbar.set_label("Custom Color Bar Label", rotation=270, labelpad=15)

plt.title(" Heatmap", fontsize=16)

plt.xlabel("X Axis Label", fontsize=12)
plt.ylabel("Y Axis Label", fontsize=12)

plt.xticks(rotation=45, ha='right', fontsize=10)
plt.yticks(rotation=0, fontsize=10)

plt.grid(visible=True, linestyle="--", alpha=0.5)
```

```
plt.tight_layout()

# Save the plot if needed
plt.savefig("extraordinary_heatmap.png")

plt.show()
```



Correlation with respect to Impressions

```
df1.corrwith(df1["Impressions"])
```

```
Impressions        1.000000
From Home          0.844698
From Hashtags      0.560760
From Explore       0.893607
From Other         0.592960
Saves              0.779231
Comments          -0.028524
Shares             0.634675
Likes              0.849835
Profile Visits     0.760981
Follows            0.889363
dtype: float64
```

Correlation with respect to Follows

```
df1.corrwith(df1["Follows"])
```

```
Impressions        0.889363
From Home          0.672675
From Hashtags      0.555485
From Explore       0.796019
From Other         0.546737
Saves              0.628461
Comments          -0.060631
Shares             0.493070
Likes              0.746333
Profile Visits     0.853152
Follows            1.000000
dtype: float64
```

Correlation with respect to Likes

```
df1.corrwith(df1["Likes"])
```

```
Impressions        0.849835
From Home          0.698330
From Hashtags      0.662124
From Explore       0.653699
From Other         0.393510
Saves              0.845643
Comments           0.123586
Shares             0.707794
Likes              1.000000
Profile Visits     0.626107
Follows            0.746333
dtype: float64
```

Correlation with respect to Comments

```
df1.corrwith(df1["Comments"])
```

```
Impressions       -0.028524
From Home          0.012716
From Hashtags      0.161439
From Explore      -0.158565
From Other        -0.108703
Saves             -0.026912
Comments           1.000000
Shares             0.016933
Likes              0.123586
Profile Visits     0.096714
Follows           -0.060631
dtype: float64
```

Correlation with respect to Profile Visits

```
df1.corrwith(df1["Profile Visits"])
```

```
Impressions       0.760981
From Home         0.531076
From Hashtags     0.691345
From Explore      0.531850
From Other        0.633080
Saves             0.360628
Comments          0.096714
Shares            0.245361
Likes             0.626107
Profile Visits    1.000000
Follows           0.853152
dtype: float64
```

In the realm of machine learning

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score,
accuracy_score, confusion_matrix,classification_report
x=df1[['Likes' , 'Saves', 'Comments' , 'Shares', 'Profile Visits',
'Follows']]
y= df1['Impressions']
```

Linear Regression Model

```
x_train, x_test, y_train, y_test = train_test_split(x,y,
test_size=0.25, random_state=40)
model = LinearRegression()
model.fit(x_train,y_train)
model.score(x_test,y_test)
```

```
0.6927358126962613
```

```
pred1=model.predict(x_test)
pred1
```

```
array([ 8266.5947079 ,  8384.94422096,  2345.86872874,  7204.54676021,
       13823.61370599, 15300.44816569,  3687.42765116,  3622.17554503,
        6204.77910687,  5841.29018639, 16639.54695864, 16101.8123349 ,
        5972.53948694,  2269.3980991 ,  5326.02517556,  4538.83112201,
        2435.38276865,  2269.3980991 ,  3936.82134114,  3643.86428787,
        9915.6743771 ,  3478.94745509,  3679.30957463, 11219.9108569 ,
        3622.17554503,  3256.80368648,  4889.20409056,  2539.496056  ,
        3687.42765116,  4150.94558315])
```

DecisionTreeClassifier

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=42)
clf.fit(x_train, y_train)

# Make predictions on the testing data
y_pred = clf.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Print the results
print(f"Accuracy: {accuracy:.2f}")
print("\nConfusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(classification_rep)
```

```
Accuracy: 0.10

Confusion Matrix:
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 1]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]

Classification Report:
              precision    recall  f1-score   support

        1941       0.00      0.00      0.00         1
        2064       0.00      0.00      0.00         0
        2191       0.00      0.00      0.00         1
        2218       0.00      0.00      0.00         0
        2327       0.00      0.00      0.00         0
        2407       0.00      0.00      0.00         0
        2518       0.00      0.00      0.00         1
        2621       0.00      0.00      0.00         1
        2826       0.00      0.00      0.00         2
        3015       0.00      0.00      0.00         2
        3169       0.00      0.00      0.00         2
        3216       0.00      0.00      0.00         0
        3234       0.00      0.00      0.00         1
        3246       0.00      0.00      0.00         0
        3454       0.00      0.00      0.00         1
        3525       0.00      0.00      0.00         0
        3606       0.00      0.00      0.00         1
```

```
        3623       0.00      0.00      0.00         1
        3630       1.00      1.00      1.00         1
        3920       0.00      0.00      0.00         0
        3924       0.00      0.00      0.00         0
        4002       1.00      1.00      1.00         1
        4021       0.00      0.00      0.00         1
        4298       0.00      0.00      0.00         1
        4467       0.00      0.00      0.00         0
        4528       0.00      0.00      0.00         0
        4681       0.00      0.00      0.00         1
        4978       0.00      0.00      0.00         0
        5055       0.00      0.00      0.00         0
        5058       0.00      0.00      0.00         1
        5273       0.00      0.00      0.00         1
        5394       0.00      0.00      0.00         0
        5409       0.00      0.00      0.00         1
        5538       0.00      0.00      0.00         0
        6339       0.00      0.00      0.00         1
        6348       0.00      0.00      0.00         1
        6814       0.00      0.00      0.00         0
        7018       0.00      0.00      0.00         1
        7571       0.00      0.00      0.00         1
       10386       1.00      1.00      1.00         1
       10667       0.00      0.00      0.00         1
       10933       0.00      0.00      0.00         0
       13700       0.00      0.00      0.00         1
       16062       0.00      0.00      0.00         1
       17713       0.00      0.00      0.00         0

    accuracy                           0.10        30
   macro avg       0.07      0.07      0.07        30
weighted avg       0.10      0.10      0.10        30


C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\metrics\_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\metrics\_classification.py:1344:
UndefinedMetricWarning: Recall and F-score are ill-defined and being
set to 0.0 in labels with no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\metrics\_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples. Use
```

```
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\metrics\_classification.py:1344:
UndefinedMetricWarning: Recall and F-score are ill-defined and being
set to 0.0 in labels with no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\metrics\_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\metrics\_classification.py:1344:
UndefinedMetricWarning: Recall and F-score are ill-defined and being
set to 0.0 in labels with no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

RandomForestClassifier

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(x_train,y_train)

RandomForestClassifier()

rf_prediction= rf.predict(x_test)
rf_prediction

array([ 6814,  7407,  2218,  7407, 10933, 10386,  3924,  2957,  4998,
        6559, 17713, 17713,  4446,  3884,  3246,  3630,  2327,  3884,
        4002,  2998,  7407,  2064,  4139,  7407,  2957,  3525,  4978,
        2407,  3924,  4467], dtype=int64)

r2_score(y_test, rf_prediction)

0.8719745700106771
```

XGBRegressor

```
import xgboost as xgb
from xgboost import XGBRegressor
xgb = XGBRegressor()
xgb.fit(x_train, y_train)

XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
```

```
             enable_categorical=False, eval_metric=None,
feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None,
importance_type=None,
             interaction_constraints=None, learning_rate=None,
max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan,
monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...)

xgb.score(x_test,y_test)

0.8131223771178916

xgb_pred = xgb.predict(x_test)
xgb_pred

array([ 5864.9824,  6847.903 ,  2531.42  ,  6199.9346, 10934.468 ,
       10386.006 ,  3922.425 ,  2778.3894,  5131.116 ,  4746.484 ,
       17512.602 , 19294.99  ,  6111.677 ,  2720.2722,  3780.0007,
        3629.9832,  2026.7338,  2720.2722,  4002.0002,  5263.793 ,
        7559.809 ,  3638.4424,  3850.2095, 10673.777 ,  2778.3894,
        4099.5664,  4907.349 ,  2340.6355,  3922.425 ,  4267.6245],
      dtype=float32)

r2_score(y_test,xgb_pred)

0.8131223771178916
```

Model feature importances

```
features = xgb.feature_importances_
features

features_importance = pd.DataFrame([x.columns , features ])
features_importance

          0         1         2         3               4         5
0     Likes     Saves  Comments    Shares  Profile Visits   Follows
1  0.658916  0.035037  0.031506  0.001938        0.081422  0.191182
```