In [2]:
```
pip install pandas numpy plotly scikit-learn
```

```
Requirement already satisfied: pandas in c:\users\sheno\anaconda3\lib\site-packages (1.5.3)Note: you may need to restart the kernel to use updated packages.

Requirement already satisfied: numpy in c:\users\sheno\anaconda3\lib\site-packages (1.24.3)
Requirement already satisfied: plotly in c:\users\sheno\anaconda3\lib\site-packages (5.9.0)
Requirement already satisfied: scikit-learn in c:\users\sheno\anaconda3\lib\site-packages (1.2.2)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\sheno\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\sheno\anaconda3\lib\site-packages (from pandas) (2022.7)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\sheno\anaconda3\lib\site-packages (from plotly) (8.2.2)
Requirement already satisfied: scipy>=1.3.2 in c:\users\sheno\anaconda3\lib\site-packages (from scikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in c:\users\sheno\anaconda3\lib\site-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\sheno\anaconda3\lib\site-packages (from scikit-learn) (2.2.0)
Requirement already satisfied: six>=1.5 in c:\users\sheno\anaconda3\lib\site-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

In [3]:
```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go

data = pd.read_csv("transformed_data.csv")
data2 = pd.read_csv("raw_data.csv")
print(data)
```

```
      CODE      COUNTRY        DATE    HDI        TC        TD       STI  \
0      AFG  Afghanistan  2019-12-31  0.498  0.000000  0.000000  0.000000
1      AFG  Afghanistan  2020-01-01  0.498  0.000000  0.000000  0.000000
2      AFG  Afghanistan  2020-01-02  0.498  0.000000  0.000000  0.000000
3      AFG  Afghanistan  2020-01-03  0.498  0.000000  0.000000  0.000000
4      AFG  Afghanistan  2020-01-04  0.498  0.000000  0.000000  0.000000
...    ...          ...         ...    ...       ...       ...       ...
50413  ZWE     Zimbabwe  2020-10-15  0.535  8.994048  5.442418  4.341855
50414  ZWE     Zimbabwe  2020-10-16  0.535  8.996528  5.442418  4.341855
50415  ZWE     Zimbabwe  2020-10-17  0.535  8.999496  5.442418  4.341855
50416  ZWE     Zimbabwe  2020-10-18  0.535  9.000853  5.442418  4.341855
50417  ZWE     Zimbabwe  2020-10-19  0.535  9.005405  5.442418  4.341855

             POP    GDPCAP
0      17.477233  7.497754
1      17.477233  7.497754
2      17.477233  7.497754
3      17.477233  7.497754
4      17.477233  7.497754
...          ...       ...
50413  16.514381  7.549491
50414  16.514381  7.549491
50415  16.514381  7.549491
50416  16.514381  7.549491
50417  16.514381  7.549491

[50418 rows x 9 columns]
```

In [4]:
```
print(data2)
```

```
      iso_code     location        date  total_cases  total_deaths  \
0          AFG  Afghanistan  2019-12-31          0.0           0.0
1          AFG  Afghanistan  2020-01-01          0.0           0.0
2          AFG  Afghanistan  2020-01-02          0.0           0.0
3          AFG  Afghanistan  2020-01-03          0.0           0.0
4          AFG  Afghanistan  2020-01-04          0.0           0.0
...        ...          ...         ...          ...           ...
50413      ZWE     Zimbabwe  2020-10-15       8055.0         231.0
50414      ZWE     Zimbabwe  2020-10-16       8075.0         231.0
50415      ZWE     Zimbabwe  2020-10-17       8099.0         231.0
50416      ZWE     Zimbabwe  2020-10-18       8110.0         231.0
50417      ZWE     Zimbabwe  2020-10-19       8147.0         231.0

       stringency_index  population  gdp_per_capita  human_development_index  \
0                  0.00    38928341        1803.987                    0.498
1                  0.00    38928341        1803.987                    0.498
2                  0.00    38928341        1803.987                    0.498
3                  0.00    38928341        1803.987                    0.498
4                  0.00    38928341        1803.987                    0.498
...                 ...         ...             ...                      ...
50413             76.85    14862927        1899.775                    0.535
50414             76.85    14862927        1899.775                    0.535
50415             76.85    14862927        1899.775                    0.535
50416             76.85    14862927        1899.775                    0.535
50417             76.85    14862927        1899.775                    0.535

        Unnamed: 9  Unnamed: 10 Unnamed: 11  Unnamed: 12  Unnamed: 13
0            #NUM!        #NUM!       #NUM!    17.477233  7.497754494
1            #NUM!        #NUM!       #NUM!    17.477233  7.497754494
2            #NUM!        #NUM!       #NUM!    17.477233  7.497754494
3            #NUM!        #NUM!       #NUM!    17.477233  7.497754494
4            #NUM!        #NUM!       #NUM!    17.477233  7.497754494
...            ...          ...         ...          ...          ...
50413  8.994048296  5.442417711  4.34185547    16.514381  7.549490737
50414  8.996528148  5.442417711  4.34185547    16.514381  7.549490737
50415  8.999495876  5.442417711  4.34185547    16.514381  7.549490737
50416  9.000853147  5.442417711  4.34185547    16.514381  7.549490737
50417   9.00540504  5.442417711  4.34185547    16.514381  7.549490737

[50418 rows x 14 columns]
```

In [5]:
```
print(data.head())
```

```
  CODE      COUNTRY        DATE    HDI   TC   TD  STI        POP    GDPCAP
0  AFG  Afghanistan  2019-12-31  0.498  0.0  0.0  0.0  17.477233  7.497754
1  AFG  Afghanistan  2020-01-01  0.498  0.0  0.0  0.0  17.477233  7.497754
2  AFG  Afghanistan  2020-01-02  0.498  0.0  0.0  0.0  17.477233  7.497754
3  AFG  Afghanistan  2020-01-03  0.498  0.0  0.0  0.0  17.477233  7.497754
4  AFG  Afghanistan  2020-01-04  0.498  0.0  0.0  0.0  17.477233  7.497754
```

In [6]:
```
print(data2.head())
```

```
  iso_code     location        date  total_cases  total_deaths  \
0      AFG  Afghanistan  2019-12-31          0.0           0.0
1      AFG  Afghanistan  2020-01-01          0.0           0.0
2      AFG  Afghanistan  2020-01-02          0.0           0.0
3      AFG  Afghanistan  2020-01-03          0.0           0.0
4      AFG  Afghanistan  2020-01-04          0.0           0.0

   stringency_index  population  gdp_per_capita  human_development_index  \
0               0.0    38928341        1803.987                    0.498
1               0.0    38928341        1803.987                    0.498
2               0.0    38928341        1803.987                    0.498
3               0.0    38928341        1803.987                    0.498
4               0.0    38928341        1803.987                    0.498

  Unnamed: 9 Unnamed: 10 Unnamed: 11  Unnamed: 12  Unnamed: 13
0      #NUM!       #NUM!       #NUM!    17.477233  7.497754494
1      #NUM!       #NUM!       #NUM!    17.477233  7.497754494
2      #NUM!       #NUM!       #NUM!    17.477233  7.497754494
3      #NUM!       #NUM!       #NUM!    17.477233  7.497754494
4      #NUM!       #NUM!       #NUM!    17.477233  7.497754494
```

In [7]:
```
data["COUNTRY"].value_counts()
```

Out[7]:
```
Afghanistan        294
Indonesia          294
Macedonia          294
Luxembourg         294
Lithuania          294
                  ...
Tajikistan         172
Comoros            171
Lesotho            158
Hong Kong           51
Solomon Islands      4
Name: COUNTRY, Length: 210, dtype: int64
```

In [8]:
```
data["COUNTRY"].value_counts().mode()
```

Out[8]:
```
0    294
Name: COUNTRY, dtype: int64
```

In [9]:
```
# Aggregating the data

code = data["CODE"].unique().tolist()
country = data["COUNTRY"].unique().tolist()
hdi = []
tc = []
td = []
sti = []
population = data["POP"].unique().tolist()
gdp = []

for i in country:
    hdi.append((data.loc[data["COUNTRY"] == i, "HDI"]).sum()/294)
    tc.append((data2.loc[data2["location"] == i, "total_cases"]).sum())
    td.append((data2.loc[data2["location"] == i, "total_deaths"]).sum())
    sti.append((data.loc[data["COUNTRY"] == i, "STI"]).sum()/294)
    population.append((data2.loc[data2["location"] == i, "population"]).sum()/294)

aggregated_data = pd.DataFrame(list(zip(code, country, hdi, tc, td, sti, population)),
                               columns = ["Country Code", "Country", "HDI",
                                          "Total Cases", "Total Deaths",
                                          "Stringency Index", "Population"])
print(aggregated_data.head())
```

```
  Country Code      Country       HDI  Total Cases  Total Deaths  \
0          AFG  Afghanistan  0.498000    5126433.0      165875.0
1          ALB      Albania  0.600765    1071951.0       31056.0
2          DZA      Algeria  0.754000    4893999.0      206429.0
3          AND      Andorra  0.659551     223576.0        9850.0
4          AGO       Angola  0.418952     304005.0       11820.0

   Stringency Index  Population
0          3.049673   17.477233
1          3.005624   14.872537
2          3.195168   17.596309
3          2.677654   11.254996
4          2.965560   17.307957
```

In [10]:
```
# Sorting Data According to Total Cases
```

```
data = aggregated_data.sort_values(by=["Total Cases"], ascending=False)
print(data.head())
```

```
    Country Code        Country      HDI  Total Cases  Total Deaths  \
200          USA  United States  0.92400  746014098.0    26477574.0
27           BRA         Brazil  0.75900  425704517.0    14340567.0
90           IND          India  0.64000  407771615.0     7247327.0
157          RUS         Russia  0.81600  132888951.0     2131571.0
150          PER           Peru  0.59949   74882695.0     3020038.0

     Stringency Index  Population
200          3.350949   19.617637
27           3.136028   19.174732
90           3.610552   21.045353
157          3.380088   18.798668
150          3.430126   17.311165
```

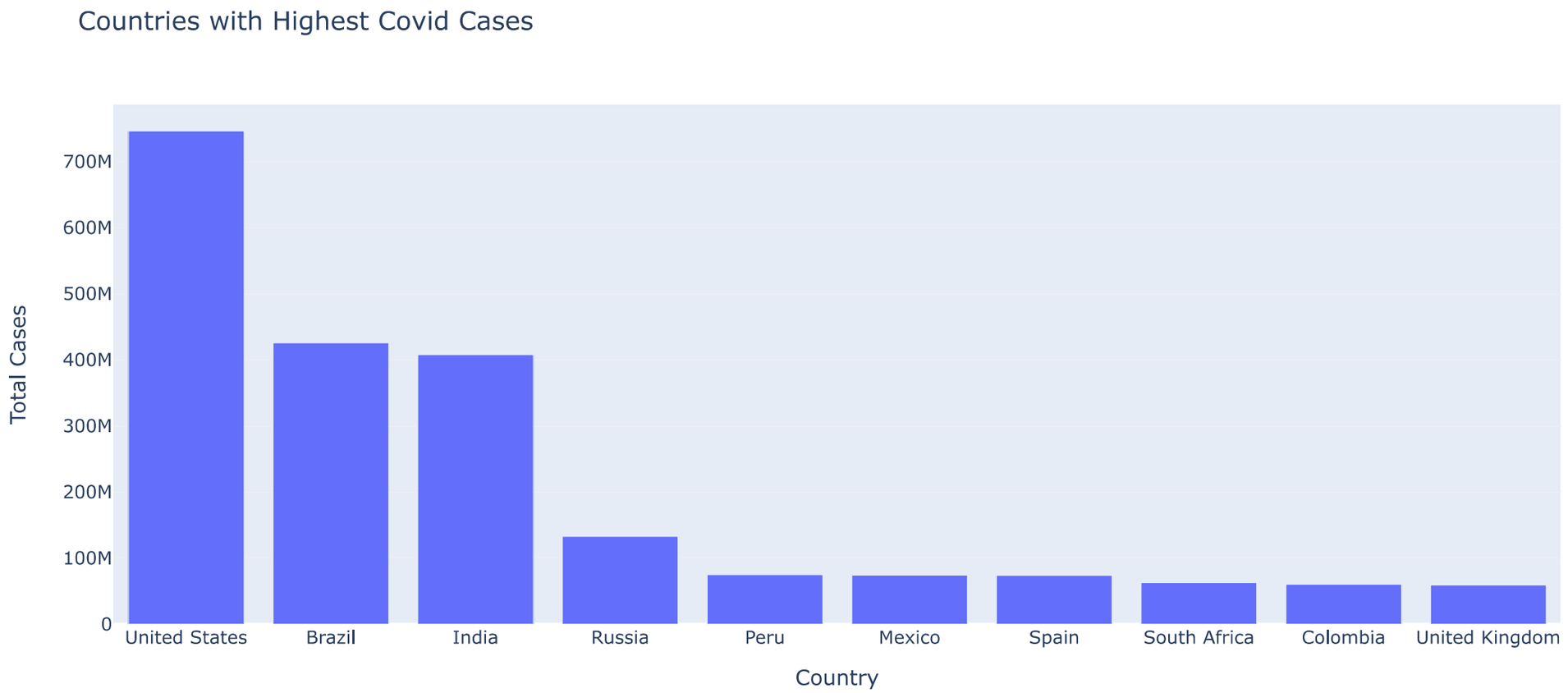
In [11]:
```
# Top 10 Countries with Highest Covid Cases

data = data.head(10)
print(data)
```

```
    Country Code         Country       HDI  Total Cases  Total Deaths  \
200          USA   United States  0.924000  746014098.0    26477574.0
27           BRA          Brazil  0.759000  425704517.0    14340567.0
90           IND           India  0.640000  407771615.0     7247327.0
157          RUS          Russia  0.816000  132888951.0     2131571.0
150          PER            Peru  0.599490   74882695.0     3020038.0
125          MEX          Mexico  0.774000   74347548.0     7295850.0
178          ESP           Spain  0.887969   73717676.0     5510624.0
175          ZAF    South Africa  0.608653   63027659.0     1357682.0
42           COL        Colombia  0.581847   60543682.0     1936134.0
199          GBR  United Kingdom  0.922000   59475032.0     7249573.0

     Stringency Index  Population
200          3.350949   19.617637
27           3.136028   19.174732
90           3.610552   21.045353
157          3.380088   18.798668
150          3.430126   17.311165
125          3.019289   18.674802
178          3.393922   17.660427
175          3.364333   17.898266
42           3.357923   17.745037
199          3.353883   18.033340
```

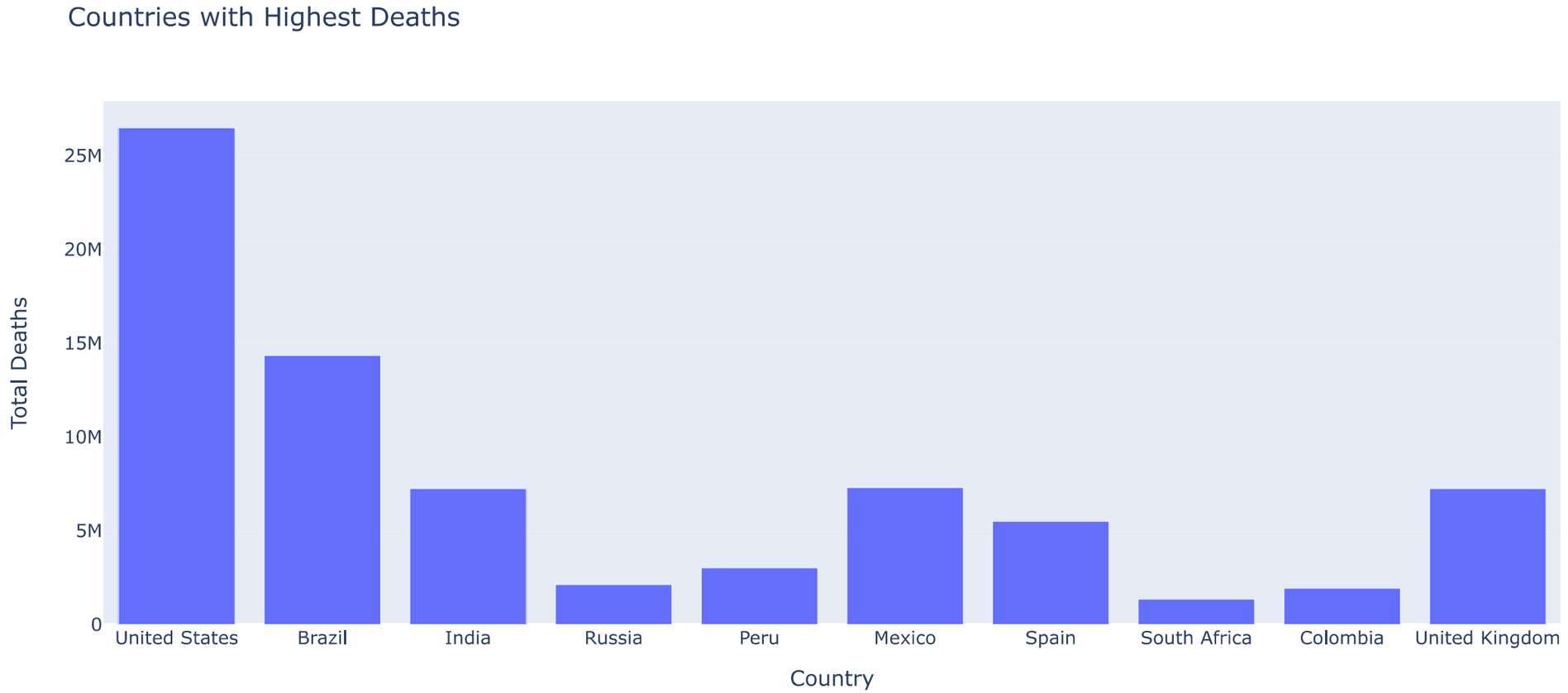
In [12]:
```
data["GDP Before Covid"] = [65279.53, 8897.49, 2100.75,
                            11497.65, 7027.61, 9946.03,
                            29564.74, 6001.40, 6424.98, 42354.41]
data["GDP During Covid"] = [63543.58, 6796.84, 1900.71,
                            10126.72, 6126.87, 8346.70,
                            27057.16, 5090.72, 5332.77, 40284.64]
print(data)
```

```
    Country Code         Country       HDI  Total Cases  Total Deaths  \
200          USA   United States  0.924000  746014098.0    26477574.0
27           BRA          Brazil  0.759000  425704517.0    14340567.0
90           IND           India  0.640000  407771615.0     7247327.0
157          RUS          Russia  0.816000  132888951.0     2131571.0
150          PER            Peru  0.599490   74882695.0     3020038.0
125          MEX          Mexico  0.774000   74347548.0     7295850.0
178          ESP           Spain  0.887969   73717676.0     5510624.0
175          ZAF    South Africa  0.608653   63027659.0     1357682.0
42           COL        Colombia  0.581847   60543682.0     1936134.0
199          GBR  United Kingdom  0.922000   59475032.0     7249573.0

     Stringency Index  Population  GDP Before Covid  GDP During Covid
200          3.350949   19.617637          65279.53          63543.58
27           3.136028   19.174732           8897.49           6796.84
90           3.610552   21.045353           2100.75           1900.71
157          3.380088   18.798668          11497.65          10126.72
150          3.430126   17.311165           7027.61           6126.87
125          3.019289   18.674802           9946.03           8346.70
178          3.393922   17.660427          29564.74          27057.16
175          3.364333   17.898266           6001.40           5090.72
42           3.357923   17.745037           6424.98           5332.77
199          3.353883   18.033340          42354.41          40284.64
```

In [13]:
```
figure = px.bar(data, y='Total Cases', x='Country',
           title="Countries with Highest Covid Cases")
figure.show()
```

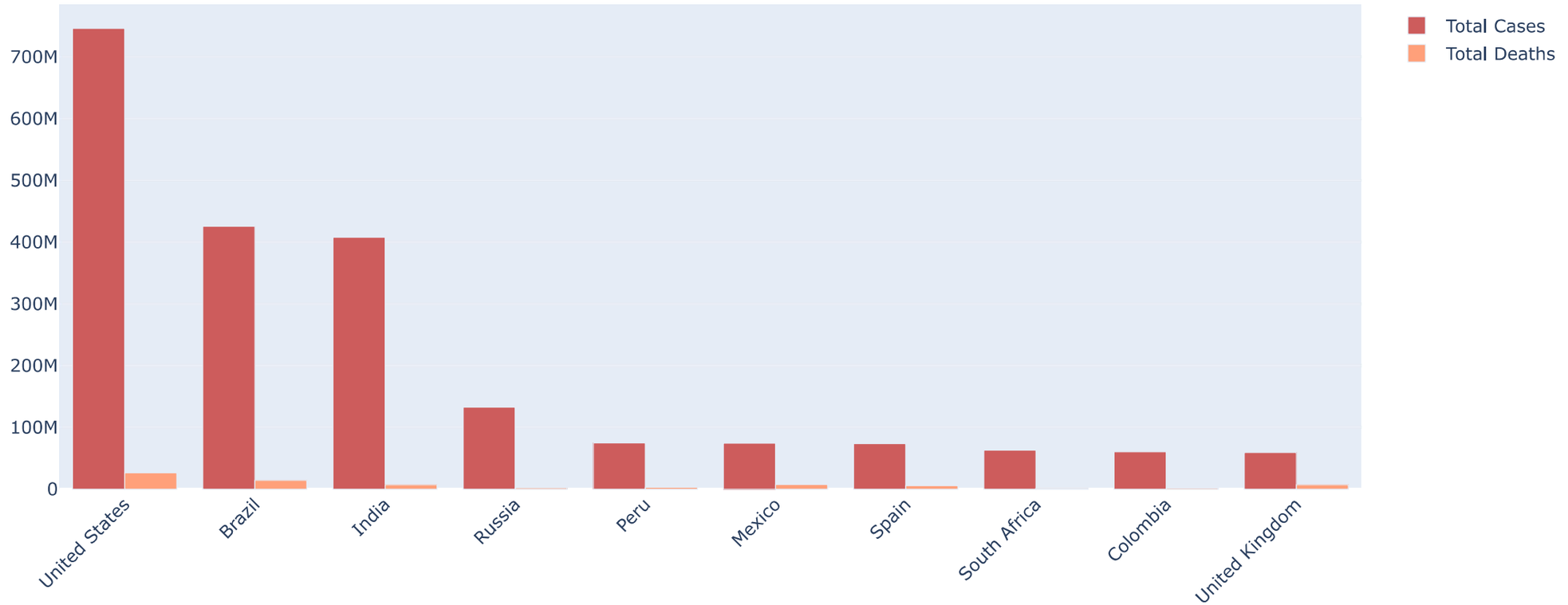


In [14]:
```
figure = px.bar(data, y='Total Deaths', x='Country',
           title="Countries with Highest Deaths")
figure.show()
```



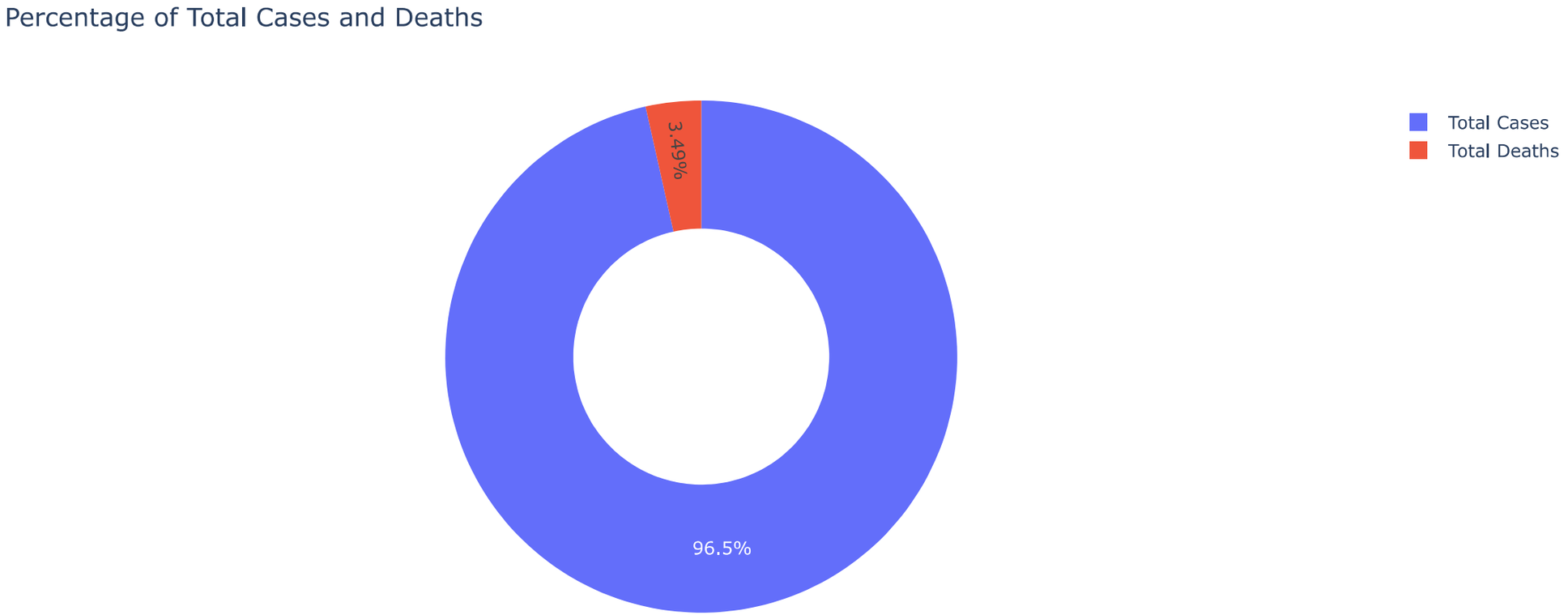
In [15]:
```
fig = go.Figure()
fig.add_trace(go.Bar(
    x=data["Country"],
    y=data["Total Cases"],
    name='Total Cases',
    marker_color='indianred'
))
fig.add_trace(go.Bar(
    x=data["Country"],
    y=data["Total Deaths"],
    name='Total Deaths',
```

```
    marker_color='lightsalmon'
))
fig.update_layout(barmode='group', xaxis_tickangle=-45)
fig.show()
```



In [16]:
```
# Percentage of Total Cases and Deaths
cases = data["Total Cases"].sum()
deceased = data["Total Deaths"].sum()

labels = ["Total Cases", "Total Deaths"]
values = [cases, deceased]
```

```
fig = px.pie(data, values=values, names=labels,
             title='Percentage of Total Cases and Deaths', hole=0.5)
fig.show()
```

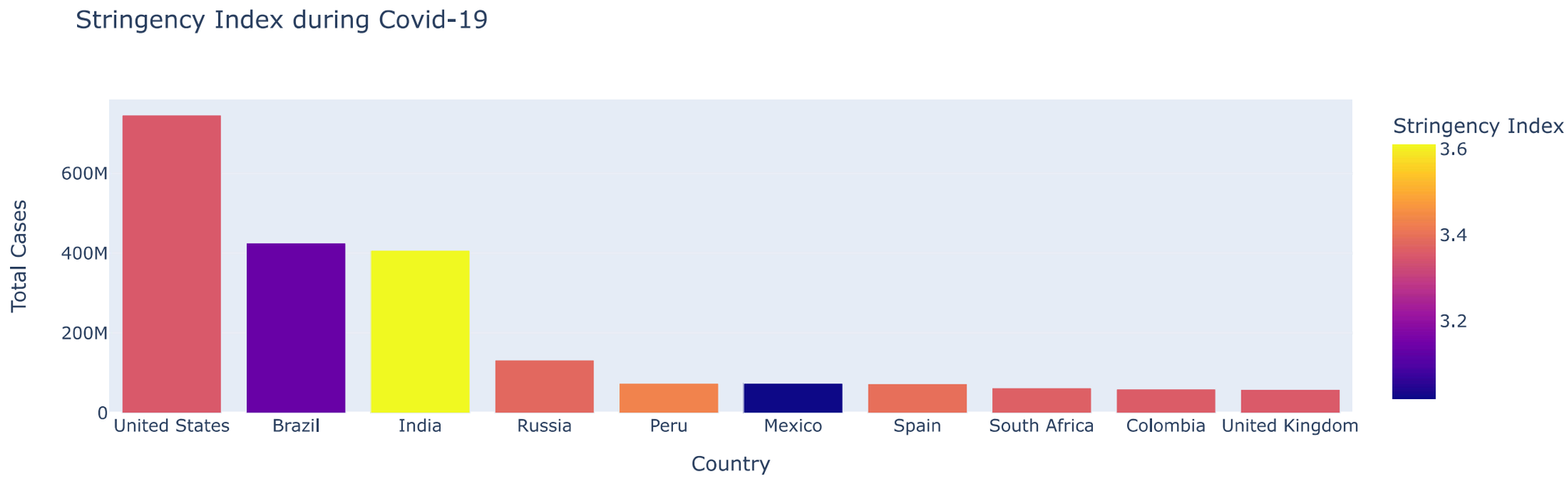


In [17]:
```
death_rate = (data["Total Deaths"].sum() / data["Total Cases"].sum()) * 100
print("Death Rate = ", death_rate)
```

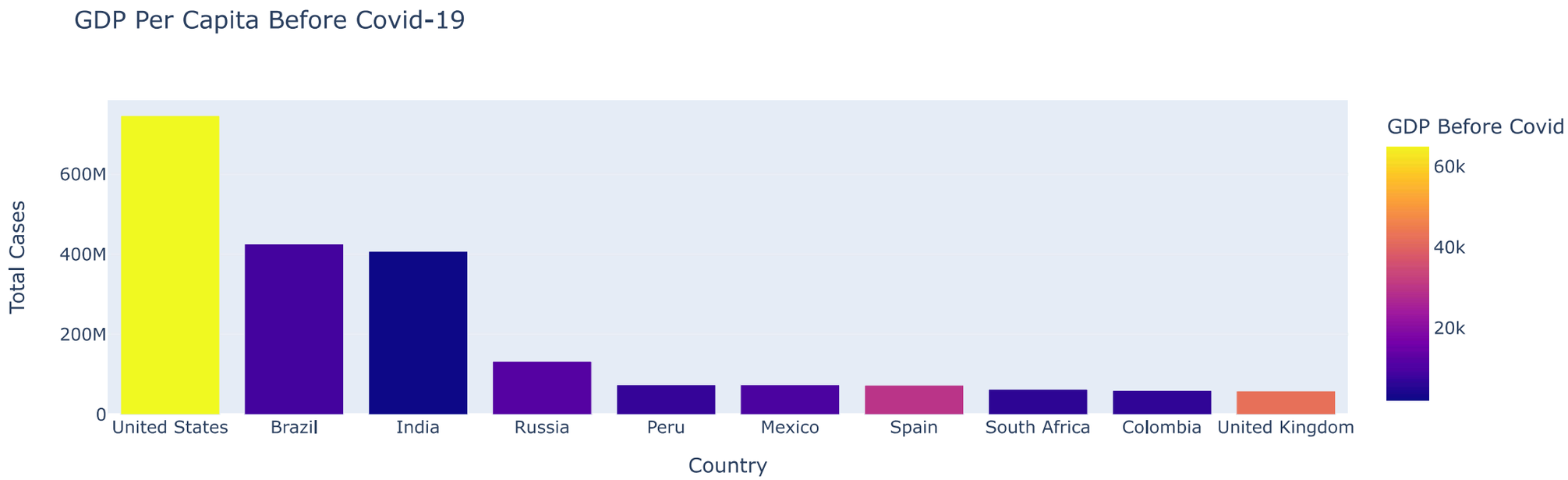
```
Death Rate =  3.6144212045653767
```

In [18]:
```
fig = px.bar(data, x='Country', y='Total Cases',
             hover_data=['Population', 'Total Deaths'],
             color='Stringency Index', height=400,
```

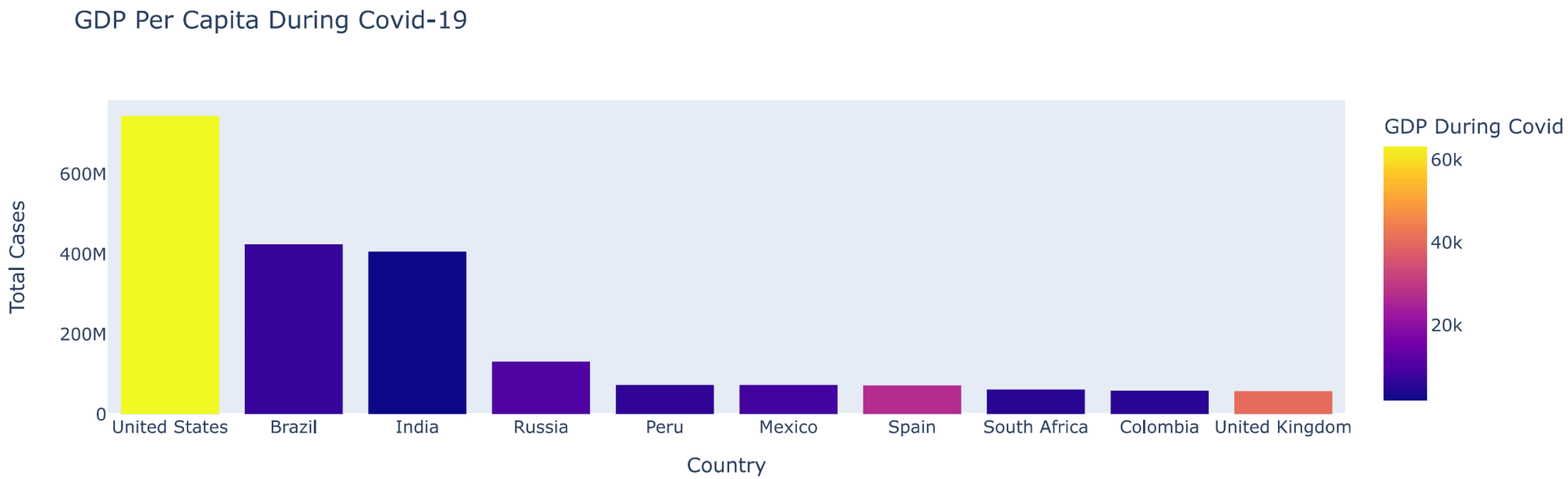
```
            title= "Stringency Index during Covid-19")
fig.show()
```



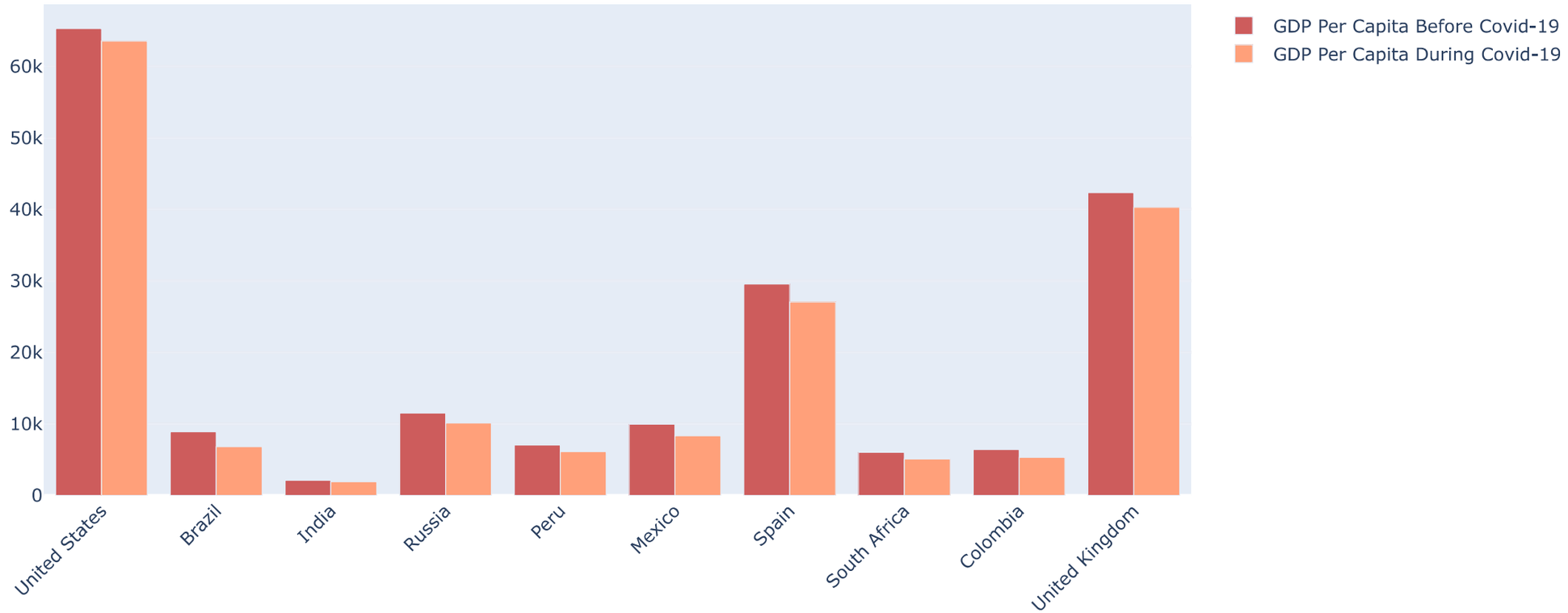
In [19]:
```
fig = px.bar(data, x='Country', y='Total Cases',
             hover_data=['Population', 'Total Deaths'],
             color='GDP Before Covid', height=400,
             title="GDP Per Capita Before Covid-19")
fig.show()
```



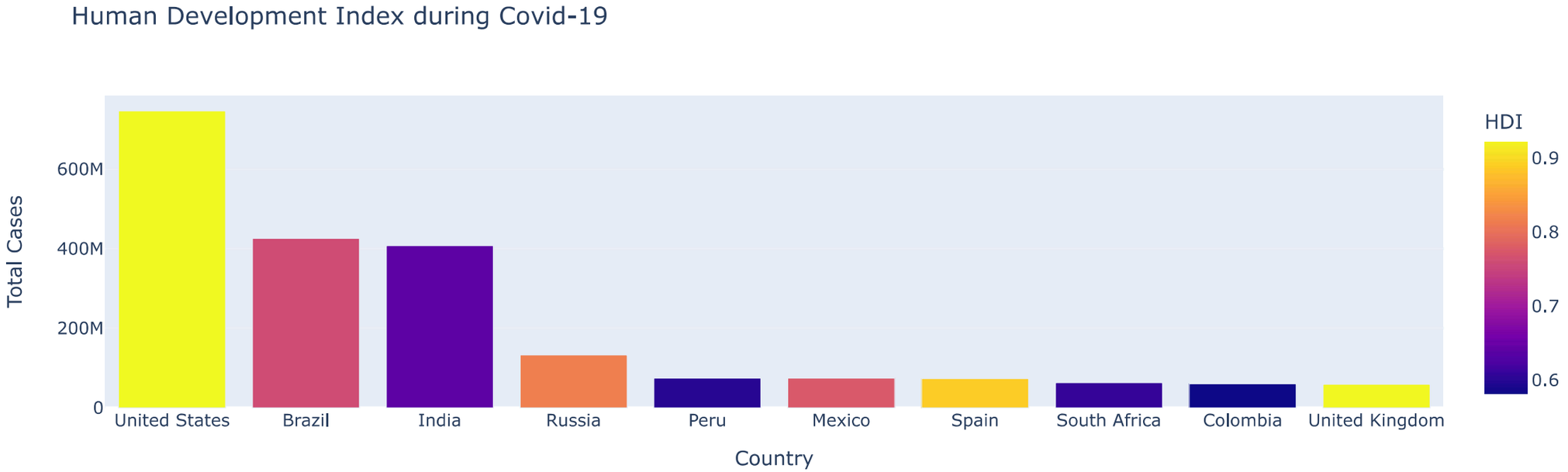
In [20]:
```
fig = px.bar(data, x='Country', y='Total Cases',
             hover_data=['Population', 'Total Deaths'],
             color='GDP During Covid', height=400,
             title="GDP Per Capita During Covid-19")
fig.show()
```



In [21]:
```
fig = go.Figure()
fig.add_trace(go.Bar(
    x=data["Country"],
    y=data["GDP Before Covid"],
    name='GDP Per Capita Before Covid-19',
    marker_color='indianred'
))
fig.add_trace(go.Bar(
    x=data["Country"],
    y=data["GDP During Covid"],
    name='GDP Per Capita During Covid-19',
    marker_color='lightsalmon'
))
fig.update_layout(barmode='group', xaxis_tickangle=-45)
fig.show()
```



In [22]:
```
fig = px.bar(data, x='Country', y='Total Cases',
             hover_data=['Population', 'Total Deaths'],
             color='HDI', height=400,
             title="Human Development Index during Covid-19")
fig.show()
```


Human Development Index during Covid-19
Total Cases
600M
400M
200M
0
United States
Brazil
India
Russia
Peru
Mexico
Spain
South Africa
Colombia
United Kingdom
Country
HDI
0.9
0.8
0.7
0.6

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
```

In [4]:
```
data = pd.read_csv("dailyActivity_merged.csv")
```

In [5]:
```
print(data.head())
```

```
           Id ActivityDate  TotalSteps  TotalDistance  TrackerDistance  \
0  1503960366    4/12/2016       13162           8.50             8.50
1  1503960366    4/13/2016       10735           6.97             6.97
2  1503960366    4/14/2016       10460           6.74             6.74
3  1503960366    4/15/2016        9762           6.28             6.28
4  1503960366    4/16/2016       12669           8.16             8.16

   LoggedActivitiesDistance  VeryActiveDistance  ModeratelyActiveDistance  \
0                       0.0                1.88                      0.55
1                       0.0                1.57                      0.69
2                       0.0                2.44                      0.40
3                       0.0                2.14                      1.26
4                       0.0                2.71                      0.41

   LightActiveDistance  SedentaryActiveDistance  VeryActiveMinutes  \
0                 6.06                      0.0                 25
1                 4.71                      0.0                 21
2                 3.91                      0.0                 30
3                 2.83                      0.0                 29
4                 5.04                      0.0                 36

   FairlyActiveMinutes  LightlyActiveMinutes  SedentaryMinutes  Calories
0                   13                   328               728      1985
1                   19                   217               776      1797
2                   11                   181              1218      1776
3                   34                   209               726      1745
4                   10                   221               773      1863
```

In [6]:
```
print(data.isnull().sum())
```

```
Id                          0
ActivityDate                0
TotalSteps                  0
TotalDistance               0
TrackerDistance             0
LoggedActivitiesDistance    0
VeryActiveDistance          0
ModeratelyActiveDistance    0
LightActiveDistance         0
SedentaryActiveDistance     0
VeryActiveMinutes           0
FairlyActiveMinutes         0
LightlyActiveMinutes        0
SedentaryMinutes            0
Calories                    0
dtype: int64
```

In [7]:
```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 940 entries, 0 to 939
Data columns (total 15 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Id                        940 non-null    int64  
 1   ActivityDate              940 non-null    object 
 2   TotalSteps                940 non-null    int64  
 3   TotalDistance             940 non-null    float64
 4   TrackerDistance           940 non-null    float64
 5   LoggedActivitiesDistance  940 non-null    float64
 6   VeryActiveDistance        940 non-null    float64
 7   ModeratelyActiveDistance  940 non-null    float64
 8   LightActiveDistance       940 non-null    float64
 9   SedentaryActiveDistance   940 non-null    float64
 10  VeryActiveMinutes         940 non-null    int64  
 11  FairlyActiveMinutes       940 non-null    int64  
 12  LightlyActiveMinutes      940 non-null    int64  
 13  SedentaryMinutes          940 non-null    int64  
 14  Calories                  940 non-null    int64  
dtypes: float64(7), int64(7), object(1)
memory usage: 110.3+ KB
None
```

In [8]:
```
data["ActivityDate"] = pd.to_datetime(data["ActivityDate"], format="%m/%d/%Y")
```

In [9]:
```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 940 entries, 0 to 939
Data columns (total 15 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Id                        940 non-null    int64
 1   ActivityDate              940 non-null    datetime64[ns]
 2   TotalSteps                940 non-null    int64
 3   TotalDistance             940 non-null    float64
 4   TrackerDistance           940 non-null    float64
 5   LoggedActivitiesDistance  940 non-null    float64
 6   VeryActiveDistance        940 non-null    float64
 7   ModeratelyActiveDistance  940 non-null    float64
 8   LightActiveDistance       940 non-null    float64
 9   SedentaryActiveDistance   940 non-null    float64
 10  VeryActiveMinutes         940 non-null    int64
 11  FairlyActiveMinutes       940 non-null    int64
 12  LightlyActiveMinutes      940 non-null    int64
 13  SedentaryMinutes          940 non-null    int64
 14  Calories                  940 non-null    int64
dtypes: datetime64[ns](1), float64(7), int64(7)
memory usage: 110.3 KB
None
```

In [10]:
```
data["TotalMinutes"] = data["VeryActiveMinutes"] + data["FairlyActiveMinutes"] + data["LightlyActiveMinutes"] + data["SedentaryMinutes"]
```

In [11]:
```
print(data["TotalMinutes"].sample(5))
```

```
769    1440
301    1440
569     986
317     984
527    1027
Name: TotalMinutes, dtype: int64
```

In [12]:
```
print(data.describe())
```

	Id	TotalSteps	TotalDistance	TrackerDistance
count	9.400000e+02	940.000000	940.000000	940.000000
mean	4.855407e+09	7637.910638	5.489702	5.475351
std	2.424805e+09	5087.150742	3.924606	3.907276
min	1.503960e+09	0.000000	0.000000	0.000000
25%	2.320127e+09	3789.750000	2.620000	2.620000
50%	4.445115e+09	7405.500000	5.245000	5.245000
75%	6.962181e+09	10727.000000	7.712500	7.710000
max	8.877689e+09	36019.000000	28.030001	28.030001

	LoggedActivitiesDistance	VeryActiveDistance	ModeratelyActiveDistance
count	940.000000	940.000000	940.000000
mean	0.108171	1.502681	0.567543
std	0.619897	2.658941	0.883580
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.000000	0.210000	0.240000
75%	0.000000	2.052500	0.800000
max	4.942142	21.920000	6.480000

	LightActiveDistance	SedentaryActiveDistance	VeryActiveMinutes
count	940.000000	940.000000	940.000000
mean	3.340819	0.001606	21.164894
std	2.040655	0.007346	32.844803
min	0.000000	0.000000	0.000000
25%	1.945000	0.000000	0.000000
50%	3.365000	0.000000	4.000000
75%	4.782500	0.000000	32.000000
max	10.710000	0.110000	210.000000

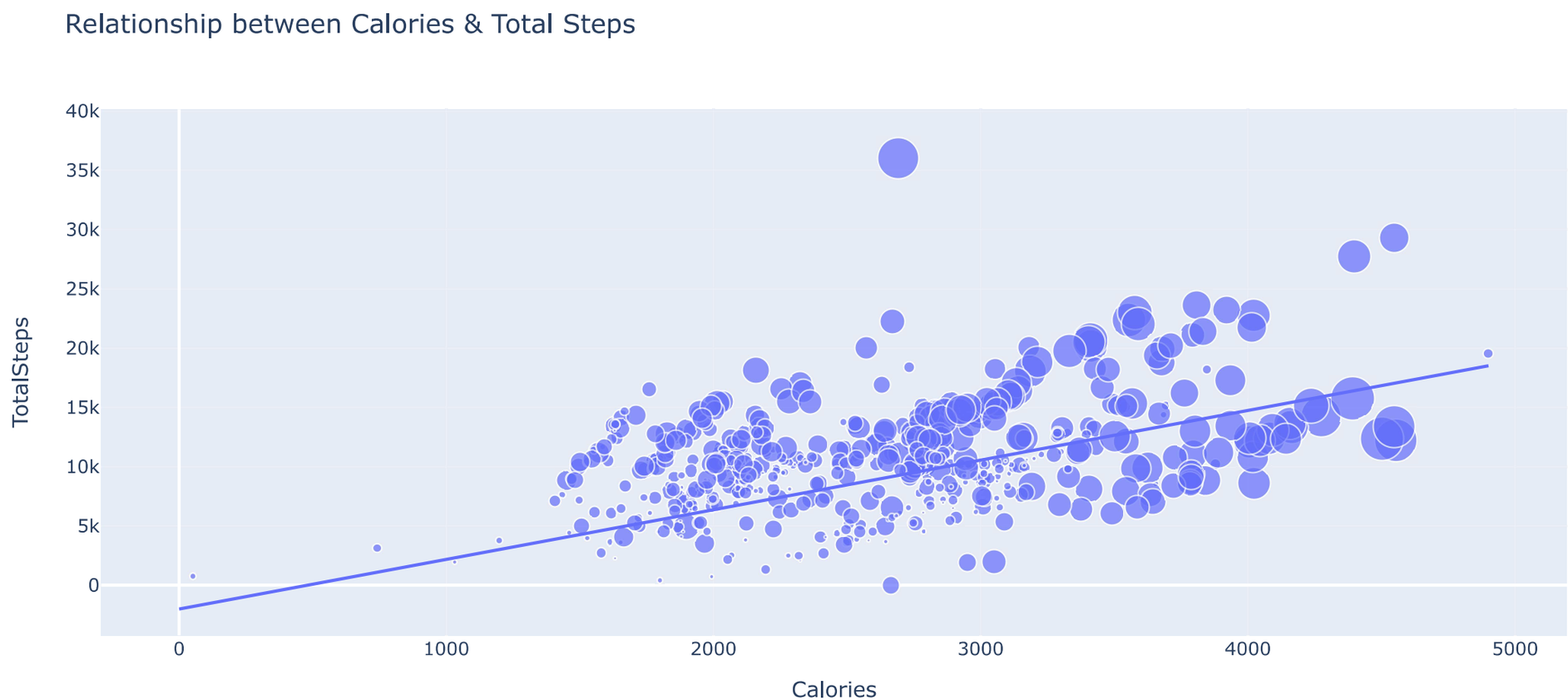
	FairlyActiveMinutes	LightlyActiveMinutes	SedentaryMinutes
count	940.000000	940.000000	940.000000
mean	13.564894	192.812766	991.210638
std	19.987404	109.174700	301.267437
min	0.000000	0.000000	0.000000
25%	0.000000	127.000000	729.750000
50%	6.000000	199.000000	1057.500000
75%	19.000000	264.000000	1229.500000
max	143.000000	518.000000	1440.000000

	Calories	TotalMinutes
count	940.000000	940.000000
mean	2303.609574	1218.753191
min	0.000000	2.000000

```
25%    1828.500000    989.750000
50%    2134.000000   1440.000000
75%    2793.250000   1440.000000
max    4900.000000   1440.000000
```

In [13]:
```
figure = px.scatter(data_frame = data, x ="Calories", y="TotalSteps", size="VeryActiveMinutes", trendline="ols", title="Relationship between Calories & Total St
```

In [14]:
```
figure.show()
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

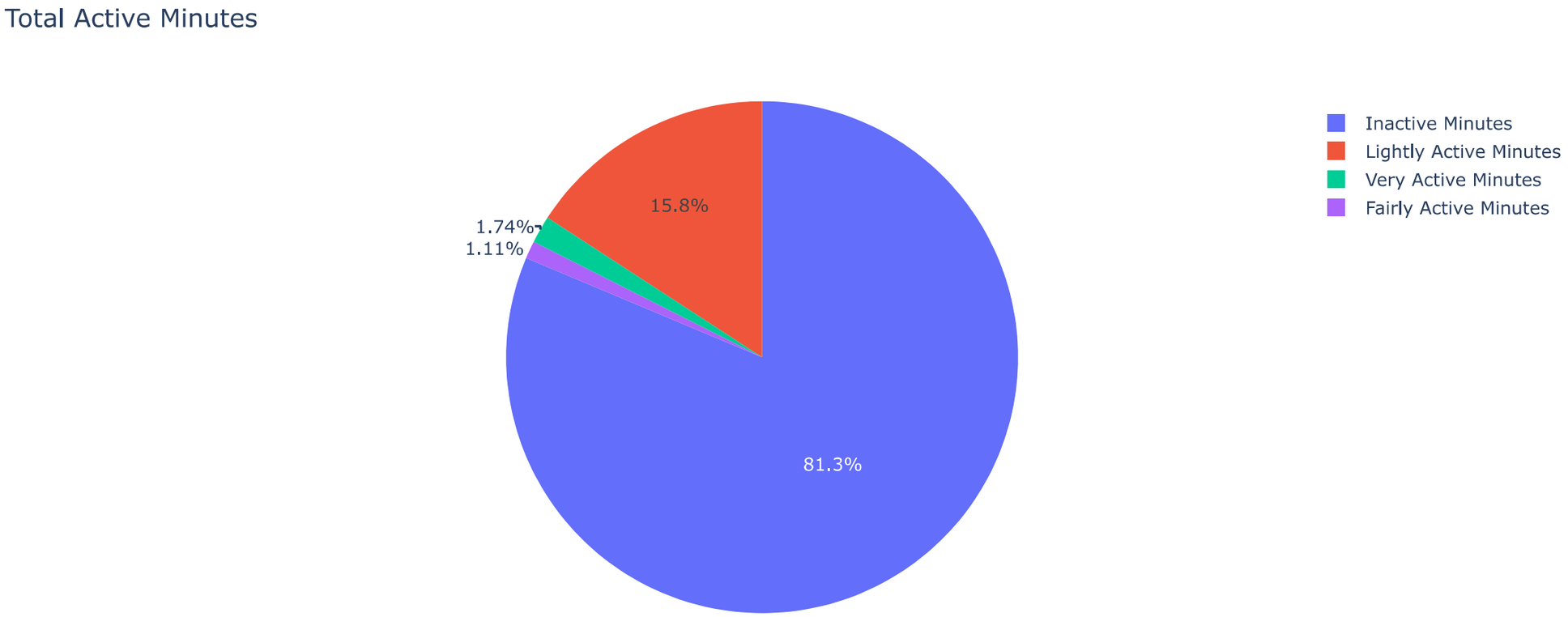
In [15]:
```
label = ["Very Active Minutes", "Fairly Active Minutes",
         "Lightly Active Minutes", "Inactive Minutes"]
```

In [16]:
```
counts = data[["VeryActiveMinutes", "FairlyActiveMinutes",
               "LightlyActiveMinutes", "SedentaryMinutes"]].mean()
```

In [17]:
```
colors = ['gold','lightgreen', "pink", "blue"]
```

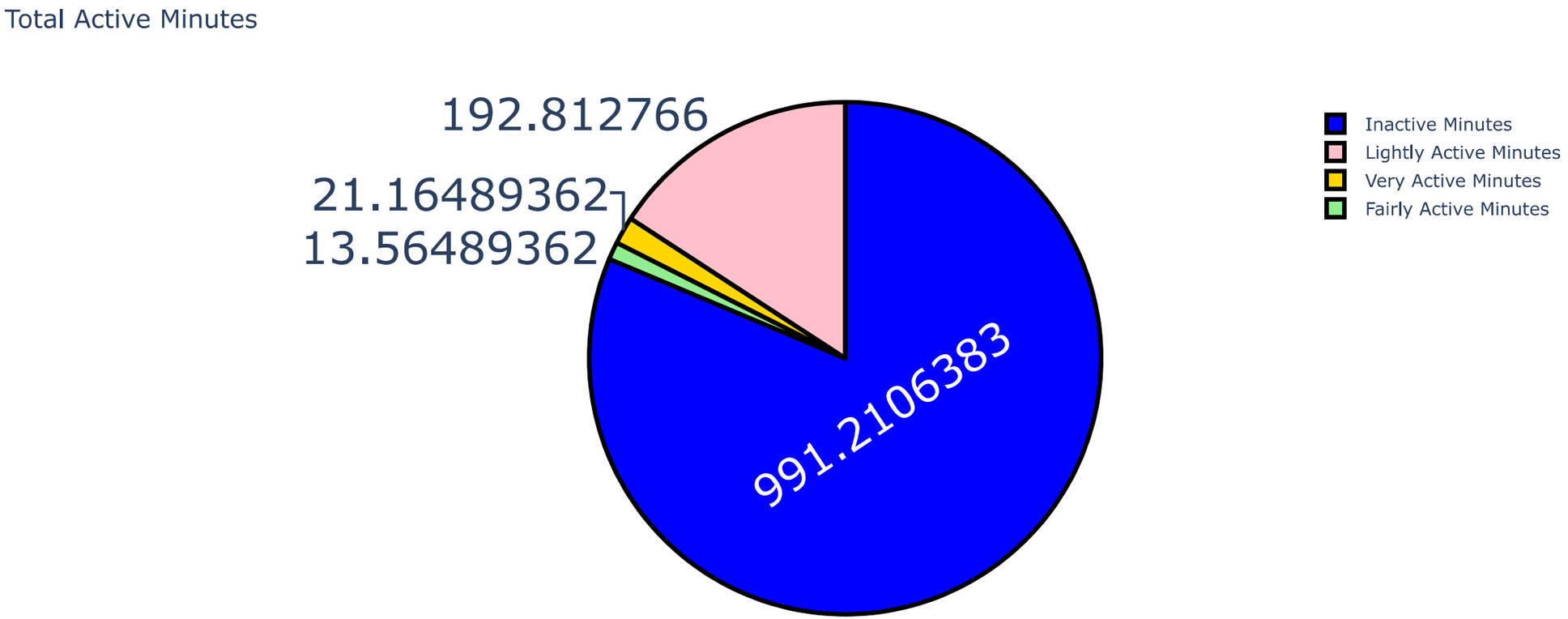
In [18]:
```
fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
```

In [19]:
```
fig.update_layout(title_text='Total Active Minutes')
```

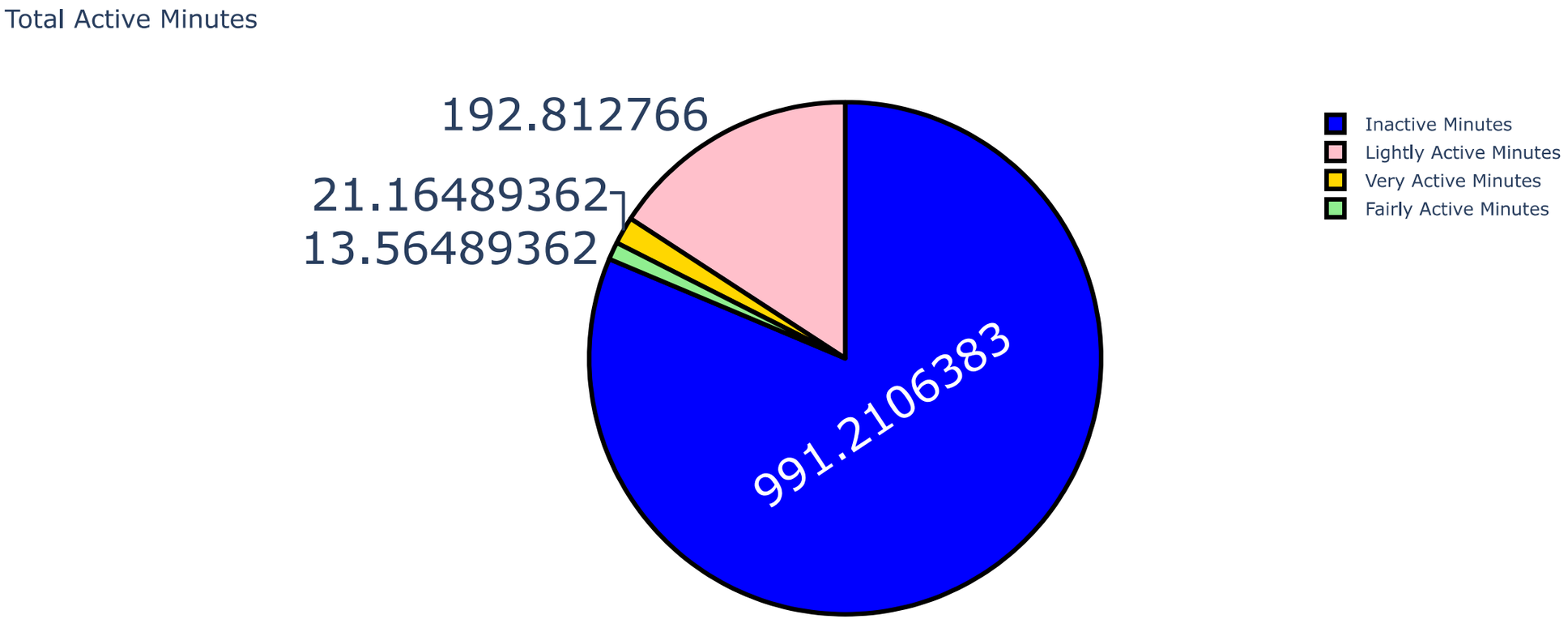


In [20]:
```
fig.update_traces(hoverinfo='label+percent', textinfo='value', textfont_size=30,
                  marker=dict(colors=colors, line=dict(color='black', width=3)))
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [21]:
```
fig.show()
```



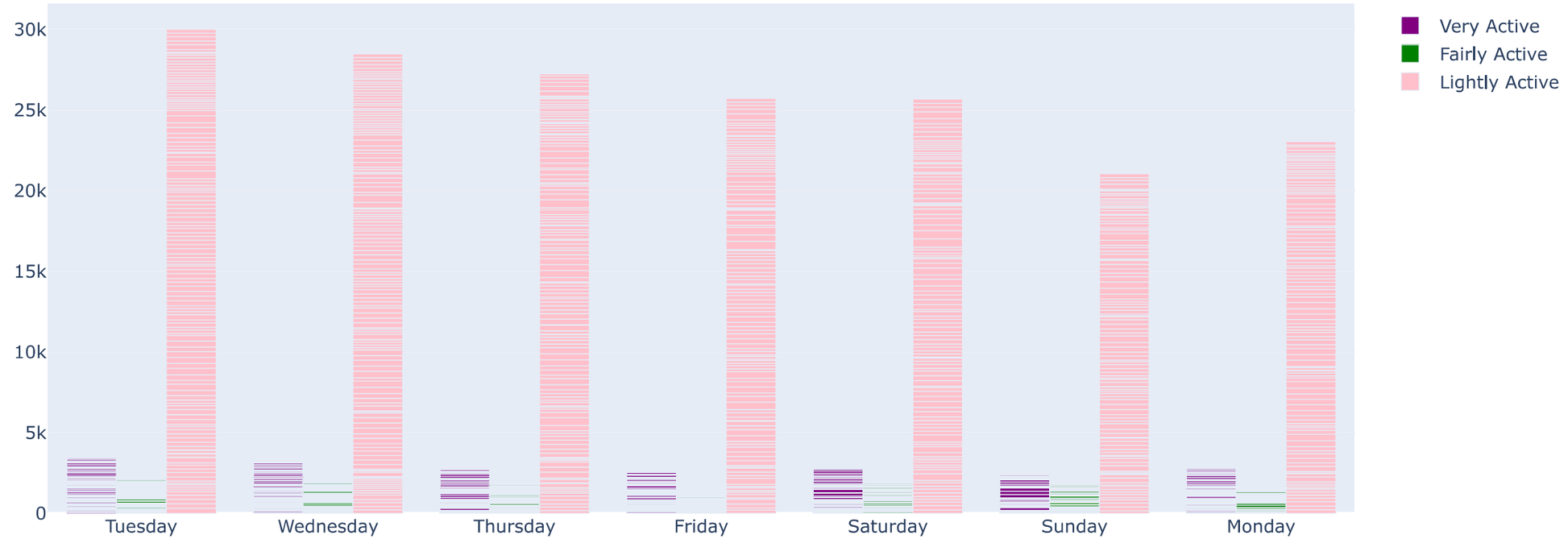
In [22]:
```
data["Day"] = data["ActivityDate"].dt.day_name()
print(data["Day"].head())
```

```
0      Tuesday
1    Wednesday
2     Thursday
3       Friday
4     Saturday
Name: Day, dtype: object
```

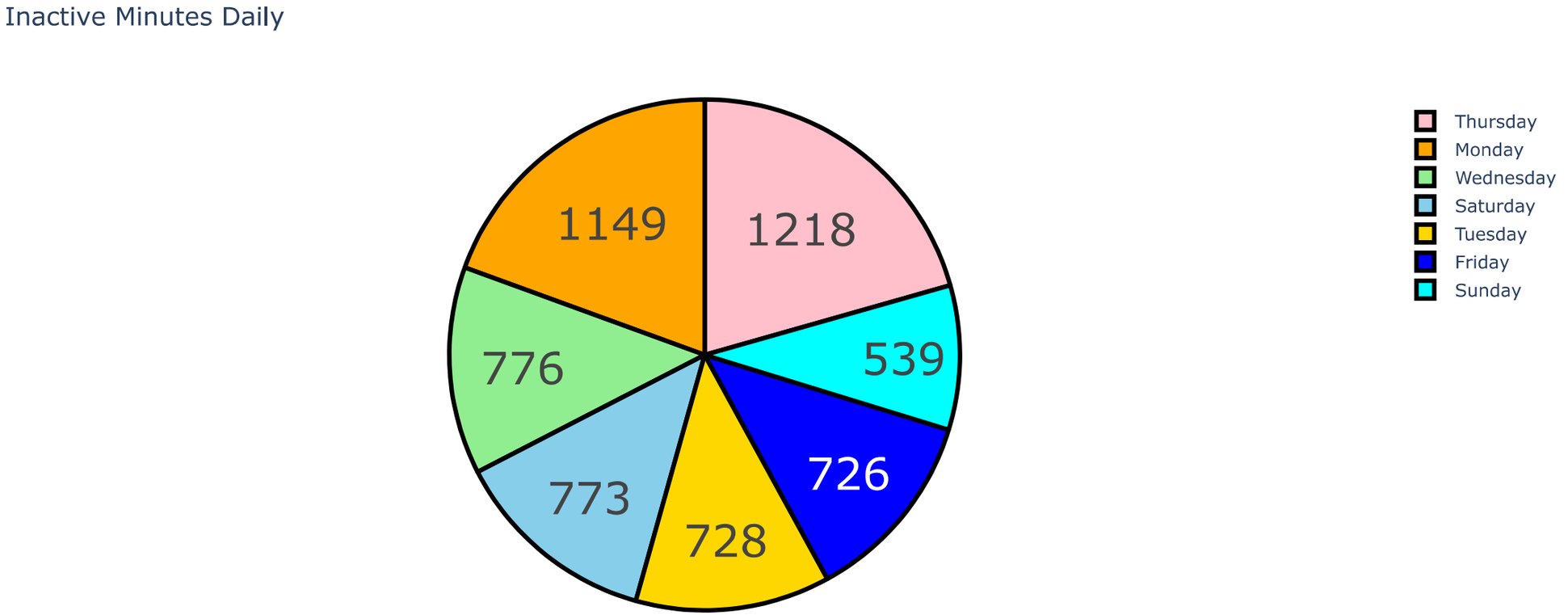
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
fig.add_trace(go.Bar(
```

```
    x=data["Day"],
    y=data["VeryActiveMinutes"],
    name='Very Active',
    marker_color='purple'
))
fig.add_trace(go.Bar(
    x=data["Day"],
    y=data["FairlyActiveMinutes"],
    name='Fairly Active',
    marker_color='green'
))
fig.add_trace(go.Bar(
    x=data["Day"],
    y=data["LightlyActiveMinutes"],
    name='Lightly Active',
    marker_color='pink'
))
fig.update_layout(barmode='group', xaxis_tickangle=0)
fig.show()
```

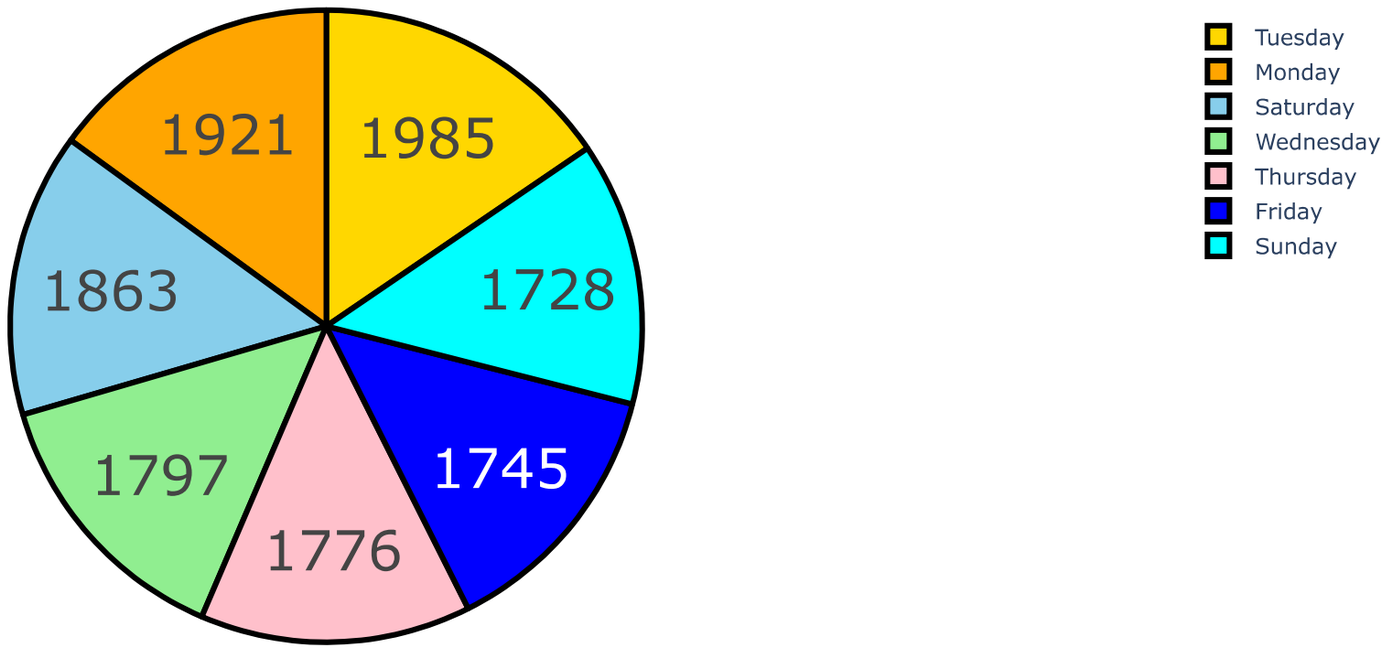


In [24]:
```
day = data["Day"].value_counts()
label = day.index
counts = data["SedentaryMinutes"]
colors = ['gold','lightgreen', "pink", "blue", "skyblue", "cyan", "orange"]

fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
fig.update_layout(title_text='Inactive Minutes Daily')
fig.update_traces(hoverinfo='label+percent', textinfo='value', textfont_size=30,
                  marker=dict(colors=colors, line=dict(color='black', width=3)))
fig.show()
```



In [25]:
```
calories = data["Day"].value_counts()
label = calories.index
counts = data["Calories"]
colors = ['gold','lightgreen', "pink", "blue", "skyblue", "cyan", "orange"]

fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
fig.update_layout(title_text='Calories Burned Daily')
fig.update_traces(hoverinfo='label+percent', textinfo='value', textfont_size=30,
                  marker=dict(colors=colors, line=dict(color='black', width=3)))
fig.show()
```

Calories Burned Daily



In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

data = pd.read_csv("https://raw.githubusercontent.com/amankharwal/Website-data/master/flipkart_reviews.csv")
print(data.head())
```

```
                                     Product_name  \
0  Lenovo Ideapad Gaming 3 Ryzen 5 Hexa Core 5600...
1  Lenovo Ideapad Gaming 3 Ryzen 5 Hexa Core 5600...
2  Lenovo Ideapad Gaming 3 Ryzen 5 Hexa Core 5600...
3  DELL Inspiron Athlon Dual Core 3050U - (4 GB/2...
4  DELL Inspiron Athlon Dual Core 3050U - (4 GB/2...

                                          Review  Rating
0  Best under 60k Great performanceI got it for a...       5
1                                Good perfomence...       5
2  Great performance but usually it has also that...       5
3          My wife is so happy and best product ✌️😘       5
4  Light weight laptop with new amazing features,...       5
```

In [2]:
```
print(data.isnull().sum())
```

```
Product_name    0
Review          0
Rating          0
dtype: int64
```

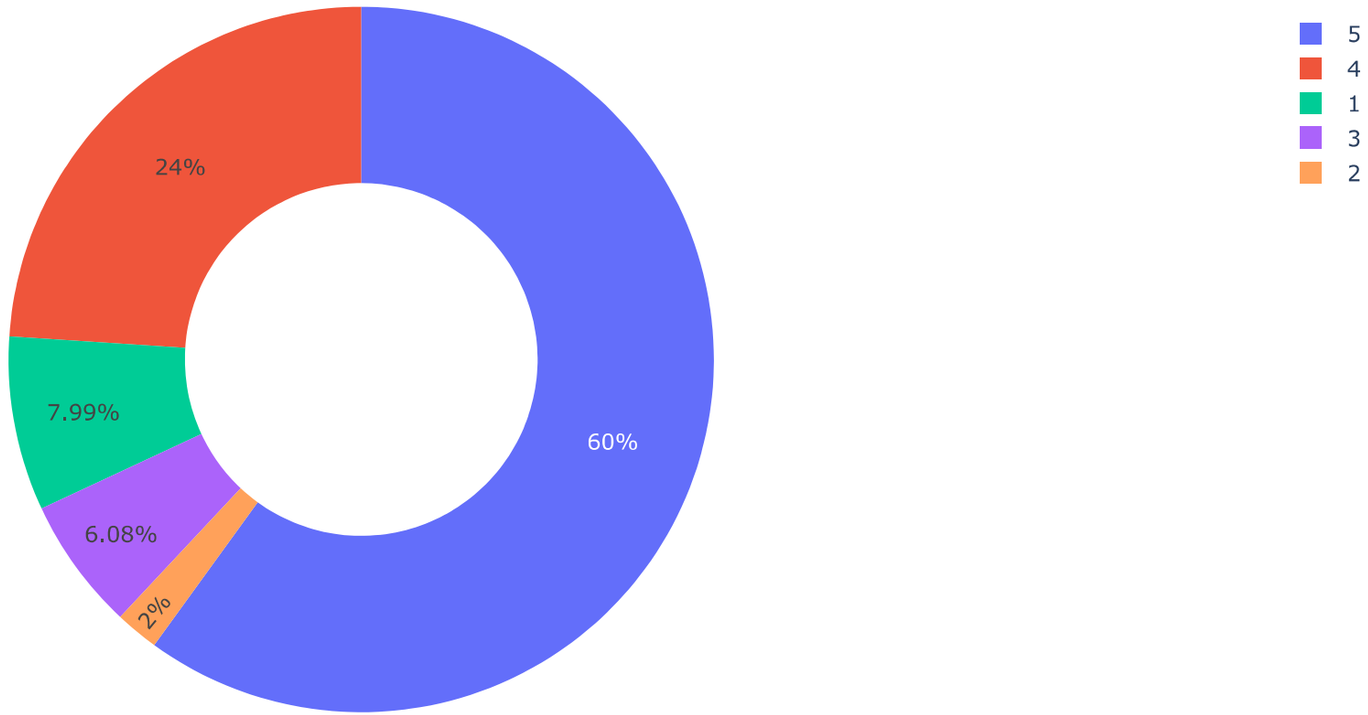
In [3]:
```
import nltk
import re
nltk.download('stopwords')
stemmer = nltk.SnowballStemmer("english")
from nltk.corpus import stopwords
import string
stopword=set(stopwords.words('english'))

def clean(text):
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
```

```
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = [word for word in text.split(' ') if word not in stopword]
    text=" ".join(text)
    text = [stemmer.stem(word) for word in text.split(' ')]
    text=" ".join(text)
    return text
data["Review"] = data["Review"].apply(clean)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\sheno\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [4]:

```
ratings = data["Rating"].value_counts()
numbers = ratings.index
quantity = ratings.values

import plotly.express as px
figure = px.pie(data,
             values=quantity,
             names=numbers,hole = 0.5)
figure.show()
```



In [5]:
```
text = " ".join(i for i in data.Review)
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords,
                      background_color="white").generate(text)
plt.figure( figsize=(15,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

In [6]:
```
nltk.download('vader_lexicon')
sentiments = SentimentIntensityAnalyzer()
data["Positive"] = [sentiments.polarity_scores(i)["pos"] for i in data["Review"]]
data["Negative"] = [sentiments.polarity_scores(i)["neg"] for i in data["Review"]]
data["Neutral"] = [sentiments.polarity_scores(i)["neu"] for i in data["Review"]]
data = data[["Review", "Positive", "Negative", "Neutral"]]
print(data.head())
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\sheno\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

	Review	Positive	Negative	Neutral
0	best great performancei got around backup bi...	0.395	0.101	0.504
1	good perfom	0.744	0.000	0.256
2	great perform usual also game laptop issu batt...	0.277	0.000	0.723
3	wife happi best product 👌 😘	0.512	0.000	0.488
4	light weight laptop new amaz featur batteri li...	0.000	0.000	1.000

In [7]:
```
x = sum(data["Positive"])
y = sum(data["Negative"])
z = sum(data["Neutral"])

def sentiment_score(a, b, c):
    if (a>b) and (a>c):
        print("Positive 😊 ")
    elif (b>a) and (b>c):
        print("Negative 😠 ")
    else:
        print("Neutral 🙂 ")
sentiment_score(x, y, z)
```

```
Neutral 🙂
```

In [8]:
```
print("Positive: ", x)
print("Negative: ", y)
print("Neutral: ", z)
```

```
Positive:  923.5529999999985
Negative:  96.77500000000013
Neutral:  1283.6880000000006
```

In [7]:
```
pip install pandas numpy matplotlib seaborn plotly wordcloud scikit-learn
```

```
Requirement already satisfied: pandas in c:\users\sheno\anaconda3\lib\site-packages (1.5.3)
Requirement already satisfied: numpy in c:\users\sheno\anaconda3\lib\site-packages (1.24.3)
Requirement already satisfied: matplotlib in c:\users\sheno\anaconda3\lib\site-packages (3.7.1)
Requirement already satisfied: seaborn in c:\users\sheno\anaconda3\lib\site-packages (0.12.2)
Requirement already satisfied: plotly in c:\users\sheno\anaconda3\lib\site-packages (5.9.0)
Collecting wordcloud
  Downloading wordcloud-1.9.2-cp311-cp311-win_amd64.whl (151 kB)
                                        0.0/151.4 kB ? eta -:--:--
     --------------------------          112.6/151.4 kB 3.3 MB/s eta 0:00:01
     ---------------------------------- 151.4/151.4 kB 2.3 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn in c:\users\sheno\anaconda3\lib\site-packages (1.2.2)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\sheno\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\sheno\anaconda3\lib\site-packages (from pandas) (2022.7)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\sheno\anaconda3\lib\site-packages (from matplotlib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\sheno\anaconda3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\sheno\anaconda3\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\sheno\anaconda3\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\sheno\anaconda3\lib\site-packages (from matplotlib) (23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\sheno\anaconda3\lib\site-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\sheno\anaconda3\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\sheno\anaconda3\lib\site-packages (from plotly) (8.2.2)
Requirement already satisfied: scipy>=1.3.2 in c:\users\sheno\anaconda3\lib\site-packages (from scikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in c:\users\sheno\anaconda3\lib\site-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\sheno\anaconda3\lib\site-packages (from scikit-learn) (2.2.0)
Requirement already satisfied: six>=1.5 in c:\users\sheno\anaconda3\lib\site-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
Installing collected packages: wordcloud
Successfully installed wordcloud-1.9.2
Note: you may need to restart the kernel to use updated packages.
```

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
from sklearn.model_selection import train_test_split
from sklearn.linear_model import PassiveAggressiveRegressor
```

In [3]:
```
data = pd.read_csv("Instagram data.csv", encoding = 'latin1')
print(data.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...
```

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

Out[4]:
```
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
```

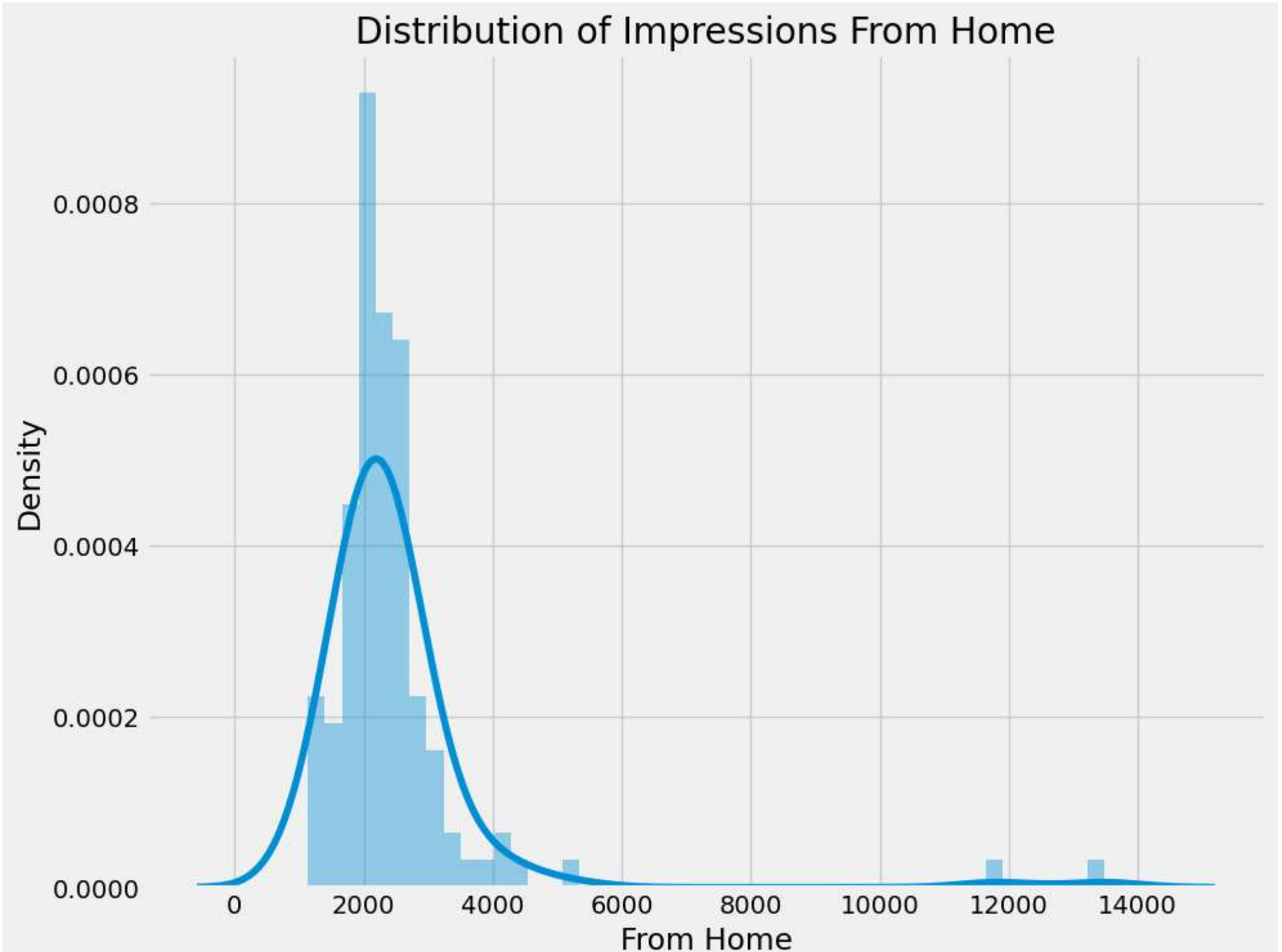
In [6]:
```
data = data.dropna()
```

In [7]:
```
data.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
```

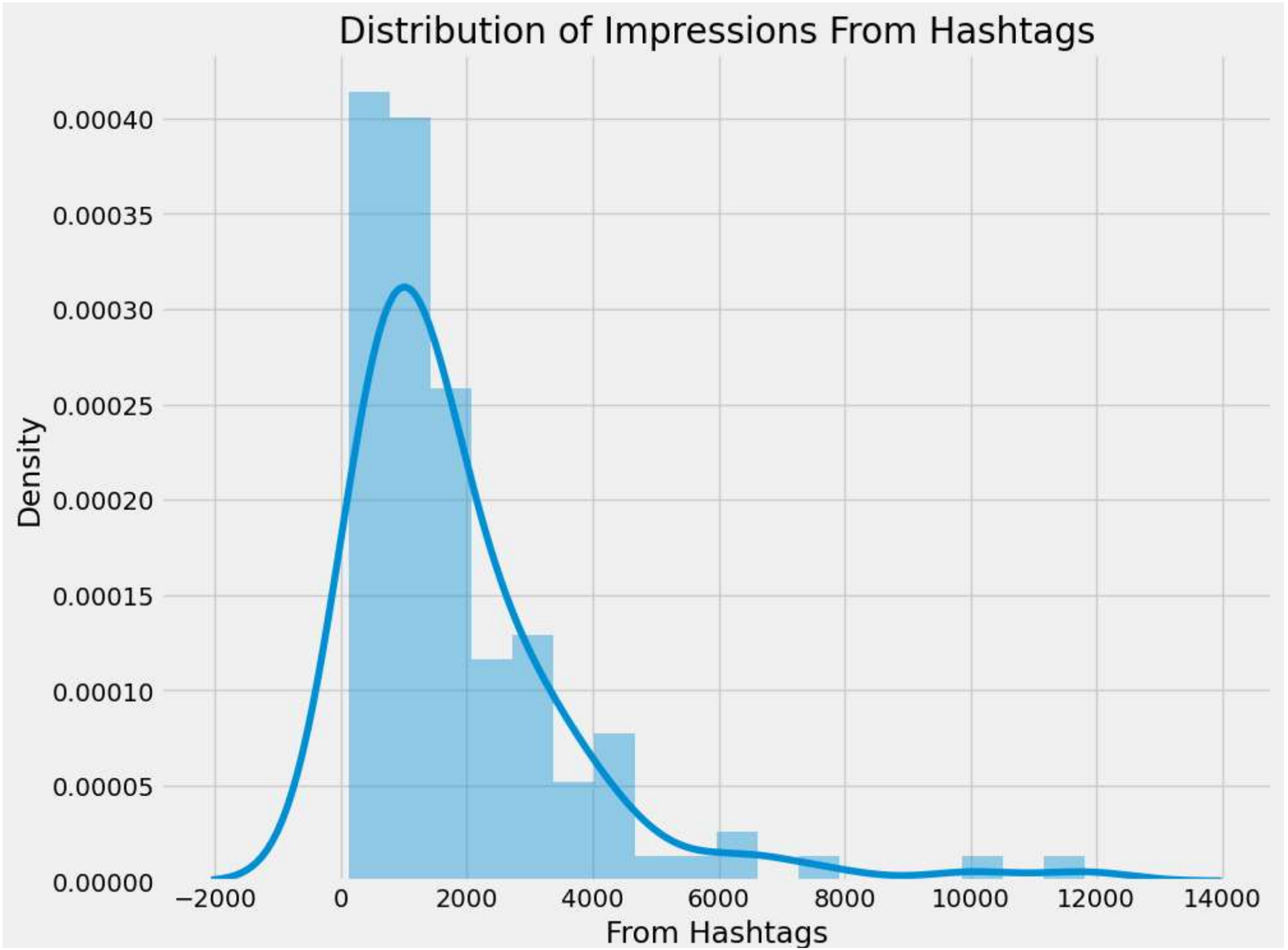
In [11]:
```
plt.figure(figsize=(10, 8))
plt.style.use('fivethirtyeight')
plt.title("Distribution of Impressions From Home")
sns.distplot(data['From Home'])
plt.show()
```

```
C:\Users\sheno\AppData\Local\Temp\ipykernel_9120\4127021947.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(data['From Home'])
```


Distribution of Impressions From Home
Density
0.0008
0.0006
0.0004
0.0002
0.0000
0
2000
4000
6000
8000
10000
12000
14000
From Home

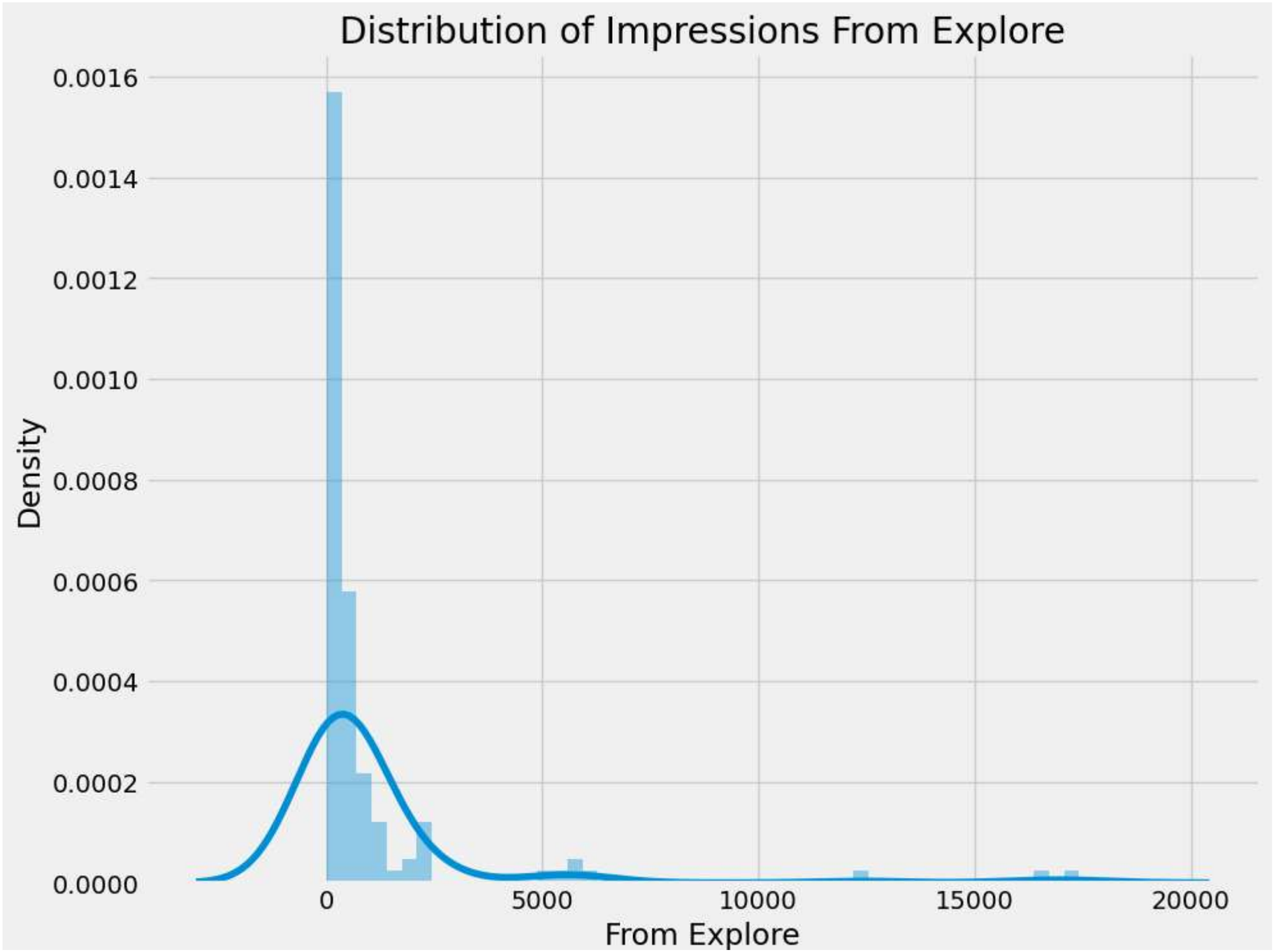
In [12]:
```
plt.figure(figsize=(10, 8))
plt.title("Distribution of Impressions From Hashtags")
sns.distplot(data['From Hashtags'])
plt.show()
```

```
C:\Users\sheno\AppData\Local\Temp\ipykernel_9120\671336047.py:3: 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(data['From Hashtags'])
```

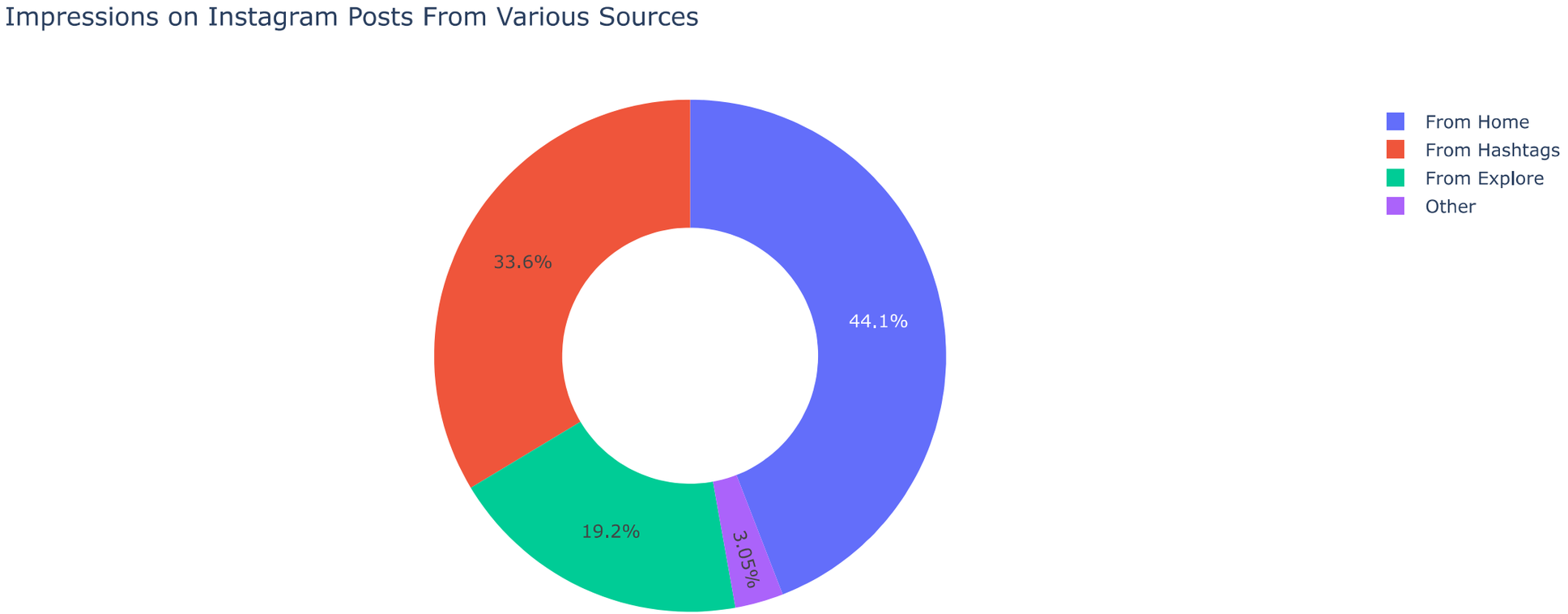

Distribution of Impressions From Hashtags
Density
0.00040
0.00035
0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
−2000
0
2000
4000
6000
8000
10000
12000
14000
From Hashtags

In [13]:
```
plt.figure(figsize=(10, 8))
plt.title("Distribution of Impressions From Explore")
sns.distplot(data['From Explore'])
plt.show()
```

```
C:\Users\sheno\AppData\Local\Temp\ipykernel_9120\2610232579.py:3: 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(data['From Explore'])
```

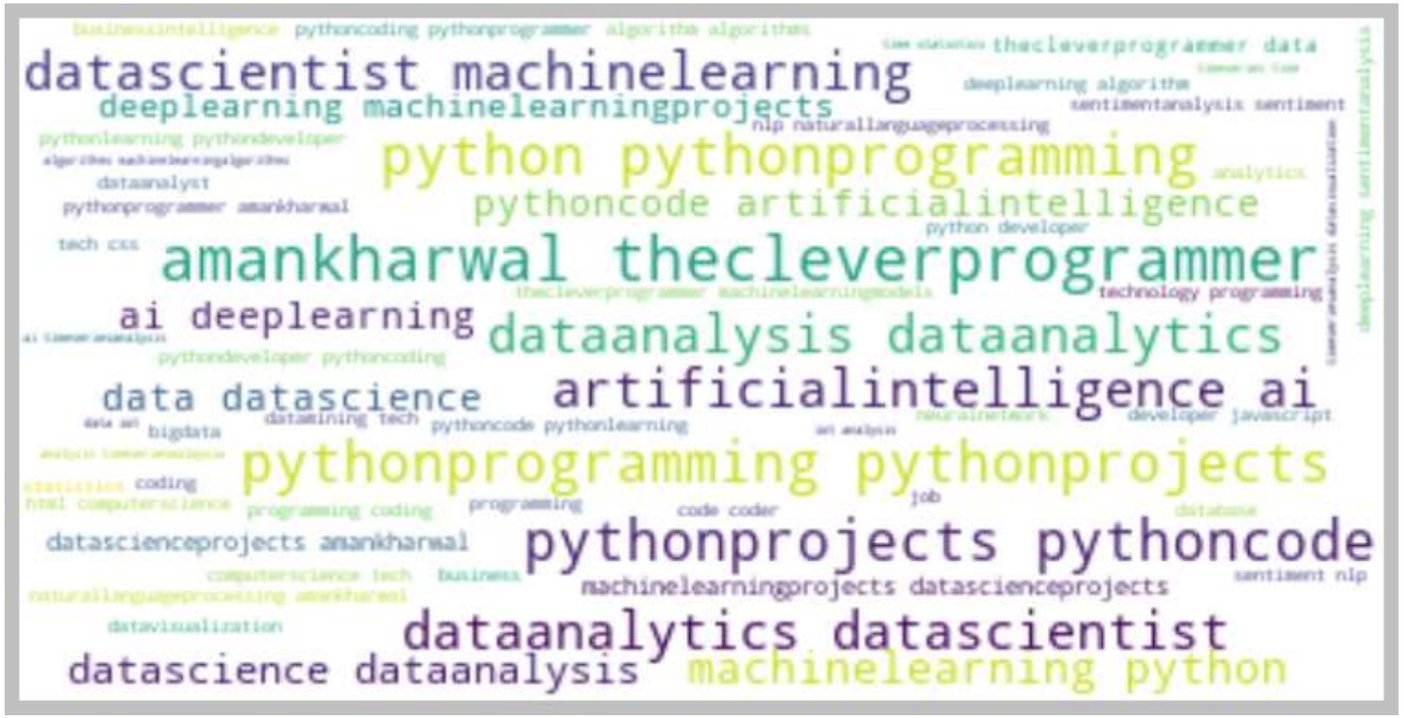

Distribution of Impressions From Explore
Density
0.0016
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
0
5000
10000
15000
20000
From Explore

In [14]:
```
home = data["From Home"].sum()
hashtags = data["From Hashtags"].sum()
explore = data["From Explore"].sum()
other = data["From Other"].sum()

labels = ['From Home','From Hashtags','From Explore','Other']
values = [home, hashtags, explore, other]

fig = px.pie(data, values=values, names=labels,
             title='Impressions on Instagram Posts From Various Sources', hole=0.5)
fig.show()
```

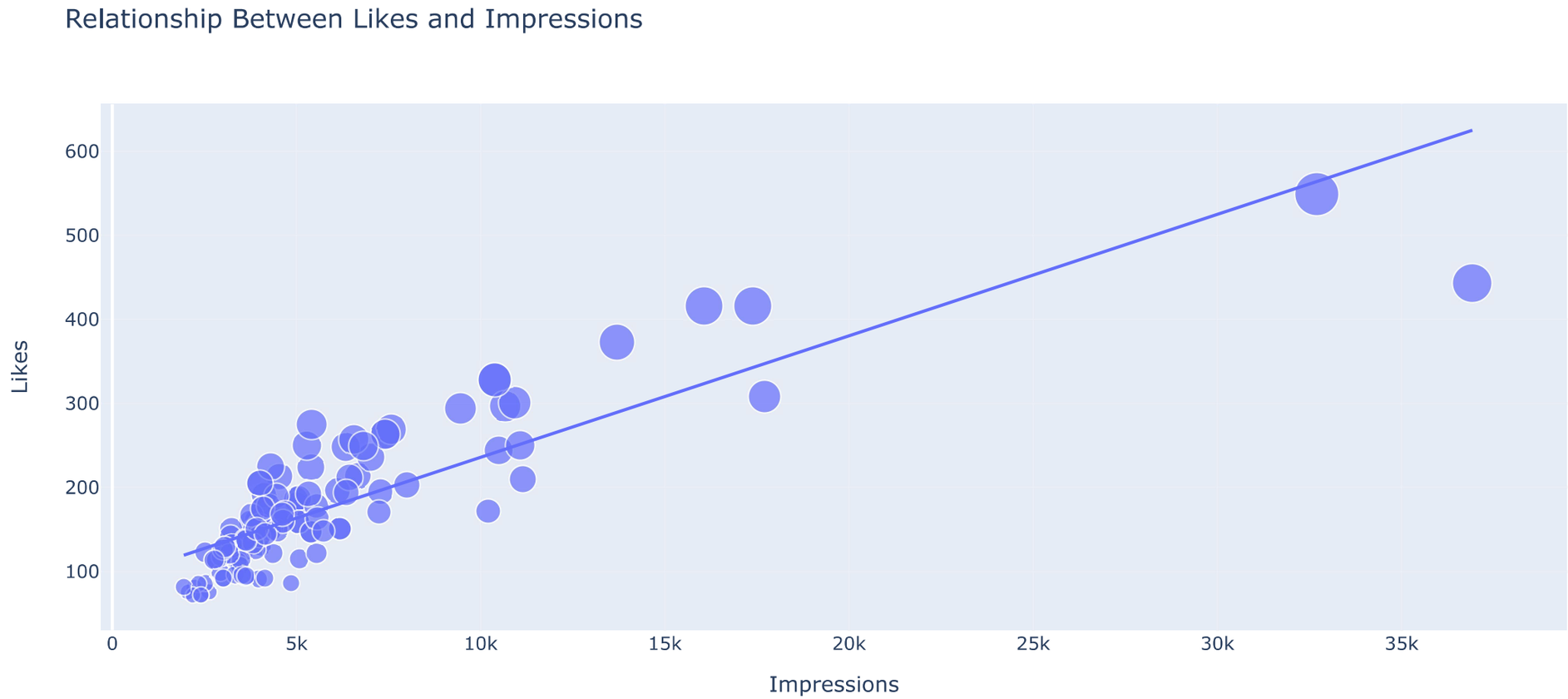


In [15]:
```
text = " ".join(i for i in data.Caption)
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(text)
plt.style.use('classic')
plt.figure( figsize=(12,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

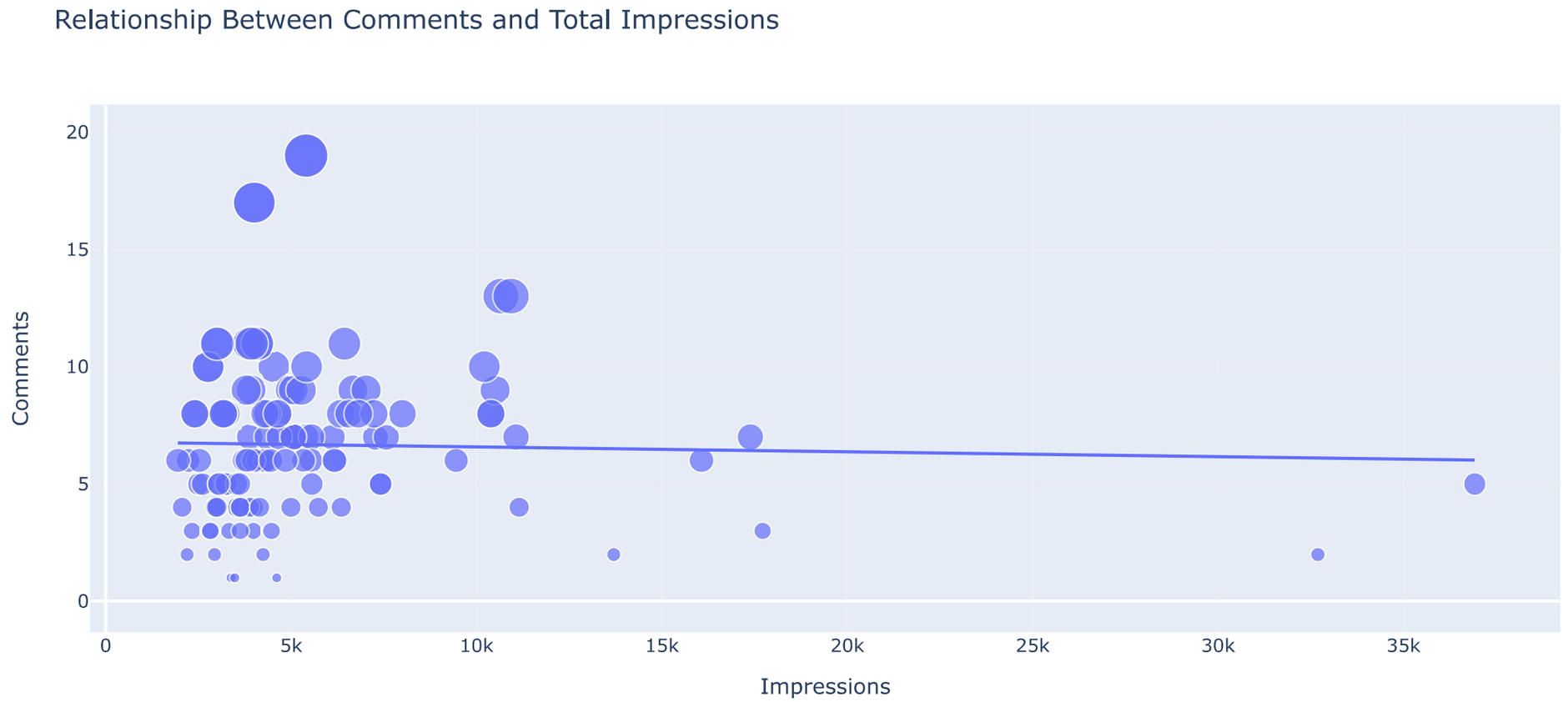
In [16]:
```
text = " ".join(i for i in data.Hashtags)
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(text)
plt.figure( figsize=(12,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



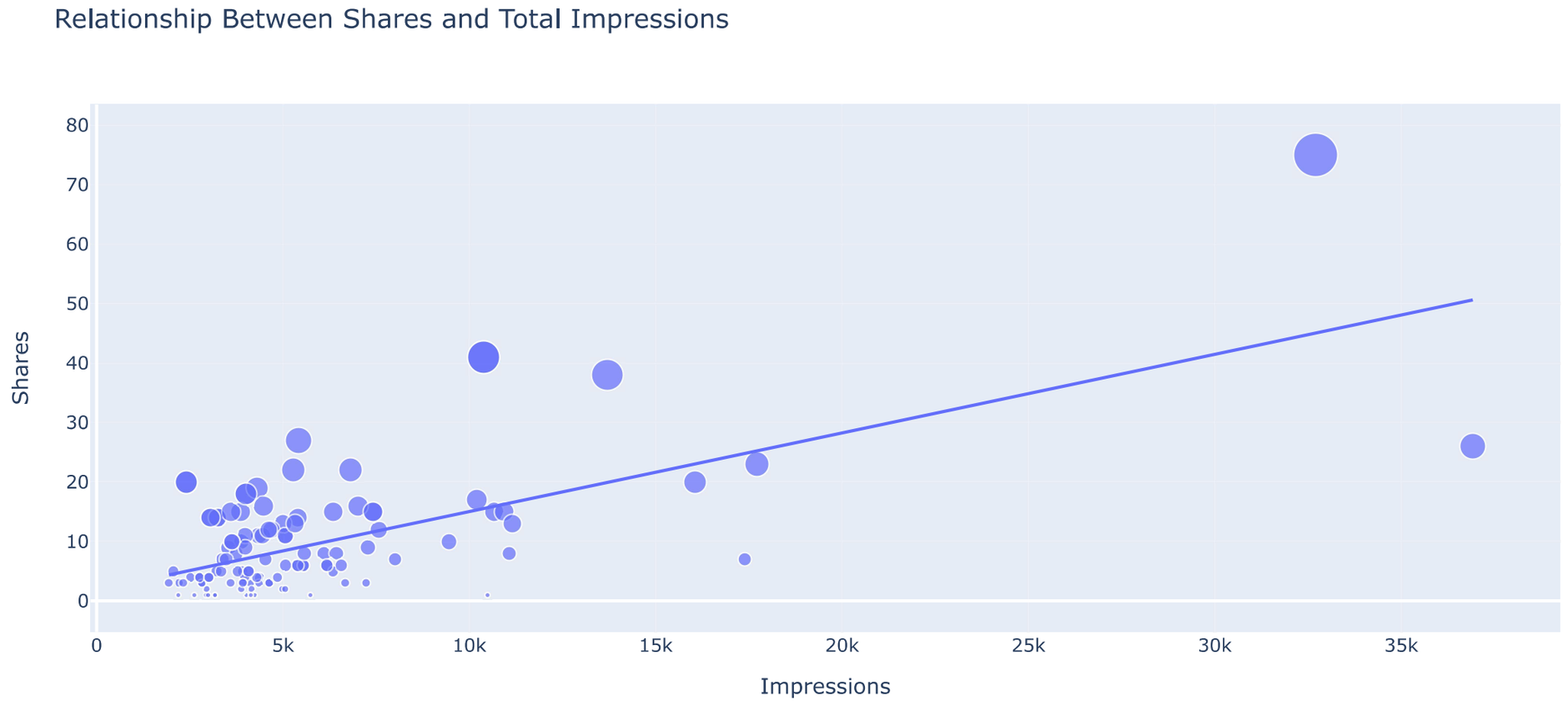
In [17]:
```
figure = px.scatter(data_frame = data, x="Impressions",
                    y="Likes", size="Likes", trendline="ols",
                    title = "Relationship Between Likes and Impressions")
figure.show()
```



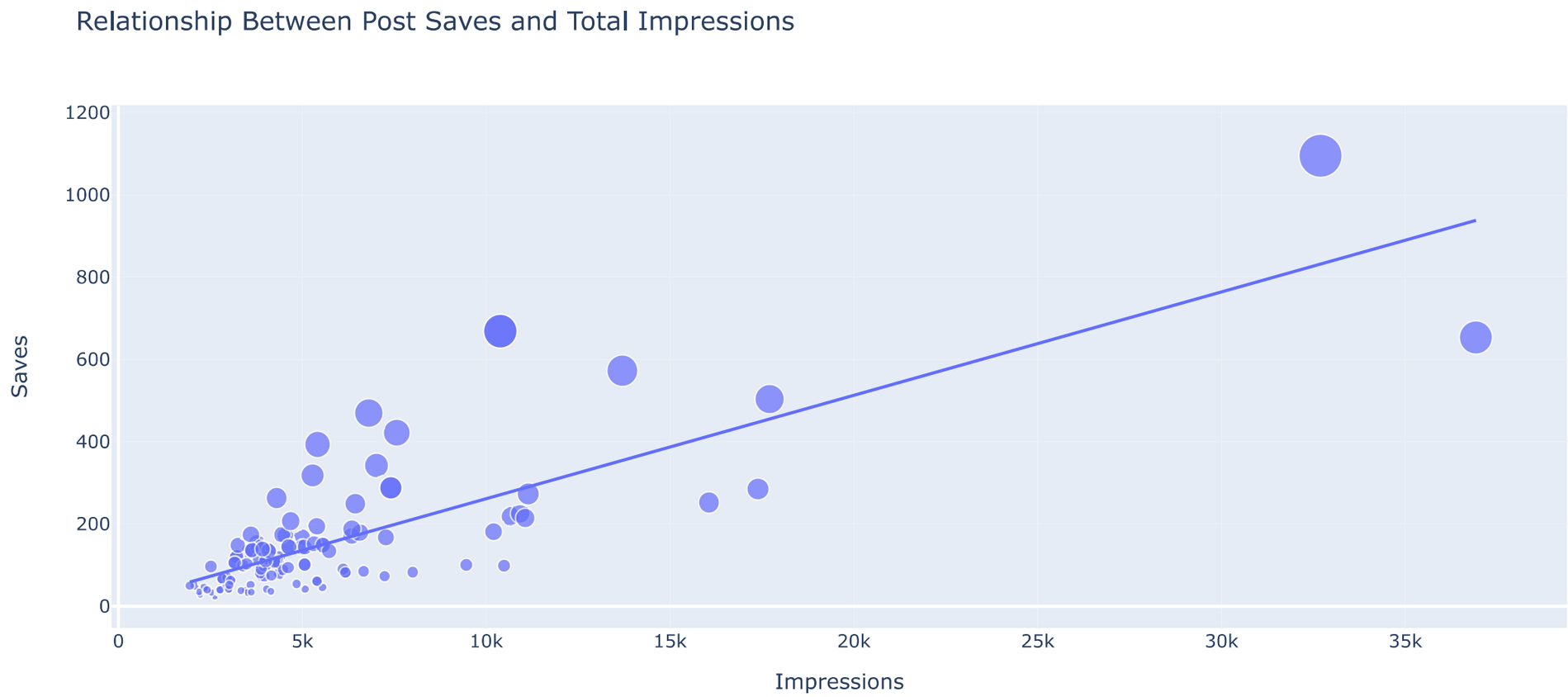
In [18]:
```
figure = px.scatter(data_frame = data, x="Impressions",
                    y="Comments", size="Comments", trendline="ols",
                    title = "Relationship Between Comments and Total Impressions")
figure.show()
```



In [20]:
```
figure = px.scatter(data_frame = data, x="Impressions",
                    y="Shares", size="Shares", trendline="ols",
                    title = "Relationship Between Shares and Total Impressions")
figure.show()
```



In [21]:
```
figure = px.scatter(data_frame = data, x="Impressions",
                    y="Saves", size="Saves", trendline="ols",
                    title = "Relationship Between Post Saves and Total Impressions")
figure.show()
```



In [22]:
```
correlation = data.corr()
print(correlation["Impressions"].sort_values(ascending=False))
```

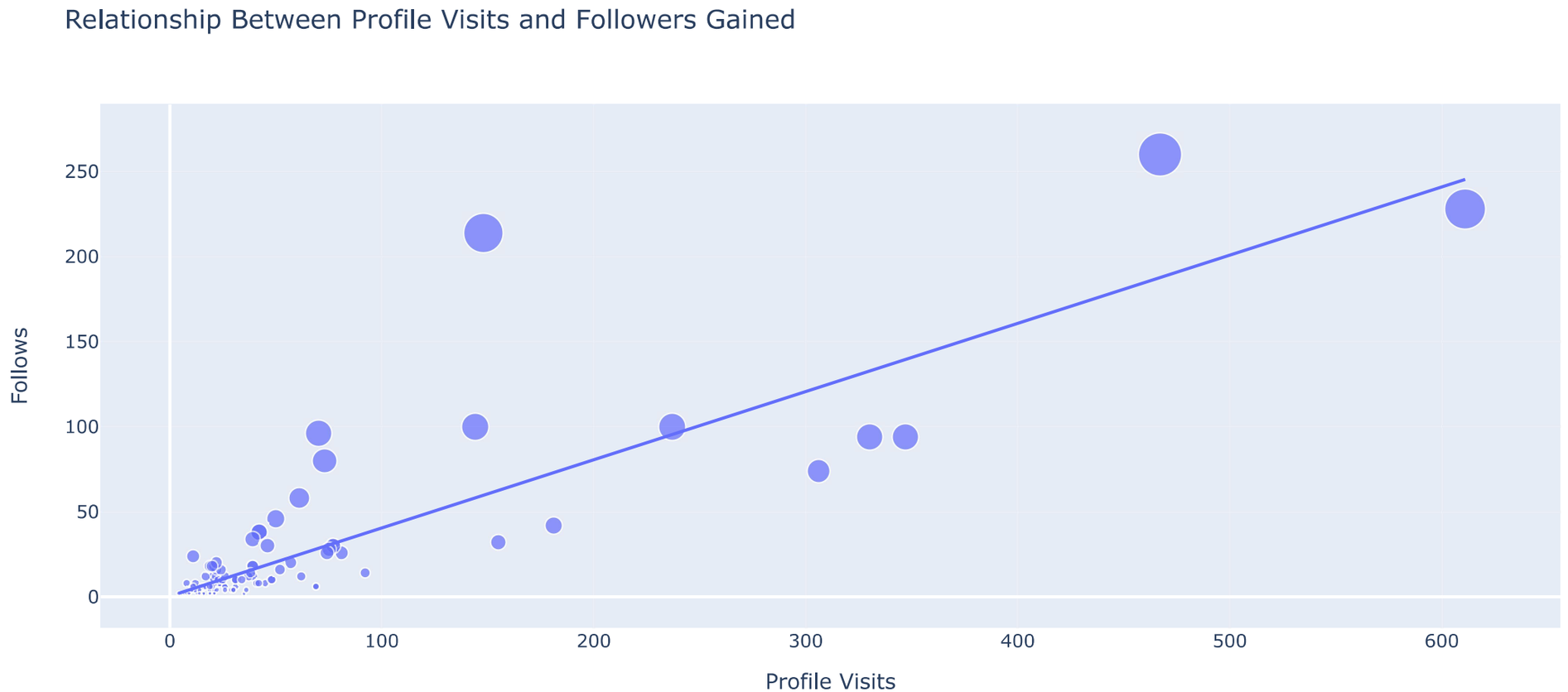
```
Impressions       1.000000
From Explore      0.893607
Follows           0.889363
Likes             0.849835
From Home         0.844698
Saves             0.779231
Profile Visits    0.760981
Shares            0.634675
From Other        0.592960
From Hashtags     0.560760
Comments         -0.028524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

```
C:\Users\sheno\AppData\Local\Temp\ipykernel_9120\3935629544.py:1: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the valu
e of numeric_only to silence this warning.
```

In [23]:
```
conversion_rate = (data["Follows"].sum() / data["Profile Visits"].sum()) * 100
print(conversion_rate)
```

```
41.00265604249668
```

In [24]:
```
figure = px.scatter(data_frame = data, x="Profile Visits",
                    y="Follows", size="Follows", trendline="ols",
                    title = "Relationship Between Profile Visits and Followers Gained")
figure.show()
```



In [25]:
```
x = np.array(data[['Likes', 'Saves', 'Comments', 'Shares',
                   'Profile Visits', 'Follows']])
y = np.array(data["Impressions"])
xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                                test_size=0.2,
                                                random_state=42)
```

In [26]:
```
model = PassiveAggressiveRegressor()
model.fit(xtrain, ytrain)
model.score(xtest, ytest)
```

Out[26]:
```
0.5484995130018204
```

In [27]:
```
# Features = [['Likes','Saves', 'Comments', 'Shares', 'Profile Visits', 'Follows']]
features = np.array([[282.0, 233.0, 4.0, 9.0, 165.0, 54.0]])
model.predict(features)
```

Out[27]:
```
array([8360.06081595])
```

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.corpus import stopwords
import string
import re
nltk.download('stopwords')
stemmer = nltk.SnowballStemmer("english")

data = pd.read_csv("tiktok.csv")
print(data.head())
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\sheno\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
                               reviewId              userName  \
0  68ccaec8-1415-4301-a85e-3004679a3a83          Cassie Moore
1  d84cbfd3-6aa3-485c-aaf9-c5dca27dc966         Kaleb Plummer
2  96618aa1-31e5-4259-8649-89b75d962f00           Rylee Maher
3  078c0bda-598b-474e-a04e-d7cb3e6f6301  Kittykatelyn Romilly
4  8e68c5cd-b12a-4206-a8da-6bfdbff44ae3      Loveness Malenga

                                      userImage  \
0  https://play-lh.googleusercontent.com/a/ALm5wu...
1  https://play-lh.googleusercontent.com/a-/ACNPE...
2  https://play-lh.googleusercontent.com/a/ALm5wu...
3  https://play-lh.googleusercontent.com/a-/ACNPE...
4  https://play-lh.googleusercontent.com/a-/ACNPE...

                                         content  score  thumbsUpCount  \
0                                       No words      5              0
1                             Great fun app so far!      5              0
2  The app would get a higher rating but I litera...      1              0
3  I WISH I COULD GIVE THIS A 100 PERCENT RATING ...      5              0
4                               Pictures and record      5              0

  reviewCreatedVersion                   at replyContent repliedAt
0               27.1.3  2022-11-29 21:55:37          NaN       NaN
1                  NaN  2022-11-29 21:55:04          NaN       NaN
2               27.1.3  2022-11-29 21:54:48          NaN       NaN
3                  NaN  2022-11-29 21:54:35          NaN       NaN
4                  NaN  2022-11-29 21:54:21          NaN       NaN
```

```
C:\Users\sheno\AppData\Local\Temp\ipykernel_31700\2242630289.py:12: DtypeWarning: Columns (8,9) have mixed types. Specify dtype option on import or set low_memory=False.
  data = pd.read_csv("tiktok.csv")
```

In [3]:
```
data = data[["content", "score"]]
print(data.head())
```

```
                                         content  score
0                                       No words      5
1                             Great fun app so far!      5
2  The app would get a higher rating but I litera...      1
3  I WISH I COULD GIVE THIS A 100 PERCENT RATING ...      5
4                               Pictures and record      5
```

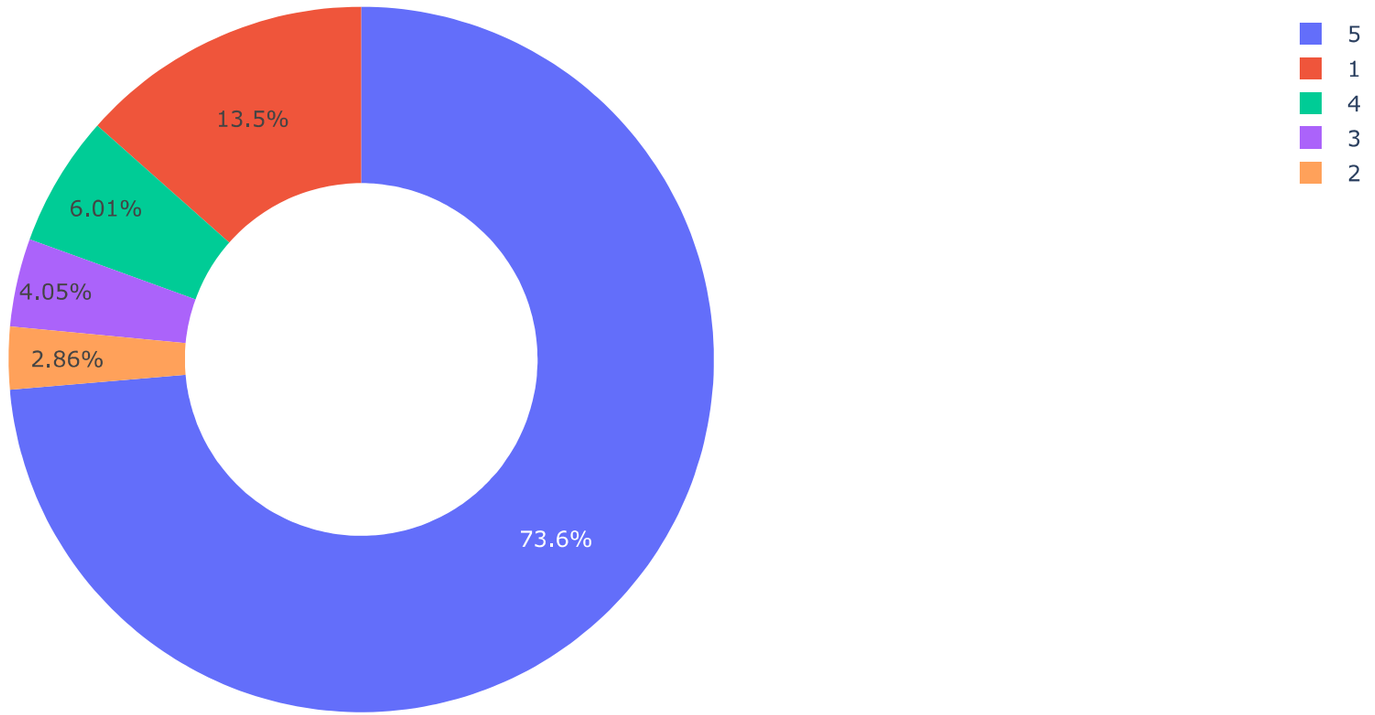
In [4]:
```
print(data.isnull().sum())
```

```
content    16
score       0
dtype: int64
```

In [5]:
```
data = data.dropna()
```

In [6]:
```
stopword=set(stopwords.words('english'))
def clean(text):
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = [word for word in text.split(' ') if word not in stopword]
    text=" ".join(text)
    text = [stemmer.stem(word) for word in text.split(' ')]
    text=" ".join(text)
    return text
data["content"] = data["content"].apply(clean)
```

In [7]:
```
ratings = data["score"].value_counts()
numbers = ratings.index
quantity = ratings.values
import plotly.express as px
figure = px.pie(data,
             values=quantity,
             names=numbers,hole = 0.5)
figure.show()
```



In [8]:
```
text = " ".join(i for i in data.content)
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(text)
plt.figure( figsize=(15,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

In [9]:
```
nltk.download('vader_lexicon')
sentiments = SentimentIntensityAnalyzer()
data["Positive"] = [sentiments.polarity_scores(i)["pos"] for i in data["content"]]
data["Negative"] = [sentiments.polarity_scores(i)["neg"] for i in data["content"]]
data["Neutral"] = [sentiments.polarity_scores(i)["neu"] for i in data["content"]]
data = data[["content", "Positive", "Negative", "Neutral"]]
print(data.head())
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\sheno\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

	content	Positive	Negative	Neutral
0	word	0.000	0.00	1.000
1	great fun app far	0.787	0.00	0.213
2	app would get higher rate liter cant sign seco...	0.000	0.07	0.930
3	wish could give percent rate love 💕 😘	0.633	0.00	0.367
4	pictur record	0.000	0.00	1.000

In [10]:
```
positive =' '.join([i for i in data['content'][data['Positive'] > data["Negative"]]])
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(positive)
plt.figure( figsize=(15,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

In [11]:
```
negative =' '.join([i for i in data['content'][data['Negative'] > data["Positive"]]])
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(negative)
plt.figure( figsize=(15,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```


tiktok
app
video
bad
problem
peopl
tik tok
dont
got ban
im
ban
use
ban reason
content
phone
u
good
updat
thing
annoy
account warn
ban account
stupid
stop
keep
make
tri
show
go
post
see
want
time
work
hate
cant
suck
delet
follow
bad app
still
say
report
one
tiktok team
solv problem
back
take
reason
creator
block
need
fix
realli
idk
never
mani
stuff
think
much
noth
even
live
ive
someth
pleas fix
worst app
give
issu
id
comment
account ban
tiktok account
friend
kid
made
help
anyth
yall
app bad
everyth
know
may
look
lot
put
option
almay
wrong
addict
chang
first
bore
day
guy
instal
actual
poor
allow
cant even
good app
right
user
hate app
edit
communiti guidelin
download
dont know
new updat
wont let
tell
happen
plea
start
post video
use app
sometim
said
sad
ad
match
yet
star
featur
dont want
new account
account got

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import nltk
import re
from nltk.corpus import stopwords
import string
```

In [2]:
```
data = pd.read_csv("filename.csv")
print(data.head())
```

```
                    id      conversation_id               created_at  \
0  1630366235354451969  1630152070530576385  2023-02-28 00:36:15 UTC
1  1630366226424778753  1630366226424778753  2023-02-28 00:36:13 UTC
2  1630366225930027011  1630366225930027011  2023-02-28 00:36:13 UTC
3  1630366223056662530  1630351686974992385  2023-02-28 00:36:12 UTC
4  1630366221483884545  1629903982255644672  2023-02-28 00:36:12 UTC

         date      time  timezone             user_id     username  \
0  2023-02-28  00:36:15         0  1493761817406894086  tomasliptai
1  2023-02-28  00:36:13         0  1526694166662721536  paperfloure
2  2023-02-28  00:36:13         0  1053018392939167746    katetbar1
3  2023-02-28  00:36:12         0            602371247    jlhrdhmom
4  2023-02-28  00:36:12         0  1053594763214184448    phemikali

                 name place  ... geo source user_rt_id user_rt retweet_id  \
0        Tomas Liptai   NaN  ... NaN    NaN        NaN     NaN        NaN
1     Smell the roses   NaN  ... NaN    NaN        NaN     NaN        NaN
2               @etak   NaN  ... NaN    NaN        NaN     NaN        NaN
3             JLHrdh   NaN  ... NaN    NaN        NaN     NaN        NaN
4  rolarkcybersecurity   NaN  ... NaN    NaN        NaN     NaN        NaN

                                        reply_to  retweet_date  translate  \
0  [{'screen_name': 'nazijaeger__', 'name': 'nazi...           NaN        NaN
1                                                []           NaN        NaN
2                                                []           NaN        NaN
3  [{'screen_name': 'MainelifeR', 'name': 'Mainel...           NaN        NaN
4  [{'screen_name': 'Pottingpinks', 'name': 'GRS'...           NaN        NaN

  trans_src trans_dest
0       NaN        NaN
1       NaN        NaN
2       NaN        NaN
3       NaN        NaN
4       NaN        NaN

[5 rows x 36 columns]
```

In [3]:
```
print(data.columns)
```

```
Index(['id', 'conversation_id', 'created_at', 'date', 'time', 'timezone',
       'user_id', 'username', 'name', 'place', 'tweet', 'language', 'mentions',
       'urls', 'photos', 'replies_count', 'retweets_count', 'likes_count',
       'hashtags', 'cashtags', 'link', 'retweet', 'quote_url', 'video',
       'thumbnail', 'near', 'geo', 'source', 'user_rt_id', 'user_rt',
       'retweet_id', 'reply_to', 'retweet_date', 'translate', 'trans_src',
       'trans_dest'],
      dtype='object')
```

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

Out[4]:

```
id                    0
conversation_id       0
created_at            0
date                  0
time                  0
timezone              0
user_id               0
username              0
name                  0
place             10011
tweet                 0
language              0
mentions              0
urls                  0
photos                0
replies_count         0
retweets_count        0
likes_count           0
hashtags              0
cashtags              0
link                  0
retweet               0
quote_url          9235
video                 0
thumbnail          8927
near              10014
geo               10014
source            10014
user_rt_id        10014
user_rt           10014
retweet_id        10014
reply_to              0
retweet_date      10014
translate         10014
trans_src         10014
trans_dest        10014
dtype: int64
```

In [5]:

```
data["language"].value_counts()
```

Out[5]:
```
en     8858
pt      440
it      194
qme     105
und      60
in       47
ru       44
ja       42
es       36
ca       20
qht      20
th       19
fr       18
de       14
ko        9
vi        8
nl        8
ro        7
fi        7
ar        6
zxx       6
uk        6
cs        6
zh        5
pl        5
qam       4
tl        4
da        3
eu        2
no        2
hi        2
tr        2
hu        1
cy        1
lv        1
el        1
bn        1
Name: language, dtype: int64
```

In [6]:
```
nltk.download('stopwords')
stemmer = nltk.SnowballStemmer("english")
stopword=set(stopwords.words('english'))

def clean(text):
    text = str(text).lower()
```

```
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = [word for word in text.split(' ') if word not in stopword]
    text=" ".join(text)
    text = [stemmer.stem(word) for word in text.split(' ')]
    text=" ".join(text)
    return text
data["tweet"] = data["tweet"].apply(clean)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\sheno\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

In [7]:

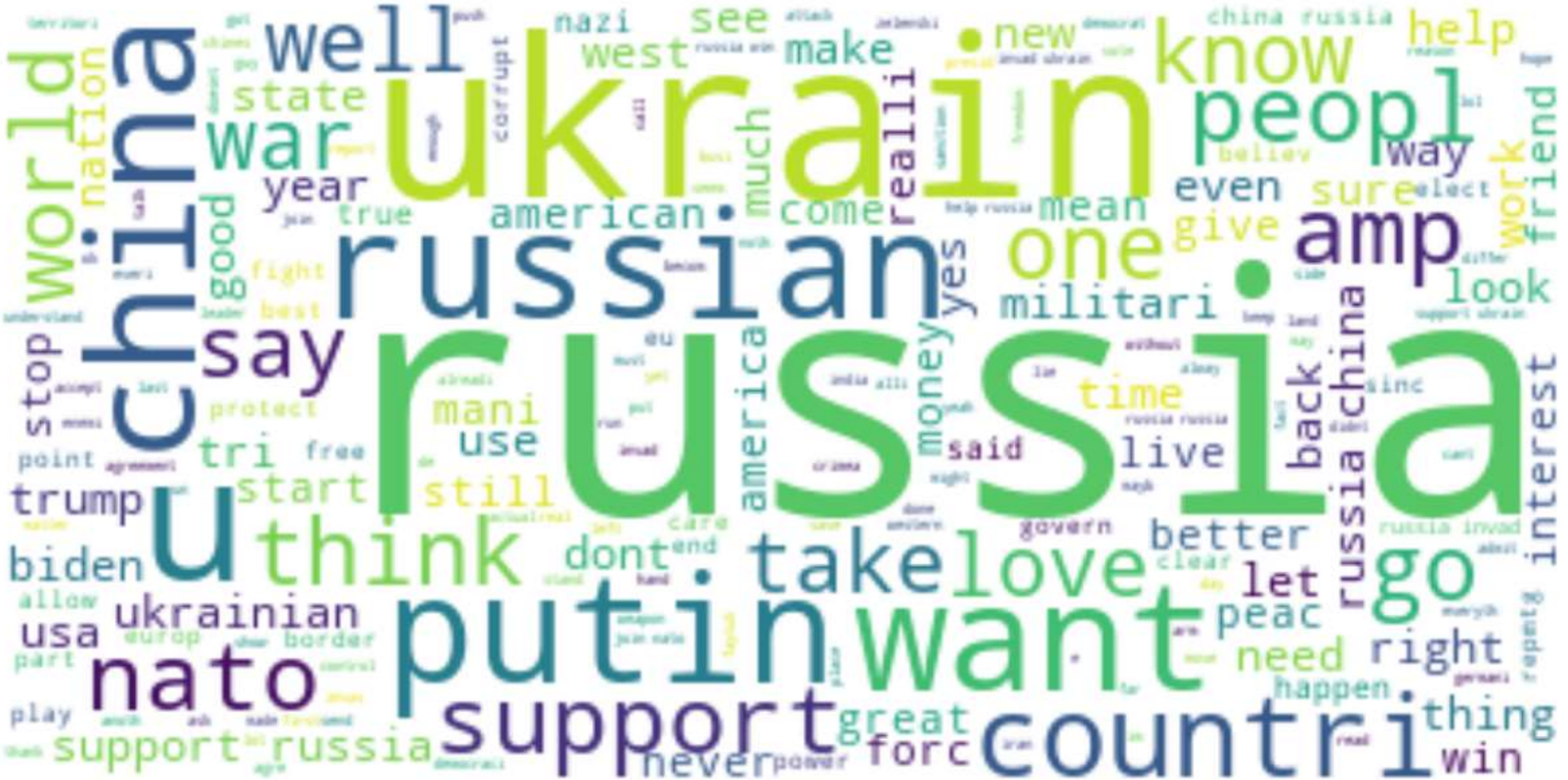
```
text = " ".join(i for i in data.tweet)
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(text)
plt.figure( figsize=(15,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

In [8]:
```
nltk.download('vader_lexicon')
sentiments = SentimentIntensityAnalyzer()
data["Positive"] = [sentiments.polarity_scores(i)["pos"] for i in data["tweet"]]
data["Negative"] = [sentiments.polarity_scores(i)["neg"] for i in data["tweet"]]
data["Neutral"] = [sentiments.polarity_scores(i)["neu"] for i in data["tweet"]]
data = data[["tweet", "Positive", "Negative", "Neutral"]]
print(data.head())
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\sheno\AppData\Roaming\nltk_data...
```

	tweet	Positive	Negative	Neutral
0	nazijaeg derwen russia place satan rule well	0.259	0.000	0.741
1	russia haarp could destroy usa one fell swoop ...	0.000	0.280	0.720
2	putin give steven seagal order friendship	0.367	0.000	0.633
3	mainelif baddcompani it alway project russia	0.000	0.000	1.000
4	pottingpink mfarussia modrussia milhistrf muze...	0.068	0.078	0.854

In [9]:
```
positive =' '.join([i for i in data['tweet'][data['Positive'] > data["Negative"]]])
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(positive)
plt.figure( figsize=(15,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



In [10]:
```
negative =' '.join([i for i in data['tweet'][data['Negative'] > data["Positive"]]])
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(negative)
plt.figure( figsize=(15,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```


russia
ukrain
u
china
putin
nato
war
countri
peopl
russian
amp
say
stop
one
think
want
need
take
lie
right
go
ukrainian
time
make
biden
know
world
war russia
attack
fight
trump
tri
use
russia china
ukrain war
war ukrain
west
back
new
state
sinc
support
bad
let
still
dont
live
look
side
usa
never
help
said
destroy
forc
militari
see
arm
even
happen
american
america
start
peac
kill
mani
year
border
call
way

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv("uber.csv")
data["Date/Time"] = data["Date/Time"].map(pd.to_datetime)
data.head()
```

Out[1]:

	Date/Time	Lat	Lon	Base
0	2014-09-01 00:01:00	40.2201	-74.0021	B02512
1	2014-09-01 00:01:00	40.7500	-74.0027	B02512
2	2014-09-01 00:03:00	40.7559	-73.9864	B02512
3	2014-09-01 00:06:00	40.7450	-73.9889	B02512
4	2014-09-01 00:11:00	40.8145	-73.9444	B02512

In [2]:
```
data["Day"] = data["Date/Time"].apply(lambda x: x.day)
data["Weekday"] = data["Date/Time"].apply(lambda x: x.weekday())
data["Hour"] = data["Date/Time"].apply(lambda x: x.hour)
print(data.head())
```

```
            Date/Time      Lat      Lon    Base  Day  Weekday  Hour
0 2014-09-01 00:01:00  40.2201 -74.0021  B02512    1        0     0
1 2014-09-01 00:01:00  40.7500 -74.0027  B02512    1        0     0
2 2014-09-01 00:03:00  40.7559 -73.9864  B02512    1        0     0
3 2014-09-01 00:06:00  40.7450 -73.9889  B02512    1        0     0
4 2014-09-01 00:11:00  40.8145 -73.9444  B02512    1        0     0
```

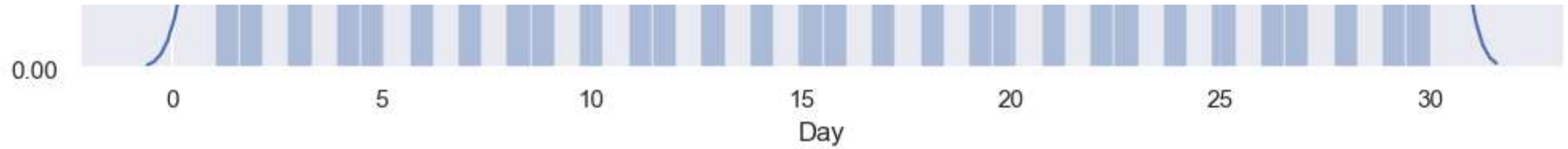
In [3]:
```
sns.set(rc={'figure.figsize':(12, 10)})
sns.distplot(data["Day"])
```

```
C:\Users\sheno\AppData\Local\Temp\ipykernel_32904\1438482754.py:2: 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(data["Day"])
```

Out[3]:

```
<Axes: xlabel='Day', ylabel='Density'>
```

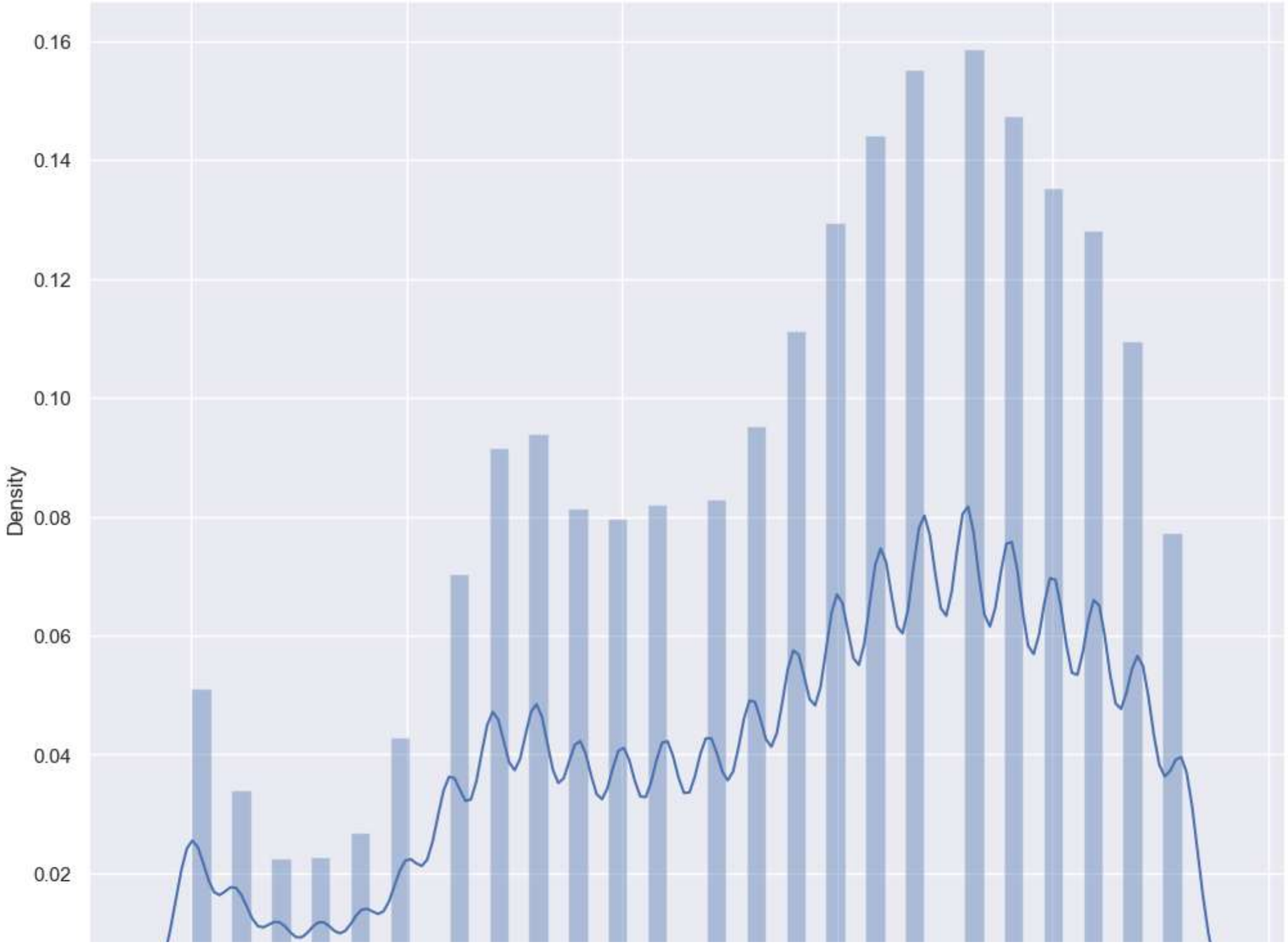

0.07
0.06
0.05
0.04
0.03
0.02
0.01
Density

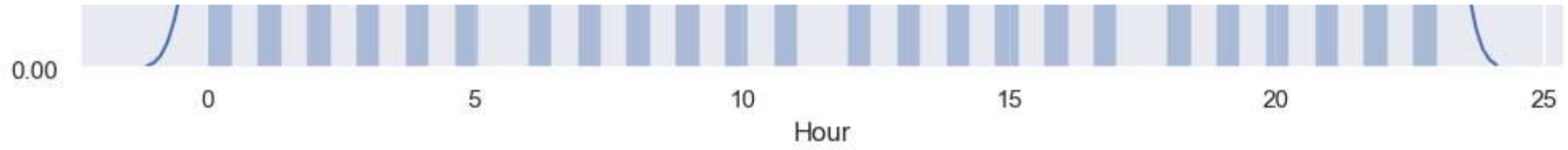


In [4]:
```
sns.distplot(data["Hour"])
```

```
C:\Users\sheno\AppData\Local\Temp\ipykernel_32904\96630979.py:1: 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(data["Hour"])
```

Out[4]:
```
<Axes: xlabel='Hour', ylabel='Density'>
```

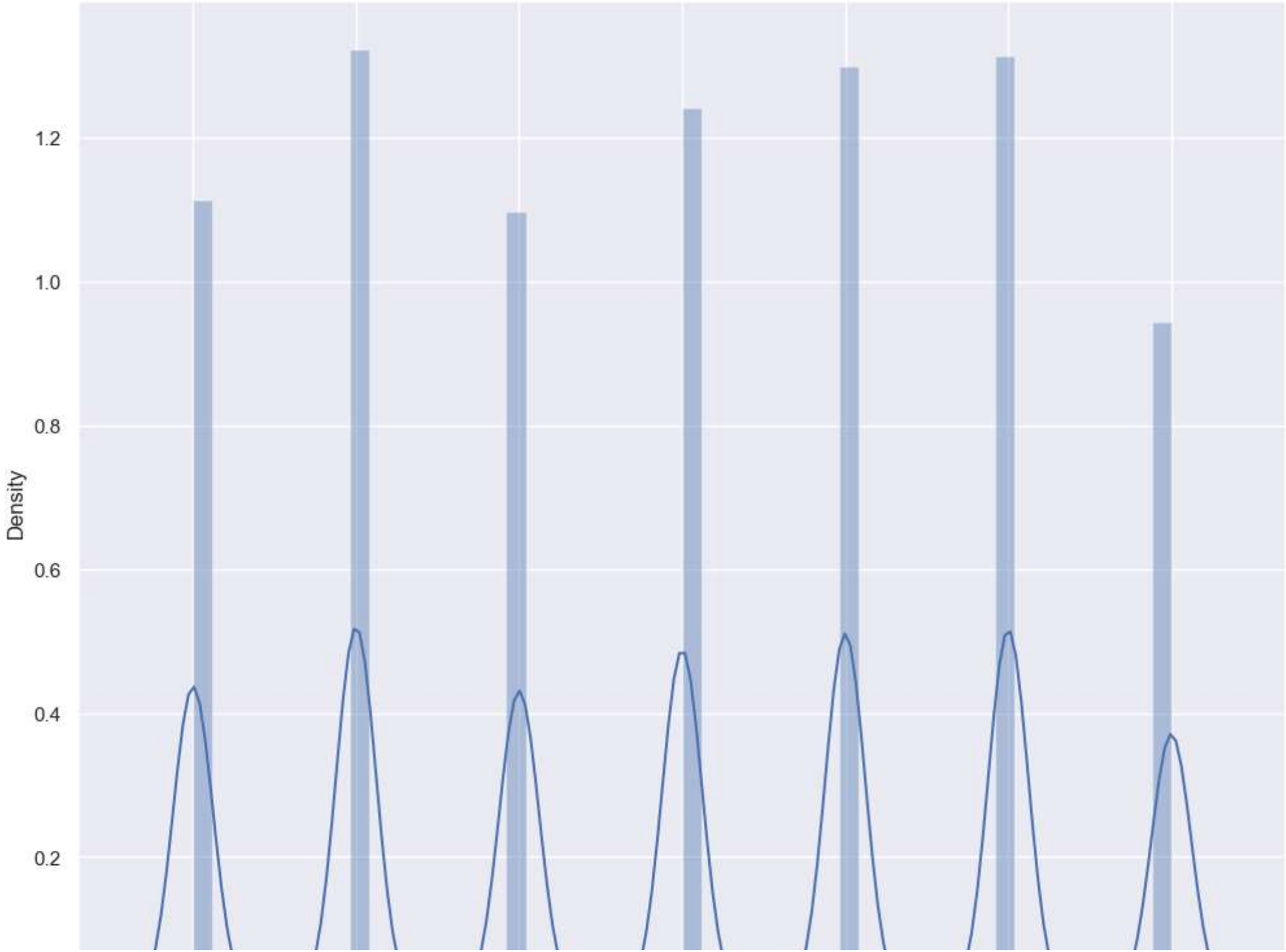

0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
Density



In [5]:
```
sns.distplot(data["Weekday"])
```

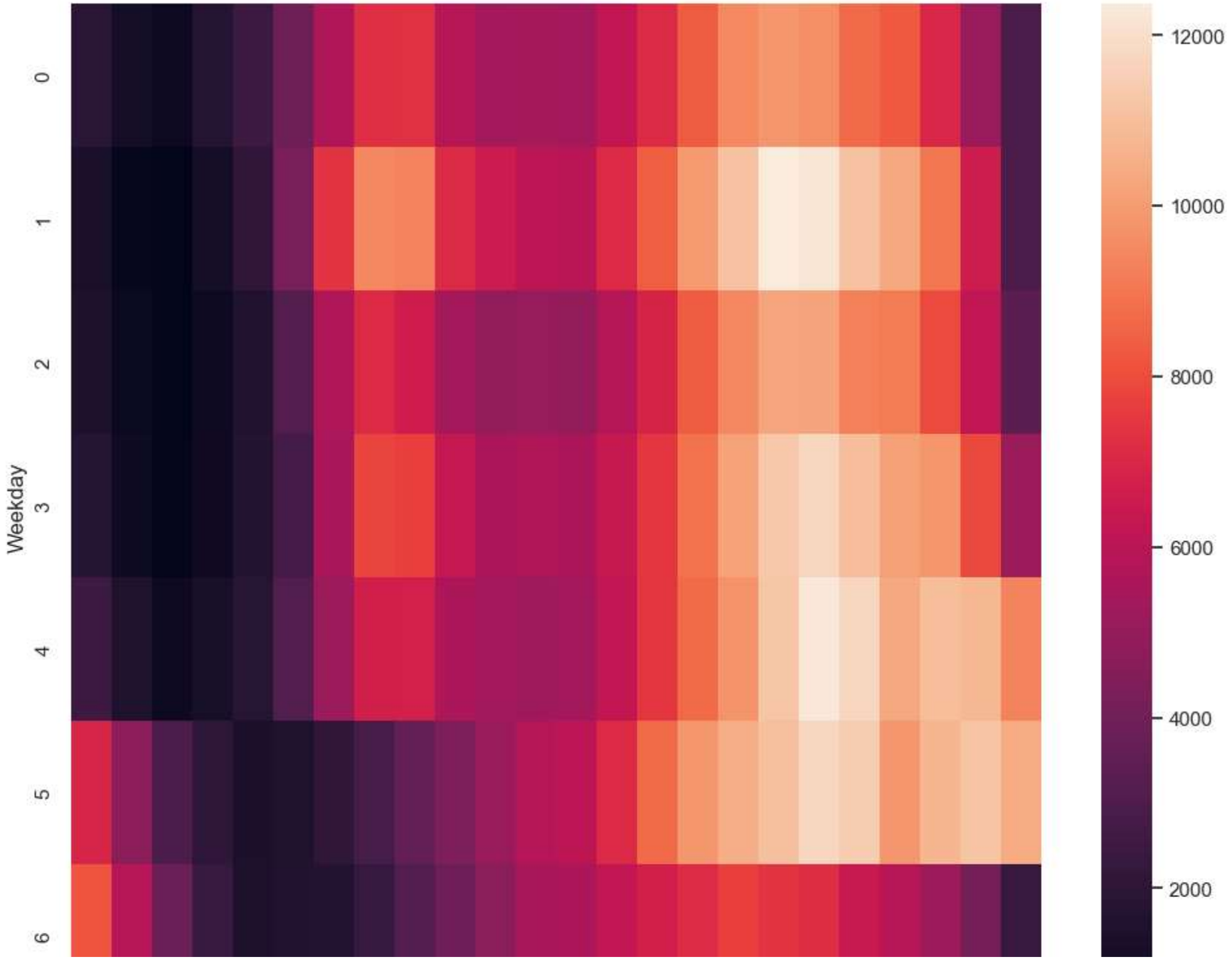
```
C:\Users\sheno\AppData\Local\Temp\ipykernel_32904\2809950383.py:1: UserWarning:

`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(data["Weekday"])
```

Out[5]:
```
<Axes: xlabel='Weekday', ylabel='Density'>
```


1.2
1.0
0.8
Density
0.6
0.4
0.2

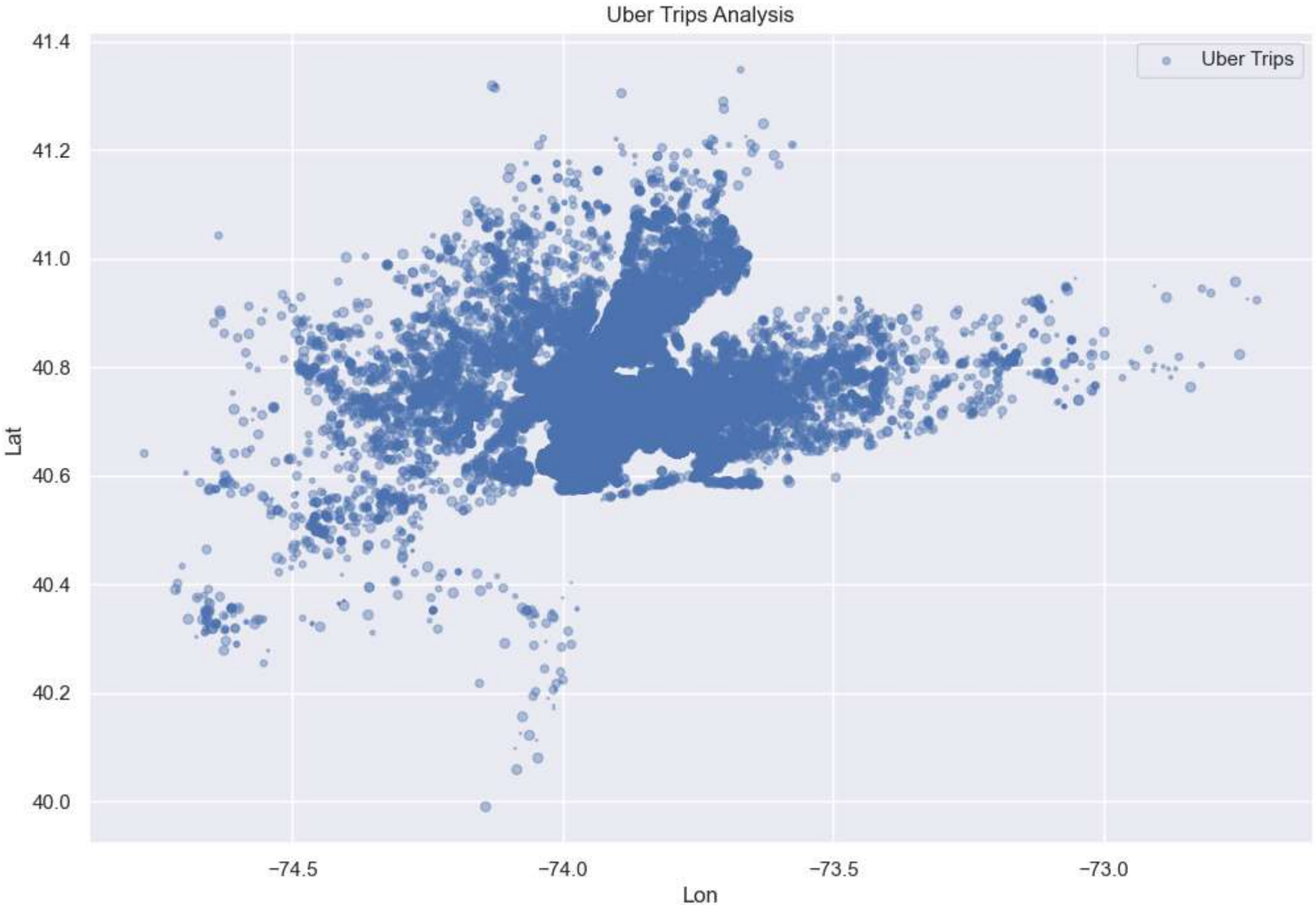
In [6]:
```
# Correlation of Weekday and Hour
df = data.groupby(["Weekday", "Hour"]).apply(lambda x: len(x))
df = df.unstack()
sns.heatmap(df, annot=False)
```

Out[6]:
```
<Axes: xlabel='Hour', ylabel='Weekday'>
```


Weekday
0
1
2
3
4
5
6
12000
10000
8000
6000
4000
2000

In [7]:
```
data.plot(kind='scatter', x='Lon', y='Lat', alpha=0.4, s=data['Day'], label='Uber Trips',
figsize=(12, 8), cmap=plt.get_cmap('jet'))
plt.title("Uber Trips Analysis")
plt.legend()
plt.show()
```

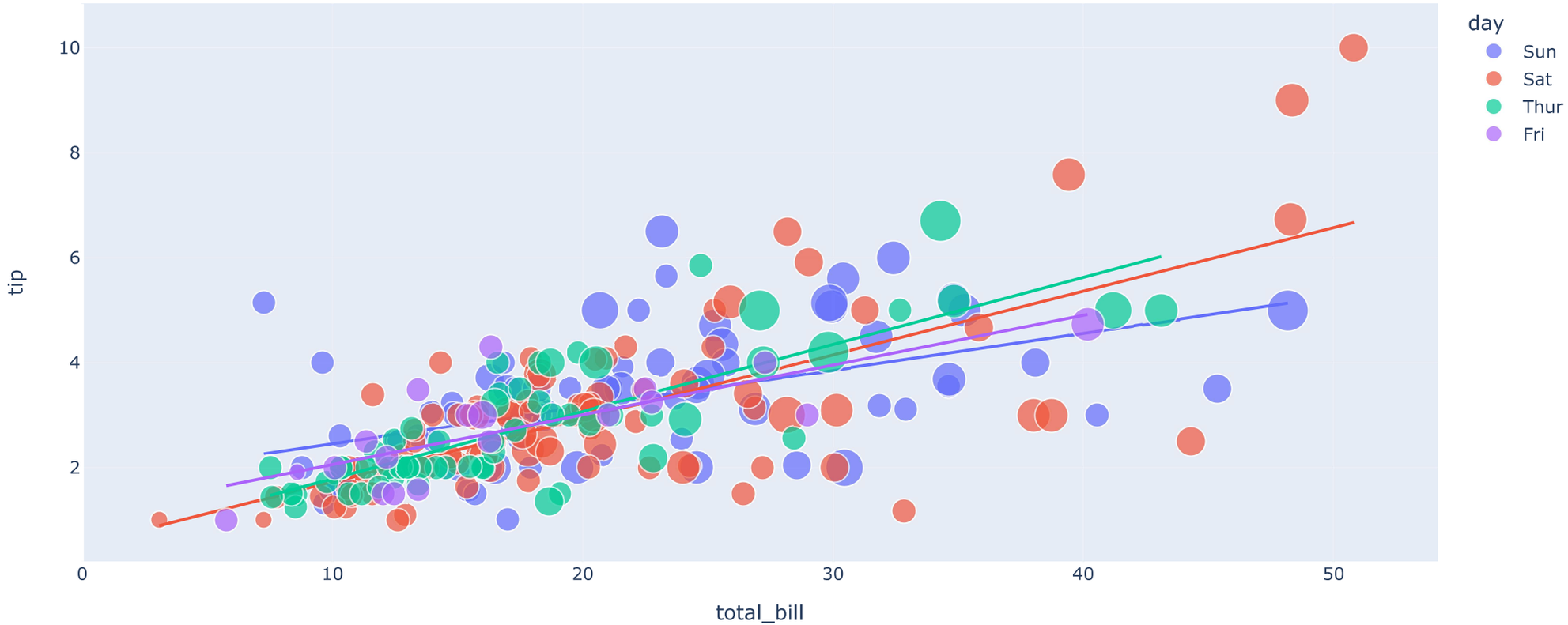
```
C:\Users\sheno\anaconda3\Lib\site-packages\pandas\plotting\_matplotlib\core.py:1259: UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap'
will be ignored
  scatter = ax.scatter(
```


Uber Trips Analysis
Uber Trips
41.4
41.2
41.0
40.8
40.6
40.4
40.2
40.0
Lat
−74.5
−74.0
−73.5
−73.0
Lon

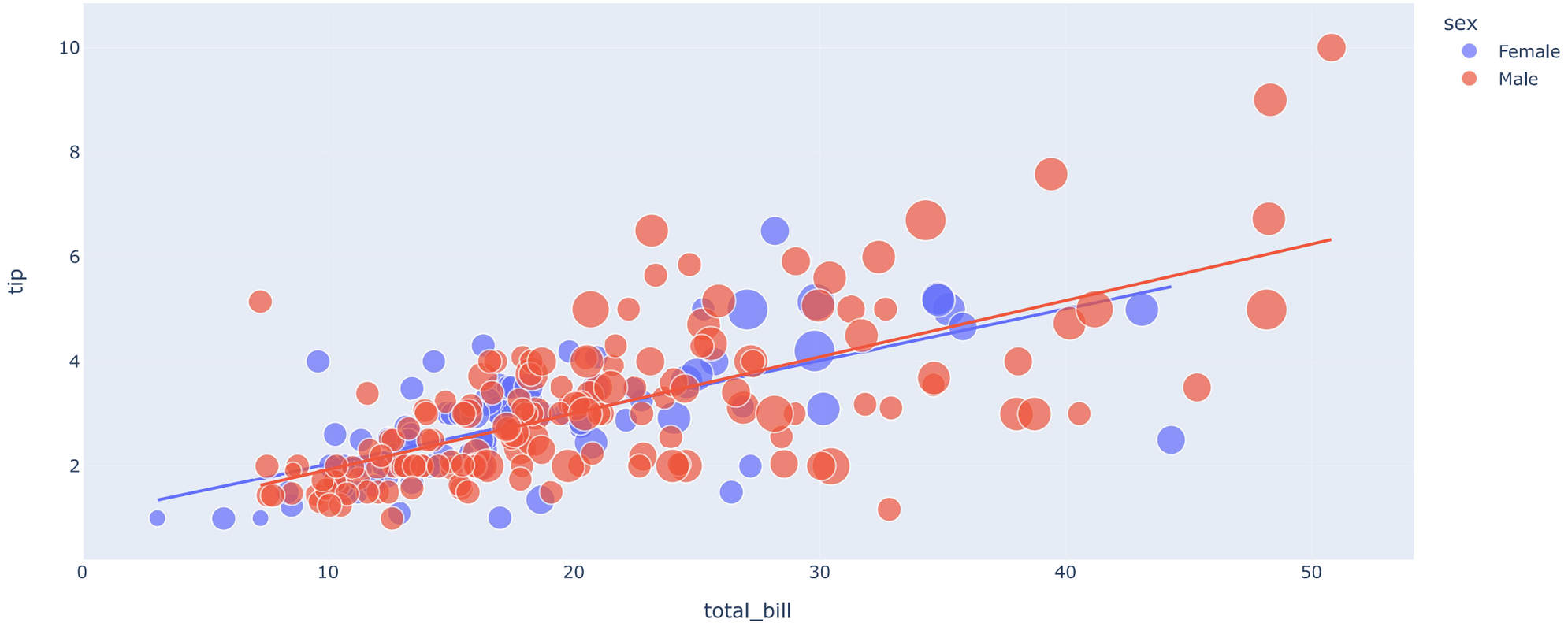
In [1]:
```
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go

data = pd.read_csv("tips.csv")
print(data.head())
```

```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female     No  Sun  Dinner     4
```

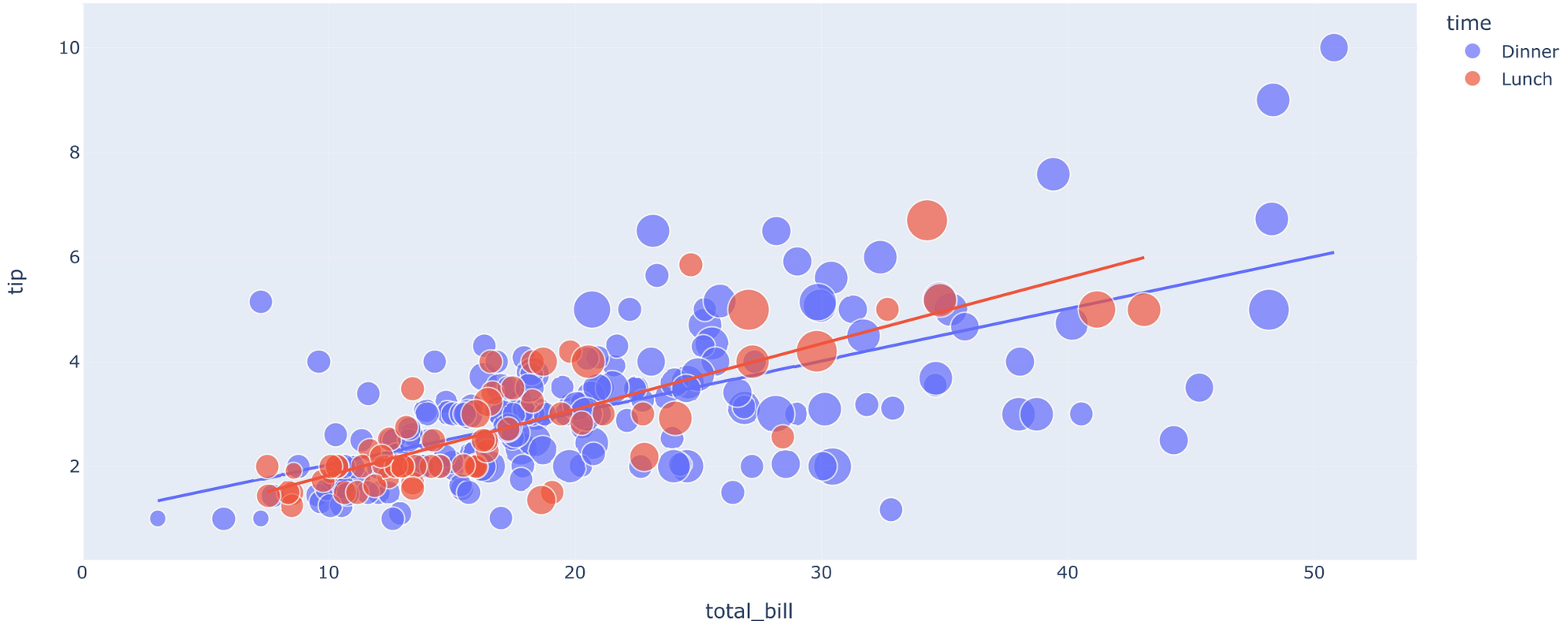
In [2]:
```
figure = px.scatter(data_frame = data, x="total_bill",
                    y="tip", size="size", color= "day", trendline="ols")
figure.show()
```



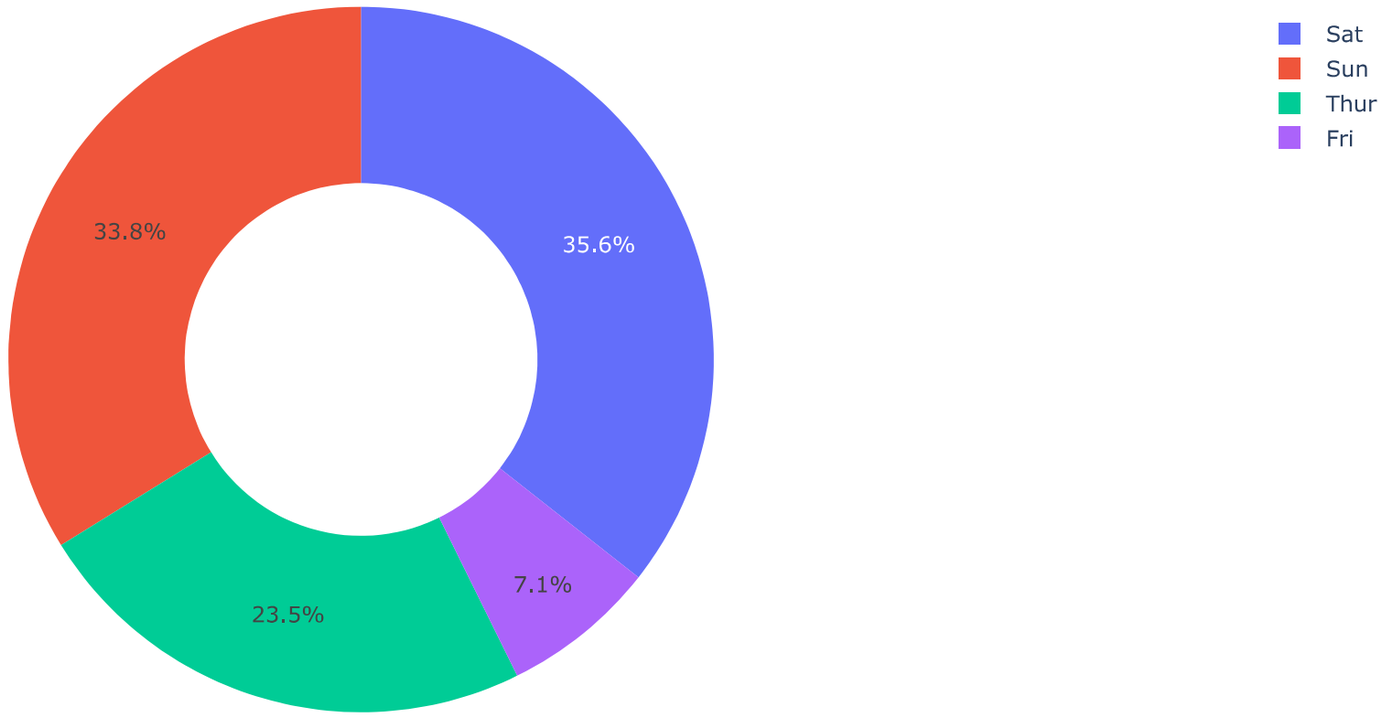
In [3]:
```
figure = px.scatter(data_frame = data, x="total_bill",
                    y="tip", size="size", color= "sex", trendline="ols")
figure.show()
```



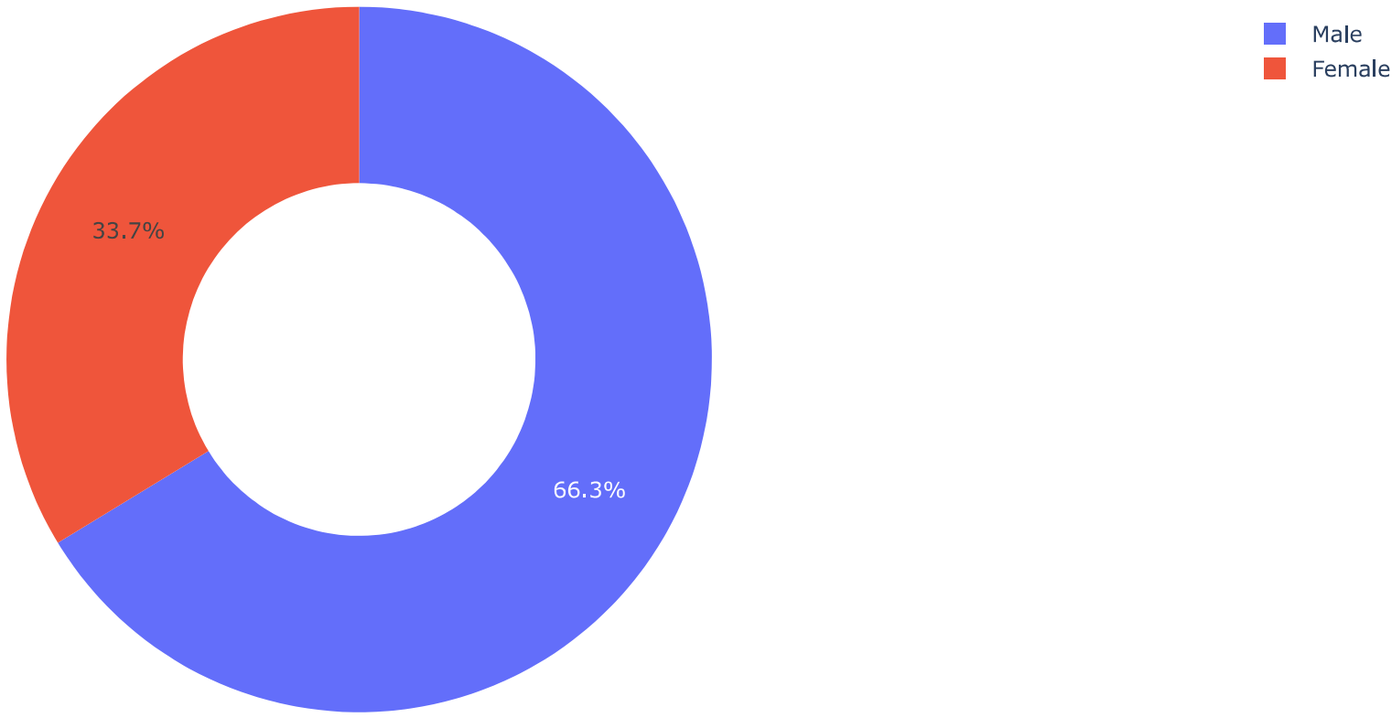
In [4]:
```
figure = px.scatter(data_frame = data, x="total_bill",
                    y="tip", size="size", color= "time", trendline="ols")
figure.show()
```



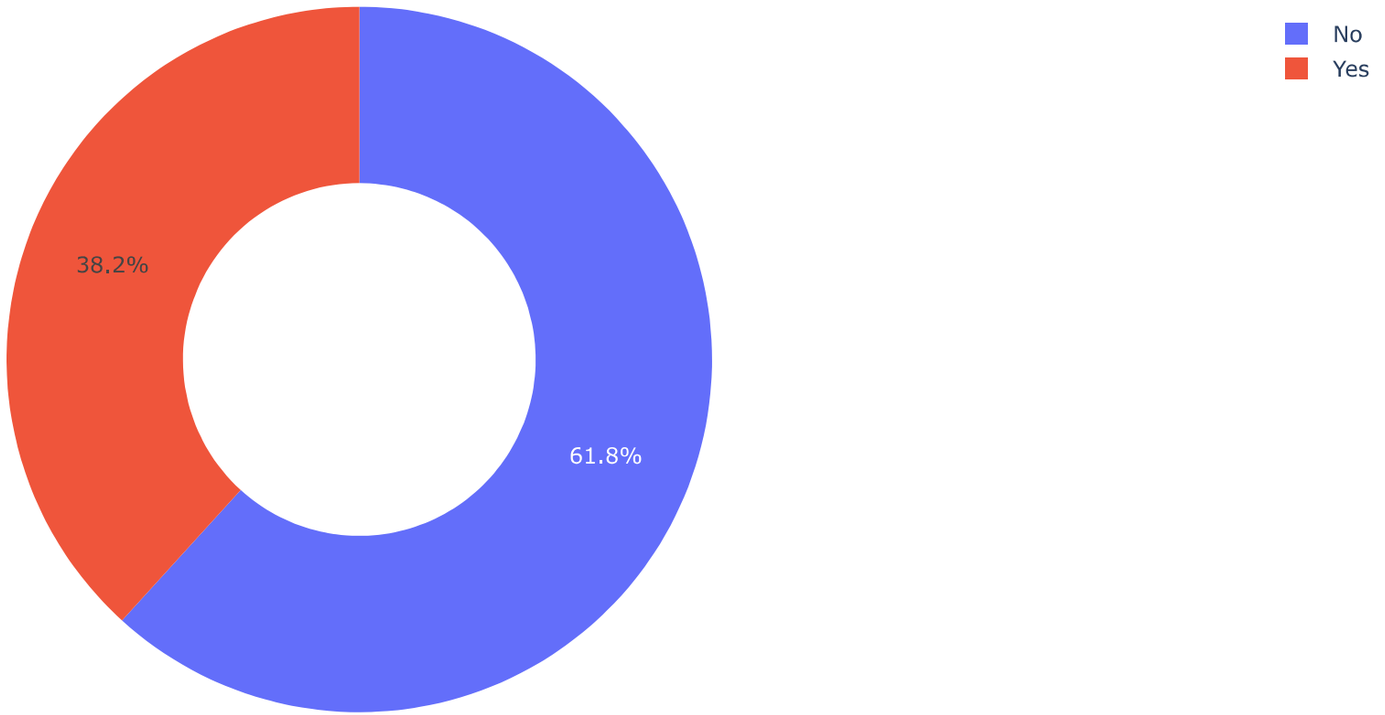
In [5]:
```
figure = px.pie(data,
                values='tip',
                names='day',hole = 0.5)
figure.show()
```



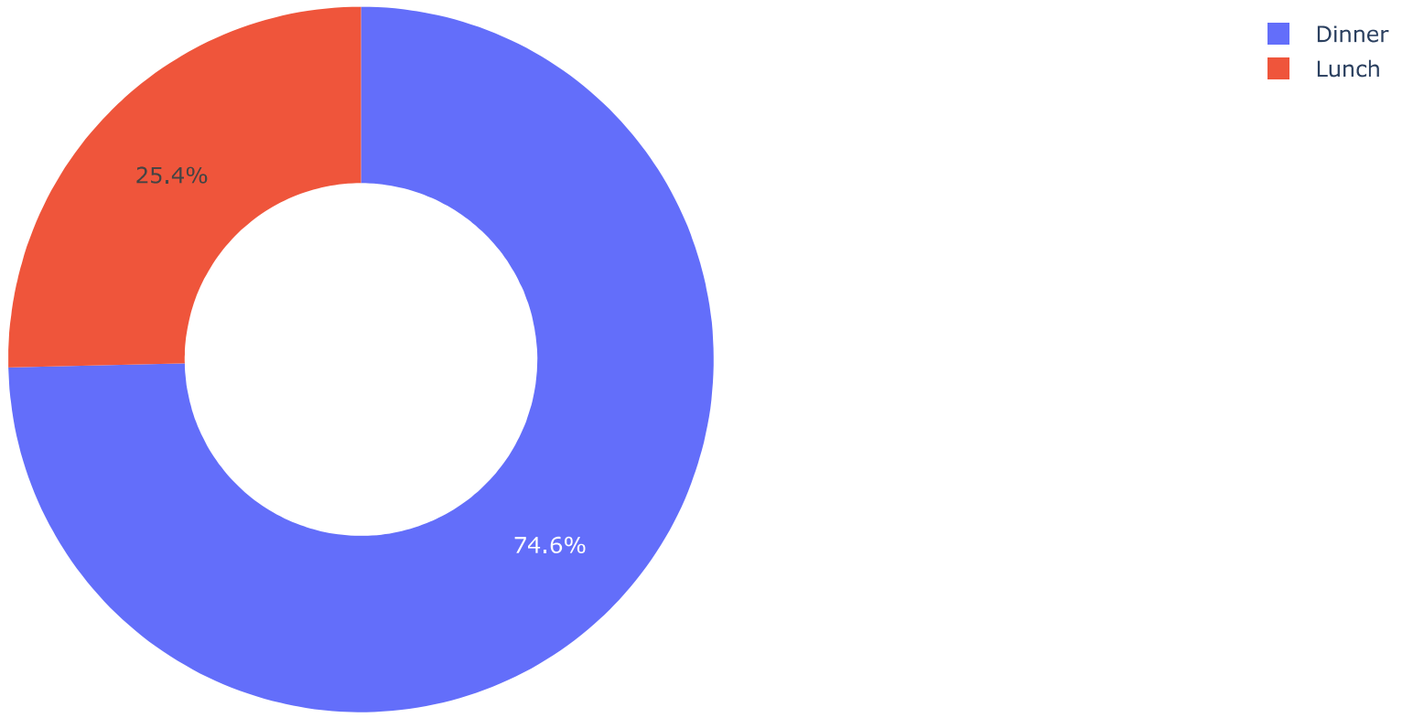
In [6]:
```
figure = px.pie(data,
                values='tip',
                names='sex',hole = 0.5)
figure.show()
```



In [7]:
```
figure = px.pie(data,
             values='tip',
             names='smoker',hole = 0.5)
figure.show()
```



In [8]:
```
figure = px.pie(data,
             values='tip',
             names='time',hole = 0.5)
figure.show()
```



In [9]:
```
data["sex"] = data["sex"].map({"Female": 0, "Male": 1})
data["smoker"] = data["smoker"].map({"No": 0, "Yes": 1})
data["day"] = data["day"].map({"Thur": 0, "Fri": 1, "Sat": 2, "Sun": 3})
data["time"] = data["time"].map({"Lunch": 0, "Dinner": 1})
data.head()
```

Out[9]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	0	0	3	1	2
1	10.34	1.66	1	0	3	1	3
2	21.01	3.50	1	0	3	1	3
3	23.68	3.31	1	0	3	1	2
4	24.59	3.61	0	0	3	1	4

In [10]:
```
x = np.array(data[["total_bill", "sex", "smoker", "day",
                   "time", "size"]])
y = np.array(data["tip"])

from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                                test_size=0.2,
                                                random_state=42)
```

In [11]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(xtrain, ytrain)
```

Out[11]:

▾ LinearRegression

LinearRegression()

In [12]:
```
# features = [[total_bill, "sex", "smoker", "day", "time", "size"]]
features = np.array([[24.50, 1, 0, 0, 1, 4]])
model.predict(features)
```

Out[12]:
```
array([3.73742609])
```