# Cricket World Cup 2023 Analysis

October 31, 2023

- Cricket World Cup 2023 Analysis -

[134]:
```python
from IPython.display import Image
Image(filename="ICC-Cricket-World-Cup-2023.jpg" )
```

[134]:

[135]:
```python
# Project By: Uvesh Ahmad
# Data Set Link: https://github.com/Uvesh-Ahmad
# Portfolio: https://uvesh-ahmad.github.io/uvesh.ah/
# Linkedin : https://www.linkedin.com/in/uvesh-ahmad-a2aa6816a/
```

[136]:
```python
# Import the NumPy library and alias it as np

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
```

[137]:

```
# Import the warnings module to manage warnings in Python code
# Disable all warnings to suppress warning messages during execution

import warnings
warnings.filterwarnings('ignore')
```

[138]:

```
# The 'data' list contains information about cricket matches, each represented␣
 ↪as a list of details.

data = [
["Thursday - October 5 2023", "ENGLAND vs NEW ZEALAND", "New Zealand Won", "2:
 ↪00 PM", "Ahmedabad"],
["Friday - October 6 2023", "PAKISTAN vs NETHERLANDS", "Pakistan Won", "2:00␣
 ↪PM", "Hyderabad"],
["Saturday - October 7 2023", "AFGHANISTAN vs BANGLADESH", "Bangladesh Won",␣
 ↪"10:30 AM", "Dharamsala"],
["Saturday - October 7 2023", "SOUTH AFRICA vs SRI LANKA", "South Africa Won",␣
 ↪"2:00 PM", "Delhi"],
["Sunday - October 8 2023", "AUSTRALIA vs INDIA", "India Won", "2:00 PM",␣
 ↪"Chennai"],
["Monday - October 9 2023", "NEW ZEALAND vs NETHERLANDS", "New Zealand Won", "2:
 ↪00 PM", "Hyderabad"],
["Tuesday - October 10 2023", "ENGLAND vs BANGLADESH", "England Won", "10:30␣
 ↪AM", "Dharamsala"],
["Tuesday - October 10 2023", "SRI LANKA vs PAKISTAN", "Pakistan Won", "2:00␣
 ↪PM", "Hyderabad"],
["Wednesday - October 11 2023", "INDIA vs AFGHANISTAN", "India Won", "2:00 PM",␣
 ↪"Delhi"],
["Thursday - October 12 2023", "AUSTRALIA vs SOUTH AFRICA", "South Africa Won",␣
 ↪"2:00 PM", "Lucknow"],
["Friday - October 13 2023", "NEW ZEALAND vs BANGLADESH", "New Zealand Won", "2:
 ↪00 PM", "Chennai"],
["Saturday - October 14 2023", "INDIA vs PAKISTAN", "India Won", "2:00 PM",␣
 ↪"Ahmedabad"],
["Sunday - October 15 2023", "ENGLAND vs AFGHANISTAN", "Afghanistan Won", "2:00␣
 ↪PM", "Delhi"],
["Monday - October 16 2023", "AUSTRALIA vs SRI LANKA", "Australia Won", "2:00␣
 ↪PM", "Lucknow"],
["Tuesday - October 17 2023", "SOUTH AFRICA vs NETHERLANDS", "Netherlands Won",␣
 ↪"2:00 PM", "Dharamsala"],
["Wednesday - October 18 2023", "NEW ZEALAND vs AFGHANISTAN", "New Zealand␣
 ↪Won", "2:00 PM", "Chennai"],
["Thursday - October 19 2023", "INDIA vs BANGLADESH", "India Won", "2:00 PM",␣
 ↪"Pune"],
["Friday - October 20 2023", "AUSTRALIA vs PAKISTAN", "Australia Won", "2:00␣
 ↪PM", "Bengaluru"],
["Saturday - October 21 2023", "NETHERLANDS vs SRI LANKA", "Sri Lanka Won", "10:
 ↪30 AM", "Lucknow"],
["Saturday - October 21 2023", "ENGLAND vs SOUTH AFRICA", "South Africa Won",␣
 ↪"2:00 PM", "Mumbai"],
["Sunday - October 22 2023", "INDIA vs NEW ZEALAND", "India Won", "2:00 PM",␣
 ↪"Dharamsala"],
["Monday - October 23 2023", "PAKISTAN vs AFGHANISTAN", "Afganistan Won", "2:00␣
 ↪PM", "Chennai"],
["Tuesday - October 24 2023", "SOUTH AFRICA vs BANGLADESH", "South Africa Won",␣
 ↪"2:00 PM", "Mumbai"],
["Wednesday - October 25 2023", "AUSTRALIA vs NETHERLANDS", "Australia Won", "2:
 ↪00 PM", "Delhi"],
["Thursday - October 26 2023", "ENGLAND vs SRI LANKA", "Sri Lanka Won", "2:00␣
 ↪PM", "Bengaluru"],
["Friday - October 27 2023", "PAKISTAN vs SOUTH AFRICA", "South Africa Won", "2:
 ↪00 PM", "Chennai"],
["Saturday - October 28 2023", "AUSTRALIA vs NEW ZEALAND", "Australia Won", "10:
 ↪30 AM", "Dharamsala"],
["Saturday - October 28 2023", "NETHERLANDS vs BANGLADESH", "Netherlands Won",␣
 ↪"2:00 PM", "Kolkata"],
["Sunday - October 29 2023", "INDIA vs ENGLAND", "-", "2:00 PM", "Lucknow"],
["Monday - October 30 2023", "AFGHANISTAN vs SRI LANKA", "-", "2:00 PM",␣
 ↪"Pune"],
["Tuesday - October 31 2023", "PAKISTAN vs BANGLADESH", "-", "2:00 PM",␣
 ↪"Kolkata"],
["Wednesday - November 1 2023", "NEW ZEALAND vs SOUTH AFRICA", "-", "2:00 PM",␣
 ↪"Pune"],
["Thursday - November 2 2023", "INDIA vs SRI LANKA", "-", "2:00 PM", "Mumbai"],
["Friday - November 3 2023", "NETHERLANDS vs AFGHANISTAN", "-", "2:00 PM",␣
 ↪"Lucknow"],
["Saturday - November 4 2023", "NEW ZEALAND vs PAKISTAN", "-", "10:30 AM",␣
 ↪"Bengaluru"],
["Saturday - November 4 2023", "ENGLAND vs AUSTRALIA", "-", "2:00 PM",␣
 ↪"Ahmedabad"],
["Sunday - November 5 2023", "INDIA vs SOUTH AFRICA", "-", "2:00 PM",␣
 ↪"Kolkata"],
["Monday - November 6 2023", "BANGLADESH vs SRI LANKA", "-", "2:00 PM",␣
 ↪"Delhi"],
["Tuesday - November 7 2023", "AUSTRALIA vs AFGHANISTAN", "-", "2:00 PM",␣
 ↪"Mumbai"],
["Wednesday - November 8 2023", "ENGLAND vs NETHERLANDS", "-", "2:00 PM",␣
 ↪"Pune"],
["Thursday - November 9 2023", "NEW ZEALAND vs SRI LANKA", "-", "2:00 PM",␣
 ↪"Bengaluru"],
["Friday - November 10 2023", "SOUTH AFRICA vs AFGHANISTAN", "-", "2:00 PM",␣
 ↪"Ahmedabad"],
["Saturday - November 11 2023", "AUSTRALIA vs BANGLADESH", "-", "10:30 AM",␣
 ↪"Pune"],
["Saturday - November 11 2023", "ENGLAND vs PAKISTAN", "-", "2:00 PM",␣
 ↪"Kolkata"],
["Sunday - November 12 2023", "INDIA vs NETHERLANDS", "-", "2:00 PM",␣
 ↪"Bengaluru"],
["Wednesday - November 15 2023", "Semi-Final 1", "-", "2:00 PM", "Mumbai"],
["Thursday - November 16 2023", "Semi-Final 2", "-", "2:00 PM", "Kolkata"],
["Sunday - November 19 2023", "Final Match", "-", "2:00 PM", "Ahmedabad"],
]
```

[139]:

```
# Create a Pandas DataFrame 'df' from the 'data' list with specified column␣
 ↪names.

df = pd.DataFrame(data, columns = ["Day & Date", "Matches", "Status", "Time",␣
 ↪"Venue"])
df
```

[139]:

```
                      Day & Date                      Matches  \
0      Thursday - October 5 2023       ENGLAND vs NEW ZEALAND
1        Friday - October 6 2023      PAKISTAN vs NETHERLANDS
2      Saturday - October 7 2023     AFGHANISTAN vs BANGLADESH
3      Saturday - October 7 2023     SOUTH AFRICA vs SRI LANKA
4        Sunday - October 8 2023            AUSTRALIA vs INDIA
5        Monday - October 9 2023    NEW ZEALAND vs NETHERLANDS
6      Tuesday - October 10 2023         ENGLAND vs BANGLADESH
7      Tuesday - October 10 2023         SRI LANKA vs PAKISTAN
8    Wednesday - October 11 2023          INDIA vs AFGHANISTAN
9     Thursday - October 12 2023     AUSTRALIA vs SOUTH AFRICA
10      Friday - October 13 2023     NEW ZEALAND vs BANGLADESH
11    Saturday - October 14 2023             INDIA vs PAKISTAN
12      Sunday - October 15 2023        ENGLAND vs AFGHANISTAN
13      Monday - October 16 2023        AUSTRALIA vs SRI LANKA
14     Tuesday - October 17 2023   SOUTH AFRICA vs NETHERLANDS
15   Wednesday - October 18 2023    NEW ZEALAND vs AFGHANISTAN
16    Thursday - October 19 2023           INDIA vs BANGLADESH
17      Friday - October 20 2023         AUSTRALIA vs PAKISTAN
18    Saturday - October 21 2023       NETHERLANDS vs SRI LANKA
19    Saturday - October 21 2023       ENGLAND vs SOUTH AFRICA
20      Sunday - October 22 2023          INDIA vs NEW ZEALAND
21      Monday - October 23 2023       PAKISTAN vs AFGHANISTAN
22     Tuesday - October 24 2023    SOUTH AFRICA vs BANGLADESH
23   Wednesday - October 25 2023      AUSTRALIA vs NETHERLANDS
24    Thursday - October 26 2023          ENGLAND vs SRI LANKA
25      Friday - October 27 2023      PAKISTAN vs SOUTH AFRICA
```

```
26    Saturday - October 28 2023     AUSTRALIA vs NEW ZEALAND
27    Saturday - October 28 2023    NETHERLANDS vs BANGLADESH
28      Sunday - October 29 2023             INDIA vs ENGLAND
29      Monday - October 30 2023     AFGHANISTAN vs SRI LANKA
30     Tuesday - October 31 2023       PAKISTAN vs BANGLADESH
31   Wednesday - November 1 2023  NEW ZEALAND vs SOUTH AFRICA
32    Thursday - November 2 2023           INDIA vs SRI LANKA
33      Friday - November 3 2023   NETHERLANDS vs AFGHANISTAN
34    Saturday - November 4 2023      NEW ZEALAND vs PAKISTAN
35    Saturday - November 4 2023         ENGLAND vs AUSTRALIA
36      Sunday - November 5 2023        INDIA vs SOUTH AFRICA
37      Monday - November 6 2023      BANGLADESH vs SRI LANKA
38     Tuesday - November 7 2023     AUSTRALIA vs AFGHANISTAN
39   Wednesday - November 8 2023       ENGLAND vs NETHERLANDS
40    Thursday - November 9 2023     NEW ZEALAND vs SRI LANKA
41     Friday - November 10 2023  SOUTH AFRICA vs AFGHANISTAN
42   Saturday - November 11 2023      AUSTRALIA vs BANGLADESH
43   Saturday - November 11 2023          ENGLAND vs PAKISTAN
44     Sunday - November 12 2023         INDIA vs NETHERLANDS
45  Wednesday - November 15 2023                 Semi-Final 1
46   Thursday - November 16 2023                 Semi-Final 2
47     Sunday - November 19 2023                  Final Match

              Status      Time       Venue
0    New Zealand Won   2:00 PM   Ahmedabad
1       Pakistan Won   2:00 PM   Hyderabad
2     Bangladesh Won  10:30 AM  Dharamsala
3   South Africa Won   2:00 PM       Delhi
4          India Won   2:00 PM     Chennai
5    New Zealand Won   2:00 PM   Hyderabad
6        England Won  10:30 AM  Dharamsala
7       Pakistan Won   2:00 PM   Hyderabad
8          India Won   2:00 PM       Delhi
9   South Africa Won   2:00 PM     Lucknow
10   New Zealand Won   2:00 PM     Chennai
11         India Won   2:00 PM   Ahmedabad
12   Afghanistan Won   2:00 PM       Delhi
13     Australia Won   2:00 PM     Lucknow
14   Netherlands Won   2:00 PM  Dharamsala
15   New Zealand Won   2:00 PM     Chennai
16         India Won   2:00 PM        Pune
17     Australia Won   2:00 PM   Bengaluru
18     Sri Lanka Won  10:30 AM     Lucknow
19  South Africa Won   2:00 PM      Mumbai
20         India Won   2:00 PM  Dharamsala
21    Afganistan Won   2:00 PM     Chennai
22  South Africa Won   2:00 PM      Mumbai
```

```
23      Australia Won   2:00 PM       Delhi
24      Sri Lanka Won   2:00 PM   Bengaluru
25   South Africa Won   2:00 PM     Chennai
26      Australia Won  10:30 AM  Dharamsala
27    Netherlands Won   2:00 PM     Kolkata
28                  -   2:00 PM     Lucknow
29                  -   2:00 PM        Pune
30                  -   2:00 PM     Kolkata
31                  -   2:00 PM        Pune
32                  -   2:00 PM      Mumbai
33                  -   2:00 PM     Lucknow
34                  -  10:30 AM   Bengaluru
35                  -   2:00 PM   Ahmedabad
36                  -   2:00 PM     Kolkata
37                  -   2:00 PM       Delhi
38                  -   2:00 PM      Mumbai
39                  -   2:00 PM        Pune
40                  -   2:00 PM   Bengaluru
41                  -   2:00 PM   Ahmedabad
42                  -  10:30 AM        Pune
43                  -   2:00 PM     Kolkata
44                  -   2:00 PM   Bengaluru
45                  -   2:00 PM      Mumbai
46                  -   2:00 PM     Kolkata
47                  -   2:00 PM   Ahmedabad
```

[140]:
```
# Get the shape (number of rows and columns) of the DataFrame 'df'.
df.shape
```

[140]:
```
(48, 5)
```

[141]:
```
#  Retrieve the column names of the DataFrame 'df'.
df.columns
```

[141]:
```
Index(['Day & Date', 'Matches', 'Status', 'Time', 'Venue'], dtype='object')
```

[142]:
```
# Calculate and count the number of duplicated rows in the DataFrame 'df'.
df.duplicated().sum()
```

[142]:
```
0
```

[143]:
```
# Count the number of missing (null) values in each column of the DataFrame␣
 ↪'df'.
df.isnull().sum()
```

[143]:
```
Day & Date    0
Matches       0
Status        0
```

```
Time          0
Venue         0
dtype: int64
```

[144]:
```
# Display a summary of the DataFrame 'df' including data types, non-null values, and memory usage.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48 entries, 0 to 47
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Day & Date  48 non-null     object
 1   Matches     48 non-null     object
 2   Status      48 non-null     object
 3   Time        48 non-null     object
 4   Venue       48 non-null     object
dtypes: object(5)
memory usage: 2.0+ KB
```

[145]:
```
# Extract date components from the 'Day & Date' column using a regular expression pattern.
# The pattern captures the day of the week, month, day of the month, and year.

pattern = r'(\w+)\s[--]\s(\w+)\s(\d+)\s(\d{4})'
df[['Day of the Week', 'Month', 'Day of the Month', 'Year']] = df['Day & Date'].str.extract(pattern)
```

[146]:
```
# Save the data from the DataFrame 'df' to a CSV file named "Cricket_worldcup_2023.csv."
# The 'index=False' parameter ensures that the index column is not included in the CSV file.
df.to_csv("Cricket_worldcup_2023.csv", index=False)
```

[147]:
```
# Display the DataFrame 'df,' showing its contents in the current Jupyter Notebook cell.
df
```

[147]:
```
                    Day & Date                      Matches  \
0      Thursday - October 5 2023       ENGLAND vs NEW ZEALAND
1        Friday - October 6 2023      PAKISTAN vs NETHERLANDS
2      Saturday - October 7 2023    AFGHANISTAN vs BANGLADESH
3      Saturday - October 7 2023    SOUTH AFRICA vs SRI LANKA
4        Sunday - October 8 2023           AUSTRALIA vs INDIA
5        Monday - October 9 2023   NEW ZEALAND vs NETHERLANDS
6      Tuesday - October 10 2023        ENGLAND vs BANGLADESH
```

```
7      Tuesday - October 10 2023        SRI LANKA vs PAKISTAN
8    Wednesday - October 11 2023         INDIA vs AFGHANISTAN
9     Thursday - October 12 2023    AUSTRALIA vs SOUTH AFRICA
10      Friday - October 13 2023    NEW ZEALAND vs BANGLADESH
11    Saturday - October 14 2023            INDIA vs PAKISTAN
12      Sunday - October 15 2023       ENGLAND vs AFGHANISTAN
13      Monday - October 16 2023       AUSTRALIA vs SRI LANKA
14     Tuesday - October 17 2023  SOUTH AFRICA vs NETHERLANDS
15   Wednesday - October 18 2023   NEW ZEALAND vs AFGHANISTAN
16    Thursday - October 19 2023          INDIA vs BANGLADESH
17      Friday - October 20 2023        AUSTRALIA vs PAKISTAN
18    Saturday - October 21 2023     NETHERLANDS vs SRI LANKA
19    Saturday - October 21 2023      ENGLAND vs SOUTH AFRICA
20      Sunday - October 22 2023         INDIA vs NEW ZEALAND
21      Monday - October 23 2023      PAKISTAN vs AFGHANISTAN
22     Tuesday - October 24 2023   SOUTH AFRICA vs BANGLADESH
23   Wednesday - October 25 2023     AUSTRALIA vs NETHERLANDS
24    Thursday - October 26 2023         ENGLAND vs SRI LANKA
25      Friday - October 27 2023     PAKISTAN vs SOUTH AFRICA
26    Saturday - October 28 2023     AUSTRALIA vs NEW ZEALAND
27    Saturday - October 28 2023    NETHERLANDS vs BANGLADESH
28      Sunday - October 29 2023             INDIA vs ENGLAND
29      Monday - October 30 2023     AFGHANISTAN vs SRI LANKA
30     Tuesday - October 31 2023       PAKISTAN vs BANGLADESH
31   Wednesday - November 1 2023  NEW ZEALAND vs SOUTH AFRICA
32    Thursday - November 2 2023           INDIA vs SRI LANKA
33      Friday - November 3 2023   NETHERLANDS vs AFGHANISTAN
34    Saturday - November 4 2023      NEW ZEALAND vs PAKISTAN
35    Saturday - November 4 2023         ENGLAND vs AUSTRALIA
36      Sunday - November 5 2023        INDIA vs SOUTH AFRICA
37      Monday - November 6 2023      BANGLADESH vs SRI LANKA
38     Tuesday - November 7 2023     AUSTRALIA vs AFGHANISTAN
39   Wednesday - November 8 2023       ENGLAND vs NETHERLANDS
40    Thursday - November 9 2023     NEW ZEALAND vs SRI LANKA
41     Friday - November 10 2023  SOUTH AFRICA vs AFGHANISTAN
42   Saturday - November 11 2023      AUSTRALIA vs BANGLADESH
43   Saturday - November 11 2023          ENGLAND vs PAKISTAN
44     Sunday - November 12 2023         INDIA vs NETHERLANDS
45  Wednesday - November 15 2023                 Semi-Final 1
46   Thursday - November 16 2023                 Semi-Final 2
47     Sunday - November 19 2023                  Final Match

              Status      Time       Venue Day of the Week    Month  \
0    New Zealand Won   2:00 PM   Ahmedabad        Thursday  October
1       Pakistan Won   2:00 PM   Hyderabad          Friday  October
2     Bangladesh Won  10:30 AM  Dharamsala        Saturday  October
3   South Africa Won   2:00 PM       Delhi        Saturday  October
```

```
4          India Won   2:00 PM     Chennai          Sunday   October
5    New Zealand Won   2:00 PM   Hyderabad          Monday   October
6        England Won  10:30 AM  Dharamsala         Tuesday   October
7       Pakistan Won   2:00 PM   Hyderabad         Tuesday   October
8          India Won   2:00 PM       Delhi       Wednesday   October
9   South Africa Won   2:00 PM     Lucknow        Thursday   October
10   New Zealand Won   2:00 PM     Chennai          Friday   October
11         India Won   2:00 PM   Ahmedabad        Saturday   October
12   Afghanistan Won   2:00 PM       Delhi          Sunday   October
13     Australia Won   2:00 PM     Lucknow          Monday   October
14   Netherlands Won   2:00 PM  Dharamsala         Tuesday   October
15   New Zealand Won   2:00 PM     Chennai       Wednesday   October
16         India Won   2:00 PM        Pune        Thursday   October
17     Australia Won   2:00 PM   Bengaluru          Friday   October
18     Sri Lanka Won  10:30 AM     Lucknow        Saturday   October
19  South Africa Won   2:00 PM      Mumbai        Saturday   October
20         India Won   2:00 PM  Dharamsala          Sunday   October
21    Afganistan Won   2:00 PM     Chennai          Monday   October
22  South Africa Won   2:00 PM      Mumbai         Tuesday   October
23     Australia Won   2:00 PM       Delhi       Wednesday   October
24     Sri Lanka Won   2:00 PM   Bengaluru        Thursday   October
25  South Africa Won   2:00 PM     Chennai          Friday   October
26     Australia Won  10:30 AM  Dharamsala        Saturday   October
27   Netherlands Won   2:00 PM     Kolkata        Saturday   October
28                 -   2:00 PM     Lucknow          Sunday   October
29                 -   2:00 PM        Pune          Monday   October
30                 -   2:00 PM     Kolkata         Tuesday   October
31                 -   2:00 PM        Pune       Wednesday  November
32                 -   2:00 PM      Mumbai        Thursday  November
33                 -   2:00 PM     Lucknow          Friday  November
34                 -  10:30 AM   Bengaluru        Saturday  November
35                 -   2:00 PM   Ahmedabad        Saturday  November
36                 -   2:00 PM     Kolkata          Sunday  November
37                 -   2:00 PM       Delhi          Monday  November
38                 -   2:00 PM      Mumbai         Tuesday  November
39                 -   2:00 PM        Pune       Wednesday  November
40                 -   2:00 PM   Bengaluru        Thursday  November
41                 -   2:00 PM   Ahmedabad          Friday  November
42                 -  10:30 AM        Pune        Saturday  November
43                 -   2:00 PM     Kolkata        Saturday  November
44                 -   2:00 PM   Bengaluru          Sunday  November
45                 -   2:00 PM      Mumbai       Wednesday  November
46                 -   2:00 PM     Kolkata        Thursday  November
47                 -   2:00 PM   Ahmedabad          Sunday  November

   Day of the Month  Year
0                 5  2023
```

| | | |
|---|---|---|
| 1 | 6 | 2023 |
| 2 | 7 | 2023 |
| 3 | 7 | 2023 |
| 4 | 8 | 2023 |
| 5 | 9 | 2023 |
| 6 | 10 | 2023 |
| 7 | 10 | 2023 |
| 8 | 11 | 2023 |
| 9 | 12 | 2023 |
| 10 | 13 | 2023 |
| 11 | 14 | 2023 |
| 12 | 15 | 2023 |
| 13 | 16 | 2023 |
| 14 | 17 | 2023 |
| 15 | 18 | 2023 |
| 16 | 19 | 2023 |
| 17 | 20 | 2023 |
| 18 | 21 | 2023 |
| 19 | 21 | 2023 |
| 20 | 22 | 2023 |
| 21 | 23 | 2023 |
| 22 | 24 | 2023 |
| 23 | 25 | 2023 |
| 24 | 26 | 2023 |
| 25 | 27 | 2023 |
| 26 | 28 | 2023 |
| 27 | 28 | 2023 |
| 28 | 29 | 2023 |
| 29 | 30 | 2023 |
| 30 | 31 | 2023 |
| 31 | 1 | 2023 |
| 32 | 2 | 2023 |
| 33 | 3 | 2023 |
| 34 | 4 | 2023 |
| 35 | 4 | 2023 |
| 36 | 5 | 2023 |
| 37 | 6 | 2023 |
| 38 | 7 | 2023 |
| 39 | 8 | 2023 |
| 40 | 9 | 2023 |
| 41 | 10 | 2023 |
| 42 | 11 | 2023 |
| 43 | 11 | 2023 |
| 44 | 12 | 2023 |
| 45 | 15 | 2023 |
| 46 | 16 | 2023 |
| 47 | 19 | 2023 |

[148]:

```
# Display a summary of the DataFrame's information, including data types,␣
 ↪non-null values, and memory usage.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48 entries, 0 to 47
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Day & Date        48 non-null     object
 1   Matches           48 non-null     object
 2   Status            48 non-null     object
 3   Time              48 non-null     object
 4   Venue             48 non-null     object
 5   Day of the Week   48 non-null     object
 6   Month             48 non-null     object
 7   Day of the Month  48 non-null     object
 8   Year              48 non-null     object
dtypes: object(9)
memory usage: 3.5+ KB
```

[149]:

```
# Convert the 'Month' column to numerical month values (e.g., 'October' to 10)␣
 ↪based on the specified format.
df['Month'] = pd.to_datetime(df['Month'], format='%B').dt.month

# Convert the 'Day of the Month' and 'Year' columns to numeric values, handling␣
 ↪errors by converting non-numeric values to NaN.
df['Day of the Month'] = pd.to_numeric(df['Day of the Month'], errors =␣
 ↪'coerce')
df['Year'] = pd.to_numeric(df['Year'], errors = 'coerce')
```

[150]:

```
# Display a summary of the DataFrame's information, including data types,␣
 ↪non-null values, and memory usage.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48 entries, 0 to 47
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Day & Date        48 non-null     object
 1   Matches           48 non-null     object
 2   Status            48 non-null     object
 3   Time              48 non-null     object
 4   Venue             48 non-null     object
 5   Day of the Week   48 non-null     object
 6   Month             48 non-null     int32
 7   Day of the Month  48 non-null     int64
 8   Year                 48 non-null     int64
dtypes: int32(1), int64(2), object(6)
memory usage: 3.3+ KB
```

[151]:
```
# Generate a summary of descriptive statistics for the DataFrame 'df,'␣
 ↪including count, mean, standard deviation, and more.
df.describe()
```

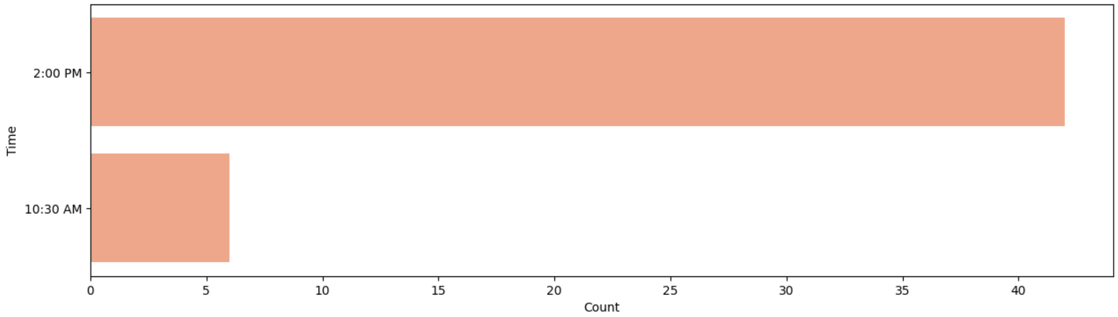
[151]:
```
           Month  Day of the Month    Year
count  48.000000         48.000000    48.0
mean   10.354167         14.479167  2023.0
std     0.483321          8.412224     0.0
min    10.000000          1.000000  2023.0
25%    10.000000          7.750000  2023.0
50%    10.000000         12.500000  2023.0
75%    11.000000         21.000000  2023.0
max    11.000000         31.000000  2023.0
```

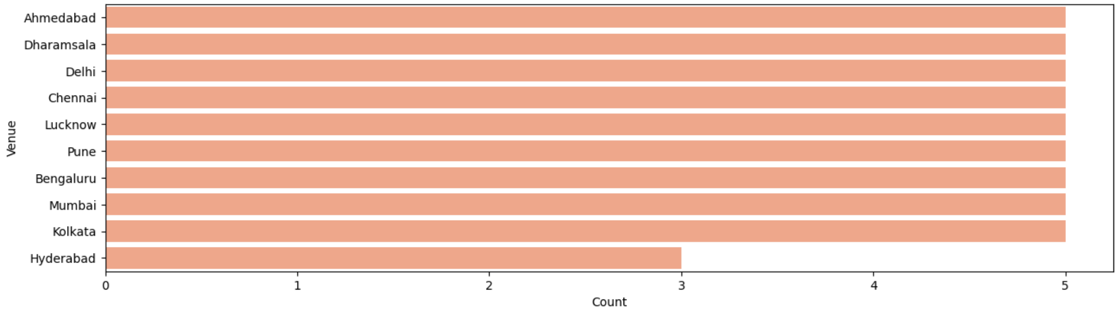
[152]:
```
# Calculate the number of unique values in each column of the DataFrame 'df.'
df.nunique()
```

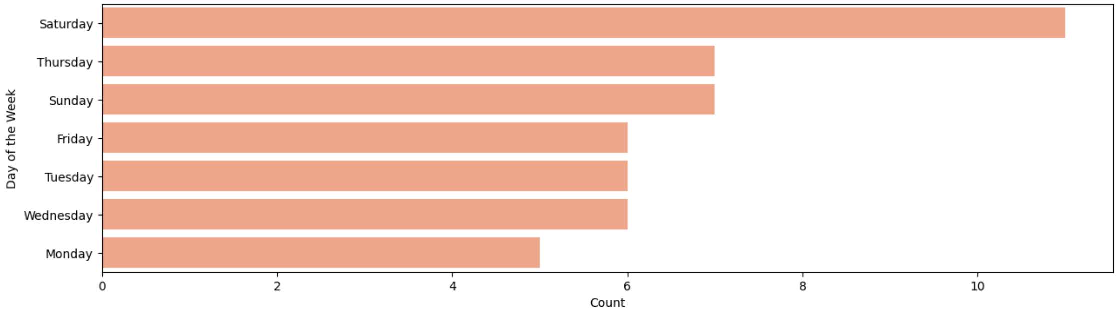
[152]:
```
Day & Date          42
Matches             48
Status              12
Time                 2
Venue               10
Day of the Week      7
Month                2
Day of the Month    31
Year                 1
dtype: int64
```

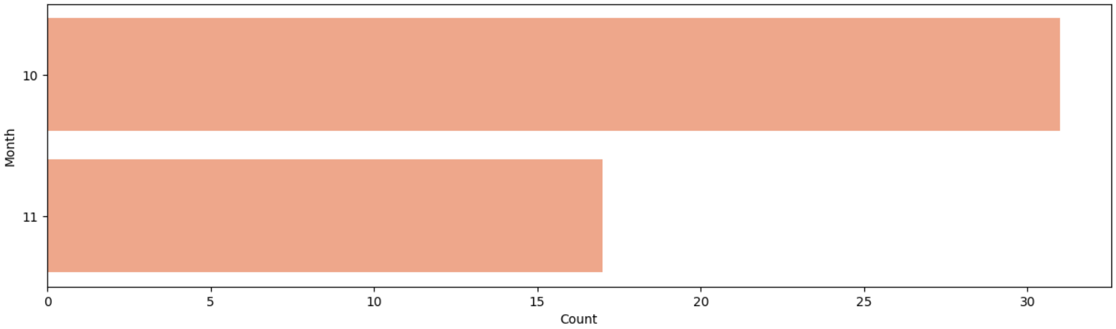
[153]:
```
# Define a list of feature column names, which are selected for analysis or␣
 ↪modeling purposes.
features = ['Time', 'Venue', 'Day of the Week', 'Month']
```

[154]:
```
# Create count plots for each feature in the 'features' list.
# The count plots display the distribution of each feature's values.

# Create a new figure with a specified size.
# Generate a count plot using Seaborn, displaying the count of unique values␣
 ↪for the feature.
# Set the x-axis label to 'Count.'
# Set the y-axis label to the name of the feature.
# Display the count plot.

for feature in features:
    plt.figure(figsize=(15,4))
    sns.countplot(data=df, y=feature, order=df[feature].value_counts().
↪index,color ="LightSalmon")
    plt.xlabel('Count')
    plt.ylabel(feature)
    plt.show()
```



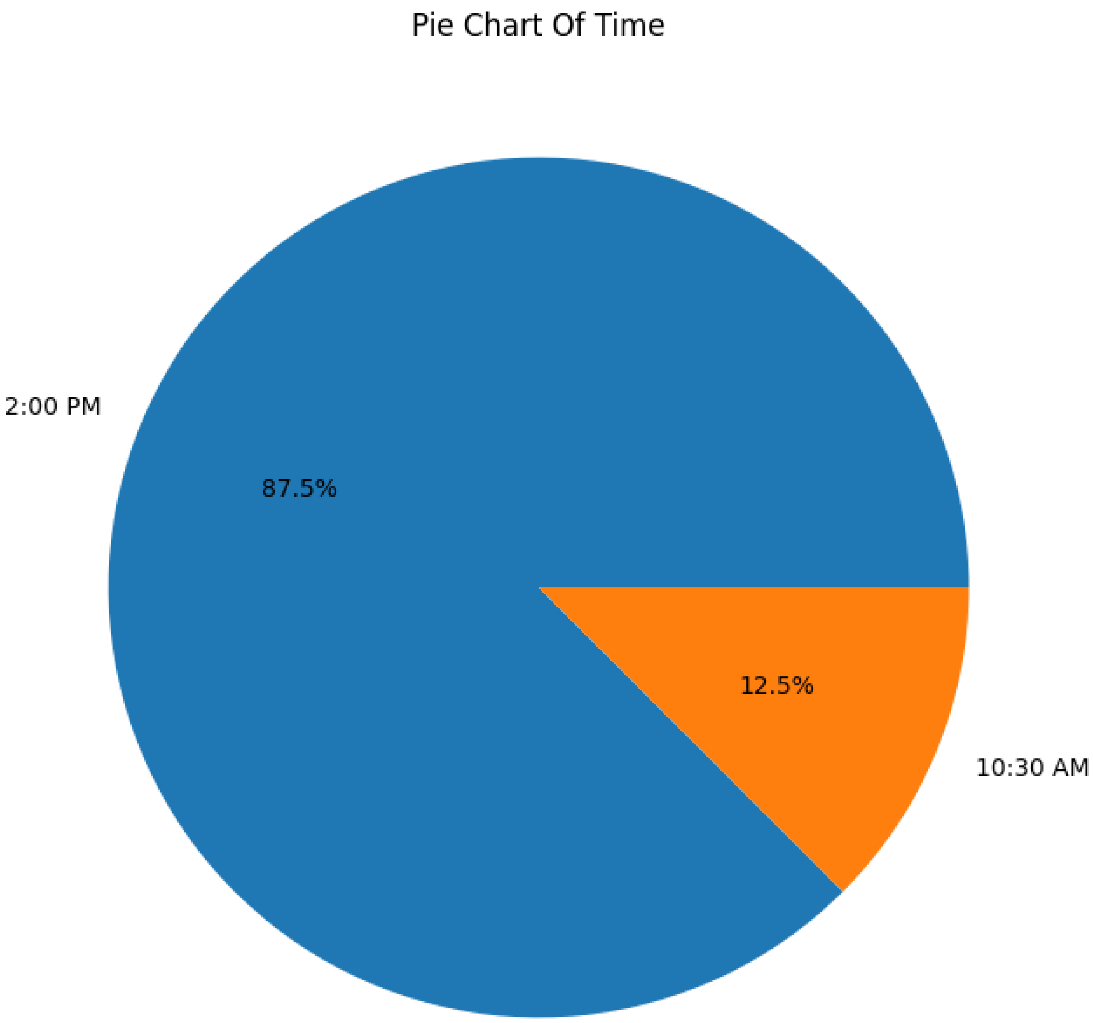


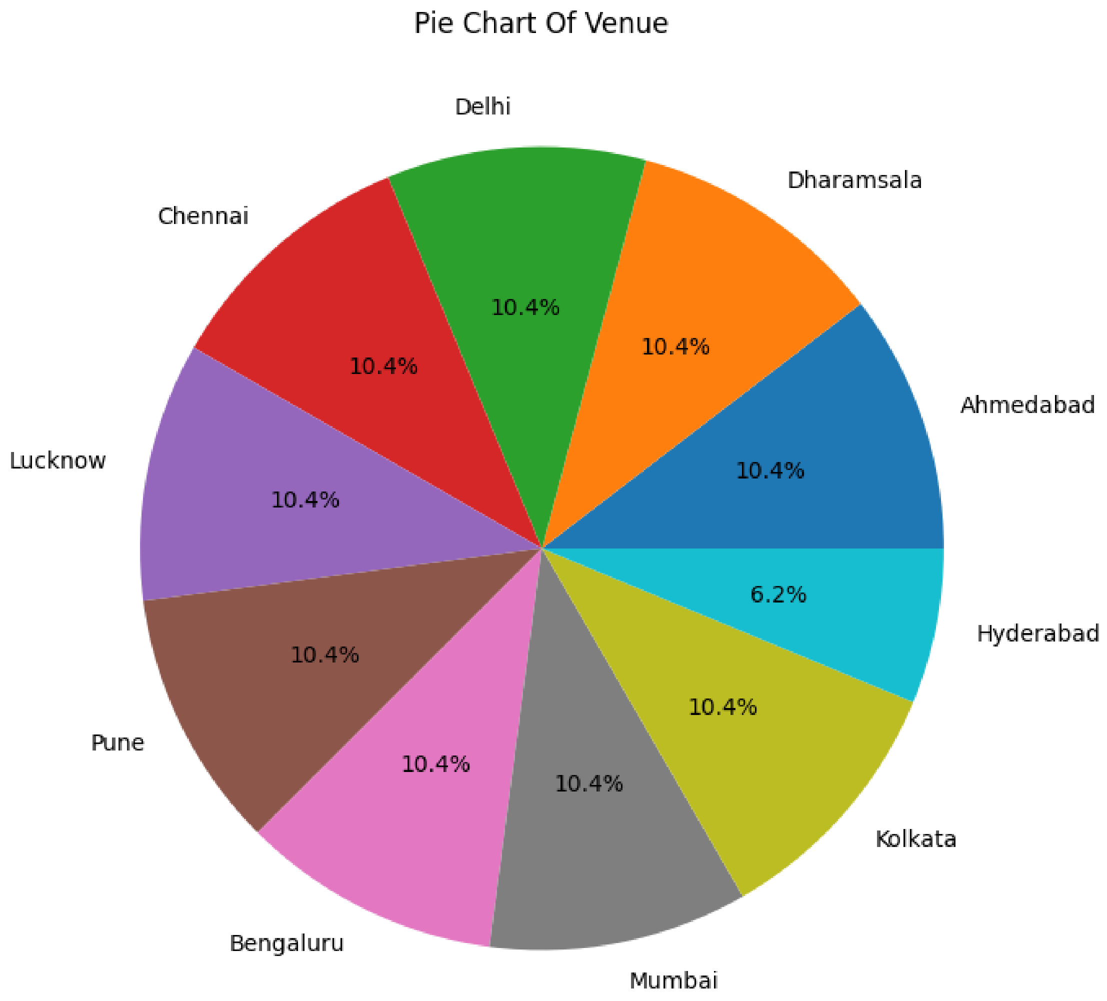


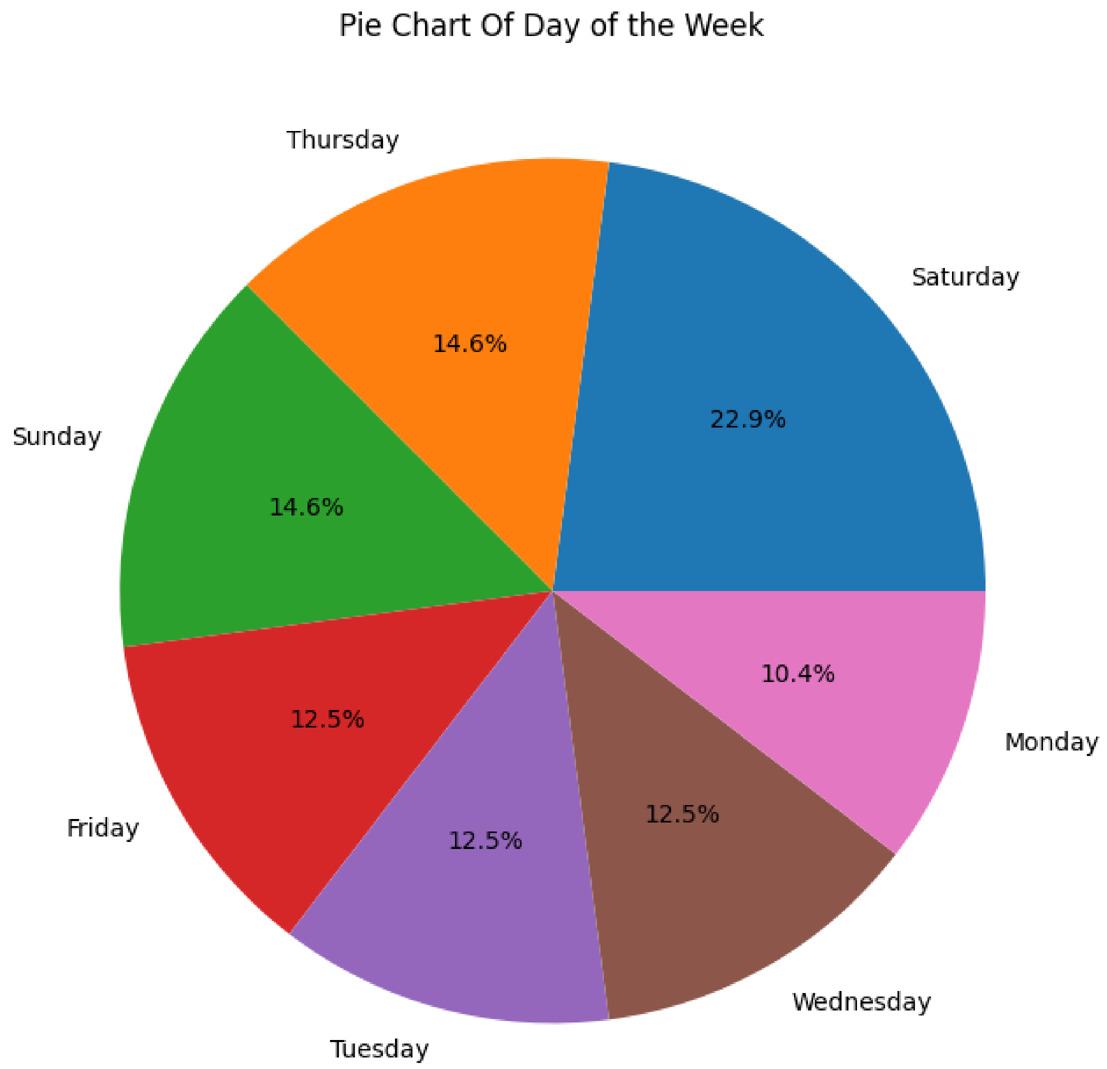


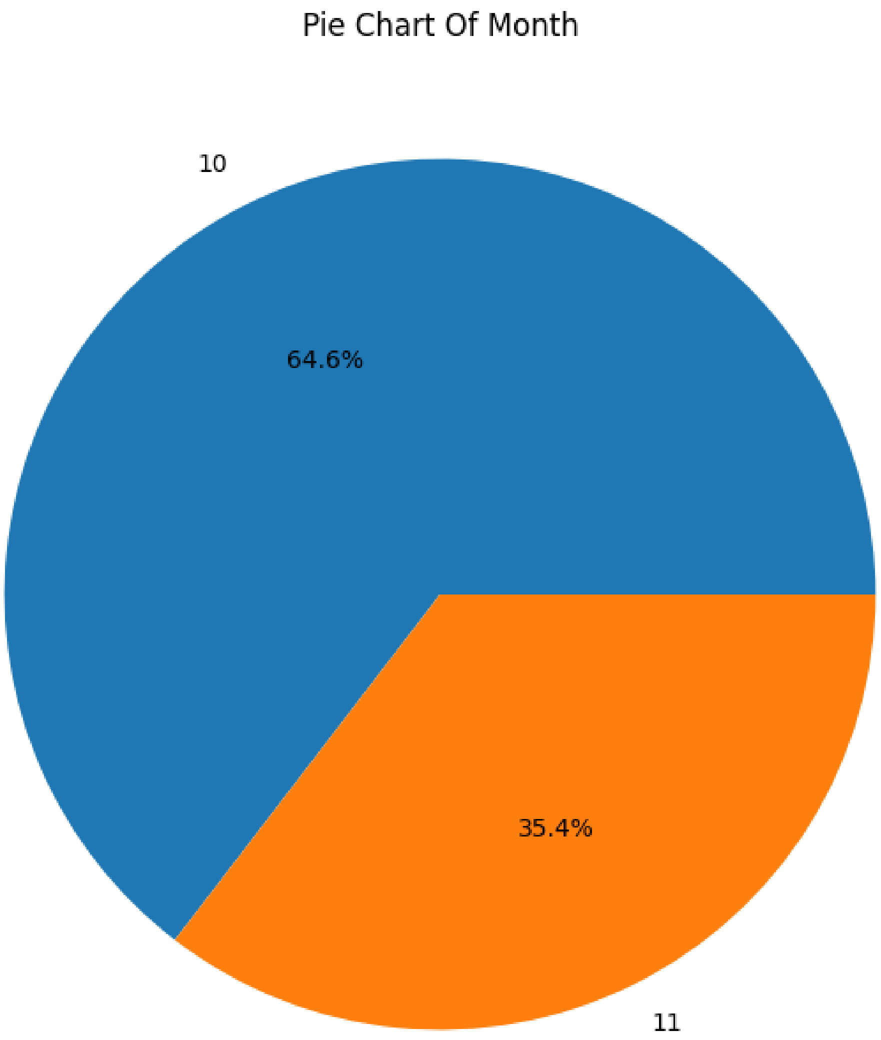
[155]:

```
# Create count plots for each feature in the 'features' list.
# The count plots display the distribution of each feature's values.
# Create a new figure with a specified size.
# Generate a count plot using Seaborn, displaying the count of unique values␣
 ↪for the feature.
# Set the x-axis label to 'Count.'
# Set the y-axis label to the name of the feature.
# Display the count plot.


for feature in features:
    plt.figure(figsize=(8,8))
    plt.title(f'Pie Chart Of {feature}')
    #df[feature].value_counts().plot.pie(autopct='%1.1f%%')
    df[feature].value_counts().plot.pie(autopct='%1.1f%%')
    plt.ylabel('')
    plt.show()
```


Pie Chart Of Time
2:00 PM
87.5%
12.5%
10:30 AM

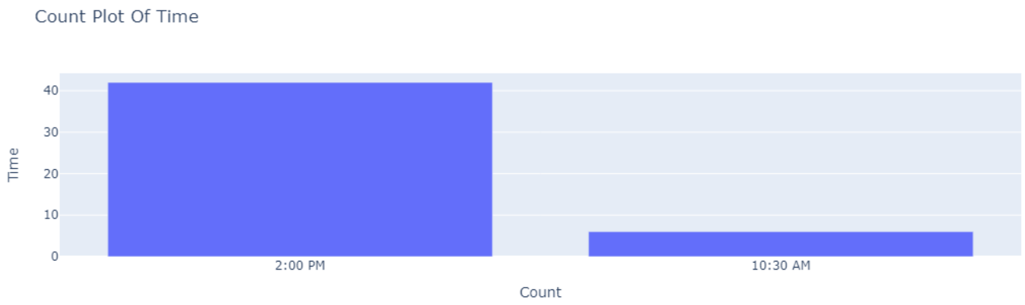

Pie Chart Of Venue
Delhi
10.4%
Dharamsala
10.4%
Chennai
10.4%
Ahmedabad
10.4%
Lucknow
10.4%
Hyderabad
6.2%
Pune
10.4%
Kolkata
10.4%
Bengaluru
10.4%
Mumbai
10.4%


Pie Chart Of Day of the Week
Thursday
Saturday
14.6%
22.9%
Sunday
14.6%
12.5%
10.4%
Monday
Friday
12.5%
12.5%
Wednesday
Tuesday

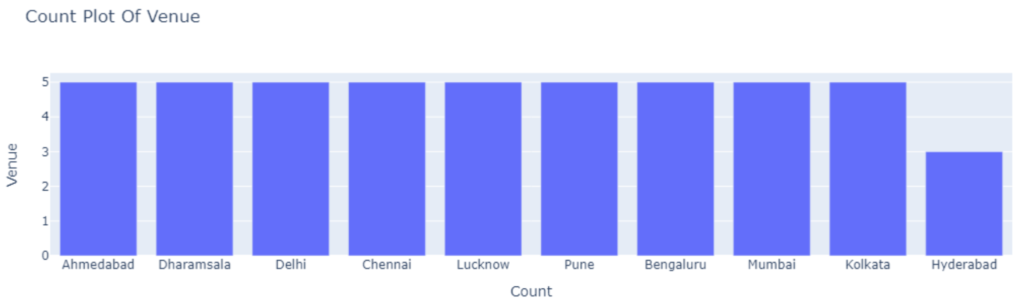


[156]:
```
# Create count bar plots for each feature in the 'features' list using Plotly␣
↪Express.
# Create a bar plot with the count of unique values for the current feature.
# Set the x-axis title to 'Count.'
# Set the y-axis title to the name of the current feature.
# Show the count bar plot.


for feature in features:
    count_fig = px.bar(df, x=df[feature].value_counts().index, y=df[feature].
↪value_counts().values,title=f' Count Plot Of {feature}',)
```

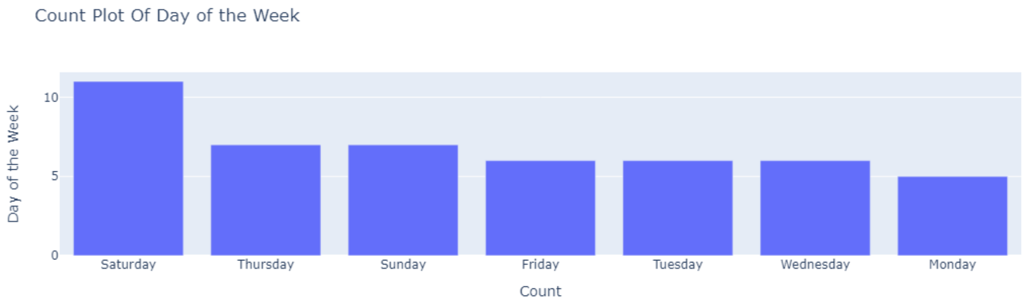
```
    count_fig.update_xaxes(title_text='Count')
    count_fig.update_yaxes(title_text=feature)
    count_fig.show()
```



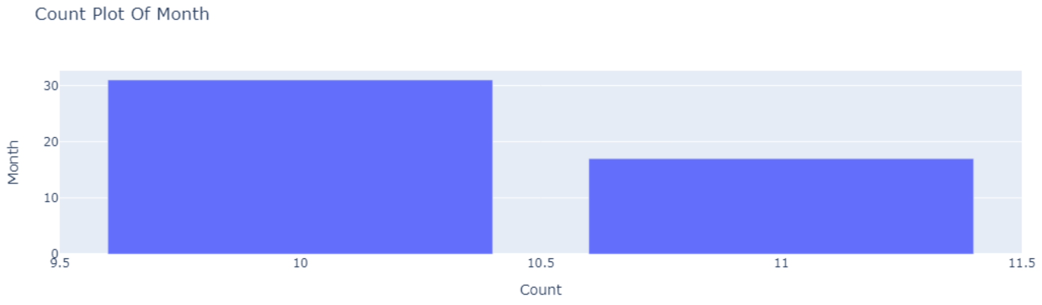
Count Plot Of Time



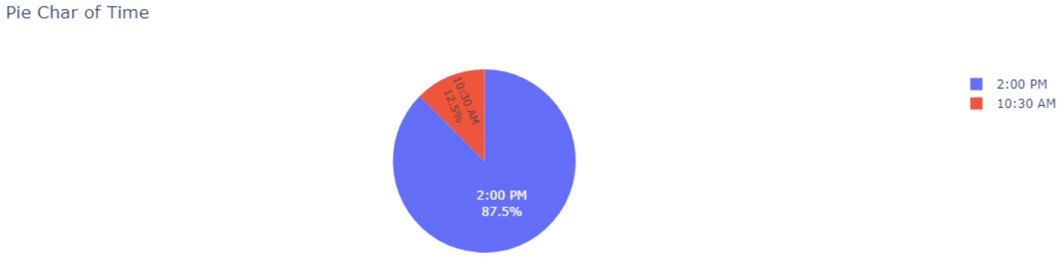
Count Plot Of Venue



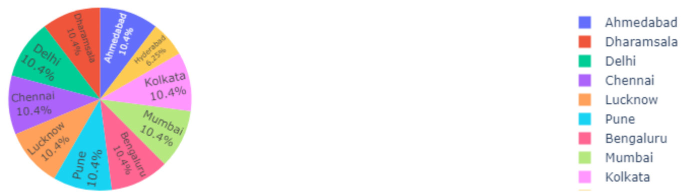
Count Plot Of Day of the Week



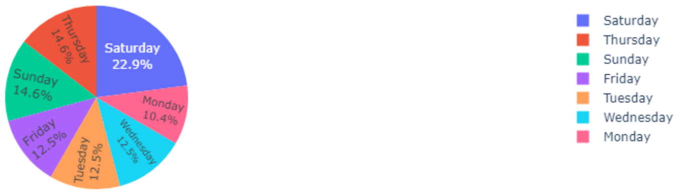
```
[157]: # Create pie charts for each feature in the 'features' list using Plotly␣
         ↪Express.
       # Create a pie chart with unique value names and their corresponding counts as␣
         ↪values.
       # Configure pie chart settings, placing the percentage labels inside the chart␣
         ↪slices.
       # Show the pie chart.

       for feature in features:
           pie_fig = px.pie(df, names=df[feature].value_counts().index,␣
         ↪values=df[feature].value_counts().values,
                            title=f'Pie Char of {feature}')
           pie_fig.update_traces(textposition = 'inside', textinfo='percent+label')
           pie_fig.show()
```



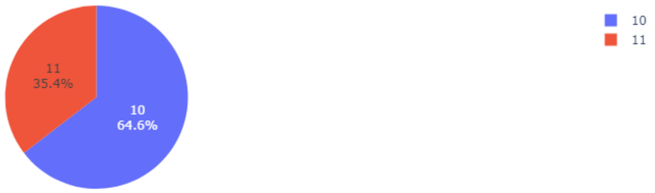
Pie Char of Venue



Pie Char of Day of the Week

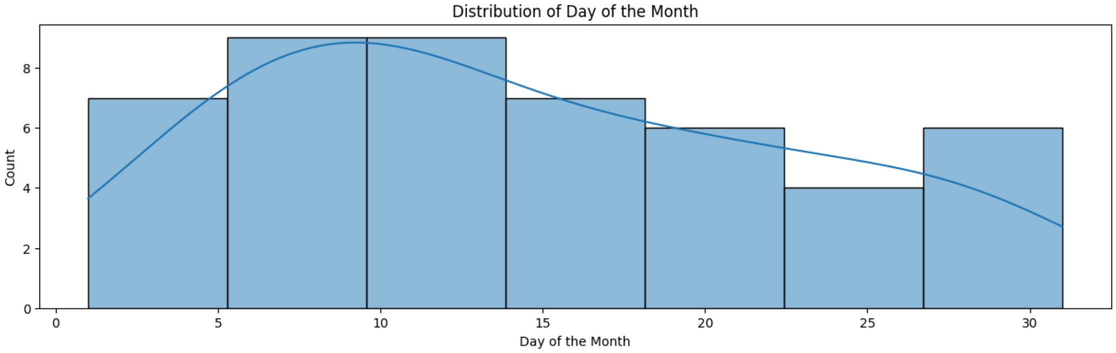


Pie Char of Month



[158]:
```
# Define a list of numerical columns that specifically includes 'Day of the Month.'
numerical_columns = ['Day of the Month']
```

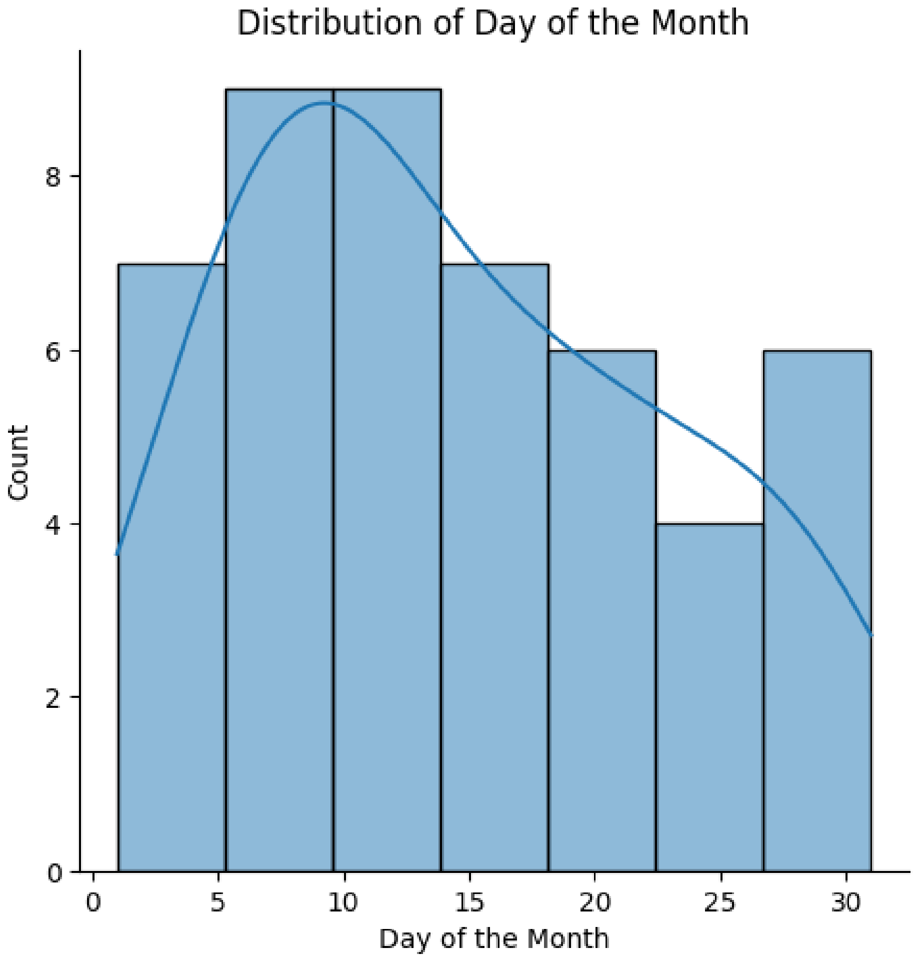
[159]:
```
# Create histograms for each column in the 'numerical_columns' list.
# Create a new figure with a specified size.
# Generate a histogram of the data in the current numerical column, including a␣
 ↪kernel density estimate (KDE).
# Set the title of the histogram based on the current column.
# Set the x-axis label to the name of the current column.
# Display the histogram.

for column in numerical_columns:
    plt.figure(figsize=(15,4))
    sns.histplot(df[column], kde=True)
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)
    plt.show()
```



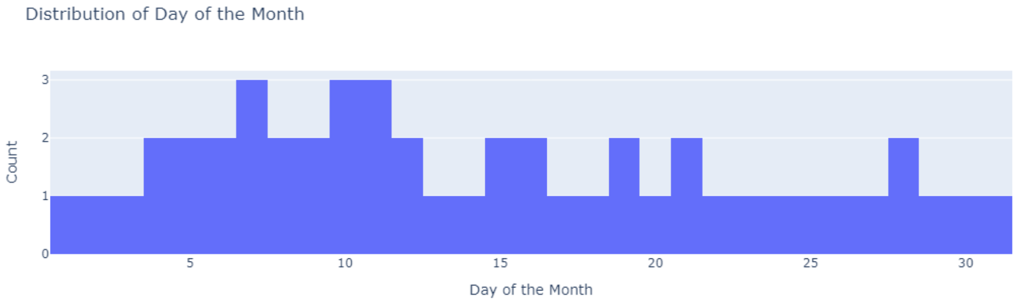
[160]:

```
# Create distribution plots for each column in the 'numerical_columns' list.
# Create a new figure with a specified size.
# Generate a distribution plot of the data in the current numerical column,␣
 ↪including a kernel density estimate (KDE).
# Set the title of the distribution plot based on the current column.
# Set the x-axis label to the name of the current column.
# Display the distribution plot.


for column in numerical_columns:
    plt.figure(figsize=(15,4))
    sns.displot(df[column], kde=True)
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)
    plt.show()
```

```
<Figure size 1500x400 with 0 Axes>
```



[161]:
```
# Create histogram plots for each column in the 'numerical_columns' list using␣
↪Plotly Express.
# Create a histogram plot with a specified number of bins, labels, and title␣
↪based on the current column.
# Update the layout to set the x-axis and y-axis titles.
# Show the histogram plot.

for column in numerical_columns:
    fig = px.histogram(df, x=column, nbins=50, labels={column: f'Distribution␣
↪of {column}'}, title=f'Distribution of {column}')
    fig.update_layout(xaxis_title=column, yaxis_title='Count')
    fig.show()
```

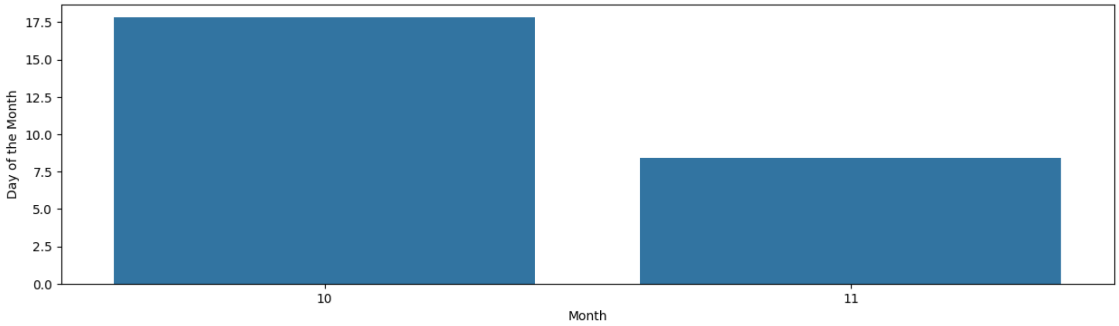


[162]:
```
# Create a cross-tabulation (cross-tab) between the 'Month' and 'Day of the␣
 ↪Week' columns in the DataFrame 'df'.
cross_tab = pd.crosstab(df['Month'],df['Day of the Week'])
```

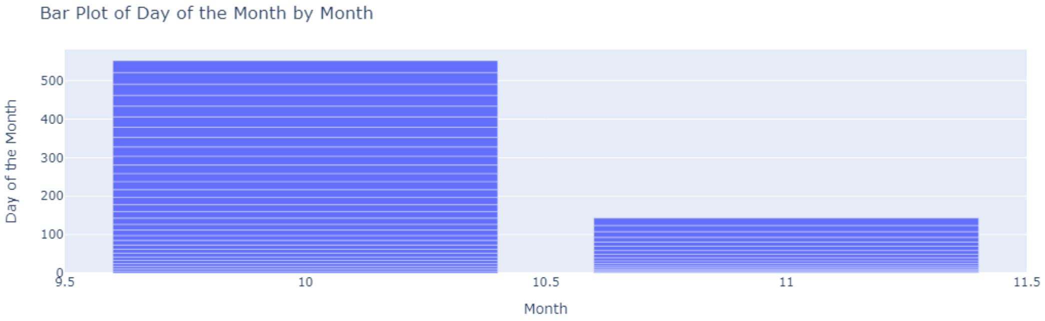
[163]:
```
# 'cross_tab' is a cross-tabulation (cross-tab) DataFrame that displays the␣
 ↪frequency of combinations of months and days of the week.
cross_tab
```

[163]:
```
Day of the Week  Friday  Monday  Saturday  Sunday  Thursday  Tuesday  \
Month
10                    4       4         7       4         4        5
11                    2       1         4       3         3        1

Day of the Week  Wednesday
Month
10                       3
11                       3
```

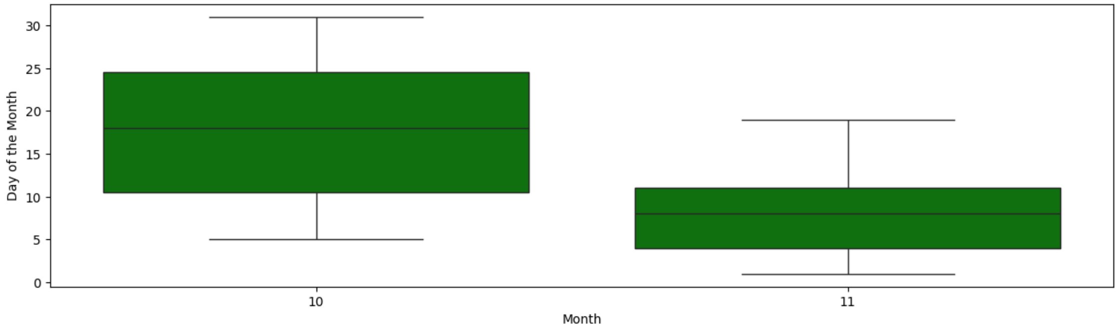
[164]:
```
# Create a bar plot to visualize the relationship between the 'Month' and 'Day␣
 ↪of the Month' columns.
# The 'ci' parameter is set to 'None' to suppress confidence intervals.

plt.figure(figsize=(15,4))
sns.barplot(x='Month', y='Day of the Month', ci=None,data=df)
plt.show()
```



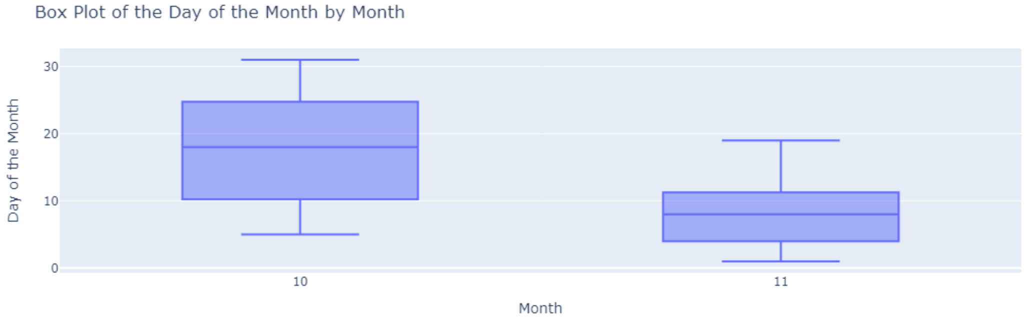
```
[165]: # Create a bar plot to visualize the relationship between 'Month' and 'Day of␣
        ↪the Month' using Plotly Express.
       # Update the layout to set the plot title.
       # Show the bar plot.

       fig = px.bar(df, x='Month', y='Day of the Month')
       fig.update_layout(title='Bar Plot of Day of the Month by Month')
       fig.show()
```



```
[166]: # Create a bar plot to visualize the relationship between 'Month' and 'Day of␣
        ↪the Month' using Plotly Express.
       # Update the layout to set the plot title.
       # Show the bar plot.

       plt.figure(figsize=(15,4))
       sns.boxplot(x='Month', y='Day of the Month', data=df, color="green")
       plt.show()
```

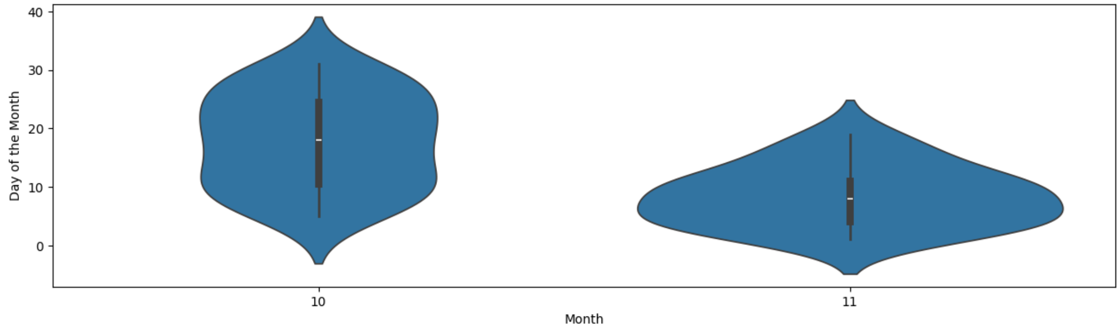


[167]:
```
# Create a box plot to visualize the distribution and relationship between␣
↪'Month' and 'Day of the Month' using Plotly Express.
# Update the layout to set the plot title.
# Show the box plot.

fig = px.box(df, x='Month', y='Day of the Month')
fig.update_layout(title ='Box Plot of the Day of the Month by Month')
fig.show()
```

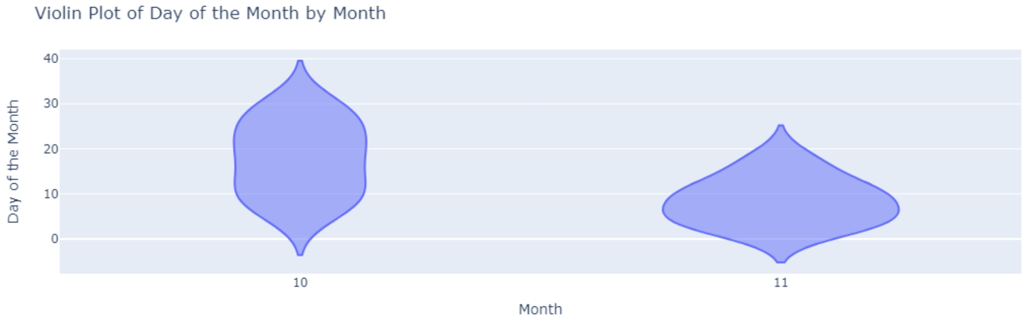


[168]:
```
# Create a violin plot to visualize the distribution and relationship between␣
↪'Month' and 'Day of the Month' columns.
plt.figure(figsize=(15,4))
sns.violinplot(x='Month', y='Day of the Month', data=df)
```

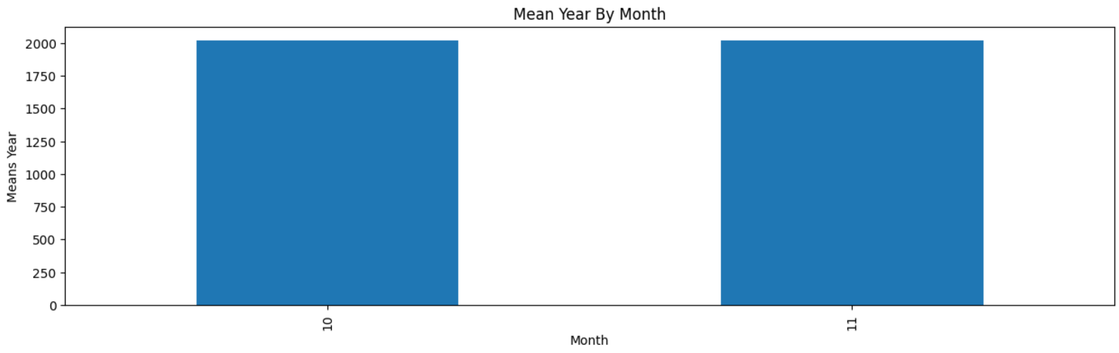
[168]:
```
<Axes: xlabel='Month', ylabel='Day of the Month'>
```



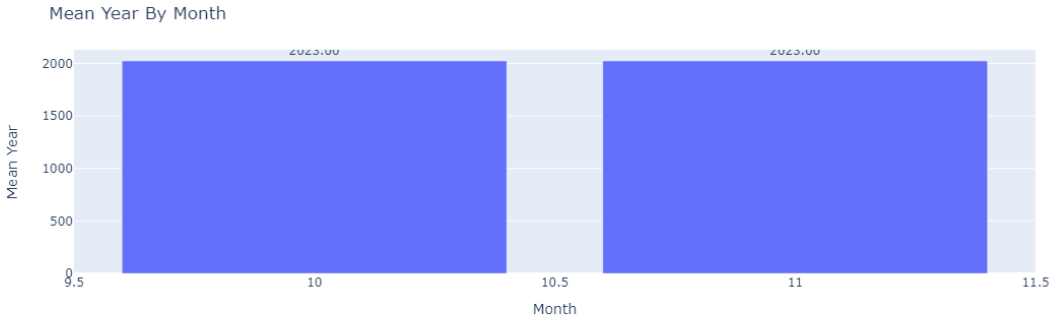
[169]:
```
# Create a violin plot to visualize the distribution and relationship between␣
↪'Month' and 'Day of the Month' using Plotly Express.
# Update the layout to set the plot title.
# Show the violin plot.

fig = px.violin(df, x='Month', y='Day of the Month')
fig.update_layout(title='Violin Plot of Day of the Month by Month')
fig.show()
```



[170]:
```
# Create a bar plot to display the mean year by month.
# Set the title, x-axis label, and y-axis label for the plot.
plt.figure(figsize=(15,4))
df.groupby('Month')['Year'].mean().plot(kind='bar')
# Show the bar plot.

plt.title('Mean Year By Month')
plt.xlabel('Month')
plt.ylabel('Means Year')
plt.show()
```

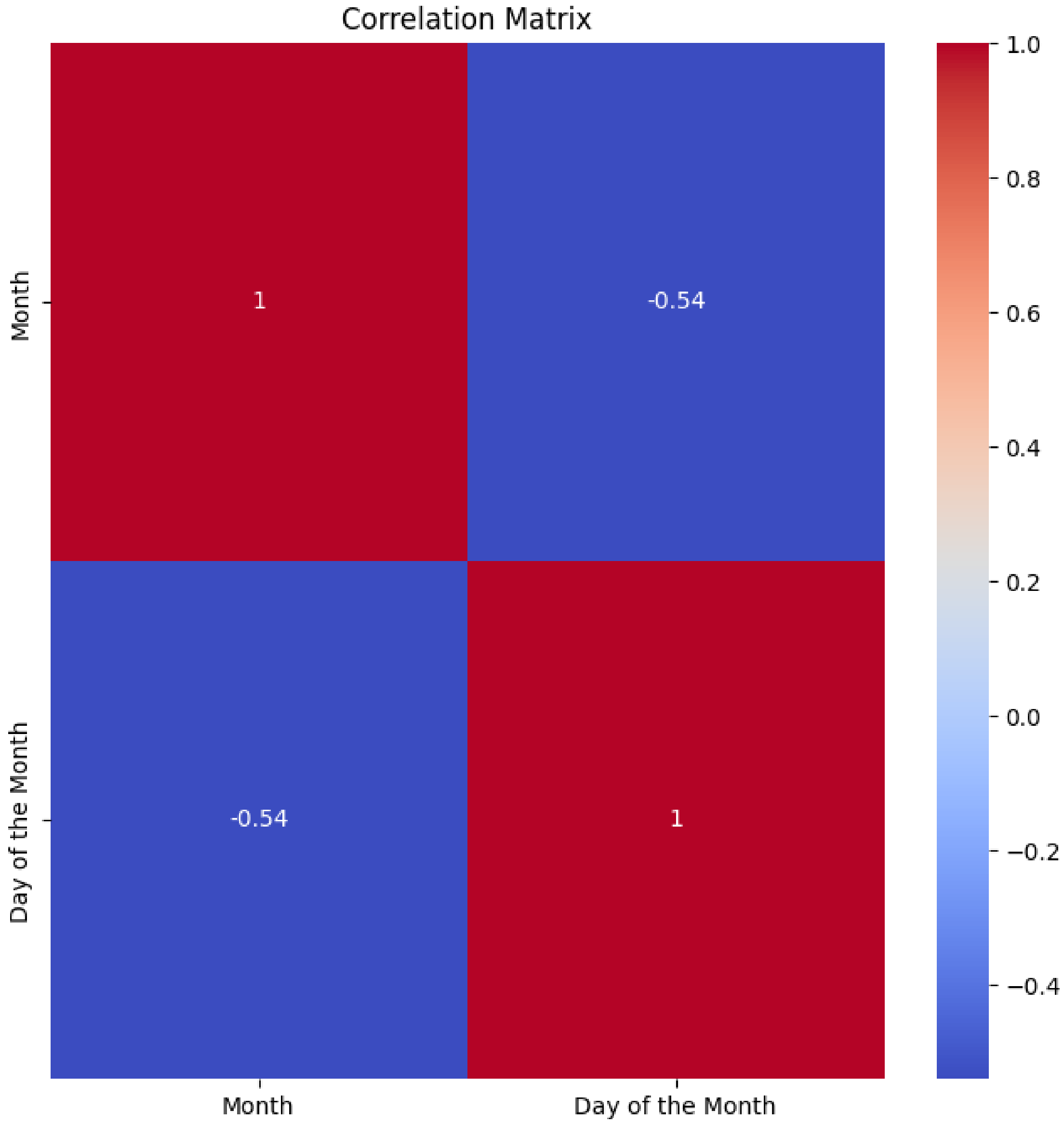


[171]:
```
# Create a bar plot to visualize the mean year by month using Plotly Express.
# Calculate the mean year by grouping the data by month and resetting the index.
mean_year_by_month = df.groupby('Month')['Year'].mean().reset_index()

# Create the bar plot using Plotly Express, specifying the x-axis, y-axis, and␣
 ↪text values.
fig = px.bar(mean_year_by_month, x='Month', y='Year', text='Year')
fig.update_traces(texttemplate='%{text:.2f}', textposition='outside')
fig.update_layout(title='Mean Year By Month', xaxis_title='Month',␣
 ↪yaxis_title='Mean Year')
# Show the bar plot.
fig.show()
```

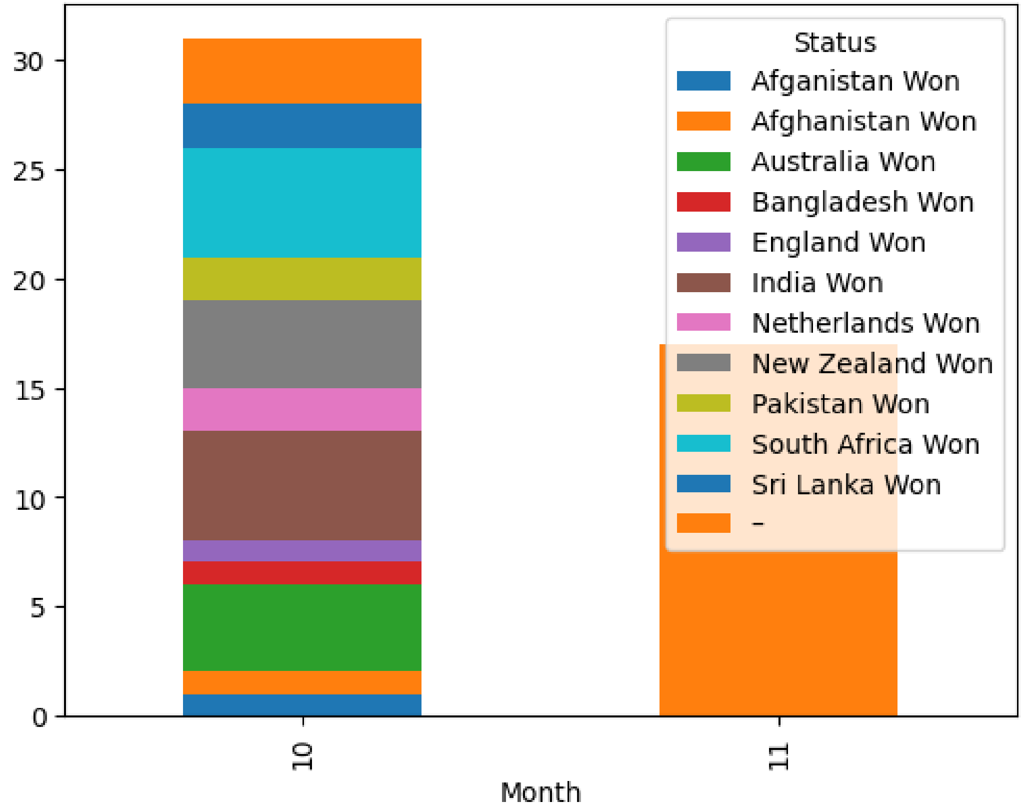


[172]:
```
# Calculate the correlation matrix between 'Month' and 'Day of the Month'.
correlation_matrix = df[['Month', 'Day of the Month']].corr()
# Create a heatmap to visualize the correlation matrix with annotations and a␣
 ↪coolwarm color map.
plt.figure(figsize=(8,8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
# Set the title for the heatmap.
```

```
plt.title("Correlation Matrix")
plt.show()
```

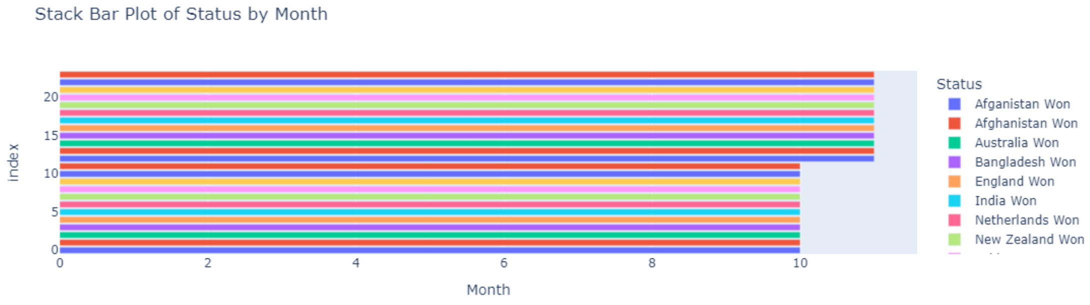


[173]:
```
# Create a cross-tabulation (cross-tab) between 'Month' and 'Status' columns␣
↪and store it in 'cross_tab'.
# Generate a stacked bar plot to visualize the cross-tabulation results.
# Display the stacked bar plot.

cross_tab = pd.crosstab(df['Month'], df['Status'])
cross_tab.plot(kind='bar', stacked=True)
plt.show()
```



[174]:

```
# Transform the stacked cross-tabulation into a long-format DataFrame with a␣
  ↪'Count' column.
# Create a figure and a bar plot using Plotly Express to display the 'Status'␣
  ↪by 'Month' with different colors.
# Show the stacked bar plot.

cross_tab_long = cross_tab.stack().reset_index(name='Count')
plt.figure(figsize=(12,8))
fig = px.bar(cross_tab_long, x='Month', color='Status', title='Stack Bar Plot␣
  ↪of Status by Month')
fig.show()
```



```
<Figure size 1200x800 with 0 Axes>
```

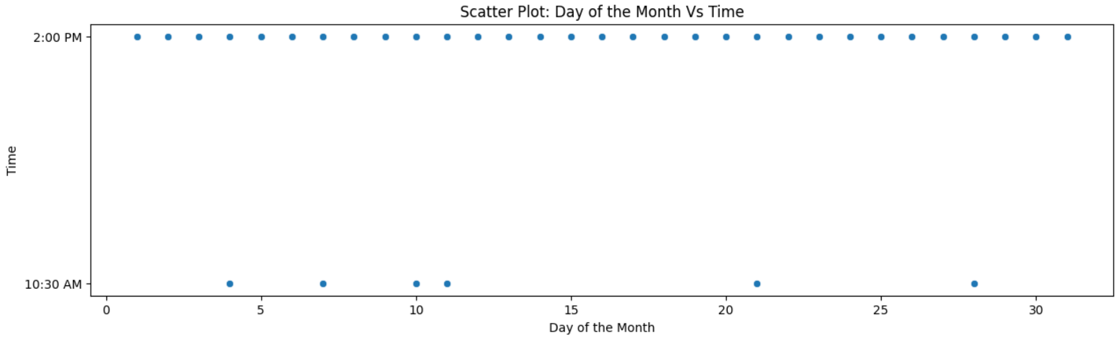
[175]:
```
cross_tab
```

[175]:
```
Status  Afganistan Won  Afghanistan Won  Australia Won  Bangladesh Won  \
Month
10                   1                1              4               1
11                   0                0              0               0

Status  England Won  India Won  Netherlands Won  New Zealand Won  \
Month
10                1          5                2                4
11                0          0                0                0

Status  Pakistan Won  South Africa Won  Sri Lanka Won   -
Month
10                 2                 5              2   3
11                 0                 0              0  17
```

[176]:
```
df.columns
```

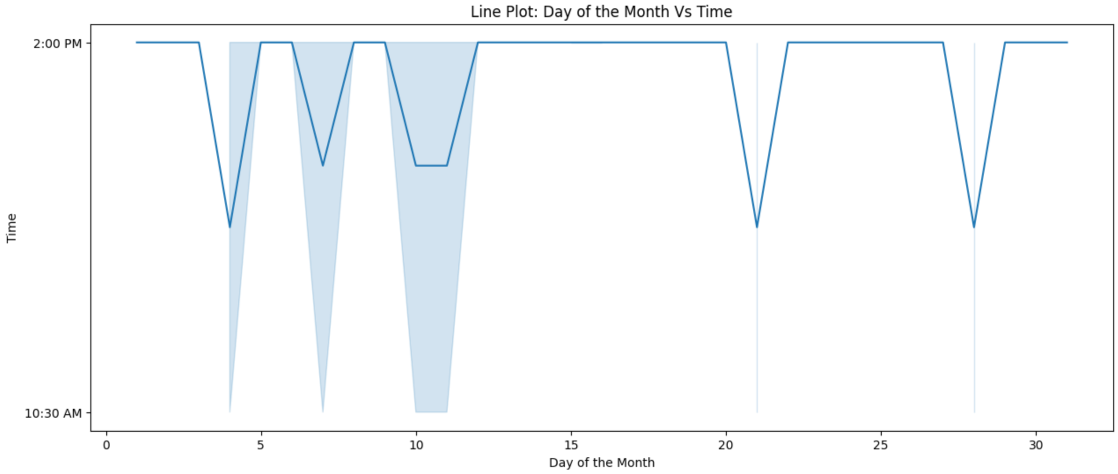
[176]:
```
Index(['Day & Date', 'Matches', 'Status', 'Time', 'Venue', 'Day of the Week',
       'Month', 'Day of the Month', 'Year'],
      dtype='object')
```

[177]:
```
# Create a scatter plot to visualize the relationship between 'Day of the␣
 ↪Month' and 'Time'.
# Set the title for the scatter plot.
# Show the scatter plot.


plt.figure(figsize=(15,4))
sns.scatterplot(data=df, x= 'Day of the Month', y='Time')
plt.title('Scatter Plot: Day of the Month Vs Time')
plt.show()
```



[178]:

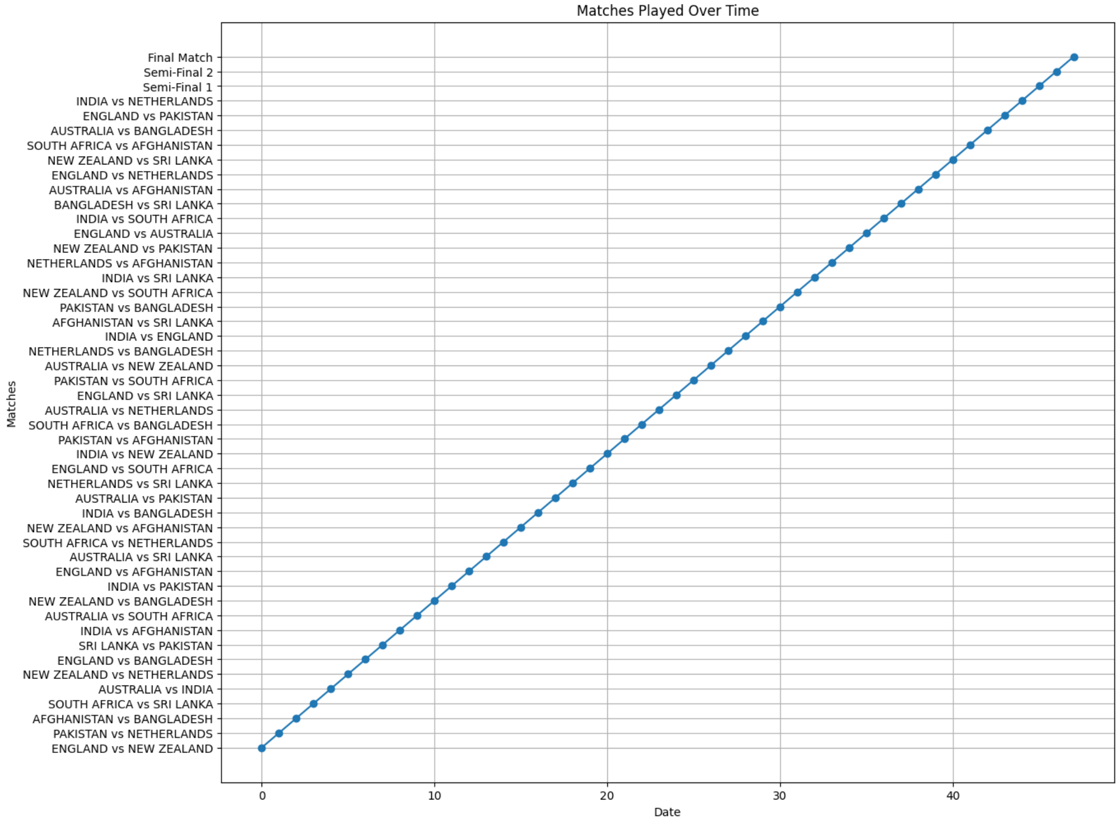
```
# Create a line plot to visualize the relationship between 'Day of the Month'␣
 ↪and 'Time'.
# Set the title for the line plot.
# Show the line plot.

plt.figure(figsize=(15,6))
sns.lineplot(data=df, x='Day of the Month', y='Time')
plt.title('Line Plot: Day of the Month Vs Time')
plt.show()
```



[179]:

```
# Create a line plot to visualize the number of matches played over time.
# Set the title, x-axis label, and y-axis label for the plot.
# Add grid lines to the plot.
# Show the line plot.
```

```
plt.figure(figsize=(14,12))
plt.plot(df.index, df['Matches'], marker='o', linestyle='-')
plt.title('Matches Played Over Time')
plt.xlabel('Date')
plt.ylabel('Matches')
plt.grid(True)
plt.show()
```



[ ]:
```
plt.figure(figsize=(16,14))
fig = go.Figure(data=go.Scatter(x=df.index, y=df['Matches'],␣
 ↪mode='lines+markers'))
fig.update_layout(
    title='Matches Played Over Time',
    xaxis_title='Date',
    yaxis_title='Matches',
    xaxis=dict(showline=True, showgrid=False),
    yaxis=dict(showline=True, showgrid=False),
)
fig.show()
```

[ ]:
```
df['Winning Team'] =df ['Status'].str.split(' vs ').str[0]
```

[ ]:

```
df['Winning Team'] = df ['Winning Team'].str.replace(' Won ', '')
```

[ ]:

```
df
```

[ ]:

```
winning_counts = df ['Winning Team'].value_counts()
```

[ ]:

```
winning_counts
```

[ ]:

```
matches_played = df['Matches'].str.split(' vs ', expand=True)
```

[ ]:

```
matches_played
```

[ ]:

```
df[['Team 1', 'Team 2']] = df ['Matches'].str.split(' vs ', expand=True)
```

[ ]:

```
df
```

[ ]:

```
df_filtered = df.iloc[:28]
```

[ ]:

```
team_matches_played = {}
```

[ ]:

```
for _, row in df_filtered.iterrows():

    team1 = row['Team 1']
    team2 = row ['Team 2']


    if team1 in team_matches_played:
        team_matches_played[team1] += 1
    else:
        team_matches_played[team1] = 1
    if team2 in team_matches_played:
        team_matches_played[team2] += 1
    else:
        team_matches_played[team2] = 1

summary_df = pd.DataFrame({
    'Team' : list(team_matches_played.keys()),
    'Matches Played' : list(team_matches_played.values())
})
```

[ ]:

```
summary_df = summary_df.sort_values(by='Matches Played', ascending=False)
```

[ ]:

```
team_stats = {
'Team': [],
'Matches Played': [],
'Matches Won': []
}
def clean_team_name(team_name):
        return team_name.strip().lower()
unique_teams = set()
for _, row in df_filtered.iterrows():
    winning_team = clean_team_name(row['Winning Team'])
    team1 = clean_team_name(row['Team 1'])
    team2 = clean_team_name(row['Team 2'])
    unique_teams.add(winning_team)
    unique_teams.add(team1)
    unique_teams.add(team2)
team_stats['Team'] = list(unique_teams)
team_stats['Matches Played'] = [0] * len(unique_teams)
team_stats['Matches Won'] = [0] * len(unique_teams)
for _, row in df_filtered.iterrows():
    winning_team = clean_team_name(row['Winning Team'])
    team1 = clean_team_name(row['Team 1'])
    team2 = clean_team_name(row['Team 2'])
    team_stats['Matches Played'][team_stats['Team'].index(team1)] += 1
    team_stats['Matches Played'][team_stats['Team'].index(team2)] += 1
    if winning_team != '-':
     team_stats['Matches Won'][team_stats['Team'].index(winning_team)] += 1
summary_df = pd.DataFrame(team_stats)
summary_df = summary_df.sort_values(by='Matches Won', ascending=False)
```

[ ]:

```
summary_df
```

Thank You !

- Connect Me! -

[ ]:

```
# Project By: Uvesh Ahmad
# Data Set Link: https://github.com/Uvesh-Ahmad
# Portfolio: https://uvesh-ahmad.github.io/uvesh.ah/
# Linkedin : https://www.linkedin.com/in/uvesh-ahmad-a2aa6816a/
```

[ ]:

```
```