In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries instal
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/d
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will lis

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that ge
# You can also write temporary files to /kaggle/temp/, but they won't be savec
```

```
/kaggle/input/general-election-2024-india/GE_2024_Results new.csv
```

## Follow me on linkedin - https://www.linkedin.com/in/sshankt/

In [2]:
```
df = pd.read_csv("/kaggle/input/general-election-2024-india/GE_2024_Results ne
```

In [3]:
```
df.head(10)
```

Out[3]:

| | State | Constituency | Candidate | Party | EVM Votes | Postal Votes | Total Votes | % o Vote |
|---|---|---|---|---|---|---|---|---|
| **0** | Andaman & Nicobar Islands | Andaman & Nicobar Islands | BISHNU PADA RAY | Bharatiya Janata Party | 102182 | 254 | 102436 | 50.5 |
| **1** | Andaman & Nicobar Islands | Andaman & Nicobar Islands | KULDEEP RAI SHARMA | Indian National Congress | 77829 | 211 | 78040 | 38.5 |
| **2** | Andaman & Nicobar Islands | Andaman & Nicobar Islands | MANOJ PAUL | Andaman Nicobar Democratic Congress | 8236 | 18 | 8254 | 04.0 |
| **3** | Andaman & Nicobar Islands | Andaman & Nicobar Islands | D AYYAPPAN | Communist Party of India (Marxist) | 6009 | 8 | 6017 | 2.9 |
| **4** | Andaman & Nicobar Islands | Andaman & Nicobar Islands | V.K. ABDUL AZIZ | Independent | 2195 | 8 | 2203 | 01.0 |
| **5** | Andaman & Nicobar Islands | Andaman & Nicobar Islands | K J B SELVARAJ | All India Anna Dravida Munnetra Kazhagam | 911 | 3 | 914 | 0.4 |
| **6** | Andaman & Nicobar Islands | Andaman & Nicobar Islands | DR ARUN KUMAR MALLIK | Bahujan Samaj Party | 714 | 5 | 719 | 0.3 |
| **7** | Andaman & Nicobar Islands | Andaman & Nicobar Islands | RINKU MALA MONDAL | Independent | 539 | 3 | 542 | 0.2 |
| **8** | Andaman & Nicobar Islands | Andaman & Nicobar Islands | K VENKAT RAM BABU | Independent | 506 | 1 | 507 | 0.2 |
| **9** | Andaman & Nicobar Islands | Andaman & Nicobar Islands | USHA KUMARI | Independent | 378 | 2 | 380 | 0.1 |

In [4]:
```
df.tail(10)
```

Out[4]:

| | State | Constituency | Candidate | Party | EVM Votes | Postal Votes | Total Votes | % Vot |
|---|---|---|---|---|---|---|---|---|
| **8892** | West Bengal | Uluberia | MOFIKUL ISLAM | All India Secular Front | 38971 | 20 | 38991 | 2 |
| **8893** | West Bengal | Uluberia | ARIT KARAK | Independent | 7287 | 5 | 7292 | 0. |
| **8894** | West Bengal | Uluberia | AMAL KUMAR DEYATI | Bharatiya Nyay-Adhikar Raksha Party | 5445 | 6 | 5451 | 0. |
| **8895** | West Bengal | Uluberia | RAMESH KHANRA | Independent | 5094 | 10 | 5104 | 0. |
| **8896** | West Bengal | Uluberia | BIMALESH KUMAR HELA | Bahujan Samaj Party | 4867 | 8 | 4875 | 0. |
| **8897** | West Bengal | Uluberia | SK. SAPIYAR ALI | Independent | 2924 | 0 | 2924 | 0. |
| **8898** | West Bengal | Uluberia | NIKHIL BERA | Socialist Unity Centre Of India (COMMUNIST) | 2095 | 12 | 2107 | 0. |
| **8899** | West Bengal | Uluberia | AMAL KUMAR BARMAN | Independent | 1997 | 3 | 2000 | 0. |
| **8900** | West Bengal | Uluberia | RAMPRASAD GHORAI | Indian Unity Centre | 1568 | 4 | 1572 | 0. |
| **8901** | West Bengal | Uluberia | NOTA | None of the Above | 11263 | 55 | 11318 | 0. |

In [5]:
```
df.shape
```

Out[5]:
```
(8902, 9)
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8902 entries, 0 to 8901
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   State         8902 non-null   object
 1   Constituency  8902 non-null   object
 2   Candidate     8902 non-null   object
 3   Party         8902 non-null   object
 4   EVM Votes     8902 non-null   object
 5   Postal Votes  8902 non-null   int64 
 6   Total Votes   8902 non-null   int64 
 7   % of Votes    8902 non-null   object
 8   Result        8902 non-null   object
dtypes: int64(2), object(7)
memory usage: 626.1+ KB
```

In [7]:
```
df.describe()
```

Out[7]:

| | Postal Votes | Total Votes |
|---|---|---|
| **count** | 8902.000000 | 8.902000e+03 |
| **mean** | 420.636711 | 7.249646e+04 |
| **std** | 1339.607914 | 1.798988e+05 |
| **min** | 0.000000 | 0.000000e+00 |
| **25%** | 3.000000 | 1.094250e+03 |
| **50%** | 9.000000 | 2.781000e+03 |
| **75%** | 49.000000 | 9.759500e+03 |
| **max** | 19827.000000 | 1.471885e+06 |

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
State           0
Constituency    0
Candidate       0
Party           0
EVM Votes       0
Postal Votes    0
Total Votes     0
% of Votes      0
Result          0
dtype: int64
```

In [9]:
```
df.duplicated().sum()
```

Out[9]:
```
0
```

In [10]:
```
df.columns
```

```
Out[10]: Index(['State', 'Constituency', 'Candidate', 'Party', 'EVM Votes',
               'Postal Votes', 'Total Votes', '% of Votes', 'Result'],
              dtype='object')
```

```
In [11]: df["% of Votes"] = (
             df["% of Votes"]
             .astype(str)                # convert to string
             .str.replace("%", "")       # remove %
             .str.strip()                # remove leading/trailing spaces
             .str.replace(",", "")       # remove commas if present
         )

         # Handle empty strings or non-numeric values
         df["% of Votes"] = pd.to_numeric(df["% of Votes"], errors="coerce")
```

```
In [12]: df["EVM Votes"] = pd.to_numeric(df["EVM Votes"], errors="coerce")
         df["Postal Votes"] = pd.to_numeric(df["Postal Votes"], errors="coerce")
         df["check_total"] = df["EVM Votes"]+ df["Postal Votes"]
```

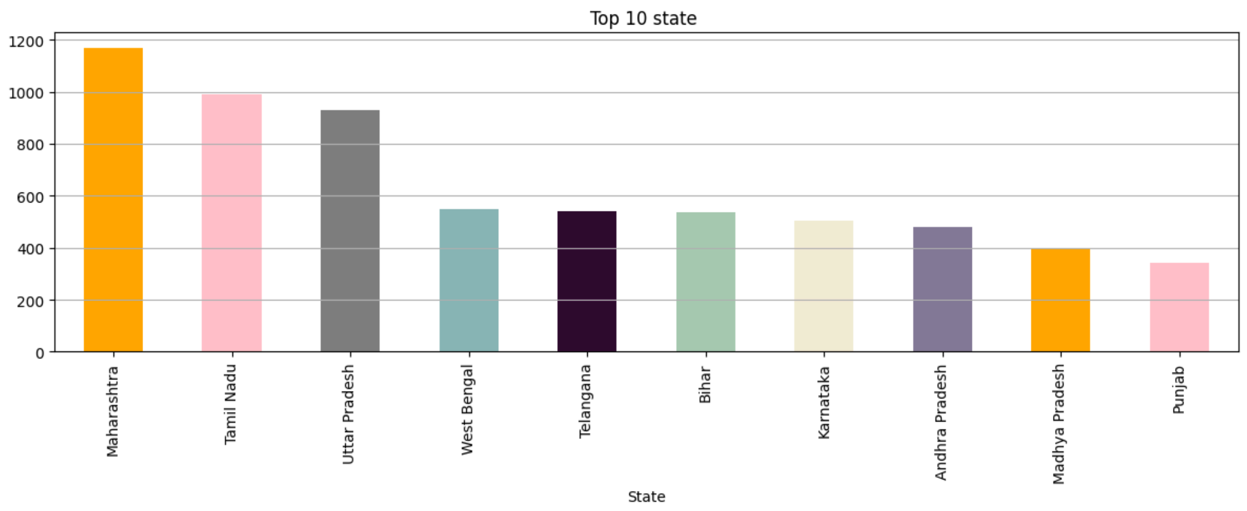
```
In [13]: df['Mismatch'] = df['Total Votes'] - df['check_total']
         print(df['Mismatch'].sum())

        0.0
```

```
In [14]: df["State"].value_counts().head(10)
```

```
Out[14]: State
         Maharashtra       1169
         Tamil Nadu         989
         Uttar Pradesh      931
         West Bengal        549
         Telangana          542
         Bihar              537
         Karnataka          502
         Andhra Pradesh     479
         Madhya Pradesh     398
         Punjab             341
         Name: count, dtype: int64
```

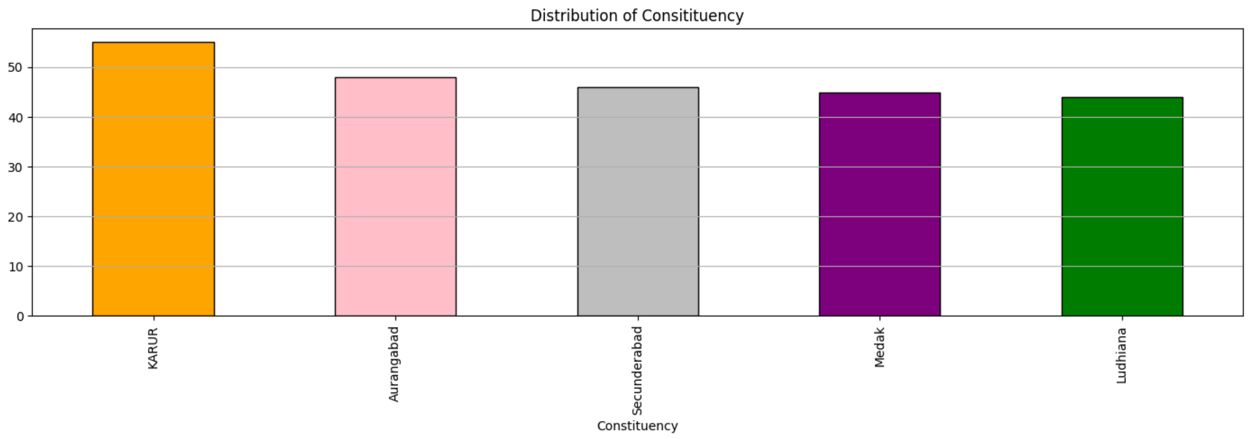
```
In [15]: plt.figure(figsize=(12,5))
         df["State"].value_counts().head(10).plot(kind="bar", color = ["orange", "pink"
         plt.title("Top 10 state ")
         plt.tight_layout()
         plt.grid(axis = "y")
         plt.show()
```



```
In [16]: df["Constituency"].value_counts().head()
```

```
Out[16]: Constituency
         KARUR            55
         Aurangabad       48
         Secunderabad     46
         Medak            45
         Ludhiana         44
         Name: count, dtype: int64
```

```
In [17]: plt.figure(figsize=(14,5))
         df["Constituency"].value_counts().head().plot(kind = "bar", color = ["orange",
         plt.title("Distribution of Consitituency")
         plt.tight_layout()
         plt.grid(axis = "y")
         plt.show()
```

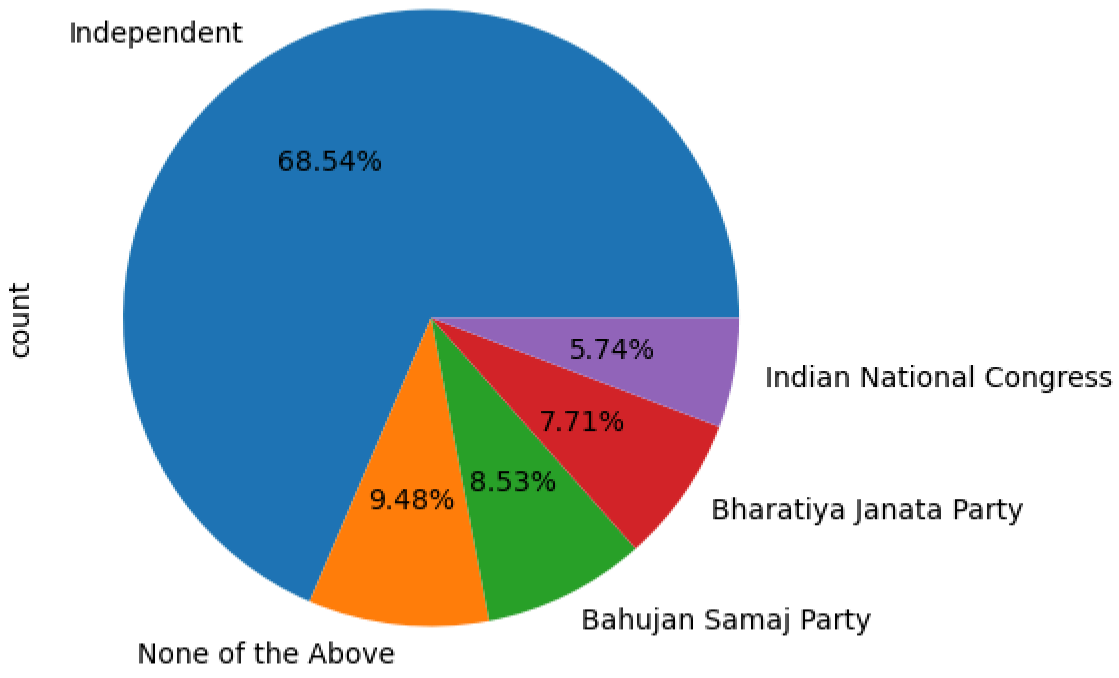


```
In [18]: df["Party"].value_counts().head()
```

```
Out[18]: Party
         Independent                 3920
         None of the Above            542
         Bahujan Samaj Party          488
         Bharatiya Janata Party       441
         Indian National Congress     328
         Name: count, dtype: int64
```

```
In [19]: plt.figure(figsize=(14,5))
         df["Party"].value_counts().head().plot(kind = "pie", autopct = "%1.2f%%")
```

```
Out[19]: <Axes: ylabel='count'>
```



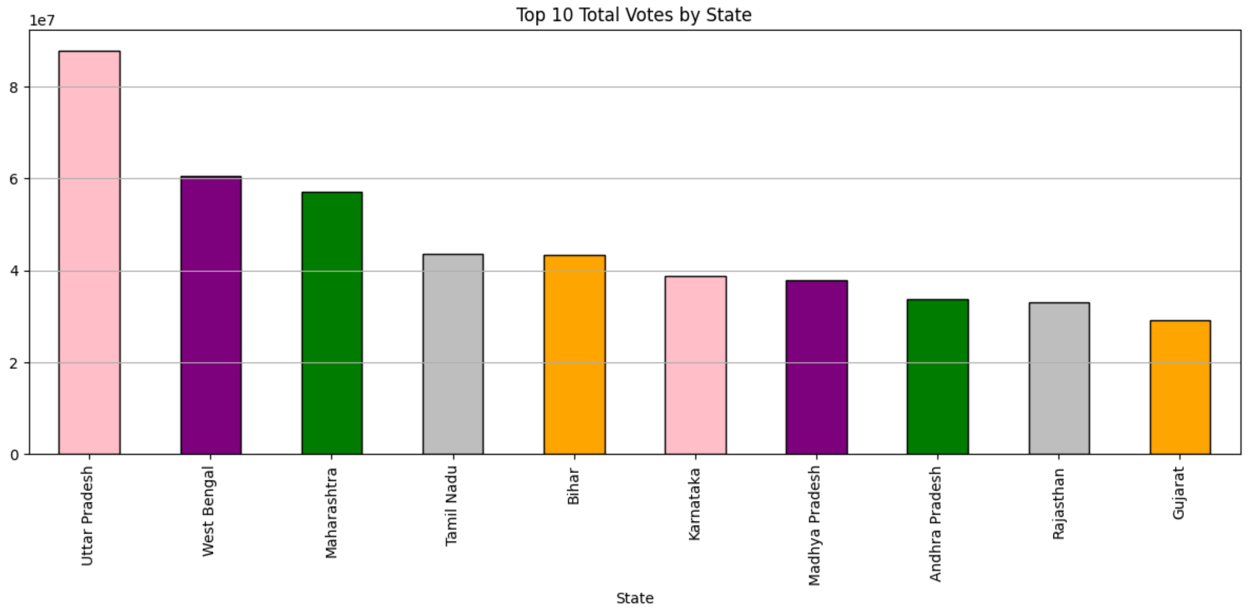
```
In [20]: df.columns
```

```
Out[20]: Index(['State', 'Constituency', 'Candidate', 'Party', 'EVM Votes',
                'Postal Votes', 'Total Votes', '% of Votes', 'Result', 'check_total',
                'Mismatch'],
               dtype='object')
```

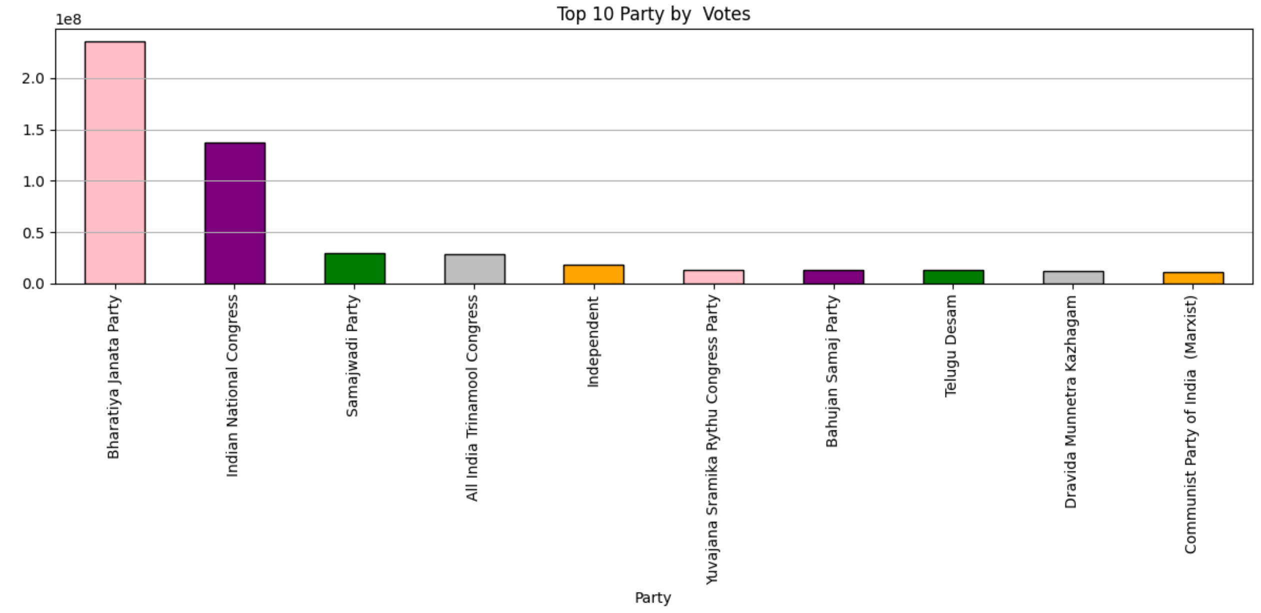
```
In [21]: states_votes = df.groupby("State")["Total Votes"].sum().sort_values(ascending=
         states_votes
```

```
Out[21]: State
         Uttar Pradesh     87911642
         West Bengal       60483687
         Maharashtra       57179133
         Tamil Nadu        43674048
         Bihar             43448949
         Karnataka         38793617
         Madhya Pradesh    37940251
         Andhra Pradesh    33729342
         Rajasthan         33164877
         Gujarat           29115599
         Name: Total Votes, dtype: int64
```

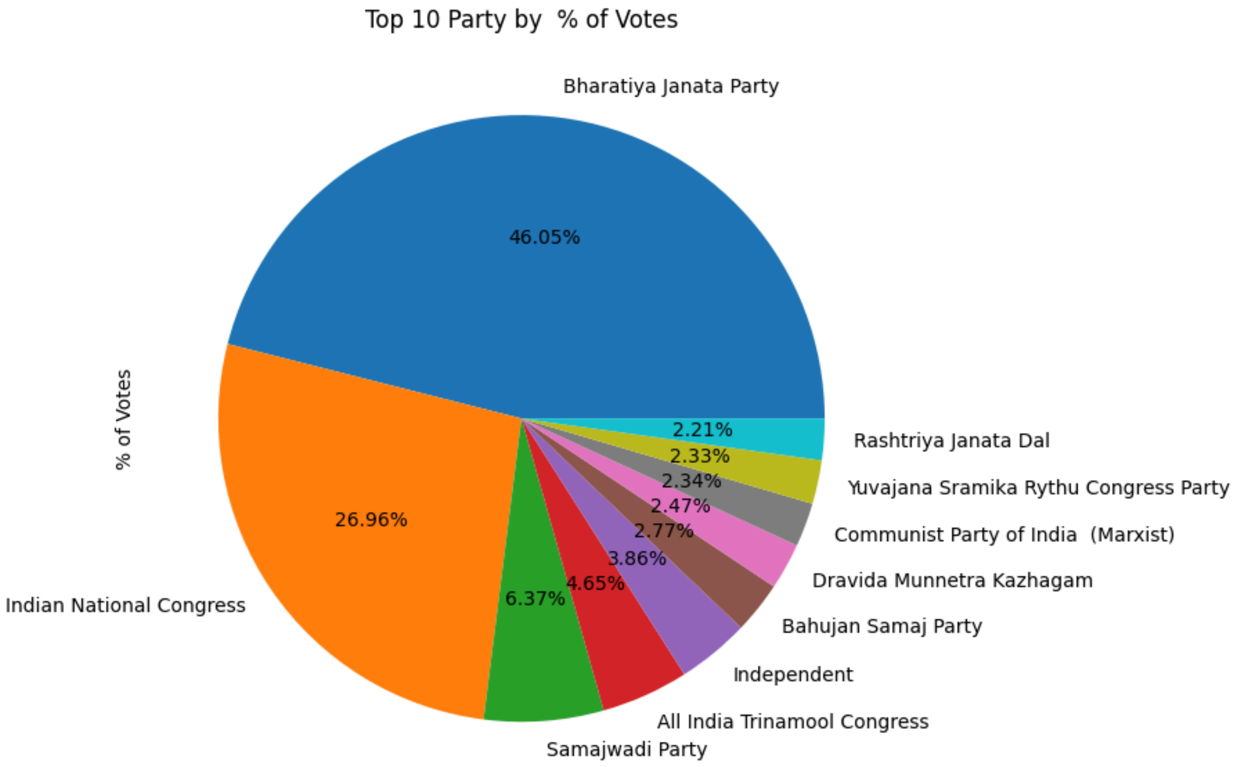
```
In [22]: plt.figure(figsize = (12,6))
         states_votes = df.groupby("State")["Total Votes"].sum().sort_values(ascending=
         states_votes.plot(kind = "bar", color = ["Pink","Purple","green","silver","ora
         plt.title("Top 10 Total Votes by State")
         plt.grid(axis = "y")
         plt.tight_layout()
         plt.show()
```



```
In [23]: plt.figure(figsize = (12,6))
         states_votes = df.groupby("Party")["Total Votes"].sum().sort_values(ascending=
         states_votes.plot(kind = "bar", color = ["Pink","Purple","green","silver","ora
         plt.title("Top 10 Party by  Votes")
         plt.grid(axis = "y")
         plt.tight_layout()
         plt.show()
```



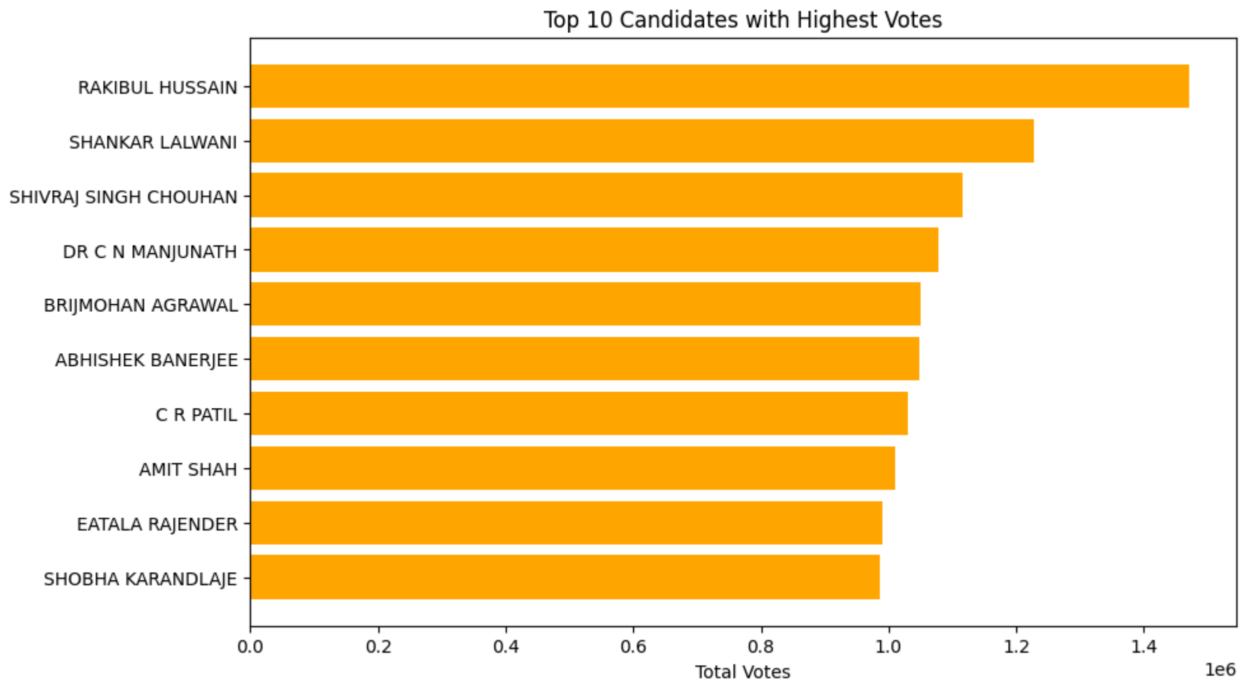
In [24]:
```
plt.figure(figsize = (12,6))
states_votes = df.groupby("Party")["% of Votes"].sum().sort_values(ascending=F
states_votes.plot(kind = "pie", color = ["Pink","Purple","green","silver","ora
plt.title("Top 10 Party by  % of Votes")
plt.tight_layout()
plt.show()
```



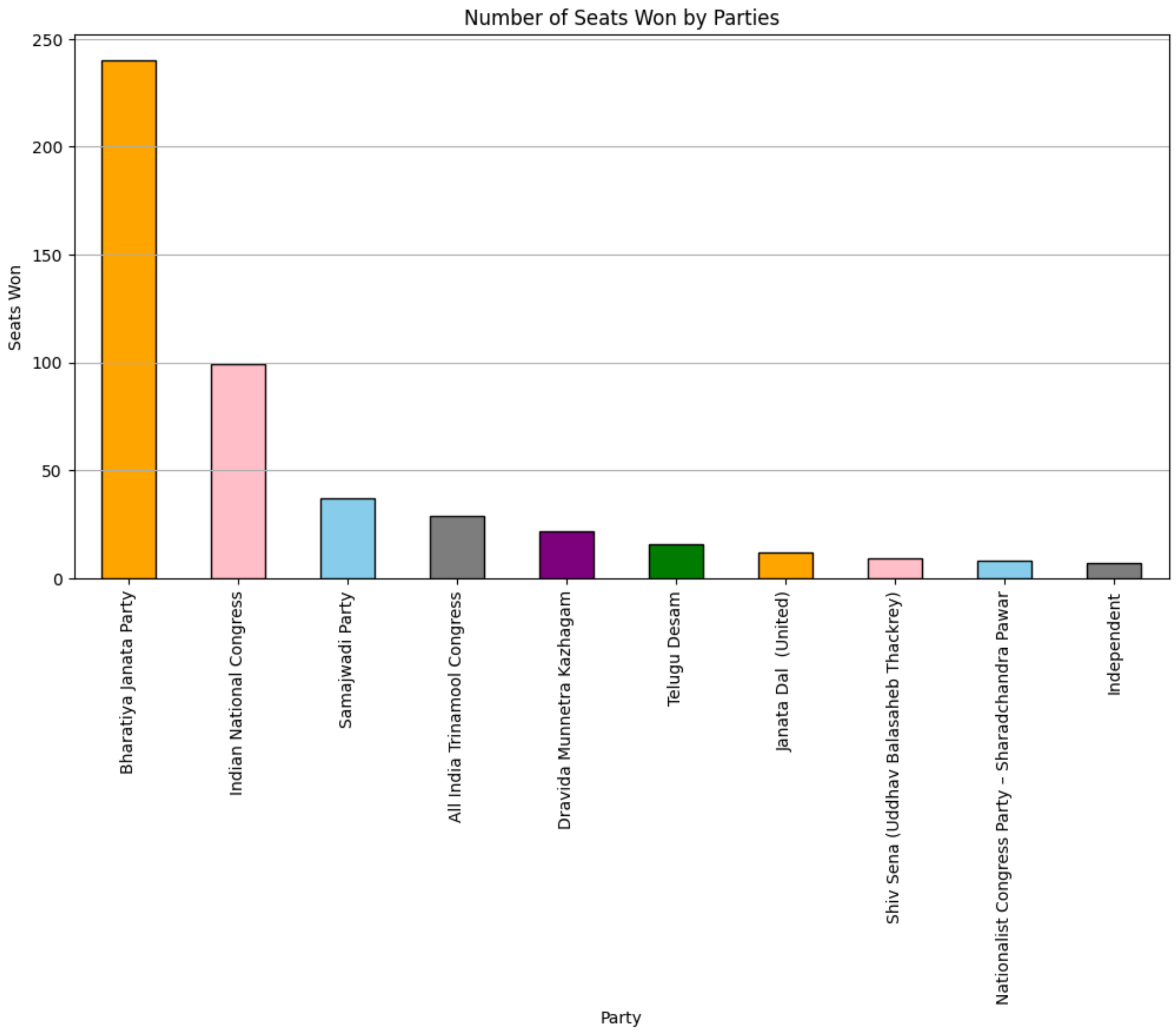
In [25]:
```
df.dtypes
```

```
Out[25]: State               object
         Constituency        object
         Candidate           object
         Party               object
         EVM Votes          float64
         Postal Votes         int64
         Total Votes          int64
         % of Votes         float64
         Result              object
         check_total        float64
         Mismatch           float64
         dtype: object
```

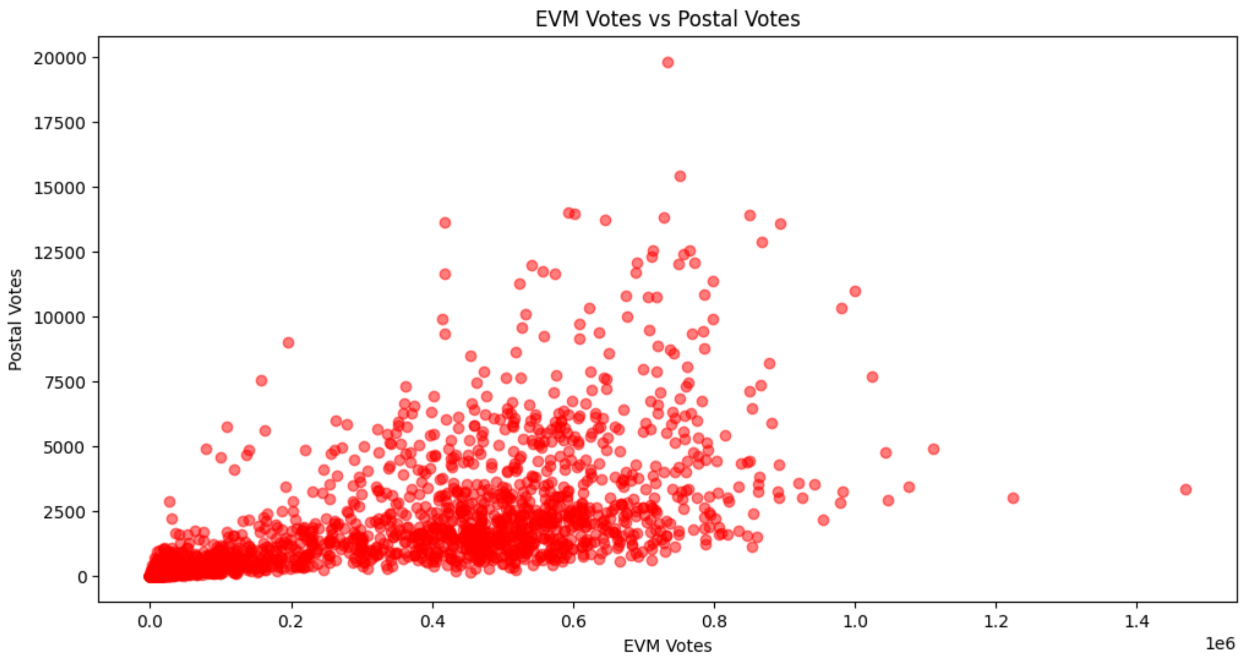
```
In [26]: top_candidates = df.nlargest(10, 'Total Votes')
         plt.figure(figsize=(10,6))
         plt.barh(top_candidates['Candidate'], top_candidates['Total Votes'], color='or
         plt.title("Top 10 Candidates with Highest Votes")
         plt.xlabel("Total Votes")
         plt.gca().invert_yaxis()
         plt.show()
```



```
In [27]: winners = df[df['Result'] == 'Won']
         party_winners = winners['Party'].value_counts().head(10)
         party_winners.plot(kind='bar', figsize=(12,6), color=["orange","pink", "skyblu
         plt.title("Number of Seats Won by Parties")
         plt.ylabel("Seats Won")
         plt.grid(axis = "y")
         plt.show()
```



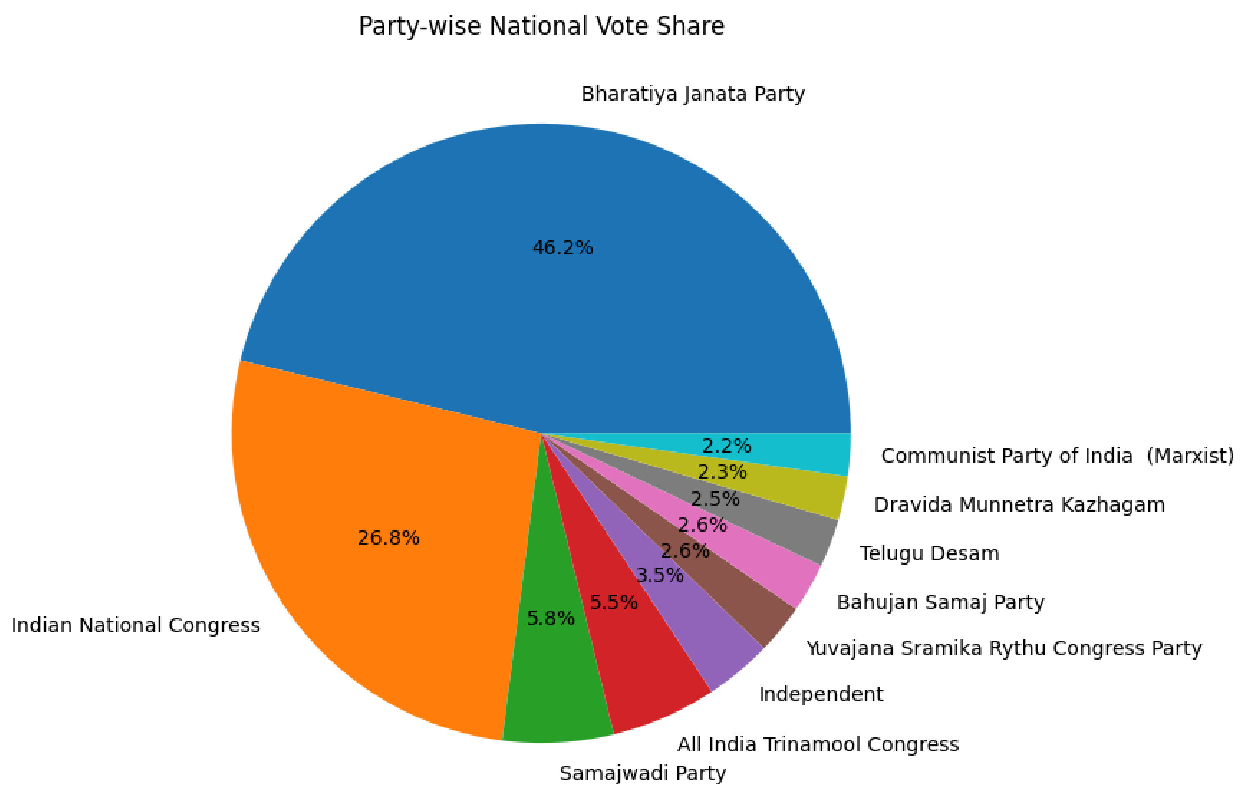
In [28]:
```
plt.figure(figsize=(12,6))
plt.scatter(df['EVM Votes'], df['Postal Votes'], alpha=0.5, color='red')
plt.title("EVM Votes vs Postal Votes")
plt.xlabel("EVM Votes")
plt.ylabel("Postal Votes")
plt.show()
```



```
In [29]: df.columns
```

```
Out[29]: Index(['State', 'Constituency', 'Candidate', 'Party', 'EVM Votes',
                'Postal Votes', 'Total Votes', '% of Votes', 'Result', 'check_total',
                'Mismatch'],
               dtype='object')
```

```
In [30]: party_share = df.groupby('Party')['Total Votes'].sum()
         party_share = (party_share / party_share.sum()) * 100
         party_share.sort_values(ascending=False).head(10).plot(kind='pie', autopct='%1
         plt.title("Party-wise National Vote Share")
         plt.ylabel("")
         plt.show()
```



## Follow me on linkedin - https://www.linkedin.com/in/sshankt/

## Analyse by - Shashank Tiwari

```
In [ ]:
```