IMPORT OF MODULES FOR ANALYSIS

In [36]:

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import geopandas as gpd
import plotly.express as px
```

In [57]:

```

ts = pd.read_csv("D:/VISUALISATION_WORKSHOP/tselection/DATASET/Telengana_Assembly_election_2023.csv")
df = pd.read_csv("D:/VISUALISATION_WORKSHOP/tselection/DATASET/Telengana_Assembly_election_2023.csv")
#Importing the spatial data
gdf = gpd.read_file("D:/VISUALISATION_WORKSHOP/tselection/DATASET/telangana_ac.geojson.txt")
#reference file
no_gdf = pd.read_csv("D:/VISUALISATION_WORKSHOP/tselection/DATASET/Telangana_Constituency_No.csv")
```

In [40]:

```
ts.head()
```

Out[40]:

	Unnamed: 0	S.N.	Candidate	Party	EVM Votes	Postal Votes	Total Votes	% of Votes	Constituency
0	0	1	GUVVALA BALARAJU	Bharat Rashtra Samithi	65661	350	66011	33.74	82 - Achampet
1	1	2	CHIKKUDU VAMSHI KRISHNA	Indian National Congress	113761	1576	115337	58.96	82 - Achampet
2	2	3	DEVANI SATHYANARAYANA ALIAS DEVANI SATISH MADIGA	Bharatiya Janata Party	4201	66	4267	2.18	82 - Achampet
3	3	4	MOTHUKURI NAGARJUN	Bahujan Samaj Party	1174	15	1189	0.61	82 - Achampet
4	4	5	KUNDA MALLIKARJUN	Yuga Thulasi Party	239	0	239	0.12	82 - Achampet

In [41]:

```
ts.tail()
```

Out[41]:

	Unnamed: 0	S.N.	Candidate	Party	EVM Votes	Postal Votes	Total Votes	% of Votes	Constituency
2404	2404	19	RAMULU HUGGELLY	Independent	665	3	668	0.32	38 - Zahirabad
2405	2405	20	SHIVASHANKAR	Independent	137	0	137	0.07	38 - Zahirabad
2406	2406	21	SRINIVAS	Independent	85	0	85	0.04	38 - Zahirabad
2407	2407	22	HEMANAND TABLA	Independent	82	0	82	0.04	38 - Zahirabad
2408	2408	23	NOTA	None of the Above	611	2	613	0.29	38 - Zahirabad

In [58]:

```
#Extracting the number from contituency column
df['Constituency_no'] = df['Constituency'].str.extract(r'(\d+)')

#Extracting the text from contituency column
df['Constituency_Alphabets'] = df['Constituency'].str.extract(r'\d+ - (.+)$')

#dropping columns
df = df.drop(['Unnamed: 0','S.N.','Constituency'], axis = 1)
#Renaming columns
df.rename(columns={'Constituency_Alphabets': 'Constituency'}, inplace=True)
#head data
print("Sample of data")
df.head(3)
```

```
Sample of data
```

Out[58]:

	Candidate	Party	EVM Votes	Postal Votes	Total Votes	% of Votes	Constituency_no	Constituency
0	GUVVALA BALARAJU	Bharat Rashtra Samithi	65661	350	66011	33.74	82	Achampet
1	CHIKKUDU VAMSHI KRISHNA	Indian National Congress	113761	1576	115337	58.96	82	Achampet
2	DEVANI SATHYANARAYANA ALIAS DEVANI SATISH MADIGA	Bharatiya Janata Party	4201	66	4267	2.18	82	Achampet

In [59]:

```
gdf_merged = pd.merge(gdf, no_gdf, on = 'id', suffixes = ('','_no_gdf'))

#Constituency no column as Integer
df['Constituency_no'] = df['Constituency_no'].astype(int)

spatial_df_d = gdf_merged.loc[:,['id','assembly','constituency_no','geometry']]


spatial_df_total = pd.merge(spatial_df_d, df, left_on ='constituency_no', right_on = 'Constituency_no')

spatial_df = spatial_df_total.loc[:,['id','constituency_no','Constituency','geometry']]

spatial_df = spatial_df.drop_duplicates()
print("Sample of Spatial data")
spatial_df.head()
```

```
Sample of Spatial data
```

Out[59]:

	id	constituency_no	Constituency	geometry
0	1	3	Bellampalli	MULTIPOLYGON (((79.48229 19.19617, 79.48282 19...
14	2	8	Boath	MULTIPOLYGON (((78.34869 19.88415, 78.34989 19...
25	3	2	Chennur	MULTIPOLYGON (((79.78546 19.05654, 79.78529 19...
40	4	6	Khanapur	MULTIPOLYGON (((79.01319 19.21401, 79.01499 19...
52	5	4	Mancherial	MULTIPOLYGON (((79.15034 19.09220, 79.15041 19...

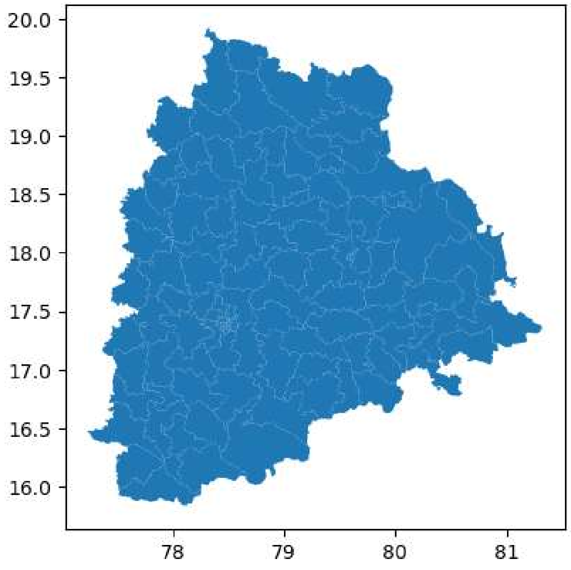
TELANGANA MAP WITH GEOPANDAS

In [60]:

```
print('Map of Telangana State with Constituency boundaries')

spatial_df.plot()
```

```
Map of Telangana State with Constituency boundaries
```

Out[60]: `<Axes: >`



In [ ]:

```

```

In [3]:
```
#FINDING THE DESCRIBE
ts.describe()
```

Out[3]:

	Unnamed: 0	S.N.	EVM Votes	Postal Votes	Total Votes	% of Votes
count	2409.000000	2409.000000	2409.000000	2409.000000	2409.000000	2409.000000
mean	1204.000000	12.148194	9646.899543	83.218348	9730.117891	4.939797
std	695.562722	8.479849	25608.097526	282.182086	25833.309113	12.762166
min	0.000000	1.000000	17.000000	0.000000	17.000000	0.010000
25%	602.000000	6.000000	151.000000	0.000000	152.000000	0.080000
50%	1204.000000	11.000000	393.000000	1.000000	394.000000	0.200000
75%	1806.000000	17.000000	1388.000000	7.000000	1394.000000	0.730000
max	2408.000000	49.000000	187327.000000	4501.000000	187999.000000	64.890000

In [6]:
```
ts.dtypes
```

Out[6]:
```
Unnamed: 0        int64
S.N.              int64
Candidate        object
Party            object
EVM Votes         int64
Postal Votes      int64
Total Votes       int64
% of Votes      float64
Constituency     object
dtype: object
```

In [10]:
```
#FINDING THE MEAN
mean_values = ts.mean(numeric_only=True)
print("Mean:")
print(mean_values)

#here showing the mean from evm votes , postal votes , total votes and percentage of votes.
```

```
Mean:
Unnamed: 0      1204.000000
S.N.              12.148194
EVM Votes       9646.899543
Postal Votes      83.218348
Total Votes     9730.117891
% of Votes         4.939797
dtype: float64
```

In [13]:
```
#finding the median
median_values = ts.median(numeric_only=True)
print("\nMedian:")
print(median_values)

#here showing the median from evm votes , postal votes , total votes and percentage of votes.
```

```

Median:
Unnamed: 0      1204.0
S.N.              11.0
EVM Votes        393.0
Postal Votes       1.0
Total Votes      394.0
% of Votes         0.2
dtype: float64
```

In [15]:

```
#finding the mode
mode_values = ts.mode().iloc[0]  # Using iloc[0] to handle multiple modes in a column
print("\nMode:")
print(mode_values)

#here, it is showing the mode (center) value from dataset
```

```
Mode:
Unnamed: 0                           0
S.N.                               1.0
Candidate                         NOTA
Party                      Independent
EVM Votes                        128.0
Postal Votes                       0.0
Total Votes                      149.0
% of Votes                        0.04
Constituency    49 - Lal Bahadur Nagar
Name: 0, dtype: object
```

In [17]:

```
#finding the
# unimodal = means single mode value
# bimodal = two modes from the dataset
# multimodal = two or more modes from dataset

mode_counts = ts.apply(lambda x: x.value_counts().iloc[0])

# Identify if the distribution is unimodal, bimodal, or multimodal
num_modes = mode_counts.value_counts().index.size
if num_modes == 1:
    print("Unimodal distribution")
elif num_modes == 2:
    print("Bimodal distribution")
else:
    print("Multimodal distribution")

# Find the value that repeats the maximum number of times
max_repeats = mode_counts.max()
max_repeats_value = mode_counts[mode_counts == max_repeats].index[0]

print("Value that repeats maximum times:", max_repeats_value)
```

```
Multimodal distribution
Value that repeats maximum times: Party
```

In [19]:

```
# Calculate the variance for numerical columns
variance_values = ts.var(numeric_only=True)

print("Variance:")
print(variance_values)
```

```
Variance:
Unnamed: 0      4.838075e+05
S.N.            7.190785e+01
EVM Votes       6.557747e+08
Postal Votes    7.962673e+04
Total Votes     6.673599e+08
% of Votes      1.628729e+02
dtype: float64
```

In [20]:

```
# Find the maximum value for numerical columns
max_values = ts.max()

# Find the minimum value for numerical columns
min_values = ts.min()

print("Maximum values:")
print(max_values)
print("\nMinimum values:")
print(min_values)
```

```
Maximum values:
Unnamed: 0                        2408
S.N.                                49
Candidate                ZEENATH BEGUM
Party                 Yuva Taram Party
EVM Votes                       187327
Postal Votes                      4501
Total Votes                     187999
% of Votes                       64.89
Constituency    99 - Ghanpur (Station)
dtype: object

Minimum values:
Unnamed: 0                     0
S.N.                           1
Candidate       A ANJANEYA CHARY
Party               Aabaad Party
EVM Votes                     17
Postal Votes                   0
Total Votes                   17
% of Votes                  0.01
Constituency          1 - Sirpur
dtype: object
```

In [24]:

```
# Calculate the skewness for numerical columns
skewness_values = ts.skew(numeric_only=True)

print("Skewness:")
print(skewness_values)
```

```
Skewness:
Unnamed: 0      0.000000
S.N.            1.011864
EVM Votes       3.030310
Postal Votes    5.971769
Total Votes     3.029969
% of Votes      2.773123
dtype: float64
```

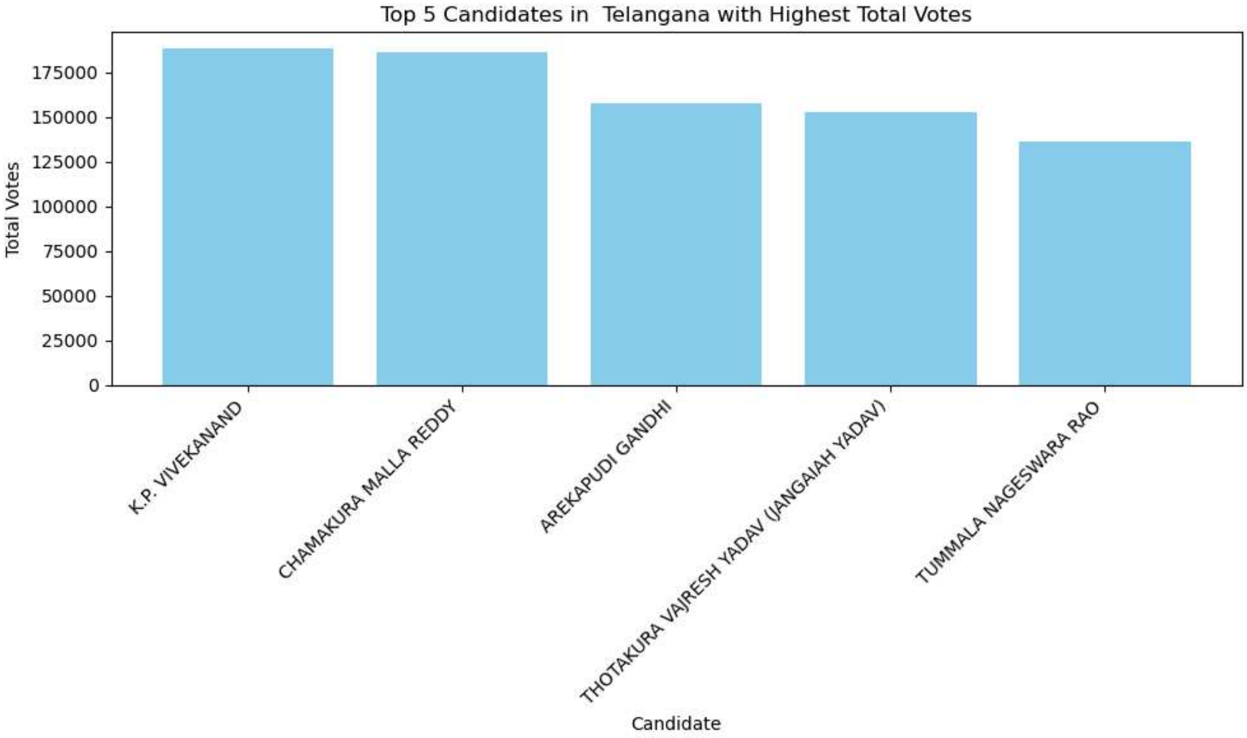
In [25]:

```
# Classify skewness
negative_skew = skewness_values[skewness_values < 0].index.tolist()
positive_skew = skewness_values[skewness_values > 0].index.tolist()
normal_skew = skewness_values[skewness_values.abs() < 0.5].index.tolist()

print("Negative Skewness:")
print(negative_skew)
print("\nPositive Skewness:")
print(positive_skew)
print("\nApproximately Symmetric (Skewness close to 0):")
print(normal_skew)
```

```
Negative Skewness:
[]

Positive Skewness:
['S.N.', 'EVM Votes', 'Postal Votes', 'Total Votes', '% of Votes']

Approximately Symmetric (Skewness close to 0):
['Unnamed: 0']
```

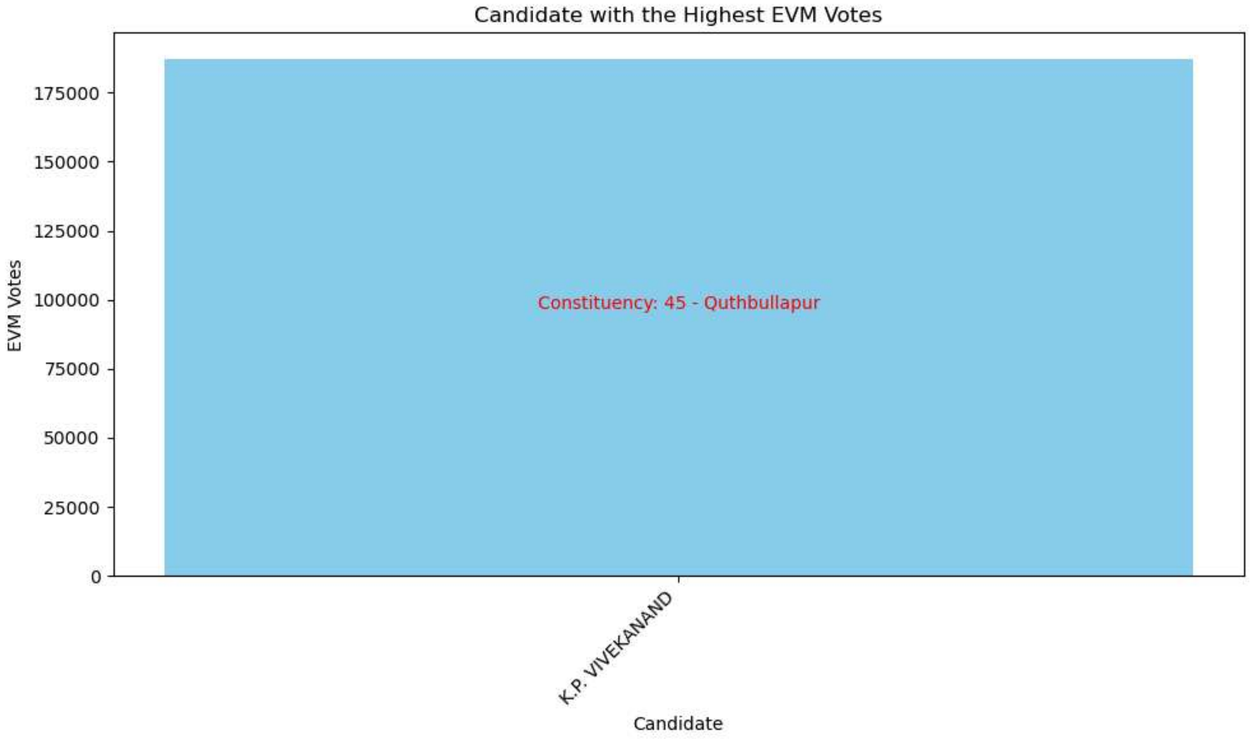
In [30]:

```
# Sort the DataFrame by total votes in descending order
sorted_df = ts.sort_values(by='Total Votes', ascending=False)

# Select the top 5 candidates
top_5 = sorted_df.head(5)

# Create a bar plot
plt.figure(figsize=(10, 6))
plt.bar(top_5['Candidate'], top_5['Total Votes'], color='skyblue')
plt.xlabel('Candidate')
plt.ylabel('Total Votes')
plt.title('Top 5 Candidates in  Telangana with Highest Total Votes')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()

# Display the plot
plt.show()
```



In [33]:

```
# Find the candidate with the highest number of EVM votes
candidate_highest_evm_votes = ts.loc[ts['EVM Votes'].idxmax()]

# Get the name and constituency of the candidate
candidate_name = candidate_highest_evm_votes['Candidate']
constituency = candidate_highest_evm_votes['Constituency']
evm_votes = candidate_highest_evm_votes['EVM Votes']

# Create a bar plot
plt.figure(figsize=(10, 6))
plt.bar(candidate_name, evm_votes, color='skyblue')
plt.xlabel('Candidate')
plt.ylabel('EVM Votes')
plt.title('Candidate with the Highest EVM Votes')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()

# Annotate the constituency
plt.annotate(f'Constituency: {constituency}', xy=(0.5, 0.5), xycoords='axes fraction', ha='center', va='center', fon

# Display the plot
plt.show()
```



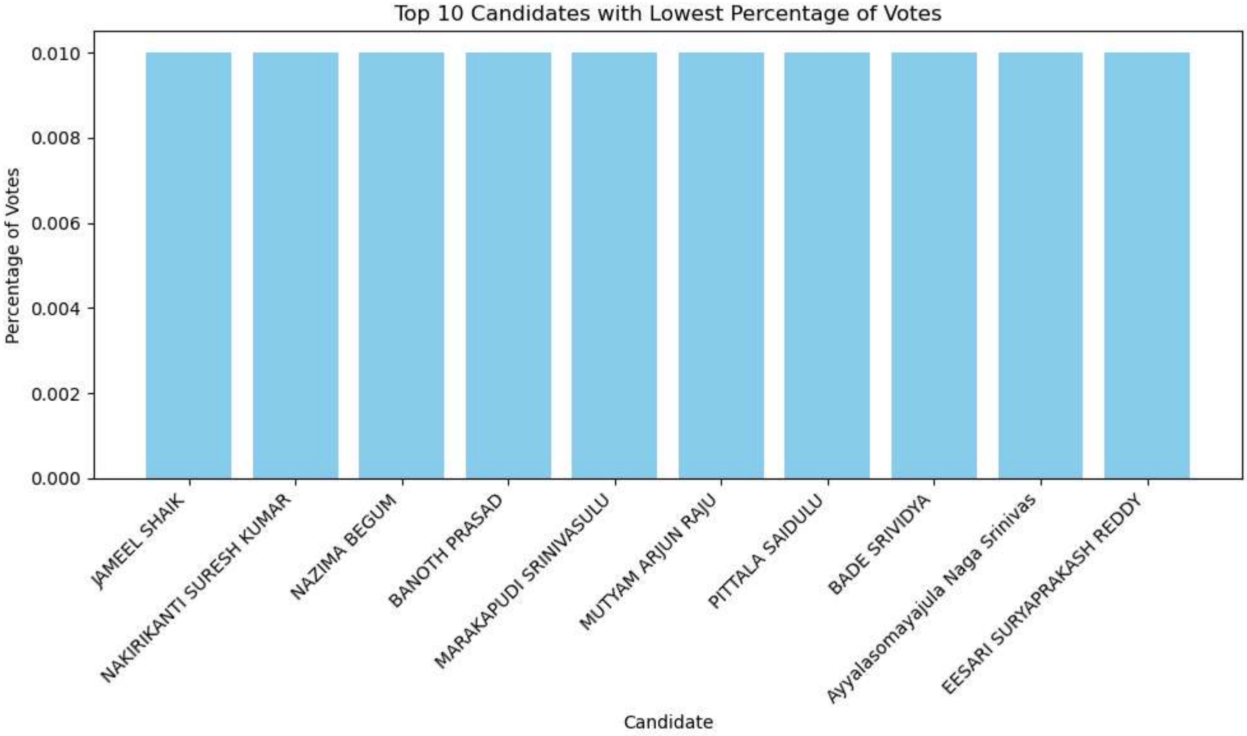
In [35]:

```
# Print the column names
print(ts.columns)
```

```
Index(['Unnamed: 0', 'S.N.', 'Candidate', 'Party', 'EVM Votes', 'Postal Votes',
       'Total Votes', '% of Votes', 'Constituency'],
      dtype='object')
```

In [36]:

```
# Sort the DataFrame by percentage of votes in ascending order and select the top 10
top_10_lowest_percentage = ts.nsmallest(10, '% of Votes')

# Create a bar plot
plt.figure(figsize=(10, 6))
plt.bar(top_10_lowest_percentage['Candidate'], top_10_lowest_percentage['% of Votes'], color='skyblue')
plt.xlabel('Candidate')
plt.ylabel('Percentage of Votes')
plt.title('Top 10 Candidates with Lowest Percentage of Votes')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()

# Display the plot
plt.show()
```



In [40]:

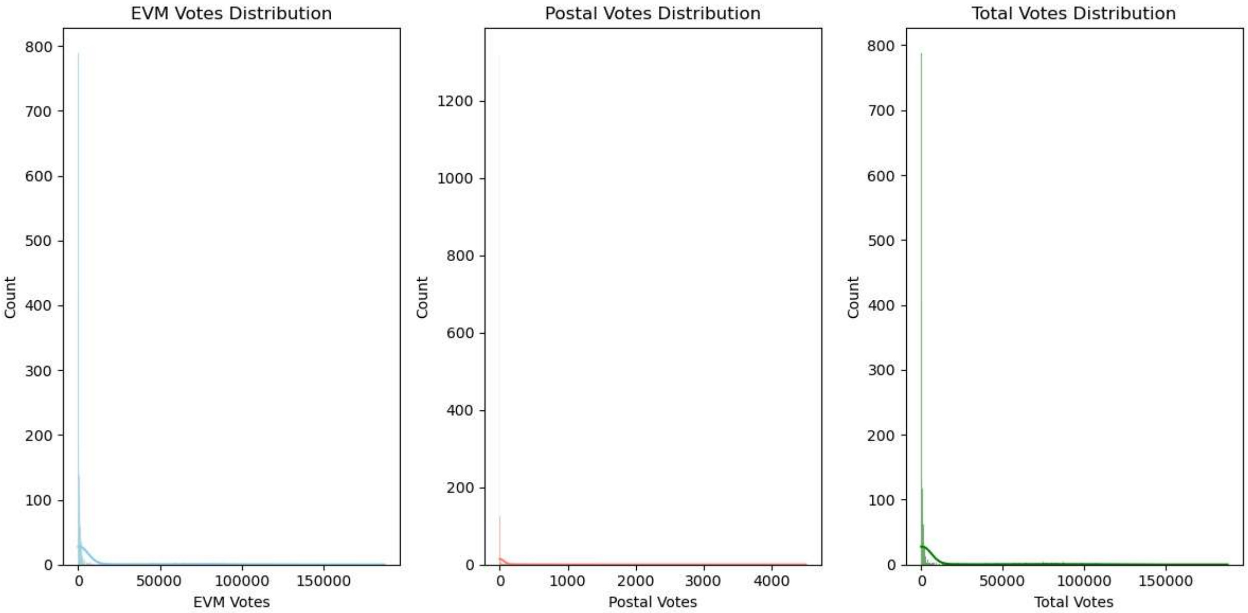
```
# Sort the DataFrame by percentage of votes in ascending order and select the top 10
top_10_lowest_percentage = ts.nsmallest(10, '% of Votes')

# Display the candidate names and their corresponding constituency names with the lowest percentage of votes
candidate_and_constituency_names = top_10_lowest_percentage[['Candidate', 'Constituency']]
print("Candidate names and their corresponding constituency names with the lowest percentage of votes:")
print(candidate_and_constituency_names.to_string(index=False))
```

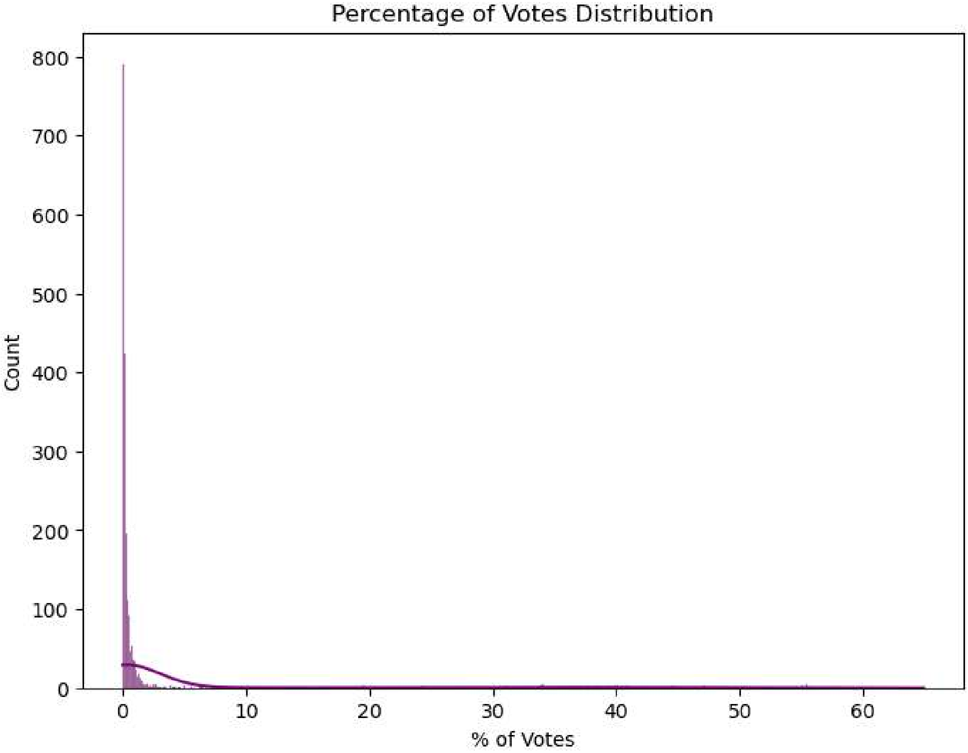
```
Candidate names and their corresponding constituency names with the lowest percentage of votes:
                     Candidate           Constituency
                  JAMEEL SHAIK          112 - Khammam
      NAKIRIKANTI SURESH KUMAR          112 - Khammam
                  NAZIMA BEGUM          112 - Khammam
                 BANOTH PRASAD          112 - Khammam
        MARAKAPUDI SRINIVASULU          112 - Khammam
             MUTYAM ARJUN RAJU          112 - Khammam
               PITTALA SAIDULU             90 - Kodad
                 BADE SRIVIDYA             90 - Kodad
Ayyalasomayajula Naga Srinivas 49 - Lal Bahadur Nagar
     EESARI SURYAPRAKASH REDDY 49 - Lal Bahadur Nagar
```

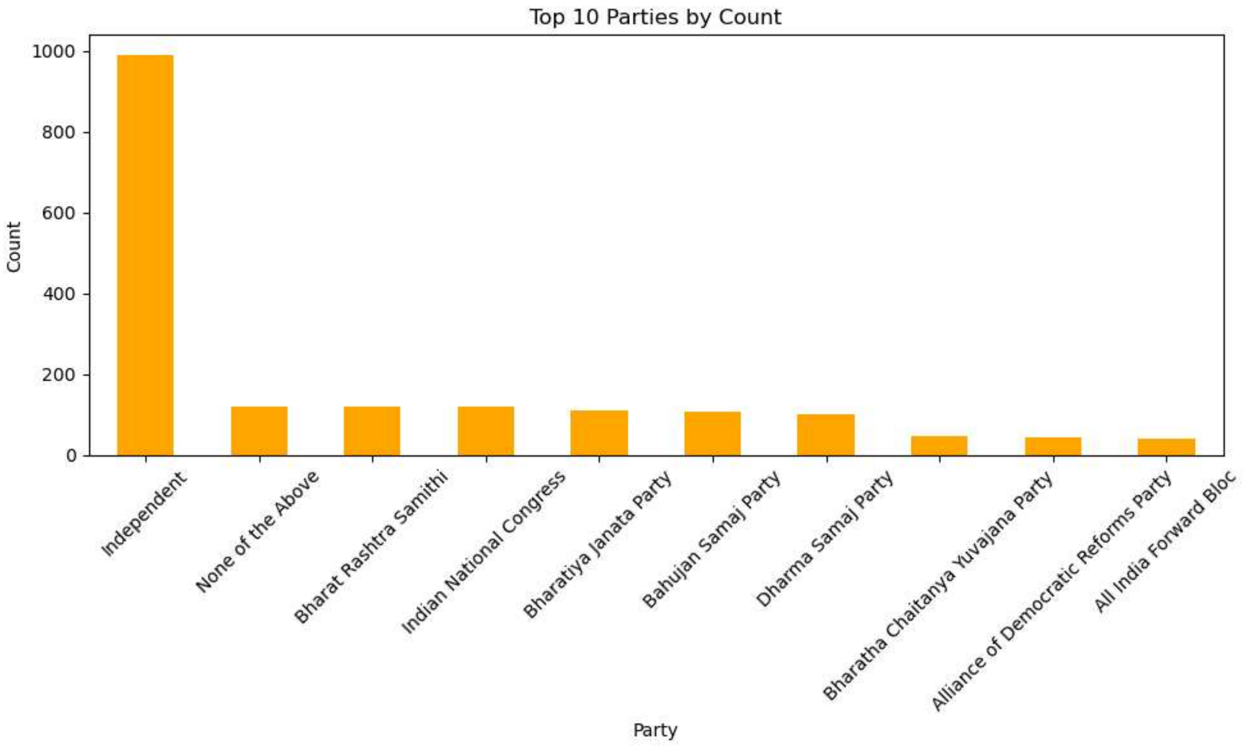
In [41]:

```
# Descriptive statistics for integer columns
print("Descriptive statistics for integer columns:")
print(ts[['EVM Votes', 'Postal Votes', 'Total Votes']].describe())

# Descriptive statistics for float columns
print("\nDescriptive statistics for float columns:")
print(ts['% of Votes'].describe())

# Count of unique values for object columns
print("\nCount of unique values for object columns:")
print(ts[['Candidate', 'Party', 'Constituency']].nunique())

# Create histograms for integer columns
plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
sns.histplot(ts['EVM Votes'], kde=True, color='skyblue')
plt.title('EVM Votes Distribution')
plt.subplot(1, 3, 2)
sns.histplot(ts['Postal Votes'], kde=True, color='salmon')
plt.title('Postal Votes Distribution')
plt.subplot(1, 3, 3)
sns.histplot(ts['Total Votes'], kde=True, color='green')
plt.title('Total Votes Distribution')
plt.tight_layout()
plt.show()

# Create histogram for float column
plt.figure(figsize=(8, 6))
sns.histplot(ts['% of Votes'], kde=True, color='purple')
plt.title('Percentage of Votes Distribution')
plt.show()

# Create bar plot for top parties
top_parties = ts['Party'].value_counts().nlargest(10)
plt.figure(figsize=(10, 6))
top_parties.plot(kind='bar', color='orange')
plt.title('Top 10 Parties by Count')
plt.xlabel('Party')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
Descriptive statistics for integer columns:
           EVM Votes  Postal Votes    Total Votes
count    2409.000000   2409.000000    2409.000000
mean     9646.899543     83.218348    9730.117891
std     25608.097526    282.182086   25833.309113
min        17.000000      0.000000      17.000000
25%       151.000000      0.000000     152.000000
50%       393.000000      1.000000     394.000000
75%      1388.000000      7.000000    1394.000000
max    187327.000000   4501.000000  187999.000000

Descriptive statistics for float columns:
count    2409.000000
mean        4.939797
std        12.762166
min         0.010000
25%         0.080000
50%         0.200000
75%         0.730000
max        64.890000
Name: % of Votes, dtype: float64

Count of unique values for object columns:
Candidate       2265
Party            105
Constituency     119
dtype: int64
```


EVM Votes Distribution
Count
0
100
200
300
400
500
600
700
800
0
50000
100000
150000
EVM Votes
Postal Votes Distribution
Count
0
200
400
600
800
1000
1200
0
1000
2000
3000
4000
Postal Votes
Total Votes Distribution
Count
0
100
200
300
400
500
600
700
800
0
50000
100000
150000
Total Votes


Percentage of Votes Distribution
Count
0
100
200
300
400
500
600
700
800
0
10
20
30
40
50
60
% of Votes



In [42]:

```
# Filter the DataFrame for independent candidates
independent_candidates = ts[ts['Party'] == 'Independent']

# Sort the DataFrame by total votes in descending order and select the top 10 independent candidates
top_10_independent = independent_candidates.nlargest(10, 'Total Votes')

# Display the required information
required_info = top_10_independent[['Candidate', 'Constituency', 'Total Votes', '% of Votes']]
print("Top 10 Independent Candidates with Highest Total Votes:")
print(required_info)
```

```
Top 10 Independent Candidates with Highest Total Votes:
                        Candidate      Constituency  Total Votes  % of Votes
141            KOTNAKA VIJAY KUMAR      5 - Asifabad        16469        8.92
969   Karne Shireesha @Barrelakka     85 - Kollapur         5754        2.99
558                GADDA. SATHISH     32 - Husnabad         5104        2.47
2072               PILLI SAI KUMAR    33 - Siddipet         4970        2.74
1369              MANMOHAN JADHAV      10 - Mudhole         4939        2.44
1738     MADAVAPEDDI VENKAT REDDY  100 - Palakurthi         4146        1.88
1903       SOMARAPU SATYANARAYANA   23 - Ramagundam         4048        2.65
406               Sherla Mahendar   22 - Dharmapuri         3847        2.12
1381       Maddila Venkateshwarlu       109 - Mulug         3709        1.98
191                  PUTTA BHASKAR     14 - Banswada         3671        2.29
```

In [45]:

```
# Filter the DataFrame for candidates belonging to Bharat Rashtra Samithi (BRs)
brs_df = ts[ts['Party'] == 'Bharat Rashtra Samithi']

# Calculate the total votes for Bharat Rashtra Samithi (BRs)
total_votes_brs = brs_df['Total Votes'].sum()

# Display the total votes for Bharat Rashtra Samithi (BRs)
print("Total Votes for Bharat Rashtra Samithi (BRs):", total_votes_brs)
```

```
Total Votes for Bharat Rashtra Samithi (BRs): 8753924
```

In [50]:

```
ts.dtypes
```

Out[50]:

```
Unnamed: 0        int64
S.N.              int64
Candidate        object
Party            object
EVM Votes         int64
Postal Votes      int64
Total Votes       int64
% of Votes      float64
Constituency     object
dtype: object
```

In [51]:

```
# Calculate the total votes from all parties
total_votes_all_parties = ts['Total Votes'].sum()

# Display the total votes from all parties
print("Total Votes from all parties:", total_votes_all_parties)
```

```
Total Votes from all parties: 23439854
```

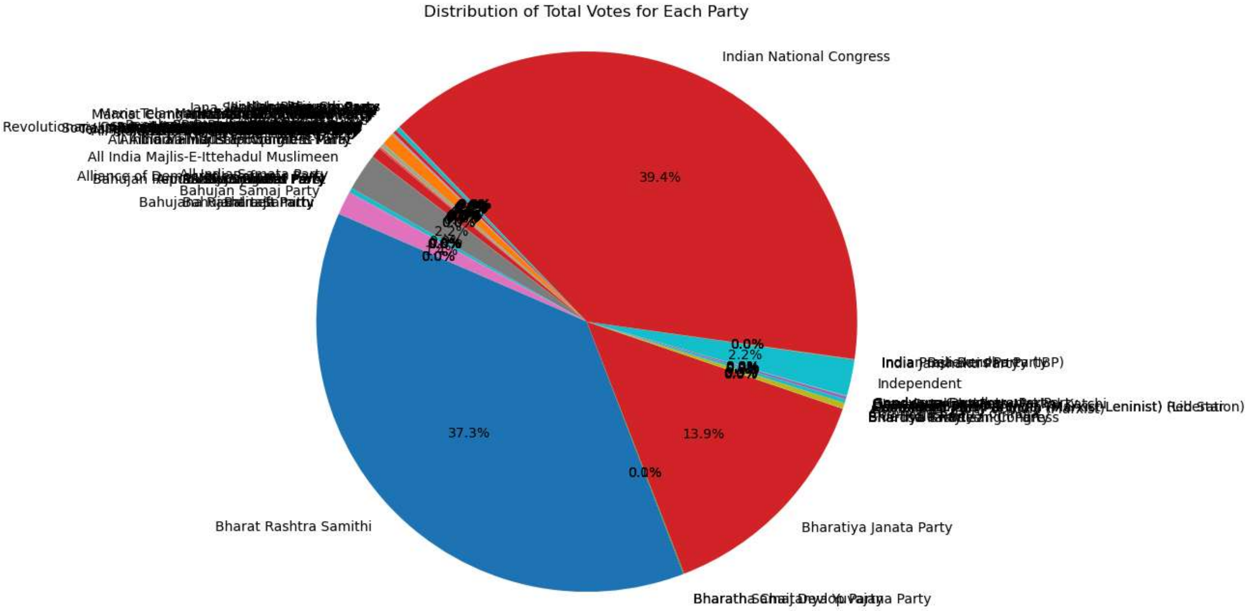
In [53]:

```
percentage_votes_brs = (total_votes_brs / total_votes_all_parties) * 100

# Display the percentage
print("BRs vote percentage relative to contested candidates:", percentage_votes_brs)
```

```
BRs vote percentage relative to contested candidates: 37.34632476806382
```

In [54]:

```
# Group the DataFrame by Party and sum the Total Votes for each party
party_votes = ts.groupby('Party')['Total Votes'].sum()

# Plotting the pie chart
plt.figure(figsize=(10, 8))
plt.pie(party_votes, labels=party_votes.index, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Total Votes for Each Party')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```



In [55]:

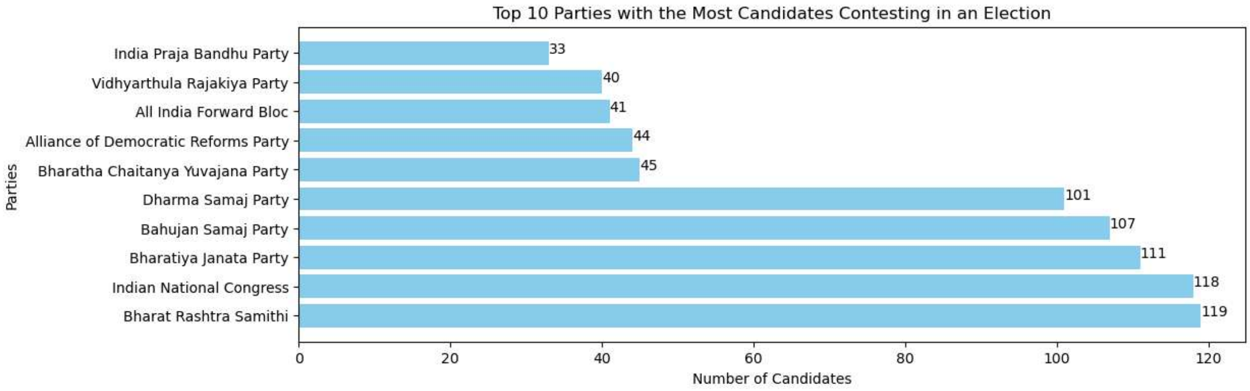
```
# Filter the DataFrame to include only rows where the candidate is NOTA
nota_df = ts[ts['Candidate'] == 'NOTA']

# Group the filtered DataFrame by constituency and candidate, and sum the votes
nota_votes_by_constituency = nota_df.groupby(['Constituency', 'Candidate'])['Total Votes'].sum()

# Find the constituency where NOTA votes are the highest
constituency_with_highest_nota_votes = nota_votes_by_constituency.idxmax()

# Retrieve the names of candidates and their votes for all candidates in the constituency with highest NOTA votes
candidates_in_constituency = ts[ts['Constituency'] == constituency_with_highest_nota_votes[0]][['Candidate', 'Total '

# Display the results
print("Constituency with the highest NOTA votes:", constituency_with_highest_nota_votes[0])
print("Candidates and their votes in the constituency:")
print(candidates_in_constituency)
```

```
Constituency with the highest NOTA votes: 45 - Quthbullapur
Candidates and their votes in the constituency:
                     Candidate  Total Votes
1839      KUNA SRISAILAM GOUD       102423
1840     MOHAMMED AHMED LAMRA         1759
1841          K.P. VIVEKANAND       187999
1842     KOLAN HANMANTH REDDY       101554
1843      MEKALA KARTHIK YADAV         746
1844    CHOUDHARY GARI SWATIKA         247
1845             THOTA SUVARNA         256
1846   DHONTULA RAMESH MUDIRAJ         137
1847            MOHAMMED WAJID         150
1848                  RAVINDAR         199
1849              D. DURGA RAO         299
1850    MUTHYAPAGA SHIVA KUMAR         435
1851          MOHAMMED MAHISAN         279
1852     BAGILI SRINIVAS REDDY         502
1853        SAI KUMAR PANTHULA         603
1854                      NOTA        4079
```

PArty wise contested bar plot

In [61]:

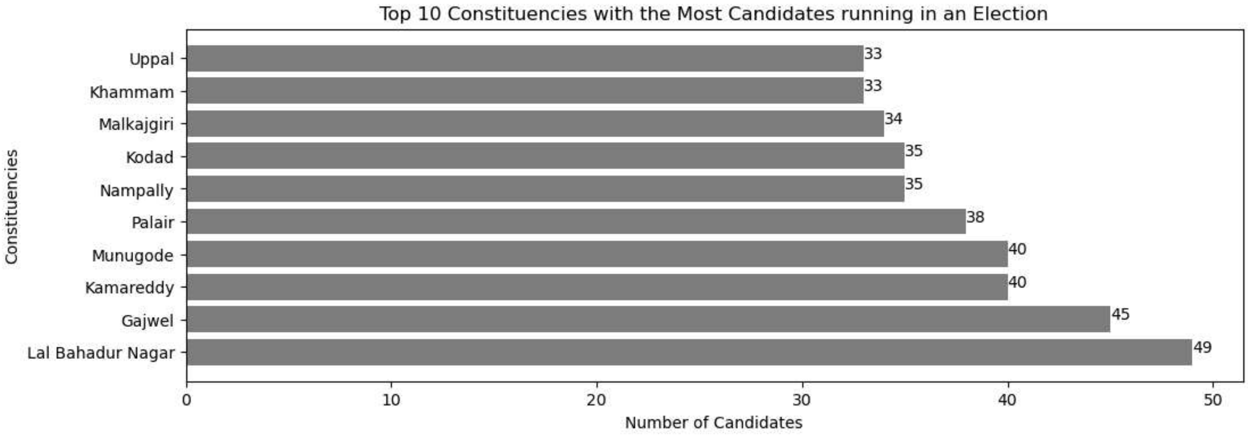
```
#Party wise candidates contested
contest = df.groupby('Party')['Candidate'].count().reset_index()
contest = contest.sort_values(by = 'Candidate', ascending = False)

#removing the independent and None of the above

contest = contest.loc[~((contest['Party'] == 'Independent') | (contest['Party'] == 'None of the Above'))]
contest.columns = ['Party', 'Candidates_contested']

top10_contest = contest.head(10)

#Plotting the Top10 Parties with most candidates participating in an election

plt.figure(figsize = (12,4))
plt.title("Top 10 Parties with the Most Candidates Contesting in an Election")

# Adding labels
plt.xlabel('Number of Candidates')
plt.ylabel('Parties')

plt.barh(top10_contest['Party'], top10_contest['Candidates_contested'], color='skyblue')

for index, value in enumerate(top10_contest['Candidates_contested']):
    plt.text(value, index, str(value))
```



Cosntituency wise candidates

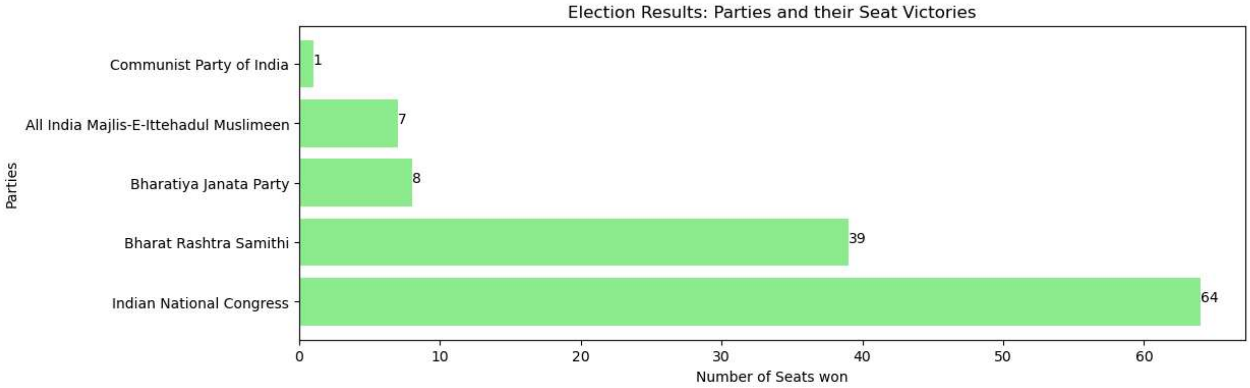
In [62]:

```
#Cosntituency wise candidates

const = df.groupby('Constituency')['Candidate'].count().reset_index()

const = const.sort_values(by = 'Candidate', ascending = False)

const.columns = ['Constituency', 'Candidates contested']

top10_const = const.head(10)

#Plotting the bar for constituency wise candidates
plt.figure(figsize = (12,4))
plt.title("Top 10 Constituencies with the Most Candidates running in an Election")

plt.barh(top10_const['Constituency'], top10_const['Candidates contested'], color='grey')

# Adding labels
plt.xlabel('Number of Candidates')
plt.ylabel('Constituencies')

for index, value in enumerate(top10_const['Candidates contested']):
    plt.text(value, index, str(value))
```



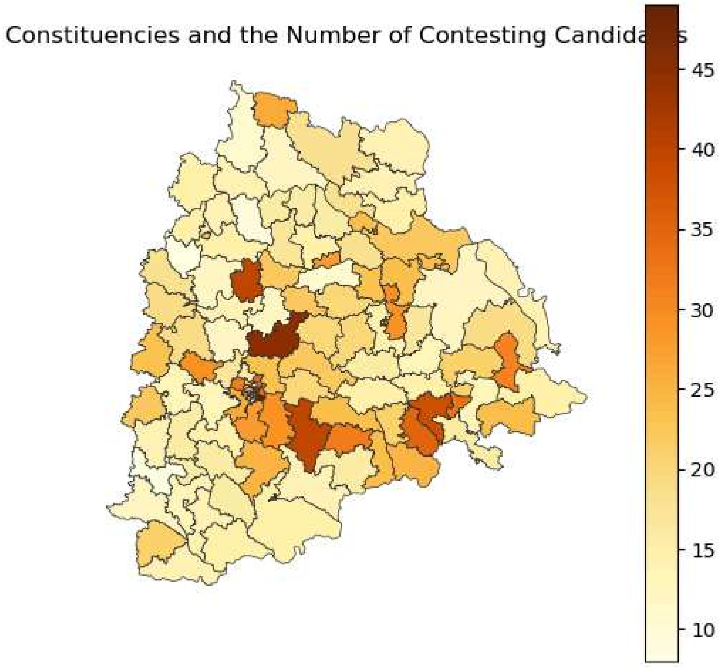
In [65]:

```
#Winning status of each candidate
df['result'] = df.groupby('Constituency')['Total Votes'].transform(max) == df['Total Votes']
df['result'] = df['result'].replace({True: "Win", False: "Loss"})

#Winners
won_candidates = df[df['result'] == 'Win']

#Party wise no of winners
party_won = won_candidates.groupby(['Party']).count().sort_values(by = 'Total Votes', ascending = False).reset_index

party_won = party_won.loc[:,['Party','Candidate']]

#Renaming the columns
column_mapping2 = {'Party': 'party', 'Candidate':'candidates_won'}
party_won = party_won.rename(columns = column_mapping2, inplace = False)

#Plotting Parties with highest winning seats
plt.figure(figsize = (12,4))
plt.title("Election Results: Parties and their Seat Victories")

plt.barh(party_won['party'], party_won['candidates_won'], color='lightgreen')

# Adding labels
plt.xlabel('Number of Seats won')
plt.ylabel('Parties')

for index, value in enumerate(party_won['candidates_won']):
    plt.text(value, index, str(value))
```

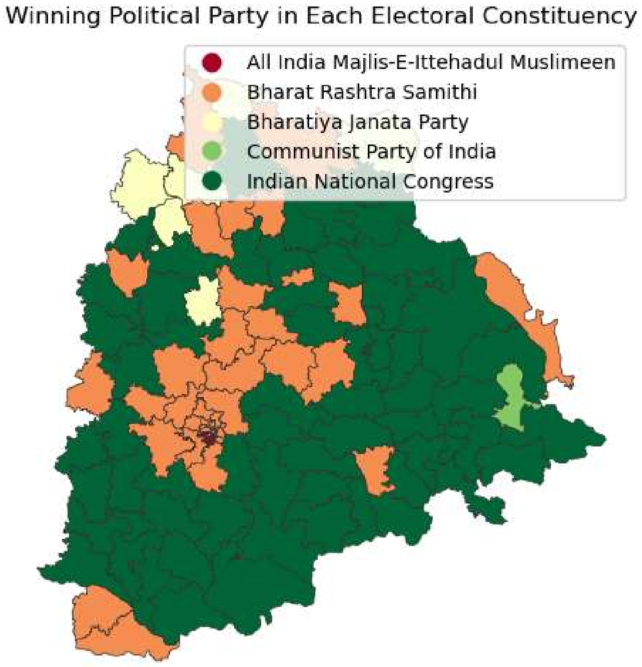


ts map analytics using geopandas

```
In [63]: #Merging the data
         map_const = pd.merge(spatial_df,const, on='Constituency')

         #Map plot
         fig, ax = plt.subplots(1, figsize=(6, 6))
         plt.title("Constituencies and the Number of Contesting Candidates")
         ax.axis('off')
         fig = map_const.plot(column='Candidates contested', cmap='YlOrBr', linewidth=0.5, ax=ax, edgecolor='0.2',legend=True
```

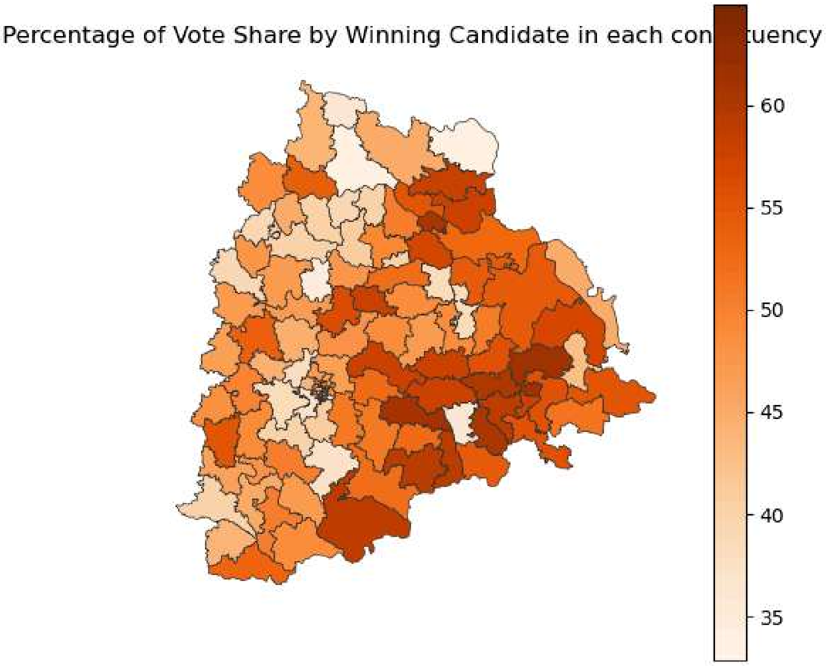


```
In [73]: #Merging the data
         map_won = pd.merge(spatial_df,won_candidates, on='Constituency')

         #Map plot
         fig2, ax = plt.subplots(1, figsize=(6, 6))
         plt.title("Winning Political Party in Each Electoral Constituency")
         ax.axis('off')
         fig2 = map_won.plot(column='Party', cmap='RdYlGn', linewidth=0.5, ax=ax, edgecolor='0.2', legend = True)
```


Winning Political Party in Each Electoral Constituency
● All India Majlis-E-Ittehadul Muslimeen
● Bharat Rashtra Samithi
● Bharatiya Janata Party
● Communist Party of India
● Indian National Congress

In [74]:

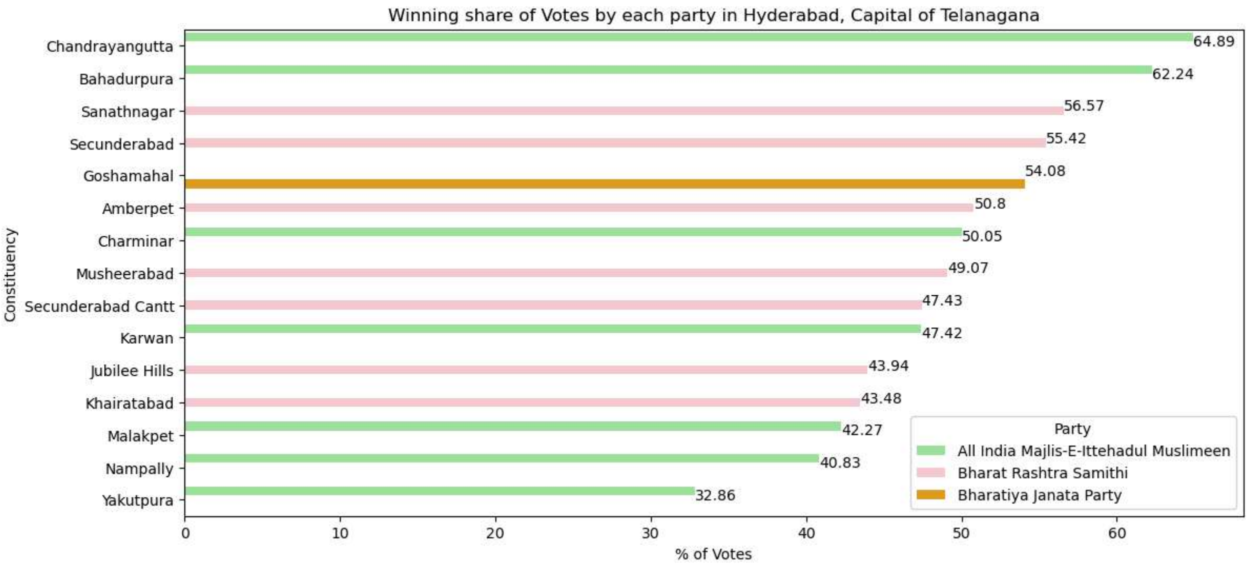
```
#Map plot
fig3, ax = plt.subplots(1, figsize=(6, 6))
plt.title("Percentage of Vote Share by Winning Candidate in each constituency ")
ax.axis('off')
fig3 = map_won.plot(column='% of Votes', cmap='Oranges', linewidth=0.5, ax=ax, edgecolor='0.2', legend = True)
```


Percentage of Vote Share by Winning Candidate in each con tuency
60
55
50
45
40
35

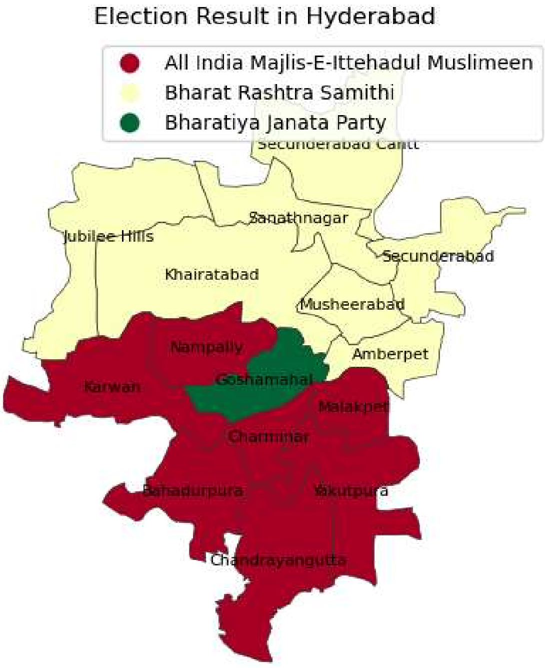
voing in the capital city

In [75]:

```
df_city = won_candidates.copy()
df_city['Constituency_no'] = df_city['Constituency_no'].astype(int)

#Constituency numbers within Hyderabad from 57-71. Lets filter the Hyderabad data
Capital_city = df_city[(df_city['Constituency_no'] >= 57) & (df_city['Constituency_no'] <= 71)]

Capital_city = Capital_city.sort_values(by = '% of Votes', ascending = False)


plt.figure(figsize = (13, 6))

# Create a dictionary to map parties to specific colors
party_colors = {'Bharat Rashtra Samithi': 'pink', 'All India Majlis-E-Ittehadul Muslimeen': 'lightgreen', 'Bharatiya

plt.title('Winning share of Votes by each party in Hyderabad, Capital of Telanagana')
sns.barplot(x='% of Votes', y='Constituency', hue='Party', data=Capital_city, dodge=True, palette=party_colors)

for index, value in enumerate(Capital_city['% of Votes']):
    plt.text(value, index, str(value))
```



```
In [76]: #Merging the data
         map_capital = pd.merge(spatial_df,Capital_city, on='Constituency')

         map_capital['geometry'] = map_capital['geometry'].to_crs(epsg=3395)
         #Map plot
         fig3, ax = plt.subplots(1, figsize=(6, 6))
         plt.title("Election Result in Hyderabad")
         ax.axis('off')
         fig3 = map_capital.plot(column='Party', cmap='RdYlGn', linewidth=0.5, ax=ax, edgecolor='0.2', legend = True)

         for x, y, label in zip(map_capital.geometry.centroid.x, map_capital.geometry.centroid.y, map_capital['Constituency']
             ax.text(x, y, label, fontsize=8, ha='center', va='center', color='black')
```


Election Result in Hyderabad
● All India Majlis-E-Ittehadul Muslimeen
● Bharat Rashtra Samithi
● Bharatiya Janata Party
Secunderabad Cantt
Jubilee Hills
Sanathnagar
Secunderabad
Khairatabad
Musheerabad
Nampally
Amberpet
Karwan
Goshamahal
Malakpet
Charminar
Bahadurpura
Yakutpura
Chandrayangutta