In [1]:
```
import pandas as pd
```

Reading the file

In [64]:
```
df=pd.read_csv('TN election 2021.csv')
df
```

Out[64]:

	State_Name	Assembly_No	Constituency_No	Year	month	DelimID	Poll_No	Position	Candidate	Sex	...	No_Terms	Turnc
0	Tamil_Nadu	12	1	2021	5	4	0	1	GOVINDARAJAN T.J	M	...	1.0	Fa
1	Tamil_Nadu	12	1	2021	5	4	0	2	PRAKASH M	M	...	0.0	Fa
2	Tamil_Nadu	12	1	2021	5	4	0	3	USHA	F	...	0.0	Fa
3	Tamil_Nadu	12	1	2021	5	4	0	4	DILLIY K.M	M	...	0.0	Fa
4	Tamil_Nadu	12	1	2021	5	4	0	5	None Of The Above	NaN	...	NaN	N
...	...	...	...	...	...	...	...	...	...	...	...	...	
4227	Tamil_Nadu	12	234	2021	5	4	0	11	JOHN BENADICT G	M	...	0.0	Fa
4228	Tamil_Nadu	12	234	2021	5	4	0	12	ANCHALOSE R	M	...	0.0	Fa
4229	Tamil_Nadu	12	234	2021	5	4	0	13	SHAJU SINGH M R	M	...	0.0	Fa
4230	Tamil_Nadu	12	234	2021	5	4	0	14	VIJIKUMAR K	M	...	0.0	Fa

	State_Name	Assembly_No	Constituency_No	Year	month	DelimID	Poll_No	Position	Candidate	Sex	...	No_Terms	Turnc
4231	Tamil_Nadu	12	234	2021	5	4	0	15	THANKAPPAN C	M	...	0.0	Fa

4232 rows × 47 columns

Checking if all the values in selected columns are the same

In [65]:
```
columns_to_check = ["State_Name", "Assembly_No", "Year", "month", "DelimID", "Poll_No", "Election_Type","last_poll"]

#CREATING A NEW DICTIONARY TO STORE THE RESULTS
results = {}

# CHECKING IN EACH COLUMN USING IF ELSE
for col in columns_to_check:
    unique_values = df[col].unique()
    if len(unique_values) == 1:
        results[col] = True
    else:
        results[col] = False


for col, result in results.items():
    print(f"{col}: All values are the same - {result}")
```

```
State_Name: All values are the same - True
Assembly_No: All values are the same - True
Year: All values are the same - True
month: All values are the same - True
DelimID: All values are the same - True
Poll_No: All values are the same - True
Election_Type: All values are the same - True
last_poll: All values are the same - True
```

Removing coulmns which have same values

In [66]:
```
columns_to_remove = ["State_Name", "Assembly_No", "Year", "month", "DelimID", "Poll_No", "Election_Type","last_poll"]
df = df.drop(columns=columns_to_remove)
df.head()
```

Out[66]:

	Constituency_No	Position	Candidate	Sex	Party	Votes	Age	Candidate_Type	Valid_Votes	Electors	...	Same_Party	No_Terr
0	1	1	GOVINDARAJAN T.J	M	DMK	126452	60.0	NaN	222069	284412	...	NaN	1
1	1	2	PRAKASH M	M	PMK	75514	50.0	NaN	222069	284412	...	NaN	(
2	1	3	USHA	F	NTK	11701	31.0	NaN	222069	284412	...	NaN	(
3	1	4	DILLIY K.M	M	DMDK	2576	45.0	NaN	222069	284412	...	NaN	(
4	1	5	None Of The Above	NaN	NOTA	1783	NaN	NaN	222069	284412	...	NaN	Na

5 rows × 39 columns

Checking if all values in selected rows are null

In [67]:
```
#SELECTING REQUIRED COLUMNS
columns_to_check = ["Candidate_Type", "Party_Type_TCPD"]


results = {}

# CHECKING CONDITION FOR NULL IN EACH ROW
for col in columns_to_check:
    if df[col].isnull().all():
        results[col] = True
    else:
        results[col] = False

# Print results
for col, result in results.items():
    if result:
        print(f"{col}: All values are null")
    else:
        print(f"{col}: Not all values are null")
```

```
Candidate_Type: All values are null
Party_Type_TCPD: All values are null
```

Removing columns which have null values

In [68]:
```
columns_to_remove1 = ["Candidate_Type", "Party_Type_TCPD"]



df = df.drop(columns=columns_to_remove1)
df
```

Out[68]:

	Constituency_No	Position	Candidate	Sex	Party	Votes	Age	Valid_Votes	Electors	Constituency_Name	...	Same_Par
0	1	1	GOVINDARAJAN T.J	M	DMK	126452	60.0	222069	284412	GUMMIDIPUNDI	...	N
1	1	2	PRAKASH M	M	PMK	75514	50.0	222069	284412	GUMMIDIPUNDI	...	N
2	1	3	USHA	F	NTK	11701	31.0	222069	284412	GUMMIDIPUNDI	...	N
3	1	4	DILLIY K.M	M	DMDK	2576	45.0	222069	284412	GUMMIDIPUNDI	...	N
4	1	5	None Of The Above	NaN	NOTA	1783	NaN	222069	284412	GUMMIDIPUNDI	...	N
...	...	...	...	...	...	...	...	...	...	...	...	
4227	234	11	JOHN BENADICT G	M	IND	235	42.0	169918	257959	KILLIYOOR	...	N
4228	234	12	ANCHALOSE R	M	IND	213	61.0	169918	257959	KILLIYOOR	...	N
4229	234	13	SHAJU SINGH M R	M	IND	213	45.0	169918	257959	KILLIYOOR	...	N
4230	234	14	VIJIKUMAR K	M	National Democratic Party of South India	186	34.0	169918	257959	KILLIYOOR	...	N
4231	234	15	THANKAPPAN C	M	Tamilnadu Mahatma Gandhi Makkal Katchi	162	72.0	169918	257959	KILLIYOOR	...	N

4232 rows × 37 columns

Checking if count of Nota = Number of constituencies(234)

In [69]:
```
count_NOTA = (df['Party'] == 'NOTA').sum()


print("Occurrences of 'NOTA' in the 'Party' column:",count_NOTA)
```

```
Occurrences of 'NOTA' in the 'Party' column: 234
```

Checking if there any wrong values entered in selected column

In [70]:
```
unique_sex = df["Sex"].unique()

# PRINTING UNIQUE VALUES
print("Unique values in the 'Sex' column:")
print(unique_sex)
```

```
Unique values in the 'Sex' column:
['M' 'F' nan 'O']
```

In [71]:
```
filtered_df = df[df['Sex'].isna() | (df['Sex'] == 'O')]
pd.DataFrame(filtered_df)
```

Out[71]:

	Constituency_No	Position	Candidate	Sex	Party	Votes	Age	Valid_Votes	Electors	Constituency_Name	...	Same_Party	No_Te
4	1	5	None Of The Above	NaN	NOTA	1783	NaN	222069	284412	GUMMIDIPUNDI	...	NaN	I
18	2	6	None Of The Above	NaN	NOTA	1554	NaN	210354	268994	PONNERI	...	NaN	I
28	3	5	None Of The Above	NaN	NOTA	1665	NaN	232624	295358	TIRUTTANI	...	NaN	I
43	4	5	None Of The Above	NaN	NOTA	1872	NaN	214243	277516	THIRUVALLUR	...	NaN	I
57	5	7	None Of The Above	NaN	NOTA	2867	NaN	263736	361331	POONAMALLEE	...	NaN	I
...	...	...	...	...	...	...	...	...	...	...	...	...	
4164	230	6	None Of The Above	NaN	NOTA	930	NaN	184185	274657	NAGERCOIL	...	NaN	I
4182	231	8	None Of The Above	NaN	NOTA	878	NaN	182969	271991	COLACHEL	...	NaN	I
4193	232	6	None Of The Above	NaN	NOTA	1036	NaN	170156	242802	PADMANABHAPURAM	...	NaN	I
4206	233	6	None Of The Above	NaN	NOTA	782	NaN	167836	251138	VILAVANCODE	...	NaN	I
4224	234	8	None Of The Above	NaN	NOTA	754	NaN	169918	257959	KILLIYOOR	...	NaN	I

236 rows × 37 columns

In [72]:
```
# as we know that NAN will be 234 the rest has to be for O so O has 2 entries
```

displaying rows with gender 'O'

In [73]:
```
df_gender_O = df[df['Sex'] == 'O']
df_gender_O
```

Out[73]:

	Constituency_No	Position	Candidate	Sex	Party	Votes	Age	Valid_Votes	Electors	Constituency_Name	...	Same_Party
542	25	8	RADHA M (TRANSGENDER)	O	IND	203	53.0	153430	271077	MYLAPORE	...	NaN
3449	192	11	BHARATHI KANNAMMA. S	O	New Generation People's Party	140	63.0	147833	231713	MADURAI SOUTH	...	NaN

2 rows × 37 columns

In [74]:
```
# thus displaying the details of sex O who participated in the tn election 2021
```

Checking if all candidate's votes are less than total votes of the constituency

In [75]:
```
votes_greater_than_valid = df['Votes'] > df['Valid_Votes']
if votes_greater_than_valid.all():
    print("Votes are more than Valid_Votes for each candidate in the entire dataset.")
else:
    print("Votes are not more than Valid_Votes for each candidate in the entire dataset.")
```

```
Votes are not more than Valid_Votes for each candidate in the entire dataset.
```

Checking for null values in certain columns

In [76]:
```
columns_to_check = ['Constituency_No', 'Position', 'Candidate', 'Party', 'Votes',
                     'Valid_Votes', 'Electors', 'Constituency_Name', 'Constituency_Type',
                     'District_Name', 'Sub_Region', 'N_Cand', 'Turnout_Percentage',
                     'Vote_Share_Percentage', 'ENOP', 'Party_ID']
# CHECKING FOR NULL VALUES IN CERTAIN COLUMNS
null_values_check = df[columns_to_check].isnull().any()

for column, has_null_values in null_values_check.items():
    if has_null_values:
        print(f"Column '{column}' has null values.")
    else:
        print(f"Column '{column}' has no null values.")
```

```
Column 'Constituency_No' has no null values.
Column 'Position' has no null values.
Column 'Candidate' has no null values.
Column 'Party' has no null values.
Column 'Votes' has no null values.
Column 'Valid_Votes' has no null values.
Column 'Electors' has no null values.
Column 'Constituency_Name' has no null values.
Column 'Constituency_Type' has no null values.
Column 'District_Name' has no null values.
Column 'Sub_Region' has no null values.
Column 'N_Cand' has no null values.
Column 'Turnout_Percentage' has no null values.
Column 'Vote_Share_Percentage' has no null values.
Column 'ENOP' has no null values.
Column 'Party_ID' has no null values.
```

Checking wrong data in age column and changing it using mean

In [77]:
```
rows_with_null_age_count = len(df[(df['Candidate'] != 'None Of The Above') & (df['Age'].isnull())])
print("Number of rows where 'Age' is null, excluding 'None Of The Above (NOTA)' row:", rows_with_null_age_count)
```

```
Number of rows where 'Age' is null, excluding 'None Of The Above (NOTA)' row: 0
```

In [78]:
```
rows_with_age_below_21 = df[df['Age'] < 21]
pd.DataFrame(rows_with_age_below_21)
```

Out[78]:

	Constituency_No	Position	Candidate	Sex	Party	Votes	Age	Valid_Votes	Electors	Constituency_Name	...	Same_Party
3616	201	7	MUTHU MUNEESWARAN.M	M	IND	451	4.0	202275	291001	CUMBUM	...	NaN

1 rows × 37 columns

In [79]:
```
mean_age_excluding_nota = df.loc[df['Party'] != 'NOTA', 'Age'].mean()

# ROUND OFF
rounded_mean_age = round(mean_age_excluding_nota)

# REPLACING
df.loc[df['Age'] == 4, 'Age'] = rounded_mean_age

muthu_muneeswaran_age = df.loc[df['Candidate'] == 'MUTHU MUNEESWARAN.M', 'Age'].values[0]

# UPDATION
print(mean_age_excluding_nota)
print("Age of MUTHU MUNEESWARAN.M:", muthu_muneeswaran_age)
```

```
44.50725362681341
Age of MUTHU MUNEESWARAN.M: 45.0
```

Renaming columns

In [80]:
```
df = df.rename(columns={'MyNeta_education': 'Education_Qualification','TCPD_Prof_Main': 'Main_Profession','TCPD_Prof_Main_Desc
df
```

Out[80]:

	Constituency_No	Position	Candidate	Sex	Party	Votes	Age	Valid_Votes	Electors	Constituency_Name	...	Same_Par
0	1	1	GOVINDARAJAN T.J	M	DMK	126452	60.0	222069	284412	GUMMIDIPUNDI	...	N
1	1	2	PRAKASH M	M	PMK	75514	50.0	222069	284412	GUMMIDIPUNDI	...	N
2	1	3	USHA	F	NTK	11701	31.0	222069	284412	GUMMIDIPUNDI	...	N
3	1	4	DILLIY K.M	M	DMDK	2576	45.0	222069	284412	GUMMIDIPUNDI	...	N
4	1	5	None Of The Above	NaN	NOTA	1783	NaN	222069	284412	GUMMIDIPUNDI	...	N
...	...	...	...	...	...	...	...	...	...	...	...	
4227	234	11	JOHN BENADICT G	M	IND	235	42.0	169918	257959	KILLIYOOR	...	N
4228	234	12	ANCHALOSE R	M	IND	213	61.0	169918	257959	KILLIYOOR	...	N
4229	234	13	SHAJU SINGH M R	M	IND	213	45.0	169918	257959	KILLIYOOR	...	N
4230	234	14	VIJIKUMAR K	M	National Democratic Party of South India	186	34.0	169918	257959	KILLIYOOR	...	N
4231	234	15	THANKAPPAN C	M	Tamilnadu Mahatma Gandhi Makkal Katchi	162	72.0	169918	257959	KILLIYOOR	...	N

4232 rows × 37 columns

Creating a new column called result and saying who won and who lost

In [81]:
```
import numpy as np

def determine_result(position):
    if position == 1:
        return 'Won'
    else:
        return 'Lost'

# Apply the function to the 'Position' column to create the 'Result' column
df['Result'] = df['Position'].apply(determine_result)

# Set 'Result' as NaN for rows where the party is 'NOTA'
df.loc[df['Party'] == 'NOTA', 'Result'] = np.nan
```

In [82]:
```
df
```

Out[82]:

	Constituency_No	Position	Candidate	Sex	Party	Votes	Age	Valid_Votes	Electors	Constituency_Name	...	No_Term
0	1	1	GOVINDARAJAN T.J	M	DMK	126452	60.0	222069	284412	GUMMIDIPUNDI	...	1.
1	1	2	PRAKASH M	M	PMK	75514	50.0	222069	284412	GUMMIDIPUNDI	...	0.
2	1	3	USHA	F	NTK	11701	31.0	222069	284412	GUMMIDIPUNDI	...	0.
3	1	4	DILLIY K.M	M	DMDK	2576	45.0	222069	284412	GUMMIDIPUNDI	...	0.
4	1	5	None Of The Above	NaN	NOTA	1783	NaN	222069	284412	GUMMIDIPUNDI	...	Na
...	...	...	...	...	...	...	...	...	...	...	...	..
4227	234	11	JOHN BENADICT G	M	IND	235	42.0	169918	257959	KILLIYOOR	...	0.
4228	234	12	ANCHALOSE R	M	IND	213	61.0	169918	257959	KILLIYOOR	...	0.
4229	234	13	SHAJU SINGH M R	M	IND	213	45.0	169918	257959	KILLIYOOR	...	0.
4230	234	14	VIJIKUMAR K	M	National Democratic Party of South India	186	34.0	169918	257959	KILLIYOOR	...	0.
4231	234	15	THANKAPPAN C	M	Tamilnadu Mahatma Gandhi Makkal Katchi	162	72.0	169918	257959	KILLIYOOR	...	0.

4232 rows × 38 columns

Creating a new column called Non-Voters

```
In [83]: df['Non_Voters'] = df['Electors'] - df['Valid_Votes']
         df
```

Out[83]:

	Constituency_No	Position	Candidate	Sex	Party	Votes	Age	Valid_Votes	Electors	Constituency_Name	...	Turncoat
0	1	1	GOVINDARAJAN T.J	M	DMK	126452	60.0	222069	284412	GUMMIDIPUNDI	...	False
1	1	2	PRAKASH M	M	PMK	75514	50.0	222069	284412	GUMMIDIPUNDI	...	False
2	1	3	USHA	F	NTK	11701	31.0	222069	284412	GUMMIDIPUNDI	...	False
3	1	4	DILLIY K.M	M	DMDK	2576	45.0	222069	284412	GUMMIDIPUNDI	...	False
4	1	5	None Of The Above	NaN	NOTA	1783	NaN	222069	284412	GUMMIDIPUNDI	...	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
4227	234	11	JOHN BENADICT G	M	IND	235	42.0	169918	257959	KILLIYOOR	...	False
4228	234	12	ANCHALOSE R	M	IND	213	61.0	169918	257959	KILLIYOOR	...	False
4229	234	13	SHAJU SINGH M R	M	IND	213	45.0	169918	257959	KILLIYOOR	...	False
4230	234	14	VIJIKUMAR K	M	National Democratic Party of South India	186	34.0	169918	257959	KILLIYOOR	...	False
4231	234	15	THANKAPPAN C	M	Tamilnadu Mahatma Gandhi Makkal Katchi	162	72.0	169918	257959	KILLIYOOR	...	False

4232 rows × 39 columns

Trying to study about Invalid_Votes

Creating a new column called Non-Voters percentage

In [84]:
```
df['Non_Voters_Percentage'] = 100 - df['Turnout_Percentage']
df
```

Out[84]:

	Constituency_No	Position	Candidate	Sex	Party	Votes	Age	Valid_Votes	Electors	Constituency_Name	...	Incumber
0	1	1	GOVINDARAJAN T.J	M	DMK	126452	60.0	222069	284412	GUMMIDIPUNDI	...	Fal
1	1	2	PRAKASH M	M	PMK	75514	50.0	222069	284412	GUMMIDIPUNDI	...	Fal
2	1	3	USHA	F	NTK	11701	31.0	222069	284412	GUMMIDIPUNDI	...	Fal
3	1	4	DILLIY K.M	M	DMDK	2576	45.0	222069	284412	GUMMIDIPUNDI	...	Fal
4	1	5	None Of The Above	NaN	NOTA	1783	NaN	222069	284412	GUMMIDIPUNDI	...	Na
...	...	...	...	...	...	...	...	...	...	...	...	
4227	234	11	JOHN BENADICT G	M	IND	235	42.0	169918	257959	KILLIYOOR	...	Fal
4228	234	12	ANCHALOSE R	M	IND	213	61.0	169918	257959	KILLIYOOR	...	Fal
4229	234	13	SHAJU SINGH M R	M	IND	213	45.0	169918	257959	KILLIYOOR	...	Fal
4230	234	14	VIJIKUMAR K	M	National Democratic Party of South India	186	34.0	169918	257959	KILLIYOOR	...	Fal
4231	234	15	THANKAPPAN C	M	Tamilnadu Mahatma Gandhi Makkal Katchi	162	72.0	169918	257959	KILLIYOOR	...	Fal

4232 rows × 40 columns

Creating a new column called Non-Voters percentage

In [85]:
```
df['Non_Voters_Percentage_1'] = (df['Electors'] - df['Valid_Votes']) / df['Electors'] * 100
df
```

Out[85]:

	Constituency_No	Position	Candidate	Sex	Party	Votes	Age	Valid_Votes	Electors	Constituency_Name	...	Recontes
0	1	1	GOVINDARAJAN T.J	M	DMK	126452	60.0	222069	284412	GUMMIDIPUNDI	...	Fals
1	1	2	PRAKASH M	M	PMK	75514	50.0	222069	284412	GUMMIDIPUNDI	...	Fals
2	1	3	USHA	F	NTK	11701	31.0	222069	284412	GUMMIDIPUNDI	...	Fals
3	1	4	DILLIY K.M	M	DMDK	2576	45.0	222069	284412	GUMMIDIPUNDI	...	Fals
4	1	5	None Of The Above	NaN	NOTA	1783	NaN	222069	284412	GUMMIDIPUNDI	...	Na
...	...	...	...	...	...	...	...	...	...	...	...	..
4227	234	11	JOHN BENADICT G	M	IND	235	42.0	169918	257959	KILLIYOOR	...	Fals
4228	234	12	ANCHALOSE R	M	IND	213	61.0	169918	257959	KILLIYOOR	...	Fals
4229	234	13	SHAJU SINGH M R	M	IND	213	45.0	169918	257959	KILLIYOOR	...	Fals
4230	234	14	VIJIKUMAR K	M	National Democratic Party of South India	186	34.0	169918	257959	KILLIYOOR	...	Fals
4231	234	15	THANKAPPAN C	M	Tamilnadu Mahatma Gandhi Makkal Katchi	162	72.0	169918	257959	KILLIYOOR	...	Fals

4232 rows × 41 columns

In [86]:
```
df['Invalid_Votes_percentage'] = abs(df['Non_Voters_Percentage_1'] - df['Non_Voters_Percentage']) * 100
df
```

Out[86]:

	Constituency_No	Position	Candidate	Sex	Party	Votes	Age	Valid_Votes	Electors	Constituency_Name	...	Educatio
0	1	1	GOVINDARAJAN T.J	M	DMK	126452	60.0	222069	284412	GUMMIDIPUNDI	...	
1	1	2	PRAKASH M	M	PMK	75514	50.0	222069	284412	GUMMIDIPUNDI	...	
2	1	3	USHA	F	NTK	11701	31.0	222069	284412	GUMMIDIPUNDI	...	
3	1	4	DILLIY K.M	M	DMDK	2576	45.0	222069	284412	GUMMIDIPUNDI	...	
4	1	5	None Of The Above	NaN	NOTA	1783	NaN	222069	284412	GUMMIDIPUNDI	...	
...	...	...	...	...	...	...	...	...	...	...	...	
4227	234	11	JOHN BENADICT G	M	IND	235	42.0	169918	257959	KILLIYOOR	...	Gradu
4228	234	12	ANCHALOSE R	M	IND	213	61.0	169918	257959	KILLIYOOR	...	
4229	234	13	SHAJU SINGH M R	M	IND	213	45.0	169918	257959	KILLIYOOR	...	
4230	234	14	VIJIKUMAR K	M	National Democratic Party of South India	186	34.0	169918	257959	KILLIYOOR	...	
4231	234	15	THANKAPPAN C	M	Tamilnadu Mahatma Gandhi Makkal Katchi	162	72.0	169918	257959	KILLIYOOR	...	

4232 rows × 42 columns

In [87]:
```python
# CONDITION STATEMENT
df_invalid_votes_greater_than_0_5 = df[df['Invalid_Votes_percentage'] > 0.05]

if df_invalid_votes_greater_than_0_5.empty:
    print("No constituencies with 'Invalid_Votes_percentage' greater than 0.05.")
else:
    print("Constituencies where 'Invalid_Votes_percentage' is greater than 0.05:")
    print(df_invalid_votes_greater_than_0_5[['Constituency_Name', 'Sub_Region', 'Constituency_Type', 'District_Name']])
```

```
No constituencies with 'Invalid_Votes_percentage' greater than 0.05.
```

Extensively studying about pid

In [88]:
```python
if df['pid'].nunique() == len(df['pid']):
    print("All values in 'pid' are distinct.")
else:
   #retrieving repeated values in dataframe
    repeated_pid = df[df.duplicated(subset=['pid'], keep=False)]['pid']
    print("Values in 'pid' that are repeated:")
    print(repeated_pid)
```

```
Values in 'pid' that are repeated:
4             NaN
18            NaN
21      GETN10499
28            NaN
43            NaN
          ...
4164          NaN
4182          NaN
4193          NaN
4206          NaN
4224          NaN
Name: pid, Length: 244, dtype: object
```

In [89]:
```python
df_not_nota = df[df['Party'] != 'NOTA']

#checking if all reapeated value is of NOTA
if len(df_not_nota['pid'].unique()) != len(df_not_nota):
    repeated_pid_values = df_not_nota[df_not_nota.duplicated(subset=['pid'], keep=False)]['pid'].unique()
    print("Values with repeated 'pid':")
    print(repeated_pid_values)
else:
    print("All values in 'pid' column are distinct.")
```

```
Values with repeated 'pid':
['GETN10499' 'AETN272830' 'AETN271206' 'AETN273808' 'AETN275818']
```

In [ ]:

In [ ]:

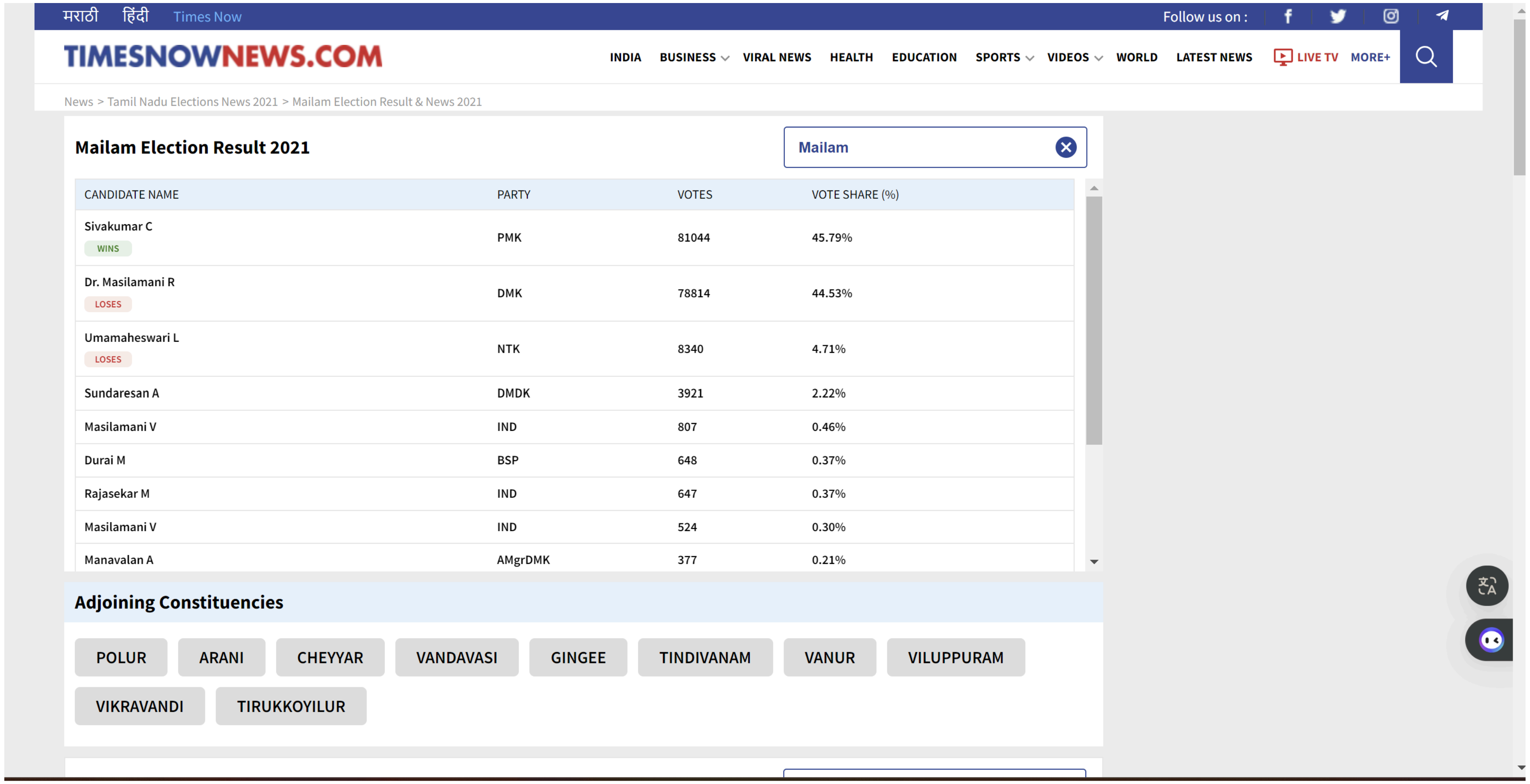
In [90]:

```
pid_values_to_retrieve = ['GETN10499', 'AETN272830', 'AETN271206', 'AETN273808', 'AETN275818']

# Filter the DataFrame to include only rows with the specified 'pid' values
df_filtered = df[df['pid'].isin(pid_values_to_retrieve)]
# Select specific columns from the filtered DataFrame
df_selected_columns = df_filtered[['Candidate', 'Sex', 'Party', 'Votes', 'Age', 'Constituency_Name', 'Education_Qualification'

df_sorted = df_selected_columns.sort_values(by='Candidate')
df_sorted
```

Out[90]:

	Candidate	Sex	Party	Votes	Age	Constituency_Name	Education_Qualification	Main_Profession	Main_Profession_De
951	KALASTHRI.S.	M	All India Uzhavargal Uzhaippalargal Katchi	870	71.0	JOLARPET	10th Pass	Agriculture	Na
1298	MASILAMANI V	M	IND	807	58.0	MAILAM	Illiterate	Labourer or Daily Wage	Na
1301	MASILAMANI V	M	IND	524	56.0	MAILAM	Illiterate	Labourer or Daily Wage	Na
2593	NIRMAL KUMAR M	M	IND	544	41.0	MANAPPARAI	Others	Other	Na
148	NIRMAL KUMAR. M	M	IND	101	35.0	MADAVARAM	12th Pass	Salaried Work or Employed	Na
3678	PALANICHAMY, P.	M	Bahujan Dravida Party	97	34.0	SATTUR	Post Graduate	Liberal Profession or Professional	Journalist or Edit
3711	PALANICHAMY.P	M	Bahujan Dravida Party	61	34.0	SIVAKASI	Post Graduate	Liberal Profession or Professional	Journalist or Edit
966	S. KALASHTHRI	M	All India Uzhavargal Uzhaippalargal Katchi	724	71.0	TIRUPPATTUR	10th Pass	Agriculture	Na
21	SUGUMAR. K	M	IND	298	40.0	PONNERI	Graduate Professional	Liberal Profession or Professional	Lawy
2236	Sugumar, K.	M	AMMK	1141	60.0	POLLACHI	Graduate	Business	Na

In [91]:
```
# KALASTHRI.S is the same person who has stood in 2 different constituencies
#MASILAMANI V is a noisy data or fraud data
#NIRMAL KUMAR M are two different people but are coming under the same pid so wrong
#PALANICHAMY.P is the same person who has stood in 2 different constituencies
#SUGUMAR. K are two different people but are coming under the same pid so wrong
```



Mailam Election Result 2021

CANDIDATE NAME	PARTY	VOTES	VOTE SHARE (%)
Sivakumar C WINS	PMK	81044	45.79%
Dr. Masilamani R LOSES	DMK	78814	44.53%
Umamaheswari L LOSES	NTK	8340	4.71%
Sundaresan A	DMDK	3921	2.22%
Masilamani V	IND	807	0.46%
Durai M	BSP	648	0.37%
Rajasekar M	IND	647	0.37%
Masilamani V	IND	524	0.30%
Manavalan A	AMgrDMK	377	0.21%

Adjoining Constituencies

POLUR ARANI CHEYYAR VANDAVASI GINGEE TINDIVANAM VANUR VILUPPURAM VIKRAVANDI TIRUKKOYILUR

In [92]:
```
#df.to_csv("wednesdaynight.csv", index=False)
```

Checking if N_cand is wrong for the whole dataset

In [93]:
```
#RETREIVING RECDS WITH MARGIN==0
df_margin_zero = df[df['Margin'] == 0]

# DIFFERENCE CONDITION
if (df_margin_zero['Position'] - df_margin_zero['N_Cand'] == 1).all():
    print("The difference between position and N_cand is 1 for all candidates with margin 0.")
else:
    print("The difference between position and N_cand is not 1 for all candidates with margin 0.")
```

```
The difference between position and N_cand is not 1 for all candidates with margin 0.
```

In [94]:
```
a=df_margin_zero['Position'] - df_margin_zero['N_Cand'] == 1
pd.DataFrame(a)
```

Out[94]:

	0
12	True
23	True
38	True
50	True
65	True
...	...
4187	True
4200	True
4216	True
4228	False
4231	True

325 rows × 1 columns

In [95]:
```
df_a = pd.DataFrame(a)
#CALCULATING TRUE AND FALSE
```

```
counts = df_a[0].value_counts()

# Print the counts
print("Number of True values:", counts[True])
print("Number of False values:", counts[False])
```

```
Number of True values: 234
Number of False values: 91
```

In [96]:
```
df['N_Cand'] = df['N_Cand'] + 1
df['N_Cand']
```

Out[96]:
```
0       13
1       13
2       13
3       13
4       13
        ..
4227    15
4228    15
4229    15
4230    15
4231    15
Name: N_Cand, Length: 4232, dtype: int64
```

In [97]:
```
df.rename(columns={'N_Cand': 'No_of_Candidates'}, inplace=True)
```

In [98]:
```
df
```

Out[98]:

	Constituency_No	Position	Candidate	Sex	Party	Votes	Age	Valid_Votes	Electors	Constituency_Name	...	Educatio
0	1	1	GOVINDARAJAN T.J	M	DMK	126452	60.0	222069	284412	GUMMIDIPUNDI	...	
1	1	2	PRAKASH M	M	PMK	75514	50.0	222069	284412	GUMMIDIPUNDI	...	
2	1	3	USHA	F	NTK	11701	31.0	222069	284412	GUMMIDIPUNDI	...	
3	1	4	DILLIY K.M	M	DMDK	2576	45.0	222069	284412	GUMMIDIPUNDI	...	
4	1	5	None Of The Above	NaN	NOTA	1783	NaN	222069	284412	GUMMIDIPUNDI	...	
...	...	...	...	...	...	...	...	...	...	...	...	
4227	234	11	JOHN BENADICT G	M	IND	235	42.0	169918	257959	KILLIYOOR	...	Gradu
4228	234	12	ANCHALOSE R	M	IND	213	61.0	169918	257959	KILLIYOOR	...	
4229	234	13	SHAJU SINGH M R	M	IND	213	45.0	169918	257959	KILLIYOOR	...	
4230	234	14	VIJIKUMAR K	M	National Democratic Party of South India	186	34.0	169918	257959	KILLIYOOR	...	
4231	234	15	THANKAPPAN C	M	Tamilnadu Mahatma Gandhi Makkal Katchi	162	72.0	169918	257959	KILLIYOOR	...	

4232 rows × 42 columns

In [ ]:

Checking for null values in personal details of candidate

In [99]:
```
count_NOTA = (df['Party'] == 'NOTA').sum()
print("Occurrences of 'NOTA' in the 'Party' column:",count_NOTA)
```

```
Occurrences of 'NOTA' in the 'Party' column: 234
```

In [100…
```
#UNIQUE VALUES IN EDU
education_counts = df['Education_Qualification'].value_counts(dropna=False)
#DISPLAYING UNIQUE VALUES
print("Unique values in 'Education_Qualification' and their counts, including NaN:")
print(education_counts)
```

```
Unique values in 'Education_Qualification' and their counts, including NaN:
Education_Qualification
10th Pass                690
Graduate Professional    539
Post Graduate            530
Graduate                 453
NaN                      447
8th Pass                 425
12th Pass                412
5th Pass                 325
Others                   205
Illiterate               109
Literate                  64
Doctorate                 33
Name: count, dtype: int64
```

In [101…
```
#COUNT OF NULL VALUES IN EDU
nan_count = df['Education_Qualification'].isna().sum()

# SUBRACTING 234 FROM THE COUNT OF NAN VALUES
result = nan_count - 234
```

```
print("Result after subtracting 234 from the count of NaN values:",result)
```

```
Result after subtracting 234 from the count of NaN values: 213
```

In [102…

```
#UNIQUE VALUES IN MAIN_PROF
main_profession_counts = df['Main_Profession'].value_counts(dropna=False)
print("Unique values in 'Main_Profession' and their counts, including NaN:")
print(main_profession_counts)
```

```
Unique values in 'Main_Profession' and their counts, including NaN:
Main_Profession
Other                                  777
Business                               765
Agriculture                            642
Labourer or Daily Wage                 514
Liberal Profession or Professional     430
NaN                                    422
Small Business or Self-employed        236
Salaried Work or Employed              136
Agricultural Labour                     90
Social Work                             62
Politics                                60
Retired or Pension                      33
Education                               29
Former Government                       19
Unemployed                               9
Traditional Occupation                   4
Student                                  2
Religious Occupation                     2
Name: count, dtype: int64
```

In [103…

```
#COUNT OF NULL VALUES IN MAIN_PROF
nan_count = df['Main_Profession'].isna().sum()
# SUBRACTING 234 FROM THE COUNT OF NAN VALUES
result = nan_count - 234
print("Result after subtracting 234 from the count of NaN values:",result)
```

```
Result after subtracting 234 from the count of NaN values: 188
```

In [104…

```
#COUNT OF NULL VALUES IN EDU
second_profession_counts = df['Second_Profession'].value_counts(dropna=False)
```

```
print("Unique values in 'Second_Profession' and their counts, including NaN:")
print(second_profession_counts)
```

```
Unique values in 'Second_Profession' and their counts, including NaN:
Second_Profession
NaN                                   4018
Agriculture                            124
Social Work                             33
Politics                                17
Salaried Work or Employed               14
Liberal Profession or Professional      10
Small Business or Self-employed          7
Education                                3
Student                                  2
Labourer or Daily Wage                   1
Agricultural Labour                      1
Retired or Pension                       1
Traditional Occupation                   1
Name: count, dtype: int64
```

In [105…

```
#COUNT OF NULL VALUES IN EDU
nan_count = df['Second_Profession'].isna().sum()
# SUBRACTING 234 FROM THE COUNT OF NAN VALUES
result = nan_count - 234
print("Result after subtracting 234 from the count of NaN values:",result)
```

```
Result after subtracting 234 from the count of NaN values: 3784
```

Creating new columns number of terms lost

In [106…

```
#ADDING NEW COLUMN
df['No_Terms_lost'] = df['Contested'] - df['No_Terms']
df
```

Out[106…

	Constituency_No	Position	Candidate	Sex	Party	Votes	Age	Valid_Votes	Electors	Constituency_Name	...	Main_Pro
0	1	1	GOVINDARAJAN T.J	M	DMK	126452	60.0	222069	284412	GUMMIDIPUNDI	...	E
1	1	2	PRAKASH M	M	PMK	75514	50.0	222069	284412	GUMMIDIPUNDI	...	Salaried En
2	1	3	USHA	F	NTK	11701	31.0	222069	284412	GUMMIDIPUNDI	...	Agr
3	1	4	DILLIY K.M	M	DMDK	2576	45.0	222069	284412	GUMMIDIPUNDI	...	
4	1	5	None Of The Above	NaN	NOTA	1783	NaN	222069	284412	GUMMIDIPUNDI	...	
...	...	...	...	...	...	...	...	...	...	...	...	
4227	234	11	JOHN BENADICT G	M	IND	235	42.0	169918	257959	KILLIYOOR	...	Profe Prof
4228	234	12	ANCHALOSE R	M	IND	213	61.0	169918	257959	KILLIYOOR	...	E
4229	234	13	SHAJU SINGH M R	M	IND	213	45.0	169918	257959	KILLIYOOR	...	Small Bus Self-en
4230	234	14	VIJIKUMAR K	M	National Democratic Party of South India	186	34.0	169918	257959	KILLIYOOR	...	E
4231	234	15	THANKAPPAN C	M	Tamilnadu Mahatma Gandhi Makkal Katchi	162	72.0	169918	257959	KILLIYOOR	...	Agr

4232 rows × 43 columns

In [107…
```
df_sort=df.sort_values(by='Contested', ascending=False)
```

In [108…
```
col=['Candidate','Age','Constituency_Name','pid','Party', 'Sub_Region','Contested']
a=df_sort[col]
a.head()
```

Out[108…

	Candidate	Age	Constituency_Name	pid	Party	Sub_Region	Contested
2360	M.A.ANDI AMBALAM	65.0	NATHAM	AETN197588	DMK	SOUTHERN REGION	13.0
815	DURAIMURUGAN	83.0	KATPADI	AETN271302	DMK	CHENNAI CITY REGION	12.0
1973	SENGOTTAIYAN K.A	73.0	GOBICHETTIPALAYAM	AETN211232	ADMK	WESTERN REGION	10.0
2622	Nehru, K.N.	68.0	TIRUCHIRAPALLI (West)	AETN200955	DMK	CENTRAL REGION	9.0
229	M.K. STALIN	68.0	KOLATHUR	AETN202458	DMK	CHENNAI CITY REGION	9.0

In [109…
```
filtered_df = df_sort[df_sort['Contested'].isin([2,3,4,5,6,7,8,9,10,11,12])]
#FINDING AVG
average_contested = filtered_df['Contested'].sum()
# COUNT OF ROWS
total_rows = len(filtered_df)
a=average_contested/total_rows

# Print the average contested value and total number of rows
print(a)
```

```
2.7684824902723735
```

In [110…
```
df.loc[df['pid'] == 'AETN197588', 'Contested']=3
```

In [111…
```
andi_ambalam=df[df['pid'] == 'AETN197588']
col=['Candidate','Age','Constituency_Name','pid','Party', 'Sub_Region','Contested']
pd.DataFrame(andi_ambalam[col])
```

Out[111…

	Candidate	Age	Constituency_Name	pid	Party	Sub_Region	Contested
2360	M.A.ANDI AMBALAM	65.0	NATHAM	AETN197588	DMK	SOUTHERN REGION	3.0

Creating new column called Alliance

In [112…

```
import numpy as np
df['Alliance'] = None  # INITIALIZATION

#UPDATION
df.loc[df['Party'] == 'DMK', 'Alliance'] = 'SPA'
df.loc[df['Party'] == 'INC', 'Alliance'] = 'SPA'
df.loc[df['Party'] == 'CPI', 'Alliance'] = 'SPA'
df.loc[df['Party'] == 'CPM', 'Alliance'] = 'SPA'
df.loc[df['Party'] == 'VCK', 'Alliance'] = 'SPA'
df.loc[df['Party'] == 'IUML', 'Alliance'] = 'SPA'
df.loc[df['Party'] == 'ADMK', 'Alliance'] = 'NDA'
df.loc[df['Party'] == 'PMK', 'Alliance'] = 'NDA'
df.loc[df['Party'] == 'BJP', 'Alliance'] = 'NDA'
df.loc[df['Party'] == 'AMMK', 'Alliance'] = 'PF'
df.loc[df['Party'] == 'DMDK', 'Alliance'] = 'PF'
df.loc[df['Party'] == 'SDPI', 'Alliance'] = 'PF'
df.loc[df['Party'] == 'AIMIM', 'Alliance'] = 'PF'
df.loc[df['Party'] == 'MNM', 'Alliance'] = 'PFA'
df.loc[df['Party'] == 'IJK', 'Alliance'] = 'PFA'
df.loc[df['Party'] == 'NTK', 'Alliance'] = 'NTK'
df.loc[df['Party'] == 'BSP', 'Alliance'] = 'BSP'
df.loc[df['Party'] == 'NOTA', 'Alliance'] = np.nan

# IND FOR OTHER SMALL PARTIES
df.loc[df['Alliance'].isnull(), 'Alliance']='IND'
df
```

Out[112…

	Constituency_No	Position	Candidate	Sex	Party	Votes	Age	Valid_Votes	Electors	Constituency_Name	...	Main_Pro
0	1	1	GOVINDARAJAN T.J	M	DMK	126452	60.0	222069	284412	GUMMIDIPUNDI	...	
1	1	2	PRAKASH M	M	PMK	75514	50.0	222069	284412	GUMMIDIPUNDI	...	Qualifie
2	1	3	USHA	F	NTK	11701	31.0	222069	284412	GUMMIDIPUNDI	...	
3	1	4	DILLIY K.M	M	DMDK	2576	45.0	222069	284412	GUMMIDIPUNDI	...	
4	1	5	None Of The Above	NaN	NOTA	1783	NaN	222069	284412	GUMMIDIPUNDI	...	
...	...	...	...	...	...	...	...	...	...	...	...	
4227	234	11	JOHN BENADICT G	M	IND	235	42.0	169918	257959	KILLIYOOR	...	
4228	234	12	ANCHALOSE R	M	IND	213	61.0	169918	257959	KILLIYOOR	...	
4229	234	13	SHAJU SINGH M R	M	IND	213	45.0	169918	257959	KILLIYOOR	...	
4230	234	14	VIJIKUMAR K	M	National Democratic Party of South India	186	34.0	169918	257959	KILLIYOOR	...	
4231	234	15	THANKAPPAN C	M	Tamilnadu Mahatma Gandhi Makkal Katchi	162	72.0	169918	257959	KILLIYOOR	...	

4232 rows × 44 columns

In [113…

```
df['Current_Status'] ='Unknown'
df.loc[(df['Result'] =='Won') & (df['Incumbent'] == True), 'Current_Status']='Consecutive win'
df.loc[(df['Result'] =='Won') & (df['Incumbent'] == False), 'Current_Status']='Position gained'
df.loc[(df['Result'] =='Lost') & (df['Incumbent'] == True), 'Current_Status']='Ex-MLA'
df.loc[(df['Result'] =='Lost') & (df['Incumbent'] == False), 'Current_Status']='Lost'


df.loc[df['Current_Status'] == 'Unknown', 'Current_Status'] = np.nan

df
```

Out[113…

	Constituency_No	Position	Candidate	Sex	Party	Votes	Age	Valid_Votes	Electors	Constituency_Name	...	Second_P
0	1	1	GOVINDARAJAN T.J	M	DMK	126452	60.0	222069	284412	GUMMIDIPUNDI	...	
1	1	2	PRAKASH M	M	PMK	75514	50.0	222069	284412	GUMMIDIPUNDI	...	
2	1	3	USHA	F	NTK	11701	31.0	222069	284412	GUMMIDIPUNDI	...	
3	1	4	DILLIY K.M	M	DMDK	2576	45.0	222069	284412	GUMMIDIPUNDI	...	
4	1	5	None Of The Above	NaN	NOTA	1783	NaN	222069	284412	GUMMIDIPUNDI	...	
...	...	...	...	...	...	...	...	...	...	...	...	
4227	234	11	JOHN BENADICT G	M	IND	235	42.0	169918	257959	KILLIYOOR	...	
4228	234	12	ANCHALOSE R	M	IND	213	61.0	169918	257959	KILLIYOOR	...	
4229	234	13	SHAJU SINGH M R	M	IND	213	45.0	169918	257959	KILLIYOOR	...	
4230	234	14	VIJIKUMAR K	M	National Democratic Party of South India	186	34.0	169918	257959	KILLIYOOR	...	
4231	234	15	THANKAPPAN C	M	Tamilnadu Mahatma Gandhi Makkal Katchi	162	72.0	169918	257959	KILLIYOOR	...	

4232 rows × 45 columns

Creating a new dataframe for candidates having participated in the previous election/elections also

In [114…

```
#DATA FRAME FOR CANDIDATES WITH PRIOR ELECTION EXPERIENCE
df_candidates_with_experience = df[((df['Contested'] != 1) | (df['Contested'].isnull())) & (df['Party'] != 'NOTA')]
print("DataFrame containing candidates with experience:")
df_candidates_with_experience
```

```
DataFrame containing candidates with experience:
```

Out[114…

	Constituency_No	Position	Candidate	Sex	Party	Votes	Age	Valid_Votes	Electors	Constituency_Name	...	Second_Profe
14	2	2	BALARAMAN. P	M	ADMK	84839	59.0	210354	268994	PONNERI	...	
17	2	5	PON. RAJA	M	AMMK	2832	45.0	210354	268994	PONNERI	...	
25	3	2	G.Hari	M	ADMK	91061	60.0	232624	295358	TIRUTTANI	...	
35	3	12	Shettu	M	IND	147	47.0	232624	295358	TIRUTTANI	...	
39	4	1	RAAJENDRAN, V.G.	M	DMK	107709	53.0	214243	277516	THIRUVALLUR	...	
...	...	...	...	...	...	...	...	...	...	...	...	
4188	232	1	Mano Thangaraj, T.	M	DMK	87744	53.0	170156	242802	PADMANABHAPURAM	...	
4201	233	1	VIJAYADHARANI S	F	INC	87473	51.0	167836	251138	VILAVANCODE	...	
4202	233	2	JAYASEELAN R	M	BJP	58804	56.0	167836	251138	VILAVANCODE	...	
4205	233	5	IDEN SONI L	M	DMDK	2447	45.0	167836	251138	VILAVANCODE	...	
4217	234	1	RAJESH KUMAR S	M	INC	101541	46.0	169918	257959	KILLIYOOR	...	

515 rows × 45 columns

In [52]:
```
df
```

Out[52]:

	Constituency_No	Position	Candidate	Sex	Party	Votes	Age	Valid_Votes	Electors	Constituency_Name	...	Main_Pro
0	1	1	GOVINDARAJAN T.J	M	DMK	126452	60.0	222069	284412	GUMMIDIPUNDI	...	
1	1	2	PRAKASH M	M	PMK	75514	50.0	222069	284412	GUMMIDIPUNDI	...	Qualifie
2	1	3	USHA	F	NTK	11701	31.0	222069	284412	GUMMIDIPUNDI	...	
3	1	4	DILLIY K.M	M	DMDK	2576	45.0	222069	284412	GUMMIDIPUNDI	...	
4	1	5	None Of The Above	NaN	NOTA	1783	NaN	222069	284412	GUMMIDIPUNDI	...	
...	...	...	...	...	...	...	...	...	...	...	...	
4227	234	11	JOHN BENADICT G	M	IND	235	42.0	169918	257959	KILLIYOOR	...	
4228	234	12	ANCHALOSE R	M	IND	213	61.0	169918	257959	KILLIYOOR	...	
4229	234	13	SHAJU SINGH M R	M	IND	213	45.0	169918	257959	KILLIYOOR	...	
4230	234	14	VIJIKUMAR K	M	National Democratic Party of South India	186	34.0	169918	257959	KILLIYOOR	...	
4231	234	15	THANKAPPAN C	M	Tamilnadu Mahatma Gandhi Makkal Katchi	162	72.0	169918	257959	KILLIYOOR	...	

4232 rows × 44 columns

In [62]:

```
#df.to_csv("tnelectionsformlmodels.csv")
```

In [112…

```
#df_candidates_with_experience.to_csv("candidates_with_experience1.csv")
```

VISUALISATION

VISUALISING THE SHARE OF EACH ALLIANCE FOR THE NEW ROW CREATED (CURRENT STATUS)

In [113…

```
import plotly.express as px
#ALLIANCE COLOUR MAPPING
alliance_color_map = {
    'NDA': 'darkgreen',
    'SPA': 'red',
    'PF': 'lightgreen',
    'NTK': 'black',
    'PFA': 'white',
    'BSP': 'blue'
}

# Filter the DataFrame to exclude 'IND' entries from the 'Alliance' column
df_filtered = df[df['Alliance'] != 'IND']

# Group the filtered DataFrame by 'Current Status' and 'Alliance', and count the occurrences
grouped_df = df_filtered.groupby(['Current_Status', 'Alliance']).size().reset_index(name='Count')

# Create a pie plot for each 'Current Status' category
for i in grouped_df['Current_Status'].unique():
    # Filter the grouped DataFrame for the current status
    status_df = grouped_df[grouped_df['Current_Status'] == i]
    fig = px.pie(status_df, values='Count', names='Alliance', title=f"{i} - Distribution by Alliance",
                 hole=0.3, labels={'Alliance': 'Alliance'}, hover_name='Alliance', color='Alliance', color_discrete_map=allian
    fig.show()
```

In [114…

```
#THIS IS A PIE PLOT WITH A HOLE USED WITH INTERATIVE PLOT PLOTLY EXPRESS
#AS WE CAN SEE FROM THE GRAPH WE CAN SAY THAT SPA HAS A HIGER MARGIN IN BOTH POSITION GAINED AND CONSECUTIVE WIN THUS
#MAKING THEM WIN THE ELECTION
# THERE IS A LARGE SHARE OF EX MLA FOR NDA ALLIANCE THUS WE CAN SAY THAT PEOPLE HAVE LOST TRUST IN PEOPLE WHO RULED THEM
```

In [115…

```
import plotly.express as px

# Filter the DataFrame for the 'won' entries in 'result' column
df_won = df[df['Result'] == 'Won']

# Group the filtered DataFrame by 'Sub_Region' and 'Alliance', and count the occurrences
grouped_df = df_won.groupby(['Sub_Region', 'Alliance']).size().reset_index(name='Count')

# Group the DataFrame by 'Sub_Region' and sum the counts for each alliance
total_counts_by_region = grouped_df.groupby('Sub_Region')['Count'].sum().reset_index()

# Display the total number of constituencies in each region
print("Total number of constituencies in each region:")
print(total_counts_by_region)

# Define the color mapping for parties
party_color_map = {
    'DMK': 'red',
    'ADMK': 'darkgreen',
    'PMK': 'yellow',
    'INC': 'lightgreen',
    'CPM': 'coral',
    'VCK': 'blue',
    'CPI': 'pink',
    'BJP': 'orange',
    'CPM': 'darkred'
}

# Create a stacked barplot for each subregion
for sub_region in df['Sub_Region'].unique():
    # Filter the DataFrame for the current subregion
    df_subregion = df_won[df_won['Sub_Region'] == sub_region]

    # Group the filtered DataFrame by 'Alliance' and 'Party', and count the occurrences
    grouped_df_subregion = df_subregion.groupby(['Alliance', 'Party']).size().reset_index(name='Count')

    # Create a stacked barplot
    fig = px.bar(grouped_df_subregion, x='Alliance', y='Count', color='Party',
                 title=f"DISTRIBUTION OF VICTORY IN {sub_region}",
                 labels={'Alliance': 'Alliance', 'Count': 'Count of won'},
                 hover_name='Party', barmode='stack',
                 color_discrete_map=party_color_map)  # Assign colors based on the mapping

    # Show the barplot
    fig.show()
```

```
Total number of constituencies in each region:
            Sub_Region  Count
0       CENTRAL REGION     50
1  CHENNAI CITY REGION     69
2      SOUTHERN REGION     58
3       WESTERN REGION     57
```

In [116…

```
#THIS IS A BARPLOT WITH STACKED BAR GRAPH WHICH TELLS US THE PARTIES USED WITH INTERACTIVE PLOTLY EXPRESS
# WE CAN SEE THAT SPA HAS A CLEAR VICTORY COUNT OVER NDA IN THE CHENNAI AND CENTRAL REGION TELLING US THAT SPA IS STRONG HERE
# WHEREAS IN WESTERN REGION WE CAN SEE THAT THERE IS A DEVIATION FROM ALL THE OTHER REGIONS WHERE NDA HAS MORE VICTORIES OVER
#SPA THUS MAKING THEM STRONG HERE AND EVENTHOUGH THERE WAS A TREND OBSERVED IN OTHER PLACE IT WASNT REFLECTED HERE
```

In [ ]:

```
```

THIRD POSITION

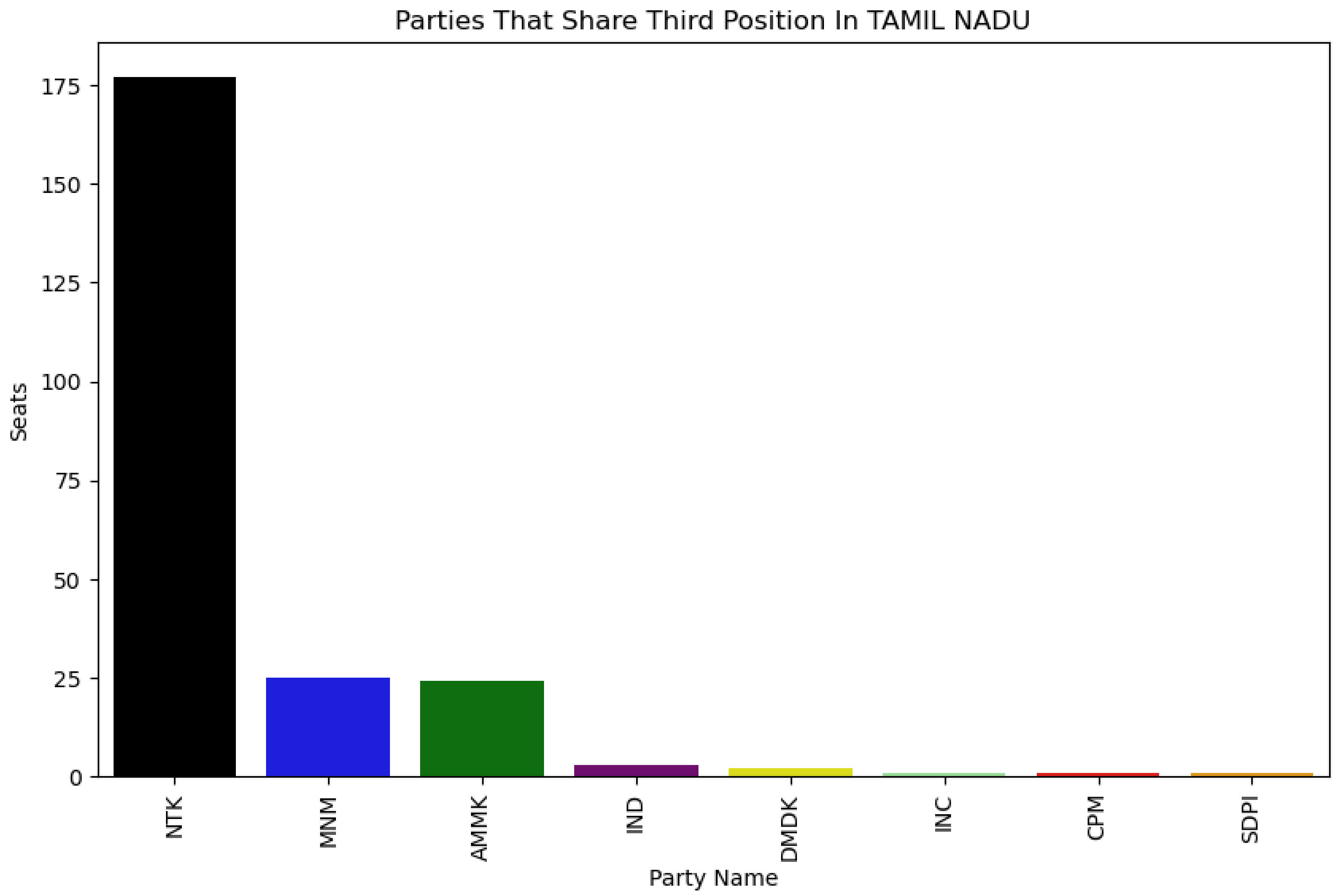
In [117…

```
# Filter the DataFrame for rows where position is 3
position_three_df = df[df['Position'] == 3]
position_three_df
# Calculate the count of unique alliances in the filtered DataFrame
alliance_count_position_three = position_three_df['Party'].nunique()

print("Count of alliances in position 3:", alliance_count_position_three)
```

```
Count of alliances in position 3: 8
```

In [118…

```
import seaborn as sns
import matplotlib.pyplot as plt

# Filter the DataFrame for rows where position is 3
position_three_df = df[df['Position'] == 3]

# Group the filtered DataFrame by 'Party' and count the occurrences
party_counts = position_three_df['Party'].value_counts()

# Define custom colors for each party
custom_palette = {
    'NTK': 'black',
    'MNM': 'blue',
    'AMMK': 'green',
    'IND': 'purple',
    'DMDK': 'yellow',
    'INC': 'lightgreen',
    'CPM': 'red',
    'SDPI': 'orange',

}

# Plot the counts of parties using Seaborn's barplot
plt.figure(figsize=(10, 6))
sns.barplot(x=party_counts.index, y=party_counts.values, palette=custom_palette)
plt.title('Parties That Share Third Position In TAMIL NADU')
plt.xlabel('Party Name')
plt.ylabel('Seats')
plt.xticks(rotation=90)
plt.show()
```



In [119…

```
#THIS IS A BAR PLOT WHICH IS USED WITH SEABORN
#AS WE CAN SEE THE THIRD PLACE IS GRABBED BY NTK IN MOST OF THE PLACES THAN ANYOTHER PARTY THUS MAKING IT A CONSISTENT THIRD
#PARTY IN TAMIL NADU WHICH HAS ALWAYS BEEN A TWO-PARTY STATE
# WE CAN ALSO SEE THAT WE HAVE INDEPENDENT CANDIDATES ALSO SECURED 3 PLACE THIS COULD BE BECAUSE OF THER POPULARITY AMONGST
#WHICH IS MUCH MORE THAN THAT OF THE SYMBOL WHICH IS THE SO CALLED IDENTITY OF EACH PARTY
```

In [120…

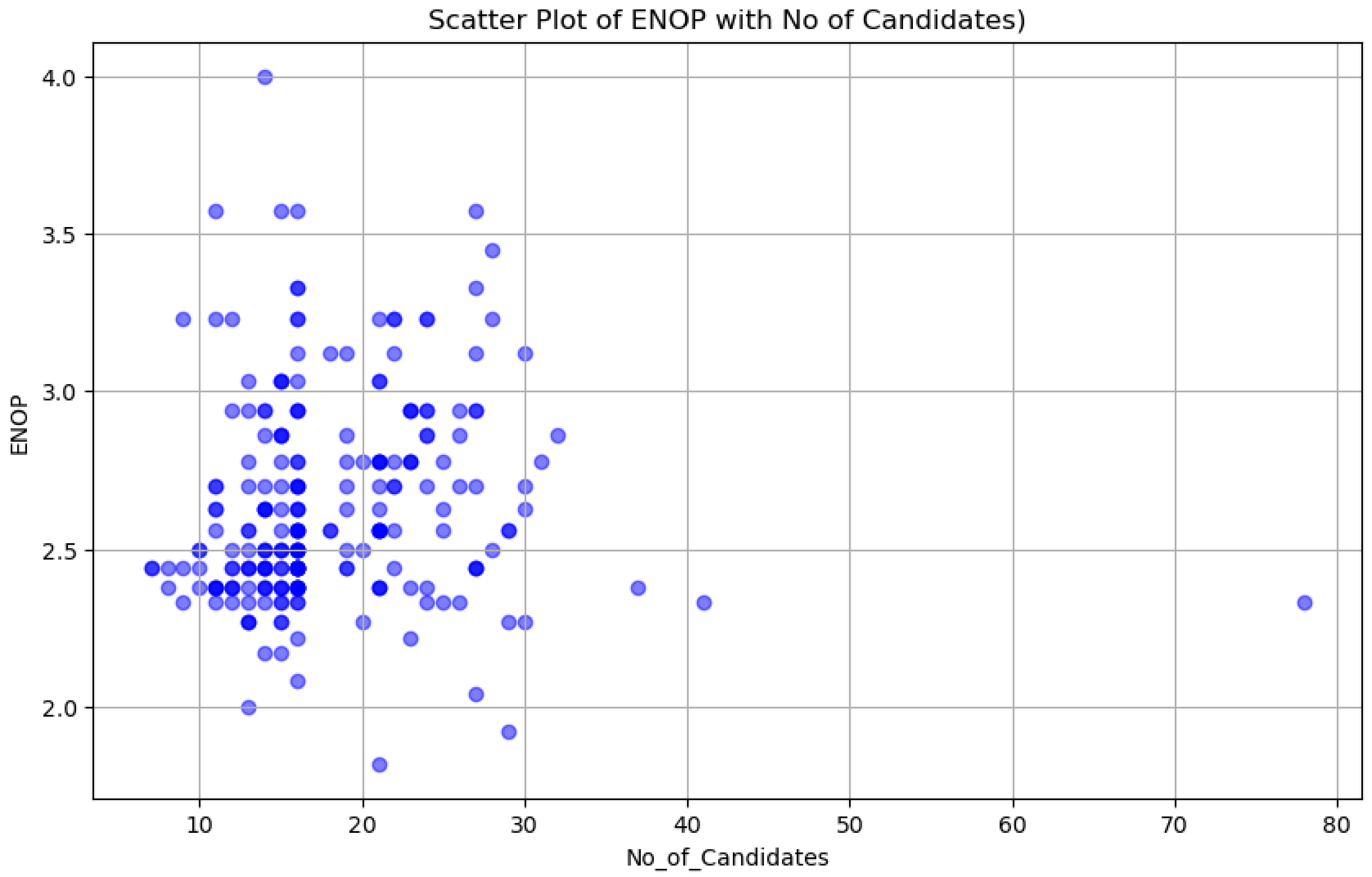
```
ind_position_three_df_selected = ind_position_three_df[['Constituency_No', 'Position', 'Candidate', 'Sex', 'Party', 'Votes', '
pd.DataFrame(ind_position_three_df_selected)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[120], line 1
----> 1 ind_position_three_df_selected = ind_position_three_df[['Constituency_No', 'Position', 'Candidate', 'Sex', 'Party', 'Vo
tes', 'Age', 'Valid_Votes', 'Constituency_Name','Contested', 'Vote_Share_Percentage', 'Last_Party', 'Same_Party']]
      2 pd.DataFrame(ind_position_three_df_selected)

NameError: name 'ind_position_three_df' is not defined
```

COMPARITIVE STUDY ON TWO NUMERIC VALUES NAMELY NO_OF_CANDIDATES AND ENOP

In [53]:

```
import matplotlib.pyplot as plt

# Filter the DataFrame for rows where position is 1
position_one_df = df[df['Position'] == 1]

# Plot scatter plot for ENOP with No_of_Candidates as x-axis
plt.figure(figsize=(10, 6))
plt.scatter(position_one_df['No_of_Candidates'], position_one_df['ENOP'], color='blue', alpha=0.5)
plt.title('Scatter Plot of ENOP with No of Candidates)')
plt.xlabel('No_of_Candidates')
plt.ylabel('ENOP')
plt.grid(True)
plt.show()
```



In [ ]:

```
#THIS IS A SCATTER PLOT USED WITH SEABORN WHICH CALCULATES THE TWO VALUES OF ENOP AND NO_OF_CANDIDATES FOR EACH CONSTITUTION
# AS WE CAN SEE THAT THERE CAN BE A LINEAR REGRESSION LINE THAT CAN BE DRAWN OVER THE POINTS SO USING INFERENTIAL STATISTICAL
#MODEL WE WILL BE ABLE TO PREDICT THE ENOP FOR A GIVEN NO_OF_CANDIDATES OR VICE-VERSA
# THERE ARE OUTLIER IN THE DATA WHERE IN A CONSTITUENCY THE NO OF CANDIDATES WHO STOOD ARE NEARLY 80 WHICH TOTALLY DEVIATES
#FROM THE GRAPH THUS MAKING IT AN OUTLIER FOR THE MODEL
```

TOTAL VOTES FOR EACH ALLIANCE

In [ ]:
```
import matplotlib.pyplot as plt

# Group the DataFrame by 'Alliance' and sum the 'Votes' column
alliance_votes = df.groupby('Alliance')['Votes'].sum().sort_values(ascending=False)

# Plot an area graph for total votes of each alliance
plt.figure(figsize=(10, 6))
alliance_votes.plot(kind='area', color='skyblue')
plt.title('Total Votes of Each Alliance')
plt.xlabel('Alliance')
plt.ylabel('Total Votes')
plt.xticks(rotation=90)
plt.grid(True)

plt.show()
```

In [ ]:
```
# Group the DataFrame by 'Alliance' and sum the 'Votes' column
alliance_votes = df.groupby('Alliance')['Votes'].sum().sort_values(ascending=False)

# Convert the series into a DataFrame
alliance_votes_df = alliance_votes.reset_index(name='Total_Votes')

# Display the DataFrame
print("Total Votes of Each Alliance:")
print(alliance_votes_df)
```

In [ ]:
```
#THIS IS AN AREA GRAPH WHICH GIVES US THE IDEA ABOUT THE VOTE DISTRIBUTION AMONG ALLIANCES WHICH IS USED WITH SEABORN
#AS WE CAN SEE THE GRAPH SPA HAVE GOT THE MOST VOTES BUT WE CAN ALSO SEE THAT THE VOTE DIFFERENCE BETWEEN THEM IS AROUND 20
#LAKHS WHICH IS SMALL COMPARED TO THE POPULATION SIZE
#WE CAN ALSO INFER THAT THE NTK THE THIRD HIGHEST VOTES PARTY AFTER THE FIRST TWO HAVE VOTES DRASTICALLY LESS THAN THE OTHER T
#WHICH STATES THAT PEOPLE OF TN WERE IN A MINDSET TO VOTE EITHER THIS(NDA) OR THAT(SPA) ALLIANCE MAJORITARILY
```

STUDY BETWEEN TURNOUT PERCENTAGE AND CONSTITUENCY TYPE

In [ ]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Create a boxplot of turnout percentage for each constituency type
plt.figure(figsize=(10, 6))
sns.boxplot(x='Constituency_Type', y='Turnout_Percentage', data=df,color='red')
sns.swarmplot(x='Constituency_Type', y='Turnout_Percentage', data=df, size=2)  # Adjust size here
plt.title('Boxplot of Turnout Percentage by Constituency Type')
plt.xlabel('Constituency Type')
plt.ylabel('Turnout Percentage')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

In [ ]:
```python
# THIS IS A SWARMPLOT ON TOP OF A BOX PLOT GIVING US MORE INSIGHTS ABOUT THE TWO ATTRIBUTES USED WITH SEABORN.
#WE CAN INFER THAT THERE IS A LARGER TURNOUT COUNT PERCENTAGE IN SC AND ST CONSTITUENCIES SINCE THESE CONSTITUENCIES ARE OFTEN
#ARE HOTSPOTS FOR POLICITIANS AND ARE FOCUSSED MORE IN CAMPAIGN WHICH MIGHT HAVE RESULTED TO A HIGHER TURNOUT PERCENTAGE.
#THERE DATA ARE DISTRIBUTED IN A SAME MANNER WITH THE IQR RANGING FROM 67 TO 77 APPROXIMATELY IN GEN AND 73 TO 77 IN SC BUT
#IS VERY SMALL FOR ST SINCE THERE ARE ONLY 2 CONSTITUENCIES AND THEY HAVE GOT SIMILAR TURNOUT PERCENTAGE
```

CURRENT_STATUS OF IMPORTANT CANDIDATES

In [ ]:
```python
import plotly.express as px
party_color_map = {
    'DMK': 'red',
    'ADMK': 'darkgreen',
    'PMK': 'yellow',
    'INC': 'lightgreen',
    'CPM': 'coral',
    'VCK': 'blue',
    'CPI': 'pink',
    'BJP': 'orange',
    'CPM': 'darkred',
    'AMMK': 'green'
}

# Filter the DataFrame for the specified Current_Status categories
filtered_df = df[df['Current_Status'].isin(['Position gained', 'Consecutive win', 'Ex-MLA'])]

# Group the filtered DataFrame by 'Current_Status' and 'Party', and count the occurrences
grouped_df = filtered_df.groupby(['Current_Status', 'Party']).size().reset_index(name='Count')

# Create an interactive stacked barplot
fig = px.bar(grouped_df, x='Current_Status', y='Count', color='Party',
             title='Count of Current Status by Party',
             labels={'Count': 'Count of Current Status'},
             hover_name='Party', barmode='stack',color_discrete_map=party_color_map )

# Show the interactive stacked barplot
fig.show()
```

In [ ]:

```
#THIS IS A STACKED BAR PLOT WHICH USES PARTY AS STACKING AND TELLS US THE WHAT IS THE CURRENT STATUS IMPORTANT MEMBER THEY ARE
#IMPORTANT SINCE THEY HAVE ATLEAST BEEN IN LEGISLATIVE ASSEMBLY ONCE.
#AS WE CAN SEE THAT MOST OF THE LAST TIME WINNERS OF ADMK PARTY HAVE BEEN REJECTED SO WE CAN INFER PEOPLE HAVE CHANGED THEIR
#MIND.
# WE CAN ALSO INFER THAT THERE ARE A LOT OF NEW PARTIES WHICH HAVE MEMBERS WHO HAVE GAINED POSITION NAMELY VCK,PMK,CPI,CPM,BJP
#WHICH IS APPRECIATABLE SINCE THERE WILL BE VOICE OF PEOPLE FROM ALL GROUPS OF SOCIETY.
```

A STUDY ON DEPOSIT WITH THE THIRD POSITION

In [340…

```
# Filter the DataFrame for rows where position is 3 and deposit lost is 'no'
filtered_df = df[(df['Position'] == 3) & (df['Deposit_Lost'] == 'no')]

# Display the filtered DataFrame
pd.DataFrame(filtered_df)
```

Out[340…

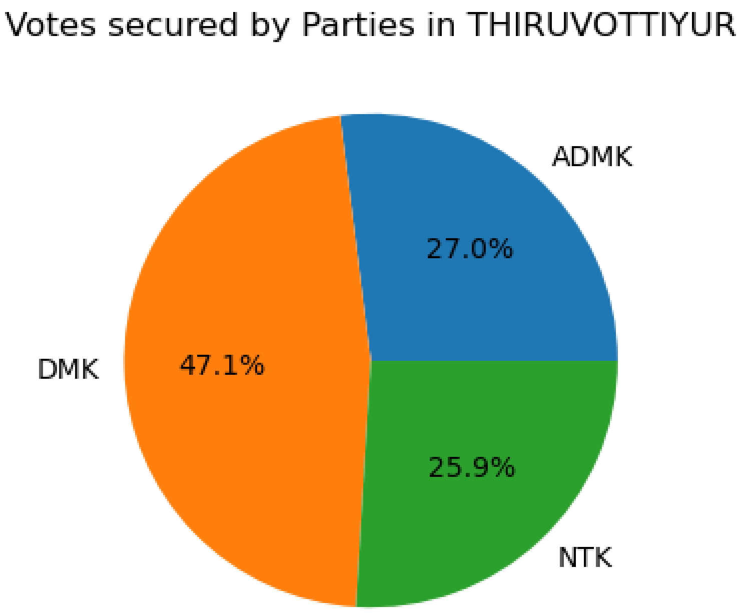
	Constituency_No	Position	Candidate	Sex	Party	Votes	Age	Valid_Votes	Electors	Constituency_Name	...	Second_Professio
155	10	3	SEEMAN	M	NTK	48597	55.0	199996	306977	THIRUVOTTIYUR	...	Nal
2174	120	3	MAYURA S JAYAKUMAR	M	INC	42383	47.0	154765	255009	COIMBATORE SOUTH	...	Nal
2196	121	3	Mahendran, R	M	MNM	36855	58.0	202021	327265	SINGANALLUR	...	Liberal Professio or Professiona
3045	167	3	KAMARAJ S	M	AMMK	40481	54.0	193229	259926	MANNARGUDI	...	Agricultur
3126	173	3	KARTHIKEYAN, V.	M	AMMK	37469	44.0	211430	270682	THIRUVAIYARU	...	Nal
3301	184	3	Dherpoki V Pandi	M	AMMK	44864	45.0	212456	320785	KARAIKUDI	...	Agricultur
3372	188	3	SELVARAJ A.	M	AMMK	34262	57.0	182782	246237	MELUR	...	Nal
3535	197	3	MAHENDRAN I	M	AMMK	55491	55.0	212513	287879	USILAMPATTI	...	Nal
3659	204	3	RAJAVARMAN, M.S.R.	M	AMMK	32916	48.0	191783	255132	SATTUR	...	Agricultur
3959	218	3	K.Srinivasan	M	CPM	37380	57.0	180928	268319	KOVILPATTI	...	Nal
4055	223	3	HARI NADAR.A	M	IND	37727	39.0	203478	262891	ALANGULAM	...	Social Wor

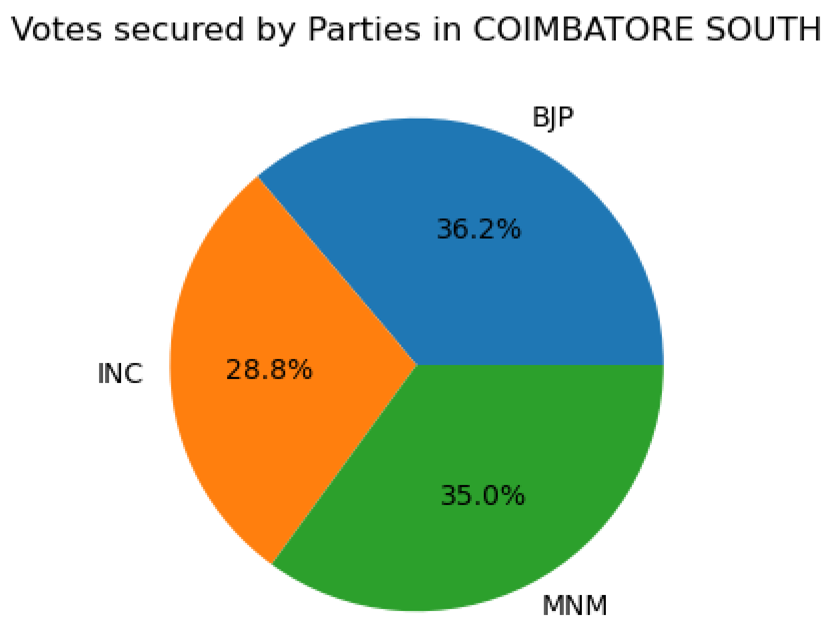
11 rows × 45 columns

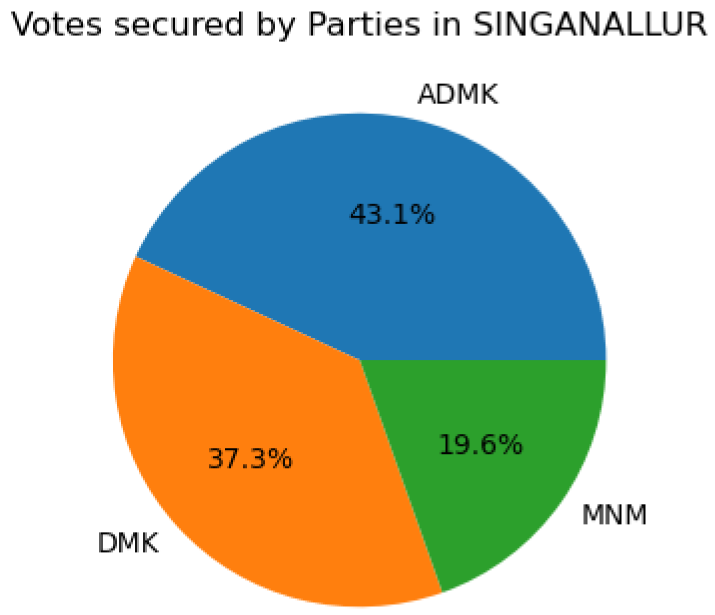
In [202…

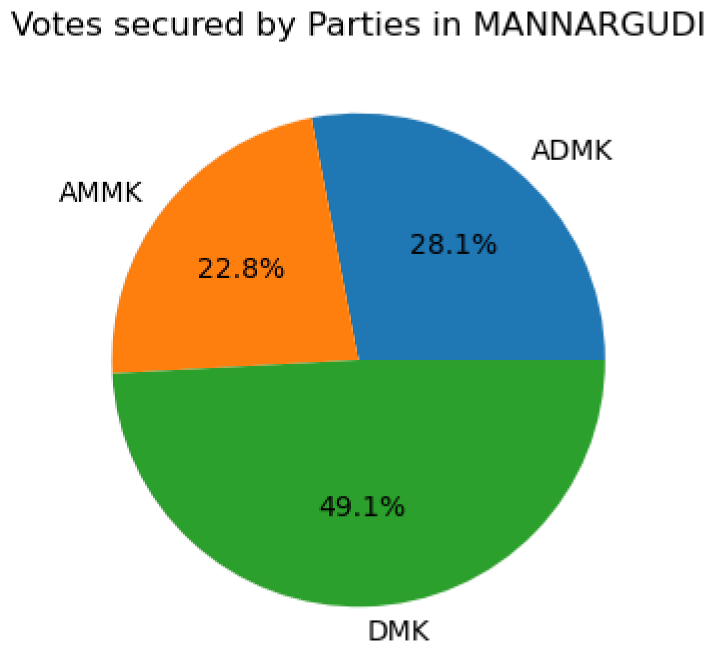
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Define the list of constituencies in order
constituency_names = ['THIRUVOTTIYUR', 'COIMBATORE SOUTH', 'SINGANALLUR', 'MANNARGUDI', 'THIRUVAIYARU',
                       'KARAIKUDI', 'MELUR', 'USILAMPATTI', 'SATTUR', 'KOVILPATTI', 'ALANGULAM']
```

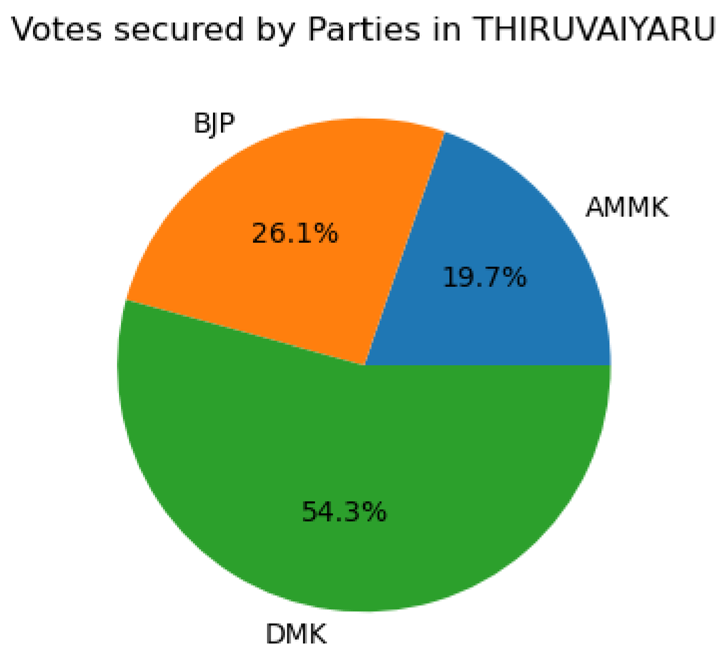
```python
# Filter the DataFrame for the specified constituencies and positions
filtered_df = df[df['Constituency_Name'].isin(constituency_names)]
filtered_df = filtered_df[filtered_df['Position'].isin([1, 2, 3])]

# Create separate pie plots for each constituency
for constituency_name in constituency_names:
    # Filter the DataFrame for the current constituency name
    constituency_df = filtered_df[filtered_df['Constituency_Name'] == constituency_name]

    # Group the filtered DataFrame by 'Party' and sum the votes for each position
    grouped_df = constituency_df.groupby('Party')['Votes'].sum().reset_index()

    # Plot a pie plot for the current constituency
    plt.figure(figsize=(4, 4))
    plt.pie(grouped_df['Votes'], labels=grouped_df['Party'], autopct='%1.1f%%')
    plt.title(f'Votes secured by Parties in {constituency_name}')
    plt.show()
```

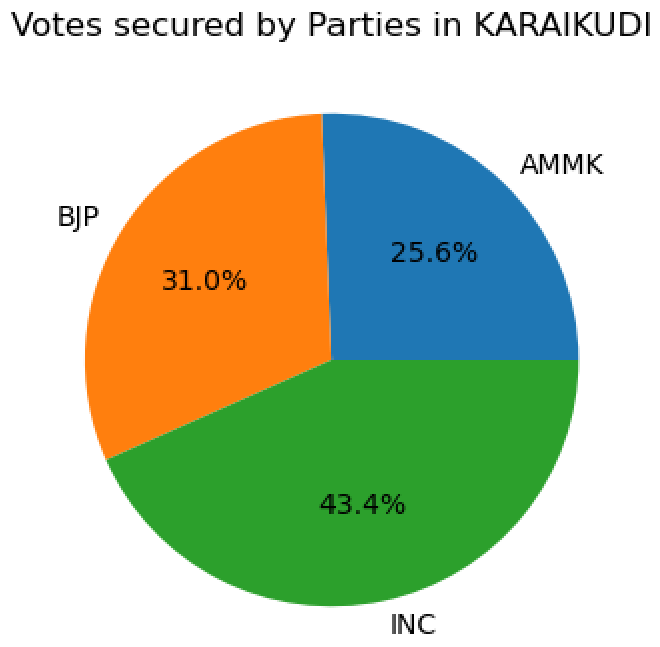


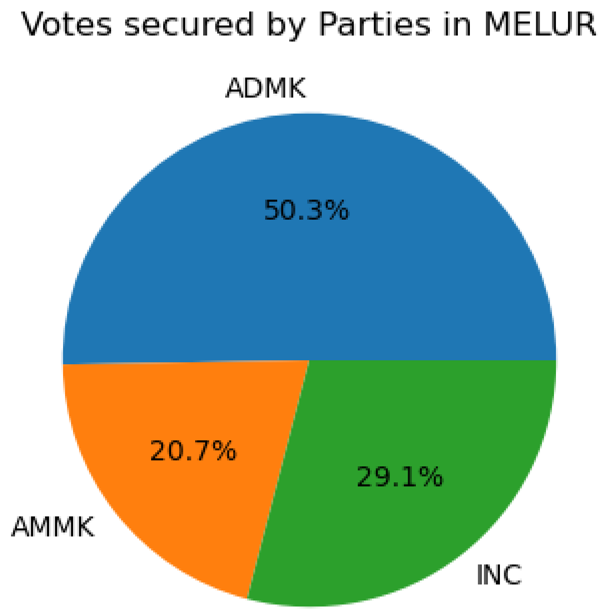

Votes secured by Parties in COIMBATORE SOUTH
BJP
36.2%
INC
28.8%
35.0%
MNM

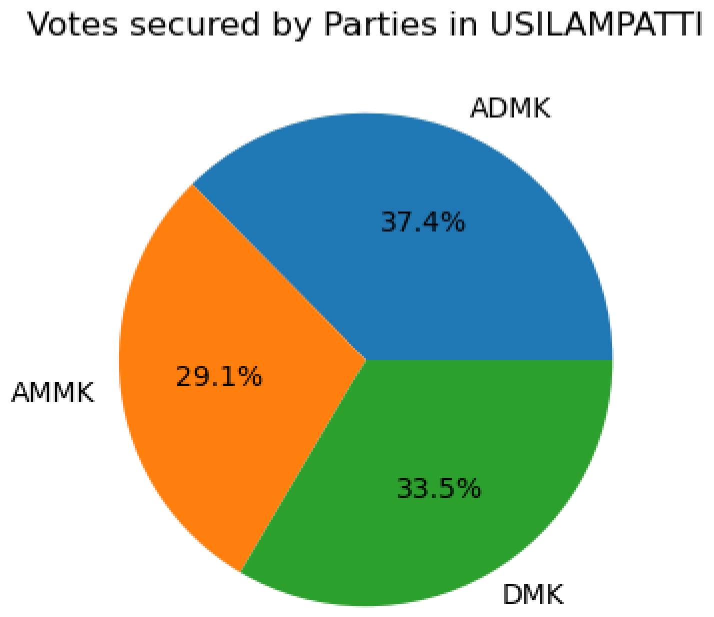

Votes secured by Parties in SINGANALLUR
ADMK
43.1%
DMK
37.3%
MNM
19.6%

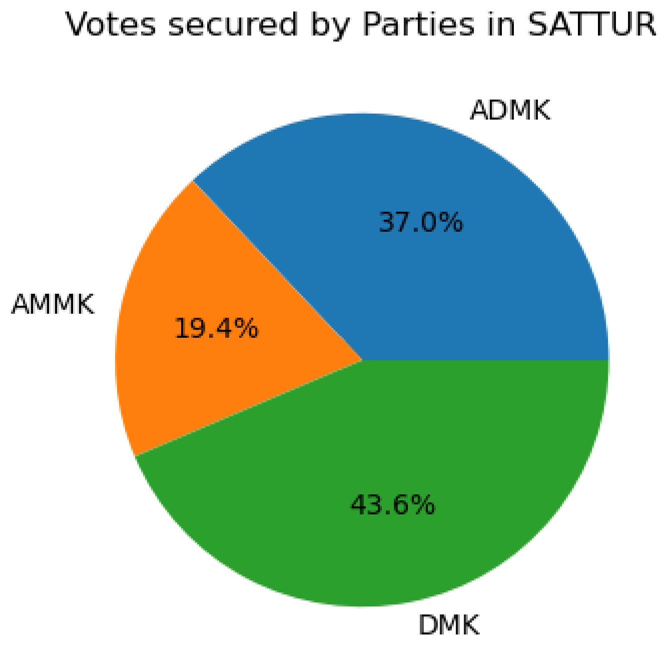

Votes secured by Parties in MANNARGUDI
ADMK
AMMK
DMK
28.1%
22.8%
49.1%

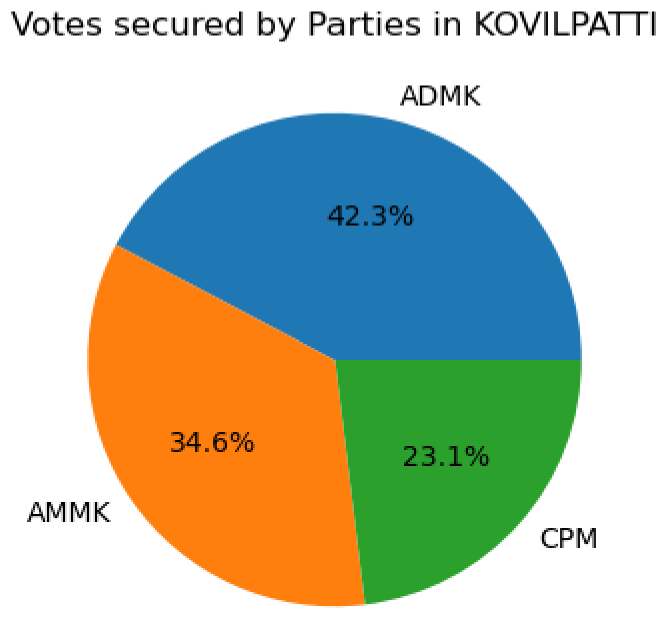

Votes secured by Parties in THIRUVAIYARU
BJP
26.1%
AMMK
19.7%
54.3%
DMK

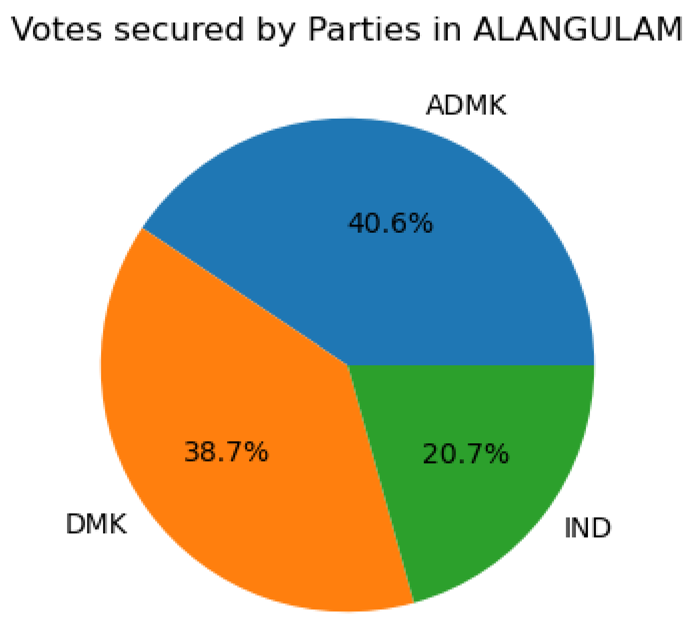

Votes secured by Parties in KARAIKUDI
AMMK
25.6%
BJP
31.0%
43.4%
INC


Votes secured by Parties in MELUR
ADMK
50.3%
AMMK
20.7%
INC
29.1%


Votes secured by Parties in USILAMPATTI
ADMK
37.4%
AMMK
29.1%
33.5%
DMK


Votes secured by Parties in SATTUR
ADMK
37.0%
AMMK
19.4%
43.6%
DMK


Votes secured by Parties in KOVILPATTI
ADMK
42.3%
34.6%
AMMK
23.1%
CPM

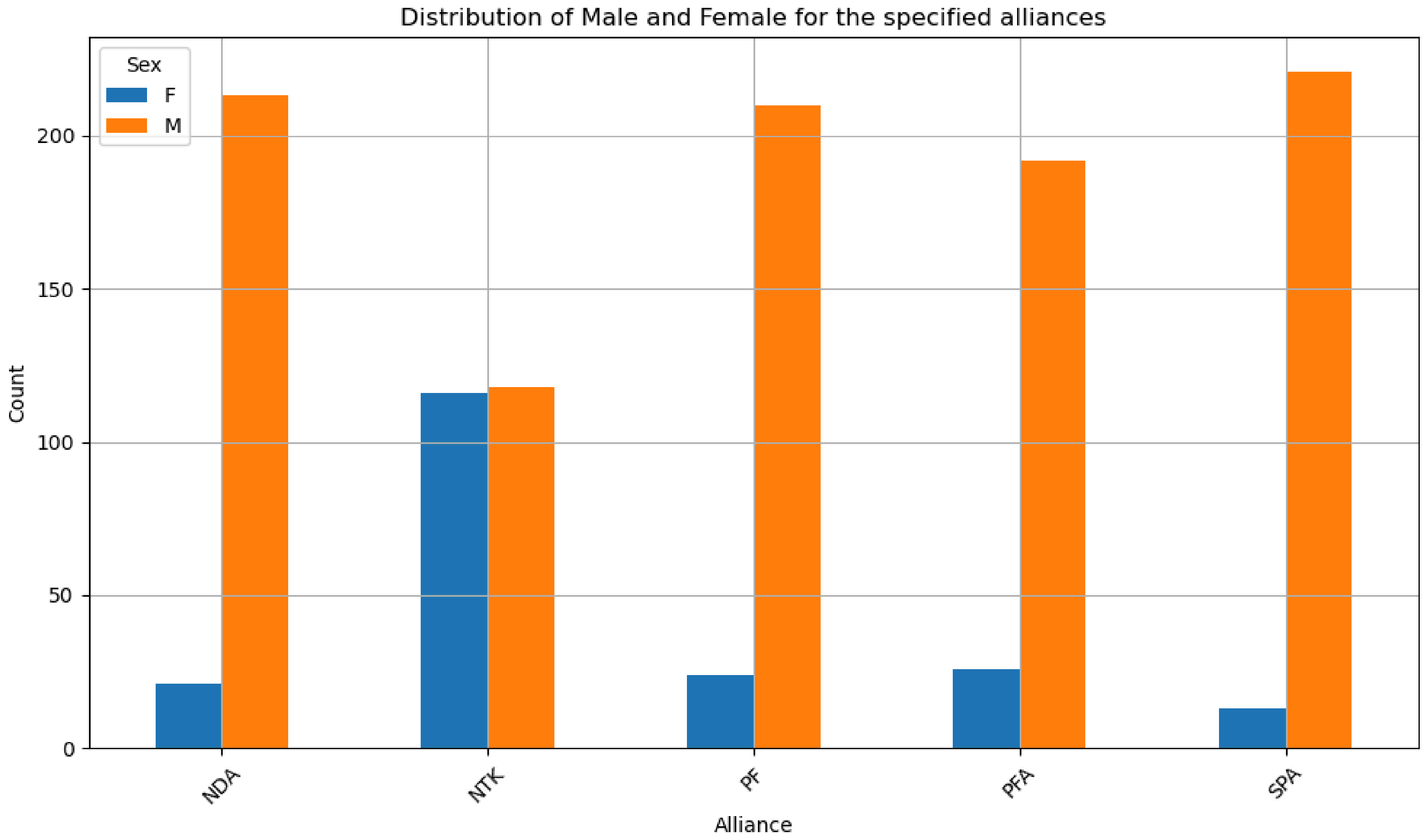


In [ ]:
```
# THESE ARE THE PIE CHARTS DISPLAYING THE VOTES SECURED BY PARTIES IN SELECTED CONSTITUENCIES BY USING SEABORN AND MATPLOTLIB
# WE CAN SEE THAT MOST OF THE CHARTS HAVE A CLOSE CALL BETWEEN THE THREE CANDIDATES THIS IS NOT COMMONLY SEEN THUS IT IS
#GENERALLY SEEN AS A PRIDE TO ATLEAST GET THE DEPOSIT
# WE CAN ALSO SEE THAT IT IS ALWAYS SPA AND NDA IN THE PIE THUS RESTATING THE FACT THAT TN PEOPLE HAVE A MINDSET OF THIS OR TH
```

DISTRIBUTION OF MALE AND FEMALE IN ALLIANCES

In [203…
```
import pandas as pd
import matplotlib.pyplot as plt

alliance_list = ['SPA', 'NDA', 'NTK', 'PF', 'PFA']

# Filter the DataFrame for the specified list of alliances
filtered_df = df[df['Alliance'].isin(alliance_list)]

# Group the filtered DataFrame by both 'Alliance' and 'Sex' columns and count the occurrences
gender_counts = filtered_df.groupby(['Alliance', 'Sex']).size().unstack()
```

```
# Fill missing values with 0
gender_counts.fillna(0, inplace=True)

# Plot the grouped bar plot
gender_counts.plot(kind='bar', figsize=(10, 6))

# Add labels and title
plt.title('Distribution of Male and Female Alliances')
plt.xlabel('Alliance')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend(title='Sex')
plt.grid(True)

# Show plot
plt.tight_layout()
plt.show()
```

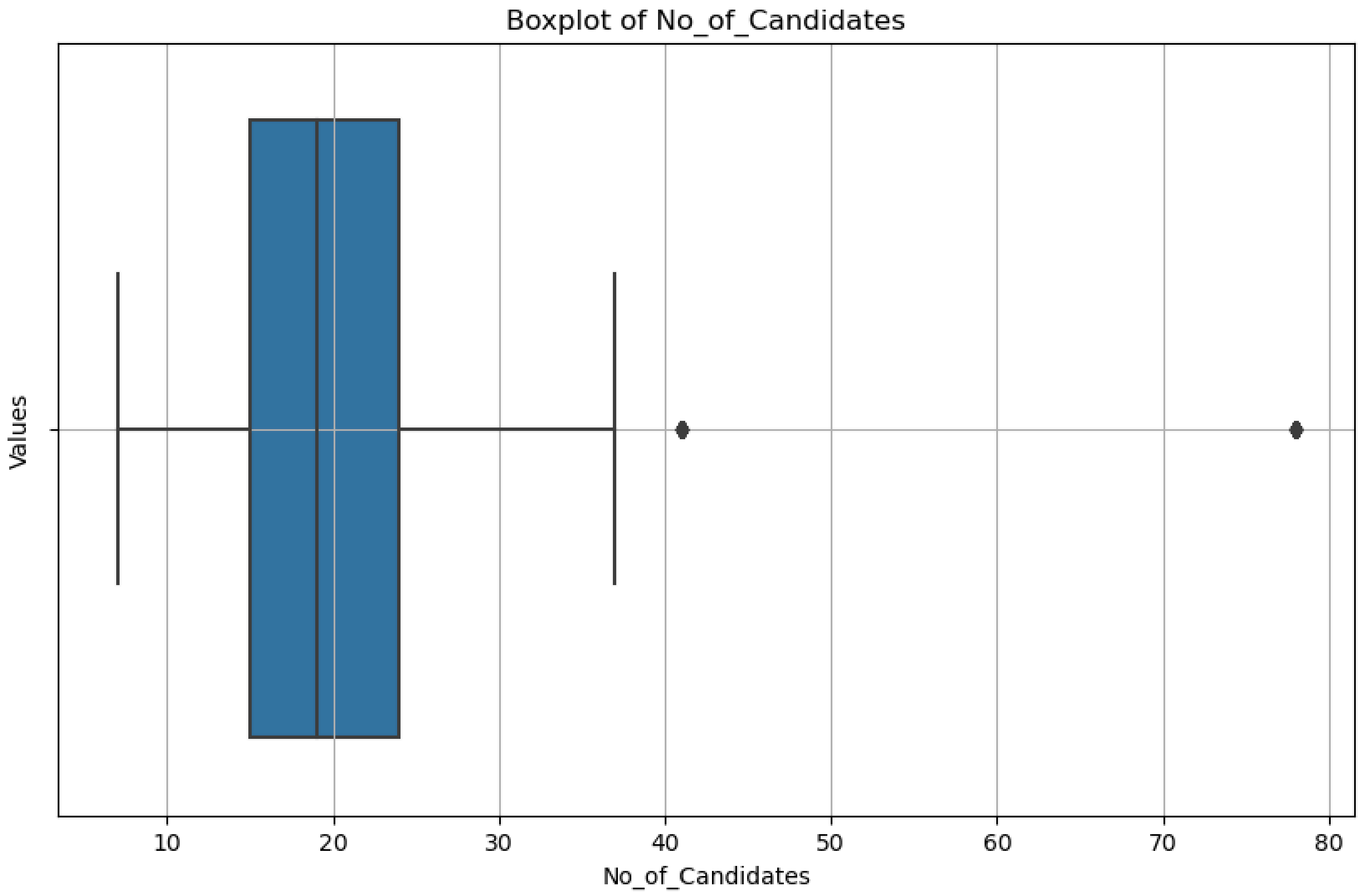


```
In [ ]: # THIS IS A GROUPED BAR PLOT DONE WITH SEABORN AND MATPLOTLIB
        #WE CAN SEE THAT NTK ALLIANCE HAS DISTRIBUTED ITS SEATS WITH MEN AND WOMEN EQUALLY WHICH IS ACTUALLY A FORWARD MOVEMENT IN
        #ELECTION HISTORY
        #WE CAN ALSO SEE THAT MOSTLY ALL OTHER PARTIES ARE BIASED TOWARDS MALE CANDIDATES AND OFFER SEAT ONLY TO THEM
```

Analysis on number of candidates

In [204…

```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a boxplot for No_of_Candidates
plt.figure(figsize=(10, 6))
sns.boxplot(x='No_of_Candidates', data=df)
plt.title('Boxplot of Number of Candidates')
plt.xlabel('No of Candidates')
plt.ylabel('Values')
plt.grid(True)
plt.show()
```



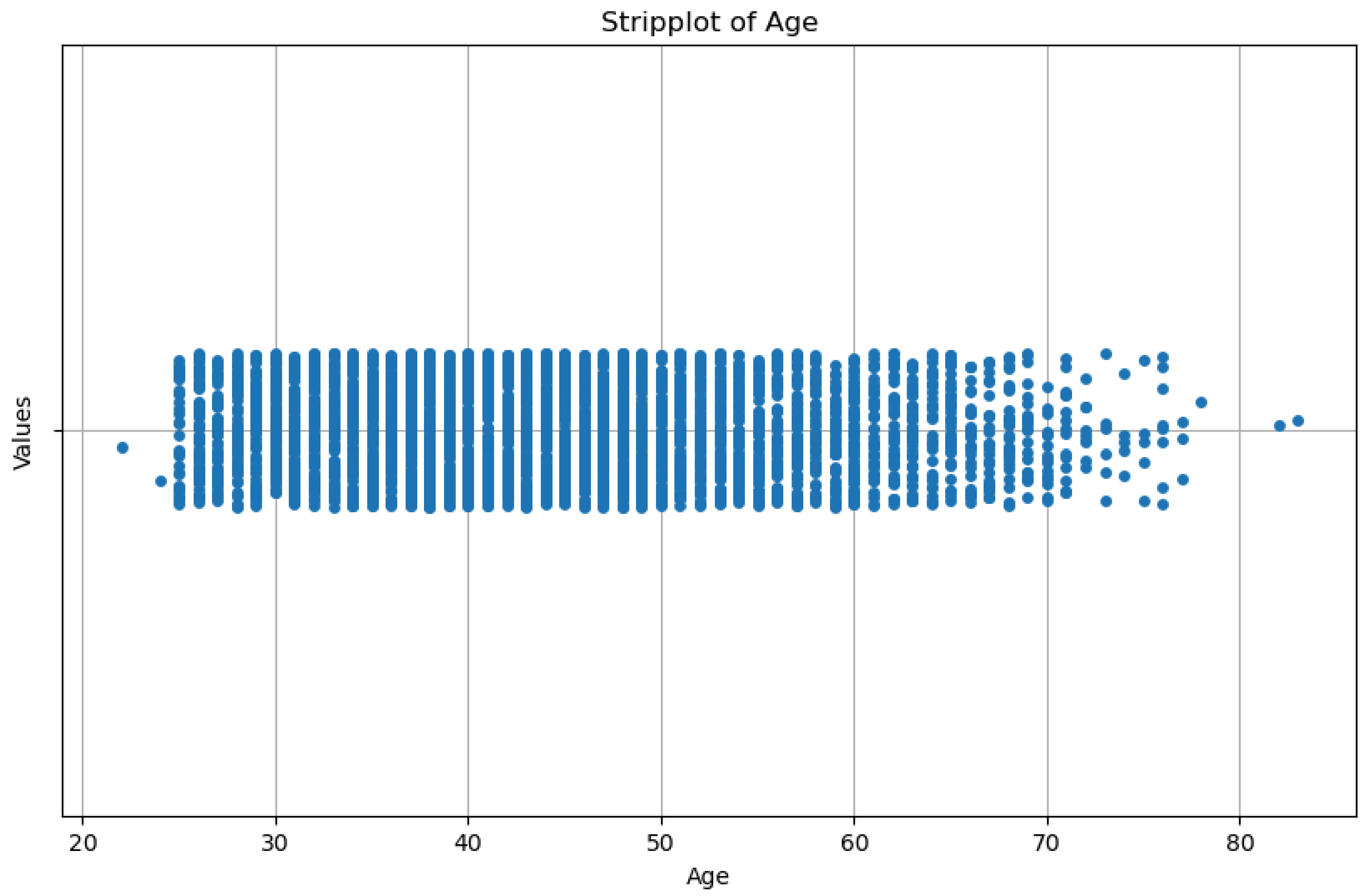
In [ ]:
```
#it is a univariate analysis which is done with Number of candidates in each constituency using seaborn and matplotlib
#we can see that the iqr ranges between 15 to 25 approximately which is relatively good since people have many choices in fron
#of them they will be able to choose
# we can also see outliers of a value just less than 80 which is present
```

In [ ]:

Analysis on age

In [205…

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Create a swarmplot for the Age column
plt.figure(figsize=(10, 6))
sns.stripplot(x='Age', data=df)
plt.title('Stripplot of Age')
plt.xlabel('Age')
plt.ylabel('Values')
plt.grid(True)
plt.show()
```



In [ ]:
```
#We can see that the age ranges from a wide range of 25 to 65 uniformaly which is considered to be the working period of anyon
#in india
# We can also see that there are a minimal number of points after 70 we can say that candidates might think it isnt suitable
#for them after that
```

In [2]:
```
import pandas as pd
```

```
df=pd.read_csv("tnelectionsformlmodels.csv")
```

In [3]:
```
df
```

Out[3]:

	Unnamed: 0	Constituency_No	Position	Candidate	Sex	Party	Votes	Age	Valid_Votes	Electors	...	Second_Profession
0	0	1	1	GOVINDARAJAN T.J	M	DMK	126452	60.0	222069	284412	...	NaN
1	1	1	2	PRAKASH M	M	PMK	75514	50.0	222069	284412	...	NaN
2	2	1	3	USHA	F	NTK	11701	31.0	222069	284412	...	NaN
3	3	1	4	DILLIY K.M	M	DMDK	2576	45.0	222069	284412	...	NaN
4	4	1	5	None Of The Above	NaN	NOTA	1783	NaN	222069	284412	...	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
4227	4227	234	11	JOHN BENADICT G	M	IND	235	42.0	169918	257959	...	NaN
4228	4228	234	12	ANCHALOSE R	M	IND	213	61.0	169918	257959	...	NaN
4229	4229	234	13	SHAJU SINGH M R	M	IND	213	45.0	169918	257959	...	NaN
4230	4230	234	14	VIJIKUMAR K	M	National Democratic Party of South India	186	34.0	169918	257959	...	NaN
4231	4231	234	15	THANKAPPAN C	M	Tamilnadu Mahatma Gandhi Makkal Katchi	162	72.0	169918	257959	...	NaN

4232 rows × 46 columns

In [20]:
```
import numpy as np
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from scipy.stats import linregress

df = df[df['Position'] == 1]

# Extract Electors and Valid_Votes columns
electors = df['Electors']
valid_votes = df['Valid_Votes']

# Create scatter plot with regression line using Plotly Express
fig = px.scatter(x=electors, y=valid_votes, trendline="ols", labels={'x': 'Electors', 'y': 'Valid Votes'},
                 title='Scatter Plot of Electors vs Valid Votes with Regression Line')
fig.show()
```

In [19]:
```
# Calculate linear regression
slope, intercept, r_value, p_value, std_err = linregress(valid_votes, elector)

# Calculate R-squared
r_squared = r_value ** 2

# Calculate number of observations and predictors
n = len(elector)
```

```
k = 1  # We have only one predictor in this case (ENOP)

# Calculate adjusted R-squared
r_adj_squared = 1 - ((1 - r_squared) * (n - 1) / (n - k - 1))

a=r_squared
b=r_adj_squared

print(f"R-squared:",a)
print(f"Adjusted R-squared:",b)
```

```
R-squared: 0.7143546524877135
Adjusted R-squared: 0.7131180925417728
```

In [21]:

```
import numpy as np
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from scipy.stats import linregress
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression


X = df[['Electors']]
y = df['Valid_Votes']


# Creating and fitting linear regression model
linear_reg = LinearRegression()
param_grid = {
    'fit_intercept': [True, False],
    'positive': [True, False]
}
grid_search = GridSearchCV(estimator=linear_reg, param_grid=param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search.fit(X, y)

c=grid_search.best_estimator_.score(X, y)

# Printing best parameters and R2 score of test
print(grid_search.best_params_)
print('R2 Score of Test is :', c)
```

```
# Extracting slope and intercept from the best estimator
best_slope = grid_search.best_estimator_.coef_[0]
best_intercept = grid_search.best_estimator_.intercept_

c=grid_search.best_estimator_.score(X, y)
```

```
{'fit_intercept': True, 'positive': True}
R2 Score of Test is : 0.7143546524877136
```

In [22]:

```
import plotly.graph_objects as go

# Data for bar graph
labels = ['R-squared', 'Adjusted R-squared', 'R2 Score of Test']
values = [a, b, c]

# Create bar graph using Plotly
fig = go.Figure([go.Bar(x=labels, y=values)])

# Update layout
fig.update_layout(title='Comparison of R-squared, Adjusted R-squared, and R2 Score',
                  xaxis_title='Metrics',
                  yaxis_title='Value')

# Show plot
fig.show()
```

In [23]:
```
import numpy as np
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from scipy.stats import linregress
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
```

```
# Assuming df is your DataFrame containing 'Electors' and 'Valid_Votes' columns

# Splitting data into training and testing sets
X = df[['Electors']]
y = df['Valid_Votes']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=25)

# Creating and fitting Decision Tree Regressor model
decision_tree_reg = DecisionTreeRegressor()
decision_tree_reg.fit(X_train, y_train)

# Calculate R2 Score of Test
c = decision_tree_reg.score(X_test, y_test)

# Calculate linear regression
slope, intercept, r_value, p_value, std_err = linregress(valid_votes, electors)

# Calculate R-squared
r_squared = r_value ** 2

# Calculate number of observations and predictors
n = len(electors)
k = 1  # We have only one predictor in this case (Electors)

# Calculate adjusted R-squared
r_adj_squared = 1 - ((1 - r_squared) * (n - 1) / (n - k - 1))

# Data for bar graph
labels = ['R-squared', 'Adjusted R-squared', 'R2 Score of Test', 'Decision Tree R2 Score']
values = [r_squared, r_adj_squared, c, c]

# Create bar graph using Plotly
fig = go.Figure([go.Bar(x=labels, y=values)])

# Update layout
fig.update_layout(title='Comparison of R-squared, Adjusted R-squared, and R2 Score between Linear Regression and Decision Tree
                  xaxis_title='Metrics',
                  yaxis_title='Value')
```

```
# Show plot
fig.show()
```

In [24]:
```
import numpy as np
import pandas as pd
import plotly.graph_objects as go
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

```python
from sklearn.model_selection import train_test_split

# Assuming df is your DataFrame containing 'Electors' and 'Valid_Votes' columns

# Splitting data into training and testing sets
X = df[['Electors']]
y = df['Valid_Votes']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=25)

# Creating Polynomial Features
poly = PolynomialFeatures(degree=2)  # You can change the degree as per your requirement
X_poly_train = poly.fit_transform(X_train)
X_poly_test = poly.transform(X_test)

# Creating and fitting Polynomial Regression model
poly_reg = LinearRegression()
poly_reg.fit(X_poly_train, y_train)

# Calculate R2 Score of Test
c = poly_reg.score(X_poly_test, y_test)

# Calculate linear regression
slope, intercept, r_value, p_value, std_err = linregress(valid_votes, electors)

# Calculate R-squared
r_squared = r_value ** 2

# Calculate number of observations and predictors
n = len(electors)
k = 1  # We have only one predictor in this case (Electors)

# Calculate adjusted R-squared
r_adj_squared = 1 - ((1 - r_squared) * (n - 1) / (n - k - 1))

# Data for bar graph
labels = ['R-squared', 'Adjusted R-squared', 'R2 Score of Test', 'Polynomial Regression R2 Score']
values = [r_squared, r_adj_squared, c, c]

# Create bar graph using Plotly
fig = go.Figure([go.Bar(x=labels, y=values)])
```

```
# Update layout
fig.update_layout(title='Comparison of R-squared, Adjusted R-squared, and R2 Score between Linear Regression and Polynomial Re
                  xaxis_title='Metrics',
                  yaxis_title='Value')

# Show plot
fig.show()
```

```
In [ ]:
```