In [ ]:
```
Ravi Thange
ID::IN9240186
```

In [2]:
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

import pandas as pd
df=pd.read_csv("C:\\Users\\Prime\\Pictures\\EV.csv")
```

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

Out[3]:

| | VIN (1-10) | County | City | State | Postal Code | Model Year | Make | Model | Electric Vehicle Type | Al I |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | JTMEB3FV6N | Monroe | Key West | FL | 33040 | 2022 | TOYOTA | RAV4 PRIME | Plug-in Hybrid Electric Vehicle (PHEV) | A Fu |
| **1** | 1G1RD6E45D | Clark | Laughlin | NV | 89029 | 2013 | CHEVROLET | VOLT | Plug-in Hybrid Electric Vehicle (PHEV) | A Fu |
| **2** | JN1AZ0CP8B | Yakima | Yakima | WA | 98901 | 2011 | NISSAN | LEAF | Battery Electric Vehicle (BEV) | A Fu |
| **3** | 1G1FW6S08H | Skagit | Concrete | WA | 98237 | 2017 | CHEVROLET | BOLT EV | Battery Electric Vehicle (BEV) | A Fu |
| **4** | 3FA6P0SU1K | Snohomish | Everett | WA | 98201 | 2019 | FORD | FUSION | Plug-in Hybrid Electric Vehicle (PHEV) | N d |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **112629** | 7SAYGDEF2N | King | Duvall | WA | 98019 | 2022 | TESLA | MODEL Y | Battery Electric Vehicle (BEV) | |
| **112630** | 1N4BZ1CP7K | San Juan | Friday Harbor | WA | 98250 | 2019 | NISSAN | LEAF | Battery Electric Vehicle (BEV) | A Fu |
| **112631** | 1FMCU0KZ4N | King | Vashon | WA | 98070 | 2022 | FORD | ESCAPE | Plug-in Hybrid Electric Vehicle (PHEV) | A Fu |
| **112632** | KNDCD3LD4J | King | Covington | WA | 98042 | 2018 | KIA | NIRO | Plug-in Hybrid Electric Vehicle (PHEV) | N d |

| | VIN (1-10) | County | City | State | Postal Code | Model Year | Make | Model | Electric Vehicle Type | Al |
|---|---|---|---|---|---|---|---|---|---|---|
| **112633** | YV4BR0CL8N | King | Covington | WA | 98042 | 2022 | VOLVO | XC90 | Plug-in Hybrid Electric Vehicle (PHEV) | N d |

112634 rows × 17 columns

In [4]:
```
df.head()
```

Out[4]:

| | VIN (1-10) | County | City | State | Postal Code | Model Year | Make | Model | Electric Vehicle Type | Clea Alternativ Fue Vehicl (CAFV Eligibilit |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | JTMEB3FV6N | Monroe | Key West | FL | 33040 | 2022 | TOYOTA | RAV4 PRIME | Plug-in Hybrid Electric Vehicle (PHEV) | Clea Alternativ Fuel Vehicl Eligibl |
| **1** | 1G1RD6E45D | Clark | Laughlin | NV | 89029 | 2013 | CHEVROLET | VOLT | Plug-in Hybrid Electric Vehicle (PHEV) | Clea Alternativ Fuel Vehicl Eligibl |
| **2** | JN1AZ0CP8B | Yakima | Yakima | WA | 98901 | 2011 | NISSAN | LEAF | Battery Electric Vehicle (BEV) | Clea Alternativ Fuel Vehicl Eligibl |
| **3** | 1G1FW6S08H | Skagit | Concrete | WA | 98237 | 2017 | CHEVROLET | BOLT EV | Battery Electric Vehicle (BEV) | Clea Alternativ Fuel Vehicl Eligibl |
| **4** | 3FA6P0SU1K | Snohomish | Everett | WA | 98201 | 2019 | FORD | FUSION | Plug-in Hybrid Electric Vehicle (PHEV) | Not eligibl due to lov batter rang |

In [5]:
```
df.tail()
```

Out[5]:

| | VIN (1-10) | County | City | State | Postal Code | Model Year | Make | Model | Electric Vehicle Type | Clea Alternativ Fu Vehicl (CAFV Eligibilit |
|---|---|---|---|---|---|---|---|---|---|---|
| **112629** | 7SAYGDEF2N | King | Duvall | WA | 98019 | 2022 | TESLA | MODEL Y | Battery Electric Vehicle (BEV) | Eligibili unknow as batter range ha not b |
| **112630** | 1N4BZ1CP7K | San Juan | Friday Harbor | WA | 98250 | 2019 | NISSAN | LEAF | Battery Electric Vehicle (BEV) | Clea Alternativ Fuel Vehic Eligib |
| **112631** | 1FMCU0KZ4N | King | Vashon | WA | 98070 | 2022 | FORD | ESCAPE | Plug-in Hybrid Electric Vehicle (PHEV) | Clea Alternativ Fuel Vehic Eligib |
| **112632** | KNDCD3LD4J | King | Covington | WA | 98042 | 2018 | KIA | NIRO | Plug-in Hybrid Electric Vehicle (PHEV) | Not eligib due to lo batter rang |
| **112633** | YV4BR0CL8N | King | Covington | WA | 98042 | 2022 | VOLVO | XC90 | Plug-in Hybrid Electric Vehicle (PHEV) | Not eligib due to lo batter rang |

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

Out[6]:

| | Postal Code | Model Year | Electric Range | Base MSRP | Legislative District | DOL Vehicle ID | 2( |
|---|---|---|---|---|---|---|---|
| **count** | 112634.000000 | 112634.000000 | 112634.000000 | 112634.000000 | 112348.000000 | 1.126340e+05 | 1.1 |
| **mean** | 98156.226850 | 2019.003365 | 87.812987 | 1793.439681 | 29.805604 | 1.994567e+08 | 5.2 |
| **std** | 2648.733064 | 2.892364 | 102.334216 | 10783.753486 | 14.700545 | 9.398427e+07 | 1.6 |
| **min** | 1730.000000 | 1997.000000 | 0.000000 | 0.000000 | 1.000000 | 4.777000e+03 | 1.1 |
| **25%** | 98052.000000 | 2017.000000 | 0.000000 | 0.000000 | 18.000000 | 1.484142e+08 | 5.3 |
| **50%** | 98119.000000 | 2020.000000 | 32.000000 | 0.000000 | 34.000000 | 1.923896e+08 | 5.3 |
| **75%** | 98370.000000 | 2022.000000 | 208.000000 | 0.000000 | 43.000000 | 2.191899e+08 | 5.3 |
| **max** | 99701.000000 | 2023.000000 | 337.000000 | 845000.000000 | 49.000000 | 4.792548e+08 | 5.6 |

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112634 entries, 0 to 112633
Data columns (total 17 columns):
 #   Column                                             Non-Null Count   Dtype  
---  ------                                             --------------   -----  
 0   VIN (1-10)                                         112634 non-null  object 
 1   County                                             112634 non-null  object 
 2   City                                               112634 non-null  object 
 3   State                                              112634 non-null  object 
 4   Postal Code                                        112634 non-null  int64  
 5   Model Year                                         112634 non-null  int64  
 6   Make                                               112634 non-null  object 
 7   Model                                              112614 non-null  object 
 8   Electric Vehicle Type                              112634 non-null  object 
 9   Clean Alternative Fuel Vehicle (CAFV) Eligibility  112634 non-null  object 
 10  Electric Range                                     112634 non-null  int64  
 11  Base MSRP                                          112634 non-null  int64  
 12  Legislative District                               112348 non-null  float64
 13  DOL Vehicle ID                                     112634 non-null  int64  
 14  Vehicle Location                                   112610 non-null  object 
 15  Electric Utility                                   112191 non-null  object 
 16  2020 Census Tract                                  112634 non-null  int64  
dtypes: float64(1), int64(6), object(10)
memory usage: 14.6+ MB
```

In [8]:
```
df.shape
```

Out[8]:
```
(112634, 17)
```

In [9]:
```
df.columns
```

Out[9]:
```
Index(['VIN (1-10)', 'County', 'City', 'State', 'Postal Code', 'Model Year',
       'Make', 'Model', 'Electric Vehicle Type',
       'Clean Alternative Fuel Vehicle (CAFV) Eligibility', 'Electric Range',
       'Base MSRP', 'Legislative District', 'DOL Vehicle ID',
       'Vehicle Location', 'Electric Utility', '2020 Census Tract'],
      dtype='object')
```

In [10]:
```
df.columns = df.columns.str.replace(' ', '_')
df.columns
```

Out[10]:
```
Index(['VIN_(1-10)', 'County', 'City', 'State', 'Postal_Code', 'Model_Year',
       'Make', 'Model', 'Electric_Vehicle_Type',
       'Clean_Alternative_Fuel_Vehicle_(CAFV)_Eligibility', 'Electric_Range',
       'Base_MSRP', 'Legislative_District', 'DOL_Vehicle_ID',
       'Vehicle_Location', 'Electric_Utility', '2020_Census_Tract'],
      dtype='object')
```

In [11]:
```
df.rename(columns={'Clean_Alternative_Fuel_Vehicle_(CAFV)_Eligibility':'CAFV_Eligibili
df.columns
```

Out[11]:
```
Index(['VIN_(1-10)', 'County', 'City', 'State', 'Postal_Code', 'Model_Year',
       'Make', 'Model', 'Electric_Vehicle_Type', 'CAFV_Eligibility',
       'Electric_Range', 'Base_MSRP', 'Legislative_District', 'DOL_Vehicle_ID',
       'Vehicle_Location', 'Electric_Utility', '2020_Census_Tract'],
      dtype='object')
```

In [12]:
```
df
```

Out[12]:

| | VIN_(1-10) | County | City | State | Postal_Code | Model_Year | Make | Model | I |
|---|---|---|---|---|---|---|---|---|---|
| **0** | JTMEB3FV6N | Monroe | Key West | FL | 33040 | 2022 | TOYOTA | RAV4 PRIME | |
| **1** | 1G1RD6E45D | Clark | Laughlin | NV | 89029 | 2013 | CHEVROLET | VOLT | |
| **2** | JN1AZ0CP8B | Yakima | Yakima | WA | 98901 | 2011 | NISSAN | LEAF | |
| **3** | 1G1FW6S08H | Skagit | Concrete | WA | 98237 | 2017 | CHEVROLET | BOLT EV | |
| **4** | 3FA6P0SU1K | Snohomish | Everett | WA | 98201 | 2019 | FORD | FUSION | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **112629** | 7SAYGDEF2N | King | Duvall | WA | 98019 | 2022 | TESLA | MODEL Y | |
| **112630** | 1N4BZ1CP7K | San Juan | Friday Harbor | WA | 98250 | 2019 | NISSAN | LEAF | |
| **112631** | 1FMCU0KZ4N | King | Vashon | WA | 98070 | 2022 | FORD | ESCAPE | |
| **112632** | KNDCD3LD4J | King | Covington | WA | 98042 | 2018 | KIA | NIRO | |
| **112633** | YV4BR0CL8N | King | Covington | WA | 98042 | 2022 | VOLVO | XC90 | |

112634 rows × 17 columns

In [13]:

```
print(df.isnull().sum())
```

```
VIN_(1-10)               0
County                   0
City                     0
State                    0
Postal_Code              0
Model_Year               0
Make                     0
Model                   20
Electric_Vehicle_Type    0
CAFV_Eligibility         0
Electric_Range           0
Base_MSRP                0
Legislative_District   286
DOL_Vehicle_ID           0
Vehicle_Location        24
Electric_Utility       443
2020_Census_Tract        0
dtype: int64
```

In [14]:
```
df_dropna = df.dropna(inplace=True)
```

In [15]:
```
print(df.isnull().sum())
```

```
VIN_(1-10)             0
County                 0
City                   0
State                  0
Postal_Code            0
Model_Year             0
Make                   0
Model                  0
Electric_Vehicle_Type  0
CAFV_Eligibility       0
Electric_Range         0
Base_MSRP              0
Legislative_District   0
DOL_Vehicle_ID         0
Vehicle_Location       0
Electric_Utility       0
2020_Census_Tract      0
dtype: int64
```

# task 1

Non-Visual Univariate Analysis

In [16]:
```
numerical_columns = ['Postal_Code', 'Model_Year', 'Electric_Range', 'Base_MSRP', 'Legi


categorical_columns = ['VIN_(1-10)', 'County', 'City', 'State', 'Make', 'Model', 'Elec

discrete_df = df.select_dtypes(include=['object'])
numerical_df = df.select_dtypes(include=['int64', 'float64'])
```

In [ ]:

In [17]:
```
def discrete_univariate_analysis(discrete_data):
    for col_name in discrete_data:
        print("-"*10, col_name, "-"*10)
        print(discrete_data[col_name].agg(['count', 'nunique', 'unique']))
        print('Value Counts: \n', discrete_data[col_name].value_counts())
        print()
```

In [ ]:

In [18]:
```
discrete_univariate_analysis(discrete_df)
```

```
---------- VIN_(1-10) ----------
count                                                  112152
nunique                                                  7522
unique     [JN1AZ0CP8B, 1G1FW6S08H, 3FA6P0SU1K, 5YJ3E1EB5...
Name: VIN_(1-10), dtype: object
Value Counts:
 5YJYGDEE9M    471
5YJYGDEE0M    463
5YJYGDEE7M    447
5YJYGDEE8M    446
5YJYGDEE2M    435
             ...
YV4BR0DL8M      1
JTJHKCFZ5N      1
WA1J2BFZ3N      1
KNDC4DLC5P      1
WA1LAAGE5M      1
Name: VIN_(1-10), Length: 7522, dtype: int64

---------- County ----------
count                                                  112152
nunique                                                    39
unique     [Yakima, Skagit, Snohomish, Island, Thurston, ...
Name: County, dtype: object
Value Counts:
 King            58980
Snohomish       12412
Pierce           8525
Clark            6681
Thurston         4109
Kitsap           3828
Whatcom          2839
Spokane          2785
Benton           1376
Island           1298
Skagit           1228
Clallam           728
San Juan          717
Jefferson         698
Chelan            654
Yakima            617
Cowlitz           569
Mason             547
Lewis             431
Grays Harbor      402
Kittitas          392
Franklin          365
Grant             335
Walla Walla       312
Douglas           221
Whitman           177
Klickitat         175
Okanogan          149
Pacific           145
Skamania          139
Stevens            91
Asotin             48
Wahkiakum          39
Adams              34
Pend Oreille       32
Lincoln            30
```

```
Ferry              27
Columbia           13
Garfield            4
Name: County, dtype: int64

---------- City ----------
count                                                 112152
nunique                                                  435
unique     [Yakima, Concrete, Everett, Bothell, Mukilteo,...
Name: City, dtype: object
Value Counts:
 Seattle           20295
Bellevue           5919
Redmond            4199
Vancouver          4013
Kirkland           3598
                  ...
Walla Walla Co        1
Clallam Bay           1
Malott                1
Rockport              1
Uniontown             1
Name: City, Length: 435, dtype: int64

---------- State ----------
count      112152
nunique         1
unique       [WA]
Name: State, dtype: object
Value Counts:
 WA    112152
Name: State, dtype: int64

---------- Make ----------
count                                                 112152
nunique                                                   34
unique     [NISSAN, CHEVROLET, FORD, TESLA, KIA, AUDI, BM...
Name: Make, dtype: object
Value Counts:
 TESLA             51883
NISSAN            12846
CHEVROLET         10140
FORD               5780
BMW                4660
KIA                4469
TOYOTA             4368
VOLKSWAGEN         2507
AUDI               2320
VOLVO              2256
CHRYSLER           1780
HYUNDAI            1407
JEEP               1143
RIVIAN              883
FIAT                820
PORSCHE             817
HONDA               788
MINI                631
MITSUBISHI          585
POLESTAR            557
MERCEDES-BENZ       503
SMART               271
```

```
JAGUAR              218
LINCOLN             167
CADILLAC            108
LUCID MOTORS         65
SUBARU               59
LAND ROVER           38
LEXUS                33
FISKER               19
GENESIS              18
AZURE DYNAMICS        7
TH!NK                 3
BENTLEY               3
Name: Make, dtype: int64

---------- Model ----------
count                                                 112152
nunique                                                  114
unique     [LEAF, BOLT EV, FUSION, MODEL 3, SOUL, Q5 E, M...
Name: Model, dtype: object
Value Counts:
 MODEL 3        23042
MODEL Y        17086
LEAF           12846
MODEL S         7346
BOLT EV         4895
               ...
745LE              2
S-10 PICKUP        1
SOLTERRA           1
918                1
FLYING SPUR        1
Name: Model, Length: 114, dtype: int64

---------- Electric_Vehicle_Type ----------
count                                                 112152
nunique                                                    2
unique     [Battery Electric Vehicle (BEV), Plug-in Hybri...
Name: Electric_Vehicle_Type, dtype: object
Value Counts:
 Battery Electric Vehicle (BEV)            85732
Plug-in Hybrid Electric Vehicle (PHEV)    26420
Name: Electric_Vehicle_Type, dtype: int64

---------- CAFV_Eligibility ----------
count                                                 112152
nunique                                                    3
unique     [Clean Alternative Fuel Vehicle Eligible, Not ...
Name: CAFV_Eligibility, dtype: object
Value Counts:
 Clean Alternative Fuel Vehicle Eligible                          58395
Eligibility unknown as battery range has not been researched    39097
Not eligible due to low battery range                            14660
Name: CAFV_Eligibility, dtype: int64

---------- Vehicle_Location ----------
count                                                 112152
nunique                                                  516
unique     [POINT (-120.50721 46.60448), POINT (-121.7515...
Name: Vehicle_Location, dtype: object
Value Counts:
 POINT (-122.13158 47.67858)    2914
```

```
POINT (-122.2066 47.67887)     2059
POINT (-122.1872 47.61001)     2001
POINT (-122.31765 47.70013)    1878
POINT (-122.12096 47.55584)    1851
                               ...
POINT (-121.59274 48.48758)       1
POINT (27.25316 67.01865)         1
POINT (-124.16705 47.11487)       1
POINT (-123.00026 48.61989)       1
POINT (-117.08742 46.53906)       1
Name: Vehicle_Location, Length: 516, dtype: int64

---------- Electric_Utility ----------
count                                               112152
nunique                                                 73
unique     [PACIFICORP, PUGET SOUND ENERGY INC, PUD NO 2 ...
Name: Electric_Utility, dtype: object
Value Counts:
 PUGET SOUND ENERGY INC||CITY OF TACOMA - (WA)
40231
PUGET SOUND ENERGY INC
22166
CITY OF SEATTLE - (WA)|CITY OF TACOMA - (WA)
21439
BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF CLARK COUNTY - (WA)
6522
BONNEVILLE POWER ADMINISTRATION||CITY OF TACOMA - (WA)||PENINSULA LIGHT COMPANY
5049

...
BONNEVILLE POWER ADMINISTRATION||PENINSULA LIGHT COMPANY
1
BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF ASOTIN COUNTY
1
CITY OF SEATTLE - (WA)
1
BONNEVILLE POWER ADMINISTRATION||NESPELEM VALLEY ELEC COOP, INC
1
BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF CLALLAM COUNTY|PUD NO 1 OF JEFFERSON COU
NTY        1
Name: Electric_Utility, Length: 73, dtype: int64
```

In [ ]:

```

```

In [19]:

```python
def numerical_univariate_analysis(numerical_data):
    for col_name in numerical_data:
        print("-"*10, col_name, "-"*10)
        print(numerical_data[col_name].agg(['min', 'max', 'mean', 'median', 'std']))
        print()
```

In [20]:

```python
numerical_univariate_analysis(numerical_df)
```

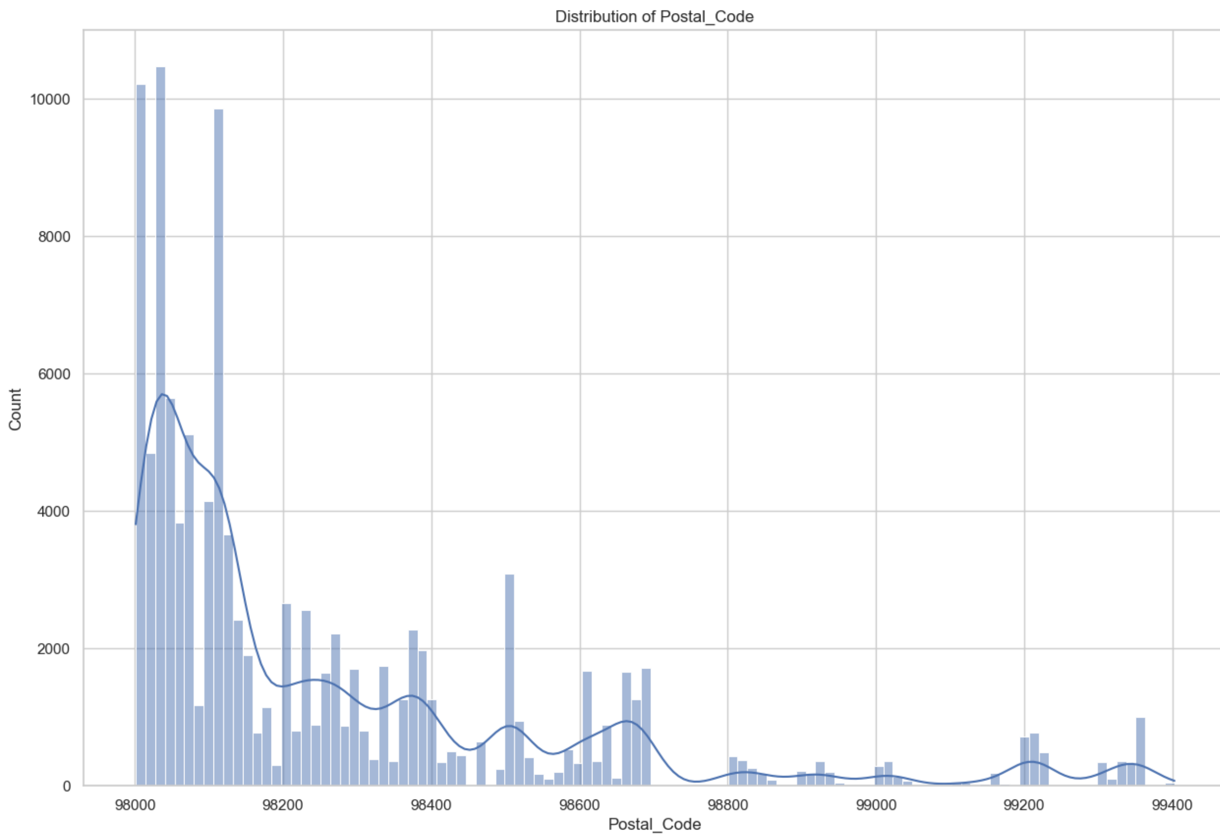
```
---------- Postal_Code ----------
min       98001.000000
max       99403.000000
mean      98258.856659
median    98121.000000
std         302.889935
Name: Postal_Code, dtype: float64

---------- Model_Year ----------
min       1997.000000
max       2023.000000
mean      2019.004494
median    2020.000000
std          2.891859
Name: Model_Year, dtype: float64

---------- Electric_Range ----------
min         0.000000
max       337.000000
mean       87.829651
median     32.000000
std       102.336645
Name: Electric_Range, dtype: float64

---------- Base_MSRP ----------
min            0.000000
max       845000.000000
mean        1793.882320
median         0.000000
std        10785.259118
Name: Base_MSRP, dtype: float64

---------- Legislative_District ----------
min        1.000000
max       49.000000
mean      29.817703
median    34.000000
std       14.698726
Name: Legislative_District, dtype: float64

---------- DOL_Vehicle_ID ----------
min       4.777000e+03
max       4.792548e+08
mean      1.994712e+08
median    1.923916e+08
std       9.401842e+07
Name: DOL_Vehicle_ID, dtype: float64

---------- 2020_Census_Tract ----------
min       5.300195e+10
max       5.307794e+10
mean      5.303958e+10
median    5.303303e+10
std       1.617788e+07
Name: 2020_Census_Tract, dtype: float64
```

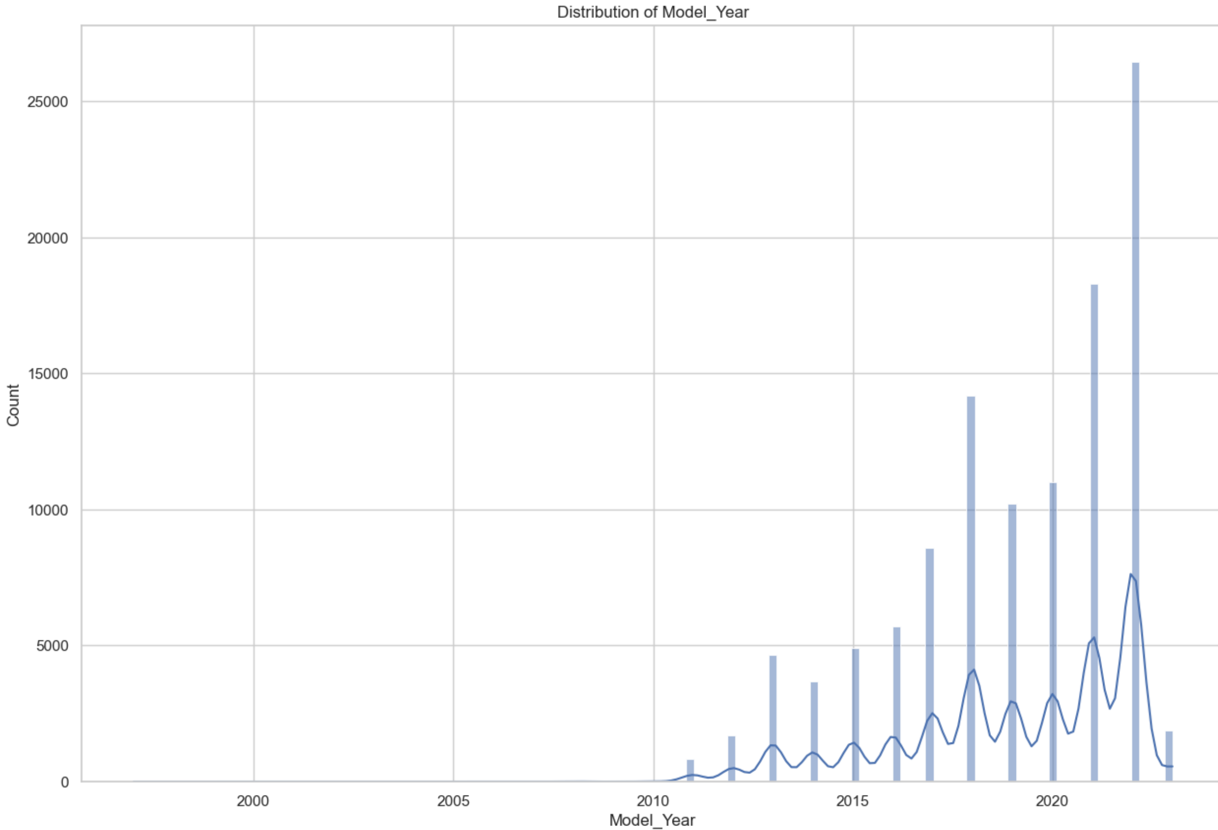
In [ ]:

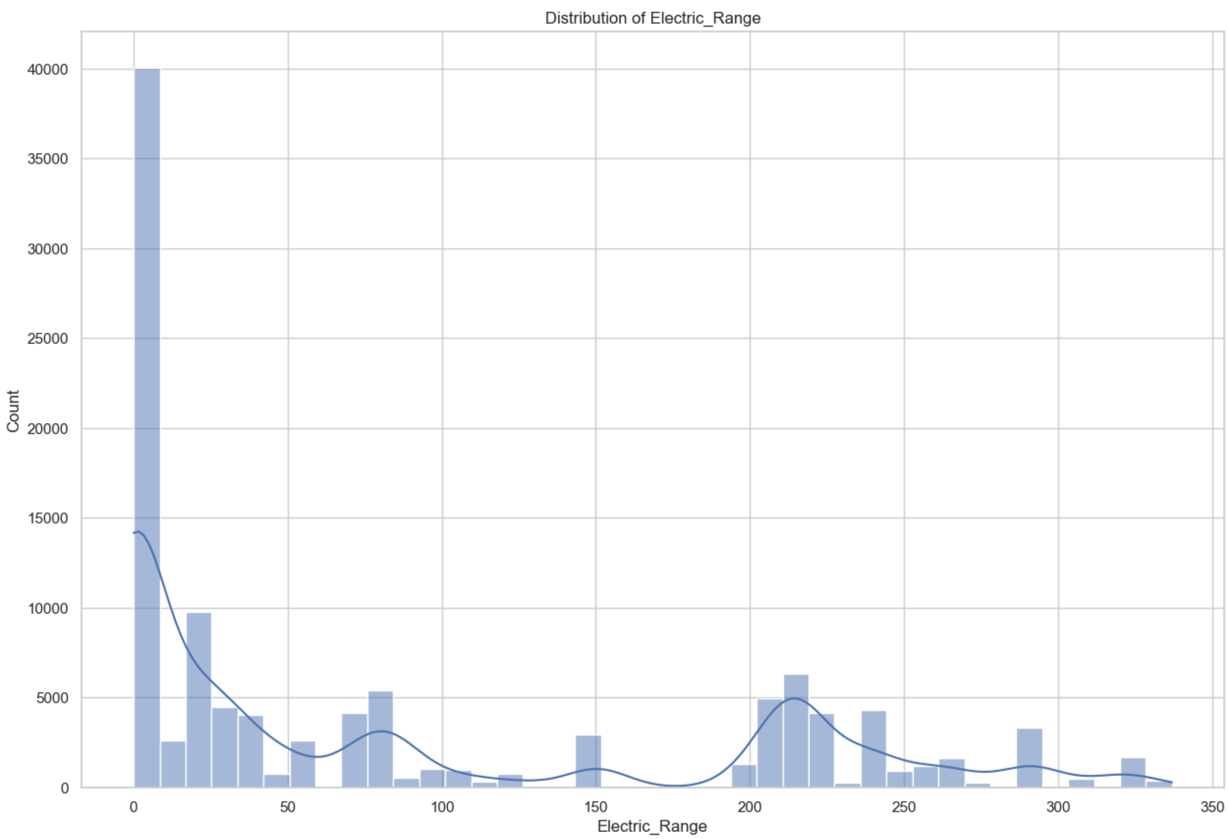
# Visual Univariate Analysis on Numerical Columns

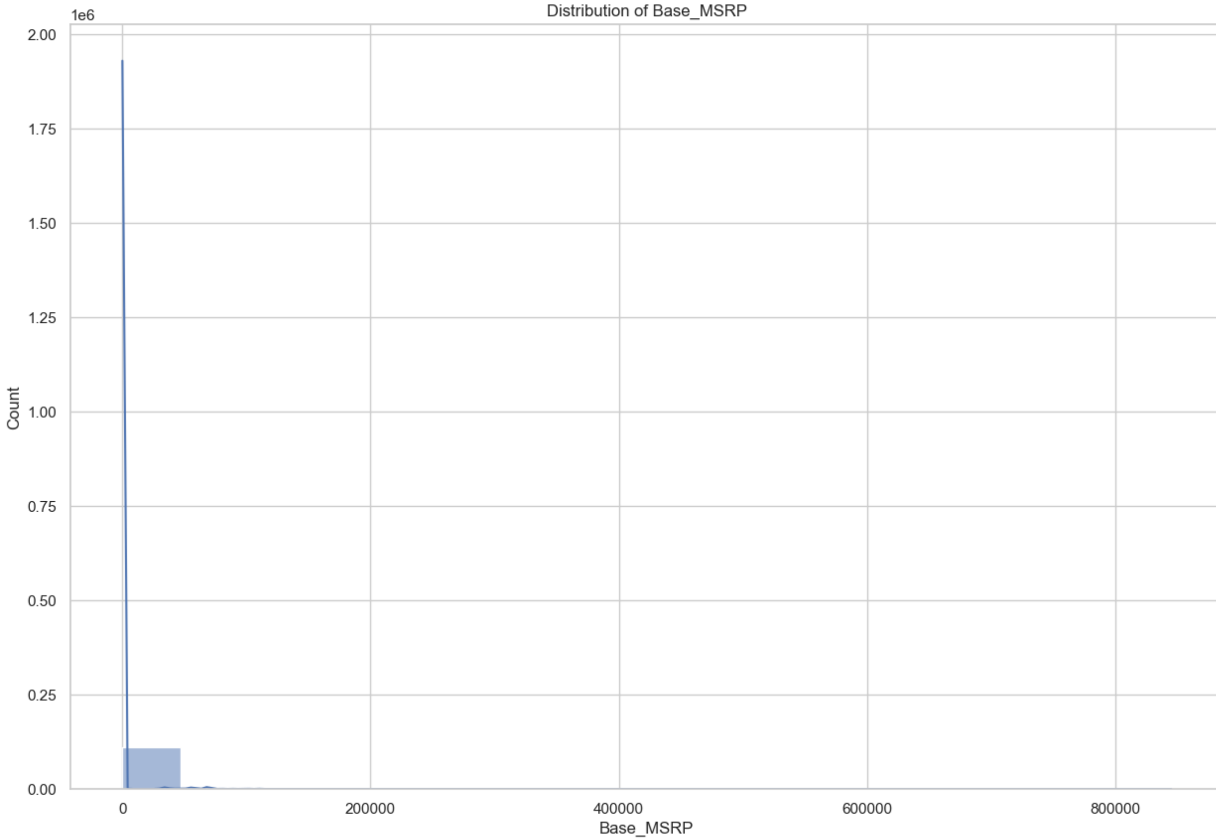
Frequency Distribution

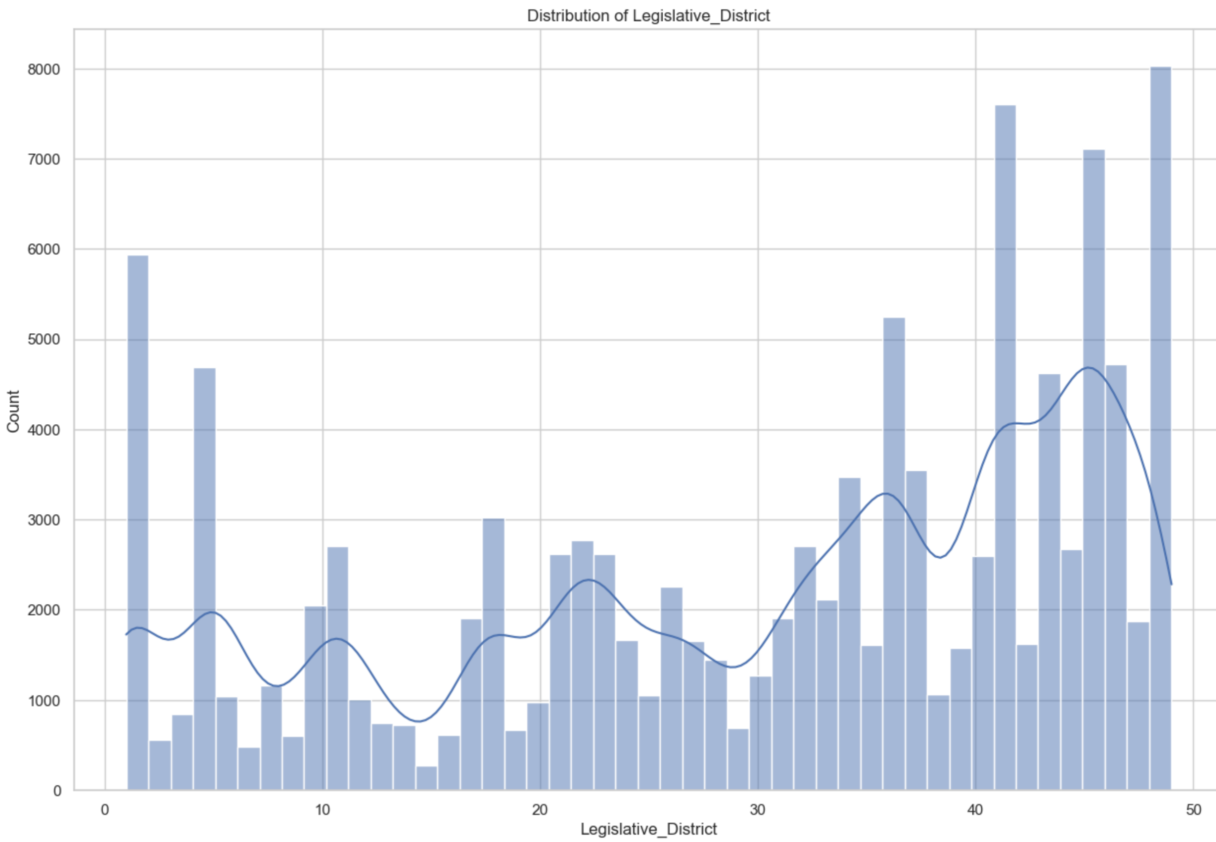
In [21]:
```
sns.set(style="whitegrid")# Univariate Analysis: Distribution of Numerical Columns

# Plot histograms for numerical columns

for column in numerical_columns:
    plt.figure(figsize=(15, 10))

    sns.histplot(df[column], kde=True)
    plt.title(f'Distribution of {column}')
plt.tight_layout()
plt.show()
```

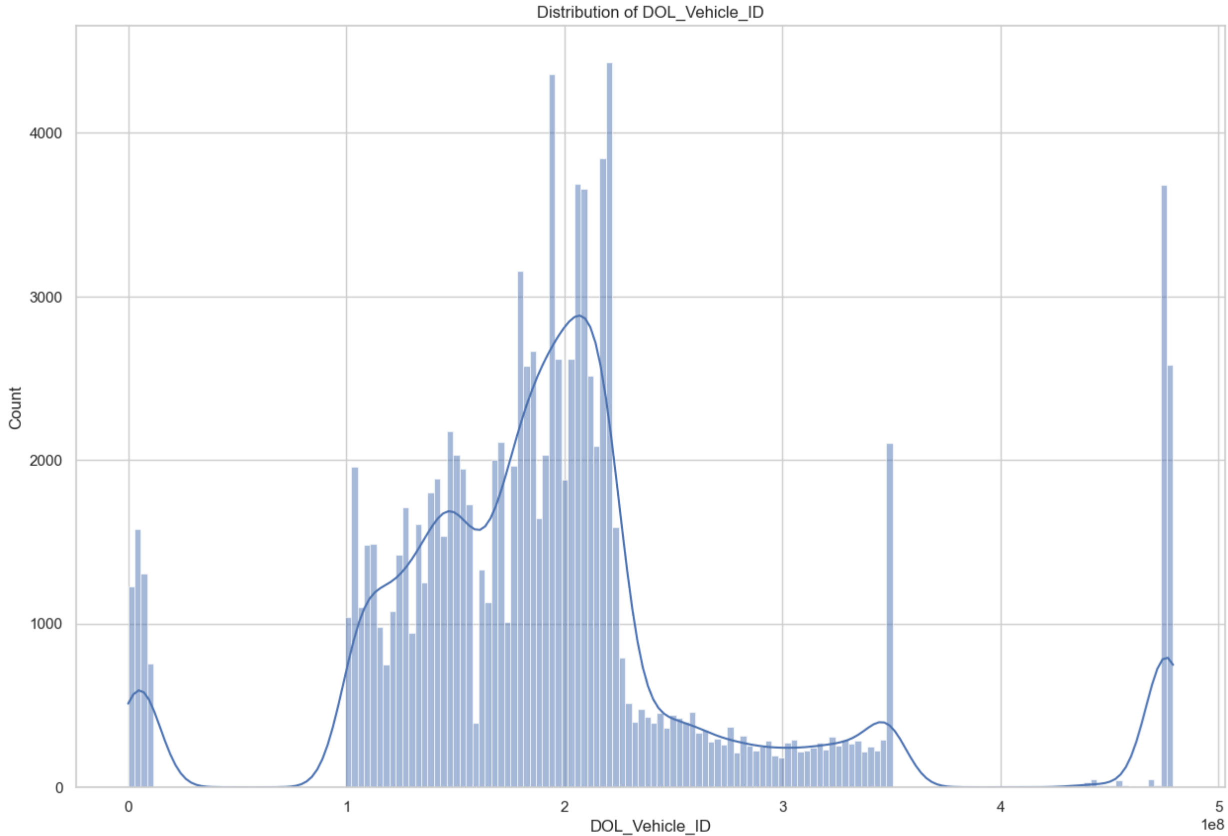


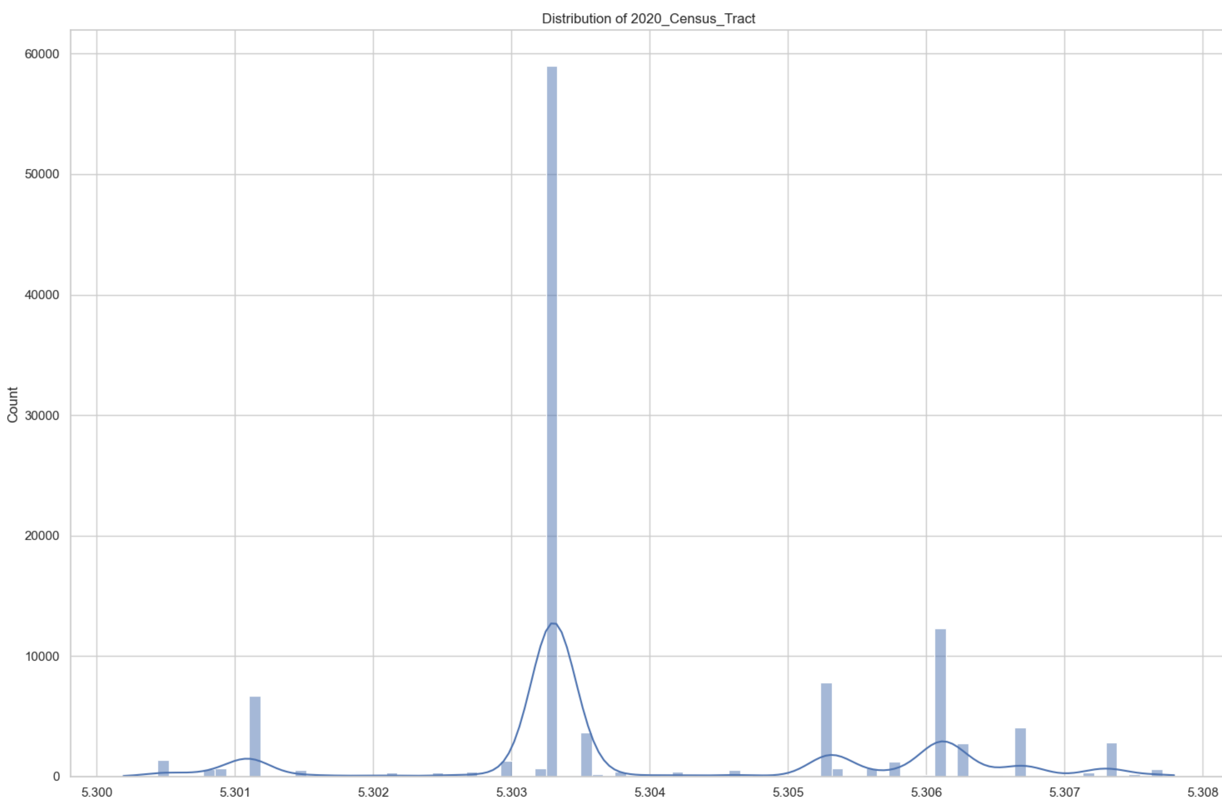

Distribution of Model_Year
Count
Model_Year
25000
20000
15000
10000
5000
0
2000
2005
2010
2015
2020


Distribution of Electric_Range
Count
Electric_Range
40000
35000
30000
25000
20000
15000
10000
5000
0
0
50
100
150
200
250
300
350


Distribution of Base_MSRP
1e6
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Count
0
200000
400000
600000
800000
Base_MSRP


Distribution of Legislative_District
8000
7000
6000
5000
4000
3000
2000
1000
0
Count
0
10
20
30
40
50
Legislative_District

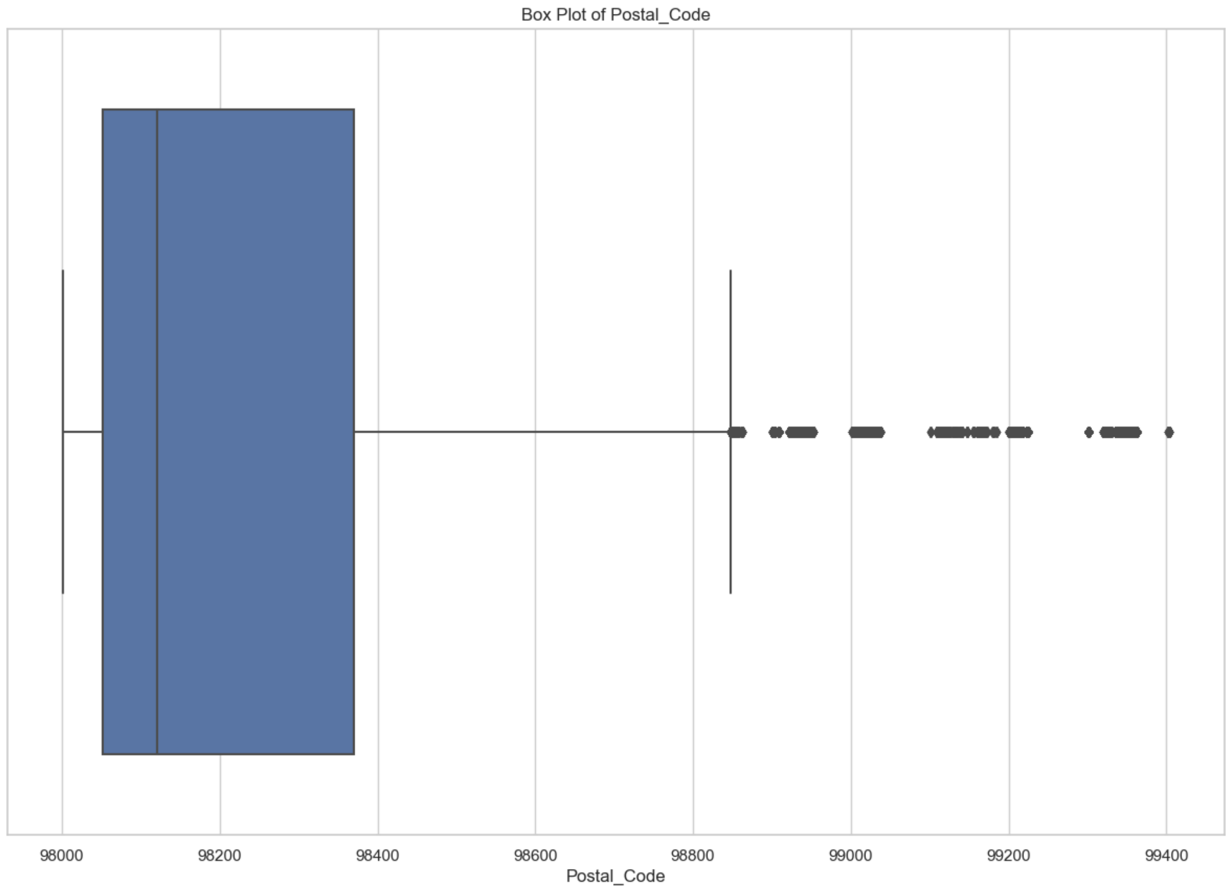


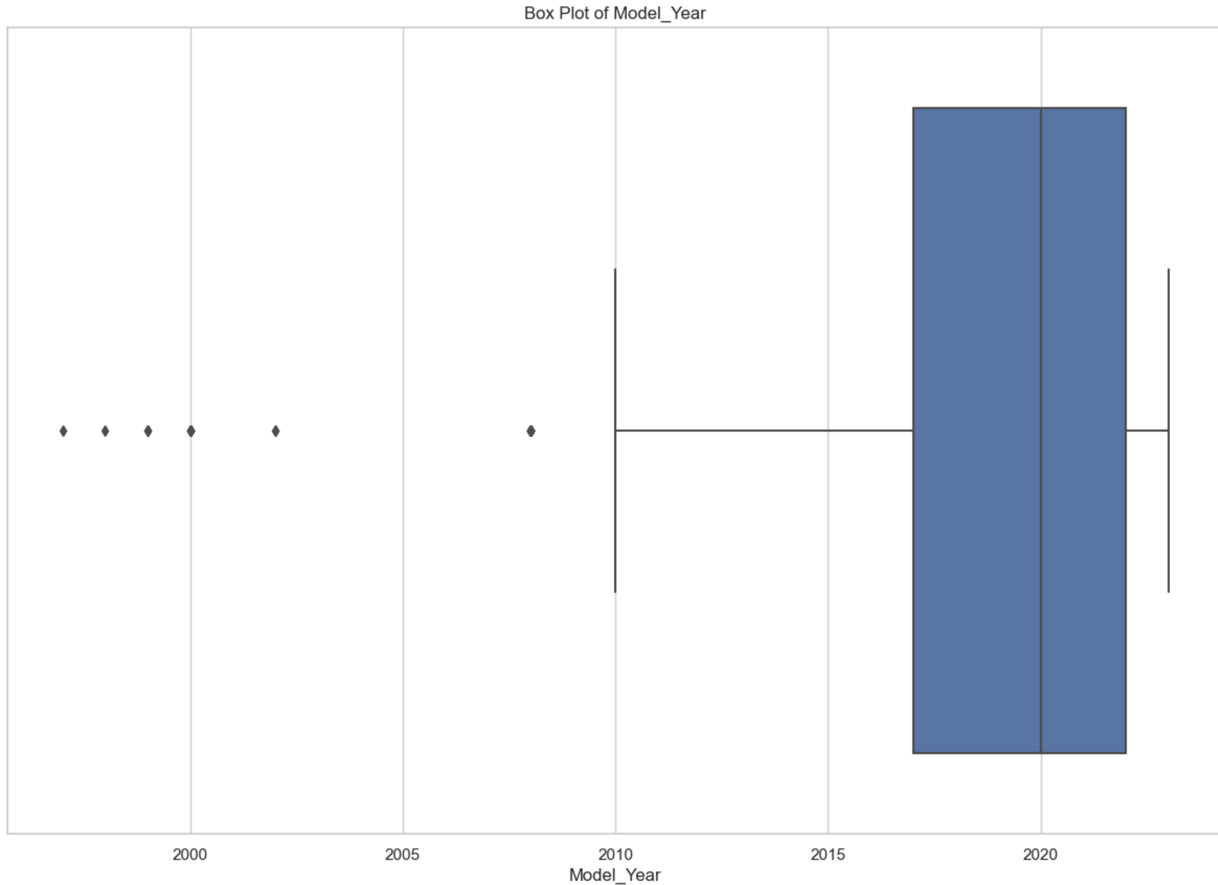



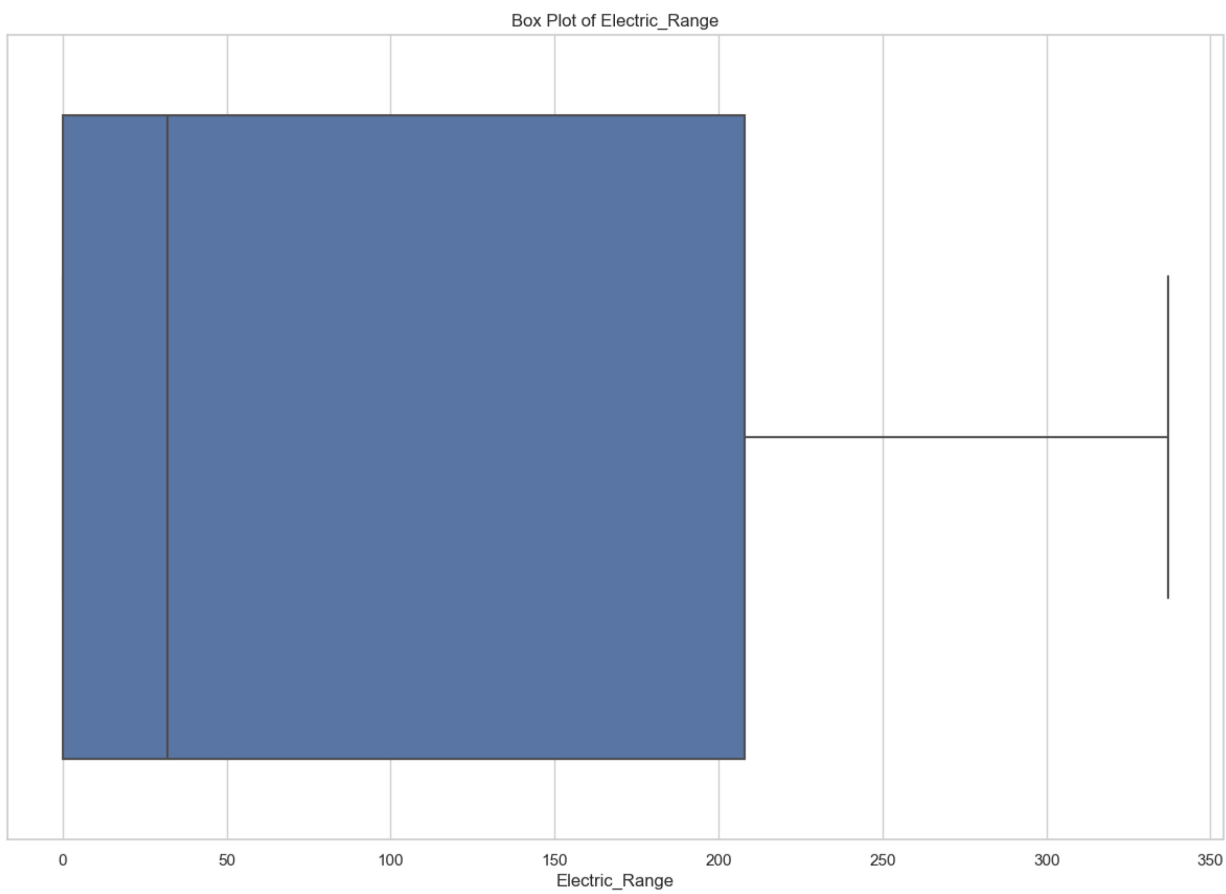

In [ ]:

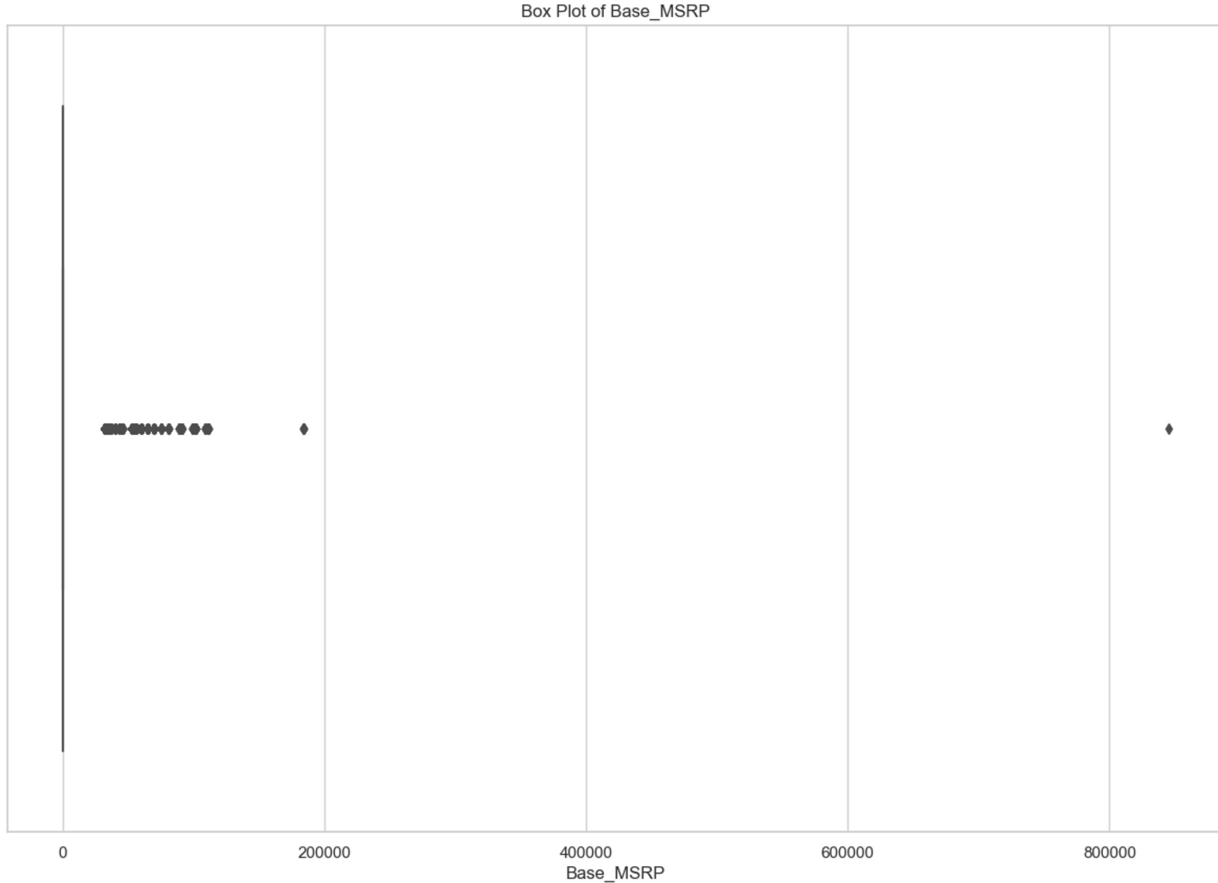
# Outlier Detection

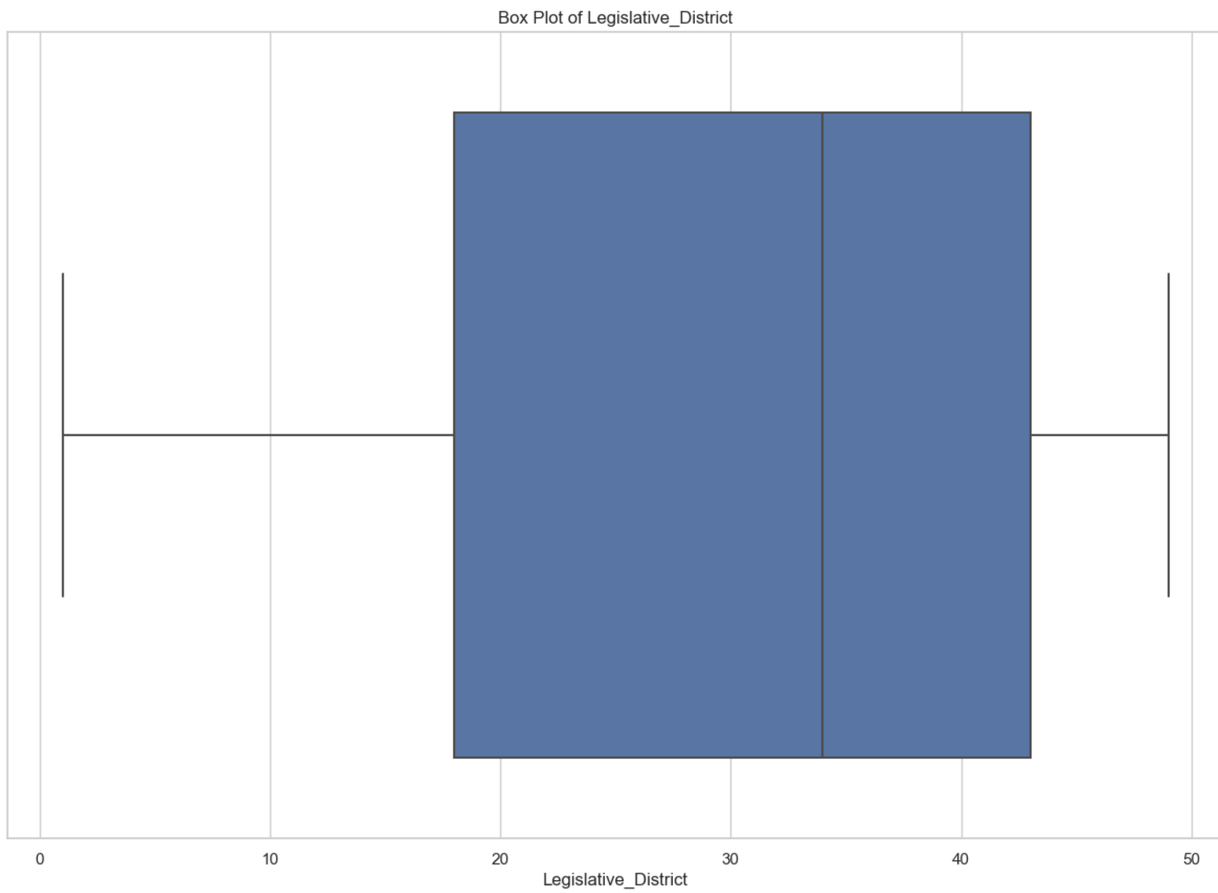
In [22]:
```
for column in numerical_columns:
    plt.figure(figsize=(15, 10))

    sns.boxplot(x=df[column])
    plt.title(f'Box Plot of {column}')
plt.tight_layout()
plt.show()
```

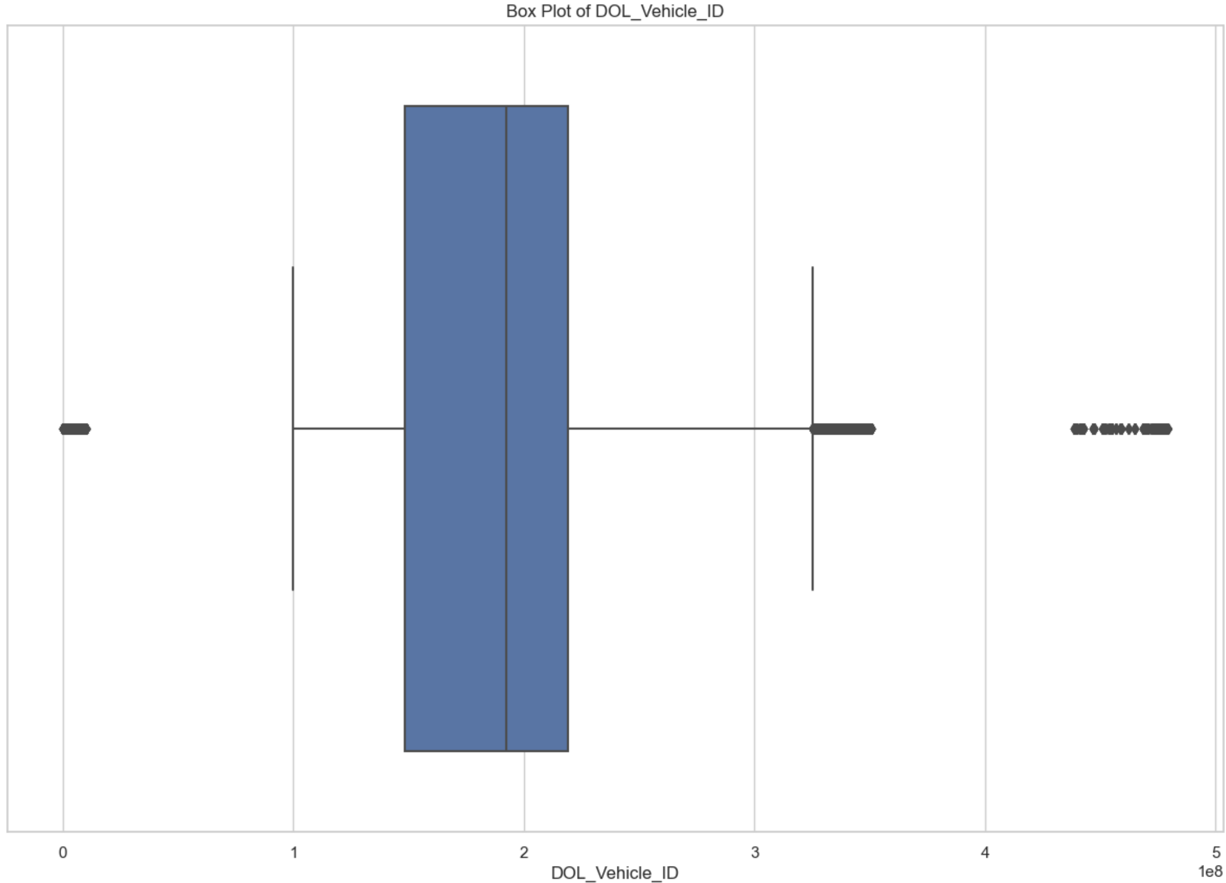


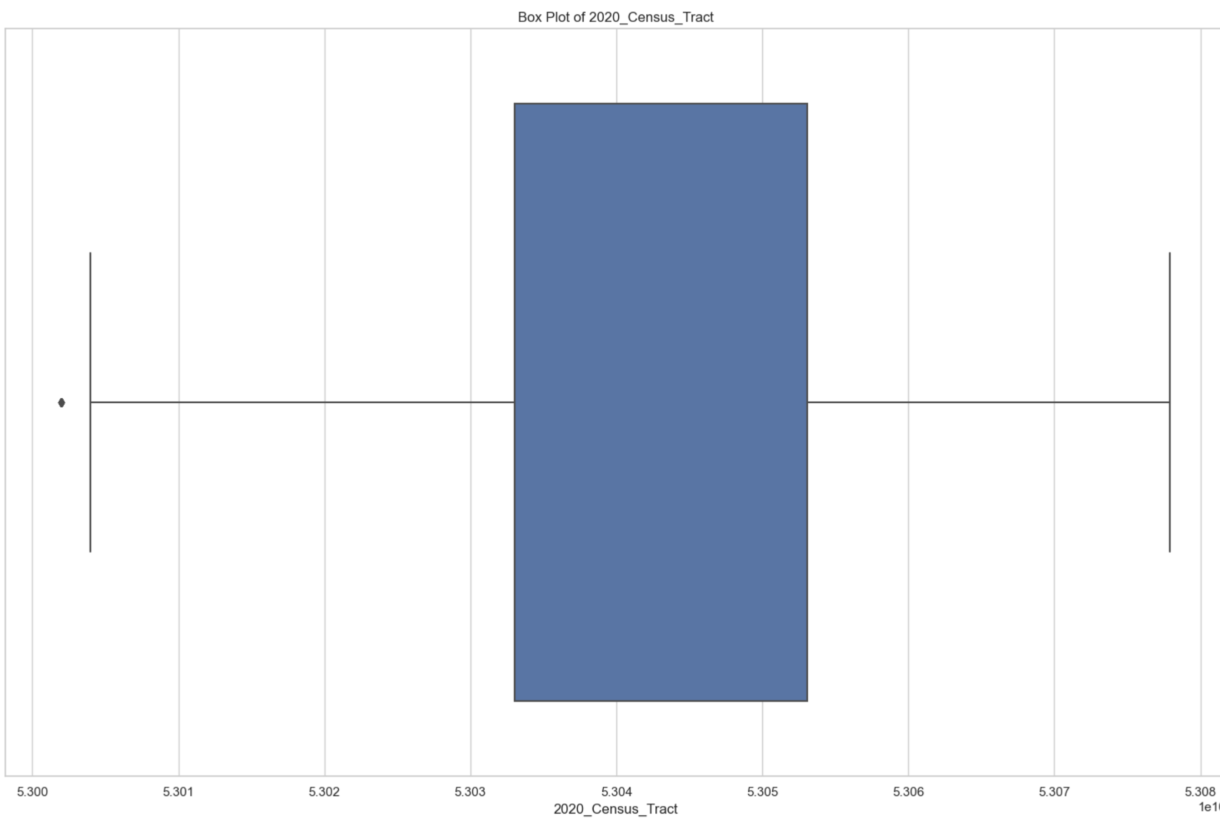

Box Plot of Model_Year
2000
2005
2010
2015
2020
Model_Year


Box Plot of Electric_Range
0
50
100
150
200
250
300
350
Electric_Range


Box Plot of Base_MSRP
0
200000
400000
600000
800000
Base_MSRP


Box Plot of Legislative_District
0
10
20
30
40
50
Legislative_District




In [ ]:

In [23]:
```
def describe_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
```

```
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)]
    print(f"\
Column: {column}")
    print(f"Number of outliers: {len(outliers)}")
    print(f"Percentage of outliers: {len(outliers) / len(df) * 100:.2f}%")
    print(f"Range of outliers: {outliers[column].min()} to {outliers[column].max()}")
    print(f"Range of non-outliers: {df[(df[column] >= lower_bound) & (df[column] <= up

for column in numerical_columns:
    describe_outliers(df, column)
```

```
Column: Postal_Code
Number of outliers: 6514
Percentage of outliers: 5.81%
Range of outliers: 98848 to 99403
Range of non-outliers: 98001 to 98847
Column: Model_Year
Number of outliers: 40
Percentage of outliers: 0.04%
Range of outliers: 1997 to 2008
Range of non-outliers: 2010 to 2023
Column: Electric_Range
Number of outliers: 0
Percentage of outliers: 0.00%
Range of outliers: nan to nan
Range of non-outliers: 0 to 337
Column: Base_MSRP
Number of outliers: 3498
Percentage of outliers: 3.12%
Range of outliers: 31950 to 845000
Range of non-outliers: 0 to 0
Column: Legislative_District
Number of outliers: 0
Percentage of outliers: 0.00%
Range of outliers: nan to nan
Range of non-outliers: 1.0 to 49.0
Column: DOL_Vehicle_ID
Number of outliers: 15483
Percentage of outliers: 13.81%
Range of outliers: 4777 to 479254772
Range of non-outliers: 100021575 to 325344519
Column: 2020_Census_Tract
Number of outliers: 34
Percentage of outliers: 0.03%
Range of outliers: 53001950100 to 53001950500
Range of non-outliers: 53003960100 to 53077940007
```

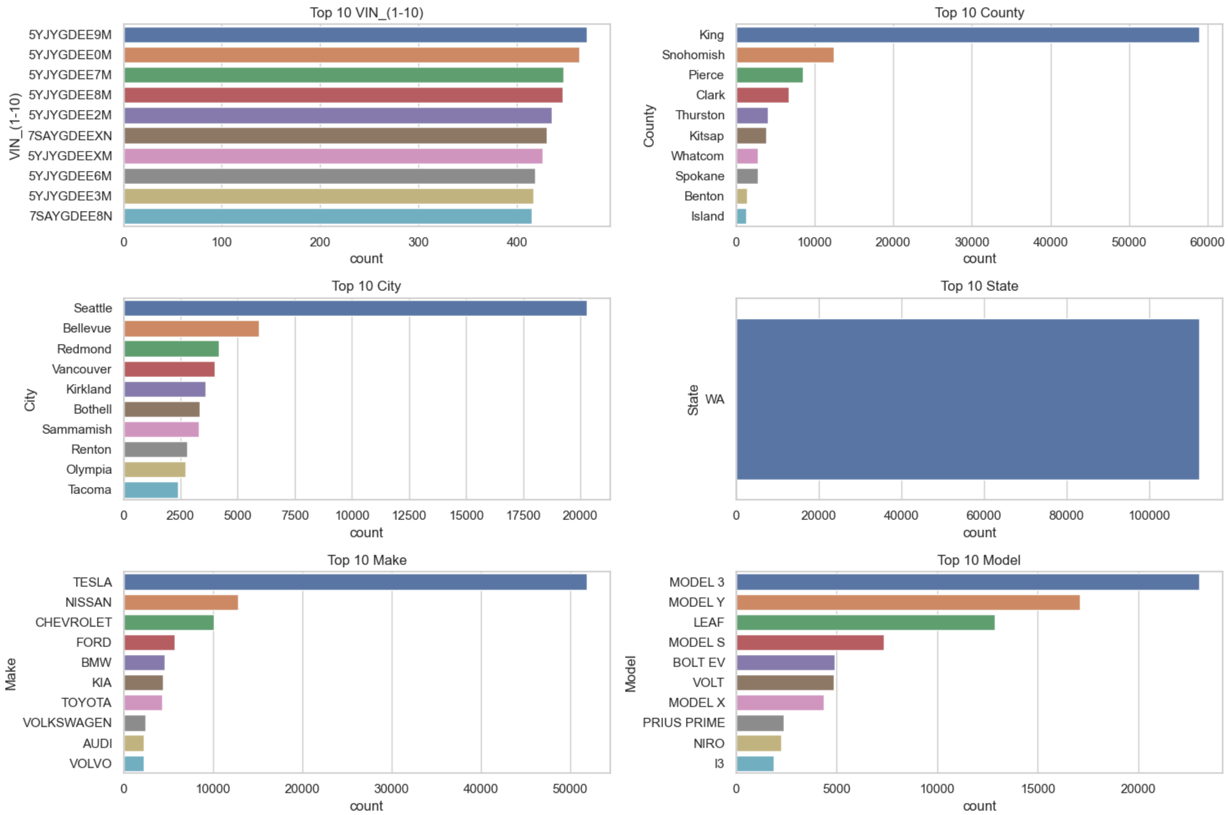
In [ ]:

# Visual Univariate Analysis on Categorical Variables

In [24]:

```
plt.figure(figsize=(15, 10))
for i, column in enumerate(categorical_columns[:6], 1):  # Limiting to first 6 for clo
    plt.subplot(3, 2, i)
```

```
    sns.countplot(y=df[column], order=df[column].value_counts().index[:10])
    plt.title(f'Top 10 {column}')
plt.tight_layout()
plt.show()
```



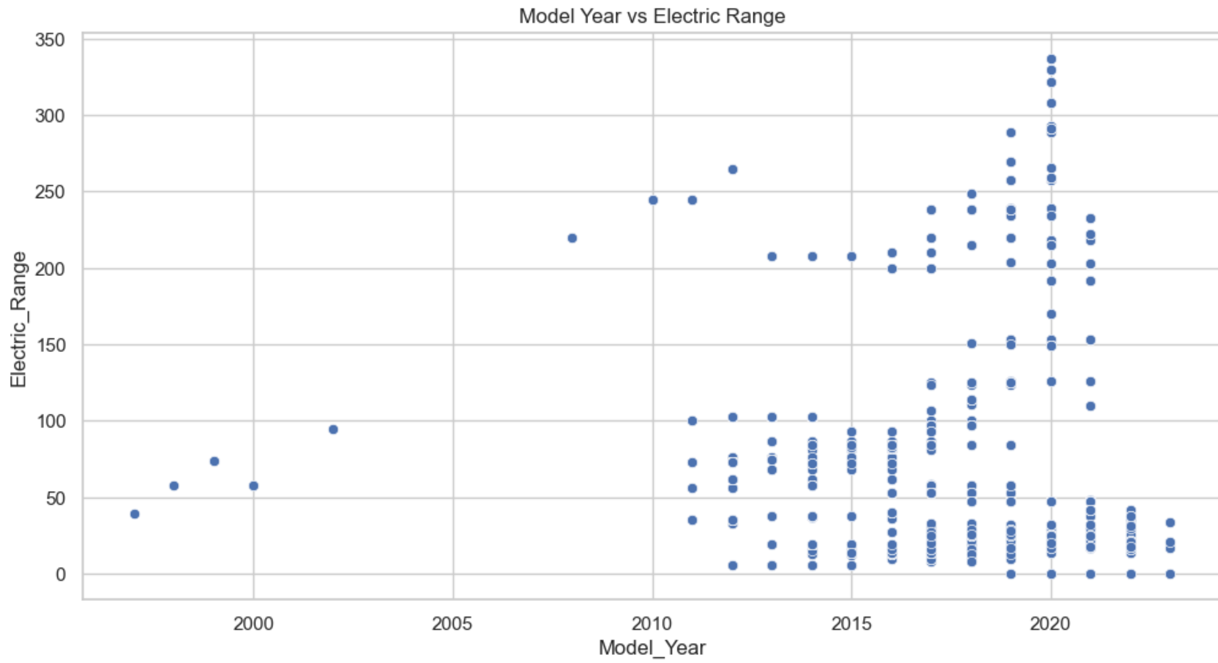

In [ ]:

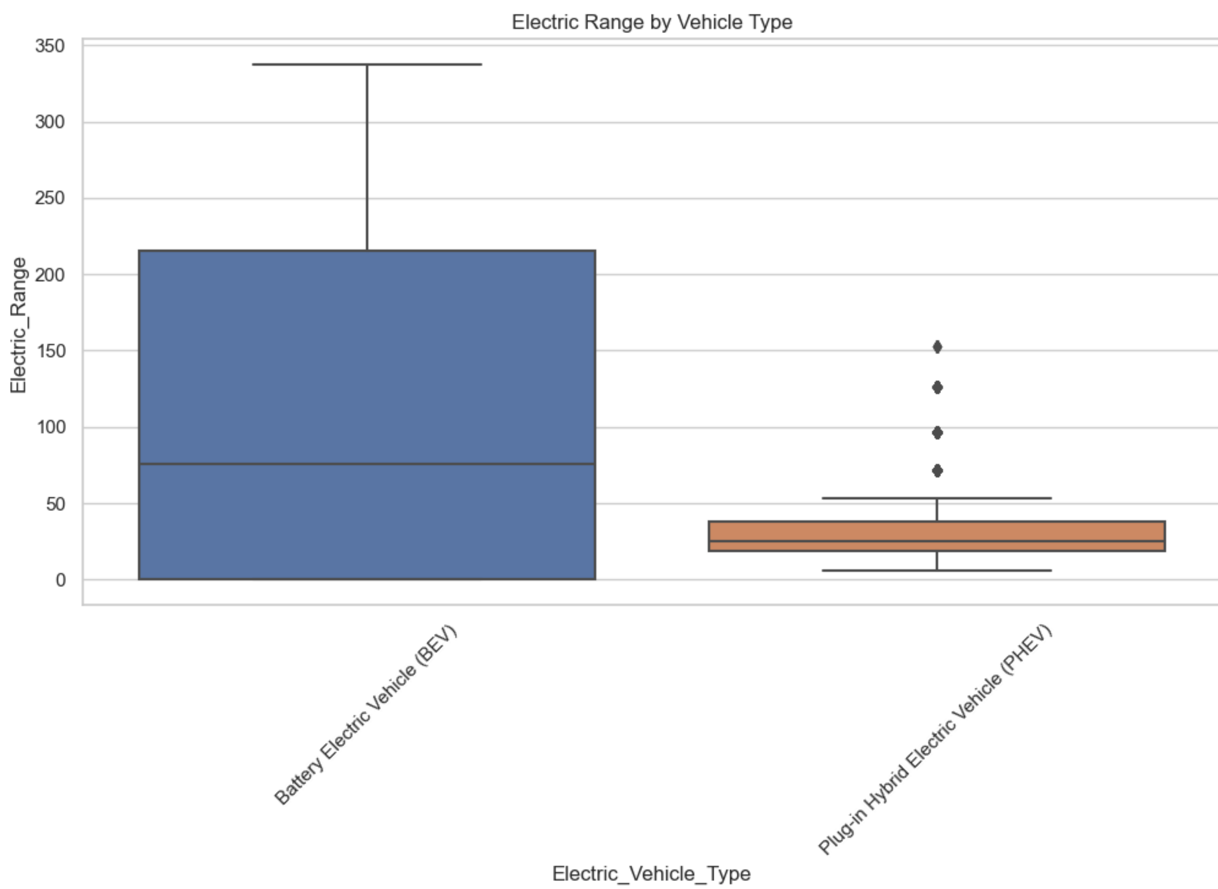
# Bivariate Analysis

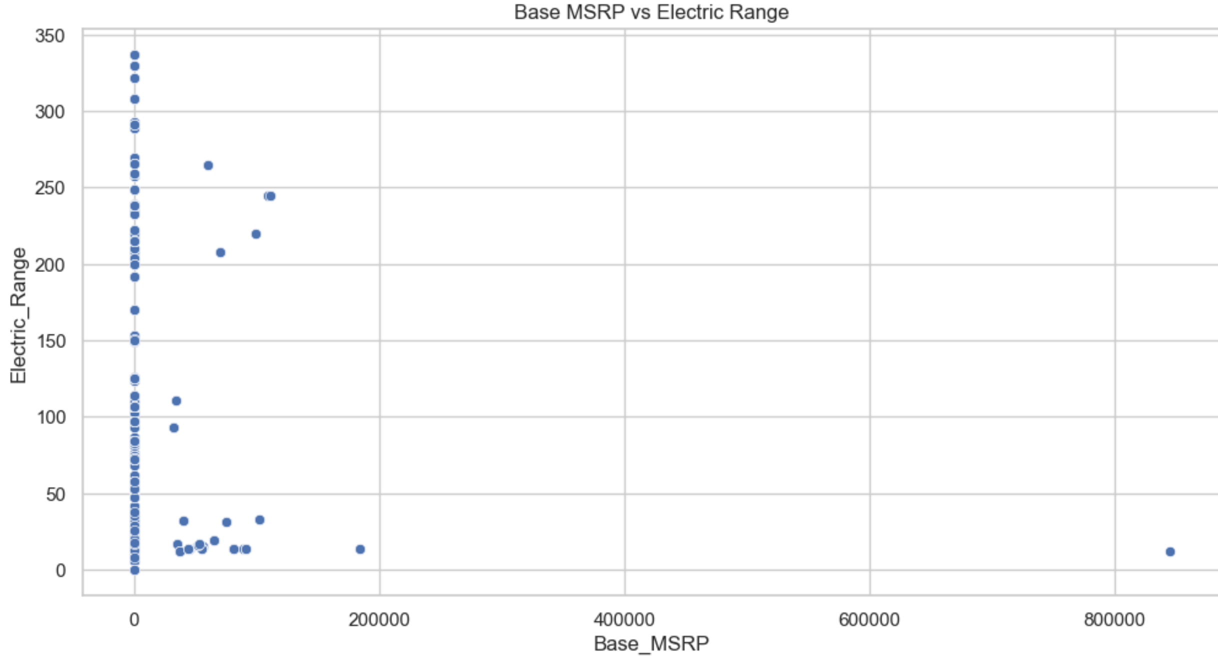
In [25]:

```
# 1. Relationship between Model Year and Electric Range
plt.figure(figsize=(12, 6))
sns.scatterplot(x='Model_Year', y='Electric_Range', data=df)
plt.title('Model Year vs Electric Range')
plt.show()

# 2. Comparison of Electric Range across different Electric Vehicle Types
plt.figure(figsize=(12, 6))
sns.boxplot(x='Electric_Vehicle_Type', y='Electric_Range', data=df)
plt.title('Electric Range by Vehicle Type')
plt.xticks(rotation=45)
plt.show()

# 3. Correlation between Electric Range and Base MSRP
# First, let's check if Base MSRP has non-zero values
if df['Base_MSRP'].sum() > 0:
    plt.figure(figsize=(12, 6))
    sns.scatterplot(x='Base_MSRP', y='Electric_Range', data=df)
    plt.title('Base MSRP vs Electric Range')
    plt.show()
else:
    print("Base MSRP column contains only zero values. Skipping this analysis.")
```

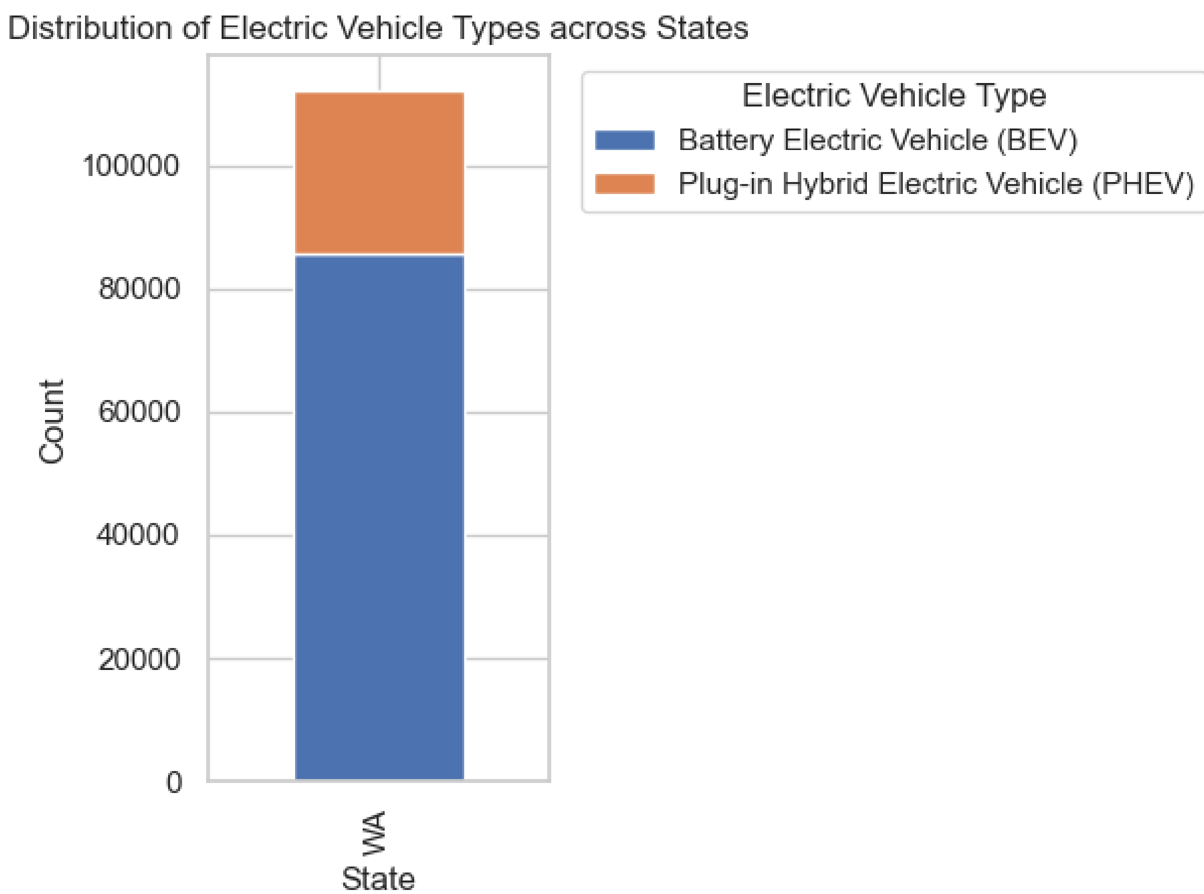
```
# 4. Distribution of Electric Vehicle Types across different States
vehicle_type_by_state = df.groupby('State')['Electric_Vehicle_Type'].value_counts().ur
plt.figure(figsize=(15, 8))
vehicle_type_by_state.plot(kind='bar', stacked=True)
plt.title('Distribution of Electric Vehicle Types across States')
plt.xlabel('State')
plt.ylabel('Count')
plt.legend(title='Electric Vehicle Type', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```





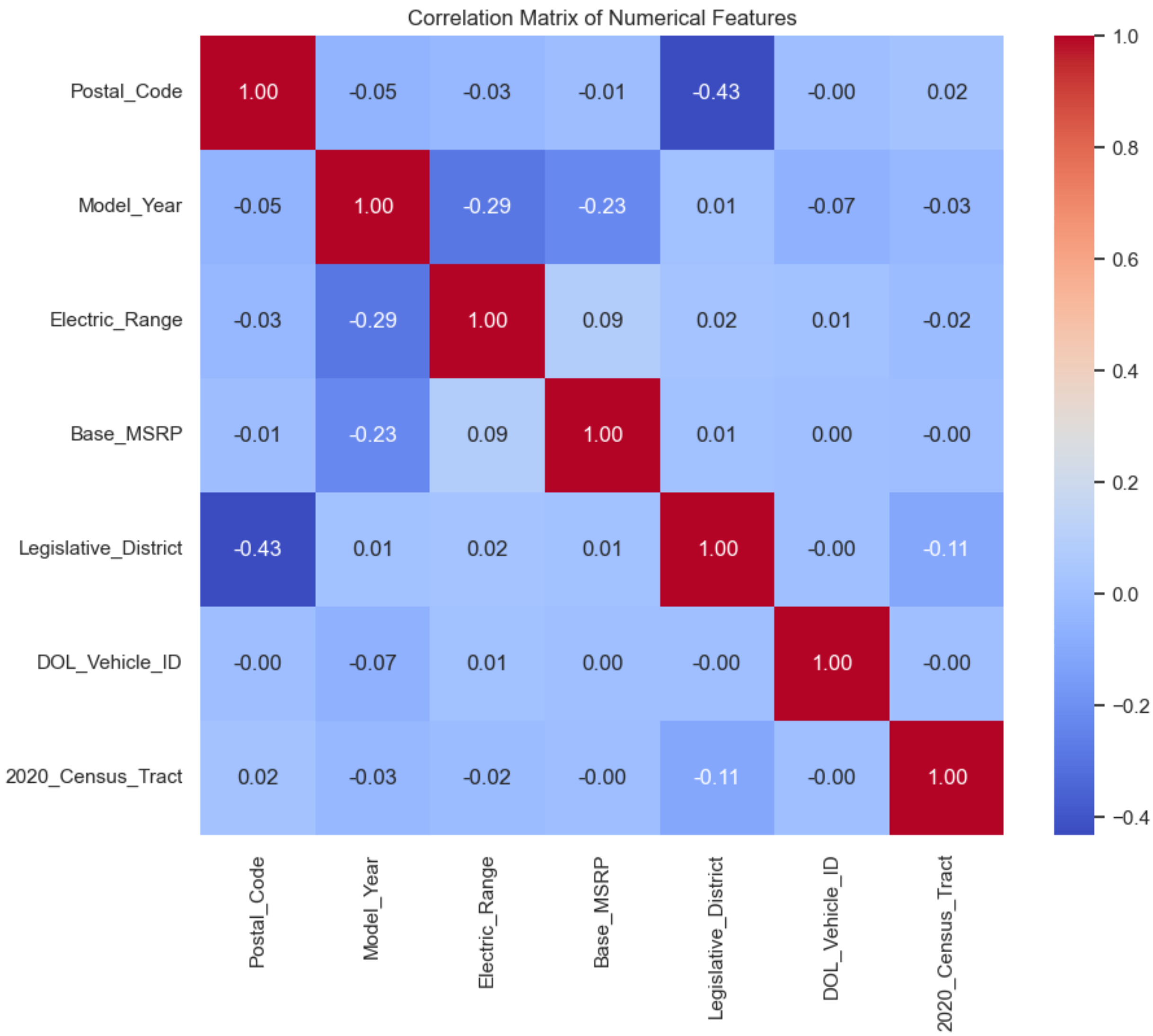

```
<Figure size 1500x800 with 0 Axes>
```

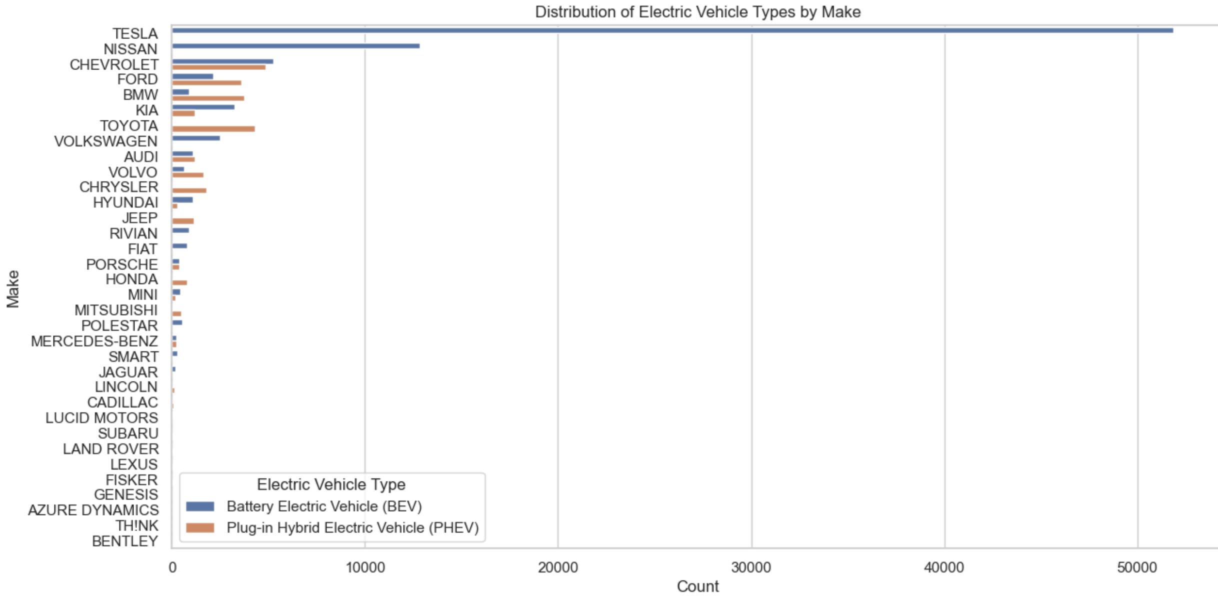



In [26]:
```
# 5. Correlation matrix for numerical variables
plt.figure(figsize=(10, 8))
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix of Numerical Features')
plt.show()

# 6. Distribution of Electric Vehicle Types by Make
```

```
plt.figure(figsize=(14, 7))
sns.countplot(y='Make', hue='Electric_Vehicle_Type', data=df, order=df['Make'].value_c
plt.title('Distribution of Electric Vehicle Types by Make')
plt.xlabel('Count')
plt.ylabel('Make')
plt.legend(title='Electric Vehicle Type')
plt.show()
```

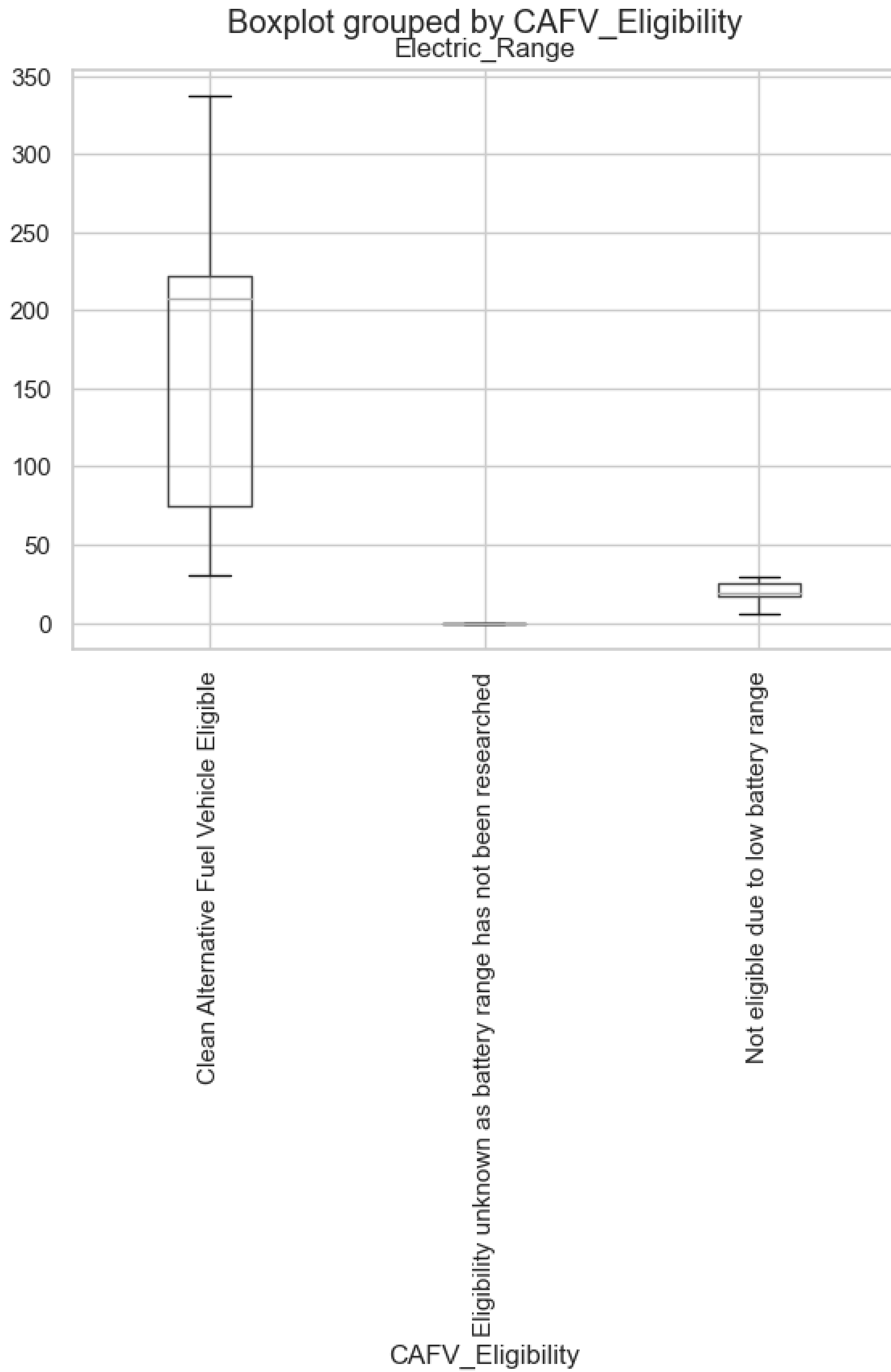
```
C:\Users\Prime\AppData\Local\Temp\ipykernel_11660\3143839223.py:3: FutureWarning: The
default value of numeric_only in DataFrame.corr is deprecated. In a future version, i
t will default to False. Select only valid columns or specify the value of numeric_on
ly to silence this warning.
  correlation_matrix = df.corr()
```






In [ ]:

In [27]:
```python
# Assuming 'df' is your DataFrame
df.boxplot(by="CAFV_Eligibility", column=['Electric_Range'])

# Rotate x-axis labels by 90 degrees
plt.xticks(rotation=90)

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




```
In [ ]:
```

# Task 2: Create a Choropleth using

# plotly.express to display the number of EV vehicles based on location

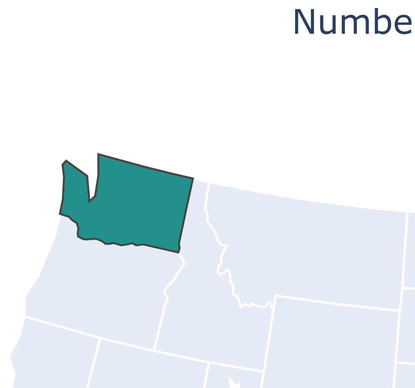
In [28]:
```
import plotly.express as px
```

In [29]:
```
ev_count_by_state = df.groupby('State').size().reset_index(name='Number_of_EV_Vehicles
ev_count_by_state
```

Out[29]:

| | State | Number_of_EV_Vehicles |
|---|---|---|
| **0** | WA | 112152 |

In [30]:
```
# Count the number of EVs per state
ev_count_by_state = df['State'].value_counts().reset_index()
ev_count_by_state.columns = ['State', 'EV_Count']

# Create the Choropleth map
fig = px.choropleth(ev_count_by_state,
                    locations='State',
                    locationmode="USA-states",
                    color='EV_Count',
                    scope="usa",
                    color_continuous_scale="Viridis",
                    title="Number of Electric Vehicles by State")

# Update the layout
fig.update_layout(
    title_x=0.5,
    geo_scope='usa',
)

fig.show()

# Save the plot as an HTML file
fig.write_html("ev_choropleth_map.html")

print("Choropleth map has been created and saved as 'ev_choropleth_map.html'.")
print("\
Top 5 states by EV count:")
print(ev_count_by_state.head().to_string(index=False))
```




```
Choropleth map has been created and saved as 'ev_choropleth_map.html'.
Top 5 states by EV count:
State  EV_Count
   WA    112152
```

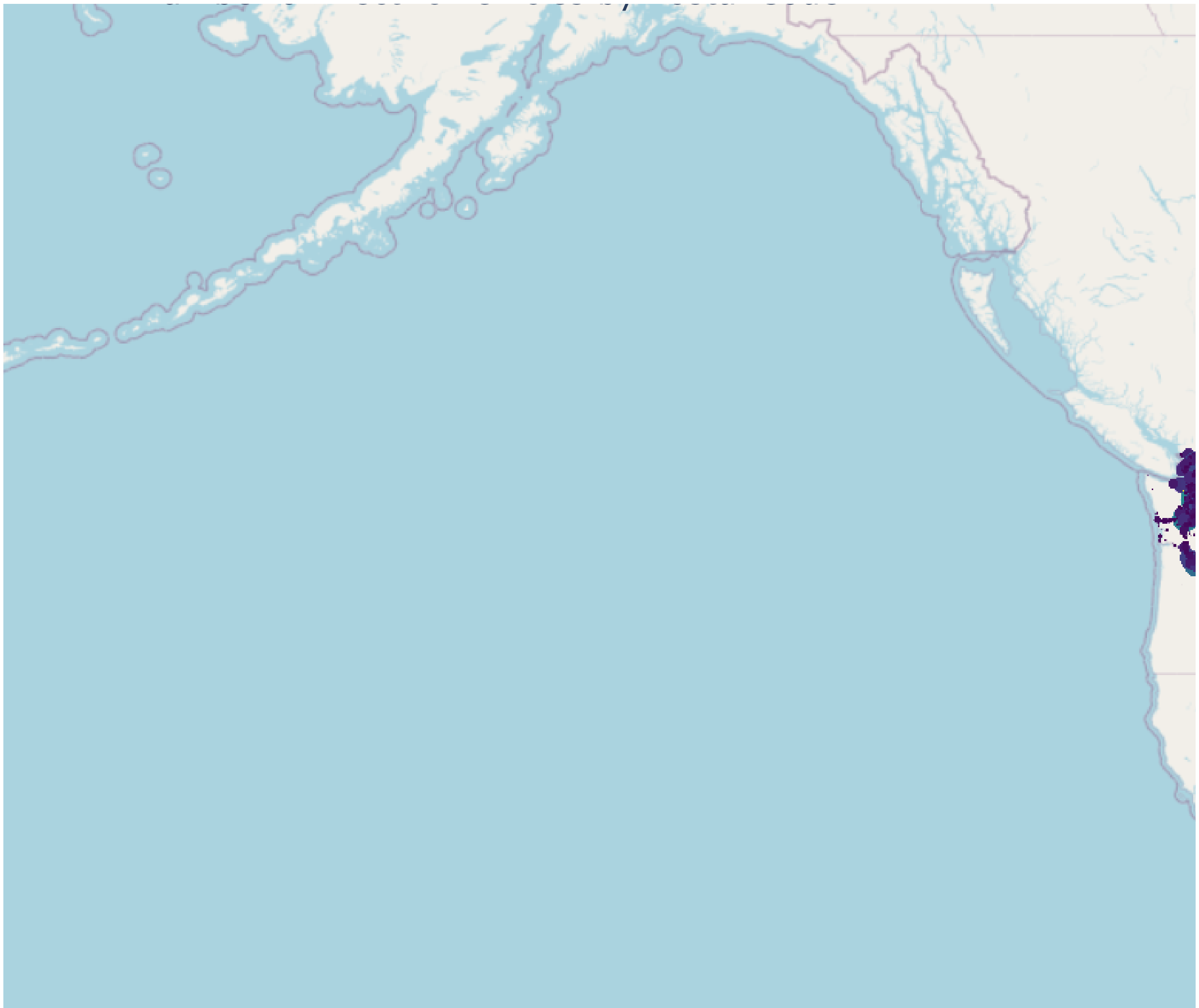
In [ ]:

```

```

In [31]:

```
import pandas as pd
import plotly.express as px

# Load the dataset
df = pd.read_csv('C:\\Users\\Prime\\Pictures\\EV.csv', encoding='ascii')

# Count the number of EVs per postal code
ev_count_by_postal = df['Postal Code'].value_counts().reset_index()
ev_count_by_postal.columns = ['Postal Code', 'EV_Count']

# Merge the count with the original dataframe to get location data
df_merged = df.merge(ev_count_by_postal, on='Postal Code')

# Extract latitude and longitude from the 'Vehicle Location' column
df_merged['Longitude'] = df_merged['Vehicle Location'].str.extract('POINT \(([-\d.]+)
df_merged['Latitude'] = df_merged['Vehicle Location'].str.extract(' ([-\d.]+)\)')

# Convert to numeric
df_merged['Longitude'] = pd.to_numeric(df_merged['Longitude'])
df_merged['Latitude'] = pd.to_numeric(df_merged['Latitude'])
```

```
# Create the scatter plot on a map
fig = px.scatter_mapbox(df_merged,
                        lat='Latitude',
                        lon='Longitude',
                        color='EV_Count',
                        size='EV_Count',
                        hover_name='Postal Code',
                        hover_data=['City', 'State', 'EV_Count'],
                        color_continuous_scale="Viridis",
                        size_max=15,
                        zoom=3,
                        title="Number of Electric Vehicles by Postal Code")

fig.update_layout(mapbox_style="open-street-map")
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})

# Save the plot as an HTML file
fig.write_html("ev_postal_code_map.html")

fig.show()

print("Scatter map based on postal codes has been created and saved as 'ev_postal_code
print("\
Top 10 postal codes by EV count:")
print(ev_count_by_postal.head(10).to_string(index=False))

# Display some statistics
print("\
Total number of unique postal codes:", len(ev_count_by_postal))
print("Average number of EVs per postal code:", round(ev_count_by_postal['EV_Count'].m
print("Median number of EVs per postal code:", ev_count_by_postal['EV_Count'].median()
print("Maximum number of EVs in a single postal code:", ev_count_by_postal['EV_Count']
```



```
Scatter map based on postal codes has been created and saved as 'ev_postal_code_map.h
tml'.
Top 10 postal codes by EV count:
 Postal Code  EV_Count
       98052      2916
       98033      2059
       98004      2001
       98115      1880
       98006      1852
       98012      1850
       98072      1661
       98040      1639
       98074      1594
       98034      1578
Total number of unique postal codes: 773
Average number of EVs per postal code: 145.71
Median number of EVs per postal code: 7.0
Maximum number of EVs in a single postal code: 2916
```

In [ ]:

# Task 3: Create a Racing Bar Plot to display the animation of EV Make and its count each

# year.v

In [ ]:

In [ ]:

In [55]:

```
Requirement already satisfied: bar-chart-race in c:\users\prime\anaconda3\lib\site-pa
ckages (0.1.0)
Requirement already satisfied: pandas>=0.24 in c:\users\prime\anaconda3\lib\site-pack
ages (from bar-chart-race) (1.5.3)
Requirement already satisfied: matplotlib>=3.1 in c:\users\prime\anaconda3\lib\site-p
ackages (from bar-chart-race) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\prime\anaconda3\lib\site-
packages (from matplotlib>=3.1->bar-chart-race) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\prime\anaconda3\lib\site-pack
ages (from matplotlib>=3.1->bar-chart-race) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\prime\anaconda3\lib\site
-packages (from matplotlib>=3.1->bar-chart-race) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\prime\anaconda3\lib\site
-packages (from matplotlib>=3.1->bar-chart-race) (1.4.4)
Requirement already satisfied: numpy>=1.20 in c:\users\prime\anaconda3\lib\site-packa
ges (from matplotlib>=3.1->bar-chart-race) (1.24.3)
Requirement already satisfied: packaging>=20.0 in c:\users\prime\anaconda3\lib\site-p
ackages (from matplotlib>=3.1->bar-chart-race) (23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\prime\anaconda3\lib\site-pac
kages (from matplotlib>=3.1->bar-chart-race) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\prime\anaconda3\lib\site-
packages (from matplotlib>=3.1->bar-chart-race) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\prime\anaconda3\lib\s
ite-packages (from matplotlib>=3.1->bar-chart-race) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\prime\anaconda3\lib\site-pack
ages (from pandas>=0.24->bar-chart-race) (2022.7)
Requirement already satisfied: six>=1.5 in c:\users\prime\anaconda3\lib\site-packages
(from python-dateutil>=2.7->matplotlib>=3.1->bar-chart-race) (1.16.0)
```

In [ ]:

In [60]:

Out[60]:

| | VIN (1-10) | County | City | State | Postal Code | Model Year | Make | Model | Electric Vehicle Type | Al I |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | JTMEB3FV6N | Monroe | Key West | FL | 33040 | 2022 | TOYOTA | RAV4 PRIME | Plug-in Hybrid Electric Vehicle (PHEV) | A Fu |
| **1** | 1G1RD6E45D | Clark | Laughlin | NV | 89029 | 2013 | CHEVROLET | VOLT | Plug-in Hybrid Electric Vehicle (PHEV) | A Fu |
| **2** | JN1AZ0CP8B | Yakima | Yakima | WA | 98901 | 2011 | NISSAN | LEAF | Battery Electric Vehicle (BEV) | A Fu |
| **3** | 1G1FW6S08H | Skagit | Concrete | WA | 98237 | 2017 | CHEVROLET | BOLT EV | Battery Electric Vehicle (BEV) | A Fu |
| **4** | 3FA6P0SU1K | Snohomish | Everett | WA | 98201 | 2019 | FORD | FUSION | Plug-in Hybrid Electric Vehicle (PHEV) | Nc d |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **112629** | 7SAYGDEF2N | King | Duvall | WA | 98019 | 2022 | TESLA | MODEL Y | Battery Electric Vehicle (BEV) | |
| **112630** | 1N4BZ1CP7K | San Juan | Friday Harbor | WA | 98250 | 2019 | NISSAN | LEAF | Battery Electric Vehicle (BEV) | A Fu |
| **112631** | 1FMCU0KZ4N | King | Vashon | WA | 98070 | 2022 | FORD | ESCAPE | Plug-in Hybrid Electric Vehicle (PHEV) | A Fu |
| **112632** | KNDCD3LD4J | King | Covington | WA | 98042 | 2018 | KIA | NIRO | Plug-in Hybrid Electric Vehicle (PHEV) | Nc d |

| | VIN (1-10) | County | City | State | Postal Code | Model Year | Make | Model | Electric Vehicle Type | Al |
|---|---|---|---|---|---|---|---|---|---|---|
| **112633** | YV4BR0CL8N | King | Covington | WA | 98042 | 2022 | VOLVO | XC90 | Plug-in Hybrid Electric Vehicle (PHEV) | Nc d |

112634 rows × 17 columns

In [62]:

In [63]:

In [85]:
```
!pip install bar_chart_race
```

```
Requirement already satisfied: bar_chart_race in c:\users\prime\anaconda3\lib\site-pa
ckages (0.1.0)
Requirement already satisfied: pandas>=0.24 in c:\users\prime\anaconda3\lib\site-pack
ages (from bar_chart_race) (1.5.3)
Requirement already satisfied: matplotlib>=3.1 in c:\users\prime\anaconda3\lib\site-p
ackages (from bar_chart_race) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\prime\anaconda3\lib\site-
packages (from matplotlib>=3.1->bar_chart_race) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\prime\anaconda3\lib\site-pack
ages (from matplotlib>=3.1->bar_chart_race) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\prime\anaconda3\lib\site
-packages (from matplotlib>=3.1->bar_chart_race) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\prime\anaconda3\lib\site
-packages (from matplotlib>=3.1->bar_chart_race) (1.4.4)
Requirement already satisfied: numpy>=1.20 in c:\users\prime\anaconda3\lib\site-packa
ges (from matplotlib>=3.1->bar_chart_race) (1.24.3)
Requirement already satisfied: packaging>=20.0 in c:\users\prime\anaconda3\lib\site-p
ackages (from matplotlib>=3.1->bar_chart_race) (23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\prime\anaconda3\lib\site-pac
kages (from matplotlib>=3.1->bar_chart_race) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\prime\anaconda3\lib\site-
packages (from matplotlib>=3.1->bar_chart_race) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\prime\anaconda3\lib\s
ite-packages (from matplotlib>=3.1->bar_chart_race) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\prime\anaconda3\lib\site-pack
ages (from pandas>=0.24->bar_chart_race) (2022.7)
Requirement already satisfied: six>=1.5 in c:\users\prime\anaconda3\lib\site-packages
(from python-dateutil>=2.7->matplotlib>=3.1->bar_chart_race) (1.16.0)
```

In [93]:
```
import bar_chart_race as bcr
import warnings
```

In [92]:
```
df['Model Year'] = df['Model Year'].astype(str)

# Group the data by 'Model Year' and 'Make', then count the occurrences
grouped_data = df.groupby(['Model Year', 'Make']).size().reset_index(name='Count')
```

```
# Pivot the data to have 'Model Year' as the index and 'Make' as columns
pivoted_data = grouped_data.pivot(index='Model Year', columns='Make', values='Count')

# Fill missing values with 0 (for years where some makes might have no entries)
pivoted_data = pivoted_data.fillna(0)

# Create the bar chart race animation and save it as a GIF
bcr.bar_chart_race(df=pivoted_data, filename='EV_racing_bar_plot.gif',
                   orientation='h', sort='desc', n_bars=10,
                   title='EV Make Count Over the Years', filter_column_colors=True, pe
```

```
C:\Users\Prime\anaconda3\Lib\site-packages\bar_chart_race\_make_chart.py:286: UserWar
ning:

FixedFormatter should only be used together with FixedLocator

C:\Users\Prime\anaconda3\Lib\site-packages\bar_chart_race\_make_chart.py:287: UserWar
ning:

FixedFormatter should only be used together with FixedLocator

MovieWriter imagemagick unavailable; using Pillow instead.
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: