Electric Vehicle Analysis:

This project focuses on analyzing the registration data of Battery Electric Vehicles (BEVs) and Plug-in Hybrid Electric Vehicles (PHEVs) in Washington State, aiming to uncover trends, patterns, and adoption dynamics to gain insights into the growth and development of sustainable transportation.

Problem Statement:

Conduct a Data Analysis on Electric Vehicles using the provided dataset to uncover insights.

Task 1:

> Perform Exploratory Data Analysis (EDA), including both Univariate and Bivariate analyses, to explore patterns and trends within the dataset.

Task 2:

> Create a Choropleth map using Plotly Express to visualize the distribution of electric vehicles based on location.

Task 3:

> Develop a Racing Bar Plot to animate the changes in the number of electric vehicle makes over the years.

```
# importing the required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

warnings.filterwarnings('ignore')

# load the dataset
PATH = r'/content/drive/MyDrive/Colab Notebooks/data/ev_dataset.csv'
ev_df = pd.read_csv(PATH)

# Overview of the dataset
ev_df.head()
```

```
{"type":"dataframe","variable_name":"ev_df"}
```

```
ev_df.shape
```

```
(112634, 17)
```

```
ev_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
```

```
ev_df.describe().T
```

```
{"summary":"{\n  \"name\": \"ev_df\",\n  \"rows\": 7,\n  \"fields\":
[\n    {\n      \"column\": \"count\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 108.09783928063898,\n
\"min\": 112348.0,\n        \"max\": 112634.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n
112348.0,\n          112634.0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"mean\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 20007020693.25397,\n        \"min\":
29.805604016092854,\n        \"max\": 52966495754.096825,\n
\"num_unique_values\": 7,\n        \"samples\": [\n
98156.22684979669,\n          2019.0033648809417\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"std\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 637243096.1203485,\n
\"min\": 2.8923640117558467,\n        \"max\": 1699104499.621227,\n
\"num_unique_values\": 7,\n        \"samples\": [\n
2648.733064096689,\n          2.8923640117558467\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"min\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 416138878.1723344,\n
\"min\": 0.0,\n        \"max\": 1101001400.0,\n
\"num_unique_values\": 6,\n        \"samples\": [\n          1730.0,\n
1997.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"25%\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 20035313918.826183,\n        \"min\": 0.0,\n        \"max\":
53033008500.0,\n        \"num_unique_values\": 6,\n
\"samples\": [\n          98052.0,\n          2017.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"50%\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 20032603538.483376,\n
\"min\": 0.0,\n        \"max\": 53033029305.0,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          98119.0,\
n          2020.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"75%\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 20038529056.330368,\n        \"min\": 0.0,\n        \"max\":
53053072506.0,\n        \"num_unique_values\": 7,\n
\"samples\": [\n          98370.0,\n          2022.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"max\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 21148987088.325718,\n
\"min\": 49.0,\n        \"max\": 56033000100.0,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          99701.0,\
n          2023.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

```python
ev_df.isna().sum().sort_values(ascending=False)
```

```
Electric Utility                                         443
Legislative District                                     286
Vehicle Location                                          24
Model                                                     20
```

```
VIN (1-10)                                             0
Clean Alternative Fuel Vehicle (CAFV) Eligibility      0
DOL Vehicle ID                                         0
Base MSRP                                              0
Electric Range                                         0
Electric Vehicle Type                                  0
County                                                 0
Make                                                   0
Model Year                                             0
Postal Code                                            0
State                                                  0
City                                                   0
2020 Census Tract                                      0
dtype: int64
```

Data Cleaning

```
## Copy the dataset
df = ev_df.copy()

df.isna().sum().sort_values(ascending=False)
```

```
Electric Utility                                     443
Legislative District                                 286
Vehicle Location                                      24
Model                                                 20
VIN (1-10)                                             0
Clean Alternative Fuel Vehicle (CAFV) Eligibility      0
DOL Vehicle ID                                         0
Base MSRP                                              0
Electric Range                                         0
Electric Vehicle Type                                  0
County                                                 0
Make                                                   0
Model Year                                             0
Postal Code                                            0
State                                                  0
City                                                   0
2020 Census Tract                                      0
dtype: int64
```

```
df.dropna(inplace=True)
df.reset_index(drop=True, inplace=True)

df.isna().sum().sort_values(ascending=False)
```

```
VIN (1-10)                                           0
Clean Alternative Fuel Vehicle (CAFV) Eligibility    0
Electric Utility                                     0
Vehicle Location                                     0
```

```
DOL Vehicle ID                                       0
Legislative District                                 0
Base MSRP                                            0
Electric Range                                       0
Electric Vehicle Type                                0
County                                               0
Model                                                0
Make                                                 0
Model Year                                           0
Postal Code                                          0
State                                                0
City                                                 0
2020 Census Tract                                    0
dtype: int64
```

```
df.shape
```

```
(112152, 17)
```

```
total_rows = ev_df.shape[0]
rows_after_removal = df.shape[0]
removed_rows = total_rows - rows_after_removal

removed_rows_percentage = (removed_rows / total_rows) * 100

print(f"Total rows              :: {total_rows}")
print(f"Rows after removal      :: {rows_after_removal}")
print(f"Removed rows            :: {removed_rows}")
print(f"Removed rows percentage :: {removed_rows_percentage:.3f} %")
```

```
Total rows              :: 112634
Rows after removal      :: 112152
Removed rows            :: 482
Removed rows percentage :: 0.428 %
```

```
# Checking for duplicates in the dataset
df.duplicated().sum()
```

```
0
```

```
# Convert columns like 'Model Year' to numerical and 'Postal Code' to
strings if necessary
df['Model Year'] = pd.to_numeric(df['Model Year'], errors='coerce')
df['Postal Code'] = df['Postal Code'].astype(str)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112152 entries, 0 to 112151
Data columns (total 17 columns):
 #   Column                                             Non-Null Count
```

```
Dtype
---  ------                                             --------------
-----
 0   VIN (1-10)                                         112152 non-
null  object
 1   County                                             112152 non-
null  object
 2   City                                               112152 non-
null  object
 3   State                                              112152 non-
null  object
 4   Postal Code                                        112152 non-
null  object
 5   Model Year                                         112152 non-
null  int64
 6   Make                                               112152 non-
null  object
 7   Model                                              112152 non-
null  object
 8   Electric Vehicle Type                              112152 non-
null  object
 9   Clean Alternative Fuel Vehicle (CAFV) Eligibility  112152 non-
null  object
 10  Electric Range                                     112152 non-
null  int64
 11  Base MSRP                                          112152 non-
null  int64
 12  Legislative District                               112152 non-
null  float64
 13  DOL Vehicle ID                                     112152 non-
null  int64
 14  Vehicle Location                                   112152 non-
null  object
 15  Electric Utility                                   112152 non-
null  object
 16  2020 Census Tract                                  112152 non-
null  int64
dtypes: float64(1), int64(5), object(11)
memory usage: 14.5+ MB
```

```
# Drop irrelevant columns for analysis (e.g., VIN, DOL Vehicle ID)
df.drop(columns=['VIN (1-10)', 'DOL Vehicle ID'], inplace=True)

# Inspect the cleaned data
print(df.info())
print(df.describe().T)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112152 entries, 0 to 112151
Data columns (total 15 columns):
 #   Column                                             Non-Null Count
Dtype
---  ------                                             --------------
-----
 0   County                                             112152 non-
null  object
 1   City                                               112152 non-
null  object
 2   State                                              112152 non-
null  object
 3   Postal Code                                        112152 non-
null  object
 4   Model Year                                         112152 non-
null  int64
 5   Make                                               112152 non-
null  object
 6   Model                                              112152 non-
null  object
 7   Electric Vehicle Type                              112152 non-
null  object
 8   Clean Alternative Fuel Vehicle (CAFV) Eligibility  112152 non-
null  object
 9   Electric Range                                     112152 non-
null  int64
 10  Base MSRP                                          112152 non-
null  int64
 11  Legislative District                               112152 non-
null  float64
 12  Vehicle Location                                   112152 non-
null  object
 13  Electric Utility                                   112152 non-
null  object
 14  2020 Census Tract                                  112152 non-
null  int64
dtypes: float64(1), int64(4), object(10)
memory usage: 12.8+ MB
None
                         count          mean           std
min  \
Model Year            112152.0  2.019004e+03  2.891859e+00
1.997000e+03
Electric Range        112152.0  8.782965e+01  1.023366e+02
0.000000e+00
Base MSRP             112152.0  1.793882e+03  1.078526e+04
0.000000e+00
Legislative District  112152.0  2.981770e+01  1.469873e+01
1.000000e+00
2020 Census Tract     112152.0  5.303958e+10  1.617788e+07
5.300195e+10
```

```
                               25%           50%           75%
max
Model Year              2.017000e+03  2.020000e+03  2.022000e+03
2.023000e+03
Electric Range          0.000000e+00  3.200000e+01  2.080000e+02
3.370000e+02
Base MSRP               0.000000e+00  0.000000e+00  0.000000e+00
8.450000e+05
Legislative District    1.800000e+01  3.400000e+01  4.300000e+01
4.900000e+01
2020 Census Tract       5.303301e+10  5.303303e+10  5.305307e+10
5.307794e+10
```

Univariate Analysis

Analyzing the Numerical Columns

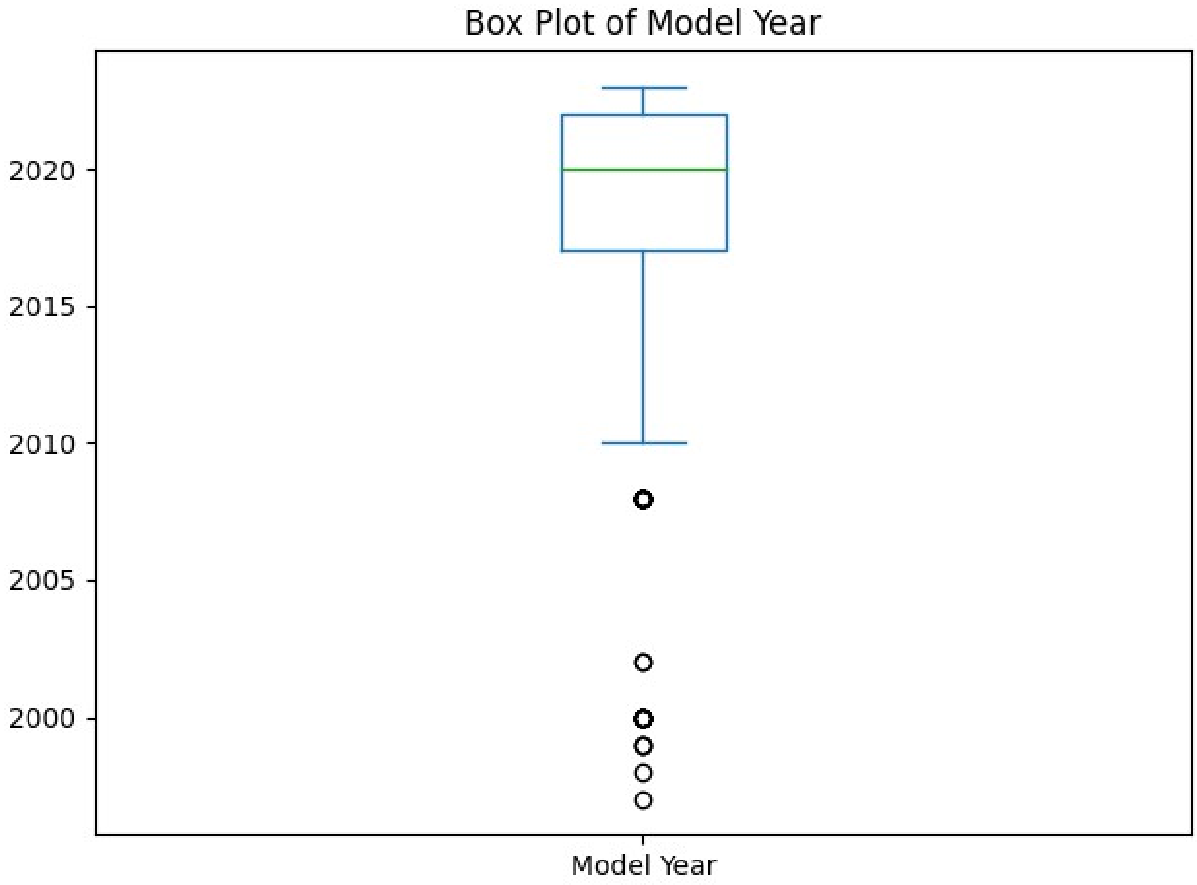
```
# Analysing the numerical columns
numerical_df = df.select_dtypes(include=['int64', 'float64'])

def univariate_analysis_numeric(df:pd.DataFrame) -> None:
  for col_name in df:
    print("*"*10, col_name, "*"*10)
    print(df[col_name].agg(['min', 'max', 'mean', 'median', 'std']))
    print()

univariate_analysis_numeric(numerical_df)

********** 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
```

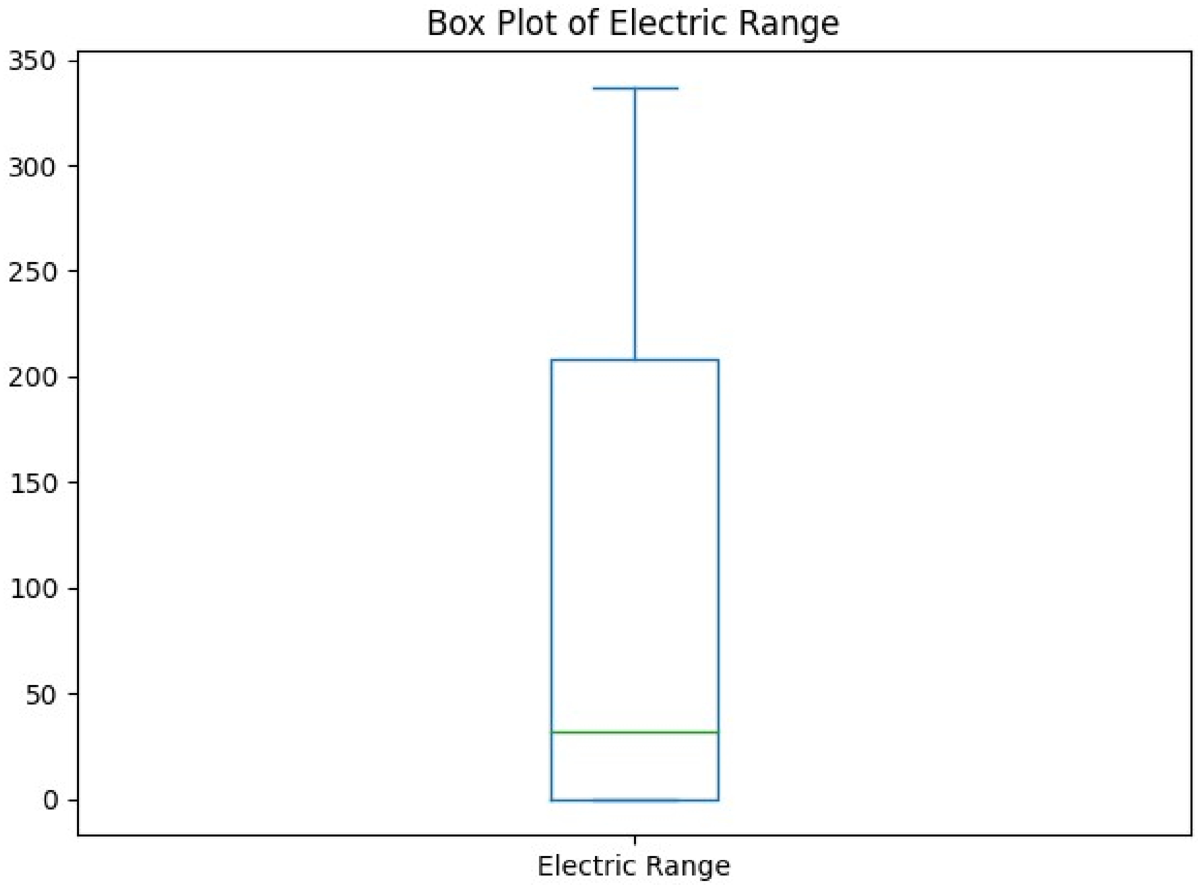
```
median          0.000000
std         10785.259118
Name: Base MSRP, dtype: float64

********** Legislative District **********
min        1.000000
max       49.000000
mean      29.817703
median    34.000000
std       14.698726
Name: Legislative District, dtype: float64

********** 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
```

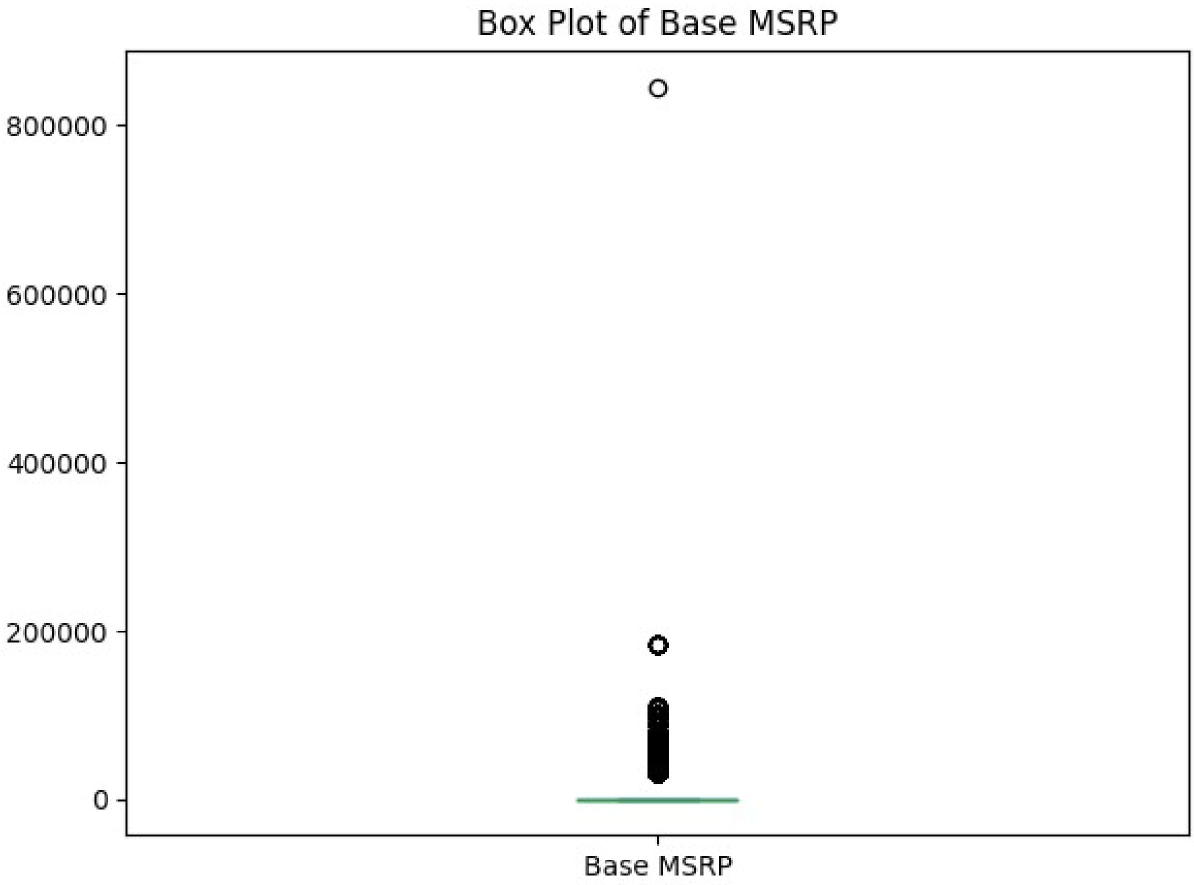
Visualization of numeric columns -

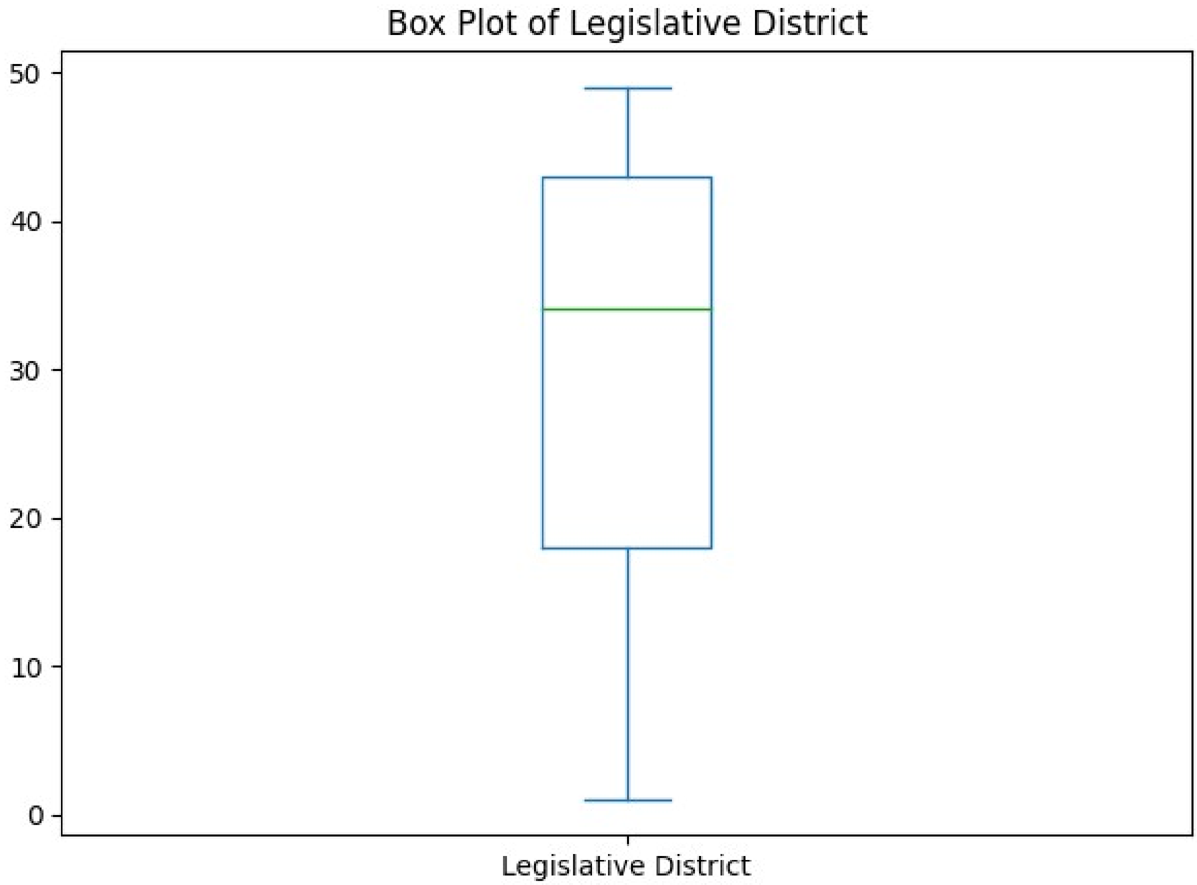
- To understand how the data is distributed can be effectively visualized using a box plot.
- It displays the spread of data, highlighting the median, quartiles, and potential outliers, making it a great choice for summarizing distributions in a single plot.

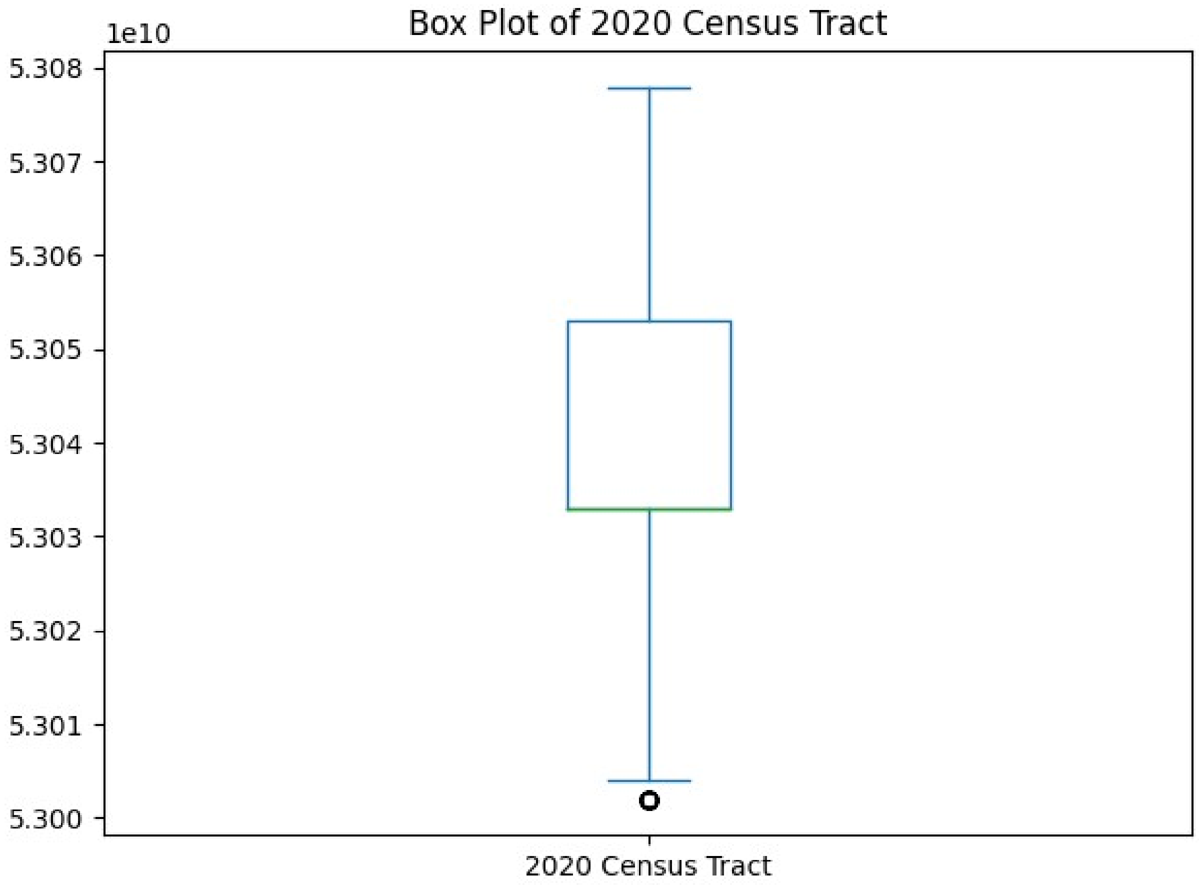
```
def numerical_df_visualization(numeric_data):
    for col in numeric_data.columns:
        numeric_data[col].plot(kind="box")
        plt.title(f"Box Plot of {col}")
        plt.tight_layout()
        plt.show()


numerical_df_visualization(numerical_df)
```


Box Plot of Model Year
2020
2015
2010
2005
2000
Model Year

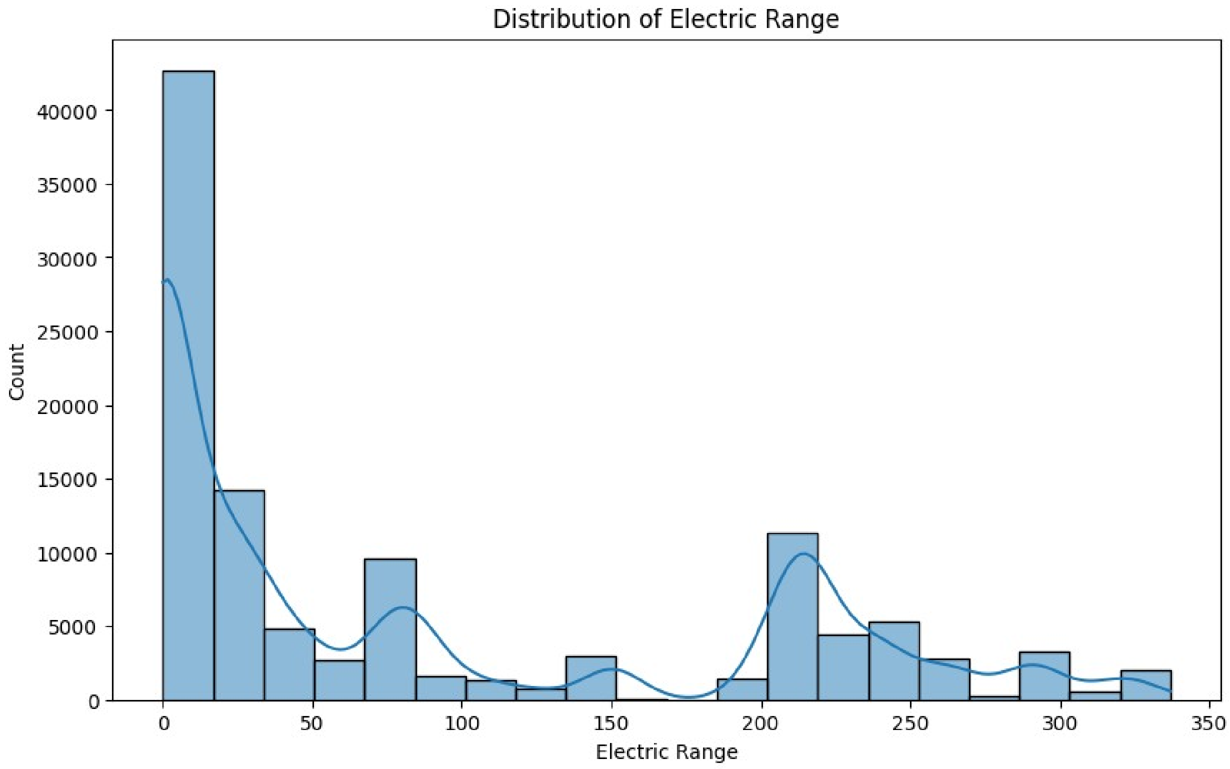

Box Plot of Electric Range
350
300
250
200
150
100
50
0
Electric Range


Box Plot of Base MSRP
800000
600000
400000
200000
0
Base MSRP


Box Plot of Legislative District
50
40
30
20
10
0
Legislative District



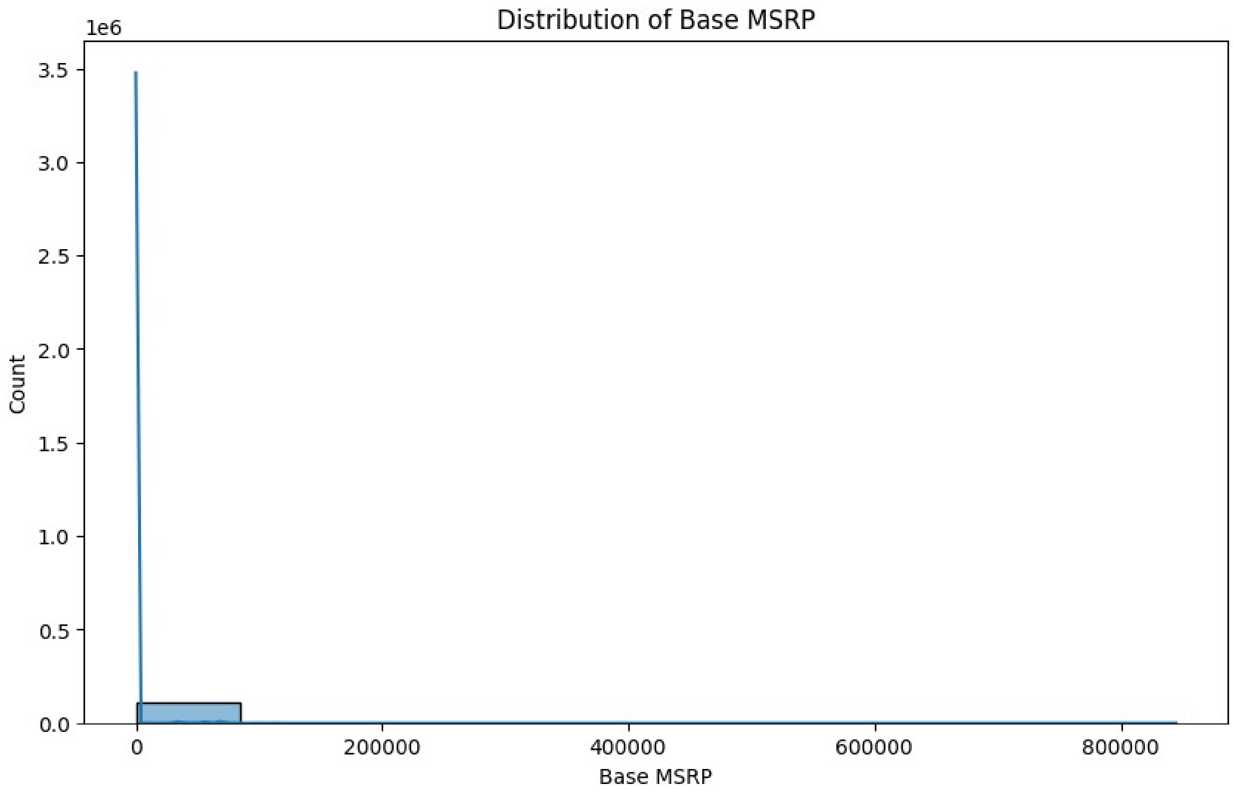
```
# Univariate Analysis of Numerical Columns
plt.figure(figsize=(10, 6))
sns.histplot(df['Electric Range'], bins=20, kde=True)
plt.title('Distribution of Electric Range')
plt.show()
```



Insight:

- Most vehicles have a range of 0-50 miles, indicating limited electric-only driving capability.
- Popular electric ranges are around 50, 100, and 200 miles, reflecting different vehicle tiers.
- A smaller number of vehicles exceed 200 miles on a single charge.
- The distribution is right-skewed, with most vehicles concentrated in the lower range.

```
plt.figure(figsize=(10, 6))
sns.histplot(df['Base MSRP'], bins=10, kde=True)
plt.title('Distribution of Base MSRP')
plt.show()
```

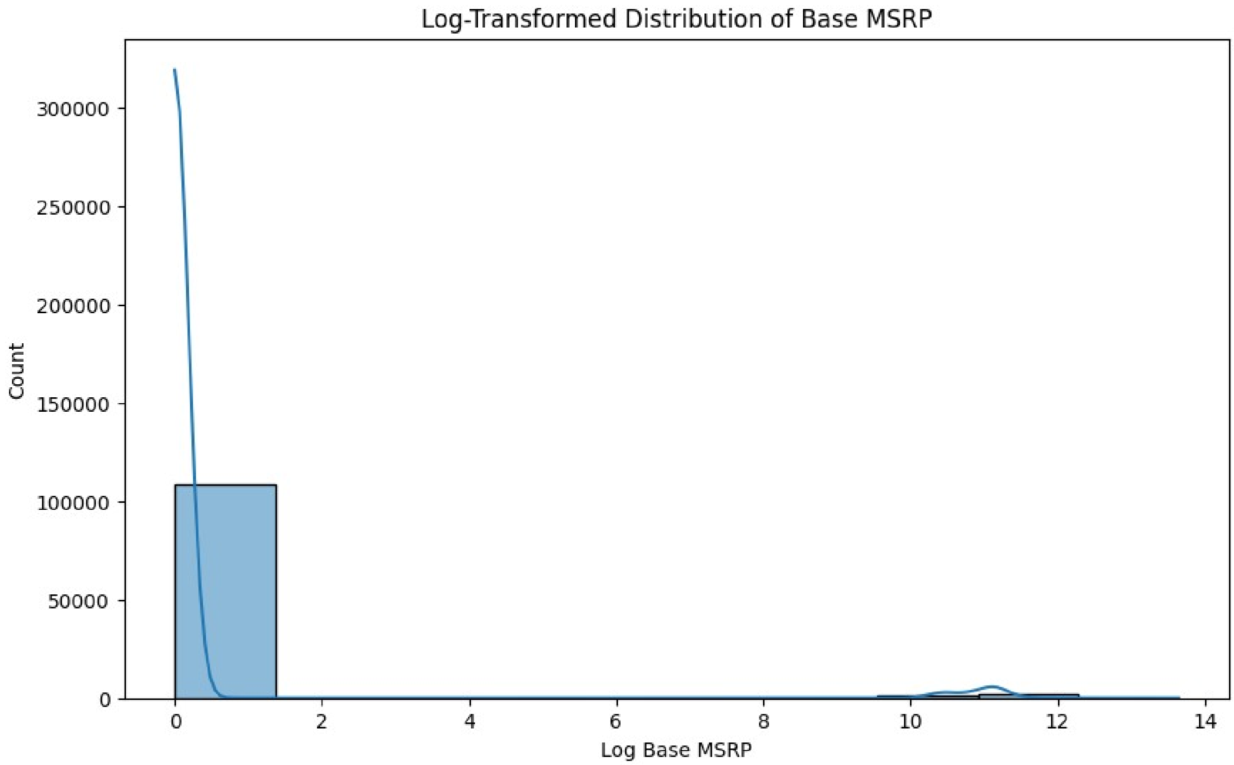


Insight :

- The graph shows the distribution of Base MSRP, which is heavily right-skewed with most values concentrated at lower prices.
- There are extreme outliers extending beyond $800,000.
- But, the majority of vehicles are likely priced below $50,000, though exact details are hard to see due to the wide scale.

Therefore, we have to transform the data (e.g., log transformation) to better visualize the distribution.

```
# Apply log transformation (use np.log1p to handle zeros)
df['Log Base MSRP'] = np.log1p(df['Base MSRP'])

# Plot the log-transformed data
plt.figure(figsize=(10, 6))
sns.histplot(df['Log Base MSRP'], bins=10, kde=True)
plt.title('Log-Transformed Distribution of Base MSRP')
plt.show()
```



Insights :

- Most values remain concentrated in the lower MSRP range, as indicated by the sharp decline in count at the lower end of the log scale.

Analyzing the Categorical columns

```
categorical_df = ev_df.select_dtypes(include='object')

categorical_df.head()

{"type":"dataframe","variable_name":"categorical_df"}

def categorical_univariate_analysis(cat_data):
    for i in cat_data:
        print("*"*10, i ,"*"*10)
        print("Unique values:",cat_data[i].unique())
        print("Nunique values:",cat_data[i].nunique())

print("--------------------------------------------------------")

categorical_univariate_analysis(categorical_df)

********** VIN (1-10) **********
Unique values: ['JTMEB3FV6N' '1G1RD6E45D' 'JN1AZ0CP8B' ...
'KMHE14L25K' 'WA1LAAGE5M'
 'YV4ED3GM0P']
Nunique values: 7548
```

```
--------------------------------------------------------
********** County **********
Unique values: ['Monroe' 'Clark' 'Yakima' 'Skagit' 'Snohomish'
'Island' 'Thurston'
 'Grant' 'St. Clair' 'Pierce' 'Saratoga' 'Stevens' 'King' 'Kitsap'
 'Newport News' 'Jackson' 'Whitman' 'Lake' 'Spokane' 'Clallam'
'Cowlitz'
 'Kittitas' 'Grays Harbor' 'Chelan' 'Whatcom' 'Benton' 'Walla Walla'
 'Mason' 'San Juan' 'Lewis' 'Jefferson' 'Douglas' 'Klickitat' 'Geary'
 'Skamania' 'Fairfax' 'Adams' 'Franklin' 'Okanogan' 'Sonoma' 'Asotin'
 'Ferry' 'Pacific' 'Riverside' 'Orange' 'Columbia' 'Wahkiakum'
 'Leavenworth' 'Contra Costa' 'Howard' 'Larimer' 'District of
Columbia'
 'Washington' 'Tipton' 'San Diego' 'Sumter' "Prince George's" 'New
Haven'
 'Lincoln' 'Las Animas' 'Frederick' 'Hidalgo' 'Pend Oreille' 'Bexar'
 'Garfield' 'Pennington' 'Honolulu' 'Anne Arundel' 'Montgomery'
'Houston'
 'Charleston' 'Monterey' 'Kern' 'Napa' 'Loudoun' 'Harrison' 'Pulaski'
 'Cumberland' 'Los Angeles' 'Ray' 'Salt Lake' 'Solano' 'Allegheny'
 'Carroll' 'Clackamas' 'Kent' 'Harris' 'Ventura' 'Hamilton' 'Polk'
 'Placer' 'Calvert' 'Sheridan' 'Kings' 'El Paso' 'Portsmouth' 'Elmore'
 'Santa Clara' 'Pinal' 'Wayne' 'Alameda' 'Maricopa' 'Stafford'
 'Santa Barbara' 'Fairbanks North Star' 'Plaquemines' 'Rock Island'
 'Chaves' 'Palm Beach' 'Danville' 'Galveston' 'Virginia Beach'
'Suffolk'
 'Louisa' 'Hillsborough' 'Denton' 'Bell' 'Norfolk' 'Okaloosa'
'Rockdale'
 'Cook' 'Chesapeake' 'Alexandria' 'Charles' 'Boulder' 'Beaufort'
 'St. Louis' "St. Mary's" 'Marin' 'Arapahoe' 'Laramie' 'Multnomah'
'Hoke'
 'Sarasota' 'Santa Cruz' 'Queens' 'Wichita' 'San Bernardino' 'Oldham'
 'Onslow' 'Arlington' 'Sarpy' 'Moore' 'Sevier' 'Bartow' 'Sacramento'
 'Camden' 'Hennepin' 'Middlesex' 'New London' 'Platte' 'Penobscot'
 'Nassau' 'Richmond' 'Newport' 'Rockingham' 'San Mateo' 'DeKalb'
'Kauai'
 'Burlington' 'St. Tammany' 'Bryan' 'Dorchester' 'Williams'
'Kootenai']
Nunique values: 165
--------------------------------------------------------
********** City **********
Unique values: ['Key West' 'Laughlin' 'Yakima' 'Concrete' 'Everett'
'Bothell' 'Mukilteo'
 'Clinton' 'Anacortes' 'Lacey' 'Moses Lake' 'Mascoutah' 'Rochester'
 'Burlington' 'Kapowsin' 'Marysville' 'Lynnwood' 'Greenfield Center'
 'Edmonds' 'Nine Mile Falls' 'Olympia' 'Seattle' 'Auburn' 'Langley'
 'Snohomish' 'Bremerton' 'Newport News' 'Altus' 'Pullman' 'Highland
Park'
 'Spokane' 'Suquamish' 'Monroe' 'Sequim' 'Keyport' 'Gurnee' 'Maple
```

```
Valley'
 'Kent' 'Lake Forest Park' 'Poulsbo' 'Redmond' 'Issaquah' 'Longview'
 'Tacoma' 'Ellensburg' 'Burien' 'Gig Harbor' 'South Hill' 'Sammamish'
 'Westport' 'Vancouver' 'Airway Heights' 'Mercer Island' 'Stanwood'
 'Tumwater' 'Bainbridge Island' 'Entiat' 'Lakewood' 'Lake Tapps'
 'Bellevue' 'Kirkland' 'Newcastle' 'Port Orchard' 'Bellingham'
'Richland'
 'Camano Island' 'Wenatchee' 'Lake Stevens' 'Roy' 'Des Moines'
'Renton'
 'Camas' 'Kennewick' 'Battle Ground' 'Bonney Lake' 'Walla Walla'
 'North Bend' 'Mount Vernon' 'Woodland' 'Woodinville' 'Allyn' 'Brier'
 'Snoqualmie' 'Fall City' 'Puyallup' 'Friday Harbor' 'Point Roberts'
 'Dupont' 'Castle Rock' 'Blaine' 'Morton' 'Port Townsend' 'Roslyn'
 'Kenmore' 'Covington' 'Federal Way' 'Silverdale' 'Medina' 'Shoreline'
 'Enumclaw' 'Orondo' 'Grandview' 'Mill Creek' 'Zillah' 'Edgewood'
'Vashon'
 'White Salmon' 'Normandy Park' 'Fircrest' 'East Wenatchee'
'Peshastin'
 'Grapeview' 'Steilacoom' 'Sumner' 'Junction City' 'Greenacres'
'Shelton'
 'Chehalis' 'Pacific Beach' 'Everson' 'Black Diamond' 'North
Bonneville'
 'Coupeville' 'Seabeck' 'Arlington' 'Alexandria' 'Palouse' 'Bow'
'Lakebay'
 'University Place' 'Clyde Hill' 'Cle Elum' 'Yacolt' 'Oak Harbor'
 'Goldendale' 'Port Hadlock' 'Acme' 'Ritzville' 'Union' 'Orting'
'Tahuya'
 'Fox Island' 'Moxee' 'Port Angeles' 'Spanaway' 'Lopez Island'
 'Hunts Point' 'Leavenworth' 'Seatac' 'Stevenson' 'Pasco' 'Yelm'
 'Tonasket' 'Liberty Lake' 'Hansville' 'Eastsound' 'Nordland'
'Touchet'
 'Spokane Valley' 'Tukwila' 'Selah' 'Fife' 'Lynden' 'Aberdeen'
 'Anderson Island' 'Orcas Is' 'Kingston' 'Randle' 'Sedro-Woolley'
 'Carnation' 'Belfair' 'Cheney' 'Elma' 'Olalla' 'Granite Falls'
'Ephrata'
 'Preston' 'Ridgefield' 'Mccleary' 'Ferndale' 'Mountlake Terrace'
 'Freeland' 'Sonoma' 'Yarrow Point' 'Rainier' 'Sunnyside' 'Salkum'
 'Colville' 'Duvall' 'Otis Orchards' 'Twisp' 'Eatonville' 'Chattaroy'
 'Ocean Shores' 'Washougal' 'Port Ludlow' 'Benton City' 'Clarkston'
 'Ravensdale' 'Kelso' 'Curlew' 'Deming' 'Prosser' 'Milton' 'Artondale'
 'Hoodsport' 'West Richland' 'Parkland' 'Chelan' 'Graham' 'Raymond'
 'Brush Prairie' 'Rock Island' 'La Conner' 'St John' 'Mead' 'Hoquiam'
 'Deer Park' 'Electric City' 'Chimacum' 'Burbank' 'Quincy' 'Omaha'
 'La Center' 'Ronald' 'Long Beach' 'Valley' 'Beaux Arts' 'Kalama'
 'Indianola' 'Winthrop' 'Wildomar' 'Aliso Viejo' 'Woodway' 'Buckley'
 'Montesano' 'Las Vegas' 'Dayton' 'Vaughn' 'Onalaska' 'Medical Lake'
 'Nooksack' 'Centralia' 'Sultan' 'Trout Lake' 'Seaview' 'Carson'
'Colbert'
 'Lummi Island' 'Newman Lake' 'Cathlamet' 'Veradale' 'Valleyford'
 'Cashmere' 'Ariel' 'Cosmopolis' 'Bz Corner' 'Ilwaco' 'Oakville'
'Algona'
 'Silverlake' 'Lopez Is' 'Winlock' 'Greenbank' 'Tenino' 'Royal City'
 'Tulalip' 'Fort Leavenworth' 'Custer' 'Moraga' 'College Place'
 'Underwood' 'Amboy' 'Bingen' 'Ryderwood' 'Clearlake' 'Naches'
'Surfside'
 'Olga' 'Ocean Park' 'Othello' 'Rosalia' 'Snoqualmie Pass' 'Timnath'
 'Republic' 'Washington' 'Keedysville' 'Atoka' 'San Diego' 'Sumter'
 'Upper Marlboro' 'Madison' 'Lincoln City' 'Grand Coulee' 'Trinidad'
 'Chewelah' 'Packwood' 'Thorp' 'Frederick' 'Malaga' 'Lind'
 'Joint Base Lewis Mcchord' 'Granger' 'Wilbur' 'Toledo' 'Pacific'
 'Toppenish' 'Eltopia' 'Sekiu' 'Sedro Woolley' 'Garfield' 'Lincoln'
 'Mcallen' 'Newport' 'Harrington' 'San Antonio' 'Ethel' 'Pomeroy'
 'Longbranch' 'Connell' 'Brinnon' 'Skykomish' 'Reardan' 'Maple Falls'
 'Rapid City' 'Coulee City' 'Dallesport' 'Vantage' 'Oroville' 'Manson'
 'Honolulu' 'Omak' 'Bridgeport Bar' 'Mesa' 'Odenton' 'Waterville'
 'Chinook' 'Gold Bar' 'Soap Lake' 'Nahcotta' 'Tieton' 'Silver Spring'
 'Warner Robins' 'Mattawa' 'Addy' 'Ruston' 'Loon Lake' 'Charleston
Afb'
 'Forks' 'Wapato' 'Naselle' 'Quilcene' 'Asotin' 'Monterey' 'Easton'
 'Fairchild Air Force Base' 'Ridgecrest' 'Skamokawa' 'Lilliwaup'
'Napa'
 'Aldie' 'Marlin' 'Warden' 'Biloxi' 'Seven Bays' 'Chula Vista'
 'Little Rock' 'Fayetteville' 'Kettle Falls' 'South Bend' 'Rayville'
 'Okanogan' 'Mansfield' 'Pateros' 'Sumas' 'Salt Lake City' 'McCleary'
 'North Las Vegas' 'Cusick' 'Vacaville' 'Fort Campbell' 'Wexford'
 'Ashford' 'Elk' 'Carbonado' 'Rockford' 'Lyle' 'Latah' 'Westminster'
 'Carlton' 'Darrington' 'West Linn' 'Mossyrock' 'Dover' 'Tumtum'
'Arnold'
 'Tavares' 'Houston' 'Ventura' 'Riverside' 'North Cove' 'Bay Center'
 'Brewster' 'Springdale' 'Cougar' 'Endicott' 'Inchelium' 'Wilkeson'
 'Cincinnati' 'Salem' 'Roseville' 'Chesapeake Beach' 'Frances'
'Colfax'
 'White Swan' 'Grayland' 'Rice' 'Sheridan' 'Neah Bay' 'Lemoore'
 'Montgomery' 'Colorado Springs' 'Davenport' 'Portsmouth' 'Lancaster'
 'Coulee Dam' 'Union Gap' 'Menifee' 'Shaw Island' 'Mountain Home Afb'
 'Marblemount' 'Santa Clara' 'Baring' 'Spangle' 'Glacier' 'Maricopa'
 'Carmel By The Sea' 'Deer Meadows' 'Silver Creek' 'Goldsboro' 'Menlo'
 'Tokeland' 'Berkeley' 'Phoenix' 'Stafford' 'Roosevelt' 'Chandler'
 'Wahkiacus' 'Lompoc' 'Alhambra' 'Buena Park' 'Snowden' 'Walla Walla
Co'
 'Outlook' 'Vader' 'Fairbanks' 'Cupertino' 'Kittitas' 'Mica'
 'Indian Wells' 'Mazama' 'Hunters' 'Camarillo' 'Belle Chasse' 'Evans'
 'Beaver' 'Grays River' 'West Valley City' 'Oceanside' 'Odessa' 'Usk'
 'Roswell' 'Gambrills' 'Oysterville' 'Potomac' 'Mineral' 'Amanda Park'
 'Toutle' 'Curtis' 'Cinebar' 'Hartline' 'Waitsburg' 'Gaithersburg'
'Husum'
 'El Paso' 'Klickitat' 'Elkton' 'Edwall' 'Sprague' 'West Palm Beach'
 'Tekoa' 'Coronado' 'Pe Ell' 'Methow' 'Danville' 'Annapolis' 'Murdock'
 'Mechanicsburg' 'Ford' 'Dickinson' 'Virginia Beach' 'Benicia'
'Moclips'
 'Suffolk' 'Holden Village' 'Bumpass' 'Lithia' 'Felts Mills' 'Little
Elm'
 'Andrews Air Force Base' 'Lyman' 'Harker Heights' 'Pittsburg'
 'Norristown' 'Mcchord Afb' 'Norfolk' 'Valparaiso' 'Conyers' 'Skokie'
 'Glenwood' 'Chesapeake' 'Fpo' 'Hughesville' 'Belleville' 'Colton'
 'South Prairie' 'Clallam Bay' 'Longmont' 'Malott' 'Okatie' 'Saint
Louis'
 'Lexington Park' 'Rockport' 'San Rafael' 'Bucoda' 'Germantown'
 'Smith Creek' 'Englewood' 'Lebam' 'South Range' 'Tempe' 'Fort Bragg'
 'Ewa Beach' 'Glenoma' 'Cheyenne' 'Portland' 'Burke' 'Hawthorne'
 'Copalis Beach' 'Satsop' 'Palisades' 'Goodyear' 'Gardena'
'Southworth'
 'Raeford' 'Rosamond' 'Clayton' 'Quinault' 'Sarasota' 'Santa Cruz'
 'Jamaica' 'Wichita Falls' 'Maryhill' 'Yermo' 'Vienna' 'Waldron'
 'Clarksville' 'Goshen' 'Herndon' 'Fruitland' 'Mccutcheon Field'
 'South Cle Elum' 'Irvine' 'Centerville' 'Fairfield' 'Lamont' 'Santa
Rosa'
 'Southern Pines' 'Copalis Crossing' 'De Queen' 'Taholah' 'Mountain
View'
 'Adairsville' 'Sacramento' 'Port Gamble' 'Apple Valley' 'Rosburg'
 'Stratford' 'Haddonfield' 'Minneapolis' 'Chelmsford' 'Old Lyme'
 'Platte City' 'Hanscom Afb' 'Fort George G Meade' 'Bangor' 'Matlock'
 'Canoga Park' 'Jericho' 'Gifford' 'Santa Ana' 'Augusta' 'San
Clemente'
 'Middletown' 'Prescott' 'San Mateo' 'Aurora' 'Bedford' 'Carrolls'
 'Fredericksburg' 'Waterford' 'Decatur' 'Mililani' 'Kekaha' 'Medford'
 'Edgewater' 'Slidell' 'Pawcatuck' 'Groton' 'Richmond Hill' 'Mabton'
 'Joint Base Mdl' 'Uniontown' 'Palo Alto' 'North Conway' 'Summerville'
 'Lansing' 'Williston' 'Worley']
Nunique values: 629
--------------------------------------------------------
********** State **********
Unique values: ['FL' 'NV' 'WA' 'IL' 'NY' 'VA' 'OK' 'KS' 'CA' 'NE' 'MD'
'CO' 'DC' 'TN'
 'SC' 'CT' 'OR' 'TX' 'SD' 'HI' 'GA' 'MS' 'AR' 'NC' 'MO' 'UT' 'PA' 'DE'
 'OH' 'WY' 'AL' 'ID' 'AZ' 'AK' 'LA' 'NM' 'WI' 'KY' 'NJ' 'MN' 'MA' 'ME'
 'RI' 'NH' 'ND']
Nunique values: 45
--------------------------------------------------------
********** Make **********
Unique values: ['TOYOTA' 'CHEVROLET' 'NISSAN' 'FORD' 'TESLA' 'KIA'
'AUDI' 'FIAT' 'BMW'
 'PORSCHE' 'CADILLAC' 'HONDA' 'MITSUBISHI' 'CHRYSLER' 'RIVIAN'
'HYUNDAI'
 'VOLVO' 'VOLKSWAGEN' 'MERCEDES-BENZ' 'JEEP' 'MINI' 'SMART' 'SUBARU'
 'POLESTAR' 'LUCID MOTORS' 'LINCOLN' 'JAGUAR' 'FISKER' 'LAND ROVER'
 'LEXUS' 'TH!NK' 'GENESIS' 'BENTLEY' 'AZURE DYNAMICS']
```

```
Nunique values: 34
--------------------------------------------------------
********** Model **********
Unique values: ['RAV4 PRIME' 'VOLT' 'LEAF' 'BOLT EV' 'FUSION' 'MODEL
3' 'SOUL' 'Q5 E'
 'MODEL X' '500' 'X5' '530E' 'TAYCAN' 'X3' 'A3' 'SOUL EV' 'C-MAX'
 'MODEL S' 'F-150' 'CT6' 'I3' 'CLARITY' 'MODEL Y' 'NIRO' 'OUTLANDER'
 'PACIFICA' 'R1T' 'KONA ELECTRIC' 'XC40' 'ID.4' 'PRIUS PLUG-IN'
 'MUSTANG MACH-E' 'EQB-CLASS' 'E-GOLF' 'PRIUS PRIME' 'C40' 'SORENTO'
 'XC60' 'CAYENNE' 'WRANGLER' 'COUNTRYMAN' 'S60' 'EV6'
 'FORTWO ELECTRIC DRIVE' 'GRAND CHEROKEE' '330E' 'CROSSTREK' 'IONIQ 5'
 'IONIQ' 'E-TRON' 'ROADSTER' 'KONA' 'XC90' 'SPARK' 'PS2' 'A7'
'HARDTOP'
 'ESCAPE' 'LUCID AIR' 'E-TRON SPORTBACK' 'Q5' 'RAV4' 'AVIATOR' 'E-TRON
GT'
 'EDV' 'IX' 'FORTWO' 'I-PACE' 'SANTA FE' 'B-CLASS' 'KARMA' 'I4'
'OPTIMA'
 'GLC-CLASS' 'Q4' 'SONATA' 'EQ FORTWO' 'FOCUS' 'RANGE ROVER SPORT'
 'TRANSIT' 'PANAMERA' 'I8' 'BOLT EUV' 'CORSAIR' 'ELR' 'GLE-CLASS'
'V60'
 'EQS-CLASS SEDAN' 'R1S' 'I-MIEV' 'NX' '740E' 'SPORTAGE' 'C-CLASS'
 'S-CLASS' 'CITY' 'S90' 'TUCSON' 'GV60' 'EQS-CLASS SUV' 'A8 E'
 'RANGE ROVER' nan 'RS E-TRON GT' 'RANGER' 'BENTAYGA' '745E'
 'TRANSIT CONNECT ELECTRIC' 'ACCORD' 'S-10 PICKUP' 'SOLTERRA' 'G80'
'918'
 'FLYING SPUR' '745LE']
Nunique values: 114
--------------------------------------------------------
********** Electric Vehicle Type **********
Unique values: ['Plug-in Hybrid Electric Vehicle (PHEV)' 'Battery
Electric Vehicle (BEV)']
Nunique values: 2
--------------------------------------------------------
********** Clean Alternative Fuel Vehicle (CAFV) Eligibility
**********
Unique values: ['Clean Alternative Fuel Vehicle Eligible'
 'Not eligible due to low battery range'
 'Eligibility unknown as battery range has not been researched']
Nunique values: 3
--------------------------------------------------------
********** Vehicle Location **********
Unique values: ['POINT (-81.80023 24.5545)' 'POINT (-114.57245
35.16815)'
 'POINT (-120.50721 46.60448)' 'POINT (-121.7515 48.53892)'
 'POINT (-122.20596 47.97659)' 'POINT (-122.18384 47.8031)'
 'POINT (-122.23019 47.94949)' 'POINT (-122.29196 47.89908)'
 'POINT (-122.35803 47.9796)' 'POINT (-122.61214 48.51748)'
 'POINT (-122.75379 47.06316)' 'POINT (-119.2771 47.13196)'
 'POINT (-89.79939 38.49028)' 'POINT (-123.08743 46.82175)'
 'POINT (-122.33029 48.46846)' nan 'POINT (-122.19388 48.15353)'
 'POINT (-122.27734 47.83785)' 'POINT (-73.84643 43.1284)'
 'POINT (-122.31768 47.87166)' 'POINT (-117.54392 47.77676)'
 'POINT (-122.92333 47.03779)' 'POINT (-122.27981 47.85727)'
 'POINT (-122.3026 47.72656)' 'POINT (-122.23035 47.3074)'
 'POINT (-122.31765 47.70013)' 'POINT (-122.37689 47.81116)'
 'POINT (-122.40618 48.0399)' 'POINT (-122.82324 47.04437)'
 'POINT (-122.89166 47.03956)' 'POINT (-122.18637 47.89252)'
 'POINT (-122.1389 47.87115)' 'POINT (-122.66122 47.56573)'
 'POINT (-122.29245 47.82557)' 'POINT (-122.25527 47.90456)'
 'POINT (-76.53585 37.10499)' 'POINT (-120.52301 46.60138)'
 'POINT (-120.56916 46.58514)' 'POINT (-99.33374 34.63783)'
 'POINT (-117.18147 46.73015)' 'POINT (-87.7992 42.18569)'
 'POINT (-117.45674 47.69963)' 'POINT (-122.55242 47.73162)'
 'POINT (-121.98087 47.8526)' 'POINT (-123.10367 48.07965)'
 'POINT (-122.62316 47.70251)' 'POINT (-122.80277 46.99409)'
 'POINT (-87.91441 42.36495)' 'POINT (-122.04526 47.39394)'
 'POINT (-122.1769 48.06114)' 'POINT (-122.17743 47.41185)'
 'POINT (-122.31327 47.75736)' 'POINT (-122.64681 47.73689)'
 'POINT (-122.09305 47.91265)' 'POINT (-122.13158 47.67858)'
 'POINT (-122.00292 47.54748)' 'POINT (-122.95058 46.14681)'
 'POINT (-122.52054 47.26887)' 'POINT (-120.54872 46.99703)'
 'POINT (-122.3317 47.50314)' 'POINT (-122.72457 47.38165)'
 'POINT (-122.35051 47.15252)' 'POINT (-122.02054 47.60326)'
 'POINT (-122.32427 47.63433)' 'POINT (-124.10424 46.89078)'
 'POINT (-122.49212 45.60365)' 'POINT (-122.20563 47.76144)'
 'POINT (-117.59311 47.6443)' 'POINT (-122.03539 47.61344)'
 'POINT (-122.21238 47.57816)' 'POINT (-122.37265 48.24159)'
 'POINT (-122.03287 47.68555)' 'POINT (-122.521 47.62728)'
 'POINT (-122.23741 47.3807)' 'POINT (-122.30716 47.62687)'
 'POINT (-122.31009 47.60803)' 'POINT (-122.41067 47.57894)'
 'POINT (-122.21698 47.28317)' 'POINT (-120.20831 47.67562)'
 'POINT (-122.54795 47.17997)' 'POINT (-122.17144 47.19175)'
 'POINT (-122.35186 47.54286)' 'POINT (-122.44164 47.25517)'
 'POINT (-122.11867 47.63131)' 'POINT (-122.64695 45.65675)'
 'POINT (-117.3186 47.6505)' 'POINT (-122.2066 47.67887)'
 'POINT (-122.30866 47.57874)' 'POINT (-122.37571 47.16104)'
 'POINT (-122.38591 47.67597)' 'POINT (-122.15771 47.50549)'
 'POINT (-122.65745 47.4916)' 'POINT (-122.47122 48.75235)'
 'POINT (-122.62934 45.63201)' 'POINT (-119.26844 46.31484)'
 'POINT (-122.1872 47.61001)' 'POINT (-122.40049 48.23986)'
 'POINT (-120.30522 47.41494)' 'POINT (-122.35436 47.67596)'
 'POINT (-122.23825 47.49461)' 'POINT (-122.06402 48.01497)'
 'POINT (-122.5476 45.62832)' 'POINT (-122.62731 45.71668)'
 'POINT (-122.45516 48.74487)' 'POINT (-122.54332 47.00328)'
 'POINT (-122.60302 47.36223)' 'POINT (-122.34223 47.61085)'
 'POINT (-122.29592 47.40139)' 'POINT (-122.36498 47.72238)'
 'POINT (-122.40199 45.58694)' 'POINT (-119.3118 46.20664)'
 'POINT (-122.38415 47.53755)' 'POINT (-122.63847 47.54103)'
 'POINT (-122.31307 47.66127)' 'POINT (-122.08747 47.4466)'
 'POINT (-122.21061 47.83448)' 'POINT (-122.34118 47.46665)'
 'POINT (-119.27372 46.27391)' 'POINT (-122.2668 47.55115)'
 'POINT (-122.5331 45.78092)' 'POINT (-122.70303 45.70954)'
 'POINT (-122.34468 47.61578)' 'POINT (-118.34261 46.07068)'
 'POINT (-121.7831 47.49348)' 'POINT (-122.12096 47.55584)'
 'POINT (-122.22901 47.72201)' 'POINT (-122.18463 47.49929)'
 'POINT (-122.87741 47.05997)' 'POINT (-122.19564 47.37271)'
 'POINT (-122.32267 48.41626)' 'POINT (-122.58009 47.328)'
 'POINT (-122.1621 47.64441)' 'POINT (-122.74017 45.9094)'
 'POINT (-122.15545 47.75448)' 'POINT (-122.82983 47.38174)'
 'POINT (-122.3684 47.64586)' 'POINT (-121.82432 47.52716)'
 'POINT (-121.89086 47.56812)' 'POINT (-122.28449 47.146)'
 'POINT (-123.01648 48.53448)' 'POINT (-122.38418 47.70044)'
 'POINT (-123.05688 48.98867)' 'POINT (-122.62954 47.09583)'
 'POINT (-122.346385 47.630685)' 'POINT (-122.07479 47.75264)'
 'POINT (-122.90724 46.27509)' 'POINT (-122.74888 48.99404)'
 'POINT (-122.55149 45.69345)' 'POINT (-119.11698 46.20804)'
 'POINT (-122.03439 47.5301)' 'POINT (-117.45005 47.7333)'
 'POINT (-122.30346 47.55379)' 'POINT (-117.3797 47.65548)'
 'POINT (-122.27511 46.55465)' 'POINT (-122.77263 48.1212)'
 'POINT (-120.99607 47.22593)' 'POINT (-122.24369 47.75892)'
 'POINT (-122.09124 47.33778)' 'POINT (-122.3303 47.30151)'
 'POINT (-122.69275 47.65171)' 'POINT (-122.3503 47.71868)'
 'POINT (-122.40092 47.65908)' 'POINT (-122.23892 47.61613)'
 'POINT (-122.21152 47.43954)' 'POINT (-121.99136 47.20433)'
 'POINT (-122.33891 48.41644)' 'POINT (-120.22795 47.62352)'
 'POINT (-119.90123 46.25606)' 'POINT (-120.26186 46.40186)'
 'POINT (-122.28556 47.18709)' 'POINT (-122.4573 47.44929)'
 'POINT (-122.52886 47.24977)' 'POINT (-121.48704 45.72776)'
 'POINT (-122.57722 45.64251)' 'POINT (-122.32945 47.60357)'
 'POINT (-122.5338 47.23594)' 'POINT (-121.97745 47.53433)'
 'POINT (-120.29473 47.41515)' 'POINT (-120.60364 47.56805)'
 'POINT (-122.82364 47.32767)' 'POINT (-122.59802 47.17303)'
 'POINT (-117.46996 47.59431)' 'POINT (-122.23972 47.2022)'
 'POINT (-96.83046 39.02826)' 'POINT (-119.141 46.19162)'
 'POINT (-117.1748 47.65699)' 'POINT (-123.10565 47.21248)'
 'POINT (-122.41249 47.21584)' 'POINT (-122.96462 46.6621)'
 'POINT (-117.36043 47.63396)' 'POINT (-122.46495 47.16778)'
 'POINT (-124.20177 47.20886)' 'POINT (-122.34536 48.92009)'
 'POINT (-122.36646 47.19269)' 'POINT (-122.00143 47.30893)'
 'POINT (-122.51766 47.27779)' 'POINT (-121.97 45.64119)'
 'POINT (-122.64443 45.67871)' 'POINT (-122.12053 47.61334)'
 'POINT (-122.68558 48.21857)' 'POINT (-122.81585 47.64509)'
 'POINT (-122.1264 48.19471)' 'POINT (-122.92057 47.0031)'
 'POINT (-77.05745 38.74002)' 'POINT (-117.0768 46.9101)'
 'POINT (-122.397 48.56045)' 'POINT (-122.35206 47.30297)'
 'POINT (-122.17663 47.32326)' 'POINT (-122.61624 47.57772)'
 'POINT (-122.75493 47.25035)' 'POINT (-122.53691 47.20606)'
 'POINT (-122.6462 47.63132)' 'POINT (-122.49756 48.7999)'
 'POINT (-120.93943 47.195)' 'POINT (-122.40908 45.86679)'
 'POINT (-122.64682 48.29077)' 'POINT (-122.41894 47.15806)'
 'POINT (-120.82548 45.823)' 'POINT (-122.75878 48.03591)'
 'POINT (-122.20348 48.71768)' 'POINT (-118.37977 47.1274)'
 'POINT (-123.09944 47.35689)' 'POINT (-122.20166 47.09596)'
 'POINT (-123.04061 47.37733)' 'POINT (-122.65107 47.27375)'
 'POINT (-122.41666 47.30682)' 'POINT (-120.39197 46.55621)'
 'POINT (-123.46296 48.11653)' 'POINT (-122.36178 47.49408)'
 'POINT (-122.41046 47.06512)' 'POINT (-122.59219 45.62158)'
 'POINT (-117.4077 47.74527)' 'POINT (-122.91395 48.52342)'
 'POINT (-120.65754 47.5982)' 'POINT (-122.30164 47.45775)'
 'POINT (-121.88258 45.69417)' 'POINT (-119.09467 46.23542)'
 'POINT (-122.44718 47.20144)' 'POINT (-122.60735 46.94239)'
 'POINT (-119.43683 48.70854)' 'POINT (-117.1015 47.66829)'
 'POINT (-122.54758 47.91795)' 'POINT (-122.91109 48.69389)'
 'POINT (-117.42571 47.64779)' 'POINT (-122.69107 48.05105)'
 'POINT (-118.66919 46.04238)' 'POINT (-117.2335 47.67666)'
 'POINT (-120.53113 46.65404)' 'POINT (-117.37056 47.70402)'
 'POINT (-122.35686 47.23679)' 'POINT (-122.45079 48.9429)'
 'POINT (-123.81619 46.97606)' 'POINT (-122.67876 47.17794)'
 'POINT (-123.00026 48.61989)' 'POINT (-122.49771 47.79803)'
 'POINT (-121.95585 46.53534)' 'POINT (-122.48271 47.24737)'
 'POINT (-117.41162 47.65726)' 'POINT (-122.23857 48.50858)'
 'POINT (-121.91353 47.64901)' 'POINT (-122.43874 47.22448)'
 'POINT (-122.29537 47.19044)' 'POINT (-122.82764 47.45054)'
 'POINT (-117.57445 47.48928)' 'POINT (-123.40382 47.00379)'
 'POINT (-122.45691 47.26496)' 'POINT (-122.54729 47.42602)'
 'POINT (-121.96931 48.08157)' 'POINT (-119.55125 47.31867)'
 'POINT (-122.74595 45.81539)' 'POINT (-123.26405 47.04946)'
 'POINT (-122.59351 48.84756)' 'POINT (-122.30116 47.1165)'
 'POINT (-122.32806 47.46155)' 'POINT (-122.28879 47.44538)'
 'POINT (-122.31111 47.78803)' 'POINT (-122.47554 47.21835)'
 'POINT (-122.67156 45.63248)' 'POINT (-123.4313 48.11872)'
 'POINT (-122.52664 48.00956)' 'POINT (-122.45728 38.29188)'
 'POINT (-122.68993 46.88897)' 'POINT (-120.00949 46.32379)'
 'POINT (-122.6306 46.53186)' 'POINT (-117.90454 48.54657)'
 'POINT (-121.98609 47.74068)' 'POINT (-117.3973 47.67573)'
 'POINT (-117.11195 47.70763)' 'POINT (-120.1231 48.3652)'
 'POINT (-122.26832 46.86733)' 'POINT (-117.35237 47.89594)'
 'POINT (-124.16408 47.01156)' 'POINT (-122.35341 45.57923)'
 'POINT (-122.68475 47.92989)' 'POINT (-119.48756 46.26543)'
 'POINT (-117.04556 46.41402)' 'POINT (-121.98233 47.35298)'
 'POINT (-122.89868 46.14425)' 'POINT (-118.60061 48.88101)'
 'POINT (-117.39322 47.63374)' 'POINT (-122.21766 48.8268)'
 'POINT (-119.76877 46.20645)' 'POINT (-122.32172 47.24898)'
 'POINT (-123.14135 47.40639)' 'POINT (-119.33788 46.2969)'
 'POINT (114.32466 4.17756)' 'POINT (-102.69968 22.95716)'
 'POINT (-120.01454 47.83985)' 'POINT (-122.29477 47.05703)'
 'POINT (-123.72994 46.68867)' 'POINT (-122.54578 45.73473)'
 'POINT (-120.14545 47.37163)' 'POINT (-122.51495 47.16195)'
 'POINT (-122.49724 48.38874)' 'POINT (-117.5821 47.09185)'
 'POINT (-117.35586 47.7628)' 'POINT (-123.88689 46.97982)'
 'POINT (-117.47154 47.95431)' 'POINT (-119.03591 47.93348)'
 'POINT (-122.77295 48.01182)' 'POINT (-119.01561 46.1992)'
 'POINT (-119.85338 47.23748)' 'POINT (-95.94513 41.30481)'
 'POINT (-122.67244 45.86161)' 'POINT (-121.02627 47.23781)'
 'POINT (-124.05439 46.35232)' 'POINT (-117.72555 48.17623)'
 'POINT (-122.84226 46.00813)' 'POINT (-122.62749 47.565)'
 'POINT (-122.51843 47.7487)' 'POINT (-120.17739 48.47396)'
 'POINT (-117.26237 33.61236)' 'POINT (-117.72682 33.56964)'
 'POINT (-122.02523 47.16299)' 'POINT (-123.60076 46.98384)'
 'POINT (-115.29715 36.11477)' 'POINT (-117.2718 47.65388)'
 'POINT (-122.31902 48.01306)' 'POINT (-117.97577 46.32189)'
 'POINT (-122.76355 47.34347)' 'POINT (-122.69891 46.57613)'
 'POINT (-117.68269 47.57261)' 'POINT (-122.32181 48.92857)'
 'POINT (-122.95298 46.72894)' 'POINT (-121.81688 47.8623)'
 'POINT (-121.52529 45.99588)' 'POINT (-124.05475 46.3354)'
 'POINT (-121.82048 45.72558)' 'POINT (-117.35414 47.83743)'
 'POINT (-122.68321 48.72043)' 'POINT (-117.06882 47.7251)'
 'POINT (-123.38568 46.20728)' 'POINT (-117.19631 47.65706)'
 'POINT (-117.23995 47.5353)' 'POINT (-120.47005 47.5224)'
 'POINT (-122.53658 45.98974)' 'POINT (-123.76901 46.95413)'
 'POINT (-124.03547 46.3088)' 'POINT (-123.23286 46.84077)'
 'POINT (-122.70639 47.68732)' 'POINT (-122.8112 46.29694)'
 'POINT (-122.93881 46.49114)' 'POINT (-122.56821 48.08844)'
 'POINT (-122.85135 46.85752)' 'POINT (-119.63136 46.90248)'
 'POINT (-122.32738 47.58616)' 'POINT (-94.93427 39.34433)'
 'POINT (-122.64219 48.91865)' 'POINT (-122.12775 37.85356)'
 'POINT (-122.57908 47.10349)' 'POINT (-118.38864 46.03456)'
 'POINT (-76.99179 39.2416)' 'POINT (-121.52357 45.72902)'
 'POINT (-122.42148 45.91958)' 'POINT (-121.46623 45.71513)'
 'POINT (-123.04373 46.37331)' 'POINT (-122.23452 48.46376)'
 'POINT (-120.69972 46.7309)' 'POINT (-121.93654 47.54371)'
 'POINT (-124.0495 46.49147)' 'POINT (-122.8357 48.62356)'
 'POINT (-119.1742 46.82616)' 'POINT (-117.37047 47.23428)'
 'POINT (-121.41201 47.41873)' 'POINT (-104.98231 40.52978)'
 'POINT (-118.73748 48.64426)' 'POINT (-77.04405 38.92033)'
 'POINT (-77.69839 39.48824)' 'POINT (-89.77858 35.44156)'
 'POINT (-117.07993 32.8347)' 'POINT (-80.34689 33.95059)'
 'POINT (-76.84002 38.90272)' 'POINT (-72.59507 41.27973)'
 'POINT (-124.01768 44.93009)' 'POINT (-119.00041 47.94283)'
 'POINT (-104.50818 37.16772)' 'POINT (-117.71555 48.28136)'
 'POINT (-121.66939 46.61019)' 'POINT (-120.66621 47.05139)'
 'POINT (-120.47794 46.55282)' 'POINT (-77.38938 39.40161)'
 'POINT (-120.20227 47.37238)' 'POINT (-118.61566 46.97135)'
 'POINT (-120.18721 46.33937)' 'POINT (-118.70912 47.75942)'
 'POINT (-122.84784 46.44067)' 'POINT (-120.31298 46.37508)'
 'POINT (-119.01647 46.45698)' 'POINT (-124.30041 48.25978)'
 'POINT (-117.13359 47.00706)' 'POINT (-118.41899 47.82937)'
 'POINT (-117.04398 48.18285)' 'POINT (-118.25682 47.4829)'
 'POINT (-122.24692 47.26465)' 'POINT (-122.33145 47.60622)'
 'POINT (-98.52212 29.61445)' 'POINT (-122.7478 46.53239)'
 'POINT (-117.58937 46.47398)' 'POINT (-122.75905 47.21454)'
 'POINT (-118.86063 46.66146)' 'POINT (-122.89646 47.69838)'
 'POINT (-121.35912 47.71192)' 'POINT (-117.87704 47.66951)'
 'POINT (-122.07728 48.92369)' 'POINT (-103.2308 44.08146)'
 'POINT (-119.28816 47.61233)' 'POINT (-121.14532 45.61623)'
 'POINT (-119.9899 46.94468)' 'POINT (-119.43558 48.93881)'
 'POINT (-120.16034 47.8854)' 'POINT (-157.94001 21.34183)'
 'POINT (-119.52724 48.41078)' 'POINT (-119.78093 48.09967)'
 'POINT (-119.00503 46.57833)' 'POINT (-76.69505 39.0814)'
 'POINT (-120.07145 47.64963)' 'POINT (-96.16579 41.20002)'
 'POINT (-123.94534 46.27264)' 'POINT (-121.69743 47.85565)'
 'POINT (-119.48323 47.38898)' 'POINT (-120.75541 46.70257)'
 'POINT (-77.07354 39.09303)' 'POINT (-83.66512 32.60833)'
 'POINT (-119.9 46.73777)' 'POINT (-117.8379 48.35609)'
 'POINT (-117.63731 48.06288)' 'POINT (-80.066 32.89244)'
 'POINT (-124.38543 47.95014)' 'POINT (-120.42051 46.44779)'
 'POINT (-123.81154 46.36545)' 'POINT (-122.8775 47.82642)'
 'POINT (-117.04784 46.34056)' 'POINT (-121.89647 36.5982)'
 'POINT (-121.17954 47.2378)' 'POINT (-117.64458 47.64046)'
 'POINT (-122.50156 47.13872)' 'POINT (-117.66798 35.62248)'
 'POINT (-123.45797 46.2716)' 'POINT (-123.0921 47.47203)'
 'POINT (-117.10893 32.71148)' 'POINT (-122.28541 38.29911)'
 'POINT (-117.17188 32.81936)' 'POINT (-77.6513 38.97842)'
 'POINT (-118.98504 47.40718)' 'POINT (-119.04573 46.96971)'
 'POINT (-88.9941 30.39451)' 'POINT (-118.14913 47.65405)'
 'POINT (-117.17592 32.73722)' 'POINT (-116.97437 32.61628)'
 'POINT (-92.32538 34.75693)' 'POINT (-121.28544 45.6939)'
 'POINT (-78.90159 35.05792)' 'POINT (-118.11725 33.861)'
 'POINT (-118.06211 48.61053)' 'POINT (-123.81481 46.66382)'
 'POINT (-94.06331 39.34793)' 'POINT (-122.40527 47.2556)'
 'POINT (-119.57959 48.3654)' 'POINT (-119.63759 47.8118)'
 'POINT (-119.89975 48.05502)' 'POINT (-122.26496 48.99956)'
 'POINT (-117.24736 32.74626)' 'POINT (-111.89452 40.73992)'
 'POINT (-115.20278 36.29143)' 'POINT (-115.21837 36.09378)'
 'POINT (-117.30443 48.33866)' 'POINT (-121.99263 38.35218)'
 'POINT (-87.43844 36.64424)' 'POINT (-122.03243 46.75908)'
 'POINT (-117.27816 48.01723)' 'POINT (-122.04735 47.08015)'
 'POINT (-76.988642 38.904734)' 'POINT (-117.13261 47.45227)'
 'POINT (-117.15511 47.27966)' 'POINT (-76.99225 39.57467)'
 'POINT (-120.0051 48.12868)' 'POINT (-121.60421 48.25431)'
 'POINT (-122.64782 45.36625)' 'POINT (-122.48446 46.52948)'
 'POINT (-75.50358 39.10597)' 'POINT (-111.94855 40.6527)'
 'POINT (-117.67024 47.8927)' 'POINT (-76.50718 39.0329)'
 'POINT (-81.73215 28.80419)' 'POINT (-95.57424 29.74768)'
 'POINT (-122.53087 47.12729)' 'POINT (-119.29232 34.28082)'
 'POINT (-119.50853 48.5012)' 'POINT (-124.09366 46.80933)'
 'POINT (-117.74137 48.05738)' 'POINT (-122.30857 46.04647)'
 'POINT (-117.68485 46.92752)' 'POINT (-118.19657 48.29812)'
 'POINT (-120.10803 48.25448)' 'POINT (-84.36724 39.19105)'
 'POINT (-123.05598 44.94796)' 'POINT (-121.33747 38.80276)'
 'POINT (-76.53375 38.69549)' 'POINT (-115.20881 36.15352)'
 'POINT (-117.36477 46.87965)' 'POINT (-115.05535 36.24407)'
 'POINT (-120.73063 46.38287)' 'POINT (-118.16165 48.42934)'
 'POINT (-106.95587 44.79792)' 'POINT (-124.62735 48.36778)'
 'POINT (-119.78637 36.30101)' 'POINT (-86.27374 32.3611)'
 'POINT (-104.70202 38.91191)' 'POINT (-76.3379 36.83436)'
 'POINT (-118.24538 34.65105)' 'POINT (-80.37 33.89648)'
 'POINT (-118.97669 47.9708)' 'POINT (-117.15392 33.685)'
 'POINT (-122.92991 48.58348)' 'POINT (-115.85601 43.05683)'
 'POINT (-121.41604 48.52739)' 'POINT (-121.95011 37.39684)'
 'POINT (-98.65734 29.46536)' 'POINT (-121.48491 47.77193)'
 'POINT (-117.38035 47.42909)' 'POINT (-117.16171 32.71568)'
 'POINT (-95.94026 41.26068)' 'POINT (-112.02436 33.05319)'
 'POINT (-121.92442 36.55443)' 'POINT (-104.83531 38.9291)'
 'POINT (-122.60209 46.52793)' 'POINT (-77.95024 35.37084)'
 'POINT (-123.99814 46.71227)' 'POINT (-122.2962 37.85103)'
 'POINT (-112.14075 33.86739)' 'POINT (-77.40815 38.42366)'
 'POINT (-104.63826 38.74006)' 'POINT (-111.85047 33.25707)'
 'POINT (-121.10717 45.84657)' 'POINT (-120.45788 34.64048)'
 'POINT (-118.12778 34.09442)' 'POINT (-98.67331 29.49379)'
 'POINT (-118.01268 33.83899)' 'POINT (-117.17638 33.71394)'
 'POINT (-120.08512 46.3294)' 'POINT (-122.95015 46.40201)'
 'POINT (-147.72213 64.84527)' 'POINT (-122.02929 37.31913)'
 'POINT (-120.41752 46.98472)' 'POINT (-117.21611 47.56436)'
 'POINT (-116.33881 33.72413)' 'POINT (-120.4072 48.59369)'
 'POINT (-118.20299 48.11777)' 'POINT (-119.05072 34.2234)'
 'POINT (-89.9878 29.85069)' 'POINT (-90.57607 41.51196)'
 'POINT (-118.02345 48.71331)' 'POINT (-124.33152 48.05431)'
 'POINT (-77.41203 39.41574)' 'POINT (-123.61022 46.35588)'
 'POINT (-112.04165 40.68741)' 'POINT (-118.68999 47.33323)'
 'POINT (-117.28054 48.31375)' 'POINT (-117.06451 32.90323)'
 'POINT (-104.52275 33.39509)' 'POINT (-76.6969 39.0606)'
 'POINT (-77.20884 39.01796)' 'POINT (-122.64751 47.55671)'
 'POINT (-122.18768 46.71902)' 'POINT (-123.89691 47.45917)'
 'POINT (-122.73237 46.32447)' 'POINT (-123.13056 46.56094)'
 'POINT (-122.49165 46.60409)' 'POINT (-119.10557 47.69015)'
 'POINT (-118.15448 46.27013)' 'POINT (-77.22708 39.14162)'
 'POINT (-106.55656 31.88532)' 'POINT (-123.56707 43.63677)'
 'POINT (-117.95109 47.50524)' 'POINT (-117.97378 47.30036)'
 'POINT (-80.13342 26.64804)' 'POINT (-117.07351 47.22679)'
 'POINT (-117.17089 32.67619)' 'POINT (-123.29831 46.57001)'
 'POINT (-79.4172 36.58598)' 'POINT (-76.49014 38.97678)'
 'POINT (-117.05805 32.57923)' 'POINT (-76.96675 40.23089)'
 'POINT (-117.80582 47.90886)' 'POINT (-95.05026 29.4608)'
 'POINT (-76.20908 36.80823)' 'POINT (-122.15351 38.05285)'
 'POINT (-124.21349 47.23932)' 'POINT (-76.42443 36.8752)'
 'POINT (-77.73727 37.96459)' 'POINT (-82.17029 27.86266)'
 'POINT (-75.75904 44.0205)' 'POINT (-96.9387 33.15985)'
 'POINT (-76.8907 38.81605)' 'POINT (-122.05976 48.52525)'
 'POINT (-97.66091 31.08871)' 'POINT (-121.88851 38.01946)'
 'POINT (-115.32375 36.29409)' 'POINT (-75.40705 40.15076)'
 'POINT (-76.99953 38.88594)' 'POINT (-76.23458 36.93595)'
 'POINT (-86.49296 30.51166)' 'POINT (-83.97862 33.61951)'
 'POINT (-87.75545 42.02634)' 'POINT (-121.29272 46.01842)'
 'POINT (-96.18046 41.23687)' 'POINT (-76.21549 36.92478)'
 'POINT (-76.40587 36.84526)' 'POINT (-117.99804 33.8652)'
 'POINT (-77.04341 38.8046)' 'POINT (-76.78372 38.53236)'
 'POINT (-77.10727 38.81797)' 'POINT (-89.98423 38.5124)'
 'POINT (-77.05604 38.82399)' 'POINT (-117.12984 46.56801)'
 'POINT (-122.09624 47.13954)' 'POINT (-124.26015 48.25371)'
 'POINT (-105.10015 40.16394)' 'POINT (-111.83173 40.6253)'
 'POINT (-80.95653 32.30585)' 'POINT (-90.21137 38.7483)'
 'POINT (-76.13652 36.84315)' 'POINT (-76.45275 38.26564)'
 'POINT (-121.59274 48.48758)' 'POINT (-122.53288 37.97461)'
 'POINT (27.25316 67.01865)' 'POINT (-77.23717 39.17915)'
 'POINT (-104.89239 39.61914)' 'POINT (-91.98379 46.60774)'
 'POINT (-111.93722 33.42554)' 'POINT (-76.29676 36.86523)'
 'POINT (-78.9973 35.13558)' 'POINT (-158.00833 21.31431)'
 'POINT (-104.82154 41.13481)' 'POINT (-122.64942 45.50606)'
 'POINT (-77.27191 38.79355)' 'POINT (-118.35273 33.91822)'
 'POINT (-124.16705 47.11487)' 'POINT (-123.48162 47.00236)'
 'POINT (-119.9176 47.41492)' 'POINT (-117.25255 32.77752)'
 'POINT (-112.44862 33.49778)' 'POINT (-118.32472 33.89612)'
 'POINT (-118.1924 33.76672)' 'POINT (-117.15432 32.79315)'
 'POINT (-79.22405 34.9786)' 'POINT (-118.50797 48.99237)'
 'POINT (-118.16344 34.86438)' 'POINT (-117.57228 47.99685)'
 'POINT (-77.05942 38.7748)' 'POINT (-123.84618 47.46807)'
 'POINT (-82.559 27.27335)' 'POINT (-120.20117 45.74956)'
 'POINT (-122.01027 36.97542)' 'POINT (-117.23769 32.7263)'
 'POINT (-73.7959 40.67448)' 'POINT (-98.57603 33.86725)'
 'POINT (-116.83609 34.90718)' 'POINT (-77.26399 38.90063)'
 'POINT (-123.03598 48.68843)' 'POINT (-87.37923 36.54824)'
 'POINT (-85.57785 38.40025)' 'POINT (-77.4108 38.94955)'
 'POINT (-118.19894 48.0784)' 'POINT (-77.47087 34.72943)'
 'POINT (-82.13029 33.53342)' 'POINT (-120.94822 47.18891)'
 'POINT (-77.13581 38.86995)' 'POINT (-92.33405 34.77306)'
 'POINT (-117.77144 33.73287)' 'POINT (-120.903 45.75198)'
 'POINT (-95.95882 41.13181)' 'POINT (-117.17125 47.38522)'
 'POINT (-117.90629 47.20139)' 'POINT (-117.37445 33.21097)'
 'POINT (-122.72292 38.45663)' 'POINT (-124.02697 46.54905)'
 'POINT (-79.39147 35.17441)' 'POINT (-124.07463 47.11059)'
 'POINT (-94.33945 34.03674)' 'POINT (-124.28628 47.34257)'
 'POINT (-122.07056 37.36769)' 'POINT (-84.92804 34.37047)'
 'POINT (-121.46547 38.55173)' 'POINT (-117.20243 34.48186)'
 'POINT (-123.6386 46.33397)' 'POINT (-119.27844 47.4263)'
 'POINT (-75.03015 39.89896)' 'POINT (-77.015 38.87754)'
 'POINT (-122.27026 37.90054)' 'POINT (-115.34614 36.17207)'
 'POINT (-117.76724 33.70884)' 'POINT (-93.28272 44.94292)'
 'POINT (-117.71277 33.67538)' 'POINT (-71.34587 42.59853)'
 'POINT (-72.33425 41.32392)' 'POINT (-94.77998 39.37027)'
 'POINT (-71.28309 42.45982)' 'POINT (-123.95518 46.63188)'
 'POINT (-76.73517 39.10852)' 'POINT (-68.77077 44.80171)'
 'POINT (-123.40727 47.23519)' 'POINT (-118.59524 34.2271)'
 'POINT (-73.53813 40.791)' 'POINT (-118.13633 48.31085)'
 'POINT (-117.29317 33.21394)' 'POINT (-117.8943 33.75341)'
 'POINT (-82.1221 33.39842)' 'POINT (-117.60698 33.43024)'
 'POINT (-71.29923 41.51624)' 'POINT (-117.30761 33.25426)'
 'POINT (-118.31243 46.29961)' 'POINT (-122.31485 37.54704)'
 'POINT (-104.758 39.80523)' 'POINT (-71.28305 42.49301)'
 'POINT (-122.86239 46.07142)' 'POINT (-77.44058 38.33761)'
 'POINT (-72.13852 41.34593)' 'POINT (-84.29658 33.77437)'
 'POINT (-158.00759 21.45509)' 'POINT (-159.71288 21.96605)'
 'POINT (-77.0745 38.92255)' 'POINT (-74.82236 39.88938)'
 'POINT (-76.56552 38.92607)' 'POINT (-89.72188 30.29146)'
 'POINT (-78.84846 35.0396)' 'POINT (-71.84172 41.37642)'
 'POINT (-122.13017 46.53072)' 'POINT (-72.07145 41.35486)'
 'POINT (-81.31202 31.94967)' 'POINT (-119.99854 46.21092)'
 'POINT (-74.58482 40.05056)' 'POINT (-117.08742 46.53906)'
 'POINT (-122.12576 37.44719)' 'POINT (-71.12513 44.04945)'
 'POINT (-80.17601 33.01897)' 'POINT (-94.89874 39.23762)'
 'POINT (7.86484 51.32975)' 'POINT (-116.91895 47.40077)']
Nunique values: 758
--------------------------------------------------------
********** Electric Utility **********
Unique values: [nan 'PACIFICORP' 'PUGET SOUND ENERGY INC' 'PUD NO 2 OF
GRANT COUNTY'
 'PUGET SOUND ENERGY INC||CITY OF TACOMA - (WA)'
 'CITY OF SEATTLE - (WA)|CITY OF TACOMA - (WA)' 'AVISTA CORP'
 'MODERN ELECTRIC WATER COMPANY'
 'BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF COWLITZ COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||CITY OF TACOMA - (WA)||PENINSULA
LIGHT COMPANY'
 'BONNEVILLE POWER ADMINISTRATION||CITY OF ELLENSBURG - (WA)'
 'BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF GRAYS HARBOR COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF CLARK COUNTY - (WA)'
 'BONNEVILLE POWER ADMINISTRATION||AVISTA CORP||INLAND POWER & LIGHT
COMPANY'
 'BONNEVILLE POWER ADMINISTRATION||PUD 1 OF SNOHOMISH COUNTY'
 'PUD NO 1 OF CHELAN COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||VERA IRRIGATION DISTRICT #15'
 'BONNEVILLE POWER ADMINISTRATION||CITY OF TACOMA - (WA)||ELMHURST
MUTUAL POWER & LIGHT CO|PENINSULA LIGHT COMPANY'
```

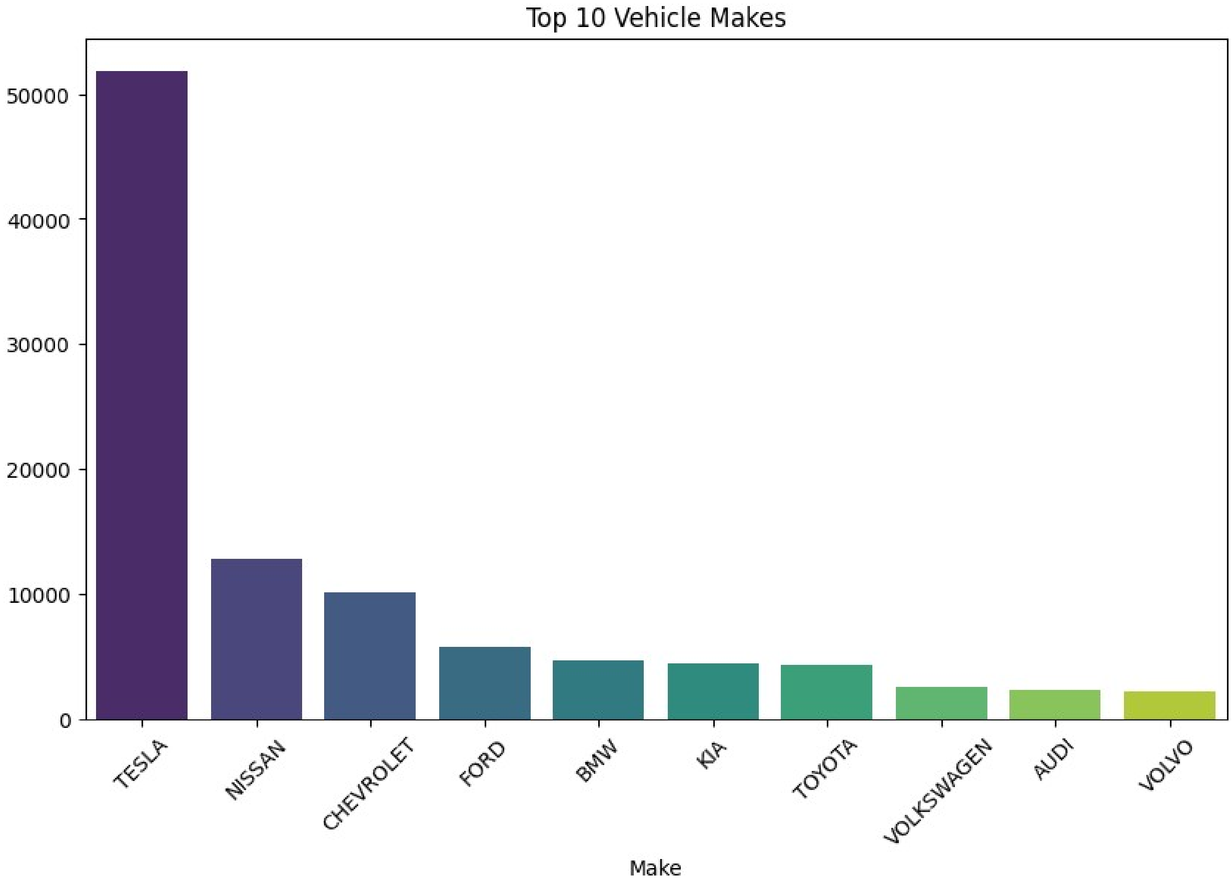
```
 'PUGET SOUND ENERGY INC||PUD NO 1 OF WHATCOM COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||CITY OF RICHLAND - (WA)'
 'BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF BENTON COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||CITY OF TACOMA - (WA)||PUD NO 3 OF
MASON COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||ORCAS POWER & LIGHT COOP'
 'BONNEVILLE POWER ADMINISTRATION||CITY OF TACOMA - (WA)||PUD NO 1 OF
LEWIS COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||PUGET SOUND ENERGY INC||PUD NO 1 OF
JEFFERSON COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF CLALLAM COUNTY'
 'PUD NO 1 OF DOUGLAS COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF KLICKITAT COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||TOWN OF STEILACOOM|CITY OF TACOMA -
(WA)||PENINSULA LIGHT COMPANY'
 'BONNEVILLE POWER ADMINISTRATION||INLAND POWER & LIGHT COMPANY'
 'BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF SKAMANIA CO'
 'BONNEVILLE POWER ADMINISTRATION||CITY OF TACOMA - (WA)||LAKEVIEW
LIGHT & POWER|PENINSULA LIGHT COMPANY'
 'BONNEVILLE POWER ADMINISTRATION||AVISTA CORP||BIG BEND ELECTRIC
COOP, INC'
 'BONNEVILLE POWER ADMINISTRATION||CITY OF TACOMA - (WA)||PUD NO 1 OF
MASON COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF FRANKLIN COUNTY'
 'PUD NO 1 OF OKANOGAN COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||CITY OF MCCLEARY - (WA)'
 'BONNEVILLE POWER ADMINISTRATION||PACIFICORP||BENTON RURAL ELECTRIC
ASSN'
 'BONNEVILLE POWER ADMINISTRATION||CITY OF PORT ANGELES - (WA)'
 'OKANOGAN COUNTY ELEC COOP, INC'
 'BONNEVILLE POWER ADMINISTRATION||CITY OF TACOMA - (WA)||OHOP MUTUAL
LIGHT COMPANY, INC|PENINSULA LIGHT COMPANY'
 'BONNEVILLE POWER ADMINISTRATION||AVISTA CORP||PUD NO 1 OF ASOTIN
COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF FERRY COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||CITY OF MILTON - (WA)|CITY OF
TACOMA - (WA)'
 'BONNEVILLE POWER ADMINISTRATION||CITY OF TACOMA - (WA)||PARKLAND
LIGHT & WATER COMPANY|PENINSULA LIGHT COMPANY'
 'BONNEVILLE POWER ADMINISTRATION||PUD NO 2 OF PACIFIC COUNTY'
 'CITY OF TACOMA - (WA)||TANNER ELECTRIC COOP'
 'CITY OF BLAINE - (WA)||PUD NO 1 OF WHATCOM COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||PACIFICORP||PUD NO 1 OF CLARK
COUNTY - (WA)'
 'BONNEVILLE POWER ADMINISTRATION||COLUMBIA RURAL ELEC ASSN, INC'
 'BONNEVILLE POWER ADMINISTRATION||PACIFICORP||COLUMBIA RURAL ELEC
ASSN, INC'
 'BONNEVILLE POWER ADMINISTRATION||CITY OF CENTRALIA - (WA)|CITY OF
TACOMA - (WA)'
```

```
 'PUD NO 1 OF WHATCOM COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF WAHKIAKUM COUNTY'
 'CITY OF CHENEY - (WA)' 'CITY OF CHEWELAH'
 'BONNEVILLE POWER ADMINISTRATION||BENTON RURAL ELECTRIC ASSN'
 'PUD NO 1 OF PEND OREILLE COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||BIG BEND ELECTRIC COOP, INC'
 'BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF MASON COUNTY|PUD NO 1
OF JEFFERSON COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF KITTITAS COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||TOWN OF EATONVILLE - (WA)|CITY OF
TACOMA - (WA)'
 'BONNEVILLE POWER ADMINISTRATION||TOWN OF RUSTON - (WA)|CITY OF
TACOMA - (WA)||PENINSULA LIGHT COMPANY'
 'BONNEVILLE POWER ADMINISTRATION||CITY OF TACOMA - (WA)||BENTON RURAL
ELECTRIC ASSN|PENINSULA LIGHT COMPANY'
 'CITY OF SEATTLE - (WA)'
 'CITY OF SUMAS - (WA)||PUD NO 1 OF WHATCOM COUNTY'
 'BONNEVILLE POWER ADMINISTRATION||CITY OF COULEE DAM - (WA)'
 'BONNEVILLE POWER ADMINISTRATION||PENINSULA LIGHT COMPANY'
 'CITY OF TACOMA - (WA)'
 'BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF ASOTIN COUNTY'
 'PORTLAND GENERAL ELECTRIC CO'
 'BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF ASOTIN COUNTY||INLAND
POWER & LIGHT COMPANY'
 'BONNEVILLE POWER ADMINISTRATION||NESPELEM VALLEY ELEC COOP, INC'
 'BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF CLALLAM COUNTY|PUD NO 1
OF JEFFERSON COUNTY']
Nunique values: 73
--------------------------------------------------------
```

```python
# All of the categorical data columns
print(categorical_df.columns)
```

```
Index(['VIN (1-10)', 'County', 'City', 'State', 'Make', 'Model',
       'Electric Vehicle Type',
       'Clean Alternative Fuel Vehicle (CAFV) Eligibility', 'Vehicle
Location',
       'Electric Utility'],
      dtype='object')
```

```python
# Analyze unique counts for categorical columns
print("Unique Makes  :", df['Make'].nunique())
print("Unique Models :", df['Model'].nunique())

# Top 10 most common vehicle makes
top_makes = df['Make'].value_counts().nlargest(10)
plt.figure(figsize=(10, 6))
sns.barplot(x=top_makes.index, y=top_makes.values,palette='viridis')
plt.title('Top 10 Vehicle Makes')
plt.xticks(rotation=45)
plt.show()

Unique Makes  : 34
Unique Models : 114
```

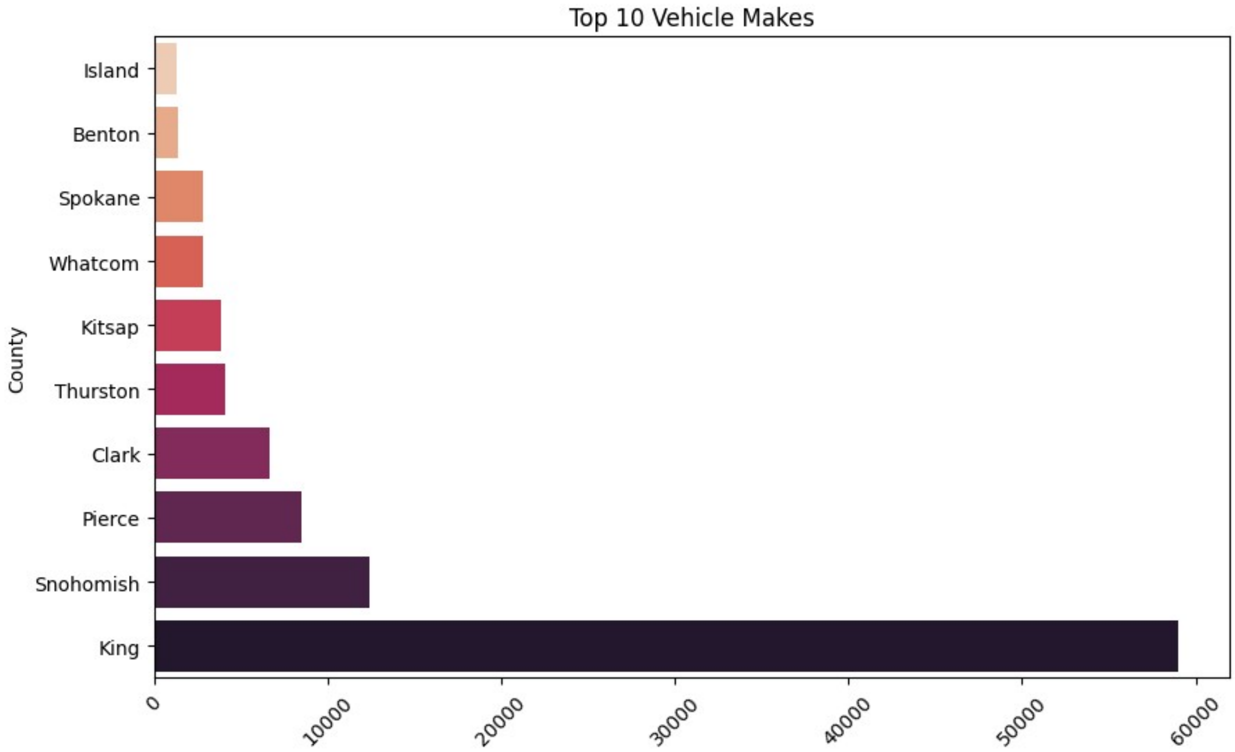


Question : Which county has the highest population?

```
top_10_counties =
df["County"].value_counts().sort_values(ascending=False).head(10)

print("*"*5, "Top 10 Counties", "*"*5)
for county, count in top_10_counties.items():
  print("{0} : {1}".format(county, count))

***** Top 10 Counties *****
King : 58980
Snohomish : 12412
Pierce : 8525
Clark : 6681
Thurston : 4109
Kitsap : 3828
Whatcom : 2839
```

```
Spokane : 2785
Benton : 1376
Island : 1298
```

```
top_10_counties_sorted = top_10_counties.sort_values(ascending=True)

plt.figure(figsize=(10, 6))
ax = sns.barplot(y=top_10_counties_sorted.index,
x=top_10_counties_sorted.values, palette='rocket_r')
plt.title('Top 10 Vehicle Makes')
plt.xticks(rotation=45)
plt.show()
```



```
df["City"].value_counts().sort_values(ascending=False).head(10)
# df["Model"].value_counts()
# df["Electric Vehicle Type"].value_counts().plot(kind = "barh")
# df["Clean Alternative Fuel Vehicle (CAFV)
Eligibility"].value_counts().plot(kind = "barh")
```

```
City
Seattle      20295
Bellevue      5919
Redmond       4199
Vancouver     4013
Kirkland      3598
Bothell       3334
```

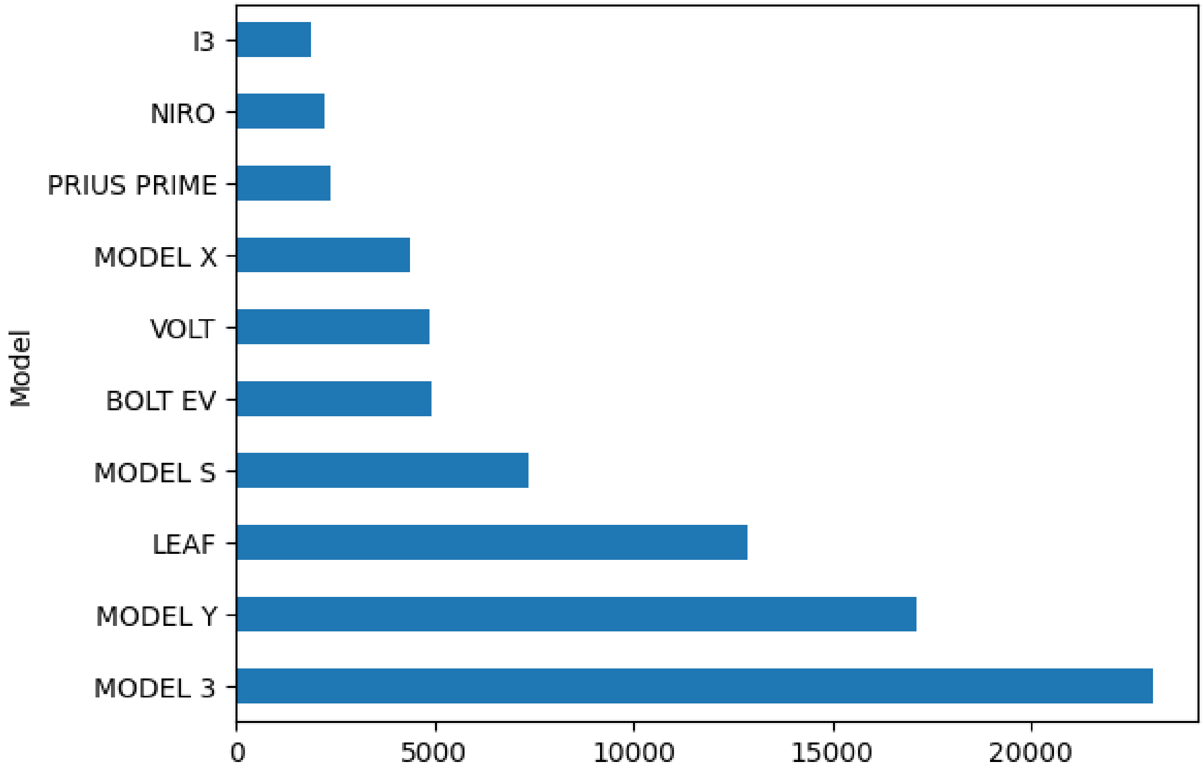
```
Sammamish      3291
Renton         2777
Olympia        2729
Tacoma         2375
Name: count, dtype: int64
```

Question:

How does the popularity of electric car models compare, and what trends can be observed from the high sales of the Model 3 and Model Y versus the lower sales of models like the i3 and Niro?

```
df["Model"].value_counts().head(10).plot(kind = "barh")
```

```
<Axes: ylabel='Model'>
```



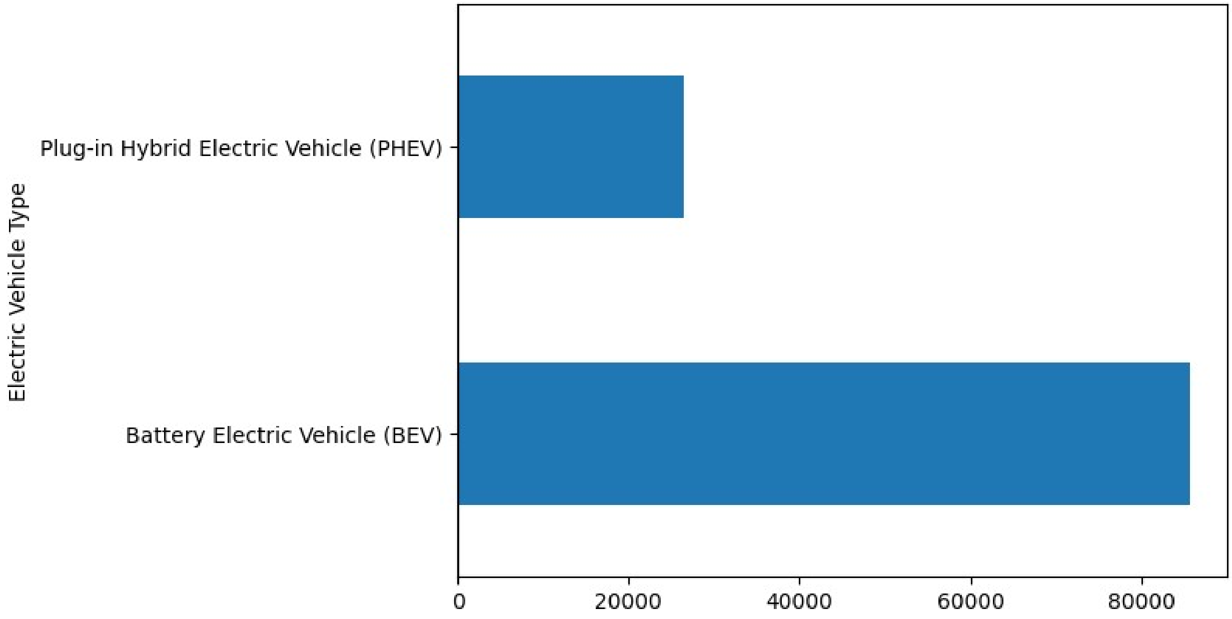
Insight:

- The popularity of different electric car models.
- The most popular model is the Model 3, followed by the Model Y.
- The least popular models are the i3 and Niro.
- The overall popularity of electric cars seems to be increasing, as evidenced by the relatively high sales of the Model 3 and Model Y.

Question:

What does the distribution of electric vehicle types reveal about consumer preferences, given the higher popularity of Battery Electric Vehicles (BEV) compared to Plug-in Hybrid Electric Vehicles (PHEV), and how might factors like range and operating costs influence this trend?

```
df["Electric Vehicle Type"].value_counts().plot(kind = "barh")

<Axes: ylabel='Electric Vehicle Type'>
```



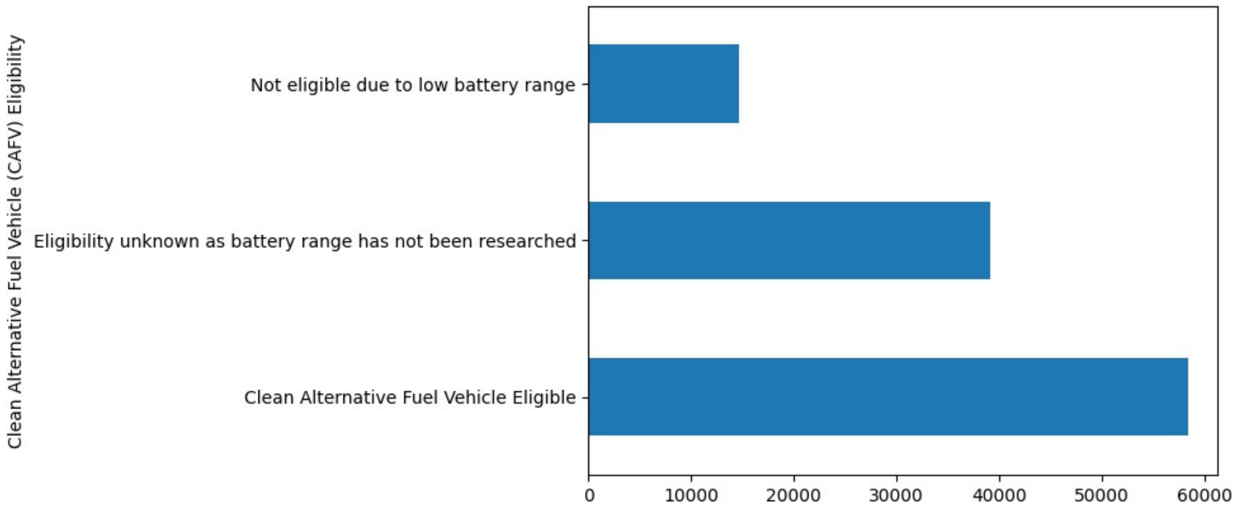
Insights:

- The distribution of electric vehicle types.
- The majority of electric vehicles are Battery Electric Vehicles (BEV), while a smaller portion are Plug-in Hybrid Electric Vehicles (PHEV).
- This suggests that BEVs are currently more popular among consumers, potentially due to factors such as longer range and lower operating costs.

Question:

How does the eligibility of electric vehicles for Clean Alternative Fuel Vehicle (CAFV) incentives vary, and what role does limited battery range and lack of research play in determining eligibility?

```
df["Clean Alternative Fuel Vehicle (CAFV)
Eligibility"].value_counts().plot(kind = "barh")

<Axes: ylabel='Clean Alternative Fuel Vehicle (CAFV) Eligibility'>
```

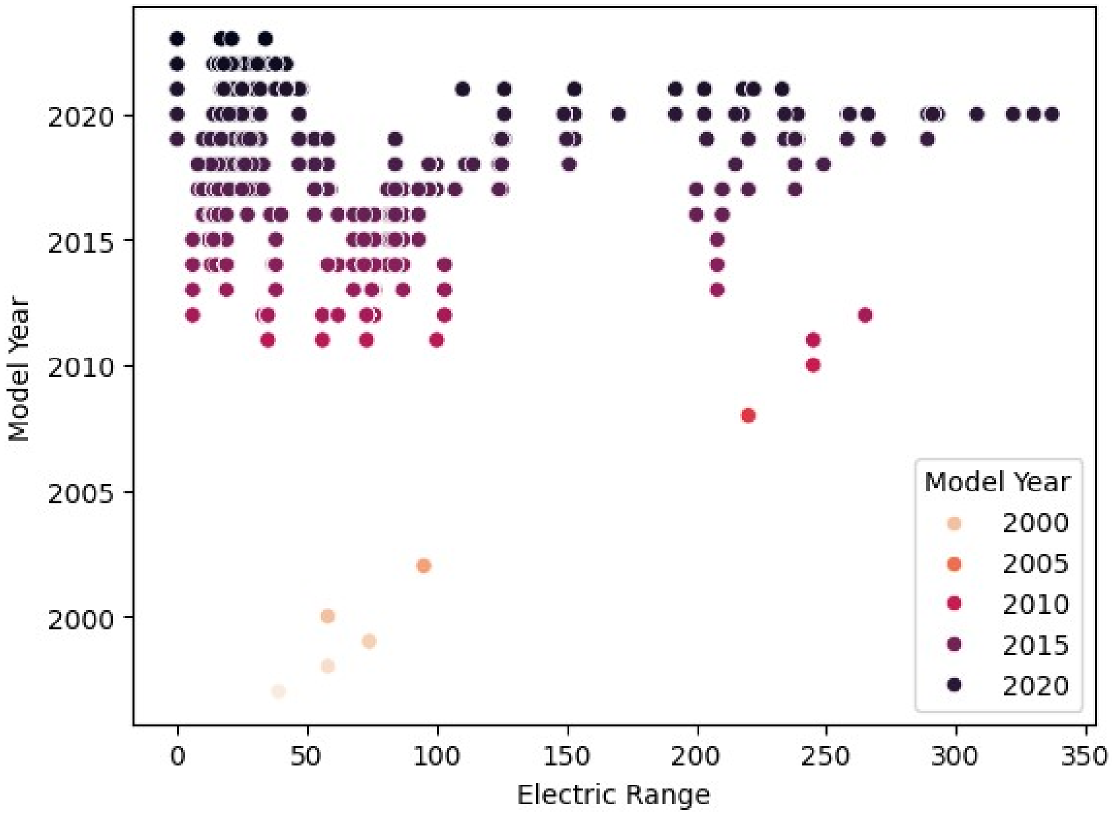


Insights:

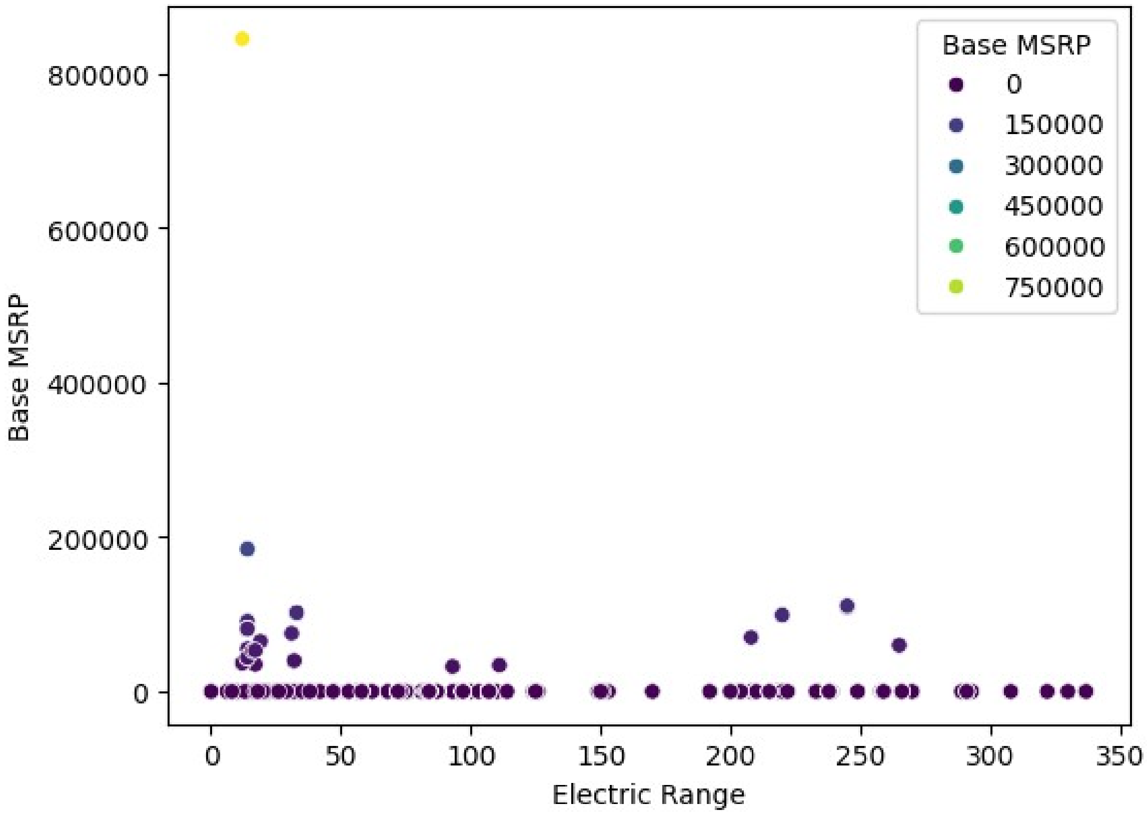
- The eligibility of different electric vehicles for Clean Alternative Fuel Vehicle (CAFV) incentives.
- The majority of vehicles are eligible, while some are ineligible due to low battery range.
- A significant number of vehicles have not been researched for battery range, and their eligibility is therefore unknown.
- This suggests that there is a need for more information on electric vehicle battery ranges to determine their eligibility for CAFV incentives.

Bivariate Analysis

```
# Categorical vs Categorical
correlation = ev_df['Electric Range'].corr(ev_df['Model Year'])
print("Pearson Correlation Coefficient between 'Electric Range' and
'Model Year':{:.2f}".format(correlation))

Pearson Correlation Coefficient between 'Electric Range' and 'Model
Year':-0.29

correlation = ev_df['Electric Range'].corr(ev_df['Base MSRP'])

print("Pearson Correlation Coefficient between 'Electric Range' and
'Model Year':{:.2f}".format(correlation))

Pearson Correlation Coefficient between 'Electric Range' and 'Model
Year':0.09

sns.scatterplot(df,x= df["Electric Range"],y=df["Model
Year"],hue=df["Model Year"], palette='rocket_r')
plt.show()
```



```
sns.scatterplot(df,x=df["Electric Range"],y = df["Base MSRP"],hue =
df["Base MSRP"], palette='viridis')
plt.show()
```



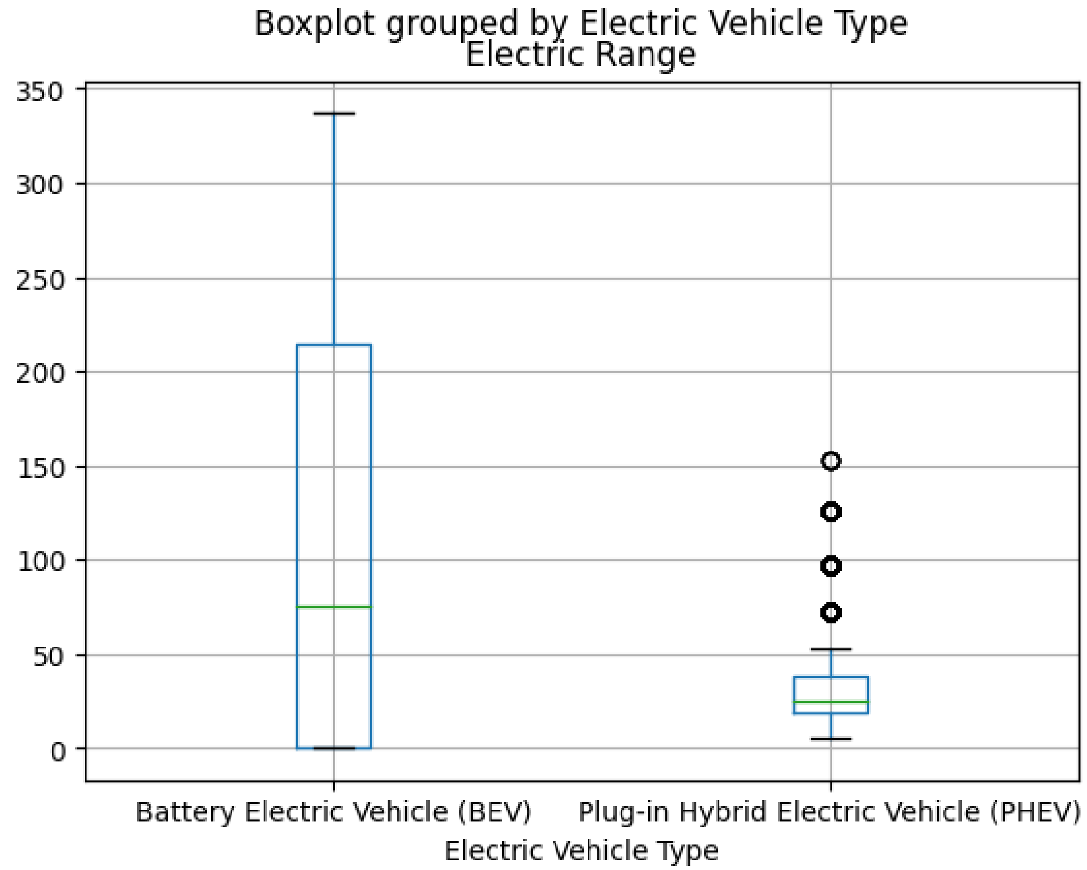
```
# Categorical Vs Numerical
df.groupby(by = "Make")["Model Year"].value_counts()

Make   Model Year
AUDI   2022          584
       2021          542
       2019          387
       2020          224
       2016          214
                    ...
VOLVO  2019          190
       2020          162
       2017          115
       2016          112
       2023            1
Name: count, Length: 209, dtype: int64

df.groupby(by = "Make")["Electric Range"].mean().head(10)

Make
AUDI                62.628448
AZURE DYNAMICS      56.000000
BENTLEY             18.666667
BMW                 46.681545
CADILLAC            35.537037
```

```
CHEVROLET         109.862032
CHRYSLER           32.360674
FIAT               85.628049
FISKER             33.000000
FORD               16.840484
Name: Electric Range, dtype: float64
```

```
df.groupby(by = ["Make","County"])["Model Year"].value_counts()
```

```
Make   County   Model Year
AUDI   Adams    2017          1
       Benton   2022         16
                2021          7
                2017          2
                2019          1
                             ..
VOLVO  Whitman  2022          1
       Yakima   2022          4
                2021          3
                2018          2
                2016          1
Name: count, Length: 3749, dtype: int64
```

```
df.boxplot(column="Electric Range", by = "Electric Vehicle
Type")#figsize = (8,6))
plt.show()
```



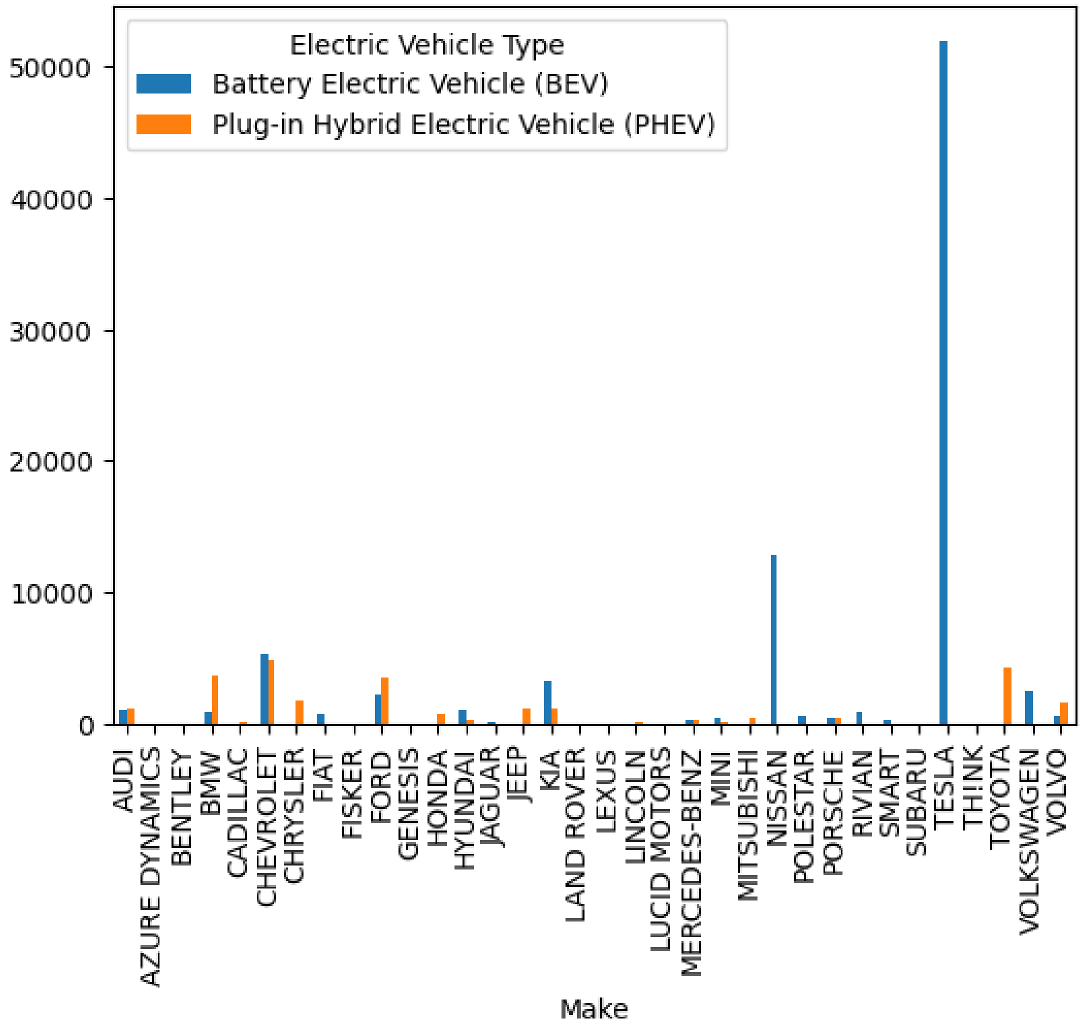
```
# Categorical Vs Categorical
df.head()

{"type":"dataframe","variable_name":"df"}

cross_tab = pd.crosstab(df["Make"],df["County"]).count()
cross_tab

County
Adams           34
Asotin          34
Benton          34
Chelan          34
Clallam         34
Clark           34
Columbia        34
Cowlitz         34
Douglas         34
Ferry           34
Franklin        34
Garfield        34
Grant           34
Grays Harbor    34
```

```
Island          34
Jefferson       34
King            34
Kitsap          34
Kittitas        34
Klickitat       34
Lewis           34
Lincoln         34
Mason           34
Okanogan        34
Pacific         34
Pend Oreille    34
Pierce          34
San Juan        34
Skagit          34
Skamania        34
Snohomish       34
Spokane         34
Stevens         34
Thurston        34
Wahkiakum       34
Walla Walla     34
Whatcom         34
Whitman         34
Yakima          34
dtype: int64
```

```
grouped_df = df.groupby(by=["Make", "Model"])


pd.crosstab(index=df["County"], columns=[df["Make"], df["Model"]])
```

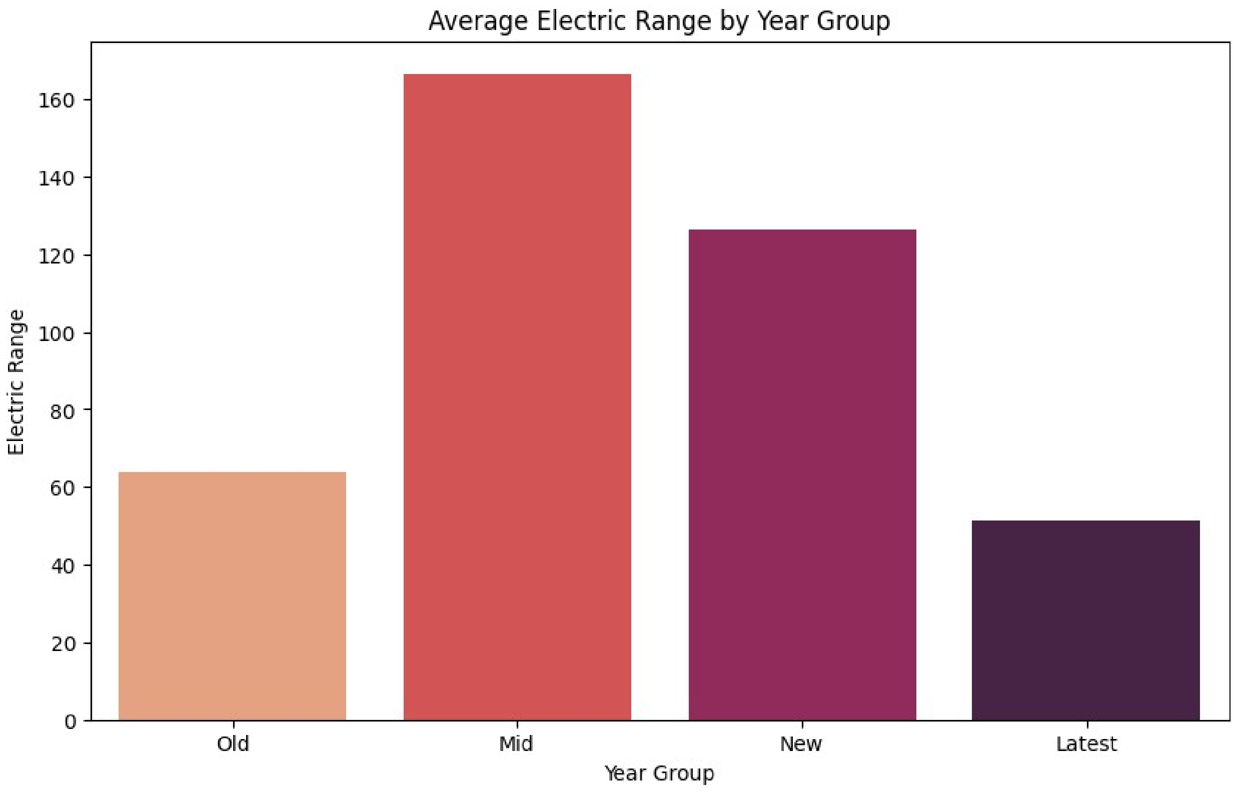
```
{"type":"dataframe"}
```

```
pd.crosstab(index=df["Make"],columns=df["Electric Vehicle
Type"]).plot(kind = "bar")
plt.show()
```



Optional Step : Feature Engineering

```
# Feature 1: Price per Electric Mile
df['Price per Mile'] = df['Base MSRP'] / df['Electric Range']

# Feature 2: Binning Model Year into categories
bins = [1980, 2000, 2010, 2020, 2025]
labels = ['Old', 'Mid', 'New', 'Latest']
df['Year Group'] = pd.cut(df['Model Year'], bins=bins, labels=labels,
right=False)

# Groupby Year Group and summarize MSRP and Electric Range
grouped_year = df.groupby('Year Group').agg({'Base MSRP': 'mean',
'Electric Range': 'mean'}).reset_index()
plt.figure(figsize=(10, 6))
sns.barplot(x='Year Group', y='Electric Range', data=grouped_year,
palette='rocket_r')
plt.title('Average Electric Range by Year Group')
plt.show()

# Feature 3: Create region feature from postal code (e.g., group by
first digit or state)
df['Region'] = df['Postal Code'].str[0]  # Example: use the first
digit of postal code as a proxy for region
```



Create a Choropleth using plotly.express to display the number of EV vehicles based on location.

```
!pip install plotly
```

```
Requirement already satisfied: plotly in
/usr/local/lib/python3.10/dist-packages (5.24.1)
Requirement already satisfied: tenacity>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from plotly) (9.0.0)
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from plotly) (24.1)
```

```
import plotly.express as px
```

```
ev_count_per_state =
ev_df.groupby('State').size().reset_index(name='ev_count')
ev_count_per_state.head(10)
```

```
{"summary":"{\n  \"name\": \"ev_count_per_state\",\n  \"rows\": 45,\n
\"fields\": [\n    {\n      \"column\": \"State\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 45,\n        \"samples\": [\n          \"TX\",\
n          \"NE\",\n          \"NH\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"ev_count\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 16746,\n
\"min\": 1,\n        \"max\": 112348,\n        \"num_unique_values\":
14,\n        \"samples\": [\n          5,\n          14,\n          1\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"ev_count_per_state"}
```

```
fig = px.choropleth(ev_count_per_state,
                    locations='State',  # Column representing state
locations
                    locationmode="USA-states",  # Use USA state-level
mapping
                    color='ev_count',  # Column representing the count
of EVs
                    scope="usa",  # Focus the map on the USA
                    color_continuous_scale="Viridis",  # Color scale
                    title='Number of EV Vehicles by State')

fig.show()

try:
  fig.write_image("choropleth_map.png")
  print("File saved.")
except Exception as e:
  print(f"An error occurred: {e}")
```

```
An error occurred:
Image export using the "kaleido" engine requires the kaleido package,
which can be installed using pip:
    $ pip install -U kaleido
```

Create a Racing Bar Plot to display the animation of EV Make and its count each year.

```
!pip install bar_chart_race
```

```
Requirement already satisfied: bar_chart_race in
/usr/local/lib/python3.10/dist-packages (0.1.0)
```

```
Requirement already satisfied: pandas>=0.24 in
/usr/local/lib/python3.10/dist-packages (from bar_chart_race) (2.2.2)
Requirement already satisfied: matplotlib>=3.1 in
/usr/local/lib/python3.10/dist-packages (from bar_chart_race) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1-
>bar_chart_race) (1.3.0)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1-
>bar_chart_race) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1-
>bar_chart_race) (4.54.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1-
>bar_chart_race) (1.4.7)
Requirement already satisfied: numpy>=1.20 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1-
>bar_chart_race) (1.26.4)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1-
>bar_chart_race) (24.1)
Requirement already satisfied: pillow>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1-
>bar_chart_race) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1-
>bar_chart_race) (3.1.4)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1-
>bar_chart_race) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=0.24-
>bar_chart_race) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.10/dist-packages (from pandas>=0.24-
>bar_chart_race) (2024.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7-
>matplotlib>=3.1->bar_chart_race) (1.16.0)
```

```python
import bar_chart_race as bcr

make_counts_per_year = df.groupby(['Model Year',
'Make']).size().reset_index(name='Count')

pivot_df = make_counts_per_year.pivot(index='Model Year',
columns='Make', values='Count').fillna(0)

# bcr.bar_chart_race(df=pivot_df,
#                     title='Electric Vehicles Make Count Over Time',

#                      n_bars=10,

#                     period_length=1000,
# )

# Create the bar chart race with color effects
bcr.bar_chart_race(
    df=pivot_df,
    title='Electric Vehicles Make Count Over Time',
    n_bars=10,                    # Top 10 bars to display
    period_length=1000,            # Speed of the race (1000 ms = 1
second per frame)
    fixed_order=False,             # Let the order of bars change as
they race
    fixed_max=False,               # Allow the bar lengths to change
dynamically
    steps_per_period=30,           # Smoother transitions between
frames
    interpolate_period=False,      # No interpolation between frames
for cleaner transitions
    period_label={'x': .99, 'y': .25, 'ha': 'right', 'va': 'center'},
# Customize the period label
    bar_label_size=7,              # Label size for the bars
    tick_label_size=8,             # Size for the tick labels
    figsize=(6, 4),                # Size of the chart
    dpi=144,                       # DPI for higher resolution
    # cmap='viridis'                 # You can also add colormap for
gradient effect
)
```

```
<IPython.core.display.HTML object>
```