In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
```

In [2]:
```
warnings.filterwarnings("ignore")
```

In [3]:
```
df = pd.read_csv(r"C:\Users\kajal\Downloads\dataset (1).csv")
```

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

Out[4]:

	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	Legislativ Distri
0	JTMEB3FV6N	Monroe	Key West	FL	33040	2022	TOYOTA	RAV4 PRIME	Plug-in Hybrid Electric Vehicle (PHEV)	Clean Alternative Fuel Vehicle Eligible	42	0	Na
1	1G1RD6E45D	Clark	Laughlin	NV	89029	2013	CHEVROLET	VOLT	Plug-in Hybrid Electric Vehicle (PHEV)	Clean Alternative Fuel Vehicle Eligible	38	0	Na
2	JN1AZ0CP8B	Yakima	Yakima	WA	98901	2011	NISSAN	LEAF	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	73	0	15
3	1G1FW6S08H	Skagit	Concrete	WA	98237	2017	CHEVROLET	BOLT EV	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	238	0	39
4	3FA6P0SU1K	Snohomish	Everett	WA	98201	2019	FORD	FUSION	Plug-in Hybrid Electric Vehicle (PHEV)	Not eligible due to low battery range	26	0	38

Data Description Vin: Vehicle Identification Number, a unique code used to identify individual motor vehicles. County: The county where the vehicle is registered City:The city where the vehicle is registered. State:The state abbreviation where the vehicle is registered. Postal Code:The postal code associated with the vehicle's registration location. Model Year:The year the vehicle model was manufactured. Make:The manufacturer or brand of the vehicle Model:The specific model of the vehicle (e.g., RAV4 PRIME, VOLT) Electric Vehicle Type:The type of electric vehicle Cafv Eligibility:Indicates whether the vehicle qualifies as a Clean Alternative Fuel Vehicle Electric Range: The maximum distance the vehicle can travel on electric power alone. Base Msrp: Manufacturer's suggested retail price for the base model of the vehicle. Legislative District:The legislative district associated with the vehicle registration. Dol Vehicle Id:Department of Licensing vehicle identification number. Vehicle Location: Geographic location of the vehicle, often in point format (longitude, latitude). Electric Utility:The utility company supplying electricity to the vehicle owner 2020 Census Tract:The census tract identifier from the 2020 census, representing a specific geographic area.

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

Out[5]:
```
(112634, 17)
```

In [6]:
```
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 [7]:
```
df.columns
```

Out[7]:
```
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 [8]:
```
df.rename(columns = {"Clean Alternative Fuel Vehicle (CAFV) Eligibility" : "CAFV Eligibility"}, inplace = True)
```

In [9]:
```
df.columns = [ col.lower().strip() for col in df.columns ]   #Fixing the column names
df.columns = df.columns.str.title()
```

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

Out[10]:
```
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 [11]:
```
df.isna().sum()
```

Out[11]:
```
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 [12]:
```
columns = ["Model", "Legislative District","Vehicle Location","Electric Utility"]
```

In [13]:
```
for col in columns:                                    # handling Missing values
   df[col] =  df[col].fillna(df[col].mode()[0])
```

In [14]:
```
df.isna().sum()
```

Out[14]:
```
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
```

In [15]:
```
df.nunique()
```

Out[15]:
```
Vin (1-10)                7548
County                     165
City                       629
State                       45
Postal Code                773
Model Year                  20
Make                        34
Model                      114
Electric Vehicle Type        2
Cafv Eligibility             3
Electric Range             101
Base Msrp                   30
Legislative District        49
Dol Vehicle Id          112634
Vehicle Location           758
Electric Utility            73
2020 Census Tract         2026
dtype: int64
```

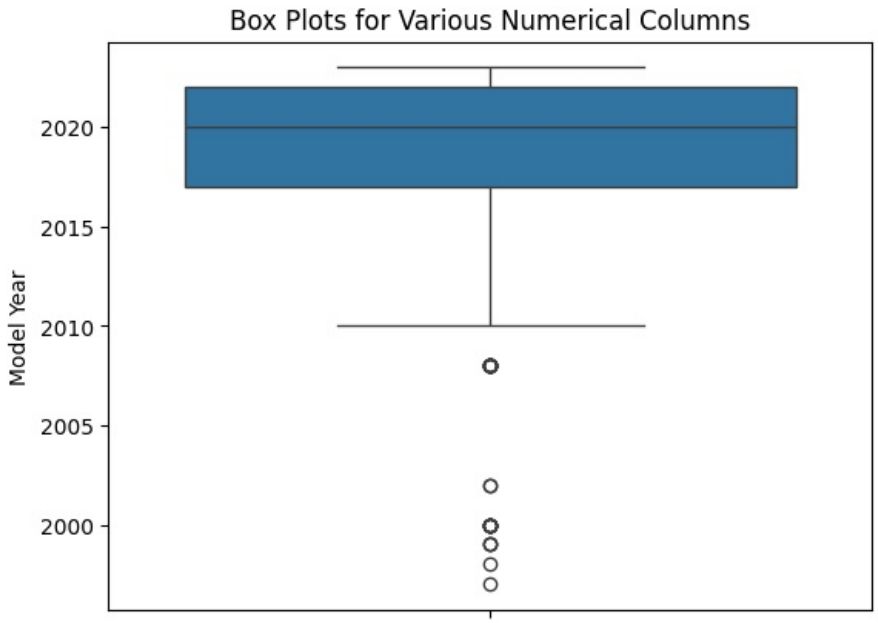
In [16]:
```
df.head()
```

Out[16]:

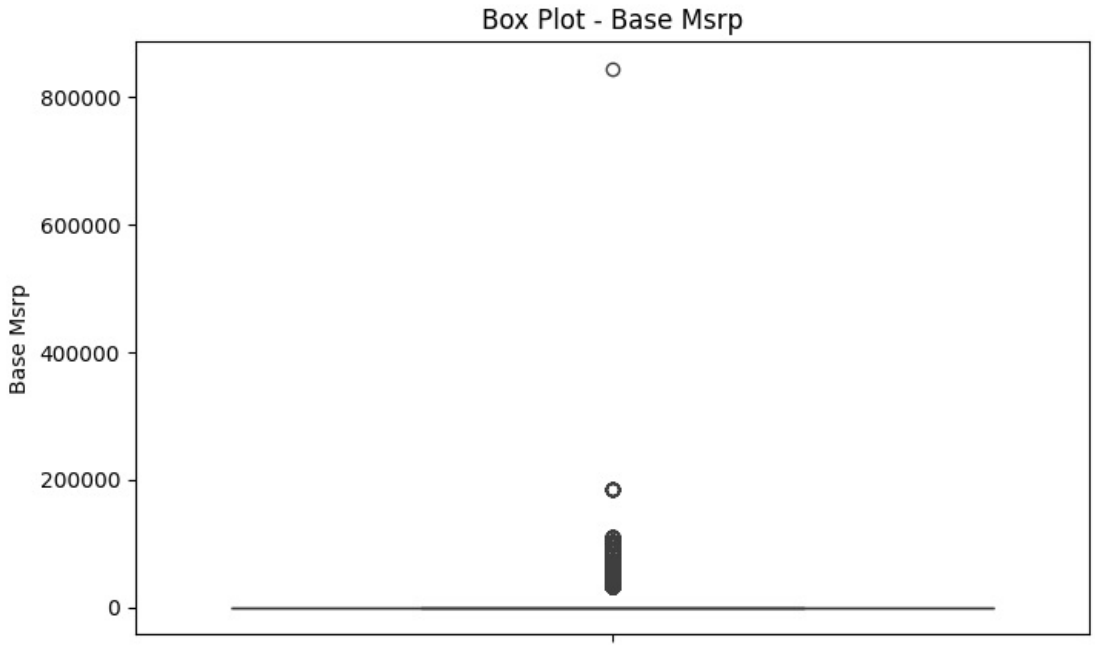
	Vin (1-10)	County	City	State	Postal Code	Model Year	Make	Model	Electric Vehicle Type	Cafv Eligibility	Electric Range	Base Msrp	Legislative District
0	JTMEB3FV6N	Monroe	Key West	FL	33040	2022	TOYOTA	RAV4 PRIME	Plug-in Hybrid Electric Vehicle (PHEV)	Clean Alternative Fuel Vehicle Eligible	42	0	41.0
1	1G1RD6E45D	Clark	Laughlin	NV	89029	2013	CHEVROLET	VOLT	Plug-in Hybrid Electric Vehicle (PHEV)	Clean Alternative Fuel Vehicle Eligible	38	0	41.0
2	JN1AZ0CP8B	Yakima	Yakima	WA	98901	2011	NISSAN	LEAF	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	73	0	15.0
3	1G1FW6S08H	Skagit	Concrete	WA	98237	2017	CHEVROLET	BOLT EV	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	238	0	39.0
4	3FA6P0SU1K	Snohomish	Everett	WA	98201	2019	FORD	FUSION	Plug-in Hybrid Electric Vehicle (PHEV)	Not eligible due to low battery range	26	0	38.0

In [17]:
```
Filterted_num_col = ["Model Year","Base Msrp","Electric Range"]
```

In [18]:
```
## plt.figure(figsize=(8, 5))
sns.boxplot(data=df["Model Year"])
plt.title('Box Plots for Various Numerical Columns')
plt.xticks(rotation=45)
plt.show()
```

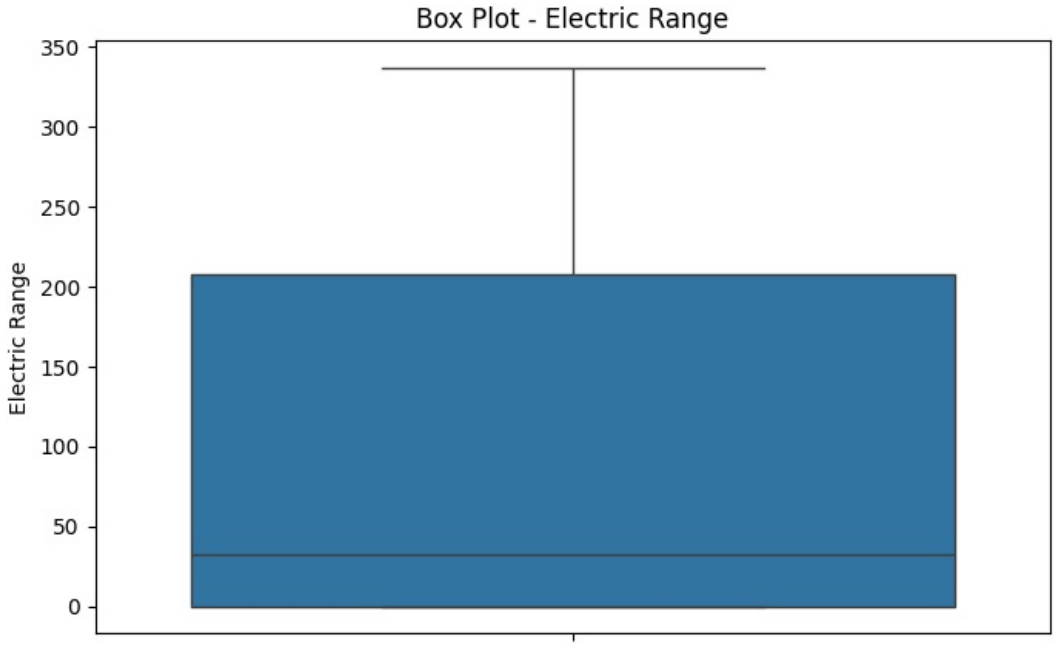


In [19]:
```
plt.figure(figsize=(8, 5))
sns.boxplot(data=df["Base Msrp"])
plt.title('Box Plot - Base Msrp')
plt.xticks(rotation=45)
plt.show()
```



In [20]:
```
plt.figure(figsize=(8, 5))
sns.boxplot(data=df["Electric Range"])
plt.title('Box Plot - Electric Range')
plt.xticks(rotation=45)
```

Out[20]:
```
([0], [Text(0, 0, '')])
```



In [21]:
```
df= df[df["Model Year"] >= 2005]
df.shape
```

Out[21]: (112617, 17)

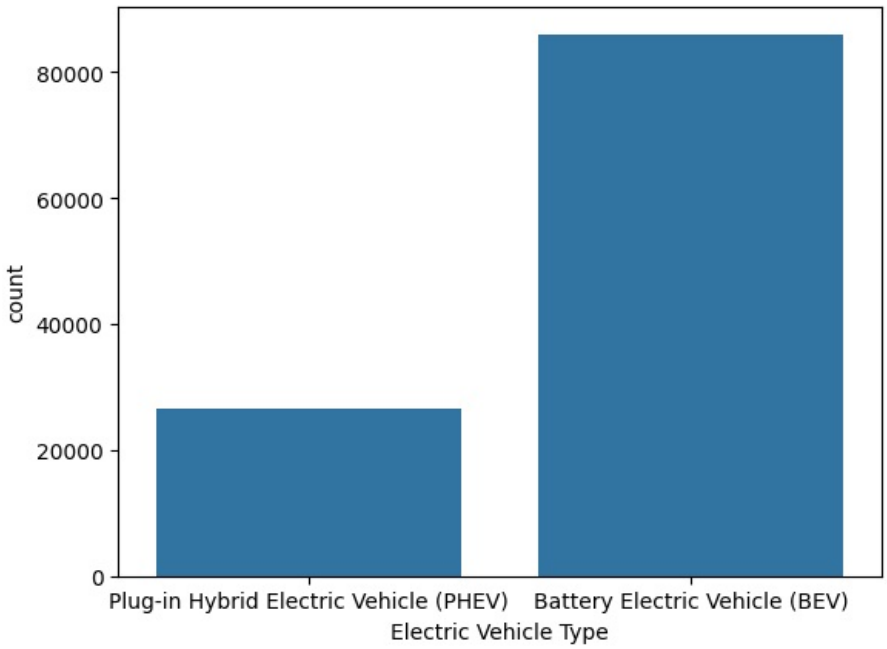
In [22]:
```
df= df[df["Base Msrp"] <= 200000]
df.shape
```

Out[22]: (112616, 17)

In [23]:
```
# Univariate - Categorical Analysis
```

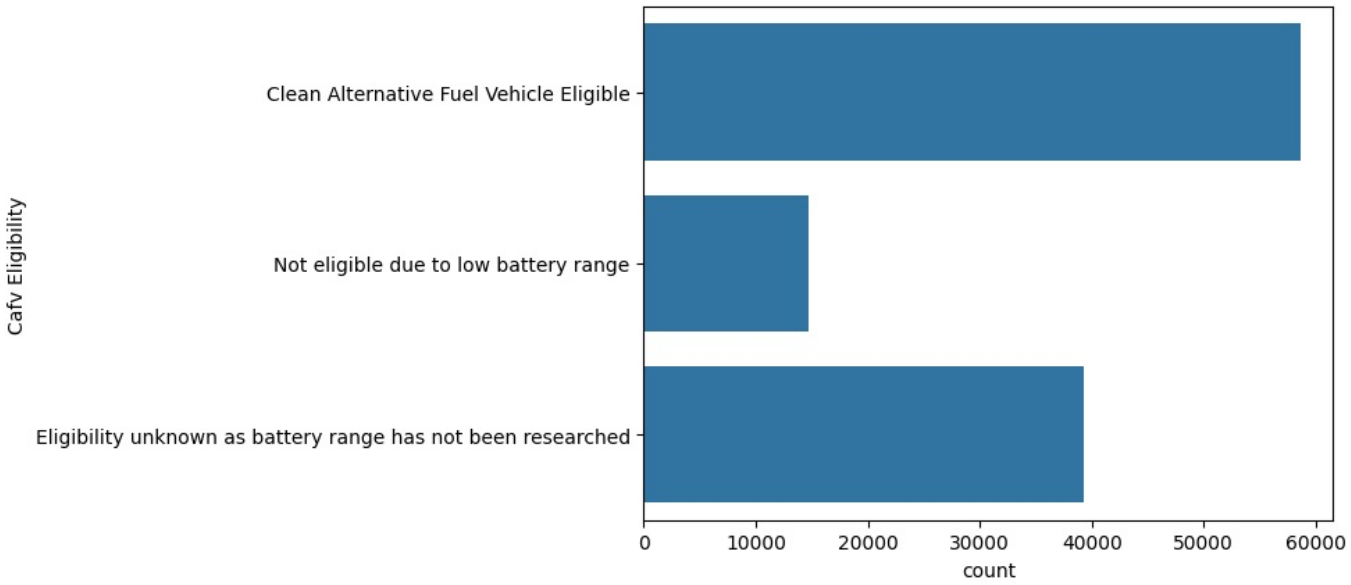
In [24]:
```
sns.countplot(data=df, x='Electric Vehicle Type')
```

Out[24]: <Axes: xlabel='Electric Vehicle Type', ylabel='count'>



In [25]:
```
sns.countplot(data=df, y='Cafv Eligibility')
```

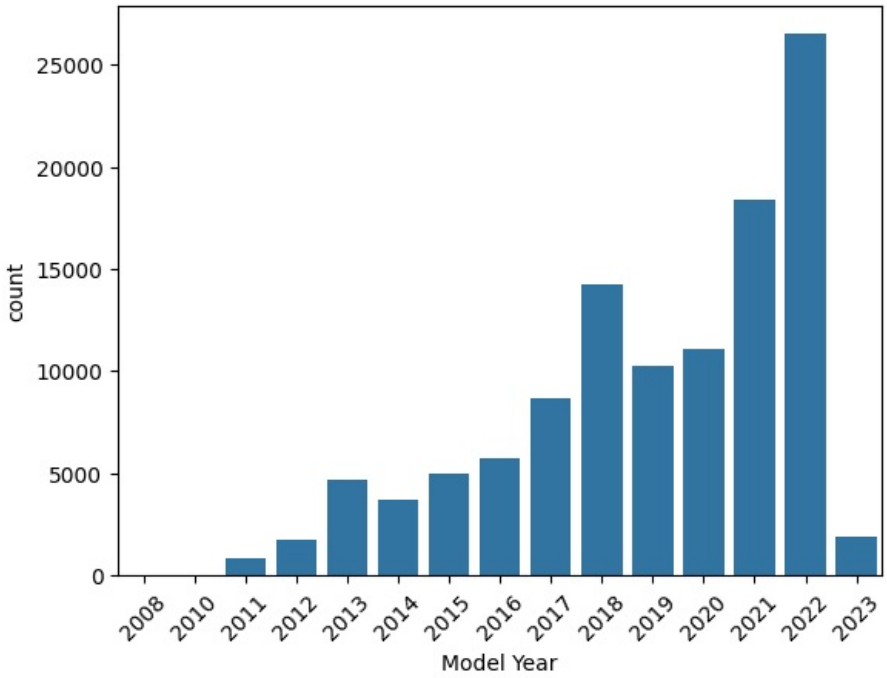
Out[25]:
```
<Axes: xlabel='count', ylabel='Cafv Eligibility'>
```



In [26]:
```
df.columns
```

Out[26]:
```
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')
```

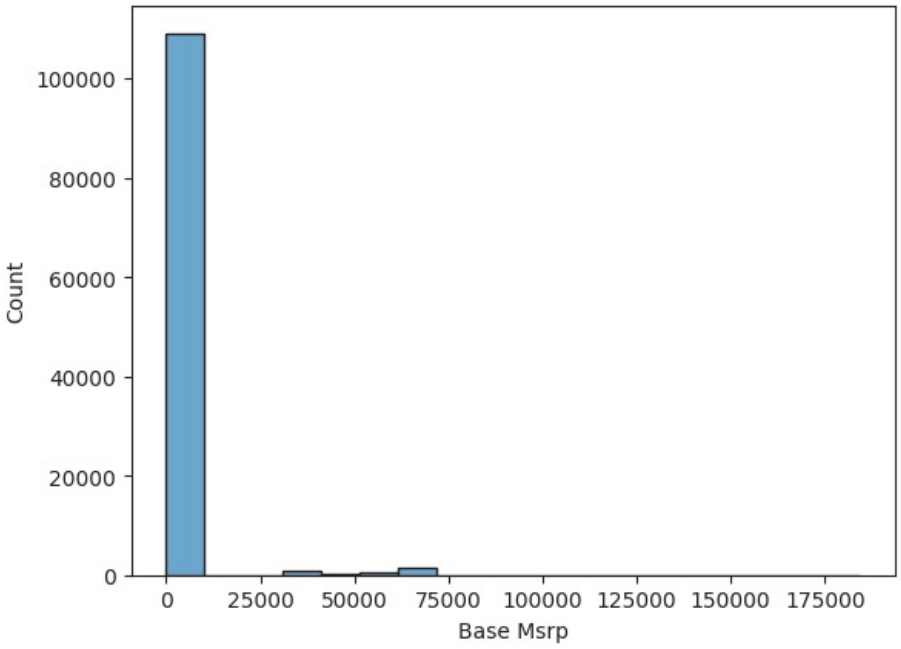
In [27]:
```
sns.countplot(data=df, x='Model Year')
plt.xticks(rotation=45)
plt.show()
```



We have highest number of vehciles with model year 2022 and lowest 2011

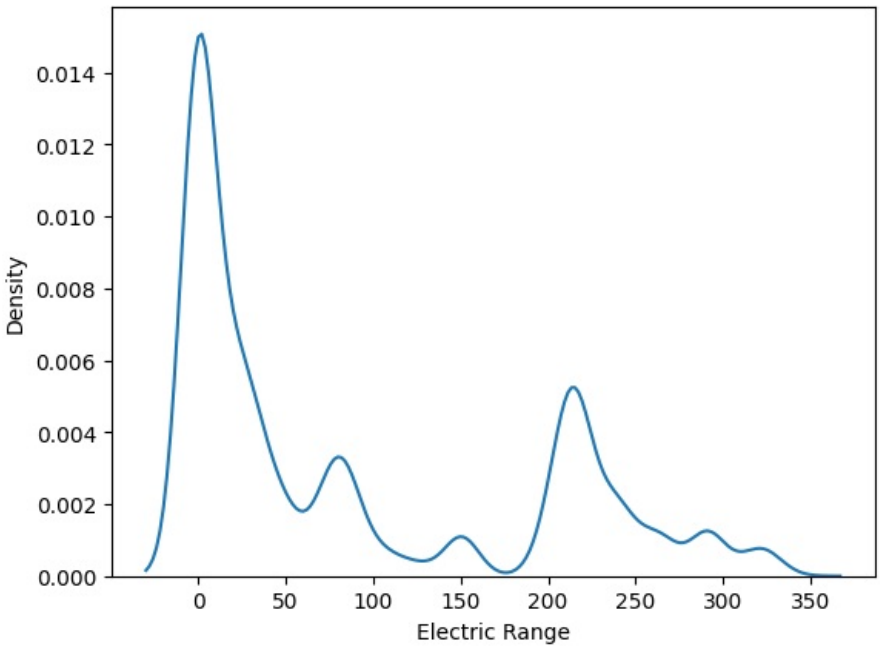
In [28]:
```
sns.histplot(data=df, x='Base Msrp')
```

Out[28]:
```
<Axes: xlabel='Base Msrp', ylabel='Count'>
```



In [29]:
```
sns.kdeplot(data=df, x='Electric Range')
```

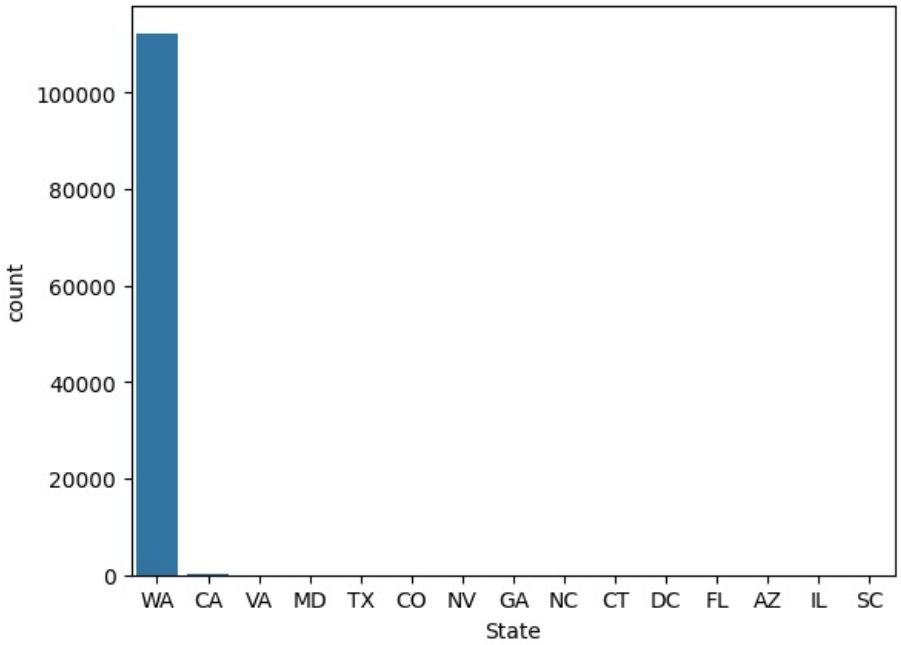
Out[29]:
```
<Axes: xlabel='Electric Range', ylabel='Density'>
```



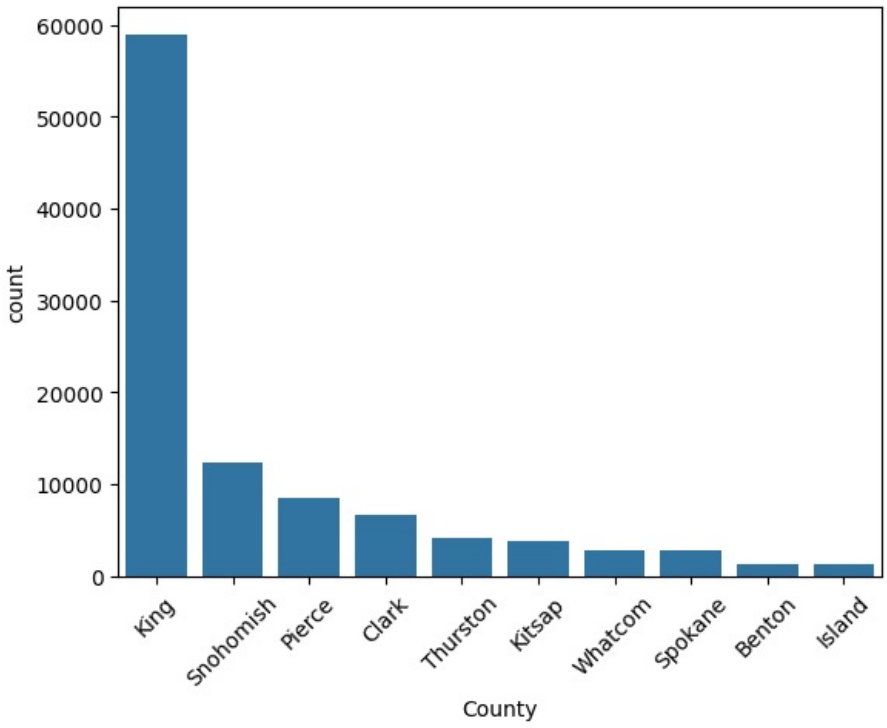
Top States registred EV Vehicles

In [30]:
```
df_top_state = df['State'].value_counts().nlargest(15).index
filtered_df = df[df['State'].isin(df_top_state)]
sns.countplot(x='State', data=filtered_df, order=df_top_state)
```

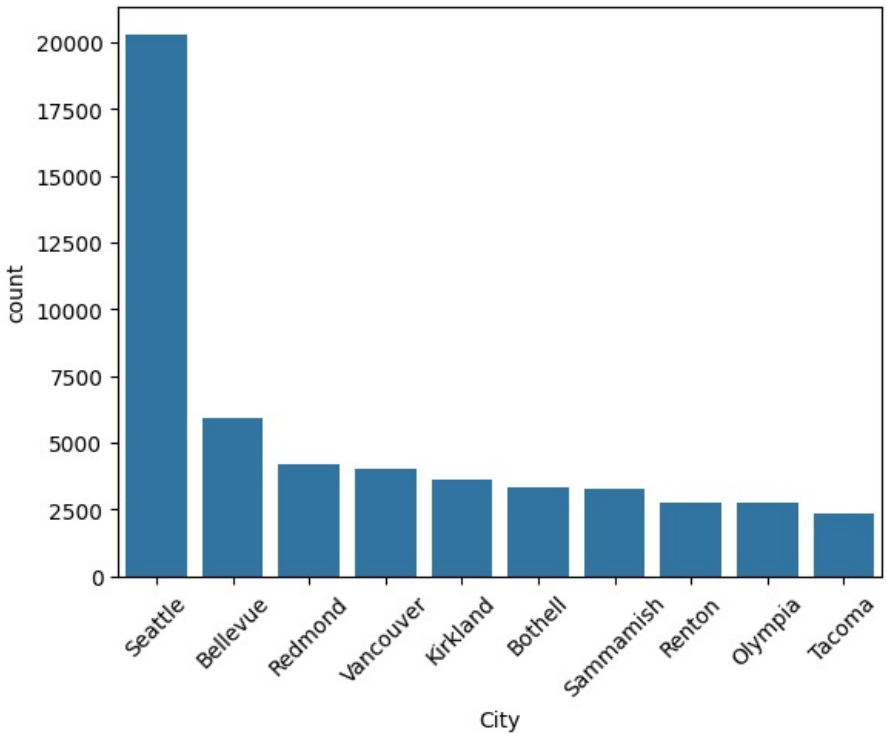
Out[30]: <Axes: xlabel='State', ylabel='count'>



In [31]:
```
df_top_County = df['County'].value_counts().nlargest(10).index
filtered_df = df[df['County'].isin(df_top_County)]
sns.countplot(x='County', data=filtered_df, order=df_top_County)
plt.xticks(rotation=45)
plt.show()
```



In [32]:
```
df_top_City = df['City'].value_counts().nlargest(10).index
filtered_df = df[df['City'].isin(df_top_City)]
sns.countplot(x='City', data=filtered_df, order=df_top_City)
plt.xticks(rotation=45)
plt.show()
```

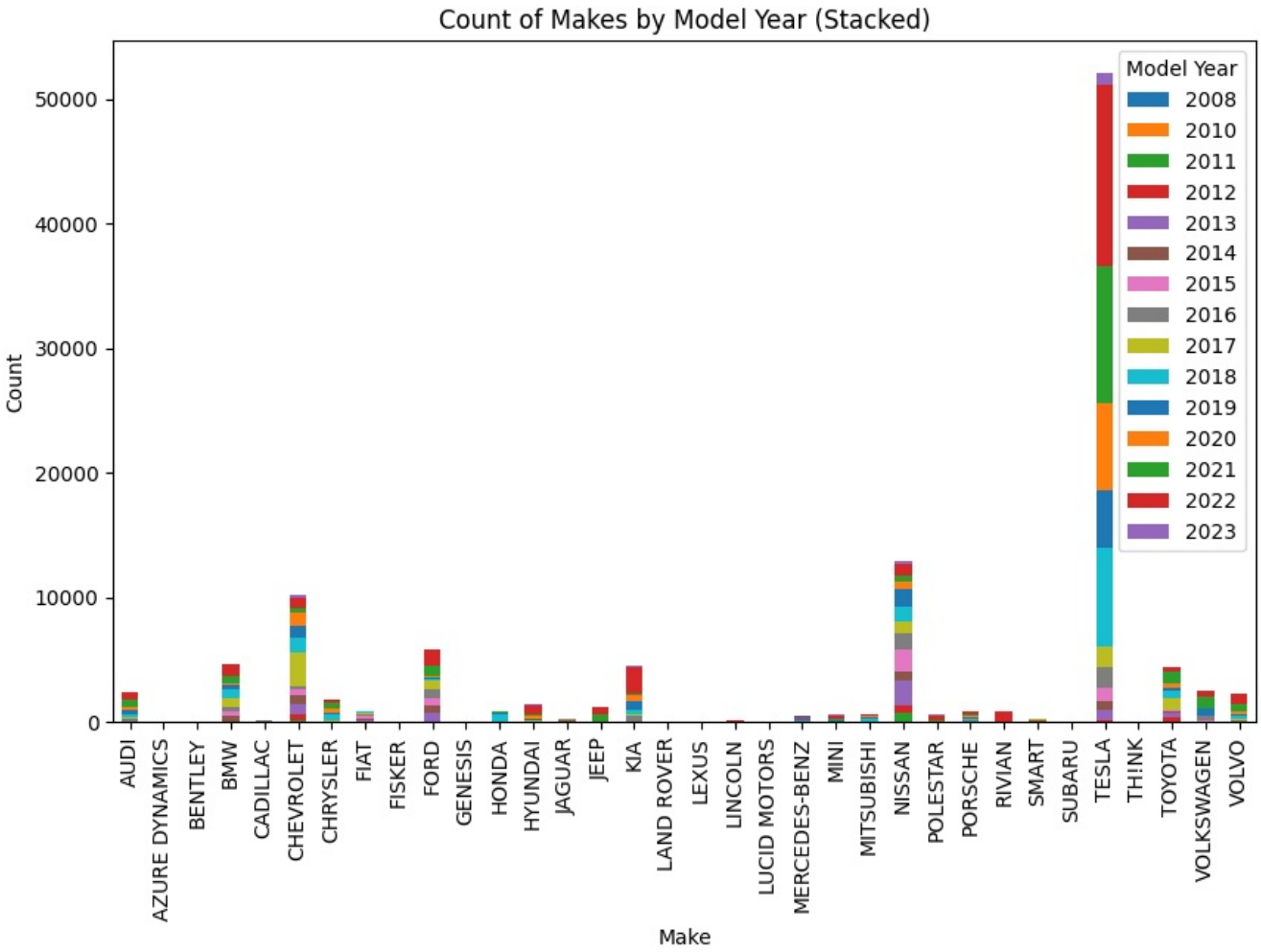


Bivariant Analysis

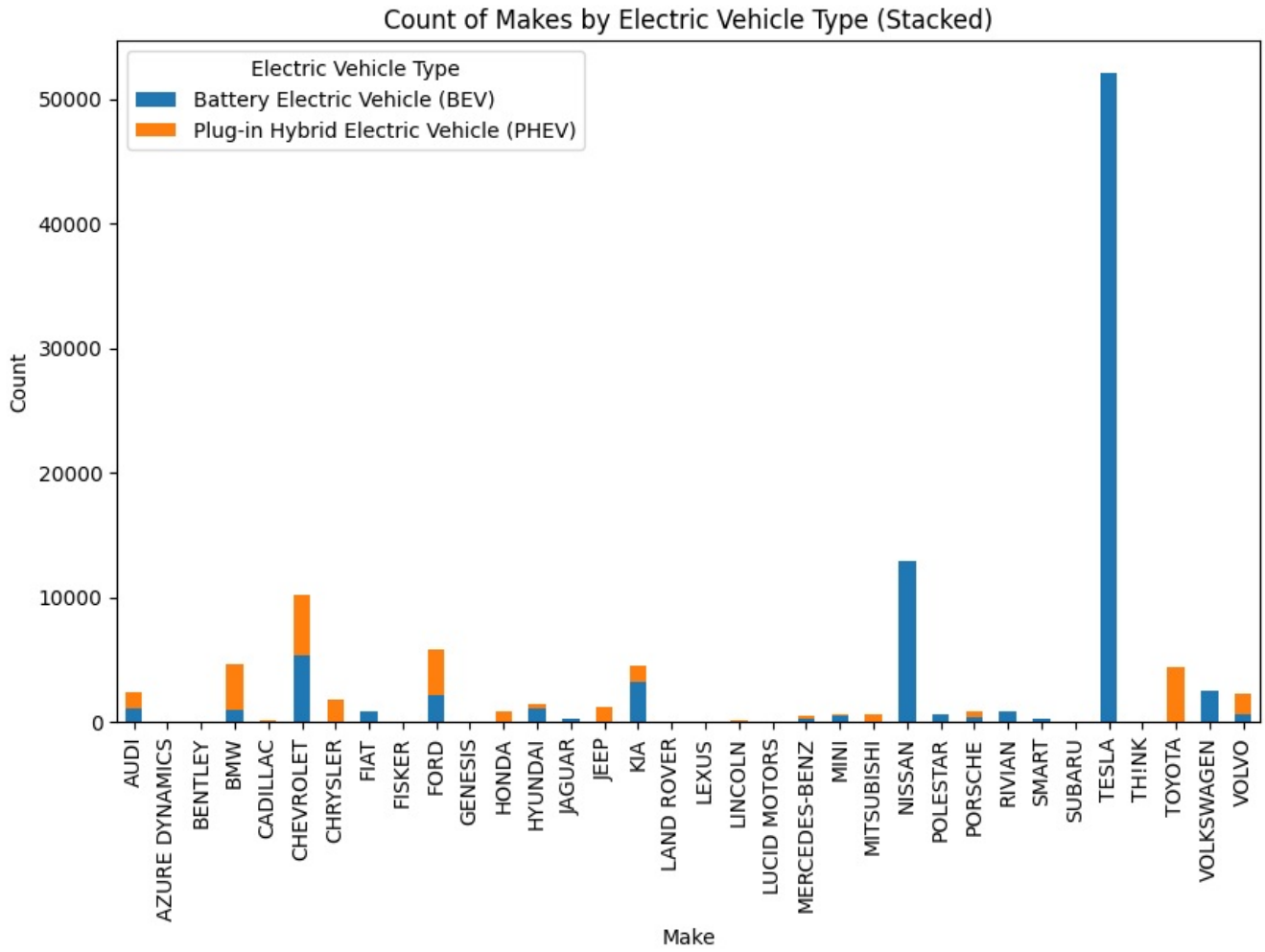
In [33]:
```
df.columns
```

Out[33]:
```
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 [34]:
```
counts = df.groupby(['Make', 'Model Year']).size().unstack()
counts.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('Count of Makes by Model Year (Stacked)')
plt.xlabel('Make')
plt.ylabel('Count')
plt.legend(title='Model Year')
plt.show()
```

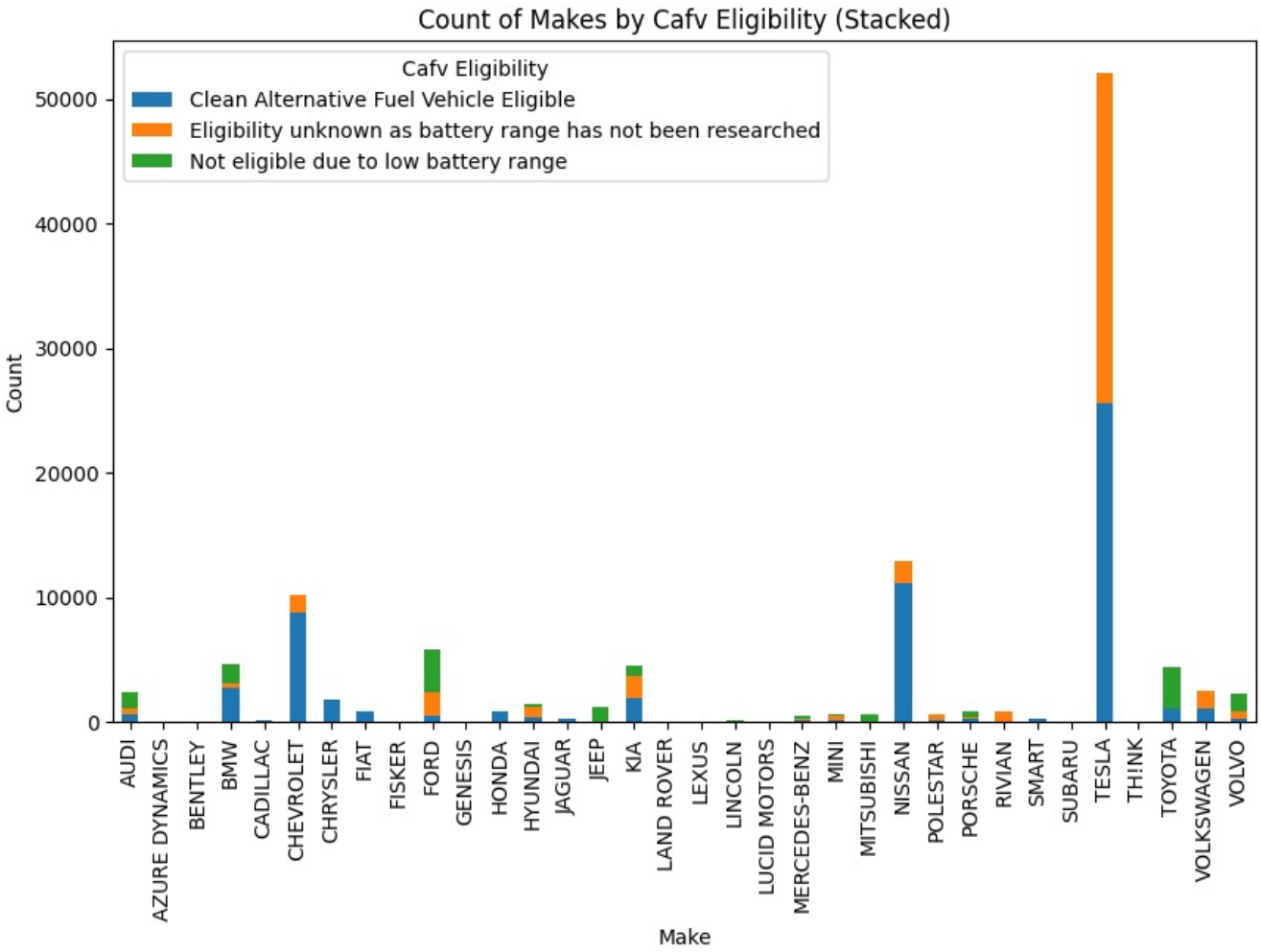


In [35]:
```
counts = df.groupby(['Make', 'Electric Vehicle Type']).size().unstack()
counts.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('Count of Makes by Electric Vehicle Type (Stacked)')
plt.xlabel('Make')
plt.ylabel('Count')
plt.legend(title='Electric Vehicle Type')
plt.show()
```



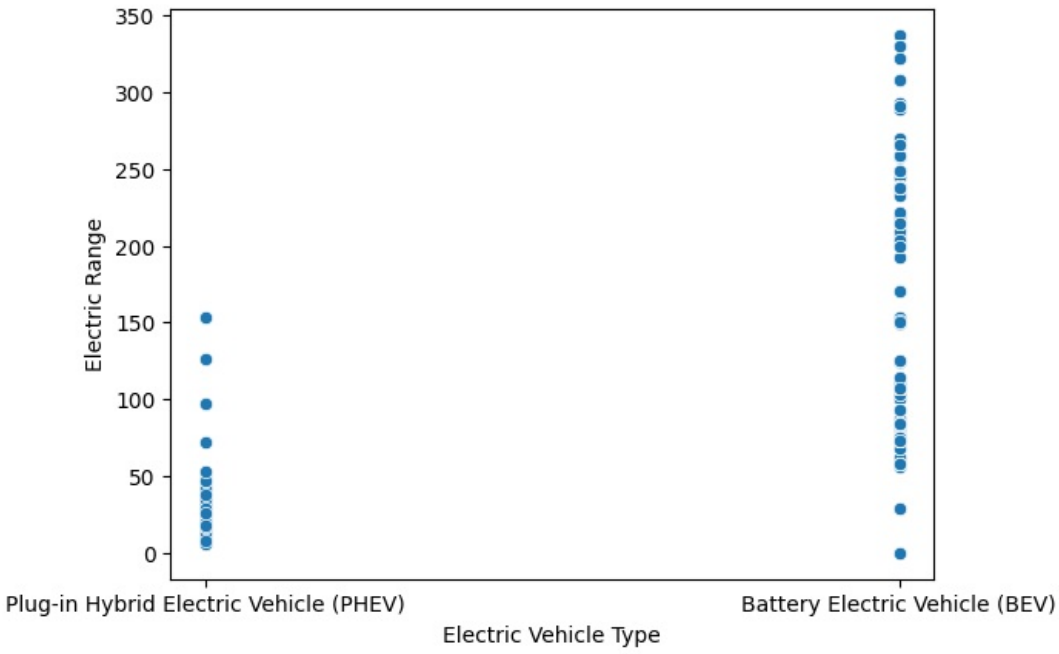
Tesla and Nissan has only BEV vehicles and Toyata & Jeep has only PHEV vehicles

In [36]:
```
counts = df.groupby(['Make', 'Cafv Eligibility']).size().unstack()
counts.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('Count of Makes by Cafv Eligibility (Stacked)')
plt.xlabel('Make')
plt.ylabel('Count')
plt.legend(title='Cafv Eligibility')
plt.show()
```

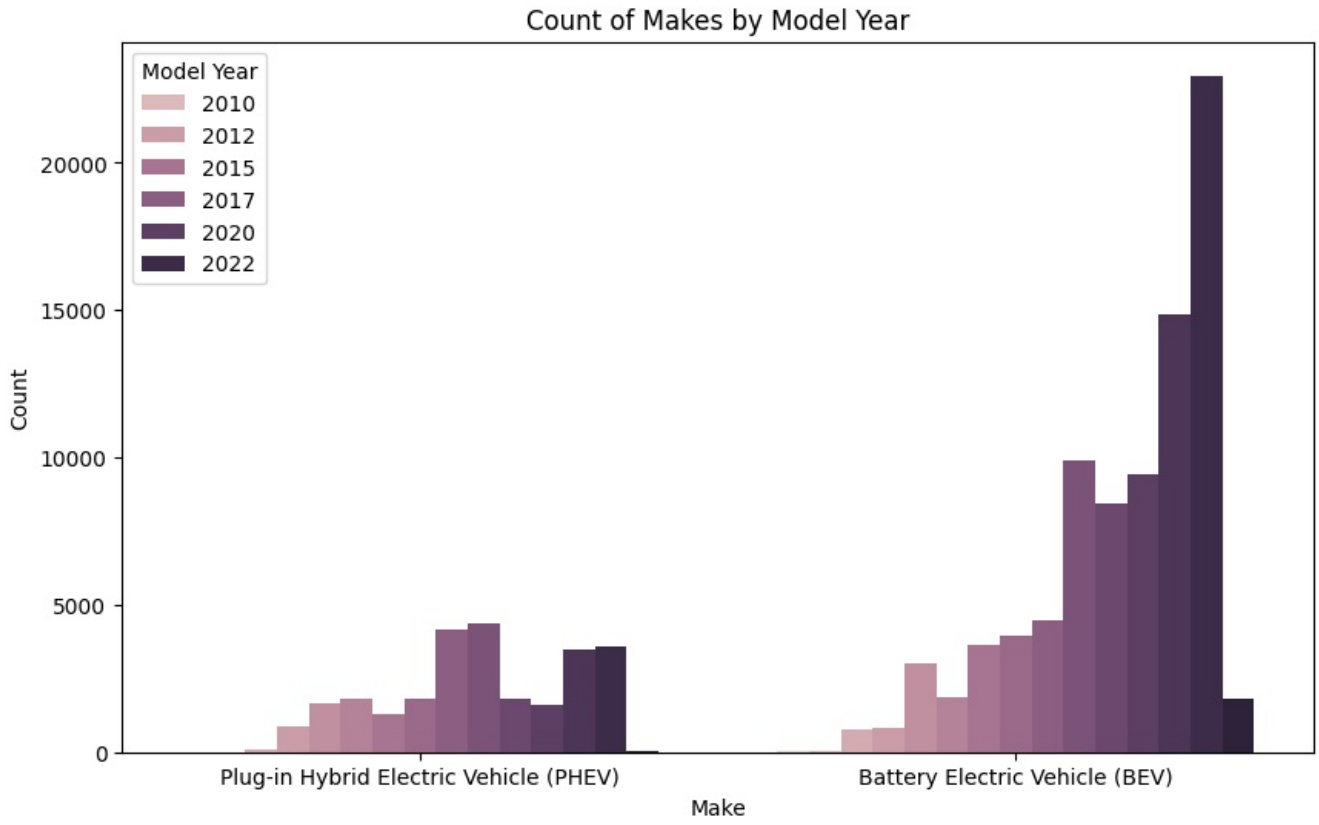


In [37]:
```
sns.scatterplot(data = df, x = 'Electric Vehicle Type', y = 'Electric Range')
```

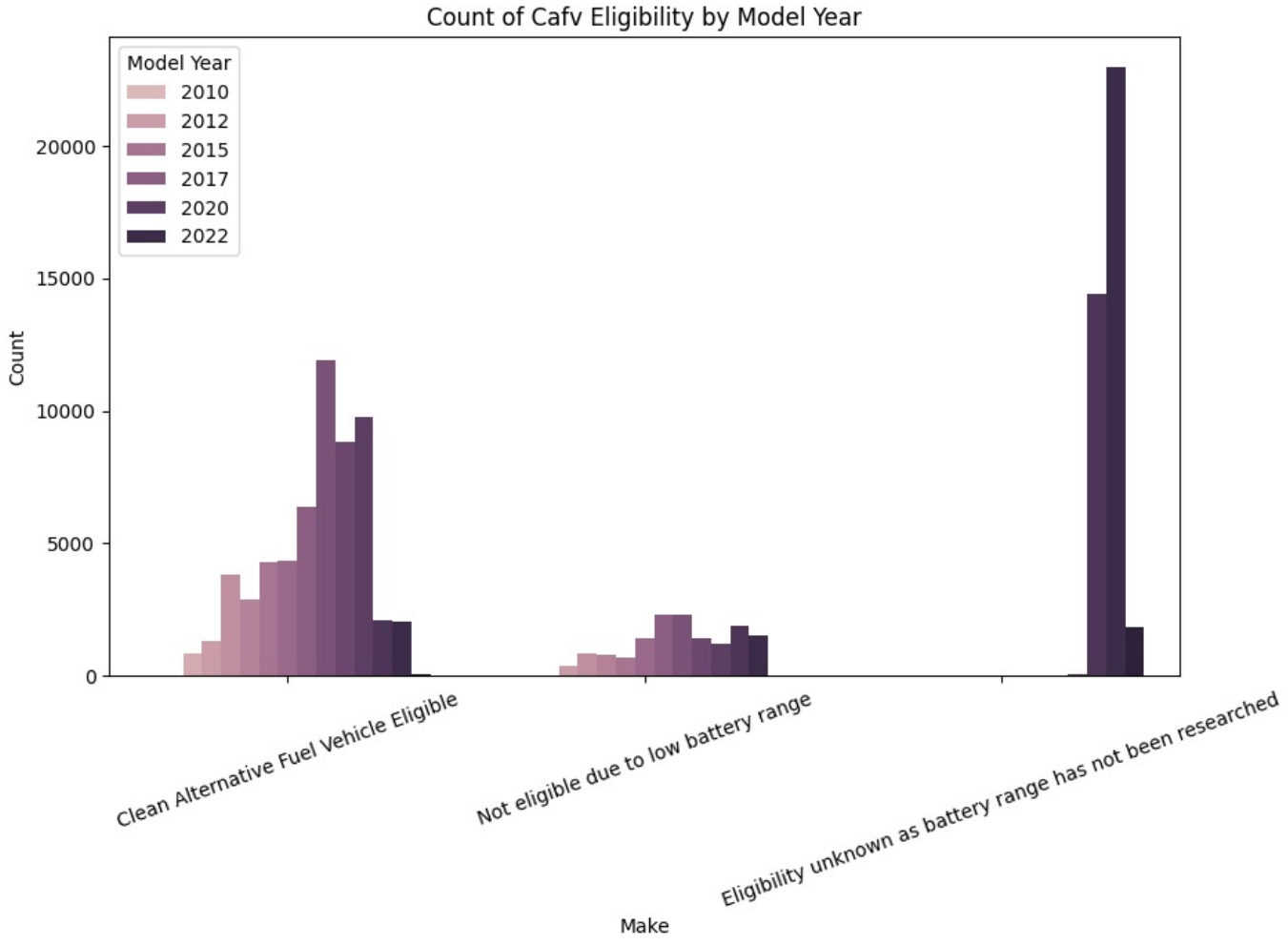
Out[37]: <Axes: xlabel='Electric Vehicle Type', ylabel='Electric Range'>



In [38]:
```
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Electric Vehicle Type', hue='Model Year')
plt.title('Count of Makes by Model Year')
plt.xlabel('Make')
plt.ylabel('Count')
plt.legend(title='Model Year')
plt.show()
```



In [39]:
```
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Cafv Eligibility', hue='Model Year')
plt.title('Count of Cafv Eligibility by Model Year')
plt.xlabel('Make')
plt.ylabel('Count')
plt.legend(title='Model Year')
plt.xticks(rotation=20)
plt.show()
```



Plotly

In [40]:
```
! pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\kajal\anaconda3\envs\kajalds\lib\site-packages (5.24.1)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\kajal\anaconda3\envs\kajalds\lib\site-packages (from
plotly) (9.0.0)
Requirement already satisfied: packaging in c:\users\kajal\anaconda3\envs\kajalds\lib\site-packages (from plotly
) (23.1)
```

In [41]:
```
import plotly.express as px
```

In [99]:
```
ev_count_by_state = df.groupby('State').size().reset_index(name='EV_Count')
```

In [97]:
```
fig = px.choropleth(
    ev_count_by_state,
    locations='State',
    locationmode='USA-states',
    color='EV_Count',
    scope='usa',
    labels={'EV_Count': 'Number of EVs'},
    title='Number of Electric Vehicles by State'
)

fig.update_layout(
    title_x=0.5,
)

fig.show()
```

In [103…
```
ev_count_by_state = df.groupby(['State', 'Model Year']).size().reset_index(name='EV_Count')
fig = px.choropleth(
    ev_count_by_state,
    locations='State',
    locationmode='USA-states',
    color='EV_Count',
    scope='usa',
    labels={'EV_Count': 'Number of EVs'},
    title='Number of Electric Vehicles by State and year',
    animation_frame = "Model Year"
)

fig.update_layout(
    title_x=0.5,
)

fig.show()
```

In [1]:
```
!pip install bar-chart-race
```

```
Requirement already satisfied: bar-chart-race in c:\users\kajal\anaconda3\envs\kajalds\lib\site-packages (0.1.0)
Requirement already satisfied: pandas>=0.24 in c:\users\kajal\anaconda3\envs\kajalds\lib\site-packages (from bar
-chart-race) (2.2.1)
Requirement already satisfied: matplotlib>=3.1 in c:\users\kajal\anaconda3\envs\kajalds\lib\site-packages (from
bar-chart-race) (3.9.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\kajal\anaconda3\envs\kajalds\lib\site-packages (from
matplotlib>=3.1->bar-chart-race) (1.2.1)
Requirement already satisfied: cycler>=0.10 in c:\users\kajal\anaconda3\envs\kajalds\lib\site-packages (from mat
plotlib>=3.1->bar-chart-race) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\kajal\anaconda3\envs\kajalds\lib\site-packages (fro
m matplotlib>=3.1->bar-chart-race) (4.53.1)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\kajal\anaconda3\envs\kajalds\lib\site-packages (fro
m matplotlib>=3.1->bar-chart-race) (1.4.5)
Requirement already satisfied: numpy>=1.23 in c:\users\kajal\anaconda3\envs\kajalds\lib\site-packages (from matp
lotlib>=3.1->bar-chart-race) (1.26.4)
Requirement already satisfied: packaging>=20.0 in c:\users\kajal\anaconda3\envs\kajalds\lib\site-packages (from
matplotlib>=3.1->bar-chart-race) (23.1)
Requirement already satisfied: pillow>=8 in c:\users\kajal\anaconda3\envs\kajalds\lib\site-packages (from matplo
tlib>=3.1->bar-chart-race) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\kajal\anaconda3\envs\kajalds\lib\site-packages (from
matplotlib>=3.1->bar-chart-race) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\kajal\anaconda3\envs\kajalds\lib\site-packages (
from matplotlib>=3.1->bar-chart-race) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\kajal\anaconda3\envs\kajalds\lib\site-packages (from pan
das>=0.24->bar-chart-race) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\kajal\anaconda3\envs\kajalds\lib\site-packages (from p
andas>=0.24->bar-chart-race) (2024.1)
Requirement already satisfied: six>=1.5 in c:\users\kajal\anaconda3\envs\kajalds\lib\site-packages (from python-
dateutil>=2.7->matplotlib>=3.1->bar-chart-race) (1.16.0)
```

In [46]:
```
import bar_chart_race as bcr
```

In [98]:
```
ev_counts = df.groupby(['Model Year', 'Make']).size().unstack(fill_value=0)
```

In [88]:
```
bcr.bar_chart_race(
    df=ev_count_by_state,
    title='EV Make and its Count Each Year',
    orientation='h',
    sort='desc',
    n_bars=10,
    steps_per_period=40,
    period_length=2000,
    bar_size=0.95,
    title_size=24,
    period_label={'x': .95, 'y': .25, 'fontsize': 12},
    perpendicular_bar_func='median',
    period_summary_func=lambda v, r: {'x': .2, 'y': .9, 's': f'Total EVs: {v.sum():,.0f}', 'ha': 'center', 'siz
    figsize=(6, 4),
    dpi=144,
```

```
    cmap='tab20'
)
```

Out[88]: Your browser does not support the video tag.

In [ ]:

In [ ]:

In [ ]: