ev-eda-3-1

October 10, 2024

0.1 Amaravathi Sai Ashritha

0.2 ID : IN9240302

[81]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[83]:
```python
df = pd.read_csv("dataset.csv")
```

[85]:
```python
df.head()
```

[85]:
```
   VIN (1-10)     County      City State  Postal Code  Model Year       Make  \
0  JTMEB3FV6N     Monroe  Key West    FL        33040        2022     TOYOTA
1  1G1RD6E45D      Clark  Laughlin    NV        89029        2013  CHEVROLET
2  JN1AZ0CP8B     Yakima    Yakima    WA        98901        2011     NISSAN
3  1G1FW6S08H     Skagit  Concrete    WA        98237        2017  CHEVROLET
4  3FA6P0SU1K  Snohomish   Everett    WA        98201        2019       FORD

        Model                   Electric Vehicle Type  \
0  RAV4 PRIME  Plug-in Hybrid Electric Vehicle (PHEV)
1        VOLT  Plug-in Hybrid Electric Vehicle (PHEV)
2        LEAF          Battery Electric Vehicle (BEV)
3     BOLT EV          Battery Electric Vehicle (BEV)
4      FUSION  Plug-in Hybrid Electric Vehicle (PHEV)

  Clean Alternative Fuel Vehicle (CAFV) Eligibility  Electric Range  \
0           Clean Alternative Fuel Vehicle Eligible              42
1           Clean Alternative Fuel Vehicle Eligible              38
2           Clean Alternative Fuel Vehicle Eligible              73
3           Clean Alternative Fuel Vehicle Eligible             238
4             Not eligible due to low battery range              26

   Base MSRP  Legislative District  DOL Vehicle ID  \
0          0                   NaN       198968248
1          0                   NaN         5204412
2          0                  15.0       218972519
```

```
3          0                 39.0       186750406
4          0                 38.0         2006714

             Vehicle Location        Electric Utility  2020 Census Tract
0    POINT (-81.80023 24.5545)                    NaN        12087972100
1  POINT (-114.57245 35.16815)                    NaN        32003005702
2  POINT (-120.50721 46.60448)             PACIFICORP        53077001602
3   POINT (-121.7515 48.53892)  PUGET SOUND ENERGY INC        53057951101
4  POINT (-122.20596 47.97659)  PUGET SOUND ENERGY INC        53061041500
```

[87]:
```
df.describe()
```

[87]:
```
         Postal Code     Model Year  Electric Range      Base MSRP  \
count  112634.000000  112634.000000   112634.000000  112634.000000
mean    98156.226850    2019.003365       87.812987    1793.439681
std      2648.733064       2.892364      102.334216   10783.753486
min      1730.000000    1997.000000        0.000000       0.000000
25%     98052.000000    2017.000000        0.000000       0.000000
50%     98119.000000    2020.000000       32.000000       0.000000
75%     98370.000000    2022.000000      208.000000       0.000000
max     99701.000000    2023.000000      337.000000  845000.000000

       Legislative District  DOL Vehicle ID  2020 Census Tract
count         112348.000000    1.126340e+05       1.126340e+05
mean              29.805604    1.994567e+08       5.296650e+10
std               14.700545    9.398427e+07       1.699104e+09
min                1.000000    4.777000e+03       1.101001e+09
25%               18.000000    1.484142e+08       5.303301e+10
50%               34.000000    1.923896e+08       5.303303e+10
75%               43.000000    2.191899e+08       5.305307e+10
max               49.000000    4.792548e+08       5.603300e+10
```

[89]:
```
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
```

[91]:
```
df.shape
```

[91]:
```
(112634, 17)
```

[93]:
```
df.columns
```

[93]:
```
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')
```

[95]:
```
df.columns = df.columns.str.replace(' ', '_')
df.columns
```

[95]:
```
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')
```

[97]:
```
df.rename(columns={'Clean_Alternative_Fuel_Vehicle_(CAFV)_Eligibility':
 ↪'CAFV_Eligibility'}, inplace=True)
df.columns
```

[97]:
```
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')
```

[99]:
```
print(df.isnull().sum())
```

```
VIN_(1-10)                 0
```

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

[101]:
```python
df_dropna = df.dropna()

df_dropna.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 112152 entries, 2 to 112633
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  int64
 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   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(6), object(10)
memory usage: 15.4+ MB
```

```
[ ]:
```

0.3 Task - 1

0.3.1 Non-Visual Univariate Analysis

```
[106]: numerical_columns = ['Postal_Code', 'Model_Year', 'Electric_Range',␣
        ↪'Base_MSRP', 'Legislative_District', 'DOL_Vehicle_ID', '2020_Census_Tract']


       categorical_columns = ['VIN_(1-10)', 'County', 'City', 'State', 'Make',␣
        ↪'Model', 'Electric_Vehicle_Type', 'CAFV_Eligibility', 'Vehicle_Location',␣
        ↪'Electric_Utility']

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

```
[108]: def discrete_univariate_analysis(discrete_data):
           for col_name in discrete_data:

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

```
[110]: discrete_univariate_analysis(discrete_df)
```

```
---------- VIN_(1-10) ----------
count                                                  112634
nunique                                                  7548
unique     [JTMEB3FV6N, 1G1RD6E45D, JN1AZ0CP8B, 1G1FW6S08…
Name: VIN_(1-10), dtype: object
Value Counts:
 VIN_(1-10)
5YJYGDEE9M    472
5YJYGDEE0M    465
5YJYGDEE8M    448
5YJYGDEE7M    448
5YJYGDEE2M    437
             …
WA1LAAGE9M      1
5UXKT0C50H      1
5YJYGAED3M      1
WDC0G5DBXL      1
YV4ED3GM0P      1
Name: count, Length: 7548, dtype: int64
```

```
---------- County ----------
count                                                   112634
nunique                                                    165
unique     [Monroe, Clark, Yakima, Skagit, Snohomish, Isl…
Name: County, dtype: object
Value Counts:
 County
King         59000
Snohomish    12434
Pierce        8535
Clark         6689
Thurston      4126
             …
Pinal            1
Elmore           1
Portsmouth       1
Kings            1
Kootenai         1
Name: count, Length: 165, dtype: int64

---------- City ----------
count                                                   112634
nunique                                                    629
unique     [Key West, Laughlin, Yakima, Concrete, Everett…
Name: City, dtype: object
Value Counts:
 City
Seattle        20305
Bellevue        5921
Redmond         4201
Vancouver       4013
Kirkland        3598
               …
Hartline           1
Gaithersburg       1
El Paso            1
Klickitat          1
Worley             1
Name: count, Length: 629, dtype: int64

---------- State ----------
count                                                   112634
nunique                                                     45
unique     [FL, NV, WA, IL, NY, VA, OK, KS, CA, NE, MD, C…
Name: State, dtype: object
Value Counts:
 State
WA    112348
```

```
CA         76
VA         36
MD         26
TX         14
CO          9
NV          8
GA          7
NC          7
CT          6
DC          6
FL          6
AZ          6
IL          6
SC          5
OR          5
NE          5
HI          4
UT          4
AR          4
NY          4
TN          3
KS          3
MO          3
PA          3
MA          3
LA          3
NJ          3
NH          2
OH          2
WY          2
ID          2
KY          1
RI          1
ME          1
MN          1
SD          1
WI          1
NM          1
AK          1
MS          1
AL          1
DE          1
OK          1
ND          1
Name: count, dtype: int64

---------- Make ----------
count                                              112634
```

```
nunique                                                     34
unique     [TOYOTA, CHEVROLET, NISSAN, FORD, TESLA, KIA, …
Name: Make, dtype: object
Value Counts:
 Make
TESLA             52078
NISSAN            12880
CHEVROLET         10182
FORD               5819
BMW                4680
KIA                4483
TOYOTA             4405
VOLKSWAGEN         2514
AUDI               2332
VOLVO              2288
CHRYSLER           1794
HYUNDAI            1412
JEEP               1152
RIVIAN              885
FIAT                822
PORSCHE             818
HONDA               792
MINI                632
MITSUBISHI          588
POLESTAR            558
MERCEDES-BENZ       506
SMART               273
JAGUAR              219
LINCOLN             168
CADILLAC            108
LUCID MOTORS         65
SUBARU               59
LAND ROVER           38
LEXUS                33
FISKER               20
GENESIS              18
AZURE DYNAMICS        7
TH!NK                 3
BENTLEY               3
Name: count, dtype: int64

---------- Model ----------
count                                                   112614
nunique                                                    114
unique     [RAV4 PRIME, VOLT, LEAF, BOLT EV, FUSION, MODE…
Name: Model, dtype: object
Value Counts:
 Model
```

```
MODEL 3        23135
MODEL Y        17142
LEAF           12880
MODEL S         7377
BOLT EV         4910
               …
745LE              2
S-10 PICKUP        1
SOLTERRA           1
918                1
FLYING SPUR        1
Name: count, Length: 114, dtype: int64

---------- Electric_Vehicle_Type ----------
count                                               112634
nunique                                                  2
unique     [Plug-in Hybrid Electric Vehicle (PHEV), Batte…
Name: Electric_Vehicle_Type, dtype: object
Value Counts:
 Electric_Vehicle_Type
Battery Electric Vehicle (BEV)            86044
Plug-in Hybrid Electric Vehicle (PHEV)    26590
Name: count, dtype: int64

---------- CAFV_Eligibility ----------
count                                               112634
nunique                                                  3
unique     [Clean Alternative Fuel Vehicle Eligible, Not …
Name: CAFV_Eligibility, dtype: object
Value Counts:
 CAFV_Eligibility
Clean Alternative Fuel Vehicle Eligible                         58639
Eligibility unknown as battery range has not been researched    39236
Not eligible due to low battery range                           14759
Name: count, dtype: int64

---------- Vehicle_Location ----------
count                                               112610
nunique                                                758
unique     [POINT (-81.80023 24.5545), POINT (-114.57245 …
Name: Vehicle_Location, dtype: object
Value Counts:
 Vehicle_Location
POINT (-122.13158 47.67858)    2916
POINT (-122.2066 47.67887)     2059
POINT (-122.1872 47.61001)     2001
POINT (-122.31765 47.70013)    1880
POINT (-122.12096 47.55584)    1852
```

```
                                          …
POINT (-124.33152 48.05431)       1
POINT (-77.41203 39.41574)        1
POINT (-123.61022 46.35588)       1
POINT (-112.04165 40.68741)       1
POINT (-116.91895 47.40077)       1
Name: count, Length: 758, dtype: int64

---------- Electric_Utility ----------
count                                                      112191
nunique                                                        73
unique      [nan, PACIFICORP, PUGET SOUND ENERGY INC, PUD …
Name: Electric_Utility, dtype: object
Value Counts:
 Electric_Utility
PUGET SOUND ENERGY INC||CITY OF TACOMA - (WA)
40247
PUGET SOUND ENERGY INC
22172
CITY OF SEATTLE - (WA)|CITY OF TACOMA - (WA)
21447
BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF CLARK COUNTY - (WA)
6522
BONNEVILLE POWER ADMINISTRATION||CITY OF TACOMA - (WA)||PENINSULA LIGHT COMPANY
5053
            …
BONNEVILLE POWER ADMINISTRATION||PENINSULA LIGHT COMPANY
1
BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF ASOTIN COUNTY
1
CITY OF SEATTLE - (WA)
1
BONNEVILLE POWER ADMINISTRATION||NESPELEM VALLEY ELEC COOP, INC
1
BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF CLALLAM COUNTY|PUD NO 1 OF
JEFFERSON COUNTY        1
Name: count, Length: 73, dtype: int64
```

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

[ ]:
```python
numerical_univariate_analysis(numerical_df)
```

0.3.2 Visual Univariate Analysis on Numerical Columns

Frequency Distribution

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

     # Plot histograms for numerical columns

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

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

Outlier Detection

```
[ ]: # Box plots for numerical columns

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

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

```
[ ]: def describe_outliers(df, column):
         Q1 = df[column].quantile(0.25)
         Q3 = df[column].quantile(0.75)
         IQR = Q3 - Q1
         lower_bound = Q1 - 1.5 * IQR
         upper_bound = Q3 + 1.5 * IQR
         outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)]
         print(f"\
     Column: {column}")
         print(f"Number of outliers: {len(outliers)}")
         print(f"Percentage of outliers: {len(outliers) / len(df) * 100:.2f}%")
         print(f"Range of outliers: {outliers[column].min()} to {outliers[column].
      ↪max()}")
         print(f"Range of non-outliers: {df[(df[column] >= lower_bound) &␣
      ↪(df[column] <= upper_bound)][column].min()} to {df[(df[column] >=␣
      ↪lower_bound) & (df[column] <= upper_bound)][column].max()}")

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

0.3.3 Visual Univariate Analysis on Categorical Variables

[ ]:
```python
# Plot bar charts for categorical columns
plt.figure(figsize=(15, 10))
for i, column in enumerate(categorical_columns[:6], 1):  # Limiting to first 6␣
↪for clarity
    plt.subplot(3, 2, i)
    sns.countplot(y=df[column], order=df[column].value_counts().index[:10])
    plt.title(f'Top 10 {column}')
plt.tight_layout()
plt.show()
```

0.3.4 Bivariate Analysis

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

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

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

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

```
plt.tight_layout()
plt.show()
```

[ ]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
df = pd.read_csv("dataset.csv")
# 5. Correlation matrix for numerical variables
plt.figure(figsize=(10, 8))
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix of Numerical Features')
plt.show()

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

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

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

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

0.4 Task 2: Create a Choropleth using plotly.express to display the number of EV vehicles based on location

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

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

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

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

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

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

fig.show()

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

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

[ ]:
```python
import pandas as pd
import plotly.express as px

# Load the dataset
df = pd.read_csv('dataset.csv', encoding='ascii')

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

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

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

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

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

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

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

fig.show()

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

# Display some statistics
print("\
Total number of unique postal codes:", len(ev_count_by_postal))
print("Average number of EVs per postal code:", round(ev_count_by_postal['EV_Count'].mean(), 2))
print("Median number of EVs per postal code:", ev_count_by_postal['EV_Count'].median())
print("Maximum number of EVs in a single postal code:", ev_count_by_postal['EV_Count'].max())
```

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

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

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

[ ]:
```
# Convert 'Model Year' to string for grouping
df['Model Year'] = df['Model Year'].astype(str)

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

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

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

# Create the bar chart race animation and save it as a GIF
bcr.bar_chart_race(df=pivoted_data, filename='EV_racing_bar_plot.gif',
                   orientation='h', sort='desc', n_bars=10,
                   title='EV Make Count Over the Years',␣
 ↪filter_column_colors=True, period_length=1000)
```

[ ]:
```

```