Title : Electric Vehicle Data Analysis Assignment

Name : Mamta Bhagde

Date : 10-10-2025

Course: Data Analysis

Introduction : The dataset contains information on Battery Electric Vehicles (BEVs) and Plug-in Hybrid Electric Vehicles (PHEVs). The main aim of this study is to analyze electric vehicle (EV) , understand EV adoption patterns across different regions and vehicle types, and build predictive models to estimate the electric range of vehicles. By examining vehicle characteristics, pricing, and incentives, this analysis provides a comprehensive view of EV trends.

Section 1 :Data Cleaning Questions

1.How many missing values exist in the dataset, and in which columns?

```python
import pandas as pd
df = pd.read_csv(r"D:\DA\python\Electric_Vehicle_Population_Data (3).csv")

# Count missing values
missing_values = df.isnull().sum()
missing_report = pd.DataFrame({
    'Missing Values': missing_values})
# Print the missing values
print(missing_report)
```

```
                                                   Missing Values
VIN (1-10)                                                      0
County                                                         10
City                                                           10
State                                                           0
Postal Code                                                    10
Model Year                                                      0
Make                                                            0
Model                                                           0
Electric Vehicle Type                                           0
Clean Alternative Fuel Vehicle (CAFV) Eligibility               0
Electric Range                                                  3
Base MSRP                                                       3
```

```
Legislative District                                              628
DOL Vehicle ID                                                      0
Vehicle Location                                                   18
Electric Utility                                                   10
2020 Census Tract                                                  10
```

1. How should missing or zero values in the Base MSRP and Electric Range columns be handled ?

```
# Missing values = the cells have no data.
# Zero values = For Base MSRP or Electric Range, a value of zero
doesn’t make sense — cars don’t cost $0 and EVs can’t have 0 miles of
range

# Remove rows where Base MSRP or Electric Range is missing or zero
df_clean = df[(df['Base MSRP'] > 0) & (df['Electric Range'] > 0)]

# Replace zeros or NaN in Base MSRP
median_msrp = df['Base MSRP'][df['Base MSRP'] > 0].median()
df['Base MSRP'] = df['Base MSRP'].replace(0, median_msrp)
df['Base MSRP'] = df['Base MSRP'].fillna(median_msrp)

# Replace zeros or NaN in Electric Range
median_range = df['Electric Range'][df['Electric Range'] > 0].median()
df['Electric Range'] = df['Electric Range'].replace(0, median_range)
df['Electric Range'] = df['Electric Range'].fillna(median_range)

print(df['Base MSRP'].describe())
print(df['Electric Range'].describe())

count    261698.000000
mean      59866.481612
std        3005.824570
min       31950.000000
25%       59900.000000
50%       59900.000000
75%       59900.000000
max      845000.000000
Name: Base MSRP, dtype: float64
count    261698.000000
mean         75.198794
std          66.975490
min           1.000000
25%          53.000000
50%          53.000000
75%          53.000000
max         337.000000
Name: Electric Range, dtype: float64
```

1. Are there duplicate records in the dataset? If so, how should they be managed?

```
# Checking for duplicate rows
duplicates = df.duplicated()

# total duplicates
total_duplicates = duplicates.sum()
print("Total duplicate rows:", total_duplicates)

Total duplicate rows: 0

# Create a new column 'VIN_anon' with unique numeric IDs
df['VIN_anon'] = range(1, len(df)+1)

# Drop the original VIN if needed
df = df.drop(columns=['VIN (1-10)'])
df.head()

     County          City State  Postal Code  Model Year    Make
Model  \
0    Yakima        Yakima    WA      98902.0        2013  TOYOTA
PRIUS
1    Kitsap  Port Orchard    WA      98366.0        2025    FORD
ESCAPE
2    Kitsap      Kingston    WA      98346.0        2024   MAZDA
CX-90
3  Thurston       Olympia    WA      98501.0        2023   TESLA
MODEL Y
4  Thurston       Rainier    WA      98576.0        2019   TESLA
MODEL 3

                    Electric Vehicle Type  \
0  Plug-in Hybrid Electric Vehicle (PHEV)
1  Plug-in Hybrid Electric Vehicle (PHEV)
2  Plug-in Hybrid Electric Vehicle (PHEV)
3          Battery Electric Vehicle (BEV)
4          Battery Electric Vehicle (BEV)

   Clean Alternative Fuel Vehicle (CAFV) Eligibility  Electric
Range  \
0             Not eligible due to low battery range             6.0

1           Clean Alternative Fuel Vehicle Eligible            37.0

2             Not eligible due to low battery range            26.0

3  Eligibility unknown as battery range has not b...           53.0

4           Clean Alternative Fuel Vehicle Eligible           220.0


   Base MSRP  Legislative District  DOL Vehicle ID  \
0    59900.0                  15.0       165252538
```

```
1    59900.0                    26.0       278572521
2    59900.0                    23.0       275123642
3    59900.0                    35.0       249569323
4    59900.0                    20.0       283135107

              Vehicle Location        Electric Utility  2020 Census
Tract  \
0  POINT (-120.51904 46.59783)              PACIFICORP
5.307700e+10
1  POINT (-122.63847 47.54103)  PUGET SOUND ENERGY INC
5.303509e+10
2   POINT (-122.4977 47.79802)  PUGET SOUND ENERGY INC
5.303509e+10
3  POINT (-122.89165 47.03954)  PUGET SOUND ENERGY INC
5.306701e+10
4  POINT (-122.68993 46.88897)  PUGET SOUND ENERGY INC
5.306701e+10

   VIN_anon
0         1
1         2
2         3
3         4
4         5

# Vehicle Location is stored as GPS coordinates (longitude, latitude).
They may have issues like: Missing values, Wrong formatting (e.g.,
strings instead of numbers), Hard to read in analysis or maps
```

1. How can VINs be anonymized while maintaining uniqueness?

```
import hashlib

df = pd.read_csv(r"D:\DA\python\Electric_Vehicle_Population_Data
(3).csv")

----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[3], line 1
----> 1 df = pd.read_csv(r"D:\DA\python\
Electric_Vehicle_Population_Data (3).csv")

NameError: name 'pd' is not defined

import pandas as pd
import hashlib

df = pd.read_csv(r"D:\DA\python\Electric_Vehicle_Population_Data
(3).csv")
```

```
df.head()

   VIN (1-10)    County          City State  Postal Code  Model Year
Make  \
0  JTDKN3DP2D    Yakima        Yakima    WA      98902.0        2013
TOYOTA
1  1FMCU0E1XS    Kitsap  Port Orchard    WA      98366.0        2025
FORD
2  JM3KKBHA9R    Kitsap      Kingston    WA      98346.0        2024
MAZDA
3  7SAYGDEE8P  Thurston       Olympia    WA      98501.0        2023
TESLA
4  5YJ3E1EB5K  Thurston       Rainier    WA      98576.0        2019
TESLA

     Model                   Electric Vehicle Type  \
0    PRIUS  Plug-in Hybrid Electric Vehicle (PHEV)
1   ESCAPE  Plug-in Hybrid Electric Vehicle (PHEV)
2    CX-90  Plug-in Hybrid Electric Vehicle (PHEV)
3  MODEL Y          Battery Electric Vehicle (BEV)
4  MODEL 3          Battery Electric Vehicle (BEV)

   Clean Alternative Fuel Vehicle (CAFV) Eligibility  Electric
Range  \
0              Not eligible due to low battery range             6.0

1            Clean Alternative Fuel Vehicle Eligible            37.0

2              Not eligible due to low battery range            26.0

3  Eligibility unknown as battery range has not b...             0.0

4            Clean Alternative Fuel Vehicle Eligible           220.0


   Base MSRP  Legislative District  DOL Vehicle ID  \
0        0.0                  15.0       165252538
1        0.0                  26.0       278572521
2        0.0                  23.0       275123642
3        0.0                  35.0       249569323
4        0.0                  20.0       283135107

              Vehicle Location       Electric Utility  2020 Census
Tract
0  POINT (-120.51904 46.59783)             PACIFICORP
5.307700e+10
1  POINT (-122.63847 47.54103)  PUGET SOUND ENERGY INC
5.303509e+10
2   POINT (-122.4977 47.79802)  PUGET SOUND ENERGY INC
5.303509e+10
```

```
3  POINT (-122.89165 47.03954)  PUGET SOUND ENERGY INC
5.306701e+10
4  POINT (-122.68993 46.88897)  PUGET SOUND ENERGY INC
5.306701e+10
```

```python
# Function to anonymize VIN
def anonymize_vin(vin):
    return hashlib.sha256(vin.encode()).hexdigest()

# Apply to your VIN column (replace 'VIN' with your actual column name
if different)
df['Anon_VIN'] = df['VIN (1-10)'].apply(anonymize_vin)
df.head()
```

```
   VIN (1-10)    County          City State  Postal Code  Model Year
Make  \
0  JTDKN3DP2D    Yakima        Yakima    WA      98902.0        2013
TOYOTA
1  1FMCU0E1XS    Kitsap  Port Orchard    WA      98366.0        2025
FORD
2  JM3KKBHA9R    Kitsap      Kingston    WA      98346.0        2024
MAZDA
3  7SAYGDEE8P  Thurston       Olympia    WA      98501.0        2023
TESLA
4  5YJ3E1EB5K  Thurston       Rainier    WA      98576.0        2019
TESLA

     Model                   Electric Vehicle Type  \
0    PRIUS  Plug-in Hybrid Electric Vehicle (PHEV)
1   ESCAPE  Plug-in Hybrid Electric Vehicle (PHEV)
2    CX-90  Plug-in Hybrid Electric Vehicle (PHEV)
3  MODEL Y          Battery Electric Vehicle (BEV)
4  MODEL 3          Battery Electric Vehicle (BEV)

   Clean Alternative Fuel Vehicle (CAFV) Eligibility  Electric
Range  \
0              Not eligible due to low battery range              6.0

1            Clean Alternative Fuel Vehicle Eligible             37.0

2              Not eligible due to low battery range             26.0

3  Eligibility unknown as battery range has not b...              0.0

4            Clean Alternative Fuel Vehicle Eligible            220.0


   Base MSRP  Legislative District  DOL Vehicle ID  \
0        0.0                  15.0       165252538
1        0.0                  26.0       278572521
2        0.0                  23.0       275123642
```

```
3       0.0                  35.0       249569323
4       0.0                  20.0       283135107

              Vehicle Location       Electric Utility  2020 Census
Tract  \
0  POINT (-120.51904 46.59783)             PACIFICORP
5.307700e+10
1  POINT (-122.63847 47.54103)  PUGET SOUND ENERGY INC
5.303509e+10
2   POINT (-122.4977 47.79802)  PUGET SOUND ENERGY INC
5.303509e+10
3  POINT (-122.89165 47.03954)  PUGET SOUND ENERGY INC
5.306701e+10
4  POINT (-122.68993 46.88897)  PUGET SOUND ENERGY INC
5.306701e+10

                                            Anon_VIN
0  bf01895762f04150a8ff5b0210e4d1c199986b50f45bcb...
1  a720d326091898dfa57b91cfa7466fe461b99a14f6bc78...
2  ef506f78a5a27e7e7582fd6924e13b4bfbac05984680a2...
3  fb3f4d8c8632615cdf99cc78f0f8e21e1e97d1e30d6dd0...
4  5fb1eb0d5a655b4eada221a1fa28fa1c5d7fbc960c0a97...

# See how the column looks
print(df['Vehicle Location'].head(20))

0     POINT (-120.51904 46.59783)
1     POINT (-122.63847 47.54103)
2      POINT (-122.4977 47.79802)
3     POINT (-122.89165 47.03954)
4     POINT (-122.68993 46.88897)
5      POINT (-122.1389 47.87115)
6     POINT (-122.70348 47.52028)
7     POINT (-122.37265 48.24159)
8     POINT (-122.30866 47.57874)
9     POINT (-122.92333 47.03779)
10    POINT (-122.69275 47.65171)
11     POINT (-122.2066 47.67887)
12     POINT (-122.90787 46.9461)
13    POINT (-122.06402 48.01497)
14    POINT (-122.35029 47.71871)
15    POINT (-122.16335 47.53505)
16    POINT (-122.24369 47.75892)
17    POINT (-122.20563 47.76144)
18    POINT (-122.68993 46.88897)
19    POINT (-122.18637 47.89251)
Name: Vehicle Location, dtype: object

# 5.How can Vehicle Location (GPS coordinates) be cleaned or converted
for better readability?
```

```python
import pandas as pd

# Load CSV
df = pd.read_csv(r"D:\DA\python\Electric_Vehicle_Population_Data
(3).csv")

# Get first 5 rows
df.head()
```

```
   VIN (1-10)    County          City State  Postal Code  Model Year
Make  \
0  JTDKN3DP2D    Yakima        Yakima    WA      98902.0        2013
TOYOTA
1  1FMCU0E1XS    Kitsap  Port Orchard    WA      98366.0        2025
FORD
2  JM3KKBHA9R    Kitsap      Kingston    WA      98346.0        2024
MAZDA
3  7SAYGDEE8P  Thurston       Olympia    WA      98501.0        2023
TESLA
4  5YJ3E1EB5K  Thurston       Rainier    WA      98576.0        2019
TESLA

     Model                   Electric Vehicle Type  \
0    PRIUS  Plug-in Hybrid Electric Vehicle (PHEV)
1   ESCAPE  Plug-in Hybrid Electric Vehicle (PHEV)
2    CX-90  Plug-in Hybrid Electric Vehicle (PHEV)
3  MODEL Y          Battery Electric Vehicle (BEV)
4  MODEL 3          Battery Electric Vehicle (BEV)

   Clean Alternative Fuel Vehicle (CAFV) Eligibility  Electric
Range  \
0              Not eligible due to low battery range              6.0

1            Clean Alternative Fuel Vehicle Eligible             37.0

2              Not eligible due to low battery range             26.0

3  Eligibility unknown as battery range has not b...              0.0

4            Clean Alternative Fuel Vehicle Eligible            220.0


   Base MSRP  Legislative District  DOL Vehicle ID  \
0        0.0                  15.0       165252538
1        0.0                  26.0       278572521
2        0.0                  23.0       275123642
3        0.0                  35.0       249569323
4        0.0                  20.0       283135107

              Vehicle Location        Electric Utility  2020 Census
Tract
```

```
0  POINT (-120.51904 46.59783)              PACIFICORP
5.307700e+10
1  POINT (-122.63847 47.54103)  PUGET SOUND ENERGY INC
5.303509e+10
2   POINT (-122.4977 47.79802)  PUGET SOUND ENERGY INC
5.303509e+10
3  POINT (-122.89165 47.03954)  PUGET SOUND ENERGY INC
5.306701e+10
4  POINT (-122.68993 46.88897)  PUGET SOUND ENERGY INC
5.306701e+10
```

```
# Remove invalid entries
df['Vehicle Location'] = df['Vehicle
Location'].astype(str).str.strip()
df = df[df['Vehicle Location'].notna()]             # Remove missing
values
df = df[df['Vehicle Location'] != '']             # Remove empty
strings
df = df[df['Vehicle Location'].str.lower() != 'nan']  # Remove string
'nan'

# Only keep rows containing a comma (expected format: "lat, lon")
df_valid = df[df['Vehicle Location'].str.contains(',',
na=False)].copy()

# If no valid rows exist, we stop here
if df_valid.empty:
    print("No valid GPS coordinates found.")
```

```
No valid GPS coordinates found.
```

```
if not df_valid.empty:
    # Split by comma
    split_coords = df_valid['Vehicle Location'].str.split(',', n=1,
expand=True)

    # Make sure split worked
    if split_coords.shape[1] >= 2:
        split_coords.columns = ['Latitude', 'Longitude']

        # Convert to float safely
        def to_float(x):
            try:
                return float(str(x).strip())
            except:
                return None

        split_coords['Latitude'] =
split_coords['Latitude'].apply(to_float)
        split_coords['Longitude'] =
split_coords['Longitude'].apply(to_float)
```

```
        # Keep only valid numeric rows
        split_coords = split_coords.dropna(subset=['Latitude',
'Longitude'])

        # Add back to the original DataFrame
        df_clean = df_valid.loc[split_coords.index].copy()
        df_clean['Latitude'] = split_coords['Latitude'].round(5)
        df_clean['Longitude'] = split_coords['Longitude'].round(5)

        print("Cleaned GPS coordinates:")
        print(df_clean[['Vehicle Location', 'Latitude',
'Longitude']].head())

import pandas as pd

# Load CSV
df = pd.read_csv(r"D:\DA\python\Electric_Vehicle_Population_Data
(3).csv")

# Clean Vehicle Location column
df['Vehicle Location'] = df['Vehicle
Location'].astype(str).str.strip()
df = df[df['Vehicle Location'].notna()]
df = df[df['Vehicle Location'] != '']
df = df[df['Vehicle Location'].str.lower() != 'nan']

# Keep only rows with a comma
df_valid = df[df['Vehicle Location'].str.contains(',',
na=False)].copy()

if df_valid.empty:
    print("No valid Vehicle Location entries. df_clean cannot be
created.")
else:
    split_coords = df_valid['Vehicle Location'].str.split(',', n=1,
expand=True)

    if split_coords.shape[1] >= 2:
        split_coords.columns = ['Latitude', 'Longitude']

        def to_float(x):
            try:
                return float(str(x).strip())
            except:
                return None

        split_coords['Latitude'] =
split_coords['Latitude'].apply(to_float)
        split_coords['Longitude'] =
```

```
split_coords['Longitude'].apply(to_float)
        split_coords = split_coords.dropna(subset=['Latitude',
'Longitude'])

        df_clean = df_valid.loc[split_coords.index].copy()
        df_clean['Latitude'] = split_coords['Latitude'].round(5)
        df_clean['Longitude'] = split_coords['Longitude'].round(5)

        # Now df_clean exists, and you can save it
        df_clean.to_csv(r"D:\DA\python\
Electric_Vehicle_Population_Clean.csv", index=False)
        print("Cleaned CSV saved successfully.")
    else:
        print("Split did not produce two columns. df_clean cannot be
created.")

No valid Vehicle Location entries. df_clean cannot be created.
```

Section 2 : Data Exploration Questions

A > What are the top 5 most common EV makes and models in the dataset?

```
import pandas as pd
df = pd.read_csv(r"D:\DA\python\Electric_Vehicle_Population_Data
(3).csv")

df.head()

   VIN (1-10)    County          City State  Postal Code  Model Year
Make  \
0  JTDKN3DP2D    Yakima        Yakima    WA      98902.0        2013
TOYOTA
1  1FMCU0E1XS    Kitsap  Port Orchard    WA      98366.0        2025
FORD
2  JM3KKBHA9R    Kitsap      Kingston    WA      98346.0        2024
MAZDA
3  7SAYGDEE8P  Thurston       Olympia    WA      98501.0        2023
TESLA
4  5YJ3E1EB5K  Thurston       Rainier    WA      98576.0        2019
TESLA

     Model                   Electric Vehicle Type  \
0    PRIUS  Plug-in Hybrid Electric Vehicle (PHEV)
1   ESCAPE  Plug-in Hybrid Electric Vehicle (PHEV)
2    CX-90  Plug-in Hybrid Electric Vehicle (PHEV)
3  MODEL Y          Battery Electric Vehicle (BEV)
4  MODEL 3          Battery Electric Vehicle (BEV)

   Clean Alternative Fuel Vehicle (CAFV) Eligibility  Electric
Range  \
```

```
0                 Not eligible due to low battery range               6.0

1               Clean Alternative Fuel Vehicle Eligible              37.0

2                 Not eligible due to low battery range              26.0

3  Eligibility unknown as battery range has not b...                 0.0

4               Clean Alternative Fuel Vehicle Eligible             220.0


   Base MSRP  Legislative District  DOL Vehicle ID  \
0        0.0                  15.0       165252538
1        0.0                  26.0       278572521
2        0.0                  23.0       275123642
3        0.0                  35.0       249569323
4        0.0                  20.0       283135107

              Vehicle Location        Electric Utility  2020 Census
Tract
0  POINT (-120.51904 46.59783)              PACIFICORP
5.307700e+10
1  POINT (-122.63847 47.54103)  PUGET SOUND ENERGY INC
5.303509e+10
2   POINT (-122.4977 47.79802)  PUGET SOUND ENERGY INC
5.303509e+10
3  POINT (-122.89165 47.03954)  PUGET SOUND ENERGY INC
5.306701e+10
4  POINT (-122.68993 46.88897)  PUGET SOUND ENERGY INC
5.306701e+10
```

```python
# Top 5 EV Makes
top_makes = df['Make'].value_counts().head(5)
print("Top 5 EV Makes:")
display(top_makes)
```

```
Top 5 EV Makes:

Make
TESLA        108777
CHEVROLET     18908
NISSAN        16224
FORD          13988
KIA           12849
Name: count, dtype: int64
```

```python
# Top 5 EV Models
top_models = df['Model'].value_counts().head(5)
print("Top 5 EV Models:")
display(top_models)
```

```
Top 5 EV Models:

Model
MODEL Y     54720
MODEL 3     37774
LEAF        13852
MODEL S      7945
BOLT EV      7873
Name: count, dtype: int64
```

B > What is the distribution of EVs by county? Which county has the most registrations?

```
df.columns

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')

# EVs are registered in each county
county_distribution = df['County'].value_counts()

print("Distribution of EVs by County:")
display(county_distribution)

Distribution of EVs by County:

County
King            130129
Snohomish        32335
Pierce           21624
Clark            15925
Thurston          9506
                 ...
Platte               1
Manatee              1
Escambia             1
Utah                 1
Denton               1
Name: count, Length: 236, dtype: int64

# top county
top_county = county_distribution.idxmax()
top_count = county_distribution.max()
print(f"\nCounty with the most EV registrations: {top_county}
({top_count} vehicles)")
```

```
County with the most EV registrations: King (130129 vehicles)
```

C> How has EV adoption changed over different model years?

```python
# EVs are registered for each model year
ev_by_year = df['Model Year'].value_counts().sort_index()

print("EV Adoption by Model Year:")
display(ev_by_year)
```

```
EV Adoption by Model Year:

Model Year
2000        8
2002        1
2003        1
2008       20
2010       22
2011      631
2012     1440
2013     4081
2014     3327
2015     4574
2016     5253
2017     8767
2018    14524
2019    11043
2020    12395
2021    20937
2022    29647
2023    60215
2024    49869
2025    29495
2026     5448
Name: count, dtype: int64
```

D > What is the average electric range of EVs in the dataset ?

```python
df.columns
```

```
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')
```

```
# Average Electric Range
average_range = df['Electric Range'].mean()
print("Average Electric Range of EVs:", round(average_range, 2),
"miles")
```

```
Average Electric Range of EVs: 42.62 miles
```

E > What percentage of EVs are eligible for Clean Alternative Fuel Vehicle (CAFV) incentives?

```
total_vehicles = len(df)
eligible_vehicles = df[df['Clean Alternative Fuel Vehicle (CAFV)
Eligibility'] == 'Clean Alternative Fuel Vehicle Eligible'].shape[0]

# Calculate percentage
percentage_eligible = (eligible_vehicles / total_vehicles) * 100
print(f"Percentage of EVs eligible for CAFV incentives:
{percentage_eligible:.2f}%")
```

```
Percentage of EVs eligible for CAFV incentives: 29.35%
```

F > How does the electric range vary across different makes and models?

```
# Average Electric Range by Make
avg_range_by_make = df.groupby('Make')['Electric
Range'].mean().sort_values(ascending=False)
print("Average Electric Range by Make:")
display(avg_range_by_make)
```

```
Average Electric Range by Make:

Make
JAGUAR                    181.267606
WHEEGO ELECTRIC CARS      100.000000
TH!NK                     100.000000
CHEVROLET                  82.355458
FIAT                       75.490588
NISSAN                     65.670118
SMART                      61.686441
AZURE DYNAMICS             56.000000
TESLA                      55.961150
PORSCHE                    49.788640
LAND ROVER                 48.523316
AUDI                       38.013809
ALFA ROMEO                 33.000000
POLESTAR                   32.812458
MITSUBISHI                 32.203347
CHRYSLER                   32.167143
BENTLEY                    31.250000
KIA                        30.250370
DODGE                      29.045514
```

```
BMW                              28.391618
TOYOTA                           27.806731
MAZDA                            25.559971
LINCOLN                          24.631579
LAMBORGHINI                      22.909091
LEXUS                            22.112295
JEEP                             21.890893
VOLVO                            17.828012
VOLKSWAGEN                       16.729367
HONDA                            15.838747
MERCEDES-BENZ                    15.323832
MINI                             14.038972
HYUNDAI                          13.090533
FORD                              7.579139
FISKER                            2.275862
CADILLAC                          1.909250
SUBARU                            0.628639
BRIGHTDROP                        0.000000
ACURA                             0.000000
LUCID                             0.000000
GMC                               0.000000
GENESIS                           0.000000
MULLEN AUTOMOTIVE INC.            0.000000
ROLLS-ROYCE                       0.000000
RIVIAN                            0.000000
RAM                               0.000000
VINFAST                           0.000000
Name: Electric Range, dtype: float64
```

```
# Average Electric Range by Make & Model
avg_range_by_model = df.groupby(['Make', 'Model'])['Electric
Range'].mean().sort_values(ascending=False)
print("Average Electric Range by Make and Model:")
# show top 10 models
display(avg_range_by_model.head(10))
```

```
Average Electric Range by Make and Model:
```

```
Make        Model
TESLA       ROADSTER    230.000000
JAGUAR      I-PACE      181.267606
CHEVROLET   BOLT EV     168.491680
TESLA       MODEL S     165.849339
AUDI        E-TRON      128.594881
TESLA       MODEL X     118.113199
VOLKSWAGEN  E-GOLF      107.096408
PORSCHE     MACAN       105.070632
TOYOTA      RAV4        102.728814
TH!NK       CITY        100.000000
Name: Electric Range, dtype: float64
```

G > What is the average Base MSRP for each EV model?

```
# Average Base MSRP for Each Model
# Group by Make and Model, then calculate average MSRP
avg_msrp_by_model = df.groupby(['Make', 'Model'])['Base
MSRP'].mean().sort_values(ascending=False)

print("Average Base MSRP for each EV model:")
display(avg_msrp_by_model)
```

```
Average Base MSRP for each EV model:

Make      Model
PORSCHE   918          845000.000000
TESLA     ROADSTER     103563.541667
FISKER    KARMA        102000.000000
BMW       740E          90287.037037
CADILLAC  CT6           75095.000000
                            ...
VOLVO     EX90              0.000000
          V60               0.000000
          S90               0.000000
          S60               0.000000
          XC40              0.000000
Name: Base MSRP, Length: 181, dtype: float64
```

```
avg_msrp_by_model.head(10)
```

```
Make                     Model
PORSCHE                  918           845000.000000
TESLA                    ROADSTER      103563.541667
FISKER                   KARMA         102000.000000
BMW                      740E           90287.037037
CADILLAC                 CT6            75095.000000
BMW                      530E           35430.091533
WHEEGO ELECTRIC CARS     WHEEGO         32995.000000
KIA                      SOUL           30868.695652
SUBARU                   CROSSTREK      24570.957447
MINI                     COUNTRYMAN     15601.259446
Name: Base MSRP, dtype: float64
```

H > Are there any regional trends in EV adoption (e.g., urban vs. rural areas)?

```
# By following steps , can analyze this directly from your EV dataset.
# EVs by County or City
ev_by_region = df['County'].value_counts()
print("EV Registrations by County:")
display(ev_by_region.head(10))
```

```
EV Registrations by County:
```

```
County
King           130129
Snohomish       32335
Pierce          21624
Clark           15925
Thurston         9506
Kitsap           8787
Spokane          7370
Whatcom          6406
Benton           3572
Skagit           3067
Name: count, dtype: int64

# Categorize as Urban or Rural
urban_counties = ['King', 'Snohomish', 'Pierce', 'Clark', 'Thurston']
df['Region Type'] = df['County'].apply(lambda x: 'Urban' if x in
urban_counties else 'Rural')

region_summary = df['Region Type'].value_counts(normalize=True) * 100
print("EV Adoption by Region Type:")
display(region_summary)

EV Adoption by Region Type:

Region Type
Urban    80.061368
Rural    19.938632
Name: proportion, dtype: float64

# This means most EVs are registered in urban areas
```

Section 3 : Data Visualization Questions

A > Create a bar chart showing the top 5 EV makes and models by count.

```
# Top 5 EV makes by count
top5_makes = df['Make'].value_counts().head(5)
# Display as a table
print("Top 5 EV Makes by Count:")
display(top5_makes)

Top 5 EV Makes by Count:

Make
TESLA          108777
CHEVROLET       18908
NISSAN          16224
FORD            13988
KIA             12849
Name: count, dtype: int64
```

```python
# Top 5 EV Models.
top5_models = df['Model'].value_counts().head(5)

print("Top 5 EV Models by Count:")
display(top5_models)
```

```
Top 5 EV Models by Count:

Model
MODEL Y    54720
MODEL 3    37774
LEAF       13852
MODEL S     7945
BOLT EV     7873
Name: count, dtype: int64
```

B > Use a heatmap or choropleth map to visualize EV distribution by county.

```python
!pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\hp\appdata\local\
programs\python\python312\lib\site-packages (6.3.1)
Requirement already satisfied: narwhals>=1.15.1 in c:\users\hp\
appdata\local\programs\python\python312\lib\site-packages (from
plotly) (2.7.0)
Requirement already satisfied: packaging in c:\users\hp\appdata\local\
programs\python\python312\lib\site-packages (from plotly) (25.0)
```

```python
# Aggregate EV counts by county
ev_by_county = df['County'].value_counts().reset_index()
ev_by_county.columns = ['County', 'EV_Count']
ev_by_county.head()
```

```
      County  EV_Count
0       King    130129
1  Snohomish     32335
2     Pierce     21624
3      Clark     15925
4   Thurston      9506
```

```python
!pip install matplotlib
```

```
Collecting matplotlib
  Downloading matplotlib-3.10.6-cp312-cp312-win_amd64.whl.metadata (11
kB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.3.3-cp312-cp312-win_amd64.whl.metadata (5.5
kB)
Collecting cycler>=0.10 (from matplotlib)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
```

```
  Downloading fonttools-4.60.1-cp312-cp312-win_amd64.whl.metadata (114
kB)
Collecting kiwisolver>=1.3.1 (from matplotlib)
  Downloading kiwisolver-1.4.9-cp312-cp312-win_amd64.whl.metadata (6.4
kB)
Requirement already satisfied: numpy>=1.23 in c:\users\hp\appdata\
local\programs\python\python312\lib\site-packages (from matplotlib)
(2.3.3)
Requirement already satisfied: packaging>=20.0 in c:\users\hp\appdata\
local\programs\python\python312\lib\site-packages (from matplotlib)
(25.0)
Collecting pillow>=8 (from matplotlib)
  Downloading pillow-11.3.0-cp312-cp312-win_amd64.whl.metadata (9.2
kB)
Collecting pyparsing>=2.3.1 (from matplotlib)
  Downloading pyparsing-3.2.5-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\hp\
appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\hp\appdata\local\
programs\python\python312\lib\site-packages (from python-
dateutil>=2.7->matplotlib) (1.17.0)
Downloading matplotlib-3.10.6-cp312-cp312-win_amd64.whl (8.1 MB)
   ---------------------------------------- 0.0/8.1 MB ? eta -:--:--
   --- ------------------------------------ 0.8/8.1 MB 5.6 MB/s eta
0:00:02
   ---------- ----------------------------- 2.1/8.1 MB 5.9 MB/s eta
0:00:02
   ------------------ --------------------- 3.7/8.1 MB 6.6 MB/s eta
0:00:01
   ----------------------- ---------------- 4.7/8.1 MB 6.8 MB/s eta
0:00:01
   ----------------------------- ---------- 6.0/8.1 MB 6.3 MB/s eta
0:00:01
   ------------------------------------- -- 7.6/8.1 MB 6.3 MB/s eta
0:00:01
   ---------------------------------------- 8.1/8.1 MB 6.2 MB/s
0:00:01
Downloading contourpy-1.3.3-cp312-cp312-win_amd64.whl (226 kB)
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.60.1-cp312-cp312-win_amd64.whl (2.3 MB)
   ---------------------------------------- 0.0/2.3 MB ? eta -:--:--
   ------------- -------------------------- 0.8/2.3 MB 3.0 MB/s eta
0:00:01
   ------------------------------------- -- 2.1/2.3 MB 4.5 MB/s eta
0:00:01
   ---------------------------------------- 2.3/2.3 MB 4.4 MB/s
0:00:00
Downloading kiwisolver-1.4.9-cp312-cp312-win_amd64.whl (73 kB)
```

```
Downloading pillow-11.3.0-cp312-cp312-win_amd64.whl (7.0 MB)
   ---------------------------------------- 0.0/7.0 MB ? eta -:--:--
   --- ------------------------------------ 0.5/7.0 MB 4.2 MB/s eta
0:00:02
   --------- ------------------------------ 1.6/7.0 MB 4.2 MB/s eta
0:00:02
   --------------- ------------------------ 2.6/7.0 MB 4.6 MB/s eta
0:00:01
   ------------------ --------------------- 3.1/7.0 MB 4.6 MB/s eta
0:00:01
   ------------------------ --------------- 4.2/7.0 MB 4.3 MB/s eta
0:00:01
   ------------------------------ --------- 5.2/7.0 MB 4.4 MB/s eta
0:00:01
   ------------------------------------ --- 6.3/7.0 MB 4.5 MB/s eta
0:00:01
   ---------------------------------------- 7.0/7.0 MB 4.4 MB/s
0:00:01
Downloading pyparsing-3.2.5-py3-none-any.whl (113 kB)
Installing collected packages: pyparsing, pillow, kiwisolver,
fonttools, cycler, contourpy, matplotlib

   ---------------------------------------- 0/7 [pyparsing]
   ---------------------------------------- 0/7 [pyparsing]
   ---------------------------------------- 0/7 [pyparsing]
   ----- ---------------------------------- 1/7 [pillow]
   ----- ---------------------------------- 1/7 [pillow]
   ----- ---------------------------------- 1/7 [pillow]
   ----- ---------------------------------- 1/7 [pillow]
   ----- ---------------------------------- 1/7 [pillow]
   ----- ---------------------------------- 1/7 [pillow]
   ----- ---------------------------------- 1/7 [pillow]
   ----- ---------------------------------- 1/7 [pillow]
   ----- ---------------------------------- 1/7 [pillow]
   ----- ---------------------------------- 1/7 [pillow]
   ----- ---------------------------------- 1/7 [pillow]
   ----- ---------------------------------- 1/7 [pillow]
   ----- ---------------------------------- 1/7 [pillow]
   ----- ---------------------------------- 1/7 [pillow]
   ----- ---------------------------------- 1/7 [pillow]
   ----- ---------------------------------- 1/7 [pillow]
   ----- ---------------------------------- 1/7 [pillow]
   ----- ---------------------------------- 1/7 [pillow]
   ----------- ---------------------------- 2/7 [kiwisolver]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
```

```
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
```

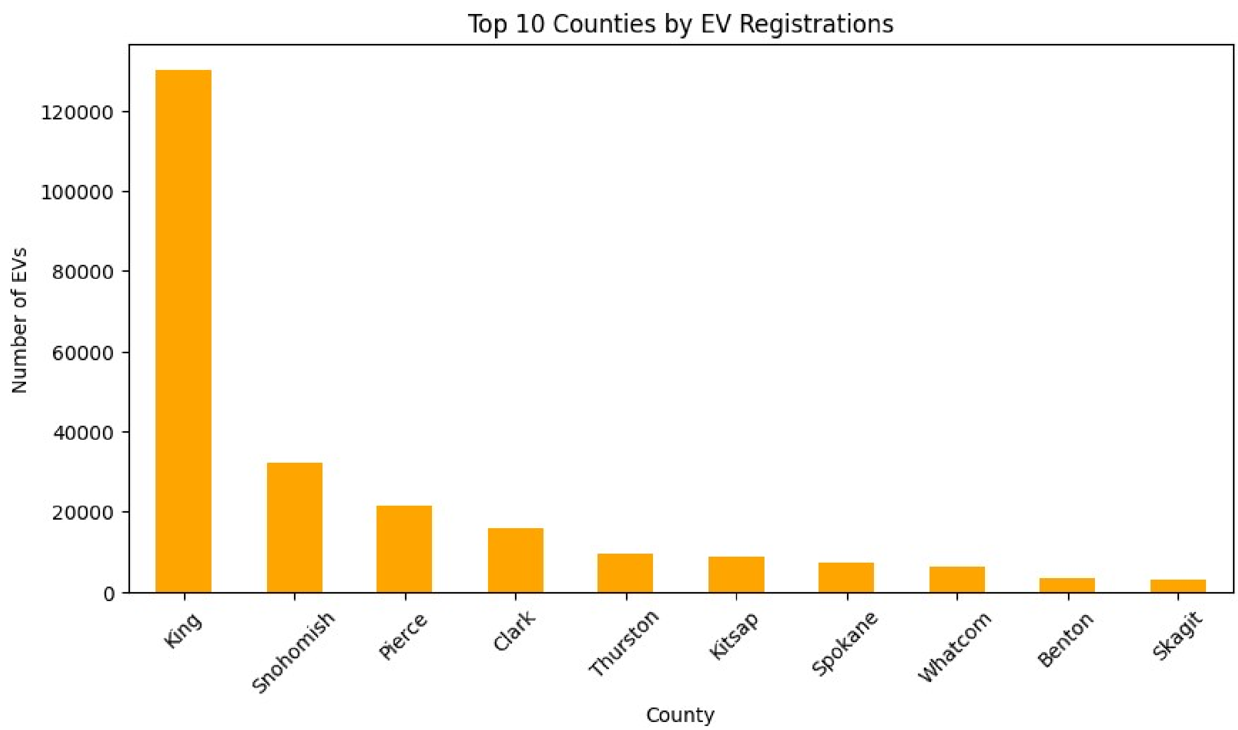
```
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ----------------- ---------------------- 3/7 [fonttools]
   ---------------------------- ----------- 5/7 [contourpy]
   ---------------------------- ----------- 5/7 [contourpy]
   ---------------------------- ----------- 5/7 [contourpy]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
```

```
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------- ----- 6/7 [matplotlib]
   ---------------------------------------- 7/7 [matplotlib]

Successfully installed contourpy-1.3.3 cycler-0.12.1 fonttools-4.60.1
kiwisolver-1.4.9 matplotlib-3.10.6 pillow-11.3.0 pyparsing-3.2.5
```

```python
import pandas as pd
import matplotlib.pyplot as plt

# Aggregate EV counts by county
ev_by_county = df['County'].value_counts().reset_index()
ev_by_county.columns = ['County', 'EV_Count']
```

```
# Plot top 10 counties
top_counties = ev_by_county.head(10)

top_counties.plot(kind='bar', x='County', y='EV_Count',
figsize=(10,5), color='orange', legend=False)
plt.title("Top 10 Counties by EV Registrations")
plt.xlabel("County")
plt.ylabel("Number of EVs")
plt.xticks(rotation=45)
plt.show()
```

```
Matplotlib is building the font cache; this may take a moment.
```

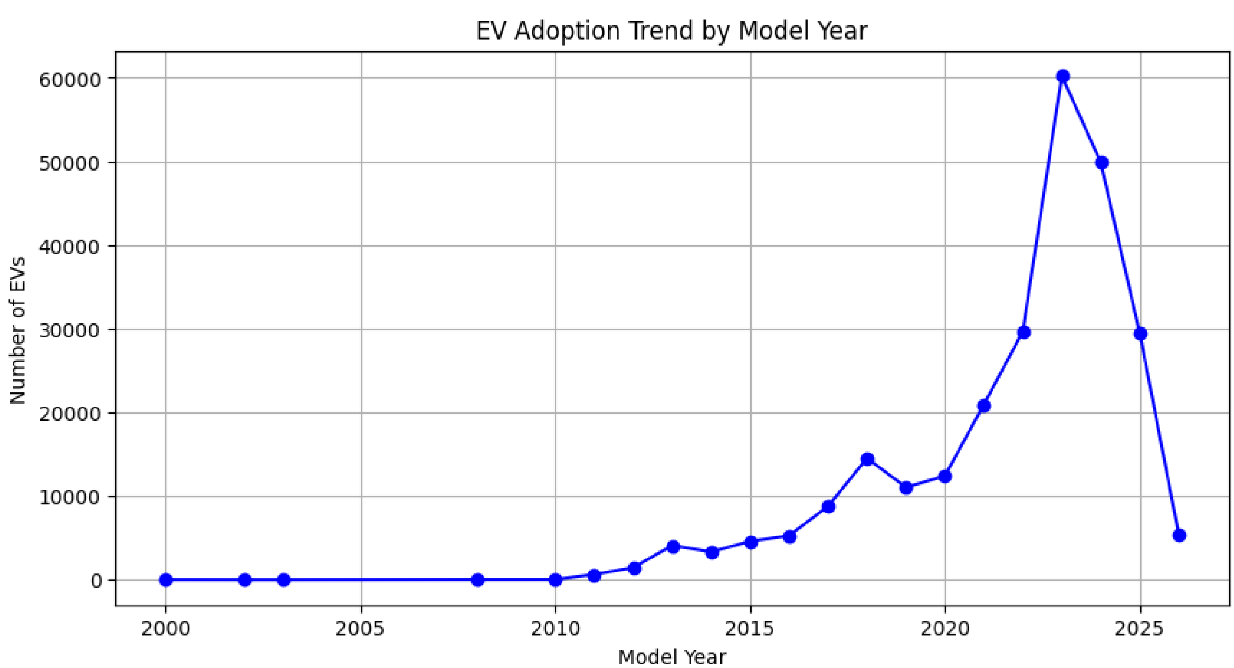


C > Create a line graph showing the trend of EV adoption by model year.

```
# EVs by Model Year
ev_by_year = df['Model Year'].value_counts().sort_index()

print(ev_by_year)
```

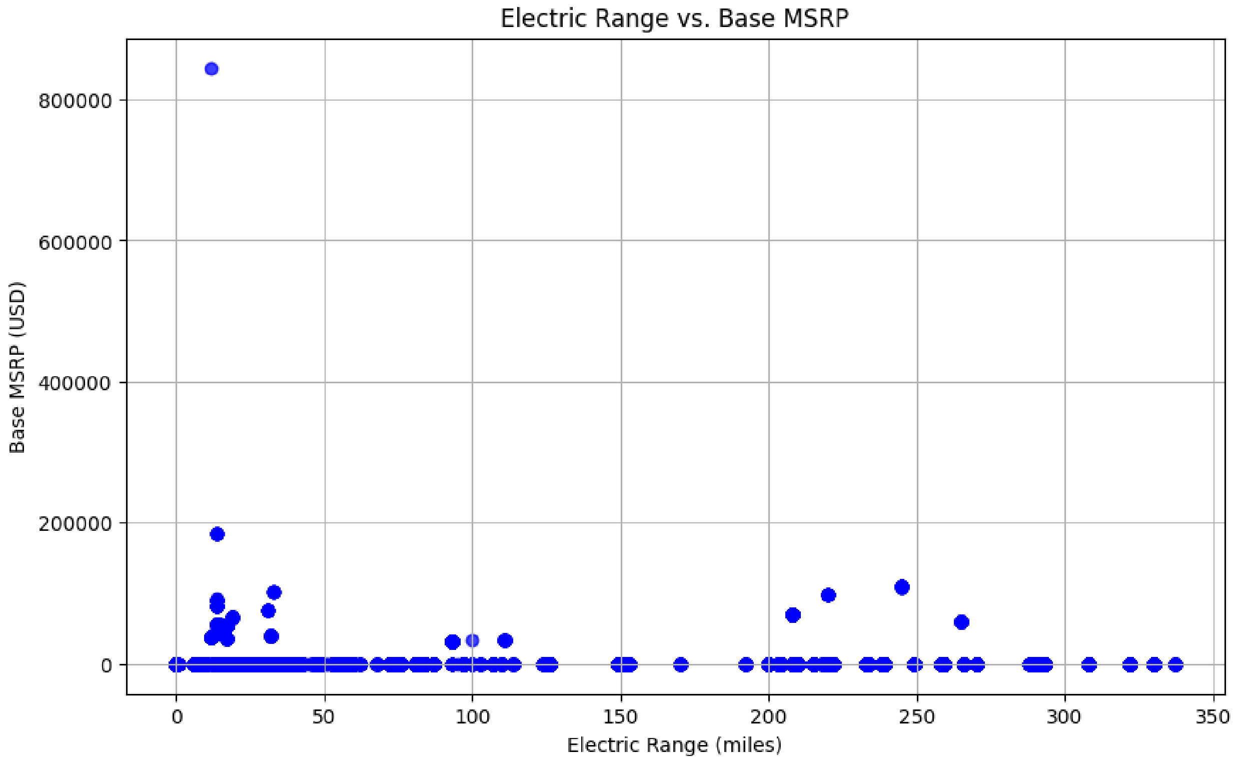
```
Model Year
2000       8
2002       1
2003       1
2008      20
2010      22
2011     631
```

```
2012      1440
2013      4081
2014      3327
2015      4574
2016      5253
2017      8767
2018     14524
2019     11043
2020     12395
2021     20937
2022     29647
2023     60215
2024     49869
2025     29495
2026      5448
Name: count, dtype: int64

# Creating the line graph
plt.figure(figsize=(10,5))
plt.plot(ev_by_year.index, ev_by_year.values, marker='o',
color='blue')
plt.title("EV Adoption Trend by Model Year")
plt.xlabel("Model Year")
plt.ylabel("Number of EVs")
plt.grid(True)
plt.show()
```



D > Generate a scatter plot comparing electric range vs. base MSRP to see pricing trends

```
# Create a scatter plot
plt.figure(figsize=(10,6))
plt.scatter(df['Electric Range'], df['Base MSRP'], alpha=0.5,
color='blue')
plt.title("Electric Range vs. Base MSRP")
plt.xlabel("Electric Range (miles)")
plt.ylabel("Base MSRP (USD)")
plt.grid(True)
plt.show()
```



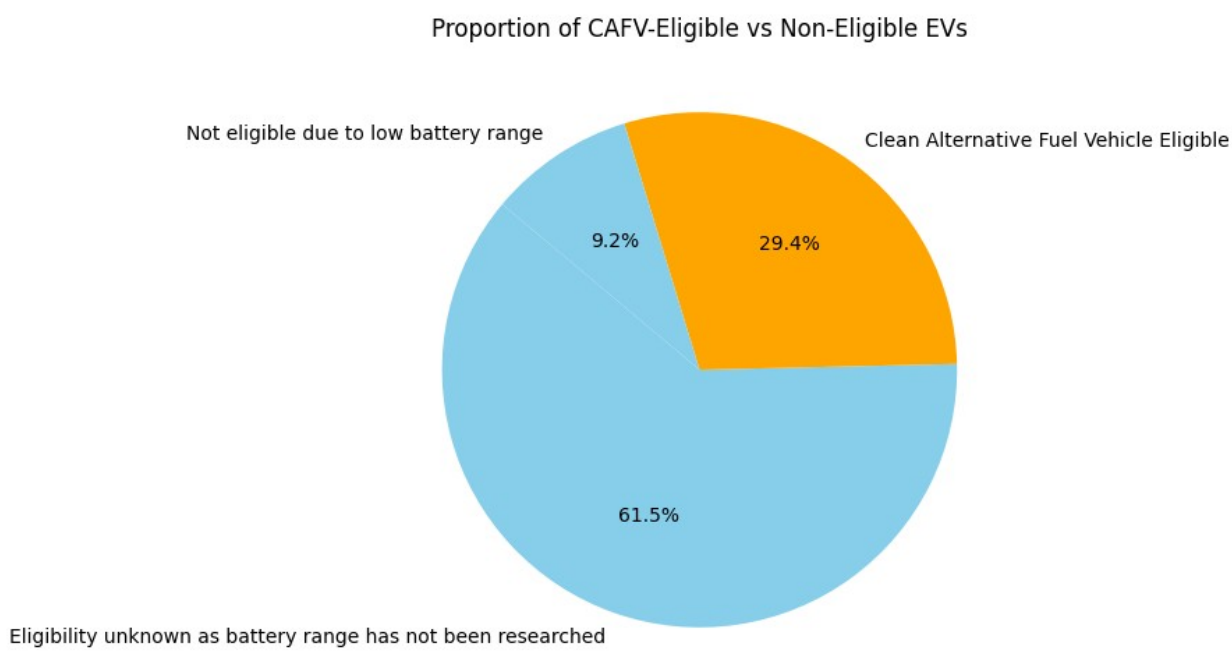
E > Plot a pie chart showing the proportion of CAFV-eligible vs. non-eligible EVs.

```
# Count CAFV eligibility
cafv_counts = df['Clean Alternative Fuel Vehicle (CAFV)
Eligibility'].value_counts()
cafv_counts
```

```
Clean Alternative Fuel Vehicle (CAFV) Eligibility
Eligibility unknown as battery range has not been researched    160888
Clean Alternative Fuel Vehicle Eligible                          76819
Not eligible due to low battery range                            23991
Name: count, dtype: int64
```

```
# Plot pie chart
plt.figure(figsize=(6,6))
plt.pie(cafv_counts, labels=cafv_counts.index, autopct='%1.1f%%',
```

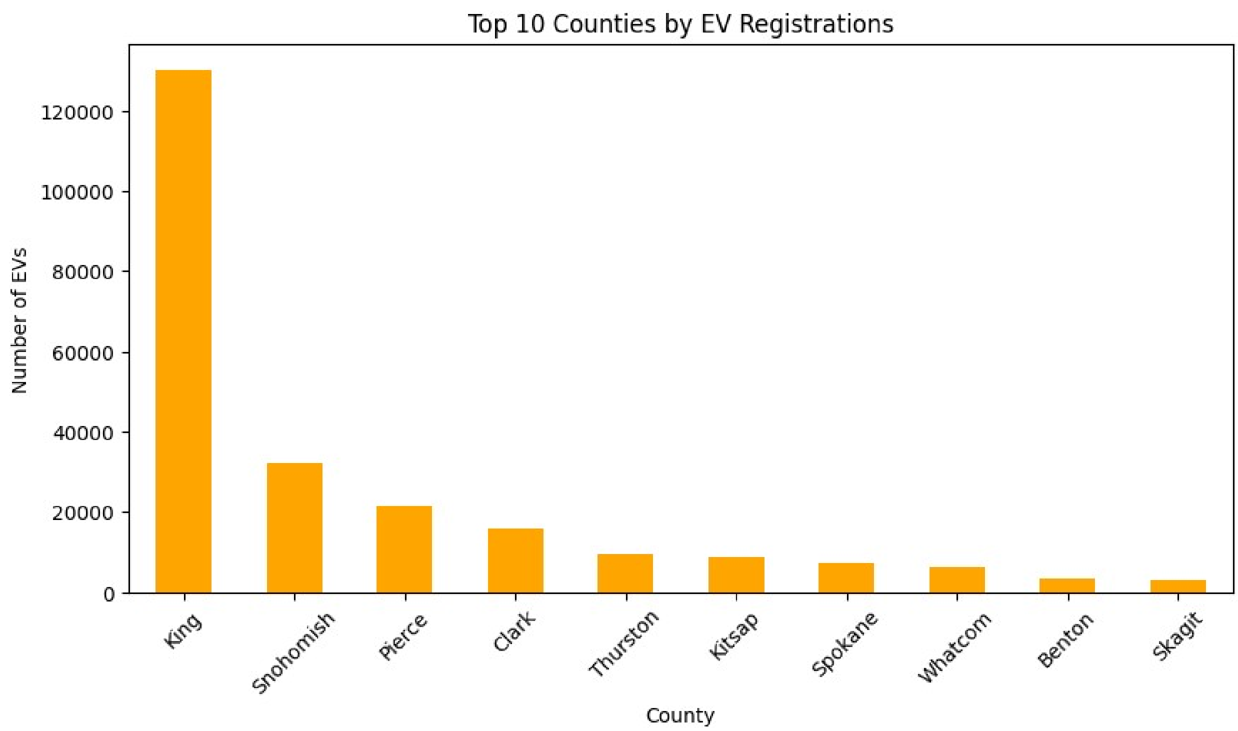
```
startangle=140, colors=['skyblue', 'orange'])
plt.title("Proportion of CAFV-Eligible vs Non-Eligible EVs")
plt.show()
```



F > Use a geospatial map to display EV registrations based on vehicle location.

```
df.columns

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',
       'Region Type'],
      dtype='object')

# Count EVs by county
ev_by_county = df['County'].value_counts().reset_index()
ev_by_county.columns = ['County', 'EV_Count']

# Display top 10 counties
ev_by_county.head(10)

      County  EV_Count
0       King    130129
1  Snohomish     32335
```

```
2      Pierce     21624
3       Clark     15925
4   Thurston       9506
5     Kitsap       8787
6    Spokane       7370
7    Whatcom       6406
8     Benton       3572
9     Skagit       3067

# Visualize with a bar chart
import matplotlib.pyplot as plt

top_counties = ev_by_county.head(10)
top_counties.plot(kind='bar', x='County', y='EV_Count',
figsize=(10,5), color='orange', legend=False)
plt.title("Top 10 Counties by EV Registrations")
plt.xlabel("County")
plt.ylabel("Number of EVs")
plt.xticks(rotation=45)
plt.show()
```




```
# data set not having latitude and longitude columns
```

Section 4 : Linear Regression Model Questions

A > How can we use Linear Regression to predict the Electric Range of a vehicle?

Ans : Linear Regression predicts a continuous target variable (here, Electric Range) based on one or more features (independent variables) like: Base MSRP (price of the vehicle), Battery Capacity, Vehicle Weight, Motor Power, Model Year

```
!pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\hp\appdata\
local\programs\python\python312\lib\site-packages (1.7.2)
Requirement already satisfied: numpy>=1.22.0 in c:\users\hp\appdata\
local\programs\python\python312\lib\site-packages (from scikit-learn)
(2.3.3)
Requirement already satisfied: scipy>=1.8.0 in c:\users\hp\appdata\
local\programs\python\python312\lib\site-packages (from scikit-learn)
(1.16.2)
Requirement already satisfied: joblib>=1.2.0 in c:\users\hp\appdata\
local\programs\python\python312\lib\site-packages (from scikit-learn)
(1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\hp\
appdata\local\programs\python\python312\lib\site-packages (from
scikit-learn) (3.6.0)
```

```
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

print("scikit-learn version:", sklearn.__version__)
```

```
scikit-learn version: 1.7.2
```

```
df.columns
```

```
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')
```

```
# numeric columns in your dataset
print(df.select_dtypes(include=['int64', 'float64']).columns)
```

```
Index(['Postal Code', 'Model Year', 'Electric Range', 'Base MSRP',
       'Legislative District', 'DOL Vehicle ID', '2020 Census Tract'],
      dtype='object')
```

```
# Convert 'Make' and 'Model' to numeric features
df_encoded = pd.get_dummies(df, columns=['Make', 'Model'],
drop_first=True)
```

```python
# Now df_encoded has numeric columns instead of text for Make and
Model
print(df_encoded.head())
```

```
   VIN (1-10)    County          City State  Postal Code  Model
Year  \
0  JTDKN3DP2D    Yakima        Yakima    WA      98902.0        2013

1  1FMCU0E1XS    Kitsap  Port Orchard    WA      98366.0        2025

2  JM3KKBHA9R    Kitsap      Kingston    WA      98346.0        2024

3  7SAYGDEE8P  Thurston       Olympia    WA      98501.0        2023

4  5YJ3E1EB5K  Thurston       Rainier    WA      98576.0        2019


                   Electric Vehicle Type  \
0  Plug-in Hybrid Electric Vehicle (PHEV)
1  Plug-in Hybrid Electric Vehicle (PHEV)
2  Plug-in Hybrid Electric Vehicle (PHEV)
3          Battery Electric Vehicle (BEV)
4          Battery Electric Vehicle (BEV)

   Clean Alternative Fuel Vehicle (CAFV) Eligibility  Electric
Range  \
0              Not eligible due to low battery range            6.0

1            Clean Alternative Fuel Vehicle Eligible           37.0

2              Not eligible due to low battery range           26.0

3  Eligibility unknown as battery range has not b...            0.0

4            Clean Alternative Fuel Vehicle Eligible          220.0


   Base MSRP  ...  Model_WHEEGO  Model_WRANGLER Model_X3 Model_X5
Model_XC40  \
0        0.0  ...         False           False    False    False
False
1        0.0  ...         False           False    False    False
False
2        0.0  ...         False           False    False    False
False
3        0.0  ...         False           False    False    False
False
4        0.0  ...         False           False    False    False
False

   Model_XC60  Model_XC90  Model_XM  Model_ZDX  Model_ZEVO
```

```
0       False       False     False      False       False
1       False       False     False      False       False
2       False       False     False      False       False
3       False       False     False      False       False
4       False       False     False      False       False

[5 rows x 240 columns]
```

```
# Numeric features
numeric_features = ['Base MSRP', 'Model Year']  # replace with numeric
columns you have

# Combine with encoded categorical features
X = pd.concat([df[numeric_features], df_encoded], axis=1)
y = df['Electric Range']

# Train-test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Automatically select numeric columns
numeric_features = df.select_dtypes(include=['int64',
'float64']).columns.tolist()

# Make sure target 'Electric Range' is excluded from features
if 'Electric Range' in numeric_features:
    numeric_features.remove('Electric Range')

X = df[numeric_features]  # only numeric features
y = df['Electric Range']

# Handle missing numeric values
X = X.fillna(X.mean())  # fill missing numeric values with mean
y = y.fillna(y.mean())

# Train-test split and Linear Regression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
# Evaluate Model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

```
Mean Squared Error: 4636.422413685628
R-squared: 0.3006988849618796
```

B > What independent variables (features) can be used to predict Electric Range? (e.g., Model Year, Base MSRP, Make)

Ans : For predicting the Electric Range of a vehicle, you should use features that influence how far a car can travel on a single charge. These features are called independent variables in regression analysis. Like Price & Market Variables : Base MSRP → Often correlates with battery size and advanced technology. Trim Level → Higher trims may have better motors or larger batteries. Make & Model : Make (Manufacturer) → Different manufacturers design vehicles with different efficiencies. Model → Specific model design affects energy usage. Technology & Year : Model Year → Newer models usually have better battery tech and efficiency. Technology Features → Regenerative braking, energy-saving modes, etc For Ex :

```
df.head()
```

```
   VIN (1-10)    County          City State  Postal Code  Model Year
Make  \
0  JTDKN3DP2D    Yakima        Yakima    WA      98902.0        2013
41
1  1FMCU0E1XS    Kitsap  Port Orchard    WA      98366.0        2025
13
2  JM3KKBHA9R    Kitsap      Kingston    WA      98346.0        2024
26
3  7SAYGDEE8P  Thurston       Olympia    WA      98501.0        2023
39
4  5YJ3E1EB5K  Thurston       Rainier    WA      98576.0        2019
39

   Model                           Electric Vehicle Type  \
0    118  Plug-in Hybrid Electric Vehicle (PHEV)
1     55  Plug-in Hybrid Electric Vehicle (PHEV)
2     39  Plug-in Hybrid Electric Vehicle (PHEV)
3    105          Battery Electric Vehicle (BEV)
4    102          Battery Electric Vehicle (BEV)

   Clean Alternative Fuel Vehicle (CAFV) Eligibility  Electric
Range  \
0             Not eligible due to low battery range             6.0

1           Clean Alternative Fuel Vehicle Eligible            37.0

2             Not eligible due to low battery range            26.0

3  Eligibility unknown as battery range has not b...            0.0
```

```
4             Clean Alternative Fuel Vehicle Eligible           220.0


   Base MSRP  Legislative District  DOL Vehicle ID  \
0        0.0                  15.0       165252538
1        0.0                  26.0       278572521
2        0.0                  23.0       275123642
3        0.0                  35.0       249569323
4        0.0                  20.0       283135107

              Vehicle Location        Electric Utility  2020 Census
Tract
0  POINT (-120.51904 46.59783)              PACIFICORP
5.307700e+10
1  POINT (-122.63847 47.54103)  PUGET SOUND ENERGY INC
5.303509e+10
2   POINT (-122.4977 47.79802)  PUGET SOUND ENERGY INC
5.303509e+10
3  POINT (-122.89165 47.03954)  PUGET SOUND ENERGY INC
5.306701e+10
4  POINT (-122.68993 46.88897)  PUGET SOUND ENERGY INC
5.306701e+10

# Features available in your dataset
features = ['Base MSRP', 'Model Year', 'Make', 'Model']

# Target variable
target = 'Electric Range'

# Create feature and target DataFrames
X = df[features]
y = df[target]

X = X.copy()
X['Base MSRP'] = X['Base MSRP'].fillna(X['Base MSRP'].mean())
X['Model Year'] = X['Model Year'].fillna(X['Model Year'].mean())
X[['Make','Model']] = X[['Make','Model']].fillna('Unknown')

# Encode Categorical Variables
X_encoded = pd.get_dummies(X, columns=['Make','Model'],
drop_first=True)
# Split Data
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y,
test_size=0.2, random_state=42)

# Train Linear Regression
model = LinearRegression()
model.fit(X_train, y_train)

LinearRegression()
```

```
# Evaluate Model
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R² Score:", r2)
```

```
Mean Squared Error: 3181.4734499626393
R² Score: 0.5201455491941502
```

```
# Interpret Base MSRP Influence
coeff_df = pd.DataFrame({'Feature': X_encoded.columns, 'Coefficient':
model.coef_})
print(coeff_df.sort_values(by='Coefficient',
ascending=False).head(10))
```

```
       Feature  Coefficient
55     Model_9   436.590933
101   Model_55   106.770988
69    Model_23   101.935617
34     Make_33   101.602689
50     Model_4    91.082590
145   Model_99    77.466139
108   Model_62    76.620076
180  Model_134    74.910094
89    Model_43    70.739262
207  Model_161    67.073938
```

C > How do we handle categorical variables like Make and Model in regression analysis?

Ans : In regression analysis, categorical variables like Make and Model cannot be used directly, because regression algorithms require numerical inputs. We need to convert these categorical variables into numbers

```
import pandas as pd

# One-hot encode 'Make' and 'Model'
X_encoded = pd.get_dummies(X, columns=['Make','Model'],
drop_first=True)
```

Use One-Hot Encoding for Make and Model.If there are hundreds of categories, consider keeping top N frequent categories and label others as "Other". For ex:

```
# Original features
features = ['Base MSRP', 'Model Year', 'Make', 'Model']
X = df[features]

# One-hot encode categorical variables
```

```
X_encoded = pd.get_dummies(X, columns=['Make','Model'],
drop_first=True)

# Now X_encoded can be used for Linear Regression

X = df[['Base MSRP', 'Model Year', 'Make', 'Model']]
y = df['Electric Range']

#Handle Missing Values
X = X.copy()
X['Base MSRP'] = X['Base MSRP'].fillna(X['Base MSRP'].mean())
X['Model Year'] = X['Model Year'].fillna(X['Model Year'].mean())
X[['Make','Model']] = X[['Make','Model']].fillna('Unknown')
y = y.fillna(y.mean())

# Convert Categorical Columns
X_encoded = pd.get_dummies(X, columns=['Make','Model'],
drop_first=True)
#Split Data
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y,
test_size=0.2, random_state=42)

# Train Model
model = LinearRegression()
model.fit(X_train, y_train)
# Evaluate Model
y_pred = model.predict(X_test)

print("R² Score:", r2_score(y_test, y_pred))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))

R² Score: 0.5201455491941502
Mean Squared Error: 3181.4734499626393

#Predict for a New EV
new_vehicle = pd.DataFrame({
    'Base MSRP':[55000],
    'Model Year':[2022],
    'Make':['TESLA'],
    'Model':['Model 3']
})

new_vehicle_encoded = pd.get_dummies(new_vehicle)
new_vehicle_encoded =
new_vehicle_encoded.reindex(columns=X_encoded.columns, fill_value=0)

predicted_range = model.predict(new_vehicle_encoded)
print("Predicted Electric Range:", predicted_range[0])

Predicted Electric Range: -19.010836518660653
```

D > What is the R^2 score of the model, and what does it indicate about prediction accuracy?

Ans : R^2 (R-squared) is a statistical measure of how well your regression model explains the variation in the target variable. It ranges from 0 to 1 (sometimes slightly negative if the model is very bad). $R^2 = 1$ → Perfect prediction (all points fit the model exactly) $R^2 = 0$ → Model does not explain any variation in the data R^2 tells us how much of the variation in Electric Range can be explained by our model using Base MSRP, Model Year, Make, and Model. For ex.

```
# Get R² Score in Python
from sklearn.metrics import r2_score

r2 = r2_score(y_test, y_pred)
print("R² Score:", r2)

R² Score: 0.5201455491941502
```

E > How does the Base MSRP influence the Electric Range according to the regression model?

In Linear Regression, each feature (like Base MSRP) gets a coefficient. The coefficient shows how much the target (Electric Range) changes when that feature changes — keeping others constant. So if the Base MSRP coefficient is positive, it means: As the Base MSRP (price) increases, the Electric Range tends to increase. If it's negative, then: As the Base MSRP increases, the Electric Range tends to decrease.

```
# Check it in Python
coeff_df = pd.DataFrame({'Feature': X_encoded.columns, 'Coefficient':
model.coef_})
coeff_df[coeff_df['Feature'] == 'Base MSRP']

     Feature  Coefficient
0  Base MSRP    -0.000804
```

For ex : If the coefficient of Base MSRP is -0.000804 , it means: For every 1 unit increase in Base MSRP, the Electric Range decreases slightly by about 0.000804 miles. Since the coefficient is negative, it shows a small negative relationship between vehicle price and range.

The coefficient of Base MSRP is -0.000804, which means there is a slight negative relationship between vehicle price and electric range. As the Base MSRP increases, the electric range decreases a little. This may be because higher-priced EVs often focus on premium features or performance rather than only extending range.

E > What steps are needed to improve the accuracy of the Linear Regression model?

1 > Add More Relevant Features Right now, the model only uses Base MSRP, Model Year, Make, and Model

2 > Remove or Handle Outliers Outliers (unusual values) can mislead the model. You can detect them using:

```
df.describe()
```

```
          Postal Code    Model Year           Make          Model  \
count  261688.000000  261698.000000  261698.000000  261698.000000
mean    98176.150699    2021.772493      28.929549      99.273812
std      2555.753410       3.034041      13.125556      36.812807
min      1469.000000    2000.000000       0.000000       0.000000
25%     98052.000000    2020.000000      17.000000      90.000000
50%     98133.000000    2023.000000      39.000000     103.000000
75%     98382.000000    2024.000000      39.000000     106.000000
max     99577.000000    2026.000000      45.000000     180.000000

       Electric Range      Base MSRP  Legislative District  DOL
Vehicle ID  \
count   261695.000000  261698.000000         261070.000000
2.616980e+05
mean        42.615071     695.503563             28.881955
2.412577e+08
std         81.226054    6942.979857             14.889697
6.574252e+07
min          0.000000       0.000000              1.000000
4.385000e+03
25%          0.000000       0.000000             17.000000
2.150419e+08
50%          0.000000       0.000000             32.000000
2.594588e+08
75%         35.000000       0.000000             42.000000
2.746481e+08
max        337.000000  845000.000000             49.000000
4.791150e+08

       2020 Census Tract
count       2.616880e+05
mean        5.297261e+10
std         1.628791e+09
min         1.001020e+09
25%         5.303301e+10
50%         5.303303e+10
75%         5.305307e+10
max         6.601095e+10
```

3 > Normalize / Scale Numeric Data Large differences in feature scales can confuse the model. Use:

```python
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_encoded)
```

This ensures all features have equal importance.

4 > Encode Categorical Data Properly Use One-Hot Encoding (as we did) to convert Make and Model correctly. If there are too many categories, group the least common ones into "Other"

5 > Remove Irrelevant or Correlated Features Too many similar features can reduce performance.

6 > Try Advanced Models Sometimes Linear Regression isn't enough

7 > Cross-Validation Instead of using a single train-test split, use:

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, X_encoded, y, cv=5)
print(scores.mean())

0.5232756907684915
```

F > Can we use this model to predict the range of new EV models based on their specifications?

Yes, We can use the Linear Regression model to predict the Electric Range of new EV models based on their specifications — like Base MSRP, Model Year, Make, and Model. For Ex :

```
# Example: Predict Electric Range for a new EV
new_vehicle = pd.DataFrame({
    'Base MSRP':[55000],
    'Model Year':[2022],
    'Make':['TESLA'],
    'Model':['Model 3']
})

# Convert to same format as training data
new_vehicle_encoded = pd.get_dummies(new_vehicle)
new_vehicle_encoded =
new_vehicle_encoded.reindex(columns=X_encoded.columns, fill_value=0)

# Predict
predicted_range = model.predict(new_vehicle_encoded)
print("Predicted Electric Range:", predicted_range[0], "miles")

Predicted Electric Range: -19.010836518660653 miles
```

Conclusion : This analysis of data helps us understand which electric vehicles are most popular, how EV adoption varies by region, and how incentives affect registrations. Cleaning the data made it easier to work with, and visualizations showed clear trends. Using a regression model, we found that factors like Base MSRP, Model Year, and Make can help predict a vehicle's electric range.