```
import numpy as np
import pandas as pd
import hashlib
df = pd.read_csv("Electric_Vehicle_Population_Data (1).csv")
df.info()
df.head()

# DATA CLEANING
# How many missing values exist in the dataset, and in which columns?
missing_values = df.isnull().sum()
print ("\n the total missing values in dataset", missing_values.sum())
print ("\n the total missing values in each column", missing_values)
#How should missing or zero values in the Base MSRP and Electric Range columns be handled?
 # replace zero values
df["Base MSRP"] = df["Base MSRP"].replace(0, np.nan)
df["Electric Range"] = df["Electric Range"].replace(0, np.nan)
 # replace missing values by median values
df["Base MSRP"] = df["Base MSRP"].fillna(df["Base MSRP"].median())
df["Electric Range"] = df["Electric Range"].fillna(df["Electric Range"].median())
print ("\n missing and zero values in base MSRP and Electric Range columns handled", df["Base MSRP"].isnull().sum(), d
#Are there duplicate records in the dataset? If so, how should they be managed?
duplicate_value = df.duplicated().sum()
print ("\n duplicate records in dataset", duplicate_value)
df.drop_duplicates(inplace=True)
#How can VINs be anonymized while maintaining uniqueness?
 # unique Id in this dataset is VIN[1-10]
df["VIN (1-10)"] = df["VIN (1-10)"].apply(lambda x: hashlib.sha256(x.encode()).hexdigest())
print ("\n maintaing uniqueness ", df["VIN (1-10)"])
#How can Vehicle Location (GPS coordinates) be cleaned or converted for better readability?
df["Latitude"] = df["Vehicle Location"].str.extract(r'POINT \((-?\d+\.\d+) (-?\d+\.\d+)\)')[1]
df["Longitude"] = df["Vehicle Location"].str.extract(r'POINT \((-?\d+\.\d+) (-?\d+\.\d+)\)')[0]
   # convert into neumeric
df["Latitude"] = pd.to_numeric(df["Latitude"])
df["Longitude"] = pd.to_numeric(df["Longitude"])
print ("\n cleaned or converted for better readability", df["Latitude"], df["Longitude"])
```

```
County                                              3
City                                                3
```

```
235689    55dfe19ac9780e08e476343bca6f00a135e5d2984e82c2...
235690    b473fe66e7749a8624292b8cbdceeec21229f650d206c4...
235691    084df2ada60f054781d8acd25a187f68e244c0bfa566df...
Name: VIN (1-10), Length: 235692, dtype: object

 cleaned or converted for better readability 0        47.49461
1         47.73689
2         47.42602
3         47.64509
4         46.88897
            ...
235687    47.29238
235688    48.24159
235689    47.67858
235690    48.01497
235691    47.53010
Name: Latitude, Length: 235692, dtype: float64 0        -122.23825
1        -122.64681
2        -122.54729
3        -122.81585
4        -122.68993
            ...
235687   -122.51134
235688   -122.37265
235689   -122.13158
235690   -122.06402
235691   -122.03439
Name: Longitude, Length: 235692, dtype: float64
```

```
import numpy as np
import pandas as pd
import hashlib
df = pd.read_csv("Electric_Vehicle_Population_Data (1).csv")
# Data EXPLORATION
# What are the top 5 most common EV makes and models in the dataset
top_5_makes = df["Make"].value_counts().head(5)
top_5_models = df["Model"].value_counts().head(5)
print ("\n top 5 makes ", top_5_makes)
print ("\n top 5 models ", top_5_models)
#What is the distribution of EVs by county? Which county has the most registrations?
ev_distribution_by_county = df["County"].value_counts()
most_registered_county = ev_distribution_by_county.idxmax()
print ("\n distribution of EVs by county ", ev_distribution_by_county)
print ("\n county with most registrations ", most_registered_county)
# How has EV adoption changed over different model years?
ev_adoptation_over_model_year = df["Model Year"].value_counts()
print ("\n adoptation changed by year", ev_adoptation_over_model_year)
# What is the average electric range of EVs in the dataset?
average_electric_range = df["Electric Range"].mean()
print ("\n average electric range ", average_electric_range)
# What percentage of EVs are eligible for Clean Alternative Fuel Vehicle (CAFV) incentives?
cafv_incentives_percentage = (df["Clean Alternative Fuel Vehicle (CAFV) Eligibility"] == "Clean Alternative Fuel Vehic
print ("\n percentage of EVs eligible for CAFV incentives ", cafv_incentives_percentage)
# How does the electric range vary across different makes and models?
electric_range_by_make = df.groupby('Make')['Electric Range'].mean()
print ("\n electric range across different makes and models ", electric_range_by_make)
electric_range_by_model = df.groupby('Model')['Electric Range'].mean()
print ("\n electric range across different models ", electric_range_by_model)
# What is the average Base MSRP for each EV model?
average_base_msrp_by_ev_model = df.groupby('Model')['Base MSRP'].mean()
print ("\n average Base MSRP for each EV model ", average_base_msrp_by_ev_model)
# Are there any regional trends in EV adoption (e.g., urban vs. rural areas)?
def check_urban_rural(census):
  if census >= 2000000000:
    return "urban"
  else:
    return "rural"
```

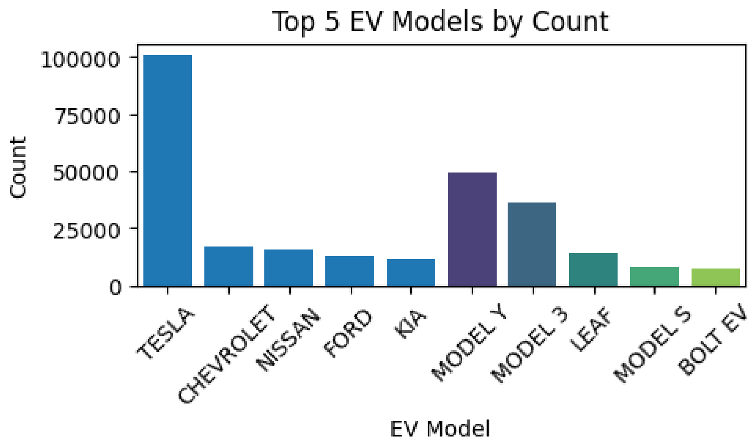
```
df["Urban/Rural"]= df["2020 Census Tract"].apply(check_urbal_rural)
ev_adoption_by_urban_rural = df.groupby('Urban/Rural')['Model'].count()
print ("\n regional trends in EV adoption ", ev_adoption_by_urban_rural)
```

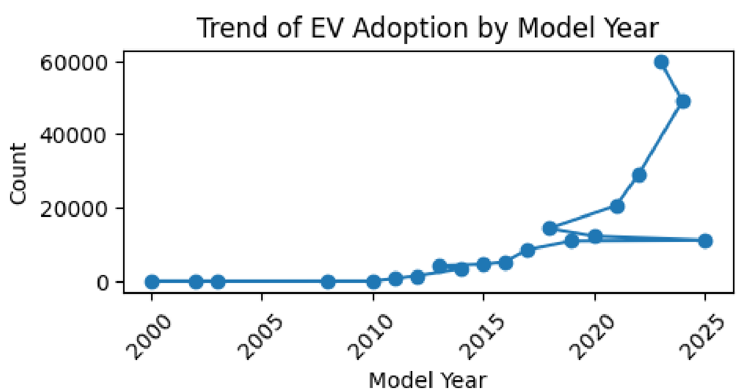
```
LAND ROVER                43.969697
LEXUS                     20.909281
LINCOLN                   24.188366
LUCID                      0.000000
MAZDA                     25.668627
MERCEDES-BENZ             11.362947
MINI                      14.279857
MITSUBISHI                31.368941
MULLEN AUTOMOTIVE INC.     0.000000
NISSAN                    70.090587
POLESTAR                  29.957143
PORSCHE                   55.243767
RAM                        0.000000
RIVIAN                     0.000000
ROLLS-ROYCE                0.000000
SMART                     61.801653
SUBARU                     0.785785
TESLA                     60.346241
TH!NK                    100.000000
TOYOTA                    27.708272
VINFAST                    0.000000
VOLKSWAGEN                19.246653
VOLVO                     17.864893
WHEEGO ELECTRIC CARS     100.000000
Name: Electric Range, dtype: float64

 electric range across different models  Model
330E    18.431068
500     85.693277
500E     0.000000
530E    16.165138
550E    40.000000
          ...
XC60    27.823014
XC90    24.630869
XM      31.000000
ZDX      0.000000
ZEVO     0.000000
Name: Electric Range, Length: 171, dtype: float64

 average Base MSRP for each EV model  Model
330E    15466.213592
500         0.000000
500E        0.000000
530E    35630.045872
550E        0.000000
            ...
XC60     7112.883052
XC90     2822.502498
XM          0.000000
ZDX         0.000000
ZEVO        0.000000
Name: Base MSRP, Length: 171, dtype: float64

 regional trends in EV adoption  Urban/Rural
rural        11
urban    235681
Name: Model, dtype: int64
```

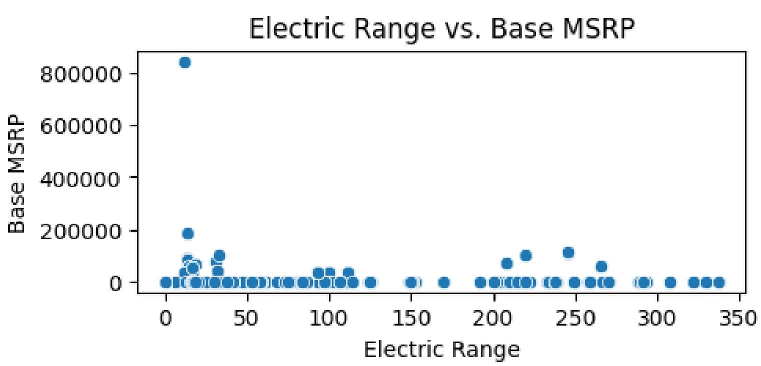
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv("Electric_Vehicle_Population_Data (1).csv")

# DATA VISUALIZATION
```

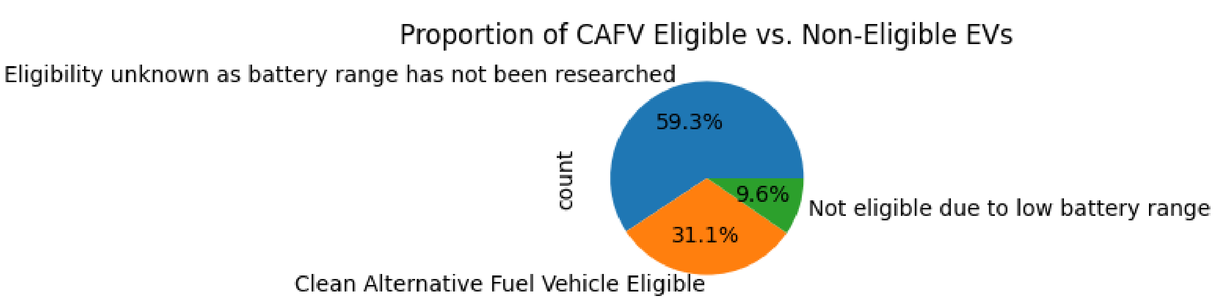
```
# DATA VISUALIZATION
#Create a bar chart showing the top 5 EV makes and models by count
plt.figure(figsize=(5, 2))
plt.bar(top_5_makes.index, top_5_makes.values)
plt.xlabel("EV Make")
plt.ylabel("Count")
plt.title("Top 5 EV Makes by Count")
plt.xticks(rotation=45)
plt.show
sns.countplot(x="Model", data=df, order=df["Model"].value_counts().head(5).index, palette="viridis")
plt.xlabel("EV Model")
plt.ylabel("Count")
plt.title("Top 5 EV Models by Count")
plt.xticks(rotation=45)
plt.show
#Use a heatmap or choropleth map to visualize EV distribution by county.

#Create a line graph showing the trend of EV adoption by model year
plt.figure(figsize=(5, 2))
ev_adoptation_over_model_year.plot(kind="line", marker="o")
plt.xlabel("Model Year")
plt.ylabel("Count")
plt.title("Trend of EV Adoption by Model Year")
plt.xticks(rotation=45)
# Generate a scatter plot comparing electric range vs. base MSRP to see pricing trends.
plt.figure(figsize=(5, 2))
sns.scatterplot(x = "Electric Range", y= "Base MSRP", data=df)
plt.xlabel("Electric Range")
plt.ylabel("Base MSRP")
plt.title("Electric Range vs. Base MSRP")
plt.show
# Plot a pie chart showing the proportion of CAFV-eligible vs. non-eligible EVs.
plt.figure(figsize=(5, 2))
df["Clean Alternative Fuel Vehicle (CAFV) Eligibility"].value_counts().plot(kind="pie", autopct="%1.1f%%")
plt.title("Proportion of CAFV Eligible vs. Non-Eligible EVs")
# Use a geospatial map to display EV registrations based on vehicle location.
import folium
#map = folium.Map(location=[df["Latitude"].mean(), df["Longitude"].mean()], zoom_start=6)
```

```
<ipython-input-24-8370a36c7dd4>:16: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable

  sns.countplot(x="Model", data=df, order=df["Model"].value_counts().head(5).index, palette="viridis")
```









Generated code may be subject to a license | GalvarezWarmi/Modelado-Preedictivo | Carmelkuta/UsedCarModelProject

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
df = pd.read_csv("Electric_Vehicle_Population_Data (1).csv")
# Linear Regression

# Linear Regression Model
features = ['Model Year', 'Base MSRP']
df = df.dropna(subset=['Electric Range', 'Base MSRP'])
X = df[features]
y = df['Electric Range']
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)
y_pred = model.predict(X_test)

r2 = r2_score(y_test, y_pred)
print(f'R² score: {r2:.2f}')

# Influence of Base MSRP on Electric Range
plt.figure(figsize=(8, 5))
sns.scatterplot(x=df['Base MSRP'], y=df['Electric Range'], alpha=0.5)
plt.title('Electric Range vs. Base MSRP')
plt.xlabel('Base MSRP')
plt.ylabel('Electric Range')
plt.show()

# Prediction for a new EV model
new_ev = pd.DataFrame({'Model Year': [2025], 'Base MSRP': [40000]})
predicted_range = model.predict(new_ev)
print(f'Predicted Electric Range for new EV: {predicted_range[0]:.2f} miles')




# 4.6 What steps are needed to improve the accuracy of the Linear Regression model?
# Possible steps: Feature engineering, adding more relevant features, using more complex models, etc.

# 4.7 Can we use this model to predict the range of new EV models based on their specifications?
# Yes, the model can be used to predict the range of new EV models by providing their specifications.
```