



CURSO: Tecnologia Em Ciência De Dados

POLO DE APOIO PRESENCIAL: Jundiaí e Higienópolis e Campinas

SEMESTRE: 1/2024

COMPONENTE CURRICULAR / TEMA: **PROJETO APLICADO I {TURMA 02A} 2024/1**

NOME DO GRUPO – MacGyver

RA 10415058 – EDUARDO DAVID - 10415058@MACKENZISTA.COM.BR
RA 10415270 – FELIPE JOSÉ DA CUNHA - 10415270@MACKENZISTA.COM.BR
RA 10415636 – NATÁLIA FRANÇOZO - 10415636@MACKENZISTA.COM.BR
RA 10415977 – ANA VITÓRIA SILVA - 10415977@MACKENZISTA.COM.BR

NOME DO PROFESSOR: **Prof. Dr. Felipe Albino dos Santos**

ETAPA 1

- a) Definir o grupo de trabalho.
- b) Estabelecer as premissas do projeto: escolha da empresa, área de atuação e apresentação dos dados que serão utilizados (imagem ou texto).
- c) Elaborar objetivos e metas.
- d) Definir cronograma de atividades.



1. Título:	3
2. Introdução	3
3. Objetivos:	3
4. Metas:	4
5. Cronograma:	5
6. Fluxo Baseado em Pensamento Computacional em Contextos Organizacionais:	6
A. <i>Decomposição</i> :.....	6
B. <i>Reconhecimento de padrões</i> :.....	6
C. <i>Abstração</i> :	6
D. <i>Design de Algoritmos</i> :	6
7. Artefatos do Projeto:	6
A. <i>Link Github</i> :	6
B. <i>Link Projeto</i> :	6
8. Referências de aquisição do dataset:	6
9. Organização e o contexto em que os dados foram gerados:	6
10. Dataset e Metadados	8
A. <i>Dataset</i> :	8
B. <i>Descrição do Dataset</i> :	8
C. <i>Metadados</i> :.....	9

Tabelas

<i>Tabela 01</i>	<i>Cronograma de Tarefas PAII</i>	5
<i>Tabela 02</i>	<i>Data Columns</i>	9



1. Título:

Empresa: Green Energy

Core Business: Infraestrutura de Carregamento para Automóveis Elétricos.

2. Introdução

Os veículos elétricos emergiram como uma resposta promissora aos desafios ambientais e sociais globais. Sua importância reside não apenas na redução das emissões de gases de efeito estufa e na melhoria da qualidade do ar nas áreas urbanas, mas também na diminuição da dependência de combustíveis fósseis e na diversificação das fontes de energia. Além disso, os veículos elétricos apresentam uma oportunidade única de impulsionar a transição para uma economia mais sustentável, criando novos empregos na indústria de energia limpa e estimulando a inovação tecnológica.

No entanto, a autonomia dos veículos elétricos ainda é um fator crucial para sua aceitação em massa. Embora os avanços na tecnologia tenham estendido significativamente a autonomia dos veículos elétricos nos últimos anos, ainda existe uma necessidade de expandir as redes de abastecimento para garantir uma experiência de condução conveniente e livre de preocupações para os seus proprietários. Isso implica investimentos contínuos em infraestrutura de carregamento, incluindo estações de carregamento rápido em áreas urbanas e rodovias, bem como soluções para carregamento em domicílio ou em locais de trabalho.

Uma rede robusta de abastecimento não só aumenta a confiança do consumidor na adoção de veículos elétricos, mas também desempenha um papel fundamental na redução das emissões de gases de efeito estufa e na promoção de uma mobilidade sustentável em todo o mundo.

3. Objetivos:

Este estudo tem como foco a análise e otimização da infraestrutura de carregamento de veículos elétricos no estado de WA (Washington) dos Estados Unidos. O objetivo principal é compreender a distribuição atual e as necessidades futuras dessa infraestrutura para suportar eficientemente o crescimento contínuo do mercado de veículos elétricos. Especificamente, buscaremos:

- Avaliar a distribuição geográfica da infraestrutura de carregamento de veículos elétricos no estado de WA.
- Identificar áreas com alta demanda de carregamento de veículos elétricos e baixa disponibilidade de estações de recarga.
- Identificar quais as Marcas de veículos elétricos mais vendidas.



- Analisar a evolução temporal da infraestrutura de carregamento em WA.
- Identificar padrões de crescimento e lacunas na infraestrutura de carregamento em relação ao aumento do número de veículos elétricos emplacados.
- Propor recomendações para otimizar a expansão da infraestrutura com base nas análises realizadas.

4. Metas:

Buscamos representar através da escolha das metas um plano estruturado para analisar e entender o desenvolvimento do mercado de veículos elétricos (BEVS) nos Estados Unidos, um setor em rápido crescimento e de grande importância para as estratégias de sustentabilidade e inovação tecnológica. As metas foram cuidadosamente selecionadas para abranger aspectos cruciais da dinâmica do mercado de BEVS, desde a aquisição de veículos até a infraestrutura de carregamento.

- Coletar e integrar dados de emplacamento de veículos elétricos em WA nos EUA de fontes confiáveis.
- Desenvolver uma metodologia robusta para analisar a distribuição e evolução da infraestrutura de carregamento.
- Realizar análises geoespaciais para mapear a cobertura atual e identificar lacunas na infraestrutura de carregamento.
- Utilizar técnicas de visualização de dados para comunicar eficazmente os resultados da análise.
- Produzir um relatório final com insights açãoáveis e recomendações para stakeholders relevantes.



5. Cronograma:

Link:

<https://github.com/meddavid/Mackenzie-Projeto-Aplicado>

[II/blob/9a2bdc2681d80fe877677c40268520b7b067057d/01.%20ENTREGA%20ETAPA%2001/CRONOGRAMA%20-%20Projeto%20Aplicado%20II.xlsx](https://blob/9a2bdc2681d80fe877677c40268520b7b067057d/01.%20ENTREGA%20ETAPA%2001/CRONOGRAMA%20-%20Projeto%20Aplicado%20II.xlsx)

EDT	Nome da tarefa	Duração	Início	Término	Nomes dos recursos	IIRR	TTRR
1	PROJETO - Green Energy	83 dias	Qui 08/02/24	Sex 31/05/24		Qui 08/02/24	ND
1.1	FASE I – PREPARAÇÃO DE DATASET	20 dias	Qui 08/02/24	Qua 06/03/24		Qui 08/02/24	ND
1.1.1	Artefatos do projeto	1 dia	Qui 08/02/24	Qui 08/02/24		Qui 08/02/24	Qui 08/02/24
1.1.1.1	Link GITHUB	1 dia	Qui 08/02/24	Qui 08/02/24	Eduardo[50%]	Qui 08/02/24	Qui 08/02/24
1.1.1.2	Dataset	1 dia	Qui 08/02/24	Qui 08/02/24	Eduardo[50%]	Qui 08/02/24	Qui 08/02/24
1.1.2	Contexto do Estudo	1 dia	Sex 09/02/24	Sex 09/02/24		Sex 09/02/24	Sex 09/02/24
1.1.2.1	Premissas do Projeto (definição de empresa, Core Business)	1 dia	Sex 09/02/24	Sex 09/02/24	Natalia	Sex 09/02/24	Sex 09/02/24
1.1.2.2	Objetivos	1 dia	Sex 09/02/24	Sex 09/02/24	Felipe	Sex 09/02/24	Sex 09/02/24
1.1.2.3	Metas	1 dia	Sex 09/02/24	Sex 09/02/24	Felipe	Sex 09/02/24	Sex 09/02/24
1.1.2.4	Cronograma de Atividades	1 dia	Sex 09/02/24	Sex 09/02/24	Eduardo[50%]	Sex 09/02/24	Sex 09/02/24
1.1.2.5	Pensamento Computacional em contextos organizacionais	1 dia	Sex 09/02/24	Sex 09/02/24	Eduardo[50%];Felipe;Natalia	Sex 09/02/24	Sex 09/02/24
1.1.3	Referências de Aquisição do Dataset	1 dia	Qua 21/02/24	Qua 21/02/24		Qua 21/02/24	Qua 21/02/24
1.1.3.1	Origem dos Dados	1 dia	Qua 21/02/24	Qua 21/02/24	Eduardo;Felipe;Natalia	Qua 21/02/24	Qua 21/02/24
1.1.3.2	Limitação de Uso	1 dia	Qua 21/02/24	Qua 21/02/24	Eduardo;Felipe;Natalia	Qua 21/02/24	Qua 21/02/24
1.1.3.3	Período de Coleta	1 dia	Qua 21/02/24	Qua 21/02/24	Eduardo;Felipe;Natalia	Qua 21/02/24	Qua 21/02/24
1.1.4	Descrição da Origem	1 dia	Seg 04/03/24	Seg 04/03/24		Seg 04/03/24	Seg 04/03/24
1.1.4.1	Informações sobre a Organização que gerou os dados	1 dia	Seg 04/03/24	Seg 04/03/24	Eduardo;Felipe;Natalia	Seg 04/03/24	Seg 04/03/24
1.1.4.2	Contexto em que os dados foram gerados	1 dia	Seg 04/03/24	Seg 04/03/24	Eduardo;Felipe;Natalia	Seg 04/03/24	Seg 04/03/24
1.1.5	Descrição do Dataset	1 dia	Ter 05/03/24	Ter 05/03/24		Ter 05/03/24	Ter 05/03/24
1.1.5.1	Conteúdo	1 dia	Ter 05/03/24	Ter 05/03/24	Eduardo;Felipe;Natalia	Ter 05/03/24	Ter 05/03/24
1.1.5.2	Proposta	1 dia	Ter 05/03/24	Ter 05/03/24	Eduardo;Felipe;Natalia	Ter 05/03/24	Ter 05/03/24
1.1.5.3	Registro de Problemas ou Fenômenos	1 dia	Ter 05/03/24	Ter 05/03/24	Eduardo;Felipe;Natalia	Ter 05/03/24	Ter 05/03/24
1.1.6	Entrega Moodle - Etapa I	1 dia	Qua 06/03/24	Qua 06/03/24	Eduardo	Qua 06/03/24	Qua 06/03/24
1.2	FASE II - Definição de Método Analítico	19 dias	Qui 07/03/24	Ter 02/04/24	Eduardo;Felipe;Natalia;Ana	Qui 07/03/24	Qui 07/03/24
1.2.1	Definição da linguagem de programação usada no projeto.	1 dia	Qui 07/03/24	Qui 07/03/24	Eduardo;Felipe;Natalia;Ana	Qui 07/03/24	Qui 07/03/24
1.2.2	Análise exploratória da base de dados escolhida.	1 dia	Sex 08/03/24	Sex 08/03/24	Eduardo;Felipe;Natalia;Ana	Sex 08/03/24	Sex 08/03/24
1.2.3	Tratamento da base de dados (Preparação e treinamento).	1 dia	Seg 11/03/24	Seg 11/03/24	Eduardo;Felipe;Natalia;Ana	Seg 11/03/24	Seg 11/03/24
1.2.4	Definição e descrição das bases teóricas dos métodos.	1 dia	Ter 12/03/24	Ter 12/03/24	Eduardo;Felipe;Natalia;Ana	Ter 12/03/24	Ter 12/03/24
1.2.5	Definição e descrição de como será calculada a acurácia.	1 dia	Ter 12/03/24	Ter 12/03/24	Eduardo;Felipe;Natalia;Ana	Ter 12/03/24	Ter 12/03/24
1.2.6	BB Professor	1 dia	Ter 19/03/24	Ter 19/03/24	Eduardo;Felipe;Natalia;Ana	Ter 19/03/24	Ter 19/03/24
1.2.9	Entrega Moodle - Etapa II	1 dia	Ter 02/04/24	Ter 02/04/24	Eduardo	Ter 02/04/24	Ter 02/04/24
1.3	FASE III - TBD	18 dias	Qua 03/04/24	Sáb 27/04/24	Eduardo;Felipe;Natalia	ND	ND
1.3.1	BB Professor	1 dia	Qua 03/04/24	Qua 03/04/24	Eduardo;Felipe;Natalia	ND	ND
1.3.2	Atividades - TBD	1 dia	Qua 03/04/24	Qua 03/04/24	Eduardo;Felipe;Natalia	ND	ND
1.3.3	BB Professor	1 dia	Qua 03/04/24	Qua 03/04/24	Eduardo;Felipe;Natalia	ND	ND
1.3.4	Entrega Moodle - Etapa III	0 dias	Sáb 27/04/24	Sáb 27/04/24	Eduardo	ND	ND
1.4	FASE IV - TBD	26 dias	Sáb 27/04/24	Sex 31/05/24	Eduardo;Felipe;Natalia	ND	ND
1.4.1	BB Professor	1 dia	Sáb 27/04/24	Seg 29/04/24	Eduardo;Felipe;Natalia	ND	ND
1.4.2	Atividades - TBD	1 dia	Sáb 27/04/24	Seg 29/04/24	Eduardo;Felipe;Natalia	ND	ND
1.4.3	Entrega Moodle - Etapa IV	1 dia	Sex 31/05/24	Sex 31/05/24	Eduardo	ND	ND

Tabela 1



6. Fluxo Baseado em Pensamento Computacional em Contextos Organizacionais:

A. Decomposição:

Dividir o problema em partes menores: quantidade de marcas que produzem e vendem veículos elétricos, quantidade de veículos em circulação em Washington.

B. Reconhecimento de padrões:

Analizar relação entre marcas, comparar o consumo entre diferentes regiões, entre outros.

C. Abstração:

Construir uma análise exploratória sobre veículos: Utilizar dados e pesquisas atuais para criar análise.

D. Design de Algoritmos:

Criar um relatório para tomada de decisão: Com base nas análises, produzir relatório com recomendações para organizações.

7. Artefatos do Projeto:

A. Link Github:

<https://github.com/meddavid/>

B. Link Projeto:

<https://github.com/meddavid/Mackenzie-Projeto-Aplicado-II>

8. Referências de aquisição do dataset:

Os dados têm origem no site oficial do governo dos Estados Unidos que apresenta um conjunto de dados que mostra os Veículos Elétricos de Bateria (BEVs) e os Veículos Elétricos Híbridos Plug-in (PHEVs) que estão atualmente registrados através do Departamento de Licenciamento (DOL) do Estado de Washington.

Este conjunto de dados destina-se ao acesso e uso público e foi atualizado em 17 de fevereiro de 2024.

9. Organização e o contexto em que os dados foram gerados:



Este projeto será conduzido em algumas fases essenciais, garantindo precisão técnica e relevância dos resultados:

Coleta e Limpeza de Dados: A fase inicial envolve a coleta de dados sobre emplacamentos de veículos elétricos, seguida de um processo rigoroso de limpeza de dados para assegurar precisão e usabilidade.

Análise Exploratória de Dados: Em seguida, uma análise exploratória será realizada para compreender a integridade, estrutura e qualidades dos dados, estabelecendo uma base sólida para análises mais complexas.

Modelagem Analítica: Com os dados preparados, procederemos ao desenvolvimento de modelos analíticos focados na avaliação da infraestrutura de carregamento de veículos elétricos. Estes modelos buscarão identificar padrões, tendências e deficiências.

Visualização de Dados: Utilizaremos a linguagem Python para visualizações avançadas de dados para uma interpretação e comunicação eficaz dos resultados, tornando as informações mais acessíveis e compreensíveis.

Formulação de Recomendações: Baseando-se nos insights analíticos, serão desenvolvidas recomendações estratégicas para orientar a otimização e expansão da infraestrutura de carregamento, em linha com as necessidades do mercado.

Acurácia : Para calcular a acurácia de um modelo de regressão linear, iremos utilizar o coeficiente de determinação R^2 , que é a métrica comumente usada para avaliar o desempenho de modelos de regressão. O coeficiente varia de 0 a 1, onde 1 indica um ajuste perfeito do modelo aos dados. No código você pode encontrá-lo na linha `model.score(X_test, y_test)`.

Cada etapa é projetada para garantir uma abordagem técnica rigorosa, desde a coleta de dados até a formulação de recomendações, assegurando que o projeto ofereça diretrizes eficazes para o desenvolvimento da infraestrutura de carregamento de veículos elétricos.



10. Dataset e Metadados

A. Dataset:

Fonte: <https://catalog.data.gov/dataset/electric-vehicle-population-data>

Link para download: https://github.com/meddavid/Mackenzie-Projeto-Aplicado-II/blob/5c65804450e8705bae21684d3d047de2523a6646/01.%20ENTREGA%20ETAPA%2001/Electric_Vehicle_Population_Data.rar

B. Descrição do Dataset:

Este conjunto de dados, intitulado "Electric_Vehicle_Population_Data", oferece uma visão abrangente sobre a população de veículos elétricos, abrangendo várias dimensões e características. Vamos detalhar o que cada parte deste conjunto de dados representa e como utilizaremos para análises diversas:

Identificação e Detalhes do Veículo: Cada entrada no conjunto de dados começa com um Número de Identificação do Veículo (VIN), seguido por informações essenciais como marca, modelo, ano do modelo e o tipo de veículo elétrico. Os tipos de veículos elétricos são categorizados principalmente como "Battery Electric Vehicle (BEV)" ou "Plug-in Hybrid Electric Vehicle (PHEV)", indicando se são totalmente elétricos ou híbridos. Esta seção é crucial para entender a variedade e popularidade de diferentes veículos elétricos no mercado.

Localização e Demografia: A localização geográfica é um aspecto fundamental deste conjunto de dados. Inclui detalhes como condado, cidade, estado e código postal. Além disso, há coordenadas geográficas precisas para cada veículo. Isso possibilita uma análise regional da adoção de veículos elétricos, revelando padrões geográficos e potenciais lacunas na infraestrutura de suporte.

Elegibilidade Ambiental e Alcance Elétrico: Uma característica interessante é a indicação de se um veículo é classificado como um "Clean Alternative Fuel Vehicle (CAFV)" e seu alcance elétrico. Isso reflete a eficiência e o impacto ambiental dos veículos, essenciais para avaliar o progresso em direção a objetivos de sustentabilidade.

Aspectos Econômicos: O conjunto de dados inclui o preço base (MSRP) de cada veículo, embora muitos registros mostrem valores zerados, o que pode limitar análises econômicas. Teoricamente, se estes dados estivessem completos, poderiam oferecer insights sobre o custo médio e acessibilidade de veículos elétricos.



Fornecedor de Energia: Cada entrada lista a companhia de energia elétrica associada ao veículo. Essa informação é valiosa para entender a relação entre a infraestrutura de energia e a adoção de veículos elétricos.

Dados do Censo: A inclusão de códigos do censo de 2020 abre possibilidades para análises demográficas detalhadas em relação à propriedade de veículos elétricos.

C. Metadados:

O conjunto de dados "Electric_Vehicle_Population_Data" carregado em Python como infra descrito, possui a seguinte estrutura e informações estatísticas:

Estrutura:

Total de Entradas: 173.533

Total de Colunas: 17

Tipos de Dados: Objeto (strings), float64 e int64

As colunas incluídas no conjunto de dados são:

#	Column	Non-Nu	Il Count	Dtype
---	-----	-----	-----	-----
0	VIN (1-10)	173533	non-null	object
1	County	173528	non-null	object
2	City	173528	non-null	object
3	State	173533	non-null	object
4	Postal Code	173528	non-null	float64
5	Model Year	173533	non-null	int64
6	Make	173533	non-null	object
7	Model	173533	non-null	object
8	Electric Vehicle Type	173533	non-null	object
9	Clean Alternative Fuel Vehicle (CAFV) Eligibility	173533	non-null	object
10	Electric Range	173532	non-null	float64
11	Base MSRP	173532	non-null	float64
12	Legislative District	173157	non-null	float64
13	DOL Vehicle ID	173533	non-null	int64
14	Vehicle Location	173523	non-null	object
15	Electric Utility	173528	non-null	object
16	2020 Census Tract	173528	non-null	float64

Tabela 2



Resumo Estatístico:

Código Postal: Varia de 1.545 a 99.577 com a média aproximada de 98.174.

Ano do Modelo: Os anos vão de 1997 a 2024, sendo a maioria dos modelos de 2020 em diante.

Alcance Elétrico: Varia de 0 a 337, com uma média de 60.

Preço Base (MSRP): A maioria dos valores está a zero, limitando a análise econômica. O valor máximo registrado é de 845.000.

Distrito Legislativo: Varia de 1 a 49.

ID do Veículo DOL: Os IDs variam em uma grande faixa, indicando um número substancial de veículos registrados.

As primeiras linhas mostram exemplos de registros de veículos elétricos, incluindo informações sobre marca, modelo, tipo de veículo, localização e características técnicas como alcance elétrico.

Código utilizado:

```
'''  
=====  
Programa.....: SUMMARY  
Autor.....: Eduardo David  
Data.....: 01/03/2024  
Descrição / Objetivo: Exibição de Análise de Metadados  
Doc. Origem.....: Electric_Vehicle_Population_Data.csv  
Solicitante.....: Professor Felipe Cunha  
Uso.....: Projeto Apliado II  
Modificações.....: 01/03/2024 - Desenvolvimento  
=====  
'  
  
import pandas as pd  
  
# Caminho atualizado do arquivo CSV  
file_path = "N:\\\\Drives compartilhados\\\\.....EDUARDO\\\\... MACKENZIE\\\\...TERCEIRO SEMESTRE\\\\PROJETO-APLICADO-II\\\\01. ENTREGA ETAPA 01\\\\Electric_Vehicle_Population_Data.csv"  
  
# Carregando o arquivo CSV  
df = pd.read_csv(file_path)  
  
# Exibindo as primeiras linhas para uma visão geral  
first_rows = df.head()  
  
# Resumo da estrutura do conjunto de dados  
structure = df.info()
```



```
# Resumo estatístico básico
summary = df.describe()

first_rows, structure, summary

#Começamos por Carregar as bibliotecas
#utilizadas neste Projeto

import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
import random
import seaborn as sns
import fsspec
import numpy as np
import seaborn as sns
```

```
Nossa base de dados
base_dados = pd.read_csv('D:/OneDrive - Instituto Presbiteriano Mackenzie/Mackenzie/Aulas/3 semestre/Projeto Aplicado II/Electric_Vehicle_Population_Data.csv')
base_dados.head()
base_dados.info()
base_dados.dropna()
```

base_dados.head()

Out[3]:

	VIN (1-10)	County ...	Electric Utility 2020 Census Tract	
0	5UXKT0C59G	Yakima ...	PACIFICORP	5.307700e+10
1	5YJ3E1EA2J	Snohomish ...	PUGET SOUND ENERGY INC	5.306105e+10
2	1G1RE6E4XE	Kitsap ...	PUGET SOUND ENERGY INC	5.303509e+10
3	2C4RC1L76M	Skagit ...	PUGET SOUND ENERGY INC	5.305795e+10
4	5YJ3E1EA2J	Thurston ...	PUGET SOUND ENERGY INC	5.306701e+10

[5 rows x 17 columns]



```
base_dados.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 173533 entries, 0 to 173532
Data columns (total 17 columns):
 #   Column           Non-Null Count Dtype  
 --- 
 0   VIN (1-10)      173533 non-null  object 
 1   County          173528 non-null  object 
 2   City            173528 non-null  object 
 3   State           173533 non-null  object 
 4   Postal Code    173528 non-null  float64 
 5   Model Year     173533 non-null  int64  
 6   Make            173533 non-null  object 
 7   Model           173533 non-null  object 
 8   Electric Vehicle Type 173533 non-null  object 
 9   Clean Alternative Fuel Vehicle (CAFV) Eligibility 173533 non-null  object 
 10  Electric Range 173532 non-null  float64 
 11  Base MSRP      173532 non-null  float64 
 12  Legislative District 173157 non-null  float64 
 13  DOL Vehicle ID 173533 non-null  int64  
 14  Vehicle Location 173523 non-null  object 
 15  Electric Utility 173528 non-null  object 
 16  2020 Census Tract 173528 non-null  float64 

dtypes: float64(5), int64(2), object(10)
memory usage: 22.5+ MB
```

```
base_dados.dropna()
Out[5]:
VIN (1-10) ... 2020 Census Tract
0  5UXKT0C59G ... 5.307700e+10
1  5YJ3E1EA2J ... 5.306105e+10
2  1G1RE6E4XE ... 5.303509e+10
3  2C4RC1L76M ... 5.305795e+10
4  5YJ3E1EA2J ... 5.306701e+10
...
173528 5YJ3E1EA0P ... 5.303303e+10
```



```
173529 5YJXCBE22H ... 5.306105e+10
173530 1C4RJXR65R ... 5.303303e+10
173531 5UXKT0C50G ... 5.303301e+10
173532 5YJSA1E5XM ... 5.303300e+10
```

[173151 rows x 17 columns]

```
Preparaçao e Tratamento dos dados
base_dados.isnull().sum() / len(base_dados)

print('Before', len(base_dados))
base_dados = base_dados.dropna()
print('After', len(base_dados))
    base_dados.isnull().sum() / len(base_dados)
```

Out[6]:

VIN (1-10)	0.000000
County	0.000029
City	0.000029
State	0.000000
Postal Code	0.000029
Model Year	0.000000
Make	0.000000
Model	0.000000
Electric Vehicle Type	0.000000
Clean Alternative Fuel Vehicle (CAFV) Eligibility	0.000000
Electric Range	0.000006
Base MSRP	0.000006
Legislative District	0.002167
DOL Vehicle ID	0.000000
Vehicle Location	0.000058
Electric Utility	0.000029
2020 Census Tract	0.000029
dtype:	float64

```
print('Before', len(base_dados))
```



Before 173533

```
base_dados = base_dados.dropna()
```

```
print('After', len(base_dados))
```

After 173151

```
Analisando o Banco de Dados selecionamos as colunas para o projeto
dados_selecionados = pd.DataFrame(base_dados)
colunas_selecionadas = ['City', 'State', 'Model Year',
'Make', 'Model',
                           'Electric Vehicle Type', 'Electric Range',
                           'Electric Utility', 'Vehicle Location']

df = dados_selecionados[colunas_selecionadas]
```

```
df.head(15)
df.tail()
df.index
df.columns
df.shape
```

df.head(15)

Out[19]:

	City ...	Vehicle Location
0	Zillah ...	POINT (-120.26317 46.40556)
1	Edmonds ...	POINT (-122.37507 47.80807)
2	Port Orchard ...	POINT (-122.6847073 47.50524)
3	Bow ...	POINT (-122.440636 48.5613885)
4	Olympia ...	POINT (-122.817545 46.98876)
5	Snohomish ...	POINT (-122.15134 47.8851158)
6	Olympia ...	POINT (-122.8874781 47.0519573)
7	Edmonds ...	POINT (-122.37507 47.80807)
8	Seattle ...	POINT (-122.32226 47.64058)
9	Lacey ...	POINT (-122.8285 47.03646)



```
10  Bothell ... POINT (-122.179458 47.802589)
11  Fall City ... POINT (-121.8936184 47.5640832)
12  Olympia ... POINT (-122.92145 47.045935)
13  Bothell ... POINT (-122.1873 47.820245)
14  Seattle ... POINT (-122.37275 47.68968)
```

```
df.tail()
Out[20]:
      City ... Vehicle Location
173528 Redmond ... POINT (-122.12302 47.67668)
173529 Snohomish ... POINT (-122.15134 47.8851158)
173530 Kent ... POINT (-122.2012521 47.3931814)
173531 Seattle ... POINT (-122.394185 47.639195)
173532 Seattle ... POINT (-122.37275 47.68968)

df.index
Out[21]:
Index([ 0,  1,  2,  3,  4,  5,  6,  7,  8,
       9,
       ...
       173523, 173524, 173525, 173526, 173527, 173528, 173529, 173530, 173531,
       173532],
      dtype='int64', length=173151)

df.index
Out[21]:
Index([ 0,  1,  2,  3,  4,  5,  6,  7,  8,
       9,
       ...
       173523, 173524, 173525, 173526, 173527, 173528, 173529, 173530, 173531,
       173532],
      dtype='int64', length=173151)
```

Agrupamos os dados por marca, separamos armazenamos em subdataframes e criamos um script para consulta

```
# Separar os dados por marcas
marcas = df['Make'].unique()

# Dicionário para armazenar os subdataframes
subdataframes = {}

# Iterando sobre as marcas únicas
```



```
for marca in marcas:  
    # Criando o subdataframe para cada marca  
    subdataframes[marca] = df[df['Make'] == marca]  
  
    # Consultando e criando um novo dataset com as primeiras li-  
    nhas do subdataframe por marca  
    for marca_desejada in subdataframes.keys():  
        print(f"\nSubdataframe da marca {marca_desejada}:")  
  
sub_df = subdataframes[marca_desejada] #aqui pode colocar a consulta, ex .head()  
globals()[f"sub_df_{marca_desejada}"] = sub_df  
print(sub_df)
```



Subdataframe da marca VOLKSWAGEN:

	City	...	Vehicle Location
54	Brier	...	POINT (-122.316675 47.819365)
93	Hansville	...	POINT (-122.57781 47.903975)
173	Edmonds	...	POINT (-122.335685 47.80372)
196	Redmond	...	POINT (-122.12302 47.67668)
199	Hansville	...	POINT (-122.57781 47.903975)
...			
173251	East Wenatchee	...	POINT (-120.28674 47.4176)
173351	Tacoma	...	POINT (-122.490985 47.26365)
173359	Seattle	...	POINT (-122.34301 47.659185)
173463	Seattle	...	POINT (-122.37275 47.68968)
173505	Issaquah	...	POINT (-121.9993659 47.5484866)

[4876 rows x 9 columns]

Subdataframe da marca FIAT:

	City	...	Vehicle Location
69	Langley	...	POINT (-122.408015 48.03557)
203	Bellevue	...	POINT (-122.16937 47.571015)
299	Port Townsend	...	POINT (-122.7644197 48.1195874)
692	Olympia	...	POINT (-122.8874781 47.0519573)
2138	Seattle	...	POINT (-122.38679 47.56484)
...			
171881	Seattle	...	POINT (-122.296385 47.71558)
172614	Seatac	...	POINT (-122.29179 47.43473)
172692	Bellingham	...	POINT (-122.486115 48.761615)
172892	Seattle	...	POINT (-122.3185 47.67949)
173352	Bellevue	...	POINT (-122.11832 47.6245)

[801 rows x 9 columns]

Subdataframe da marca MITSUBISHI:

	City	...	Vehicle Location
81	Olympia	...	POINT (-122.89692 47.043535)
145	Shoreline	...	POINT (-122.3175 47.7578146)
307	Seattle	...	POINT (-122.355145 47.505655)
420	Gig Harbor	...	POINT (-122.6657985 47.383359)
486	Spokane Valley	...	POINT (-117.1407 47.673675)
...			



171570	West Richland	...	POINT	(-119.3535873	46.2778489)
171640	Issaquah	...	POINT	(-121.9993659	47.5484866)
172560	Chelan	...	POINT	(-120.015875	47.839895)
173328	Olympia	...	POINT	(-122.89692	47.043535)
173357	Olympia	...	POINT	(-122.817545	46.98876)

[966 rows x 9 columns]

Subdataframe da marca JAGUAR:

	City	...	Vehicle Location
88	Zillah	...	POINT (-120.26317 46.40556)
369	Kenmore	...	POINT (-122.2504747 47.7617128)
484	Bothell	...	POINT (-122.179458 47.802589)
1104	Renton	...	POINT (-122.1298876 47.4451257)
1220	Kirkland	...	POINT (-122.209285 47.71124)

172679	Mill Creek	...	POINT (-122.1873 47.820245)
172781	Snohomish	...	POINT (-122.091505 47.915555)
173292	Lacey	...	POINT (-122.7474291 47.0821119)
173446	Kenmore	...	POINT (-122.2504747 47.7617128)
173506	Redmond	...	POINT (-122.12302 47.67668)

[227 rows x 9 columns]

Subdataframe da marca HONDA:

	City	...	Vehicle Location
89	Bremerton	...	POINT (-122.611365 47.575195)
95	Oak Harbor	...	POINT (-122.6788673 48.2897314)
520	Redmond	...	POINT (-122.12302 47.67668)
579	Vancouver	...	POINT (-122.6483953 45.7010427)
583	Olympia	...	POINT (-122.8285 47.03646)

173277	Black Diamond	...	POINT (-122.00451 47.312185)
173342	Seattle	...	POINT (-122.37815 47.66866)
173474	White Salmon	...	POINT (-121.48347 45.72977)
173512	Kent	...	POINT (-122.2012521 47.3931814)
173516	Edmonds	...	POINT (-122.335685 47.80372)

[826 rows x 9 columns]



Universidade Presbiteriana Mackenzie

Subdataframe da marca MERCEDES-BENZ:

	City ...	Vehicle Location
102	Yakima ...	POINT (-120.500225 46.6043)
190	Seattle ...	POINT (-122.329815 47.57981)
194	Kent ...	POINT (-122.111625 47.36078)
700	Edmonds ...	POINT (-122.37507 47.80807)
828	Renton ...	POINT (-122.180505 47.500055)

172972	Bothell ...	POINT (-122.1873 47.820245)
173179	Redmond ...	POINT (-122.12302 47.67668)
173333	Monroe ...	POINT (-121.972215 47.85674)
173442	Bellevue ...	POINT (-122.201905 47.61385)
173485	Seattle ...	POINT (-122.3185 47.67949)

[1516 rows x 9 columns]

Subdataframe da marca VOLVO:

	City ...	Vehicle Location
112	Olympia ...	POINT (-122.92145 47.045935)
176	Bellingham ...	POINT (-122.45493 48.76809)
220	Issaquah ...	POINT (-121.9993659 47.5484866)
233	Olympia ...	POINT (-122.9131017 47.0135926)
240	Kirkland ...	POINT (-122.20264 47.6785)

173282	Marysville ...	POINT (-122.1713847 48.10433)
173370	Seattle ...	POINT (-122.38679 47.56484)
173375	Winthrop ...	POINT (-120.1774093 48.4741766)
173429	Seattle ...	POINT (-122.37275 47.68968)
173502	Sammamish ...	POINT (-121.9993659 47.5484866)

[4075 rows x 9 columns]

Subdataframe da marca LEXUS:

	City ...	Vehicle Location
121	Spokane ...	POINT (-117.42694 47.67946)
202	Seatac ...	POINT (-122.29179 47.43473)
2183	Kirkland ...	POINT (-122.209285 47.71124)
3650	Seabeck ...	POINT (-122.847462 47.63836)
4044	Longview ...	POINT (-122.9379953 46.1372997)



171159	Mukilteo	...	POINT	(-122.299965	47.94171)
171309	Redmond	...	POINT	(-122.12302	47.67668)
171558	Federal Way	...	POINT	(-122.36363	47.30675)
173111	Seattle	...	POINT	(-122.38679	47.56484)
173157	Seattle	...	POINT	(-122.34301	47.659185)

[348 rows x 9 columns]

Subdataframe da marca LUCID:

	City	...	Vehicle Location
152	Coupeville	...	POINT (-122.6880708
1127	Cheney	...	POINT (-117.57579
1623	Castle Rock	...	POINT (-122.90778
1931	Gig Harbor	...	POINT (-122.5835454
2860	Sammamish	...	POINT (-122.0313266

169177	Seattle	...	POINT (-122.319115
171829	Renton	...	POINT (-122.15734
172024	Bremerton	...	POINT (-122.611365
172356	Lake Forest Park	...	POINT (-122.3175
173268	Seattle	...	POINT (-122.394185

[236 rows x 9 columns]

Subdataframe da marca SUBARU:

	City	...	Vehicle Location
170	Bothell	...	POINT (-122.20578
696	Woodinville	...	POINT (-122.151665
831	Shoreline	...	POINT (-122.3175
967	Renton	...	POINT (-122.15734
1056	Seattle	...	POINT (-122.356145

172746	Seattle	...	POINT (-122.394185
172816	Olympia	...	POINT (-122.89692
172832	Seattle	...	POINT (-122.374105
173313	Seattle	...	POINT (-122.34848
173368	Puyallup	...	POINT (-122.2987976

[806 rows x 9 columns]



Subdataframe da marca SMART:

	City	...	Vehicle Location
219	Bellevue	...	POINT (-122.147385 47.599975)
1383	Seattle	...	POINT (-122.234385 47.494545)
2618	Blaine	...	POINT (-122.74499 48.99505)
2663	Vancouver	...	POINT (-122.4853873 45.6083347)
4338	Lake Forest Park	...	POINT (-122.3175 47.7578146)

169226	Bellingham	...	POINT (-122.486115 48.761615)
170717	Vancouver	...	POINT (-122.5918493 45.6617058)
171880	Maple Valley	...	POINT (-122.05191 47.357985)
172558	Seattle	...	POINT (-122.37275 47.68968)
173227	Lake Stevens	...	POINT (-122.112265 48.0047)

[274 rows x 9 columns]

Subdataframe da marca PORSCHE:

	City	...	Vehicle Location
248	Bellevue	...	POINT (-122.201905 47.61385)
297	Bellevue	...	POINT (-122.11832 47.6245)
548	Redmond	...	POINT (-122.12302 47.67668)
911	Camas	...	POINT (-122.405565 45.59009)
927	Bellevue	...	POINT (-122.201905 47.61385)

172987	Woodinville	...	POINT (-122.151665 47.75855)
173106	Union Gap	...	POINT (-120.477805 46.553505)
173138	Seattle	...	POINT (-122.382425 47.77279)
173208	Mercer Island	...	POINT (-122.2377542 47.582905)
173311	Mercer Island	...	POINT (-122.2377542 47.582905)

[1130 rows x 9 columns]

Subdataframe da marca POLESTAR:

	City	...	Vehicle Location
422	Snohomish	...	POINT (-122.091505 47.915555)
611	Shoreline	...	POINT (-122.34584 47.76726)
616	Issaquah	...	POINT (-122.03646 47.534065)
1185	Bothell	...	POINT (-122.179458 47.802589)
1246	Spokane	...	POINT (-117.460225 47.64927)



```
172531 Seattle ... POINT (-122.363815 47.63046)
172577 Sammamish ... POINT (-121.9993659 47.5484866)
172881 Kenmore ... POINT (-122.2504747 47.7617128)
173392 Woodinville ... POINT (-122.151665 47.75855)
173435 Renton ... POINT (-122.1298876 47.4451257)
```

[876 rows x 9 columns]

Subdataframe da marca MINI:

	City	...	Vehicle Location
438	Seattle	...	POINT (-122.3185 47.67949)
1159	Fife	...	POINT (-122.36151 47.241885)
1334	Seattle	...	POINT (-122.34584 47.76726)
1838	Normandy Park	...	POINT (-122.341345 47.465925)
2020	Seattle	...	POINT (-122.30764 47.62523)

172280	Shelton	...	POINT (-123.105305 47.211085)
172364	Sammamish	...	POINT (-121.9993659 47.5484866)
172417	Dupont	...	POINT (-122.643815 47.097455)
172889	Seattle	...	POINT (-122.374105 47.54468)
173479	Snohomish	...	POINT (-122.15134 47.8851158)

[890 rows x 9 columns]

Subdataframe da marca FISKER:

	City	...	Vehicle Location
559	Coupeville	...	POINT (-122.6880708 48.2179983)
6446	Vancouver	...	POINT (-122.51692 45.6228)
7049	Edmonds	...	POINT (-122.335685 47.80372)
7644	Seattle	...	POINT (-122.28339 47.549285)
11839	Kirkland	...	POINT (-122.209285 47.71124)
19025	Seattle	...	POINT (-122.38679 47.56484)
20714	Seattle	...	POINT (-122.388675 47.5415)
22636	Kent	...	POINT (-122.235475 47.3809)
23172	North Bend	...	POINT (-121.7814012 47.4935316)
26674	Kent	...	POINT (-122.2012521 47.3931814)
50785	Bow	...	POINT (-122.440636 48.5613885)
68741	Camas	...	POINT (-122.405565 45.59009)
73678	Vancouver	...	POINT (-122.641835 45.638545)
74613	Marysville	...	POINT (-122.17673 48.05542)



74884	Maple Valley	...	POINT	(-122.05191 47.357985)
75704	Bainbridge Island	...	POINT	(-122.5235781 47.6293323)
81068	Mukilteo	...	POINT	(-122.299965 47.94171)
82273	Belfair	...	POINT	(-122.8551647 47.4495785)
82475	Edmonds	...	POINT	(-122.335685 47.80372)
83571	Bellingham	...	POINT	(-122.4569227 48.7470973)
85772	Lake Stevens	...	POINT	(-122.112265 48.0047)
97444	Seattle	...	POINT	(-122.335345 47.61079)
99060	Snohomish	...	POINT	(-122.15134 47.8851158)
101539	Greenacres	...	POINT	(-117.1407 47.673675)
101711	Raymond	...	POINT	(-123.72855 46.686115)
111061	Bellevue	...	POINT	(-122.201905 47.61385)
115027	Sammamish	...	POINT	(-122.0313266 47.6285782)
118256	Friday Harbor	...	POINT	(-123.022255 48.531355)
130972	Snohomish	...	POINT	(-122.091505 47.915555)
138336	Bothell	...	POINT	(-122.179458 47.802589)
142519	Mercer Island	...	POINT	(-122.2377542 47.582905)
145877	Poulsbo	...	POINT	(-122.64177 47.737525)
147969	Seattle	...	POINT	(-122.30839 47.610365)
157890	Ravensdale	...	POINT	(-122.05191 47.357985)
162723	Seattle	...	POINT	(-122.30764 47.62523)

[35 rows x 9 columns]

Subdataframe da marca GENESIS:

	City	...	Vehicle Location
568	Seattle	...	POINT (-122.34848 47.632405)
3739	Bellevue	...	POINT (-122.11832 47.6245)
4121	Bainbridge Island	...	POINT (-122.5235781 47.6293323)
4806	Longview	...	POINT (-122.9379953 46.1372997)
6324	Lake Stevens	...	POINT (-122.112265 48.0047)

164789	Kirkland	...	POINT (-122.209285 47.71124)
165717	Duvall	...	POINT (-121.9810747 47.7377962)
171856	Stanwood	...	POINT (-122.3684051 48.2414921)
172320	Seattle	...	POINT (-122.394185 47.639195)
172981	Dupont	...	POINT (-122.643815 47.097455)

[173 rows x 9 columns]



Subdataframe da marca CADILLAC:

	City	...	Vehicle Location
663	Puyallup	...	POINT (-122.275748 47.1395924)
1289	Brush Prairie	...	POINT (-122.5485715 45.7336587)
1521	Puyallup	...	POINT (-122.28718 47.190465)
1775	Lynnwood	...	POINT (-122.2551991 47.8650827)
2007	Kent	...	POINT (-122.111625 47.36078)

168916	Renton	...	POINT (-122.15734 47.487175)
169355	Marysville	...	POINT (-122.1713847 48.10433)
169985	Shelton	...	POINT (-123.105305 47.211085)
171828	Seattle	...	POINT (-122.34584 47.76726)
173109	Sammamish	...	POINT (-122.03309 47.58153)

[307 rows x 9 columns]

Subdataframe da marca MAZDA:

	City	...	Vehicle Location
745	Snoqualmie	...	POINT (-121.8740496 47.5345546)
1134	Issaquah	...	POINT (-121.9993659 47.5484866)
1199	Bellevue	...	POINT (-122.16085 47.624515)
1286	Kennewick	...	POINT (-119.1973001 46.1911488)
1546	Shoreline	...	POINT (-122.34584 47.76726)

172292	Redmond	...	POINT (-122.12302 47.67668)
172299	Seattle	...	POINT (-122.34301 47.659185)
172493	Seattle	...	POINT (-122.3185 47.67949)
172555	Redmond	...	POINT (-122.12302 47.67668)
172611	Bothell	...	POINT (-122.1873 47.820245)

[431 rows x 9 columns]

Subdataframe da marca LINCOLN:

	City	...	Vehicle Location
871	Puyallup	...	POINT (-122.275748 47.1395924)
1879	Edmonds	...	POINT (-122.37507 47.80807)
2161	Camas	...	POINT (-122.405565 45.59009)
3111	Shoreline	...	POINT (-122.3175 47.7578146)
3430	Longview	...	POINT (-122.9379953 46.1372997)



169937	Olympia	...	POINT (-122.89692 47.043535)
169993	Mill Creek	...	POINT (-122.1873 47.820245)
170153	Lake Stevens	...	POINT (-122.112265 48.0047)
171332	Custer	...	POINT (-122.6410958 48.919121)
173043	Redmond	...	POINT (-122.12302 47.67668)

[259 rows x 9 columns]

Subdataframe da marca ALFA ROMEO:

	City	...	Vehicle Location
3163	Seattle	...	POINT (-122.30839 47.610365)
8554	Kirkland	...	POINT (-122.20264 47.6785)
14687	Redmond	...	POINT (-122.0222799 47.6958998)
15922	Marysville	...	POINT (-122.17673 48.05542)
16514	Kirkland	...	POINT (-122.20264 47.6785)
30467	Yarrow Point	...	POINT (-122.201905 47.61385)
30773	Seattle	...	POINT (-122.344125 47.61546)
33348	Seattle	...	POINT (-122.34301 47.659185)
39000	Seattle	...	POINT (-122.30823 47.581975)
40541	Kingston	...	POINT (-122.50156 47.8019)
41586	Mukilteo	...	POINT (-122.299965 47.94171)
55156	Gig Harbor	...	POINT (-122.5835454 47.3234488)
57790	Mukilteo	...	POINT (-122.299965 47.94171)
66387	Monroe	...	POINT (-121.972215 47.85674)
68052	Seattle	...	POINT (-122.34848 47.632405)
70337	Woodinville	...	POINT (-122.151665 47.75855)
84931	Lacey	...	POINT (-122.7474291 47.0821119)
85174	Puyallup	...	POINT (-122.3085456 47.1042426)
92255	Seattle	...	POINT (-122.34848 47.632405)
93401	Kirkland	...	POINT (-122.20264 47.6785)
100813	Bellevue	...	POINT (-122.16085 47.624515)
110403	Mukilteo	...	POINT (-122.299965 47.94171)
111687	Bellingham	...	POINT (-122.45493 48.76809)
112391	Seattle	...	POINT (-122.382425 47.77279)
120571	Woodinville	...	POINT (-122.151665 47.75855)
121332	Bothell	...	POINT (-122.20578 47.762405)
127751	Mercer Island	...	POINT (-122.2377542 47.582905)
134450	Bainbridge Island	...	POINT (-122.5235781 47.6293323)
137624	Bothell	...	POINT (-122.1873 47.820245)
137891	Yakima	...	POINT (-120.6027202 46.5965625)



148429	Bothell ...	POINT (-122.179458 47.802589)
149661	Sammamish ...	POINT (-122.03309 47.58153)
150234	Sammamish ...	POINT (-122.0313266 47.6285782)
150550	Redmond ...	POINT (-122.12302 47.67668)
170546	Bothell ...	POINT (-122.179458 47.802589)

[35 rows x 9 columns]

Subdataframe da marca LAND ROVER:

	City ...	Vehicle Location
3937	Orondo ...	POINT (-120.19519 47.704075)
5239	Woodinville ...	POINT (-122.151665 47.75855)
11852	Seattle ...	POINT (-122.30839 47.610365)
12789	Bothell ...	POINT (-122.1873 47.820245)
12831	Seattle ...	POINT (-122.335345 47.61079)
13870	Sammamish ...	POINT (-122.0313266 47.6285782)
14445	Seattle ...	POINT (-122.34848 47.632405)
20271	Bonney Lake ...	POINT (-122.183805 47.18062)
23721	Sammamish ...	POINT (-122.03309 47.58153)
28723	Sammamish ...	POINT (-122.0313266 47.6285782)
31605	Yakima ...	POINT (-120.6027202 46.5965625)
33517	Anacortes ...	POINT (-122.615305 48.501275)
36743	Bellevue ...	POINT (-122.16085 47.624515)
38539	Issaquah ...	POINT (-121.9993659 47.5484866)
40999	Gig Harbor ...	POINT (-122.589645 47.342345)
41083	Mercer Island ...	POINT (-122.2377542 47.582905)
46080	Seattle ...	POINT (-122.296385 47.71558)
46636	Camas ...	POINT (-122.405565 45.59009)
55273	Olympia ...	POINT (-122.8285 47.03646)
55869	Seattle ...	POINT (-122.34584 47.76726)
57865	Bellevue ...	POINT (-122.201905 47.61385)
59481	Sammamish ...	POINT (-122.03309 47.58153)
61977	Wenatchee ...	POINT (-120.32009 47.42255)
63082	Kirkland ...	POINT (-122.20264 47.6785)
65459	Mead ...	POINT (-117.35761 47.76885)
67512	Spokane ...	POINT (-117.431895 47.667155)
70438	Woodinville ...	POINT (-122.151665 47.75855)
71500	Burien ...	POINT (-122.341345 47.465925)
76179	Seattle ...	POINT (-122.28339 47.549285)
76720	Seattle ...	POINT (-122.234385 47.494545)



77760	Seattle	...	POINT (-122.3185 47.67949)
78875	Ravensdale	...	POINT (-121.98104 47.358625)
80651	Vancouver	...	POINT (-122.70302 45.703706)
83722	Auburn	...	POINT (-122.2849393 47.3384055)
88052	North Bend	...	POINT (-121.7814012 47.4935316)
92265	South Hill	...	POINT (-122.3085456 47.1042426)
101399	Liberty Lake	...	POINT (-117.0923638 47.6643385)
103381	Redmond	...	POINT (-122.0222799 47.6958998)
105130	Seattle	...	POINT (-122.30764 47.62523)
106973	Gig Harbor	...	POINT (-122.5835454 47.3234488)
111838	Bellingham	...	POINT (-122.486115 48.761615)
113074	Kirkland	...	POINT (-122.20264 47.6785)
115481	Friday Harbor	...	POINT (-123.022255 48.531355)
117809	Bellevue	...	POINT (-122.16937 47.571015)
120193	Ocean Shores	...	POINT (-124.1599804 47.0075271)
120949	Redmond	...	POINT (-122.12302 47.67668)
126560	Hunts Point	...	POINT (-122.201905 47.61385)
126860	Carnation	...	POINT (-121.9105947 47.6483005)
133910	Issaquah	...	POINT (-121.9993659 47.5484866)
139120	Seattle	...	POINT (-122.34301 47.659185)
156877	Gig Harbor	...	POINT (-122.6657985 47.383359)
162006	Sammamish	...	POINT (-122.0313266 47.6285782)
163583	Bellingham	...	POINT (-122.45493 48.76809)

[53 rows x 9 columns]

Subdataframe da marca GMC:

	City	...	Vehicle Location
5965	Tukwila	...	POINT (-122.286465 47.476)
52925	Edmonds	...	POINT (-122.335685 47.80372)

[2 rows x 9 columns]

Subdataframe da marca TH!NK:

	City	...	Vehicle Location
9514	Battle Ground	...	POINT (-122.53218 45.77945)
95687	Bellingham	...	POINT (-122.486115 48.761615)
108207	Bellingham	...	POINT (-122.486115 48.761615)
136800	Yacolt	...	POINT (-122.4066726 45.8651816)
151175	Redmond	...	POINT (-122.12302 47.67668)



[5 rows x 9 columns]

Subdataframe da marca DODGE:

	City	...	Vehicle Location
15351	Bainbridge Island	...	POINT (-122.5235781 47.6293323)
19142	Tukwila	...	POINT (-122.29179 47.43473)
19445	Tukwila	...	POINT (-122.29179 47.43473)
20615	Tukwila	...	POINT (-122.29179 47.43473)
21779	Tukwila	...	POINT (-122.29179 47.43473)

137409	Kenmore	...	POINT (-122.2504747 47.7617128)
139182	Olympia	...	POINT (-122.9131017 47.0135926)
145463	Seattle	...	POINT (-122.388675 47.5415)
147509	Olympia	...	POINT (-122.89692 47.043535)
164623	Puyallup	...	POINT (-122.3085456 47.1042426)

[381 rows x 9 columns]

Subdataframe da marca WHEEGO ELECTRIC CARS:

	City	...	Vehicle Location
20635	Tacoma	...	POINT (-122.490985 47.26365)
93730	Olympia	...	POINT (-122.89692 47.043535)
126322	Spokane	...	POINT (-117.369705 47.62637)

[3 rows x 9 columns]

Subdataframe da marca BENTLEY:

	City	...	Vehicle Location
21432	Yakima	...	POINT (-120.500225 46.6043)
32699	Mukilteo	...	POINT (-122.299965 47.94171)
35050	Seattle	...	POINT (-122.329815 47.57981)

[3 rows x 9 columns]

Subdataframe da marca AZURE DYNAMICS:

	City	...	Vehicle Location
34226	Seattle	...	POINT (-122.34301 47.659185)
96828	Seattle	...	POINT (-122.3268963 47.5499519)
109259	Lummi Island	...	POINT (-122.6888403 48.7199947)



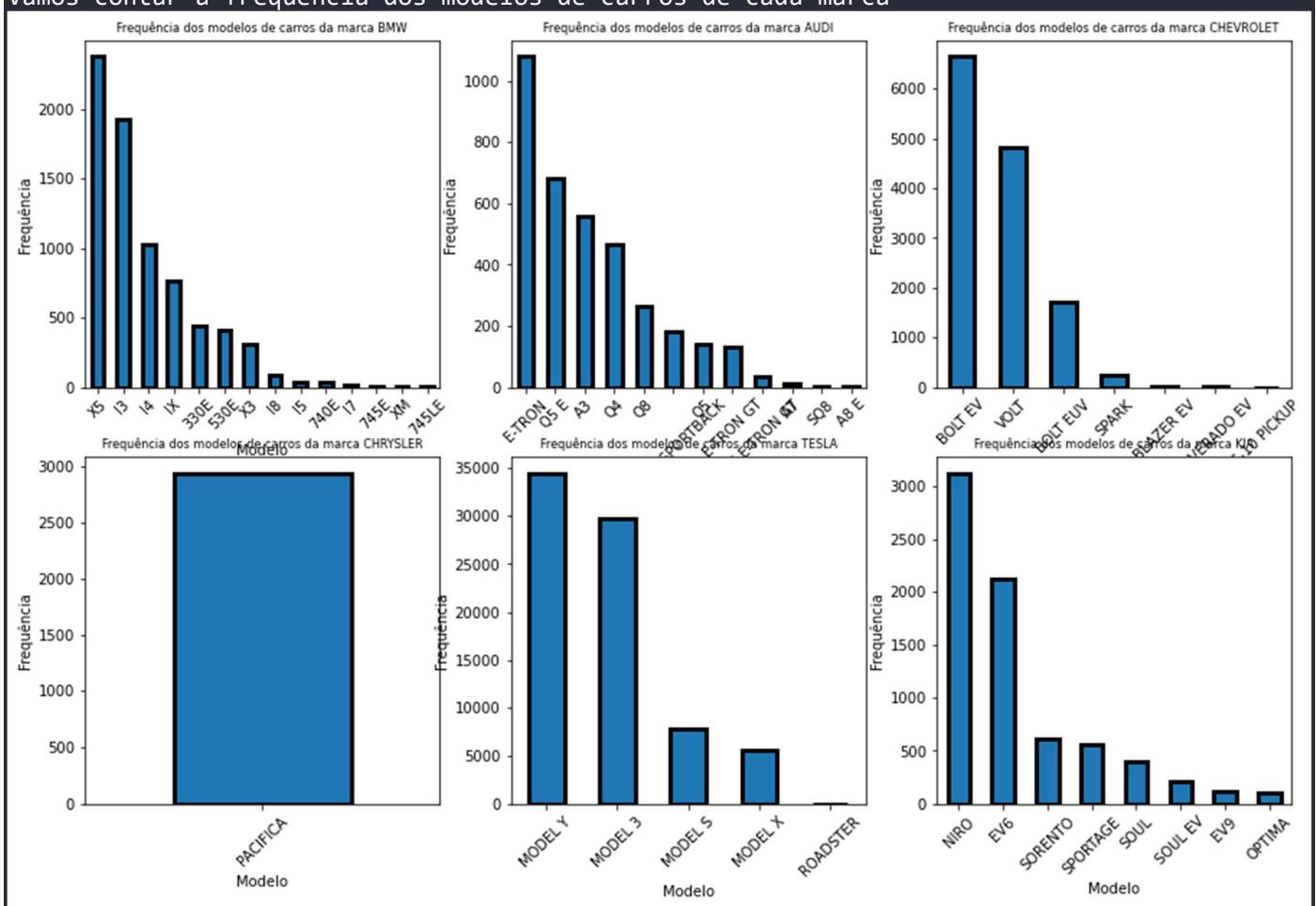
111673 Bainbridge Island ... POINT (-122.5235781 47.6293323)
113920 Enumclaw ... POINT (-121.98953 47.20347)
151477 Bellingham ... POINT (-122.4569227 48.7470973)
158723 Kennewick ... POINT (-119.14533 46.187395)

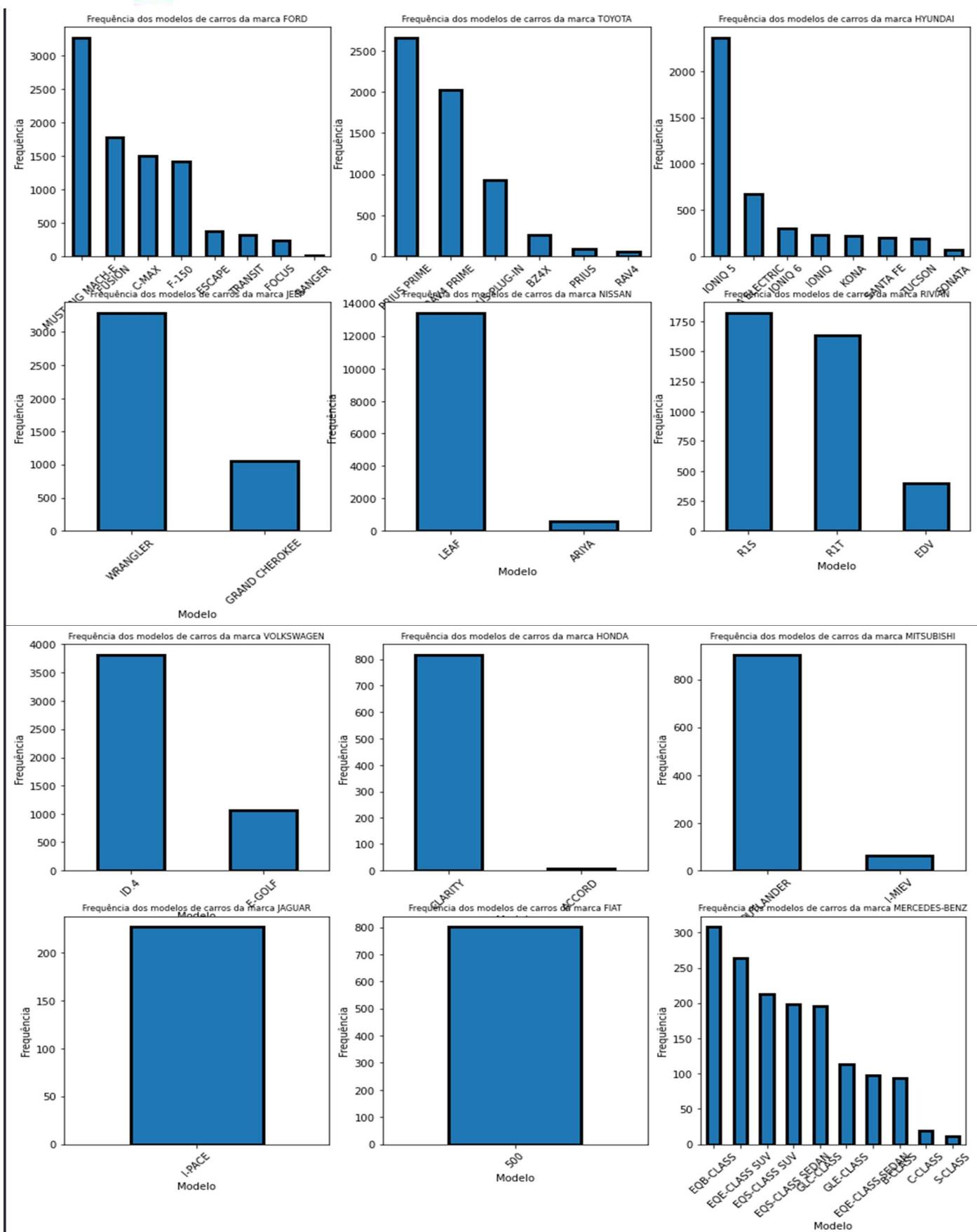
[7 rows x 9 columns]

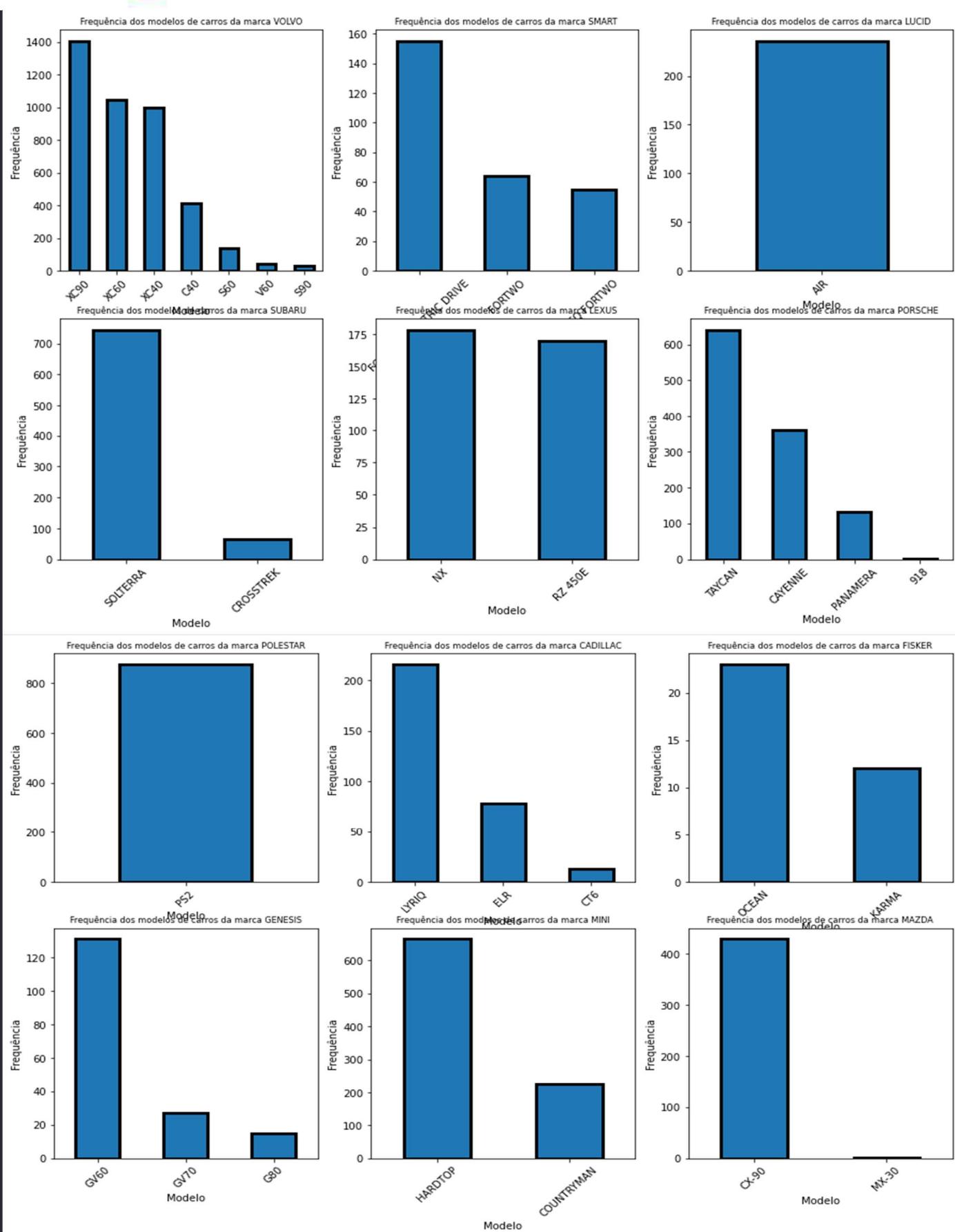
Subdataframe da marca ROLLS ROYCE:

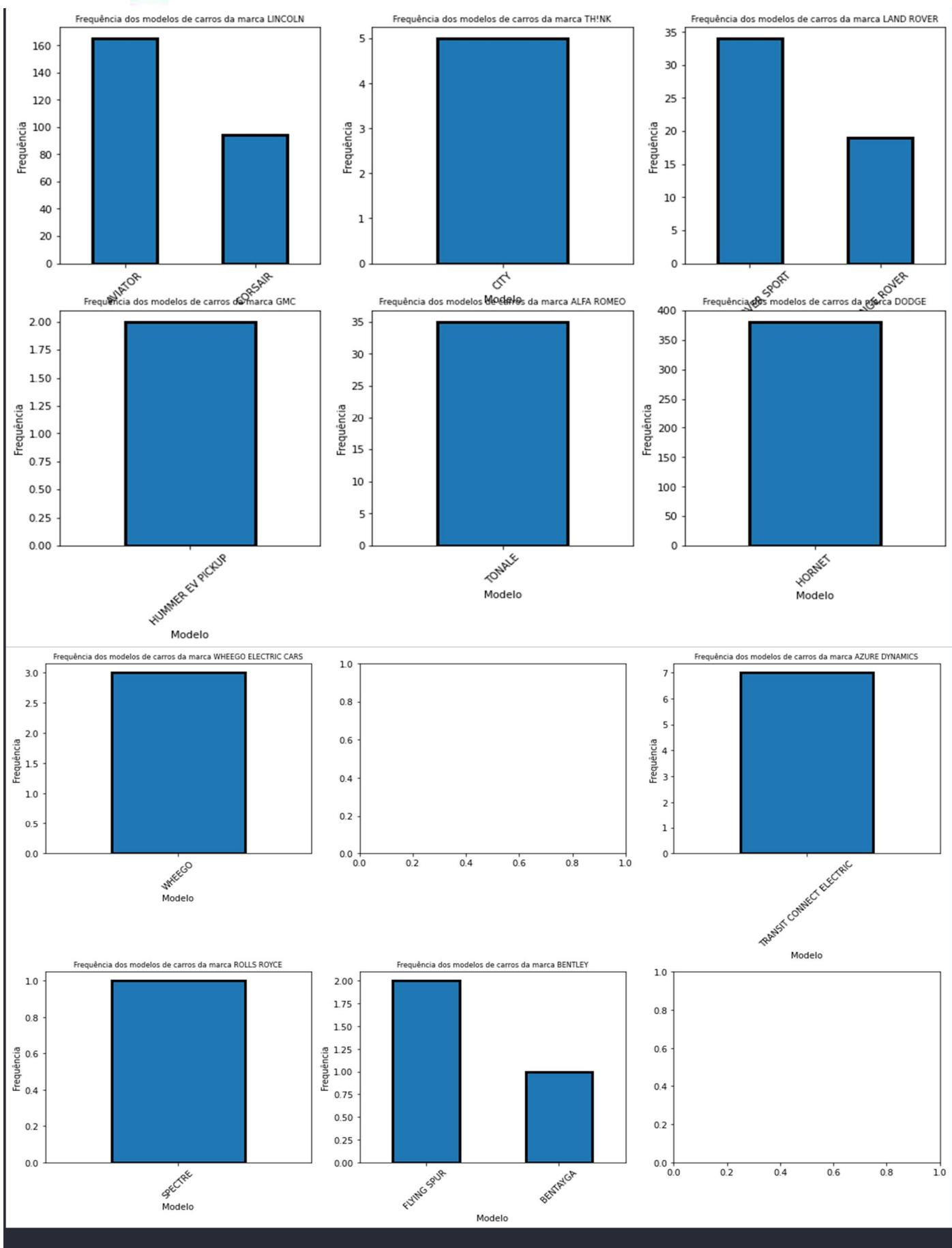
City State ...	Electric Utility	Vehicle Location
101417 Langley WA ... PUGET SOUND ENERGY INC	POINT (-122.408015 48.03557)	

Geramos graficos automatizados para todos os subdataframes para analise exploratoria. Vamos contar a frequencia dos modelos de carros de cada marca











Visualisando os dados entendemos ser necessária a separação entre carros elétricos 'BEV' e Híbridos 'PHEV'

Significado das siglas BEV e PHEV

```
#BEV(Battery Electric Vehicle): Um BEV é um veiculo elétrico que é alimentado  
#exclusivamente por bateria. Isso significa que ele não possui  
#um motor de combustão interna e depende apenas de uma bateria  
#recarregável para fornecer energia.
```

```
#PHEV(Plug-in-Hybrid Electric Vehicle): Um PHEV é um tipo de  
#veiculo eletrico que possui tanto um motor eletrico quanto  
#um motor a combustão interna. Esses veiculos sao equipados  
#com uma bateria recarregável que alimenta o motor elétrico  
#e tambem possuem um tanque de combustivel para alimentar o motor  
#a combustao
```

```
# Iterando sobre os subdatasets para contar o número de BEVs e PHEVs  
for marca, sub_df in subdataframes.items():  
    # Contando o número de BEVs e PHEVs em cada marca  
    contagem_ev_type = sub_df['Electric Vehicle Type'].value_counts()
```

```
# Exibindo as contagens de BEVs e PHEVs  
print(f"\nMarca: {marca}")  
print("Contagem de Electric Vehicle Type:")  
print(contagem_ev_type)
```

```
# Defina o número de gráficos por página  
graficos_por_pagina = 6
```

```
# Tamanho da fonte do título do gráfico  
tamanho_fonte_titulo = 8
```

```
# Inicialize uma variável para contar os gráficos  
contador_graficos = 0
```

```
# Inicialize uma variável para contar as páginas  
contador_paginas = 0
```

```
# Iterando sobre os subdatasets para gerar gráficos  
for marca, sub_df in subdataframes.items():  
    # Contando o número de BEVs e PHEVs em cada marca  
    contagem_ev_type = sub_df['Electric Vehicle Type'].value_counts()
```

```
# Verificando se é necessário criar uma nova página  
if contador_graficos % graficos_por_pagina == 0:  
    # Criando uma nova figura e eixos  
    fig, axs = plt.subplots(2, 3, figsize=(15, 10))
```

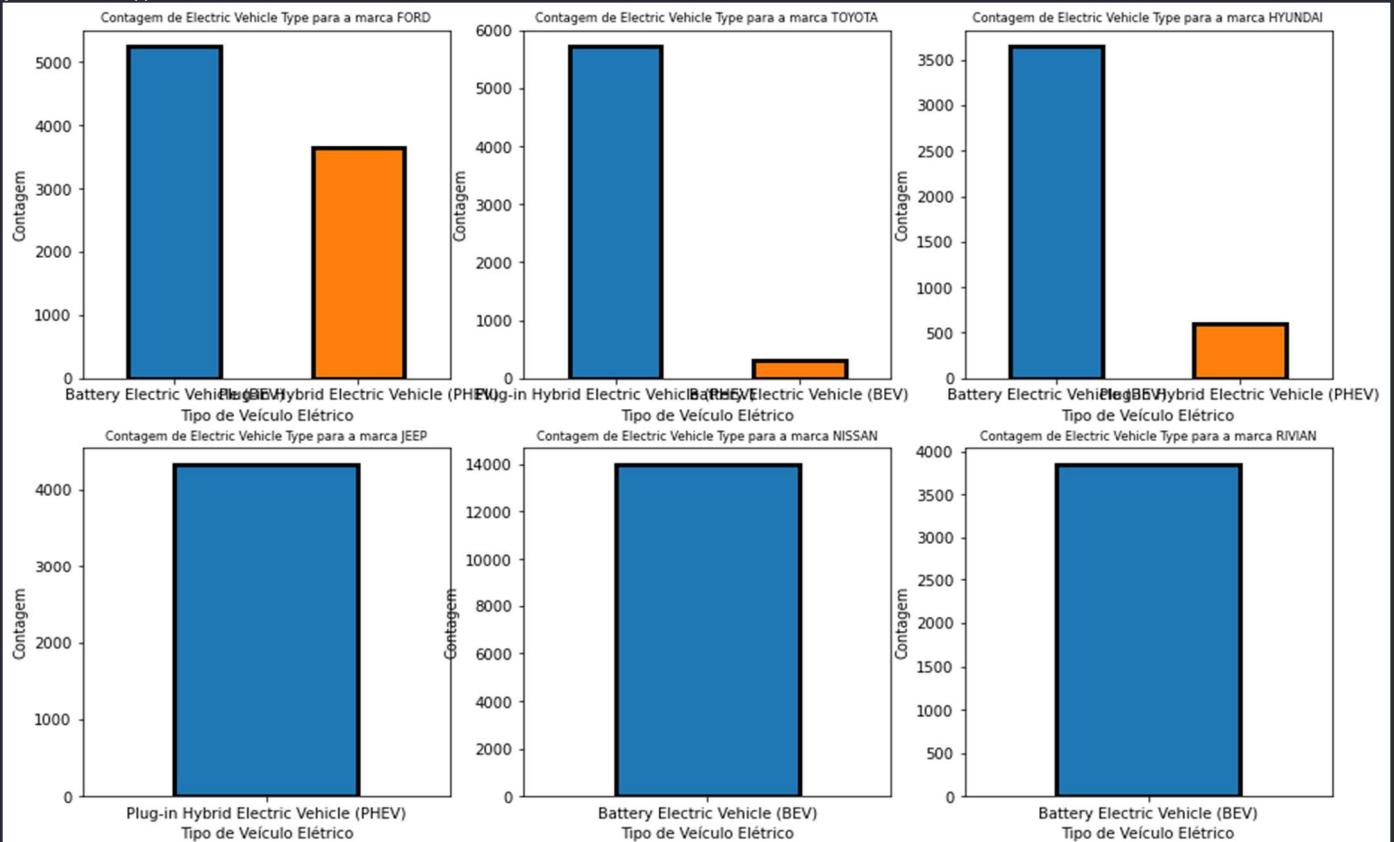


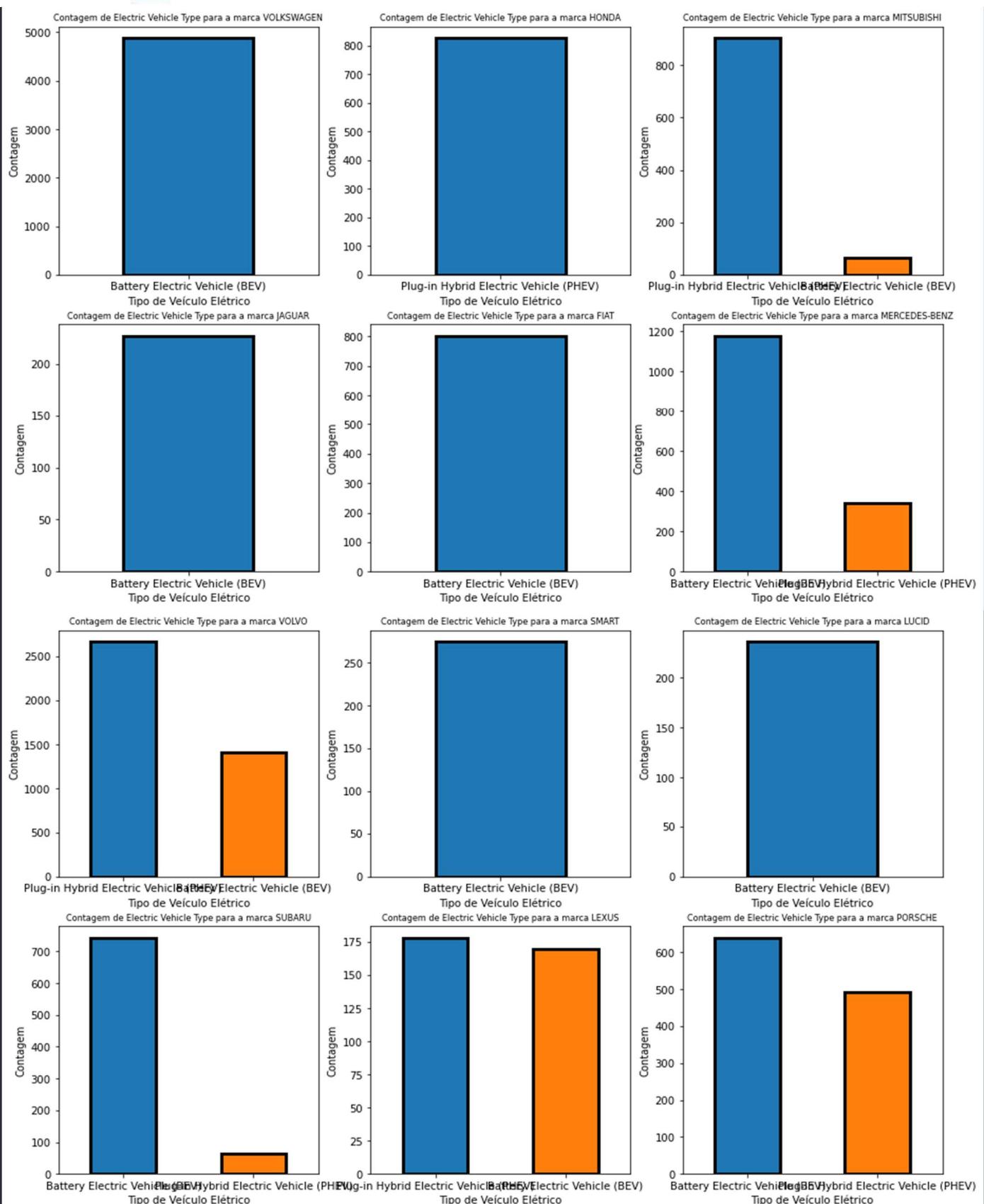
```
contador_paginas += 1
```

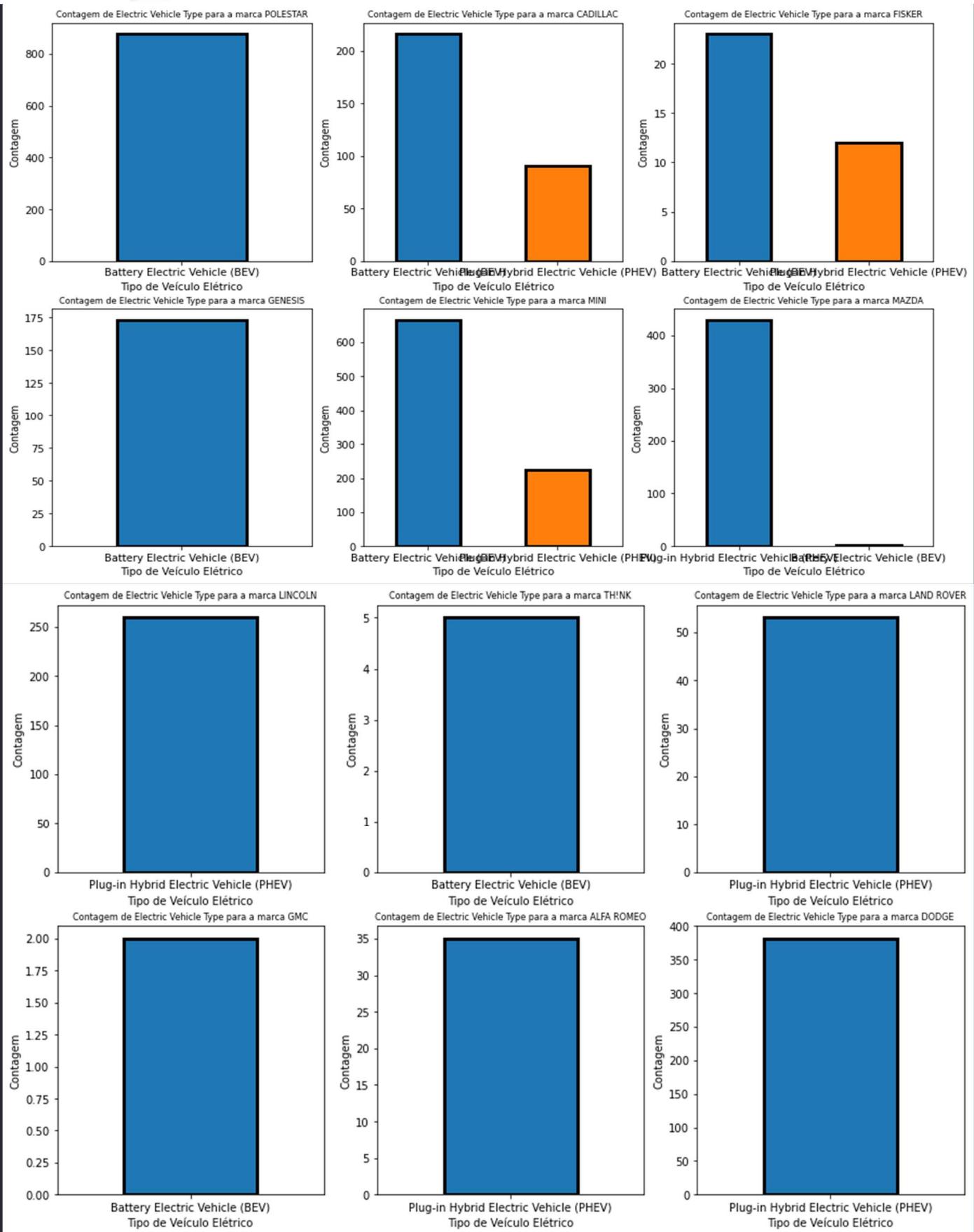
```
# Criando o gráfico de barras
linha = contador_graficos % (graficos_por_pagina // 3)
coluna = contador_graficos % 3
ax = axs[linha, coluna]
contagem_ev_type.plot(kind='bar', color=['#1f77b4', '#ff7f0e'], edgecolor='black',
linewidth=3, ax=ax)
ax.set_title(f'Contagem de Electric Vehicle Type para a marca {marca}', fontsize=tamanho_fonte_titulo)
ax.set_xlabel('Tipo de Veículo Elétrico')
ax.set_ylabel('Contagem')
ax.tick_params(axis='x', rotation=0)

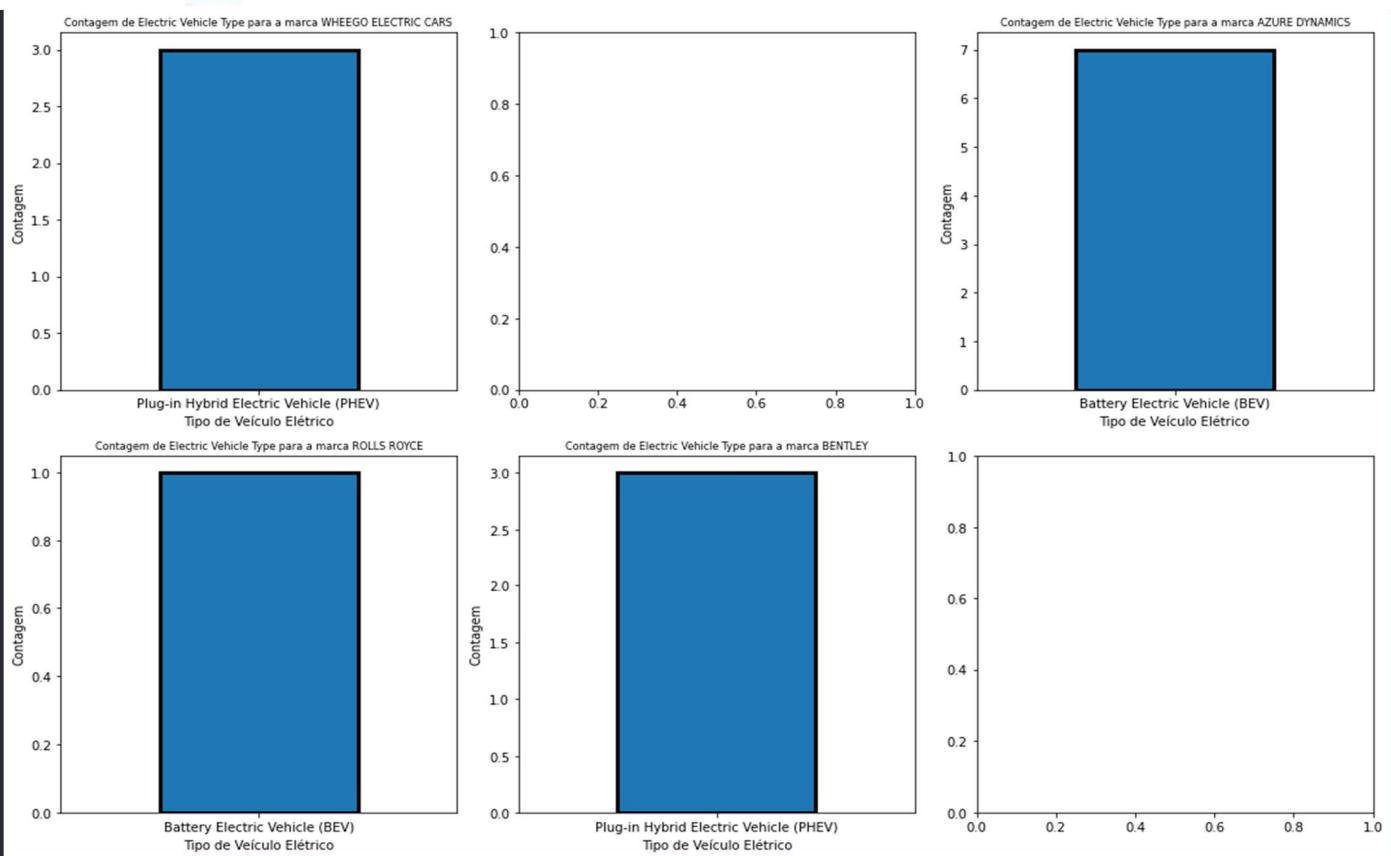
# Incrementando o contador de gráficos
contador_graficos += 1

# Ajustando o layout da última página de gráficos
plt.tight_layout()
# Exibindo a última página de gráficos
plt.show()
```









Agora sabemos que o foco do nosso projeto será os 'BEV' - Veículos Elétricos, encontramos nosso Target.

```
import matplotlib.pyplot as plt
```

```
# Defina o número de gráficos por página
graficos_por_pagina = 6

# Tamanho da fonte do título do gráfico
tamanho_fonte_titulo = 8

# Dicionário para armazenar o número de veículos BEV por marca
num_bev_por_marca = {}

# Iterar sobre os subdatasets originais
for marca, sub_df in subdataframes.items():
    # Filtrar o subdataset para selecionar apenas os carros do tipo BEV
    sub_df_bev = sub_df[sub_df['Electric Vehicle Type'] == 'Battery Electric Vehicle (BEV)']

    # Armazenar o número de veículos BEV por marca
    num_bev = len(sub_df_bev)

    # Adicionar a marca e o número de veículos ao dicionário apenas se o número for diferente de 0
    if num_bev != 0:
        num_bev_por_marca[marca] = num_bev
```

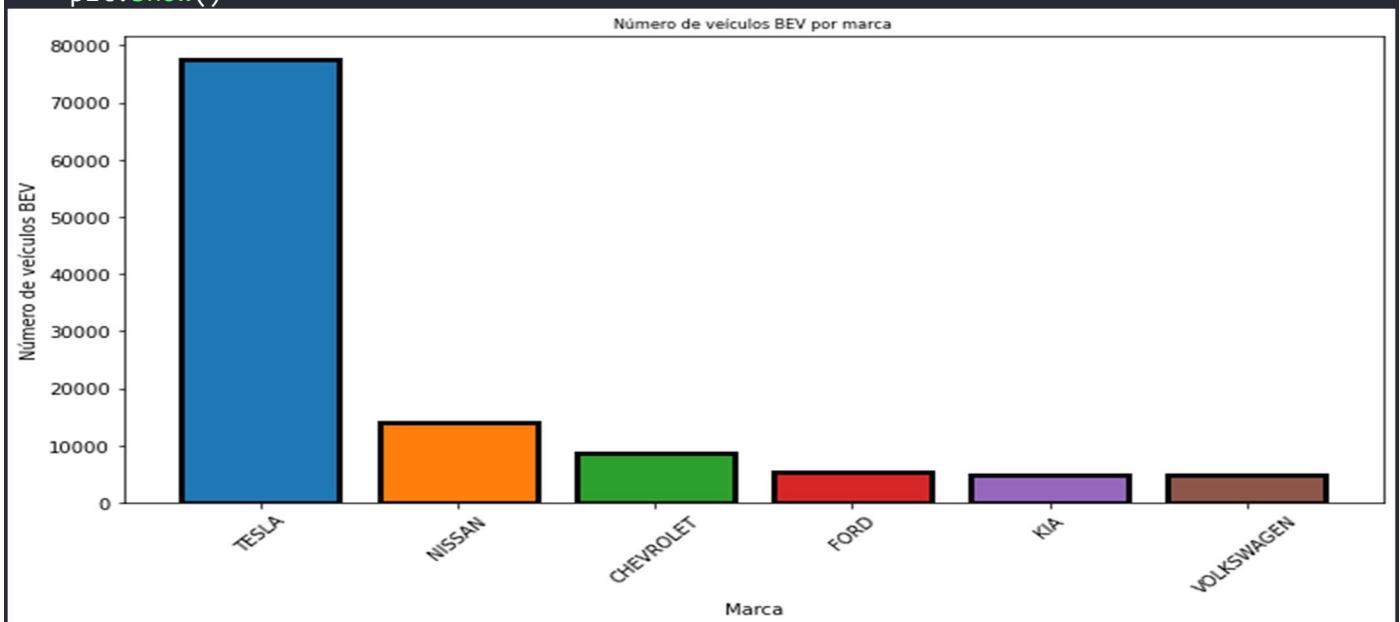


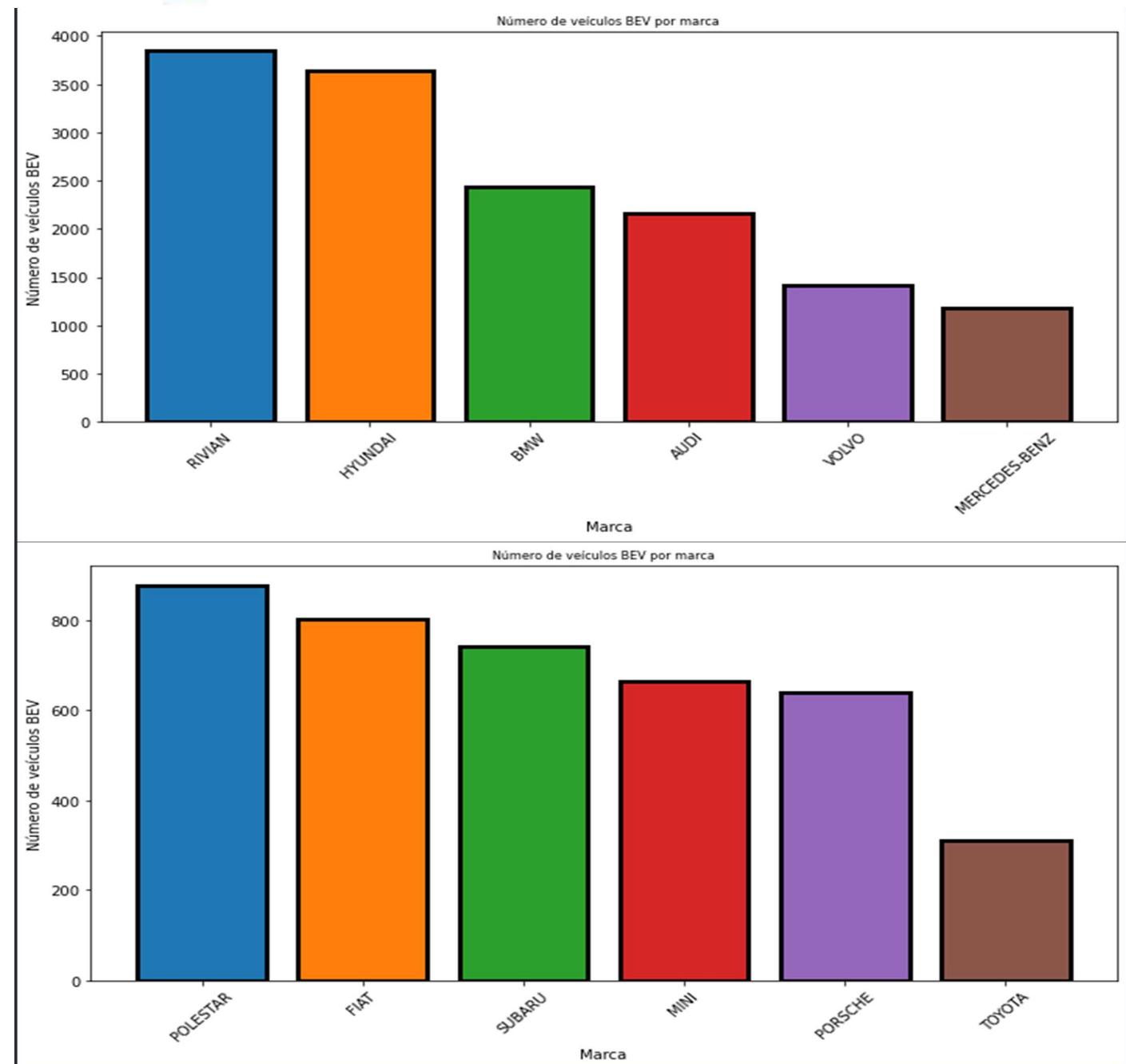
```
# Ordenar o dicionário num_bev_por_marca em ordem decrescente por valor
num_bev_por_marca_ordenado = sorted(num_bev_por_marca.items(), key=lambda x: x[1], reverse=True)

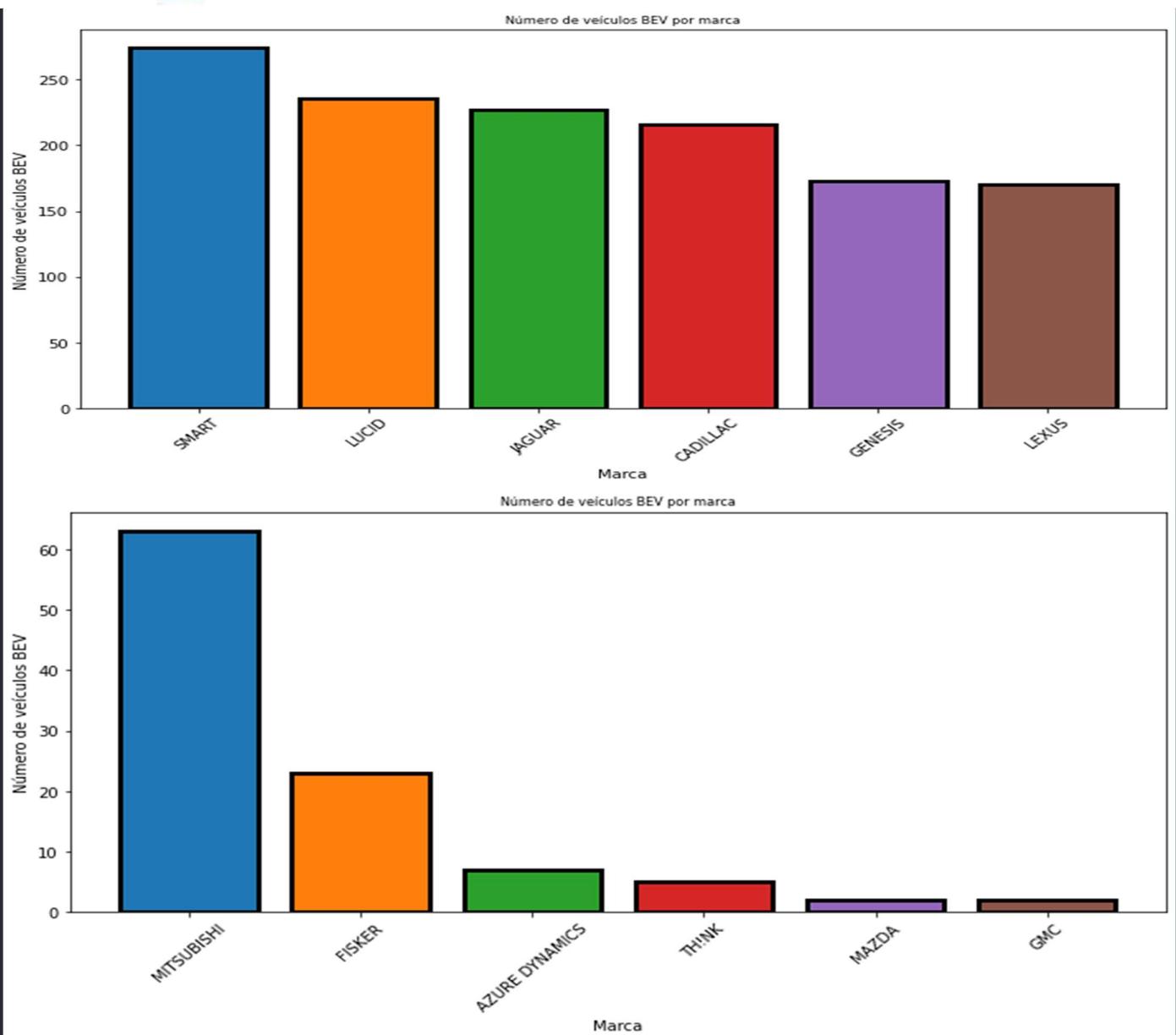
# Cores para as barras
cores = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b', '#e377c2',
 '#7f7f7f', '#bcbd22', '#17becf']

# Dividir o dicionário ordenado em lotes de tamanho graficos_por_pagina
lotes = [num_bev_por_marca_ordenado[i:i+graficos_por_pagina] for i in range(0,
len(num_bev_por_marca_ordenado), graficos_por_pagina)]

# Iterar sobre os lotes para criar e exibir gráficos
for lote in lotes:
    # Criar o gráfico de barras
    plt.figure(figsize=(10, 6))
    for i, (marca, num_bev) in enumerate(lote):
        plt.bar(marca, num_bev, color=cores[i], edgecolor='black', linewidth=3)
    plt.title('Número de veículos BEV por marca', fontsize=tamanho_fonte_titulo)
    plt.xlabel('Marca')
    plt.ylabel('Número de veículos BEV')
    plt.xticks(rotation=45)
    plt.tight_layout()
    # Exibir o gráfico
    plt.show()
```







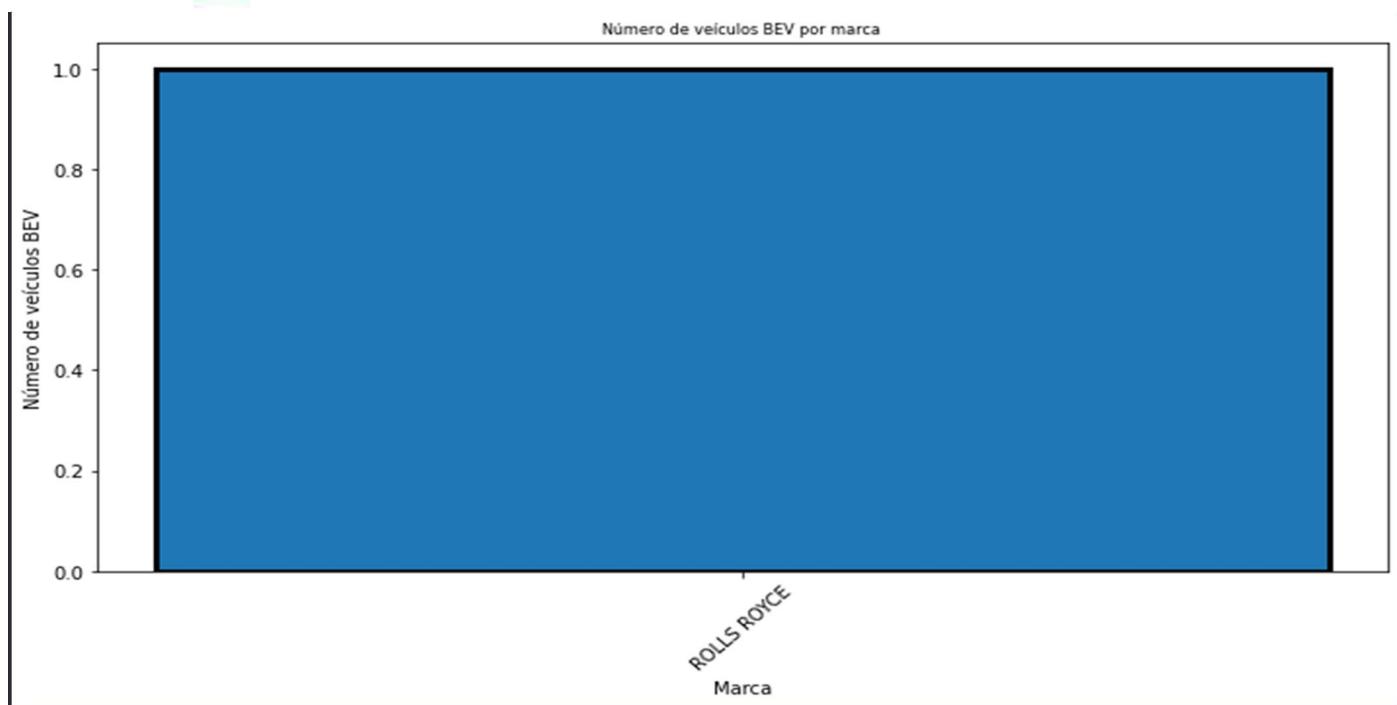


Tabela com o número de veículos BEV por marca e a quantidade total:

	Marca	Número de veículos BEV
0	TESLA	77624
1	NISSAN	13992
2	CHEVROLET	8628
3	FORD	5241
4	KIA	4889
5	VOLKSWAGEN	4876
6	RIVIAN	3848
7	HYUNDAI	3641
8	BMW	2442
9	AUDI	2159
10	VOLVO	1415
11	MERCEDES-BENZ	1176
12	POLESTAR	876
13	FIAT	801
14	SUBARU	742
15	MINI	665
16	PORSCHE	639
17	TOYOTA	311
18	SMART	274
19	LUCID	236
20	JAGUAR	227
21	CADILLAC	216
22	GENESIS	173
23	LEXUS	170
24	MITSUBISHI	63
25	FISKER	23
26	AZURE DYNAMICS	7
27	TH!NK	5



```
28      MAZDA          2
29      GMC            2
30  ROLLS ROYCE       1
31  Total           135364
```

Separamos nosso Target

Selecionamos e trabalhamos as colunas que queríamos extrair as informações
colunas_selecionadas = df.copy()

```
colunas_selecionadas = colunas_selecionadas[['State','Groups With Access Code','EV Level1
EVSE Num', 'EV Level2 EVSE Num', 'EV DC Fast Count', 'Latitude', 'Longitude']]
```

```
# Crie um novo DataFrame contendo apenas as linhas onde a coluna 'category' é igual a 'pu-
blic'
```

```
colunas_selecionadas_public = colunas_selecionadas[colunas_selecionadas['Groups With Ac-
cess Code'] == 'Public']
```

```
# Crie um novo DataFrame contendo apenas as linhas onde a coluna 'State' é igual a 'WA'
colunas_selecionadas_public_wa = colunas_selecionadas_public[colunas_selecionadas_pu-
blic['State'] == 'WA']
```

```
# Crie uma cópia do DataFrame original para manter as outras colunas inalteradas
df_filtered = colunas_selecionadas_public_wa.copy()
```

```
# Selecione apenas as colunas específicas para verificar se todas estão preenchidas com
NaN
```

```
columns_to_check = ['EV Level1 EVSE Num', 'EV Level2 EVSE Num', 'EV DC Fast Count']
```

```
# Remova as linhas onde todas as três colunas especificadas estão preenchidas com NaN
df_filtered = df_filtered.dropna(subset=columns_to_check, how='all')
```

```
df_filtered[['EV Level1 EVSE Num', 'EV Level2 EVSE Num', 'EV DC Fast Count']] = df_fil-
tered[['EV Level1 EVSE Num', 'EV Level2 EVSE Num', 'EV DC Fast Count']].fillna(0)
```

```
colunas_selecionadas_preenchidas = df_filtered.copy()
```

```
df = pd.read_csv('D:/OneDrive - Instituto Presbiteriano Mackenzie/Aulas/3 semes-
tre/Projeto Aplicado II/alt_fuel_stations (Mar 18 2024).csv')
```

Selecionamos e trabalhamos as colunas que queríamos extrair as informações

Criamos graficos para analise exploratoria e entender como estava a distribuicao por capa-
cidade dos pontos de recarga

```
# Selecione apenas as colunas desejadas para calcular a soma
```

```
colunas_selecionadas_soma = ['EV Level1 EVSE Num', 'EV Level2 EVSE Num', 'EV DC Fast
Count']
```

```
# Calcule a soma das instâncias das colunas selecionadas
```



```
soma_instancias = df_filtered[colunas_selecionadas_soma].sum()

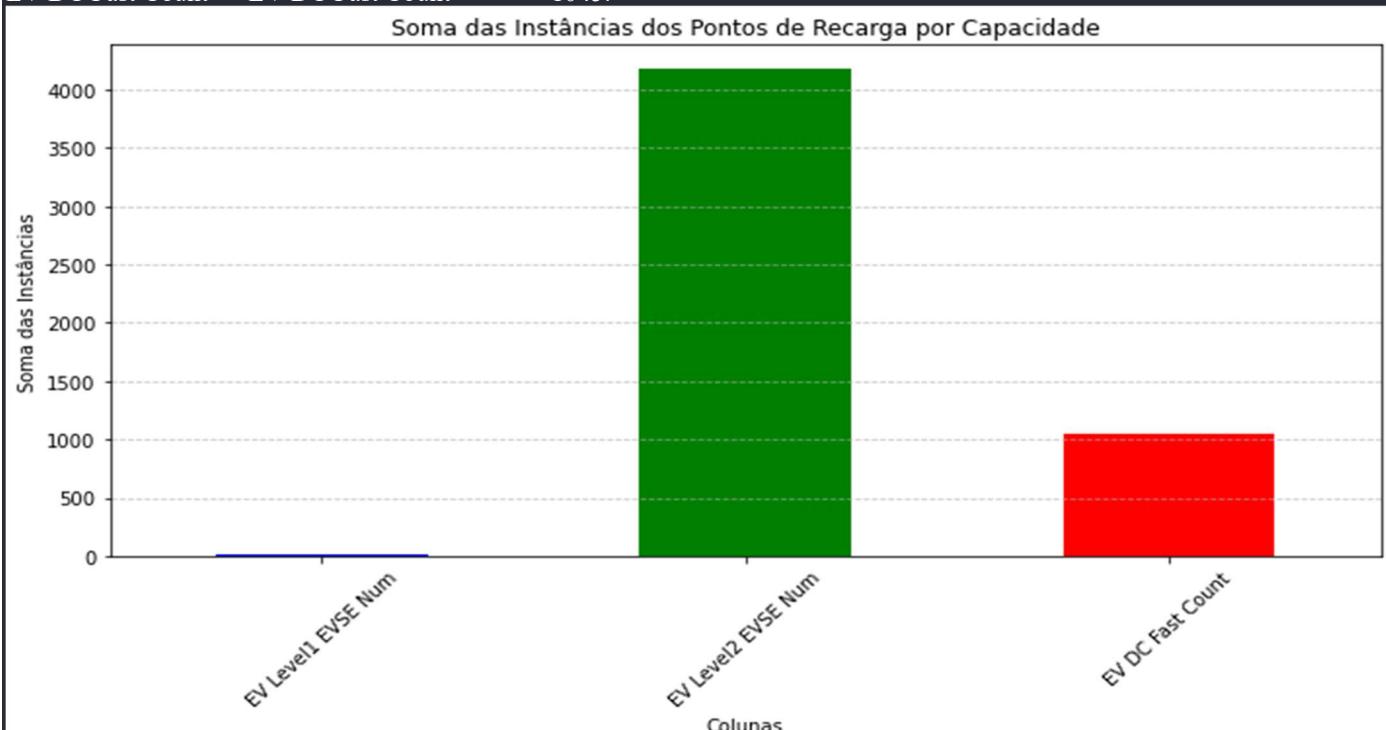
# Crie uma tabela com o resultado da soma
tabela_soma = pd.DataFrame({'Colunas': colunas_selecionadas_soma, 'Soma das Instâncias': soma_instancias})

# Crie o gráfico de barras coloridas com base nas somas das instâncias
plt.figure(figsize=(10, 6))
soma_instancias.plot(kind='bar', color=['blue', 'green', 'red'])
plt.title('Soma das Instâncias dos Pontos de Recarga por Capacidade')
plt.xlabel('Colunas')
plt.ylabel('Soma das Instâncias')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

# Exiba a tabela com o resultado da soma
print("Tabela com o resultado da soma das instâncias das colunas selecionadas:")
print(tabela_soma)
```

Tabela com o resultado da soma das instâncias das colunas selecionadas:

Colunas	Soma das Instâncias
EV Level1 EVSE Num	17.0
EV Level2 EVSE Num	4175.0
EV DC Fast Count	1045.



selecionamos as coordenadas GPS Para plotarmos em um grafico, dos dois banco de dados

```
colunas_gps = colunas_selecionadas_preenchidas[['Latitude', 'Longitude']]
```



```
import pandas as pd
import re

# Função para extrair latitude e longitude de uma string no formato POINT (longitude latitude)
def extrair_latitude_longitude(point_string):
    # Use uma expressão regular para extrair os valores numéricos
    matches = re.findall(r"-?\d+\.\d+", point_string)
    if len(matches) >= 2:
        return float(matches[1]), float(matches[0]) # A ordem é invertida para corresponder ao formato (latitude, longitude)
    else:
        return None, None

seu_dataframe = base_dados
# Iterar sobre as linhas do dataframe e extrair latitude e longitude
dados_selecionados = []
for index, linha in seu_dataframe.iterrows():
    localizacao_str = linha['Vehicle Location']
    latitude, longitude = extrair_latitude_longitude(localizacao_str)
    if latitude is not None and longitude is not None:
        dados_selecionados.append({'Latitude': latitude, 'Longitude': longitude})

# Imprimir os dados
for dado in dados_selecionados:
    print(dado)

gps_selecionado_carros = dado

{'Latitude': 47.820245, 'Longitude': -122.1873}
{'Latitude': 47.241885, 'Longitude': -122.36151}
{'Latitude': 45.67862, 'Longitude': -122.5146473}
{'Latitude': 47.500055, 'Longitude': -122.180505}
{'Latitude': 45.7010427, 'Longitude': -122.6483953}
{'Latitude': 47.2923828, 'Longitude': -122.5113356}
{'Latitude': 47.476, 'Longitude': -122.286465}
{'Latitude': 47.342345, 'Longitude': -122.589645}
{'Latitude': 47.043535, 'Longitude': -122.89692}
{'Latitude': 47.67668, 'Longitude': -122.12302}
{'Latitude': 47.67668, 'Longitude': -122.12302}
{'Latitude': 47.66866, 'Longitude': -122.37815}
{'Latitude': 47.659185, 'Longitude': -122.34301}
{'Latitude': 48.76809, 'Longitude': -122.45493}
{'Latitude': 48.501275, 'Longitude': -122.615305}
{'Latitude': 47.762405, 'Longitude': -122.20578}
{'Latitude': 47.582905, 'Longitude': -122.2377542}
{'Latitude': 47.915555, 'Longitude': -122.091505}
{'Latitude': 47.61546, 'Longitude': -122.344125}
```



```
{'Latitude': 47.161465, 'Longitude': -122.029685}
{'Latitude': 47.18062, 'Longitude': -122.183805}
{'Latitude': 47.534065, 'Longitude': -122.03646}
{'Latitude': 47.6285782, 'Longitude': -122.0313266}
{'Latitude': 47.820245, 'Longitude': -122.1873}
{'Latitude': 48.19485, 'Longitude': -122.12324}
{'Latitude': 47.61598, 'Longitude': -122.228025}
{'Latitude': 48.41333, 'Longitude': -122.338975}
{'Latitude': 47.03646, 'Longitude': -122.8285}
{'Latitude': 48.9461196, 'Longitude': -122.4584536}
{'Latitude': 47.68968, 'Longitude': -122.37275}
{'Latitude': 47.534065, 'Longitude': -122.03646}
{'Latitude': 47.599975, 'Longitude': -122.147385}
{'Latitude': 47.61598, 'Longitude': -122.228025}
{'Latitude': 48.19485, 'Longitude': -122.12324}
{'Latitude': 47.820245, 'Longitude': -122.1873}
{'Latitude': 48.08125, 'Longitude': -123.105015}
{'Latitude': 47.58153, 'Longitude': -122.03309}
{'Latitude': 47.802589, 'Longitude': -122.179458}
{'Latitude': 45.59009, 'Longitude': -122.405565}
{'Latitude': 47.71558, 'Longitude': -122.296385}
{'Latitude': 47.9796552, 'Longitude': -122.359364}
{'Latitude': 48.1195874, 'Longitude': -122.7644197}
{'Latitude': 47.9156409, 'Longitude': -122.2247757}
{'Latitude': 47.68968, 'Longitude': -122.37275}
{'Latitude': 47.66132, 'Longitude': -122.319115}
{'Latitude': 47.819365, 'Longitude': -122.316675}
{'Latitude': 47.045935, 'Longitude': -122.92145}
{'Latitude': 48.3829111, 'Longitude': -122.5135345}
{'Latitude': 47.58153, 'Longitude': -122.03309}
{'Latitude': 47.67949, 'Longitude': -122.3185}
{'Latitude': 47.820245, 'Longitude': -122.1873}
{'Latitude': 47.80807, 'Longitude': -122.37507}
{'Latitude': 47.50524, 'Longitude': -122.6847073}
{'Latitude': 47.487175, 'Longitude': -122.15734}
{'Latitude': 45.703706, 'Longitude': -122.70302}
{'Latitude': 47.534065, 'Longitude': -122.03646}
{'Latitude': 47.820245, 'Longitude': -122.1873}
{'Latitude': 47.19178, 'Longitude': -122.299155}
{'Latitude': 47.5484866, 'Longitude': -121.9993659}
{'Latitude': 45.8662548, 'Longitude': -122.6706246}
{'Latitude': 47.65942, 'Longitude': -117.19651}
{'Latitude': 47.19178, 'Longitude': -122.299155}
{'Latitude': 47.75855, 'Longitude': -122.151665}
{'Latitude': 47.9156409, 'Longitude': -122.2247757}
{'Latitude': 47.56484, 'Longitude': -122.38679}
{'Latitude': 48.761615, 'Longitude': -122.486115}
{'Latitude': 47.476, 'Longitude': -122.286465}
```



```
{'Latitude': 47.9156409, 'Longitude': -122.2247757}
{'Latitude': 47.64058, 'Longitude': -122.32226}
{'Latitude': 47.80372, 'Longitude': -122.335685}
{'Latitude': 48.501275, 'Longitude': -122.615305}
{'Latitude': 47.802589, 'Longitude': -122.179458}
{'Latitude': 46.553505, 'Longitude': -120.477805}
{'Latitude': 47.241885, 'Longitude': -122.36151}
{'Latitude': 47.6785, 'Longitude': -122.20264}
{'Latitude': 47.58153, 'Longitude': -122.03309}
{'Latitude': 48.2192797, 'Longitude': -122.5310901}
{'Latitude': 47.56484, 'Longitude': -122.38679}
{'Latitude': 47.67668, 'Longitude': -122.12302}
{'Latitude': 47.1042426, 'Longitude': -122.3085456}
{'Latitude': 47.84182, 'Longitude': -122.297265}
{'Latitude': 47.8650827, 'Longitude': -122.2551991}
{'Latitude': 45.59009, 'Longitude': -122.405565}
{'Latitude': 48.2897314, 'Longitude': -122.6788673}
{'Latitude': 47.4935316, 'Longitude': -121.7814012}
{'Latitude': 47.476, 'Longitude': -122.286465}
{'Latitude': 47.57192, 'Longitude': -122.65223}
{'Latitude': 47.659185, 'Longitude': -122.34301}
{'Latitude': 47.903975, 'Longitude': -122.57781}
{'Latitude': 47.77279, 'Longitude': -122.382425}
{'Latitude': 47.582905, 'Longitude': -122.2377542}
{'Latitude': 47.18062, 'Longitude': -122.183805}
{'Latitude': 47.820245, 'Longitude': -122.1873}
{'Latitude': 47.610365, 'Longitude': -122.30839}
{'Latitude': 47.819365, 'Longitude': -122.316675}
{'Latitude': 48.1195874, 'Longitude': -122.7644197}
{'Latitude': 47.5415, 'Longitude': -122.388675}
{'Latitude': 47.62523, 'Longitude': -122.30764}
{'Latitude': 47.500055, 'Longitude': -122.180505}
{'Latitude': 47.043535, 'Longitude': -122.89692}
{'Latitude': 47.36607, 'Longitude': -120.199045}
{'Latitude': 47.4797047, 'Longitude': -122.21024}
{'Latitude': 47.67949, 'Longitude': -122.3185}
{'Latitude': 47.2198, 'Longitude': -122.47913}
{'Latitude': 47.77279, 'Longitude': -122.382425}
{'Latitude': 47.6293323, 'Longitude': -122.5235781}
{'Latitude': 47.0821119, 'Longitude': -122.7474291}
{'Latitude': 47.0135926, 'Longitude': -122.9131017}
{'Latitude': 47.8851158, 'Longitude': -122.15134}
{'Latitude': 47.50524, 'Longitude': -122.6847073}
{'Latitude': 47.487175, 'Longitude': -122.15734}
{'Latitude': 47.67668, 'Longitude': -122.12302}
{'Latitude': 47.67668, 'Longitude': -122.12302}
{'Latitude': 47.802589, 'Longitude': -122.179458}
{'Latitude': 47.4451257, 'Longitude': -122.1298876}
```



```
{'Latitude': 47.915555, 'Longitude': -122.091505}
{'Latitude': 47.6245, 'Longitude': -122.11832}
{'Latitude': 47.211085, 'Longitude': -123.105305}
{'Latitude': 48.4152, 'Longitude': -122.322955}
{'Latitude': 47.476, 'Longitude': -122.286465}
{'Latitude': 47.487175, 'Longitude': -122.15734}
{'Latitude': 47.0821119, 'Longitude': -122.7474291}
{'Latitude': 45.818445, 'Longitude': -122.74291}
{'Latitude': 47.659185, 'Longitude': -122.34301}
{'Latitude': 45.641205, 'Longitude': -122.666325}
{'Latitude': 47.4451257, 'Longitude': -122.1298876}
{'Latitude': 47.8851158, 'Longitude': -122.15134}
{'Latitude': 47.487175, 'Longitude': -122.15734}
{'Latitude': 47.80372, 'Longitude': -122.335685}
{'Latitude': 47.802589, 'Longitude': -122.179458}
{'Latitude': 47.61546, 'Longitude': -122.344125}
{'Latitude': 47.43473, 'Longitude': -122.29179}
{'Latitude': 45.6228731, 'Longitude': -122.589388}
{'Latitude': 47.045935, 'Longitude': -122.92145}
{'Latitude': 47.5373, 'Longitude': -122.639265}
{'Latitude': 47.46233, 'Longitude': -122.32863}
{'Latitude': 47.5345546, 'Longitude': -121.8740496}
{'Latitude': 45.72977, 'Longitude': -121.48347}
{'Latitude': 47.8851158, 'Longitude': -122.15134}
{'Latitude': 47.915555, 'Longitude': -122.091505}
{'Latitude': 47.476, 'Longitude': -122.286465}
{'Latitude': 47.820245, 'Longitude': -122.1873}
{'Latitude': 47.201625, 'Longitude': -122.23825}
{'Latitude': 47.500055, 'Longitude': -122.180505}
{'Latitude': 47.820245, 'Longitude': -122.1873}
{'Latitude': 47.67668, 'Longitude': -122.12302}
{'Latitude': 47.571015, 'Longitude': -122.16937}
{'Latitude': 47.357985, 'Longitude': -122.05191}
{'Latitude': 47.36078, 'Longitude': -122.111625}
{'Latitude': 47.476, 'Longitude': -122.286465}
{'Latitude': 47.5345546, 'Longitude': -121.8740496}
{'Latitude': 47.6285782, 'Longitude': -122.0313266}
{'Latitude': 47.639195, 'Longitude': -122.394185}
{'Latitude': 47.8851158, 'Longitude': -122.15134}
{'Latitude': 47.802589, 'Longitude': -122.179458}
{'Latitude': 47.6285782, 'Longitude': -122.0313266}
{'Latitude': 47.500055, 'Longitude': -122.180505}
{'Latitude': 47.582905, 'Longitude': -122.2377542}
{'Latitude': 45.8662548, 'Longitude': -122.6706246}
{'Latitude': 48.03557, 'Longitude': -122.408015}
{'Latitude': 47.67949, 'Longitude': -122.3185}
{'Latitude': 47.487175, 'Longitude': -122.15734}
{'Latitude': 47.097455, 'Longitude': -122.643815}
```



```
{'Latitude': 47.3198995, 'Longitude': -122.1820969}
{'Latitude': 47.6285782, 'Longitude': -122.0313266}
{'Latitude': 47.58153, 'Longitude': -122.03309}
{'Latitude': 47.0821119, 'Longitude': -122.7474291}
{'Latitude': 47.5970083, 'Longitude': -120.6619153}
{'Latitude': 47.8650827, 'Longitude': -122.2551991}
{'Latitude': 45.58359, 'Longitude': -122.35465}
{'Latitude': 47.903975, 'Longitude': -122.57781}
{'Latitude': 45.678565, 'Longitude': -122.66592}
{'Latitude': 47.37483, 'Longitude': -122.199755}
{'Latitude': 45.59009, 'Longitude': -122.405565}
{'Latitude': 47.582905, 'Longitude': -122.2377542}
{'Latitude': 47.61546, 'Longitude': -122.344125}
{'Latitude': 47.68968, 'Longitude': -122.37275}
{'Latitude': 47.153995, 'Longitude': -122.43827}
{'Latitude': 47.52104, 'Longitude': -122.356145}
{'Latitude': 47.66132, 'Longitude': -122.319115}
{'Latitude': 47.6018, 'Longitude': -122.329075}
{'Latitude': 48.0047, 'Longitude': -122.112265}
{'Latitude': 47.84182, 'Longitude': -122.297265}
{'Latitude': 47.820245, 'Longitude': -122.1873}
{'Latitude': 47.0135926, 'Longitude': -122.9131017}
{'Latitude': 47.68968, 'Longitude': -122.37275}
{'Latitude': 48.474765, 'Longitude': -122.33079}
{'Latitude': 47.3809, 'Longitude': -122.235475}
{'Latitude': 47.5484866, 'Longitude': -121.9993659}
{'Latitude': 47.635365, 'Longitude': -117.425265}
{'Latitude': 47.68968, 'Longitude': -122.37275}
{'Latitude': 47.820245, 'Longitude': -122.1873}
{'Latitude': 47.342345, 'Longitude': -122.589645}
{'Latitude': 48.0047, 'Longitude': -122.112265}
{'Latitude': 48.4152, 'Longitude': -122.322955}
{'Latitude': 47.5345546, 'Longitude': -121.8740496}
{'Latitude': 45.59009, 'Longitude': -122.405565}
{'Latitude': 45.818445, 'Longitude': -122.74291}
{'Latitude': 47.820245, 'Longitude': -122.1873}
{'Latitude': 47.190465, 'Longitude': -122.28718}
{'Latitude': 48.761615, 'Longitude': -122.486115}
{'Latitude': 47.58153, 'Longitude': -122.03309}
{'Latitude': 47.52104, 'Longitude': -122.356145}
{'Latitude': 47.6958998, 'Longitude': -122.0222799}
{'Latitude': 47.03646, 'Longitude': -122.8285}
{'Latitude': 47.581975, 'Longitude': -122.30823}
{'Latitude': 47.045935, 'Longitude': -122.92145}
{'Latitude': 47.66866, 'Longitude': -122.37815}
{'Latitude': 47.659185, 'Longitude': -122.34301}
{'Latitude': 45.678565, 'Longitude': -122.66592}
{'Latitude': 47.4935316, 'Longitude': -121.7814012}
```

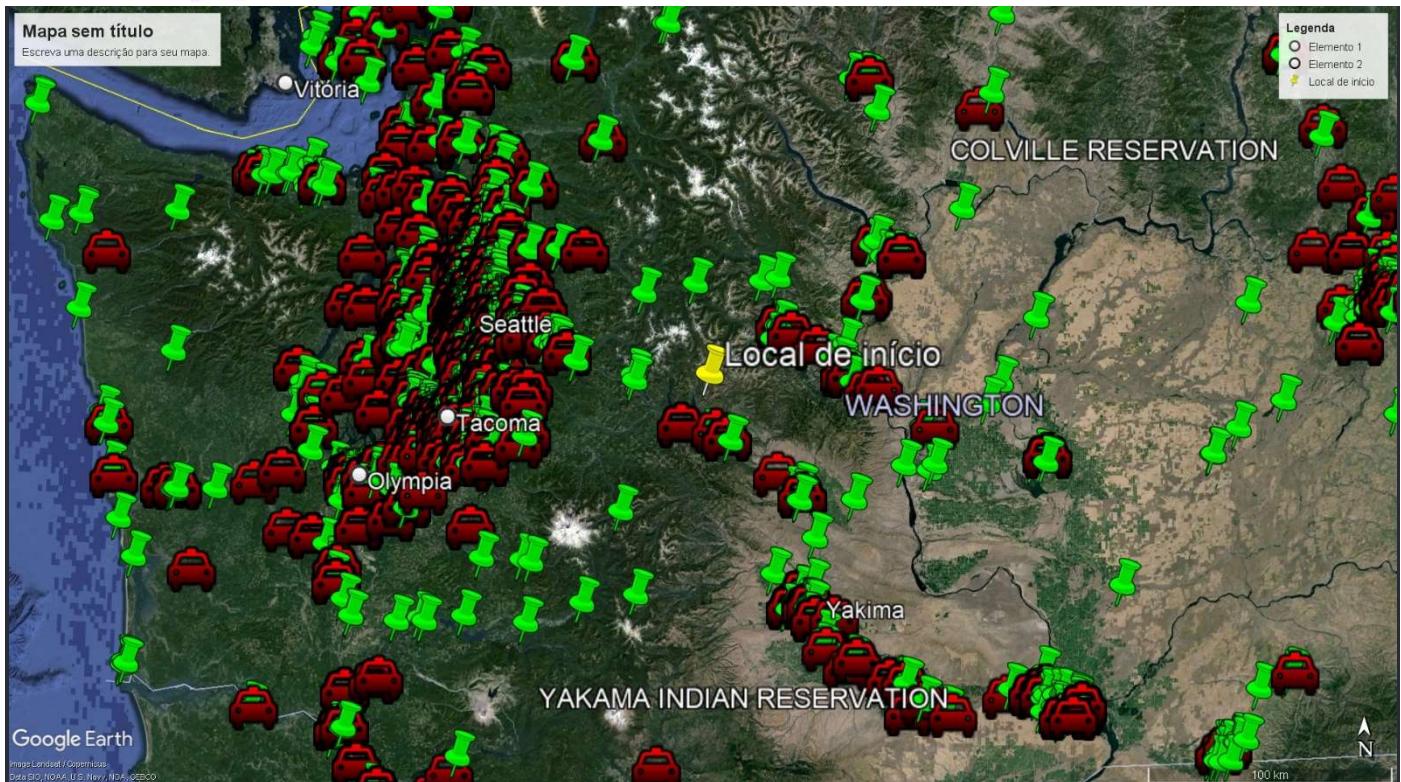


```
{'Latitude': 47.18062, 'Longitude': -122.183805}
{'Latitude': 47.7578146, 'Longitude': -122.3175}
{'Latitude': 47.737525, 'Longitude': -122.64177}
{'Latitude': 47.75855, 'Longitude': -122.151665}
{'Latitude': 47.68968, 'Longitude': -122.37275}
{'Latitude': 47.639195, 'Longitude': -122.394185}
{'Latitude': 47.4176, 'Longitude': -120.28674}
{'Latitude': 47.802589, 'Longitude': -122.179458}
{'Latitude': 47.75855, 'Longitude': -122.151665}
{'Latitude': 45.59009, 'Longitude': -122.405565}
{'Latitude': 45.678565, 'Longitude': -122.66592}
{'Latitude': 47.43473, 'Longitude': -122.29179}
{'Latitude': 47.67668, 'Longitude': -122.12302}
{'Latitude': 46.204995, 'Longitude': -119.769315}
{'Latitude': 46.29747, 'Longitude': -119.28753}
{'Latitude': 47.61385, 'Longitude': -122.201905}
{'Latitude': 47.043535, 'Longitude': -122.89692}
{'Latitude': 47.61385, 'Longitude': -122.201905}
{'Latitude': 47.476, 'Longitude': -122.286465}
{'Latitude': 47.2254086, 'Longitude': -122.6066806}
{'Latitude': 46.66113, 'Longitude': -122.96692}
{'Latitude': 45.638545, 'Longitude': -122.641835}
{'Latitude': 47.67949, 'Longitude': -122.3185}
{'Latitude': 47.639195, 'Longitude': -122.394185}
{'Latitude': 48.761615, 'Longitude': -122.486115}
{'Latitude': 47.549285, 'Longitude': -122.28339}
{'Latitude': 47.599975, 'Longitude': -122.147385}
{'Latitude': 45.7010427, 'Longitude': -122.6483953}
{'Latitude': 47.659185, 'Longitude': -122.34301}
{'Latitude': 47.18062, 'Longitude': -122.183805}
{'Latitude': 47.3234488, 'Longitude': -122.5835454}
```

```
import webbrowser

# Caminho para o arquivo HTML
caminho_arquivo_html = r'D:\OneDrive - Instituto Presbiteriano Mackenzie\GitHub\google
earth\imagem de distribuicao automoveis x pontos de carga.html'

# Abrir o arquivo HTML em um navegador padrão
webbrowser.open('file://' + caminho_arquivo_html)
```



No grafico conseguimos visualizar a distribuição dos automoveis eletricos e dos pontos de recarga, mas precisamos verificar isto em numeros.
Para isto realizmos calculos de densidade

```
#Precisamos calcular a densidade de veiculos por unidade
#por area: Formula Densidade = Numero de veiculos/Area da Regiao em quilometros quadrados
#area da regiao de Washington em km2 177 quilometros segundo site https://www.gree-
lane.com/pt/humanidades/geografia/washington-dc-geography-
1435747/#:~:text=DC%20tem%2068%20milhas%20quadradas%20A%20%C3%A1rea%20to-
tal,m%29%20e%20est%C3%A1%20localizado%20no%20bairro%20de%20Tenleytown.
Formula_Densidade = total / 177
print(Formula_Densidade)
print(Formula_Densidade)
764.7683615819209

#Densidade de veiculos em relação a capacidade de carga.
#levar em consideração a capacidade de carga dos pontos em relação
#aos diferentes tipos de pontos com capacidade variada
#formula densidade = numero de veiculos / capacidade total dos pontos de recarga
capacidade_EV_Level1 = total / 17
print(capacidade_EV_Level1)
print(capacidade_EV_Level1)
7962.588235294118

capacidade_EV_Level2= 135364 / 1045
print(capacidade_EV_Level2)print(capacidade_EV_Level1)
print(capacidade_EV_Level2)
129.53492822966507EV_DC_Fast_Count = 135364 / 4175
```



```
print(EV_DC_Fast_Count)
print(EV_DC_Fast_Count)
32.42251497005988
```

E analisar tendencias futuras usando graficos e estatisticas para analisar tendencias

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Carregar os dados
base_dados = pd.read_csv('D:/OneDrive - Instituto Presbiteriano Mackenzie/Aulas/3 semestre/Projeto Aplicado II/Electric_Vehicle_Population_Data.csv', index_col=False)

# Filtrar os dados para incluir apenas veículos desde 2015 e do tipo BEV
base = base_dados[(base_dados['Model Year'] >= 2015) & (base_dados['Electric Vehicle Type'] == 'Battery Electric Vehicle (BEV)')]

# Agrupar por ano e calcular a quantidade total de veículos por ano
base_agrupada = base.groupby('Model Year').size().reset_index(name='Quantidade')

# Pré-processamento de dados
X = base_agrupada["Model Year"].values.reshape(-1, 1)
y = base_agrupada["Quantidade"].values

# Dividir dados em conjuntos de treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1234)

# Treinar o modelo
model = LinearRegression()
model.fit(X_train, y_train)

# Avaliar o modelo
print("R²:", model.score(X_test, y_test))
print("RMSE:", np.sqrt(mean_squared_error(y_test, model.predict(X_test)))))

# Fazer previsões para um futuro "Model Year"
ano_futuro = 2030
previsao = model.predict([[ano_futuro]])

print(f"Previsão de vendas para {ano_futuro}: {previsao}")

# Criar um vetor de "Model Year" para as previsões
anos_futuro = np.arange(2023, 2030)
```



```
# Fazer previsões para os "Model Years" futuros  
previsoes = model.predict(anos_futuro.reshape(-1, 1))
```

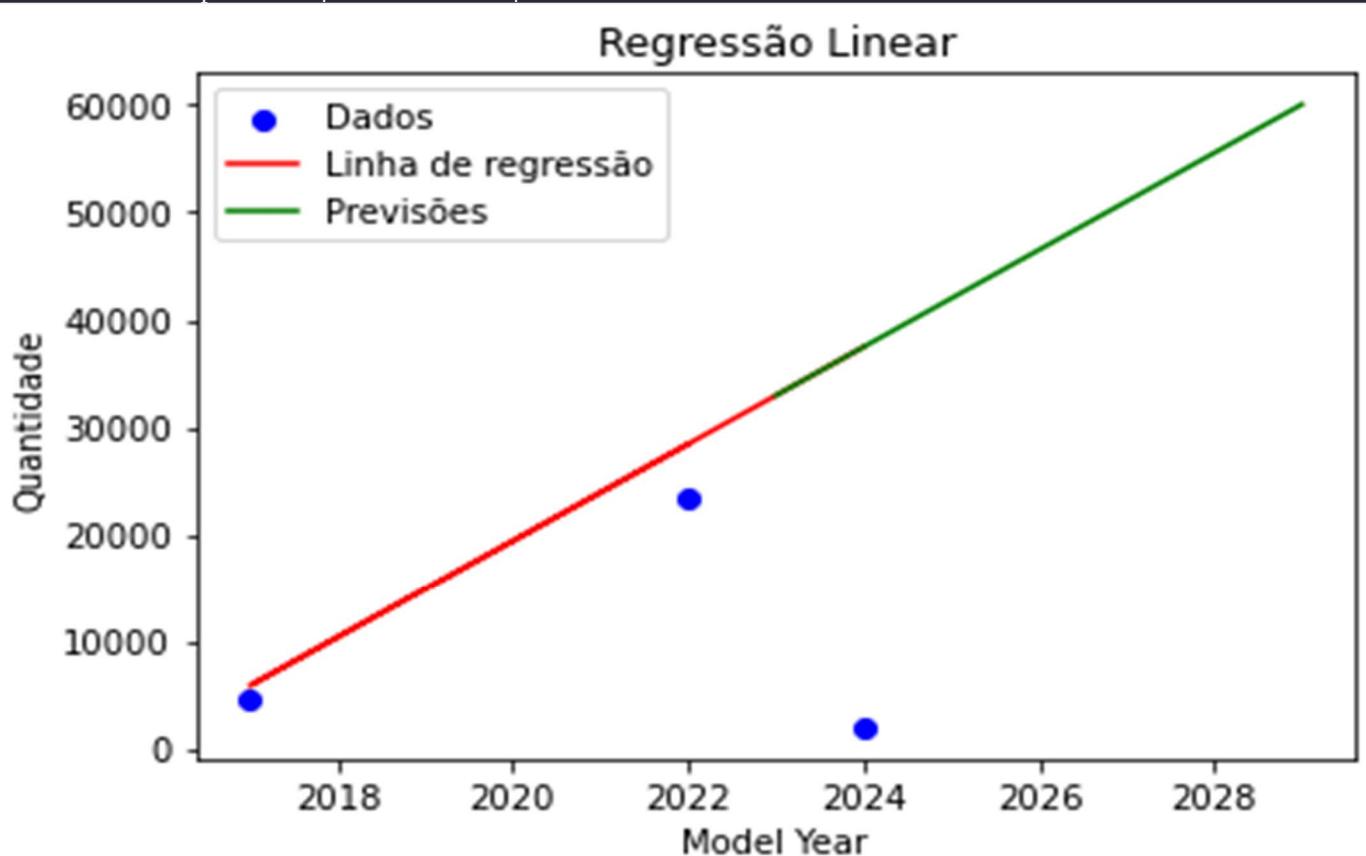
```
# Plotar os dados  
plt.scatter(X_test, y_test, color="blue", label="Dados")  
plt.plot(X_test, model.predict(X_test), color="red", label="Linha de regressão")  
plt.plot(anos_futuro, previsoes, color="green", label="Previsões")
```

```
# Ajustar o gráfico  
plt.legend()  
plt.xlabel("Model Year")  
plt.ylabel("Quantidade")  
plt.title("Regressão Linear")  
plt.show()
```

R²: -3.6839355378023946

RMSE: 20703.888494630515

Previsão de vendas para 2030: [64586.74390244]



Agora com base em suposições sobre o crescimento futuro

```
# Importar as bibliotecas necessárias  
import pandas as pd  
import numpy as np  
from sklearn.linear_model import LinearRegression  
  
# Dados de modelo linear ajustado previamente  
lm = LinearRegression()
```



```
lm.fit(X_train, y_train)

# Cenário otimista: crescimento de 10% ao ano
optimistic_growth_rate = 0.10
ano_ottimista = 2030
crescimento_estimado_ottimista = model.predict([[ano_ottimista]])[0] * (1 + optimistic_growth_rate)
print("Cenário otimista - Quantidade estimada de veículos em 2030:", crescimento_estimado_ottimista)
print("Cenário otimista - Quantidade estimada de veículos em 2030:", crescimento_estimado_ottimista)
Cenário otimista - Quantidade estimada de veículos em 2030: 71045.41829268319

# Cenário pessimista: crescimento de 5% ao ano
pessimistic_growth_rate = 0.05
ano_pessimista = 2030
crescimento_estimado_pessimista = model.predict([[ano_pessimista]])[0] * (1 + pessimistic_growth_rate)
print("Cenário pessimista - Quantidade estimada de veículos em 2030:", crescimento_estimado_pessimista)
print("Cenário pessimista - Quantidade estimada de veículos em 2030:", crescimento_estimado_pessimista)
Cenário pessimista - Quantidade estimada de veículos em 2030: 67816.08109756121

# Cenário realista: crescimento de 7% ao ano
realistic_growth_rate = 0.07
ano_realista = 2030
crescimento_estimado_realista = model.predict([[ano_realista]])[0] * (1 + realistic_growth_rate)
print("Cenário realista - Quantidade estimada de veículos em 2030:", crescimento_estimado_realista)
print("Cenário realista - Quantidade estimada de veículos em 2030:", crescimento_estimado_realista)
Cenário realista - Quantidade estimada de veículos em 2030: 69107.81597561
```