```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

import pandas as pd
import numpy as np

df = pd.read_excel("/content/drive/MyDrive/dataset.xlsx")
df.head(5)
```

{"type":"dataframe","variable_name":"df"}

```
print(df.shape,df.size)

(112634, 17) 1914778

df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112634 entries, 0 to 112633
Data columns (total 17 columns):
 #   Column                                             Non-Null Count
Dtype
---  ------                                             --------------
-----
 0   VIN (1-10)                                         112634 non-
null  object
 1   County                                             112634 non-
null  object
 2   City                                               112634 non-
null  object
 3   State                                              112634 non-
null  object
 4   Postal Code                                        112634 non-
null  int64
 5   Model Year                                         112634 non-
null  int64
 6   Make                                               112634 non-
null  object
 7   Model                                              112614 non-
null  object
 8   Electric Vehicle Type                              112634 non-
null  object
 9   Clean Alternative Fuel Vehicle (CAFV) Eligibility  112634 non-
null  object
 10  Electric Range                                     112634 non-
null  int64
 11  Base MSRP                                          112634 non-
```

```
null  int64
 12  Legislative District                                 112348 non-
null  float64
 13  DOL Vehicle ID                                       112634 non-
null  int64
 14  Vehicle Location                                     112610 non-
null  object
 15  Electric Utility                                     112191 non-
null  object
 16  2020 Census Tract                                    112634 non-
null  int64
dtypes: float64(1), int64(6), object(10)
memory usage: 14.6+ MB

# as we see clearly that there are many null values in our data in
differ differ columns

df = df.dropna()

df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 112152 entries, 2 to 112633
Data columns (total 17 columns):
 #   Column                                               Non-Null Count
Dtype
---  ------                                               --------------
-----
 0   VIN (1-10)                                           112152 non-
null  object
 1   County                                               112152 non-
null  object
 2   City                                                 112152 non-
null  object
 3   State                                                112152 non-
null  object
 4   Postal Code                                          112152 non-
null  int64
 5   Model Year                                           112152 non-
null  int64
 6   Make                                                 112152 non-
null  object
 7   Model                                                112152 non-
null  object
 8   Electric Vehicle Type                                112152 non-
null  object
 9   Clean Alternative Fuel Vehicle (CAFV) Eligibility  112152 non-
null  object
 10  Electric Range                                       112152 non-
null  int64
```

```
 11  Base MSRP                                         112152 non-
null  int64
 12  Legislative District                              112152 non-
null  float64
 13  DOL Vehicle ID                                    112152 non-
null  int64
 14  Vehicle Location                                  112152 non-
null  object
 15  Electric Utility                                  112152 non-
null  object
 16  2020 Census Tract                                 112152 non-
null  int64
dtypes: float64(1), int64(6), object(10)
memory usage: 15.4+ MB
```

```
# so now our data have no null values

# in our data , datatype is perfectly arrange already

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')
```

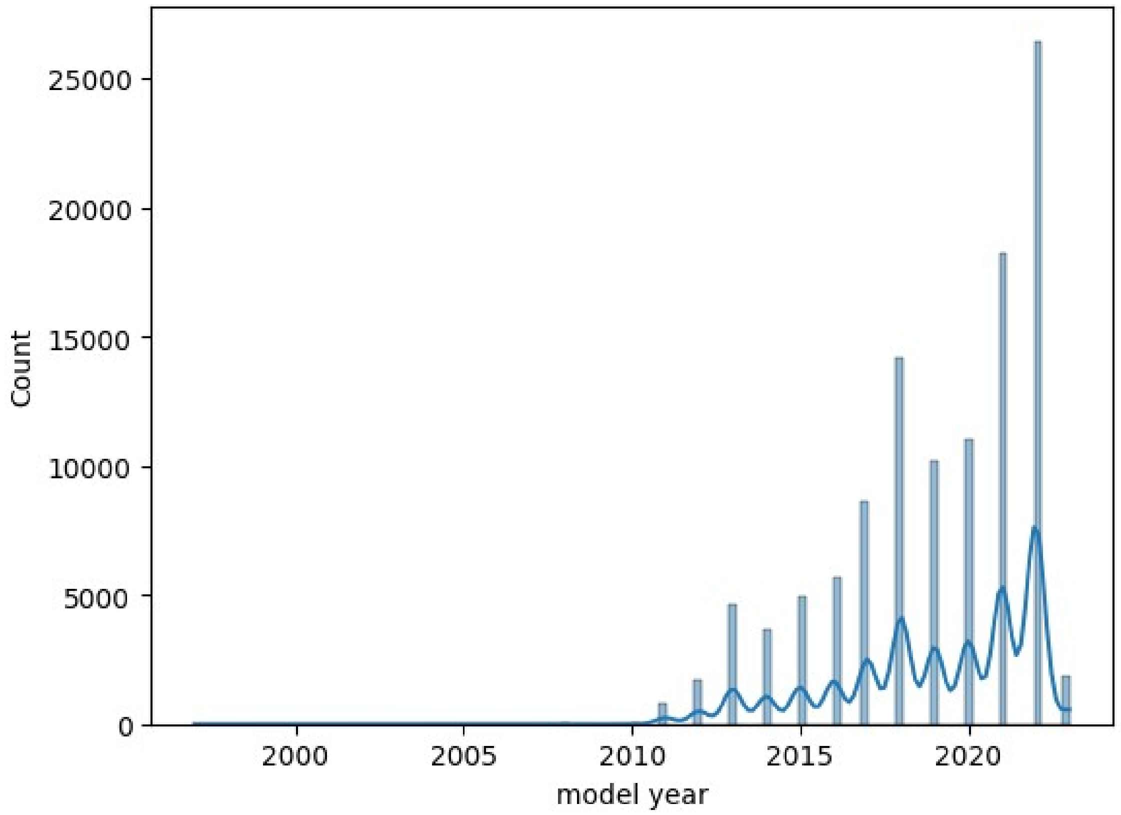
```
# there is an mistake in a column name county
df.rename(columns={'County':'Country'},inplace=True)

df.columns = df.columns.str.strip().str.lower()

df.columns
```

```
Index(['vin (1-10)', 'country', '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')
```

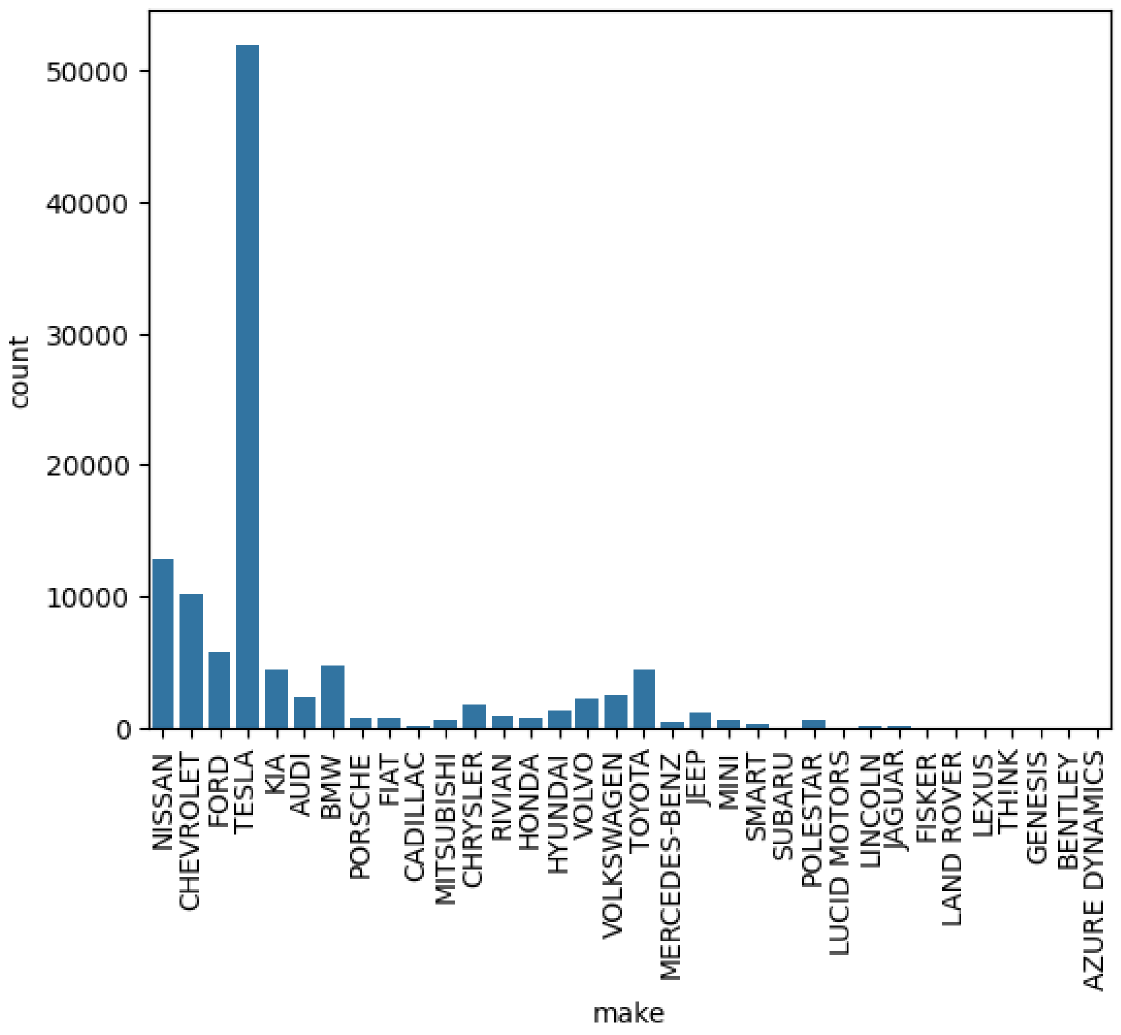
```
# so our columns is now perfectly organise
```

Univariate Analysis

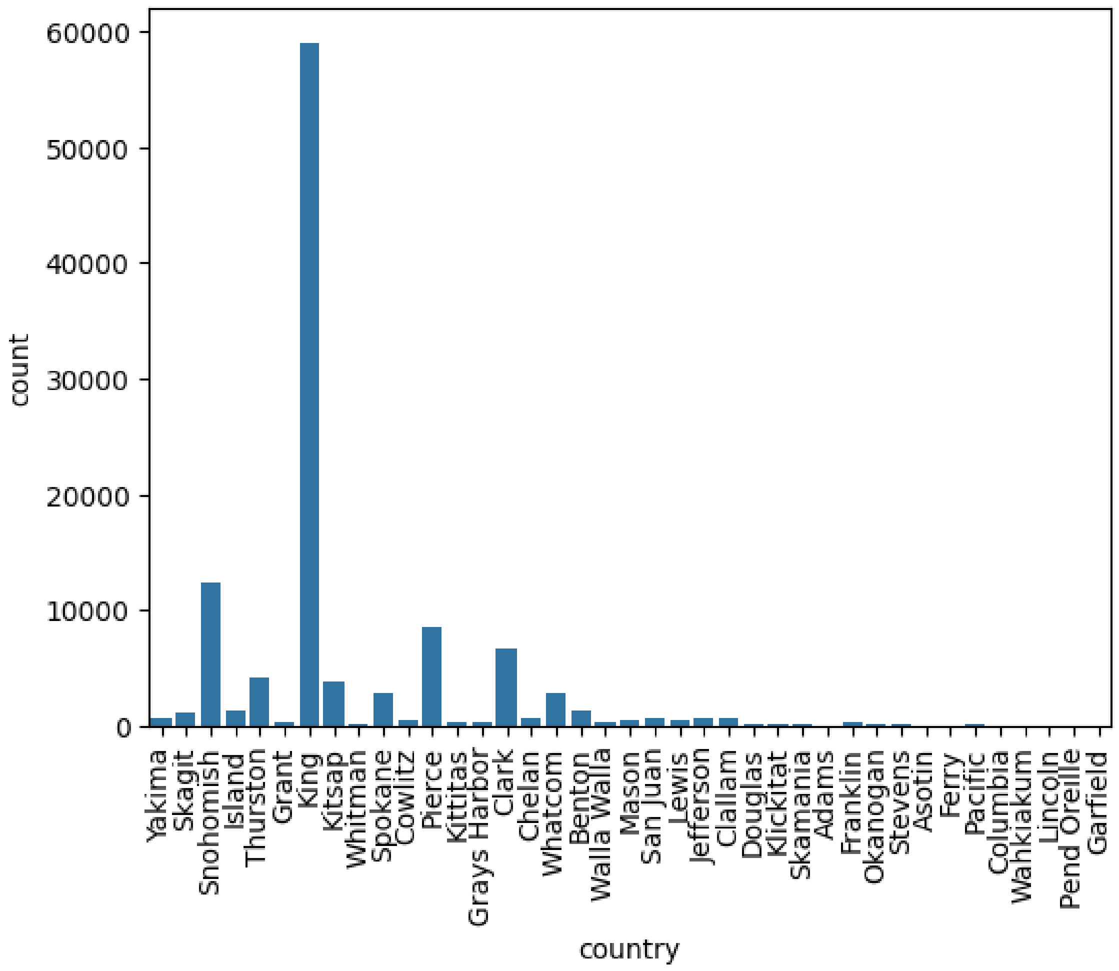
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.histplot(df["model year"],kde=True)
plt.show()
```



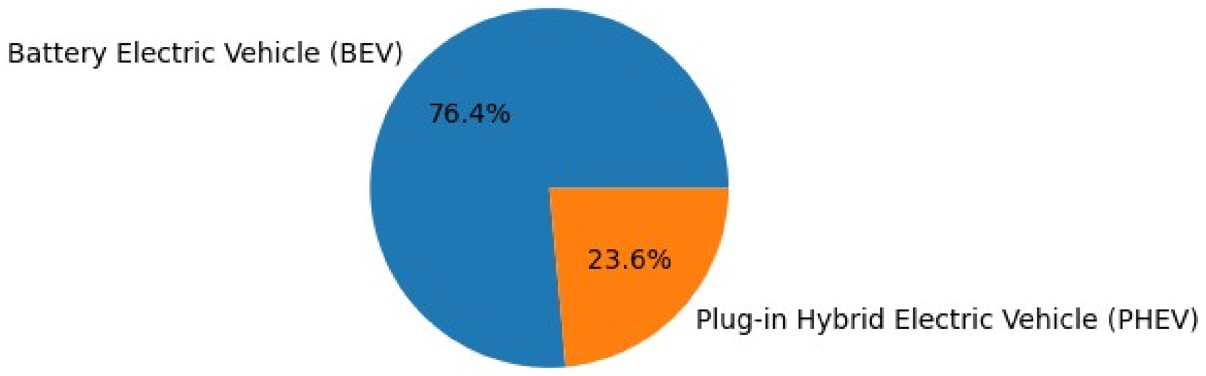
```
# by this visualization you see that electric vehicles are start
manufacturing from 2000
# but boom came in market after 2015 and in 2022 this market went high
top

sns.countplot(x='make', data=df)
plt.xticks(rotation=90)
plt.show()
```



```python
# Really TESLA make this market in ploting you see that TESLA is only
brand which has high block .
# we also see that other brand also include it but in comparison with
tesla they are too small .

sns.countplot(x='country', data=df)
plt.xticks(rotation=90)
plt.show()
```



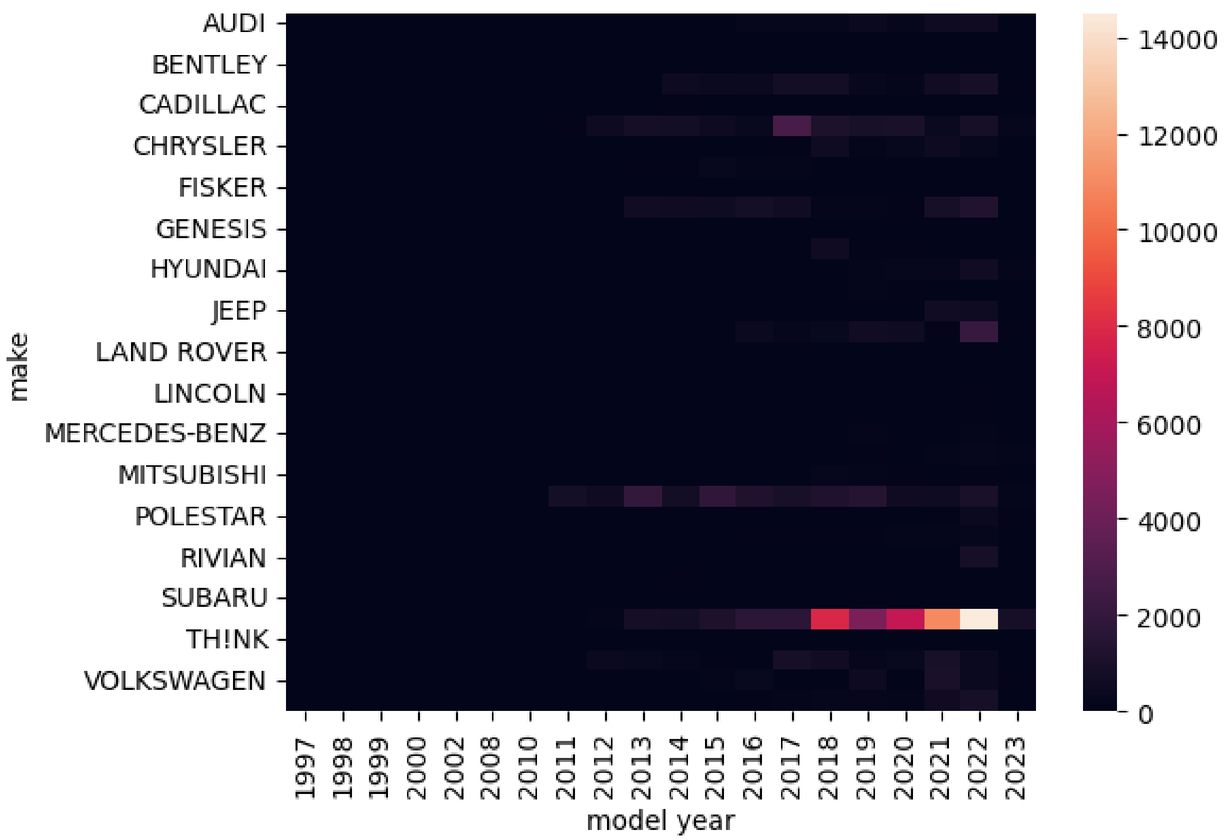
```
# unbeliable only country which make noise in whole market woww.....

plt.pie(df['electric vehicle type'].value_counts(),labels=df['electric
vehicle type'].value_counts().index,autopct='%1.1f%%')
fig = plt.gcf()
fig.set_size_inches(3,3)
plt.show()
```



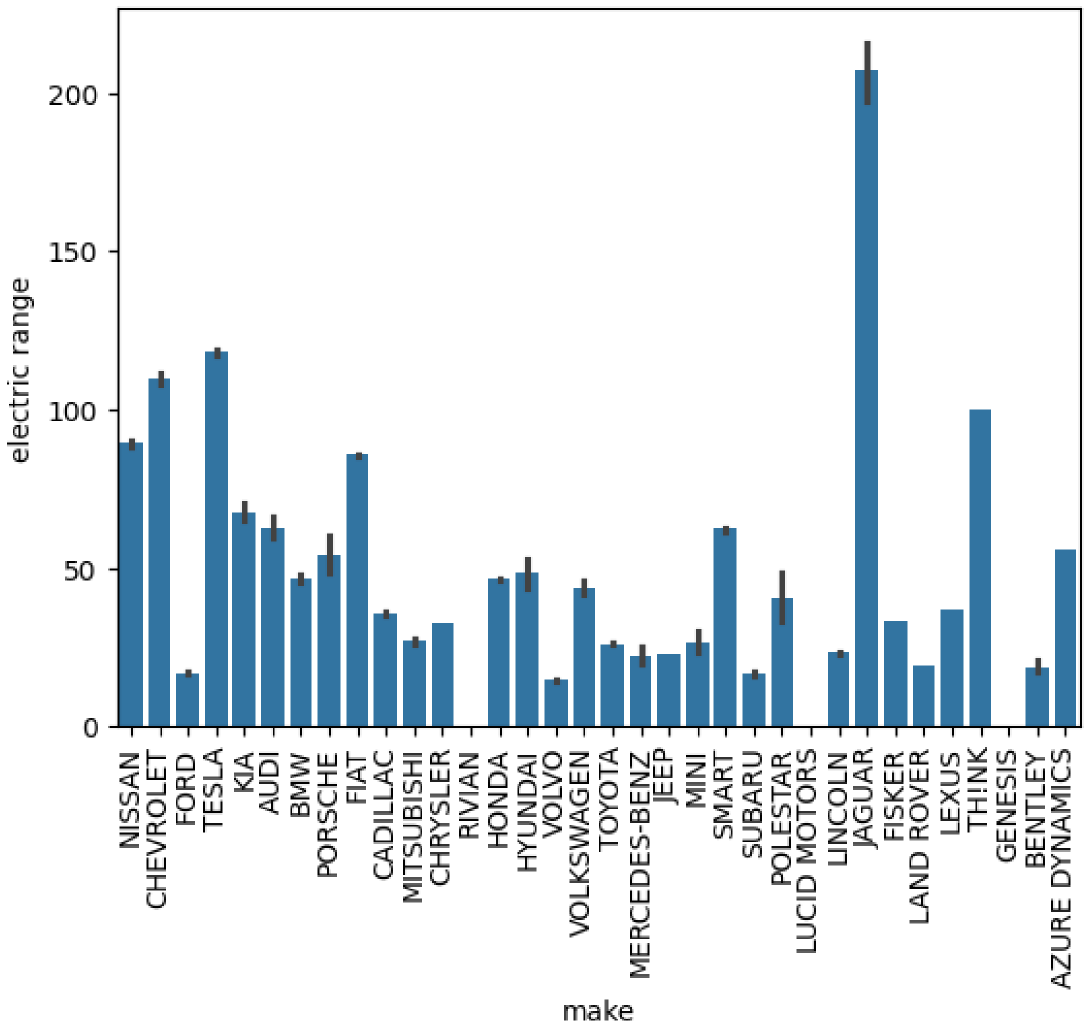
```
# PHEV vehicles are also have some amount of vehicles in market
```

Bi-Variate Analysis

```
car_data = df.groupby(['make', 'model
year']).size().unstack(fill_value=0)
sns.heatmap(car_data,  fmt='d')
plt.xticks(rotation=90)
plt.show()
```

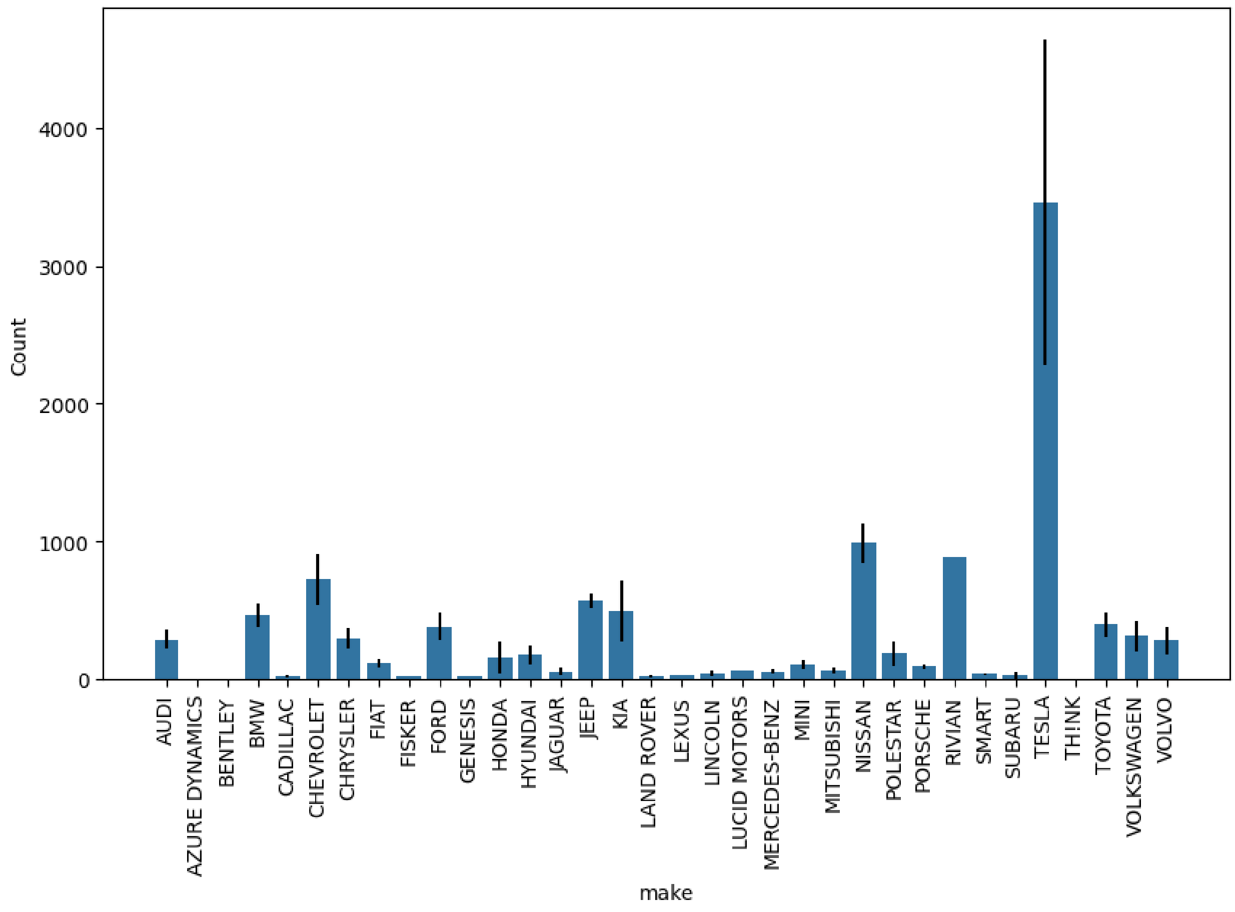


```
# as you see SUBARU brand have made mostly car after 2018 and most at
2022

sns.barplot(x='make' , y='electric range' , data=df)
plt.xticks(rotation=90)
plt.show()
```



```
# jaguar cars has covering high range now-a-days

car_data = df.groupby(['make', 'model
year']).size().reset_index(name='Count')
mean_counts = car_data.groupby('make')['Count'].mean()
std_counts = car_data.groupby('make')['Count'].std()
error = std_counts / np.sqrt(car_data.groupby('make')
['Count'].count())

plt.figure(figsize=(10, 6))
sns.barplot(x='make', y='Count', data=car_data, ci=None)  # ci=None to
avoid automatic error bars
plt.errorbar(x=mean_counts.index, y=mean_counts, yerr=error,
fmt='none', c='black')
plt.xticks(rotation=90)
plt.show()
```

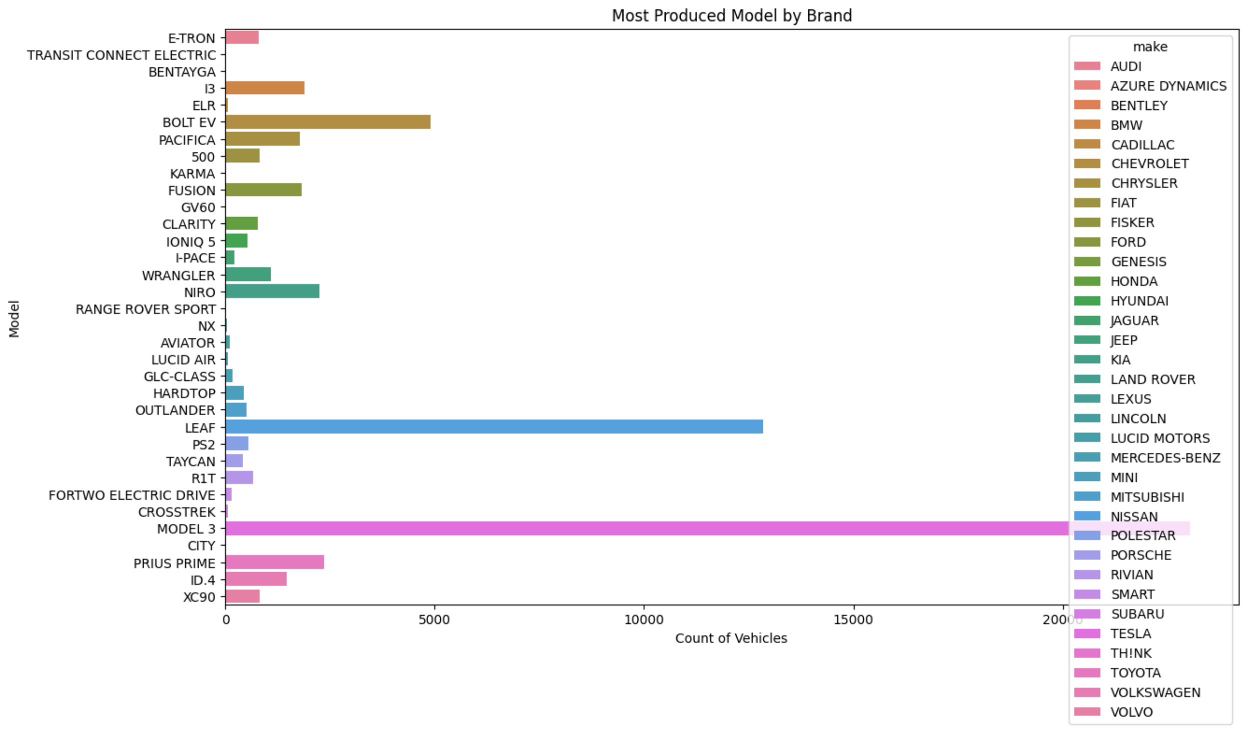
```
<ipython-input-29-3ef96cada37c>:2: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same
effect.

  sns.barplot(x='make', y='Count', data=car_data, ci=None)  # ci=None
to avoid automatic error bars
```



```
model_counts = df.groupby(['make',
'model']).size().reset_index(name='Count')
most_produced_models = model_counts.loc[model_counts.groupby('make')
['Count'].idxmax()]
most_produced_models_sorted =
most_produced_models.sort_values(by='Count', ascending=False)
print(most_produced_models_sorted)
```

```
               make                        model  Count
95            TESLA                      MODEL 3  23042
81           NISSAN                         LEAF  12846
27        CHEVROLET                      BOLT EV   4895
102          TOYOTA                  PRIUS PRIME   2365
```

```
57                 KIA                        NIRO   2252
18                 BMW                          I3   1888
38                FORD                      FUSION   1827
31            CHRYSLER                    PACIFICA   1780
106         VOLKSWAGEN                        ID.4   1480
55                JEEP                    WRANGLER   1096
32                FIAT                         500    820
113              VOLVO                        XC90    817
3                 AUDI                      E-TRON    795
45               HONDA                     CLARITY    779
89              RIVIAN                         R1T    672
82            POLESTAR                         PS2    557
47             HYUNDAI                     IONIQ 5    542
80          MITSUBISHI                   OUTLANDER    520
78                MINI                     HARDTOP    439
86             PORSCHE                      TAYCAN    418
53              JAGUAR                      I-PACE    218
74       MERCEDES-BENZ                   GLC-CLASS    179
92               SMART       FORTWO ELECTRIC DRIVE    152
66             LINCOLN                     AVIATOR    117
25            CADILLAC                         ELR     76
68        LUCID MOTORS                   LUCID AIR     65
93              SUBARU                   CROSSTREK     58
65               LEXUS                          NX     33
64          LAND ROVER           RANGE ROVER SPORT     24
33              FISKER                       KARMA     19
43             GENESIS                        GV60     13
10     AZURE DYNAMICS   TRANSIT CONNECT ELECTRIC      7
100              TH!NK                        CITY      3
11             BENTLEY                    BENTAYGA      2

# in this analysis we again seee that tesla has huge amount of market
now-a-days
# by this text analysis you are able to understand our next plot

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
model_counts = df.groupby(['make',
'model']).size().reset_index(name='Count')
most_produced_models = model_counts.loc[model_counts.groupby(['make'])
['Count'].idxmax()]

plt.figure(figsize=(14, 8))
sns.barplot(data=most_produced_models, x='Count', y='model',
hue='make', dodge=False)

plt.title('Most Produced Model by Brand')
plt.xlabel('Count of Vehicles')
plt.ylabel('Model')
plt.show()
```



```
# wow such an amazing result so we clearly see that tesla model 3 has
huge demand in the market
# and nissan leaf has 2nd in this race
```

By the whole analysis some good insights find by me.

(1) there are many brands present in market but only some of them are making profit and still stand in market like :- Tesla,Nissan,Chervolet , etc....

(2) when i looked at range covered by vehicle then jaguar beat tesla so i think tesla will has to work on this problem .....

(3) only few countries are there which have a huge amount of electric vehicle market and and some countries has negligible market so i think brands should be increase marketing in that countries so that company will make profit .

Thank You

TASK - 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')

import pandas as pd
import plotly.express as px

df = pd.read_excel('/content/drive/MyDrive/dataset.xlsx')
ev_count_by_state =
df.groupby('State').size().reset_index(name='EV_Count')
fig = px.choropleth(ev_count_by_state,
                    locations='State',
                    locationmode="USA-states",
                    color='EV_Count',
                    scope="usa",
                    title="Number of EV Vehicles by State",
                    color_continuous_scale="Viridis")
fig.show()

import pandas as pd
import plotly.express as px

# Load the dataset
df = pd.read_('/content/drive/MyDrive/dataset.xlsx')

ev_count_by_state_year = df.groupby(['State', 'Model
Year']).size().reset_index(name='EV_Count')

# Creating a Choropleth map with animation and enhanced hover data
fig = px.choropleth(ev_count_by_state_year,
                    locations='State',
                    locationmode="USA-states",
                    color='EV_Count',
                    animation_frame='Model Year', # Animating by
'Model Year'
                    hover_name='State',
                    hover_data={'EV_Count': True, 'Model Year': True},
                    scope="usa",
                    title="Number of EV Vehicles by State Over the
Years",
                    color_continuous_scale="Viridis")

# Enhancing layout and visuals
fig.update_layout(
    geo=dict(
        lakecolor='rgb(255, 255, 255)', # Change lake color
        projection_scale=1  # Adjust the projection scale
    ),
    title_x=0.5,  # Center the title
    coloraxis_colorbar=dict(
        title="EV Count",  # Labeling the color bar
        ticks="outside"  # Show ticks on the outside
    )
)

# Display the enhanced map
fig.show()
```

#TASK -3

```
!pip install bar_chart_race
```

```
Requirement already satisfied: bar_chart_race in
/usr/local/lib/python3.10/dist-packages (0.1.0)
Requirement already satisfied: pandas>=0.24 in
/usr/local/lib/python3.10/dist-packages (from bar_chart_race) (2.2.2)
Requirement already satisfied: matplotlib>=3.1 in
/usr/local/lib/python3.10/dist-packages (from bar_chart_race) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1-
>bar_chart_race) (1.3.0)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1-
>bar_chart_race) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1-
>bar_chart_race) (4.54.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1-
>bar_chart_race) (1.4.7)
Requirement already satisfied: numpy>=1.20 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1-
>bar_chart_race) (1.26.4)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1-
>bar_chart_race) (24.1)
Requirement already satisfied: pillow>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1-
>bar_chart_race) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1-
>bar_chart_race) (3.1.4)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1-
>bar_chart_race) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=0.24-
>bar_chart_race) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.10/dist-packages (from pandas>=0.24-
>bar_chart_race) (2024.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7-
>matplotlib>=3.1->bar_chart_race) (1.16.0)
```

```
!pip install bar-chart-race
```

```
Collecting bar-chart-race
  Downloading bar_chart_race-0.1.0-py3-none-any.whl.metadata (4.2 kB)
Requirement already satisfied: pandas>=0.24 in
/usr/local/lib/python3.10/dist-packages (from bar-chart-race) (2.2.2)
Requirement already satisfied: matplotlib>=3.1 in
/usr/local/lib/python3.10/dist-packages (from bar-chart-race) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1->bar-
chart-race) (1.3.0)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1->bar-
chart-race) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1->bar-
chart-race) (4.54.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1->bar-
chart-race) (1.4.7)
Requirement already satisfied: numpy>=1.20 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1->bar-
chart-race) (1.26.4)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1->bar-
chart-race) (24.1)
Requirement already satisfied: pillow>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1->bar-
chart-race) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1->bar-
chart-race) (3.1.4)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.1->bar-
chart-race) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=0.24->bar-chart-
race) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in
```

```
/usr/local/lib/python3.10/dist-packages (from pandas>=0.24->bar-chart-
race) (2024.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7-
>matplotlib>=3.1->bar-chart-race) (1.16.0)
Downloading bar_chart_race-0.1.0-py3-none-any.whl (156 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 156.8/156.8 kB 3.0 MB/s eta
0:00:00
```

```
import bar_chart_race as bcr
ev_make_counts = df.groupby(['model year',
'make']).size().unstack().fillna(0)

bcr.bar_chart_race(
    df=ev_make_counts,
    filename='ev_make_racing_bar.mp4',
    orientation='h',
    sort='desc',
    title='EV Make Count Over the Years',
    steps_per_period=50,
    period_length=2000,
    period_label={'x': .95, 'y': .15, 'ha': 'right', 'va': 'center',
'size': 72, 'weight': 'semibold'},
    bar_kwargs={'alpha': .99, 'lw': 0},
    period_fmt='{x:.0f}',
)
```

```
/usr/local/lib/python3.10/dist-packages/bar_chart_race/
_make_chart.py:286: UserWarning: FixedFormatter should only be used
together with FixedLocator
  ax.set_yticklabels(self.df_values.columns)
/usr/local/lib/python3.10/dist-packages/bar_chart_race/_make_chart.py:
287: UserWarning: FixedFormatter should only be used together with
FixedLocator
  ax.set_xticklabels([max_val] * len(ax.get_xticks()))
```