Assignment 3 - Deadline: Oct 16, 2024, Wed 11pm

DSAI 510 Fall 2024

Complete the assignment below and upload both the .ipynb file and its pdf to https://moodle.bogazici.edu.tr by the deadline given above. The submission page on Moodle will close automatically after this date and time.

To make a pdf, this may work: Hit CMD+P or CTRL+P, and save it as PDF. You may also use other options from the File menu.

```
# Run this cell first

import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

# Set the display option to show all rows scrolling with a slider
pd.set_option('display.max_rows', None)
# To disable this, run the line below:
# pd.reset_option('display.max_rows')
```

Note:

In the problems below, if they ask "show the number of records that are nonzero", the answer is a number; so you don't need to show the records themselves. But if it asks, "show the records with NaN", it wants you to print those records (rows) containing NAN and other entries, not asking how many such records there are. So be careful about what you're asked.

Problem 1 (10 pts)

(a) Load the Ames house dataset from the file **train.csv**.

(b) Display the records with sale price greater than 500000 USD and LotFrontage less than 150 feet. Show only these columns: **Id**, **LotFrontage** and **SalePrice**.

(c) Print the list of all possible distinct values for the column **SaleCondition** for the records where sale price is greater than 500000 USD and LotFrontage is less than 150 feet.

(d) Create an interactive scatter plot of LotFrontage versus SalePrice, displaying only the records identified in the previous step. When hovering over the dots, the plot should display the **SaleCondition** in addition to **LotFrontage** and **SalePrice** information.

```
# Break your computations into multiple cells

# a
df = pd.read_csv("train.csv")
```

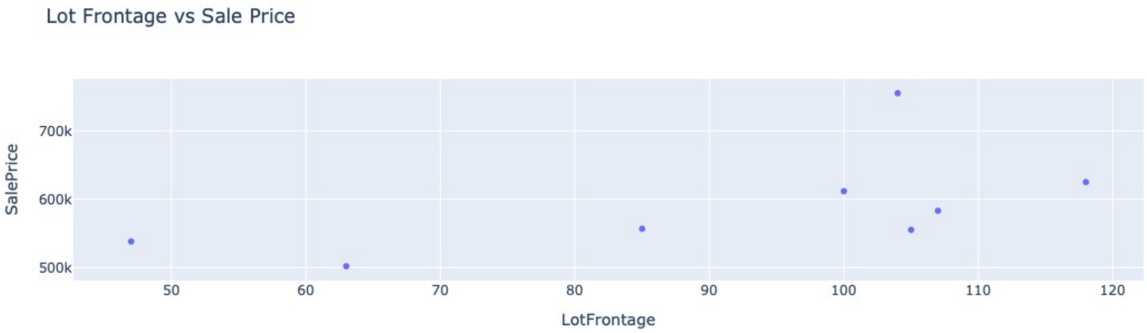
```
# b

cols = ["Id", "LotFrontage", "SalePrice"]

condition1 = df["SalePrice"] > 500000
condition2 = df["LotFrontage"] < 150

result_df = df[cols][condition2 & condition1]
display( result_df  )
```

```
        Id  LotFrontage  SalePrice
178    179         63.0     501837
440    441        105.0     555000
691    692        104.0     755000
769    770         47.0     538000
803    804        107.0     582933
898    899        100.0     611657
1046  1047         85.0     556581
1169  1170        118.0     625000
```

```
# c

print(df[condition1 & condition2]["SaleCondition"].unique())
```

```
['Partial' 'Normal']
```

```
# d
import plotly.express as px

fig = px.scatter(
    df[condition1 & condition2],
    x="LotFrontage",
    y="SalePrice",
    hover_name="Id",  # Display Id on hover
    hover_data={
        "SaleCondition": True  # Include SaleCondition in hover data
    },
    title="Lot Frontage vs Sale Price",
)

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



Problem 2 (10 pts)

(a) Display the list of neighborhood names and 'mean sale price' for those neighborhoods for the records whose SaleCondition is 'Normal'.

(b) Display the list of neighborhood names and the difference "max sale price - mean sale price" for each neighborhood. (Here "-" is for subtraction.)

(c) Recreate the boxplot comparing Neighborhood to SalePrice that we made in class. This time, order the neighborhoods based on their medians in ascending order, from left to right. In other words, the neighborhood with the lowest SalePrice median should be on the far left.

```
# a

df_mean_neigh = df[["Neighborhood", "SalePrice"]][df.SaleCondition ==
"Normal"].groupby("Neighborhood").mean("SalePrice")

display(df_mean_neigh)
```

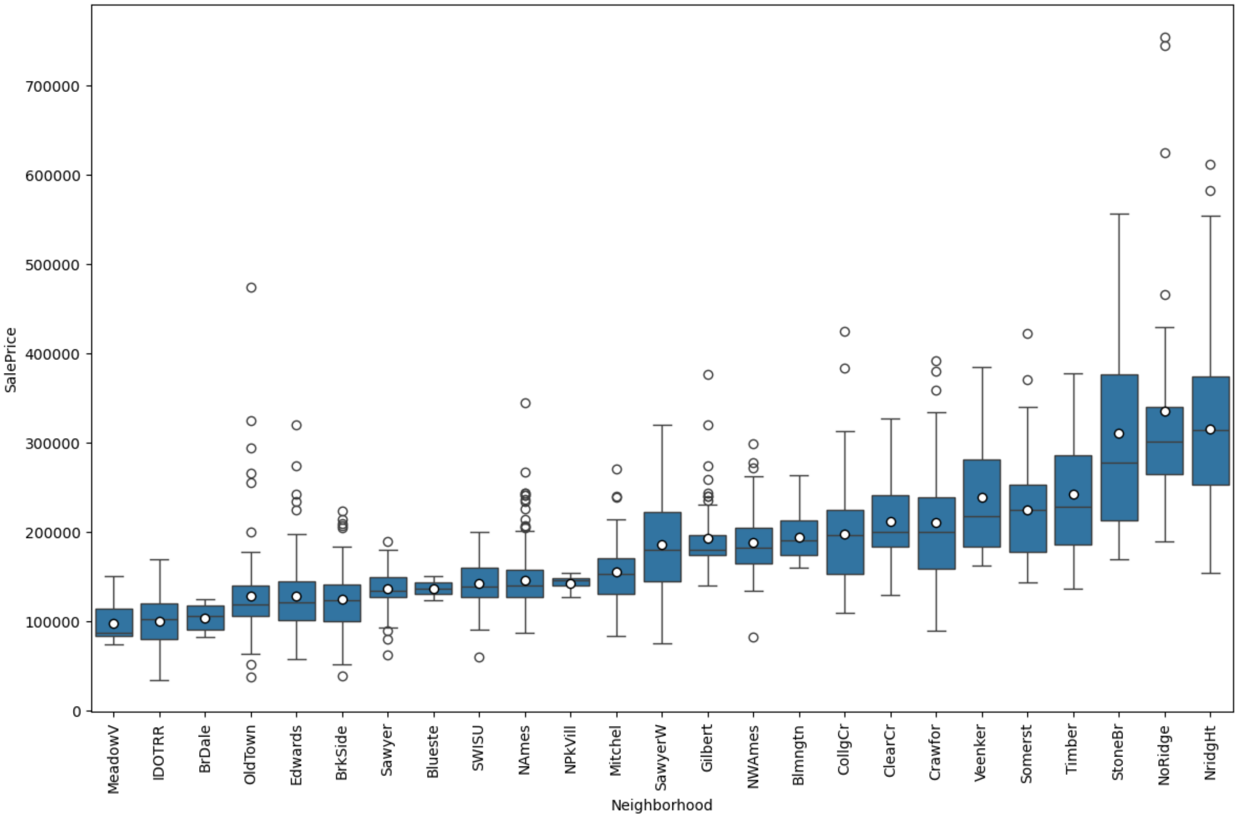
```
                    SalePrice
Neighborhood
Blmngtn         188977.083333
Blueste         137500.000000
BrDale          107916.666667
BrkSide         125588.425926
ClearCr         220993.000000
CollgCr         193877.224806
Crawfor         204863.651163
Edwards         127803.048780
Gilbert         189392.812500
IDOTRR          108575.862069
MeadowV          98987.500000
Mitchel         155410.714286
NAmes           147533.550505
NPkVill         143031.250000
NWAmes          193799.296875
NoRidge         328219.135135
```

```
NridgHt       285046.666667
OldTown       133173.489362
SWISU         139788.636364
Sawyer        136976.611940
SawyerW       191505.600000
Somerst       217760.653061
StoneBr       264870.375000
Timber        238484.392857
Veenker       238772.727273
```

```python
# b

df_max_neigh = df[["Neighborhood", "SalePrice"]][df.SaleCondition ==
"Normal"].groupby("Neighborhood").max("SalePrice")

display(df_max_neigh - df_mean_neigh)
```

```
                  SalePrice
Neighborhood
Blmngtn        45022.916667
Blueste        13500.000000
BrDale         17083.333333
BrkSide        97911.574074
ClearCr       107007.000000
CollgCr       119122.775194
Crawfor       176136.348837
Edwards       192196.951220
Gilbert       188107.187500
IDOTRR         60924.137931
MeadowV        52412.500000
Mitchel       115589.285714
NAmes         197466.449495
NPkVill        11968.750000
NWAmes        106000.703125
NoRidge       426780.864865
NridgHt       269953.333333
OldTown       341826.510638
SWISU          57211.363636
Sawyer         53023.388060
SawyerW       128494.400000
Somerst       122239.346939
StoneBr       273129.625000
Timber        140015.607143
Veenker       146227.272727
```

```python
# create the index based on the description of question
df_index=pd.DataFrame(
    df.groupby(['Neighborhood'])['SalePrice']
    .median()
    .sort_values(ascending=True)).index

# Create a boxplot ordered by neighborhood median sale prices
plt.figure(figsize=(12, 8))
sns.boxplot(x='Neighborhood', y='SalePrice', data=df, order=df_index,
showmeans=True,
            meanprops={"marker":"o", "markerfacecolor":"white",
"markeredgecolor":"black"},whis=1.5)

# Rotate on x-axis
plt.xticks(rotation=90)

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



Problem 3 (10 pts)

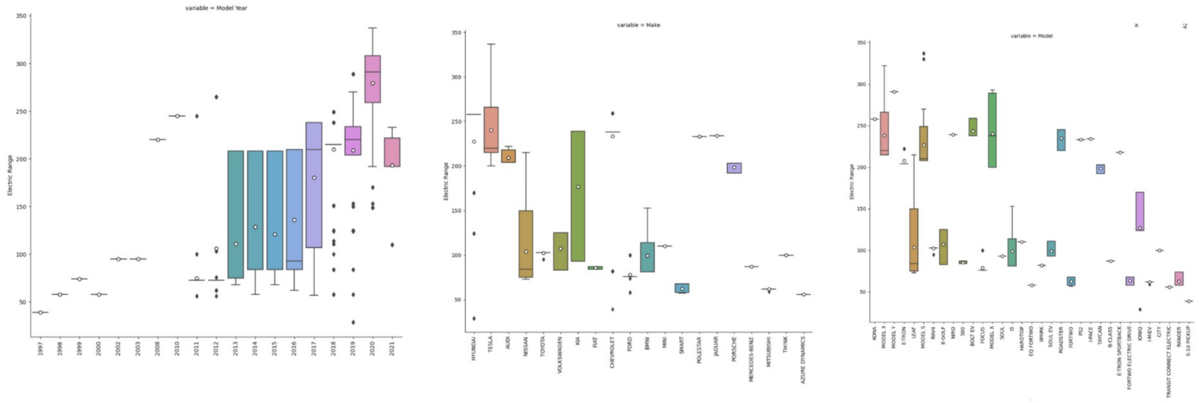
Here we'll show some of the houses on the map.

(a) Install the folium package with `!pip install folium`.

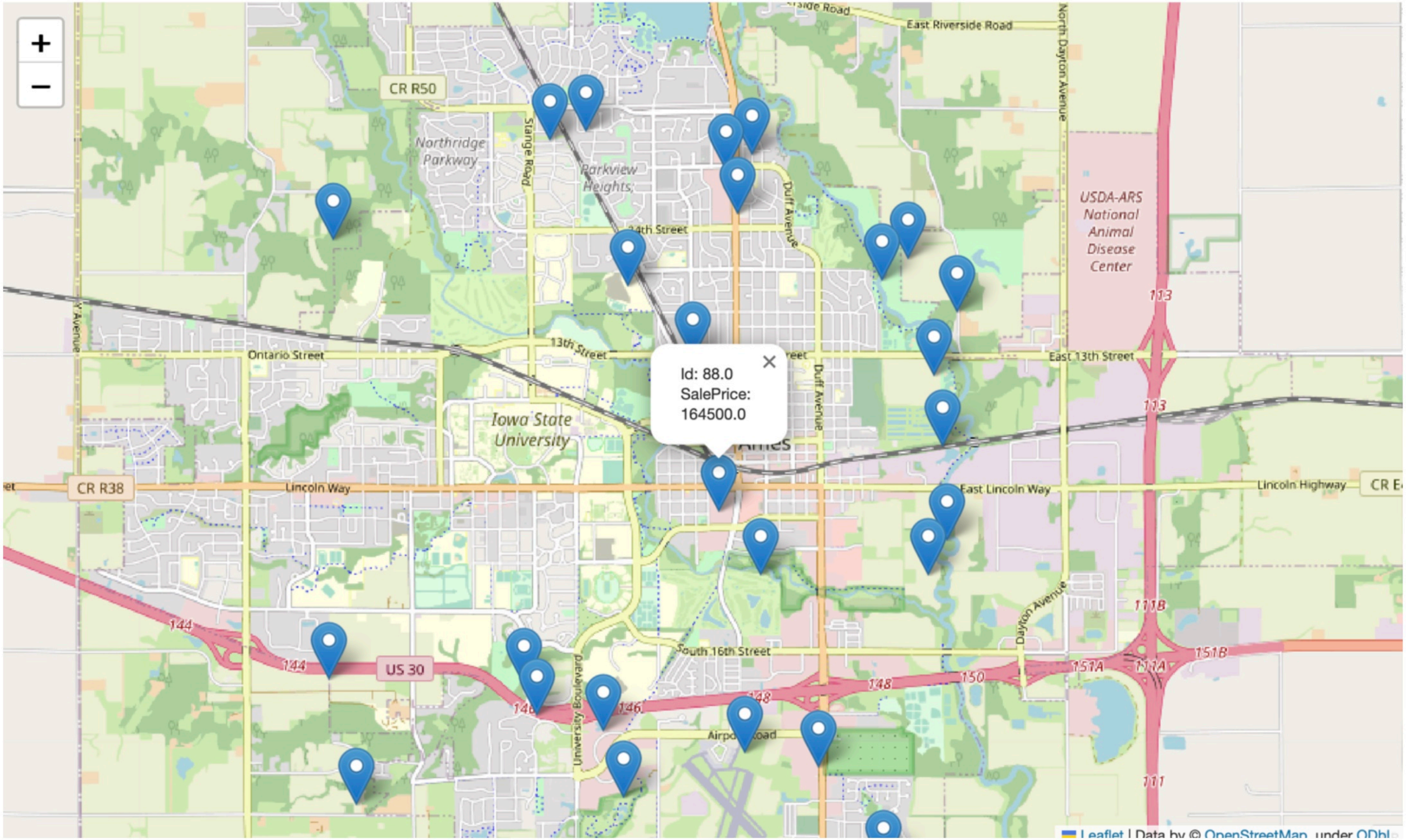
(b) Suppose your manager at the yellow website fromtheowner.com wants you to make a webpage showing houses on sale. Create the interactive map by using `folium` package to produce the map shown below for the 30 houses whose Id's and coordinates are given in

locations.csv. When you click on any pin on the map, the box should show the Id and SalePrice of that house as shown in the map below. You can find the SalePrice information in **train.csv**, and it's connected to **locations.csv** by the common column **Id**.

(Hint for folium usage: ChatGPT, Google, folium documentary...)



```
# !pip install folium

df_location = pd.read_csv("locations.csv")
df_train = pd.read_csv("train.csv")

# Merge two data on "Id" using left join
df_merged = df_location.merge(df_train[["Id", "SalePrice"]], on="Id",
how="left", suffixes=('_location', '_train'))


import folium

# Step 1: Create a map centered at a specific latitude and longitude
coordinates = list(df_merged.mean())
map_center = coordinates[1:3]  # Average of the latitude and longitude
for our dataset
map = folium.Map( location=map_center, zoom_start=13 )

# Step 2: Add markers to the map
def mapper(i):
    point_data = list(i)
    folium.Marker(
        location=[point_data[1], point_data[2]],
        popup=f'Id: {point_data[0]}\nSalePrice: {point_data[3]}',
        icon=folium.Icon(color='blue')
    ).add_to(map)

for i in range(len(df_merged)):
    mapper(df_merged.iloc[i])
```



Id: 88.0
SalePrice: 164500.0

```
# Step 3: Display the map
map

<folium.folium.Map at 0x151f859d0>
```

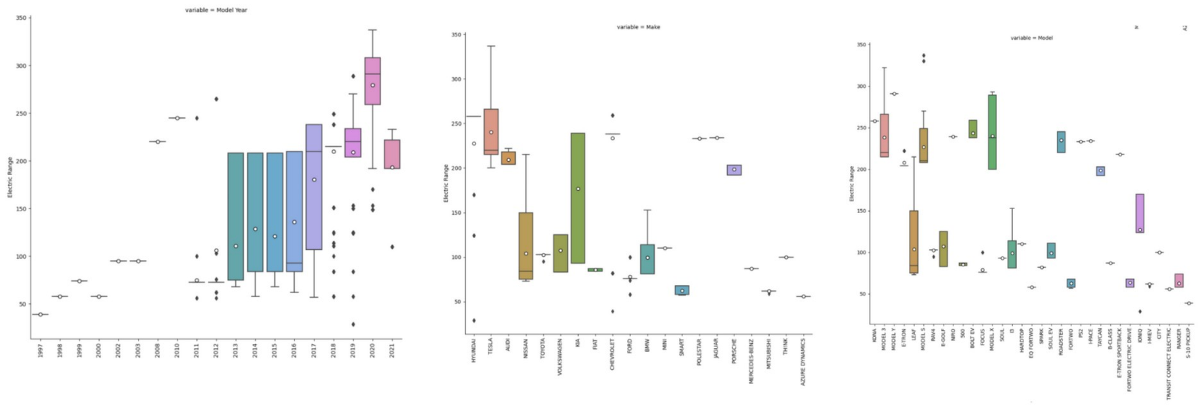
Problem 4 (10 pts)

(a) Load the data from **Electric_Vehicle_Population_Data.csv** into a dataframe `df`. Show the first five records (don't run `df` to show all records; jupyter notebook crashes as the data has 150482 rows).

(b) Make `df2` where it only includes the records whose **Electric Range** is 'Battery Electric Vehicle (BEV)' and **Electric Range** is greater than zero. (There should be ~47000 records satisfying these conditions; check the length of your final dataframe before proceeding!).

(c) Use `df2`to plot the histrogram of the column **Electric Range**

(d) Use `df2`to create three boxplots as we did in the class for Electric Range in the y-axis and 'Model Year', 'Make' and 'Model' categories in the x-axis. Use sns library and set `col_wrap=1, sharex=False, sharey=False, height=10` so that we don't get two or more boxplots side by side. Your plots should look like this (I put them side by side to save space here; yours will be stacked vertically in the Jupyter notebook):



(e) "What story do these boxplots convey?" To answer this question, write at least two observations for each of the three boxplots (in total at least six observations).

(f) Based on the box plots, does any of 'Model Year', 'Make' and 'Model' not determine the **Electric Range**, or do all of these determine it?

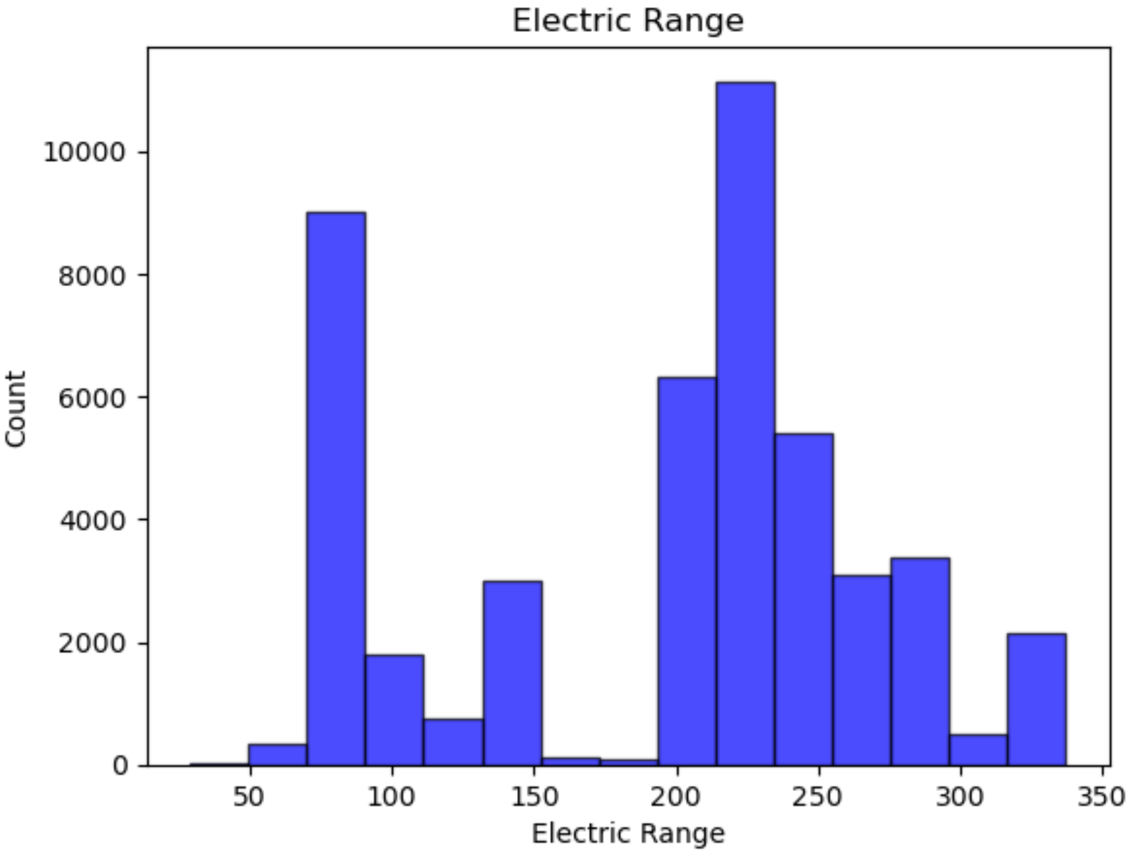
```
# a

df = pd.read_csv("Electric_Vehicle_Population_Data.csv")

df.head()
```

```
     County  Model Year     Make           Model  \
0      King        2020  HYUNDAI            KONA
1      King        2022     JEEP  GRAND CHEROKEE
2    Yakima        2023     JEEP  GRAND CHEROKEE
3      King        2018    TESLA         MODEL 3
4  Thurston        2018      BMW              I3

                    Electric Vehicle Type  Electric Range
0          Battery Electric Vehicle (BEV)             258
1  Plug-in Hybrid Electric Vehicle (PHEV)              25
2  Plug-in Hybrid Electric Vehicle (PHEV)              25
3          Battery Electric Vehicle (BEV)             215
4  Plug-in Hybrid Electric Vehicle (PHEV)              97
```

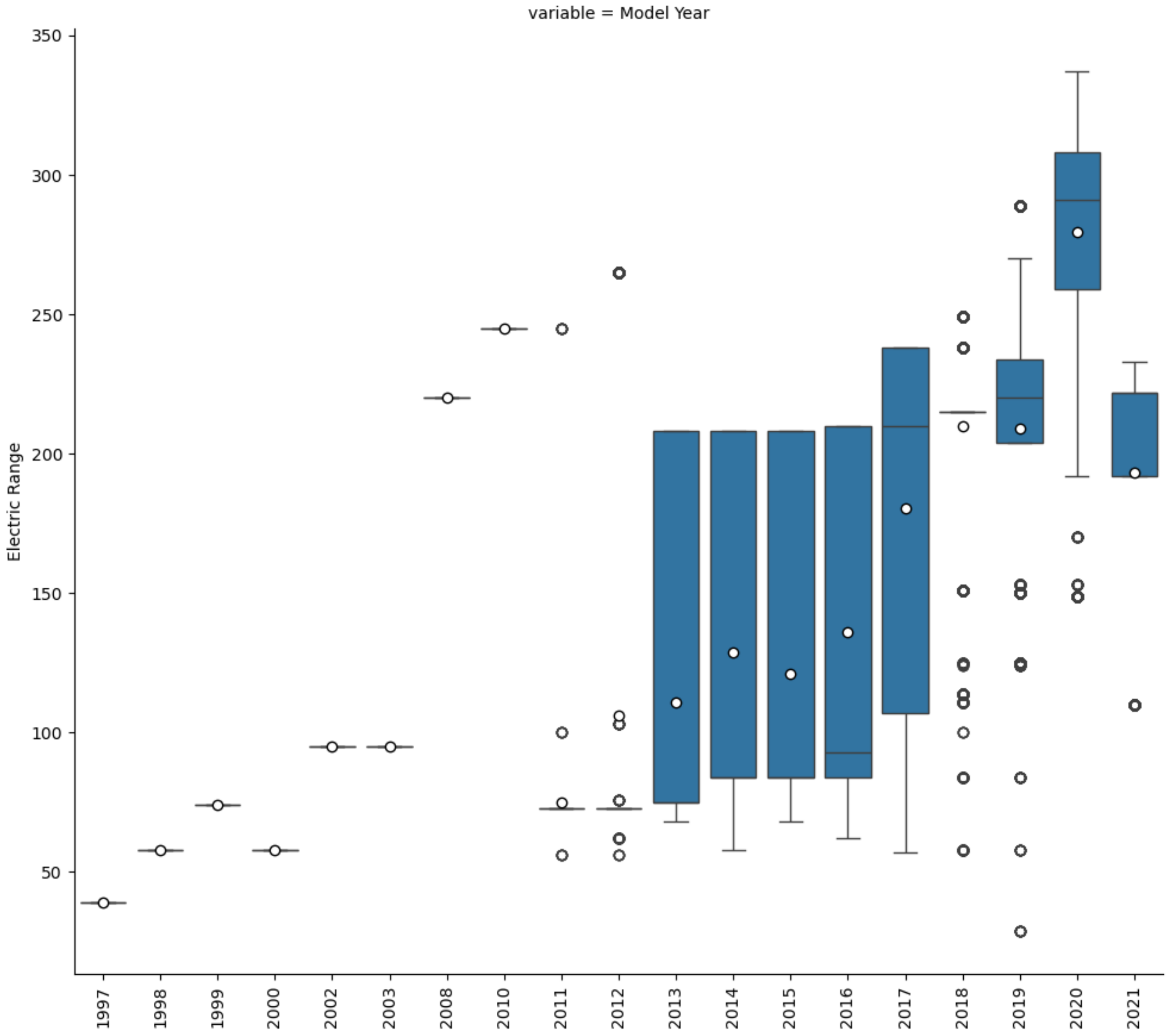
```python
# b

df2 = df[(df["Electric Vehicle Type"] == "Battery Electric Vehicle
(BEV)") & (df["Electric Range"] > 0)]

print(len(df2))
```

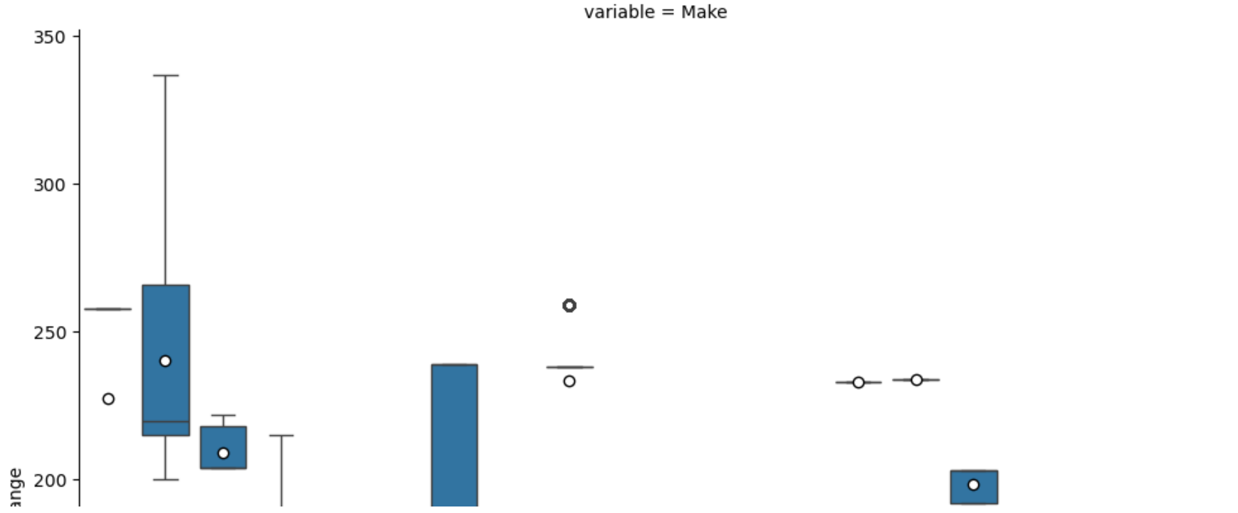
```
47109
```

```python
# c

import matplotlib.pyplot as plt


# Histogram for column 'Electric Range'
df2["Electric Range"].plot(kind='hist', bins=15, alpha=0.7,
color='blue', edgecolor='black')
plt.title('Electric Range')
plt.xlabel('Electric Range')
plt.ylabel('Count')


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



```
# d

import matplotlib.pyplot as plt
import seaborn as sns



x_ = 'Electric Range'
y_ = ['Model Year', 'Make', 'Model']

# Define boxplot fnc we'll use below
def boxplot(x, y, **kwargs):
    # lines within the boxes will show medians, white dots will show
means
    sns.boxplot(x=x, y=y,showmeans=True,meanprops={"marker":"o",
"markerfacecolor":"white", "markeredgecolor":"black"})
    x=plt.xticks(rotation=90)

# Taken from class notes
# It takes original data and melt with x_ and y_ values
melted_categorical = pd.melt(df2, id_vars=x_, value_vars=y_)
g = sns.FacetGrid(melted_categorical, col="variable",  col_wrap=1,
sharex=False, sharey=False, height=10)
g = g.map(boxplot, "value", x_)
```


variable = Model Year
Electric Range
350
300
250
200
150
100
50
1997
1998
1999
2000
2002
2003
2008
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021


variable = Make
ange
350
300
250
200

```
# e

# Plot 1

# Observation 1
    # Technological advancement is seen clearly, in the first plot, as
the average range is increased.

# Observation 2
    # There is an instant increase in the difference Q3 - Q1 after
2013. So, there is an increase on the variety of electric vehicles in
the market


# Plot 2

# Observation 1
    # There is no corrolation between Make. Some Brands can make
better cars in terms of range and some do not. Still some are better
than other like Tesla
# Observation 2
    # The boxplots indicates that the data is left skewed mostly.
Because brands created middle range vehicles usually.
    # Majority of their work goes to a focused model to increase more
ranged vehicles.


# Plot 3

# Observation 1
    # If the model is heavy then the range falls like with Ranger or
Pickup.
# Observation 2
    # TESLA has one range option only in MODEL Y. This is like its
special car.

# f
    # There is a corrolation between Electric Range with Year and
Model, yet Make needs more debate. Mean for Make plots does not
preserve the increase in the Range
```