EDA Project 2 - Analysis on Electric Vehicles

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [2]:
```
path=r"C:\Users\Sruth\Downloads\dataset.csv"
df=pd.read_csv(path)
df
```

Out[2]:

	VIN (1-10)	County	City	State	Postal Code	Model Year	Make	Mode
0	JTMEB3FV6N	Monroe	Key West	FL	33040	2022	TOYOTA	RAV4 PRIM
1	1G1RD6E45D	Clark	Laughlin	NV	89029	2013	CHEVROLET	VOL
2	JN1AZ0CP8B	Yakima	Yakima	WA	98901	2011	NISSAN	LEA
3	1G1FW6S08H	Skagit	Concrete	WA	98237	2017	CHEVROLET	BOL EV
4	3FA6P0SU1K	Snohomish	Everett	WA	98201	2019	FORD	FUSION
...	...	...	...	...	...	...	...	.
112629	7SAYGDEF2N	King	Duvall	WA	98019	2022	TESLA	MODE Y
112630	1N4BZ1CP7K	San Juan	Friday Harbor	WA	98250	2019	NISSAN	LEA
112631	1FMCU0KZ4N	King	Vashon	WA	98070	2022	FORD	ESCAPE

	VIN (1-10)	County	City	State	Postal Code	Model Year	Make	Mode
112632	KNDCD3LD4J	King	Covington	WA	98042	2018	KIA	NIRC
112633	YV4BR0CL8N	King	Covington	WA	98042	2022	VOLVO	XC9(

112634 rows × 17 columns

which determines how many rows and columns present in dataframe

In [3]:
```
# Gives number of rows(112634) and columns(17) in a dataframe

df.shape
```

Out[3]: (112634, 17)

In [4]:
```
# Size of dataframe is 1914778


df.size
```

Out[4]: 1914778

In [5]:
```
#Gives total information about the dataframe

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
```

In [6]:
```
# Gives the which type of columns are prsent in dataframe
df.dtypes
```

Out[6]:
```
VIN (1-10)                                           object
County                                               object
City                                                 object
State                                                object
Postal Code                                           int64
Model Year                                            int64
Make                                                 object
Model                                                object
Electric Vehicle Type                                object
Clean Alternative Fuel Vehicle (CAFV) Eligibility    object
Electric Range                                        int64
Base MSRP                                             int64
Legislative District                                float64
DOL Vehicle ID                                        int64
Vehicle Location                                     object
Electric Utility                                     object
2020 Census Tract                                     int64
dtype: object
```

categorical columns

In [7]:
```
cat_cols=df.select_dtypes(include='object').columns
cat_cols
```

Out[7]:
```
Index(['VIN (1-10)', 'County', 'City', 'State', 'Make', 'Model',
       'Electric Vehicle Type',
       'Clean Alternative Fuel Vehicle (CAFV) Eligibility', 'Vehicle Location',
       'Electric Utility'],
      dtype='object')
```

Numerical columns

In [8]:
```
num_cols= df.select_dtypes(exclude='object').columns
num_cols
```

Out[8]:
```
Index(['Postal Code', 'Model Year', 'Electric Range', 'Base MSRP',
       'Legislative District', 'DOL Vehicle ID', '2020 Census Tract'],
      dtype='object')
```

In [9]:
```
# len gives that how many number of columns
len(num_cols)
```

Out[9]:
```
7
```

In [10]:
```
# By default gives first 5 rows.

df.head()
```

Out[10]:

	VIN (1-10)	County	City	State	Postal Code	Model Year	Make	Model	Elect Vehi Ty
0	JTMEB3FV6N	Monroe	Key West	FL	33040	2022	TOYOTA	RAV4 PRIME	Plug Hyb Elect Vehi (PHE
1	1G1RD6E45D	Clark	Laughlin	NV	89029	2013	CHEVROLET	VOLT	Plug Hyb Elect Vehi (PHE
2	JN1AZ0CP8B	Yakima	Yakima	WA	98901	2011	NISSAN	LEAF	Batte Elect Vehi (BE
3	1G1FW6S08H	Skagit	Concrete	WA	98237	2017	CHEVROLET	BOLT EV	Batte Elect Vehi (BE
4	3FA6P0SU1K	Snohomish	Everett	WA	98201	2019	FORD	FUSION	Plug Hyb Elect Vehi (PHE

In [11]:
```
# By deafult gives last 5 rows
df.tail()
```

Out[11]:

	VIN (1-10)	County	City	State	Postal Code	Model Year	Make	Model	Elect Vehi Ty
112629	7SAYGDEF2N	King	Duvall	WA	98019	2022	TESLA	MODEL Y	Batt Elec Vehi (BI
112630	1N4BZ1CP7K	San Juan	Friday Harbor	WA	98250	2019	NISSAN	LEAF	Batt Elec Vehi (BI
112631	1FMCU0KZ4N	King	Vashon	WA	98070	2022	FORD	ESCAPE	Plug Hyb Elec Vehi (PHI
112632	KNDCD3LD4J	King	Covington	WA	98042	2018	KIA	NIRO	Plug Hyb Elec Vehi (PHI
112633	YV4BR0CL8N	King	Covington	WA	98042	2022	VOLVO	XC90	Plug Hyb Elec Vehi (PHI

Checking whether missing values are present

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
VIN (1-10)                                             0
County                                                 0
City                                                   0
State                                                  0
Postal Code                                            0
Model Year                                             0
Make                                                   0
Model                                                 20
Electric Vehicle Type                                  0
Clean Alternative Fuel Vehicle (CAFV) Eligibility      0
Electric Range                                         0
Base MSRP                                              0
Legislative District                                 286
DOL Vehicle ID                                         0
Vehicle Location                                      24
Electric Utility                                     443
2020 Census Tract                                      0
dtype: int64
```

in above set we contains missing values in some columns

Filling the missing values

In [13]:
```
cat_cols
```

Out[13]:
```
Index(['VIN (1-10)', 'County', 'City', 'State', 'Make', 'Model',
       'Electric Vehicle Type',
       'Clean Alternative Fuel Vehicle (CAFV) Eligibility', 'Vehicle Location',
       'Electric Utility'],
      dtype='object')
```

In [14]:
```
num_cols
```

Out[14]:
```
Index(['Postal Code', 'Model Year', 'Electric Range', 'Base MSRP',
       'Legislative District', 'DOL Vehicle ID', '2020 Census Tract'],
      dtype='object')
```

Filling missing vlaues by using mode

In [15]:
```
df=pd.read_csv(path)
Model=df['Model'].mode()
df['Model'].fillna(Model[0],inplace=True)

Legislative_District=df['Legislative District'].mode()
df['Legislative District'].fillna(Legislative_District[0],inplace=True)


Vehicle_Location=df['Vehicle Location'].mode()
df['Vehicle Location'].fillna(Vehicle_Location[0],inplace=True)

Electric_Utility =df['Electric Utility'].mode()
df['Electric Utility'].fillna(Electric_Utility[0],inplace=True )
```

In [16]:
```
df
```

Out[16]:

	VIN (1-10)	County	City	State	Postal Code	Model Year	Make	Mode
0	JTMEB3FV6N	Monroe	Key West	FL	33040	2022	TOYOTA	RAV4 PRIM
1	1G1RD6E45D	Clark	Laughlin	NV	89029	2013	CHEVROLET	VOL
2	JN1AZ0CP8B	Yakima	Yakima	WA	98901	2011	NISSAN	LEA
3	1G1FW6S08H	Skagit	Concrete	WA	98237	2017	CHEVROLET	BOL EV
4	3FA6P0SU1K	Snohomish	Everett	WA	98201	2019	FORD	FUSION
...	...	...	...	...	...	...	...	.
112629	7SAYGDEF2N	King	Duvall	WA	98019	2022	TESLA	MODE Y
112630	1N4BZ1CP7K	San Juan	Friday Harbor	WA	98250	2019	NISSAN	LEA
112631	1FMCU0KZ4N	King	Vashon	WA	98070	2022	FORD	ESCAPE

	VIN (1-10)	County	City	State	Postal Code	Model Year	Make	Mode
112632	KNDCD3LD4J	King	Covington	WA	98042	2018	KIA	NIRO
112633	YV4BR0CL8N	King	Covington	WA	98042	2022	VOLVO	XC90

112634 rows × 17 columns

Here missing values all are filled

In [17]:
```
# all missing values are filled , 0 indicates all values are filled
df.isnull().sum()
```

Out[17]:
```
VIN (1-10)                                         0
County                                             0
City                                               0
State                                              0
Postal Code                                        0
Model Year                                         0
Make                                               0
Model                                              0
Electric Vehicle Type                              0
Clean Alternative Fuel Vehicle (CAFV) Eligibility  0
Electric Range                                     0
Base MSRP                                          0
Legislative District                               0
DOL Vehicle ID                                     0
Vehicle Location                                   0
Electric Utility                                   0
2020 Census Tract                                  0
dtype: int64
```

Categorical columns analysis

In [18]:
```
cat_cols
```

Out[18]:
```
Index(['VIN (1-10)', 'County', 'City', 'State', 'Make', 'Model',
       'Electric Vehicle Type',
       'Clean Alternative Fuel Vehicle (CAFV) Eligibility', 'Vehicle Location',
       'Electric Utility'],
      dtype='object')
```

In [19]:
```
df['County'].nunique()
```

Out[19]: 165

In [20]:
```
df['County'].value_counts()
```

Out[20]:
```
County
King          59000
Snohomish     12434
Pierce         8535
Clark          6689
Thurston       4126
              ...
Pinal             1
Elmore            1
Portsmouth        1
Kings             1
Kootenai          1
Name: count, Length: 165, dtype: int64
```

In [21]:
```
df['City'].value_counts()
```

Out[21]:
```
City
Seattle         20305
Bellevue         5921
Redmond          4201
Vancouver        4013
Kirkland         3598
                ...
Hartline            1
Gaithersburg        1
El Paso             1
Klickitat           1
Worley              1
Name: count, Length: 629, dtype: int64
```

In [22]:
```
df['City'].nunique()
```

Out[22]: 629

Univariate analysis

In [23]:
```
df
```

Out[23]:

	VIN (1-10)	County	City	State	Postal Code	Model Year	Make	Mode
0	JTMEB3FV6N	Monroe	Key West	FL	33040	2022	TOYOTA	RAV4 PRIM
1	1G1RD6E45D	Clark	Laughlin	NV	89029	2013	CHEVROLET	VOL
2	JN1AZ0CP8B	Yakima	Yakima	WA	98901	2011	NISSAN	LEA
3	1G1FW6S08H	Skagit	Concrete	WA	98237	2017	CHEVROLET	BOL EV
4	3FA6P0SU1K	Snohomish	Everett	WA	98201	2019	FORD	FUSION
...	...	...	...	...	...	...	...	.
112629	7SAYGDEF2N	King	Duvall	WA	98019	2022	TESLA	MODE Y
112630	1N4BZ1CP7K	San Juan	Friday Harbor	WA	98250	2019	NISSAN	LEA
112631	1FMCU0KZ4N	King	Vashon	WA	98070	2022	FORD	ESCAPE

	VIN (1-10)	County	City	State	Postal Code	Model Year	Make	Mode
112632	KNDCD3LD4J	King	Covington	WA	98042	2018	KIA	NIRC
112633	YV4BR0CL8N	King	Covington	WA	98042	2022	VOLVO	XC9(

112634 rows × 17 columns

describe

In [24]:
```
df.describe()
```

Out[24]:

	Postal Code	Model Year	Electric Range	Base MSRP	Legislative District	DC
count	112634.000000	112634.000000	112634.000000	112634.000000	112634.000000	1.12
mean	98156.226850	2019.003365	87.812987	1793.439681	29.834029	1.99
std	2648.733064	2.892364	102.334216	10783.753486	14.692675	9.39
min	1730.000000	1997.000000	0.000000	0.000000	1.000000	4.77
25%	98052.000000	2017.000000	0.000000	0.000000	18.000000	1.48
50%	98119.000000	2020.000000	32.000000	0.000000	34.000000	1.92
75%	98370.000000	2022.000000	208.000000	0.000000	43.000000	2.19
max	99701.000000	2023.000000	337.000000	845000.000000	49.000000	4.79

Skewness for numerical columns

In [25]:
```
for i in num_cols:
    skew=round(df[i].skew(),2)
    print(f"skew of  column {i} is '{skew}'")
```

```
skew of  column Postal Code is '-27.96'
skew of  column Model Year is '-0.82'
skew of  column Electric Range is '0.82'
skew of  column Base MSRP is '10.1'
skew of  column Legislative District is '-0.55'
skew of  column DOL Vehicle ID is '1.15'
skew of  column 2020 Census Tract is '-25.01'
```

kurtosis of numerical columns

In [26]:
```
for i in num_cols:
    kurt=round(df[i].kurt(),2)
    print(f"kutosis of  column {i} is '{kurt}'")
```

```
kutosis of  column Postal Code is '820.87'
kutosis of  column Model Year is '-0.0'
kutosis of  column Electric Range is '-0.88'
kutosis of  column Base MSRP is '371.7'
kutosis of  column Legislative District is '-0.98'
kutosis of  column DOL Vehicle ID is '2.47'
kutosis of  column 2020 Census Tract is '645.9'
```

Covariance of matrix

In [27]:
```
df.cov(numeric_only=True)
```

Out[27]:

	Postal Code	Model Year	Electric Range	Base MSRP	Legislative District
Postal Code	7.015787e+06	-3.436157e+01	1.044193e+02	3.288626e+04	-3.077283e+03
Model Year	-3.436157e+01	8.365770e+00	-8.537278e+01	-7.146681e+03	4.407048e-01
Electric Range	1.044193e+02	-8.537278e+01	1.047229e+04	9.382912e+04	3.649108e+01
Base MSRP	3.288626e+04	-7.146681e+03	9.382912e+04	1.162893e+08	1.953132e+03
Legislative District	-3.077283e+03	4.407048e-01	3.649108e+01	1.953132e+03	2.158747e+02
DOL Vehicle ID	8.376140e+08	-1.856498e+07	9.311599e+07	5.110403e+08	-2.528649e+06
2020 Census Tract	2.255502e+12	3.509447e+06	1.256066e+08	1.794588e+10	-8.429475e+08

Correlation matrix

In [28]:
```
correlation_data=df.corr(numeric_only=True)
correlation_data
```

Out[28]:

	Postal Code	Model Year	Electric Range	Base MSRP	Legislative District	DOL Vehicle ID	2020 Census Tract
Postal Code	1.000000	-0.004485	0.000385	0.001151	-0.079073	0.003365	0.501170
Model Year	-0.004485	1.000000	-0.288433	-0.229130	0.010370	-0.068295	0.000714
Electric Range	0.000385	-0.288433	1.000000	0.085025	0.024270	0.009682	0.000722
Base MSRP	0.001151	-0.229130	0.085025	1.000000	0.012327	0.000504	0.000979
Legislative District	-0.079073	0.010370	0.024270	0.012327	1.000000	-0.001831	-0.033766
DOL Vehicle ID	0.003365	-0.068295	0.009682	0.000504	-0.001831	1.000000	0.002754
2020 Census Tract	0.501170	0.000714	0.000722	0.000979	-0.033766	0.002754	1.000000

Heatmap visualzation of correlation matrix

correlation tells about how much relation between two variables

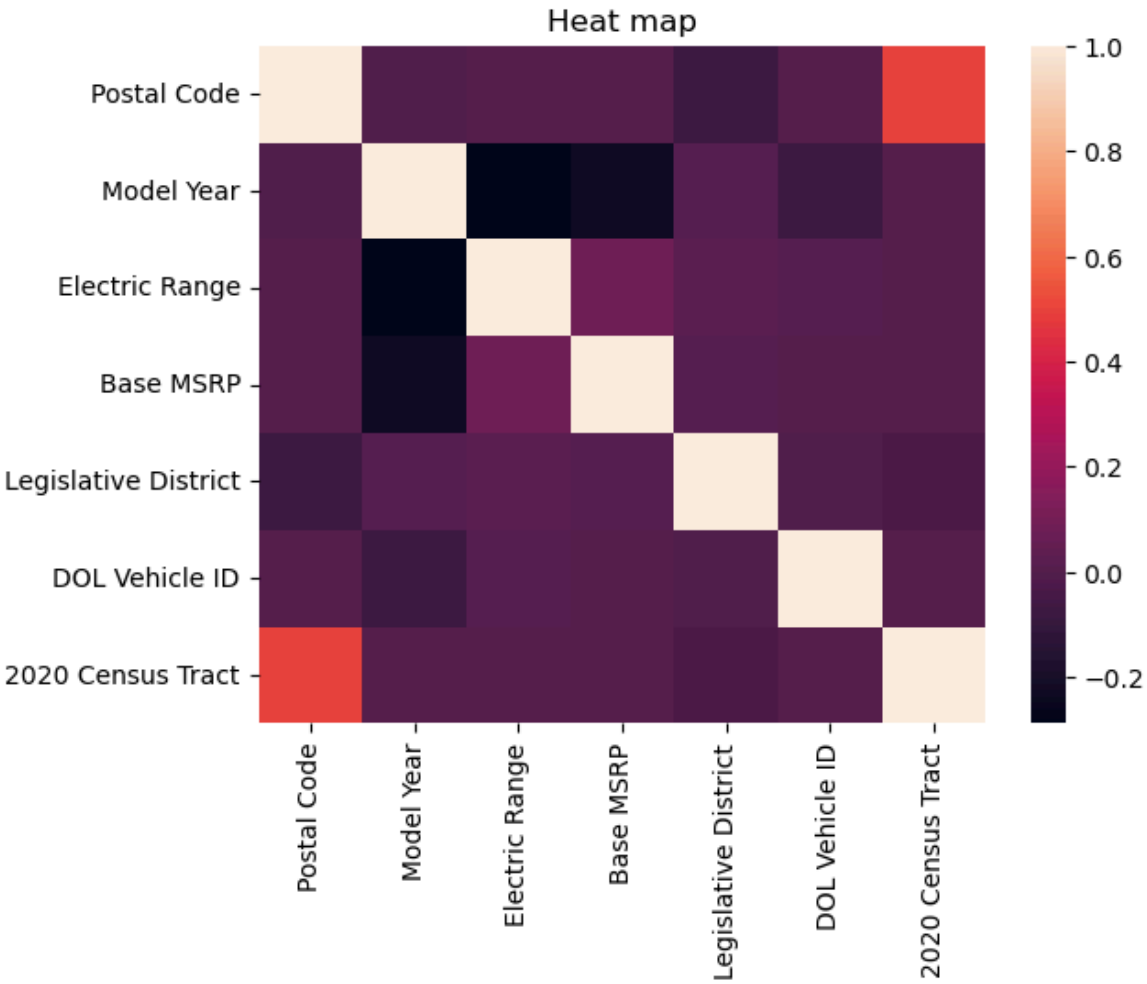
denotes with r , r varies -1 to +1

-1 to 0 indicates negative relation

0 to 1 indivates positive relation

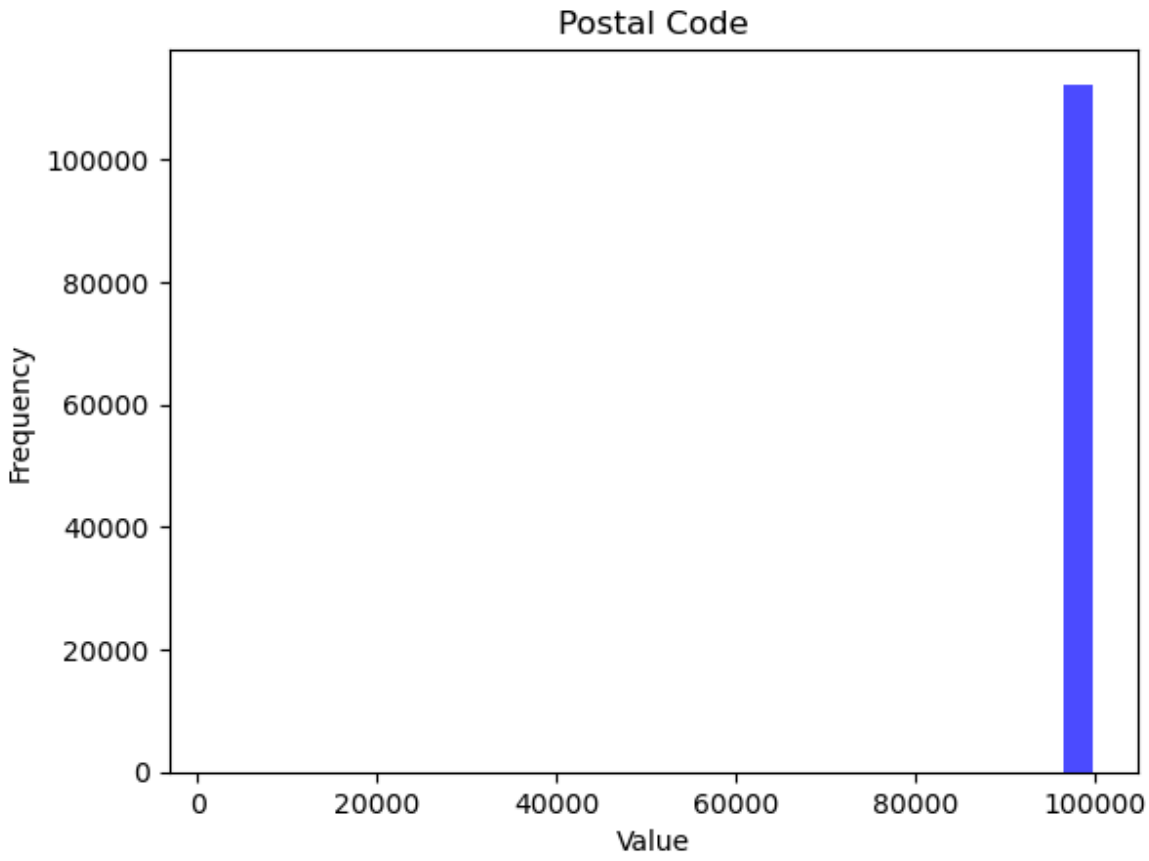
0 indicates no relation

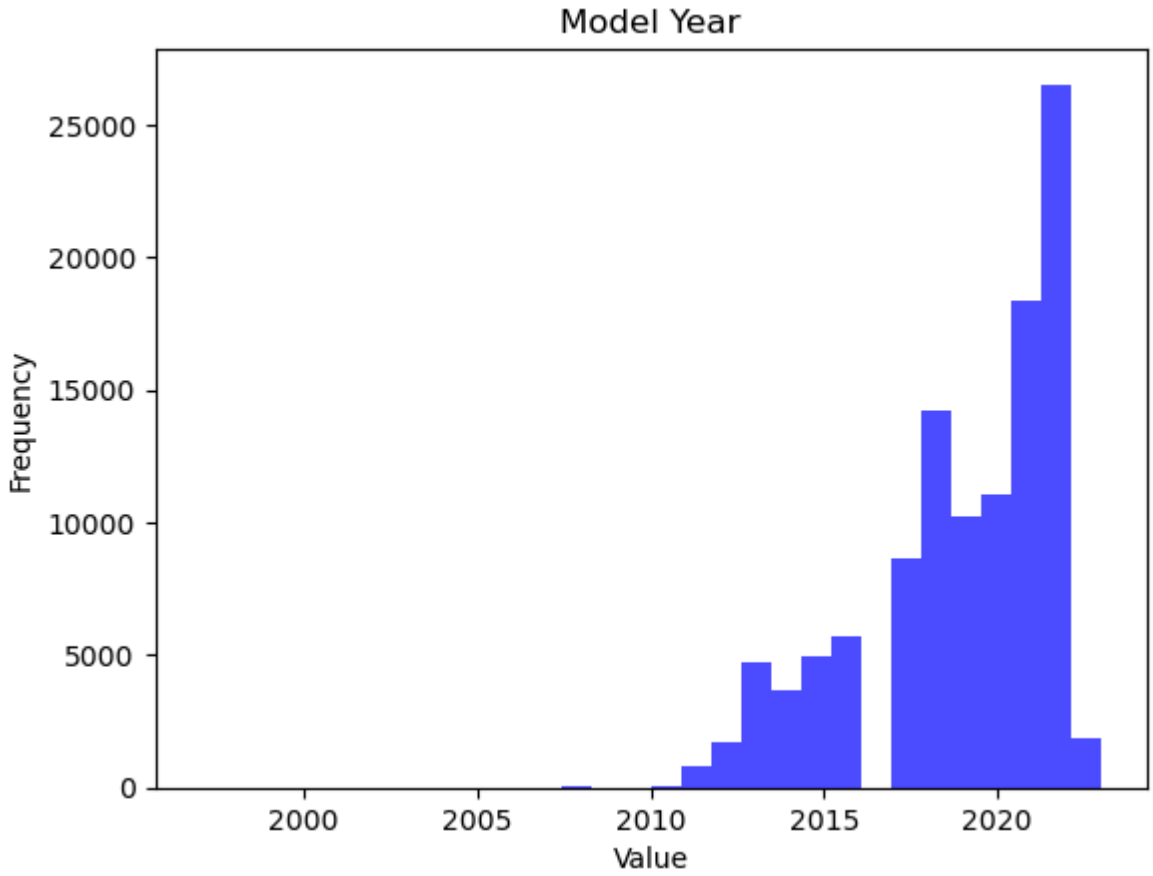
In [29]:
```
sns.heatmap(correlation_data)
plt.title('Heat map')
plt.show()
```

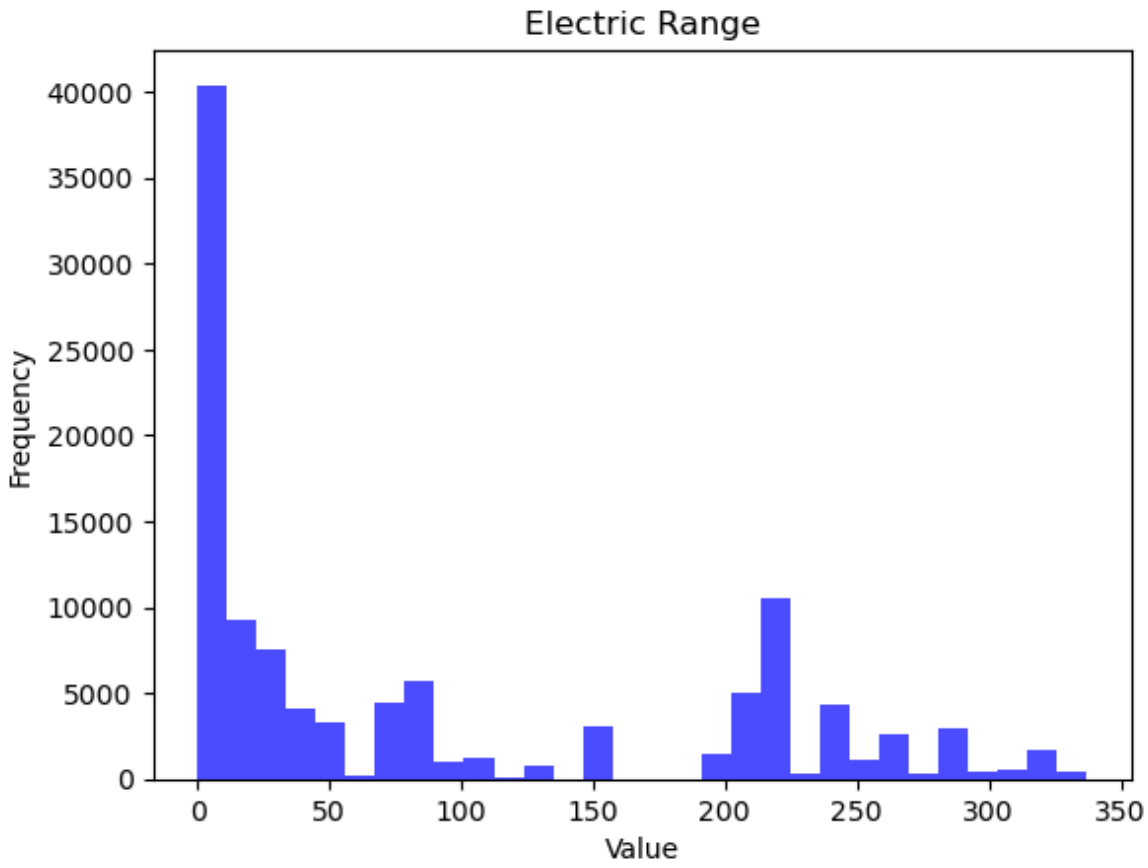


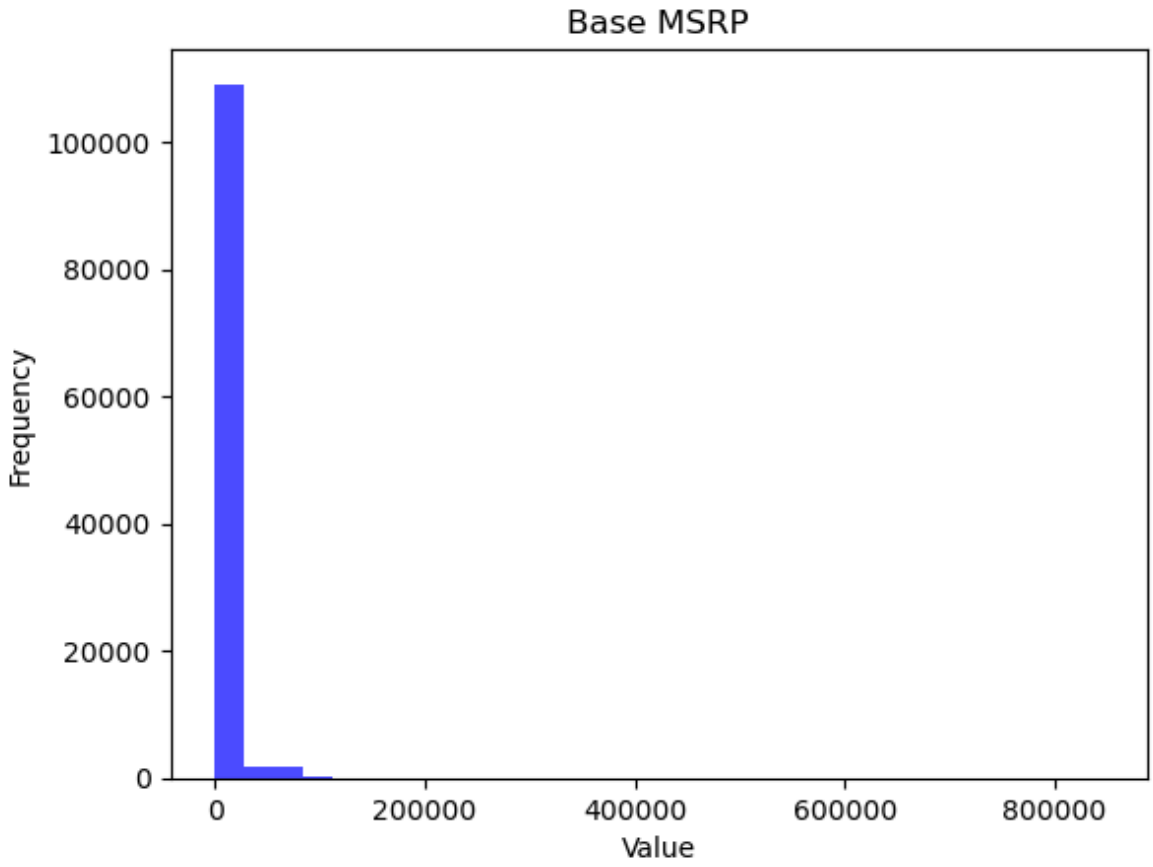
Histogram analysis for numerical columns

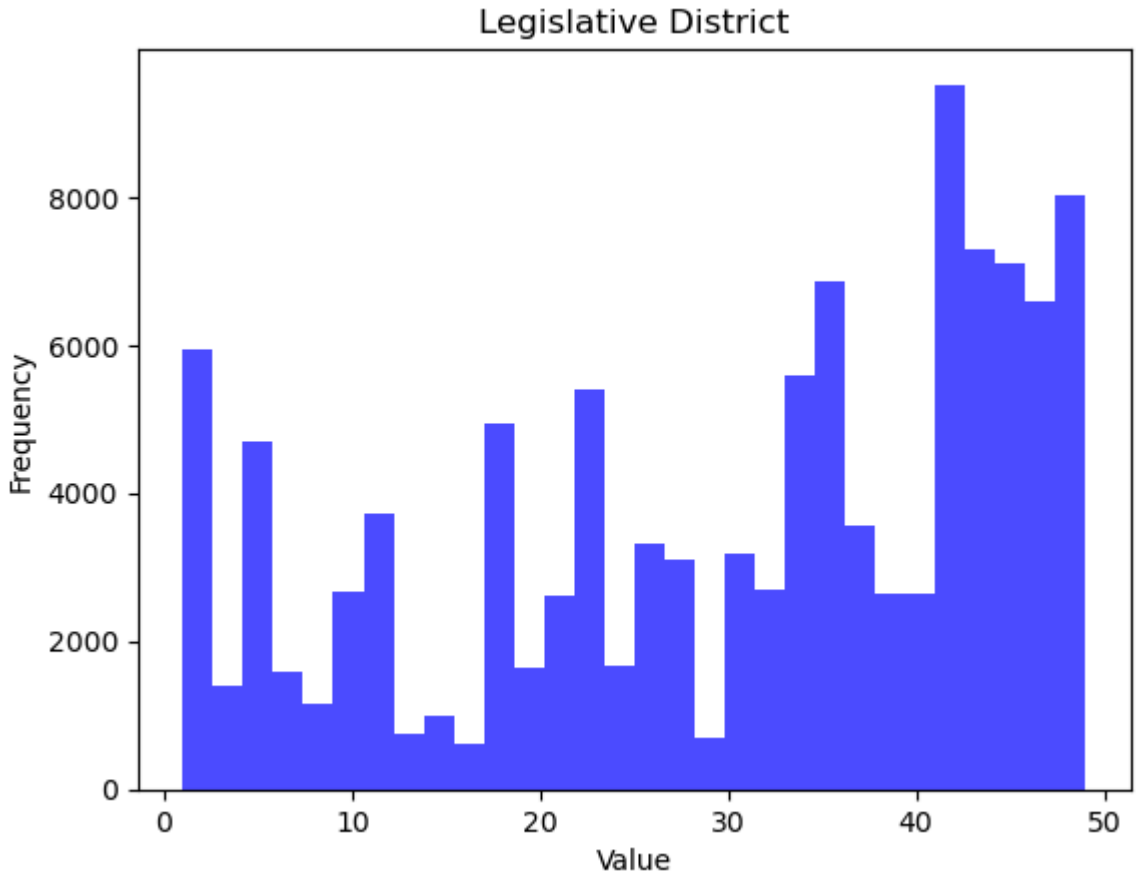
In [30]:
```
for i in num_cols:
    plt.hist(df[i],bins=30,alpha=0.7,color='blue')
    plt.title(i)
    plt.xlabel('Value')
    plt.ylabel("Frequency")
    plt.show()
```

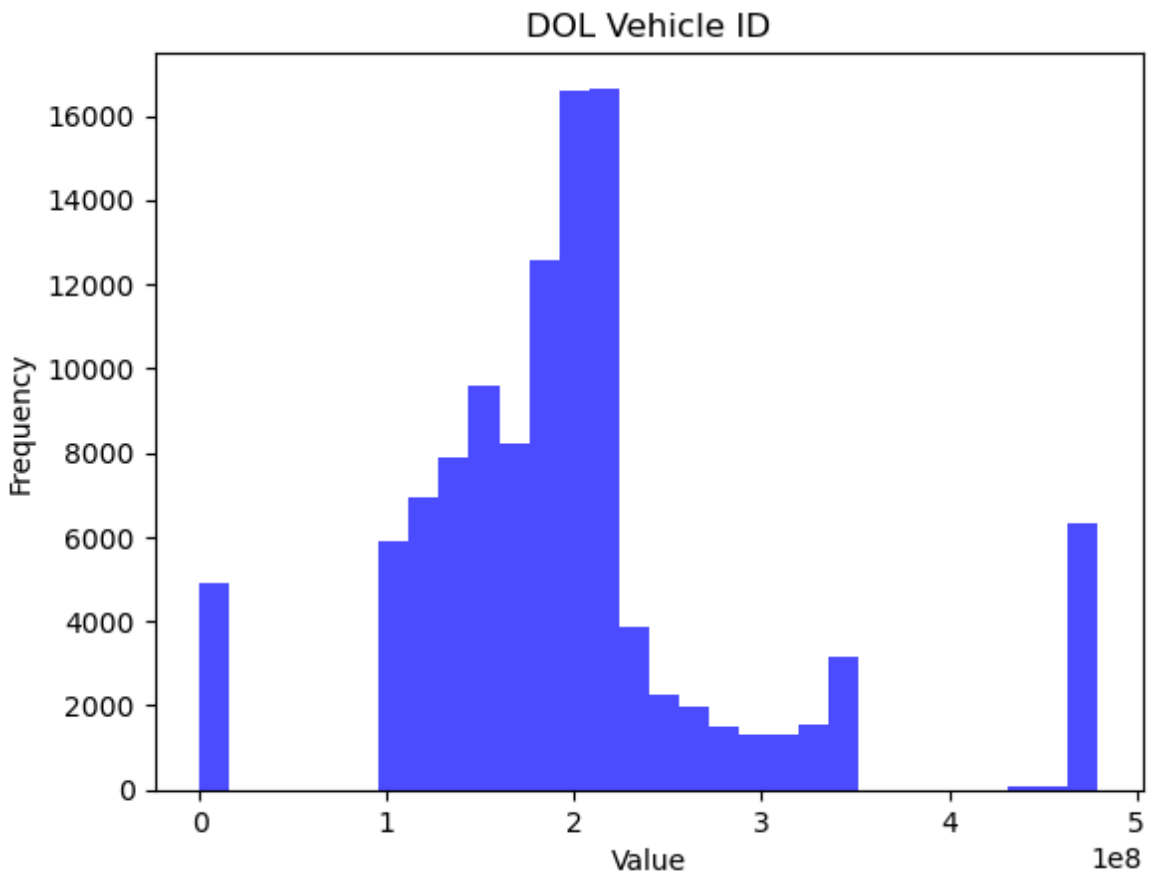

Postal Code
Frequency
100000
80000
60000
40000
20000
0
0
20000
40000
60000
80000
100000
Value

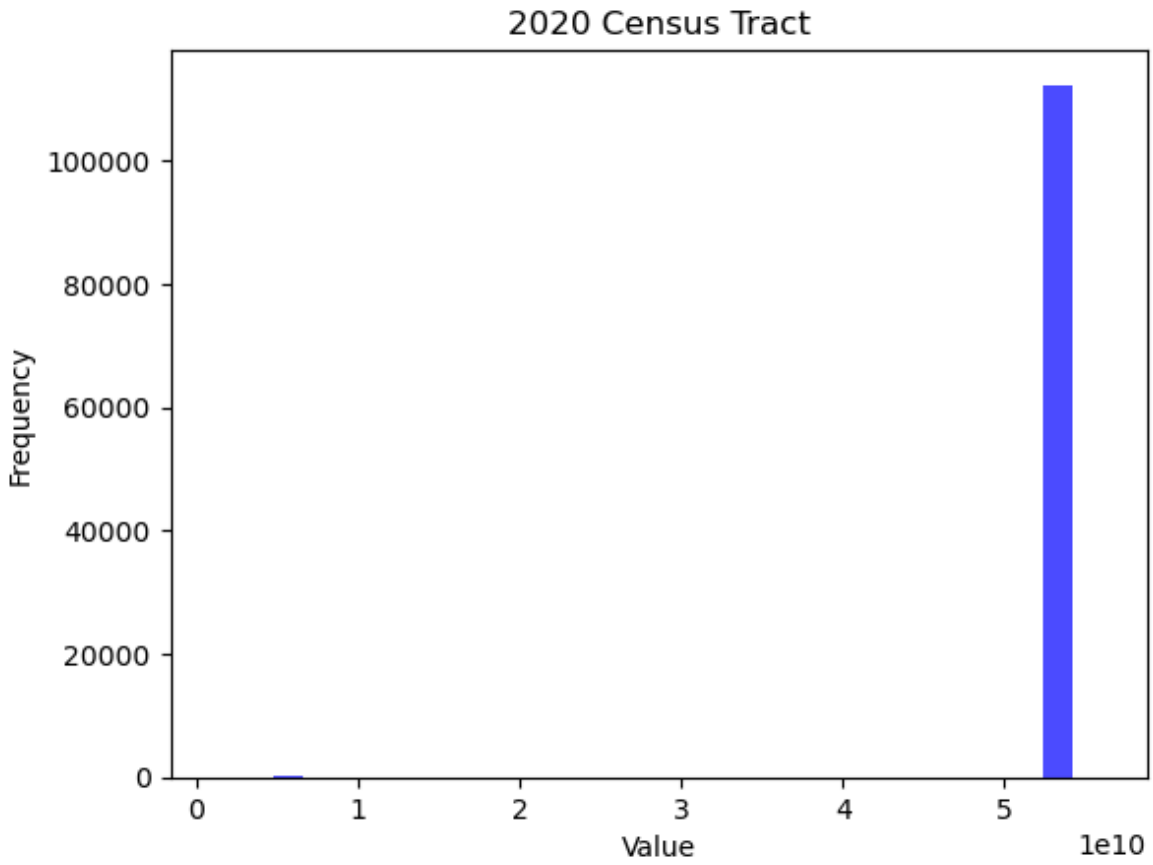

Model Year
Frequency
25000
20000
15000
10000
5000
0
2000
2005
2010
2015
2020
Value


Electric Range
Frequency
Value
0
5000
10000
15000
20000
25000
30000
35000
40000
0
50
100
150
200
250
300
350


Base MSRP
Frequency
Value
0
20000
40000
60000
80000
100000
0
200000
400000
600000
800000


Legislative District
Frequency
Value
0
2000
4000
6000
8000
0
10
20
30
40
50

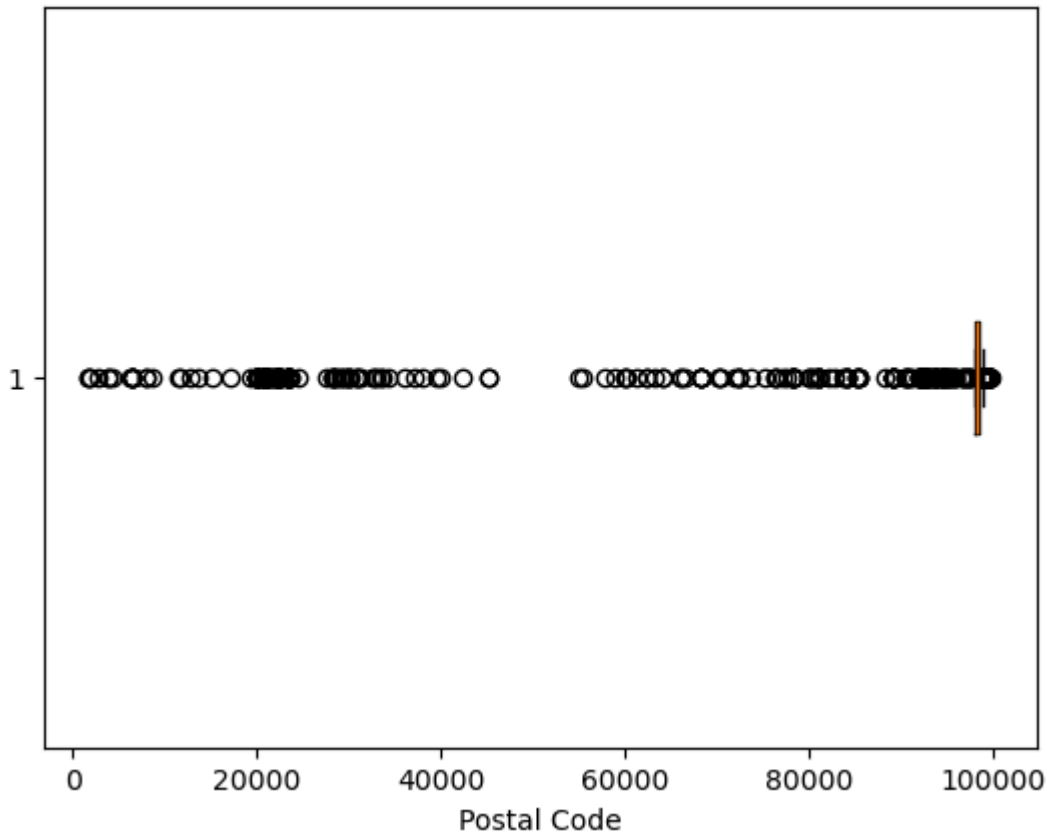

DOL Vehicle ID
Frequency
Value
1e8
0
2000
4000
6000
8000
10000
12000
14000
16000
0
1
2
3
4
5

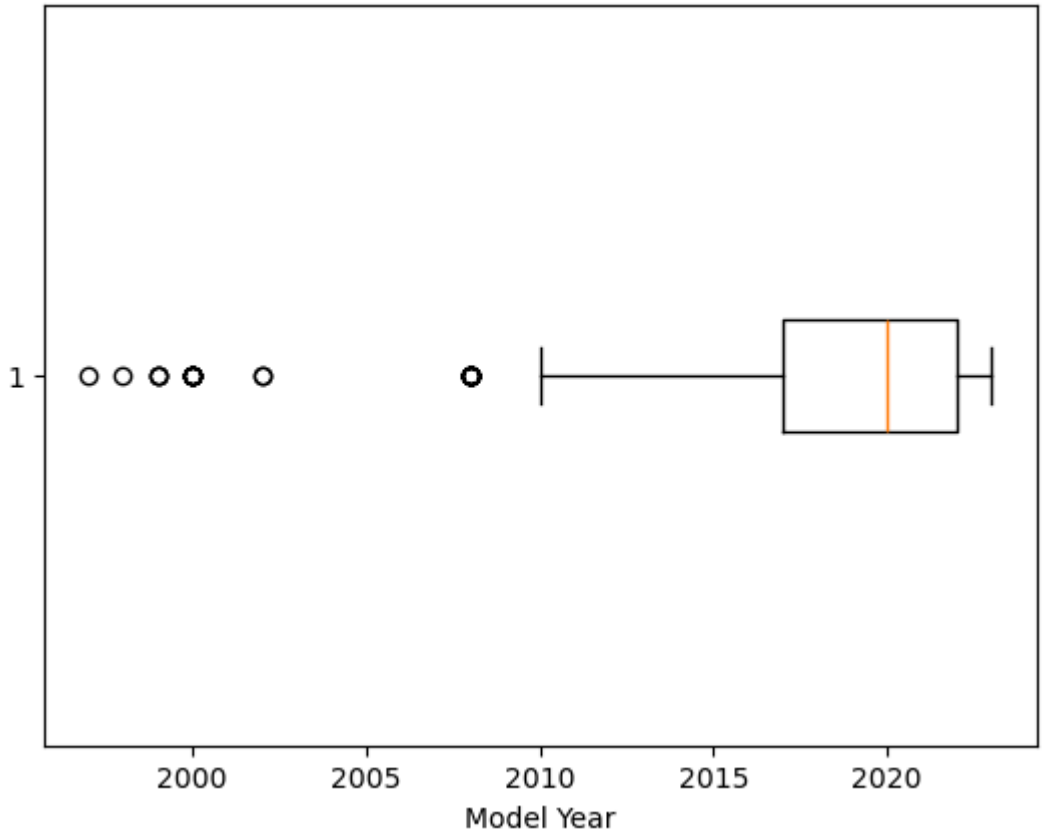


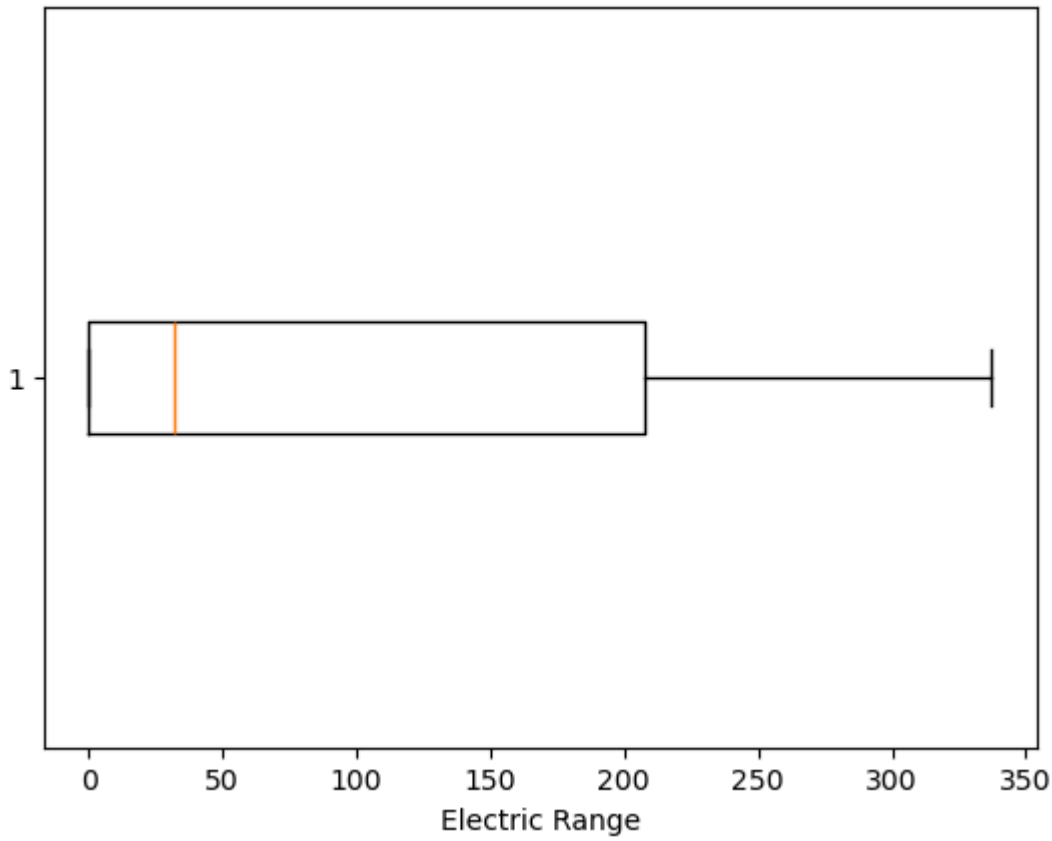
Outliers are prsent in each numerical columns

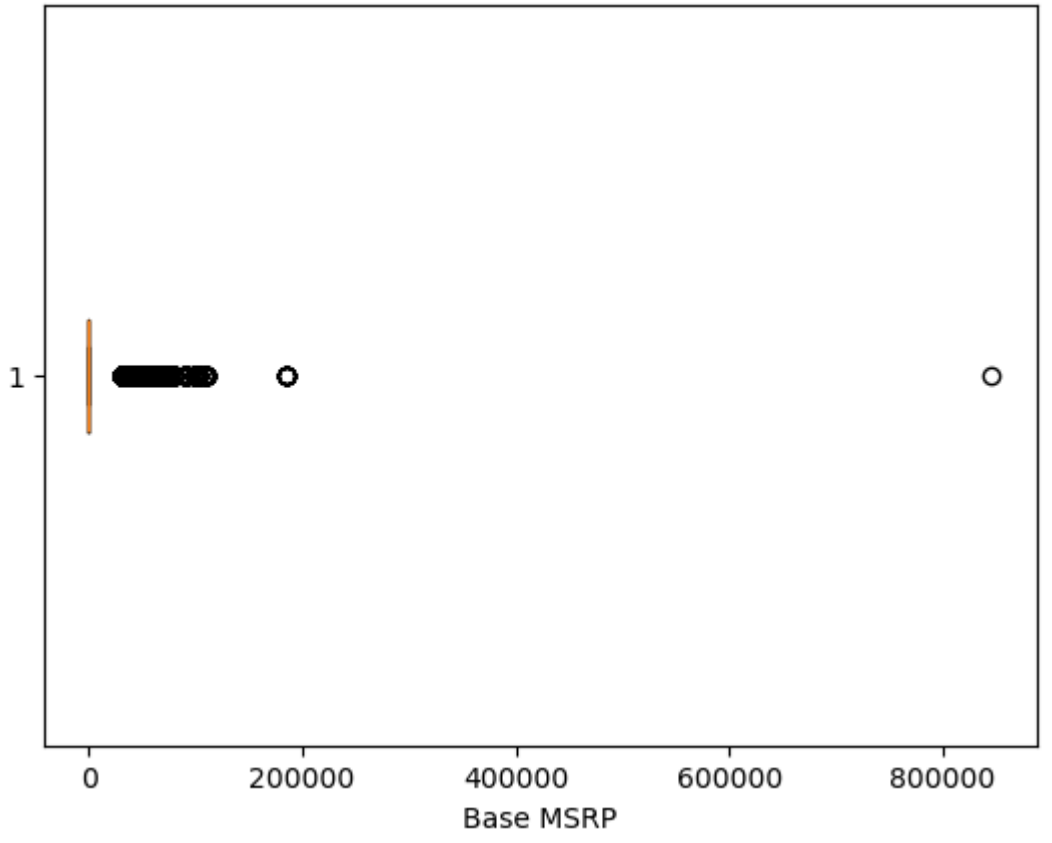
Box plot dectation for outliers

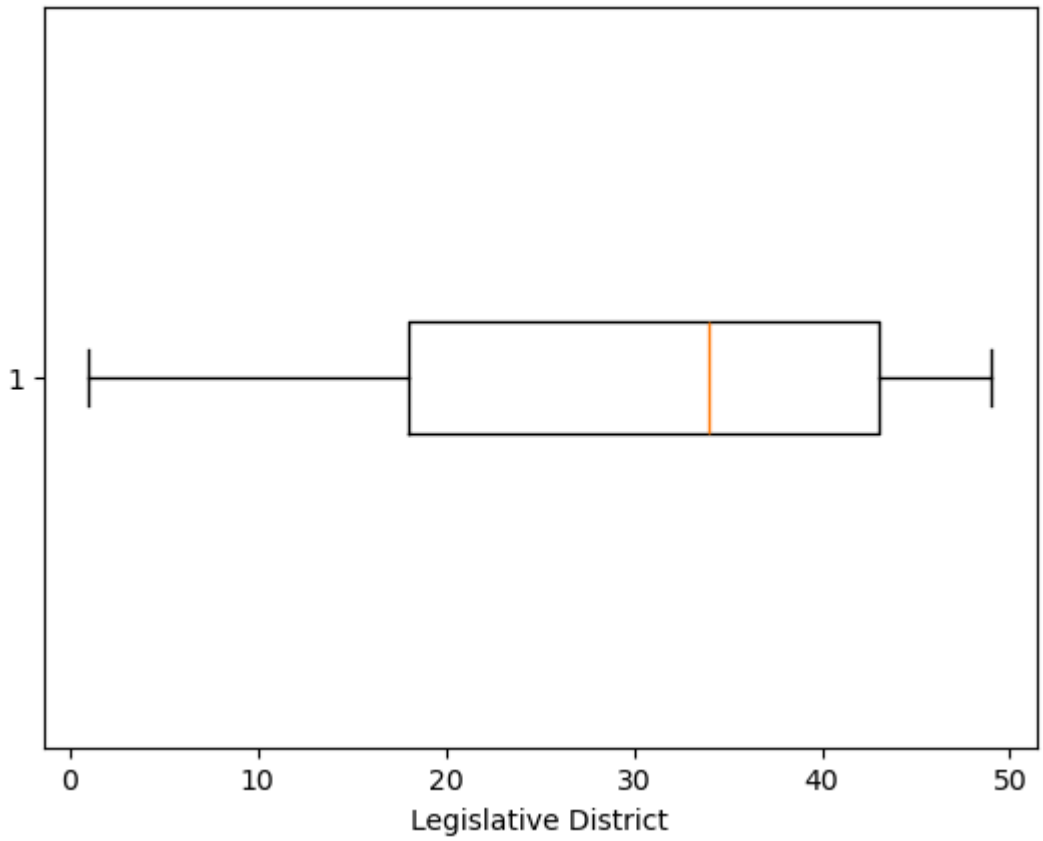
In [31]:
```
for i in num_cols:
    year_data=df[i]
    plt.boxplot(year_data,vert=False)
    plt.xlabel(i)
    plt.show()
```

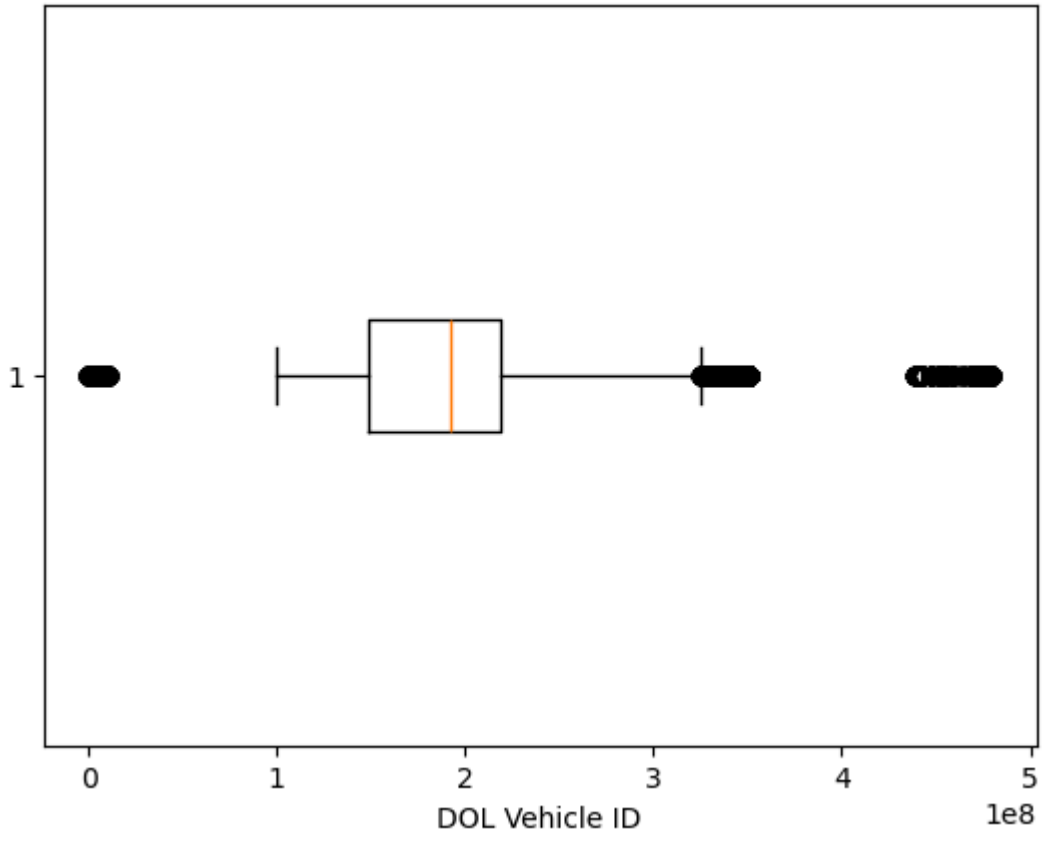

1
0
20000
40000
60000
80000
100000
Postal Code

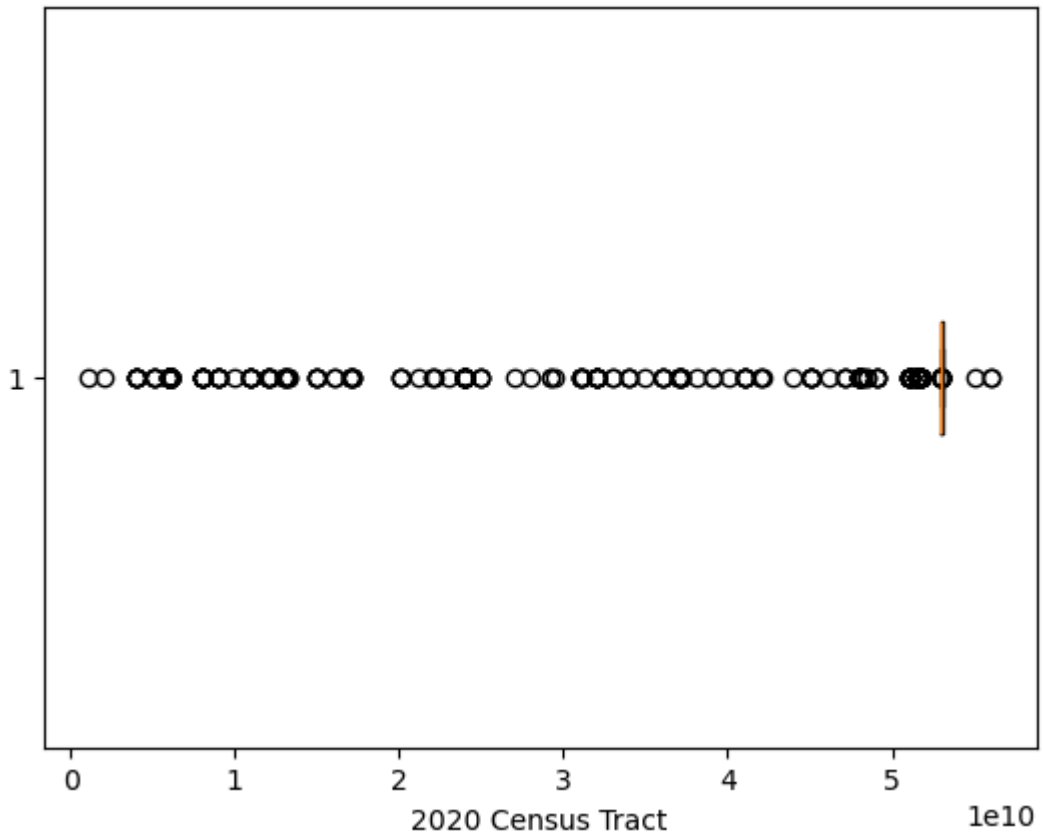

1
2000
2005
2010
2015
2020
Model Year


1
0
50
100
150
200
250
300
350
Electric Range

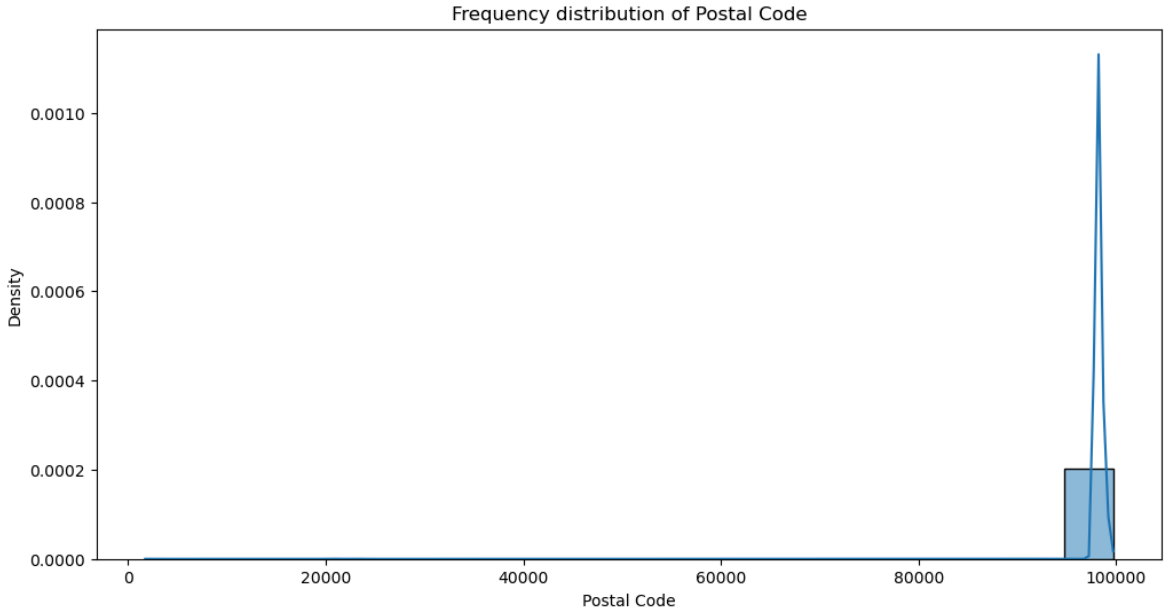

1
0
200000
400000
600000
800000
Base MSRP


1
0
10
20
30
40
50
Legislative District

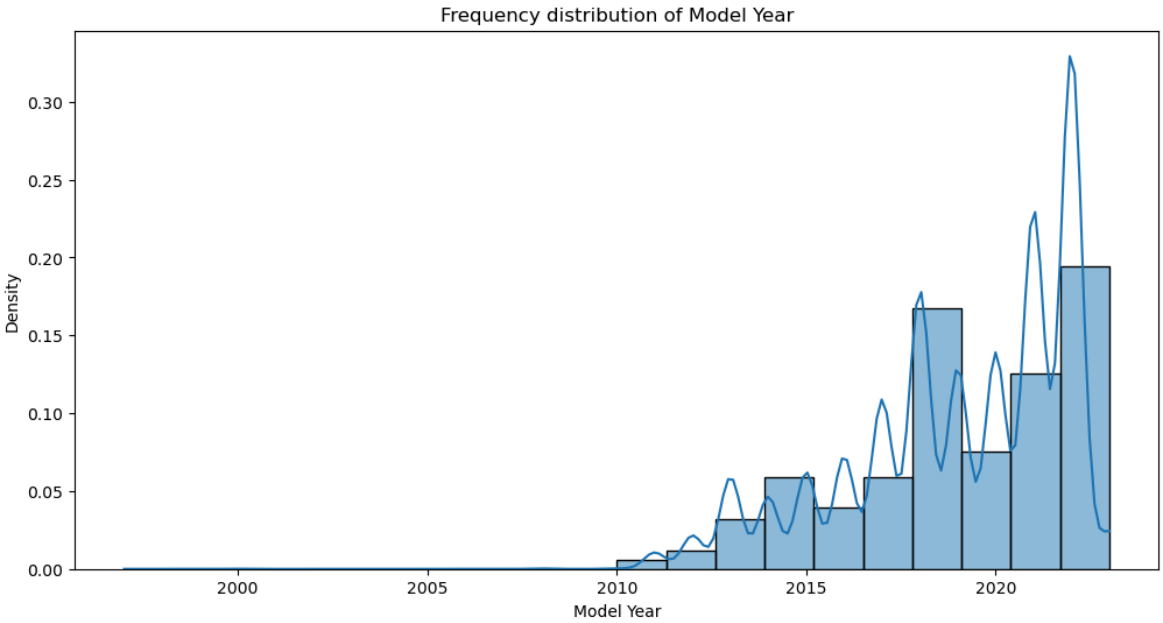

1
0
1
2
3
4
5
DOL Vehicle ID
1e8



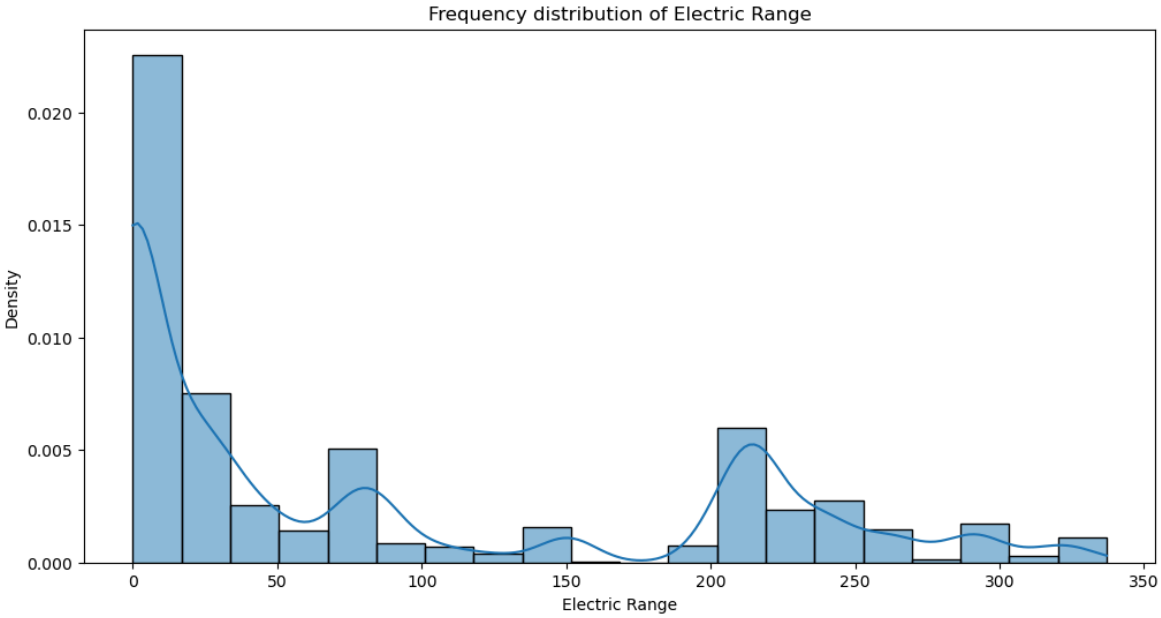
In [32]:
```
import warnings
warnings.filterwarnings('ignore')
for i in num_cols:
    plt.figure(figsize=(12,6))
    sns.histplot(df[i],bins=20,kde=True,stat='density')
    plt.title(f"Frequency distribution of {i}")
    plt.xlabel(i)
    plt.ylabel('Density')
    plt.show()

    prob_dist=df[i].value_counts(normalize=True)
    print(f"Probablity distribution of {i}: \n{prob_dist}\n")
```



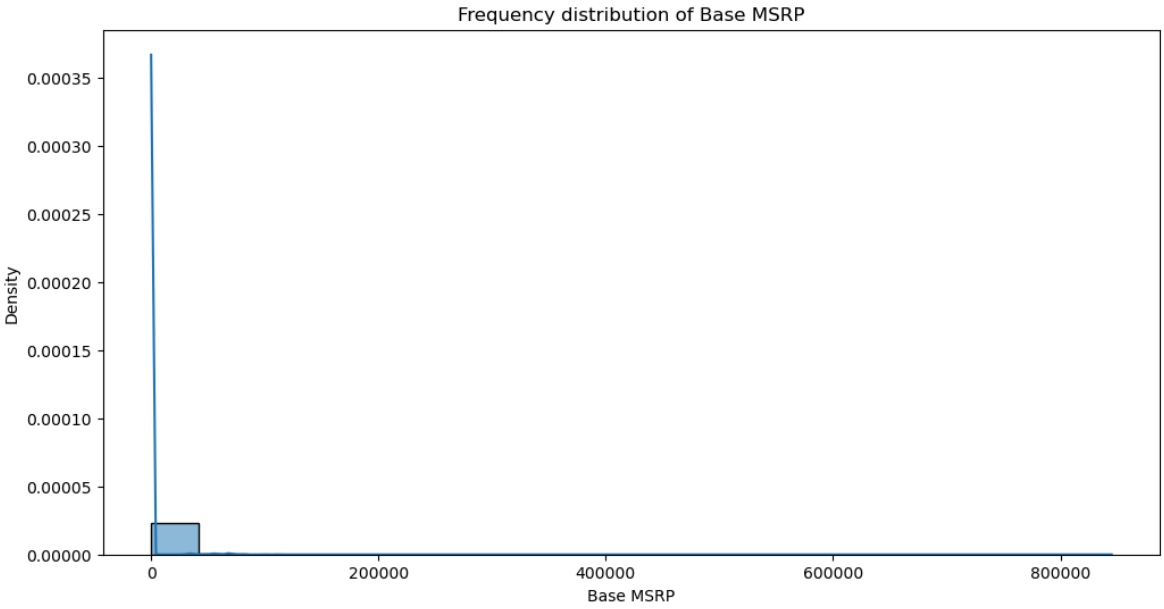
Probablity distribution of Postal Code:
Postal Code
98052   0.025889
98033   0.018280
98004   0.017766
98115   0.016691
98006   0.016443
          ...
21701   0.000009
98621   0.000009
84128   0.000009
92051   0.000009
83876   0.000009
Name: proportion, Length: 773, dtype: float64



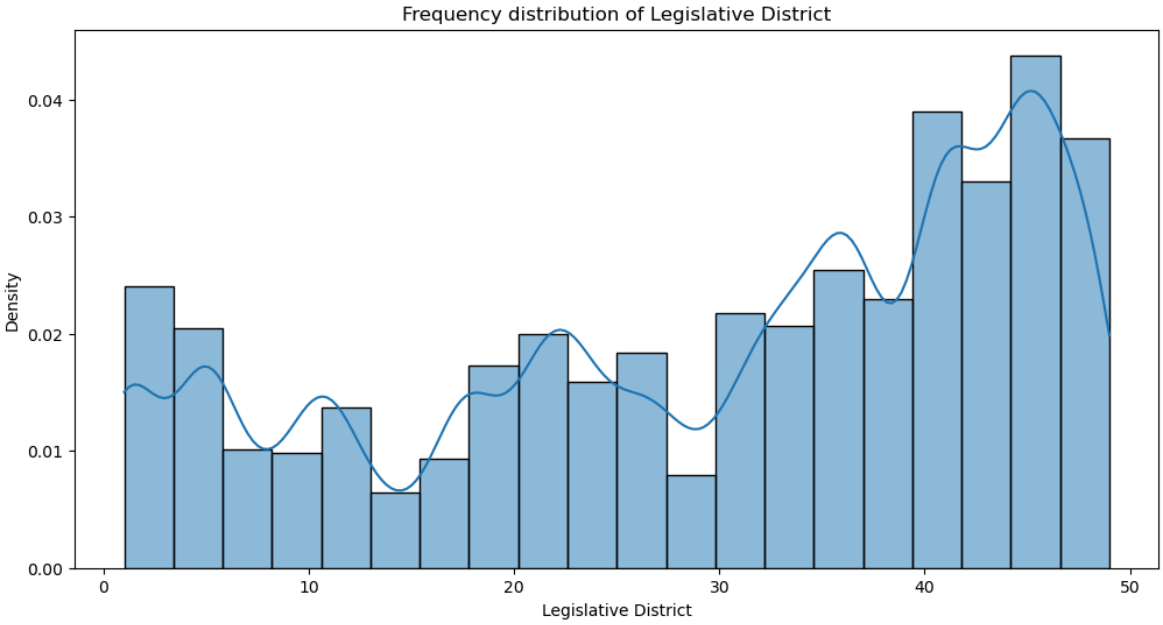
Probablity distribution of Model Year:
Model Year
2022   0.235542
2021   0.163041
2018   0.126480
2020   0.097999
2019   0.091145
2017   0.076744
2016   0.050917
2015   0.043859
2013   0.041648
2014   0.032717
2023   0.016744
2012   0.015138
2011   0.007458
2010   0.000213
2008   0.000204
2000   0.000089
1999   0.000027
2002   0.000018
1997   0.000009
1998   0.000009
Name: proportion, dtype: float64



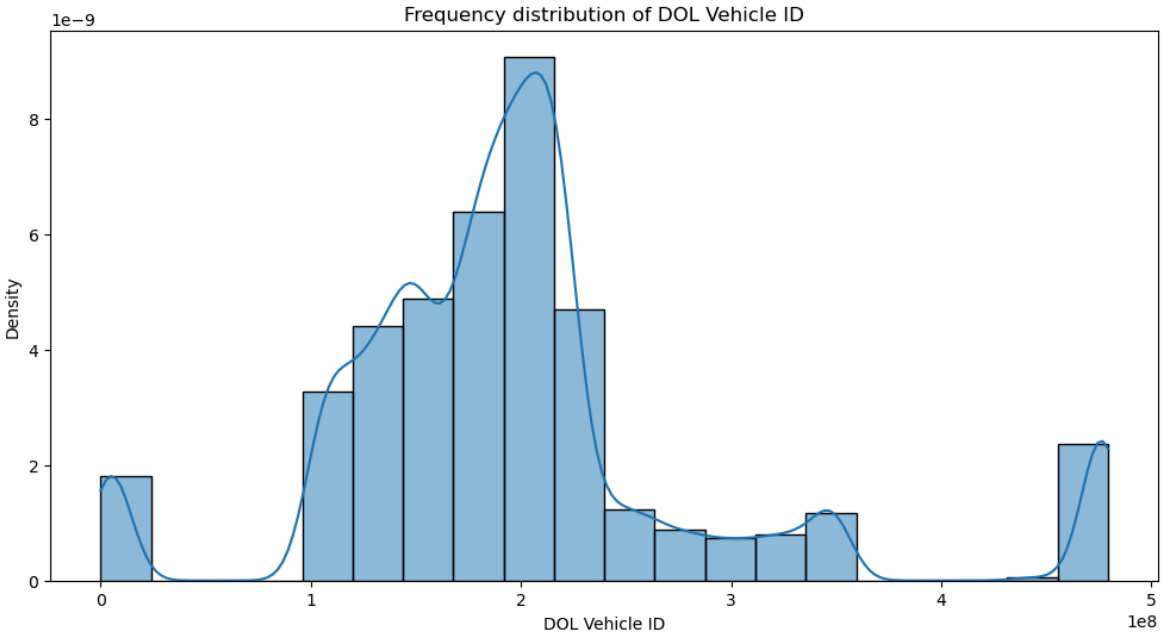

```
Probablity distribution of Electric Range:
Electric Range
0      0.348350
215    0.055987
84     0.036561
220    0.035797
238    0.030834
         ...
11     0.000027
95     0.000018
57     0.000009
39     0.000009
59     0.000009
Name: proportion, Length: 101, dtype: float64
```



```
Probablity distribution of Base MSRP:
Base MSRP
0         0.968819
69900     0.013291
31950     0.003613
52900     0.001900
32250     0.001421
54950     0.001199
59900     0.001190
39995     0.001057
36900     0.000888
44100     0.000861
64950     0.000737
33950     0.000693
45600     0.000675
52650     0.000595
34995     0.000515
36800     0.000444
55700     0.000417
53400     0.000249
110950    0.000213
98950     0.000204
102000    0.000178
90700     0.000169
81100     0.000169
75095     0.000142
184400    0.000107
43700     0.000089
109000    0.000062
89100     0.000062
91250     0.000036
845000    0.000009
Name: proportion, dtype: float64
```

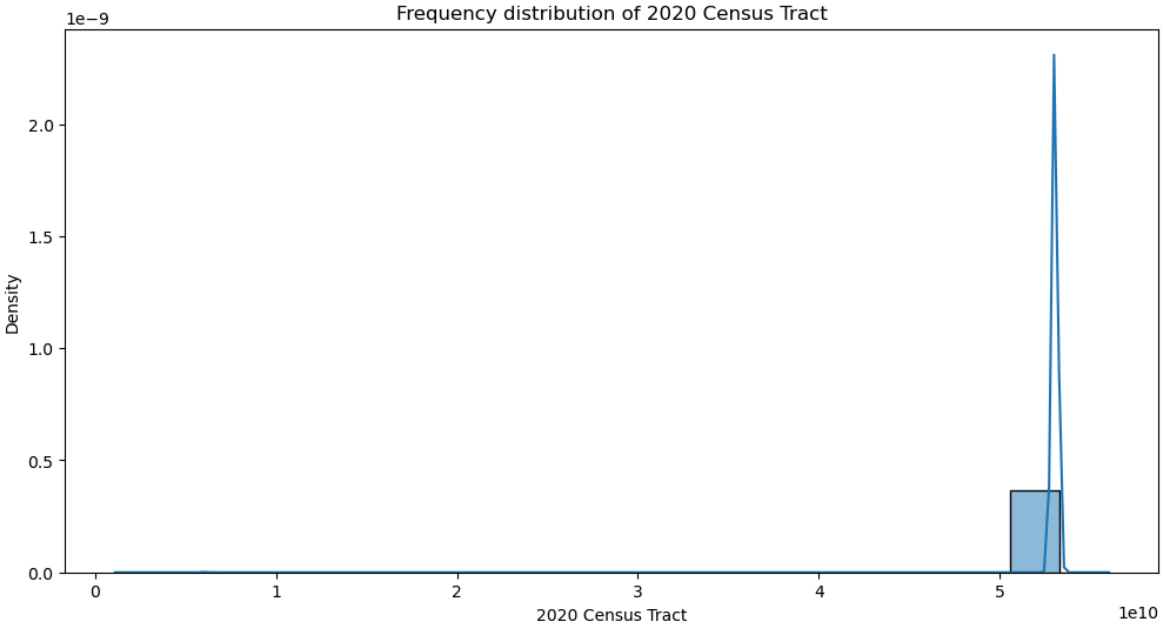


Probablity distribution of Legislative District:
Legislative District
41.0   0.070059
45.0   0.063143
48.0   0.057372
36.0   0.046620
46.0   0.041932
1.0   0.041861
5.0   0.041675
43.0   0.041027
37.0   0.031571
34.0   0.030879
18.0   0.026848
22.0   0.024699
32.0   0.024051
11.0   0.024034
44.0   0.023705
40.0   0.023377
23.0   0.023314
21.0   0.023235
26.0   0.020127
33.0   0.018751
10.0   0.018298
31.0   0.016975
17.0   0.016931
47.0   0.016656
24.0   0.014774
27.0   0.014685
42.0   0.014436
35.0   0.014383
39.0   0.013974
49.0   0.013966
28.0   0.012856
30.0   0.011258
2.0   0.010885
8.0   0.010272
38.0   0.009580
25.0   0.009313
6.0   0.009242
12.0   0.008914
20.0   0.008639
4.0   0.007502
13.0   0.006641
14.0   0.006392
29.0   0.006144
19.0   0.005966
16.0   0.005425
9.0   0.005380
3.0   0.004945
7.0   0.004830
15.0   0.002459
Name: proportion, dtype: float64



Probablity distribution of DOL Vehicle ID:

```
DOL Vehicle ID
198968248    0.000009
180703806    0.000009
475706185    0.000009
219886068    0.000009
303587483    0.000009
               ...
215103542    0.000009
124720459    0.000009
308978253    0.000009
151867096    0.000009
194673692    0.000009
Name: proportion, Length: 112634, dtype: float64
```



```
Probablity distribution of 2020 Census Tract:
2020 Census Tract
53033028500    0.005176
53033032321    0.004883
53033007800    0.003720
53033024100    0.003560
53033005600    0.003498
                 ...
6111005503     0.000009
17161023200    0.000009
53077001300    0.000009
24021750805    0.000009
16055940000    0.000009
Name: proportion, Length: 2026, dtype: float64
```

Frquency distribution of eacj categorical columns

In [33]:
```python
frequency_dis={i: df[i].value_counts() for i in cat_cols}

for i,freq in frequency_dis.items():
    print(f"Frequncy distribution for {i}:\n{freq}\n")
```

```
Frequncy distribution for VIN (1-10):
VIN (1-10)
5YJYGDEE9M    472
5YJYGDEE0M    465
5YJYGDEE8M    448
5YJYGDEE7M    448
5YJYGDEE2M    437
              ...
WA1LAAGE9M      1
5UXKT0C50H      1
5YJYGAED3M      1
WDC0G5DBXL      1
YV4ED3GM0P      1
Name: count, Length: 7548, dtype: int64

Frequncy distribution for County:
County
King          59000
Snohomish     12434
Pierce         8535
Clark          6689
Thurston       4126
               ...
Pinal             1
Elmore            1
Portsmouth        1
Kings             1
Kootenai          1
Name: count, Length: 165, dtype: int64

Frequncy distribution for City:
City
Seattle         20305
Bellevue         5921
Redmond          4201
Vancouver        4013
Kirkland         3598
                 ...
Hartline            1
Gaithersburg        1
El Paso             1
Klickitat           1
Worley              1
Name: count, Length: 629, dtype: int64

Frequncy distribution for State:
State
WA    112348
CA        76
VA        36
MD        26
TX        14
CO         9
NV         8
GA         7
NC         7
CT         6
DC         6
FL         6
AZ         6
```

```
IL         6
SC         5
OR         5
NE         5
HI         4
UT         4
AR         4
NY         4
TN         3
KS         3
MO         3
PA         3
MA         3
LA         3
NJ         3
NH         2
OH         2
WY         2
ID         2
KY         1
RI         1
ME         1
MN         1
SD         1
WI         1
NM         1
AK         1
MS         1
AL         1
DE         1
OK         1
ND         1
Name: count, dtype: int64

Frequncy distribution for Make:
Make
TESLA             52078
NISSAN            12880
CHEVROLET         10182
FORD               5819
BMW                4680
KIA                4483
TOYOTA             4405
VOLKSWAGEN         2514
AUDI               2332
VOLVO              2288
CHRYSLER           1794
HYUNDAI            1412
JEEP               1152
RIVIAN              885
FIAT                822
PORSCHE             818
HONDA               792
MINI                632
MITSUBISHI          588
POLESTAR            558
MERCEDES-BENZ       506
SMART               273
JAGUAR              219
LINCOLN             168
```

```
CADILLAC            108
LUCID MOTORS         65
SUBARU               59
LAND ROVER           38
LEXUS                33
FISKER               20
GENESIS              18
AZURE DYNAMICS        7
TH!NK                 3
BENTLEY               3
Name: count, dtype: int64

Frequncy distribution for Model:
Model
MODEL 3        23155
MODEL Y        17142
LEAF           12880
MODEL S         7377
BOLT EV         4910
               ...
745LE              2
S-10 PICKUP        1
SOLTERRA           1
918                1
FLYING SPUR        1
Name: count, Length: 114, dtype: int64

Frequncy distribution for Electric Vehicle Type:
Electric Vehicle Type
Battery Electric Vehicle (BEV)            86044
Plug-in Hybrid Electric Vehicle (PHEV)    26590
Name: count, dtype: int64

Frequncy distribution for Clean Alternative Fuel Vehicle (CAFV) Eligibility:
Clean Alternative Fuel Vehicle (CAFV) Eligibility
Clean Alternative Fuel Vehicle Eligible                          58639
Eligibility unknown as battery range has not been researched    39236
Not eligible due to low battery range                            14759
Name: count, dtype: int64

Frequncy distribution for Vehicle Location:
Vehicle Location
POINT (-122.13158 47.67858)    2940
POINT (-122.2066 47.67887)     2059
POINT (-122.1872 47.61001)     2001
POINT (-122.31765 47.70013)    1880
POINT (-122.12096 47.55584)    1852
                               ...
POINT (-124.33152 48.05431)       1
POINT (-77.41203 39.41574)        1
POINT (-123.61022 46.35588)       1
POINT (-112.04165 40.68741)       1
POINT (-116.91895 47.40077)       1
Name: count, Length: 758, dtype: int64

Frequncy distribution for Electric Utility:
Electric Utility
PUGET SOUND ENERGY INC||CITY OF TACOMA - (WA)
40690
PUGET SOUND ENERGY INC
```

```
22172
CITY OF SEATTLE - (WA)|CITY OF TACOMA - (WA)
21447
BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF CLARK COUNTY - (WA)
6522
BONNEVILLE POWER ADMINISTRATION||CITY OF TACOMA - (WA)||PENINSULA LIGHT COMPANY
5053

...
BONNEVILLE POWER ADMINISTRATION||PENINSULA LIGHT COMPANY
1
BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF ASOTIN COUNTY
1
CITY OF SEATTLE - (WA)
1
BONNEVILLE POWER ADMINISTRATION||NESPELEM VALLEY ELEC COOP, INC
1
BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF CLALLAM COUNTY|PUD NO 1 OF JEFFERSON
COUNTY        1
Name: count, Length: 73, dtype: int64
```

Removal of outliers by using IQR(Inter Quartile Range) Technique

In [34]:
```
#by using IQR we are able to remove outliers in data

data=df['Model Year']
q1=np.percentile(data,25)
q2=np.percentile(data,50)
q3=np.percentile(data,75)

IQR=q3-q1

lb=(q1-(1.5*IQR))
ub=(q3+(1.5*IQR))

con1=data>lb
con2=data<ub
con3=con1&con2
non_outliers_data=data[con3]
non_outliers_data

data=df['Base MSRP']
q1=np.percentile(data,25)
q2=np.percentile(data,50)
q3=np.percentile(data,75)

IQR=q3-q1

lb=(q1-(1.5*IQR))
ub=(q3+(1.5*IQR))

con1=data>lb
con2=data<ub
con3=con1&con2
non_outliers_data=data[con3]
non_outliers_data
```

Out[34]:
```
Series([], Name: Base MSRP, dtype: int64)
```

In [35]:

```
#by using IQR we are able to remove outliers in data
data=df['Postal Code']
q1=np.percentile(data,25)
q2=np.percentile(data,50)
q3=np.percentile(data,75)

IQR=q3-q1

lb=(q1-(1.5*IQR))
ub=(q3+(1.5*IQR))

con1=data>lb
con2=data<ub
con3=con1&con2
non_outliers_data=data[con3]
non_outliers_data

#by using IQR we are able to remove outliers in data
data=df['DOL Vehicle ID']
q1=np.percentile(data,25)
q2=np.percentile(data,50)
q3=np.percentile(data,75)

IQR=q3-q1

lb=(q1-(1.5*IQR))
ub=(q3+(1.5*IQR))

con1=data>lb
con2=data<ub
con3=con1&con2
non_outliers_data=data[con3]
non_outliers_data

#by using IQR we are able to remove outliers in data
data=df['2020 Census Tract']
q1=np.percentile(data,25)
q2=np.percentile(data,50)
q3=np.percentile(data,75)

IQR=q3-q1

lb=(q1-(1.5*IQR))
ub=(q3+(1.5*IQR))

con1=data>lb
con2=data<ub
con3=con1&con2
non_outliers_data=data[con3]
non_outliers_data
```

Out[35]:
```
2         53077001602
3         53057951101
4         53061041500
5         53061051916
6         53061040900
             ...
112629    53033032401
112630    53055960301
112631    53033027702
112632    53033032007
112633    53033032005
Name: 2020 Census Tract, Length: 112314, dtype: int64
```

In [36]:
```
import warnings
warnings.filterwarnings('ignore')
non_outliers_df=df[con3]
non_outliers_df.dropna(inplace=True)
non_outliers_df
```

Out[36]:

	VIN (1-10)	County	City	State	Postal Code	Model Year	Make	Mode
2	JN1AZ0CP8B	Yakima	Yakima	WA	98901	2011	NISSAN	LEA
3	1G1FW6S08H	Skagit	Concrete	WA	98237	2017	CHEVROLET	BOL EV
4	3FA6P0SU1K	Snohomish	Everett	WA	98201	2019	FORD	FUSION
5	5YJ3E1EB5J	Snohomish	Bothell	WA	98021	2018	TESLA	MODE 3
6	1N4AZ0CP4D	Snohomish	Everett	WA	98203	2013	NISSAN	LEA
...	...	...	...	...	...	...	...	.
112629	7SAYGDEF2N	King	Duvall	WA	98019	2022	TESLA	MODE Y
112630	1N4BZ1CP7K	San Juan	Friday Harbor	WA	98250	2019	NISSAN	LEA
112631	1FMCU0KZ4N	King	Vashon	WA	98070	2022	FORD	ESCAPE

	VIN (1-10)	County	City	State	Postal Code	Model Year	Make	Mode
112632	KNDCD3LD4J	King	Covington	WA	98042	2018	KIA	NIRO
112633	YV4BR0CL8N	King	Covington	WA	98042	2022	VOLVO	XC90

112314 rows × 17 columns

Extracted dataframe without any outliers

Difference of outliers and non_out_liers dataframe

In [37]:
```
out=len(df)-len(non_outliers_df)
print(len(df),"Which contains outlier data")
print(len(non_outliers_df), "Length which doesnot contains any outliers")
print(f"The number of outliers present in data are : {out}")
```

```
112634 Which contains outlier data
112314 Length which doesnot contains any outliers
The number of outliers present in data are : 320
```

Task 2: Create a Choropleth using plotly.express to display the number of EV vehicles based on location.

Step 1 - Installing plotly module. You can do it inside Jupyter Notebook as shown below

In [38]:
```
pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\sruth\anaconda3\lib\site-packag
es (5.9.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\sruth\anaconda3\lib\si
te-packages (from plotly) (8.2.2)
Note: you may need to restart the kernel to use updated packages.
```

Step 2 - Reading the csv data into a dataframe.

In [39]:
```
df
```

Out[39]:

	VIN (1-10)	County	City	State	Postal Code	Model Year	Make	Mode
0	JTMEB3FV6N	Monroe	Key West	FL	33040	2022	TOYOTA	RAV4 PRIM
1	1G1RD6E45D	Clark	Laughlin	NV	89029	2013	CHEVROLET	VOL
2	JN1AZ0CP8B	Yakima	Yakima	WA	98901	2011	NISSAN	LEA
3	1G1FW6S08H	Skagit	Concrete	WA	98237	2017	CHEVROLET	BOL EV
4	3FA6P0SU1K	Snohomish	Everett	WA	98201	2019	FORD	FUSION
...	...	...	...	...	...	...	...	.
112629	7SAYGDEF2N	King	Duvall	WA	98019	2022	TESLA	MODE Y
112630	1N4BZ1CP7K	San Juan	Friday Harbor	WA	98250	2019	NISSAN	LEA
112631	1FMCU0KZ4N	King	Vashon	WA	98070	2022	FORD	ESCAPE

	VIN (1-10)	County	City	State	Postal Code	Model Year	Make	Mode
112632	KNDCD3LD4J	King	Covington	WA	98042	2018	KIA	NIRC
112633	YV4BR0CL8N	King	Covington	WA	98042	2022	VOLVO	XC9(

112634 rows × 17 columns

Step 3 - Import required library - plotly.express

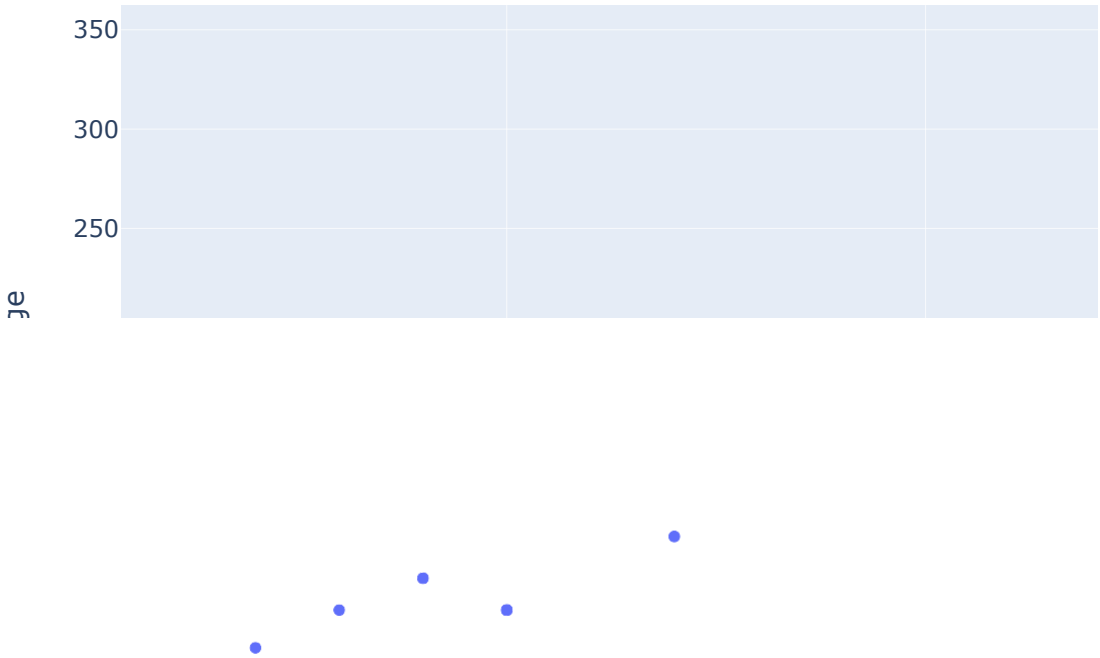
In [40]:
```
import plotly.express as px
```

Step 4 - Scatter Plot using plotly.express

Note - Scatter Plot is a bivariate plot.

Bivariate means it requires two columns. You should make a note that both the variables should be real numerical valued.

In [41]:
```
# scatter plot

px.scatter(df,x ='Model Year',y ='Electric Range')
```



Step 5 - Box Plot using plotly.express

Note - Box Plot can be used to create a univariate or bivariate plot.

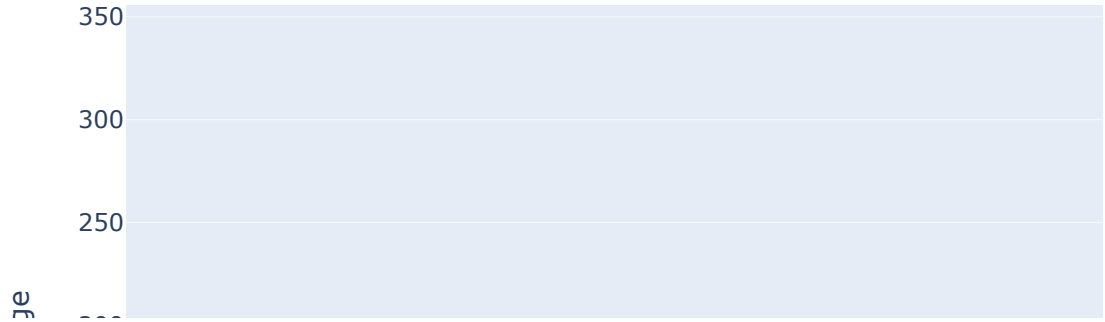
For a univariate box plot, the column type should be real numerical.

For a bivariate box plot, one column should be categorical and another column should be real numerical.

Below is an example of code for bivariate box plot.

Bivariate box plot

In [42]:
```
# Box plot in plotly

px.box(df,x='Electric Vehicle Type',y='Electric Range')
```



Step 6 - Pie Chart Plot using plotly.express

Note - Pie Chart Plot can be used to create a bivariate plot.

For a bivariate pie chart plot, one column should be categorical and another column should be real numerical.

Below is an example of code for the plot.

names: It should be categorical column

values: It should be numeric column

In [43]:
```
px.pie(df,names='Model',values='Model Year')
```

Step 7 - Choropleth Plot using plotly.express

locations: It can be columns like - 'Country', 'Zip Code', etc..

color: It can be a column, value of which is used to assign color to marks

locationmode: It should be either one of 'ISO-3', 'USA-states', or 'country names'.

In [44]:
```
fig=px.choropleth(df,
                  locations='County',
                  color='Postal Code',
                  hover_name='Postal Code',
                  locationmode='country names'
                 )
fig.show()
```

Step 8 - Animated Choropleth Plot using plotly.express

Note - Parameters for choropleth plot:

animation_frame: It should be a column like day, year, month, etc on which animation will be applied.

In [45]:
```
fig=px.choropleth(df,
                  locations='County',
                  color='County',
                  hover_name='Postal Code',
                  locationmode='country names',
                  animation_frame='Postal Code')
fig.show()
```

Task 3: Create a Racing Bar Plot to display the animation of EV Make and its count each year.

Step 1 - Installing bar-chart-race module

In [46]:
```
!pip install bar-chart-race
```

```
Requirement already satisfied: bar-chart-race in c:\users\sruth\anaconda3\lib\sit
e-packages (0.1.0)
Requirement already satisfied: pandas>=0.24 in c:\users\sruth\anaconda3\lib\site-
packages (from bar-chart-race) (2.1.4)
Requirement already satisfied: matplotlib>=3.1 in c:\users\sruth\anaconda3\lib\si
te-packages (from bar-chart-race) (3.8.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\sruth\anaconda3\lib\s
ite-packages (from matplotlib>=3.1->bar-chart-race) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\sruth\anaconda3\lib\site-
packages (from matplotlib>=3.1->bar-chart-race) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\sruth\anaconda3\lib
\site-packages (from matplotlib>=3.1->bar-chart-race) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\sruth\anaconda3\lib
\site-packages (from matplotlib>=3.1->bar-chart-race) (1.4.4)
Requirement already satisfied: numpy<2,>=1.21 in c:\users\sruth\anaconda3\lib\sit
e-packages (from matplotlib>=3.1->bar-chart-race) (1.26.4)
Requirement already satisfied: packaging>=20.0 in c:\users\sruth\anaconda3\lib\si
te-packages (from matplotlib>=3.1->bar-chart-race) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\sruth\anaconda3\lib\site
-packages (from matplotlib>=3.1->bar-chart-race) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\sruth\anaconda3\lib\s
ite-packages (from matplotlib>=3.1->bar-chart-race) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\sruth\anaconda3\l
ib\site-packages (from matplotlib>=3.1->bar-chart-race) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\sruth\anaconda3\lib\site-
packages (from pandas>=0.24->bar-chart-race) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\sruth\anaconda3\lib\sit
e-packages (from pandas>=0.24->bar-chart-race) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\sruth\anaconda3\lib\site-pack
ages (from python-dateutil>=2.7->matplotlib>=3.1->bar-chart-race) (1.16.0)
```

Step 2 - Import required library - bar_chart_race

In [47]:
```
import bar_chart_race as bcr
```

Step 3 - Call bar_chart_race function with following parameters.

In [48]:
```
pip install pandas  matplotlib pillow
```

```
Requirement already satisfied: pandas in c:\users\sruth\anaconda3\lib\site-packag
es (2.1.4)
Requirement already satisfied: matplotlib in c:\users\sruth\anaconda3\lib\site-pa
ckages (3.8.0)
Requirement already satisfied: pillow in c:\users\sruth\anaconda3\lib\site-packag
es (10.2.0)
Requirement already satisfied: numpy<2,>=1.23.2 in c:\users\sruth\anaconda3\lib\s
ite-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\sruth\anaconda3
\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\sruth\anaconda3\lib\site-
packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\sruth\anaconda3\lib\sit
e-packages (from pandas) (2023.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\sruth\anaconda3\lib\s
ite-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\sruth\anaconda3\lib\site-
packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\sruth\anaconda3\lib
\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\sruth\anaconda3\lib
\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\sruth\anaconda3\lib\si
te-packages (from matplotlib) (23.1)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\sruth\anaconda3\lib\s
ite-packages (from matplotlib) (3.0.9)
Requirement already satisfied: six>=1.5 in c:\users\sruth\anaconda3\lib\site-pack
ages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [51]:

```
try:

    import pandas as pd
    import bar_chart_race as bcr

    sales_data=df.groupby(['Model Year','Make']).size().unstack(fill_value=0)
    sales_data=sales_data.T
    bcr.bar_chart_race(
        df=sales_data,

        title='Year-wise Electric Vehicle Sales by Maker',
        n_bars=10,
        sort='desc',
        fixed_order=False,
        bar_label_size=7,
        bar_size=0.5,
        title_size=20,
        figsize=(8,5),
        dpi=300,
     # adjust the position and style of the period label
        period_label={'x': .95, 'y': .15,
                      'ha': 'right',
                      'va': 'center',
                      'size': 72,
                      'weight': 'semibold'
                      },

    )
    years=sales_data.columns
    ani=animation.FuncAnimation(figsize,frames=years,repeat=True,interval=500)
```

```
    ani.save('racing_bar_plot.gif',writer='pillow',fps=2)
    print("Racing bar plot craeted and saved as 'Electric_vehicle_sales_racing.m

except Exception as e:
            print(e)
```

```
You do not have ffmpeg installed on your machine. Download
                            ffmpeg from here: https://www.ffmpeg.org/download.htm
l.

                            Matplotlib's original error message below:

                            Requested MovieWriter (ffmpeg) not available
```

In [50]:
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from matplotlib.ticker import MaxNLocator

sales_data=df.groupby(['Model Year','Make']).size().unstack(fill_value=0)
sales_data=sales_data.T

sales_data
```

Out[50]:

Model Year Make	1997	1998	1999	2000	2002	2008	2010	2011	2012	2013	2014
AUDI	0	0	0	0	0	0	0	0	0	0	0
AZURE DYNAMICS	0	0	0	0	0	0	0	4	3	0	0
BENTLEY	0	0	0	0	0	0	0	0	0	0	0
BMW	0	0	0	0	0	0	0	0	0	0	457
CADILLAC	0	0	0	0	0	0	0	0	0	0	58
CHEVROLET	1	0	0	0	0	0	0	71	496	818	724
CHRYSLER	0	0	0	0	0	0	0	0	0	0	0
FIAT	0	0	0	0	0	0	0	0	0	106	97
FISKER	0	0	0	0	0	0	0	0	20	0	0
FORD	0	1	3	10	0	0	0	0	15	662	628
GENESIS	0	0	0	0	0	0	0	0	0	0	0
HONDA	0	0	0	0	0	0	0	0	0	0	9
HYUNDAI	0	0	0	0	0	0	0	0	0	0	0
JAGUAR	0	0	0	0	0	0	0	0	0	0	0
JEEP	0	0	0	0	0	0	0	0	0	0	0
KIA	0	0	0	0	0	0	0	0	0	0	0
LAND ROVER	0	0	0	0	0	0	0	0	0	0	0
LEXUS	0	0	0	0	0	0	0	0	0	0	0
LINCOLN	0	0	0	0	0	0	0	0	0	0	0
LUCID MOTORS	0	0	0	0	0	0	0	0	0	0	0
MERCEDES-BENZ	0	0	0	0	0	0	0	0	0	0	31
MINI	0	0	0	0	0	0	0	0	0	0	0
MITSUBISHI	0	0	0	0	0	0	0	0	42	0	10
NISSAN	0	0	0	0	0	0	0	755	610	1966	694
POLESTAR	0	0	0	0	0	0	0	0	0	0	0
PORSCHE	0	0	0	0	0	0	0	0	0	0	8
RIVIAN	0	0	0	0	0	0	0	0	0	0	0
SMART	0	0	0	0	0	0	0	0	0	29	71
SUBARU	0	0	0	0	0	0	0	0	0	0	0
TESLA	0	0	0	0	0	23	24	7	134	814	683

Model Year	1997	1998	1999	2000	2002	2008	2010	2011	2012	2013	2014
Make											
TH!NK	0	0	0	0	0	0	0	3	0	0	0
TOYOTA	0	0	0	0	2	0	0	0	385	296	215
VOLKSWAGEN	0	0	0	0	0	0	0	0	0	0	0
VOLVO	0	0	0	0	0	0	0	0	0	0	0

In [ ]: