In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plot
import seaborn as sns
```

In [2]:

```
data=pd.read_csv('miles-driven.csv')
```

In [3]:

```
data.head()
data.tail()
data.shape
col_name=data.columns
```

In [4]:

```
col_name
```

Out[4]:

```
Index(['state|million_miles_annually'], dtype='object')
```

In [5]:

```
data.info
```

Out[5]:

```
<bound method DataFrame.info of    state|million_miles_annually
0                  Alabama|64914
1                   Alaska|4593
2                  Arizona|59575
3                 Arkansas|32953
4               California|320784
5                 Colorado|46606
6               Connecticut|31197
7                  Delaware|9028
8     District of Columbia|3568
9                  Florida|191855
10                 Georgia|108454
11                  Hawaii|10066
12                   Idaho|15937
13                Illinois|103234
14                  Indiana|76485
15                     Iowa|31274
16                   Kansas|30021
17                  Kentucky|48061
18                Louisiana|46513
19                     Maine|14248
20                  Maryland|56221
21            Massachusetts|54792
22                 Michigan|94754
23                Minnesota|56685
24              Mississippi|38851
25                 Missouri|68789
26                  Montana|11660
27                 Nebraska|19093
28                   Nevada|24189
29            New Hampshire|12720
30               New Jersey|73094
31               New Mexico|25650
32                New York|127726
33          North Carolina|103772
34              North Dakota|9131
35                    Ohio|111990
36                Oklahoma|47464
37                  Oregon|33373
38             Pennsylvania|99204
39              Rhode Island|7901
40           South Carolina|48730
41              South Dakota|9002
42                Tennessee|70751
43                   Texas|237440
44                    Utah|26222
45                 Vermont|7141
46                Virginia|80974
47              Washington|56955
48            West Virginia|18963
49                Wisconsin|58554
50                   Wyoming|9245>
```

In [6]:

```
data.isnull().sum()
```

Out[6]:

```
state|million_miles_annually    0
dtype: int64
```

In [7]:

```
null_col=[feature for feature in col_name if data [feature].isnull().sum()>0]
```

In [8]:

```
len(null_col)
```

Out[8]:

```
0
```

In [9]:

```
data[null_col]
```

Out[9]:

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33

In [10]:

```
len(null_col)
```

Out[10]:

```
0
```

In [11]:

```
for x in null_col:
    null=data[x].isnull().sum()
    pn=round((null/data.shape[0])*100,2)
    print(x,'has',null,'&',pn,'% of null values')
```

In [12]:

```
    drop=[]
    for x in null_col:
        null=data[x].isnull().sum()
        pn=round((null/data.shape[0])*100,2)
        if pn>10:
            drop.append(x)
        print(x,'has',null,'&',pn,'% of null values')
```

In [13]:

```
drop
```

Out[13]:

```
[]
```

In [14]:

```
data_drop=data.drop(drop,axis=1)
```

In [15]:

```
data_drop.shape
```

Out[15]:

```
(51, 1)
```

In [25]:

```
null_col1=[x for x in null_col if x not in drop]
```

In [26]:

```
len(null_col1)
```

Out[26]:

```
0
```

In [27]:

```
data_drop[null_col1].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51 entries, 0 to 50
Empty DataFrame
```

In [28]:

```
null_qual=[x for x in null_col1 if data_drop[x].dtype=="0"]
```

In [29]:

```
len(null_qual)
```

Out[29]:

```
0
```

In [30]:

```
null_quanti=[x for x in null_col1 if x not  in null_qula]
```

In [31]:

```
null_quanti
```

Out[31]:

```
[]
```

In [32]:

```
data_drop[null_quanti]
```

Out[32]:

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33

In [35]:

```
for x in null_quanti:
    median=data_drop[x].median()
    data_drop[x]=data_drop[x].fillna(median)
```

In [36]:

```
data_drop[null_quanti].isnull().sum()
```

Out[36]:

```
Series([], dtype: float64)
```

In [38]:

```
for x  in null_qual:
    data_drop[x]=data_drop[x].fillna('missing')
```

In [39]:

```
data_drop[null_qual].isnull().sum()
```

Out[39]:

```
Series([], dtype: float64)
```

In [40]:

```
data_md=data_drop.copy()
```

In [41]:

```
col_name=data_md.columns
```

In [42]:

```
qual_col=[x for x in col_name if data_md[x].dtype=='O']
```

In [43]:

```
len(qual_col)
```

Out[43]:

```
1
```

In [44]:

```
num_col=[x for x in col_name if data_md[x].dtype!='O']
```

In [45]:

```
len(num_col)
```

Out[45]:

```
0
```

In [46]:

```
cont_col=[x for x in num_col if len(data_md[x].unique())>25]
```

In [47]:

```
len(cont_col)
```

Out[47]:

```
0
```

In [48]:

```
disc_col=[x for x in num_col if len(data_md[x].unique())<=25]
```

In [49]:

```
len(disc_col)
```

Out[49]:

```
0
```

In [50]:

```
year_col=[x for x in col_name if "Yr" in x or 'Year' in x]
```

In [51]:

```
len(year_col)
```

Out[51]:

```
0
```

In [52]:

```
for x in cont_col:
    data_md.hist(column=x)
```

In [53]:

```
for x in cont_col:
    sns.histplot(data_md[x],kde=True,color='green')
    plot.title(x)
    plot.show()
```

In [54]:

```
for x in cont_col:
    print(data_md[x].skew())
```

In [55]:

```
r_skew=[]
l_skew=[]
for x in cont_col:
    if data_md[x].skew()>0:
        r_skew.append(x)
    else:
        l_skew.append(x)
print("right skewed are:", r_skew,"\nleft skewed are:",l_skew)
```

```
right skewed are: []
left skewed are: []
```

In [57]:

```
for x in cont_col:
    print(x,data_md[x].kurtosis())
```

In [58]:

```
lepto=[]
meso=[]
plati=[]
for x in cont_col:
    if data_hp[x].kurtosis()>3:
        lepto.append(x)
    elif data_hp[x].kurtosis()<3:
        plati.append(x)
    else:
        meso.append(x)
print('lepto--->',lepto,'\meso--->',meso,'\nplati--->',plati)
```

```
lepto---> [] \meso---> []
plati---> []
```

In [60]:

```
data_md.describe()
```

Out[60]:

	state|million_miles_annually
count	51
unique	51
top	Alabama|64914
freq	1

In [65]:

```
for x in cont_col:
    avg=desc_stat.loc['mean',x]
    sd=desc_stat.loc['std',x]
    ul=round(avg+sd,2)
    ll=round(avg-sd,1)
    print('{} has 68% of data within{}-{}range'.format(x,ll,ul))
```

In [66]:

```
outlier=[]
for x in cont_col:
    iqr=desc_stat.loc['75%',x]-desc_stat.loc['25%',x]
    ub=desc_stat.loc['75%',x]+1.5*iqr
    lb=desc_stat.loc['25%',x]-1.5*iqr


    if desc_stat.loc['max',x]>ub or desc_stat.loc['min',x]<lb:
        outlier.append(x)

print(outlier)
print(len(outlier))
```

```
[]
0
```

In [67]:

```
for x in cont_col:
    data_md.boxplot(column=x)
    plot.title(x)
    plot.show()
print(len(cont_col))
```

```
0
```

In [75]:

```
data_md.corr()
```

Out[75]:

—

In [76]:

```
corr_data=data_md.corr()
```

In [78]:

```
sp_corr=corr_data['statemillion_miles_annually']
selected=sp_corr[sp_corr>0.5].index
len(selected)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last
t)
File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3621, in In
dex.get_loc(self, key, method, tolerance)
   3620 try:
-> 3621     return self._engine.get_loc(casted_key)
   3622 except KeyError as err:

File ~\anaconda3\lib\site-packages\pandas\_libs\index.pyx:136, in pandas._
libs.index.IndexEngine.get_loc()

File ~\anaconda3\lib\site-packages\pandas\_libs\index.pyx:163, in pandas._
libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5198, in pandas._libs.hashtab
le.PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5206, in pandas._libs.hashtab
le.PyObjectHashTable.get_item()

KeyError: 'state|million_miles_annually'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last
t)
Input In [78], in <cell line: 1>()
----> 1 sp_corr=corr_data['state|million_miles_annually']
      2 selected=sp_corr[sp_corr>0.5].index
      3 len(selected)

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:3505, in DataFram
e.__getitem__(self, key)
   3503 if self.columns.nlevels > 1:
   3504     return self._getitem_multilevel(key)
-> 3505 indexer = self.columns.get_loc(key)
   3506 if is_integer(indexer):
   3507     indexer = [indexer]

File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3623, in In
dex.get_loc(self, key, method, tolerance)
   3621     return self._engine.get_loc(casted_key)
   3622 except KeyError as err:
-> 3623     raise KeyError(key) from err
   3624 except TypeError:
   3625     # If we have a listlike key, _check_indexing_error will raise
   3626     #  InvalidIndexError. Otherwise we fall through and re-raise
   3627     #  the TypeError.
   3628     self._check_indexing_error(key)

KeyError: 'state|million_miles_annually'
```

In [ ]:

```
```