Bonus Question (20 points)

Create a data quality report for the Auto-MPG dataset.

Provide the data quality tables, distributions of categorical and nominal variables.

Also provide your solutions for handling outliers and missing values.

Create the data quality tables after handling outliers and missing values.

Provide this as a separate PDF file. You can use the cells below to find statistics and create visualizations.

In [1]:

```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
adf = pd.read_csv('auto-mpg.csv')
adf.head()
```

Out[2]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	carname
0	18.0	8	307.0	130.0	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	1	ford torino

Data Quality Report before handling missing values and outliers

In [3]:

```
class DQR():
    def __init__(self):

        self.dqr_cont = pd.DataFrame(columns=['Feature','Count','% Miss','Cardinality','Min','1st Q
','Mean','Median','3rd Q','Max', 'S.D.'])
        self.dqr_cat = pd.DataFrame(columns=['Feature','Count','% Miss','Cardinality','Mode','Mode
freq','Mode %','2nd Mode','2nd Mode freq','2nd Mode %'])
        self.dqr_cont['Feature'] = ['mpg','displacement','horsepower','weight','acceleration','cyli
nders']
        self.dqr_cat['Feature'] = ['origin','carname','year']

    def report(self, data):
        data = pd.DataFrame(data)
        for col in data.columns:
            if(col == 'origin' or col == 'carname' or col == 'year'):
                mode = data[col].value_counts().index
                mode_freq = data[col].value_counts().values

                self.dqr_cat.loc[self.dqr_cat['Feature'] == col,'Count'] = data[col].count()
                self.dqr_cat.loc[self.dqr_cat['Feature'] == col,'Cardinality'] = data[col].nunique(
)

                self.dqr_cat.loc[self.dqr_cat['Feature'] == col,'Mode'] = mode[0]
                self.dqr_cat.loc[self.dqr_cat['Feature'] == col,'Mode freq'] =  mode_freq[0]
                self.dqr_cat.loc[self.dqr_cat['Feature'] == col,'Mode %'] = (mode_freq[0] /
(data[col].count()))*100
                self.dqr_cat.loc[self.dqr_cat['Feature'] == col,'2nd Mode'] = mode[1]
                self.dqr_cat.loc[self.dqr_cat['Feature'] == col,'2nd Mode'] = mode[1]
                self.dqr_cat.loc[self.dqr_cat['Feature'] == col,'2nd Mode freq'] = mode_freq[1]
                self.dqr_cat.loc[self.dqr_cat['Feature'] == col,'2nd Mode %'] = (mode_freq[1] /
(data[col].count()))*100
                self.dqr_cat.loc[self.dqr_cat['Feature'] == col,'% Miss'] = ((data[col].isnull().su
m())/ (data[col].count()))*100

            else:
                self.dqr_cont.loc[self.dqr_cont['Feature'] == col,'Count'] = data[col].count()
                self.dqr_cont.loc[self.dqr_cont['Feature'] == col,'% Miss'] = ((data[col].isnull().
sum())/ (data[col].count()))*100
                self.dqr_cont.loc[self.dqr_cont['Feature'] == col,'Cardinality'] =
data[col].nunique()
                self.dqr_cont.loc[self.dqr_cont['Feature'] == col,'Min'] = data[col].min()
                self.dqr_cont.loc[self.dqr_cont['Feature'] == col,'1st Q'] =
data[col].quantile(0.25)
                self.dqr_cont.loc[self.dqr_cont['Feature'] == col,'Mean'] = data[col].mean()
                self.dqr_cont.loc[self.dqr_cont['Feature'] == col,'Median'] = data[col].median()
                self.dqr_cont.loc[self.dqr_cont['Feature'] == col,'3rd Q'] =
data[col].quantile(0.75)
                self.dqr_cont.loc[self.dqr_cont['Feature'] == col,'Max'] = data[col].max()
                self.dqr_cont.loc[self.dqr_cont['Feature'] == col,'S.D.'] = data[col].std()


obj1 = DQR()
obj1.report(adf.copy())
# print("Data Quality report for Catgorical variable: \n")
# print(obj1.dqr_cat)
# print("Data Quality report for Continuous variable: \n")
# print(obj1.dqr_cont)
```

In [4]:

```
print("\n Data Quality report for Continuous variable:- \n")
obj1.dqr_cont.head(6)
```

```
 Data Quality report for Continuous variable:-
```

Out[4]:

	Feature	Count	% Miss	Cardinality	Min	1st Q	Mean	Median	3rd Q	Max	S.D.
0	mpg	398	2.01005	129	9	17.5	23.5146	23	29	46.6	7.81598
1	displacement	406	0	83	68	105	194.78	151	302	455	104.922
2	horsepower	400	1.5	93	46	75.75	105.082	95	130	230	38.7688
3	weight	406	0	357	19	2220	2952.31	2811	3612	5140	891.587
4	acceleration	406	0	96	8	13.7	15.5197	15.5	17.175	24.8	2.80336
5	cylinders	406	0	6	3	4	5.5	4	8	16	1.78989

In [5]:

```
print("\n Data Quality report for Categorical variable:- \n")
obj1.dqr_cat.head()
```

```
 Data Quality report for Categorical variable:-
```

Out[5]:

	Feature	Count	% Miss	Cardinality	Mode	Mode freq	Mode %	2nd Mode	2nd Mode freq	2nd Mode %
0	origin	406	0	3	1	254	62.5616	3	79	19.4581
1	carname	406	0	312	ford pinto	6	1.47783	toyota corolla	5	1.23153
2	year	406	0	13	73	40	9.85222	78	36	8.867

Handling outliers

In [6]:

```
# Answer to Q4 goes here
# print(adf.describe())



print(adf.describe())
print("Skew before :--------")
print( adf.skew())
def outlier_detection (col):
    median = col.median()
    std = col.std()
    min_range = median - 3*std
    max_range = median + 3*std
    outlier = (col[(col > max_range) | (col < min_range)])
    print("Outliers in ",col.name)
    print(outlier)
    col.loc[col > max_range] = (col[col < max_range]).max()
    col.loc[col < min_range] = (col[col > min_range]).min()
    return col

adf['mpg'] = outlier_detection(adf['mpg'].copy())
adf['displacement'] = outlier_detection(adf['displacement'].copy())
adf['horsepower'] = outlier_detection(adf['horsepower'].copy())
adf['acceleration'] = outlier_detection(adf['acceleration'].copy())
adf['weight'] = outlier_detection(adf['weight'].copy())
adf['cylinders'] = outlier_detection(adf['cylinders'].copy())


print("Skew after :--------")
print(adf.skew())
print(adf.describe())
```

```
              mpg   cylinders  displacement  horsepower       weight  \
count  398.000000  406.000000    406.000000  400.000000   406.000000
mean    23.514573    5.500000    194.779557  105.082500  2952.305419
std      7.815984    1.789889    104.922458   38.768779   891.587329
min      9.000000    3.000000     68.000000   46.000000    19.000000
25%     17.500000    4.000000    105.000000   75.750000  2220.000000
50%     23.000000    4.000000    151.000000   95.000000  2811.000000
75%     29.000000    8.000000    302.000000  130.000000  3612.000000
max     46.600000   16.000000    455.000000  230.000000  5140.000000

       acceleration        year      origin
count    406.000000  406.000000  406.000000
mean      15.519704   75.921182    1.568966
std        2.803359    3.748737    0.797479
min        8.000000   70.000000    1.000000
25%       13.700000   73.000000    1.000000
50%       15.500000   76.000000    1.000000
75%       17.175000   79.000000    2.000000
max       24.800000   82.000000    3.000000
Skew before :--------
mpg             0.457066
cylinders       0.906124
displacement    0.694130
horsepower      1.034079
weight          0.163454
acceleration    0.230224
year            0.020912
origin          0.932399
dtype: float64
Outliers in  mpg
329    46.6
Name: mpg, dtype: float64
Outliers in  displacement
Series([], Name: displacement, dtype: float64)
Outliers in  horsepower
6      220.0
7      215.0
8      225.0
19     225.0
```

```
31     215.0
101    215.0
102    225.0
123    230.0
Name: horsepower, dtype: float64
Outliers in  acceleration
306    24.8
402    24.6
Name: acceleration, dtype: float64
Outliers in  weight
194    42
227    19
344    22
398    26
Name: weight, dtype: int64
Outliers in  cylinders
260    16
Name: cylinders, dtype: int64
Skew after :---------
mpg             0.445905
cylinders       0.501657
displacement    0.694130
horsepower      0.952266
weight          0.492936
acceleration    0.187489
year            0.020912
origin          0.932399
dtype: float64
              mpg   cylinders  displacement  horsepower       weight  \
count  398.000000  406.000000    406.000000  400.000000   406.000000
mean    23.509548    5.480296    194.779557  104.857500  2967.928571
std      7.801735    1.716544    104.922458   38.111723   853.167769
min      9.000000    3.000000     68.000000   46.000000  1613.000000
25%     17.500000    4.000000    105.000000   75.750000  2220.000000
50%     23.000000    4.000000    151.000000   95.000000  2811.000000
75%     29.000000    8.000000    302.000000  130.000000  3612.000000
max     44.600000    8.000000    455.000000  210.000000  5140.000000

       acceleration        year      origin
count    406.000000  406.000000  406.000000
mean      15.514778   75.921182    1.568966
std        2.788013    3.748737    0.797479
min        8.000000   70.000000    1.000000
25%       13.700000   73.000000    1.000000
50%       15.500000   76.000000    1.000000
75%       17.175000   79.000000    2.000000
max       23.700000   82.000000    3.000000
```

Handling missing Values

In [7]:

```
# Answer to Q5 goes here
# your code ....
from sklearn.impute import KNNImputer
knn_imputer = KNNImputer(n_neighbors=3)
impute_copy = adf[['mpg', 'horsepower', 'cylinders', 'displacement', 'weight']].copy()
print((impute_copy.isnull()).sum())

adf_transformed = knn_imputer.fit_transform(impute_copy)

print(sum(np.isnan(adf_transformed)))

adf_trans = pd.DataFrame(index=range(adf.shape[0]), columns=['mpg', 'horsepower', 'cylinders', 'dis
placement', 'weight'])
adf_trans = pd.DataFrame(adf_transformed, dtype=None, copy=False,index = adf_trans.index, columns=a
df_trans.columns )
adf_trans[['acceleration','year','origin','carname']] = 
adf[['acceleration','year','origin','carname']]
```

```
mpg             8
horsepower      6
cylinders       0
displacement    0
```

```
weight          0
dtype: int64
[0 0 0 0 0]
```

In [8]:

```
adf_trans.head()
```

Out[8]:

	mpg	horsepower	cylinders	displacement	weight	acceleration	year	origin	carname
0	18.0	130.0	8.0	307.0	3504.0	12.0	70	1	chevrolet chevelle malibu
1	15.0	165.0	8.0	350.0	3693.0	11.5	70	1	buick skylark 320
2	18.0	150.0	8.0	318.0	3436.0	11.0	70	1	plymouth satellite
3	16.0	150.0	8.0	304.0	3433.0	12.0	70	1	amc rebel sst
4	17.0	140.0	8.0	302.0	3449.0	10.5	70	1	ford torino

Data Quality Report after handling outliers and missing values

In [9]:

```
obj2 = DQR()
obj2.report(adf_trans)
```

In [10]:

```
print("\n Data Quality report for Continuous variable:- \n")
obj2.dqr_cont.head(6)
```

```
 Data Quality report for Continuous variable:-
```

Out[10]:

	Feature	Count	% Miss	Cardinality	Min	1st Q	Mean	Median	3rd Q	Max	S.D.
0	mpg	406	0	133	9	17	23.435	22.75	29	44.6	7.80953
1	displacement	406	0	83	68	105	194.78	151	302	455	104.922
2	horsepower	406	0	94	46	75	104.547	95	129	210	37.9754
3	weight	406	0	353	1613	2220	2967.93	2811	3612	5140	853.168
4	acceleration	406	0	94	8	13.7	15.5148	15.5	17.175	23.7	2.78801
5	cylinders	406	0	5	3	4	5.4803	4	8	8	1.71654

In [11]:

```
print("\n Data Quality report for Catgorical variable:------ \n")
obj2.dqr_cat.head()
```

```
 Data Quality report for Catgorical variable:------
```

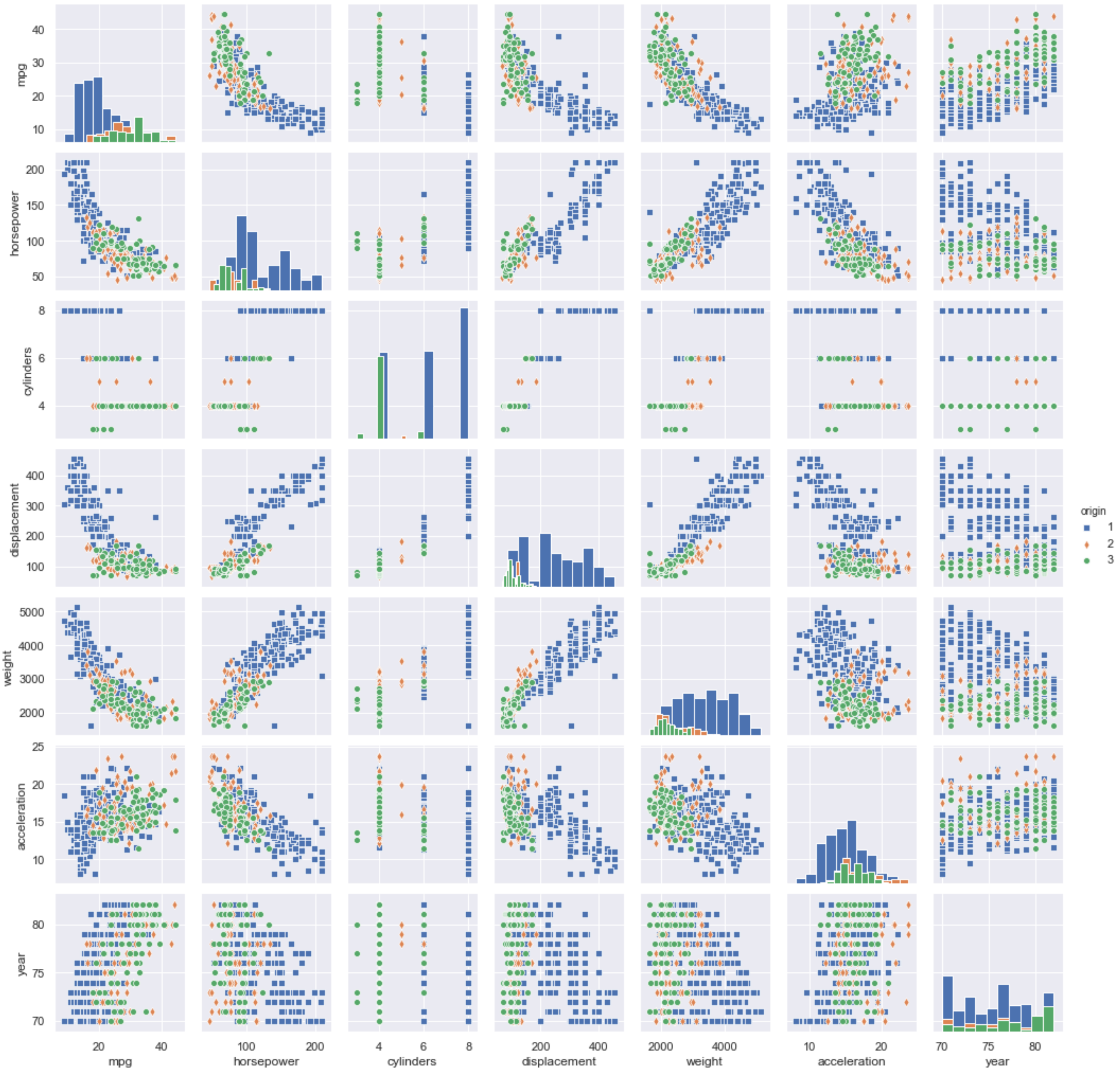
Out[11]:

	Feature	Count	% Miss	Cardinality	Mode	Mode freq	Mode %	2nd Mode	2nd Mode freq	2nd Mode %
0	origin	406	0	3	1	254	62.5616	3	79	19.4581
1	carname	406	0	312	ford pinto	6	1.47783	toyota corolla	5	1.23153
2		406	0	13	73	40	9.85222	78	36	8.867

2	year	406	0	13	73	40	9.85222	78	36	8.867
	Feature	**Count**	**% Miss**	**Cardinality**	**Mode**	**Mode freq**	**Mode %**	**2nd Mode**	**2nd Mode**	**2nd Mode**

In [12]:

```
import seaborn as sns
sns.set()
sns.pairplot(adf_trans,hue ='origin',diag_kind='hist', markers = ['s','d','o'], height = 2.0)
plt.show()
```


mpg
horsepower
cylinders
displacement
weight
acceleration
year
origin