## 2.2 Problem 2

Load the auto-mpg sample dataset into Python using a Pandas dataframe. The horsepower feature has a few missing values with a? - replace these with a NaN from NumPy, and calculate summary statistics for each numerical column. How do the summary statistics vary when excluding the NaNs, vs. imputing them with the mean (Hint: Use an Imputer from Scikit) - can we do better than just using the overall sample mean?

## **Answer:**

Summary statistics excluding NaN –

count 398 unique 94 top 150 freq 22

Name: horsepower, dtype: object

## Summary statistics after imputing -

 count
 392.000000

 mean
 104.469388

 std
 38.491160

 min
 46.000000

 25%
 NaN

 50%
 NaN

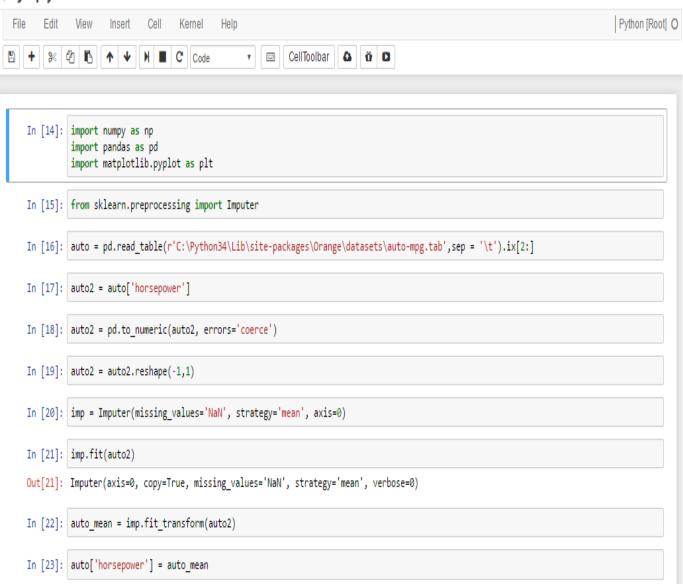
 75%
 NaN

 max
 230.000000

Name: horsepower, dtype: float64

Here, the count has changed as the missing values has been replaced by the mean. Yes, we can replace the values by using median or mode and improve the overall result but standard deviation increases.

## Jupyter SUM 2 HW1 (autosaved)



```
In [24]: auto.describe()
Out[24]:
                   horsepower
            count 398.000000
                  104.469388
            std
                  38.199187
                   46.000000
            min
            25%
                   76.000000
           50%
                   95.000000
            75%
                   125.000000
                  230.000000
           max
In [25]: imp = Imputer(missing_values='NaN', strategy='median', axis=0)
           imp = Imputer(missing_values= waw, str
imp.fit(auto2)
auto_median = imp.fit_transform(auto2)
auto['horsepower'] = auto_median
auto.describe()
Out[25]:
                   horsepower
            count 398.000000
                  104.304020
                   38.222625
                   46.000000
           25%
                  76.000000
                  93.500000
            50%
           75%
                   125.000000
           max
                  230.000000
In [26]: imp = Imputer(missing_values='NaN', strategy='most_frequent', axis=0)
           imp.fit(auto2)
           auto_frequent = imp.fit_transform(auto2)
           auto['horsepower'] = auto_frequent
           auto.describe()
Out[26]:
                   horsepower
           count 398.000000
           mean 105.155779
                   38.600986
                   46.000000
           min
           25%
                  76.000000
           50%
                   95.000000
           75%
                   130.000000
                  230.000000
```