Data Manipulation Exercises

- Data Cleaning & Preparation Exercises
  - Dealing with Missing & Duplicated Data
  - String Manipulation (Regular Expression)
  - Data Transformation

Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
sns.set()
```

==========

Data Cleaning

Importing Data

In [2]:
```
cars = pd.read_csv('mpg-unclean.csv')
```

Inspecting the DataFrame and Identifing the Inconsistent Data

In [6]:
```
cars.head(20)
```

Out[6]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	
0	18.0	8	307.0	130.0 hp	3504	12.0	70	United States	ch c
1	15.0	8	350.0	165.0 hp	3693	11.5	70	United States	skyl
2	18.0	8	318.0	150.0 hp	3436	11.0	70	United States	ply s
3	16.0	8	304.0	150.0 hp	3433	12.0	70	usa	am
4	17.0	8	302.0	140.0 hp	3449	10.5	70	usa	T
5	15.0	8	429.0	198.0 hp	4341	10.0	70	usa	G
6	14.0	8	454.0	220.0 hp	4354	9.0	70	usa	ch
7	14.0	8	440.0	215.0 hp	4312	8.5	70	usa	ply
8	14.0	8	455.0	225.0 hp	4425	10.0	70	usa	I c
9	15.0	8	390.0	190.0 hp	3850	8.5	70	usa	amba
10	15.0	8	383.0	170.0 hp	3563	10.0	70	usa	cha
11	14.0	8	340.0	160.0 hp	3609	8.0	70	usa	ply 'cu
12	15.0	8	400.0	150.0 hp	3761	9.5	70	usa	ch mont
13	14.0	8	455.0	225.0 hp	3086	10.0	70	usa	buick wag
14	22.0	6	198.0	95.0 hp	2833	15.5	70	usa	ply
15	18.0	6	199.0	97.0 hp	2774	15.5	70	usa	amc
16	21.0	6	200.0	85.0 hp	2587	16.0	70	usa	m

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	
17	26.0	4	97.0	46.0 hp	1835	20.5	70	europe	v
18	25.0	4	110.0	87.0 hp	2672	17.5	70	europe	p
19	24.0	4	107.0	90.0 hp	2430	14.5	70	europe	aud

In [8]:
```
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 329 entries, 0 to 328
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           329 non-null    float64
 1   cylinders     329 non-null    int64  
 2   displacement  329 non-null    float64
 3   horsepower    329 non-null    object 
 4   weight        329 non-null    int64  
 5   acceleration  329 non-null    float64
 6   model year    329 non-null    int64  
 7   origin        329 non-null    object 
 8   name          329 non-null    object 
dtypes: float64(3), int64(3), object(3)
memory usage: 23.3+ KB
```

In [9]:
```
cars.describe().round(3)
```

Out[9]:

	mpg	cylinders	displacement	weight	acceleration	model year
count	329.000	329.000	329.000	329.000	329.000	329.000
mean	21.655	5.802	217.005	3229.082	15.400	104.495
std	7.295	1.720	104.719	1376.307	2.923	232.499
min	0.061	4.000	68.000	1800.000	8.000	70.000
25%	16.000	4.000	121.000	2464.000	13.500	73.000
50%	20.200	6.000	200.000	3102.000	15.400	76.000
75%	26.000	8.000	305.000	3821.000	17.000	79.000
max	44.300	8.000	455.000	23000.000	24.800	1973.000

In [14]:
```
cars.plot(subplots =  True, figsize=(20,10))
plt.show()
```

In [ ]:

Q. Identify one column label that should be changed and adjust/rename the column label!

In [15]:
```
cars.columns
```

Out[15]:
```
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
       'acceleration', 'model year', 'origin', 'name'],
      dtype='object')
```

In [16]:
```
cars.rename(columns = {'model year' : 'model_year'}, inplace= True)
```

In [17]:
```
cars
```

Out[17]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
0	18.0	8	307.0	130.0 hp	3504	12.0	70	United States
1	15.0	8	350.0	165.0 hp	3693	11.5	70	United States
2	18.0	8	318.0	150.0 hp	3436	11.0	70	United States
3	16.0	8	304.0	150.0 hp	3433	12.0	70	usa
4	17.0	8	302.0	140.0 hp	3449	10.5	70	usa
...	...	...	...	...	...	...	...	...
324	12.0	8	429.0	198.0 hp	4952	11.5	73	usa
325	27.0	4	101.0	83.0 hp	2202	15.3	76	europe
326	17.0	6	250.0	100.0 hp	3329	15.5	71	usa
327	14.5	8	351.0	152.0 hp	4215	12.8	76	usa
328	25.0	6	181.0	110.0 hp	2945	16.4	82	usa

329 rows × 9 columns

In [ ]:

Q. Have a closer look to the origin column by analyzing the frequency/count of unique values! Can you find any inconsistency?

In [18]:
```
cars['origin'].value_counts()
```

Out[18]:
```
origin
usa              254
europe            72
United States      3
Name: count, dtype: int64
```

In [ ]:

Q. Replace the value "United States" in the origin column! Save the change!

In [21]:
```
cars.origin.replace({'United States' : 'usa'}, inplace=True)
```

In [24]:
```
cars
```

Out[24]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
0	18.0	8	307.0	130.0	3504	12.0	70	usa
1	15.0	8	350.0	165.0	3693	11.5	70	usa
2	18.0	8	318.0	150.0	3436	11.0	70	usa
3	16.0	8	304.0	150.0	3433	12.0	70	usa
4	17.0	8	302.0	140.0	3449	10.5	70	usa
...	...	...	...	...	...	...	...	...
324	12.0	8	429.0	198.0	4952	11.5	73	usa
325	27.0	4	101.0	83.0	2202	15.3	76	europe
326	17.0	6	250.0	100.0	3329	15.5	71	usa
327	14.5	8	351.0	152.0	4215	12.8	76	usa
328	25.0	6	181.0	110.0	2945	16.4	82	usa

329 rows × 9 columns

Q. Inspect and identify the problem in the column horsepower!

In [22]:
```
cars['horsepower'] = cars['horsepower'].str.replace(" hp", "")
```

In [23]:
```
cars
```

Out[23]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
0	18.0	8	307.0	130.0	3504	12.0	70	usa
1	15.0	8	350.0	165.0	3693	11.5	70	usa
2	18.0	8	318.0	150.0	3436	11.0	70	usa
3	16.0	8	304.0	150.0	3433	12.0	70	usa
4	17.0	8	302.0	140.0	3449	10.5	70	usa
...	...	...	...	...	...	...	...	...
324	12.0	8	429.0	198.0	4952	11.5	73	usa
325	27.0	4	101.0	83.0	2202	15.3	76	europe
326	17.0	6	250.0	100.0	3329	15.5	71	usa
327	14.5	8	351.0	152.0	4215	12.8	76	usa
328	25.0	6	181.0	110.0	2945	16.4	82	usa

329 rows × 9 columns

In [ ]:

Q. Now you can convert the datatype in the column horsepower! Overwrite the column!

In [27]:
```
cars.horsepower.replace('Not available', np.nan, inplace=True)
```

In [30]:
```
cars.horsepower = cars.horsepower.astype('float')
```

In [31]:
```
cars.dtypes
```

Out[31]:
```
mpg             float64
cylinders         int64
displacement    float64
horsepower      float64
weight            int64
acceleration    float64
model_year        int64
origin           object
name             object
dtype: object
```

In [ ]:

Q. What about the 'name' column?

In [32]:
```
cars.name
```

Out[32]:
```
0        chevrolet chevelle malibu
1                buick skylark 320
2               plymouth satellite
3                    amc rebel sst
4                      FORD TORINO
                    ...
324       mercury marquis brougham
325                    renault 12tl
326      chevrolet chevelle malibu
327                ford gran torino
328           buick century limited
Name: name, Length: 329, dtype: object
```

In [33]:
```
cars.name = cars.name.str.lower().str.strip()
```

In [34]:
```
cars
```

Out[34]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
0	18.0	8	307.0	130.0	3504	12.0	70	usa
1	15.0	8	350.0	165.0	3693	11.5	70	usa
2	18.0	8	318.0	150.0	3436	11.0	70	usa
3	16.0	8	304.0	150.0	3433	12.0	70	usa
4	17.0	8	302.0	140.0	3449	10.5	70	usa
...	...	...	...	...	...	...	...	...
324	12.0	8	429.0	198.0	4952	11.5	73	usa
325	27.0	4	101.0	83.0	2202	15.3	76	europe
326	17.0	6	250.0	100.0	3329	15.5	71	usa
327	14.5	8	351.0	152.0	4215	12.8	76	usa
328	25.0	6	181.0	110.0	2945	16.4	82	usa

329 rows × 9 columns

In [ ]:

Q. Inspect the column **model_year** in more detail by analyzing the **frequency/counts** of unique values! Anything **strange**?

In [36]:
```
cars.model_year
```

Out[36]:
```
0      70
1      70
2      70
3      70
4      70
       ..
324    73
325    76
326    71
327    76
328    82
Name: model_year, Length: 329, dtype: int64
```

In [37]:
```
cars.model_year.value_counts()
```

Out[37]:
```
model_year
73      33
76      32
70      29
78      28
79      27
75      26
71      25
82      23
72      23
74      22
77      22
80      17
81      17
1973     5
Name: count, dtype: int64
```

In [38]:
```
cars.model_year.replace(1973,73, inplace=True)
```

In [39]:
```
cars
```

Out[39]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
0	18.0	8	307.0	130.0	3504	12.0	70	usa
1	15.0	8	350.0	165.0	3693	11.5	70	usa
2	18.0	8	318.0	150.0	3436	11.0	70	usa
3	16.0	8	304.0	150.0	3433	12.0	70	usa
4	17.0	8	302.0	140.0	3449	10.5	70	usa
...	...	...	...	...	...	...	...	...
324	12.0	8	429.0	198.0	4952	11.5	73	usa
325	27.0	4	101.0	83.0	2202	15.3	76	europe
326	17.0	6	250.0	100.0	3329	15.5	71	usa
327	14.5	8	351.0	152.0	4215	12.8	76	usa
328	25.0	6	181.0	110.0	2945	16.4	82	usa

329 rows × 9 columns

In [40]:
```
cars.describe()
```

Out[40]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	mod
count	329.000000	329.000000	329.000000	323.000000	329.000000	329.000000	329.
mean	21.655199	5.802432	217.004559	111.185759	3229.082067	15.400000	75.
std	7.294651	1.719825	104.719009	39.846088	1376.306985	2.922756	3.
min	0.060606	4.000000	68.000000	46.000000	1800.000000	8.000000	70.
25%	16.000000	4.000000	121.000000	83.500000	2464.000000	13.500000	73.
50%	20.200000	6.000000	200.000000	100.000000	3102.000000	15.400000	76.
75%	26.000000	8.000000	305.000000	141.000000	3821.000000	17.000000	79.
max	44.300000	8.000000	455.000000	230.000000	23000.000000	24.800000	82.

In [ ]:

Q. Inspect the column weight by sorting the values from high to low. Can you see the extreme value?

In [41]:
```
cars.weight.max()
```

Out[41]:
```
np.int64(23000)
```

In [42]:
```
cars.weight.idxmax()  # record of max weight
```

Out[42]:
```
126
```

In [45]:
```
cars.loc[cars.weight.idxmax(), 'weight'] = 2300
```

In [46]:
```
cars
```

Out[46]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
0	18.0	8	307.0	130.0	3504	12.0	70	usa
1	15.0	8	350.0	165.0	3693	11.5	70	usa
2	18.0	8	318.0	150.0	3436	11.0	70	usa
3	16.0	8	304.0	150.0	3433	12.0	70	usa
4	17.0	8	302.0	140.0	3449	10.5	70	usa
...	...	...	...	...	...	...	...	...
324	12.0	8	429.0	198.0	4952	11.5	73	usa
325	27.0	4	101.0	83.0	2202	15.3	76	europe
326	17.0	6	250.0	100.0	3329	15.5	71	usa
327	14.5	8	351.0	152.0	4215	12.8	76	usa
328	25.0	6	181.0	110.0	2945	16.4	82	usa

329 rows × 9 columns

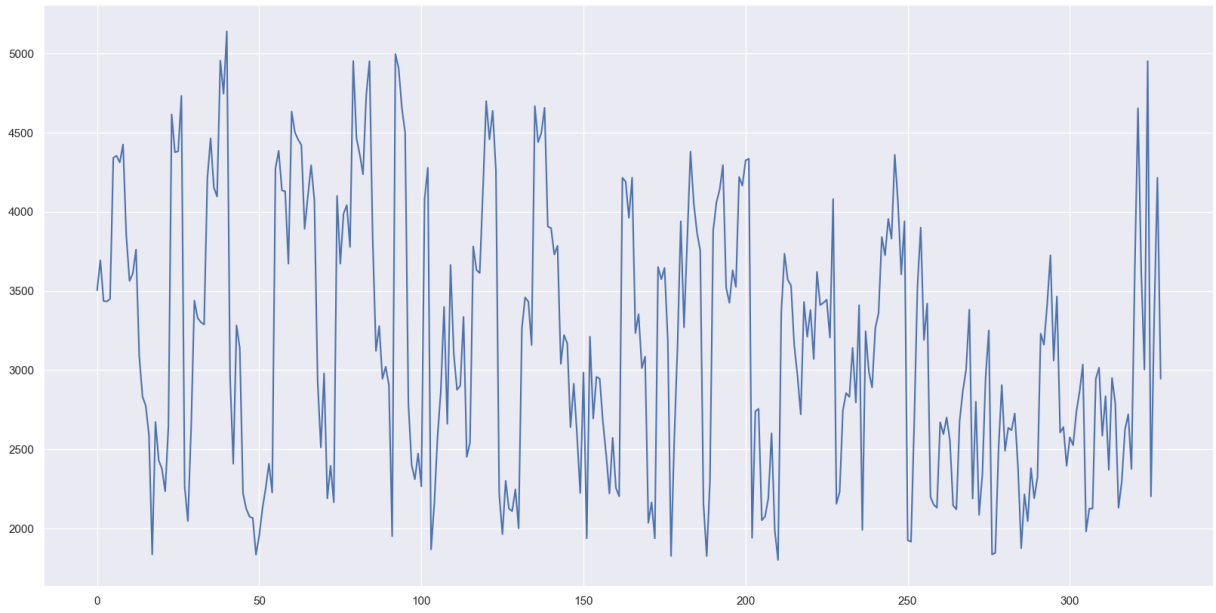
In [47]:
```
cars.describe()
```

Out[47]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	mode
count	329.000000	329.000000	329.000000	323.000000	329.000000	329.000000	329.0
mean	21.655199	5.802432	217.004559	111.185759	3166.164134	15.400000	75.6
std	7.294651	1.719825	104.719009	39.846088	837.344920	2.922756	3.6
min	0.060606	4.000000	68.000000	46.000000	1800.000000	8.000000	70.0
25%	16.000000	4.000000	121.000000	83.500000	2451.000000	13.500000	73.0
50%	20.200000	6.000000	200.000000	100.000000	3086.000000	15.400000	76.0
75%	26.000000	8.000000	305.000000	141.000000	3820.000000	17.000000	79.0
max	44.300000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.0

In [49]:
```
plt.figure(figsize=(20,10))
cars.weight.plot()
```

Out[49]: <Axes: >



In [ ]:
```
```

Q. Let's check out the column mpg too

In [52]:
```
cars.loc[cars['mpg'].idxmin()]
```

Out[52]:
```
mpg                          0.060606
cylinders                           8
displacement                    351.0
horsepower                      138.0
weight                           3955
acceleration                     13.2
model_year                         79
origin                            usa
name            mercury grand marquis
Name: 244, dtype: object
```

In [53]:
```
cars.loc[cars['mpg'].idxmin(), 'mpg'] = 1/cars.loc[cars['mpg'].idxmin(), 'mpg']
```

In [54]:
```
cars
```

Out[54]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
0	18.0	8	307.0	130.0	3504	12.0	70	usa
1	15.0	8	350.0	165.0	3693	11.5	70	usa
2	18.0	8	318.0	150.0	3436	11.0	70	usa
3	16.0	8	304.0	150.0	3433	12.0	70	usa
4	17.0	8	302.0	140.0	3449	10.5	70	usa
...	...	...	...	...	...	...	...	...
324	12.0	8	429.0	198.0	4952	11.5	73	usa
325	27.0	4	101.0	83.0	2202	15.3	76	europe
326	17.0	6	250.0	100.0	3329	15.5	71	usa
327	14.5	8	351.0	152.0	4215	12.8	76	usa
328	25.0	6	181.0	110.0	2945	16.4	82	usa

329 rows × 9 columns

In [55]:
```
cars.describe().round(2)
```

Out[55]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year
count	329.00	329.00	329.00	323.00	329.00	329.00	329.00
mean	21.71	5.80	217.00	111.19	3166.16	15.40	75.62
std	7.20	1.72	104.72	39.85	837.34	2.92	3.64
min	9.00	4.00	68.00	46.00	1800.00	8.00	70.00
25%	16.00	4.00	121.00	83.50	2451.00	13.50	73.00
50%	20.20	6.00	200.00	100.00	3086.00	15.40	76.00
75%	26.00	8.00	305.00	141.00	3820.00	17.00	79.00
max	44.30	8.00	455.00	230.00	5140.00	24.80	82.00

In [ ]:
```

```

Q. Select all rows with at least one missing/na value!

In [57]:
```
cars.isna().sum()
```

Out[57]:
```
mpg             0
cylinders       0
displacement    0
horsepower      6
weight          0
acceleration    0
model_year      0
origin          0
name            0
dtype: int64
```

In [59]:
```
cars.dropna(subset=['horsepower'], inplace=True)
```

In [60]:
```
cars.isna().sum()
```

Out[60]:
```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model_year      0
origin          0
name            0
dtype: int64
```

In [ ]:
```

```

Q. Finding the duplicated records in care names

In [63]:
```
cars.duplicated().any(axis=0)  # axis = 0 for col not like pandas
```

Out[63]: np.True_

In [64]:
```
cars.duplicated().sum()
```

Out[64]: np.int64(10)

In [67]:
```
cars.loc[cars.duplicated(subset=['name'])].sort_values('name')
```

Out[67]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
323	24.3	4	151.0	90.0	3003	20.1	80	usa
268	24.3	4	151.0	90.0	3003	20.1	80	usa
29	19.0	6	232.0	100.0	2634	13.0	71	usa
147	20.0	6	232.0	100.0	2914	16.0	75	usa
96	18.0	6	232.0	100.0	2789	15.0	73	usa
...	...	...	...	...	...	...	...	...
172	29.0	4	90.0	70.0	1937	14.2	76	europe
177	29.5	4	97.0	71.0	1825	12.2	76	europe
277	29.8	4	89.0	62.0	1845	15.3	80	europe
264	41.5	4	98.0	76.0	2144	14.7	80	europe
250	31.9	4	89.0	71.0	1925	14.0	79	europe

86 rows × 9 columns

In [68]:
```
cars.drop_duplicates(inplace=True)
```

In [69]:
```
cars
```

Out[69]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
0	18.0	8	307.0	130.0	3504	12.0	70	usa
1	15.0	8	350.0	165.0	3693	11.5	70	usa
2	18.0	8	318.0	150.0	3436	11.0	70	usa
3	16.0	8	304.0	150.0	3433	12.0	70	usa
4	17.0	8	302.0	140.0	3449	10.5	70	usa
...	...	...	...	...	...	...	...	...
314	27.0	4	140.0	86.0	2790	15.6	82	usa
315	44.0	4	97.0	52.0	2130	24.6	82	europe
316	32.0	4	135.0	84.0	2295	11.6	82	usa
317	28.0	4	120.0	79.0	2625	18.6	82	usa
318	31.0	4	119.0	82.0	2720	19.4	82	usa

313 rows × 9 columns

In [ ]:

Q. It's a good practice to save the cleaned version of your dataset again

In [70]:
```
cars.to_csv('mpg-cleanedd.csv')
```

In [ ]:

==========

String Manipulation (Regular Expression)

Check if String Contain Only Defined Characters using Regex

In [71]:
```
import re
if re.search(r'^[1234]+$', '2134'):
    print(True)
```

```
True
```

In [ ]:
```

```

Count Uppercase, Lowercase, and numeric values using Regex

In [76]:
```
s = 'My name is Atlas Home, I am 43 years old'

import re
upper = re.findall(r'[A-Z]', s)
lower = re.findall(r'[a-z]', s)
numeric = re.findall(r'[0-9]', s)

print('The no. of uppercase characters is:', len(upper))
print('The no. of lowercase characters is:', len(lower))
print('The no. of numerical characters is:', len(numeric))
```

```
The no. of uppercase characters is: 4
The no. of lowercase characters is: 24
The no. of numerical characters is: 2
```

In [ ]:
```

```

Regex to extract maximum numeric value from a string

In [77]:
```
s = '100khj26io58sgtq1723mnb'
import re
numeric = re.findall(r'\d+', s)
max([int(i) for i in numeric])
```

Out[77]:
```
1723
```

In [ ]:
```

```

Remove all characters except letters and numbers

In [78]:
```
s = "123abcjw:, .@! eiw"
re.sub('[\W_]+', '', s)
```

```
<>:2: SyntaxWarning: invalid escape sequence '\W'
<>:2: SyntaxWarning: invalid escape sequence '\W'
C:\Users\alhef\AppData\Local\Temp\ipykernel_18248\2561307421.py:2: SyntaxWarning: in
valid escape sequence '\W'
  re.sub('[\W_]+', '', s)
```

Out[78]:
```
'123abcjweiw'
```

In [ ]:
```

```

Regex to put spaces between words starting with capital letters

In [79]:
```
results = re.findall('[A-Z][a-z]*', 'AtlasSoft-Home')
' '.join(results)
```

Out[79]: 'Atlas Soft Home'

==========

GOOD LUCK!