SLICING & DICING A MOTORCARS DATASET WITH PYTHON

EUROPEAN + JAPANESE CARS

DR. ALVIN ANG

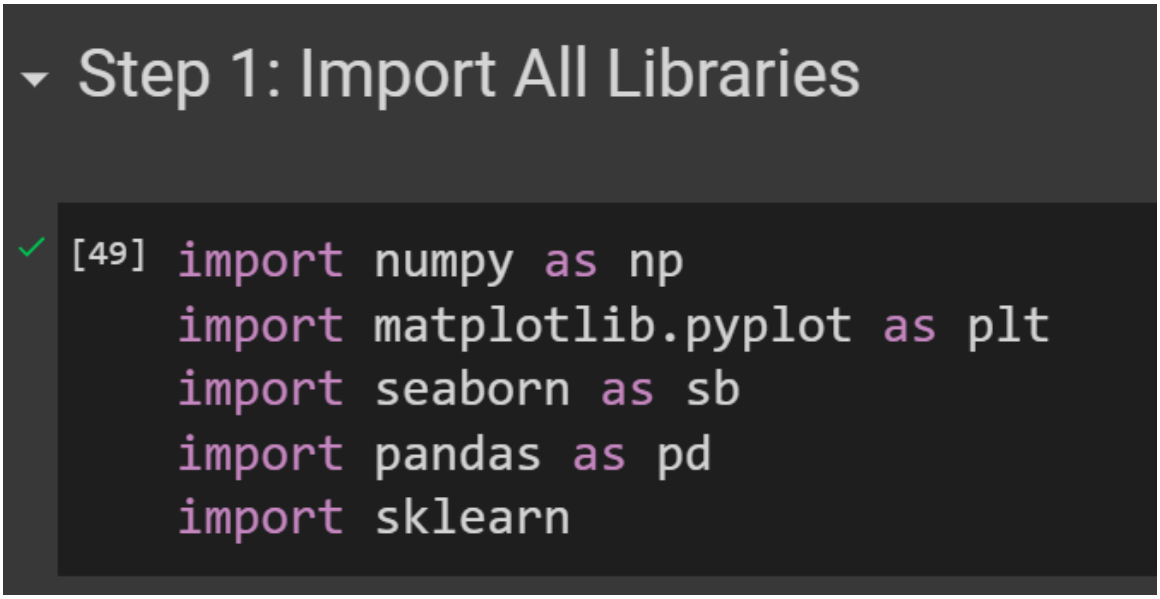
CONTENTS

I. STEP 1: IMPORT ALL LIBRARIES

https://www.alvinang.sg/s/mtcars.csv

https://www.alvinang.sg/s/SLICING__DICING_A_MOTORCARS_DATASET_WITH_PYTHON_by_Dr_Alvin_Ang.ipynb

	A	B	C	D	E	F	G	H	I	J	K	L
1	car_names	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
2	Mazda RX4	21	6	160	110	3.9	2.62	16.46	0	1	4	4
3	Mazda RX4 Wag	21	6	160	110	3.9	2.875	17.02	0	1	4	4
4	Datsun 710	22.8	4	108	93	3.85	2.32	18.61	1	1	4	1
5	Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
6	Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.02	0	0	3	2
7	Valiant	18.1	6	225	105	2.76	3.46	20.22	1	0	3	1



II. STEP 2: EXPLORING THE MTCARS.CSV DATASET USING DATAPREP.AI

A. READING IN THE CSV

Step 2: Exploring the Mtcars.csv Dataset using DataPrep.ai

https://dataprep.ai/

2a) Reading in the CSV

```
[50] mtcars = pd.read_csv('https://www.alvinang.sg/s/mtcars.csv')
```

```
[51] mtcars.sample(5)
```

	car_names	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
24	Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
22	AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
11	Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
12	Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3

B. USING DATAPREP TO PREVIEW

2b) Using DataPrep to Preview

```
!pip install -U dataprep
```

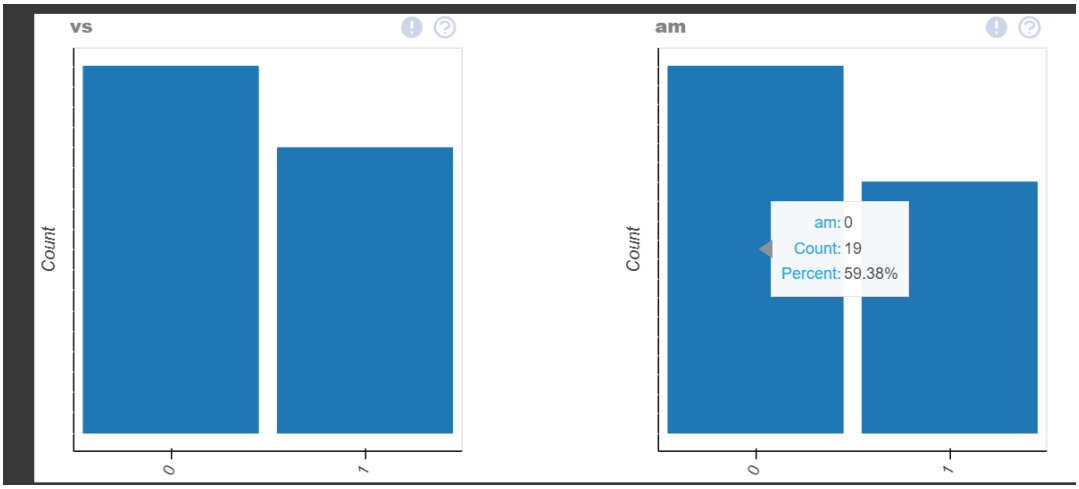
```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: dataprep in /usr/local/lib/python3.7/dist-packages (0.4.3)
Requirement already satisfied: python-stdnum<2.0,>=1.16 in /usr/local/lib/python3.7/dist-packages (from da
Requirement already satisfied: aiohttp<4.0,>=3.6 in /usr/local/lib/python3.7/dist-packages (from dataprep)
Requirement already satisfied: nltk<4.0.0,>=3.6.7 in /usr/local/lib/python3.7/dist-packages (from dataprep
Requirement already satisfied: pydantic<2.0,>=1.6 in /usr/local/lib/python3.7/dist-packages (from dataprep
Requirement already satisfied: flask_cors<4.0.0,>=3.0.10 in /usr/local/lib/python3.7/dist-packages (from d
Requirement already satisfied: bokeh<3,>=2 in /usr/local/lib/python3.7/dist-packages (from dataprep) (2.3.
Requirement already satisfied: pandas<2.0,>=1.1 in /usr/local/lib/python3.7/dist-packages (from dataprep)
Requirement already satisfied: flask<3,>=2 in /usr/local/lib/python3.7/dist-packages (from dataprep) (2.1.
Requirement already satisfied: scipy<=1.7.1 in /usr/local/lib/python3.7/dist-packages (from dataprep) (1.4
Requirement already satisfied: jsonpath-ng<2.0,>=1.5 in /usr/local/lib/python3.7/dist-packages (from datap
Requirement already satisfied: pystache<0.7.0,>=0.6.0 in /usr/local/lib/python3.7/dist-packages (from data
```

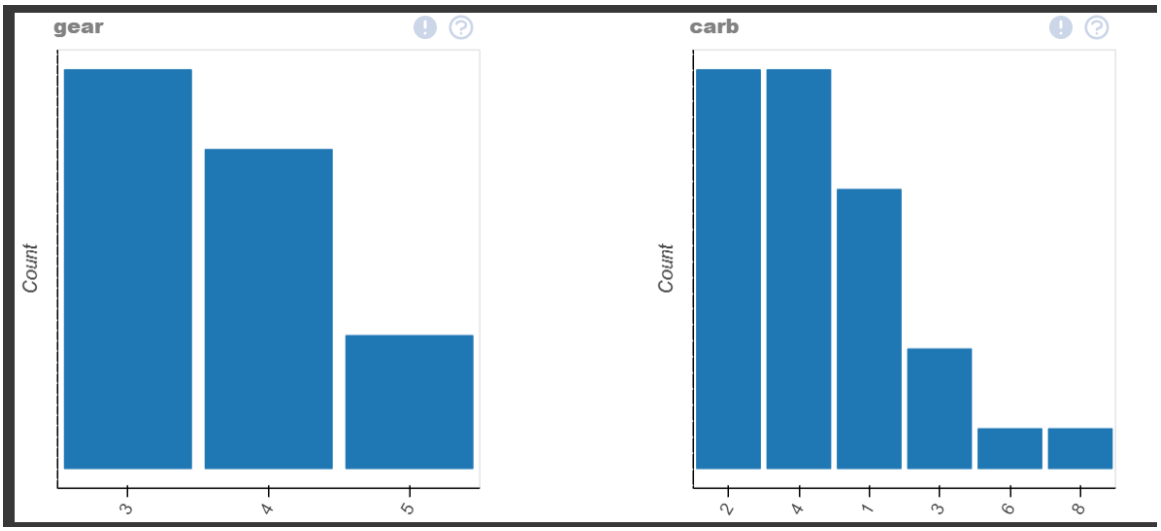
```
from dataprep.eda import plot

plot(mtcars)
```

Hide Stats and Insights

Dataset Statistics	
Number of Variables	12
Number of Rows	32
Missing Cells	0
Missing Cells (%)	0.0%
Duplicate Rows	0
Duplicate Rows (%)	0.0%
Total Size in Memory	5.0 KB
Average Row Size in Memory	160.9 B
Variable Types	Categorical: 6 Numerical: 6

Dataset Insights	
`drat` and `wt` have similar distributions	Similar Distribution
`hp` is skewed	Skewed
`drat` is skewed	Skewed
`wt` is skewed	Skewed
`cyl` has constant length 1	Constant Length
`vs` has constant length 1	Constant Length
`am` has constant length 1	Constant Length
`gear` has constant length 1	Constant Length
`carb` has constant length 1	Constant Length
`car_names` has all distinct values	Unique





C. USING DATAPREP TO CREATE REPORT

2c) Using Dataprep to Create Report

```
[54] from dataprep.eda import create_report

     create_report(mtcars)
```

DataPrep Report | Overview | Variables ≡ | Interactions | Correlations | Missing Values

Overview

Overview

Dataset Statistics

Number of Variables	12
Number of Rows	32
Missing Cells	0
Missing Cells (%)	0.0%
Duplicate Rows	0
Duplicate Rows (%)	0.0%
Total Size in Memory	5.0 KB
Average Row Size in Memory	160.9 B
Variable Types	Categorical: 6 Numerical: 6

Dataset Insights

`drat` and `wt` have similar distributions	Similar Distribution
`hp` is skewed	Skewed
`drat` is skewed	Skewed
`wt` is skewed	Skewed
`cyl` has constant length 1	Constant Length
`vs` has constant length 1	Constant Length
`am` has constant length 1	Constant Length
`gear` has constant length 1	Constant Length
`carb` has constant length 1	Constant Length
`car_nam…` has all distinct values	Unique

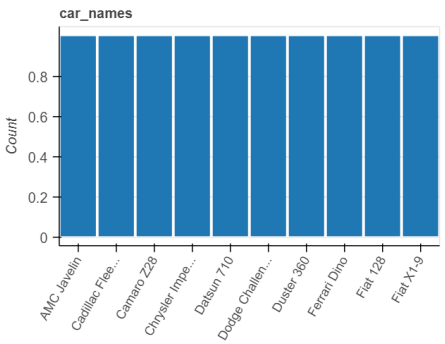
Variables

Sort by Feature order ☐ Reverse order

car_names
categorical

Hide Details

Approximate Distinct Count	32
Approximate Unique (%)	100.0%
Missing	0
Missing (%)	0.0%
Memory Size	2.4 KB



Top 10 of 32 car_names

Stats | Pie Chart | Word Frequency | Word Length

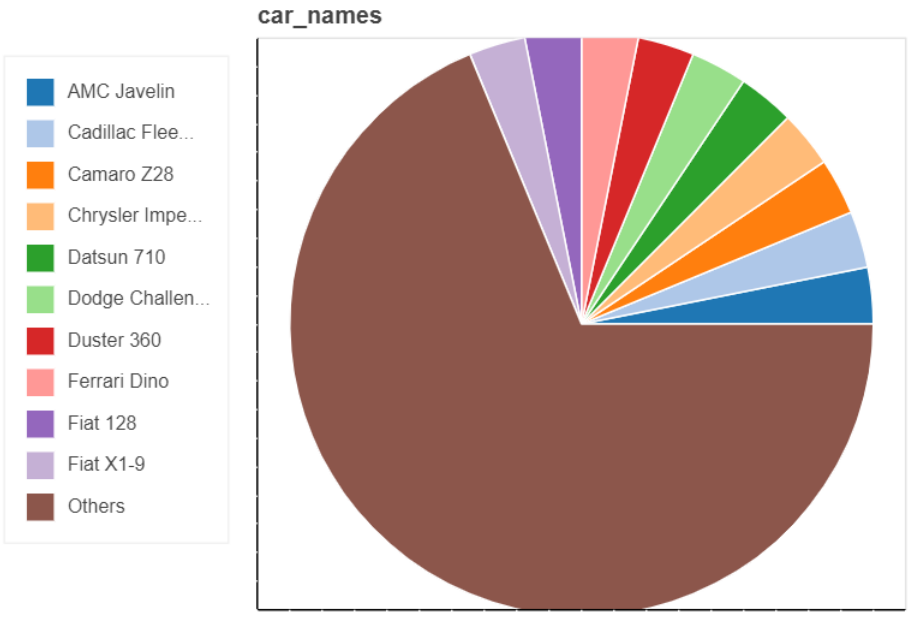
Length	
Mean	11.9062
Standard Deviation	3.2064
Median	11
Minimum	7
Maximum	19

Sample	
1st row	Mazda RX4
2nd row	Mazda RX4 Wag
3rd row	Datsun 710
4th row	Hornet 4 Drive
5th row	Hornet Sportabout

Letter	
Count	301
Lowercase Letter	234
Space Separator	34
Uppercase Letter	67
Dash Punctuation	2
Decimal Number	44

Top 10 of 32 car_names

Stats | Pie Chart | Word Frequency | Word Length



Step 3: Learning to Export / Import Mtcars as .CSV / .XLS

```
[55] #export to CSV

     mtcars.to_csv('cars.csv')
```

```
[56] #export to XLS

     mtcars.to_excel('cars.xlsx',
                              sheet_name='cars', index=False)
```

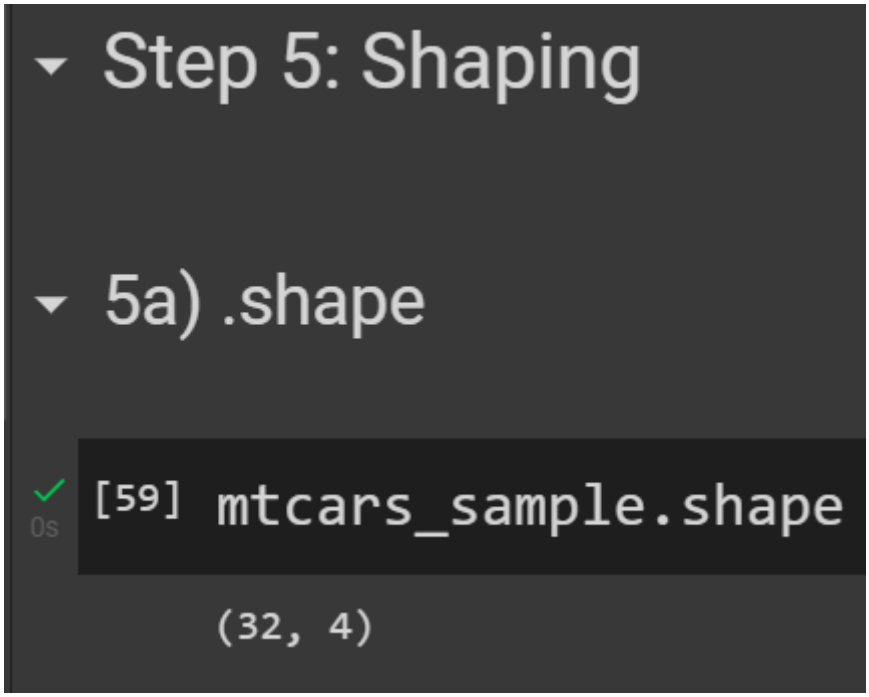
```
[57] #import in XLS

     mtcars_2 = pd.read_excel('cars.xlsx', sheet_name = 'cars')
```

Step 4: Subsetting the Mtcars to Mtcars_sample

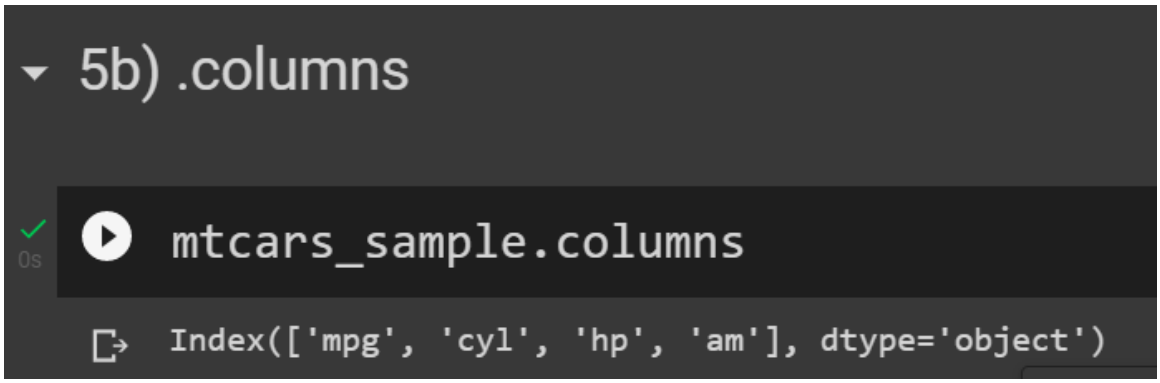
```
[58] mtcars_sample = pd.read_csv('https://www.alvinang.sg/s/mtcars.csv',
                                 index_col = 'car_names',
                                 usecols = ['car_names',
                                            'mpg', 'hp', 'cyl', 'am'])
```

V. STEP 5: SHAPING

A. .SHAPE



B. .COLUMNS



C. .INDEX

5c) .index

```
[61] mtcars_sample.index
```

```
Index(['Mazda RX4', 'Mazda RX4 Wag', 'Datsun 710', 'Hornet 4 Drive',
       'Hornet Sportabout', 'Valiant', 'Duster 360', 'Merc 240D', 'Merc 230',
       'Merc 280', 'Merc 280C', 'Merc 450SE', 'Merc 450SL', 'Merc 450SLC',
       'Cadillac Fleetwood', 'Lincoln Continental', 'Chrysler Imperial',
       'Fiat 128', 'Honda Civic', 'Toyota Corolla', 'Toyota Corona',
       'Dodge Challenger', 'AMC Javelin', 'Camaro Z28', 'Pontiac Firebird',
       'Fiat X1-9', 'Porsche 914-2', 'Lotus Europa', 'Ford Pantera L',
       'Ferrari Dino', 'Maserati Bora', 'Volvo 142E'],
      dtype='object', name='car_names')
```

D. .VALUES

5d) .values

```
[62] mtcars_sample['mpg'].values
```

```
array([21. , 21. , 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17.8,
       16.4, 17.3, 15.2, 10.4, 10.4, 14.7, 32.4, 30.4, 33.9, 21.5, 15.5,
       15.2, 13.3, 19.2, 27.3, 26. , 30.4, 15.8, 19.7, 15. , 21.4])
```

E. .VALUE_COUNTS()

5e) .value_counts()

```
[63] mtcars_sample['cyl'].value_counts()
```

```
8    14
4    11
6     7
Name: cyl, dtype: int64
```

A. .SORT_VALUES()

Step 6: Pivoting

6a) .sort_values()

```
mtcars_sample.sort_values(by='cyl', ascending = True)

#sort by ascending values of the CYL
```

car_names	mpg	cyl	hp	am
Volvo 142E	21.4	4	109	1
Datsun 710	22.8	4	93	1
Lotus Europa	30.4	4	113	1
Porsche 914-2	26.0	4	91	1

B. .GROUPBY().MEAN()

6b) .groupby().mean()

```
[87] mtcars_sample.groupby(['cyl']).mean()

     #groupby CYL, mean the values of all other columns
```

cyl	mpg	hp	am
4	26.663636	82.636364	0.727273
6	19.742857	122.285714	0.428571
8	15.100000	209.214286	0.142857

```
mtcars_sample.groupby(['cyl', 'am']).mean()

#groupby CYL & AM, mean the values of all other columns
```

cyl	am	mpg	hp
4	0	22.900000	84.666667
	1	28.075000	81.875000
6	0	19.125000	115.250000
	1	20.566667	131.666667
8	0	15.050000	194.166667
	1	15.400000	299.500000

```
[88] mtcars_sample.groupby(['cyl']).hp.mean()

     #groupby CYL, mean the values of hp
```

```
cyl
4     82.636364
6    122.285714
8    209.214286
Name: hp, dtype: float64
```

```
[93] #another way....
     mtcars_sample.groupby('cyl')['hp'].mean()

     #groupby CYL, mean the values of hp
```

```
cyl
4     82.636364
6    122.285714
8    209.214286
Name: hp, dtype: float64
```

```
[92] #another way....
     mtcars_sample[['cyl', 'hp']].groupby(['cyl']).mean()

     #groupby CYL, mean the values of hp
```

cyl	hp
4	82.636364
6	122.285714
8	209.214286

C. .GROUPBY().SUM()

6c) .groupby().sum()

```
[94] mtcars_sample.groupby(['cyl']).sum()

     #groupby CYL, sum the values of all other columns
```

cyl	mpg	hp	am
4	293.3	909	8
6	138.2	856	3
8	211.4	2929	2

D. .GROUPBY().AGG()

6d) .groupby().agg()

```
[95] mtcars_sample.groupby(['cyl']).agg(['mean','count'])

     #groupby CYL, aggregate the MEAN and COUNT of all other columns
```

	mpg		hp		am	
	mean	count	mean	count	mean	count
cyl						
4	26.663636	11	82.636364	11	0.727273	11
6	19.742857	7	122.285714	7	0.428571	7
8	15.100000	14	209.214286	14	0.142857	14

```
mtcars_sample.groupby(['cyl','am']).agg(['mean', 'count'])

#groupby CYL and AM, aggregate the MEAN and COUNT of all other columns
```

		mpg		hp	
		mean	count	mean	count
cyl	am				
4	0	22.900000	3	84.666667	3
	1	28.075000	8	81.875000	8
6	0	19.125000	4	115.250000	4
	1	20.566667	3	131.666667	3
8	0	15.050000	12	194.166667	12
	1	15.400000	2	299.500000	2

```
[99] mtcars_sample.groupby('cyl').hp.agg(['mean','median','max'])

     #groupby CYL, aggregate by the mean / median / max of the MPG
```

cyl	mean	median	max
4	82.636364	91.0	113
6	122.285714	110.0	175
8	209.214286	192.5	335

E. .GROUPBY().AGG(LAMBDA X:...)

6e) .groupby().agg(lambda x:...)

```
[96] mtcars_sample.groupby('cyl').agg(lambda x: max(x) - min(x))

    #groupby CYL, aggregate by the RANGE of all other columns
    #using Lambda
```

cyl	mpg	hp	am
4	12.5	61	1
6	3.6	70	1
8	8.8	185	1

F. .GROUPBY().DESCRIBE()

6f) .groupby().describe()

```
mtcars_sample.groupby('cyl').hp.describe()

#groupby CYL, describe the HP
```

cyl	count	mean	std	min	25%	50%	75%	max
4	11.0	82.636364	20.934530	52.0	65.50	91.0	96.00	113.0
6	7.0	122.285714	24.260491	105.0	110.00	110.0	123.00	175.0
8	14.0	209.214286	50.976886	150.0	176.25	192.5	241.25	335.0

G. .PIVOT()

6g) .pivot()

```
mtcars_sample.pivot(columns = 'cyl', values = 'hp')

#pivot by CYL, showing values of HP
```

cyl	4	6	8
car_names			
AMC Javelin	NaN	NaN	150.0
Cadillac Fleetwood	NaN	NaN	205.0
Camaro Z28	NaN	NaN	245.0
Chrysler Imperial	NaN	NaN	230.0
Datsun 710	93.0	NaN	NaN
Dodge Challenger	NaN	NaN	150.0
Duster 360	NaN	NaN	245.0

H. .PIVOT().MEAN()

6h) .pivot().mean()

```
[102] mtcars_sample.pivot(columns = 'cyl', values = 'hp').mean()

      #pivot by CYL, showing mean of HP
```

```
cyl
4     82.636364
6    122.285714
8    209.214286
dtype: float64
```

I. .PIVOT_TABLE()

6i) .pivot_table()

```
[103] mtcars_sample.pivot_table(index = 'cyl', columns = 'am',
                                values = 'hp', aggfunc = 'mean')

    #pivot by CYL and AM, showing the mean of HP
```

am	0	1
cyl		
4	84.666667	81.875000
6	115.250000	131.666667
8	194.166667	299.500000

A. USING >

Step 7: Filtering

7a) Using >

```
[78] #Filtering out Cyl > 4

     mtcars_sample[mtcars_sample['cyl'] > 4]
```

car_names	mpg	cyl	hp	am
Mazda RX4	21.0	6	110	1
Mazda RX4 Wag	21.0	6	110	1
Hornet 4 Drive	21.4	6	110	0
Hornet Sportabout	18.7	8	175	0

B. USING |

7b) Using |

```
#Filtering out MPG > 20 OR CYL < 6

mtcars_sample[(mtcars_sample['mpg'] > 20) | (mtcars_sample['cyl'] > 6)]
```

car_names	mpg	cyl	hp	am
Mazda RX4	21.0	6	110	1
Mazda RX4 Wag	21.0	6	110	1
Datsun 710	22.8	4	93	1
Hornet 4 Drive	21.4	6	110	0
Hornet Sportabout	18.7	8	175	0
Duster 360	14.3	8	245	0

C. USING ==

7c) Using ==

```
#Filtering AM == 1

mtcars_sample[mtcars_sample['am'] == 1]
```

car_names	mpg	cyl	hp	am
Mazda RX4	21.0	6	110	1
Mazda RX4 Wag	21.0	6	110	1
Datsun 710	22.8	4	93	1
Fiat 128	32.4	4	66	1
Honda Civic	30.4	4	52	1
Toyota Corolla	33.9	4	65	1

D. USING ISIN

7d) Using ISIN

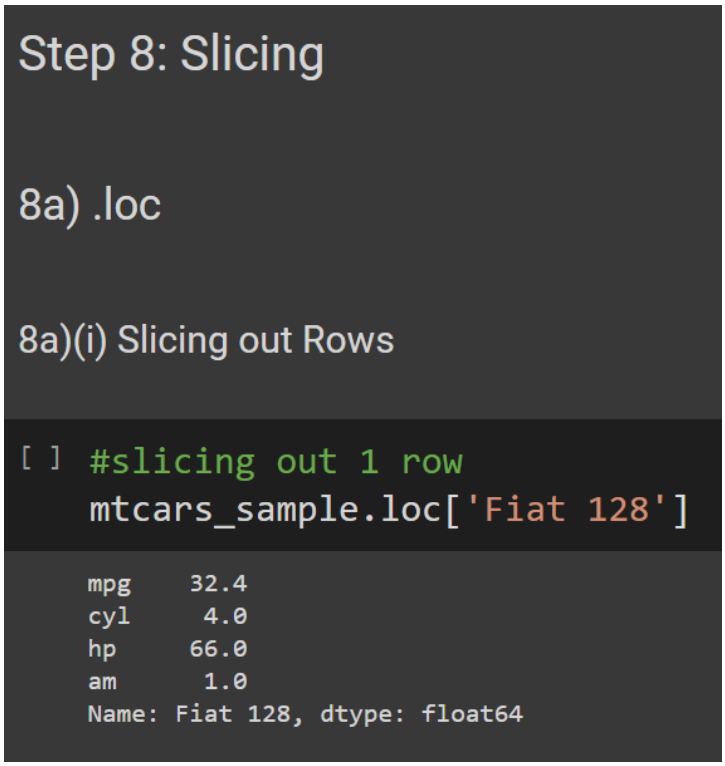
```
[ ] #Filtering out CYL == 6

    mtcars_sample[mtcars_sample['cyl'].isin([6])]
```

car_names	mpg	cyl	hp	am
Mazda RX4	21.0	6	110	1
Mazda RX4 Wag	21.0	6	110	1
Hornet 4 Drive	21.4	6	110	0
Valiant	18.1	6	105	0
Merc 280	19.2	6	123	0
Merc 280C	17.8	6	123	0
Ferrari Dino	19.7	6	175	1

VII. STEP 8: SLICING

A. .LOC

1. SLICING OUT ROWS




```
#slicing out 2 rows
mtcars_sample.loc[['Fiat 128', 'Lotus Europa']]
```

car_names	mpg	cyl	hp	am
Fiat 128	32.4	4	66	1
Lotus Europa	30.4	4	113	1

2. SLICING OUT ROWS & COLUMNS

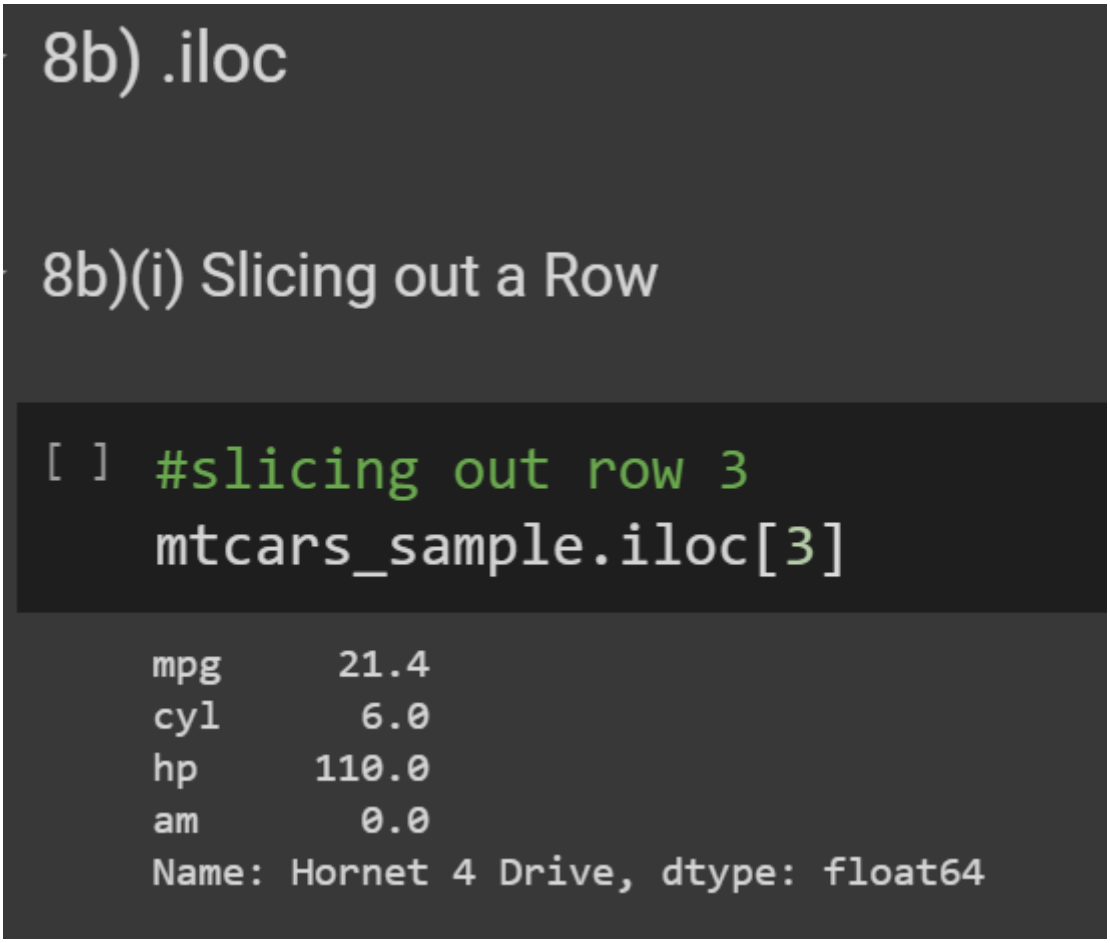
8a)(ii) Slicing out Rows & Columns

```
[ ] #slicing out 2 rows and columns mpg to
    mtcars_sample.loc[['Fiat 128', 'Lotus Europa'], 'mpg':'hp']
```

car_names	mpg	cyl	hp
Fiat 128	32.4	4	66
Lotus Europa	30.4	4	113

B. .ILOC

1. SLICING OUT A ROW



2. SLICING OUT ROWS

8b)(ii) Slicing out Rows

```
[ ] #slicing out rows 3 AND 5
    mtcars_sample.iloc[[3,5]]
```

car_names	mpg	cyl	hp	am
Hornet 4 Drive	21.4	6	110	0
Valiant	18.1	6	105	0

```
#slicing out rows 3 TO 5
mtcars_sample.iloc[3:6]
```

car_names	mpg	cyl	hp	am
Hornet 4 Drive	21.4	6	110	0
Hornet Sportabout	18.7	8	175	0
Valiant	18.1	6	105	0

```
#slicing out rows 1 to 3
mtcars_sample.iloc[:3]
```

car_names	mpg	cyl	hp	am
Mazda RX4	21.0	6	110	1
Mazda RX4 Wag	21.0	6	110	1
Datsun 710	22.8	4	93	1

3. SLICING OUT ROWS AND COLUMNS

8b)(iii) Slicing out Rows and Columns

```
[ ] #slicing out rows 1 to 2 AND columns 2 to 3
    mtcars_sample.iloc[1:3, 2:4]
```

car_names	hp	am
Mazda RX4 Wag	110	1
Datsun 710	93	1

C. C FOR C IN... SLICING OUT ROWS WITH PARTICULAR WORDS

1. TOYOTA

8c) C for C in....slicing out rows with particular words

8c)(i) Toyota

```
[ ] toyota = [c for c in mtcars_sample.index if 'Toyota' in c]
    toyota

    #slicing out rows with particular word
```

```
['Toyota Corolla', 'Toyota Corona']
```

```
toyota_cars = mtcars_sample.loc[toyota]
toyota_cars
```

car_names	mpg	cyl	hp	am
Toyota Corolla	33.9	4	65	1
Toyota Corona	21.5	4	97	0

2. MERC

8c)(ii) Merc

```
merc = [c for c in mtcars_sample.index if 'Merc' in c]

merc_cars = mtcars_sample.loc[merc]

merc_cars

#slicing out rows with particular word
```

car_names	mpg	cyl	hp	am
Merc 240D	24.4	4	62	0
Merc 230	22.8	4	95	0
Merc 280	19.2	6	123	0
Merc 280C	17.8	6	123	0
Merc 450SE	16.4	8	180	0
Merc 450SL	17.3	8	180	0
Merc 450SLC	15.2	8	180	0

Step 9: Concatenating

```
[ ] merc_toyota_cars = pd.concat([merc_cars, toyota_cars], axis = 0)
    merc_toyota_cars
```

car_names	mpg	cyl	hp	am
Merc 240D	24.4	4	62	0
Merc 230	22.8	4	95	0
Merc 280	19.2	6	123	0
Merc 280C	17.8	6	123	0
Merc 450SE	16.4	8	180	0
Merc 450SL	17.3	8	180	0
Merc 450SLC	15.2	8	180	0
Toyota Corolla	33.9	4	65	1
Toyota Corona	21.5	4	97	0

Step 10: Appending

```
toyota_merc_cars_2 = toyota_cars.append(merc_cars)
toyota_merc_cars_2
```

car_names	mpg	cyl	hp	am
Toyota Corolla	33.9	4	65	1
Toyota Corona	21.5	4	97	0
Merc 240D	24.4	4	62	0
Merc 230	22.8	4	95	0
Merc 280	19.2	6	123	0
Merc 280C	17.8	6	123	0
Merc 450SE	16.4	8	180	0
Merc 450SL	17.3	8	180	0
Merc 450SLC	15.2	8	180	0

ABOUT DR. ALVIN ANG

Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at www.AlvinAng.sg.