Cesar Acosta Ph.D.

Introduction to library Pandas

INTRODUCTION

Python libraries

- numpy
- pandas
- matplotlib
- statsmodels
- scikit-learn

pandas

pandas is *the* library for Data Analysis

It provides

- Data structures for Data Analysis

and modules for

- Data summarization
- Data visualization

pandas

Data Sets include variables that are

- numerical
- categorical (factors)

Variables are usually classified as

- response (target)
- predictor**s** (features)

pandas data structures

- pandas Series
- pandas DataFrames

pandas *Series*

- Represents a one-dimensional set of values (all of the same data type)
- It is used to represent a single variable
- An *index* is used to extract individual values from the *Series*

pandas *DataFrame*

- The *DataFrame* is a two-dimensional data set
- A column *label* is used to select columns from the DataFrame
- A row *index* is used to select rows from the DataFrame
- Each column is used for a variable
- All columns must have same number of entries

DataFrame cars93

	Manufacturer	Model	Type	Min.Price	Price	Max.Price
1	Acura	Integra	Small	12.9	15.9	18.8
2	Acura	Legend	Midsize	29.2	33.9	38.7
3	Audi	90	Compact	25.9	29.1	32.3
4	Audi	100	Midsize	30.8	37.7	44.6
5	BMW	535i	Midsize	23.7	30	36.2
6	Buick	Century	Midsize	14.2	15.7	17.3
7	Buick	LeSabre	Large	19.9	20.8	21.7
8	Buick	Roadmaster	Large	22.6	23.7	24.9
9	Buick	Riviera	Midsize	26.3	26.3	26.3
10	Cadillac	DeVille	Large	33	34.7	36.3
11	Cadillac	Seville	Midsize	37.5	40.1	42.7
12	Chevrolet	Cavalier	Compact	8.5	13.4	18.3
13	Chevrolet	Corsica	Compact	11.4	11.4	11.4
14	Chevrolet	Camaro	Sporty	13.4	15.1	16.8

pandas *DataFrame*

columns

index

	Manufacturer	Model	Type	Min.Price	Price	Max.Price
1	Acura	Integra	Small	12.9	15.9	18.8
2	Acura	Legend	Midsize	29.2	33.9	38.7
3	Audi	90	Compact	25.9	29.1	32.3
4	Audi	100	Midsize	30.8	37.7	44.6
5	BMW	535i	Midsize	23.7	30	36.2
6	Buick	Century	Midsize	14.2	15.7	17.3
7	Buick	LeSabre	Large	19.9	20.8	21.7
8	Buick	Roadmaster	Large	22.6	23.7	24.9
9	Buick	Riviera	Midsize	26.3	26.3	26.3
10	Cadillac	DeVille	Large	33	34.7	36.3
11	Cadillac	Seville	Midsize	37.5	40.1	42.7
12	Chevrolet	Cavalier	Compact	8.5	13.4	18.3
13	Chevrolet	Corsica	Compact	11.4	11.4	11.4
14	Chevrolet	Camaro	Sporty	13.4	15.1	16.8

values

pandas *DataFrame*

	Manufacturer	Model	Type	Min.Price	Price	Max.Price	I
1	Acura	Integra	Small	12.9	15.9	18.8	
2	Acura	Legend	Midsize	29.2	33.9	38.7	
3	Audi	90	Compact	25.9	29.1	32.3	
4	Audi	100	Midsize	30.8	37.7	44.6	
5	BMW	535i	Midsize	23.7	30	36.2	
6	Buick	Century	Midsize	14.2	15.7	17.3	
7	Buick	LeSabre	Large	19.9	20.8	21.7	
8	Buick	Roadmaster	Large	22.6	23.7	24.9	
9	Buick	Riviera	Midsize	26.3	26.3	26.3	
10	Cadillac	DeVille	Large	33	34.7	36.3	
11	Cadillac	Seville	Midsize	37.5	40.1	42.7	
12	Chevrolet	Cavalier	Compact	8.5	13.4	18.3	
13	Chevrolet	Corsica	Compact	11.4	11.4	11.4	
14	Chevrolet	Camaro	Sporty	13.4	15.1	16.8	

Python data types

Values in a column should be of the same **data type**

- int64
- float64
- datetime64
- object
- bool

slice

A *slice* is a reference to a portion of the pandas Data structure

Also called a *view.*

Modifying a slice, modifies the original data structure too

	Manufacturer	Model	Type	Min.Price	Price	Max.Price	MPG.city	MPG.highway
1	Acura	Integra	Small	12.9	15.9	18.8	25	31
2	Acura	Legend	Midsize	29.2	33.9	38.7	18	25
3	Audi	90	Compact	25.9	29.1	32.3	20	26
4	Audi	100	Midsize	30.8	37.7	44.6	19	26
5	BMW	535i	Midsize	23.7	30	36.2	22	30
6	Buick	Century	Midsize	14.2	15.7	17.3	22	31
7	Buick	LeSabre	Large	19.9	20.8	21.7	19	28
8	Buick	Roadmaster	Large	22.6	23.7	24.9	16	25
9	Buick	Riviera	Midsize	26.3	26.3	26.3	19	27
10	Cadillac	DeVille	Large	33	34.7	36.3	16	25
11	Cadillac	Seville	Midsize	37.5	40.1	42.7	16	25
12	Chevrolet	Cavalier	Compact	8.5	13.4	18.3	25	36
13	Chevrolet	Corsica	Compact	11.4	11.4	11.4	25	34
14	Chevrolet	Camaro	Sporty	13.4	15.1	16.8	19	28
15	Chevrolet	Lumina	Midsize	13.4	15.9	18.4	21	29
16	Chevrolet	Lumina_APV	Van	14.7	16.3	18	18	23
17	Chevrolet	Astro	Van	14.7	16.6	18.6	15	20
18	Chevrolet	Caprice	Large	18	18.8	19.6	17	26
19	Chevrolet	Corvette	Sporty	34.6	38	41.5	17	25
20	Chrylser	Concorde	Large	18.4	18.4	18.4	20	28
21	Chrysler	LeBaron	Compact	14.5	15.8	17.1	23	28

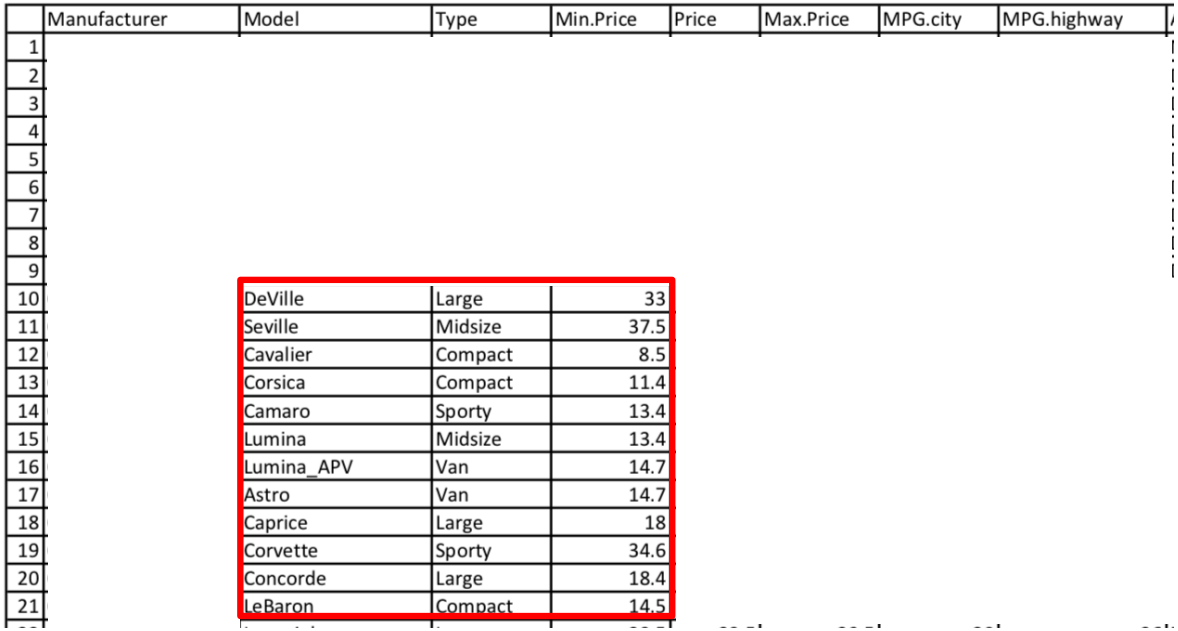
a slice

slice

A *slice* is a reference to a portion of the pandas Data structure

Also called a *view*

Modifying a slice, modifies the original data structure too



	Manufacturer	Model	Type	Min.Price	Price	Max.Price	MPG.city	MPG.highway
1								
2								
3								
4								
5								
6								
7								
8								
9								
10		DeVille	Large	33				
11		Seville	Midsize	37.5				
12		Cavalier	Compact	8.5				
13		Corsica	Compact	11.4				
14		Camaro	Sporty	13.4				
15		Lumina	Midsize	13.4				
16		Lumina_APV	Van	14.7				
17		Astro	Van	14.7				
18		Caprice	Large	18				
19		Corvette	Sporty	34.6				
20		Concorde	Large	18.4				
21		LeBaron	Compact	14.5				

Example
Cars93 dataset

A dataset of 93 cars

	Manufacturer	Model	Type	Min.Price	Price	Max.Price	MPG.city	MPG.highway	AirBags	DriveTrain	Cylinders	EngineSize	Horsepower	RPM	Rev.per.mile	Man.trans.av	Fuel.tank.cap	Passengers	Length	Wheelbase	Width	Turn.circle	Rear.seat.roo	Luggage.roor	Weight	Origin
1	Acura	Integra	Small	12.9	15.9	18.8	25	31	None	Front	4	1.8	140	6300	2890	Yes	13.2	5	177	102	68	37	26.5	11	2705	non-USA
2	Acura	Legend	Midsize	29.2	33.9	38.7	18	25	Driver & Passenger	Front	6	3.2	200	5500	2335	Yes	18	5	195	115	71	38	30	15	3560	non-USA
3	Audi	90	Compact	25.9	29.1	32.3	20	26	Driver only	Front	6	2.8	172	5500	2280	Yes	16.9	5	180	102	67	37	28	14	3375	non-USA
4	Audi	100	Midsize	30.8	37.7	44.6	19	26	Driver & Passenger	Front	6	2.8	172	5500	2535	Yes	21.1	6	193	106	70	37	31	17	3405	non-USA
5	BMW	535i	Midsize	23.7	30	36.2	22	30	Driver only	Rear	4	3.5	208	5700	2545	Yes	21.1	4	186	109	69	39	27	13	3640	non-USA
6	Buick	Century	Midsize	14.2	15.7	17.3	22	31	Driver only	Front	4	2.2	110	5200	2565	No	16.4	6	189	105	69	41	28	16	2880	USA
7	Buick	LeSabre	Large	19.9	20.8	21.7	19	28	Driver only	Front	6	3.8	170	4800	1570	No	18	6	200	111	74	42	30.5	17	3470	USA
8	Buick	Roadmaster	Large	22.6	23.7	24.9	16	25	Driver only	Rear	6	5.7	180	4000	1320	No	23	6	216	116	78	45	30.5	21	4105	USA
9	Buick	Riviera	Midsize	26.3	26.3	26.3	19	27	Driver only	Front	6	3.8	170	4800	1690	No	18.8	5	198	108	73	41	26.5	14	3495	USA
10	Cadillac	DeVille	Large	33	34.7	36.3	16	25	Driver only	Front	8	4.9	200	4100	1510	No	18	6	206	114	73	43	35	18	3620	USA
11	Cadillac	Seville	Midsize	37.5	40.1	42.7	16	25	Driver & Passenger	Front	8	4.6	295	5000	1985	No	20	5	204	111	74	44	31	14	3935	USA
12	Chevrolet	Cavalier	Compact	8.5	13.4	18.3	25	36	None	Front	4	2.2	110	5200	2380	Yes	15.2	5	182	101	66	38	25	13	2490	USA
13	Chevrolet	Corsica	Compact	11.4	11.4	11.4	25	34	Driver only	Front	4	2.2	110	5200	2665	Yes	15.6	5	184	103	68	39	26	14	2785	USA
14	Chevrolet	Camaro	Sporty	13.4	15.1	16.8	19	28	Driver & Passenger	Rear	6	3.4	160	4600	1805	Yes	15.5	4	193	101	74	43	25	13	3240	USA
15	Chevrolet	Lumina	Midsize	13.4	15.9	18.4	21	29	None	Front	4	2.2	110	5200	2595	No	16.5	6	198	108	71	40	28.5	16	3195	USA
16	Chevrolet	Lumina_APV	Van	14.7	16.3	18	18	23	None	Front	6	3.8	170	4800	1690	No	20	7	178	110	74	44	30.5	NA	3715	USA
17	Chevrolet	Astro	Van	14.7	16.6	18.6	15	20	None	4WD	6	4.3	165	4000	1790	No	27	8	194	111	78	42	33.5	NA	4025	USA
18	Chevrolet	Caprice	Large	18	18.8	19.6	17	26	Driver only	Rear	8	5	170	4200	1350	No	23	6	214	116	77	42	29.5	20	3910	USA
19	Chevrolet	Corvette	Sporty	34.6	38	41.5	17	25	Driver only	Rear	8	5.7	300	5000	1450	Yes	20	2	179	96	74	43	NA	NA	3380	USA
20	Chrylser	Concorde	Large	18.4	18.4	18.4	20	28	Driver & Passenger	Front	6	3.3	153	5300	1990	No	18	6	203	113	74	40	31	15	3515	USA
21	Chrysler	LeBaron	Compact	14.5	15.8	17.1	23	28	Driver & Passenger	Front	4	3	141	5000	2090	No	16	6	183	104	68	41	30.5	14	3085	USA
22	Chrysler	Imperial	Large	29.5	29.5	29.5	20	26	Driver only	Front	6	3.3	147	4800	1785	No	16	6	203	110	69	44	36	17	3570	USA
23	Dodge	Colt	Small	7.9	9.2	10.6	29	33	None	Front	4	1.5	92	6000	3285	Yes	13.2	5	174	98	66	32	26.5	11	2270	USA
24	Dodge	Shadow	Small	8.4	11.3	14.2	23	29	Driver only	Front	4	2.2	93	4800	2595	Yes	14	5	172	97	67	38	26.5	13	2670	USA
25	Dodge	Spirit	Compact	11.9	13.3	14.7	22	27	Driver only	Front	4	2.5	100	4800	2535	Yes	16	6	181	104	68	39	30.5	14	2970	USA
26	Dodge	Caravan	Van	13.6	19	24.4	17	21	Driver only	4WD	6	3	142	5000	1970	No	20	7	175	112	72	42	26.5	NA	3705	USA
27	Dodge	Dynasty	Midsize	14.8	15.6	16.4	21	27	Driver only	Front	4	2.5	100	4800	2465	No	16	6	192	105	69	42	30.5	16	3080	USA
28	Dodge	Stealth	Sporty	18.5	25.8	33.1	18	24	Driver only	4WD	6	3	300	6000	2120	Yes	19.8	4	180	97	72	40	20	11	3805	USA
29	Eagle	Summit	Small	7.9	12.2	16.5	29	33	None	Front	4	1.5	92	5000	2505	Yes	13.2	5	174	98	66	36	26.5	11	2295	USA
30	Eagle	Vision	Large	17.5	19.3	21.2	20	28	Driver & Passenger	Front	6	3.5	214	5800	1980	No	18	6	202	113	74	40	30	15	3490	USA
31	Ford	Festiva	Small	6.9	7.4	7.9	31	33	None	Front	4	1.3	63	5000	3150	Yes	10	4	141	90	63	33	26	12	1845	USA
32	Ford	Escort	Small	8.4	10.1	11.9	23	30	None	Front	4	1.8	127	6500	2410	Yes	13.2	5	171	98	67	36	28	12	2530	USA
33	Ford	Tempo	Compact	10.4	11.3	12.2	22	27	None	Front	4	2.3	96	4200	2805	Yes	15.9	5	177	100	68	39	27.5	13	2690	USA
34	Ford	Mustang	Sporty	10.8	15.9	21	22	29	Driver only	Rear	4	2.3	105	4600	2285	Yes	15.4	4	180	101	68	40	24	12	2850	USA
35	Ford	Probe	Sporty	12.8	14	15.2	24	30	Driver only	Front	4	2	115	5500	2340	Yes	15.5	4	179	103	70	38	23	18	2710	USA
36	Ford	Aerostar	Van	14.5	19.9	25.3	15	20	Driver only	4WD	6	3	145	4800	2080	Yes	21	7	176	119	72	45	30	NA	3735	USA
37	Ford	Taurus	Midsize	15.6	20.2	24.8	21	30	Driver only	Front	6	3	140	4800	1885	No	16	5	192	106	71	40	27.5	18	3325	USA
38	Ford	Crown_Victoria	Large	20.1	20.9	21.7	18	26	Driver only	Rear	8	4.6	190	4200	1415	No	20	6	212	114	78	43	30	21	3950	USA
39	Geo	Metro	Small	6.7	8.4	10	46	50	None	Front	3	1	55	5700	3755	Yes	10.6	4	151	93	63	34	27.5	10	1695	non-USA
40	Geo	Storm	Sporty	11.5	12.5	13.5	30	36	Driver only	Front	4	1.6	90	5400	3250	Yes	12.4	4	164	97	67	37	24.5	11	2475	non-USA
41	Honda	Prelude	Sporty	17	19.8	22.7	24	31	Driver & Passenger	Front	4	2.3	160	5800	2855	Yes	15.9	4	175	100	70	39	23.5	8	2865	non-USA
42	Honda	Civic	Small	8.4	12.1	15.8	42	46	Driver only	Front	4	1.5	102	5900	2650	Yes	11.9	4	173	103	67	36	28	12	2350	non-USA
43	Honda	Accord	Compact	13.8	17.5	21.2	24	31	Driver & Passenger	Front	4	2.2	140	5600	2610	Yes	17	4	185	107	67	41	28	14	3040	non-USA
44	Hyundai	Excel	Small	6.8	8	9.2	29	33	None	Front	4	1.5	81	5500	2710	Yes	11.9	5	168	94	63	35	26	11	2345	non-USA
45	Hyundai	Elantra	Small	9	10	11	22	29	None	Front	4	1.8	124	6000	2745	Yes	13.7	5	172	98	66	36	28	12	2620	non-USA
46	Hyundai	Scoupe	Sporty	9.1	10	11	26	34	None	Front	4	1.5	92	5550	2540	Yes	11.9	4	166	94	64	34	23.5	9	2285	non-USA
47	Hyundai	Sonata	Midsize	12.4	13.9	15.3	20	27	None	Front	4	2	128	6000	2335	Yes	17.2	5	184	104	69	41	31	14	2885	non-USA
48	Infiniti	Q45	Midsize	45.4	47.9	50.4	17	22	Driver only	Rear	8	4.5	278	6000	1955	No	22.5	5	200	113	72	42	29	15	4000	non-USA
49	Lexus	ES300	Midsize	27.5	28	28.4	18	24	Driver only	Front	6	3	185	5200	2325	Yes	18.5	5	188	103	70	40	27.5	14	3510	non-USA
50	Lexus	SC300	Midsize	34.7	35.2	35.6	18	23	Driver & Passenger	Rear	6	3	225	6000	2510	Yes	20.6	4	191	106	71	39	25	9	3515	non-USA
51	Lincoln	Continental	Midsize	33.3	34.3	35.3	17	26	Driver & Passenger	Front	6	3.8	160	4400	1835	No	18.4	6	205	109	73	42	30	19	3695	USA
52	Lincoln	Town_Car	Large	34.4	36.1	37.8	18	26	Driver & Passenger	Rear	8	4.6	210	4600	1840	No	20	6	219	117	77	45	31.5	22	4055	USA
53	Mazda	323	Small	7.4	8.3	9.1	29	37	None	Front	4	1.6	82	5000	2370	Yes	13.2	4	164	97	66	34	27	16	2325	non-USA
54	Mazda	Protege	Small	10.9	11.6	12.3	28	36	None	Front	4	1.8	103	5500	2220	Yes	14.5	5	172	98	66	36	26.5	13	2440	non-USA
55	Mazda	626	Compact	14.3	16.5	18.7	26	34	Driver only	Front	4	2.5	164	5600	2505	Yes	15.5	5	184	103	69	40	29.5	14	2970	non-USA
56	Mazda	MPV	Van	16.6	19.1	21.7	18	24	None	4WD	6	3	155	5000	2240	No	19.6	7	190	110	72	39	27.5	NA	3735	non-USA
57	Mazda	RX-7	Sporty	32.5	32.5	32.5	17	25	Driver only	Rear	rotary	1.3	255	6500	2325	Yes	20	2	169	96	69	37	NA	NA	2895	non-USA
58	Mercedes-Benz	190E	Compact	29	31.9	34.9	20	29	Driver only	Rear	4	2.3	130	5100	2425	Yes	14.5	5	175	105	67	34	26	12	2920	non-USA
59	Mercedes-Benz	300E	Midsize	43.8	61.9	80	19	25	Driver & Passenger	Rear	6	3.2	217	5500	2220	No	18.5	5	187	110	69	37	27	15	3525	non-USA
60	Mercury	Capri	Sporty	13.3	14.1	15	23	26	Driver only	Front	4	1.6	100	5750	2475	Yes	11.1	4	166	95	65	36	19	6	2450	USA

Data Manipulation Operations

- Find dataset size
- Summarize numerical vars
- Remove rows with NA
- Sort dataset
- Find categorical vars and their categories
- Summarize categorical vars
- Count rows by categories (cross tabulation)
- Construct pivot tables
- Select a subset of dataset
- Select rows by condition
- Combine categories
- Select rows at random
- Aggregate datasets
- Split a column into two or more columns

Data Manipulation Operations

Open libraries (dependencies)

```
import numpy as np
import pandas as pd
```

Read csv file

```
df = pd.read_csv('Cars93.csv')
```

Find its size

```
df.shape
```

```
(93, 27)
```

Data Manipulation Operations

- **Review the dataset**

```
df.shape
```

```
(93, 27)
```

```
# there are 93 rows and 27 columns
```

```
df.index
```

row identifiers

```
RangeIndex(start=0, stop=93, step=1)
```

```
df.columns
```

27 column names

```
Index(['Manufacturer', 'Model', 'Type', 'Min.Price', 'Price', 'Max.Price', 'MPG.city',
       'MPG.highway', 'AirBags', 'DriveTrain', 'Cylinders', 'EngineSize', 'Horsepower',
       'RPM', 'Rev.per.mile', 'Man.trans.avail', 'Fuel.tank.capacity', 'Passengers',
       'Length', 'Wheelbase', 'Width', 'Turn.circle', 'Rear.seat.room', 'Luggage.room',
       'Weight', 'Origin', 'Make'],
      dtype='object')
```

Data Manipulation Operations

```
df[:2]
```

show first two rows

	Manufacturer	Model	Type	Min.Price	Price	Max.Price	MPG.city	MPG.highway	AirBags
0	Acura	Integra	Small	12.9	15.9	18.8	25	31	None
1	Acura	Legend	Midsize	29.2	33.9	38.7	18	25	Driver & Passenger

...

2 rows × 27 columns

```
df[-2:]
```

show last two rows

	Manufacturer	Model	Type	Min.Price	Price	Max.Price	MPG.city	MPG.highway	AirBags
91	Volvo	240	Compact	21.8	22.7	23.5	21	28	Driver only
92	Volvo	850	Midsize	24.8	26.7	28.5	20	28	Driver & Passenger

...

2 rows × 27 columns

Data Manipulation Operations

```
df.head(2)
```

show first two rows

	Manufacturer	Model	Type	Min.Price	Price	Max.Price	MPG.city	MPG.highway	AirBags
0	Acura	Integra	Small	12.9	15.9	18.8	25	31	None
1	Acura	Legend	Midsize	29.2	33.9	38.7	18	25	Driver & Passenger

...

2 rows × 27 columns

```
df.tail(2)
```

show last two rows

	Manufacturer	Model	Type	Min.Price	Price	Max.Price	MPG.city	MPG.highway	AirBags
91	Volvo	240	Compact	21.8	22.7	23.5	21	28	Driver only
92	Volvo	850	Midsize	24.8	26.7	28.5	20	28	Driver & Passenger

...

2 rows × 27 columns

Data Manipulation Operations – select rows at random

```
df.sample(3)
```

	Manufacturer	Model	Type	Min.Price	Price	Max.Price	MPG.city	MPG.highway	AirBags	DriveTrain
8	Buick	Riviera	Midsize	26.3	26.3	26.3	19	27	Driver only	Front
56	Mazda	RX-7	Sporty	32.5	32.5	32.5	17	25	Driver only	Rear
92	Volvo	850	Midsize	24.8	26.7	28.5	20	28	Driver & Passenger	Front

...

3 rows × 27 columns

Data Manipulation Operations – select columns at random

```
df.sample(3,axis=1)
```

	Passengers	Max.Price	Rev.per.mile
0	5	18.8	2890
1	5	38.7	2335
2	5	32.3	2280
3	6	44.6	2535
4	4	36.2	2545
...	...	...	...
88	7	22.7	2915
89	5	22.4	2685
90	4	23.7	2385
91	5	23.5	2215
92	5	28.5	2310

93 rows × 3 columns

Data Manipulation Operations – First and last columns

```
df.columns
```

```
Index(['Manufacturer', 'Model', 'Type', 'Min.Price', 'Price', 'Max.Price', 'MPG.city',
       'MPG.highway', 'AirBags', 'DriveTrain', 'Cylinders', 'EngineSize', 'Horsepower',
       'RPM', 'Rev.per.mile', 'Man.trans.avail', 'Fuel.tank.capacity', 'Passengers',
       'Length', 'Wheelbase', 'Width', 'Turn.circle', 'Rear.seat.room', 'Luggage.room',
       'Weight', 'Origin', 'Make'],
      dtype='object')
```

```
df['Manufacturer']
```

```
0           Acura
1           Acura
2            Audi
3            Audi
4             BMW
          ...
88     Volkswagen
89     Volkswagen
90     Volkswagen
91          Volvo
92          Volvo
```

```
df['Make']
```

```
0           Acura Integra
1            Acura Legend
2                 Audi 90
3                Audi 100
4                BMW 535i
              ...
88     Volkswagen Eurovan
89      Volkswagen Passat
90     Volkswagen Corrado
91              Volvo 240
92              Volvo 850
```

Data Manipulation Operations – First and last columns

```
df.columns
```

```
Index(['Manufacturer', 'Model', 'Type', 'Min.Price', 'Price', 'Max.Price', 'MPG.city',
       'MPG.highway', 'AirBags', 'DriveTrain', 'Cylinders', 'EngineSize', 'Horsepower',
       'RPM', 'Rev.per.mile', 'Man.trans.avail', 'Fuel.tank.capacity', 'Passengers',
       'Length', 'Wheelbase', 'Width', 'Turn.circle', 'Rear.seat.room', 'Luggage.room',
       'Weight', 'Origin', 'Make'],
      dtype='object')
```

```
df.Manufacturer
```

```
0           Acura
1           Acura
2            Audi
3            Audi
4             BMW
          ...
88     Volkswagen
89     Volkswagen
90     Volkswagen
91          Volvo
92          Volvo
```

```
df.Make
```

```
0           Acura Integra
1            Acura Legend
2                 Audi 90
3                Audi 100
4                BMW 535i
              ...
88    Volkswagen Eurovan
89     Volkswagen Passat
90    Volkswagen Corrado
91              Volvo 240
92              Volvo 850
```

DataFrame - show column data types

Use .dtypes to identify categorical and numerical columns

```
df.dtypes
```

```
Manufacturer          object
Model                 object
Type                  object
Min.Price            float64
Price                float64
Max.Price            float64
MPG.city               int64
MPG.highway            int64
AirBags               object
DriveTrain            object
Cylinders             object
EngineSize           float64
Horsepower             int64
RPM                    int64
Rev.per.mile           int64
Man.trans.avail       object
Fuel.tank.capacity   float64
Passengers             int64
Length                 int64
Wheelbase              int64
Width                  int64
```

Numerical columns

vs

Categorical columns

Data Manipulation Operations – drop columns

```
# drop the following columns (keeping 13 columns only)

list1 = [3,5,11,12,13,14,15,16,17,19,21,22,23,26]
df.drop(df.columns[list1],axis = 1, inplace = True)
df.shape
```

```
(93, 13)
```

```
df[:5]
```

	Manufacturer	Model	Type	Price	MPG.city	MPG.highway	AirBags
0	Acura	Integra	Small	15.9	25	31	None
1	Acura	Legend	Midsize	33.9	18	25	Driver & Passenger
2	Audi	90	Compact	29.1	20	26	Driver only
3	Audi	100	Midsize	37.7	19	26	Driver & Passenger
4	BMW	535i	Midsize	30.0	22	30	Driver only

...

Data Manipulation Operations – categories of one column

```
df.dtypes
```

```
Manufacturer        object
Model               object
Type                object
```

```
pd.unique(df['Type'])
```

```
array(['Small', 'Midsize', 'Compact', 'Large', 'Sporty', 'Van'],
      dtype=object)
```

```
# find number of rows from each category
```

```
pd.value_counts(df['Type'])
```

```
Midsize    22
Small      21
Compact    16
Sporty     14
Large      11
Van         9
Name: Type, dtype: int64
```

```
df['Type'].value_counts()
```

```
Midsize    22
Small      21
Compact    16
Sporty     14
Large      11
Van         9
Name: Type, dtype: int64
```

Data Manipulation Operations

Converting a variable.

Numerical

to

categorical

```
df['Passengers'] = df['Passengers'].astype(object)
```

```
df.dtypes
```

Manufacturer	object
Model	object
Type	object
Min.Price	float64
Price	float64
Max.Price	float64
MPG.city	int64
MPG.highway	int64
AirBags	object
DriveTrain	object
Cylinders	object
EngineSize	float64
Horsepower	int64
RPM	int64
Rev.per.mile	int64
Man.trans.avail	object
Fuel.tank.capacity	float64
Passengers	int64
Length	int64
Wheelbase	int64
Width	int64

Data Manipulation Operations

```
# select columns with categorical variables
df2 = df.loc[:, df.dtypes == object]
df2[:5]
```

	Manufacturer	Model	Type	AirBags	DriveTrain	Cylinders	I
0	Acura	Integra	Small	None	Front	4	
1	Acura	Legend	Midsize	Driver & Passenger	Front	6	
2	Audi	90	Compact	Driver only	Front	6	
3	Audi	100	Midsize	Driver & Passenger	Front	6	
4	BMW	535i	Midsize	Driver only	Rear	4	

Data Manipulation – excluding some columns

```
# Manufacturer and Model have too many categories
# so I will exclude them
```

```
# to exclude      column Model
df22 = df2.loc[:, df2.columns != 'Model']
```

```
# to exclude more columns use
df22 = df2.drop(['Manufacturer','Model'], axis=1)
```

Data Manipulation – categories in all columns

```
for column in df22:
    print(column)
    print(df22[column].value_counts())
```

```
Type
Midsize     22
Small       21
Compact     16
Sporty      14
Large       11
Van          9
Name: Type, dtype: int64
AirBags
Driver only            43
None                   34
Driver & Passenger     16
Name: AirBags, dtype: int64
DriveTrain
Front     67
Rear      16
4WD       10
Name: DriveTrain, dtype: int64
Cylinders
4          49
6          31
8           7
3           3
5           2
rotary      1
Name: Cylinders, dtype: int64
Origin
USA        48
non-USA    45
Name: Origin, dtype: int64
```

Data Manipulation – categories in all columns

```
for column in df22:
    print(column)
    print(df22[column].value_counts())
```

```
Type
Midsize     22
Small       21
Compact     16
Sporty      14
Large       11
Van          9
Name: Type, dtype: int64
AirBags
Driver only            43
None                   34
Driver & Passenger     16
Name: AirBags, dtype: int64
DriveTrain
Front     67
Rear      16
4WD       10
Name: DriveTrain, dtype: int64
Cylinders
4          49
6          31
8           7
3           3
5           2
rotary      1
Name: Cylinders, dtype: int64
Origin
USA        48
non-USA    45
Name: Origin, dtype: int64
```

Data Manipulation – categories in all columns

```
for column in df22:
    print('\n',column)
    print(df22[column].value_counts())
```

```
 Type
Midsize    22
Small      21
Compact    16
Sporty     14
Large      11
Van         9
Name: Type, dtype: int64

 AirBags
Driver only           43
None                  34
Driver & Passenger    16
Name: AirBags, dtype: int64

 DriveTrain
Front    67
Rear     16
4WD      10
Name: DriveTrain, dtype: int64

 Cylinders
4          49
6          31
8           7
3           3
5           2
rotary      1
Name: Cylinders, dtype: int64

 Origin
USA        48
non-USA    45
Name: Origin, dtype: int64
```

Data Manipulation Operations

```
# select numerical columns
```

```
df2 = df.loc[:, df.dtypes != object]
df2[:5]
```

	Min.Price	Price	Max.Price	MPG.city	MPG.highway	EngineSize	Horsepower	RPM
0	12.9	15.9	18.8	25	31	1.8	140	6300
1	29.2	33.9	38.7	18	25	3.2	200	5500
2	25.9	29.1	32.3	20	26	2.8	172	5500
3	30.8	37.7	44.6	19	26	2.8	172	5500
4	23.7	30.0	36.2	22	30	3.5	208	5700

Summary

- Numerical columns
- Categorical columns

Example – Summary Statistics

A statistical summary of all **numerical variables**

```
df.describe()
```

	Price	MPG.city	MPG.highway	Length	Width	Weight
count	93.000000	93.000000	93.000000	93.000000	93.000000	93.000000
mean	19.509677	22.365591	29.086022	183.204301	69.376344	3072.903226
std	9.659430	5.619812	5.331726	14.602382	3.778986	589.896510
min	7.400000	15.000000	20.000000	141.000000	60.000000	1695.000000
25%	12.200000	18.000000	26.000000	174.000000	67.000000	2620.000000
50%	17.700000	21.000000	28.000000	183.000000	69.000000	3040.000000
75%	23.300000	25.000000	31.000000	192.000000	72.000000	3525.000000
max	61.900000	46.000000	50.000000	219.000000	78.000000	4105.000000

Example – Summary Statistics rounded

```
np.around(df.describe(),2)
```

	Price	MPG.city	MPG.highway	Length	Width	Weight
count	93.00	93.00	93.00	93.0	93.00	93.0
mean	19.51	22.37	29.09	183.2	69.38	3072.9
std	9.66	5.62	5.33	14.6	3.78	589.9
min	7.40	15.00	20.00	141.0	60.00	1695.0
25%	12.20	18.00	26.00	174.0	67.00	2620.0
50%	17.70	21.00	28.00	183.0	69.00	3040.0
75%	23.30	25.00	31.00	192.0	72.00	3525.0
max	61.90	46.00	50.00	219.0	78.00	4105.0

Example – Summary for column *Price* only

```
df['Price'].describe()
```

```
count     93.000000
mean      19.509677
std        9.659430
min        7.400000
25%       12.200000
50%       17.700000
75%       23.300000
max       61.900000
Name: Price, dtype: float64
```

Example – Summary for column *Price* only

```
df['Price'].describe()
```

```
count     93.000000
mean      19.509677
std        9.659430
min        7.400000
25%       12.200000
50%       17.700000
75%       23.300000
max       61.900000
Name: Price, dtype: float64
```

percentiles

Example – Summary for categorical columns

Select columns with Categorical variables

```
df2 = df.loc[:, df.dtypes == object]
df2[:5]
```

	Manufacturer	Model	Type	AirBags	DriveTrain	Cylinders	Origin
0	Acura	Integra	Small	None	Front	4	non-USA
1	Acura	Legend	Midsize	Driver & Passenger	Front	6	non-USA
2	Audi	90	Compact	Driver only	Front	6	non-USA
3	Audi	100	Midsize	Driver & Passenger	Front	6	non-USA
4	BMW	535i	Midsize	Driver only	Rear	4	non-USA

Example – Summary for categorical columns

```
df2[:5]
```

columns

	Manufacturer	Model	Type	AirBags	DriveTrain	Cylinders	Origin
0	Acura	Integra	Small	None	Front	4	non-USA
1	Acura	Legend	Midsize	Driver & Passenger	Front	6	non-USA
2	Audi	90	Compact	Driver only	Front	6	non-USA
3	Audi	100	Midsize	Driver & Passenger	Front	6	non-USA

```
df2.describe()
```

summary

	Manufacturer	Model	Type	AirBags	DriveTrain	Cylinders	Origin
count	93	93	93	93	93	93	93
unique	32	93	6	3	3	6	2
top	Chevrolet	Sonata	Midsize	Driver only	Front	4	USA
freq	8	1	22	43	67	49	48

Example – Summary for categorical columns

```
df2[:5]
```

	Manufacturer	Model	Type	AirBags	DriveTrain	Cylinders	Origin
0	Acura	Integra	Small	None	Front	4	non-USA
1	Acura	Legend	Midsize	Driver & Passenger	Front	6	non-USA
2	Audi	90	Compact	Driver only	Front	6	non-USA
3	Audi	100	Midsize	Driver & Passenger	Front	6	non-USA

```
df2.describe()
```

	Manufacturer	Model	Type	AirBags	DriveTrain	Cylinders	Origin	
count	93	93	93	93	93	93	93	
unique	32	93	6	3	3	6	2	categories
top	Chevrolet	Sonata	Midsize	Driver only	Front	4	USA	most frequent category
freq	8	1	22	43	67	49	48	rows in most freq category

Summary for numerical vs. Summary for categorical variables

summary for a numerical variable

```
df['Price'].describe()
```

```
count    93.000000
mean     19.509677
std       9.659430
min       7.400000
25%      12.200000
50%      17.700000
75%      23.300000
max      61.900000
```

summary for a categorical variable

```
df2['Manufacturer'].describe()
```

```
count        93
unique       32
top       Ford
freq          8
```

Subsetting (selecting portions of the dataset)

Selecting rows and columns

- by names
- by numbers
- Selecting rows by condition

Example – select single column

```
df['Price']
```

```
0      15.9
1      33.9
2      29.1
3      37.7
4      30.0
       ...
88     19.7
89     20.0
90     23.3
91     22.7
92     26.7
Name: Price, Length: 93, dtype: float64
```

```
type(df.Price)
```

```
pandas.core.series.Series
```

Example – single column (getting Series or DataFrame)

```
df['Price']
```

```
0      15.9
1      33.9
2      29.1
3      37.7
4      30.0
       ...
88     19.7
89     20.0
90     23.3
91     22.7
92     26.7
Name: Price, Length: 93, dtype: float64
```

```
type(df.Price)
```

```
pandas.core.series.Series
```

```
pd.DataFrame(df['Price'])
```

	Price
0	15.9
1	33.9
2	29.1
3	37.7
4	30.0
...	...
88	19.7
89	20.0
90	23.3
91	22.7
92	26.7

93 rows × 1 columns

Example – select two columns

```
list2 = ['Model','Price']
df4 = df[list2]
df4
```

	Model	Price
0	Integra	15.9
1	Legend	33.9
2	90	29.1
3	100	37.7
4	535i	30.0
...	...	...
88	Eurovan	19.7
89	Passat	20.0
90	Corrado	23.3
91	240	22.7
92	850	26.7

93 rows × 2 columns

Example – select two columns

```
list2 = ['Model','Price']
df4 = df[list2]
df4
```

	Model	Price
0	Integra	15.9
1	Legend	33.9
2	90	29.1
3	100	37.7
4	535i	30.0
...	...	...
88	Eurovan	19.7
89	Passat	20.0
90	Corrado	23.3
91	240	22.7
92	850	26.7

93 rows × 2 columns

```
df4 = df[['Model','Price']]
df4
```

	Model	Price
0	Integra	15.9
1	Legend	33.9
2	90	29.1
3	100	37.7
4	535i	30.0
...	...	...
88	Eurovan	19.7
89	Passat	20.0
90	Corrado	23.3
91	240	22.7
92	850	26.7

93 rows × 2 columns

Cesar Acosta Ph.D.

Subsetting with dataframe property **.iloc**

Example – **iloc** to select rows/cols by numbering

```
df.shape
```

```
(93, 13)
```

```
df[:5]
```

	0 Manufacturer	1 Model	2 Type	3 Price	4 MPG.city	5 MPG.highway	6 AirBags
0	Acura	Integra	Small	15.9	25	31	None
1	Acura	Legend	Midsize	33.9	18	25	Driver & Passenger
2	Audi	90	Compact	29.1	20	26	Driver only
3	Audi	100	Midsize	37.7	19	26	Driver & Passenger
4	BMW	535i	Midsize	30.0	22	30	Driver only

7 DriveTrain	8 Cylinders	9 Length	10 Width	11 Weight	12 Origin
Front	4	177	68	2705	non-USA
Front	6	195	71	3560	non-USA

Example – select columns 1 and 3

```
df[:5]
```

	Manufacturer	1 Model	Type	3 Price	MPG.city	MPG.highway	AirBags
0	Acura	Integra	Small	15.9	25	31	None
1	Acura	Legend	Midsize	33.9	18	25	Driver & Passenger
2	Audi	90	Compact	29.1	20	26	Driver only
3	Audi	100	Midsize	37.7	19	26	Driver & Passenger
4	BMW	535i	Midsize	30.0	22	30	Driver only

```
df.iloc[:,[1,3]]
```

	Model	Price
0	Integra	15.9
1	Legend	33.9
2	90	29.1
3	100	37.7
4	535i	30.0
...	...	...
88	Eurovan	19.7
89	Passat	20.0
90	Corrado	23.3
91	240	22.7
92	850	26.7

Example – select rows 0 and 2, and, columns 3 and 6

```
df[:5]
```

		Manufacturer	Model	Type	3 Price	MPG.city	MPG.highway	6 AirBags
0	0	Acura	Integra	Small	15.9	25	31	None
	1	Acura	Legend	Midsize	33.9	18	25	Driver & Passenger
2	2	Audi	90	Compact	29.1	20	26	Driver only
	3	Audi	100	Midsize	37.7	19	26	Driver & Passenger
	4	BMW	535i	Midsize	30.0	22	30	Driver only

```
df.iloc[[0,2],[3,6]]
```

	Price	AirBags
0	15.9	None
2	29.1	Driver only

Example – select rows 0 to 2 and columns 3 to 6

```
df[:5]
```

				3	4	5	6
	Manufacturer	Model	Type	Price	MPG.city	MPG.highway	AirBags
0	0 Acura	Integra	Small	15.9	25	31	None
1	1 Acura	Legend	Midsize	33.9	18	25	Driver & Passenger
2	2 Audi	90	Compact	29.1	20	26	Driver only
	3 Audi	100	Midsize	37.7	19	26	Driver & Passenger
	4 BMW	535i	Midsize	30.0	22	30	Driver only

```
df.iloc[0:2,3:6]
```

	Price	MPG.city	MPG.highway
0	15.9	25	31
1	33.9	18	25

Example

```
# Select columns by names and rows by number
```

```
df[['Model','Price']].iloc[0:5]
```

	Model	Price
0	Integra	15.9
1	Legend	33.9
2	90	29.1
3	100	37.7
4	535i	30.0

Cesar Acosta Ph.D.

Subsetting with dataframe property **.loc**

Example – **loc** to select cols by names

```
df.loc[0:3,'MPG.city':'Width']
```

	MPG.city	MPG.highway	AirBags	DriveTrain	Cylinders	Length	Width
0	25	31	None	Front	4	177	68
1	18	25	Driver & Passenger	Front	6	195	71
2	20	26	Driver only	Front	6	180	67
3	19	26	Driver & Passenger	Front	6	193	70

Example – **loc** to select cols by dtype

```
# Select the Columns with categorical variables only
```

```
df2 = df.loc[:, df.dtypes == object]
df2[:5]
```

	Manufacturer	Model	Type	AirBags	DriveTrain	Cylinders	Origin
0	Acura	Integra	Small	None	Front	4	non-USA
1	Acura	Legend	Midsize	Driver & Passenger	Front	6	non-USA
2	Audi	90	Compact	Driver only	Front	6	non-USA
3	Audi	100	Midsize	Driver & Passenger	Front	6	non-USA
4	BMW	535i	Midsize	Driver only	Rear	4	non-USA

Example – **loc** to select cols by dtype

```
# Select numerical Columns only
```

```
df3 = df.loc[:, df.dtypes != object]
df3[:5]
```

	Price	MPG.city	MPG.highway	Length	Width	Weight
0	15.9	25	31	177	68	2705
1	33.9	18	25	195	71	3560
2	29.1	20	26	180	67	3375
3	37.7	19	26	193	70	3405
4	30.0	22	30	186	69	3640

Select rows by condition

Example – select rows by condition

```
# Find most expensive car (do not sort)
```

```
df[df.Price == df.Price.max()]
```

	Manufacturer	Model	Type	Price	MPG.city	MPG.highway	AirBags	DriveTrain
58	Mercedes-Benz	300E	Midsize	61.9	19	25	Driver & Passenger	Rear

Example – select rows by condition

```
# How many cars with 3 cylinders?
```

```
pd.value_counts(df.Cylinders)
```

```
4          49
6          31
8           7
3           3
5           2
rotary      1
Name: Cylinders, dtype: int64
```

...

Example – select rows by condition

```
# How many cars with 3 cylinders?
```

```
pd.value_counts(df.Cylinders)
```

```
4          49
6          31
8           7
3           3
5           2
rotary      1
Name: Cylinders, dtype: int64
```

```
df[df.Cylinders == '3']
```

show the cars

	Manufacturer	Model	Type	Price	MPG.city	MPG.highway	AirBags	DriveTrain
38	Geo	Metro	Small	8.4	46	50	None	Front
79	Subaru	Justy	Small	8.4	33	37	None	4WD
82	Suzuki	Swift	Small	8.6	39	43	None	Front

...

Example – select rows by condition

```
# Find all cars with Price > 40
```

```
df[df.Price > 40]
```

	Manufacturer	Model	Type	Price	MPG.city	MPG.highway	AirBags	DriveTrain
10	Cadillac	Seville	Midsize	40.1	16	25	Driver & Passenger	Front
47	Infiniti	Q45	Midsize	47.9	17	22	Driver only	Rear
58	Mercedes-Benz	300E	Midsize	61.9	19	25	Driver & Passenger	Rear

...

Example – select rows by condition

```
# Find all cars with Price > 40
```

```
df[df.Price > 40]
```

	Manufacturer	Model	Type	Price	MPG.city	MPG.highway	AirBags	DriveTrain
10	Cadillac	Seville	Midsize	40.1	16	25	Driver & Passenger	Front
47	Infiniti	Q45	Midsize	47.9	17	22	Driver only	Rear
58	Mercedes-Benz	300E	Midsize	61.9	19	25	Driver & Passenger	Rear

...

All columns are displayed

What if we want to display selected columns, only?

Example – select rows by condition, some columns only

```
df.loc[df.Price > 40,['Model','Price']]
```

	Model	Price
10	Seville	40.1
47	Q45	47.9
58	300E	61.9

.loc allows to choose rows/cols by their name

.iloc allows to choose rows/cols by their number

.loc and **iloc** are not needed to choose rows by condition

Example – Average Price of cars with Price > 40

```
# Find all cars with Price > 40
```

```
df[df.Price > 40]
```

	Manufacturer	Model	Type	Price	MPG.city	MPG.highway	AirBags	DriveTrain
10	Cadillac	Seville	Midsize	40.1	16	25	Driver & Passenger	Front
47	Infiniti	Q45	Midsize	47.9	17	22	Driver only	Rear
58	Mercedes-Benz	300E	Midsize	61.9	19	25	Driver & Passenger	Rear

Average is 49.96

Example

```
# Average Price of all cars with Price > 40
```

```
df.Price[df.Price > 40].mean()
```

```
49.96666666666667
```

Example

```
# Average Price of all cars with Price > 40
```

```
df.Price[df.Price > 40].mean()
```

```
49.96666666666667
```

```
# Average Length of all cars with Price > 40
```

```
df.Length[df.Price > 40].mean()
```

```
197.0
```

Example – select rows by multiple conditions (**and**)

```
df[(df.Price < 10) & (df['MPG.city'] > 30)]
```

	Manufacturer	Model	Type	Price	MPG.city	MPG.highway	AirBags	DriveTrain	Cylinders
30	Ford	Festiva	Small	7.4	31	33	None	Front	4
38	Geo	Metro	Small	8.4	46	50	None	Front	3
72	Pontiac	LeMans	Small	9.0	31	41	None	Front	4
79	Subaru	Justy	Small	8.4	33	37	None	4WD	3
82	Suzuki	Swift	Small	8.6	39	43	None	Front	3
83	Toyota	Tercel	Small	9.8	32	37	Driver only	Front	4

Example – select rows by multiple conditions (**or**)

```
df[(df.Price < 10) | (df['MPG.city']>30)]
```

	Manufacturer	Model	Type	Price	MPG.city	MPG.highway	AirBags	DriveTrain	Cylinders
22	Dodge	Colt	Small	9.2	29	33	None	Front	4
30	Ford	Festiva	Small	7.4	31	33	None	Front	4
38	Geo	Metro	Small	8.4	46	50	None	Front	3
41	Honda	Civic	Small	12.1	42	46	Driver only	Front	4
43	Hyundai	Excel	Small	8.0	29	33	None	Front	4
52	Mazda	323	Small	8.3	29	37	None	Front	4
72	Pontiac	LeMans	Small	9.0	31	41	None	Front	4
79	Subaru	Justy	Small	8.4	33	37	None	4WD	3
82	Suzuki	Swift	Small	8.6	39	43	None	Front	3
83	Toyota	Tercel	Small	9.8	32	37	Driver only	Front	4
87	Volkswagen	Fox	Small	9.1	25	33	None	Front	4

Sorting

Example – Sort by column *Price* (ascending)

```
df.sort_values(by='Price')[:9]
```

	Manufacturer	Model	Type	Price	MPG.city	MPG.highway	AirBags	DriveTrain	Cylinders
30	Ford	Festiva	Small	7.4	31	33	None	Front	4
43	Hyundai	Excel	Small	8.0	29	33	None	Front	4
52	Mazda	323	Small	8.3	29	37	None	Front	4
38	Geo	Metro	Small	8.4	46	50	None	Front	3
79	Subaru	Justy	Small	8.4	33	37	None	4WD	3
82	Suzuki	Swift	Small	8.6	39	43	None	Front	3
72	Pontiac	LeMans	Small	9.0	31	41	None	Front	4
87	Volkswagen	Fox	Small	9.1	25	33	None	Front	4
22	Dodge	Colt	Small	9.2	29	33	None	Front	4

Example – Sort by column *Price* (descending)

```
df.sort_values(by='Price',ascending=False)[:6]
```

	Manufacturer	Model	Type	Price	MPG.city	MPG.highway	AirBags	DriveTrain
58	Mercedes-Benz	300E	Midsize	61.9	19	25	Driver & Passenger	Rear
47	Infiniti	Q45	Midsize	47.9	17	22	Driver only	Rear
10	Cadillac	Seville	Midsize	40.1	16	25	Driver & Passenger	Front
3	Audi	100	Midsize	37.7	19	26	Driver & Passenger	Front
51	Lincoln	Town_Car	Large	36.1	18	26	Driver & Passenger	Rear
49	Lexus	SC300	Midsize	35.2	18	23	Driver & Passenger	Rear

Example – Sort (ascending and descending)

```
df.sort_values(by=['Price','MPG.city'],ascending=[True,False])[:6]
```

	Manufacturer	Model	Type	Price	MPG.city	MPG.highway	AirBags	DriveTrain	Cylinders
30	Ford	Festiva	Small	7.4	31	33	None	Front	4
43	Hyundai	Excel	Small	8.0	29	33	None	Front	4
52	Mazda	323	Small	8.3	29	37	None	Front	4
38	Geo	Metro	Small	8.4	46	50	None	Front	3
79	Subaru	Justy	Small	8.4	33	37	None	4WD	3
82	Suzuki	Swift	Small	8.6	39	43	None	Front	3

Cross Tabulation

for categorical columns only

Example – Cross tabulation

How many rows are in each category

- value_counts( )
- crosstab( )

Example – Cross tabulation for one categorical

```
# number of cars by DriveTrain

pd.value_counts(df.DriveTrain)
```

```
Front    63
Rear     14
4WD       5
Name: DriveTrain, dtype: int64
```

This is a called a
one-way classification table

Example – Cross tabulation for two categoricals

```
# number of cars by DriveTrain and Airbags
```

```
pd.crosstab(df.DriveTrain,df.AirBags)
```

AirBags DriveTrain	Driver & Passenger	Driver only	None
4WD	0	2	3
Front	11	28	24
Rear	5	8	1

This is a called a
two-way classification table

Example – Cross tabulation with **row/cols totals**

```
pd.crosstab(df.DriveTrain,df.AirBags,margins=True)
```

AirBags DriveTrain	Driver & Passenger	Driver only	None	All
4WD	0	5	5	10
Front	11	28	28	67
Rear	5	10	1	16
All	16	43	34	93

Example – Cross tabulation for **one** categorical

```
# number of cars by Type
```

```
pd.value_counts(df.Type)
```

```
Midsize     22
Small       21
Compact     16
Sporty      14
Large       11
Van          9
Name: Type, dtype: int64
```

This is a one-way table

Example – Cross tabulation for **two** categoricals

```
# number of cars by Type and Origin
```

```
pd.crosstab(df.Type,df.Origin)
```

Origin	USA	non-USA
Type		
Compact	7	9
Large	11	0
Midsize	10	12
Small	7	14
Sporty	8	6
Van	5	4

This is a two-way table

Example – Cross tabulation for two categoricals

```
# number of cars by Type and Origin
```

```
pd.crosstab(df.Type,df.Origin,margins = True)
```

Origin Type	USA	non-USA	All
Compact	7	9	16
Large	11	0	11
Midsize	10	12	22
Small	7	14	21
Sporty	8	6	14
Van	5	4	9
All	48	45	93

Example – Cross tabulation for two categoricals

```
# number of cars by Type and Origin
```

```
pd.crosstab(df.Type,df.Origin,margins = True)
```

Origin	USA	non-USA
Type		
Compact	7	9
Large	11	0
Midsize	10	12
Small	7	14
Sporty	8	6
Van	5	4
All	48	45

Find proportions
with respect to column totals

Example – Cross tabulation for proportions

```
# Distribution of Origin across Type (% across cols)

pd.crosstab(df.Type,df.Origin,normalize = 'columns')
```

Origin	USA	non-USA
Type		
Compact	0.145833	0.200000
Large	0.229167	0.000000
Midsize	0.208333	0.266667
Small	0.145833	0.311111
Sporty	0.166667	0.133333
Van	0.104167	0.088889
	1	1

Find proportions
with respect to **column** totals

Example – Cross tabulation for proportions

```
# Distribution of Type across Origin (% across rows)

pd.crosstab(df.Type,df.Origin,normalize = 'index')
```

Origin	USA	non-USA	
Type			
Compact	0.437500	0.562500	1
Large	1.000000	0.000000	1
Midsize	0.454545	0.545455	1
Small	0.333333	0.666667	1
Sporty	0.571429	0.428571	1
Van	0.555556	0.444444	1

Find proportions
with respect to **row** totals

Cesar Acosta Ph.D.

Pivot Tables

Pivot Table

Pivot Table is used

to summarize a numerical attribute

by the categories of

one or more categorical variables

Pivot Table

Pivot Table is used

to summarize a numerical attribute

(mean, median, range, variance, etc.)

by the categories of

one or more categorical variables

Numerical methods for DataFrames

Method	Returns
`count()`	The number of non-null entries
`cumprod()`	The cumulative product over an axis
`cumsum()`	The cumulative sum over an axis
`max()`	The maximum of the entries
`mean()`	The average of the entries
`median()`	The median of the entries
`min()`	The minimum of the entries
`mode()`	The most common element(s)
`prod()`	The product of the elements
`sum()`	The sum of the elements
`var()`	The variance of the elements

Pivot Table methods

- pivot_table( )
- groupby( )

Introduction – Price of Ford cars

```
# Use .loc to Select rows by condition, columns by name

df.loc[df.Manufacturer == 'Ford',['Price']]
```

	Price
30	7.4
31	10.1
32	11.3
33	15.9
34	14.0
35	19.9
36	20.2

There are seven Ford cars

Introduction – Average Price of Ford cars

```
# Use .loc to Select rows by condition, columns by name

df.loc[df.Manufacturer == 'Ford',['Price']]
```

	Price
30	7.4
31	10.1
32	11.3
33	15.9
34	14.0
35	19.9
36	20.2

What is the average price of Ford cars?

```
# average price of Ford cars

df.loc[df.Manufacturer == 'Ford',['Price']].mean()
```

```
Price    14.9625
```

Introduction – Average Price of Ford cars

```
# Use .loc to Select rows by condition, columns by name

df.loc[df.Manufacturer == 'Ford',['Price']]
```

	Price
30	7.4
31	10.1
32	11.3
33	15.9
34	14.0
35	19.9
36	20.2

What is the average price of Ford cars?

```
# average price of Ford cars

df.loc[df.Manufacturer == 'Ford',['Price']].mean()
```

```
Price    14.9625
```

What is the average price of the other car brands?

Example – average price by Manufacturer

```
df.pivot_table(values = 'Price',
               index = 'Manufacturer')
```

Manufacturer	Price
Acura	24.900000
Audi	33.400000
BMW	30.000000
Buick	21.625000
Cadillac	37.400000
Chevrolet	18.187500
Chrylser	18.400000
Chrysler	22.650000
Dodge	15.700000
Eagle	15.750000
Ford	14.962500
Geo	10.450000
Honda	16.466667
Hyundai	10.475000
Infiniti	47.900000
Lexus	31.600000
Lincoln	35.200000
Mazda	17.600000
Mercedes-Benz	46.900000
Mercury	14.500000
Mitsubishi	18.200000
Nissan	17.025000
Oldsmobile	17.500000
Plymouth	14.400000
Pontiac	16.140000
Saab	28.700000
Saturn	11.100000
Subaru	12.933333
Suzuki	8.600000
Toyota	17.275000
Volkswagen	18.025000
Volvo	24.700000

Example – largest average price by Manufacturer

.

store the pivot table in a new dataframe,

then find the largest average price

```
df9 = df.pivot_table(values = 'Price',
                     index = 'Manufacturer')
df9[df9.Price == df9.Price.max()]
```

	Price
Manufacturer	
Infiniti	47.9

Example – Average Price and MPG by Manufacturer

```
df.pivot_table(values = ['Price','MPG.city'],index = 'Manufacturer').\
                                sort_values('Price',ascending = False).head()
```

Manufacturer	MPG.city	Price
Infiniti	17.0	47.9
Mercedes-Benz	19.5	46.9
Cadillac	16.0	37.4
Lincoln	17.5	35.2
Audi	19.5	33.4

sorted descending by Price

This is a
one-way table

two-row labeling makes
head() to show five rows only

Example – Two-way pivot table

Find the car's median price for each **Type** and Number of **Airbags**

AirBags	Driver & Passenger	Driver only	None
Type			
Compact	16.65	19.50	13.40
Large	21.85	20.90	NaN
Midsize	35.20	21.50	15.40
Small	NaN	11.30	9.15
Sporty	17.70	17.15	14.40
Van	NaN	19.90	19.10

Median prices

How to get this table?

Example – Two-way pivot table

```
df.pivot_table('Price',index = 'Type', columns = 'AirBags',
               aggfunc = np.median)
```

AirBags	Driver & Passenger	Driver only	None
Type			
Compact	16.65	19.50	13.40
Large	21.85	20.90	NaN
Midsize	35.20	21.50	15.40
Small	NaN	11.30	9.15
Sporty	17.70	17.15	14.40
Van	NaN	19.90	19.10

Cesar Acosta Ph.D.

Pivot Tables with groupby( )

Example – groupby Type

```
# Find average price and average MPG.city, by Type

df.groupby('Type')[['Price','MPG.city']].agg('mean')
```

Type	Price	MPG.city
Compact	18.212500	22.687500
Large	24.300000	18.363636
Midsize	27.218182	19.545455
Small	10.166667	29.857143
Sporty	19.392857	21.785714
Van	19.100000	17.000000

Example – groupby Type and DriveTrain

```
df.groupby(['Type','DriveTrain'])\
                        [['Price','MPG.city']].agg('mean')
```

		Price	MPG.city
Type	**DriveTrain**		
Compact	**4WD**	19.500000	23.000000
	Front	16.715385	23.000000
	Rear	27.300000	20.500000
Large	**Front**	23.971429	19.000000
	Rear	24.875000	17.250000
Midsize	**Front**	24.052941	19.705882
	Rear	37.980000	19.000000
Small	**4WD**	9.650000	29.000000
	Front	10.221053	29.947368

...

Example – groupby Type and DriveTrain

```
df.groupby(['Type','DriveTrain'])\
                        [['Price','MPG.city']].agg('mean')
```

		Price	MPG.city
Type	**DriveTrain**		
Compact	**4WD**	19.500000	23.000000
	Front	16.715385	23.000000
	Rear	27.300000	20.500000
Large	**Front**	23.971429	19.000000
	Rear	24.875000	17.250000
Midsize	**Front**	24.052941	19.705882
	Rear	37.980000	19.000000
Small	**4WD**	9.650000	29.000000
	Front	10.221053	29.947368

...

Example – groupby Type and DriveTrain

```
df.groupby(['Type','DriveTrain'],as_index = False)\
                        [['Price','MPG.city']].agg('mean')
```

	Type	DriveTrain	Price	MPG.city
0	Compact	4WD	19.500000	23.000000
1	Compact	Front	16.715385	23.000000
2	Compact	Rear	27.300000	20.500000
3	Large	Front	23.971429	19.000000
4	Large	Rear	24.875000	17.250000
5	Midsize	Front	24.052941	19.705882
6	Midsize	Rear	37.980000	19.000000
7	Small	4WD	9.650000	29.000000
8	Small	Front	10.221053	29.947368

Example – numerical columns in dataframe df

```
df[:3]
```

	Manufacturer	Model	Type	Price	MPG.city	MPG.highway
0	Acura	Integra	Small	15.9	25	31
1	Acura	Legend	Midsize	33.9	18	25
2	Audi	90	Compact	29.1	20	26

	AirBags	DriveTrain	Cylinders	Length	Width	Weight	Origin
	None	Front	4	177	68	2705	non-USA
	Driver & Passenger	Front	6	195	71	3560	non-USA
	Driver only	Front	6	180	67	3375	non-USA

numerical columns

Example – Display average of *all* cols **by *Type***

do not specify the column names

```
df.groupby('Type').agg('mean')
```

Type	Price	MPG.city	MPG.highway	Length	Width	Weight
Compact	18.212500	22.687500	29.875000	182.125000	67.250000	2918.125000
Large	24.300000	18.363636	26.727273	204.818182	74.727273	3695.454545
Midsize	27.218182	19.545455	26.727273	192.545455	70.636364	3400.000000
Small	10.166667	29.857143	35.476190	167.190476	65.285714	2312.857143
Sporty	19.392857	21.785714	28.785714	175.214286	69.285714	2899.642857
Van	19.100000	17.000000	21.888889	185.666667	73.222222	3830.555556

groupby( )
with a
user-defined function

Example – create df3 with numerical cols only

```
df3 = df.loc[:, df.dtypes != object]
df3[:5]
```

	Price	MPG.city	MPG.highway	Length	Width	Weight
0	15.9	25	31	177	68	2705
1	33.9	18	25	195	71	3560
2	29.1	20	26	180	67	3375
3	37.7	19	26	193	70	3405
4	30.0	22	30	186	69	3640

Example – Create *range* function

```
df3 = df.loc[:, df.dtypes != object]
df3[:5]
```

	Price	MPG.city	MPG.highway	Length	Width	Weight
0	15.9	25	31	177	68	2705
1	33.9	18	25	195	71	3560
2	29.1	20	26	180	67	3375
3	37.7	19	26	193	70	3405
4	30.0	22	30	186	69	3640

```
def range(array):
    return array.max() - array.min()
```

Example – Apply *range* function on all cars in df3

```
df3 = df.loc[:, df.dtypes != object]
df3[:5]
```

	Price	MPG.city	MPG.highway	Length	Width	Weight
0	15.9	25	31	177	68	2705
1	33.9	18	25	195	71	3560
2	29.1	20	26	180	67	3375
3	37.7	19	26	193	70	3405
4	30.0	22	30	186	69	3640

find range of all columns

```
def range(array):
    return array.max() - array.min()
```

```
range(df3)
```

```
Price          54.5
MPG.city       31.0
MPG.highway    30.0
Length         78.0
Width          18.0
Weight       2410.0
dtype: float64
```

Example – Apply *range* function

```
df3 = df.loc[:, df.dtypes != object]
df3[:5]
```

	Price	MPG.city	MPG.highway	Length	Width	Weight
0	15.9	25	31	177	68	2705
1	33.9	18	25	195	71	3560
2	29.1	20	26	180	67	3375
3	37.7	19	26	193	70	3405
4	30.0	22	30	186	69	3640

```
def range(array):
    return array.max() - array.min()
```

```
range(df3)
```

```
Price          54.5
MPG.city       31.0
MPG.highway    30.0
Length         78.0
Width          18.0
Weight       2410.0
dtype: float64
```

Now find the range by **Type**

Example – Find the range by **Type**

```
df.groupby('Type').agg([range])
```

Type	Price range	MPG.city range	MPG.highway range	Cylinders range	Length range	Width range	Weight range
Compact	20.8	6	10	2.0	15	3	885
Large	17.7	4	3	2.0	42	9	635
Midsize	48.0	7	9	4.0	21	5	1120
Small	8.5	24	21	1.0	36	8	1010
Sporty	28.0	13	12	8.0	37	11	1520
Van	6.4	3	4	2.0	19	7	395

Example – Find the mean and range by **Type**

```
df.groupby('Type').agg([np.mean,range])
```

	Price		MPG.city		MPG.highway		Length	
	mean	range	mean	range	mean	range	mean	range
Type								
Compact	18.212500	20.8	22.687500	6	29.875000	10	182.125000	15
Large	24.300000	17.7	18.363636	4	26.727273	3	204.818182	42
Midsize	27.218182	48.0	19.545455	7	26.727273	9	192.545455	21
Small	10.166667	8.5	29.857143	24	35.476190	21	167.190476	36
Sporty	19.392857	28.0	21.785714	13	28.785714	12	175.214286	37
Van	19.100000	6.4	17.000000	3	21.888889	4	185.666667	19

Example – Find the mean and range by **Type**

```
df.groupby('Type').agg([np.mean,range])
```

Type	Price		MPG.city		MPG.highway		Length	
	mean	range	mean	range	mean	range	mean	range
Compact	18.212500	20.8	22.687500	6	29.875000	10	182.125000	15
Large	24.300000	17.7	18.363636	4	26.727273	3	204.818182	42
Midsize	27.218182	48.0	19.545455	7	26.727273	9	192.545455	21
Small	10.166667	8.5	29.857143	24	35.476190	21	167.190476	36
Sporty	19.392857	28.0	21.785714	13	28.785714	12	175.214286	37
Van	19.100000	6.4	17.000000	3	21.888889	4	185.666667	19