BASICS OF PANDAS

SOME BASICS FUNCTIONS OF DATAFRAME
HOW TO ADD ROWS AND COLUMNS INTO THE DATAFRAME
SORTING DATAFRAME
HANDLING DATE & TIME FIELDS
STATISTICAL FUNCTIONS IN PANDAS
PANDAS INDEXING
DATA GROUPING
AGGREGATE FOR MULTIPLE MATHEMATICAL FUNCTIONS
DATA VISUALIZATION
ADDING NEW DATA

- MERGE, JOIN, CONCATENATE, APPEND

Importing pandas

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

Loading Data using pandas

In [2]:
```
df = pd.read_csv('commodity_price.csv', low_memory=False)
```

In [3]:
```
print(pd.read_csv.__doc__) # Can view docs of a particular methods/attribute
```

```
Read a comma-separated values (csv) file into DataFrame.

Also supports optionally iterating or breaking of the file
into chunks.

Additional help can be found in the online docs for
`IO Tools <https://pandas.pydata.org/pandas-docs/stable/user_guide/io.html>`_.

Parameters
----------
filepath_or_buffer : str, path object or file-like object
    Any valid string path is acceptable. The string could be a URL. Valid
    URL schemes include http, ftp, s3, gs, and file. For file URLs, a host is
    expected. A local file could be: file://localhost/path/to/table.csv.

    If you want to pass in a path object, pandas accepts any ``os.PathLike``.

    By file-like object, we refer to objects with a ``read()`` method, such as
```

In [4]:
```
df.shape
```

Out[4]:
```
(78651, 13)
```

In [5]:
```
df.head(10)
```

Out[5]:

	s_no	state	district	market	pincode	country	commodity	variety	arrival_date	min_price	max_price	modal_price	update_date
0	0	Karnataka	Belgaum	Athani	100001	India	Bull	Bull	26-01-2020	20000.0	40000.0	40000.0	NaN
1	1	Karnataka	Belgaum	Athani	100002	India	Bull	Bull	16-02-2020	10000.0	12000.0	11000.0	NaN
2	2	Karnataka	Belgaum	Athani	100003	India	Bull	Bull	01-03-2020	32000.0	40000.0	35000.0	NaN
3	3	Karnataka	Belgaum	Ramdurga	100004	India	Bull	Bull	05-01-2020	20000.0	25000.0	20000.0	NaN
4	4	Karnataka	Belgaum	Ramdurga	100005	India	BULL	Bull	12-01-2020	20000.0	35000.0	25000.0	NaN
5	5	Karnatak	Belgaum	Ramdurga	100006	India	BULL	Bull	19-01-2020	25000.0	35000.0	30000.0	NaN
6	6	Karnataka	Belgaum	Ramdurga	100007	India	BULL	Bulll	26-01-2020	20000.0	35000.0	25000.0	NaN
7	7	Karnataka	Belgaum	Ramdurga	100008	India	BULL	Bull	23-02-2020	25000.0	25000.0	25000.0	NaN
8	8	Karnataka	Belgaum	Ramdurga	100009	India	BULL	Bull	08-03-2020	25000.0	25000.0	25000.0	NaN
9	9	Karnataka	Belgaum	Ramdurga	100010	India	BULL	Bull	15-03-2020	25000.0	25000.0	25000.0	NaN

In [6]:
```
df.tail(10)
```

Out[6]:

	s_no	state	district	market	pincode	country	commodity	variety	arrival_date	min_price	max_price	modal_price	update_date
78641	78593	West Bengal	Murshidabad	Jangipur	178594	India	Turmeric	Finger	21-12-2022	9460.0	9575.0	9525.0	NaN
78642	78594	West Bengal	Murshidabad	Jangipur	178595	India	Turmeric	Finger	22-12-2022	9460.0	9575.0	9525.0	NaN
78643	78595	West Bengal	Murshidabad	Jangipur	178596	India	Turmeric	Finger	23-12-2022	9450.0	9560.0	9515.0	NaN
78644	78596	West Bengal	Murshidabad	Jangipur	178597	India	Turmeric	Finger	24-12-2022	9450.0	9560.0	9515.0	NaN
78645	78597	West Bengal	Murshidabad	Jangipur	178598	India	Turmeric	Finger	25-12-2022	9440.0	9550.0	9500.0	NaN
78646	78598	West Bengal	Murshidabad	Jangipur	178599	India	Turmeric	Finger	26-12-2022	9440.0	9550.0	9500.0	NaN
78647	78599	West Bengal	Murshidabad	Jangipur	178600	India	Turmeric	Finger	27-12-2022	9435.0	9550.0	9490.0	NaN
78648	78600	West Bengal	Murshidabad	Jangipur	178601	India	Turmeric	Finger	28-12-2022	9410.0	9530.0	9475.0	NaN
78649	78601	West Bengal	Murshidabad	Jangipur	178602	India	Turmeric	Finger	29-12-2022	9425.0	9550.0	9490.0	NaN
78650	78602	West Bengal	Murshidabad	Jangipur	178603	India	Turmeric	Finger	30-12-2022	9425.0	9550.0	9490.0	NaN

In [7]:
```
df.describe()
```

Out[7]:

	s_no	pincode	min_price	max_price	modal_price
count	78651.000000	78651.000000	78483.000000	78467.000000	78651.000000
mean	39324.978030	139325.978030	6578.085401	8112.236460	7360.387507
std	22704.694333	22704.694333	16594.528933	19494.764562	17951.236404
min	0.000000	100001.000000	2.000000	0.000000	2.000000
25%	19662.500000	119663.500000	3535.000000	4870.000000	4200.000000
50%	39325.000000	139326.000000	5229.000000	7000.000000	6000.000000
75%	58987.500000	158988.500000	7600.000000	8850.000000	8210.000000
max	78602.000000	178603.000000	1000000.000000	1000000.000000	1000000.000000

Some basic functions of DataFrames

In [8]:
```
df['state']
```

Out[8]:
```
0          Karnataka
1          Karnataka
2          Karnataka
3          Karnataka
4          Karnataka
             ...
78646    West Bengal
78647    West Bengal
78648    West Bengal
78649    West Bengal
78650    West Bengal
Name: state, Length: 78651, dtype: object
```

In [9]:
```
df[['state', 'district']]
```

Out[9]:

	state	district
0	Karnataka	Belgaum
1	Karnataka	Belgaum
2	Karnataka	Belgaum
3	Karnataka	Belgaum
4	Karnataka	Belgaum
...	...	...
78646	West Bengal	Murshidabad
78647	West Bengal	Murshidabad
78648	West Bengal	Murshidabad
78649	West Bengal	Murshidabad
78650	West Bengal	Murshidabad

78651 rows × 2 columns

In [10]:
```
df[['state', 'district']].head(10)
```

Out[10]:

	state	district
0	Karnataka	Belgaum
1	Karnataka	Belgaum
2	Karnataka	Belgaum
3	Karnataka	Belgaum
4	Karnataka	Belgaum
5	Karnatak	Belgaum
6	Karnataka	Belgaum
7	Karnataka	Belgaum
8	Karnataka	Belgaum
9	Karnataka	Belgaum

In [11]:
```
df[['state', 'district']].tail(10)
```

Out[11]:

	state	district
78641	West Bengal	Murshidabad
78642	West Bengal	Murshidabad
78643	West Bengal	Murshidabad
78644	West Bengal	Murshidabad
78645	West Bengal	Murshidabad
78646	West Bengal	Murshidabad
78647	West Bengal	Murshidabad
78648	West Bengal	Murshidabad
78649	West Bengal	Murshidabad
78650	West Bengal	Murshidabad

In [12]:
```
df[df['state']=='Tamil Nadu']
```

Out[12]:

	s_no	state	district	market	pincode	country	commodity	variety	arrival_date	min_price	max_price	modal_price	update_date
3519	3519	Tamil Nadu	Ariyalur	Andimadom	103520	India	Copra	Ball	04-11-2022	7600.0	7800.0	7700.0	NaN
3520	3520	Tamil Nadu	Coimbatore	Anaimalai	103521	India	Copra	other	01-01-2022	8940.0	9050.0	9000.0	NaN
3521	3521	Tamil Nadu	Coimbatore	Anaimalai	103522	India	Copra	other	06-01-2022	8950.0	9230.0	9090.0	NaN
3522	3522	Tamil Nadu	Coimbatore	Anaimalai	103523	India	Copra	other	12-01-2022	8840.0	9025.0	8950.0	NaN
3523	3523	Tamil Nadu	Coimbatore	Anaimalai	103524	India	Copra	other	13-01-2022	8840.0	9025.0	8950.0	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
76302	76302	Tamil Nadu	Vellore	Vellore	176303	India	Turmeric	Other	22-09-2022	2020.0	2020.0	2020.0	NaN
76303	76303	Tamil Nadu	Vellore	Vellore	176304	India	Turmeric	Other	26-09-2022	6687.0	6687.0	6687.0	NaN
76304	76304	Tamil Nadu	Vellore	Vellore	176305	India	Turmeric	Other	17-10-2022	2212.0	2212.0	2212.0	NaN
76305	76305	Tamil Nadu	Vellore	Vellore	176306	India	Turmeric	Other	30-11-2022	4088.0	4088.0	4088.0	NaN
76306	76306	Tamil Nadu	Vellore	Vellore	176307	India	Turmeric	Other	08-12-2022	3620.0	3620.0	3620.0	NaN

10516 rows × 13 columns

In [13]:
```
df[(df['state']=='Tamil Nadu') & (df['min_price'] > 7000)]
```

Out[13]:

	s_no	state	district	market	pincode	country	commodity	variety	arrival_date	min_price	max_price	modal_price	update_date
3519	3519	Tamil Nadu	Ariyalur	Andimadom	103520	India	Copra	Ball	04-11-2022	7600.0	7800.0	7700.0	NaN
3520	3520	Tamil Nadu	Coimbatore	Anaimalai	103521	India	Copra	other	01-01-2022	8940.0	9050.0	9000.0	NaN
3521	3521	Tamil Nadu	Coimbatore	Anaimalai	103522	India	Copra	other	06-01-2022	8950.0	9230.0	9090.0	NaN
3522	3522	Tamil Nadu	Coimbatore	Anaimalai	103523	India	Copra	other	12-01-2022	8840.0	9025.0	8950.0	NaN
3523	3523	Tamil Nadu	Coimbatore	Anaimalai	103524	India	Copra	other	13-01-2022	8840.0	9025.0	8950.0	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
76287	76287	Tamil Nadu	Vellore	Vellore	176288	India	Turmeric	Other	10-01-2022	7200.0	7200.0	7200.0	NaN
76290	76290	Tamil Nadu	Vellore	Vellore	176291	India	Turmeric	Other	20-04-2022	13010.0	14010.0	14000.0	NaN
76293	76293	Tamil Nadu	Vellore	Vellore	176294	India	Turmeric	Other	20-06-2022	8876.0	8876.0	8876.0	NaN
76300	76300	Tamil Nadu	Vellore	Vellore	176301	India	Turmeric	Other	12-09-2022	7086.0	7086.0	7086.0	NaN
76301	76301	Tamil Nadu	Vellore	Vellore	176302	India	Turmeric	Other	19-09-2022	7086.0	7086.0	7086.0	NaN

4160 rows × 13 columns

How to add rows and coloumns into the DataFrame

Add column

In [14]:
```
column = [10] * 78651
```

In [15]:
```
column
```

```
 10,
 10,
 10,
 10,
 10,
 10,
 10,
 10,
 10,
 10,
 10,
 10,
 10,
 10,
 10,
 10,
 10,
 10,
 10,
```

In [16]:
```
df['column'] = column
```

In [17]:
```
df.head(10)
```

Out[17]:

	s_no	state	district	market	pincode	country	commodity	variety	arrival_date	min_price	max_price	modal_price	update_date	colu
0	0	Karnataka	Belgaum	Athani	100001	India	Bull	Bull	26-01-2020	20000.0	40000.0	40000.0	NaN	
1	1	Karnataka	Belgaum	Athani	100002	India	Bull	Bull	16-02-2020	10000.0	12000.0	11000.0	NaN	
2	2	Karnataka	Belgaum	Athani	100003	India	Bull	Bull	01-03-2020	32000.0	40000.0	35000.0	NaN	
3	3	Karnataka	Belgaum	Ramdurga	100004	India	Bull	Bull	05-01-2020	20000.0	25000.0	20000.0	NaN	
4	4	Karnataka	Belgaum	Ramdurga	100005	India	BULL	Bull	12-01-2020	20000.0	35000.0	25000.0	NaN	
5	5	Karnatak	Belgaum	Ramdurga	100006	India	BULL	Bull	19-01-2020	25000.0	35000.0	30000.0	NaN	
6	6	Karnataka	Belgaum	Ramdurga	100007	India	BULL	Bulll	26-01-2020	20000.0	35000.0	25000.0	NaN	
7	7	Karnataka	Belgaum	Ramdurga	100008	India	BULL	Bull	23-02-2020	25000.0	25000.0	25000.0	NaN	
8	8	Karnataka	Belgaum	Ramdurga	100009	India	BULL	Bull	08-03-2020	25000.0	25000.0	25000.0	NaN	
9	9	Karnataka	Belgaum	Ramdurga	100010	India	BULL	Bull	15-03-2020	25000.0	25000.0	25000.0	NaN	

In [18]:
```
df.tail(10)
```

Out[18]:

	s_no	state	district	market	pincode	country	commodity	variety	arrival_date	min_price	max_price	modal_price	update_date
78641	78593	West Bengal	Murshidabad	Jangipur	178594	India	Turmeric	Finger	21-12-2022	9460.0	9575.0	9525.0	NaN
78642	78594	West Bengal	Murshidabad	Jangipur	178595	India	Turmeric	Finger	22-12-2022	9460.0	9575.0	9525.0	NaN
78643	78595	West Bengal	Murshidabad	Jangipur	178596	India	Turmeric	Finger	23-12-2022	9450.0	9560.0	9515.0	NaN
78644	78596	West Bengal	Murshidabad	Jangipur	178597	India	Turmeric	Finger	24-12-2022	9450.0	9560.0	9515.0	NaN
78645	78597	West Bengal	Murshidabad	Jangipur	178598	India	Turmeric	Finger	25-12-2022	9440.0	9550.0	9500.0	NaN
78646	78598	West Bengal	Murshidabad	Jangipur	178599	India	Turmeric	Finger	26-12-2022	9440.0	9550.0	9500.0	NaN
78647	78599	West Bengal	Murshidabad	Jangipur	178600	India	Turmeric	Finger	27-12-2022	9435.0	9550.0	9490.0	NaN
78648	78600	West Bengal	Murshidabad	Jangipur	178601	India	Turmeric	Finger	28-12-2022	9410.0	9530.0	9475.0	NaN
78649	78601	West Bengal	Murshidabad	Jangipur	178602	India	Turmeric	Finger	29-12-2022	9425.0	9550.0	9490.0	NaN
78650	78602	West Bengal	Murshidabad	Jangipur	178603	India	Turmeric	Finger	30-12-2022	9425.0	9550.0	9490.0	NaN

In [19]:
```
count = [] # Just I've used 'for loop' to add numbers from 0 - 78651 to column

for i in range(0, 78651):
    count.append(i)
print(count)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 5
9, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87,
88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 1
13, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135,
136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158,
159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204,
205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227,
228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250,
251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273,
274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296,
297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319,
320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342,
343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365,
366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388,
389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411,
412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434,
435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457,
```

In [20]:
```
df['column'] = count
```

In [21]:
```
df.head(10)
```

Out[21]:

	s_no	state	district	market	pincode	country	commodity	variety	arrival_date	min_price	max_price	modal_price	update_date	colu
0	0	Karnataka	Belgaum	Athani	100001	India	Bull	Bull	26-01-2020	20000.0	40000.0	40000.0	NaN	
1	1	Karnataka	Belgaum	Athani	100002	India	Bull	Bull	16-02-2020	10000.0	12000.0	11000.0	NaN	
2	2	Karnataka	Belgaum	Athani	100003	India	Bull	Bull	01-03-2020	32000.0	40000.0	35000.0	NaN	
3	3	Karnataka	Belgaum	Ramdurga	100004	India	Bull	Bull	05-01-2020	20000.0	25000.0	20000.0	NaN	
4	4	Karnataka	Belgaum	Ramdurga	100005	India	BULL	Bull	12-01-2020	20000.0	35000.0	25000.0	NaN	
5	5	Karnatak	Belgaum	Ramdurga	100006	India	BULL	Bull	19-01-2020	25000.0	35000.0	30000.0	NaN	
6	6	Karnataka	Belgaum	Ramdurga	100007	India	BULL	Bulll	26-01-2020	20000.0	35000.0	25000.0	NaN	
7	7	Karnataka	Belgaum	Ramdurga	100008	India	BULL	Bull	23-02-2020	25000.0	25000.0	25000.0	NaN	
8	8	Karnataka	Belgaum	Ramdurga	100009	India	BULL	Bull	08-03-2020	25000.0	25000.0	25000.0	NaN	
9	9	Karnataka	Belgaum	Ramdurga	100010	India	BULL	Bull	15-03-2020	25000.0	25000.0	25000.0	NaN	

In [22]:
```
df.tail(10)
```

Out[22]:

	s_no	state	district	market	pincode	country	commodity	variety	arrival_date	min_price	max_price	modal_price	update_da
78641	78593	West Bengal	Murshidabad	Jangipur	178594	India	Turmeric	Finger	21-12-2022	9460.0	9575.0	9525.0	N
78642	78594	West Bengal	Murshidabad	Jangipur	178595	India	Turmeric	Finger	22-12-2022	9460.0	9575.0	9525.0	N
78643	78595	West Bengal	Murshidabad	Jangipur	178596	India	Turmeric	Finger	23-12-2022	9450.0	9560.0	9515.0	N
78644	78596	West Bengal	Murshidabad	Jangipur	178597	India	Turmeric	Finger	24-12-2022	9450.0	9560.0	9515.0	N
78645	78597	West Bengal	Murshidabad	Jangipur	178598	India	Turmeric	Finger	25-12-2022	9440.0	9550.0	9500.0	N
78646	78598	West Bengal	Murshidabad	Jangipur	178599	India	Turmeric	Finger	26-12-2022	9440.0	9550.0	9500.0	N
78647	78599	West Bengal	Murshidabad	Jangipur	178600	India	Turmeric	Finger	27-12-2022	9435.0	9550.0	9490.0	N

Add Row

In [23]:
```
row = ['78651','Tamil Nadu', 'Vellore', 'Katpadi', '123456', 'India', 'Turmeric', 'Finger', '31-12-2022', '9425.0', '9
```

In [24]:
```
new_df_row = pd.DataFrame([row], columns=df.columns.values)
```

In [25]:
```
df.append(new_df_row, ignore_index=True)  # Since it is deprecated, will do it in alternative way
```

```
C:\Users\sprav\AppData\Local\Temp\ipykernel_4504\1653866130.py:1: FutureWarning: The frame.append method is deprecat
ed and will be removed from pandas in a future version. Use pandas.concat instead.
  df.append(new_df_row, ignore_index=True)  # Since it is deprecated, will do it in alternative way
```

Out[25]:

	s_no	state	district	market	pincode	country	commodity	variety	arrival_date	min_price	max_price	modal_price	update_d
0	0	Karnataka	Belgaum	Athani	100001	India	Bull	Bull	26-01-2020	20000.0	40000.0	40000.0	I
1	1	Karnataka	Belgaum	Athani	100002	India	Bull	Bull	16-02-2020	10000.0	12000.0	11000.0	I
2	2	Karnataka	Belgaum	Athani	100003	India	Bull	Bull	01-03-2020	32000.0	40000.0	35000.0	I
3	3	Karnataka	Belgaum	Ramdurga	100004	India	Bull	Bull	05-01-2020	20000.0	25000.0	20000.0	I
4	4	Karnataka	Belgaum	Ramdurga	100005	India	BULL	Bull	12-01-2020	20000.0	35000.0	25000.0	I
...	...	...	...	...	...	...	...	...	...	...	...	...	...
78647	78599	West Bengal	Murshidabad	Jangipur	178600	India	Turmeric	Finger	27-12-2022	9435.0	9550.0	9490.0	I
78648	78600	West Bengal	Murshidabad	Jangipur	178601	India	Turmeric	Finger	28-12-2022	9410.0	9530.0	9475.0	I
78649	78601	West Bengal	Murshidabad	Jangipur	178602	India	Turmeric	Finger	29-12-2022	9425.0	9550.0	9490.0	I
78650	78602	West Bengal	Murshidabad	Jangipur	178603	India	Turmeric	Finger	30-12-2022	9425.0	9550.0	9490.0	I
78651	78651	Tamil Nadu	Vellore	Katpadi	123456	India	Turmeric	Finger	31-12-2022	9425.0	9550.0	9490.0	31-12-2

78652 rows × 14 columns

Alternative way to add a row

In [26]:
```
row = ['78651','Tamil Nadu', 'Vellore', 'Katpadi', '123456', 'India', 'Turmeric', 'Finger', '31-12-2022', '9425.0', '9
```

In [27]:
```
df.loc['78651'] = row
```

In [28]:
```
df.tail(10)
```

Out[28]:

	s_no	state	district	market	pincode	country	commodity	variety	arrival_date	min_price	max_price	modal_price	update_date
78642	78594	West Bengal	Murshidabad	Jangipur	178595	India	Turmeric	Finger	22-12-2022	9460.0	9575.0	9525.0	NaN
78643	78595	West Bengal	Murshidabad	Jangipur	178596	India	Turmeric	Finger	23-12-2022	9450.0	9560.0	9515.0	NaN
78644	78596	West Bengal	Murshidabad	Jangipur	178597	India	Turmeric	Finger	24-12-2022	9450.0	9560.0	9515.0	NaN
78645	78597	West Bengal	Murshidabad	Jangipur	178598	India	Turmeric	Finger	25-12-2022	9440.0	9550.0	9500.0	NaN
78646	78598	West Bengal	Murshidabad	Jangipur	178599	India	Turmeric	Finger	26-12-2022	9440.0	9550.0	9500.0	NaN
78647	78599	West Bengal	Murshidabad	Jangipur	178600	India	Turmeric	Finger	27-12-2022	9435.0	9550.0	9490.0	NaN
78648	78600	West Bengal	Murshidabad	Jangipur	178601	India	Turmeric	Finger	28-12-2022	9410.0	9530.0	9475.0	NaN
78649	78601	West Bengal	Murshidabad	Jangipur	178602	India	Turmeric	Finger	29-12-2022	9425.0	9550.0	9490.0	NaN
78650	78602	West Bengal	Murshidabad	Jangipur	178603	India	Turmeric	Finger	30-12-2022	9425.0	9550.0	9490.0	NaN
78651	78651	Tamil Nadu	Vellore	Katpadi	123456	India	Turmeric	Finger	31-12-2022	9425.0	9550.0	9490.0	31-12-2022

Sorting DataFrames

In [29]:
```
df.dtypes # Before sorting, it's advisable to check whether the numberic columns are int and float. If not let's chang
```

Out[29]:
```
s_no            object
state           object
district        object
market          object
pincode         object
country         object
commodity       object
variety         object
arrival_date    object
min_price       object
max_price       object
modal_price     object
update_date     object
column          object
dtype: object
```

In [32]:
```
min_price = df[{'min_price': 'float'}] # Okay! it seems this method is deprecated, So will try it in alernative way
```

```
C:\Users\sprav\AppData\Local\Temp\ipykernel_4504\2597332284.py:1: FutureWarning: Passing a dict as an indexer is dep
recated and will raise in a future version. Use a list instead.
  min_price = df[{'min_price': 'float'}]
```

In [33]:
```
min_price.dtypes
```

Out[33]:
```
min_price    object
dtype: object
```

In [34]:
```
df['min_price'] = df['min_price'].astype(float) # Alternative way to change dtypes of a column or multiple.
```

In [35]:
```
df.dtypes
```

Out[35]:
```
s_no             object
state            object
district         object
market           object
pincode          object
country          object
commodity        object
variety          object
arrival_date     object
min_price       float64
max_price        object
modal_price      object
update_date      object
column           object
dtype: object
```

In [36]:
```
df.sort_values('min_price')
```

Out[36]:

	s_no	state	district	market	pincode	country	commodity	variety	arrival_date	min_price	max_price	modal_price	update
46420	46420	Odisha	Nowarangpur	Nawarangpur	146421	India	Egg	Egg	03-07-2022	2.00	2.0	2.0	
7146	7146	Odisha	Nowarangpur	Nawarangpur	107147	India	Egg	Egg	03-07-2022	2.00	2.0	2.0	
46421	46421	Odisha	Nowarangpur	Nawarangpur	146422	India	Egg	Egg	28-09-2022	2.50	2.5	2.5	
7147	7147	Odisha	Nowarangpur	Nawarangpur	107148	India	Egg	Egg	28-09-2022	2.50	2.5	2.5	
6777	6777	Odisha	Ganjam	Digapahandi	106778	India	Egg	Egg	06-04-2022	3.35	3.55	3.45	
...	...	...	...	...	...	...	...	...	...	...	...	...	
58857	58857	Punjab	Fatehgarh	Sirhind	158858	India	Grapes	Other	10-01-2022	NaN	NaN	6000.0	
58858	58858	Punjab	Fatehgarh	Sirhind	158859	India	Grapes	Other	11-01-2022	NaN	NaN	6000.0	
58860	58860	Punjab	Fatehgarh	Sirhind	158861	India	Grapes	Other	17-01-2022	NaN	NaN	6000.0	
58863	58863	Punjab	Fatehgarh	Sirhind	158864	India	Grapes	Other	24-01-2022	NaN	NaN	6000.0	
64473	64473	Uttar Pradesh	Faizabad	Faizabad	164474	India	Grapes	Green	13-12-2022	NaN	NaN	10000.0	

78652 rows × 14 columns

In [37]:
```
df.sort_values(['state', 'min_price'])
```

Out[37]:

	s_no	state	district	market	pincode	country	commodity	variety	arrival_date	min_price	max_price	modal_price	update_dat
32273	32273	Andhra Pradesh	Cuddapah	Cuddapah	132274	India	Turmeric	Finger	08-08-2022	2020.0	5625.0	4257.0	Na
71547	71547	Andhra Pradesh	Cuddapah	Cuddapah	171548	India	Turmeric	Finger	08-08-2022	2020.0	5625.0	4257.0	Na
32106	32106	Andhra Pradesh	Cuddapah	Cuddapah	132107	India	Turmeric	Bulb	25-01-2022	2189.0	6500.0	6425.0	Na
71380	71380	Andhra Pradesh	Cuddapah	Cuddapah	171381	India	Turmeric	Bulb	25-01-2022	2189.0	6500.0	6425.0	Na
32249	32249	Andhra Pradesh	Cuddapah	Cuddapah	132250	India	Turmeric	Finger	18-05-2022	2488.0	6362.0	5475.0	Na
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1474	1474	kerala	Idukki	Thodupuzha	101475	India	Copra	other	14-07-2022	8800.0	9000.0	9000.0	Na
1475	1475	kerala	Idukki	Thodupuzha	101476	India	Copra	other	15-07-2022	8800.0	9000.0	9000.0	Na
1468	1468	kerala	Idukki	Thodupuzha	101469	India	Copra	other	06-07-2022	9200.0	9300.0	9300.0	Na
1470	1470	kerala	Idukki	Thodupuzha	101471	India	Copra	other	08-07-2022	9200.0	9300.0	9300.0	Na
1471	1471	kerala	Idukki	Thodupuzha	101472	India	Copra	other	11-07-2022	9200.0	9300.0	9300.0	Na

78652 rows × 14 columns

Handling Date & Time fields in Pandas

In [41]:
```
type(df['arrival_date'])
```

Out[41]:
```
pandas.core.series.Series
```

In [44]:
```
df['arrival_date'].dtype # Object type
```

Out[44]:
```
dtype('O')
```

In [49]:
```
pd.to_datetime(df['arrival_date'], dayfirst=True)
```

Out[49]:
```
0       2020-01-26
1       2020-02-16
2       2020-03-01
3       2020-01-05
4       2020-01-12
           ...
78647   2022-12-27
78648   2022-12-28
78649   2022-12-29
78650   2022-12-30
78651   2022-12-31
Name: arrival_date, Length: 78652, dtype: datetime64[ns]
```

In [54]:
```
df['arrival_date'] = pd.to_datetime(df['arrival_date'], dayfirst=True)
```

In [55]:
```
df['arrival_date'][0] # Converted into Timestamp!
```

Out[55]:
```
Timestamp('2020-01-26 00:00:00')
```

In [57]:
```
df.dtypes
```

Out[57]:
```
s_no                    object
state                   object
district                object
market                  object
pincode                 object
country                 object
commodity               object
variety                 object
arrival_date    datetime64[ns]
min_price              float64
max_price               object
modal_price             object
update_date             object
column                  object
dtype: object
```

In [61]:
```
df['arrival_date'].dt.date
```

Out[61]:
```
0        2020-01-26
1        2020-01-26
2        2020-01-26
3        2020-01-26
4        2020-01-26
            ...
78647    2020-01-26
78648    2020-01-26
78649    2020-01-26
78650    2020-01-26
78651    2020-01-26
Name: arrival_date, Length: 78652, dtype: object
```

In [62]:
```
df['arrival_date'].dt.month
```

Out[62]:
```
0        1
1        1
2        1
3        1
4        1
        ..
78647    1
78648    1
78649    1
78650    1
78651    1
Name: arrival_date, Length: 78652, dtype: int64
```

In [63]:
```
df['arrival_date'].dt.year
```

Out[63]:
```
0        2020
1        2020
2        2020
3        2020
4        2020
         ...
78647    2020
78648    2020
78649    2020
78650    2020
78651    2020
Name: arrival_date, Length: 78652, dtype: int64
```

Statistical functions in pandas

In [65]:
```
# All about pandas supports mathematical functions

#Sum
#Min
#Max
#Mean
#Correlation
```

In [70]:
```
df
```

Out[70]:

	s_no	state	district	market	pincode	country	commodity	variety	arrival_date	min_price	max_price	modal_price	update_d
0	0	Karnataka	Belgaum	Athani	100001	India	Bull	Bull	2020-01-26	20000.0	40000.0	40000.0	I
1	1	Karnataka	Belgaum	Athani	100002	India	Bull	Bull	2020-01-26	10000.0	12000.0	11000.0	I
2	2	Karnataka	Belgaum	Athani	100003	India	Bull	Bull	2020-01-26	32000.0	40000.0	35000.0	I
3	3	Karnataka	Belgaum	Ramdurga	100004	India	Bull	Bull	2020-01-26	20000.0	25000.0	20000.0	I
4	4	Karnataka	Belgaum	Ramdurga	100005	India	BULL	Bull	2020-01-26	20000.0	35000.0	25000.0	I
...	...	...	...	...	...	...	...	...	...	...	...	...	...
78647	78599	West Bengal	Murshidabad	Jangipur	178600	India	Turmeric	Finger	2020-01-26	9435.0	9550.0	9490.0	I
78648	78600	West Bengal	Murshidabad	Jangipur	178601	India	Turmeric	Finger	2020-01-26	9410.0	9530.0	9475.0	I
78649	78601	West Bengal	Murshidabad	Jangipur	178602	India	Turmeric	Finger	2020-01-26	9425.0	9550.0	9490.0	I
78650	78602	West Bengal	Murshidabad	Jangipur	178603	India	Turmeric	Finger	2020-01-26	9425.0	9550.0	9490.0	I
78651	78651	Tamil Nadu	Vellore	Katpadi	123456	India	Turmeric	Finger	2020-01-26	9425.0	9550.0	9490.0	31-12-2

78652 rows × 14 columns

Min

In [67]:
```
df['min_price'].min()
```

Out[67]:
```
2.0
```

In [71]:
```
df['state'][df['min_price'] == df['min_price'].min()]
```

Out[71]:
```
7146     Odisha
46420    Odisha
Name: state, dtype: object
```

Max

In [68]:
```
df['min_price'].max()
```

Out[68]:
```
1000000.0
```

In [72]:
```
df['state'][df['min_price'] == df['min_price'].max()]
```

Out[72]:
```
12269    Himachal Pradesh
51543    Himachal Pradesh
Name: state, dtype: object
```

Mean

In [69]:
```
df['min_price'].mean()
```

Out[69]:
```
6578.121674991081
```

Converting one more column to float dtype

In [74]:
```
df['max_price'] = df['max_price'].astype(float)
```

In [75]:
```
df['max_price'].dtype
```

Out[75]:
```
dtype('float64')
```

In [76]:
```
df['sum'] = df['min_price'] + df['max_price']
```

In [77]:
```
df
```

Out[77]:

	s_no	state	district	market	pincode	country	commodity	variety	arrival_date	min_price	max_price	modal_price	update_d
0	0	Karnataka	Belgaum	Athani	100001	India	Bull	Bull	2020-01-26	20000.0	40000.0	40000.0	I
1	1	Karnataka	Belgaum	Athani	100002	India	Bull	Bull	2020-01-26	10000.0	12000.0	11000.0	I
2	2	Karnataka	Belgaum	Athani	100003	India	Bull	Bull	2020-01-26	32000.0	40000.0	35000.0	I
3	3	Karnataka	Belgaum	Ramdurga	100004	India	Bull	Bull	2020-01-26	20000.0	25000.0	20000.0	I
4	4	Karnataka	Belgaum	Ramdurga	100005	India	BULL	Bull	2020-01-26	20000.0	35000.0	25000.0	I
...	...	...	...	...	...	...	...	...	...	...	...	...	...
78647	78599	West Bengal	Murshidabad	Jangipur	178600	India	Turmeric	Finger	2020-01-26	9435.0	9550.0	9490.0	I
78648	78600	West Bengal	Murshidabad	Jangipur	178601	India	Turmeric	Finger	2020-01-26	9410.0	9530.0	9475.0	I
78649	78601	West Bengal	Murshidabad	Jangipur	178602	India	Turmeric	Finger	2020-01-26	9425.0	9550.0	9490.0	I
78650	78602	West Bengal	Murshidabad	Jangipur	178603	India	Turmeric	Finger	2020-01-26	9425.0	9550.0	9490.0	I
78651	78651	Tamil Nadu	Vellore	Katpadi	123456	India	Turmeric	Finger	2020-01-26	9425.0	9550.0	9490.0	31-12-2

78652 rows × 15 columns

In [78]:
```
df['sum'][0] # First index value of sum column
```

Out[78]:
```
60000.0
```

Correlation

In [80]:
```
df.corr(numeric_only=[True,False])  # Will show only the numerical values
```

Out[80]:

	min_price	max_price	sum
min_price	1.000000	0.984861	0.995573
max_price	0.984861	1.000000	0.996794
sum	0.995573	0.996794	1.000000

In [82]:
```
df[['min_price', 'sum']].corr(numeric_only=True)
```

Out[82]:

	min_price	sum
min_price	1.000000	0.995573
sum	0.995573	1.000000

Pandas Indexing

In [118]:
```
import pandas as pd
import seaborn as sns
```

In [120]:
```
data = pd.read_csv('Binary predictors.csv')
df = pd.DataFrame(data)
df.head(10)
```

Out[120]:

	Marks	Admitted	Gender
0	1363	No	Male
1	1792	Yes	Female
2	1954	Yes	Female
3	1653	No	Male
4	1593	No	Male
5	1755	Yes	Female
6	1775	Yes	Female
7	1887	Yes	Female
8	1893	Yes	Female
9	1580	No	Male

In [122]:
```
print(df.index.names) # No index values
```

```
[None]
```

Add Index

In [82]:
```
df.dtypes
```

Out[82]:
```
s_no              int64
state            object
district         object
market           object
pincode           int64
country          object
commodity        object
variety          object
arrival_date     object
min_price       float64
max_price       float64
modal_price     float64
update_date      object
dtype: object
```

In [125]:
```
df.shape
```

Out[125]:
```
(168, 3)
```

In [128]:
```
po_number = []

for i in range(0,168):
    po_number.append(i)
print(po_number)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 3
1, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 6
0, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 8
9, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 11
4, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 13
7, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 16
0, 161, 162, 163, 164, 165, 166, 167]
```

In [129]:
```
df['PONumber'] = po_number # It's added at the end. So, we can interchange it as primary using index.
```

In [130]:
```
df
```

Out[130]:

	Marks	Admitted	Gender	PONumber
0	1363	No	Male	0
1	1792	Yes	Female	1
2	1954	Yes	Female	2
3	1653	No	Male	3
4	1593	No	Male	4
...	...	...	...	...
163	1722	Yes	Female	163
164	1750	Yes	Male	164
165	1555	No	Male	165
166	1524	No	Male	166
167	1461	No	Male	167

168 rows × 4 columns

In [131]:
```
df.set_index("PONumber", inplace=True) # inplace set to True, because to save the PONumber column in the dataset.
```

In [132]:
```
df
```

Out[132]:

PONumber	Marks	Admitted	Gender
0	1363	No	Male
1	1792	Yes	Female
2	1954	Yes	Female
3	1653	No	Male
4	1593	No	Male
...	...	...	...
163	1722	Yes	Female
164	1750	Yes	Male
165	1555	No	Male
166	1524	No	Male
167	1461	No	Male

168 rows × 3 columns

In [133]:
```
print(df.index.names) # Now we added values to the index
```

```
['PONumber']
```

In [134]:
```
df.loc[123]
```

Out[134]:
```
Marks       1430
Admitted      No
Gender      Male
Name: 123, dtype: object
```

Delete Index

In [135]:
```
df.reset_index(inplace=True)
```

In [136]:
```
print(df.index.names)
```

```
[None]
```

In [137]:
```
df
```

Out[137]:

	PONumber	Marks	Admitted	Gender
0	0	1363	No	Male
1	1	1792	Yes	Female
2	2	1954	Yes	Female
3	3	1653	No	Male
4	4	1593	No	Male
...	...	...	...	...
163	163	1722	Yes	Female
164	164	1750	Yes	Male
165	165	1555	No	Male
166	166	1524	No	Male
167	167	1461	No	Male

168 rows × 4 columns

In [138]:
```
df.drop(columns=['PONumber'], inplace=True)
# Can delete the column
```

In [139]:
```
df
```

Out[139]:

	Marks	Admitted	Gender
0	1363	No	Male
1	1792	Yes	Female
2	1954	Yes	Female
3	1653	No	Male
4	1593	No	Male
...	...	...	...
163	1722	Yes	Female
164	1750	Yes	Male
165	1555	No	Male
166	1524	No	Male
167	1461	No	Male

168 rows × 3 columns

In [143]:
```
df.set_index('Gender', inplace=True) # can change any column as index
```

In [144]:
```
df
```

Out[144]:

Gender	Marks	Admitted
Male	1363	No
Female	1792	Yes
Female	1954	Yes
Male	1653	No
Male	1593	No
...	...	...
Female	1722	Yes
Male	1750	Yes
Male	1555	No
Male	1524	No
Male	1461	No

168 rows × 2 columns

In [145]:
```
print(df.index.names)
```

```
['Gender']
```

In [146]:
```
df.reset_index(inplace=True) # Resetting again
```

In [148]:
```
print(df.index.names)
```

```
[None]
```

Data Grouping

In [185]:
```
data = pd.read_csv('commodity_price.csv', low_memory=False)
df = pd.DataFrame(data, )
df.head(10)
```

Out[185]:

	s_no	state	district	market	pincode	country	commodity	variety	arrival_date	min_price	max_price	modal_price	update_date
0	0	Karnataka	Belgaum	Athani	100001	India	Bull	Bull	26-01-2020	20000.0	40000.0	40000.0	NaN
1	1	Karnataka	Belgaum	Athani	100002	India	Bull	Bull	16-02-2020	10000.0	12000.0	11000.0	NaN
2	2	Karnataka	Belgaum	Athani	100003	India	Bull	Bull	01-03-2020	32000.0	40000.0	35000.0	NaN
3	3	Karnataka	Belgaum	Ramdurga	100004	India	Bull	Bull	05-01-2020	20000.0	25000.0	20000.0	NaN
4	4	Karnataka	Belgaum	Ramdurga	100005	India	BULL	Bull	12-01-2020	20000.0	35000.0	25000.0	NaN
5	5	Karnatak	Belgaum	Ramdurga	100006	India	BULL	Bull	19-01-2020	25000.0	35000.0	30000.0	NaN
6	6	Karnataka	Belgaum	Ramdurga	100007	India	BULL	Bulll	26-01-2020	20000.0	35000.0	25000.0	NaN
7	7	Karnataka	Belgaum	Ramdurga	100008	India	BULL	Bull	23-02-2020	25000.0	25000.0	25000.0	NaN
8	8	Karnataka	Belgaum	Ramdurga	100009	India	BULL	Bull	08-03-2020	25000.0	25000.0	25000.0	NaN
9	9	Karnataka	Belgaum	Ramdurga	100010	India	BULL	Bull	15-03-2020	25000.0	25000.0	25000.0	NaN

In [193]:
```
df.groupby('state').sum(numeric_only=[False,True])
```

Out[193]:

state	s_no	pincode	min_price	max_price	modal_price
Andhra Pradesh	61707270	189508548	6.921120e+06	11326192.0	9705870.0
Bihar	13484346	63284844	2.840900e+06	3475440.0	3133740.0
Chandigarh	2395668	11195756	2.780000e+05	605000.0	441500.0
Chattisgarh	42119668	150520752	4.720404e+06	5381560.0	5055020.0
Goa	9381178	48181566	1.798800e+06	2158800.0	1978700.0
Gujarat	7152094	33352356	1.509500e+06	2603700.0	2173300.0
Haryana	219798540	907805420	2.534252e+07	33684732.0	30474736.0
Himachal Pradesh	91242588	381845494	4.785420e+07	58430400.0	53170300.0
Jammu and Kashmir	40546250	165547500	8.667600e+06	10276600.0	9462618.0
Karala	33979	1833997	1.455000e+05	153900.0	151200.0
Karnatak	208	800216	1.470000e+05	231000.0	193000.0
Karnataka	135931515	628536441	4.270001e+07	53697498.0	47867634.0
Karnatka	151	400155	8.100000e+04	112000.0	98000.0
Kerala	529816076	1963330411	1.539824e+08	160670540.0	157648560.0
Madhya Pradesh	43919468	164320672	3.107820e+06	4342950.0	3645596.0
Maharashtra	261428654	835634396	2.658896e+07	40293868.0	33631680.0
Manipur	3291498	15891624	3.234000e+06	3505000.0	3369000.0
Meghalaya	1948698	5548734	5.735000e+04	67400.0	61900.0
NCT of Delhi	28271684	102472426	2.913400e+06	6795000.0	4718006.0
Nagaland	3140932	8940990	1.978000e+05	215800.0	205200.0
Odisha	30749298	129150282	1.714351e+06	1784615.5	1747927.5
Punjab	364112316	1262321298	4.200117e+07	63971960.0	51021502.0
Rajasthan	77801640	258803450	7.118200e+06	10467800.0	8817100.0
Tamil Nadu	415163046	1466773562	7.109754e+07	86619718.0	80942726.0
Telangana	298150122	866755808	2.334058e+07	34172106.0	29474512.0

state	s_no	pincode	min_price	max_price	modal_price
Tripura	3108069	16008198	2.601800e+05	485000.0	375520.0
Uttar Pradesh	284362858	911969134	2.619769e+07	27699280.8	26967302.3
Uttrakhand	83411258	259013014	5.266208e+06	7046428.0	6141408.0
West Bengal	40469472	107670144	6.120920e+06	6204670.0	6164380.0
kerala	10303	710310	6.280000e+04	63900.0	63900.0

In [192]:
```
df.groupby('state')['min_price'].sum(numeric_only=True)
```

Out[192]:
```
state
Andhra Pradesh       6.921120e+06
Bihar                2.840900e+06
Chandigarh           2.780000e+05
Chattisgarh          4.720404e+06
Goa                  1.798800e+06
Gujarat              1.509500e+06
Haryana              2.534252e+07
Himachal Pradesh     4.785420e+07
Jammu and Kashmir    8.667600e+06
Karala               1.455000e+05
Karnatak             1.470000e+05
Karnataka            4.270001e+07
Karnatka             8.100000e+04
Kerala               1.539824e+08
Madhya Pradesh       3.107820e+06
Maharashtra          2.658896e+07
Manipur              3.234000e+06
Meghalaya            5.735000e+04
NCT of Delhi         2.913400e+06
Nagaland             1.978000e+05
Odisha               1.714351e+06
Punjab               4.200117e+07
Rajasthan            7.118200e+06
Tamil Nadu           7.109754e+07
Telangana            2.334058e+07
Tripura              2.601800e+05
Uttar Pradesh        2.619769e+07
Uttrakhand           5.266208e+06
West Bengal          6.120920e+06
kerala               6.280000e+04
Name: min_price, dtype: float64
```

In [194]:
```
df.groupby('state').min(numeric_only=[False,True])
```

Out[194]:

state	s_no	pincode	min_price	max_price	modal_price
Andhra Pradesh	6205	106206	2020.0	4300.0	2600.0
Bihar	7316	107317	3000.0	3200.0	3100.0
Chandigarh	7565	107566	3000.0	6000.0	4500.0
Chattisgarh	6296	106297	200.0	200.0	200.0
Goa	55	100056	3000.0	3000.0	3000.0
Gujarat	6309	106310	1500.0	3000.0	2250.0
Haryana	8093	108094	100.0	125.0	110.0
Himachal Pradesh	11035	111036	1500.0	1600.0	1550.0
Jammu and Kashmir	12488	112489	3000.0	3500.0	3250.0
Karala	1483	101484	7500.0	8000.0	7800.0
Karnatak	5	100006	14000.0	21000.0	18000.0
Karnataka	0	100001	89.0	300.0	200.0
Karnatka	35	100036	18000.0	21000.0	21000.0
Kerala	1320	101321	600.0	0.0	600.0
Madhya Pradesh	6452	106453	250.0	700.0	250.0
Maharashtra	16886	116887	12.0	60.0	50.0
Manipur	6455	106456	23000.0	27000.0	25000.0
Meghalaya	34485	134486	1125.0	1200.0	1150.0
NCT of Delhi	18280	118281	1500.0	4500.0	3375.0
Nagaland	34503	134504	1700.0	2000.0	1900.0
Odisha	6518	106519	2.0	2.0	2.0
Punjab	18656	118657	300.0	600.0	500.0
Rajasthan	6519	106520	1000.0	2000.0	1500.0
Tamil Nadu	3519	103520	450.0	780.0	550.0
Telangana	6542	106543	100.0	100.0	100.0

state	s_no	pincode	min_price	max_price	modal_price
Tripura	44	100045	390.0	550.0	500.0
Uttar Pradesh	6666	106667	5.0	5.7	5.4
Uttrakhand	27413	127414	400.0	700.0	600.0
West Bengal	39017	139018	8535.0	8670.0	8600.0
kerala	1468	101469	8800.0	9000.0	9000.0

In [195]:
```
df.groupby('state').max(numeric_only=[False,True])
```

Out[195]:

state	s_no	pincode	min_price	max_price	modal_price
Andhra Pradesh	71914	171915	20000.0	38999.0	26000.0
Bihar	46838	146839	8000.0	10000.0	9000.0
Chandigarh	46882	146883	5000.0	9000.0	7000.0
Chattisgarh	71936	171937	22000.0	26000.0	24000.0
Goa	47264	147265	6200.0	8100.0	7150.0
Gujarat	47366	147367	16000.0	26000.0	25000.0
Haryana	68061	168062	15000.0	28000.0	26500.0
Himachal Pradesh	51761	151762	1000000.0	1000000.0	1000000.0
Jammu and Kashmir	52386	152387	28000.0	32000.0	30000.0
Karala	2126	102127	9800.0	10000.0	10000.0
Karnatak	32	100033	25000.0	35000.0	30000.0
Karnataka	72073	172074	61010.0	73999.0	61010.0
Karnatka	43	100044	22000.0	38000.0	30000.0
Kerala	72277	172278	150000.0	180000.0	160000.0
Madhya Pradesh	72288	172289	13000.0	28800.0	15250.0
Maharashtra	73758	173759	50000.0	70000.0	60000.0
Manipur	45791	145792	27000.0	28500.0	27500.0
Meghalaya	73776	173777	2000.0	2500.0	2100.0
NCT of Delhi	57924	157925	10000.0	26000.0	20000.0
Nagaland	73805	173806	4700.0	5000.0	4800.0
Odisha	73886	173887	9950.0	10000.0	10000.0
Punjab	62420	162421	31400.0	34300.0	31400.0
Rajasthan	63312	163313	16000.0	30000.0	22500.0
Tamil Nadu	76306	176307	70000.0	85000.0	76000.0
Telangana	78124	178125	72500.0	72500.0	72500.0

state	s_no	pincode	min_price	max_price	modal_price
Tripura	46001	146002	22000.0	45000.0	35000.0
Uttar Pradesh	78289	178290	19780.0	20150.0	19900.0
Uttrakhand	78290	178291	6000.0	8000.0	7000.0
West Bengal	78602	178603	9460.0	9575.0	9525.0
kerala	1475	101476	9200.0	9300.0	9300.0

In [196]:
```
df.groupby(['state', 'district']).min(numeric_only=True)
```

Out[196]:

state	district	s_no	pincode	min_price	max_price	modal_price
Andhra Pradesh	**Cuddapah**	32093	132094	2020.0	4790.0	4257.0
	Guntur	6205	106206	2600.0	4300.0	2600.0
	Kurnool	6278	106279	5441.0	26786.0	10319.0
Bihar	**Araria**	7316	107317	3800.0	4450.0	4200.0
	Gopalgang	7406	107407	8000.0	8150.0	8100.0
...	**...**	...	...	...	...	...
Uttrakhand	**Haridwar**	27932	127933	2000.0	2800.0	2500.0
	Nanital	28050	128051	2500.0	3000.0	2700.0
	UdhamSinghNagar	28091	128092	1900.0	2000.0	1950.0
West Bengal	**Murshidabad**	39017	139018	8535.0	8670.0	8600.0
kerala	**Idukki**	1468	101469	8800.0	9000.0	9000.0

256 rows × 5 columns

In [197]:
```
df.groupby(['state', 'district']).max(numeric_only=True)
```

Out[197]:

state	district	s_no	pincode	min_price	max_price	modal_price
Andhra Pradesh	Cuddapah	71547	171548	6930.0	7668.0	7382.0
	Guntur	71914	171915	20000.0	28000.0	26000.0
	Kurnool	45569	145570	7800.0	38999.0	24112.0
Bihar	Araria	46679	146680	7200.0	8200.0	7400.0
	Gopalgang	46681	146682	8000.0	9250.0	8500.0
...	...	...	...	...	...	...
Uttrakhand	Haridwar	67323	167324	3300.0	4000.0	3400.0
	Nanital	67364	167365	6000.0	7000.0	6500.0
	UdhamSinghNagar	67563	167564	6000.0	8000.0	7000.0
West Bengal	Murshidabad	78602	178603	9460.0	9575.0	9525.0
kerala	Idukki	1475	101476	9200.0	9300.0	9300.0

256 rows × 5 columns

In [199]:
```
df.groupby(['state', 'district'])[['min_price', 'max_price', 'modal_price']].max(numeric_only=True)
```

Out[199]:

state	district	min_price	max_price	modal_price
Andhra Pradesh	Cuddapah	6930.0	7668.0	7382.0
	Guntur	20000.0	28000.0	26000.0
	Kurnool	7800.0	38999.0	24112.0
Bihar	Araria	7200.0	8200.0	7400.0
	Gopalgang	8000.0	9250.0	8500.0
...	...	...	...	...
Uttrakhand	Haridwar	3300.0	4000.0	3400.0
	Nanital	6000.0	7000.0	6500.0
	UdhamSinghNagar	6000.0	8000.0	7000.0
West Bengal	Murshidabad	9460.0	9575.0	9525.0
kerala	Idukki	9200.0	9300.0	9300.0

256 rows × 3 columns

Aggregate - Can able to do multiple mathematical functions

In [210]:
```
df.groupby(['state', 'district']).agg({'min_price': 'min', 'max_price': 'max'})
```

Out[210]:

state	district	min_price	max_price
Andhra Pradesh	Cuddapah	2020.0	7668.0
	Guntur	2600.0	28000.0
	Kurnool	5441.0	38999.0
Bihar	Araria	3800.0	8200.0
	Gopalgang	8000.0	9250.0
...	...	...	...
Uttrakhand	Haridwar	2000.0	4000.0
	Nanital	2500.0	7000.0
	UdhamSinghNagar	1900.0	8000.0
West Bengal	Murshidabad	8535.0	9575.0
kerala	Idukki	8800.0	9300.0

256 rows × 2 columns

Data Visualization

In [212]:
```
data = pd.read_csv('Binary predictors.csv')
df = pd.DataFrame(data)
df.head(10)
```

Out[212]:

	Marks	Admitted	Gender
0	1363	No	Male
1	1792	Yes	Female
2	1954	Yes	Female
3	1653	No	Male
4	1593	No	Male
5	1755	Yes	Female
6	1775	Yes	Female
7	1887	Yes	Female
8	1893	Yes	Female
9	1580	No	Male

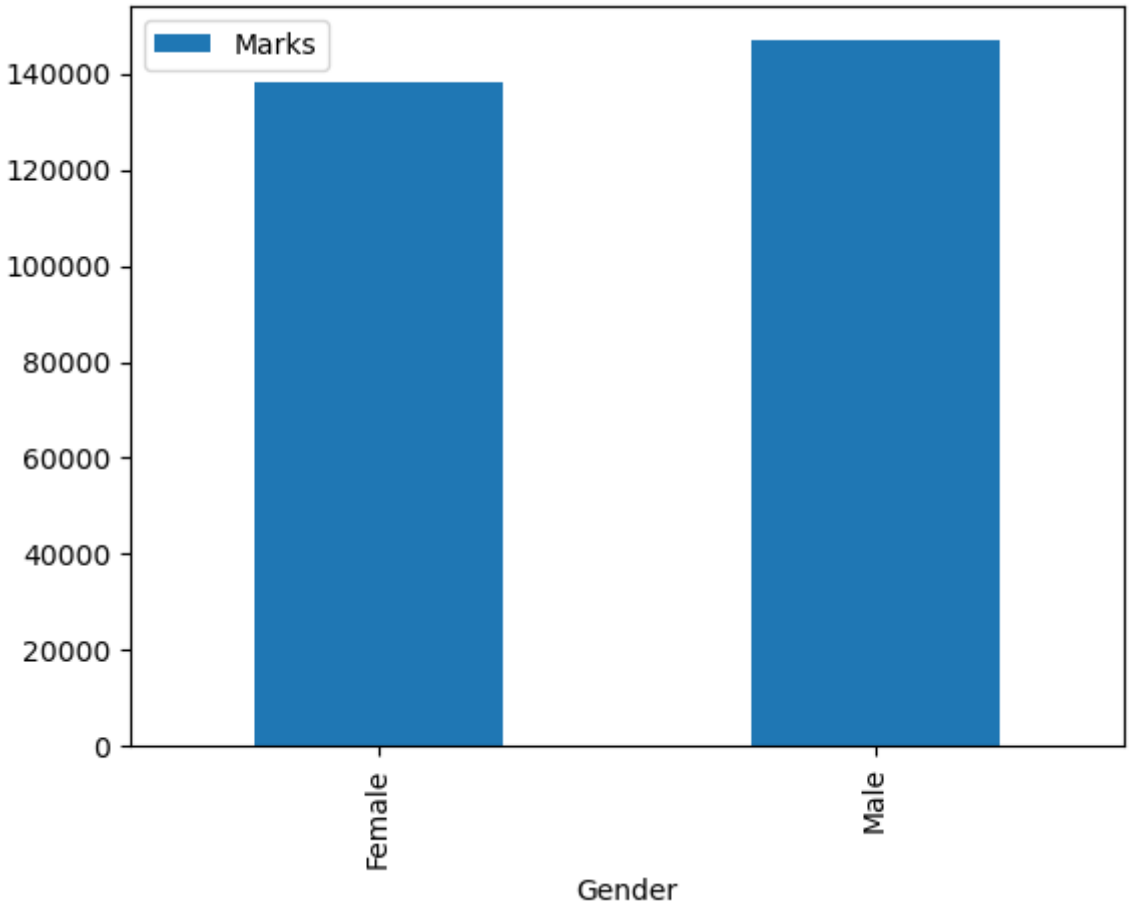
In [214]:
```
groupby_df = df.groupby(['Gender'])[['Marks']].sum(numeric_only=True)
groupby_df
```

Out[214]:

	Marks
Gender	
Female	138112
Male	146694

In [216]:
```
groupby_df.plot.bar()
```

Out[216]: <Axes: xlabel='Gender'>

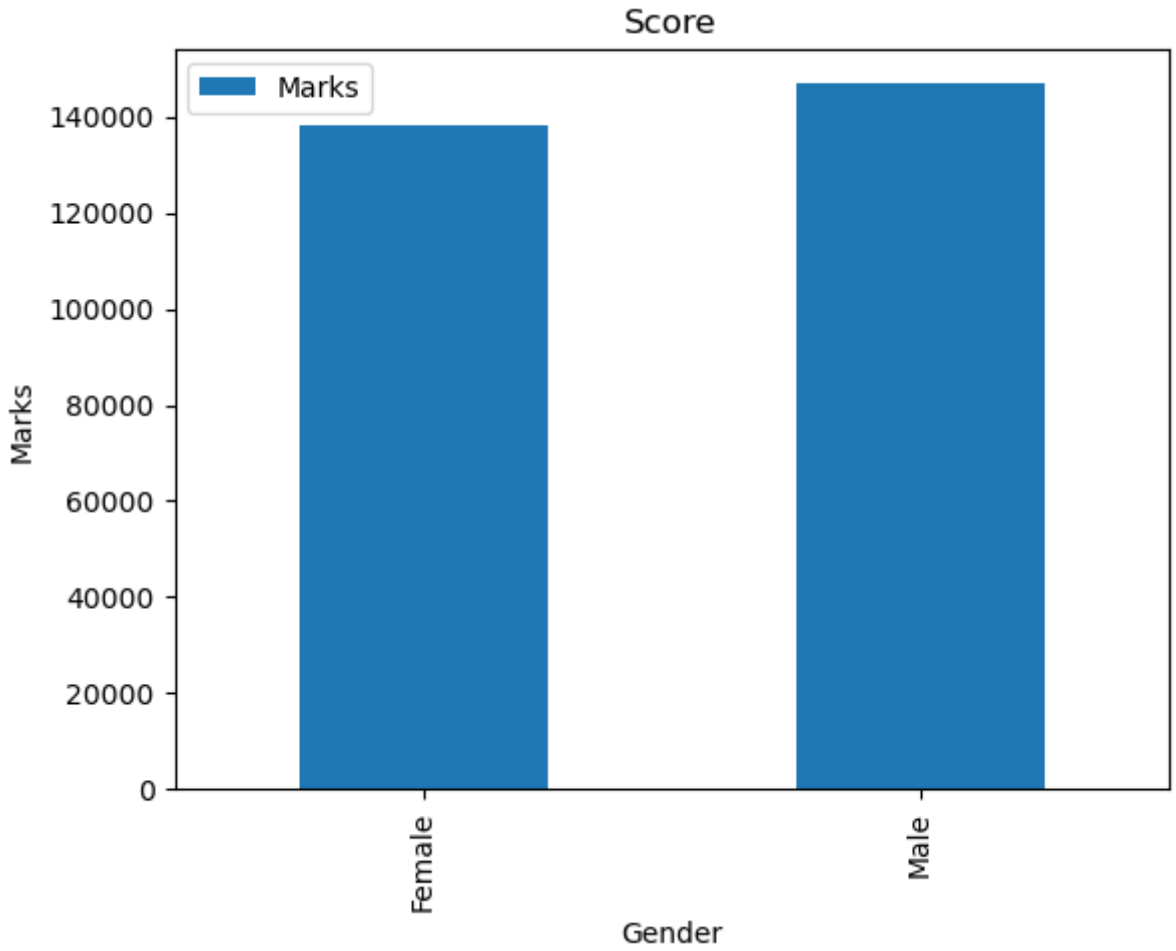


In [217]:
```
print(groupby_df.plot.__doc__) # Can view the documentation of a particular method/attribute

# Here you can see, we can able to use different plots
```

```
    Make plots of Series or DataFrame.

    Uses the backend specified by the
    option ``plotting.backend``. By default, matplotlib is used.

    Parameters
    ----------
    data : Series or DataFrame
        The object for which the method is called.
    x : label or position, default None
        Only used if data is a DataFrame.
    y : label, position or list of label, positions, default None
        Allows plotting of one column versus another. Only used if data is a
        DataFrame.
    kind : str
        The kind of plot to produce:

        - 'line' : line plot (default)
```

```
In [229]: groupby_df.plot.bar(title='Score', xlabel='Gender', ylabel='Marks')
```

```
Out[229]: <Axes: title={'center': 'Score'}, xlabel='Gender', ylabel='Marks'>
```



Adding New Data - Merge, Join, concatenate, append

In [230]:
```
import seaborn as sns
```

In [238]:
```
data = sns.load_dataset('tips')
new_df = pd.DataFrame(data)
new_df
```

Out[238]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

In [239]:
```
data1 = sns.load_dataset('iris')
new_df1 = pd.DataFrame(data1)
new_df1
```

Out[239]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

Merge

In [245]:
```
new_df.merge(new_df1, left_on='size', right_on='petal_width')
```

```
C:\Users\sprav\AppData\Local\Temp\ipykernel_10788\4134792489.py:1: UserWarning: You are merging on int and float col
umns where the float values are not equal to their int representation.
  new_df.merge(new_df1, left_on='size', right_on='petal_width')
```

Out[245]:

	total_bill	tip	sex	smoker	day	time	size	sepal_length	sepal_width	petal_length	petal_width	species
0	16.99	1.01	Female	No	Sun	Dinner	2	6.5	3.2	5.1	2.0	virginica
1	16.99	1.01	Female	No	Sun	Dinner	2	5.7	2.5	5.0	2.0	virginica
2	16.99	1.01	Female	No	Sun	Dinner	2	5.6	2.8	4.9	2.0	virginica
3	16.99	1.01	Female	No	Sun	Dinner	2	7.7	2.8	6.7	2.0	virginica
4	16.99	1.01	Female	No	Sun	Dinner	2	7.9	3.8	6.4	2.0	virginica
...	...	...	...	...	...	...	...	...	...	...	...	...
959	8.58	1.92	Male	Yes	Fri	Lunch	1	6.0	2.2	4.0	1.0	versicolor
960	8.58	1.92	Male	Yes	Fri	Lunch	1	5.8	2.7	4.1	1.0	versicolor
961	8.58	1.92	Male	Yes	Fri	Lunch	1	5.7	2.6	3.5	1.0	versicolor
962	8.58	1.92	Male	Yes	Fri	Lunch	1	5.5	2.4	3.7	1.0	versicolor
963	8.58	1.92	Male	Yes	Fri	Lunch	1	5.0	2.3	3.3	1.0	versicolor

964 rows × 12 columns

Join

In [243]:
```
new_df.join(new_df1)
```

Out[243]:

	total_bill	tip	sex	smoker	day	time	size	sepal_length	sepal_width	petal_length	petal_width	species
0	16.99	1.01	Female	No	Sun	Dinner	2	5.1	3.5	1.4	0.2	setosa
1	10.34	1.66	Male	No	Sun	Dinner	3	4.9	3.0	1.4	0.2	setosa
2	21.01	3.50	Male	No	Sun	Dinner	3	4.7	3.2	1.3	0.2	setosa
3	23.68	3.31	Male	No	Sun	Dinner	2	4.6	3.1	1.5	0.2	setosa
4	24.59	3.61	Female	No	Sun	Dinner	4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3	NaN	NaN	NaN	NaN	NaN
240	27.18	2.00	Female	Yes	Sat	Dinner	2	NaN	NaN	NaN	NaN	NaN
241	22.67	2.00	Male	Yes	Sat	Dinner	2	NaN	NaN	NaN	NaN	NaN
242	17.82	1.75	Male	No	Sat	Dinner	2	NaN	NaN	NaN	NaN	NaN
243	18.78	3.00	Female	No	Thur	Dinner	2	NaN	NaN	NaN	NaN	NaN

244 rows × 12 columns

Concatenate

In [247]:
```
new_df = df.copy()
new_df1 = df.copy()
```

In [280]:
```
new_df.shape # Before concatenate rows
```

Out[280]:
```
(168, 3)
```

In [262]:
```
concat = pd.concat([new_df, new_df1]) # works for rows
```

In [266]:
```
concat.shape # After concatenate, rows increased!
```

Out[266]:
```
(336, 3)
```

In [272]:
```
index = []

for i in range(0, 336):
    index.append(i)
concat['index'] = index
concat.set_index('index', inplace=True)
```

In [276]:
```
print(concat.index.names)
```

```
['index']
```

In [278]:
```
concat.loc[123]
```

Out[278]:
```
Marks       1430
Admitted      No
Gender      Male
Name: 123, dtype: object
```

In [285]:
```
new_df.shape # Befor concatenate columns
```

Out[285]:
```
(168, 3)
```

In [281]:
```
concat_row = pd.concat([new_df, new_df1], axis=1) # works for columns
```

In [282]:
```
concat_row
```

Out[282]:

	Marks	Admitted	Gender	Marks	Admitted	Gender
0	1363	No	Male	1363	No	Male
1	1792	Yes	Female	1792	Yes	Female
2	1954	Yes	Female	1954	Yes	Female
3	1653	No	Male	1653	No	Male
4	1593	No	Male	1593	No	Male
...	...	...	...	...	...	...
163	1722	Yes	Female	1722	Yes	Female
164	1750	Yes	Male	1750	Yes	Male
165	1555	No	Male	1555	No	Male
166	1524	No	Male	1524	No	Male
167	1461	No	Male	1461	No	Male

168 rows × 6 columns

In [284]:
```
concat_row.shape # After concatenate, columns increased!
```

Out[284]:
```
(168, 6)
```

Append

In [286]:
```
df.append([new_df, new_df1]) # Gonna remove this methode from pandas. So, we can consider using Concat itself:-)
```

```
C:\Users\sprav\AppData\Local\Temp\ipykernel_10788\562343117.py:1: FutureWarning: The frame.append method is deprecat
ed and will be removed from pandas in a future version. Use pandas.concat instead.
  df.append([new_df, new_df1])
```

Out[286]:

	Marks	Admitted	Gender
0	1363	No	Male
1	1792	Yes	Female
2	1954	Yes	Female
3	1653	No	Male
4	1593	No	Male
...	...	...	...
163	1722	Yes	Female
164	1750	Yes	Male
165	1555	No	Male
166	1524	No	Male
167	1461	No	Male

504 rows × 3 columns