

!



**1. Define the Pandas/Python pandas?**

**2. Why Use Pandas?**

**3. What Can Pandas Do?**

**4. What is a Series?**

**5. Labels**

**7. What is a DataFrame?**

**8. Load Files Into a DataFrame?**

**(i) Excel file**

**(ii) CSV file**

**(iii) Json file**

**(iv) github to read file**

## 9. Data Cleaning

## 10. statistical analysis

## 11. check the null values

## 12. Discovering Duplicates

## 13. some query from df

## 14. Data Correlations

## 15. Basic operation

### 1) Define the Pandas/Python pandas?

**Ans :** Pandas is defined as an open-source library that provides high-performance data manipulation in Python. The name of

Pandas is derived from the word Panel Data, which means an Econometrics from Multidimensional data. It can be used for data

analysis in Python and developed by Wes McKinney in 2008. It can perform five significant steps that are required for

processing and analysis of data irrespective of the origin of the data, i.e., load, manipulate, prepare, model, and analyze.

**\*\*Pandas is a Python library.**

**Pandas is used to analyze data.**

```
In [1]: # Important Libraries.....
```

```
import pandas as pd
```

```
In [2]: ## pd is here alias.
```

```
#Pandas is usually imported under the pd alias.
```

### Why Use Pandas?

**Ans:** Pandas allows us to analyze big data and make conclusions based on statistical theories.

Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science.

## What Can Pandas Do?

**Ans** Pandas gives you answers about the data. Like:

Is there a correlation between two or more columns? What is average value? Max value? Min value? Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called cleaning the data.

## Installation of Pandas

If you have Python and PIP already installed on a system, then installation of Pandas is very easy.

Install it using this command:

```
pip install pandas
```

If this command fails, then use a python distribution that already has Pandas installed like, Anaconda, Spyder etc.

## Import Pandas

Once Pandas is installed, import it in your applications by adding the import keyword:

```
In [3]: import pandas
```

```
In [4]: import pandas

mydataset = {
    'Name' : ['Himanshu', 'Virat'],
    'course' : ['Data scientist', 'Data analytics'],
    'location' : ['New Delhi', 'Uttarakhand'],
}
myvar = pd.DataFrame(mydataset)
print(myvar)
```

	Name	course	location
0	Himanshu	Data scientist	New Delhi
1	Virat	Data analytics	Uttarakhand

## Checking Pandas Version

The version string is stored under **version** attribute.

```
In [5]: import pandas as pd  
print(pd.__version__)
```

1.5.3

## What is a Series?

A Pandas Series is like a column in a table.

It is a one-dimensional array holding data of any type.

```
In [6]: import pandas as pd  
a = [1,2,3,4,5,6]  
myvar = pd.Series(a)  
print(myvar)
```

```
0    1  
1    2  
2    3  
3    4  
4    5  
5    6  
dtype: int64
```

## Labels

If nothing else is specified, the values are labeled with their index number. First value has index 0, second value has index 1

etc.

This label can be used to access a specified value.

```
In [7]: # return the first values of the dataframe  
myvar[a[0]]
```

Out[7]: 2

## Create Labels

With the index argument, you can name your own labels.

```
In [8]: import pandas as pd

a = ['Himanshu', 'Virat', 'Roman']
myvar = pd.Series(a, index = ['Student', 'Cricketer', 'wrestler'])
print(myvar)
```

```
Student      Himanshu
Cricketer    Virat
wrestler     Roman
dtype: object
```

```
In [9]: # When you have created labels, you can access an item by referring to the label
```

```
In [10]: # Return the value of "Cricketer":
```

```
In [11]: print(myvar['Cricketer'])
```

```
Virat
```

## What is a DataFrame?

A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.

```
In [12]: import pandas as pd

data = {
    "calories" : [429, 454, 232],
    "Duration" : [50, 40, 30]
}
# Load data into pandas dataframe
df = pd.DataFrame(data)
print(df)
```

```
   calories  Duration
0        429         50
1        454         40
2        232         30
```

## Locate Row

As you can see from the result above, the DataFrame is like a table with rows and columns.

Pandas use the loc attribute to return one or more specified row(s)

```
In [13]: # Return row 0:
print(df.loc[0])
```

```
calories    429
Duration     50
Name: 0, dtype: int64
```

```
In [14]: # Return row 0 and 1:
print(df.loc[[0,1]])
```

```
   calories  Duration
0        429         50
1        454         40
```

## Named Indexes

With the index argument, you can name your own indexes.

```
In [15]: import pandas as pd

data = {
    'calories' : [478,453,423],
    'duration' : [40,50,30]
}
df = pd.DataFrame(data,index = ["day1","day2","day3"])
print(df)
```

```
   calories  duration
day1      478         40
day2      453         50
day3      423         30
```

## Locate Named Indexes

Use the named index in the loc attribute to return the specified row(s).

```
In [16]: print(df.loc["day1"])
```

```
calories    478
duration     40
Name: day1, dtype: int64
```

```
In [17]: print(df.loc["day2"])
```

```
calories    453
duration     50
Name: day2, dtype: int64
```

## Load Files Into a DataFrame

If your data sets are stored in a file, Pandas can load them into a DataFrame.

**load (excel file) into a DataFrame:**

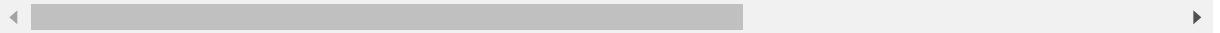
```
In [18]: df = pd.read_excel(r'E:\himanshu_2022\Download\data fsds -20230411T064049Z-001')
```

```
In [19]: df
```

```
Out[19]:
```

	Dress_ID	Style	Price	Rating	Size	Season	NeckLine	SleeveLength	waiseline	
0	1006032852	Sexy	Low	4.6	M	Summer	o-neck	sleeveless	empire	
1	1212192089	Casual	Low	0.0	L	Summer	o-neck	Petal	natural	r
2	1190380701	vintage	High	0.0	L	Autumn	o-neck	full	natural	
3	966005983	Brief	Average	4.6	L	Spring	o-neck	full	natural	
4	876339541	cute	Low	4.5	M	Summer	o-neck	butterfly	natural	chi
...	...	...	...	...	...	...	...	...	...	...
495	713391965	Casual	Low	4.7	M	Spring	o-neck	full	natural	
496	722565148	Sexy	Low	4.3	free	Summer	o-neck	full	empire	
497	532874347	Casual	Average	4.7	M	Summer	v-neck	full	empire	
498	655464934	Casual	Average	4.6	L	winter	boat-neck	sleeveless	empire	
499	919930954	Casual	Low	4.4	free	Summer	v-neck	short	empire	

500 rows × 14 columns



```
In [20]: ##### check the type of data...
```

```
In [21]: type(df)
```

```
Out[21]: pandas.core.frame.DataFrame
```

```
In [22]: ## shows only first 5 record..
df.head(5)
```

```
Out[22]:
```

	Dress_ID	Style	Price	Rating	Size	Season	NeckLine	SleeveLength	waiseline	M
0	1006032852	Sexy	Low	4.6	M	Summer	o-neck	sleeveless	empire	
1	1212192089	Casual	Low	0.0	L	Summer	o-neck	Petal	natural	mic
2	1190380701	vintage	High	0.0	L	Autumn	o-neck	full	natural	p
3	966005983	Brief	Average	4.6	L	Spring	o-neck	full	natural	
4	876339541	cute	Low	4.5	M	Summer	o-neck	butterfly	natural	chiffo



In [23]: *## shows only last 5 record...*  
`df.tail()`

Out[23]:

	Dress_ID	Style	Price	Rating	Size	Season	NeckLine	SleeveLength	waiseline	Material
<b>495</b>	713391965	Casual	Low	4.7	M	Spring	o-neck	full	natural	poly
<b>496</b>	722565148	Sexy	Low	4.3	free	Summer	o-neck	full	empire	cc
<b>497</b>	532874347	Casual	Average	4.7	M	Summer	v-neck	full	empire	cc
<b>498</b>	655464934	Casual	Average	4.6	L	winter	boat-neck	sleeveless	empire	
<b>499</b>	919930954	Casual	Low	4.4	free	Summer	v-neck	short	empire	cc

## load csv file into a DataFrame:

In [24]: `df1 = pd.read_csv(r'E:\himanshu_2022\Download\data fsds -20230411T064049Z-001')`

In [25]: `df1`

Out[25]:

	30	64	1	1.1
<b>0</b>	30	62	3	1
<b>1</b>	30	65	0	1
<b>2</b>	31	59	2	1
<b>3</b>	31	65	4	1
<b>4</b>	33	58	10	1
...	...	...	...	...
<b>300</b>	75	62	1	1
<b>301</b>	76	67	0	1
<b>302</b>	77	65	3	1
<b>303</b>	78	65	1	2
<b>304</b>	83	58	2	2

305 rows × 4 columns

In [26]: *### first record not shows column name ...*  
`df1 = pd.read_csv(r'E:\himanshu_2022\Download\data fsds -20230411T064049Z-001')`



In [27]: df1

Out[27]:

	0	1	2	3
0	30	64	1	1
1	30	62	3	1
2	30	65	0	1
3	31	59	2	1
4	31	65	4	1
...	...	...	...	...
301	75	62	1	1
302	76	67	0	1
303	77	65	3	1
304	78	65	1	2
305	83	58	2	2

306 rows × 4 columns

In [28]: *## I want to name this header column...*  
df1 = pd.read\_csv(r'E:\himanshu\_2022\Download\data fsds -20230411T064049Z-001')

In [29]: df1

Out[29]:

	Age	Age of patient at time of operation	Patient's year of operation	Number of positive axillary nodes detected	Survival status
0	30	64	1	1	NaN
1	30	62	3	1	NaN
2	30	65	0	1	NaN
3	31	59	2	1	NaN
4	31	65	4	1	NaN
...	...	...	...	...	...
301	75	62	1	1	NaN
302	76	67	0	1	NaN
303	77	65	3	1	NaN
304	78	65	1	2	NaN
305	83	58	2	2	NaN

306 rows × 5 columns

```
In [30]: # any github repo read the csv data...
df2 = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Sr
```

```
In [31]: df2
```

```
Out[31]:
```

	Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today	Direction
0	2001	0.381	-0.192	-2.624	-1.055	5.010	1.19130	0.959	Up
1	2001	0.959	0.381	-0.192	-2.624	-1.055	1.29650	1.032	Up
2	2001	1.032	0.959	0.381	-0.192	-2.624	1.41120	-0.623	Down
3	2001	-0.623	1.032	0.959	0.381	-0.192	1.27600	0.614	Up
4	2001	0.614	-0.623	1.032	0.959	0.381	1.20570	0.213	Up
...	...	...	...	...	...	...	...	...	...
1245	2005	0.422	0.252	-0.024	-0.584	-0.285	1.88850	0.043	Up
1246	2005	0.043	0.422	0.252	-0.024	-0.584	1.28581	-0.955	Down
1247	2005	-0.955	0.043	0.422	0.252	-0.024	1.54047	0.130	Up
1248	2005	0.130	-0.955	0.043	0.422	0.252	1.42236	-0.298	Down
1249	2005	-0.298	0.130	-0.955	0.043	0.422	1.38254	-0.489	Down

1250 rows × 9 columns

```
In [32]: df3 = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/B0
```

```
In [33]: df3
```

```
Out[33]:
```

	date	values.I2P	values.P2I	values.JPI
0	1996-04-01	-2.580556	-0.836444	63.3
1	1996-05-01	-2.882258	-1.972581	66.3
2	1996-06-01	-1.742778	-0.478778	59.7
3	1996-07-01	-0.397796	-1.574301	63.3
4	1996-08-01	-0.977151	-1.797312	63.8
...	...	...	...	...
112	2005-08-01	-2.144935	-1.646989	49.0
113	2005-09-01	-3.543270	-1.769889	49.9
114	2005-10-01	-4.500484	-2.309946	50.6
115	2005-11-01	-4.098556	-1.528611	57.4
116	2005-12-01	-5.797043	-2.872849	52.9

117 rows × 4 columns

```
In [34]: # Load html data....
df4 = pd.read_html('https://www.basketball-reference.com/leagues/NBA_2015_tota
```

```
In [35]: df4
```

```
Out[35]: [      B  \
0      1      Quincy Acy  PF  24  NYK  68  22  1287  152  331  ...  .784  7
9
1      2      Jordan Adams  SG  20  MEM  30  0  248  35  86  ...  .609
9
2      3      Steven Adams  C  21  OKC  70  67  1771  217  399  ...  .502  19
9
3      4      Jeff Adrien  PF  28  MIN  17  0  215  19  44  ...  .579  2
3
4      5      Arron Afflalo  SG  29  TOT  78  72  2502  375  884  ...  .843  2
7
..  ...  ...  ..  ..  ...  ..  ..  ...  ...  ...  ...  ...
...
670  490  Thaddeus Young  PF  26  TOT  76  68  2434  451  968  ...  .655  12
7
671  490  Thaddeus Young  PF  26  MIN  48  48  1605  289  641  ...  .682  7
5
672  490  Thaddeus Young  PF  26  BRK  28  20  829  162  327  ...  .606  5
2
673  491      Cody Zeller  C  22  CHO  62  45  1487  172  373  ...  .774  9
7
674  492      Tyler Zeller  C  25  BOS  82  59  1731  340  619  ...  .823  14
6

      DRB  TRB  AST  STL  BLK  TOV  PF  PTS
0      222  301   68   27   22   60  147  398
1       19   28   16   16    7   14   24   94
2      324  523   66   38   86   99  222  537
3       54   77   15    4    9    9   30   60
4      220  247  129   41    7  116  167 1035
..  ...  ...  ...  ...  ..  ...  ...  ...
670  284  411  173  124   25  117  171 1071
671  170  245  135   86   17   75  115  685
672  114  166   38   38    8   42   56  386
673  265  362  100   34   49   62  156  472
674  319  465  113   18   52   76  205  833

[675 rows x 30 columns]]
```

```
In [36]: type(df4)
```

```
Out[36]: list
```

```
In [37]: len(df4)
```


```
Out[37]: 1
```

In [38]: `df4[0]`

Out[38]:

	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	...	FT%	ORB	DRB	TRB	AST
0	1	Quincy Acy	PF	24	NYK	68	22	1287	152	331	...	.784	79	222	301	68
1	2	Jordan Adams	SG	20	MEM	30	0	248	35	86	...	.609	9	19	28	16
2	3	Steven Adams	C	21	OKC	70	67	1771	217	399	...	.502	199	324	523	66
3	4	Jeff Adrien	PF	28	MIN	17	0	215	19	44	...	.579	23	54	77	15
4	5	Arron Afflalo	SG	29	TOT	78	72	2502	375	884	...	.843	27	220	247	129
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
670	490	Thaddeus Young	PF	26	TOT	76	68	2434	451	968	...	.655	127	284	411	173
671	490	Thaddeus Young	PF	26	MIN	48	48	1605	289	641	...	.682	75	170	245	135
672	490	Thaddeus Young	PF	26	BRK	28	20	829	162	327	...	.606	52	114	166	38
673	491	Cody Zeller	C	22	CHO	62	45	1487	172	373	...	.774	97	265	362	100
674	492	Tyler Zeller	C	25	BOS	82	59	1731	340	619	...	.823	146	319	465	113

675 rows × 30 columns



json data..

In [39]: `#readf json data....`  
`df5 = pd.read_json('https://api.github.com/repos/pandas-dev/pandas/issues')`

In [40]: df5

Out[40]:

	url	repository_url	label
0	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pa
1	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pa
2	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pa
3	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pa
4	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pa
5	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pa
6	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pa
7	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pa
8	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pa
9	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pa
10	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pa
11	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pa
12	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pa
13	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pa
14	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pa
15	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pa
16	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pa

	url	repository_url	label
17	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pa dev/pa
18	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pa dev/pa
19	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pa dev/pa
20	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pa dev/pa
21	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pa dev/pa
22	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pa dev/pa
23	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pa dev/pa
24	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pa dev/pa
25	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pa dev/pa
26	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pa dev/pa
27	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pa dev/pa
28	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pa dev/pa
29	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pa dev/pa

30 rows × 30 columns

```
In [41]: js = pd.read_json('https://api.github.com/repos/pandas-dev/pandas/issues')
```

```
In [42]: js.columns
```

```
Out[42]: Index(['url', 'repository_url', 'labels_url', 'comments_url', 'events_url',  
              'html_url', 'id', 'node_id', 'number', 'title', 'user', 'labels',  
              'state', 'locked', 'assignee', 'assignees', 'milestone', 'comments',  
              'created_at', 'updated_at', 'closed_at', 'author_association',  
              'active_lock_reason', 'body', 'reactions', 'timeline_url',  
              'performed_via_github_app', 'state_reason', 'draft', 'pull_request'],  
              dtype='object')
```

```
In [43]: js['user'][0]
```

```
Out[43]: {'login': 'zbs',  
          'id': 1444551,  
          'node_id': 'MDQ6VXNlcjE0NDQ1NTE=',  
          'avatar_url': 'https://avatars.githubusercontent.com/u/1444551?v=4',  
          'gravatar_id': '',  
          'url': 'https://api.github.com/users/zbs',  
          'html_url': 'https://github.com/zbs',  
          'followers_url': 'https://api.github.com/users/zbs/followers',  
          'following_url': 'https://api.github.com/users/zbs/following{/other_user}',  
          'gists_url': 'https://api.github.com/users/zbs/gists{/gist_id}',  
          'starred_url': 'https://api.github.com/users/zbs/starred{/owner}/{/repo}',  
          'subscriptions_url': 'https://api.github.com/users/zbs/subscriptions',  
          'organizations_url': 'https://api.github.com/users/zbs/orgs',  
          'repos_url': 'https://api.github.com/users/zbs/repos',  
          'events_url': 'https://api.github.com/users/zbs/events{/privacy}',  
          'received_events_url': 'https://api.github.com/users/zbs/received_events',  
          'type': 'User',  
          'site_admin': False}
```

```
In [44]: df = pd.read_excel(r'E:\himanshu_2022\Download\data fsds -20230411T064049Z-001')
```



In [45]: df

Out[45]:

	Dress_ID	Style	Price	Rating	Size	Season	NeckLine	SleeveLength	waiseline	
0	1006032852	Sexy	Low	4.6	M	Summer	o-neck	sleeveless	empire	
1	1212192089	Casual	Low	0.0	L	Summer	o-neck	Petal	natural	r
2	1190380701	vintage	High	0.0	L	Autumn	o-neck	full	natural	
3	966005983	Brief	Average	4.6	L	Spring	o-neck	full	natural	
4	876339541	cute	Low	4.5	M	Summer	o-neck	butterfly	natural	chi
...	...	...	...	...	...	...	...	...	...	...
495	713391965	Casual	Low	4.7	M	Spring	o-neck	full	natural	
496	722565148	Sexy	Low	4.3	free	Summer	o-neck	full	empire	
497	532874347	Casual	Average	4.7	M	Summer	v-neck	full	empire	
498	655464934	Casual	Average	4.6	L	winter	boat-neck	sleeveless	empire	
499	919930954	Casual	Low	4.4	free	Summer	v-neck	short	empire	

500 rows × 14 columns



```
In [46]: # I want to convert this data into a mongodb....
# and I save it ...
df.to_json('test.json')
```

## Data Cleaning

Data cleaning means fixing bad data in your data set.

Bad data could be:

.Empty cells

.Data in wrong format

.Wrong data

.Duplicates

```
In [47]: # Load the data ...
import pandas as pd
```

In [48]: df

Out[48]:

	Dress_ID	Style	Price	Rating	Size	Season	NeckLine	SleeveLength	waiseline	
0	1006032852	Sexy	Low	4.6	M	Summer	o-neck	sleeveless	empire	
1	1212192089	Casual	Low	0.0	L	Summer	o-neck	Petal	natural	r
2	1190380701	vintage	High	0.0	L	Automn	o-neck	full	natural	
3	966005983	Brief	Average	4.6	L	Spring	o-neck	full	natural	
4	876339541	cute	Low	4.5	M	Summer	o-neck	butterfly	natural	chi
...	...	...	...	...	...	...	...	...	...	
495	713391965	Casual	Low	4.7	M	Spring	o-neck	full	natural	
496	722565148	Sexy	Low	4.3	free	Summer	o-neck	full	empire	
497	532874347	Casual	Average	4.7	M	Summer	v-neck	full	empire	
498	655464934	Casual	Average	4.6	L	winter	boat-neck	sleeveless	empire	
499	919930954	Casual	Low	4.4	free	Summer	v-neck	short	empire	

500 rows × 14 columns

In [49]: df.head()

Out[49]:

	Dress_ID	Style	Price	Rating	Size	Season	NeckLine	SleeveLength	waiseline	M
0	1006032852	Sexy	Low	4.6	M	Summer	o-neck	sleeveless	empire	
1	1212192089	Casual	Low	0.0	L	Summer	o-neck	Petal	natural	mic
2	1190380701	vintage	High	0.0	L	Automn	o-neck	full	natural	p
3	966005983	Brief	Average	4.6	L	Spring	o-neck	full	natural	
4	876339541	cute	Low	4.5	M	Summer	o-neck	butterfly	natural	chiffo

In [50]: df.tail()

Out[50]:

	Dress_ID	Style	Price	Rating	Size	Season	NeckLine	SleeveLength	waiseline	Mat
495	713391965	Casual	Low	4.7	M	Spring	o-neck	full	natural	poly
496	722565148	Sexy	Low	4.3	free	Summer	o-neck	full	empire	cc
497	532874347	Casual	Average	4.7	M	Summer	v-neck	full	empire	cc
498	655464934	Casual	Average	4.6	L	winter	boat-neck	sleeveless	empire	
499	919930954	Casual	Low	4.4	free	Summer	v-neck	short	empire	cc

```
In [51]: ## find list of all the columns....  
df.columns
```

```
Out[51]: Index(['Dress_ID', 'Style', 'Price', 'Rating', 'Size', 'Season', 'NeckLine',  
              'SleeveLength', 'waiseline', 'Material', 'FabricType', 'Decoration',  
              'Pattern Type', 'Recommendation'],  
             dtype='object')
```

```
In [52]: # see one by one columns...  
df['Dress_ID']
```

```
Out[52]: 0      1006032852  
        1      1212192089  
        2      1190380701  
        3       966005983  
        4       876339541  
        ...  
        495     713391965  
        496     722565148  
        497     532874347  
        498     655464934  
        499     919930954  
        Name: Dress_ID, Length: 500, dtype: int64
```

```
In [53]: #types of columns...  
type(df['Dress_ID'])
```

```
Out[53]: pandas.core.series.Series
```

### dataframe types of every columns...

```
In [54]: df.dtypes
```

```
Out[54]: Dress_ID      int64  
        Style        object  
        Price        object  
        Rating      float64  
        Size        object  
        Season      object  
        NeckLine    object  
        SleeveLength object  
        waiseline    object  
        Material    object  
        FabricType  object  
        Decoration  object  
        Pattern Type object  
        Recommendation int64  
        dtype: object
```

```
In [55]: df[['Dress_ID']]
```

```
Out[55]:
```

	Dress_ID
0	1006032852
1	1212192089
2	1190380701
3	966005983
4	876339541
...	...
495	713391965
496	722565148
497	532874347
498	655464934
499	919930954

500 rows × 1 columns

```
In [56]: type(df[['Dress_ID']])
```

```
Out[56]: pandas.core.frame.DataFrame
```

```
In [57]: df['Dress_ID']
```

```
Out[57]:
```

0	1006032852
1	1212192089
2	1190380701
3	966005983
4	876339541
...	...
495	713391965
496	722565148
497	532874347
498	655464934
499	919930954

Name: Dress\_ID, Length: 500, dtype: int64

```
In [58]: type(df['Dress_ID'])
```

```
Out[58]: pandas.core.series.Series
```

```
In [59]: df.columns
```

```
Out[59]: Index(['Dress_ID', 'Style', 'Price', 'Rating', 'Size', 'Season', 'NeckLine',  
               'SleeveLength', 'waiseline', 'Material', 'FabricType', 'Decoration',  
               'Pattern Type', 'Recommendation'],  
              dtype='object')
```

fetch the some columns ...

```
In [60]: df[['Dress_ID', 'Style', 'Price', 'Material']]
```

```
Out[60]:
```

	Dress_ID	Style	Price	Material
0	1006032852	Sexy	Low	NaN
1	1212192089	Casual	Low	microfiber
2	1190380701	vintage	High	polyester
3	966005983	Brief	Average	silk
4	876339541	cute	Low	chiffonfabric
...	...	...	...	...
495	713391965	Casual	Low	polyester
496	722565148	Sexy	Low	cotton
497	532874347	Casual	Average	cotton
498	655464934	Casual	Average	silk
499	919930954	Casual	Low	cotton

500 rows × 4 columns

check the statistical analysis of df...

```
In [61]: df.describe().T
```

```
Out[61]:
```

	count	mean	std	min	25%	50%	
Dress_ID	500.0	9.055417e+08	1.736190e+08	444282011.0	767316420.0	908329553.0	1.0
Rating	500.0	3.528600e+00	2.005364e+00	0.0	3.7	4.6	4.8
Recommendation	500.0	4.200000e-01	4.940528e-01	0.0	0.0	0.0	1.0

observation -> only giving numerical data analysis

```
In [62]: df.dtypes
```

```
Out[62]: Dress_ID          int64
Style          object
Price          object
Rating         float64
Size          object
Season         object
NeckLine       object
SleeveLength   object
waixeline      object
Material       object
FabricType     object
Decoration     object
Pattern Type   object
Recommendation int64
dtype: object
```

```
In [63]: ## find those columns that datatypes is object bydefault...
df.dtypes == 'object'
```

```
Out[63]: Dress_ID          False
Style          True
Price          True
Rating         False
Size          True
Season         True
NeckLine       True
SleeveLength   True
waixeline      True
Material       True
FabricType     True
Decoration     True
Pattern Type   True
Recommendation False
dtype: bool
```

```
In [64]: # those list wherever its condition is valid
df.dtypes[df.dtypes == 'object']
```

```
Out[64]: Style          object
Price          object
Size          object
Season         object
NeckLine       object
SleeveLength   object
waixeline      object
Material       object
FabricType     object
Decoration     object
Pattern Type   object
dtype: object
```

```
In [65]: #List of all the columns that datatpes columns is object ....
df.dtypes[df.dtypes == 'object'].index
```

```
Out[65]: Index(['Style', 'Price', 'Size', 'Season', 'NeckLine', 'SleeveLength',
               'waiseline', 'Material', 'FabricType', 'Pattern Type'],
              dtype='object')
```

```
In [66]: # know i filter out of all the this object datatpes columns into dataframe
df[df.dtypes[df.dtypes == 'object'].index]
```

```
Out[66]:
```

	Style	Price	Size	Season	NeckLine	SleeveLength	waiseline	Material	FabricType
0	Sexy	Low	M	Summer	o-neck	sleeveless	empire	NaN	chiffon
1	Casual	Low	L	Summer	o-neck	Petal	natural	microfiber	NaN
2	vintage	High	L	Automn	o-neck	full	natural	polyster	NaN
3	Brief	Average	L	Spring	o-neck	full	natural	silk	chiffon
4	cute	Low	M	Summer	o-neck	butterfly	natural	chiffonfabric	chiffon
...	...	...	...	...	...	...	...	...	...
495	Casual	Low	M	Spring	o-neck	full	natural	polyster	NaN
496	Sexy	Low	free	Summer	o-neck	full	empire	cotton	NaN
497	Casual	Average	M	Summer	v-neck	full	empire	cotton	NaN
498	Casual	Average	L	winter	boat-neck	sleeveless	empire	silk	broadcloth
499	Casual	Low	free	Summer	v-neck	short	empire	cotton	Corduroy

500 rows × 11 columns

```
In [67]: # know checks the statistics of these columns...
df[df.dtypes[df.dtypes == 'object'].index].describe()
```

```
Out[67]:
```

	Style	Price	Size	Season	NeckLine	SleeveLength	waiseline	Material	FabricType
count	500	498	500	498	497	498	413	372	234
unique	13	7	7	8	16	17	4	23	22
top	Casual	Average	M	Summer	o-neck	sleeveless	natural	cotton	chiffon
freq	232	252	177	159	271	223	304	152	135

```
In [68]: # types of entire dataset...
type(df.dtypes)
```

```
Out[68]: pandas.core.series.Series
```

In [69]: `df.dtypes`

```
Out[69]: Dress_ID          int64
Style          object
Price          object
Rating         float64
Size          object
Season         object
NeckLine       object
SleeveLength   object
waiseline      object
Material       object
FabricType     object
Decoration     object
Pattern Type   object
Recommendation int64
dtype: object
```

In [70]: *##list of all the columns that datatpes columns float ....*  
`df.dtypes[df.dtypes == 'float'].index`

Out[70]: Index(['Rating'], dtype='object')

In [71]: *# know checks the statistics of these columns...*  
`df[df.dtypes[df.dtypes == 'float'].index].describe()`

```
Out[71]:
```

	Rating
count	500.000000
mean	3.528600
std	2.005364
min	0.000000
25%	3.700000
50%	4.600000
75%	4.800000
max	5.000000

In [72]: `df.columns`

```
Out[72]: Index(['Dress_ID', 'Style', 'Price', 'Rating', 'Size', 'Season', 'NeckLine',
'SleeveLength', 'waiseline', 'Material', 'FabricType', 'Decoration',
'Pattern Type', 'Recommendation'],
dtype='object')
```



```
In [73]: df['Dress_ID']
```

```
Out[73]: 0      1006032852
         1      1212192089
         2      1190380701
         3       966005983
         4       876339541
         ...
        495     713391965
        496     722565148
        497     532874347
        498     655464934
        499     919930954
        Name: Dress_ID, Length: 500, dtype: int64
```

```
In [74]: # show me the data index 0 to 4....
         df['Dress_ID'][0:4]
```

```
Out[74]: 0      1006032852
         1      1212192089
         2      1190380701
         3       966005983
        Name: Dress_ID, dtype: int64
```

```
In [75]: # List jumped operation
         df['Dress_ID'][1:40:2]
```

```
Out[75]: 1      1212192089
         3       966005983
         5      1068332458
         7      1219677488
         9       985292672
        11       898481530
        13       749031896
        15      1162628131
        17       830467746
        19      1113221101
        21       856178100
        23       840516484
        25      1139843344
        27      1235426503
        29       629131530
        31      1150275464
        33       978773911
        35       640823350
        37      1060207186
        39       941190190
        Name: Dress_ID, dtype: int64
```

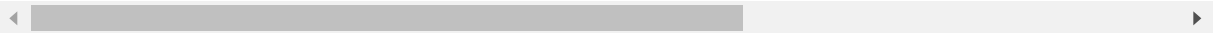
## add more columns in df...

In [76]: df

Out[76]:

	Dress_ID	Style	Price	Rating	Size	Season	NeckLine	SleeveLength	waiseline	
0	1006032852	Sexy	Low	4.6	M	Summer	o-neck	sleevless	empire	
1	1212192089	Casual	Low	0.0	L	Summer	o-neck	Petal	natural	r
2	1190380701	vintage	High	0.0	L	Automn	o-neck	full	natural	
3	966005983	Brief	Average	4.6	L	Spring	o-neck	full	natural	
4	876339541	cute	Low	4.5	M	Summer	o-neck	butterfly	natural	chi
...	...	...	...	...	...	...	...	...	...	
495	713391965	Casual	Low	4.7	M	Spring	o-neck	full	natural	
496	722565148	Sexy	Low	4.3	free	Summer	o-neck	full	empire	
497	532874347	Casual	Average	4.7	M	Summer	v-neck	full	empire	
498	655464934	Casual	Average	4.6	L	winter	boat-neck	sleevless	empire	
499	919930954	Casual	Low	4.4	free	Summer	v-neck	short	empire	

500 rows × 14 columns



In [77]: df['Style1'] = "Fantastic"

In [78]: df

Out[78]:

	Dress_ID	Style	Price	Rating	Size	Season	NeckLine	SleeveLength	waiseline	
0	1006032852	Sexy	Low	4.6	M	Summer	o-neck	sleevless	empire	
1	1212192089	Casual	Low	0.0	L	Summer	o-neck	Petal	natural	r
2	1190380701	vintage	High	0.0	L	Automn	o-neck	full	natural	
3	966005983	Brief	Average	4.6	L	Spring	o-neck	full	natural	
4	876339541	cute	Low	4.5	M	Summer	o-neck	butterfly	natural	chi
...	...	...	...	...	...	...	...	...	...	
495	713391965	Casual	Low	4.7	M	Spring	o-neck	full	natural	
496	722565148	Sexy	Low	4.3	free	Summer	o-neck	full	empire	
497	532874347	Casual	Average	4.7	M	Summer	v-neck	full	empire	
498	655464934	Casual	Average	4.6	L	winter	boat-neck	sleevless	empire	
499	919930954	Casual	Low	4.4	free	Summer	v-neck	short	empire	

500 rows × 15 columns



## check the null values...

In [79]: `df.isnull().sum()`

```
Out[79]: Dress_ID      0
Style      0
Price      2
Rating     0
Size       0
Season     2
NeckLine   3
SleeveLength 2
waiseline  87
Material   128
FabricType 266
Decoration 236
Pattern Type 109
Recommendation 0
Style1     0
dtype: int64
```

**observation** -> We have some sort of missing value in df

## Pandas DataFrame dropna() Method

In [101]: `new_df = df.dropna()`

In [102]: `new_df`

```
Out[102]:
```

	Dress_ID	Style	Price	Rating	Size	Season	NeckLine	SleeveLength	waiseline
3	966005983	Brief	Average	4.6	L	Spring	o-neck	full	natural
4	876339541	cute	Low	4.5	M	Summer	o-neck	butterfly	natural
8	1113094204	Flare	Average	0.0	free	Spring	v-neck	short	empire
10	1117293701	party	Average	5.0	free	Summer	o-neck	full	natural
28	942808364	cute	Low	4.3	free	Automn	o-neck	sleevless	natural
...	...	...	...	...	...	...	...	...	...
488	511503677	Casual	Low	4.4	M	Summer	o-neck	halfsleeve	empire
490	641665398	Casual	Low	4.8	free	winter	bowneck	full	natural
493	817353671	bohemian	Low	4.6	free	Summer	o-neck	sleevless	natural
498	655464934	Casual	Average	4.6	L	winter	boat-neck	sleevless	empire
499	919930954	Casual	Low	4.4	free	Summer	v-neck	short	empire

99 rows × 10 columns

```
In [103]: new_df.isnull().sum()
```

```
Out[103]: Dress_ID      0
          Style        0
          Price        0
          Rating       0
          Size         0
          Season       0
          NeckLine     0
          SleeveLength  0
          waiseline    0
          Material     0
          FabricType   0
          Decoration   0
          Pattern Type  0
          Recommendation 0
          Style1       0
          dtype: int64
```

***The dropna() method removes the rows that contains NULL values.***

***The dropna() method returns a new DataFrame object unless the inplace parameter is set to True, in that case the dropna() method does the removing in the original DataFrame instead.***

## Discovering Duplicates

Duplicate rows are rows that have been registered more than one time.

**To discover duplicates, we can use the duplicated() method.**

**The duplicated() method returns a Boolean values for each row:**

Returns True for every row that is a duplicate, othwerwise False:

```
In [104]: print(new_df.duplicated())
```

```
3      False
4      False
8      False
10     False
28     False
...
488    False
490    False
493    False
498    False
499    False
Length: 99, dtype: bool
```

```
In [106]: new_df.duplicated().sum()
```

```
Out[106]: 0
```

**observation -> we have no duplicated data in my new\_df .**

```
In [107]: # check the original df...  
print(df.duplicated())
```

```
3      False  
4      False  
8      False  
10     False  
28     False  
...  
488    False  
490    False  
493    False  
498    False  
499    False  
Length: 99, dtype: bool
```

```
In [108]: df.duplicated().sum()
```

```
Out[108]: 0
```

```
In [109]: new_df.columns
```

```
Out[109]: Index(['Dress_ID', 'Style', 'Price', 'Rating', 'Size', 'Season', 'NeckLine',  
                'SleeveLength', 'waiseline', 'Material', 'FabricType', 'Decoration',  
                'Pattern Type', 'Recommendation', 'Style1'],  
                dtype='object')
```

**Try out different-different query in df...**

**1. Find out those Dress\_ID which have maximum Rating point ?**

```
In [111]: df[df['Rating'] == max(df['Rating'])]
```

```
Out[111]:
```

	Dress_ID	Style	Price	Rating	Size	Season	NeckLine	SleeveLength	waiseline
10	1117293701	party	Average	5.0	free	Summer	o-neck	full	natural
40	943844640	Brief	Average	5.0	free	Spring	v-neck	sleeless	natural
65	1019134411	Casual	Average	5.0	M	Automn	v-neck	short	natural
80	1130994272	Casual	Low	5.0	M	Summer	v-neck	sleeless	empire
176	956453735	cute	low	5.0	L	Winter	o-neck	threequarter	natural
326	912614690	bohemian	Low	5.0	free	Spring	o-neck	short	natural
327	1072784739	Casual	low	5.0	free	Spring	bowneck	sleeless	natural
358	818295153	bohemian	low	5.0	free	Summer	o-neck	sleeless	empire
410	932913675	Casual	Low	5.0	free	Summer	o-neck	sleeless	natural

## 2. Find out those data that season is only summer ?

```
In [112]: df['Season']
```

```
Out[112]:
```

3	Spring
4	Summer
8	Spring
10	Summer
28	Automn
	...
488	Summer
490	winter
493	Summer
498	winter
499	Summer

Name: Season, Length: 99, dtype: object

```
In [114]: df[df['Season'] == 'Summer']
```

```
Out[114]:
```

	Dress_ID	Style	Price	Rating	Size	Season	NeckLine	SleeveLength	waiseli
4	876339541	cute	Low	4.5	M	Summer	o-neck	butterfly	natur
10	1117293701	party	Average	5.0	free	Summer	o-neck	full	natur
47	1175184188	Casual	Average	0.0	M	Summer	o-neck	halfsleeve	natur
66	1072477320	Casual	Low	0.0	free	Summer	v-neck	halfsleeve	natur
80	1130994272	Casual	Low	5.0	M	Summer	v-neck	sleevless	empi
99	805438829	Sexy	Low	4.8	L	Summer	v-neck	full	natur
127	893330898	Sexy	Low	4.6	free	Summer	v-neck	sleevless	natur
141	547555172	cute	Low	4.7	XL	Summer	o-neck	short	empi
149	1021375534	cute	low	4.5	L	Summer	o-neck	sleevless	natur
171	980295760	Casual	Low	4.8	XL	Summer	o-neck	short	empi
172	913978120	Sexy	Medium	4.6	L	Summer	o-neck	short	natur
179	1167448608	cute	low	0.0	free	Summer	peterpan-collor	short	empi
217	756620535	Casual	Low	4.6	free	Summer	o-neck	full	natur
238	1024954013	Casual	Average	4.7	L	Summer	o-neck	sleevless	natur
312	1246945687	Novelty	Average	0.0	free	Summer	o-neck	full	natur
350	754316257	Casual	Low	4.7	L	Summer	o-neck	sleevless	natur
356	944930838	vintage	low	4.0	free	Summer	turndowncollor	short	natur
358	818295153	bohemian	low	5.0	free	Summer	o-neck	sleevless	empi
370	1039384371	Casual	Average	4.8	free	Summer	o-neck	sleevless	natur
375	1069577979	Casual	Low	4.0	free	Summer	v-neck	sleevless	natur
381	845853510	Casual	Average	4.5	L	Summer	o-neck	short	natur
410	932913675	Casual	Low	5.0	free	Summer	o-neck	sleevless	natur
414	924108301	Casual	Low	4.7	L	Summer	o-neck	full	natur
417	934830377	bohemian	Low	4.6	free	Summer	v-neck	sleevless	natur
469	953168456	cute	Average	4.7	L	Summer	o-neck	sleevless	natur
488	511503677	Casual	Low	4.4	M	Summer	o-neck	halfsleeve	empi
493	817353671	bohemian	Low	4.6	free	Summer	o-neck	sleevless	natur
499	919930954	Casual	Low	4.4	free	Summer	v-neck	short	empi

3. find those columns that have solid pattern type ?

In [115]: `df['Pattern Type']`

Out[115]:

```

3      print
4      dot
8      solid
10     solid
28     striped
...
488    solid
490    solid
493    solid
498    print
499    solid
Name: Pattern Type, Length: 99, dtype: object

```

In [116]: `df[df['Pattern Type'] == 'solid']`

Out[116]:

	Dress_ID	Style	Price	Rating	Size	Season	NeckLine	SleeveLength	waiseline
8	1113094204	Flare	Average	0.0	free	Spring	v-neck	short	empire
10	1117293701	party	Average	5.0	free	Summer	o-neck	full	natural
30	851945460	Casual	Average	4.6	L	Autumn	o-neck	sleeveless	empire
31	1150275464	Casual	Low	0.0	M	Spring	o-neck	sleeveless	natural
40	943844640	Brief	Average	5.0	free	Spring	v-neck	sleeveless	natural
...	...	...	...	...	...	...	...	...	...
487	1223469038	Sexy	Average	0.0	free	winter	slash-neck	sleeveless	natural
488	511503677	Casual	Low	4.4	M	Summer	o-neck	halfsleeve	empire
490	641665398	Casual	Low	4.8	free	winter	bowneck	full	natural
493	817353671	bohemian	Low	4.6	free	Summer	o-neck	sleeveless	natural
499	919930954	Casual	Low	4.4	free	Summer	v-neck	short	empire

62 rows × 15 columns

4. find out those dress id that have Rating more than 3 ?



In [118]: `df[df['Rating'] > 3]`

Out[118]:

	Dress_ID	Style	Price	Rating	Size	Season	NeckLine	SleeveLength	waixeline
<b>3</b>	966005983	Brief	Average	4.6	L	Spring	o-neck	full	natural
<b>4</b>	876339541	cute	Low	4.5	M	Summer	o-neck	butterfly	natural
<b>10</b>	1117293701	party	Average	5.0	free	Summer	o-neck	full	natural
<b>28</b>	942808364	cute	Low	4.3	free	Autumn	o-neck	sleeveless	natural
<b>29</b>	629131530	cute	Low	4.7	M	Spring	ruffled	short	empire
...	...	...	...	...	...	...	...	...	...
<b>488</b>	511503677	Casual	Low	4.4	M	Summer	o-neck	halfsleeve	empire
<b>490</b>	641665398	Casual	Low	4.8	free	winter	bowneck	full	natural
<b>493</b>	817353671	bohemian	Low	4.6	free	Summer	o-neck	sleeveless	natural
<b>498</b>	655464934	Casual	Average	4.6	L	winter	boat-neck	sleeveless	empire
<b>499</b>	919930954	Casual	Low	4.4	free	Summer	v-neck	short	empire

75 rows × 15 columns



**5. find out those dress id that have Rating less than 1 ?**

In [121]: `df[df['Rating'] < 1]`

Out[121]:

	Dress_ID	Style	Price	Rating	Size	Season	NeckLine	SleeveLength	waiseline
8	1113094204	Flare	Average	0.0	free	Spring	v-neck	short	empire
31	1150275464	Casual	Low	0.0	M	Spring	o-neck	sleeveless	natural
47	1175184188	Casual	Average	0.0	M	Summer	o-neck	halfsleeve	natural
52	1160536550	Casual	Average	0.0	L	Spring	o-neck	halfsleeve	natural
66	1072477320	Casual	Low	0.0	free	Summer	v-neck	halfsleeve	natural
71	1210582154	sexy	Low	0.0	M	Spring	v-neck	short	natural
84	1096779299	Casual	Average	0.0	free	Winter	o-neck	halfsleeve	empire
95	1027818824	Casual	Medium	0.0	M	Spring	v-neck	threequarter	natural
108	1002440915	Casual	Medium	0.0	free	Spring	o-neck	sleeveless	natural
135	1225512606	cute	High	0.0	M	Winter	o-neck	threequarter	empire
161	1246749980	vintage	Average	0.0	S	Spring	o-neck	full	natural
165	832391864	Casual	Average	0.0	L	Winter	open	short	natural
179	1167448608	cute	low	0.0	free	Summer	peterpan-collor	short	empire
233	1216538883	cute	Average	0.0	M	Winter	o-neck	short	natural
244	659466129	Casual	Average	0.0	M	winter	o-neck	full	natural
312	1246945687	Novelty	Average	0.0	free	Summer	o-neck	full	natural
409	1135456589	Casual	Average	0.0	XL	Winter	v-neck	short	empire
430	907669618	Sexy	Average	0.0	free	Spring	v-neck	full	natural
443	1249825438	Sexy	Average	0.0	free	Autumn	o-neck	full	natural
477	1122991519	Sexy	Low	0.0	free	winter	o-neck	sleeveless	natural
479	974438263	cute	Low	0.0	free	Spring	v-neck	sleeveless	natural
481	1061890181	Casual	Average	0.0	L	Spring	o-neck	sleeveless	natural chi
486	1109819647	Casual	Average	0.0	free	winter	o-neck	short	natural
487	1223469038	Sexy	Average	0.0	free	winter	slash-neck	sleeveless	natural

In [153]: `df7 = pd.read_csv(r'E:\himanshu_2022\Download\data fsds -20230411T064049Z-001\c`

In [154]: df7

Out[154]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDER
0	10107	30	95.70	2	2871.00	2/24
1	10121	34	81.35	5	2765.90	5/7/200
2	10134	41	94.74	2	3884.34	7/1/200
3	10145	45	83.26	6	3746.70	8/25
4	10159	49	100.00	14	5205.27	10/10
...	...	...	...	...	...	...
2818	10350	20	100.00	15	2244.40	12/2
2819	10373	29	100.00	1	3978.51	1/31
2820	10386	43	100.00	4	5417.57	3/1/200
2821	10397	34	62.24	1	2116.16	3/28
2822	10414	47	65.52	9	3079.44	5/6/200

2823 rows × 25 columns



In [160]: df7.columns

Out[160]: Index(['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'ORDERLINENUMBER', 'SALES', 'ORDERDATE', 'STATUS', 'QTR\_ID', 'MONTH\_ID', 'YEAR\_ID', 'PRODUCTLINE', 'MSRP', 'PRODUCTCODE', 'CUSTOMERNAME', 'PHONE', 'ADDRESSLINE1', 'ADDRESSLINE2', 'CITY', 'STATE', 'POSTALCODE', 'COUNTRY', 'TERRITORY', 'CONTACTLASTNAME', 'CONTACTFIRSTNAME', 'DEALSIZE'], dtype='object')

## 6. find out those country that have maximum sales?

In [156]: df7[df7['SALES'] == max(df7['SALES'])]['COUNTRY']

Out[156]: 598 USA  
Name: COUNTRY, dtype: object

## 7. find out those city that have minimum sales ?

```
In [157]: df7[df7['SALES'] == min(df7['SALES'])]['COUNTRY']
```

```
Out[157]: 2249    France
          Name: COUNTRY, dtype: object
```

### 8. find out those city that have 2278 above sales ?

```
In [173]: df7[df7['SALES'] > 2278]['COUNTRY']
```

```
Out[173]: 0      USA
          1    France
          2    France
          3      USA
          4      USA
          ...
          2816  Denmark
          2817      USA
          2819  Finland
          2820    Spain
          2822      USA
          Name: COUNTRY, Length: 2053, dtype: object
```

### 9. Find out that years that have maximum sales?

```
In [167]: df7[df7['SALES'] == max(df7['SALES'])]['YEAR_ID']
```

```
Out[167]: 598    2005
          Name: YEAR_ID, dtype: int64
```

### 10. Find out that years that have minimum sales?

```
In [168]: df7[df7['SALES'] == min(df7['SALES'])]['YEAR_ID']
```

```
Out[168]: 2249    2005
          Name: YEAR_ID, dtype: int64
```

### 11. Find out the that Month\_ID that have maximum sales ?

```
In [170]: df7[df7['SALES'] == max(df7['SALES'])]['MONTH_ID']
```

```
Out[170]: 598    4
          Name: MONTH_ID, dtype: int64
```

### 12 . Find out the that Month\_ID that have minimum sales ?

```
In [172]: df7[df7['SALES'] == min(df7['SALES'])]['MONTH_ID']
```

```
Out[172]: 2249    5
          Name: MONTH_ID, dtype: int64
```

## Data Correlations

A great aspect of the Pandas module is the **corr()** method.

The **corr()** method calculates the relationship between each column in your data set.

In [175]: `df7.corr()`

C:\Users\Balodi\AppData\Local\Temp\ipykernel\_18652\2906877236.py:1: FutureWarning: The default value of `numeric_only` in `DataFrame.corr` is deprecated. In a future version, it will default to `False`. Select only valid columns or specify the value of `numeric_only` to silence this warning.

`df7.corr()`

Out[175]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER
ORDERNUMBER	1.000000	0.065543	-0.002935	-0.055550
QUANTITYORDERED	0.065543	1.000000	0.005564	-0.018397
PRICEEACH	-0.002935	0.005564	1.000000	-0.020965
ORDERLINENUMBER	-0.055550	-0.018397	-0.020965	1.000000
SALES	0.039919	0.551426	0.657841	-0.058400
QTR_ID	-0.051383	-0.035323	0.008712	0.040716
MONTH_ID	-0.039723	-0.039048	0.005152	0.034016
YEAR_ID	0.904596	0.069535	-0.005938	-0.057367
MSRP	-0.010280	0.017881	0.670625	-0.021067

Note : The **corr()** method ignores **not numeric** columns.

The Result of the **corr()** method is a table with a lot of numbers that represents how well the **relationship** is between **two columns**.

The number varies from **-1 to 1**.

1 means that there is a 1 to 1 relationship (a perfect correlation), and for this data set, each time a value went up in the first column, the other one went up as well.

0.9 is also a good relationship, and if you increase one value, the other will probably increase as well.

-0.9 would be just as good relationship as 0.9, but if you increase one value, the other will probably go down.

0.2 means NOT a good relationship, meaning that if one value goes up does not mean that the other will.

**pandas basic operation**

In [81]: `import pandas as pd`

In [82]: `data = {'name' : ['sudh','himu','krish','sunny'],  
          'salary' : [400,300,200,100],  
          'mail_id' : ['sudh@ineuron.ai','himu@ineuron.ai','krish@ineuron.ai','sunny@ineuron.ai'],  
          'addr' : ['blr','blr','blr','blr']}`

In [83]: `# convert this data into dataframe...`

In [84]: `df = pd.DataFrame(data)`

In [85]: `# Note : which list is key and pair must be the same .`

In [86]: `# the index name to create our own...  
df = pd.DataFrame(data)`

In [87]: `# extract the data...  
df.loc[5:6]`

Out[87]:

	name	salary	mail_id	addr
--	------	--------	---------	------

In [88]: `# loc takes always the name indexes.`

In [89]: `df.iloc[5:6]`

Out[89]:

	name	salary	mail_id	addr
--	------	--------	---------	------

In [90]: `# Its not given any dataset because iloc always goes after the a default indexes`

In [91]: `df.iloc[1:3]`

Out[91]:

	name	salary	mail_id	addr
1	himu	300	himu@ineuron.ai	blr
2	krish	200	krish@ineuron.ai	blr

```
In [92]: data1 = {'pf_nu' : [34,56,56,767],
                  'income_tax' : [343,454,56,45],
                  'mobile_no' : ['232','4334','4234','2342'],
                  'course' : ['datascience','bigdata','web dev','dataanalytics']
                }
```

```
In [93]: df1 = pd.DataFrame(data1)
```

```
In [94]: df1
```

Out[94]:

	pf_nu	income_tax	mobile_no	course
0	34	343	232	datascience
1	56	454	4334	bigdata
2	56	56	4234	web dev
3	767	45	2342	dataanalytics

```
In [95]: df
```

Out[95]:

	name	salary	mail_id	addr
0	sudh	400	sudh@ineuron.ai	blr
1	himu	300	himu@ineuron.ai	blr
2	krish	200	krish@ineuron.ai	blr
3	sunny	100	sunny@ineuron.ai	blr

**concat these two DataFrame**

```
In [96]: pd.concat([df,df1])
```

Out[96]:

	name	salary	mail_id	addr	pf_nu	income_tax	mobile_no	course
0	sudh	400.0	sudh@ineuron.ai	blr	NaN	NaN	NaN	NaN
1	himu	300.0	himu@ineuron.ai	blr	NaN	NaN	NaN	NaN
2	krish	200.0	krish@ineuron.ai	blr	NaN	NaN	NaN	NaN
3	sunny	100.0	sunny@ineuron.ai	blr	NaN	NaN	NaN	NaN
0	NaN	NaN	NaN	NaN	34.0	343.0	232	datascience
1	NaN	NaN	NaN	NaN	56.0	454.0	4334	bigdata
2	NaN	NaN	NaN	NaN	56.0	56.0	4234	web dev
3	NaN	NaN	NaN	NaN	767.0	45.0	2342	dataanalytics

In [97]: *#this is horizontal concatination.*

In [98]: `pd.concat([df,df1],axis=1)`

Out[98]:

	name	salary	mail_id	addr	pf_nu	income_tax	mobile_no	course
0	sudh	400	sudh@ineuron.ai	blr	34	343	232	datascience
1	himu	300	himu@ineuron.ai	blr	56	454	4334	bigdata
2	krish	200	krish@ineuron.ai	blr	56	56	4234	web dev
3	sunny	100	sunny@ineuron.ai	blr	767	45	2342	dataanalytics

```
In [99]: data2 = {'0' : [34,56,56,767],
                  '1' : [343,454,56,45],
                  '2' : ['232','4334','4234','2342'],
                  '3' : ['datascience','bigdata','web dev','dataanalytics']
                  }
```

```
In [100]: data3 = {'0' : ['sudh','himu','krish','sunny'],
                  '1' : [400,300,200,100],
                  '2' : ['sudh@ineuron.ai','himu@ineuron.ai','krish@ineuron.ai','sunny@ineuron.ai'],
                  '3' : ['blr','blr','blr','blr']
                  }
```

In [101]: `df3 = pd.DataFrame(data2)`

In [102]: `df4 = pd.DataFrame(data3)`

In [103]: `pd.concat([df3,df4])`

Out[103]:

	0	1	2	3
0	34	343	232	datascience
1	56	454	4334	bigdata
2	56	56	4234	web dev
3	767	45	2342	dataanalytics
0	sudh	400	sudh@ineuron.ai	blr
1	himu	300	himu@ineuron.ai	blr
2	krish	200	krish@ineuron.ai	blr
3	sunny	100	sunny@ineuron.ai	blr

In [104]: *# we see the dataframe in vertical way.*

## merge operation



```
In [138]: data4 = {'emp_id' : [111,222,345433,444],
                  'salary' : [234,455,200,344],
                  'Pf'      : [223243,2334,452323,232354]
                  }
```

```
In [139]: data5 = {'emp_id' : [111,222,333,444],
                  'mob_no' : [4343,343656,3434,3433],
                  'house' : [242,5645,3534,453]
                  }
```

```
In [140]: df5 = pd.DataFrame(data4)
```

```
In [141]: df5
```

Out[141]:

	emp_id	salary	Pf
0	111	234	223243
1	222	455	2334
2	345433	200	452323
3	444	344	232354

```
In [142]: df6 = pd.DataFrame(data5)
```

```
In [143]: df6
```

Out[143]:

	emp_id	mob_no	house
0	111	4343	242
1	222	343656	5645
2	333	3434	3534
3	444	3433	453

```
In [145]: pd.merge(df5,df6)
```

Out[145]:

	emp_id	salary	Pf	mob_no	house
0	111	234	223243	4343	242
1	222	455	2334	343656	5645
2	444	344	232354	3433	453

```
In [146]: # whatever columns value is same that one is merge here.
```

```
In [147]: pd.merge(df5,df6,how = 'left')
```

```
Out[147]:
```

	emp_id	salary	Pf	mob_no	house
0	111	234	223243	4343.0	242.0
1	222	455	2334	343656.0	5645.0
2	345433	200	452323	NaN	NaN
3	444	344	232354	3433.0	453.0

```
In [150]: # whatever our left df it extract every data .
```

```
In [149]: pd.merge(df5,df6,how = 'right')
```

```
Out[149]:
```

	emp_id	salary	Pf	mob_no	house
0	111	234.0	223243.0	4343	242
1	222	455.0	2334.0	343656	5645
2	333	NaN	NaN	3434	3534
3	444	344.0	232354.0	3433	453

```
In [151]: # whatever our right df it extract every data .
```

```
In [154]: pd.merge(df5,df6,how = 'right',on = 'emp_id')
```

```
Out[154]:
```

	emp_id	salary	Pf	mob_no	house
0	111	234.0	223243.0	4343	242
1	222	455.0	2334.0	343656	5645
2	333	NaN	NaN	3434	3534
3	444	344.0	232354.0	3433	453

```
In [160]: data7 = {'emp_id1' : [111,222,345433,444],
                  'salary' : [234,455,200,344],
                  'Pf' : [223243,2334,452323,232354]
                  }

data8 = {'emp_id2' : [111,222,333,444],
         'mob_no' : [4343,343656,3434,3433],
         'house' : [242,5645,3534,453]
         }
```

```
In [161]: df7 = pd.DataFrame(data7)
```

In [162]: df7

Out[162]:

	emp_id1	salary	Pf
0	111	234	223243
1	222	455	2334
2	345433	200	452323
3	444	344	232354

In [163]: df8 = pd.DataFrame(data8)

In [164]: df8

Out[164]:

	emp_id2	mob_no	house
0	111	4343	242
1	222	343656	5645
2	333	3434	3534
3	444	3433	453

In [165]: *# I have no same columns here.*  
*# there not be extract the merge operation we can see this...*

In [166]: `pd.merge(df7,df8)`

---

**MergeError**

Traceback (most recent call last)

Cell In[166], line 1

----&gt; 1 pd.merge(df7,df8)

File ~\anaconda3\lib\site-packages\pandas\core\reshape\merge.py:110, in merge(left, right, how, on, left\_on, right\_on, left\_index, right\_index, sort, suffixes, copy, indicator, validate)

```

    93 @Substitution("\nleft : DataFrame or named Series")
    94 @Appender(_merge_doc, indents=0)
    95 def merge(
    (...)
    108     validate: str | None = None,
    109 ) -> DataFrame:
--> 110     op = _MergeOperation(
    111         left,
    112         right,
    113         how=how,
    114         on=on,
    115         left_on=left_on,
    116         right_on=right_on,
    117         left_index=left_index,
    118         right_index=right_index,
    119         sort=sort,
    120         suffixes=suffixes,
    121         indicator=indicator,
    122         validate=validate,
    123     )
    124     return op.get_result(copy=copy)

```

File ~\anaconda3\lib\site-packages\pandas\core\reshape\merge.py:685, in \_MergeOperation.\_\_init\_\_(self, left, right, how, on, left\_on, right\_on, axis, left\_index, right\_index, sort, suffixes, indicator, validate)

```

    681     # stacklevel chosen to be correct when this is reached via pd.mer
    ge
    682     # (and not DataFrame.join)
    683     warnings.warn(msg, FutureWarning, stacklevel=find_stack_level())
--> 685 self.left_on, self.right_on = self._validate_left_right_on(left_on, r
    ight_on)
    687 cross_col = None
    688 if self.how == "cross":

```

File ~\anaconda3\lib\site-packages\pandas\core\reshape\merge.py:1434, in \_MergeOperation.\_validate\_left\_right\_on(self, left\_on, right\_on)

```

    1432 common_cols = left_cols.intersection(right_cols)
    1433 if len(common_cols) == 0:
--> 1434     raise MergeError(
    1435         "No common columns to perform merge on. "
    1436         f"Merge options: left_on={left_on}, "
    1437         f"right_on={right_on}, "
    1438         f"left_index={self.left_index}, "
    1439         f"right_index={self.right_index}"
    1440     )
    1441 if (
    1442     not left_cols.join(common_cols, how="inner").is_unique
    1443     or not right_cols.join(common_cols, how="inner").is_unique
    1444 ):

```

```
1445     raise MergeError(f"Data columns not unique: {repr(common_cols)}")
```

**MergeError:** No common columns to perform merge on. Merge options: left\_on=None, right\_on=None, left\_index=False, right\_index=False

In [167]: *# so how to take the merge operation here these to df let's do.*

In [169]: `pd.merge(df7,df8 , left_on = 'emp_id1',right_on = 'emp_id2',how = 'inner')`

Out[169]:

	emp_id1	salary	Pf	emp_id2	mob_no	house
0	111	234	223243	111	4343	242
1	222	455	2334	222	343656	5645
2	444	344	232354	444	3433	453

```
In [172]: data9 = {'emp_id' : [111,222,345433,444],
                  'salary' : [234,455,200,344],
                  'Pf'     : [223243,2334,452323,232354]
                  }

data10 = {'emp_id' : [111,222,333,444],
          'mob_no' : [4343,343656,3434,3433],
          'house' : [242,5645,3534,453]
          }
```

In [174]: `df9 = pd.DataFrame(data9)`

In [175]: `df9`

Out[175]:

	emp_id	salary	Pf
0	111	234	223243
1	222	455	2334
2	345433	200	452323
3	444	344	232354

In [176]: `df10 = pd.DataFrame(data10)`

In [177]: df10

Out[177]:

	emp_id	mob_no	house
0	111	4343	242
1	222	343656	5645
2	333	3434	3534
3	444	3433	453

In [ ]: