01_ How to find the version

In [ ]:
```
import pandas as pd
pd._version_
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
c:\Users\hp\Documents\VScode\Python_chilla2\02_Pandas\pandas_tips_and_tricks\tips_and_trick.ipynb Cell 2 in <cell line: 2>()
      <a href='vscode-notebook-cell:/c%3A/Users/hp/Documents/VScode/Python_chilla2/02_Pandas/pandas_tips_and_tricks/tips_and_trick.ipynb#ch0000001?line=0'>1</a> import pandas as pd
----> <a href='vscode-notebook-cell:/c%3A/Users/hp/Documents/VScode/Python_chilla2/02_Pandas/pandas_tips_and_tricks/tips_and_trick.ipynb#ch0000001?line=1'>2</a> pd._version_

File c:\Users\hp\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\__init__.py:261, in __getattr__(name)
    257     from pandas.core.arrays.sparse import SparseArray as _SparseArray
    259     return _SparseArray
--> 261 raise AttributeError(f"module 'pandas' has no attribute '{name}'")

AttributeError: module 'pandas' has no attribute '_version_'
```

In [ ]:
```
# another way
pd.show_versions()
```

```
INSTALLED VERSIONS
------------------
commit           : 4bfe3d07b4858144c219b9346329027024102ab6
python           : 3.10.5.final.0
python-bits      : 64
OS               : Windows
OS-release       : 10
Version          : 10.0.19044
machine          : AMD64
processor        : Intel64 Family 6 Model 78 Stepping 3, GenuineIntel
byteorder        : little
LC_ALL           : None
LANG             : None
LOCALE           : English_United States.1252

pandas           : 1.4.2
numpy            : 1.23.0
pytz             : 2022.1
dateutil         : 2.8.2
pip              : 22.1.2
setuptools       : 58.1.0
Cython           : None
pytest           : None
hypothesis       : None
sphinx           : None
blosc            : None
feather          : None
xlsxwriter       : None
lxml.etree       : None
html5lib         : None
pymysql          : None
psycopg2         : None
jinja2           : 3.1.2
IPython          : 8.4.0
pandas_datareader: None
bs4              : 4.11.1
bottleneck       : None
brotli           : None
fastparquet      : None
fsspec           : None
gcsfs            : None
markupsafe       : 2.1.1
matplotlib       : 3.5.2
numba            : None
numexpr          : None
odfpy            : None
openpyxl         : None
pandas_gbq       : None
pyarrow          : None
pyreadstat       : None
pyxlsb           : None
s3fs             : None
scipy            : 1.8.1
snappy           : None
sqlalchemy       : None
tables           : None
tabulate         : None
xarray           : None
xlrd             : None
```

```
xlwt            : None
zstandard       : None
```

02_ Make a dataframe

In [ ]:
```
df = pd.DataFrame({'A column': [1, 2, 3], 'B column': [4, 5, 6]})
# Arrays length must be same
df.head()
```

Out[ ]:

	A column	B column
0	1	4
1	2	5
2	3	6

In [ ]:
```
# numpy array use to create dataframe
import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
pd.DataFrame(arr)
```

Out[ ]:

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

In [ ]:
```
# numpy array use to create dataframe
pd.DataFrame(np.random.rand(5, 8), columns=list('abcdefgh'))
```

Out[ ]:

	a	b	c	d	e	f	g	h
0	0.939286	0.524361	0.581001	0.426939	0.721313	0.340628	0.040033	0.595789
1	0.768659	0.742884	0.057065	0.980454	0.631342	0.645103	0.830245	0.674773
2	0.205075	0.459884	0.475205	0.937829	0.993325	0.847359	0.623166	0.986127
3	0.962928	0.156861	0.054984	0.209913	0.733159	0.843104	0.788658	0.186113
4	0.396559	0.820173	0.028567	0.882142	0.479061	0.834604	0.790579	0.598915

3- How to rename columns

In [ ]:
```
df.rename(columns={'A column': 'a', 'B column': 'b'})
```

Out[ ]:

	a	b
0	1	4
1	2	5
2	3	6

In [ ]:
```
# second way
df.columns = ['a_a', 'b_b']
df
```

Out[ ]:

	a_a	b_b
0	1	4
1	2	5
2	3	6

In [ ]:
```
# replace character or string in column name
df.columns = df.columns.str.replace('_', '#')
df
```

Out[ ]:

	a#a	b#b
0	1	4
1	2	5
2	3	6

In [ ]:
```
# Add prefixes and suffixes to column names
df.add_prefix('col_')
```

Out[ ]:

	col_a#a	col_b#b
0	1	4
1	2	5
2	3	6

In [ ]:
```
df.add_suffix('_')
```

Out[ ]:

	a#a_	b#b_
0	1	4
1	2	5
2	3	6

In [ ]:
```
df.columns = ['col_a', 'col_b']
df
```

Out[ ]:

	col_a	col_b
0	1	4
1	2	5
2	3	6

4- USing Template data

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns

df = sns.load_dataset('tips')
df.head()
```

Out[ ]:

	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

In [ ]:
```
# summary
df.describe()

# column names
df.columns
```

Out[ ]:
```
Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size'], dtype='object')
```

In [ ]:
```

```

```
Collecting openpyxl
  Downloading openpyxl-3.0.10-py2.py3-none-any.whl (242 kB)
     ------------------------------------ 242.1/242.1 kB 875.1 kB/s eta 0:00:00
Collecting et-xmlfile
  Downloading et_xmlfile-1.1.0-py3-none-any.whl (4.7 kB)
Installing collected packages: et-xmlfile, openpyxl
Successfully installed et-xmlfile-1.1.0 openpyxl-3.0.10
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:
```
# saving dataset
df.to_csv('tips.csv')
# pip install openpyxl
df.to_excel('tips.xlsx')
# df.to
```

5- Using your own data

In [ ]:
```
import pandas as pd
# df =pd.read_csv('tips_save.csv')
# df.head()
df=pd.read_excel('tips.xlsx')
df.head()
```

Out[ ]:

	Unnamed: 0	total_bill	tip	sex	smoker	day	time	size
0	0	16.99	1.01	Female	No	Sun	Dinner	2
1	1	10.34	1.66	Male	No	Sun	Dinner	3
2	2	21.01	3.50	Male	No	Sun	Dinner	3
3	3	23.68	3.31	Male	No	Sun	Dinner	2
4	4	24.59	3.61	Female	No	Sun	Dinner	4

6-Reverse Row order

In [ ]:
```
import seaborn as sns
import pandas as pd
df = sns.load_dataset('titanic')
df.head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN

In [ ]:
```
# Every row order is reversed
df.loc[::-1].head()
# df.head(-6)
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	de
890	0	3	male	32.0	0	0	7.75	Q	Third	man	True	Na
889	1	1	male	26.0	0	0	30.00	C	First	man	True	
888	0	3	female	NaN	1	2	23.45	S	Third	woman	False	Na
887	1	1	female	19.0	0	0	30.00	S	First	woman	False	
886	0	2	male	27.0	0	0	13.00	S	Second	man	True	Na

In [ ]:
```
df.loc[::-1].reset_index(drop=True).head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
0	0	3	male	32.0	0	0	7.75	Q	Third	man	True	NaN
1	1	1	male	26.0	0	0	30.00	C	First	man	True	C
2	0	3	female	NaN	1	2	23.45	S	Third	woman	False	NaN
3	1	1	female	19.0	0	0	30.00	S	First	woman	False	B
4	0	2	male	27.0	0	0	13.00	S	Second	man	True	NaN

7- Reverse Column order

In [ ]:
```
df.loc[:, ::-1].head()
```

Out[ ]:

	alone	alive	embark_town	deck	adult_male	who	class	embarked	fare	parch	sibsp	a
0	False	no	Southampton	NaN	True	man	Third	S	7.2500	0	1	22
1	False	yes	Cherbourg	C	False	woman	First	C	71.2833	0	1	38
2	True	yes	Southampton	NaN	False	woman	Third	S	7.9250	0	0	26
3	False	yes	Southampton	C	False	woman	First	S	53.1000	0	1	35
4	True	no	Southampton	NaN	True	man	Third	S	8.0500	0	0	35

8-Select a column by dtype

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

Out[ ]:
```
survived          int64
pclass            int64
sex              object
age             float64
sibsp             int64
parch             int64
fare            float64
embarked         object
class          category
who              object
adult_male         bool
deck           category
embark_town      object
alive            object
alone              bool
dtype: object
```

In [ ]:
```
# only select those columns which are numeric type
df.select_dtypes(include=['number']).head()
```

Out[ ]:

	survived	pclass	age	sibsp	parch	fare
0	0	3	22.0	1	0	7.2500
1	1	1	38.0	1	0	71.2833
2	1	3	26.0	0	0	7.9250
3	1	1	35.0	1	0	53.1000
4	0	3	35.0	0	0	8.0500

In [ ]:
```
# only select those columns which are numeric and float having object data type
df.select_dtypes(include=['object']).head()
```

Out[ ]:

	sex	embarked	who	embark_town	alive
0	male	S	man	Southampton	no
1	female	C	woman	Cherbourg	yes
2	female	S	woman	Southampton	yes
3	female	S	woman	Southampton	yes
4	male	S	man	Southampton	no

In [ ]:
```
# only select those have multiple type
df.select_dtypes(include=['object', 'category','number']).head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	deck	embark_tov
0	0	3	male	22.0	1	0	7.2500	S	Third	man	NaN	Southampt
1	1	1	female	38.0	1	0	71.2833	C	First	woman	C	Cherbou
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	NaN	Southampt
3	1	1	female	35.0	1	0	53.1000	S	First	woman	C	Southampt
4	0	3	male	35.0	0	0	8.0500	S	Third	man	NaN	Southampt

In [ ]:
```
# exclude those columns which are numeric type from dataframe
df.select_dtypes(exclude=['number']).head()
```

Out[ ]:

	sex	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	male	S	Third	man	True	NaN	Southampton	no	False
1	female	C	First	woman	False	C	Cherbourg	yes	False
2	female	S	Third	woman	False	NaN	Southampton	yes	True
3	female	S	First	woman	False	C	Southampton	yes	False
4	male	S	Third	man	True	NaN	Southampton	no	True

9- Convert Strings to numbers

In [ ]:
```
df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6], 'C': [7, 8, 9]})
df
# This data is numeric
```

Out[ ]:

	A	B	C
0	1	4	7
1	2	5	8
2	3	6	9

In [ ]:
```
df = pd.DataFrame({'A': ['1', '2', '3'], 'B': ['4', '5', '6'], 'C': ['7', '8', '9']})
df
```

Out[ ]:

	A	B	C
0	1	4	7
1	2	5	8
2	3	6	9

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

Out[ ]:
```
A    object
B    object
C    object
dtype: object
```

In [ ]:
```
df.astype({'A': 'int64', 'B': 'int64', 'C': 'int64'}).dtypes
```

Out[ ]:
```
A    int64
B    int64
C    int64
dtype: object
```

In [ ]:
```
# convert float to numeric, Type casting
pd.to_numeric(df['A'], errors='coerce')
```

Out[ ]:
```
0    1
1    2
2    3
Name: A, dtype: int64
```

10- Reduce Data frame size

In [ ]:
```
# Often it is diffucult to load data due to large size of data
df = sns.load_dataset('titanic')
df.shape
```

Out[ ]:
```
(891, 15)
```

In [ ]:
```
df.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 313.7 KB
```

In [ ]:
```
# In this way we reduce colums
df.sample(frac=0.1).shape
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

11- Copy data from clipboard

In [ ]:
```
# dataset download
import seaborn as sns
import pandas as pd

df = sns.load_dataset('titanic')
df.to_excel('kashti.xlsx')
```

In [ ]:
```
# read clipboard in python
df= pd.read_clipboard()
df
df.to_csv('copied_data.csv')
```

Split dataframe in 2 subsets

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

df = sns.load_dataset('titanic')
df.head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN

In [ ]:
```
len(df)
```

Out[ ]:
```
891
```

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

Out[ ]:
```
(891, 15)
```

In [ ]:
```
kashti_1=df.sample(frac=0.5, random_state=1)
kashti_1.shape
```

Out[ ]:
```
(446, 15)
```

In [ ]:
```
kashti_2=df.drop(kashti_1.index) # drop the sample data from main dataframe
kashti_2.shape
```

Out[ ]:
```
(445, 15)
```

In [ ]:
```
# kashti_1.head()
# kashti_2.head()
```

In [ ]:
```
len(kashti_1) + len(kashti_2)
```

Out[ ]:
```
891
```

13- Joining 2 datasets

In [ ]:
```
```

In [ ]:
```
# Join two dataframe
df1 = kashti_1.append(kashti_2)
df1.shape
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_9548\3891973634.py:2: FutureWarning: The fra
me.append method is deprecated and will be removed from pandas in a future version. U
se pandas.concat instead.
  df1 = kashti_1.append(kashti_2)
```

Out[ ]:
```
(891, 15)
```

14-Filtering datasets

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

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN

In [ ]:
```
df.sex.unique()
```

Out[ ]:
```
array(['male', 'female'], dtype=object)
```

In [ ]:
```
df[(df.sex=="female")]
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
8	1	3	female	27.0	0	2	11.1333	S	Third	woman	False
9	1	2	female	14.0	1	0	30.0708	C	Second	child	False
...	...	...	...	...	...	...	...	...	...	...	...
880	1	2	female	25.0	0	1	26.0000	S	Second	woman	False
882	0	3	female	22.0	0	0	10.5167	S	Third	woman	False
885	0	3	female	39.0	0	5	29.1250	Q	Third	woman	False
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False

314 rows × 15 columns

In [ ]:
```
df.embark_town.unique()
df[(df.embark_town=='Southampton')]
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True
6	0	1	male	54.0	0	0	51.8625	S	First	man	True
...	...	...	...	...	...	...	...	...	...	...	...
883	0	2	male	28.0	0	0	10.5000	S	Second	man	True
884	0	3	male	25.0	0	0	7.0500	S	Third	man	True
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False

644 rows × 15 columns

In [ ]:
```
df[(df.embark_town=='Southampton') |
    (df.embark_town=='Quennstown') &
    (df.sex=="female")]
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True
6	0	1	male	54.0	0	0	51.8625	S	First	man	True
...	...	...	...	...	...	...	...	...	...	...	...
883	0	2	male	28.0	0	0	10.5000	S	Second	man	True
884	0	3	male	25.0	0	0	7.0500	S	Third	man	True
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False

644 rows × 15 columns

In [ ]:
```
df[df.embark_town.isin(['Queenstown'])].head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	dec
5	0	3	male	NaN	0	0	8.4583	Q	Third	man	True	NaN
16	0	3	male	2.0	4	1	29.1250	Q	Third	child	False	NaN
22	1	3	female	15.0	0	0	8.0292	Q	Third	child	False	NaN
28	1	3	female	NaN	0	0	7.8792	Q	Third	woman	False	NaN
32	1	3	female	NaN	0	0	7.7500	Q	Third	woman	False	NaN

In [ ]:
```
# age column more than 30
df[df.age>30].shape
```

Out[ ]:
```
(305, 15)
```

15- Filtering by large categories

In [ ]:
```
df.embark_town.value_counts()
```

Out[ ]:
```
Southampton    644
Cherbourg      168
Queenstown      77
Name: embark_town, dtype: int64
```

In [ ]:
```
df.age.value_counts().nlargest(3)
```

Out[ ]:
```
24.0    30
22.0    27
18.0    26
Name: age, dtype: int64
```

In [ ]:
```
counts = df.who.value_counts()
counts.nlargest(3)
```

Out[ ]:
```
man      537
woman    271
child     83
Name: who, dtype: int64
```

In [ ]:
```
# show those who are most frequent in who column.
df[df.who.isin(counts.nlargest(1).index)].head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN

16- Splitting a string in multiple columns

In [ ]:
```
#import libraries
import pandas as pd

df = pd.DataFrame({'name':['Ali muaaz', 'Ahmed muaaz', 'Nabeel muaaz', 'sajjad muaaz'
                   'location':['Lahore Pakistan', 'karachi Pakistan', 'peshawar Pakis
df
```

Out[ ]:

	name	location
0	Ali muaaz	Lahore Pakistan
1	Ahmed muaaz	karachi Pakistan
2	Nabeel muaaz	peshawar Pakistan
3	sajjad muaaz	Sargodha Pakistan

In [ ]:
```
df.name.str.split(' ', expand=True)
```

Out[ ]:

	0	1
0	Ali	muaaz
1	Ahmed	muaaz
2	Nabeel	muaaz
3	sajjad	muaaz

In [ ]:
```
# Adding those splits into new columns
df[["first_name", "last_name"]]= df.name.str.split(' ', expand=True)
df
```

Out[ ]:

	name	location	first_name	last_name	city	country
0	Ali muaaz	Lahore Pakistan	Ali	muaaz	Lahore	Pakistan
1	Ahmed muaaz	karachi Pakistan	Ahmed	muaaz	karachi	Pakistan
2	Nabeel muaaz	peshawar Pakistan	Nabeel	muaaz	peshawar	Pakistan
3	sajjad muaaz	Sargodha Pakistan	sajjad	muaaz	Sargodha	Pakistan

In [ ]:
```
#location split
df.location.str.split(' ', expand=True)
```

Out[ ]:

	0	1
0	Lahore	Pakistan
1	karachi	Pakistan
2	peshawar	Pakistan
3	Sargodha	Pakistan

In [ ]:
```
df[["city", "country"]] = df.location.str.split(' ', expand=True)
df
```

Out[ ]:

	name	location	first_name	last_name	city	country
0	Ali muaaz	Lahore Pakistan	Ali	muaaz	Lahore	Pakistan
1	Ahmed muaaz	karachi Pakistan	Ahmed	muaaz	karachi	Pakistan
2	Nabeel muaaz	peshawar Pakistan	Nabeel	muaaz	peshawar	Pakistan
3	sajjad muaaz	Sargodha Pakistan	sajjad	muaaz	Sargodha	Pakistan

In [ ]:
```
# Refine data manipulation
df = df[['first_name', 'last_name', 'city', 'country']]
df
```

Out[ ]:

	first_name	last_name	city	country
0	Ali	muaaz	Lahore	Pakistan
1	Ahmed	muaaz	karachi	Pakistan
2	Nabeel	muaaz	peshawar	Pakistan
3	sajjad	muaaz	Sargodha	Pakistan

17- Aggregate by multiple groups/functions

In [ ]:
```
# Libraries
import pandas as pd
import seaborn as sns

# import dataset
df = sns.load_dataset("titanic")
```

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

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN

In [ ]:
```
df.groupby('who').count()
```

Out[ ]:

who	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	adult_male	deck	embark
child	83	83	83	83	83	83	83	83	83	83	13	
man	537	537	537	413	537	537	537	537	537	537	99	
woman	271	271	271	218	271	271	271	269	271	271	91	

In [ ]:
```
df.groupby('sex').count()
```

Out[ ]:

sex	survived	pclass	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark
female	314	314	261	314	314	314	312	314	314	314	97	
male	577	577	453	577	577	577	577	577	577	577	106	

In [ ]:
```
len(df.groupby('fare').count())
```

Out[ ]:
```
248
```

In [ ]:
```
df.groupby(['sex','pclass', 'adult_male']).count()
```

Out[ ]:

			survived	age	sibsp	parch	fare	embarked	class	who	deck	embark
sex	**pclass**	**adult_male**										
female	**1**	**False**	94	85	94	94	94	92	94	94	81	
	2	**False**	76	74	76	76	76	76	76	76	10	
	3	**False**	144	102	144	144	144	144	144	144	6	
male	**1**	**False**	3	3	3	3	3	3	3	3	3	
		True	119	98	119	119	119	119	119	119	91	
	2	**False**	9	9	9	9	9	9	9	9	3	
		True	99	90	99	99	99	99	99	99	3	
	3	**False**	28	28	28	28	28	28	28	28	1	
		True	319	225	319	319	319	319	319	319	5	

In [ ]:
```
# select columns
df[['sex','class']]
```

Out[ ]:

	sex	class
0	male	Third
1	female	First
2	female	Third
3	female	First
4	male	Third
...	...	...
886	male	Second
887	female	First
888	female	Third
889	male	First
890	male	Third

891 rows × 2 columns

In [ ]:
```
# If you want to select the rows of any data frame and theri different ways
df.describe()
```

Out[ ]:

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [ ]:
```
df.describe().loc[['min','25%','50%', '75%', 'max']]
```

Out[ ]:

	survived	pclass	age	sibsp	parch	fare
min	0.0	1.0	0.420	0.0	0.0	0.0000
25%	0.0	2.0	20.125	0.0	0.0	7.9104
50%	0.0	3.0	28.000	0.0	0.0	14.4542
75%	1.0	3.0	38.000	1.0	0.0	31.0000
max	1.0	3.0	80.000	8.0	6.0	512.3292

In [ ]:
```
df.describe().loc['min':'max']
```

Out[ ]:

	survived	pclass	age	sibsp	parch	fare
min	0.0	1.0	0.420	0.0	0.0	0.0000
25%	0.0	2.0	20.125	0.0	0.0	7.9104
50%	0.0	3.0	28.000	0.0	0.0	14.4542
75%	1.0	3.0	38.000	1.0	0.0	31.0000
max	1.0	3.0	80.000	8.0	6.0	512.3292

In [ ]:
```
# df.describe().loc['min':'max', ['survived','age']]
df.describe().loc['min':'max', 'survived':'age']
```

Out[ ]:

	survived	pclass	age
min	0.0	1.0	0.420
25%	0.0	2.0	20.125
50%	0.0	3.0	28.000
75%	1.0	3.0	38.000
max	1.0	3.0	80.000

19- Reshape Multi Index Series

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

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN

In [ ]:
```
df.survived.mean()
```

Out[ ]:
```
0.3838383838383838
```

In [ ]:
```
df.groupby('sex').survived.mean()
```

Out[ ]:
```
sex
female    0.742038
male      0.188908
Name: survived, dtype: float64
```

In [ ]:
```
df.groupby(['sex','pclass']).survived.mean()
```

Out[ ]:
```
sex     pclass
female  1         0.968085
        2         0.921053
        3         0.500000
male    1         0.368852
        2         0.157407
        3         0.135447
Name: survived, dtype: float64
```

20- Continuous to catagorical data conversion

In [ ]:
```
df.age.head()
```

Out[ ]:
```
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
Name: age, dtype: float64
```

In [ ]:
```
pd.cut(df.age, bins = [0,18, 25, 99], labels=['child', 'yound_adult', 'adult']).head()
df['new_age'] = pd.cut(df.age, bins = [0,18,25,99], labels=['child', 'yound_adult', 'a
df.head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN

21- Convert one set of values into another one

In [ ]:
```
df.sex.head()
```

Out[ ]:
```
0      male
1    female
2    female
3    female
4      male
Name: sex, dtype: object
```

In [ ]:
```
# but if we replace male female with numbers.
df['sex']= df.sex.map({'male':0, 'female':1})
df.head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
0	0	3	0	22.0	1	0	7.2500	S	Third	man	True	NaN	S
1	1	1	1	38.0	1	0	71.2833	C	First	woman	False	C	
2	1	3	1	26.0	0	0	7.9250	S	Third	woman	False	NaN	S
3	1	1	1	35.0	1	0	53.1000	S	First	woman	False	C	S
4	0	3	0	35.0	0	0	8.0500	S	Third	man	True	NaN	S

In [ ]:
```
df.embarked.unique()
```

Out[ ]:
```
array(['S', 'C', 'Q', nan], dtype=object)
```

In [ ]:
```
df['embark_town'] = df.embarked.factorize()[0]
df.head(15)
# Hot encoding
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
0	0	3	0	22.0	1	0	7.2500	S	Third	man	True	NaN
1	1	1	1	38.0	1	0	71.2833	C	First	woman	False	C
2	1	3	1	26.0	0	0	7.9250	S	Third	woman	False	NaN
3	1	1	1	35.0	1	0	53.1000	S	First	woman	False	C
4	0	3	0	35.0	0	0	8.0500	S	Third	man	True	NaN
5	0	3	0	NaN	0	0	8.4583	Q	Third	man	True	NaN
6	0	1	0	54.0	0	0	51.8625	S	First	man	True	E
7	0	3	0	2.0	3	1	21.0750	S	Third	child	False	NaN
8	1	3	1	27.0	0	2	11.1333	S	Third	woman	False	NaN
9	1	2	1	14.0	1	0	30.0708	C	Second	child	False	NaN
10	1	3	1	4.0	1	1	16.7000	S	Third	child	False	G
11	1	1	1	58.0	0	0	26.5500	S	First	woman	False	C
12	0	3	0	20.0	0	0	8.0500	S	Third	man	True	NaN
13	0	3	0	39.0	1	5	31.2750	S	Third	man	True	NaN
14	0	3	1	14.0	0	0	7.8542	S	Third	child	False	NaN

22- Transpose a wide dataframe

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

In [ ]:
```
# create a new df
df = pd.DataFrame(np.random.rand(200,25), columns= list('abcdefghijklmnopqrstuvwxy'))
df
```

Out[ ]:

	a	b	c	d	e	f	g	h	i	
0	0.211724	0.745927	0.821923	0.492195	0.190694	0.958568	0.823038	0.950209	0.444938	0.27104
1	0.047412	0.221082	0.232977	0.397836	0.320086	0.272763	0.828997	0.161886	0.295566	0.17650
2	0.867057	0.853779	0.625592	0.121421	0.523719	0.198992	0.458375	0.957293	0.085888	0.26044
3	0.132461	0.461769	0.854432	0.042779	0.805111	0.951088	0.435323	0.650683	0.501593	0.91106
4	0.089483	0.528960	0.920848	0.156145	0.701578	0.296688	0.200747	0.659971	0.895080	0.22017
...	...	...	...	...	...	...	...	...	...	
195	0.577903	0.044732	0.354547	0.523064	0.532930	0.887298	0.591463	0.544589	0.618525	0.66362
196	0.724661	0.456966	0.754921	0.785628	0.485415	0.180245	0.769119	0.729093	0.345096	0.45681
197	0.287439	0.427786	0.281038	0.556450	0.243235	0.996919	0.909025	0.646672	0.253292	0.43451
198	0.920099	0.636696	0.223891	0.419252	0.160736	0.738045	0.482875	0.537713	0.169820	0.11618
199	0.927731	0.795743	0.430028	0.708235	0.838917	0.894741	0.775171	0.600944	0.346875	0.64115

200 rows × 25 columns

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

Out[ ]:

	0	1	2	3	4	5	6	7	8	9
a	0.211724	0.047412	0.867057	0.132461	0.089483	0.579728	0.271109	0.287010	0.170414	0.264592
b	0.745927	0.221082	0.853779	0.461769	0.528960	0.576501	0.981834	0.626219	0.813389	0.501426
c	0.821923	0.232977	0.625592	0.854432	0.920848	0.740654	0.761948	0.126769	0.685673	0.469585
d	0.492195	0.397836	0.121421	0.042779	0.156145	0.900932	0.910711	0.135705	0.555520	0.302107
e	0.190694	0.320086	0.523719	0.805111	0.701578	0.096471	0.209308	0.466341	0.981947	0.469286
f	0.958568	0.272763	0.198992	0.951088	0.296688	0.717802	0.514057	0.062708	0.329392	0.524902
g	0.823038	0.828997	0.458375	0.435323	0.200747	0.762162	0.300667	0.267055	0.876633	0.806150
h	0.950209	0.161886	0.957293	0.650683	0.659971	0.844293	0.599332	0.341070	0.973228	0.827194
i	0.444938	0.295566	0.085888	0.501593	0.895080	0.104510	0.866510	0.489076	0.400820	0.625729
j	0.271046	0.176502	0.260441	0.911068	0.220177	0.527602	0.510318	0.569477	0.873050	0.488237
k	0.256652	0.639552	0.648676	0.414696	0.492947	0.112816	0.300013	0.947373	0.603913	0.706254
l	0.536588	0.318569	0.783270	0.491752	0.547374	0.625142	0.125772	0.262164	0.724501	0.941607
m	0.300326	0.004390	0.540713	0.280268	0.438407	0.389160	0.759846	0.677078	0.785851	0.295068
n	0.710437	0.111049	0.707424	0.671645	0.607314	0.163958	0.135356	0.580775	0.863109	0.342308
o	0.205949	0.775872	0.775863	0.119185	0.108076	0.652830	0.846240	0.140891	0.322013	0.564612
p	0.718807	0.572366	0.429120	0.254341	0.901657	0.061431	0.675961	0.901006	0.593889	0.485498
q	0.304762	0.391882	0.685937	0.955214	0.482364	0.207413	0.632755	0.388737	0.181044	0.837802
r	0.982934	0.938575	0.999262	0.483788	0.317964	0.269982	0.802576	0.901642	0.770138	0.487349
s	0.681947	0.107403	0.403125	0.473705	0.134215	0.987216	0.333710	0.457472	0.200810	0.572059
t	0.216433	0.450130	0.666585	0.196001	0.063959	0.607980	0.877729	0.759958	0.270377	0.135281
u	0.390709	0.591889	0.372798	0.394741	0.235880	0.821428	0.118445	0.721942	0.223452	0.607073
v	0.229121	0.855404	0.758776	0.633490	0.744447	0.862375	0.023980	0.230422	0.109529	0.847887
w	0.371542	0.031970	0.892928	0.790133	0.810398	0.370669	0.040634	0.986483	0.940028	0.510853
x	0.355820	0.503347	0.214599	0.498747	0.229281	0.217128	0.037173	0.618490	0.265773	0.980178
y	0.714223	0.916559	0.009413	0.938746	0.222225	0.084317	0.753409	0.559744	0.170307	0.567152

23- Reshaping a dataframe

In [ ]:
```
fasla = pd.DataFrame([['1234',100,200,300], ['1244',500,400,300], ['1244',500,400,300]
fasla.head()
```

Out[ ]:

	zip	factory	warehouse	retail
0	1234	100	200	300
1	1244	500	400	300
2	1244	500	400	300

In [ ]:
```
fasla.head().T
# But we want to reshape data in other way, we want our column to be long formate inst
```

Out[ ]:

	0	1	2
zip	1234	1244	1244
factory	100	500	500
warehouse	200	400	400
retail	300	300	300

In [ ]:
```
fasla2 = pd.DataFrame([[1, '123', 'factory'], [2, '124', 'warehouse'], [3, '125', 'ret
fasla2.head()
```

Out[ ]:

	id	zip	type
0	1	123	factory
1	2	124	warehouse
2	3	125	retail

In [ ]:
```
# fasla.melt(id_vars=['zip'], value_vars=['factory', 'warehouse', 'retail'], var_name=

fasla_long = fasla.melt(id_vars='zip', var_name='type', value_name='distance')
# numeric variables and categorical variables are easily separated. After this separai
```

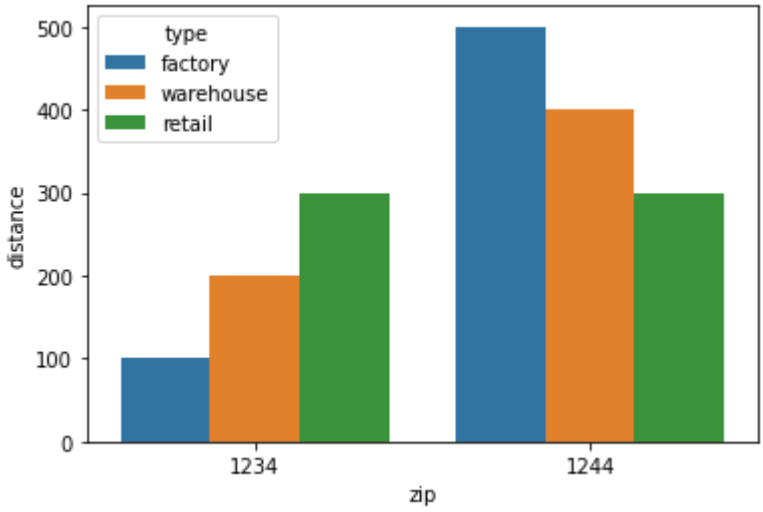
In [ ]:
```
 fasla_long.head()
```

Out[ ]:

	zip	type	distance
0	1234	factory	100
1	1244	factory	500
2	1244	factory	500
3	1234	warehouse	200
4	1244	warehouse	400

In [ ]:
```
import seaborn as sns
sns.barplot(x='zip', y='distance', hue='type', data=fasla_long)
```

Out[ ]:
```
<AxesSubplot:xlabel='zip', ylabel='distance'>
```


type
factory
warehouse
retail
500
400
300
200
100
0
distance
1234
1244
zip

In [ ]: