In [ ]:
```
================== EDA ==========================
1) Pandas : Data frame applications

2) Numpy  : Numerical python All math applications

3) Matplotlib: used for plotting

4) Seaborn   : used for plot

5) plotly    : use for plot

6) Bokhe     : plot

=================== Machine learning ==================
7) Sickit-learn: sklearn

===================== Deep Learning ==================
8) Tensorflow : Google

9) Keras      : Google

10) Pytorch   : FaceBook

====================== NLP ============================
11) NLTK    : Natural langague tool kit

12) Scipy

================ BERT ============================
13) Transformers : Hugging face (Google)

============ Google models ======================
14) Specific google packages

============== Azure ML=========================
15) Azure packages

============= GenAI ==============================
16) Gemini : Google

17) ChatGPT : OpenAI

====================== LangChain==================
18) LangChain packages
```

Create a dataframe using list

In [ ]:
```
- list === dictionary

- dictionary === list
```

$Step - 1$:

Create a dataframe

In [1]:
```
import pandas as pd
names=['Ayush','Avinash','Akash']
```

```
pd.DataFrame() # empty dataframe no rows and no columns
```

Out[1]: —

$Step - 2$:

Provide the data

In [2]:
```
names=['Ayush','Avinash','Akash']
pd.DataFrame(names)
```

Out[2]:

	0
0	Ayush
1	Avinash
2	Akash

In [4]:
```
names=['Ayush','Avinash','Akash']
age=[25,30,35]
pd.DataFrame(zip(names,age))
# what is the first postitional argument: data
# second postional argument : index
```

Out[4]:

	0	1
0	Ayush	25
1	Avinash	30
2	Akash	35

In [5]:
```
names=['Ayush','Avinash','Akash']
age=[25,30,35]
city=['hyd','blr','chennai']
pd.DataFrame(zip(names,age,city))
```

Out[5]:

	0	1	2
0	Ayush	25	hyd
1	Avinash	30	blr
2	Akash	35	chennai

In [7]:
```
names=['Ayush','Avinash','Akash']
age=[25,30,35]
city=['hyd','blr','chennai']
data=zip(names,age)
pd.DataFrame(data,city)
```

Out[7]:

	0	1
hyd	Ayush	25
blr	Avinash	30
chennai	Akash	35

$Step - 3$:

Provide the column names

In [11]:
```
names=['Ayush','Avinash','Akash']
age=[25,30,35]
pd.DataFrame(zip(names,age),
             columns=['Names','Age'])
```

Out[11]:

	Names	Age
0	Ayush	25
1	Avinash	30
2	Akash	35

In [13]:
```
names=['Ayush','Avinash','Akash']
age=[25,30,35]
city=['hyd','blr','chennai']

data=zip(names,age,city)
cols=['Names','Age','City']
df=pd.DataFrame(data,columns=cols)
```

In [14]:
```
type(df)
```

Out[14]:
```
pandas.core.frame.DataFrame
```

$Step - 4$

ADD new column

- If you want to add new column
- First you need to check the number of rows in already existed data
- For example in above the dataframe has 3 rows
- You need to create a new list with 3 rows
- that list equate to the dataframe

In [16]:
```
job=['DS','DE','CS']
df['JOB']=job
df
```

Out[16]:

	Names	Age	City	JOB
0	Ayush	25	hyd	DS
1	Avinash	30	blr	DE
2	Akash	35	chennai	CS

In [17]:
```
import pandas as pd
names=['Ayush','Avinash','Akash']
age=[25,30,35]
city=['hyd','blr','chennai']

data=zip(names,age,city)
cols=['Names','Age','City']
df=pd.DataFrame(data,columns=cols)
df
```

Out[17]:

	Names	Age	City
0	Ayush	25	hyd
1	Avinash	30	blr
2	Akash	35	chennai

In [18]:
```
job=['DS','DE','CS']
df['JOB']=job
df
```

Out[18]:

	Names	Age	City	JOB
0	Ayush	25	hyd	DS
1	Avinash	30	blr	DE
2	Akash	35	chennai	CS

$Step - 5$

Update the already Existed Column

- Createv new column and update new column both are same way

In [21]:
```
df['Age']=[35,40,45]
df
```

Out[21]:

	Names	Age	City	JOB
0	Ayush	35	hyd	DS
1	Avinash	40	blr	DE
2	Akash	45	chennai	CS

$Step - 6$

Change the index

In [25]:
```
import pandas as pd
names=['Ayush','Avinash','Akash','Anvi']
age=[25,30,35,40]
city=['hyd','blr','chennai','Mumbai']

data=zip(names,age,city)
cols=['Names','Age','City']
idx=['A','B','C','D']
df=pd.DataFrame(data,
                index=idx,
                columns=cols)
df
```

Out[25]:

	Names	Age	City
A	Ayush	25	hyd
B	Avinash	30	blr
C	Akash	35	chennai
D	Anvi	40	Mumbai

Note

- Number of list equals to **Number of columns**
- Number of values inside list equals to **Number of rows**
- When you create the data all the lists will be in same length

In [26]:
```
import pandas as pd
names=['Ayush','Avinash','Akash','Anvi']
age=[25,30,35]
city=['hyd','blr','chennai','Mumbai']

data=zip(names,age,city)
cols=['Names','Age','City']
idx=['A','B','C','D']
df=pd.DataFrame(data,
                index=idx,
                columns=cols)
df
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[26], line 9
      7 cols=['Names','Age','City']
      8 idx=['A','B','C','D']
----> 9 df=pd.DataFrame(data,
     10                  index=idx,
     11                  columns=cols)
     12 df

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:814, in DataFrame.__init_
_(self, data, index, columns, dtype, copy)
    805         columns = ensure_index(columns)
    806     arrays, columns, index = nested_data_to_arrays(
    807         # error: Argument 3 to "nested_data_to_arrays" has incompatible
    808         # type "Optional[Collection[Any]]"; expected "Optional[Index]"
   (...)
    812         dtype,
    813     )
--> 814     mgr = arrays_to_mgr(
    815         arrays,
    816         columns,
    817         index,
    818         dtype=dtype,
    819         typ=manager,
    820     )
    821 else:
    822     mgr = ndarray_to_mgr(
    823         data,
    824         index,
   (...)
    828         typ=manager,
    829     )

File ~\anaconda3\Lib\site-packages\pandas\core\internals\construction.py:119, in
arrays_to_mgr(arrays, columns, index, dtype, verify_integrity, typ, consolidate)
    116         index = ensure_index(index)
    118     # don't force copy because getting jammed in an ndarray anyway
--> 119     arrays, refs = _homogenize(arrays, index, dtype)
    120     # _homogenize ensures
    121     #  - all(len(x) == len(index) for x in arrays)
    122     #  - all(x.ndim == 1 for x in arrays)
   (...)
    125
    126 else:
    127     index = ensure_index(index)

File ~\anaconda3\Lib\site-packages\pandas\core\internals\construction.py:630, in
_homogenize(data, index, dtype)
    627         val = lib.fast_multiget(val, oindex._values, default=np.nan)
    629     val = sanitize_array(val, index, dtype=dtype, copy=False)
--> 630     com.require_length_match(val, index)
    631     refs.append(None)
    633 homogenized.append(val)

File ~\anaconda3\Lib\site-packages\pandas\core\common.py:561, in require_length_m
atch(data, index)
    557 """
    558 Check the length of data matches the length of the index.
    559 """
```

```
    560 if len(data) != len(index):
--> 561     raise ValueError(
    562         "Length of values "
    563         f"({len(data)}) "
    564         "does not match length of index "
    565         f"({len(index)})"
    566     )

ValueError: Length of values (3) does not match length of index (4)
```

$Step - 7$

shape

- Number of rows
- Number of columns

In [28]:
```
df.shape
# Matrix format
# 4 rows and 3 columns
print("the number of rows are:",df.shape[0])
print("the number of columns are:",df.shape[1])
```

```
the number of rows are: 4
the number of columns are: 3
```

$Step - 8$

Drop the column

- In order to drop column we required 3 arguments
- Column name
  - The coulmn name we need to mention
- axis
  - axis=0 **represents rows**
  - axis=1 **represnts columns**
- inplace
  - we are drop column means we are modifying the data frame
  - so this modifications we want to save in same variable or different variable
  - If you want to keep in the same varaible the **inplace=True**

In [30]:
```
df
```

Out[30]:

	Names	Age	City
A	Ayush	25	hyd
B	Avinash	30	blr
C	Akash	35	chennai
D	Anvi	40	Mumbai

In [31]:
```
df.drop('City',
        axis=1,
        inplace=True)
```

In [32]:
```
df
```

Out[32]:

	Names	Age
A	Ayush	25
B	Avinash	30
C	Akash	35
D	Anvi	40

In [36]:
```
df1=df.drop('Age',axis=1)  # do not give inplace=True
```

In [37]:
```
df # has two columns
```

Out[37]:

	Names	Age
A	Ayush	25
B	Avinash	30
C	Akash	35
D	Anvi	40

In [38]:
```
df1
```

Out[38]:

	Names
A	Ayush
B	Avinash
C	Akash
D	Anvi

In [39]:
```
import pandas as pd
names=['Ayush','Avinash','Akash','Anvi']
age=[25,30,35,40]
city=['hyd','blr','chennai','Mumbai']

data=zip(names,age,city)
```

```
cols=['Names','Age','City']
idx=['A','B','C','D']
df=pd.DataFrame(data,
                index=idx,
                columns=cols)
df
```

Out[39]:

	Names	Age	City
A	Ayush	25	hyd
B	Avinash	30	blr
C	Akash	35	chennai
D	Anvi	40	Mumbai

In [40]:
```
df.drop('City',axis=1,inplace=True)
```

In [41]:
```
df
```

Out[41]:

	Names	Age
A	Ayush	25
B	Avinash	30
C	Akash	35
D	Anvi	40

In [42]:
```
df.drop('Age',axis=1)
```

Out[42]:

	Names
A	Ayush
B	Avinash
C	Akash
D	Anvi

In [43]:
```
df
```

Out[43]:

	Names	Age
A	Ayush	25
B	Avinash	30
C	Akash	35
D	Anvi	40

$Step - 9$

Multiple columns drop

In [44]:
```
#again read the data
import pandas as pd

names=['Ayush','Avinash','Akash','Anvi']
age=[25,30,35,40]
city=['hyd','blr','chennai','Mumbai']

data=zip(names,age,city)
cols=['Names','Age','City']
idx=['A','B','C','D']

df=pd.DataFrame(data,
                index=idx,
                columns=cols)
df
```

Out[44]:

	Names	Age	City
A	Ayush	25	hyd
B	Avinash	30	blr
C	Akash	35	chennai
D	Anvi	40	Mumbai

In [45]:
```
df.drop(['Age','City'],axis=1,inplace=True)
```

In [46]:
```
df
```

Out[46]:

	Names
A	Ayush
B	Avinash
C	Akash
D	Anvi

$Step - 10$

Drop the rows

In [47]:
```
#again read the data
import pandas as pd

names=['Ayush','Avinash','Akash','Anvi']
age=[25,30,35,40]
city=['hyd','blr','chennai','Mumbai']

data=zip(names,age,city)
cols=['Names','Age','City']
idx=['A','B','C','D']

df=pd.DataFrame(data,
                index=idx,
```

```
                columns=cols)
df
```

Out[47]:

	Names	Age	City
A	Ayush	25	hyd
B	Avinash	30	blr
C	Akash	35	chennai
D	Anvi	40	Mumbai

In [49]:
```
df.drop('A',axis=0,inplace=True)
```

In [50]:
```
df
```

Out[50]:

	Names	Age	City
B	Avinash	30	blr
C	Akash	35	chennai
D	Anvi	40	Mumbai

$Step - 11$

How to save the data frame

- Generally data frames are two types
- csv (comma seperated value)
- excel

In [57]:
```
df
```

Out[57]:

	Names	Age	City
B	Avinash	30	blr
C	Akash	35	chennai
D	Anvi	40	Mumbai

In [56]:
```
# file name=data
# extention=.csv
df.to_csv('data.csv')
df.to_csv('data1.csv',index=False)
```

In [58]:
```
# Extension
#.xlsx
df.to_excel("data2.xlsx",index=False)
```

In [52]:
```
name=['Aayush','Avinash','Akash','Avani']
age=[20,25,30,35]
city=['Ahmedabad','Pune','Hydrabad','Mumbai']
```

```
country=['India','India','India','India']
data=zip(name,age,city,country)
cols=['Names','Age','City','Country']
idx=['A','B','C','D']
df_bhavesh=pd.DataFrame(data,index=idx,columns=cols)
df_bhavesh
```

Out[52]:

	Names	Age	City	Country
A	Aayush	20	Ahmedabad	India
B	Avinash	25	Pune	India
C	Akash	30	Hydrabad	India
D	Avani	35	Mumbai	India

In [53]:
```
df_bhavesh.drop('A',axis=0,inplace=True)
```

In [54]:
```
df_bhavesh
```

Out[54]:

	Names	Age	City	Country
B	Avinash	25	Pune	India
C	Akash	30	Hydrabad	India
D	Avani	35	Mumbai	India

In [ ]:
```

```