Step-1:Import required packages

In [1]:
```
import pandas as pd                     # Dataframe operations
import numpy as np                      # Math operations
import matplotlib.pyplot as plt         # Plotting
import seaborn as sns                   # Plotting
```

Step-2:Create a Dataframe using A list

In [5]:
```
names=['Ramesh','Suresh','Mahesh']
age=[20,22,24]
pd.DataFrame(zip(names,age))
```

Out[5]:

	0	1
0	Ramesh	20
1	Suresh	22
2	Mahesh	24

pd.DataFrame(data,index,columns)

- data=None,
- index : Index or array-like
- columns: Index or array-like

In [16]:
```
names=['Ramesh','Suresh','Mahesh']
age=[20,22,24]
#pd.DataFrame(data,index,columns)
pd.DataFrame(zip(names,age))
```

Out[16]:

	0	1
0	Ramesh	20
1	Suresh	22
2	Mahesh	24

Step-3: Add the column names

In [25]:
```
names=['Ramesh','Suresh','Mahesh']
age=[20,22,24]
cols=['Names','Age']
#pd.DataFrame(data,index,columns)
pd.DataFrame(zip(names,age),
            columns=['Names','Age'])
```

Out[25]:

	Names	Age
0	Ramesh	20
1	Suresh	22
2	Mahesh	24

Step-4: Change the Index

In [30]:
```
names=['Ramesh','Suresh','Mahesh']
age=[20,22,24]
cols=['Names','Age']
idx=['A','B','C']
#pd.DataFrame(data,index,columns)
pd.DataFrame(zip(names,age),
             index=idx,
             columns=cols)
```

Out[30]:

	Names	Age
A	Ramesh	20
B	Suresh	22
C	Mahesh	24

Step-5:Create empty Dataframe and update the df

In [33]:
```
# Above one we created dataframe with values directly
# Now first we will create empty df
# then we will add the values

df=pd.DataFrame()
df
```

Out[33]: —

In [35]:
```
df['Names']
# If I run this if any column values available under  names it wil display
# otherwise error
# Either you are creating a new colum
# or you are modify the column values
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[35], line 1
----> 1 df['Names']

File E:\Anaconda\Lib\site-packages\pandas\core\frame.py:4102, in DataFrame.__geti
tem__(self, key)
   4100 if self.columns.nlevels > 1:
   4101     return self._getitem_multilevel(key)
-> 4102 indexer = self.columns.get_loc(key)
   4103 if is_integer(indexer):
   4104     indexer = [indexer]

File E:\Anaconda\Lib\site-packages\pandas\core\indexes\range.py:417, in RangeInde
x.get_loc(self, key)
    415         raise KeyError(key) from err
    416 if isinstance(key, Hashable):
--> 417     raise KeyError(key)
    418 self._check_indexing_error(key)
    419 raise KeyError(key)

KeyError: 'Names'
```

In [37]:
```
names=['Ramesh','Suresh','Mahesh']
df['Names']=names
df['Age']=[20,22,24]
df
```

Out[37]:

	Names	Age
0	Ramesh	20
1	Suresh	22
2	Mahesh	24

In [39]:
```
df=pd.DataFrame()
df['Names']=['Ramesh','Suresh','Mahesh']
df['Age']=[20,22,24]
df
```

Out[39]:

	Names	Age
0	Ramesh	20
1	Suresh	22
2	Mahesh	24

In [43]:
```
# My dataframe name is : df
df['Age']
```

Out[43]:
```
0    20
1    22
2    24
Name: Age, dtype: int64
```

In [45]:
```
df['Age']=[30,32,34] # Updating
df['age']=[100,200,300] # Creating
```

```
df['City']=['Hyd','Blr','Chennai'] # Creating
df
```

Out[45]:

	Names	Age	age	City
0	Ramesh	30	100	Hyd
1	Suresh	32	200	Blr
2	Mahesh	34	300	Chennai

Notes

- Python case sensitive
- 'Age' and 'age' consider as different
- Updated column values always should be len dataframe
- Dont expect Null like sql if we miss any value

In [50]:
```
df['Age']=[30,34]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[50], line 1
----> 1 df['Age']=[30,34]

File E:\Anaconda\Lib\site-packages\pandas\core\frame.py:4311, in DataFrame.__seti
tem__(self, key, value)
   4308     self._setitem_array([key], value)
   4309 else:
   4310     # set column
-> 4311     self._set_item(key, value)

File E:\Anaconda\Lib\site-packages\pandas\core\frame.py:4524, in DataFrame._set_i
tem(self, key, value)
   4514 def _set_item(self, key, value) -> None:
   4515     """
   4516     Add series to DataFrame in specified column.
   4517
   (...)
   4522     ensure homogeneity.
   4523     """
-> 4524     value, refs = self._sanitize_column(value)
   4526     if (
   4527         key in self.columns
   4528         and value.ndim == 1
   4529         and not isinstance(value.dtype, ExtensionDtype)
   4530     ):
   4531         # broadcast across multiple columns if necessary
   4532         if not self.columns.is_unique or isinstance(self.columns, MultiIn
dex):

File E:\Anaconda\Lib\site-packages\pandas\core\frame.py:5266, in DataFrame._sanit
ize_column(self, value)
   5263     return _reindex_for_setitem(value, self.index)
   5265 if is_list_like(value):
-> 5266     com.require_length_match(value, self.index)
   5267 arr = sanitize_array(value, self.index, copy=True, allow_2d=True)
   5268 if (
   5269     isinstance(value, Index)
   5270     and value.dtype == "object"
   (...)
   5273     # TODO: Remove kludge in sanitize_array for string mode when enforcin
g
   5274     # this deprecation

File E:\Anaconda\Lib\site-packages\pandas\core\common.py:573, in require_length_m
atch(data, index)
    569 """
    570 Check the length of data matches the length of the index.
    571 """
    572 if len(data) != len(index):
--> 573     raise ValueError(
    574         "Length of values "
    575         f"({len(data)}) "
    576         "does not match length of index "
    577         f"({len(index)})"
    578     )

ValueError: Length of values (2) does not match length of index (3)
```

Step-6: Drop the column

- labels:
- axis: 'Axis' = 0,
- index:
- columns:
- inplace: 'bool' = False,

In [54]:
```
df
```

Out[54]:

	Names	Age	age	City
0	Ramesh	30	100	Hyd
1	Suresh	32	200	Blr
2	Mahesh	34	300	Chennai

In [ ]:
```
# what is our dataframe name : df
# I want to drop 'age' column
# what is the age column index
# 2 or [2]
# 2 or [2] indictaes row or column
# have we mentioned to python is it a row or column : NOOOO
# python confuse
# axis
df.drop(2,axis=0) # Explictly drop row
df.drop(2) # No error 2nd means 3rd rows
df.drop(2,axis=1) # Exp drop colum
```

In [ ]:
```
# always execute in individual cell
df.drop(2)  # row
df.drop([2]) #
df.drop(2,axis=0)
df.drop(2,axis=1)
df.drop(20)
```

In [56]:
```
df.drop(2)
```

Out[56]:

	Names	Age	age	City
0	Ramesh	30	100	Hyd
1	Suresh	32	200	Blr

In [60]:
```
df.drop([1,2])
```

Out[60]:

	Names	Age	age	City
0	Ramesh	30	100	Hyd

In [62]:
```
df.drop(2,axis=0)
```

Out[62]:

	Names	Age	age	City
0	Ramesh	30	100	Hyd
1	Suresh	32	200	Blr

In [68]:

```
df
```

Out[68]:

	Names	Age	age	City
0	Ramesh	30	100	Hyd
1	Suresh	32	200	Blr
2	Mahesh	34	300	Chennai

In [70]:

```
df.drop(columns=['age'])
```

Out[70]:

	Names	Age	City
0	Ramesh	30	Hyd
1	Suresh	32	Blr
2	Mahesh	34	Chennai

In [72]:

```
df.drop(index=[2])
```

Out[72]:

	Names	Age	age	City
0	Ramesh	30	100	Hyd
1	Suresh	32	200	Blr

In [78]:

```
df.columns[2]
```

Out[78]:

```
'age'
```

In [80]:

```
df.drop('age',axis=1)
```

Out[80]:

	Names	Age	City
0	Ramesh	30	Hyd
1	Suresh	32	Blr
2	Mahesh	34	Chennai

In [82]:

```
df
df.drop(2) # works
df.drop(2,axis=0) # works
df.drop(2,axis=1) # fail
df.drop('age',axis=1) # works
df.drop(labels=[2])    # works
df.drop(columns=['age']) # works
```

Out[82]:

	Names	Age	age	City
0	Ramesh	30	100	Hyd
1	Suresh	32	200	Blr
2	Mahesh	34	300	Chennai

In [84]:
```
#df.drop('age') # python checks 'age' naaam pe row
df.drop('age',axis=1)
df.drop(columns=['age'])
```

Out[84]:

	Names	Age	City
0	Ramesh	30	Hyd
1	Suresh	32	Blr
2	Mahesh	34	Chennai

In [86]:
```
df
```

Out[86]:

	Names	Age	age	City
0	Ramesh	30	100	Hyd
1	Suresh	32	200	Blr
2	Mahesh	34	300	Chennai

In [88]:
```
df.drop(columns=['age'],
        inplace=True)
```

In [90]:
```
df
```

Out[90]:

	Names	Age	City
0	Ramesh	30	Hyd
1	Suresh	32	Blr
2	Mahesh	34	Chennai

In [92]:
```
df
```

Out[92]:

	Names	Age	City
0	Ramesh	30	Hyd
1	Suresh	32	Blr
2	Mahesh	34	Chennai

In [ ]:
```
# Assignment-1
df.rename()
```

In [ ]:
```
# Dataframe name df
# Assignment-2
df.shape
df.size
df.columns
df.isnull()
len(df)
df.info
df.drop_duplicates
df.dtypes
```

In [ ]:
```
# Import the packages
# Created a dataframe using list
# Add a column names
# Add our own index
# Created empty dataframe and we updated with data
# We learned how to modify the rows for all the column
# Drop a column and index
# we understood the arguments
#    axis=0   rows   axis=1
#    inplace

# Whenever we open any notebook
# 1.Import your pcakges cell you need to run
# 2. data read you need to run
```

In [3]:
```
df=pd.DataFrame()
df['Names']=['Ramesh','Suresh','Sathish']
df['Age']=[20,22,24]
df['City']=['Hyd','Blr','Pune']
df
```

Out[3]:

	Names	Age	City
0	Ramesh	20	Hyd
1	Suresh	22	Blr
2	Sathish	24	Pune

In [ ]:
```
df.drop(2)  # 2means label is it row or columns axis axis=0
df.drop(2,axis=0)
df.drop(index=2)
df.drop([1,2])
df.drop([1,2],axis=0)
df.drop(index=[1,2])

df.drop('City',axis=1)
df.drop(['Names','City'],axis=1)
df.drop(columns='City')
df.drop(columns=['Names','City'])
```

Step-7:Rename the column and index

- mapper: 'Renamer | None' = None,
- index: 'Renamer | None' = None,
- columns: 'Renamer | None' = None,

- axis: 'Axis | None' = None,
- inplace: 'bool' = False,

mapper means dictionary

```
In [10]: df.columns
         dict1={'City':'city'}
         #df.rename(dict1)  # this mapper for index or column
         df.rename(dict1,axis=1)
```

```
Out[10]:
     Names  Age  city
  0  Ramesh  20   Hyd
  1  Suresh  22   Blr
  2 Sathish  24  Pune
```

```
In [12]: df.rename(columns=dict1)
```

```
Out[12]:
     Names  Age  city
  0  Ramesh  20   Hyd
  1  Suresh  22   Blr
  2 Sathish  24  Pune
```

```
In [14]: df.columns=['A','B','C']
         # Inplace also not required
```

```
In [18]: df.index=['I','II','III']
```

```
In [20]: df
```

```
Out[20]:
            A   B     C
    I  Ramesh  20   Hyd
   II  Suresh  22   Blr
  III Sathish  24  Pune
```

```
In [26]: df.columns=['Names','Age','City']
         df.index=[0,1,2]
         df
```

```
Out[26]:
     Names  Age  City
  0  Ramesh  20   Hyd
  1  Suresh  22   Blr
  2 Sathish  24  Pune
```

Step-8: append the rows

- we need to use **loc**
- df.loc[,]
    - Example : df.loc[2,'City']
- Multiple rows from multiple columns
    - Example : df.loc[[1,2],['Names','City']]
- df.loc[start:stop:step,]
    - Example : df.loc[0:3,'City']

In [ ]:
```
df.loc[2] # 2nd index of all column
df.loc[[2]] # 2nd index of all column
df.loc[[1,2]]
df.loc[0:3]  # [[0,1,2]]
df.loc[:]
df.loc[2,'City'] # s
df.loc[2,['City']]  # s
df.loc[[2],['City']] # df
df.loc[[1,2,],['Names','City']]  # df
df.loc[:,'City'] # s
df.loc[:,['City']] # df : [0,1,2]
```

In [42]:
```
list(range(0,3))
```

Out[42]:
```
[0, 1, 2]
```

In [32]:
```
type(df.loc[2])
```

Out[32]:
```
pandas.core.series.Series
```

In [36]:
```
type(df.loc[[2]])
```

Out[36]:
```
pandas.core.frame.DataFrame
```

In [ ]:
```
[[]]  dataframe 2d
[]   1d series
```

In [48]:
```
df.loc[2,['City']]  # series
```

Out[48]:
```
City    Pune
Name: 2, dtype: object
```

In [50]:
```
df.loc[[2],['City']]
```

Out[50]:

	City
2	Pune

In [52]:
```
df.loc[[2],'City']
```

Out[52]:
```
2    Pune
Name: City, dtype: object
```

iloc

```
In [ ]: df.loc[<rows>,<column names>]
        df.loc[<rows>,'City']
        df.iloc[<rows>,2]
```

```
In [ ]: df.iloc[2]
        df.iloc[[2]]
        df.iloc[[1,2]]
        df.iloc[0:3]
        df.iloc[:]
        df.iloc[2,2]
        df.iloc[2,[2]]
        df.iloc[[2],[2]]
        df.iloc[[1,2,],[0,2]]
        df.iloc[:,2]
        df.iloc[:,[2]]
```

```
In [55]: df.iloc[:,[2]]
         df.iloc[0:3,[2]] # 0:3  [0,1,2]
         df.iloc[[0,1,2],[2]]
```

Out[55]:

	City
0	Hyd
1	Blr
2	Pune

```
In [57]: df.iloc[[0,1,2],2]
```

```
Out[57]: 0      Hyd
         1      Blr
         2     Pune
         Name: City, dtype: object
```

Step-9: Save the dataframe

- where i want to save
- on what name i want to save
- on what extenstion: .csv,.xlsx

```
In [63]: path1=r"C:\Users\omkar\OneDrive\Documents\Gen_AI\Batch_april_2025\employees.csv"
         path2=r"employees.csv"
         path3=r"C:\Users\omkar\OneDrive\Documents\Gen_AI\Batch_april_2025\employees.xlsx
         path4=r"employees.xlsx"

         df.to_csv(path1)
         df.to_csv(path2)
         df.to_excel(path3)
         df.to_excel(path4)
```

Step-10: Read the dataframe

In [66]:
```
pd.read_csv(path1)
```
Out[66]:

	Unnamed: 0	Names	Age	City
0	0	Ramesh	20	Hyd
1	1	Suresh	22	Blr
2	2	Sathish	24	Pune

In [68]:
```
pd.read_excel(path3)
```
Out[68]:

	Unnamed: 0	Names	Age	City
0	0	Ramesh	20	Hyd
1	1	Suresh	22	Blr
2	2	Sathish	24	Pune

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

	Names	Age	City
0	Ramesh	20	Hyd
1	Suresh	22	Blr
2	Sathish	24	Pune

In [72]:
```
path1=r"C:\Users\omkar\OneDrive\Documents\Gen_AI\Batch_april_2025\employees.csv"
path2=r"employees.csv"
path3=r"C:\Users\omkar\OneDrive\Documents\Gen_AI\Batch_april_2025\employees.xlsx
path4=r"employees.xlsx"

df.to_csv(path1,index=False)
df.to_csv(path2,index=False)
df.to_excel(path3,index=False)
df.to_excel(path4,index=False)
```
In [74]:
```
pd.read_excel(path3)
```
Out[74]:

	Names	Age	City
0	Ramesh	20	Hyd
1	Suresh	22	Blr
2	Sathish	24	Pune

In [ ]:
```

```