In [1]:
```
# creating a dictionary for student
s={"name":["sai","lokesh","ganesh"],
  "rno":[120,311,522],
  "branch":["cse","IT","ECE"]}

```

In [2]:
```
import pandas as pd
df=pd.DataFrame(s)
df
```

Out[2]:

	name	rno	branch
0	sai	120	cse
1	lokesh	311	IT
2	ganesh	522	ECE

In [3]:
```
#to get column names from dataframe
df.columns
```

Out[3]:
```
Index(['name', 'rno', 'branch'], dtype='object')
```

In [5]:
```
# to get row index
df.index
```

...

In [7]:
```
# to get values from the dataframe
df.values
```

...

In [8]:
```
# to get top 2 records
df.head(2)
```

...

In [9]:
```
# to get first record from a dataframe
df.head(1)
```

...

In [10]:
```
# to get no of rows and columns from a dataframe

df.shape
```

Out[10]:
```
(3, 3)
```

In [12]:
```
# to get last record
df.tail(1)
```
...

In [13]:
```
df.tail(2)
```
...

In [14]:
```
df["name"]  # to access a particular column values
```
...

In [16]:
```
# to access multiple columns

df[["name","branch"]]
```
...

In [15]:
```
# to add a new column marks to existing dataframe s
df["marks"]=[89,78,85]
df
```
...

In [17]:
```
# to sort data based on name
df.sort_values("name")
```
...

In [18]:
```
df["java"]=[75,95,85]
df["python"]=[85,75,88]
df
```
...

In [19]:
```
df["Total"]=df["java"]+df["python"]
df
```

Out[19]:

	name	rno	branch	marks	java	python	Total
0	sai	120	cse	89	75	85	160
1	lokesh	311	IT	78	95	75	170
2	ganesh	522	ECE	85	85	88	173

In [20]:
```
# indexing
# iloc -> intereger based indexing
# loc -> both int and string based indexing
```

In [21]:
```
df.iloc[1,1]
```

Out[21]:
```
311
```

In [22]:
```
df.loc[(df["name"]=="lokesh")]
```
...

In [23]:
```
df.loc[(df["branch"]=="ECE")]
```
...

In [24]:
```
# get student records whose,java marks are
# in between 80 and 98
df.loc[(df["java"]>80) & (df["java"]<98) ]
```
...

In [26]:
```
# to add a new record
df.loc[3]=["krishna",543,"ECE",87,89,98,187]
df
```

Out[26]:

	name	rno	branch	marks	java	python	Total
0	sai	120	cse	89	75	85	160
1	lokesh	311	IT	78	95	75	170
2	ganesh	522	ECE	85	85	88	173
3	krishna	543	ECE	87	89	98	187

In [27]:
```
# to get row with index 1
df.iloc[1]
```
...

In [28]:
```
df.iloc[3]
```
...

In [30]:
```
# to update particular value
df.loc[1,"rno"]=200    # row index,col name
df
```
...

In [32]:
```
df.loc[3,"branch"]="EEE"
df
```
...

In [33]:
```
df.loc[0,"branch"]="CSE"
df
```
...

In [35]:
```
# renaming a particular column name
df.rename(columns={'rno':'Roll_No'},inplace=True)
df
```

...

In [36]:
```
# renaming all the columns
df.columns=['Name','Roll_No','Branch','Marks','Java','Python','Total']
df
```

...

In [37]:
```
# to rename row index
df.index=['a','b','c','d']
df
```

...

In [38]:
```
# to delete data from a dataframe
# drop
# axis=0 -> row based
# axis=1 -> column based
```

In [39]:
```
l=[[67,87,98],[90,87,56]]
df=pd.DataFrame(l)
df
```

...

In [40]:
```
df.columns=["stu1","stu2","stu3"]
df
```

...

In [41]:
```
#  to delete the 0th index record
df.drop(0,axis=0,inplace=True)
df
```

...

In [42]:
```
# to delete column stu2
df.drop("stu2",axis=1,inplace=True)
df
```

...

In [43]:
```
df.loc[2]=[89,67]
df.loc[3]=[56,76]
df
```

...

In [44]:
```
# to delete all the rows
df.drop(df.index,inplace=True)
df
```

...

In [45]:
```
# to delete all the columns
df.drop(df.columns,axis=1,inplace=True)
df
```

Out[45]:
```
_
```

File I/O

In [4]:
```
import pandas as pd
df=pd.read_csv("C://Users//cselab4//Downloads//Salary_Data.csv")
df
```

...

In [5]:
```
# to get no of rows and coumns
df.shape
```

...

In [6]:
```
# to get column names
df.columns
```

...

In [7]:
```
# to get row index
df.index
```

...

In [8]:
```
# to get values
df.values
```

...

In [9]:
```
df.describe()
```

...

In [10]:
```
# to get count of female and males
df["Gender"].value_counts()
```

...

In [12]:
```
# count of only females
len(df["Gender"]=="Female")
```

Out[12]:
```
6704
```

In [14]:
```
# to get no of job-titles
df["Job Title"].value_counts()
```

...

In [17]:
```
# to get the records of Data analyst job title
df[df["Job Title"]=="Data Analyst"]
```

...

In [19]:
```
# to update any value
df.loc[2,"Education Level"]="PHD"
df
```

...

In [20]:
```
# To get the maximum salary records
df[df["Salary"]==max(df["Salary"])]
```

...

In [25]:
```
# sorting values based on the salary
df.sort_values("Salary",ascending=False)
```

...

In [31]:
```
df["Job Title"].unique()
```

...

In [32]:
```
len(df["Job Title"].unique())
```

Out[32]:
```
194
```

In [26]:
```
# statistics
df.max()
```

```
C:\Users\cselab4\AppData\Local\Temp\ipykernel_1376\2107108110.py:2: FutureWarni
ng: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=No
ne') is deprecated; in a future version this will raise TypeError.  Select only
valid columns before calling the reduction.
  df.max()
```

Out[26]:
```
Age                        62.0
Years of Experience        34.0
Salary                 250000.0
dtype: float64
```

In [27]:
```
df.min()
```

```
C:\Users\cselab4\AppData\Local\Temp\ipykernel_1376\3962516015.py:1: FutureWarni
ng: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=No
ne') is deprecated; in a future version this will raise TypeError.  Select only
valid columns before calling the reduction.
  df.min()
```

Out[27]:
```
Age                      21.0
Years of Experience       0.0
Salary                  350.0
dtype: float64
```

In [28]:
```
df.std()
```

```
C:\Users\cselab4\AppData\Local\Temp\ipykernel_1376\3390915376.py:1: FutureWarni
ng: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=No
ne') is deprecated; in a future version this will raise TypeError.  Select only
valid columns before calling the reduction.
  df.std()
```

Out[28]:
```
Age                         7.614633
Years of Experience         6.059003
Salary                  52786.183911
dtype: float64
```

In [29]:
```
df.mean()
```

...

Data Cleaning with pandas

*NAN-> not a number

- to deal with duplicates and missing values
  - isnull()
  - notnull()
  - dropna()
  - fillna()
  - replace()

Outliers:

- outliers are the observations that are significantly differ from other data points

In [49]:
```
emp=pd.read_csv("C://Users//cselab4//Downloads//employe.csv")
emp
```

Out[49]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing
1	Thomas	Male	3/31/1996	6:53 AM	61933	4.170	True	NaN
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance
3	Jerry	Male	3/4/2005	1:00 PM	138705	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 PM	101004	1.389	True	Client Services
...	...	...	...	...	...	...	...	...
995	Henry	NaN	11/23/2014	6:09 AM	132483	16.655	False	Distribution
996	Phillip	Male	1/31/1984	6:30 AM	42392	19.675	False	Finance
997	Russell	Male	5/20/2013	12:39 PM	96914	1.421	False	Product
998	Larry	Male	4/20/2013	4:45 PM	60500	11.985	False	Business Development
999	Albert	Male	5/15/2012	6:24 PM	129949	10.169	True	Sales

1000 rows × 8 columns

In [50]:
```
emp.head()
```

Out[50]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing
1	Thomas	Male	3/31/1996	6:53 AM	61933	4.170	True	NaN
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance
3	Jerry	Male	3/4/2005	1:00 PM	138705	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 PM	101004	1.389	True	Client Services

- isnull() - Detect the missing values for array-like objects
- notnull() - Detect non-missing values for array like objects

In [51]:
```
emp.shape
```

Out[51]:
```
(1000, 8)
```

In [52]:
```
# no of missing values or null values in every column
emp.isnull().sum()
```

Out[52]:
```
First Name            67
Gender               145
Start Date             0
Last Login Time        0
Salary                 0
Bonus %                0
Senior Management     67
Team                  43
dtype: int64
```

In [53]:
```
emp.notnull().sum() # non missing values count
```

Out[53]:
```
First Name            933
Gender                855
Start Date           1000
Last Login Time      1000
Salary               1000
Bonus %              1000
Senior Management     933
Team                  957
dtype: int64
```

- dropna() - dropna method removes the rows that contains null values

In [54]:
```
emp.dropna()
```

Out[54]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance
3	Jerry	Male	3/4/2005	1:00 PM	138705	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 PM	101004	1.389	True	Client Services
5	Dennis	Male	4/18/1987	1:35 AM	115163	10.125	False	Legal
...	...	...	...	...	...	...	...	...
994	George	Male	6/21/2013	5:47 PM	98874	4.479	True	Marketing
996	Phillip	Male	1/31/1984	6:30 AM	42392	19.675	False	Finance
997	Russell	Male	5/20/2013	12:39 PM	96914	1.421	False	Product
998	Larry	Male	4/20/2013	4:45 PM	60500	11.985	False	Business Development
999	Albert	Male	5/15/2012	6:24 PM	129949	10.169	True	Sales

764 rows × 8 columns

In [55]:
```
emp.dropna().sum()
```

Out[55]:
```
First Name            DouglasMariaJerryLarryDennisRubyAngelaFrancesJ...
Gender                MaleFemaleMaleMaleMaleFemaleFemaleFemaleFemale...
Start Date            8/6/19934/23/19933/4/20051/24/19984/18/19878/1...
Last Login Time       12:42 PM11:17 AM1:00 PM4:47 PM1:35 AM4:20 PM6:...
Salary                                                         69090962
Bonus %                                                        7753.103
Senior Management                                                   381
Team                  MarketingFinanceFinanceClient ServicesLegalPro...
dtype: object
```

- fillna() - to fill the null values with user specific value
- filling missing value
    - mean
    - median
    - mode
    - constant

In [56]:
```
emp["Gender"].isnull()
```

Out[56]:
```
0      False
1      False
2      False
3      False
4      False
       ...
995     True
996    False
997    False
998    False
999    False
Name: Gender, Length: 1000, dtype: bool
```

In [57]:
```
emp["Gender"].isnull().sum()
```

Out[57]:
```
145
```

In [58]:
```
emp["Gender"].fillna("No Gender")
```

...

In [59]:
```
emp["Gender"].fillna(method="pad")    # previous value
```

...

In [60]:
```
emp["Gender"].fillna(method="bfill")  # backward value
```

...

In [61]:
```
emp["Gender"].fillna(0)
```

...

In [64]:
```
# replace-> to replace a value with some other value
emp.replace(to_replace="Male",value="MALE")
```

...

In [ ]:
```

```