```
#NAME: DIKSHA TIDKE
#PRN: 202401050012
#BATCH: CC2
#ROLL NO:27
#Their are some pandas operation are explained below along with function of operation and syntax for it
import pandas as pd
```

```
#Store the employee information in a structured table format.
data = {
    'EmpID': [101, 102, 103, 104, 105],
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Age': [28, 34, 25, 40, 30],
    'Department': ['HR', 'IT', 'IT', 'Finance', 'HR'],
    'Salary': [50000, 70000, 55000, 80000, 62000],
    'JoiningDate': ['15-01-2020', '22-03-2019', '30-07-2021', '10-11-2018', '18-05-2022']
}

df = pd.DataFrame(data)
print(data)
```

```
{'EmpID': [101, 102, 103, 104, 105], 'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'], 'Age': [28, 34, 25, 40, 30], 'Department': ['HR', 'IT', 'IT', 'Finance', 'HR'], 'Sala
```

```
#Display Basic Information
#Idea: Quickly check number of rows, columns, and data types.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   EmpID        5 non-null      int64
 1   Name         5 non-null      object
 2   Age          5 non-null      int64
 3   Department   5 non-null      object
 4   Salary       5 non-null      int64
 5   JoiningDate  5 non-null      object
dtypes: int64(3), object(3)
memory usage: 372.0+ bytes
```

```
#Gives mean, median, std dev, min, max for numeric columns like Age, Salary.
df.describe()
```

	EmpID	Age	Salary
count	5.000000	5.000000	5.000000
mean	103.000000	31.400000	63400.000000
std	1.581139	5.813777	11949.895397
min	101.000000	25.000000	50000.000000
25%	102.000000	28.000000	55000.000000
50%	103.000000	30.000000	62000.000000
75%	104.000000	34.000000	70000.000000
max	105.000000	40.000000	80000.000000

```
#Sometimes you only need to work with a few columns
df[['Name', 'Department']]
```

	Name	Department
0	Alice	HR
1	Bob	IT
2	Charlie	IT
3	David	Finance
4	Eve	HR

```
#Select specific rows that match a condition — for example, employees from HR.
df[df['Department'] == 'HR']
```

	EmpID	Name	Age	Department	Salary	JoiningDate
0	101	Alice	28	HR	50000	15-01-2020
4	105	Eve	30	HR	62000	18-05-2022

```
#Arrange rows in ascending/descending order based on a column, like Salary.
# Sort by Salary descending
df.sort_values(by='Salary', ascending=False)
```

	EmpID	Name	Age	Department	Salary	JoiningDate
3	104	David	40	Finance	80000	10-11-2018
1	102	Bob	34	IT	70000	22-03-2019
4	105	Eve	30	HR	62000	18-05-2022
2	103	Charlie	25	IT	55000	30-07-2021
0	101	Alice	28	HR	50000	15-01-2020

```
#Group by a category (like Department) and calculate things like mean salary.
df.groupby('Department')['Salary'].mean()
```

Department	Salary
Finance	80000.0
HR	56000.0
IT	62500.0

dtype: float64

```
#We can calculate new columns, like years of experience based on JoiningDate.
# Convert JoiningDate to datetime
df['JoiningDate'] = pd.to_datetime(df['JoiningDate'], dayfirst=True)

import datetime
today = datetime.datetime.today()

# Calculate Experience
df['Experience'] = (today - df['JoiningDate']).dt.days // 365

print(df[['Name', 'Experience']])
```

```
      Name  Experience
0    Alice           5
1      Bob           6
2  Charlie           3
3    David           6
4      Eve           2
```

```
#Change column names to something easier or cleaner.
df.rename(columns={'EmpID': 'EmployeeID', 'JoiningDate': 'DateOfJoining'}, inplace=True)
print(df.head())
```

```
   EmployeeID     Name  Age Department  Salary DateOfJoining  Experience
0         101    Alice   28         HR   50000    2020-01-15           5
1         102      Bob   34         IT   70000    2019-03-22           6
2         103  Charlie   25         IT   55000    2021-07-30           3
3         104    David   40    Finance   80000    2018-11-10           6
4         105      Eve   30         HR   62000    2022-05-18           2
```

```
#Remove unwanted columns from the dataset.
# Drop the 'Experience' column
df.drop('Experience', axis=1, inplace=True)
print(df.head())
```

```
   EmployeeID     Name  Age Department  Salary DateOfJoining
0         101    Alice   28         HR   50000    2020-01-15
1         102      Bob   34         IT   70000    2019-03-22
2         103  Charlie   25         IT   55000    2021-07-30
3         104    David   40    Finance   80000    2018-11-10
4         105      Eve   30         HR   62000    2022-05-18
```

```
#See which columns have missing data (null values).
df.isnull().sum()
```

	0
EmployeeID	0
Name	0
Age	0
Department	0
Salary	0
DateOfJoining	0

dtype: int64

```
#Create a new DataFrame with only specific rows — e.g., Salary > 60000.
high_salary_df = df[df['Salary'] > 60000]
```

```
print(high_salary_df)
```

```
   EmployeeID   Name  Age Department  Salary DateOfJoining
1         102    Bob   34         IT   70000    2019-03-22
3         104  David   40    Finance   80000    2018-11-10
4         105    Eve   30         HR   62000    2022-05-18
```

```
#Find out how many unique departments are there, etc.
df['Department'].nunique()
```

```
3
```

```
#How many employees in each department? (Frequency count)
df['Department'].value_counts()
```

Department	count
HR	2
IT	2
Finance	1

dtype: int64

```
#Run a custom function for every row or column.
df['NewSalary'] = df['Salary'].apply(lambda x: x * 1.10)
print(df[['Name', 'Salary', 'NewSalary']])
```

```
      Name  Salary  NewSalary
0    Alice   50000    55000.0
1      Bob   70000    77000.0
2  Charlie   55000    60500.0
3    David   80000    88000.0
4      Eve   62000    68200.0
```

```
#Convert a column into a different data type (e.g., string to datetime).
df['DateOfJoining'] = pd.to_datetime(df['DateOfJoining'])
print(df.dtypes)
```

```
EmployeeID                int64
Name                     object
Age                       int64
Department               object
Salary                    int64
DateOfJoining    datetime64[ns]
NewSalary               float64
dtype: object
```