```
In [ ]: Lab Assignment 6 - Python
        UTKARSH
```

```
In [ ]:
```

```
In [ ]: """ 1. Create a 4X2 integer array and Prints its attributes """
```

```
In [9]: import numpy as np
        p= np.array([[1, 2], [3, 4], [5, 6], [7, 8]])
        print(p.ndim)
        print(p.shape)
        print(p.size)
        print(p.T)
        print(p.dtype)
```

```
2
(4, 2)
8
[[1 3 5 7]
 [2 4 6 8]]
int32
```

```
In [ ]: """ 2. Create a 5X2 integer array from a range between 100 to 200 such
        that the difference between each element is 10 . """
```

```
In [29]: p = np.arange(100, 200, 10).reshape(5, 2)

         print("Array:")
         print(p)
```

```
Array:
[[100 110]
 [120 130]
 [140 150]
 [160 170]
 [180 190]]
```

```
In [ ]: """ 3. Create a numPy array. Return array of items by taking the third
        column from all rows """
```

```
In [35]: p= np.array([[1, 2, 3], [3, 4 , 9], [5, 6 , 11], [7, 8 , 13]])
         p_drop=p[:,2]
         print(p_drop)
```

```
[ 3  9 11 13]
```

```
In [ ]: """ 4. Create a numpy array with 3 elements. Do all the arithmetic
        operations with scalar element. """
```

```
In [45]: p1= np.array([1, 2, 3])
         p2=np.array([4, 5, 7])
         add1=np.add(p1,p2)
         print(add1)
         sub1=np.subtract(p1,p2)
```

```
print(sub1)
prod1=np.multiply(p1,p2)
print(prod1)
divd=np.divide(p1,p2)
print(divd)
```

```
[ 5  7 10]
[-3 -3 -4]
[ 4 10 21]
[0.25       0.4        0.42857143]
```

In [ ]:

```
"""  5. Create two 2-D numpy array. Do array concatenation, stack, vstack
and hstack on that. """
```

In [70]:

```
a1 = np.array([[1, 2, 3], [4, 5, 6]])
a2 = np.array([[7, 8, 9], [10, 11, 12]])

concatenated = np.concatenate((a1, a2))
stacked = np.stack((a1, a2))
vstacked = np.vstack((a1, a2))
hstacked = np.hstack((a1, a2))

print("Concatenated:")
print(concatenated)

print("\nStacked:")
print(stacked)

print("\nVertical Stack:")
print(vstacked)

print("\nHorizontal Stack:")
print(hstacked)
```

```
Concatenated:
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]

Stacked:
[[[ 1  2  3]
  [ 4  5  6]]

 [[ 7  8  9]
  [10 11 12]]]

Vertical Stack:
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]

Horizontal Stack:
[[ 1  2  3  7  8  9]
 [ 4  5  6 10 11 12]]
```

```
In [ ]:
```

```
In [ ]: 6. Create an Nd numpy array. Using array filter retrieve all the even
        elements in that array.
```

```
In [75]: import numpy as np
         b1 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
         even_elements = b1[b1 % 2 == 0]
         print("Even Elements:")
         print(even_elements)
```

```
Even Elements:
[2 4 6 8]
```

```
In [ ]:
```

```
In [ ]: 7. Write a python program to read a dataset from titanic file and do
        the following:
        • Read values to dataframe and print the first 6 rows.
        • Print the concise summary of the dataframe.
        • calculate the mean of each numeric column in dataframe.
        • find the count of null values in dataframe.
        • Take the average people survived.
        • Take the average male and female survived.
        • Group the entire dataframe based on Gender and Pclass.
```

```
In [84]: import pandas as pd
         dfw=pd.read_csv('titanic.csv')
         dfw.head(6)
```

Out[84]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.283
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.050
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.458

In [86]:

```
print(dfw.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
None
```

In [90]:

```
dfw.mean(numeric_only=True)
```

```
Out[90]: PassengerId    446.000000
         Survived         0.383838
         Pclass           2.308642
         Age             29.699118
         SibSp            0.523008
         Parch            0.381594
         Fare            32.204208
         dtype: float64
```

```
In [91]: dfw.isnull().sum()
```

```
Out[91]: PassengerId      0
         Survived         0
         Pclass           0
         Name             0
         Sex              0
         Age            177
         SibSp            0
         Parch            0
         Ticket           0
         Fare             0
         Cabin          687
         Embarked         2
         dtype: int64
```

```
In [92]: dfw['Survived'].mean()
```

```
Out[92]: 0.3838383838383838
```

```
In [100…  dfw.groupby("Sex")["Survived"].mean()
```

```
Out[100…  Sex
          female    0.742038
          male      0.188908
          Name: Survived, dtype: float64
```

```
In [108…  dfw.groupby(["Pclass","Sex"]).describe()
```

Out[108…

		PassengerId								Su
		count	mean	std	min	25%	50%	75%	max	count
Pclass	Sex									
1	female	94.0	469.212766	247.476723	2.0	293.50	447.0	698.25	888.0	94.0
	male	122.0	455.729508	247.026449	7.0	255.50	480.5	660.75	890.0	122.0
2	female	76.0	443.105263	243.627288	10.0	269.75	439.5	616.75	881.0	76.0
	male	108.0	447.962963	256.922546	18.0	225.75	416.5	677.50	887.0	108.0
3	female	144.0	399.729167	267.232416	3.0	165.25	376.0	636.00	889.0	144.0
	male	347.0	455.515850	261.921251	1.0	209.50	466.0	687.50	891.0	347.0

6 rows × 48 columns

In [ ]:
```
#8. Write a python program to read a dataset from weather file and do
#the following:
#• Read values to dataframe and print the first 6 rows.
#• Print the concise summary of the dataframe.
#• calculate the mean of each numeric column in dataframe.
#• find the count of null values in dataframe.
#• Take the maximum temperature from the weather data.
#• Retrieve the average windspeed from the weather data.
#• Make a new dataframe which contains minmum values of all columns based on event.
#• Make a new dataframe which contains minmum temperature based on event.
```

In [3]:
```
import pandas as pd
wd=pd.read_csv('weather.csv')
wd.head(6)
```

Out[3]:

	Unnamed: 0	day	temperature	windspeed	event
0	0	01-01-2022	32.0	6.0	Rain
1	1	01-04-2022	NaN	9.0	Sunny
2	2	01-05-2022	28.0	NaN	Snow
3	3	01-06-2022	NaN	7.0	NaN
4	4	01-07-2022	32.0	NaN	Rain
5	5	01-08-2022	NaN	NaN	Sunny

In [4]:
```
wd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Unnamed: 0   11 non-null     int64  
 1   day          11 non-null     object 
 2   temperature  7 non-null      float64
 3   windspeed    7 non-null      float64
 4   event        9 non-null      object 
dtypes: float64(2), int64(1), object(2)
memory usage: 572.0+ bytes
```

In [5]:
```
wd.describe()
```

Out[5]:

	Unnamed: 0	temperature	windspeed
count	11.000000	7.000000	7.000000
mean	5.000000	34.285714	8.857143
std	3.316625	4.386125	2.340126
min	0.000000	28.000000	6.000000
25%	2.500000	32.000000	7.500000
50%	5.000000	34.000000	8.000000
75%	7.500000	37.000000	10.500000
max	10.000000	40.000000	12.000000

In [6]:
```
wd.mean(numeric_only=True)
```

Out[6]:
```
Unnamed: 0      5.000000
temperature    34.285714
windspeed       8.857143
dtype: float64
```

In [7]:
```
wd.isnull().sum()
```

Out[7]:
```
Unnamed: 0     0
day            0
temperature    4
windspeed      4
event          2
dtype: int64
```

In [8]:
```
wd['temperature'].max()
```

Out[8]:
```
40.0
```

In [9]:
```
wd['windspeed'].mean()
```

Out[9]:
```
8.857142857142858
```

```
In [10]: wd.groupby('event').min()
```

Out[10]:

event	Unnamed: 0	day	temperature	windspeed
Cloudy	7	01-10-2022	34.0	8.0
Rain	0	01-01-2022	32.0	6.0
Snow	2	01-05-2022	28.0	NaN
Sunny	1	01-04-2022	40.0	9.0

```
In [11]: wd.groupby('event')['temperature'].min()
```

```
Out[11]: event
Cloudy    34.0
Rain      32.0
Snow      28.0
Sunny     40.0
Name: temperature, dtype: float64
```

```
In [ ]:
```

```
In [ ]: #9. Write a python program to read a dataset from diamonds file and
        #do the following:
        #• Read values to dataframe and print the first 6 rows.
        #• Print the concise summary of the dataframe.
        #• calculate the mean of each numeric column in dataframe.
        #• find the count of null values in dataframe.
        #• Calculate min, max price for each cut of diamonds.
        #• Take the count of all duplicate rows in the dataframe.
```

```
In [15]: import pandas as pd
         di=pd.read_csv('diamonds.csv')
         di.head(6)
```

Out[15]:

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
5	6	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48

```
In [16]: di.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53940 entries, 0 to 53939
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  53940 non-null  int64  
 1   carat       53940 non-null  float64
 2   cut         53940 non-null  object 
 3   color       53940 non-null  object 
 4   clarity     53940 non-null  object 
 5   depth       53940 non-null  float64
 6   table       53940 non-null  float64
 7   price       53940 non-null  int64  
 8   x           53940 non-null  float64
 9   y           53940 non-null  float64
 10  z           53940 non-null  float64
dtypes: float64(6), int64(2), object(3)
memory usage: 4.5+ MB
```

In [17]:
```
di.describe()
```

Out[17]:

	Unnamed: 0	carat	depth	table	price	x
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000
mean	26970.500000	0.797940	61.749405	57.457184	3932.799722	5.731157
std	15571.281097	0.474011	1.432621	2.234491	3989.439738	1.121761
min	1.000000	0.200000	43.000000	43.000000	326.000000	0.000000
25%	13485.750000	0.400000	61.000000	56.000000	950.000000	4.710000
50%	26970.500000	0.700000	61.800000	57.000000	2401.000000	5.700000
75%	40455.250000	1.040000	62.500000	59.000000	5324.250000	6.540000
max	53940.000000	5.010000	79.000000	95.000000	18823.000000	10.740000

In [18]:
```
di.mean(numeric_only=True)
```

Out[18]:
```
Unnamed: 0    26970.500000
carat             0.797940
depth            61.749405
table            57.457184
price          3932.799722
x                 5.731157
y                 5.734526
z                 3.538734
dtype: float64
```

In [19]:
```
di.groupby('cut')['price'].agg(['min', 'max'])
```

Out[19]:

	min	max
cut		
Fair	337	18574
Good	327	18788
Ideal	326	18806
Premium	326	18823
Very Good	336	18818

In [20]:
```
di.duplicated().sum()
```

Out[20]:
```
0
```

In [ ]:

In [ ]: