In [7]:

```
import pandas as pd
course_name = ['Data Science', 'Machine Learning', 'Big Data', 'Data Engineer']
duration = [2,3,6,4]
df = pd.DataFrame(data = {'course_name' : course_name, 'duration' : duration})
df
```

Out[7]:

| | course_name | duration |
|---|---|---|
| **0** | Data Science | 2 |
| **1** | Machine Learning | 3 |
| **2** | Big Data | 6 |
| **3** | Data Engineer | 4 |

**Q1. Write a code to print the data present in the second row of the dataframe, df.**

In [12]:

```
df.iloc[1]
```

Out[12]:

```
course_name    Machine Learning
duration                      3
Name: 1, dtype: object
```

**Q2. What is the difference between the functions loc and iloc in pandas.DataFrame?**

The main distinction between the two methods is: loc gets rows (and/or columns) with particular labels. iloc gets rows (and/or columns) at integer locations

**Q3. Reindex the given dataframe using a variable, reindex = [3,0,1,2] and store it in the variable, new_df then find the output for both new_df.loc[2] and new_df.iloc[2].**

In [21]:

```
import pandas as pd
import numpy as np
columns = ['column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6']
indices = [1,2,3,4,5,6]
#Creating a dataframe:
df1 = pd.DataFrame(np.random.rand(6,6), columns = columns, index = indices)
df1
```

Out[21]:

| | column_1 | column_2 | column_3 | column_4 | column_5 | column_6 |
|---|---|---|---|---|---|---|
| **1** | 0.866744 | 0.888399 | 0.017106 | 0.026650 | 0.010944 | 0.822894 |
| **2** | 0.714704 | 0.652493 | 0.676323 | 0.717649 | 0.089189 | 0.888430 |
| **3** | 0.672389 | 0.270711 | 0.090705 | 0.364217 | 0.862041 | 0.678693 |
| **4** | 0.771413 | 0.242139 | 0.320329 | 0.372368 | 0.791202 | 0.338436 |
| **5** | 0.313913 | 0.458761 | 0.193215 | 0.798067 | 0.858152 | 0.973502 |
| **6** | 0.292589 | 0.584765 | 0.837104 | 0.845160 | 0.893634 | 0.816158 |

In [15]:

```
new_df = df1.reindex([3,0,1,2])
print(new_df.loc[2])
print(new_df.iloc[2])
```

```
column_1    0.950626
column_2    0.717886
column_3    0.515652
column_4    0.847331
column_5    0.016785
column_6    0.057582
Name: 2, dtype: float64
column_1    0.681934
column_2    0.354616
column_3    0.196357
column_4    0.626691
column_5    0.886148
column_6    0.726332
Name: 1, dtype: float64
```

**Q4. Write a code to find the following statistical measurements for the above dataframe df1:**

(i) mean of each and every column present in the dataframe.

(ii) standard deviation of column, ‘column_2’

In [16]:

```
df1.mean()
```

Out[16]:

```
column_1    0.629442
column_2    0.394127
column_3    0.255347
column_4    0.531029
column_5    0.458118
column_6    0.287794
dtype: float64
```

In [17]:

```
df1['column_2'].std()
```

Out[17]:

```
0.16955279896790482
```

**Q5. Replace the data present in the second row of column, 'column_2' by a string variable then find the mean of column, column_2. If you are getting errors in executing it then explain why.**

In [20]:

```
df1.iloc[1] = 'pwskills'
df1['column_2'].mean()
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_2680\2777297277.py in <module>
      1 df1.iloc[1] = 'pwskills'
----> 2 df1['column_2'].mean()

~\anaconda3\lib\site-packages\pandas\core\generic.py in mean(self, axis, s
kipna, level, numeric_only, **kwargs)
  11122             **kwargs,
  11123         ):
> 11124             return NDFrame.mean(self, axis, skipna, level, numeric
_only, **kwargs)
  11125
  11126         setattr(cls, "mean", mean)

~\anaconda3\lib\site-packages\pandas\core\generic.py in mean(self, axis, s
kipna, level, numeric_only, **kwargs)
  10692         **kwargs,
  10693     ) -> Series | float:
> 10694         return self._stat_function(
  10695             "mean", nanops.nanmean, axis, skipna, level, numeric_o
nly, **kwargs
  10696         )

~\anaconda3\lib\site-packages\pandas\core\generic.py in _stat_function(sel
f, name, func, axis, skipna, level, numeric_only, **kwargs)
  10644                 name, axis=axis, level=level, skipna=skipna, numer
ic_only=numeric_only
  10645             )
> 10646         return self._reduce(
  10647             func, name=name, axis=axis, skipna=skipna, numeric_onl
y=numeric_only
  10648         )

~\anaconda3\lib\site-packages\pandas\core\series.py in _reduce(self, op, n
ame, axis, skipna, numeric_only, filter_type, **kwds)
   4469                 )
   4470             with np.errstate(all="ignore"):
-> 4471                 return op(delegate, skipna=skipna, **kwds)
   4472
   4473     def _reindex_indexer(

~\anaconda3\lib\site-packages\pandas\core\nanops.py in _f(*args, **kwargs)
     91             try:
     92                 with np.errstate(invalid="ignore"):
---> 93                     return f(*args, **kwargs)
     94             except ValueError as e:
     95                 # we want to transform an object array

~\anaconda3\lib\site-packages\pandas\core\nanops.py in f(values, axis, ski
pna, **kwds)
    153                     result = alt(values, axis=axis, skipna=skipna,
**kwds)
    154             else:
--> 155                 result = alt(values, axis=axis, skipna=skipna, **k
wds)
    156
    157             return result
```

```
~\anaconda3\lib\site-packages\pandas\core\nanops.py in new_func(values, ax
is, skipna, mask, **kwargs)
    408             mask = isna(values)
    409
--> 410         result = func(values, axis=axis, skipna=skipna, mask=mask,
**kwargs)
    411
    412         if datetimelike:

~\anaconda3\lib\site-packages\pandas\core\nanops.py in nanmean(values, axi
s, skipna, mask)
    696
    697     count = _get_counts(values.shape, mask, axis, dtype=dtype_coun
t)
--> 698     the_sum = _ensure_numeric(values.sum(axis, dtype=dtype_sum))
    699
    700     if axis is not None and getattr(the_sum, "ndim", False):

~\anaconda3\lib\site-packages\numpy\core\_methods.py in _sum(a, axis, dtyp
e, out, keepdims, initial, where)
     46 def _sum(a, axis=None, dtype=None, out=None, keepdims=False,
     47          initial=_NoValue, where=True):
---> 48     return umr_sum(a, axis, dtype, out, keepdims, initial, where)
     49
     50 def _prod(a, axis=None, dtype=None, out=None, keepdims=False,

TypeError: unsupported operand type(s) for +: 'float' and 'str'
```

we can't find mean of String type

**Q6. What do you understand about the windows function in pandas and list the types of windows functions?**

Windows function in Pandas can be broadly divided into three categories namely- Aggregate, Ranking, and Value. The group by aggregate function can be used to partition and group the entire data frame by some column. We can specify the column name in the parameter of the pandas.

pandas supports 4 types of windowing operations:

1.Rolling window: Generic fixed or variable sliding window over the values.

2.Weighted window: Weighted, non-rectangular window supplied by the scipy.signal library.

3.Expanding window: Accumulating window over the values.

4.Exponentially Weighted window: Accumulating and exponentially weighted window over the values.

**Q7. Write a code to print only the current month and year at the time of answering this question.**

In [22]:

```
from datetime import date
today = date.today()
print(today.year)
print(today.month)
print(today.day)
```

```
2023
4
13
```

**Q8. Write a Python program that takes in two dates as input (in the format YYYY-MM-DD) and calculates the difference between them in days, hours, and minutes using Pandas time delta. The program should prompt the user to enter the dates and display the result.**

In [24]:

```
date1 = pd.to_datetime(input('Insert a Date1:-'))
date2 = pd.to_datetime(input('Insert a Date2:-'))
time1 = pd.Timedelta(days = date1.day,hours = date1.hour,minutes = date1.minute)
time2 = pd.Timedelta(days = date2.day,hours = date2.hour,minutes= date2.minute)
print(time2-time1)
```

```
Insert a Date1:-2023-04-01
Insert a Date2:-2023-04-13
12 days 00:00:00
```

**Q9. Write a Python program that reads a CSV file containing categorical data and converts a specified column to a categorical data type. The program should prompt the user to enter the file path, column name, and category order, and then display the sorted data.**

In [ ]:

```
def cate(path,col):
    data = pd.read_csv(path)
    cate = data['col']
    cate = pd.categorical(cate)
    print(cate.sort_values(ascending = True))


file = input('Enter a File Name:-')
col = input('Enter a Column Name:-')
```

**Q10. Write a Python program that reads a CSV file containing sales data for different products and visualizes the data using a stacked bar chart to show the sales of each product category over time. The program should prompt the user to enter the file path and display the chart.**

In [ ]:

```
import pandas as pd
import matplotlib.pyplot as plt

file_path = input("Enter the file path of the CSV file: ")
df = pd.read_csv(file_path)
sales_data = df.groupby(['Product Category', 'Year']).sum()['Sales']
pivot_table = sales_data.unstack()
pivot_table.plot(kind='bar', stacked=True)
plt.title('Sales by Product Category')
plt.xlabel('Year')
plt.ylabel('Sales')
plt.show()
```

**Q11. You are given a CSV file containing student data that includes the student ID and their test score. Write a Python program that reads the CSV file, calculates the mean, median, and mode of the test scores, and displays the results in a table.**