In [1]:

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

�������� − 2
:

Read the data

In [2]: Out[2]:

```python
file_path="C:\\Users\\omkar\\OneDrive\\Documents\\Data science\\Naresh IT\\D
visa_df=pd.read_csv(file_path)
visa_df
```

case_id continent education_of_employee

has_job_experience requires_job_trainin 0 EZYV01 Asia

High School N

1 EZYV02 Asia Master's Y

2 EZYV03 Asia Bachelor's N

3 EZYV04 Asia Bachelor's N

4 EZYV05 Africa Master's Y

... ...

25475 EZYV25476 Asia Bachelor's Y **25476** EZYV25477

Asia High School Y **25477** EZYV25478 Asia Master's Y

25478 EZYV25479 Asia Master's Y **25479** EZYV25480 Asia

Bachelor's Y

25480 rows × 12 columns

In [3]:

```python
type(visa_df)
```

�������� − 1
:

import packages

```
Out[3]: pandas.core.frame.DataFrame
```

```
In [4]: In [7]:              dir(visa_df)
                             abs ,
                             'add',
                             'add_prefix',
                             'add_suffix',
                             'agg',
                             'aggregate',
                             'align',
                             'all',
                             'any',
                             'apply',
                             'applymap',
                             'asfreq',
                             'asof',
                             'assign',
                             'astype',
                             'at',
                             'at_time',
                             'attrs',
                             'axes',
                             'backfill',
                             'between time'
```

shape=visa_df.shape shape

��ℎ������

```
Out[7]: (25480, 12)
           type(shap
           e)
In [8]:

Out[8]: tuple
```

```
In [10]: In [13]:        print("the number of columns
                         are:",shape[1])

                         the number of observations
                         are: 25480 the number of
                         columns are: 12
```

��������

not callable means brackets are not required

```
print("the number of
observations are:",shape[0])    visa_df.size

Out[13]: 305760
```

```
In [14]:                   # number of rows*
shape[0]*shape[1]          number of columns

Out[14]: 305760
```

������������

```
In [17]:                    categorical column #
data_types=visa_df.dtyp     int or float means
es                          numerical column
data_types
# object means

Out[17]:  case_id  object  continent  object
```

```
         education_of_employee      object
         has_job_experience         object
         requires_job_training      object
         no_of_employees             int64
         yr_of_estab                 int64
         region_of_employment       object
         prevailing_wage           float64
         unit_of_wage               object
         full_time_position         object
         case_status object dtype: object
```

```
In [18]:    type(data_ty
            pes)

Out[18]: pandas.core.series.Series
```

```
In [19]:                # if you apply index the
                        output in list
data_types.index

# any series

Out[19]: Index(['case_id', 'continent', 'education_of_employee', 'has_job_experienc
         e',
            'requires_job_training',   'no_of_employees',   'yr_of_estab',
          'region_of_employment',     'prevailing_wage',     'unit_of_wage',
          'full_time_position', 'case_status'],
           dtype='object')
```

```
In [20]:    data_types.va
            lues

 Out[20]: array([dtype('O'), dtype('O'), dtype('O'), dtype('O'), dtype('O'),
  dtype('int64'), dtype('int64'), dtype('O'), dtype('float64'),  dtype('O'),
                  dtype('O'), dtype('O')], dtype=object)
```

the series is a combination of index and values

if you want seperate both we need to use index and values

values will coming interms of array

array means numpy array

�������� − 1

sepearte numerical columns and categorical columns sepeareatly

```
In [31]:         dict(data_types)['
                 case_id']

Out[31]: dtype('O')
```

```
In [35]: In [36]:                        data_types=visa_df.dtypes
                                         dict1=dict(data_types)
                                         for i in dict1:
                                          if dict1[i]=='O':
                                          cat.append(i)
                                          else:
                                          num.append(i)


                                         #cat=[i for i in dict(data_types) if
                                         dict(data_types)[i]=='object'] #num=[i
                                         for i in dict(data_types) if
                                         dict(data_types)[i]!='object' ]


cat=[]                                   num
num=[]

Out[36]: ['no_of_employees', 'yr_of_estab', 'prevailing_wage']
     [37]:
     cat
In

Out[37]: ['case_id',
          'continent',
          'education_of_employee',
          'has_job_experience',
          'requires_job_training',
          'region_of_employment',
          'unit_of_wage',
          'full_time_position',
          'case_status']
```

ℎ������

top 5 values

```
In [38]: Out[38]:


# dataframe name= visa_df
visa_df.head()
```

case_id continent education_of_employee

In [40]: Out[40]:

has_job_experience requires_job_training no_o 0 EZYV01

Asia High School N N **1** EZYV02 Asia Master's Y N **2** EZYV03

Asia Bachelor's N Y **3** EZYV04 Asia Bachelor's N N **4**

EZYV05 Africa Master's Y N

```
In [39]: Out[39]:
```

```
In [41]:
visa_df.head(2)
```

case_id continent education_of_employee

has_job_experience requires_job_training no_o 0 EZYV01

Asia High School N N **1** EZYV02 Asia Master's Y N

��������

```
visa_df.tail() # last 5 values
```

case_id continent education_of_employee

has_job_experience requires_job_trainin 25475

EZYV25476 Asia Bachelor's Y **25476** EZYV25477 Asia High

School Y **25477** EZYV25478 Asia Master's Y **25478**

EZYV25479 Asia Master's Y **25479** EZYV25480 Asia

Bachelor's Y

���������������

```
visa_df.columns
# retrive all the columns
```

```
Out[41]: Index(['case_id', 'continent', 'education_of_employee', 'has_job_experienc
          e',
             'requires_job_training',   'no_of_employees',   'yr_of_estab',
          'region_of_employment',     'prevailing_wage',     'unit_of_wage',
          'full_time_position', 'case_status'],
           dtype='object')
```

������

```
In [42]:            # number of
len(visa_df)        rows

Out[42]: 25480
```

shape
size
dtypes
we seperated cat and num columns
    series object converted into dictionary
head
tail
columns

```
# inside take , one more argument is axis
# the default value of axis: 0
# axis=0 means rows
# which means 100 ,200,300 row will come
# axis=1 means columns
```

In [43]: Out[43]:

case_id continent education_of_employee

has_job_experience requires_job_training n 100 EZYV101

Asia Master's Y N **200** EZYV201 Asia Doctorate Y N **300**

EZYV301 Asia Master's Y N

��������

```
visa_df.take([100,200,300])

# take only specific index values
In [44]: visa_df.take([100,200,300],axis=1)

        # how many columns: 12
```

```
---------------------------------------------------------------------------
-
IndexError Traceback (most recent call las t)
Cell In[44], line 1
----> 1 visa_df.take([100,200,300],axis=1)

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:3909, in NDFram
e.take(self, indices, axis, **kwargs)
  3833 """
  3834 Return the elements in the given *positional* indices along an axi
s.
  3835
  (...)
  3904 3 lion mammal 80.5
  3905 """
  3907 nv.validate_take((), kwargs)
-> 3909 return self._take(indices, axis)

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:3932, in NDFram
e._take(self, indices, axis, convert_indices)
  3924 if (
  3925 axis == 0
  3926 and indices.ndim == 1
  3927 and using_copy_on_write()
           3928 and is_range_indexer(indices, len(self))
  3929 ):
  3930 return self.copy(deep=None)
-> 3932 new_data = self._mgr.take(
  3933 indices,
  3934 axis=self._get_block_manager_axis(axis),
  3935 verify=True,
  3936 convert_indices=convert_indices,
  3937 )
  3938 return self._constructor(new_data).__finalize__(self, method="tak
e")

File ~\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:960,
in BaseBlockManager.take(self, indexer, axis, verify, convert_indices)
958 n = self.shape[axis]
  959 if convert_indices:
--> 960 indexer = maybe_convert_indices(indexer, n, verify=verify)
 962 new_labels = self.axes[axis].take(indexer)
  963 return self.reindex_indexer(
  964 new_axis=new_labels,
  965 indexer=indexer,
  (...)
  968 copy=None,
  969 )

File ~\anaconda3\Lib\site-packages\pandas\core\indexers\utils.py:284, in m
aybe_convert_indices(indices, n, verify)
  282 mask = (indices >= n) | (indices < 0)
  283 if mask.any():
--> 284 raise IndexError("indices are out-of-bounds")  285
 return indices

IndexError: indices are out-of-bounds
```
```
In [45]: Out[45]:
```

In [46]: Out[46]:

In [50]: Out[50]:

```
visa_df.take([1,2,3])
```

case_id continent education_of_employee

has_job_experience requires_job_training no_o 1 EZYV02

Asia Master's Y N **2** EZYV03 Asia Bachelor's N Y **3** EZYV04

Asia Bachelor's N N

```
visa_df.take([1,2,3],axis=1)
```

	continent	education_of_employee	has_job_experience
0	Asia	High School	N
1	Asia	Master's	Y
2	Asia	Bachelor's	N
3	Asia	Bachelor's	N
4	Africa	Master's	Y

...	...	...	...
25475	Asia	Bachelor's	Y
25476	Asia	High School	Y
25477	Asia	Master's	Y
25478	Asia	Master's	Y
25479	Asia	Bachelor's	Y

25480 rows × 3 columns

��������� – 2

I want 150,300,450 rows from 5,8,12 by using take

```
d1=visa_df.take([5,8,11],axis=1)
d1.take([150,300,450])
# python index start with zero
# we have total 12 columns
# but in python 12th column index is 11
```

	no_of_employees	prevailing_wage	case_status
150	50351	529.1105	Denied
300	3268	101371.2100	Certified
450	1543	78402.7200	Denied

In [51]: Out[51]:

In [ ]:

In [54]: Out[54]:

In [55]: Out[55]:

```
visa_df.take([5,8,11],axis=1).take([150,300
,450])
```

no_of_employees prevailing_wage case_status

150 50351 529.1105 Denied

300 3268 101371.2100 Certified

450 1543 78402.7200 Denied

������-��������

```
#data.iloc(<rows>,<columns>)
#data.iloc(start:end,start:end)

# assume that i want 20:25 rows
# 3:6 columns


visa_df.iloc[20:25,3:6]

# last=end-1 25-1=24 6-1=5
```

has_job_experience requires_job_training no_of_employees

20 N N 880

21 Y N 1706

22 Y N 2878

23 N N 1517

24 Y N 241

```
visa_df.iloc[20:25] # all the columns
```

case_id continent education_of_employee

has_job_experience requires_job_training no_ 20 EZYV21

Asia Master's N N **21** EZYV22 North

America Master's Y N **22** EZYV23 Asia Master's Y N **23**

EZYV24 North

America High School N N **24** EZYV25 Europe Doctorate Y N

In [56]: Out[56]:

In [58]: Out[58]:

In [59]: Out[59]:

In [61]: Out[61]:

```
In [63]:
list1=[100,200,300]
visa_df.iloc[list1] #visa_df.take(list1)
```

case_id continent education_of_employee

has_job_experience requires_job_training n 100 EZYV101

Asia Master's Y N **200** EZYV201 Asia Doctorate Y N **300**

EZYV301 Asia Master's Y N

```
rows=[100,200,300]
columns=[5,8,11]
visa_df.iloc[rows,columns]

#visa_df.take(columns,axis=1).take(rows)
```

no_of_employees prevailing_wage case_status

100 2227 28243.79 Certified

200 3282 74441.11 Certified

300 3268 101371.21 Certified

```
# case status : 100,200 rows
rows=[100,200]
columns=[11]
visa_df.iloc[rows,columns]

# 100 columns
```

```
directly we can provide column name
```

case_status

100 Certified

200 Certified

case_status

100 Certified

200 Certified

```
visa_df.iloc[rows,[11]] # we need to
provide number
visa_df.loc[rows,['case_status']] #
```

```
visa_df.columns[:3]
```

```
 Out[63]: Index(['case_id', 'continent', 'education_of_employee'], dtype='object')
In [68]: Out[68]:
```

```
In [ ]:
visa_df.loc[[200],['case_id',
'continent', 'education_of_employee']]
visa_df.iloc[[200],[1,2,3]]
```

continent education_of_employee has_job_experience

200 Asia Doctorate Y

compare to take , iloc is better
compare to iloc, loc is better
in take we need provide axis for columns
in iloc and loc no need provide axis
in iloc we need provide column index number
if there are huge columns are there, it is not good
the count the specific column number instead of
that loc function directly will take the column name