Import required packages

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Read the data

In [2]:
```
visa_df=pd.read_csv(r"C:\Users\omkar\OneDrive\Documents\Data science\Naresh IT\N
visa_df
```

Out[2]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_t
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

head

- Top 5 rows

In [3]:
```
visa_df.head()
```

Out[3]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training
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 [5]:
```
visa_df.head(3)
```

Out[5]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training
0	EZYV01	Asia	High School	N	N
1	EZYV02	Asia	Master's	Y	N
2	EZYV03	Asia	Bachelor's	N	Y

tail

- last 5 rows

In [4]:
```
visa_df.tail()
```

Out[4]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_t
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	

shape

- number of rows and number of columns

In [5]:
```
visa_df.shape
```

Out[5]:
```
(25480, 12)
```

In [6]:
```
print(f"the number of rows are {visa_df.shape[0]}")
print(f"the number of rows are {visa_df.shape[1]}")
```

```
the number of rows are 25480
the number of rows are 12
```

size

- number of rows * number of columns

In [7]:
```
visa_df.size
```

Out[7]:
```
305760
```

In [10]:
```
25480*12
```

Out[10]:
```
305760
```

columns

In [8]:
```
visa_df.columns
```

Out[8]:
```
Index(['case_id', 'continent', 'education_of_employee', 'has_job_experience',
       'requires_job_training', 'no_of_employees', 'yr_of_estab',
       'region_of_employment', 'prevailing_wage', 'unit_of_wage',
       'full_time_position', 'case_status'],
      dtype='object')
```

dtypes

In [9]:
```
visa_df.dtypes

# object means categorical
# int float mean numerical
```

Out[9]:
```
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
```

- understanding data types are very important
- visa_df type is: **data frame**
- visa_df.values is: **series**
- Series objects can convert into dictionary
- after that you can use dictionary methods

In [10]:
```
type(visa_df)
```

Out[10]:
```
pandas.core.frame.DataFrame
```

In [11]:
```
type(visa_df.dtypes)
```

Out[11]:
```
pandas.core.series.Series
```

task

- Create categorical list
- Create numerical column list

idea

- convert series type into dictionary
- iterate through loop
- apply the condition get the result in the list

In [12]:
```
dtypes=dict(visa_df.dtypes)
dtypes
```

Out[12]:
```
{'case_id': dtype('O'),
 'continent': dtype('O'),
 'education_of_employee': dtype('O'),
 'has_job_experience': dtype('O'),
 'requires_job_training': dtype('O'),
 'no_of_employees': dtype('int64'),
 'yr_of_estab': dtype('int64'),
 'region_of_employment': dtype('O'),
 'prevailing_wage': dtype('float64'),
 'unit_of_wage': dtype('O'),
 'full_time_position': dtype('O'),
 'case_status': dtype('O')}
```

In [13]:
```
cat_cols=[key for key,value in dtypes.items() if value=='object']
num_cols=[key for key,value in dtypes.items() if value!='object']
```

select-data-types

- select data types will give a data frame of desired data types
- the result also a data frame
- all above options like head tail columns

In [14]:
```
visa_df.select_dtypes(include='object')
```

Out[14]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_t
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 × 9 columns

In [15]:
```
visa_df.select_dtypes(include='object').columns
```

Out[15]:
```
Index(['case_id', 'continent', 'education_of_employee', 'has_job_experience',
       'requires_job_training', 'region_of_employment', 'unit_of_wage',
       'full_time_position', 'case_status'],
      dtype='object')
```

In [16]:
```
visa_df.select_dtypes(exclude='object').columns
```

Out[16]:
```
Index(['no_of_employees', 'yr_of_estab', 'prevailing_wage'], dtype='object')
```

is-null

- is null used to identify the any missing values are available
- is null means we are asking a qn to the computer
- It will give True or False
- True : yes value is missed
- False : No value is not missed

Note

If we see bound method means add the brackets

In [18]:
```
visa_df.isnull()
```

Out[18]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_trai
0	False	False	False	False	F
1	False	False	False	False	F
2	False	False	False	False	F
3	False	False	False	False	F
4	False	False	False	False	F
...	...	...	...	...	
25475	False	False	False	False	F
25476	False	False	False	False	F
25477	False	False	False	False	F
25478	False	False	False	False	F
25479	False	False	False	False	F

25480 rows × 12 columns

In [19]:
```
visa_df.isna()
```

Out[19]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_trai
0	False	False	False	False	F
1	False	False	False	False	F
2	False	False	False	False	F
3	False	False	False	False	F
4	False	False	False	False	F
...	...	...	...	...	
25475	False	False	False	False	F
25476	False	False	False	False	F
25477	False	False	False	False	F
25478	False	False	False	False	F
25479	False	False	False	False	F

25480 rows × 12 columns

In [20]:
```
visa_df.isnull().sum()
```

Out[20]:
```
case_id                 0
continent               0
education_of_employee   0
has_job_experience      0
requires_job_training   0
no_of_employees         0
yr_of_estab             0
region_of_employment    0
prevailing_wage         0
unit_of_wage            0
full_time_position      0
case_status             0
dtype: int64
```

drop-duplicates

- if any duplicate rows are there then we can drop it

In [21]:
```
visa_df.drop_duplicates()
```

Out[21]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_t
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 [22]:
```
visa_df.drop_duplicates().all()
```

Out[22]:
```
case_id                 True
continent               True
education_of_employee   True
has_job_experience      True
requires_job_training   True
no_of_employees         True
yr_of_estab             True
region_of_employment    True
prevailing_wage         True
unit_of_wage            True
full_time_position      True
case_status             True
dtype: bool
```

info

In [23]:
```
visa_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25480 entries, 0 to 25479
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   case_id                25480 non-null  object 
 1   continent              25480 non-null  object 
 2   education_of_employee  25480 non-null  object 
 3   has_job_experience     25480 non-null  object 
 4   requires_job_training  25480 non-null  object 
 5   no_of_employees        25480 non-null  int64  
 6   yr_of_estab            25480 non-null  int64  
 7   region_of_employment   25480 non-null  object 
 8   prevailing_wage        25480 non-null  float64
 9   unit_of_wage           25480 non-null  object 
 10  full_time_position     25480 non-null  object 
 11  case_status            25480 non-null  object 
dtypes: float64(1), int64(2), object(9)
memory usage: 2.3+ MB
```

take

- take is used to extract the particular index of data
- take has one parameter called axis
- axis=1 represents columns
- axis=0 represnts rows
- by default axis=0 is there
- In below example we given 100,200,300 and we did not provide axis values
- It understood we are talking about rows

In [24]:
```
visa_df.take([100,200,300])
```

Out[24]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
100	EZYV101	Asia	Master's	Y	
200	EZYV201	Asia	Doctorate	Y	
300	EZYV301	Asia	Master's	Y	

In [25]:
```
visa_df.take([100,200,300],axis=1)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[25], line 1
----> 1 visa_df.take([100,200,300],axis=1)

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:4068, in NDFrame.take(self, indices, axis, **kwargs)
   4063     # We can get here with a slice via DataFrame.__getitem__
   4064     indices = np.arange(
   4065         indices.start, indices.stop, indices.step, dtype=np.intp
   4066     )
-> 4068 new_data = self._mgr.take(
   4069     indices,
   4070     axis=self._get_block_manager_axis(axis),
   4071     verify=True,
   4072 )
   4073 return self._constructor_from_mgr(new_data, axes=new_data.axes).__finalize__(
   4074     self, method="take"
   4075 )

File ~\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:874, in BaseBlockManager.take(self, indexer, axis, verify)
    871 # Caller is responsible for ensuring indexer annotation is accurate
    873 n = self.shape[axis]
--> 874 indexer = maybe_convert_indices(indexer, n, verify=verify)
    876 new_labels = self.axes[axis].take(indexer)
    877 return self.reindex_indexer(
    878     new_axis=new_labels,
    879     indexer=indexer,
   (...)
    882     copy=None,
    883 )

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

IndexError: indices are out-of-bounds
```

In [26]:
```
visa_df.take([2,5,7],axis=1)
```

Out[26]:

	education_of_employee	no_of_employees	region_of_employment
0	High School	14513	West
1	Master's	2412	Northeast
2	Bachelor's	44444	West
3	Bachelor's	98	West
4	Master's	1082	South
...	...	...	...
25475	Bachelor's	2601	South
25476	High School	3274	Northeast
25477	Master's	1121	South
25478	Master's	1918	West
25479	Bachelor's	3195	Midwest

25480 rows × 3 columns

In [27]:
```
visa_df.take([100:105]) # Not working
```

```
  Cell In[27], line 1
    visa_df.take([100:105])
                      ^
SyntaxError: invalid syntax
```

In [ ]:
```
# qn
# I want to rows 100 200 300 and columns :2,5,7
# with single syntax not possibile
```

In [29]:
```
visa_df.take([100,200,300]).take([2,5,7],axis=1)
```

Out[29]:

	education_of_employee	no_of_employees	region_of_employment
100	Master's	2227	Northeast
200	Doctorate	3282	West
300	Master's	3268	Midwest

iloc

In [ ]:
```
# visa_df.iloc[<rows>,<columns>]
# visa_df.iloc[<start:end>,<start:end>]
# rows = [100,200,300]
# columns= [2,5,7]
```

In [30]:
```
visa_df.iloc[[100,200,300],[2,5,7]]
```

Out[30]:

	education_of_employee	no_of_employees	region_of_employment
100	Master's	2227	Northeast
200	Doctorate	3282	West
300	Master's	3268	Midwest

In [31]:
```
visa_df.iloc[[100,200,300]] # same like take
```

Out[31]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
100	EZYV101	Asia	Master's	Y	
200	EZYV201	Asia	Doctorate	Y	
300	EZYV301	Asia	Master's	Y	

In [32]:
```
visa_df.iloc[100:105,[2,5,7]]
```

Out[32]:

	education_of_employee	no_of_employees	region_of_employment
100	Master's	2227	Northeast
101	Master's	334	Midwest
102	Bachelor's	224	Midwest
103	Doctorate	367	West
104	Master's	306	Northeast

In [33]:
```
visa_df.iloc[100:105,2:5]
```

Out[33]:

	education_of_employee	has_job_experience	requires_job_training
100	Master's	Y	N
101	Master's	Y	N
102	Bachelor's	Y	N
103	Doctorate	Y	N
104	Master's	Y	N

In [36]:
```
visa_df.iloc[100:105,2]
```

Out[36]:
```
100      Master's
101      Master's
102    Bachelor's
103     Doctorate
104      Master's
Name: education_of_employee, dtype: object
```

In [37]:
```
visa_df.iloc[100:105,[2]]
```

Out[37]:

	education_of_employee
100	Master's
101	Master's
102	Bachelor's
103	Doctorate
104	Master's

In [ ]:
```
visa_df.iloc[[100]]
visa_df.iloc[[100],:]
visa_df.iloc[[100],0:]
visa_df.iloc[100]
visa_df.iloc[100,:]
```

In [39]:
```
visa_df.iloc[[100]]
```

Out[39]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
100	EZYV101	Asia	Master's	Y	

Note

- No bracket then series
- With Saqure bracket dataframe

In [40]:
```
# no of employess data
visa_df.iloc[:,[5]]
```

Out[40]:

	no_of_employees
0	14513
1	2412
2	44444
3	98
4	1082
...	...
25475	2601
25476	3274
25477	1121
25478	1918
25479	3195

25480 rows × 1 columns

In [44]:
```
type(visa_df.columns)
```

Out[44]:
```
pandas.core.indexes.base.Index
```

In [46]:
```
cols=list(visa_df.columns)
cols.index('no_of_employees')
```

Out[46]:
```
5
```

In [43]:
```
l=['A','B','D']
l.index('D')
```

Out[43]:
```
2
```

loc

- loc will take directly columns names

In [48]:
```
visa_df.loc[[100,200,300],['no_of_employees']]
```

Out[48]:

	no_of_employees
100	2227
200	3282
300	3268

- head
- tail

- shape
- size
- columns
- dtypes
- select dtypes
- is null
- is na
- drop duplicates
- info
- take
- iloc
- loc

In [50]:
```
visa_df.loc[100:300,['no_of_employees','continent']]
```

Out[50]:

	no_of_employees	continent
100	2227	Asia
101	334	Asia
102	224	Asia
103	367	Asia
104	306	Asia
...	...	...
296	1017	Europe
297	1624	Asia
298	3891	Asia
299	2009	Asia
300	3268	Asia

201 rows × 2 columns

In [ ]:
```
```