Lab Exercise 03

1. Slicing the Data Frame

In [1]:
```
import pandas as pd
import csv
```

In [2]:
```
a=pd.read_csv("bounce rate.csv")
```

In [3]:
```
a.head(2)
```

Out[3]:

	Unnamed: 0	Bounce rate	Day	Visitors
0	0	20	1	1000
1	1	20	2	700

In [4]:
```
a.tail(2)
```

Out[4]:

	Unnamed: 0	Bounce rate	Day	Visitors
4	4	10	5	400
5	5	34	6	350

2. Concatenation of two frames

a. Concatenate following two dataframes

```
In [5]: df1 = pd.DataFrame({"HPI":[80,90,70,60],"Int_Rate":[2,1,2,3], "IND_GDP":[50,45,45,67]}, index=[2001, 2002,2003,2004])

        df2 = pd.DataFrame({"HPI":[80,90,70,60],"Int_Rate":[2,1,2,3],"IND_GDP":[50,45,45,67]}, index=[2005, 2006,2007,2008])
        df=pd.concat([df1,df2])
```

```
In [6]: df
```

```
Out[6]:
```

	HPI	Int_Rate	IND_GDP
2001	80	2	50
2002	90	1	45
2003	70	2	45
2004	60	3	67
2005	80	2	50
2006	90	1	45
2007	70	2	45
2008	60	3	67

b. Concatenate following two dataframe using axis

```
In [7]: pd.concat([df1,df2],axis=1)
```

Out[7]:

	HPI	Int_Rate	IND_GDP	HPI	Int_Rate	IND_GDP
2001	80.0	2.0	50.0	NaN	NaN	NaN
2002	90.0	1.0	45.0	NaN	NaN	NaN
2003	70.0	2.0	45.0	NaN	NaN	NaN
2004	60.0	3.0	67.0	NaN	NaN	NaN
2005	NaN	NaN	NaN	80.0	2.0	50.0
2006	NaN	NaN	NaN	90.0	1.0	45.0
2007	NaN	NaN	NaN	70.0	2.0	45.0
2008	NaN	NaN	NaN	60.0	3.0	67.0

3.Change the index

```
In [8]: df= pd.DataFrame({"Day":[1,2,3,4], "Visitors":[200, 100,230,300],"Bounce_Rate":[20,45,60,10]})
```

```
In [9]: df
```

Out[9]:

	Day	Visitors	Bounce_Rate
0	1	200	20
1	2	100	45
2	3	230	60
3	4	300	10

In [10]:
```
df.set_index("Day")
```

Out[10]:
```
     Visitors  Bounce_Rate
Day
  1       200           20
  2       100           45
  3       230           60
  4       300           10
```

4. Change the Column Headers from “Visitors” to “Users”

In [11]:
```
df.rename(columns = {'Visitors':'Users'}, inplace = True)
```

In [12]:
```
df
```

Out[12]:
```
   Day  Users  Bounce_Rate
0    1    200           20
1    2    100           45
2    3    230           60
3    4    300           10
```

5. A data frame is made from the csv file and the data frame is sorted in ascending order of Names of Players.

In [13]:
```
single = pd.read_csv("nba.csv")
single
```

Out[13]:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
...	...	...	...	...	...	...	...	...	...
453	Shelvin Mack	Utah Jazz	8.0	PG	26.0	6-3	203.0	Butler	2433333.0
454	Raul Neto	Utah Jazz	25.0	PG	24.0	6-1	179.0	NaN	900000.0
455	Tibor Pleiss	Utah Jazz	21.0	C	26.0	7-3	256.0	NaN	2900000.0
456	Jeff Withey	Utah Jazz	24.0	C	26.0	7-0	231.0	Kansas	947276.0
457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

458 rows × 9 columns

In [14]:
```python
single.sort_values(by=['Name'])
```

Out[14]:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
152	Aaron Brooks	Chicago Bulls	0.0	PG	31.0	6-0	161.0	Oregon	2250000.0
356	Aaron Gordon	Orlando Magic	0.0	PF	20.0	6-9	220.0	Arizona	4171680.0
328	Aaron Harrison	Charlotte Hornets	9.0	SG	21.0	6-6	210.0	Kentucky	525093.0
404	Adreian Payne	Minnesota Timberwolves	33.0	PF	25.0	6-10	237.0	Michigan State	1938840.0
312	Al Horford	Atlanta Hawks	15.0	C	30.0	6-10	245.0	Florida	12000000.0
...	...	...	...	...	...	...	...	...	...
270	Xavier Munford	Memphis Grizzlies	14.0	PG	24.0	6-3	180.0	Rhode Island	NaN
402	Zach LaVine	Minnesota Timberwolves	8.0	PG	21.0	6-5	189.0	UCLA	2148360.0
271	Zach Randolph	Memphis Grizzlies	50.0	PF	34.0	6-9	260.0	Michigan State	9638555.0
237	Zaza Pachulia	Dallas Mavericks	27.0	C	32.0	6-11	275.0	NaN	5200000.0
457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

458 rows × 9 columns

6. Find the positions of numbers that are multiples of 3 from ser.

In [15]:
```python
import numpy as np
```

```
In [16]: ser = pd.Series(np.random.randint(1, 10, 7))
         ser
```

```
Out[16]: 0    4
         1    3
         2    8
         3    8
         4    7
         5    5
         6    9
         dtype: int32
```

```
In [17]: ser
```

```
Out[17]: 0    4
         1    3
         2    8
         3    8
         4    7
         5    5
         6    9
         dtype: int32
```

7. How to extract items at given positions from a series?

```
In [18]: ser = pd.Series(list('abcdefghijklmnopqrstuvwxyz'))
         pos = [0, 4, 8, 14, 20]
```

```
In [19]: ser.take(pos)
```

```
Out[19]: 0     a
         4     e
         8     i
         14    o
         20    u
         dtype: object
```

8. How to convert the first character of each element in a series to uppercase?

```
In [20]: ser = pd.Series(['how', 'to', 'kick', 'ass?'])
```

```
In [21]: ser.map(lambda x: x.title())
```

```
Out[21]: 0     How
         1      To
         2    Kick
         3    Ass?
         dtype: object
```

9. How to compute difference of differences between consecutive numbers of a series?

```
In [22]: ser = pd.Series([1, 3, 6, 10, 15, 21, 27, 35])
```

```
In [23]: print(ser.diff().tolist())
         print(ser.diff().diff().tolist())
```

```
[nan, 2.0, 3.0, 4.0, 5.0, 6.0, 6.0, 8.0]
[nan, nan, 1.0, 1.0, 1.0, 1.0, 0.0, 2.0]
```

10. How to get the day of month, week number, day of year and day of week from a series of date strings?

```
In [24]: ser = pd.Series(['01 Jan 2010', '02-02-2011', '20120303', '2013/04/04', '2014-05-05',
         '2015-06-06T12:20'])
```

```
In [25]: from dateutil.parser import parse
         ser_ts = ser.map(lambda x: parse(x))
```

```
In [26]: print("Date: ", ser_ts.dt.day.tolist())      # day of month
```

```
Date:  [1, 2, 3, 4, 5, 6]
```

```
In [27]: print("Week number: ", ser_ts.dt.weekofyear.tolist())     # week number
```

```
Week number:  [53, 5, 9, 14, 19, 23]

<ipython-input-27-eac522016a6d>:1: FutureWarning: Series.dt.weekofyear and Series.dt.week have been deprecated.  Please
use Series.dt.isocalendar().week instead.
  print("Week number: ", ser_ts.dt.weekofyear.tolist())     # week number
```

```
In [28]: print("Day number of year: ", ser_ts.dt.dayofyear.tolist())      # day of year
```

```
Day number of year:  [1, 33, 63, 94, 125, 157]
```

```
In [29]: print("Day of week: ", ser_ts.dt.day_name())        # day of week
```

```
Day of week:  0       Friday
1    Wednesday
2     Saturday
3     Thursday
4       Monday
5     Saturday
dtype: object
```

11. How to filter words that contain at least 2 vowels from a series?

```
In [30]: ser = pd.Series(['Apple', 'Orange', 'Plan', 'Python', 'Money'])
In [31]: from collections import Counter
         mask = ser.map(lambda x: sum([Counter(x.lower()).get(i, 0) for i in list('aeiou')]) >= 2)
         ser[mask]
```

```
Out[31]: 0     Apple
         1    Orange
         4     Money
         dtype: object
```

12. How to filter valid emails from a series?

```
In [32]: emails = pd.Series(['buying books at amazom.com', 'rameses@egypt.com', 'matt@t.co',
         'narendra@modi.com'])
         pattern ='[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,4}'
```

```
In [33]: import re
         pattern ='[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,4}'
         mask = emails.map(lambda x: bool(re.match(pattern, x)))
         emails[mask]
```

```
Out[33]: 1    rameses@egypt.com
         2            matt@t.co
         3    narendra@modi.com
         dtype: object
```

13. How to find all the local maxima (or peaks) in a numeric series?

```
In [34]: ser = pd.Series([2, 10, 3, 4, 9, 10, 2, 7, 3])
```

```
In [35]: dd = np.diff(np.sign(np.diff(ser)))
         peak_locs = np.where(dd == -2)[0] + 1
         peak_locs
```

```
Out[35]: array([1, 5, 7], dtype=int64)
```

14. How to change the order of columns of a dataframe?

a) In df, interchange columns 'a' and 'c'.

In [36]:
```
df = pd.DataFrame(np.arange(20).reshape(-1, 5), columns=list('abcde'))
df[list('cbade')]
```

Out[36]:

	c	b	a	d	e
0	2	1	0	3	4
1	7	6	5	8	9
2	12	11	10	13	14
3	17	16	15	18	19

b) Create a generic function to interchange two columns, without hardcoding column names.

In [37]:
```
def switch_columns(df, col1=None, col2=None):
    colnames = df.columns.tolist()
    i1, i2 = colnames.index(col1), colnames.index(col2)
    colnames[i2], colnames[i1] = colnames[i1], colnames[i2]
    return df[colnames]
df1 = switch_columns(df, 'a', 'c')
df1
```

Out[37]:

	c	b	a	d	e
0	2	1	0	3	4
1	7	6	5	8	9
2	12	11	10	13	14
3	17	16	15	18	19

c) Sort the columns in reverse alphabetical order, that is colume 'e' first through column 'a' last.

In [38]:
```
df.sort_index(axis=1, ascending=False, inplace=True)
df
```

Out[38]:

	e	d	c	b	a
0	4	3	2	1	0
1	9	8	7	6	5
2	14	13	12	11	10
3	19	18	17	16	15

15. How to create a new column that contains the row number of nearest column by euclidean distance?

In [41]:
```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randint(1,100, 40).reshape(10, -1), columns=list('pqrs'), index=list('abcdefghij'))

nearest_rows = []
nearest_distance = []

for i, row in df.iterrows():
    curr = row
    rest = df.drop(i)
    e_dists = {}
    for j, contestant in rest.iterrows():
        e_dists.update({j: round(np.linalg.norm(curr.values - contestant.values))})
    nearest_rows.append(max(e_dists, key=e_dists.get))
    nearest_distance.append(max(e_dists.values()))

df['nearest_row'] = nearest_rows
df['dist'] = nearest_distance
```

```
In [42]: df
```

Out[42]:

	p	q	r	s	nearest_row	dist
a	66	53	79	92	b	101
b	24	51	35	11	f	105
c	16	33	21	50	h	114
d	67	79	11	31	i	106
e	7	97	86	54	f	131
f	93	11	44	79	i	136
g	16	2	47	65	h	123
h	85	91	90	42	g	123
i	6	88	97	31	f	136
j	47	51	22	21	e	94

```
In [ ]:
```