In [1]:
```
import pandas as pd
import numpy as np
```

#Write a Pandas program to select the rows where the number of attempts in the examination is greater than 2. Sample Python dictionary data and list labels: exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'], 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']} labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

In [10]:
```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df=pd.DataFrame(exam_data,index=labels)
df[df['attempts']>2]
```

Out[10]:

	name	score	attempts	qualify
b	Dima	9.0	3	no
d	James	NaN	3	no
f	Michael	20.0	3	yes

Write a Pandas program to count the number of rows and columns of a DataFrame. Go to the editor Sample Python dictionary data and list labels: exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'], 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']} labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'] Expected Output: Number of Rows: 10 Number of Columns: 4

In [12]:
```
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df=pd.DataFrame(exam_data,index=labels)
df.shape
```

Out[12]:
```
(10, 4)
```

Write a Pandas program to select the rows where the score is missing, i.e. is NaN. Go to the editor Sample Python dictionary data and list labels: exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'], 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']} labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'] Expected Output: Rows where score is missing: attempts name qualify score d 3 James no NaN h 1 Laura no NaN

In [16]:
```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df=pd.DataFrame(exam_data,index=labels)
df[df['score'].isnull()]
```

Out[16]:

	name	score	attempts	qualify
d	James	NaN	3	no
h	Laura	NaN	1	no

Write a Pandas program to select the rows the score is between 15 and 20 (inclusive). Go to the editor Sample Python dictionary data and list labels: exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'], 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']} labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'] Expected Output: Rows where score between 15 and 20 (inclusive): attempts name qualify score c 2 Katherine yes 16.5 f 3 Michael yes 20.0 j 1 Jonas yes 19.0

In [19]:
```
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df=pd.DataFrame(exam_data,index=labels)
df[(df['score']>15) & (df['score']<=20)]
```

Out[19]:

	name	score	attempts	qualify
c	Katherine	16.5	2	yes
f	Michael	20.0	3	yes
j	Jonas	19.0	1	yes

Write a Pandas program to select the rows where number of attempts in the examination is less than 2 and score greater than 15. Go to the editor Sample Python dictionary data and list labels: exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'], 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']} labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'] Expected Output: Number of attempts in the examination is less than 2 and score greater than 15 : name score attempts qualify j Jonas 19.0 1 yes

In [21]:
```
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)

df[(df['attempts']<2) & (df['score']>15)]
```

Out[21]:

	name	score	attempts	qualify
j	Jonas	19.0	1	yes

Write a Pandas program to change the score in row 'd' to 11.5. Go to the editor Sample Python dictionary data and list labels: exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'], 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']} labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'] Expected Output: Change the score in row 'd' to 11.5: attempts name qualify score a 1 Anastasia yes 12.5 b 3 Dima no 9.0 c 2 Katherine yes 16.5 ... i 2 Kevin no 8.0 j 1 Jonas yes 19.0

In [24]:
```
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
df.loc['d','score']=11.5
df
```

Out[24]:

	name	score	attempts	qualify
a	Anastasia	12.5	1	yes
b	Dima	9.0	3	no
c	Katherine	16.5	2	yes
d	James	11.5	3	no
e	Emily	9.0	2	no
f	Michael	20.0	3	yes
g	Matthew	14.5	1	yes
h	Laura	NaN	1	no
i	Kevin	8.0	2	no
j	Jonas	19.0	1	yes

Write a Pandas program to calculate the sum of the examination attempts by the students. Go to the editor Sample Python dictionary data and list labels: exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'], 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']} labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'] Expected Output: Sum of the examination attempts by the students: 19

In [26]:
```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df=pd.DataFrame(exam_data,index=labels)
df['attempts'].sum()
```

Out[26]:
```
19
```

Write a Pandas program to calculate the mean score for each different student in DataFrame. Go to the editor Sample Python dictionary data and list labels: exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'], 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']} labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'] Expected Output: Mean score for each different student in data frame: 13.5625

In [27]:
```
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
```

```
df = pd.DataFrame(exam_data , index=labels)
df['score'].mean()
```

Out[27]:
```
13.5625
```

Write a Pandas program to append a new row 'k' to data frame with given values for each column. Now delete the new row and return the original DataFrame. Go to the editor Sample Python dictionary data and list labels: exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'], 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']} labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'] Values for each column will be: name : "Suresh", score: 15.5, attempts: 1, qualify: "yes", label: "k" Expected Output: Append a new row: Print all records after insert a new record: attempts name qualify score a 1 Anastasia yes 12.5 b 3 Dima no 9.0 ..... j 1 Jonas yes 19.0 k 1 Suresh yes 15.5 Delete the new row and display the original rows: attempts name qualify score a 1 Anastasia yes 12.5 b 3 Dima no 9.0 ..... i 2 Kevin no 8.0 j 1 Jonas yes 19.0

In [40]:
```
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
df.loc['d']=['James',np.nan,3,'no']
df.loc['k'] = ['Suresh', 15.5 ,1,'yes']
df.drop('k')
```

Out[40]:

	name	score	attempts	qualify
a	Anastasia	12.5	1	yes
b	Dima	9.0	3	no
c	Katherine	16.5	2	yes
d	James	NaN	3	no
e	Emily	9.0	2	no
f	Michael	20.0	3	yes
g	Matthew	14.5	1	yes
h	Laura	NaN	1	no
i	Kevin	8.0	2	no
j	Jonas	19.0	1	yes

Write a Pandas program to sort the DataFrame first by 'name' in descending order, then by 'score' in ascending order. Go to the editor Sample Python dictionary data and list labels: exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'], 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']} labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'] Values for each column will be: name : "Suresh", score: 15.5, attempts: 1, qualify: "yes", label: "k" Expected Output: Orginal rows: name score attempts qualify a Anastasia 12.5 1 yes b Dima 9.0 3 no c Katherine 16.5 2 yes d James NaN 3 no e Emily 9.0 2 no f Michael 20.0 3 yes g Matthew 14.5 1 yes h Laura NaN 1 no i Kevin 8.0 2 no j Jonas 19.0 1 yes Sort the data frame first by 'name' in descending order, then by 'score' in ascending order: name score attempts qualify a Anastasia 12.5 1 yes b Dima 9.0 3 no c Katherine 16.5 2 yes d James NaN 3 no e Emily 9.0 2 no f Michael 20.0 3 yes g Matthew 14.5 1 yes h Laura NaN 1 no i Kevin 8.0 2 no j Jonas 19.0 1 yes

In [43]:
```
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
```

```
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
df.sort_values(by=['name','score'],ascending=[False,True])
```

Out[43]:

	name	score	attempts	qualify
f	Michael	20.0	3	yes
g	Matthew	14.5	1	yes
h	Laura	NaN	1	no
i	Kevin	8.0	2	no
c	Katherine	16.5	2	yes
j	Jonas	19.0	1	yes
d	James	NaN	3	no
e	Emily	9.0	2	no
b	Dima	9.0	3	no
a	Anastasia	12.5	1	yes

Write a Pandas program to replace the 'qualify' column contains the values 'yes' and 'no' with True and False. Go to the editor Sample Python dictionary data and list labels: exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'], 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']} labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'] Expected Output: Replace the 'qualify' column contains the values 'yes' and 'no' with T rue and False: attempts name qualify score a 1 Anastasia True 12.5 b 3 Dima False 9.0 ..... i 2 Kevin False 8.0 j 1 Jonas True 19.0

In [53]:
```
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
df['qualify']=df['qualify'].map({'yes':True,'no':False})
df
```

Out[53]:

	name	score	attempts	qualify
a	Anastasia	12.5	1	True
b	Dima	9.0	3	False
c	Katherine	16.5	2	True
d	James	NaN	3	False
e	Emily	9.0	2	False
f	Michael	20.0	3	True
g	Matthew	14.5	1	True
h	Laura	NaN	1	False
i	Kevin	8.0	2	False
j	Jonas	19.0	1	True

Write a Pandas program to change the name 'James' to 'Suresh' in name column of the DataFrame. Go to the editor Sample Python dictionary data and list labels: exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'], 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']} labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'] Expected Output: Change the name 'James' to \?Suresh\?: attempts name qualify score a 1 Anastasia yes 12.5 b 3 Dima no 9.0 ..... i 2 Kevin no 8.0 j 1 Jonas yes 19.0

In [60]:
```
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
df['name']=df['name'].replace('James','suresh')
df
```

Out[60]:

	name	score	attempts	qualify
a	Anastasia	12.5	1	yes
b	Dima	9.0	3	no
c	Katherine	16.5	2	yes
d	suresh	NaN	3	no
e	Emily	9.0	2	no
f	Michael	20.0	3	yes
g	Matthew	14.5	1	yes
h	Laura	NaN	1	no
i	Kevin	8.0	2	no
j	Jonas	19.0	1	yes

Write a Pandas program to delete the 'attempts' column from the DataFrame. Go to the editor Sample Python dictionary data and list labels: exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'], 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2,

3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']} labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'] Expected Output: Delete the 'attempts' column from the data frame: name qualify score a Anastasia yes 12.5 b Dima no 9.0 ..... i Kevin no 8.0 j Jonas yes 19.0

In [62]:
```
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
df.drop('attempts',axis=1)
```

Out[62]:

	name	score	qualify
a	Anastasia	12.5	yes
b	Dima	9.0	no
c	Katherine	16.5	yes
d	James	NaN	no
e	Emily	9.0	no
f	Michael	20.0	yes
g	Matthew	14.5	yes
h	Laura	NaN	no
i	Kevin	8.0	no
j	Jonas	19.0	yes

Write a Pandas program to insert a new column in existing DataFrame. Go to the editor Sample Python dictionary data and list labels: exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'], 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']} labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'] Expected Output: New DataFrame after inserting the 'color' column attempts name qualify score color a 1 Anastasia yes 12.5 Red b 3 Dima no 9.0 Blue ..... i 2 Kevin no 8.0 Green j 1 Jonas yes 19.0 Red

In [64]:
```
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
color=['Red','Blue','Orange','Red','White','White','Blue','Green','Green','Red']
df = pd.DataFrame(exam_data , index=labels)
df['color']=color
df.head()
```

Out[64]:

	name	score	attempts	qualify	color
a	Anastasia	12.5	1	yes	Red
b	Dima	9.0	3	no	Blue
c	Katherine	16.5	2	yes	Orange
d	James	NaN	3	no	Red
e	Emily	9.0	2	no	White

Write a Pandas program to iterate over rows in a DataFrame. Go to the editor Sample Python dictionary data and list labels: exam_data = [{'name':'Anastasia', 'score':12.5}, {'name':'Dima','score':9}, {'name':'Katherine','score':16.5}] Expected Output: Anastasia 12.5 Dima 9.0 Katherine 16.5

In [66]:
```
exam_data = [{'name':'Anastasia', 'score':12.5}, {'name':'Dima','score':9}, {'name':'K
df = pd.DataFrame(exam_data)
for i,r in df.iterrows():
    print(r['name'],r['score'])
```

```
Anastasia 12.5
Dima 9.0
Katherine 16.5
```

Write a Pandas program to get list from DataFrame column headers. Go to the editor Sample Python dictionary data and list labels: exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'], 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']} labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'] Expected Output: ['attempts', 'name', 'qualify', 'score']

In [71]:
```
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
l=list()
l=df.columns.values
l
```

Out[71]:
```
array(['name', 'score', 'attempts', 'qualify'], dtype=object)
```

Write a Pandas program to rename columns of a given DataFrame Go to the editor Sample data: Original DataFrame col1 col2 col3 0 1 4 7 1 2 5 8 2 3 6 9 New DataFrame after renaming columns: Column1 Column2 Column3 0 1 4 7 1 2 5 8 2 3 6 9

In [75]:
```
d = {'col1': [1, 2, 3], 'col2': [4, 5, 6], 'col3': [7, 8, 9]}
df = pd.DataFrame(data=d)
df=df.rename(columns={'col1':'Column1','col2':'Column2','col3':'Column3'})
df.head()
```

Out[75]:

	Column1	Column2	Column3
0	1	4	7
1	2	5	8
2	3	6	9

Write a Pandas program to select rows from a given DataFrame based on values in some columns. Go to the editor Sample data: Original DataFrame col1 col2 col3 0 1 4 7 1 4 5 8 2 3 6 9 3 4 7 0 4 5 8 1 Rows for colum1 value == 4 col1 col2 col3 1 4 5 8 3 4 7 0

In [3]:
```
d = {'col1': [1, 4, 3, 4, 5], 'col2': [4, 5, 6, 7, 8], 'col3': [7, 8, 9, 0, 1]}
df = pd.DataFrame(data=d)

df[df['col1']==4]
```

Out[3]:

	col1	col2	col3
1	4	5	8
3	4	7	0

Write a Pandas program to change the order of a DataFrame columns. Go to the editor Sample data: Original DataFrame col1 col2 col3 0 1 4 7 1 4 5 8 2 3 6 9 3 4 7 0 4 5 8 1 After altering col1 and col3 col3 col2 col1 0 7 4 1 1 8 5 4 2 9 6 3 3 0 7 4 4 1 8 5

In [6]:
```
d = {'col1': [1, 4, 3, 4, 5], 'col2': [4, 5, 6, 7, 8], 'col3': [7, 8, 9, 0, 1]}
df = pd.DataFrame(data=d)

df[['col3','col2','col1']]
```

Out[6]:

	col3	col2	col1
0	7	4	1
1	8	5	4
2	9	6	3
3	0	7	4
4	1	8	5

Write a Pandas program to add one row in an existing DataFrame. Go to the editor Sample data: Original DataFrame col1 col2 col3 0 1 4 7 1 4 5 8 2 3 6 9 3 4 7 0 4 5 8 1 After add one row: col1 col2 col3 0 1 4 7 1 4 5 8 2 3 6 9 3 4 7 0 4 5 8 1 5 10 11 12

In [8]:
```
d = {'col1': [1, 4, 3, 4, 5], 'col2': [4, 5, 6, 7, 8], 'col3': [7, 8, 9, 0, 1]}
df = pd.DataFrame(data=d)
df.loc['5']=[10,11,12]
df
```

Out[8]:

	col1	col2	col3
0	1	4	7
1	4	5	8
2	3	6	9
3	4	7	0
4	5	8	1
5	10	11	12

Write a Pandas program to write a DataFrame to CSV file using tab separator. Go to the editor Sample data: Original DataFrame col1 col2 col3 0 1 4 7 1 4 5 8 2 3 6 9 3 4 7 0 4 5 8 1 Data from new_file.csv file: col1\tcol2\tcol3 0 1\t4\t7 1 4\t5\t8 2 3\t6\t9 3 4\t7\t0 4 5\t8\t1

In [9]:
```
d = {'col1': [1, 4, 3, 4, 5], 'col2': [4, 5, 6, 7, 8], 'col3': [7, 8, 9, 0, 1]}
df = pd.DataFrame(data=d)
df.to_csv('new_file.csv',sep='\t',index=False)

newdf=pd.read_csv('new_file.csv')
newdf
```

Out[9]:

	col1\tcol2\tcol3
0	1\t4\t7
1	4\t5\t8
2	3\t6\t9
3	4\t7\t0
4	5\t8\t1

Write a Pandas program to count city wise number of people from a given of data set (city, name of the person). Go to the editor Sample data: city Number of people 0 California 4 1 Georgia 2 2 Los Angeles 4

In [11]:
```
df1 = pd.DataFrame({'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Mich
'city': ['California', 'Los Angeles', 'California', 'California', 'California', 'Los A

df1['city'].value_counts()
```

Out[11]:
```
California     4
Los Angeles    4
Georgia        2
Name: city, dtype: int64
```

Write a Pandas program to delete DataFrame row(s) based on given column value. Go to the editor Sample data: Original DataFrame col1 col2 col3 0 1 4 7 1 4 5 8 2 3 6 9 3 4 7 0 4 5 8 1 New DataFrame col1 col2 col3 0 1 4 7 2 3 6 9 3 4 7 0 4 5 8 1

In [16]:
```
d = {'col1': [1, 4, 3, 4, 5], 'col2': [4, 5, 6, 7, 8], 'col3': [7, 8, 9, 0, 1]}
df = pd.DataFrame(data=d)
print("Original Dataframe=\n",df)
df=df[df['col2']!=5]
df
```

```
Original Dataframe=
    col1  col2  col3
0     1     4     7
1     4     5     8
2     3     6     9
3     4     7     0
4     5     8     1
```

Out[16]:

	col1	col2	col3
0	1	4	7
2	3	6	9
3	4	7	0
4	5	8	1

Write a Pandas program to select a row of series/dataframe by given integer index. Go to the editor Sample data: Original DataFrame col1 col2 col3 0 1 4 7 1 4 5 8 2 3 6 9 3 4 7 0 4 5 8 1 Index-2: Details col1 col2 col3 2 3 6 9

In [18]:
```
d = {'col1': [1, 4, 3, 4, 5], 'col2': [4, 5, 6, 7, 8], 'col3': [7, 8, 9, 0, 1]}
df = pd.DataFrame(data=d)
print("Original DF=\n",df)
df[df.index==2]
```

```
Original DF=
    col1  col2  col3
0     1     4     7
1     4     5     8
2     3     6     9
3     4     7     0
4     5     8     1
```

Out[18]:

	col1	col2	col3
2	3	6	9

Write a Pandas program to replace all the NaN values with Zero's in a column of a dataframe. Go to the editor Sample data: Original DataFrame attempts name qualify score 0 1 Anastasia yes 12.5 1 3 Dima no 9.0 2 2 Katherine yes 16.5 3 3 James no NaN 4 2 Emily no 9.0 5 3 Michael yes 20.0 6 1 Matthew yes 14.5 7 1 Laura no NaN 8 2 Kevin no 8.0 9 1 Jonas yes 19.0 New DataFrame replacing all NaN with 0: attempts name qualify score 0 1 Anastasia yes 12.5 1 3 Dima no 9.0 2 2 Katherine yes 16.5 3 3 James no 0.0 4 2 Emily no 9.0 5 3 Michael yes 20.0 6 1 Matthew yes 14.5 7 1 Laura no 0.0 8 2 Kevin no 8.0 9 1 Jonas yes 19.0

In [22]:
```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
df = pd.DataFrame(exam_data)
print("Original DataFrame")
print(df)
df=df.fillna(0)
df
```

```
Original DataFrame
        name  score  attempts qualify
0  Anastasia   12.5         1     yes
1       Dima    9.0         3      no
2  Katherine   16.5         2     yes
3      James    NaN         3      no
4      Emily    9.0         2      no
5    Michael   20.0         3     yes
6    Matthew   14.5         1     yes
7      Laura    NaN         1      no
8      Kevin    8.0         2      no
9      Jonas   19.0         1     yes
```

Out[22]:

	name	score	attempts	qualify
0	Anastasia	12.5	1	yes
1	Dima	9.0	3	no
2	Katherine	16.5	2	yes
3	James	0.0	3	no
4	Emily	9.0	2	no
5	Michael	20.0	3	yes
6	Matthew	14.5	1	yes
7	Laura	0.0	1	no
8	Kevin	8.0	2	no
9	Jonas	19.0	1	yes

In [ ]:

```
Write a Pandas program to convert index in a column of the given dataframe. Go to the
Sample data:
Original DataFrame
attempts name qualify score
0 1 Anastasia yes 12.5
1 3 Dima no 9.0
2 2 Katherine yes 16.5
3 3 James no NaN
4 2 Emily no 9.0
5 3 Michael yes 20.0
6 1 Matthew yes 14.5
7 1 Laura no NaN
8 2 Kevin no 8.0
9 1 Jonas yes 19.0
After converting index in a column:
index attempts name qualify score
0 0 1 Anastasia yes 12.5
1 1 3 Dima no 9.0
2 2 2 Katherine yes 16.5
3 3 3 James no NaN
4 4 2 Emily no 9.0
5 5 3 Michael yes 20.0
6 6 1 Matthew yes 14.5
7 7 1 Laura no NaN
8 8 2 Kevin no 8.0
9 9 1 Jonas yes 19.0
Hiding index:
index attempts name qualify score
0 1 Anastasia yes 12.5
1 3 Dima no 9.0
2 2 Katherine yes 16.5
3 3 James no NaN
4 2 Emily no 9.0
5 3 Michael yes 20.0
6 1 Matthew yes 14.5
7 1 Laura no NaN
8 2 Kevin no 8.0
9 1 Jonas yes 19.0
```

In [34]:
```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', '
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
df = pd.DataFrame(exam_data)
print("Original DataFrame")
print(df)
#df['index']=df.index
df.insert(loc=0,column='index',value=df.index)
print('After adding inex as a column')
print(df)
df2=pd.DataFrame(df)
print("Without index DataFrame")
print(df2.to_string(index=False))
```

```
Original DataFrame
        name  score  attempts qualify
0  Anastasia   12.5         1     yes
1       Dima    9.0         3      no
2  Katherine   16.5         2     yes
3      James    NaN         3      no
4      Emily    9.0         2      no
5    Michael   20.0         3     yes
6    Matthew   14.5         1     yes
7      Laura    NaN         1      no
8      Kevin    8.0         2      no
9      Jonas   19.0         1     yes
After adding inex as a column
   index       name  score  attempts qualify
0      0  Anastasia   12.5         1     yes
1      1       Dima    9.0         3      no
2      2  Katherine   16.5         2     yes
3      3      James    NaN         3      no
4      4      Emily    9.0         2      no
5      5    Michael   20.0         3     yes
6      6    Matthew   14.5         1     yes
7      7      Laura    NaN         1      no
8      8      Kevin    8.0         2      no
9      9      Jonas   19.0         1     yes
Without index DataFrame
 index      name  score  attempts qualify
     0 Anastasia   12.5         1     yes
     1      Dima    9.0         3      no
     2 Katherine   16.5         2     yes
     3     James    NaN         3      no
     4     Emily    9.0         2      no
     5   Michael   20.0         3     yes
     6   Matthew   14.5         1     yes
     7     Laura    NaN         1      no
     8     Kevin    8.0         2      no
     9     Jonas   19.0         1     yes
```

bWrite a Pandas program to set a given value for particular cell in DataFrame using index value. Go to the editor Sample data: Original DataFrame attempts name qualify score 0 1 Anastasia yes 12.5 1 3 Dima no 9.0 2 2 Katherine yes 16.5 3 3 James no NaN 4 2 Emily no 9.0 5 3 Michael yes 20.0 6 1 Matthew yes 14.5 7 1 Laura no NaN 8 2 Kevin no 8.0 9 1 Jonas yes 19.0 Set a given value for particular cell in the DataFrame attempts name qualify score 0 1 Anastasia yes 12.5 1 3 Dima no 9.0 2 2 Katherine yes 16.5 3 3 James no NaN 4 2 Emily no 9.0 5 3 Michael yes 20.0 6 1 Matthew yes 14.5 7 1 Laura no NaN 8 2 Kevin no 10.2 9 1 Jonas yes 19.0

In [38]:
```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
df = pd.DataFrame(exam_data)
print("Original DataFrame")
print(df)
df.at[8,'score']=10.2
print("Set a given value for particular cell in the DataFrame")
print(df)
```

```
Original DataFrame
        name  score  attempts qualify
0  Anastasia   12.5         1     yes
1       Dima    9.0         3      no
2  Katherine   16.5         2     yes
3      James    NaN         3      no
4      Emily    9.0         2      no
5    Michael   20.0         3     yes
6    Matthew   14.5         1     yes
7      Laura    NaN         1      no
8      Kevin    8.0         2      no
9      Jonas   19.0         1     yes
Set a given value for particular cell in the DataFrame
        name  score  attempts qualify
0  Anastasia   12.5         1     yes
1       Dima    9.0         3      no
2  Katherine   16.5         2     yes
3      James    NaN         3      no
4      Emily    9.0         2      no
5    Michael   20.0         3     yes
6    Matthew   14.5         1     yes
7      Laura    NaN         1      no
8      Kevin   10.2         2      no
9      Jonas   19.0         1     yes
```

Write a Pandas program to count the NaN values in one or more columns in DataFrame. Go to the editor Sample data: Original DataFrame attempts name qualify score 0 1 Anastasia yes 12.5 1 3 Dima no 9.0 2 2 Katherine yes 16.5 3 3 James no NaN 4 2 Emily no 9.0 5 3 Michael yes 20.0 6 1 Matthew yes 14.5 7 1 Laura no NaN 8 2 Kevin no 8.0 9 1 Jonas yes 19.0 Number of NaN values in one or more columns: 2

In [42]:
```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
df = pd.DataFrame(exam_data)
print("Original DataFrame")
print(df)

df.isna().values.sum()
```

```
Original DataFrame
        name  score  attempts qualify
0  Anastasia   12.5         1     yes
1       Dima    9.0         3      no
2  Katherine   16.5         2     yes
3      James    NaN         3      no
4      Emily    9.0         2      no
5    Michael   20.0         3     yes
6    Matthew   14.5         1     yes
7      Laura    NaN         1      no
8      Kevin    8.0         2      no
9      Jonas   19.0         1     yes
2
```

Out[42]:

Write a Pandas program to drop a list of rows from a specified DataFrame. Go to the editor Sample data: Original DataFrame col1 col2 col3 0 1 4 7 1 4 5 8 2 3 6 9 3 4 7 0 4 5 8 1 New DataFrame after removing 2nd & 4th rows: col1 col2 col3 0 1 4 7 1 4 5 8 3 4 7 0

In [44]:
```
d = {'col1': [1, 4, 3, 4, 5], 'col2': [4, 5, 6, 7, 8], 'col3': [7, 8, 9, 0, 1]}
df = pd.DataFrame(d)
print("Original DataFrame")
print(df)
print("New DataFrame after removing 2nd & 4th rows:")
df.drop(index=[2,4],inplace=True)
print(df)
```

```
Original DataFrame
   col1  col2  col3
0     1     4     7
1     4     5     8
2     3     6     9
3     4     7     0
4     5     8     1
New DataFrame after removing 2nd & 4th rows:
   col1  col2  col3
0     1     4     7
1     4     5     8
3     4     7     0
```

Write a Pandas program to reset index in a given DataFrame. Go to the editor Sample data: Original DataFrame attempts name qualify score 0 1 Anastasia yes 12.5 1 3 Dima no 9.0 2 2 Katherine yes 16.5 3 3 James no NaN 4 2 Emily no 9.0 5 3 Michael yes 20.0 6 1 Matthew yes 14.5 7 1 Laura no NaN 8 2 Kevin no 8.0 9 1 Jonas yes 19.0 After removing first and second rows attempts name qualify score 2 2 Katherine yes 16.5 3 3 James no NaN 4 2 Emily no 9.0 5 3 Michael yes 20.0 6 1 Matthew yes 14.5 7 1 Laura no NaN 8 2 Kevin no 8.0 9 1 Jonas yes 19.0 Reset the Index: index attempts name qualify score 0 2 2 Katherine yes 16.5 1 3 3 James no NaN 2 4 2 Emily no 9.0 3 5 3 Michael yes 20.0 4 6 1 Matthew yes 14.5 5 7 1 Laura no NaN 6 8 2 Kevin no 8.0 7 9 1 Jonas yes 19.0

In [48]:
```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', '
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
df = pd.DataFrame(exam_data)
print("Original DataFrame")
print(df)
print("\nAfter removing first and second rows")
df.drop(index=[0,1],inplace=True)
print(df)
print("\nAfter resetting indes")
df.reset_index()
```

```
Original DataFrame
        name  score  attempts qualify
0  Anastasia   12.5         1     yes
1       Dima    9.0         3      no
2  Katherine   16.5         2     yes
3      James    NaN         3      no
4      Emily    9.0         2      no
5    Michael   20.0         3     yes
6    Matthew   14.5         1     yes
7      Laura    NaN         1      no
8      Kevin    8.0         2      no
9      Jonas   19.0         1     yes

After removing first and second rows
        name  score  attempts qualify
2  Katherine   16.5         2     yes
3      James    NaN         3      no
4      Emily    9.0         2      no
5    Michael   20.0         3     yes
6    Matthew   14.5         1     yes
7      Laura    NaN         1      no
8      Kevin    8.0         2      no
9      Jonas   19.0         1     yes

After resetting indes
```

Out[48]:

	index	name	score	attempts	qualify
0	2	Katherine	16.5	2	yes
1	3	James	NaN	3	no
2	4	Emily	9.0	2	no
3	5	Michael	20.0	3	yes
4	6	Matthew	14.5	1	yes
5	7	Laura	NaN	1	no
6	8	Kevin	8.0	2	no
7	9	Jonas	19.0	1	yes

In [ ]:

```
Write a Pandas program to combining two series into a DataFrame. Go to the editor
Sample data:
Data Series:
0 100
1 200
2 python
3 300.12
4 400
dtype: object
0 10
1 20
2 php
3 30.12
4 40
dtype: object
New DataFrame combining two series:
0 1
0 100 10
```

```
1 200 20
2 python php
3 300.12 30.12
4 400 40
```

In [51]:
```
s1 = pd.Series(['100', '200', 'python', '300.12', '400'])
s2 = pd.Series(['10', '20', 'php', '30.12', '40'])
print("Data Series:")
print(s1)
print(s2)

df=pd.concat([s1,s2],axis=1)
print('Dataframe')
print(df)
```

```
Data Series:
0       100
1       200
2    python
3    300.12
4       400
dtype: object
0       10
1       20
2      php
3    30.12
4       40
dtype: object
Dataframe
        0      1
0     100     10
1     200     20
2  python    php
3  300.12  30.12
4     400     40
```

##Write a Pandas program to shuffle a given DataFrame rows. Go to the editor Sample data: Original DataFrame: attempts name qualify score 0 1 Anastasia yes 12.5 1 3 Dima no 9.0 2 2 Katherine yes 16.5 3 3 James no NaN 4 2 Emily no 9.0 5 3 Michael yes 20.0 6 1 Matthew yes 14.5 7 1 Laura no NaN 8 2 Kevin no 8.0 9 1 Jonas yes 19.0 New DataFrame: attempts name qualify score 5 3 Michael yes 20.0 0 1 Anastasia yes 12.5 9 1 Jonas yes 19.0 6 1 Matthew yes 14.5 7 1 Laura no NaN 1 3 Dima no 9.0 3 3 James no NaN 4 2 Emily no 9.0 8 2 Kevin no 8.0 2 2 Katherine yes 16.5

In [53]:
```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', '
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
df = pd.DataFrame(exam_data)
print("Original DataFrame:")
print(df)
df.sample(frac=1)
```

```
Original DataFrame:
        name  score  attempts qualify
0  Anastasia   12.5         1     yes
1       Dima    9.0         3      no
2  Katherine   16.5         2     yes
3      James    NaN         3      no
4      Emily    9.0         2      no
5    Michael   20.0         3     yes
6    Matthew   14.5         1     yes
7      Laura    NaN         1      no
8      Kevin    8.0         2      no
9      Jonas   19.0         1     yes
```

Out[53]:

	name	score	attempts	qualify
7	Laura	NaN	1	no
2	Katherine	16.5	2	yes
9	Jonas	19.0	1	yes
5	Michael	20.0	3	yes
6	Matthew	14.5	1	yes
4	Emily	9.0	2	no
8	Kevin	8.0	2	no
0	Anastasia	12.5	1	yes
3	James	NaN	3	no
1	Dima	9.0	3	no

In [ ]:
```
Write a Pandas program to convert DataFrame column type from string to datetime.
Go to the editor
Sample data:
String Date:
0 3/11/2000
1 3/12/2000
2 3/13/2000
dtype: object
Original DataFrame (string to datetime):
0
0 2000-03-11
1 2000-03-12
2 2000-03-13
```

In [56]:
```
s = pd.Series(['3/11/2000', '3/12/2000', '3/13/2000'])
print("String Date:")
print(s)
s=pd.to_datetime(s)
df=pd.DataFrame(s)
df
```

```
String Date:
0    3/11/2000
1    3/12/2000
2    3/13/2000
dtype: object
```

Out[56]:

	0
0	2000-03-11
1	2000-03-12
2	2000-03-13

Write a Pandas program to find the row for where the value of a given column is maximum. Go to the editor Sample Output: Original DataFrame col1 col2 col3 0 1 4 7 1 2 5 8 2 3 6 12 3 4 9 1 4 7 5 11 Row where col1 has maximum value: 4 Row where col2 has maximum value: 3 Row where col3 has maximum value: 2

In [77]:
```
d = {'col1': [1, 2, 3, 4, 7], 'col2': [4, 5, 6, 9, 5], 'col3': [7, 8, 12, 1, 11]}
df = pd.DataFrame(data=d)
print("Original DataFrame")
print(df)

for i,r in df.iteritems():
    print(f"Max elment {r.max()} in {i} at row {df[i].argmax()}")
```

```
Original DataFrame
   col1  col2  col3
0     1     4     7
1     2     5     8
2     3     6    12
3     4     9     1
4     7     5    11
Max elment 7 in col1 at row 4
Max elment 9 in col2 at row 3
Max elment 12 in col3 at row 2
```

Write a Pandas program to check whether a given column is present in a DataFrame or not. Go to the editor Sample data: Original DataFrame col1 col2 col3 0 1 4 7 1 2 5 8 2 3 6 12 3 4 9 1 4 7 5 11 Col4 is not present in DataFrame. Col1 is present in DataFrame.

In [81]:
```
df = pd.DataFrame(data=d)
print("Original DataFrame")
print(df)

#check col4 and col1 is present or not

if 'col4' in df.columns:
    print('col4 is present in DataFrame')
else:
    print('col4 is not present in DataFrame')

if 'col1' in df.columns:
    print('col1 is present in DataFrame')
else:
    print('col1 is not present in DataFrame')
```

```
Original DataFrame
   col1  col2  col3
0     1     4     7
1     2     5     8
2     3     6    12
3     4     9     1
4     7     5    11
col4 is not present in DataFrame
col1 is present in DataFrame
```

Write a Pandas program to remove infinite values from a given DataFrame. Go to the editor Sample data: Original DataFrame: 0 0 1000.000000 1 2000.000000 2 3000.000000 3 -4000.000000 4 inf 5 -inf Removing infinite values: 0 0 1000.0 1 2000.0 2 3000.0 3 -4000.0 4 NaN 5 NaN

In [83]:
```
df = pd.DataFrame([1000, 2000, 3000, -4000, np.inf, -np.inf])
print("Original DataFrame:")
print(df)
print("Removing infinite values:")
df=df.replace([np.inf, -np.inf],np.nan)
print(df)
```

```
Original DataFrame:
        0
0  1000.0
1  2000.0
2  3000.0
3 -4000.0
4     inf
5    -inf
Removing infinite values:
        0
0  1000.0
1  2000.0
2  3000.0
3 -4000.0
4     NaN
5     NaN
```

Write a Pandas program to insert a given column at a specific column index in a DataFrame. Go to the editor Sample data: Original DataFrame col2 col3 0 4 7 1 5 8 2 6 12 3 9 1 4 5 11 New DataFrame col1 col2 col3 0 1 4 7 1 2 5 8 2 3 6 12 3 4 9 1 4 7 5 11

In [85]:
```
d = {'col2': [4, 5, 6, 9, 5], 'col3': [7, 8, 12, 1, 11]}
df = pd.DataFrame(data=d)
print("Original DataFrame")
print(df)
new_col = [1, 2, 3, 4, 7]
i=0
df.insert(loc=i,column='col1',value=new_col)
print("New DataFrame")
print(df)
```

```
Original DataFrame
   col2  col3
0     4     7
1     5     8
2     6    12
3     9     1
4     5    11
New DataFrame
   col1  col2  col3
0     1     4     7
1     2     5     8
2     3     6    12
3     4     9     1
4     7     5    11
```

In [ ]:
```
Write a Pandas program to convert a given list of lists into a Dataframe. Go to the ed
Sample data:
Original list of lists:
[[2, 4], [1, 3]]
```

```
New DataFrame
col1 col2
0 2 4
1 1 3
```

In [87]:
```
l=[[2,4],[1,3]]
df=pd.DataFrame(l,columns=['col1','col2'])
df
```

Out[87]:

	col1	col2
0	2	4
1	1	3

Write a Pandas program to group by the first column and get second column as lists in rows. Go to the editor Sample data: Original DataFrame col1 col2 0 C1 1 1 C1 2 2 C2 3 3 C2 3 4 C2 4 5 C3 6 6 C2 5 Group on the col1: col1 C1 [1, 2] C2 [3, 3, 4, 5] C3 [6] Name: col2, dtype: object

In [91]:
```
df = pd.DataFrame( {'col1':['C1','C1','C2','C2','C2','C3','C2'], 'col2':[1,2,3,3,4,6,5
print("Original DataFrame")
print(df)
df=df.groupby(by='col1')['col2'].apply(list)
df
```

```
Original DataFrame
  col1  col2
0   C1     1
1   C1     2
2   C2     3
3   C2     3
4   C2     4
5   C3     6
6   C2     5
```

Out[91]:
```
col1
C1          [1, 2]
C2    [3, 3, 4, 5]
C3             [6]
Name: col2, dtype: object
```

Write a Pandas program to get column index from column name of a given DataFrame. Go to the editor Sample Output: Original DataFrame col1 col2 col3 0 1 4 7 1 2 5 8 2 3 6 12 3 4 9 1 4 7 5 11 Index of 'col2' 1

In [94]:
```
d = {'col1': [1, 2, 3, 4, 7], 'col2': [4, 5, 6, 9, 5], 'col3': [7, 8, 12, 1, 11]}
df = pd.DataFrame(data=d)
print("Original DataFrame")
print(df)
print("\nIndex of 'col2'")

df.columns.get_loc('col2')
```

```
Original DataFrame
   col1  col2  col3
0     1     4     7
1     2     5     8
2     3     6    12
3     4     9     1
4     7     5    11

Index of 'col2'
```

Out[94]: 1

Write a Pandas program to select all columns, except one given column in a DataFrame. Go to the editor Sample Output: Original DataFrame col1 col2 col3 0 1 4 7 1 2 5 8 2 3 6 12 3 4 9 1 4 7 5 11 All columns except 'col3': col1 col2 0 1 4 1 2 5 2 3 6 3 4 9 4 7 5

In [95]:
```
d = {'col1': [1, 2, 3, 4, 7], 'col2': [4, 5, 6, 9, 5], 'col3': [7, 8, 12, 1, 11]}
df = pd.DataFrame(data=d)
print("Original DataFrame")
print(df)
print("\nAll columns except 'col3':")
df = df.loc[:, df.columns != 'col3']
print(df)
```

```
Original DataFrame
   col1  col2  col3
0     1     4     7
1     2     5     8
2     3     6    12
3     4     9     1
4     7     5    11

All columns except 'col3':
   col1  col2
0     1     4
1     2     5
2     3     6
3     4     9
4     7     5
```

Write a Pandas program to get first n records of a DataFrame. Go to the editor Sample Output: Original DataFrame col1 col2 col3 0 1 4 7 1 2 5 5 2 3 6 8 3 4 9 12 4 7 5 1 5 11 0 11 First 3 rows of the said DataFrame': col1 col2 col3 0 1 4 7 1 2 5 5 2 3 6 8

In [97]:
```
d = {'col1': [1, 2, 3, 4, 7, 11], 'col2': [4, 5, 6, 9, 5, 0], 'col3': [7, 5, 8, 12, 1,
df = pd.DataFrame(data=d)
print("Original DataFrame")
print(df)
print("\nFirst 3 rows of the said DataFrame':")
print(df.head(3))
print("\nLast 3 rows of the said DataFrame':")
print(df.tail(3))
```

```
Original DataFrame
   col1  col2  col3
0     1     4     7
1     2     5     5
2     3     6     8
3     4     9    12
4     7     5     1
5    11     0    11

First 3 rows of the said DataFrame':
   col1  col2  col3
0     1     4     7
1     2     5     5
2     3     6     8

Last 3 rows of the said DataFrame':
   col1  col2  col3
3     4     9    12
4     7     5     1
5    11     0    11
```

Write a Pandas program to get topmost n records within each group of a DataFrame. Go to the editor Sample Output: Original DataFrame col1 col2 col3 0 1 4 7 1 2 5 5 2 3 6 8 3 4 9 12 4 7 5 1 5 11 0 11 topmost n records within each group of a DataFrame: col1 col2 col3 5 11 0 11 4 7 5 1 3 4 9 12 col1 col2 col3 3 4 9 12 2 3 6 8 1 2 5 5 4 7 5 1 col1 col2 col3 3 4 9 12 5 11 0 11 2 3 6 8

In [98]:
```
d = {'col1': [1, 2, 3, 4, 7, 11], 'col2': [4, 5, 6, 9, 5, 0], 'col3': [7, 5, 8, 12, 1,
df = pd.DataFrame(data=d)
print("Original DataFrame")
print(df)
print("\ntopmost n records within each group of a DataFrame:")
df1 = df.nlargest(3, 'col1')
print(df1)
df2 = df.nlargest(3, 'col2')
print(df2)
df3 = df.nlargest(3, 'col3')
print(df3)
```

```
Original DataFrame
   col1  col2  col3
0     1     4     7
1     2     5     5
2     3     6     8
3     4     9    12
4     7     5     1
5    11     0    11

topmost n records within each group of a DataFrame:
   col1  col2  col3
5    11     0    11
4     7     5     1
3     4     9    12
   col1  col2  col3
3     4     9    12
2     3     6     8
1     2     5     5
   col1  col2  col3
3     4     9    12
5    11     0    11
2     3     6     8
```

Write a Pandas program to remove first n rows and last n rows of a given DataFrame. Go to the editor Sample Output: Original DataFrame col1 col2 col3 0 1 4 7 1 2 5 5 2 3 6 8 3 4 9 12 4 7 5 1 5 11 0 11 After removing first 3 rows of the said DataFrame: col1 col2 col3 3 4 9 12 4 7 5 1 5 11 0 11 After removing last 3 rows of the said DataFrame: col1 col2 col3 0 1 4 7 1 2 5 5 2 3 6 8

In [101…
```
d = {'col1': [1, 2, 3, 4, 7, 11], 'col2': [4, 5, 6, 9, 5, 0], 'col3': [7, 5, 8, 12, 1,
df = pd.DataFrame(data=d)
print("Original DataFrame")
print(df)
print("\nAfter removing first 3 rows of the said DataFrame:")
print(df.iloc[3:])
print("\nAfter removing last 3 rows of the said DataFrame:")
print(df.iloc[:3])
```

```
Original DataFrame
   col1  col2  col3
0     1     4     7
1     2     5     5
2     3     6     8
3     4     9    12
4     7     5     1
5    11     0    11

After removing first 3 rows of the said DataFrame:
   col1  col2  col3
3     4     9    12
4     7     5     1
5    11     0    11

After removing last 3 rows of the said DataFrame:
   col1  col2  col3
0     1     4     7
1     2     5     5
2     3     6     8
```

Write a Pandas program to reverse order (rows, columns) of a given DataFrame. Go to the editor Sample Output: Original DataFrame W X Y Z 0 68 78 84 86 1 75 85 94 97 2 86 96 89 96 3 80 80 83 72 4 66 86 86 83 Reverse column order: Z Y X W 0 86 84 78 68 1 97 94 85 75 2 96 89 96 86 3 72 83 80 80 4 83 86 86 66 Reverse row order: W X Y Z 4 66 86 86 83 3 80 80 83 72 2 86 96 89 96 1 75 85 94 97 0 68 78 84 86 Reverse row order and reset index: W X Y Z 0 66 86 86 83 1 80 80 83 72 2 86 96 89 96 3 75 85 94 97 4 68 78 84 86

In [102…
```
df = pd.DataFrame({'W':[68,75,86,80,66],'X':[78,85,96,80,86], 'Y':[84,94,89,83,86],'Z'
print("Original DataFrame")
print(df)
print("\nReverse column order:")
print(df.loc[:, ::-1])
print("\nReverse row order:")
print(df.loc[::-1])
print("\nReverse row order and reset index:")
print(df.loc[::-1].reset_index(drop = True))
```

```
Original DataFrame
    W   X   Y   Z
0  68  78  84  86
1  75  85  94  97
2  86  96  89  96
3  80  80  83  72
4  66  86  86  83

Reverse column order:
    Z   Y   X   W
0  86  84  78  68
1  97  94  85  75
2  96  89  96  86
3  72  83  80  80
4  83  86  86  66

Reverse row order:
    W   X   Y   Z
4  66  86  86  83
3  80  80  83  72
2  86  96  89  96
1  75  85  94  97
0  68  78  84  86

Reverse row order and reset index:
    W   X   Y   Z
0  66  86  86  83
1  80  80  83  72
2  86  96  89  96
3  75  85  94  97
4  68  78  84  86
```

Write a Pandas program to select columns by data type of a given DataFrame. Go to the editor Sample Output: Original DataFrame name date_of_birth age 0 Alberto Franco 17/05/2002 18.5 1 Gino Mcneill 16/02/1999 21.2 2 Ryan Parkes 25/09/1998 22.5 3 Eesha Hinton 11/05/2002 22.0 4 Syed Wharton 15/09/1997 23.0 Select numerical columns age 0 18.5 1 21.2 2 22.5 3 22.0 4 23.0 Select string columns name date_of_birth 0 Alberto Franco 17/05/2002 1 Gino Mcneill 16/02/1999 2 Ryan Parkes 25/09/1998 3 Eesha Hinton 11/05/2002 4 Syed Wharton 15/09/1997

In [103…

```
df = pd.DataFrame({
    'name': ['Alberto Franco','Gino Mcneill','Ryan Parkes', 'Eesha Hinton', 'Syed Whar
    'date_of_birth': ['17/05/2002','16/02/1999','25/09/1998','11/05/2002','15/09/1997'
    'age': [18.5, 21.2, 22.5, 22, 23]
})

print("Original DataFrame")
print(df)
print("\nSelect numerical columns")
print(df.select_dtypes(include = "number"))
print("\nSelect string columns")
print(df.select_dtypes(include = "object"))
```

```
Original DataFrame
             name date_of_birth   age
0  Alberto Franco    17/05/2002  18.5
1    Gino Mcneill    16/02/1999  21.2
2     Ryan Parkes    25/09/1998  22.5
3    Eesha Hinton    11/05/2002  22.0
4    Syed Wharton    15/09/1997  23.0

Select numerical columns
    age
0  18.5
1  21.2
2  22.5
3  22.0
4  23.0

Select string columns
             name date_of_birth
0  Alberto Franco    17/05/2002
1    Gino Mcneill    16/02/1999
2     Ryan Parkes    25/09/1998
3    Eesha Hinton    11/05/2002
4    Syed Wharton    15/09/1997
```

Write a Pandas program to split a given DataFrame into two random subsets. Go to the editor Sample Output: Original Dataframe and shape: name date_of_birth age 0 Alberto Franco 17/05/2002 18 1 Gino Mcneill 16/02/1999 21 2 Ryan Parkes 25/09/1998 22 3 Eesha Hinton 11/05/2002 22 4 Syed Wharton 15/09/1997 23 (5, 3) Subset-1 and shape: name date_of_birth age 1 Gino Mcneill 16/02/1999 21 4 Syed Wharton 15/09/1997 23 2 Ryan Parkes 25/09/1998 22 (3, 3) Subset-2 and shape: name date_of_birth age 0 Alberto Franco 17/05/2002 18 3 Eesha Hinton 11/05/2002 22 (2, 3)

In [104…

```
df = pd.DataFrame({
    'name': ['Alberto Franco','Gino Mcneill','Ryan Parkes', 'Eesha Hinton', 'Syed Whar
    'date_of_birth': ['17/05/2002','16/02/1999','25/09/1998','11/05/2002','15/09/1997'
    'age': ['18', '21', '22', '22', '23']
})

df_1 = df.sample(frac = 0.6)
df_2 = df.drop(df_1.index)
print("Original Dataframe and shape:")
print(df)
print(df.shape)
print("\nSubset-1 and shape:")
print(df_1)
print(df_1.shape)
print("\nSubset-2 and shape:")
print(df_2)
print(df_2.shape)
```

```
Original Dataframe and shape:
             name date_of_birth age
0  Alberto Franco    17/05/2002  18
1    Gino Mcneill    16/02/1999  21
2     Ryan Parkes    25/09/1998  22
3    Eesha Hinton    11/05/2002  22
4    Syed Wharton    15/09/1997  23
(5, 3)

Subset-1 and shape:
             name date_of_birth age
0  Alberto Franco    17/05/2002  18
4    Syed Wharton    15/09/1997  23
3    Eesha Hinton    11/05/2002  22
(3, 3)

Subset-2 and shape:
           name date_of_birth age
1  Gino Mcneill    16/02/1999  21
2   Ryan Parkes    25/09/1998  22
(2, 3)
```

Write a Pandas program to convert continuous values of a column in a given DataFrame to categorical. Input: { 'Name': ['Alberto Franco','Gino Mcneill','Ryan Parkes', 'Eesha Hinton', 'Syed Wharton'], 'Age': [18, 22, 40, 50, 80, 5] } Output: Age group: 0 kids 1 adult 2 elderly 3 adult 4 elderly 5 kids Name: age_groups, dtype: category Categories (3, object): [kids < adult < elderly] ######################## Use `cut` when you need to segment and sort data values into bins. This function is also useful for going from a continuous variable to a categorical variable. For example, `cut` could convert ages to groups of age ranges. Supports binning into an equal number of bins, or a pre-specified array of bins.

In [105…

```
df = pd.DataFrame({
    'name': ['Alberto Franco','Gino Mcneill','Ryan Parkes', 'Eesha Hinton', 'Syed Whar
      'age': [18, 22, 85, 50, 80, 5]
})
print("Original DataFrame:")
print(df)
print('\nAge group:')
df["age_groups"] = pd.cut(df["age"], bins = [0, 18, 65, 99], labels = ["kids", "adult"
print(df["age_groups"])
```

```
Original DataFrame:
             name  age
0  Alberto Franco   18
1    Gino Mcneill   22
2     Ryan Parkes   85
3    Eesha Hinton   50
4    Syed Wharton   80
5   Kierra Gentry    5

Age group:
0       kids
1      adult
2    elderly
3      adult
4    elderly
5       kids
Name: age_groups, dtype: category
Categories (3, object): ['kids' < 'adult' < 'elderly']
```

Write a Pandas program to combine many given series to create a DataFrame.

In [106…
```
sr1 = pd.Series(['php', 'python', 'java', 'c#', 'c++'])
sr2 = pd.Series([1, 2, 3, 4, 5])
print("Original Series:")
print(sr1)
print(sr2)
print("Combine above series to a dataframe:")
ser_df = pd.DataFrame(sr1, sr2).reset_index()
print(ser_df.head())
print("\nUsing pandas concat:")
ser_df = pd.concat([sr1, sr2], axis = 1)
print(ser_df.head())
print("\nUsing pandas DataFrame with a dictionary, gives a specific name to the columr
ser_df = pd.DataFrame({"col1":sr1, "col2":sr2})
print(ser_df.head(5))
```

```
Original Series:
0       php
1    python
2      java
3        c#
4       c++
dtype: object
0    1
1    2
2    3
3    4
4    5
dtype: int64
Combine above series to a dataframe:
   index       0
0      1  python
1      2    java
2      3      c#
3      4     c++
4      5     NaN

Using pandas concat:
        0  1
0     php  1
1  python  2
2    java  3
3      c#  4
4     c++  5

Using pandas DataFrame with a dictionary, gives a specific name to the columns:
     col1  col2
0     php     1
1  python     2
2    java     3
3      c#     4
4     c++     5
```

Write a Pandas program to create DataFrames that contains random values, contains missing values, contains datetime values and contains mixed values.

In [107…
```
print("DataFrame: Contains random values:")
df1 = pd.util.testing.makeDataFrame() # contains random values
print(df1)
print("\nDataFrame: Contains missing values:")
df2 = pd.util.testing.makeMissingDataframe() # contains missing values
```

```
print(df2)
print("\nDataFrame: Contains datetime values:")
df3 = pd.util.testing.makeTimeDataFrame() # contains datetime values
print(df3)
print("\nDataFrame: Contains mixed values:")
df4 = pd.util.testing.makeMixedDataFrame() # contains mixed values
print(df4)
```

```
DataFrame: Contains random values:
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\util\__init__.py:15: FutureWarning:
pandas.util.testing is deprecated. Use the functions in the public API at pandas.test
ing instead.
  import pandas.util.testing
```

	A	B	C	D
B3FXAYV4fz	0.587605	0.107533	-1.147934	0.474723
xjqME57bLi	2.247462	1.540096	0.363321	0.435162
v030vMVrRk	-0.195550	1.742053	0.154721	-1.053879
NrrItR5stq	-0.546887	0.473228	-1.409329	-0.742640
BSIXSnMkXw	-1.738015	-1.560340	0.242148	0.377922
K9ECbedRPo	-0.841780	-0.348611	0.338261	-0.958874
Lk4FccrIaX	1.172033	0.459320	1.214993	0.806133
OTId56lOcc	-0.942006	-0.437506	-2.694267	-1.553345
Bv5pZOLR9T	-0.854697	2.218631	-0.244218	0.088089
Jb43GUYhPZ	-0.749721	1.027039	0.420651	0.761741
fmALWKoc03	1.444411	0.498855	-1.397814	1.368255
fXjRdwaPPx	-1.843844	-0.314049	1.336622	1.428593
dBOxC6ZnZZ	-1.267593	-0.508868	-0.442341	-1.245129
tL16zzIjAl	0.301845	0.559365	1.474814	0.218138
XLdWarh599	0.148016	1.490441	0.034504	-1.265769
4h4w7cZrYi	1.823574	0.952786	-0.135453	1.149732
IiB76QQ5GD	0.406084	0.792556	-0.286849	0.649881
QrA4fKgZIj	-0.856989	-0.532229	0.307425	1.456354
QNkGAKtzuj	0.738536	0.008734	-0.177423	0.798883
OVFs1zeMDu	0.695087	1.159816	-0.161247	-0.095908
9rkPTmZk4c	0.016454	0.996087	-1.713753	-0.504046
Pz8d57cjmA	0.539636	-0.278082	-0.276771	-0.754891
LlE6bjMOvm	-0.186557	2.011474	0.735133	-1.889602
93ExmHtwDC	0.928557	0.245442	-0.045550	2.688261
Yw8bmN7yle	-1.131798	1.729858	-0.740448	-1.531884
6ZsYmEUqAW	-0.287618	-1.856861	-0.108448	-1.565971
kgjkZT4Oar	-1.258625	-0.693008	0.036435	-0.087004
TNsexysezI	-0.227161	-0.996115	0.142517	-0.603883
0ScfREaWPk	0.806731	0.941814	-0.544253	0.762081
DkT7zc9W72	-1.190645	1.137539	0.766127	-1.210543

DataFrame: Contains missing values:

	A	B	C	D
Dm803YR8bz	NaN	-1.582195	1.461316	-1.613588
A3FcUYtaQO	0.883712	-0.590911	1.103069	0.485296
1rUygfs62i	-0.704537	-1.209787	-0.882303	-0.045147
PyGYx8PtFo	1.405353	-0.149982	-0.737613	0.864703
NhbsDEaT2o	-1.171245	0.417585	2.278700	-1.709328
hIFHkmsjaD	-0.494813	1.290701	0.332845	0.026735
5snsIdKsEi	-1.146490	-0.171380	0.589835	-0.457561
qbcxqhTQyB	-1.294119	1.408289	1.952877	0.433759
LDeofEY0Iu	-1.516586	0.562437	1.180226	1.688796
YurDlxb1K8	1.405688	0.441443	0.722811	-0.295847
gW0atBCu7X	0.293313	-1.220962	-0.339880	-0.316958
8NVylasDLn	-0.655848	0.976145	-1.496459	-0.585115
RHOLlbgKK7	-0.824430	NaN	-1.301185	0.262254
f1qRSS2QqW	-0.316913	0.405711	2.809891	NaN
ojocBvR8vx	0.371853	0.671500	0.250191	0.473218
bfGaj8eLcJ	-0.466541	-0.265559	-0.047602	0.654575
xTjr6SUwci	1.517273	-0.265273	0.299366	-0.783705
IQfE1pKF91	-0.606904	1.383847	-0.069066	NaN
yDUK8OTXJR	1.104287	1.616396	0.878289	-0.194804
Yu4fljm9Kw	1.389784	-0.615099	-0.100763	-0.128802
DbXoUyHfwD	0.038203	0.579051	0.070654	-2.588547
M4xXvX92qk	-0.011414	-1.786680	0.772243	-1.429149
WFYKHlpQKG	1.826997	1.742795	-0.202795	-0.129528
cqYB4CxPX2	0.337315	1.303763	0.191970	NaN
pPkJDMUrca	0.531459	0.340154	NaN	1.577392
qC2IEJLtC0	0.441601	1.047011	-1.335575	-0.627604

```
oXx0ihERoc  1.972030       NaN  1.671284 -0.741188
IMTNbE1dWf -0.185845 -0.195265  1.267861       NaN
hZm89HYYF1 -1.157487       NaN       NaN -1.361327
J5HmRW8P8t       NaN       NaN -0.871547  0.482397

DataFrame: Contains datetime values:
                   A         B         C         D
2000-01-03  0.591912 -0.420650  1.606941 -0.641610
2000-01-04 -0.276832  2.048313  1.368127 -0.591242
2000-01-05 -1.624009  0.637154  0.598927 -1.077046
2000-01-06  1.350961 -0.803230 -2.434233 -0.091322
2000-01-07 -2.154207 -0.997123 -0.048094 -0.253525
2000-01-10 -1.406254  0.117945  1.425746  0.581087
2000-01-11 -0.927337 -0.166547  0.912324  0.114818
2000-01-12 -0.651277 -1.529438  1.187044  0.455948
2000-01-13 -1.756117 -0.350211  0.221553 -0.470707
2000-01-14  0.756431  0.849796 -1.161769 -0.248169
2000-01-17  0.941636  0.976857  2.064888 -0.468196
2000-01-18  0.399471  1.585234 -0.659053 -1.109401
2000-01-19 -0.499043 -1.643125 -0.406614 -0.194525
2000-01-20 -1.389122 -0.704505  2.208646 -0.809514
2000-01-21  0.513131  0.264326  0.875261  0.309354
2000-01-24 -1.086480  0.757354 -0.702693  0.102711
2000-01-25  0.307293 -0.008438  0.338022 -1.658686
2000-01-26 -0.058707  0.450098  0.568403 -0.482140
2000-01-27 -0.504674 -0.153408 -0.794863 -0.655199
2000-01-28 -2.547005  1.134949  0.550185 -2.533519
2000-01-31  2.080040 -0.257757 -0.413968 -0.512594
2000-02-01  2.034460  2.377856  0.825792 -0.053152
2000-02-02 -0.152000 -0.414619  2.556268  0.438615
2000-02-03  0.449971 -1.695305 -1.338642 -0.187961
2000-02-04 -0.493802  1.513680  1.820798 -0.962606
2000-02-07  0.495558  0.585282  0.780716  0.021513
2000-02-08  0.888564  0.514983 -0.921295  1.301263
2000-02-09 -0.173770 -3.253251  1.177058  1.964002
2000-02-10  0.118242 -1.390452  0.159690 -1.377449
2000-02-11 -2.422569  0.808976  0.336398  0.819701

DataFrame: Contains mixed values:
     A    B     C          D
0  0.0  0.0  foo1 2009-01-01
1  1.0  1.0  foo2 2009-01-02
2  2.0  0.0  foo3 2009-01-05
3  3.0  1.0  foo4 2009-01-06
4  4.0  0.0  foo5 2009-01-07
```

Write a Pandas program to fill missing values in time series data. From Wikipedia , in the mathematical field of numerical analysis, interpolation is a type of estimation, a method of constructing new data points within the range of a discrete set of known data points.

In [108…

```
sdata = {"c1":[120, 130 ,140, 150, np.nan, 170], "c2":[7, np.nan, 10, np.nan, 5.5, 16.
df = pd.DataFrame(sdata)
df.index = pd.util.testing.makeDateIndex()[0:6]
print("Original DataFrame:")
print(df)
print("\nDataFrame after interpolate:")
print(df.interpolate())
```

```
Original DataFrame:
               c1    c2
2000-01-03  120.0   7.0
2000-01-04  130.0   NaN
2000-01-05  140.0  10.0
2000-01-06  150.0   NaN
2000-01-07    NaN   5.5
2000-01-10  170.0  16.5

DataFrame after interpolate:
               c1     c2
2000-01-03  120.0   7.00
2000-01-04  130.0   8.50
2000-01-05  140.0  10.00
2000-01-06  150.0   7.75
2000-01-07  160.0   5.50
2000-01-10  170.0  16.50
```

Write a Pandas program to check for inequality of two given DataFrames.

In [110…

```
df1 = pd.DataFrame({'W':[68,75,86,80,None],'X':[78,85,None,80,86], 'Y':[84,94,89,83,8
df2 = pd.DataFrame({'W':[78,75,86,80,None],'X':[78,85,96,80,76], 'Y':[84,84,89,83,86],
print("Original DataFrames:")
print(df1)
print(df2)
print("\nCheck for inequality of the said dataframes:")
print(df1.ne(df2))
```

```
Original DataFrames:
      W     X   Y   Z
0  68.0  78.0  84  86
1  75.0  85.0  94  97
2  86.0   NaN  89  96
3  80.0  80.0  83  72
4   NaN  86.0  86  83
      W   X   Y   Z
0  78.0  78  84  86
1  75.0  85  84  97
2  86.0  96  89  96
3  80.0  80  83  72
4   NaN  76  86  83

Check for inequality of the said dataframes:
       W      X      Y      Z
0   True  False  False  False
1  False  False   True  False
2  False   True  False  False
3  False  False  False  False
4   True   True  False  False
```