In [30]:

```
#Dataframe creation in pandas: While we create dataframe using below ,columns gets sorted a
#Here Country would be first column in the dataframe
import pandas as pd
import numpy as np
df=pd.DataFrame({"Name":["Messi","Neymer","Oskar","Marcelo","Dybala","Suarez"],
                 "Country":["Arg","Brazil","Brazil","Brazil","Arg","Uruguay"],"Rank":[1,2,2
df.columns
```

Out[30]:

```
Index(['Country', 'Name', 'Rank'], dtype='object')
```

# # Group by in Pandas

In [23]:

```
df
```

Out[23]:

| | Country | Name | Rank |
|---|---|---|---|
| **0** | Arg | Messi | 1 |
| **1** | Brazil | Neymer | 2 |
| **2** | Brazil | Oskar | 2 |
| **3** | Brazil | Marcelo | 2 |
| **4** | Arg | Dybala | 1 |
| **5** | Uruguay | Suarez | 3 |

In [27]:

```
df.groupby("Country")['Rank'].sum()
```

Out[27]:

```
Country
Arg        2
Brazil     6
Uruguay    3
Name: Rank, dtype: int64
```

In [26]:

```
df.groupby("Country")['Rank'].sum().reset_index()
```

Out[26]:

| | Country | Rank |
|---|---|---|
| **0** | Arg | 2 |
| **1** | Brazil | 6 |
| **2** | Uruguay | 3 |

In [25]:

```
#df.groupby(["Country","Name"],as_index = False)['Rank'].sum()
df.groupby("Country",as_index = False)['Rank'].sum()
```

Out[25]:

| | Country | Rank |
|---|---|---|
| **0** | Arg | 2 |
| **1** | Brazil | 6 |
| **2** | Uruguay | 3 |

In [28]:

```
df.groupby(["Country","Name"],as_index = False)['Rank'].sum()
```

Out[28]:

| | Country | Name | Rank |
|---|---|---|---|
| **0** | Arg | Dybala | 1 |
| **1** | Arg | Messi | 1 |
| **2** | Brazil | Marcelo | 2 |
| **3** | Brazil | Neymer | 2 |
| **4** | Brazil | Oskar | 2 |
| **5** | Uruguay | Suarez | 3 |

In [32]:

```
#The below code is supported only in >0.25 version of pandas
df.groupby("Country").agg(
        Sum_Rank=('Rank', 'sum'),
        average_Rank=('Rank', np.mean)
    )
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-32-d3bc70319f18> in <module>()
      1 df.groupby("Country").agg(
      2         Sum_Rank=('Rank', 'sum'),
----> 3         average_Rank=('Rank', np.mean)
      4     )

TypeError: aggregate() missing 1 required positional argument: 'arg'
```

In [37]:

```
df.groupby("Country").agg({
        'Rank':[('S_Rank', 'sum'),
        ('M_Rank', 'mean')]
                }).reset_index()
```

Out[37]:

| | Country | Rank | |
|---|---|---|---|
| | | S_Rank | M_Rank |
| 0 | Arg | 2 | 1 |
| 1 | Brazil | 6 | 2 |
| 2 | Uruguay | 3 | 3 |

# # Rank in pandas

In [38]:

```
import pandas as pd
import numpy as np

#Create a DataFrame
d = {
'Name':['Alisa','Bobby','Cathrine','Alisa','Bobby','Cathrine',
'Alisa','Bobby','Cathrine','Alisa','Bobby','Cathrine'],
'Subject':['Mathematics','Mathematics','Mathematics','Science','Science','Science',
'History','History','History','Economics','Economics','Economics'],
'Score':[62,47,55,74,31,77,85,63,42,62,89,85]}

df = pd.DataFrame(d,columns=['Name','Subject','Score'])
df
```

Out[38]:

| | Name | Subject | Score |
|---|---|---|---|
| 0 | Alisa | Mathematics | 62 |
| 1 | Bobby | Mathematics | 47 |
| 2 | Cathrine | Mathematics | 55 |
| 3 | Alisa | Science | 74 |
| 4 | Bobby | Science | 31 |
| 5 | Cathrine | Science | 77 |
| 6 | Alisa | History | 85 |
| 7 | Bobby | History | 63 |
| 8 | Cathrine | History | 42 |
| 9 | Alisa | Economics | 62 |
| 10 | Bobby | Economics | 89 |
| 11 | Cathrine | Economics | 85 |

In [39]:

```
# Ranking of score ascending order

df['score_ranked']=df['Score'].rank(ascending=1)
df
```

Out[39]:

| | Name | Subject | Score | score_ranked |
|---|---|---|---|---|
| **0** | Alisa | Mathematics | 62 | 5.5 |
| **1** | Bobby | Mathematics | 47 | 3.0 |
| **2** | Cathrine | Mathematics | 55 | 4.0 |
| **3** | Alisa | Science | 74 | 8.0 |
| **4** | Bobby | Science | 31 | 1.0 |
| **5** | Cathrine | Science | 77 | 9.0 |
| **6** | Alisa | History | 85 | 10.5 |
| **7** | Bobby | History | 63 | 7.0 |
| **8** | Cathrine | History | 42 | 2.0 |
| **9** | Alisa | Economics | 62 | 5.5 |
| **10** | Bobby | Economics | 89 | 12.0 |
| **11** | Cathrine | Economics | 85 | 10.5 |

In [40]:

```
# Ranking of score descending order

df['score_ranked']=df['Score'].rank(ascending=0)
df
```

Out[40]:

| | Name | Subject | Score | score_ranked |
|---|---|---|---|---|
| **0** | Alisa | Mathematics | 62 | 7.5 |
| **1** | Bobby | Mathematics | 47 | 10.0 |
| **2** | Cathrine | Mathematics | 55 | 9.0 |
| **3** | Alisa | Science | 74 | 5.0 |
| **4** | Bobby | Science | 31 | 12.0 |
| **5** | Cathrine | Science | 77 | 4.0 |
| **6** | Alisa | History | 85 | 2.5 |
| **7** | Bobby | History | 63 | 6.0 |
| **8** | Cathrine | History | 42 | 11.0 |
| **9** | Alisa | Economics | 62 | 7.5 |
| **10** | Bobby | Economics | 89 | 1.0 |
| **11** | Cathrine | Economics | 85 | 2.5 |

In [41]:

```
# Ranking of score in descending order by minimum value

df['score_ranked']=df['Score'].rank(ascending=0,method='min')
df
```

Out[41]:

| | Name | Subject | Score | score_ranked |
|---|---|---|---|---|
| **0** | Alisa | Mathematics | 62 | 7.0 |
| **1** | Bobby | Mathematics | 47 | 10.0 |
| **2** | Cathrine | Mathematics | 55 | 9.0 |
| **3** | Alisa | Science | 74 | 5.0 |
| **4** | Bobby | Science | 31 | 12.0 |
| **5** | Cathrine | Science | 77 | 4.0 |
| **6** | Alisa | History | 85 | 2.0 |
| **7** | Bobby | History | 63 | 6.0 |
| **8** | Cathrine | History | 42 | 11.0 |
| **9** | Alisa | Economics | 62 | 7.0 |
| **10** | Bobby | Economics | 89 | 1.0 |
| **11** | Cathrine | Economics | 85 | 2.0 |

In [42]:

```
# Ranking of score in descending order by maximum value

df['score_ranked']=df['Score'].rank(ascending=0,method='max')
df
```

Out[42]:

| | Name | Subject | Score | score_ranked |
|---|---|---|---|---|
| **0** | Alisa | Mathematics | 62 | 8.0 |
| **1** | Bobby | Mathematics | 47 | 10.0 |
| **2** | Cathrine | Mathematics | 55 | 9.0 |
| **3** | Alisa | Science | 74 | 5.0 |
| **4** | Bobby | Science | 31 | 12.0 |
| **5** | Cathrine | Science | 77 | 4.0 |
| **6** | Alisa | History | 85 | 3.0 |
| **7** | Bobby | History | 63 | 6.0 |
| **8** | Cathrine | History | 42 | 11.0 |
| **9** | Alisa | Economics | 62 | 8.0 |
| **10** | Bobby | Economics | 89 | 1.0 |
| **11** | Cathrine | Economics | 85 | 3.0 |

In [43]:

```
df['score_ranked']=df['Score'].rank(ascending=0,method='dense')
df
```

Out[43]:

| | Name | Subject | Score | score_ranked |
|---|---|---|---|---|
| **0** | Alisa | Mathematics | 62 | 6.0 |
| **1** | Bobby | Mathematics | 47 | 8.0 |
| **2** | Cathrine | Mathematics | 55 | 7.0 |
| **3** | Alisa | Science | 74 | 4.0 |
| **4** | Bobby | Science | 31 | 10.0 |
| **5** | Cathrine | Science | 77 | 3.0 |
| **6** | Alisa | History | 85 | 2.0 |
| **7** | Bobby | History | 63 | 5.0 |
| **8** | Cathrine | History | 42 | 9.0 |
| **9** | Alisa | Economics | 62 | 6.0 |
| **10** | Bobby | Economics | 89 | 1.0 |
| **11** | Cathrine | Economics | 85 | 2.0 |

In [44]:

```
# Rank by Group

df["group_rank"] = df.groupby("Subject")["Score"].rank(ascending=0,method='dense')
df
```

Out[44]:

| | Name | Subject | Score | score_ranked | group_rank |
|---|---|---|---|---|---|
| **0** | Alisa | Mathematics | 62 | 6.0 | 1.0 |
| **1** | Bobby | Mathematics | 47 | 8.0 | 3.0 |
| **2** | Cathrine | Mathematics | 55 | 7.0 | 2.0 |
| **3** | Alisa | Science | 74 | 4.0 | 2.0 |
| **4** | Bobby | Science | 31 | 10.0 | 3.0 |
| **5** | Cathrine | Science | 77 | 3.0 | 1.0 |
| **6** | Alisa | History | 85 | 2.0 | 1.0 |
| **7** | Bobby | History | 63 | 5.0 | 2.0 |
| **8** | Cathrine | History | 42 | 9.0 | 3.0 |
| **9** | Alisa | Economics | 62 | 6.0 | 3.0 |
| **10** | Bobby | Economics | 89 | 1.0 | 1.0 |
| **11** | Cathrine | Economics | 85 | 2.0 | 2.0 |

# # Pandas concatenation

In [ ]:

```
#pd.concat() function: the most multi-purpose and can be used to
combine multiple DataFrames along either axis.
#DataFrame.append() method: a quick way to add rows to your DataFrame,
but not applicable for adding columns.
#pd.merge() function: great for joining two DataFrames together when
we have one column (key) containing common values.
#DataFrame.join() method: a quicker way to join two DataFrames,
but works only off index labels rather than columns.
```

In [14]:

```
# Python program to concatenate
# dataframes using Panda

# Creating first dataframe
import pandas as pd
df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
'B': ['B0', 'B1', 'B2', 'B3'],
'C': ['C0', 'C1', 'C2', 'C3'],
'D': ['D0', 'D1', 'D2', 'D3']},
index = [0, 1, 2, 3])

# Creating second dataframe
df2 = pd.DataFrame({'A': ['A4', 'A5', 'A6', 'A7'],
'B': ['B4', 'B5', 'B6', 'B7'],
'C': ['C4', 'C5', 'C6', 'C7'],
'D': ['D4', 'D5', 'D6', 'D7']},
index = [4, 5, 6, 7])

# Creating third dataframe
df3 = pd.DataFrame({'A': ['A8', 'A9', 'A10', 'A11'],
'B': ['B8', 'B9', 'B10', 'B11'],
'C': ['C8', 'C9', 'C10', 'C11'],
'D': ['D8', 'D9', 'D10', 'D11']},
index = [8, 9, 10, 11])

# Concatenating the dataframes
display(pd.concat([df1, df2],axis=0))#concating rows
display(pd.concat([df1, df2],axis=1))#concating columns
```

| | A | B | C | D |
|---|---|---|---|---|
| **0** | A0 | B0 | C0 | D0 |
| **1** | A1 | B1 | C1 | D1 |
| **2** | A2 | B2 | C2 | D2 |
| **3** | A3 | B3 | C3 | D3 |
| **4** | A4 | B4 | C4 | D4 |
| **5** | A5 | B5 | C5 | D5 |
| **6** | A6 | B6 | C6 | D6 |
| **7** | A7 | B7 | C7 | D7 |

| | A | B | C | D | A | B | C | D |
|---|---|---|---|---|---|---|---|---|
| **0** | A0 | B0 | C0 | D0 | NaN | NaN | NaN | NaN |
| **1** | A1 | B1 | C1 | D1 | NaN | NaN | NaN | NaN |
| **2** | A2 | B2 | C2 | D2 | NaN | NaN | NaN | NaN |
| **3** | A3 | B3 | C3 | D3 | NaN | NaN | NaN | NaN |
| **4** | NaN | NaN | NaN | NaN | A4 | B4 | C4 | D4 |
| **5** | NaN | NaN | NaN | NaN | A5 | B5 | C5 | D5 |
| **6** | NaN | NaN | NaN | NaN | A6 | B6 | C6 | D6 |
| **7** | NaN | NaN | NaN | NaN | A7 | B7 | C7 | D7 |

In [6]:

```
pd.merge([df1, df2]) #merge always require positional argument
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-f8fb817eca84> in <module>()
----> 1 pd.merge([df1, df2])

TypeError: merge() missing 1 required positional argument: 'right'
```

In [9]:

```
df1.append(df2)
```

Out[9]:

| | A | B | C | D |
|---|---|---|---|---|
| **0** | A0 | B0 | C0 | D0 |
| **1** | A1 | B1 | C1 | D1 |
| **2** | A2 | B2 | C2 | D2 |
| **3** | A3 | B3 | C3 | D3 |
| **4** | A4 | B4 | C4 | D4 |
| **5** | A5 | B5 | C5 | D5 |
| **6** | A6 | B6 | C6 | D6 |
| **7** | A7 | B7 | C7 | D7 |

In [ ]:

```
#Why Concat is faster than append?
#APPEND() method in Pandas doesn't modify the original object.
Instead it creates a new one with combined data
```

In [33]:

```
l = pd.DataFrame([['a', 1], ['b', 2]], list('XY'), list('AB'))
r = pd.DataFrame([['a', 3], ['b', 4]], list('XY'), list('AC'))
print(l)
print(r)
print(pd.merge(l,r, on="A",how="inner"))
#merge does create new indexing,keeps only one column from key  after joining
print(pd.concat([l,r],join="inner",axis=1))
#concat keeps original index,keeps both columns from key  after joining
```

```
   A  B
X  a  1
Y  b  2
   A  C
X  a  3
Y  b  4
   A  B  C
0  a  1  3
1  b  2  4
   A  B  A  C
X  a  1  a  3
Y  b  2  b  4
```

In [ ]:

```

```