In [1]:

```
# Concatenating and Merging Data
# load titanic dataset

import matplotlib.pyplot as plt
import seaborn as sns
titanic_data = sns.load_dataset('titanic')
```

In [2]:

```
# Oftentimes, you need to concatenate or join multiple Pandas
# dataframes horizontally or vertically. So, let's first see how to
# concatenate or join Pandas dataframes vertically or in row
# order.
# Using Titanic data, we will create two Pandas dataframes. The
# first dataframe consists of rows where the passenger class is
# First, while the second dataframe consists of rows where the
# passenger class is Second.

titanic_pclass1_data = titanic_data[titanic_data["class"] == "First"]
print(titanic_pclass1_data.shape)
titanic_pclass2_data = titanic_data[titanic_data["class"] == "Second"]
print(titanic_pclass2_data.shape)

# The output shows that both the newly created dataframes
# have 15 columns. It is important to mention that while
# concatenating data vertically, both the dataframes should
# have an equal number of columns.
```

```
(216, 15)
(184, 15)
```

In [3]:

```
# There are two ways to concatenate datasets horizontally. You
# can call the append() method via the first dataframe and pass
# the second dataframe as a parameter to the append()
# method. Look at the following script:

final_data = titanic_pclass1_data.append(titanic_pclass2_data,
ignore_index=True)
print(final_data.shape)

# The output now shows that the total number of rows is 400,
# which is the sum of the number of rows in the two dataframes
# that we concatenated.
```

```
(400, 15)
```

In [5]:

```
# The other way to concatenate two dataframes is by passing
# both the dataframes as parameters to the concat() method of
# the Pandas module. The following script shows how to do
# that.

import pandas as pd
final_data = pd.concat([titanic_pclass1_data, titanic_pclass2_data])
print(final_data.shape)
```

```
(400, 15)
```

In [6]:

```
# Concatenating Columns
# To concatenate dataframes horizontally, make sure that the
# dataframes have an equal number of rows. You can use the
# concat() method to concatenate dataframes horizontally as
# well. However, you will need to pass 1 as the value for the axis
# attribute. Furthermore, to reset dataset indexes, you need to
# pass True as the value for the ignore_index attribute.

df1 = final_data[:200]
print(df1.shape)
df2 = final_data[200:]
print(df2.shape)
final_data2 = pd.concat([df1, df2], axis = 1, ignore_index = True)
print(final_data2.shape)
```

```
(200, 15)
(200, 15)
(400, 30)
```

In [7]:

```
# Merging Data
# You can merge multiple dataframes based on common values
# between any columns of the two dataframes.
# Let's create two dataframes: scores1 and scores2.

import pandas as pd
scores1 = [
{'Subject':'Mathematics', 'Score':85, 'Grade': 'B', 'Remarks': 'Good',
},
{'Subject':'History', 'Score':98, 'Grade': 'A','Remarks':
'Excellent'},
{'Subject':'English', 'Score':76, 'Grade': 'C','Remarks': 'Fair'},
{'Subject':'Chemistry', 'Score':72, 'Grade': 'C','Remarks': 'Fair'},
]
scores2 = [
{'Subject':'Arts', 'Score':70, 'Grade': 'C','Remarks': 'Fair'},
{'Subject':'Physics', 'Score':75, 'Grade': 'C','Remarks': 'Fair'},
{'Subject':'English', 'Score':92, 'Grade': 'A','Remarks':
'Excellent'},
{'Subject':'Chemistry', 'Score':91, 'Grade': 'A','Remarks':
'Excellent'},
]
scores1_df = pd.DataFrame(scores1)
scores2_df = pd.DataFrame(scores2)
```

In [8]:

```
# The following script prints the header of the scores1
# dataframe.

scores1_df.head()
```

Out[8]:

| | Subject | Score | Grade | Remarks |
|---|---|---|---|---|
| **0** | Mathematics | 85 | B | Good |
| **1** | History | 98 | A | Excellent |
| **2** | English | 76 | C | Fair |
| **3** | Chemistry | 72 | C | Fair |

In [9]:

```
# The following script prints the header of the scores2
# dataframe

scores2_df.head()
```

Out[9]:

| | Subject | Score | Grade | Remarks |
|---|---|---|---|---|
| **0** | Arts | 70 | C | Fair |
| **1** | Physics | 75 | C | Fair |
| **2** | English | 92 | A | Excellent |
| **3** | Chemistry | 91 | A | Excellent |

In [10]:

```
# You can see that the two dataframes, scores1 and scores2,
# have the same columns. You can merge or join two
# dataframes on any column.

# The merge() function in Pandas can be used to merge two
# dataframes based on common values between columns of the
# two dataframes. The merge() is similar to SQL JOIN
# operations.

# The script below merges scores1 and scores2 dataframe using
# the INNER JOIN strategy.

# To merge two dataframes, you call the merge() function on
# the dataframe that you want on the left side of the join
# operation. The dataframe to be merged on the right is passed
# as the first parameter. Next, the column name on which you
# want to apply the merge operation is passed to the "on"
# attribute. Finally, the merge strategy is passed to the "how"
# attribute, as shown in the script below.

# With INNER JOIN, only those records from both the
# dataframes are returned, where there are common values in
# the column used for merging the dataframes.
# For instance, in the scores1 and scores2 dataframes, English
# and Chemistry are two subjects that exist in the Subject
# column of both the dataframes.

# Therefore, in the output of the script below, you can see the
# merged dataframes containing only these two records. Since
# the column names, Score, Grade, and Remarks, are similar in
# both the dataframes, the column names from the dataset on
# the left (which is scores1 in the following script) have been
# appended with "_x". While the column names in the dataframe
# on the right are appended with "_y".

# In case both the dataframes in the merge operation contain
# different column names, you will see the original column
# names in the merged dataframe.
```

In [11]:

```
join_inner_df = scores1_df.merge(scores2_df, on='Subject', how='inner')
join_inner_df.head()
```

Out[11]:

| | Subject | Score_x | Grade_x | Remarks_x | Score_y | Grade_y | Remarks_y |
|---|---|---|---|---|---|---|---|
| **0** | English | 76 | C | Fair | 92 | A | Excellent |
| **1** | Chemistry | 72 | C | Fair | 91 | A | Excellent |

In [12]:

```
# Merging with Left Join
# When you merge two Pandas dataframes using a LEFT join, all
# the records from the dataframe to the left side of the merge()
# function are returned, whereas, for the right dataframe, only
# those records are returned where a common value is found on
# the column being merged.

join_inner_df = scores1_df.merge(scores2_df, on='Subject', how='left')
join_inner_df.head()

# In the output below, you can see the records for the
# Mathematics and History subjects in the dataframe on the left.
# But since these records do not exist in the dataframe on the
# right, you see null values for columns Score_y, Grade_y, and
# Remarks_y. Only those records can be seen for the right
# dataframe where there are common values in the Subject
# column from both the dataframes.
```

Out[12]:

| | Subject | Score_x | Grade_x | Remarks_x | Score_y | Grade_y | Remarks_y |
|---|---|---|---|---|---|---|---|
| **0** | Mathematics | 85 | B | Good | NaN | NaN | NaN |
| **1** | History | 98 | A | Excellent | NaN | NaN | NaN |
| **2** | English | 76 | C | Fair | 92.0 | A | Excellent |
| **3** | Chemistry | 72 | C | Fair | 91.0 | A | Excellent |

In [13]:

```
# Merging with Right Join
# In merging with right join, the output contains all the records
# from the right dataframe while only those records are
# returned from the left dataframe where there are common
# values in the merge column.
# Here is an example:

join_inner_df = scores1_df.merge(scores2_df, on='Subject', how='right')
join_inner_df.head()
```

Out[13]:

| | Subject | Score_x | Grade_x | Remarks_x | Score_y | Grade_y | Remarks_y |
|---|---|---|---|---|---|---|---|
| **0** | Arts | NaN | NaN | NaN | 70 | C | Fair |
| **1** | Physics | NaN | NaN | NaN | 75 | C | Fair |
| **2** | English | 76.0 | C | Fair | 92 | A | Excellent |
| **3** | Chemistry | 72.0 | C | Fair | 91 | A | Excellent |

In [14]:

```
# Merging with Outer Join
# With outer join, all the records are returned from both right
# and left dataframes. Here is an example:

join_inner_df = scores1_df.merge(scores2_df, on='Subject', how='outer')
join_inner_df.head()
```

Out[14]:

| | Subject | Score_x | Grade_x | Remarks_x | Score_y | Grade_y | Remarks_y |
|---|---|---|---|---|---|---|---|
| **0** | Mathematics | 85.0 | B | Good | NaN | NaN | NaN |
| **1** | History | 98.0 | A | Excellent | NaN | NaN | NaN |
| **2** | English | 76.0 | C | Fair | 92.0 | A | Excellent |
| **3** | Chemistry | 72.0 | C | Fair | 91.0 | A | Excellent |
| **4** | Arts | NaN | NaN | NaN | 70.0 | C | Fair |