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

In [2]:
```
titanic_src = pd.read_csv('/home/ANA522/Titanic.csv', sep=',')
titanic_src.shape
```

Out[2]:
```
(891, 12)
```

In [3]:
```
titanic_src1 = titanic_src.copy()
titanic_src1.columns
```

Out[3]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [4]:
```
TDS_Pclass_Sex = titanic_src.set_index(['Survived','Sex'])
TDS_Pclass_Sex_Sorted = TDS_Pclass_Sex.sort_index()
TDS_Pclass_Sex_Sorted
```

Out[4]:

		PassengerId	Pclass	Name	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
Survived	**Sex**										
0	**female**	15	3	Vestrom, Miss. Hulda Amanda Adolfina	14.0	0	0	350406	7.8542	NaN	S
	female	19	3	Vander Planke, Mrs. Julius (Emelia Maria Vande...	31.0	1	0	345763	18.0000	NaN	S
	female	25	3	Palsson, Miss. Torborg Danira	8.0	3	1	349909	21.0750	NaN	S
	female	39	3	Vander Planke, Miss. Augusta Maria	18.0	2	0	345764	18.0000	NaN	S
	female	41	3	Ahlin, Mrs. Johan (Johanna Persdotter Larsson)	40.0	1	0	7546	9.4750	NaN	S
...	**...**	...	...	...	...	...	...	...	...	...	...
1	**male**	839	3	Chip, Mr. Chang	32.0	0	0	1601	56.4958	NaN	S
	male	840	1	Marechal, Mr. Pierre	NaN	0	0	11774	29.7000	C47	C
	male	858	1	Daly, Mr. Peter Denis	51.0	0	0	113055	26.5500	E17	S
	male	870	3	Johnson, Master. Harold Theodor	4.0	1	1	347742	11.1333	NaN	S
	male	890	1	Behr, Mr. Karl Howell	26.0	0	0	111369	30.0000	C148	C

891 rows × 10 columns

In [5]:
```
###Q01: Transform the Titanic dataset with hierarchical indexing by the multiple index in the order of Survived,
###and Pclass attributes.

TDS_Pclass_Sex = titanic_src.set_index(['Survived','Sex', 'Pclass'])
TDS_Pclass_Sex_Sorted = TDS_Pclass_Sex.sort_index()
TDS_Pclass_Sex_Sorted
```

Out[5]:

			PassengerId	Name	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
Survived	**Sex**	**Pclass**									
0	**female**	**1**	178	Isham, Miss. Ann Elizabeth	50.0	0	0	PC 17595	28.7125	C49	C
		1	298	Allison, Miss. Helen Loraine	2.0	1	2	113781	151.5500	C22 C26	S
		1	499	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	25.0	1	2	113781	151.5500	C22 C26	S
		2	42	Turpin, Mrs. William John Robert (Dorothy Ann ...	27.0	1	0	11668	21.0000	NaN	S
		2	200	Yrois, Miss. Henriette ("Mrs Harbeck")	24.0	0	0	248747	13.0000	NaN	S
...	**...**	**...**	...	...	...	...	...	...	...	...	...
1	**male**	**3**	805	Hedman, Mr. Oskar Arvid	27.0	0	0	347089	6.9750	NaN	S
		3	822	Lulic, Mr. Nikola	27.0	0	0	315098	8.6625	NaN	S

			PassengerId	Name	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
Survived	**Sex**	**Pclass**									
		3	829	McCormack, Mr. Thomas Joseph	NaN	0	0	367228	7.7500	NaN	Q
		3	839	Chip, Mr. Chang	32.0	0	0	1601	56.4958	NaN	S
		3	870	Johnson, Master. Harold Theodor	4.0	1	1	347742	11.1333	NaN	S

891 rows × 9 columns

In [6]:
```
###Q02: Transform the Titanic dataset with hierarchical indexing by the multiple index in the order of Sex and P
##only PassengerId, Survived, Age, Fare, and Embarked attributes included. Sort by row indices and name the new
##TDS_Sex_Pclass_Sorted to be used in the following questions.

TDS_Pclass_Sex = titanic_src.set_index(['Sex', 'Pclass'])
TDS_Pclass_Sex_Sorted = TDS_Pclass_Sex.sort_index()
TDS_Pclass_Sex_Sorted = TDS_Pclass_Sex_Sorted.loc[:, ('PassengerId', 'Survived', 'Age', 'Fare', 'Embarked')]
TDS_Pclass_Sex_Sorted
```

Out[6]:

		PassengerId	Survived	Age	Fare	Embarked
Sex	**Pclass**					
female	**1**	2	1	38.0	71.2833	C
	1	4	1	35.0	53.1000	S
	1	12	1	58.0	26.5500	S
	1	32	1	NaN	146.5208	C
	1	53	1	49.0	76.7292	C
...	**...**	...	...	...	...	...
male	**3**	878	0	19.0	7.8958	S
	3	879	0	NaN	7.8958	S
	3	882	0	33.0	7.8958	S
	3	885	0	25.0	7.0500	S
	3	891	0	32.0	7.7500	Q

891 rows × 5 columns

In [7]:
```
###Q03: Follow up from Q02. Setup names for index and columns of the multi-index frame for TDS_Sex_Pclass_Sortec
###and “Roomclass” for index names respectively. Use “Profile” for columns’ name.

TDS_Pclass_Sex_Sorted.index.names = ['Gender', 'Roomclass']
TDS_Pclass_Sex_Sorted.columns.names = ['Profile']
TDS_Pclass_Sex_Sorted
```

Out[7]:

	Profile	PassengerId	Survived	Age	Fare	Embarked
Gender	**Roomclass**					
female	**1**	2	1	38.0	71.2833	C
	1	4	1	35.0	53.1000	S
	1	12	1	58.0	26.5500	S
	1	32	1	NaN	146.5208	C
	1	53	1	49.0	76.7292	C
...	**...**	...	...	...	...	...
male	**3**	878	0	19.0	7.8958	S
	3	879	0	NaN	7.8958	S
	3	882	0	33.0	7.8958	S
	3	885	0	25.0	7.0500	S
	3	891	0	32.0	7.7500	Q

891 rows × 5 columns

In [8]:
```
###Q04: Create a Series of data for PassengerId, Survived, Age, Fare, and Embarked for every multi-index combina
#and Roomclass(Pclass) in TDS_Sex_Pclass_Sorted


TDS_Pclass_Sex_Sorted.stack()
```

Out[8]:
```
Gender  Roomclass  Profile
female  1          PassengerId          2
                   Survived             1
                   Age               38.0
                   Fare           71.2833
                   Embarked             C
                                     ...
male    3          PassengerId        891
                   Survived             0
                   Age               32.0
                   Fare              7.75
                   Embarked             Q
Length: 4276, dtype: object
```

In [9]:
```
###Q05: Adjust the hierarchical indexing so that the Roomclass is at the first level followed by Gender from TDS
#frame.swaplevel(0, 1).sort_index(level=0)

TDS_Pclass_Sex_Sorted.swaplevel(0, 1).sort_index(level=0)
```

Out[9]:

	Profile	PassengerId	Survived	Age	Fare	Embarked
Roomclass	**Gender**					
1	**female**	2	1	38.0	71.2833	C
	female	4	1	35.0	53.1000	S
	female	12	1	58.0	26.5500	S
	female	32	1	NaN	146.5208	C
	female	53	1	49.0	76.7292	C
...	**...**	...	...	...	...	...
3	**male**	878	0	19.0	7.8958	S
	male	879	0	NaN	7.8958	S
	male	882	0	33.0	7.8958	S
	male	885	0	25.0	7.0500	S
	male	891	0	32.0	7.7500	Q

891 rows × 5 columns

In [10]:
```
###Q06: Compute total Fare of all passengers by Roomclass using TDS_Sex_Pclass_Sorted

TDS_Pclass_Sex_Sorted.Fare.sum(level='Roomclass', axis=0)
```

Out[10]:
```
Roomclass
1    18177.4125
2     3801.8417
3     6714.6951
Name: Fare, dtype: float64
```

In [11]:
```
###Q07 Compute average Fare of all passengers by female and male using TDS_Sex_Pclass_Sorted

TDS_Pclass_Sex_Sorted.Fare.mean(level='Gender')
```

Out[11]:
```
Gender
female    44.479818
male      25.523893
Name: Fare, dtype: float64
```

In [12]:
```
### Q08 Compute average Fare of all passengers from TDS_Sex_Pclass_Sorted up to 2 decimal points.

val = TDS_Pclass_Sex_Sorted['Fare'].mean().round(2)
print("The average Fare for all passengers:", val)
```

```
The average Fare for all passengers: 32.2
```

In [13]:

```
###Q09 Transform TDS_Sex_Pclass_Sorted so that PassengerId and Age are multiple indices while Gender and Roomcla
###Display the result with Survived values only.

dataframe_reset = TDS_Pclass_Sex_Sorted.reset_index()
dataframe_reset.pivot(['PassengerId','Age'], ['Gender', 'Roomclass'], 'Survived')
```

Out[13]:

	Gender	female			male		
	Roomclass	1	2	3	1	2	3
PassengerId	Age						
1	**22.0**	NaN	NaN	NaN	NaN	NaN	0.0
2	**38.0**	1.0	NaN	NaN	NaN	NaN	NaN
3	**26.0**	NaN	NaN	1.0	NaN	NaN	NaN
4	**35.0**	1.0	NaN	NaN	NaN	NaN	NaN
5	**35.0**	NaN	NaN	NaN	NaN	NaN	0.0
...	**...**	...	...	...	...	...	...
887	**27.0**	NaN	NaN	NaN	NaN	0.0	NaN
888	**19.0**	1.0	NaN	NaN	NaN	NaN	NaN
889	**NaN**	NaN	NaN	0.0	NaN	NaN	NaN
890	**26.0**	NaN	NaN	NaN	1.0	NaN	NaN
891	**32.0**	NaN	NaN	NaN	NaN	NaN	0.0

891 rows × 6 columns

In [ ]:

```
### Q10 Create a dataframe from TDS_Sex_Pclass_Sorted that uses PassengerId as the key index to make all tuples
##and value to be data entries in the table.

melted = pd.melt(TDS_Pclass_Sex_Sorted, ['PassengerId'])
melted
```

Out[ ]:

	PassengerId	Profile	value
0	2	Survived	1
1	4	Survived	1
2	12	Survived	1
3	32	Survived	1
4	53	Survived	1
...	...	...	...
3559	878	Embarked	S
3560	879	Embarked	S
3561	882	Embarked	S
3562	885	Embarked	S
3563	891	Embarked	Q

3564 rows × 3 columns

In [ ]: