### **Qn 1. How do we find the number of rows and columns in a df?**

I. len(df) & len(df.columns)

II. df.shape()

III. len(df.rows) & len(df.columns)

Option I. is correct as len(df) will give us the number of rows in df while len(df.columns) will give us the number fo columns.

Option II. is incorrect because it is df.shape instead of df.shape(). # syntax error

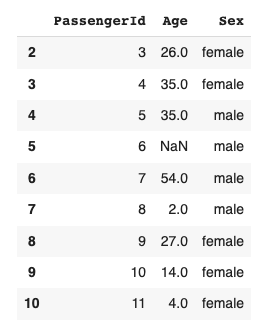
Option III is incorrect because it is df.rows is invalid.

Hence, Option I only.

### **Qn 2A. What would be the result of the df following this line of code?**

| df.loc[2:10, ["PassengerId", "Age", "Sex"]] |
| --- |

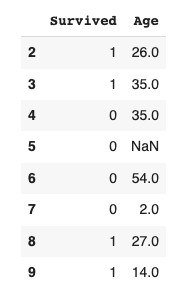
.loc is primarily label based. Hence, pandas will select the range from index 2 to index 10 as labelled in the df (red box). Hence, only option II is correct as shown below.



### **Qn 2B. What would be the result of the df following this line of code?**

| df.iloc[2:10,[1,5]] |
| --- |

.iloc is purely integer-location-based indexing. This means it is selected by the row’s position and not based on the index label. Hence, pandas will be similar to python syntax for this case and select from index 2 to index 9 and exclude index 10 in the following position.



### **Qn 3. What do you think would happen to the df following these lines?**

| 1. df["Age"]=df["Age"].astype(str)  2. df["Age"]=df["Age"].astype(int) |
| --- |

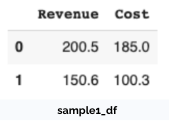
The type in the “Age” column was originally float. Hence, when it converts from float to string, the decimal points are also present in the string (eg. 22.0-> “22.0”). This will lead to value error when we convert from string type to integer type:

invalid literal for int() with base 10: '22.0'

### **Qn 4. Is it possible to apply a function where we reference multiple columns within a row and create a new column? (Eg. create a new column "net profit" using profit and cost)**

Yes, suggested code:

| sample1\_df['profit']=sample1\_df.apply(lambda row: row['Revenue']-row['Cost'],axis=1) |
| --- |

->

### **Qn 5. How do I find all the passengers with name but do not have an age using df\_merge1 and df\_merge2?**

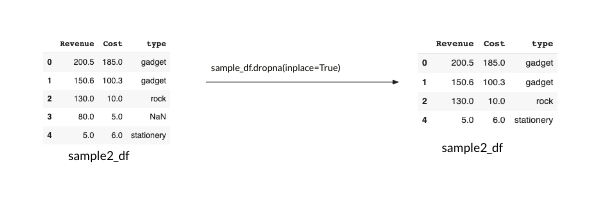
Only Option 2 is correct.

* RIGHT join is wrong because we need passengers with name and sex and RIGHT join will include passenger id 7 where there are no valid name and sex.
* INNER join is wrong because we will exclude passenger id 1 who also has valid name and sex but do not have age.
* Hence, only LEFT join is valid.

Suggested codes to achieve the output:

| df\_after\_merge=df\_merge1.merge(df\_merge2, on= 'PassengerId', how='left')  df\_after\_merge[df\_after\_merge['Age'].isnull()] |
| --- |

### **Qn 6 context.**

****

### **Qn 6A. Given the scenario where I perform dropna function, what would happen if I perform the following line of code?**

| Sample2\_df.loc[3,] |
| --- |

In the example, row 3 is removed due to the dropna function. Although there are only 4 rows left, pandas do not automatically reset the index. Hence, as .loc is based on the index label on the df, it cannot find index label 3 and result in a key error.

### **Qn 6B. Given the scenario where I perform dropna function, what would happen if I perform the following line of code?**

| Sample2\_df.iloc[3,] |
| --- |

.iloc is an integer-location-based indexing. This means it is selected by the position and not by the index label. Hence, the last row is now positioned at index 3 after dropna function and will output the last row.