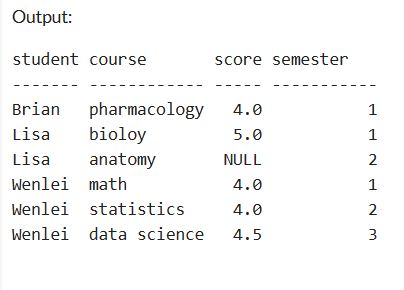
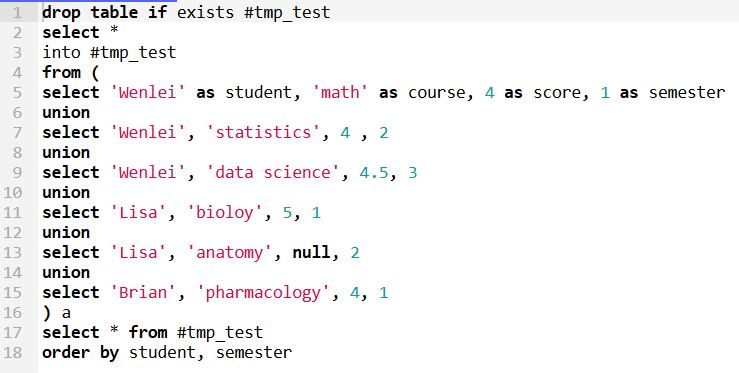
I wrote a post about using pandas to do some basic SQL operation a while back. Those are good if you just get started. When dealing more involved logic, often time than not, you will need to use analytics function and think how to implement that in the pandas as well. This post will focus on this part.

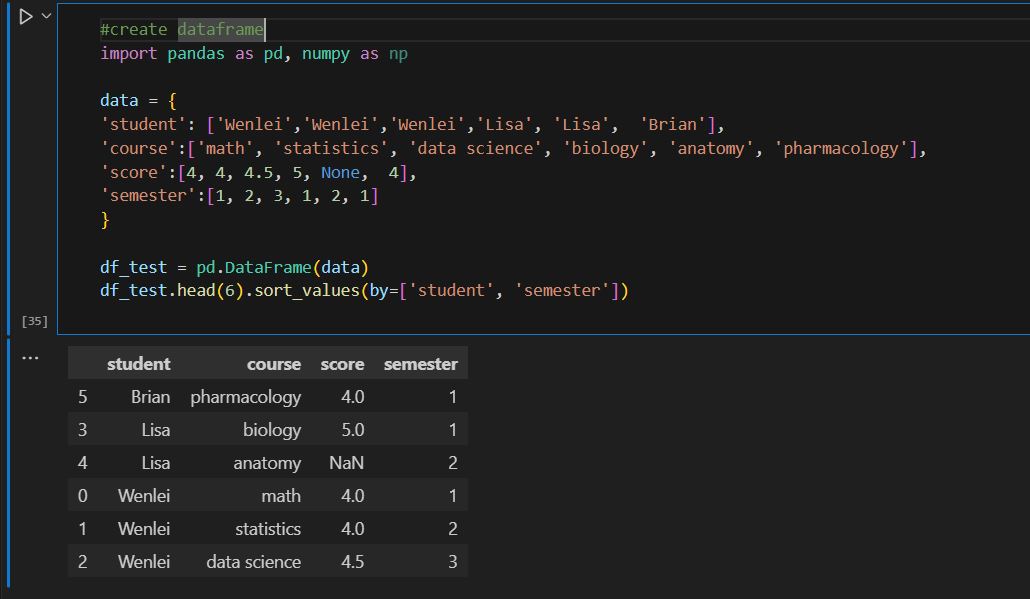
I will use the SQL Server online compiler for this post. The address is as follows. You can paste the SQL code in the window and run it to see results

<https://onecompiler.com/sqlserver/>

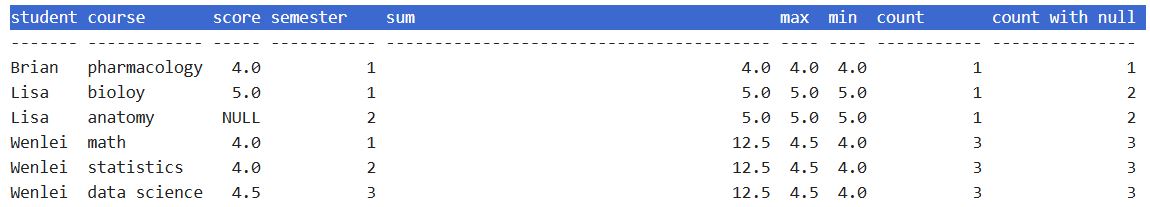
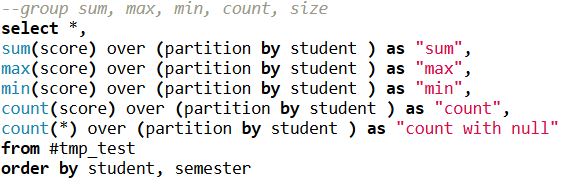
Let us first create a toy dataset. I use union to create a temp table, which will be using to demo what the results are supposed to be and compare with pandas result, which I will demo in the Jupyter notebook.

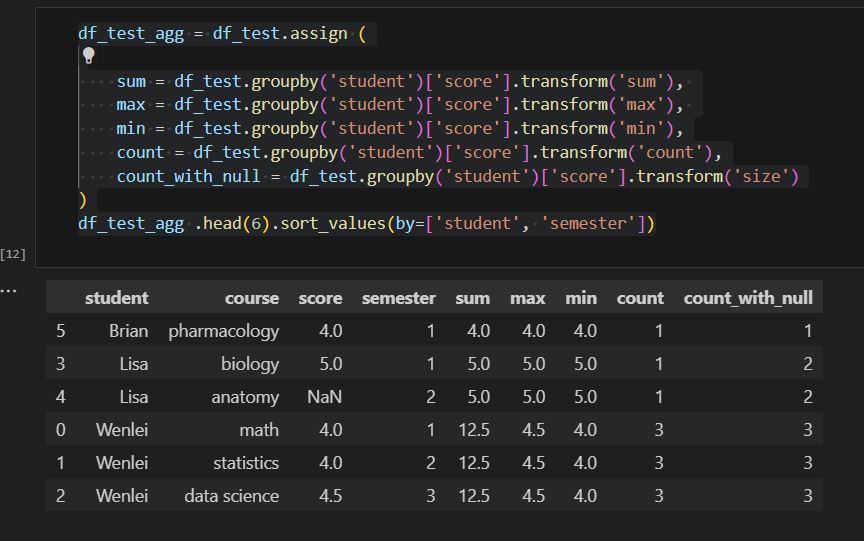


I deliberately include the null value here so that we can observe how SQL and pandas handle it. Pandas create a dataframe which likes tables in a database.

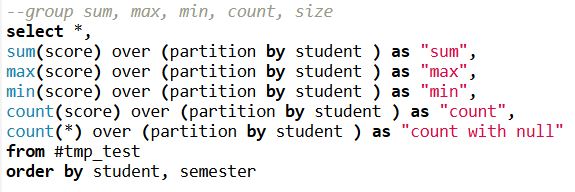


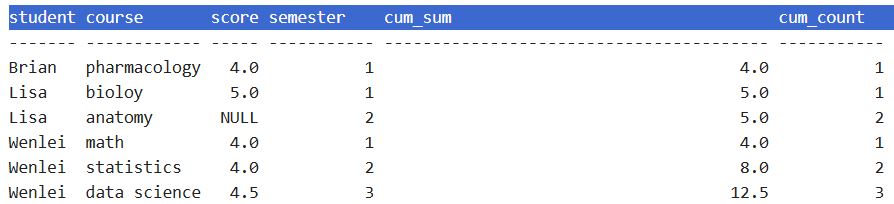
The first challenge is using regular aggregation like sum/avg. With over (partition by …) syntax, this allow you to get aggregation at each original row without reducing the row number. This will help when you need to calculate different metrics but don’t want to reduce the record. I have count(score) and cout(\*) here. The former will ignore null value in the score column, but count(\*) will return all record count.



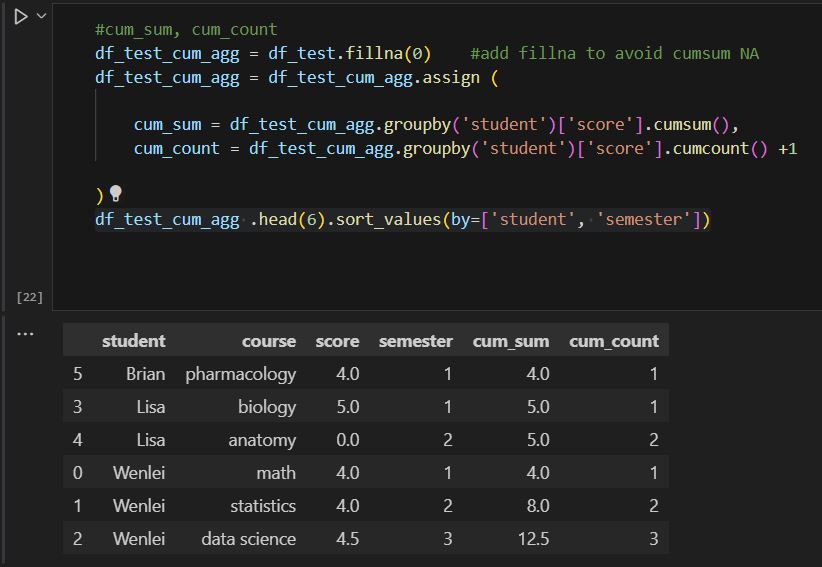
This can be achieved in pandas by groupby and transform function, Notice when you need to incude null, you will need to use size function instead. 

Another form of aggregation is cumulative aggregation. In SQL, you will need to add sequence via order by, so that SQL engine know how to add sequentially.

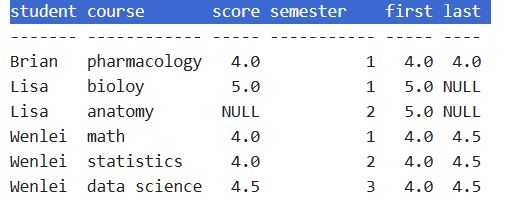
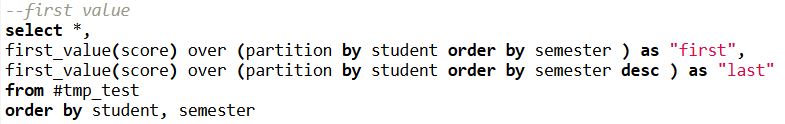




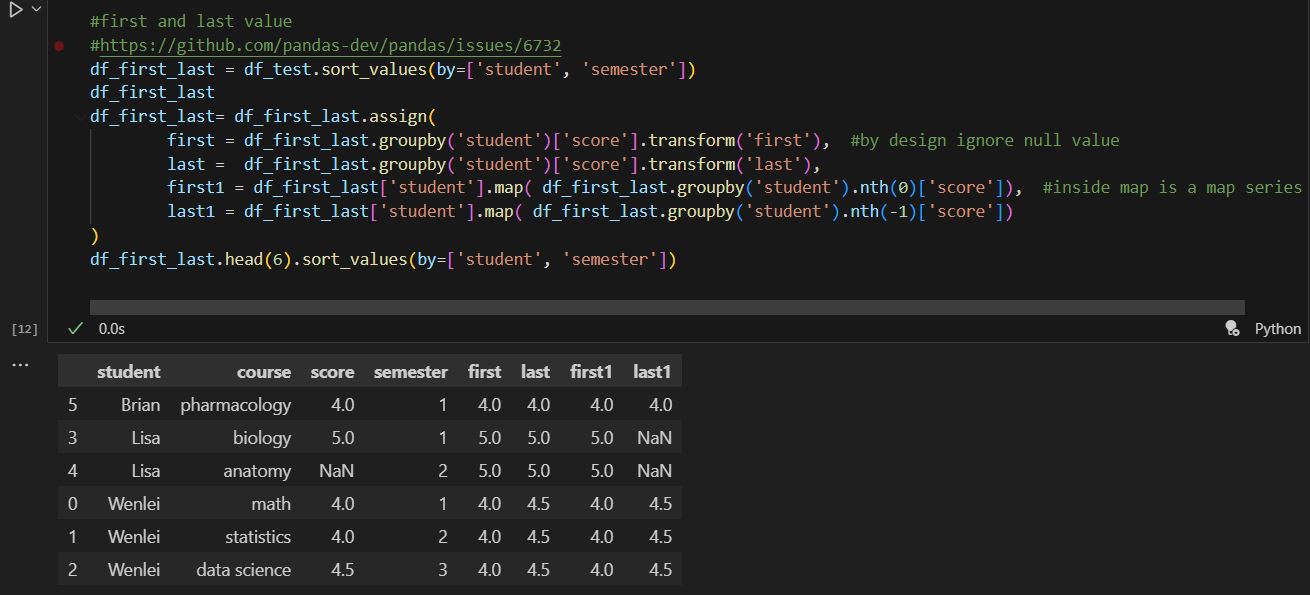
Because data contains null value, if we simply aggregate in pandas, null+number will end up with null. To fix that, we will need to fillna first, then we can use pandas cumsum and cumcount to get the result, because cumcount starts with 0, we will need to add 1 to match the results on the SQL side.



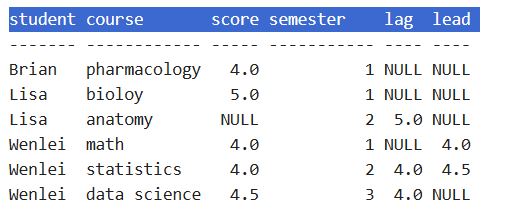
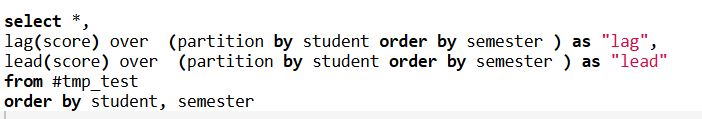
Sometimes, we need to find the first or last record in one group, based on certain sequence. This can be achieved by first\_value function combined with different order as follows.



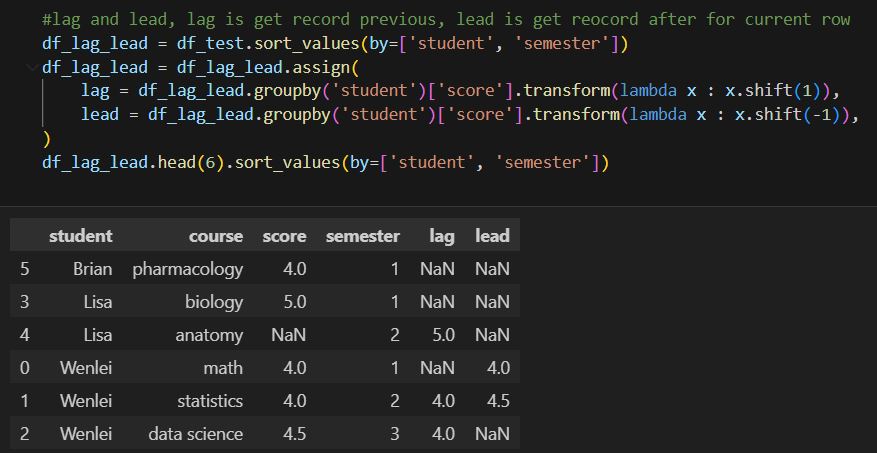
In pandas, it does has first and last function, but unfortunately, it exclude null value by default (see this github question). In order to simulate the same behavior in SQL, we will have to work around with nth function which only work with groupby object. The last two columns show same behavior as SQL, inside map function, df\_first\_last.groupby('student').nth(0)['score'] get first score for each group. If we use nth(-1), that will be last score.



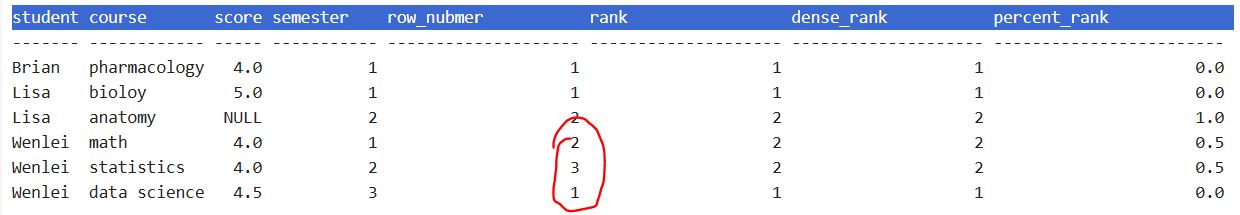
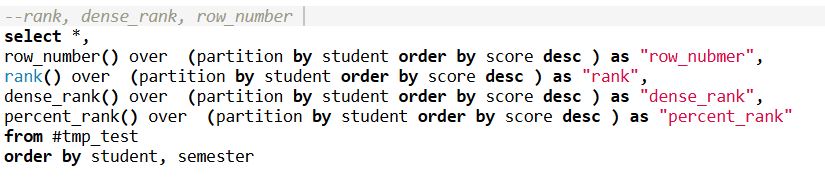
Occasionally, you will need to compare current record with other records in the same group. In SQL, you can use lead/lag. For e.g., you can do the followings. Lag means to compare with the previous record depending on the sequence you set in the order by. Lead to get next one.



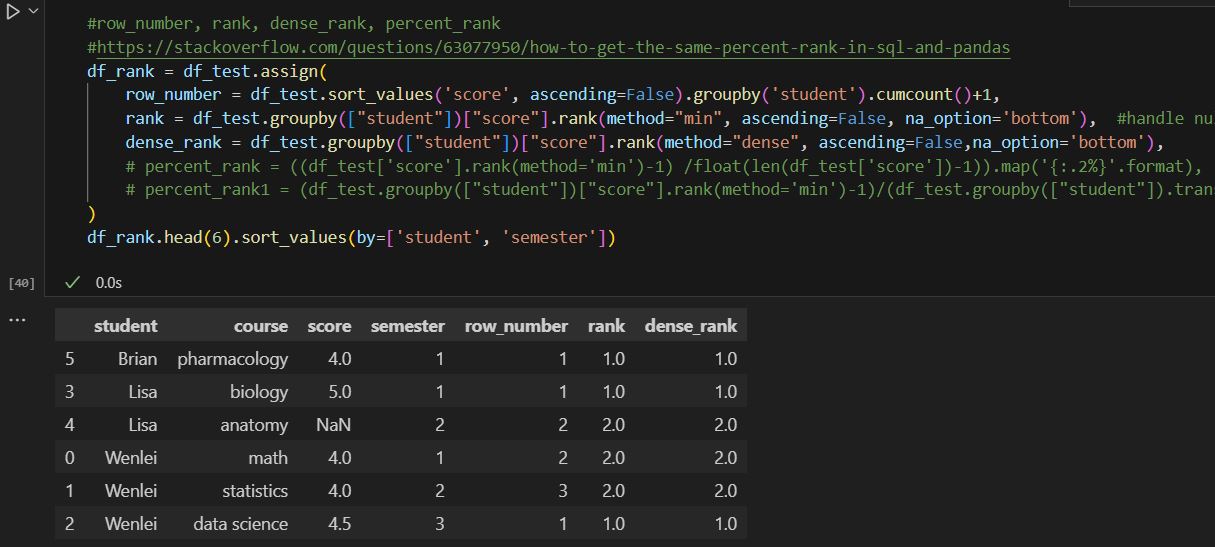
You can use transform and shift to achieve the same. Since we will need to pass in param, I use lambda function here. Another thing to notice, I would think to use -1 to get the privous record, but in pandas, it uses 1 ☺



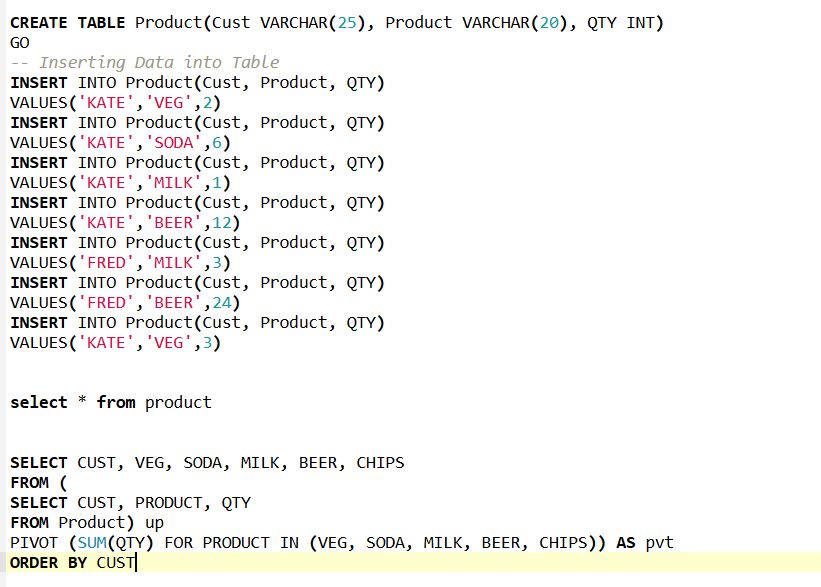
It is not uncommon in the real work, you will need to use row\_number, rank, and dense rank function among the same group. Row\_nubmer will always give the different number even value is equal, rank will give the same number if value is equal, but the sequence number will skip. However, Dense\_rank will not skip.

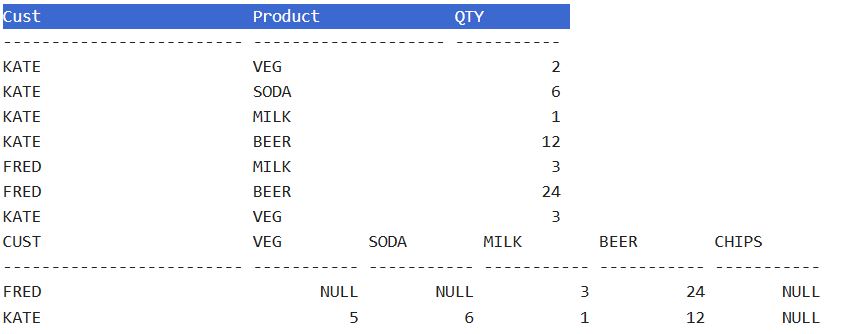


Obviously, I found a bug here in onecompiler.com marked by red circle. It should be 2, 2, 1 since math and statistics are both 4.0.

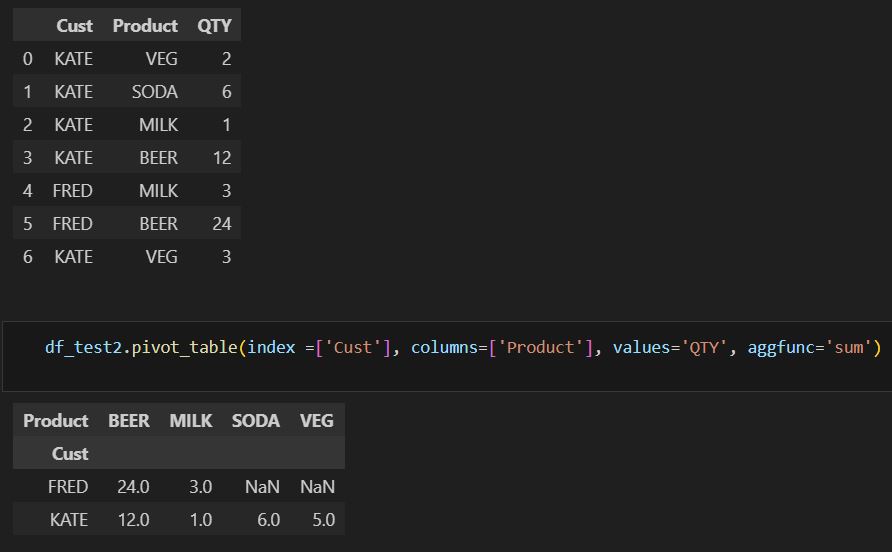
Pandas can handle this by use rank function. Just notice, pandas ignore null value, if you need to include that, add na\_option. 

You might come across the situation that require you to pivot table to get some insight. In SQL, you can use pivot function. Here we created a separate table demo this function. In the second part of result, you can see product column has been pivot to be column name. Also notice, that you can name the column you want. In this case, we add ‘chip’, which is not in the value of original product, but it does just fine. Recently, some database include more rank function, such as percent\_rank which is used to let you know where the record stands percentage wise, but there is no corresponding function yet on the pandas side, I have tried different ways, so far, no luck yet, but I include the stack overflow discussion if you would like to give a shot.





We create a df\_test2 dataframe containing same data. We can use pivot\_table function to tell what I wound like on the row and column and what would be value to be aggregate and what method I would like to aggreage (mean, sum, et al). Notice, there is no way, you can insert “chips” here. Also, pivot\_table has a brother function called pivot. Pivot\_table can handle multiple columns. That is why you see the parameter is passed in as list. Pivot can only handle one column at a time.



Pandas also offer function to pivot row to column without aggregation. In this case, you will need to use unstack and stack function. You will need to set index to the column you would like to pivot. By default, it will pivot the inner layer. But you can use level parameter to adjust. You can reverse the unstack with stack function. 