General advice about how to work through complicated queries

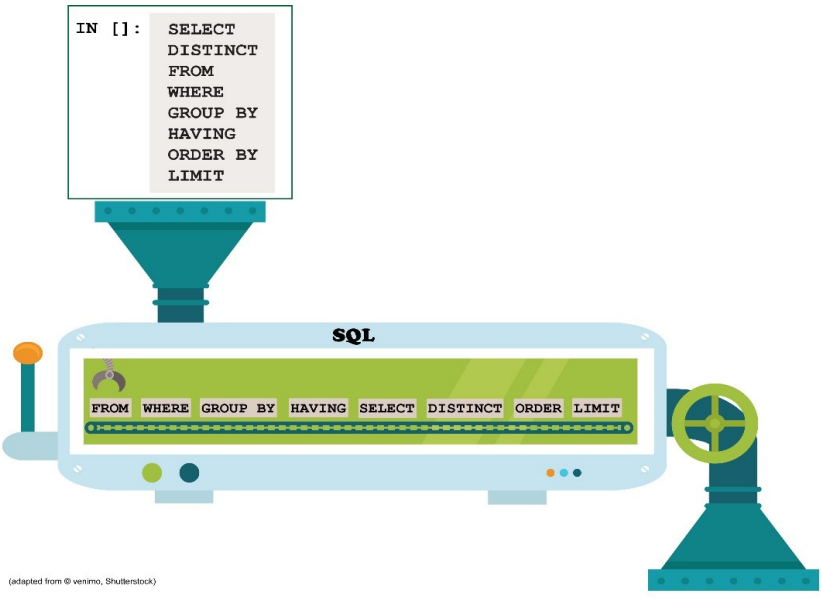
You will learn how to write some elaborate queries in this course. You will likely have to do a lot of troubleshooting, but in the end, you will be very proud of all you have accomplished!

To prepare you for the work you will do, I wanted to give you some advice about how to tackle a complicated query. My first piece of advice is to use words to describe all the information you need to answer the question the query is meant to address. You might even want to write out an outline of the information and how it relates to each other on a piece of paper. Then, once you can describe the information you need in words, identify the fields in your dataset that represent that information, and what tables they are kept in. Write down what tables you need to join in order to combine the fields you are interested in. After that, break the computations you need to combine into pieces, and work through each piece of a query step by step, never adding more to the query unless the steps you have already completed execute correctly.

One strategy some people use to help them write queries with multiple joins is to start by writing all the join clauses first with SELECT * statements. When it is clear that the joins are outputting the information you intend, add your column names in the SELECT statement, and carefully check that they are aggregated appropriately. Next, add your GROUP BY statement (if needed), and finish with your WHERE/HAVING statements. As you work through the process, you will find where you need subqueries. Troubleshoot each subquery on its own, separately, before adding it to a main query.

A different strategy some people use is to write as much of a query as is possible with one table first, starting with the table from which you will retrieve the greatest number of columns. Once the query works with one table, add the next table and check the results to make sure they make sense. Keep adding one table at a time, checking the results as you progress to make sure the syntax is correct, until the query is complete.

Yet a third strategy you can try is writing the query according to the order the query will be processed. Recall this figure from MySQL Exercise 6:



When queries have a lot of nested clauses, it is easy to lose track of what gets executed first, so even if your query runs, it may give different results than you expect. If you write the clauses of your query in the same order as the clauses appear on the conveyer belt in the figure, it will help make sure the logic behind your query is reflected by the output of the query.

I suggest trying out many different query-writing strategies until you find out what works best for you. As you learn, I strongly encourage you to share advice with your peers about what you find most successful. Also, don't forget to use self-explanatory aliases, and to format your queries with indentations and strategic capitalization so that they are easy to read and troubleshoot. Good luck, and have fun!