**Don’t Start Your SQL Queries with the ‘Select’ Statement**

**Follow this right approach to write your SQL queries**

**The Problem**

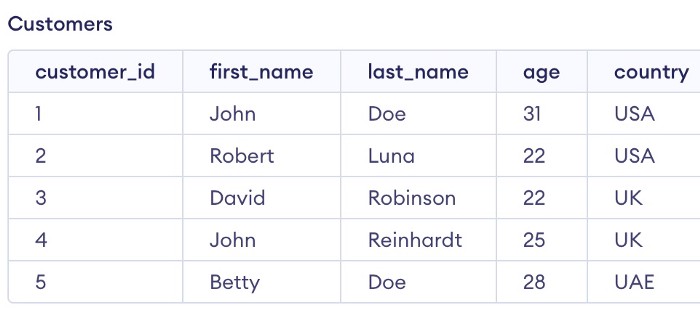
The majority of developers start writing their SQL queries with the ‘SELECT’ clause, then write ‘FROM’, ‘WHERE’, ‘HAVING’….and so on. But this is not the ‘right’ way of writing your SQL queries as this is very prone to syntactic errors…

**The Solution**

The ‘ideal’ query writing sequence should be in line with how the SQL executor executes the queries. This will ensure that you don’t commit any syntactic errors and write efficient SQL queries. You will know how to filter data before performing join, when to use ‘HAVING’ or ‘WHERE’ clause and more.

In this blog post, we will look at the ‘ideal’ way of writing a SQL query that will help you become an efficient SQL developer.

We will be using the Customer and Order tables (below) to find the top 2 customers from the USA/UK who have a total spend of more than $300.



Customer Table (Image by Author)



Order Table (Image by Author)

Let’s dive into the right way of writing SQL queries.

**1. Always start with FROM/JOIN**

Intuitively, the first step is to read the tables using the FROM clause and perform JOIN(if required). So, you should always start your query with the ‘FROM’/‘JOIN’ statement.

FROM Customers  
INNER JOIN Orders  
ON Customers.customer\_id = Orders.customer\_id

We can also filter rows from the input tables even before the join is executed. We can do this by adding the ‘AND’ clause after the ‘ON’ clause of the join.

-- Filter happens before Join  
  
FROM Customers  
INNER JOIN Orders  
ON Customers.customer\_id = Orders.customer\_id  
AND country in ('USA','UK')

**2. Then move to WHERE**

The second clause in the order of execution is the WHERE clause. It is used to filter the data tables after the join has been applied.

The WHERE clause is very helpful to reduce the number of rows especially when we are working with big datasets having millions of rows.

FROM Customers  
INNER JOIN Orders  
ON Customers.customer\_id = Orders.customer\_id  
WHERE country in ('USA','UK')

**3. Then use GROUP BY**

Group By clause should be written after the Where clause. It is used to group the rows based on the selected column/columns.

In the following query, we are grouping the rows based on the customer id. After grouping, each customer id will have one row in the output. We generally use aggregations(sum, min, max, etc.) when we group the data. In this example, we will find the sum of the amount column in the Orders table.

FROM Customers  
INNER JOIN Orders  
ON Customers.customer\_id = Orders.customer\_id  
WHERE country in ('USA','UK')  
GROUP BY Customers.customer\_id

**4. HAVING after GROUP BY**

The HAVING clause gets executed after GROUP BY, it is used to filter the aggregated rows that were generated in the group by operation.

In our example, we will filter the sum of the amount spent by each customer to be greater than 300.

FROM Customers  
INNER JOIN Orders  
ON Customers.customer\_id = Orders.customer\_id  
WHERE country in ('USA','UK')  
GROUP BY Customers.customer\_id  
HAVING sum(amount) >300

The WHERE clause gets executed before GROUP BY while HAVING gets executed after it. So, the WHERE clause cannot filter aggregated data.

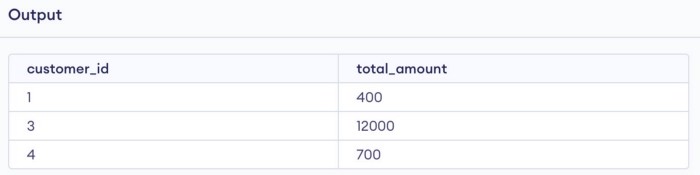
**5. Then write the SELECT clause**

Columns that we want to show in the output are selected using the SELECT clause.

If we group our data using the GROUP BY clause, we need to select the grouped column using the SELECT statement.

In our example, we will select the customer id and sum(amount) to show the spending corresponding to each customer.

select Customers.customer\_id, sum(amount) as total\_amount  
FROM Customers  
INNER JOIN Orders  
ON Customers.customer\_id = Orders.customer\_id  
WHERE country in ('USA','UK')  
GROUP BY Customers.customer\_id  
HAVING sum(amount) >300



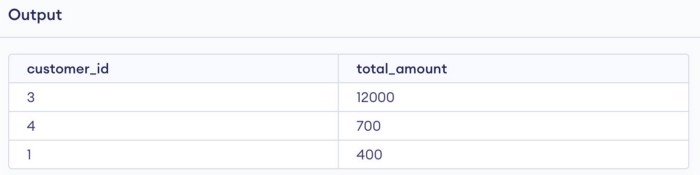
Output (Image by Author)

**6. Use ORDER BY after the SELECT clause**

After selecting the columns, the next step is to provide the order in which we want to output the rows.

In our example, we can use the ORDER BY clause to order the rows in descending order of total spend.

SELECT Customers.customer\_id, sum(amount) as total\_amount  
FROM Customers  
INNER JOIN Orders  
ON Customers.customer\_id = Orders.customer\_id  
WHERE country in ('USA','UK')  
GROUP BY Customers.customer\_id  
HAVING sum(amount) >=300  
ORDER BY total\_amount desc



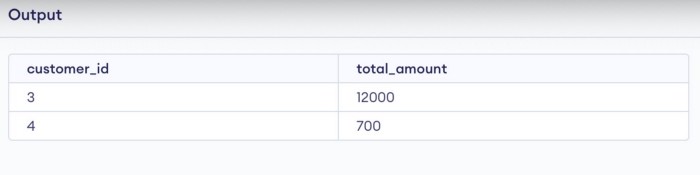
Output (Image By Author)

**7. Write the LIMIT clause at last!**

The last step in the writing sequence is to limit the number of rows that we want to see in the output.

In our example, we can limit the total number of output rows to 2.

SELECT Customers.customer\_id, sum(amount) as total\_amount  
FROM Customers  
INNER JOIN Orders  
ON Customers.customer\_id = Orders.customer\_id  
WHERE country in ('USA','UK')  
GROUP BY Customers.customer\_id  
HAVING sum(amount) >=300  
ORDER BY total\_amount desc  
LIMIT 2



Output (Image by Author)

**Conclusion**

We looked at the ideal way of writing the SQL queries which is in line with how the SQL queries are executed.

I hope you will be writing your SQL queries in the below sequence if you are not doing that already.

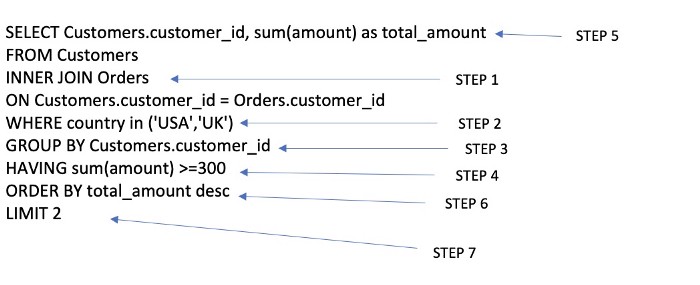


Image by Author

**Thank You for reading!**

*You can get all my posts in your inbox.* [***Do that here****!*](https://anmol3015.medium.com/subscribe)

*If you like to experience Medium yourself, consider supporting me and thousands of other writers by* [***signing up for a membership***](https://anmol3015.medium.com/membership)*. It only costs $5 per month, it supports us, writers, greatly, and you get to access all the amazing stories on Medium.*

***Follow me*** *to see my data science posts in your feed.*