Aggregating Statements in MySQL

John Wensink

ITS410 Database Management

Colorado State University-Global Campus

Dr, Dan Morrill

December 15, 2019

Aggregating Statements in MySQL

This week we discovered how to use aggregating statements in MySQL such as AVG, SUM, COUNT, MAX, and MIN (W3Schools, 2019.). We can use aggregating functions to analyze many values and perform an operation to return a single desired value. Aggregating functions work best on columns with numeric data types (Boronczyk, 2019). This week’s assignment asks us to use the World-X database to display the languages spoken in each country and order the results based on which language is more common in each country. Surprisingly, the desired output can be returned without an aggregating function using the following query:

**SELECT country.name AS 'Country Name',**

**language AS 'Language',**

**percentage AS 'Percentage'**

**FROM country, countrylanguage**

**WHERE country.code = countrylanguage.countrycode**

**ORDER BY country.name, percentage DESC, language**

The syntax here is relatively straightforward. SELECT country.name, language, and percentage FROM country, countrylanguage returns every country and every language even if the percentage is 0%. This is not useful and it makes sense to use a WHERE modifier with an ORDER BY clause in order to manipulate the output into what we need. The second option is more relevant to this week’s curriculum where we use the *classicmodels* database to aggregate the total sales for years after 2003. When using SQL to create a query, it is useful to look out for keywords that should be interpreted to require different clauses in your query such as average, total, minimum, maximum. and count (Oracle, 2019). Here, the keyword that suggests that we need to aggregate is ‘total’ in which the SUM function will yield the desired results. The query that outputs the desired results is:

**SELECT sum(amount)**

**FROM payments**

**WHERE paymentdate > 2002-12-31**

Again, this data is not very useful as all orders and payments in *classicmodels*were taken between January 13, 2003, and June 6, 2005. The result is the same as a SELECT \* FROM payments query. More interesting, would be ‘what was the company’s net revenue in 2004?’ Here we can see the utility of the SUM function in MySQL with the query:

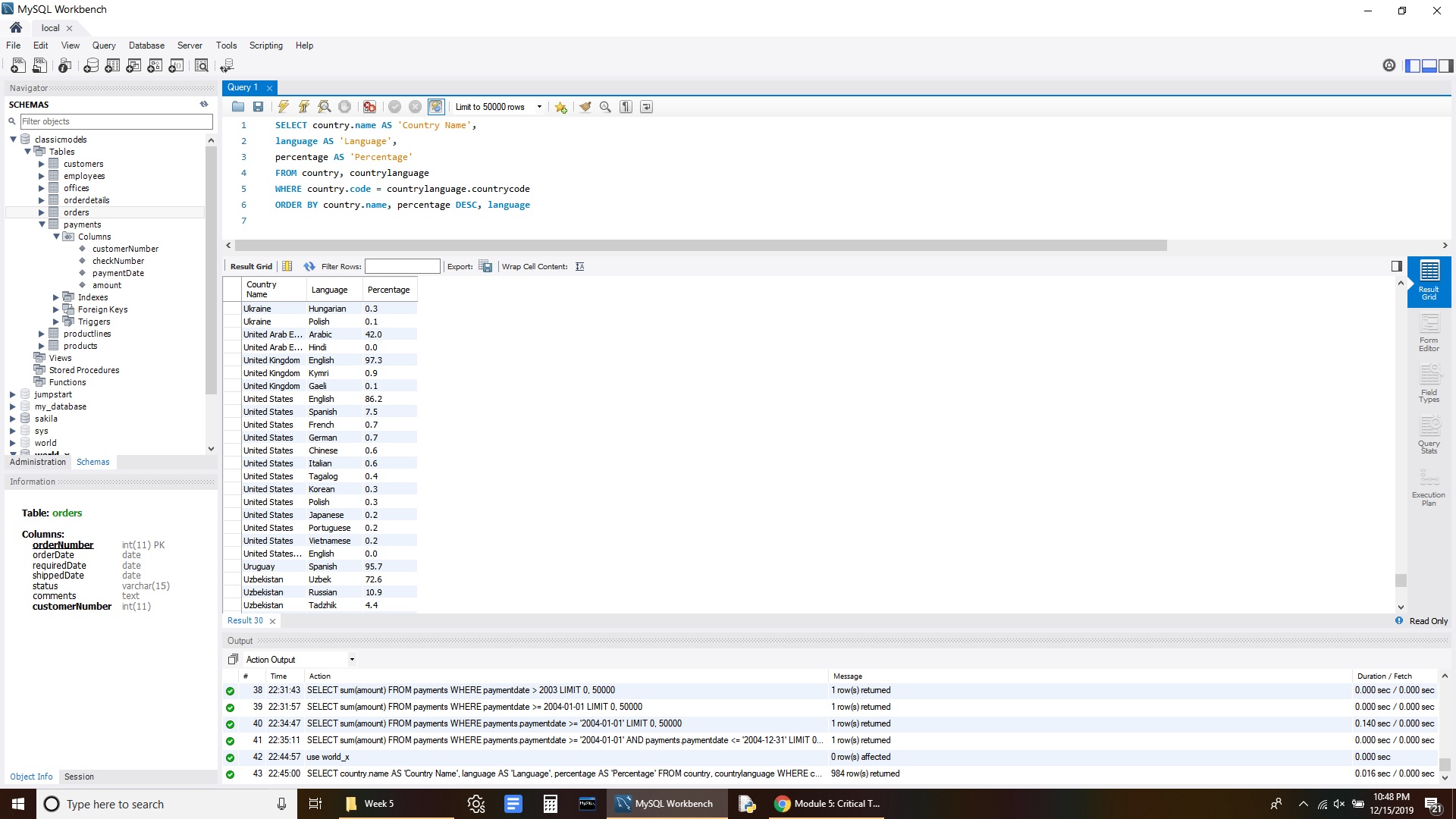
**SELECT sum(amount)**

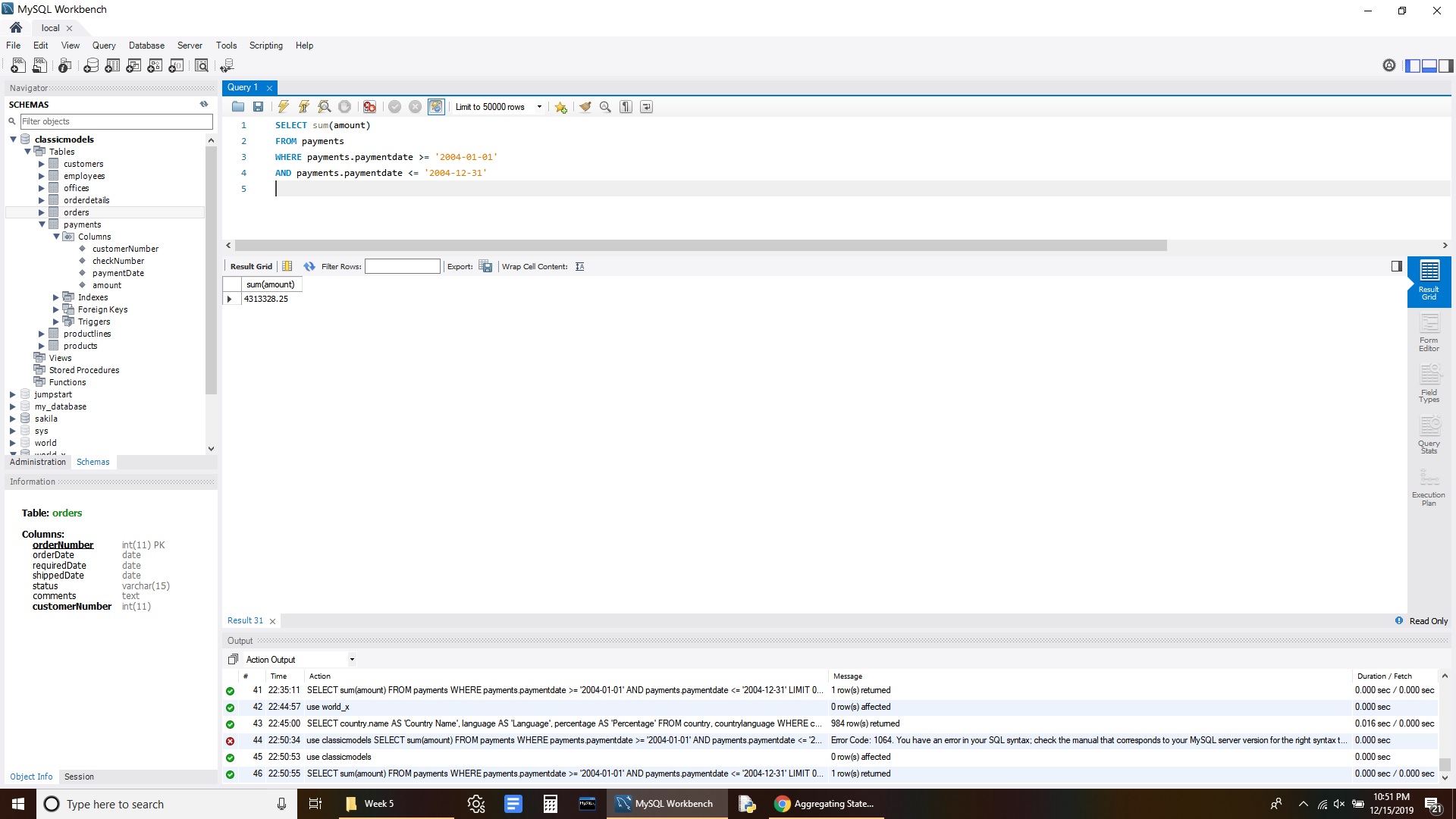
**FROM payments**

**WHERE payments.paymentdate >= '2004-01-01'**

**AND payments.paymentdate <= '2004-12-31'**

This query returns that the company’s net revenue was $4,313,328.25 in 2004 and we have successfully made meaningful use of an aggregation function in MySQL.





References

Boronczyk, T. (2015). Jump start MySQL. Retrieved from <https://bookshelf.vitalsource.com/#/books/9781457192838/cfi/69!/4/4@0.00:16.1>

W3schools. (n.d.). SQL COUNT(), AVG() and SUM() Functions. Retrieved December 15, 2019, from <https://www.w3schools.com/sql/sql_count_avg_sum.asp>.

Oracle Corporation. (2019). MySQL 8.0 Reference Manual :: 12.20.1 Aggregate (GROUP BY) Function Descriptions. Retrieved December 15, 2019, from <https://dev.mysql.com/doc/refman/8.0/en/group-by-functions.html>.