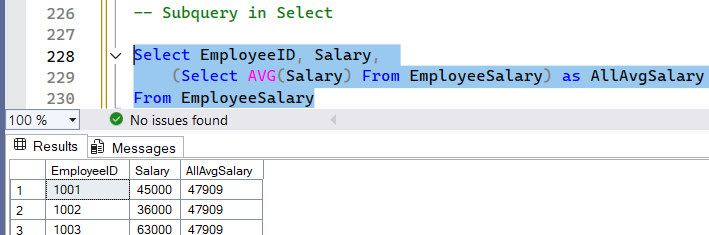
**SUBQUERIES (in the SELECT, FROM, and WHERE Statement)**

**-**Also called nested queries or inner queries

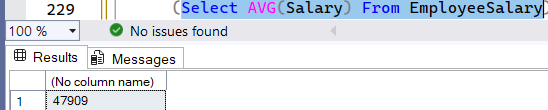
SQL Subquery: A Comprehensive Guide

<https://www.datacamp.com/tutorial/sql-subquery>

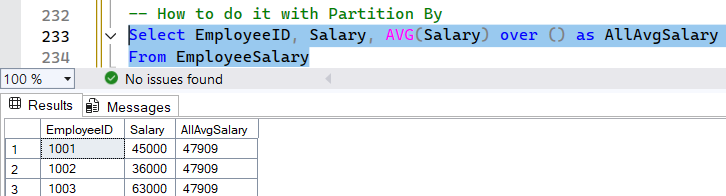
Used case 1:



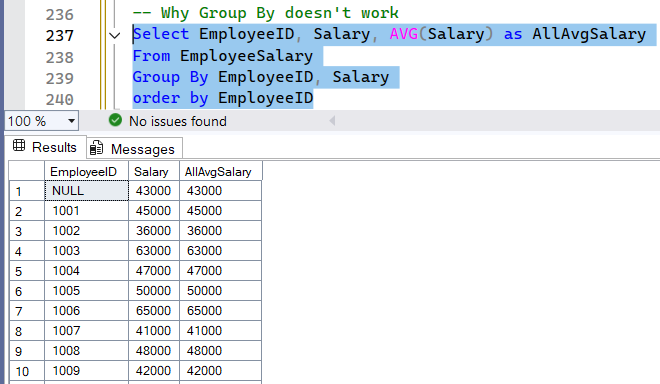
Checking subquery:



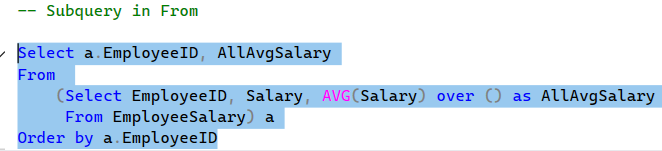
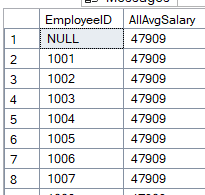
Use case 2: with partition



Use case 3: using group by doesn’t work

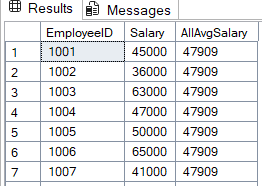


Use case 4: Subquery in ‘From’. More like CTE and temp table but slower. Subquery needed to be written each time compared to CTE and temp table that is only written once and can query many time from it

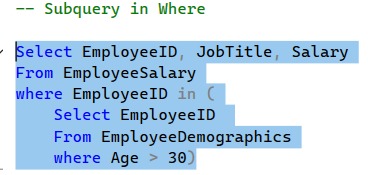
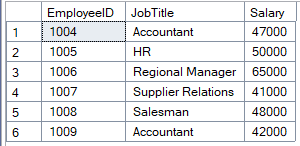
 

What the subquery looks like

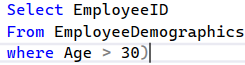
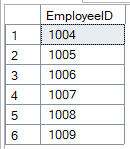




Use Case 5:

What subquery looks like

**Project 1 (2:17:09) LEARNINGS**

1. SQL server only accepts .csv file. CSV files don’t store formatting, including things like:

* Autofit column width
* Font styles
* Cell colors
* Borders
* Formulas (they get converted to values)

So when saving excel file, which should you use?

* Use regular "CSV (Comma delimited)" unless you're specifically dealing with an old system that requires one of the others.
* If you're sharing files with people on Macs or Linux, using UTF-8 encoding (you may have to save it as CSV UTF-8) is usually more important than the specific CSV type.

### How to keep the autofit column widths:

If you want to **preserve your formatting**, including autofit widths, do **one of these** instead:

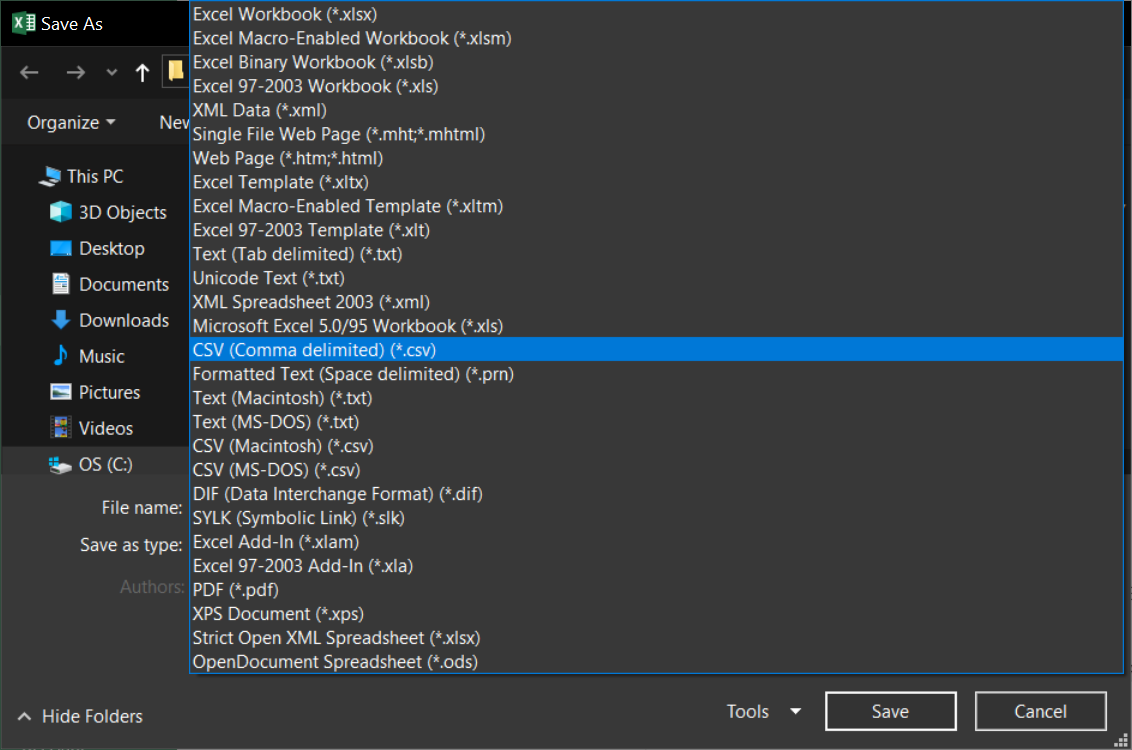
#### ✅ Option 1: Save as Excel Workbook (.xlsx)

* File → Save As → Choose **Excel Workbook (\*.xlsx)**
* This will keep everything: column widths, formatting, formulas, etc.

#### ✅ Option 2: Save a copy in both formats

If you need a **CSV for sharing or importing into another system**, but also want to keep your formatting:

1. Save your file as **Excel Workbook (.xlsx)** first — this keeps all your formatting.
2. Then do **File → Save As → CSV (.csv)** as a copy for export.

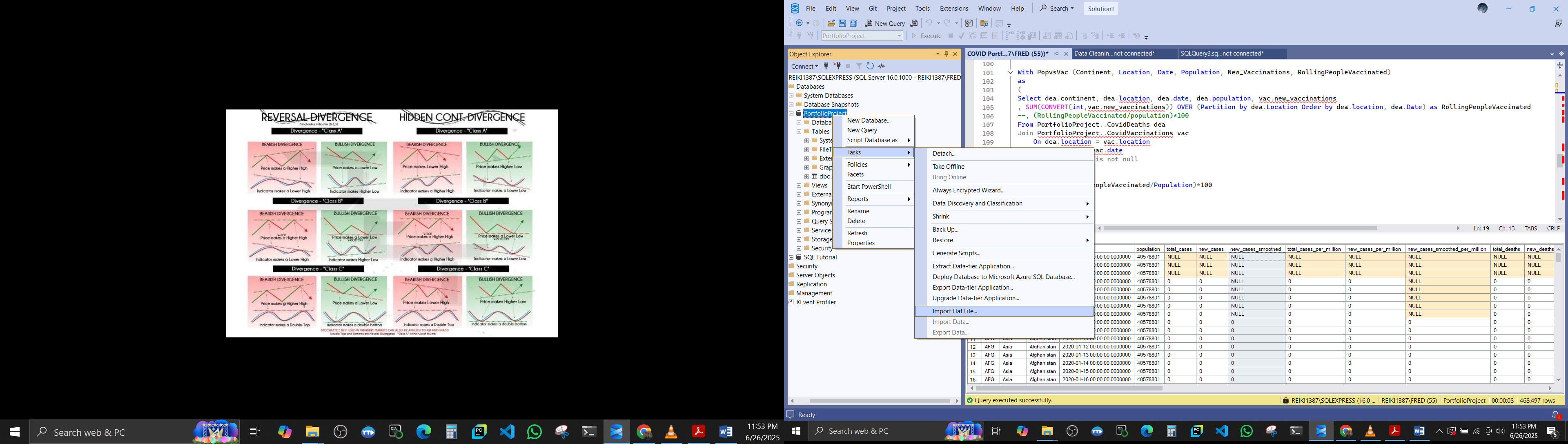


2. In .xls file format the columns so that the file type of the numeric column is “numeric” or ‘float and the date is in date format and other data type according to its type.

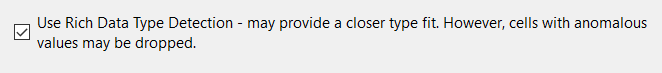
*Select column or columns –right click- format cell- select the category*

This is a necessary step because some numbers have comma or symbols

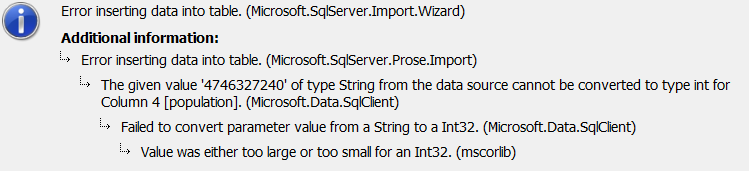
1. In uploading .csv to SQL

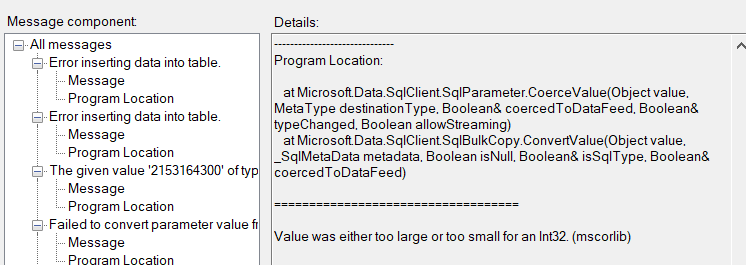


1. Use Rich Data Type Detection to automatically detect datatype based on the formatted cells in csv

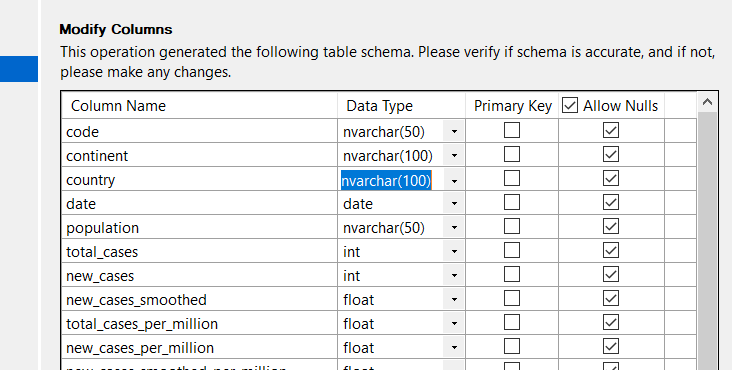


1. Sometimes there will be an error where a numeric value cannot be converted to in so you will opt to make it a nvarchar and cast it during query



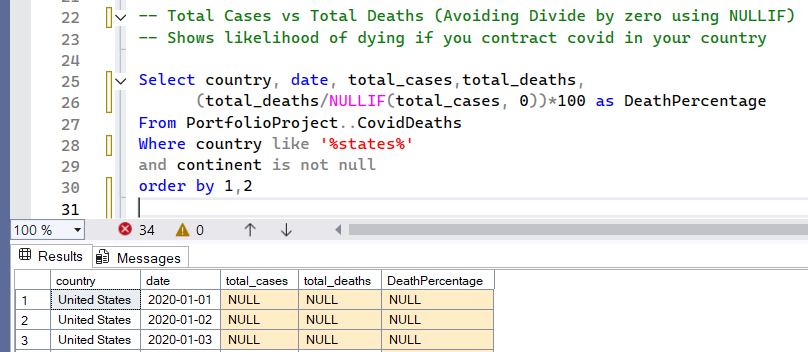


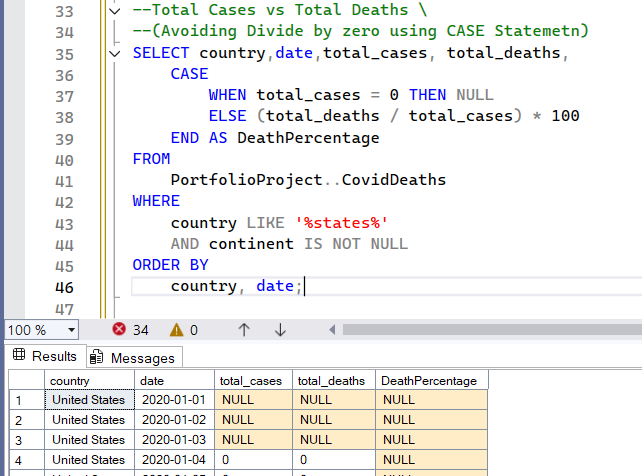
DO NOT FORGET TO ALLOW NULLS



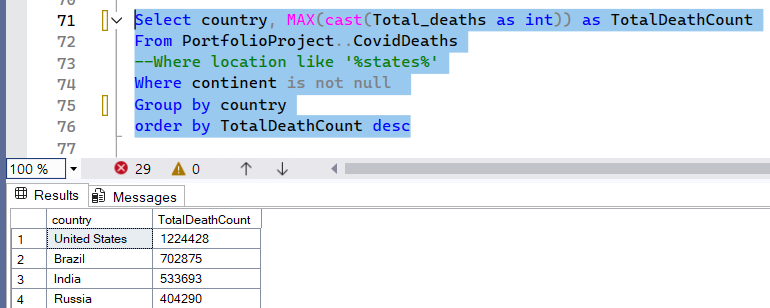
QUERYING

* 1. Avoiding zero division

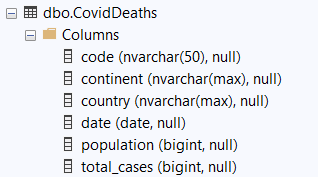




* 1. Querying using casting



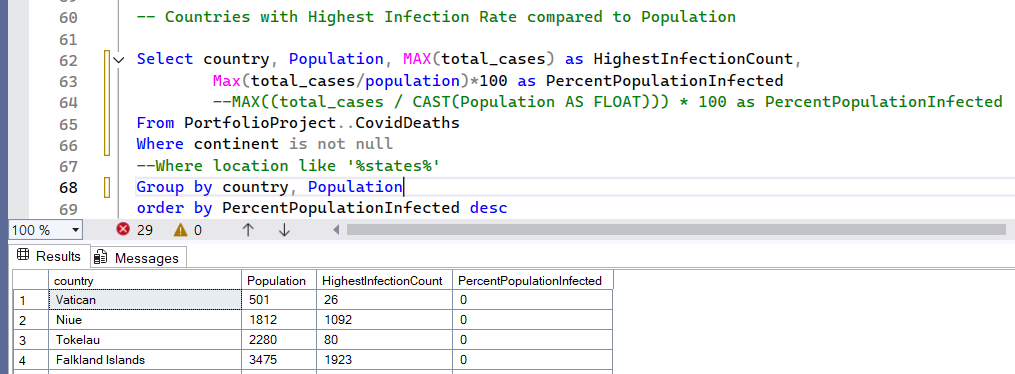
But you can check on the table columns what data type each column is



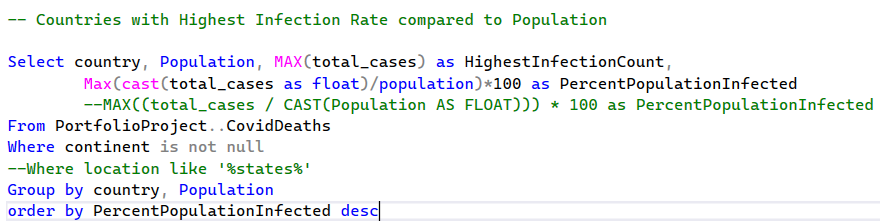
* 1. Dividing ‘bigint to bigint’ or ‘int to int’ result to zero when the dividend is smaller (The top number is our dividend and the bottom is the divisor).

When both total\_cases and population are BIGINT, SQL Server performs **integer division** by default, which means it **truncates** any decimal result

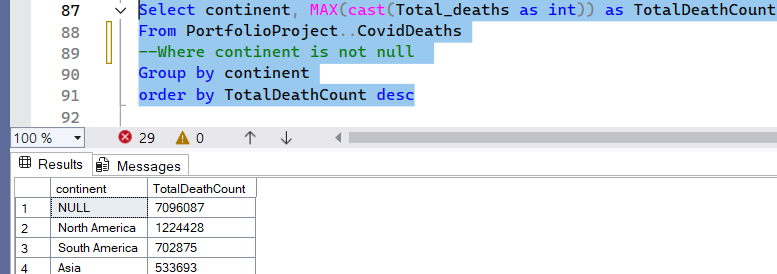
In the example below, the division result is very small (e.g., 0.0000123), the result is truncated to zero



To rectify, cast either one to float to avoid returning a ‘bigint’ or “int” which truncates the decimal



* 1. Some columns have NULL. To rectify that, use WHERE



* 1. Rolling number

The error:

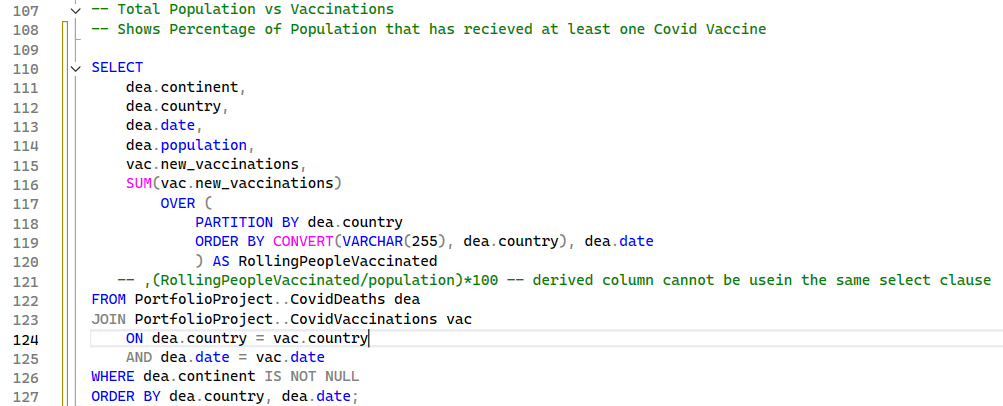


means that you're using a LOB (Large Object) data type—like TEXT, NTEXT, IMAGE, or VARCHAR(MAX)—in a RANGE-based OVER clause (ORDER BY inside a window function), and SQL Server doesn’t allow that. In below example, ‘country’ with data type ‘nvarchar(max)’ is the LOB

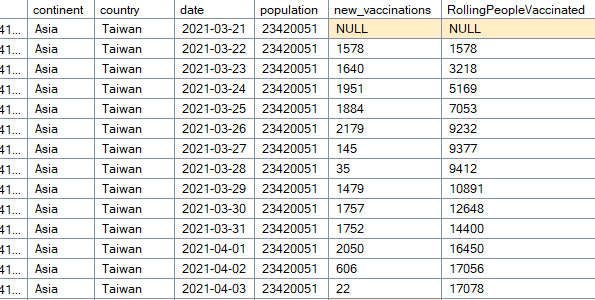




CONVERT- another way of casting



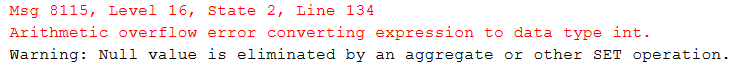
Output is a rolling number



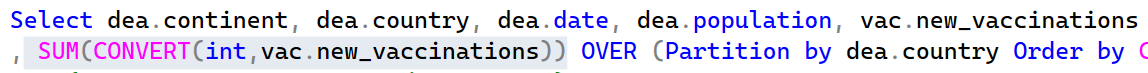
NOTE:

In SQL Server, **you can’t reference a derived column alias (like RollingPeopleVaccinated) in the same SELECT clause** where it’s defined. That’s because SQL Server evaluates expressions in the SELECT clause all at once—it doesn’t process them top-to-bottom.

Another error:

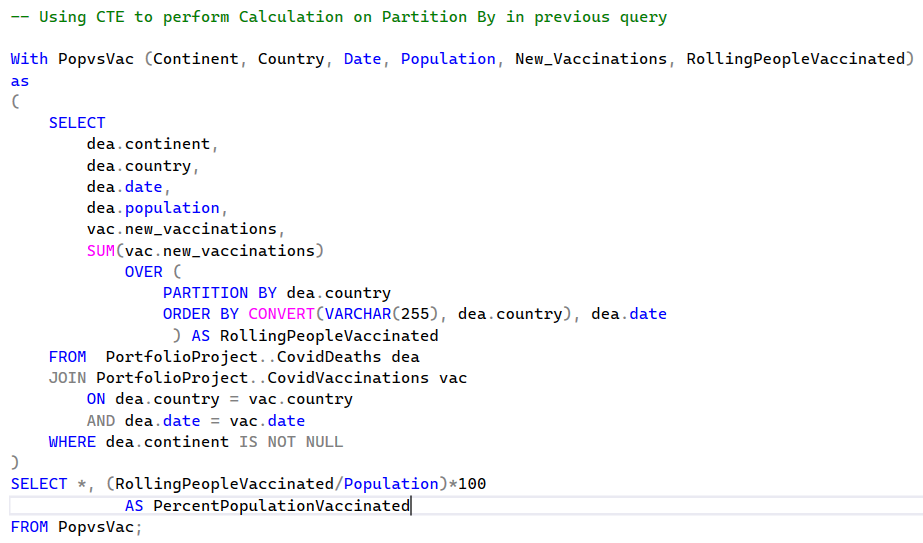


This part:

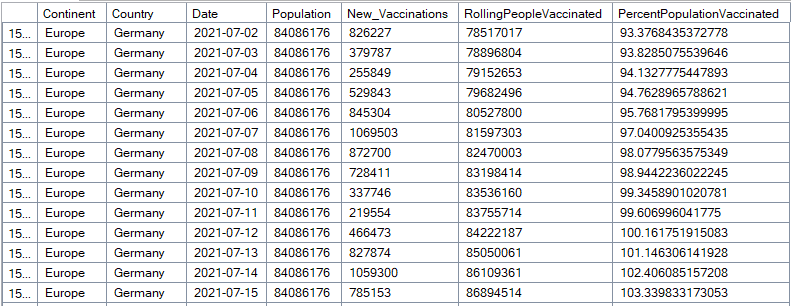


Where ‘new\_vaccinations’ is ‘bigint’ is throwing an **overflow error** because you're **converting a BIGINT to an INT**, and the value (or the total after summing) **exceeds the maximum size of an INT**, which is:

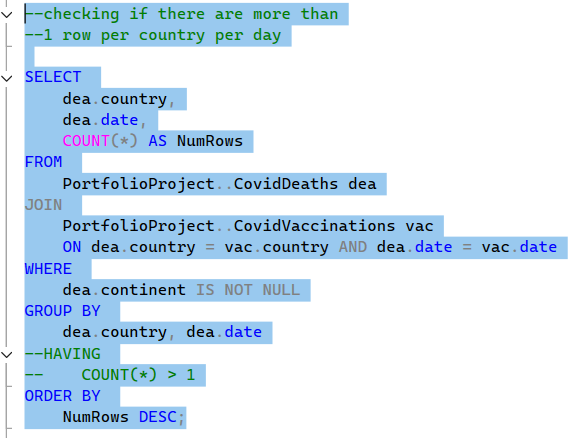
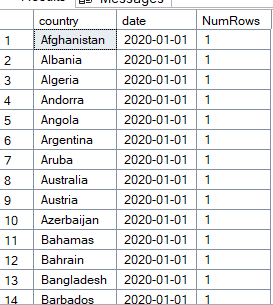
* INT: −2,147,483,648 to 2,147,483,647
* BIGINT: much larger: −9 quintillion to +9 quintillion
  1. Rolling number using CTE



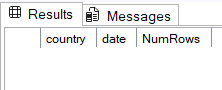
Most countries exceed total population vaccinations



First, check for duplicate rows. All country only have one row per day

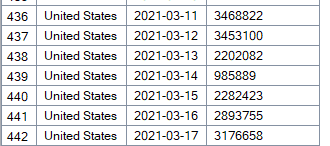
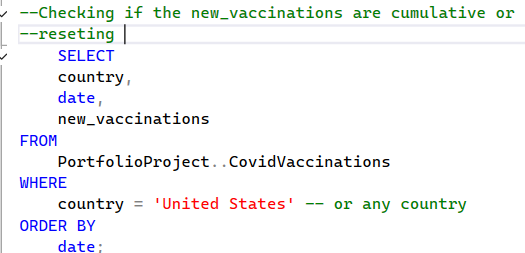
 

Executing this condition result to no rows are greater than 1 per day per country

Second is to check if the new\_vaccinations per day are cumulative. It should be reseting.

It seems like the values are going up and down. It means it is not cumulative



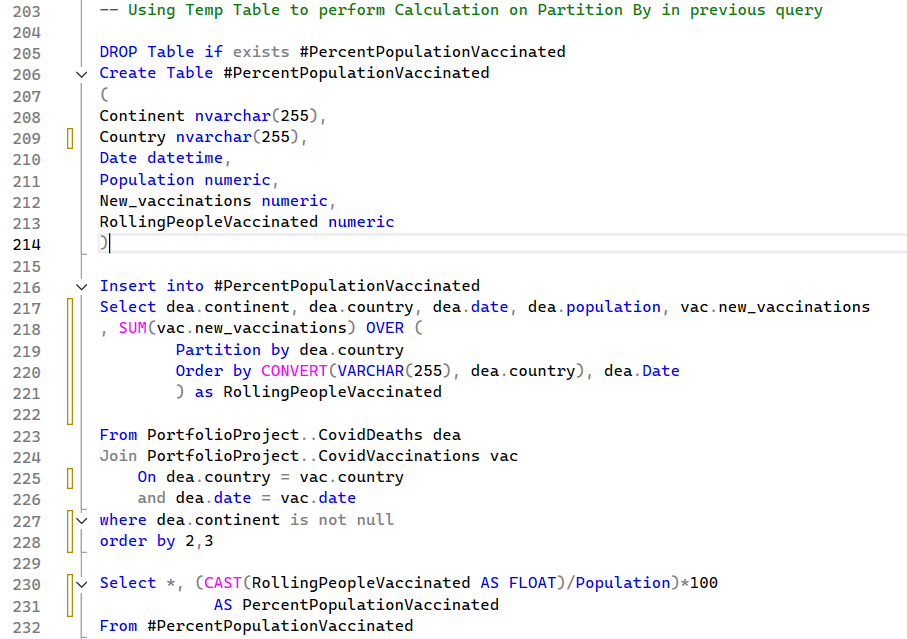
### Next Steps:

If you're sure:

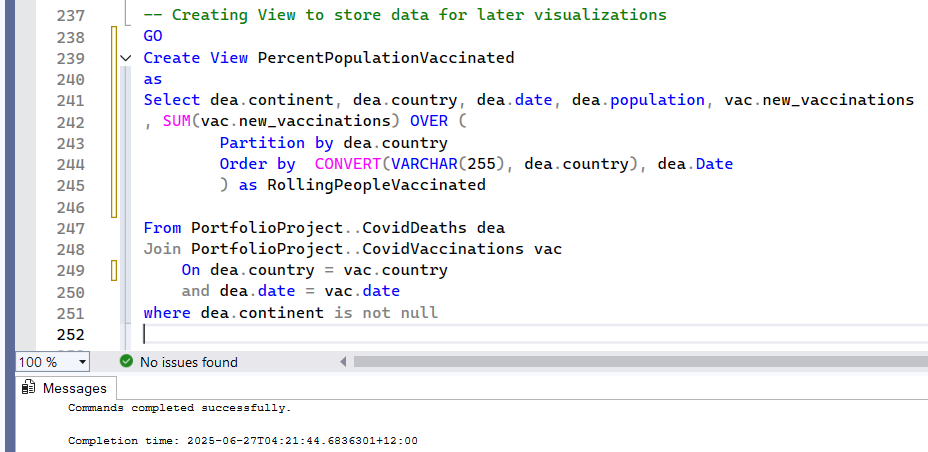
* new\_vaccinations is daily, not cumulative,
* and population values are reasonable,
* and you're not duplicating rows,

then your query is likely working correctly — but **some countries legitimately report over 100%** due to:

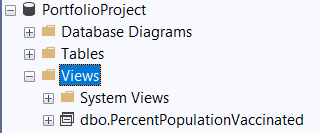
* **multiple doses per person**
* **vaccinations of non-residents** (tourists, migrants)
* **data inconsistencies**
  1. Using temp table



* 1. CREATE VIEW



It is located in



Minor error:





That red squiggly line and the error "CREATE VIEW must be the only statement in the batch" is a common issue in SQL Server Management Studio (SSMS) or similar tools when you run CREATE VIEW along with other statements in the same batch (like a GO or a SELECT).

To fix it:

1. Either **highlight just the CREATE VIEW statement** and run it alone
2. Or **put a GO statement before it** to start a new batch