Lab 05: Home Improvement: AI Edition

**Introduction:**

You may have found that sometimes the SemanticFunction works and provides an accurate result, and sometimes it appears to calculate things incorrectly or have flaws in its logic.

In this lab you improve the the capabilities in two ways:

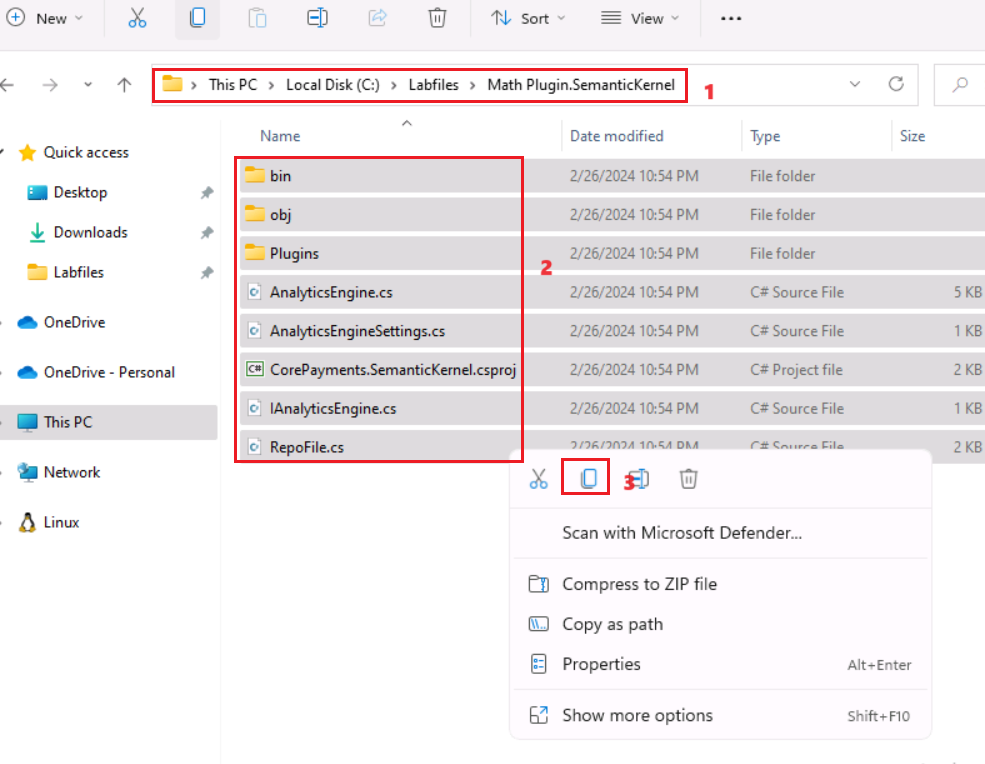
* Improve the prompt text to reduce any dependency on the model's parametric knowledge (this is the knowledge the model has learned during pre-training) so that it only uses the data you supply and the user query in the processing.
* Improve the handling of numbers. Sometimes numbers are better handled with native code. You will need to improve SemanticFunction by leveraging the SequentialPlanner that uses a Semantic Kernel plugin for numeric operations in addition to your SemanticFunction.

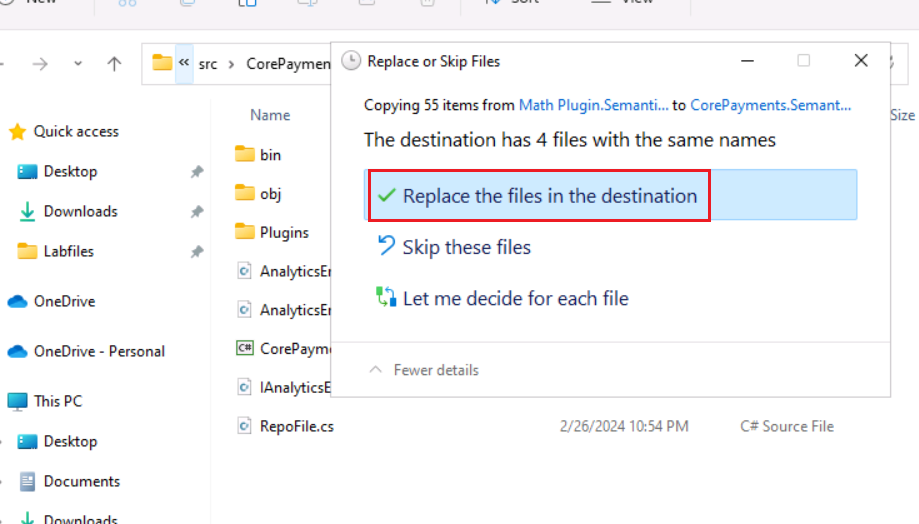
Objective:

1. Improve the SemanticFunction to only use contextual knowledge that you provide it.
2. Improve the handling of numbers.

## Task 1: Create a Math plugin for handling numbers

1. Navigate to C:\Labfiles copy all the files from **C:\LabFiles\Math Plugin.SemanticKernel** folder and replace the contents of folder **C:\Labfiles\Real-time-Payment-Transaction-Processing-at-Scale\src\CorePayments.SematicKernel**





Important: Ensure that the close the **Visual Studio**

1. Close the**Visual Studio 2022** and reopen it**.** Click on the Start button, search for +++**Visual Studio**+++ and select **Visual Studio 2022**.

A screenshot of a computer

Description automatically generated

1. Click on **Open a local folder**.

A screenshot of a computer

Description automatically generated

1. Navigate and select **Real-time-Payment-Transaction-Processing-at-Scale** folder from **C:\LabFiles** and click on the **Select Folder** button.

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

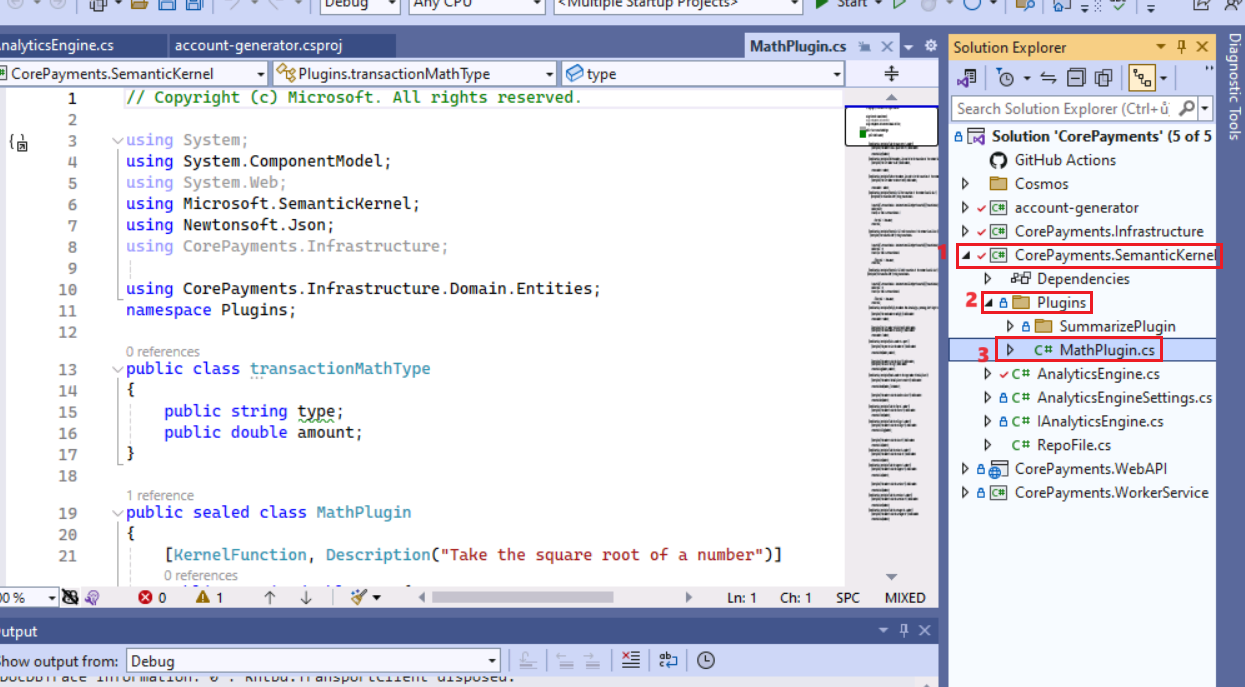
Description automatically generated**

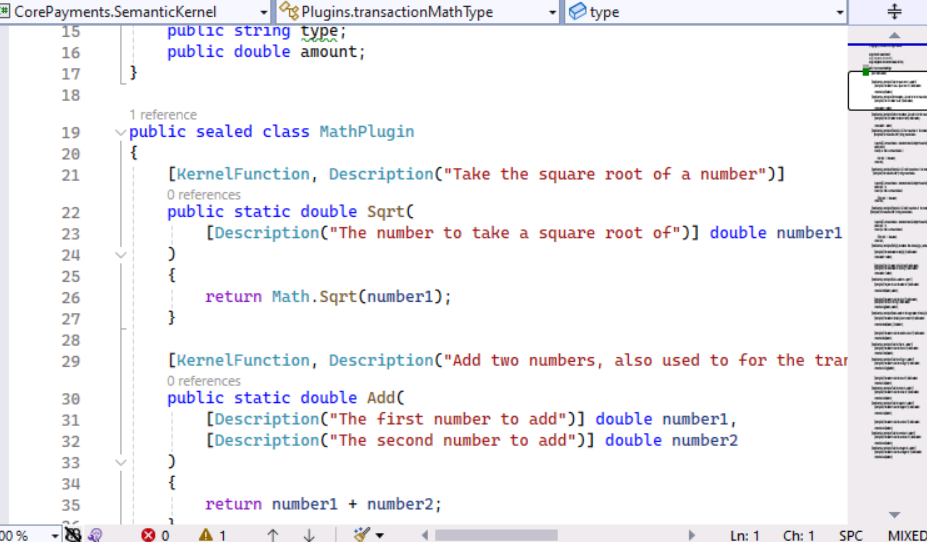
1. In Visual Studio,On **Solution Explorer-Folder view** navigate and click on **CorePayments.sln** file.

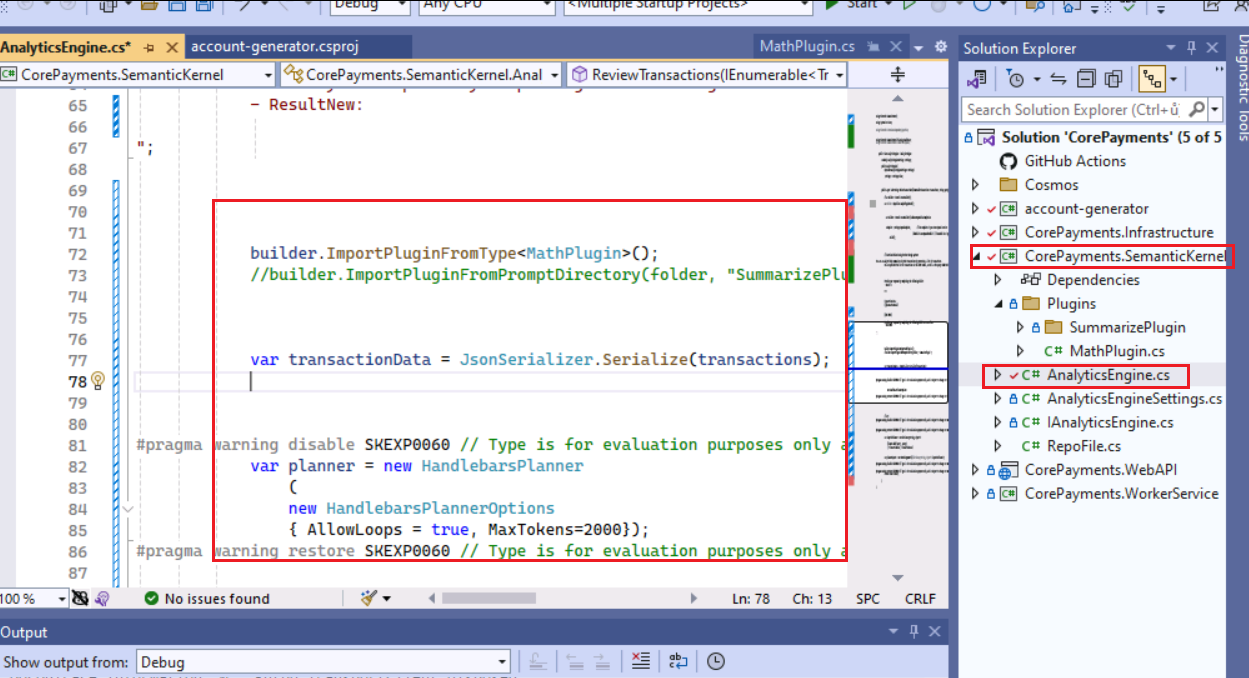
**A screenshot of a computer

Description automatically generated**

1. In the **CorePayments.SemanticKernel** Select the **MathPlugin.cs** file from the **Repository** folder

****

1. Go through the **MathPlugin to understand the KernelFunction and the different operations that are defined in there.**  ****
2. In the **CorePayments.SemanticKernel** Select the **AnalyticsEngine.cs** file from the **Repository** folder

****

1. You are now ready to start debugging the solution locally. To do this, press **F5** or select **Debug > Start Debugging** from the menu.

A screenshot of a computer

Description automatically generated

1. The build gets started and the Swagger app tries to open in the browser.

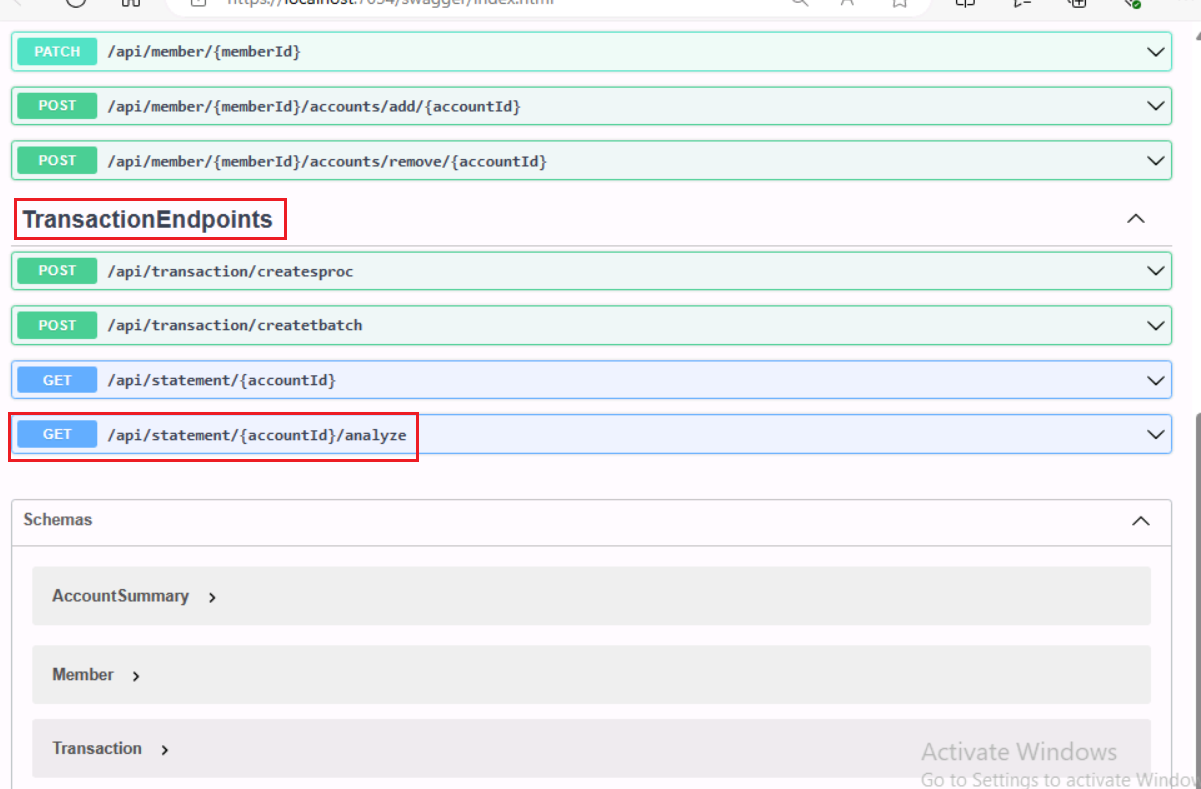
A screenshot of a computer

Description automatically generated

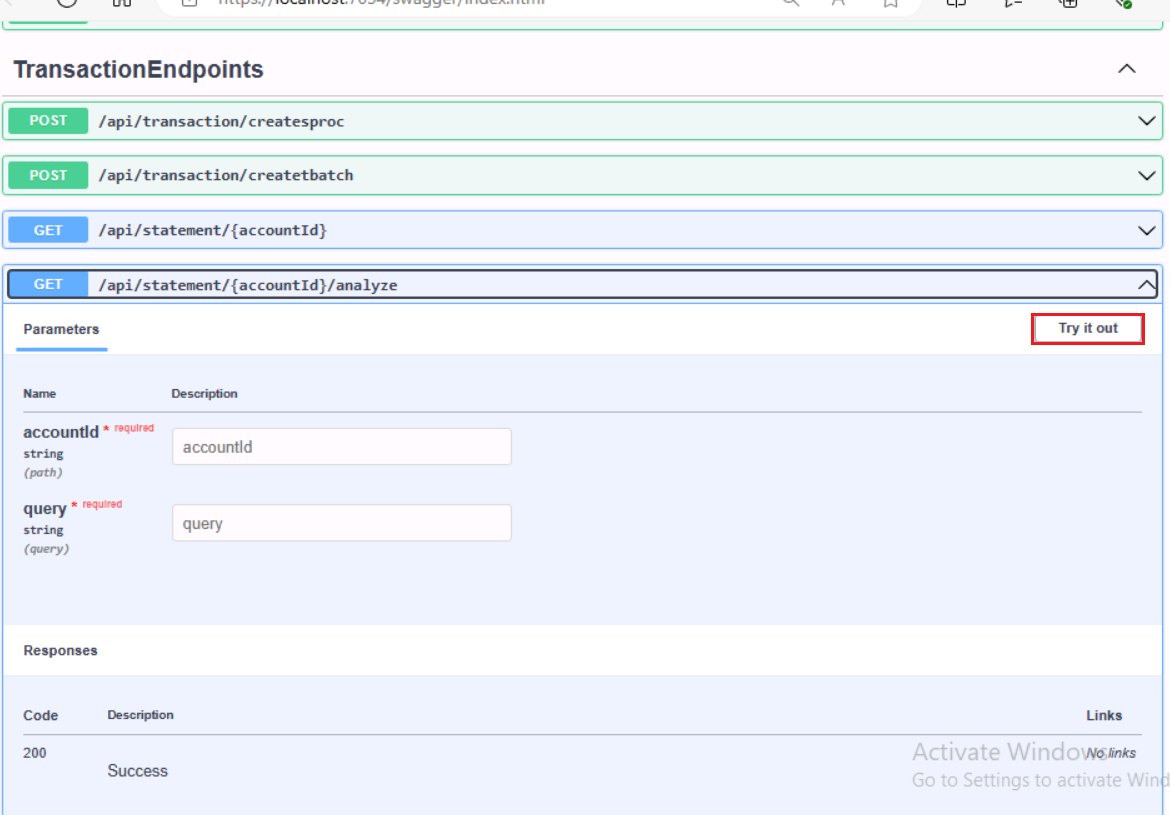
A screenshot of a computer

Description automatically generated

1. In the **Swagger** app, under the **TransactionEndpoints** select **GET /api/statement/{accountId}/analayze**

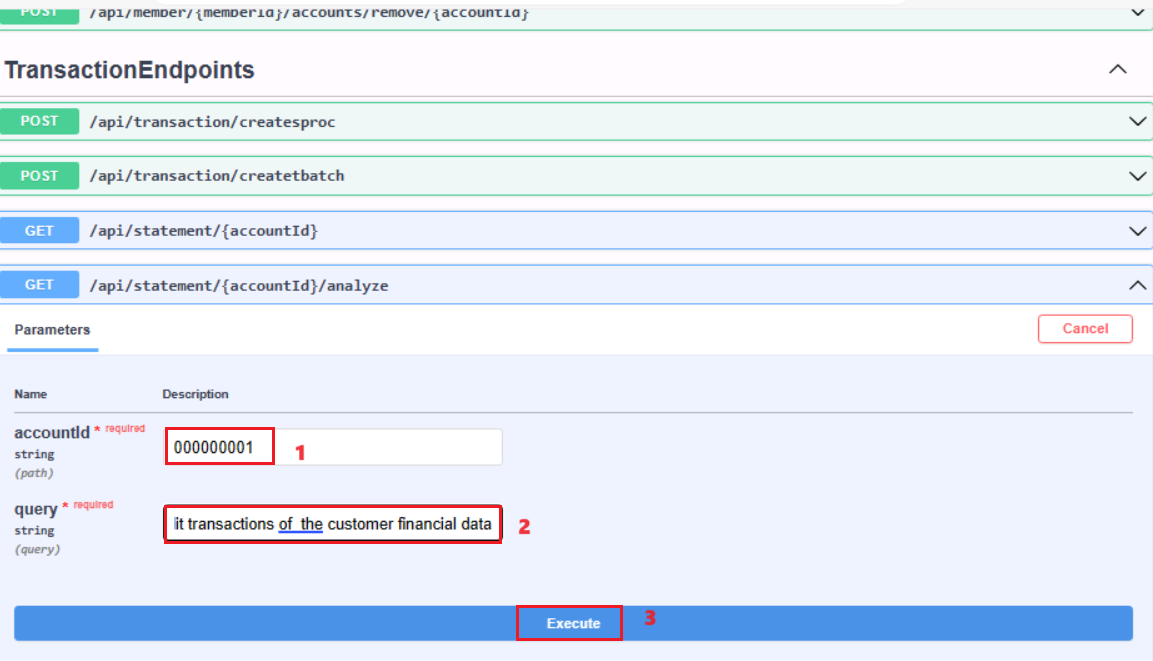


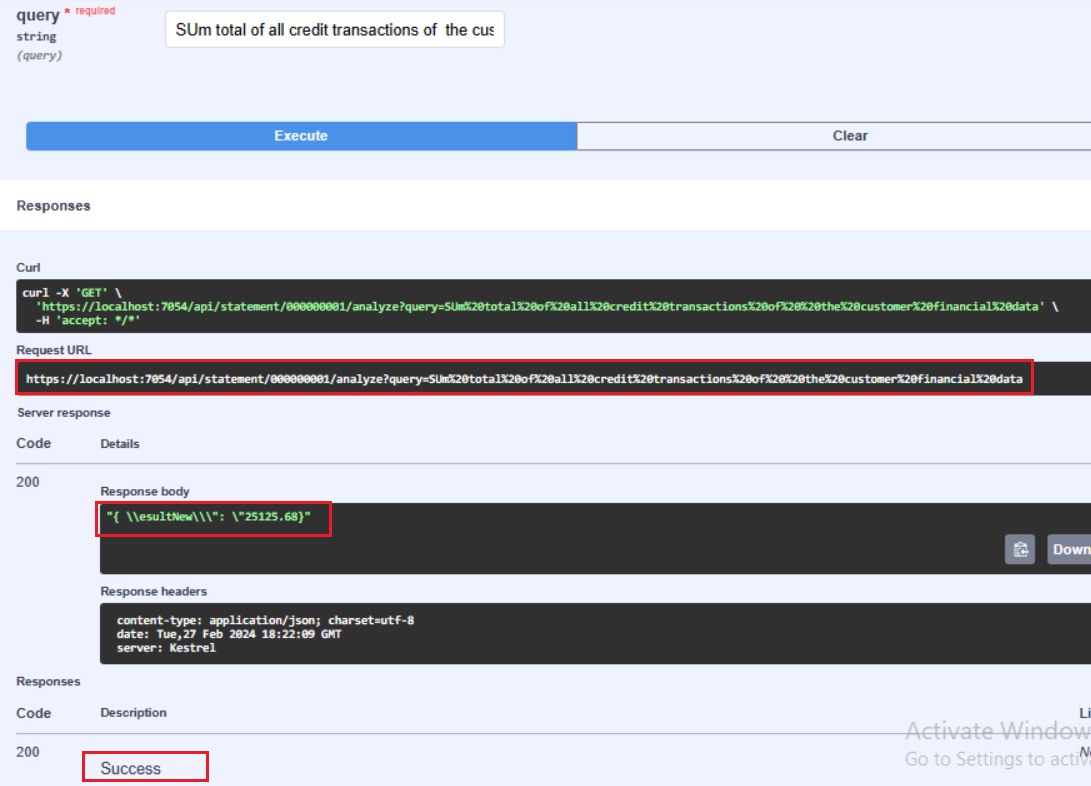
1. Click on the **Try it out** button.



1. Select and copy any accountId from localhost:3000 and enter into the accountid filed.  In the **Query** box, enter the the following statement and click on **Execute** button. Review the response.

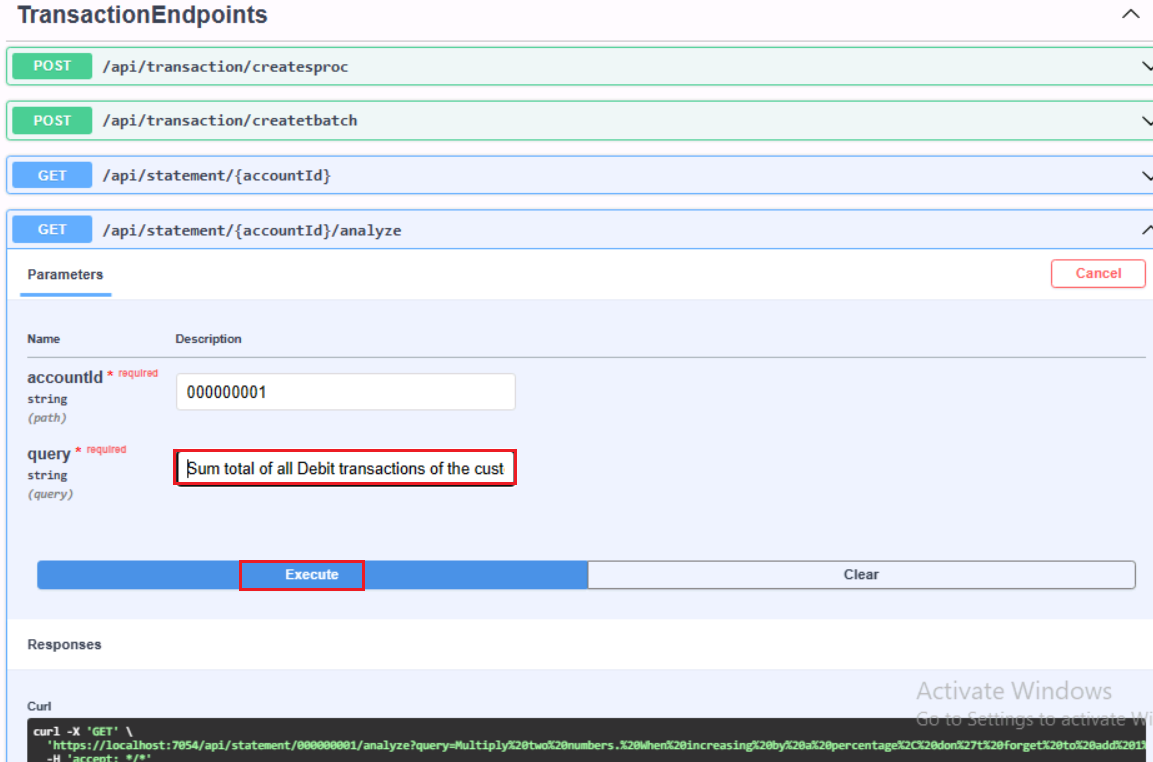
+++ **Sum total of all credit transactions of the customer financial data+++**

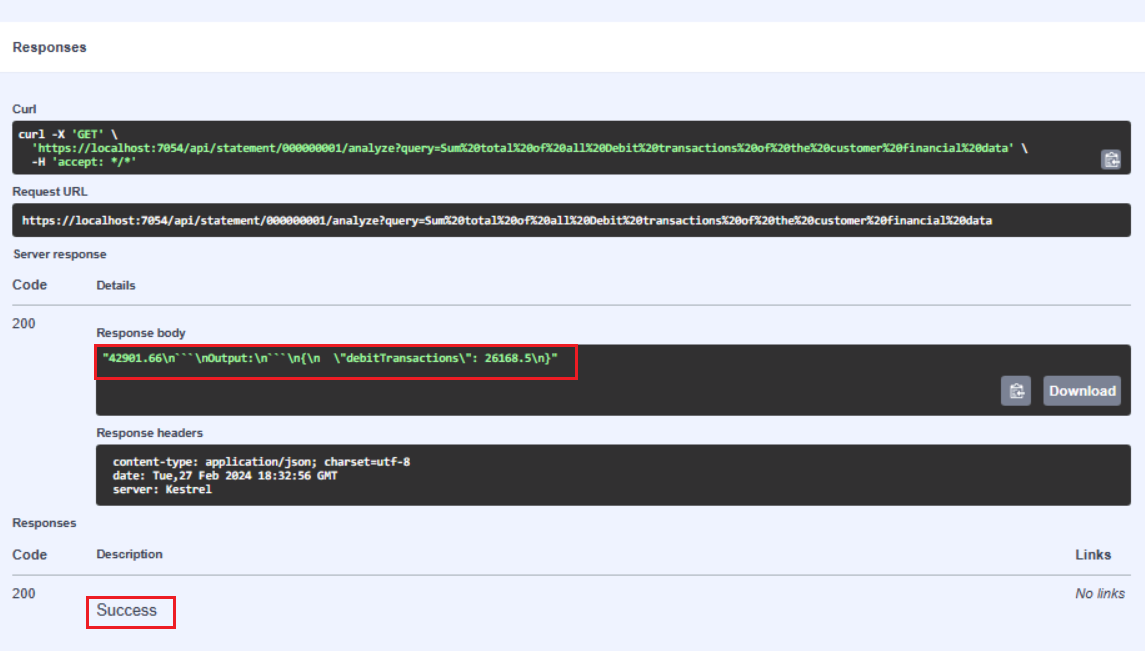




1. In the **Query** box, replace the current text with the following statement and click on **Execute** button. Review the response.

+++ **Sum total of all Debit transactions of the customer financial data+++**





1. In the **Query** box, replace the current text with the following statement and click on **Execute** button. Review the response.

**+++Average the all Debit transactions of the customer financial data+++**

