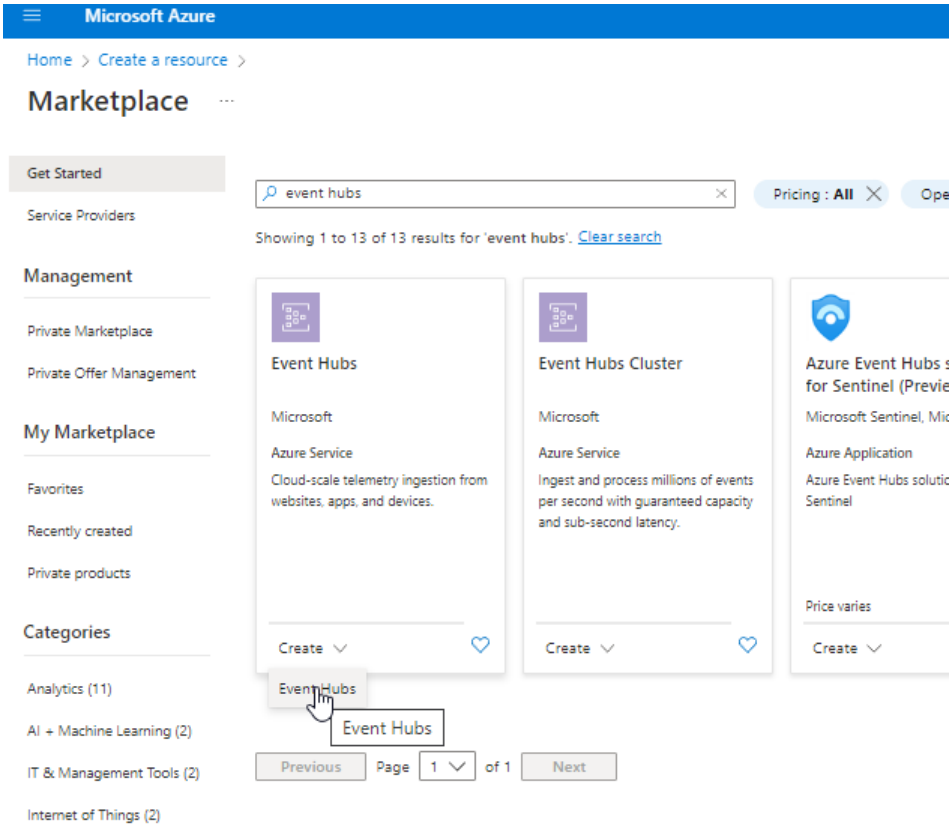
Assignment 3

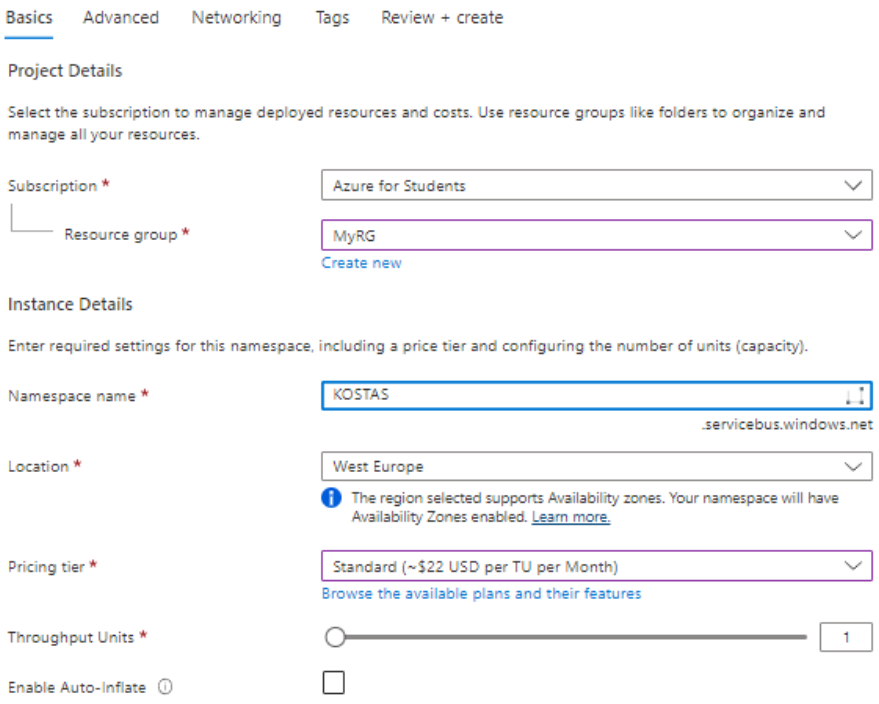
MODERN DATA MANAGEMENT & BUSINESS INTELLIGENCE

Politis Kostas - Vaidomarkakis Panagiotis | Modern Data Management & Business Intelligence
15/01/2023

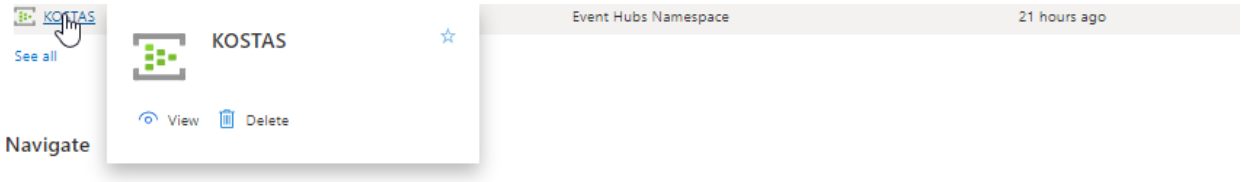
After creating the account, first we add the Event Hubs module from marketplace.

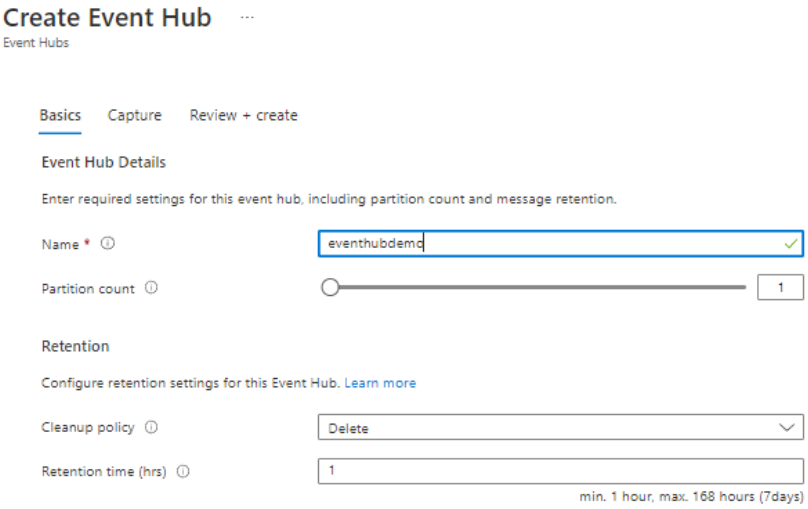


We set up the resource group, namespace of the hub etc. and proceed by creating it.

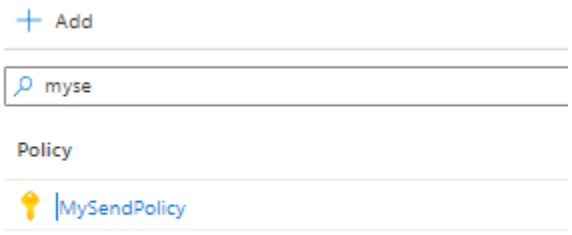


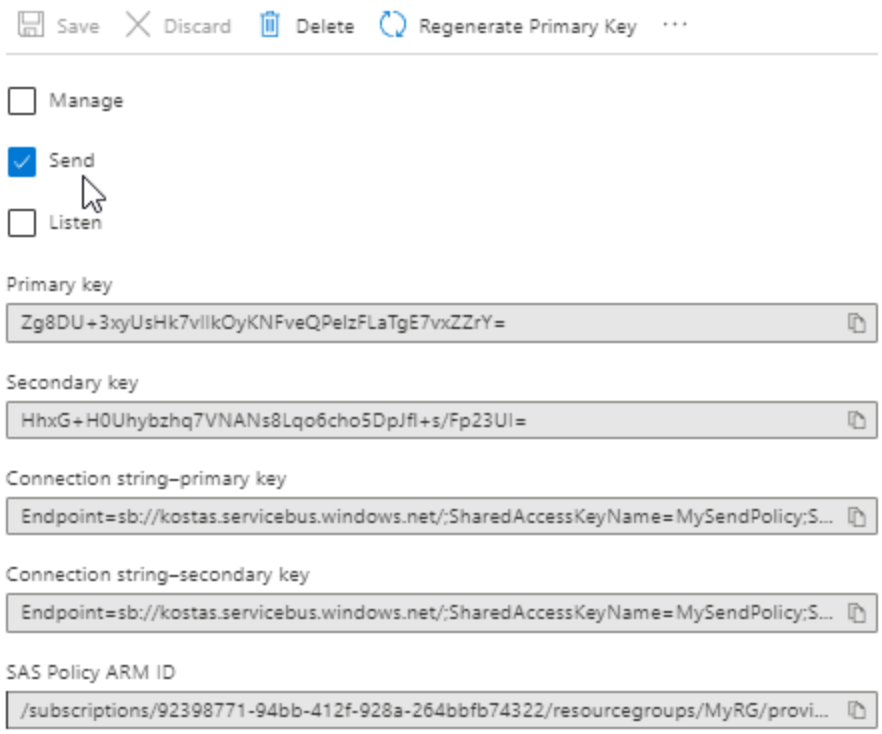
We then select the hub namespace and create a new event hub.



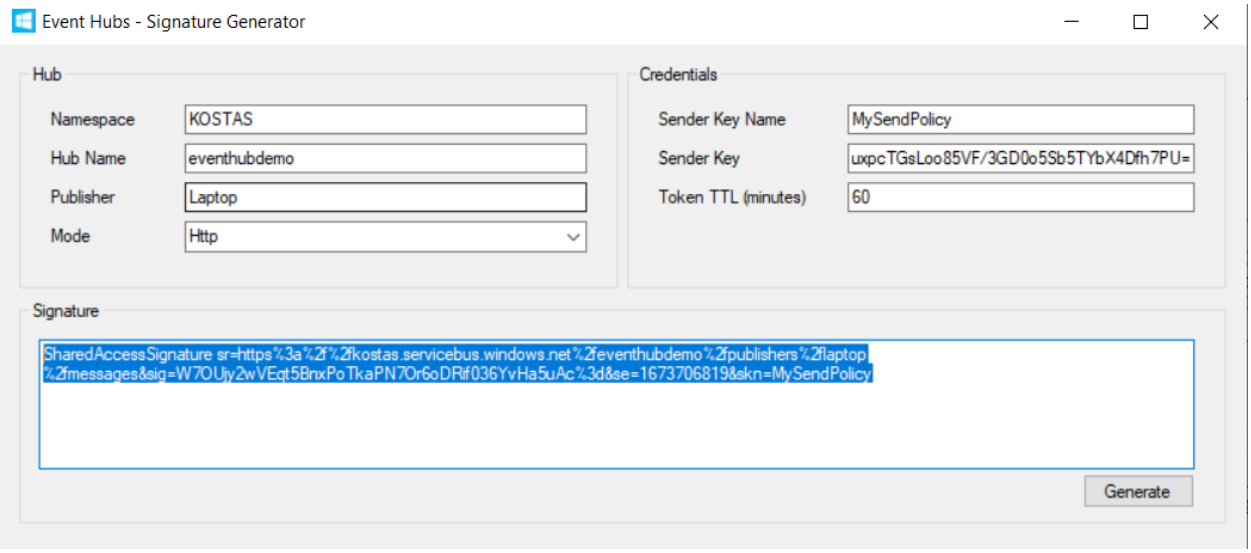


On the newly created event hub, we create a sent policy.





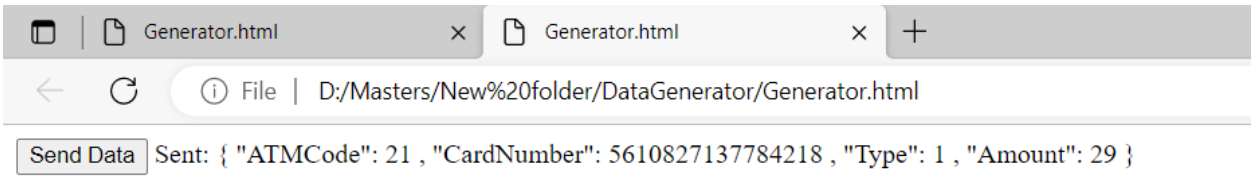
Using the primary key produced by the sent policy, we generate a signature using the Event Hub-Signature Regenerate.



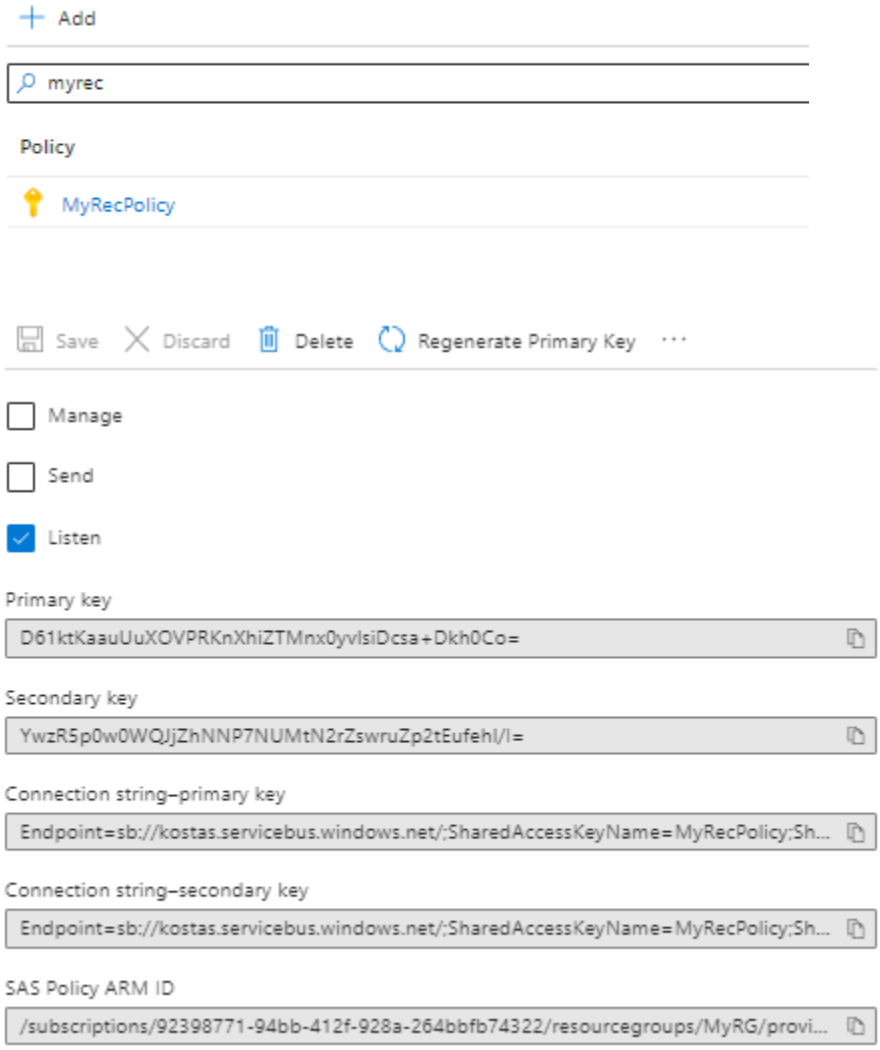
We then fill in the produced signature on the Generator file as well as the service namespace, the hub name and the device name.

```
<html>
<head>
    <script src="js/lodash.js"></script>
</head>
<body>
<input type="button" value="Send Data" onclick="sendDummyData()" />
<div id="status" style="display: inline-block;"></div>
<script type="text/javascript">
function sendDummyData() {

  /**************/
  /*** CONFIG ***/
  /**************/

  //Use the signature generator: https://github.com/sandrinodimattia/RedDog/releases
    var sas = "SharedAccessSignature sr=https%3a%2f%2fkostas.servicebus.windows.net%2feventhubdemo%2fpublishers%2fl
  var serviceNamespace = "mscba-aueb";
  var hubName = "eventhubdemo";
  var deviceName = "Laptop";

  /*****************/
  /*** GENERATOR ***/
  /*****************/
  var atms = [{"atm_code":1,"area_code":20},{"atm_code":2,"area_code":17},{"atm_code":3,"area_code":18},{"atm_code"

  var customers = [{"card_number":5446210381593272,"first_name":"Eugene","last_name":"Mason","age":67,"gender":"Mal

  var jsonData;

  var RND_Customer    = 0;
  var RND_ATMCode     = 0;
  var RND_CardNumber  = 0;
  var RND_Type        = 0;
  var RND_Amount      = 0;

  setInterval(function(){

    RND_Customer = _.random(0,19);

    if( _.random(0,1) == 1 ) {
```

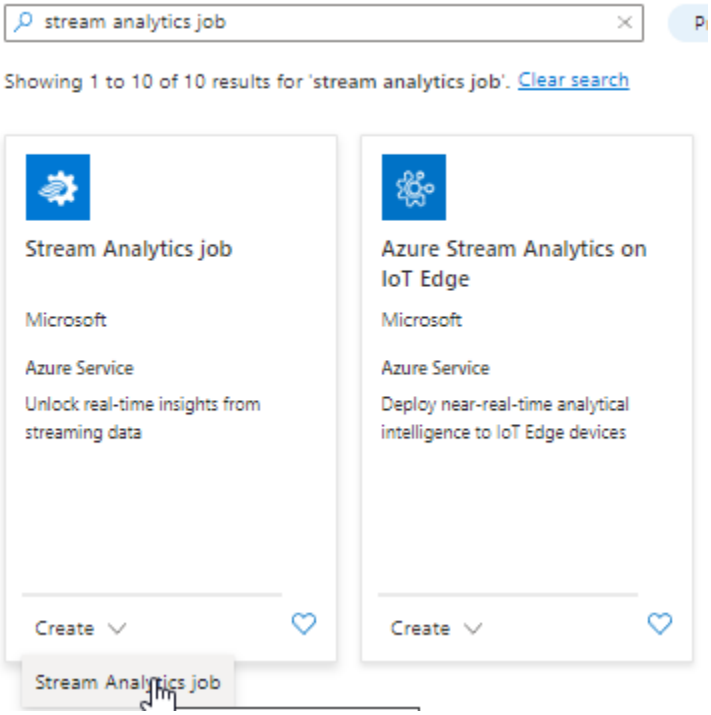
We then open the Generator file and by clicking on Send Date, the process of streaming data on our hub begins.



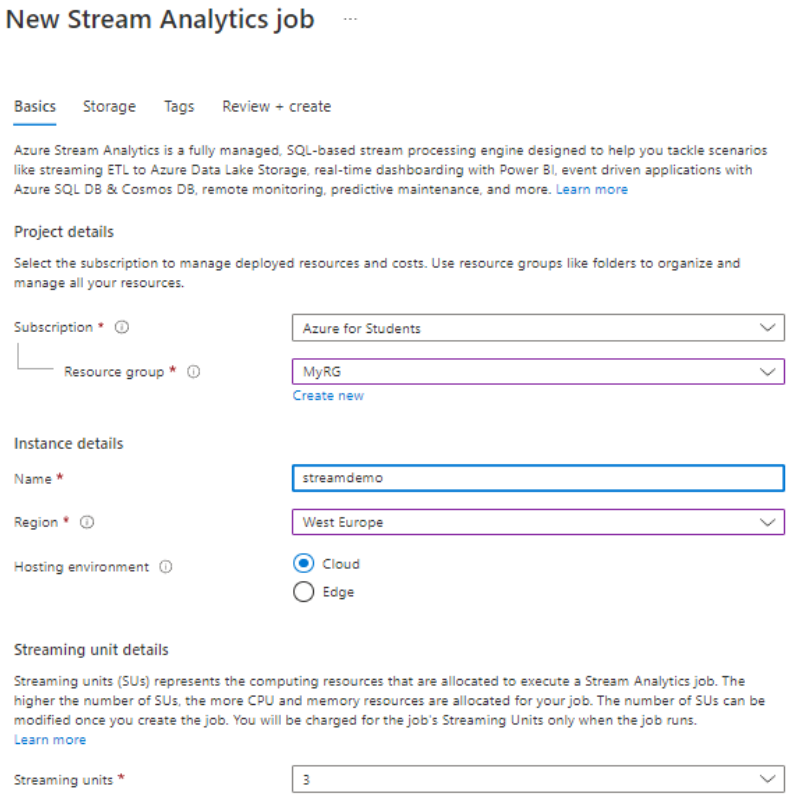
We then proceed by creating a SAS Policy with the Listen configuration.



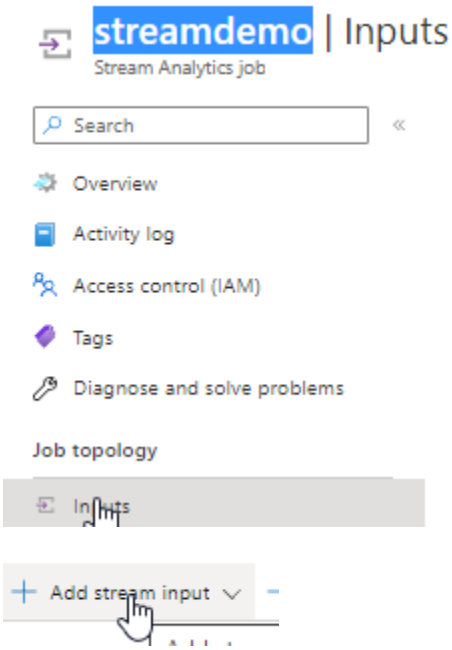
We then create a stream analytics job which we find from the marketplace.



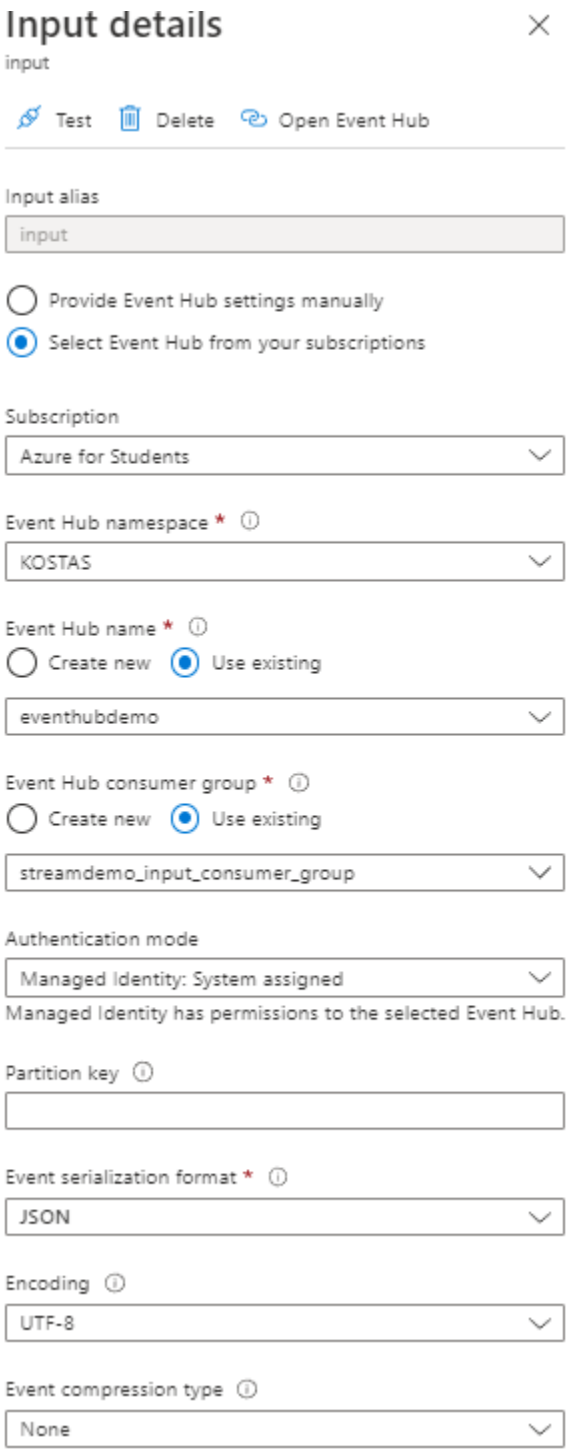
After making all the required configuration, we click on create.

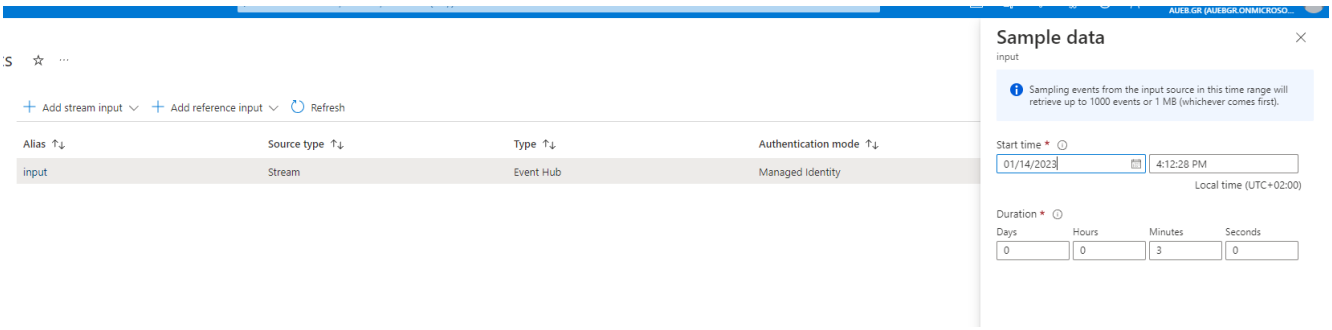


On the Stream Analytics Job that we created, on the Inputs section we select to add a stream input.

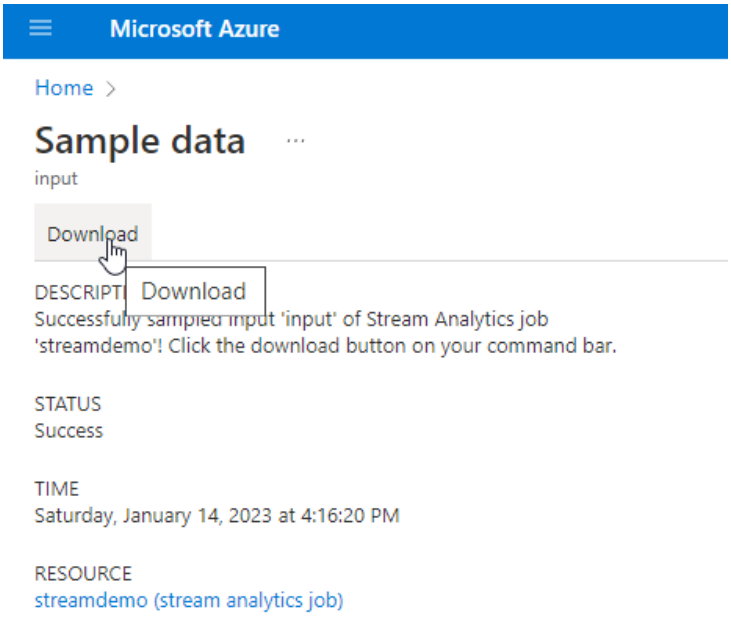


We configure the new input and we select to receive sample data every 3 minutes.

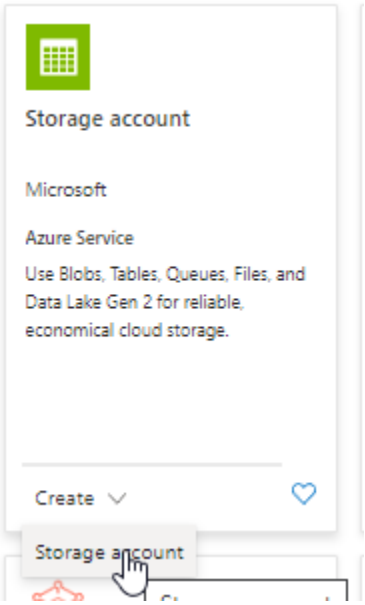




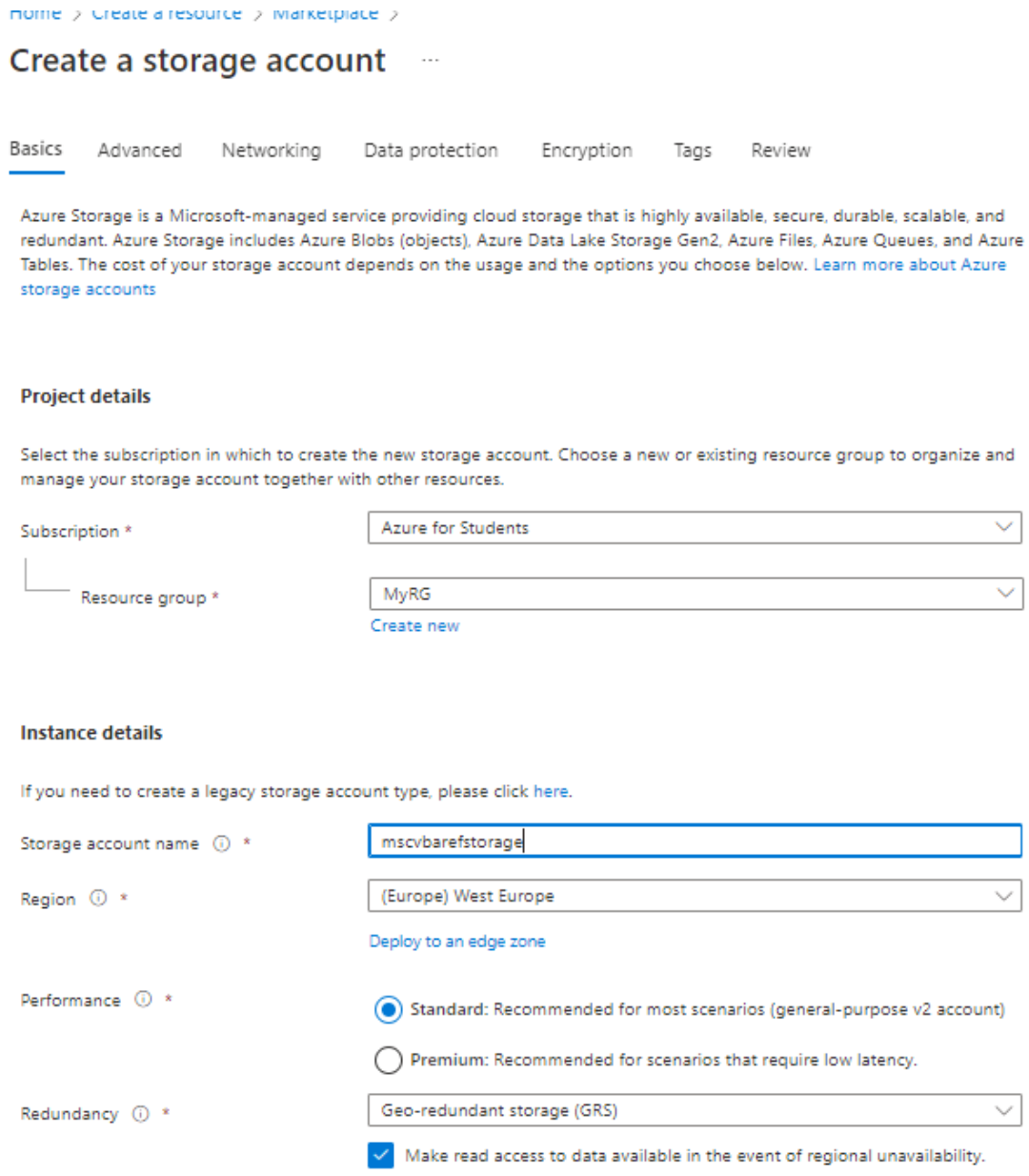
We then download the Sample data, exporting them on a JSON file.



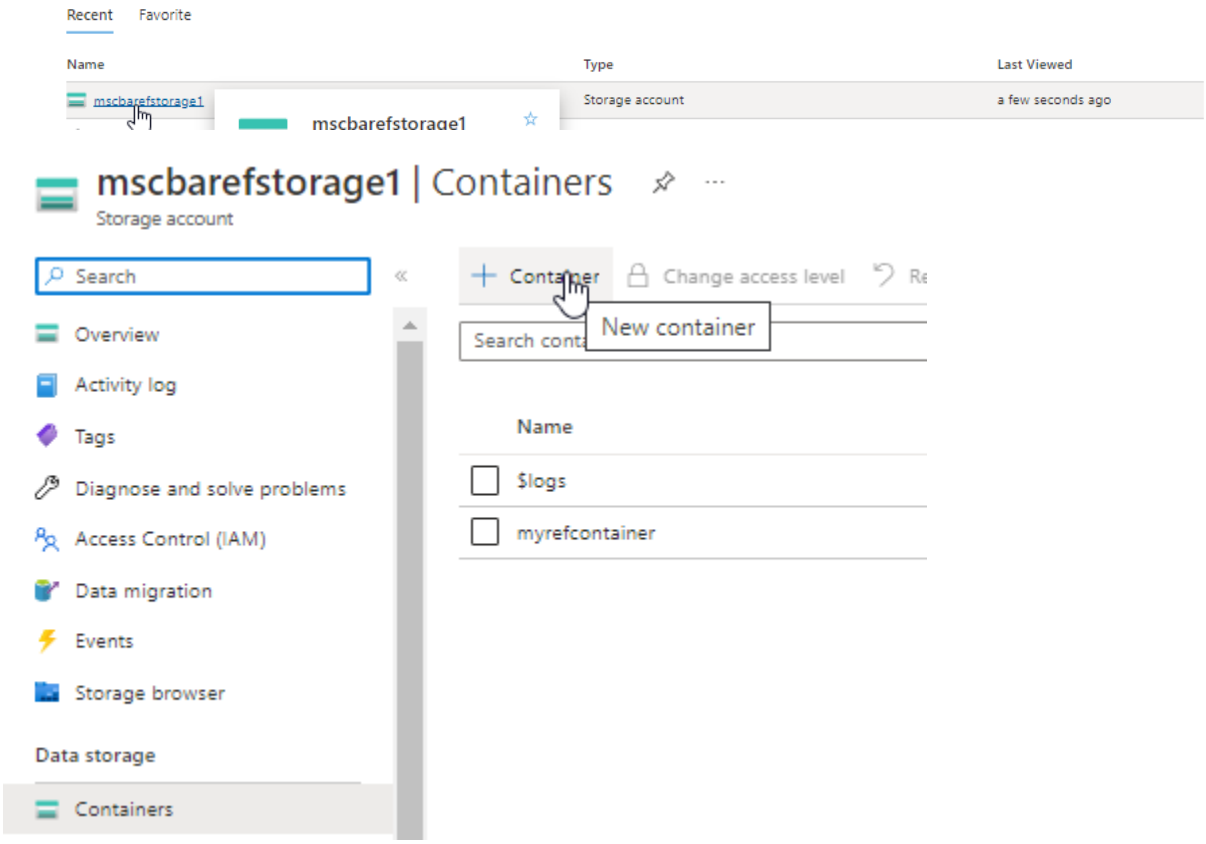
We then create a new Storage Account from the marketplace.



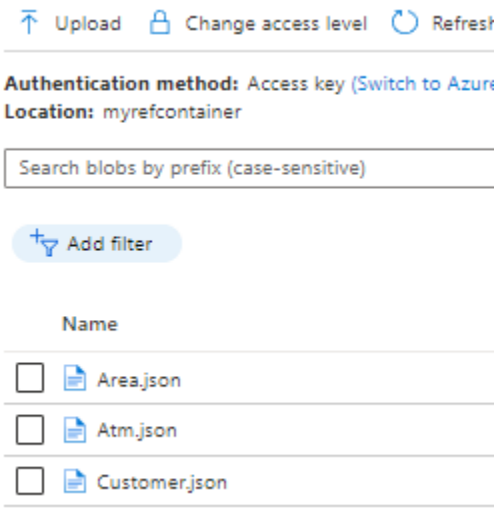
After the appropriate configuration, we click on create.



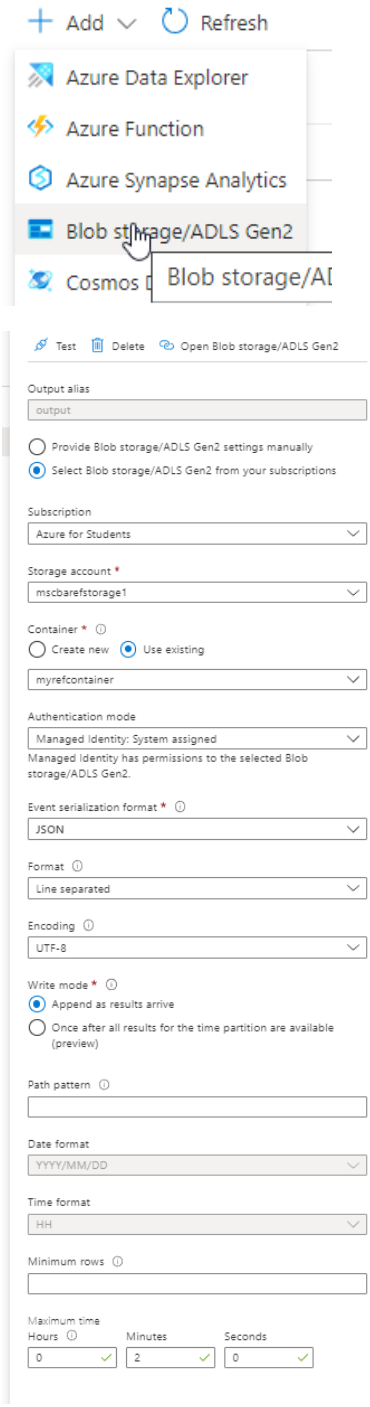
We then select the newly created storage account and on the Container section we click on Create New Container.



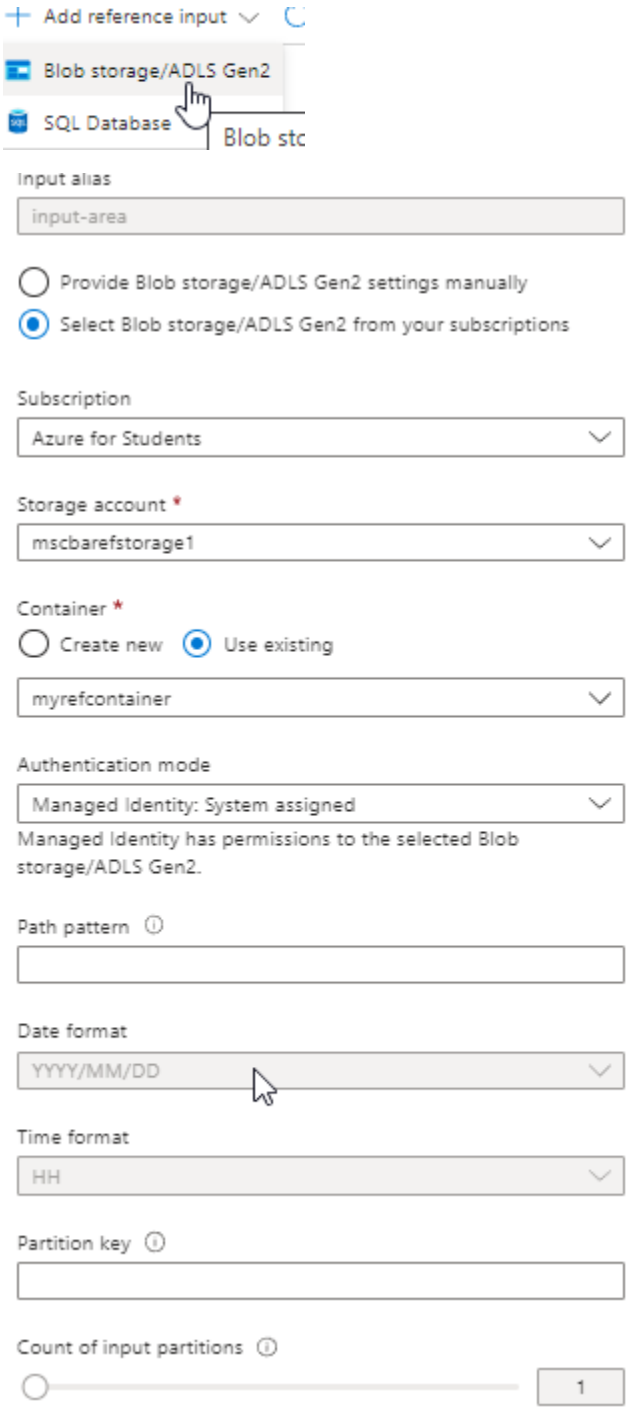
After the container is created, we upload the three JSON files.



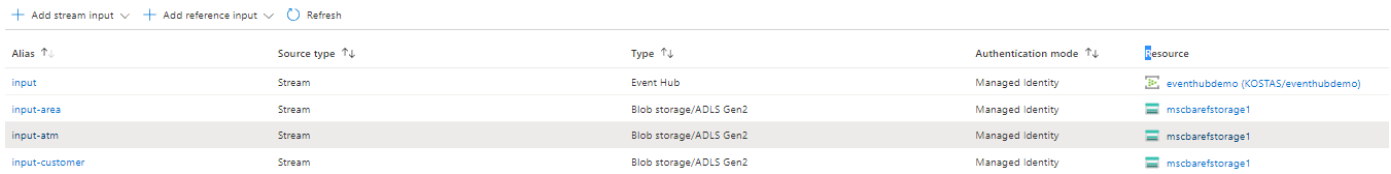
Back on the Stream Analytics Job, on the output section, we add a Blob storage and make the appropriate configuration.



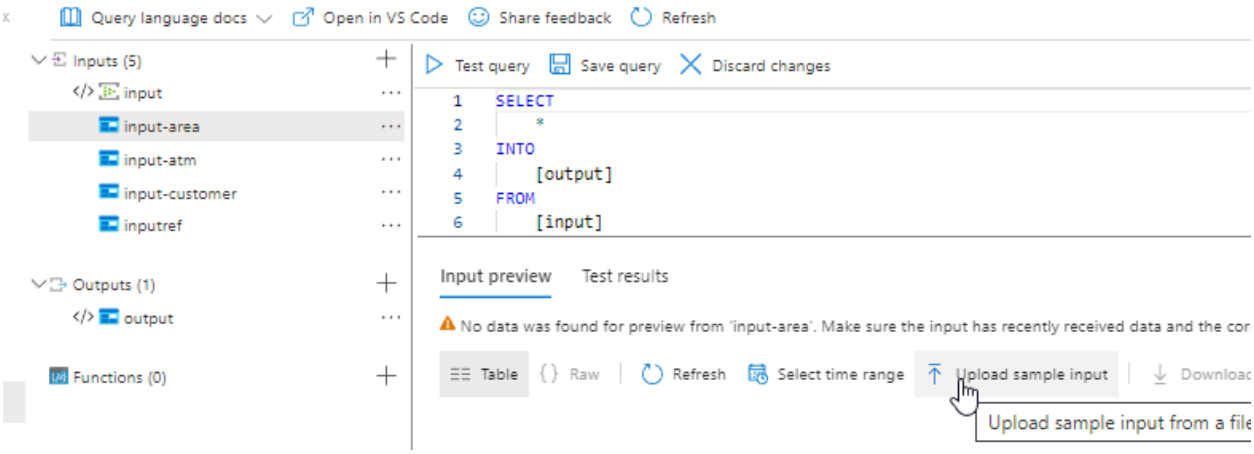
On the input section of the Stream Analytic Job, we click on Add reference input/Blob storage and we make the appropriate configuration.



We create three such inputs for Area, Customer and ATM JSON files.

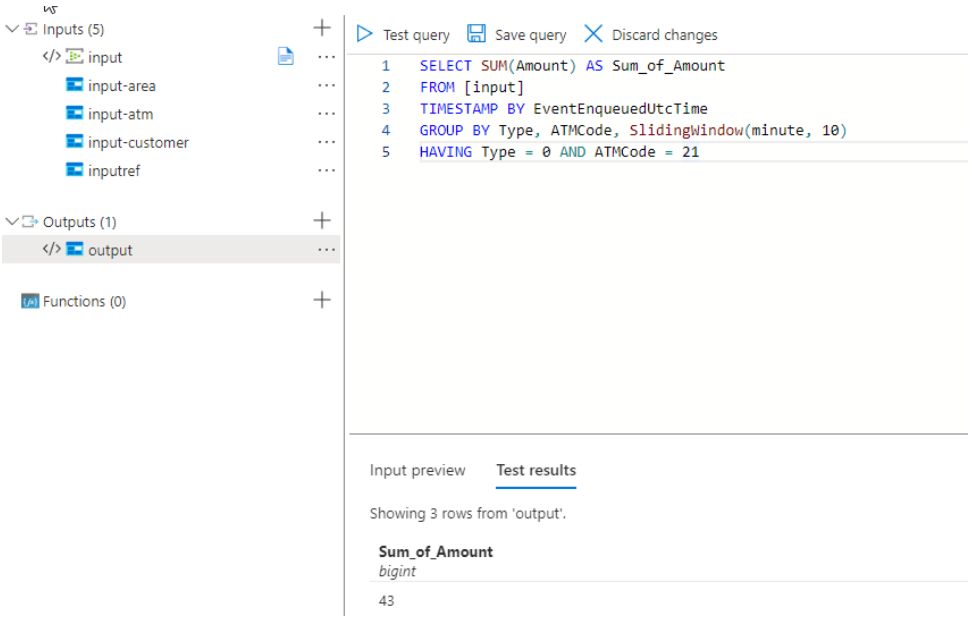


On the Query section of the Stream Analytic Job, we upload the respectively JSON files for area, atm and customer.

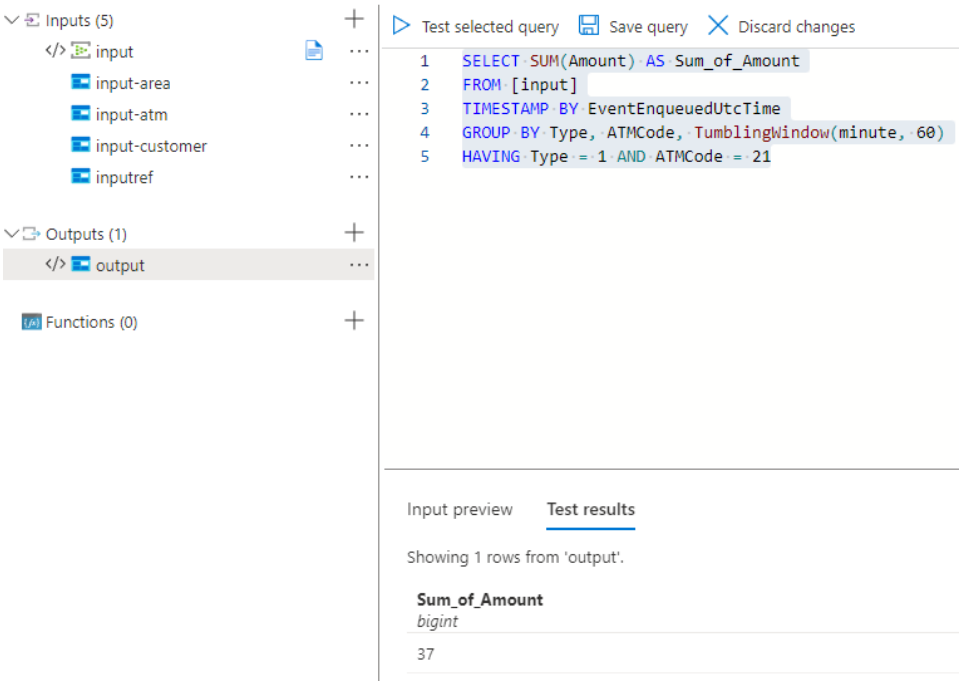


We then proceed by writing the appropriate queries:

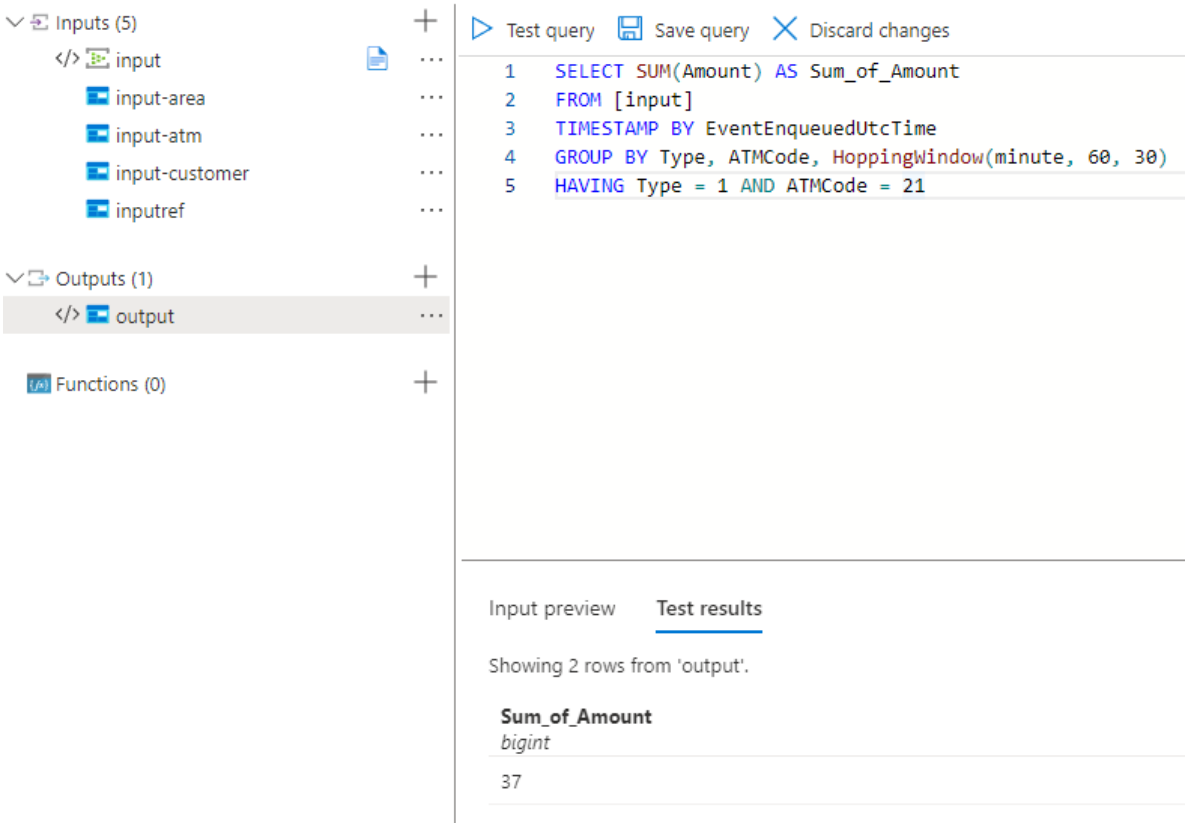
Query 1: Show the total "Amount" of "Type = 0" transactions at "ATM Code = 21" of the last 10 minutes. Repeat as new events keep flowing in (use a sliding window).



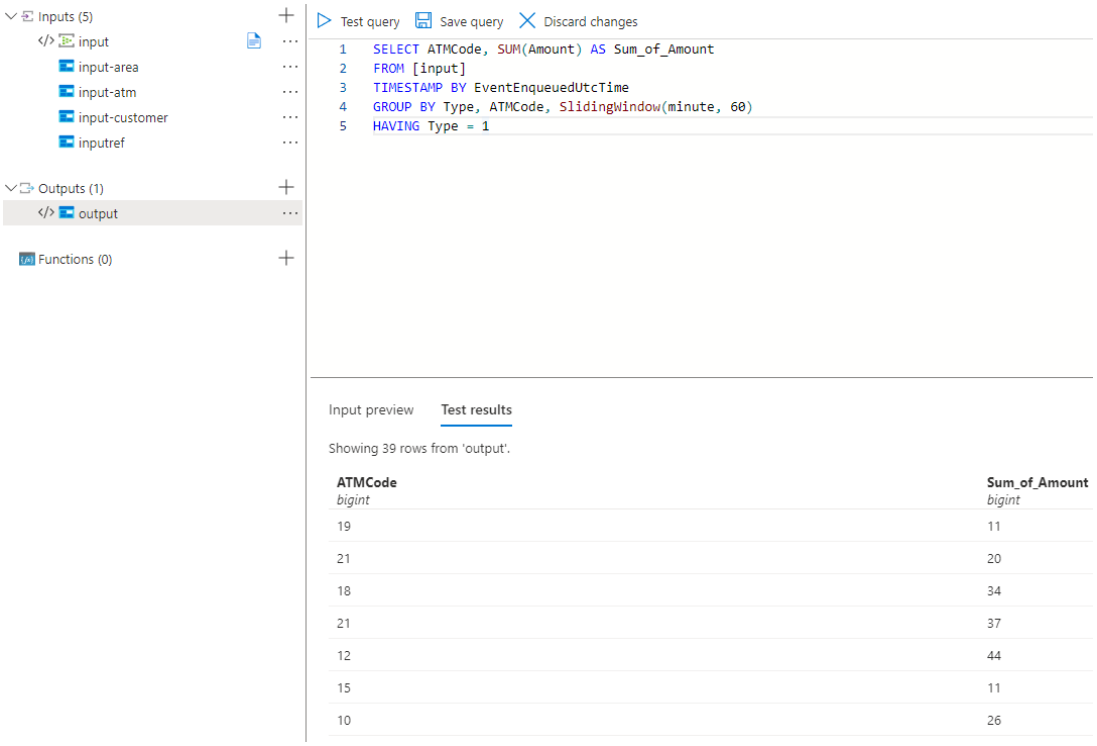
Query 2: Show the total "Amount" of "Type = 1" transactions at "ATM Code = 21" of the last hour. Repeat once every hour (use a tumbling window).



Query 3: Show the total "Amount" of "Type = 1" transactions at "ATM Code = 21" of the last hour. Repeat once every 30 minutes (use a hopping window).

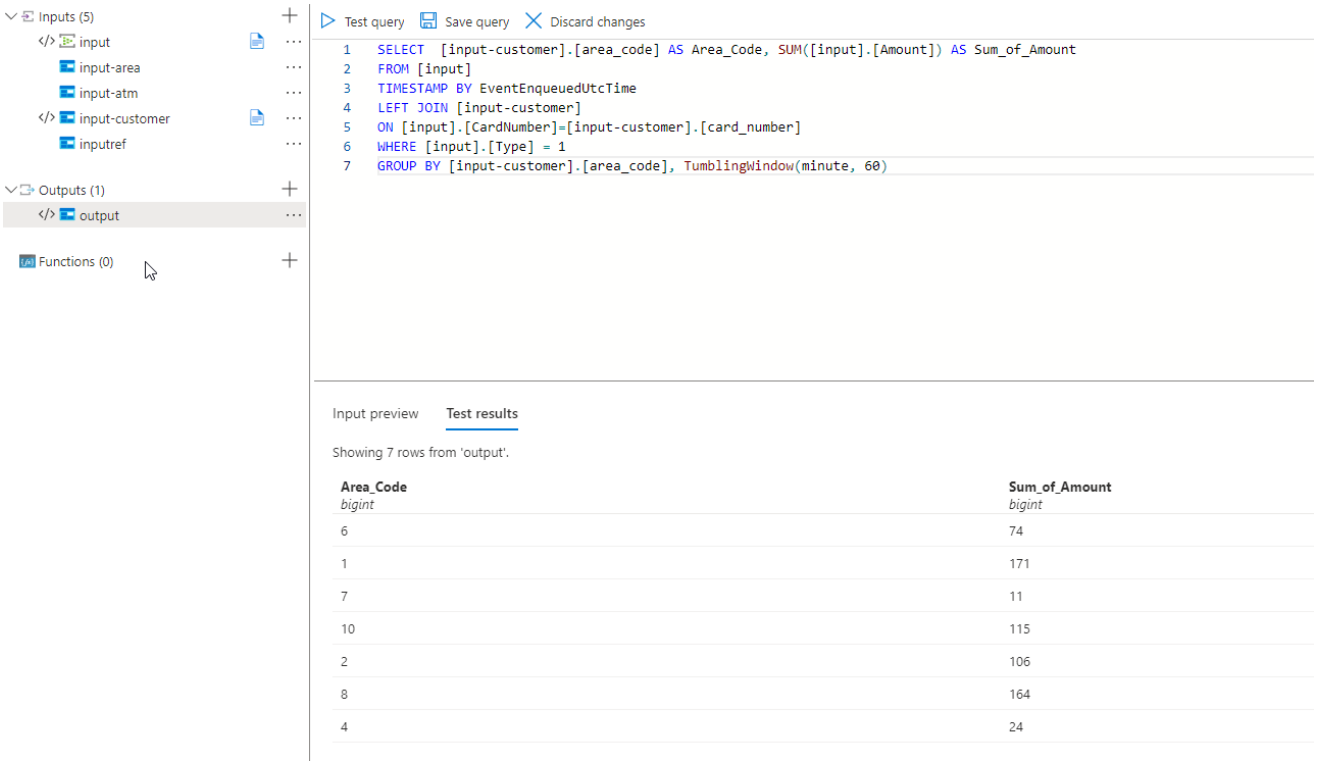


Query 4: Show the total “Amount” of “Type = 1” transactions per “ATM Code” of the last one hour (use a sliding window).



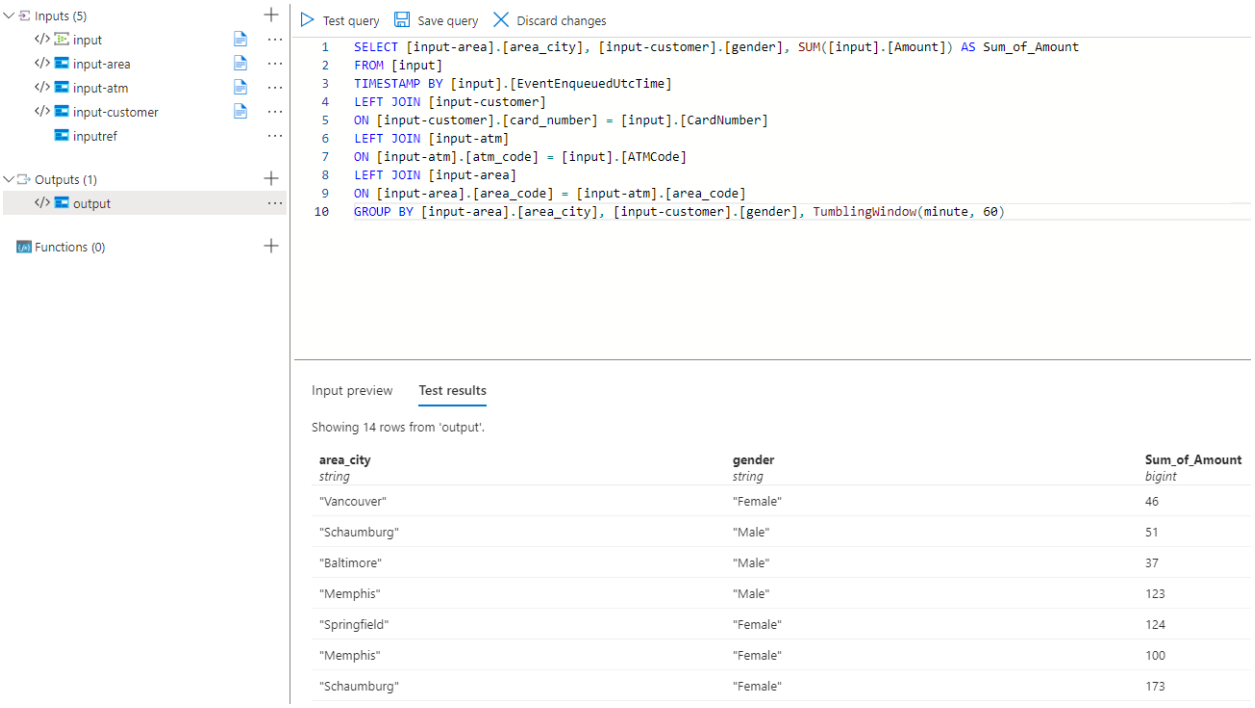
ATMCode *bigint*	Sum_of_Amount *bigint*
19	11
21	20
18	34
21	37
12	44
15	11
10	26

Query 5: Show the total “Amount” of “Type = 1” transactions per “Area Code” of the last hour. Repeat once every hour (use a tumbling window).

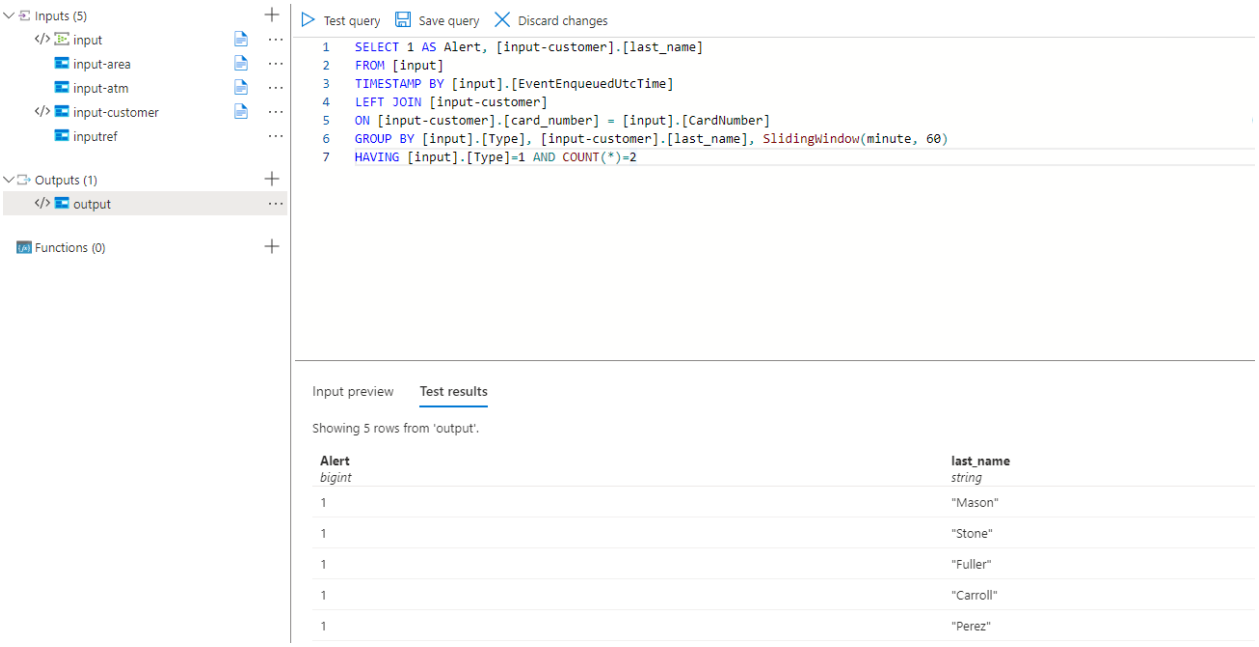


Area_Code *bigint*	Sum_of_Amount *bigint*
6	74
1	171
7	11
10	115
2	106
8	164
4	24

Query 6: Show the total “Amount” per ATM’s “City” and Customer’s “Gender” of the last hour. Repeat once every hour (use a tumbling window).



Query 7: Alert (Do a simple SELECT “1”) if a customer has performed two transactions of “Type = 1” in a window of an hour (use a sliding window).



Query 8: Alert (Do a simple SELECT “1”) if the “Area Code” of the ATM of the transaction is not the same as the “Area Code” of the “Card Number” (Customer’s Area Code) - (use a sliding window).