Modern Data Management

&

Business Intelligence

Assignment 3

Jan 2022

Student

Maria Skoli (p2822131)

Class

Part Time

Table of Contents

Introduction

In this report we are going to use Azure Stream Analytics to process a data stream of ATM transactions and answer possible tasks we would like to be executed in real time.

The first part of this report is about setting up an Event Hub and a Storage account where our results are going to be stored. Moreover, we are going to feed our stream with the Reference data we already have. Specifically, we have as reference data 3 json files. The first one "Area.json" contains geographical information, the second one "ATM.json" contains information about the ATM machines (e.g., the Atm code) and the last one, "Customer.json" contains information about each customer (e.g., his card number).

In the second part of this report, we are going to create an input and an output for our data and to execute queries that answers business questions. We will present the relative outputs as well.

Part 1 – Steps to Set up An Event Hub

Create a Resource Group

First, we should create a resource group which is a logical collection of Azure resources. We named the resource group as: *"**MyEH**"* and as region we selected the "(Europe) UK South".

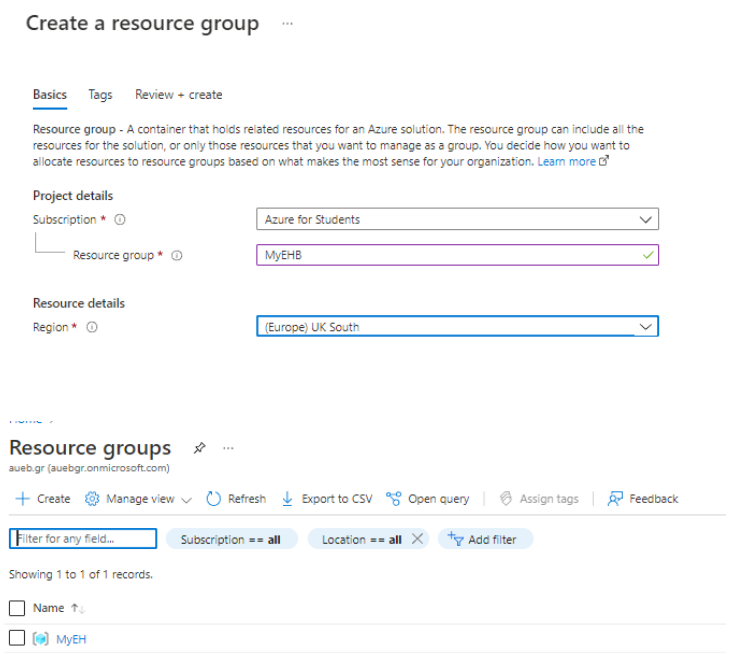

Figure 1: Creating a resource group

Create a Namespace

Before we create an event hub, we should first create a Namespace. An Event Hubs namespace provides a unique scoping container in which we will create the Event Hub. As inputs in the requested fields, we inserted the name of the resource group that we have created before, we enter the name ***"ATMtrans"*** for the namespace, and we select the location of the namespace as "UK South". As for the pricing tier we selected the Standard one.

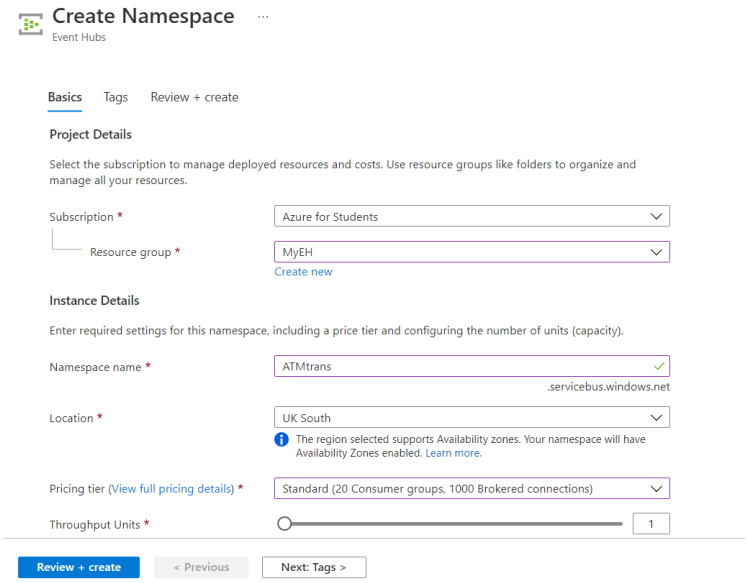

Figure 2: Creating a Namespace

We can see below that the deployment of the namespace *"ATMtrans"* we have created was successful.

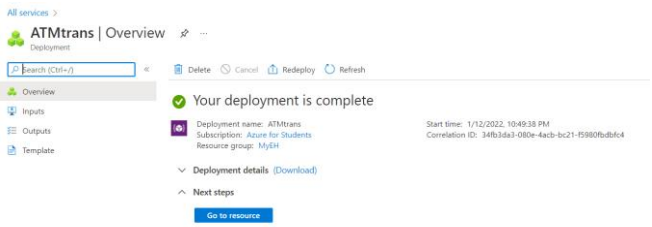

Figure 3: Successful Deployment of the Namespace

Create an Event Hub

The next step was to enter the Namespace we have created and create a new Event Hub. We named the event hub as: myeventhub.

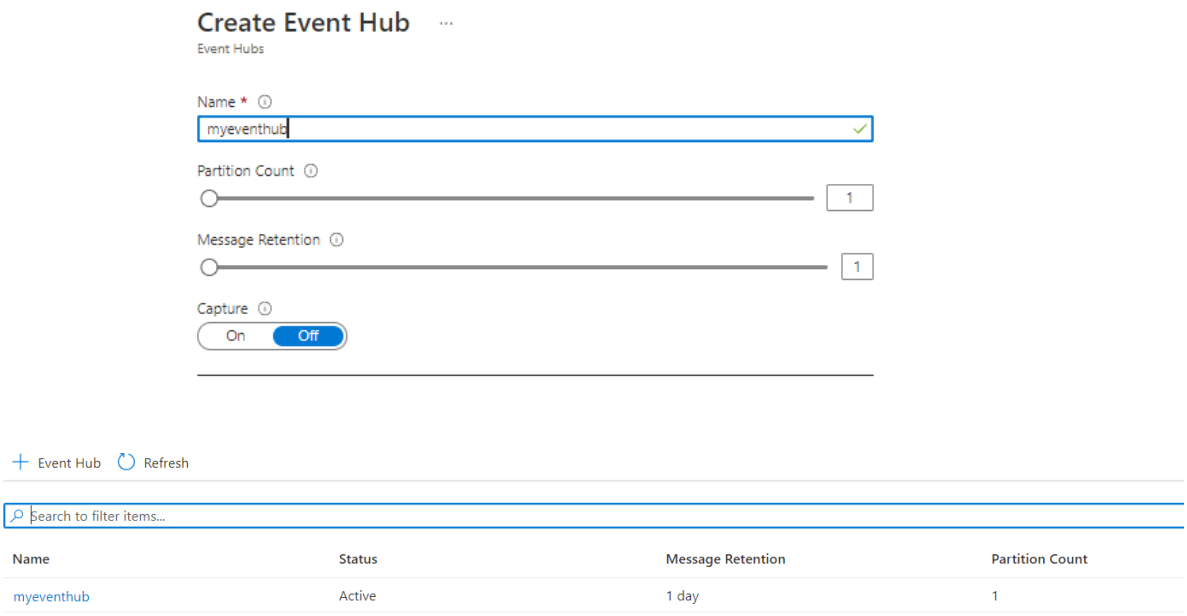

Figure 4: Creation of an Event Hub

Create Sending and Recording Policy

Then we enter the Event hub, and we press the "Shared access policies" to create a Sending and a Recording Policy.

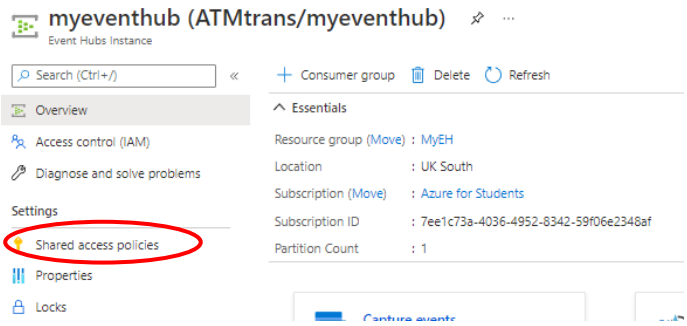

Figure 5: Creation of Sending and Recording Policy

MySendPolicy is used to send data into the Azure stream and the MyRecPolicy is used to listen the data that was send.

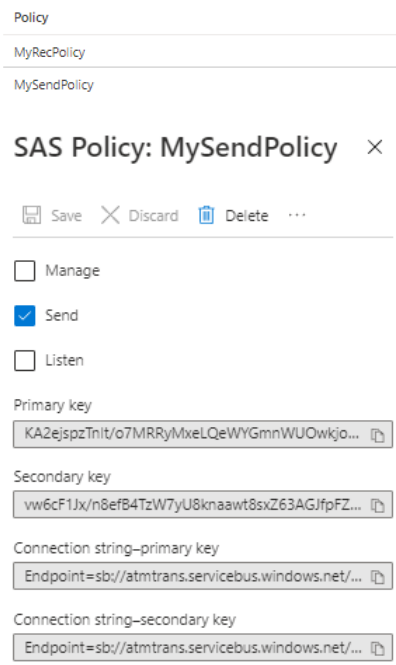

Figure 6: Creation of Sending Policy

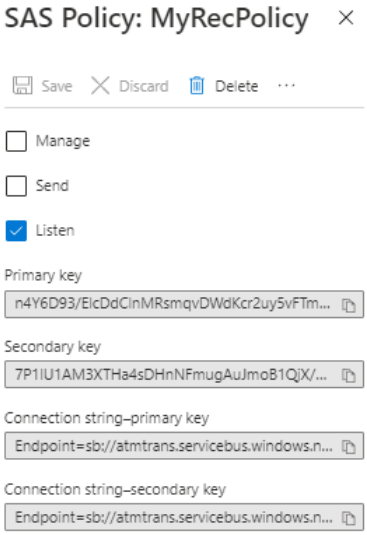

Figure 7: Creation of Recording Policy

Event Hub Signature Generator

Then we are going to generate a signature from the signature generator to update our Html file. “Generator.html”. We fill the blank fields with the names of Namespace, Hub name and Sender key we created before. Then we press Generate and then the Signature link will appear.

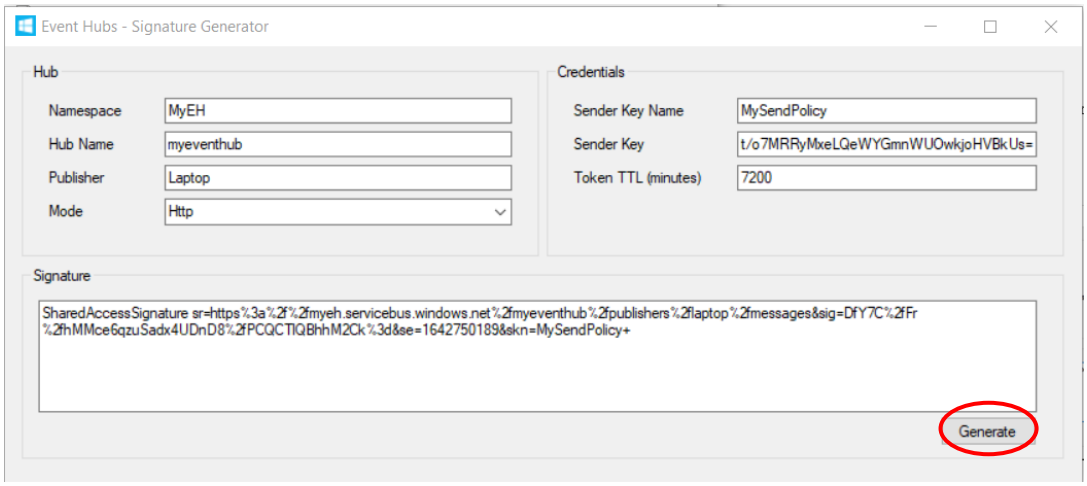

Figure 8: Signature Generator inputs

We opened the Generator.html file (with Notepad++) and we did the following transformations. First, we changed the link in the "*sas variable*" with the link that was generated from the Generator app above (Table 5). Then we changed the "*service Namespace*" to "*ATMtrans*" and the Hub Name to "myeventhub" to corresponds with the resources we have created in Azure.

```
<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%2fatmtrans.servicebus.windows.net%2fmyeventhub%2fpublishers%2flaptop%2fmessages&s
  var serviceNamespace = "ATMtrans";
  var hubName = "myeventhub";
  var deviceName = "Laptop";
```

Figure 9: Making the appropriate changes in Generator html file

Then we open the above file "Generator.html" with Chrome and and we press Send Data. We can see the system started to send data to the event hub.

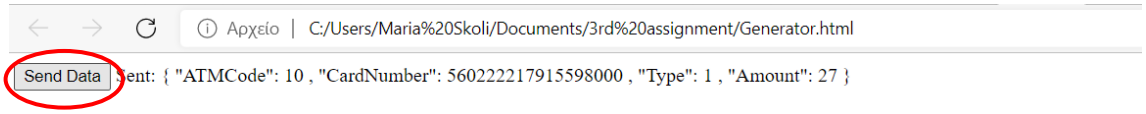

Figure 10: Send data to the source

Create a Storage Account

Then we created a storage account where the results of our analysis will be saved.

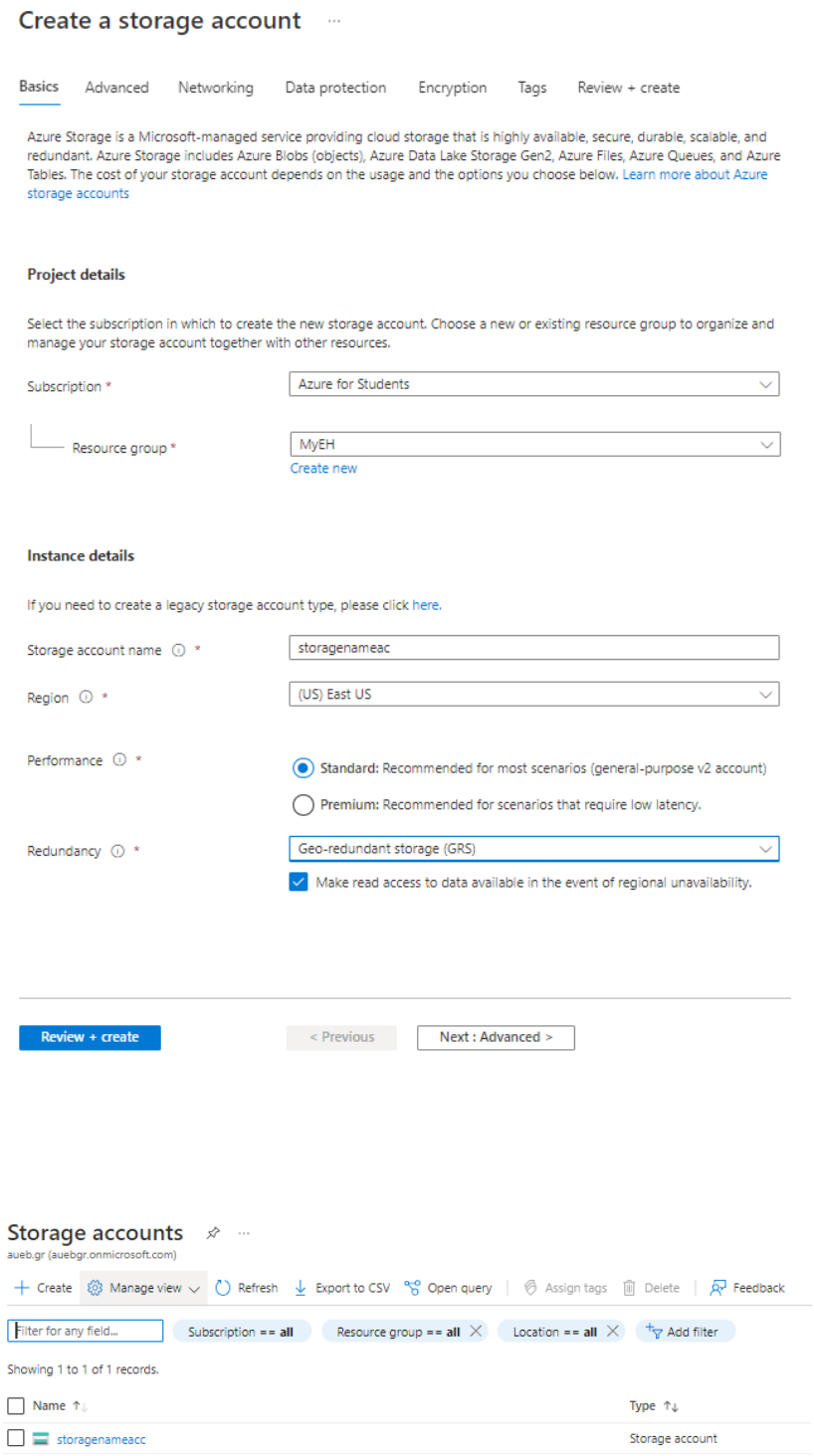

Figure 11: Creating a Storage Account

Create a Container into Storage Account

Then we enter the Storage account we created, and we add a new container. We named the container as “mycontainer”. This will be the place where the outputs of our queries will be saved. As public access level we select the private one.

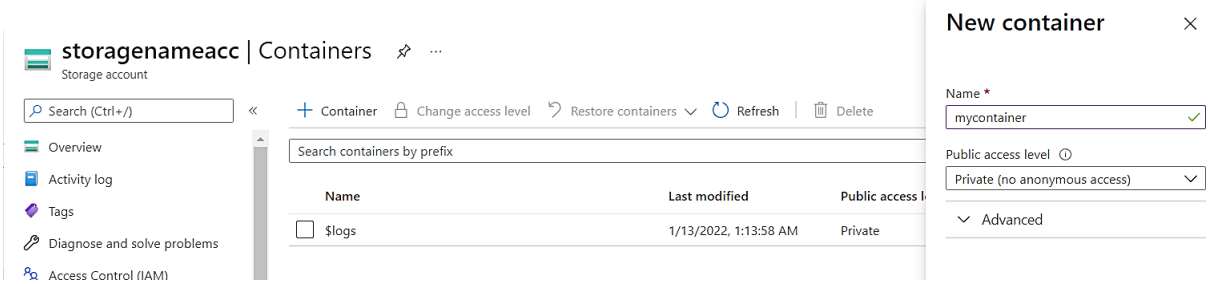

Figure 12: Creating a Container

The next step is to upload the Reference data into the container.

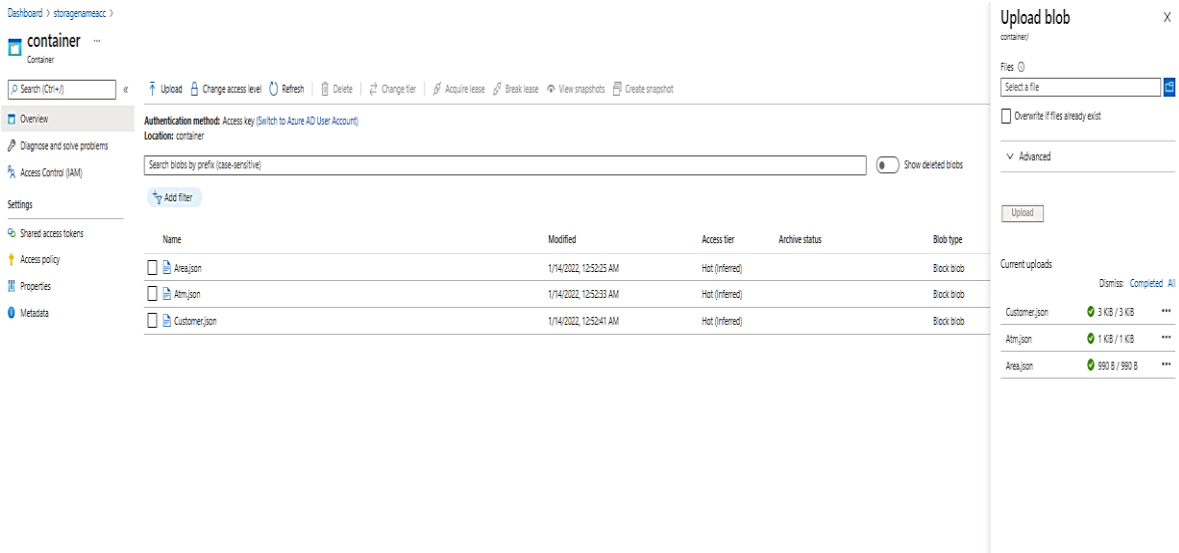

Figure 13: Insert Reference Data into the container

Set up a Stream Analytics Job

To run live stream task in Azure Analytics we should first create a Stream Analytics Job.

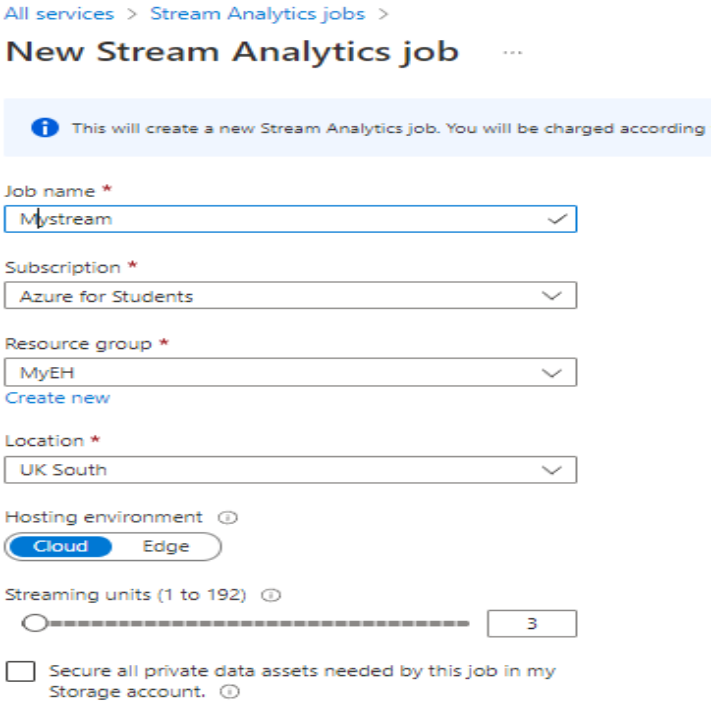

Figure 14: Creation of a Stream Analytics Job

We can see below that the deployment of the new steam Analytics job was successful. Then we press “Go to resource”.

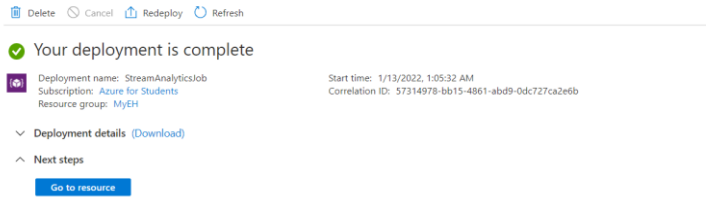

Figure 15: Successful Deployment of a Stream Analytics Job

Part 2 – Azure Analytics Tasks and Solutions

After creating the Stream Analytic Job “Mystream”, we are going to create inputs, outputs for the different queries we are going to examine.

Create Inputs

We created 4 Inputs. The main input that we named as “Input” is a Stream – Event Hub Input and we will use it to download a data sample to test the validity of our queries. The extra inputs that we named as “InputArea”, “InputATM” and “InputCustomer”, are Reference – Blob storage inputs one for each reference data (Area, ATM and Customer). These inputs will be useful in joining the tables together in queries.

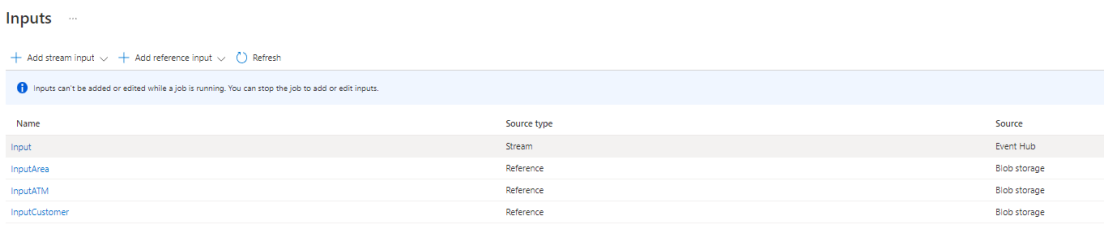

Figure 16: The appropriate inputs for the queries

In the outputs below we can see how we created these inputs.

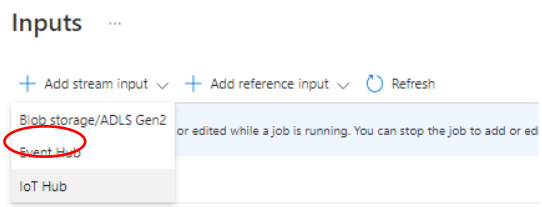

Figure 17: Creation of the Stream Event Hub input

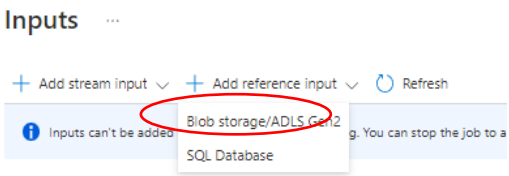

Figure 18: Creation of the Reference – blob storage input

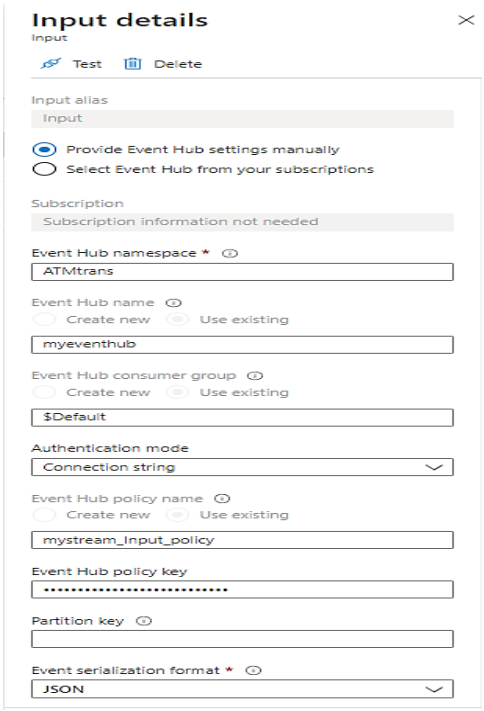

Figure 19: Creation of the Stream Event Hub Input

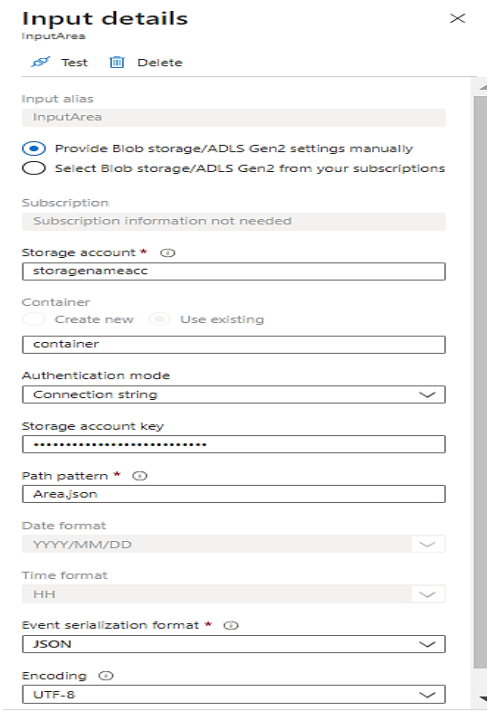

Figure 20: Creation of the Reference input

Create Outputs

In this section we are going to create 9 Blob storage outputs. One main output that we can use to check the validity of our queries in the Test environment and 8 extra queries to create different folders into the container, one for each query.

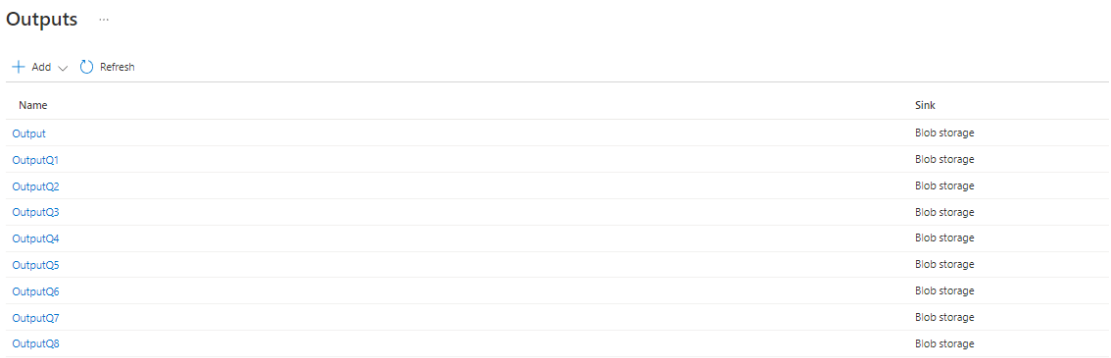

Figure 21: Presentation of the outputs we have created

In the output below we can see how we created these Outputs. We should note here that for each Output Qi we declare the path pattern as Query i.

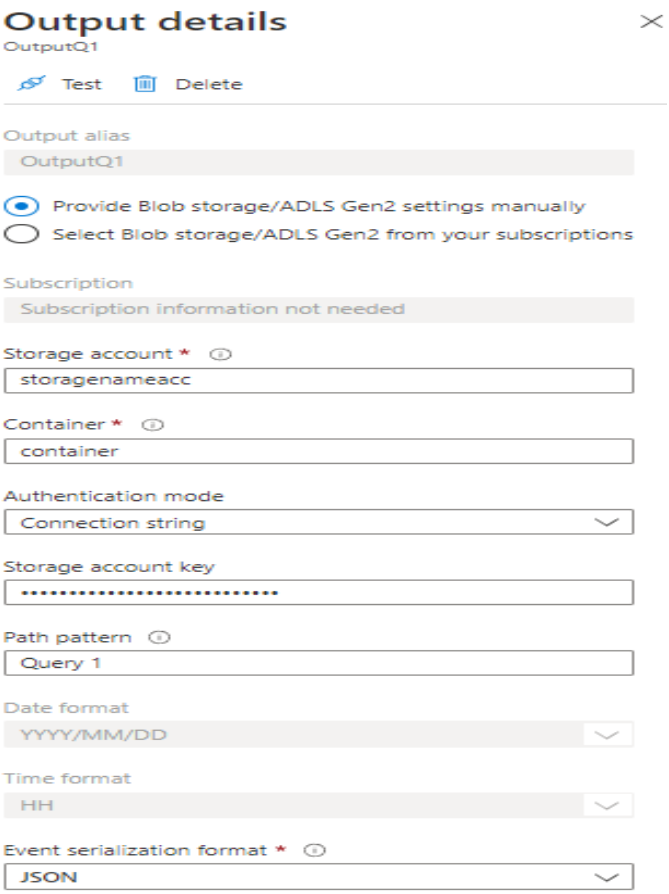

Figure 22: Creation of the Outputs

After running the queries, we will see the following classification in the container.

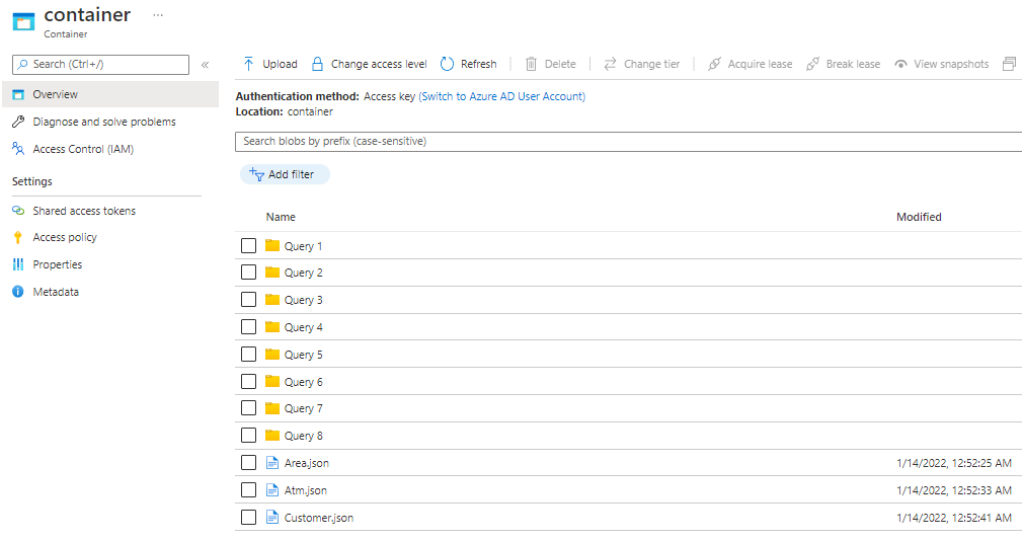

Figure 23: Folder classification of the outputs

Feed queries with a data sample

The process we are going to follow to execute a Stream analytic job query is the following. We first test the validity of the query using a small data sample from the “Input” input and using the reference data in the test environment. Once a query is well written we Save the query, and we execute the stream pressing “Start” in the main page of “mystream”. Once the stream finishes, we will be able to view the results in the container page. Below there are presented some outputs from the described process.

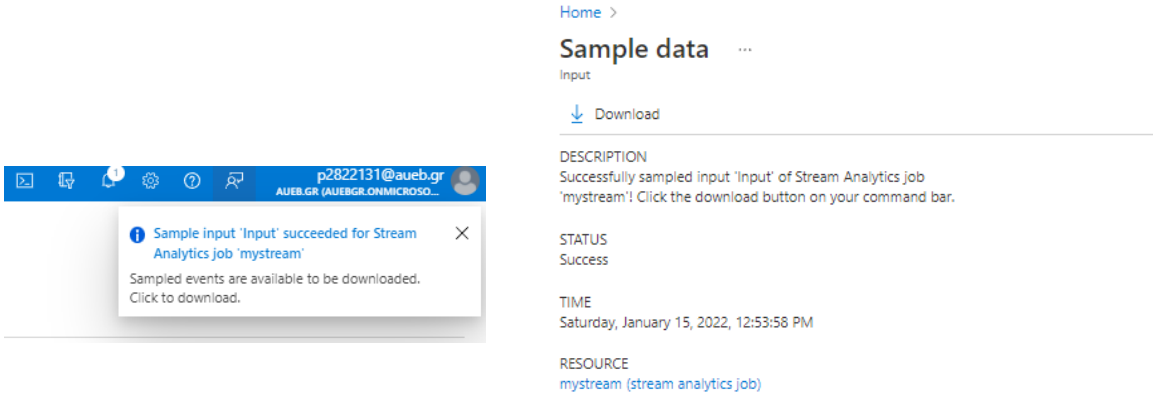

Figure 24: Downloading a Data Sample

Below we can see a part of the data sample we have downloaded.

```
[{"ATMCode":19,
"CardNumber":5200253312538103,
"Type":1,
"Amount":32,
"EventProcessedUtcTime":"2022-01-15T23:59:31.5647107Z",
"PartitionId":0,
"EventEnqueuedUtcTime":"2022-01-15T23:59:18.6430000Z"},
{"ATMCode":12,
"CardNumber":5893112367133403000,
"Type":1,
"Amount":10,
"EventProcessedUtcTime":"2022-01-15T23:59:31.5647107Z",
"PartitionId":0,
"EventEnqueuedUtcTime":"2022-01-15T23:59:18.6590000Z"},
{"ATMCode":19,
"CardNumber":50384191807294800,
"Type":0,
"Amount":19,
"EventProcessedUtcTime":"2022-01-15T23:59:31.5647107Z",
"PartitionId":0,
"EventEnqueuedUtcTime":"2022-01-15T23:59:19.3620000Z"},
{"ATMCode":18,
"CardNumber":56022176913710210,
"Type":0,
"Amount":11,
"EventProcessedUtcTime":"2022-01-15T23:59:31.5647107Z",
"PartitionId":0,
"EventEnqueuedUtcTime":"2022-01-15T23:59:20.6270000Z"},
{"ATMCode":13,
"CardNumber":50383945269330136,
"Type":0,
"Amount":36,
"EventProcessedUtcTime":"2022-01-15T23:59:31.5647107Z",
"PartitionId":0,
"EventEnqueuedUtcTime":"2022-01-15T23:59:21.6430000Z"},
{"ATMCode":13,
"CardNumber":50383945269330136,
"Type":0,
"Amount":10,
"EventProcessedUtcTime":"2022-01-15T23:59:31.5647107Z",
"PartitionId":0,
"EventEnqueuedUtcTime":"2022-01-15T23:59:22.6120000Z"},
{"ATMCode":15,
"CardNumber":30487898026193,
"Type":1,
"Amount":24,
"EventProcessedUtcTime":"2022-01-15T23:59:31.5647107Z",
"PartitionId":0,
"EventEnqueuedUtcTime":"2022-01-15T23:59:23.6590000Z"},
{"ATMCode":19,
"CardNumber":5602246755688900,
"Type":1,
"Amount":12,
```


Figure 25: The data sample that has been downloaded

We upload the data sample in the “Input” and we upload as well the Reference data files “Area.json”, “Atm.json” and “Customer.json” in the “inputArea”, “inputATM” and “InputCustomer” respectively.

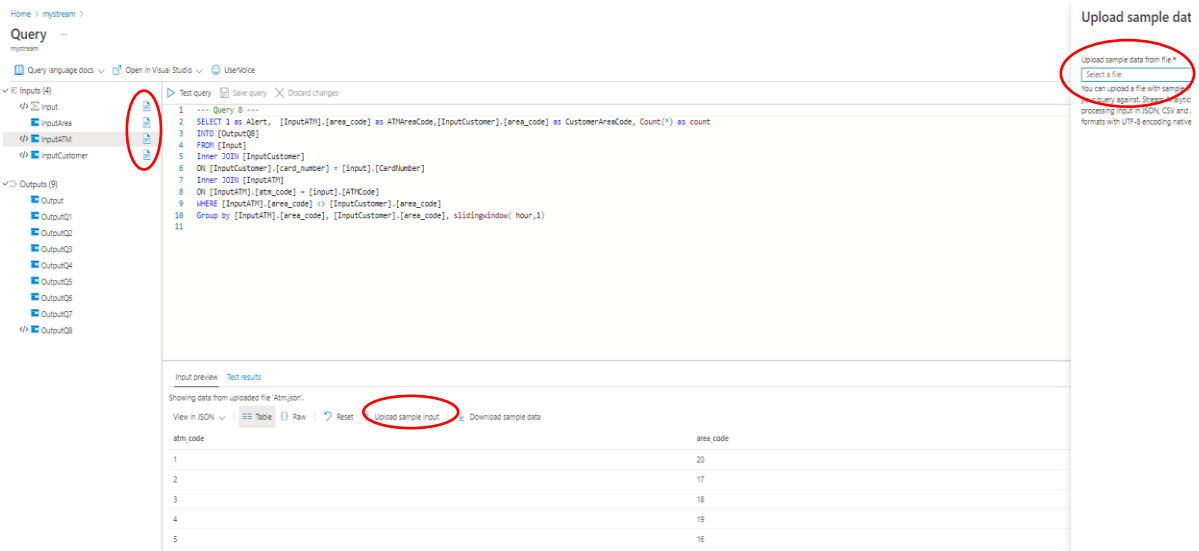

Figure 26: Upload the sample data and test the query

Last step is to press the start button.

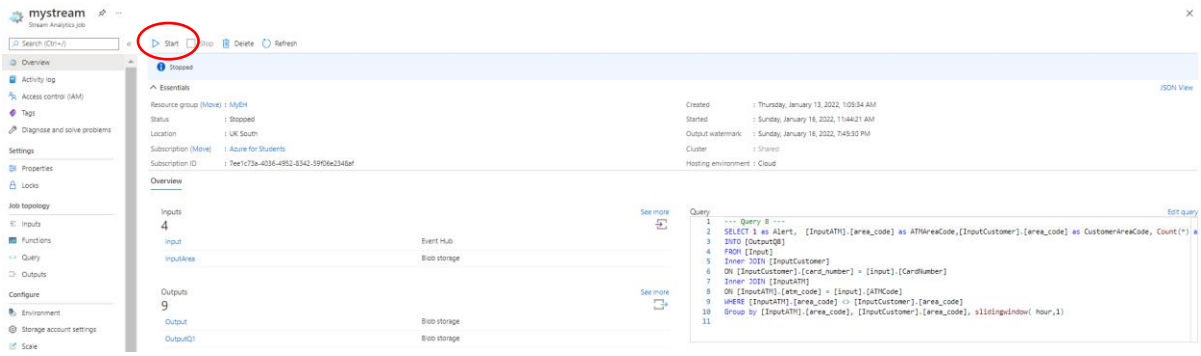

Figure 27: Starting the process

Azure Analytics Tasks and Solutions

Now we are ready to create queries that can execute important tasks at any time. The most frequently needed task that we are going to analyze are the following:

Task1: *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).*

First, we created the appropriate query, and we did a run test. The test run returns no outputs because our input sample does not include the ATMCode 21.

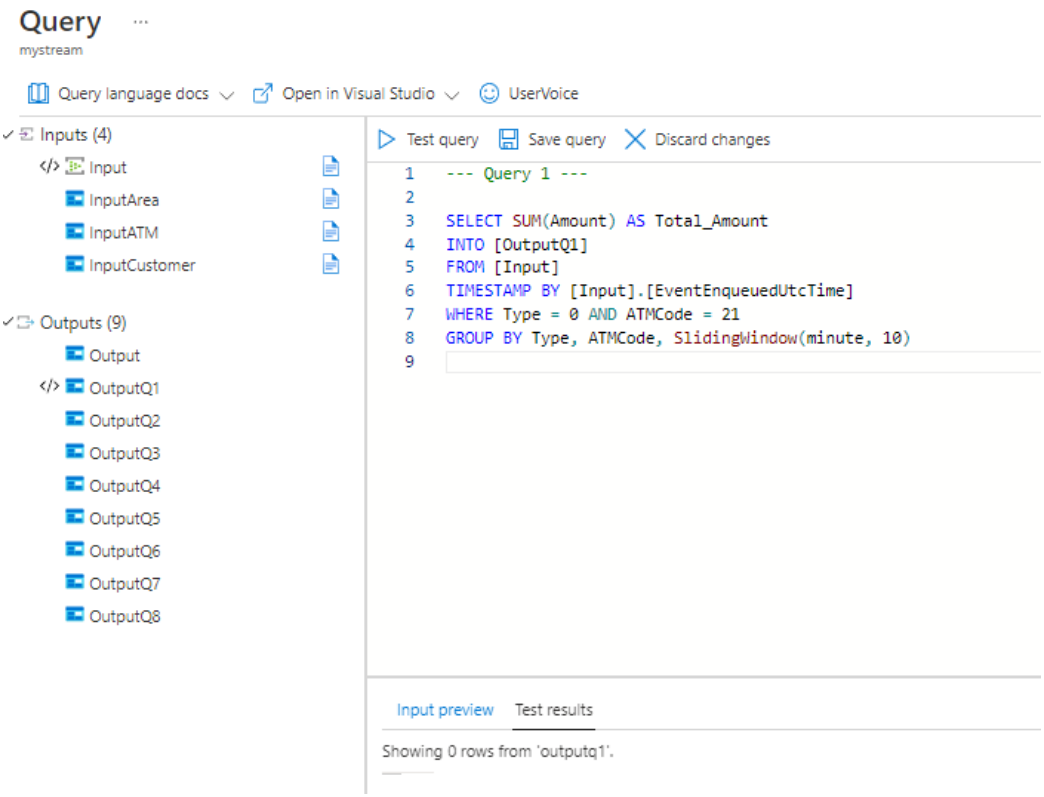

Figure 28: Testing the query

After starting the analytic stream for an hour, the output presented in the container folder "Query 1 "was the following

```
1  {"Total_Amount":18.0}
2  {"Total_Amount":45.0}
3  {"Total_Amount":71.0}
4  {"Total_Amount":116.0}
5  {"Total_Amount":162.0}
6  {"Total_Amount":208.0}
7  {"Total_Amount":247.0}
8  {"Total_Amount":229.0}
9  {"Total_Amount":202.0}
10 {"Total_Amount":176.0}
11 {"Total_Amount":218.0}
12 {"Total_Amount":173.0}
13 {"Total_Amount":127.0}
14 {"Total_Amount":81.0}
15 {"Total_Amount":42.0}
```

Figure 29:Output of the Stream

Task2: *Show the total "Amount" of "Type=1" transactions at "ATMCode=21" of the last hour. Repeat once every hour (use a tumbling window).*

First, we created the appropriate query, and we did a run test. The test run returns no outputs because our input sample does not include the ATMCode 21.

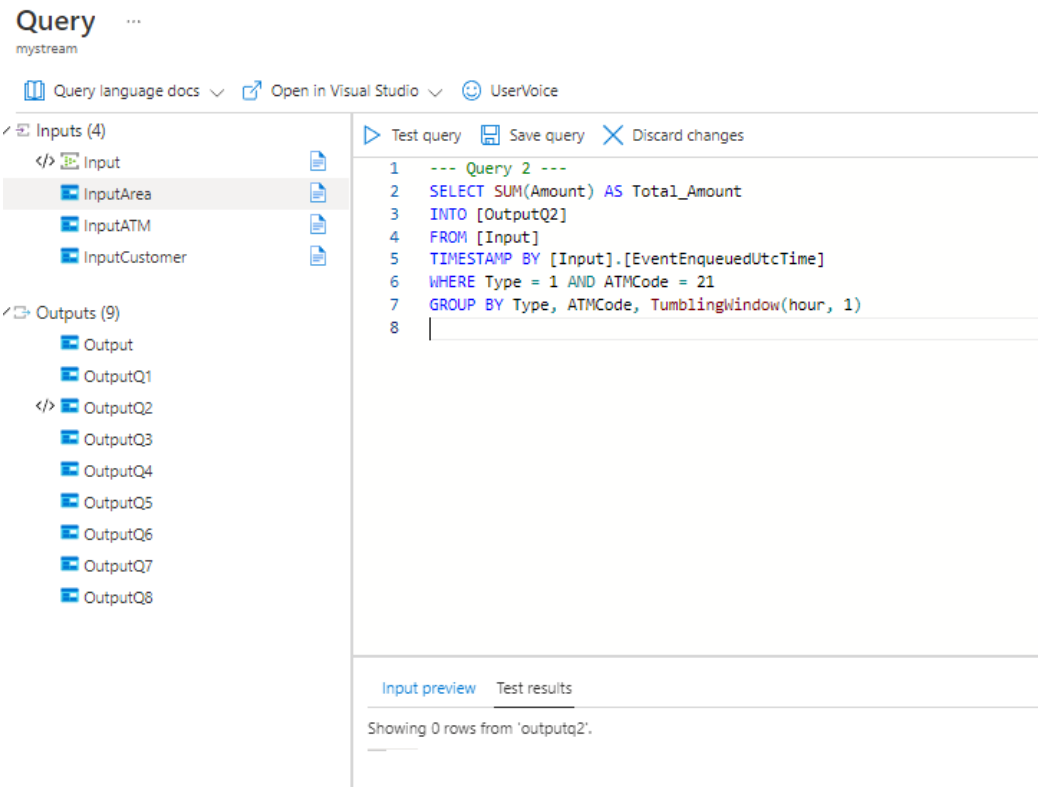

Figure 30: Testing the query

After starting the analytic stream for an hour, the output presented in the container folder "Query 2"was the following:

```
1    {"Total_Amount":472.0}
```

Figure 31: Output of the stream

Task3: *Show the total "Amount" of "Type =1" transactions at "ATM Code =21" of the last hour. Repeat once every 30 minutes (use hopping window)*

First, we created the appropriate query, and we did a run test. The test run returns no outputs because our input sample does not include the ATMCode 21.

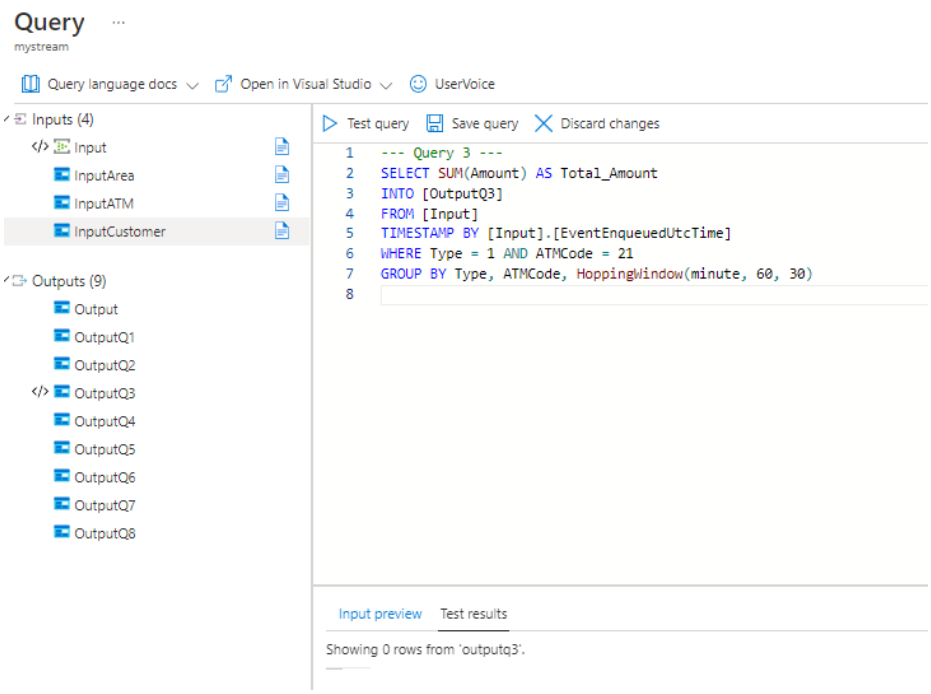

Figure 32: Testing the query

After starting the analytic stream for an hour, the output presented in the container folder "Query 3 "was the following

```
1   {"Total_Amount":42.0}
2   {"Total_Amount":137.0}
```

Figure 33: Output of the stream

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

First, we created the appropriate query, and we did a run test. The test run returns the output above.

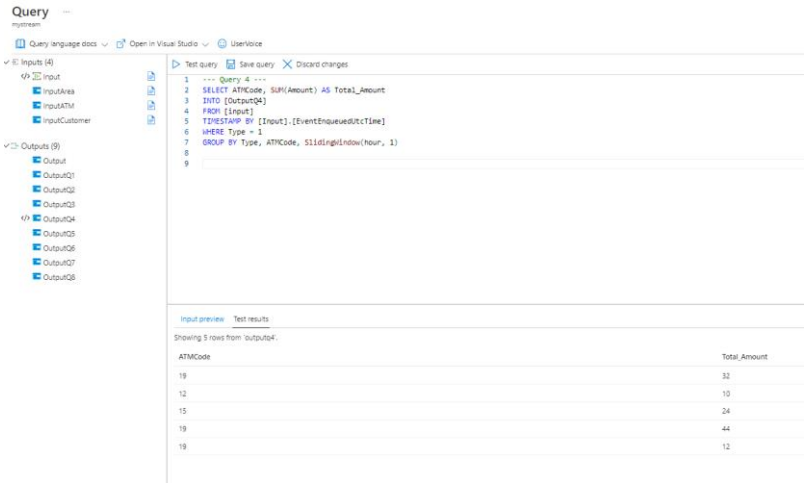

Figure 34: Testing the query

After starting the analytic stream for several hours, the output presented in the container folder "Query 4 "was the following

```
{"ATMCode":15,"Total_Amount":275.0}
{"ATMCode":18,"Total_Amount":204.0}
{"ATMCode":18,"Total_Amount":179.0}
{"ATMCode":19,"Total_Amount":80.0}
{"ATMCode":20,"Total_Amount":113.0}
{"ATMCode":18,"Total_Amount":167.0}
{"ATMCode":15,"Total_Amount":302.0}
{"ATMCode":20,"Total_Amount":100.0}
{"ATMCode":20,"Total_Amount":76.0}
{"ATMCode":19,"Total_Amount":118.0}
{"ATMCode":21,"Total_Amount":42.0}
{"ATMCode":15,"Total_Amount":264.0}
{"ATMCode":15,"Total_Amount":239.0}
{"ATMCode":13,"Total_Amount":89.0}
{"ATMCode":15,"Total_Amount":208.0}
{"ATMCode":13,"Total_Amount":73.0}
{"ATMCode":18,"Total_Amount":119.0}
{"ATMCode":15,"Total_Amount":163.0}
{"ATMCode":10,"Total_Amount":195.0}
{"ATMCode":15,"Total_Amount":149.0}
{"ATMCode":10,"Total_Amount":158.0}
{"ATMCode":19,"Total_Amount":88.0}
{"ATMCode":18,"Total_Amount":82.0}
{"ATMCode":15,"Total_Amount":102.0}
{"ATMCode":18,"Total_Amount":40.0}
{"ATMCode":20,"Total_Amount":54.0}
{"ATMCode":13,"Total_Amount":27.0}
{"ATMCode":15,"Total_Amount":54.0}
{"ATMCode":20,"Total_Amount":30.0}
{"ATMCode":15,"Total_Amount":27.0}
{"ATMCode":10,"Total_Amount":116.0}
{"ATMCode":10,"Total_Amount":85.0}
{"ATMCode":10,"Total_Amount":58.0}
{"ATMCode":10,"Total_Amount":46.0}
```


Figure 35: Output of the stream

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

First, we created the appropriate query, and we did a run test. The test run returns the output above:

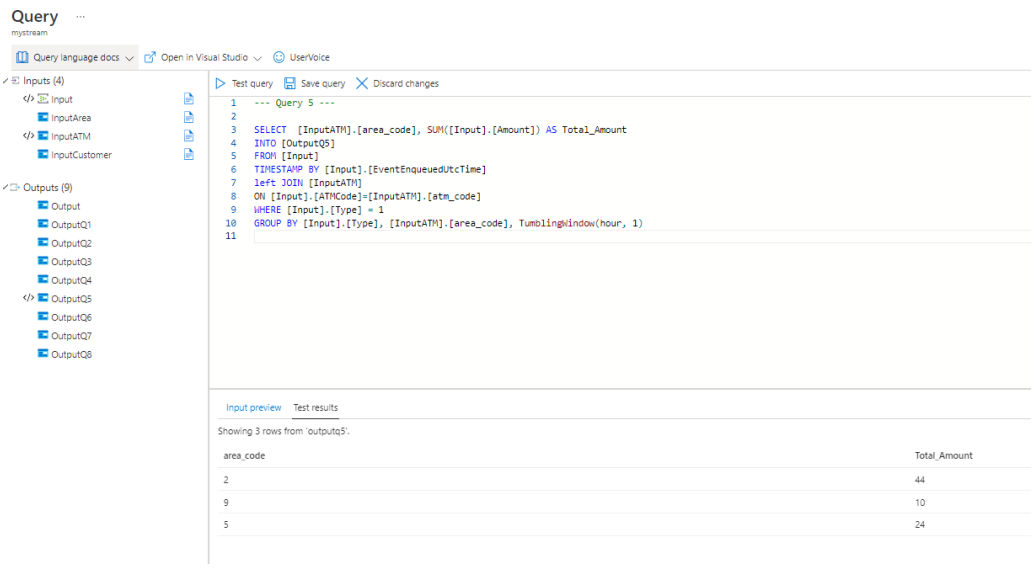

Figure 36: Testing the query

After starting the analytic stream for several hours, the output presented in the container folder "Query 5 "was the following:

```
1  {"area_code":5,"Total_Amount":2654.0}
2  {"area_code":4,"Total_Amount":2070.0}
3  {"area_code":14,"Total_Amount":50.0}
4  {"area_code":11,"Total_Amount":1779.0}
5  {"area_code":19,"Total_Amount":83.0}
6  {"area_code":3,"Total_Amount":535.0}
7  {"area_code":6,"Total_Amount":51.0}
8  {"area_code":16,"Total_Amount":31.0}
9  {"area_code":9,"Total_Amount":637.0}
10 {"area_code":12,"Total_Amount":45.0}
11 {"area_code":13,"Total_Amount":36.0}
12 {"area_code":7,"Total_Amount":400.0}
13 {"area_code":10,"Total_Amount":737.0}
14 {"area_code":1,"Total_Amount":2042.0}
15 {"area_code":2,"Total_Amount":1285.0}
16 {"area_code":5,"Total_Amount":74.0}
17 {"area_code":4,"Total_Amount":70.0}
18 {"area_code":11,"Total_Amount":260.0}
19 {"area_code":3,"Total_Amount":55.0}
20 {"area_code":7,"Total_Amount":61.0}
21 {"area_code":10,"Total_Amount":18.0}
22 {"area_code":1,"Total_Amount":126.0}
23 {"area_code":2,"Total_Amount":141.0}
```


Figure 37: Output of the stream

Task 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).*

First, we created the appropriate query, and we did a run test. The test run returns the output above:

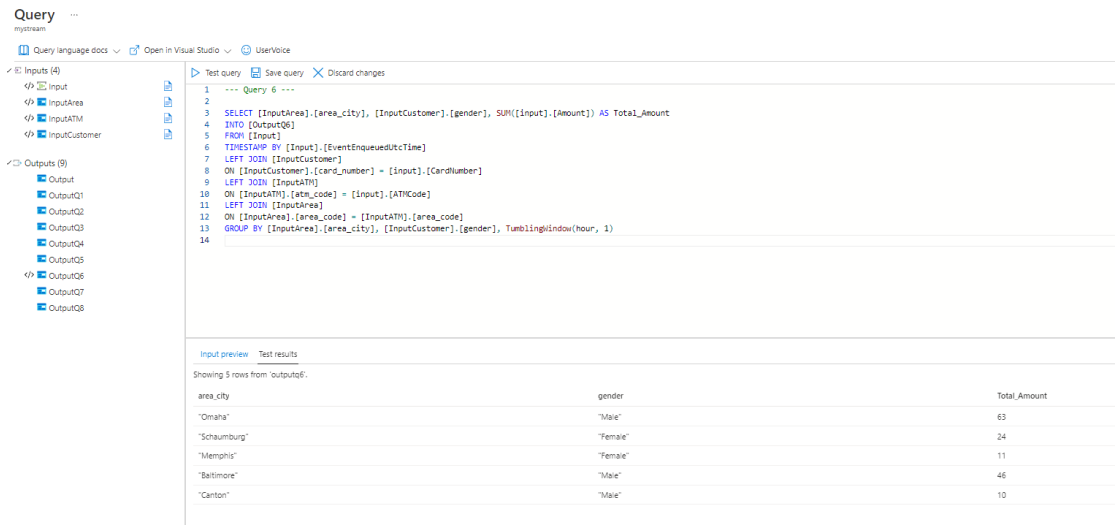

Figure 38: Testing the query

After starting the analytic stream for several hours, the output presented in the container folder "Query 6 "was the following:

```
{"area_city":"Vancouver","gender":"Female","Total_Amount":128.0}
{"area_city":"Dayton","gender":"Female","Total_Amount":49.0}
{"area_city":"Canton","gender":"Female","Total_Amount":49.0}
{"area_city":null,"gender":"Female","Total_Amount":2791.0}
{"area_city":"Schaumburg","gender":"Male","Total_Amount":1469.0}
{"area_city":"Baltimore","gender":"Male","Total_Amount":669.0}
{"area_city":"Baltimore","gender":"Female","Total_Amount":744.0}
{"area_city":"Memphis","gender":"Male","Total_Amount":2750.0}
{"area_city":"Springfield","gender":"Female","Total_Amount":1281.0}
{"area_city":null,"gender":"Male","Total_Amount":1369.0}
{"area_city":"Memphis","gender":"Female","Total_Amount":1108.0}
{"area_city":"Schaumburg","gender":"Female","Total_Amount":3348.0}
{"area_city":"Omaha","gender":"Male","Total_Amount":2084.0}
{"area_city":"Vancouver","gender":"Male","Total_Amount":928.0}
{"area_city":"Springfield","gender":"Male","Total_Amount":2719.0}
{"area_city":"Dayton","gender":"Male","Total_Amount":15.0}
{"area_city":"Tacoma","gender":"Female","Total_Amount":960.0}
{"area_city":"Omaha","gender":"Female","Total_Amount":437.0}
{"area_city":"Canton","gender":"Male","Total_Amount":1329.0}
{"area_city":null,"gender":"Female","Total_Amount":389.0}
{"area_city":"Schaumburg","gender":"Male","Total_Amount":126.0}
{"area_city":"Baltimore","gender":"Male","Total_Amount":103.0}
{"area_city":"Memphis","gender":"Male","Total_Amount":173.0}
{"area_city":"Baltimore","gender":"Female","Total_Amount":18.0}
{"area_city":"Springfield","gender":"Female","Total_Amount":140.0}
{"area_city":null,"gender":"Male","Total_Amount":137.0}
{"area_city":"Memphis","gender":"Female","Total_Amount":13.0}
{"area_city":"Schaumburg","gender":"Female","Total_Amount":133.0}
{"area_city":"Omaha","gender":"Male","Total_Amount":206.0}
{"area_city":"Vancouver","gender":"Male","Total_Amount":55.0}
{"area_city":"Springfield","gender":"Male","Total_Amount":97.0}
{"area_city":"Tacoma","gender":"Female","Total_Amount":61.0}
{"area_city":"Canton","gender":"Male","Total_Amount":53.0}
```


Figure 39: Output of the stream

Task 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).*

First, we created the appropriate query, and we did a run test. The test run returns the output above:

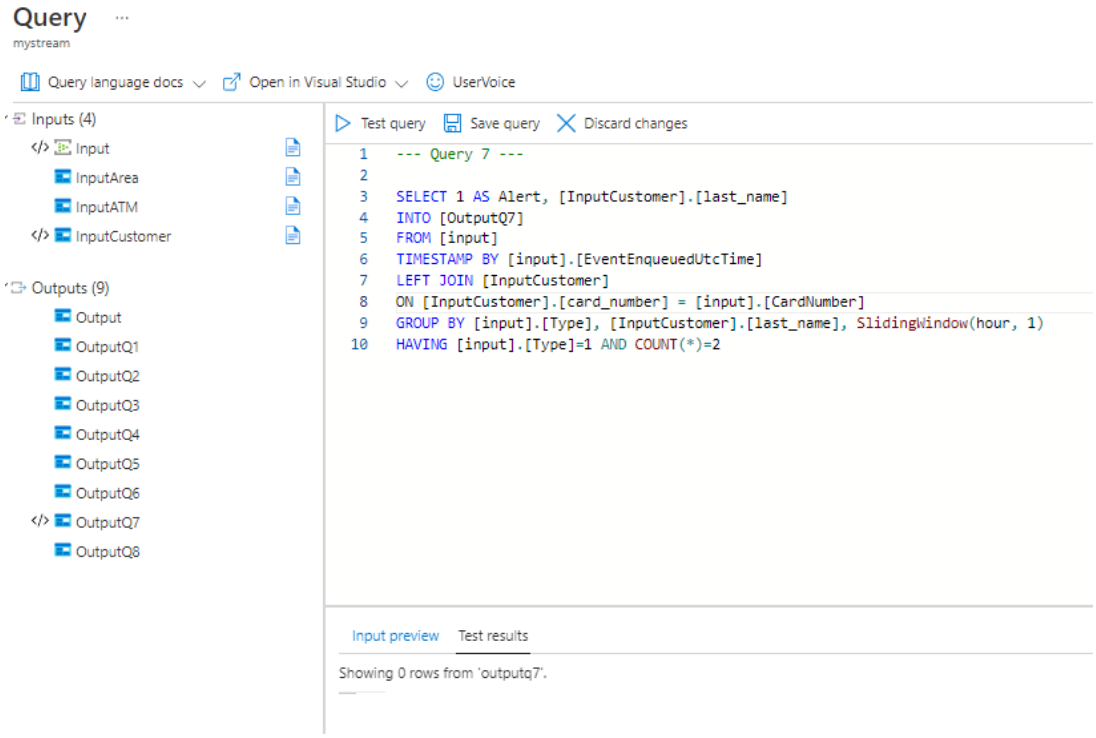

Figure 40: Testing the query

After starting the analytic stream for several hours, the output presented in the container folder “Query 7 “was the following:

```
{"Alert":1,"last_name":"Fuller"}
{"Alert":1,"last_name":"Mitchell"}
{"Alert":1,"last_name":"Perry"}
{"Alert":1,"last_name":"Perez"}
{"Alert":1,"last_name":"Simpson"}
{"Alert":1,"last_name":"Sims"}
{"Alert":1,"last_name":"Mason"}
{"Alert":1,"last_name":"Snyder"}
{"Alert":1,"last_name":"Young"}
{"Alert":1,"last_name":"Stone"}
{"Alert":1,"last_name":"Day"}
{"Alert":1,"last_name":"Russell"}
{"Alert":1,"last_name":"Jordan"}
{"Alert":1,"last_name":"Moreno"}
{"Alert":1,"last_name":"Morrison"}
{"Alert":1,"last_name":"Lee"}
{"Alert":1,"last_name":"Hansen"}
{"Alert":1,"last_name":"Cooper"}
{"Alert":1,"last_name":"Bradley"}
{"Alert":1,"last_name":"Carroll"}
{"Alert":1,"last_name":"Stone"}
{"Alert":1,"last_name":"Young"}
{"Alert":1,"last_name":"Simpson"}
{"Alert":1,"last_name":"Perry"}
{"Alert":1,"last_name":"Mitchell"}
{"Alert":1,"last_name":"Hansen"}
{"Alert":1,"last_name":"Jordan"}
{"Alert":1,"last_name":"Cooper"}
{"Alert":1,"last_name":"Moreno"}
{"Alert":1,"last_name":"Perez"}
{"Alert":1,"last_name":"Lee"}
{"Alert":1,"last_name":"Mason"}
{"Alert":1,"last_name":"Morrison"}
{"Alert":1,"last_name":"Fuller"}
{"Alert":1,"last_name":"Russell"}
{"Alert":1,"last_name":"Day"}
{"Alert":1,"last_name":"Snyder"}
{"Alert":1,"last_name":"Bradley"}
{"Alert":1,"last_name":"Sims"}
{"Alert":1,"last_name":"Russell"}
{"Alert":1,"last_name":"Bradley"}
{"Alert":1,"last_name":"Fuller"}
{"Alert":1,"last_name":"Hansen"}
{"Alert":1,"last_name":"Mitchell"}
{"Alert":1,"last_name":"Lee"}
{"Alert":1,"last_name":"Young"}
```


Figure 41: Output of the stream

Task 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)*

First, we created the appropriate query, and we did a run test. The test run returns the output above:

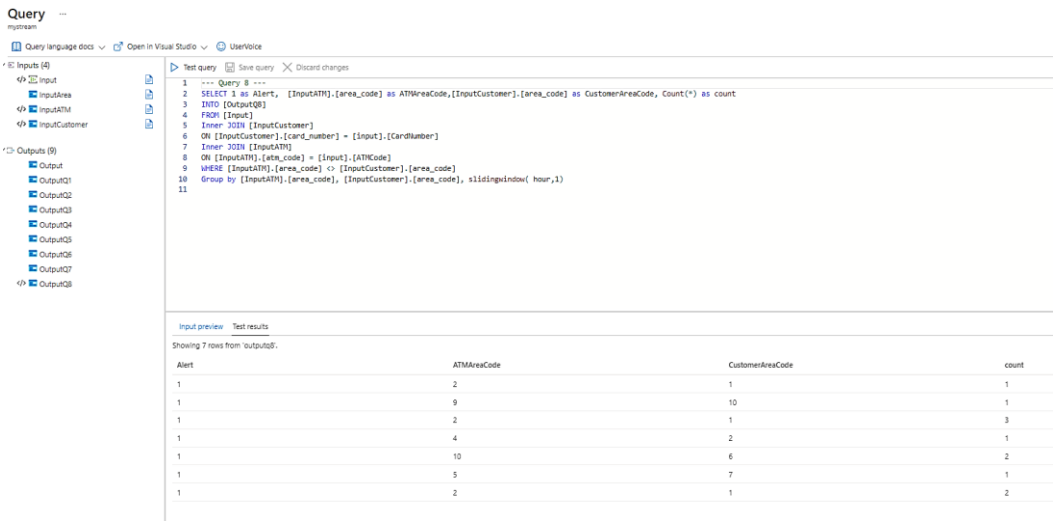

Figure 42: Testing the query

After starting the analytic stream for several hours, the output presented in the container folder "Query 8 "was the following:


```
{"Alert":1,"ATMAreaCode":10,"CustomerAreaCode":6,"count":4}
{"Alert":1,"ATMAreaCode":2,"CustomerAreaCode":1,"count":8}
{"Alert":1,"ATMAreaCode":2,"CustomerAreaCode":1,"count":7}
{"Alert":1,"ATMAreaCode":1,"CustomerAreaCode":6,"count":3}
{"Alert":1,"ATMAreaCode":5,"CustomerAreaCode":7,"count":5}
{"Alert":1,"ATMAreaCode":11,"CustomerAreaCode":8,"count":10}
{"Alert":1,"ATMAreaCode":11,"CustomerAreaCode":8,"count":9}
{"Alert":1,"ATMAreaCode":10,"CustomerAreaCode":6,"count":3}
{"Alert":1,"ATMAreaCode":10,"CustomerAreaCode":6,"count":2}
{"Alert":1,"ATMAreaCode":11,"CustomerAreaCode":8,"count":8}
{"Alert":1,"ATMAreaCode":5,"CustomerAreaCode":7,"count":4}
{"Alert":1,"ATMAreaCode":10,"CustomerAreaCode":6,"count":1}
{"Alert":1,"ATMAreaCode":11,"CustomerAreaCode":8,"count":7}
{"Alert":1,"ATMAreaCode":1,"CustomerAreaCode":6,"count":2}
{"Alert":1,"ATMAreaCode":1,"CustomerAreaCode":6,"count":1}
{"Alert":1,"ATMAreaCode":2,"CustomerAreaCode":1,"count":6}
{"Alert":1,"ATMAreaCode":2,"CustomerAreaCode":1,"count":5}
{"Alert":1,"ATMAreaCode":11,"CustomerAreaCode":8,"count":6}
{"Alert":1,"ATMAreaCode":4,"CustomerAreaCode":2,"count":7}
{"Alert":1,"ATMAreaCode":4,"CustomerAreaCode":2,"count":6}
{"Alert":1,"ATMAreaCode":4,"CustomerAreaCode":2,"count":5}
{"Alert":1,"ATMAreaCode":4,"CustomerAreaCode":2,"count":4}
{"Alert":1,"ATMAreaCode":5,"CustomerAreaCode":7,"count":3}
{"Alert":1,"ATMAreaCode":11,"CustomerAreaCode":8,"count":5}
{"Alert":1,"ATMAreaCode":5,"CustomerAreaCode":7,"count":2}
{"Alert":1,"ATMAreaCode":11,"CustomerAreaCode":8,"count":4}
{"Alert":1,"ATMAreaCode":9,"CustomerAreaCode":10,"count":1}
{"Alert":1,"ATMAreaCode":5,"CustomerAreaCode":7,"count":1}
{"Alert":1,"ATMAreaCode":2,"CustomerAreaCode":1,"count":4}
{"Alert":1,"ATMAreaCode":11,"CustomerAreaCode":8,"count":3}
{"Alert":1,"ATMAreaCode":11,"CustomerAreaCode":8,"count":2}
{"Alert":1,"ATMAreaCode":2,"CustomerAreaCode":1,"count":3}
{"Alert":1,"ATMAreaCode":11,"CustomerAreaCode":8,"count":1}
{"Alert":1,"ATMAreaCode":4,"CustomerAreaCode":2,"count":3}
{"Alert":1,"ATMAreaCode":2,"CustomerAreaCode":1,"count":2}
{"Alert":1,"ATMAreaCode":2,"CustomerAreaCode":1,"count":1}
{"Alert":1,"ATMAreaCode":4,"CustomerAreaCode":2,"count":2}
```


Figure 43: The output of the stream