Athens University of Economics and Business

School of Business

Department of Management Science & Technology

Master of Science in Business Analytics

Program:	Full-time
Quarter:	1st (Fall Quarter)
Course:	Data Management and Business Intelligence
Assignment №:	3
Students (Registration №):	Gkouma Konstantina (f2822208), Souflas Eleftherios-Efthymios (f2822217)

Data Management & Business Intelligence
Assignment #3

Students: Gkouma Konstantina, Souflas Eleftherios-Efthymios

In this assignment we used Azure Stream Analytics to process a virtual data stream of ATM transactions and answer some stream queries. We used as a guideline the notes taken from the relevant Lab's lecture and of course the lecture's presentation guideline.

In order to use Azure's Stream Analytics, we created a student's account (Figure 1) with one of ours AUEB email account, using also our password for verification from https://azure.microsoft.com/en-us/free/students/.

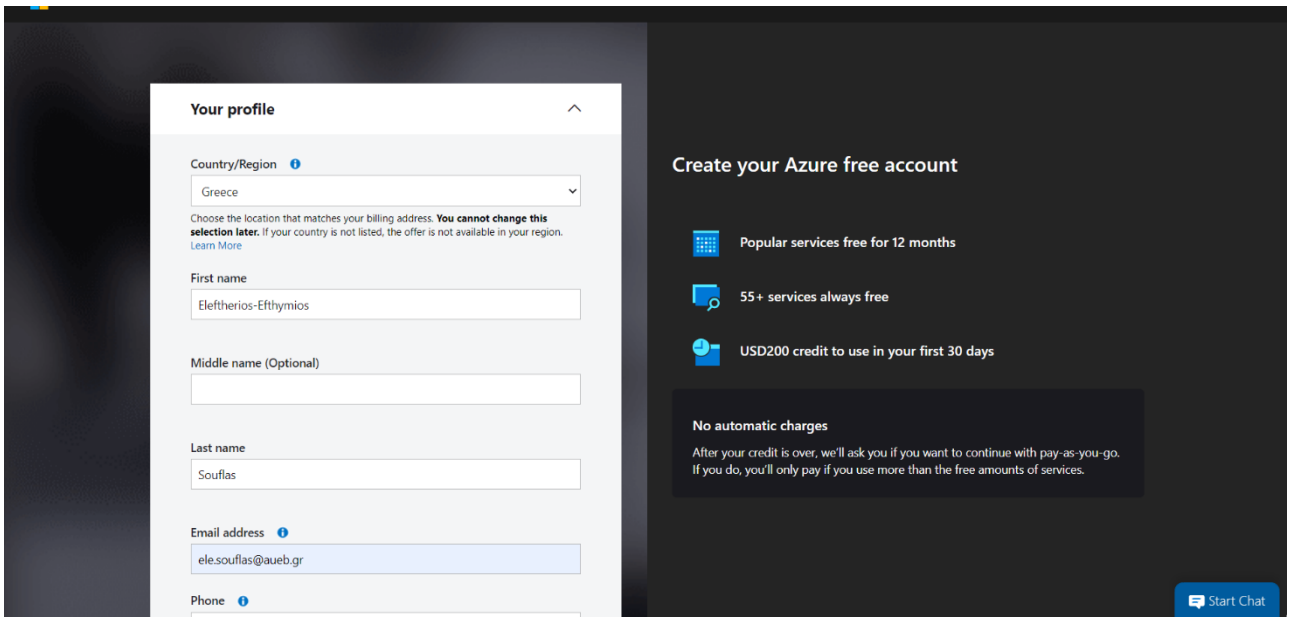

Figure 1 - Azure Student Account

We then created a Namespace (Figure 2), creating on the same way a Resource Group where all our Resources (Event Hub Namespace, Storage Account, Stream Analytics job) will belong.

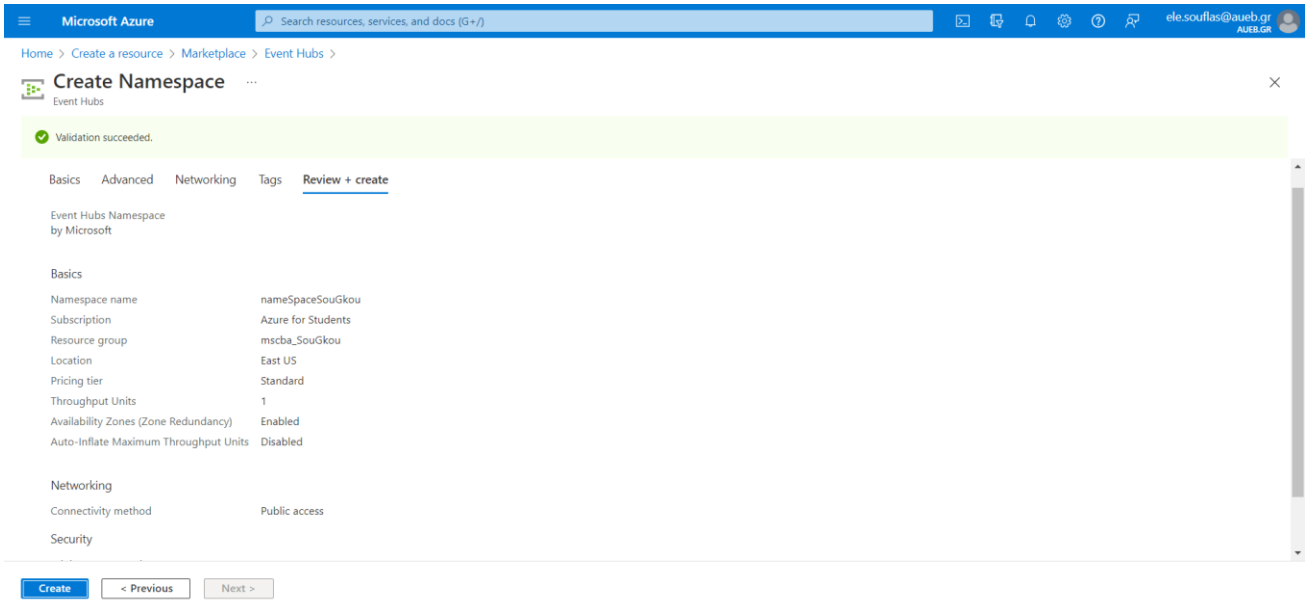

Figure 2 - Namespace creation

We then set up an Event Hub inside the Event Hub Namespace, as shown in Figure 3.

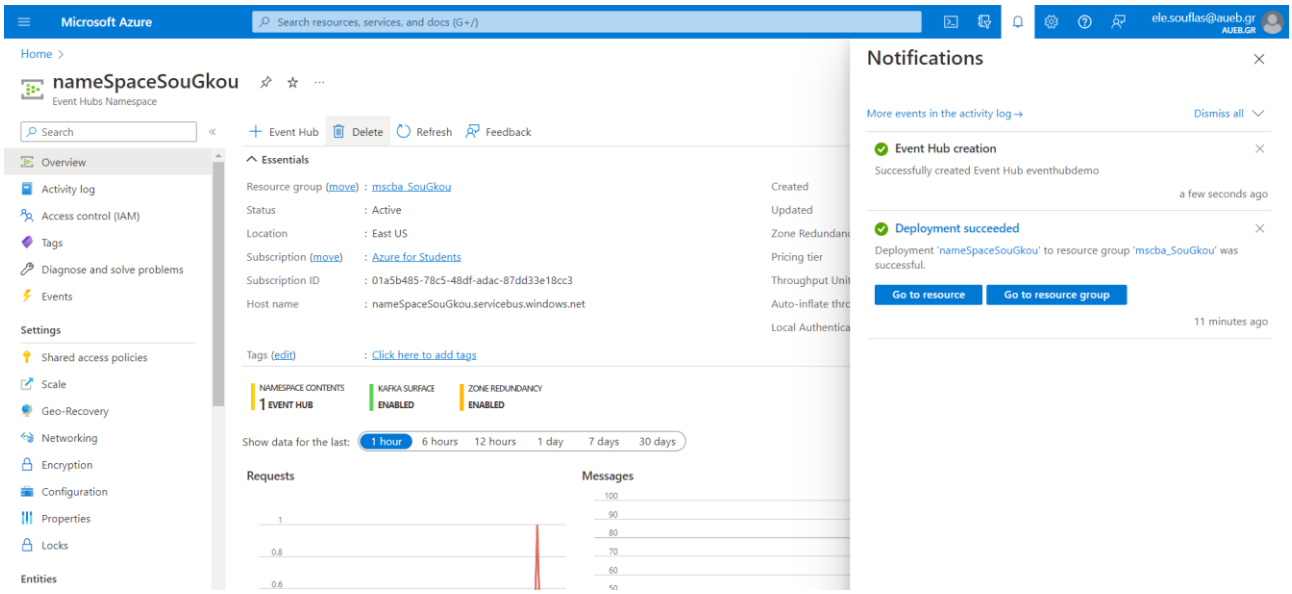

Figure 3 - Event Hub creation

After the Event Hub setup, we created a Shared access policy to manage both rights to send and receive messages to the entity, as shown in Figure 4. We named it "MySendPolicy" because firstly we created two policies (Send, Rec) to differentiate their uses, but as we could not manage to capture properly all the information, by the "MyRecPolicy", sent by the Virtual Data Stream (ATM Transactions Generator), we did not use the latter at all and we altered the former to manage both rights.

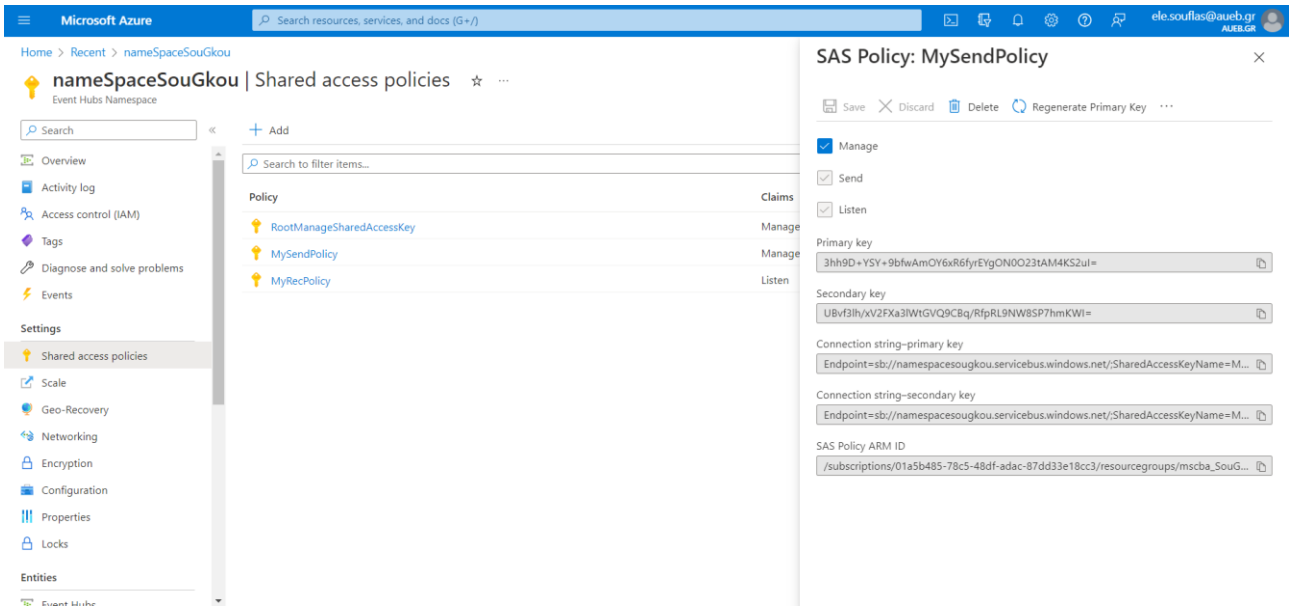

Figure 4 - SAS Policy creation

We copied the Primary Key and with the use of the Event Hubs – Signature Generator (https://github.com/sandrinodimattia/RedDog/releases), we generated a Security Access Signature (Figure 5).

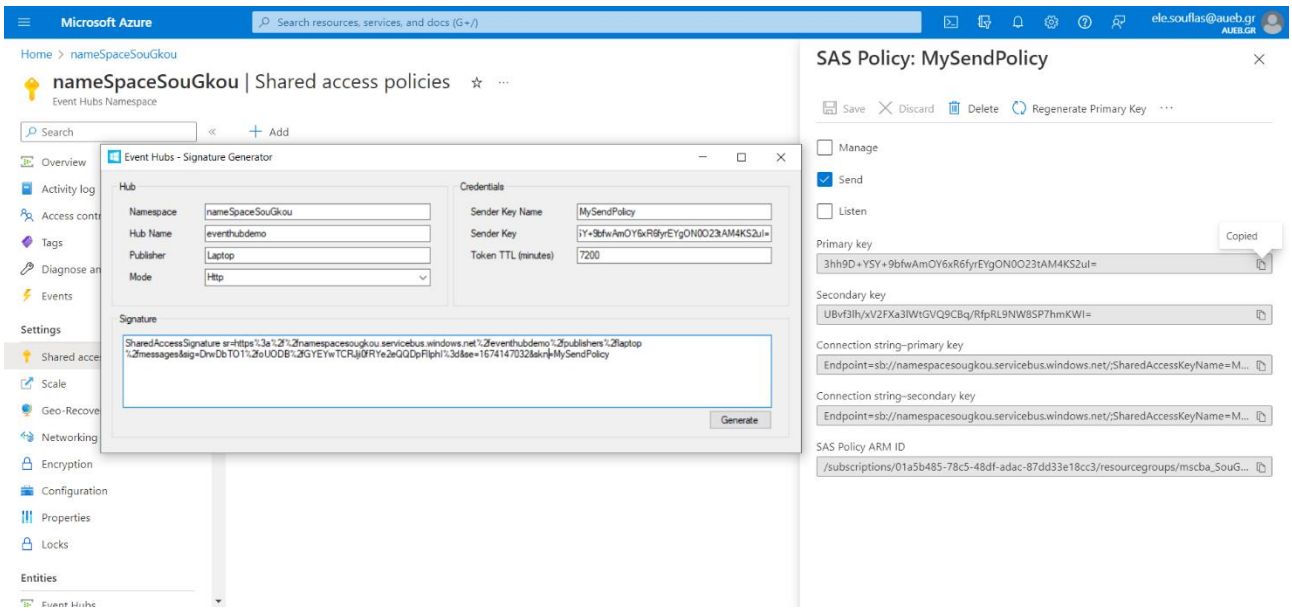

Figure 5 - Generation of SAS Signature

We then edited with Notepad++ the relevant configuration variables (green highlighted – lines 16, 17) of the Generator HTML file (Figure 6), opened the HTML file in a browser and pressed the "Send Data" button to begin the creation of virtual data stream of ATM transactions and feed the Event Hub with it (Figure 7).

Then, we created a Storage account (Figure 8) and uploaded the Reference Data files in a new container inside the Storage account, as shown in Figure 9.

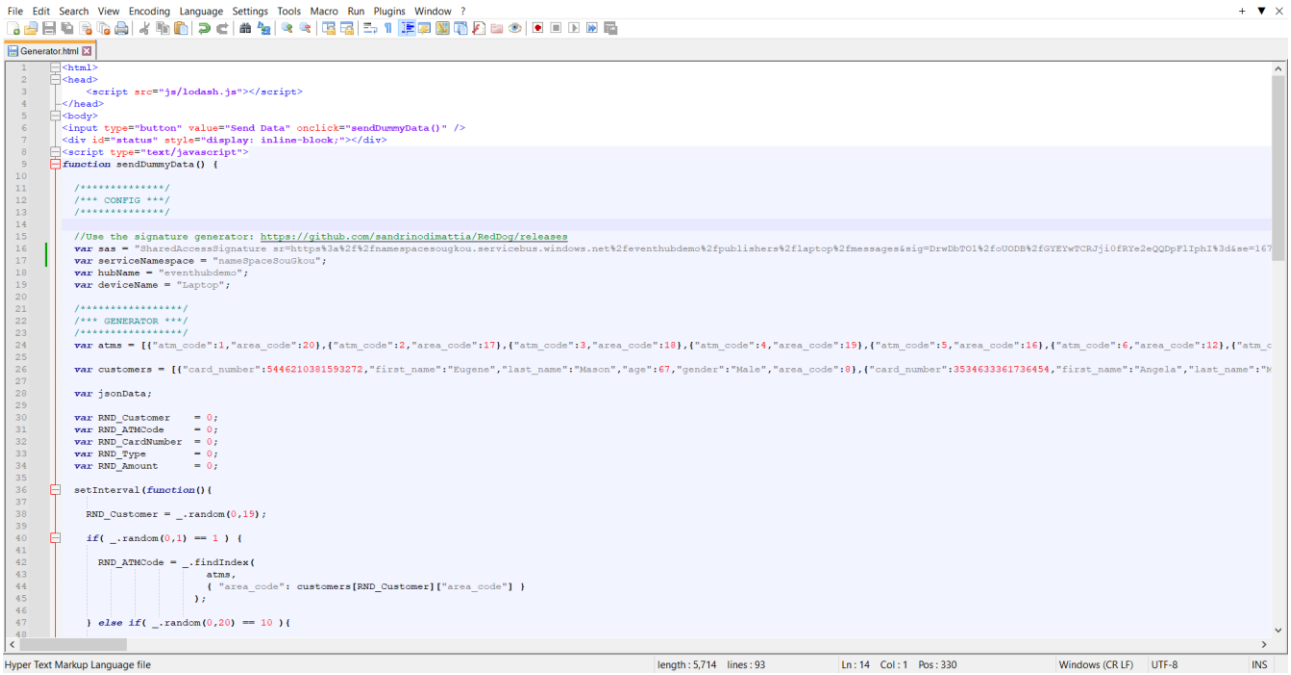

Figure 6 - Config variables update

Send Data Sent: { "ATMCode": 15 , "CardNumber": 30487898026193 , "Type": 1 , "Amount": 14 }

Elements Console Sources Network Performance Memory Application

Preserve log Disable cache No throttling

Filter Invert Hide data URLs All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other

Has blocked cookies Blocked Requests 3rd-party requests

5000 ms 10000 ms 15000 ms 20000 ms 25000 ms 30000 ms 35000 ms

Name	Status	Type	Initiator	Size	Time	Waterfall
messages	201	xhr	Generator.html:86	291 B	160 ms	
messages	200	preflight	Preflight	0 B	143 ms	
messages	201	xhr	Generator.html:86	291 B	161 ms	
messages	200	preflight	Preflight	0 B	136 ms	
messages	201	xhr	Generator.html:86	291 B	156 ms	
messages	200	preflight	Preflight	0 B	137 ms	
messages	201	xhr	Generator.html:86	291 B	200 ms	
messages	200	preflight	Preflight	0 B	137 ms	
messages	201	xhr	Generator.html:86	291 B	174 ms	
messages	200	preflight	Preflight	0 B	138 ms	
messages	201	xhr	Generator.html:86	291 B	155 ms	
messages	200	preflight	Preflight	0 B	137 ms	
messages	201	xhr	Generator.html:86	291 B	158 ms	
messages	200	preflight	Preflight	0 B	142 ms	
messages	201	xhr	Generator.html:86	291 B	155 ms	
messages	200	preflight	Preflight	0 B	137 ms	
messages	201	xhr	Generator.html:86	291 B	161 ms	
messages	200	preflight	Preflight	0 B	136 ms	
messages	201	xhr	Generator.html:86	291 B	163 ms	
messages	200	preflight	Preflight	0 B	138 ms	
messages	201	xhr	Generator.html:86	291 B	156 ms	
messages	200	preflight	Preflight	0 B	137 ms	
messages	201	xhr	Generator.html:86	291 B	156 ms	

84 requests | 12.2 kB transferred | 0 B resources

Figure 7 - ATM transactions data stream

Microsoft Azure

Home > Storage accounts >

Create a storage account

Basics Advanced Networking Data protection Encryption Tags Review

Basics

Subscription	Azure for Students
Resource Group	mscba_SouGkou
Location	eastus
Storage account name	mscbastorage1
Deployment model	Resource manager
Performance	Standard
Replication	Read-access geo-redundant storage (RA-GRS)

Advanced

Secure transfer	Enabled
Allow storage account key access	Enabled
Allow cross-tenant replication	Enabled
Default to Azure Active Directory authorization in the Azure portal	Disabled
Blob public access	Enabled
Minimum TLS version	Version 1.2
Permitted scope for copy operations (preview)	From any storage account
Enable hierarchical namespace	Disabled
Enable network file system v3	Disabled
Access tier	Hot
Enable SFTP	Disabled
Large file shares	Disabled

Networking

Network connectivity	Public endpoint (all networks)
Default routing tier	Microsoft network routing
Endpoint type	Standard

Create < Previous Next > Download a template for automation

Figure 8 - Storage account setup

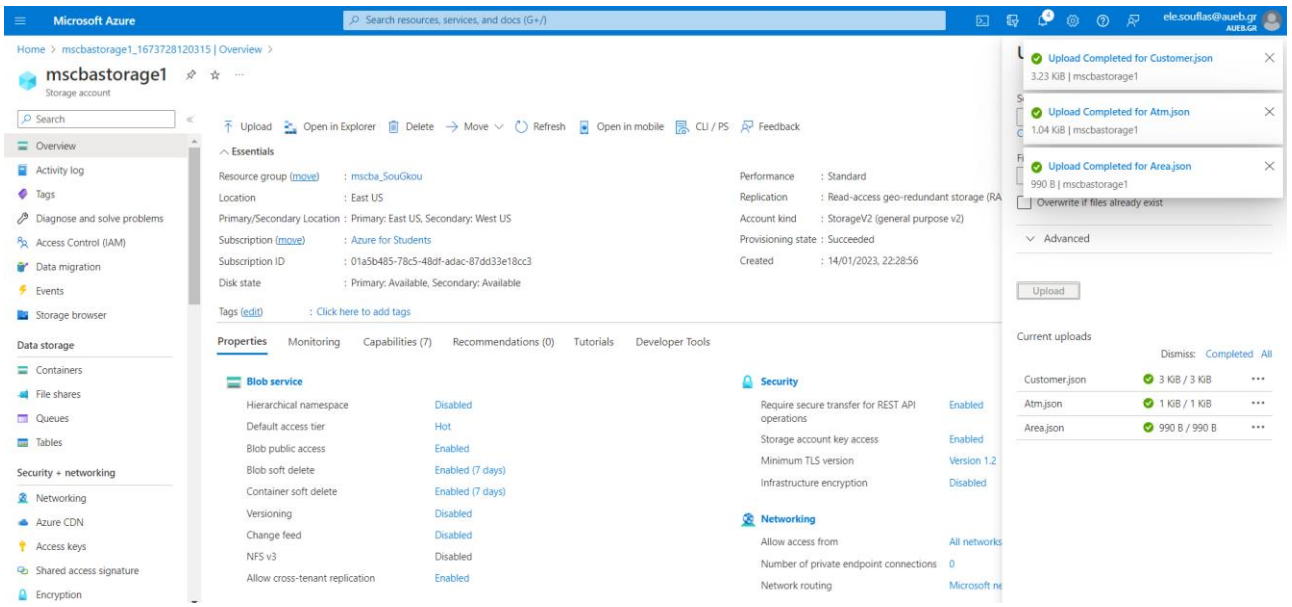

Figure 9 - Blob upload

The we created a Stream Analytics Job, which we will use for answering the Stream Queries (Figure 10). We used the Event Hub (Figure 11) and the Reference Data Files (Figure 13 and Figure 14) as a Stream input of the Job and a Blob Storage Container (we created a new container inside our Storage Account) as a Stream output (Figure 15) to store the results of the queries in JSON format. Before proceeding, we tested, by sampling data from the input source (Figure 12), that the Stream Input captures properly the data.

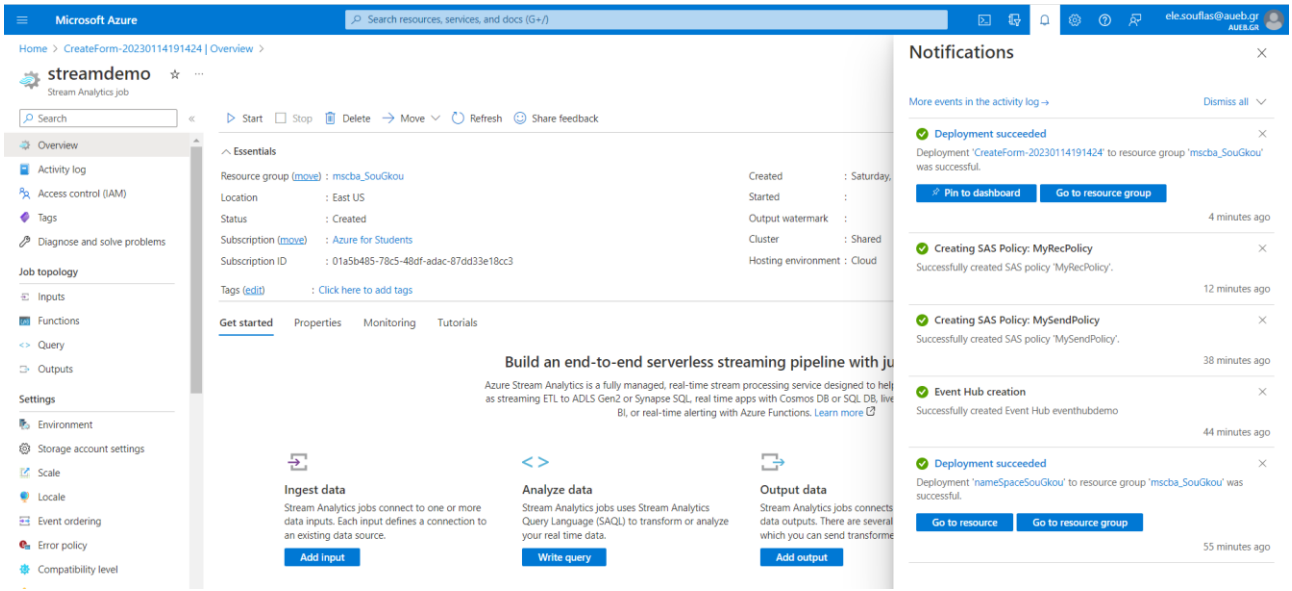

Figure 10 - Stream Analytics Job setup

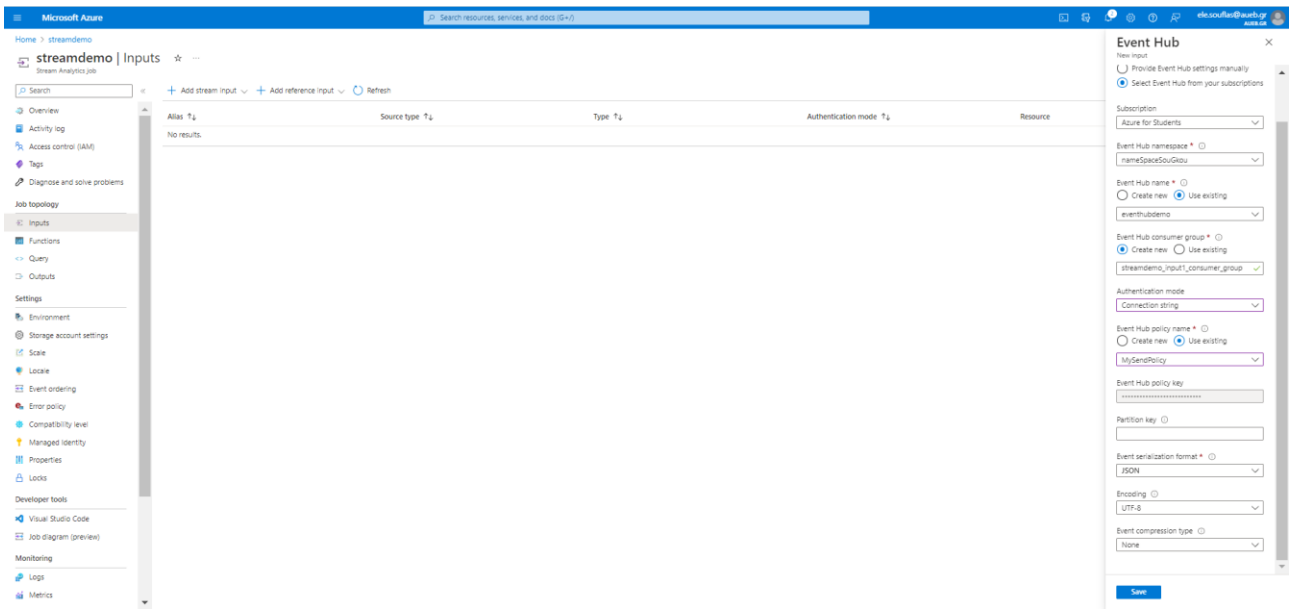

Figure 11 - Event Hub Stream Input

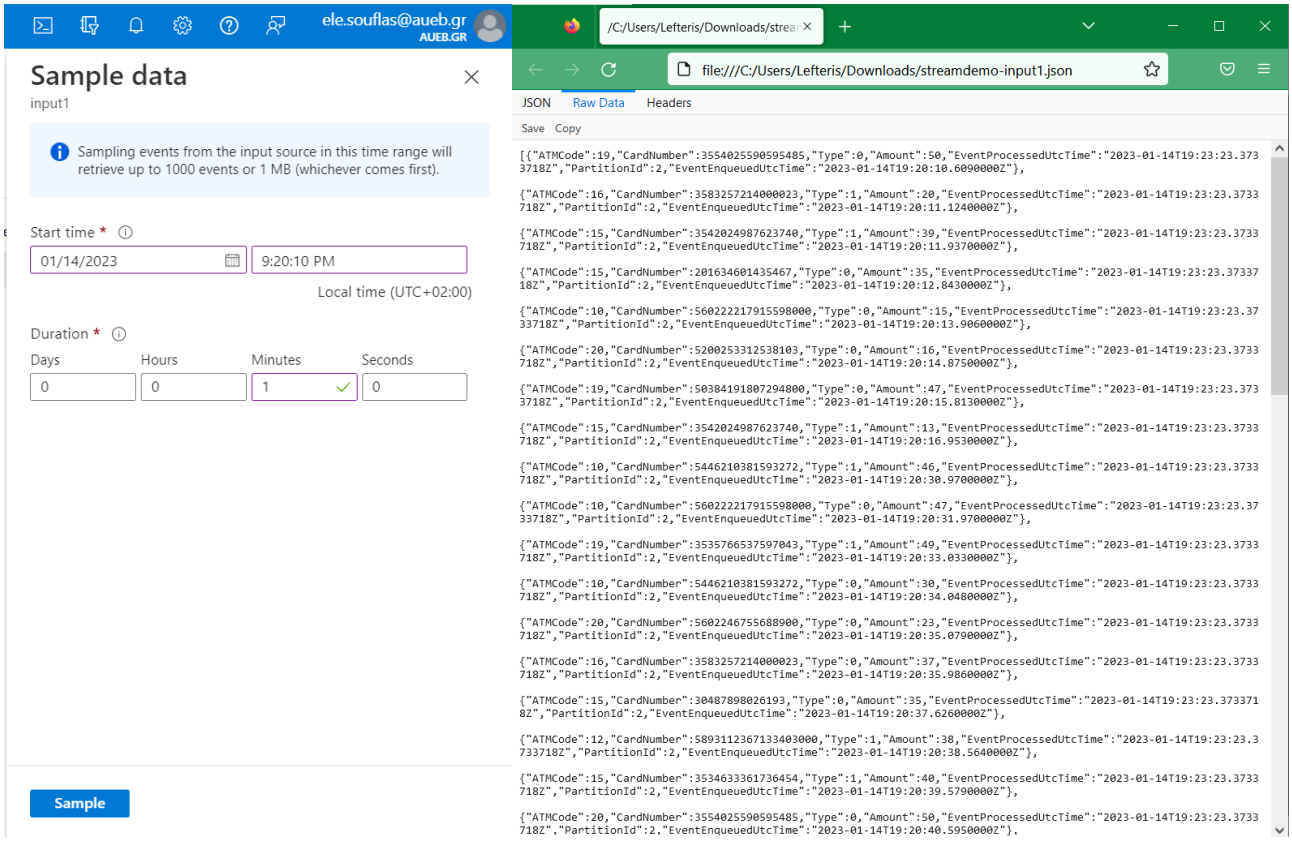

Figure 12 - Sample Input Data (JSON format)

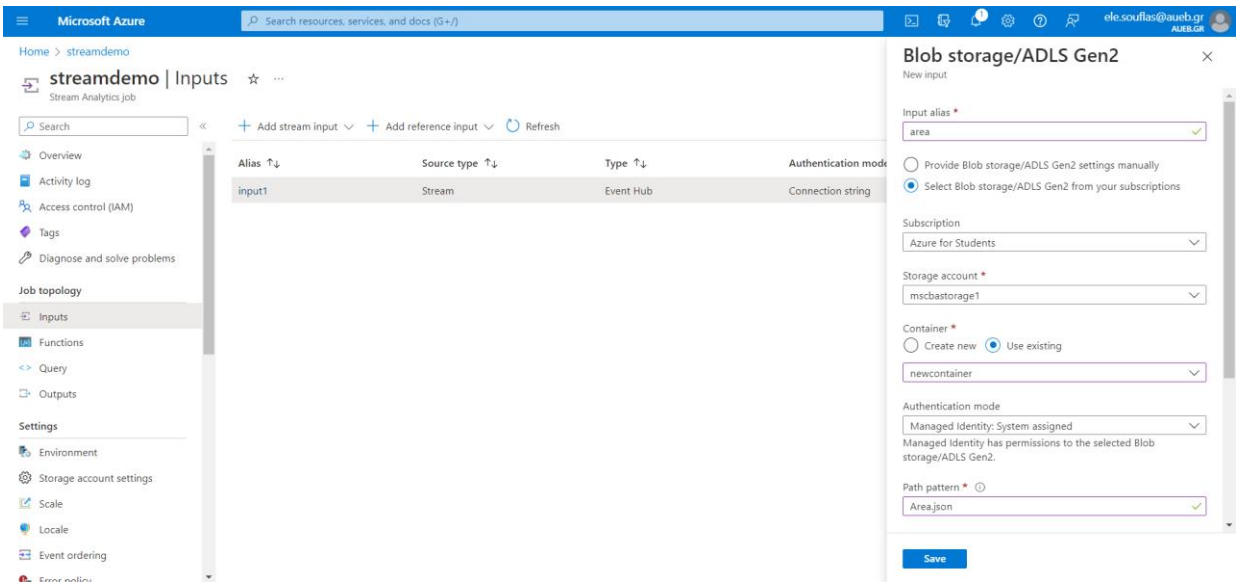

Figure 13 - Blob Storage Reference input creation

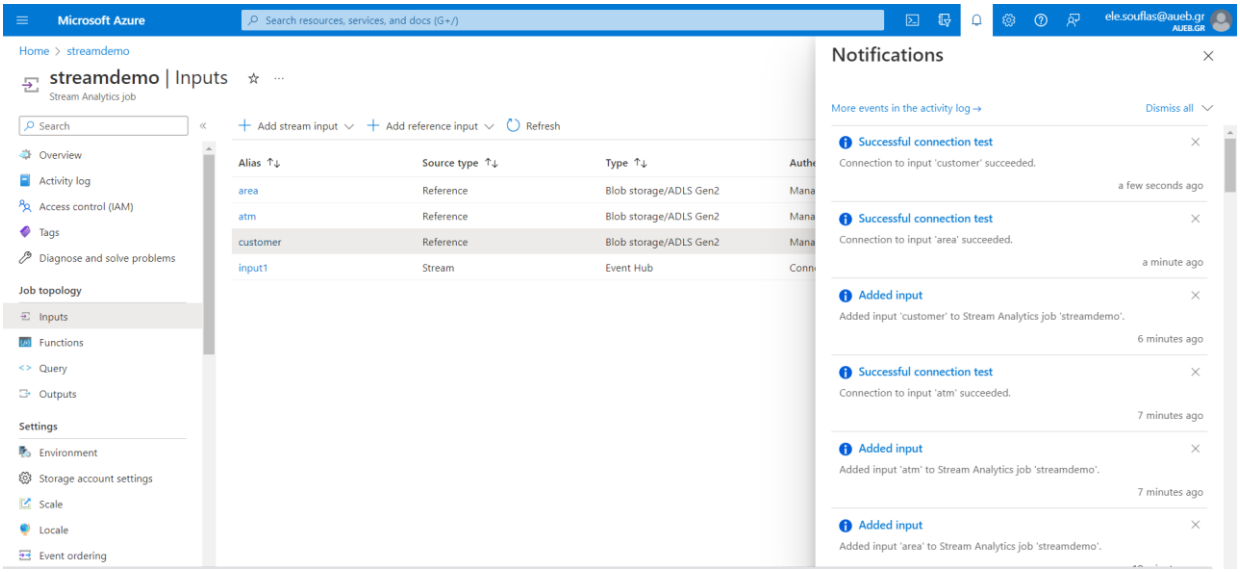

Figure 14 - All Blob Storage Reference inputs

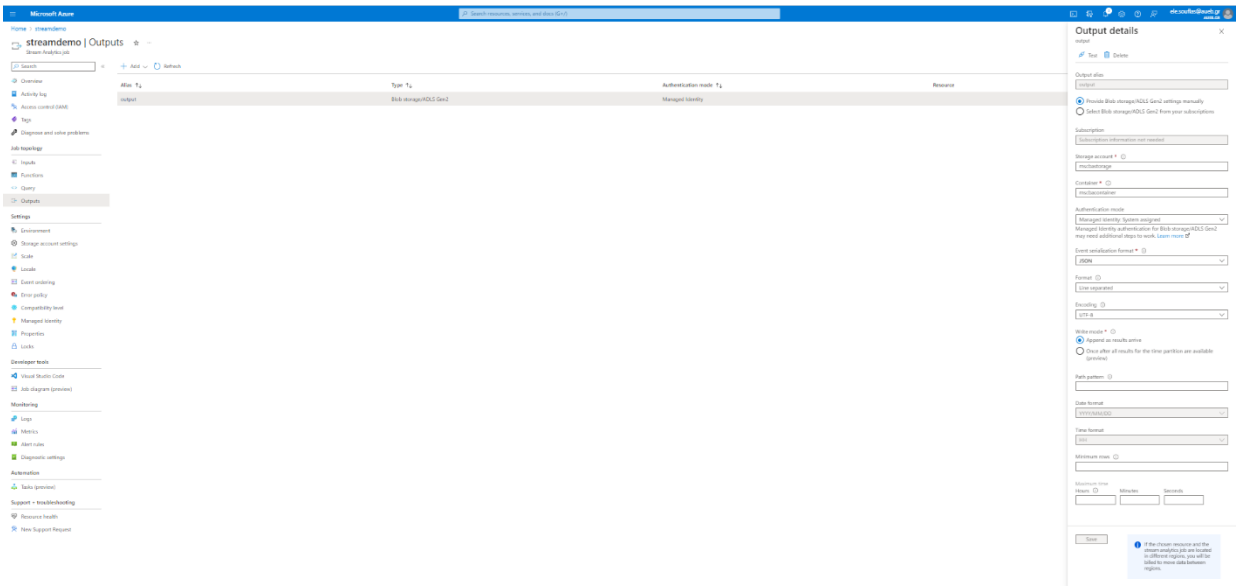

Figure 15 - Blob Storage Stream Output

Finally, we created a query that includes all possible input joins to test that all components used in the Stream Analytics Job work properly. From Figure 16, we can see that everything works as designed and expected.

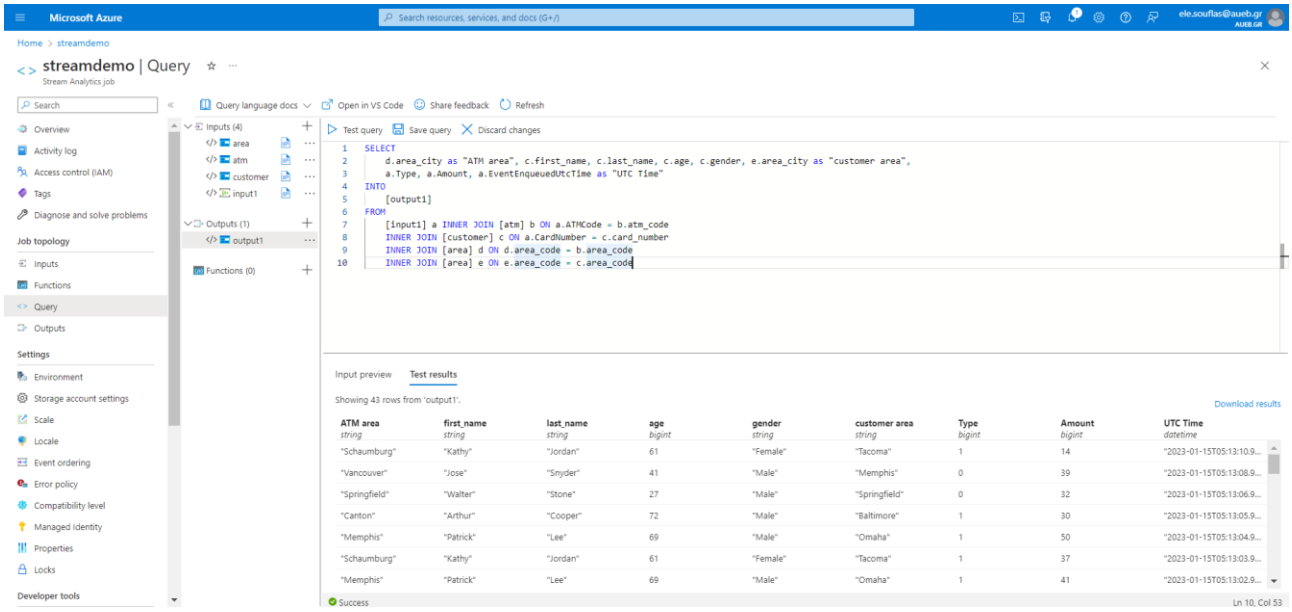

Figure 16 - Test Query with all possible joins

We are now ready to edit our 8 queries and collect our output data. To answer each query we followed the following procedure:

- We stopped previously started Stream Analytics Job.
- We edited the query.
- We tested it, refreshing data from input stream and uploading a sample input for each reference input used.
- We tested the connection and sampled data from input stream to test that everything worked fine for our inputs.
- We tested the connection to our output blob storage container.
- We started the Stream Analytics Job.
- We observed the output watermark datetime of the Stream Analytics Job Essentials (information) to be informed about the time our JSON output will be available in the Blob Storage Container.
- We clicked on the “Edit/View” of the JSON output to observe our output.

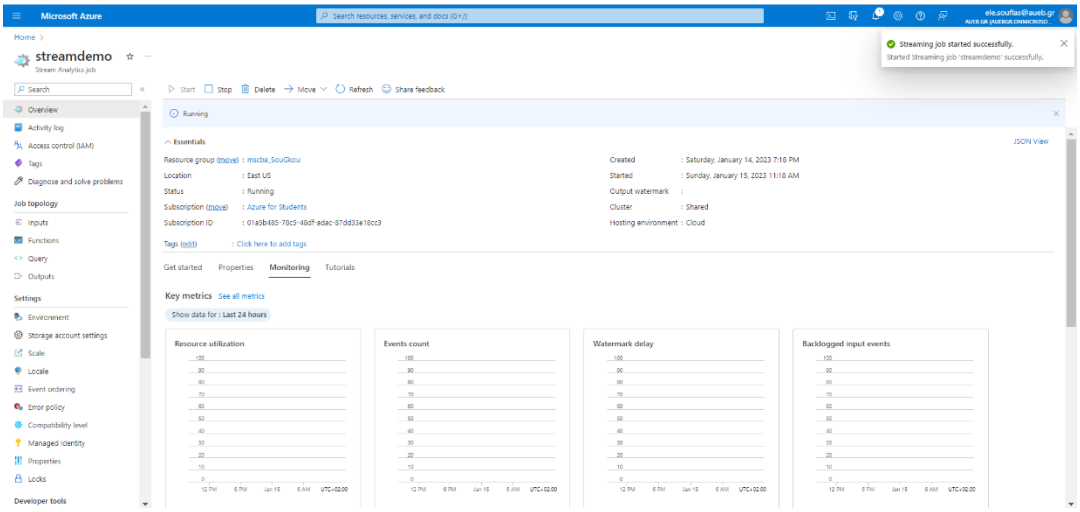

Figure 17 - Starting Stream Analytics Job

Now, we will provide screenshots of all answered 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).

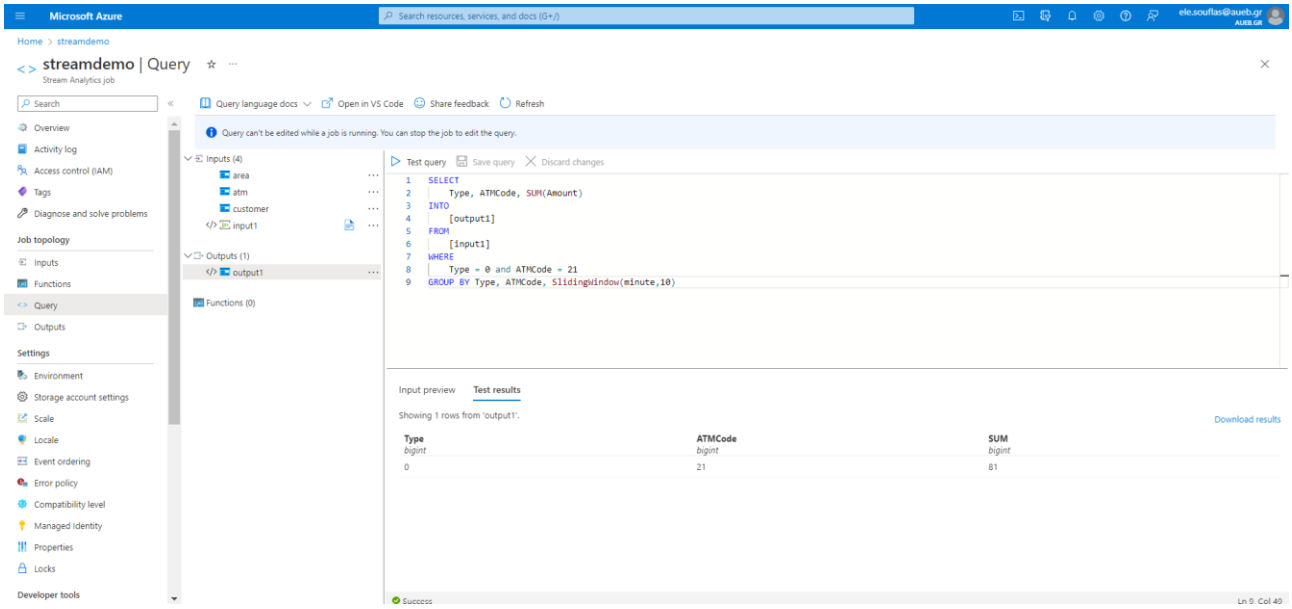

Figure 18 - Query 1 Test Results

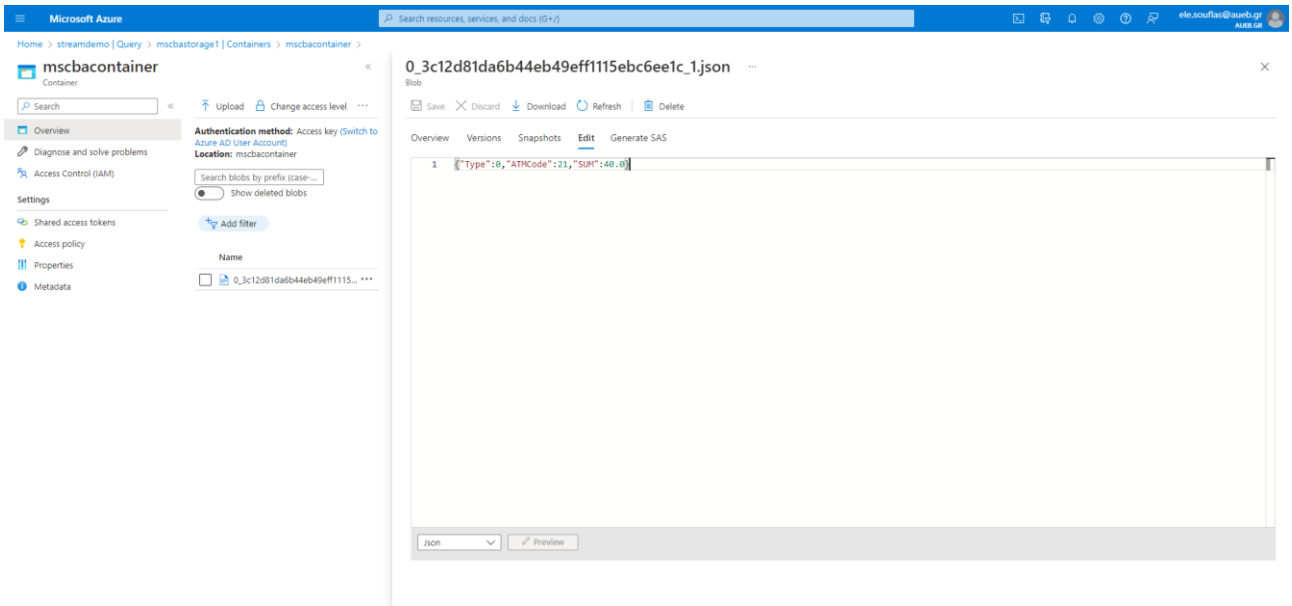

Figure 19 - Query 1 JSON Output

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).

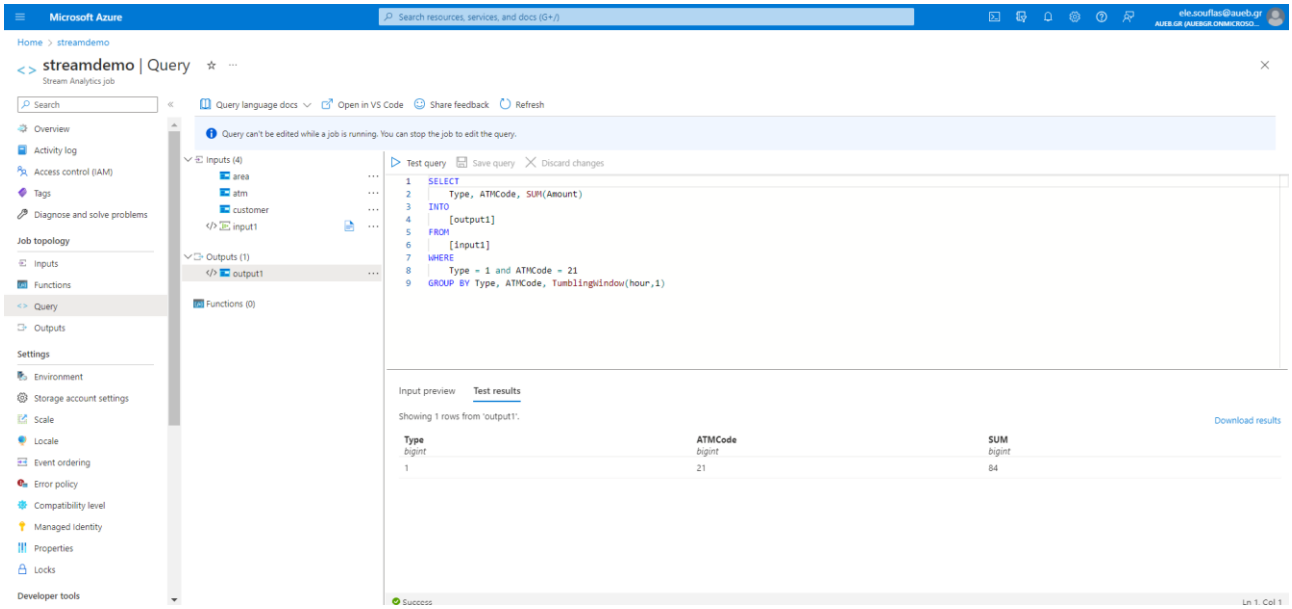

Figure 20 - Query 2 Test Results

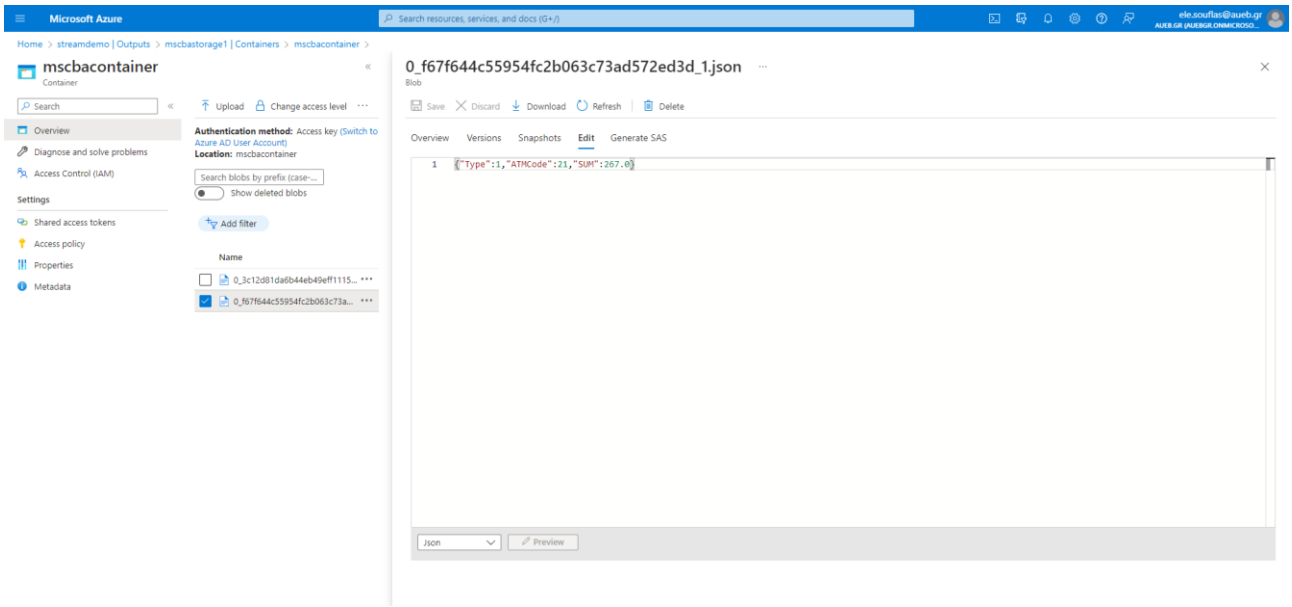

Figure 21 - Query 2 JSON Output

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).

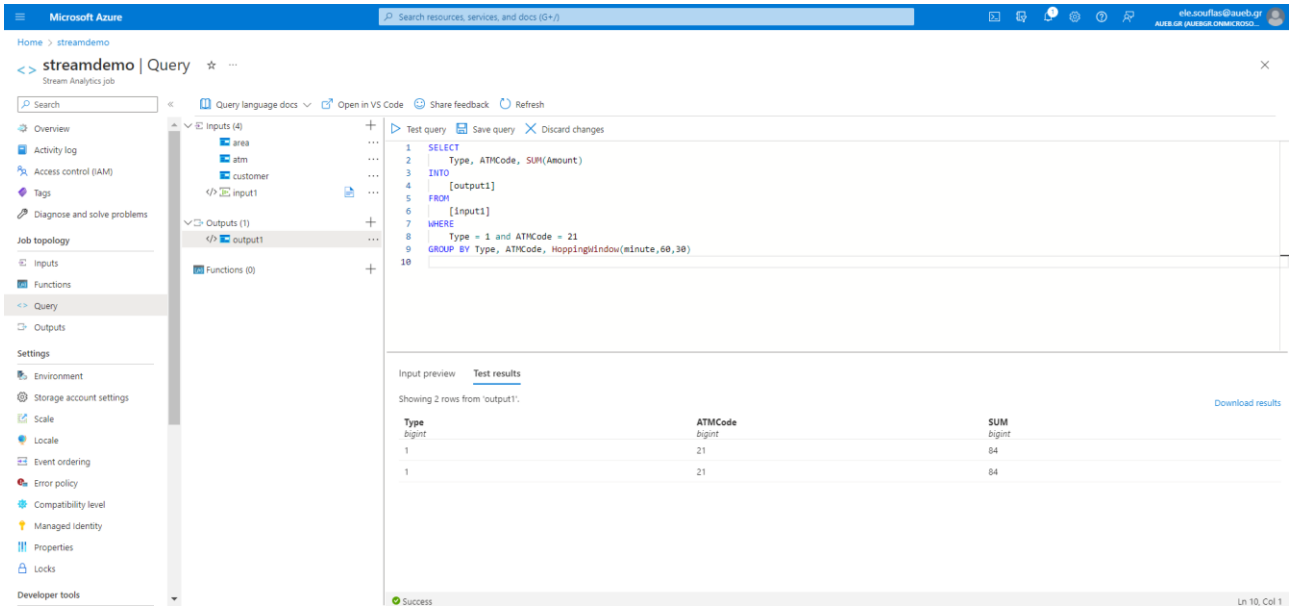

Figure 22 - Query 3 Test Results

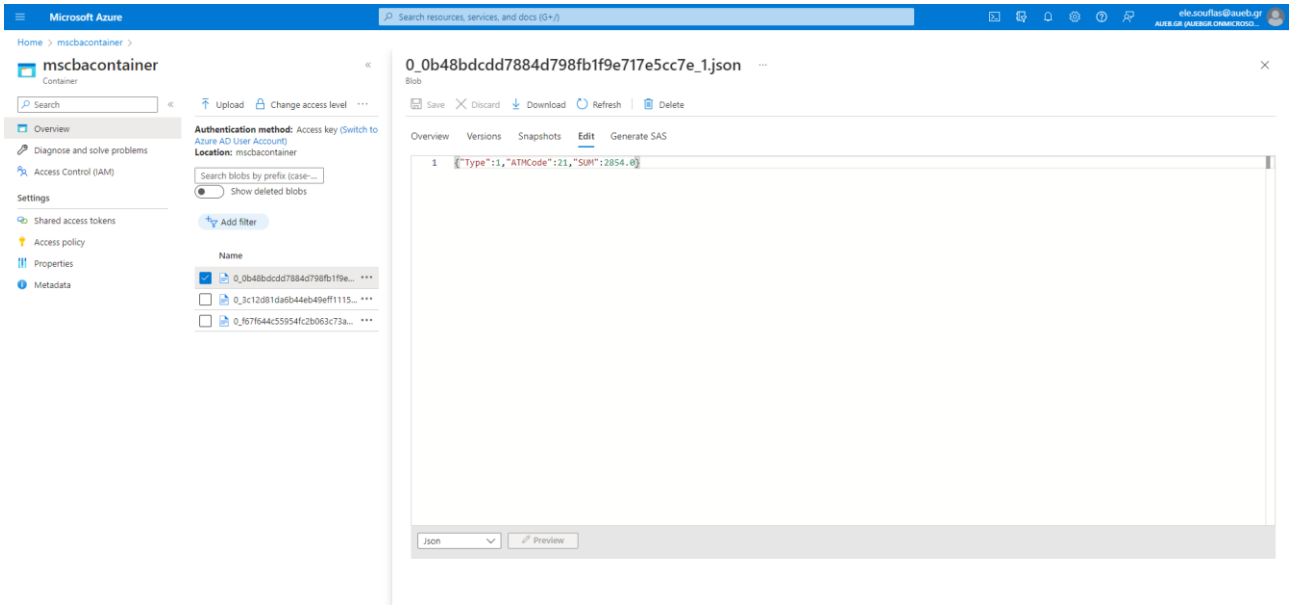

Figure 23 - Query 3 JSON Output

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

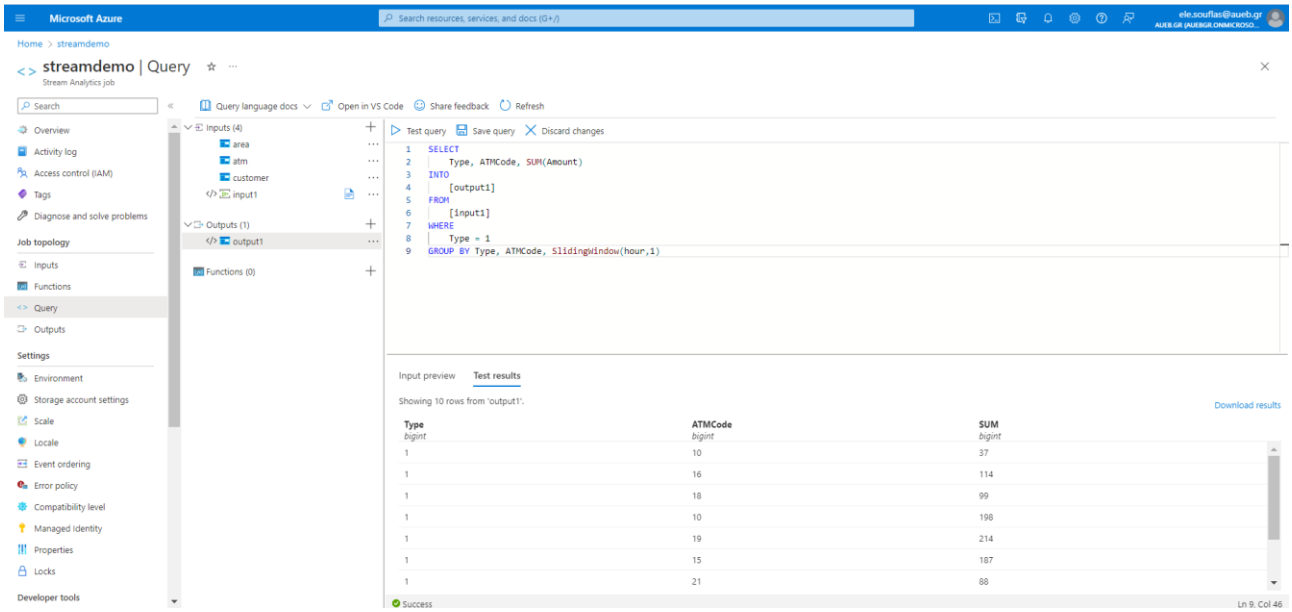

Figure 24 - Query 4 Test Results

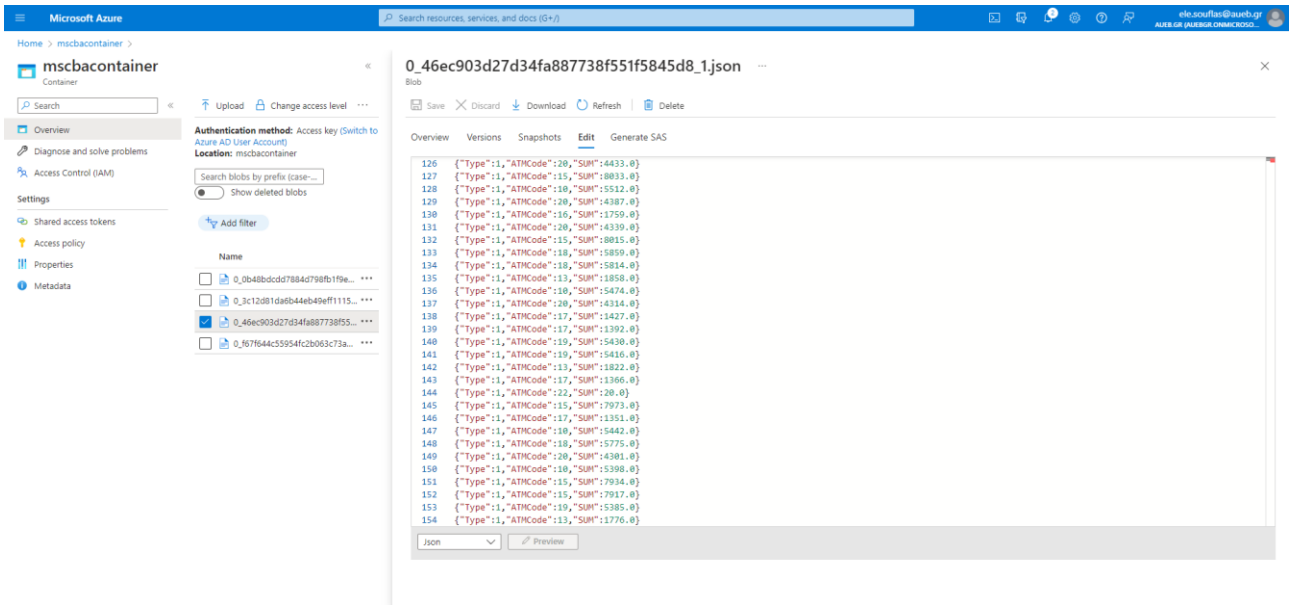

Figure 25 - Query 4 JSON Output

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).

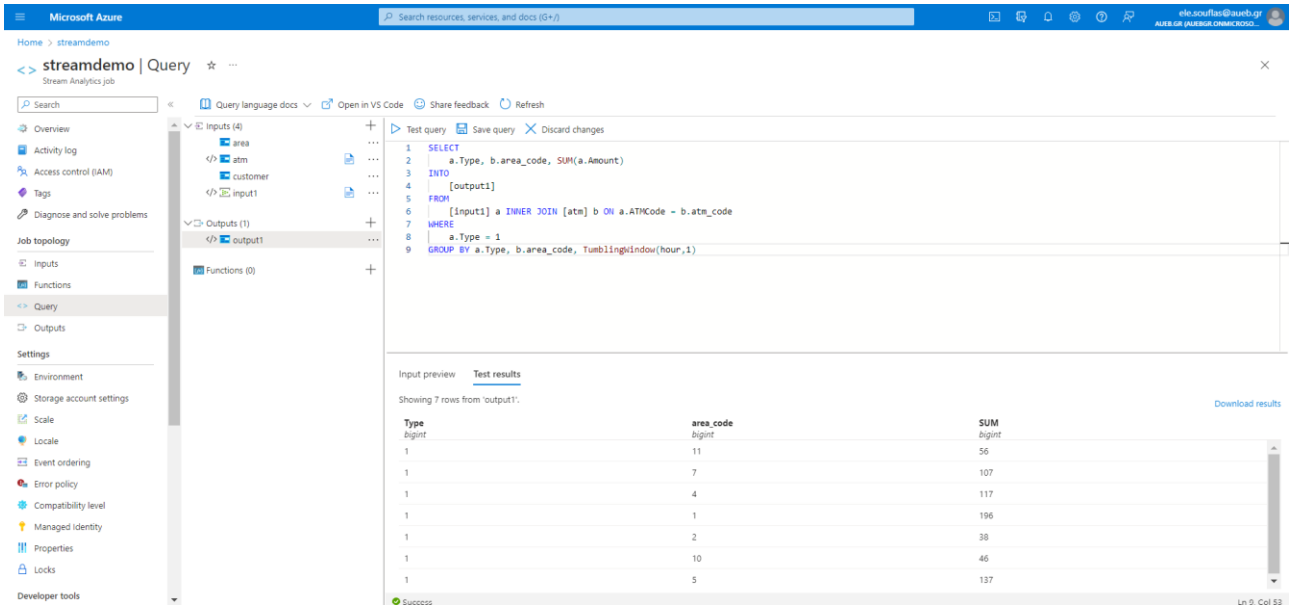

Figure 26 - Query 5 Test Results

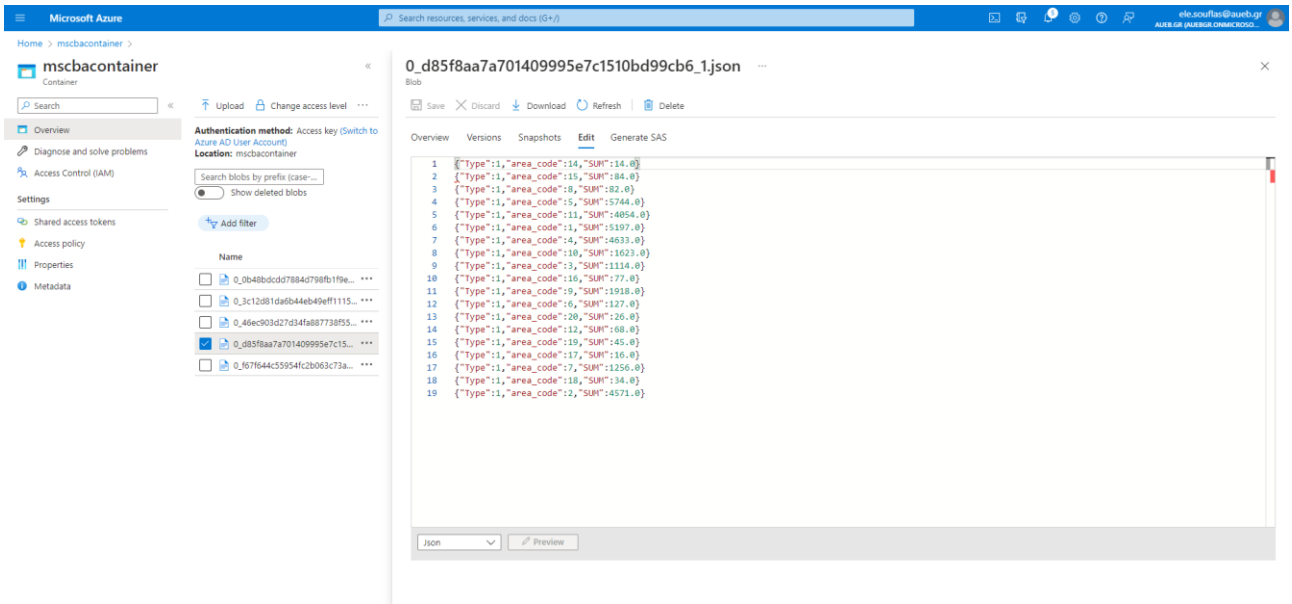

Figure 27 - Query 5 JSON Output

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).

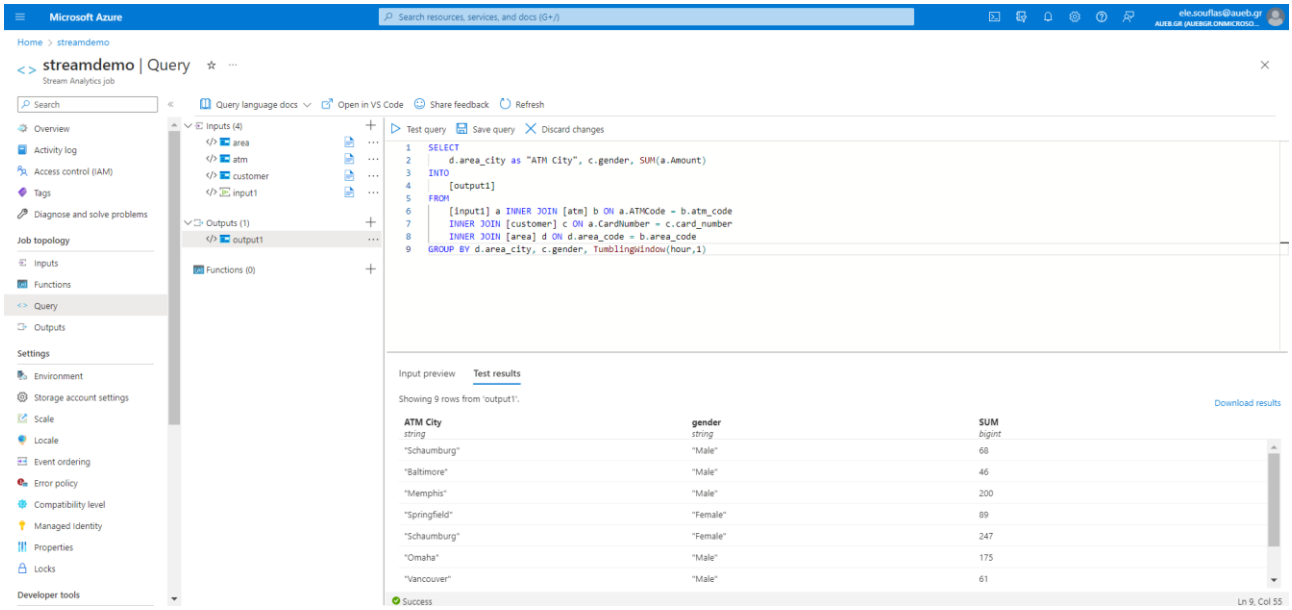

Figure 28 - Query 6 Test Results

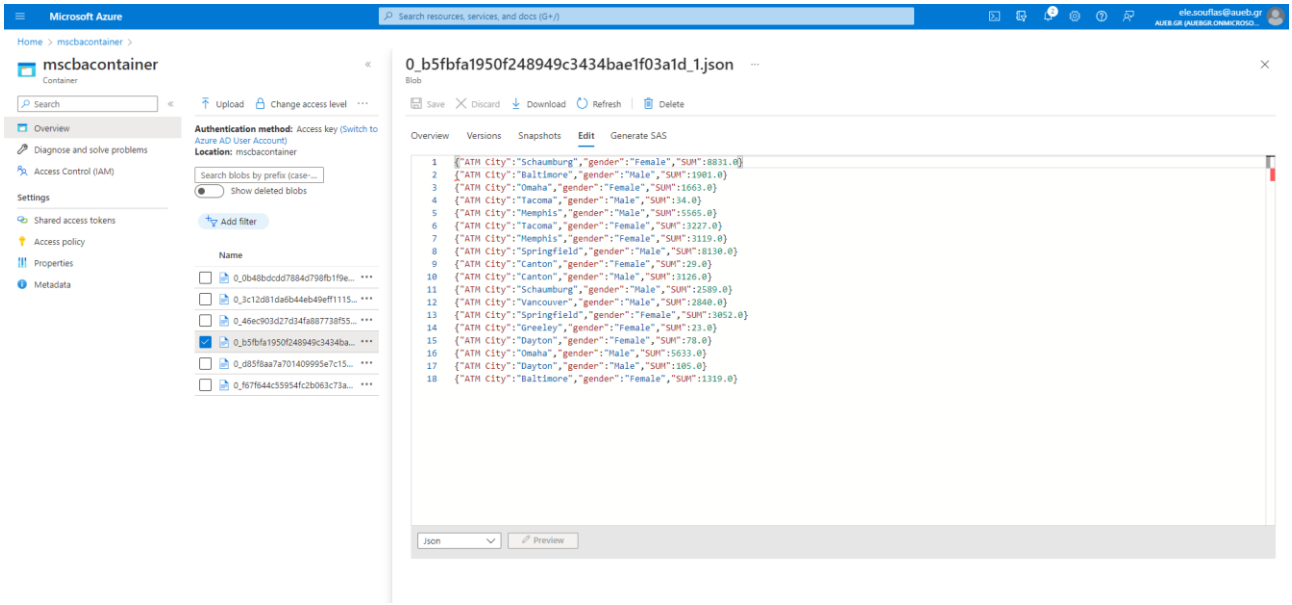

Figure 29 - Query 6 JSON Output

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).

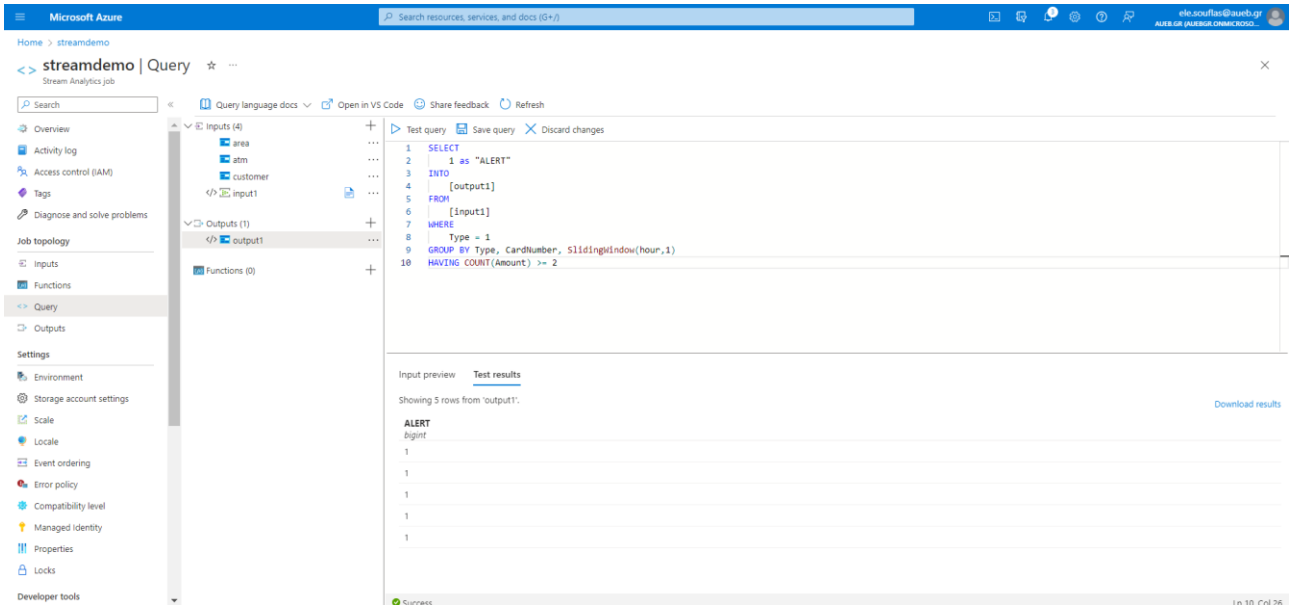

Figure 30 - Query 7 Test Results (COUNT >= 2 has been used and not COUNT=2 because a customer with 3 transactions has also performed 2 transactions in the same sliding window)

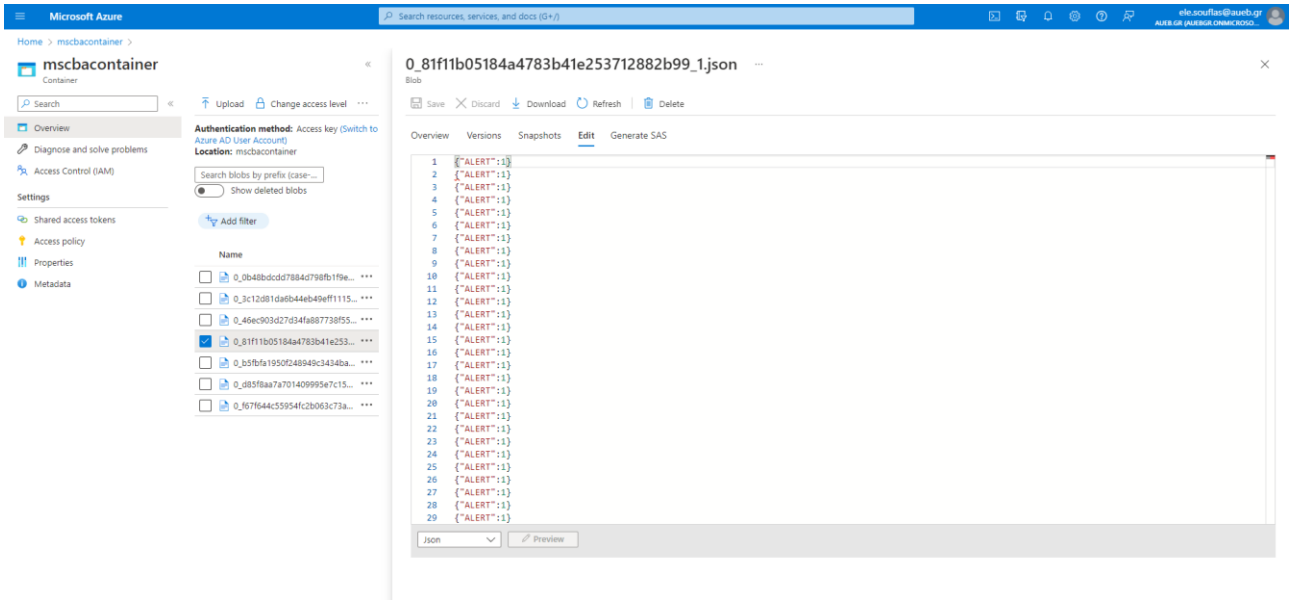

Figure 31 - Query 7 JSON Output

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)

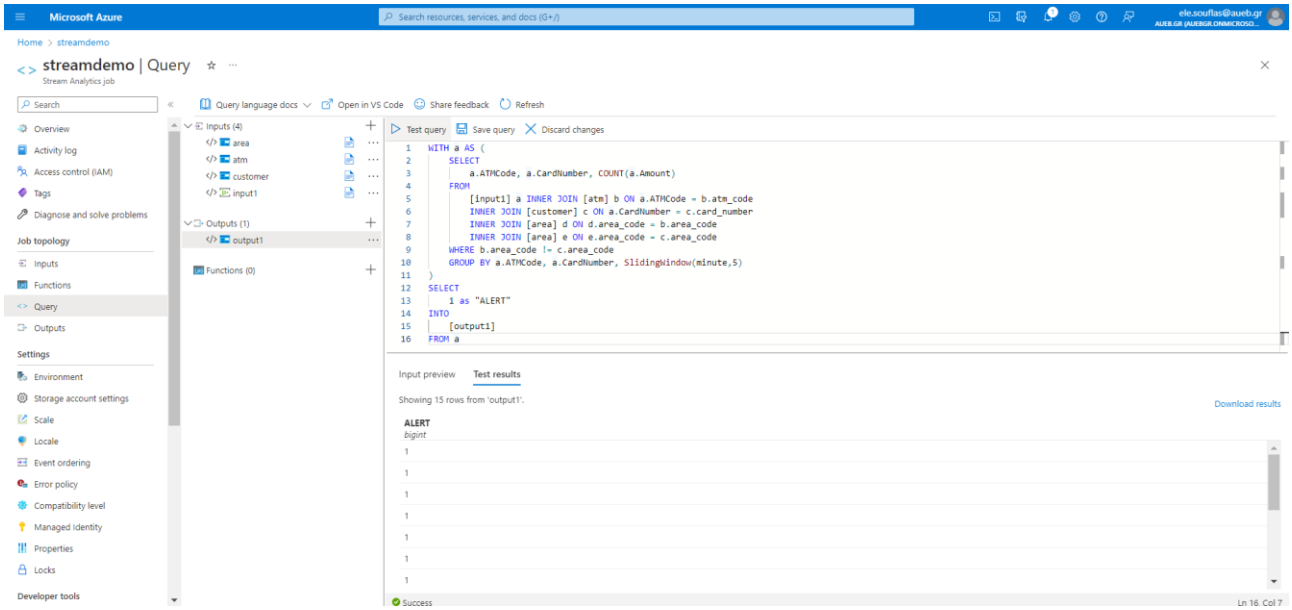

Figure 32 - Query 8 Test Results (a sliding window of 5 minutes has been arbitrarily used because it was not mentioned otherwise)

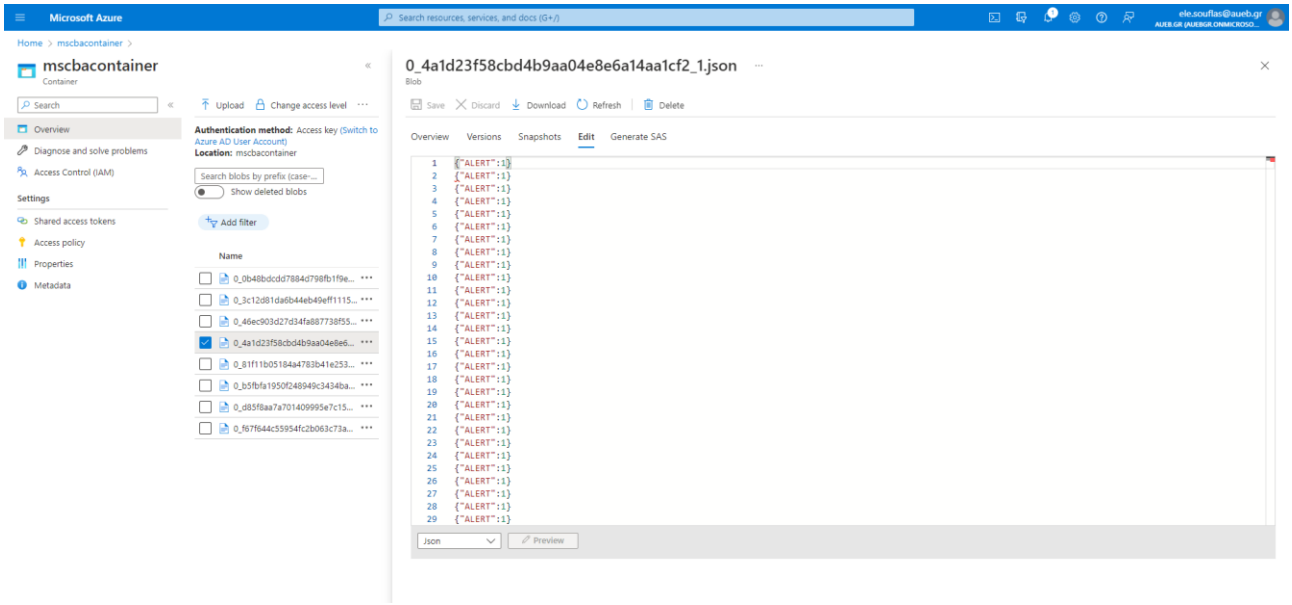

Figure 33 - Query 8 JSON Output