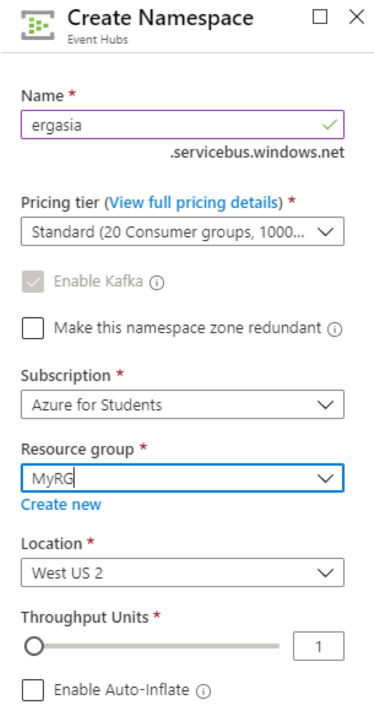
Prerequisites

Before writing and testing SQL stream queries in Azure Stream Analytics we had to follow some necessary steps. At first, we created an event hub namespace, which serves as an application container that can house multiple Event Hub topics. Then we created an event hub within the namespace and two access policies for the event hub, which will be used as sender (**MySendPolicy**) and listener (**MyRecPolicy**) to this Event Hub. Between these policies we generated a security access signature from our first policy (**MySendPolicy**) and used this along with our event hubs information to generate sample data and send it to Azure Event Hubs. Screenshots per step are presented below:



Create Namespace
Event Hubs

Name * ergasia .servicebus.windows.net

Pricing tier (View full pricing details) * Standard (20 Consumer groups, 1000...

☑ Enable Kafka

□ Make this namespace zone redundant

Subscription * Azure for Students

Resource group * MyRG

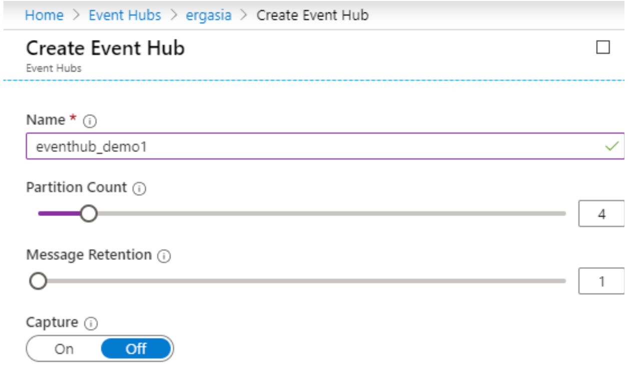
Create new

Location * West US 2

Throughput Units * 1

□ Enable Auto-Inflate

Namespace creation



Home > Event Hubs > ergasia > Create Event Hub

Create Event Hub
Event Hubs

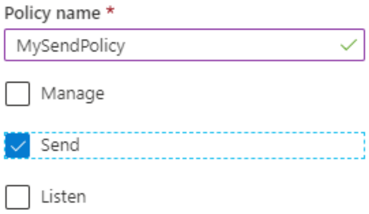
Name * eventhub_demo1

Partition Count 4

Message Retention 1

Capture On [✓ Off]

Event hub creation



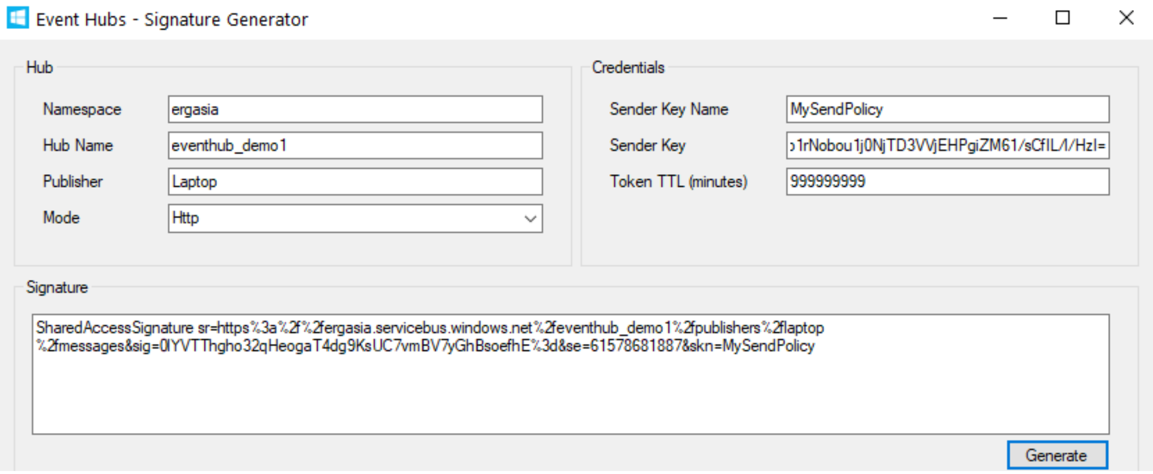
Policy name * MySendPolicy

□ Manage

☑ Send

□ Listen

Sender policy



Event Hubs - Signature Generator

Hub

Namespace: ergasia

Hub Name: eventhub_demo1

Publisher: Laptop

Mode: Http

Credentials

Sender Key Name: MySendPolicy

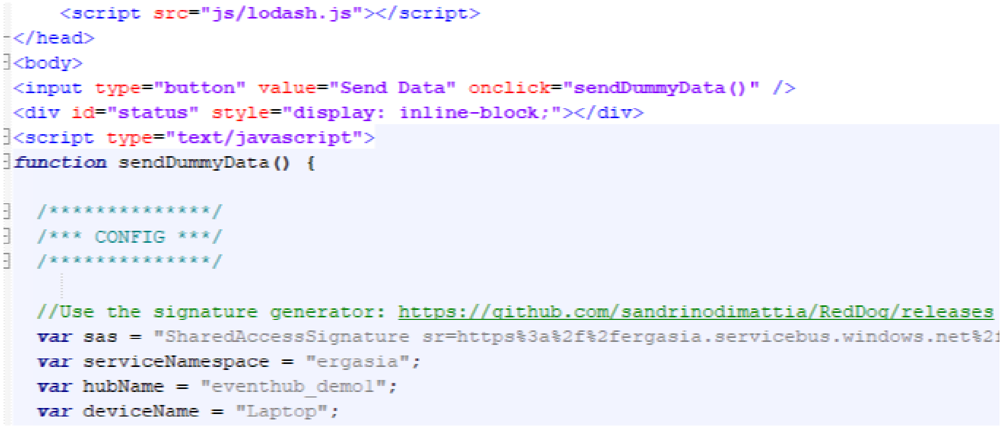
Sender Key: 1rNobou1j0NjTD3VVjEHPgiZM61/sCfIL/l/HzI=

Token TTL (minutes): 999999999

Signature

```
SharedAccessSignature sr=https%3a%2f%2fergasia.servicebus.windows.net%2feventhub_demo1%2fpublishers%2flaptop
%2fmessages&sig=0lYVTThgho32qHeogaT4dg9KsUC7vmBV7yGhBsoefhE%3d&se=61578681887&skn=MySendPolicy
```

Signature generator

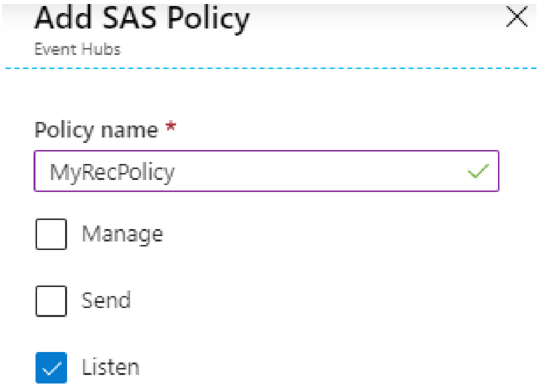


```
    <script src="js/lodash.js"></script>
</head>
<body>
<input type="button" value="Send Data" onclick="sendDummyData()" />
<div id="status" style="display: inline-block;"></div>
<script type="text/javascript">
function sendDummyData() {

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

  //Use the signature generator: https://github.com/sandrinodimattia/RedDog/releases
  var sas = "SharedAccessSignature sr=https%3a%2f%2fergasia.servicebus.windows.net%2f
  var serviceNamespace = "ergasia";
  var hubName = "eventhub_demo1";
  var deviceName = "Laptop";
```

*The created signature has been used in DataGenerator python code (check variable **sas**)*



Add SAS Policy

Event Hubs

Policy name *

MyRecPolicy

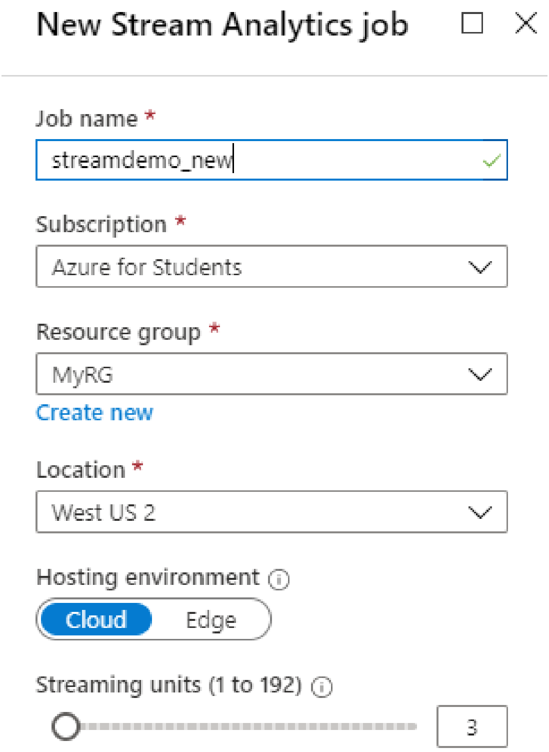
□ Manage

□ Send

☑ Listen

Listener policy

After completing the creation of the data stream, we created a Stream Analytics job that reads data from the event hub. Then we defined an input source for the job in order to read data using the event hub we created in the previous section and also an output sink for the job where it can write the transformed data.



New Stream Analytics job

Job name *

streamdemo_new

Subscription *

Azure for Students

Resource group *

MyRG

Location *

West US 2

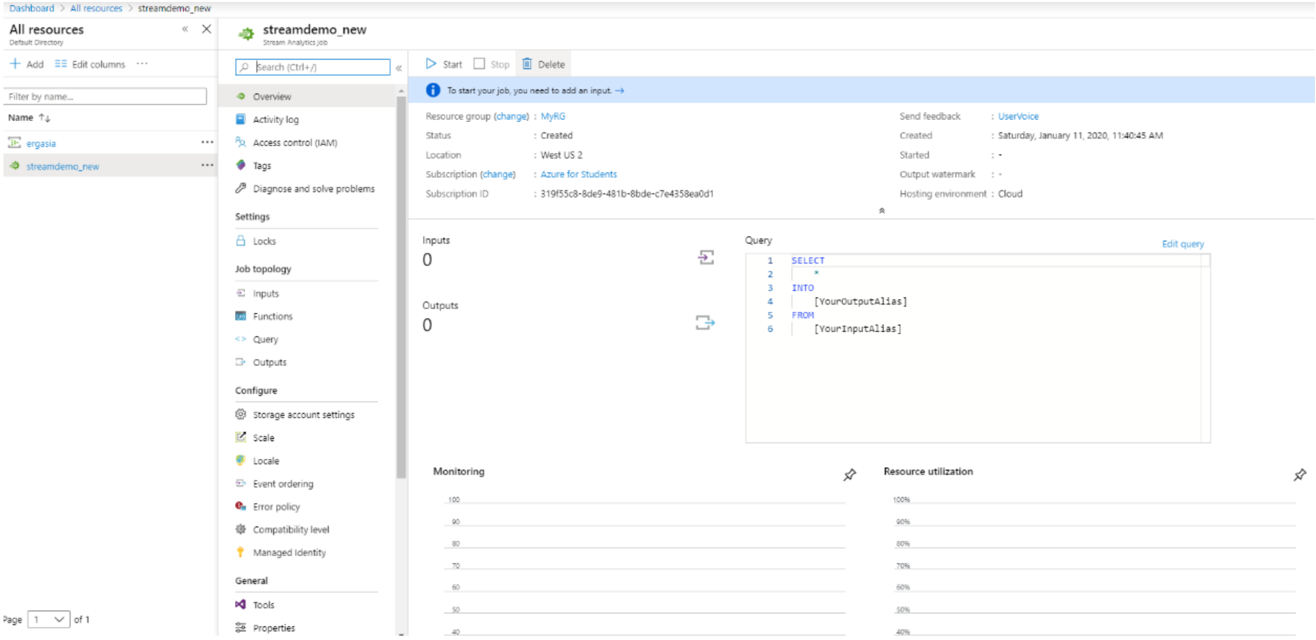
Hosting environment

Cloud Edge

Streaming units (1 to 192)

3

Stream Analytics job creation



To start your job, you need to add an input. →

Resource group (change) : MyRG
Status : Created
Location : West US 2
Subscription (change) : Azure for Students
Subscription ID : 319f55c8-8de9-481b-8bde-c7e4358ea0d1

Send feedback : UserVoice
Created : Saturday, January 11, 2020, 11:40:45 AM
Started : -
Output watermark : -
Hosting environment : Cloud

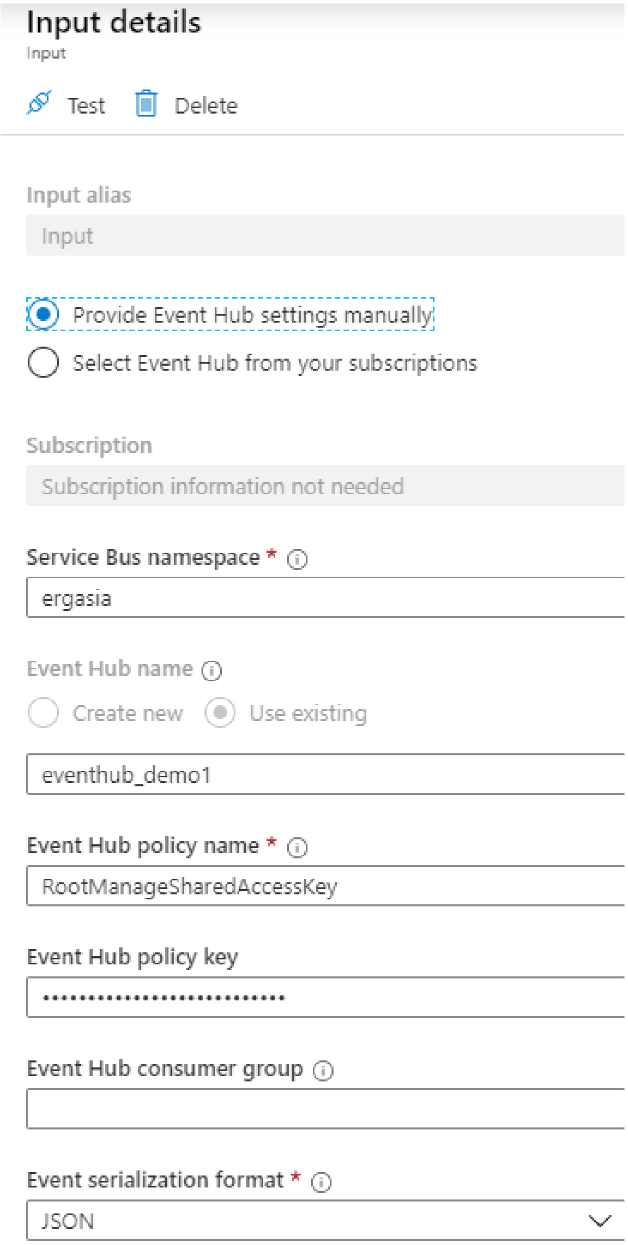
Inputs
0

Outputs
0

Query

```
SELECT
    *
INTO
    [YourOutputAlias]
FROM
    [YourInputAlias]
```

Stream analytics job main view



Input details

Input

Test Delete

Input alias

Input

● Provide Event Hub settings manually

○ Select Event Hub from your subscriptions

Subscription

Subscription information not needed

Service Bus namespace *

ergasia

Event Hub name

○ Create new ● Use existing

eventhub_demo1

Event Hub policy name *

RootManageSharedAccessKey

Event Hub policy key

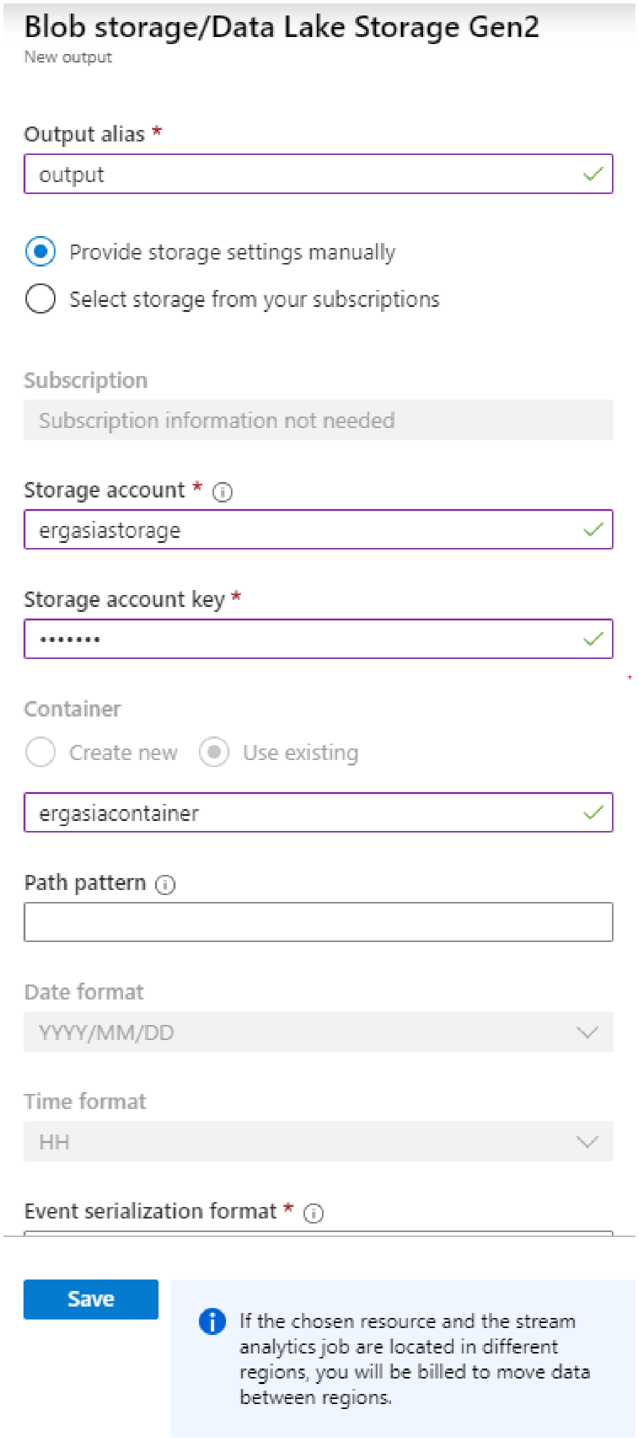
•••••••••••••••••••••••••••

Event Hub consumer group

Event serialization format *

JSON

Input event hub creation



Blob storage/Data Lake Storage Gen2

New output

Output alias *

output ✓

● Provide storage settings manually

○ Select storage from your subscriptions

Subscription

Subscription information not needed

Storage account *

ergasiastorage ✓

Storage account key *

••••••• ✓

Container

○ Create new ● Use existing

ergasiacontainer ✓

Path pattern

Date format

YYYY/MM/DD

Time format

HH

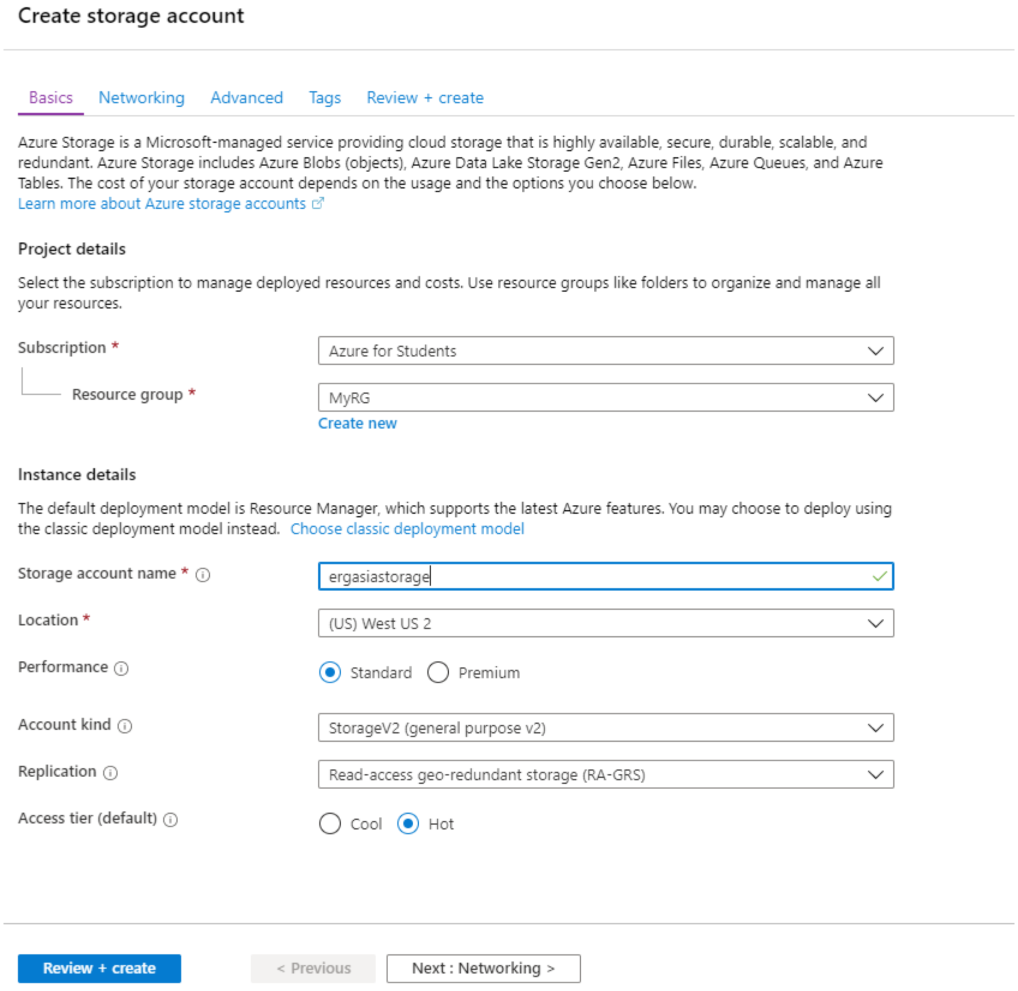
Event serialization format *

Save

If the chosen resource and the stream analytics job are located in different regions, you will be billed to move data between regions.

Output event hub creation

Another step was to create a storage account, while we were creating containers for each file of our reference data. Before doing so, we decided to convert all our files type to JSON. Another data modification was to add a header ("*licencePlate*") in wanted_cars json file in order to handle better the data input and also for our assistance while writing the queries.



Create storage account

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below.

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * : Azure for Students

Resource group * : MyRG

Instance details

The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using the classic deployment model instead.

Storage account name * : ergasiastorage ✓

Location * : (US) West US 2

Performance: ● Standard ○ Premium

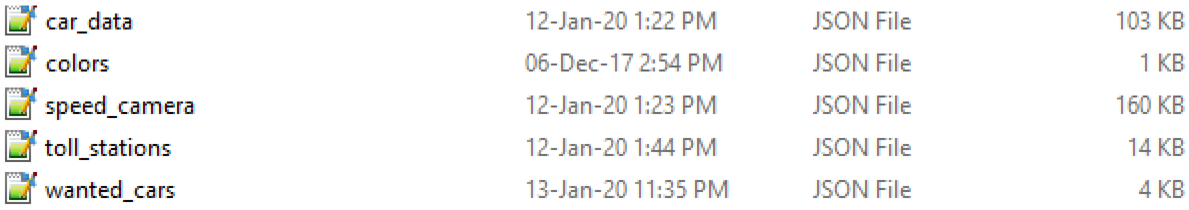
Account kind: StorageV2 (general purpose v2)

Replication: Read-access geo-redundant storage (RA-GRS)

Access tier (default): ○ Cool ● Hot

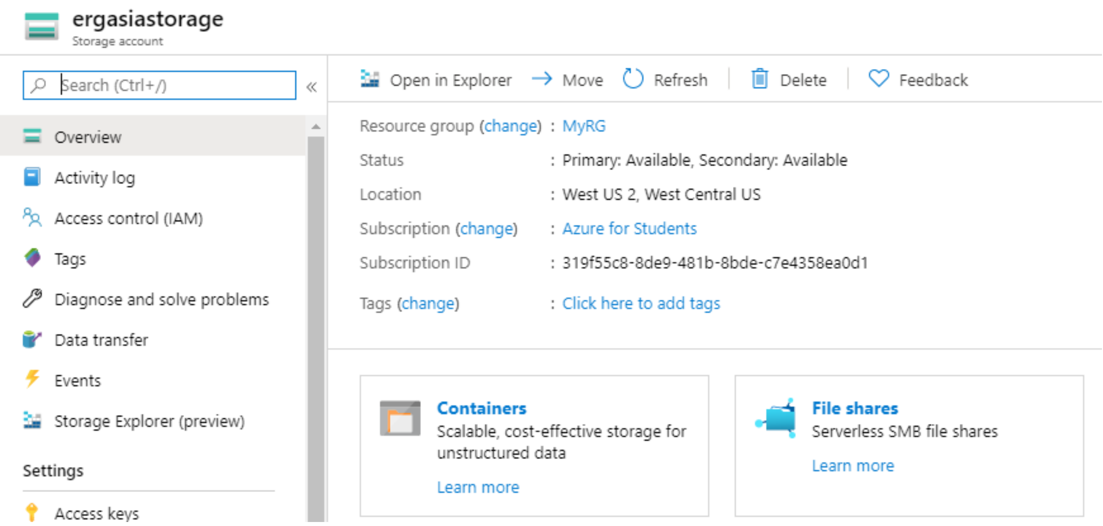
Storage account creation

Files converted to JSON



car_data	12-Jan-20 1:22 PM	JSON File	103 KB
colors	06-Dec-17 2:54 PM	JSON File	1 KB
speed_camera	12-Jan-20 1:23 PM	JSON File	160 KB
toll_stations	12-Jan-20 1:44 PM	JSON File	14 KB
wanted_cars	13-Jan-20 11:35 PM	JSON File	4 KB

Converted input reference data files

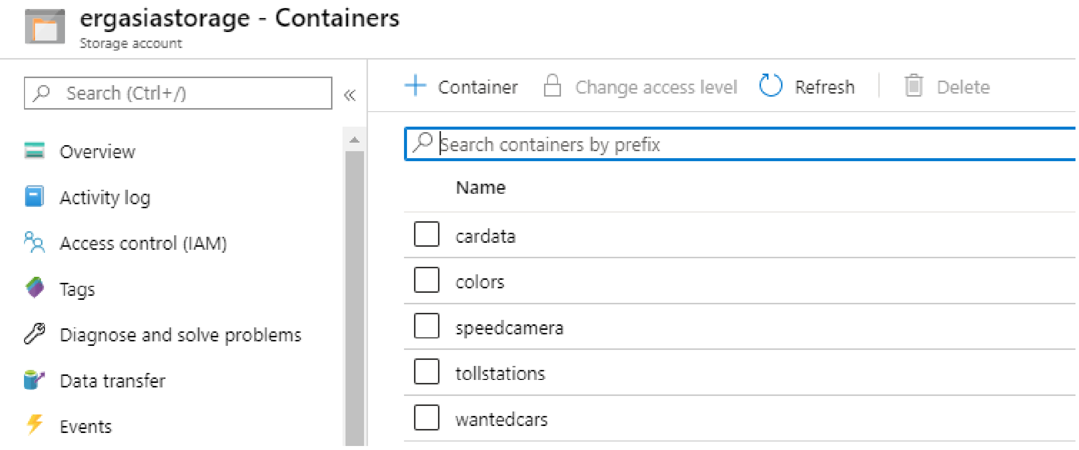


Resource group (change) : MyRG
Status : Primary: Available, Secondary: Available
Location : West US 2, West Central US
Subscription (change) : Azure for Students
Subscription ID : 319f55c8-8de9-481b-8bde-c7e4358ea0d1
Tags (change) : Click here to add tags

Containers
Scalable, cost-effective storage for unstructured data
Learn more

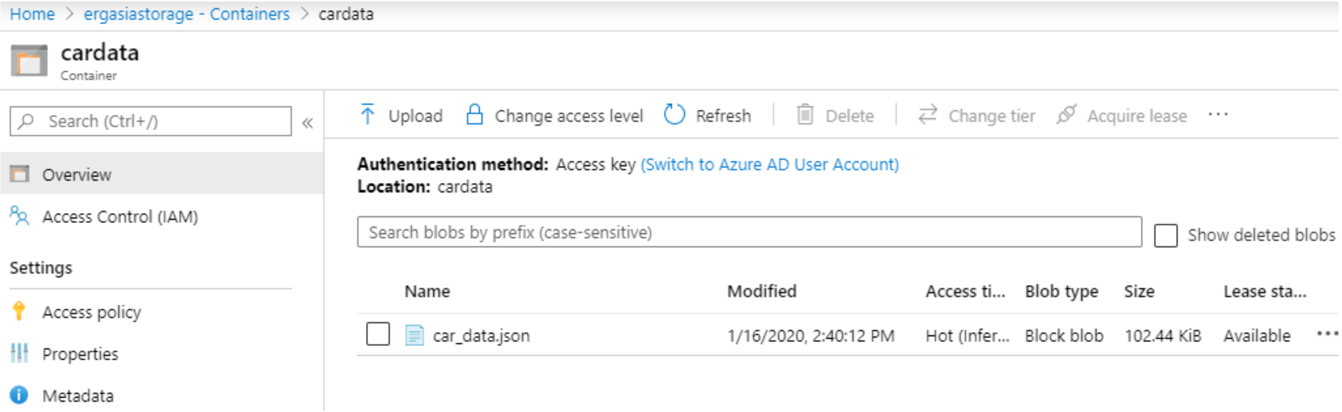
File shares
Serverless SMB file shares
Learn more

Storage main view



	Name
□	cardata
□	colors
□	speedcamera
□	tollstations
□	wantedcars

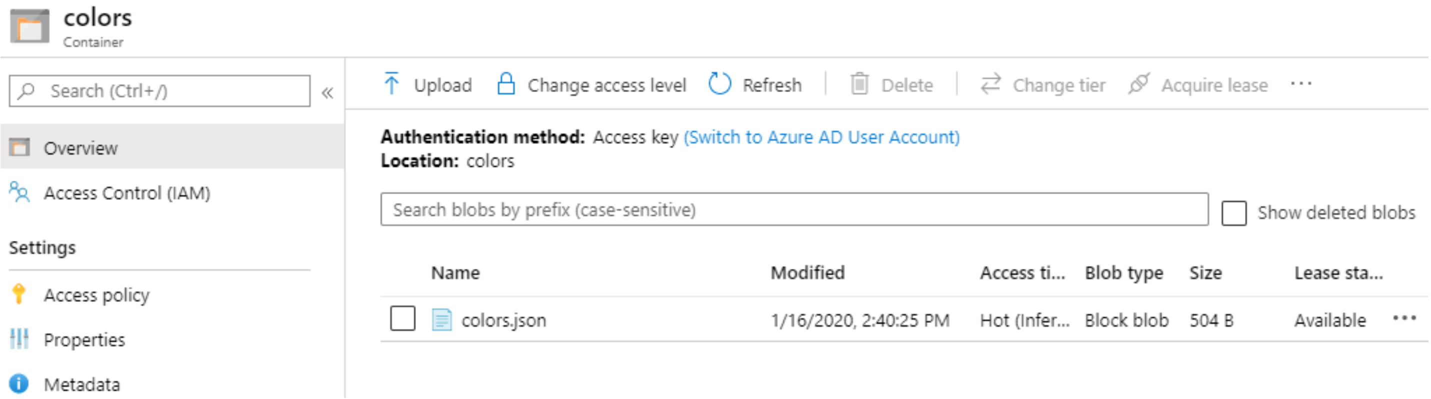
Storage containers view (reference data files)



Authentication method: Access key (Switch to Azure AD User Account)
Location: cardata

	Name	Modified	Access ti...	Blob type	Size	Lease sta...
□	car_data.json	1/16/2020, 2:40:12 PM	Hot (Infer...	Block blob	102.44 KiB	Available

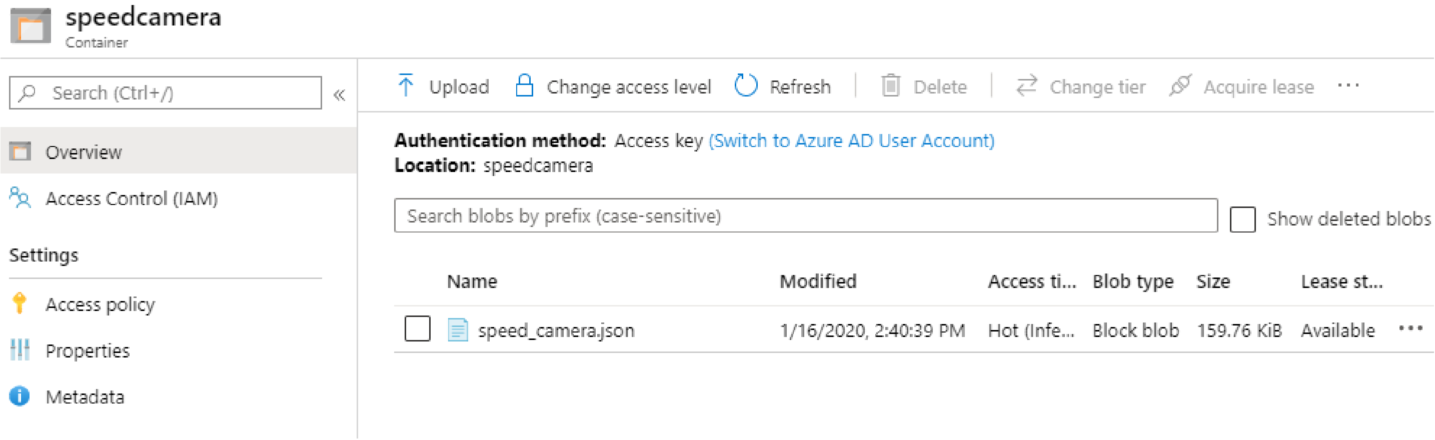
Car data container



Authentication method: Access key (Switch to Azure AD User Account)
Location: colors

	Name	Modified	Access ti...	Blob type	Size	Lease sta...	
□	colors.json	1/16/2020, 2:40:25 PM	Hot (Infer...	Block blob	504 B	Available	•••

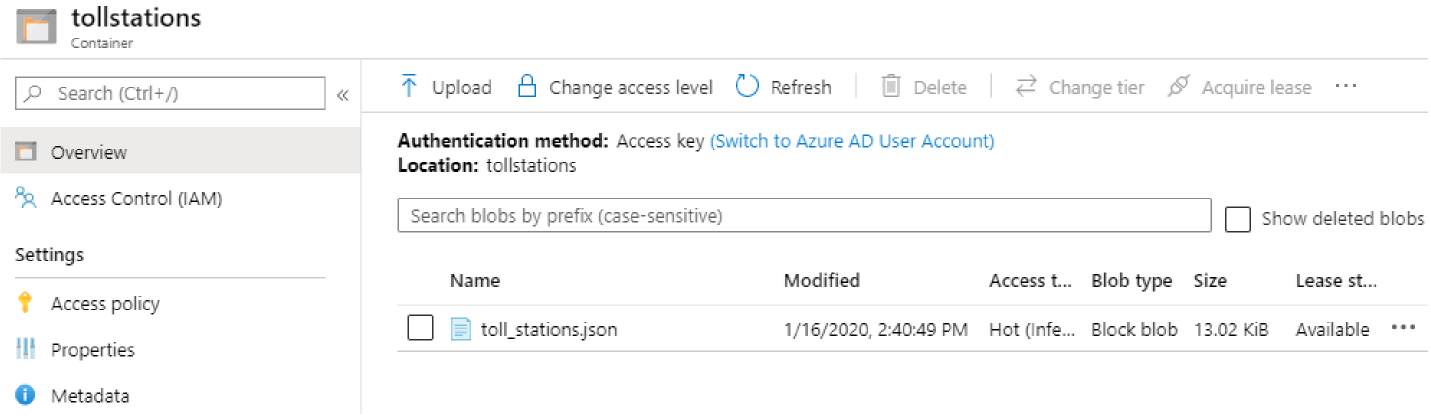
Colors container



Authentication method: Access key (Switch to Azure AD User Account)
Location: speedcamera

	Name	Modified	Access ti...	Blob type	Size	Lease st...	
□	speed_camera.json	1/16/2020, 2:40:39 PM	Hot (Infe...	Block blob	159.76 KiB	Available	•••

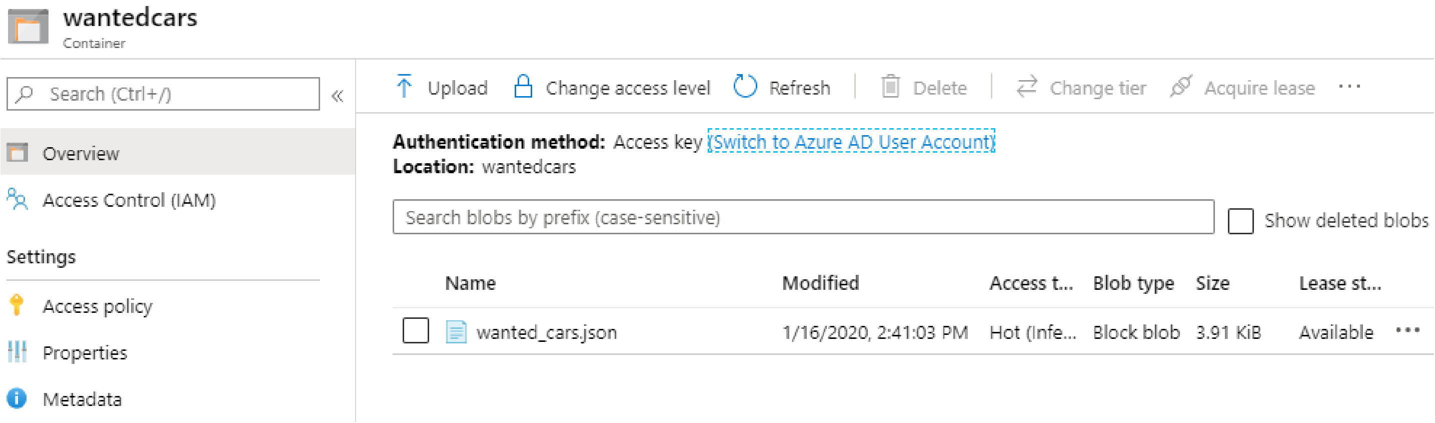
Speed camera container



Authentication method: Access key (Switch to Azure AD User Account)
Location: tollstations

	Name	Modified	Access t...	Blob type	Size	Lease st...	
□	toll_stations.json	1/16/2020, 2:40:49 PM	Hot (Infe...	Block blob	13.02 KiB	Available	•••

Toll stations container



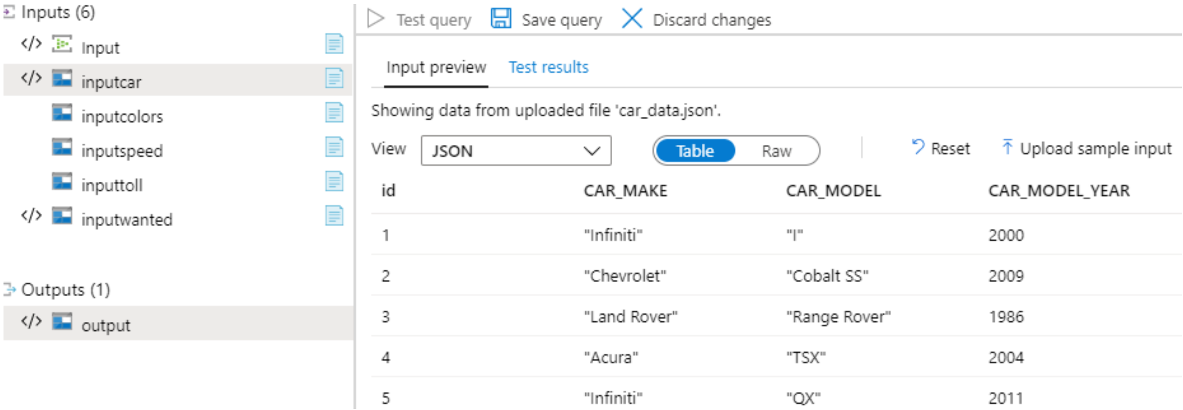
Authentication method: Access key
Location: wantedcars

Name	Modified	Access t...	Blob type	Size	Lease st...
wanted_cars.json	1/16/2020, 2:41:03 PM	Hot (Infe...	Block blob	3.91 KiB	Available

Wanted cars container

INPUTS

Next step was to create some extra inputs for each file included in the previously created containers. Specifically we created five more inputs (inputcar, inputcolors, inputspeed, inputtoll, inputwanted) as shown below.

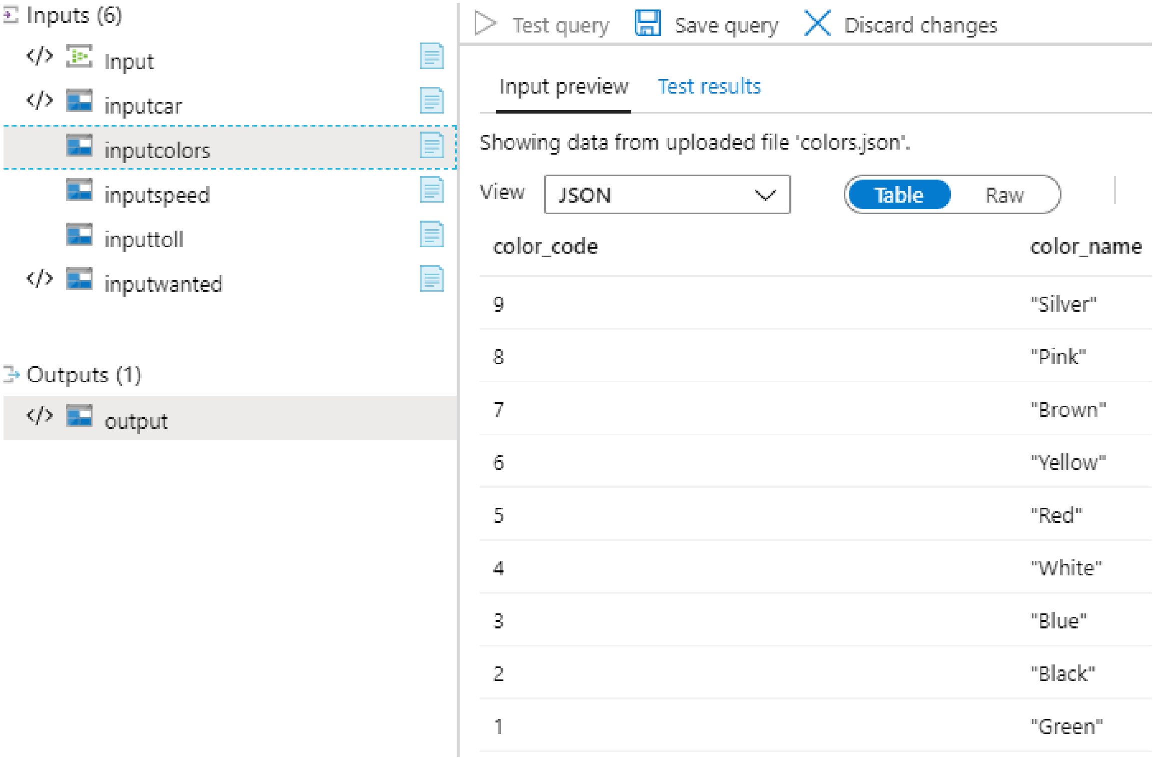


Showing data from uploaded file 'car_data.json'.

View JSON

id	CAR_MAKE	CAR_MODEL	CAR_MODEL_YEAR
1	"Infiniti"	"I"	2000
2	"Chevrolet"	"Cobalt SS"	2009
3	"Land Rover"	"Range Rover"	1986
4	"Acura"	"TSX"	2004
5	"Infiniti"	"QX"	2011

Inputcar (car data)

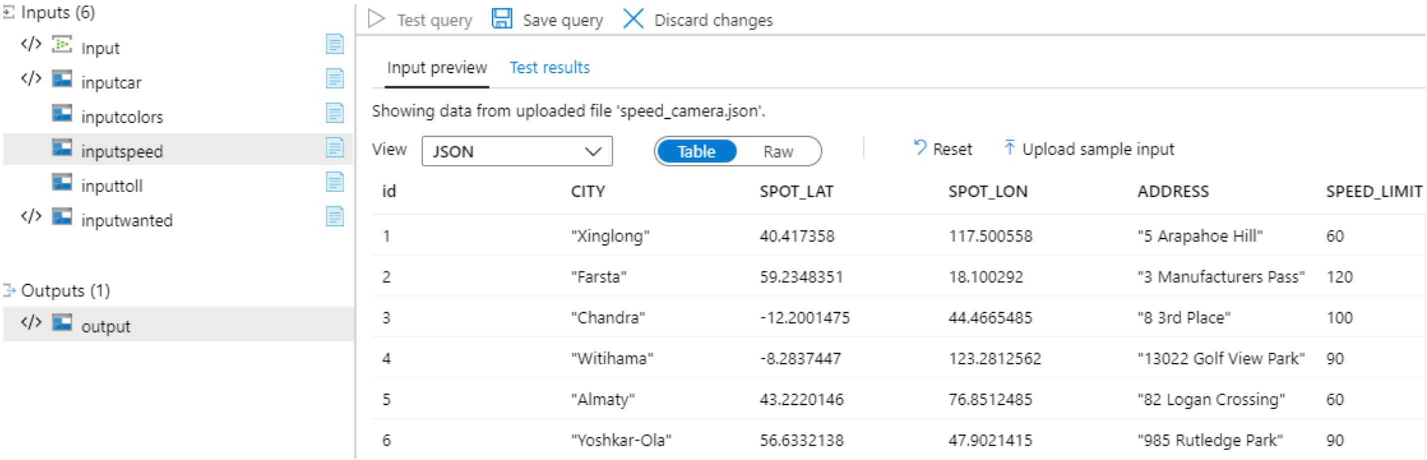


Showing data from uploaded file 'colors.json'.

View: JSON — Table | Raw

color_code	color_name
9	"Silver"
8	"Pink"
7	"Brown"
6	"Yellow"
5	"Red"
4	"White"
3	"Blue"
2	"Black"
1	"Green"

Inputcolors (colors)

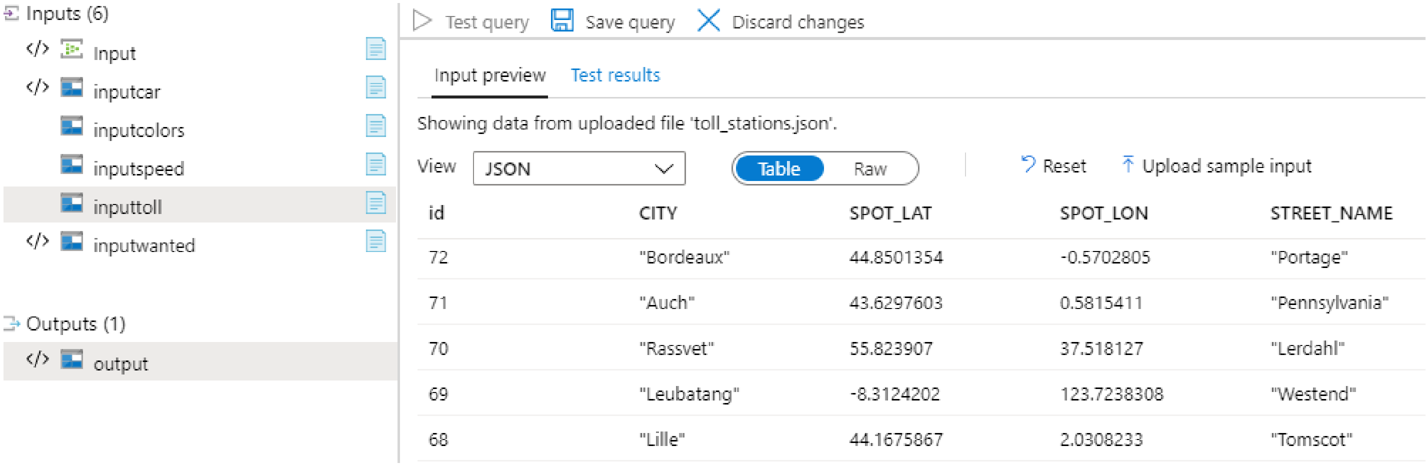


Showing data from uploaded file 'speed_camera.json'.

View: JSON — Table | Raw

id	CITY	SPOT_LAT	SPOT_LON	ADDRESS	SPEED_LIMIT
1	"Xinglong"	40.417358	117.500558	"5 Arapahoe Hill"	60
2	"Farsta"	59.2348351	18.100292	"3 Manufacturers Pass"	120
3	"Chandra"	-12.2001475	44.4665485	"8 3rd Place"	100
4	"Witihama"	-8.2837447	123.2812562	"13022 Golf View Park"	90
5	"Almaty"	43.2220146	76.8512485	"82 Logan Crossing"	60
6	"Yoshkar-Ola"	56.6332138	47.9021415	"985 Rutledge Park"	90

Inputspeed (speedcamera)

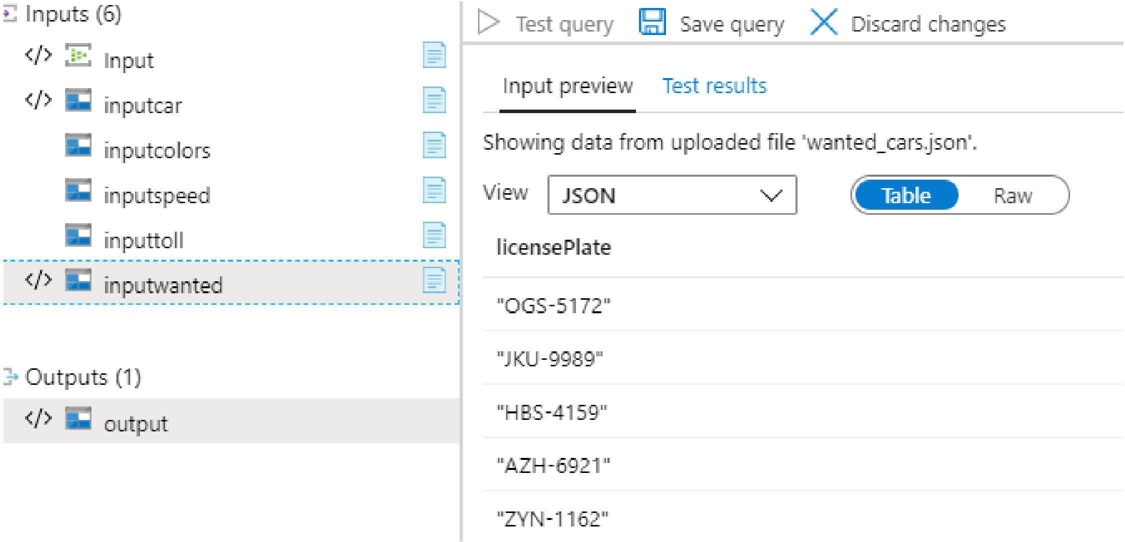


Showing data from uploaded file 'toll_stations.json'.

View: JSON — Table | Raw

id	CITY	SPOT_LAT	SPOT_LON	STREET_NAME
72	"Bordeaux"	44.8501354	-0.5702805	"Portage"
71	"Auch"	43.6297603	0.5815411	"Pennsylvania"
70	"Rassvet"	55.823907	37.518127	"Lerdahl"
69	"Leubatang"	-8.3124202	123.7238308	"Westend"
68	"Lille"	44.1675867	2.0308233	"Tomscot"

Inputtoll (tollstations)



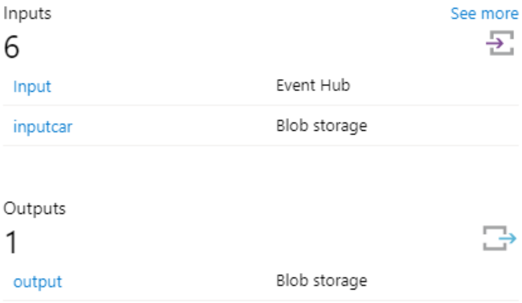
Showing data from uploaded file 'wanted_cars.json'.

View: JSON — Table | Raw

licensePlate
"OGS-5172"
"JKU-9989"
"HBS-4159"
"AZH-6921"
"ZYN-1162"

Inputwanted (wantedcars)

We end up having 6 inputs, 1 from the event hub and 5 from Blob storages, and 1 output.



Inputs

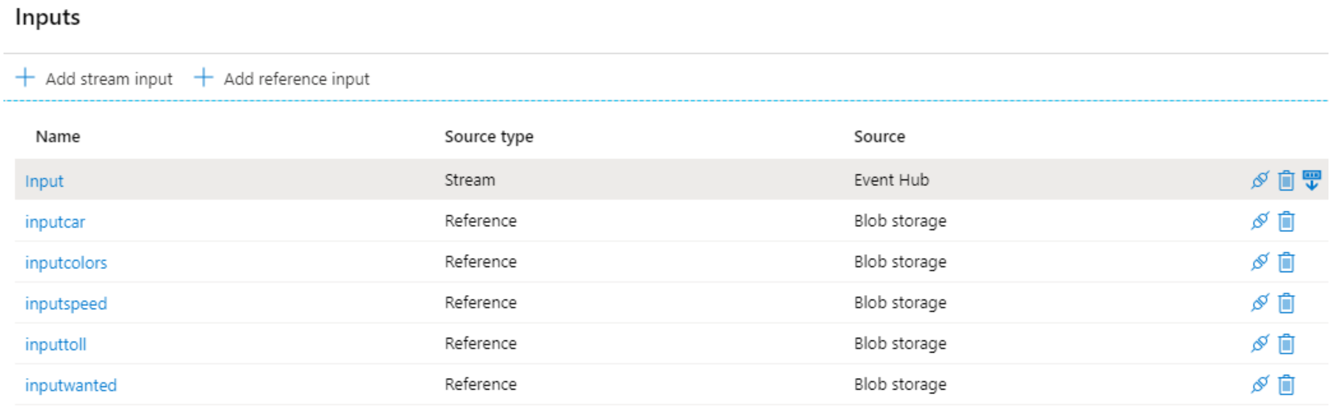
6

Input	Event Hub
inputcar	Blob storage

Outputs

1

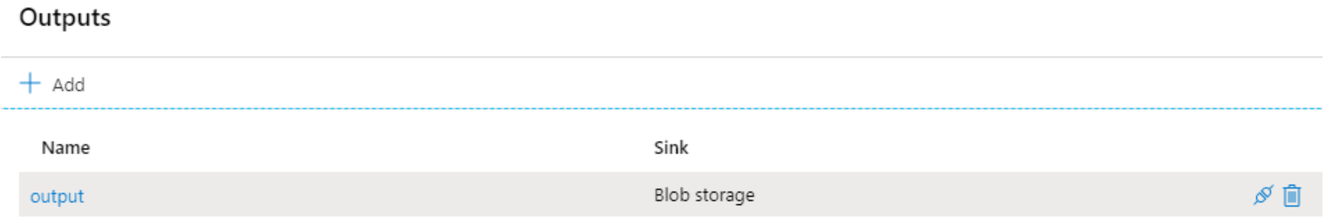
output	Blob storage



Inputs

Name	Source type	Source
Input	Stream	Event Hub
inputcar	Reference	Blob storage
inputcolors	Reference	Blob storage
inputspeed	Reference	Blob storage
inputtoll	Reference	Blob storage
inputwanted	Reference	Blob storage

Inputs summary



Outputs

Name	Sink
output	Blob storage

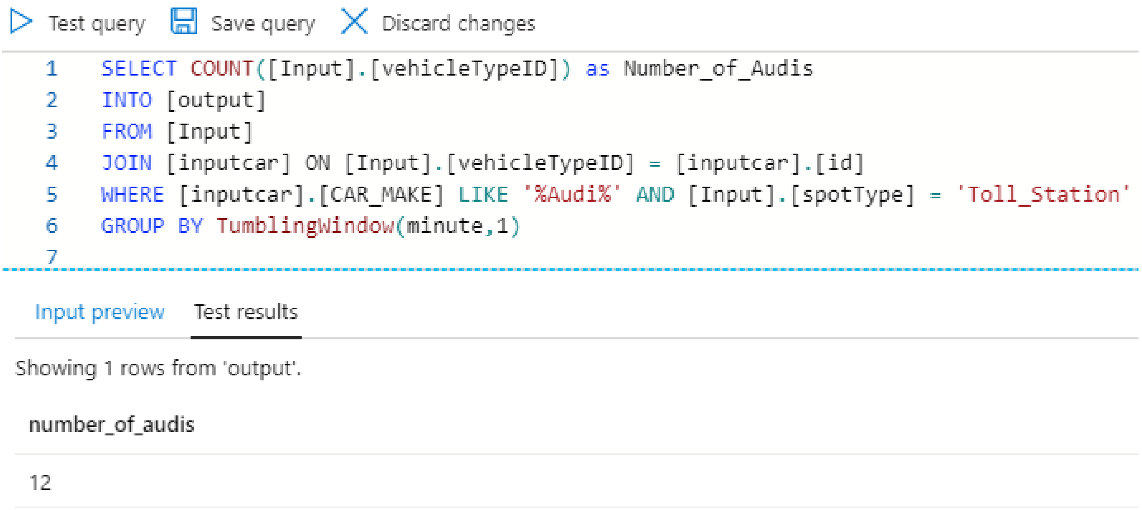
Output summary

Queries

This chapter presents the queries asked on the assignment, along with a screenshot with the results per query. Please note that we used the data generator html file, for creating data and sending them to the event hub input. Query testing, required to feed the event hub for more than 30 minutes in order to have enough data for our needs.

The modification made in data generator file was to use the signature key created for our event hub in the 'sas' variable. That was presented also in page 3.

Query 1: In a **tumbling window** of 1-minute count the number of Audis that passed through a toll station.



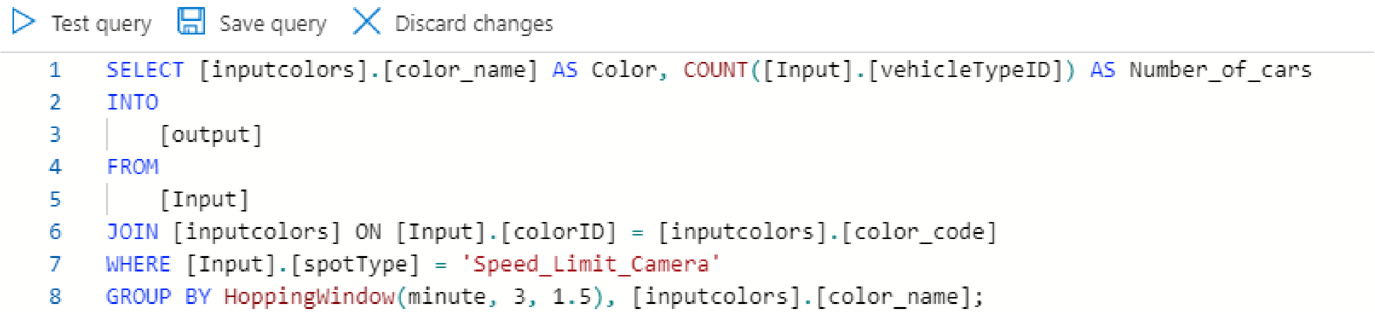
```sql
SELECT COUNT([Input].[vehicleTypeID]) as Number_of_Audis
INTO [output]
FROM [Input]
JOIN [inputcar] ON [Input].[vehicleTypeID] = [inputcar].[id]
WHERE [inputcar].[CAR_MAKE] LIKE '%Audi%' AND [Input].[spotType] = 'Toll_Station'
GROUP BY TumblingWindow(minute,1)

```

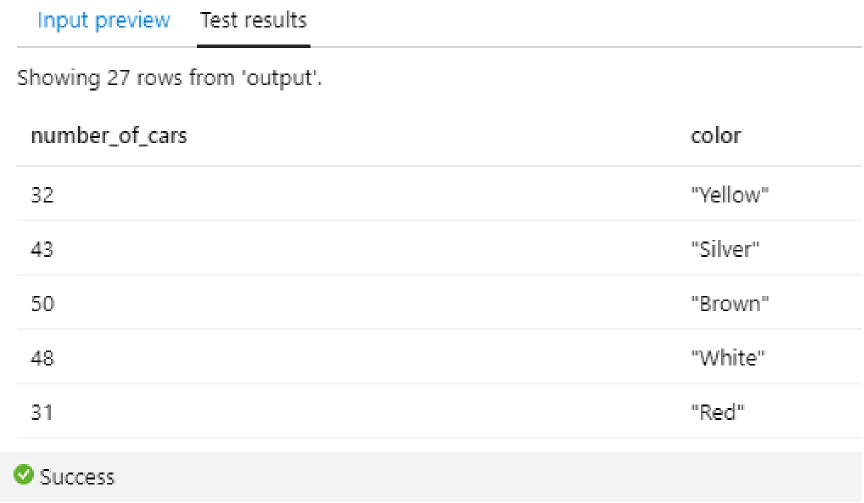
Showing 1 rows from 'output'.

number_of_audis
12

Query 2: In a **hopping window** of 3 minutes, for each color, calculate the total number of cars that passed through a police speed limit camera. Repeat every 90 seconds.



```sql
SELECT [inputcolors].[color_name] AS Color, COUNT([Input].[vehicleTypeID]) AS Number_of_cars
INTO
    [output]
FROM
    [Input]
JOIN [inputcolors] ON [Input].[colorID] = [inputcolors].[color_code]
WHERE [Input].[spotType] = 'Speed_Limit_Camera'
GROUP BY HoppingWindow(minute, 3, 1.5), [inputcolors].[color_name];
```

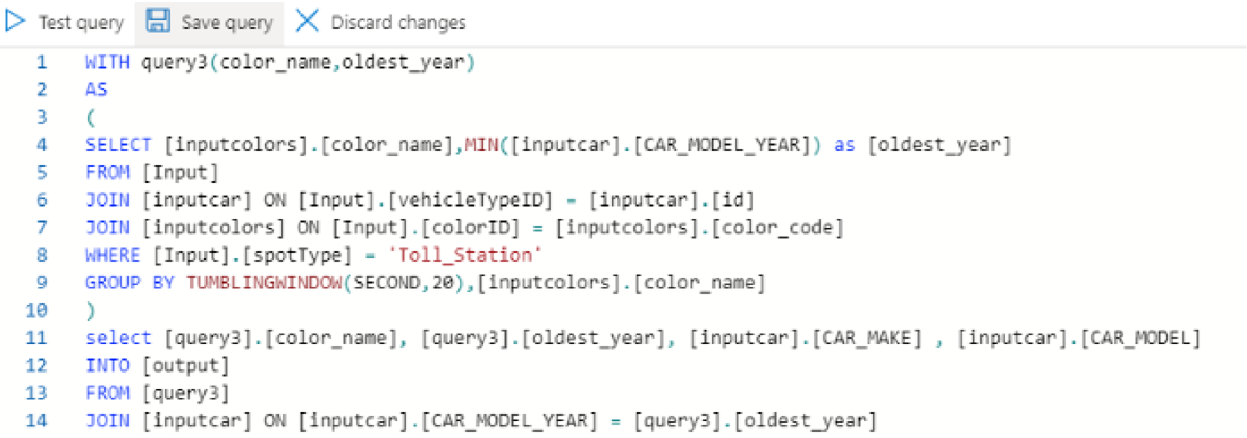


Showing 27 rows from 'output'.

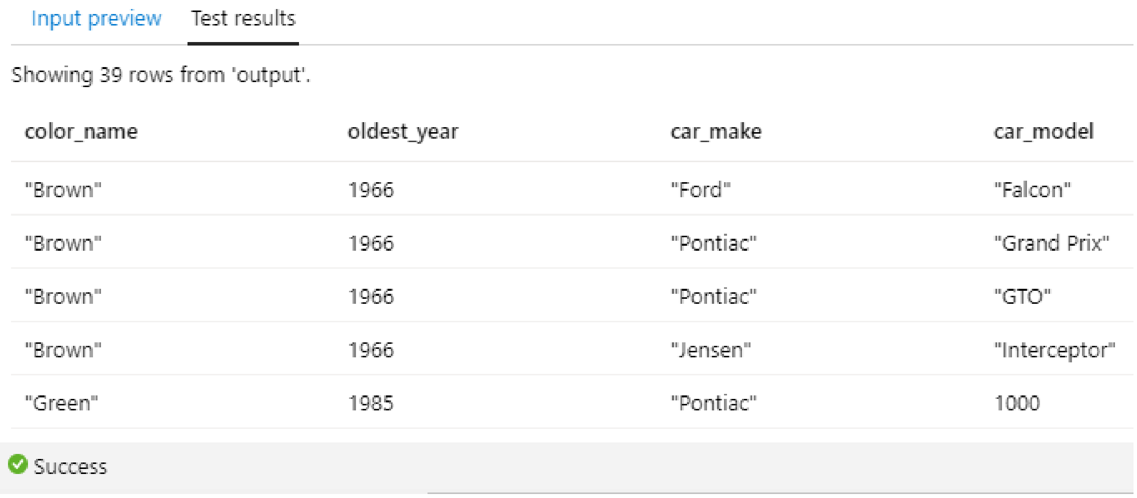
number_of_cars	color
32	"Yellow"
43	"Silver"
50	"Brown"
48	"White"
31	"Red"

Success

Query 3: In a **tumbling window** of 20 seconds, for each color, find the oldest car that passed through a toll station.



```
WITH query3(color_name,oldest_year)
AS
(
SELECT [inputcolors].[color_name],MIN([inputcar].[CAR_MODEL_YEAR]) as [oldest_year]
FROM [Input]
JOIN [inputcar] ON [Input].[vehicleTypeID] = [inputcar].[id]
JOIN [inputcolors] ON [Input].[colorID] = [inputcolors].[color_code]
WHERE [Input].[spotType] = 'Toll_Station'
GROUP BY TUMBLINGWINDOW(SECOND,20),[inputcolors].[color_name]
)
select [query3].[color_name], [query3].[oldest_year], [inputcar].[CAR_MAKE] , [inputcar].[CAR_MODEL]
INTO [output]
FROM [query3]
JOIN [inputcar] ON [inputcar].[CAR_MODEL_YEAR] = [query3].[oldest_year]
```

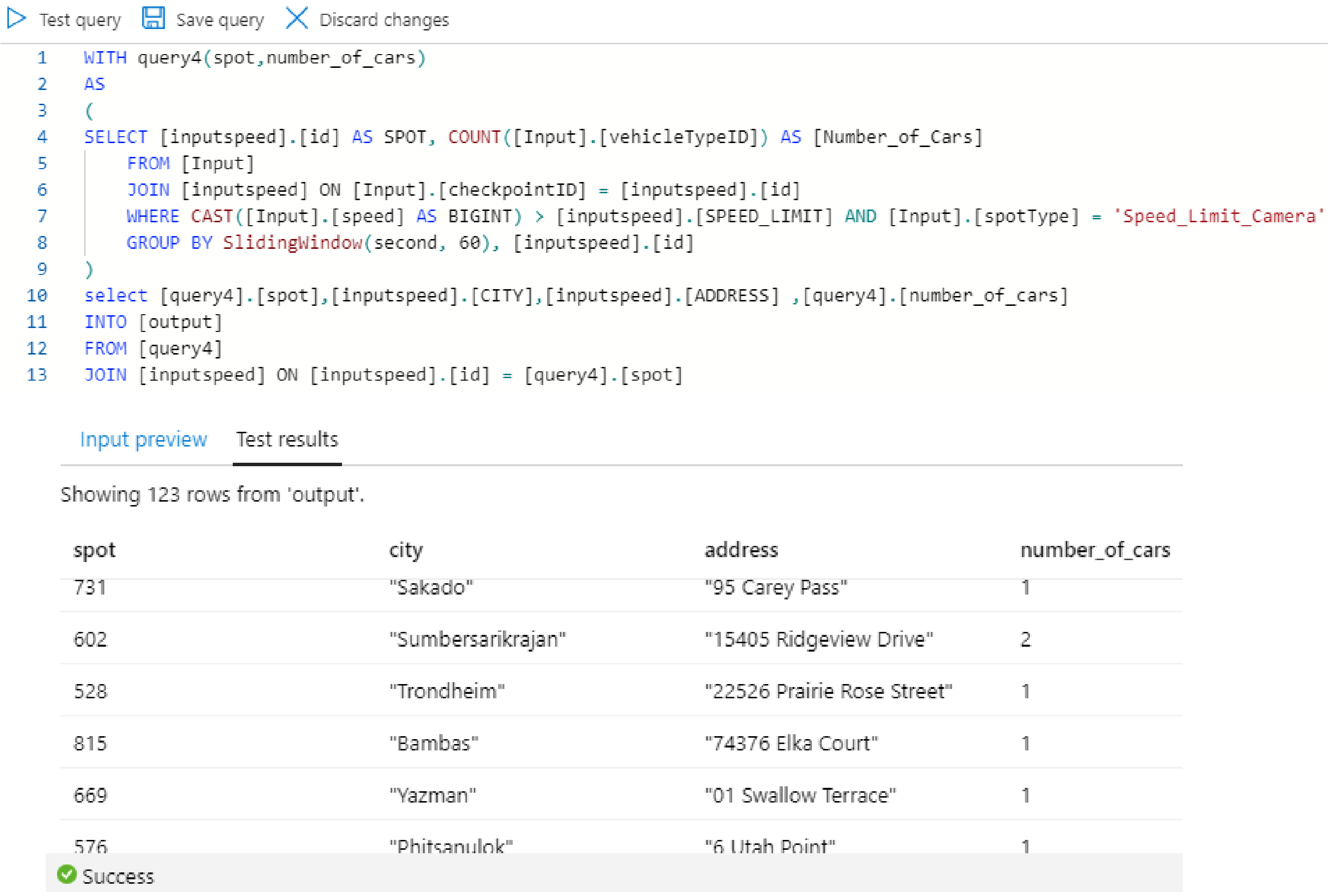


Showing 39 rows from 'output'.

color_name	oldest_year	car_make	car_model
"Brown"	1966	"Ford"	"Falcon"
"Brown"	1966	"Pontiac"	"Grand Prix"
"Brown"	1966	"Pontiac"	"GTO"
"Brown"	1966	"Jensen"	"Interceptor"
"Green"	1985	"Pontiac"	1000

Success

Query 4: In a **sliding window** of 60 seconds, calculate the speed limit camera spots where the most violations happened.

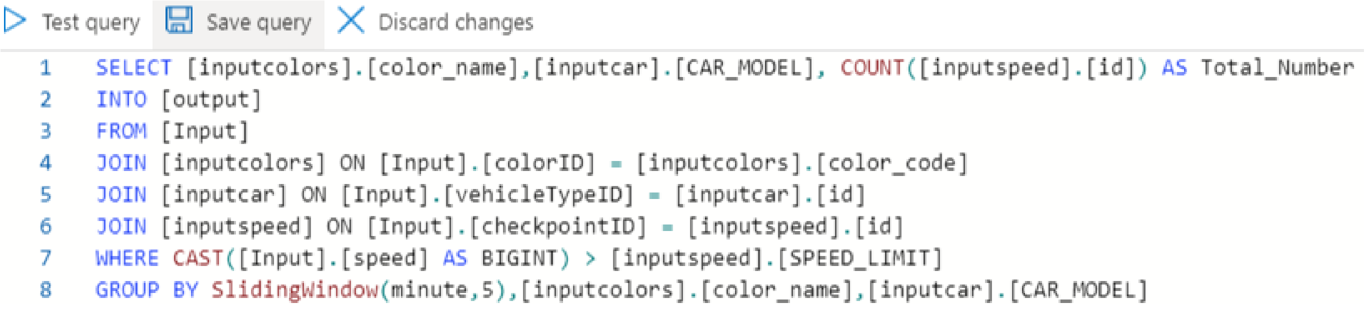


```
WITH query4(spot,number_of_cars)
AS
(
SELECT [inputspeed].[id] AS SPOT, COUNT([Input].[vehicleTypeID]) AS [Number_of_Cars]
    FROM [Input]
    JOIN [inputspeed] ON [Input].[checkpointID] = [inputspeed].[id]
    WHERE CAST([Input].[speed] AS BIGINT) > [inputspeed].[SPEED_LIMIT] AND [Input].[spotType] = 'Speed_Limit_Camera'
    GROUP BY SlidingWindow(second, 60), [inputspeed].[id]
)
select [query4].[spot],[inputspeed].[CITY],[inputspeed].[ADDRESS] ,[query4].[number_of_cars]
INTO [output]
FROM [query4]
JOIN [inputspeed] ON [inputspeed].[id] = [query4].[spot]
```

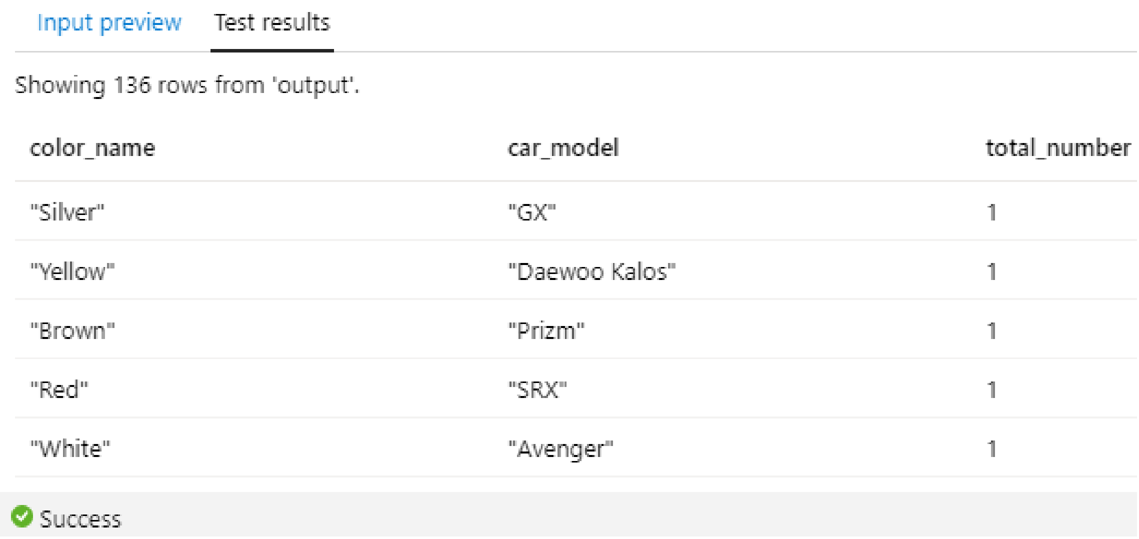
Showing 123 rows from 'output'.

spot	city	address	number_of_cars
731	"Sakado"	"95 Carey Pass"	1
602	"Sumbersarikrajan"	"15405 Ridgeview Drive"	2
528	"Trondheim"	"22526 Prairie Rose Street"	1
815	"Bambas"	"74376 Elka Court"	1
669	"Yazman"	"01 Swallow Terrace"	1
576	"Phitsanulok"	"6 Utah Point"	1

Query 5: In a **sliding window** of five minutes, for each color and car model, display the total number of cars that break the speed limit.



```
SELECT [inputcolors].[color_name],[inputcar].[CAR_MODEL], COUNT([inputspeed].[id]) AS Total_Number
INTO [output]
FROM [Input]
JOIN [inputcolors] ON [Input].[colorID] = [inputcolors].[color_code]
JOIN [inputcar] ON [Input].[vehicleTypeID] = [inputcar].[id]
JOIN [inputspeed] ON [Input].[checkpointID] = [inputspeed].[id]
WHERE CAST([Input].[speed] AS BIGINT) > [inputspeed].[SPEED_LIMIT]
GROUP BY SlidingWindow(minute,5),[inputcolors].[color_name],[inputcar].[CAR_MODEL]
```

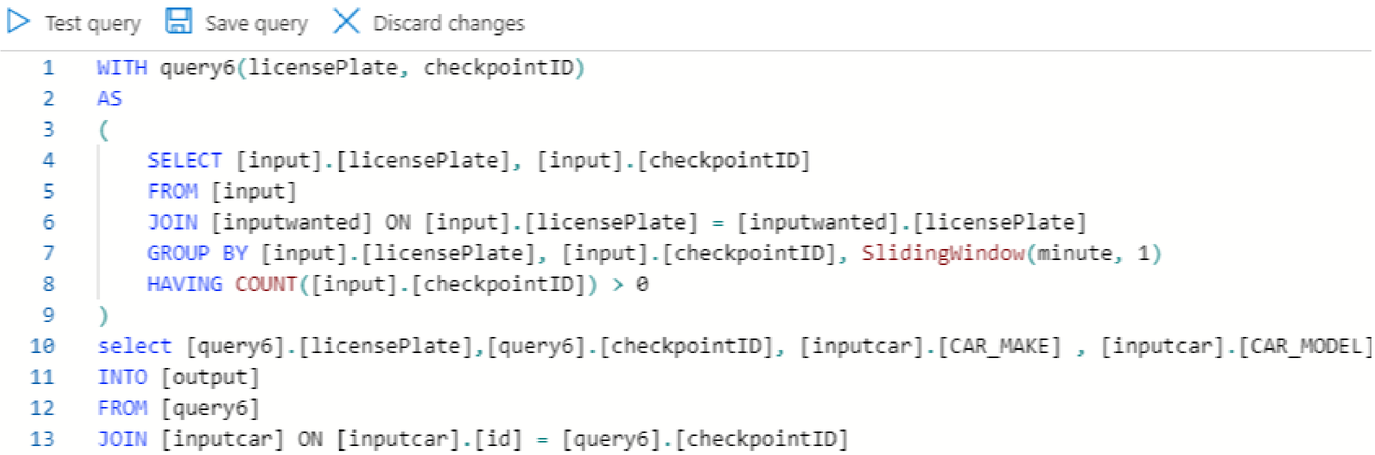


Showing 136 rows from 'output'.

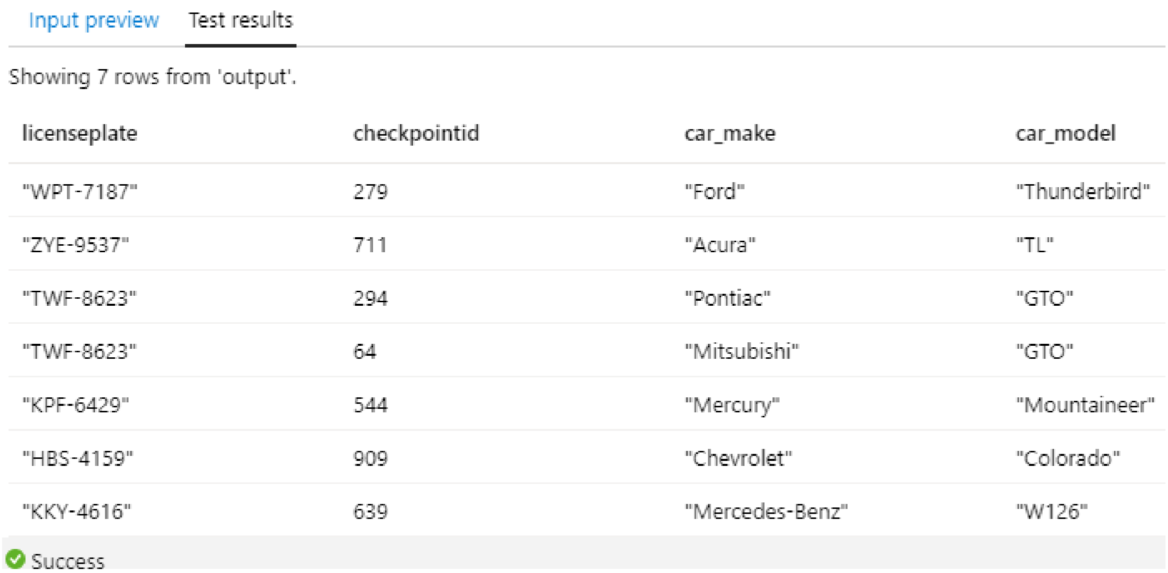
color_name	car_model	total_number
"Silver"	"GX"	1
"Yellow"	"Daewoo Kalos"	1
"Brown"	"Prizm"	1
"Red"	"SRX"	1
"White"	"Avenger"	1

Success

Query 6: You have been given a list of the license plates of police's most wanted criminals. In a **sliding window** of 1 minute, display a list of all the cars that you spotted at any checkpoint.



```
WITH query6(licensePlate, checkpointID)
AS
(
    SELECT [input].[licensePlate], [input].[checkpointID]
    FROM [input]
    JOIN [inputwanted] ON [input].[licensePlate] = [inputwanted].[licensePlate]
    GROUP BY [input].[licensePlate], [input].[checkpointID], SlidingWindow(minute, 1)
    HAVING COUNT([input].[checkpointID]) > 0
)
select [query6].[licensePlate],[query6].[checkpointID], [inputcar].[CAR_MAKE] , [inputcar].[CAR_MODEL]
INTO [output]
FROM [query6]
JOIN [inputcar] ON [inputcar].[id] = [query6].[checkpointID]
```

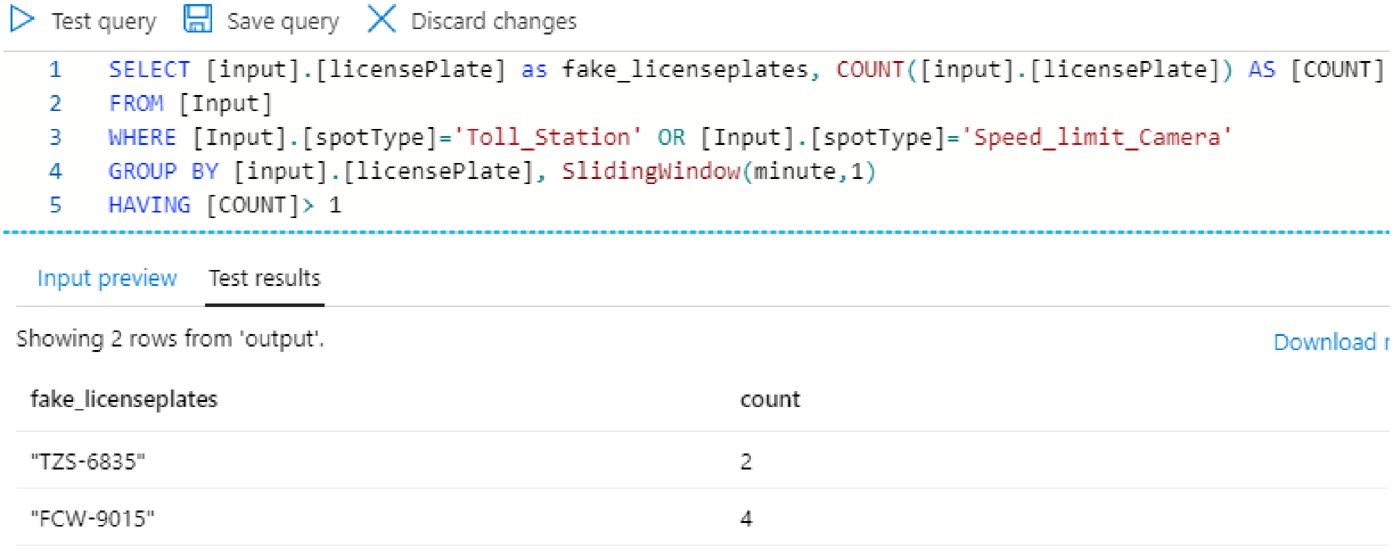


Showing 7 rows from 'output'.

licenseplate	checkpointid	car_make	car_model
"WPT-7187"	279	"Ford"	"Thunderbird"
"ZYE-9537"	711	"Acura"	"TL"
"TWF-8623"	294	"Pontiac"	"GTO"
"TWF-8623"	64	"Mitsubishi"	"GTO"
"KPF-6429"	544	"Mercury"	"Mountaineer"
"HBS-4159"	909	"Chevrolet"	"Colorado"
"KKY-4616"	639	"Mercedes-Benz"	"W126"

✓ Success

Query 7: In a **sliding window** of 1 minute, display a list of fake license plates. Check if the same license plate has passed through any type of checkpoint twice in the same time window.

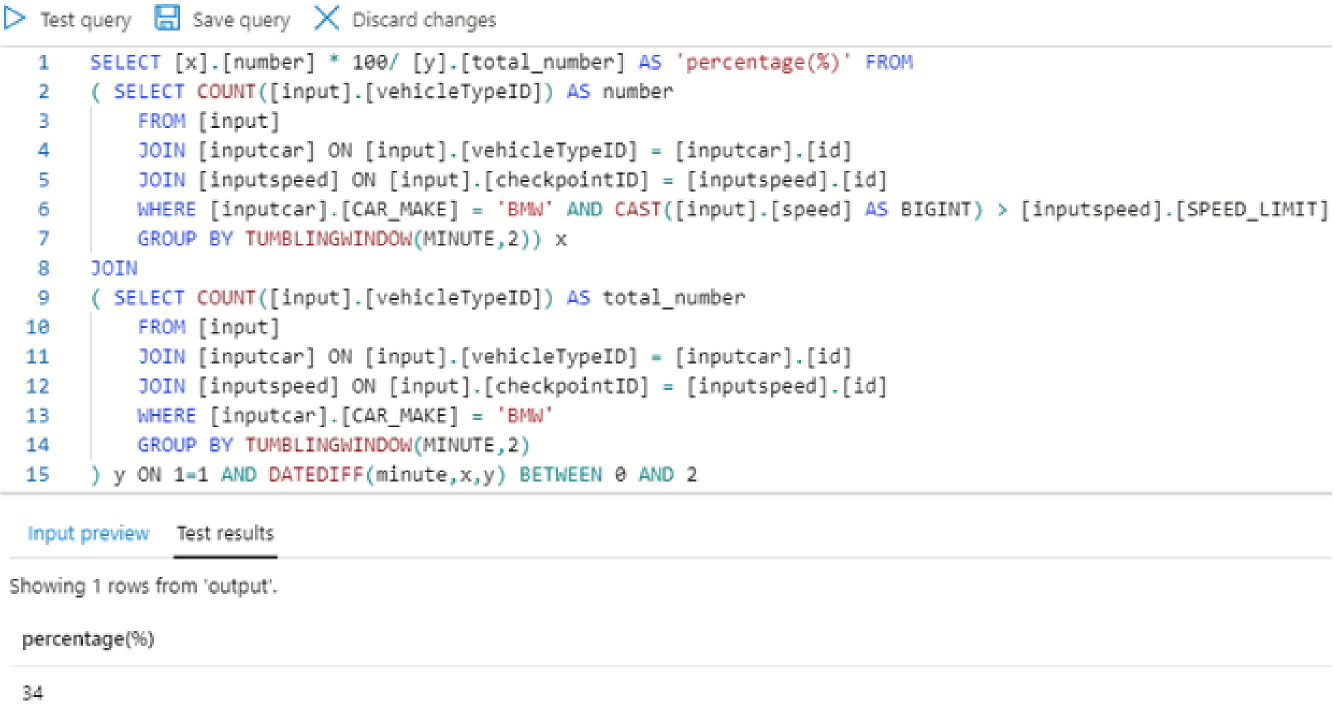


```
SELECT [input].[licensePlate] as fake_licenseplates, COUNT([input].[licensePlate]) AS [COUNT]
FROM [Input]
WHERE [Input].[spotType]='Toll_Station' OR [Input].[spotType]='Speed_limit_Camera'
GROUP BY [input].[licensePlate], SlidingWindow(minute,1)
HAVING [COUNT]> 1
```

Showing 2 rows from 'output'.

fake_licenseplates	count
"TZS-6835"	2
"FCW-9015"	4

Query 8: In a **tumbling window** of 2 minutes, calculate the percentage of BMW drivers that break the speed limit. (e.g. Out of all the BMW drivers that were identified in the last 2 minutes, 80% broke the speed limit).



```
SELECT [x].[number] * 100/ [y].[total_number] AS 'percentage(%)' FROM
( SELECT COUNT([input].[vehicleTypeID]) AS number
    FROM [input]
    JOIN [inputcar] ON [input].[vehicleTypeID] = [inputcar].[id]
    JOIN [inputspeed] ON [input].[checkpointID] = [inputspeed].[id]
    WHERE [inputcar].[CAR_MAKE] = 'BMW' AND CAST([input].[speed] AS BIGINT) > [inputspeed].[SPEED_LIMIT]
    GROUP BY TUMBLINGWINDOW(MINUTE,2)) x
JOIN
( SELECT COUNT([input].[vehicleTypeID]) AS total_number
    FROM [input]
    JOIN [inputcar] ON [input].[vehicleTypeID] = [inputcar].[id]
    JOIN [inputspeed] ON [input].[checkpointID] = [inputspeed].[id]
    WHERE [inputcar].[CAR_MAKE] = 'BMW'
    GROUP BY TUMBLINGWINDOW(MINUTE,2)
) y ON 1=1 AND DATEDIFF(minute,x,y) BETWEEN 0 AND 2
```

Showing 1 rows from 'output'.

percentage(%)
34