|  |  |
| --- | --- |
|  |  |
|  | SQL Stretch Demo Installation and Demo Script |

Contents

[Introduction 3](#_Toc453255775)

[Installation Script 4](#_Toc453255776)

[Demo Script 9](#_Toc453255777)

[Terms of Use 19](#_Toc453255778)

|  |
| --- |
| Introduction |

This demo is to showcase three new features of SQL Server 2016. These are Temporal, Stretch and Always Encrypted.

The demo is to be set up using installation scripts and some manual process. It is up to the presenter to remove any of the resources created in Azure so that they do not incur costs while not in use. It is the presenter’s responsibility to start/stop stretching the database if required. Note that this will still incur charges, you need to delete the remote DB in Azure.  
  
It is important to point out that the demo script follows a strict execution path as the frontend site is only partially functional to cater for the story flow. Some of the elements are static and purely cosmetic, any interaction with these will most likely cause undesired outcomes.

To help with this, each step in the demo script section has been clearly labeled with the elements required to tell the story.

It is assumed that the presenter has some prior knowledge with SQL Server 2016 and will be comfortable navigating around SSMS. Some basic PowerShell knowledge may be required.

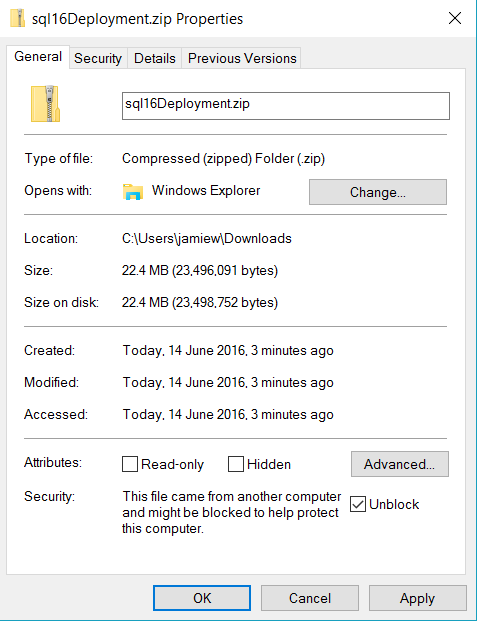
|  |
| --- |
| Installation Script |

**IMPORTANT**

This deployment script is built on Azure PowerShell 1.4.0. If you are experiencing any issues, please install this version onto your machine.

You need to run this on Windows 10 with Edge installed. The web front-end is ran locally on your machine from a lightweight portable webserver.

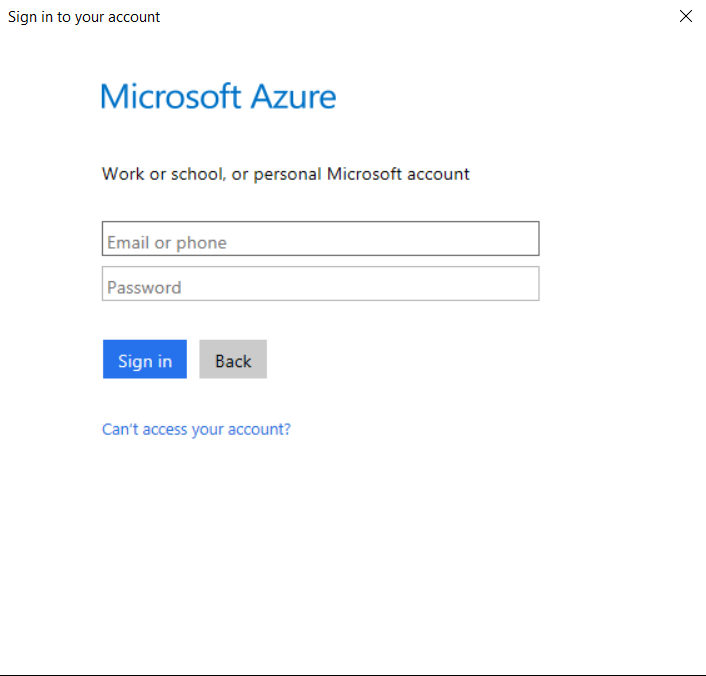
1. Locate sql16Deployment.zip
2. You first need to make sure the ZIP is unblocked. Right click properties then select unblock



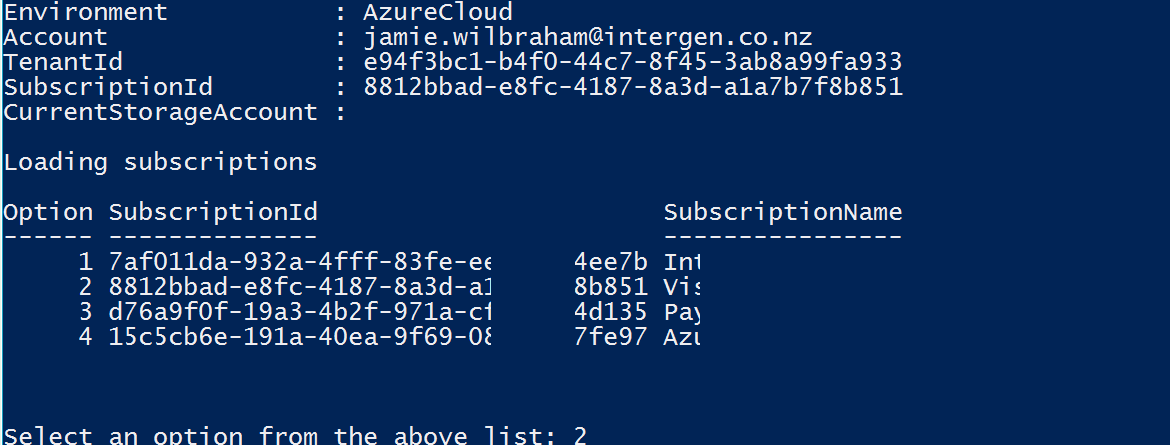
1. Open an **ADMIN** PowerShell and navigate to the Scripts folder in your sql16Deployment. Run the following command .\deploy.ps1



1. You will be prompted to sign into your Azure account.



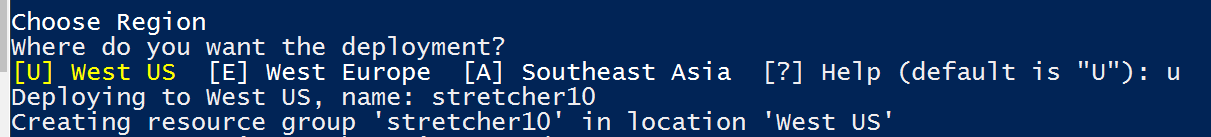
1. Select the subscription you would like to deply to.



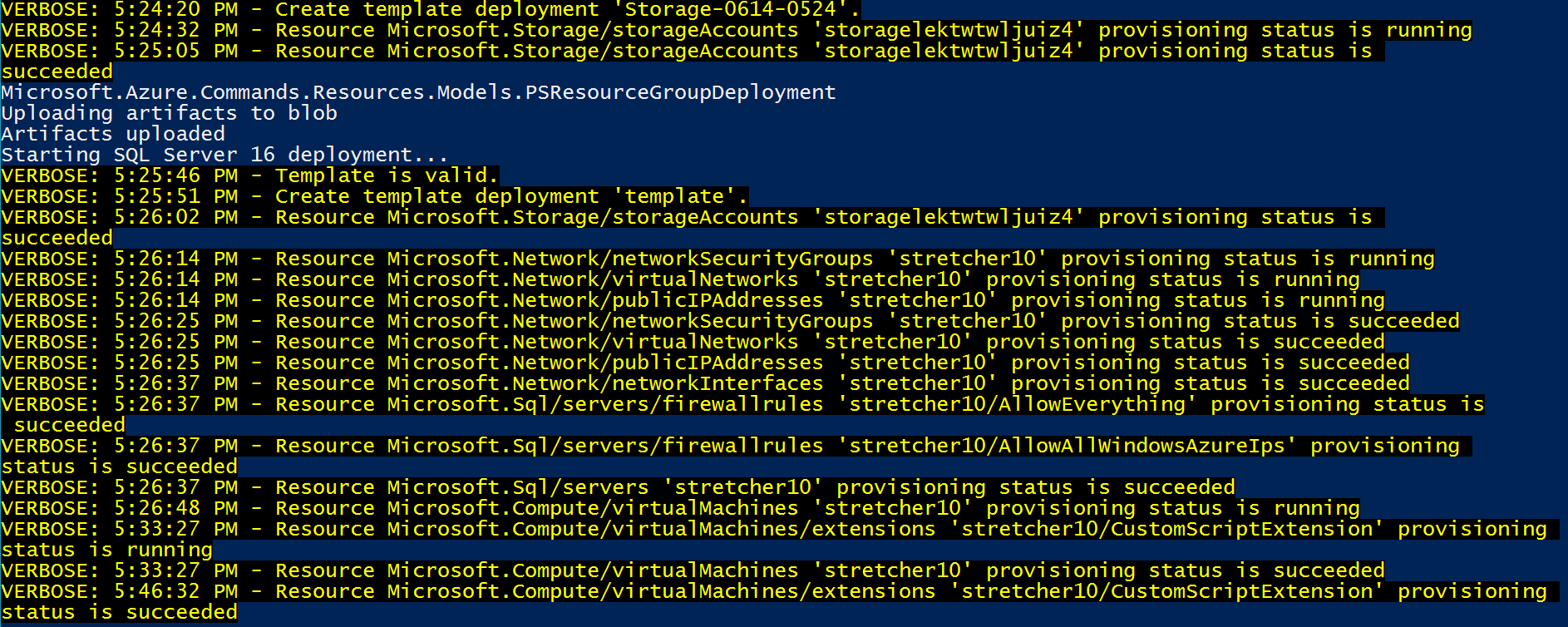
1. Give your deployment name a unique name that has to be 14 characters or less. This needs to be globally unique. Its good to put a random number at the end of the name to ensure it is unique.



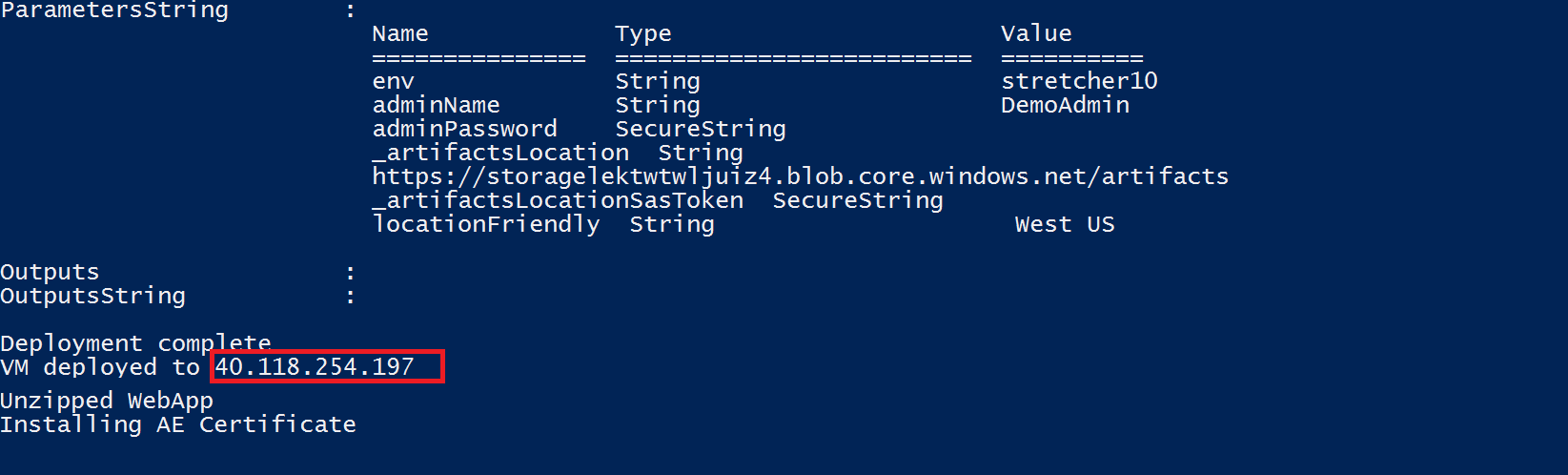
1. Choose your deployment location and the deployment will start.

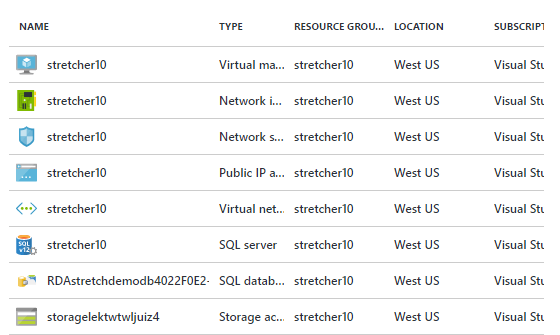


1. This process can take anywhere from 20-40 minutes to complete depending on your location. The process out the installation can be tracked from the output. An example is below.



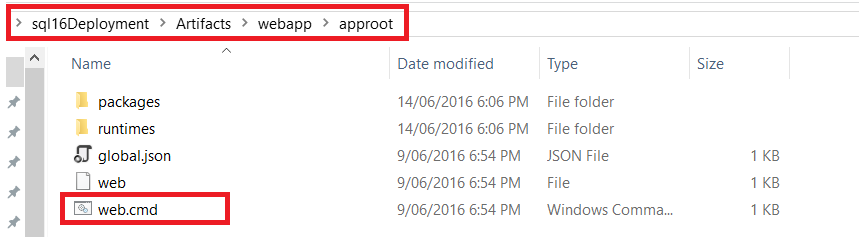
1. Once completed you will see an output of the deployment information. Take note of the Virtual Machine IP address if you want to RDP into it. It is important to note that a private cert is installed on your machine to ensure AE functionality will work.



1. You can go to Azure to confirm the components have been installed correctly. The VM database will start stretching straight away.  
     
   
2. You will notice a cmd window will pop up on completion starting a webserver. You need to browse to <http://localhost:5000> to get to the login page of your application.



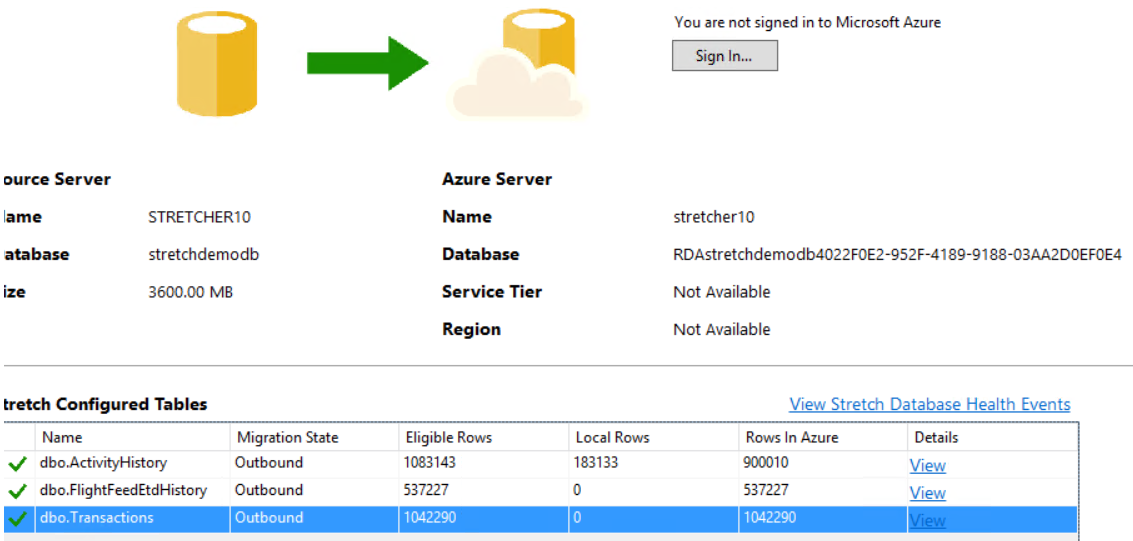
1. To run this manually in the future, run web.cmd from the approot (see below)



1. This will need to be running to connect to your frontend site. Always browse to localhost:5000 from your local machine and not the VM.
2. To Remote Desktop into the solution enter the following information:  
     
   **Computer** : *The IP address obtained in step 9*  
   **User**: DemoAdmin  
   **Password**: Pass@word2!

From here you will be able to access SSMS and view the database.

In the object explorer,   
*right-click stretchdemodb > tasks > stretch > monitor*  
to ensure the tables are being stretched. Note that this may take a while to complete.

It should look similar to below.  
  


**Installation Complete**

|  |
| --- |
| **Demo Script** |

**Scenario**

The presenter is a sales agent for Wingtips Travel Co, a travel company that specializes in providing a concierge service to their clients.   
  
In this demo, the presenter will simulate receiving a travel request from a fictional client named Jamie Campbell, for the sake of the demo it is an email request. Jamie needs to book a one-way flight from Arizona to Quebec and this flight requires a connection in Chicago.

Due to a range of factors in Chicago that can cause delays, the sales agent needs to find a suitable flight that is well priced and reliable. The sales agent (presenter) will showcase some of the new features in SQL Server 2016 that will ultimately guide them into booking a flight based on existing data.

**1.** **Login**

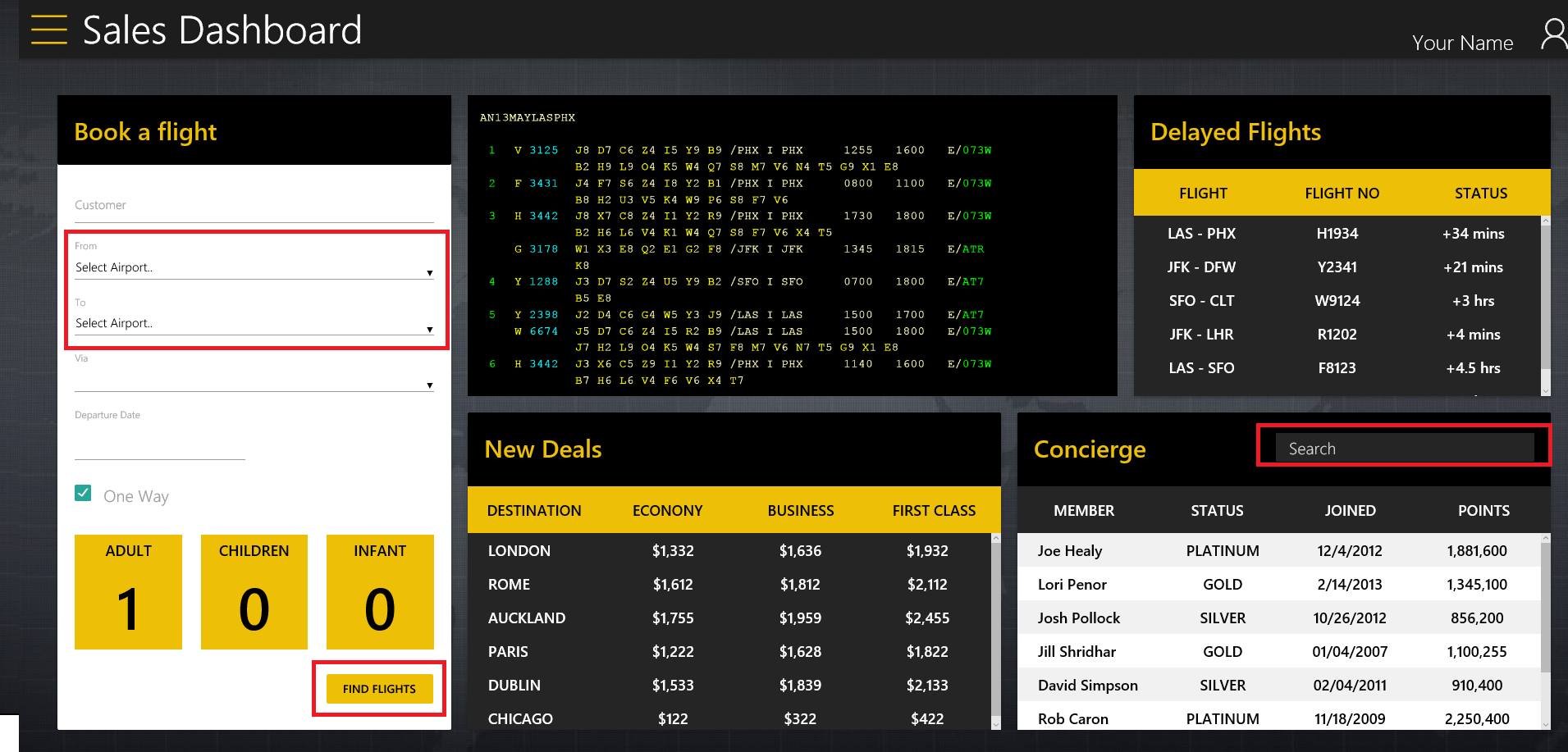
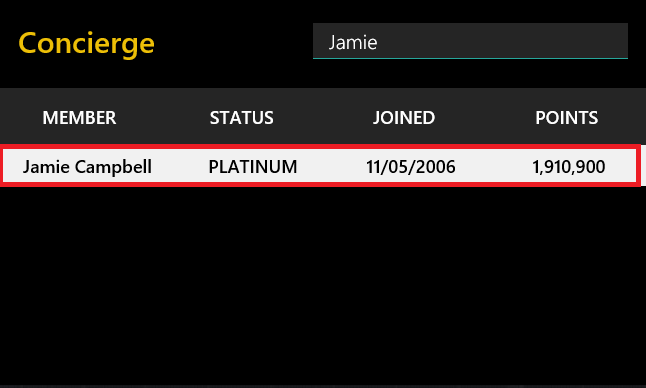
After you have arrived at your homepage in the MS Edge browser (see installation steps), you will be presented at the following login screen. The password is prepopulated, however, you have the option to enter a username. This will flow through the app, if its left blank, it will default to “Sales Agent”.



Select the login button to continue to the Sales Dashboard.

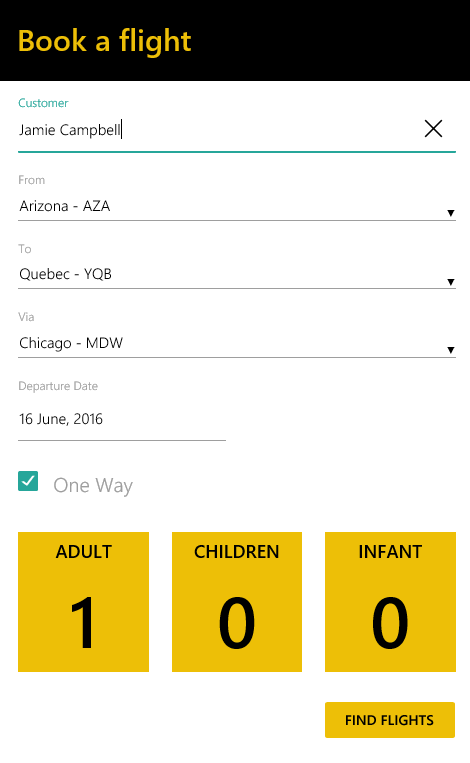
2. **Sales Dashboard**

The sales dashboard’s primary purpose is to aid the story. It doesn’t really show any features of SQL Server 2016 and should the time spent here should be brief. The presenter will be inputting some values from the clients request so that they can assess potential flights. The image below shows the elements you will need to interact with.

  
  
First, start typing the name “Jamie” in the concierge search and the list will filter with Jamie Campbell displaying. Select this user.  
  


You will now need to enter additional information into the “Book a flight” section. You will notice from the last step “Jamie Campbell” has already been prepopulated in the customer input element. Enter the following information to replicate the image below.   
  
**FROM** – Arizona - AZA  
**TO** – Quebec – YQB  
**VIA** – *this will be populated with Chicago – MDW after completing the “To” input. It’s the only option indicating the only flight is through Chicago.***DEPARTURE DATE** – Set a date in the future relative to when the demo is presented**ONE WAY** – Leave selected

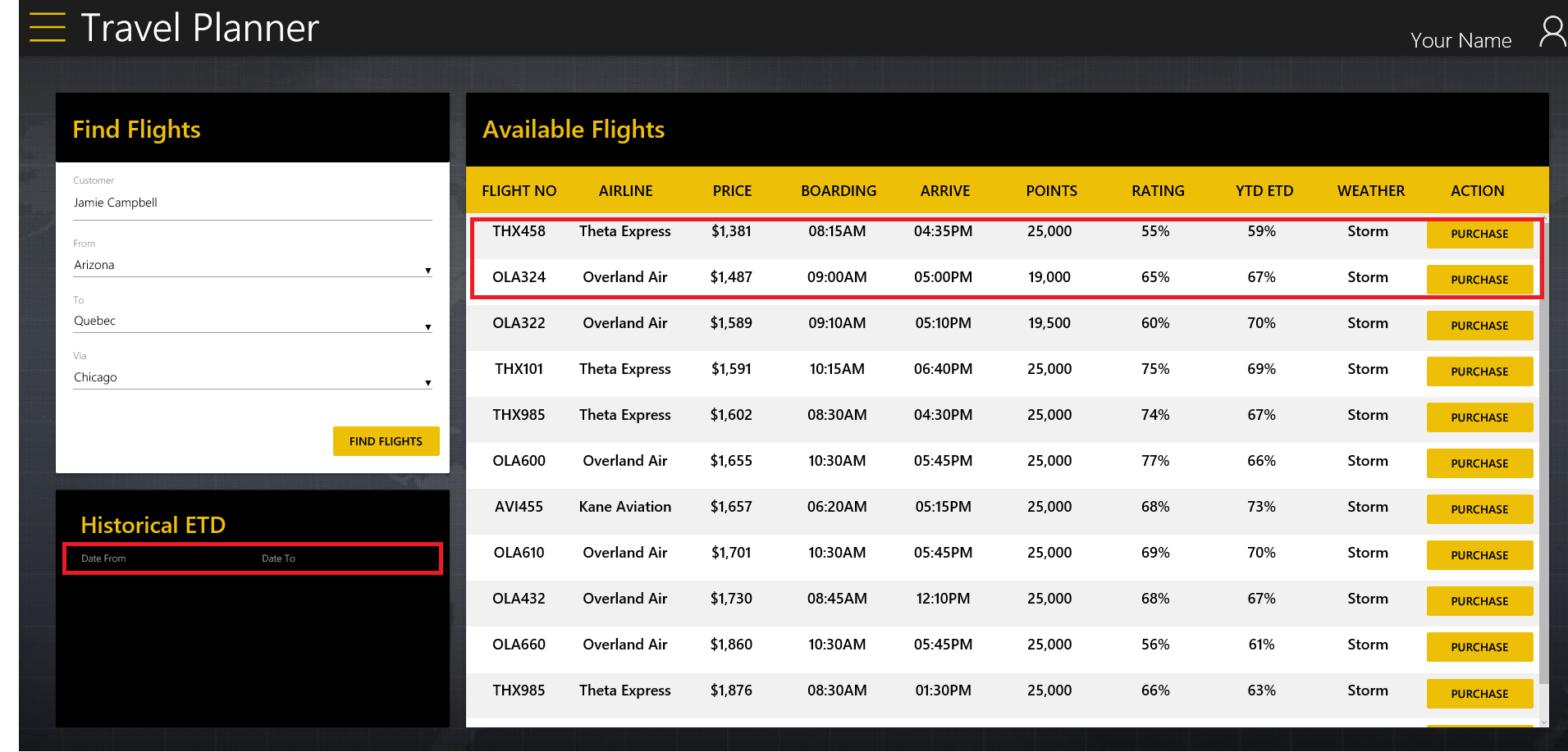
Your input should look like the following:

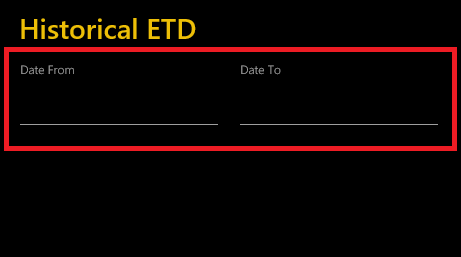


Select “Find Flights” to continue to the travel planner.

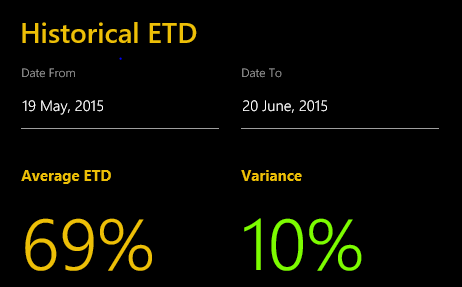
**3.Travel Planner**

The part of the demo highlights 2 features of SQL Server 2016, temporal and also stretching the history of the system-versioned temporal table. Below indicates the page elements that you will need to interact with in this stage of the demo. It will also require switching to your database in SSMS to highlight how the components are functioning in the DB.

  
  
  
The two top flights returned are the ones of interest. The YTD ETD (Year-To-Date Estimated-Time (of)- Departure) and PRICE are the columns of interest. We can see that flight THX458 have a cheaper price, but a lower chance of leaving on time.   
  
Select flight THX458 and the line will be highlighted grey.  
  
With the power of temporal you are able to query the history of data as it was at a previous time. The values here are produced client-side and not actually hooked up to the DB as it is too complex to know when the presenter will be carrying out the demo. Hypothetically, these values would be read from the dbo.FlightEtdFeed in the DB.  
  
Choose From/To dates below ensuring they are in the past (over a year ago). The dates are irrelevant and the flight selected indicates the value.



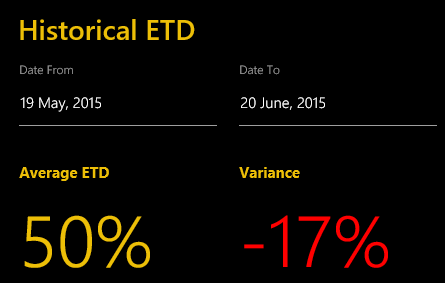
You will notice the result always yields a positive variance against the current YTD of 59%. You can repeat with different dates to get the point across. It will never generate a negative variance for the sake of the demo.



So we now know the flights history is usually shows a higher ETD currently shown, which is good information for the sales agent to help assist their decision

We are still interested in flight “OLA324” as an option, so we repeat the process. Leave the dates the same for now.  
  
Select flight OLA324

You will immediately notice the variance drops, indicating “OLA324” flights ETD history has not been as reliable as it is currently. You can repeat this with dates to express the point.



It now makes sense to book flight “THX458” as it is cheaper and is has had a relatively promising history compared to “OLA324”.

Before purchasing the flight, you need to switch to the DB and run some queries.

Explain you are going to run a query to demonstrate what the value of flight THX458 was at a given time. For the sake of the demo, we can look at what it was on the 9th of June 2016 (it will be guaranteed to be different).  
  
Run the below query on the DB

USE stretchdemodb

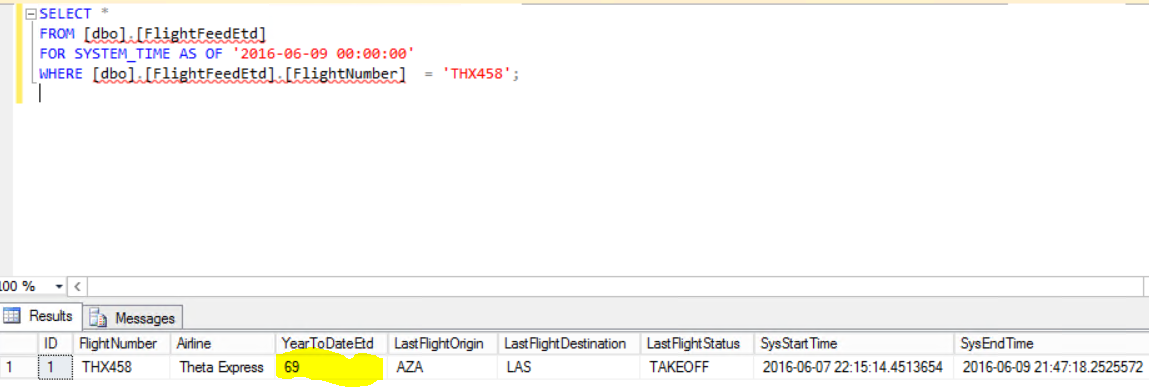
SELECT \*

FROM [dbo].[FlightFeedEtd]

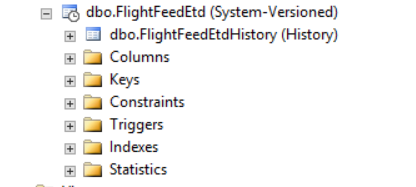
FOR SYSTEM\_TIME AS OF '2016-06-09 00:00:00'

WHERE [dbo].[FlightFeedEtd].[FlightNumber] = 'THX458';

You will notice the YTD ETD is higher that is what is being shown in the frontend which abides to the logic of the story.

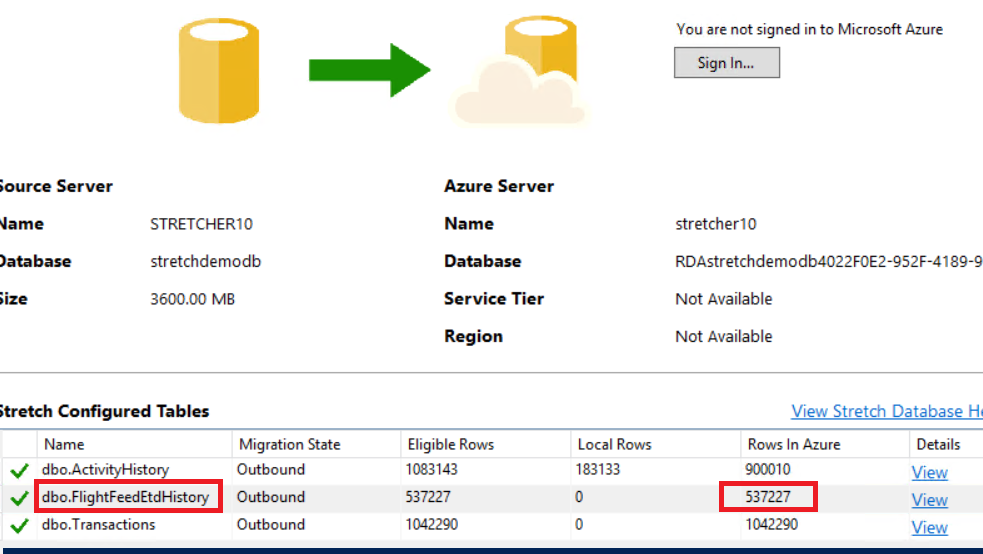


Expand the dbo.FlightFeedEtd table to show there is a history table. Explain that this is being stretched full to Azure



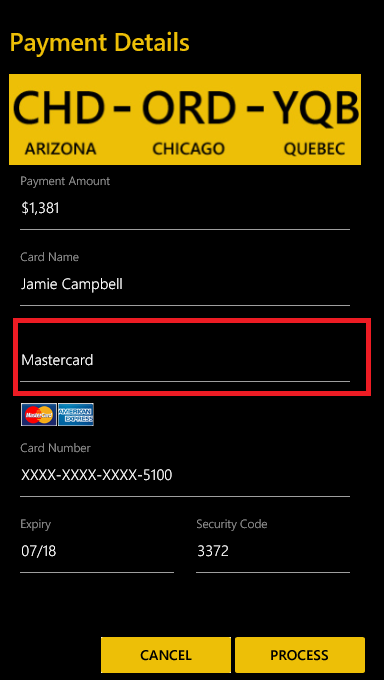
Check the monitor page to show the values in the history are being fully stretched in Azure. Point out the transaction count – you will need to bring this up after the transaction has been completed later in the demo.

In the object explorer,   
*right-click stretchdemodb > tasks > stretch > monitor*

****

Switch back to the front end website

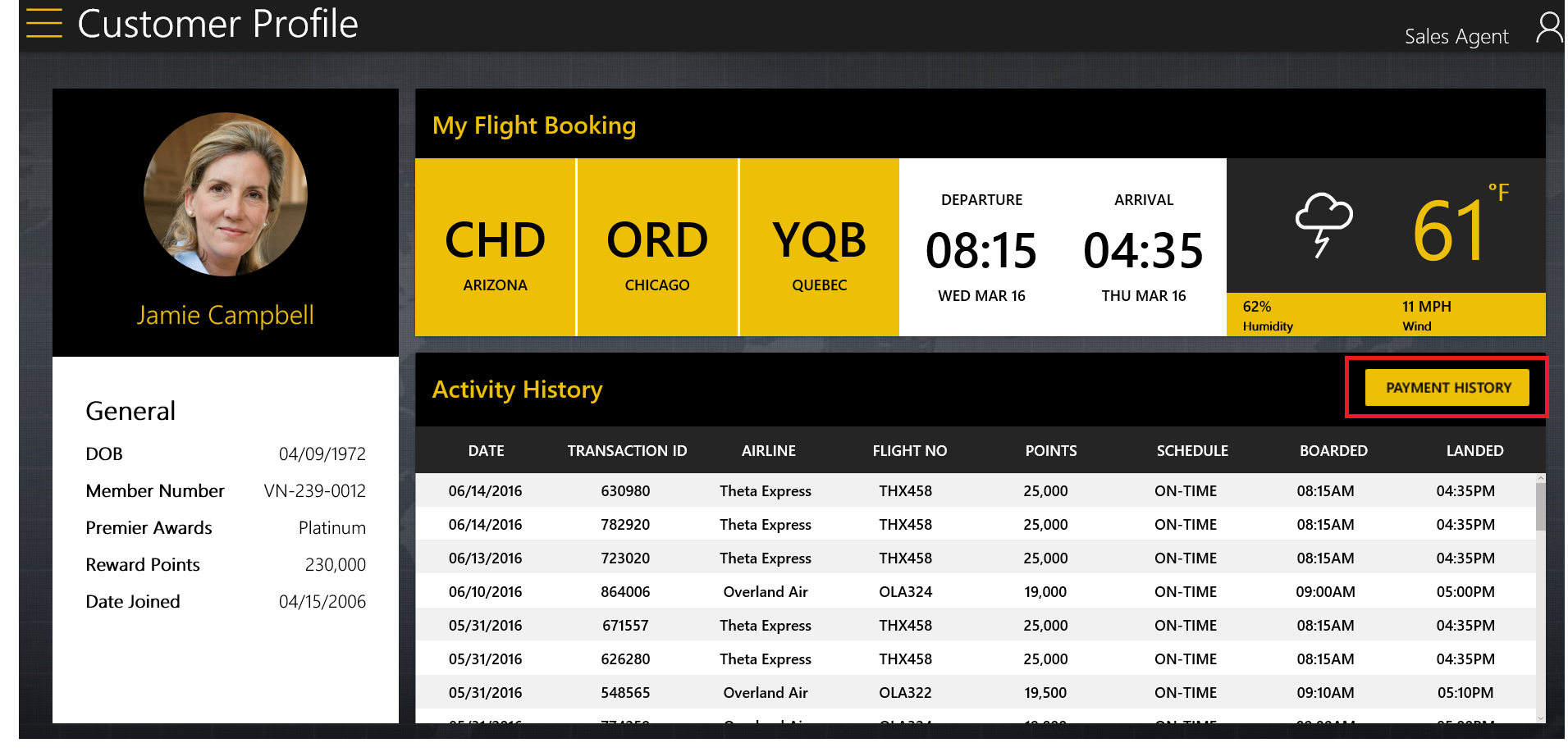
After deciding on the flight in the previous steps, click on the yellow PURCHASE button to proceed to the payment details modal.   
  
You only need to select the card from the dropdown, the rest of the information will be populated. See image below for expected input.



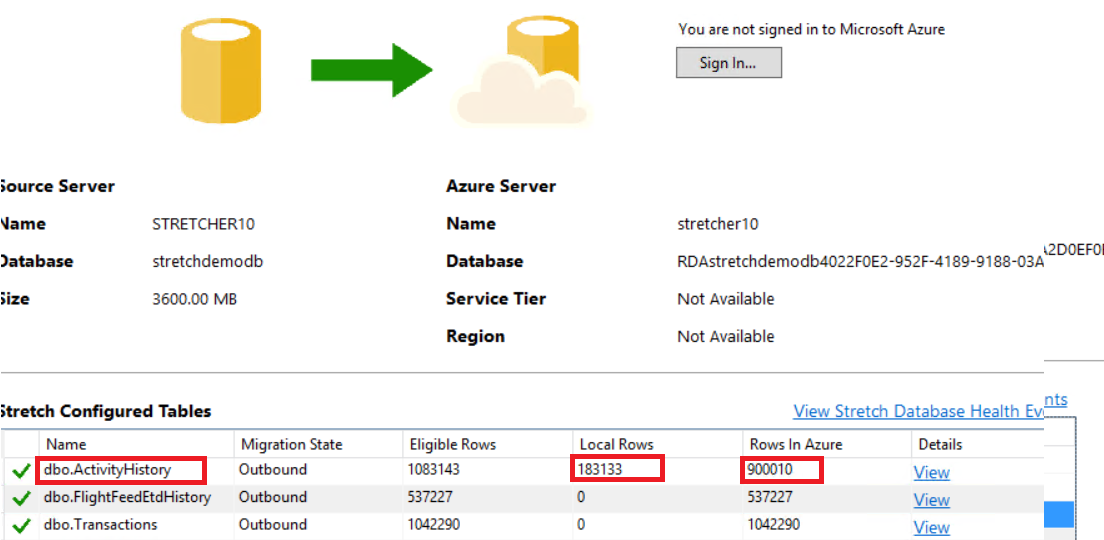
Click process to complete the transaction and be redirected to the customer profile.

**4. Customer Profile**

The purpose of this page is to highlight partially stretching a DB based on a query and also stretching Always Encrypted columns.   
You will be required to switch to the DB for this part. Below are the elements you will be interacting with.



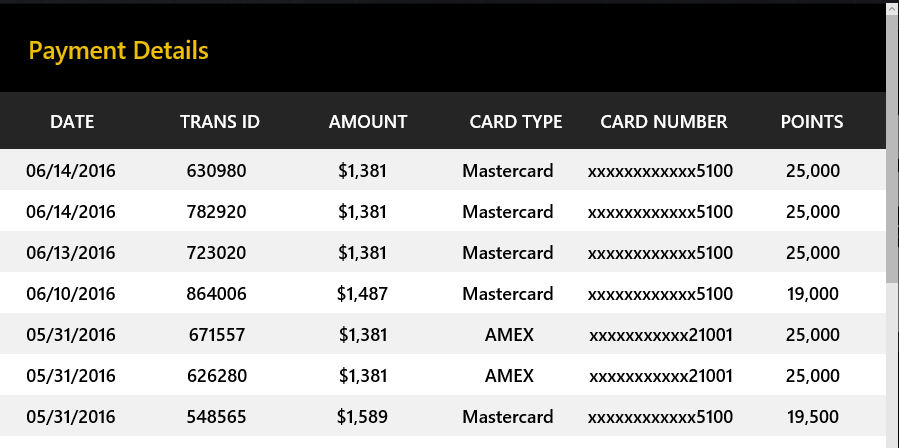
The activity history shows Jamie Campbell’s flight bookings. It is important to note that any transaction from before 1-1-2014 is being stretched to Azure based on a filter. The top flight is the one you just purchased.  
  
You will be able to see the rows that are stretched and which are local in the stretch monitor.

****

Select PAYMENT HISTORY

The history shows credit card information and purchase details that is actually encrypted in the DB. As well as being completely stretched seamlessly.

It is important to touch on that this is seamlessly handled in the API that calls the endpoint by just installing a client certificate on the API server.



Switch to the database and run the following query

Use stretchdemodb

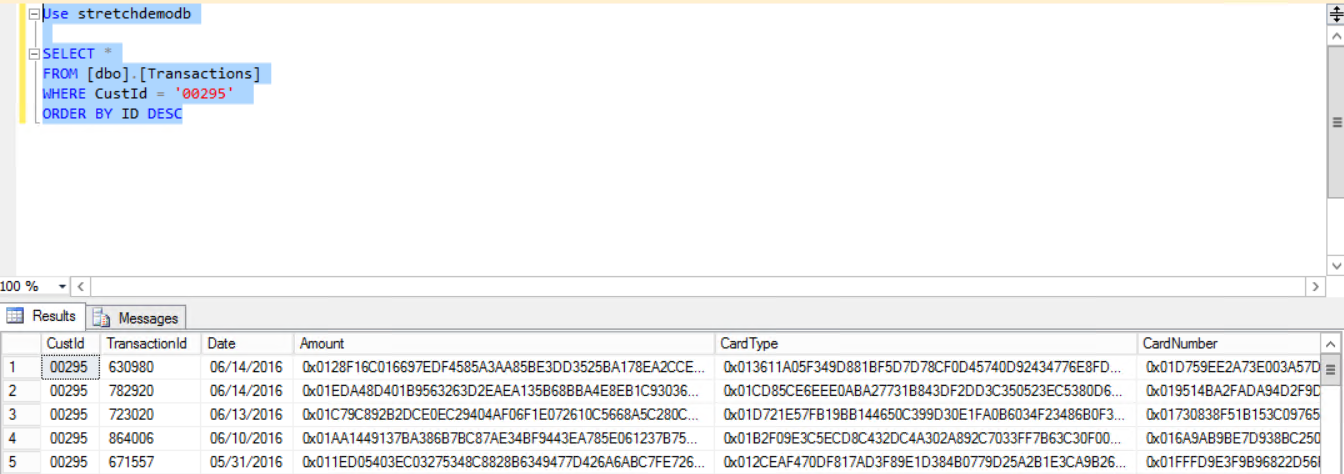
SELECT \*

FROM [dbo].[Transactions]

WHERE CustId = '00295'

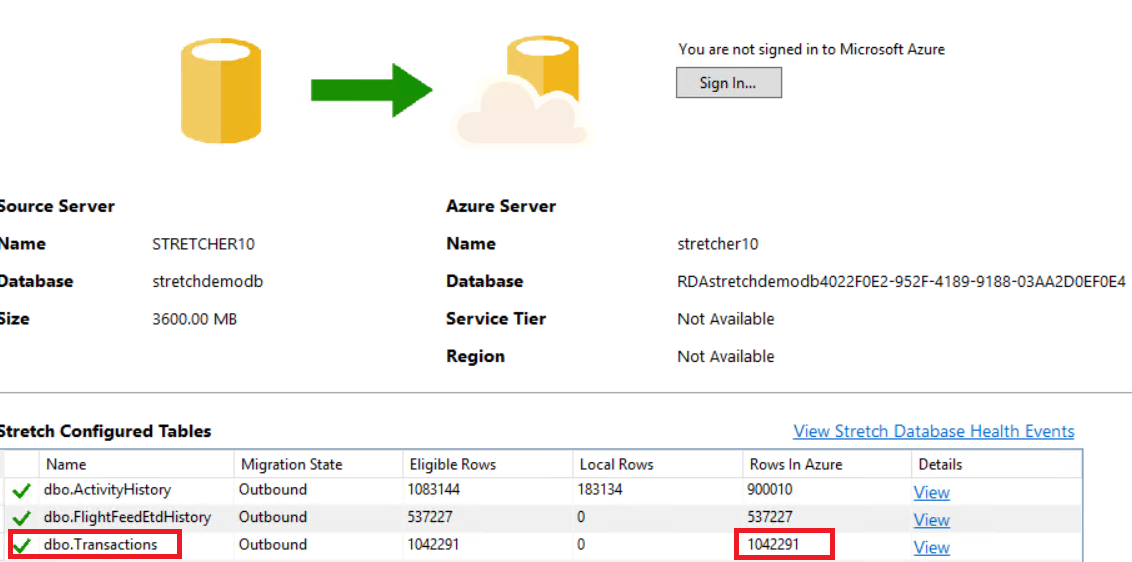
ORDER BY ID DESC

This will return all the transaction in with the latest first, you can clearly see below the returned data is encrypted.



You can switch to the monitor to also show that the dbo.transactions table is fully stretched to Azure and contains a million+ rows.

You can also point out the remote count has increased by 1 transaction  
*(1042290 -> 1042291).*

****

**End of demo.**

|  |
| --- |
| Terms of Use |

© 2015 Microsoft Corporation. All rights reserved.

By using this Hands-on Lab, you agree to the following terms:

The technology/functionality described in this Hands-on Lab is provided by Microsoft Corporation in a “sandbox” testing environment for purposes of obtaining your feedback and to provide you with a learning experience. You may only use the Hands-on Lab to evaluate such technology features and functionality and provide feedback to Microsoft.  You may not use it for any other purpose. You may not modify, copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, or sell this Hands-on Lab or any portion thereof.

COPYING OR REPRODUCTION OF THE HANDS-ON LAB (OR ANY PORTION OF IT) TO ANY OTHER SERVER OR LOCATION FOR FURTHER REPRODUCTION OR REDISTRIBUTION IS EXPRESSLY PROHIBITED.

THIS HANDS-ONLAB PROVIDES CERTAIN SOFTWARE TECHNOLOGY/PRODUCT FEATURES AND FUNCTIONALITY, INCLUDING POTENTIAL NEW FEATURES AND CONCEPTS, IN A SIMULATED ENVIRONMENT WITHOUT COMPLEX SET-UP OR INSTALLATION FOR THE PURPOSE DESCRIBED ABOVE.  THE TECHNOLOGY/CONCEPTS REPRESENTED IN THIS HANDS-ON LAB MAY NOT REPRESENT FULL FEATURE FUNCTIONALITY AND MAY NOT WORK THE WAY A FINAL VERSION MAY WORK.  WE ALSO MAY NOT RELEASE A FINAL VERSION OF SUCH FEATURES OR CONCEPTS.  YOUR EXPERIENCE WITH USING SUCH FEATURES AND FUNCITONALITY IN A PHYSICAL ENVIRONMENT MAY ALSO BE DIFFERENT.

**FEEDBACK**.  If you give feedback about the technology features, functionality and/or concepts described in this Hands-on Lab to Microsoft, you give to Microsoft, without charge, the right to use, share and commercialize your feedback in any way and for any purpose.  You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft software or service that includes the feedback.  You will not give feedback that is subject to a license that requires Microsoft to license its software or documentation to third parties because we include your feedback in them.  These rights survive this agreement.

MICROSOFT CORPORATION HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THE HANDS-ON LAB , INCLUDING ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT.  MICROSOFT DOES NOT MAKE ANY ASSURANCES OR REPRESENTATIONS WITH REGARD TO THE ACCURACY OF THE RESULTS, OUTPUT THAT DERIVES FROM USE OF THE VIRTUAL LAB, OR SUITABILITY OF THE INFORMATION CONTAINED IN THE VIRTUAL LAB FOR ANY PURPOSE.