# Big Data and Visualization

Wednesday, September 13, 2017 1:30 PM

Analyze weather data using Azure Machine Learning, summarize it in Azure SQL Data Warehouse, and visualize it using Power BI.

In this workshop, attendees will build an end-to-end solution to predict flight delays taking into account the weather forecast using Power BI, Azure SQL Data Warehouse, an Azure Machine Learning.

**What You Will Learn**

- Azure Data Platform
- Azure SQL Data Warehouse
- Azure Machine Learning
- Power BI
- Advanced Analytics

**Ideal Audience**

- CIOs
- VPs and Directors of Business Intelligence
- IT Managers
- Data Architects and DBAs
- Data Analysts and Data Scientists

# Overview

In this workshop, attendees will build an end-to-end solution to predict flight delays taking into account the weather forecast using Power BI, Azure SQL Data Warehouse, an Azure Machine Learning. AdventureWorks Travel (AWT) provides concierge services for business travelers. In an increasingly crowded market, they are always looking for ways to differentiate themselves and provide added value to their corporate customers. They are looking to pilot a web app that their internal customer service agents can use to provide additional information useful to the traveler during the flight booking process. They want to enable their agents to enter in the flight information and produce a prediction as to whether the departing flight will encounter a 15-minute or longer delay, taking into account the weather forecasted for the departure hour.

**Time Estimate:** 5.0 hours

# Requirements

## Setup Requirements

- A corporate email address (e.g., your @microsoft.com email)
- Microsoft Azure Subscription must be pay-as-you-go or MSDN
- Local machine or a virtual machine configured with:
    - Visual Studio 2015 Community Edition or later
    - Azure SDK 2.8.2 for Visual Studio
    - Azure PowerShell 1.0.0 or later

## Additional Requirements

You will need a subscription to Microsoft Azure. Please see the next page for how to create a trial subscription.

# Azure Registration

## Azure

Direct your browser to https://azure.microsoft.com/en-us/free/ and begin by clicking on the green button that reads **Start free**.

1. In the first section, complete the form in its entirety. Make sure you use your *real* email address for the important notifications.

2. In the second section, enter a *real* mobile phone number to receive a text verification number. Click send message and re-type the received code.

3. Enter a valid credit card number. **NOTE:** You will *not* be charged. This is for verification of identity only in order to comply with federal regulations. Your account statement may see a temporary hold of $1.00 from Microsoft, but, again, this is for verification only and will "fall off" your account within 2-3 banking days.

4. Agree to Microsoft's Terms and Conditions and click **Sign Up**.

This may take a minute or two, but you should see a welcome screen informing you that your subscription is ready. The Azure subscription is good for up to $200 of resources for 30 days. After 30 days, your subscription (and resources) will be suspended unless you convert your trial subscription to a paid one. And, should you choose to do so, you can elect to use a different credit card than the one you just entered.

Congratulations! You've now created an Azure tenant and subscription!

# Setup

## Azure Subscription

As stated in the requirements section, the workshop requires an active Azure subscription.

> **Recommendation**
>
> It is recommended that you do not use an Azure subscription that is currently being used for production. The CLI will create it's own resource groups, but it is not the best practice to utilize production environments for testing and workshops, such as this.

For best results, it is recommended that you setup register for the trial subscription as outlined on the previous page.

## **Exercise 0:** Before the workshop

Duration: 60 mins Synopsis: Before attending the workshop, you should follow these steps to prepare your environment for an efficient day.

### Task 1: Provision Power BI

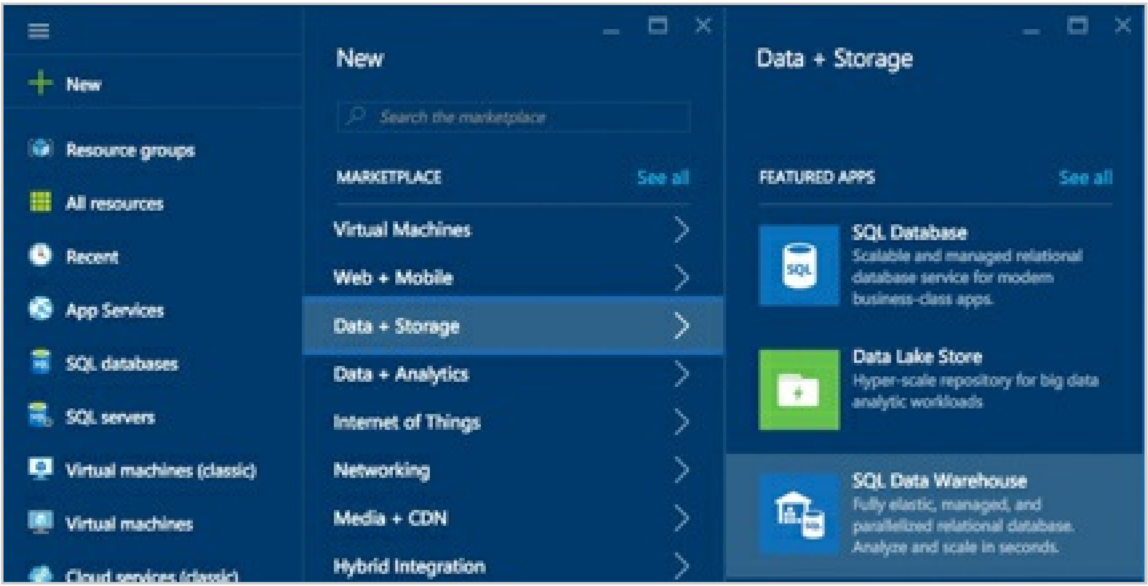
1. If you do not already have a Power BI account, go to https://www.powerbi.com.
2. On the page, enter your work email address (it should be the same account as the one you use for your Azure subscription) and select **Use it free**.


3. Follow the on-screen prompts and your Power BI environment should be ready within minutes. You can always return to it via https://app.powerbi.com.

## Task 2: Provision Azure SQL Data Warehouse

Using the Azure Portal, provision a new instance of SQL Data Warehouse.

1. Click **+New**, select **Data + Storage**, **SQL Data Warehouse**.

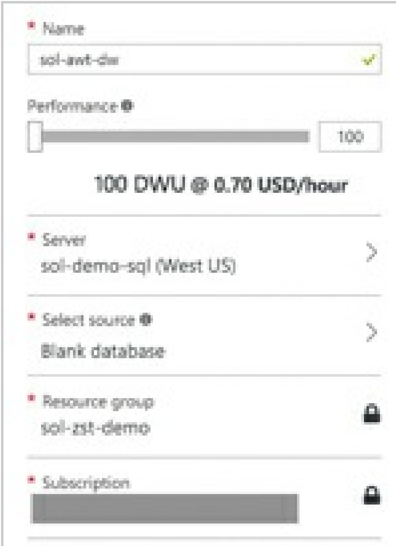


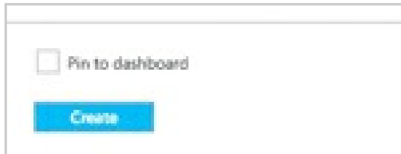

2. Provide a Name for the SQL Data Warehouse.


3. Set the **Performance** to **100 DWU**. (You will not need any more for this workshop.)


4. Select Server.


5. Create a new server or use an existing server as desired.
6. Select **Create**.

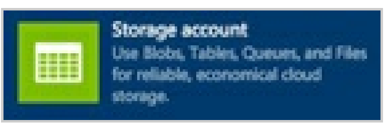


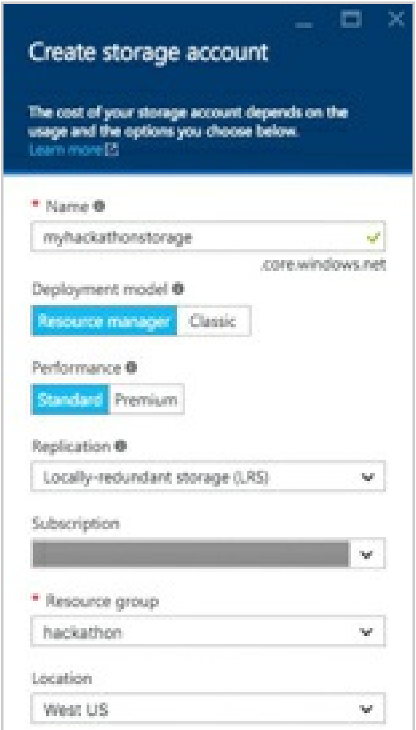

## Task 3: Provision a Storage Account

Using the Azure Portal, provision a new Azure Storage Account to use for this workshop.

1. Click **+New**, select **Data + Storage**, **Storage Account**.




2. Provide a Name for the storage account.
3. For the resource group, add it to the Resource Group you are using for this workshop.
4. For the location, choose the same Location as your SQL Data Warehouse.




5. Select **Create**.

# Environment Setup

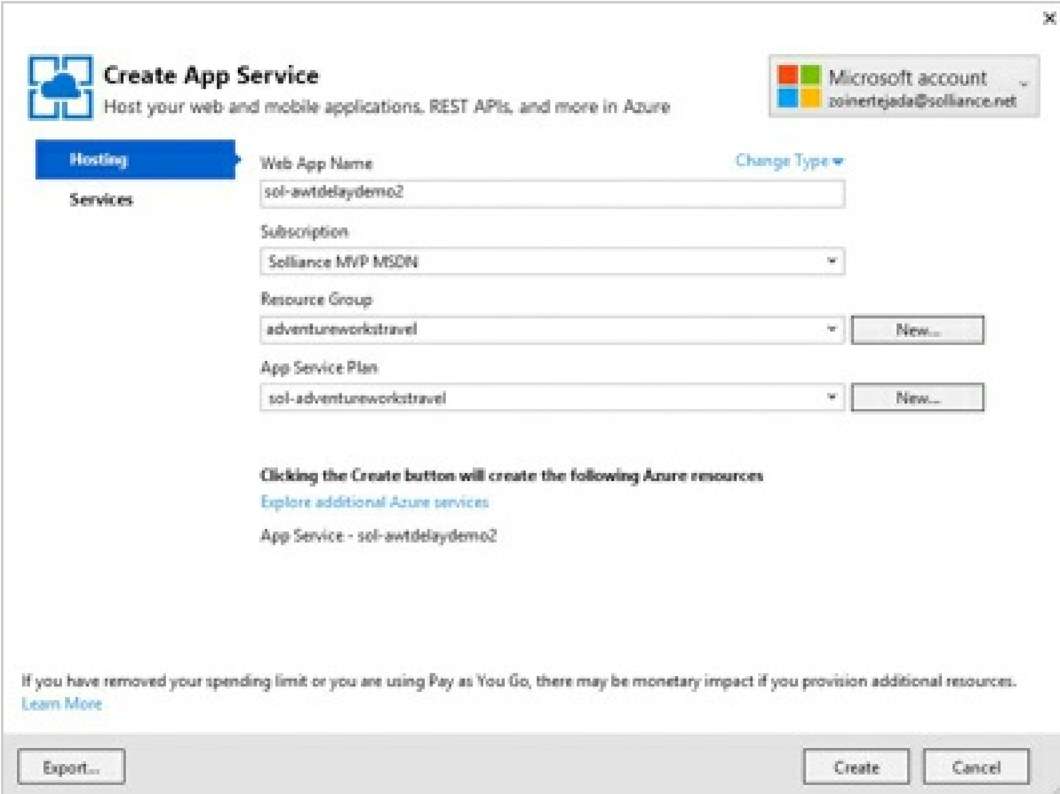
## **Exercise 1:** Environment setup

**Duration:** 20 minutes

**Synopsis:** In this exercise, attendees will download the start project files and take a tour of the provided solution.

### Task 1: Download and unzip the sample

1. Download the sample project from the following link: http://bit.ly/238cJDY.
2. Extract the ZIP file.
3. Open the solution AdventureWorksTravel.sln using Visual Studio.
4. In Solution Explorer, right-click the Project and select **Publish**.
5. In the first dialog box, under Select a Publish Target, click **Microsoft Azure App Service**.
6. Sign in to your account when prompted.
7. Click **New**.
8. In the Create App Service dialog box, provide a name for the web app. Choose your subscription and choose to use an existing App Service Plan (if you already have one) or create a new one. If you choose to create a new App Service plan, provide a name for that new App Service plan. Similarly, if creating a new App Service plan, choose an existing Resource Group or create a new one and provide a name for the new resource group (e.g., workshop).




9. Click **Create**.

10. Click **Publish**. After a few moments, the initial version of the web app should appear in the browser.

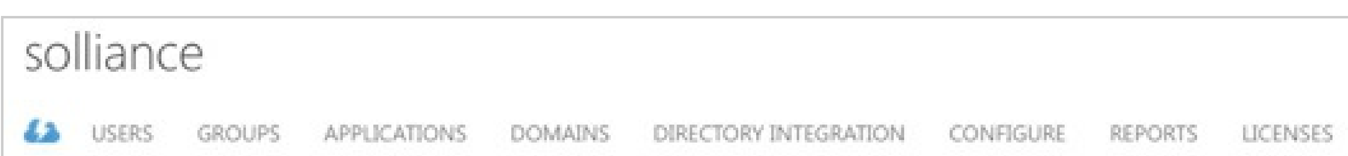
### Task 2: Register Web App with Azure Active Directory

1. Login in to the existing portal https://manage.windowsazure.com/ using the same credentials you used for Power BI.

2. Click the **active directory** tab.

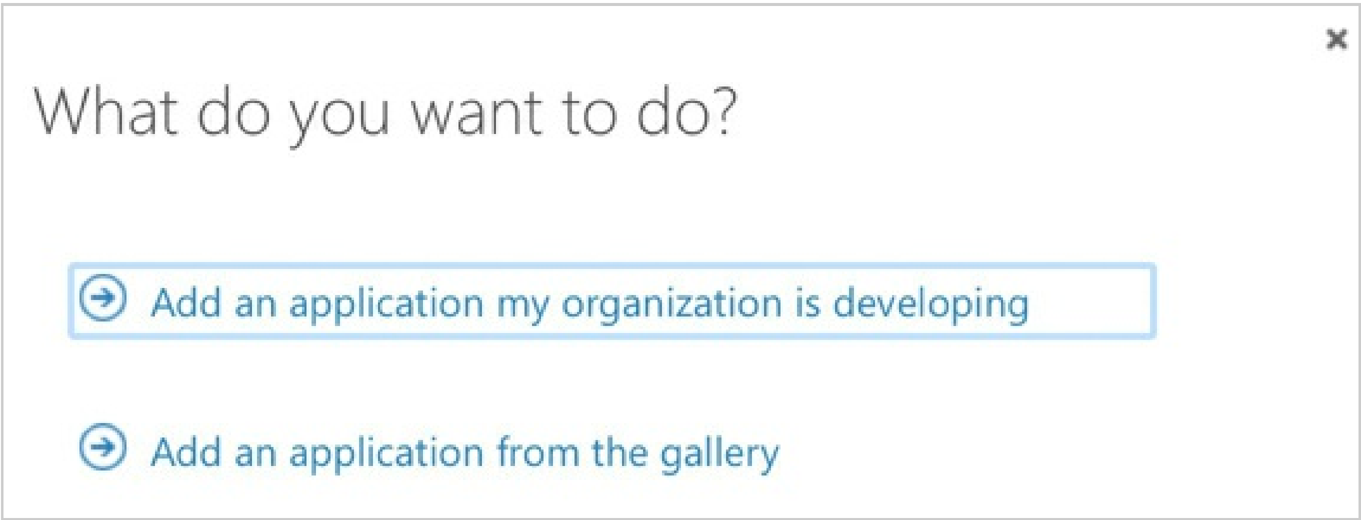

3. Click the name of your default directory in the grid.

4. Click the **Applications** tab.

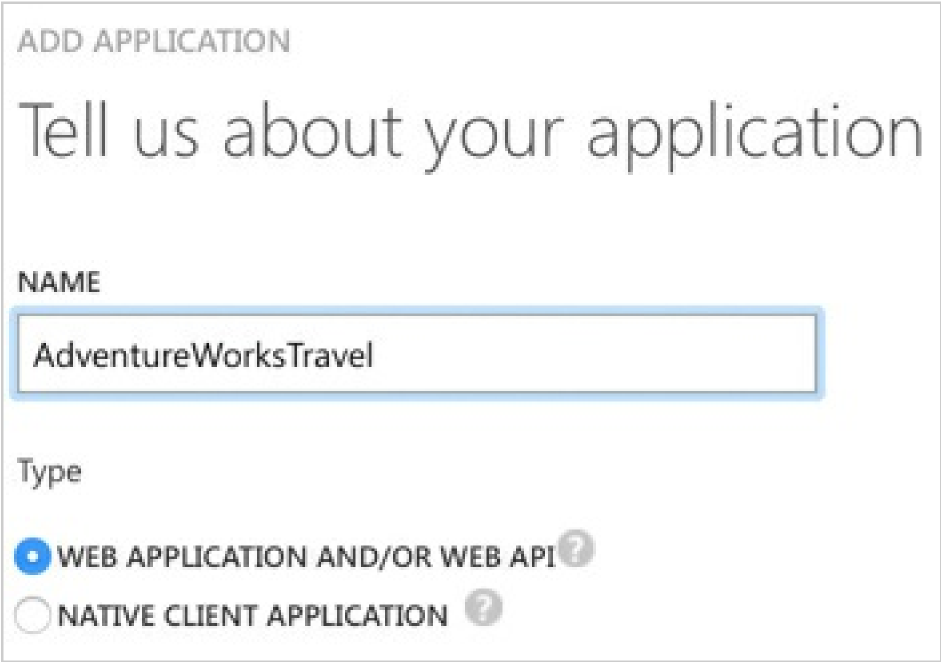



5. In the command bar at the bottom, click **Add**.

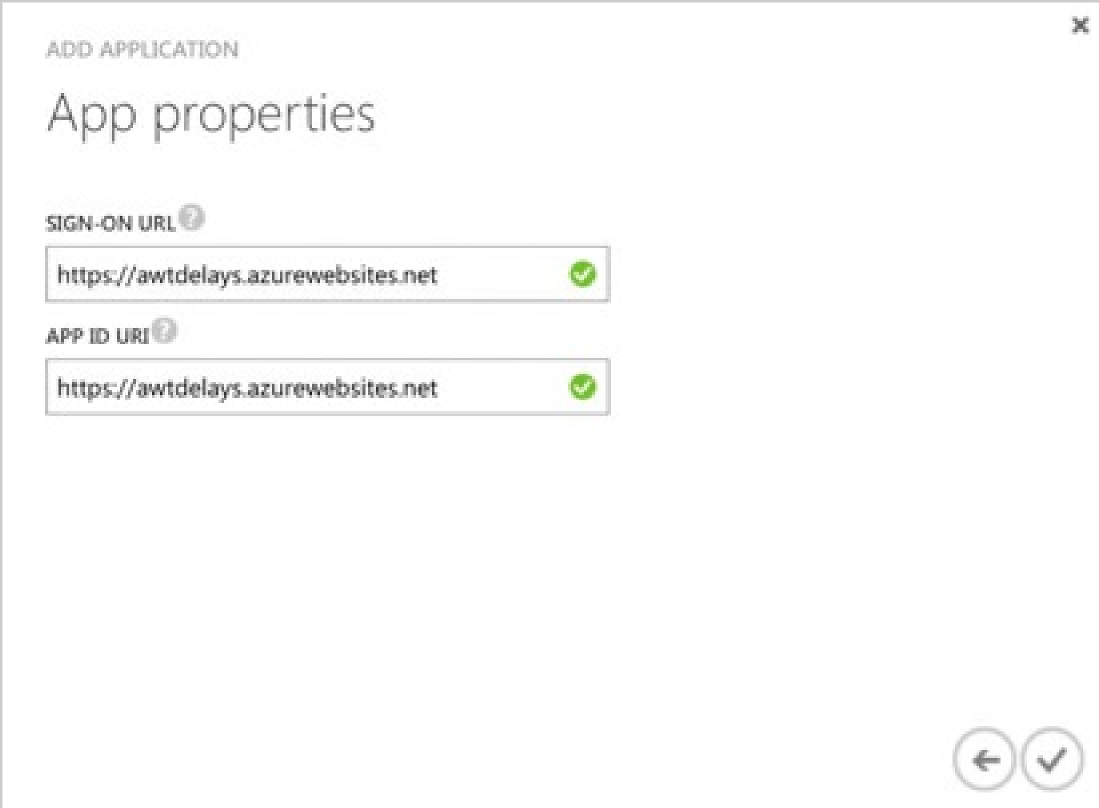

6. In the dialog box, click **Add an application my organization is developing**.




7. In the next dialog box, enter a unique name for your application (e.g., “AdventureWorksTravelYourMSAlias”) and leave the Web Application radio button selected.

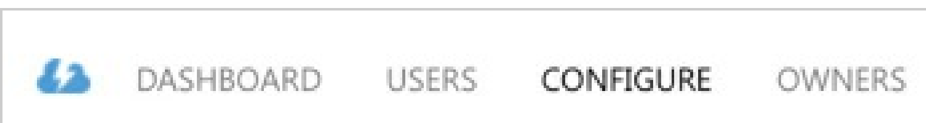



8. Click the right arrow button to move to the next screen.
9. For both fields, provide the URL to your newly deployed web app (make sure the URLs start with https).

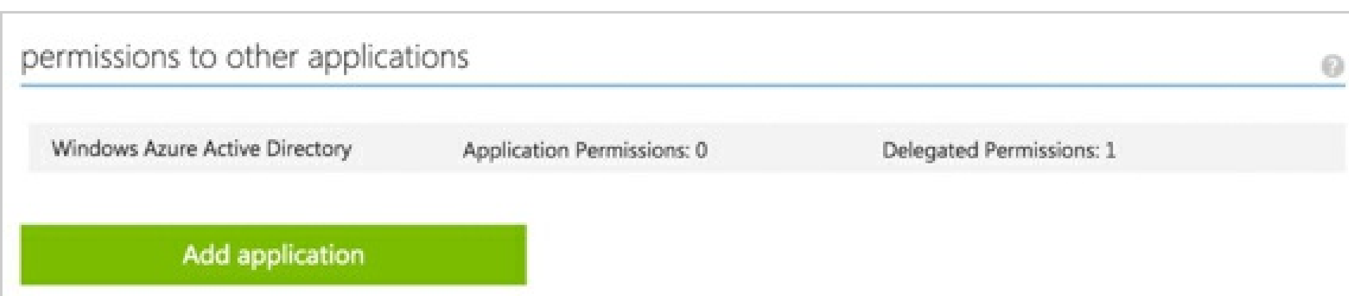



10. Click the checkmark button.

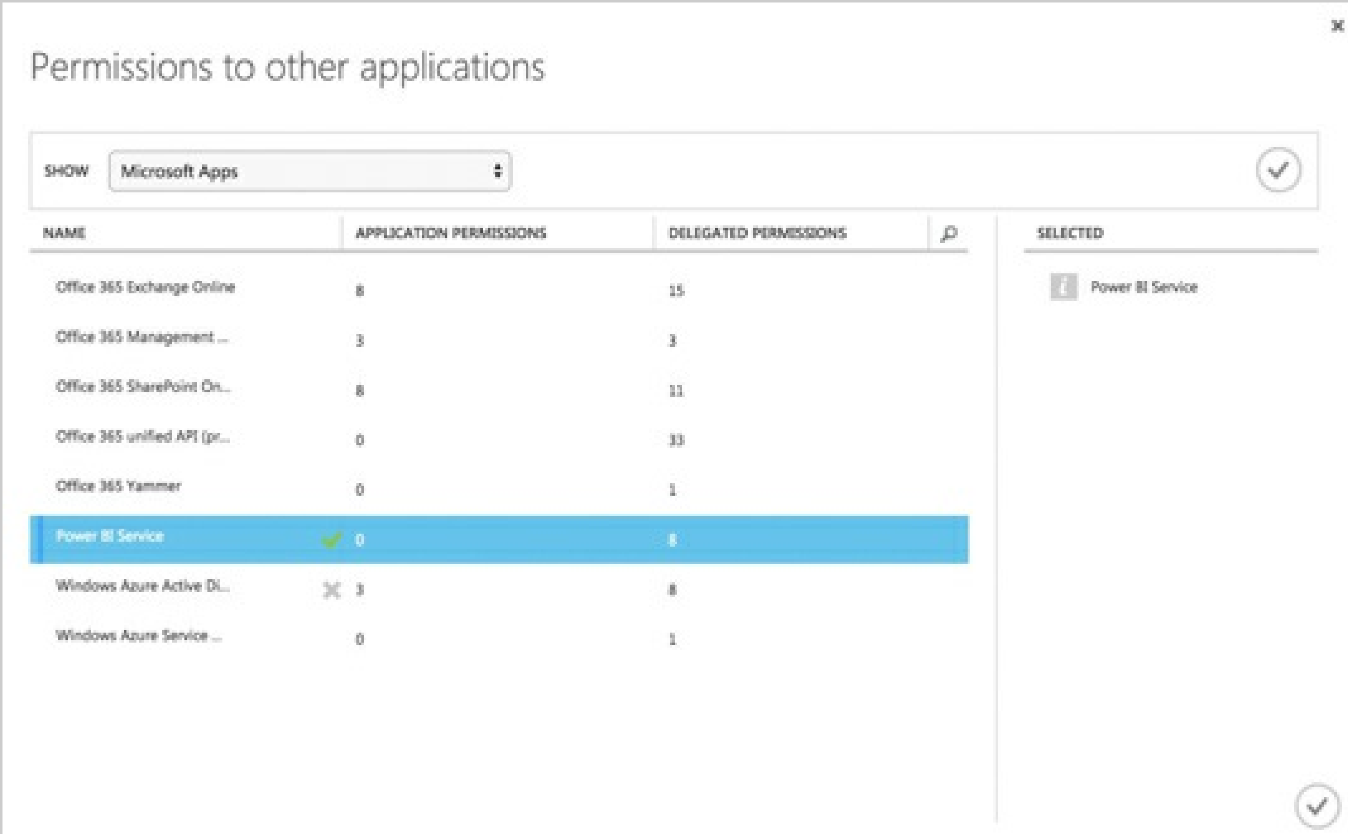
11. On the dashboard for your application, click the **Configure** tab.




12. Scroll down to the permissions to other applications section and click **Add application**



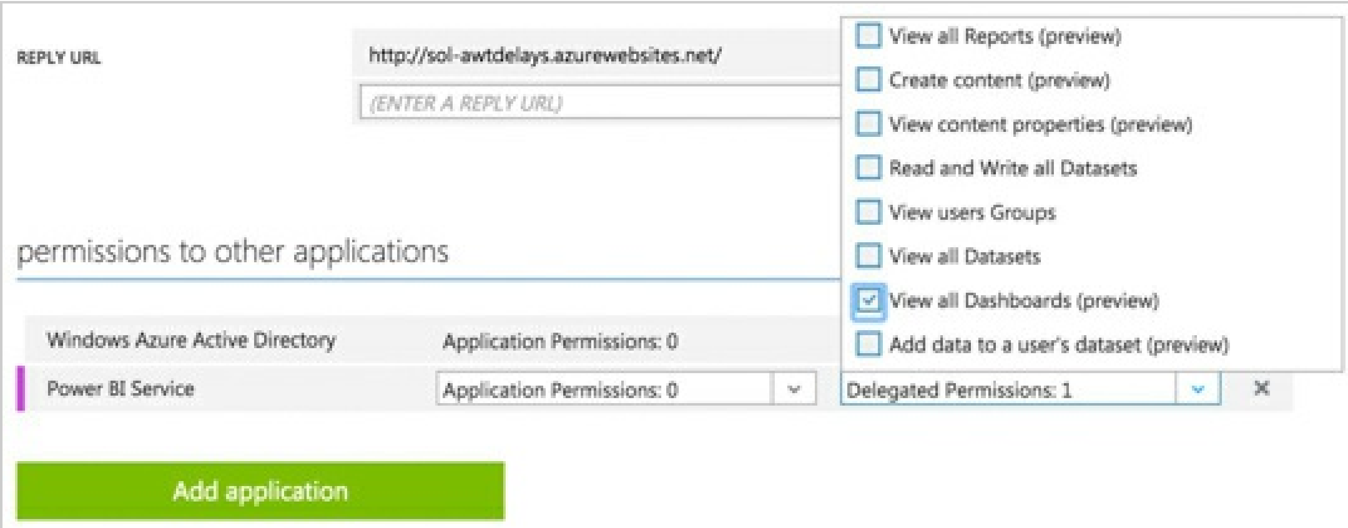

13. In the dialog click on Power BI Service so that it appears in the list on the right and then click the checkmark.




**Power BI**

If Power BI Service does not show up on the list of applications, visit [powerbi.microsoft.com](https://powerbi.microsoft.com/en-us/documentation/powerbi-admin-free-with-custom-azure-directory/) to learn how to correct this. You will need to log in to Power BI as an organizational user within your directory before the service shows up in AAD.

14. In the Power BI Service row that appears, click on **Delegated permissions** and check **View all Dashboards**.




15. Scroll up to the Keys section.

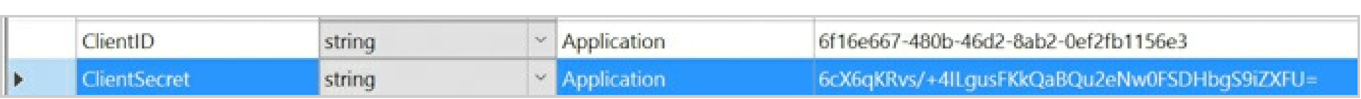
16. Click the **Select duration** drop-down and select **1 year**.


17. Click **Save** in the command bar at the bottom.

18. Copy the key value that appears after the save completes.

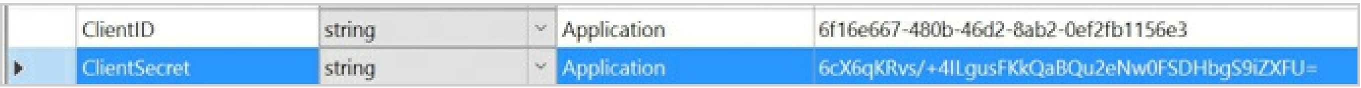

19. Return to the sample app in Visual Studio, right-click the project in Solution Explorer and select **Properties**.

20. Click the **Settings** tab. For the value of the ClientSecret setting, paste the key you just copied.




21. Return to the Configure page and copy the Client ID value.


22. In the sample Settings, paste the value of the ClientID in the ClientID setting.

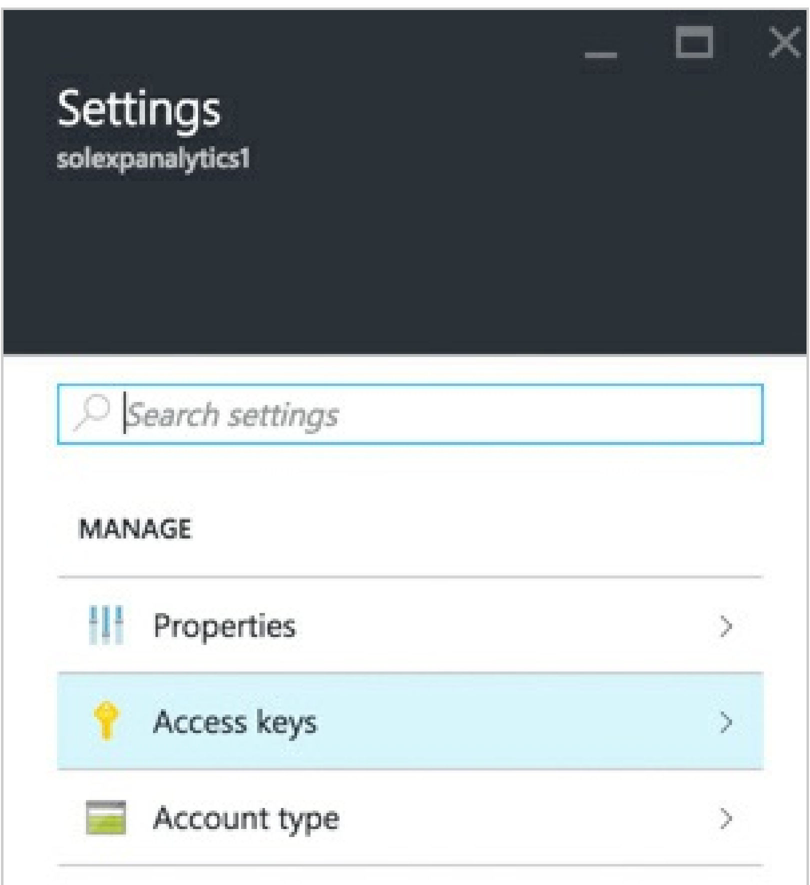



23. Set the value of the RedirectURL to the URL of your newly deployed web app (including HTTPS), and append /Redirect to the end.

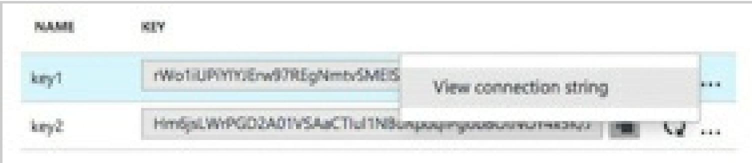

24. Save the project.

## Task 3: Update sample app with Storage Account credentials

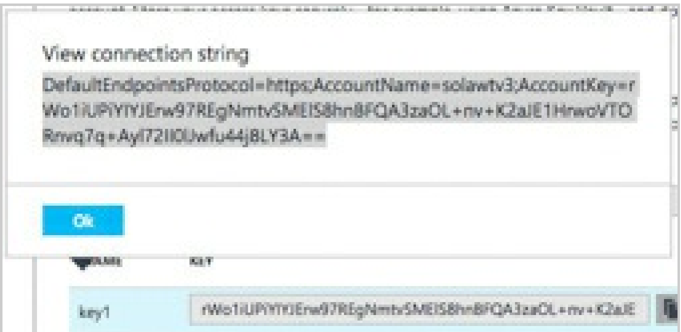
1. Using the New Portal, navigate to the blade for your Storage Account.
2. Click **Access Keys**.




3. To the right of key1 in the table, click the ellipses (...) and select View connection string.




4. Copy the displayed **Connection String**. Be sure to copy the entire Connection String, not the Access key only.




5. Return to the settings pane of the sample web app, and paste this connection string into the value field of the ML_StorageAccount setting.


6. Save the project.

# Check the Weather

## Exercise 2: Check the weather

**Duration:** 20 mins

**Synopsis:** In this exercise, attendees will integrate the 10-day weather forecast into the sample solution.

### Task 1: Register for trial account in WeatherUnderground.com

1. To retrieve the 10-day hourly weather forecast, you will use an API from WeatherUnderground.com. There is a free developer version that provides you access to the API you need for this workshop.

2. Navigate to http://www.wunderground.com/weather/api/.

3. Click **Explore My Options**.


4. On the Get Your API Key page, select **Anvil Plan**.

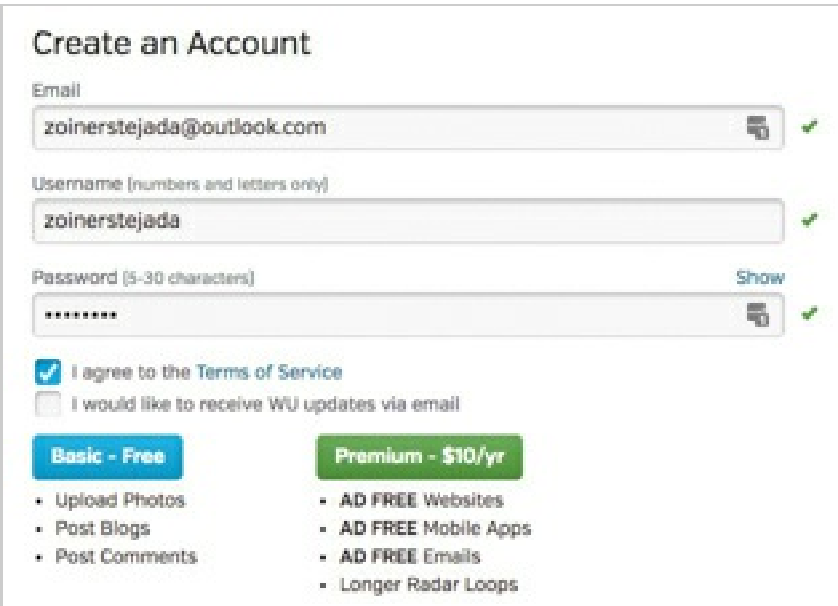

5. Scroll down until you see the area titled How much will you use our service? Ensure Developer is selected.

How much will you use our service?

| | Monthly Pricing | Calls Per Day | Calls Per Minute |
|---|---|---|---|
| Developer | $0 | 500 | 10 |
| Drizzle | $300 | 5000 | 100 |
| Shower | $600 | 100,000 | 1000 |
| Downpour | Get in touch for more than 100,000 calls per day. | | |

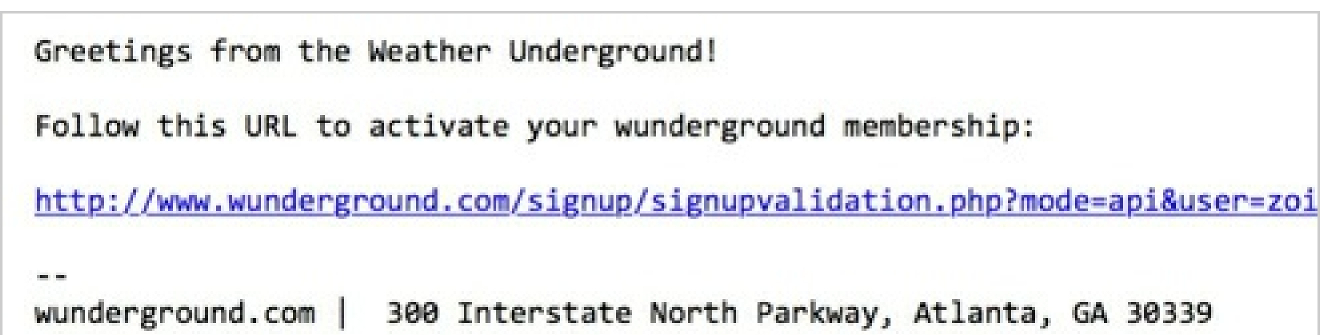
6. Click **Purchase Key**.


7. Complete the Create an Account form by providing your email, username and a password and agreeing to the terms. Click Basic – Free.



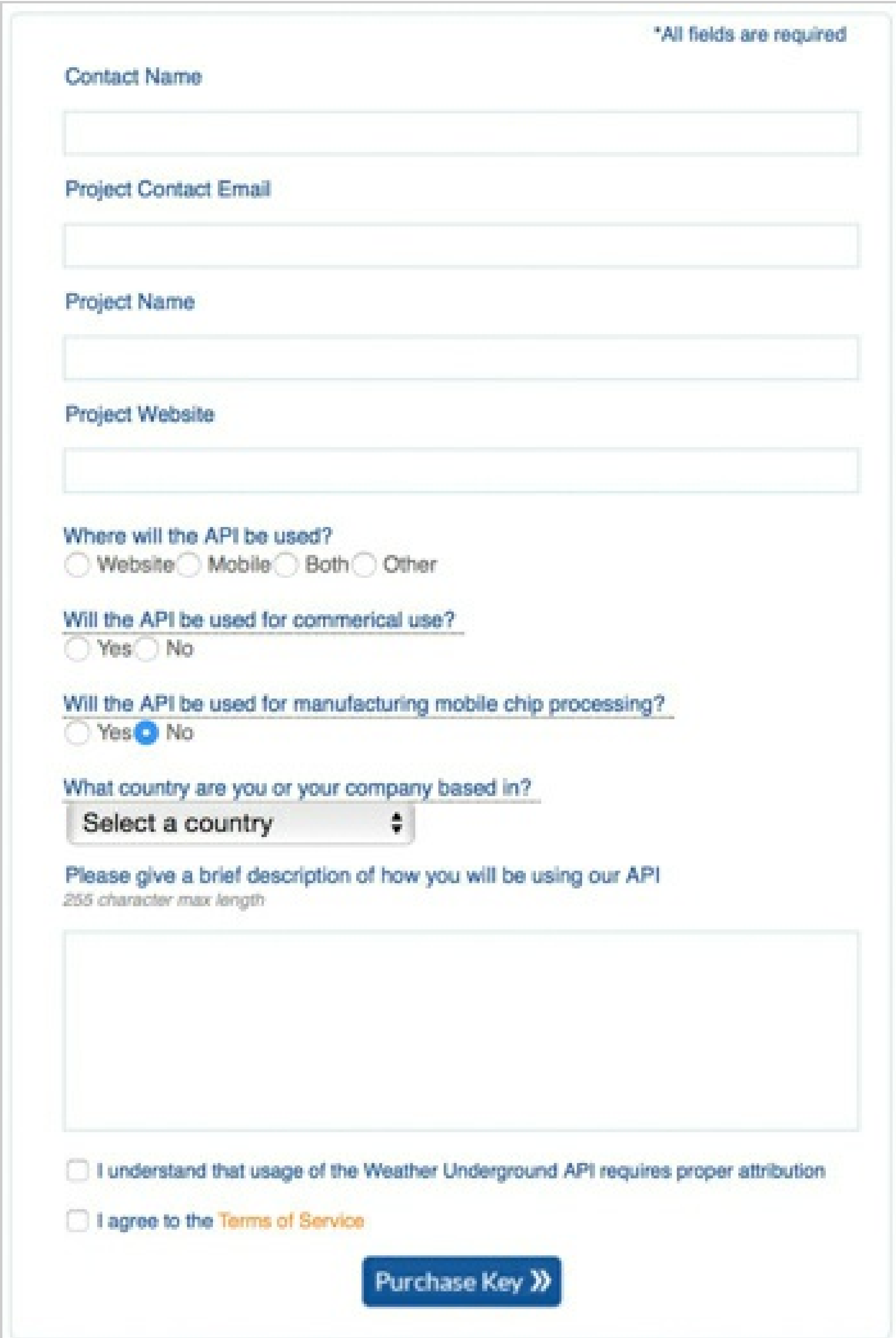

8. Back on the Get Your API Key page, re-select Anvil and click Purchase Key.

9. In a few moments you should receive a confirmation email at the email address you provided. Click the link found within the email.

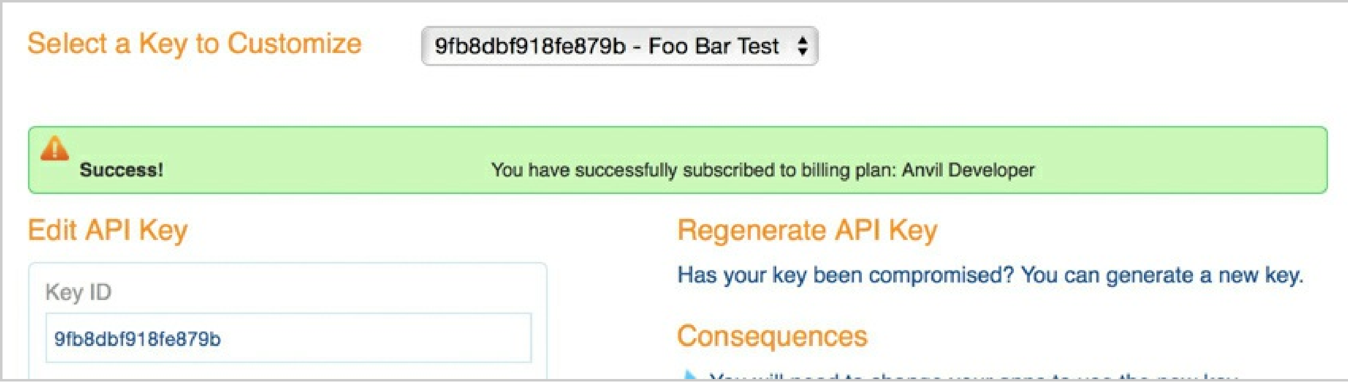



10. On the page that loads, you will see a message like the following:


11. Complete the brief contact form. When answering where will the API be used, select **Website**. For Will the API be used for commercial use, select **No**. Click **Purchase Key**.



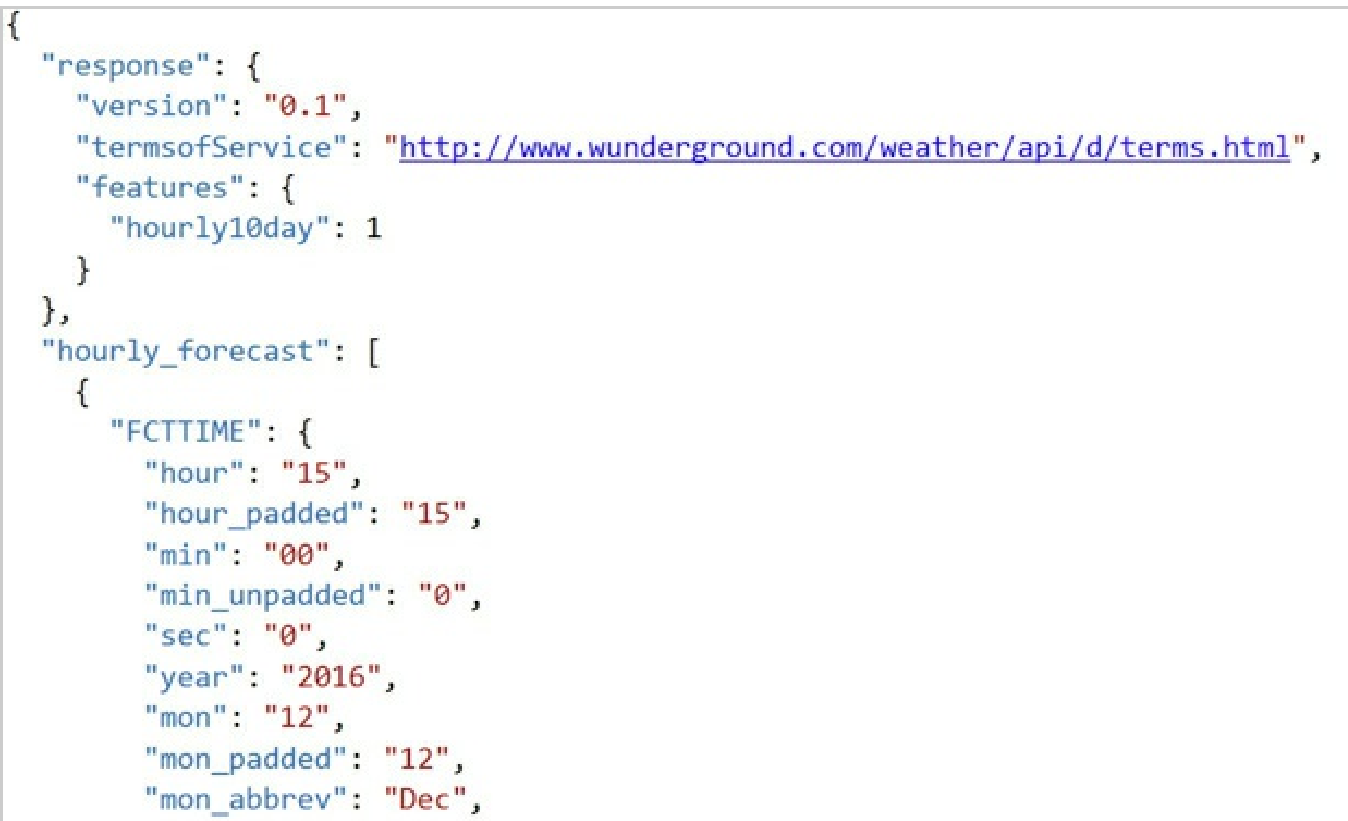

12. You should be taken to a page that displays your key, similar to the following:




13. Take note of your API Key. It is available from the text box labeled **Key ID**.

14. To verify that your API Key is working, modify the following URL to include your API Key: http://api.wunderground.com/api/<YOURAPIKEY>/hourly10day/q/CA/SEATAC.json.

15. Open your modified link in a browser, you should get a JSON result showing the 10-day, hourly weather forecast for the Seattle-Tacoma International Airport.



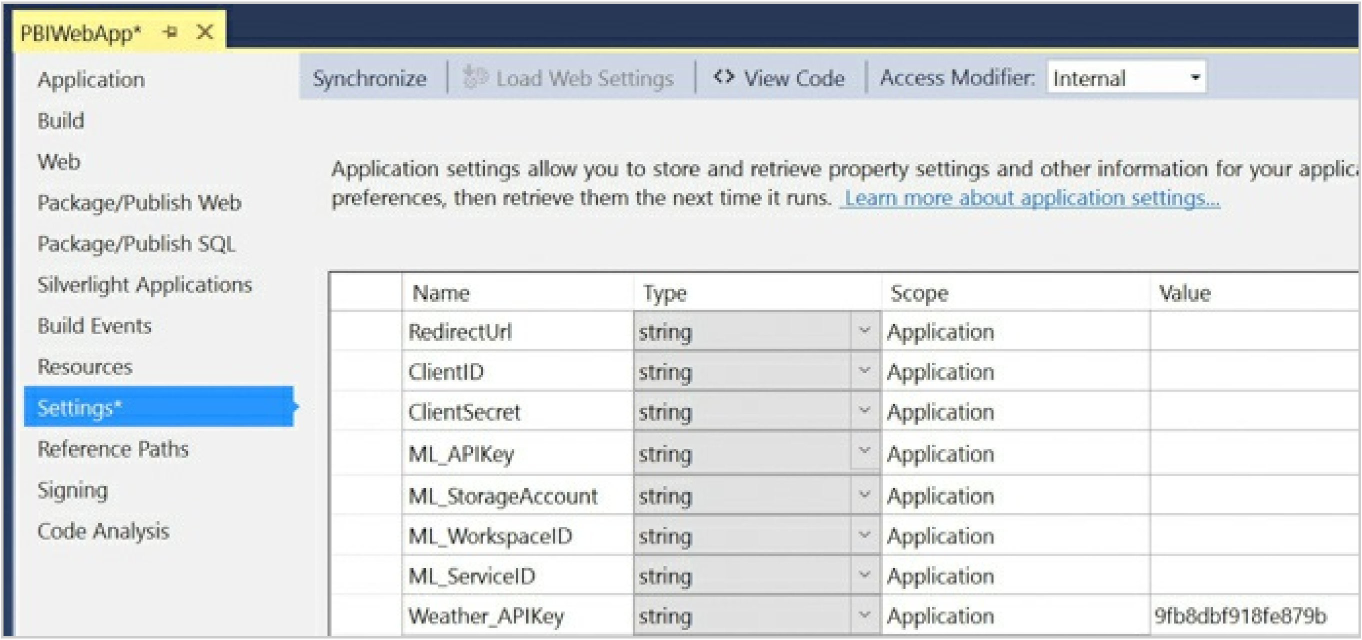
```
{
  "response": {
    "version": "0.1",
    "termsofService": "http://www.wunderground.com/weather/api/d/terms.html",
    "features": {
      "hourly10day": 1
    }
  },
  "hourly_forecast": [
    {
      "FCTTIME": {
        "hour": "15",
        "hour_padded": "15",
        "min": "00",
        "min_unpadded": "0",
        "sec": "0",
        "year": "2016",
        "mon": "12",
        "mon_padded": "12",
        "mon_abbrev": "Dec",
```


## Task 2: Update sample Web App with API Key

1. Next, you will update the sample app with your Weather Underground API Key.

2. With the sample app open in Visual Studio, in Solution Explorer, right-click the project and select **Properties**.

3. Click the **Settings** tab.




4. Edit the value for the Weather_APIKey and click **Save**.

5. In Visual Studio Solution Explorer, right-click your project, select **Publish**, and then click **Publish**.

6. On your deployed web app, change the date of the flight to tomorrow's date (or another future date no further than 10 days away).

7. Click **Predict Delay**. You should see the weather icon display. If you hover over the icon, you will get the description of the weather conditions forecasted. At this point, Delay Prediction should display not configured. You will set Delay Prediction up shortly.

# Building a ML Model

## **Exercise 3:** Building a ML Model

**Duration:** 90 mins

**Synopsis:** In this exercise, attendees will implement a classification experiment. They will load the training data from their local machine into a dataset. Then they will explore the data to identify the primary components they should use for prediction, and use two different algorithms for predicting the classification. They will evaluate the performance of both and algorithms choose the algorithm that performs best. The model selected will be exposed as a web service that is integrated with the sample web app.

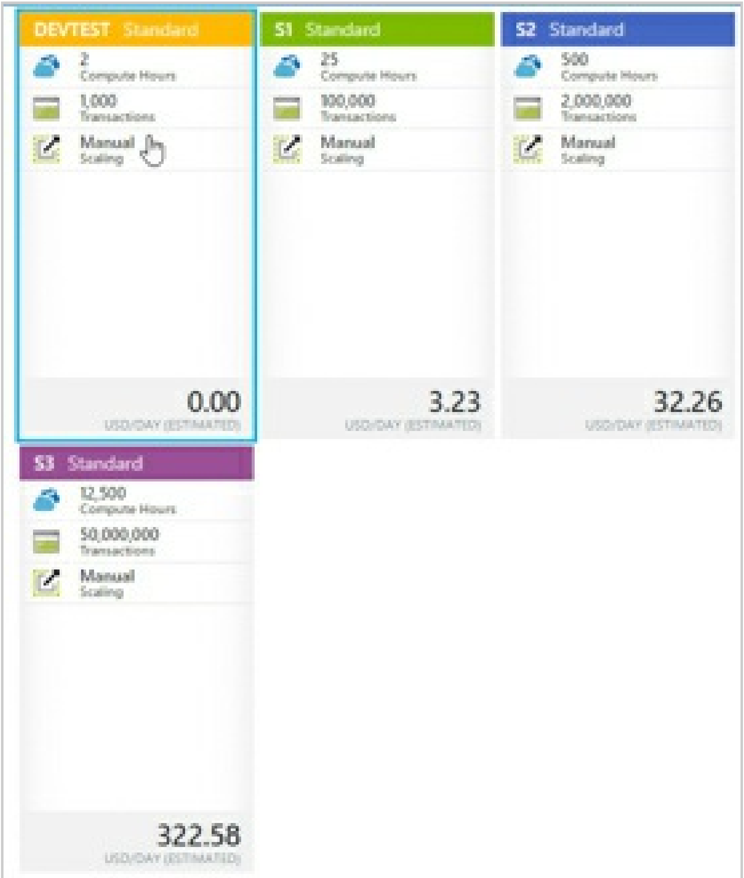
### Task 1: Create ML Workspace

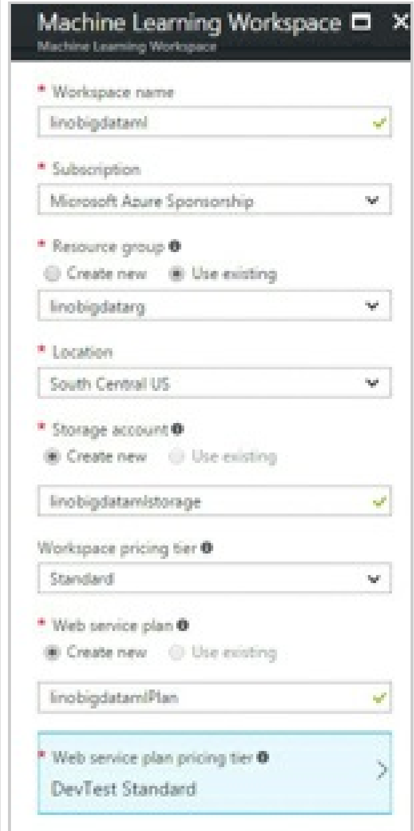
1. Login to the existing Azure portal.

2. Click **+ New**, search for **Machine Learning Workspaces**.

3. Click on Machine Learning Workspace to open the Workspaces blade and click on Create.

4. Enter a workspace name, a subscription, the existing Resource Group, location nearest you, create a new storage account, choose the Standard pricing tier, create a new Web Service plan and finally choose DevTest Standard pricing tier for free.




5. Click **Create**.

6. Navigate to your ML workspace by clicking **Machine Learning** Workspace in the Resource Group blade.

7. In the additional links, click **Launch Machine Learning Studio** to open your workspace using the development environment ML Studio.


## Task 2: Upload the Sample Datasets

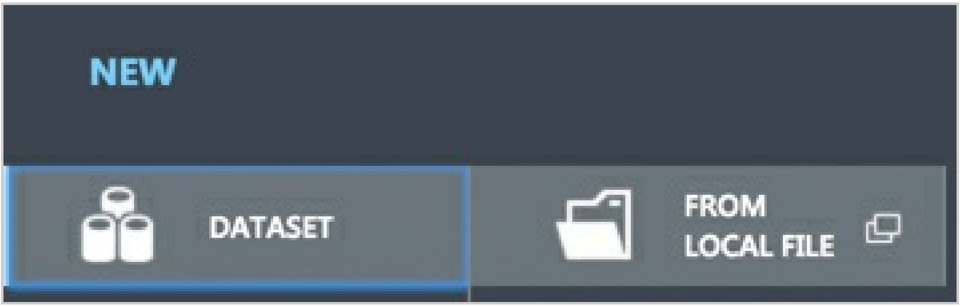
1. Before you begin creating a machine learning experiment, there are three datasets you need to load.

2. Download the three CSV sample datasets from here: http://bit.ly/1Hrm5es

3. Extract the ZIP and verify you have the following files:

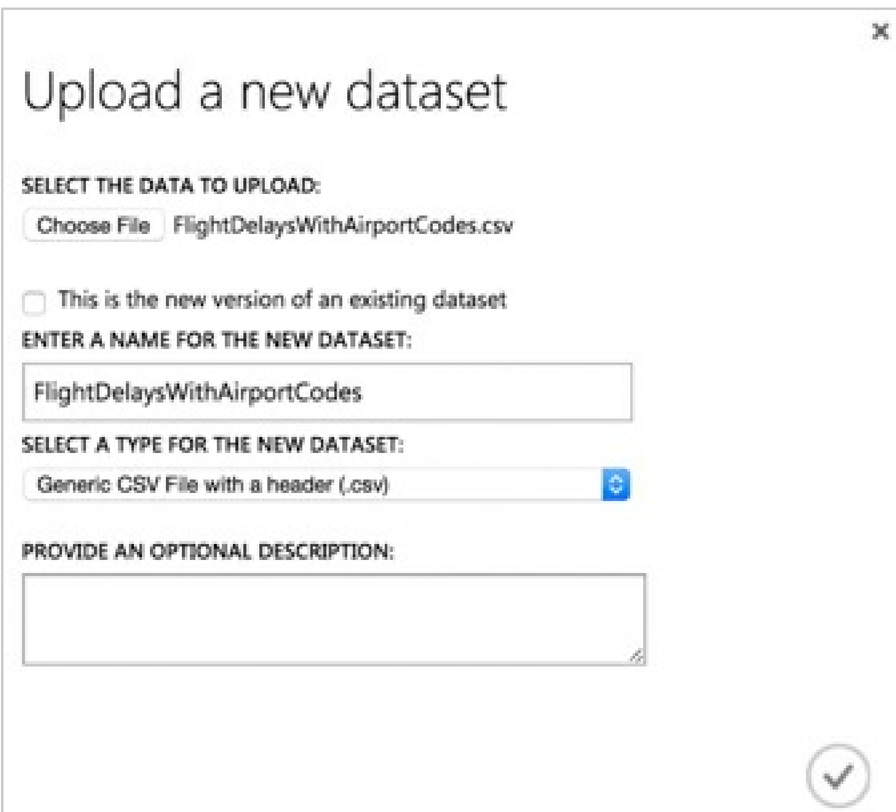
- FlightDelaysWithAirportCodes.csv
- FlightWeatherWithAirportCodes.csv
- AirportCodeLocationLookupClean.csv

4. Within Machine Learning Studio, click **+ NEW** at the bottom, point to **Dataset**, and select **From Local File**.




5. In the dialog that appears, click **Choose File** and browse to the FlightDelaysWithAirportCodes file and click **Open**.

6. Change the name of the dataset to "FlightDelaysWithAirportCodes" and click the checkmark to upload the data into a new dataset.

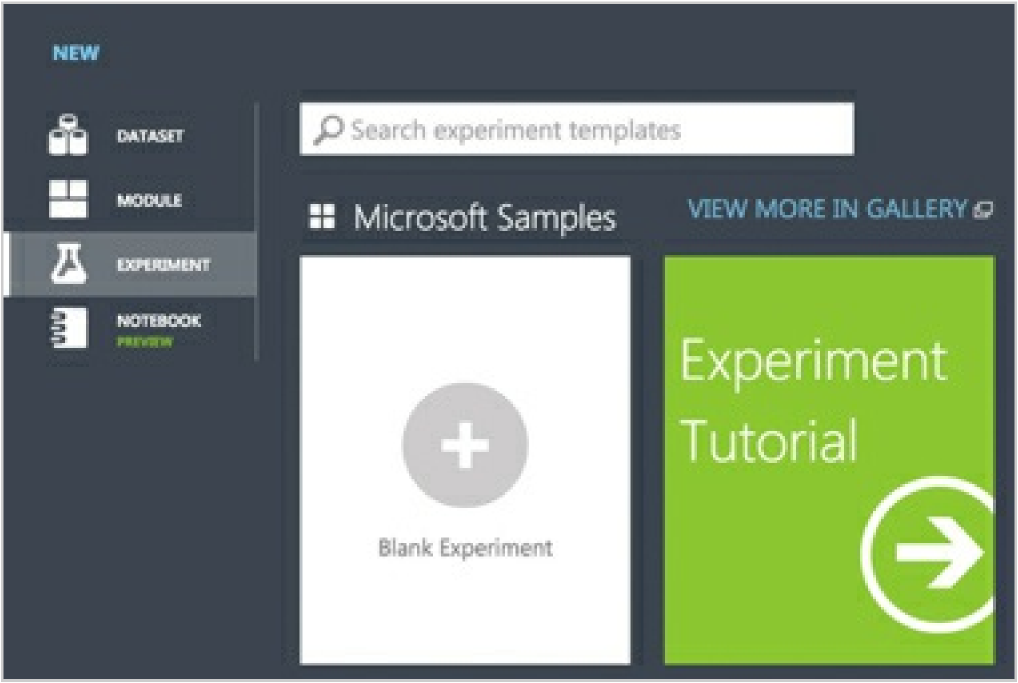



7. Repeat the previous step for the FlightWeatherWithAirportCodes and AirportCodeLocationsClean, setting the name for the dataset in a similar fashion.

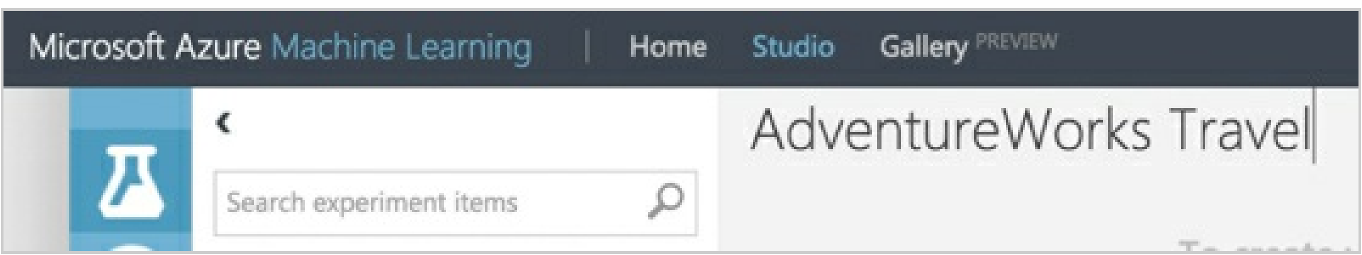
### Task 3: Start a new experiment

1. Click **+ NEW** in the command bar.

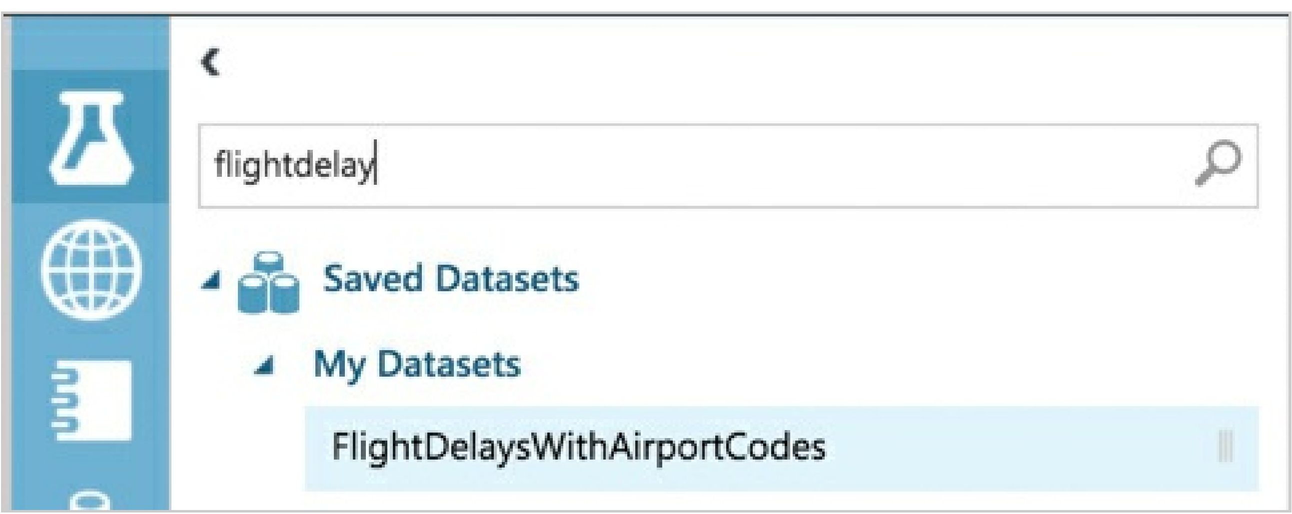
2. In the options that appear, click **Blank Experiment**.



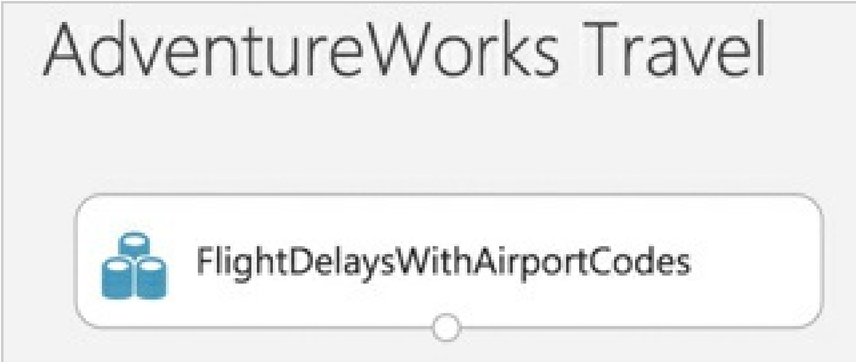

3. Give your new experiment a name, such as AdventureWorks Travel by editing the label near the top of the design surface.




4. In the toolbar on the left, in the Search experiment items box, type the name of the dataset you created with flight delay data (FlightDelaysWithAirportCodes). You should see a component for it listed under Saved Datasets, My Datasets.



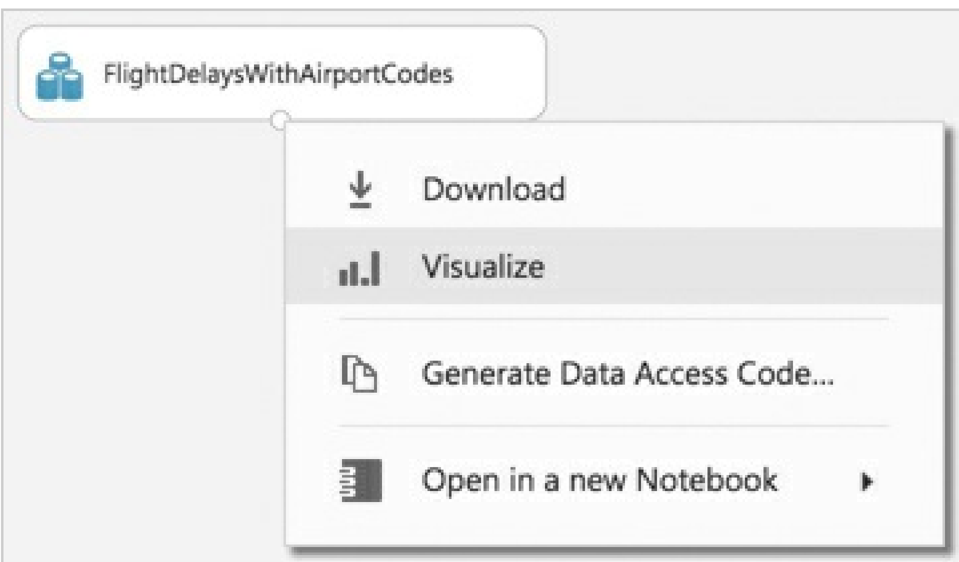

5. Click and drag on the FlightDelaysWithAirportCodes to add it to the design surface.



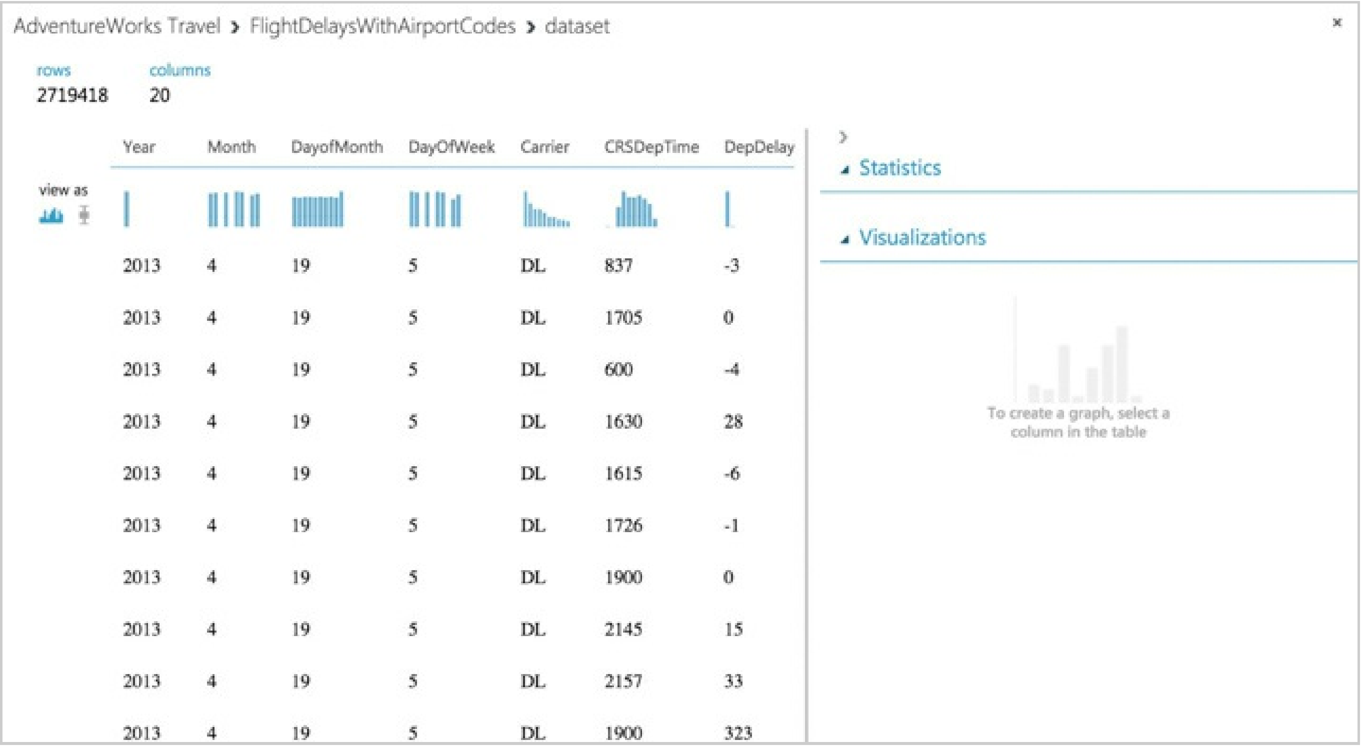

6. Next, you will explore each of the datasets to understand what kind of cleanup (aka data munging) will be necessary.

7. Hover over the output port of the FlightDelaysWithAirportCodes dataset.

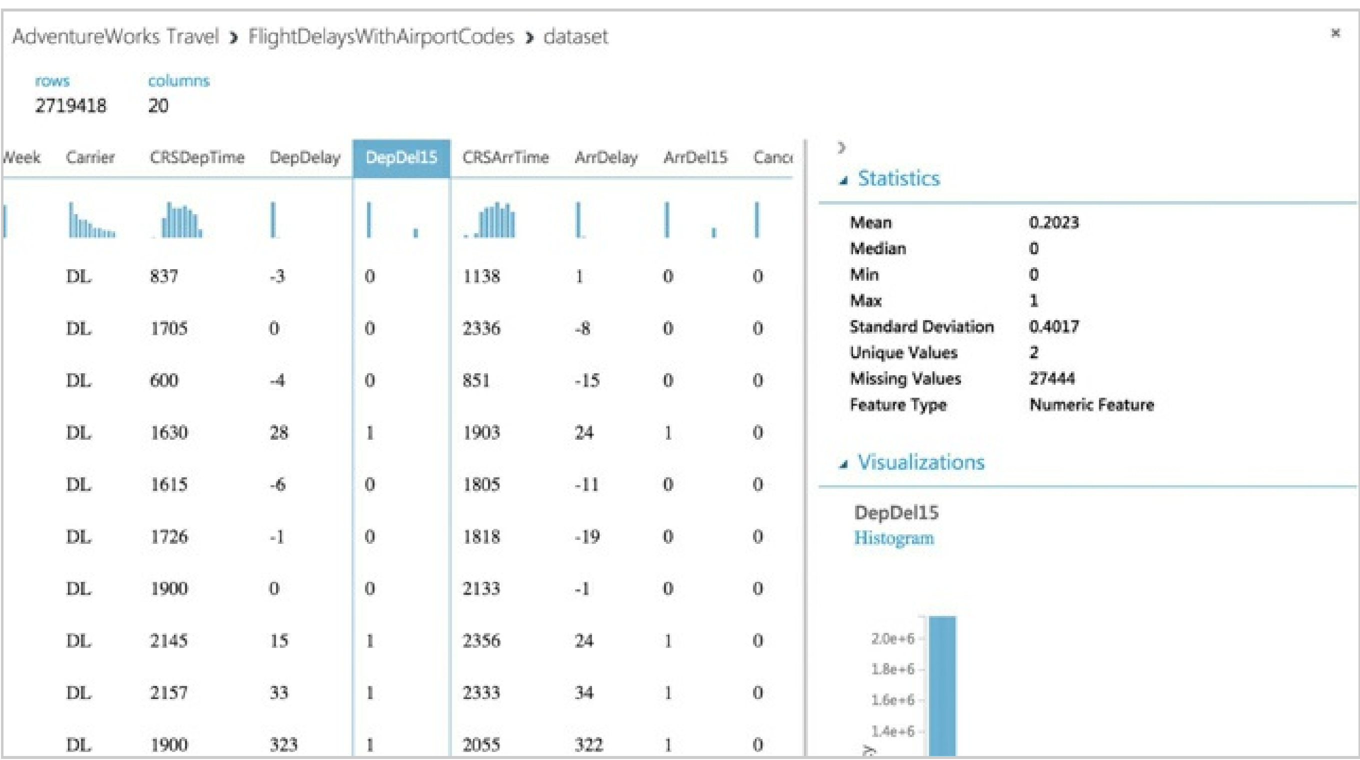

8. Right-click on the port and select **Visualize**.




9. A new dialog will appear showing a maximum of 100 rows by 100 columns sample of the dataset. You can see at the top that the dataset has a total of 2,719,418 rows (also referred to as examples in Machine Learning literature) and has 20 columns (also referred to as features).

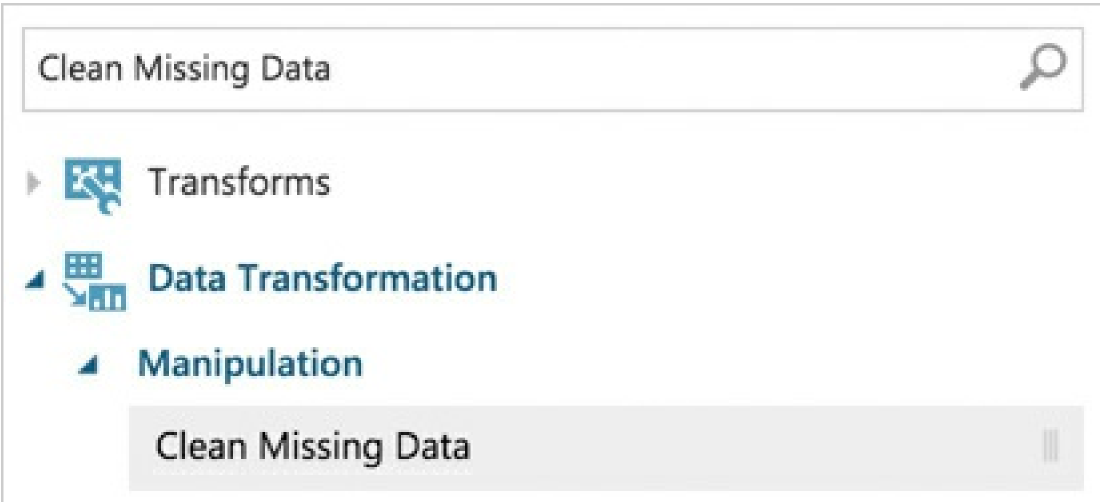


10. Because all 20 columns are displayed, you can scroll the grid horizontally. Scroll until you see the DepDel15 column and click it to view statistics about the column. The DepDel15 column displays a 1 when the flight was delayed at least 15 minutes and 0 if there was no such delay. In the model you will construct, you will try to predict the value of this column for future data.

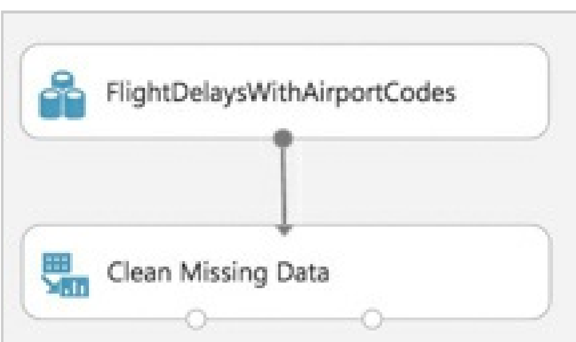


11. Notice in the Statistics panel that a value of 27444 appears for Missing Values. This means that 27,444 rows do not have a value in this column. Since this value is very important to our model, we will eliminate any rows that do not have a value for this column.

12. To eliminate these problem rows, close the dialog and go back to the design surface. From the toolbar, search for **Clean Missing Data**.



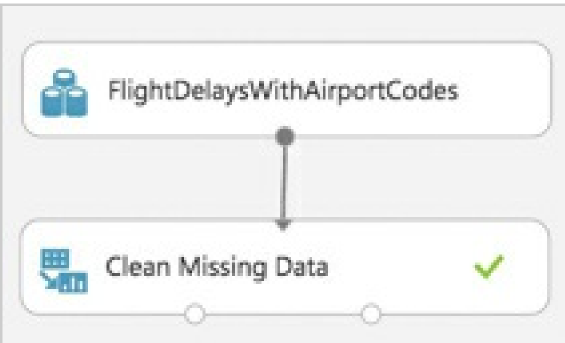

13. Drag this module on to the design surface beneath your FlightDelaysWithAirportCodes dataset. Click the small circle at the bottom of the FlightDelaysWithAirportCodes dataset, drag and release when your mouse is over the circle found in the top center of the Clean Missing Data module. These circles are referred to as ports, and by taking this action you have connected the output port of the dataset with the input port of the Clean Missing Data module, which means the data from the dataset will flow along this path.




14. Click **Save** on the command bar at the bottom to save your in-progress experiment.

15. Click **Run** in the command bar at the bottom to run the experiment.

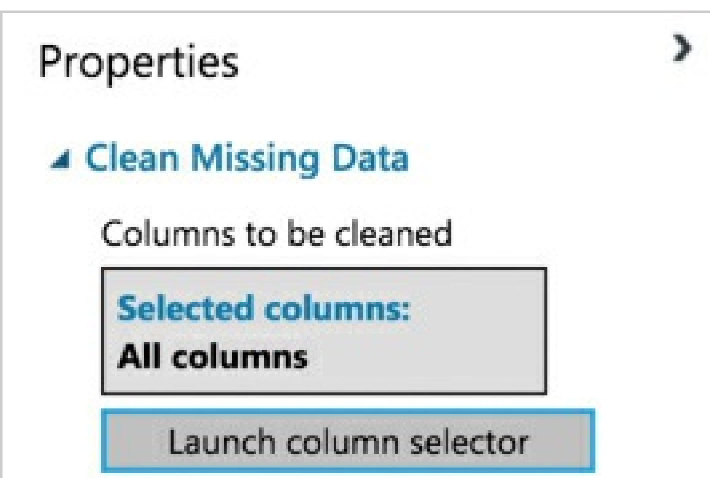

16. When the experiment is finished running, you will see a finished message in the top right corner of the design surface, and green check marks over all modules that ran.



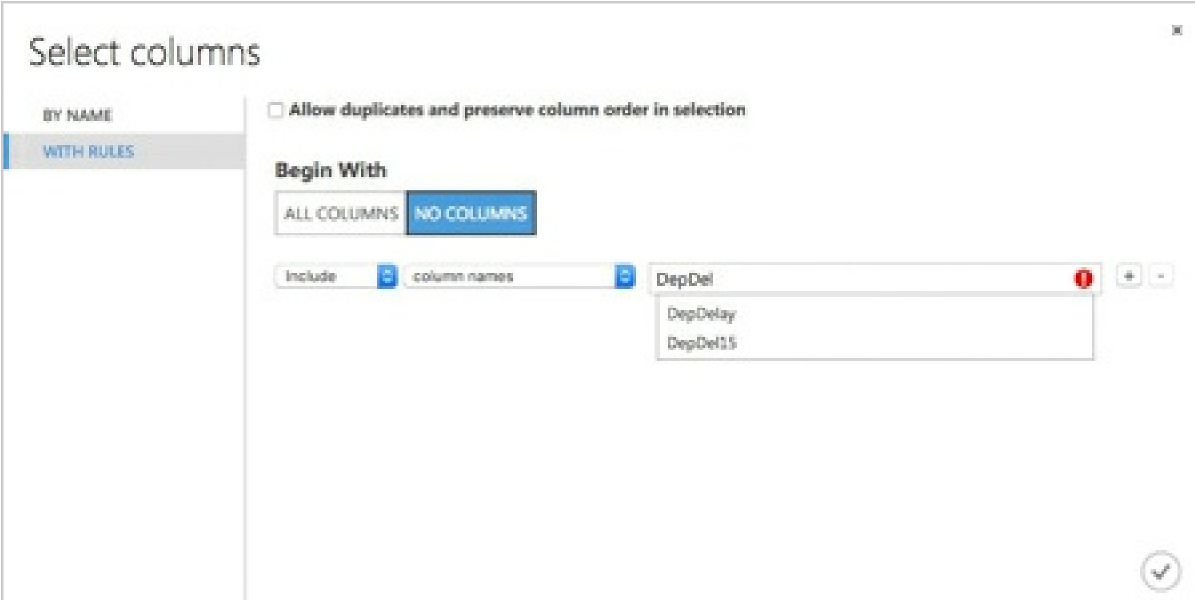

17. You should run your experiment whenever you need to update the metadata describing what data is flowing through the modules, so that newly added modules can be aware of the shape of your data (most modules have dialogs that can suggest columns, but before they can make suggestions you need to have run your experiment).

18. Click the **Clean Missing Data** module to select it. The property panel on the right will display the settings appropriate to the selected module.

19. In this case, we want to remove rows that have no value for the DepDel15 column. Begin by clicking **Launch Column Selector**.

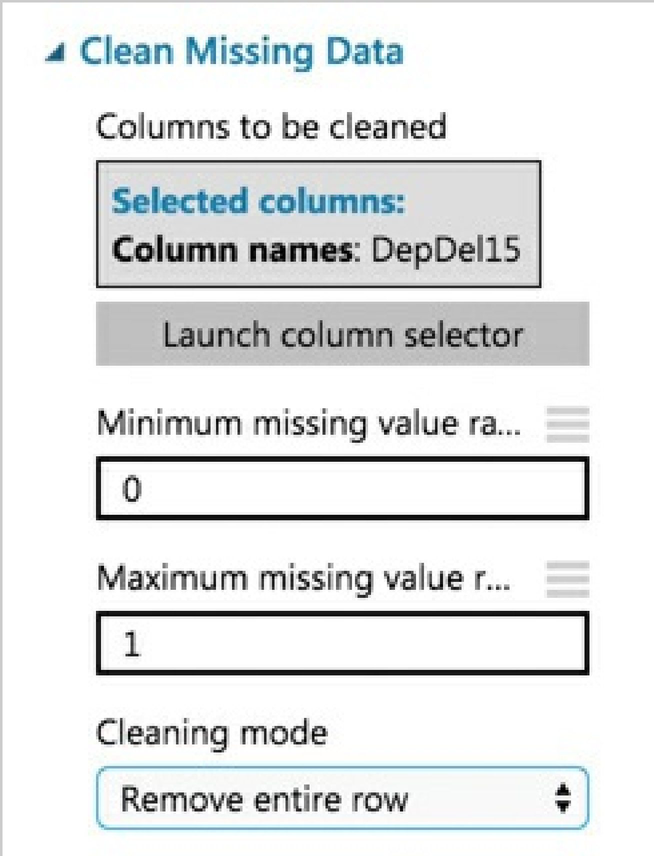



20. For the **Begin With** drop down, select **No Columns**. In the row of controls that appears, change the second drop down to **Include** and then **Column Names**. Then in the text box that appears begin to type DepDel15 and select that item from type-ahead list.



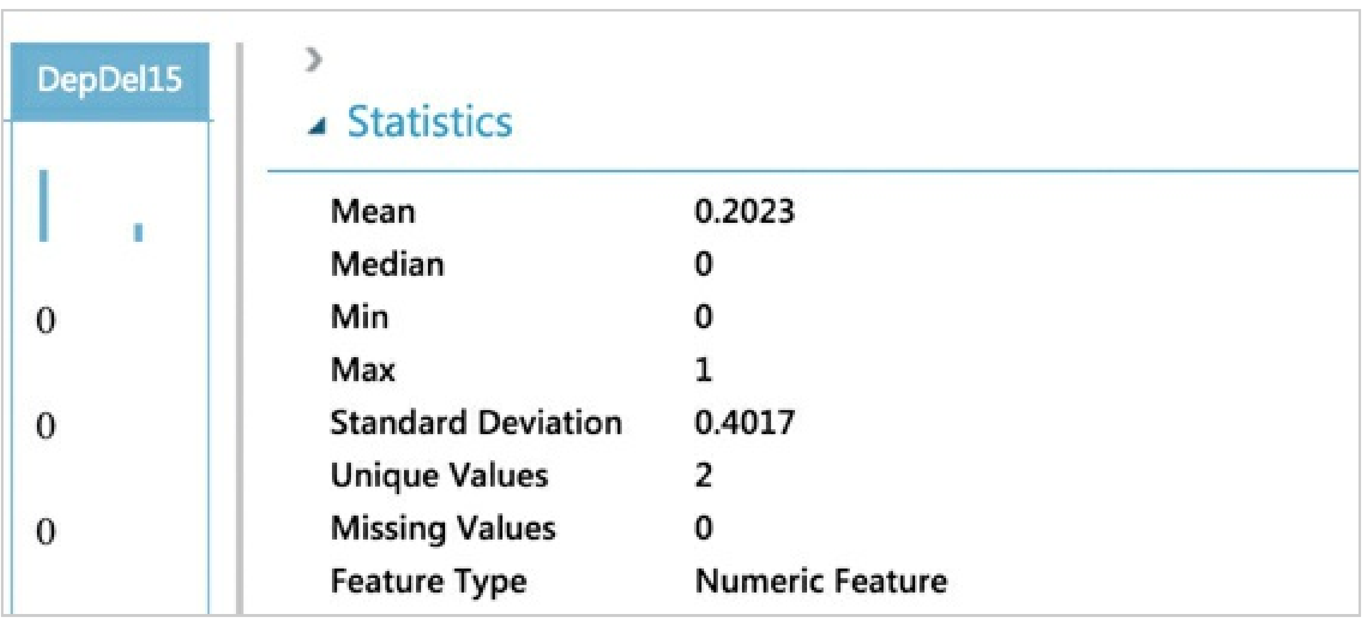

21. Click the checkmark to apply the settings. You have now indicated to the Clean Missing Data module that the DepDel15 is the only column it should act on.


22. In the **Properties** panel for **Clean Missing Data**, click the **Cleaning mode** drop-down and select **Remove entire row**. Now your Clean Missing Data module is fully configured to remove any rows that are missing values for DepDel15.




23. To verify the result, run your experiment again. After it is finished, click the left output port (Cleaned dataset) of the Clean Missing Data module and select **Visualize**.

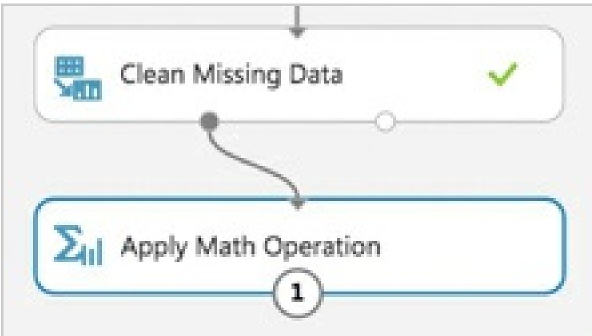
24. In the dialog that appears, scroll over to DepDel15 and click the column. In the statistics you should see that Missing Values reads 0.



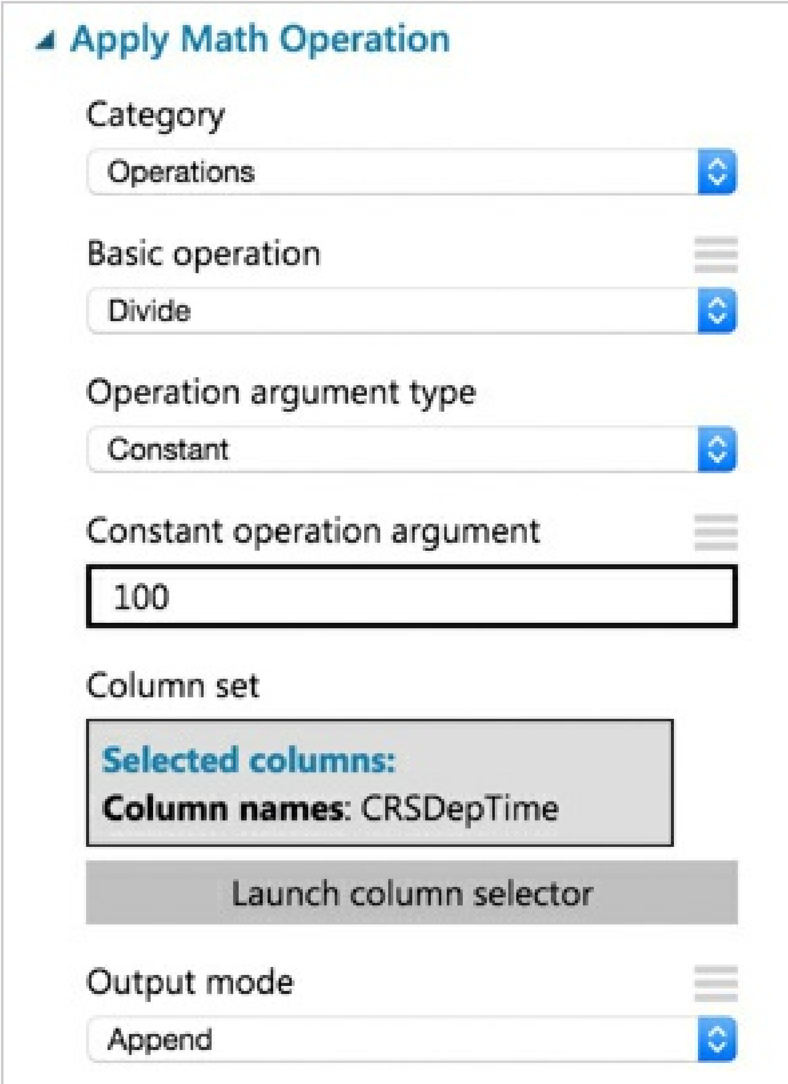

25. Our model will approximate departure times to the nearest hour, but departure time is captured as an integer. For example, 8:37 am is captured as 837. Therefore, we will need to process the CRSDepTime column and round it down to the nearest hour. To perform this

rounding will require two steps, first you will need to divide the value by 100 (so that 837 becomes 8.37). Second, you will round this value down to the nearest hour (so that 8.37 becomes 8).

26. Begin by adding an Apply Math Operation module beneath the Clean Missing Data module and connect the output port (1) of the Clean Missing Data module to the input port of the Apply Math Operation.

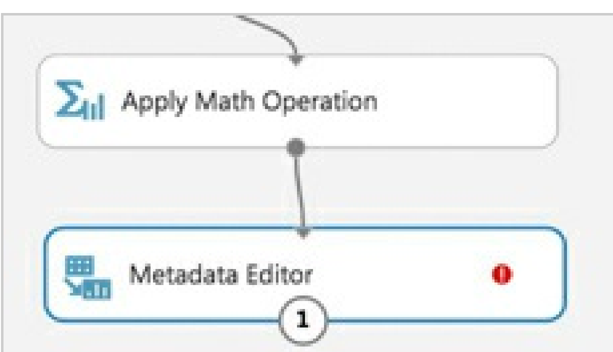



27. In the properties of the Apply Math Operation, set the Category to **Operations**, Basic operation to **Divide**, Operation argument type to **Constant**, Constant operation argument to **100**, Selected columns to **CRSDepTime**, and Output mode to **Append**.

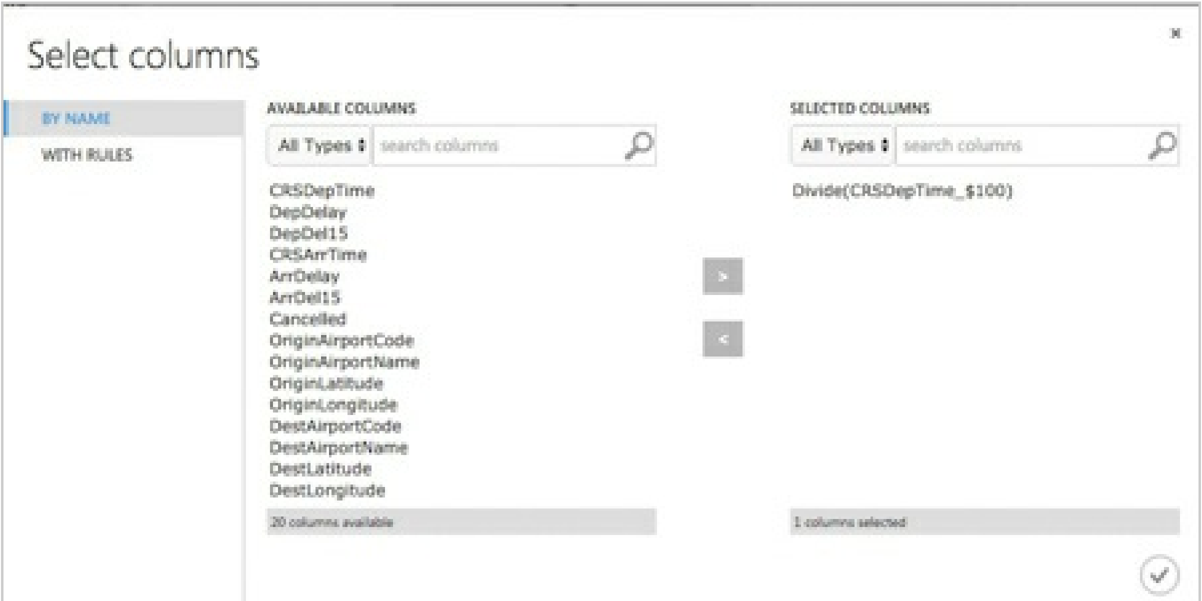



28. Run the experiment to update the metadata.

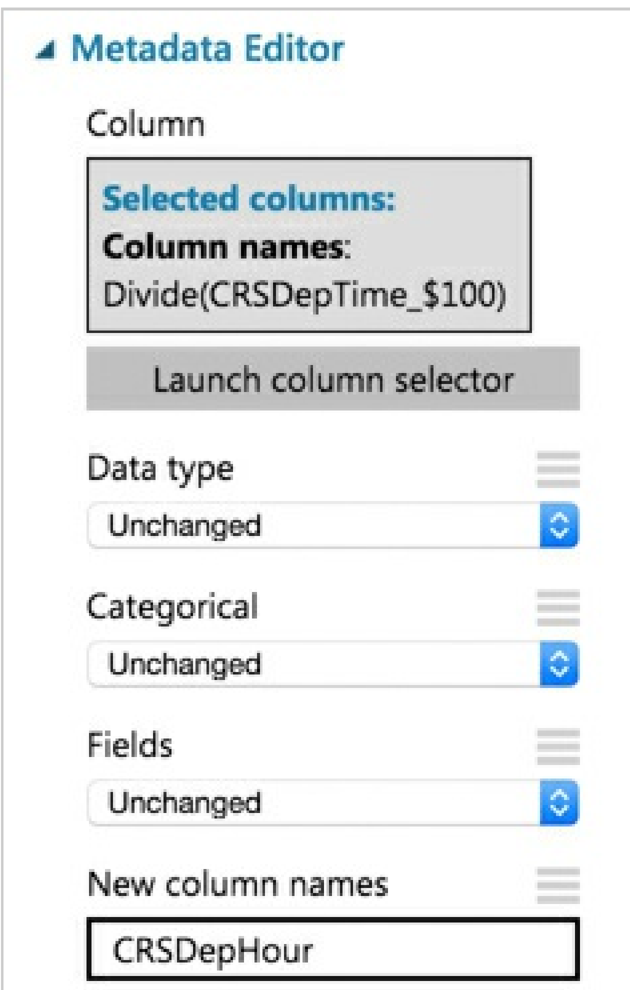
29. This module will add a new column to the dataset called Divide(CRSDeptTime_$100), but we want to rename it to CRSDepHour. To do so, add an **Edit Metadata** module and connect its input port to the output port of Apply Math Operation.




30. For the properties of the Metadata Editor, set the Selected Columns to **Divide(CRSDeptTime_$100)**

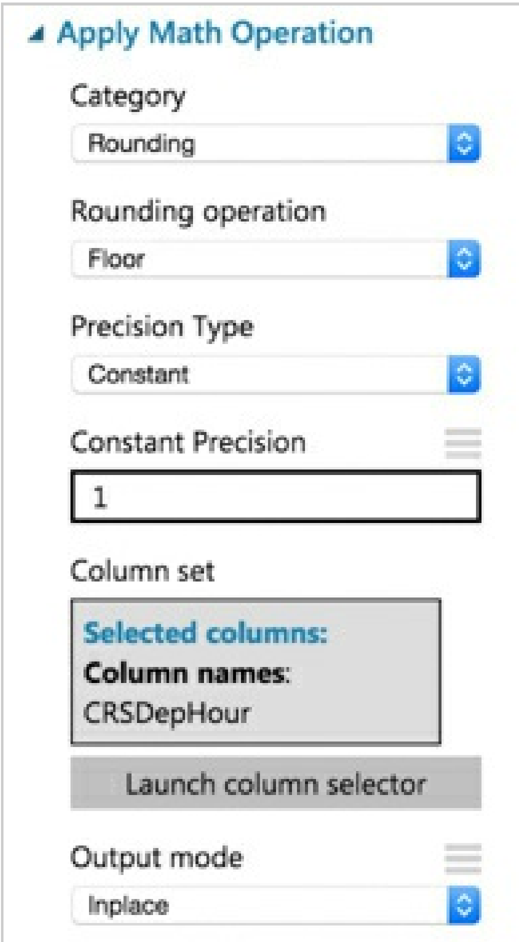



31. Back on the Properties for the Metadata Editor, set New column names to **CRSDepHour**.




32. Run the experiment to update the metadata.

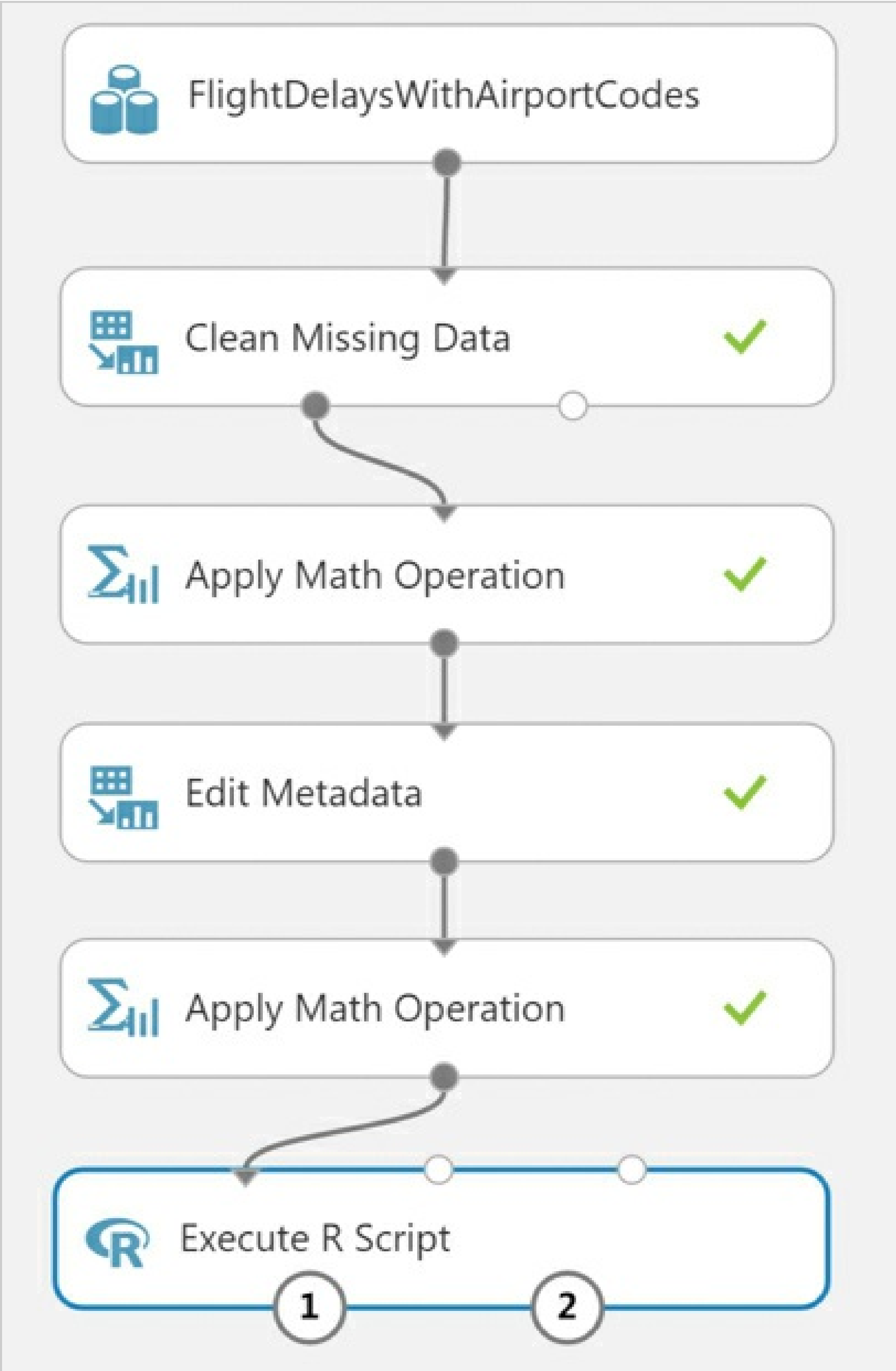
33. Add another **Apply Math Operation** module to round the time down to the nearest hour. Set the Category to **Rounding**, Selected columns to **CRSDepHour**, and Output mode to **Inplace**.




34. Run the experiment to update the metadata.

35. We do not need all of the columns present in the FlightDelaysWithAirportCodes dataset. To pare down the columns we can use multiple options, but in this case we chose to use an R Script module that selects out only the columns of interest using R code.

36. Add an **Execute R Script** module beneath the last Apply Math Operation, and connect the output of the Apply Math Operation to the first input port (leftmost) of the Execute R Script.




37. In the **Properties** panel for Execute R Script, click the **Double Windows** icon  to maximize the script editor.

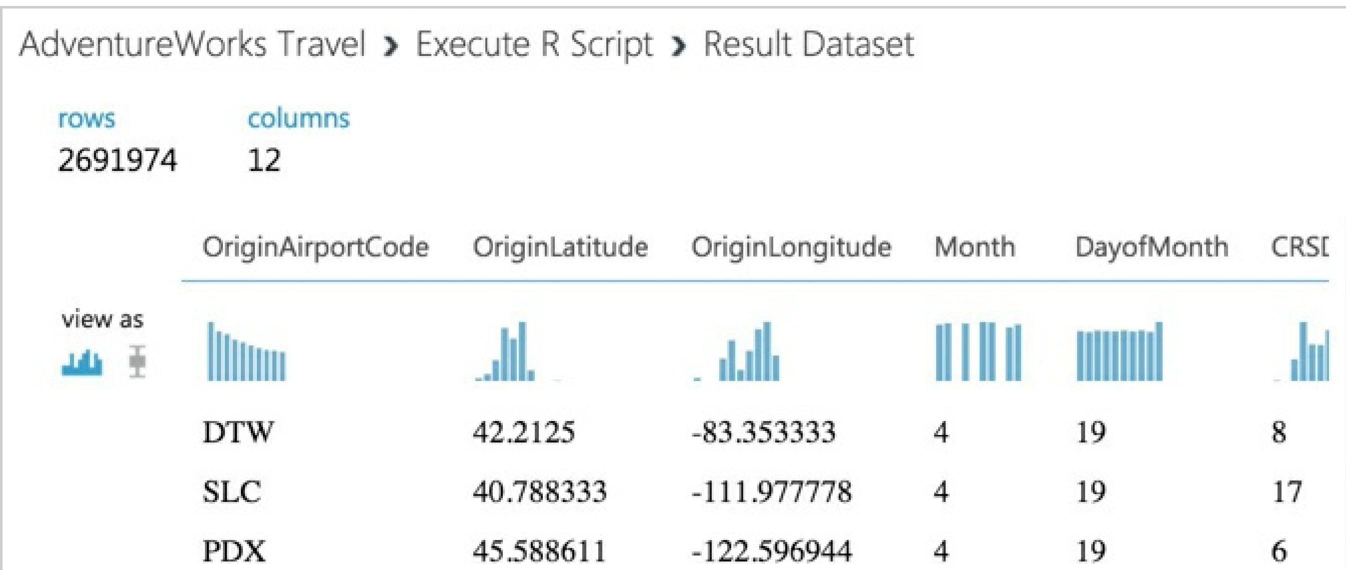
38. Replace the script with the following and click the check mark to save it. (Press CTRL+A to select all then CTRL+V to paste and then immediately click the check mark—do not worry if the formatting is off before hitting the check mark.)

```
ds.flights <- maml.mapInputPort(1)
# Trim the columns to only those we will use for the predictive model
ds.flights = ds.flights[, c("OriginAirportCode","OriginLatitude", "OriginLongitude", "Month", "DayofMonth", "CRSDepHour", "DayOfWeek", "Carrier", "DestAirportCode", "DestLatitude", "DestLongitude", "DepDel15")]
maml.mapOutputPort("ds.flights");
```

39. Run the experiment to update the metadata.

40. Right-click on the first output port of your Execute R Script module and select **Visualize**.

41. Verify that the dataset only contains the 12 columns referenced in the R script.



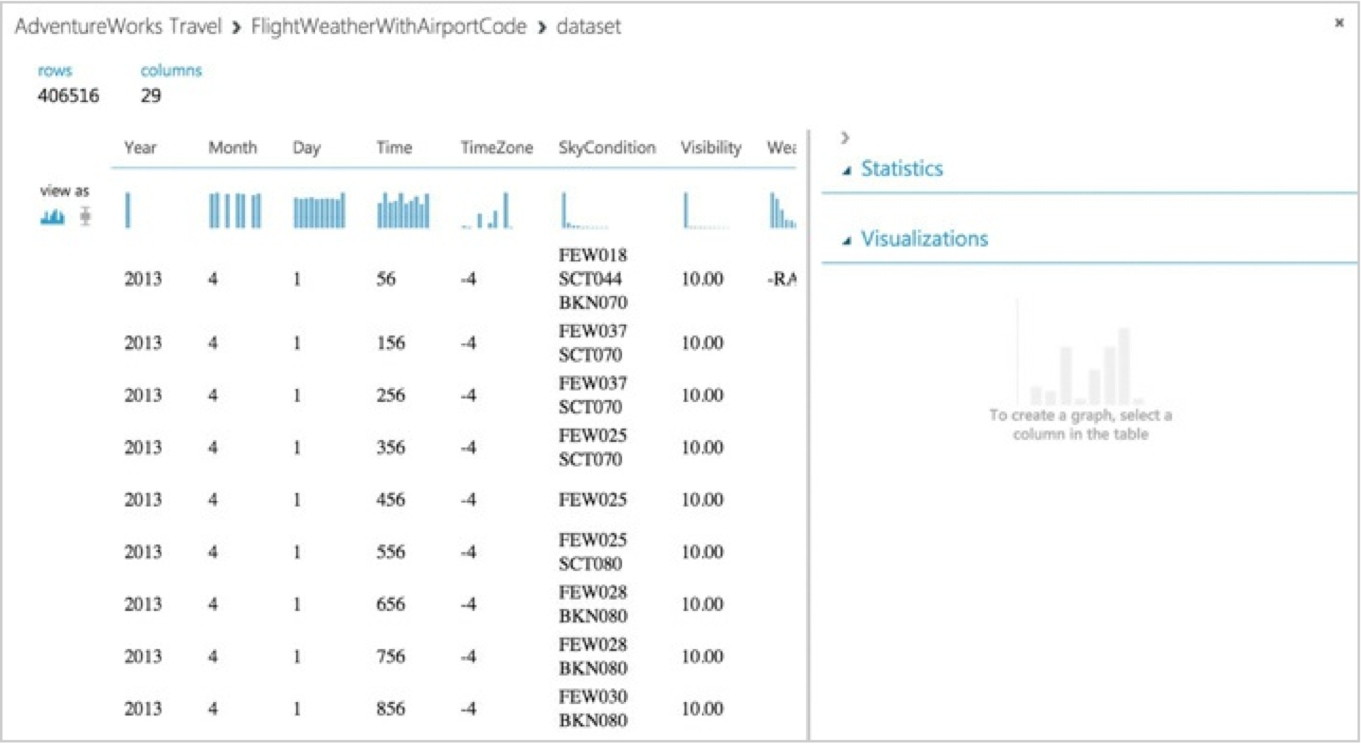

42. At this point the Flight Delay Data is prepared, and we turn to preparing the historical weather data.

## Task 4: Prepare the weather data

1. To the right of the FlightDelaysWithAirportCodes dataset, add the **FlightWeatherWithAirportCodes** dataset.

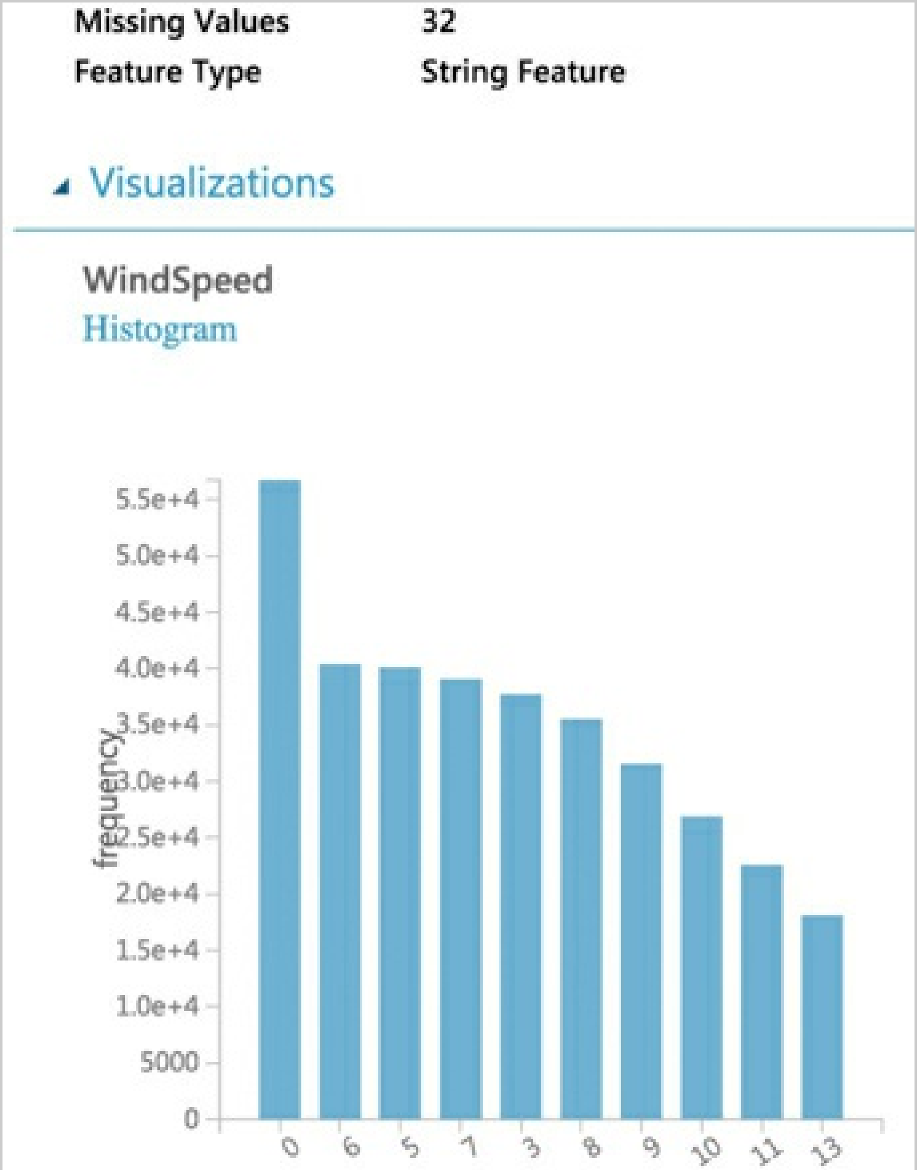

2. Right-click the output port of the FlightWeatherWithAirportCodes dataset and select

**Visualize**.

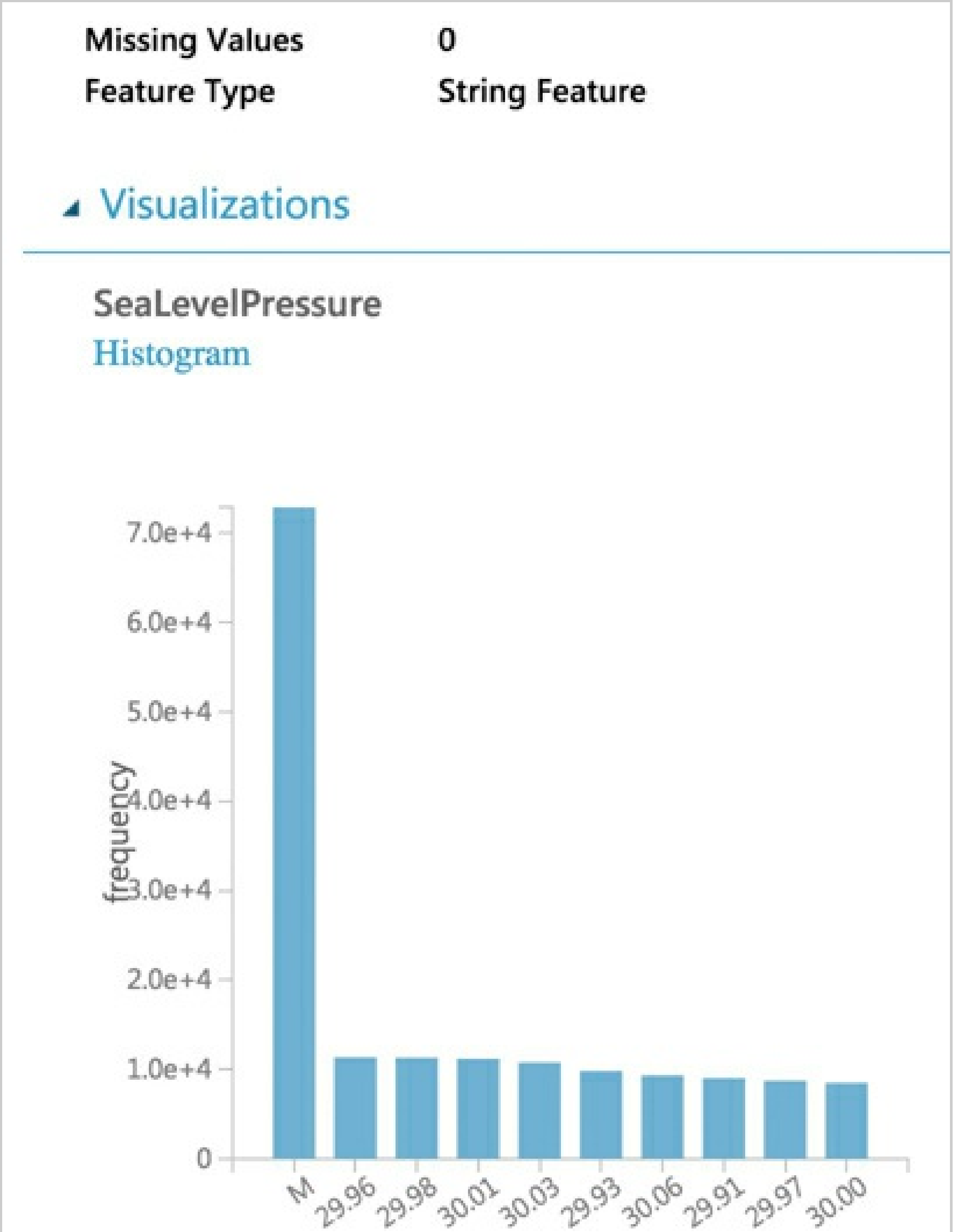



3. Observe that this data set has 406,516 rows and 29 columns. For this model, we are going to focus on predicting delays using WindSpeed (in MPH), SeaLevelPressure (in inches of Hg), and HourlyPrecip (in inches). We will focus on preparing the data for those features.

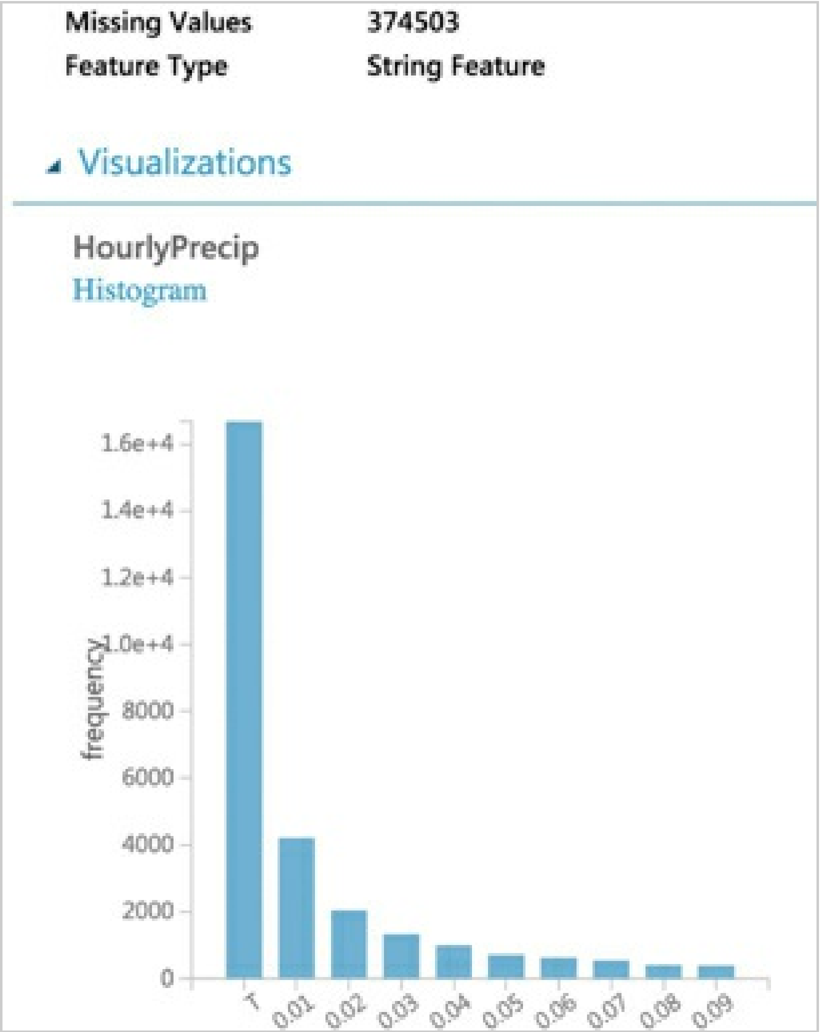
4. In the dialog, click the **WindSpeed** column and review the statistics. Observe that the Feature Type was inferred as String and that there are 32 Missing Values. Below that, examine the histogram to see that, even though the type was inferred as string, the values are all actually numbers (e.g. the x-axis values are 0, 6, 5, 7, 3, 8, 9, 10, 11, 13). We will need to ensure that we remove any missing values and convert WindSpeed to its proper type as a numeric feature.




5. Next, click the **SeaLevelPressure** column. Observe that the Feature Type was inferred as String and there are 0 Missing Values. Scroll down to the histogram, and observe that many of the features are of a numeric value (e.g., 29.96, 30.01, etc.), but there are many features with the string value of M for Missing. We will need to replace this value of “M” with a suitable numeric value so that we can convert this feature to be a numeric feature.

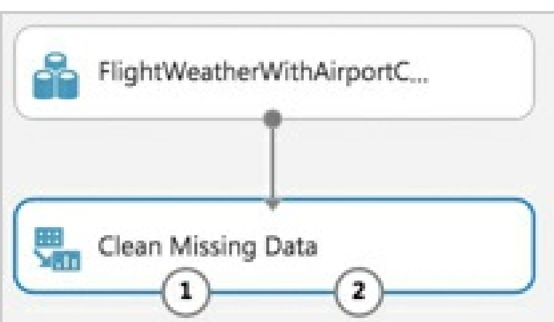



6. Finally, examine the HourlyPrecip feature. Observe that it too was inferred to have a Feature Type of String and is missing values for 374,503 rows. Looking at the histogram, observe that besides the numeric values, there is a value T for Trace amount of rain). We will need to replace the T with a suitable numeric value and covert this feature to a numeric feature.



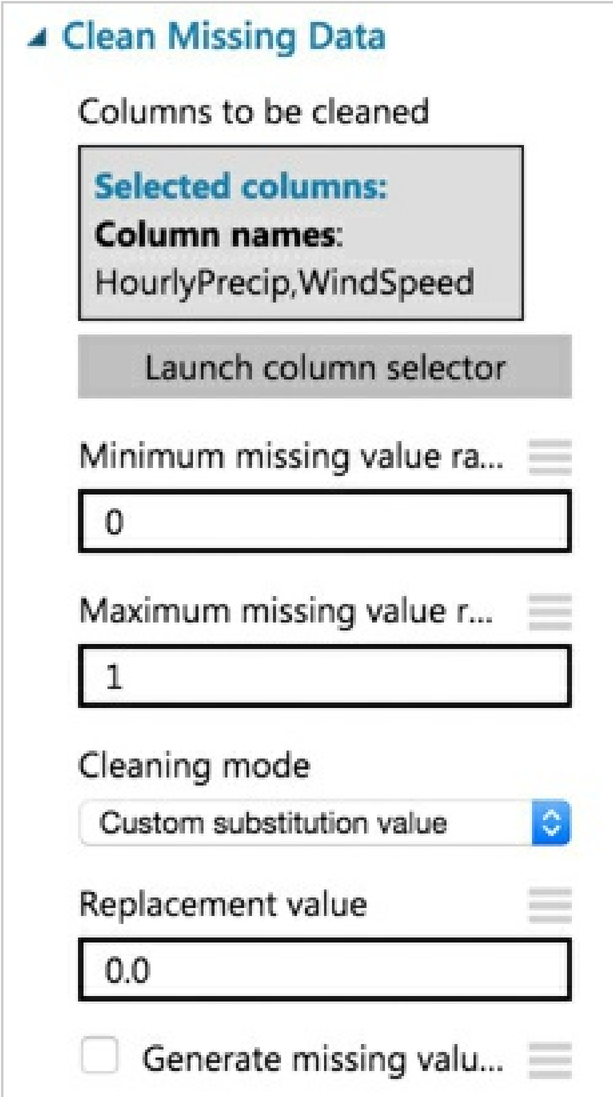

7. Let us begin by cleaning up the missing values for both WindSpeed and HourlyPrecip.

8. Below the FlightWeatherWithAirportCode dataset, drop a Clean Missing Data module and connect the output of the dataset to the input of module.

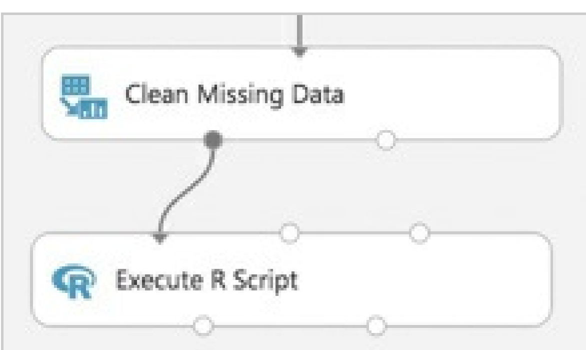



9. Run the experiment to update the metadata available to the Clean Missing Data module.

10. In the **Properties** panel for Clean Missing Data, set the Selected columns to **HourlyPrecip** and **WindSpeed**, set the Cleaning mode to **Custom substitution value** and set the Replacement value to **0.0**.




11. Next, add an Execute R Script module below the Clean Missing Data module and connect the first output port of the former to the first input port of the latter.



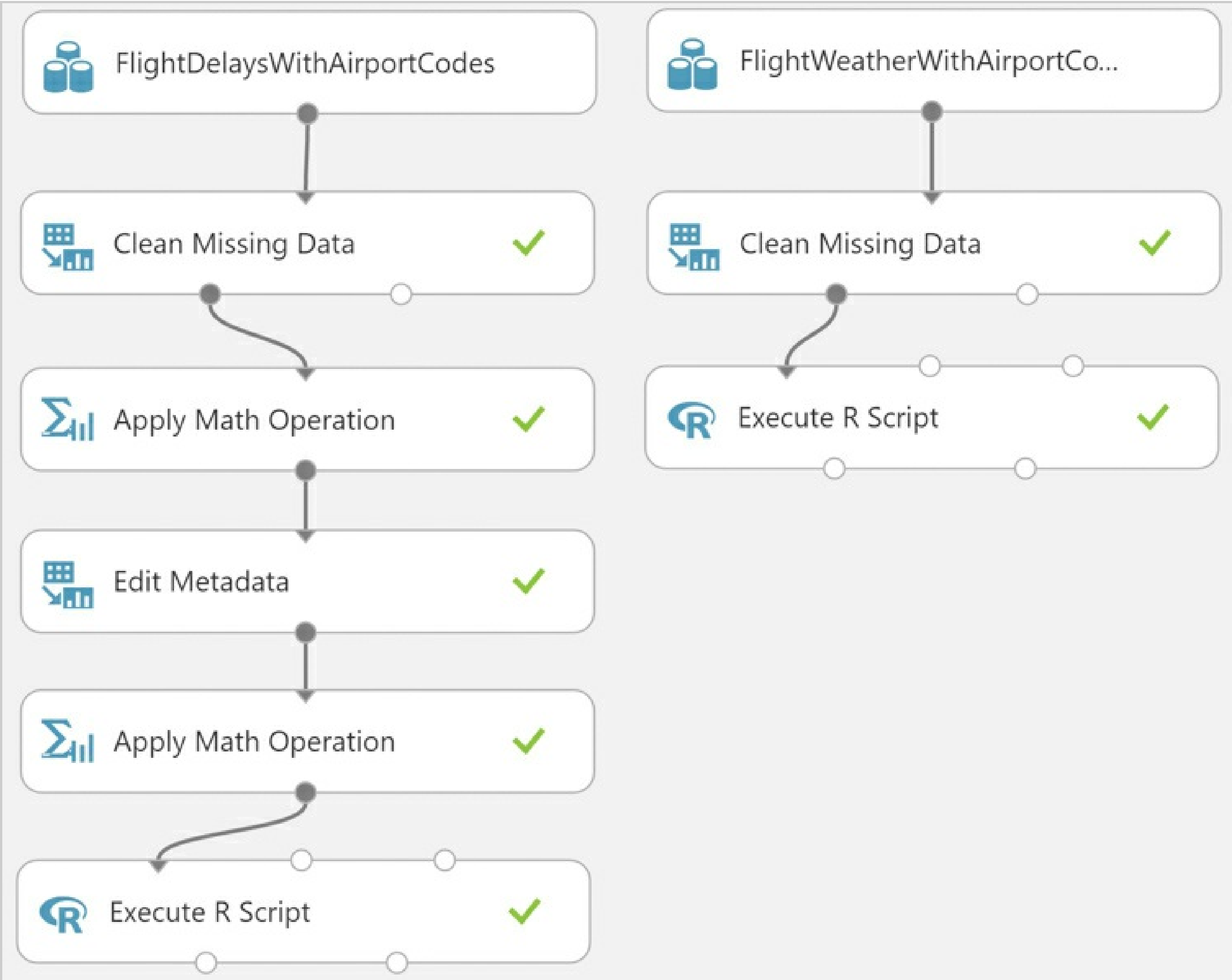

12. In the **Properties** panel for the Execute R Script, click the **Double Windows** icon to open the script editor.

13. Paste in the following script and click the checkmark (press CTRL+A to select all then CTRL+V to paste and then immediately click the checkmark-- don't worry if the formatting is off before hitting the checkmark). This script replaces the HourlyPrecip values having T with

0.005, WindSpeed values of M with 0.005, and the SeaLevelPressure values of M with the global average pressure of 29.92. It also narrows the dataset to just the few feature columns we want to use with our model.

```
ds.weather <- maml.mapInputPort(1)

# Round weather time up to the next hour since
# that's the hour for which we want to use flight data
ds.weather$Hour = ceiling(ds.weather$Time / 100)

# Replace any WindSpeed values of "M" with 0.005 and make the feature numeric
speed.num = ds.weather$WindSpeed
speed.num[speed.num == "M"] = 0.005
speed.num = as.numeric(speed.num)
ds.weather$WindSpeed = speed.num

# Replace any SeaLevelPressure values of "M" with 29.92 (the average pressure)
and make the feature numeric
pressure.num = ds.weather$SeaLevelPressure
pressure.num[pressure.num == "M"] = 29.92
pressure.num = as.numeric(pressure.num)
ds.weather$SeaLevelPressure = pressure.num

# Adjust the HourlyPrecip variable (convert "T" (trace) to 0.005)
rain = ds.weather$HourlyPrecip
rain[rain %in% c("T")] = "0.005"
ds.weather$HourlyPrecip = as.numeric(rain)

# Pare down the variables in the Weather dataset
ds.weather = ds.weather[, c("AirportCode", "Month", "Day", "Hour", "WindSpeed",
 "SeaLevelPressure", "HourlyPrecip")]

maml.mapOutputPort("ds.weather");
```

14. Run the experiment. Currently it should appear as follows:




15. Click the first output port of the Execute R Script module.

16. In the statistics, verify that WindSpeed, SeaLevelPressure, and HourlyPrecip are now all Numeric Feature types and that they have no missing values.

### Task 5: Join the Flight and Weather datasets

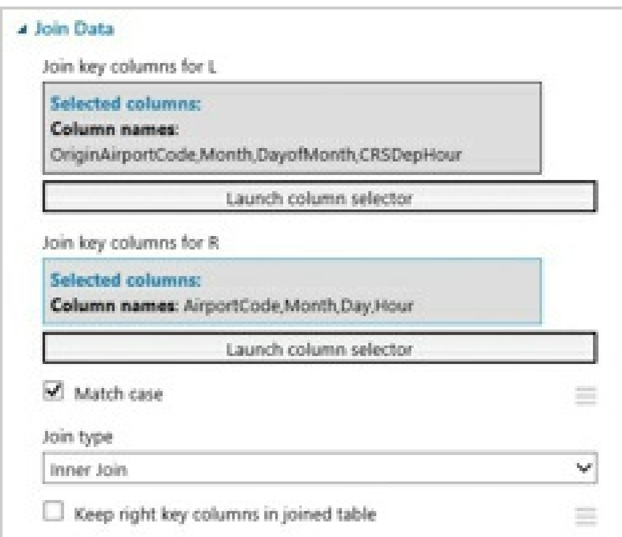
1. With both datasets ready, we want to join them together so that we can associate historical flight delay with the weather data at departure time.

2. Drag the Join Data module on to the design surface, beneath and centered between both Execute R Script modules. Connect the output port (1) of the left Execute R module to input port (1) of the Join Data module, and the output port (1) of the right Execute R module to the input port (2) of the Join Data module.

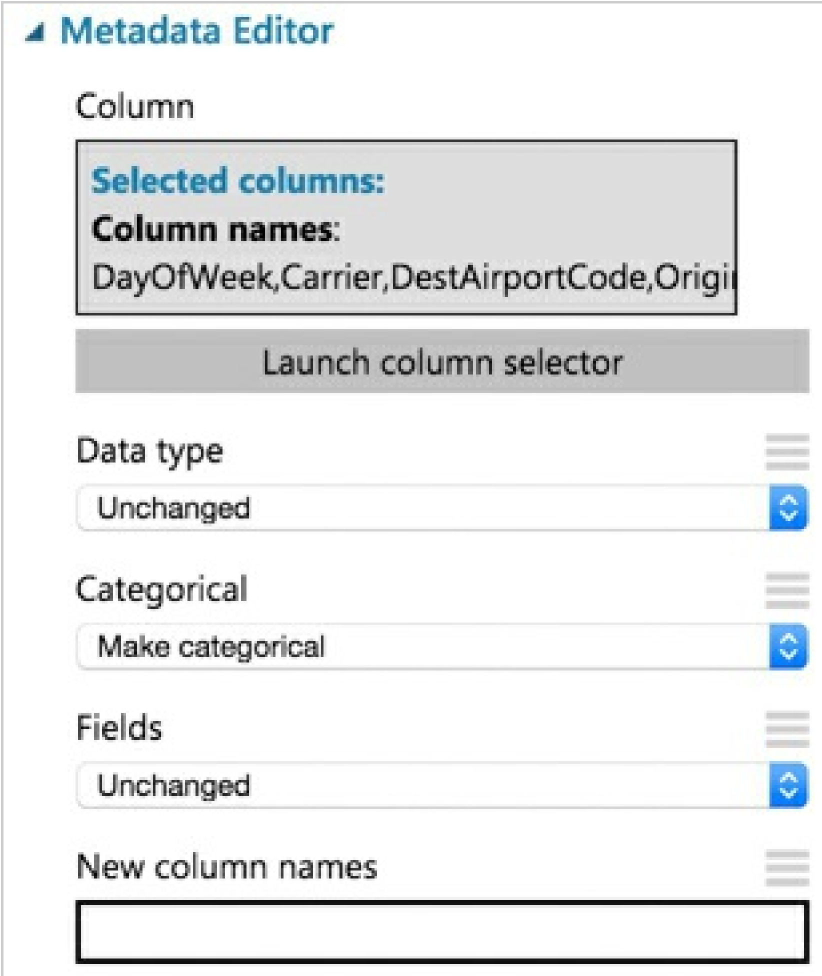

3. In the **Properties** panel, relate the rows of data between the two sets L (the flight delays) and R (the weather). Set the Join key columns for L to include OriginAirportCode, Month, DayofMonth, and CRSDepHour.

4. Set the join key columns for R to include AirportCode, Month, Day, and Hour.

5. Leave the Join Type at inner join and uncheck **Keep right key columns in joined table** (so that we do not include the redundant values of AirportCode, Month, Day, and Hour).




6. Next, add an **Edit Metadata** module and connect its input port to the output port of the Join Data module. We will use this module to convert the fields that were unbounded String feature types, to the enumeration like Categorical feature. On the **Properties** panel, set the Selected columns to DayOfWeek, Carrier, DestAirportCode, and OriginAirportCode. Set the Categorical drop down to **Make categorical**.




7. Add a **Select Columns in Dataset** module. Choose **Begin With All Columns**, choose **Exclude** and set the selected columns to exclude: **OriginLatitude**, **OriginLongitude**, **DestLatitude**, and **DestLongitude**.


8. Connect the output of the Metadata Editor to the input of the Select Columns in Dataset module.

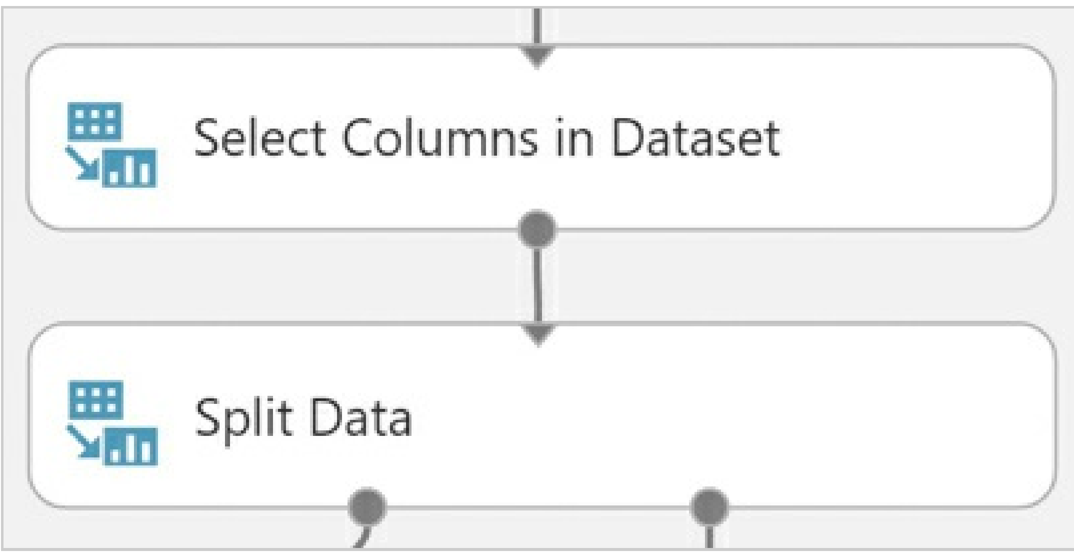
9. Save your experiment.

### Task 6: Train the model

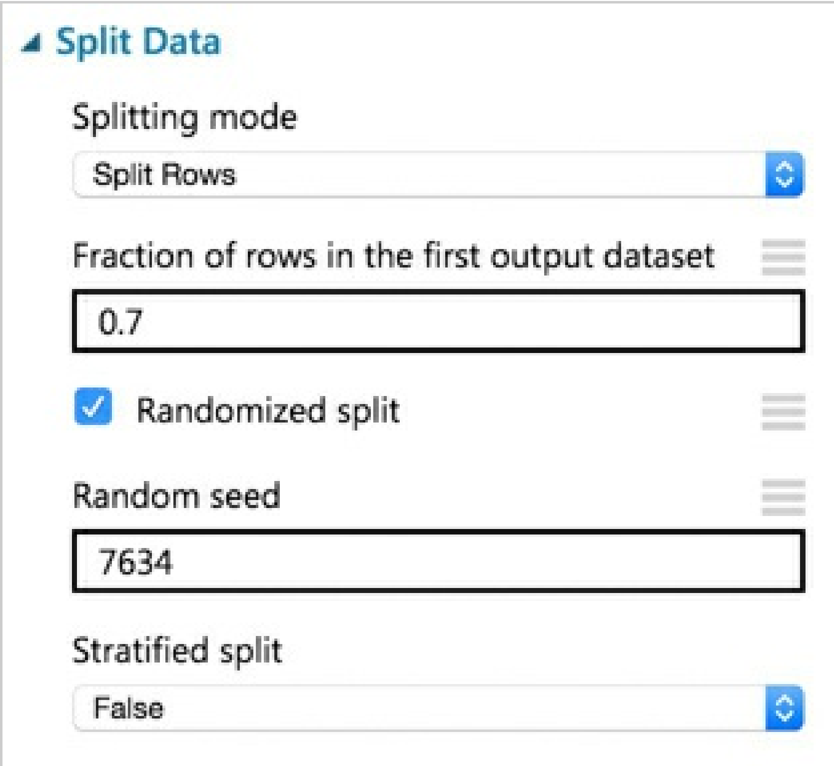
AdventureWorks Travel wants to build a model to predict if a departing flight will have a 15-minute or greater delay. In the historical data they have provided, the indicator for such a delay is found within the DepDel15 (where a value of 1 means delay, 0 means no delay). To create a model that predicts such a binary outcome, we can choose from the various Two-Class modules that Azure ML offers. For our purposes, we begin with a Two-Class Logistic Regression. This type of classification module needs to be first trained on sample data that includes the features important to making a prediction and must also include the actual historical outcome for those features.

The typical pattern is split the historical data so a portion is shown to the model for training purposes, and another portion is reserved to test just how well the trained model performs against examples it has not seen before.

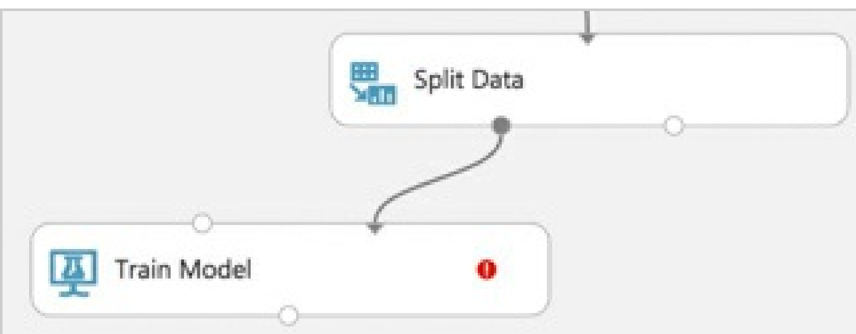
1. Drag a Split module beneath Select Columns in Dataset and connect them.



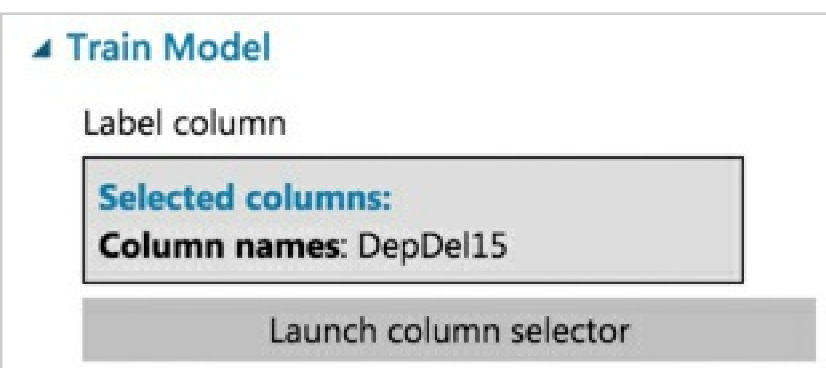

2. On the **Properties** panel for the Split Data module, set the Fraction of rows in the first output dataset to **0.7** (so 70% of the historical data will flow to output port 1). Set the Random seed to **7634**.



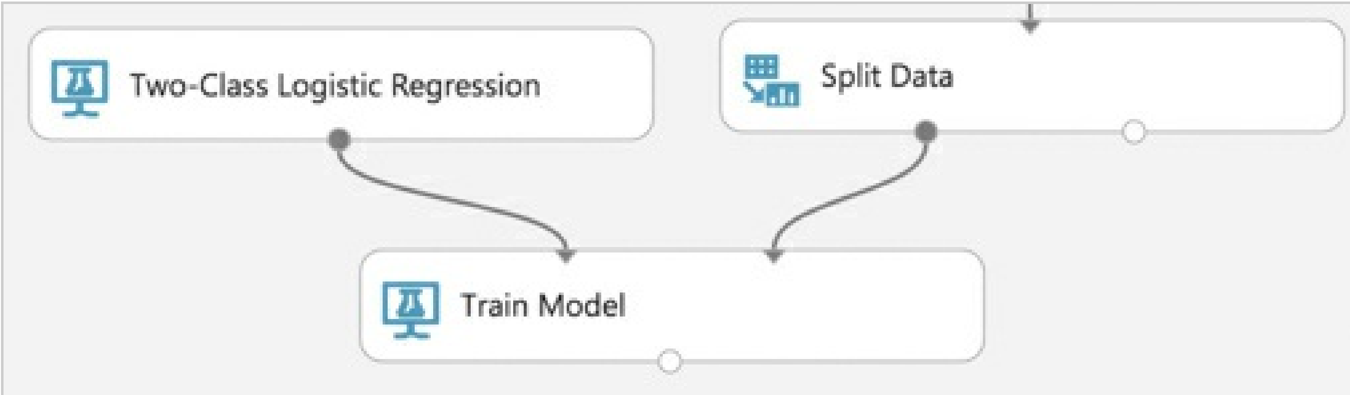

3. Next, add a Train Model module and connect it to output 1 of the Split Data module.



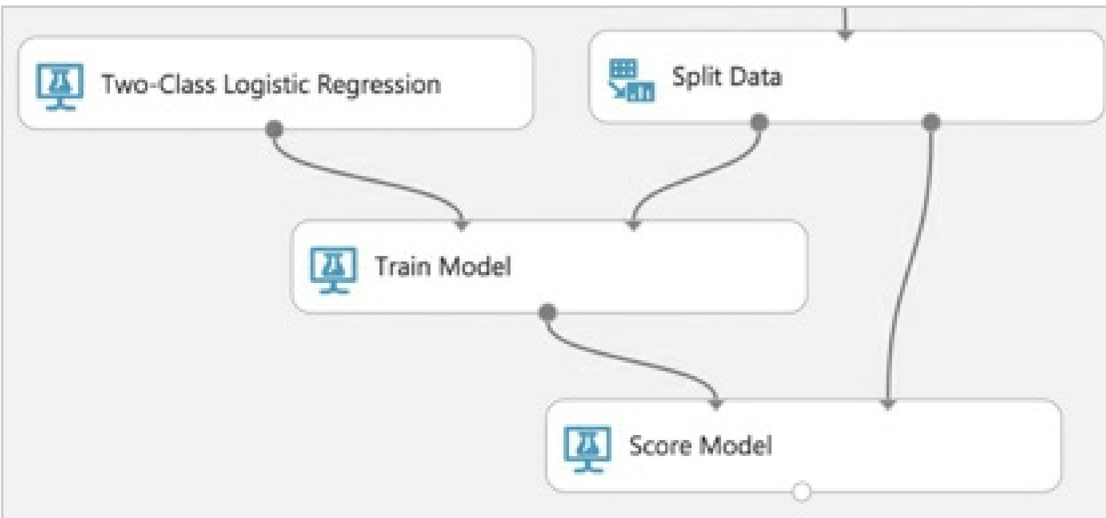

4. On the **Properties** panel for the Train Model module, set the Selected columns to **DepDel15**.




5. Drag a Two-Class Logistic Regression module above and to the left of the Train Model module and connect the output to the leftmost input of the Train Model module




6. Below the Train Model drop a Score Model module. Connect the output of the Train Model module to the leftmost input port of the Score Model and connect the rightmost output of the Split Data module to the rightmost input of the Score Model.




7. Run the experiment.

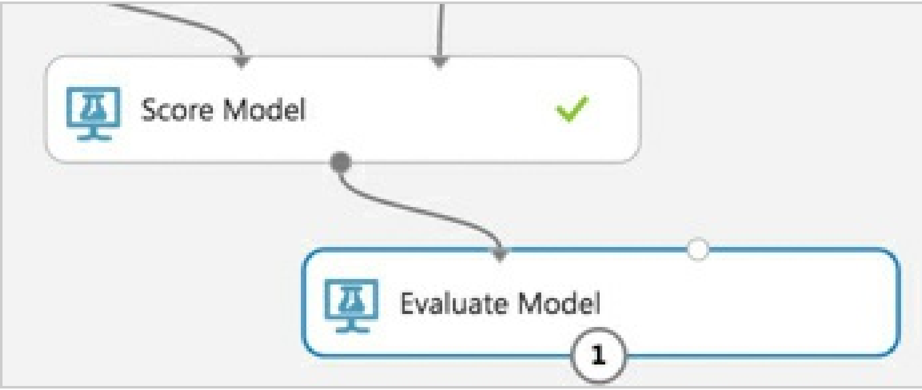
8. When the experiment is finished running (which takes a few minutes), right-click on the output port of the Score Model module and select **Visualize** to see the results of its predictions. **You should have a total of 13 columns**.


9. If you scroll to the right so that you can see the last two columns, observe there is a Scored Labels column and a Scored Probabilities column. The former is the prediction (1 for predicting delay, 0 for predicting no delay) and the latter is the probability of the prediction. In the following screenshot, for example, the last row shows a delay predication with a 53.1% probability.

| Scored Labels | Scored Probabilities |
|---|---|
| 0 | 0.224147 |
| 0 | 0.182701 |
| 0 | 0.073537 |
| 0 | 0.270637 |
| 0 | 0.267242 |
| 0 | 0.191029 |
| 0 | 0.095636 |
| 0 | 0.207545 |
| 0 | 0.22024 |
| 0 | 0.082833 |
| 1 | 0.531273 |

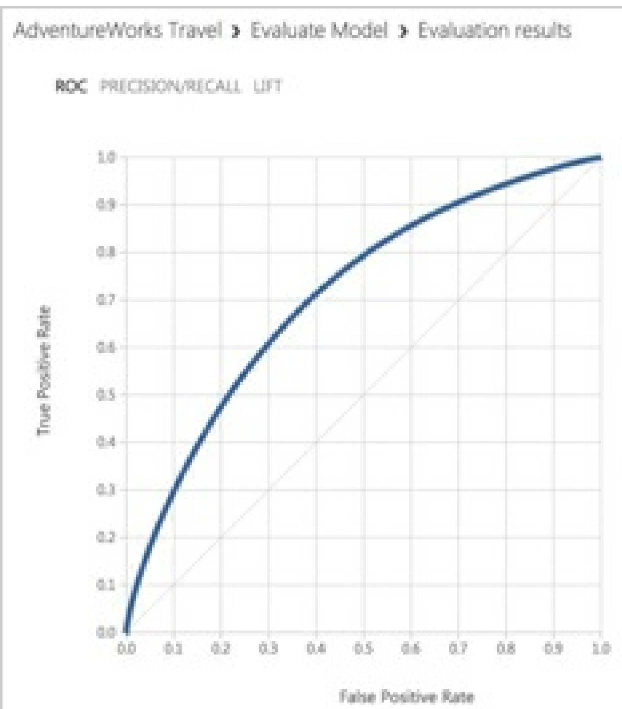
10. While this view enables you to see the prediction results for the first 100 rows, if you want to get more detailed statistics across the prediction results to evaluate your model's performance, you can use the Evaluate Model module.

11. Drag an Evaluate Model module on to the design surface beneath the Score Model module. Connect the output of the Score Model module to the leftmost input of the Evaluate Model module.




12. Run the experiment.

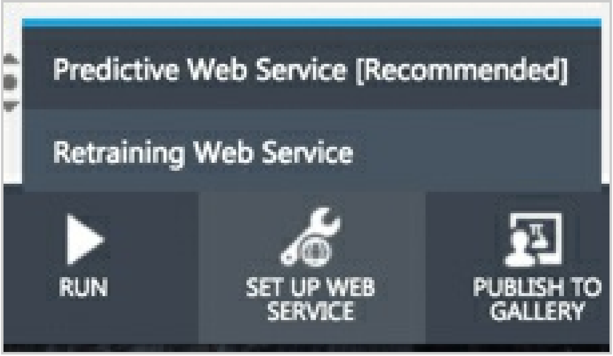
13. When the experiment is finished running, right-click the output of the Evaluate Model module and select **Visualize**. In this dialog box, you are presented with various ways to understand how your model is performing in the aggregate. While we will not cover how to interpret these results in detail, we can examine the ROC chart that tells us that at least our model (the blue curve) is performing better than random (the light gray straight line going from 0,0 to 1,1)—which is a good start for our first model!



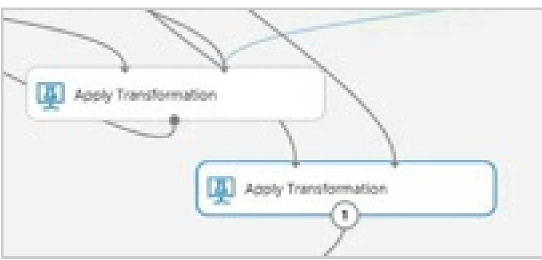

## Task 7: Operationalize the experiment

1. Now that we have a functioning model, let us package it up into a predictive experiment that can be called as a web service.

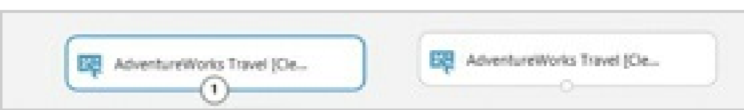
2. In the command bar at the bottom, click **Set Up Web Service** and then select **Predictive Web Service [Recommended]**. (If Predictive Web Service is grayed out, run the experiment again)




3. A copy of your training experiment is created that contains the trained model wrapped between web service input (e.g. the web service action you invoke with parameters) and web service output modules (e.g., how the result of scoring the parameters are returned).
4. We will make some adjustments to the web service input and output modules to control the parameters we require and the results we return.
5. When packaging the Predictive Web Service, Azure ML added two Apply Transformation modules which are not needed. **Delete both of the Apply Transformation Modules**.




6. The Apply Transformation Modules were added to support the Clean Missing Data modules. We will not be using these steps in our flow, so **delete both Clean Missing Data Modules**.




7. Reconnect the FlightDelaysWithAirportCodes to the input to Apply Math Operation module that is directly beneath it.

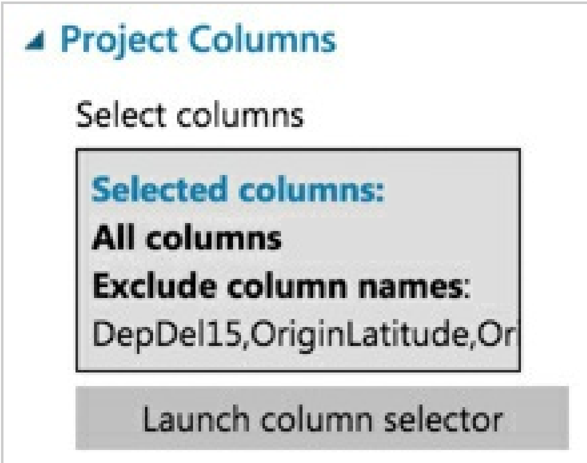

8. Reconnect the FlightWeatherWithAirportCodes module to the leftmost input port of the Execute R Script module beneath it.


9. Now move the web service input down so it is to the right of the Join Data module. Connect the output of the Web service input to input of the Metadata Editor.

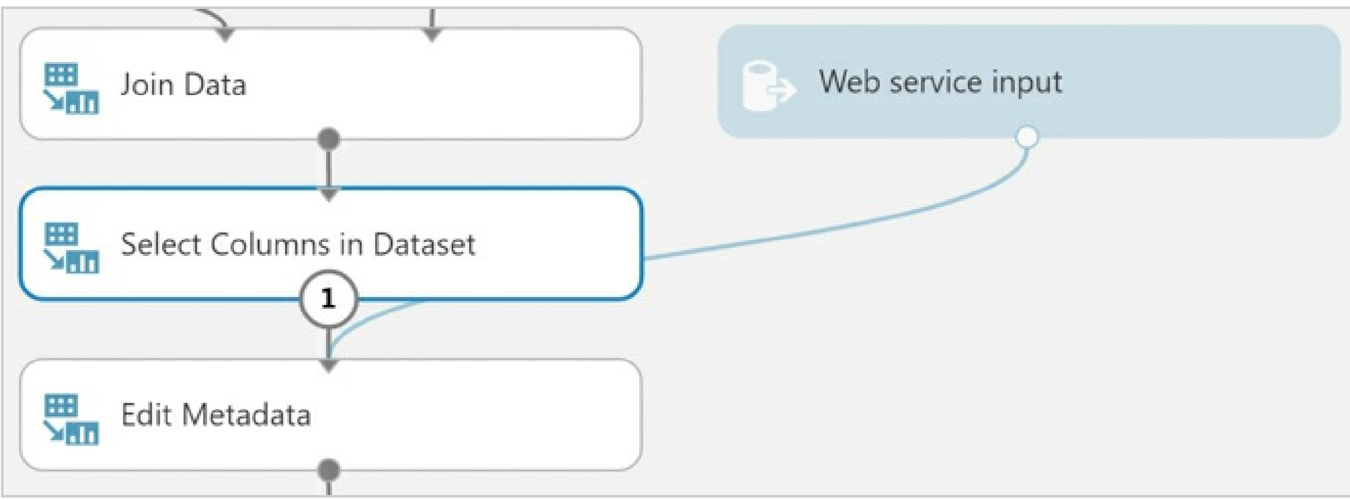

10. Right-click the line connecting the Join Data module and the Edit Metadata and select Delete.


11. In between the Join Data and the Metadata Editor modules, drop a Select Columns in Dataset module. In the **Properties** panel for the Select Columns in Dataset module, set the Select columns to **all columns**, exclude columns **DepDel15**, **OriginLatitude**, **OriginLongitude**, **DestLatitude**, and **DestLongitude**. This configuration will update the web service metadata so that these columns do not appear as required input parameters for the web

service.

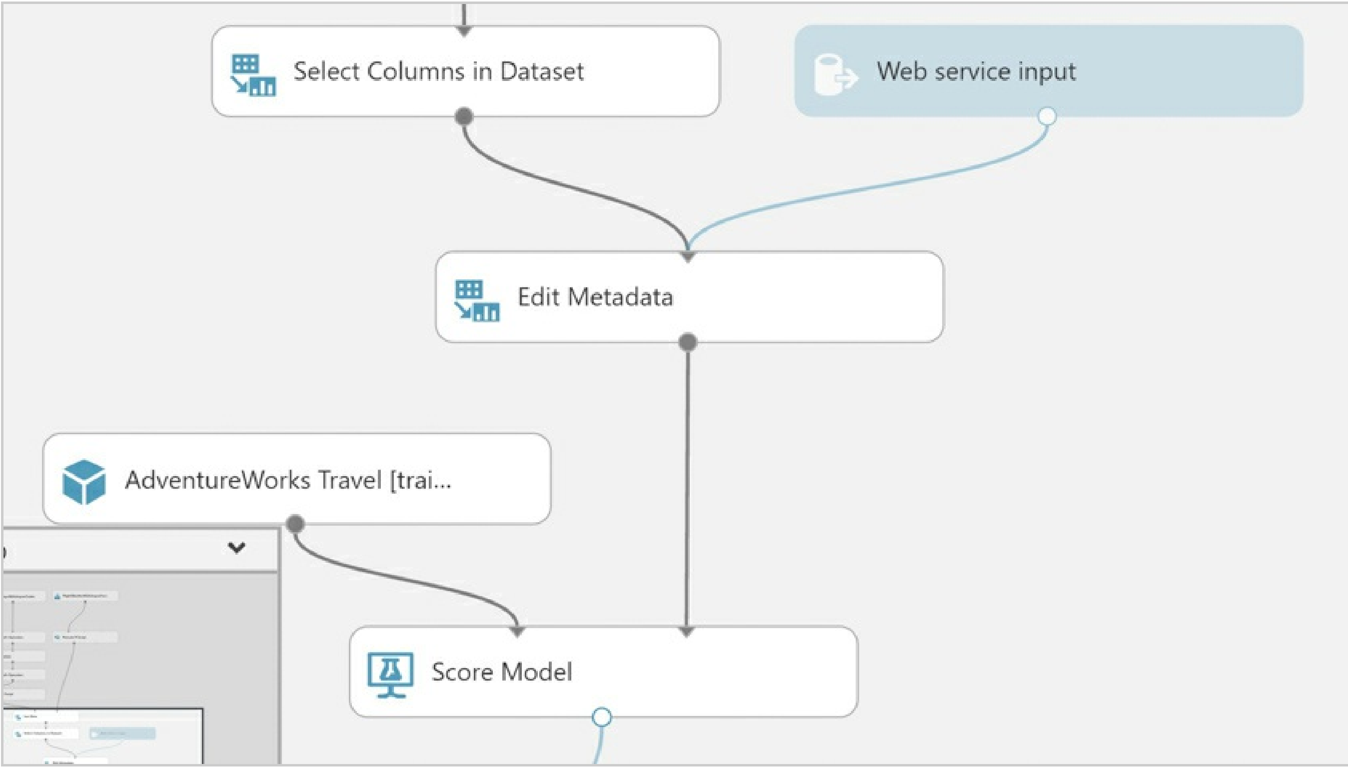



12. Connect the Select Columns in Dataset input to Join Data and its output to Metadata Editor.




13. Select the Select Columns in Dataset module that comes after the Metadata Editor and delete it.

14. Connect the output of the Edit Metadata directly to the right input of the Score Model module.

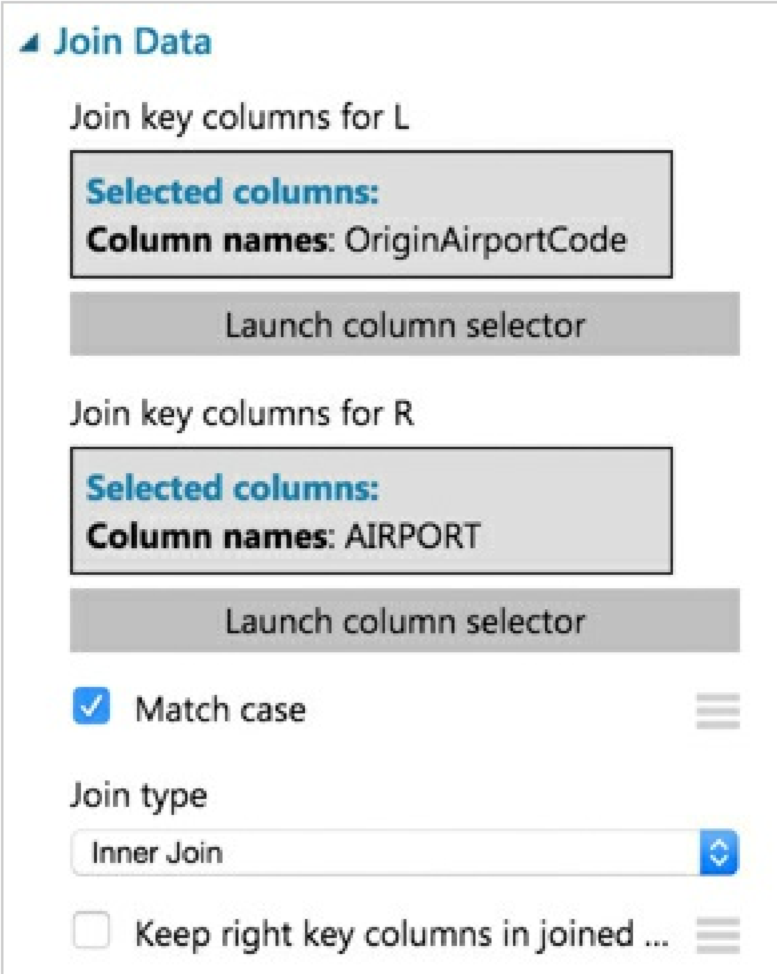



15. As we removed the latitude and longitude columns from the dataset to remove them as input to the web service, we have to add them back in before we return the result so that the results can be easily visualized on a map.

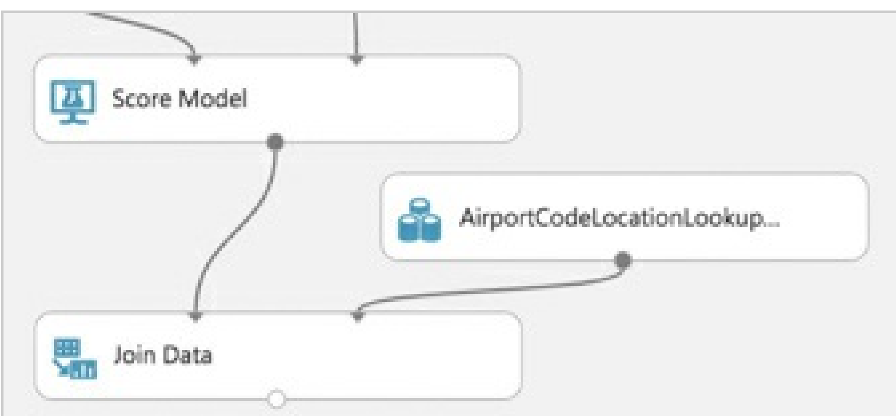
16. To add these fields back, begin by deleting the line between the Score Model and Web service output.

17. Drag the AirportCodeLocationLookupClean dataset on to the design surface, positioning it below the Score Model module.


18. Add a Join Data module. In the **Properties** panel for the Join Data module, for the Join key columns for L set the selected columns to **OriginAirportCode**. For the Join key columns for R, set the Selected columns to **AIRPORT**. Uncheck Keep right key columns in joined table.

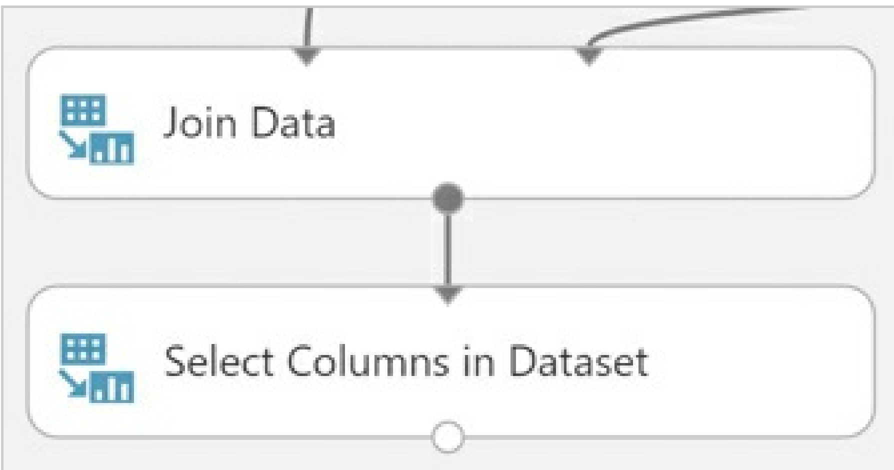



19. Connect the output of the Score Model module to the leftmost input of the Join Data module and the output of the dataset to the rightmost input of the Join Data module.

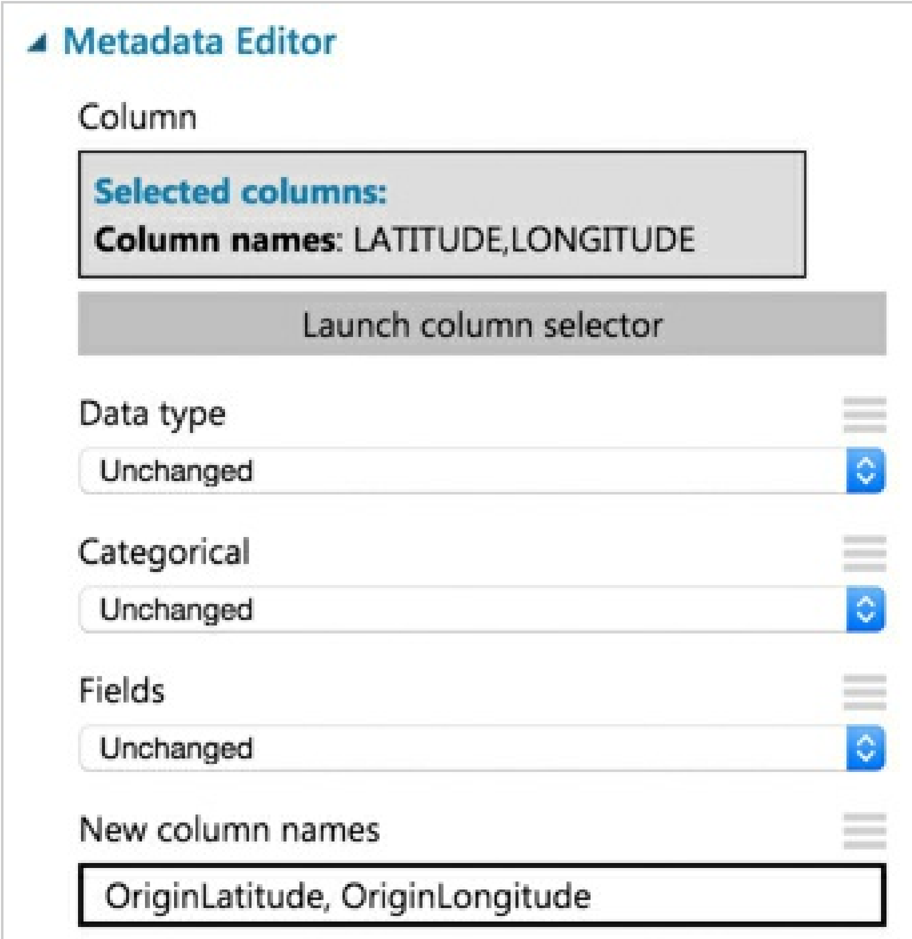



20. Add a Select Columns in Dataset module beneath the Join Data module. In the **Property** panel, begin with **All Columns**, and set the Selected columns to **exclude** the columns: **AIRPORT_ID** and **DISPLAY_AIRPORT_NAME**.

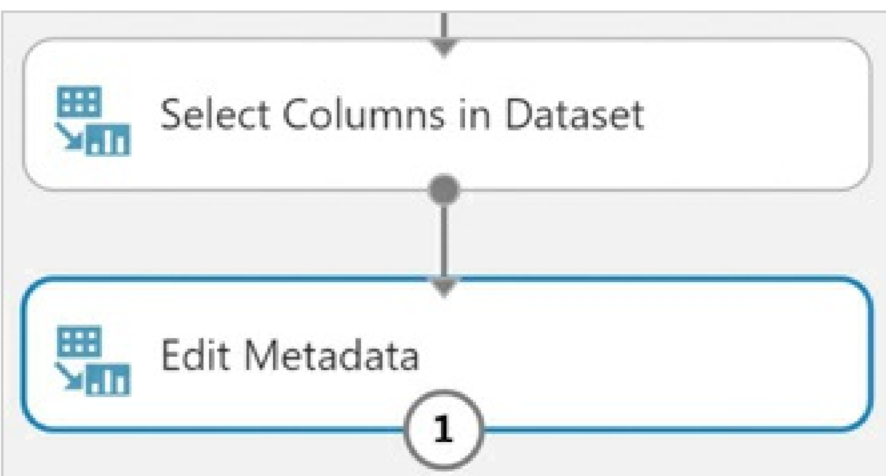

21. Connect the Join Data output to the input of the Select Columns in Dataset module.



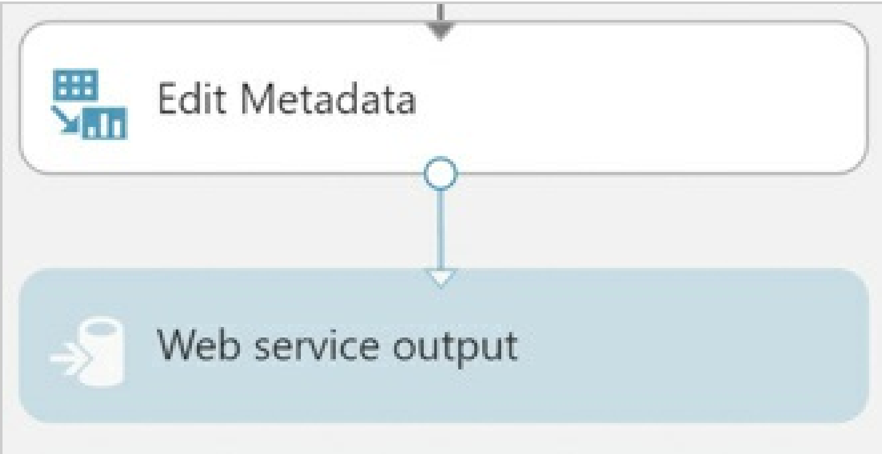

22. Add an Edit Metadata module. In the **Properties** panel for the Metadata Editor, set the Selected columns to LATITUDE and LONGITUDE. In the New column names enter: **OriginLatitude**, **OriginLongitude**.




23. Connect the output of the Select Columns in Dataset module to the input of the Edit Metadata module.

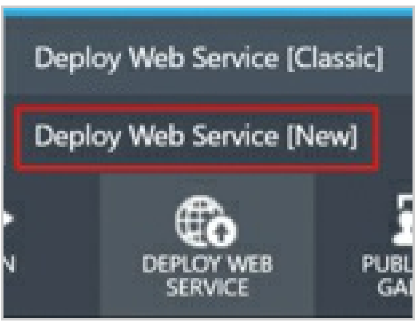



24. Connect the output of the Edit Metadata to the input of the web service output module.

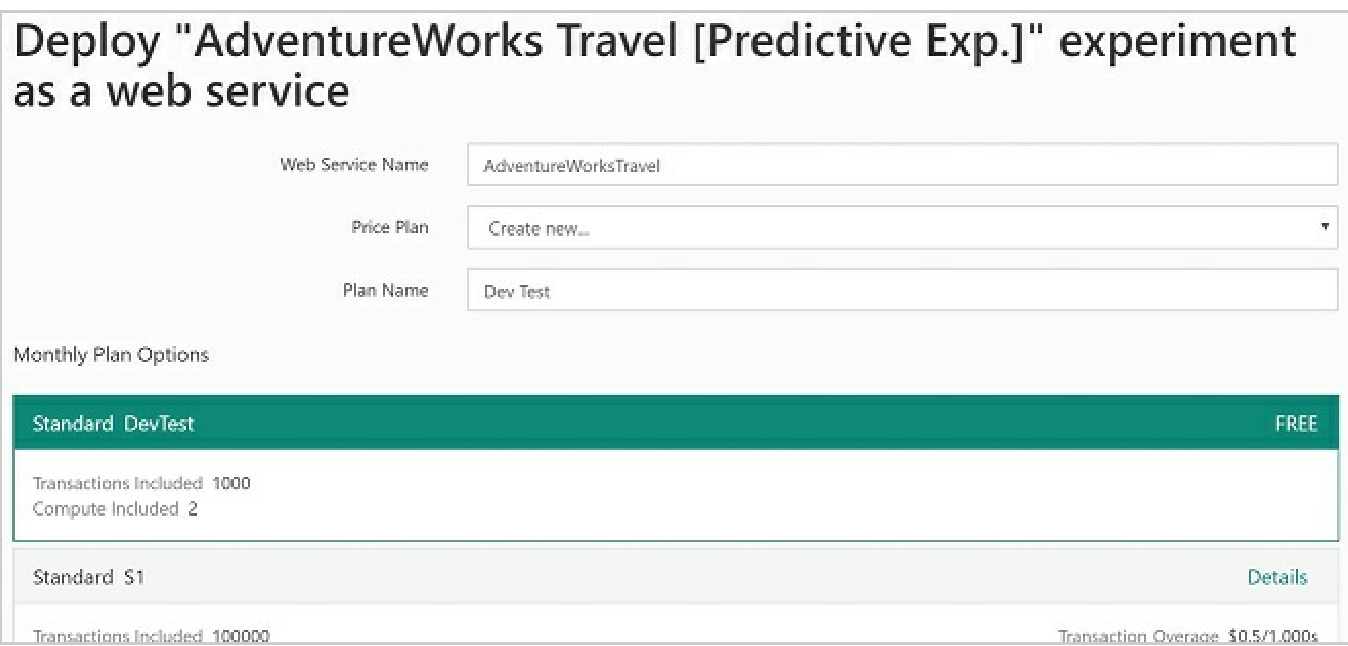



25. Run the experiment.

26. When the experiment is finished running, click **Deploy Web Service**, **Deploy Web Service [NEW]**.

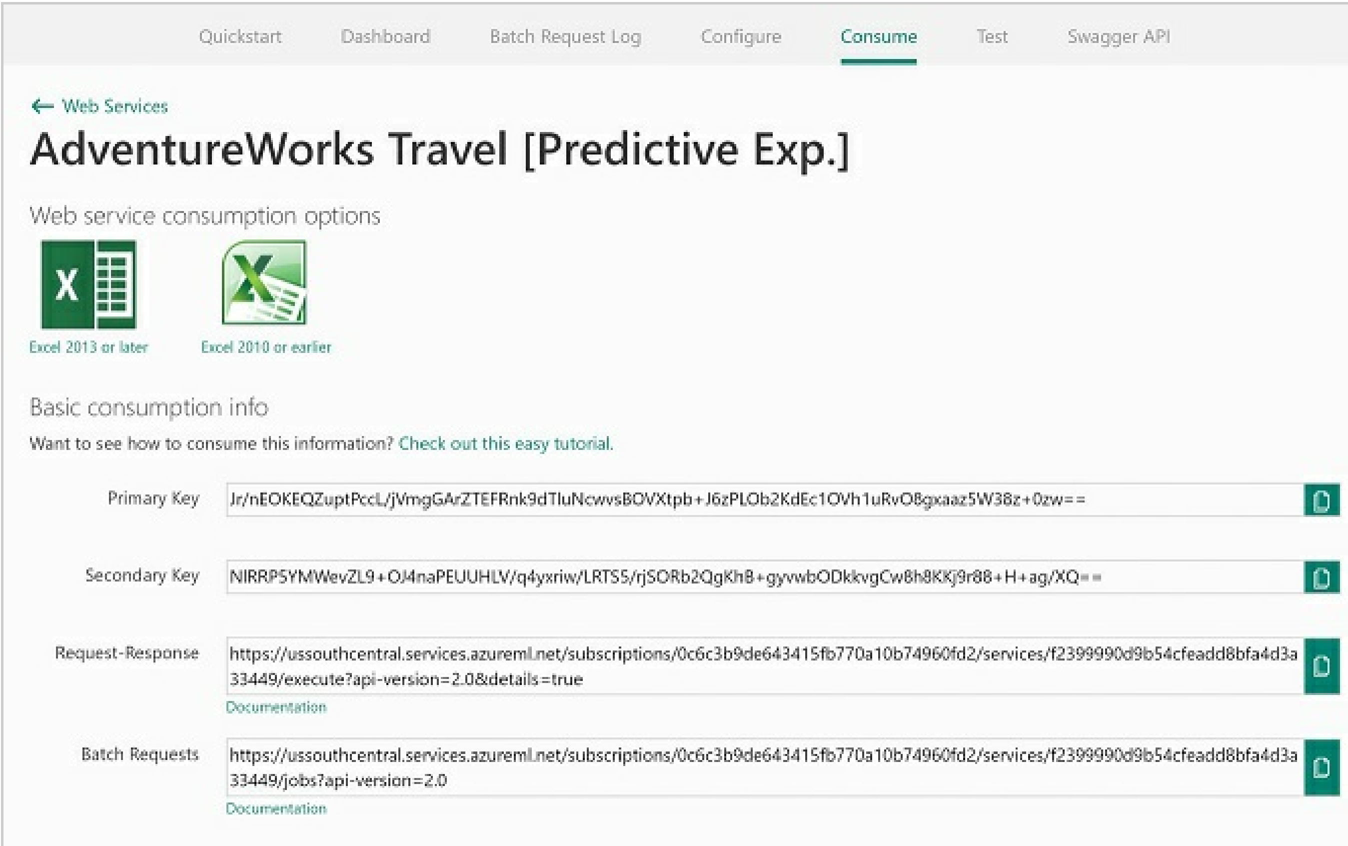



27. Enter a Web Service Name, then select the **Standard DevTest** (FREE) Monthly Plan Option.




28. Enter a Plan Name, then click **Deploy**.

29. When the deployment is complete, you will be taken to the Web Service Quickstart page. Click on the **Consume** tab.




Leave this page open for the next task.

## Task 8: Integrate the web service with the sample web app

1. From the Web Service consume page, copy the API's Primary Key.

2. With the sample web app open in Visual Studio, right-click the project properties, click Properties, and then navigate to the **Settings** tab.

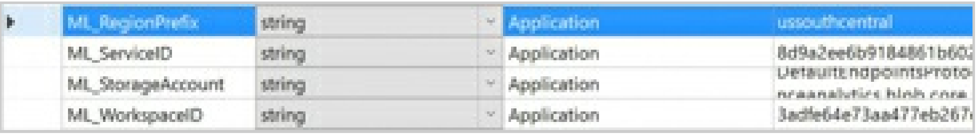
3. Paste the API KEY into the value column for the setting titled ML_APIKEY.


4. Scroll down until you see the **Request-Response** URL. From this URL you can get the Workspace ID and the Service ID.

   - **Example:** https://**ussouthcentral**.services.azureml.net/workspaces/**f8a24adcde7c45e1bb1c9ef** api-version=2.0&details=true

5. The first term (e.g., ussouthcentral) is the region prefix for your ML workspace. The first GUID after Subscriptions is your Workspace ID. The second GUID after services is your Service ID.

6. Copy each of these values into the settings ML_RegionPrefix, ML_WorkspaceID and ML_ServiceID respectively in the Settings area of the sample web app in Visual Studio.




7. Save the solution.

8. Publish the sample.

9. View the sample home page.

10. Fill out the flight criteria and click **Predict Delays** (remember to update the departure date so it is within 10 days of today). In addition to the weather forecast, you should see the delay prediction and confidence.

# Batch Score and Summarize Data

## **Exercise 4:** Batch score data with Azure ML and summarize with SQL Data Warehouse

**Duration:** 60 mins

**Synopsis:** In this exercise, attendees will batch score and prepare a summary of flight delay data in SQL Data Warehouse.

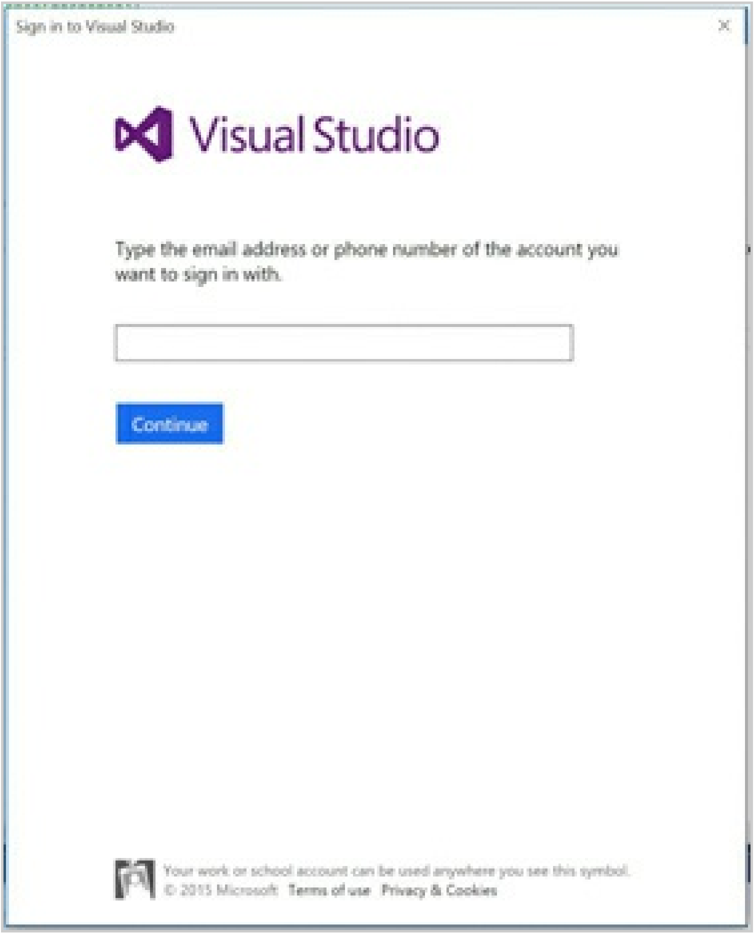
### Task 1: Batch score flight data

1. AdventureWorks has provided a sample CSV file to batch score using your Azure ML model. You can download it from http://bit.ly/1XGg6M5.

2. Open Visual Studio.

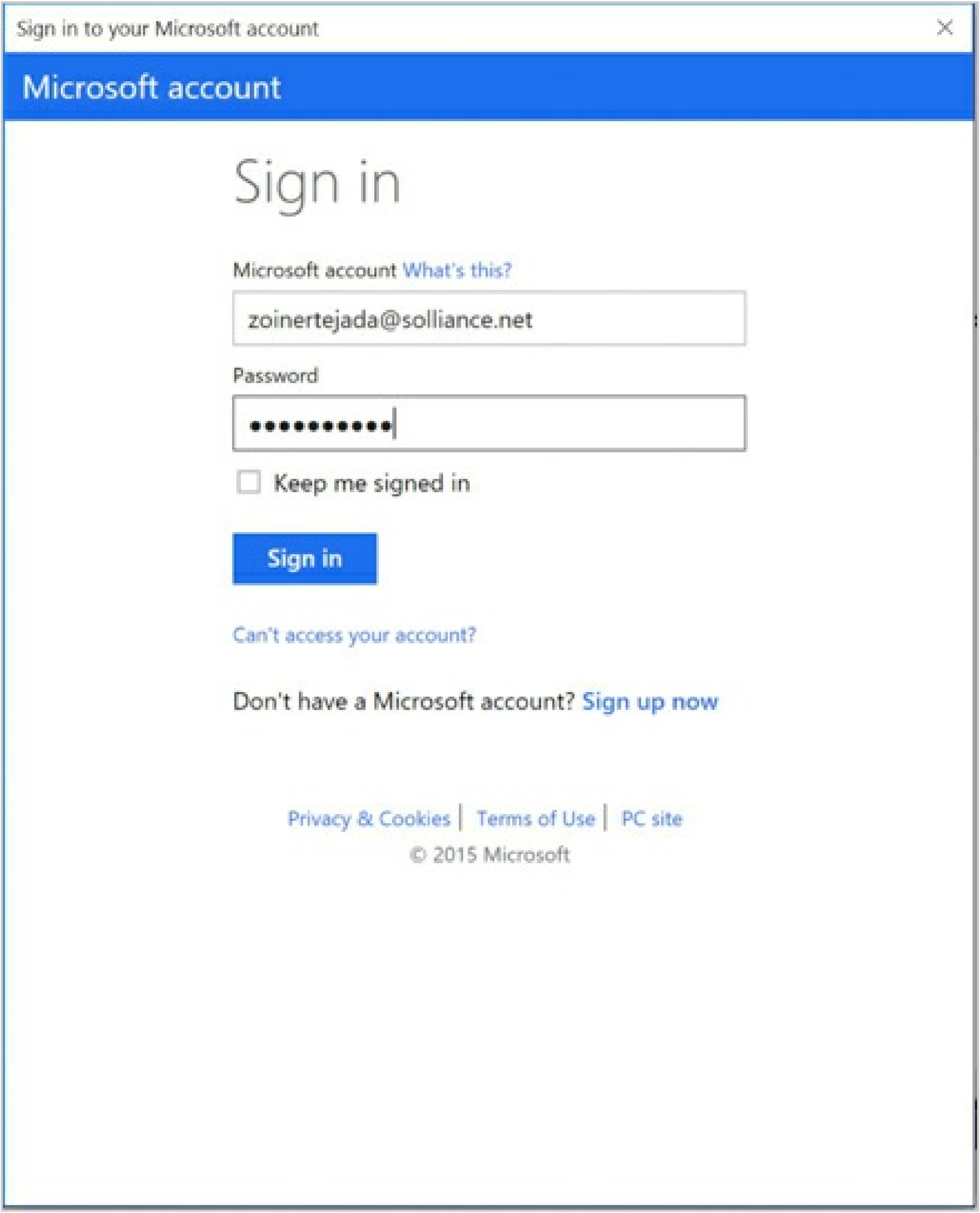
3. From the **View** menu, select **Cloud Explorer**.


4. If prompted, sign in to Visual Studio. Enter your login and click **Continue**.

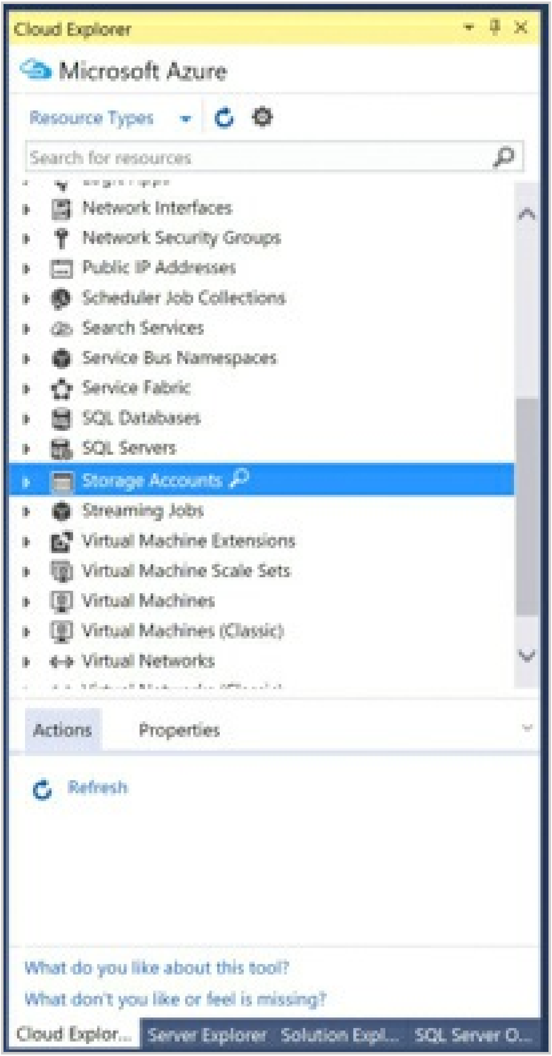



5. Select the appropriate option for work or school account or Microsoft account.

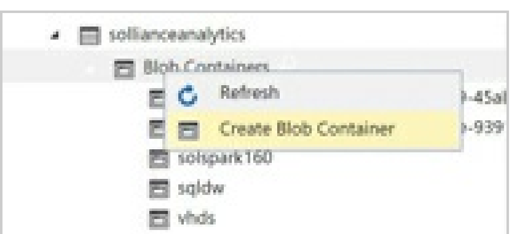

6. Enter your password and click **Sign In**.




7. Allow Cloud Explorer about 30 seconds to load your subscription resources.
8. Expand **Storage Accounts**. It may take a few moments to load.

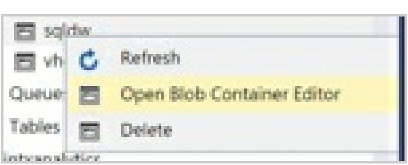



9. Expand the Storage Account you created for this workshop, then right-click **Blob Containers** and select **Create Blob Container**.



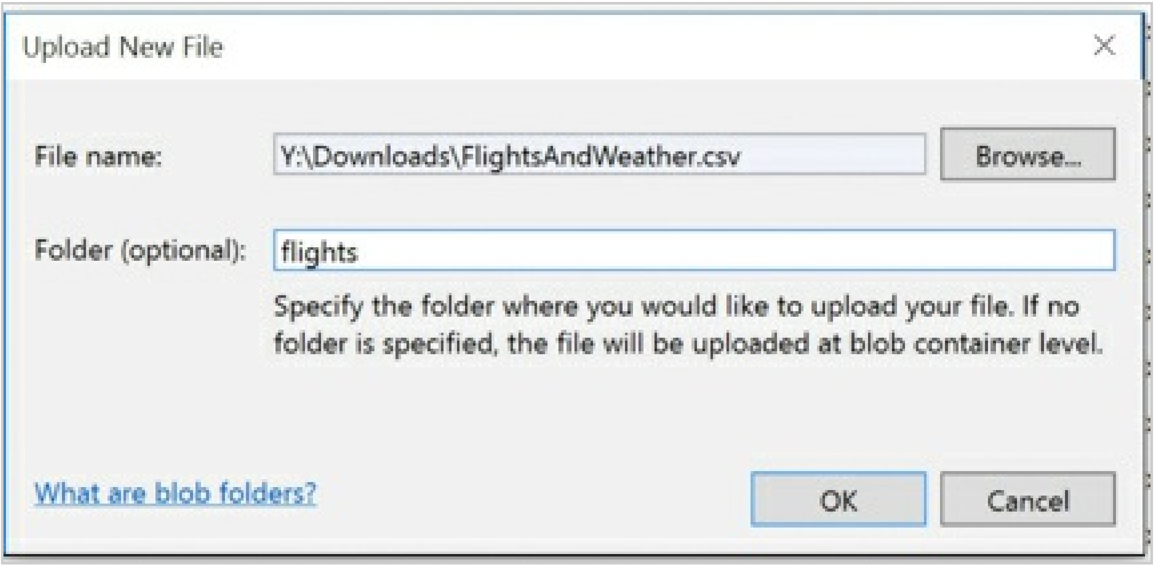

10. Provide a name for the container.

11. Right-click the new container, and select **Open Blob Container Editor**.

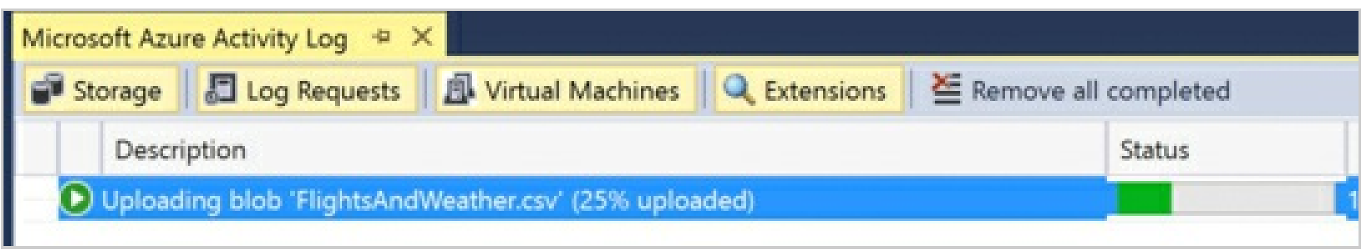



12. In the document that appears, click **Upload Blob**.

13. In the Upload New File dialog box, browse to the FlightsAndWeather.csv file you downloaded and select it. Enter flights in the Folder field, and click **OK**.




14. Wait for the upload to complete.



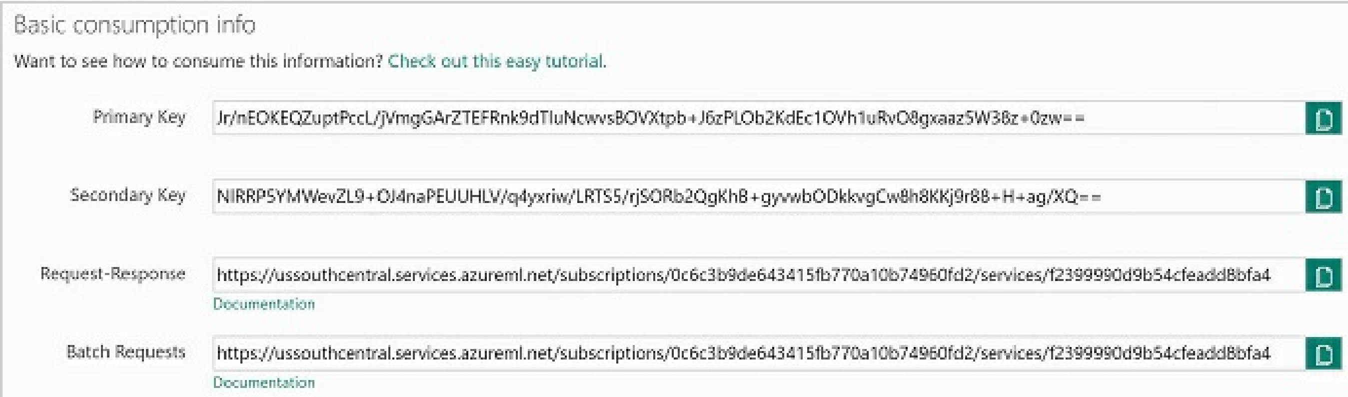

15. Open a browser and navigate to Azure Machine Learning Web Services and click **Web Services**. You can always get there by going to https://services.azureml.net/.

16. Click on your web service.

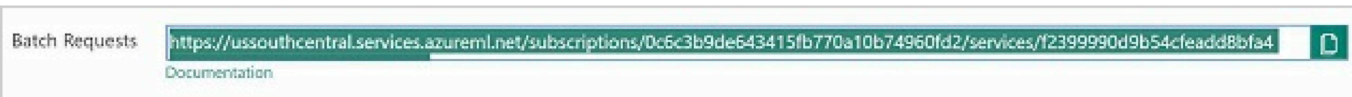
17. Click on the **Consume** tab.

18. On this page, take note of the PRIMARY KEY below the Basic Consumption Info section. You will need this value in a subsequent step.

19. Also within this section, observe the BATCH REQUESTS URL.



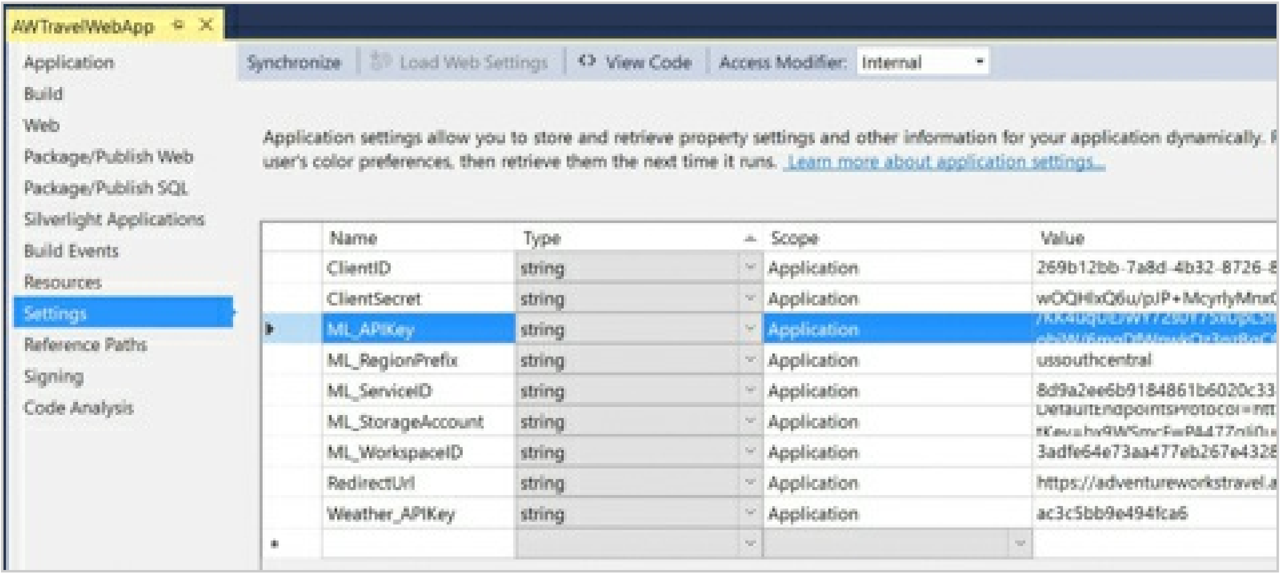

20. The sequence of letters and numbers following subscriptions/ is your workspace ID. Take note of this ID. The sequence of letters and numbers following services/ is your service ID, take note of this ID as well.




21. With the sample app open in Visual Studio. In Solution Explorer, right-click the project, select **Properties**, and then click the **Settings** tab.

22. Verify the values for the following settings from your published Machine Learning service (they should be exactly the same as what you used for the request/response task earlier):

```
ML_APIKey: PRIMARY KEY for your Machine Learning web service
ML_StorageAccount: The connection string to the Storage Account where you uploa
ded the flights CSV.
ML_WorkspaceID: The ID of the Machine Learning Workspace.
ML_ServiceID: The ID of your Machine Learning web service.
ML_RegionPrefix: The name of the region in which your ML service is deployed.
```




23. Save the solution.

24. Publish the solution.

25. Navigate to your deployed sample in the browser, but instead of going to the homepage go to BatchScore, for example:

    - http://<yourwebapp>.azurewebsites.net/BatchScore

26. Modify the values of the relative paths for the location of the file to score and the location of scored output to that it is correct for your environment, for example:

    - Relative location of file to score: **<ContainerName>**/flights/FlightsAndWeather.csv
    - Relative location of scored output file: **<ContainerName>**/flights/Scored_FlightsAndWeather.csv

27. Click **Submit Job**.

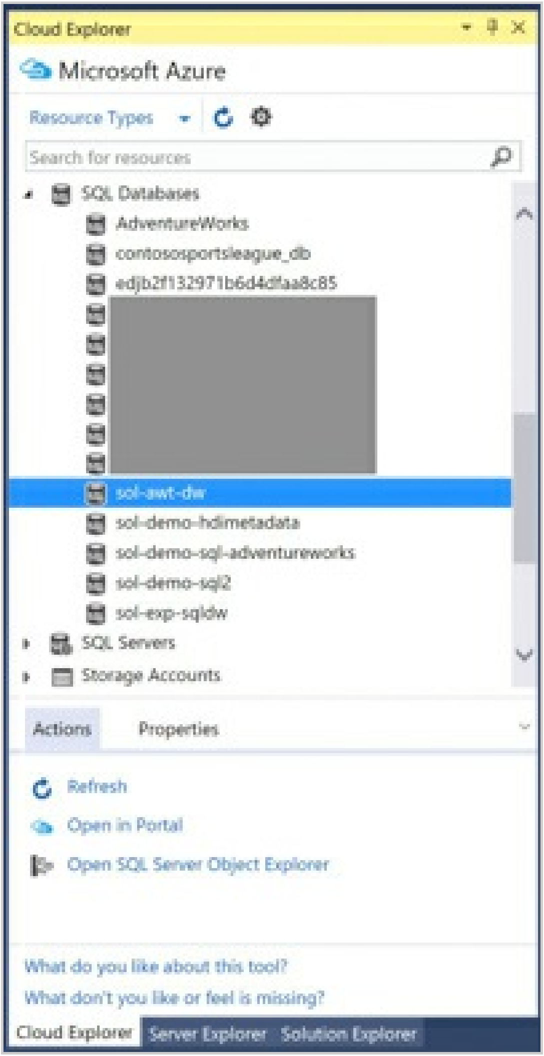
28. When a Job ID appears, click **Start Job** to begin the batch scoring process.

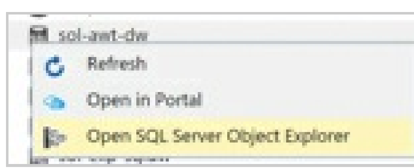
29. Periodically click check job status until you see the job status of Job is finished. This will take 1-2 minutes. You now have a scored CSV ready for summarization using SQL Data Warehouse.

## Task 2: Summarize delays by airport

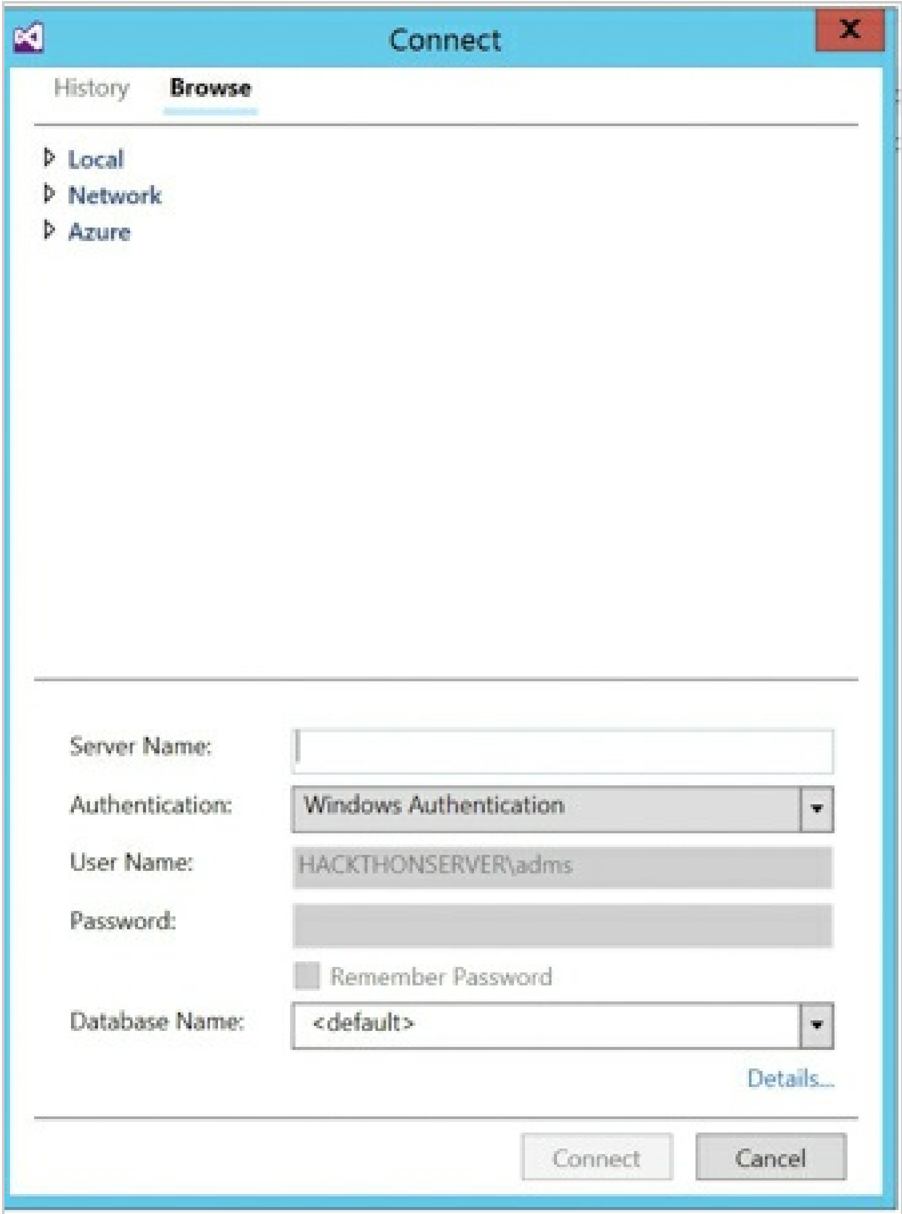
1. Using Visual Studio, go to Cloud Explorer by selecting **View**, **Cloud Explorer**.


2. Expand SQL Databases

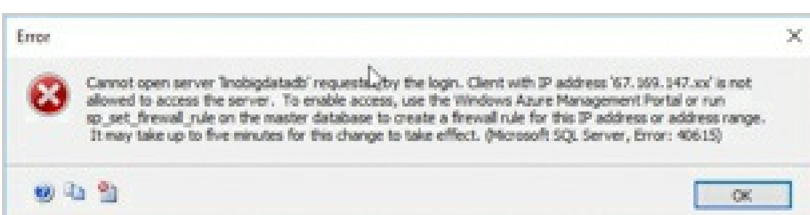



3. Right-click your SQL Data Warehouse and select **Open SQL Server Object Explorer**. If this fails with an error, you can always open SQL Server Object Explorer from the View menu and then add your SQL Server manually. 

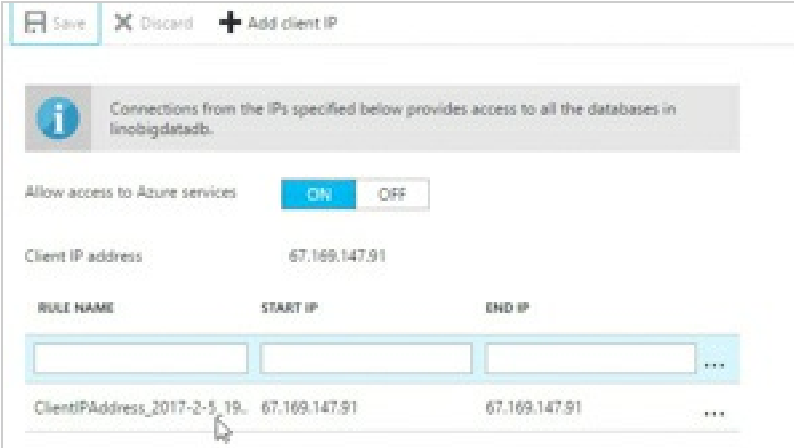

   - Use the Connect dialog as displayed below to connect to your SQL Server manually if the automatic connection failed:



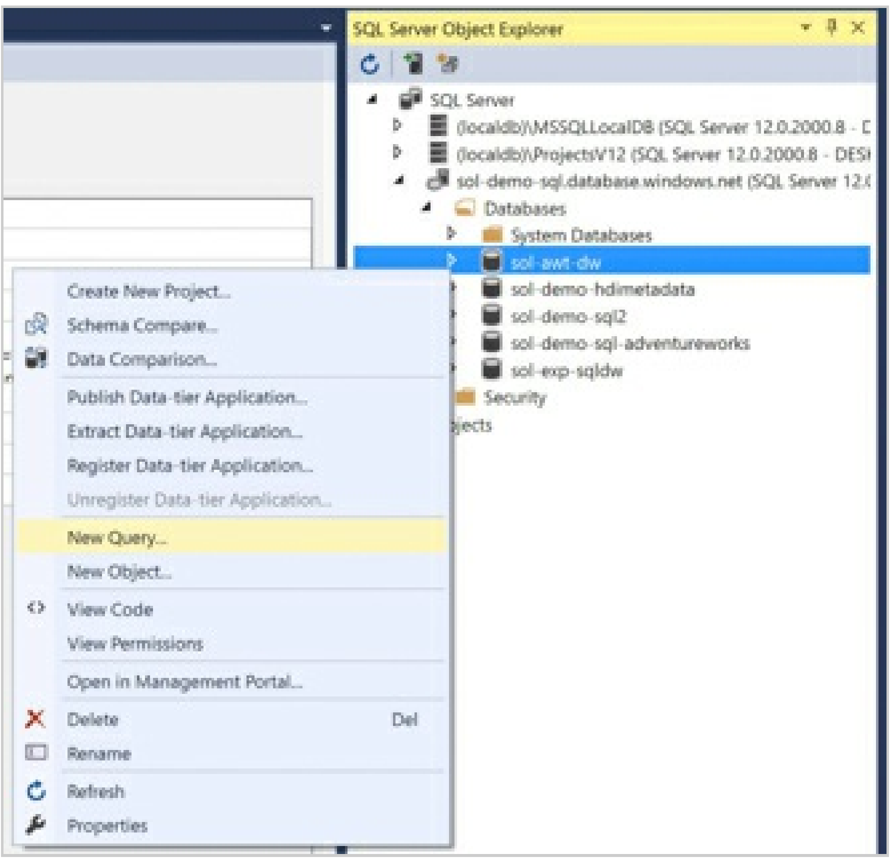

- If you get an error like the one below, it is probably because your IP Address is not added as a client that can access the SQL Database




- You will need to head to your SQL Database blade in Azure Portal and choose Firewall and click on the “Add Client IP”




4. In SQL Server Object Explorer, right-click your SQL Data Warehouse database and select **New Query**.




5. In the document that appears, paste the following query, which stores the connection information to your Azure Storage Account (that contains the Scored_FlightAndWeather.csv) and defines a particular container and path as the root for subsequent steps. Substitute in your Storage Account Name and Storage Account Key (replacing the <StorageAccountKey>, <Container>, and <StorageAccountName> in the query with the corresponding values you can get from the Azure Portal).

```
CREATE MASTER KEY;

CREATE DATABASE SCOPED CREDENTIAL ASBSecret
WITH IDENTITY = 'flightdelaysclient'
,    Secret = '<StorageAccountKey>'
;

CREATE EXTERNAL DATA SOURCE azure_storage
WITH
(
    TYPE = HADOOP
,   LOCATION ='wasbs://<Container>@<StorageAccountName>.blob.core.windows.net/f
lights'
,   CREDENTIAL = ASBSecret
);

CREATE EXTERNAL FILE FORMAT text_file_format
WITH
(
    FORMAT_TYPE = DELIMITEDTEXT
,   FORMAT_OPTIONS  (
                        FIELD_TERMINATOR =','
                    ,   USE_TYPE_DEFAULT = TRUE
                    )
);
```

6. Run the Query by clicking the **Execute** icon . The first time you run a query this way, you may get an error dialog box which you can safely ignore. All error messages will be output in the output pane just below the query window.

7. Next, append the following query which creates an external table (e.g., a table structure that is superimposed over the files in Blob storage).

```
CREATE EXTERNAL TABLE FlightDelays(OriginAirportCode char(3), Month tinyint, Da
y tinyint, Hour tinyint, DayOfWeek tinyint, Carrier varchar(4), DestAirportCode
 char(3), DepDelay15 bit, WindSpeed smallint, SeaLevelPressure decimal(4,2),
HourlyPrecipitation decimal(8,5), DelayPredicted bit, DelayProbability decimal(
5,4),   OriginLatitude decimal(14,10), OriginLongitude decimal(14,10))
WITH
(
LOCATION = '/Scored_FlightsAndWeather.csv',
DATA_SOURCE = azure_storage,
FILE_FORMAT = text_file_format,
REJECT_TYPE = value,
REJECT_VALUE = 100000
);
```

8. Highlight the query you just added and click **Execute**. (By highlighting the query, only the selection is executed.) When this completes, you should have the FlightDelays table you can query using T-SQL.

9. Run the following T-SQL to view a sample of the contents.

```sql
SELECT Top 100 * FROM FlightDelays;
```

10. You should see output similar to the following:

| | OriginAirportCode | Month | D... | Ho... | DayOfWeek | Carrier | DestAirportCode | DepDelay15 | WindSpeed | SeaLevelPressure |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | BWI | 5 | 5 | 21 | 7 | WN | ATL | 1 | 10 | 30.19 |
| 2 | BOS | 8 | 29 | 14 | 4 | DL | ATL | 1 | 15 | 30.06 |
| 3 | MEM | 7 | 20 | 9 | 6 | FL | ATL | 0 | 9 | 29.92 |
| 4 | LAX | 9 | 5 | 15 | 4 | WN | SLC | 1 | 13 | 29.83 |
| 5 | MSY | 10 | 13 | 7 | 7 | UA | IAH | 0 | 0 | 30.13 |
| 6 | HOU | 10 | 31 | 8 | 4 | WN | EWR | 0 | 13 | 29.92 |

11. Create a View called FlightDelaysSummary by copying and pasting the following SQL and then executing it:

```sql
CREATE VIEW FlightDelaysSummary AS
SELECT  OriginAirportCode, Cast(OriginLatitude as varchar(15)) + ',' + Cast(Ori
ginLongitude as varchar(15))  OriginLatLong, Month, Day, Hour, Sum(Cast(DelayPr
edicted as int)) NumDelays, Avg(DelayProbability) AvgDelayProbability
FROM FlightDelays
WHERE Month = 4
GROUP BY OriginAirportCode, Cast(OriginLatitude as varchar(15)) + ',' + Cast(Or
iginLongitude as varchar(15)) , Month, Day, Hour
Having Sum(Cast(DelayPredicted as int) ) > 1
```

12. Examine the contents of this view by running the following query.

```sql
SELECT * FROM FlightDelaysSummary
```

13. You are now ready to visualize this data with PowerBI.

# Visualizing in Power BI

## **Exercise 5:** Visualizing in Power BI

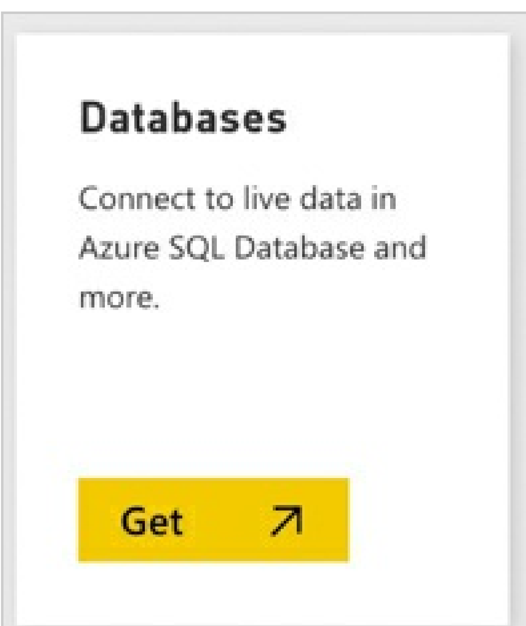
**Duration:** 20 mins

**Synopsis:** In this exercise, attendees will construct a report in Power BI that uses the map visualization to illustrate the predicted delays, using the data originally scored using Machine Learning, but summarized in a View within SQL Data Warehouse.

### Task 1: Connect Power BI to SQL Data Warehouse

1. Log in to Power BI by going to app.powerbi.com and entering your credentials.
2. Click **Get Data** from the bottom-left corner of the navigation bar.


3. In the page that appears, click **Get** located within the Databases tile.

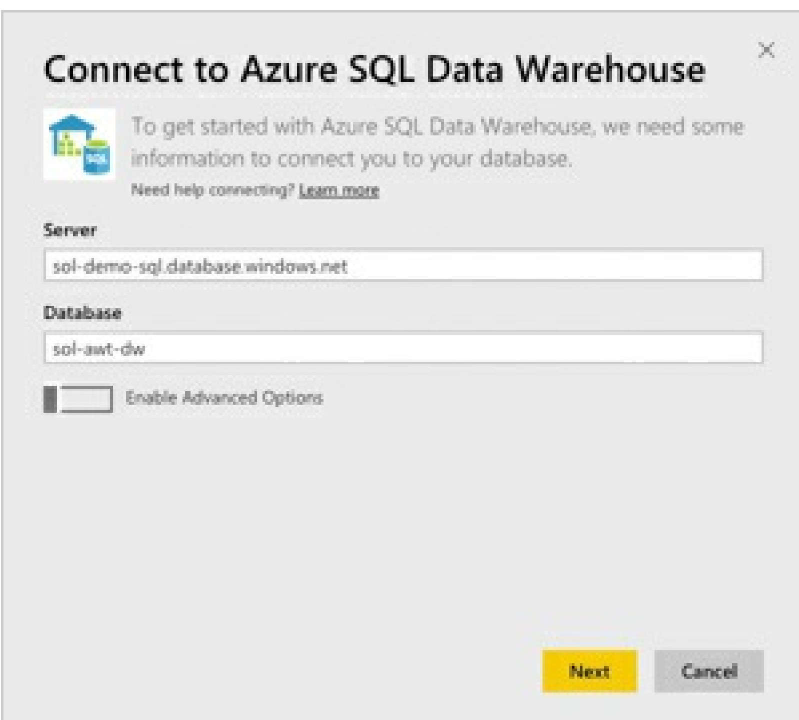


4. In the page of tiles that appears, click the **Azure SQL Data Warehouse** tile.


5. Click **Connect**.

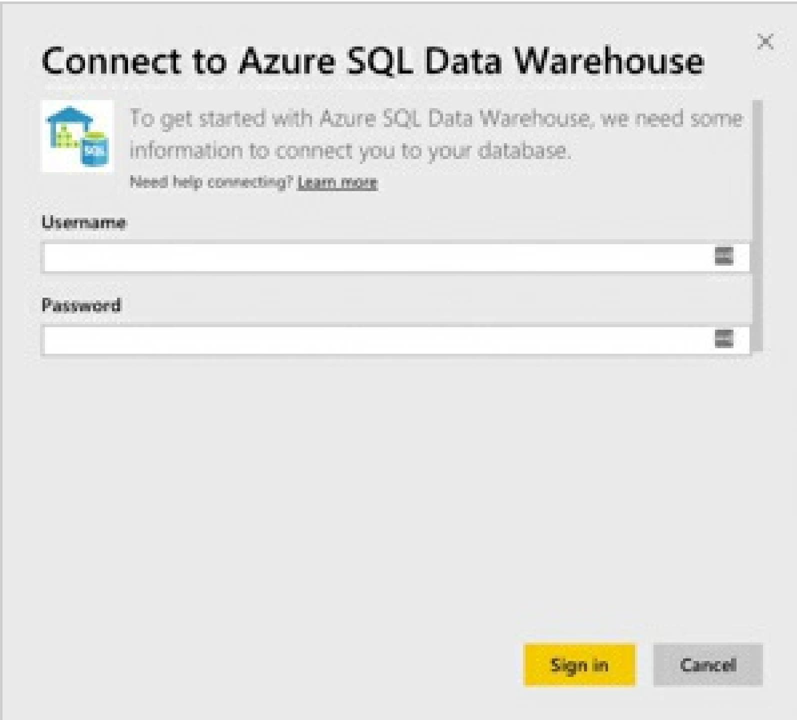

> **If prompted...**
>
> click on the button to start a free trial of Power BI Pro, which is required for connecting to SQL Data Warehouse.

6. In the dialog box, enter the fully qualified name to the SQL Server hosting your SQL Data Warehouse, and then provide the name of the database.



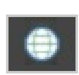
7. Click **Next**.

8. Enter your Username and Password and click **Sign In**.

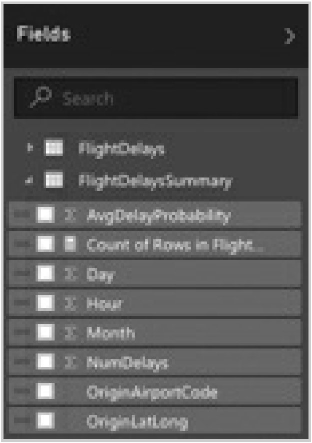



9. Give the system a few moments to connect and load the metadata.


10. When the dialog box disappears, look for the Azure SQL Data Warehouse tile on your dashboard. Select the tile.

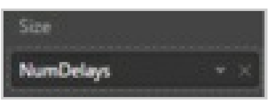

11. In the Visualizations, click the **Globe** icon  to add a Map visualization to the report design surface.

12. With the Map visualization still selected, in the Fields area at right, expand the table called FlightDelaysSummary.

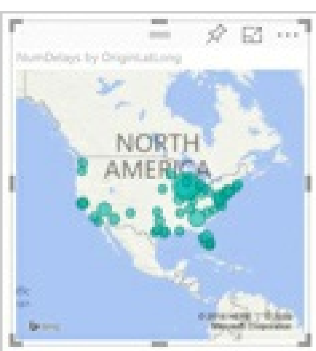



13. Click and drag the field labeled OriginLatLong and drop it into the **Location** field located just below visualizations.


14. Next, drag the field labeled NumDelays and drop it into the **Size** field.




15. Give the map a few moments to update with the new data. Your map should look something like following:

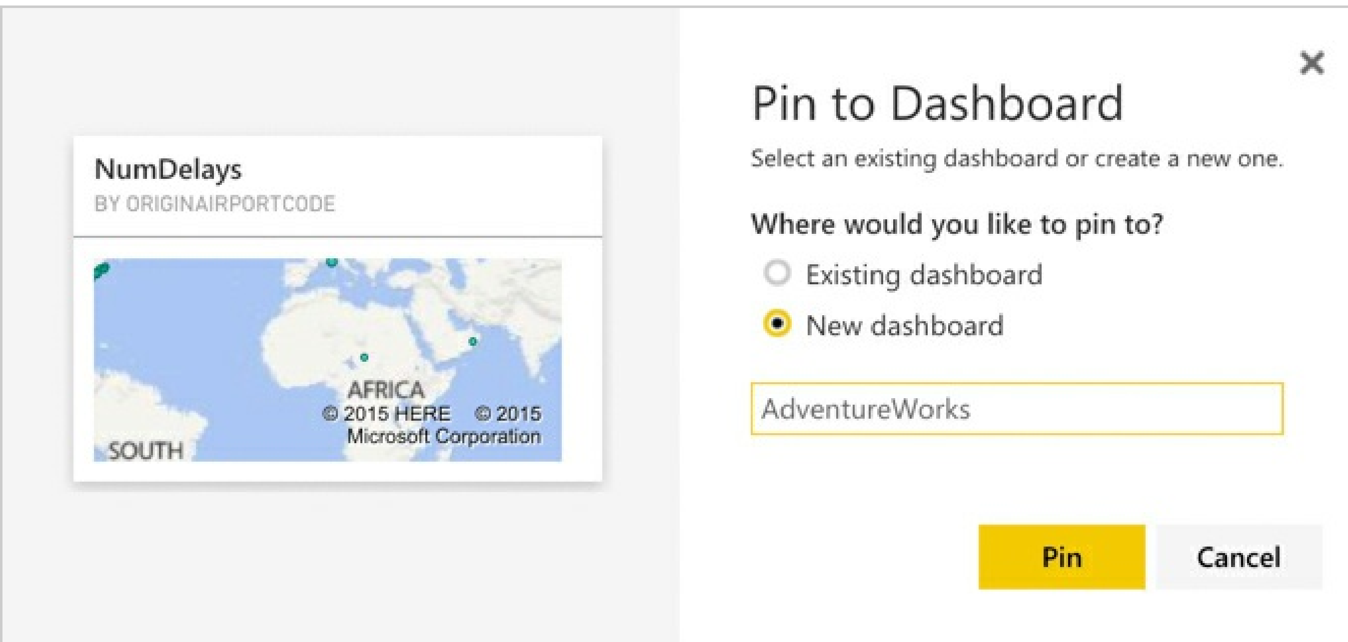



## Task 2: Create a dashboard

1. In order to create a dashboard, you need to first save your report. To give this report a name, click **File** and then click **Save**.

2. Enter FlightDelays in the field and click **Save**.

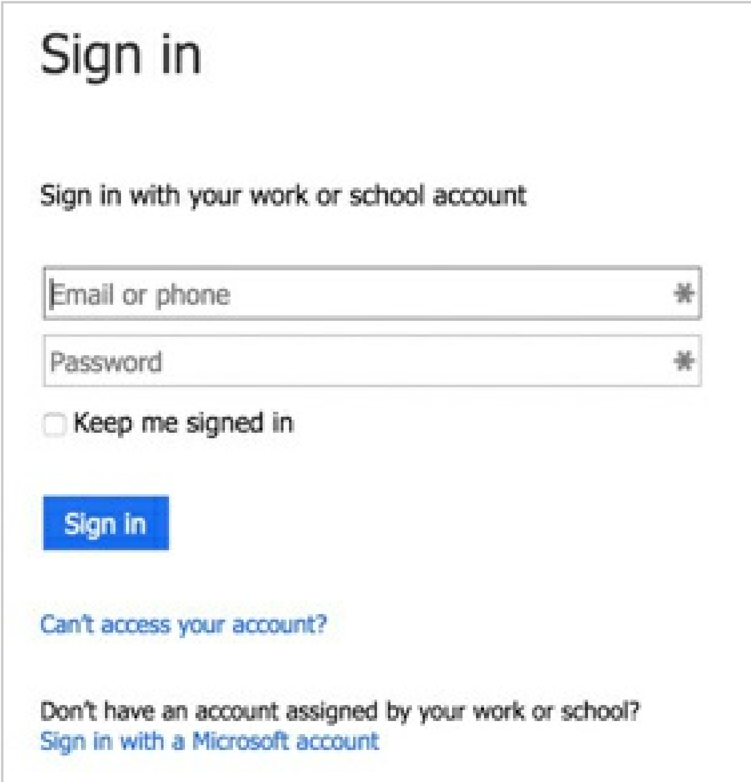

3. Next you will create a dashboard from this report.
4. Click your newly created FlightDelays report.
5. At the top-right of the map visual, click the **Pin** icon.
6. In the dialog box, select **New dashboard** and name the dashboard AdventureWorks



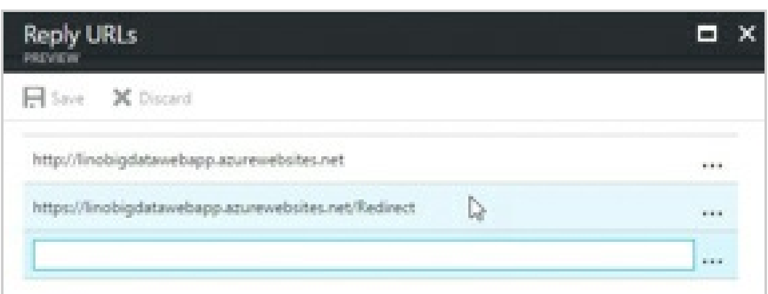

### Task 3: Observe the batch score results

1. Navigate to the deployed sample and click **Sign in to Power BI**.


2. When prompted, sign in to Power BI with your appropriate credentials.




3. If you get an error that the Reply redirect was not expected from the Azure Active Directory, make sure to open the AD in the Azure portal and confirm that the Reply URL with the "/Redirect" is added




4. Your browser will redirect back to the sample web app and the Power BI tile with the flight delays should appear.

AdventureWorks Travel
NumDelays
From
SAN
To
SEA
Date
11/15/2015
Time
5:00 PM
Carrier
Enter a carrier
PREDICT DELAY
WEATHER FORCAST
DELAY PREDICTION
expect delays
(73% confidence)
Copyright © 2015 | AdventureWorks Travel, inc., all rights reserved
WEATHER UNDERGROUND

# Cleanup

## **Exercise 6:** Cleanup

**Duration:** 10 mins

**Synopsis:** In this exercise, attendees will deprovision any Azure resources that were created in support of the workshop.

1. Delete the Resource Group in which you placed your SQL Data Warehouse, Storage Account and App Services.

    1. From the Portal, navigate to the blade of your Resource Group and click Delete in the command bar at the top.

    2. Confirm the deletion by re-typing the resource group name and clicking Delete.

2. Delete the Machine Learning Workspace

    1. From the Manage Portal, select Machine Learning.

    2. In the list of Workspaces, select the workspace you created.

    3. Click Delete in the command bar at the bottom.

    4. When prompted to confirm the deletion, click Yes.