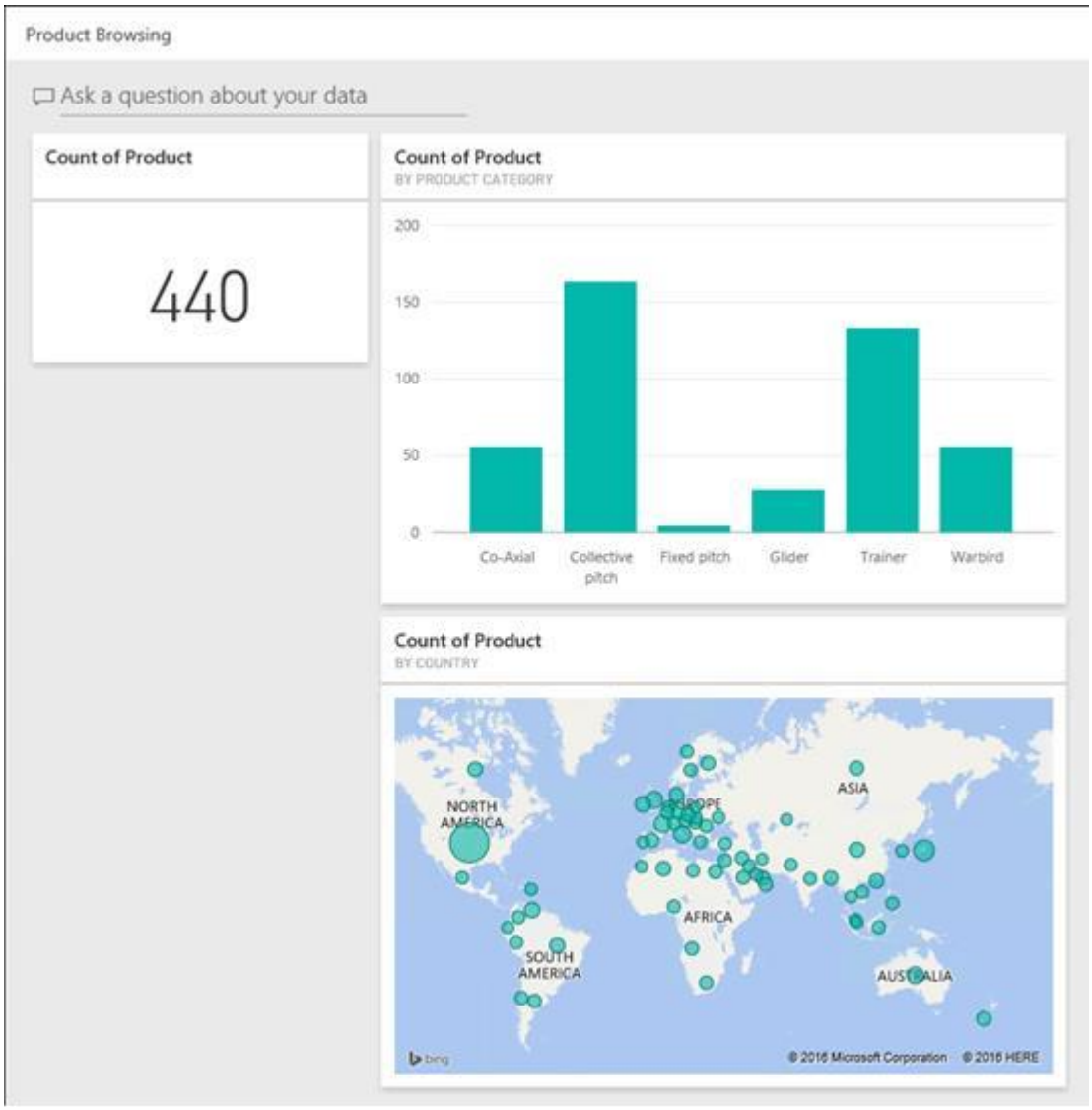
Delivering a Power BI RealTime Dashboard

Overview

The estimated time to complete the lab is 45 minutes

*You must have completed **Lab 02A** before commencing this lab.*

In this lab, you will produce a real-time dashboard by using the Power BI REST API. You will use a pre-developed application to explore each of the Power BI REST API methods, and to produce the following real-time dashboard.



Developing a Real-Time Power BI Dashboard

In this exercise, you will produce a real-time dashboard by using the Power BI REST API.

Exploring the Power BI API

In this task, as a developer, you will open a pre-developed C# Windows Forms project. You will then use the application experience to work with the Power BI REST API.

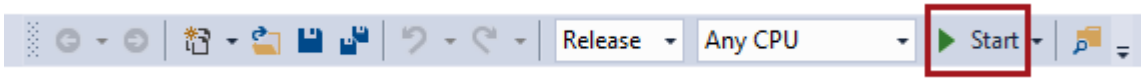
1. Open Visual Studio.

   The lab steps have been written and tested by developing a solution with Visual Studio 2017. The labs will also work in Visual Studio 2015 and Visual Studio Community; however, screenshots and lab instructions may differ slightly.

2. On the **File** menu, select **Open | Project/Solution**.
3. In the **Open Project** window, navigate to the **C:\Student\Modules\12_StreamingDatatsets\Lab\Project** folder.
4. Select **PowerBiApiExplorer.sln**, and then click **Open**.

   *If you are a developer experienced working with Visual Studio, you can enable breakpoints to allow stepping through the C# methods that implement Power BI REST API calls. To do so, open the **PowerBI.cs** file, and then on the **Debug** menu, select **Enable All Breakpoints**.*

5. To run the application, on the toolbar, click **Start**.



   This application has been designed to explore six Power BI REST API methods. Once connected, the left pane presents a treeview representation of the Power BI service and your datasets. The right pane presents console output to report on the REST API calls.

   *Before making any calls, the application must authenticate the pre-registered Azure Active Directory application. The authentication process will require the **Client ID** and **Redirect URI** values assigned to the registered application.*

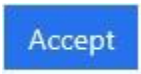
6. In the **Power BI API Explorer** window, on the **Application** menu, select **Options**.
7. In the **Options** window, hover the cursor of each of the help icons to read the help information.
8. Inside the **Client ID** box, use **Control-V** to paste the Client ID stored in the **MySolution.txt** file.

   *The **MySolution.txt** file was updated with the Client ID value in **Lab 02A**.*

9. Click **OK**.
10. To connect to the Power BI service, on the **Application** menu, select **Connect**.
11. In the **Sign In** window, enter your Power BI credentials.
12. Click **Sign In**.

13. When prompted, to acknowledge the permissions required, click **Accept**.

    These are the permissions that you delegated to the app in the previous task.



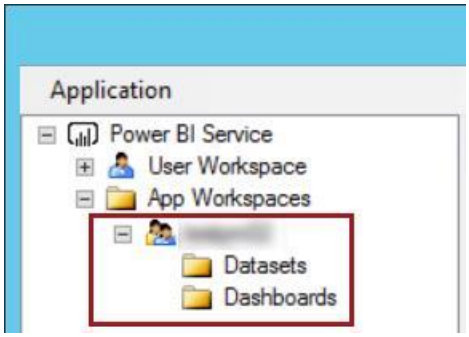
14. Review the console output reporting that an access token was acquired and cached for use by subsequent Power BI REST API calls.

```
> Started
> Calling GetAccessToken()
> Getting access token... New token acquired and cached
>
```

    The access token will accompany all requests sent to the Power BI service, and is required to authenticate each request.

15. To access your app workspace, right-click the **App Workspaces** folder, and then select **Get App Workspaces**.

16. To create a new dataset, in the left pane, expand your app workspace node.



17. Right-click the **Datasets** folder, and then select **Create Dataset**.

18. In the **Create Datasets** window, inside the **Dataset Schema** box, enter the following JSON document.

    For convenience, the JSON document can be copied from the **C:\Student\Modules\12_StreamingDatatsets\Lab*Snippets.txt*** *file.*

    Use ***Control-V*** *to paste the text into the box.*

JSON

```
{
 "name": "E-Commerce Operations",
 "tables": [
         {
                "name": "Product Browsing",
                "columns": [
                       {
                              "name": "Product",
                              "dataType": "String"
                       },
                       {
                              "name": "Product Category",
                              "dataType": "String"
                       }
                ]
         }
 ]
}
```

*The JSON document describes a new database named **E-Commerce Operations** consisting of a single table named **Product Browsing**. The table consists of three columns, each of type string.*

19. For the **Default Retention Policy** dropdown list, read the help information.

    *The maximum retention for the **None** option is five million rows.*

20. In the **Default Retention Policy** dropdown list, select **basicFIFO**.

21. Click **OK**.
22. Review the console output reporting that the access token was retrieved from cache, and the HTTP action and URI.
23. Notice also that the JSON document was added to the request body.
24. In the left pane, hover the cursor over the **E-Commerce Operations** dataset, and then review the tooltip describing the properties of the object.

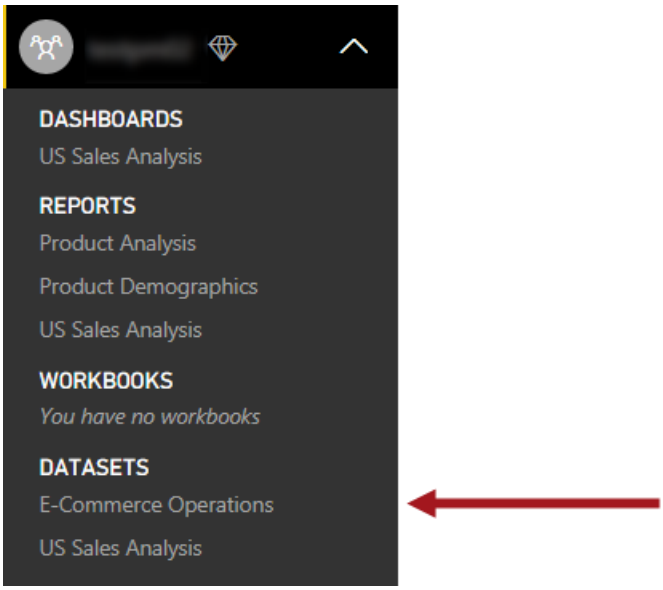
    *The **id** property (GUID) will be used to retrieve the dataset tables in the next step.*

25. To retrieve the dataset tables, in the left pane, right-click the **Tables** folder, and then select **Get Tables**.
26. Review the console output reporting the HTTP action and URI, and in particular the use of the dataset **id** property.
27. In the left pane, hover the cursor over the **Product Browsing** table, and then review the tooltip describing the single property of the object: **name**.
28. Leave the application running.

Creating a Dashboard

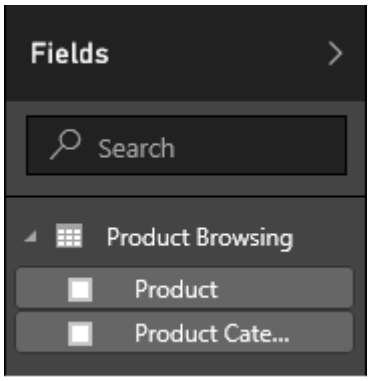
In this task, as an analyst, you will create a report based on the dataset, and then pin two visualizations to a dashboard.

1. Switch to the Power BI portal web browser.
2. Reload (F5) the browser.
3. In the **Navigation Pane**, expand the app workspace.
4. In the **Datasets** group, notice the **E-Commerce Operations** dataset that you just created.

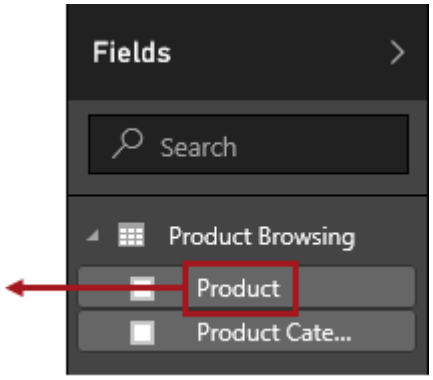


5. To create a report based on the dataset, in the **Navigation Pane**, click the **E-Commerce Operations** dataset.

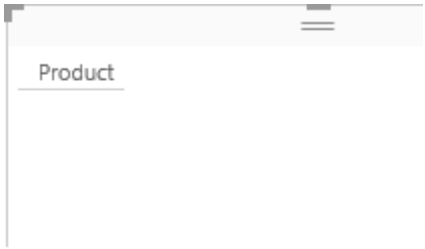
6. In the report designer, at the right, notice the **Fields** pane, and notice that it describes the table and two columns as defined in the JSON document used to create the dataset.



7. To create a visualization, from inside the **Fields** pane, drag the **Product** field and drop it at the top-left of the report canvas.

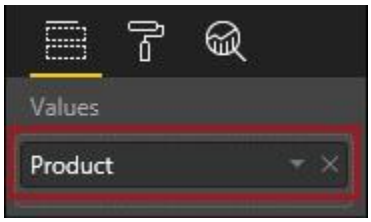


8. Notice that a single-column table is produced.

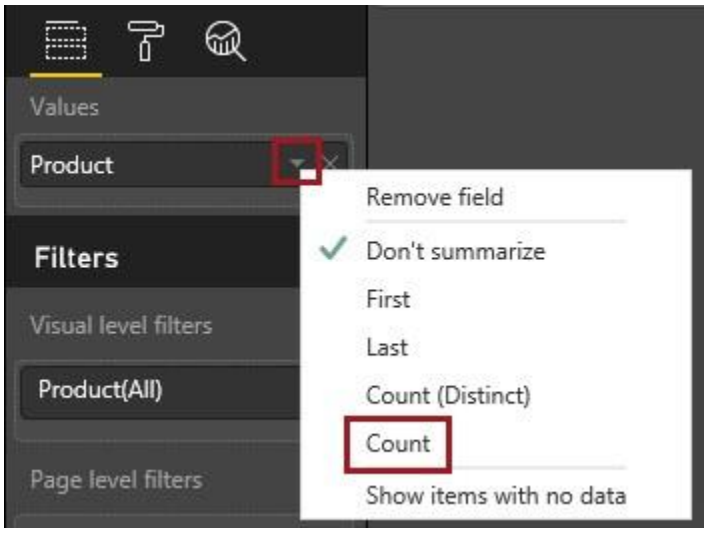


Presently the dataset contains no rows.

9. In the **Visualizations** pane, notice that the **Product** field was added to the **Values** drop zone.

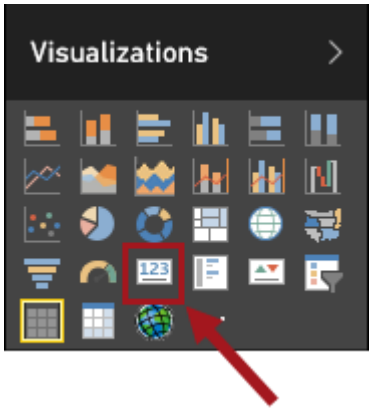


10. In the **Values** well, click the down-arrow, and then select **Count**.



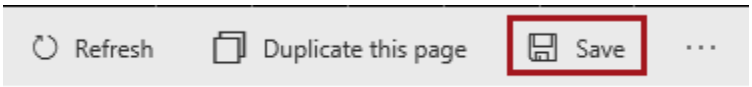
11. Change the visualization type to **Card**.

Tip: Hovering over a visualization type icon will reveal a tooltip that describes the visualization type.

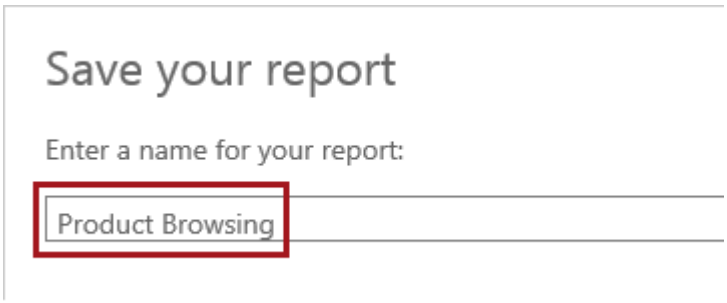


*As no data rows have been added to the dataset, the card presently displays **(Blank)**.*

12. To save the report, at the top-right corner, click **Save**.



13. In the **Save Your Report** dialog window, in the box, enter **Product Browsing**.



14. Click **Save**.

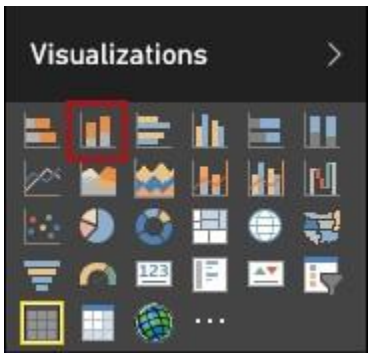
15. To add the visual as a dashboard tile, click the **Pin Visual** icon.

16. In the **Pin to Dashboard** dialog window, select the **New Dashboard** option.

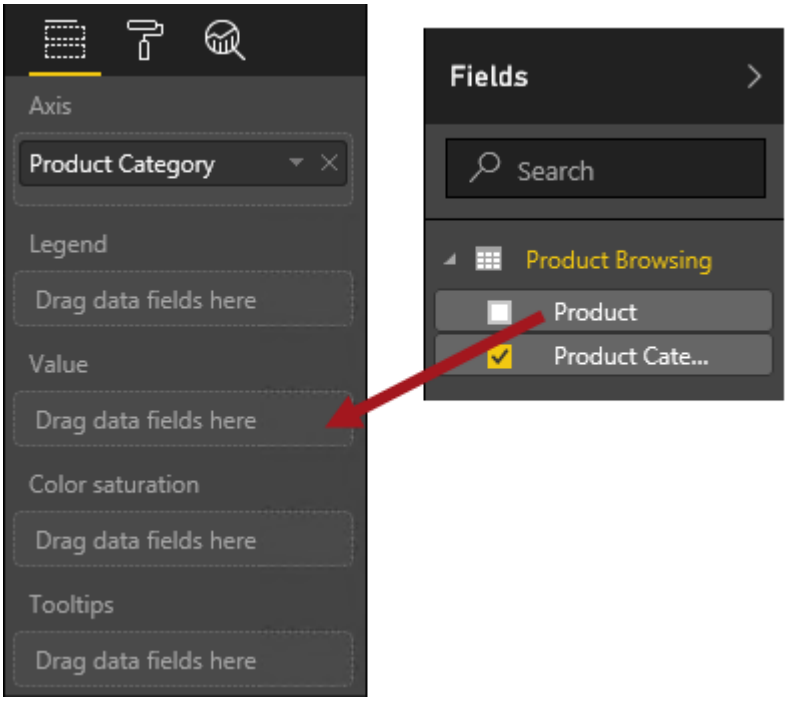
17. in the box, enter **Product Browsing**.

18. Click **Pin**.

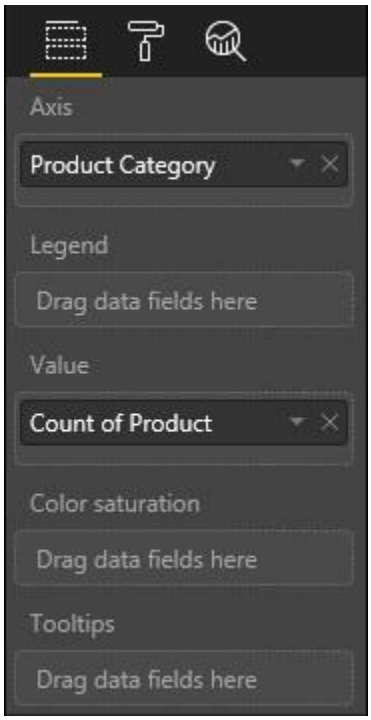
19. To create a second visual, from inside the **Fields** pane, drag the **Product Category** field and drop it at the top-right of the report canvas.
20. Modify the visualization type to **Column**.



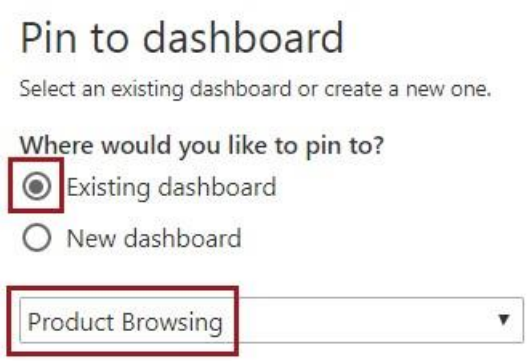
21. Drag the **Product** field and drop it into the **Value** well.



22. In the **Visualizations** pane, notice that the visual is now configured to display the count of products by product category.

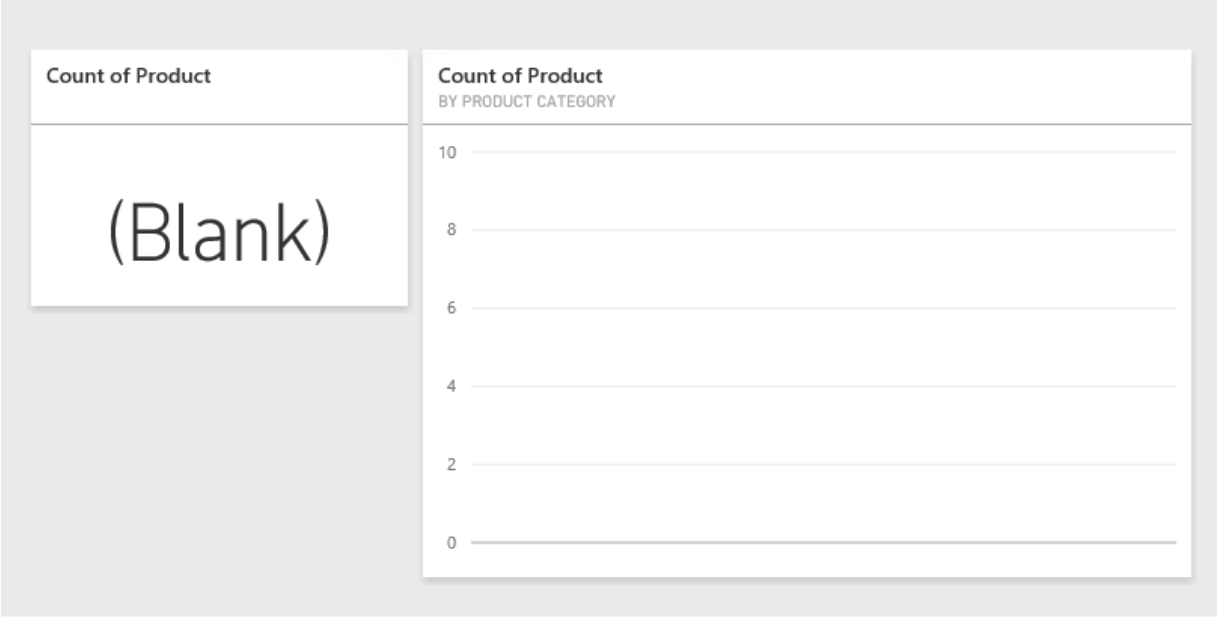


23. Save the report.
24. Pin the column visual to the existing **Product Browsing** dashboard.



25. To open to the dashboard, in the **Navigation Pane**, select the **Product Browsing** dashboard.
26. To view the dashboard in full screen mode, click **Enter Full Screen Mode**.

27. Verify that the dashboard layout looks like the following.



Pushing Data to the Dataset

In this task, you will use the C# application to push a single row to the dataset.

1. Use **Alt+Tab** to return to the **Power BI API Explorer** application.
2. If necessary, position the application window so that the two dashboard tiles are fully visible in the background.

3. In the left pane, right-click the **Product Browsing** table, and then select **Push Data | JSON Document**.

4. In the **Push Data – JSON Document** window, inside the **New Table Row(s)** box, enter the following JSON document.

   *For convenience, the JSON document can be copied from the **<CourseFolder>\PowerBIDevIAD\Lab04A\Snippets.txt** file.*

   *Use **Control-V** to paste the text into the box.*

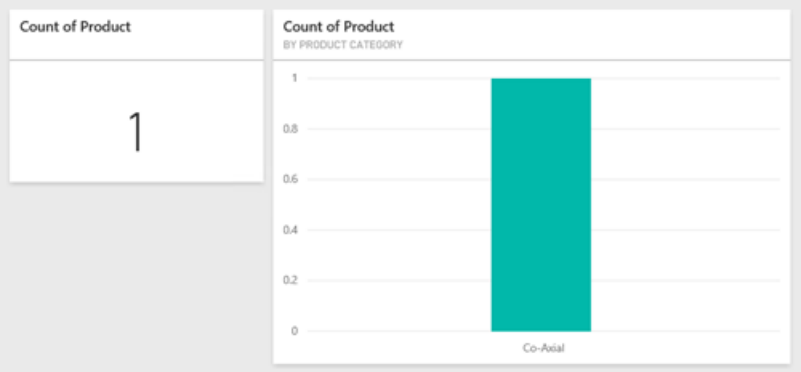
   JSON

```
{
 "rows": [
          {
                 "Product": "4CAX-B Helicopter",
                 "Product Category": "Co-Axial"
          }
 ]
}
```

   The JSON document describes a single row.

5. Click **OK**.

6. In the dashboard, notice that the two tiles immediately update to display the inserted row.



7. Review the console output reporting the HTTP action and URI, and in particular the use of the dataset **id** property.

8. Notice also that the JSON document was added to the request body.

9. In the **Power BI API Explorer** application, to clear all table rows, in the left pane, right-click the **Product Browsing** table, and then select **Clear All Rows**.

10. In the dashboard, notice that the two tiles clear.

11. Review the console output reporting the HTTP action and URI, and in particular the use of the dataset **id** property.

Updating the Table Schema

In this task, as a developer, you will update the **Product Browsing** table schema to include a new column.

1. To update the table schema, in the left pane, right-click the **Product Browsing** table, and then select **Update Schema**.

2. In the **Update Table Schema** window, inside the **New Table Schema** box, enter the following JSON document.

   *For convenience, the JSON document can be copied from the **<CourseFolder>\PowerBIDevIAD\Lab04A\Assets\Snippets.txt** file.*

   *Use **Control-V** to paste the text into the box.*

   JSON

```
{
 "name": "Product Browsing",
 "columns": [
         {
                "name": "Product",
                "dataType": "String"
         },
         {
                "name": "Product Category",
                "dataType": "String"
         },
         {
                "name": "Country",
                "dataType": "String"
         }
 ]
}
```

   *The JSON document describes an updated table schema extending the current table definition with a third column named **Country**.*

   Updating a table that already stores data will preserve existing data when the column name and data type remain unchanged.

3. Click **OK**.

4. Review the console output reporting the HTTP action and URI, and in particular the use of the dataset **id** property.

5. Notice also that the JSON document was added to the request body.

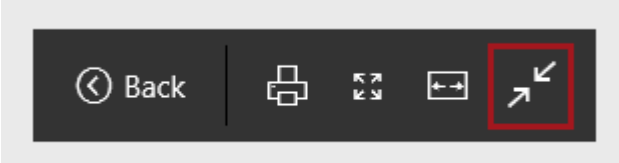
6. Leave the application running.

Adding a Map

In this task, as an analyst, you will add a map to the dashboard.

1. Return to the Power BI Portal web browser.

2. Move the cursor toward the bottom-right corner, and when the pane appears, click **Exit Full Screen Mode**.



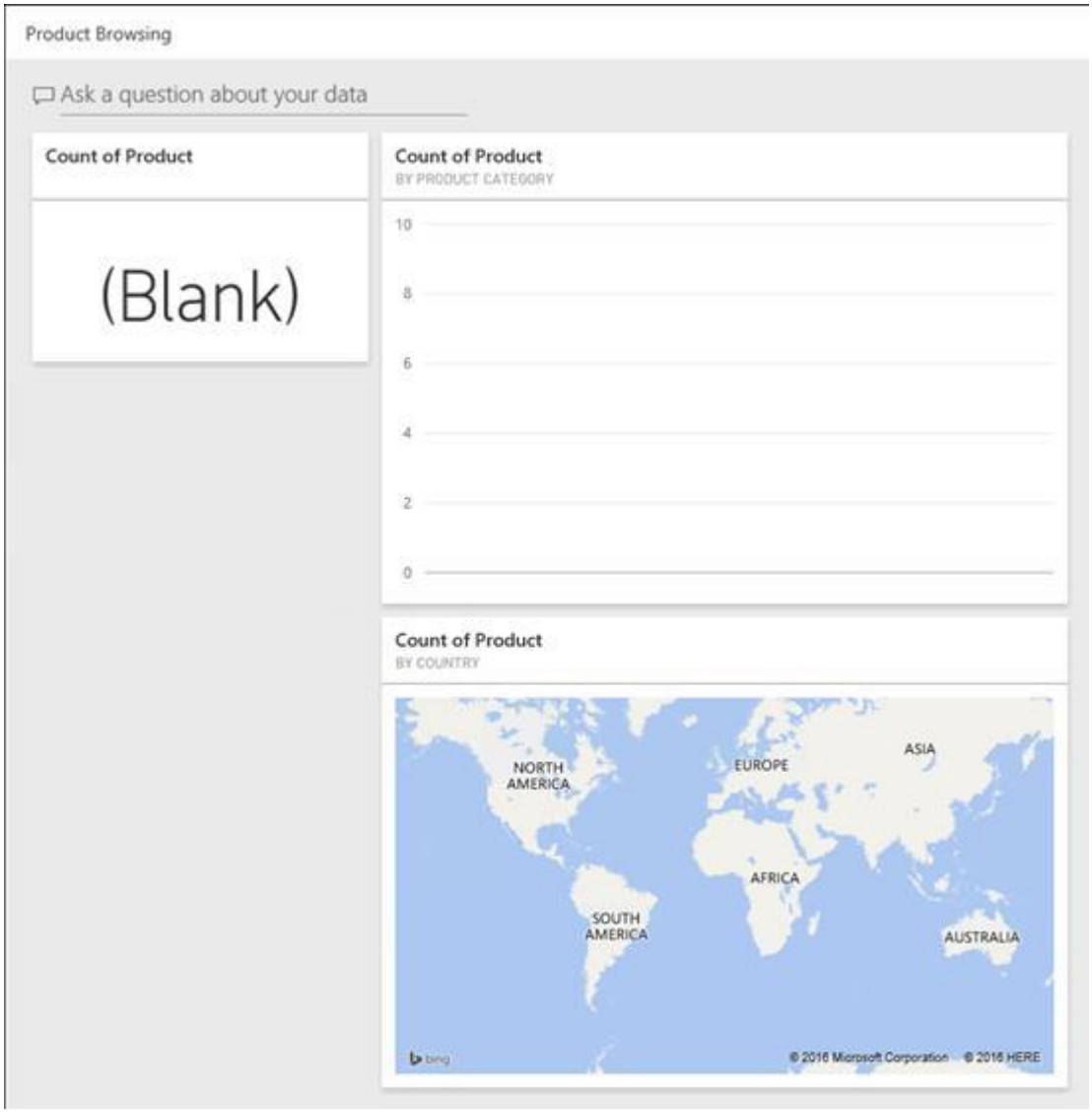
3. In the **Navigation Pane**, click the **Product Browsing** report.
4. On the toolbar, click **Edit Report**.

5. From the **Fields Pane**, notice the addition of the **Country** field.

   *Sometimes there is a delay when updating a table schema. If the **Country** field does not appear, reload the browser window. If it still does not appear, wait some time, and reload the browser (F5) and edit the report again.*

6. Drag the **Country** field to a blank area of the report canvas.
7. Drag the **Product** field and drop it into the **Size** drop zone.
8. Save the report.
9. Pin the map to the dashboard.
10. Navigate to the **Product Browsing** dashboard.

11. Drag the map tile beneath the bar chart tile.



12. View the dashboard in full screen mode.

Pushing an Event Stream to the Dataset

In this task, as a developer, you will push a stream of events to the dataset.

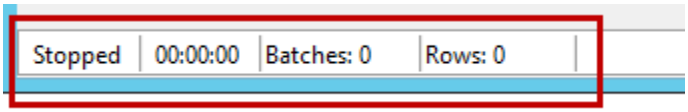
1. Return to the **Power BI API Explorer** application.
2. In the left pane, right-click the **Product Browsing** table, and then select **Push Data | Event Stream**.
3. If necessary, position the **Push Data – Event Stream** window so that all dashboard tiles are fully visible in the background.
4. In the **Event Source** dropdown list, select **[dbo].[uspECommerce_ProductBrowsing]**.

   The selection of a stored procedure will cache the data in preparation for sending batches of rows to the Power BI service. This experience is designed to simulate an event stream.

The application derives the JSON document from the query result retrieved by the stored procedure. The JSON document is produced from the column names and data types, in addition to the data rows.

5. Notice the two properties which determine the batch size (rows per request), and the delay interval between requests. Do not change their values yet.

Tip: You can monitor the streaming progress by reviewing the statistics in the window status bar.



6. Click **Start**.

   There are limits to the amount of data that can be streamed to the Power BI Service. It is one million rows/hour.

7. Observe the dashboard tiles updating.
8. After some time, in the **Push Data – Event Stream** window, click **Stop**.
9. Optionally, modify the batch size and delay interval properties to experiment achieving high volume, low latency throughput.

   The application is configured to limit a batch size to no more than 100,000 rows. Take care not to exceed the limits for your Power BI plan (one million rows/hour for the Power BI Pro license).

10. In the **Push Data – Event Stream** window, click **Close**.
11. In the left pane, right-click the **Power BI Service**, and then select **Disconnect**.
12. Review the console output reporting that the access token has been removed from the cache.

    This operation did not involve a Power BI REST API call.

Summary

In this lab, you used a pre-developed application to explore each of the Power BI REST API methods, and to produce a real-time dashboard.