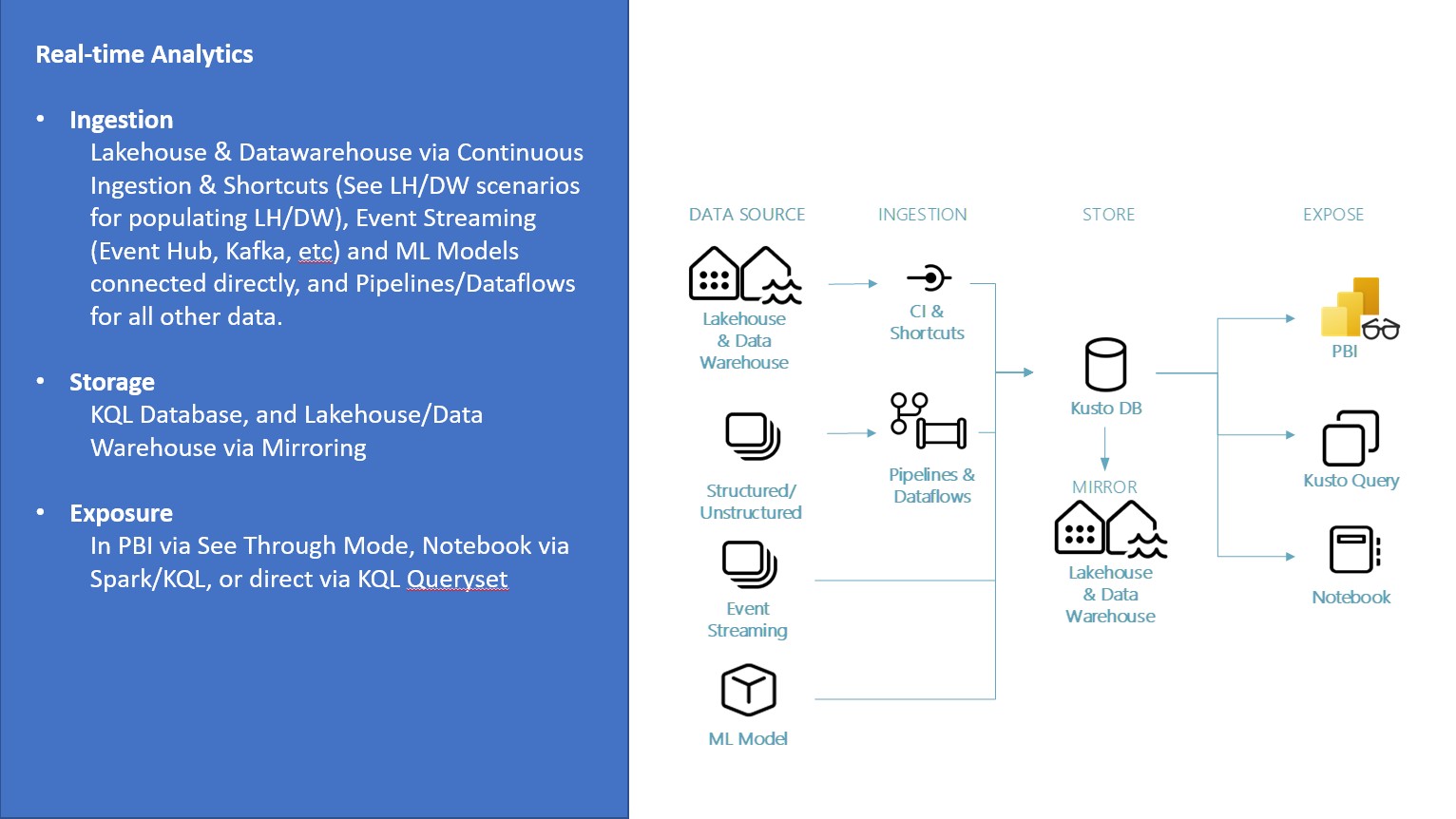
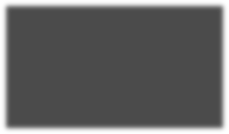
End-to-End Scenario Tutorial

Real-time Analytics

Published: July 2023

Updated : Feb 2024



Contents

[Providing feedback 3](#_bookmark0)

[Introduction 5](#_bookmark1)

[Real-time Analytics 5](#_bookmark2)

[Scenario 6](#_bookmark3)

[Prerequisites 7](#_bookmark4)

[Module 1: Create a Fabric workspace 7](#_bookmark5)

[Module 2: Build your first Real-time Analytics solution in Fabric 9](#_bookmark6)

[Create a KQL Database 9](#_bookmark7)

[Create an Eventstream 11](#_bookmark8)

[Stream data from Eventstream to KQL Database 13](#_bookmark9)

[Explore data and build Power BI report 20](#_bookmark10)

[Module 3: Extending the real-time analytics solution 22](#_bookmark11)

[Get dimension data from Blob Storage 22](#_bookmark12)

[Query data 26](#_bookmark13)

[Explore data further in the KQL Queryset 27](#_bookmark14)

[Build Power BI report 30](#_bookmark15)

[Create OneLake shortcut 33](#_bookmark16)

[Module 4: Clean up resources 36](#_bookmark17)

# Introduction

##### What is Fabric?

Fabric provides a one-stop shop for all the analytical needs for every enterprise. It covers the complete spectrum of services including data movement, data lake, data engineering, data integration and data science, real time analytics, and business intelligence. With Fabric, there is no need to stitch together different services from multiple vendors. Instead, the customer enjoys an end-to-end, highly integrated, single comprehensive product that is easy to understand, onboard, create and operate. There is no other product on the market that offers the breadth, depth, and level of integration that Fabric offers. Additionally, Microsoft Purview is included by default in every tenant to meet compliance and governance needs.

To get an overview over the components and concepts of Fabric read *Fabric - Overview and Concepts.*

##### The purpose of this tutorial

While many concepts in Fabric may be familiar to data and analytics professionals it can be challenging to apply those concepts in a new environment. This tutorial has been designed to walk step-by-step through an end-to-end scenario from data acquisition to data consumption to build a basic understanding of the Fabric UX, the various workloads and their integration points, and the Fabric professional and citizen developer experiences.

The tutorials are not intended to be a reference architecture, an exhaustive list of features and functionality, or a recommendation of specific best practices*.*

# Real-time Analytics

Real-time Analytics is a portfolio of capabilities that provides an end-to-end analytics streaming solution across Fabric experiences. It supplies high velocity, low latency data analysis, and is optimized for time-series data, including automatic partitioning and indexing of any data format and structure, such as structured data, semi-structured (JSON), and free text.

Real-time Analytics delivers high performance when it comes to your increasing volume of data. It accommodates datasets as small as a few gigabytes or as large as several petabytes and allows you to explore data from different sources and a variety of data formats.



DATA SOURCE

INGESTION

STORE

EXPOSE

Lakehouse & Data Warehouse

CI & Shortcuts

PBI

Kusto DB

Structured/ Unstructured

Pipelines & Dataflows

MIRROR

Kusto Query

Lakehouse

Event Streaming

& Data

W e

arehous

Notebook

ML Model

*Figure 1: End-to-end Scenario - Real-time Analytics*

Real-time Analytics includes integration with other Fabric experiences such as Lakehouse, Data Warehouse, Pipelines, Dataflows and Event Streaming on data sources and ingestion side. On the data exposition and consumption, Real-time analytics integrates with Power BI, and Notebooks.

You can use Real-time Analytics for a range of solutions, such as IoT analytics and log analytics, and in a number of scenarios including manufacturing operations, oil and gas, automotive, and more.

In this tutorial, you learn how to:

* Create a KQL Database
* Create Eventstream
* Stream data from Eventstream to KQL Database
* Check your data with sample queries
* Save queries as a KQL Queryset
* Create a Power BI report
* Create a OneLake shortcut

### Scenario

This tutorial is based on a *sample on New York Yellow Taxi trip data*. The dataset contains trip records of New York’s yellow taxis. The yellow taxi trip records include fields capturing pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts. You’ll use the streaming and query capabilities of Real-time Analytics to answer key questions about the trip statistics, taxi demand in the boroughs of New York and related insights.

### Prerequisites

* Power BI Premium subscription. For more information, see [*How to purchase Power BI Premium*](https://learn.microsoft.com/en-us/power-bi/enterprise/service-admin-premium-purchase).
* Workspace

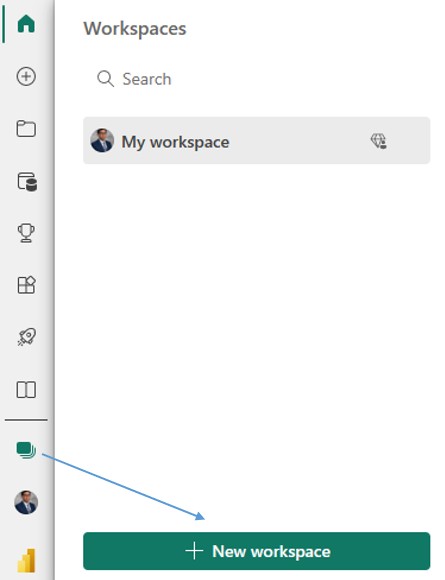
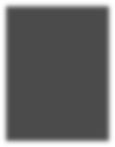
# Module 1: Create a Fabric workspace

Before you can begin building the real-time analytics solution, you will need to enable Project Fabric on your Power BI tenant and create a workspace where you will build out the remainder of the tutorial. In this module, you will learn to:

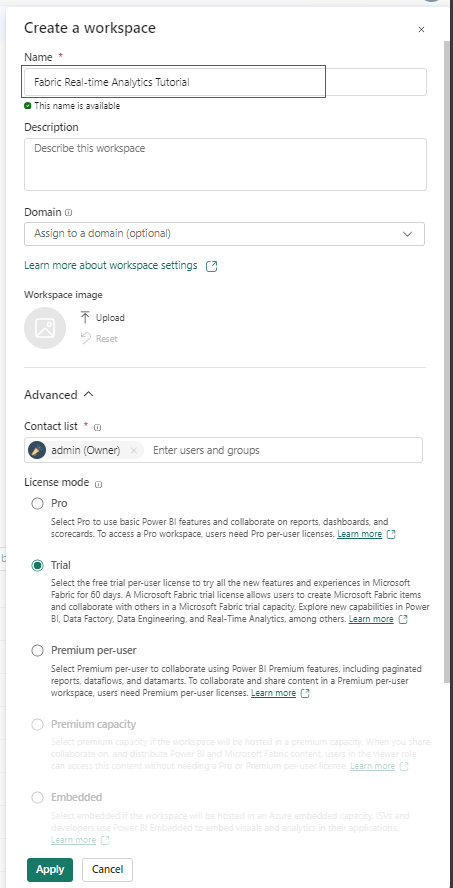
* + Create a Fabric workspace

In this step, you create a Fabric workspace in the Power BI service. The workspace will contain all the artifacts needed for real-time analytics including KQL Database, Eventstream, Power BI datasets, and reports.

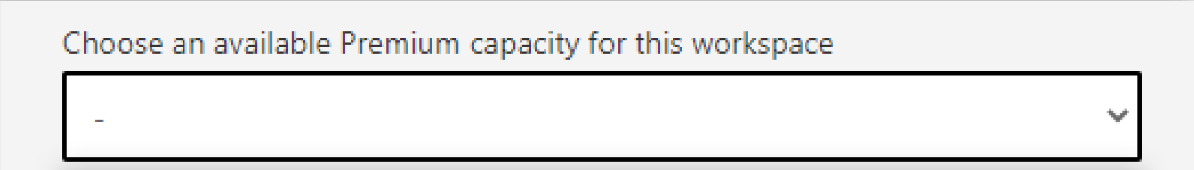
1. Sign in to [Fabric](https://app.fabric.microsoft.com/) .
2. Select **Workspaces > New Workspace**.



1. Fill out the **Create a workspace** form as follows:
   1. **Name:** Enter *Fabric Real-time Analytics Tutorial*, and some characters for uniqueness.
   2. **Description**: Optionally, enter a description for the workspace.



1. Expand the **Advanced** section.
2. Choose **Fabric capacity** in the **License Mode** section.
3. Choose a Fabric capacity you have access to**.**



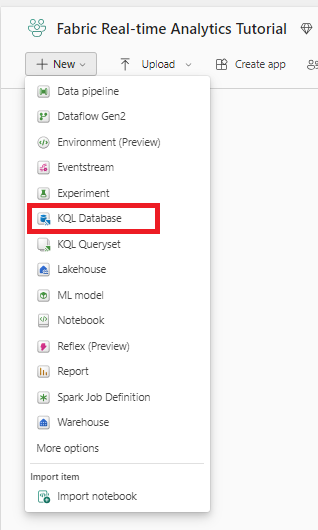
1. Select **Apply.** The workspace will be created and opened.

# Module 2: Build your first Real-time Analytics solution in Fabric

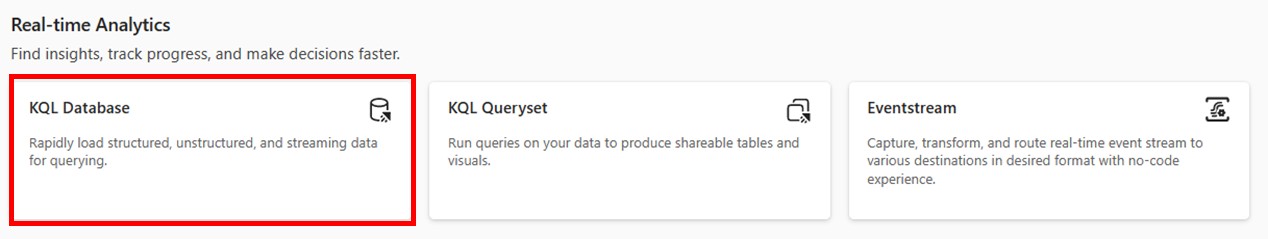
The intent of this module is to quickly build end to end journey of building a real-time analytics solution, ingesting streaming data from Eventstream and then using the KQL Database for creating a real-time refreshing Power BI report.

## Create a KQL Database

1. In the upper left corner of the Fabric Workspace home page, select **New > KQL Database**



1. Or In the **Real-time Analytics** section, select **KQL Database**.



1. On the **New KQL Database** dialog, enter a ***unique name***.
2. Select **Create**.

A screenshot of a computer

Description automatically generated

1. When provisioning is complete the KQL database editor landing page will be shown.

A screenshot of a computer

Description automatically generated

1. Select the **Database** in the Object tree.

A screenshot of a computer

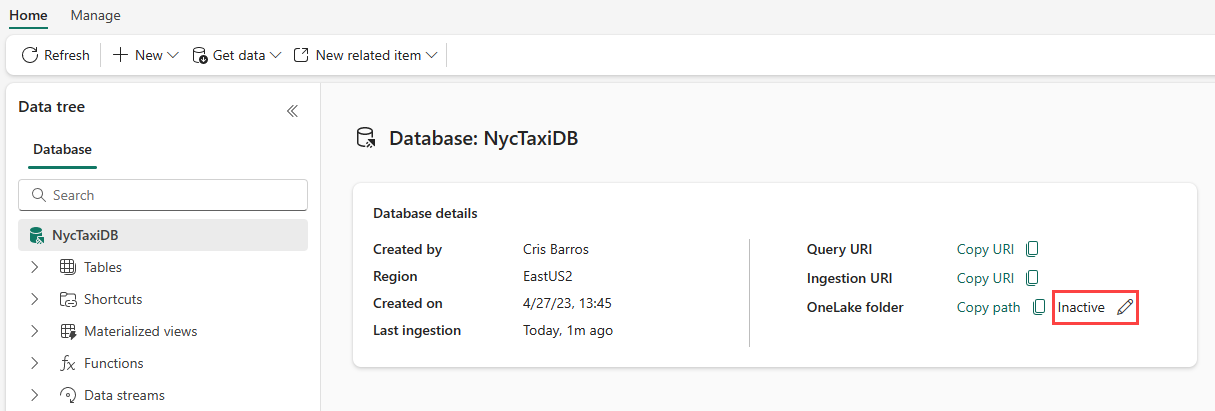
Description automatically generated

1. Select **Explore your data**



1. Enable availability in Onelake (replace **NycTaxiDB** with your database’s name) **.**

In the **Database details** card, select the **pencil** icon



Toggle the button to **Active** and select **Done**

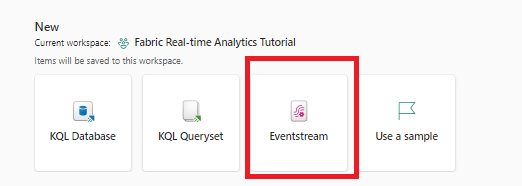
## Create an Eventstream

* 1. Return to the Real-Time Analytics home page. The **Home** icon directs you to the home page of the experience you're currently using.

A black and white logo

Description automatically generated

* 1. In the **Real-time Analytics** section, select **Eventstream**

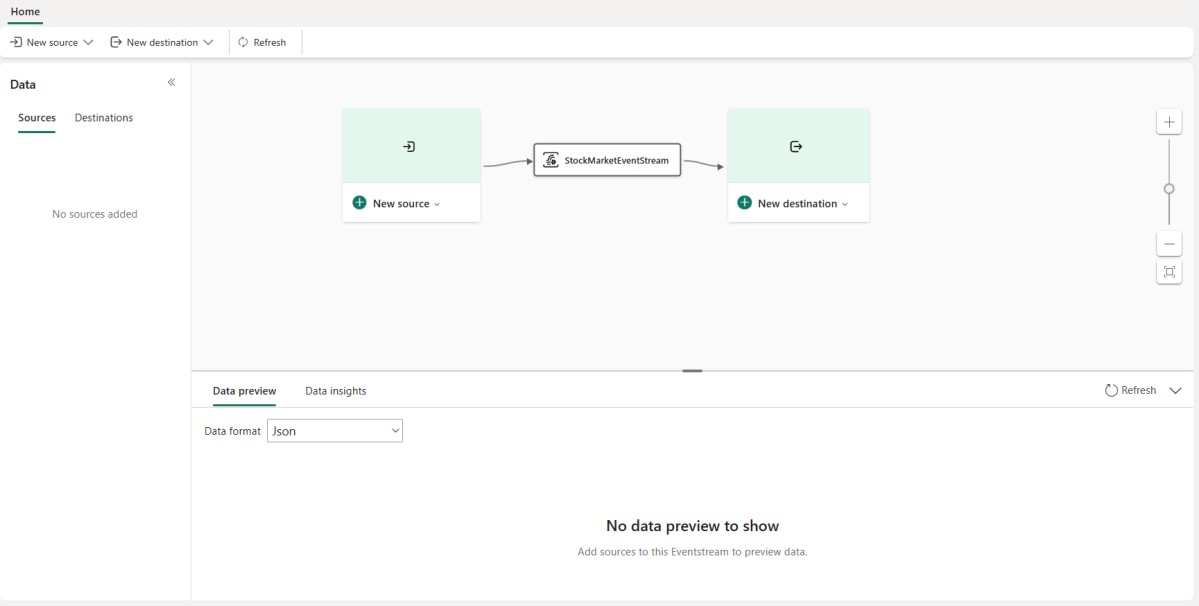
****

* 1. On the New Eventstream dialog, enter NyTaxiTripsEventstream as the name.

A screenshot of a computer

Description automatically generated

* 1. Select **Create**.
  2. When provisioning is complete, Eventstream landing page will be shown.



## Stream data from Eventstream to KQL Database

1. In the Eventstream authoring area, select **New source** and choose **Sample data.**

**A screenshot of a phone

Description automatically generated**

1. Enter **nytaxitripsdatasource** as the Source Name on the right window, choose **Yellow Taxi(high data-rate)** and click on **Add**.

A screenshot of a computer

Description automatically generated

1. In the Eventstream authoring area, select **New destination** and choose **KQL Database.**

**A screenshot of a phone

Description automatically generated**

1. Select Data ingestion Mode as **Direct Ingestion** , Enter **nytaxidatabase** as the destination name, choose your Fabric workspace from the Workspace dropdown and then choose your KQL Database that you created above and Click on **Add and Configure**

A screenshot of a phone

Description automatically generated

1. In the Get Data Screen , select **New table** and enter **nyctaxitrips** as the table name.

A screenshot of a computer

Description automatically generated

1. In the **Inspect** tab, choose **JSON** as the Data format dropdown.

A screenshot of a computer

Description automatically generated

1. After choosing JSON as the Data format, data preview will refresh and show the data in strongly typed columns.

A screenshot of a computer

Description automatically generated

1. In this step, we will change data types of multiple columns.
   1. Select Edit Columns on the top of screen For the column VendorID column name, **Change data type**, and then choose **int.**
   2. choose datatype as **long** for the columns passenger\_count, PULocationID, DOLocationID and payment\_type.
   3. choose datatype as **real** for the following columns: extra, mta\_tax, tolls\_amount, improvement\_surcharge, congestion\_charge, airport\_fee, trip\_distance, fare\_amount, tip\_amount, total\_amount

After changing the datatypes of the above columns , Click on **Apply**

A screenshot of a computer

Description automatically generated

1. Click on Finish

A screenshot of a computer

Description automatically generated

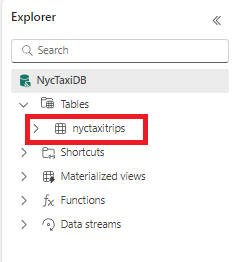
1. In the **Summary** Tab, **Continuous ingestion from Event Stream established** window, all steps will be marked with green check marks when the data connection is successfully created. The data from Eventstream will begin streaming automatically into your table. Click on **Close**

A screenshot of a computer

Description automatically generated

## Explore data and build Power BI report

1. Navigate to your Fabric workspace, select your KQL Database.
2. In the object tree, select the table **nytaxitrips**

****

1. In the top right corner, select **Explore your data**.

A screenshot of a computer

Description automatically generated

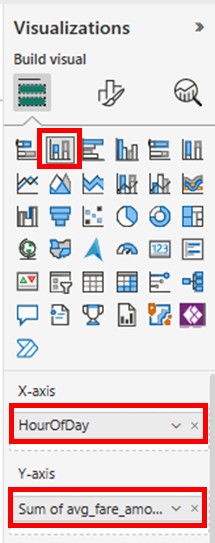
1. Paste the following query in the query editor and select **Run**

#### //Calculate average fare amount by the hour of the day. nyctaxitrips

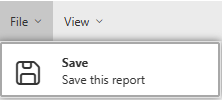
#### | summarize avg(fare\_amount) by HourOfDay = hourofday(tpep\_dropoff\_datetime)

1. Select **Build Power BI Report.** Empty PowerBI report editing window will open.
2. Select **Stacked Column Chart** in the Visualizations pane. Drag **HourOfDay** field to X-axis and

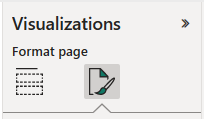
**avg\_fare\_amount** to Y-axis



1. Select **File** menu and then **Save** in the top left corner



1. Enter **nyctaxitripstats** in the **Name your file in Power BI** field, choose your workspace.
2. Once the report is saved, click on the link **Open the file in Power BI to view, edit, and get a shareable link**.
3. Click on **Edit** button to edit the Power BI report.
4. Click on an Empty Space in the Canvas , Choose **Format page**



1. Toggle **Page Refresh** to **On**

A screenshot of a computer

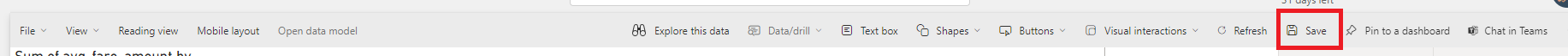
Description automatically generated

1. Expand **Page Refresh** and set the refresh interval to 10 seconds. Please note: Refresh interval will be limited by the Admin interval. Refresh interval can only be greater than or equal to the Admin interval.

A screenshot of a computer

Description automatically generated

1. Save your report.



With this tutorial, you have now built an auto-refreshing Power BI report that is querying streaming data arriving in KQL Database from Eventstream.

# Module 3: Extending the real-time analytics solution

## Get dimension data from Azure Storage

In this module, you are going to ingest Location available in a Azure storage. This data contains additional information on the pick-up locations and drop-off locations used in the trips dataset. Real-time analytics reads and ingests data directly from the blob storage without requiring any other intermediary service.

1. Navigate to KQL Database **NycTaxiDB**
2. From within the KQL Database, select **Get Data** > **Local File**.

A screenshot of a computer

Description automatically generated

##### Destination tab

In the **Destination** tab, **Database** is auto populated with the name of the selected KQL database.

1. Under **Table**, make sure that **New table** is selected, and enter ***locations*** as your table name.

A screenshot of a computer

Description automatically generated

* + Instructor: Please provide the file in advance available under the data folder of the labFile : **ny-yellow-taxi-location-info.csv** click on **+** icon and click **Next :**

1. In the **Inspect tab**, click on **Finish**

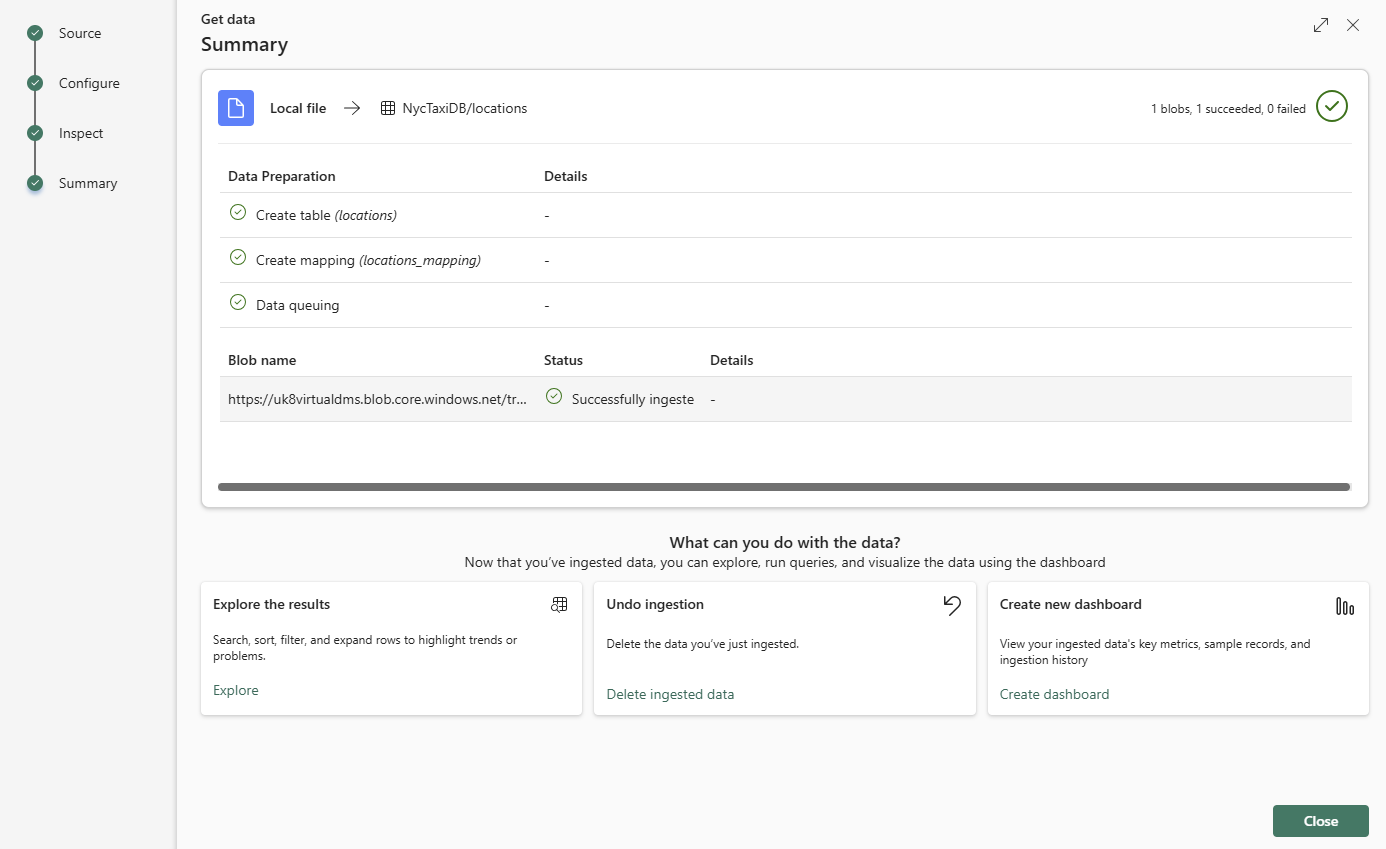
A screenshot of a computer

Description automatically generated

1. Summary tab

In the **Data ingestion is in progress** window, all steps will be marked with green check marks when the data has been

successfully ingested. The data from Azure Storage will begin streaming automatically into your table.



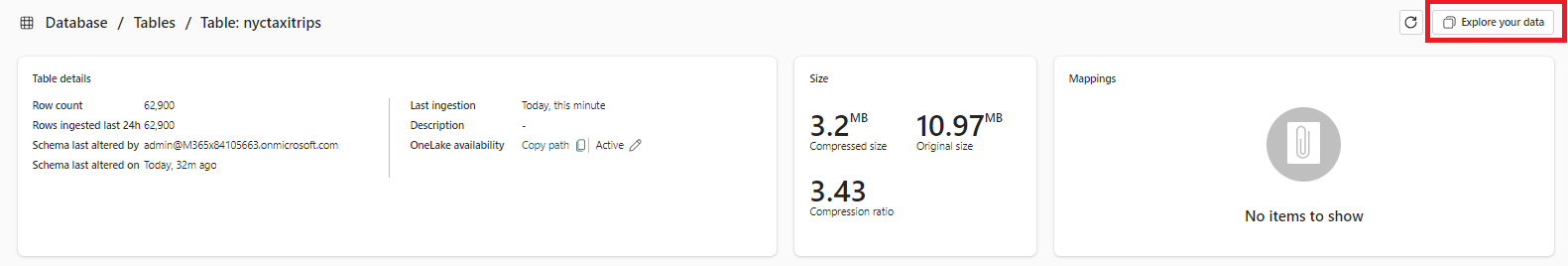
Now that you’ve got data in your database, click on **Close**. you’re going to check your data with sample

queries.

## Query data

In the following step, you’ll use the advanced data analysis capabilities of Kusto Query language to query your telemetry data.

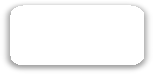
1. Select **Explore your data** on the right-hand side of your database-editor.



Note: The numbers in the screen capture above may look different in your database editor page.

1. Let’s take a look at the data itself. Paste the following query in **Explore your data** window to take 10 random records from your data.

*nyctaxitrips*



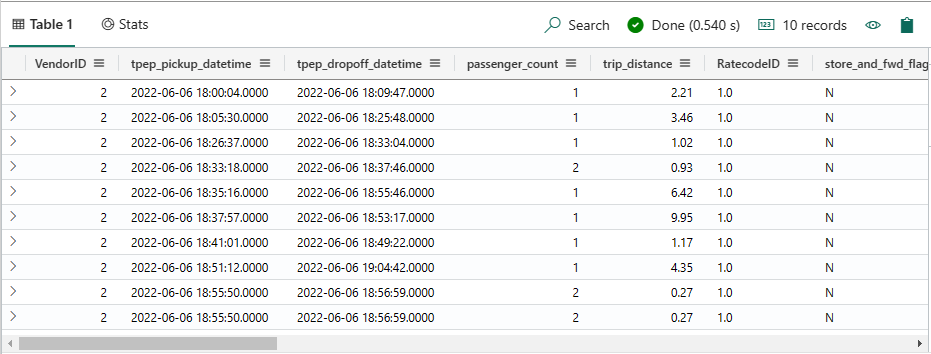
**Query to be pasted** 

*| take 10*

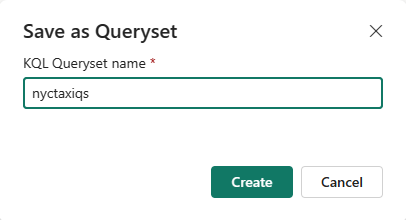
*A screenshot of a computer error

Description automatically generated*

1. Select **Run.**



1. Select **Save as KQL Query Set** to save this and future queries for later use.
2. Under **KQL Queryset name**, enter *nyctaxiqs*, then select **Create**.



**‘Explore your data’** enables you to run some quick queries to understand your data. This query can be saved as a KQL Queryset and persisted in the workspace as an item. Query set autosaves the queries as you type them and lets you resume from the point where you had stopped. In the next module, you will work with the KQL Queryset. Saving the quick query as KQL Query Set will automatically open your **KQL Queryset** with the queries that you wrote in the query editor.

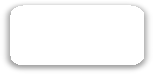
## Explore data further in the KQL Queryset

In this module, you are going to write queries using [*Kusto Query Language*](https://learn.microsoft.com/en-us/azure/data-explorer/kusto/query/)to explore the data that you have ingested from the Event hub and blob storage. Kusto Query Language is a powerful tool to explore your data and discover patterns, identify anomalies and outliers, create statistical modeling, and more. The query uses schema entities that are organized in a hierarchy similar to SQLs: databases, tables, and columns. Kusto query is a read-only request to process data and return results. The request is stated in plain text, using a data-flow model that is easy to read, author, and automate. Kusto queries are made of one or more query statements. You are going to write some simple Kusto queries to get familiar with the language and discover its power and simplicity.

Run the following queries in the new KQL Queryset you have created. Copy/paste each query into your environment, select the query and then select **Run**.

1. The following query returns the top 10 pickup locations in New York City for Yellow Taxis.

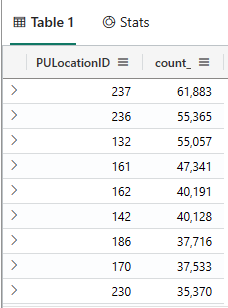
*//Top 10 pickup locations nyctaxitrips*



**Query to be pasted** 

*| summarize count() by PULocationID*

*| top 10 by count\_*

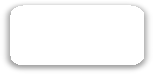


**Note**: Result of the query may not exactly match the screenshot provided as you are ingesting streaming data.

1. We will run the same query as in the previous step with an addition of looking up the corresponding zones of

the top 10 pickup locations by using the ‘locations’ table.

*//For the same top 10 locations, lookup the NYC zones --> Top 10 zones nyctaxitrips*



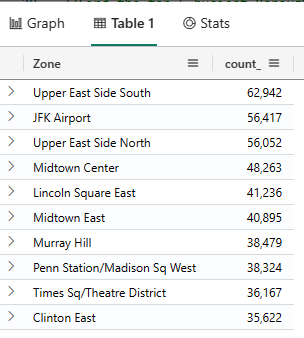
**Query to be pasted** 

*| lookup (locations) on $left.PULocationID == $right.LocationID*

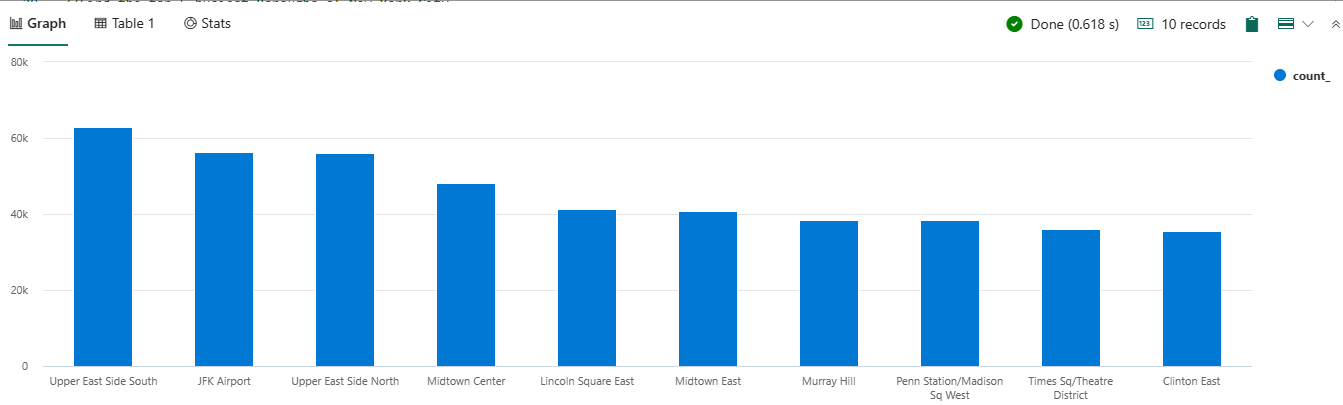
*| summarize count() by Zone*

*| top 10 by count\_*

*| render columnchart*

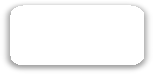


**Note**: Result of the query may not exactly match the screenshot provided as you are ingesting streaming data.



1. Let’s check anomalies in the tips that have been given by the customers in the Manhattan borough. Hover over the red dots to see the values.

*//Find anomalies in the tips given by the customers nyctaxitrips*



**Query to be pasted** 

*| lookup (locations) on $left.PULocationID==$right.LocationID*

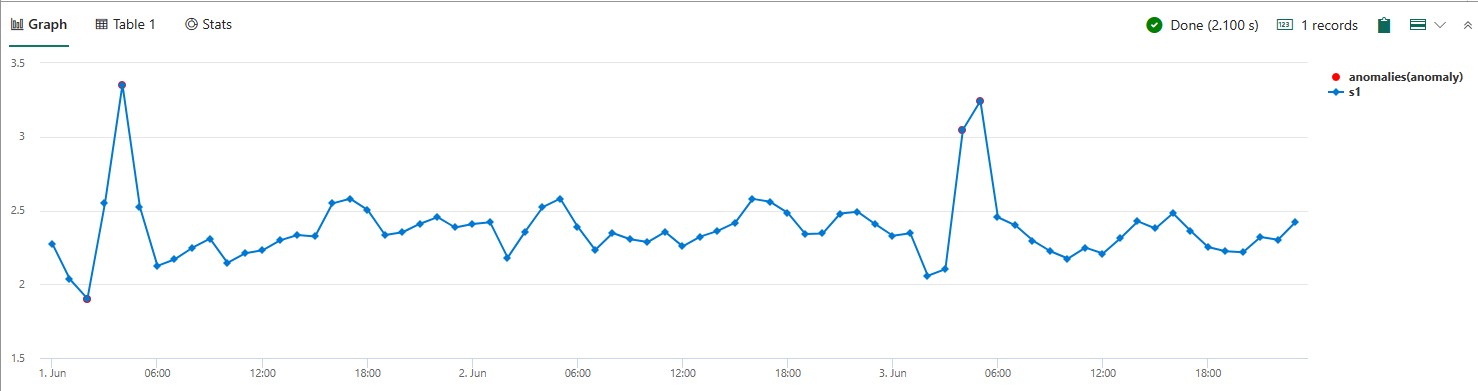
*| where Borough == "Manhattan"*

*| make-series s1 = avg(tip\_amount) on tpep\_pickup\_datetime from datetime(2022-06-01) to datetime(2022-06- 04) step 1h*

*| extend anomalies = series\_decompose\_anomalies(s1)*

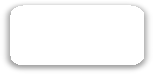
*| render anomalychart with (anomalycolumns=anomalies)*

**Note**: Result of the query may not exactly match the screenshot provided as you are ingesting streaming data.



1. To ensure that the sufficient taxis are plying in the Manhattan borough, let’s forecast the number of taxis needed per hour.

*//Forecast the number of trips that will begin from Manhattan to line up the taxis in that borough nyctaxitrips*



**Query to be pasted** 

*| lookup (locations) on $left.PULocationID==$right.LocationID*

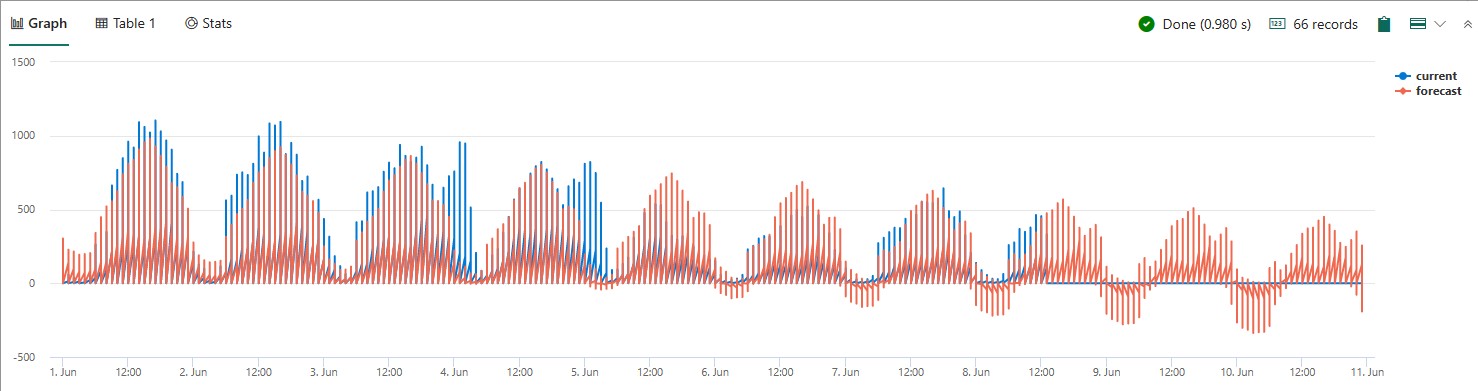
*| where Borough == "Manhattan"*

*| make-series s1 = count() on tpep\_pickup\_datetime from datetime(2022-06-01) to datetime(2022-06-08)+3d ste p 1h by PULocationID*

*| extend forecast = series\_decompose\_forecast(s1, 24\*3)*

*| render timechart*

**Note**: Result of the query may not exactly match the screenshot provided below as you are ingesting streaming data.

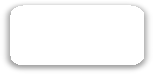


## Build Power BI report

A Power BI report is a multi-perspective view into a dataset, with visuals that represent findings and insights from that dataset.

1. Continuing the in same queryset, paste the following query. The output of this query will be used as the dataset for building the Power BI report.

*//Find the total number of trips that started and ended at the same location nyctaxitrips*



**Query to be**

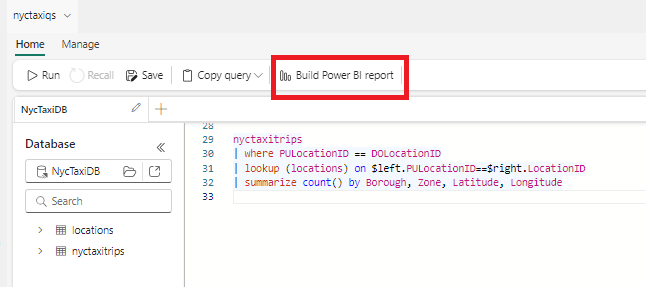
**pasted** 

*| where PULocationID == DOLocationID*

*| lookup (locations) on $left.PULocationID==$right.LocationID*

*| summarize count() by Borough, Zone, Latitude, Longitude*

1. Select the query and then select **Build Power BI report**.



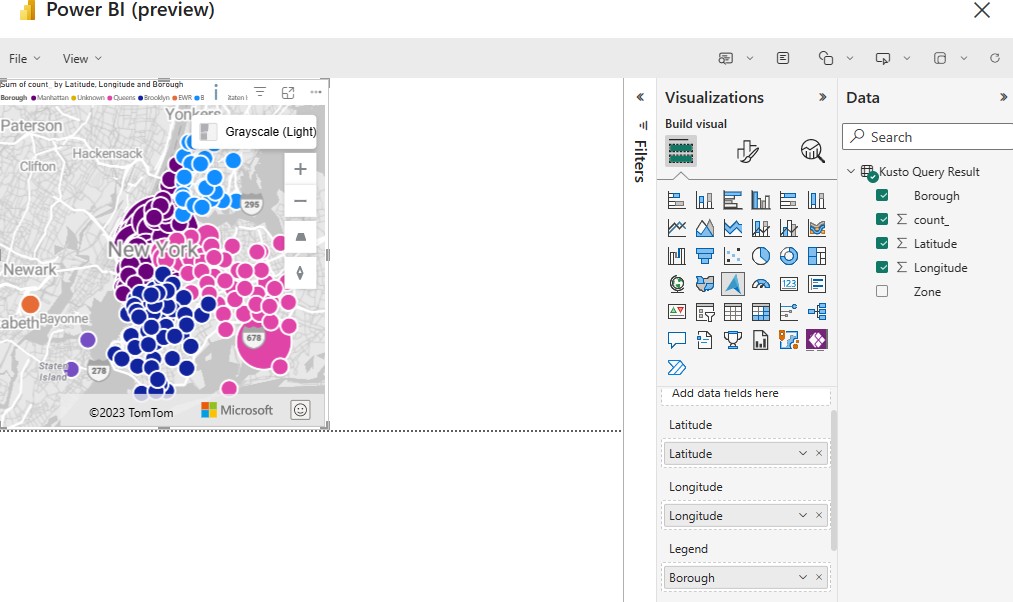
Power BI report editor will open with the result of the query available as a table with the name Kusto Query Result.

Note:

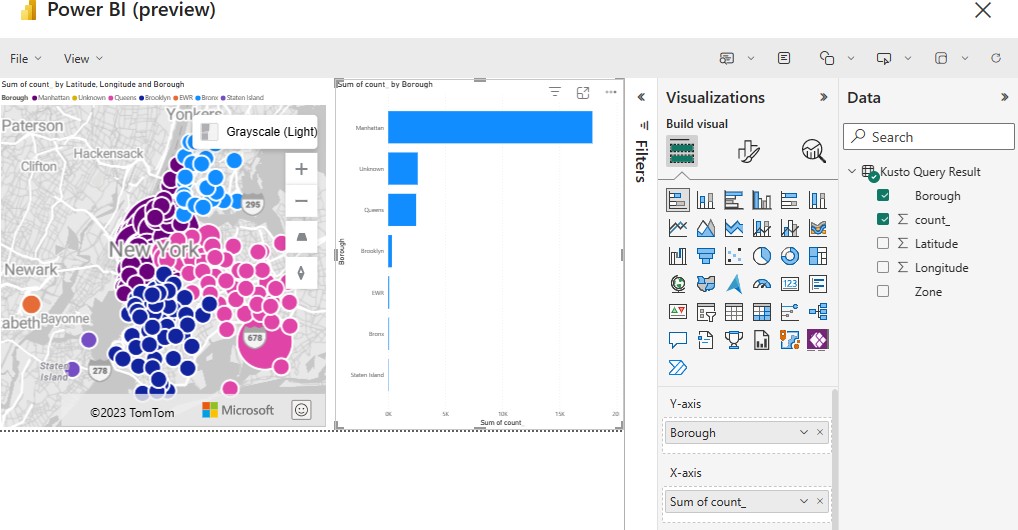
(a) When you build a report, a dataset is created and saved in your workspace. You can create multiple reports from a single dataset. (b) Result of the query may not exactly match the screenshot provided below as you are ingesting streaming data.

If you delete the dataset, your reports will also be removed.

1. In the report editor, choose **Azure Maps** as the visual, drag **Latitude** field to Latitude, **Longitude** field to Longitude, **Borough** field to Legend, and **count\_** field to Size.



1. Add a **Stacked Bar Chart** to the canvas. Drag **Borough** field to Y-Axis and **count\_** to the X-axis

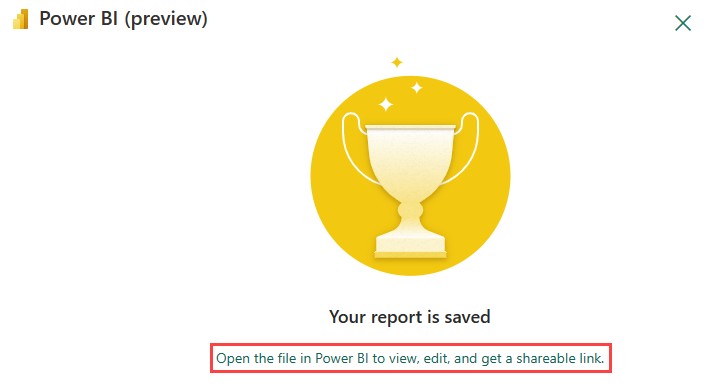


1. Click File > Save
2. Under **Name your file in Power BI**, enter *nyc-taxi-maps-report*.

A screenshot of a computer

Description automatically generated

1. Select the workspace in which you want to save this report. The report can be saved in a different workspace than the one you started in.
2. Select **Open the file in Power BI to view, edit, and get a shareable link** to view and edit your report.



## Create OneLake shortcut

Now that you have finished exploring your data, you may want to access the underlying data from other Fabric experiences.

OneLake is a single, unified, logical data lake for Fabric to store lakehouses, warehouses and other items. Shortcuts are embedded references within OneLake that point to other files’ store locations. The embedded reference makes it appear as though the files and folders are stored locally but in reality; they exist in another storage location. Once you create a shortcut, you can access your data in all of Fabric’s experiences. Shortcuts can be updated or removed from your items, but these changes will not affect the original data and its source.

1. Select **Create** in the **Navigation pane**.

A white rectangular object with black text

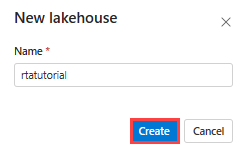
Description automatically generated

1. Under **Data engineering**, select **Lakehouse**.

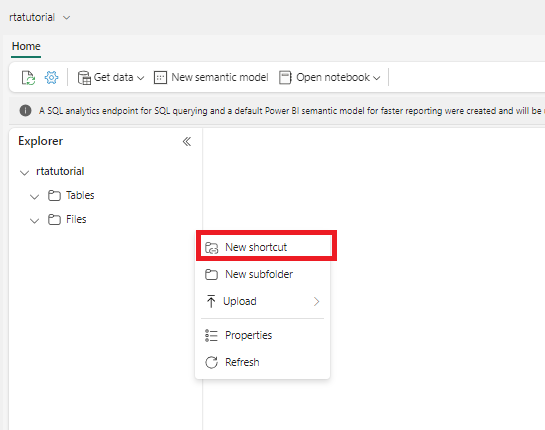
A screenshot of a computer

Description automatically generated

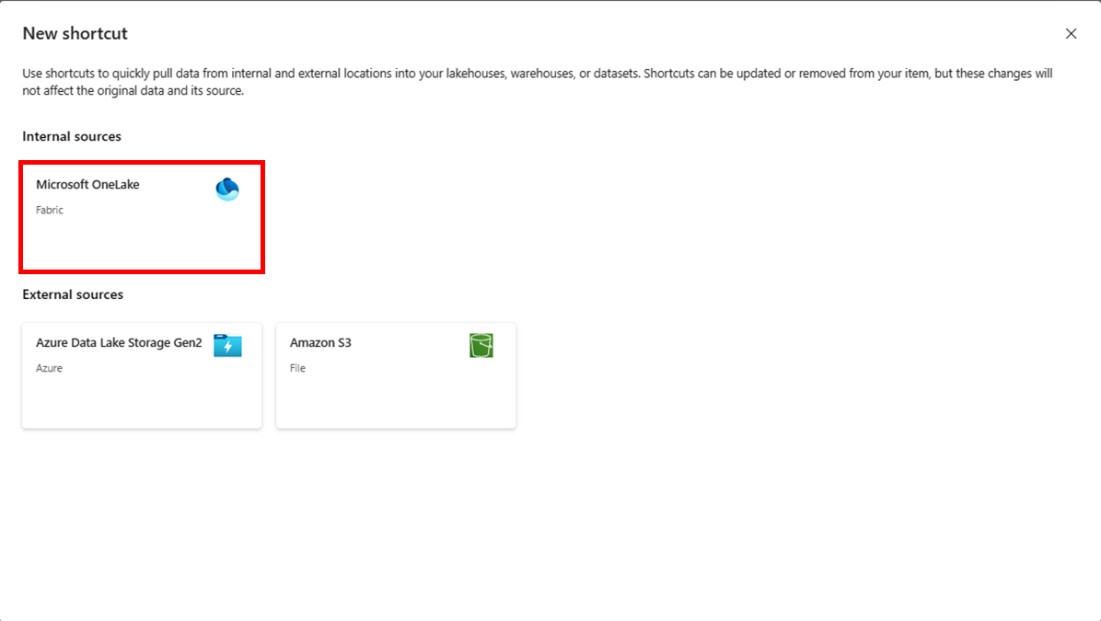
1. Enter *rtatutorial* as your Lakehouse name, then select **Create**.



1. In Lakehouse Explorer, navigate to **Files**, select “**…**”, select **New Shortcut** from the menu.



1. Under **Internal sources**, select **OneLake**.



1. In **Select a data source type**, Filter KQL Database and select ***your KQL Database***, then select **Next** to connect the data to your shortcut.

A screenshot of a computer

Description automatically generated

1. To connect the table with the data from Eventstream, select **>** to expand the tables in the left-hand pane, then select the table titled **nyctaxitrips**.

A screenshot of a computer

Description automatically generated

1. Select **Create** to create the shortcut. The Lakehouse will automatically refresh. Click on Close

A screenshot of a computer

Description automatically generated

The Lakehouse shortcut has been created. Without additional management, you now have one logical copy of your data in the Lakehouse that you can use in other Fabric experiences, such as Notebooks and Spark jobs.

# Module 4: Clean up resources

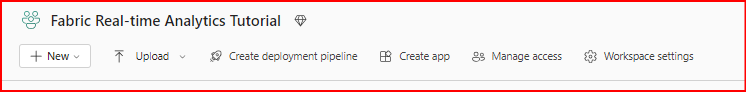
You can delete individual reports, eventstreams, KQL databases, KQL querysets, and other items or remove the entire workspace.

* 1. Select Fabric Real-time Analytics Solution Tutorial in the left-hand navigation menu to return to the workspace artifact view

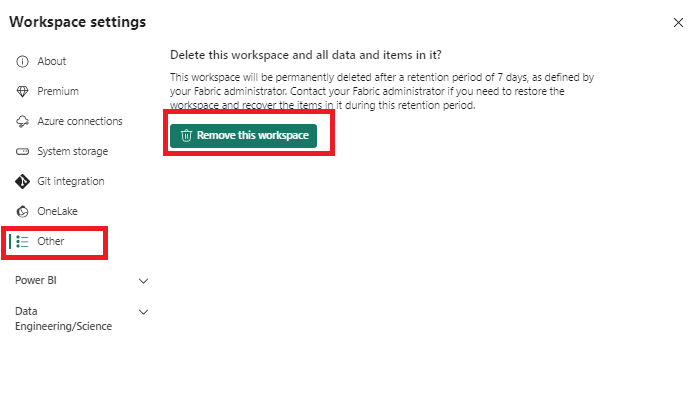
A screen shot of a phone

Description automatically generated

* 1. Below the workspace name and description at the top of the workspace header, select Workspace settings.



##### Select Other > Remove this workspace

****

* 1. Select **Delete** on the warning.