

# Project: Building an Azure Data Warehouse for Bike Share Data Analytics

---



Jun 9 2024

---

Dahi Nemutlu

---

# Contents

1. Project Overview .....	3
2. Create Azure Resources .....	5
2.1. Create Azure Database for PostgreSQL .....	5
2.2. Create Azure Synapse Workspace .....	6
3. Create the data in PostgreSQL.....	6
4. Extract the data from PostgreSQL .....	7
5. Load the data into external tables in the data warehouse .....	10
6. Transform the data to the star schema .....	10
7. Analyze and visualize data with Power BI.....	11

# 1. Project Overview

Divvy is a bike sharing program in Chicago, Illinois USA that allows riders to purchase a pass at a kiosk or use a mobile application to unlock a bike at stations around the city and use the bike for a specified amount of time. The bikes can be returned to the same station or to another station. The City of Chicago makes the anonymized bike trip data publicly available for projects like this where we can analyze the data. The dataset looks like this:

payment	rider	station	trip
PK payment_id	PK rider_id	PK station_id	PK trip_id
date	first	name	rideable_type
amount	last	latitude	start_at
rider_id	address	longitude	ended_at
	birthday		start_station_id
	account_start_date		end_station_id
	account_end_date		rider_id
	is_member		

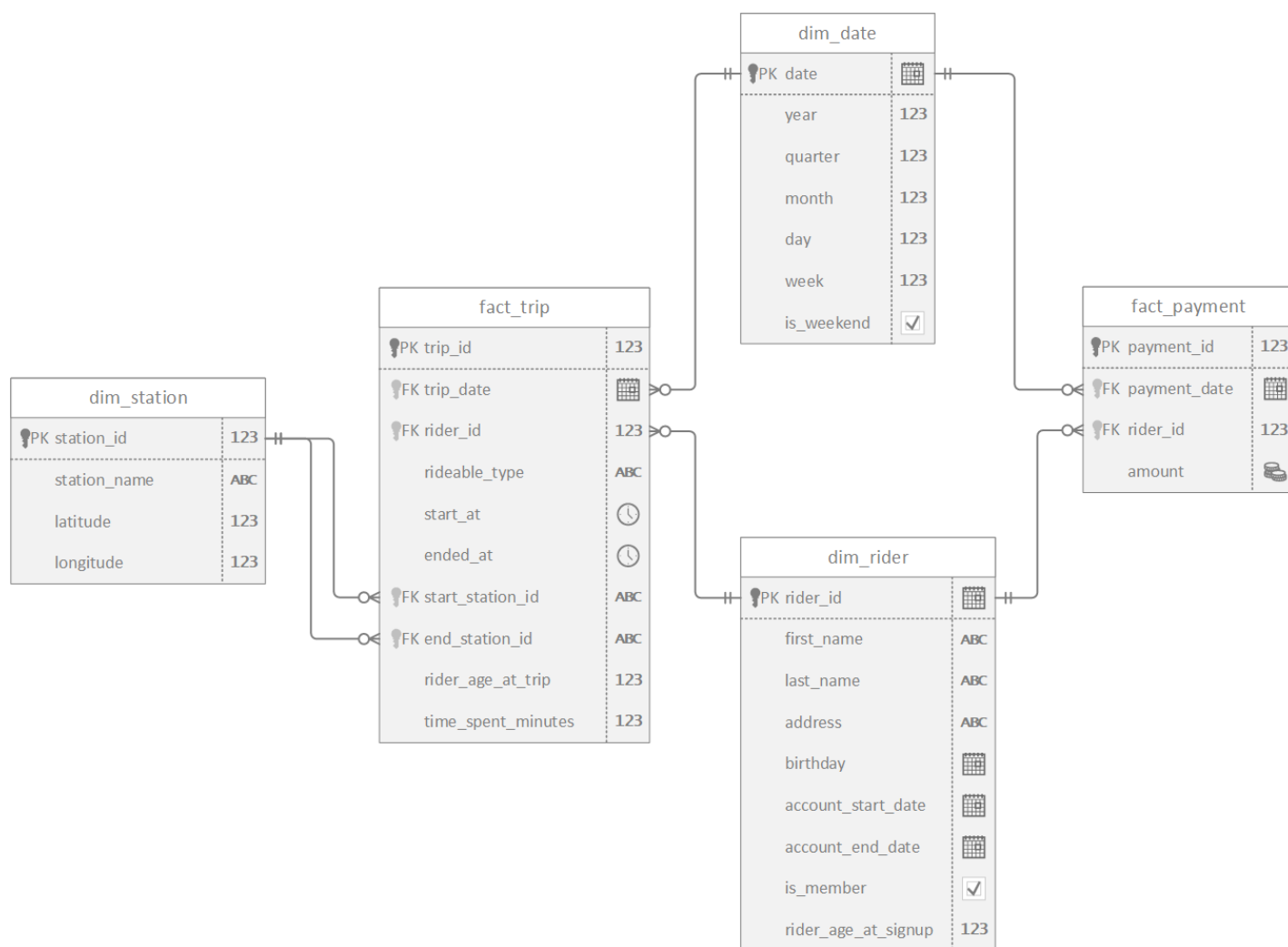
The goal of this project is to develop a data warehouse solution using Azure Synapse Analytics. We will:

- Design a star schema based on the business outcomes listed below;
- Import the data into Synapse;
- Transform the data into the star schema;
- and finally, view the reports using Power BI.

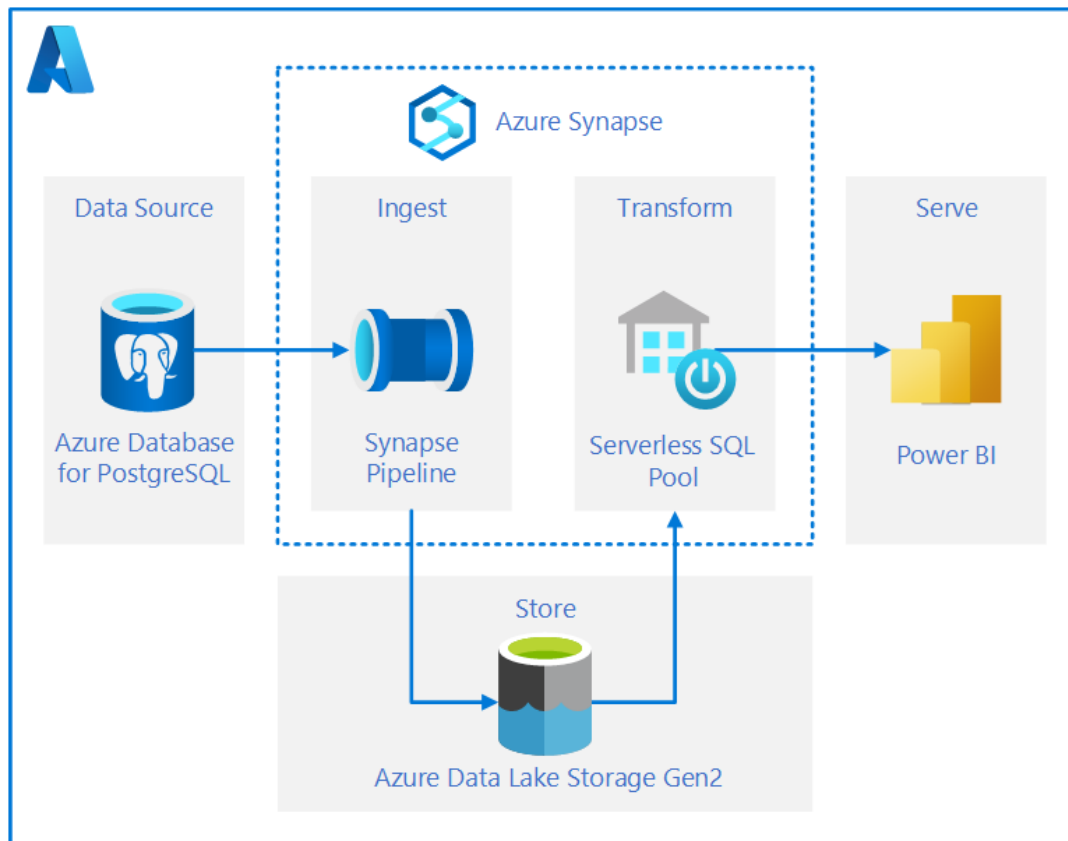
The business outcomes we are designing for are as follows:

1. Analyze how much time is spent per ride
  - Based on date and time factors such as day of week and time of day
  - Based on which station is the starting and / or ending station
  - Based on age of the rider at time of the ride
  - Based on whether the rider is a member or a casual rider
2. Analyze how much money is spent
  - Per month, quarter, year
  - Per member, based on the age of the rider at account start

Based on the provided business requirements for the data warehouse, we will create the following star schema using fact and dimension tables.



With this project, we will create an end-to-end Azure data warehousing solution, with the architecture as follows:



## 2. Create Azure Resources

### 2.1. Create Azure Database for PostgreSQL

Microsoft Azure | Search resources, services, and docs (G+)

Home > PostgreSQLFlexibleServer\_e32c1203f4f24a838a156cd6485a56a5 | Overview >

**divvy-postgresql-server**

Azure Database for PostgreSQL flexible server

Search << Connect Delete Reset password Restore Restart Upgrade Stop >>

**Overview**

- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Migration

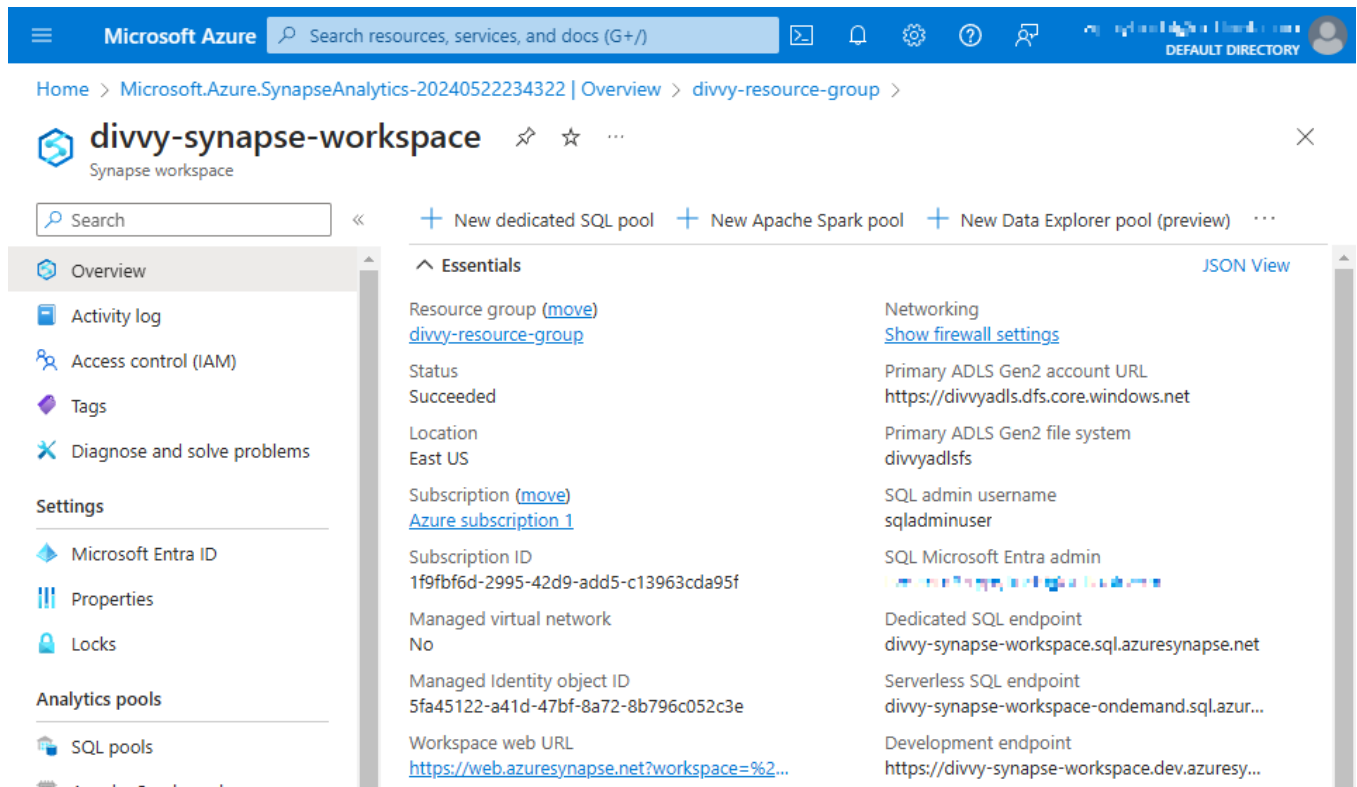
**Settings**

- Compute + storage
- Networking
- Databases
- Connect
- Server parameters

**Essentials** [JSON View](#)

Subscription <a href="#">(move)</a> <a href="#">Azure subscription 1</a>	Server name divvy-postgresql-server.postgres.database.azure...
Subscription ID 1f9bf6d-2995-42d9-add5-c13963cda95f	Server admin login name divvyadmin
Resource group <a href="#">(move)</a> <a href="#">divvy-resource-group</a>	Configuration <a href="#">Burstable, 81ms, 1 vCores, 2 GiB RAM, 32 GiB...</a>
Status Available	PostgreSQL version 16.2
Location East US	Availability zone 1
	High availability Not Enabled
	Created On 2024-05-21 23:34:51.9994482 UTC

## 2.2. Create Azure Synapse Workspace



The screenshot shows the Microsoft Azure portal interface for a Synapse workspace named 'divvy-synapse-workspace'. The top navigation bar includes the Microsoft Azure logo, a search bar, and user profile information. The breadcrumb trail indicates the path: Home > Microsoft.Azure.SynapseAnalytics-20240522234322 | Overview > divvy-resource-group > divvy-synapse-workspace. The left sidebar contains a navigation menu with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (Microsoft Entra ID, Properties, Locks), and Analytics pools (SQL pools, Apache Spark pools). The main content area is titled 'divvy-synapse-workspace' and includes a search bar and buttons to create new pools: 'New dedicated SQL pool', 'New Apache Spark pool', and 'New Data Explorer pool (preview)'. The 'Essentials' section lists key properties: Resource group (divvy-resource-group), Status (Succeeded), Location (East US), Subscription (Azure subscription 1), Subscription ID (1f9fb6d-2995-42d9-add5-c13963cda95f), Managed virtual network (No), Managed Identity object ID (5fa45122-a41d-47bf-8a72-8b796c052c3e), and Workspace web URL (https://web.azure.synapse.net?workspace=%2...). The 'Networking' section lists endpoints: Primary ADLS Gen2 account URL (https://divvyadls.dfs.core.windows.net), Primary ADLS Gen2 file system (divvyadlfs), SQL admin username (sqladminuser), SQL Microsoft Entra admin (divvy-synapse-workspace-admin), Dedicated SQL endpoint (divvy-synapse-workspace.sql.azure.synapse.net), Serverless SQL endpoint (divvy-synapse-workspace-ondemand.sql.azure...), and Development endpoint (https://divvy-synapse-workspace.dev.azure.synapse...).

## 3. Create the data in PostgreSQL

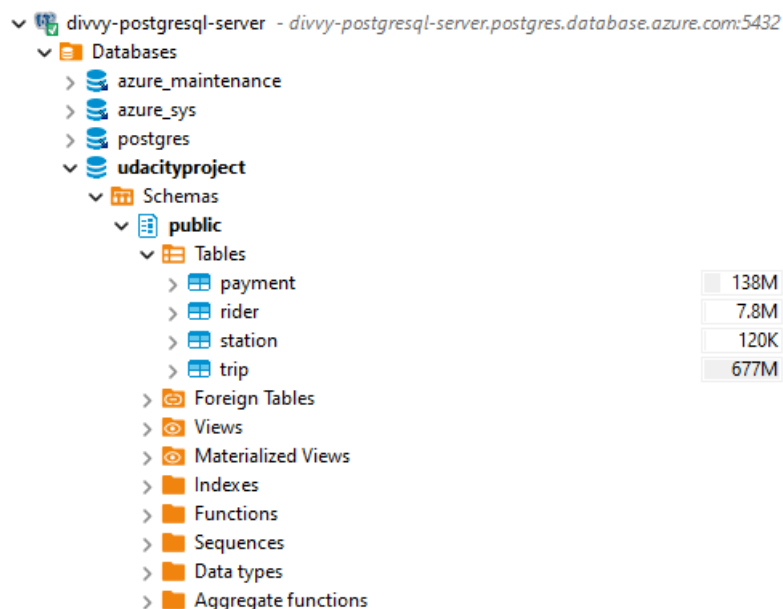
To prepare our environment for this project, we first must create the data in PostgreSQL. This will simulate the production environment where the data is being used in the OLTP system. This can be done using the Python script provided: *ProjectDataToPostgres.py*

**Note:** Add host, username, and password information for your PostgreSQL database to the script before running it.

```
PS C:\Users\dnemu\Desktop\building-an-azure-data-warehouse-for-bike-share-data-analytics> & C:/Users/dnemu/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/dnemu/Desktop/building-an-azure-data-warehouse-for-bike-share-data-analytics/ProjectDataToPostgres.py
Connection established
Connection established
Finished creating table rider
Finished populating rider
Finished creating table payment
Finished populating payment
Finished creating table station
Finished populating station
```

```
Finished creating table trip
Finished populating trip
All done!
```

Then verify that the data exists using pgAdmin or a similar PostgreSQL data tool.



## 4. Extract the data from PostgreSQL

In our Azure Synapse workspace, we will use the ingest wizard to create a one-time pipeline that ingests the data from PostgreSQL into Azure Data Lake Gen 2 storage. This will result in all four tables being represented as text files in the storage, ready for loading into the data warehouse.

## Copy Data tool

- ✓ Properties
- ✓ Source
- ✓ Destination
- ✓ Settings
- 5 Review and finish**
- Review
- Deployment

## Summary

You are running pipeline to copy data from Azure Database for PostgreSQL to Azure Data Lake Storage Gen2.



## Properties

 Edit

Task name	CopyPipeline_fhv
-----------	------------------

### Task description

## Source

 Edit

Connection name	AzurePostgreSql1
-----------------	------------------

Dataset name	SourceDataset_fhv
--------------	-------------------

Number of tables 4

## Destination

 Edit

Connection name divvy-synapse-workspace-WorkspaceDefaultStorage

Dataset name	DestinationDataset_fhv
--------------	------------------------

Column delimiter ,

Escape character	\
------------------	---

Quote char "

```
First row as header      false
```



Synapse Analytics > divvy-synapse-workspace

Analytics pools

- SQL pools
- Apache Spark pools
- Data Explorer pools (preview)

Activities

- SQL requests
- KQL requests
- Apache Spark applications
- Data flow debug

Integration

- Pipeline runs
- Trigger runs
- Integration runtimes
- Link connections

All pipeline runs > CopyPipeline\_fhv - Activity runs

Rerun Cancel Refresh Update pipeline List Gantt

ForEach

ForEach\_fhv

Activities

Copy\_fhv

Activity runs

Pipeline run ID bdfc596-d9a9-4c0e-875a-db248db44522

All status List Monitor in Azure Metrics Export to CSV

Showing 1 - 5 items

Destination	Source	Activity run ID
divvyadlsfs//payment.csv	public.payment	dc53373b-a3e9-43f1-b3f2-488e3fa163bb
divvyadlsfs//trip.csv	public.trip	c6393d8a-738f-41ba-904d-68ddc6be1493
divvyadlsfs//station.csv	public.station	274edddf-6d34-4285-8dd3-74b20a83da70
divvyadlsfs//rider.csv	public.rider	50a7dcf8-5450-43dd-86c7-977b22c865e8
divvyadlsfs//rider.csv	public.rider	84acc9cb-f59c-4784-b8d9-70744c39d4b6

divvy-synapse-workspace

Search

Synapse live Validate all Publish all

Data

Workspace Linked

Filter resources by name

- Azure Data Lake Storage Gen2 2
- divvy-synapse-workspace (Primary ...)
- divvyadlsfs (Primary)
- (Attached Containers)
- Integration datasets 2

divvyadlsfs

New SQL script New notebook More

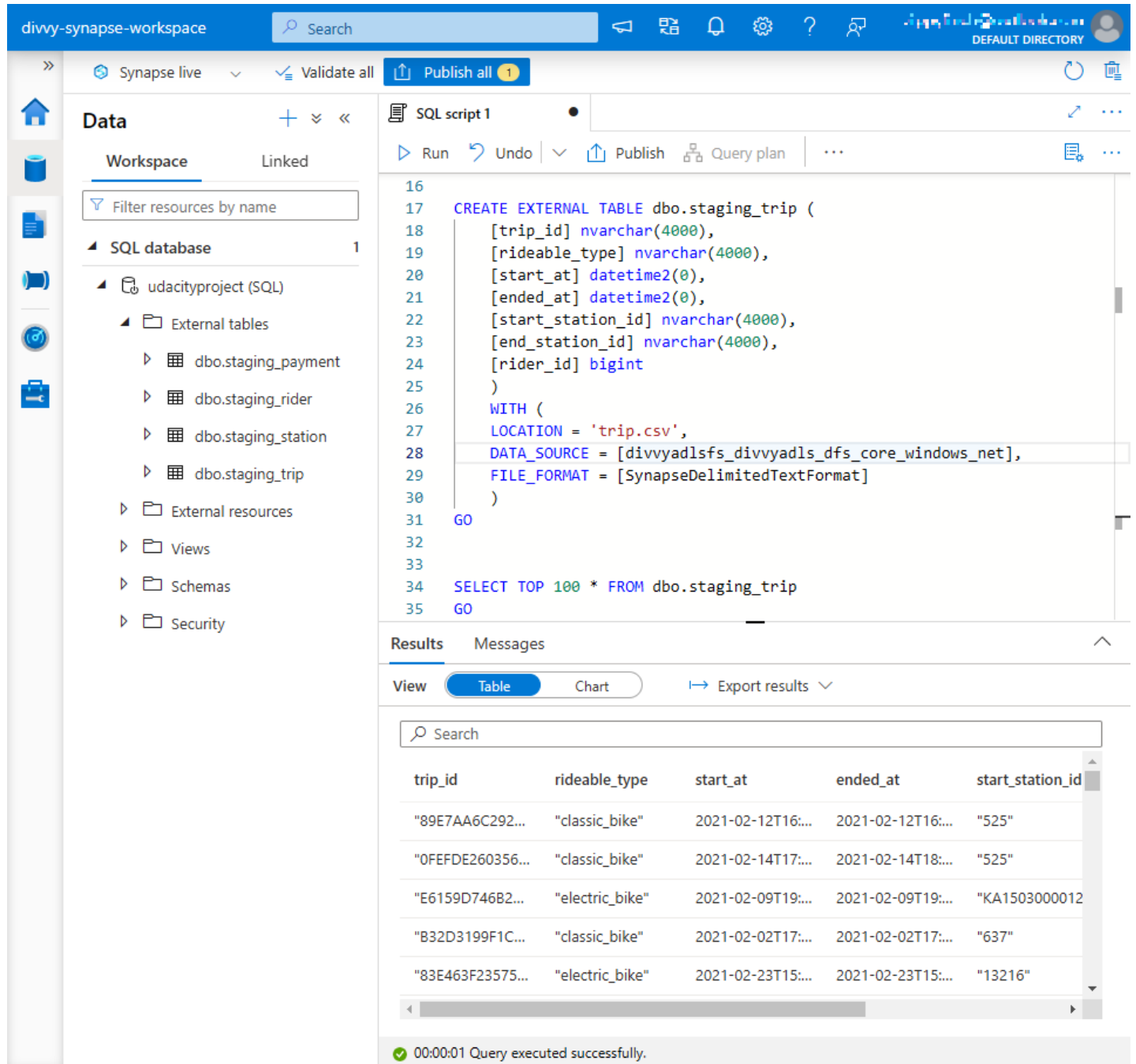
divvyadlsfs

Name	Last Modified	Content Type	Size
payment.csv	5/23/2024, 12:31:29 AM		88.1 MB
rider.csv	5/23/2024, 12:31:07 AM		8.4 MB
station.csv	5/23/2024, 12:31:06 AM		61.3 KB
trip.csv	5/23/2024, 12:32:07 AM		524.7 MB

Showing 1 to 4 of 4 cached items

## 5. Load the data into external tables in the data warehouse

Once in Data Lake storage, the files will be shown in the data lake node in the Synapse Workspace. From here, we can use the script-generating function to load the data into external staging tables in the data warehouse you created using the serverless SQL Pool.



The screenshot displays the Synapse Workspace interface. On the left, the 'Data' pane shows a tree view of resources under 'udacityproject (SQL)', including 'External tables' and 'External resources'. The 'dbo.staging\_trip' table is highlighted. The main pane shows a SQL script titled 'SQL script 1' with the following content:

```
16
17 CREATE EXTERNAL TABLE dbo.staging_trip (
18     [trip_id] nvarchar(4000),
19     [rideable_type] nvarchar(4000),
20     [start_at] datetime2(0),
21     [ended_at] datetime2(0),
22     [start_station_id] nvarchar(4000),
23     [end_station_id] nvarchar(4000),
24     [rider_id] bigint
25 )
26 WITH (
27     LOCATION = 'trip.csv',
28     DATA_SOURCE = [divvyadlsfs_divvyadls_dfs_core_windows_net],
29     FILE_FORMAT = [SynapseDelimitedTextFormat]
30 )
31 GO
32
33
34 SELECT TOP 100 * FROM dbo.staging_trip
35 GO
```

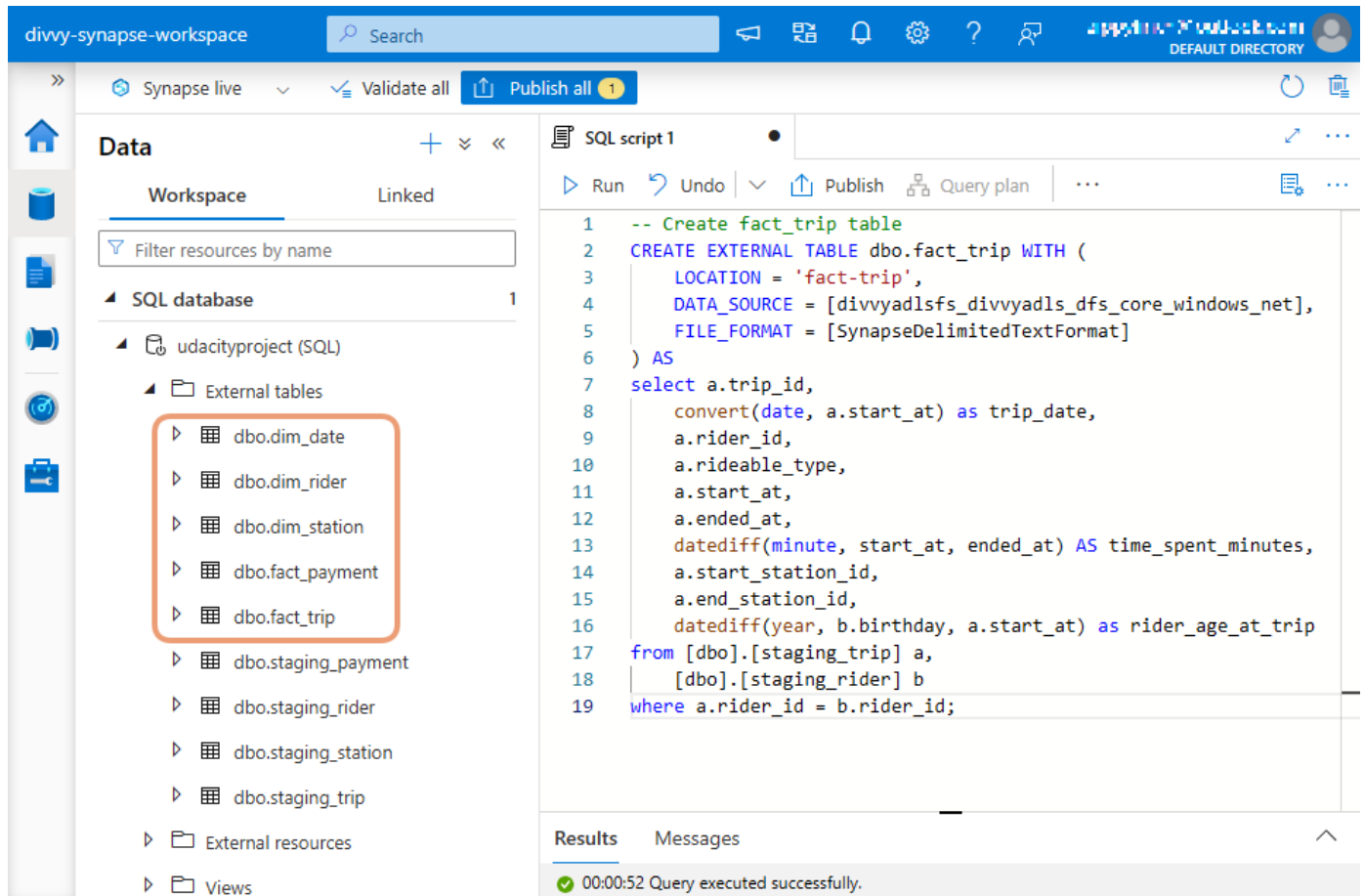
Below the script, the 'Results' pane shows a table view of the query output. The table has five columns: 'trip\_id', 'rideable\_type', 'start\_at', 'ended\_at', and 'start\_station\_id'. The first five rows of data are displayed:

trip_id	rideable_type	start_at	ended_at	start_station_id
"89E7AA6C292...	"classic_bike"	2021-02-12T16:...	2021-02-12T16:...	"525"
"0FEFDE260356...	"classic_bike"	2021-02-14T17:...	2021-02-14T18:...	"525"
"E6159D746B2...	"electric_bike"	2021-02-09T19:...	2021-02-09T19:...	"KA1503000012"
"B32D3199F1C...	"classic_bike"	2021-02-02T17:...	2021-02-02T17:...	"637"
"83E463F23575...	"electric_bike"	2021-02-23T15:...	2021-02-23T15:...	"13216"

At the bottom, a status bar indicates '00:00:01 Query executed successfully.'

## 6. Transform the data to the star schema

Now we will run SQL scripts to transform the data from the staging tables to the final star schema that we designed. The serverless SQL pool won't allow us to create persistent tables in the database, as it has no local storage. So, we will use CREATE EXTERNAL TABLE AS SELECT (CETAS) instead.



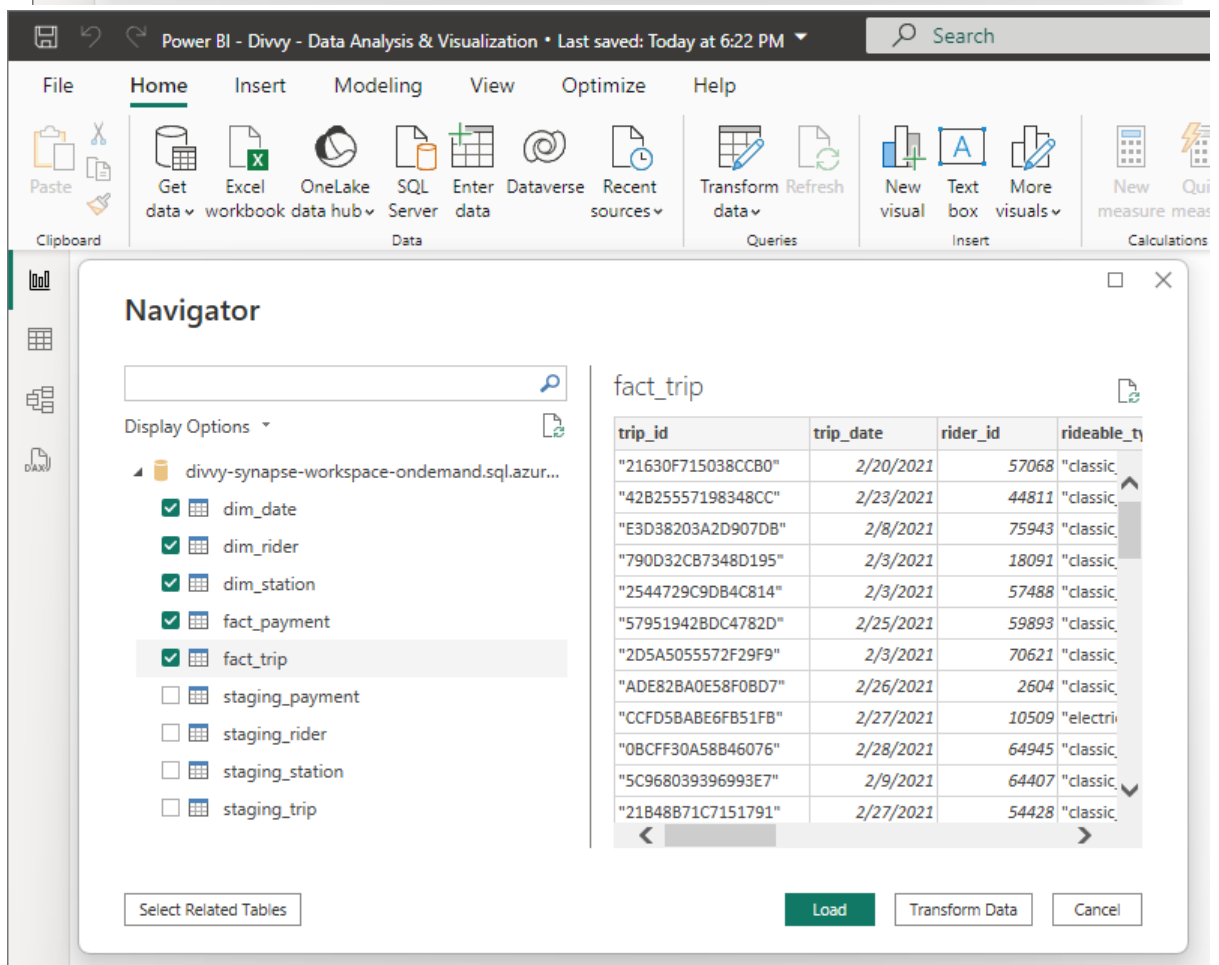
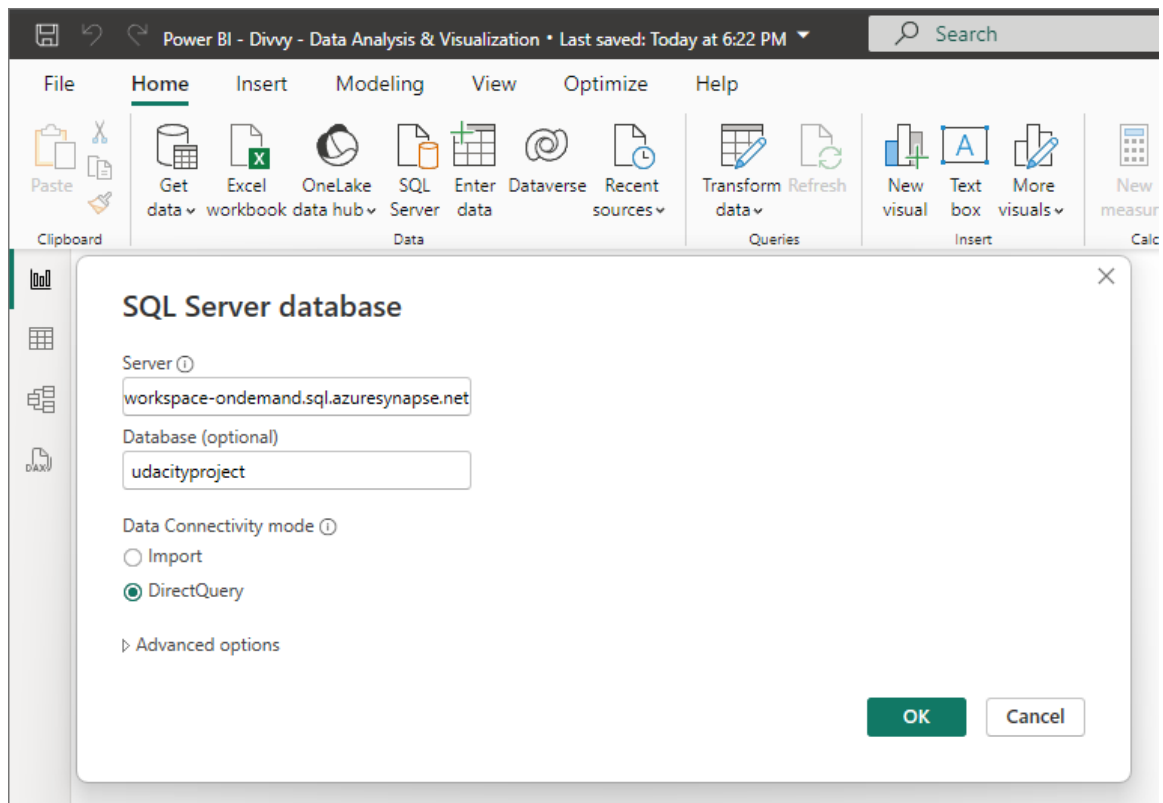
The screenshot shows the Azure Synapse Studio interface. On the left, the 'Data' pane is open, showing a tree view of the 'udacityproject (SQL)' database. Under 'External tables', several tables are listed: 'dbo.dim\_date', 'dbo.dim\_rider', 'dbo.dim\_station', 'dbo.fact\_payment', 'dbo.fact\_trip', 'dbo.staging\_payment', 'dbo.staging\_rider', 'dbo.staging\_station', and 'dbo.staging\_trip'. The 'dbo.fact\_trip' table is highlighted with an orange box. On the right, the 'SQL script 1' editor is open, displaying a SQL script that creates an external table 'dbo.fact\_trip' and then selects data from 'dbo.staging\_trip' and 'dbo.staging\_rider' to populate it. The script is as follows:

```
1 -- Create fact_trip table
2 CREATE EXTERNAL TABLE dbo.fact_trip WITH (
3     LOCATION = 'fact-trip',
4     DATA_SOURCE = [divvyadlsfs_divvyadls_dfs_core_windows_net],
5     FILE_FORMAT = [SynapseDelimitedTextFormat]
6 ) AS
7 select a.trip_id,
8     convert(date, a.start_at) as trip_date,
9     a.rider_id,
10    a.rideable_type,
11    a.start_at,
12    a.ended_at,
13    datediff(minute, start_at, ended_at) AS time_spent_minutes,
14    a.start_station_id,
15    a.end_station_id,
16    datediff(year, b.birthday, a.start_at) as rider_age_at_trip
17 from [dbo].[staging_trip] a,
18     [dbo].[staging_rider] b
19 where a.rider_id = b.rider_id;
```

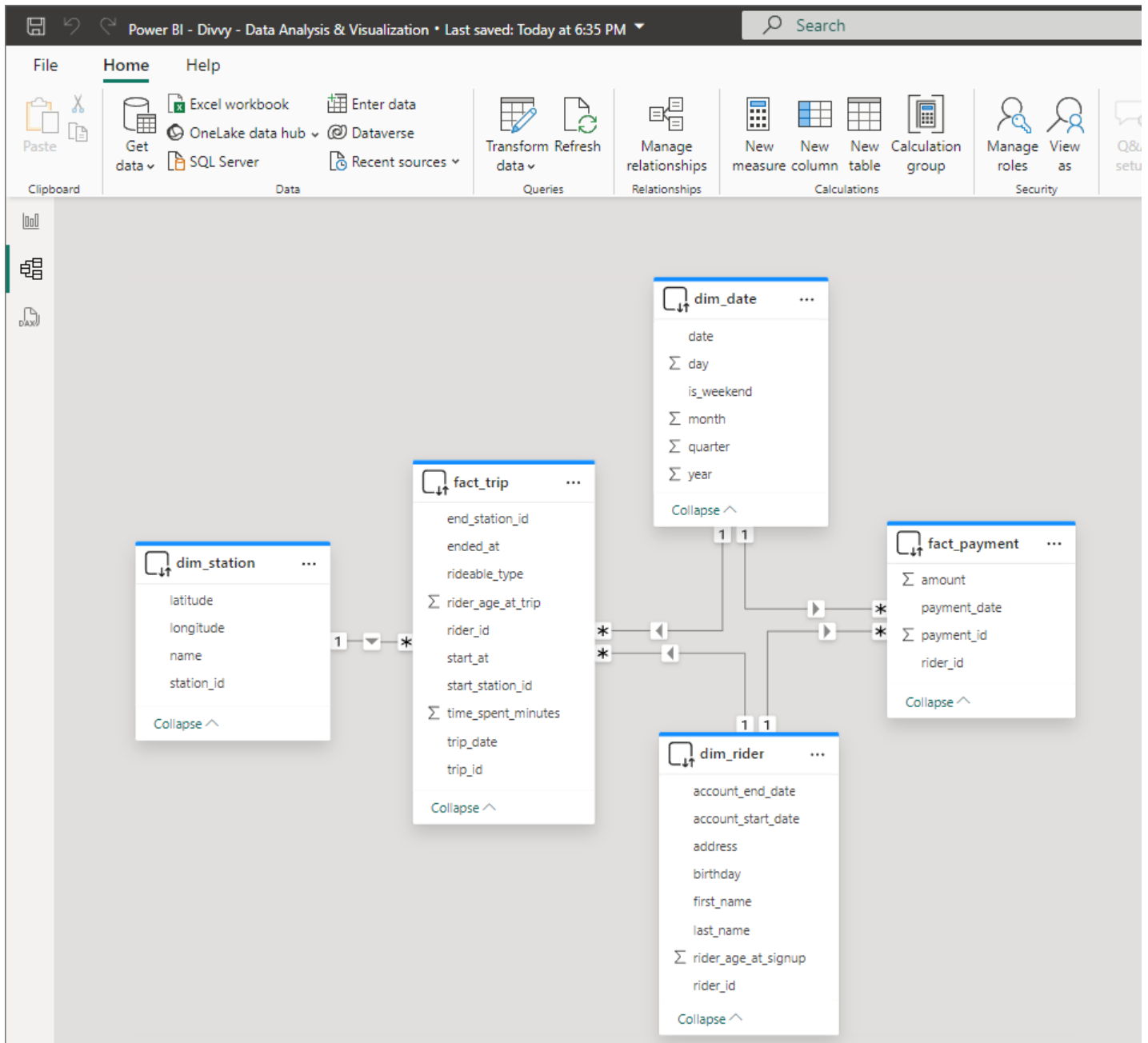
At the bottom of the SQL editor, the 'Results' pane shows a message: '00:00:52 Query executed successfully.'

## 7. Analyze and visualize data with Power BI

Connect to the SQL database using Serverless SQL endpoint and import star schema tables into Power BI.

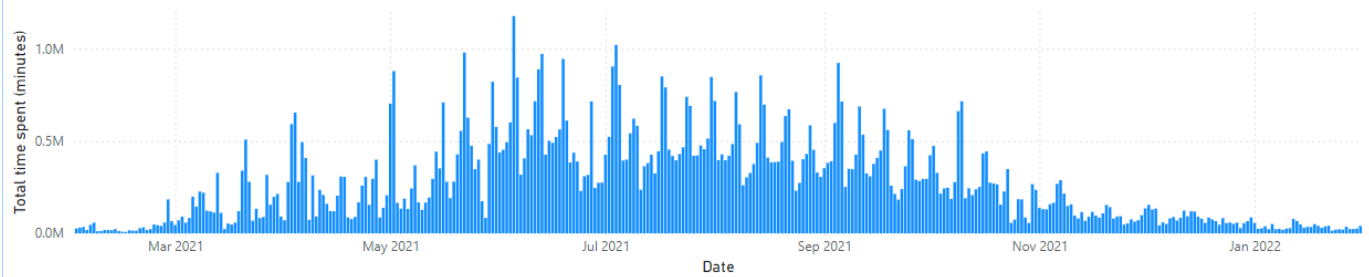


In the Model view, set the relationship between fact and dimension tables.

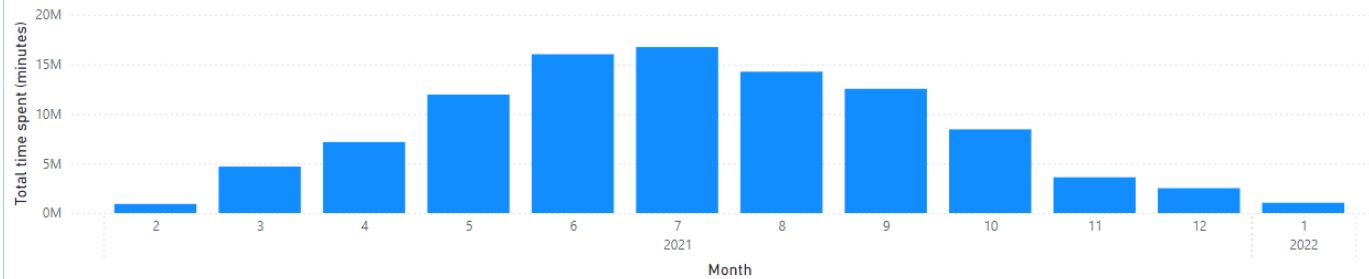


Explore and analyze the data by creating visualizations.

Daily Total Trip Duration



Monthly Total Trip Duration



Total Trip Duration by Pickup Station

