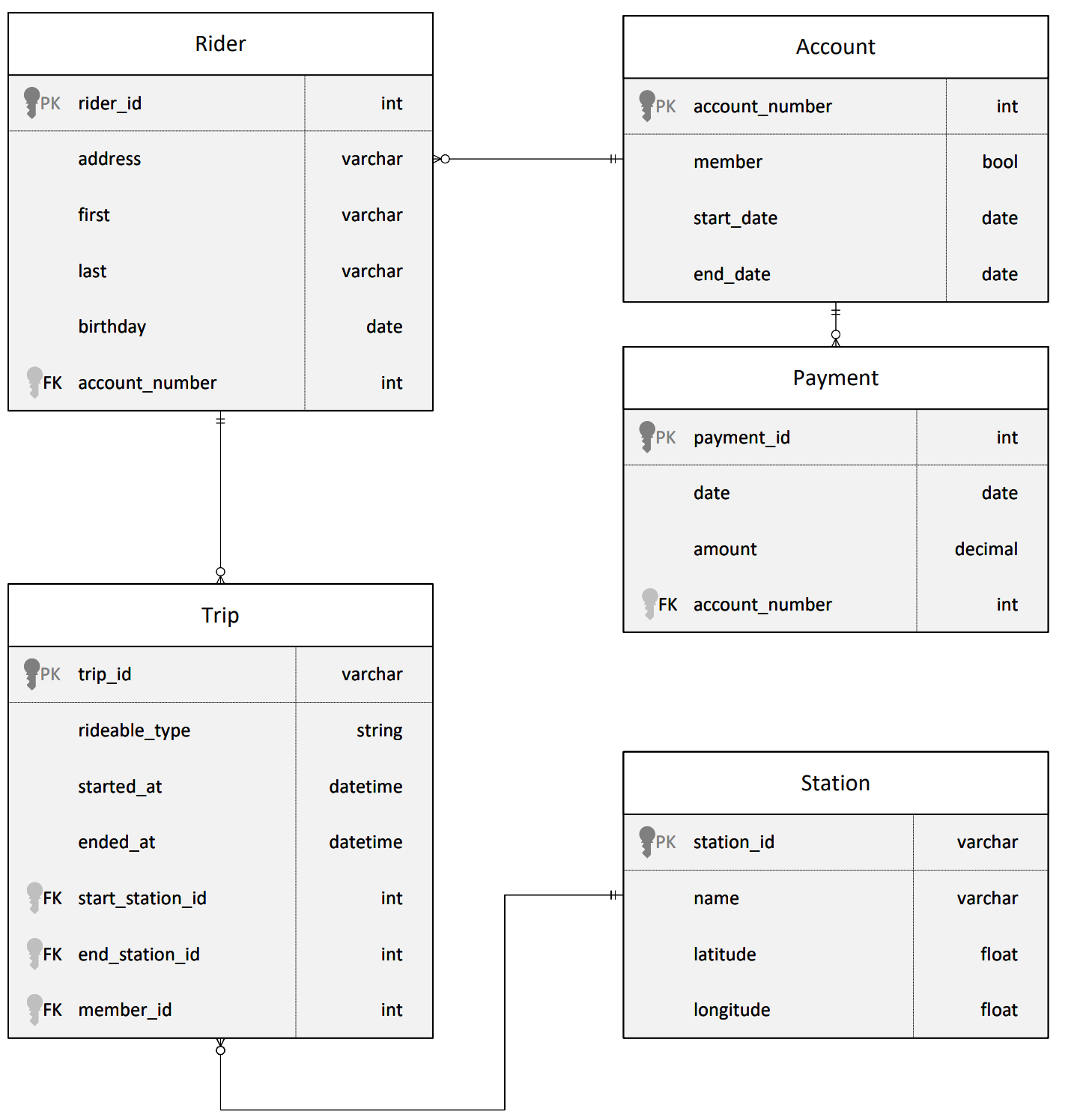
Building an Azure Data Warehouse for Bike Share Data Analytics

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.

Since the data from Divvy are anonymous, we have created fake rider and account profiles along with fake payment data to go along with the data from Divvy. The dataset looks like this:

|  |  |
| --- | --- |
|  |  |



Relational ERD for the Divvy Bikeshare Dataset (with fake data tables)

The goal of this project is to develop a data warehouse solution using Azure Synapse Analytics. You 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 from Analytics.

The business outcomes you 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
3. EXTRA CREDIT - Analyze how much money is spent per member
   * Based on how many rides the rider averages per month
   * Based on how many minutes the rider spends on a bike per month

On the next page are instructions for logging in to an Azure account where you can configure the resources, Azure Synapse Workspace, and data storage to complete the project.

If you are not fully familiar with date dimensions, this article from MSSQL Tips on [**Creating a date dimension or calendar table in SQL Server**](https://www.mssqltips.com/sqlservertip/4054/creating-a-date-dimension-or-calendar-table-in-sql-server/) is a good refresh.

## Project Environment

In order to complete this project, you'll need to use these tools:

* Access to Microsoft Azure.
* Python to run the script to load data into a PostgreSQL database on Azure to simulate your OLTP data source
* An editor to work with the Python and SQL scripts, we recommend [**Visual Studio Code**](https://code.visualstudio.com/)

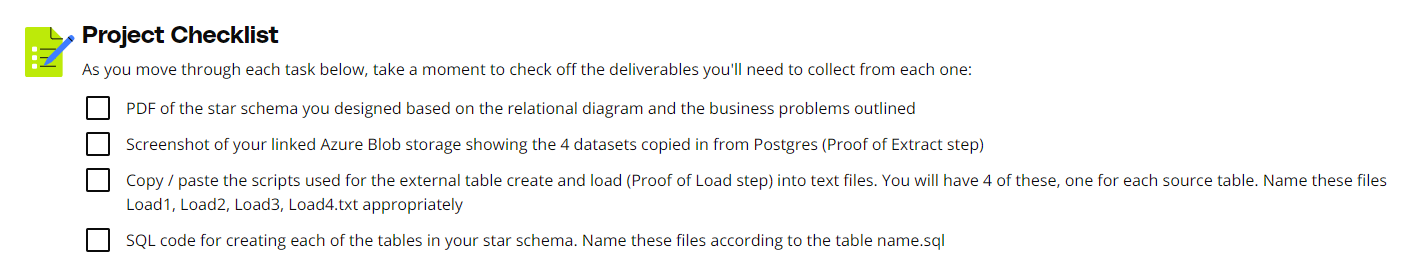
## Local Machine Instructions

To work locally on this project, you'll need to have [**Python**](https://www.python.org/downloads/) and [**Visual Studio Code**](https://code.visualstudio.com/) installed, or another editor of your choice to run Python scripts.

## Starter Code and Data

You'll need a Python script found in the **[Github project starter repository](https://github.com/udacity/Azure-Data-Warehouse-Project/tree/main/starter" \t "_blank)**. Instructions for using this script for loading data are on the next page, in the project instructions

You'll need the [**dataset**](https://video.udacity-data.com/topher/2022/March/622a5fc6_azure-data-warehouse-projectdatafiles/azure-data-warehouse-projectdatafiles.zip) from the course resources menu in the left navigation bar. Instructions for how to use the data in your project are on the next page in the project instructions



### Task 1: Create your Azure resources

* Create an Azure Database for PostgreSQL.
* Create an Azure Synapse workspace. Note that if you've previously created a Synapse Workspace, you do not need to create a second one specifically for the project.
* Use the built-in serverless SQL pool and database within the Synapse workspace

In the cloud lab Azure environment, you will only be able to use the built-in serverless SQL Pool.

### Task 2: Design a star schema

You are being provided a relational schema that describes the data as it exists in PostgreSQL. In addition, you have been given a set of business requirements related to the data warehouse. You are being asked to design a star schema using fact and dimension tables.

### Task 3: Create the data in PostgreSQL

To prepare your environment for this project, you 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 for you in **[Github: ProjectDataToPostgres.py](https://github.com/udacity/Azure-Data-Warehouse-Project/tree/main/starter" \t "_blank)**

1. Download the script file and place it in a folder where you can run a Python script
2. [**Download the data files**](https://video.udacity-data.com/topher/2022/March/622a5fc6_azure-data-warehouse-projectdatafiles/azure-data-warehouse-projectdatafiles.zip) from the classroom resources
3. Open the script file in VS Code and add the host, username, and password information for your PostgreSQL database
4. Run the script and verify that all four data files are copied/uploaded into PostgreSQL

You can verify this data exists by using pgAdmin or a similar PostgreSQL data tool.

### Task 4: EXTRACT the data from PostgreSQL

In your Azure Synapse workspace, you will use the ingest wizard to create a one-time pipeline that ingests the data from PostgreSQL into Azure Blob Storage. This will result in all four tables being represented as text files in Blob Storage, ready for loading into the data warehouse.

### Task 5: LOAD the data into external tables in the data warehouse

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

### Helpful Hints

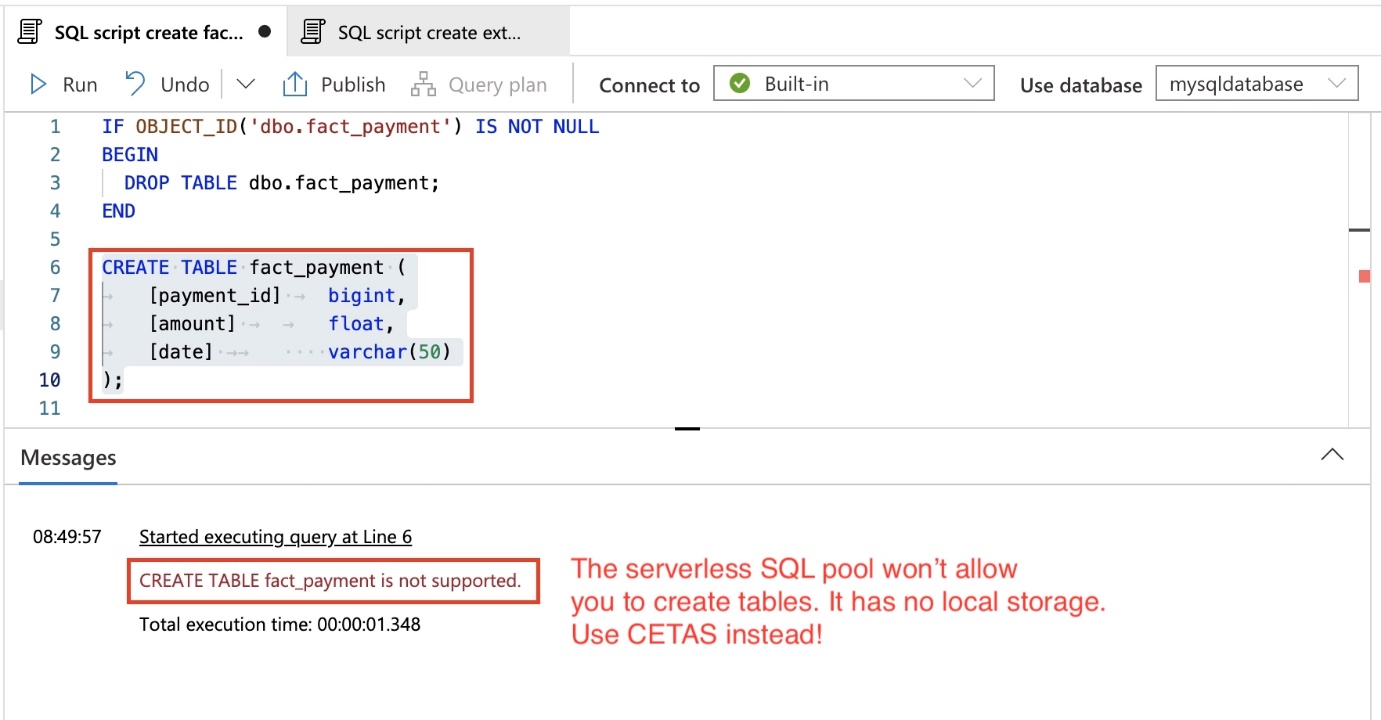
* When you use the ingest wizard, it uses the copy tool to EXTRACT into Blob storage. During this process, Azure Synapse automatically creates links for the data lake. When you start the SQL script wizard to LOAD data into external tables, start the wizard from the data lake node, not the blob storage node.
* When using the external table wizard, you may need to modify the script to put dates into a varchar field in staging rather than using the datetime data type. You can convert them during the transform step.
* When using the external table wizard, if you rename the columns in your script, it will help you when writing transform scripts. By default, they are named [C1], [C2], etc. which are not useful column names in staging.

### Task 6: TRANSFORM the data to the star schema using CETAS

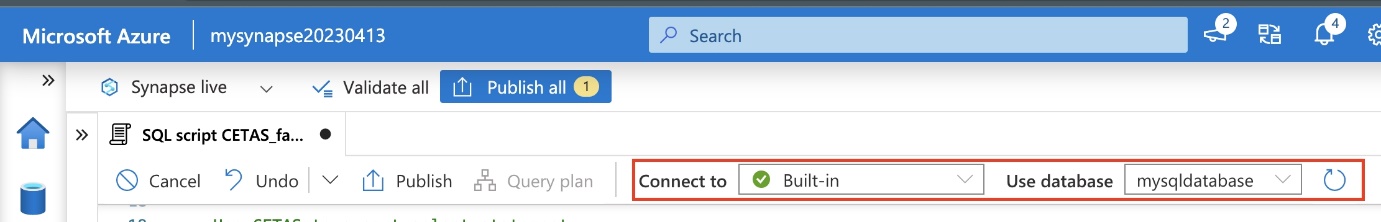
Write SQL scripts to transform the data from the staging tables to the final star schema you designed.

The serverless SQL pool won't allow you to create persistent tables in the database, as it has no local storage. So, use **CREATE EXTERNAL TABLE AS SELECT (CETAS)** instead. CETAS is a parallel operation that creates external table metadata and exports the SELECT query results to a set of files in your storage account.

**Tip**: For creating fact tables out of join between dimensions and staging tables, you can use CETAS to materialize joined reference tables to a new file and then join to this single external table in subsequent queries.



Create a new SQL script, and ensure you are connected to the serverless SQL pool and your SQL database.



We will rely on external tables, created in the previous LOAD step, as the source for CETAS. Assuming you have the staging external tables ready, use the syntax below to define the CETAS.

**1. Define the file format, if not already.** You don't have run this query for each CETAS.

*- Use the same file format as used for creating the External Tables during the LOAD step.*

**IF** NOT **EXISTS** (**SELECT** \* **FROM** sys.external\_file\_formats **WHERE** name = 'SynapseDelimitedTextFormat')

**CREATE** EXTERNAL **FILE** FORMAT [SynapseDelimitedTextFormat]

**WITH** ( FORMAT\_TYPE = DELIMITEDTEXT ,

FORMAT\_OPTIONS (

FIELD\_TERMINATOR = ',',

USE\_TYPE\_DEFAULT = FALSE

))

GO

In this snippet, the file format is being defined for reading in the data from a comma delimited file stored in blob storage. **Note** - The script above is for reference only. It was autogenerated during the LOAD step, when we created the External tables from the Blob storage. Therefore, use the one auto-generated for you.

**2. Define the data source to persist the results**.

*-- Use the same data source as used for creating the External Tables during the LOAD step.*

*-- Storage path where the result set will persist*

**IF** NOT **EXISTS** (**SELECT** \* **FROM** sys.external\_data\_sources **WHERE** name = 'mydlsfs20230413\_mydls20230413\_dfs\_core\_windows\_net')

**CREATE** EXTERNAL **DATA** SOURCE [mydlsfs20230413\_mydls20230413\_dfs\_core\_windows\_net]

**WITH** (

LOCATION = 'abfss://mydlsfs20230413@mydls20230413.dfs.core.windows.net'

)

GO

In this snippet, the external data source is being created in SQL so the data can be read directly into an external table. Use the blob storage account name as applicable to you. The script above was autogenerated during the LOAD step, when we created the External tables from the Blob storage.

**3. Use CETAS to export select statement.** ```sql IF OBJECT\_ID('dbo.fact\_payment') IS NOT NULL BEGIN DROP EXTERNAL TABLE [dbo].[fact\_payment]; END

**CREATE** EXTERNAL **TABLE** dbo.fact\_payment

**WITH** (

LOCATION = 'fact\_payment',

DATA\_SOURCE = [mydlsfs20230413\_mydls20230413\_dfs\_core\_windows\_net],

FILE\_FORMAT = [SynapseDelimitedTextFormat]

)

**AS**

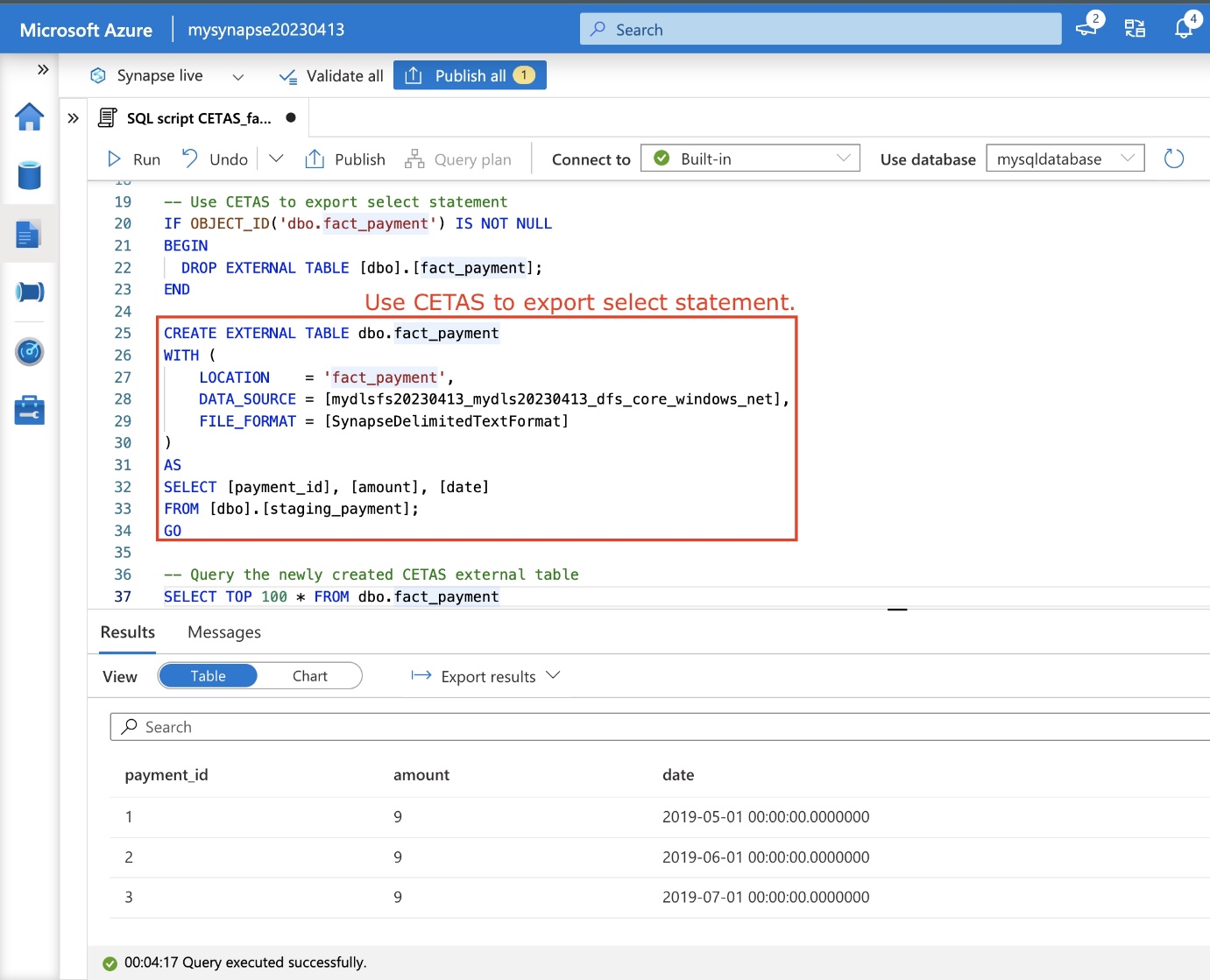
**SELECT** [payment\_id], [amount], [**date**]

**FROM** [dbo].[staging\_payment];

GO

```

The query above will read the data from **dbo.staging\_payment** external table, and persist the results in the **fact\_payment/** directory as CSV format.



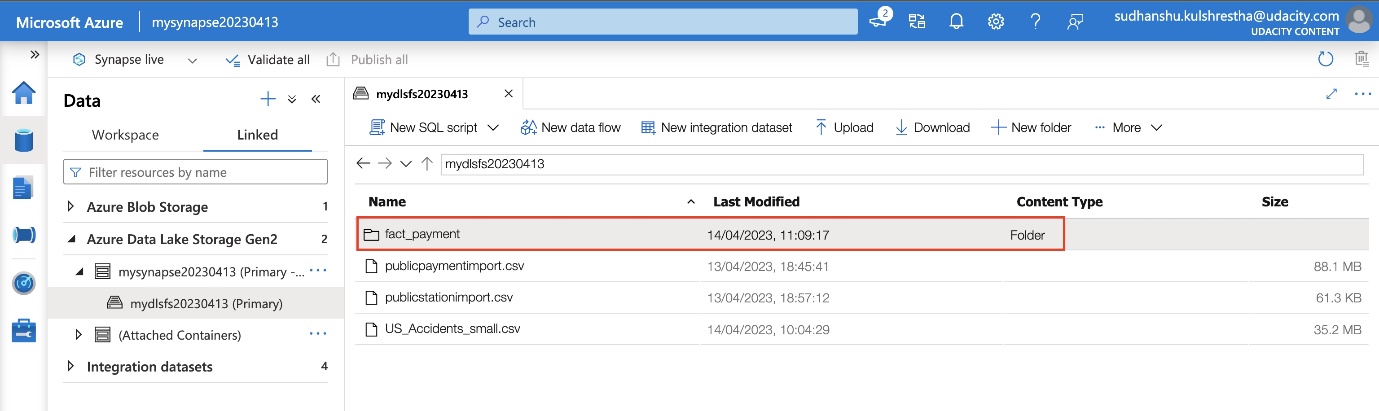
A sample illustration showing the CETAS query. The attributes in the SELECT clause and the source External table will vary for your use-case.

**4. Finally, query the newly created CETAS external table, and ensure you get the desired output**.

**SELECT** **TOP** 100 \* **FROM** dbo.fact\_payment

GO

You can also explore the Linked data source to verify the results.



Reference:

* [**CETAS in serverless SQL pool**](https://learn.microsoft.com/en-us/azure/synapse-analytics/sql/develop-tables-cetas#cetas-in-serverless-sql-pool)
* [**How to use CETAS on serverless SQL pool**](https://techcommunity.microsoft.com/t5/azure-synapse-analytics-blog/how-to-use-cetas-on-serverless-sql-pool-to-improve-performance/ba-p/3548040)
* [**Best practices for serverless SQL pool in Azure Synapse Analytics**](https://learn.microsoft.com/en-us/azure/synapse-analytics/sql/best-practices-serverless-sql-pool)

# **Project: Azure Data Warehouse Project**

## Star Schema Design

| **Success Criteria** | **Specifications** |
| --- | --- |
| The student will be able to generate fact tables based on a business need and a relational model | The star schema should have at least two fact tables. One should be related to trip facts and another should be related to payment facts. The trip fact should have a fields for trip duration and rider age at time of trip. The payment fact should have a field related to amount of payment. |
| The student will be able to generate dimension tables based on business needs and a relational model | The star schema should have dimensions related to the trip fact table that are related to: riders, stations, and dates. The schema should have dimensions related to the payment fact table that are related to: dates and riders. |

## Extract Step

| **Success Criteria** | **Specifications** |
| --- | --- |
| The screenshot will demonstrate the learner is able to extract data from PostgreSQL into Azure Blob Storage. | The screenshot should be of the Azure Blob Storage and should contain 4 text files: public.payment, public.rider, public.trip, public.station. |

## Load Step

| **Success Criteria** | **Specifications** |
| --- | --- |
| The student will be able to load data from Azure Blob Storage into external tables in the data warehouse. | The student will have uploaded 4 separate script files. The SQL files should create tables using CREATE EXTERNAL TABLE (not just CREATE TABLE). The scripts should point to the four files in Blob Storage from the extract step. |

## Transform step

| **Success Criteria** | **Specifications** |
| --- | --- |
| The scripts show the student is able to generate fact tables (CETAS) from staging tables. | The fact table (CETAS) scripts should contain appropriate keys from the dimensions. In addition, the fact table scripts should appropriately generate the correct facts based on the diagrams provided in the first step.  **Tips**:   1. Ensure the column names match the reference diagrams created in Step 1. 2. For creating fact tables out of join between dimensions and staging tables, you can use CETAS to materialize joined reference tables to a new file and then join to this single external table in subsequent queries. |
| The scripts show the student can generate dimension tables (CETAS) from staging tables. | The dimension scripts (CETAS) should match the schema diagram. Dimensions should generate appropriate keys and should not contain facts. |

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

**Completion Due December 18, 2023**

### Submission Instructions

Here is a list of things to submit for this project. Create a zip file containing:

* PDF of the star schema you designed based on the relational diagram and the business problems outlined
* Screenshot of your linked Azure Blob storage showing the 4 datasets copied in from Postgres (Proof of Extract step)
* Copy / paste of the script used for the external table create and load (Proof of Load step). You will have 4 of these, one for each source table.
* SQL code for creating each of the tables (CETAS) per your star schema.

Your submission will be assessed against this [**rubric**](https://review.udacity.com/#!/rubrics/5163/view)

Take a moment and make sure your project is complete before submitting it.