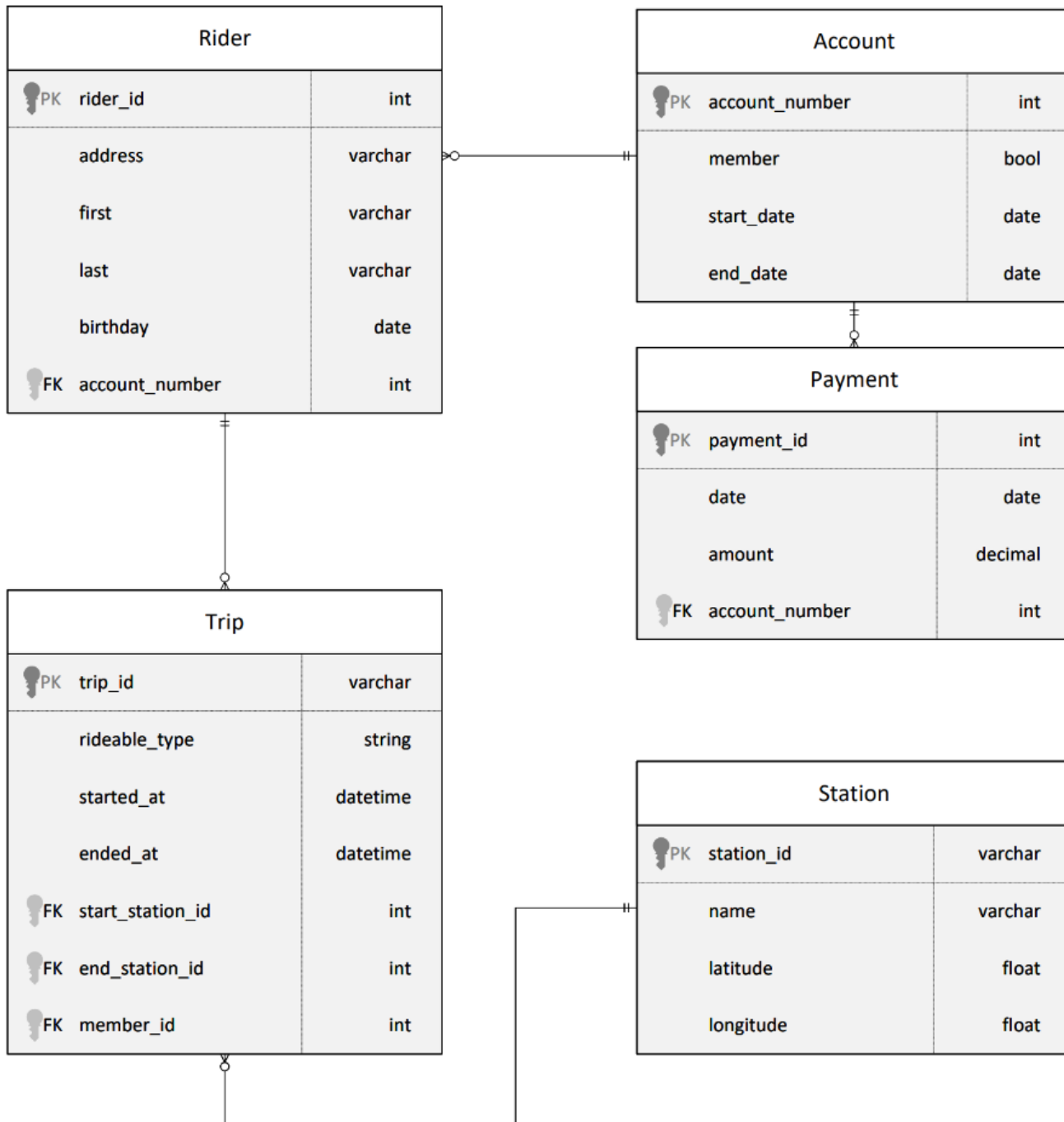


Building an Azure Data Warehouse for Bike Share Data Analytics

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:



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

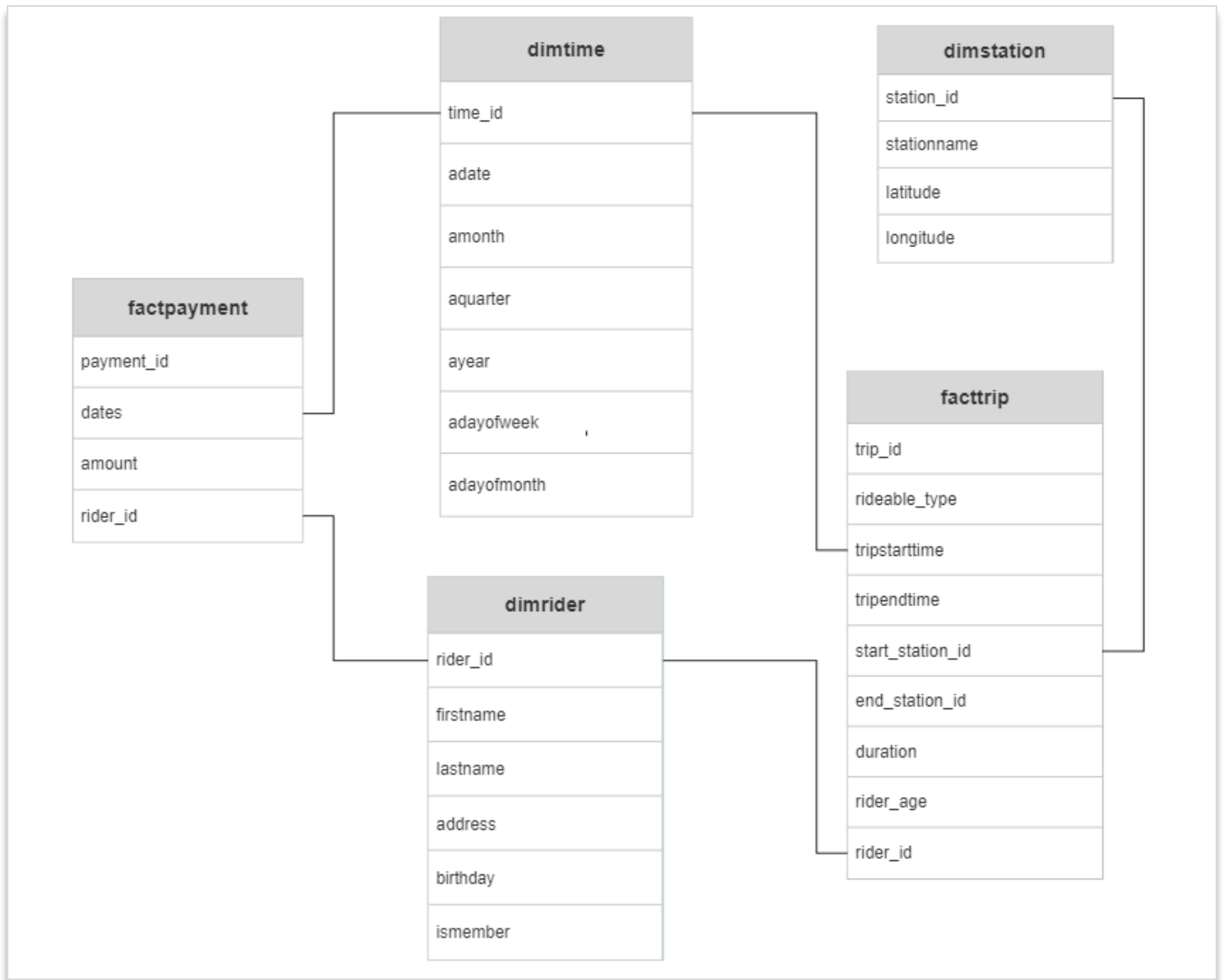
- Design a star schema based on the business outcomes listed below;
- Import the data into Synapse;
- Transform the data into the star schema;
- and finally, view the reports from Analytics.

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. 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

Task 1: Design a star schema

Designing a Star Schema based on the relation diagram and the business problems outlined above:



Star Schema design comprising 2 fact tables, and 3 dimension tables:

FACT TABLES:

- FactTrip
- FactPayment

DIMENSION TABLES:

- Dimrider
- DimTime
- Dimstation

Task 2: Create Azure resources

- Azure PostgreSQL database
- Azure Blob Storage
- Azure Synapse workspace (comes with Azure DataLake Gen2 storage in the process)
- Dedicated SQL Pool and database within the Synapse workspace (Connected on Synapse studio)

All resources

Default Directory

+ Create Manage view Refresh Export to CSV Open query Assign tags Delete

Filter for any field... Subscription equals all Resource group equals all Type equals all Location equals all Add filter

2 Unsecure resources 4 Recommendations

| Name | Type | Resource group | Location |
|---|---|------------------------|---------------------|
| divvybikesblobstorage | Storage account | Regroup_6vqGMvZtUDHgPG | Australia Southeast |
| divvybikesharepostgresql | Azure Database for PostgreSQL flexible server | Regroup_6vqGMvZtUDHgPG | Australia Southeast |
| divvybikesynapsews1 | Synapse workspace | Regroup_6vqGMvZtUDHgPG | Australia Southeast |
| divvyws1datalakestorage | Storage account | Regroup_6vqGMvZtUDHgPG | Australia Southeast |
| divvyws1sqlpool (divvybikesynapsews1/divvyws1sqlpool) | Dedicated SQL pool | Regroup_6vqGMvZtUDHgPG | Australia Southeast |

Task 3: Create the data in PostgreSQL

Downloaded Github: ProjectDataToPostgres.py and the Data Files

```
AzureDataWarehouseProject.py
host

# Create a new DB
sslmode = "require"
dbname = "postgres"
conn_string = "host={0} user={1} dbname={2} password={3} sslmode={4}".format(host, user, dbname, password, sslmode)
conn = psycopg2.connect(conn_string)
conn.set_isolation_level(ISOLATION_LEVEL_AUTOCOMMIT);
print("Connection established")

cursor = conn.cursor()
cursor.execute('DROP DATABASE IF EXISTS udacityproject')
cursor.execute('CREATE DATABASE udacityproject')
# Clean up initial connection
conn.commit()
cursor.close()
conn.close()

# Reconnect to the new DB
dbname = "udacityproject"
conn_string = "host={0} user={1} dbname={2} password={3} sslmode={4}".format(host, user, dbname, password, sslmode)
conn = psycopg2.connect(conn_string)
print("Connection established")
cursor = conn.cursor()

# Helper functions
def drop_recreate(c, tablename, create):
    c.execute("DROP TABLE IF EXISTS {0};".format(tablename))
    c.execute(create)
    print("Finished creating table {0}".format(tablename))

def populate_table(c, filename, tablename):
    f = open(filename, 'r')
    try:
        cursor.copy_from(f, tablename, sep=",", null = "")
        conn.commit()
    except (Exception, psycopg2.DatabaseError) as error:
        print("Error: %s" % error)
        conn.rollback()
        cursor.close()
    print("Finished populating {0}".format(tablename))

# Create Rider table
drop_recreate(c, "rider", "CREATE TABLE rider (id SERIAL, name VARCHAR(255), age INT, weight INT, height INT, gender VARCHAR(255), PRIMARY KEY (id));")
populate_table(c, "rider.csv", "rider")

drop_recreate(c, "payment", "CREATE TABLE payment (id SERIAL, rider_id INT, amount DECIMAL(10,2), PRIMARY KEY (id), FOREIGN KEY (rider_id) REFERENCES rider(id));")
populate_table(c, "payment.csv", "payment")

drop_recreate(c, "station", "CREATE TABLE station (id SERIAL, name VARCHAR(255), lat DECIMAL(9,6), lon DECIMAL(9,6), PRIMARY KEY (id));")
populate_table(c, "station.csv", "station")

drop_recreate(c, "trip", "CREATE TABLE trip (id SERIAL, rider_id INT, station_id INT, start_time TIMESTAMP, end_time TIMESTAMP, PRIMARY KEY (id), FOREIGN KEY (rider_id) REFERENCES rider(id), FOREIGN KEY (station_id) REFERENCES station(id));")
populate_table(c, "trip.csv", "trip")

drop_recreate(c, "payment_trip", "CREATE TABLE payment_trip (trip_id INT, amount DECIMAL(10,2), PRIMARY KEY (trip_id), FOREIGN KEY (trip_id) REFERENCES trip(id));")
populate_table(c, "payment_trip.csv", "payment_trip")

# Create Station table
drop_recreate(c, "station", "CREATE TABLE station (id SERIAL, name VARCHAR(255), lat DECIMAL(9,6), lon DECIMAL(9,6), PRIMARY KEY (id));")
populate_table(c, "station.csv", "station")

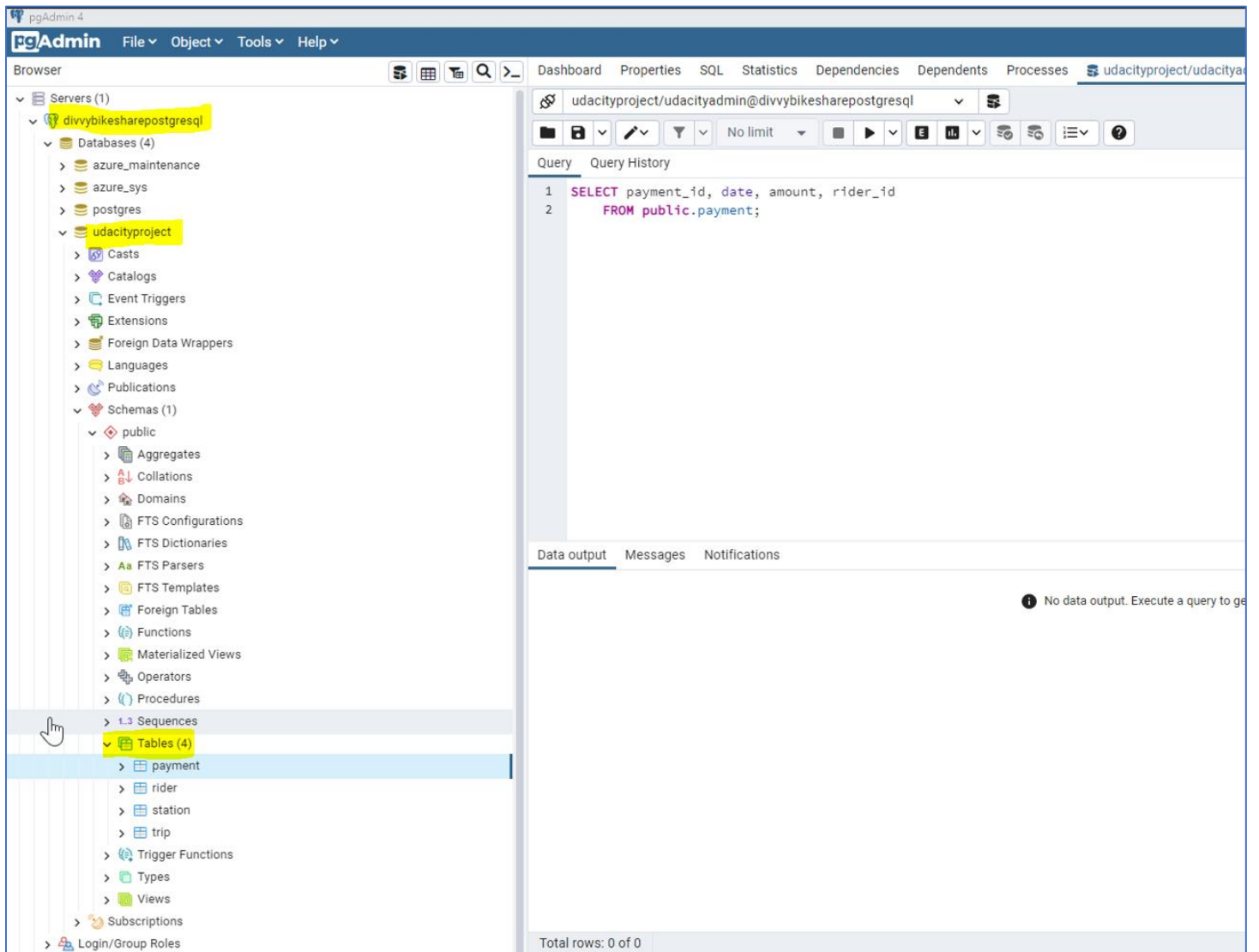
# Create Trip table
drop_recreate(c, "trip", "CREATE TABLE trip (id SERIAL, rider_id INT, station_id INT, start_time TIMESTAMP, end_time TIMESTAMP, PRIMARY KEY (id), FOREIGN KEY (rider_id) REFERENCES rider(id), FOREIGN KEY (station_id) REFERENCES station(id));")
populate_table(c, "trip.csv", "trip")

# Create Payment table
drop_recreate(c, "payment", "CREATE TABLE payment (id SERIAL, rider_id INT, amount DECIMAL(10,2), PRIMARY KEY (id), FOREIGN KEY (rider_id) REFERENCES rider(id));")
populate_table(c, "payment.csv", "payment")

# Create Payment_Trip table
drop_recreate(c, "payment_trip", "CREATE TABLE payment_trip (trip_id INT, amount DECIMAL(10,2), PRIMARY KEY (trip_id), FOREIGN KEY (trip_id) REFERENCES trip(id));")
populate_table(c, "payment_trip.csv", "payment_trip")

All done!
Press any key to continue . . .
```

Uploaded all 4 data files using the Python script file in Visual Studio, by updating the host, username, and password information of the Azure Database for PostgreSQL resource. Verified this data exists by using pgAdmin4.



Task 4: EXTRACT the data from PostgreSQL

In Azure Synapse workspace, created 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.

The screenshot shows the Microsoft Azure Copy Data tool interface. The top navigation bar indicates the workspace is 'divvybikesharesynapse'. The left sidebar shows the 'Copy Data tool' workflow with steps: Properties, Source, Destination, Settings, Review and finish (selected), Review, and Deployment. The main area displays a diagram of data flow from 'Azure Database for PostgreSQL' to 'Azure Blob Storage'. Below this, a 'Deployment complete' message is shown. A table lists the deployment steps and their status:

| Deployment step | Status |
|-------------------------------------|-------------|
| Validating copy runtime environment | ✓ Succeeded |
| ✓ Creating datasets | ✓ Succeeded |
| SourceDataset_gwz | ✓ Succeeded |
| DestinationDataset_gwz | ✓ Succeeded |
| ✓ Creating pipelines | ✓ Succeeded |
| Pipeline_azpostgreddb_to_azblob | ✓ Succeeded |
| ✓ Running pipelines | ✓ Succeeded |
| Pipeline_azpostgreddb_to_azblob | ✓ Succeeded |

Below the table, a message states: 'Datasets and pipelines have been created. You can now monitor and edit the copy pipelines or click finish to close Copy Data Tool.' At the bottom, there are buttons for 'Finish', 'Edit pipeline', and 'Monitor'.

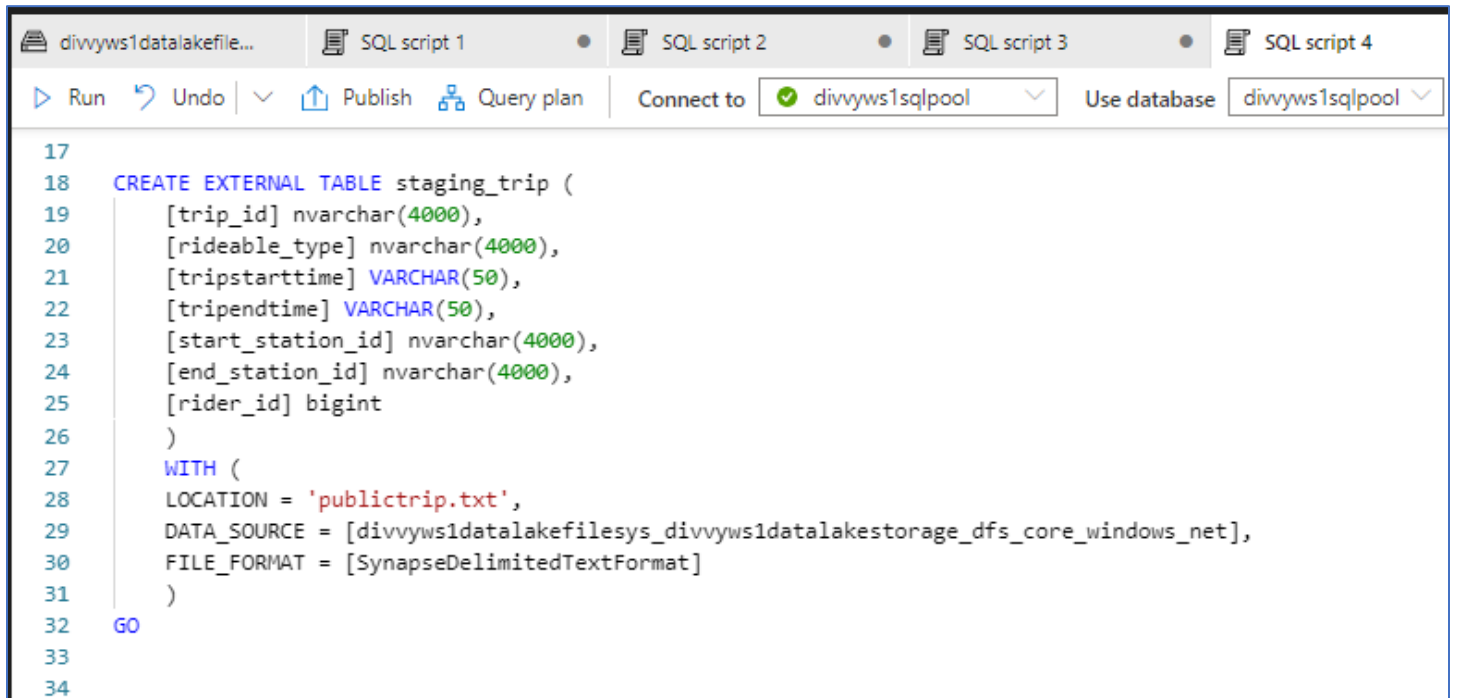
Proof of Extract: Azure Blob storage container with the 4 extracted files

The screenshot shows the Azure portal interface for a container named 'dataset'. The left sidebar shows the 'dataset' container selected. The main area displays the container's contents. The 'Authentication method' is 'Access key (Switch to Azure AD User Account)' and the 'Location' is 'dataset'. A search bar is present. Below the search bar, there is a table listing the files in the container:

| Name | Modified | Access tier | Archive status | Blob type | Size | Lease state |
|--|-----------------------|----------------|----------------|------------|------------|-------------|
| <input type="checkbox"/> publicpayment.txt | 9/21/2022, 1:13:18 AM | Hot (Inferred) | | Block blob | 88.08 MiB | Available |
| <input type="checkbox"/> publicrider.txt | 9/21/2022, 1:13:08 AM | Hot (Inferred) | | Block blob | 8.44 MiB | Available |
| <input type="checkbox"/> publicstation.txt | 9/21/2022, 1:13:08 AM | Hot (Inferred) | | Block blob | 49.75 KiB | Available |
| <input type="checkbox"/> publictrip.txt | 9/21/2022, 1:14:15 AM | Hot (Inferred) | | Block blob | 524.68 MiB | Available |

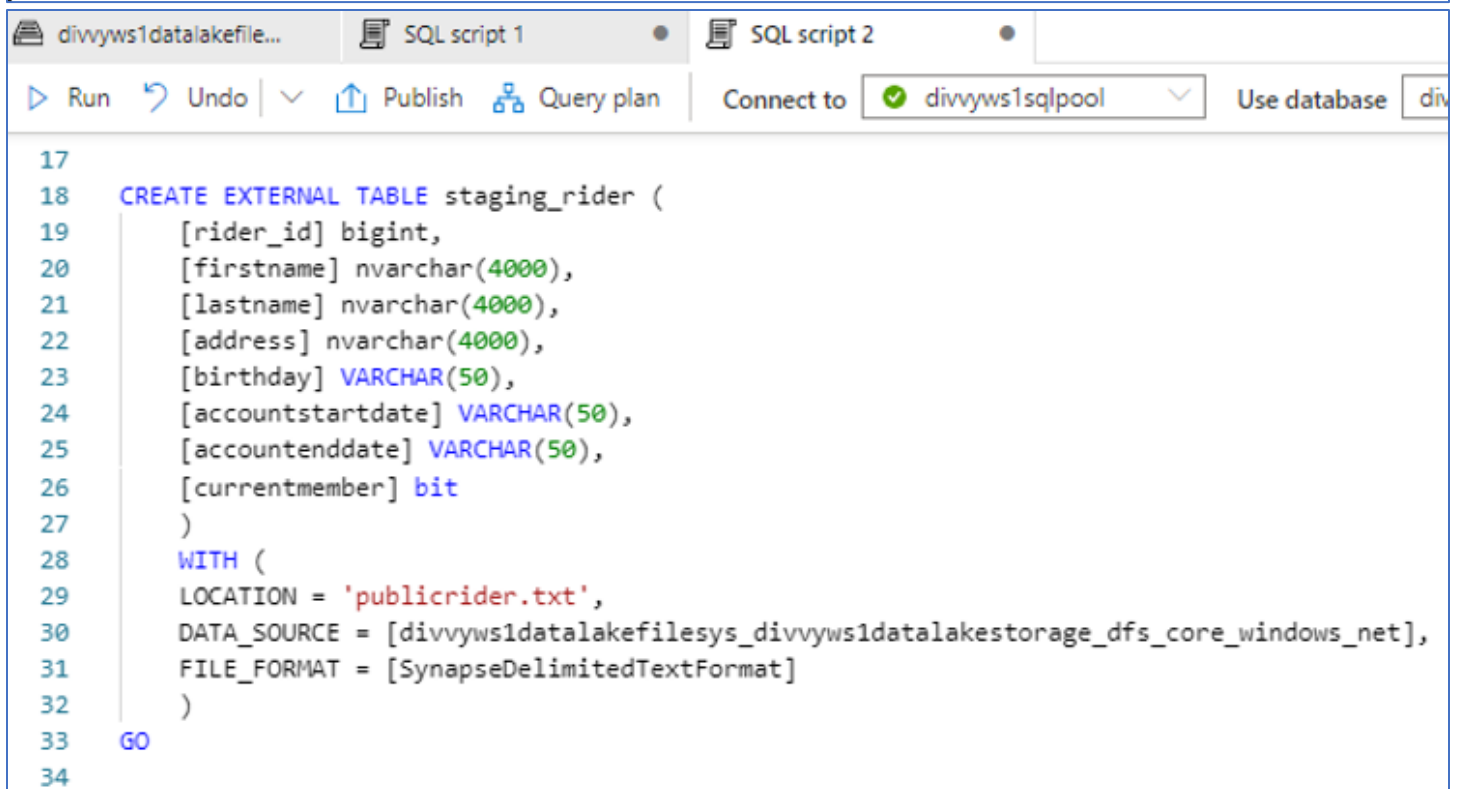
Task 5: LOAD the data into external tables in the data warehouse

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 Dedicated SQL Pool.



The screenshot shows the Synapse Studio SQL editor interface. The top toolbar includes buttons for Run, Undo, Publish, and Query plan. The 'Connect to' dropdown is set to 'divvyws1sqlpool' and the 'Use database' dropdown is also set to 'divvyws1sqlpool'. The SQL script is as follows:

```
17
18 CREATE EXTERNAL TABLE staging_trip (
19     [trip_id] nvarchar(4000),
20     [rideable_type] nvarchar(4000),
21     [tripstarttime] VARCHAR(50),
22     [tripendtime] VARCHAR(50),
23     [start_station_id] nvarchar(4000),
24     [end_station_id] nvarchar(4000),
25     [rider_id] bigint
26 )
27 WITH (
28     LOCATION = 'publictrip.txt',
29     DATA_SOURCE = [divvyws1datalakefilesys_divvyws1datalakestorage_dfs_core_windows_net],
30     FILE_FORMAT = [SynapseDelimitedTextFormat]
31 )
32 GO
33
34
```



The screenshot shows the Synapse Studio SQL editor interface. The top toolbar includes buttons for Run, Undo, Publish, and Query plan. The 'Connect to' dropdown is set to 'divvyws1sqlpool' and the 'Use database' dropdown is also set to 'divvyws1sqlpool'. The SQL script is as follows:

```
17
18 CREATE EXTERNAL TABLE staging_rider (
19     [rider_id] bigint,
20     [firstname] nvarchar(4000),
21     [lastname] nvarchar(4000),
22     [address] nvarchar(4000),
23     [birthday] VARCHAR(50),
24     [accountstartdate] VARCHAR(50),
25     [accountenddate] VARCHAR(50),
26     [currentmember] bit
27 )
28 WITH (
29     LOCATION = 'publicrider.txt',
30     DATA_SOURCE = [divvyws1datalakefilesys_divvyws1datalakestorage_dfs_core_windows_net],
31     FILE_FORMAT = [SynapseDelimitedTextFormat]
32 )
33 GO
34
```

```

18 CREATE EXTERNAL TABLE staging_payment (
19     [payment_id] bigint,
20     [dates] varchar(50),
21     [amount] float,
22     [rider_id] bigint
23 )
24 WITH (
25     LOCATION = 'publicpayment.txt',
26     DATA_SOURCE = [divvyws1datalakefilesys_divvyws1datalakestorage_dfs_core_windows_net],
27     FILE_FORMAT = [SynapseDelimitedTextFormat]
28 )
29 GO
30
31
32 SELECT TOP 100 * FROM dbo.staging_payment
33 GO

```

divvyws1datalakefile... SQL script 1 SQL script 2 SQL script 3

Run Undo Publish Query plan Connect to divvyws1sqlpool Use database divvyws1sqlp

```

17
18 CREATE EXTERNAL TABLE staging_station (
19     [station_id] nvarchar(4000),
20     [stationname] nvarchar(4000),
21     [latitude] float,
22     [longitude] float
23 )
24 WITH (
25     LOCATION = 'publicstation.txt',
26     DATA_SOURCE = [divvyws1datalakefilesys_divvyws1datalakestorage_dfs_core_windows_net],
27     FILE_FORMAT = [SynapseDelimitedTextFormat]
28 )
29 GO
30
31

```

Microsoft Azure | Synapse Analytics | divvybikesynapsews1

Synapse live Validate all Publish all

Data + - <<

Workspace Linked

Filter resources by name

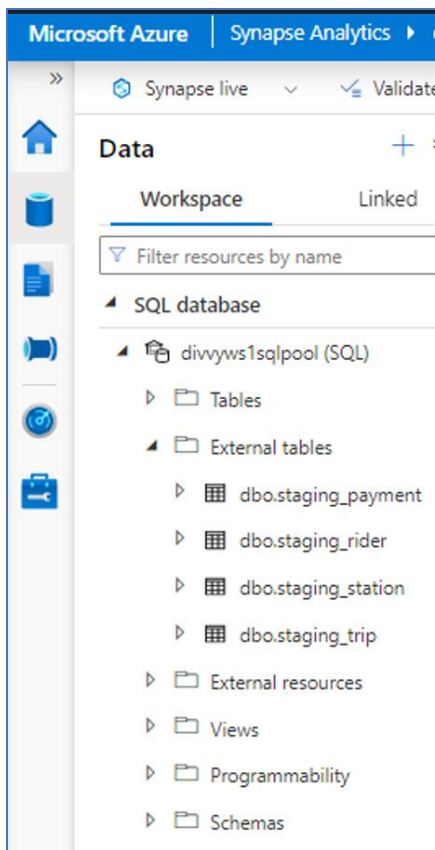
- Azure Blob Storage 1
 - azbloblink (divvybikesblobstorage)
 - dataset
- Azure Data Lake Storage Gen2 2
 - divvybikesynapsews1 (Primary - div...)
 - divvyws1datalakefilesys (Primary)
 - (Attached Containers)

dataset divvyws1datalakefile...

New SQL script New notebook Upload Download New folder Select all Rename

divvyws1datalakefilesys

| Name | Last Modified |
|-------------------|------------------------|
| publicpayment.txt | 9/23/2022, 12:24:06 PM |
| publicrider.txt | 9/23/2022, 12:24:01 PM |
| publicstation.txt | 9/23/2022, 12:24:16 PM |
| publictrip.txt | 9/23/2022, 12:30:08 PM |



Task 6: TRANSFORM the data to the star schema

We will write SQL scripts to transform the data from the staging tables to the final star schema designed.

