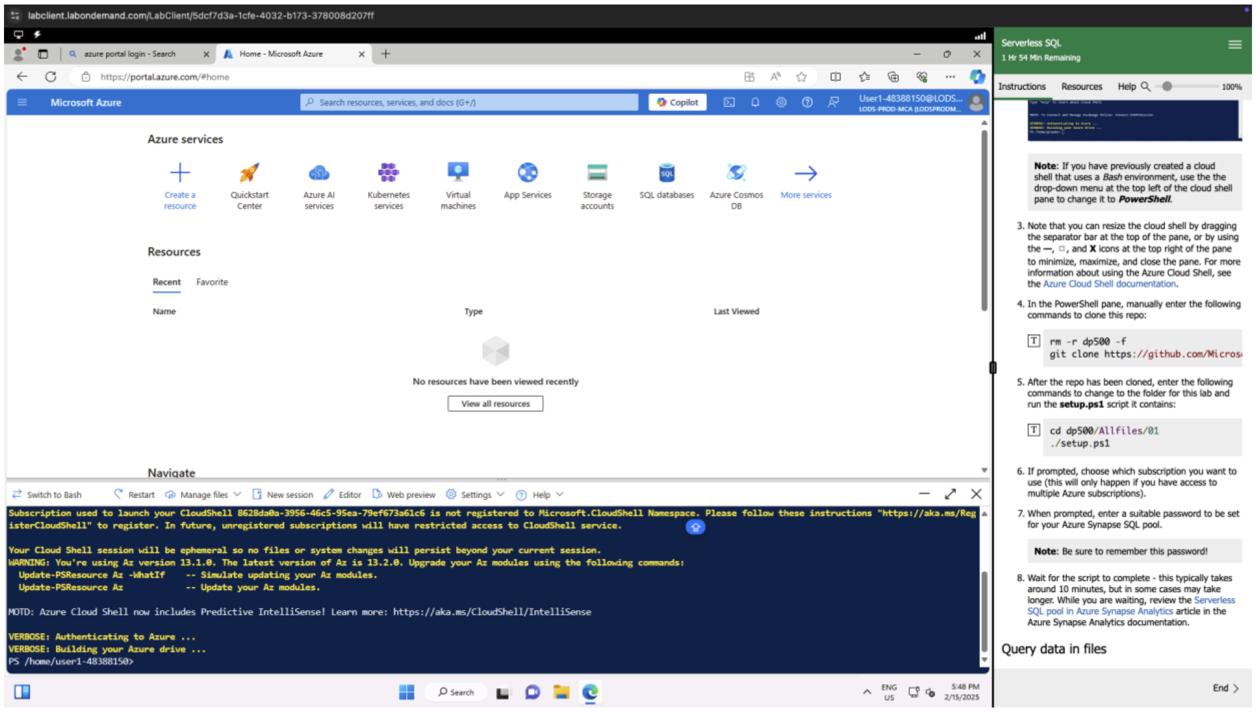
Lab 2 Serverless SQL
BUAN 6390.001 – Analytics Practicum

Name: Swastik Bhatnagar
NetID: sxb220210

Connecting to PowerShell:



```
Subscription used to launch your CloudShell 8628da0a-3956-46c5-95ea-79ef673a61c6 is not registered to Microsoft.CloudShell Namespace. Please follow these instructions "https://aka.ms/Reg
isterCloudShell" to register. In future, unregistered subscriptions will have restricted access to CloudShell service.

Your Cloud Shell session will be ephemeral so no files or system changes will persist beyond your current session.
WARNING: You're using Az version 13.1.0. The latest version of Az is 13.2.0. Upgrade your Az modules using the following commands:
  Update-PSResource Az -WhatIf    -- Simulate updating your Az modules.
  Update-PSResource Az            -- Update your Az modules.

MOTD: Azure Cloud Shell now includes Predictive IntelliSense! Learn more: https://aka.ms/CloudShell/IntelliSense

VERBOSE: Authenticating to Azure ...
VERBOSE: Building your Azure drive ...
PS /home/user1-48388150>
```

Note: If you have previously created a cloud shell that uses a *Bash* environment, use the the drop-down menu at the top left of the cloud shell pane to change it to ***PowerShell***.

3. Note that you can resize the cloud shell by dragging the separator bar at the top of the pane, or by using the —, ◻, and **X** icons at the top right of the pane to minimize, maximize, and close the pane. For more information about using the Azure Cloud Shell, see the Azure Cloud Shell documentation.
4. In the PowerShell pane, manually enter the following commands to clone this repo:

```
rm -r dp500 -f
git clone https://github.com/Micros
```

5. After the repo has been cloned, enter the following commands to change to the folder for this lab and run the **setup.ps1** script it contains:

```
cd dp500/Allfiles/01
./setup.ps1
```

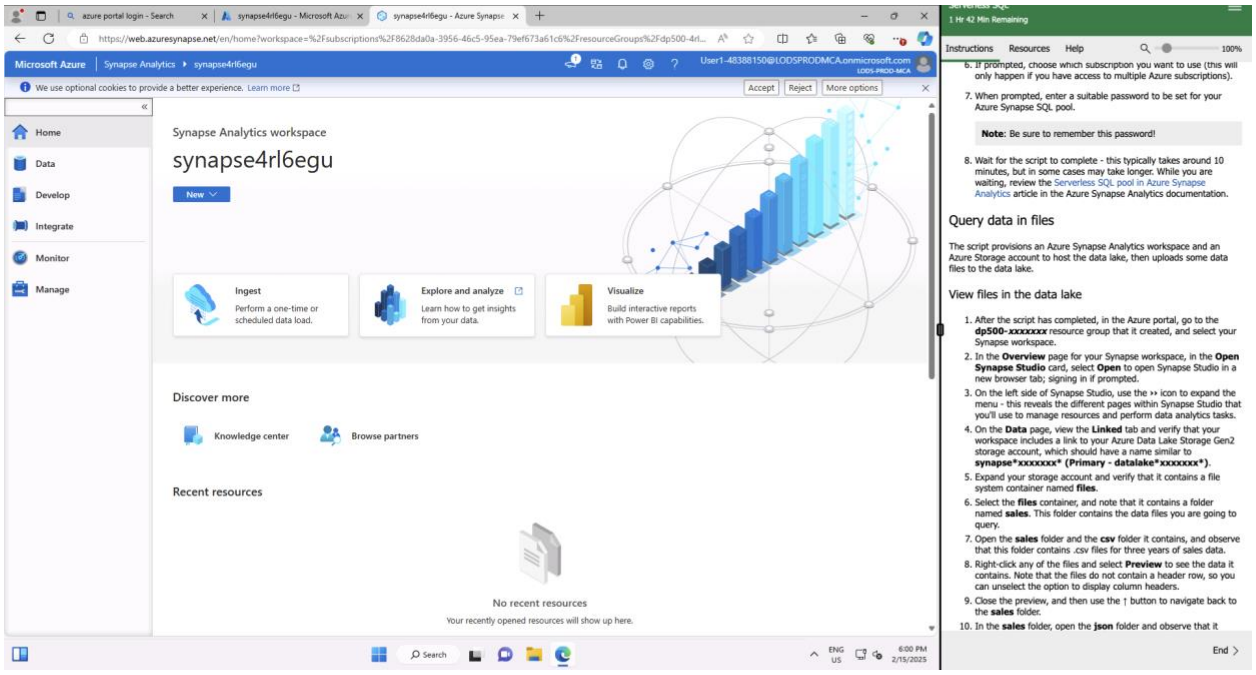
6. If prompted, choose which subscription you want to use (this will only happen if you have access to multiple Azure subscriptions).
7. When prompted, enter a suitable password to be set for your Azure Synapse SQL pool.

Note: Be sure to remember this password!

8. Wait for the script to complete - this typically takes around 10 minutes, but in some cases may take longer. While you are waiting, review the Serverless SQL pool in Azure Synapse Analytics article in the Azure Synapse Analytics documentation.

Query data in files

Opening Synapse Studio:


Microsoft Azure | Synapse Analytics › synapse4rl6egu
Home
Data
Develop
Integrate
Monitor
Manage
Synapse Analytics workspace
synapse4rl6egu
New
Ingest
Perform a one-time or scheduled data load.
Explore and analyze
Learn how to get insights from your data.
Visualize
Build interactive reports with Power BI capabilities.
Discover more
Knowledge center
Browse partners
Recent resources
No recent resources
Your recently opened resources will show up here.

Serverless SQL

1 Hr 42 Min Remaining

Instructions Resources Help 100%

6. If prompted, choose which subscription you want to use (this will only happen if you have access to multiple Azure subscriptions).
7. When prompted, enter a suitable password to be set for your Azure Synapse SQL pool.

   Note: Be sure to remember this password!

8. Wait for the script to complete - this typically takes around 10 minutes, but in some cases may take longer. While you are waiting, review the Serverless SQL pool in Azure Synapse Analytics article in the Azure Synapse Analytics documentation.

Query data in files

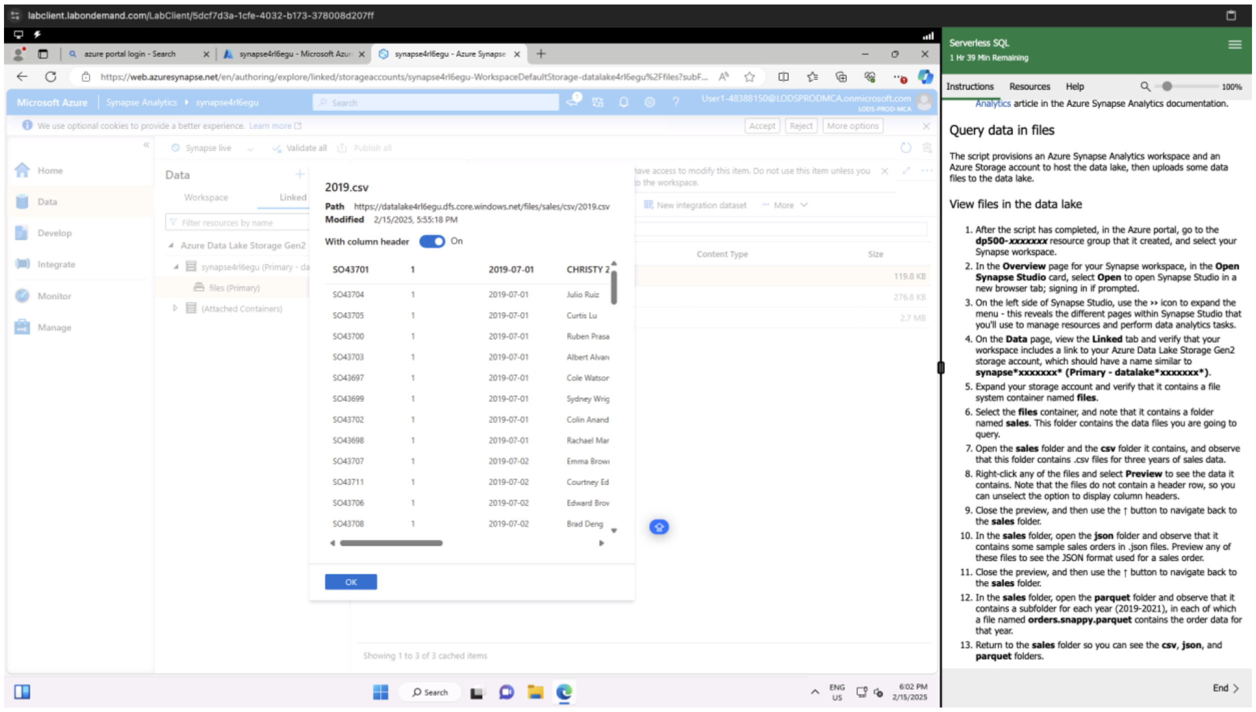
The script provisions an Azure Synapse Analytics workspace and an Azure Storage account to host the data lake, then uploads some data files to the data lake.

View files in the data lake

1. After the script has completed, in the Azure portal, go to the **dp500-*xxxxxxx*** resource group that it created, and select your Synapse workspace.
2. In the **Overview** page for your Synapse workspace, in the **Open Synapse Studio** card, select **Open** to open Synapse Studio in a new browser tab; signing in if prompted.
3. On the left side of Synapse Studio, use the ›› icon to expand the menu - this reveals the different pages within Synapse Studio that you'll use to manage resources and perform data analytics tasks.
4. On the **Data** page, view the **Linked** tab and verify that your workspace includes a link to your Azure Data Lake Storage Gen2 storage account, which should have a name similar to **synapse*xxxxxxx* (Primary - datalake*xxxxxxx*)**.
5. Expand your storage account and verify that it contains a file system container named **files**.
6. Select the **files** container, and note that it contains a folder named **sales**. This folder contains the data files you are going to query.
7. Open the **sales** folder and the **csv** folder it contains, and observe that this folder contains .csv files for three years of sales data.
8. Right-click any of the files and select **Preview** to see the data it contains. Note that the files do not contain a header row, so you can unselect the option to display column headers.
9. Close the preview, and then use the ↑ button to navigate back to the **sales** folder.
10. In the **sales** folder, open the **json** folder and observe that it

End >

Viewing the Ingested data:


labclient.labondemand.com/LabClient/5dcf7d3a-1cfe-4032-b173-378008d207ff
Microsoft Azure | Synapse Analytics › synapse4rl6egu
Data
Workspace
Linked
Azure Data Lake Storage Gen2
synapse4rl6egu (Primary - da
files (Primary)
(Attached Containers)
Showing 1 to 3 of 3 cached items

2019.csv

Path https://datalake4rl6egu.dfs.core.windows.net/files/sales/csv/2019.csv

Modified 2/15/2025, 5:55:18 PM

With column header On

SO43701	1	2019-07-01	CHRISTY Z
SO43704	1	2019-07-01	Julio Ruiz
SO43705	1	2019-07-01	Curtis Lu
SO43700	1	2019-07-01	Ruben Prasa
SO43703	1	2019-07-01	Albert Alvar
SO43697	1	2019-07-01	Cole Watsor
SO43699	1	2019-07-01	Sydney Wrig
SO43702	1	2019-07-01	Colin Anand
SO43698	1	2019-07-01	Rachael Mar
SO43707	1	2019-07-02	Emma Brow
SO43711	1	2019-07-02	Courtney Ed
SO43706	1	2019-07-02	Edward Brov
SO43708	1	2019-07-02	Brad Deng

OK

Serverless SQL

1 Hr 39 Min Remaining

Instructions Resources Help 100%

Analytics article in the Azure Synapse Analytics documentation.

Query data in files

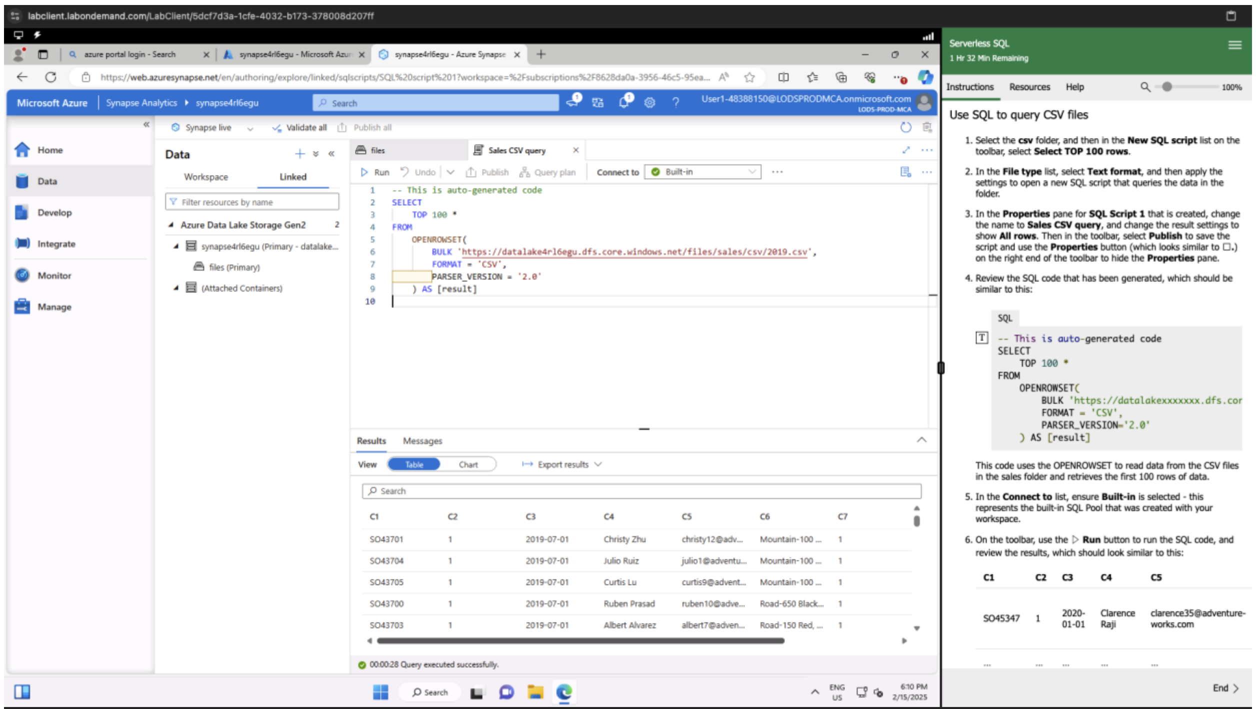
The script provisions an Azure Synapse Analytics workspace and an Azure Storage account to host the data lake, then uploads some data files to the data lake.

View files in the data lake

1. After the script has completed, in the Azure portal, go to the **dp500-*xxxxxxx*** resource group that it created, and select your Synapse workspace.
2. In the **Overview** page for your Synapse workspace, in the **Open Synapse Studio** card, select **Open** to open Synapse Studio in a new browser tab; signing in if prompted.
3. On the left side of Synapse Studio, use the ›› icon to expand the menu - this reveals the different pages within Synapse Studio that you'll use to manage resources and perform data analytics tasks.
4. On the **Data** page, view the **Linked** tab and verify that your workspace includes a link to your Azure Data Lake Storage Gen2 storage account, which should have a name similar to **synapse*xxxxxxx* (Primary - datalake*xxxxxxx*)**.
5. Expand your storage account and verify that it contains a file system container named **files**.
6. Select the **files** container, and note that it contains a folder named **sales**. This folder contains the data files you are going to query.
7. Open the **sales** folder and the **csv** folder it contains, and observe that this folder contains .csv files for three years of sales data.
8. Right-click any of the files and select **Preview** to see the data it contains. Note that the files do not contain a header row, so you can unselect the option to display column headers.
9. Close the preview, and then use the ↑ button to navigate back to the **sales** folder.
10. In the **sales** folder, open the **json** folder and observe that it contains some sample sales orders in .json files. Preview any of these files to see the JSON format used for a sales order.
11. Close the preview, and then use the ↑ button to navigate back to the **sales** folder.
12. In the **sales** folder, open the **parquet** folder and observe that it contains a subfolder for each year (2019-2021), in each of which a file named **orders.snappy.parquet** contains the order data for that year.
13. Return to the **sales** folder so you can see the **csv**, **json**, and **parquet** folders.

End >

Using SQL to query CSV files:



Sales CSV query — Connect to: Built-in

```sql
-- This is auto-generated code
SELECT
    TOP 100 *
FROM
    OPENROWSET(
        BULK 'https://datalake4rl6egu.dfs.core.windows.net/files/sales/csv/2019.csv',
        FORMAT = 'CSV',
        PARSER_VERSION = '2.0'
    ) AS [result]
```

C1	C2	C3	C4	C5	C6	C7
SO43701	1	2019-07-01	Christy Zhu	christy12@adv...	Mountain-100 ...	1
SO43704	1	2019-07-01	Julio Ruiz	julio1@adventu...	Mountain-100 ...	1
SO43705	1	2019-07-01	Curtis Lu	curtis9@advent...	Mountain-100 ...	1
SO43700	1	2019-07-01	Ruben Prasad	ruben10@adve...	Road-650 Black...	1
SO43703	1	2019-07-01	Albert Alvarez	albert7@adven...	Road-150 Red, ...	1

00:00:28 Query executed successfully.

Serverless SQL
1 Hr 32 Min Remaining

Use SQL to query CSV files

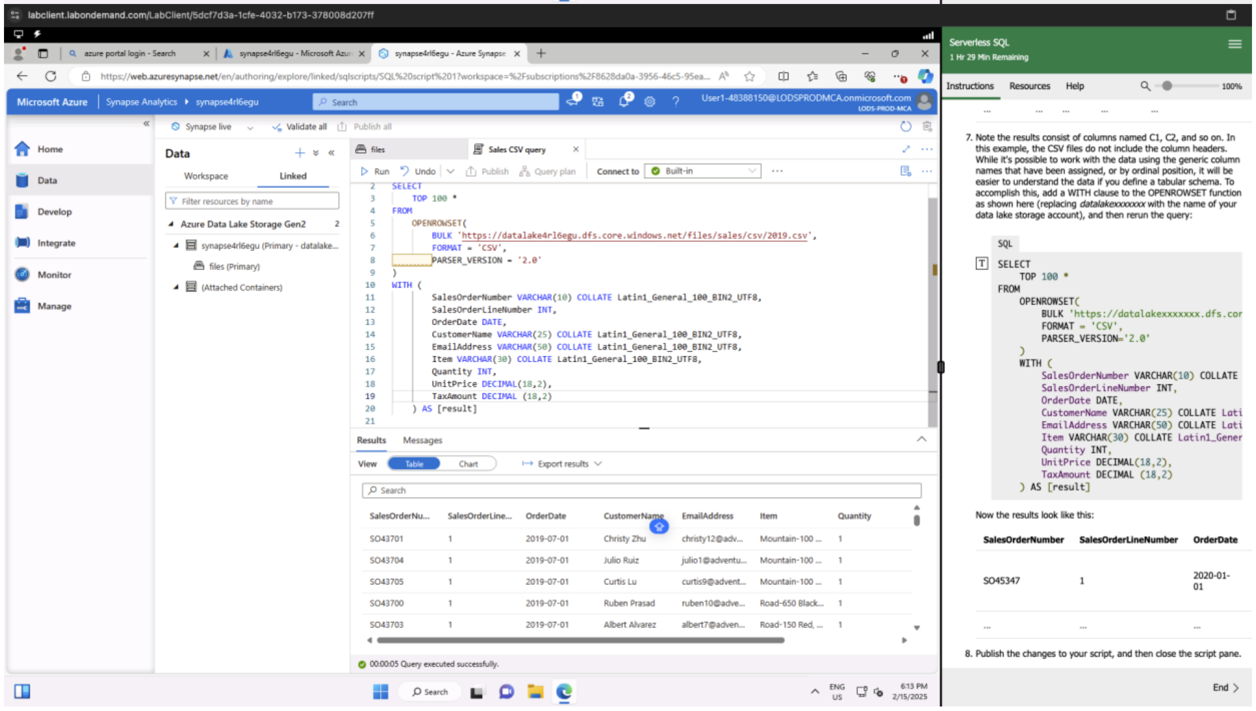
1. Select the **csv** folder, and then in the **New SQL script** list on the toolbar, select **Select TOP 100 rows**.
2. In the **File type** list, select **Text format**, and then apply the settings to open a new SQL script that queries the data in the folder.
3. In the **Properties** pane for **SQL Script 1** that is created, change the name to **Sales CSV query**, and change the result settings to show **All rows**. Then in the toolbar, select **Publish** to save the script and use the **Properties** button (which looks similar to ☐.) on the right end of the toolbar to hide the **Properties** pane.
4. Review the SQL code that has been generated, which should be similar to this:

```sql
-- This is auto-generated code
SELECT
    TOP 100 *
FROM
    OPENROWSET(
        BULK 'https://datalakexxxxxxx.dfs.cor
        FORMAT = 'CSV',
        PARSER_VERSION='2.0'
    ) AS [result]
```

This code uses the OPENROWSET to read data from the CSV files in the sales folder and retrieves the first 100 rows of data.

5. In the **Connect to** list, ensure **Built-in** is selected - this represents the built-in SQL Pool that was created with your workspace.
6. On the toolbar, use the ▷ **Run** button to run the SQL code, and review the results, which should look similar to this:

C1	C2	C3	C4	C5
SO45347	1	2020-01-01	Clarence Raji	clarence35@adventure-works.com
...	...	...	...	...



Sales CSV query — Connect to: Built-in

```sql
SELECT
    TOP 100 *
FROM
    OPENROWSET(
        BULK 'https://datalake4rl6egu.dfs.core.windows.net/files/sales/csv/2019.csv',
        FORMAT = 'CSV',
        PARSER_VERSION = '2.0'
)
WITH (
        SalesOrderNumber VARCHAR(10) COLLATE Latin1_General_100_BIN2_UTF8,
        SalesOrderLineNumber INT,
        OrderDate DATE,
        CustomerName VARCHAR(25) COLLATE Latin1_General_100_BIN2_UTF8,
        EmailAddress VARCHAR(50) COLLATE Latin1_General_100_BIN2_UTF8,
        Item VARCHAR(30) COLLATE Latin1_General_100_BIN2_UTF8,
        Quantity INT,
        UnitPrice DECIMAL(18,2),
        TaxAmount DECIMAL (18,2)
    ) AS [result]
```

SalesOrderNu...	SalesOrderLine...	OrderDate	CustomerName	EmailAddress	Item	Quantity
SO43701	1	2019-07-01	Christy Zhu	christy12@adv...	Mountain-100 ...	1
SO43704	1	2019-07-01	Julio Ruiz	julio1@adventu...	Mountain-100 ...	1
SO43705	1	2019-07-01	Curtis Lu	curtis9@advent...	Mountain-100 ...	1
SO43700	1	2019-07-01	Ruben Prasad	ruben10@adve...	Road-650 Black...	1
SO43703	1	2019-07-01	Albert Alvarez	albert7@adven...	Road-150 Red, ...	1

00:00:05 Query executed successfully.

Serverless SQL
1 Hr 29 Min Remaining

7. Note the results consist of columns named C1, C2, and so on. In this example, the CSV files do not include the column headers. While it's possible to work with the data using the generic column names that have been assigned, or by ordinal position, it will be easier to understand the data if you define a tabular schema. To accomplish this, add a WITH clause to the OPENROWSET function as shown here (replacing *datalakexxxxxxx* with the name of your data lake storage account), and then rerun the query:

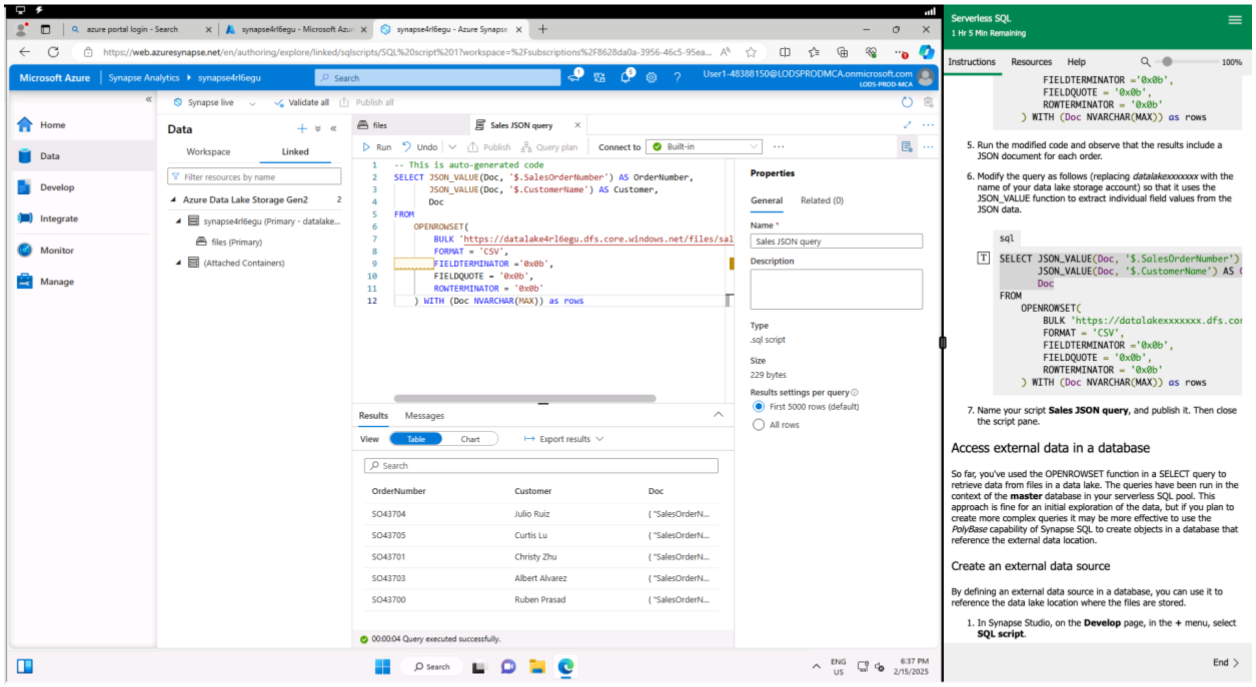
```sql
SELECT
    TOP 100 *
FROM
    OPENROWSET(
        BULK 'https://datalakexxxxxxx.dfs.cor
        FORMAT = 'CSV',
        PARSER_VERSION='2.0'
    )
    WITH (
        SalesOrderNumber VARCHAR(10) COLLATE
        SalesOrderLineNumber INT,
        OrderDate DATE,
        CustomerName VARCHAR(25) COLLATE Lati
        EmailAddress VARCHAR(50) COLLATE Lati
        Item VARCHAR(30) COLLATE Latin1_Gener
        Quantity INT,
        UnitPrice DECIMAL(18,2),
        TaxAmount DECIMAL (18,2)
    ) AS [result]
```

Now the results look like this:

SalesOrderNumber	SalesOrderLineNumber	OrderDate
SO45347	1	2020-01-01
...	...	...

8. Publish the changes to your script, and then close the script pane.

Using SQL to query JSON files:



```sql
-- This is auto-generated code
SELECT JSON_VALUE(Doc, '$.SalesOrderNumber') AS OrderNumber,
       JSON_VALUE(Doc, '$.CustomerName') AS Customer,
       Doc
FROM
    OPENROWSET(
        BULK 'https://datalake4rl6egu.dfs.core.windows.net/files/sal
        FORMAT = 'CSV',
        FIELDTERMINATOR ='0x0b',
        FIELDQUOTE = '0x0b',
        ROWTERMINATOR = '0x0b'
    ) WITH (Doc NVARCHAR(MAX)) as rows
```

Properties

Name: Sales JSON query

Type: .sql script

Size: 229 bytes

Results settings per query: [✓ First 5000 rows (default)] All rows

OrderNumber	Customer	Doc
SO43704	Julio Ruiz	{ "SalesOrderN...
SO43705	Curtis Lu	{ "SalesOrderN...
SO43701	Christy Zhu	{ "SalesOrderN...
SO43703	Albert Alvarez	{ "SalesOrderN...
SO43700	Ruben Prasad	{ "SalesOrderN...

00:00:04 Query executed successfully.

Serverless SQL
1 Hr 5 Min Remaining

```sql
        FIELDTERMINATOR ='0x0b',
        FIELDQUOTE = '0x0b',
        ROWTERMINATOR = '0x0b'
    ) WITH (Doc NVARCHAR(MAX)) as rows
```

5. Run the modified code and observe that the results include a JSON document for each order.
6. Modify the query as follows (replacing *datalakexxxxxxx* with the name of your data lake storage account) so that it uses the JSON_VALUE function to extract individual field values from the JSON data.

```sql
SELECT JSON_VALUE(Doc, '$.SalesOrderNumber')
       JSON_VALUE(Doc, '$.CustomerName') AS (
       Doc
FROM
    OPENROWSET(
        BULK 'https://datalakexxxxxxx.dfs.cor
        FORMAT = 'CSV',
        FIELDTERMINATOR ='0x0b',
        FIELDQUOTE = '0x0b',
        ROWTERMINATOR = '0x0b'
    ) WITH (Doc NVARCHAR(MAX)) as rows
```

7. Name your script **Sales JSON query**, and publish it. Then close the script pane.

Access external data in a database

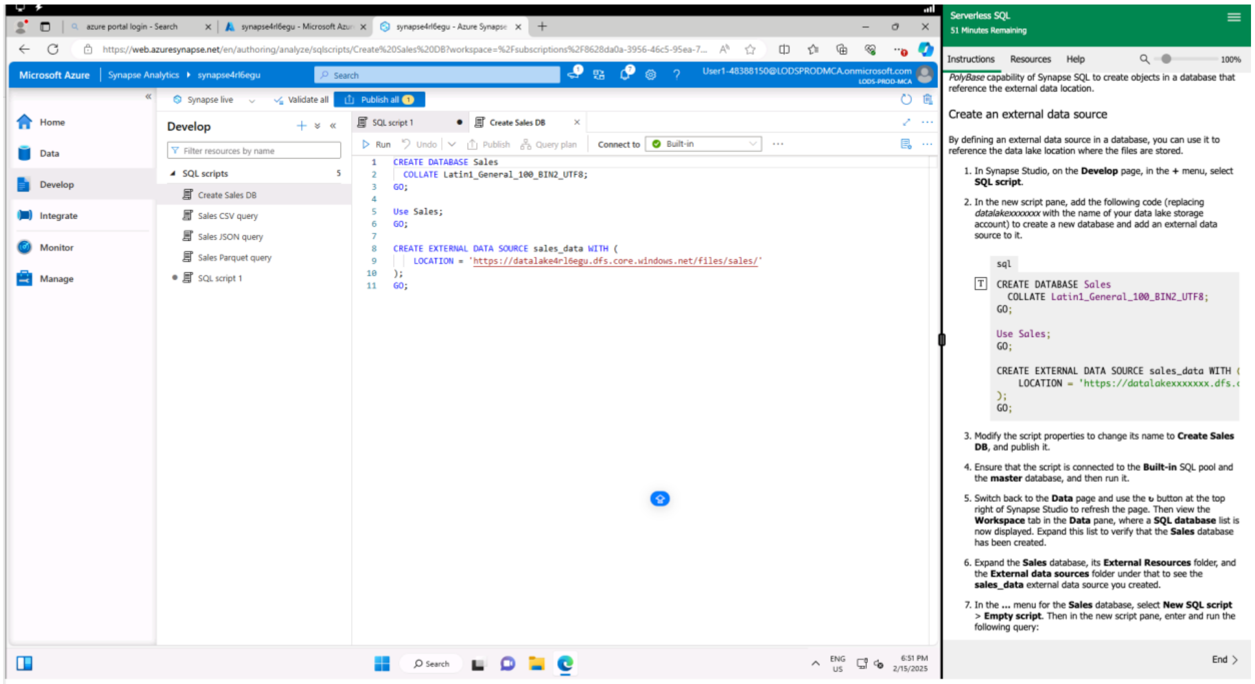
So far, you've used the OPENROWSET function in a SELECT query to retrieve data from files in a data lake. The queries have been run in the context of the **master** database in your serverless SQL pool. This approach is fine for an initial exploration of the data, but if you plan to create more complex queries it may be more effective to use the *PolyBase* capability of Synapse SQL to create objects in a database that reference the external data location.

Create an external data source

By defining an external data source in a database, you can use it to reference the data lake location where the files are stored.

1. In Synapse Studio, on the **Develop** page, in the **+** menu, select **SQL script**.

Creating External Data Source:



```sql
CREATE DATABASE Sales
  COLLATE Latin1_General_100_BIN2_UTF8;
GO;

Use Sales;
GO;

CREATE EXTERNAL DATA SOURCE sales_data WITH (
    LOCATION = 'https://datalake4rl6egu.dfs.core.windows.net/files/sales/'
);
GO;
```

Serverless SQL
51 Minutes Remaining

PolyBase capability of Synapse SQL to create objects in a database that reference the external data location.

Create an external data source

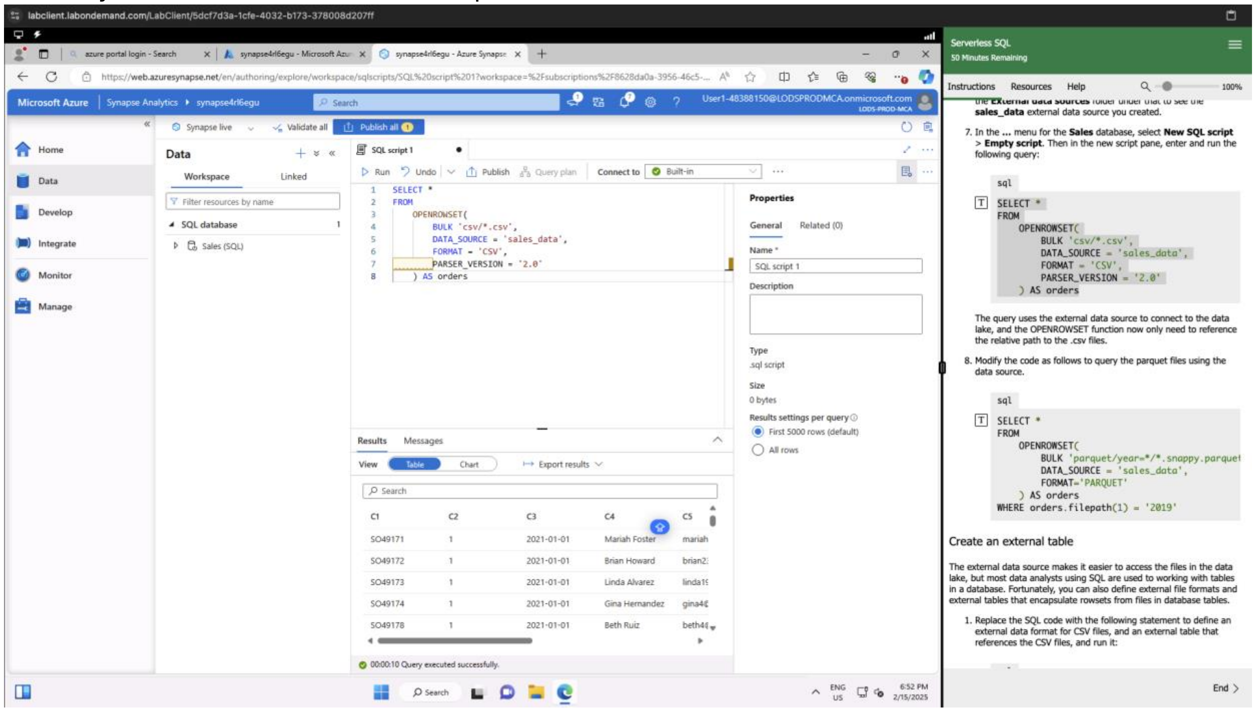
By defining an external data source in a database, you can use it to reference the data lake location where the files are stored.

1. In Synapse Studio, on the **Develop** page, in the **+** menu, select **SQL script**.
2. In the new script pane, add the following code (replacing *datalakexxxxxxx* with the name of your data lake storage account) to create a new database and add an external data source to it.

```sql
CREATE DATABASE Sales
  COLLATE Latin1_General_100_BIN2_UTF8;
GO;

Use Sales;
GO;

CREATE EXTERNAL DATA SOURCE sales_data WITH (
    LOCATION = 'https://datalakexxxxxxx.dfs.c
);
GO;
```

3. Modify the script properties to change its name to **Create Sales DB**, and publish it.
4. Ensure that the script is connected to the **Built-in** SQL pool and the **master** database, and then run it.
5. Switch back to the **Data** page and use the ↻ button at the top right of Synapse Studio to refresh the page. Then view the **Workspace** tab in the **Data** pane, where a **SQL database** list is now displayed. Expand this list to verify that the **Sales** database has been created.
6. Expand the **Sales** database, its **External Resources** folder, and the **External data sources** folder under that to see the **sales_data** external data source you created.
7. In the **...** menu for the **Sales** database, select **New SQL script** > **Empty script**. Then in the new script pane, enter and run the following query:

The query uses the external data source to connect to the data lake, and the OPENROWSET function now only need to reference the relative path to the .csv files.



```sql
SELECT *
FROM
    OPENROWSET(
        BULK 'csv/*.csv',
        DATA_SOURCE = 'sales_data',
        FORMAT = 'CSV',
        PARSER_VERSION = '2.0'
    ) AS orders
```

C1	C2	C3	C4	C5
SO49171	1	2021-01-01	Mariah Foster	mariah
SO49172	1	2021-01-01	Brian Howard	brian2:
SO49173	1	2021-01-01	Linda Alvarez	linda1!
SO49174	1	2021-01-01	Gina Hernandez	gina4(
SO49178	1	2021-01-01	Beth Ruiz	beth4(

00:00:10 Query executed successfully.

The **External data sources** folder under that to see the **sales_data** external data source you created.

7. In the ... menu for the **Sales** database, select **New SQL script** > **Empty script**. Then in the new script pane, enter and run the following query:

```sql
SELECT *
FROM
    OPENROWSET(
        BULK 'csv/*.csv',
        DATA_SOURCE = 'sales_data',
        FORMAT = 'CSV',
        PARSER_VERSION = '2.0'
    ) AS orders
```

The query uses the external data source to connect to the data lake, and the OPENROWSET function now only need to reference the relative path to the .csv files.

8. Modify the code as follows to query the parquet files using the data source.

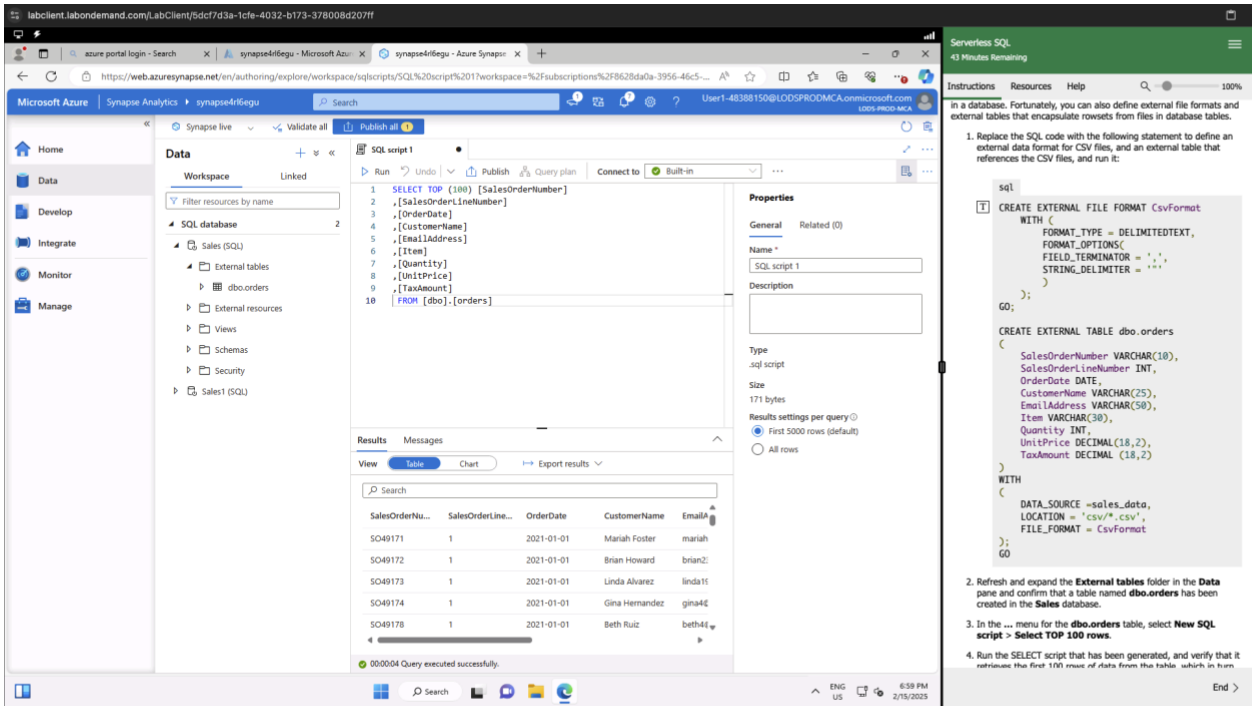
```sql
SELECT *
FROM
    OPENROWSET(
        BULK 'parquet/year=*/*.snappy.parquet
        DATA_SOURCE = 'sales_data',
        FORMAT='PARQUET'
    ) AS orders
WHERE orders.filepath(1) = '2019'
```

Create an external table

The external data source makes it easier to access the files in the data lake, but most data analysts using SQL are used to working with tables in a database. Fortunately, you can also define external file formats and external tables that encapsulate rowsets from files in database tables.

1. Replace the SQL code with the following statement to define an external data format for CSV files, and an external table that references the CSV files, and run it:

Creating an external table dbo.orders:



```sql
SELECT TOP (100) [SalesOrderNumber]
,[SalesOrderLineNumber]
,[OrderDate]
,[CustomerName]
,[EmailAddress]
,[Item]
,[Quantity]
,[UnitPrice]
,[TaxAmount]
 FROM [dbo].[orders]
```

SalesOrderNu...	SalesOrderLine...	OrderDate	CustomerName	EmailA
SO49171	1	2021-01-01	Mariah Foster	mariah
SO49172	1	2021-01-01	Brian Howard	brian2:
SO49173	1	2021-01-01	Linda Alvarez	linda1!
SO49174	1	2021-01-01	Gina Hernandez	gina4(
SO49178	1	2021-01-01	Beth Ruiz	beth4(

00:00:04 Query executed successfully.

in a database. Fortunately, you can also define external file formats and external tables that encapsulate rowsets from files in database tables.

1. Replace the SQL code with the following statement to define an external data format for CSV files, and an external table that references the CSV files, and run it:

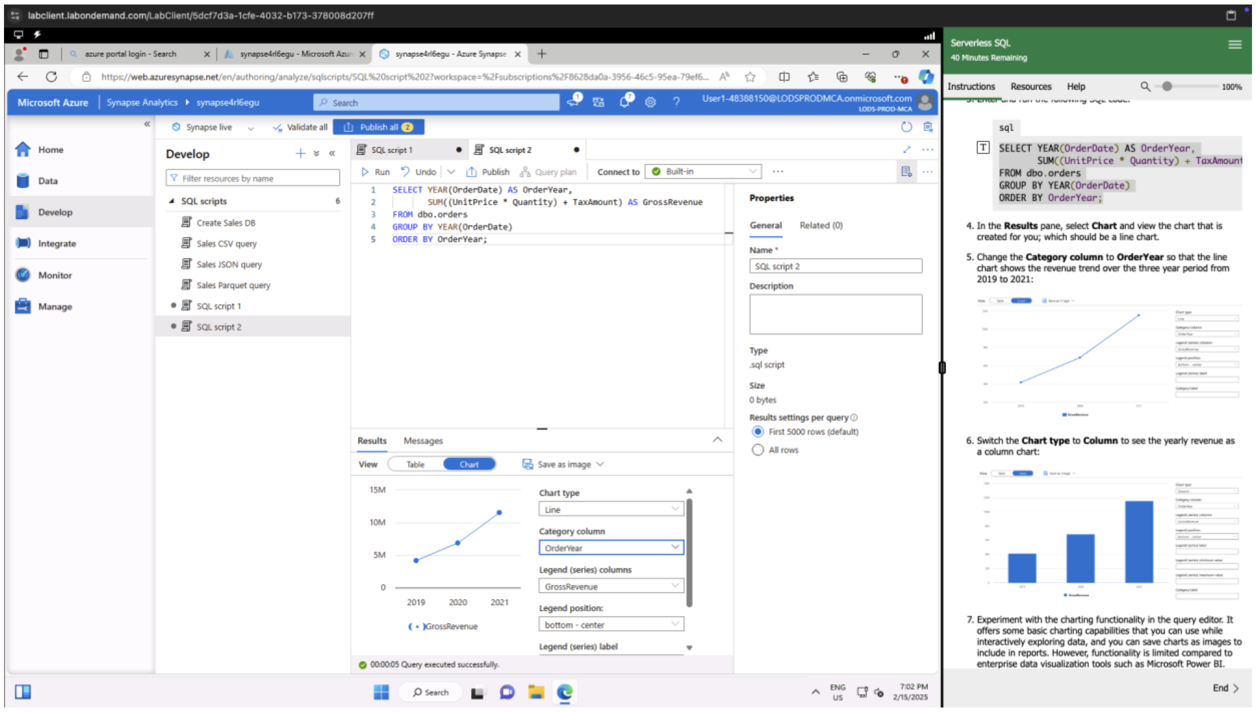
```sql
CREATE EXTERNAL FILE FORMAT CsvFormat
    WITH (
        FORMAT_TYPE = DELIMITEDTEXT,
        FORMAT_OPTIONS(
        FIELD_TERMINATOR = ',',
        STRING_DELIMITER = '"'
        )
    );
GO;

CREATE EXTERNAL TABLE dbo.orders
(
    SalesOrderNumber VARCHAR(10),
    SalesOrderLineNumber INT,
    OrderDate DATE,
    CustomerName VARCHAR(25),
    EmailAddress VARCHAR(50),
    Item VARCHAR(30),
    Quantity INT,
    UnitPrice DECIMAL(18,2),
    TaxAmount DECIMAL (18,2)
)
WITH
(
    DATA_SOURCE =sales_data,
    LOCATION = 'csv/*.csv',
    FILE_FORMAT = CsvFormat
);
GO
```

2. Refresh and expand the **External tables** folder in the **Data** pane and confirm that a table named **dbo.orders** has been created in the **Sales** database.

3. In the ... menu for the **dbo.orders** table, select **New SQL script** > **Select TOP 100 rows**.

4. Run the SELECT script that has been generated, and verify that it retrieves the first 100 rows of data from the table, which in turn

Visualizing Query Results



labclient.labondemand.com/LabClient/5dcf7d3a-1cfe-4032-b173-378008d207ff

https://web.azuresynapse.net/en/authoring/analyze/sqlscripts/SQL%20script%202?workspace=%2Fsubscriptions%2F8628da0a-3956-46c5-95ea-79ef6...

Microsoft Azure | Synapse Analytics › synapse4rl6egu

User1-48388150@LODSPRODMCA.onmicrosoft.com
LODS-PROD-MCA

Develop

SQL scripts 6
- Create Sales DB
- Sales CSV query
- Sales JSON query
- Sales Parquet query
- SQL script 1
- SQL script 2

SQL script 1 | SQL script 2

Connect to: Built-in

```sql
SELECT YEAR(OrderDate) AS OrderYear,
       SUM((UnitPrice * Quantity) + TaxAmount) AS GrossRevenue
FROM dbo.orders
GROUP BY YEAR(OrderDate)
ORDER BY OrderYear;
```

Properties

General | Related (0)

Name *: SQL script 2

Description

Type: .sql script

Size: 0 bytes

Results settings per query: (●) First 5000 rows (default) ( ) All rows

Results | Messages

View: Table | Chart

Chart type: Line

Category column: OrderYear

Legend (series) columns: GrossRevenue

Legend position: bottom - center

Legend (series) label

00:00:05 Query executed successfully.

7:02 PM 2/15/2025

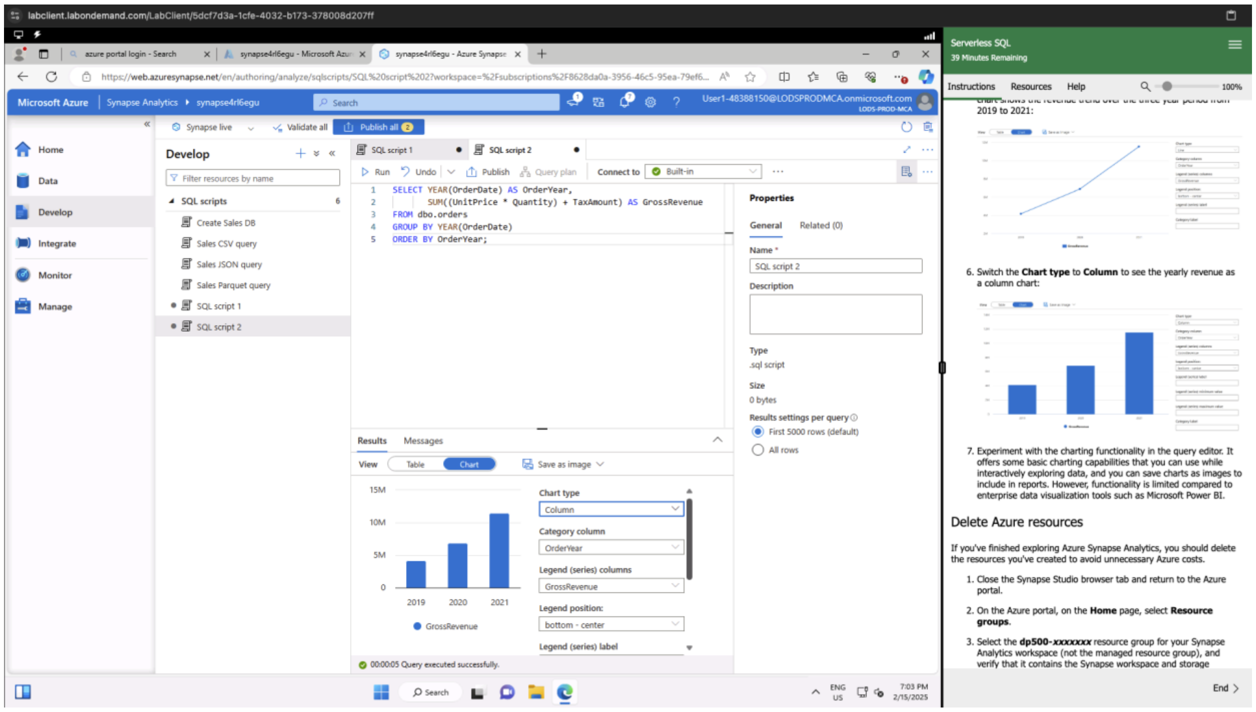
Serverless SQL
40 Minutes Remaining

Instructions | Resources | Help

```sql
SELECT YEAR(OrderDate) AS OrderYear,
       SUM((UnitPrice * Quantity) + TaxAmount
FROM dbo.orders
GROUP BY YEAR(OrderDate)
ORDER BY OrderYear;
```

4. In the **Results** pane, select **Chart** and view the chart that is created for you; which should be a line chart.
5. Change the **Category column** to **OrderYear** so that the line chart shows the revenue trend over the three year period from 2019 to 2021:
6. Switch the **Chart type** to **Column** to see the yearly revenue as a column chart:
7. Experiment with the charting functionality in the query editor. It offers some basic charting capabilities that you can use while interactively exploring data, and you can save charts as images to include in reports. However, functionality is limited compared to enterprise data visualization tools such as Microsoft Power BI.

End



labclient.labondemand.com/LabClient/5dcf7d3a-1cfe-4032-b173-378008d207ff

https://web.azuresynapse.net/en/authoring/analyze/sqlscripts/SQL%20script%202?workspace=%2Fsubscriptions%2F8628da0a-3956-46c5-95ea-79ef6...

Microsoft Azure | Synapse Analytics › synapse4rl6egu

User1-48388150@LODSPRODMCA.onmicrosoft.com
LODS-PROD-MCA

Develop

SQL scripts 6
- Create Sales DB
- Sales CSV query
- Sales JSON query
- Sales Parquet query
- SQL script 1
- SQL script 2

SQL script 1 | SQL script 2

Connect to: Built-in

```sql
SELECT YEAR(OrderDate) AS OrderYear,
       SUM((UnitPrice * Quantity) + TaxAmount) AS GrossRevenue
FROM dbo.orders
GROUP BY YEAR(OrderDate)
ORDER BY OrderYear;
```

Properties

General | Related (0)

Name *: SQL script 2

Description

Type: .sql script

Size: 0 bytes

Results settings per query: (●) First 5000 rows (default) ( ) All rows

Results | Messages

View: Table | Chart

Chart type: Column

Category column: OrderYear

Legend (series) columns: GrossRevenue

Legend position: bottom - center

Legend (series) label

00:00:05 Query executed successfully.

7:03 PM 2/15/2025

Serverless SQL
39 Minutes Remaining

Instructions | Resources | Help

2019 to 2021:

6. Switch the **Chart type** to **Column** to see the yearly revenue as a column chart:
7. Experiment with the charting functionality in the query editor. It offers some basic charting capabilities that you can use while interactively exploring data, and you can save charts as images to include in reports. However, functionality is limited compared to enterprise data visualization tools such as Microsoft Power BI.

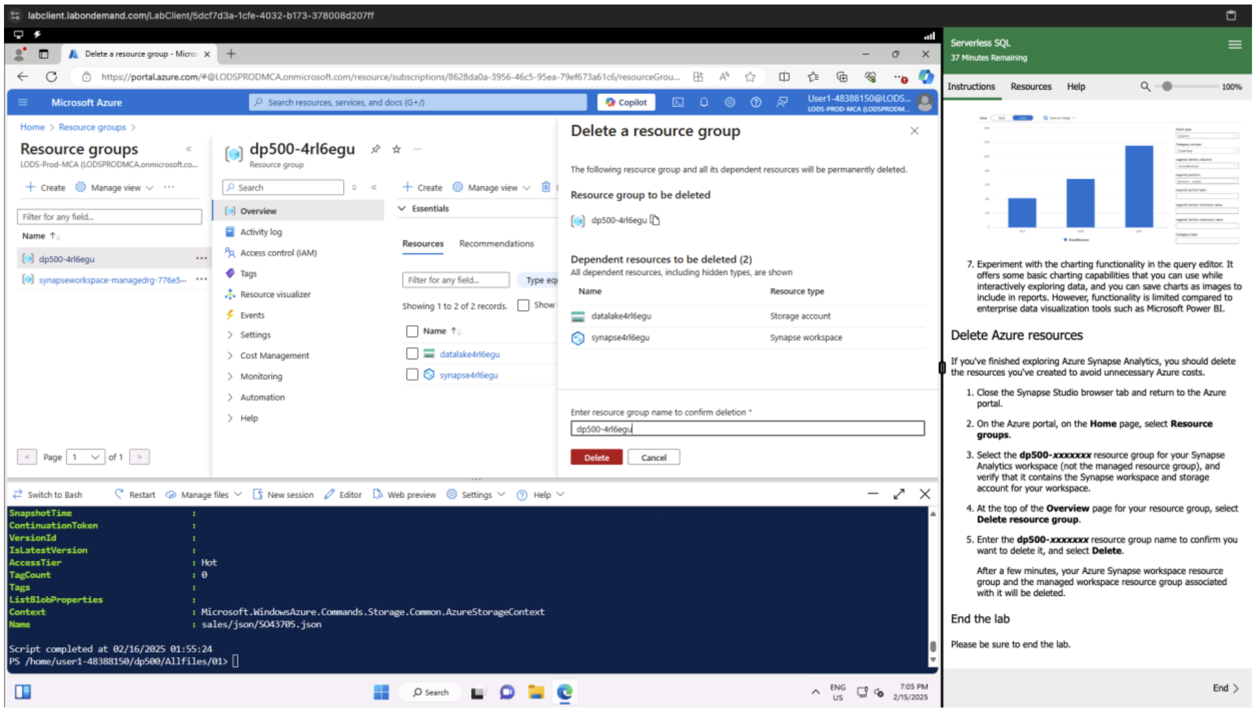
Delete Azure resources

If you've finished exploring Azure Synapse Analytics, you should delete the resources you've created to avoid unnecessary Azure costs.

1. Close the Synapse Studio browser tab and return to the Azure portal.
2. On the Azure portal, on the **Home** page, select **Resource groups**.
3. Select the **dp500-*xxxxxxx*** resource group for your Synapse Analytics workspace (not the managed resource group), and verify that it contains the Synapse workspace and storage

End

Now Deleting Azure Resources:



Delete a resource group

The following resource group and all its dependent resources will be permanently deleted.

Resource group to be deleted

dp500-4rl6egu

Dependent resources to be deleted (2)

All dependent resources, including hidden types, are shown

Name	Resource type
datalake4rl6egu	Storage account
synapse4rl6egu	Synapse workspace

Enter resource group name to confirm deletion *

dp500-4rl6egu

```
SnapshotTime                 :
ContinuationToken            :
VersionId                    :
IsLatestVersion              :
AccessTier                   : Hot
TagCount                     : 0
Tags                         :
ListBlobProperties           :
Context                      : Microsoft.WindowsAzure.Commands.Storage.Common.AzureStorageContext
Name                         : sales/json/SO43705.json

Script completed at 02/16/2025 01:55:24
PS /home/user1-48388150/dp500/Allfiles/01>
```

7. Experiment with the charting functionality in the query editor. It offers some basic charting capabilities that you can use while interactively exploring data, and you can save charts as images to include in reports. However, functionality is limited compared to enterprise data visualization tools such as Microsoft Power BI.

Delete Azure resources

If you've finished exploring Azure Synapse Analytics, you should delete the resources you've created to avoid unnecessary Azure costs.

1. Close the Synapse Studio browser tab and return to the Azure portal.
2. On the Azure portal, on the **Home** page, select **Resource groups**.
3. Select the **dp500-*xxxxxxx*** resource group for your Synapse Analytics workspace (not the managed resource group), and verify that it contains the Synapse workspace and storage account for your workspace.
4. At the top of the **Overview** page for your resource group, select **Delete resource group**.
5. Enter the **dp500-*xxxxxxx*** resource group name to confirm you want to delete it, and select **Delete**.

   After a few minutes, your Azure Synapse workspace resource group and the managed workspace resource group associated with it will be deleted.

End the lab

Please be sure to end the lab.

Conclusion:

In this lab, I explored the capabilities of Serverless SQL in Azure by querying CSV, JSON and Parquet files directly from a data lake using SQL. I learned how to create an external data source to efficiently connect to structured and semi-structured data without traditional database storage. The use of the OPENROWSET function simplified querying external files by referencing their relative paths. Additionally, I created external tables, such as dbo.orders, which allowed for more structured querying and visualization of results. Finally, I practiced resource management by deleting Azure resources to ensure cost efficiency. Overall, this lab provided hands-on experience with Serverless SQL, enhancing my understanding of querying data in a scalable and cost-effective manner.