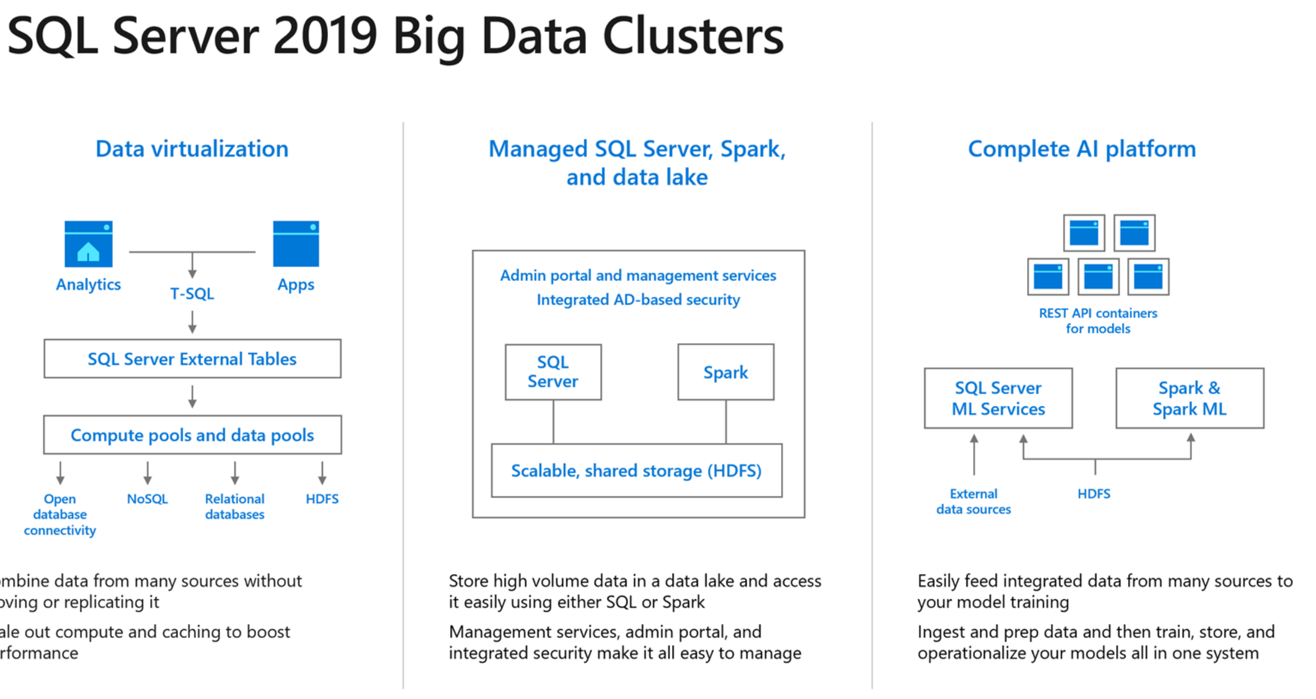
Data & AI Tech Immersion Workshop – Product Review Guide and Lab Instructions

Data, Experience 2 - Handling Big Data with SQL Server 2019 Big Data Clusters

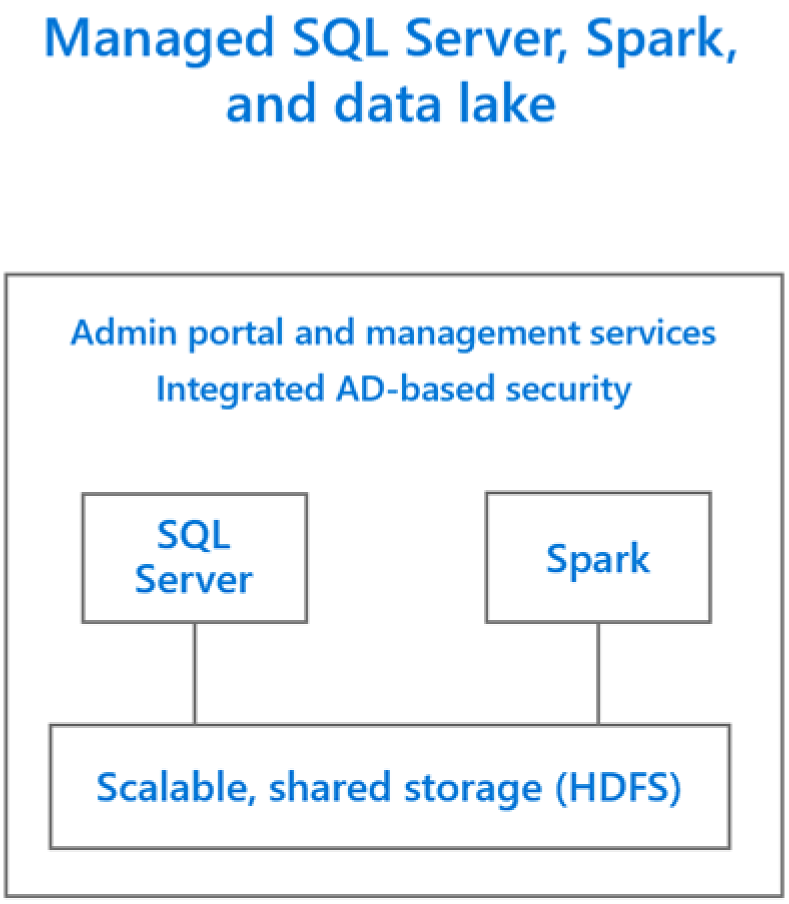
Technology overview


SQL Server 2019 Big Data Clusters
Data virtualization
Analytics
T-SQL
Apps
SQL Server External Tables
Compute pools and data pools
Open database connectivity
NoSQL
Relational databases
HDFS
Combine data from many sources without moving or replicating it
Scale out compute and caching to boost performance
Managed SQL Server, Spark, and data lake
Admin portal and management services
Integrated AD-based security
SQL Server
Spark
Scalable, shared storage (HDFS)
Store high volume data in a data lake and access it easily using either SQL or Spark
Management services, admin portal, and integrated security make it all easy to manage
Complete AI platform
REST API containers for models
SQL Server ML Services
Spark & Spark ML
External data sources
HDFS
Easily feed integrated data from many sources to your model training
Ingest and prep data and then train, store, and operationalize your models all in one system

SQL Server 2019 brings innovative security and compliance features, industry leading performance, mission-critical availability, and advanced analytics to all data workloads, now with support for big data built-in.

SQL Server 2019 is a hub for data integration. Data virtualization allows queries across relational and non-relational data without movement or replication. The enhanced PolyBase feature of SQL Server 2019 is able to connect to Hadoop clusters, Oracle, Teradata, MongoDB, and more.

Customers will be able to deliver transformational insights over structured and unstructured data with the power of SQL Server, Hadoop and Spark. SQL Server 2019 big data clusters offer scalable compute and storage composed of SQL Server, Spark and HDFS. Big data clusters will also cache data in scale-out data marts.


Managed SQL Server, Spark, and data lake
Admin portal and management services
Integrated AD-based security
SQL Server
Spark
Scalable, shared storage (HDFS)

SQL Server 2019 Big Data Clusters include a storage pool that consists of storage nodes comprised of SQL Server on Linux, Spark, and HDFS. All the storage nodes in a SQL big data cluster are members of a scalable HDFS cluster. These components of the storage pool can be combined to create a data lake to store big data, potentially ingested from multiple, disparate external sources. This data can be either structured or unstructured data. Once the big data is stored in HDFS in the big data cluster, you can analyze and query the data and combine it with your relational data.

SQL Server 2019 is a complete AI platform to train and operationalize R and Python models in SQL Server Machine Learning Services or Spark ML using Azure Data Studio notebooks.

SQL Server 2019 will give customers and ISVs the choice of programming language and platform. They will be able to build modern applications with innovative features using .NET, PHP, Node.JS, Java, Python, Ruby, and more – and deploy the application on either Windows, Linux, or containers both on-premises and in the cloud. Application developers are now able to run Java code on SQL Server and store and analyze graph data.

SQL Server 2019 allows customers to run real-time analytics on operational data using HTAP (Hybrid Transactional and Analytical Processing), leverage the in-memory technologies for faster transactions and analytical queries, and get higher concurrency and scale through persistent memory.

Scenario overview

Contoso Auto stores data in several data stores, including relational databases, NoSQL databases, data warehouses, and unstructured data stored in a data lake. They have heard of data virtualization in SQL Server 2019, and are interested to see whether this feature will allow them to more easily access their data stored in these disparate locations. They have heard of the new Big Data Clusters that can be scaled out to handle their Big Data workloads, including machine learning tasks and advanced analytics. They are also interested in any performance improvements against their internal SQL tables by moving to 2019, since the overall amount of data is growing at a rapid pace.

This experience will highlight the new features of SQL Server 2019 with a focus on Big Data Clusters and data virtualization. You will gain hands-on experience with querying both structured and unstructured data in a unified way using T-SQL. This capability will be illustrated by joining different data sets, such as product stock data in flat CSV files in Azure Storage, product reviews stored in Azure SQL Database, and transactional data in SQL Server 2019 for exploratory data analysis within Azure Data Studio. This joined data will be prepared into a table used for reporting, highlighting query performance against this table due to intelligent query processing. With the inclusion of Apache Spark packaged with Big Data Clusters, it is now possible to use Spark to train machine learning models over data lakes and use those models in SQL Server in one system. You will learn how to use Azure Data Studio to work with Jupyter notebooks to train a simple model that can predict vehicle battery lifetime, score new data and save the result as an external table. Finally, you will experience the data security and compliance features provided by SQL Server 2019 by using the Data Discovery & Classification tool in SSMS to identify tables and columns with PII and GDPR-related compliance issues, then address the issues by layering on dynamic data masking to identified columns.

Experience requirements

Before you begin this lab, you need to find the following information on the Tech Immersion Mega Data & AI Workshop On Demand Lab environment details page, or the document provided to you for this experience:

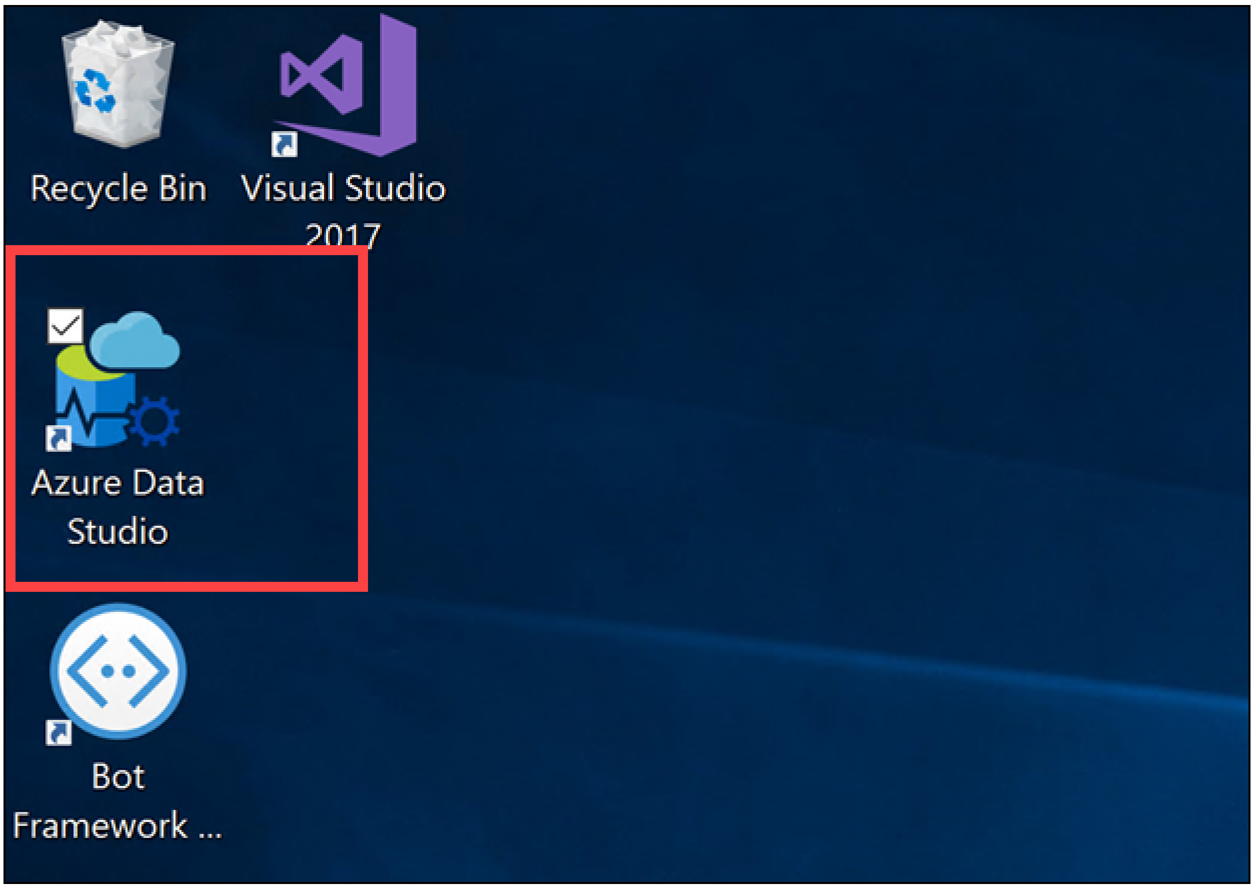
- SQL Server 2019 Big Data Cluster IP address and port number: `SQL SERVER_2019_CLUSTER URL`
- SQL username: `SQL 2019 Big Data Cluster username`
- SQL password: `SQL 2019 Big Data Cluster password`
- Sales database name (your unique copy): `SALES DB`
- Azure SQL Database server: `AZURE DATABASE SERVER`
- Azure SQL Database name: `DATABASE NAME`
- Azure SQL Database username: `DATABASE USER`
- Azure SQL Database password: `DATABASE PASSWORD`

Before the lab: Connecting to SQL Server 2019

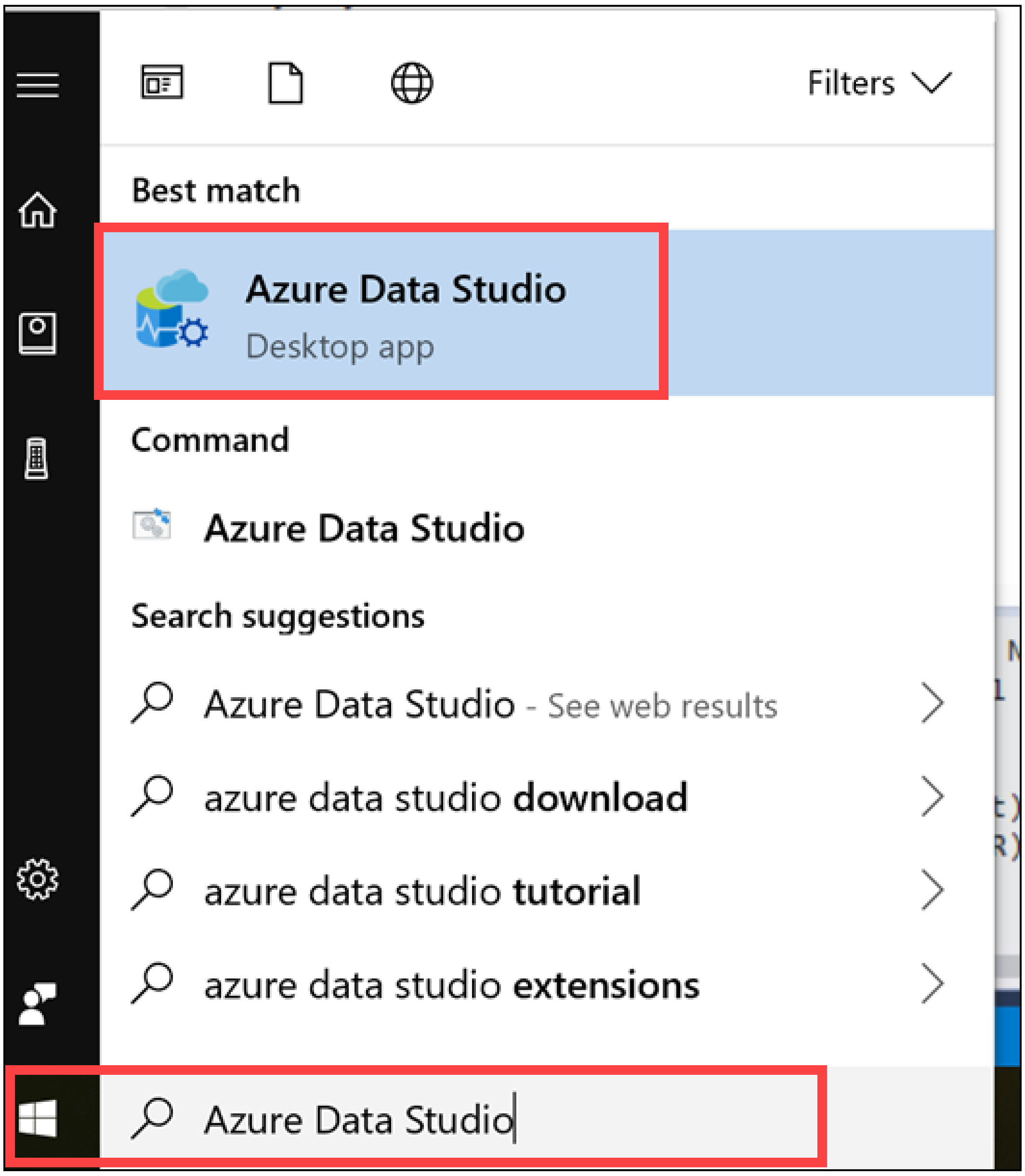
Follow the steps below to connect to your SQL Server 2019 cluster with both Azure Data Studio and SQL Server Management Studio (SSMS).

Connect with Azure Data Studio

A link to Azure Data Studio should already be on the desktop of the VM. If not, follow the instructions in Step 1 below.

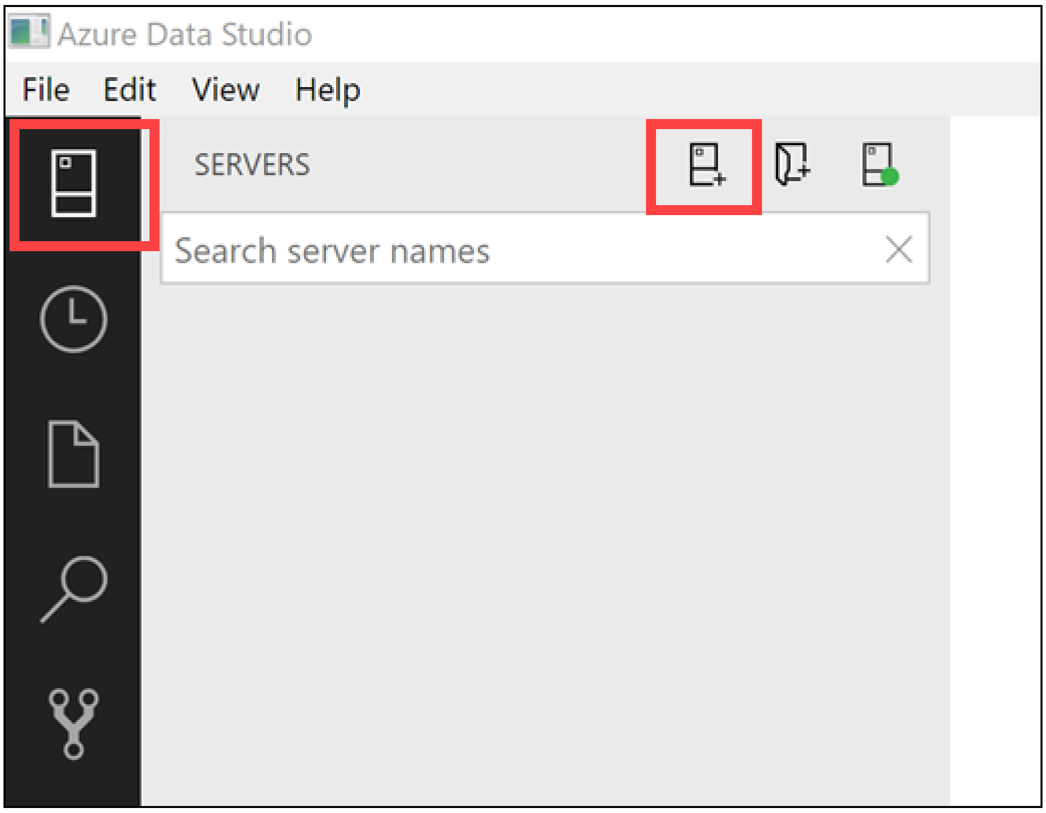


1. On the bottom-left corner of your Windows desktop, locate the search box next to the Start Menu. Type **Azure Data Studio**, then select the Azure Data Studio desktop app in the search results.


Filters
Best match
Azure Data Studio
Desktop app
Command
Azure Data Studio
Search suggestions
Azure Data Studio - See web results
azure data studio download
azure data studio tutorial
azure data studio extensions
Azure Data Studio

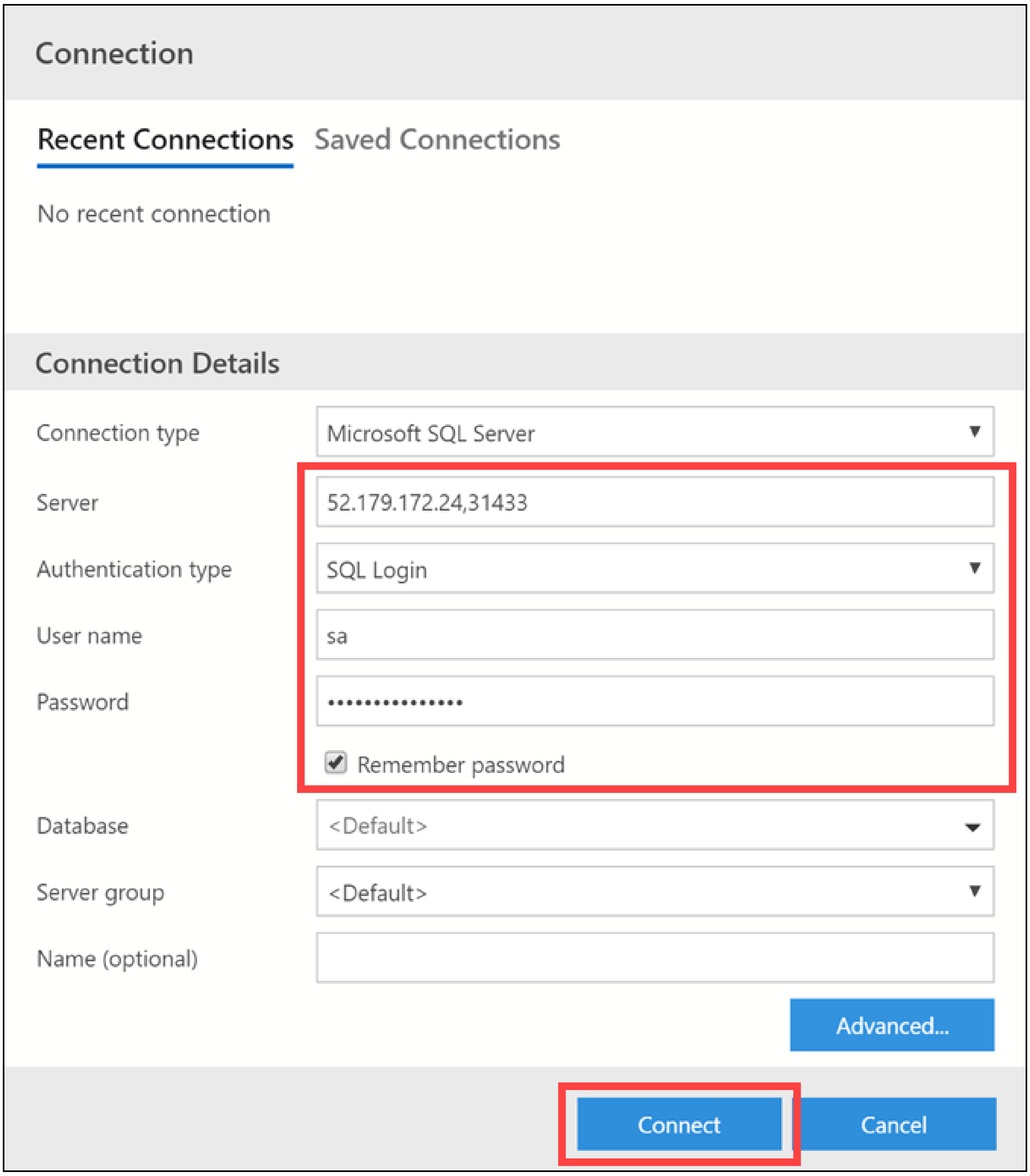
> **Please note:** If Azure Data Studio prompts you to update, please **do not apply** the update at this time. The lab has been tested with the software and library versions loaded in the provided environment.

2. Within Azure Data Studio, select **Servers** from the top of the left-hand menu, then select **New Connection** from the top toolbar to the right of the menu.



3. Within the Connection dialog, configure the following:

- **Connection type:** Select Microsoft SQL Server.
- **Server:** Enter the IP address, followed by port number `31433` to the SQL Server 2019 Big Data cluster. Use the value from the `SQL SERVER_2019_CLUSTER URL` for this from the environment documentation. It should have a format of IP separated by a comma from the port, such as: `11.122.133.144,31433`.
- **Authentication type:** Select SQL Login.
- **Username:** Enter `sa`.
- **Password:** Enter the password provided to you for this lab, you can find this value documented as `SQL 2019 Big Data Cluster password`.
- **Remember password:** Checked.
- Leave all other options at their default values.


Connection

Recent Connections | Saved Connections

No recent connection

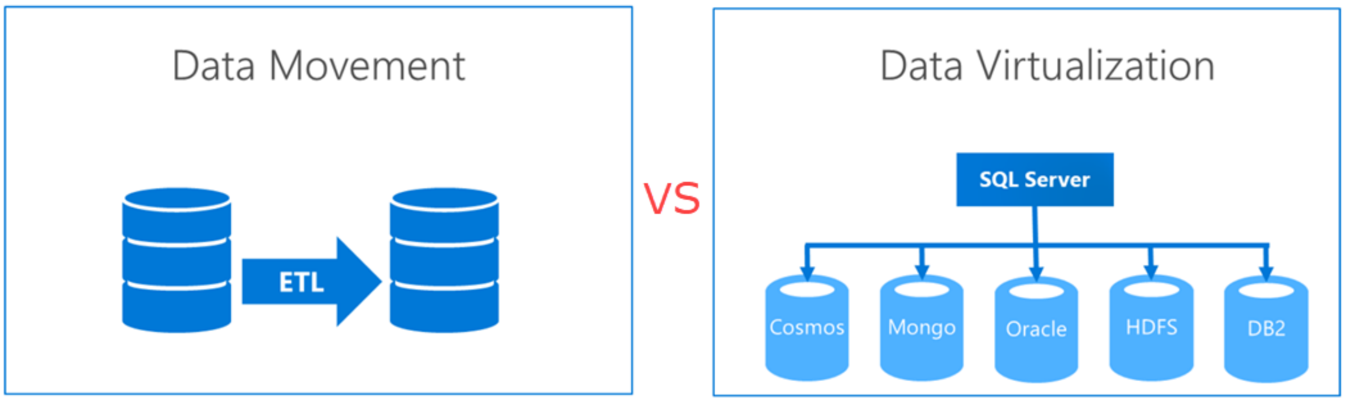
Connection Details

Connection type	Microsoft SQL Server
Server	52.179.172.24,31433
Authentication type	SQL Login
User name	sa
Password	•••••••••••••••
	✓ Remember password
Database	<Default>
Server group	<Default>
Name (optional)	

4. Click **Connect**.

Task 1: Query and join data from flat files, data from external database systems, and SQL Server

One of the key new features of SQL Server 2019 is data virtualization. What this means is that you can *virtualize* external data in a SQL Server instance, regardless of source, location, and format, so that it can be queried like any other table, or sets of tables, within your SQL Server instance. In essence, data virtualization helps you create a single "virtual" layer of data from these disparate sources, providing unified data services to support multiple applications and users. A more familiar term we could use is data lake, or perhaps data hub. Unlike a typical data lake, however, you do not have to move data out from where it lives, yet you can still query that data through a consistent interface. This is a huge advantage over traditional ETL (extract-transform-load) processes where data must be moved from its original source to a new destination, oftentimes with some data transformation or mapping. This causes delays, extra storage, additional security, and a fair amount of engineering in most cases. With data virtualization, no data movement is required, which means the data sets are up-to-date, and it is possible to query and join these different data sources through these new capabilities, thanks to the use of new PolyBase connectors. The data sources you can connect to include Cosmos DB, SQL Server (including Azure SQL Database), Oracle, HDFS (for flat files), and DB2.



The image to the left represents traditional data movement using ETL. Compare that to data virtualization, which does not require data movement and provides a unified layer over top of existing data sources.

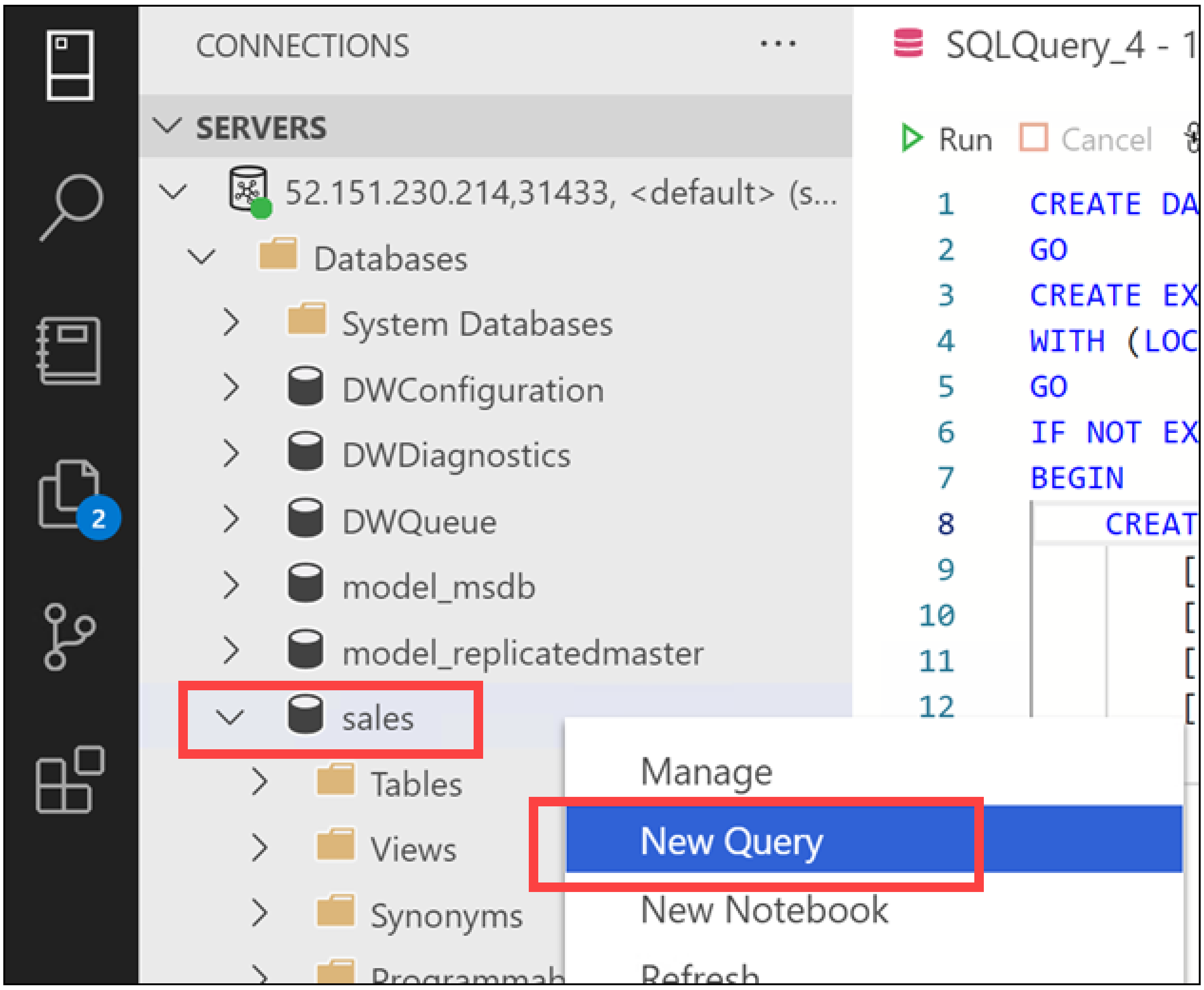
In this task, you will experience how to configure data virtualization in SQL Server 2019 by joining data in a single query from a SQL Server 2019 table, an external file stored in HDFS (Hadoop Filesystem), and an external database.

Learn more about using data virtualization with relational data sources and CSV files, using the External Table Wizard.

To start, we will use the External Table Wizard in Azure Data Studio to connect to an external Azure SQL Database.

1. Open Azure Data Studio and connect to your SQL Server 2019 cluster, following the connection steps above.

2. Expand the Databases folder, right-click on the **sales_YOUR_UNIQUE_IDENTIFIER** database, then select **Create New Query**. The `YOUR_UNIQUE_IDENTIFIER` portion of the name is the unique identifier assigned to you for this lab.

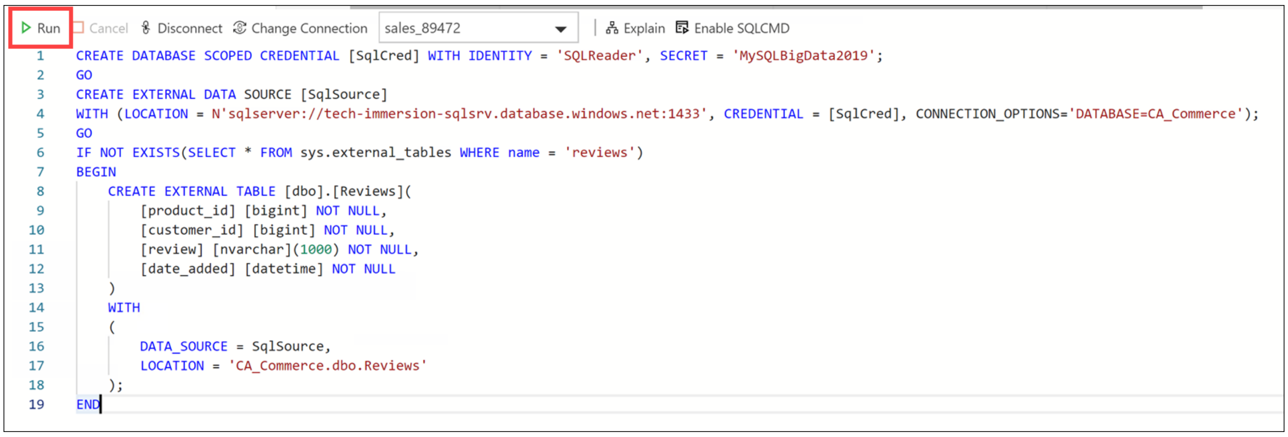


```
CREATE DA
GO
CREATE EX
WITH (LOC
GO
IF NOT EX
BEGIN
    CREAT
        [
        [
        [
        [
```

3. Paste the following script into the query window:

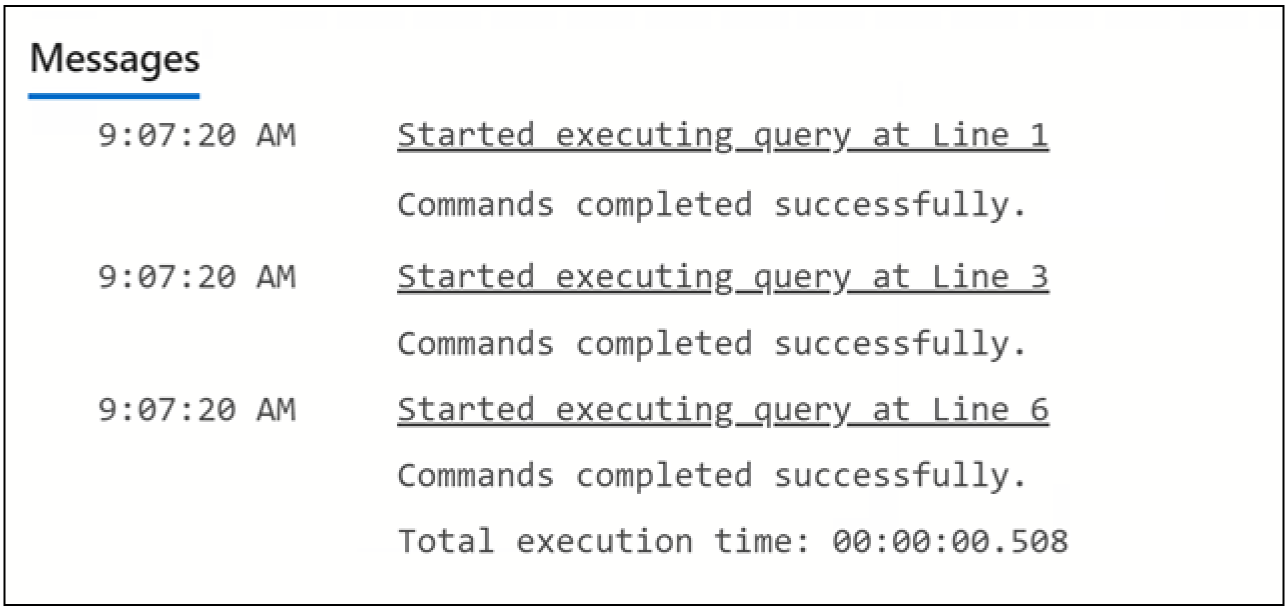
```
CREATE DATABASE SCOPED CREDENTIAL [SqlCred] WITH IDENTITY = 'SQLReader',
SECRET = 'MySQLBigData2019';
GO
CREATE EXTERNAL DATA SOURCE [SqlSource]
WITH (LOCATION = N'sqlserver://tech-immersion-
sqlsrv.database.windows.net:1433', CREDENTIAL = [SqlCred],
CONNECTION_OPTIONS='DATABASE=CA_Commerce');
GO
IF NOT EXISTS(SELECT * FROM sys.external_tables WHERE name = 'reviews')
BEGIN
    CREATE EXTERNAL TABLE [dbo].[Reviews](
        [product_id] [bigint] NOT NULL,
        [customer_id] [bigint] NOT NULL,
        [review] [nvarchar](1000) NOT NULL,
        [date_added] [datetime] NOT NULL
    )
    WITH
    (
        DATA_SOURCE = SqlSource,
        LOCATION = 'CA_Commerce.dbo.Reviews'
    );
END
```

4. Click **Run** at the top of the query window to execute:



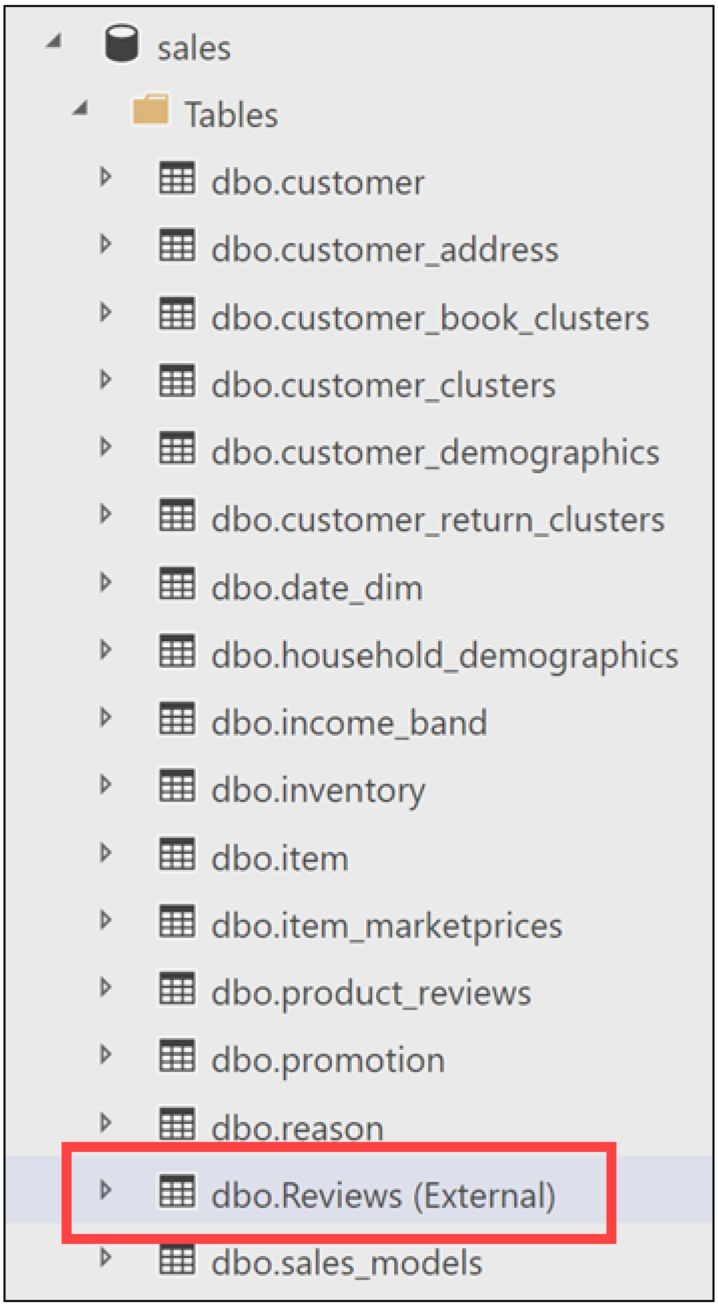
```
CREATE DATABASE SCOPED CREDENTIAL [SqlCred] WITH IDENTITY = 'SQLReader', SECRET = 'MySQLBigData2019';
GO
CREATE EXTERNAL DATA SOURCE [SqlSource]
WITH (LOCATION = N'sqlserver://tech-immersion-sqlsrv.database.windows.net:1433', CREDENTIAL = [SqlCred], CONNECTION_OPTIONS='DATABASE=CA_Commerce');
GO
IF NOT EXISTS(SELECT * FROM sys.external_tables WHERE name = 'reviews')
BEGIN
    CREATE EXTERNAL TABLE [dbo].[Reviews](
        [product_id] [bigint] NOT NULL,
        [customer_id] [bigint] NOT NULL,
        [review] [nvarchar](1000) NOT NULL,
        [date_added] [datetime] NOT NULL
    )
    WITH
    (
        DATA_SOURCE = SqlSource,
        LOCATION = 'CA_Commerce.dbo.Reviews'
    );
END
```

5. After a few moments, you will see three messages stating that the commands completed successfully.

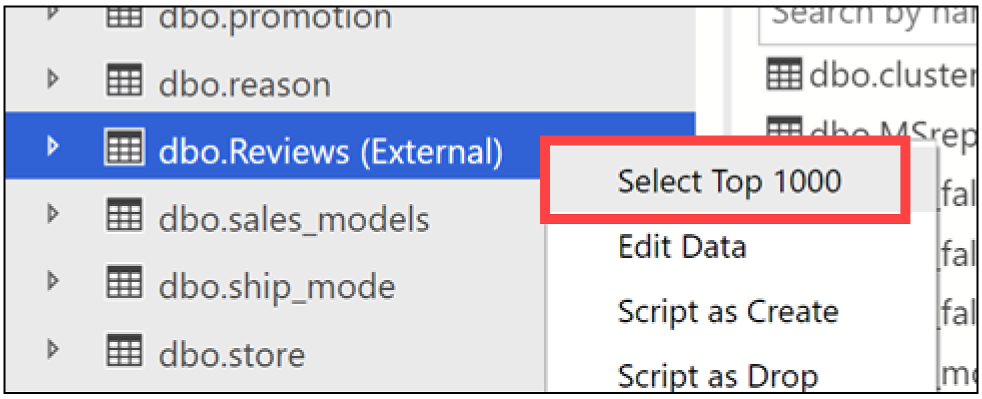


```
9:07:20 AM     Started executing query at Line 1
               Commands completed successfully.
9:07:20 AM     Started executing query at Line 3
               Commands completed successfully.
9:07:20 AM     Started executing query at Line 6
               Commands completed successfully.
               Total execution time: 00:00:00.508
```

6. Select the Servers link (Ctrl+G) on the left-hand menu, then expand the Tables list underneath your **sales_YOUR_UNIQUE_IDENTIFIER** database and find the **dbo.Reviews (External)** table. If you do not see it, right-click on the Tables folder, then select Refresh. The "(External)" portion of the table name denotes that it is a virtual data object that was added as an external table.

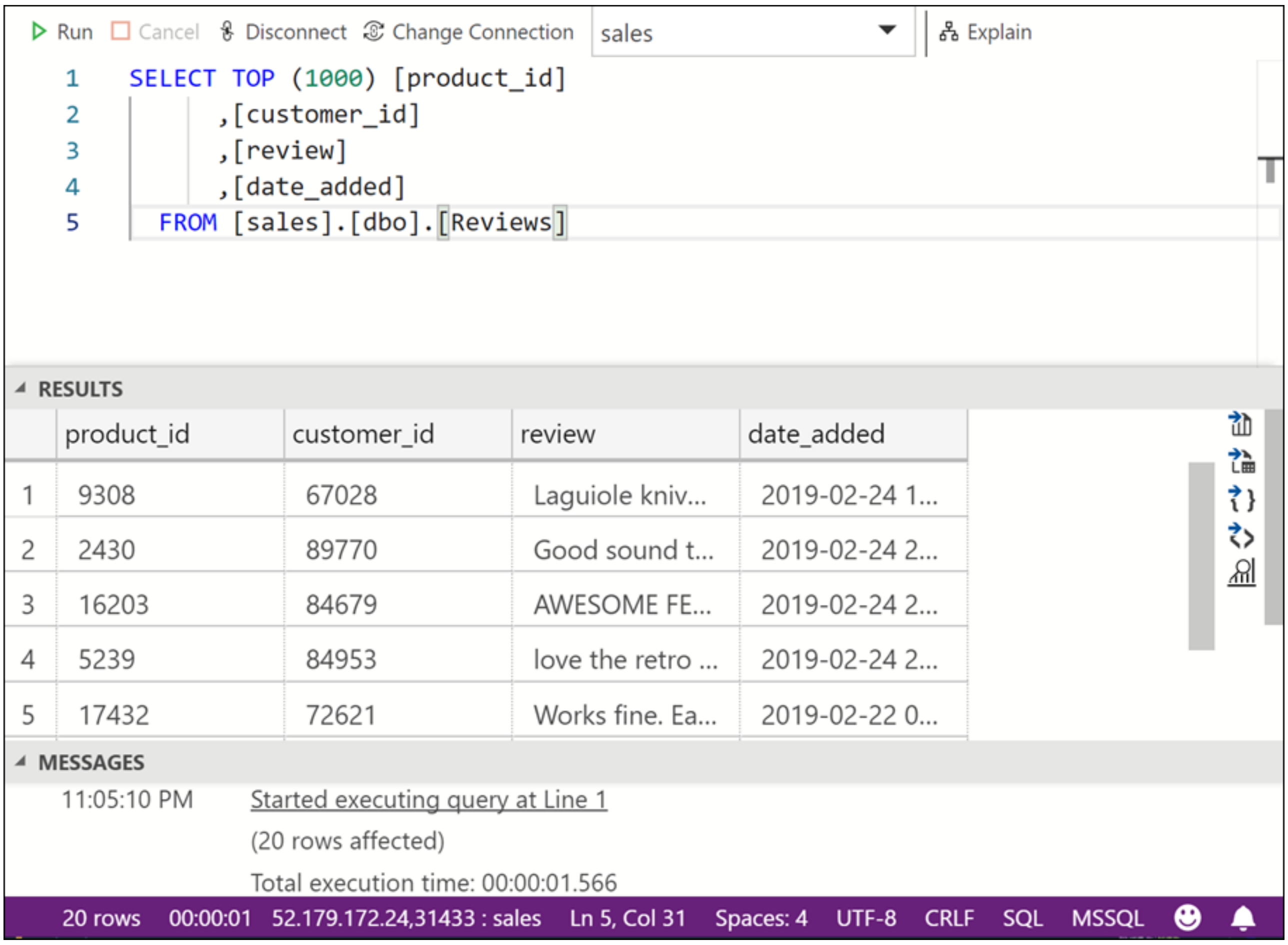


7. Right-click the **dbo.Reviews (External)** table, then select the **Select Top 1000** menu option to display the table contents.



8. You should see a SQL query selecting the top 1000 records from the Reviews table and its results. The interesting thing to note is that the query selects the table and fields using the same syntax you would use to select from any other table in the sales database. The fact that the Reviews table is external is

completely seamless and transparent to the user. This is the power of data virtualization in SQL Server 2019.



```
SELECT TOP (1000) [product_id]
        ,[customer_id]
        ,[review]
        ,[date_added]
  FROM [sales].[dbo].[Reviews]
```

RESULTS

	product_id	customer_id	review	date_added
1	9308	67028	Laguiole kniv...	2019-02-24 1...
2	2430	89770	Good sound t...	2019-02-24 2...
3	16203	84679	AWESOME FE...	2019-02-24 2...
4	5239	84953	love the retro ...	2019-02-24 2...
5	17432	72621	Works fine. Ea...	2019-02-22 0...

MESSAGES

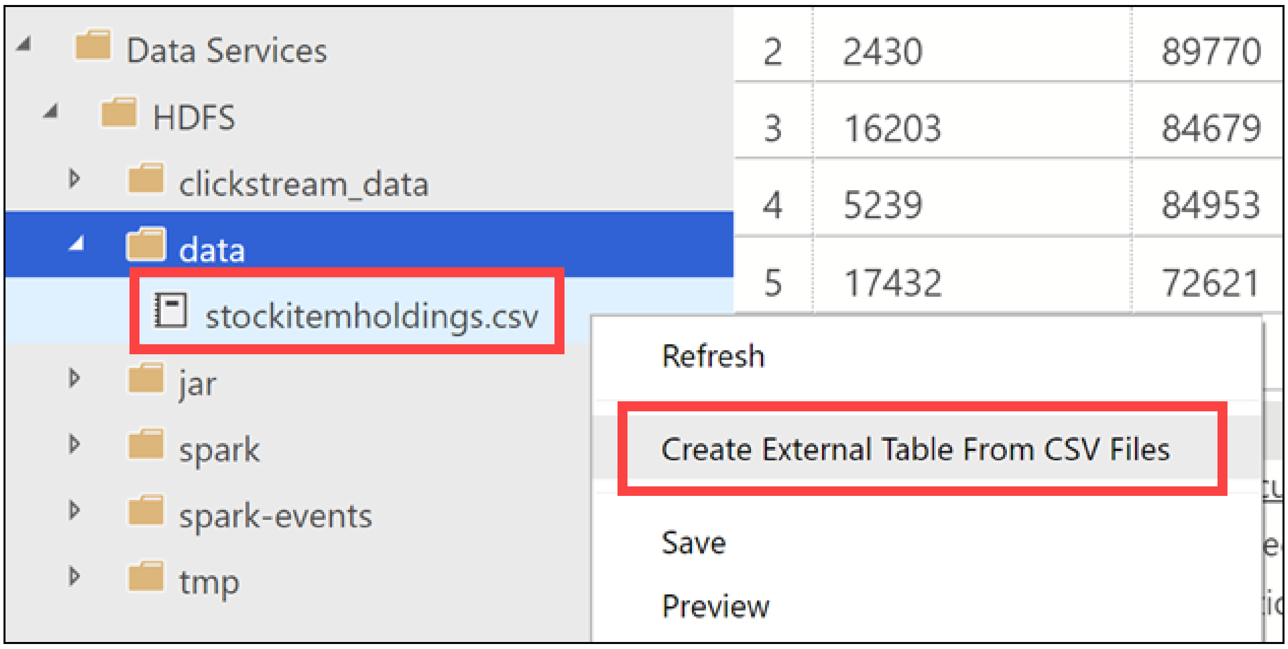
```
11:05:10 PM    Started executing query at Line 1
               (20 rows affected)
               Total execution time: 00:00:01.566
```

20 rows 00:00:01 52.179.172.24,31433 : sales Ln 5, Col 31 Spaces: 4 UTF-8 CRLF SQL MSSQL

```
SELECT TOP (1000) [product_id]
    ,[customer_id]
    ,[review]
    ,[date_added]
FROM [sales_YOUR_UNIQUE_IDENTIFIER].[dbo].[Reviews]
```

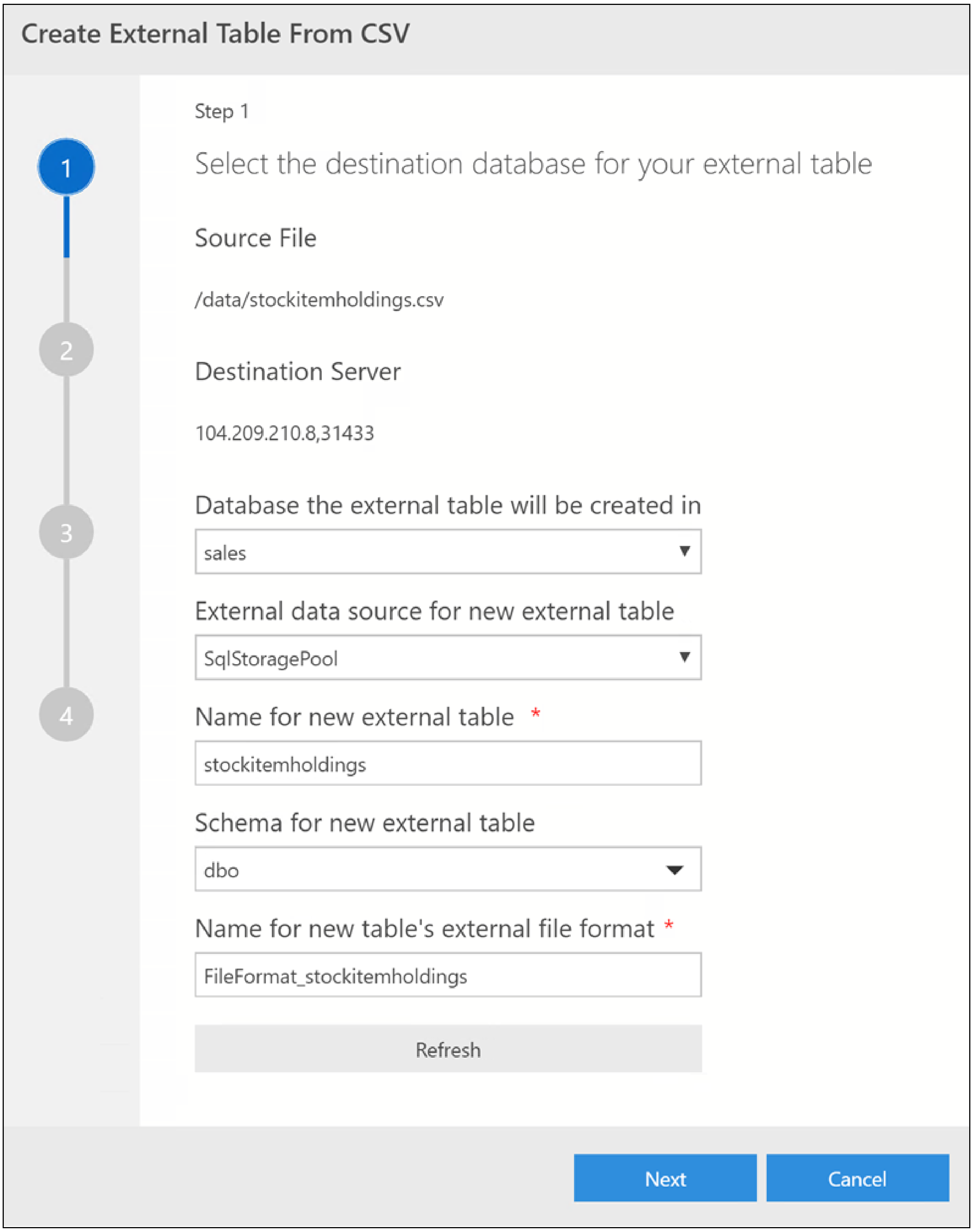
9. The next data source we will be virtualizing is a CSV file that was uploaded to HDFS.

10. Within Azure Data Studio, scroll down below the list of SQL Server 2019 databases to find the **Data Services** folder. Expand that folder, expand the **HDFS** folder, then expand the **data** subfolder. Right-click on the `stockitemholdings.csv` file, then select **Create External Table From CSV Files**.



2	2430	89770
3	16203	84679
4	5239	84953
5	17432	72621

11. In the Create External Table from CSV dialog, confirm that the **sales** database is selected and that the name of the external table is **stockitemholdings**.



Create External Table From CSV

Step 1

Select the destination database for your external table

Source File

/data/stockitemholdings.csv

Destination Server

104.209.210.8,31433

Database the external table will be created in: sales

External data source for new external table: SqlStoragePool

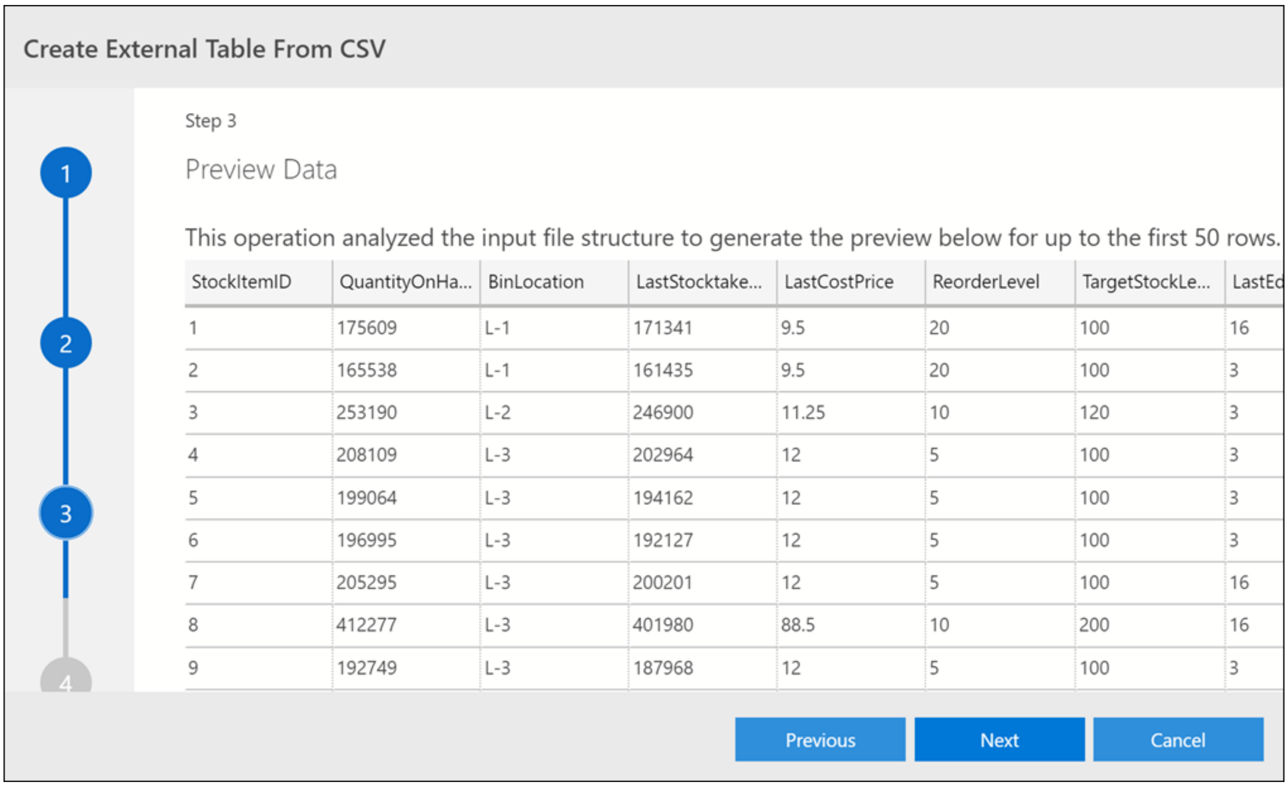
Name for new external table *: stockitemholdings

Schema for new external table: dbo

Name for new table's external file format *: FileFormat_stockitemholdings

12. Click **Next**.

13. The next step displays a preview of the first 50 rows CSV data for validation. Click **Next** to continue.



Create External Table From CSV

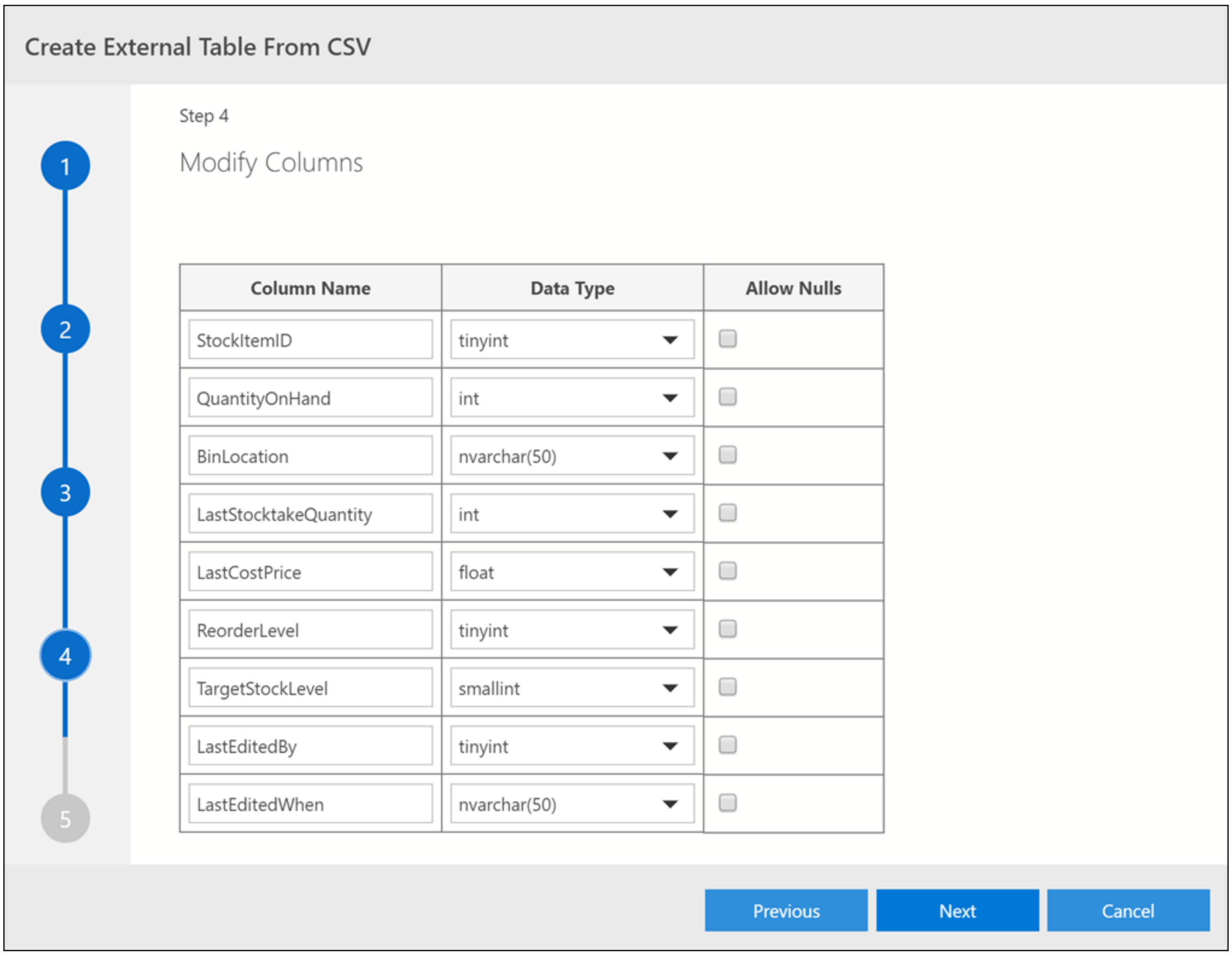
Step 3

Preview Data

This operation analyzed the input file structure to generate the preview below for up to the first 50 rows.

StockItemID	QuantityOnHa...	BinLocation	LastStocktake...	LastCostPrice	ReorderLevel	TargetStockLe...	LastEd
1	175609	L-1	171341	9.5	20	100	16
2	165538	L-1	161435	9.5	20	100	3
3	253190	L-2	246900	11.25	10	120	3
4	208109	L-3	202964	12	5	100	3
5	199064	L-3	194162	12	5	100	3
6	196995	L-3	192127	12	5	100	3
7	205295	L-3	200201	12	5	100	16
8	412277	L-3	401980	88.5	10	200	16
9	192749	L-3	187968	12	5	100	3

14. In the next step, you will be able to modify the columns of the external table you intend to create. You are able to alter the column name, change the data type, and allow for Nullable rows. For now, leave everything as-is and click **Next**.



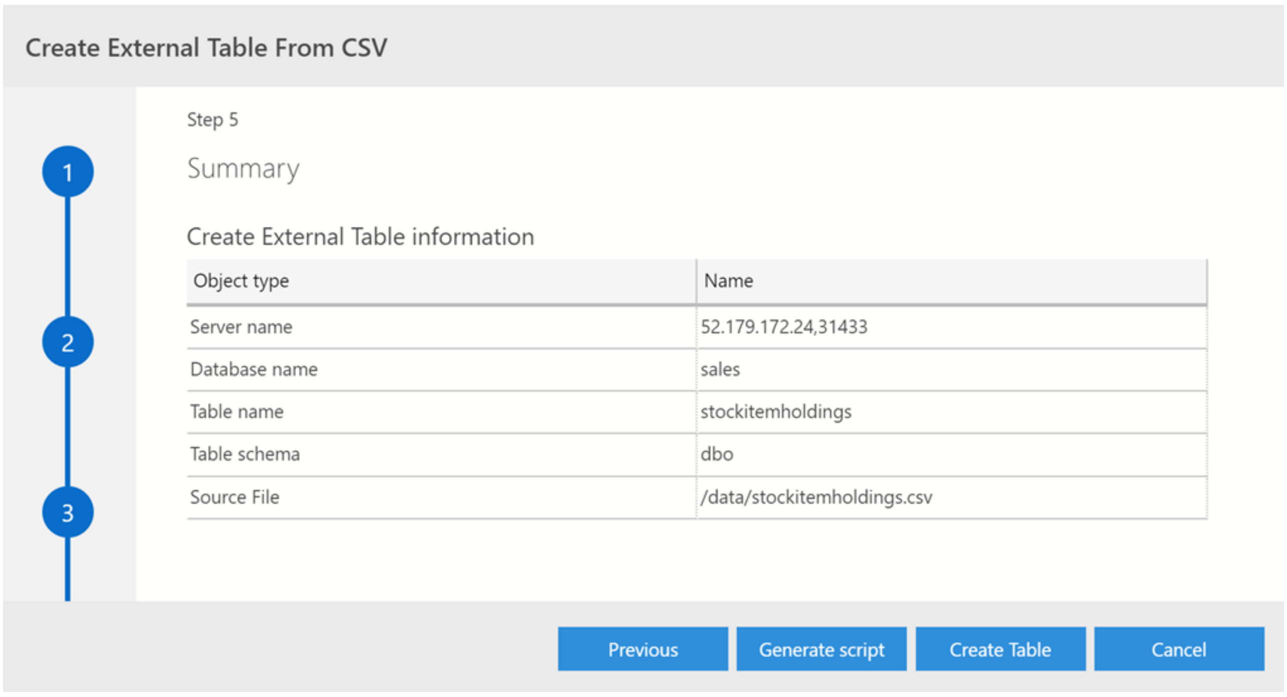
Create External Table From CSV

Step 4

Modify Columns

Column Name	Data Type	Allow Nulls
StockItemID	tinyint	☐
QuantityOnHand	int	☐
BinLocation	nvarchar(50)	☐
LastStocktakeQuantity	int	☐
LastCostPrice	float	☐
ReorderLevel	tinyint	☐
TargetStockLevel	smallint	☐
LastEditedBy	tinyint	☐
LastEditedWhen	nvarchar(50)	☐

15. Verify that everything looks correct in the Summary step, then click **Create Table**.


Create External Table From CSV

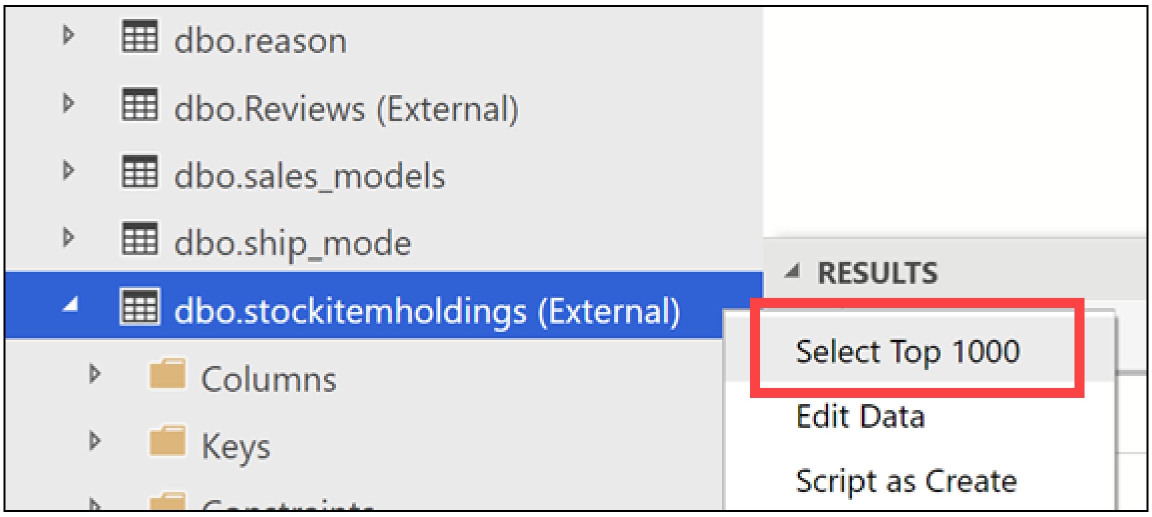
Step 5

Summary

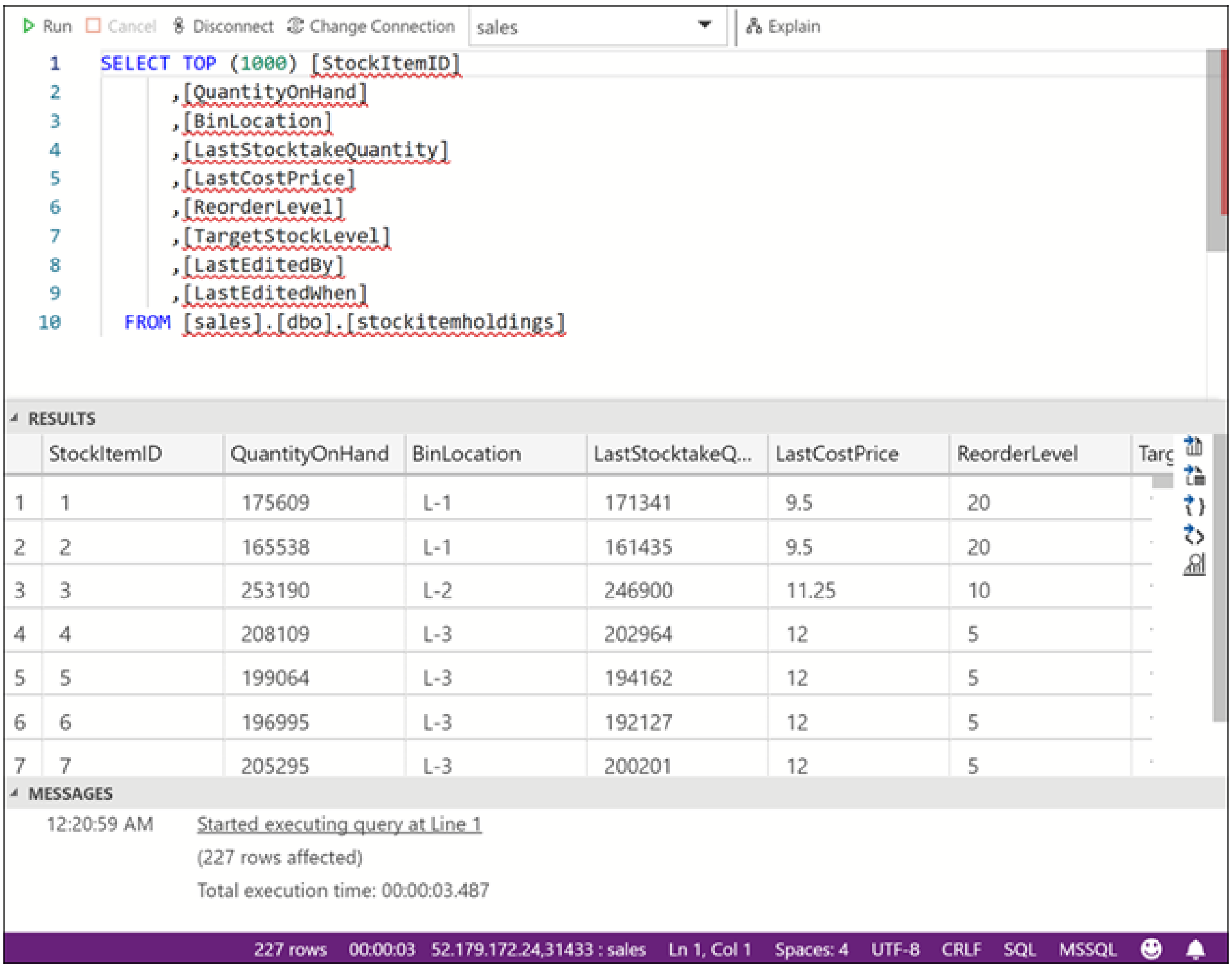
Create External Table information

Object type	Name
Server name	52.179.172.24,31433
Database name	sales
Table name	stockitemholdings
Table schema	dbo
Source File	/data/stockitemholdings.csv

16. As with the previous external table you created, a "Create External Table succeeded" dialog will appear under your task history in a few moments. Select the Servers link (Ctrl+G) on the left-hand menu, then expand the Tables list underneath your **sales** database and find the **dbo.stockitemholdings (External)** table. If you do not see it, right-click on the Tables folder, then select Refresh. **Right-click** the **dbo.stockitemholdings (External)** table, then select **Select Top 1000** from the context menu.



17. Just as before, you should see a SQL query selecting the top 1000 rows and its query results, this time from the `stockitemholdings` table. Again, the SQL query is the same type of query you would write to select from a table internal to the sales database.



```
SELECT TOP (1000) [StockItemID]
      ,[QuantityOnHand]
      ,[BinLocation]
      ,[LastStocktakeQuantity]
      ,[LastCostPrice]
      ,[ReorderLevel]
      ,[TargetStockLevel]
      ,[LastEditedBy]
      ,[LastEditedWhen]
  FROM [sales].[dbo].[stockitemholdings]
```

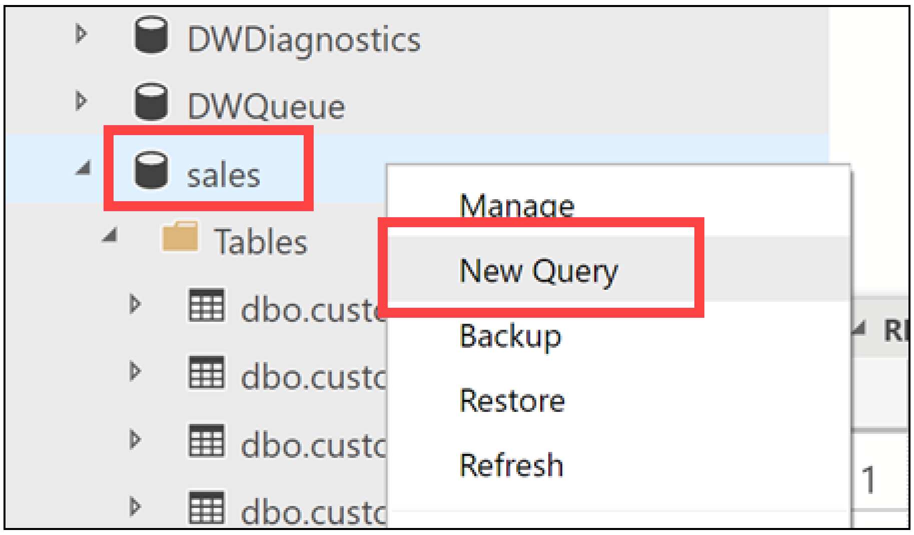
	StockItemID	QuantityOnHand	BinLocation	LastStocktakeQ...	LastCostPrice	ReorderLevel	Tarç
1	1	175609	L-1	171341	9.5	20	
2	2	165538	L-1	161435	9.5	20	
3	3	253190	L-2	246900	11.25	10	
4	4	208109	L-3	202964	12	5	
5	5	199064	L-3	194162	12	5	
6	6	196995	L-3	192127	12	5	
7	7	205295	L-3	200201	12	5	

```
12:20:59 AM    Started executing query at Line 1
               (227 rows affected)
               Total execution time: 00:00:03.487
```

227 rows 00:00:03 52.179.172.24,31433 : sales Ln 1, Col 1 Spaces: 4 UTF-8 CRLF SQL MSSQL

```
SELECT TOP (1000) [StockItemID]
    ,[QuantityOnHand]
    ,[BinLocation]
    ,[LastStocktakeQuantity]
    ,[LastCostPrice]
    ,[ReorderLevel]
    ,[TargetStockLevel]
    ,[LastEditedBy]
    ,[LastEditedWhen]
FROM [sales_YOUR_UNIQUE_IDENTIFIER].[dbo].[stockitemholdings]
```

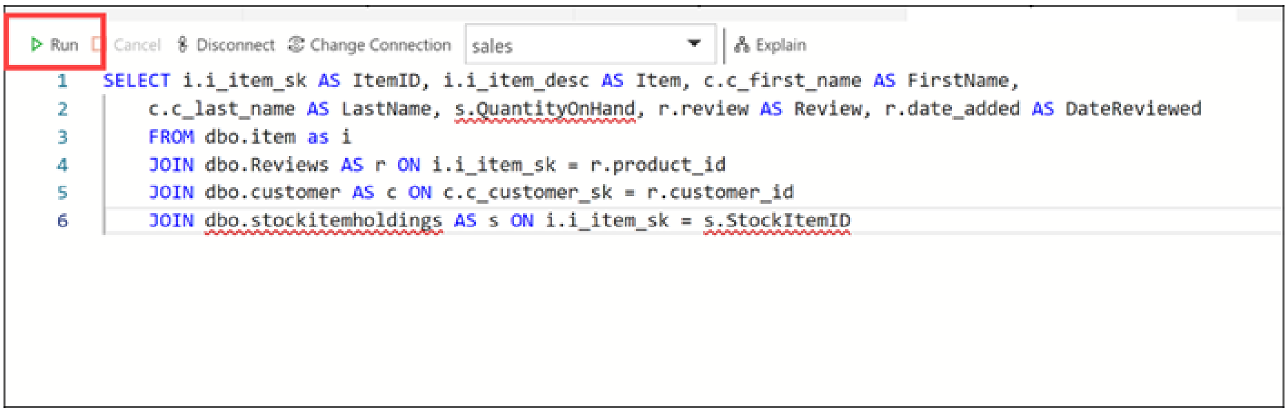
18. Now that we have our two external tables added, we will now join those two external tables and two internal tables with a new SQL query to demonstrate how you can seamlessly combine all these data sources without having to copy any files or with separate queries or additional processing of that data. **Right-click** the **sales_YOUR_UNIQUE_IDENTIFIER** database, then select **New Query**.



19. Paste the following into the new query window:

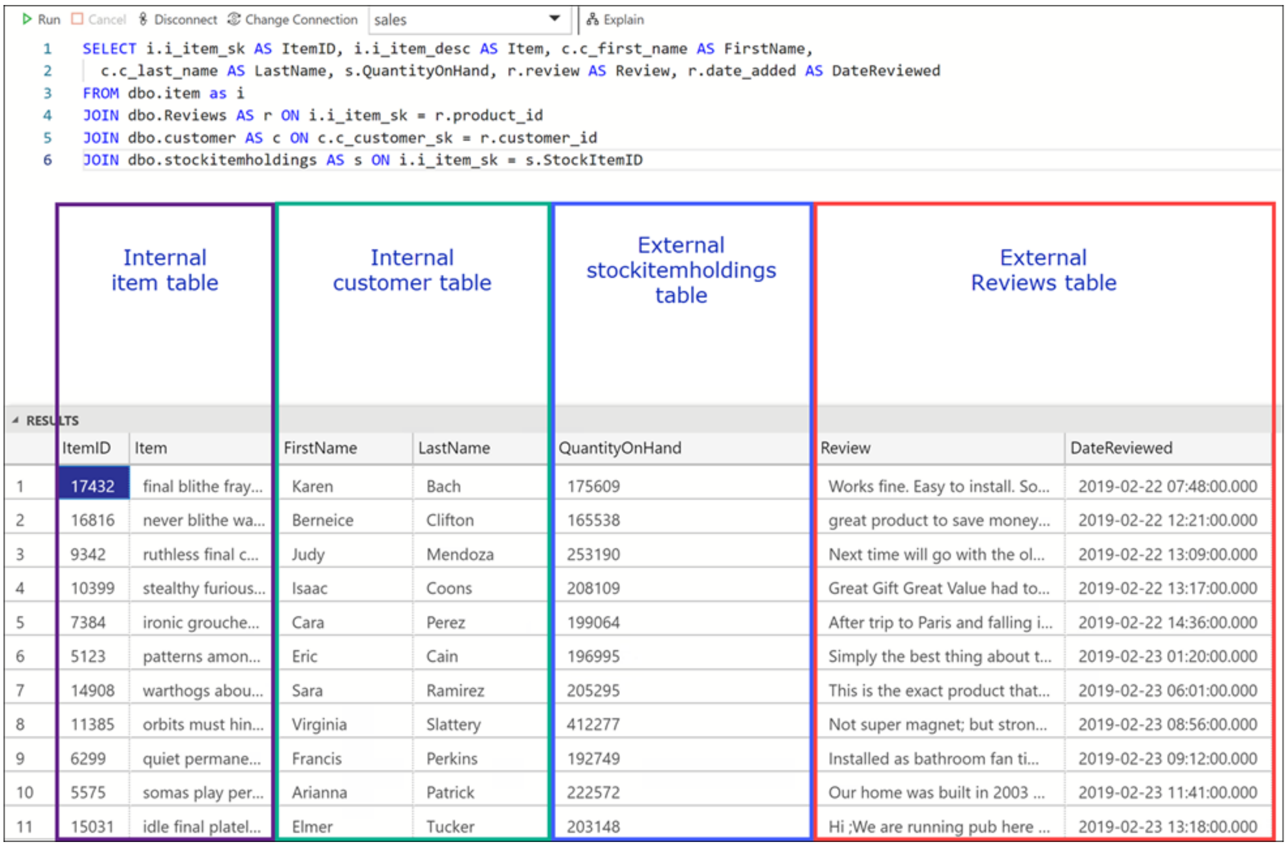
```
SELECT i.i_item_sk AS ItemID, i.i_item_desc AS Item, c.c_first_name AS
FirstName,
  c.c_last_name AS LastName, s.QuantityOnHand, r.review AS Review,
r.date_added AS DateReviewed
FROM dbo.item as i
JOIN dbo.Reviews AS r ON i.i_item_sk = r.product_id
JOIN dbo.customer AS c ON c.c_customer_sk = r.customer_id
JOIN dbo.stockitemholdings AS s ON i.i_item_sk = s.StockItemID
```

20. Click the **Run** button above the query window to execute.



```
SELECT i.i_item_sk AS ItemID, i.i_item_desc AS Item, c.c_first_name AS FirstName,
    c.c_last_name AS LastName, s.QuantityOnHand, r.review AS Review, r.date_added AS DateReviewed
    FROM dbo.item as i
    JOIN dbo.Reviews AS r ON i.i_item_sk = r.product_id
    JOIN dbo.customer AS c ON c.c_customer_sk = r.customer_id
    JOIN dbo.stockitemholdings AS s ON i.i_item_sk = s.StockItemID
```

21. At the bottom of the query window, you will see results that include columns from the four data sources.



```
SELECT i.i_item_sk AS ItemID, i.i_item_desc AS Item, c.c_first_name AS FirstName,
  c.c_last_name AS LastName, s.QuantityOnHand, r.review AS Review, r.date_added AS DateReviewed
FROM dbo.item as i
JOIN dbo.Reviews AS r ON i.i_item_sk = r.product_id
JOIN dbo.customer AS c ON c.c_customer_sk = r.customer_id
JOIN dbo.stockitemholdings AS s ON i.i_item_sk = s.StockItemID
```

	ItemID	Item	FirstName	LastName	QuantityOnHand	Review	DateReviewed
1	17432	final blithe fray...	Karen	Bach	175609	Works fine. Easy to install. So...	2019-02-22 07:48:00.000
2	16816	never blithe wa...	Berneice	Clifton	165538	great product to save money...	2019-02-22 12:21:00.000
3	9342	ruthless final c...	Judy	Mendoza	253190	Next time will go with the ol...	2019-02-22 13:09:00.000
4	10399	stealthy furious...	Isaac	Coons	208109	Great Gift Great Value had to...	2019-02-22 13:17:00.000
5	7384	ironic grouche...	Cara	Perez	199064	After trip to Paris and falling i...	2019-02-22 14:36:00.000
6	5123	patterns amon...	Eric	Cain	196995	Simply the best thing about t...	2019-02-23 01:20:00.000
7	14908	warthogs abou...	Sara	Ramirez	205295	This is the exact product that...	2019-02-23 06:01:00.000
8	11385	orbits must hin...	Virginia	Slattery	412277	Not super magnet; but stron...	2019-02-23 08:56:00.000
9	6299	quiet permane...	Francis	Perkins	192749	Installed as bathroom fan ti...	2019-02-23 09:12:00.000
10	5575	somas play per...	Arianna	Patrick	222572	Our home was built in 2003 ...	2019-02-23 11:41:00.000
11	15031	idle final platel...	Elmer	Tucker	203148	Hi ;We are running pub here ...	2019-02-23 13:18:00.000

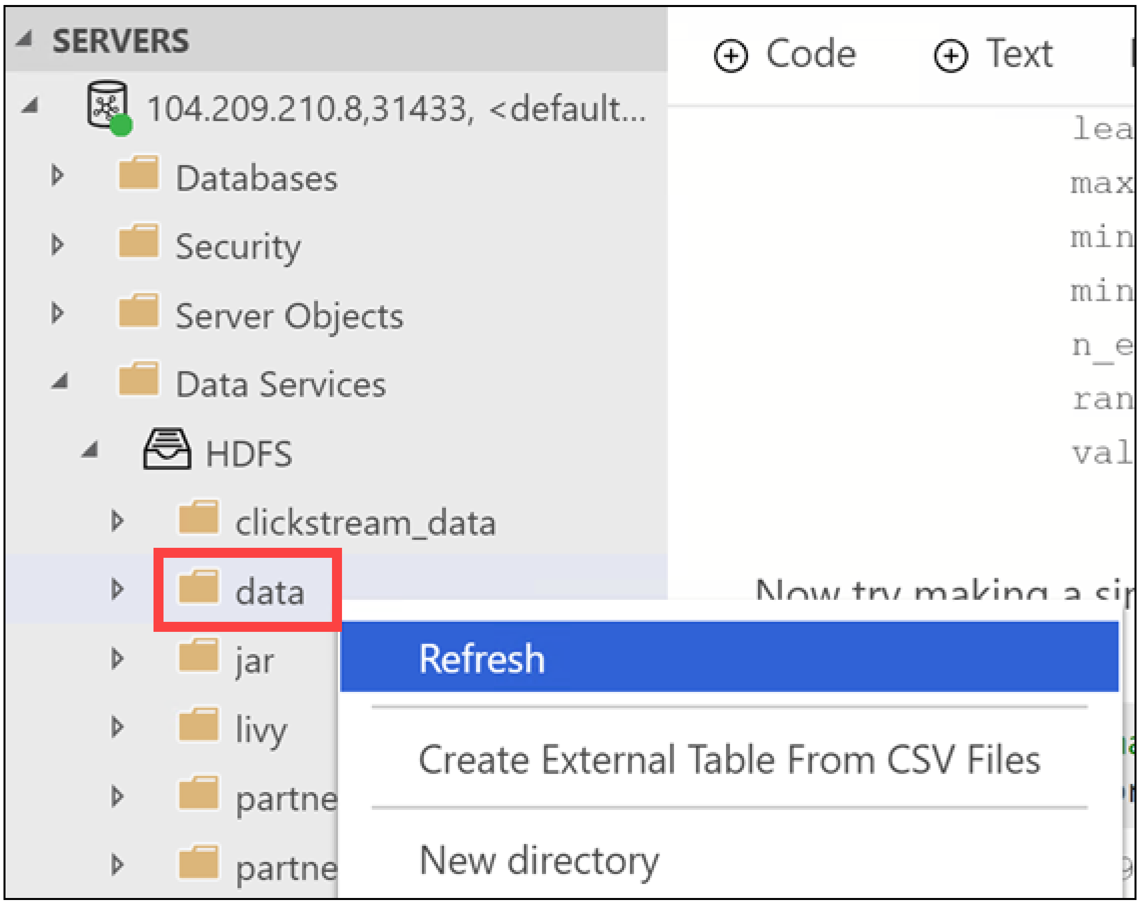
Task 2: Train a machine learning model, score and save data as external table

In this task, you will use Azure Data Studio to execute a notebook that will enable you to train a model to predict the battery lifetime, apply the model to make batch predictions against a set of vehicle telemetry and save the scored telemetry to an external table that you can query using SQL.

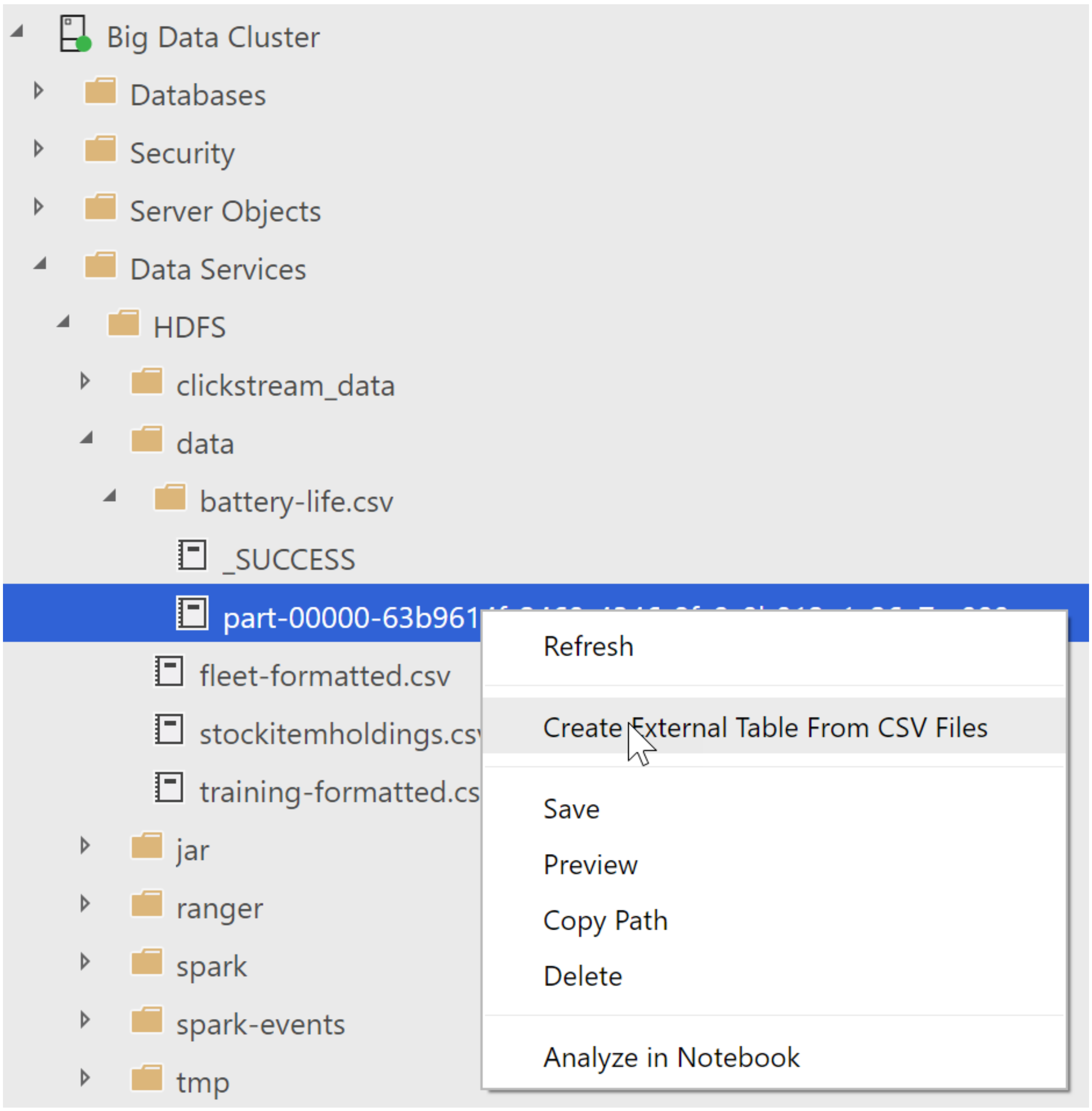
1. In Azure Data Studio, click **File**, then **Open File...**.
2. In the folder browser dialog, navigate to the **C:\lab-files\data\2** folder and select **predict-battery-life-with-sqlbdc.ipynb**.
3. When the notebook opens, you need to select the **Kernel** you would like to use to run the notebook. Locate the **Kernel** dropdown in the toolbar above the notebook, then select **PySpark3**.
4. Follow the instructions in the notebook and return to the next step after you have completed the notebook.

   > There may be a kernel error pertaining to there not being a valid SQL connection when you open the notebook. If this happens, close the notebook and Azure Data Studio, then re-launch, reconnect, then re-open the notebook.

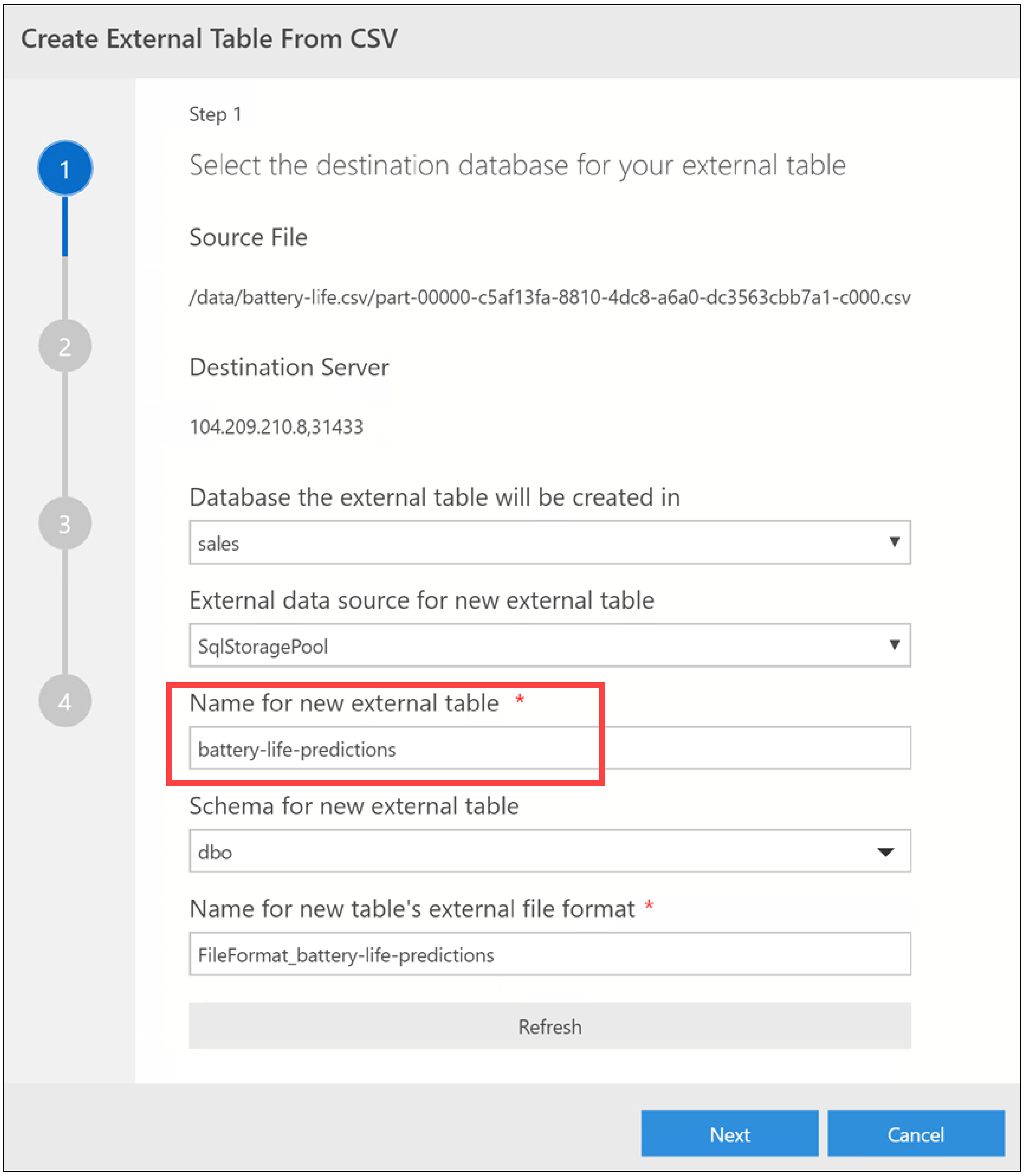
5. In Azure Data Studio, under Servers, expand your Big Data Cluster, `Data Services`, `HDFS`, `data`.
6. Right click the `data` folder and select `Refresh` to see the newly created folder.



7. You should see `battery-life-YOUR_UNIQUE_IDENTIFIER.csv` as a folder (where `YOUR_UNIQUE_IDENTIFIER` is your assigned identifier), expand it and then right click on the CSV file whose name starts with `part-00000-` and select `Create External Table From CSV Files`.



8. In Step 1 of the wizard, select the `sales` database and for the `Name for new external table` field provide **battery-life-predictions**. Click **Next**.



Create External Table From CSV

Step 1

Select the destination database for your external table

Source File

/data/battery-life.csv/part-00000-c5af13fa-8810-4dc8-a6a0-dc3563cbb7a1-c000.csv

Destination Server

104.209.210.8,31433

Database the external table will be created in: sales

External data source for new external table: SqlStoragePool

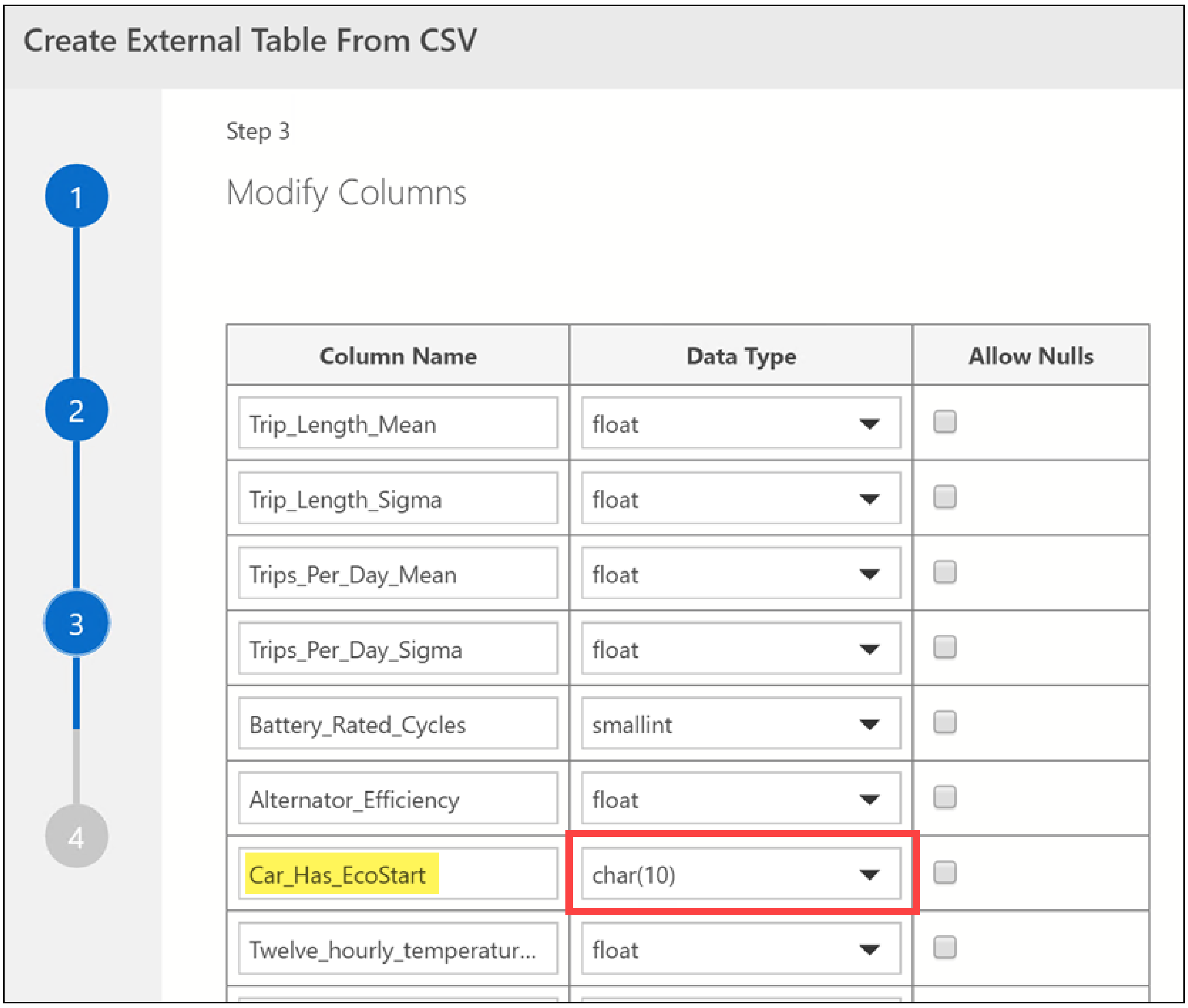
Name for new external table *: battery-life-predictions

Schema for new external table: dbo

Name for new table's external file format *: FileFormat_battery-life-predictions

9. On Step 2, Preview Data, click **Next**.

10. On Step 3, for the column `Car_Has_EcoStart` set the Data Type to **char(10)**. Click **Next**.



Create External Table From CSV

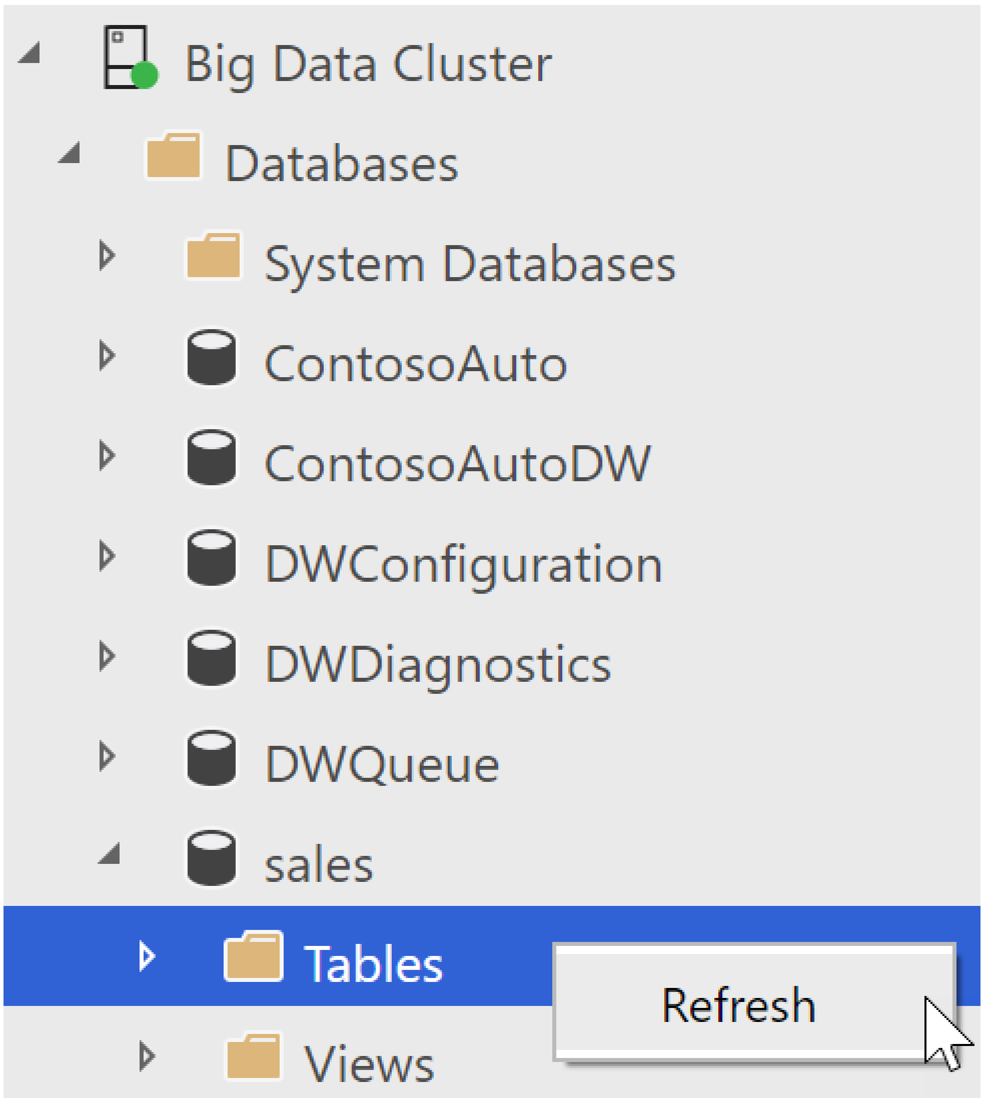
Step 3

Modify Columns

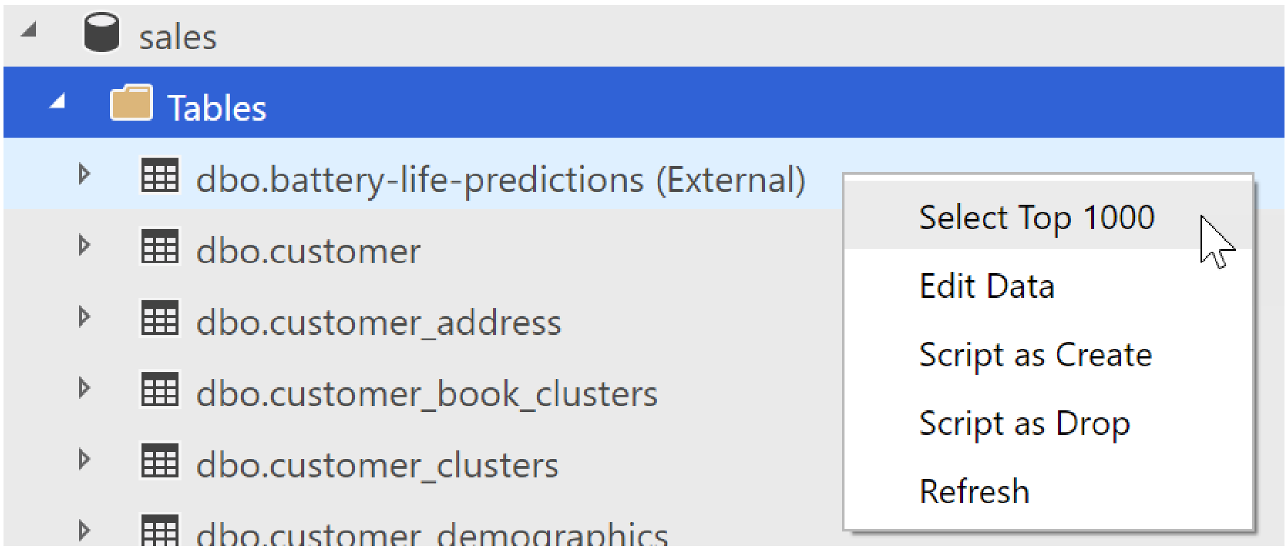
Column Name	Data Type	Allow Nulls
Trip_Length_Mean	float	☐
Trip_Length_Sigma	float	☐
Trips_Per_Day_Mean	float	☐
Trips_Per_Day_Sigma	float	☐
Battery_Rated_Cycles	smallint	☐
Alternator_Efficiency	float	☐
Car_Has_EcoStart	char(10)	☐
Twelve_hourly_temperatur...	float	☐

11. On Step, click **Create Table**. Your predictions are now available for SQL querying in the battery-life-predictions table in the sales database.

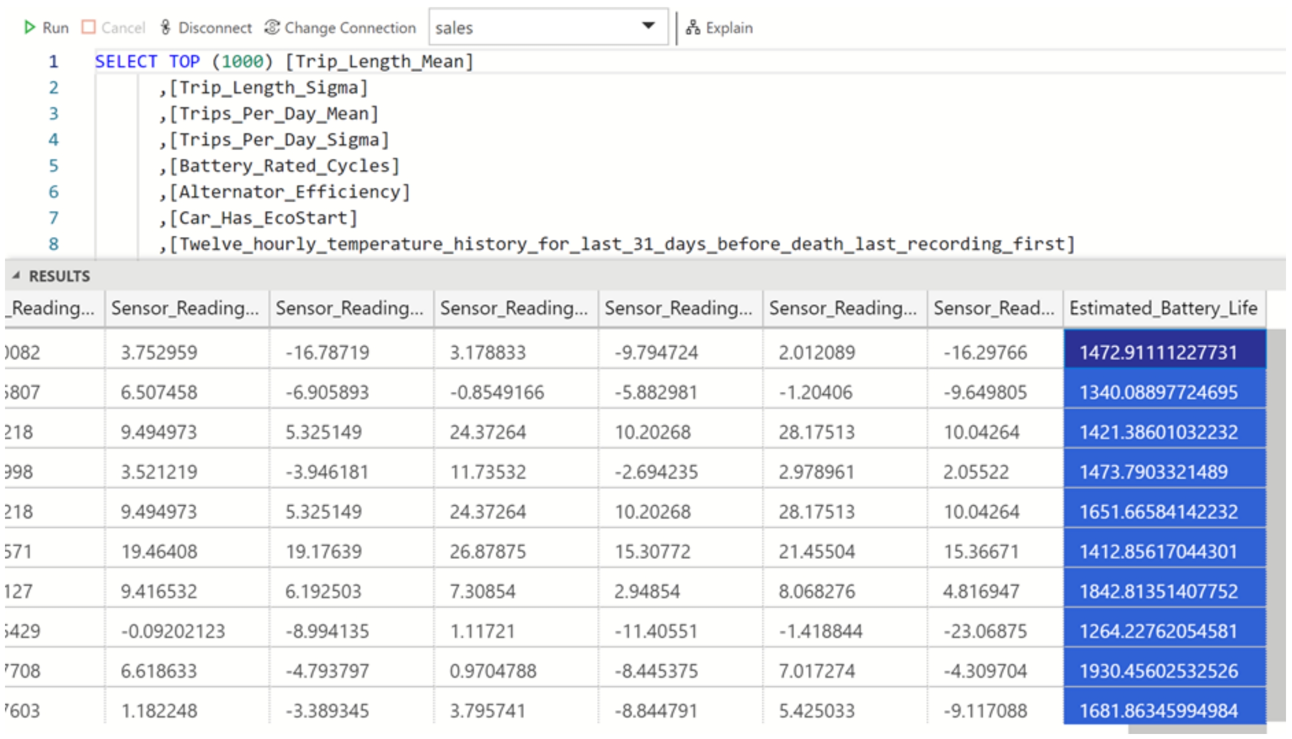
12. In Azure Data Studio, Servers, expand your Big Data Cluster, `Databases`, `sales_YOUR-UNIQUE-IDENTIFIER`, right click `Tables` and then select `Refresh`.



13. Expand `tables`, right click `battery-life-prediction` and select `query` to view the data contained by the external table.



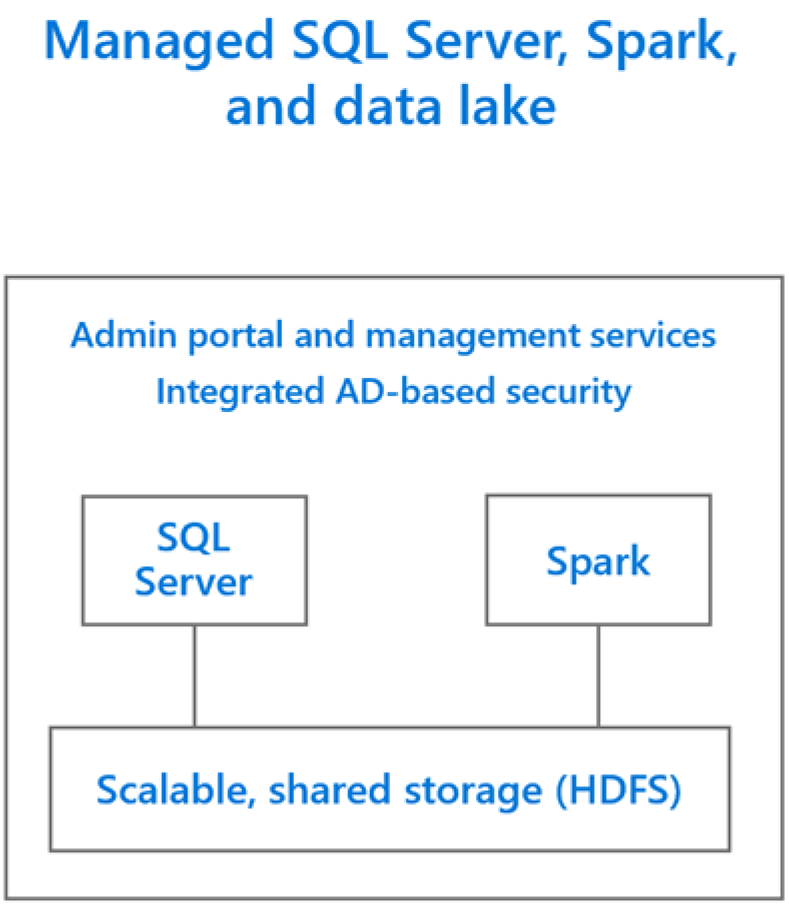
14. The vehicle telemetry along with predictions will appear. These are queried from the external table which is sourced from the CSV you created using the notebook.



```
SELECT TOP (1000) [Trip_Length_Mean]
      ,[Trip_Length_Sigma]
      ,[Trips_Per_Day_Mean]
      ,[Trips_Per_Day_Sigma]
      ,[Battery_Rated_Cycles]
      ,[Alternator_Efficiency]
      ,[Car_Has_EcoStart]
      ,[Twelve_hourly_temperature_history_for_last_31_days_before_death_last_recording_first]
```

Reading...	Sensor_Reading...	Sensor_Reading...	Sensor_Reading...	Sensor_Reading...	Sensor_Reading...	Sensor_Read...	Estimated_Battery_Life
082	3.752959	-16.78719	3.178833	-9.794724	2.012089	-16.29766	1472.91111227731
807	6.507458	-6.905893	-0.8549166	-5.882981	-1.20406	-9.649805	1340.08897724695
218	9.494973	5.325149	24.37264	10.20268	28.17513	10.04264	1421.38601032232
998	3.521219	-3.946181	11.73532	-2.694235	2.978961	2.05522	1473.7903321489
218	9.494973	5.325149	24.37264	10.20268	28.17513	10.04264	1651.66584142232
571	19.46408	19.17639	26.87875	15.30772	21.45504	15.36671	1412.85617044301
127	9.416532	6.192503	7.30854	2.94854	8.068276	4.816947	1842.81351407752
429	-0.09202123	-8.994135	1.11721	-11.40551	-1.418844	-23.06875	1264.22762054581
708	6.618633	-4.793797	0.9704788	-8.445375	7.017274	-4.309704	1930.45602532526
603	1.182248	-3.389345	3.795741	-8.844791	5.425033	-9.117088	1681.86345994984

Task 3: Mounting an Azure Data Lake Gen2 Storage Account to SQL Server 2019 Big Data Cluster using HDFS Tiering


Managed SQL Server, Spark, and data lake
Admin portal and management services
Integrated AD-based security
SQL Server
Spark
Scalable, shared storage (HDFS)

SQL Server 2019 storage pools help you create a data lake by storing data in scalable, shared storage provided by Hadoop File System (HDFS). Store high volume data in this layer and access it easily using either SQL or Apache Spark. You can expand your HDFS storage by using tiering.

HDFS also provides data persistency, as HDFS data is spread across all the storage nodes in the SQL big data cluster. However, you can add external HDFS data sources to the HDFS cluster through tiering. With tiering, applications can seamlessly access data in a variety of external stores as though the data resides in the local HDFS. This allows you to interact with the files in Azure Data Lake Storage Gen2 or Amazon S3 as if they were local files. Both options allow you to mount the data store using storage keys. However, with Azure Data Lake Storage Gen 2, you can either use an Azure Storage access key or an Azure Active Directory User Account to gain permission to the files. For this lab, we will use the access key.

1. In Windows, open the Windows Command Prompt.
2. Connect to your Microsoft SQL Server 2019 Big Data Cluster:

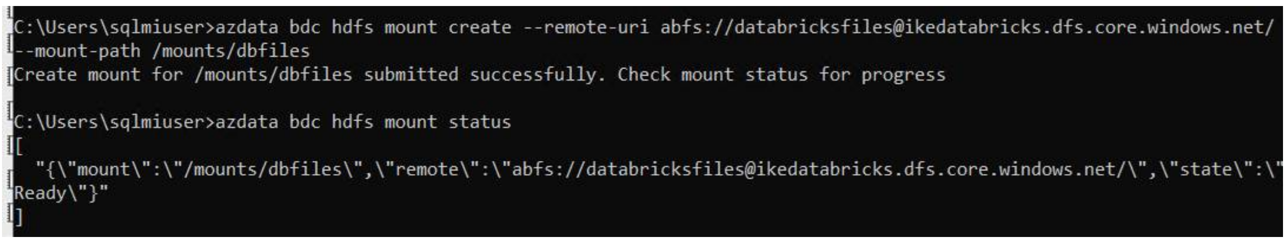
   ```
   mssqlctl login -e https://<SQL SERVER Controller IP ADDRESS>:30080
   ```

   - You will be prompted for your big data cluster name.
   - The user name is admin
   - The password is MySQLBigData2019
3. Execute the following in the command prompt to set a new environment variable with the ADLS Gen2 account name and access key credentials:

   ```
   set
   MOUNT_CREDENTIALS=fs.azure.abfs.account.name=ikedatabricks.dfs.core.windows.
   net,fs.azure.account.key.ikedatabricks.dfs.core.windows.net=HUYPk/VUjdYzkCvr
   KXTgFBObt5VQcp5DCY7C9KiSHX42lv65mjmBFmKFVTLy7Z7suQ0WV44mncuUOvnE8NkxGg==
   ```
4. Execute the following in the command prompt to mount the drive:

   ```
   azdata bdc hdfs mount create --remote-uri
   abfs://databricksfiles@ikedatabricks.dfs.core.windows.net/ --mount-path
   /mounts/dbfiles
   ```
5. Once the storage account has been mounted, you can check the status:

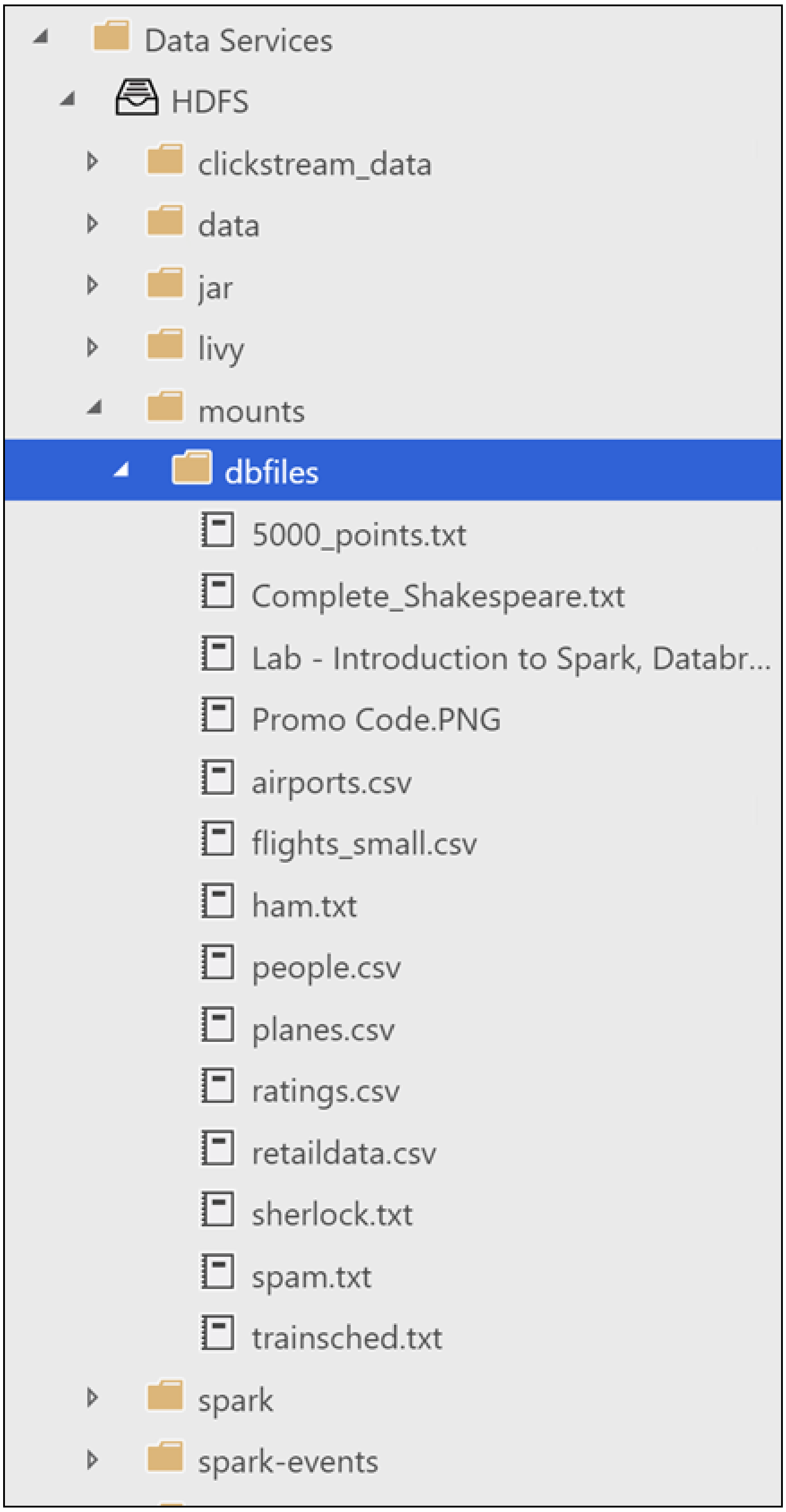
   ```
   azdata bdc hdfs mount status
   ```



```
C:\Users\sqlmiuser>azdata bdc hdfs mount create --remote-uri abfs://databricksfiles@ikedatabricks.dfs.core.windows.net/
--mount-path /mounts/dbfiles
Create mount for /mounts/dbfiles submitted successfully. Check mount status for progress

C:\Users\sqlmiuser>azdata bdc hdfs mount status
[
  "{\"mount\":\"/mounts/dbfiles\",\"remote\":\"abfs://databricksfiles@ikedatabricks.dfs.core.windows.net/\",\"state\":\"
Ready\"}"
]
```

6. Now you can use Azure Data Studio and view the files from ADLS Gen2 under your HDFS folder in your Servers pane. Look under /HDFS/mounts/dbfiles/



- Data Services
  - HDFS
    - clickstream_data
    - data
    - jar
    - livy
    - mounts
      - dbfiles
        - 5000_points.txt
        - Complete_Shakespeare.txt
        - Lab - Introduction to Spark, Databr...
        - Promo Code.PNG
        - airports.csv
        - flights_small.csv
        - ham.txt
        - people.csv
        - planes.csv
        - ratings.csv
        - retaildata.csv
        - sherlock.txt
        - spam.txt
        - trainsched.txt
    - spark
    - spark-events

7. Now that the drive is mounted, create an external file format for CSV. Open a new query window and paste the following command. Replace **sales_YOUR_UNIQUE_IDENTIFIER** with the name of your unique sales database. The `YOUR_UNIQUE_IDENTIFIER` portion of the name is the unique identifier assigned to you for this lab.

```
USE sales_YOUR_UNIQUE_IDENTIFIER;
GO
CREATE EXTERNAL FILE FORMAT csv_file
WITH (
    FORMAT_TYPE = DELIMITEDTEXT,
    FORMAT_OPTIONS(
        FIELD_TERMINATOR = ',',
        STRING_DELIMITER = '"',
        FIRST_ROW = 2,
        USE_TYPE_DEFAULT = TRUE)
);
```

8. Now create an external connection to your HDFS cluster:

```
IF NOT EXISTS(SELECT * FROM sys.external_data_sources WHERE name =
'SqlStoragePool')
BEGIN
CREATE EXTERNAL DATA SOURCE SqlStoragePool
WITH (LOCATION = 'sqlhdfs://controller-svc:8080/default');
END
```

9. Now let's create two tables to two different files that exist in the storage account:

```
CREATE EXTERNAL TABLE planes
("tailnum" VARCHAR(100),        "year" VARCHAR(4),      "type" VARCHAR(100)
,       "manufacturer" VARCHAR(100),    "model" VARCHAR(20),    "engines"
BIGINT
,       "seats" BIGINT, "speed" VARCHAR(20)
,       "engine" VARCHAR(20))
WITH
(
    DATA_SOURCE = SqlStoragePool,
    LOCATION = '/mounts/dbfiles/planes.csv',
    FILE_FORMAT = csv_file
);
GO

CREATE EXTERNAL TABLE flights
("year" BIGINT,         "month" BIGINT,         "day" BIGINT,   "dep_time"
BIGINT
    ,   "dep_delay" BIGINT,     "arr_time" BIGINT,      "arr_delay" BIGINT,
"carrier" VARCHAR(100)
    ,   "tailnum" VARCHAR(20),  "flight" VARCHAR(20),   "origin" VARCHAR(50)
    ,   "dest" VARCHAR(50),     "air_time" BIGINT,      "distance" BIGINT,
      "hour" BIGINT,    "minute" BIGINT)
WITH
(
    DATA_SOURCE = SqlStoragePool,
    LOCATION = '/mounts/dbfiles/flights_small.csv',
    FILE_FORMAT = csv_file
);
```

10. Once the tables are created, you can interact with them like normal tables. For instance, you can run a query that joins the two tables like this:

```
SELECT *
  FROM planes p
  JOIN flights f
    on p.tailnum = f.tailnum
```

Wrap-up

Thank you for participating in the SQL Server 2019 Big Data Clusters experience! We hope you are excited about the new capabilities, and will refer back to this experience to learn more about these features.

To recap, you experienced:

1. How to minimize or remove the need for ETL through **data virtualization** with relational data sources and CSV files, by being able to query against these alongside internal SQL 2019 tables with no data movement required.
2. Training a machine learning model by running a Jupyter notebook on the Big Data cluster, then scoring data with the trained model and saving it as an external table for easy access.
3. You learned to use HDFS tiering to mount files from an Azure Data Lake Store Gen2 account, which allowed you to create tables from files as if they were local to the cluster.

Additional resources and more information

- What's new in SQL Server 2019 preview
- SQL Server 2019 big data clusters overview and architecture
- How to run a sample notebook in Azure Data Studio on a SQL Server 2019 big data cluster, and leverage Spark
- What is Azure Data Studio?
- Security Center for SQL Server Database Engine and Azure SQL Database
- SQL Data Discovery and Classification tool documentation
- Intelligent query processing in SQL databases
- What's new in SQL Server Machine Learning Services
- How to run Java code in SQL Server 2019
- Learning content in GitHub: SQL Server Workshops
- SQL Server Samples Repository in GitHub. Feature demos, code samples etc.