Designing and Implementing Enterprise-Scale Analytics Solutions Using Microsoft Azure and Microsoft Power BI

Exam Ref DP-500

Daniil Maslyuk
Justin Frébault

FREE SAMPLE CHAPTER |

Microsoft

Exam Ref DP-500 Designing and Implementing Enterprise-Scale Analytics Solutions Using Microsoft Azure and Microsoft Power BI

Daniil Maslyuk
Justin Frébault

Exam Ref DP-500 Designing and Implementing Enterprise-Scale Analytics Solutions Using Microsoft Azure and Microsoft Power BI

Published with the authorization of Microsoft Corporation by:
Pearson Education, Inc.

ISBN-13: 978-0-13-809737-0
ISBN-10: 0-13-809737-2

Library of Congress Control Number: 2023938072

ScoutAutomatedPrintCode

EDITOR-IN-CHIEF
Brett Bartow

EXECUTIVE EDITOR
Loretta Yates

DEVELOPMENT EDITOR
Songlin Qiu

MANAGING EDITOR
Sandra Schroeder

SENIOR PROJECT EDITOR
Tracey Croom

COPY EDITOR
Liz Welch

INDEXER
Timothy Wright

PROOFREADER
Donna E. Mulder

TECHNICAL EDITOR
Owen Auger

EDITORIAL ASSISTANT
Cindy Teeters

COVER DESIGNER
Twist Creative, Seattle

COMPOSITOR
codeMantra

To Dasha, Leonard, and William, who served as a great source of motivation and support.

—Daniil Maslyuk

To my wife Phaedra, who doesn't have much to do with data analytics but who has everything to do with enriching the rest of my life.

—Justin Frébault

Contents at a glance

Contents

Introduction

Exam DP-500 focuses on designing and implementing enterprise-scale analytics solutions using Microsoft Azure and Microsoft Power BI. About a quarter of the book examines implementing and managing a data analytics environment, which includes both Azure and Power BI. Another quarter of the book reviews the query and data transformation by using Azure and Power BI. One more quarter of the book is dedicated to tabular data modeling in Power BI. The remainder of the book reviews the data visualization skills in Azure and Power BI.

This book was written for business intelligence developers, business intelligence architects, data engineers, and data architects. Before reading this book, you should be familiar with Azure, Power BI, the basics of Power Query, and DAX. It helps if you've passed the PL-300 exam already.

This book covers every major topic area found on the exam, but it does not cover every exam question. Only the Microsoft exam team has access to the exam questions, and Microsoft regularly adds new questions to the exam, making it impossible to cover specific questions. You should consider this book a supplement to your relevant real-world experience and other study materials. If you encounter a topic in this book that you do not feel completely comfortable with, use the "Need more review?" links you'll find in the text to find more information and take the time to research and study the topic. Great information is available on MSDN, TechNet, and in blogs and forums.

Organization of this book

This book is organized by the "Skills measured" list published for the exam. The "Skills measured" list is available for each exam on the Microsoft Learn website: *microsoft.com/learn*. Each chapter in this book corresponds to a major topic area in the list, and the technical tasks in each topic area determine a chapter's organization. If an exam covers six major topic areas, for example, the book will contain six chapters.

Preparing for the exam

Microsoft certification exams are a great way to build your résumé and let the world know about your level of expertise. Certification exams validate your on-the-job experience and product knowledge. Although there is no substitute for on-the-job experience, preparation through study and hands-on practice can help you prepare for the exam. This book is *not* designed to teach you new skills.

We recommend that you augment your exam preparation plan by using a combination of available study materials and courses. For example, you might use the *Exam Ref* and another study guide for your at-home preparation and take a Microsoft Official Curriculum course for the classroom experience. Choose the combination that you think works best for you. Learn more about available classroom training, online courses, and live events at *microsoft.com/learn*.

Note that this *Exam Ref* is based on publicly available information about the exam and the authors' experience. To safeguard the integrity of the exam, authors do not have access to the live exam.

Microsoft certifications

Microsoft certifications distinguish you by proving your command of a broad set of skills and experience with current Microsoft products and technologies. The exams and corresponding certifications are developed to validate your mastery of critical competencies as you design and develop, or implement and support, solutions with Microsoft products and technologies both on-premises and in the cloud. Certification brings a variety of benefits to the individual and to employers and organizations.

NEED MORE REVIEW? **ALL MICROSOFT CERTIFICATIONS**

For information about Microsoft certifications, including a full list of available certifications, go to *microsoft.com/learn*.

Quick access to online references

Throughout this book are addresses to webpages that the authors have recommended you visit for more information. Some of these links can be very long and painstaking to type, so we've shortened them for you to make them easier to visit. We've also compiled them into a single list that readers of the print edition can refer to while they read.

Download the list at *MicrosoftPressStore.com/ERDP500/downloads*.

The URLs are organized by chapter and heading. Every time you come across a URL in the book, find the hyperlink in the list to go directly to the webpage.

Errata, updates, & book support

We've made every effort to ensure the accuracy of this book and its companion content. You can access updates to this book—in the form of a list of submitted errata and their related corrections—at:

MicrosoftPressStore.com/ERDP500/errata

If you discover an error that is not already listed, please submit it to us at the same page.

For additional book support and information, please visit *MicrosoftPressStore.com/Support*.

Please note that product support for Microsoft software and hardware is not offered through the previous addresses. For help with Microsoft software or hardware, go to *support.microsoft.com*.

Stay in touch

Let's keep the conversation going! We're on Twitter: *twitter.com/MicrosoftPress*.

Acknowledgments

Daniil Maslyuk: I would like to thank Loretta Yates for continuing to trust me to write the official Power BI exam reference books, Malobika Chakraborty for handling the project, the editing team for making this book a better read, and everyone else at Pearson who worked on this book to make it happen. This book wouldn't be possible without Justin, who made me realize that co-authoring doesn't have to be difficult.

Justin Frébault: I would like to thank Harry Misthos, my Pearson editor, for his continuous support during the writing process. I would also like to thank Loretta Yates, and the whole Microsoft Press team at Pearson, for all their hard work. Without you, the book wouldn't be a reality. Finally, Daniil Maslyuk, my co-author. It really was a delight collaborating with him on this book. Thank you!

About the authors

DANIIL MASLYUK is an independent business intelligence consultant, trainer, and speaker who specializes in Microsoft Power BI. Daniil blogs at xxlbi.com and tweets as @DMaslyuk.

JUSTIN FRÉBAULT MCT, is an independent data solutions architect and trainer who specializes in Microsoft data products. He blogs at lafreb.com and is active on LinkedIn.

CHAPTER 2

Query and transform data

Data duplication is natural in an organization. Data needs to be queried and transformed in a format that is most suitable for the use case. This is true even at the lowest level, in a database, where data is duplicated in the form of indexes to optimize reads.

Similarly, in an organization data will be queried and transformed from the operational to the analytical system(s), and queried and transformed again through the various layers of the analytical stack. In this chapter we will review how this is done with Azure Synapse and Power BI.

Skills covered in this chapter:

- 2.1: Query data by using Azure Synapse Analytics
- 2.2: Ingest and transform data by using Power BI

Skill 2.1: Query data by using Azure Synapse Analytics

There is no doubt that the amount of data generated and used by businesses is growing. And so are the requirements for near-real-time analytics; we want more insight and sooner. On top of that, the complexity of the analysis is increasing. With predictive and prescriptive solutions, we go beyond descriptive and diagnostic analytics.

To meet all these increasingly demanding requirements, data tools are multiplying. But there is value in a single platform: ease of training, reduced cognitive load, easier cross-team collaboration, fewer silos in the organization. . . . Azure Synapse Analytics is all about bringing together data engineers, data scientists, and data analysts in order to achieve more with your data.

This skill covers how to:

- Identify an appropriate Azure Synapse pool when analyzing data
- Recommend appropriate file types for querying serverless SQL pools
- Query relational data sources in dedicated or serverless SQL pools, including querying partitioned data sources
- Use a machine learning PREDICT function in a query

Identify an appropriate Azure Synapse pool when analyzing data

Azure Synapse Analytics is an extremely rich platform (see Figure 2-1). As the requirements around data become more and more complex, so does the need for features. Azure Synapse Analytics can be seen as a one-stop shop for data in your organization, bringing together data engineers, data scientists, and data analysts.

Overall architecture of Azure Synapse Analytics

In Chapter 1, we reviewed the tight integration with Microsoft Purview, and with Power BI. Azure Synapse Analytics comes with a default data store: the Azure Data Lake Storage Gen2, although it is possible to link more data stores to your Azure Synapse Analytics. The default data store supports Delta Lake, an open source storage layer that comes on top of the Data Lake Storage and brings ACID transactions to Apache Spark.

> ***NOTE*** **ACID**
>
> **ACID is the acronym for Atomicity, Consistency, Isolation, and Durability. One database implementation of ACID is not necessarily equivalent to another. For example, there are many levels of consistency possible in a data store. Nonetheless, ACID still provides more guarantees than BASE (Basically Available, Soft state, Eventual consistency).**

Azure Synapse Analytics offers two query engines: Apache Spark and SQL. We will review both technologies and their various flavors in this chapter. These query engines allow for batch processing, parallel processing, stream processing, and machine learning workload. The beauty of Azure Synapse Analytics is also the variety of languages you can work with. SQL is the first to come to mind, but there is also Scala, Python, R, and last but not least, C#. Finally, Synapse Pipelines enables data transformations, even in a parallel computing fashion, with a low code tool.

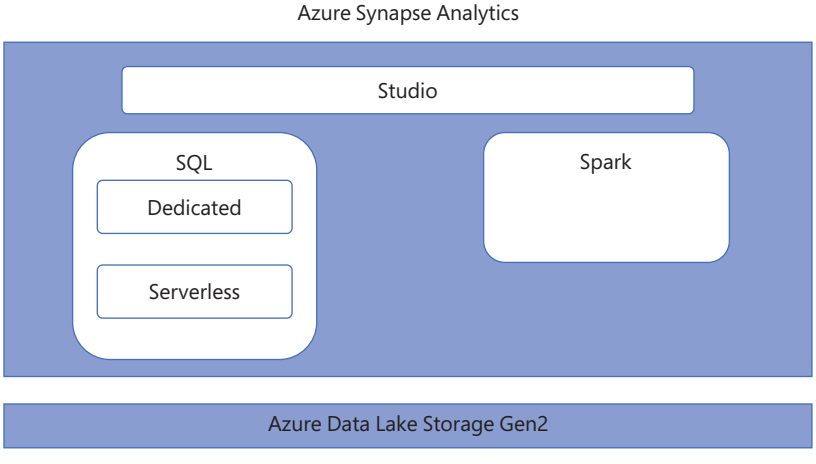


FIGURE 2-1 Overall architecture diagram of Azure Synapse Analytics

Concepts about Spark Pool

Apache Spark is a framework for in-memory parallel processing. It was first released in 2014, with the goal to provide a unified platform for data engineering and data science, as well as batch and streaming—a mission that resonates with Azure Synapse Analytics. Apache Spark is originally written in Scala, but also available in Python, SQL, and R. In Azure Synapse Analytics, Apache Spark is also available in C#.

Apache Spark runs on a cluster, as you can see in Figure 2-2.

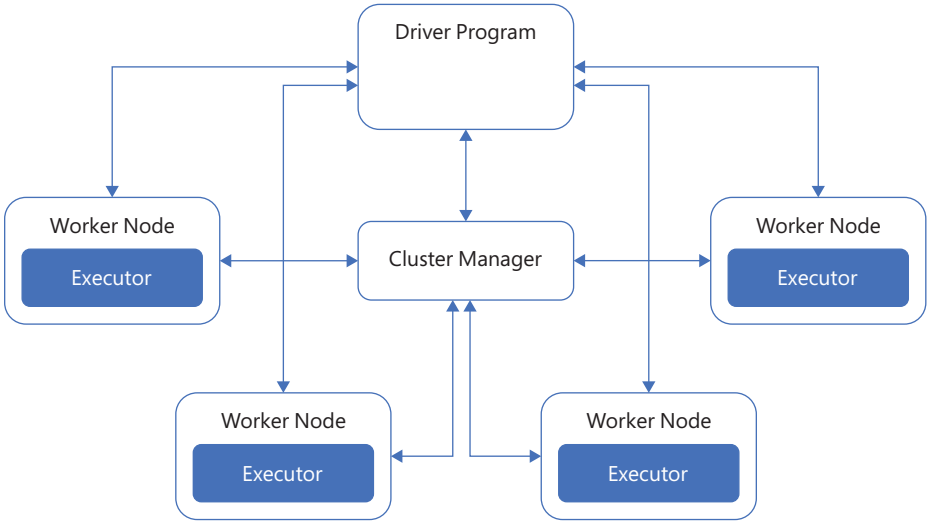


FIGURE 2-2 High-level overview of a Spark cluster

A Spark application is coordinated by the *driver program*. On a cluster, the *driver program* connects to the *cluster manager* to coordinate the work. In Azure Synapse Analytics, the *cluster manager* is YARN. The *cluster manager* is responsible for allocating resources (memory, cores, etc.) in the cluster. Once the *driver* is connected to the *cluster manager*, it acquires resources to create *executors*. *Executors* will be the ones doing the work: running in-memory computations and storing data.

An Azure Spark pool is not just a Spark instance on a cluster. Azure Synapse provides peripheral features:

- **Auto-Scale capabilities**—The Auto-Scale feature automatically scales up and down the number of nodes in a cluster. You can set a minimum and maximum number of nodes when creating a new Spark pool. Every 30 seconds, Auto-Scale monitors the number of CPUs and the amount of memory needed to complete all pending jobs, as well as the current total CPUs and memory, and scales up or down based on these metrics.
- **Apache Livy**—Apache Livy is a REST API for Apache Spark. Conveniently, it is included in Azure Spark pools. Apache Livy can be used to submit programmatically Spark jobs to the cluster.

- **Preloaded Anaconda libraries**—Apache Spark is great for parallel computing, but sometimes we don't need all that computing power, but instead want some more specialized functionalities for machine learning or visualization. Particularly during exploratory analysis, having Anaconda libraries preloaded in Azure Spark pools comes in handy.

An Azure Spark pool is a great tool when dealing with a large amount of data and when performance matters. Should it be data preparation, machine learning, or streaming data, a Spark pool is the right tool if you want to work in Python, SQL, Scala, or C#.

EXAMPLE OF HOW TO USE IT

To review how to use an Azure Spark pool, we will start by creating an Azure Synapse Analytics resource.

1. In your Azure portal, select **Create a resource**.
2. Search for and select **Azure Synapse Analytics**.
3. Select **Create**.
4. Once the resource is created, navigate to the resource and select **Open Synapse Studio**, as shown in Figure 2-3.

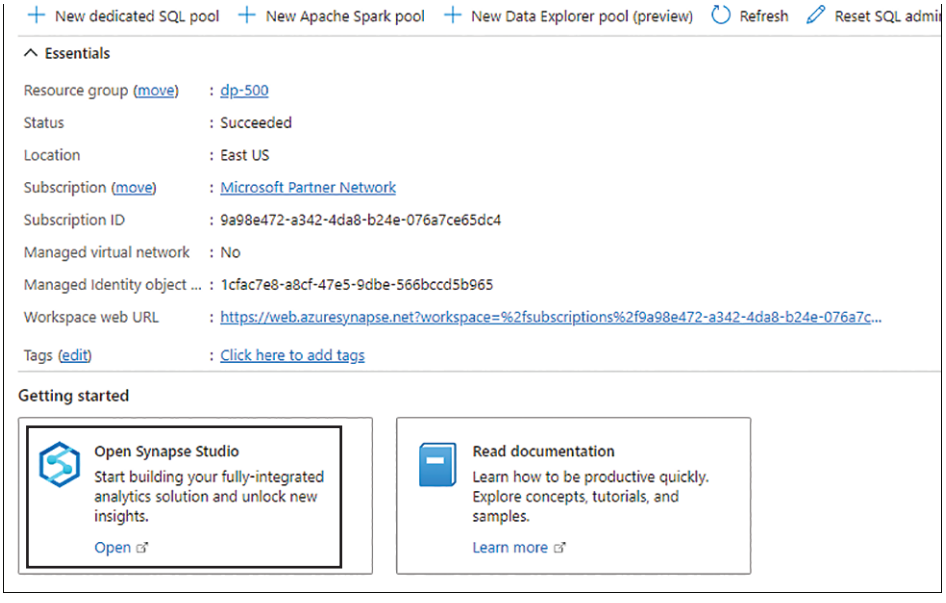


FIGURE 2-3 Synapse Resource view in the Azure portal

Now that we have our resource, let's create the Spark pool.

1. Go to **Manage**, as shown in Figure 2-4.
2. Navigate to **Apache Spark pools** > **+ New**.
3. Enter a name in **Apache Spark pool name**, like dp500.

FIGURE 2-4 The Manage menu in Azure Synapse Analytics Studio

4. Set **Isolated compute** to Disabled.
5. Set **Node size family** as Memory Optimized.
6. Set **Node size** to Small.
7. Set **Autoscale** to Enabled.
8. Set **Number of nodes** to 3 and 5.
9. Review that you have the same configuration as Figure 2-5.
10. Select **Review + create**.
11. Select **Create**.

Now that we have our compute, we can use it to query and analyze some data. To do that, we first need to get some data.

1. Navigate to https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page and download the January 2022 Green Taxi Trip Records.
2. In the Synapse Studio, navigate to the Data menu, as shown in Figure 2-6.

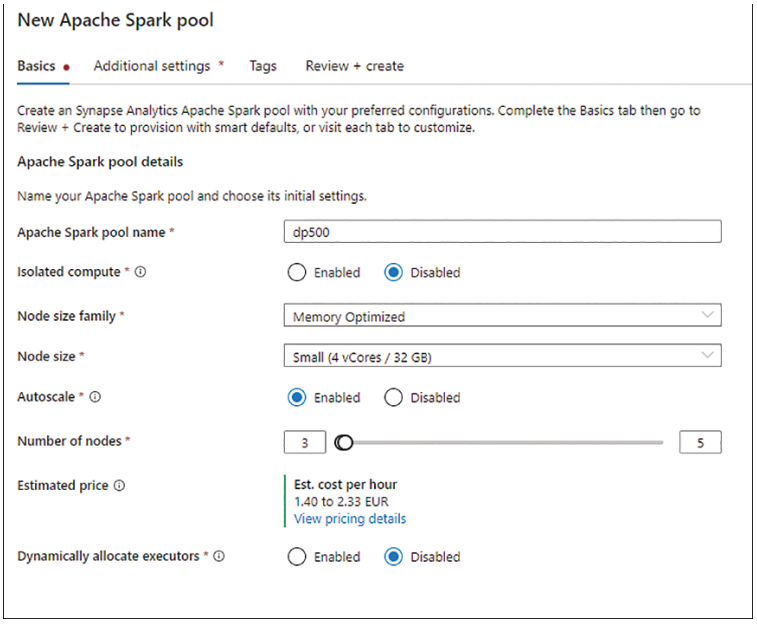
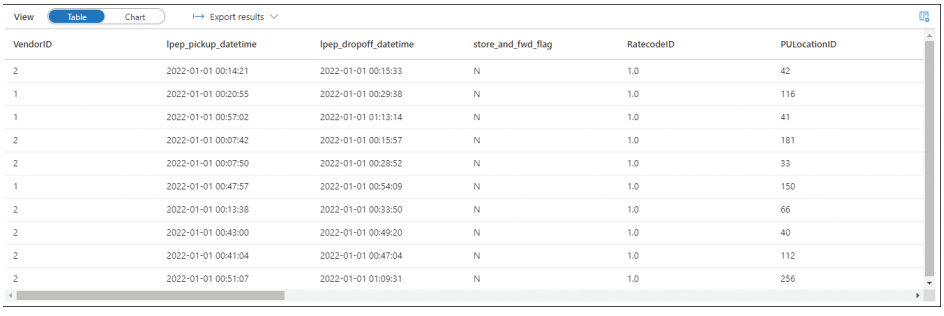


FIGURE 2-5 Configuration parameters for the Apache Spark pool

FIGURE 2-6 Data menu in Synapse Studio

3. Select the **Linked** tab and then select **Azure Data Lake Storage Gen2**.
4. Expand your primary container and select your primary file store.
5. Upload your Parquet file.
6. Right-click on the uploaded file and from the context menu select **New notebook** > **Load to Dataframe**.
7. In the new notebook, attach the Spark pool you created previously.
8. Run the first cell.

After a few minutes, the time for the Spark pool to spin up, you should see a table with the first 10 rows of the dataset, as shown in Figure 2-7. The data is now loaded in memory in a Spark DataFrame, so the analysis can proceed.


View Table Chart Export results

VendorID	lpep_pickup_datetime	lpep_dropoff_datetime	store_and_fwd_flag	RatecodeID	PULocationID
2	2022-01-01 00:14:21	2022-01-01 00:15:33	N	1.0	42
1	2022-01-01 00:20:55	2022-01-01 00:29:38	N	1.0	116
1	2022-01-01 00:57:02	2022-01-01 01:13:14	N	1.0	41
2	2022-01-01 00:07:42	2022-01-01 00:15:57	N	1.0	181
2	2022-01-01 00:07:50	2022-01-01 00:28:52	N	1.0	33
1	2022-01-01 00:47:57	2022-01-01 00:54:09	N	1.0	150
2	2022-01-01 00:13:38	2022-01-01 00:33:50	N	1.0	66
2	2022-01-01 00:43:00	2022-01-01 00:49:20	N	1.0	40
2	2022-01-01 00:41:04	2022-01-01 00:47:04	N	1.0	112
2	2022-01-01 00:51:07	2022-01-01 01:09:31	N	1.0	256

FIGURE 2-7 Results from reading the Parquet file with the Spark pool

Architecture of Synapse SQL

Before diving into the two flavors of Azure Synapse SQL, let's review the overall architecture of this part of Azure Synapse Analytics. Figure 2-8 represents the internal architecture of Azure Synapse SQL.

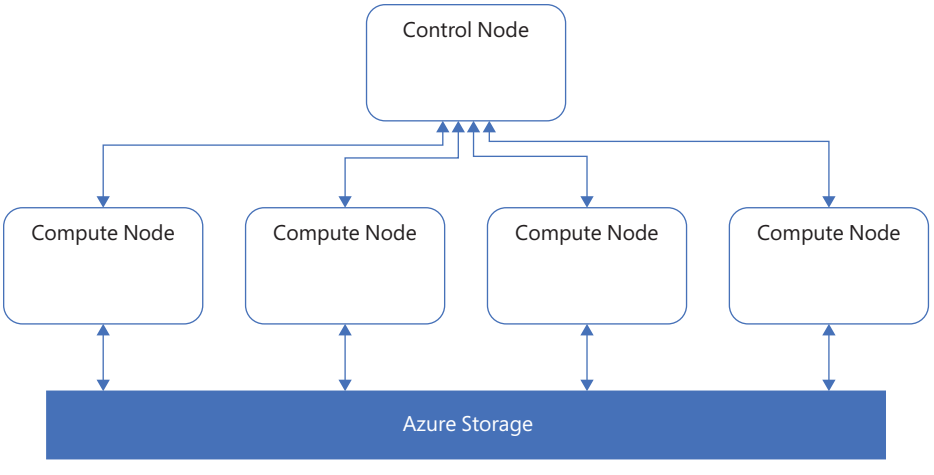


FIGURE 2-8 The internal architecture of Azure Synapse SQL

The control node is going to be the entry point of Azure Synapse SQL and the orchestrator of the distributed query to the compute node, similar to the driver node in a Spark pool. And as their name indicates, the compute nodes will do the computation.

It is important to note the decoupling of compute and storage, which will also be reflected in the pricing. You can scale your compute without affecting your storage, and vice versa.

The difference between serverless and dedicated is primarily in the number of nodes available. With serverless, the number of nodes is not predetermined and will scale automatically based on the computation needs. With dedicated, the number of nodes is preallocated.

A dedicated SQL pool also, beyond querying the data, allows ingestion of data, whereas serverless only allows querying the data. The ingestion of data will allow you to create tables in your SQL pool. Those tables will be distributed. There are three possible distributions: hash, round-robin, and replicate.

The hash distribution is typically used for large tables, such as fact tables. The round-robin is used for staging tables. And the replicate is used for small tables, such as dimensions.

EXAM TIP

You will often find questions about the optimal distribution for a use case. So it's important that you thoroughly understand each of the three distributions and when they would be appropriate to use.

NEED MORE REVIEW? **DEEP DIVE INTO DISTRIBUTIONS**

If you want to review the intricacies of the distributions in Azure Synapse SQL, see the following documentation: *https://learn.microsoft.com/en-us/azure/synapse-analytics/sql/overview-architecture#distributions*.

SQL serverless pool

Because a SQL serverless pool is serverless, that doesn't mean there is no server behind the scene, but it does mean that for you there is no infrastructure to set up or maintain. A SQL serverless pool will be automatically provisioned and ready to go when you create a Synapse Analytics workspace, and it will scale as needed. You will be charged by how much data you process.

Because of that, a SQL serverless pool is particularly suited for unpredictable workloads, such as exploratory analysis. Another perfect use case for SQL serverless pools is to create a logical data warehouse, to be queried by Power BI. Indeed, instead of provisioning a cluster so that Power BI can query your data lake, a SQL serverless pool is available whenever you need it.

NOTE **SQL SERVERLESS POOL BEST PRACTICES**

Make sure you review the best practices to get the best results out of SQL serverless pools: *https://learn.microsoft.com/en-us/azure/synapse-analytics/sql/best-practices-serverless-sql-pool.*

EXAMPLE OF HOW TO USE IT

To see how to use SQL serverless pools, let's open our Synapse Studio:

1. Go to the **Develop** menu, as shown in Figure 2-9.

FIGURE 2-9 Develop menu in Synapse Studio

2. Select **+** > **SQL script**, as shown in Figure 2-10.

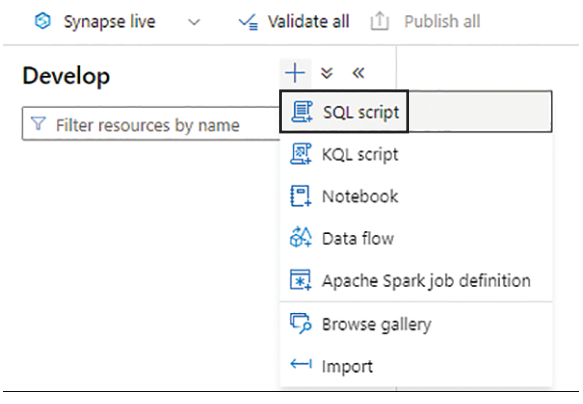


FIGURE 2-10 The Develop menu with its various capabilities

3. In the menu above the first line of the script, verify that you are connected to the **Built-in** compute. **Built-in** is the name of the SQL serverless pool automatically created for you when you create an Azure Synapse Analytics resource.

4. Using the `OPENROWSET` function, query the previously loaded taxi dataset, and display the top 10 rows:

```
SELECT TOP 10 *
FROM OPENROWSET(
        BULK 'https://yoursynapseresource.dfs.core.windows.net/yourfilestore/
synapse/workspaces/green_tripdata_2022-01.parquet',
FORMAT = 'PARQUET' ) AS [result]
```

5. Select **Run**.

After a few seconds—the start-up time is much less than for the Apache Spark pool—you should see a table with the first 10 rows of the dataset, as shown in Figure 2-11.

VendorID	lpep_pickup_d...	lpep_dropoff_...	store_and_fwd...	RatecodeID	PULocationID	DOLocationID	passenger_count	trip_distance	fare_amount
2	1640996061000...	1640996133000...	N	1	42	42	1	0.44	3.5
1	1640996455000...	1640996978000...	N	1	116	41	1	2.1	9.5
1	1640998622000...	1640999594000...	N	1	41	140	1	3.7	14.5
2	1640995662000...	1640996157000...	N	1	181	181	1	1.69	8
2	1640995670000...	1640996932000...	N	1	33	170	1	6.26	22
1	1640998077000...	1640998449000...	N	1	150	210	1	1.3	7
2	1640996018000...	1640997230000...	N	1	66	67	1	6.47	22.5
2	1640997780000...	1640998160000...	N	1	40	195	1	1.15	6
2	1640997664000...	1640998024000...	N	1	112	80	1	1.3	6
2	1640998267000...	1640999371000...	N	1	256	186	1	4.75	17

FIGURE 2-11 Results from querying the dataset with a SQL serverless pool

EXAM TIP

It is important to understand how the function `OPENROWSET` works and how to use it. If you need more review, please consult the documentation here: *https://learn.microsoft.com/en-us/sql/t-sql/functions/openrowset-transact-sql?view=sql-server-ver16*.

Concepts about SQL dedicated pools

Synapse SQL dedicated pools differ from serverless in two major ways: first, because of its dedicated nature it needs to be provisioned, and second, data can be ingested in a SQL dedicated pool.

When provisioning the pool, you will have to define its size in DWUs (data warehousing units). A DWU is an abstraction representing a unit that combines CPU, memory, and I/O.

To ingest data into the dedicated pool, you can use T-SQL to query data from external sources. This is possible thanks to PolyBase, a virtualization feature that allows you to query data from various sources without the need to install a client connection software.

Table partitioning is an important topic to review for SQL dedicated pools.

ROUND-ROBIN DISTRIBUTION

Round-robin will favor data writes, not reads, so it is suitable for rapidly loading data and therefore suitable for staging tables. It randomly distributes table rows evenly across all nodes.

Querying data distributed in a round-robin fashion, particularly with joins, will yield poor performance. To accomplish that, it's better to use hash distribution.

HASH DISTRIBUTION

Hash distribution won't distribute rows randomly, but rather it will use a deterministic hash function to assign each value to a node. If two values are the same, they will be assigned to the same node. This is important to remember when choosing the column used for partitioning. If a column has skew—that is, many times the same value—one node will be a hot spot (overused) and the other nodes will barely have any data on them. So it's important to choose a column evenly distributed. The hash distribution is typical for large fact tables. For smaller tables, like dimensions, there is a third option.

REPLICATED DISTRIBUTION

As its name indicates, replicated distribution will copy the table in each node. That could be seen as a waste of space, but in a distributed workload that's a huge performance boost, since data doesn't need to move around if each node already has it. This is only possible if the table is small, so typically this distribution is used for dimension tables. The general guidance is to use replicated for tables smaller than 2 GB.

Recommend appropriate file types for querying serverless SQL pools

Synapse serverless SQL pool scales automatically and doesn't need you to provision anything. That's why it's great for unpredictable workloads and logical data warehousing. You pay depending on how much data you query. Moreover, it's important to remember that Synapse is catering for your analytics needs, not your operational needs. This affects the access patterns appropriate for Synapse serverless SQL pools. For example, an online transaction processing (OLTP) workload, updating or reading all columns of a single row, is not the appropriate use case. Analytics needs mean online analytical processing (OLAP), in other words, querying a large number of rows, on a limited number of columns, often with some aggregations performed. The ideal access pattern will be important in your choice of file to query for Synapse Analytics.

Typically three different file types can be queried with SQL serverless pools: CSV, JSON, and Parquet. CSV (comma-separated values) files are common in many businesses. JSON (JavaScript Object Notation) files are common in web applications. And Parquet files are common for analytics applications.

The syntax is the same to query them, making use of the `OPENROWSET` function:

```
SELECT *FROM OPENROWSET(
    BULK 'https://mydatalake.blob.core.windows.net/data/files/*.csv',
    FORMAT = 'csv') AS rows
```

Why is Parquet recommended for analytics? It comes down to the access pattern. Parquet files store the data in a columnar format, whereas JSON and CSV store files in a row format. This

has two implications. First, accessing the data for OLAP queries is faster with columnar format, because the data we want to query (the whole column) is physically stored next to each other. With a CSV or JSON file, the data of the same column would be physically scattered. Second, the data is generally more homogenous in a column than in a row. Indeed, usually a row contains multiple data types, so there will be a mix of data. In a column the data has the same type and can be similar or even identical sometimes. This leads to greater potential for compression. That is why Parquet files are recommended for analytics workloads.

> ***NOTE*** **DELTA**
>
> **In 2020, a new file format named Delta was released open source. It extends Parquet and includes ACID transactions. It is compatible with Synapse Analytics and is the recommended file format if you work with Spark pools.**

How to query Parquet files

To review how to query a Parquet file, let's go back to our Synapse Studio:

1. Navigate to **Develop** > **+** > **SQL script**.
2. Make sure it is attached to **Built-in**, as a reminder, this is your serverless pool.
3. Using `OPENROWSET`, select the top 10 rows of your taxi dataset in a Parquet format:

```
SELECT
 TOP 10 *
FROM
OPENROWSET(
BULK 'https:// your-storage.dfs.core.windows.net/ your-filestore /synapse/
workspaces/green_tripdata_2022-01.parquet',
FORMAT = 'PARQUET'
) AS [result]
```

4. Select **Run**.

You should get the same results as before, as shown earlier in Figure 2-11.

Query relational data sources in dedicated or serverless SQL pools, including querying partitioned data sources

SQL pools, whether dedicated or serverless, will test your knowledge of T-SQL. Here you will learn how to query data, both with dedicated and serverless, and we will also consider partitioned data.

To review how to use SQL dedicated pools, let's go back to our Synapse Studio and:

1. Go to the **Manage** menu.
2. Under **Analytics pools** select **SQL pools** > **+ New**.
3. Set **Dedicated SQL Pool Name** to a name of your choice

4. Change the **Performance** level to DW100c; the default is DW1000c.
5. Select **Review + create > Create**.

The deployment of the dedicated SQL pool will take a few minutes. Next let's look at how to ingest the data into the dedicated pool:

6. Go to **Develop** > **+** > **SQL script**.
7. Set the **Connect to** option to your dedicated SQL pool, not to **Built-in**.
8. You can ingest from the data lake to your SQL dedicated pool with the `COPY` statement. You will need to replace the file address with your own address:

```
COPY INTO dbo.TaxiTrips
FROM 'https://your-storage.dfs.core.windows.net/your-filestore/synapse/workspaces/
green_tripdata_2022-01.parquet'
WITH
(
FILE_TYPE = 'PARQUET',
MAXERRORS = 0,
IDENTITY_INSERT = 'OFF',
AUTO_CREATE_TABLE = 'ON'
)
```

9. You can now run a `SELECT` on the newly created table in your dedicated pool. You will see that the table has been populated with the data from the Parquet file.

```
SELECT TOP 10 * FROM dbo.TaxiTrips
```

10. Select **Run**.

You should now see the first 10 rows of the table, as shown in Figure 2-12.

VendorID	lpep_pickup_d...	lpep_dropoff_...	store_and_fwd...	RatecodeID	PULocationID	DOLocationID	passenger_count	trip_distance	fare_amount
2	164150222400...	164150244400...	N	1	42	42	2	0.86	5
1	164155400300...	164155446500...	N	1	75	239	1	1.4	7.5
2	164157484400...	164157512400...	N	1	74	41	2	0.84	5
2	164158892200...	164158921000...	N	1	255	255	1	0.58	5
1	164163993800...	164164217000...	N	1	55	181	1	0	35.88
2	164165803000...	164165957600...	N	4	16	265	2	22.99	109.5
2	164167201000...	164167252000...	N	1	74	42	1	1	7
2	164172666000...	164172695300...	N	1	7	7	1	0.56	5
1	164175019800...	164175177100...	N	1	7	17	1	0	24.38
2	164180474500...	164180525200...	N	1	74	152	1	1.2	7.5

FIGURE 2-12 Results from querying the populated table with Synapse Dedicated Pool

11. Don't forget to pause your dedicated SQL pool to limit the cost of this exercise.

Oftentimes, though, Parquet files are partitioned. To review and reproduce this use case, we'll use the SQL serverless pool. So follow these steps:

1. Navigate to https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page and download the February and March 2022 Green Taxi Trip Records.
2. In Synapse Studio, select **Data** > **Linked** > **Azure Data Lake Storage Gen2**.
3. Open your primary data store.

4. With the **+ New folder** command, create a folder structure in your data store representing YEAR/MONTH, as shown in Figure 2-13.

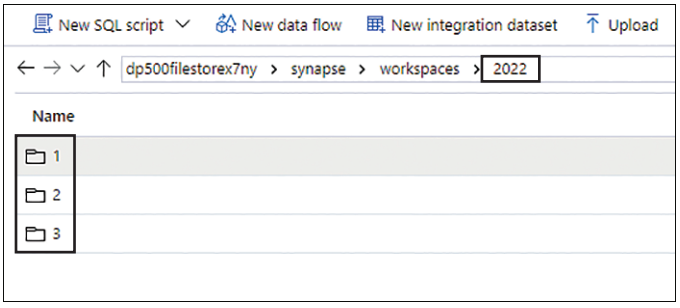


FIGURE 2-13 The file store hierarchy in Synapse Studio

5. Upload the datasets in their corresponding folders; you should have one Parquet file per month.
6. To leverage the partitioning in your query, try to count how many records are in each partition, using the `OPENROWSET` function, the Parquet format, and the `*` wildcard.
7. Run the query for Partitions 1 and 2:

```
SELECT
COUNT(*)
FROM
OPENROWSET(
BULK 'https:// your-storage.dfs.core.windows.net/ your-filestore /synapse/
workspaces/2022/*/*.parquet',
FORMAT = 'PARQUET'
) AS TaxiTrips
WHERE TaxiTrips.filepath(1) IN ('1', '2')
```

8. Run the query for Partition 1:

```
SELECT
COUNT(*)
FROM
OPENROWSET(
BULK 'https://your-storage.dfs.core.windows.net/your-filestore/synapse/
workspaces/2022/*/*.parquet',
FORMAT = 'PARQUET'
) AS TaxiTrips
WHERE TaxiTrips.filepath(1) IN ('1')
```

9. Check that the results are different.

You now know how to query a Parquet file containing partitions.

Use a machine learning PREDICT function in a query

Finally, Synapse SQL dedicated pools can be used to consume a machine learning model. For example, with our taxi dataset, we could have a model to predict the duration of the trip, based

on the pickup location and the time of day. Having the ability to score, or consume, the model directly in Synapse Analytics is useful because we don't need to move the data outside of our data analytics platform. To score our model, we will use the PREDICT function.

However, a Synapse SQL dedicated pool doesn't have the ability to train a machine learning model. So to use the PREDICT function, we will need to have the model trained outside of Synapse SQL. We could, for example, train the model in a Synapse Spark pool or in another product like Azure Machine Learning.

The trained model will need to be converted to the ONNX (Open Neural Network Exchange) format. ONNX is an open source standard format, with the very purpose of enabling exchange of models between platforms.

Load a model in a Synapse SQL dedicated pool table

The model has to be stored in a dedicated SQL pool table, as a hexadecimal string in a `varbinary(max)` column. For instance, such a table could be created with this:

```
CREATE TABLE [dbo].[Models]
(
[Id] [int] IDENTITY(1,1) NOT NULL,
[Model] [varbinary](max) NULL,
[Description] [varchar](200) NULL
)
WITH
(
DISTRIBUTION = ROUND_ROBIN,
HEAP
)
GO
```

where **Model** is the `varbinary(max)` column storing our model, or models, as hexadecimal strings.

Once the table is created, we can load it with the COPY statement:

```
COPY INTO [Models] (Model)
FROM '<enter your storage location>'
WITH (
FILE_TYPE = 'CSV',
CREDENTIAL=(IDENTITY= 'Shared Access Signature', SECRET='<enter your storage key here>')
)
```

We now have a model ready to be used for scoring.

Scoring the model

Finally, the PREDICT function will come into play to score the model. Like any machine learning scoring, you will need the input data to have the same format as the training data.

The following query shows how to use the PREDICT function. It takes the model and the data as parameters.

```
DECLARE @model varbinary(max) = (SELECT Model FROM Models WHERE Id = 1);
SELECT d.*, p.Score
```

```
FROM PREDICT(MODEL = @model,
DATA = dbo.mytable AS d, RUNTIME = ONNX)
WITH (Score float) AS p;
```

NEED MORE REVIEW? **THE PREDICT FUNCTION**

The full documentation for the PREDICT function can be found here: *https://learn.microsoft.com/en-us/sql/t-sql/queries/predict-transact-sql?view=sql-server-ver15*.

Skill 2.2: Ingest and transform data by using Power BI

Power BI includes Power Query, which is an extract-transform-load (ETL) tool that uses the M language. M is a functional, case-sensitive language that, unlike DAX, does not resemble Excel formula language in any way, and differs from DAX in important ways, too. In this section, we'll look at the problems you may need to solve when working with large amounts of data in Power Query.

When the volume or number of data sources is significant, you may face performance degradation. There are tools within Power Query that will help you identify the performance problems, and later we'll review the techniques you can use to improve performance.

In addition to Power BI Desktop, Power Query is available in Power BI dataflows, and we'll review when you'd want to use dataflows and what you'd need to consider when using them.

Combining data from different data sources will lead to data privacy issues, which we'll also discuss later in this chapter.

Finally, we'll discuss how you can use Advanced Editor to write your own queries and functions, and how you can query some of the more complex data sources by using Power Query.

NOTE **COMPANION FILE**

Most of the Power Query queries shown in this chapter are available in the companion PBIX file.

This skill covers how to:

- Identify data loading performance bottlenecks in Power Query or data sources
- Implement performance improvements in Power Query and data sources
- Create and manage scalable Power BI dataflows
- Identify and manage privacy settings on data sources
- Create queries, functions, and parameters by using the Power Query Advanced Editor
- Query advanced data sources, including JSON, Parquet, APIs, and Azure Machine Learning models

Identify data loading performance bottlenecks in Power Query or data sources

Several reasons could be responsible for poor performance when connecting to data in Power BI. Power BI Desktop has a few features that can help identify those issues.

View native query

When you get data in Power BI from some data sources, like databases, Power Query will do its best to translate the transformations you perform into the native language of the data source—for example, SQL. This feature of Power Query is known as *query folding*. Most of the time, this will make getting data more efficient. For instance, if you connect to a database and get a subset of columns from a table, Power Query may only retrieve those columns from the data source instead of loading all columns and then locally removing the ones you don't want.

In some cases, it may be possible to view the query that Power Query sent to the data source to retrieve the data you wanted. For this, you need to right-click a query step in Power Query Editor and select **View Native Query**. The window that opens looks like Figure 2-14.

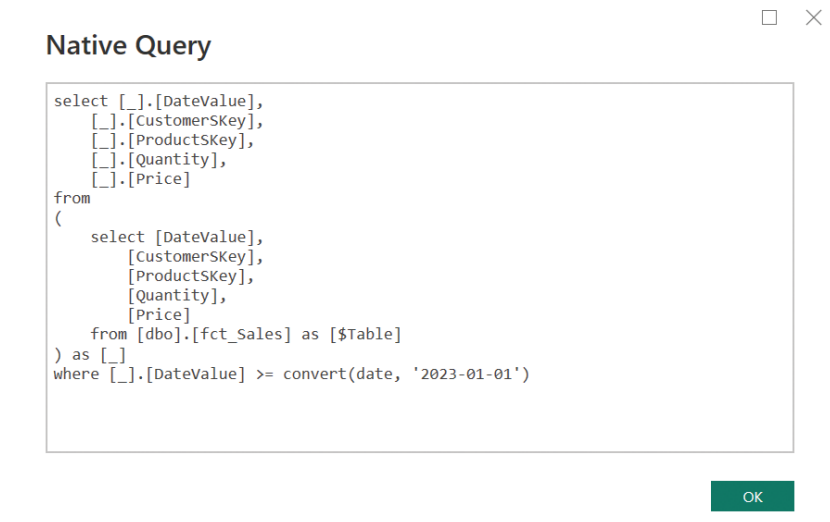


FIGURE 2-14 Native Query window

In the query shown in Figure 2-14, we connected to a SQL Server database, applied a filter, and selected a few columns. Because these operations can be translated to SQL, Power Query decided to do the transformations in the source instead of performing them after loading the whole table, which led to better performance.

You cannot edit the native query; it is provided for your information only. If you want Power BI to issue a specific query, you must provide a SQL statement when connecting to a database.

If the **View Native Query** option is grayed out, it means that the data source does not support query folding or that some query step could not be translated into the data source's native language. For example, if we applied the *Clean* transformation to a text column, the query would not fold, because there is no direct equivalent in SQL yet.

IMPORTANT **POWER QUERY STEPS ORDER**

The order of steps in Power Query matters. If you must have a transformation that cannot be folded, it's best to reorder your steps to fold as many steps as possible.

Query diagnostics

Power BI contains the query diagnostics toolset, which can help you identify performance bottlenecks. Query diagnostics allow you to see the queries that you emit while authoring or refreshing a dataset. They are especially useful for working with data sources that support query folding. By using query diagnostics, you can look at all queries that happen during data refreshes or while you author queries, or you can analyze a single step in detail.

To learn how to use query diagnostics, you'll connect to an OData feed first. It's a feed from Microsoft based on their fictitious AdventureWorks company.

1. Create a new Power BI Desktop file.
2. Select **Get data** (or **New Source** if you're already in Power Query Editor) > **OData feed**.
3. Enter **https://services.odata.org/AdventureWorksV3/AdventureWorks.svc** in the **URL** box and select **OK**.
4. If prompted, in the credentials window, ensure **Anonymous** is selected and select **Connect**.
5. Select the **CompanySales** check box in the Navigator window and select **Transform Data** or **OK** if you're already in Power Query Editor.

Now that you are connected to an OData feed, you can apply some transformations and see the effect on our query. To start recording traces in Power Query, select **Start Diagnostics** on the **Tools** ribbon; when finished, select **Stop Diagnostics**. Alternatively, you can analyze a single step—for this, you must select the **Diagnose Step** button on the **Tools** ribbon, or you can right-click a step and select **Diagnose**. We are going to analyze a single step in the following way:

1. Filter the **ProductCategory** column to **Bikes** by using the filter button on the column header.
2. Right-click the **ProductCategory** column header and select **Remove**.
3. In the **Query Settings** pane, right-click the last step and select **Diagnose**.

After Power Query finishes recording the traces, it creates a new query group called Diagnostics (as in Figure 2-15, which contains several queries whose names start with *CompanySales_Removed Columns*, all ending with the current date and time). The queries are sources

from JSON files stored locally on your computer. The *Detailed* query contains more rows and columns than the *Aggregated* query, which is a summary query.

Among other information available in the recorded traces, you will see the time it took for a query to run and whether a native query was sent to a data source, which can help you understand if query folding took place. In Aggregated and Detailed queries, you can find the **Data Source Query** column, which contains the query sent to the data source, if available.

Occasionally, you won't be able to see the native query by using the **View Native Query** feature discussed earlier in this chapter, but you will see a native query sent to a data source when using query diagnostics. We can check whether query folding took place by following these steps:

1. In the *Aggregated* diagnostics query, filter the **Operation** column to only include **CreateResult**.
2. Go to the **Data Source Query** column and select the only column cell. You should see the result shown in Figure 2-15.

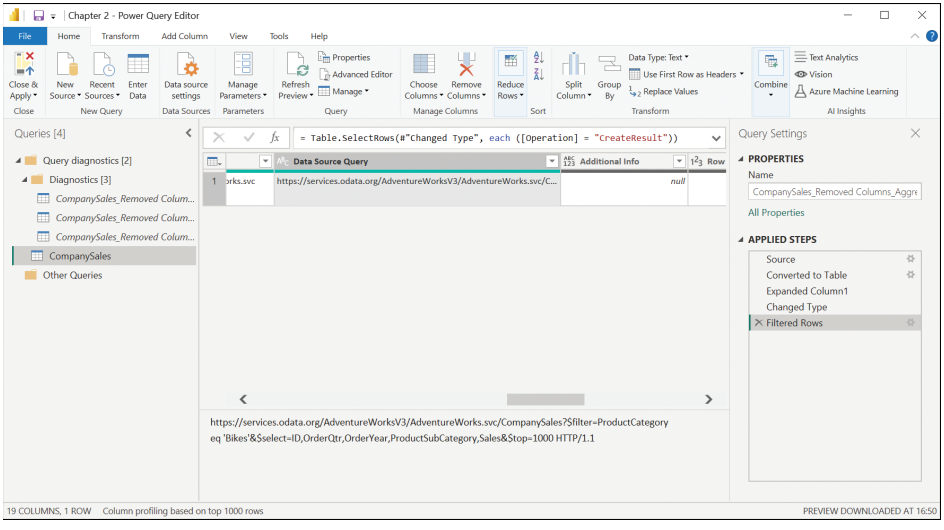


FIGURE 2-15 Native query sent to OData feed

The full query is as follows:

```
https://services.odata.org/AdventureWorksV3/AdventureWorks.svc/CompanySales?$filter=
ProductCategory eq
'Bikes'&$select=ID,OrderQtr,OrderYear,ProductSubCategory,Sales&$top=1000 HTTP/1.1
```

Note that query folding occurs; the filter we placed on the **ProductCategory** column is included in the query and the **ProductCategory** column is not included in the query result. If you only relied on the **View Native Query** feature, you would not see the query because the option would be grayed out.

Some query diagnostics may require that you run Power BI Desktop as administrator. If you are unable to record some traces, like full data refreshes, due to IT policies, you can still record traces when previewing and authoring queries in Power Query Editor. For this, go to **File** > **Options and settings** > **Options** > **Global** > **Diagnostics** > **Query diagnostics** and select **Enable in Query Editor**.

NEED MORE REVIEW? **USING QUERY DIAGNOSTICS**

For advanced information on how you can use the feature, including details on how to understand and visualize the recorded traces, see "Query Diagnostics" at *https://learn.microsoft.com/en-us/power-query/QueryDiagnostics*.

Implement performance improvements in Power Query and data sources

If your Power Query queries need to be improved in terms of performance, the techniques you'll use will highly depend on the data sources you're using and how you're using them. In this section, we'll explore a few methods of improving the data loading performance, divided into four parts:

- General advice
- Working with files
- Working with foldable data sources
- Improving the merge operations

General advice

In almost every situation, you want to load only the data you need. You should filter the data and remove columns as early as possible. Many operations in Power Query are performed in memory, and less data translates into better performance.

If you can choose, in many cases you should use table functions instead of operating on lists and records because Power Query includes some built-in optimizations that can make use of table functions.

When working with tables or lists that are referenced multiple times, you may improve performance by using `Table.Buffer` or `List.Buffer`, respectively. Buffering functions store data in memory and ensure that data isn't read multiple times from the source. When buffering data, you should note two things: first, buffering prevents query folding, so if you're working with foldable data source and buffer a table, then further transformations won't be folded even if they could be folded without buffering. Second, only scalar values are buffered; if your table or list includes nested structures such as records or other lists or tables, then those values won't be buffered, and if you need them later, then your data source will be read again.

Depending on the nature of your data sources and how you're using them, you may want to edit the **Parallel loading of tables** parameter in the current file data load settings

(**File** > **Options and settings** > **Options** > **Current File** > **Data Load**), where the default value is 6, meaning six tables are going to be loaded in parallel at most. If your data model contains more than six tables, which take a long time to refresh, you can try to set a custom parameter value higher than 6, allowing more tables to load in parallel, and your data model may refresh more quickly as a result. On the other hand, if you're getting some data from a web API and then reference the data a few times, you may want to disable parallel loading, because that way the API may be called fewer times, resulting in better data refresh performance.

In case you've got a slow data source that's updated less frequently than you refresh your dataset, you may benefit from loading data to a dataflow first and then loading it to your dataset. Dataflows are discussed in the next section.

Working with files

If you work with Excel files, then you should ensure the third parameter of `Excel.Workbook` is set to `true` to leave the column types untyped, which can make reading the files quicker.

If you can choose the file types you work with, then instead of Excel you may want to choose CSVs. While sometimes Excel files may be smaller than the equivalent CSV files, the latter are simpler in structure and therefore faster to read.

While Parquet and CSV files offer approximately the same performance, Parquet files are smaller, which may be relevant when you're using a gateway to access files from your company's network.

Working with foldable data sources

If you can make use of incremental refresh, then it may decrease the data refresh times substantially. We'll discuss incremental refresh in Chapter 3, "Implement and Manage Data Models."

In general, you should push as many transformations as you can as close to the data source as possible, because in many cases data sources may perform data operations more efficiently than Power Query. Some transformations may still fold, even if the *View native query* option of the step is grayed out, as discussed earlier in this chapter in the "Query diagnostics" section. For example, if you need to filter some SQL data based on a data source of a different type, like Excel, then instead of merging tables, you should use a native query and make use of the `WHERE` clause in SQL. For example, the following query applies a filter on the `pbi.Product` table from a SQL database based on a list of items from a different Power Query query, `ProductSubCategories`:

```
let
    FilterItems = "'" & Text.Combine(List.Buffer(ProductSubCategories), "', '") &
"'",
    Source = Sql.Databases("mydatabase.database.windows.net"),
    sakes = Source{[Name="sales"]}[Data],
    Result = Value.NativeQuery(sales, "SELECT * FROM pbi.Product WHERE [Product
Type] IN (@Items)", [Items = FilterItems])
in
    Result
```

Improving the merge operations

When you merge two tables and one table has a unique key, you should add a key to it. While it's possible to use the `Table.AddKey` function in Power Query, doing so won't guarantee uniqueness; it's preferable to remove duplicates from the key column or columns, which will add a key automatically and improve the merge performance.

While you should not keep data you don't need in general, it's especially important in case of merges; since merges take place in memory, fewer columns mean quicker merges. You can remove columns before the merge or immediately after, and the performance gains are going to be similar.

If you work with data that has its keys pre-sorted, then instead of `Table.NestedJoin`, you can use `Table.Join` with `JoinAlgorithm.SortMerge` as the last parameter. Note that if your data isn't sorted, then you're going to get unexpected results without any error message.

Create and manage scalable Power BI dataflows

In addition to Power BI Desktop, Power Query can be found in the Power BI service: you can prep, clean, and transform data in dataflows. Dataflows can be useful when you want your Power Query queries to be reused across your organization without necessarily being in the same dataset. For this reason, you cannot create a dataflow in your own workspace, because only you have access to it.

To create a dataflow in a workspace, select **New** >**Dataflow**. From there, you have several choices:

- **Add new tables**—Define new tables from scratch by using Power Query.
- **Add linked tables**—Linked entities are tables in other dataflows that you can reuse to reduce duplication of data and improve consistency across your organization.
- **Import model**—If you have a previously exported dataflow model file, you can import it.
- **Create and attach**—Attach a Common Data Model folder from your Azure Data Lake Storage Gen2 account and use it in Power BI.

The Power Query Online interface looks similar to Power Query Editor in Power BI Desktop and is shown in Figure 2-16.

Once you finish authoring your queries, you can select **Save & close** and enter the name of the new dataflow. After saving, you'll need to refresh it by selecting **Refresh now** from the dataflow options in the workspace—otherwise it won't contain any data. When a dataflow finishes refreshing, you can connect to it from Power BI Desktop and get data from it.

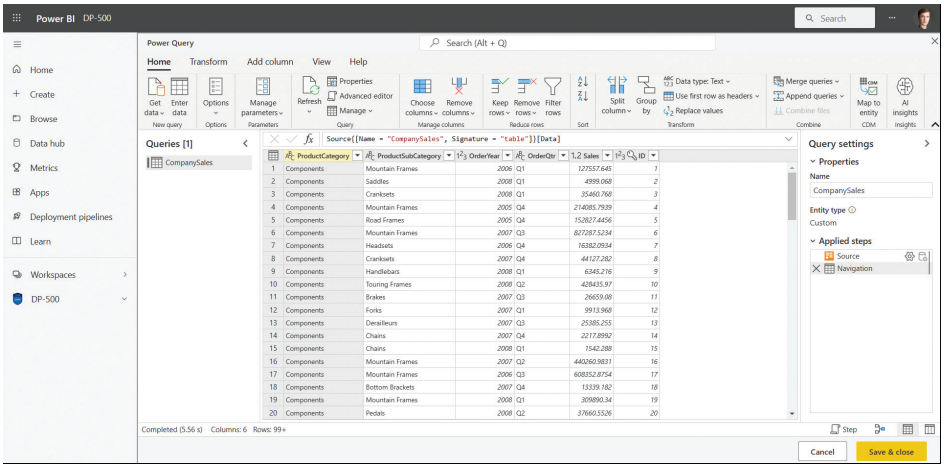


FIGURE 2-16 Power Query interface when editing a dataflow

If you transform data in dataflows, for best performance you may want to separate data extraction and transformation into different dataflows, which is especially helpful in case of slow data sources. If you then enable the enhanced compute engine in the dataflow settings, the transformations will be performed in a more efficient way.

When you used linked tables, the dataflow that depends on a linked table will be automatically refreshed when you refresh the dataflow that contains the original table.

Similar to datasets, you can configure incremental refresh in dataflows. We'll review incremental refresh in the next chapter.

Identify and manage privacy settings on data sources

When you combine data from different data sources, it is important to set the privacy levels correctly. Privacy levels determine the rules according to which data will be combined. These rules may affect the performance of queries, and in some cases, queries will not be executed at all if it is not permitted by privacy levels. To illustrate what happens in an example, we are going to filter an OData table by using data from a CSV file by performing the following steps:

1. Get the data from **https://raw.githubusercontent.com/DaniilMaslyuk/DP-500/main/ProductSubcategory.csv**.
2. Right-click the **Column1** column header and select **Drill Down**.
3. Create a new parameter by selecting **Manage Parameters** > **New Parameter**, as shown in Figure 2-17.

Index

Symbols

A

B

C

D

E

F

G

H

I

J-K

L

M

N

O

P

Q

R

S

T

U

V

W

X-Y-Z