



Implementing Analytics Solutions Using Microsoft Fabric

Exam Ref DP-600

Daniil Maslyuk
Johnny Winter
Štěpán Rešl

Exam Ref DP-600 Implementing Analytics Solutions Using Microsoft Fabric

**Daniil Maslyuk
Johnny Winter
Štěpán Rešl**

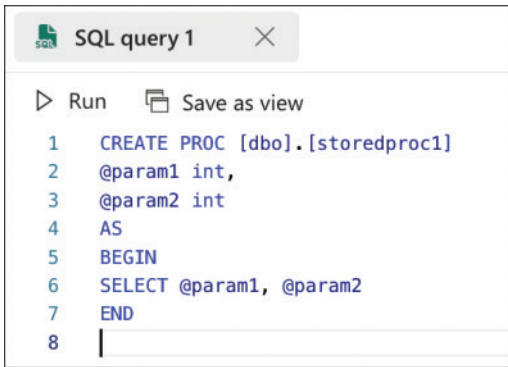


FIGURE 2-35 Create a stored procedure template

NEED MORE REVIEW? STORED PROCEDURES

For more information about stored procedures, please visit learn.microsoft.com/sql/t-sql/statements/create-procedure-transact-sql.

NOTE AZURE DATA STUDIO AND SQL MANAGEMENT STUDIO

A SQL endpoint and warehouse can be connected using Azure Data Studio or SQL Management Studio. These tools provide you with various ready-made templates for creating views, functions, and stored procedures from their environment.

Enrich data by adding new columns or tables

As you prepare the data based on the input scenarios, it may sometimes correspond to how it will need to look in the end. Often, though, additional columns or tables will need to be added and existing ones modified or removed. Microsoft Fabric, within its ingest items and T-SQL, allows you to enrich the data and thus compile the resulting views containing precisely what is needed.

Remember that Warehouse Explorer and SQL endpoint currently do *not* support ALTER TABLE ADD COLUMN within the lakehouse and warehouse data items. Therefore, extending tables with additional columns by T-SQL is impossible; instead, you must delete the table and create it again. Within Lakehouse Explorer, however, you can edit the Delta tables schema using notebooks or SparkJobs.

The `withColumn()` and `select()` functions can extend existing DataFrames. The `select()` function allows you to define the columns you want to keep from the existing DataFrame. At the same time, it allows the columns defined below to be renamed, be retyped, or to perform some function with them, such as `explode()`. This feature allows you to select the same column

twice, expanding the DataFrame with a new column. The function `withColumn()` returns a new DataFrame by adding a column or replacing the existing one with the same name.

Consider a few examples of using these functions:

```
df.select("id","name") # Will select only two columns from whole DataFrame
df.select("*",(df.UnitPrice-df.ProductionPrice).alias('ProductMargin')) # Calculates new
column based on two existing and will contain all previous columns
df.withColumn('ProductMargin', df.UnitPrice - df.ProductionPrice) # Calculates new
column based on two existingwithColumn('UnitPrice', df.UnitPrice + 10) # Replaces
current column UnitPrice with new values
```

NEED MORE REVIEW? FUNCTION EXPLODE()

For the Spark definition of function `explode()`, please visit spark.apache.org/docs/3.1.3/api/python/reference/api/pyspark.sql.functions.explode.html.

To modify the existing schema of existing tables using the schema of extended DataFrames, you can use two options parameters of the `options()` function:

- `mergeSchema` Expands the existing schema with the schema of the Spark frame being written
- `overwriteSchema` Overwrites the schema of the existing Delta table with the schema of the written Spark frame

For example, you could use:

```
df.write.mode("overwrite").option("mergeSchema", "true").saveAsTable("sales")
df.write.mode("overwrite").option("overwriteSchema", "true").saveAsTable("sales")
```

IMPORTANT REMOVING EXISTING COLUMNS BY `overwriteSchema`

Suppose an existing column is eliminated by `overwriteSchema` but not existing Parquet (like meanwhile Append mode). In that case, the data will not be discarded and will still be part of existing Parquet files. Only from the schema's point of view does the given column no longer exist, and it is not usually possible to query it.

As shown in the previous section, you can create a table using **Copy Activity**, **DataFlow Gen2**, or **notebooks** directly based on the data. You can also ingest data using a data pipeline, dataflow, or notebook. At the same time, you can create a new table using **T-SQL within the Warehouse Explorer**.

1. Select **New SQL query** to expand it.
2. Select **Table** (Figure 2-36).

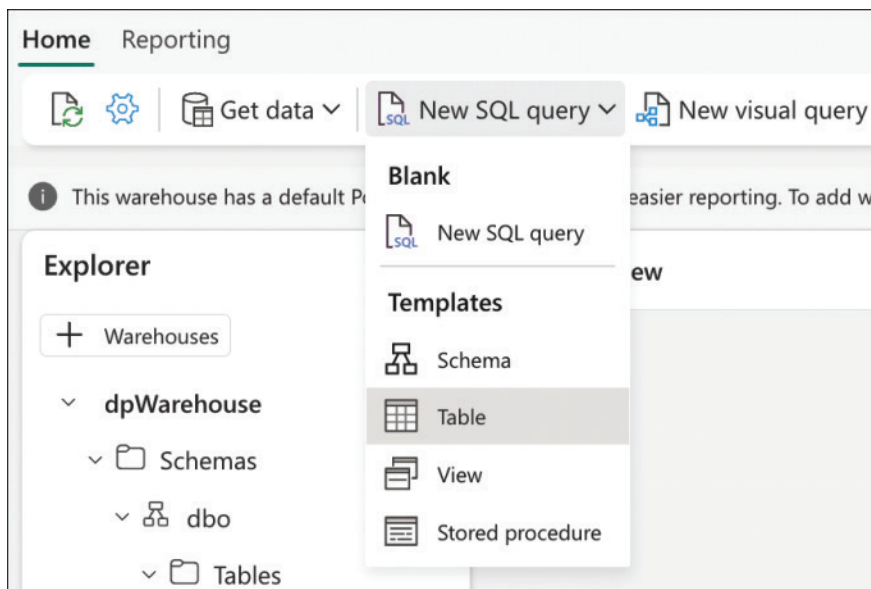


FIGURE 2-36 All current T-SQL template options in Warehouse Explorer

3. Edit the inserted code as shown in Figure 2-37.

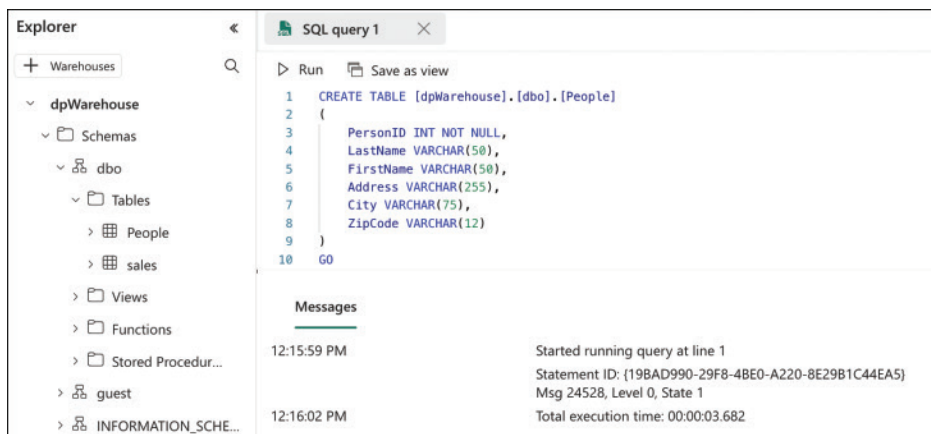
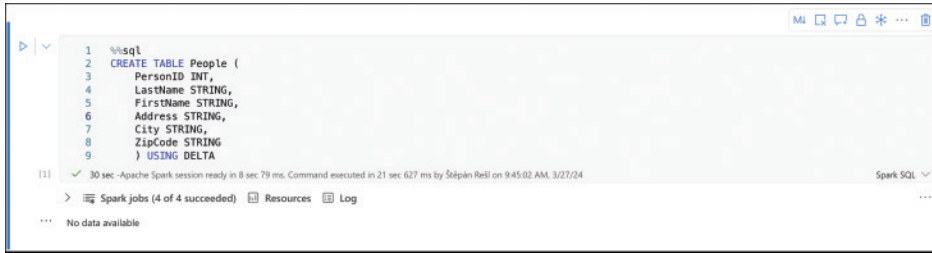


FIGURE 2-37 Creating a table with T-SQL

4. Select **Run** to create the table.

Unfortunately, this same progression does not work within SQL endpoint in Lakehouse Explorer. If you want to create a table without data, you can use PySpark and a similar syntax in SparkSQL. Changing from SQL data types to Spark data types is important; for example, use the `STRING` data type instead of `VARCHAR` (Figure 2-38).

A screenshot of a PySpark SQL interface. The main window displays a SQL command to create a table named 'People'. The command is:

```
1 %sql
2 CREATE TABLE People (
3     PersonID INT,
4     LastName STRING,
5     FirstName STRING,
6     Address STRING,
7     City STRING,
8     ZipCode STRING
9 ) USING DELTA
```

 Below the command, a status bar indicates:

```
(1) ✓ 30 sec - Apache Spark session ready in 8 sec 79 ms. Command executed in 21 sec 627 ms by Salspán Reili on 9:45:02 AM, 3/27/24
```

 At the bottom, there are tabs for 'Spark jobs (4 of 4 succeeded)', 'Resources', and 'Log'. The 'Spark jobs' tab is active, showing 'No data available'.

FIGURE 2-38 Creating a table with PySpark

Skill 2.2: Copy data

This section will show how to add your copy tasks to a pipeline and then how you can schedule either the pipeline or those individual tasks to run on a schedule.

Copying data or transferring it between individual locations always involves decisions—the first being whether copying is necessary or whether an alternative to duplicating data is available. If copying is the only option, then you must select an appropriate method. If the copying is to be carried out regularly, then the possibilities of orchestrating the entire solution must also be addressed so that the timing is correct and there are no collisions or possible copying of incomplete data.

This skill covers how to:

- Choose an appropriate method for copying data from a Fabric data source to a lakehouse or warehouse
- Copy data by using a data pipeline, dataflow, or notebook
- Implement fast copy when using dataflows
- Add stored procedures, notebooks, and dataflows to a data pipeline
- Schedule data pipelines
- Schedule dataflows and notebooks

Choose an appropriate method for copying data from a Fabric data source to a lakehouse or warehouse

Copying data always involves a critical decision about the method to use. It is always necessary to know what happens to the data during transmission, whether it should be enriched or modified during the process, and who the target audience will be who will create or manage the solution.

It would be best if you also decided whether data duplication is necessary. This topic is quite critical for Microsoft Fabric, especially from the Lakehouse Explorer point of view, because it can directly refer the user to the use of shortcuts. Thanks to them, you do not need to duplicate the data. Transformations that will be performed afterward will automatically work the most current variant of data because a shortcut provides a link to the location of the data and thus the resulting data does not have to be copied and maintained twice. However, the Warehouse Explorer does not support this.

In case it is necessary to duplicate data or perform any transformation directly during the move, it is important to validate if there is a connector for the data source. Microsoft Fabric provides three options: DataFlow Gen2, pipelines, and notebooks can access data within OneLake and thus can copy it. Of course, when you select a lakehouse as a data source, for example, the Copy Activity within a pipeline offers only lakehouses in the current workspace. Still, you can use the SQL endpoint of the given lakehouse and connect to it using the **Azure SQL Database connector**. The connection to the SQL endpoint can also be used within warehouse stored procedures, which can load data from the lakehouse to the warehouse.

You should also consider who will manage the resulting solution. More code means more freedom in solving the problem, but at the same time, there are higher demands on the technical skills of the people managing the given solution. Therefore, look at the team composition for which you're selecting and creating a solution. If users understand Power Query and create transformations, notebooks with PySpark will probably be entirely new to them. It would require high input resources to use or manage it entirely.

Lastly, consider whether changing the data storage style would be best, taking into account the data format and possibly partitions and folder structures. For example, DataFlow Gen2 currently supports saving the result of transformations only as a table. If these JSON files need to be transferred between lakehouses without changes, the DataFlow Gen2 option will not be very suitable. In this case, the Copy Activity within the data pipeline, which allows files to be moved, would serve much better. At such a moment, files can only be copied within one workspace because a SQL endpoint would not help with the files, and storing data in specific partitions would not be possible. However, a notebook is not limited to data in one workspace and can be highly efficient, although it does require working with PySpark or another of its supported languages.

Of course, extreme emphasis should be placed on the thriftiness of the chosen solution so that it is not using an excessive amount of capacity units. If you find it does, search for a more economical option. Each of the options has its pros and cons, so your decision will depend on several factors at the same time. As you weigh these, remember: Users often choose and prefer the option that they understand and that will provide them with a quick solution to their problems, even if more efficient options may exist.

NEED MORE REVIEW? MICROSOFT FABRIC DECISION GUIDE FOR COPY ACTIVITY, DATAFLOWS, AND SPARK NOTEBOOK

For more information about deciding between Copy Activity, dataflow, or Spark Notebook, please visit learn.microsoft.com/fabric/get-started/decision-guide-pipeline-dataflow-spark.

Copy data by using a data pipeline, dataflow, or notebook

Copying data within Microsoft Fabric, specifically within Lakehouse Explorer and Warehouse Explorer, has two possible origins: creating the given item that will be used for copying or opening the item to which the data will be copied. Both paths lead to the same goal, but the second can save the user a few extra clicks, through the fact that the user first opens the end item and creates copy items from it. This route automatically sets the destination to itself. For example, in DataFlow Gen2, all queries will automatically be set to store data in the Lakehouse Explorer from which it was created.

For the following examples, data obtained as samples within Lakehouse are used. This is how you can get them:

1. Create a new lakehouse or open a completely blank one.
2. Select **Start with sample data** (Figure 2-39), and wait for a moment.

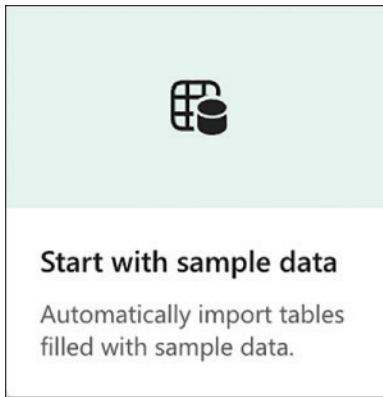


FIGURE 2-39 The Start with sample data option

EXAMPLE OF HOW TO COPY DATA BETWEEN LAKEHOUSES BY PIPELINE

You can use **Copy Activity** to copy data between lakehouses by following these steps:

1. Create and open a new lakehouse.
2. Select **New data pipeline** from **Get data** options (Figure 2-40), and choose a name for the pipeline.
3. Select **Lakehouse** and **Existing lakehouse**, and then choose the one that contains your data from the existing lakehouses (Figure 2-41). If you cannot see Lakehouse it is in a different workspace.
4. Select the **publicholidays** table (Figure 2-42).