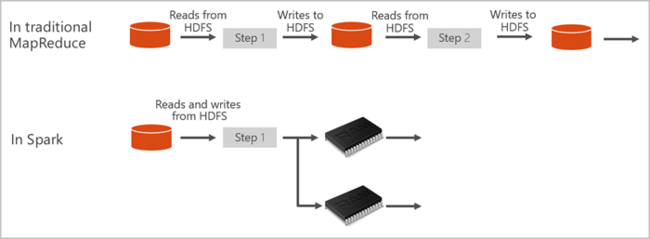
Apache Spark in Azure Synapse Analytics

Apache Spark is a parallel processing framework that supports in-memory processing to boost the performance of big-data analytic applications. Apache Spark in Azure Synapse Analytics is one of Microsoft's implementations of Apache Spark in the cloud. Azure Synapse makes it easy to create and configure a serverless Apache Spark pool in Azure. Spark pools in Azure Synapse are compatible with Azure Storage and Azure Data Lake Generation 2 Storage. So you can use Spark pools to process your data stored in Azure.

Apache Spark provides primitives for in-memory cluster computing. A Spark job can load and cache data into memory and query it repeatedly. In-memory computing is much faster than disk-based applications. Spark also integrates with multiple programming languages to let you manipulate distributed data sets like local collections. There's no need to structure everything as map and reduce operations.



**Spark pools in Azure Synapse include the following components that are available on the pools by default.**

* Spark Core. [ Includes Spark Core, Spark SQL, GraphX, and MLlib.]
* Anaconda
* Apache Livy
* Nteract notebook

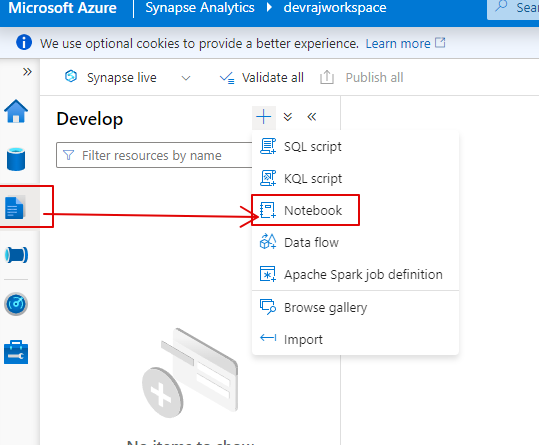
LAB :

# Create a serverless Apache Spark pool

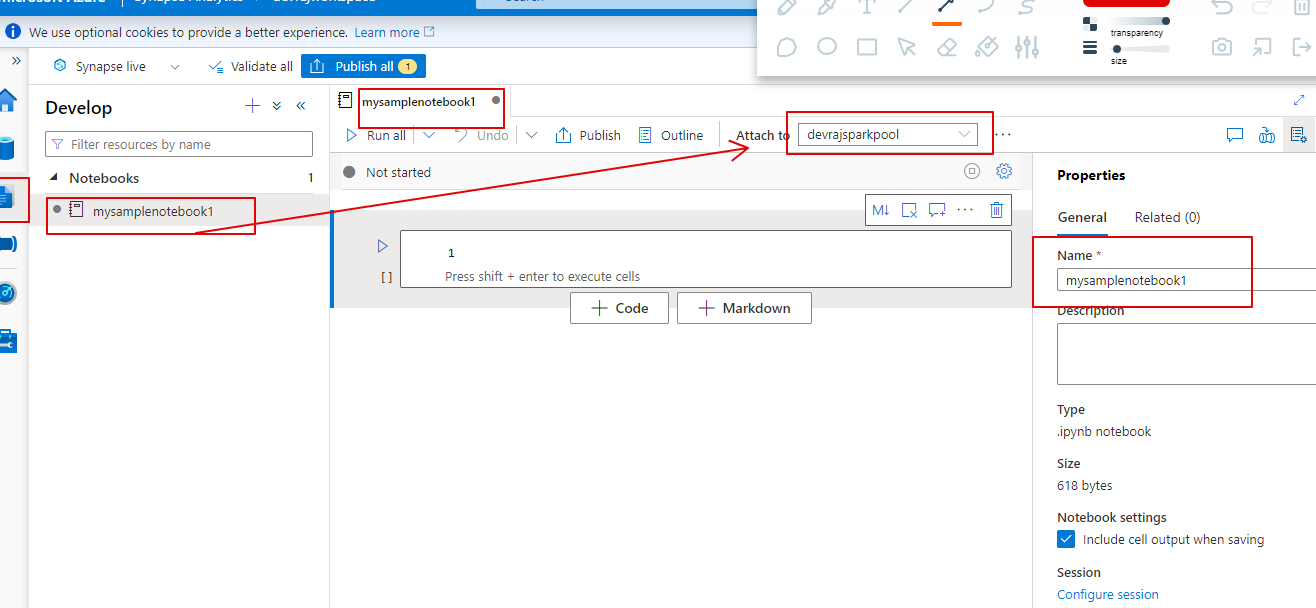
**Create a notebook**

A notebook is an interactive environment that supports various programming languages. The notebook allows you to interact with your data, combine code with markdown, text, and perform simple visualizations.

1. From the Azure portal view for the Azure Synapse workspace you want to use, select **Launch Synapse Studio**.
2. Once Synapse Studio has launched, select **Develop**. Then, select the "**+**" icon to add a new resource.
3. From there, select **Notebook**. A new notebook is created and opened with an automatically generated name.



1. In the **Properties** window, provide a name for the notebook.
2. On the toolbar, click **Publish**.



1. If there is only one Apache Spark pool in your workspace, then it's selected by default. Use the drop-down to select the correct Apache Spark pool if none is selected.
2. Click **Add code**. The default language is Pyspark. You are going to use a mix of Pyspark and Spark SQL, so the default choice is fine. Other supported languages are Scala and .NET for Spark.
3. Next you create a simple Spark DataFrame object to manipulate. In this case, you create it from code. There are three rows and three columns:

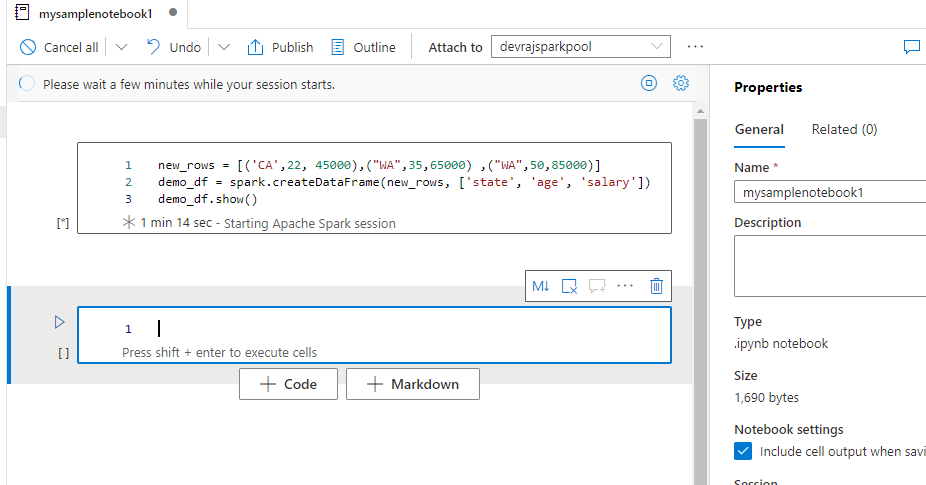
PythonCopy

new\_rows = [('CA',22, 45000),("WA",35,65000) ,("WA",50,85000)]

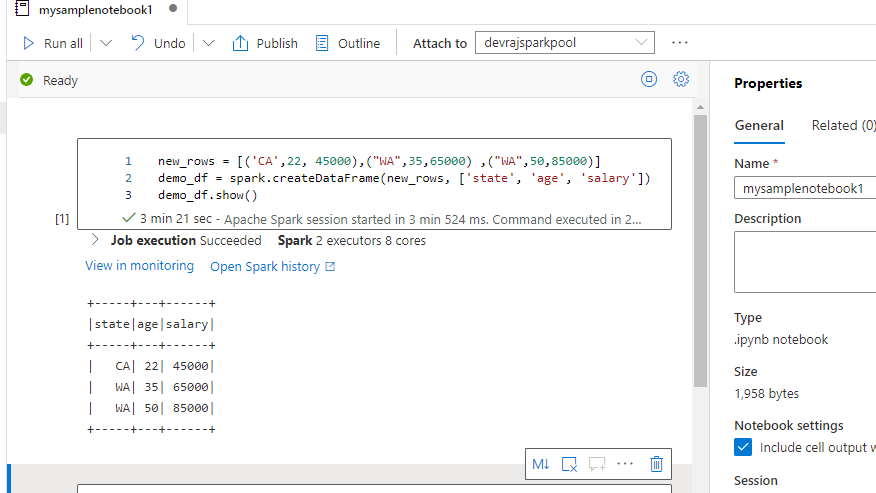
demo\_df = spark.createDataFrame(new\_rows, ['state', 'age', 'salary'])

demo\_df.show()

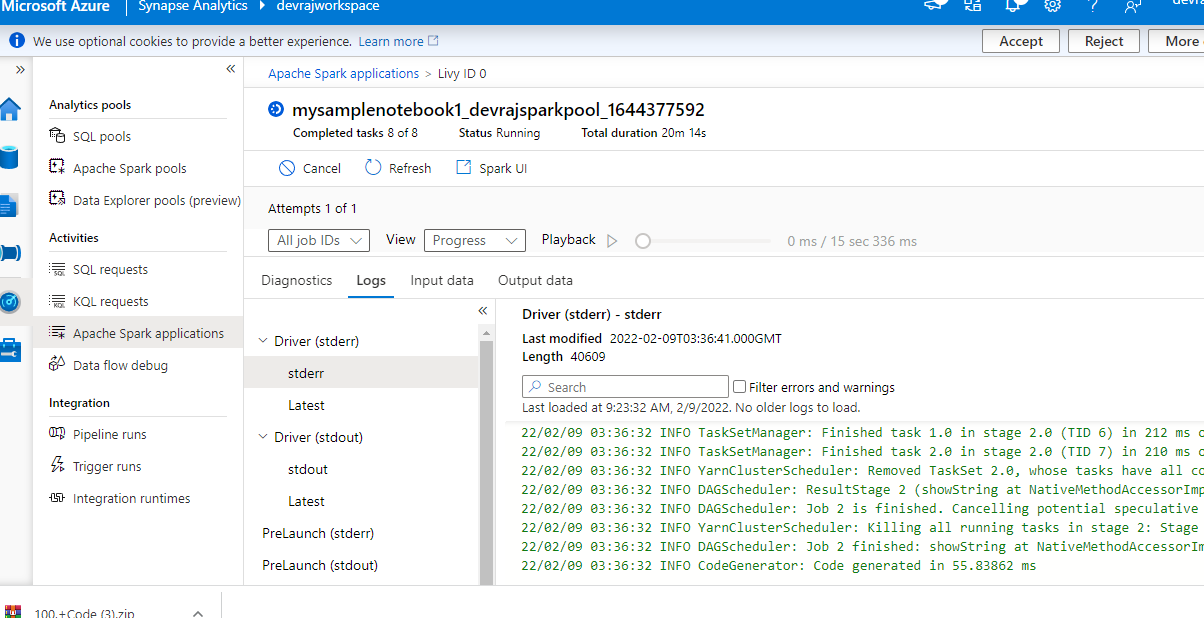
1. Now run the cell using one of the following methods:
   * Press **SHIFT + ENTER**.
   * Select the blue play icon to the left of the cell.
   * Select the **Run all** button on the toolbar.



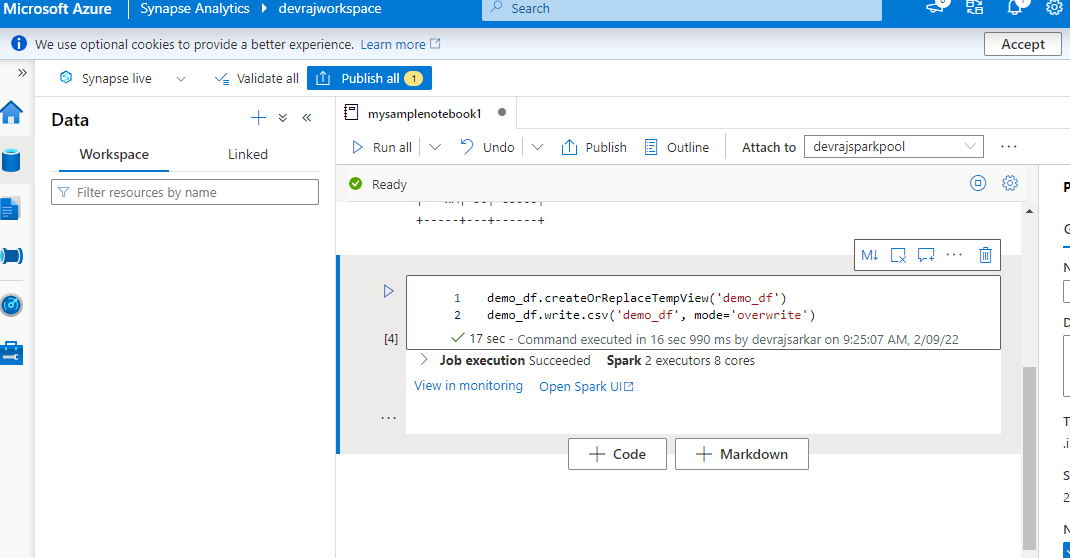
1. If the Apache Spark pool instance isn't already running, it is automatically started. You can see the Apache Spark pool instance status below the cell you are running and also on the status panel at the bottom of the notebook. Depending on the size of pool, starting should take 2-5 minutes. Once the code has finished running, information below the cell displays showing how long it took to run and its execution. In the output cell, you see the output.



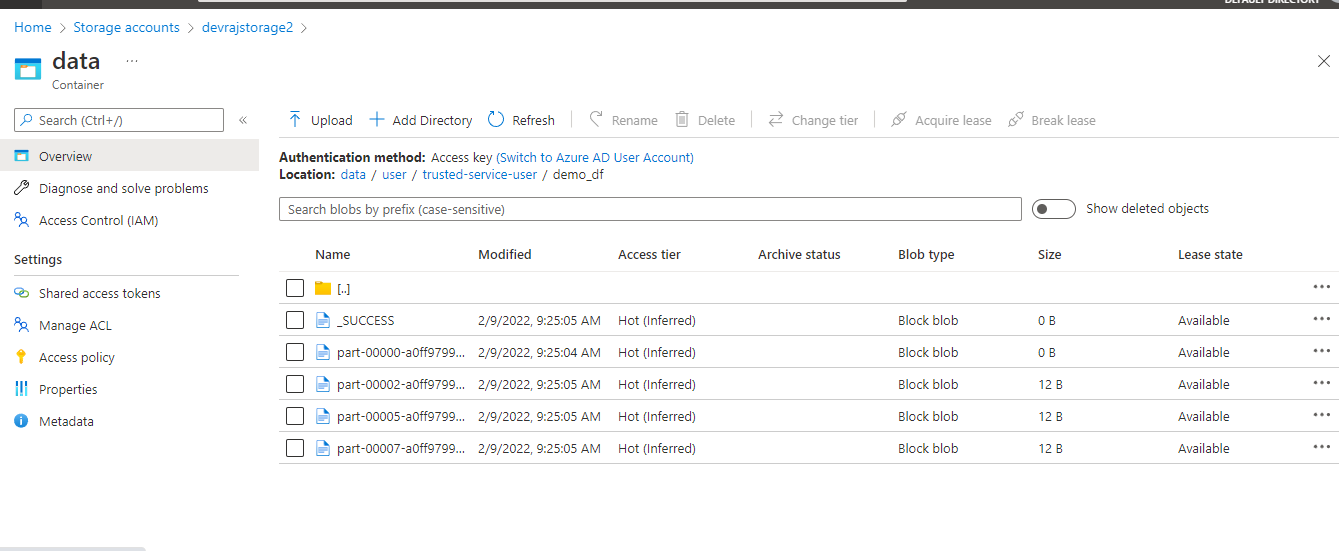
1. The data now exists in a DataFrame from there you can use the data in many different ways.

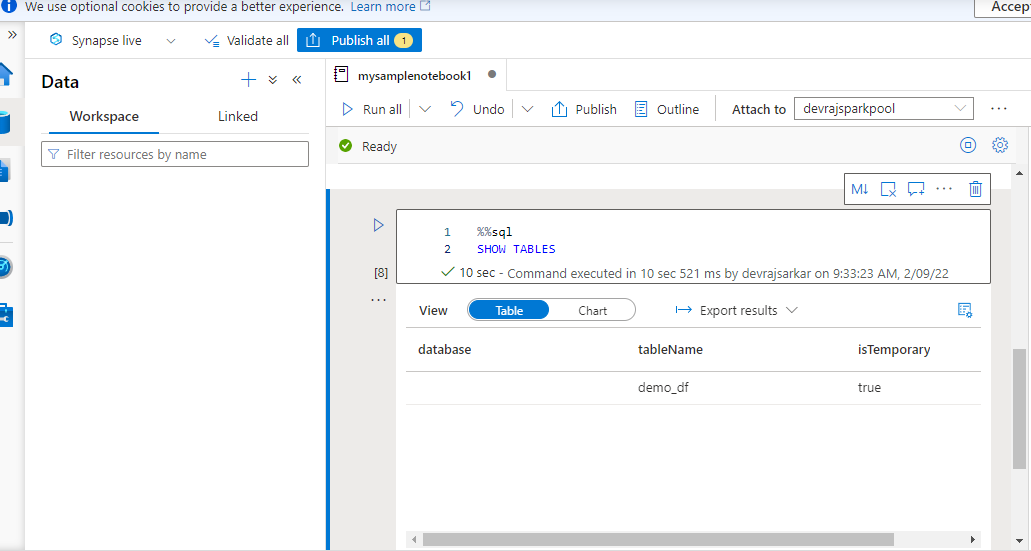


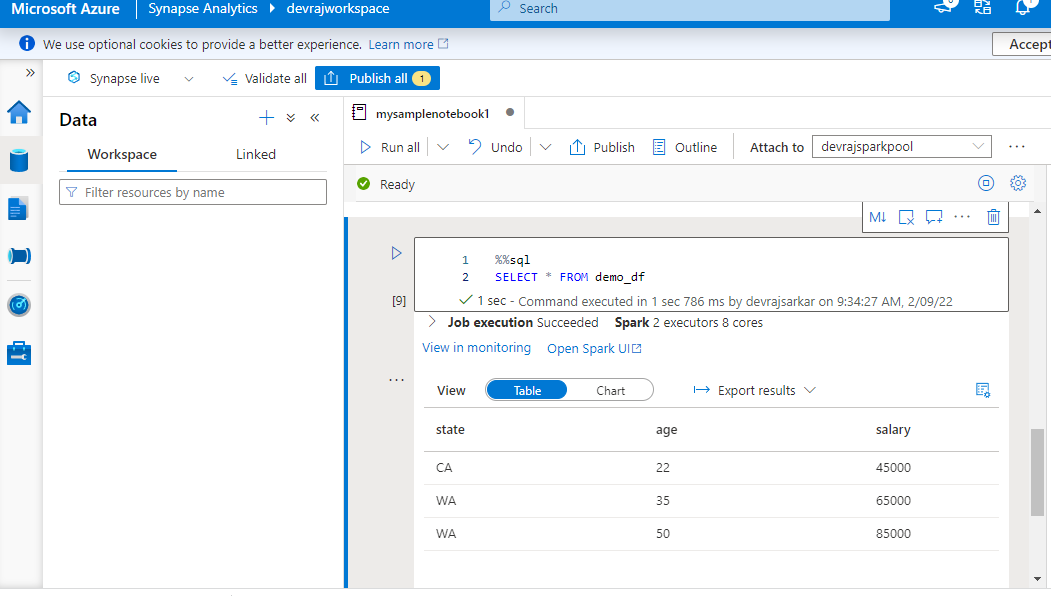
1. The data now exists in a DataFrame from there you can use the data in many different ways. You are going to need it in different formats for the rest of this quickstart.
2. Enter the code below in another cell and run it, this creates a Spark table, a CSV, all with copies of the data:



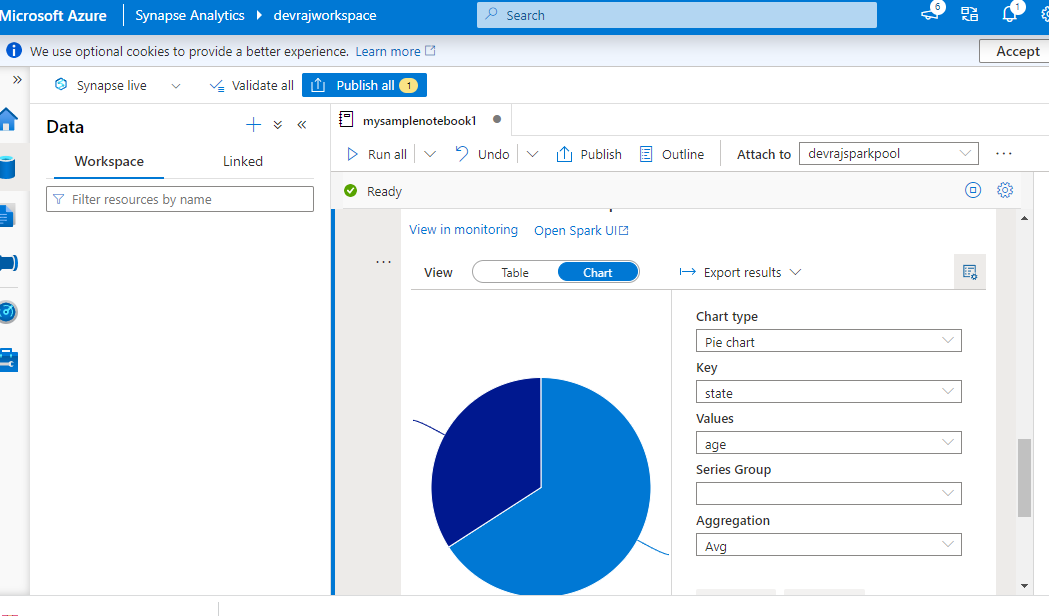
Check in storage account containers







1. In the **View** switcher, select **Chart**.
2. Select the **View options** icon from the far right-hand side.
3. In the **Chart type** field, select "pie chart".



It is possible to get the same experience of running SQL but without having to switch languages. You can do this by replacing the SQL cell above with this PySpark cell, the output experience is the same because the **display** command is used

