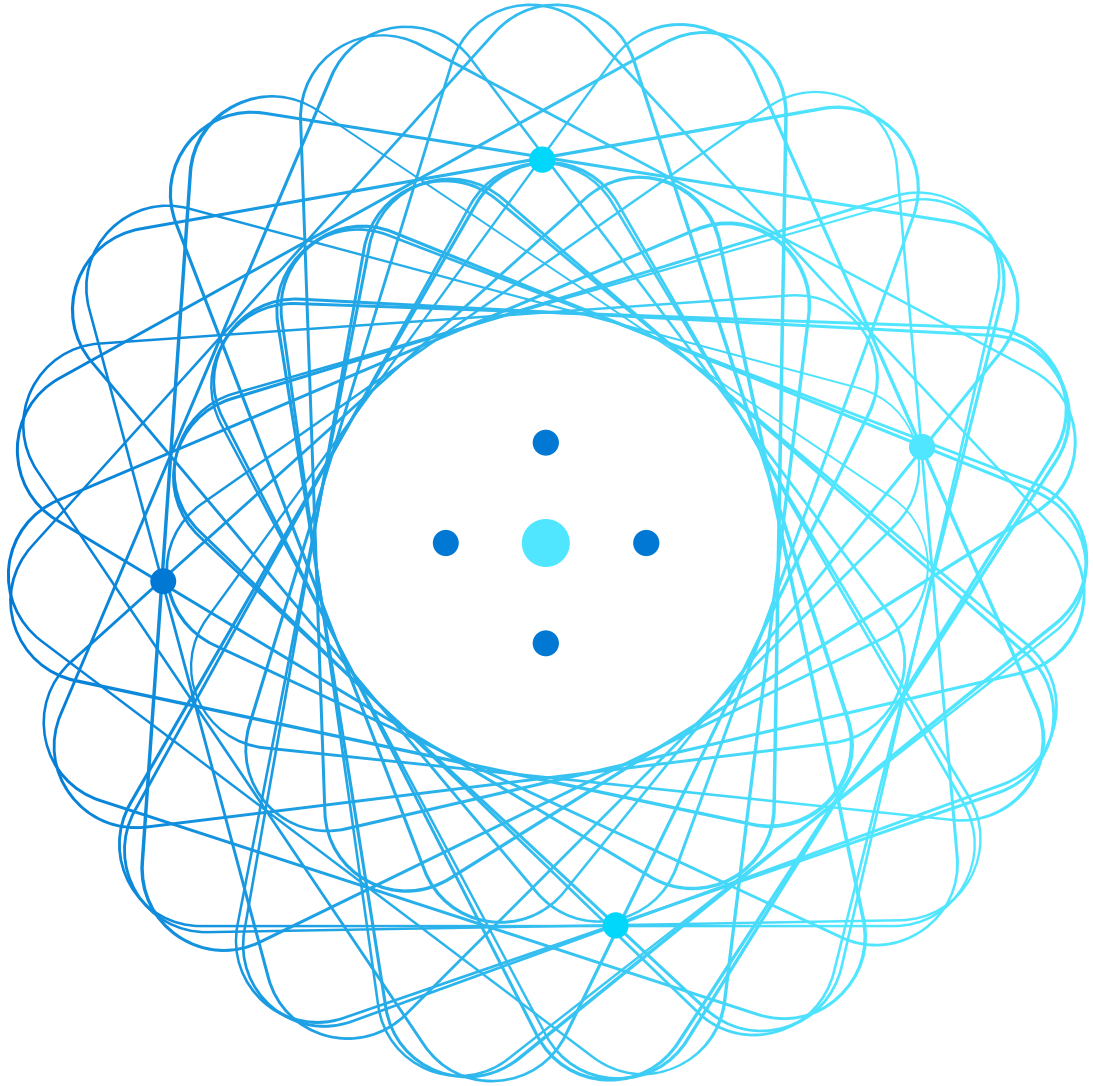
Microsoft Azure

Perform data engineering with Azure Synapse Apache Spark Pools



Agenda

Analyze data with Apache Spark in Azure Synapse Analytics

Transform data with Apache Spark in Azure Synapse Analytics

Use Delta Lake in Azure Synapse Analytics

Analyze data with Apache Spark in Azure Synapse Analytics

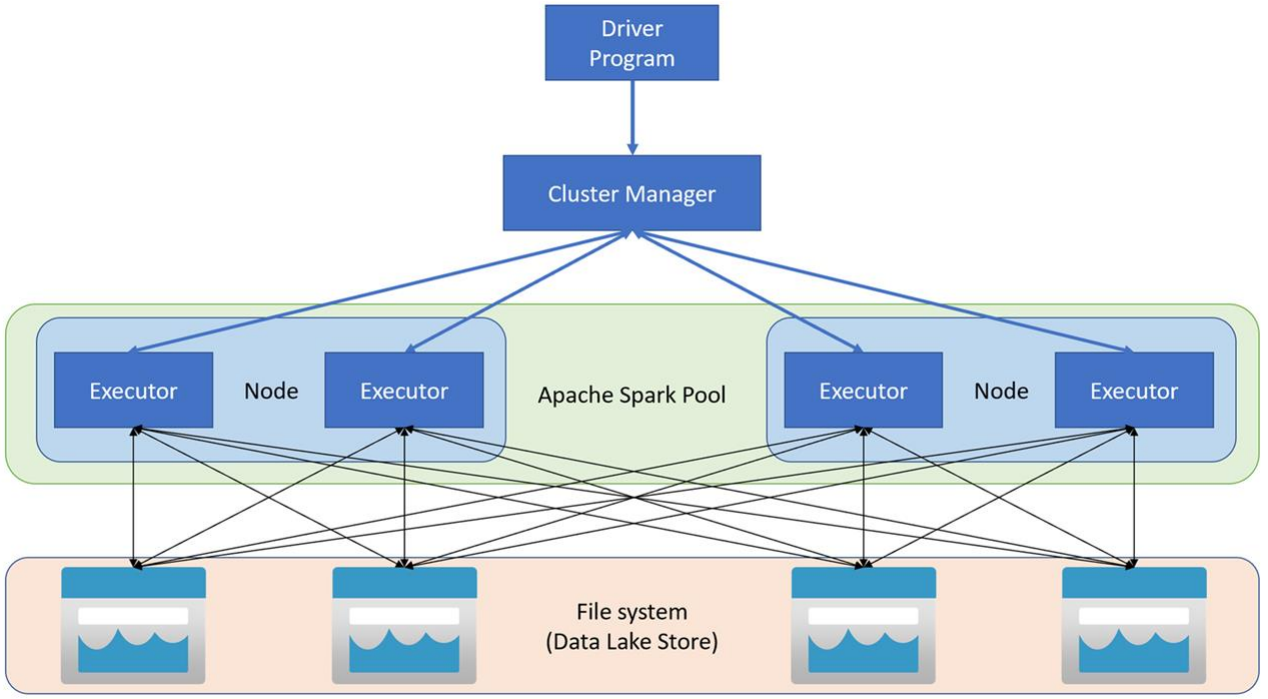
Get to know Apache Spark

Distributed data processing framework

- Code in multiple languages
- Driver program uses *SparkContext* to coordinate processing across multiple *executors* on worker nodes
- Executors run in parallel to process data in a distributed file system

Spark pools in Azure Synapse Analytics

- Named serverless cluster that auto-starts and stops - option to auto-scale
- Specific *Spark Runtime* version



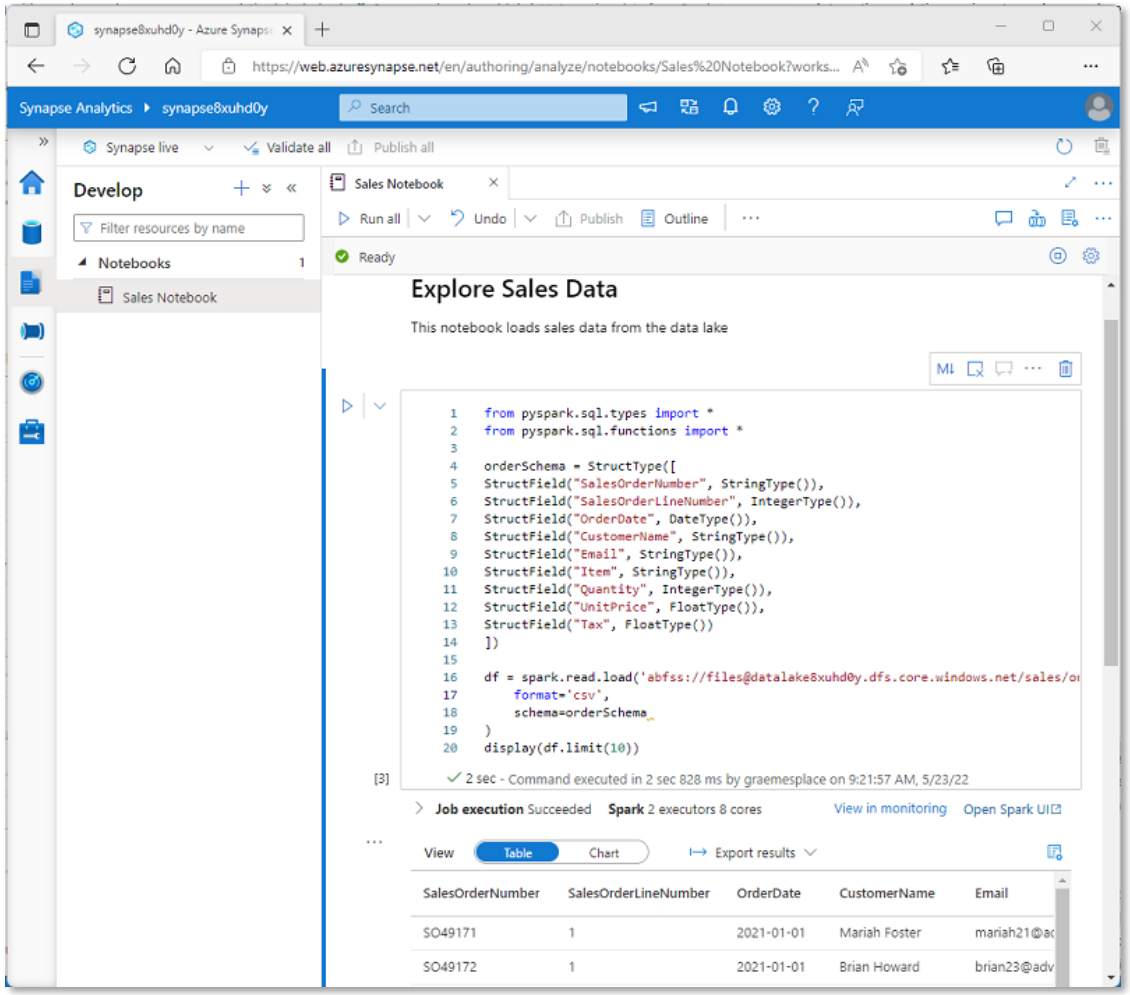
Use Spark in Azure Synapse Analytics

Integrated notebooks

- Syntax highlighting and error support
- Code auto-completion
- Interactive data visualizations
- Ability to export results

Work with data in multiple stores

- Primary workspace data lake
- Linked service storage
- Dedicated or serverless SQL pool
- Azure SQL or SQL Server database
- Azure Cosmos DB
- Azure Data Explorer Kusto database
- External Hive metastore



Analyze data with Spark

Explore data with dataframes

```
%%pyspark

# Load data
df=spark.read.load("/data/products.csv", format="csv", header=True)

# Manipulate dataframe
counts_df = df.select("ProductID", "Category").groupBy("Category").count()

# Display dataframe
display(counts_df)
```

Category	count
Headsets	3
Wheels	14
Mountain Bikes	32
...	...

Analyze data with Spark

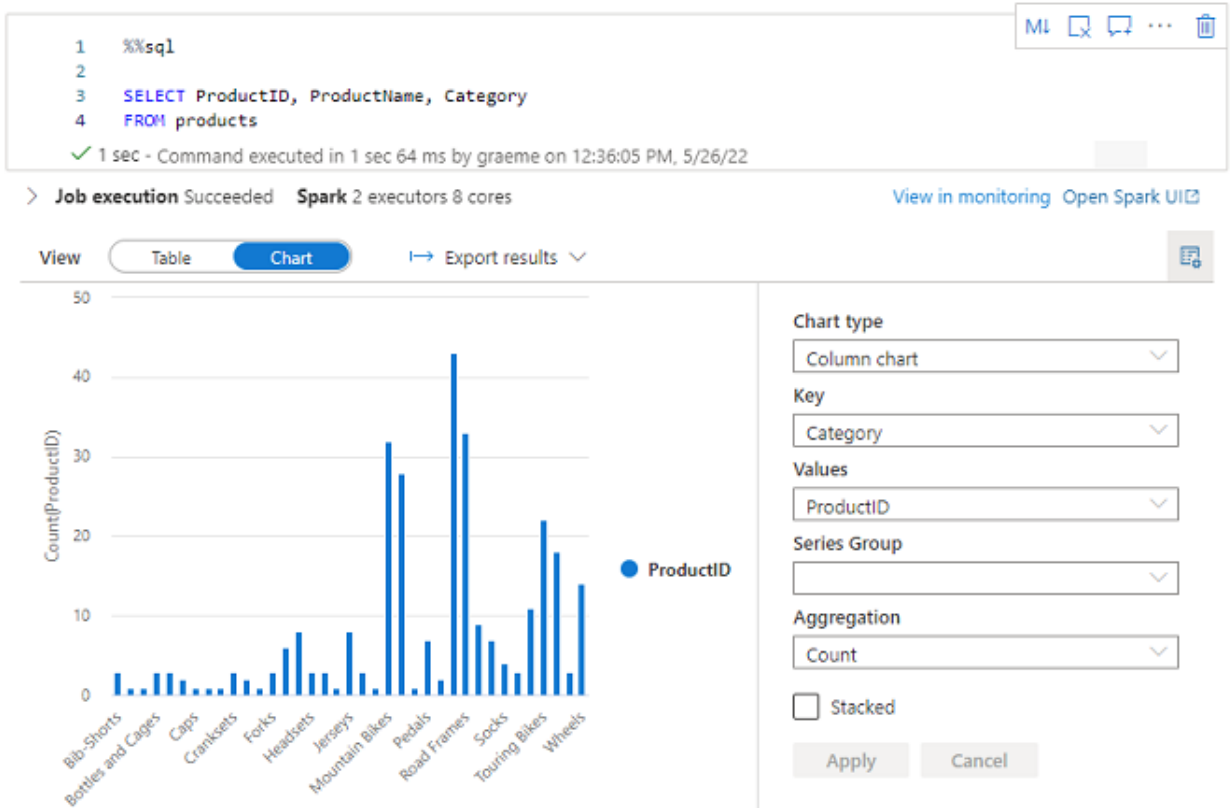
Using SQL expressions in Spark

```
%%pyspark

# Create a table in the metastore
df.createOrReplaceTempView("products")

# Use spark.sql method for inline SQL queries that return a dataframe
bikes_df = spark.sql("SELECT ProductID, ProductName, ListPrice \
                      FROM products \
                      WHERE Category IN ('Mountain Bikes', 'Road Bikes')")
display(bikes_df)
```

```
%sql

-- Use SQL to query tables in the metastore
SELECT Category, COUNT(ProductID) AS ProductCount
FROM products
GROUP BY Category
ORDER BY Category
```

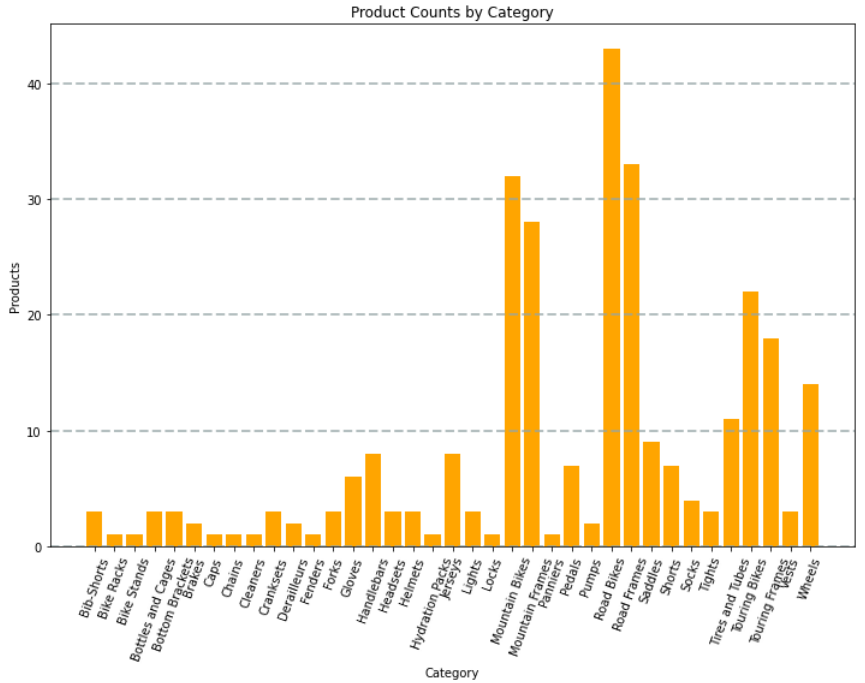
Visualize data with Spark

Built-in notebook charts



Graphics packages

```
from matplotlib import pyplot as plt

fig = plt.figure(figsize=(12,8))
plt.bar(x=data['Category'],
        height=data['ProductCount'],
        color='orange')
plt.show()
```



Demo: Analyze data with Spark

You can try this for yourself later by following the instructions at the link below:

https://aka.ms/mslearn-synapse-spark

Knowledge check

Which definition best describes Apache Spark?

- ❑ A highly scalable relational database management system
- ❑ A virtual server with a Python runtime
- ☑ A distributed platform for parallel data processing using multiple languages

You need to use Spark to analyze data in a parquet file. What should you do?

- ☑ Load the parquet file into a dataframe
- ❑ Import the data into a table in a serverless SQL pool
- ❑ Convert the data to CSV format

You want to write code in a notebook cell that uses a SQL query to retrieve data from a view in the Spark catalog. Which magic should you use?

- ❑ %%spark
- ❑ %%pyspark
- ☑ %%sql

Transform data with Apache Spark in Azure Synapse Analytics

Modify and save dataframes

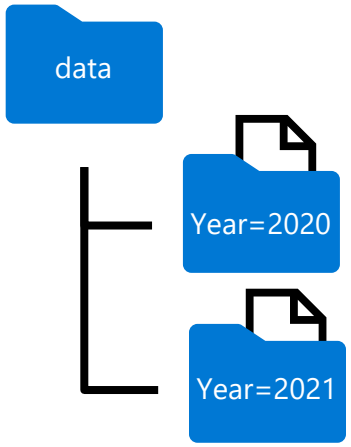
- Load source file into a dataframe
- Use dataframe methods and functions to transform the data:
  - Filter rows
  - Modify column values
  - Derive new columns
  - Drop columns
- Write the modified data
  - Specify required file format

```
from pyspark.sql.functions import year, col

# Load data
df = spark.read.load("/data/orders.csv",
                     format="csv", header=True)

# Add Year column, derived from OrderDate
df = df.withColumn("Year", year(col("OrderDate")))

# Save transformed data
df.write.mode("overwrite").parquet("/data/orders.parquet")
```

Partition data files

- Partition data by one or more columns
- Distributes data to improve performance and scalability

```
df.write.partitionBy("Year").mode("overwrite").parquet("/data")
```



Transform data with SQL

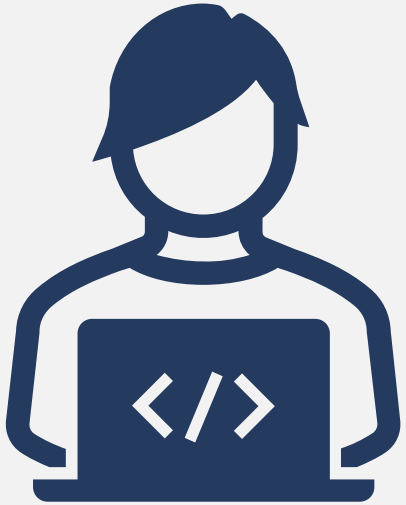
- Use the metastore to define tables and views
- Use SQL to query and transform the data
- Save transformed data as an external table
  - Dropping an external table does not delete the data files

```
# Create a view in the metastore
df.createOrReplaceTempView("sales_orders")

# Use SQL to transform data and return a dataframe
new_df = spark.sql("SELECT OrderNo, OrderDate, Year(OrderDate) As Year FROM sales_orders")

# Save the dataframe as an external table
new_df.write.partitionBy("Year").saveAsTable("transformed_orders", format="parquet",
                                             mode="overwrite", path="/transformed_orders")
```

Exercise: Transform data using Spark in Synapse Analytics

Use the hosted lab environment provided, or view the lab instructions at the link below:

https://aka.ms/mslearn-transform-spark



Knowledge check

Which method of the Dataframe object is used to save a dataframe as a file?

- ❑ toFile()
- ☑ write()
- ❑ save()

Which method is used to split the data across folders when saving a dataframe?

- ❑ splitBy()
- ❑ distributeBy()
- ☑ partitionBy()

What happens if you drop an external table that is based on existing files?

- ❑ An error – you must delete the files first
- ☑ The table is dropped from the metastore but the files remain unaffected
- ❑ The table is dropped from the metastore and the files are deleted

Use Delta Lake in Azure Synapse Analytics

What is Delta Lake?

Open-source storage layer that adds relational database semantics to Spark

- Relational tables that support querying and data modification
- Support for ACID transactions
- Data versioning and *time travel*
- Support for batch and streaming data
- Standard formats and interoperability

Create Delta Lake tables

Create a Delta Lake table from a dataframe

```
df = spark.read.load("/data/mydata.csv", format="csv", header=True)
delta_table_path = "/delta/mydata"
df.write.format("delta").save(delta_table_path)
```

Make conditional updates

```
from delta.tables import *
from pyspark.sql.functions import *

deltaTable = DeltaTable.forPath(spark, delta_table_path)
deltaTable.update(
    condition = "Category == 'Accessories'",
    set = { "Price": "Price * 0.9" })
```

Query a previous version (*time travel*)

```
df = spark.read.format("delta").option("versionAsOf", 0).load(delta_table_path)
```

Create catalog tables

- **Managed tables**
  - Defined without a specific location – files are created in metastore folder
  - Dropping the table deletes the files
- **External tables**
  - Defined with a specific file location
  - Dropping the table does not delete the files

```
df.write.format("delta").option("path","/mydata").saveAsTable("MyExternalTable")
```

```
spark.sql("CREATE TABLE MyExternalTable USING DELTA LOCATION '/mydata'")
```

```
%%sql
CREATE TABLE MyExternalTable
USING DELTA
LOCATION '/mydata'
```

Use Delta Lake with streaming data

Use Delta Lake table as a streaming source

```
from pyspark.sql.types import *
from pyspark.sql.functions import *

stream_df = spark.readStream.format("delta") \
    .option("ignoreChanges", "true") \
    .load("/delta/internetorders")

stream_df.show()
```

Use Delta Lake table as a streaming sink

```
from pyspark.sql.types import *
from pyspark.sql.functions import *

stream_df = spark.readStream.schema(jsonSchema).option("maxFilesPerTrigger", 1).json(inputPath)
table_path = '/delta/devicetable'
checkpoint_path = '/delta/checkpoint'
delta_stream = stream_df.writeStream.format("delta").option("checkpointLocation", checkpoint_path).start(table_path)
```

Use Delta Lake in a SQL pool

Query delta table files using OPENROWSET

```
SELECT *
FROM
    OPENROWSET(
        BULK 'https://mystore.dfs.core.windows.net/files/delta/mytable/',
        FORMAT = 'DELTA'
    ) AS deltadata
```

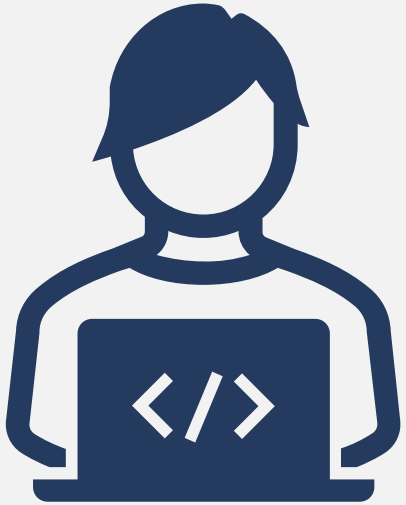
Query delta tables in Spark metastore databases

```
USE default;

SELECT * FROM MyDeltaTable;
```

Exercise: Use Delta Lake in Azure Synapse Analytics

Use the hosted lab environment provided, or view the lab instructions at the link below:

https://aka.ms/mslearn-delta-lake



Knowledge check

Which of the following descriptions best fits Delta Lake?

- ❑ A Spark API for exporting data from a relational database into CSV files
- ☑ A relational storage layer for Spark that supports tables based on Parquet files
- ❑ A synchronization solution that replicates data between SQL pools and Spark pools

You've loaded a Spark dataframe with data, that you now want to use in a Delta Lake table. What format should you use to write the dataframe to storage?

- ❑ CSV
- ❑ PARQUET
- ☑ DELTA

What feature of Delta Lake enables you to retrieve data from previous versions of a table?

- ❑ Spark Structured Streaming
- ☑ Time Travel
- ❑ Catalog Tables

Further reading

Perform data engineering with Azure Synapse Apache Spark Pools
https://aka.ms/mslearn-spark