

# Implementando uma solução de análise de dados com o Azure Synapse Analytics



# Agenda



- Introdução ao Azure Synapse Analytics
- Usar um pool de SQL sem servidor para consultar arquivos em um data lake
- Analisar dados com o Apache Spark no Azure Synapse Analytics
- Usar o Delta Lake no Azure Synapse Analytics
- Analisar s dados em um data warehouse relacional
- Carregar dados em um data warehouse relacional
- Criar um pipeline de dados no Azure Synapse Analytics

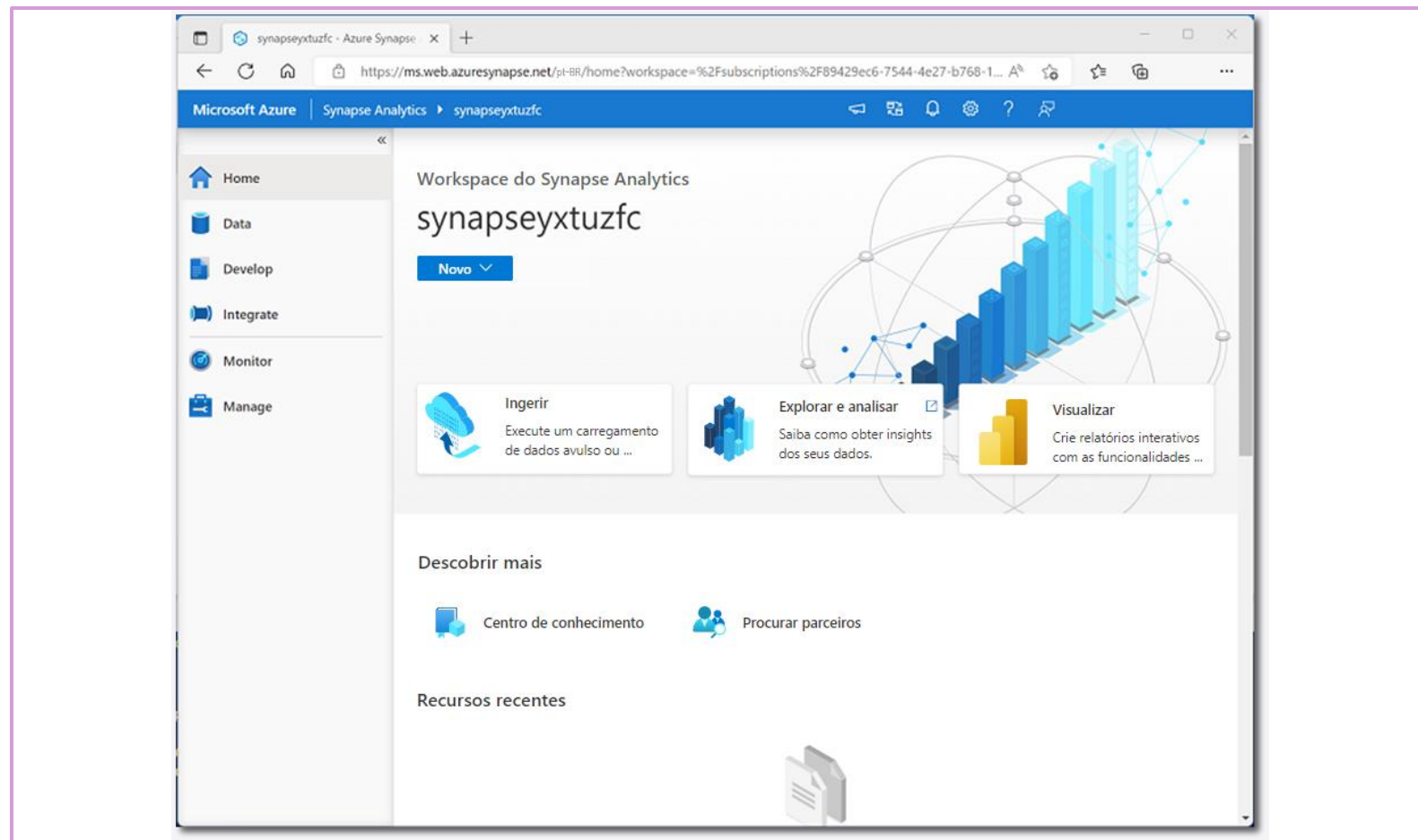
# Introdução ao Azure Synapse Analytics



# O que é o Azure Synapse Analytics?

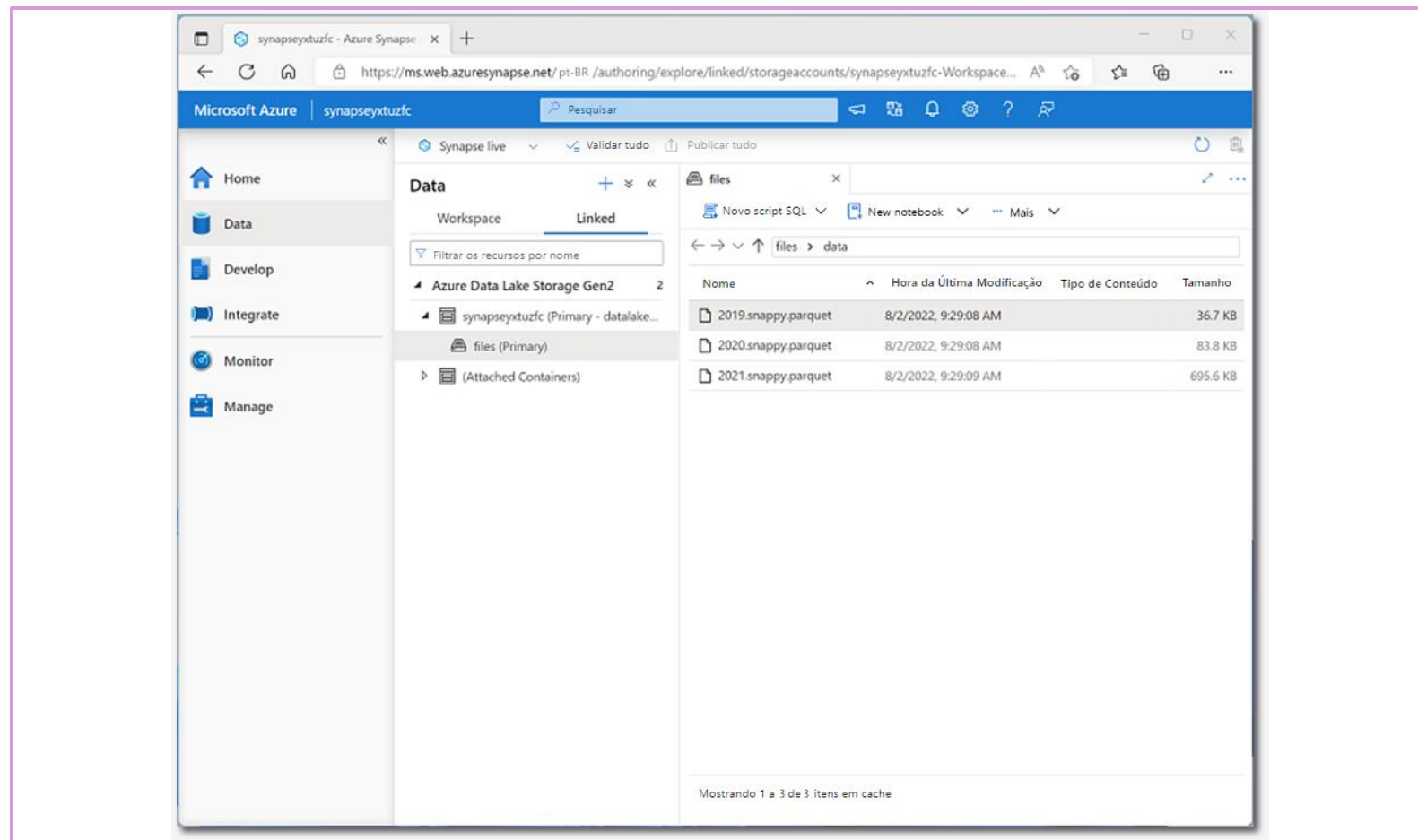
## Plataforma em nuvem para análise de dados

- Armazenamento de dados em grande escala
- Análise avançada
- Exploração e descoberta de dados
- Análise em tempo real
- Integração de dados
- Análise integrada



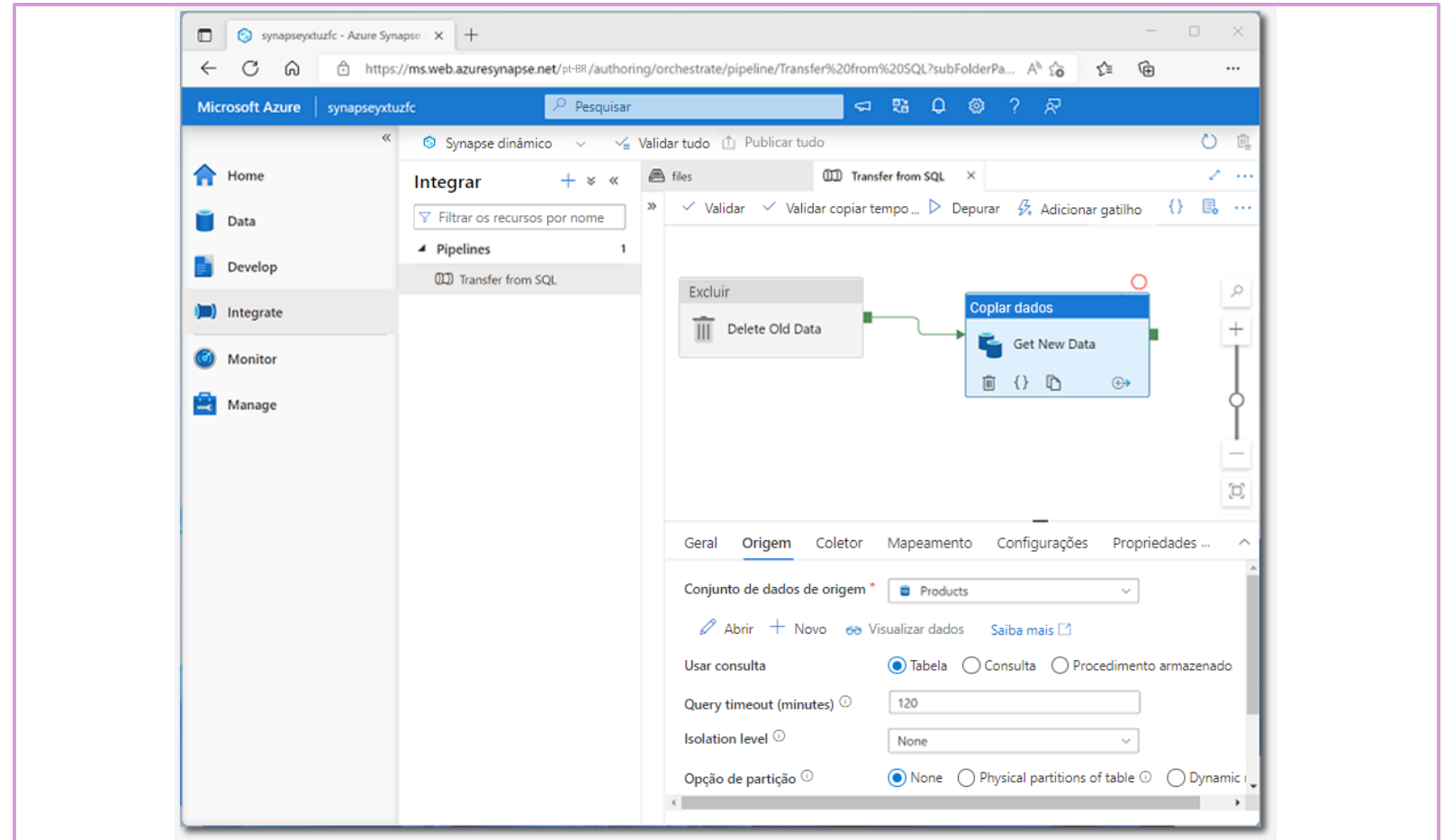
# Como trabalhar com arquivos em um data lake

- Conectar-se a um data lake storage usando *serviços vinculados*
- Cada espaço de trabalho do Azure Synapse Analytics tem um data lake padrão



# Ingestão e transformação de dados com pipelines

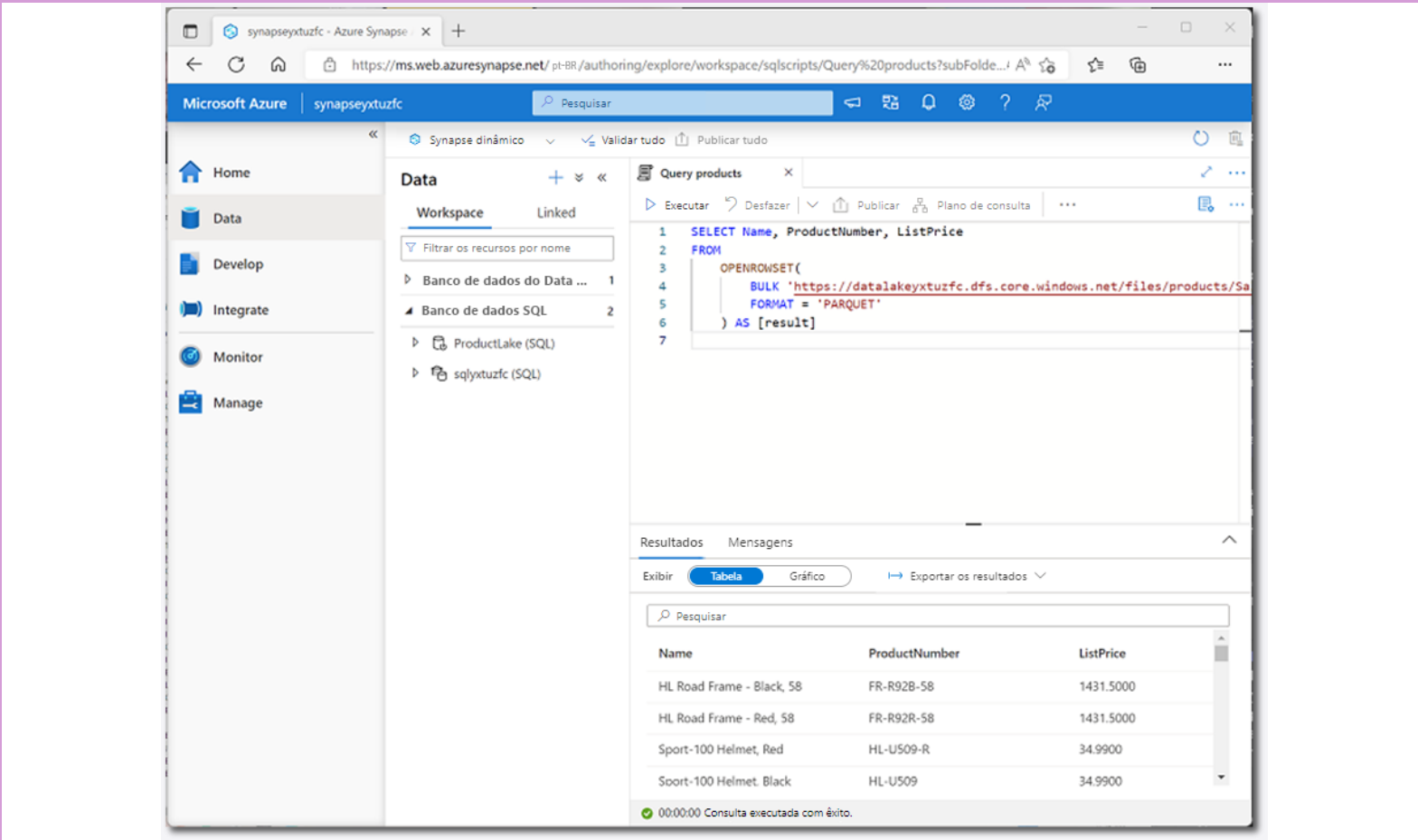
- Funcionalidade de pipeline nativo criada no Azure Data Factory
- Orquestrar atividades para ingerir, transformar e carregar dados
- Integrar com outros serviços de dados



# Consulta e manipulação de dados com o SQL

## Pools baseados no SQL Server para processamento de dados relacionais escalonáveis:

- Pool de SQL interno *sem servidor* para exploração de dados e análise de arquivos no data lake
- Pools de SQL *dedicados* personalizados para hospedar data warehouses relacionais de grande escala



The screenshot displays the Microsoft Azure Synapse Studio interface. The left sidebar shows navigation options: Home, Data, Develop, Integrate, Monitor, and Manage. The main workspace is divided into two panes. The left pane, titled 'Data', shows a list of resources under the 'Workspace' tab, including 'Banco de dados do Data ...', 'Banco de dados SQL', 'ProductLake (SQL)', and 'sqlxtzfc (SQL)'. The right pane, titled 'Query products', shows a SQL query being executed. The query is as follows:

```
1 SELECT Name, ProductNumber, ListPrice
2 FROM
3 OPENROWSET(
4     BULK 'https://datalakeyxtzfc.dfs.core.windows.net/files/products/Sa
5     FORMAT = 'PARQUET'
6 ) AS [result]
```

The bottom section of the interface shows the 'Resultados' (Results) pane, which displays a table with the following data:

Name	ProductNumber	ListPrice
HL Road Frame - Black, 58	FR-R92B-58	1431.5000
HL Road Frame - Red, 58	FR-R92R-58	1431.5000
Sport-100 Helmet, Red	HL-U509-R	34.9900
Sport-100 Helmet, Black	HL-U509	34.9900

A status message at the bottom indicates: '00:00:00 Consulta executada com êxito.' (Query executed successfully).

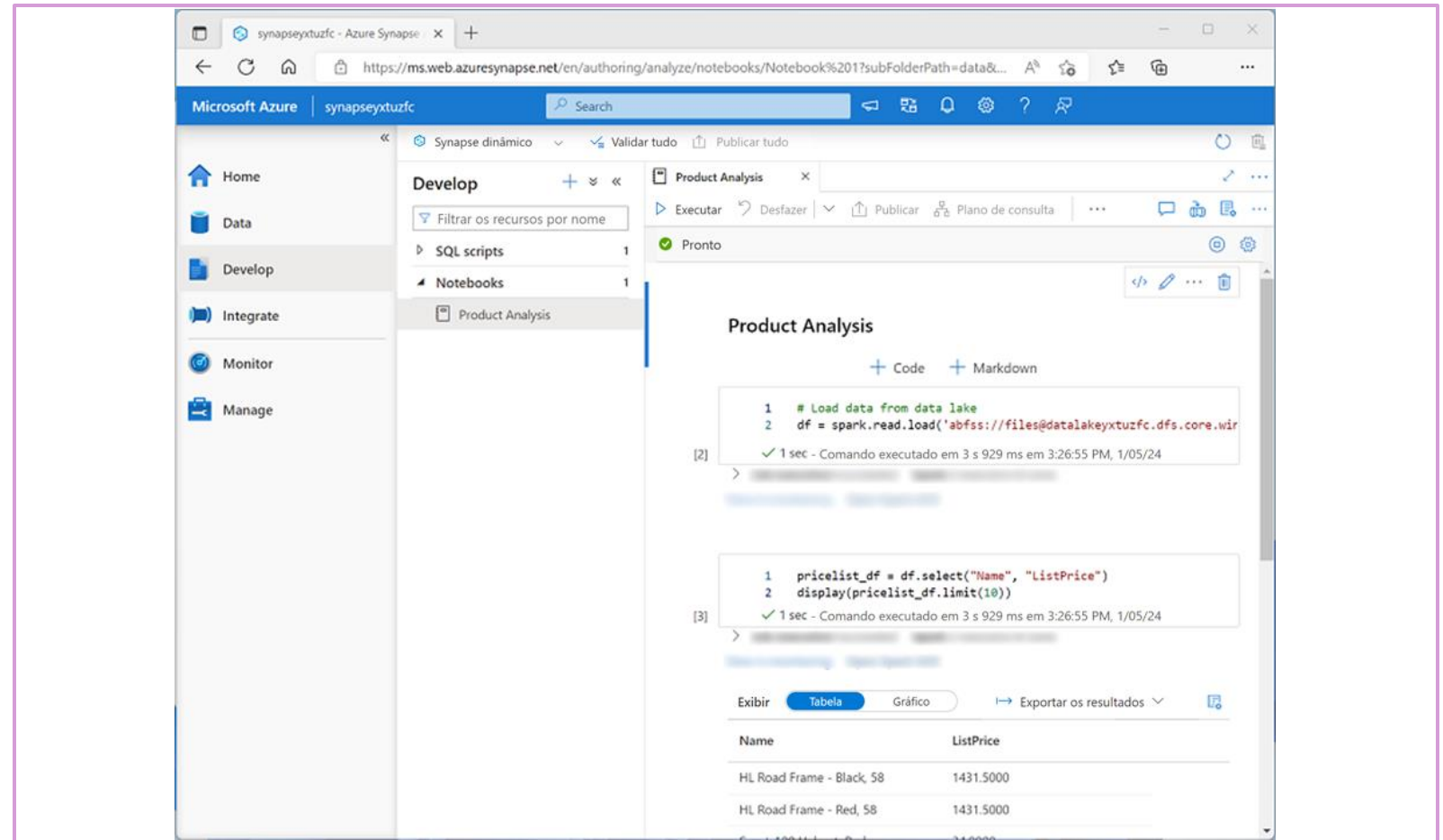


# Processar e analisar dados com o Apache Spark

## Tecnologia Spark de código aberto

- Processamento distribuído e altamente escalonável
- Bibliotecas comuns e várias linguagens de programação

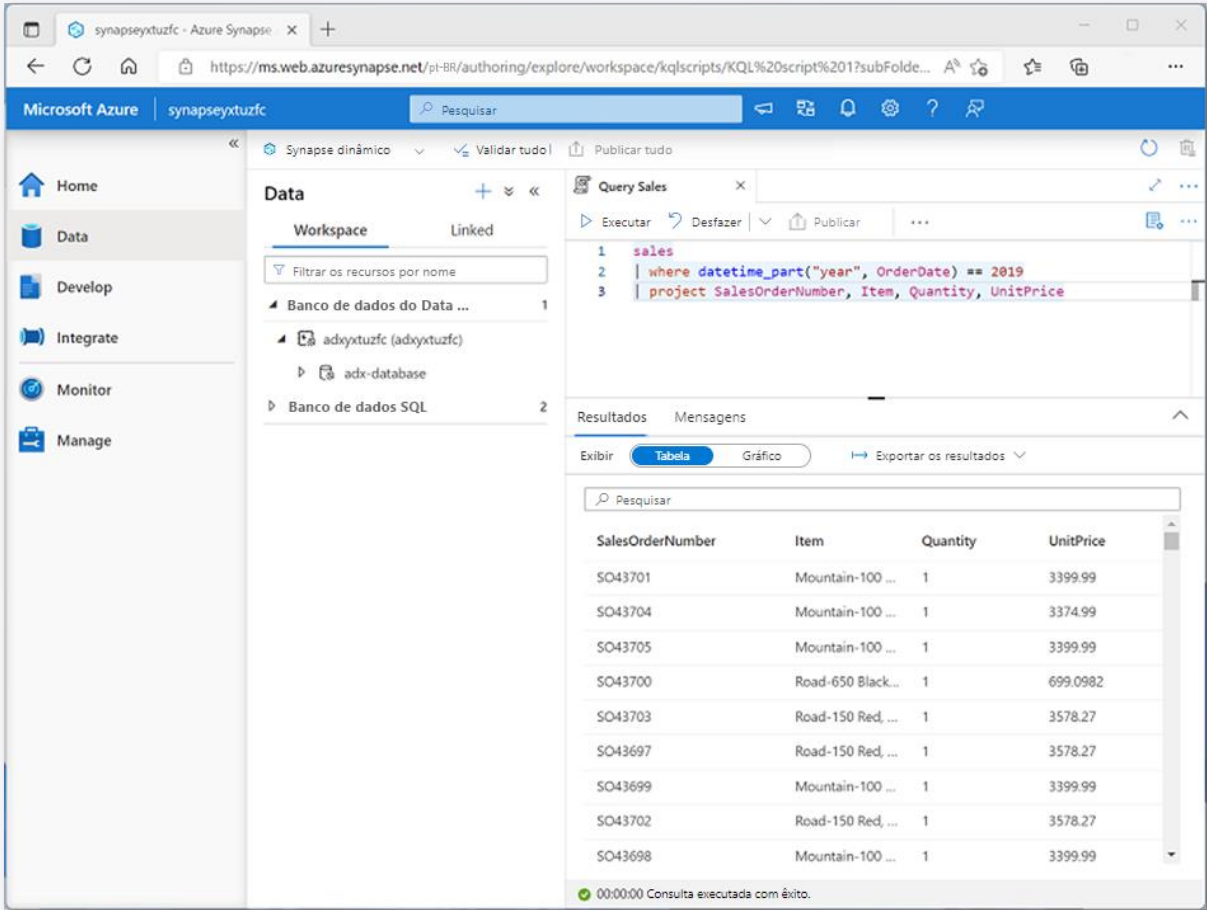
## Experiência de notebook integrada





# Explore data with Data Explorer

- High-performance real-time data analytics
- Powerful, intuitive Kusto query language



The screenshot displays the Microsoft Azure Data Explorer web interface. The left sidebar contains navigation links: Home, Data, Develop, Integrate, Monitor, and Manage. The main area is divided into three sections. The 'Data' section on the left shows a tree view of data sources under 'Workspace' and 'Linked', including 'Banco de dados do Data ...', 'adxxtuzfc (adxxtuzfc)', 'adx-database', and 'Banco de dados SQL'. The 'Query Sales' section in the center shows a Kusto query: 

```
1 sales
2 | where datetime_part("year", OrderDate) == 2019
3 | project SalesOrderNumber, Item, Quantity, UnitPrice
```

. The 'Resultados' section on the right shows the query results in a table format. The table has four columns: SalesOrderNumber, Item, Quantity, and UnitPrice. The results show 10 rows of data for the year 2019. A status bar at the bottom indicates '00:00:00 Consulta executada com êxito.'

SalesOrderNumber	Item	Quantity	UnitPrice
SO43701	Mountain-100 ...	1	3399.99
SO43704	Mountain-100 ...	1	3374.99
SO43705	Mountain-100 ...	1	3399.99
SO43700	Road-650 Black...	1	699.0982
SO43703	Road-150 Red, ...	1	3578.27
SO43697	Road-150 Red, ...	1	3578.27
SO43699	Mountain-100 ...	1	3399.99
SO43702	Road-150 Red, ...	1	3578.27
SO43698	Mountain-100 ...	1	3399.99

# Acesso ao Ambiente de Laboratório

<https://msle.learnondemand.net>

Código de Treinamento: 57BEA81D602E46D2



# Exercise: Explore Azure Synapse Analytics

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

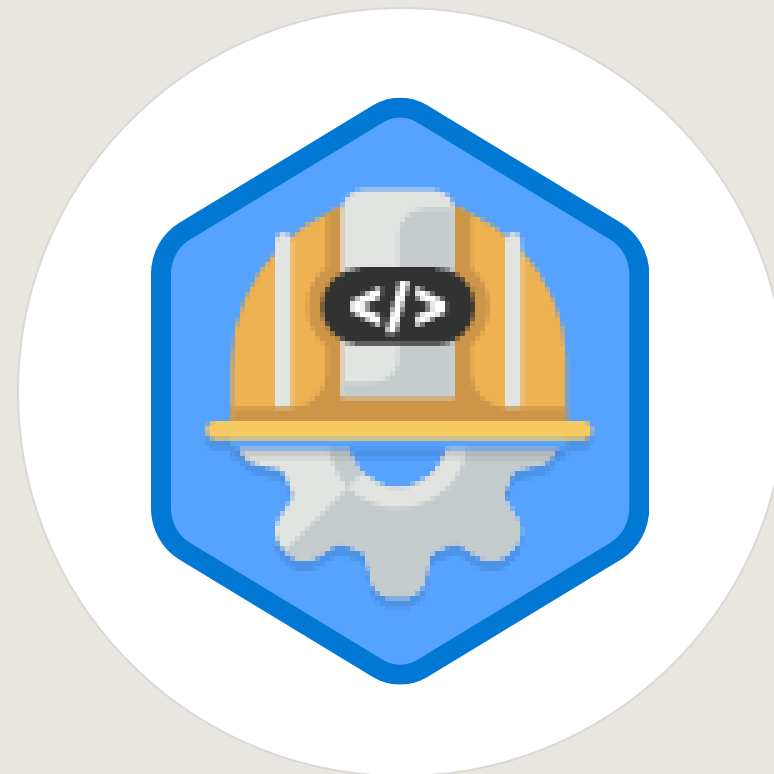
<https://aka.ms/mslearn-explore-synapse>



# Leitura adicional

Introdução à engenharia de dados no Azure

<https://aka.ms/mslearn-data-engineer-ptb>



# Usar um pool de SQL sem servidor para consultar arquivos em um data lake

<https://learn.microsoft.com/pt-br/training/modules/query-data-lake-using-azure-synapse-serverless-sql-pools/>



# Pools de SQL no Azure Synapse Analytics



## Azure Synapse Analytics



### Pool de SQL sem servidor

- Processamento de consultas SQL sob demanda
- Dados armazenados como arquivos em um data lake
- Casos de uso típicos:
  - Exploração de dados
  - Transformação de dados
  - Data warehouse lógico



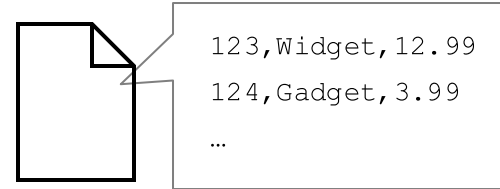
### Pools de SQL dedicados

- Banco de dados relacional em escala de nuvem
- Dados armazenados em tabelas relacionais
- Casos de uso típicos:
  - Armazém de dados relacional
  - Business Intelligence empresarial

# Consultar arquivos de texto delimitado usando um pool de SQL sem servidor

## Usar a função OPENROWSET

- Usar o parâmetro BULK especifica o(s) caminho(s) do arquivo
  - Incluir curingas conforme necessário
- Usar o parâmetro FORMAT para especificar o "csv"
- Usar parâmetros adicionais para:
  - Linha de cabeçalho
  - Caracteres delimitadores
  - Versão do analisador
  - Others
- Usar a cláusula WITH para especificar nomes e tipos de coluna



```
SELECT *  
FROM OPENROWSET(  
    BULK 'https://.../data/files/*.csv',  
    FORMAT = 'csv',  
    PARSER_VERSION = '2.0')  
WITH (  
    product_id INT,  
    product_name VARCHAR(20),  
    list_price DECIMAL(5,2)  
) AS rows
```

product_id	product_name	list_price
123	Widget	12.99
124	Gadget	3,99
...	...	...



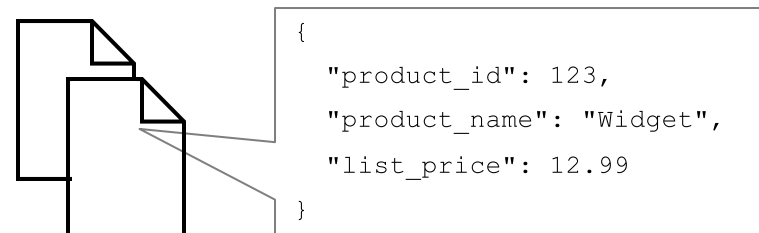
# Consultar arquivos JSON usando um pool de SQL sem servidor

## Usar a função OPENROWSET

- Usar o parâmetro BULK especifica o(s) caminho(s) do arquivo
  - Incluir curingas conforme necessário
- Usar o parâmetro FORMAT para especificar o "csv"
- Definir terminadores como '0x0b'
- Usar a cláusula WITH para especificar uma única coluna NVARCHAR

## Usar a função JSON\_VALUE para especificar propriedades JSON

- Especificar o caminho do atributo baseado em JSON na coluna NVARCHAR



```
SELECT JSON_VALUE(doc, '$.product_name') AS product,
       JSON_VALUE(doc, '$.list_price') AS price
FROM
  OPENROWSET(
    BULK 'https://.../data/files/*.json',
    FORMAT = 'csv',
    FIELDTERMINATOR = '0x0b',
    FIELDQUOTE = '0x0b',
    ROWTERMINATOR = '0x0b'
  ) WITH (doc NVARCHAR(MAX)) as rows
```

product	price
Widget	12.99
Gadget	3,99
...	...

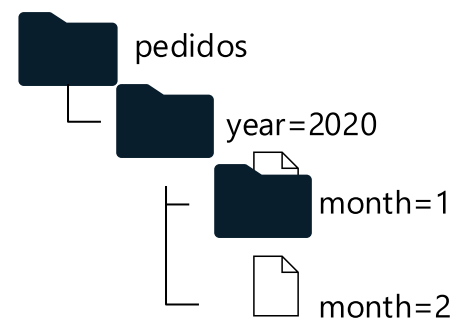
# Consultar arquivos Parquet usando um pool de SQL sem servidor

## Usar a função OPENROWSET

- Usar o parâmetro BULK especifica o(s) caminho(s) do arquivo
  - Incluir curingas conforme necessário
- Usar o parâmetro FORMAT para especificar o "parquet"

## Usar a propriedade *filepath* para filtrar por partições

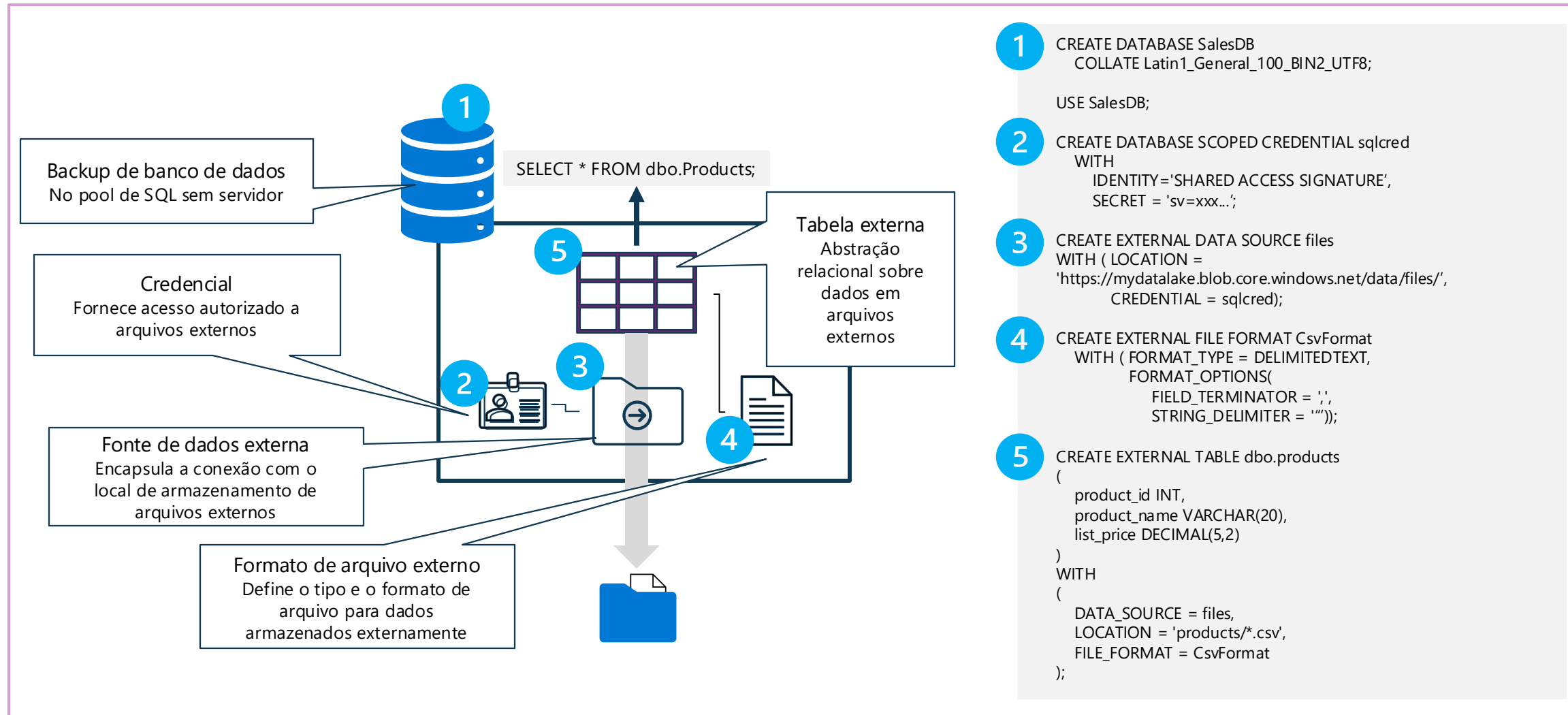
- Os parâmetros refletem a posição ordinal dos curingas
- Não específico para parquet, mas comumente usado para distribuir dados em formato parquet



```
SELECT *  
FROM OPENROWSET(  
    BULK 'https://.../data/orders/year=*/month=*/*.*',  
    FORMAT = 'parquet') AS orders  
WHERE orders.filepath(1) = '2020'  
    AND orders.filepath(2) IN ('1','2');
```

order_no	order_date	order_total
1001	2020-01-07	99.78
1002	12-01-2020	11,99
...	...	...

# Criar objetos de banco de dados externos



# Demo: Query files using a serverless SQL pool

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

<https://aka.ms/mslearn-synapse-sql>



# Analisar dados com o Apache Spark no Azure Synapse Analytics



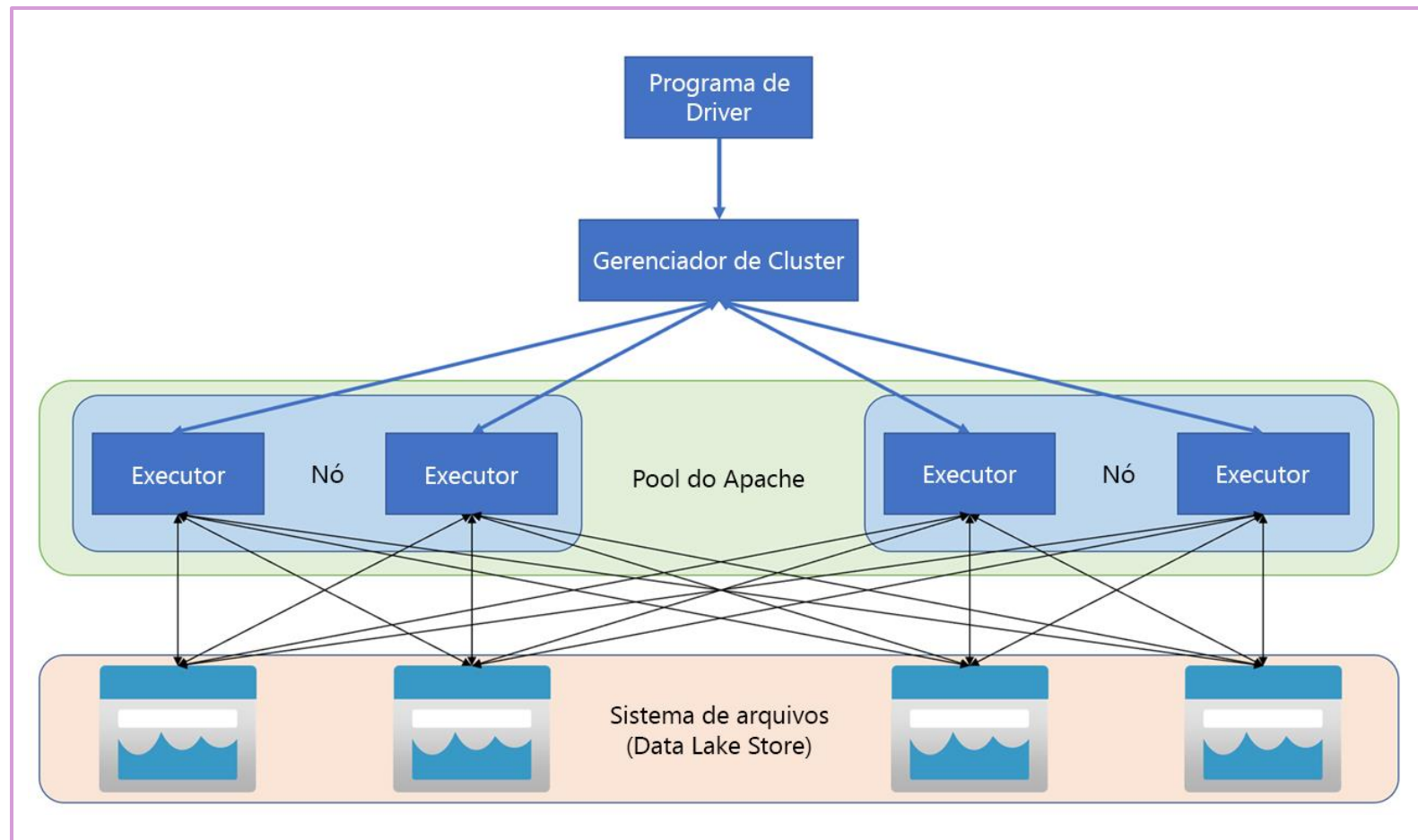
# Conhecer o Apache Spark

## Estrutura de processamento de dados distribuído

- Código em vários idiomas
- O programa de driver usa o SparkContext para coordenar o processamento em vários executores em nós de trabalho
- Os executores funcionam em paralelo para processar dados em um sistema de arquivos distribuído

## Pools do Spark no Azure Synapse Analytics

- Cluster sem servidor nomeado que inicia e para automaticamente – opção para dimensionamento automático
- Versão específica do Spark Runtime



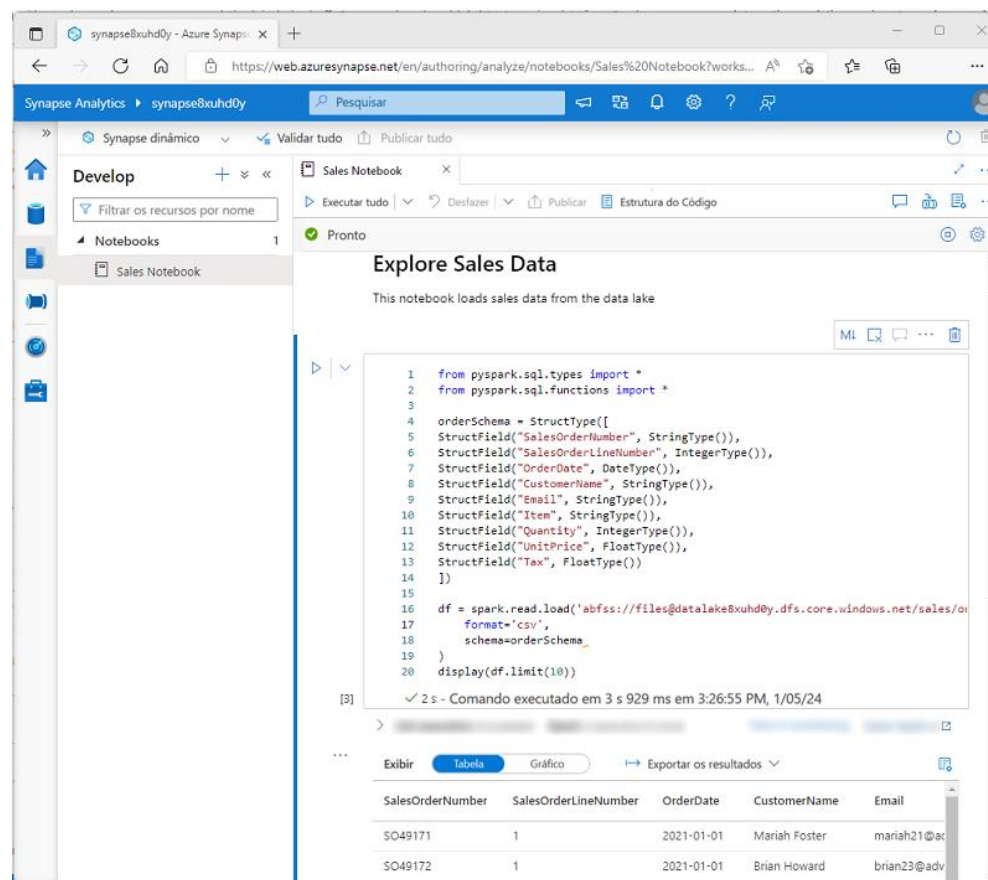
# Usar o Spark no Azure Synapse Analytics

## Notebooks integrados

- Realce de sintaxe e suporte a erros
- Preenchimento automático de código
- Visualização de dados interativos
- Capacidade de exportar resultados

## Trabalhar com dados em vários repositórios

- Data lake do espaço de trabalho primário
- Armazenamento de serviço vinculado
- Pool de SQL dedicado ou sem servidor
- Banco de dados SQL Server ou SQL do Azure
- Azure Cosmos DB
- Banco de dados Kusto do Azure Data Explorer
- Metastore do Hive externo





# Analisar dados com o Spark

## Explorar dados com 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)
```

Categoria	count
Fones de ouvido	3
Rodas	14
Mountain bikes	32
...	...

# Analisar dados com o Spark

## Usando expressões SQL no 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 \n                      FROM products \n                      WHERE Category IN ('Mountain Bikes', 'Road Bikes')")
```

```
display(bikes_df)
```

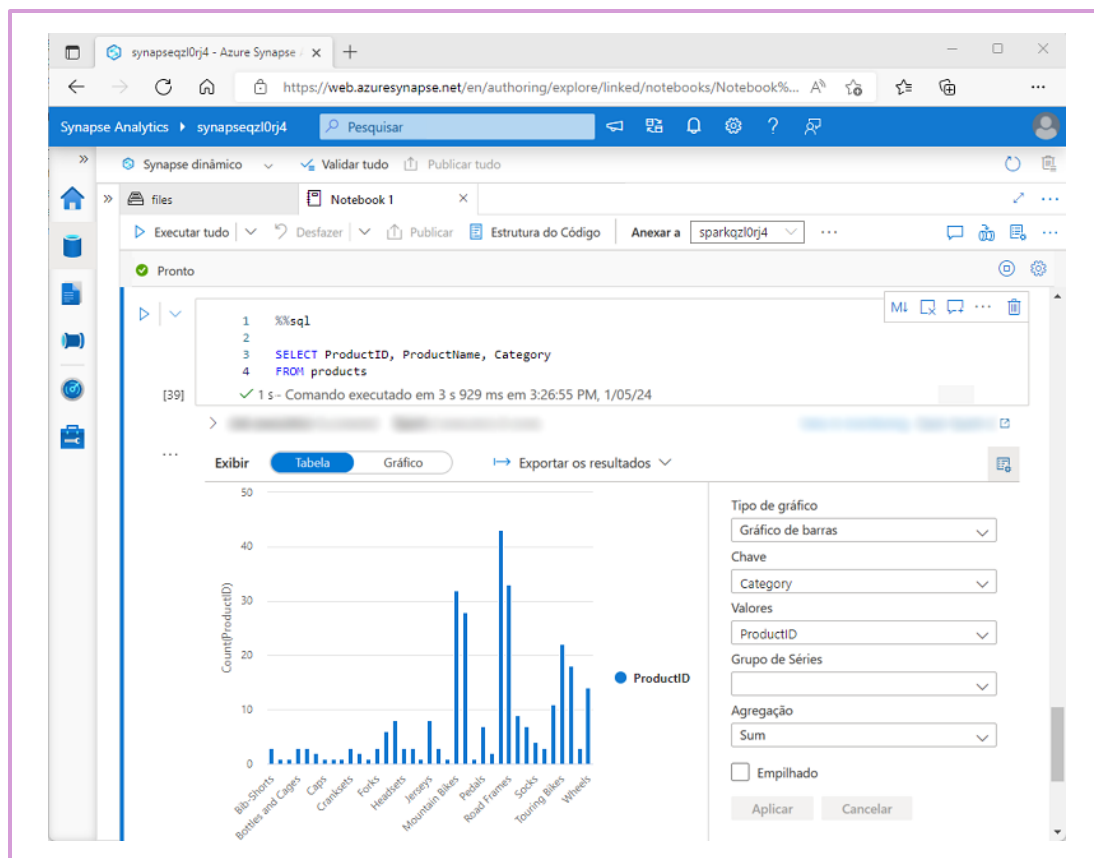
```
%sql
```

```
-- Use SQL to query tables in the metastore
```

```
SELECT Category, COUNT(ProductID) AS ProductCount\nFROM products\nGROUP BY Category\nORDER BY Category
```

# Visualizar os dados com o Spark

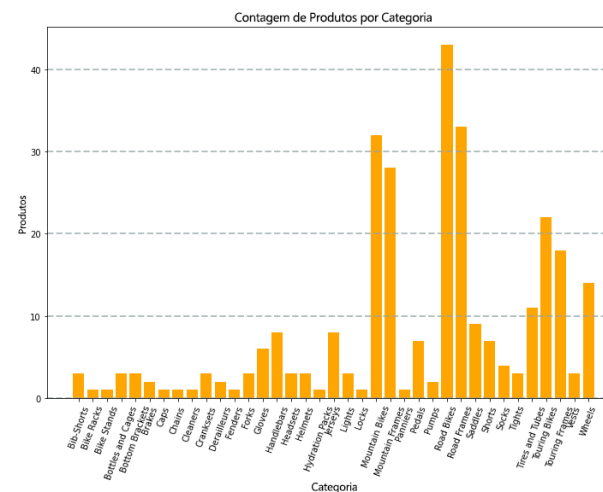
## Gráficos de notebook internos



## Pacotes gráficos

```
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>



# Usar o Delta Lake no Azure Synapse Analytics



# O que é o Delta Lake?

Camada de armazenamento de código aberto que adiciona semântica de banco de dados relacional ao Spark.

- Tabelas relacionais que dão suporte a consulta e modificação de dados
- Suporte para transações ACID
- Controle de versão de dados e *viagem no tempo*
- Suporte para dados em lote e de streaming
- Formatos padrão e interoperabilidade

# Criar tabelas do Delta Lake

## Criar uma tabela do Delta Lake de um 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)
```

## Fazer atualizações condicionais

```
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" })
```

## Consultar uma versão anterior (*viagem no tempo*)

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



# Criar tabelas de catálogo

## Tabelas gerenciadas

- Definidos sem um local específico – os arquivos são criados na pasta metastore
- Remover a tabela exclui os arquivos

## Tabelas externas

- Definido com um local de arquivo específico
- Remover a tabela não exclui os arquivos

```
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'
```

# Usar o Delta Lake com tipo de dados de streaming

## Usar a tabela Delta Lake como uma fonte de streaming

```
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()
```

## Usar a tabela Delta Lake como um coletor de streaming

```
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)
```

# Usar o Delta Lake em um pool de SQL

## Consultar arquivos de tabela delta usando OPENROWSET

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

## Consultar tabelas delta em bancos de dados do metastore do Spark

```
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>



# Leitura adicional

Executar a engenharia de dados com Pools do Apache Spark do Azure Synapse

<https://aka.ms/mslearn-spark-ptb>



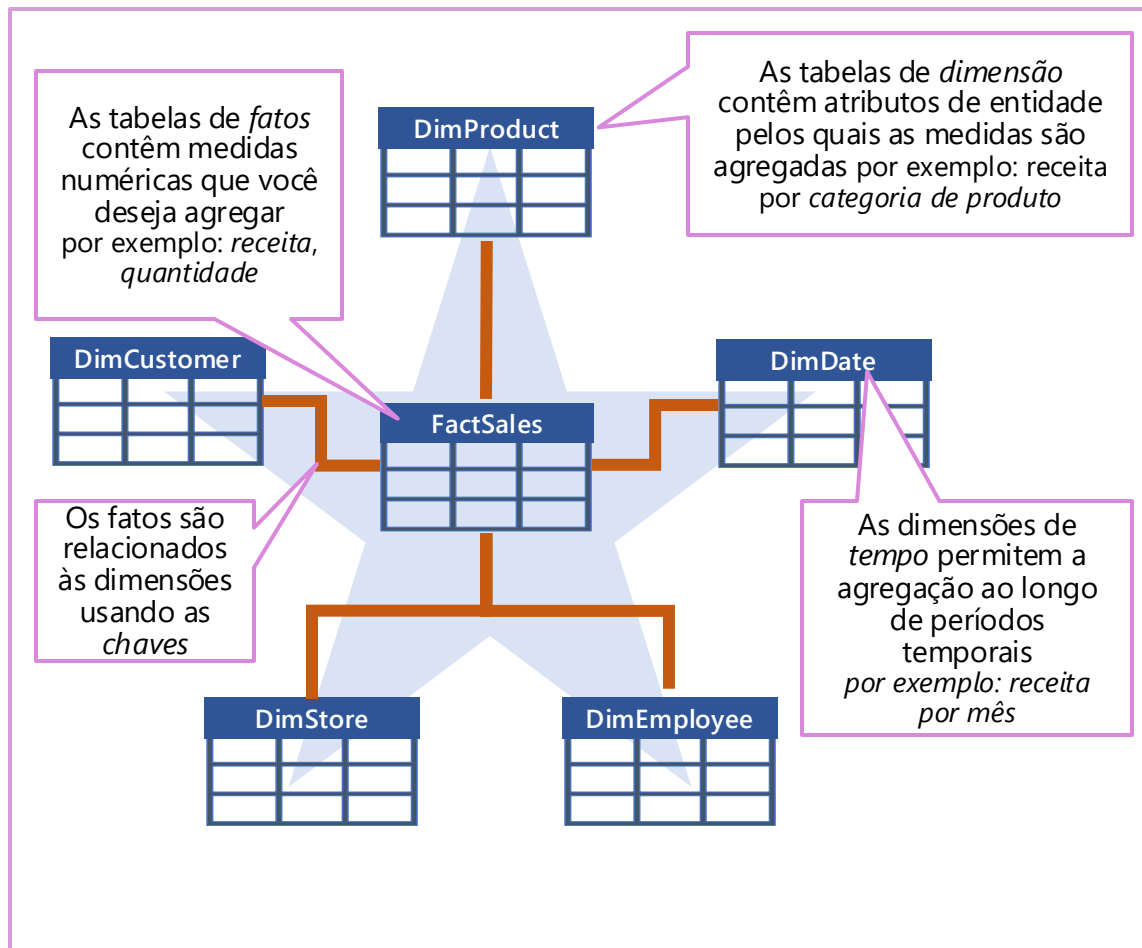
# Analisar s dados em um data warehouse relacional

<https://learn.microsoft.com/pt-br/training/modules/design-multidimensional-schema-to-optimize-analytical-workloads/>

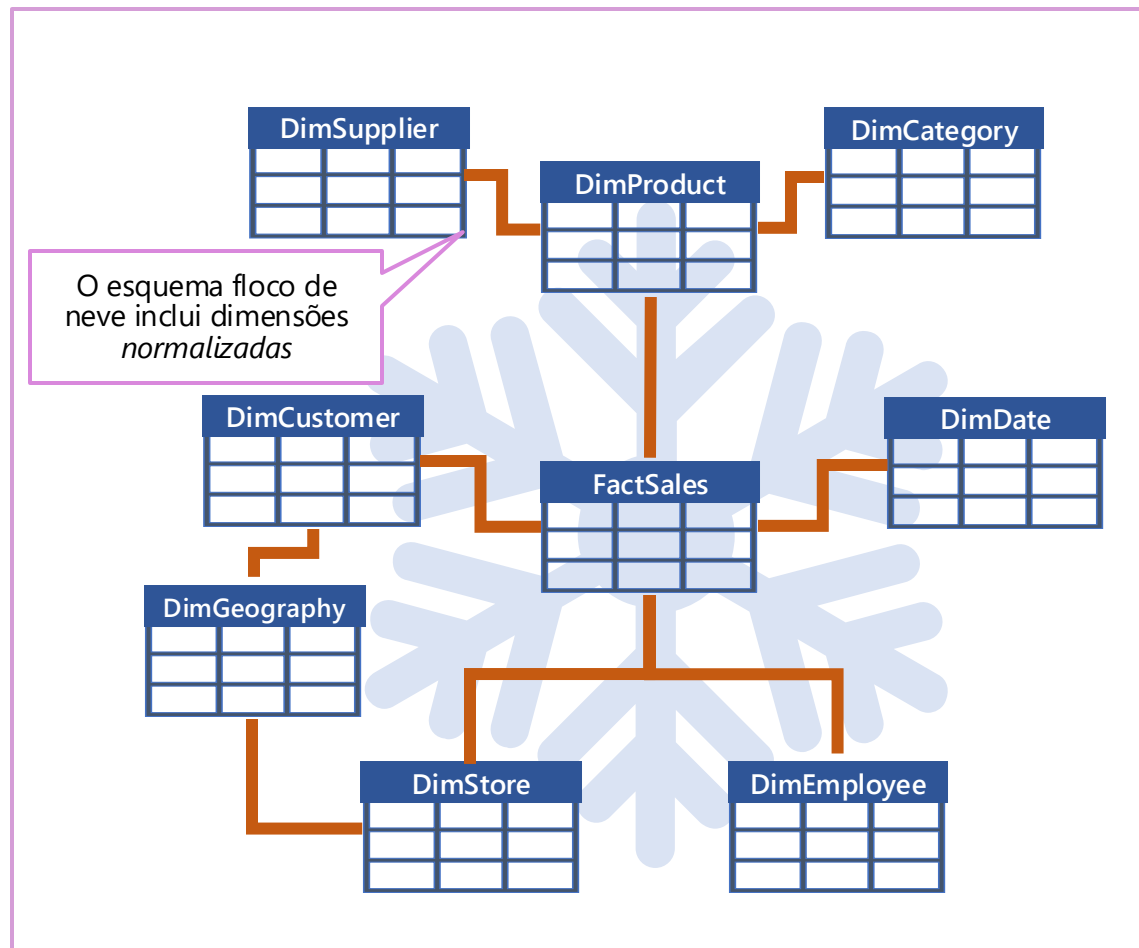


# Criar um esquema de data warehouse

## Esquema em estrela



## Esquema floco de neve





# Chaves de dimensão

## Chave alternativa

- Identifica exclusivamente uma instância de uma entidade de dimensão (ou seja, uma linha)
- Geralmente, um valor inteiro simples
- Deve ser exclusivo na tabela de dimensões

## Chave alternativa

- Identifica uma entidade no sistema de origem operacional
- Geralmente, uma chave *comercial* (por exemplo, um código de produto ou ID de cliente) ou uma chave *natural* (por exemplo, um valor datetime em uma dimensão de tempo)
- Pode ser duplicado na tabela de dimensões para representar a mesma entidade em diferentes pontos no tempo

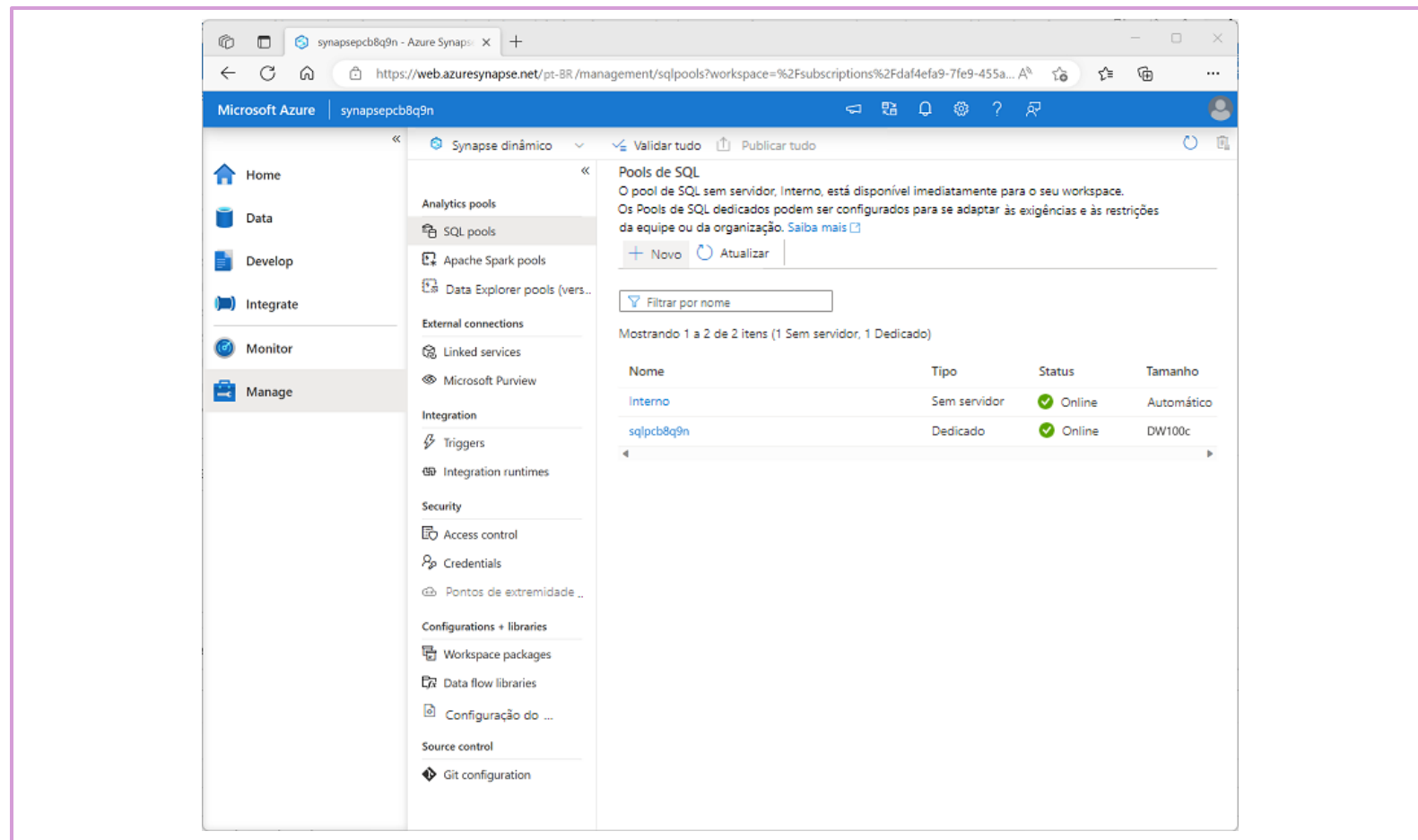
CustomerKey	CustomerAltKey	Nome	Email	Street	City	PostalCode	CountryRegion
123	I-543*	Navin Jones	navin1@contoso.com	1 Main St.	Seattle	90000	Estados Unidos
124	R-589	Mary Smith	mary2@contoso.com	234 190 <sup>th</sup> Ave	Buffalo	50001	Estados Unidos
125	I-321	Antoine Dubois	antoine1@contoso.com	2 Rue Jolie	Paris	20098	França
126	I-543*	Navin Jones	navin1@contoso.com	24 125 <sup>th</sup> Ave.	Nova Iorque	50000	Estados Unidos
...	...	...	...	...	...	...	...

\*Este cliente se mudou de Seattle para Nova York, portanto, um novo registro com a mesma chave alternativa, mas uma nova chave substituta, foi adicionada.

# Criar um data warehouse relacional no Azure Synapse Analytics

## Criar um pool de SQL *dedicado*

- Especificar nome e tamanho
- Pausar e retomar o pool conforme necessário
- O pool fornece uma instância de banco de dados relacional na qual você pode criar, carregar e consultar tabelas



# Considerações sobre a criação de tabelas de data warehouse

## Restrições de integridade dos dados

- Não há suporte para chave estrangeira e restrições exclusivas
- É necessário implementar a lógica para garantir a integridade referencial entre fatos e dimensões

## Índices

- O tipo de índice padrão é CLUSTERED COLUMNSTORE – Use isso na maioria dos casos
- Para tipos de campo não suportados em índices COLUMNSTORE, use um índice CLUSTERED em colunas apropriadas

## Distribuição de dados

- Use a distribuição de **hash** para distribuir tabelas de fatos entre nós de computação
- Use a distribuição **replicada** em tabelas de dimensões pequenas para evitar o embaralhamento de dados. No entanto, se a dimensão da tabela for muito grande para armazenar em cada nó de computação, use a distribuição de **hash**
- Use a distribuição **round-robin** em tabelas de preparo para distribuir os dados de modo uniforme entre os nós de computação

```
CREATE TABLE schema.table_name
(
    column_name DATA_TYPE, NULLABILITY,
    ...
)
WITH
(
    DISTRIBUTION = HASH(column_name)
                    | REPLICATE
                    | ROUND_ROBIN,
    INDEX_TYPE
)
```

# Tabelas externas

## Use tabelas externas para definir metadados de tabela para arquivos em um data lake

- Os dados são gerenciados independentemente da tabela
- Útil para leitura de dados em tabelas de preparo diretamente do data lake

```
CREATE EXTERNAL DATA SOURCE StagedFiles
WITH (
    LOCATION = 'https://.../file/location'
);
GO

CREATE EXTERNAL FILE FORMAT ParquetFormat
WITH (
    FORMAT_TYPE = PARQUET,
    DATA_COMPRESSION =
        'org.apache.hadoop.io.compress.SnappyCodec'
);
GO

CREATE EXTERNAL TABLE dbo.ExternalStageProduct
(
    ProductID NVARCHAR(10) NOT NULL,
    ProductName NVARCHAR(200) NOT NULL,
    ...
)
WITH
(
    DATA_SOURCE = StagedFiles,
    LOCATION = folder_name/*.parquet',
    FILE_FORMAT = ParquetFormat
);
GO
```

# Demo: Explore a data warehouse

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

<https://aka.ms/mslearn-synapse-dw>

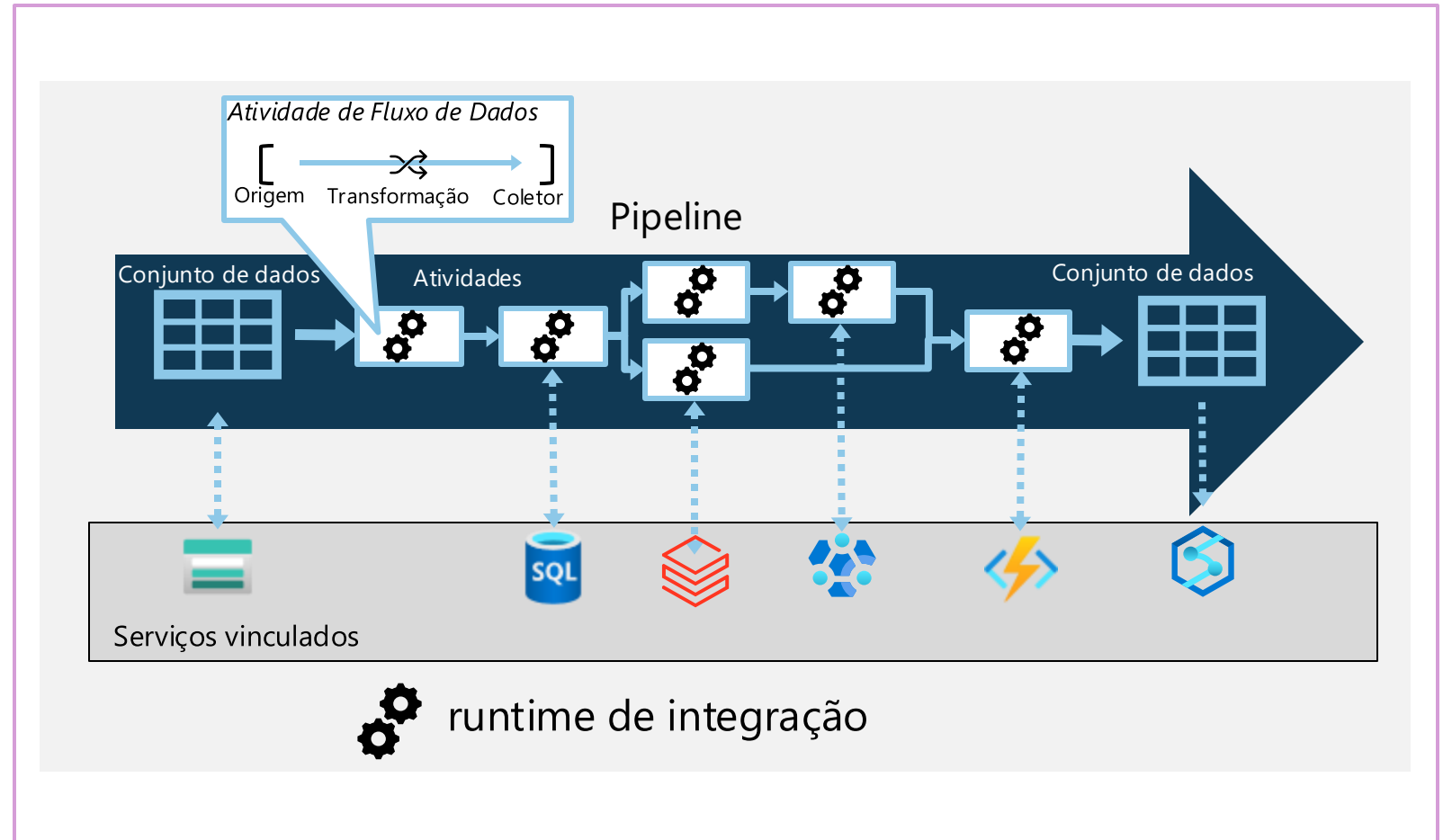


# Criar um pipeline de dados no Azure Synapse Analytics



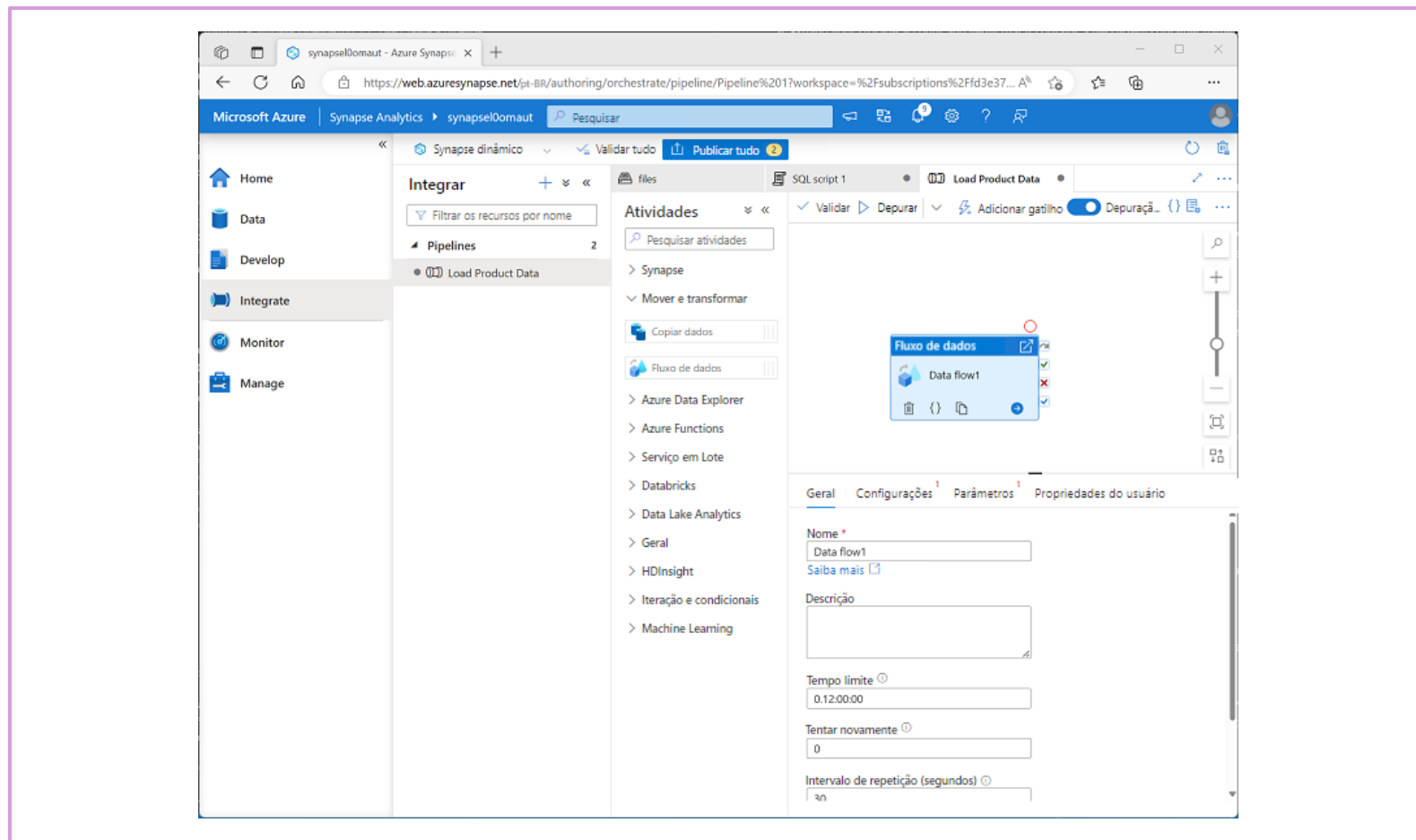
# Entender os pipelines

- Os pipelines encapsulam um fluxo de atividades que são orquestradas por um *runtime de integração*
- As atividades podem incluir:
  - Atividades de movimentação e transformação de dados que transferem dados de fontes para coletores
  - Atividades de processamento externo
  - Atividades de *fluxo de controle* que gerenciam variáveis e lógica de processamento
- Os serviços vinculados dão acesso a armazenamentos de dados e plataformas de processamento onde as atividades podem ser executadas
- Os dados processados em um pipeline são definidos em *conjuntos de dados*, acessados em serviços vinculados



# Criar um pipeline no Azure Synapse Studio

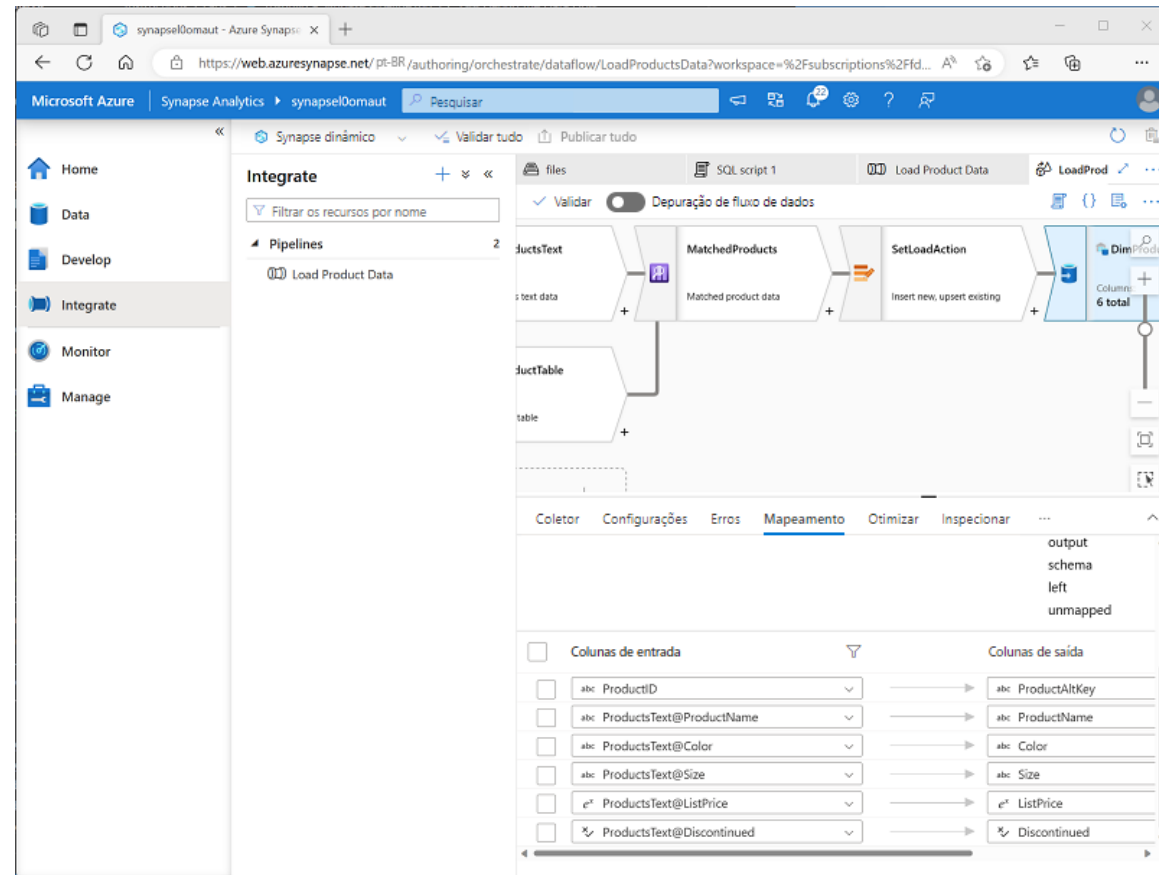
- Criar pipelines na página **Integrar**
- Adicionar e configurar atividades:
- Especificar conjuntos de dados novos ou existentes e serviços vinculados conforme necessário nas configurações
  - Eles serão adicionados às páginas **Dados** e **Gerenciar**
- Conectar atividades para definir o fluxo de processamento – Defina caminhos para:
  - Com sucesso
  - Com falha
  - Concluído





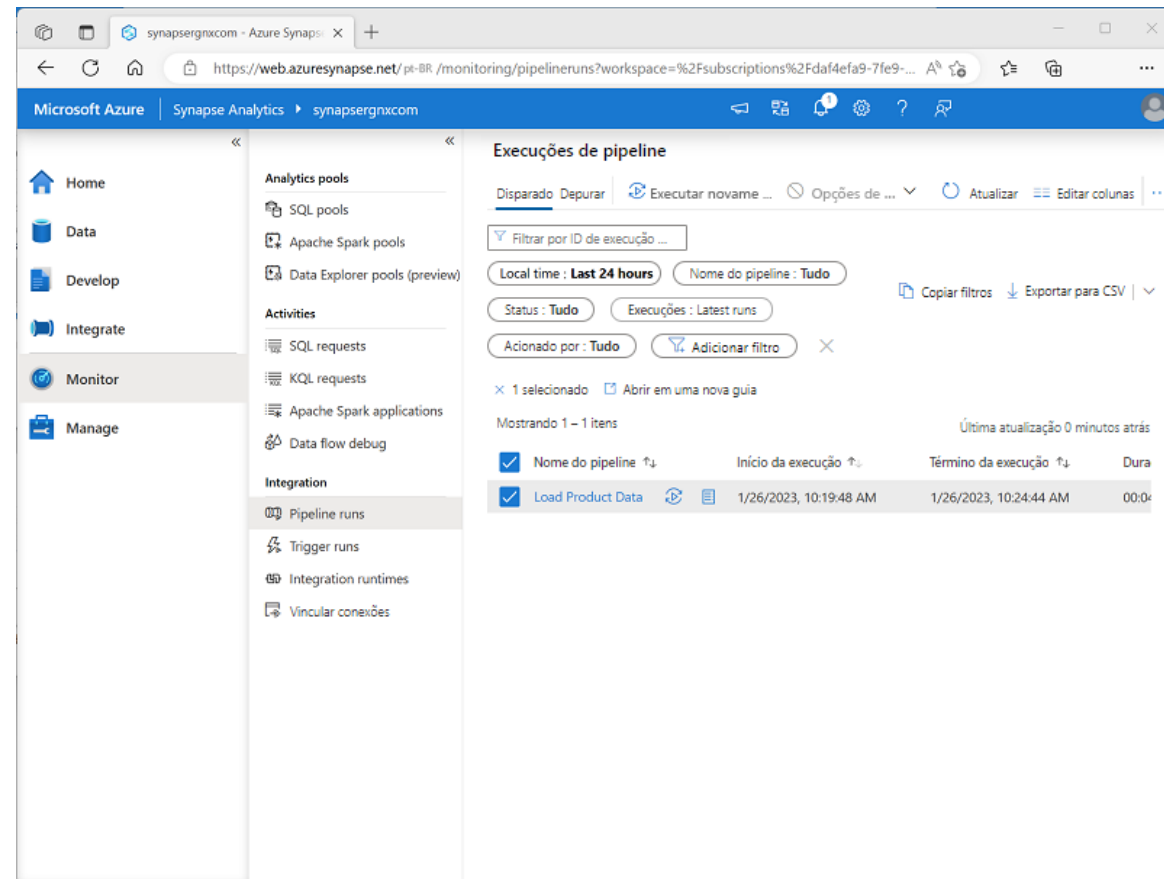
# Definir fluxos de dados

- Um **Fluxo de Dados** é um tipo de atividade muito usado para definir o fluxo e a transformação de dados
- Consiste em:
  - **Fontes** – Conjuntos de dados mapeados para armazenamentos de dados
  - **Transformações** – Operações em dados à medida que eles fluem pelo fluxo de dados
  - **Coletores** – Destinos para dados a serem carregados



# Executar um pipeline

- Depurar pipelines a testar durante o desenvolvimento
- Definir *gatilhos* para executar pipelines em produção:
  - Manual – executado imediatamente
  - Agendamento – executado em intervalos regulares
  - Evento – executado quando ocorre um evento (como o salvamento de novos dados em um armazenamento de dados)
- Monitorar execuções de pipeline no Azure Synapse Studio



# Exercício: criar um pipeline de dados no Azure Synapse Analytics

Use o ambiente de laboratório hospedado fornecido ou veja as instruções do laboratório no link abaixo:

<https://aka.ms/mslearn-build-synapse-pipeline>



# 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)

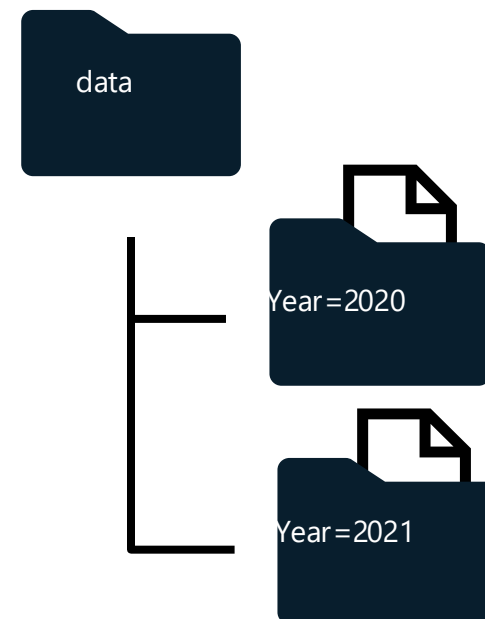
# Add Year column, derived from OrderDate
df = df.withColumn("Year",
                  year(col("OrderDate")))

# 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("transform
ed_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>

