# 😊 Creating a Dedicated SQL Pool and External Tables

A **Dedicated SQL Pool** in **Azure Synapse Analytics** (formerly Azure SQL Data Warehouse) is a resource that provides scalable, high-performance data warehousing capabilities within the Azure Synapse ecosystem. It is specifically designed for large-scale, high-volume analytical workloads.

**Key Features of Dedicated SQL Pool:**

1. **Massively Parallel Processing (MPP):**
    - The architecture leverages MPP, distributing data and queries across multiple nodes for high-performance parallel processing.
    - This enables the pool to handle large datasets efficiently.
2. **Scalability:**
    - Compute resources can be scaled up or down based on your workload requirements.
    - You pay only for the resources you use, and scaling can often be done without interrupting operations.
3. **Columnar Storage:**
    - Data is stored in a columnar format, which is highly optimized for analytical queries and reduces storage costs through compression.
4. **Data Distribution:**
    - Data is distributed across nodes using one of three strategies:
        - **Hash distribution:** Distributes rows based on a key column, useful for minimizing data movement during queries.
        - **Round-robin distribution:** Evenly distributes rows across nodes without a specific pattern.
        - **Replicated distribution:** Copies the entire table to each node, ideal for small tables used in joins.
5. **Integrations:**
    - Seamlessly integrates with other Azure services like Azure Data Factory, Azure Data Lake Storage, and Power BI.
    - Supports integration with on-premises data sources and other cloud services.
6. **Advanced Query Capabilities:**
    - Supports T-SQL for querying and managing data.

    - Includes advanced features like partitioning, indexing, and statistics to optimize query performance.

7. **Security:**
    - Provides enterprise-grade security, including role-based access control (RBAC), auditing, and encryption at rest and in transit.
8. **Concurrency Management:**
    - Optimized for managing multiple users and queries, balancing performance across concurrent operations.

**Common Use Cases:**

- **Enterprise Data Warehousing:** Centralized storage and analysis of organizational data for reporting and analytics.
- **Big Data Analytics:** Analyzing terabytes or petabytes of data with high query performance.
- **ETL and ELT Workflows:** Efficiently loading, transforming, and querying large datasets for data integration.

**Comparison with Serverless SQL Pool:**

- **Dedicated SQL Pool:**
    - Pre-allocated compute resources.
    - Best suited for predictable, high-volume workloads.
    - Charges based on provisioned resources, even when idle.
- **Serverless SQL Pool:**
    - On-demand querying with no pre-allocation.
    - Ideal for exploratory analytics or infrequent queries.
    - Charges based on data processed per query.

By utilizing Dedicated SQL Pools, organizations can run large-scale analytical queries efficiently, enabling faster decision-making and deeper insights from their data.

# 😄 Difference Between Dedicated SQL Pool and Serverless SQL Pool

The key differences between **Dedicated SQL Pool** and **Serverless SQL Pool** in **Azure Synapse Analytics** revolve around their architecture, use cases, pricing models, and operational approaches. Here's a detailed comparison:

**1. Provisioning and Architecture**

| Feature | Dedicated SQL Pool | Serverless SQL Pool |
|---|---|---|
| **Provisioning** | Pre-allocated compute resources (fixed cluster size). | No need for pre-allocation; runs on-demand. |
| **Storage** | Data is stored within the SQL pool (provisioned storage). | Queries external data directly from data lakes (e.g., Azure Data Lake). |

**2. Pricing Model**

| Feature | Dedicated SQL Pool | Serverless SQL Pool |
|---|---|---|
| **Billing** | Based on provisioned resources (compute and storage). | Based on the amount of data processed per query. |
| **Idle Cost** | Incurs charges even when not in use. | No charges when not in use. |

**3. Performance**

| Feature | Dedicated SQL Pool | Serverless SQL Pool |
|---|---|---|
| **Workload** | Designed for high-performance, predictable workloads. | Suited for lightweight, exploratory workloads. |
| **Scalability** | Scale up or down manually or programmatically. | Automatically handles workload with no scaling required. |

**4. Data Access and Querying**

| Feature | Dedicated SQL Pool | Serverless SQL Pool |
|---|---|---|
| **Data Storage** | Stores data in a structured, columnar format inside the pool. | Queries data directly from external sources (e.g., Parquet, CSV). |
| **Use Case** | For structured, preloaded datasets. | For ad-hoc querying of external datasets without loading them. |

**5. Use Cases**

| Feature | Dedicated SQL Pool | Serverless SQL Pool |
|---|---|---|
| **Best for** | Enterprise-grade data warehouses and predictable analytics. | Ad-hoc analytics, data exploration, or lightweight workloads. |
| **Integration** | Optimized for structured, relational data in Synapse. | Ideal for querying data stored in Azure Data Lake Storage. |

**6. Query Optimization**

| Feature | Dedicated SQL Pool | Serverless SQL Pool |
|---|---|---|
| **Performance Tuning** | Requires optimization like indexing, partitioning, and distribution strategies. | Automatically optimized for on-demand queries. |
| **Concurrency** | Can manage multiple concurrent queries with workload isolation. | Supports concurrent ad-hoc queries but is not designed for heavy concurrency. |

**Comparison Table Summary**

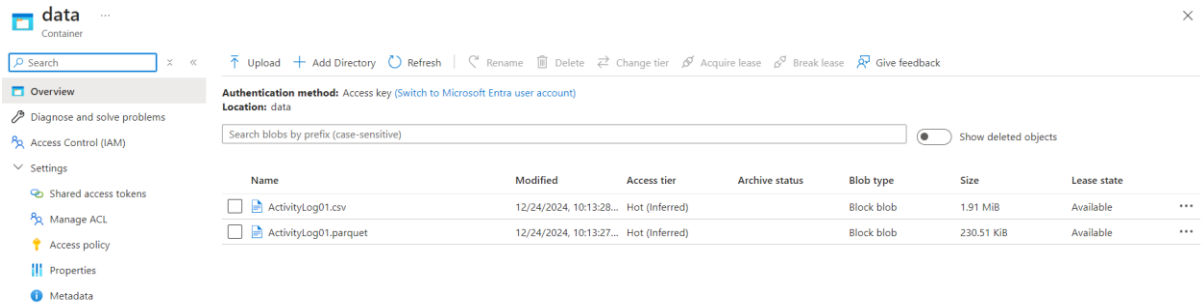
| Feature | Dedicated SQL Pool | Serverless SQL Pool |
|---|---|---|
| **Provisioning** | Fixed compute resources | No provisioning required |
| **Billing** | Based on provisioned resources | Based on data processed per query |
| **Idle Cost** | Charges even when idle | No charges when idle |
| **Use Case** | Large-scale, predictable workloads | Ad-hoc queries on external datasets |
| **Data Location** | Stored within the pool | External sources like Azure Data Lake |

**When to Choose:**

1. **Dedicated SQL Pool**:
    - Use for large-scale, high-performance workloads.
    - Ideal for scenarios where you need consistent, fast response times and handle structured data.
2. **Serverless SQL Pool**:
    - Use for exploratory data analysis or querying external data directly without data movement.
    - Ideal for infrequent or lightweight workloads with unpredictable demand.

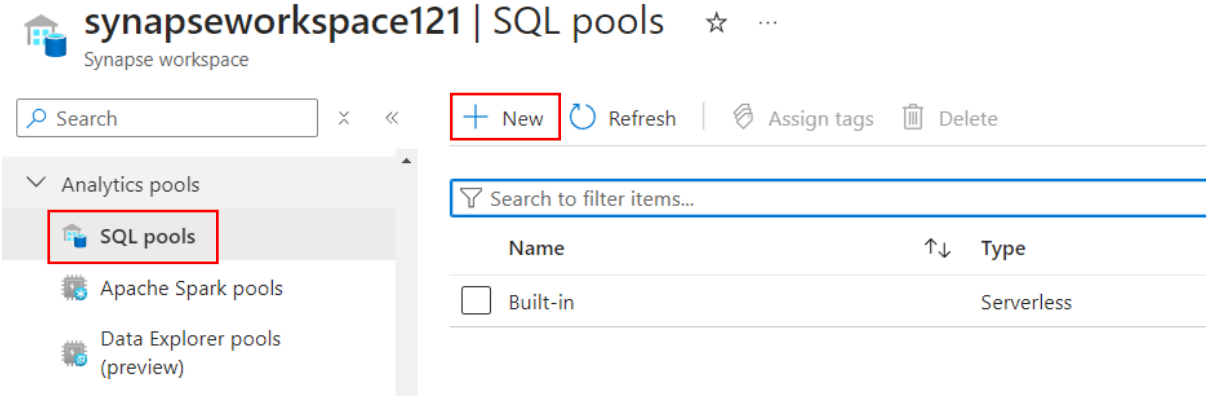
# 😄 To begin with this Lab:

1. In this lab, first we will create a dedicated SQL Pool in Azure Synapse, also known as a data warehouse. Then we will create some external and persisted tables in which we will load the data from our data lake or Gen 2 Storage account.
2. Here you can see that we have the storage account or data lake in which we have the parquet and CSV file with us.

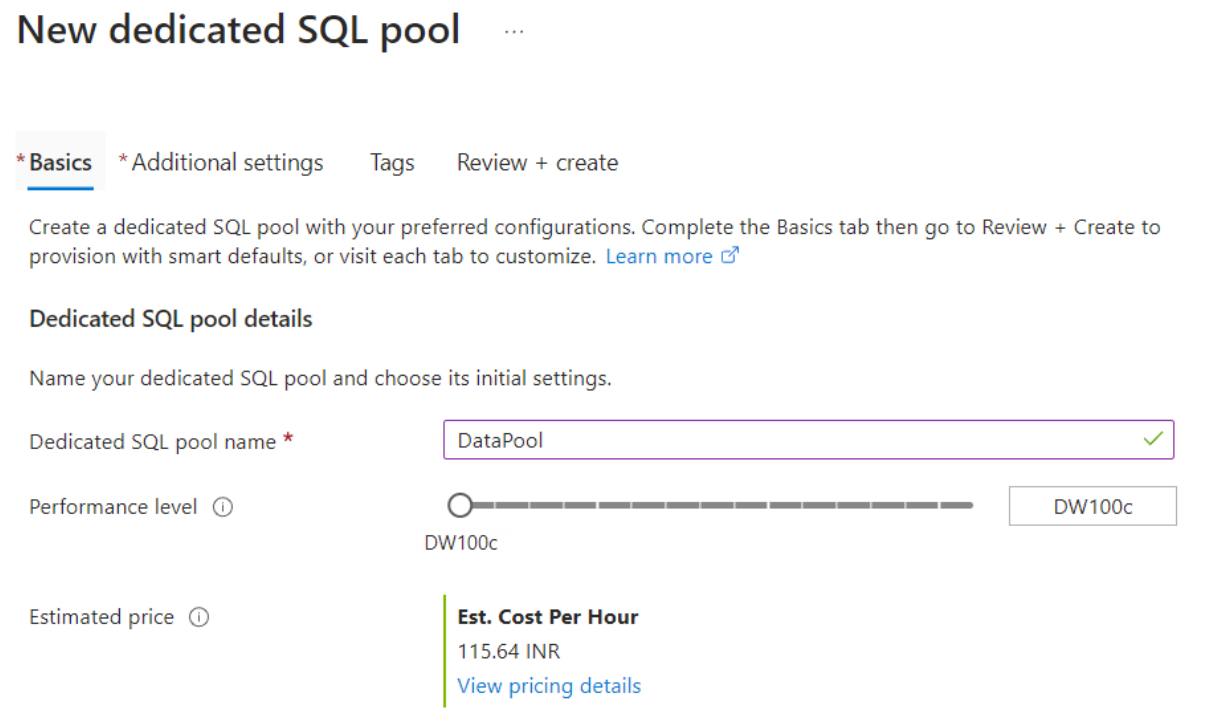



## Creating External Table CSV

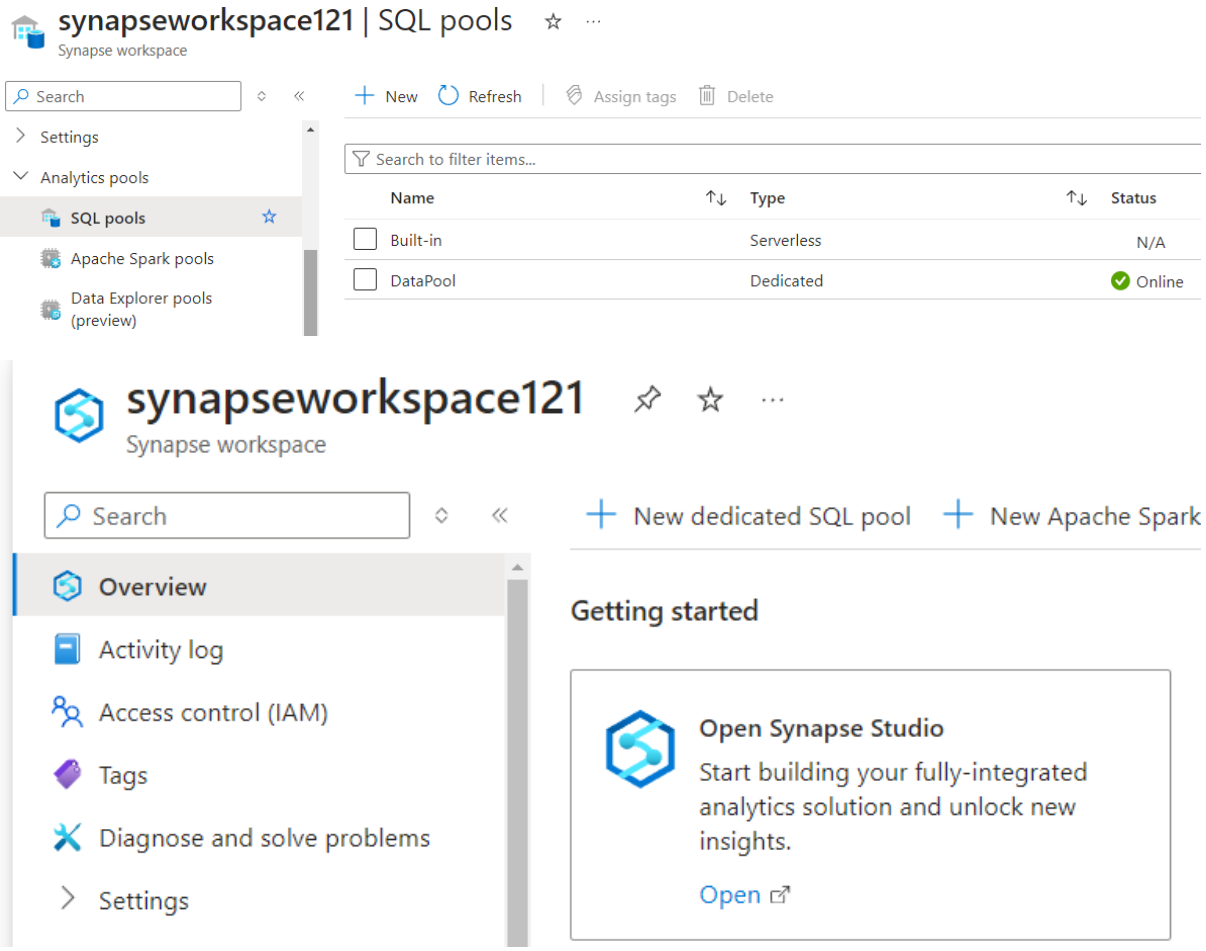
3. To create a dedicated SQL Pool, you must complete the previous 4 labs. Now open your Synapse Analytics and from the left pane come to Analytics Pools then choose SQL pools. Click on the new button to create a dedicated SQL Pool.



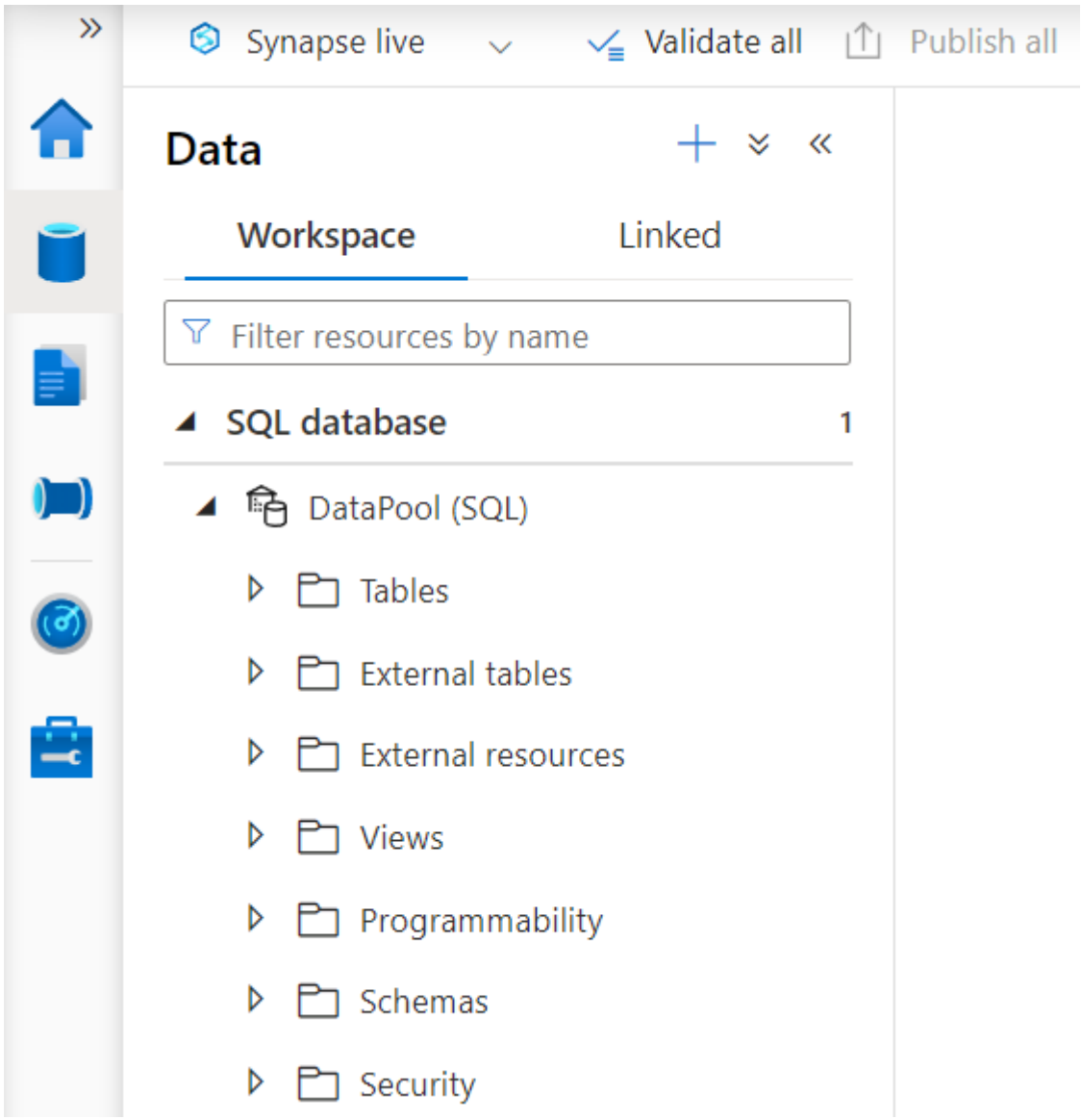

4. Give a name to your dedicated SQL Pool and choose the lowest Performance level which is DW100c and move to the Review page then create your SQL Pool or Data warehouse.



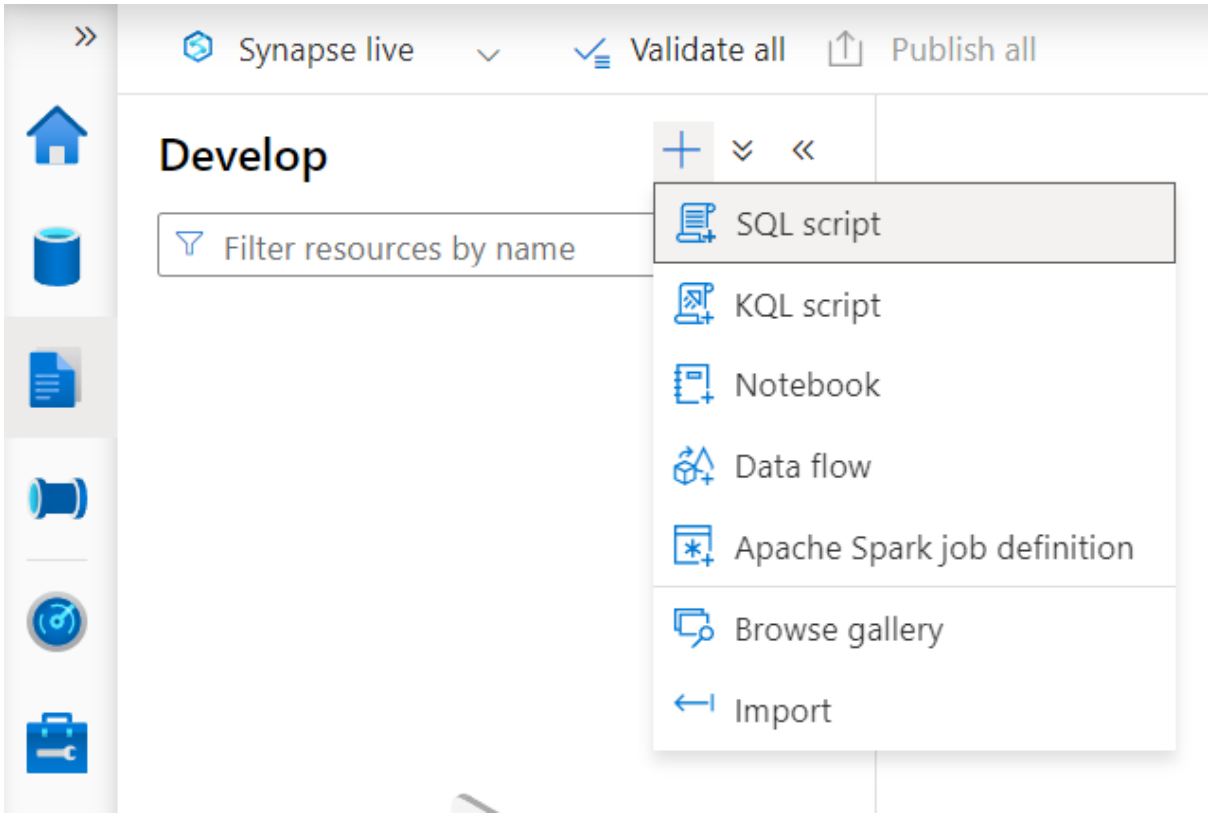

5. Now wait until the deployment is completed and you will see that in the SQL pools. Once it is done then open the Synapse Studio from the overview page.



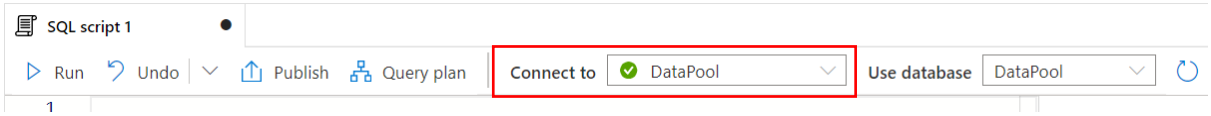
6. On Synapse Studio, go to the Data tab and in the workspace, you will see the dedicated SQL Pool as your Database here.




7. Now open the Develop Tab and click the Plus icon to create a new SQL Script.



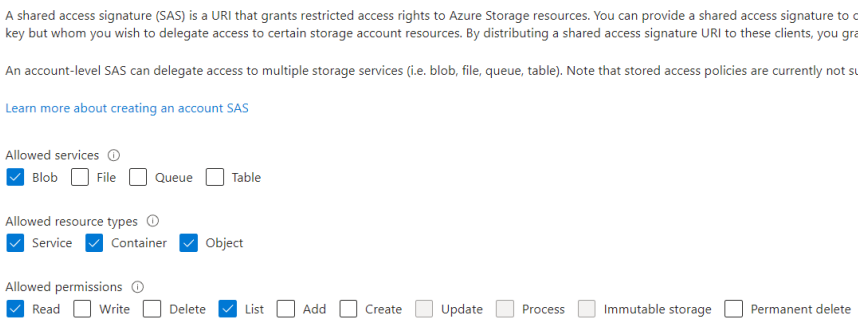

8. From the top you need to change the connection to the dedicated SQL Pool as you can see below in the snapshot.

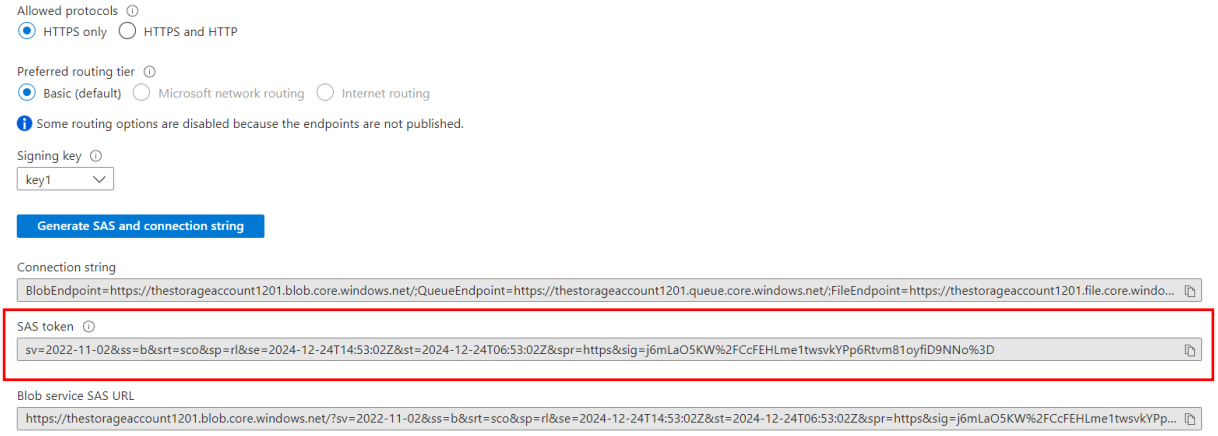



9. Now we will create an External table for CSV file inside our dedicated SQL Pool. First, we will start with creating the master key for the SAS Token and then go to the Storage account where your files have been stored. Then from the left pane choose Shared Access Signature and create an SAS Token for your code by choosing the allowed service, resource type, and permission.

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'P@ssword@123'

CREATE DATABASE SCOPED CREDENTIAL sasToken
WITH IDENTITY ='SHARED ACCESS SIGNATURE',
SECRET='sv=2022-11-02&ss=b&srt=sco&sp=rl&se=2024-12-24T14:53:02Z&st=2024-12-24T06:53:02Z&spr
```






10. Then we need to create the external data source for our table and here you can see that we are using different way to give the location type which is Hadoop, so use the same method first give the container name then follow with the @ and the storage account name. After that, we have to create an external file format as you can see below.

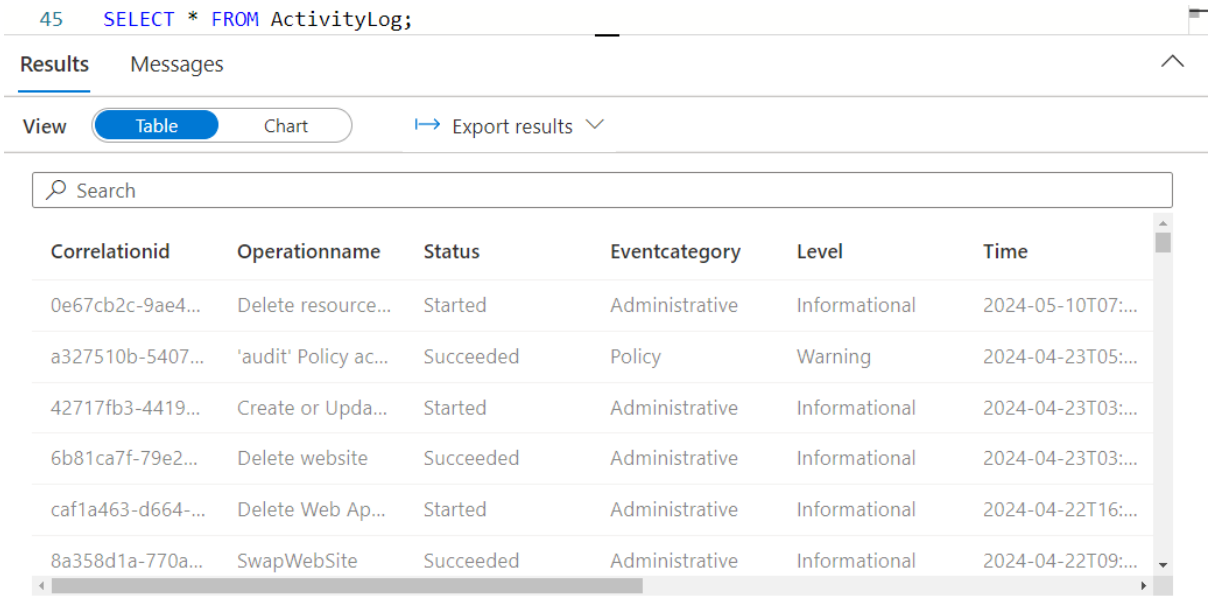
```
CREATE EXTERNAL DATA SOURCE srcActivityLog
WITH
(
    LOCATION='abfss://data@thestorageaccount1201.blob.core.windows.net',
    TYPE = HADOOP,
    CREDENTIAL=sasToken
)

 CREATE EXTERNAL FILE FORMAT delimitedTxtFileFormat WITH
 (
     FORMAT_TYPE=DELIMITEDTEXT,
     FORMAT_OPTIONS(
         FIELD_TERMINATOR=',',
         FIRST_ROW=2
     )
 )
```

11. Now we are going to create our external table in which the data will be fetched from the data lake.

```
26  CREATE EXTERNAL TABLE ActivityLog
27  (
28      [Correlationid] varchar(200),
29      [Operationname] varchar(300),
30      [Status] varchar(100),
31      [Eventcategory] varchar(100),
32      [Level] varchar(100),
33      [Time] varchar(100),
34      [Subscription] varchar(200),
35      [Eventinitiatedby] varchar(1000),
36      [Resourcetype] varchar(300),
37      [Resourcegroup] varchar(1000),
38      [Resource] varchar(2000))
39  WITH (
40      LOCATION='/ActivityLog01.csv',
41      DATA_SOURCE=srcActivityLog,
42      FILE_FORMAT=delimitedTxtFileFormat
43  )
```

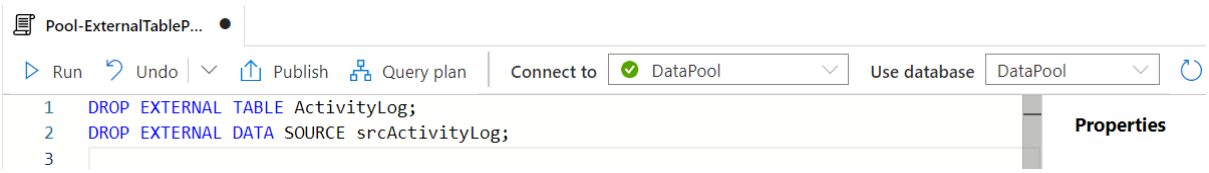
12. Below you can see that we have the data with us. Once you are done then publish your changes to save everything.




## Creating External Table Parquet

13. Now we will create the external table for the Parquet file, for that first we need to create a new SQL Script.

14. First, we will drop our external table and external data source and change the database from serverless to a dedicated SQL Pool.




```
1  DROP EXTERNAL TABLE ActivityLog;
2  DROP EXTERNAL DATA SOURCE srcActivityLog;
3
```

15. Then we will create the external data source for our parquet file, after that we will create the external file format.

```
5   CREATE EXTERNAL DATA SOURCE srcActivityLog
6   WITH
7   (
8       LOCATION='https://thestorageaccount1201.blob.core.windows.net/data',
9       CREDENTIAL=sasToken
10  )

12   CREATE EXTERNAL FILE FORMAT parquetFileFormat WITH
13   (
14       FORMAT_TYPE=PARQUET,
15       DATA_COMPRESSION='org.apache.hadoop.io.compress.SnappyCodec'
16   )
```

16. Here is our external parquet table which we need to create next.

```
18   CREATE EXTERNAL TABLE ActivityLog
19   (
20       [Correlationid] varchar(200),
21       [Operationname] varchar(300),
22       [Status] varchar(100),
23       [Eventcategory] varchar(100),
24       [Level] varchar(100),
25       [Time] varchar(100),
26       [Subscription] varchar(200),
27       [Eventinitiatedby] varchar(1000),
28       [Resourcetype] varchar(300),
29       [Resourcegroup] varchar(1000),
30       [Resource] varchar(2000))
31   WITH (
32       LOCATION='/ActivityLog01.parquet',
33       DATA_SOURCE=srcActivityLog,
34       FILE_FORMAT=parquetFileFormat
35   )
```

17. As you can see below we have the data inside our tables.

```
37   SELECT * FROM ActivityLog;
38
```

Results Messages

View Table Chart Export results

Search

| Correlationid | Operationname | Status | Eventcategory | Level | Time |
|---|---|---|---|---|---|
| e6c77441-737b... | 'auditIfNotExist... | Succeeded | Policy | Informational | 2024-05-15T11:... |
| b30f69ec-7203... | List Storage Ac... | Started | Administrative | Informational | 2024-05-15T06:... |
| 2274de78-8494... | Validate Deploy... | Started | Administrative | Informational | 2024-05-10T07:... |
| f469ad59-1a5f-... | 'audit' Policy ac... | Succeeded | Policy | Warning | 2024-05-10T06:... |

00:00:04 Query executed successfully.