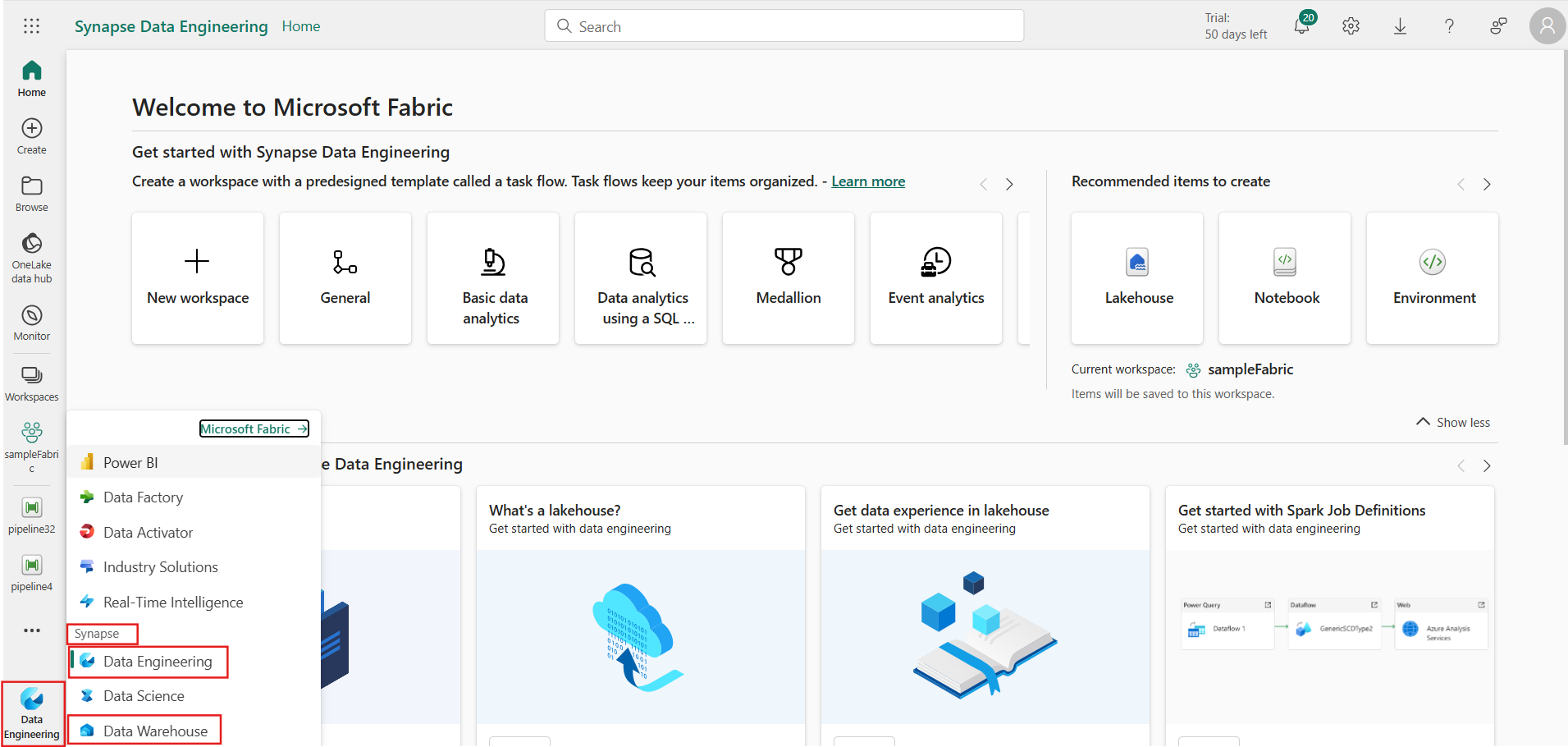
Azure Synapse Using Microsoft Fabric Synapse Data Engineering

# Microsoft Fabric



Creating or using existing Workspace

A screenshot of a computer

Description automatically generated

Creating a Pipeline

A screenshot of a computer

Description automatically generated

Click on create to create the pipeline  
**Note:** There is already a pipieline deployed will be using that pipeline

A screenshot of a computer

Description automatically generated

**Data Movement from Azure SQL to Fabric Data warehouse(Synapse):**

There are two pipelines

1. Master Pipeline(azureSql\_to\_FabricDataWarehouse) : To list all the databases dynamically inside lookup activity and passing them to Foreach activity which invokes another pipeline(Fetching tables from master pipeline)

A screenshot of a computer

Description automatically generated

1. Chile Pipeline(Fetching tables from master pipeline) : To list all the tables from the databases coming from master pipeline and loading them in the Fabric datawarehouse

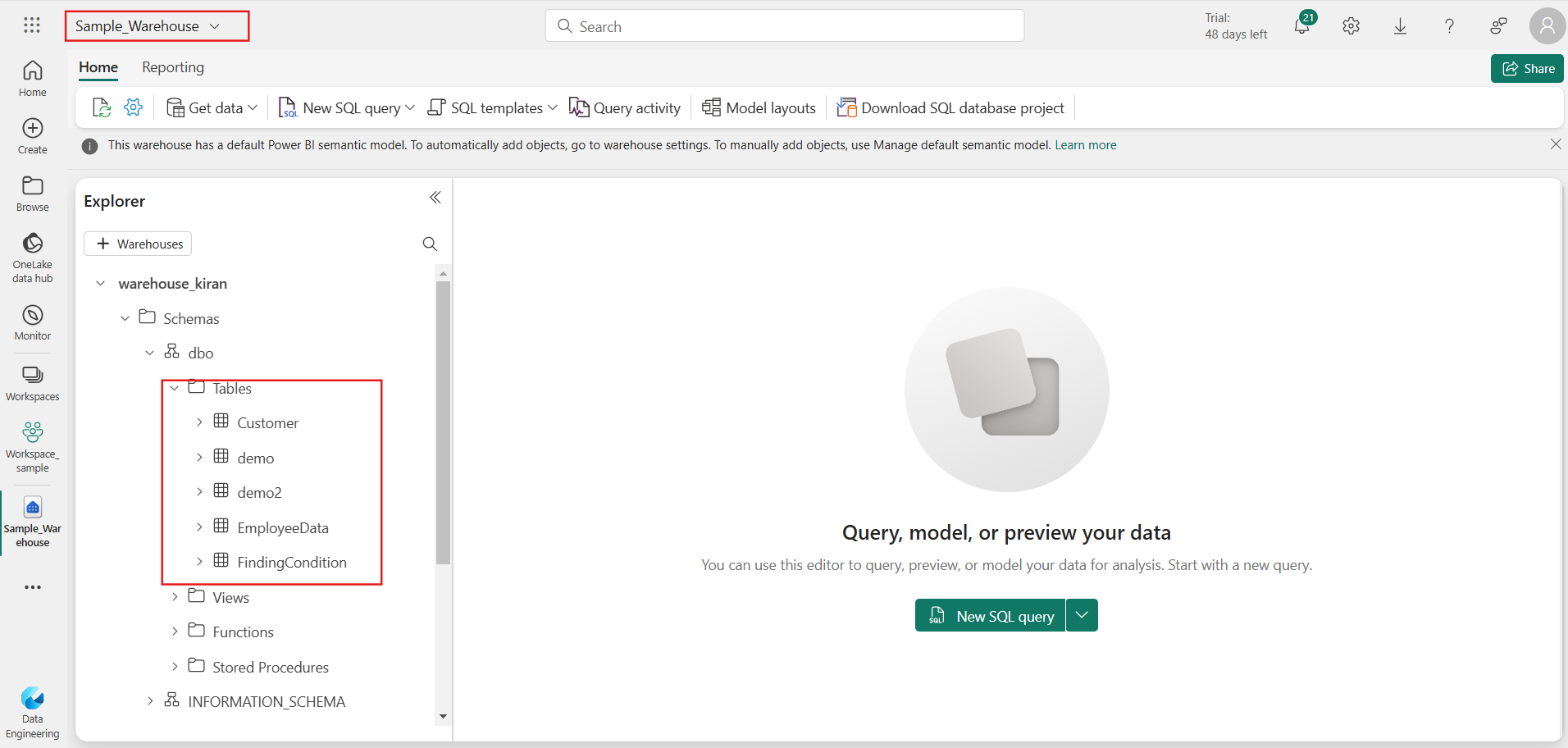
A screenshot of a computer

Description automatically generated

**Data Warehouse:**

Data warehouse is the Target where we will be deploying the data coming from the Azure Sql Database

Below snipshot displays all the tables migrating from multiple databases dynamically



**Monitoring :**

A screenshot of a computer

Description automatically generated

**Below is the JSON code to deploy the pipeline in Fabric**

# Master pipeline

{

    "name": "azureSql\_to\_FabricDatawarehouse",

    "objectId": "26085779-f527-4650-8c40-505e23cc59b1",

    "properties": {

        "description": "Dynamically moving data from azure sql to fabric data warehouse",

        "activities": [

            {

                "name": "Lookup1",

                "type": "Lookup",

                "dependsOn": [],

                "policy": {

                    "timeout": "0.12:00:00",

                    "retry": 0,

                    "retryIntervalInSeconds": 30,

                    "secureOutput": false,

                    "secureInput": false

                },

                "typeProperties": {

                    "source": {

                        "type": "AzureSqlSource",

                        "sqlReaderQuery": "select name from sys.databases where name <> 'master'\n",

                        "queryTimeout": "02:00:00",

                        "partitionOption": "None"

                    },

                    "firstRowOnly": false,

                    "datasetSettings": {

                        "annotations": [],

                        "type": "AzureSqlTable",

                        "schema": [],

                        "typeProperties": {

                            "database": {

                                "value": "@pipeline().parameters.DBNAME",

                                "type": "Expression"

                            }

                        },

                        "externalReferences": {

                            "connection": "beddee87-6573-4595-95fa-872dfc67da17"

                        }

                    }

                }

            },

            {

                "name": "ForEach1",

                "type": "ForEach",

                "dependsOn": [

                    {

                        "activity": "Lookup1",

                        "dependencyConditions": [

                            "Succeeded"

                        ]

                    }

                ],

                "typeProperties": {

                    "items": {

                        "value": "@activity('Lookup1').output.value",

                        "type": "Expression"

                    },

                    "isSequential": true,

                    "activities": [

                        {

                            "name": "Invoke pipeline1",

                            "type": "InvokePipeline",

                            "dependsOn": [],

                            "policy": {

                                "timeout": "0.12:00:00",

                                "retry": 0,

                                "retryIntervalInSeconds": 30,

                                "secureOutput": false,

                                "secureInput": false

                            },

                            "typeProperties": {

                                "waitOnCompletion": true,

                                "operationType": "InvokeFabricPipeline",

                                "pipelineId": "888b34ac-e9f4-442d-af2b-9793e78bc62a",

                                "workspaceId": "d1fb10a9-77a7-48e8-8f72-f27b7ac6d0b4",

                                "parameters": {

                                    "dbname": {

                                        "value": "@item().name",

                                        "type": "Expression"

                                    }

                                }

                            },

                            "externalReferences": {

                                "connection": "5a3083a2-5205-49ba-8e0c-5c1b0ad4ea98"

                            }

                        }

                    ]

                }

            }

        ],

        "parameters": {

            "DBNAME": {

                "type": "string",

                "defaultValue": "master"

            }

        },

        "lastModifiedByObjectId": "0b6c08b7-cc03-4a57-b842-17cda59083a9",

        "lastPublishTime": "2024-11-04T05:57:49Z"

    }

}

## Child pipeline1

{

    "name": "Fetching tables from master pipeline",

    "objectId": "888b34ac-e9f4-442d-af2b-9793e78bc62a",

    "properties": {

        "description": "Fetching all the tables and loading data into Fabric Datawarehouse\n",

        "activities": [

            {

                "name": "Lookup1",

                "type": "Lookup",

                "dependsOn": [],

                "policy": {

                    "timeout": "0.12:00:00",

                    "retry": 0,

                    "retryIntervalInSeconds": 30,

                    "secureOutput": false,

                    "secureInput": false

                },

                "typeProperties": {

                    "source": {

                        "type": "AzureSqlSource",

                        "sqlReaderQuery": {

                            "value": "@concat('select TABLE\_SCHEMA,table\_name from ',pipeline().parameters.dbname,'.INFORMATION\_SCHEMA.TABLES\nwhere TABLE\_SCHEMA <> ''sys''')",

                            "type": "Expression"

                        },

                        "queryTimeout": "02:00:00",

                        "partitionOption": "None"

                    },

                    "firstRowOnly": false,

                    "datasetSettings": {

                        "annotations": [],

                        "type": "AzureSqlTable",

                        "schema": [],

                        "typeProperties": {

                            "database": {

                                "value": "@pipeline().parameters.dbname",

                                "type": "Expression"

                            }

                        },

                        "externalReferences": {

                            "connection": "beddee87-6573-4595-95fa-872dfc67da17"

                        }

                    }

                }

            },

            {

                "name": "ForEach1",

                "type": "ForEach",

                "dependsOn": [

                    {

                        "activity": "Lookup1",

                        "dependencyConditions": [

                            "Succeeded"

                        ]

                    }

                ],

                "typeProperties": {

                    "items": {

                        "value": "@activity('Lookup1').output.value",

                        "type": "Expression"

                    },

                    "isSequential": true,

                    "activities": [

                        {

                            "name": "Copy data1",

                            "type": "Copy",

                            "dependsOn": [],

                            "policy": {

                                "timeout": "0.12:00:00",

                                "retry": 0,

                                "retryIntervalInSeconds": 30,

                                "secureOutput": false,

                                "secureInput": false

                            },

                            "typeProperties": {

                                "source": {

                                    "type": "AzureSqlSource",

                                    "queryTimeout": "02:00:00",

                                    "partitionOption": "None",

                                    "datasetSettings": {

                                        "annotations": [],

                                        "type": "AzureSqlTable",

                                        "schema": [],

                                        "typeProperties": {

                                            "schema": {

                                                "value": "@item().TABLE\_SCHEMA",

                                                "type": "Expression"

                                            },

                                            "table": {

                                                "value": "@item().table\_name",

                                                "type": "Expression"

                                            },

                                            "database": {

                                                "value": "@pipeline().parameters.dbname",

                                                "type": "Expression"

                                            }

                                        },

                                        "externalReferences": {

                                            "connection": "beddee87-6573-4595-95fa-872dfc67da17"

                                        }

                                    }

                                },

                                "sink": {

                                    "type": "DataWarehouseSink",

                                    "allowCopyCommand": true,

                                    "tableOption": "autoCreate",

                                    "datasetSettings": {

                                        "annotations": [],

                                        "linkedService": {

                                            "name": "warehouse\_kiran",

                                            "properties": {

                                                "annotations": [],

                                                "type": "DataWarehouse",

                                                "typeProperties": {

                                                    "endpoint": "kih36edybf2epb65bzecmdxu3e-veipxunho7uerd3s6j5xvrwqwq.datawarehouse.fabric.microsoft.com",

                                                    "artifactId": "024bf267-ba09-4066-804b-dca22152a626",

                                                    "workspaceId": "d1fb10a9-77a7-48e8-8f72-f27b7ac6d0b4"

                                                }

                                            }

                                        },

                                        "type": "DataWarehouseTable",

                                        "schema": [],

                                        "typeProperties": {

                                            "schema": {

                                                "value": "@item().TABLE\_SCHEMA",

                                                "type": "Expression"

                                            },

                                            "table": {

                                                "value": "@item().table\_name",

                                                "type": "Expression"

                                            }

                                        }

                                    }

                                },

                                "enableStaging": true,

                                "translator": {

                                    "type": "TabularTranslator",

                                    "typeConversion": true,

                                    "typeConversionSettings": {

                                        "allowDataTruncation": true,

                                        "treatBooleanAsNumber": false

                                    }

                                }

                            }

                        }

                    ]

                }

            }

        ],

        "parameters": {

            "dbname": {

                "type": "string"

            }

        },

        "lastModifiedByObjectId": "0b6c08b7-cc03-4a57-b842-17cda59083a9",

        "lastPublishTime": "2024-11-04T05:59:53Z"

    }

}

# Performance and Optimization :

## In Microsof Fabric:

Performance and optimization in Microsoft Fabric focus on delivering high-efficiency data operations through advanced technologies and tools tailored for analytical workloads

**1. Optimized Query Execution**

* Microsoft Fabric’s Data Warehouse engine supports **automatic statistics computation**, where query performance is enhanced by automatically generating the necessary statistics as queries are executed. This helps in providing more efficient query plans without manual intervention​.
* The use of **adaptive query execution** techniques helps the engine dynamically adjust execution strategies based on runtime statistics, ensuring optimized performance for different types of queries.

**2. Data Caching and Management**

* **Data caching** mechanisms help speed up query responses by keeping frequently accessed data in memory or on faster storage layers.
* **OneLake Integration**: Microsoft Fabric leverages OneLake as a central data repository, which allows for seamless data sharing and improved performance through a single copy of data. This minimizes redundancy and optimizes storage utilization​.

**3. Scalability and Resource Management**

* The architecture supports **scaling out resources** to manage concurrency and large workloads. This is particularly important for organizations needing to process massive datasets efficiently.
* **Resource autoscaling** dynamically adjusts the compute power based on the current workload, ensuring resources are utilized optimally without over-provisioning or underutilization.

**4. Performance Monitoring and Tuning**

* Microsoft Fabric includes a **Monitoring Hub** where users can get insights into query execution, performance bottlenecks, and overall resource consumption. This helps data engineers and administrators to fine-tune workloads and optimize performance​.
* **Workload Management**: Users can prioritize or de-prioritize specific workloads based on business needs, ensuring critical operations get the necessary resources.

**5. Data Model Optimization**

* The platform encourages the use of **columnstore indexes** for large, read-heavy analytical tables. This significantly reduces storage footprint and accelerates query performance.
* **Data compression techniques** are employed to minimize storage costs and enhance query speed, especially when working with extensive datasets.

**6. Integration and Seamless Experience**

* The integration with familiar tools like Power BI and Azure Synapse allows for an optimized analytics experience across the data lifecycle. Using built-in optimization features in these connected tools can streamline workflows and improve overall system efficiency

## In Azure Synapse Analytics:

Performance and optimization in Azure Synapse Analytics rely on a variety of techniques and best practices designed to handle complex analytical workloads efficiently.

**1. Data Distribution Strategies**

* **Table Distribution**: In Synapse, you can choose between *Round Robin*, *Hash*, or *Replicated* distribution for tables. Selecting the right distribution method helps balance the data across nodes and optimize parallel processing. For large tables, using *Hash Distribution* with a key that minimizes data skew is typically recommended.
* **Data Skew Management**: Monitoring data distribution to avoid skew ensures that data processing workloads are evenly distributed, which helps prevent performance bottlenecks.

**2. Indexing and Partitioning**

* **Clustered Columnstore Index**: This is the default for tables in Synapse Analytics and is optimized for large, read-heavy analytical queries. Columnstore indexes reduce storage size and improve query performance by compressing data.
* **Table Partitioning**: Dividing a table into partitions based on date or another commonly queried column can speed up query performance by allowing the system to read only relevant data partitions.

**3. Query Optimization Techniques**

* **Result Caching**: By caching query results, Synapse can return results faster for repeated queries. This is particularly useful for dashboards and repetitive analytical reports.
* **Query Hints and Options**: Using options like OPTION (HASH JOIN), OPTION (RECOMPILE), or setting a *query timeout* can help optimize performance by directing the query execution plan.

**4. Resource Management and Scaling**

* **Resource Class Management**: Synapse uses resource classes to manage concurrency and allocate resources. Assigning queries to appropriate resource classes based on workload size can improve performance.
* **Scaling Compute Resources**: Synapse allows you to scale up or down the compute resources (DWU - Data Warehouse Units) based on workload demands. This flexibility ensures optimal performance during peak and non-peak times.

**5. Workload Management**

* **Concurrency Limits**: Azure Synapse limits the number of concurrent queries based on your data warehouse unit (DWU) configuration. Managing concurrency effectively and setting up workload management rules helps distribute resources efficiently.
* **Workload Isolation**: You can create *dedicated SQL pools* for specific departments or heavy workloads to isolate and optimize performance for critical operations.

**6. Data Loading and Processing**

* **PolyBase for Data Ingestion**: Using PolyBase to load data from Azure Blob Storage or Azure Data Lake Storage is efficient for handling large volumes of data.
* **Batch Loading**: Loading data in batches rather than row-by-row helps optimize data ingestion performance. You can also use COPY INTO for efficient data loading.

**7. Monitoring and Diagnostics**

* **SQL Analytics Monitoring**: Synapse provides monitoring features to track query performance, data movement, and resource utilization. Utilizing the *Query Performance Insight* and *SQL Analytics Monitoring Dashboard* helps identify and fix performance issues.
* **Dynamic Management Views (DMVs)**: These views give insights into system health, query execution, and data distribution, allowing you to fine-tune performance.

**8. Materialized Views**

* **Materialized Views**: Synapse supports materialized views, which pre-compute and store query results, speeding up complex query performance for repeated access patterns.

**9. Optimize Data Storage**

* **Compression and File Format**: Storing data in optimized formats like Parquet or ORC can improve performance because these formats are compressed and faster to read.
* **Column Pruning and Predicate Pushdown**: Synapse automatically optimizes query performance by pruning unnecessary columns and pushing down query predicates closer to the data.