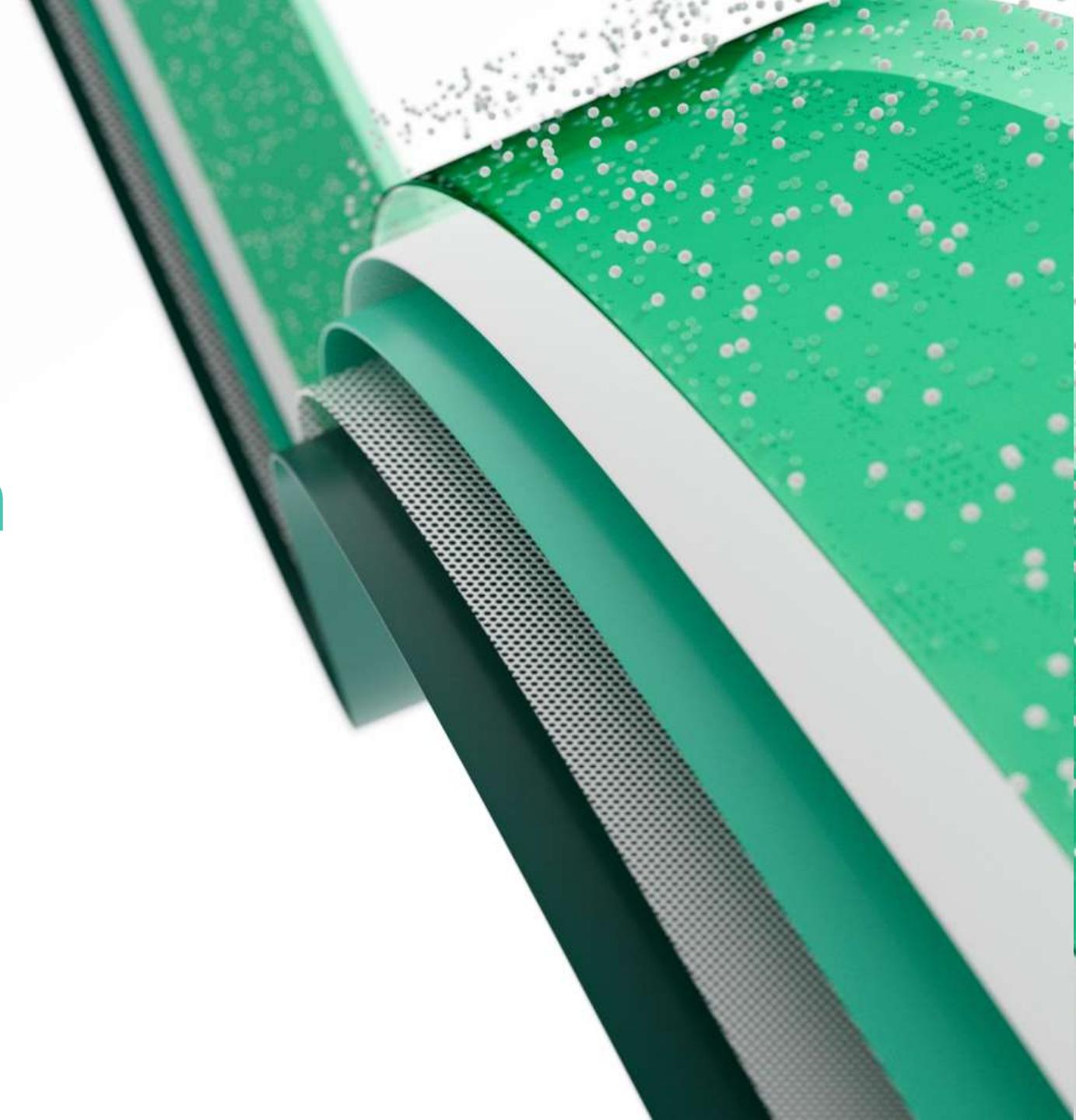


Fabric Certification Academy Day



WARNING

This deck is a “pre-release”. We’re working on a cohesive deck that you can leverage for re-delivery of this content.

This preview is targeted to Fabric Certification Academy Day attendees from the Microsoft France session on April 4th 2024.





Thank you!



Agenda (times are approximate and will be fluid with the class)

Morning

09:30 AM – 09:45 AM	Welcome
09:45 AM – 10:00 AM	Study guide for exam DP-600
10:00 AM – 10:45 AM	Module 1: Plan, implement, and manage a solution for data analytics
10:45 AM – 11:00 AM	Break
11:00 AM – 12:30 PM	Module 2: Prepare and serve data
12:30 PM – 02:00 PM	Break for lunch

Afternoon

02:00 PM – 02:45 PM	Module 3: Implement and manage semantic model
02:45 PM – 03:30 PM	Module 4: Explore and analyze data
03:30 PM – 03:45 PM	Break
03:45 PM – 04:45 PM	Practice exam
04:45 PM – 05:00 PM	Prepare for Exam Day
05:00 PM – 05:30 PM	Q&A Session



Fabric Certification Academy Day

Meet the audience

Raise your hand if...

- You're working for a Microsoft Partner
- You already passed a Microsoft Certification
- You already played with Fabric
- You have a Proof of Concept in progress with Fabric
- You already have a production workload on Fabric
- You are a member of a Fabric/Power BI user group



Microsoft Fabric Hands-on Day

Meet your proctors



E X A M

Implementing Analytics Solutions Using Microsoft Fabric

Passed

March 13, 2024 • Online • Provided by PearsonVue

[View details on provider site ↗](#)

Frédéric Gisbert
Cloud Solutions Architect
Partenaires

Christopher Maneu
Cloud Advocate Data
Azure Engineering



DP-600 Practice test



Access To the
Practice test
[Aka.ms/dp600-practice](https://aka.ms/dp600-practice)

Practice Assessment for Exam DP-600: Implementing Analytics Solutions Using Microsoft Fabric

Question 1 of 50

You are planning the configuration of a new Fabric tenant.

You need to recommend a solution to ensure that reports meet the following requirements:

- Require authentication for embedded reports.
- Allow only read-only (live) connections against Fabric capacity cloud semantic models.

Which two actions should you recommend performing from the Fabric admin portal? Each correct answer presents part of the solution.

From Capacity settings, set XMLA Endpoint to Read Write.

From Embed Codes, delete all existing codes.

From Premium Per User, set XMLA Endpoint to Off.

From Tenant settings, disable Allow XMLA endpoints and Analyze in Excel with on-premises semantic models.

From Tenant settings, disable Publish to web.

[Next >](#) [Check Your Answer](#)



Fabric Certification Academy Day resources

Access Wifi

SSID:

Event Code:



Get resources
[Aka.ms/fabric-cert-day](https://aka.ms/fabric-cert-day)

DP-600 Exam overview

Exam DP-600
Implementing **Analytics Solutions**
Using Microsoft Fabric

Skills measured

Plan, implement, and manage a solution for data analytics (10–15%)

Prepare and serve data (40–45%)

Implement and manage semantic models (20–25%)

Explore and analyze data (20–25%)

Certification versus Reality

We ship new product features every week

The certification “training date” is March 15th.

If the content is updated, it will be notified in the study guide.

Useful links	Description
Review the skills measured as of April 24, 2024	This list includes the skills that will be measured on April 24, 2024. This list is subject to change.
Review the skills measured prior to April 24, 2024	Study these skills to prepare for the certification exam.

Important

The English language version of this certification will be updated on April 25, 2024. Review the study guide linked in the Certification resources section for details about upcoming changes.

1



Module 1
Plan, implement, and manage
a solution for data analytics

Plan, implement, and manage a solution for data analytics (10-15%)

Plan a data analytics environment

Identify requirements for a solution, including components, features, performance, and capacity stock-keeping units (SKUs)

Recommend settings in the Fabric admin portal

Choose a data gateway type

Create a custom Power BI report theme

Implement and manage a data analytics environment

Implement workspace and item-level access controls for Fabric items

Implement data sharing for workspaces, warehouses, and lakehouses

Manage sensitivity labels in semantic models and lakehouses

Configure Fabric-enabled workspace settings
Manage Fabric capacity

Manage the analytics development lifecycle

Implement version control for a workspace

Create and manage a Power BI Desktop project (.pbip)

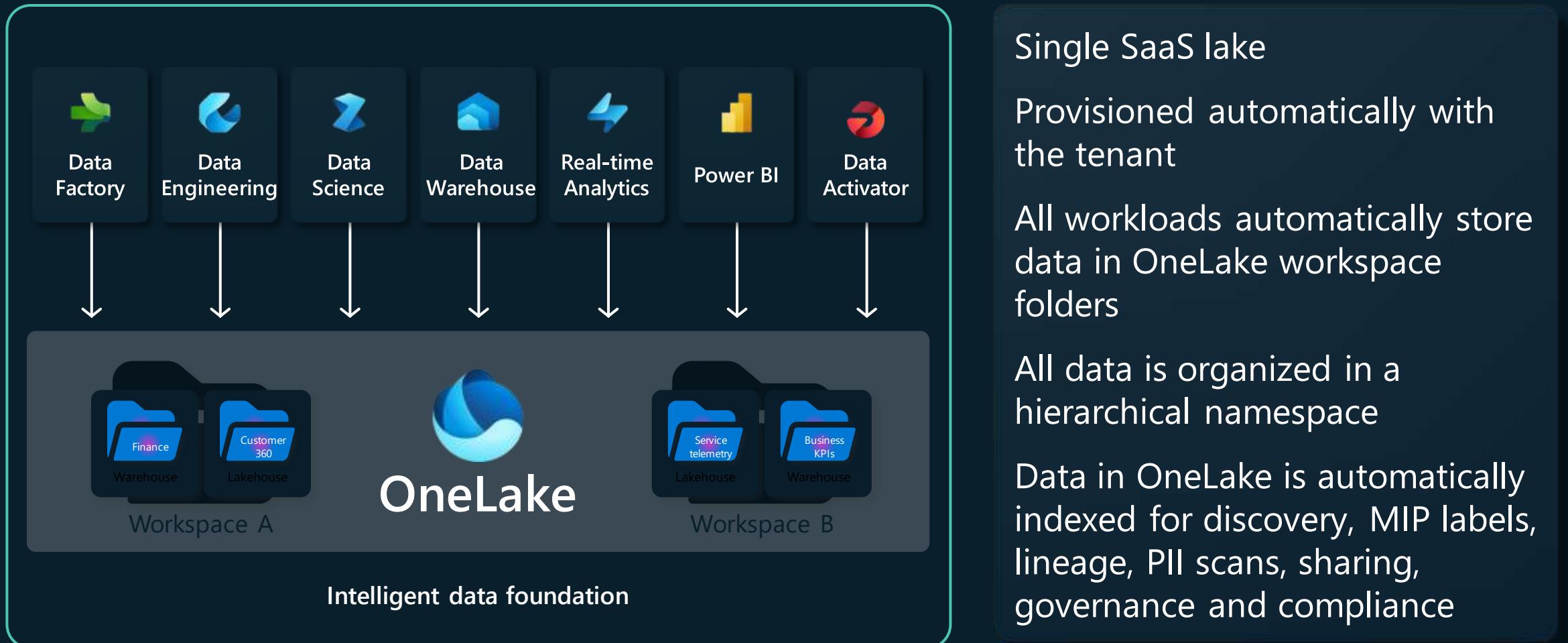
Plan and implement deployment solutions

Perform impact analysis of downstream dependencies from lakehouses, data warehouses, dataflows, and semantic models

Deploy and manage semantic models by using the XMLA endpoint

Create and update reusable assets, including Power BI template (.pbit) files, Power BI data source (.pbids) files, and shared semantic models

Understand Fabric architecture



Understand Fabric concepts

- Tenant
- Capacity
- Domain
- Workspace
- Items

Wide World Importers
Organizational tenant



Fabric admin tasks



Security and
access control



Data governance



Customization
and configuration



Monitoring and
optimization

Fabric admin tools



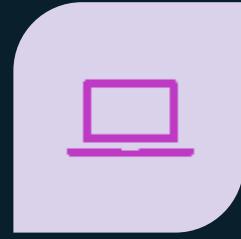
Fabric admin
portal



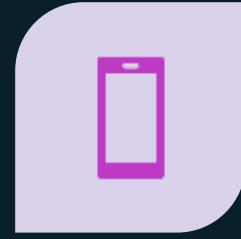
PowerShell
cmdlets



Admin APIs and
SDKs



Admin
monitoring
workspace



Fabric capacity
monitoring app

Fabric admin portal

Centrally manage, review, and apply settings for the entire tenant or by capacity

The screenshot shows the Microsoft Fabric Admin portal interface. On the left, a sidebar lists various management options: Tenant settings (New), Usage metrics, Users, Premium Per User, Audit logs, Domains (New), Capacity settings, Refresh summary, Embed Codes, Organizational visuals, Azure connections, Workspaces, Custom branding, Protection metrics, and Featured content. The Tenant settings option is highlighted with a grey bar. To the right, under the Microsoft Fabric section, there are several configuration items:

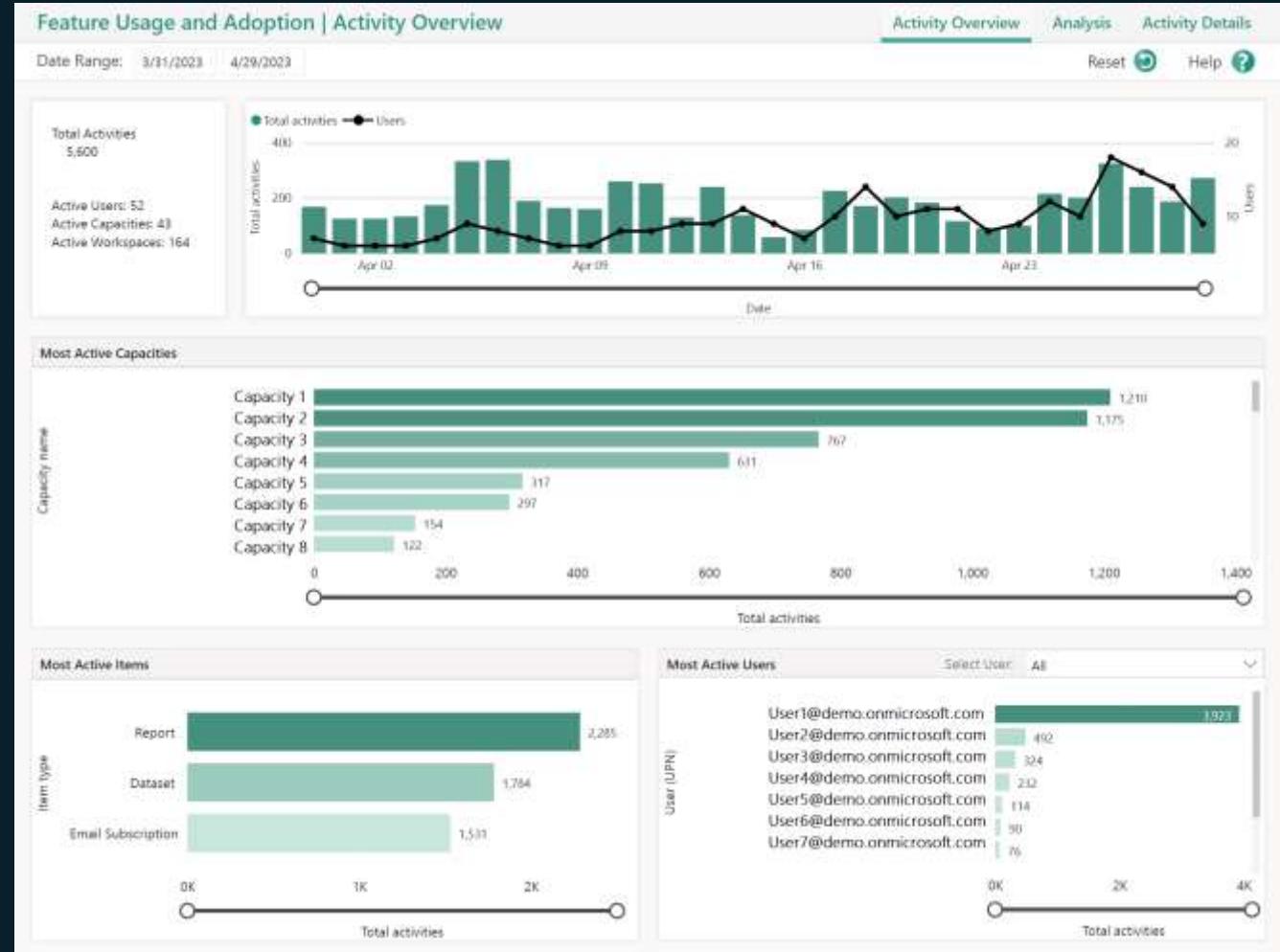
- There are new or updated tenant settings. Expand to review the changes.
- Data Activator (preview)**
Enabled for the entire organization
- Users can create Fabric items**
Enabled for the entire organization
- Help and support settings**
 - Publish "Get Help" information
Disabled for the entire organization
 - Receive email notifications for service outages or incidents
Disabled for the entire organization
 - Users can try Microsoft Fabric paid features
Enabled for the entire organization
 - Show a custom message before publishing reports
Disabled for the entire organization

A purple rectangular box highlights the "Users can create Fabric items" setting. A search bar at the top right says "Filter by keyword".

Fabric admin monitoring workspace

Monitor your environment including active:

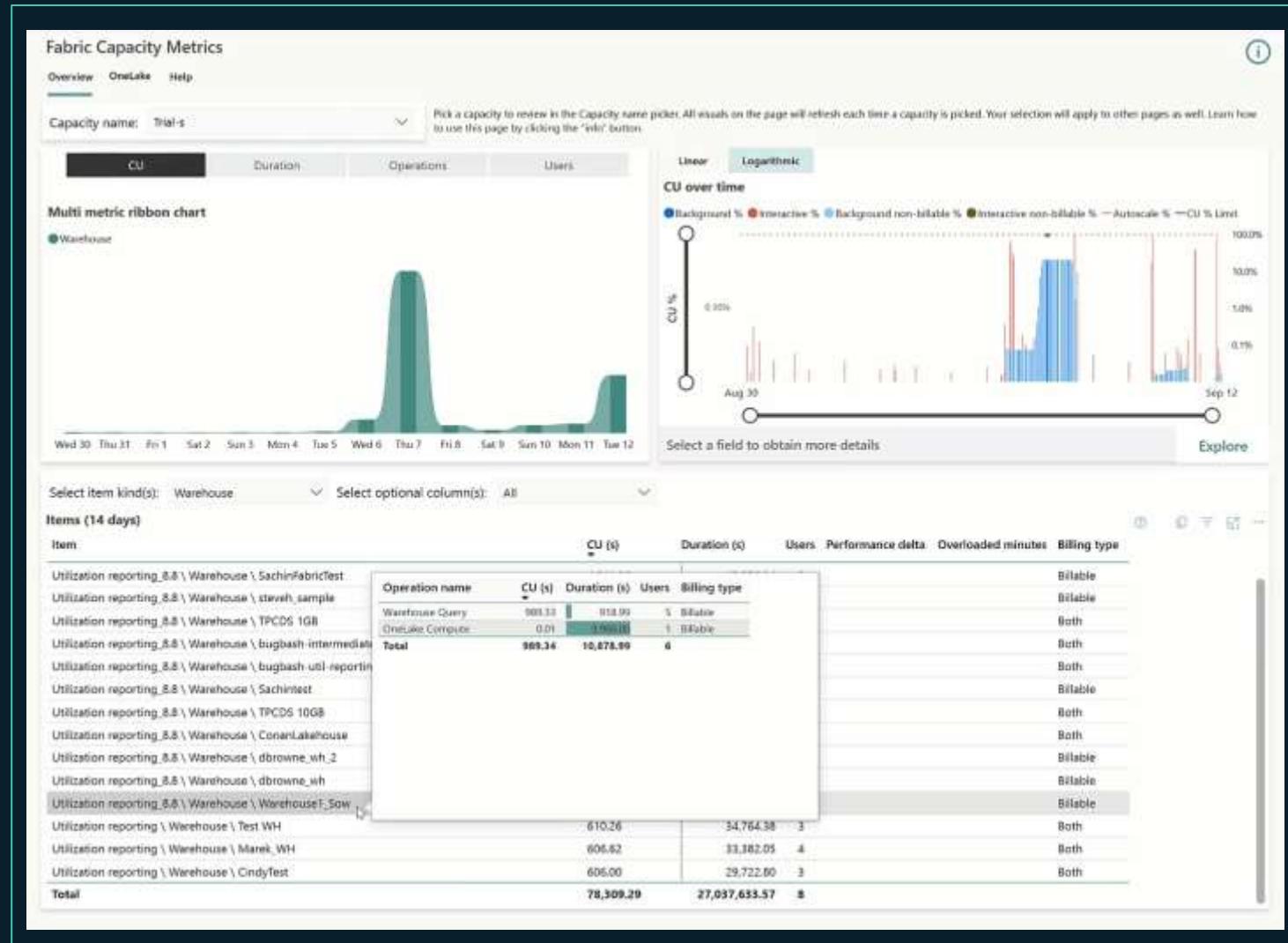
- capacities
- workspaces
- items
- activities
- users



Fabric capacity metrics app

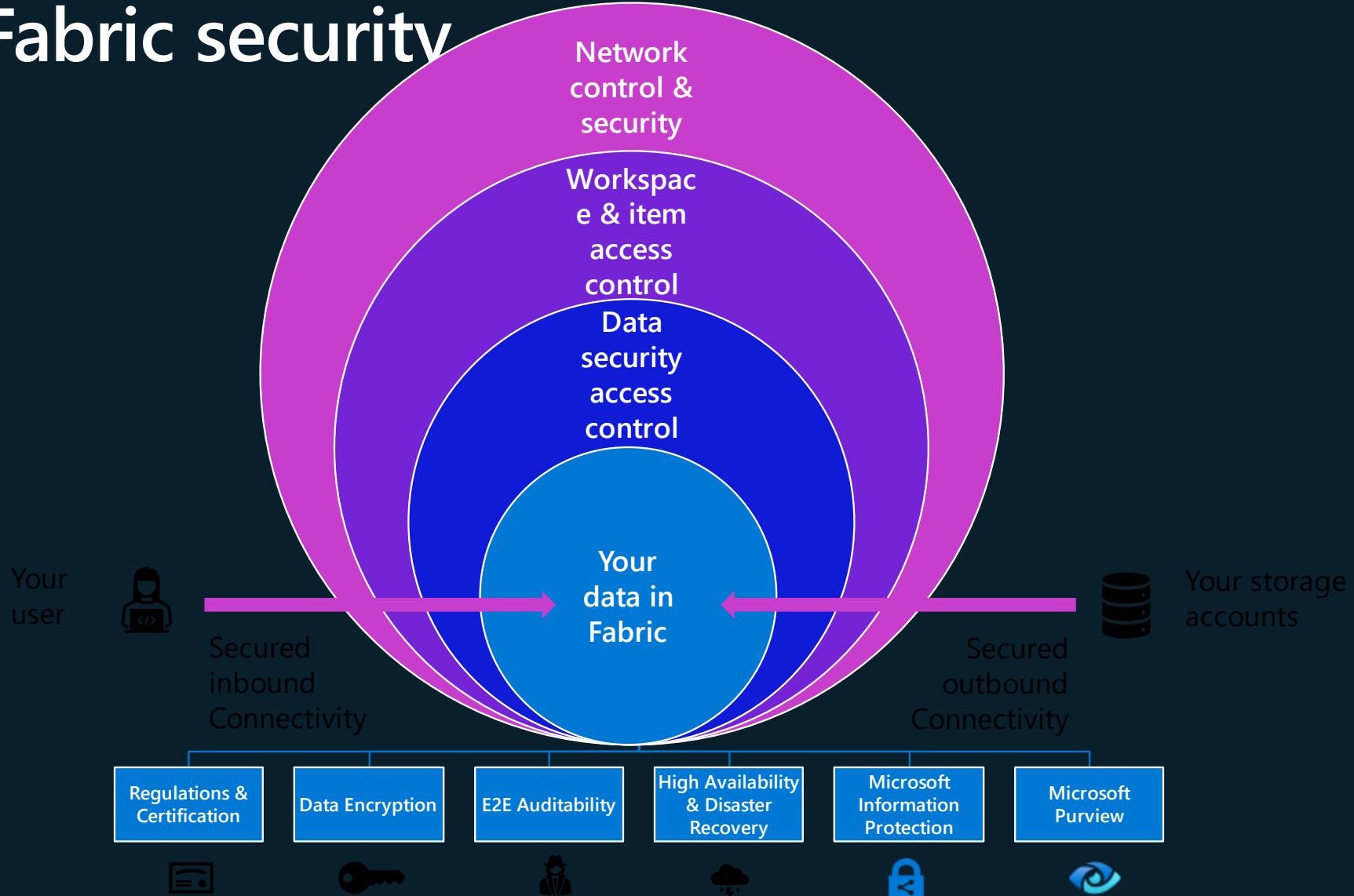
- Charts display capacity usage over a 14-day period

- Helps you understand which workloads use more capacity



Admin & workspace-level settings

Manage Fabric security



Manage users: assign and manage licenses

- Licenses control the level of access and functionality that users have within the Fabric environment.
- License management for Fabric is located in the M365 admin center.
- The next step beyond licenses is workspace management.

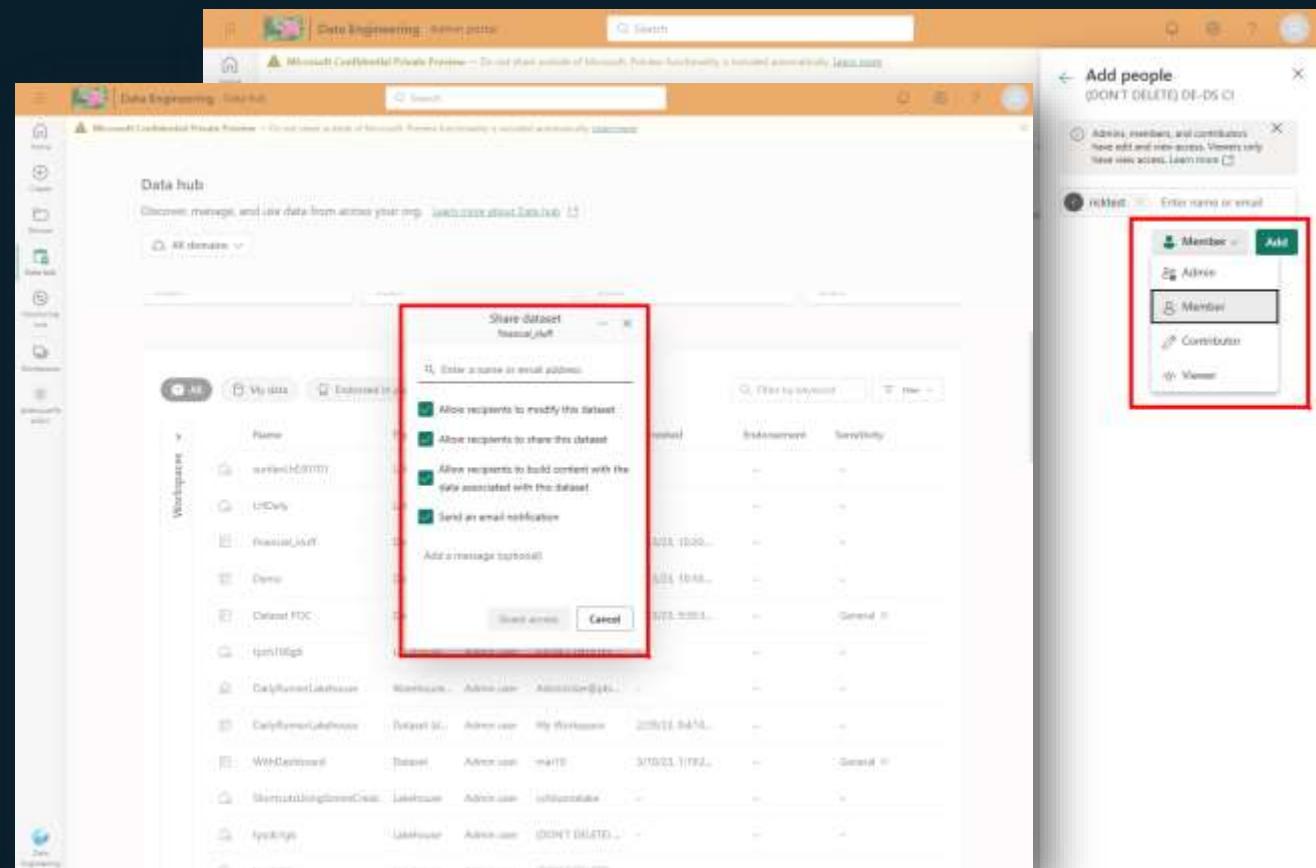
The screenshot shows the Microsoft 365 admin center interface. The left sidebar has a 'Licenses' section selected. The main content area displays a summary for 'Microsoft 365 E5 Developer (without Windows and A...' with 1 available license out of 25 total assigned. Below this, there's a list of users with their names, emails, and a checkbox column. One user, 'Alex Wilber', has a checked checkbox.

Name	Email
Alex Wilber	AlexW@beepboopbop.onmicrosoft.com
Angie Rudduck	angie@beepboopbop.onmicrosoft.com
Brian Moring	briandev@beepboopbop.onmicrosoft.com
Content Developer	contentdev@beepboopbop.onmicrosoft.com
David Ravetch	david@beepboopbop.onmicrosoft.com
Diego Siciliani	DiegoS@beepboopbop.onmicrosoft.com
Fabric User	Carlos@beepboopbop.onmicrosoft.com
Geoff Atkin	GeoffA@beepboopbop.onmicrosoft.com

Manage items and sharing

Admins can manage how users interact with Fabric in terms of sharing and distribution of content.

- Fabric workspace roles define default permissions on workload items
- Item permissions can be modified and managed via sharing
- Universal Security defines access policies on the delta tables directly and all workload compute engines will respect such policies



Workspace and item permissions

Workspaces

- Containers, store related items
- Roles grant workspace access
 - Admin
 - Contributor
 - Member
 - Viewer

Items

- Lakehouses, warehouses, semantic models, reports, dashboards, etc.
- Item-level access in workspace
 - SQL analytics endpoint access
 - Read
 - ReadData
 - ReadAll

Granular security permissions

User-specific permissions using T-SQL

- Object-level security
- Column-level security
- Row-level security
- Dynamic data masking

User access recommendations

- Limit workspace access to collaborators.
 - Use Viewer role and grant read on specific objects for read-only.
- Grant access to specific Fabric item(s), then on objects for limited use cases.
 - Use Microsoft Entra ID groups, when possible.

Govern data in Fabric

- Endorse Fabric content

The screenshot shows the Microsoft Fabric Services interface. At the top, there's a navigation bar with 'Services' (indicated by a crown icon), 'New', 'Upload', 'Create deployment pipeline', 'Create app', and a three-dot menu. To the right are search ('Filter by keyword'), filter ('Filter'), and sorting ('Sort') tools.

The main area is a table titled 'Services' with the following columns: Name, Type, Owner, Refreshed, Next refresh, Endorsement, Sensitivity, and Included in app. The table contains the following data:

Name	Type	Owner	Refreshed	Next refresh	Endorsement	Sensitivity	Included in app
Customer Service	Report	User 1	—	—	—	General ⓘ	<input checked="" type="checkbox"/> No
CustomerService_translated	Report	User 1	4/25/23, 9:22:14 AM	—	Promoted ⓘ	General ⓘ	<input checked="" type="checkbox"/> No
CustomerService_translated	Dataset	User 1	4/25/23, 9:22:14 AM	N/A	Promoted ⓘ	General ⓘ	<input type="checkbox"/> Yes
CustomerService_translated.pbix	Dashboard	User 1	—	—	—	—	<input checked="" type="checkbox"/> Yes

Below the table, a deployment pipeline diagram is shown. It consists of three boxes connected by arrows: 'Revenues Forecast' (Sharepoint, http://sharepoint.com), 'Segmentation Dataset' (Dataset, Refreshed: 4/26/23, 8:38:55 PM), and 'Finance Report' (Report). Each box has a 'refresh' button below it.

Fabric Capacities Introduction



Microsoft Fabric



Data
Factory



Synapse Data
Engineering



Synapse Data
Science



Synapse Data
Warehouse



Synapse Real
Time Analytics



Power BI



Data
Activator

AI Assisted

Shared Workspaces

Universal Compute Capacities

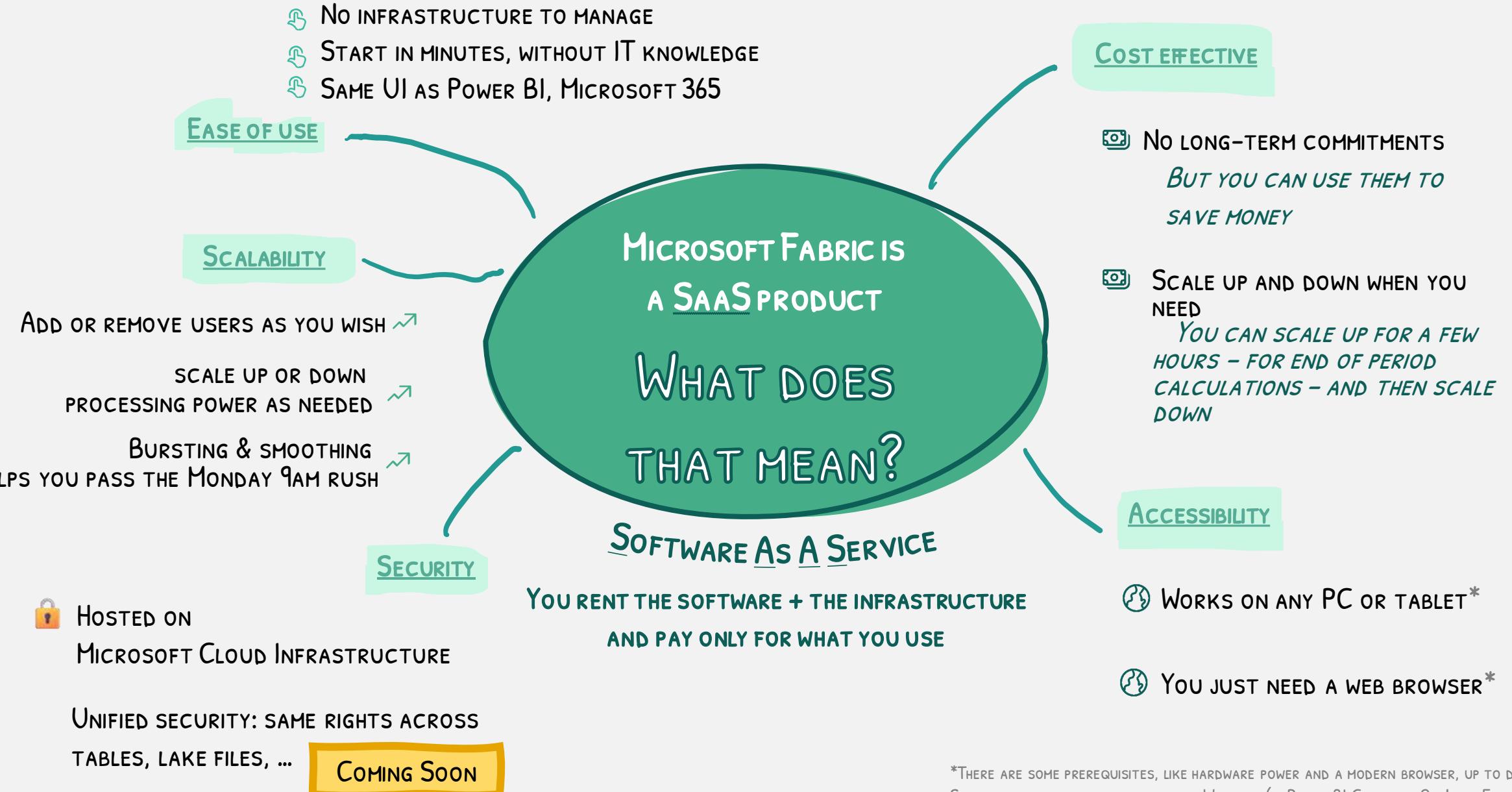
One Security

OneLake

Intelligent data foundation

Single...

- Onboarding and trials
- Sign-on
- Navigation model
- UX model
- Workspace organization
- Collaboration experience
- Data Lake
- Storage format
- Data copy for all engines
- Security model
- CI/CD
- Monitoring hub
- Governance & Capacity Metrics**
- Data Hub



Capacities are to Fabric what CPUs are to PCs

Personal Computing

When you purchase a PC you choose the number of CPU cores. The more CPU cores the more load the PC can handle.



The CPU cores are dynamically shared across all applications with no need to pre-allocate by app.



The total consumption of the CPU across all the apps cannot exceed the number of cores. CPU overload causes a slowdown.



Fabric Capacities

In Fabric, you provision a Capacity with a number of "capacity units". The more capacity units provisioned, the more compute load handled.

Unlike the PC, capacity units can be scaled up or down as needed.

The capacity units are dynamically shared across all the Fabric workloads, with no pre-allocation necessary.

A single capacity can simultaneously drive BI, DW, Spark, ML and every other compute engine in Fabric

The total consumption of the capacity across all the workloads cannot exceed the capacity units provisioned.

Overloading the capacity will throttle it (slow down).

Auto scale can dynamically increase the available compute units avoiding the slowdown.

Capacities are a shared resource

Shared across workloads

A single capacity is providing the compute power for all Fabric workloads.

There is no need to allocate compute for each workload separately.



Shared Across Projects

A single capacity typically supports dozens of separate projects simultaneously, each managed in its own workspace.

It is rare to have a capacity dedicated to a single project



Shared across users

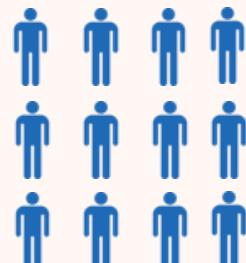
For each project, many developers will share a workspace where collaborative development and consumption at scale is managed.

Each creator can provision any artifact and run any job without the need for any pre-approval or planning

Developers/Creators



Consumers



Capacities are a SaaS resource

Always on auto-pilot

Capacities are a self-managed system that alleviates day-to-day or hour-to-hour resource management by capacity administrators

Providing complete transparency into the usage of the capacities at any grain: projects/workspaces, artifacts, queries, users and jobs

Eliminate workload management

Large jobs can be scheduled at any time without impacting any other jobs running

Protection against a single user hogging all the resources by carelessly running a very large query

Smoothing automatically prevents spiky loads from creating temporal overload of the system – thereby you are purchasing for average load vs. max load.

Always peak performance

Bursting allows all jobs and queries to always run at peak performance regardless of any other compute activity happening on the capacity

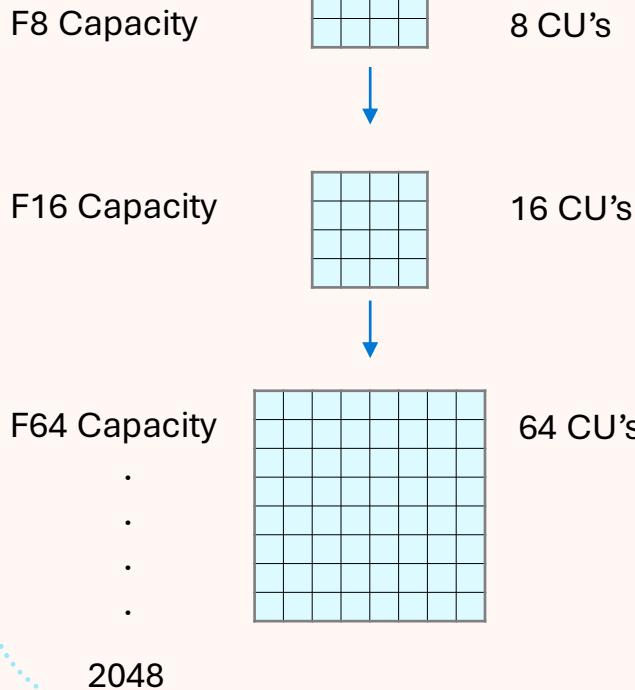
Auto scale can automatically adjust the compute units provisioned to the capacity

Capacities are flexible building blocks for growth

Capacities can be configured in endless ways to meet scale, usage and governance requirements while tuning to minimize TCO and performance goals

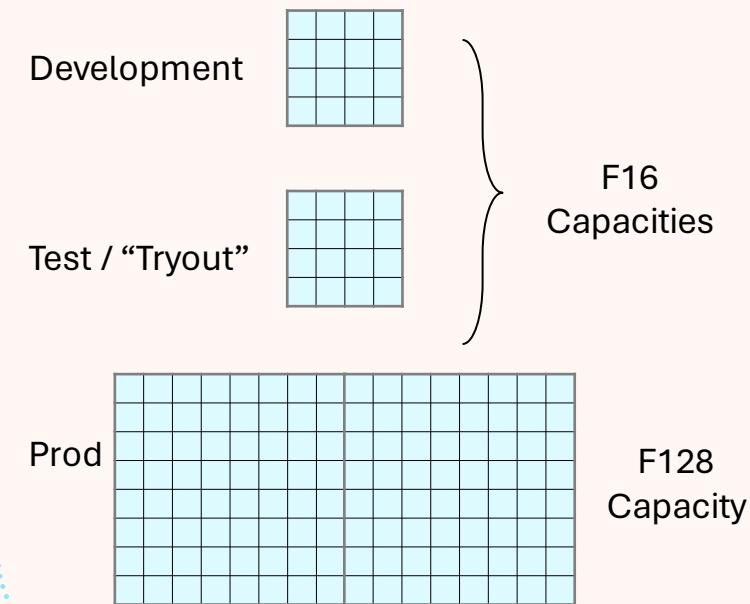
Scale Vertically

Increased capacity size provides more throughput



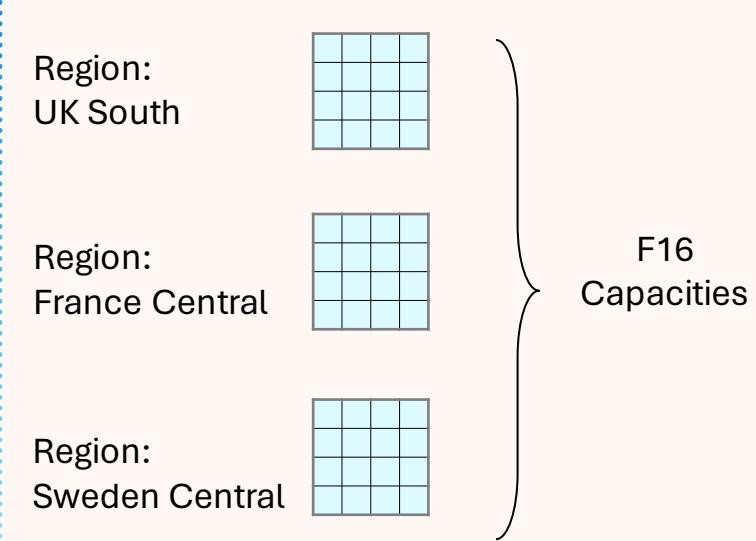
Scale Horizontally

Scale horizontally using the benefits of modular design for hardened isolation and governance



Regional Availability

Use different capacities for different regions to support GDPR / Data residency requirements



Provisioning and Deploying Capacities

Purchased in Azure

- **Purchased** either as a PAYG or RI resource
- **Provisioned with a certain amount of compute** units, analogous to CPU cores.
- The **more capacity units are provisioned, the more load** the capacity can support
 - Multiply SKU size by 30s to match platform evaluation in metrics app
- Capacities are **priced at a fixed hourly rate**, based on capacity units provisioned
- The RI commitment (1-year reserved instance) enjoys a **40% discount**

Universal Compute Capacities SKU Sizing

SKU	Capacity Units (CU)	CU's (per 30s)	Power BI SKU	Power BI V-cores
F2	2	60	-	0.25
F4	4	120	-	0.5
F8	8	240	A1	1
F16	16	480	A2	2
F32	32	960	A3	4
F64	64	1920	P1	8
F128	128	3840	P2	16
F256	256	7680	P3	32
F512	512	15360	P4	64
F1024	1024	30720	P5	128
F2048	2048	61440	-	256

Provisioning and Deploying Capacities

Deployed to Regions

- Each capacity **resides in a specific region of the buyers' choice** where both the data & compute reside
- **Workspaces are assigned to a capacity** that provides the compute and storage for all the workspace artifacts
- Multiple capacities can be purchased, deployed and managed by **different owners** residing in a single tenant allowing each business unit to pay for their own consumption

Tenant (Microsoft)

Capacity 1 (West Europe)

Workspace 1
(Data Science Team)

Item 1
Power BI Dataset

Item 2
AI Function
Evaluation

Workspace 2
(Research)

Item 3
Spark Notebook

Item 4
ML Dataflow

Capacity 2 (Central India)

Capacity 3 (West US)

Smoothing intro and benefits

Load stabilization

Smoothing helps capacities self-stabilize by flattening large spiky loads into a smooth load profile, eliminating temporal spikes

Eliminates Scheduling contention

Large/scheduled Jobs usage (not execution) are smoothed over 24 hours, eliminating the need to decide the timing and order of job execution

Performance now, pay later. Demo to follow.

Bad actor protection

Interactive operations smoothed over several minutes, preventing a single user with a very demanding query from hogging the entire capacity



What is Bursting?

Job acceleration

Bursting provides extra compute resources to jobs and queries to accelerate their completion

Go beyond

The extra resources of bursting allow jobs to **utilize far more resources than “face value”**

Instead of running a job on 64 CU and completing in 60 seconds, bursting could use 256 CUs to complete the job in 15 seconds.

Same amount of work, just completed faster

No hassle, No overload

Bursting is automatic when the system reasons it can accelerate the job by applying extra resources. No settings are required.

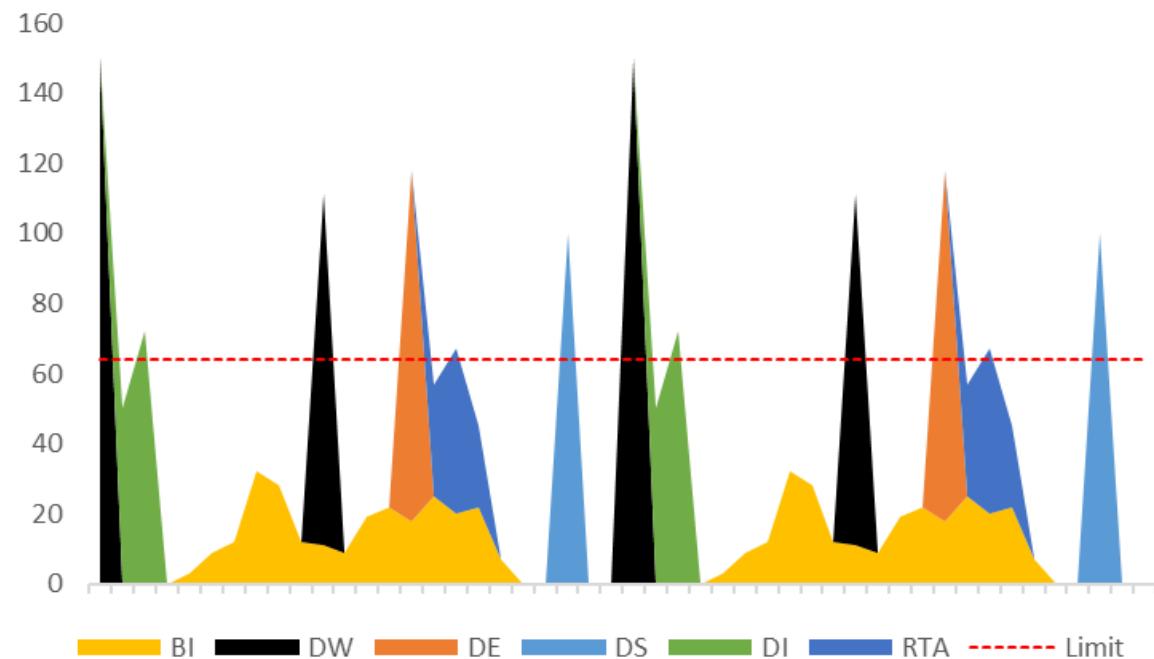
Bursting never causes an overload as the *smoothing* mechanism will always flatten the resource burst

Bursting and smoothing | before and after

Looking at an example of a 64 CU capacity, running multiple workloads over a couple of days...

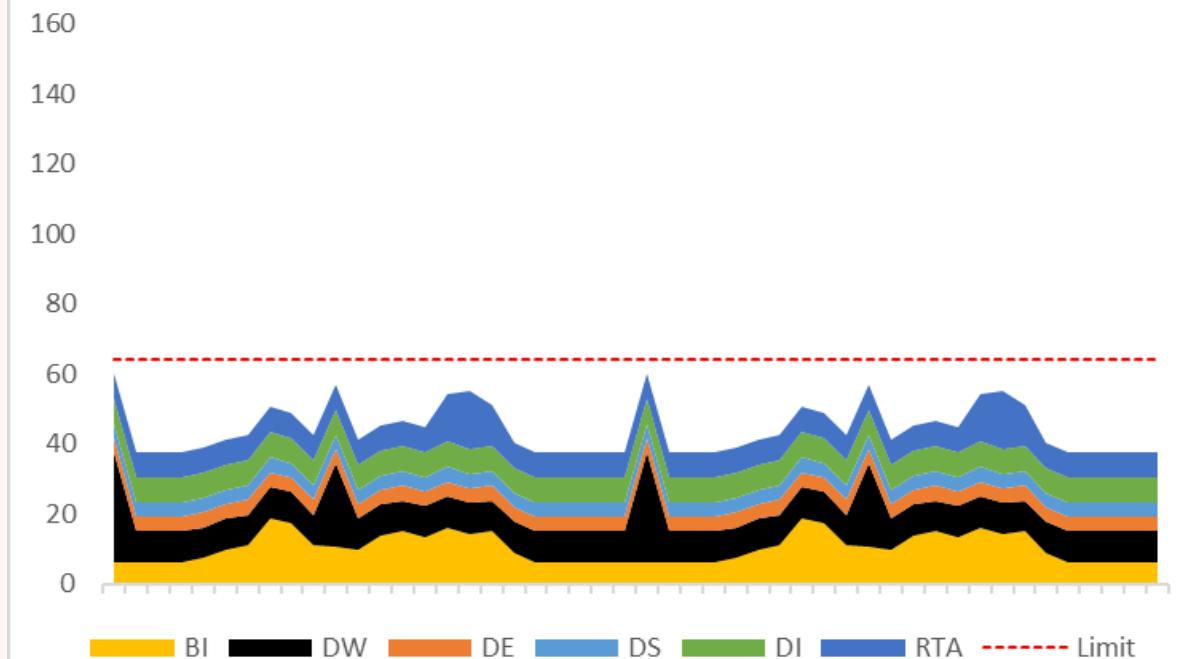
Before Smoothing

- Actual load as executed on the capacity before smoothing
- *Bursting* accelerates jobs execution by resource boosting
- The capacity could be overloaded 25% of the time
- Some of the overloads are more than 2x the limit
- There are periods of no/low usage

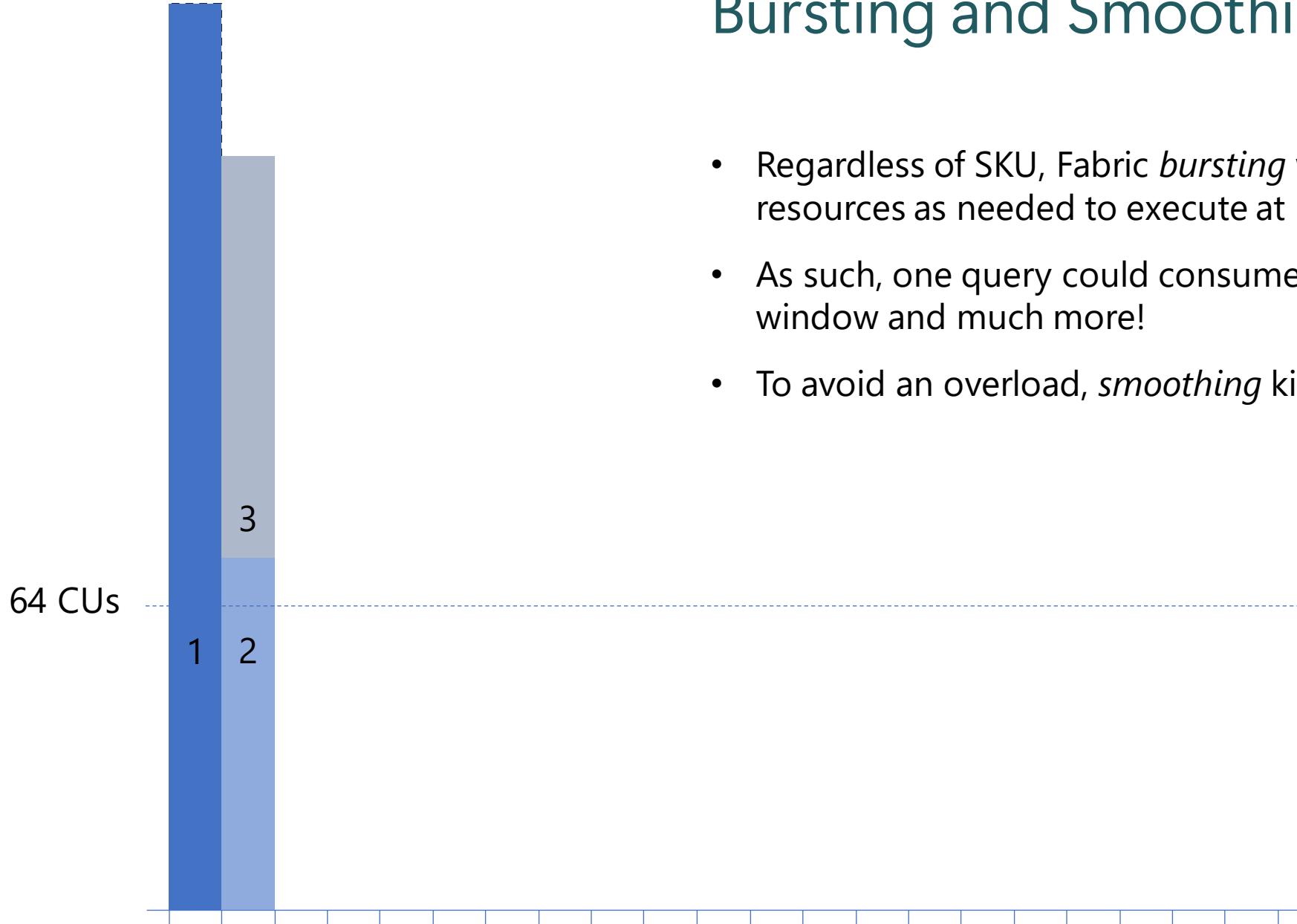


After Smoothing

- Shows the reported load (not runtime execution) against the capacity limits
- There is NO overload, and consumption is more stable
- The smoothing of usage fills in gaps



Bursting and Smoothing



- Regardless of SKU, Fabric *bursting* will automatically allocate resources as needed to execute at maximum performance
- As such, one query could consume all the quota of a single time window and much more!
- To avoid an overload, *smoothing* kicks in

Bursting and Smoothing

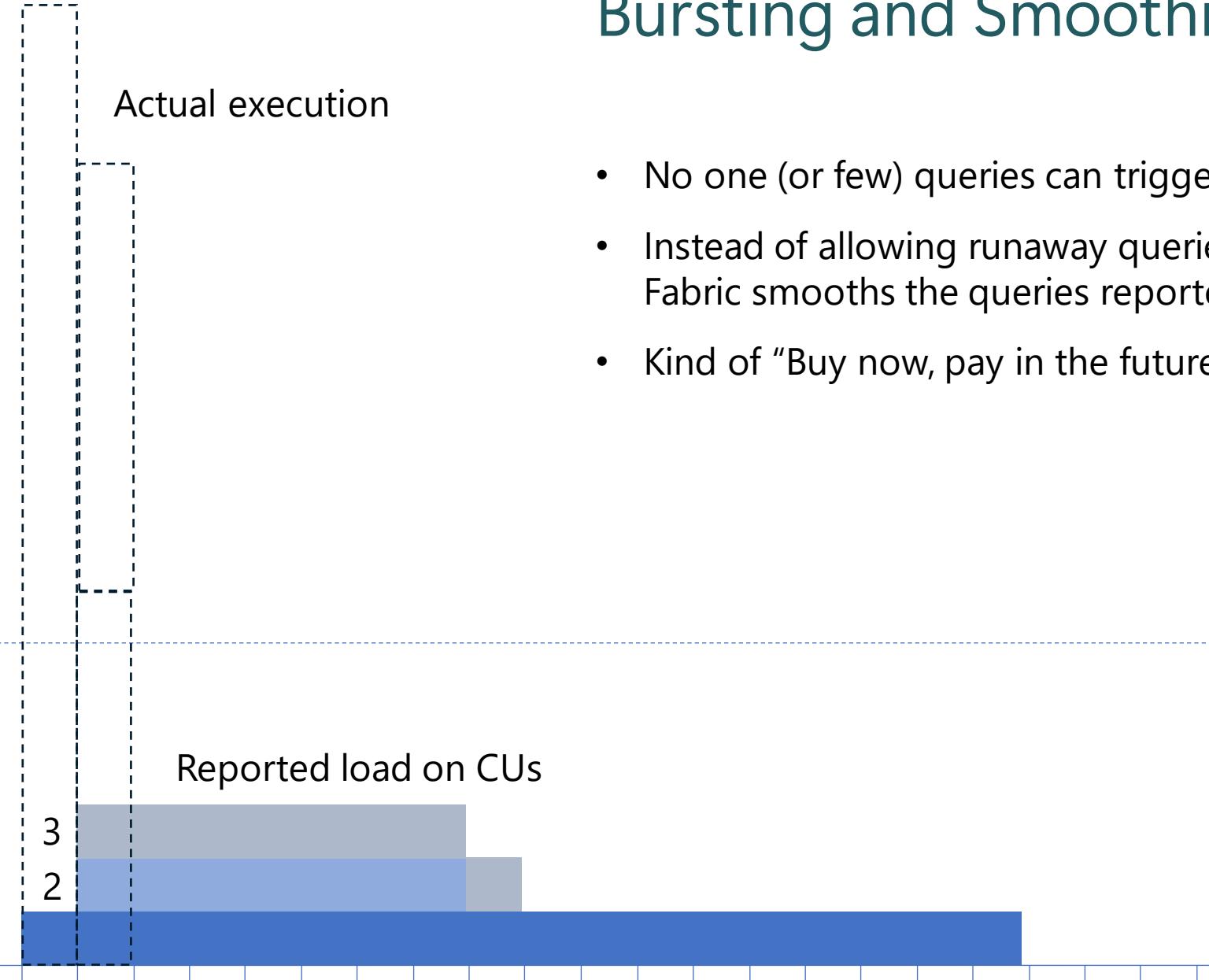
Actual execution

- No one (or few) queries can trigger an overload
- Instead of allowing runaway queries to create a local overload, Fabric smooths the queries reported usage to future time windows
- Kind of “Buy now, pay in the future” installment plan

64 CUs

Reported load on CUs

1
2
3



Bursting and Smoothing

4 – large batch job

64 CUs

1
2
3

- Large batch processes traditionally were a threat to interactive queries as they would overload the compute resource
- DBAs traditionally had to carefully schedule these jobs to off-hours to avoid interference with interactive user experiences

Bursting and Smoothing

4 – large batch job

Actual execution

64 CUs



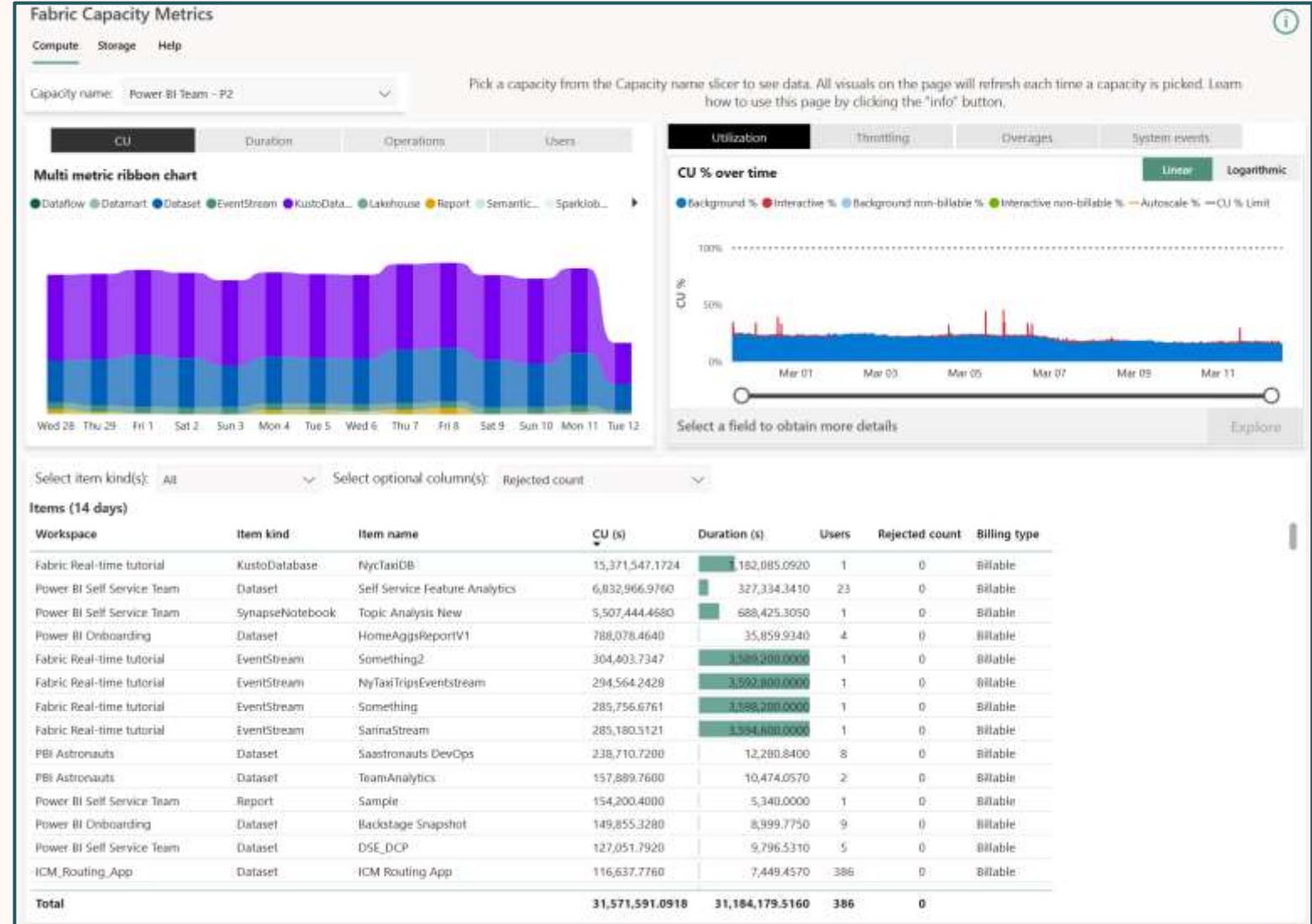
- A similar “installment plan” logic is applied for batch jobs
- But for batch jobs the smoothing is applied uniformly for the next 24 hours
- This completely liberates the DBA from any consideration of job scheduling. The load will be uniform regardless of the schedule.
- Most importantly, regardless of when batch jobs are scheduled, there will not be any degradation on interactive query performance

Monitoring with Capacity Metrics

Capacity Metrics

Monitor Capacities and Plan capacity scale-up with confidence

- Tenant wide visibility into capacity usage for all Fabric experiences
- Identify resource usage trends and their impact to autoscale & throttling
- View preview workload usage alongside production workloads to make data-driven capacity sizing decisions

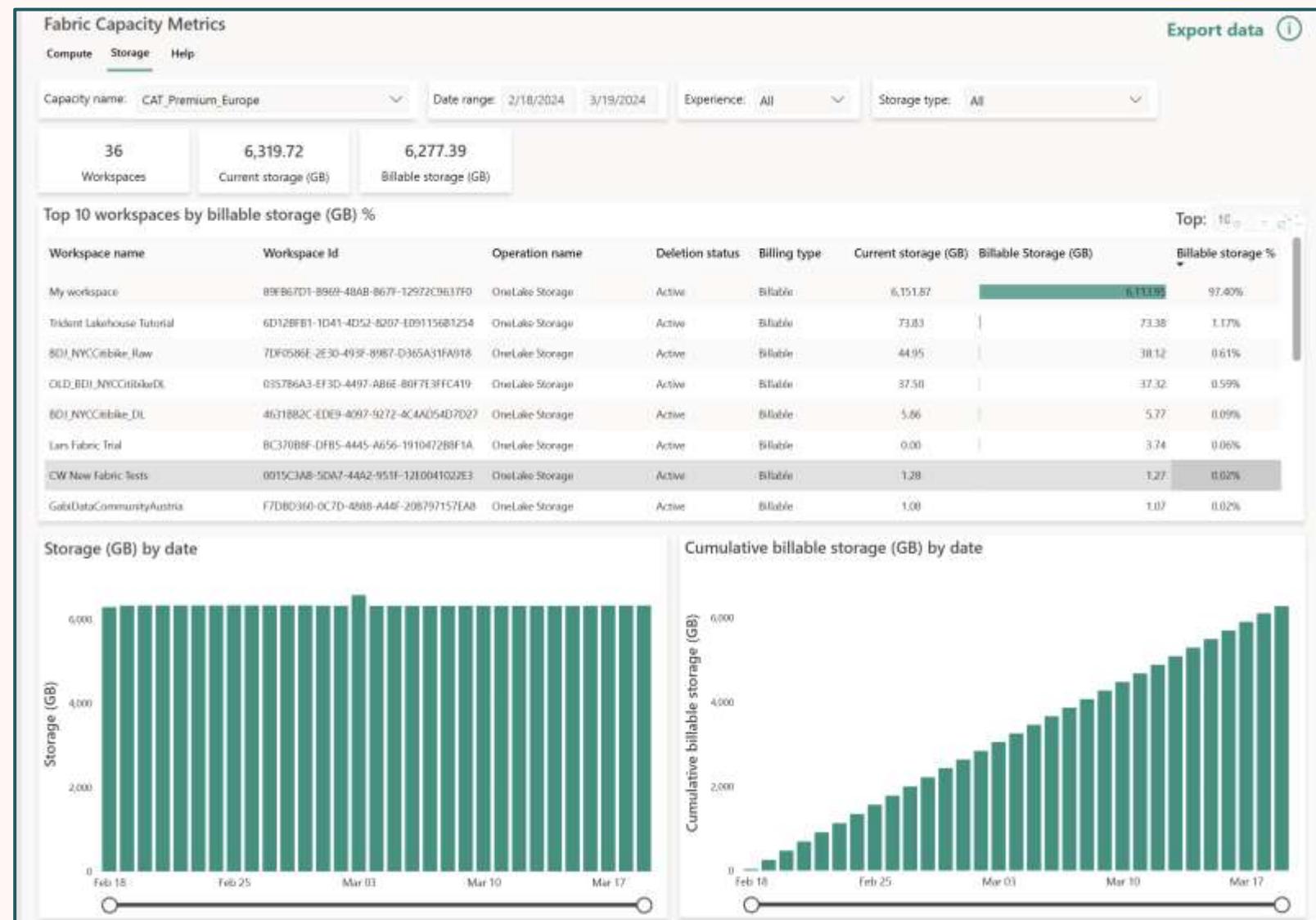


Capacity Metrics

Monitor OneDrive consumption

Measure the trends of workspace storage consumption against capacity limits, by day or hour

Reconcile costs with internal chargeback processes

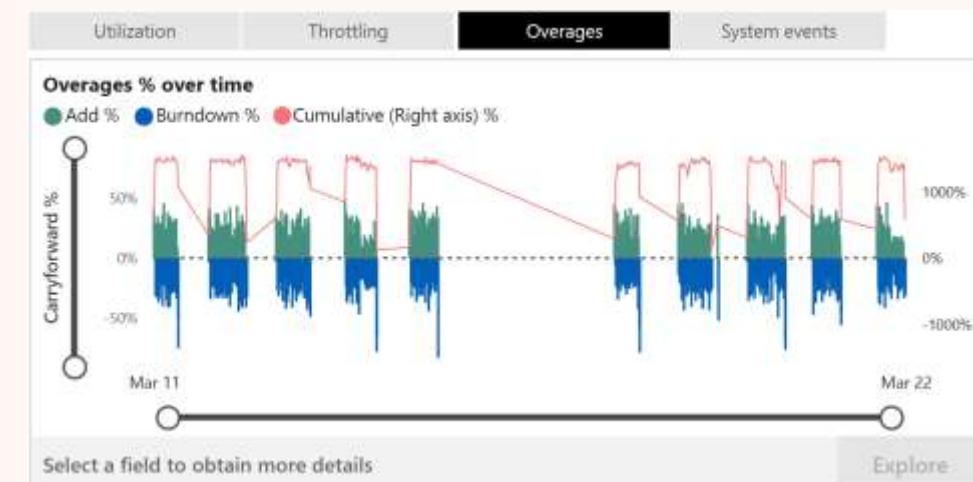
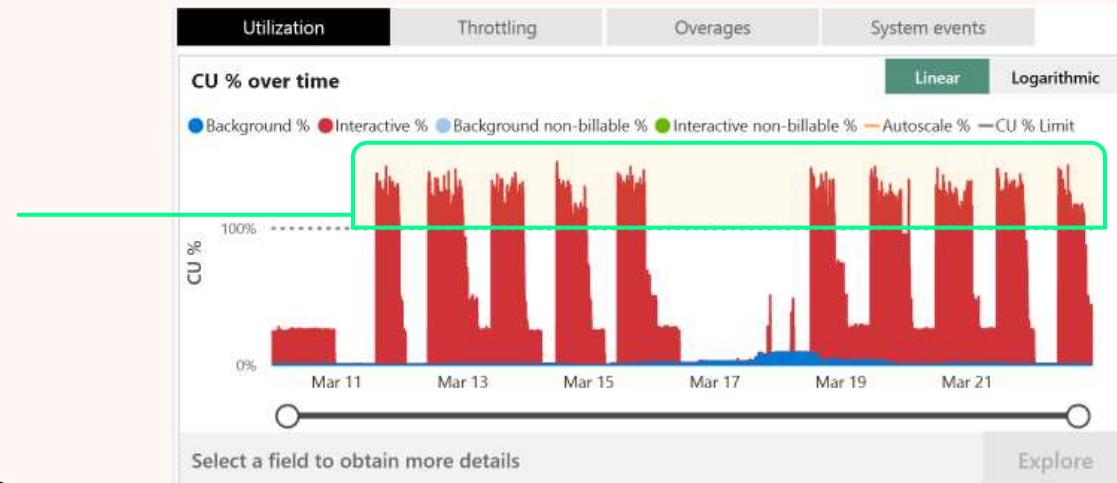


Capacity Throttling Policies

Throttling intro

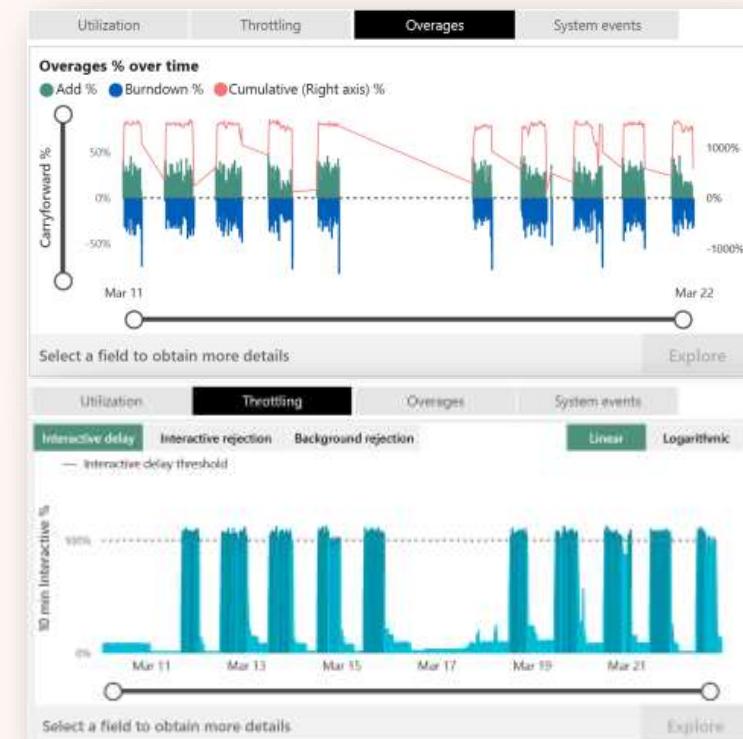
- Throttling is the platform policy for managing consumption that exceeds throughput is provided by SKU choice
- When workloads exceed the throughput of a capacity a cumulative debt is tracked to be burned down
- Cumulative debt is used to determine throttling policies and is burned down when resources are free

Overage Operation	Description
Overages - Added	<ul style="list-style-type: none">• Timepoint when job requests exceed the throughput of a capacity, overages are added to the cumulative buffer to burn down.• This graph simplifies identification of the optimal timepoint to load timepoint drill to analyze the user operations that contributed to an overage.
Overages - Burndown	<ul style="list-style-type: none">• Overages being reconciled when future capacity is free to burn down
Overages - Cumulative	<ul style="list-style-type: none">• The total amount of queued work on the capacity to be burned down in the future when the capacity is not fully utilized



Capacity throttling evolution for Fabric

- For Fabric, throttling policies were refined to deliver multiple benefits
 - Reduced throttling** for capacities that only experience occasional spikes
 - Added overage protection** – rejection policies prevent overloaded capacities from irrecoverable overload
 - Optimizations for long-running jobs:** We're optimizing the platform for long-running jobs, so if a job exceeds capacity limits, it will run to completion and the overage will be burned down against future capacity

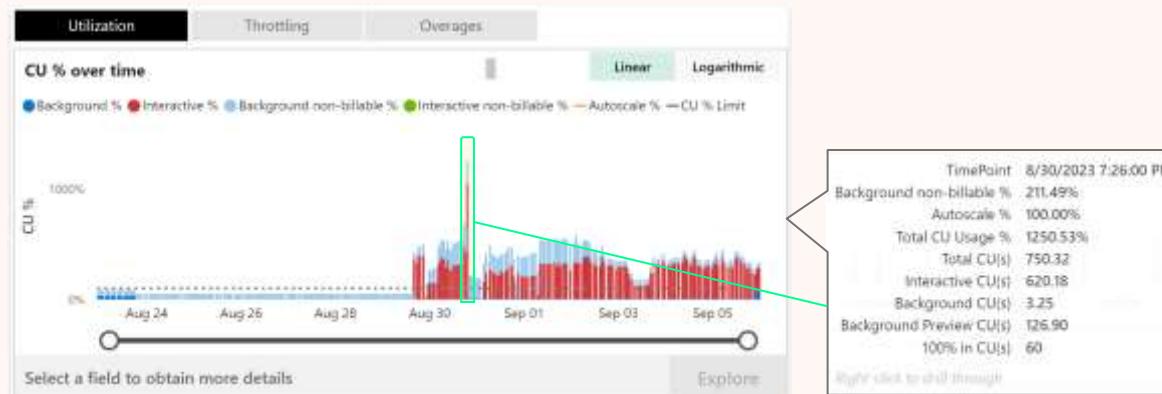


Smoothed Capacity - Future Use	Platform Policy	Customer Impact
<= 10m	Overage Protection	Jobs can consume 10 minutes of future capacity use without throttling
> 10m → <= 60m	Interactive Delay	User requested interactive type jobs will be throttled
> 60m → <= 24h	Interactive Rejection	User requested interactive type jobs will be rejected
> 24h	Background Rejection	User Scheduled background jobs will be rejected from execution

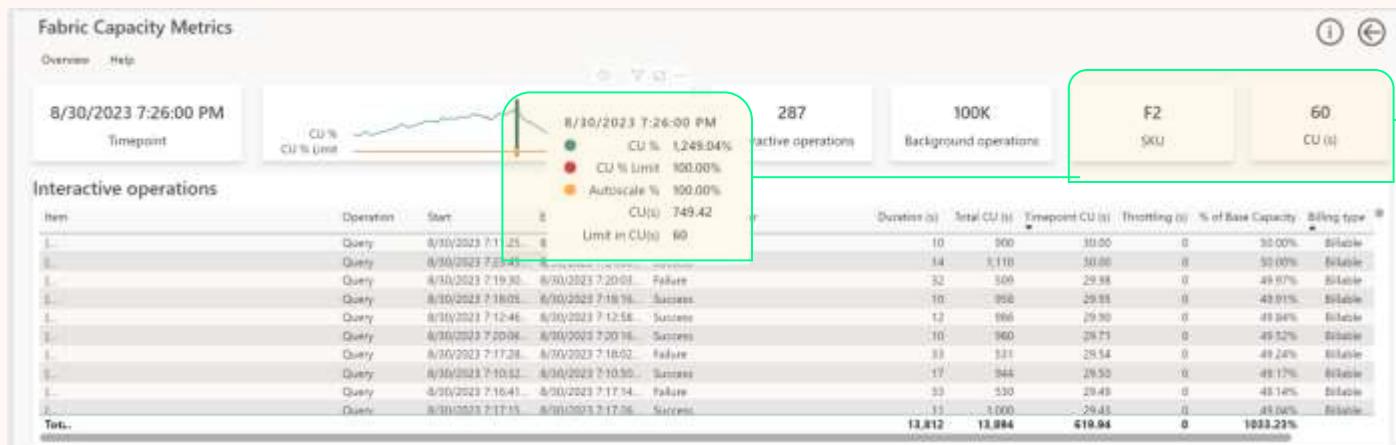
Capacity Planning with Capacity Metrics

Capacity planning case study - measurement

Start with a test or trial capacity to evaluate the load of specific Fabric Experiences i.e., Power BI Datasets, Spark Notebooks or a Datawarehouse



If usage is above the current capacity limits , choose the desired utilization rate to accommodate via capacity scale up



Load Capacity Metrics timepoint drill to analyze :

- Total CU's consumed : **749 CU(s)**
- Capacity Size : (F2)
- CU(s) available on your capacity : 60 CU(s)

Capacity planning case study – SKU selection

Universal Compute Capacities SKU Sizing

To accommodate a **749 CU(s)** load the admin can purchase an F32 capacity providing 960 CU(s) of throughput

SKU	Capacity Units (CU)	CU's (per 30s)	Power BI SKU	Power BI V-cores
F2	2	60	-	0.25
F4	4	120	-	0.5
F8	8	240	A1	1
F16	16	480	A2	2
F32	32	960	A3	4
F64	64	1920	P1	8
F128	128	3840	P2	16
F256	256	7680	P3	32
F512	512	15360	P4	64
F1024	1024	30720	P5	128
F2048	2048	61440	-	256

Pausing and Resuming Capacities



Introduction to Pausing and Resuming Capacities

Overview and Benefits

Pause and Resume lets you manage compute costs on F SKU capacities by suspending the execution of all workloads running on the capacity

- When a capacity administrator pauses a capacity:

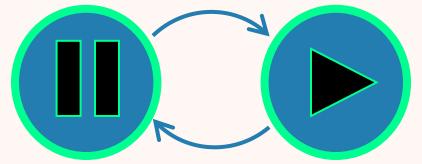
Workloads stop execution

New requests are not run

Smoothed usage will be reconciled

(details in the next demo)

Note: OneLake storage will remain active and billable while a capacity is paused



Bursting and Smoothing

Jobs Executed

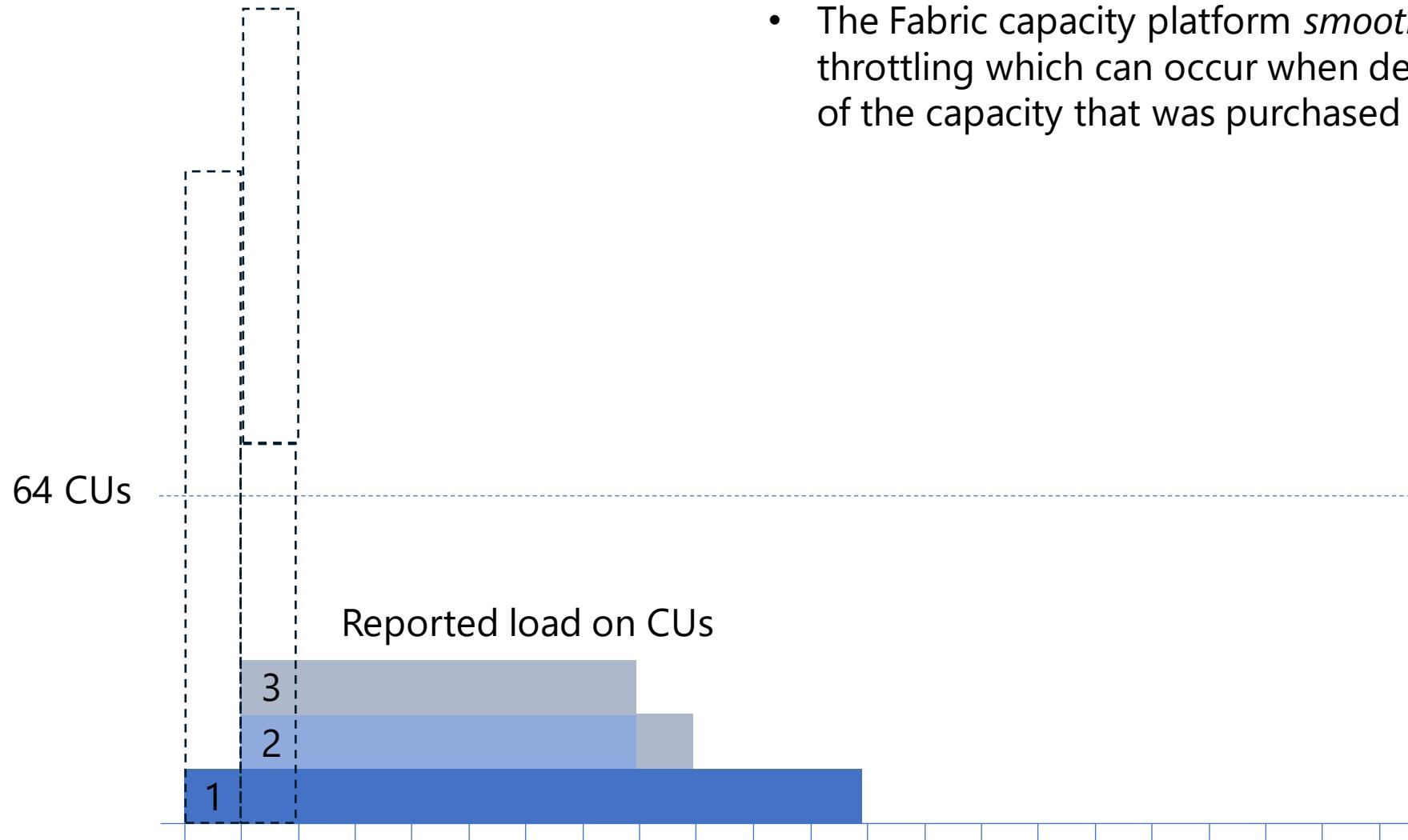


- Job execution in Fabric workloads happens on-demand via capacity powered compute engines
- Fabric *bursting* will automatically allocate resources as needed to execute at maximum performance

Actual
execution

Bursting and Smoothing

- The Fabric capacity platform *smooths* usage out to reduce throttling which can occur when demand exceeds the throughput of the capacity that was purchased



Smoothing and Paused Capacities

Pause event on Capacity

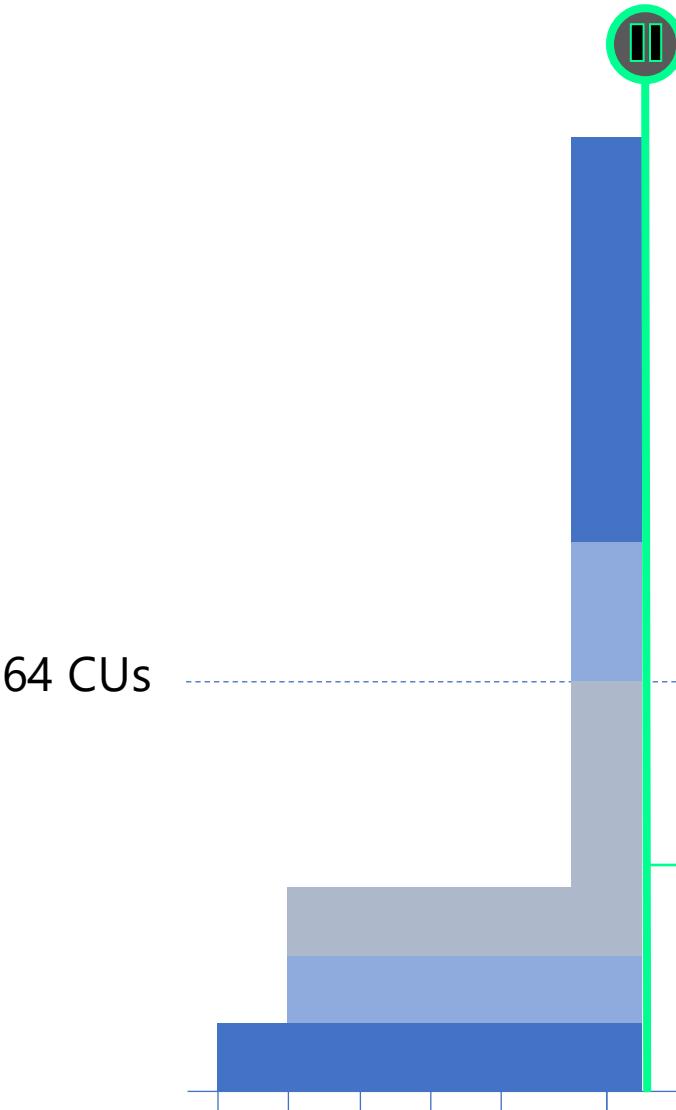


64 CUs

- When a capacity is paused...

Smoothing and Paused Capacities

Pause event on Capacity



- When a capacity is paused...
- Usage that was smoothed into the future will be “reconciled” and charged against the capacity at the timestamp the capacity was paused
- Reconciled usage will show up as a spike in capacity metrics



Smoothing and Paused Capacities

Pause event on Capacity



64 CUs

- When a capacity is paused...
- Usage that was smoothed into the future will be “reconciled” and charged against the capacity at the timestamp the capacity was paused
- Pause events can be viewed in the new System events tab

Utilization	Throttling	Overages	System events		
System events					
State transition time	Capacity state		Capacity state change reason		
12/13/2023 9:12:14 AM	Active		Created		
12/13/2023 9:29:12 AM	Suspended		ManuallyPaused		
12/13/2023 9:30:15 AM	Active		ManuallyResumed		
12/13/2023 9:33:29 AM	Suspended		ManuallyPaused		
12/13/2023 9:34:58 AM	Active		ManuallyResumed		
Select a field to obtain more details					
Explore					

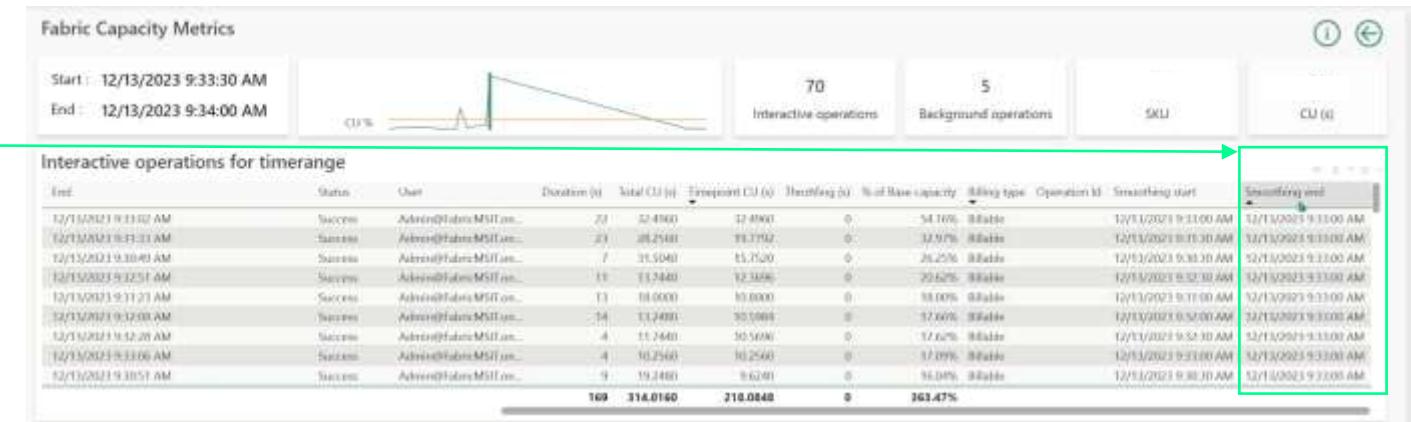
Smoothing and Paused Capacities

Pause event on Capacity



64 CUs

- When a capacity is paused...
- Usage that was smoothed into the future will be “reconciled” and charged against the capacity at the timestamp the capacity was paused
- Pause events timestamp is shown in the smoothing end field in timepoint drill views



Git integration and Deployment Pipelines

Git integration

- Git integration enables source control capabilities for developers to integrate their development processes, tools, and best practices directly into the Fabric platform
- Sync workspaces with Azure DevOps Git directly through Microsoft Fabric

Scenarios enabled by Git

- Backup and version development work
- Revert to previous stages
- Collaborate with others or work alone using git branches
- Apply the capabilities of familiar source control tools to manage Fabric items
 - Commit changes
 - Sync changes from Git
 - Manage conflict resolutions
 - ...and more!!



SQL Database Projects

Extension to design, edit, and publish schemas from a source-controlled environment

- Local representation of SQL objects that comprise the schema for a single database such as tables, stored procedures, or functions
- Enables T-SQL analytics engineers with capabilities including
 - **Source code integration** – Stored in control systems like Git to keep track of changes and perform code reviews easily
 - **Database testing** – Support the addition of “unit test” to validation functionality of the database
 - **Schema validation** – “Build” operation can be used to compile and validate T-SQL code in the project



Deployment Pipelines

What are deployment pipelines?

- Deployment pipelines in Fabric provide a tool to manage the life cycle of organizational content
- Enables creators to develop and test content in the service before it reaches end-users in a production environment

What scenarios do deployment pipelines enable?

- Clone content from one stage in the development pipeline to another – typically from dev to UAT/test to production
- Review and compare differences in schema/item definitions between stages

How can I automate deployments?

- Use deployment pipelines REST APIs to deploy content programmatically between stages
- Use new or existing Azure DevOps pipelines in conjunction with APIs



Power BI Projects, themes & DevOps

2

Module 2 Prepare and serve data



Prepare and serve data (40-45%)

Create objects in a lakehouse or warehouse

Ingest data by using a data pipeline, dataflow, or notebook

Create and manage shortcuts

Implement file partitioning for analytics workloads in a lakehouse

Create views, functions, and stored procedures

Enrich data by adding new columns or tables

Copy data

Choose an appropriate method for copying data from a Fabric data source to a lakehouse or warehouse

Copy data by using a data pipeline, dataflow, or notebook

Add stored procedures, notebooks, and dataflows to a data pipeline

Schedule data pipelines

Schedule dataflows and notebooks

Transform data

Implement a data cleansing process

Implement a star schema for a lakehouse or warehouse, including Type 1 and Type 2 slowly changing dimensions

Implement bridge tables for a lakehouse or a warehouse

Denormalize data

Aggregate or de-aggregate data

Merge or join data

Identify and resolve duplicate data, missing data, or null values

Convert data types by using SQL or PySpark

Filter data

Optimize performance

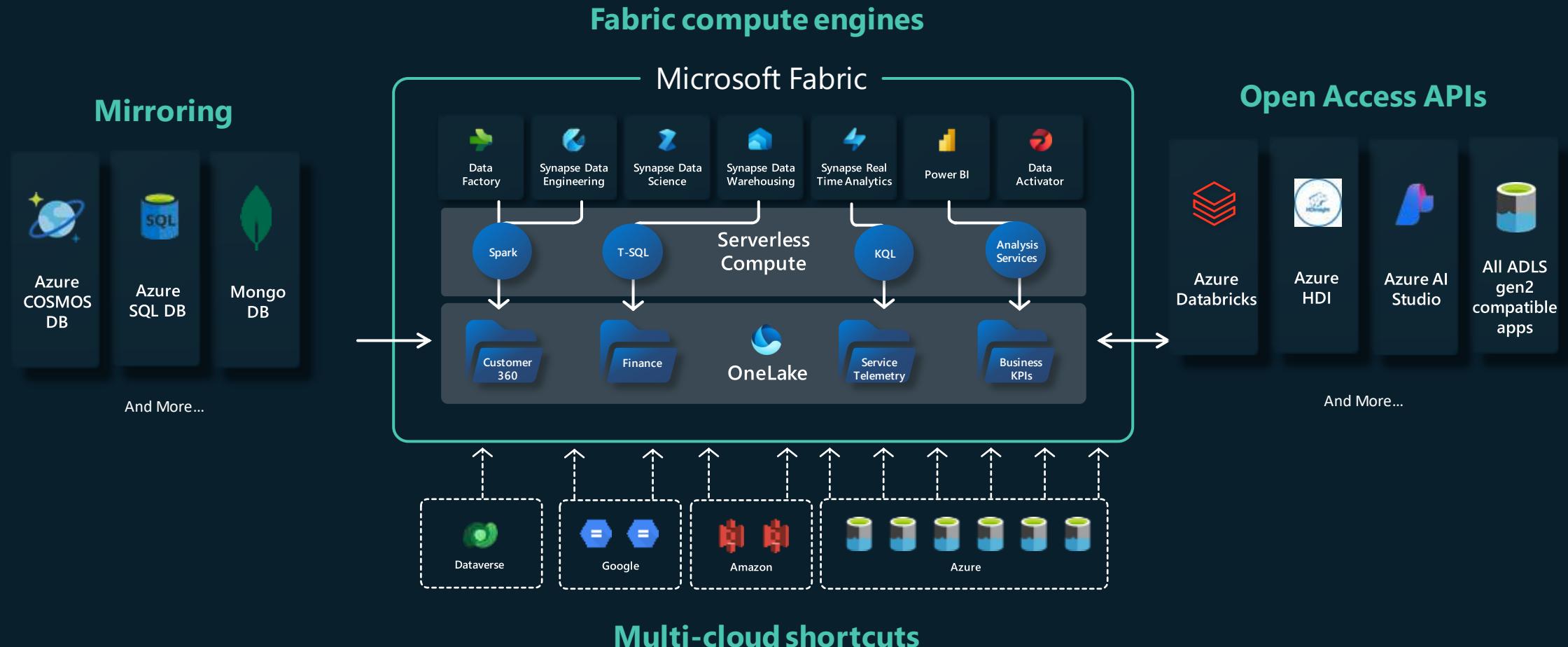
Identify and resolve data loading performance bottlenecks in dataflows, notebooks, and SQL queries

Implement performance improvements in dataflows, notebooks, and SQL queries

Identify and resolve issues with Delta table file sizes

All roads lead to OneLake

Creating Data Gravity in OneLake



Dataflow Gen2 benefits and limitations

Benefits:

- Allow self-service users access to a subset of data warehouse separately.
- Optimize performance with dataflows, which enable extracting data once for reuse, reducing data refresh time for slower sources.
- Simplify data source complexity by only exposing dataflows to larger analyst groups.
- Ensure consistency and quality of data by enabling users to clean and transform data before loading it to a destination.

Limitations:

- Not a replacement for a data warehouse.
- Row-level security isn't supported.
- Fabric capacity workspace is required.

Who is using a Medallion architecture?



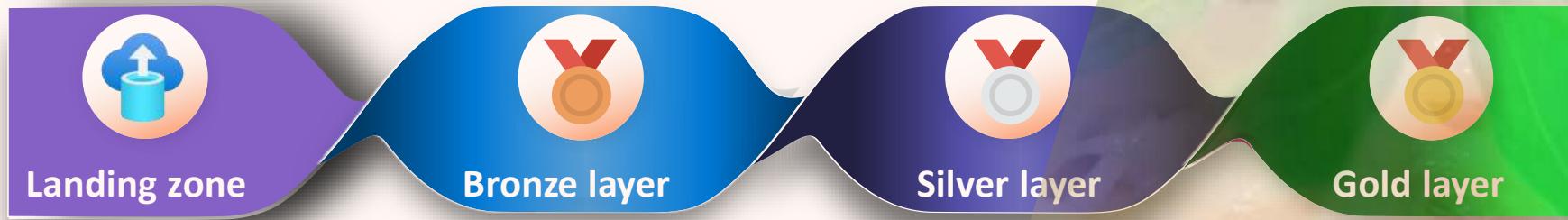
'Uniform data architecture'
From data "Spaghetti to Lasagna"



Medallion Architecture

'Data processing in different stages'

Stages



Medallion Architecture

'Data processing in different stages'

Stage:



Gold layer



Silver layer



Bronze layer



Landing zone

Definition:

- Dimensions & Facts (Star Schema)
- Historical Analysis
- Business rules
- Documentation
- Aggregated data
- Logical table names

Filetype:



Files/Tables:



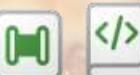
Fabric:



- Historical Data (Type 1 or 2)
- Data quality rules
- Data Cleansing
- Validated data
- No business model/data

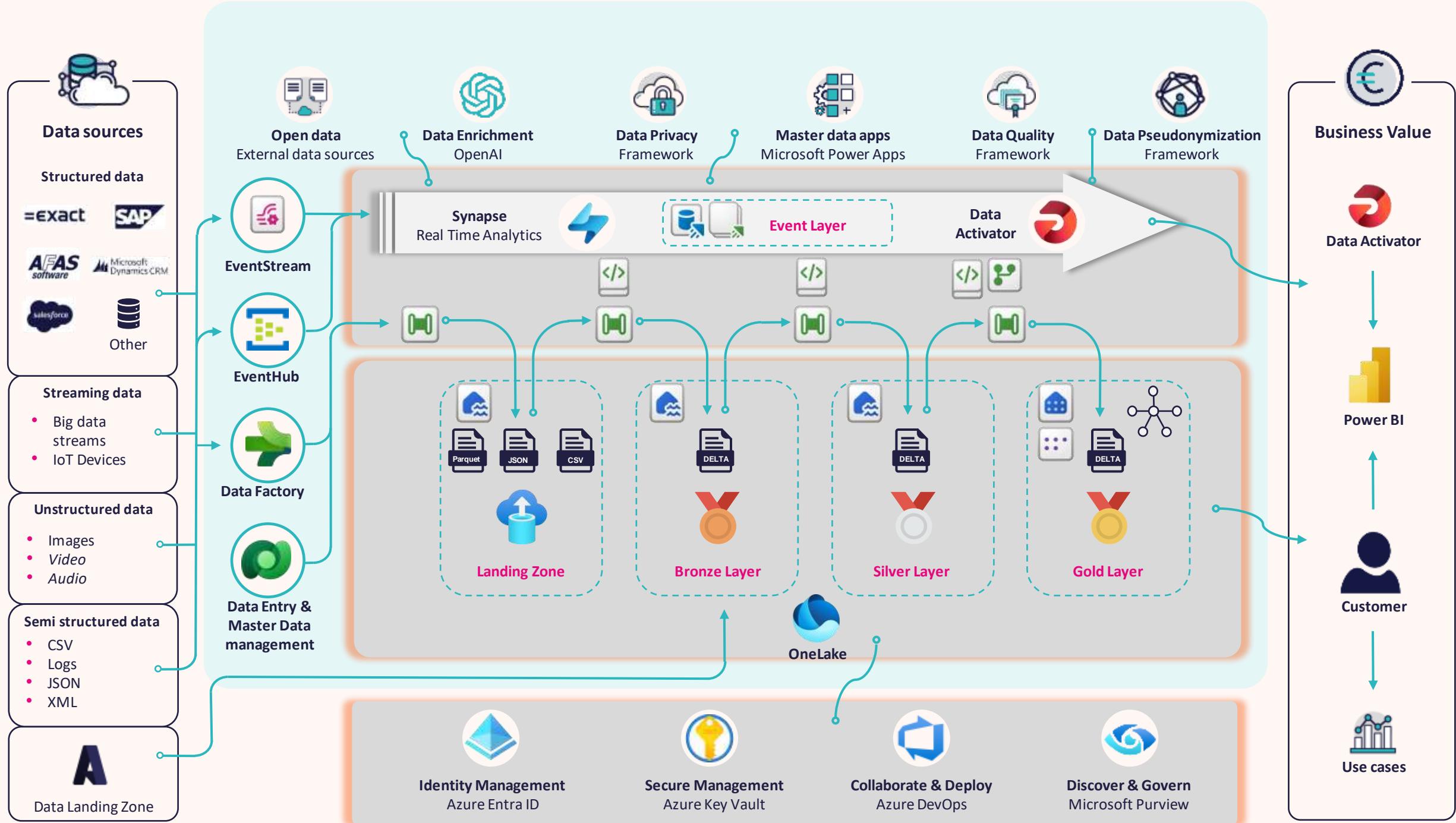


- Deduplicate data
- Add datatypes
- Data can be inconsistent
- Mostly a copy of the source
- Schema



- Structured data
- Unstructured data
- Incremental loads
- Data as is
- Stored in Datetime folder structure
- No Schema





Copy Activity

- Queue 11 seconds
- Reading from Source 12 seconds
- Writing to Sink 35 seconds
- Transfer 60 seconds

Copy data details

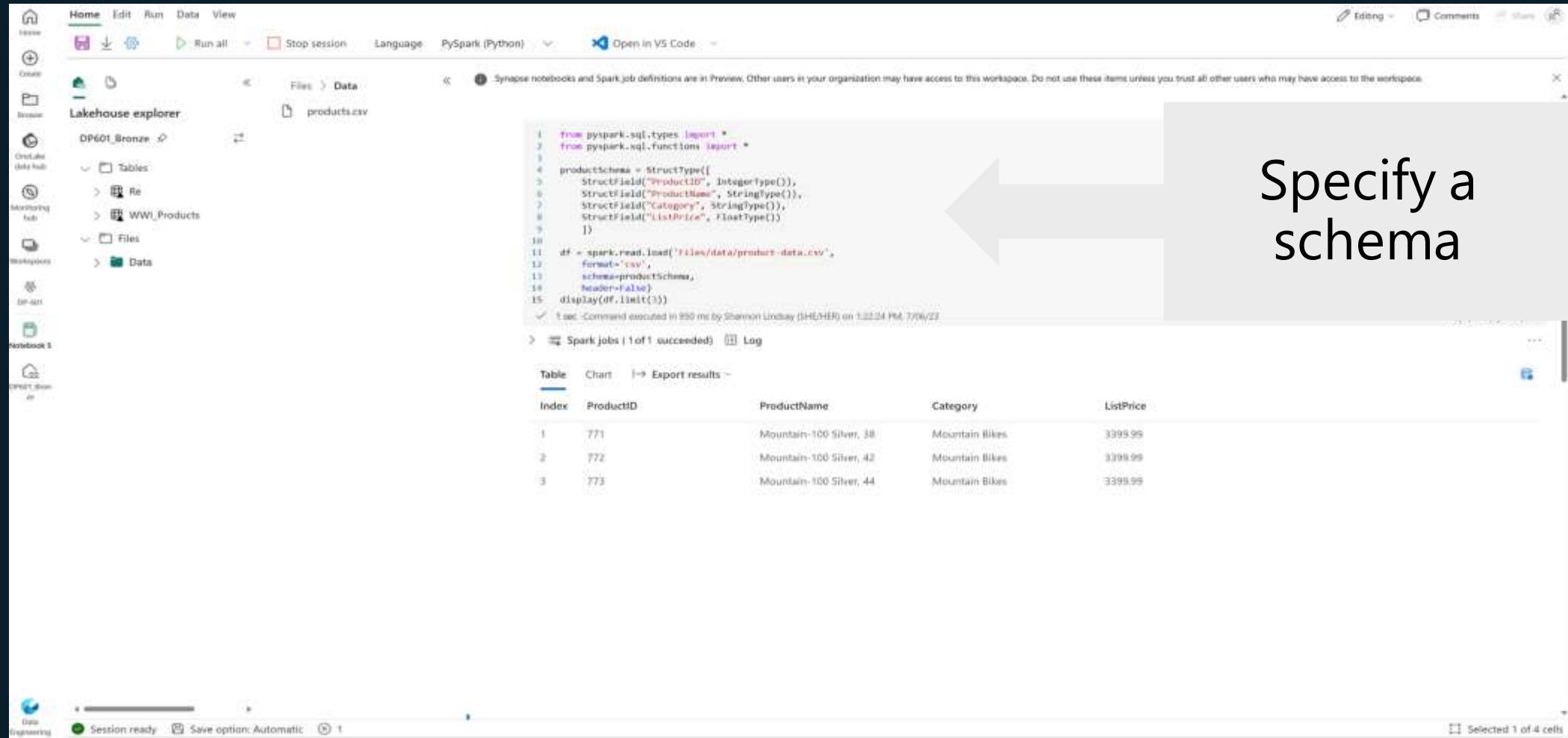
Source	Azure Data Lake Storage Gen2	Destination	Lakehouse
Data read:	3,65 GB	Data written:	3,65 GB
Files read:	21	Files written:	21

Status	✓ Succeeded
Start time	2/29/2024, 2:42:15 PM
Activity run ID	cec360c0-ed69-49c1-a3ea-c2033ecba3e6
Throughput	60,839 MB/s
Total duration	00:01:13
▼ Duration breakdown	
Start time	2/29/2024, 2:42:17 PM
Optimized throughput	Standard
Used parallel copies	6
Queue	<div style="width: 100px; background-color: blue;"></div>
Transfer	<div style="width: 900px; background-color: blue;"></div>
Reading from...	<div style="width: 200px; background-color: blue;"></div>
Writing to sink	<div style="width: 800px; background-color: blue;"></div>

Delta Lake in Microsoft Fabric

Load data in a Spark Dataframe

Schema can be either inferred or specified



The screenshot shows a Synapse notebook interface with the following details:

- Lakehouse explorer:** A sidebar showing a workspace named "DP601_Bronze" containing a "Tables" folder with "Re" and "WWI_Products" tables, and a "Files" folder with "Data".
- File:** The current file is "products.csv".
- Code:** PySpark Python code to read the CSV file and display the first three rows of the resulting DataFrame.

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

productSchema = StructType([
    StructField("ProductID", IntegerType()),
    StructField("ProductName", StringType()),
    StructField("Category", StringType()),
    StructField("ListPrice", FloatType())
])

df = spark.read.load('Files/data/product-data.csv',
    Format='csv',
    schema=productSchema,
    header=False)
display(df.limit(3))
```

- Output:** A table titled "Table" showing the first three rows of the DataFrame.

Index	ProductID	ProductName	Category	ListPrice
1	771	Mountain-100 Silver, 38	Mountain Bikes	3399.99
2	772	Mountain-100 Silver, 42	Mountain Bikes	3399.99
3	773	Mountain-100 Silver, 44	Mountain Bikes	3399.99

- Log:** Shows "1 job(s) (1 of 1 succeeded)" and a "Log" link.
- Bottom status bar:** Shows "Session ready", "Save option: Automatic", and "Selected 1 of 4 cells".

Specify a schema

Transform data in a Spark Dataframe

The screenshot shows the Microsoft Synapse Studio interface. On the left, the Lakehouse explorer sidebar displays a 'Lakehouse000' workspace with a 'Tables' section containing a 'salesorders' table and a 'Files' section. The main area shows a PySpark notebook with a code cell containing:

```
bikes_df.write.mode("overwrite").parquet('Files/product_data/bikes.parquet')
```

Below the code cell, a message indicates the command was executed successfully: "2 sec - Command executed in 1 sec. 754 ms by Shannon Lindsay (SHE/HER) on 12:16:38 AM, 7/11/23". A large arrow points from the text "Save dataframe for further analysis" to the "Save" icon in the top navigation bar.

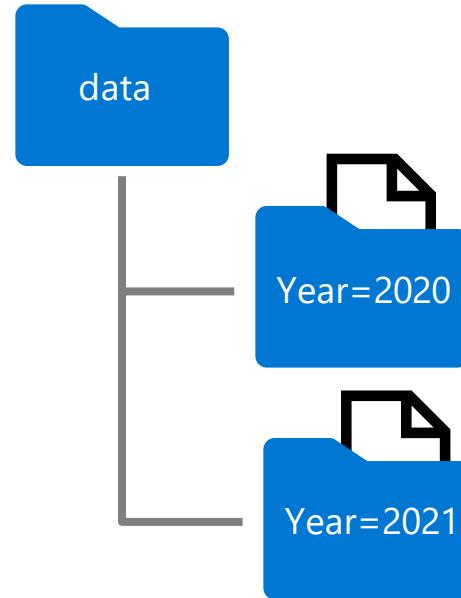
On the right, a "Spark jobs" panel shows a single job named "Job 30" with the description "parquet at NativeMethodAccessorsImpl.java:0". The job status is "Succeeded" with 1/1 stages and 1/1 tasks completed, and a duration of 1 sec.

At the bottom of the screen, the status bar shows "Session ready", "Save option: Automatic", and "Selected 1 of 2 cells".

Save dataframe
for further
analysis

Partition the output file

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



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

Work with data using Spark SQL

Use the metastore to define tables and views

```
# Create a view in the metastore  
df.createOrReplaceTempView("products_view")  
  
# Save a dataframe as a new table  
df.write.format("delta").saveAsTable("products")
```

Create external tables

```
# Create external table  
df.write.format("delta").saveAsTable("myexternaltable",path="Files/myexternaltable")
```

Query data using the Spark SQL API

Use the Spark SQL API in code written in any language to query data in the catalog

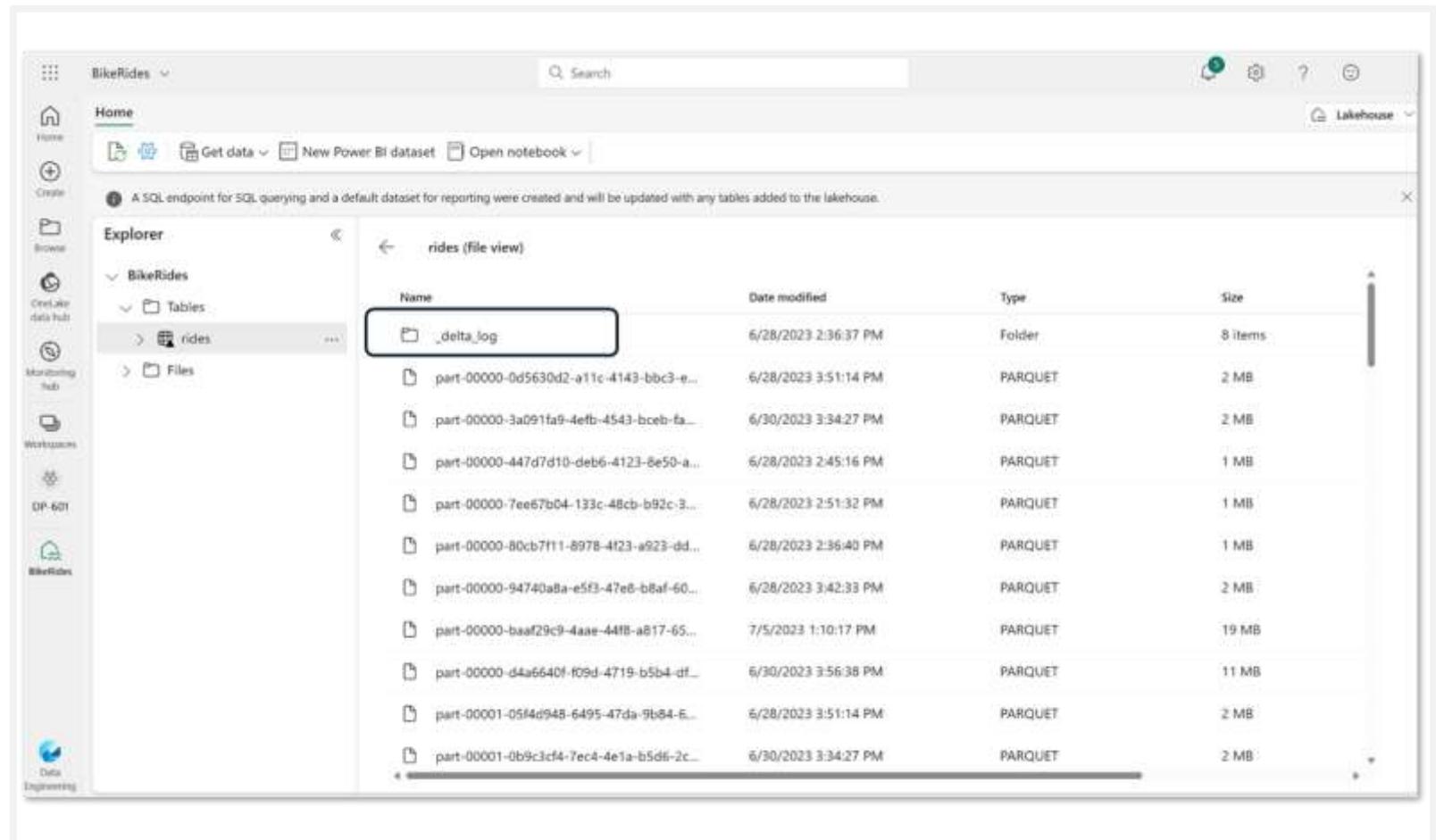
```
# Return data from the products table as a dataframe using PySpark  
  
bikes_df = spark.sql("SELECT ProductID, ProductName, ListPrice \  
                      FROM products \  
                      WHERE Category IN ('Mountain Bikes', 'Road Bikes')")  
display(bikes_df)
```

```
# Use %%sql magic to query objects in the catalog using native SQL
```

```
%%sql  
  
SELECT Category, COUNT(ProductID) AS ProductCount  
FROM products  
GROUP BY Category  
ORDER BY Category
```

Understand Delta Lake

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



Create Delta tables using code in Spark

1. Save a dataframe as a managed table

```
# Load a file into a dataframe
df = spark.read.load('Files/mydata.csv', format='csv', header=True)

# Save the dataframe as a delta table
df.write.format("delta").saveAsTable("mytable")
```

2. Use Spark SQL

```
%%sql

CREATE TABLE salesorders
(
    Orderid INT NOT NULL,
    OrderDate TIMESTAMP NOT NULL,
    CustomerName STRING,
    SalesTotal FLOAT NOT NULL
)
USING DELTA
```

3. Save a dataframe in delta format in an explicit path

```
delta_path = "Files/mydatatable"
df.write.format("delta").save(delta_path)
```

Managed vs external tables

Managed tables

- Defined without a specific location – Files are created in the default metastore folder (**Tables/...**)
- Dropping the table deletes the files

```
# Save a dataframe as a delta table  
df.write.format("delta").saveAsTable("mytable")
```

External tables

- Defined with an explicit file location outside of the default metastore folder
- Dropping the table does not delete the files

```
df.write.format("delta").saveAsTable("myexternaltable",path="Files/myexternaltable")
```

Work with Delta tables in Spark

1. Use Spark SQL to embed a SQL statement in PySpark

- Embed SQL statements in other languages using the **spark.sql** library.

```
spark.sql("INSERT INTO products VALUES (1, 'Widget', 'Accessories', 2.99)")
```

2. Native Spark SQL using %%sql magic

- Use %%sql magic in a notebook to run SQL statements.

```
%%sql
```

```
UPDATE products
SET Price = 2.49 WHERE ProductId = 1;
```

3. Use the Delta API

- Create an instance of a DeltaTable from a folder location containing files in delta format, and then use the API to modify the data in the table.

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

# Create a DeltaTable object
delta_path = "Files/mytable"
deltaTable = DeltaTable.forPath(spark, delta_path)

# Update the table (reduce price of accessories by 10%)
deltaTable.update(
    condition = "Category == 'Accessories'", 
    set = { "Price": "Price * 0.9" })
```

Data versioning and time travel

- Modifications made to delta tables are logged in the transaction log for the table.
- You can use the logged transactions to view the history of changes made to the table and to retrieve older versions of the data (known as time travel).

```
# Use SQL to see the history of a table
```

```
%%sql
```

```
DESCRIBE HISTORY products
```

```
# Use SQL to see the history of an external table, specifying folder location
```

```
%%sql
```

```
DESCRIBE HISTORY 'Files/mytable'
```

Use Delta tables with streaming data

Use Delta lake table as a streaming source

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

# Load a streaming dataframe from the Delta Table
stream_df = spark.readStream.format("delta") \
    .option("ignoreChanges", "true") \
    .load("Files/delta/internetorders")

# Now you can process the streaming data in the dataframe
# for example, show it:
stream_df.show()
```

Use Delta lake table as a streaming sink

```
# Create a stream that reads JSON data from a folder
inputPath = 'Files/streamingdata/'
jsonSchema = StructType([
    StructField("device", StringType(), False),
    StructField("status", StringType(), False)
])
stream_df = spark.readStream.schema(jsonSchema).option("maxFilesPerTrigger", 1).json(inputPath)

# Write the stream to a delta table
table_path = 'Files/delta/devicetable'
checkpoint_path = 'Files/delta/checkpoint'
delta_stream = stream_df.writeStream.format("delta").option("checkpointLocation", checkpoint_path).start(table_path)
```

INSIDE THE APACHE PARQUET FORMAT IT'S OPEN SOURCE!

IN TRADITIONAL DATABASES (POSTGRES, SQL SERVER*) AND FILE TYPES (CSV, JSON), DATA IS STORED AS ROWS...

FIRST_NAME	LAST_NAME	STREET	ISPREMIUMC USTOMER	NEXT MONTH BUYING PROBABILITY
ABIGAIL	HAYES	4009 9TH STREET	1	1
AMANDA	CLARK	87 5TH STREET	0	2
MELISSA	CLARK	12650 5TH STREET NORTH	1	0.5438465585898219
SARA	BENNETT	203 STATE STREET	0	0.15856312140709694

Customers.csv

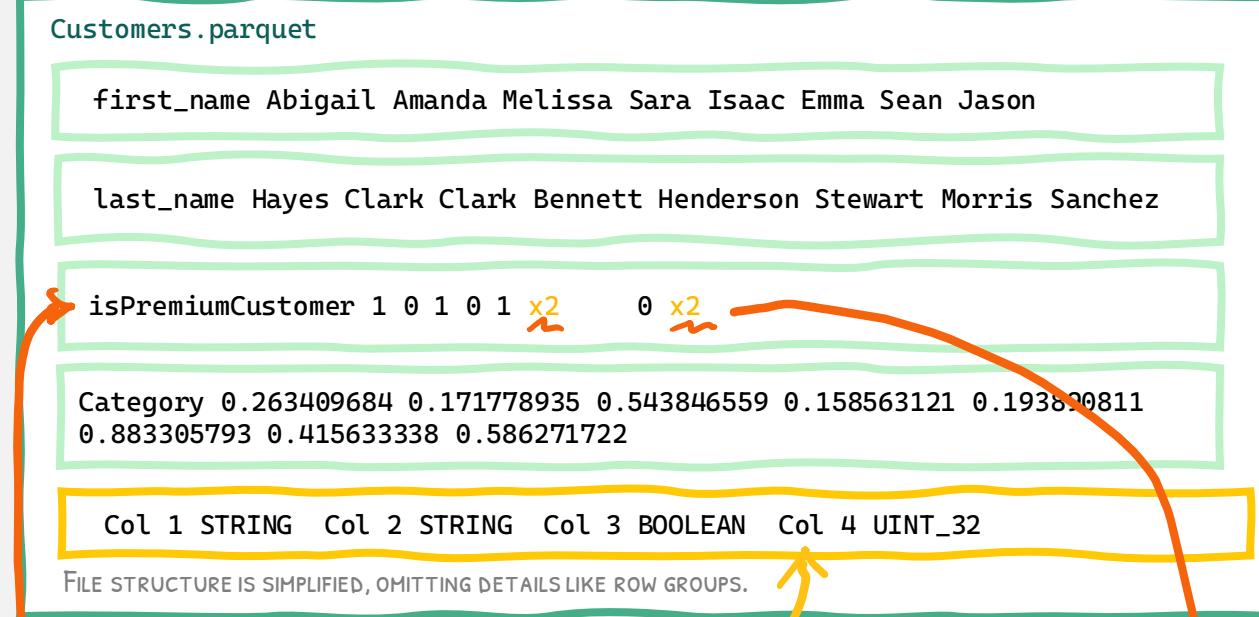
```
first_name,last_name,street,isPremiumCustomer,nextmbuyprob
Abigail,Hayes,4009 9th Street,1,0.26340968441351104
Amanda,Clark,87 5th Street,0,0.17177893540625921
Melissa,Clark,12650 5th Street North,1,0.5438465585898219
Sara,Bennett,203 State Street,0,0.15856312140709694
Isaac,Henderson,732 Magnolia Drive,1,0.19389081059172666
Emma,Stewart,47 Route 32,1,0.8833057932776623
Sean,Morris,254 Lincoln Avenue,0,0.415633382834478
Jason,Sanchez,26050 Madison Avenue,0,0.5862717219123177
Alexander,Walker,54051 Valley Road,1,0.5941814808482648
Megan,Morris,31 Delaware Avenue,1,0.6289626353900487
```

... WHICH IS NOT OPTIMAL FOR ANALYTICAL NEEDS:

- 👎 FOR SOME FILE TYPES (LIKE CSV), THERE IS NO EMBEDDED SCHEMA
- 👎 COMPRESSION IS NOT GOOD (YOU'RE KINDA LIMITED TO TEXT COMPRESSION)
- 👎 YOU NEED TO SCAN(READ) ALL THE ROWS

*UNLESS YOU'RE USING COLUMNSTORE TABLES :)

IN PARQUET, DATA IS STORED AS COLUMNS



PARQUET EMBEDS THE DATA SCHEMA

WANT THE NUMBER OF PREMIUM CUSTOMERS? YOU CAN COMPUTE THAT AGGREGATE WITHOUT READING ANY OTHER UNNECESSARY DATA!

WE CAN EFFICIENTLY COMPRESS REPETITIVE DATA LIKE THIS ONE

WITH FABRIC, YOU CAN EASILY READ PARQUET FILES STORED IN ONELAKE OR IN OTHER CLOUDS

Delta Transaction Log

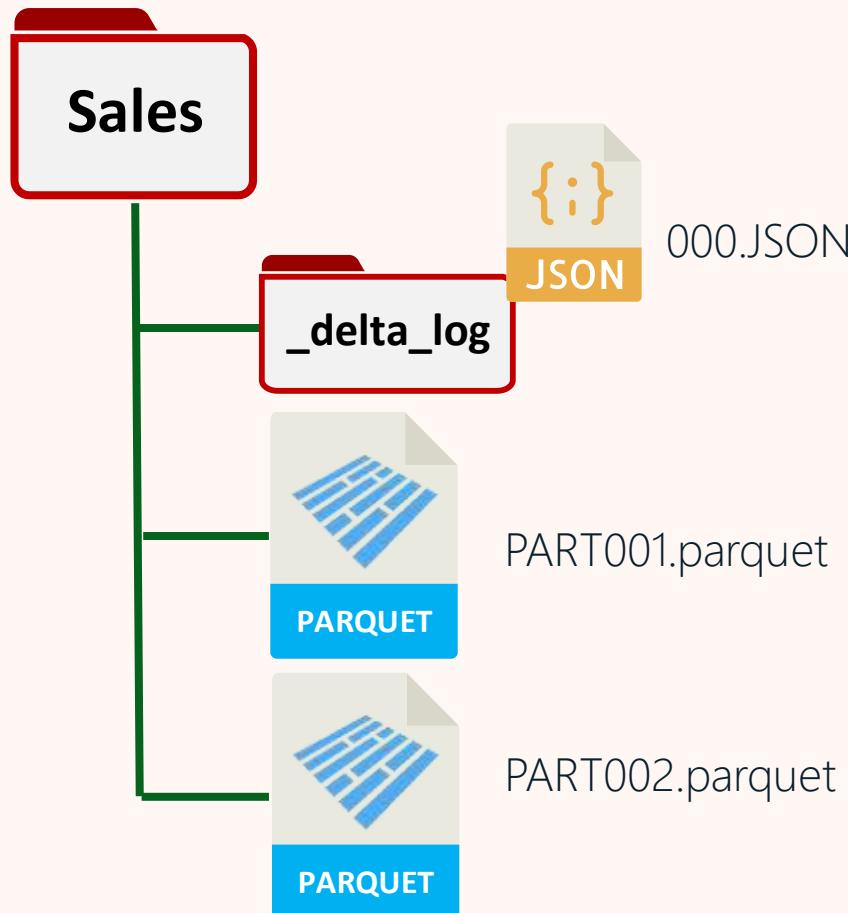


Table Version	File Changes
0	+ PART001.parquet + PART002.parquet



Delta Transaction Log

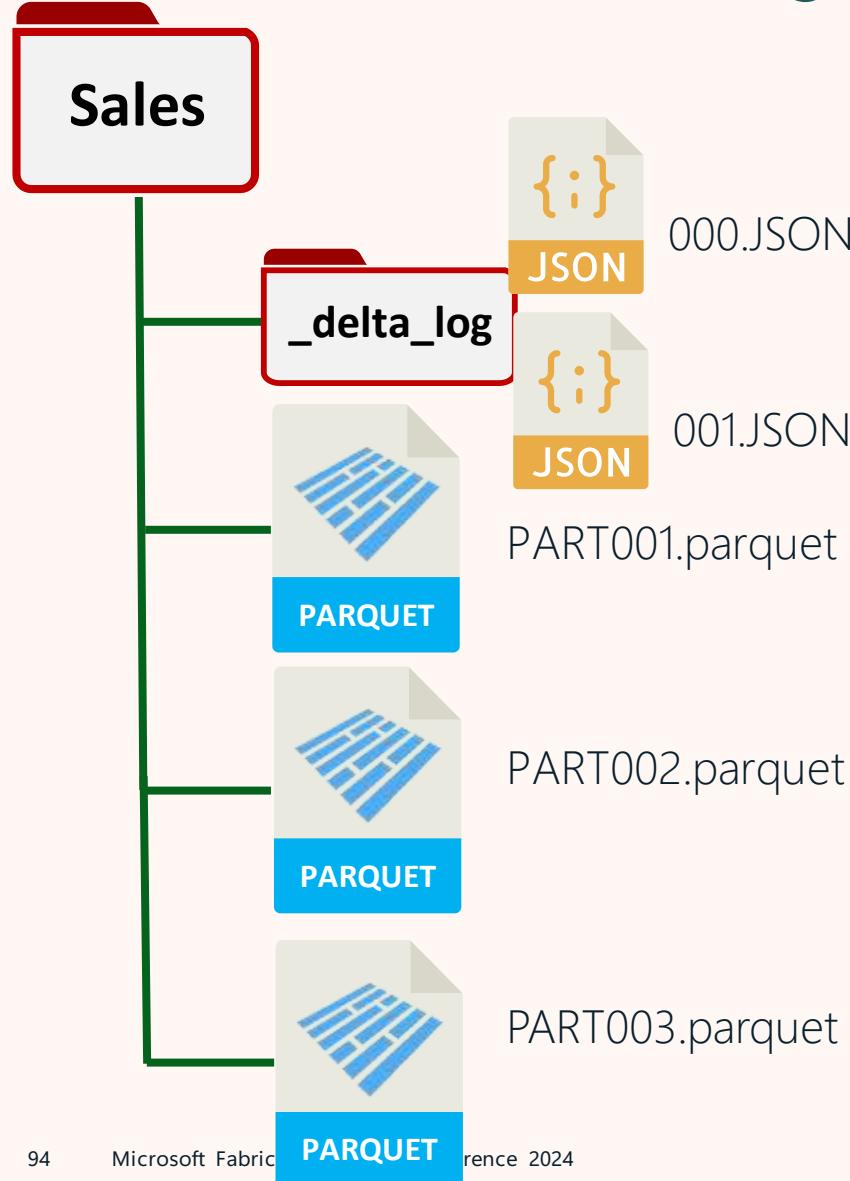


Table Version	File Changes
0	+ PART001.parquet + PART002.parquet
1	- PART001.parquet - PART002.parquet + PART003.parquet

**DELETE * FROM SALES
WHERE Segment = 3**



Reading Delta

SELECT * FROM SALES

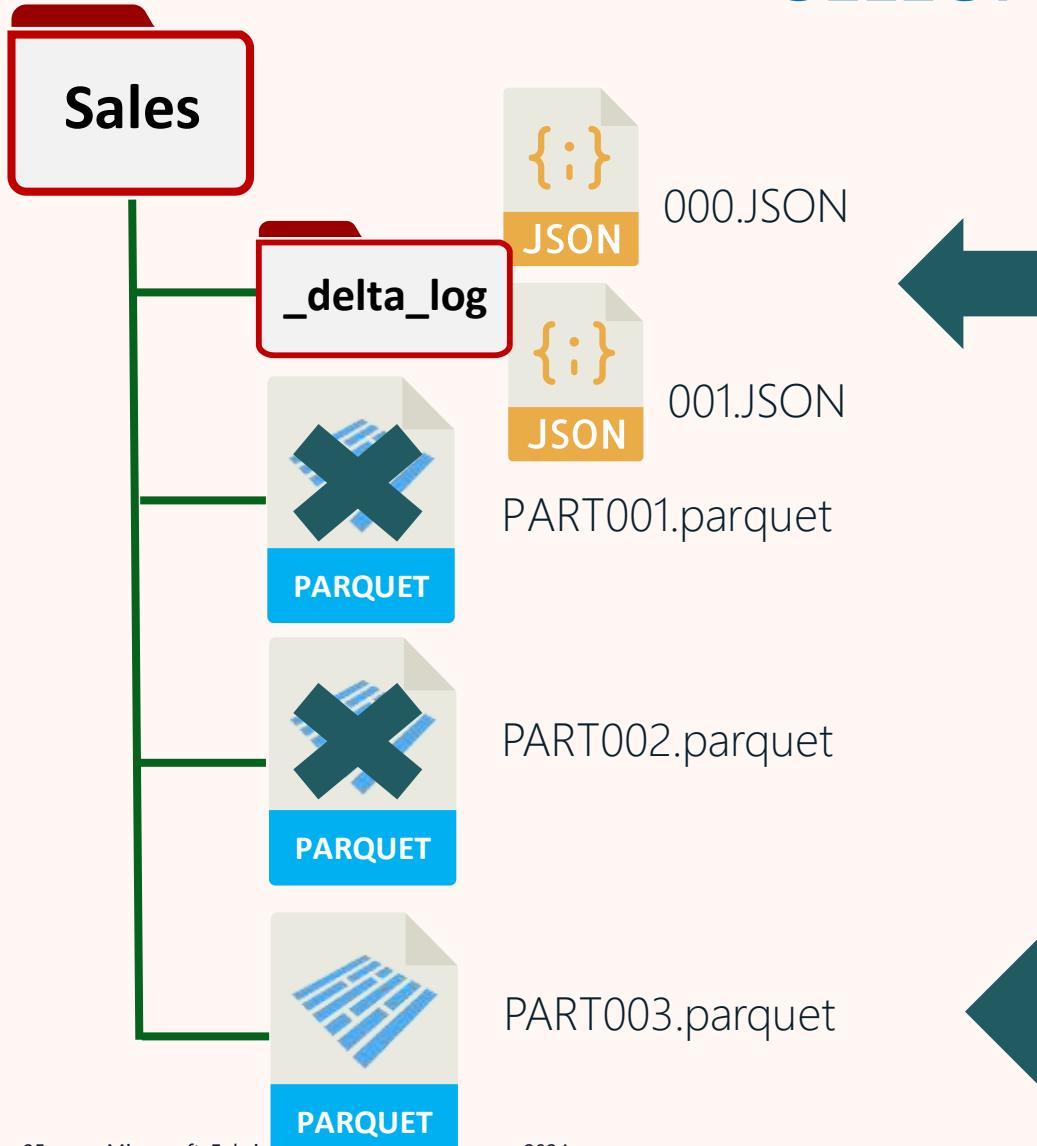


Table Version	File Changes
0	+ PART001.parquet + PART002.parquet
1	- PART001.parquet - PART002.parquet + PART003.parquet



Time Travel

SELECT * FROM SALES VERSION AS OF 0

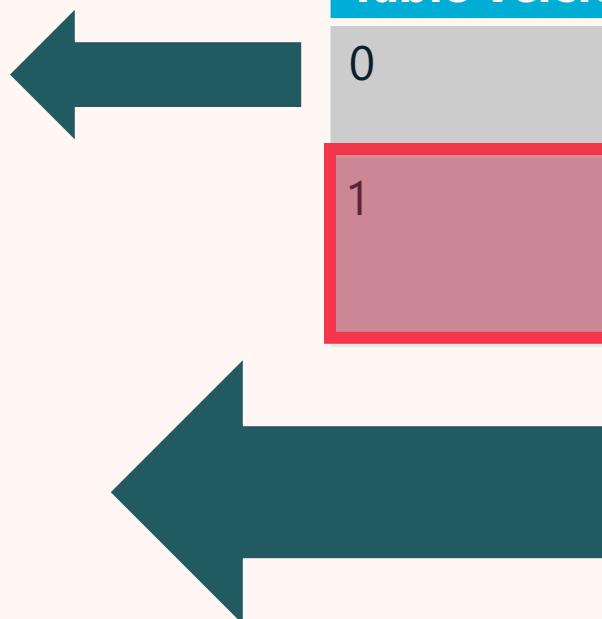
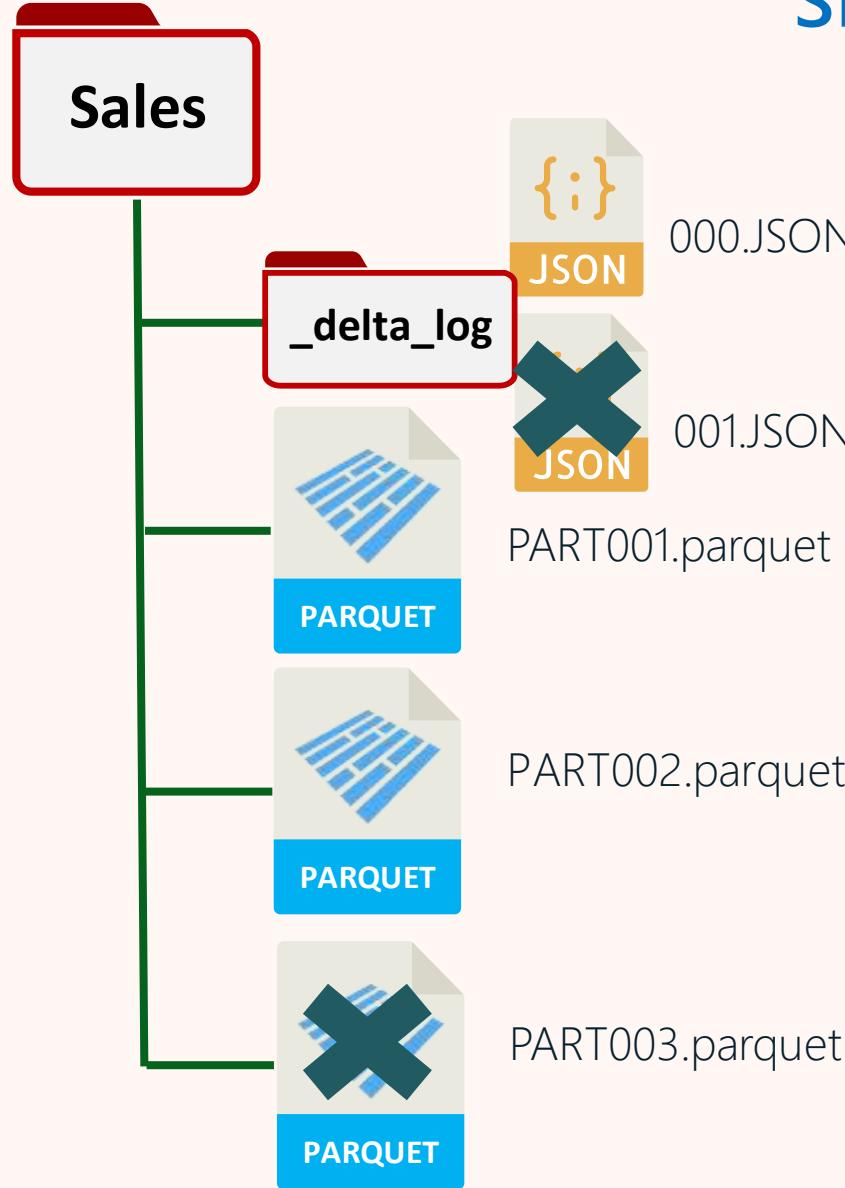
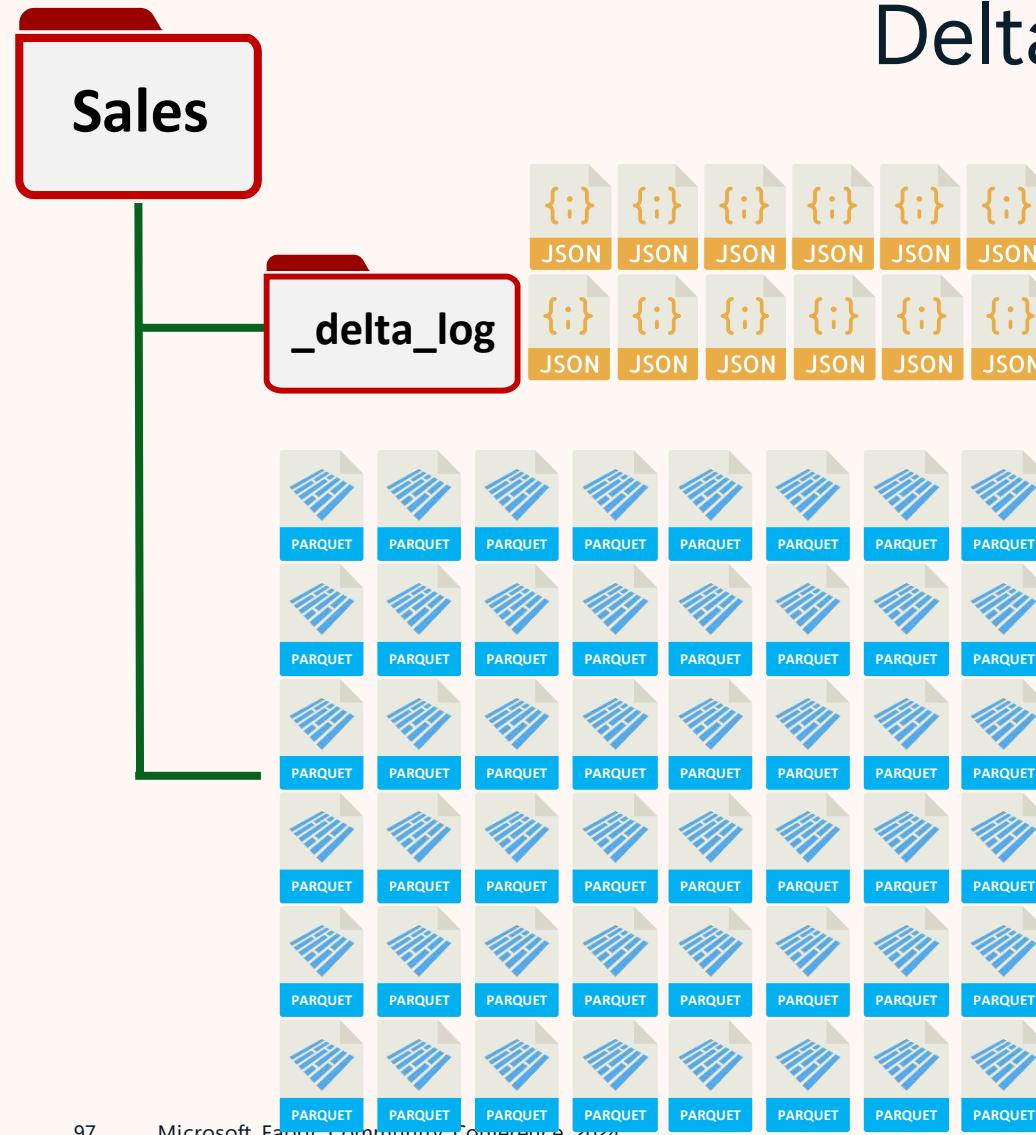


Table Version	File Changes
0	+ PART001.parquet + PART002.parquet
1	- PART001.parquet - PART002.parquet + PART003.parquet



Vacuum



To remove obsolete history files,
Delta has the **VACUUM** command

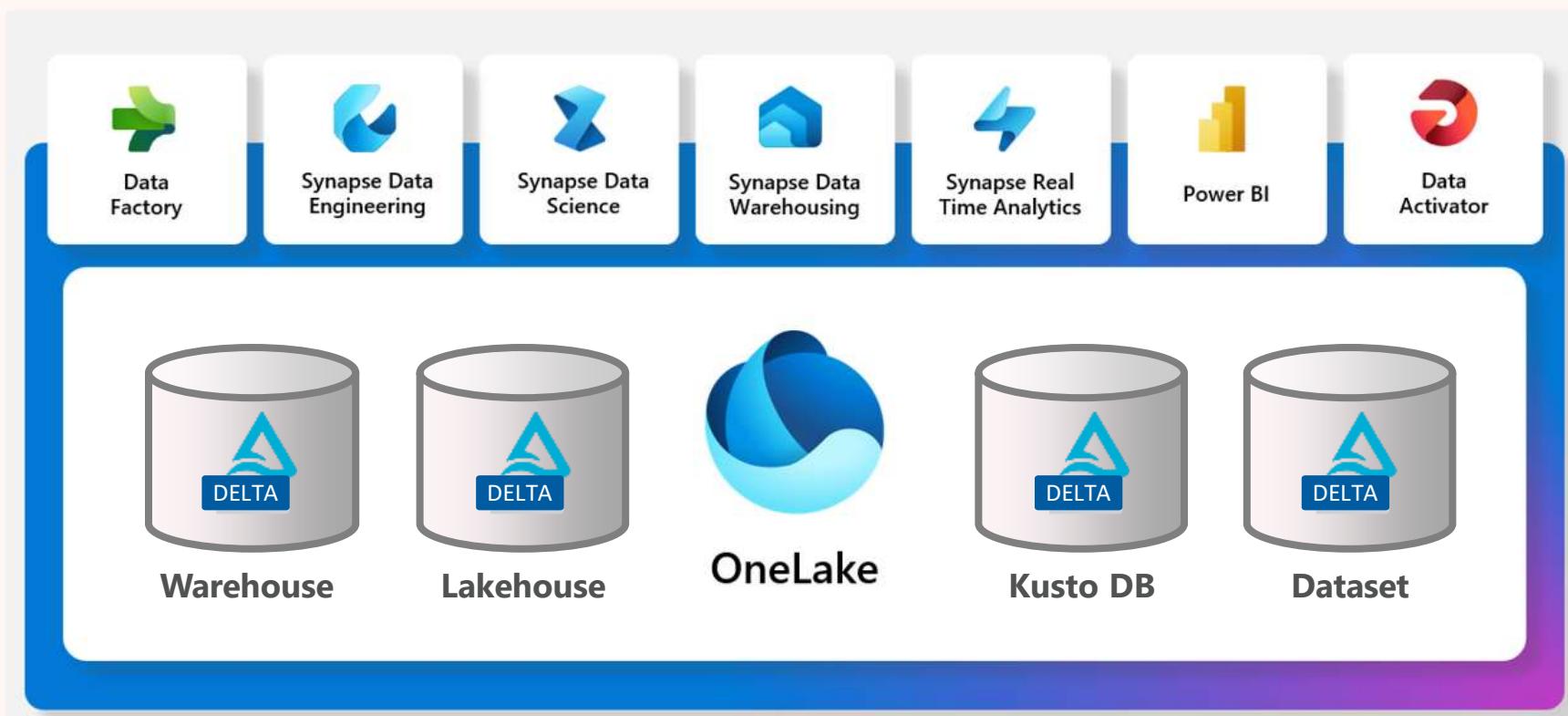
This command physically deletes
data files older than a specified
date

You **CANNOT** time travel past dates
where history has been vacuumed

Delta Lake table format in Microsoft Fabric

Delta Lake tables are the unifying table format in Microsoft Fabric.

Multiple experiences, with multiple compute engines, OneCopy of your data.



Principles and challenges

If one compute engine in Microsoft Fabric writes a Delta Lake table, **all other engines must be able to read it.**

Full interoperability and limitations documentation.

We align to the **Delta Lake open-source** protocol specification.

We contribute back to the OSS project.

Like we do to other projects such as Apache parquet, Spark, and others.

Twelve engines (as of today) in Microsoft Fabric are aligned on Delta Lake protocol,

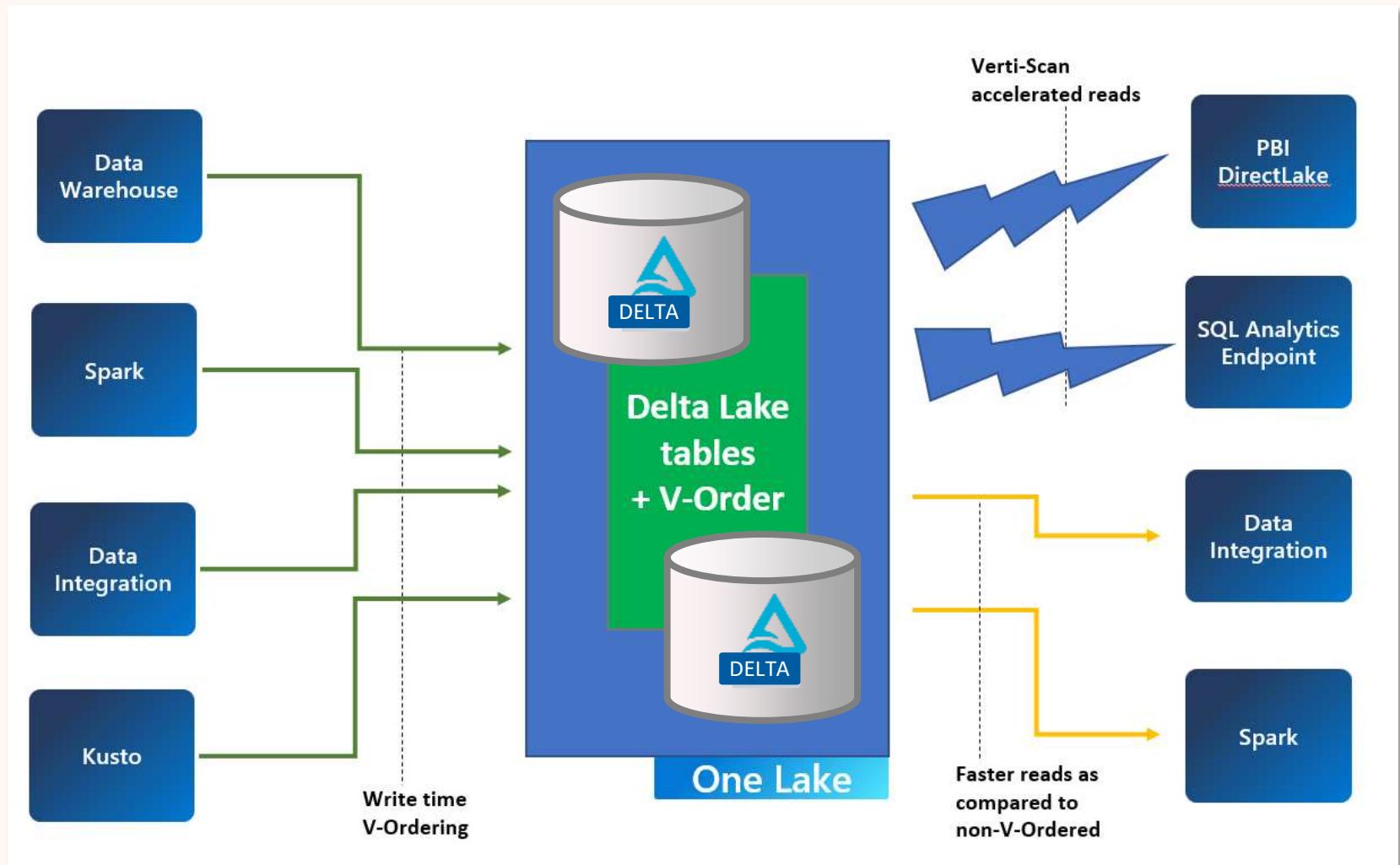
We need to go beyond the OSS provided libraries, to develop and integrate internal C++ and .NET Delta Lake protocol implementations to our products.

What is V-Order?

V-Order is a write time optimization to the parquet file format that enables lightning-fast reads under the Microsoft Fabric engines.

Microsoft Fabric engines write V-Ordered parquet files **by default**.

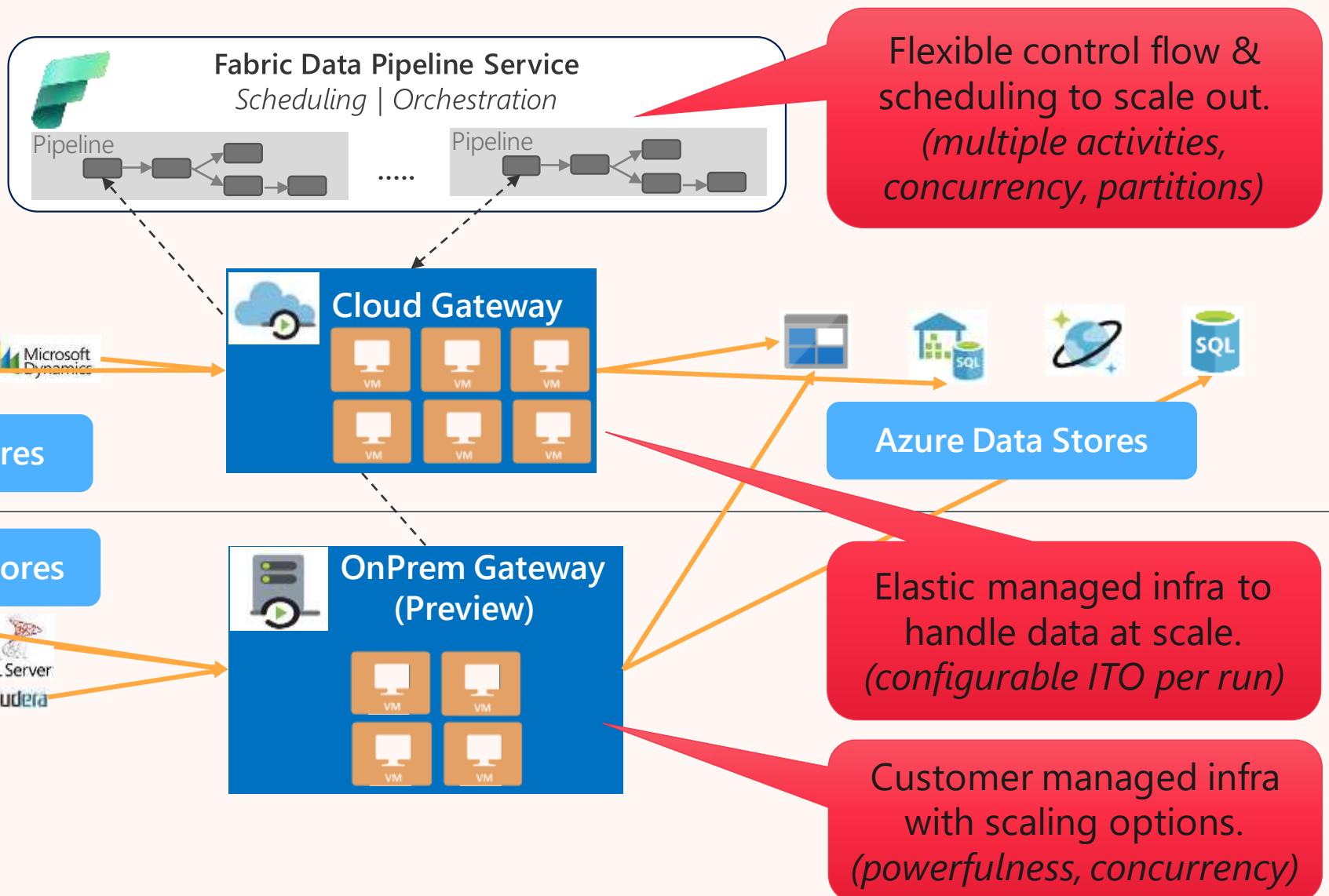
Table APIs are provided to manage V-Order on Lakehouse



Pipeline Architecture

Understand How Pipeline Scales

↔ Command and Control
→ Data



Copy Performance Metrics

Parquet -> LH (binary)

Copy data details	
loading 1t parquet to lh table binary	
Source	Destination
Azure Data Lake Storage Gen2	Lakehouse
Data read: 1,049 TB	Data written: 1,049 TB
Files read: 5,120	Files written: 5,120
Throughput	5.141 GB/s
Total duration	00:03:31
Duration breakdown	
Start time	2/26/2024, 9:05:35 PM

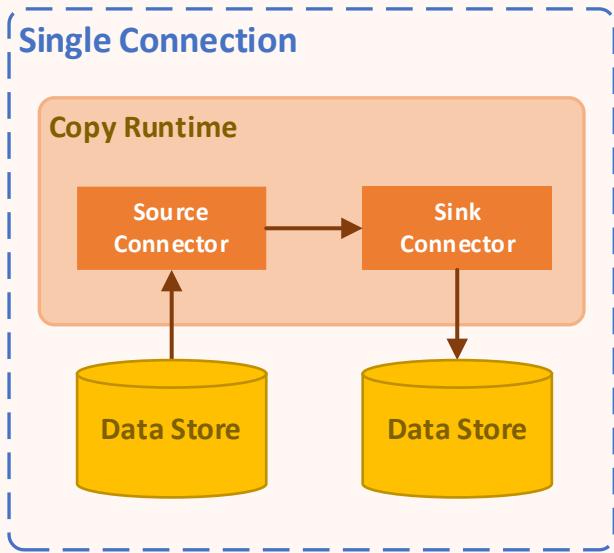
CSV -> LH

Copy data details	
Loading 1T csv to LH table	
Source	Destination
Azure Data Lake Storage Gen2	Lakehouse
Data read: 1,049 TB	Data written: 353.654 GB
Files read: 5,120	Files written: 768
Rows read: 7,583,957,915	Rows written: 7,583,957,915
Throughput	1.284 GB/s
Total duration	00:13:44
Duration breakdown	
Start time	2/26/2024, 3:21:27 PM

SQL -> LH

Copy data details	
sql perf	
Source	Destination
Azure SQL Database	Lakehouse
Data read: 127.292 GB	Data written: 24.172 GB
Rows read: 512,000,000	Files written: 256
Throughput	Rows written: 512,000,000
Total duration	Duration breakdown
Status	Succeeded
Start time	3/6/2024, 10:29:11 PM
Activity run ID	45f452c0-4172-40ea-9a09-f4a87d8cb0e6
Throughput	1.354 GB/s
Total duration	00:01:41
Duration breakdown	
Start time	3/6/2024, 10:29:13 PM

How Copy Scales – Connection Level



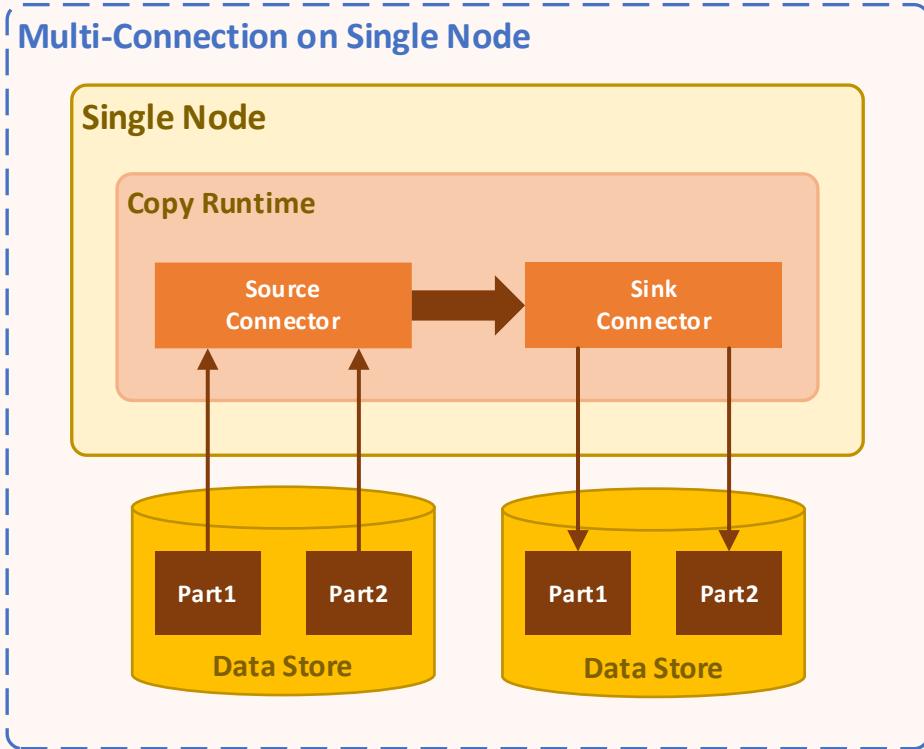
Pipeline processing

- **Less memory:** No need to load everything in memory and then write
- **Less total duration:** Read and Write are in parallel

Clock	0	1	2	3	4	5
Four batches to read						
Source connector						
Destination connector						
Total Duration						

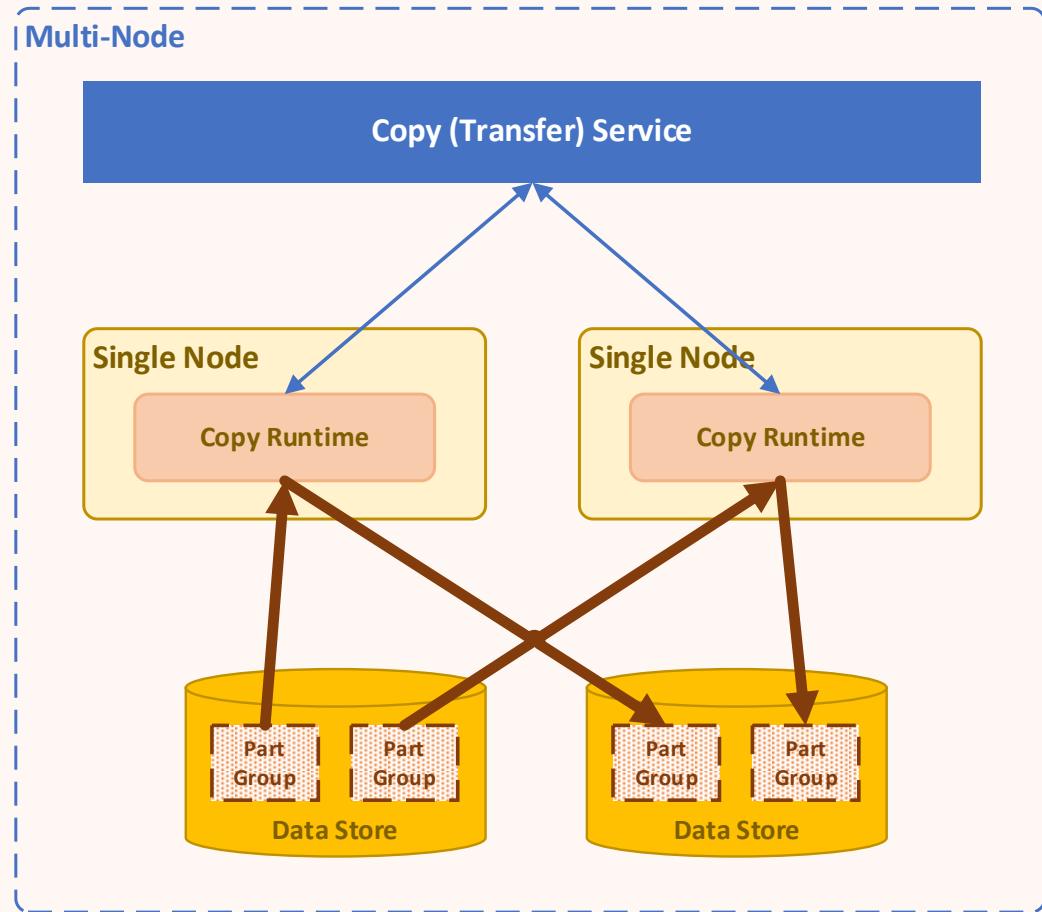
The table illustrates the pipeline processing of four batches. The columns represent time steps from 0 to 5. The first column is labeled 'Clock'. The second column is labeled 'Four batches to read'. The third column is labeled 'Source connector'. The fourth column is labeled 'Destination connector'. The fifth column is labeled 'Total Duration'. The data is represented by colored blocks: red at clock 0, blue at clock 1, purple at clock 2, green at clock 3, blue at clock 4, and red at clock 5. The 'Source connector' row shows the sequence of reading batches (red, blue, purple, green). The 'Destination connector' row shows the sequence of writing batches (blue, purple, red, red). The 'Total Duration' row shows the cumulative duration of the process.

How Copy Scales – Node Level



- Producer & Consumer design
 - Data is partitioned for multiple concurrent connections (even for single large file)
 - Full node utilization: Data can be partitioned differently between source and destination to avoid any starving / idle connection
- Partitions come from
 - Physical partitions setup on DB side
 - Dynamic partitions from different queries
 - Multiple files
 - Multiple parts from a single file

How Copy Scales – Multi-Node Level



- **Copy(Transfer) Service:** Manages Copy activities
 - Copy State Management / Monitoring
 - Cross Machine Parallelism
 - Billing
 - Manages Compute Clusters and tasks running on them
- Easily scale out to multiple stateless nodes when available partitions are more than what one node can afford.
- Theoretically no upper limit on the performance.

Copy Design Considerations

- Concepts
 - **Intelligent Throughput Optimization (ITO):** A measure that represents the power (a combination of CPU, memory, and network resource allocation) used for a single Copy activity.
 - **Parallel Copy:** The maximum number of threads within the Copy activity that read from source and write to destination in parallel.
 - **Max Concurrent Connections:** The upper limit of concurrent connections established to the data store during the activity run. (Usually used to avoid throttling)
- Implementation details
 - Files based connectors
 - 1 file = 1 partition
 - Tabular connectors that support Partition read
 - SQL family, Oracle, Teradata, Azure PostgreSQL, Netezza, SAP HANA, SAP Table, SAP Open Hub
 - **Default perf settings**
 - Intelligent throughput optimization: **Auto** (Use maximum available VM resources)

Troubleshoot Performance Issues

- Common bottlenecks
 - Network (cross region/OnPre)
 - Source/Destination data stores
 - Busy neighbors on the same data store
 - Low service/compute tier
 - Throttling mechanism on data store
 - Complex/unoptimized query (huge timeToFirstByte)

Troubleshoot Performance Issues – Example

Copy data details

Copy data1

Source: Azure SQL Database → Destination: Lakehouse

Data read: 57.733 MB, Rows read: 232,590

Data written: 47.61 MB, Files written: 1, Rows written: 232,590

Status: Succeeded, Start time: 3/6/2024, 7:45:31 PM

Activity run ID: 19f15fea-2021-4be9-a5dd-3d14fb14019, Throughput: 4.441 MB/s, Total duration: 00:00:27

Duration breakdown:

Stage	Time Spent
Queue	Very Short
Transfer	Very Short
Reading from...	Medium
Writing to sink	Very Short

Advanced

Close

1 Bottleneck at Source

Parameters Variables Settings Output

Pipeline run ID: 7fdb813b-73c7-470c-bd04-cff8d959d107 Pipeline status: Succeeded

Activity name	Activity status	Run start	Duration	Input
Copy data1	Succeeded	3/6/2024, 7:45:31 PM	33s	-

2

3

```
{
  "profile": {
    "queue": {
      "status": "Completed",
      "duration": 1
    },
    "transfer": {
      "status": "Completed",
      "duration": 13,
      "details": {
        "readingFromSource": {
          "type": "AzureSqlDatabase",
          "workingDuration": 10,
          "timeToFirstByte": 9
        },
        "writingToSink": {
          "type": "Lakehouse",
          "workingDuration": 1
        }
      }
    }
  }
}
```

Indicating that most time was spent on preparing/running the query on DB side. Optimize the query or upgrade DB tier to eliminate this bottleneck.

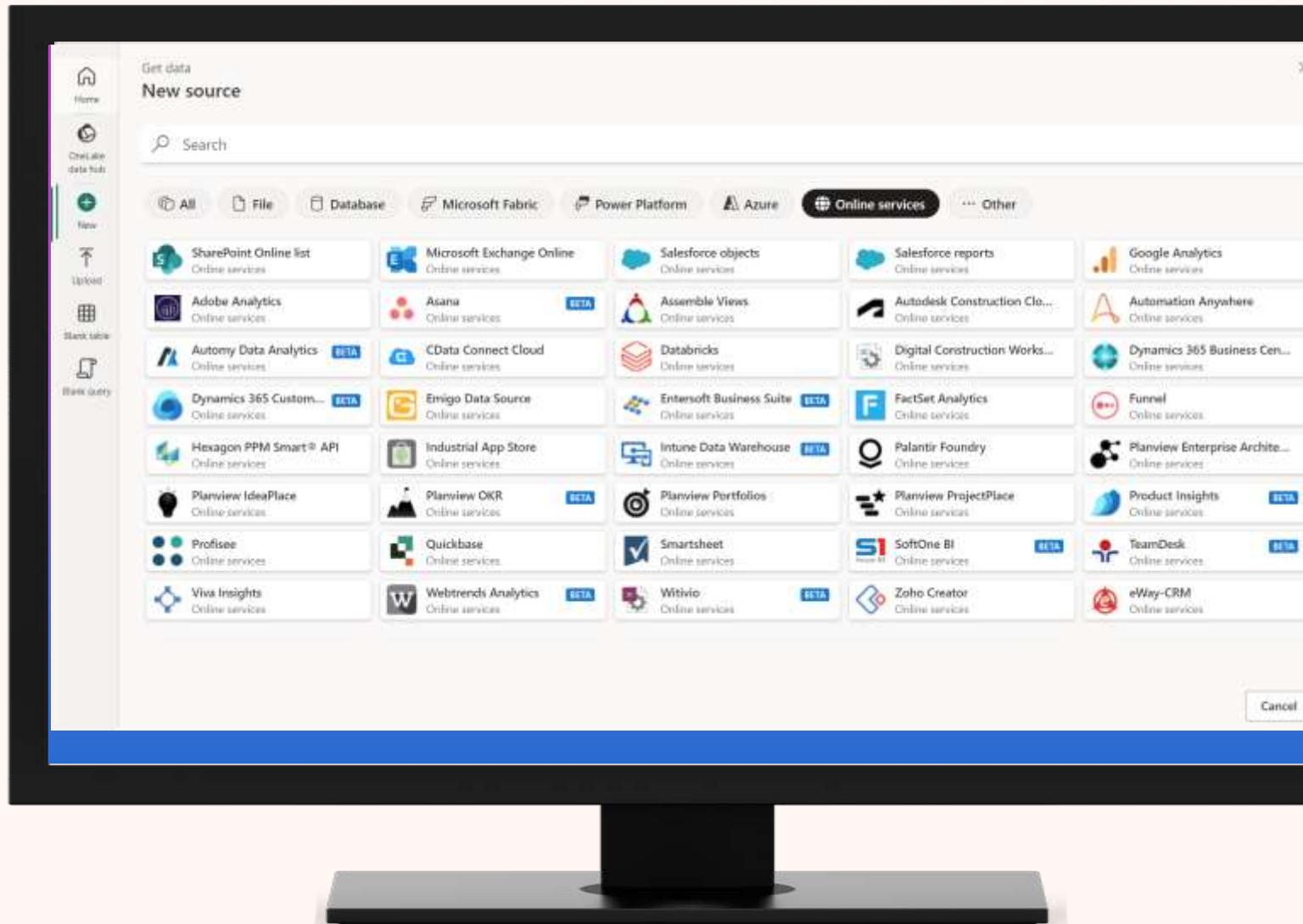
Dataflows

Dataflows Gen 2

- Dataflow Gen 2 in **Microsoft Fabric** is a powerful data preparation technology that allows you to create, transform, and load data into Fabric and Azure destinations. Here's what you can do with it:
 - **Create Dataflows:**
 - Dataflows are self-service, cloud-based tools for data preparation.
 - **Get Data:**
 - Dataflows enable you to retrieve data from 100s of on-premise and cloud data sources.
 - **Apply Transformations:**
 - Once you've connected to your data source, it's time to shape it according to your needs.
 - Use the Power Query editor to apply transformations. For instance:
 - Calculate the total number of orders per customer using the Group By feature, combine, remove columns, etc.
 - **Configure Destinations:**
 - Specify a destination to store the results of the query and transformations.
 - **Publish Dataflows:**
 - After applying transformations, you can publish your dataflow so that it can start processing data.

Get Data

- Fabric
- Database
- Azure
- ~175 connectors



Apply Transformations

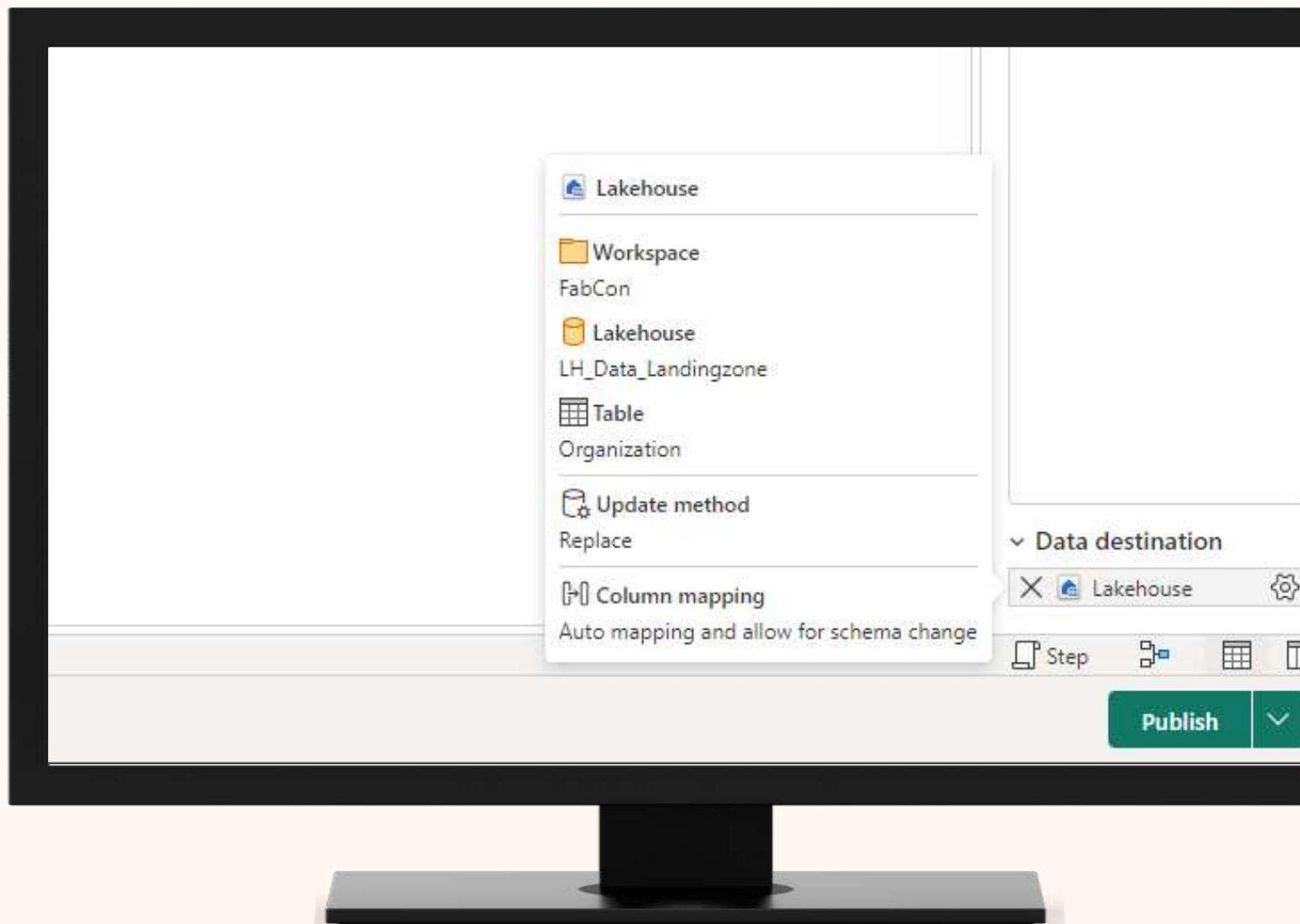
- Change data types
- Remove Columns
- Aggregations

The screenshot shows the Microsoft Power BI desktop application interface. The main area displays a table with columns: Organization ID, All names, URL, Website, Country, Description, Founded, and Industry. A context menu is open over the 'Country' column, listing various transformation options such as Copy preview data, Remove columns, Remove other columns, Duplicate column, Add column from examples, Remove duplicates, Remove errors, Split column, Replace values, Replace errors, Change type, Transform column, Group by, Fill, Unpivot columns, Unpivot other columns, Unpivot only selected columns, Rename, Move, Drill down, Add as new query, and Copy as JSON. The 'Remove columns' option is highlighted.

Organization ID	All names	URL	Website	Country	Description	Founded	Industry
1C11ba	Copy preview data	www.com/	Cape Verde	Horizontal bi-directional artificial intelligence	1971 Professional Training		
947bb6	Remove columns	www-marks.com/	Reunion	Progressive maximized instruction set	2008 Investment Management / Hedge Fund / Private Equity		
0cf0f3	Remove other columns	www.orz.org/	South Africa	Decentralized dynamic attitude	1993 Music		
c988e4	Duplicate column	www.nft/	Congo	Intuitive educating approach	2020 Information Technology / IT		
dc1c8f	Add column from examples	www.banera.com/	Eritrea	Optional well-modulated budgetary management	1987 Recreational Facilities / Services		
5218ef	Remove duplicates	www.manda.com/	Burkina Faso	Central territory task-force	1987 Consumer Electronics		
332951	Remove errors	www.anthony-brun.com/	Pitcairn Islands	Self-enabling attitude-oriented task-force	2007 Management Consulting		
760023	Split column	www.silence-huffman.net/	Nigeria	Customizable asymmetric initiative	1978 Photography		
760073	Replace values	www.mizormia-read.com/	Greece	Cross-group bottom-line archive	2011 Government Relations		
ff004d	Replace errors	www.com/	Moldova	De-engineered maximized complexity	1988 Entertainment / Movie Production		
4ff7f3	Change type	www.com/	Netherlands Antilles	Synergized eco-centric process improvement	2003 Sports		
0cc028	Transform column	www.com/	Ghana	User-friendly motivating project	2003 Law Practice / Law Firms		
71a0f8	Group by	info-expressa.it/	Colombia	Versatile foreground collaboration	2009 Hospital / Health Care		
21a899	Fill	www.com/	Germany	Total 24/7 matrix	1985 Motion Pictures / Film		
202013	Unpivot columns	www.mission-wake.com/	Turks and Caicos Islands	Enhanced coherent functionalities	1978 Biotechnology / GreenTech		
5462f1	Unpivot other columns	www.valequed-aeronaut.org/	Bulgaria	Public-key real-time groupware	2002 Architecture / Planning		
6000c5	Unpivot only selected columns	www.eden.info/	Saint Helena	Upgradable demand-driven challenge	1987 Design		
101168	Rename	www.deon.info/	Fiji Islands	Adapted demand-driven portals	1990 Animation		
122672	Move	www.koch-phelps.com/	Uganda	Reduced client-server forecast	1971 Marketing / Advertising / Sales		
272444	Drill down	www.morita.com/	United Arab Emirates	Diverse next-generation firmware	2007 Machinery		
23077e	Add as new query	www.stewart.com/	Solomon Islands	Strategic even-keen definition	2018 Mining / Metals		
23f2d3	Lens	www.lloyds.com/	Tanzania	Reactive explicit task-force	1987 Motion Pictures / Film		
142821325c68	Copy as JSON	www-sabre.info/	Guam	Business-focused eco-centric firmware	1990 Religious Institutions		
142821325c69	Copy as XML	www-baner.net/	Lao People's Democratic Republic	Secured secondary help-desk	1972 Health / Fitness		
142821325c6a	Copy as CSV	www.com/	Madagascar	Reactive high-quality installation	2021 Tobacco		

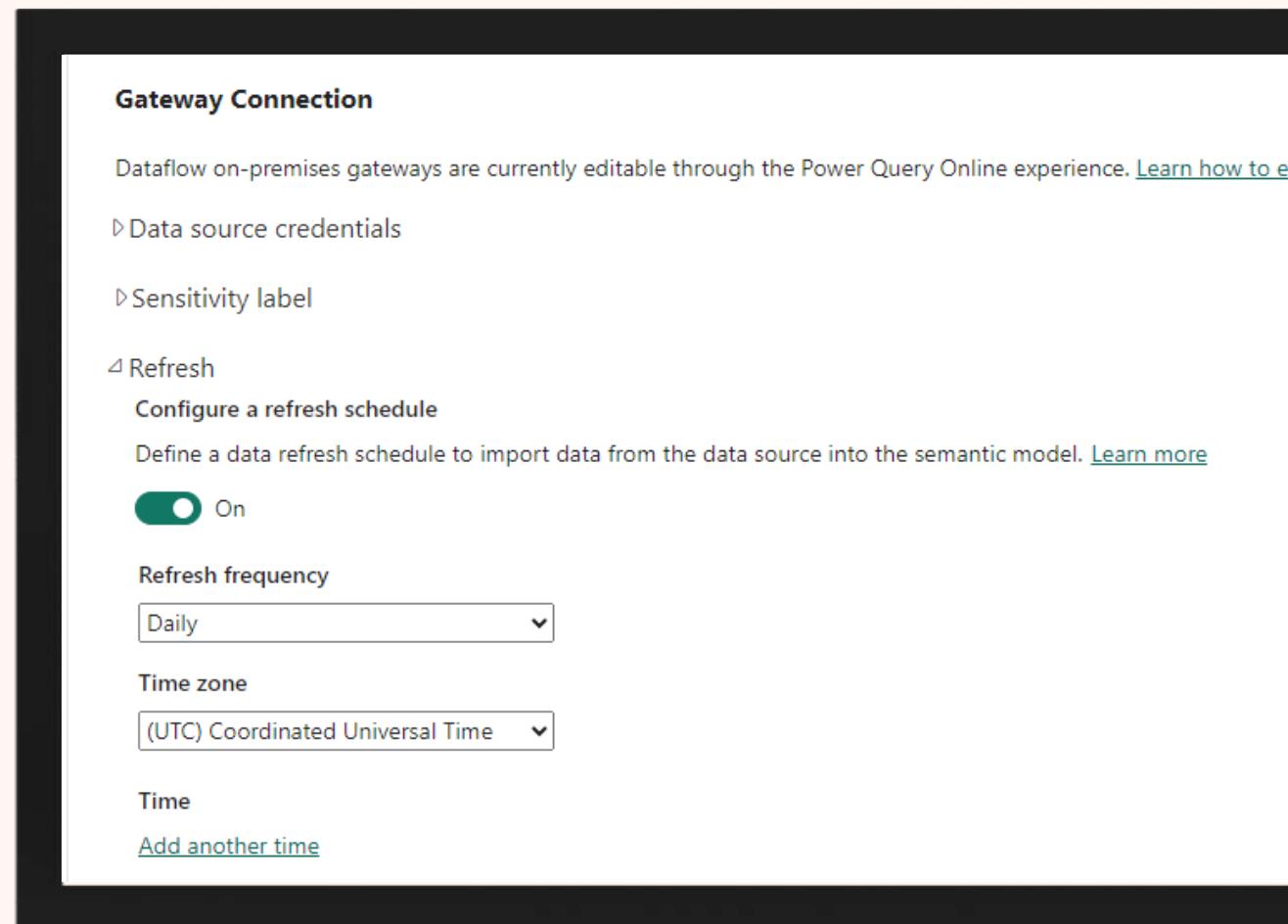
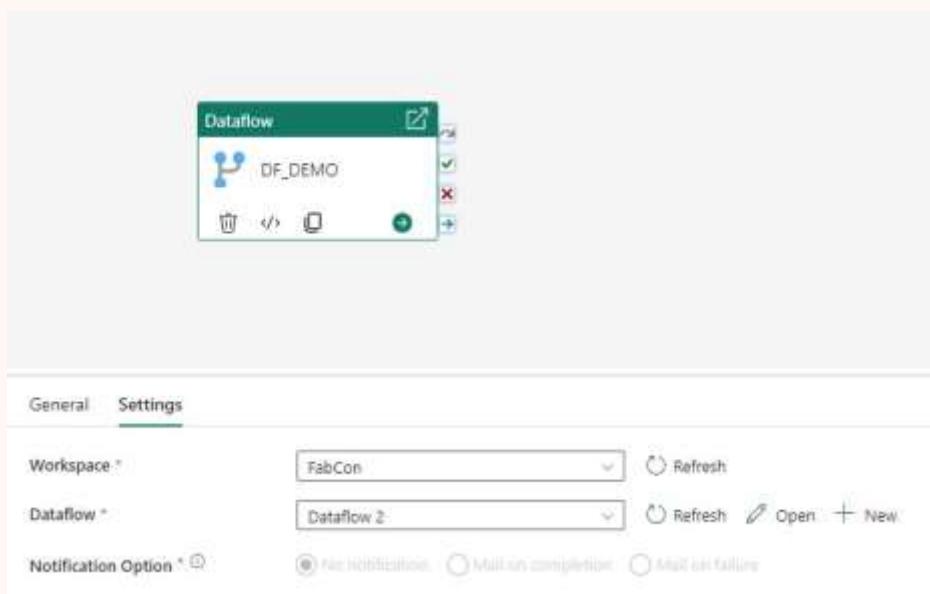
Publish Dataflows

- Select Destination
 - Lakehouse
 - Warehouse
 - Azure Data Explorer
 - Azure SQL Database
 - More to come



Refreshing Dataflows

- Schedule dataflow refreshes for automatic operation
- Execute from a pipeline

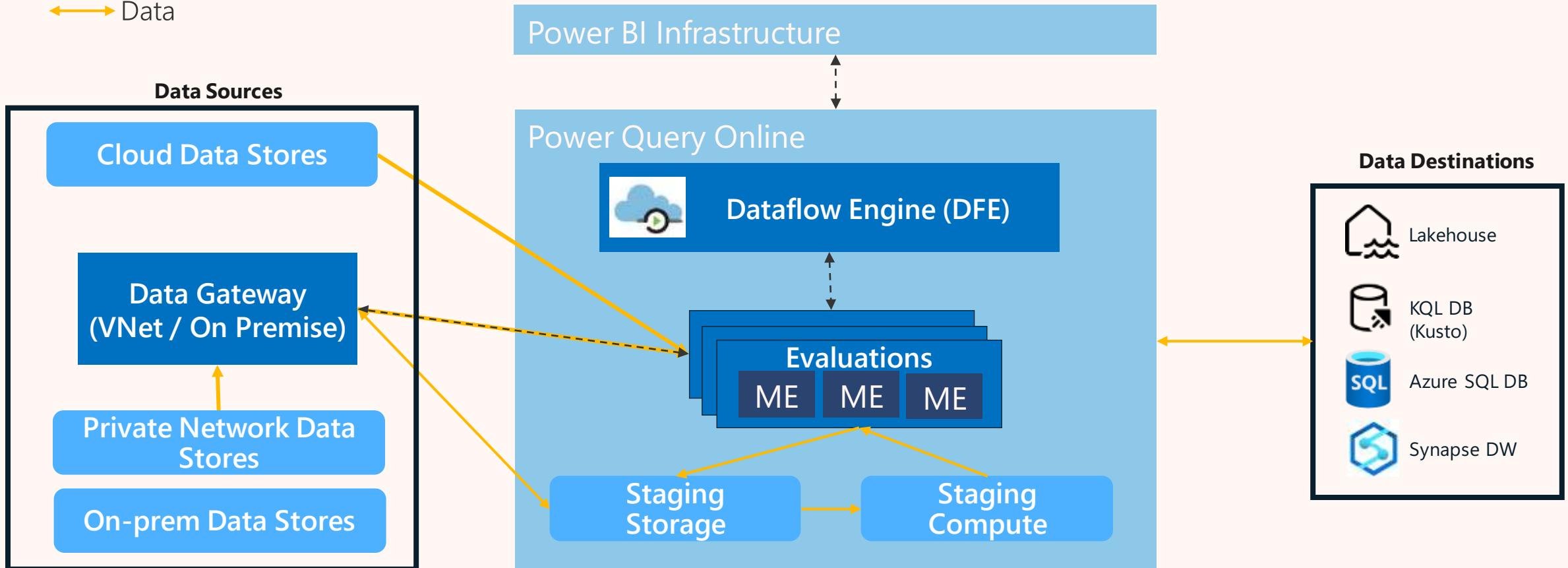


Dataflow Architecture

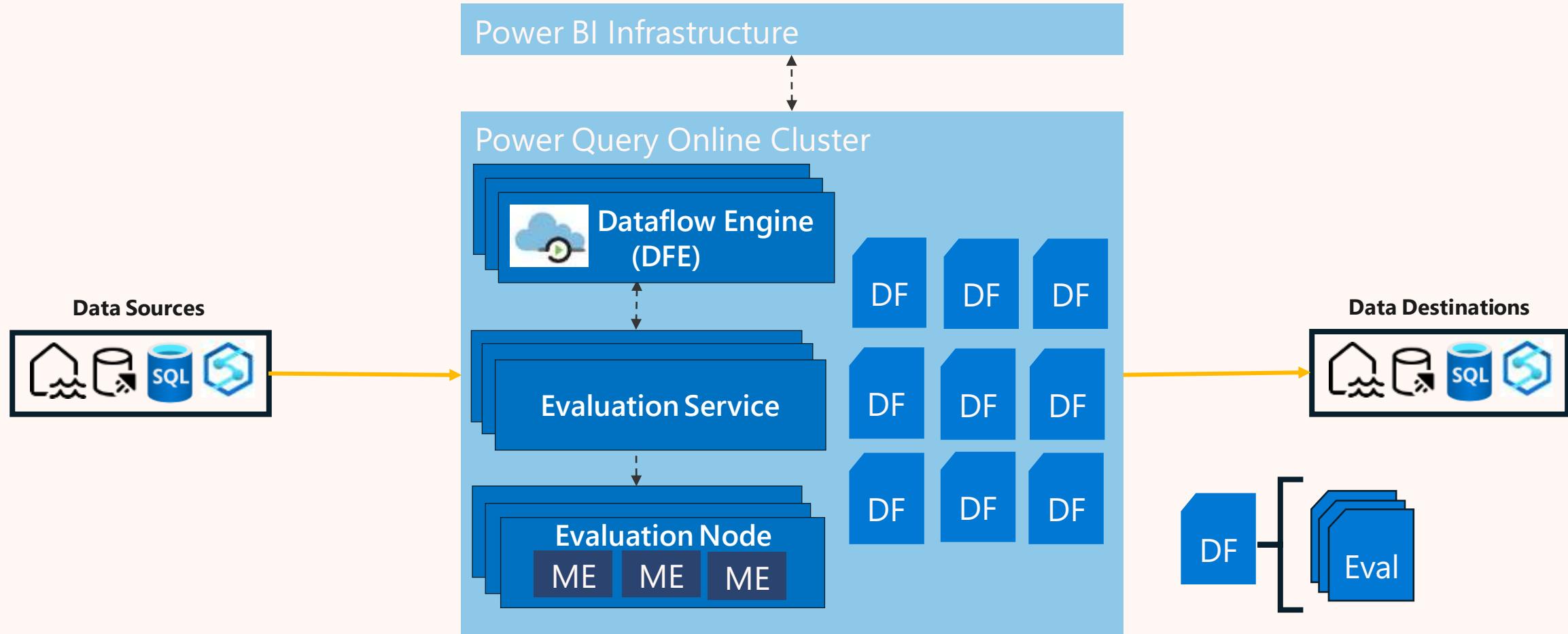
Fabric Dataflows (aka Gen2)

↔ Command and Control

↔ Data

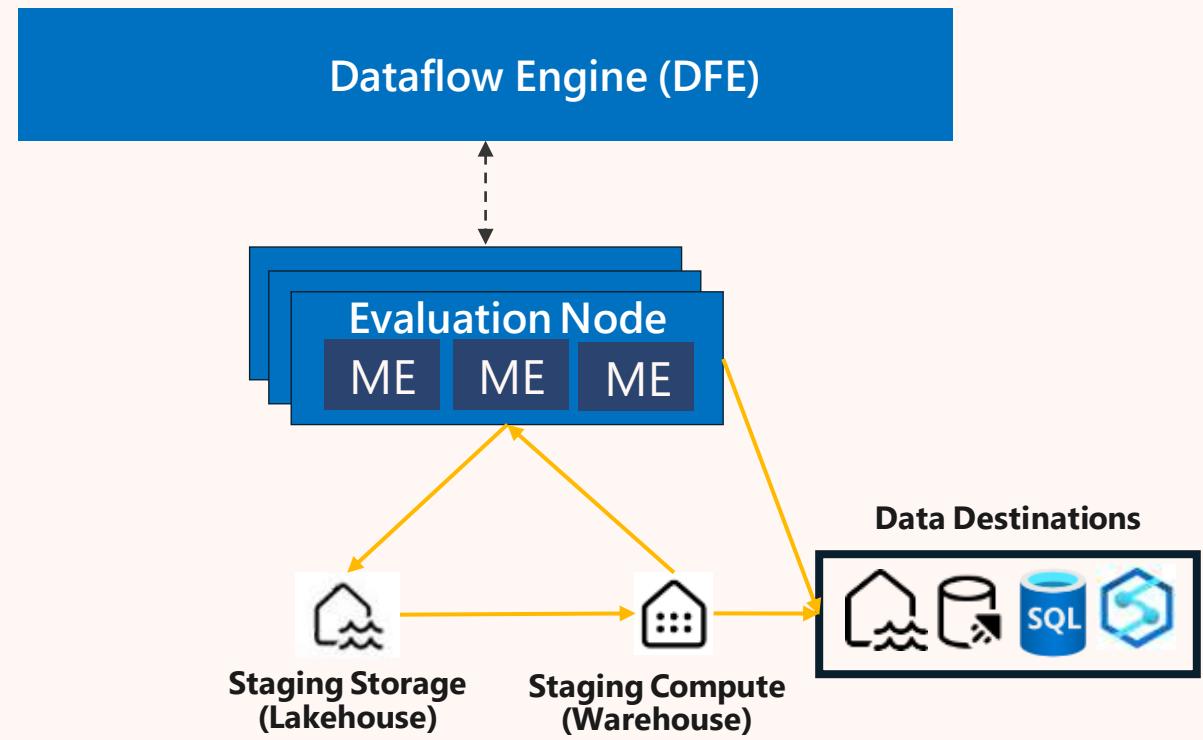


Fabric Dataflows Scale Out



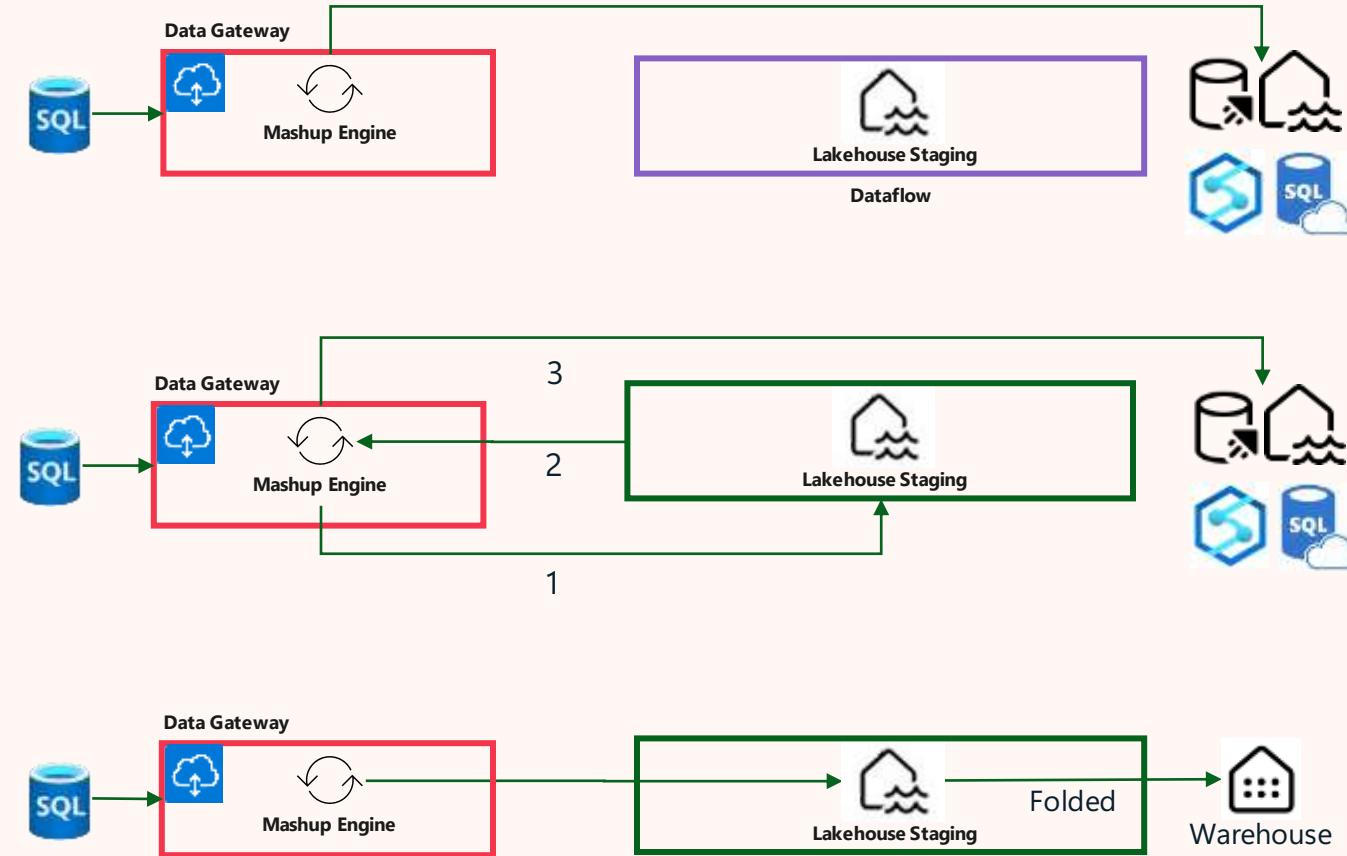
Staging

- Capabilities
 - Landing zone for incoming data
 - Provides a staging point tied to the dataflow / entity refresh
 - Provides WH Compute to speed up operations
- When to use it
 - Combining data from multiple sources
 - Stage data once for subsequent use
 - Writing to WH destination (required)
 - Merging, sorting, grouping, aggregating (uses WH compute)
- When to skip it
 - No value added for entities that write to non-WH destinations
 - See Gateway (next)



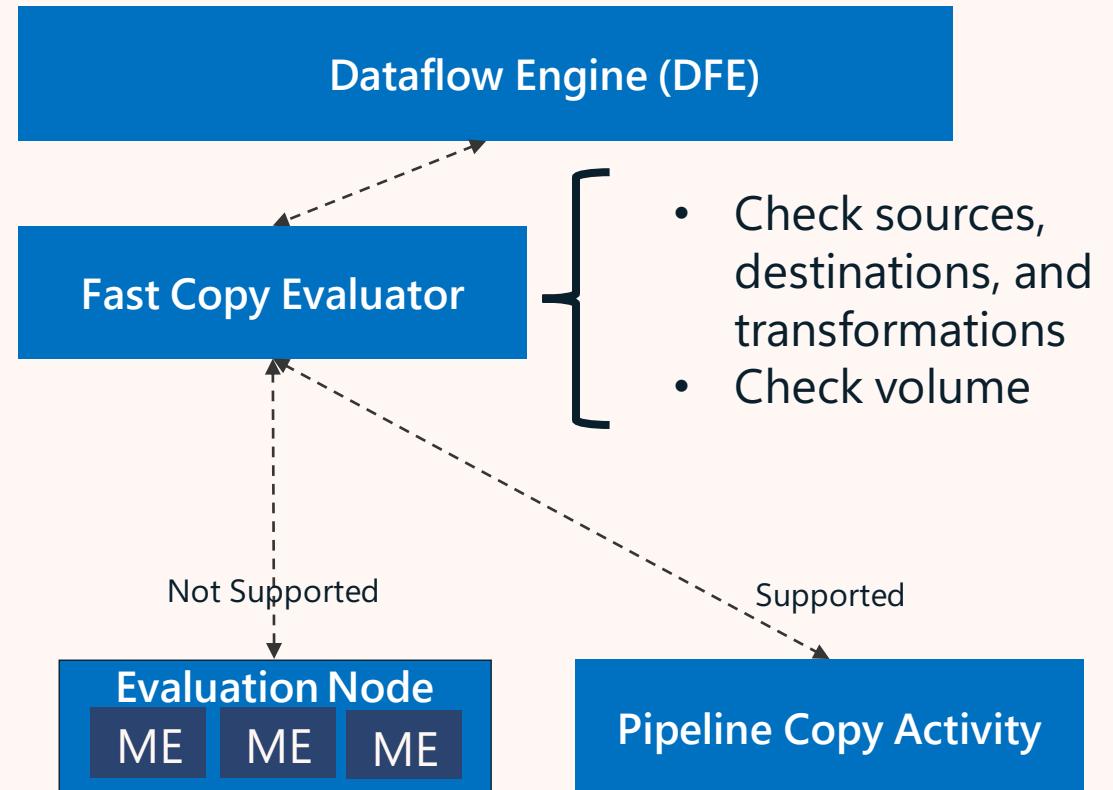
Data Gateway (VNET / On Premise)

- Capabilities
 - Enables access to data sources without exposing them to the internet
 - Data is processed locally
- When to use it
 - Data source is inside your private network
 - Move compute closer to source
 - Control where evaluations happen
- When to skip it
 - Entities that stage data and write to destination (excluding WH)
 - Split Ingest and Destination into separate dataflows (consider where to Transform)
 - Don't stage



Fast Copy

- Capabilities
 - Leverages Pipeline Copy Activity for large performance boost in ingest
 - Automatically used based on pattern matching and volume
 - Transparent (no pipeline to manage)
- When to use it
 - Whenever possible
 - Defer transformations to post ingest if they affect Fast Copy use
 - Enable in Options..Scale..Allow use of fast copy connectors
- When to skip it
 - Don't – if it's an option, use it
 - Mark as "Require fast copy" to enforce



Leveraging the Architecture

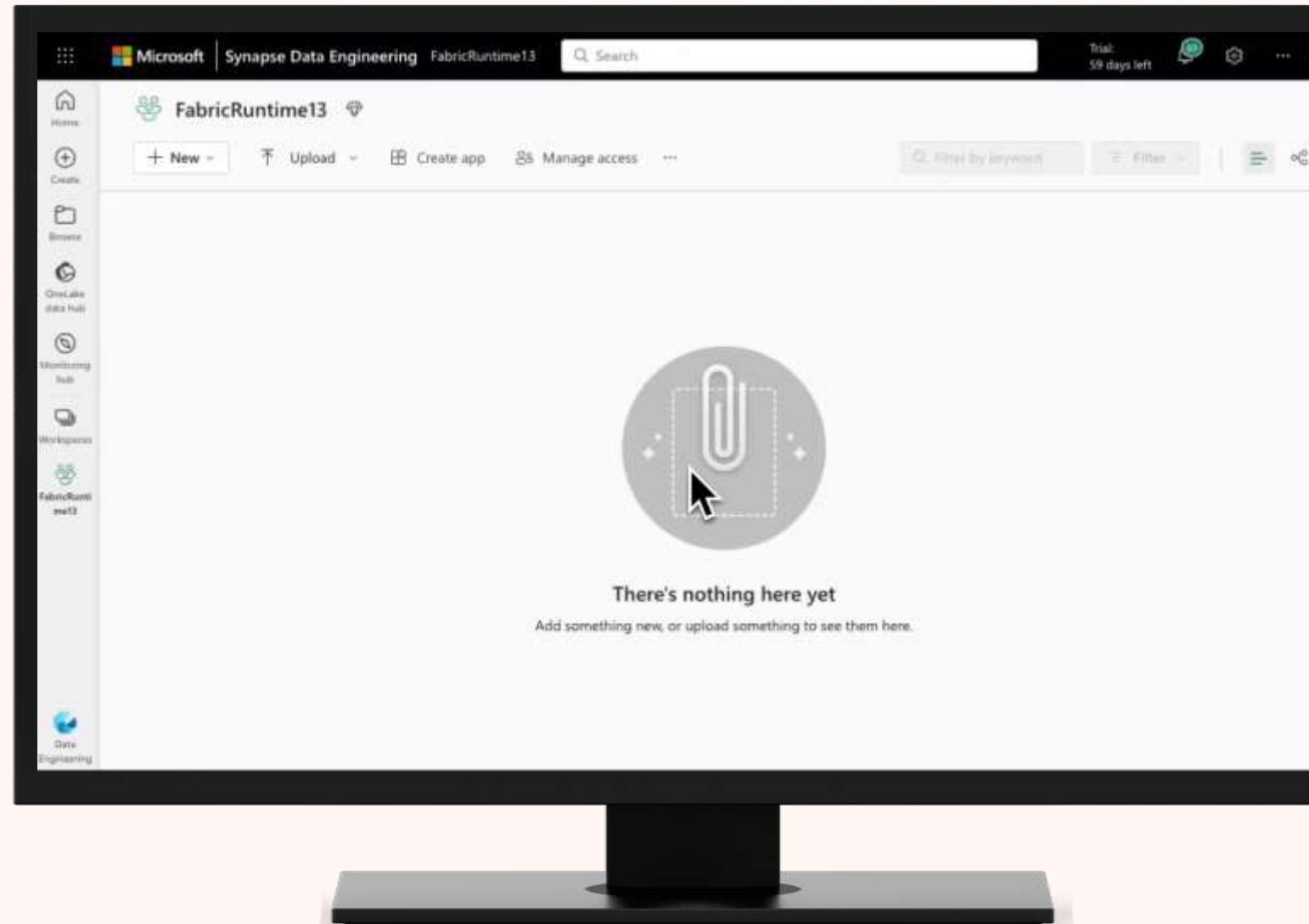
- Carefully consider ETL vs ELT when laying out dataflows
- Know why you are using staging
 - Accelerates some operations, but adds no value to others
- Avoid “double-hops” with Data Gateways
- Take advantage of Fast Copy wherever possible
 - As patterns are added, dataflows can automatically benefit
 - Enable “use fast copy connectors”
 - Ensure that queries fully fold to maximize fast copy usage
- Build for parallel processing
 - Evaluations, dataflows
 - Incremental refresh when available

Schedule data pipelines,
dataflows and notebooks

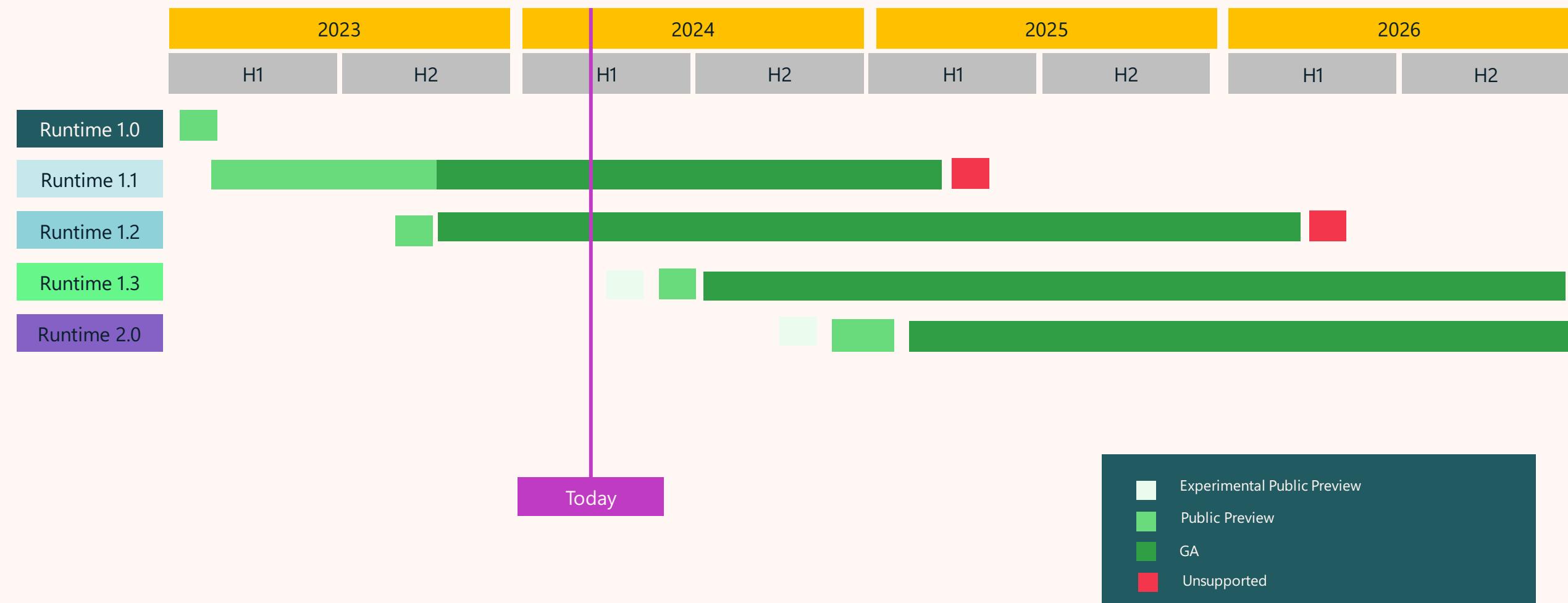
Spark Runtime

Fabric Spark Runtime

- Runtime 1.1: Spark 3.3 and Delta 2.2
- Runtime 1.2: Spark 3.4 and Delta 2.4
- Experimental Runtime: Spark 3.5 and Delta 3.0
- All Fabric artifacts automatically write Delta with V-Order optimization providing lightning-fast analytics using DirectLake in Power BI



Evolution of Fabric Runtimes



Packages and Versions Across Fabric Runtimes

	Runtime 1.1	Runtime 1.2	Runtime 1.3	Runtime 2.0
Apache Spark	3.3.1 -> 3.3.4	3.4.1 -> 3.4.2	3.5.0 -> 3.5.X	4.X
Operating System	Ubuntu 18.04	Mariner 2.0	Mariner 2.0	Mariner 3.0
Java	8	11	11	17
Scala	2.12.15	2.12.17	2.12.17	2.13
Python	3.10	3.10	3.11	3.12
Delta Lake	2.2	2.4	3.0	4.X
R	4.2.2	4.2.2	4.2.2	4.X.Y

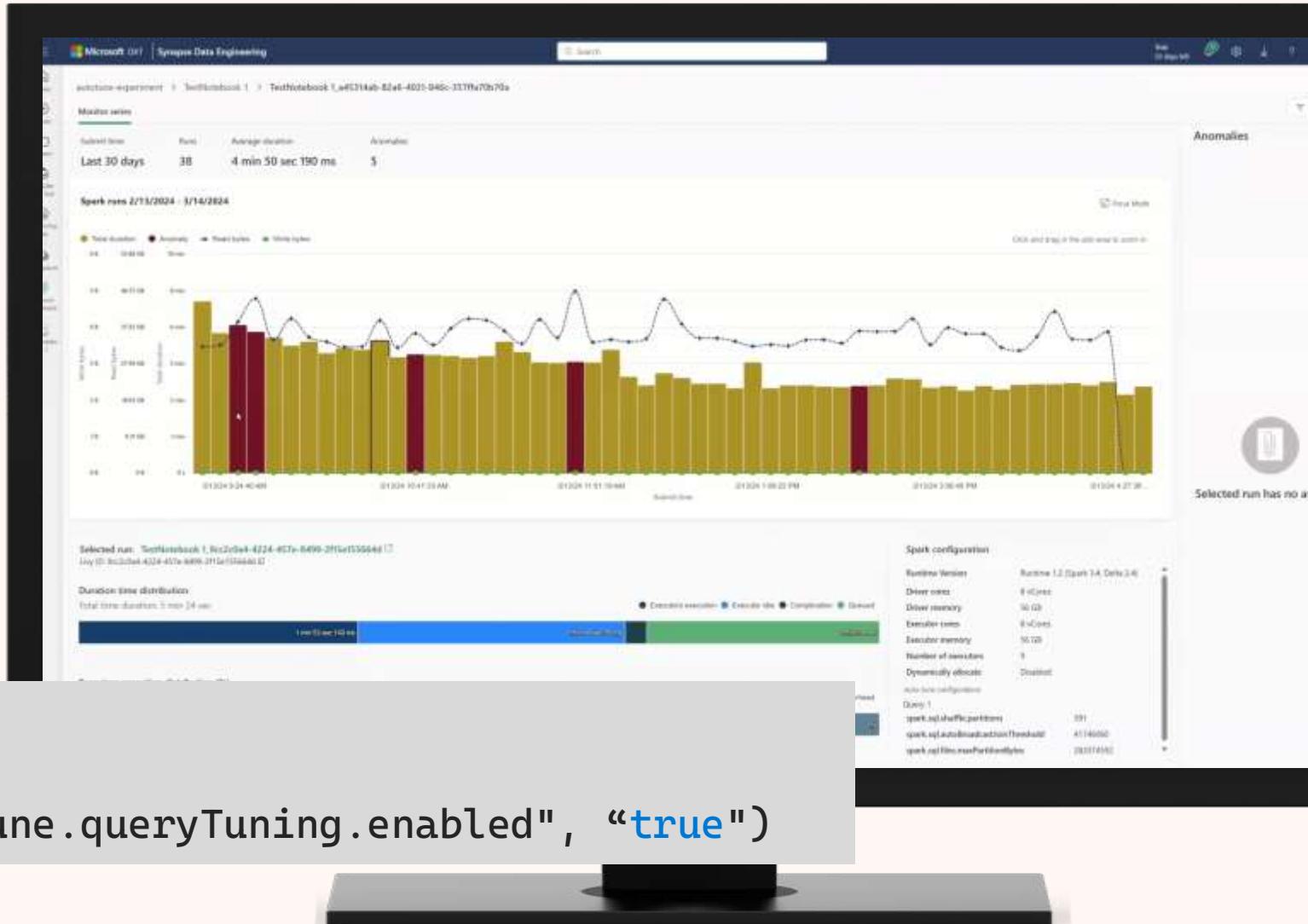
Spark Autotune

Experience a significant boost in performance

- Autotune configures three query-level Spark configurations:
 - ✓ `spark.sql.shuffle.partitions`
 - ✓ `spark.sql.autoBroadcastJoinThreshold`
 - ✓ `spark.sql.files.maxPartitionBytes`
- Enable autotune with

Scala

```
spark.conf.set("spark.ms.autotune.queryTuning.enabled", "true")
```



Low Shuffle Merge

- Merge into command optimization enabled by default in Fabric
- Reducing amount of shuffle data for join
- Efficient when each underlying file size is large and MERGE updates only few rows in the file
- For large size job merge (newly added file size > 1GB), improved performance by 2-5x
- For small size job merge (newly added file size < 1GB), improved performance by 1-2x

Scala

```
spark.conf.set("spark.microsoft.delta.merge.lowShuffle.enabled", "true")
```

SQL

SET

```
spark.microsoft.delta.merge.lowShuffle.enabled = true
```

Optimize Write

- Helps resolve many small files problem by shuffling data before writing
- Optimize Write is enabled by default in Fabric
- Targeting 1GB file size by default for VORDER query efficiency resulting in blazing fast performance when reading the tables from PowerBI or DW engines
- Write performance gain expected when a write job writing into many partition directories

Scala

```
spark.conf.set("spark.microsoft.delta.optimizeWrite.enabled", "true")
```

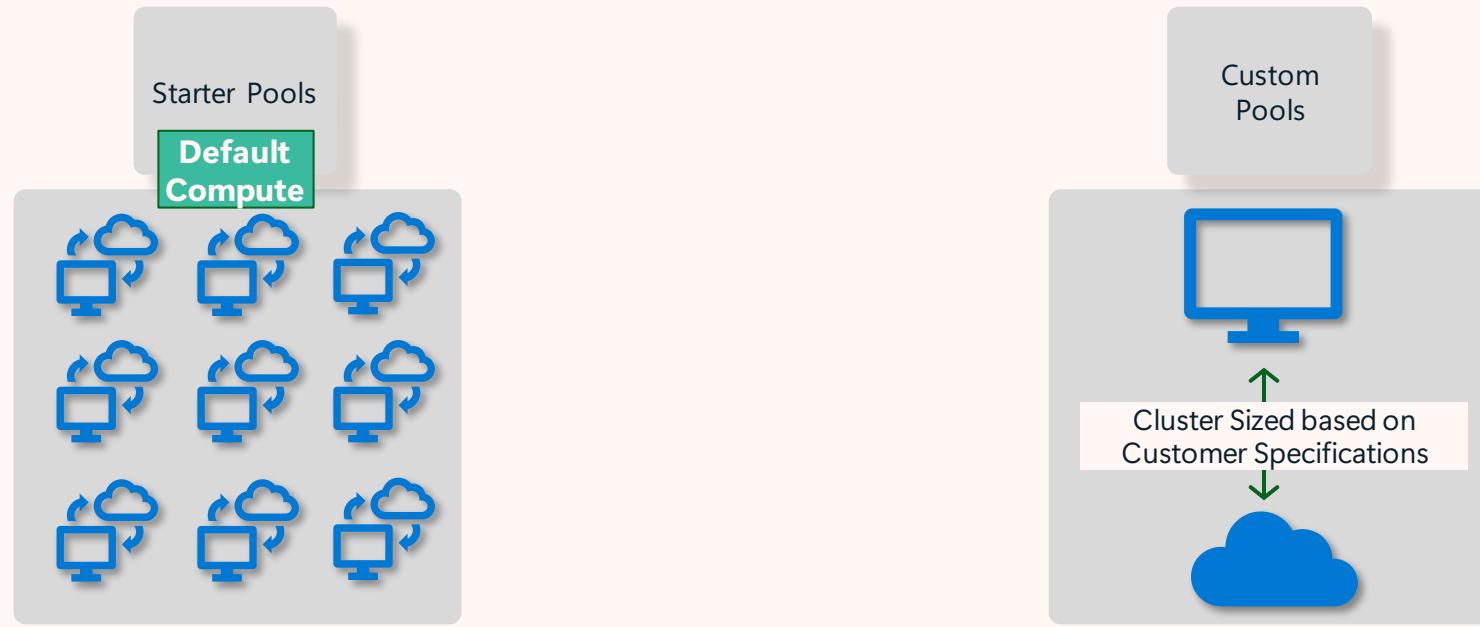
SQL

SET

```
spark.microsoft.delta.optimizeWrite.enabled = false
```

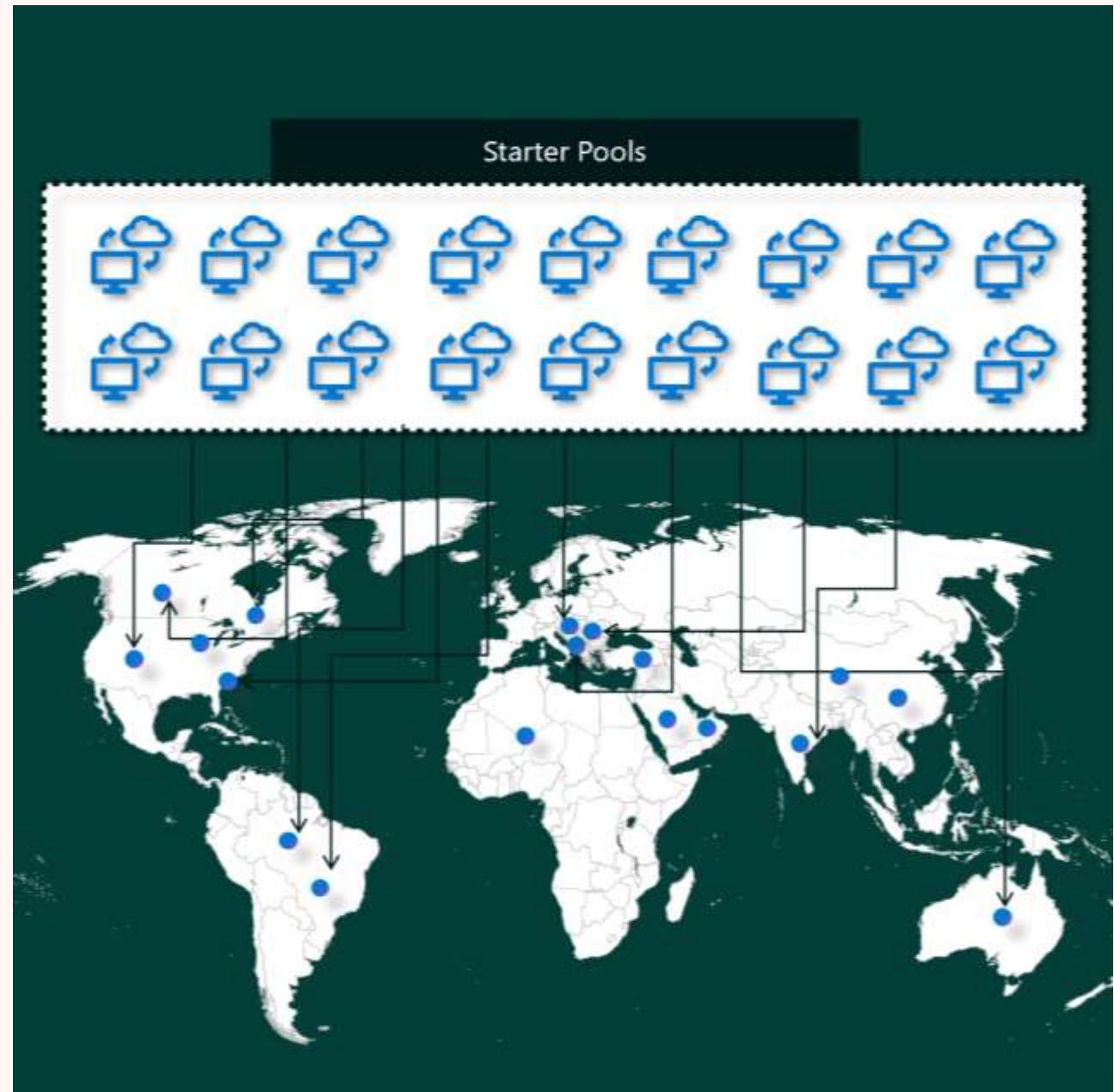
Spark Compute

- Spark Compute is a powerful and flexible way of using Spark on Microsoft Fabric.
- It enables you to perform various data engineering and data science tasks on your data stored in OneLake or other sources



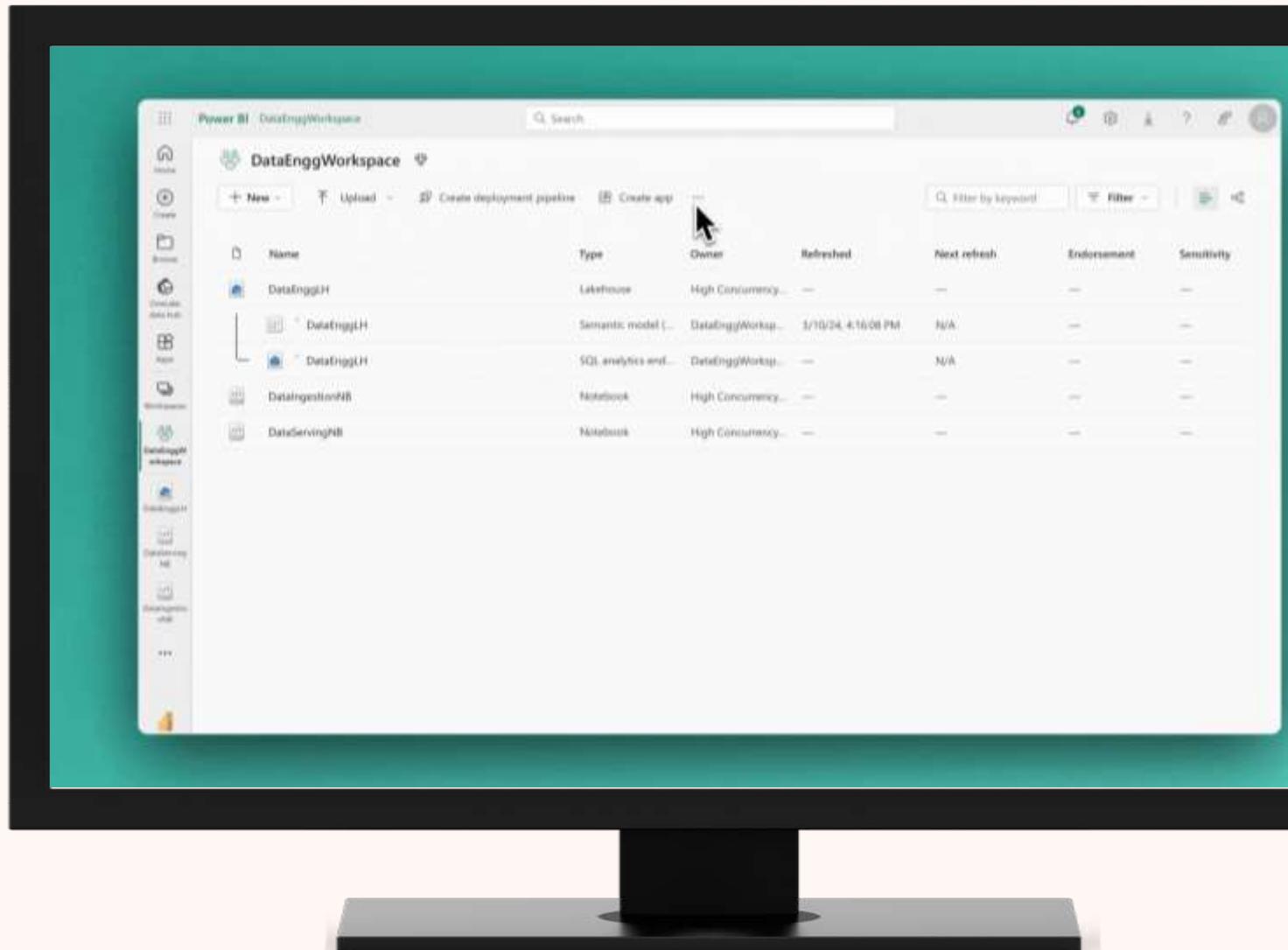
Starter Pools

- Starter pools are pre-hydrated clusters provide a spark jobs and sessions within 5 seconds
- Clusters in the Starter Pools are constantly rehydrated as they are assigned to incoming requests from users
- For workloads with smaller compute requirements - Users can set the pools to max node of 1 and still get the 5 Second session start experience.

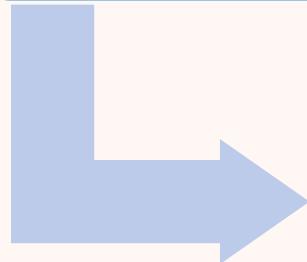


Custom Pools

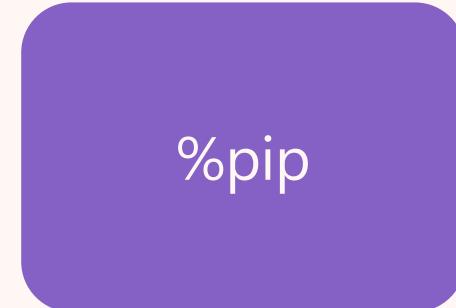
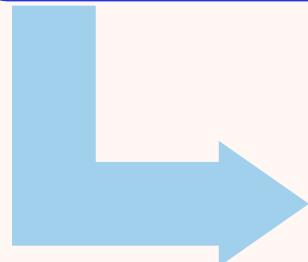
- Custom Pools can be configured by workspace or capacity admins in Fabric allow customized compute options with different node sizes
- Pools created by admins can be selected at the workspace level or in an environment
- These spark clusters will take ~3 minutes for starting a spark session as they are acquired on-demand during job submission



Session Customization



Customization across different notebooks or spark job definitions

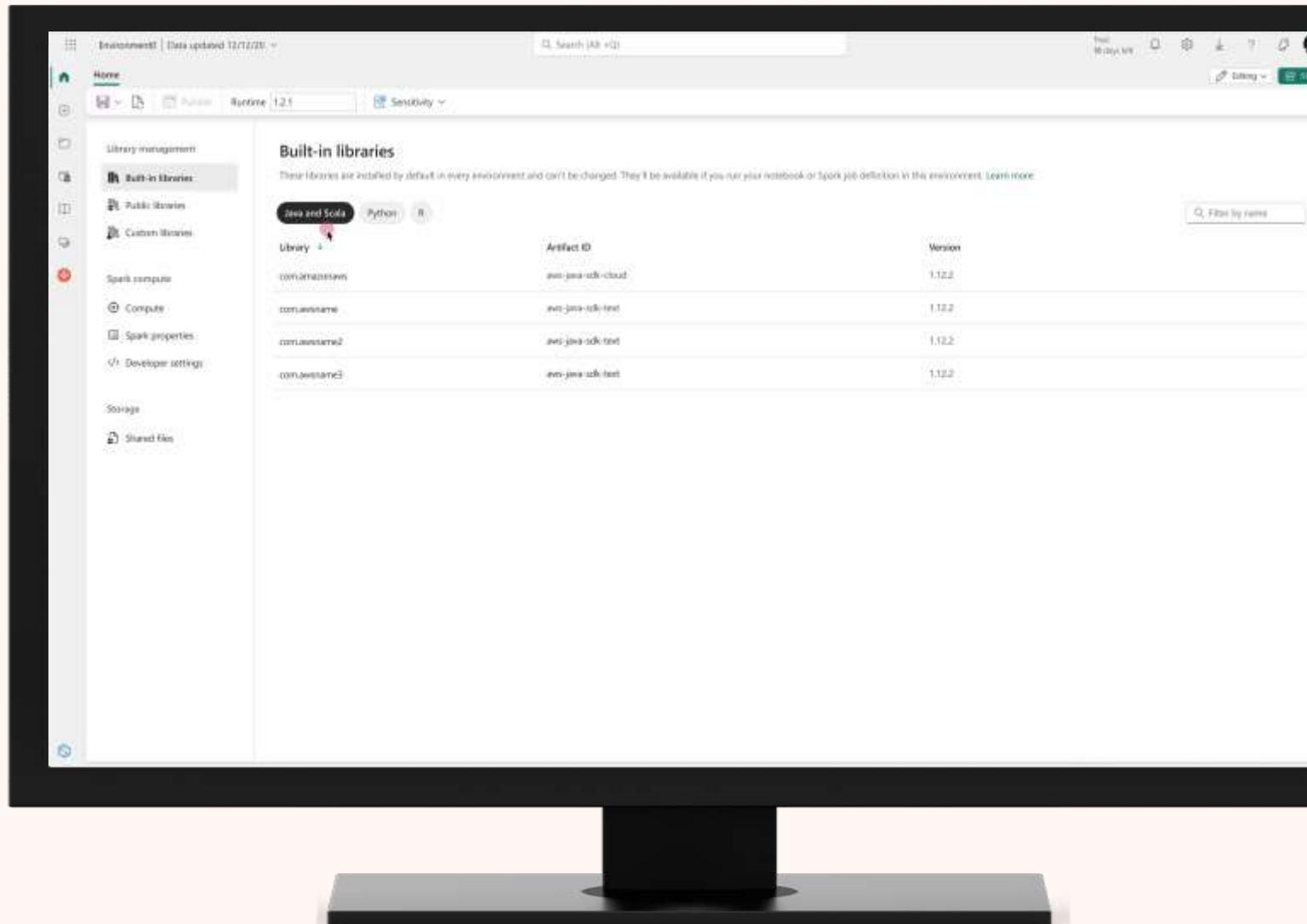


Customization before starting a Spark application

Customizations within a Spark session

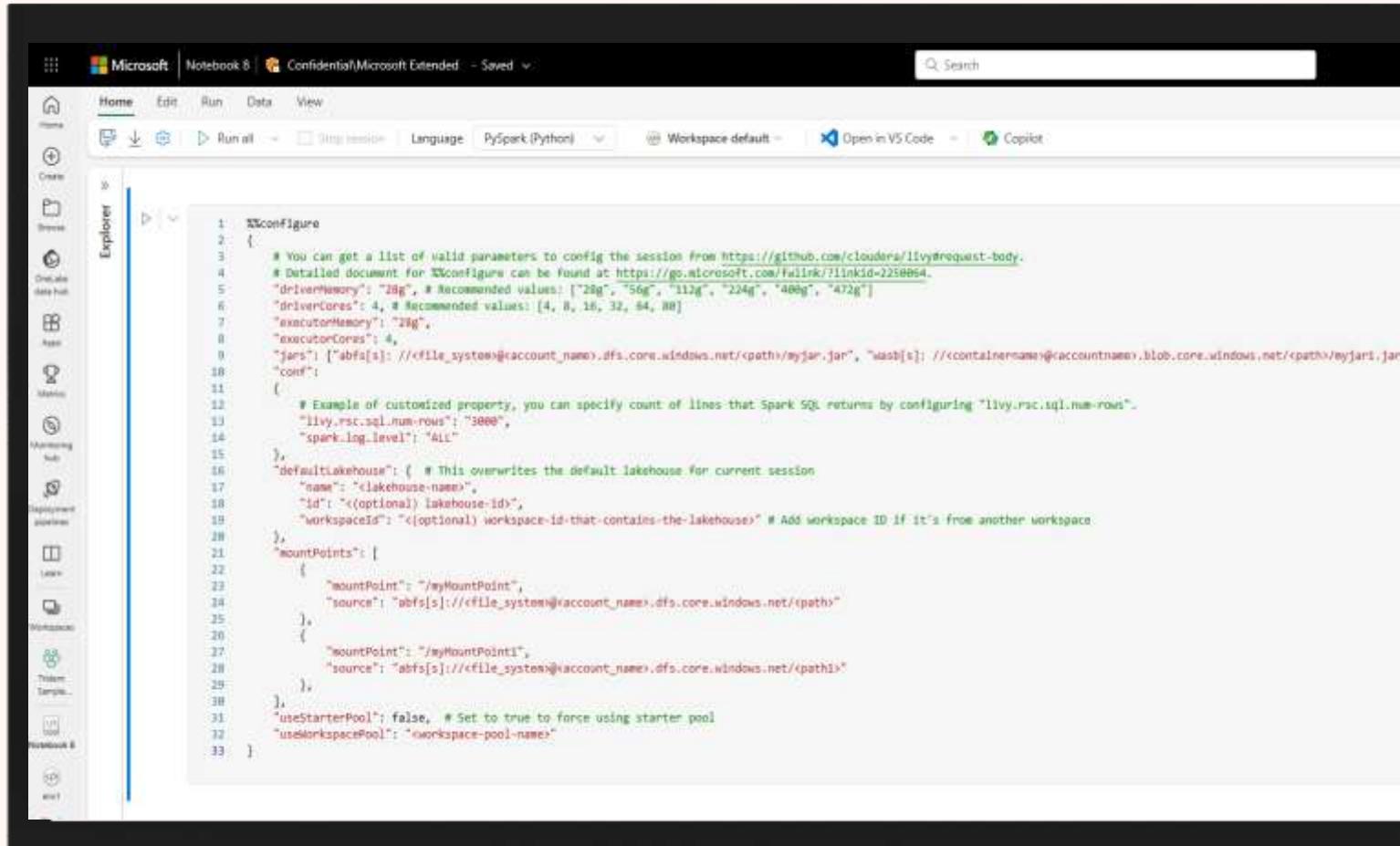
Customization with Environments and Library Management

- The environment artifact allows users to configure all their Spark and library related settings in a single place
- Users can do the following (subject to the permissions they have been granted):
 - Install libraries
 - Choose from a pre-defined user pool
 - Set Spark properties



Customization using %%configure

- Customize the driverMemory, driverCores, executorMemory, executorCores
- Configure mounts, and default lakehouse and file-based dependencies
- Configure spark properties for your sessions
- Install jars to your Spark sessions

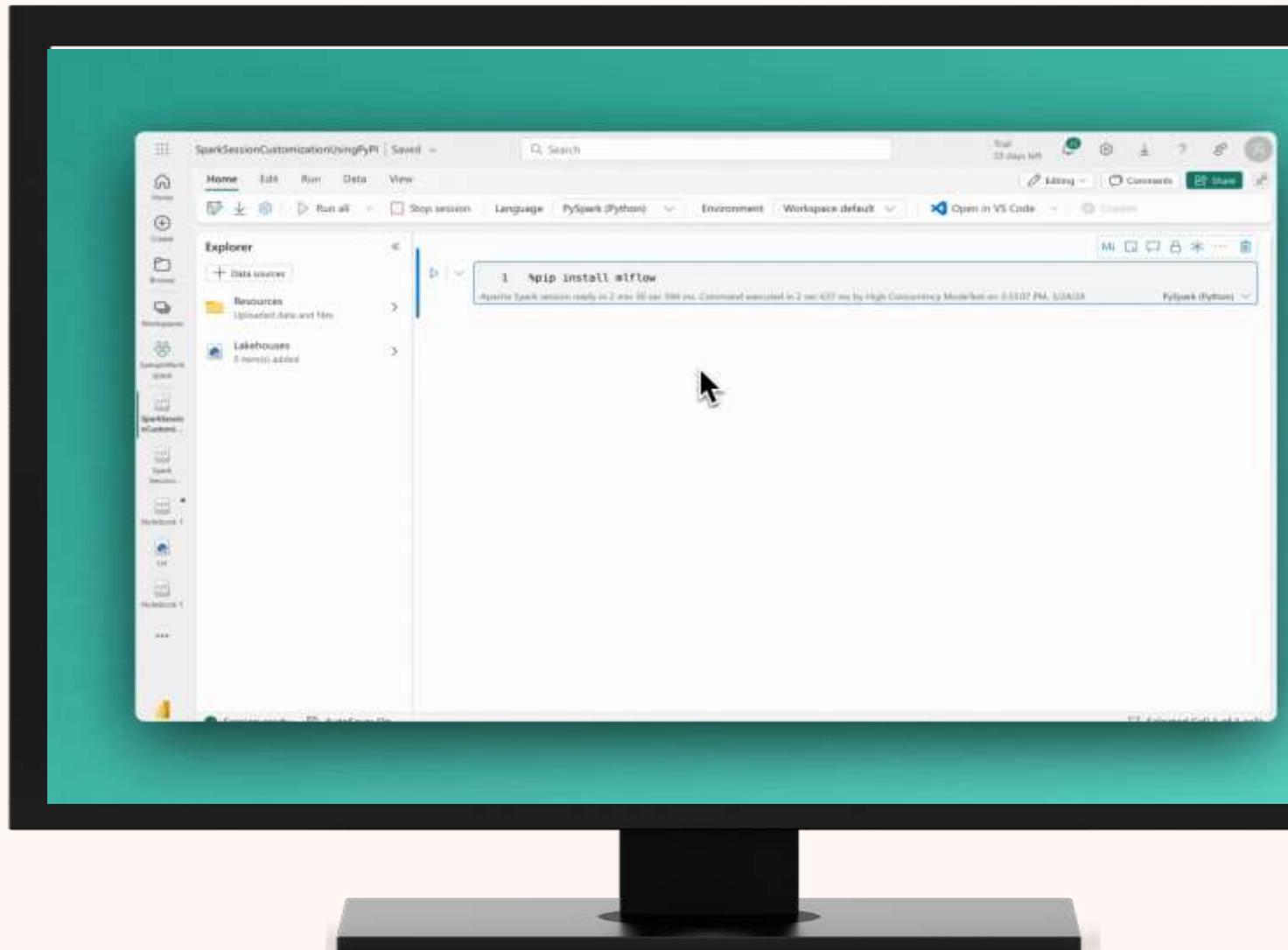


The screenshot shows a Microsoft Fabric Notebook interface with the title "Notebook 8 | ConfidentialMicrosoft Extended - Saved". The code editor displays the following %%configure block:

```
1: %%configure
2: {
3:     # You can get a list of valid parameters to config the session from https://github.com/cloudera/livy#request-body.
4:     # Detailed document for %%configure can be found at https://go.microsoft.com/fwlink/?linkid=2250064.
5:     "driverMemory": "28g", # Recommended values: ["28g", "56g", "112g", "224g", "448g", "872g"]
6:     "driverCores": 4, # Recommended values: [4, 8, 16, 32, 64, 88]
7:     "executorMemory": "28g",
8:     "executorCores": 4,
9:     "jars": ["abfs[s]://<file_system>@account_name.dfs.core.windows.net</path>/myjar.jar", "wasb[s]://<containername>@accountname.blob.core.windows.net</path>/myjar.jar"],
10:    "conf":
11:    {
12:        # Example of customized property, you can specify count of lines that Spark SQL returns by configuring "livy.rsc.sql.num-rows".
13:        "livy.rsc.sql.num-rows": "3888",
14:        "spark.log.level": "ALL"
15:    },
16:    "defaultLakehouse": { # This overwrites the default lakehouse for current session
17:        "name": "<lakehouse-name>",
18:        "id": "<(optional) lakehouse-id>",
19:        "workspaceId": "<(optional) workspace-id-that-contains-the-lakehouse>" # Add workspace ID if it's from another workspace
20:    },
21:    "mountPoints": [
22:        {
23:            "mountPoint": "/myMountPoint",
24:            "source": "abfs[s]://<file_system>@account_name.dfs.core.windows.net</path>"
25:        },
26:        {
27:            "mountPoint": "/myMountPoint1",
28:            "source": "abfs[s]://<file_system>@account_name.dfs.core.windows.net</path>"
29:        }
30:    ],
31:    "useStarterPool": false, # Set to true to force using starter pool
32:    "useWorkspacePool": "<workspace-pool-name>"
33: }
```

Customization using %pip

- Install library packages within active sessions
- %pip is recommended instead of !pip
 - As !pip will only install packages on the driver node
 - Packages that install through !pip will not affect when conflicts with built-in packages or when it's already imported in a notebook.

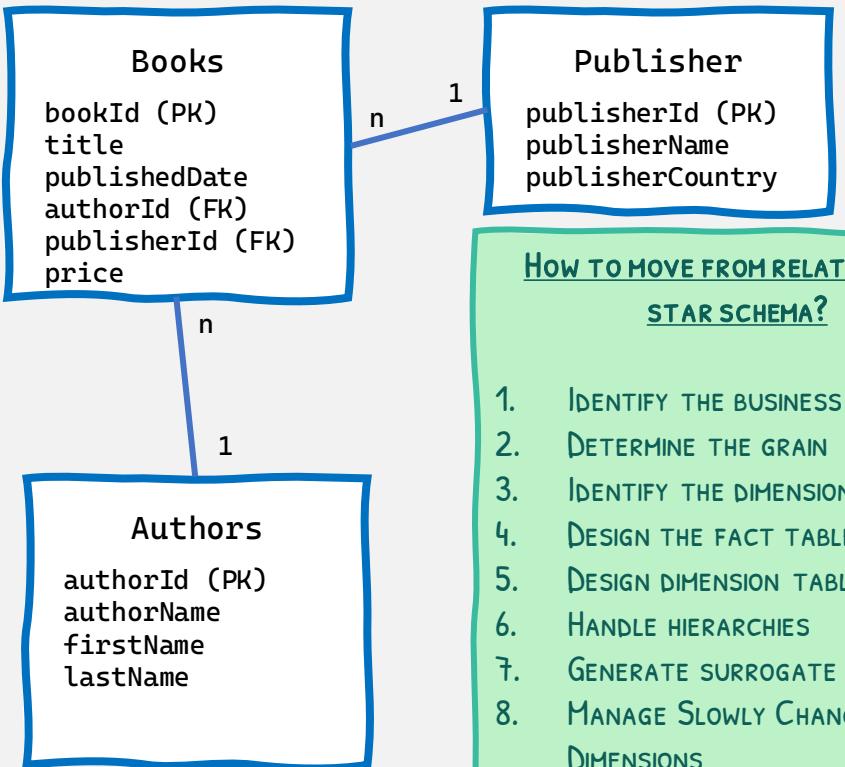


DATAWAREHOUSE DATABASE DESIGN

FROM RELATIONAL TO STAR SCHEMA



RELATIONAL SCHEMAS ARE OPTIMIZED FOR NORMALIZATION
(=REDUCE DATA REDUNDANCY) AND SPEED OF EXECUTION FOR TRANSACTIONAL QUERIES



HOW TO MOVE FROM RELATIONAL TO STAR SCHEMA?

1. IDENTIFY THE BUSINESS PROCESS
2. DETERMINE THE GRAIN
3. IDENTIFY THE DIMENSIONS
4. DESIGN THE FACT TABLE
5. DESIGN DIMENSION TABLES
6. HANDLE HIERARCHIES
7. GENERATE SURROGATE KEYS
8. MANAGE Slowly CHANGING DIMENSIONS
9. OPTIMIZE, TEST, DOCUMENT

PK: PRIMARY KEY. FK: FOREIGN KEY.

NOTE: THIS SCHEMA IS SIMPLISTIC ON PURPOSE
(E.G. A BOOK COULD BE AUTHORED BY MULTIPLE AUTHORS)

STAR SCHEMAS ARE OPTIMIZED FOR ANALYTICAL QUERY PERFORMANCES

FactBook

FactBookId(SK)
bookId (AK)
title
DimDateId (SK)
publishedDate
DimAuthorId (SK)
authorName
authorFirstName
authorLastName
DimPublisherId (SK)
publisherName
publisherCountry

SURROGATE KEYS (SK) IS A UNIQUE IDENTIFIER FOR EACH RECORD IN THE DATAWAREHOUSE.
ACTS AS PRIMARY KEY

ALTERNATE KEYS (AK) MAINTAIN A CONNECTION TO THE ORIGINAL DATA SOURCES

FACT TABLES CONTAINS THE MEASUREMENT OF "FACT" THAT YOU WANT TO ANALYZE

DimAuthor

- DimAuthorID (SK)
- authorId (Ak)
- authorName
- firstName
- lastName

DIMENSION TABLES CONTAINS ATTRIBUTES THAT ARE USED TO FILTER, GROUP OR LABEL DATA CONTAINED IN FACT TABLES

DimDate

DimDateKey
FullDate
DayOfWeek
DayOfMonth
MonthNumber
MonthName
Quarter
Year

DimPublisher

DimPublisherId (SK)
publisherId (AK)
publisherName
publisherCountry

Most of star schema have a "DATE" dimension that is common across all entities

3



Module 3
**Implement and manage a
semantic model**

Implement and manage semantic models (20–25%)

Design and build semantic models

Choose a storage mode, including Direct Lake

Identify use cases for DAX Studio and Tabular Editor 2

Implement a star schema for a semantic model

Implement relationships, such as bridge tables and many-to-many relationships

Write calculations that use DAX variables and functions, such as iterators, table filtering, windowing, and information functions

Implement calculation groups, dynamic strings, and field parameters

Design and build a large format dataset

Design and build composite models that include aggregations

Implement dynamic row-level security and object-level security

Validate row-level security and object-level security

Optimize enterprise-scale semantic models

Implement performance improvements in queries and report visuals

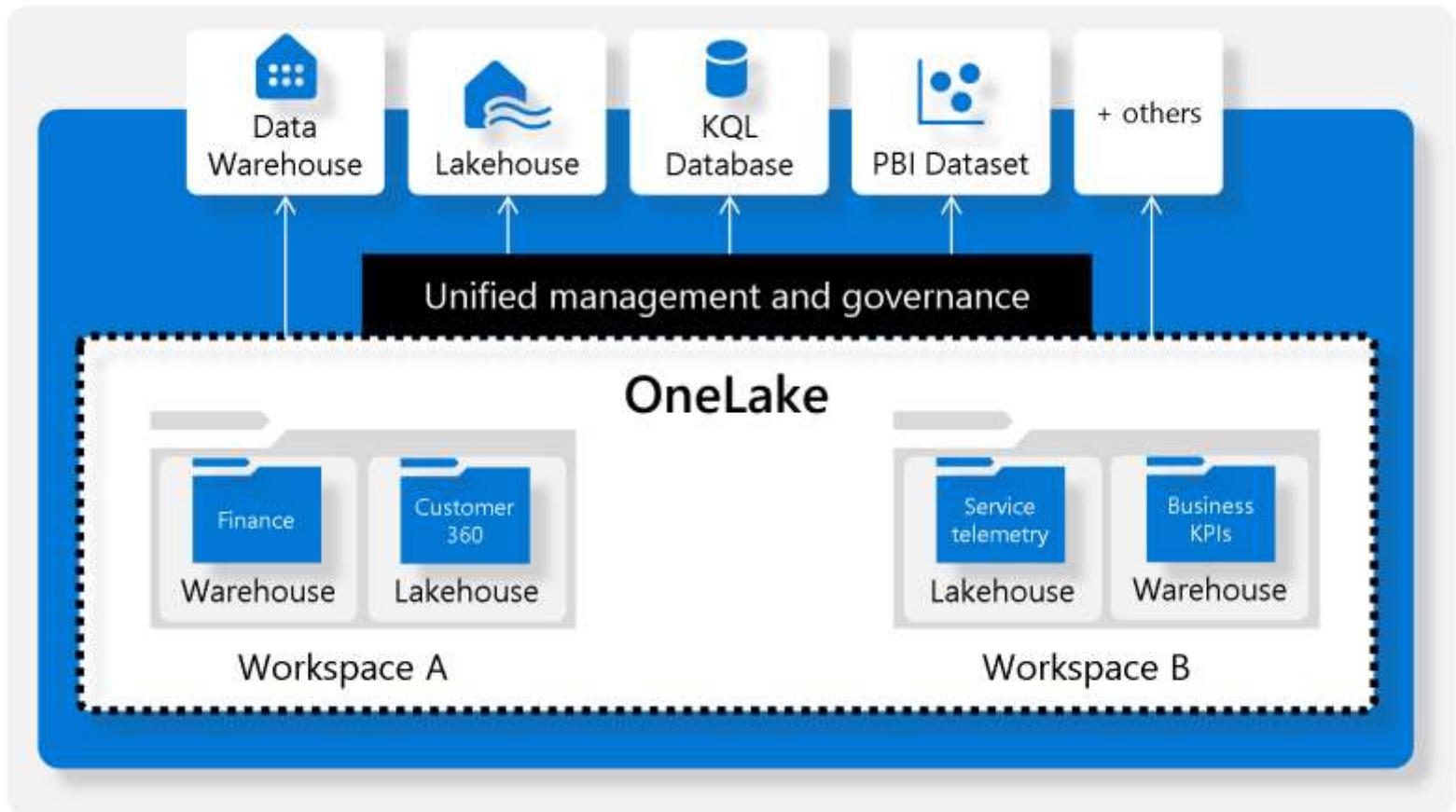
Improve DAX performance by using DAX Studio

Optimize a semantic model by using Tabular Editor 2

Implement incremental refresh

Understand Fabric warehouses

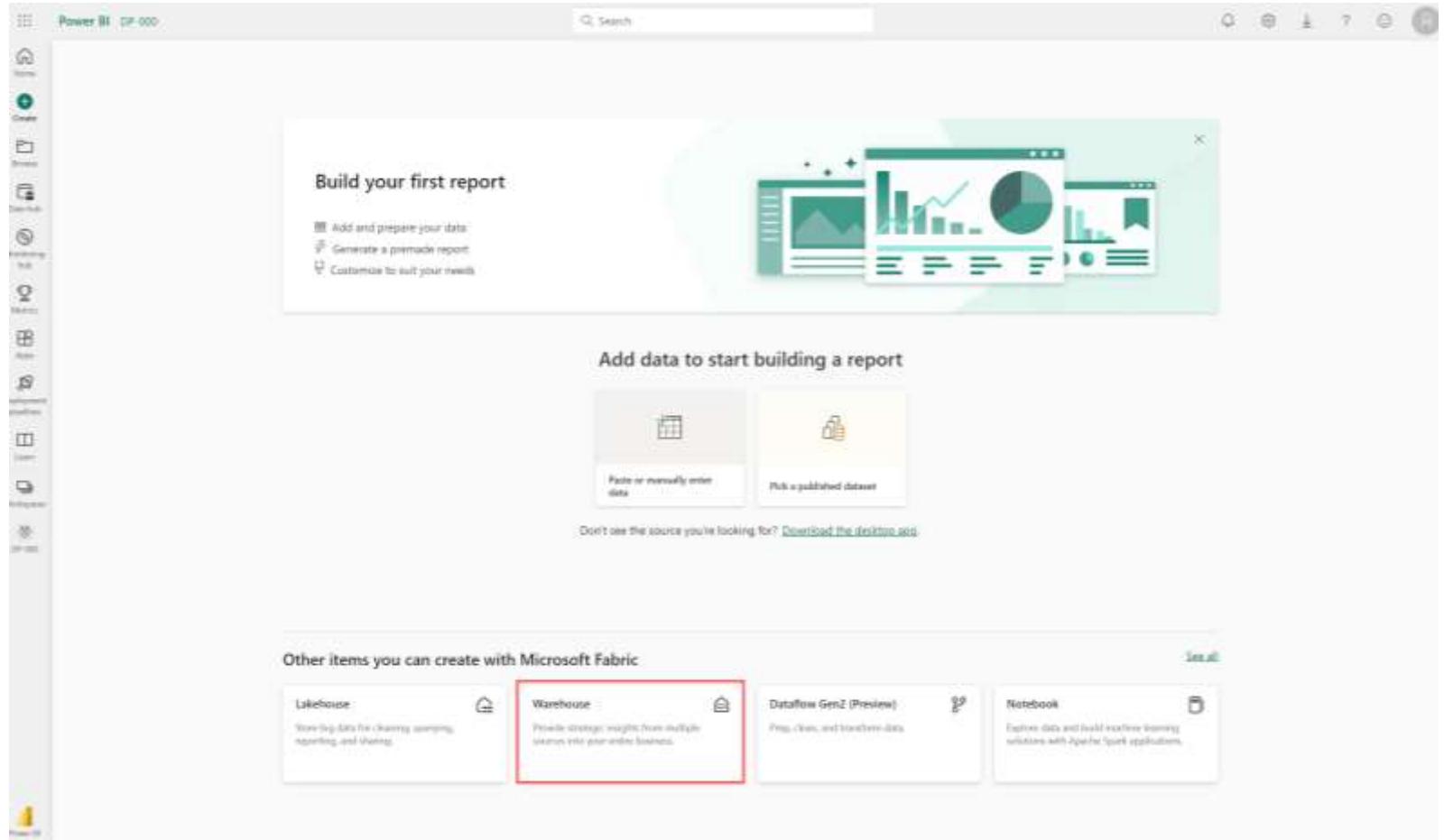
- Centered on single data lake
- Powered by Synapse Analytics
- Fully supports T-SQL
- Parquet file format



Create a data warehouse in Fabric

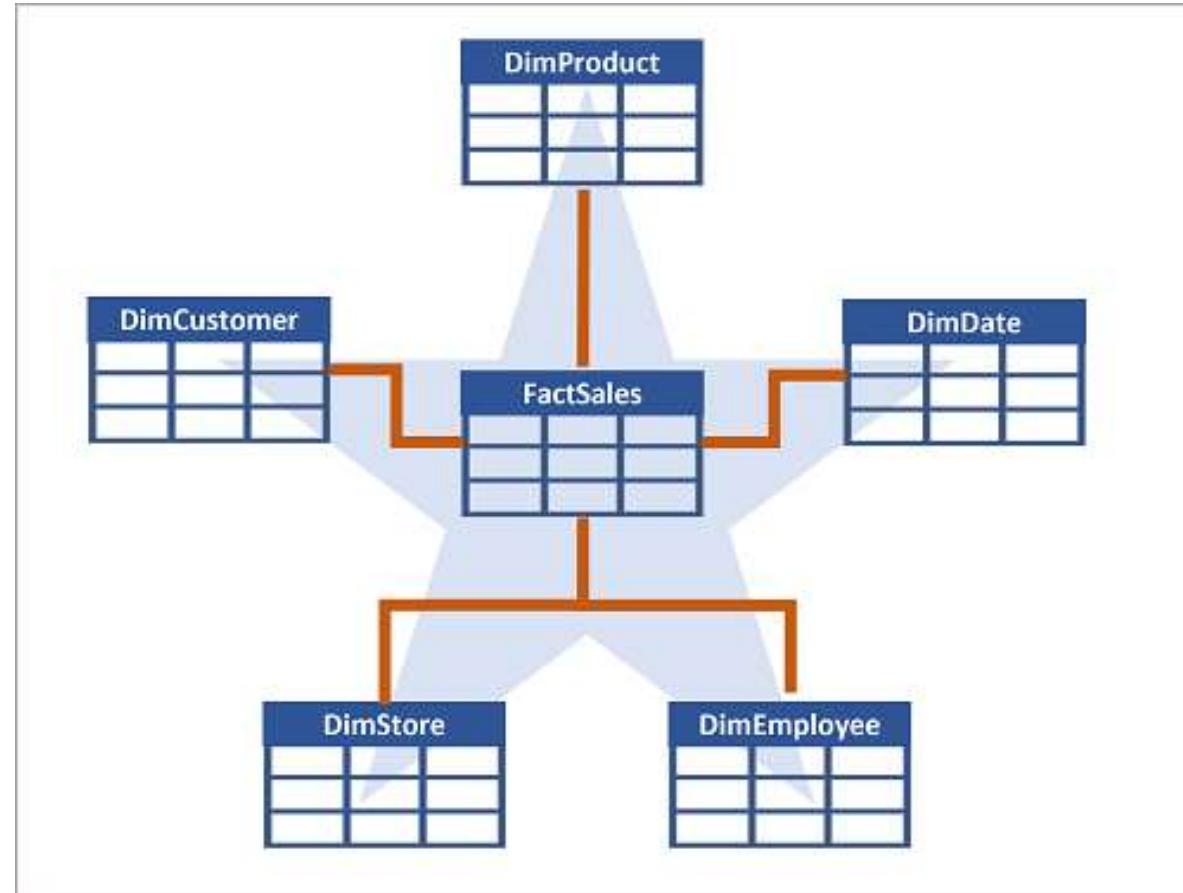
Within Fabric service portal

- Navigate to Fabric workspace
- Select new item > Warehouse
- Name warehouse



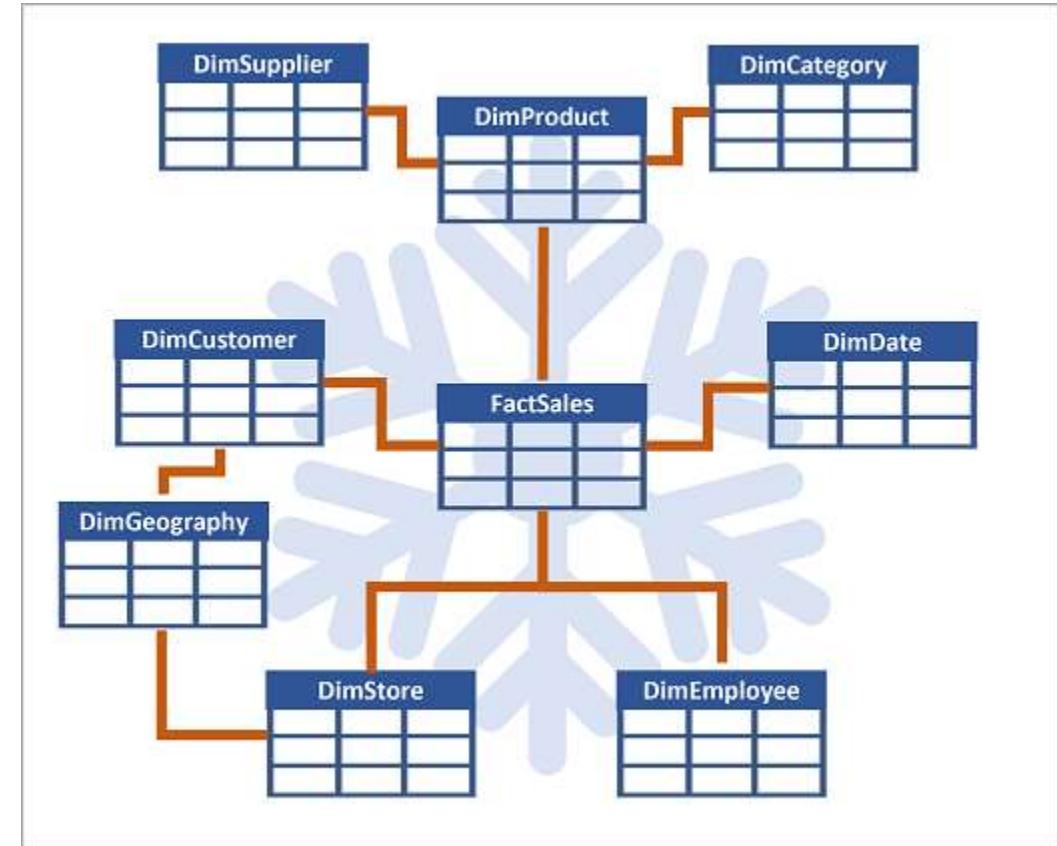
Design a dimensional warehouse

- Fact tables
- Dimension tables
- Unique keys
 - Surrogate key
 - Alternate key



Consider a snowflake schema

- Star schema, further denormalized
- More granular dimensions



Special types of dimension tables

Time dimensions

- Extensive date table
- Ideal for aggregation
- Columns may include:
 - Year
 - Quarter
 - Month
 - Day
 - Fiscal Year

Slowly changing dimensions

- Changes to attributes
- Analyze changes over time
- Changes may include:
 - Customer address
 - Product price

Ingest data into a data warehouse

Fabric
Pipelines

Fabric
Dataflow
Gen2

Cross-
database
querying

COPY INTO
command

Query data in Fabric

Use the SQL query editor

Like using SQL Server Management Studio (SSMS) or Azure Data Studio (ADS).

- Intellisense
- Code completion
- Syntax highlighting
- Client-side parsing
- Validation

The screenshot shows the Azure Data Studio interface. The top navigation bar includes 'Home' and icons for 'New SQL query' and 'New visual query'. A status message at the top right indicates a default Power BI semantic model was created. The left pane, titled 'Explorer', shows a tree view of a 'Sales' warehouse with 'Schemas' (dbo, INFORMATION_SCHEMA, sys, SECURITY), 'Tables' (dimcustomer_gold, dimdate_gold, dimproduct_gold, factsales_gold, sales_silver), 'Views' (OrderSummaryView), 'Functions', 'Stored Procedures', and 'Guest'. The right pane, titled 'SQL query 1', contains the following T-SQL code:

```
1 CREATE VIEW OrderSummaryView AS
2 SELECT
3     d.Month,
4     c.Email,
5     SUM(s.Quantity) AS TotalOrders
6 FROM
7     factsales_gold s
8 JOIN
9     dimdate_gold d ON s.OrderDate = d.OrderDate
10 JOIN
11    dimcustomer_gold c ON s.CustomerID = c.CustomerID
12 GROUP BY
13    d.Month, c.Email;
```

Visualize queries in Fabric

Create a visual query

Drag table to the canvas, then transform as needed:

- Manage columns
- Filter rows
- Replace values
- Group by
- And more

The screenshot shows the Microsoft Fabric Visual Query Editor interface. At the top, there are tabs for "SQL query 1", "SQL query 2", "SQL query 3", and "Visual query 1". The "Visual query 1" tab is active. Below the tabs is a toolbar with icons for "Manage columns", "Reduce rows", "Sort", "Transform", "Combine", "View SQL", and others. A search bar "Search commands" is also present.

The main area displays a data flow diagram. On the left, a "DimProduct" table is shown with three columns: "Source", "Database", and "Table". An arrow points from this table to a "FactSalesOrder" table, which is also shown with three columns: "Source", "Database", and "Table". The "FactSalesOrder" table has a blue border around it, indicating it is selected. To the right of the tables are several status indicators: "Merged queries", "Expanded DimPr", "Filtered rows", and a count of "1".

On the far right, a vertical sidebar contains a list of actions with corresponding icons:

- Manage columns
 - Choose columns
 - Remove columns
 - Remove other columns
- Reduce rows
 - Keep top rows
 - Keep duplicates
 - Filter rows
- Sort
 - Sort ascending
 - Sort descending
- Combine
 - Merge queries
 - Merge queries as new
 - Append queries
 - Append queries as new
- Transform table
 - Group by

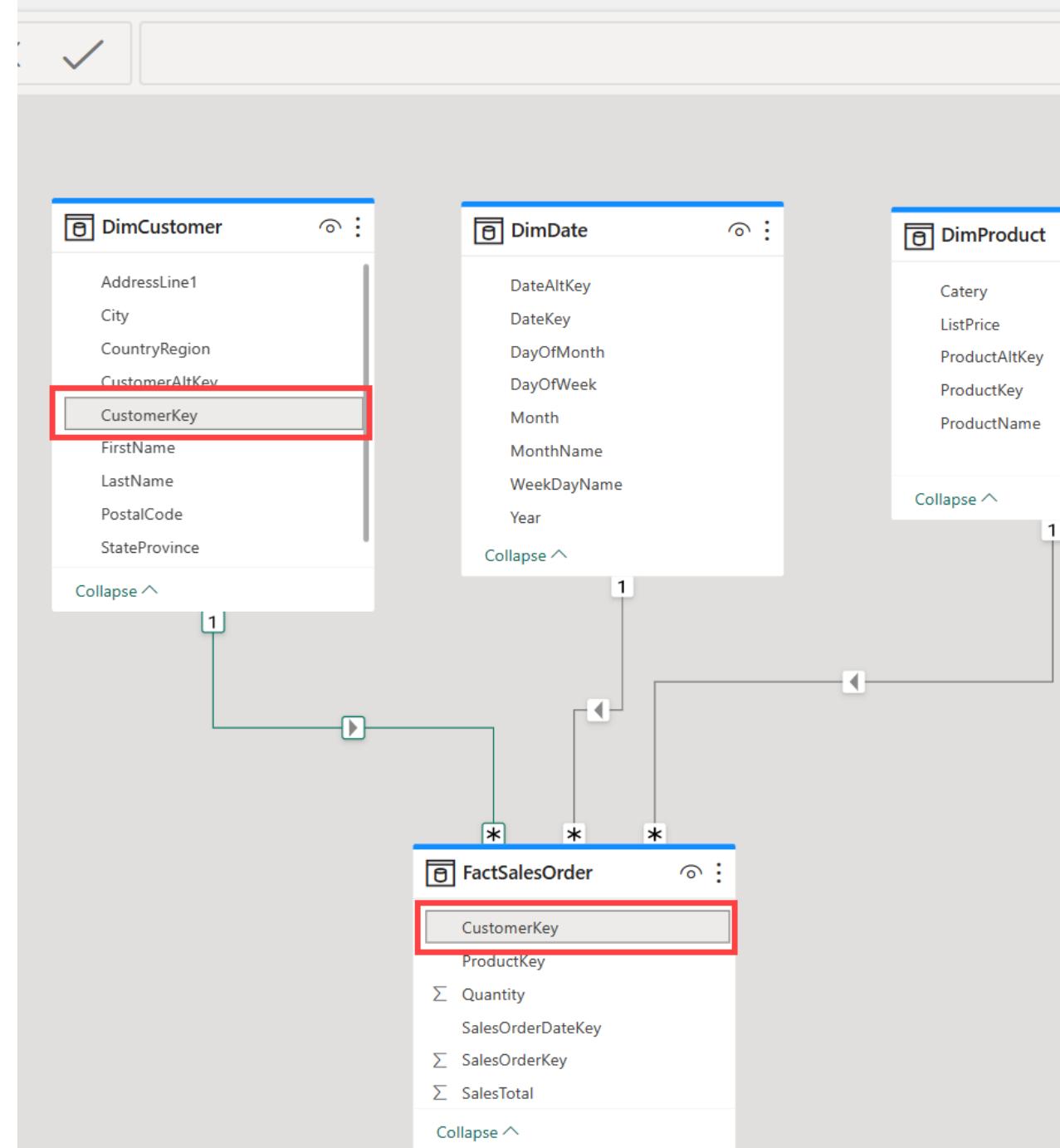
At the bottom of the interface, there is a table preview titled "Visualize results". The table has 13 rows and 8 columns. The columns are labeled: SalesOrderKey, SalesOrderDateKey, ProductKey, CustomerKey, Quantity, SalesTotal, and ProductName. The "ProductName" column contains the value "Cable Lock" for all rows. The table also includes download options: "Download Excel file" and "Visualize results".

	SalesOrderKey	SalesOrderDateKey	ProductKey	CustomerKey	Quantity	SalesTotal	ProductName
1	5306	20221121	843	480	2	50	Cable Lock
2	4685	20220819	843	52	5	125	Cable Lock
3	4690	20220821	843	29580	9	225	Cable Lock
4	5007	20221011	843	29583	3	75	Cable Lock
5	3734	20220317	843	400	11	275	Cable Lock
6	3917	20220416	843	29778	8	200	Cable Lock
7	3922	20220418	843	1	8	200	Cable Lock
8	2932	20211024	843	665	5	125	Cable Lock
9	2976	20211102	843	29915	2	50	Cable Lock
10	3199	20211120	843	564	11	275	Cable Lock
11	1319	20210118	843	29911	8	200	Cable Lock
12	1536	20210222	843	131	5	125	Cable Lock
13	1558	20210224	843	214	8	200	Cable Lock

Build relationships

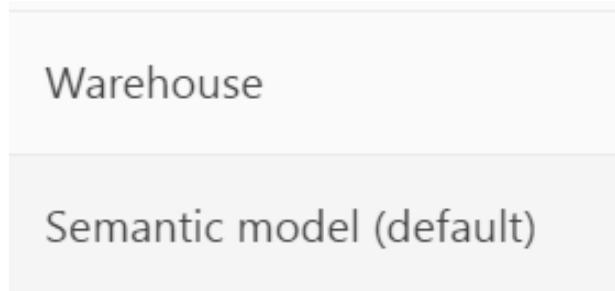
Connect fact and dimension tables

- Use **Model** view in warehouse in Fabric portal.
- Identify the appropriate key between tables.
- Drag columns from one table on top of the related column in another table.



Understand the default semantic model

Fabric automatically creates a default semantic model for Power BI users to use for report creation.

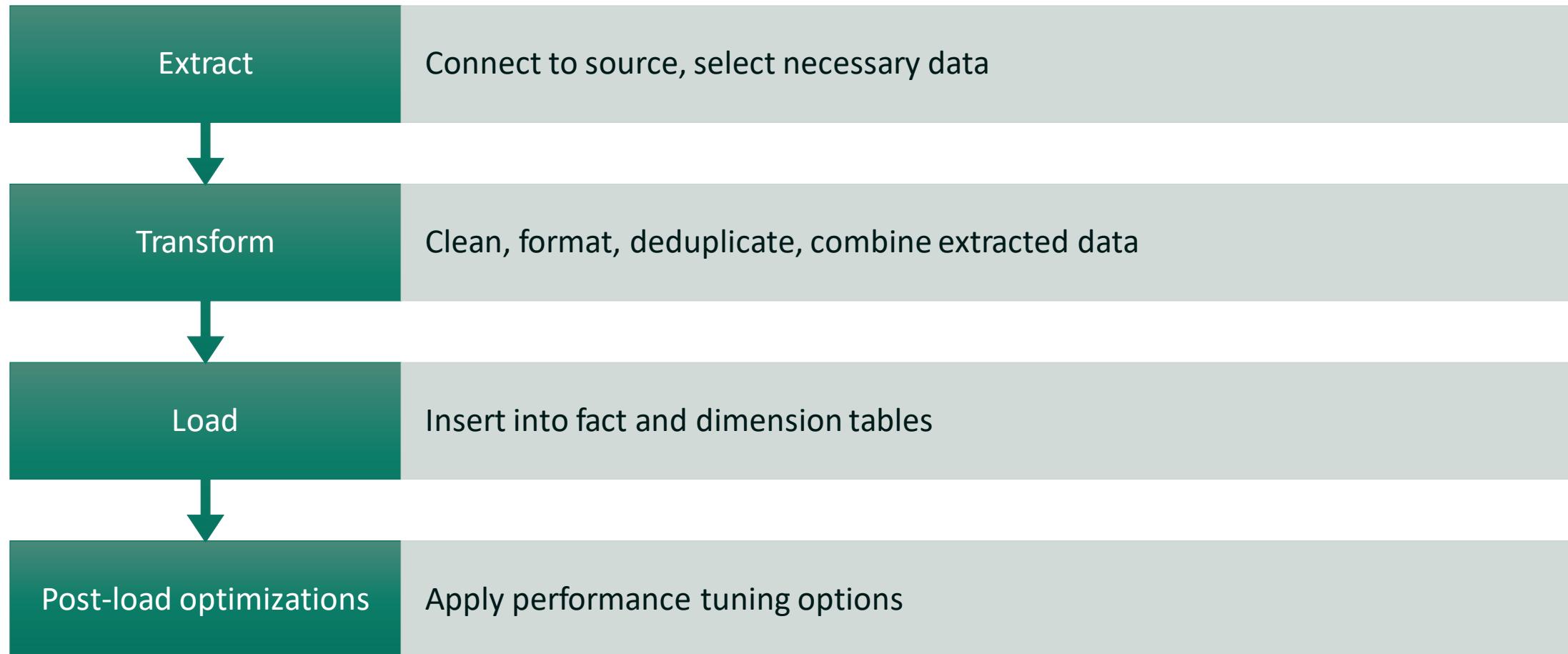


Consider shaping the semantic model to reduce downstream transformations.

- User-friendly column names
- Hide columns or tables from view

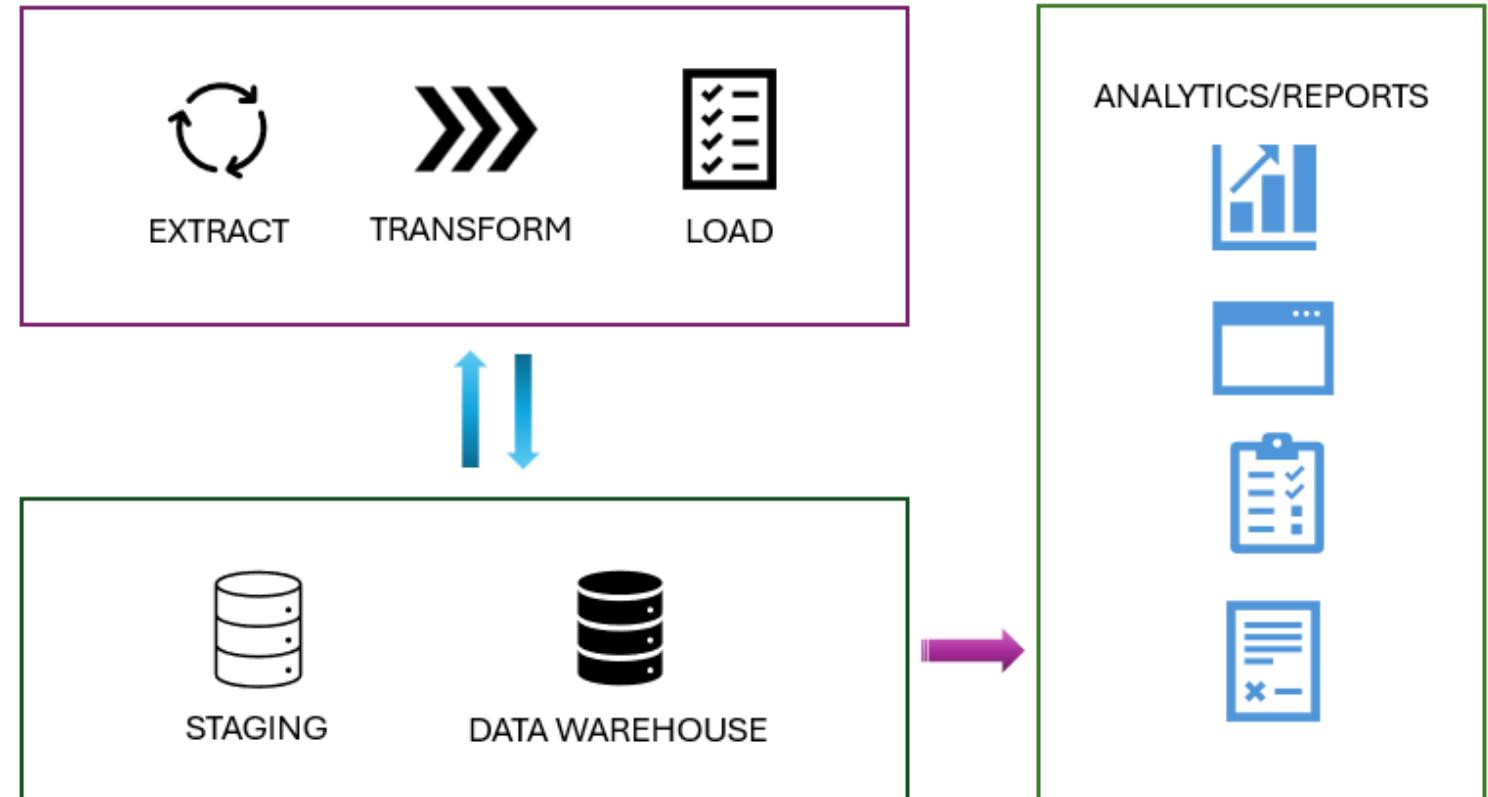
FactSalesOrder	
CustomerKey	⟲
ProductKey	⟲
Σ Quantity	
SalesOrderDateKey	⟲
Σ SalesOrderKey	⟲

Understand ETL (Extract, Transform and Load)



Stage the data

- Temporary storage and transformation
- Simplifies load operations



Different data load types

Full (initial) load

- Truncate table and load data
- Longer load time
- No history stored
- Best for initial load or refresh

Incremental load

- Append data to tables
- Faster updates
- Preserves history (timestamp)
- Best for frequent updates

Create tables with T-SQL

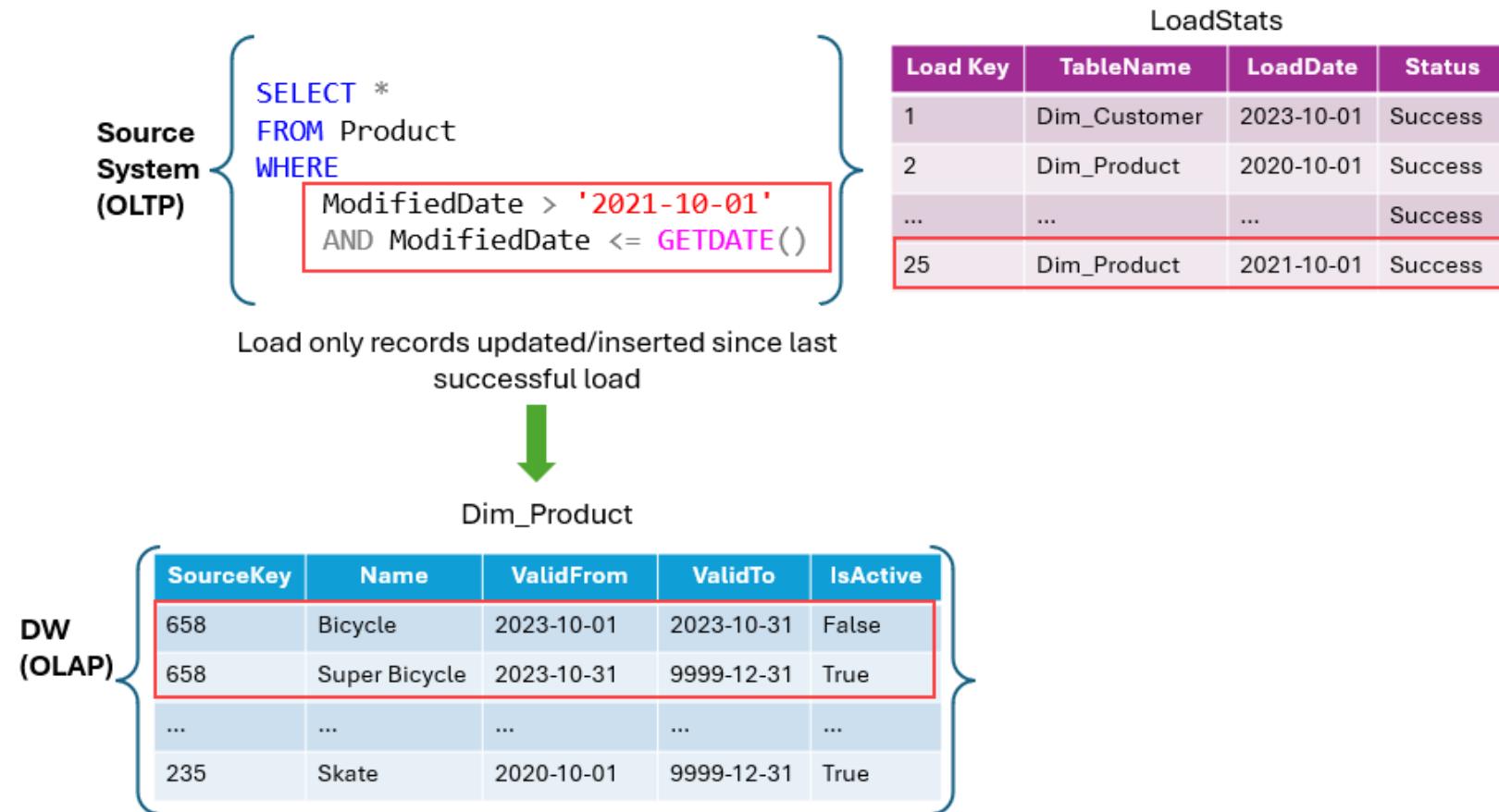
- SQL Server Management Studio
- Microsoft Fabric service portal

```
COPY INTO dbo.Region
FROM
'https://mystorageaccountxxx.blob.core.windows.net/private/Region.csv' WITH (
    FILE_TYPE = 'CSV'
    ,CREDENTIAL = (
        IDENTITY = 'Shared Access
Signature'
        , SECRET = 'xxx'
    )
    ,FIRSTROW = 2
)
GO
```

Load dimension tables

Slowly changing dimension types

- 0: never changes
- 1: overwrites, no history
- 2: appends, tracks changes
- 3: add history column
- 4: add new dimension
- 5: like type 2 for large changes
- 6: combo of types 2 and 3



Example

Update table *Dim_Products*,
designed as a **Type 2 SCD**.

```
IF EXISTS (SELECT 1 FROM Dim_Products WHERE
SourceKey = @ProductID AND IsActive =
'True')

BEGIN
    -- Existing product record
    UPDATE Dim_Products
        SET ValidTo = GETDATE(), IsActive =
'False'
        WHERE SourceKey = @ProductID AND
IsActive = 'True';

END
ELSE
BEGIN
    -- New product record
    INSERT INTO Dim_Products (SourceKey,
ProductName, StartDate, EndDate, IsActive)
        VALUES (@ProductID, @ProductName,
GETDATE(), '9999-12-31', 'True');

END
```

Load fact tables

Frequent inserts and updates.

Common uses include:

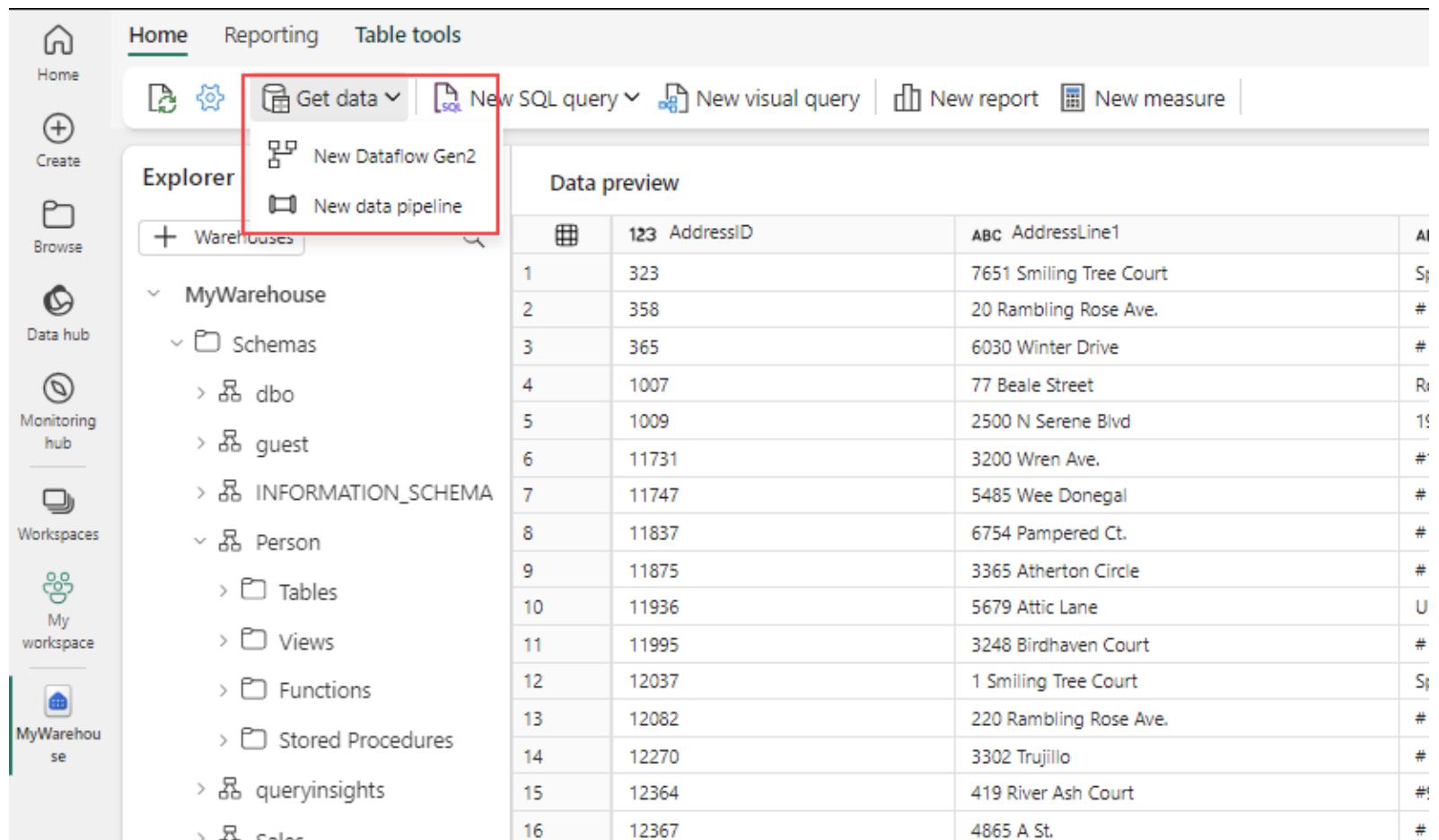
- Customers
- Claims
- Events
- Visits
- Calls
- Inventory

```
-- Lookup keys in dimension tables
INSERT INTO Fact_Sales
    (DateKey, ProductKey, OrderID, Quantity, Price,
     LoadTime)
SELECT d.DateKey
      , p.ProductKey
      , o.OrderID
      , o.Quantity
      , o.Price
      , GETDATE()
FROM Order_Detail o
JOIN Dim_Date d ON o.OrderDate = d.Date
JOIN Dim_Product p ON o.ProductID = p.ProductID;
```

Load data with Fabric pipelines

Orchestrate activities

- Copy Data
- Schedule
- Parameters
- Delete Data



The screenshot shows the Power BI Data Explorer interface. The top navigation bar includes Home, Reporting, and Table tools tabs. Below the ribbon, there are several icons: Home, Create, Browse, Data hub, Monitoring hub, Workspaces, My workspace, and MyWarehouse. The 'MyWarehouse' section is expanded, showing Schemas (dbo, guest, INFORMATION_SCHEMA), Person (Tables, Views, Functions, Stored Procedures), queryinsights, and roles. On the far left, a sidebar lists Home, Create, Browse, Data hub, Monitoring hub, Workspaces, My workspace, and MyWarehouse. The main area is titled 'Data preview' and displays a table with columns for AddressID and AddressLine1. The table contains 16 rows of data.

	AddressID	AddressLine1
1	323	7651 Smiling Tree Court
2	358	20 Rambling Rose Ave.
3	365	6030 Winter Drive
4	1007	77 Beale Street
5	1009	2500 N Serene Blvd
6	11731	3200 Wren Ave.
7	11747	5485 Wee Donegal
8	11837	6754 Pampered Ct.
9	11875	3365 Atherton Circle
10	11936	5679 Attic Lane
11	11995	3248 Birdhaven Court
12	12037	1 Smiling Tree Court
13	12082	220 Rambling Rose Ave.
14	12270	3302 Trujillo
15	12364	419 River Ash Court
16	12367	4865 A St.

Load data with T-SQL

Use COPY statement

- External Azure storage
- Specify file format
(PARQUET / CSV)
- Error handling
- Multiple files

```
-- Load as CSV
COPY my_table
FROM
'https://myaccount.blob.core.windows.net/myblobcontainer/
folder0/*.csv',
https://myaccount.blob.core.windows.net/myblobcontainer/f
older1/'
WITH (FILE_TYPE = 'CSV',
      CREDENTIAL=(IDENTITY= 'Shared Access Signature',
      SECRET='<Your_SAS_Token>')
      FIELDTERMINATOR = '|')

-- Load as PARQUET
COPY INTO test_parquet
FROM
'https://myaccount.blob.core.windows.net/myblobcontainer/
folder1/*.parquet'
WITH (CREDENTIAL=(IDENTITY= 'Shared Access Signature',
SECRET='<Your_SAS_Token>'))
```

Load from other items

Combine into one warehouse

- Requires three-part-naming
- CREATE TABLE AS SELECT
- INSERT...SELECT

```
CREATE TABLE  
[analysis_warehouse].[dbo].[combined_data]  
AS  
SELECT  
FROM [sales_warehouse].[dbo].[sales_data] sales  
INNER JOIN [social_lakehouse].[dbo].[social_data]  
social  
ON sales.[product_id] = social.[product_id];  
  
INSERT INTO  
[analysis_warehouse].[dbo].[combined_data]  
SELECT  
    sales.product_id  
FROM [sales_warehouse].[dbo].[sales_data] sales  
INNER JOIN [social_lakehouse].[dbo].[social_data]  
social  
ON sales.product_id = social.product_id;
```

Load data with Dataflow Gen2

Ingest and transform with Power Query

- Connect to data source
- Transform data – leverage copilot
- Append or replace
- Add data destination
 - Lakehouse
 - Warehouse
 - Azure SQL Database
 - Azure Data Explorer (Kusto)
 - Azure Synapse Analytics (SQL DW)

Query settings >

Properties

Name

customer-churn 1

Entity type ⓘ

Custom

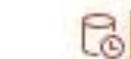
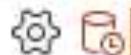
Applied steps

Source

Promoted headers

Changed column type

Custom



Data destination +

No data destination

Use the SQL query editor

Warehouse explorer in Fabric portal

- Use T-SQL
- Supports Intellisense
- Stores in **My queries**

The screenshot shows the Microsoft Fabric portal's Warehouse explorer interface. The top navigation bar includes Home, Reporting, Get data, New SQL query (highlighted with a red box and circled 1), New visual query, New report, and New measure. A message indicates a default Power BI semantic model was created. The left sidebar has sections for Home, Create, Browse, Data hub, Monitoring hub, Workspaces, and My workspace (highlighted with a blue box and circled 2). Under 'sample-dw', it shows Schemas (dbo, Geography, HackneyLicen..., Medallion, Time, Trip, Weather), Views, Functions, Stored Procedures, guest, INFORMATION_SCHEMA, queryinsights, sys, Security, and Queries (highlighted with a red box and circled 3). The main area is titled 'Data preview' and shows a table with 32 columns and 1,000 rows. The table includes columns like DateID, Date, ABC DateKey, ABC DayOfMonth, ABC DaySuffix, ABC DayName, ABC DayOfWeek, and ABC DayOfWeekMon. The bottom navigation bar includes Data, Query (highlighted with a red box), and Model.

1	DateID	Date	ABC DateKey	ABC DayOfMonth	ABC DaySuffix	ABC DayName	ABC DayOfWeek	ABC DayOfWeekMon
1	20150317	2015-03-17 00:00:00.000000	03/17/2015	17	17th	Tuesday	3	3
2	20050317	2005-03-17 00:00:00.000000	03/17/2005	17	17th	Thursday	5	3
3	20010317	2001-03-17 00:00:00.000000	03/17/2001	17	17th	Saturday	7	3
4	20090317	2009-03-17 00:00:00.000000	03/17/2009	17	17th	Tuesday	3	3
5	20040317	2004-03-17 00:00:00.000000	03/17/2004	17	17th	Wednesday	4	3
6	20000317	2000-03-17 00:00:00.000000	03/17/2000	17	17th	Friday	6	3
7	20120317	2012-03-17 00:00:00.000000	03/17/2012	17	17th	Saturday	7	3
8	20060317	2006-03-17 00:00:00.000000	03/17/2006	17	17th	Friday	6	3
9	20110317	2011-03-17 00:00:00.000000	03/17/2011	17	17th	Thursday	5	3
10	20100317	2010-03-17 00:00:00.000000	03/17/2010	17	17th	Wednesday	4	3
11	20070317	2007-03-17 00:00:00.000000	03/17/2007	17	17th	Saturday	7	3
12	20140314	2014-03-14 00:00:00.000000	03/14/2014	14	14th	Friday	6	2
13	20090314	2009-03-14 00:00:00.000000	03/14/2009	14	14th	Friday	6	2
14	20080314	2008-03-14 00:00:00.000000	03/14/2008	14	14th	Thursday	5	2
15	20110214	2011-02-14 00:00:00.000000	02/14/2011	14	14th	Monday	2	2
16	20050214	2005-02-14 00:00:00.000000	02/14/2005	14	14th	Monday	2	2
17	20010214	2001-02-14 00:00:00.000000	02/14/2001	14	14th	Wednesday	4	2
18	20060214	2006-02-14 00:00:00.000000	02/14/2006	14	14th	Tuesday	3	2
19	20020214	2002-02-14 00:00:00.000000	02/14/2002	14	14th	Thursday	5	2
20	20120214	2012-02-14 00:00:00.000000	02/14/2012	14	14th	Tuesday	3	2
21	20000214	2000-02-14 00:00:00.000000	02/14/2000	14	14th	Monday	2	2
22	20100214	2010-02-14 00:00:00.000000	02/14/2010	14	14th	Sunday	1	2
23	20070214	2007-02-14 00:00:00.000000	02/14/2007	14	14th	Wednesday	4	2
24	20130214	2013-02-14 00:00:00.000000	02/14/2013	14	14th	Thursday	5	2
25	20091031	2009-10-31 00:00:00.000000	10/31/2009	31	31st	Saturday	7	5
26	20121031	2012-10-31 00:00:00.000000	10/31/2012	31	31st	Wednesday	4	5
27	20071031	2007-10-31 00:00:00.000000	10/31/2007	31	31st	Wednesday	4	5
28	20031031	2003-10-31 00:00:00.000000	10/31/2003	31	31st	Thursday	6	5

Run queries and export results

The screenshot shows the SSMS interface with the 'Explorer' tab selected. In the left sidebar, under 'sample database', the 'dbo' schema is expanded, showing tables like 'city', 'geography', 'HackneyLicense', 'Medallion', 'Time', 'trip', 'weather', and views like 'geography'. The 'city' table is currently selected, displaying its schema and data in the main pane. The data grid shows columns: 'CityID', 'City', 'Lat', 'Long', 'StateID', 'State', 'DateCreated', 'Avg. DateBorn', 'Avg. DayOffMonth', 'Avg. DaysOff', 'Avg. DayOffName', 'Avg. DayOffWeek', and 'Avg. DayOffYearMonth'. There are 1000 rows of data.

The screenshot shows the SSMS interface with the 'Query' tab selected. A new query window titled 'SQL query 1' is open, containing the command: 'SELECT * FROM [dbo].[trip]'. The results pane displays the data from the 'trip' table. The table has columns: 'TripID', 'CityID', 'MedallionID', 'HackneyLicenseID', 'PickedUpTimeID', 'DroppedOffTimeID', 'PickupGeographyID', and 'DropoffID'. The results show 7 rows of data.

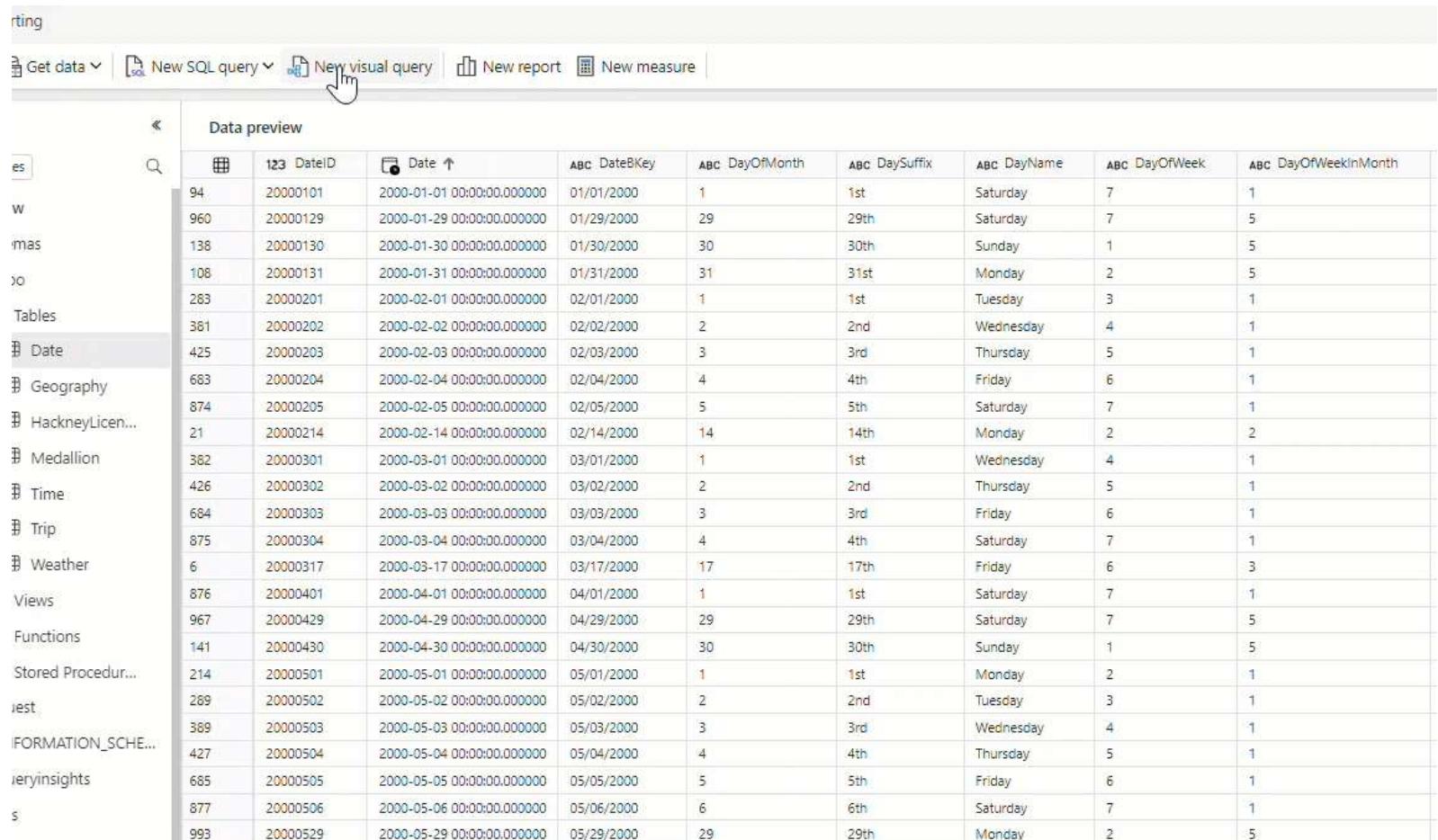
Run and save results as view or table

Download as Excel file

Explore visual query editor

Visually build queries

- Graphical interface
- Automatic query generation
- Save as table/view



The screenshot shows the Power BI Visual Query Editor interface. At the top, there's a navigation bar with tabs: 'Get data', 'New SQL query' (selected), 'New visual query' (highlighted with a red box and a mouse cursor), 'New report', and 'New measure'. On the left, there's a sidebar with a tree view of data sources and tables. Under 'Tables', 'Date' is selected and highlighted with a red box. Below the table name, there are other tables like 'Geography', 'HackneyLicen...', 'Medallion', 'Time', 'Trip', and 'Weather'. The main area is titled 'Data preview' and shows a grid of data from the selected 'Date' table. The columns are labeled: DateID, Date, DateKey, DayOfMonth, DaySuffix, DayName, DayOfWeek, and DayOfWeekInMonth. The data consists of various dates from January 1st to May 29th, 2000, along with their corresponding day names and week-day indices.

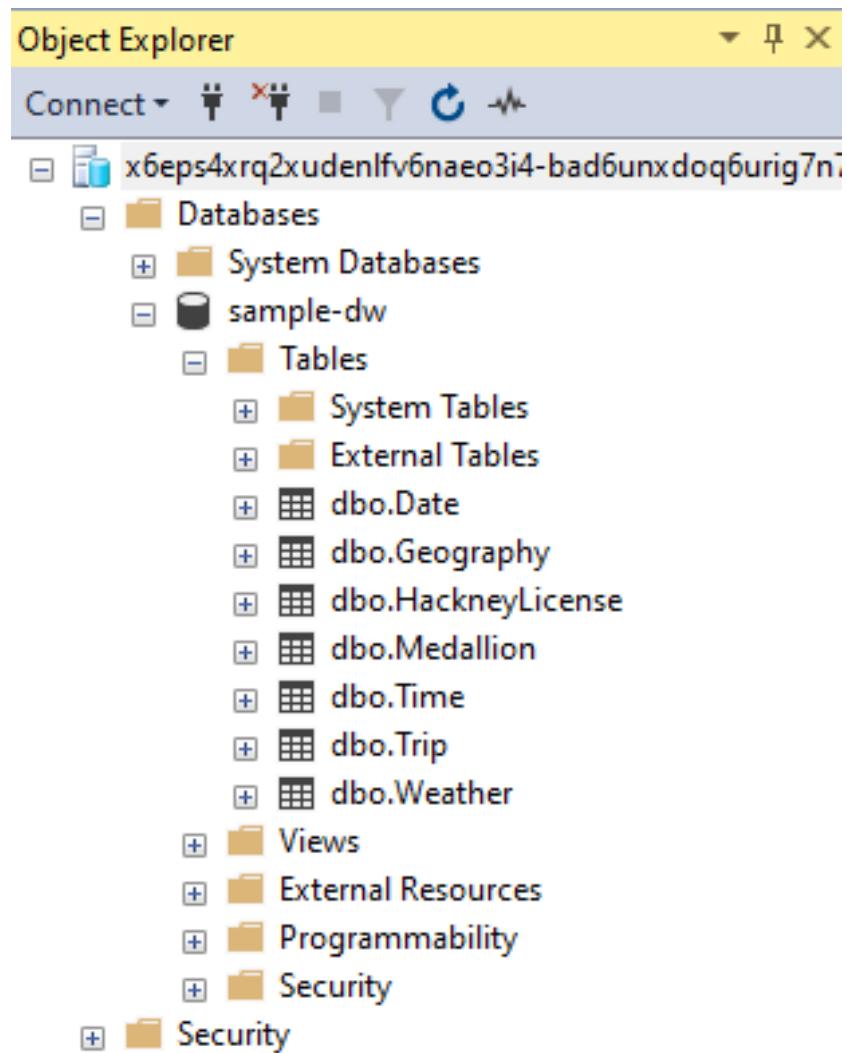
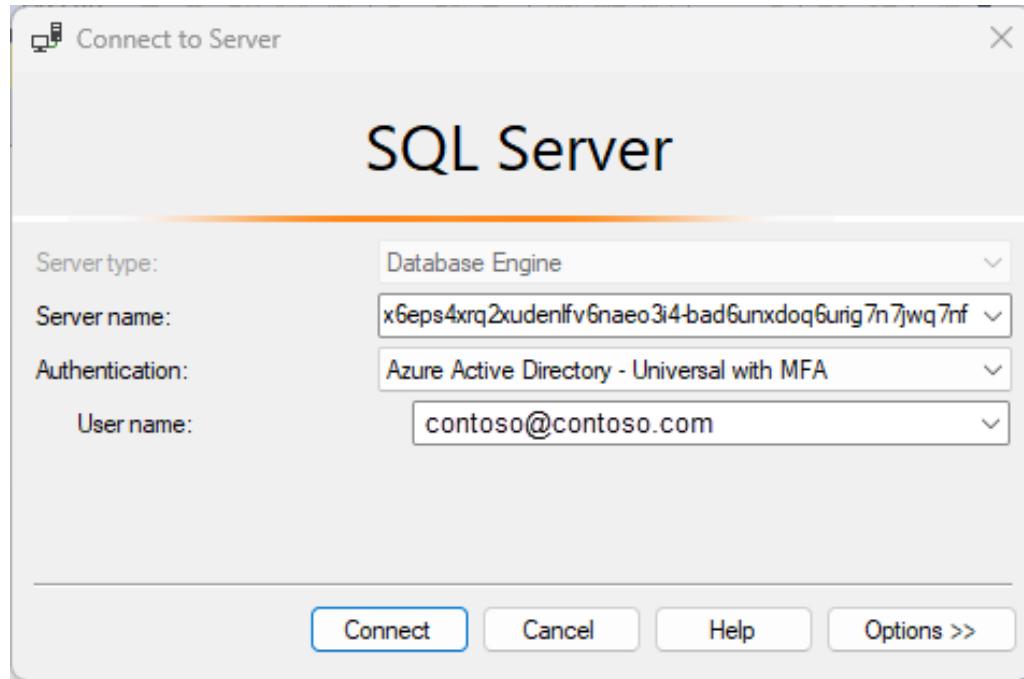
DateID	Date	DateKey	DayOfMonth	DaySuffix	DayName	DayOfWeek	DayOfWeekInMonth
94	2000-01-01 00:00:00.000000	01/01/2000	1	1st	Saturday	7	1
960	2000-01-29 00:00:00.000000	01/29/2000	29	29th	Saturday	7	5
138	2000-01-30 00:00:00.000000	01/30/2000	30	30th	Sunday	1	5
108	2000-01-31 00:00:00.000000	01/31/2000	31	31st	Monday	2	5
283	2000-02-01 00:00:00.000000	02/01/2000	1	1st	Tuesday	3	1
381	2000-02-02 00:00:00.000000	02/02/2000	2	2nd	Wednesday	4	1
425	2000-02-03 00:00:00.000000	02/03/2000	3	3rd	Thursday	5	1
683	2000-02-04 00:00:00.000000	02/04/2000	4	4th	Friday	6	1
874	2000-02-05 00:00:00.000000	02/05/2000	5	5th	Saturday	7	1
21	2000-02-14 00:00:00.000000	02/14/2000	14	14th	Monday	2	2
382	2000-03-01 00:00:00.000000	03/01/2000	1	1st	Wednesday	4	1
426	2000-03-02 00:00:00.000000	03/02/2000	2	2nd	Thursday	5	1
684	2000-03-03 00:00:00.000000	03/03/2000	3	3rd	Friday	6	1
875	2000-03-04 00:00:00.000000	03/04/2000	4	4th	Saturday	7	1
6	2000-03-17 00:00:00.000000	03/17/2000	17	17th	Friday	6	3
876	2000-04-01 00:00:00.000000	04/01/2000	1	1st	Saturday	7	1
967	2000-04-29 00:00:00.000000	04/29/2000	29	29th	Saturday	7	5
141	2000-04-30 00:00:00.000000	04/30/2000	30	30th	Sunday	1	5
214	2000-05-01 00:00:00.000000	05/01/2000	1	1st	Monday	2	1
289	2000-05-02 00:00:00.000000	05/02/2000	2	2nd	Tuesday	3	1
389	2000-05-03 00:00:00.000000	05/03/2000	3	3rd	Wednesday	4	1
427	2000-05-04 00:00:00.000000	05/04/2000	4	4th	Thursday	5	1
685	2000-05-05 00:00:00.000000	05/05/2000	5	5th	Friday	6	1
877	2000-05-06 00:00:00.000000	05/06/2000	6	6th	Saturday	7	1
993	2000-05-29 00:00:00.000000	05/29/2000	29	29th	Monday	2	5

Connect with SQL clients

SQL connection string

Copy this string and use it to connect externally to the item from Power BI desktop or client tools.

x6eps4xrq2xudenlfv6naeo3i4-tonn3enr4vau3chrhyvxxm7py.msit-datawarehouse.fabric.microsoft.com



Measure tools

category Uncategorized

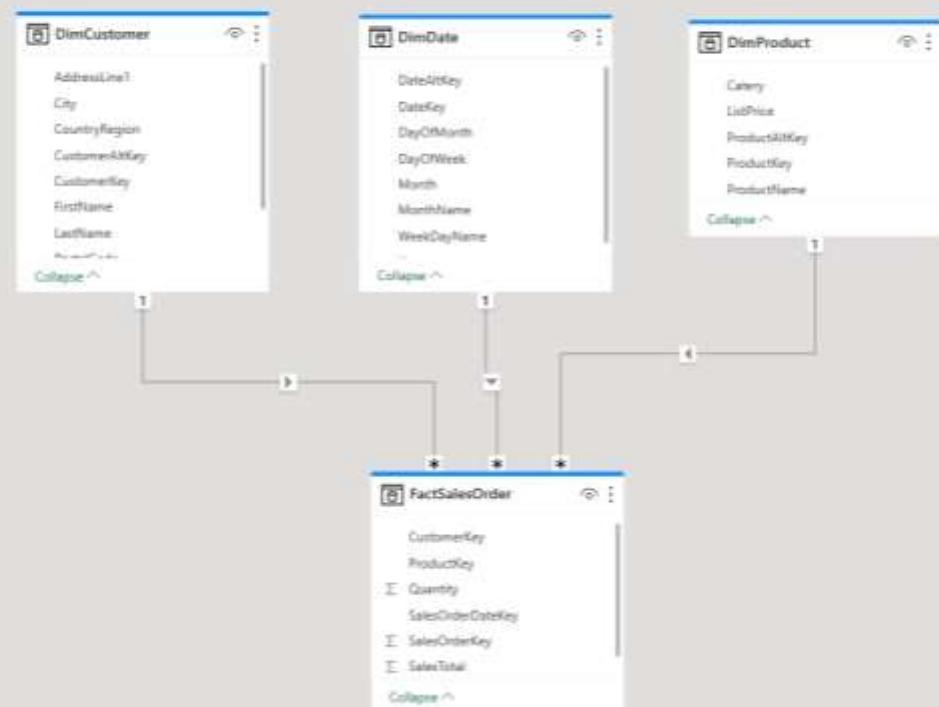
New measure

```
X ✓ 1 Month over Month Sales =  
2 VAR _CurrentMonthSales =  
3   CALCULATE ( SUM ( FactSalesOrder[SalesAmount] ), DATESYTD ( DimDate[Date] ) )  
4 VAR _PreviousMonthSales =  
5   CALCULATE (  
6     SUM ( FactSalesOrder[SalesAmount] ),  
7     DATESYTD ( DATEADD ( DimDate[Date], -1, MONTH ) )  
8   )  
9 RETURN  
10 DIVIDE ( _CurrentMonthSales - _PreviousMonthSales, _PreviousMonthSales, 100 )
```

DAX calculations

Use DAX to create measures

- Create same calculations as with T-SQL
- Hundreds of functions
- Contextual calculations
- As complex as needed



Aggregate measures

Use T-SQL to create measures

Traditional SQL Data

Warehouses relied on T-SQL
to create measures.

Common functions are:

- JOIN
- GROUP BY

```
-- Sales amounts by year and quarter
SELECT    dates.CalendarYear,
          dates.CalendarQuarter,
          SUM(sales.SalesAmount) AS TotalSales
FROM      dbo.FactSales AS sales
JOIN      dbo.DimDate AS dates
          ON sales.OrderDateKey = dates.DateKey
GROUP BY  dates.CalendarYear, dates.CalendarQuarter
ORDER BY  dates.CalendarYear,
          dates.CalendarQuarter;
```

Output – total sales by year and quarter

CalendarYear	CalendarQuarter	TotalSales
2020	1	25980.16
2020	2	27453.87
2020	3	28527.15
2020	4	31083.45
2021	1	34562.96
2021	2	36162.27

Extend measures

Building from last example

Further explores the
**quarterly sales totals by
city** based on the
customer's address details.

```
-- Quarterly total sales by customer city
SELECT    dates.CalendarYear,
          dates.CalendarQuarter,
          custs.City,
          SUM(sales.SalesAmount) AS TotalSales
FROM      dbo.FactSales AS sales
JOIN      dbo.DimDate AS dates ON sales.OrderDateKey =
dates.DateKey
JOIN      dbo.DimCustomer AS custs ON sales.CustomerKey =
custs.CustomerKey
GROUP BY  dates.CalendarYear,
          dates.CalendarQuarter, custs.City
ORDER BY  dates.CalendarYear,
          dates.CalendarQuarter, custs.City;
```

Join in a snowflake schema

Fact to extended dimension

This example joins **FactSales** and the **DimProduct** and **DimCategory** dimension tables, which requires two joins to complete.

```
-- items sold by product category
SELECT cat.ProductCategory,
       SUM(sales.OrderQuantity) AS ItemsSold
  FROM dbo.FactSales AS sales
  JOIN dbo.DimProduct AS prod ON sales.ProductKey =
                                prod.ProductKey
  JOIN dbo.DimCategory AS cat ON prod.CategoryKey =
                                cat.CategoryKey
 GROUP BY cat.ProductCategory
 ORDER BY cat.ProductCategory;
```

Output – items sold by product category

ProductCategory	ItemsSold
Accessories	28271
Bits and pieces	5368
...	...

Ranking functions

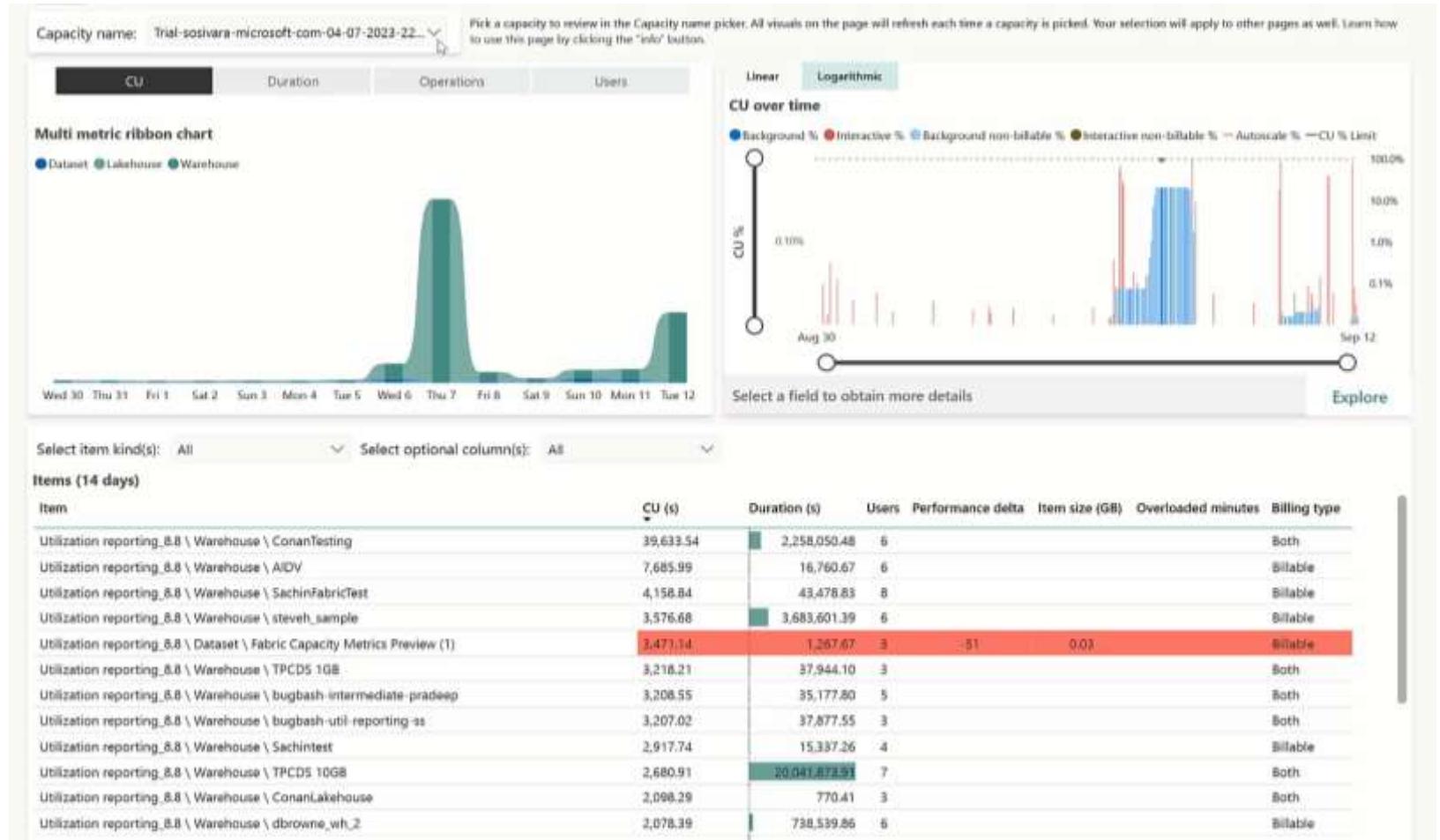
- ROW_NUMBER
- RANK
- DENSE_RANK
- NTILE

```
-- Group by category and relative rank
SELECT ProductCategory,
       ProductName,
       ListPrice,
       ROW_NUMBER() OVER
              (PARTITION BY ProductCategory ORDER BY
               ListPrice DESC) AS RowNumber,
       RANK() OVER
              (PARTITION BY ProductCategory ORDER BY
               ListPrice DESC) AS Rank,
       DENSE_RANK() OVER
              (PARTITION BY ProductCategory ORDER BY
               ListPrice DESC) AS DenseRank,
       NTILE(4) OVER
              (PARTITION BY ProductCategory ORDER BY
               ListPrice DESC) AS Quartile
FROM dbo.DimProduct
ORDER BY ProductCategory;
```

Monitor capacity metrics

Fabric Capacity Metrics app

- Monitor and manage workload costs
- Requires admin install
- Filter by activity type
- Tracks Capacity Units (CUs)
- Primarily read/writes



Monitor current activity

Dynamic Management Views (DMVs) with T-SQL:

- **sys.dm_exec_connections:**
Connections between warehouse and engine
- **sys.dm_exec_sessions:**
Sessions between item and engine
- **sys.dm_exec_requests:**
Active requests in a session

```
--- Identify long-running queries
SELECT request_id, session_id, start_time,
total_elapsed_time
    FROM sys.dm_exec_requests
    WHERE status = 'running'
    ORDER BY total_elapsed_time DESC;

--- Find session_id (user)
SELECT login_name
    FROM sys.dm_exec_sessions
    WHERE 'session_id' = 'SESSION_ID WITH
LONG-RUNNING QUERY';

--- Terminate session (ADMIN only)
KILL 'SESSION_ID WITH LONG-RUNNING QUERY';
```

Query insights views

- **queryinsights.exec_requests_history:**
Completed SQL queries
- **queryinsights.long_running_queries:**
Query execution times
- **queryinsights.frequently_run_queries:**
Frequently run queries

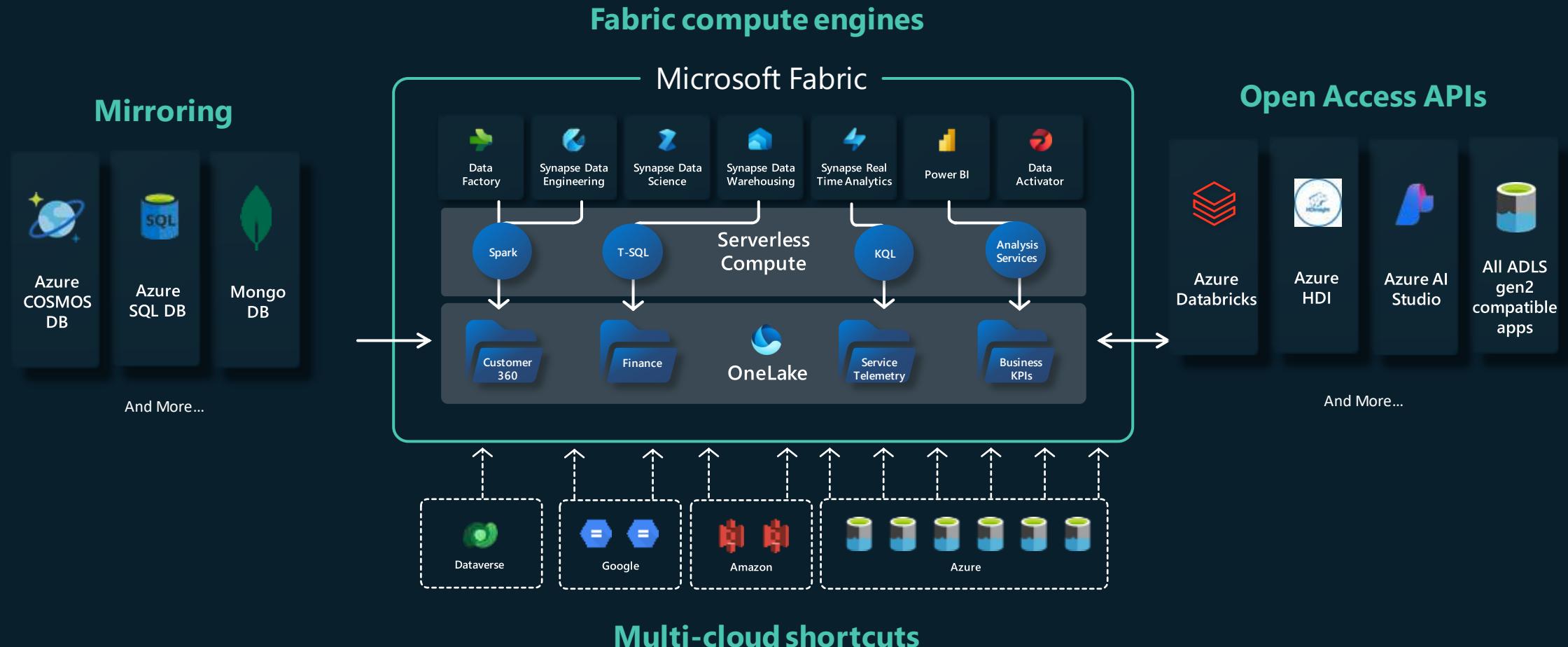
```
--- queries from previous hour
SELECT start_time, login_name, command
FROM queryinsights.exec_requests_history
WHERE start_time >= DATEADD(MINUTE, -60,
GETUTCDATE())

--- long-running queries, multiple runs
SELECT last_run_command, number_of_runs,
median_total_elapsed_time_ms,
last_run_start_time
FROM queryinsights.long_running_queries
WHERE number_of_runs > 1
ORDER BY median_total_elapsed_time_ms DESC;

-- Frequently run queries
SELECT last_run_command, number_of_runs,
number_of_successful_runs,
number_of_failed_runs
FROM queryinsights.frequently_run_queries
ORDER BY number_of_runs DESC
```

All roads lead to OneLake

Creating Data Gravity in OneLake



4

Module 4 Explore and analyze data



Explore and analyze data (20–25%)

Perform exploratory analytics

Implement descriptive and diagnostic analytics

Integrate prescriptive and predictive analytics
into a visual or report

Profile data

Query data by using SQL

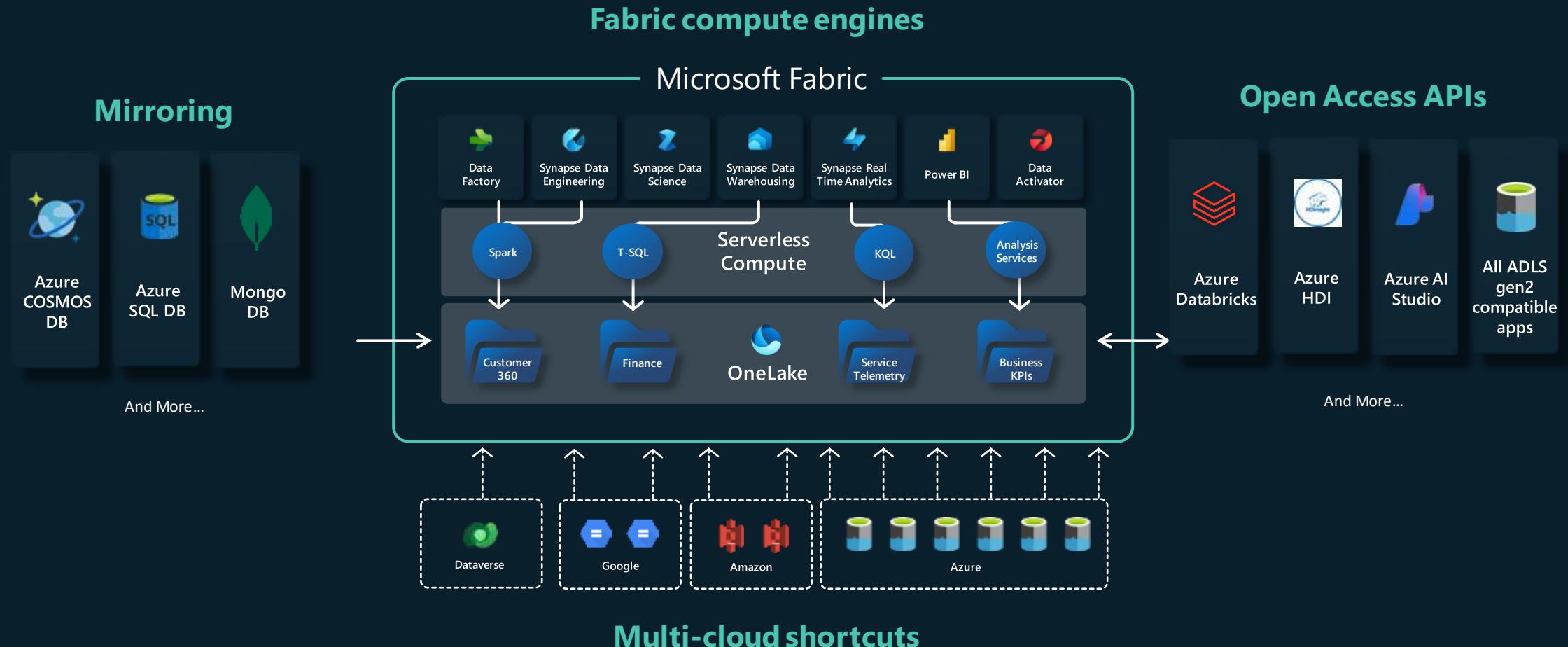
Query a lakehouse in Fabric by using SQL
queries or the visual query editor

Query a warehouse in Fabric by using SQL
queries or the visual query editor

Connect to and query datasets by using the
XMLA endpoint

All roads lead to OneLake

Creating Data Gravity in OneLake



Warehouse and Data ingestion overview

Data Ingestion in Fabric Warehouse

From source data to insights in minutes, regardless of your level of expertise

Choose your desired experience:

- No code, or low code with intuitive UX
- Code rich using T-SQL

Data is instantly optimized for queries with minimal or no knobs required

Data distribution managed automatically

Advanced storage optimizations

Data formats and data type support

All data is stored in Parquet format

Exact numeric	Approximate numeric	Date and Time	Character strings	Binary strings	Other
bit bigint int smallint decimal(p,s) numeric(p,s)	float(n) real	date datetime2(n)* time(n)*	char(n) varchar(n)	varbinary(n)	uniqueidentifier

- For **datetime2** and **time**, precision is limited to 6 digits for fractions of seconds (SQL Server supports up to 7 digits).
- char/varchar can be used in favor of nchar/nvarchar without data loss, but char/varchar uses more storage to store some Unicode characters than nchar/nvarchar.
- LOB types in general (including **char/varchar(max)**) not yet available but coming **very** soon.

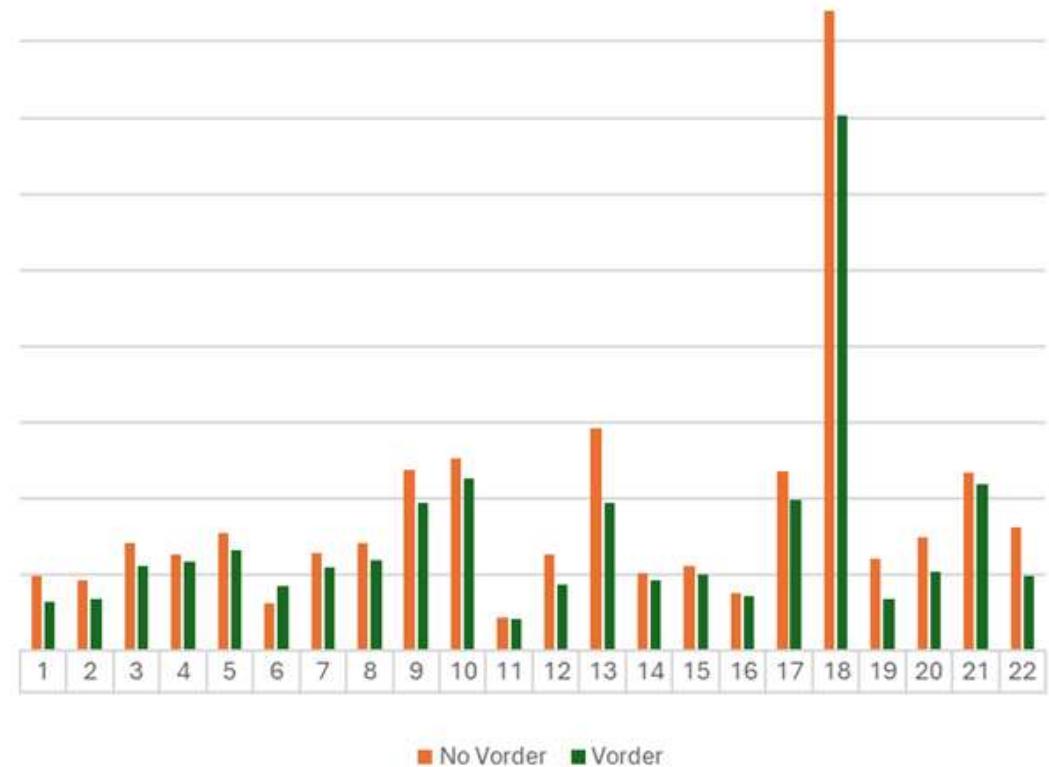
V-Order write optimization

V-Order is a write time optimization to the parquet file format that enables lightning-fast reads under the Microsoft Fabric compute engines, such as Power BI, SQL, Spark and others. It applies special sorting and compression to Parquet

Warehouse queries benefit from faster read times with v-order, still ensuring files are 100% compliant to Parquet's open-source specification

All warehouse data is written with v-order optimization at ingestion time

TPC-H 1TB Fastest Query Execution Time (s)



Data distribution

Data is stored across distributions for scalability

Fabric Warehouse supports Round Robin distribution

- No need to specify distribution column(s)
- No need to concern with data skew
- Source data is split into smaller data cells
- Each new parquet file is assigned with a cell ID between 0 and 1023. As a result, tables have a fixed number of cells: 1024
- As tables grow, we allocate cell IDs to new parquet files ensuring an evenly-distributed number of parquet files per cell ID

This allows:

- Uniform work distribution during query processing
- Scale-out ingestion operations with multiple nodes processing source data

Source data

1	A	Z
2	B	Y
3	C	X
4	D	W
5	E	V
6	F	U
7	G	T
8	H	S
9	I	Q
10	J	P
11	K	O
12	L	N



Rows evenly distributed using Round Robin

1	A	Z
5	E	V

FileA_cell_1.parquet

2	B	Y
4	D	W

FileB_cell_2.parquet

3	C	X
6	F	U

FileC_cell_1024.parquet

8	H	S
12	L	N

FileX_cell_1.parquet

7	G	T
10	J	P

FileY_cell_2.parquet

9	I	Q
11	K	O

FileZ_cell_1024.parquet

Behind the scenes: how data distribution works

Source data

Source file 1		
1	A	Z
2	B	Y
3	C	X
...

Source file 2		
2637	D	W
2638	E	V
2639	F	U
...

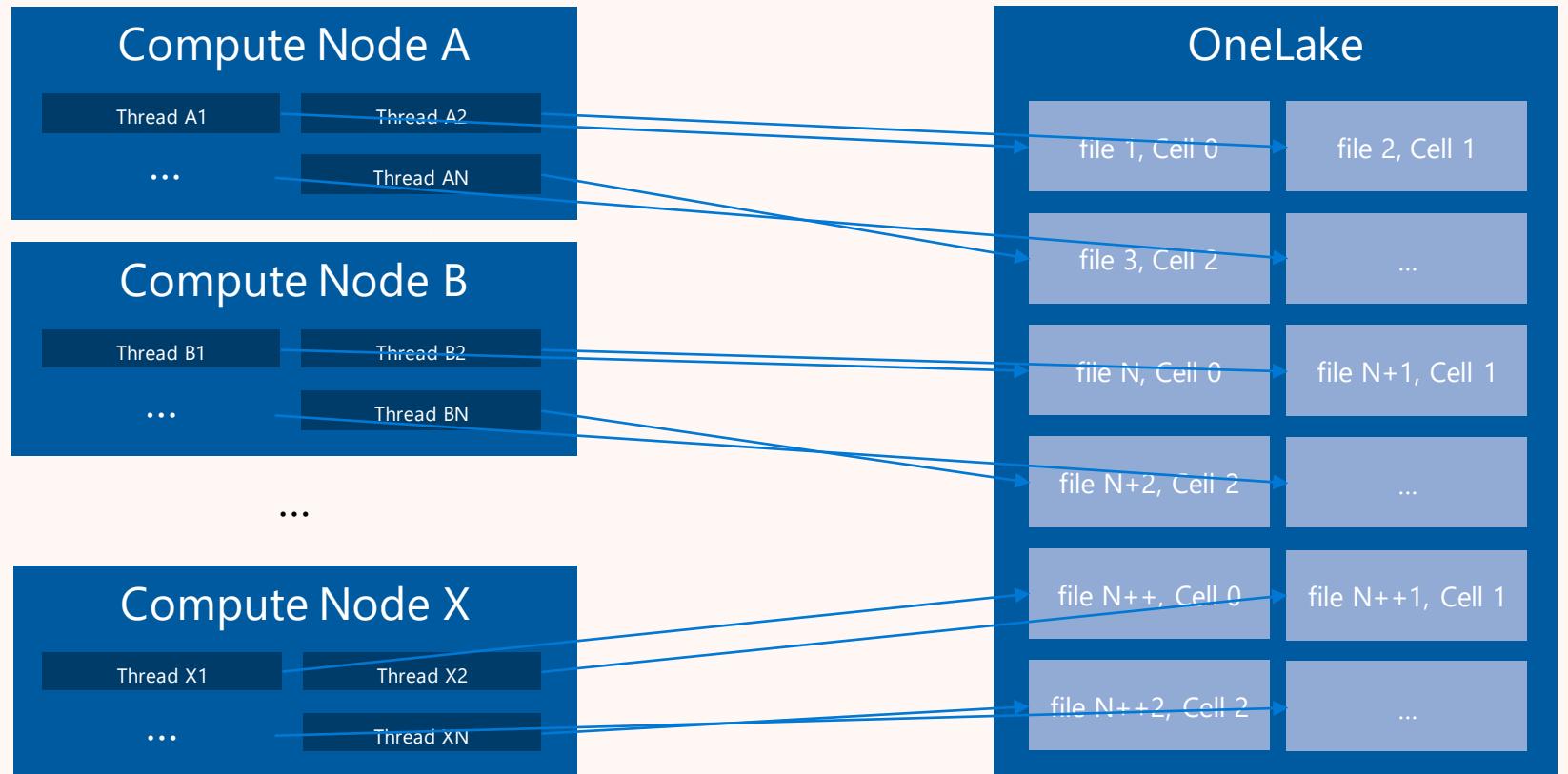
...

Source file N		
7890	G	T
7891	H	S
7892	I	R
...

UQO determines the number of compute nodes needed to process source data

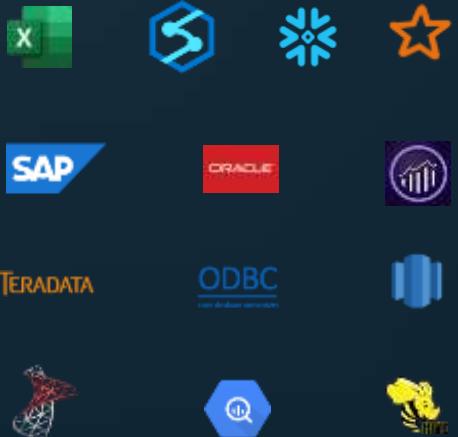
Each compute node processes a subset of the source data

Each thread writes parquet files in OneLake in parallel.
Each file is tagged with a cell ID



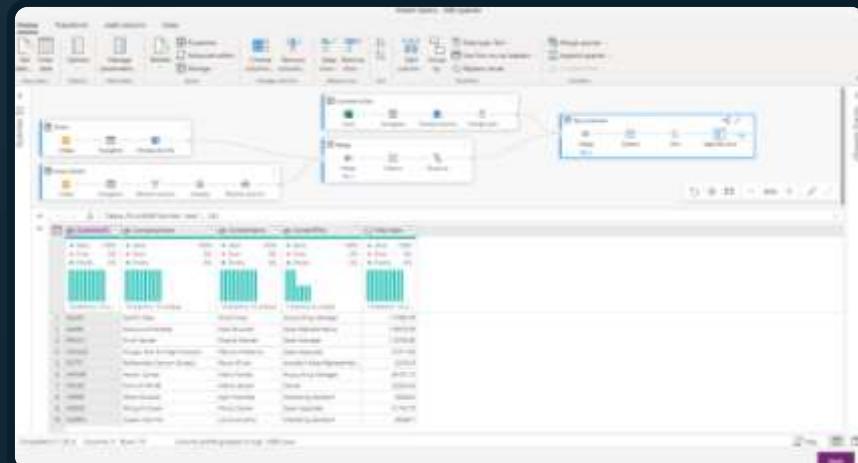
No code data prep with Dataflows gen 2

The worlds data



Through 170+ connectors

Familiar Power Query experiences



Through 400+ no code transformations

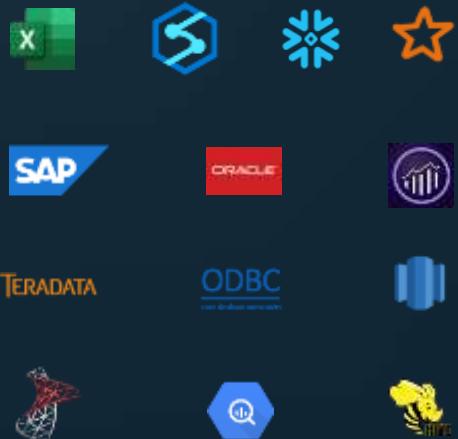


Warehouse

Ingest and transform data at scale

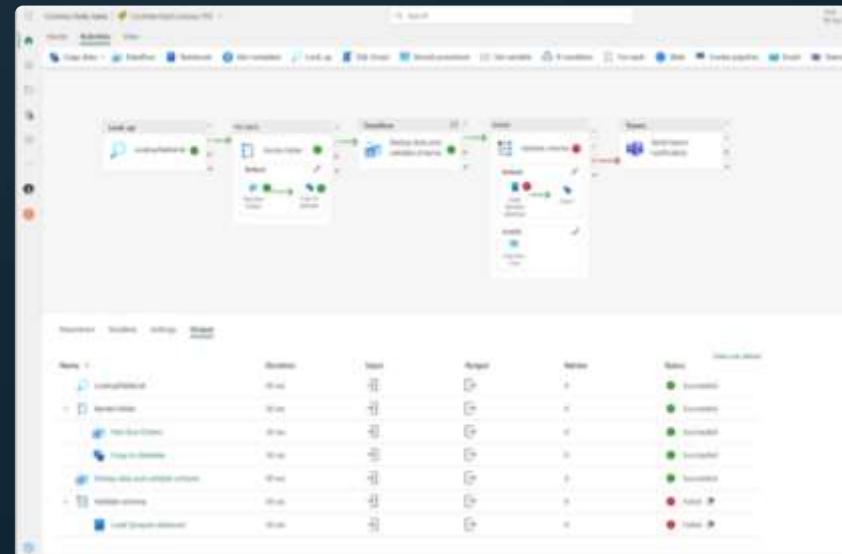
Data orchestration and data prep with Pipelines

The worlds data



Through 170+ connectors

ETL, CDC, or ELT data prep needs



Rich library of activities and pro code data prep experiences



Warehouse

Ingest and transform data at scale

...and of course, you can use SQL too!

SQL COPY INTO, transactions, and more

Support for:

COPY INTO

CTAS

INSERT ... SELECT

Common Table Expressions

Stored Procedures

Transactions
and locking:

Supports snapshot isolation

INSERTS and SELECTS only require
SCHEMA stability locks

UPDATES and DELETES do not
block tables

Simple, Familiar SQL Syntax

```
COPY INTO table_name.[(Column_list)]
FROM '<external_location>' [...n]
WITH
(
[FILE_TYPE = {'CSV' | 'PARQUET' } ]
,[CREDENTIAL = (AZURE CREDENTIAL) ]
,[COMPRESSION = { 'Gzip' | 'Snappy'}]
,[FIELDQUOTE = 'string_delimiter']
,[FIELDTERMINATOR = 'field_terminator']
,[ROWTERMINATOR = 'row_terminator']
,[FIRSTROW = first_row]
,[ENCODING = {'UTF8' | 'UTF16'}]
);
```

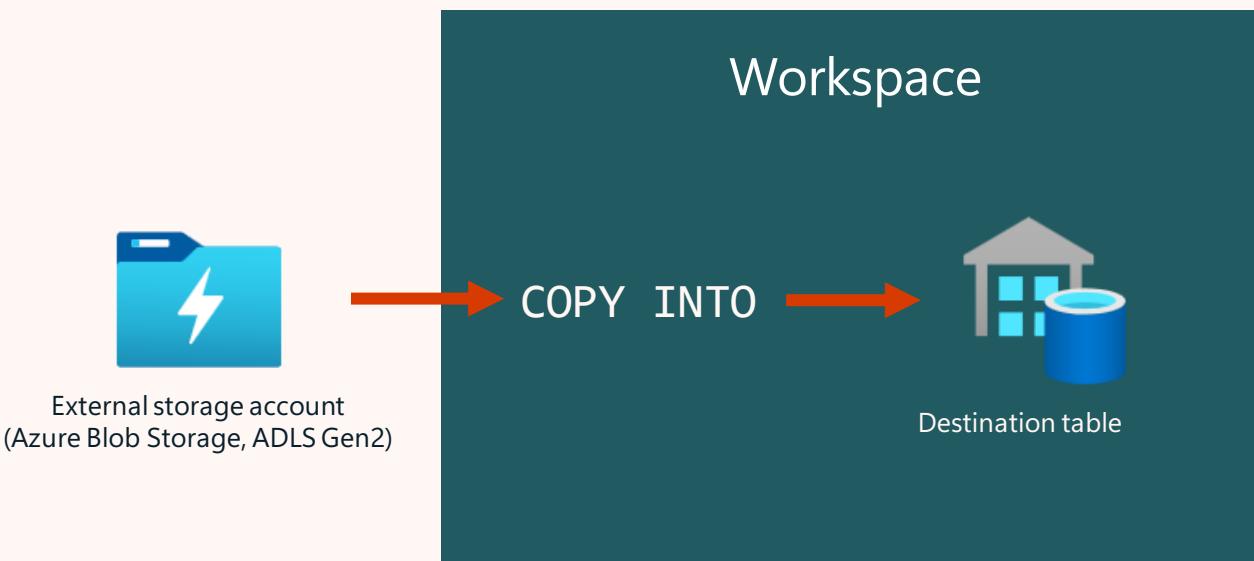
```
CREATE TABLE [dbo].[FactInternetSales_new]
AS
SELECT * FROM [dbo].[FactInternetSales];
```

```
INSERT INTO dbo.EmployeeSales
SELECT 'SELECT', sp.BusinessEntityID, c.LastName, sp.SalesYTD
FROM warehouseA.Sales.SalesPerson AS sp
INNER JOIN warehouseB.Person.Person AS c
ON sp.BusinessEntityID = c.BusinessEntityID
WHERE sp.BusinessEntityID LIKE '2%'
ORDER BY sp.BusinessEntityID, c.LastName;
```

```
WITH sales_data AS (
    SELECT customer_id, SUM(total_sales) AS total_sales
    FROM sales
    GROUP BY customer_id
)
SELECT customer_id, total_sales
FROM sales_data
WHERE total_sales > 1000;
```

COPY INTO

- Enables flexible, high-throughput data ingestion from external Azure storage accounts
- Supports PARQUET and CSV formats
- No additional database objects required



Syntax:

```
COPY INTO table_name.[(Column_list)]
FROM '<external_location>' [...n]
WITH
(
    [FILE_TYPE = { 'CSV' | 'PARQUET' } ]
    [,CREDENTIAL = (AZURE CREDENTIAL) ]
    [,ERRORFILE = '[http(s)://account/container]/directory[/]]' ]
    [,ERRORFILE_CREDENTIAL = (AZURE CREDENTIAL) ]
    [,MAXERRORS = max_errors ]
    [,COMPRESSION = { 'Gzip' | 'Snappy' }]
    [,FIELDQUOTE = 'string_delimiter']
    [,FIELDTERMINATOR = 'field_terminator']
    [,ROWTERMINATOR = 'row_terminator']
    [,FIRSTROW = first_row]
    [,ENCODING = { 'UTF8' | 'UTF16' }]
    [,PARSER_VERSION = { '1.0' | '2.0' }]
)
```

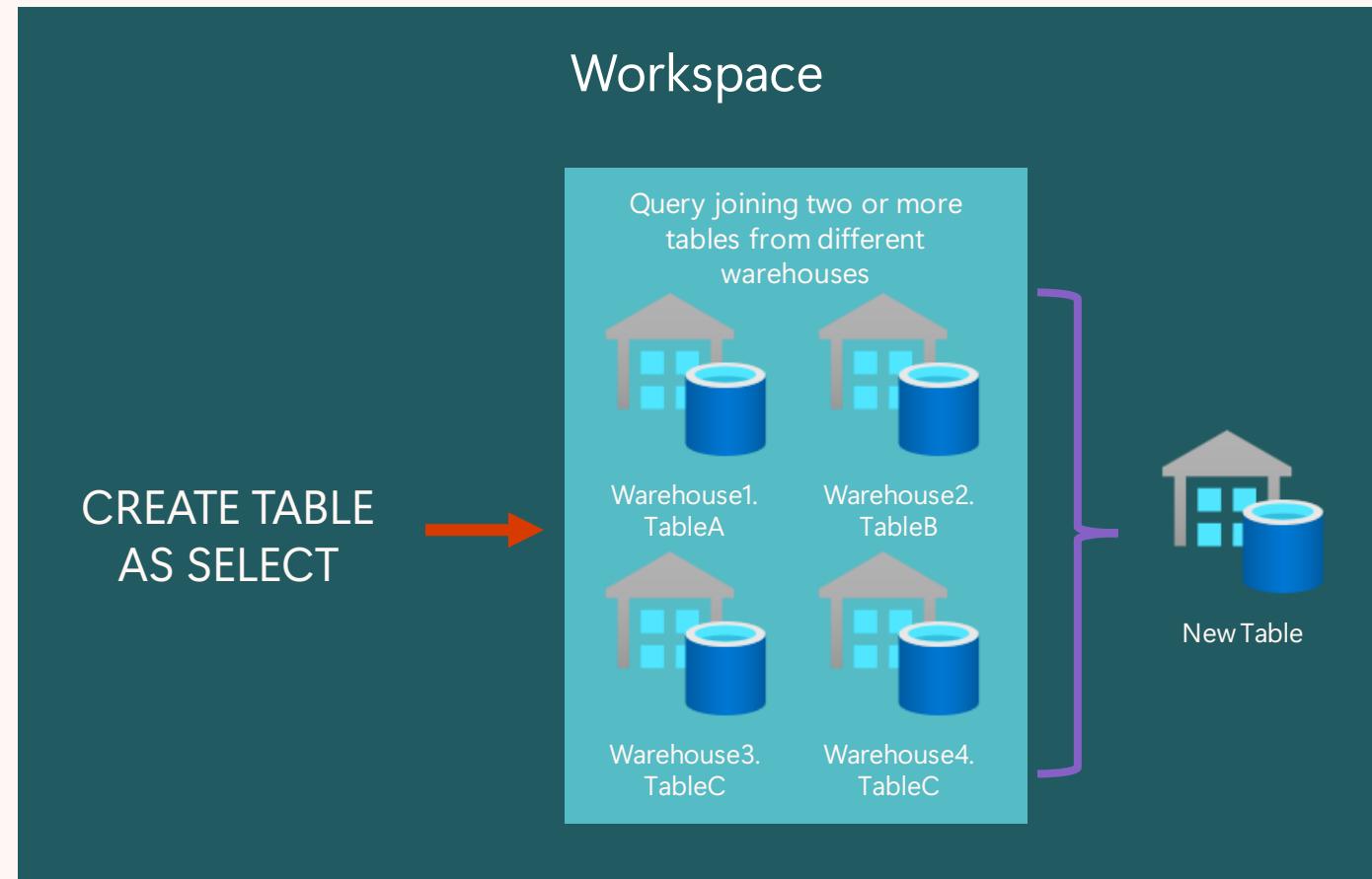
CREATE TABLE AS SELECT (CTAS)

A parallel operation that creates a new table based on the output of a SELECT statement. Cross-workspace queries are supported.

The simplest and fastest way to create and insert data into a table with a single command.

Example:

```
CREATE TABLE  
[dbo].[FactInternetSales_new]  
AS  
SELECT * FROM [dbo].[FactInternetSales];
```



INSERT...SELECT

For existing tables, INSERT...SELECT allows you to insert data from other tables using complex, cross-warehouse queries

Example:

```
INSERT INTO dbo.EmployeeSales
    SELECT 'SELECT', sp.BusinessEntityID, c.LastName, sp.SalesYTD
    FROM warehouseA.Sales.SalesPerson AS sp
    INNER JOIN warehouseB.Person.Person AS c
        ON sp.BusinessEntityID = c.BusinessEntityID
    WHERE sp.BusinessEntityID LIKE '2%'
    ORDER BY sp.BusinessEntityID, c.LastName;
```

Decision guide

	Pipelines	Dataflow Gen 2	T-SQL
Use case	Data lake and data warehouse migration, Data ingestion Light-weight transformation	Data transformation, data wrangling, data profiling Data ingestion	Data transformation, data processing, data profiling Data ingestion
Primary developer persona	Data engineer, data integrator	Data engineer, data integrator, business analyst	Data engineer, data developer
Primary developer skillset	ETL, SQL, JSON	ETL, M, SQL	T-SQL
Code written	No code Low code	No code Low code	Code
Data volume	Low to High	Low to High	Any
Development interface	Wizard Canvas	Power Query	Fabric Editor
Sources	100+ connectors	150+ connectors	ADLS, warehouse tables, lakehouse tables
Destinations	30+ connectors	Lakehouse Azure SQL database Azure Data explorer Azure Synapse analytics	Warehouse
Transformation complexity	Low: Lightweight - type conversion, column mapping, merge/split files, flatten hierarchy	Low to high: 300+ transformation functions	Low to high: Anything supported with T-SQL

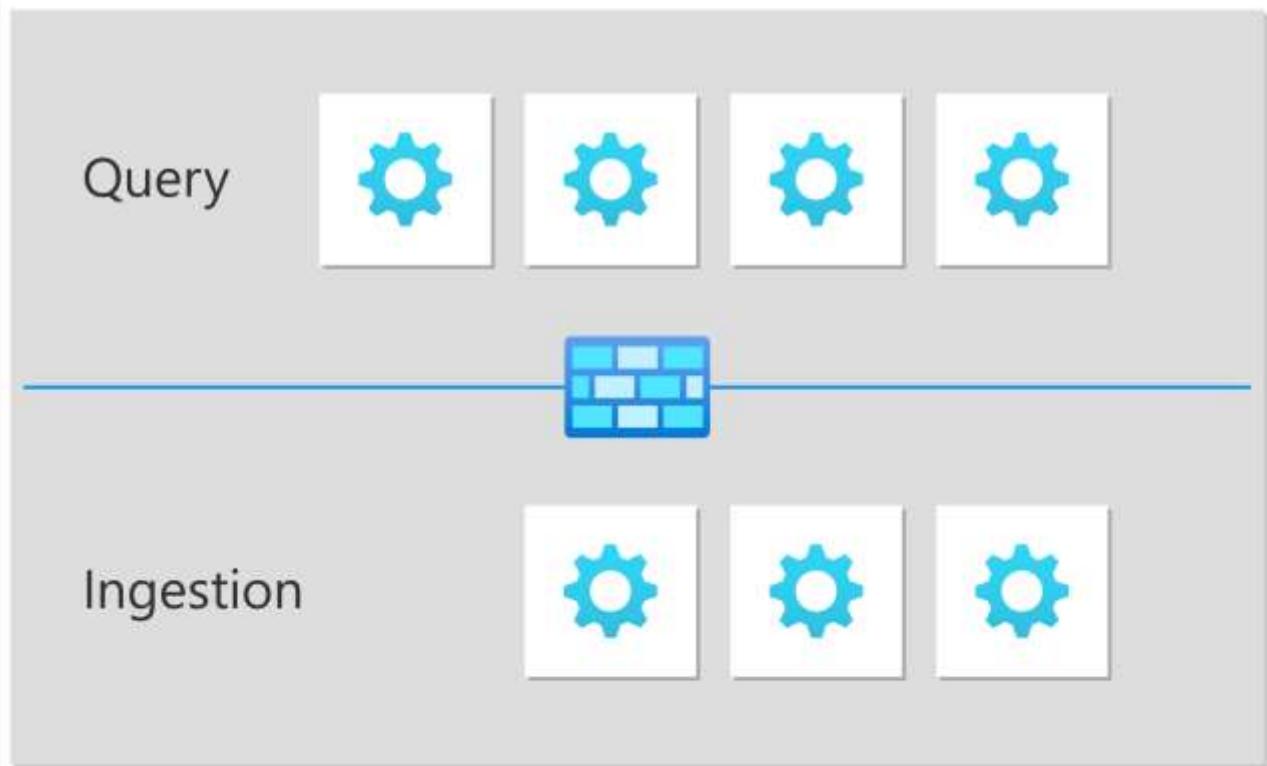
Performance considerations

Ingestion isolation

In the backend compute pool of Warehouse in Microsoft Fabric, loading activities are provided resource isolation from analytical workloads.

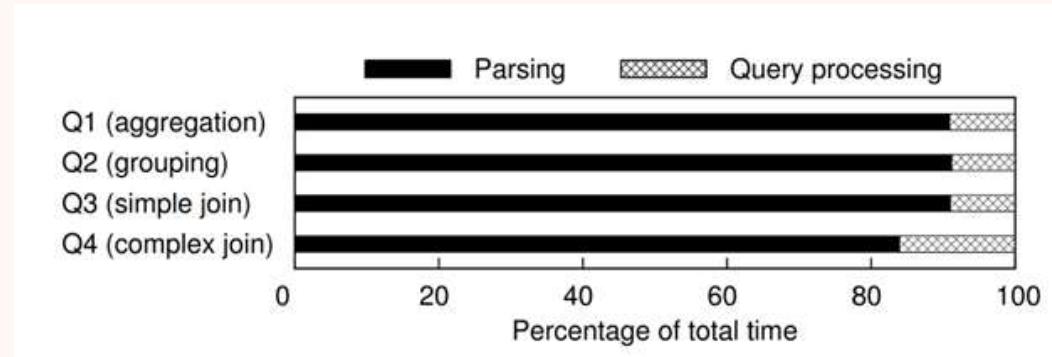
This improves performance and reliability, as ingestion jobs can run on dedicated nodes that are optimized for ETL and do not compete with other queries or applications for resources.

Backend pool



Leverage PARSE_R_VERSION 2.0 for CSV by avoiding fallback scenarios

Parsing non-binary data is a time-consuming operation



Source: Microsoft Research, "Mison: A Fast JSON Parser for Data Analytics"

COPY INTO with PARSE_R_VERSION 2.0 improves ingestion performance for CSV files.

Example:

```
COPY INTO destination_table  
FROM '[external account]'  
WITH (  
    FILE_TYPE = 'CSV',  
    ...  
    PARSER_VERSION = '2.0' --this is the default  
)
```

COPY INTO fallbacks to Parser 1.0 when:

- Compressed CSV files are used
- Files with UTF-16 encoding are used
- Multicharacter or multibyte ROWTERMINATOR, FIELDTERMINATOR, or FIELDQUOTE are provided.
 - However, '\r\n' is accepted as a default ROWTERMINATOR



Best practices

COPY INTO:

- Consider splitting large CSV or Parquet files, especially when the number of files is small
 - Using 1,000 files is 7x faster than using a single, large file with the same data volume
 - Keep file sizes to **at least** 4MB
- Parallel loads achieve higher throughput

All ingestion:

- Data ingestion scales with larger Fabric Capacities
 - If more throughput is needed, consider using a larger capacity and parallel loads
- Avoid singleton INSERT INTO statements (a.k.a.: trickle INSERTS) as they each work as a single ingestion operation and produce small Parquet files

Write T-SQL queries

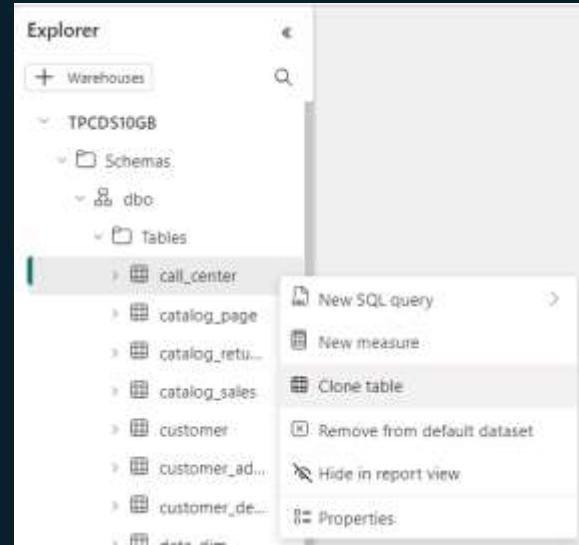
SQL Query Editor and Visual Query Editor

S: 1 min

Object Explorer

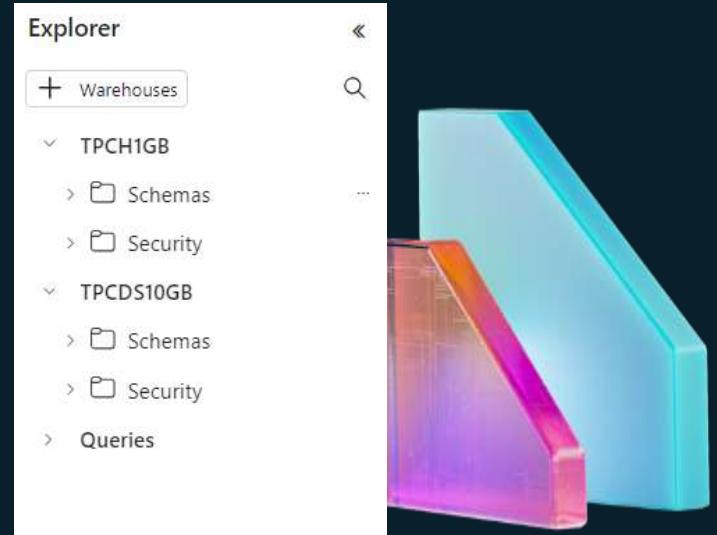
Templates based on existing objects

- Quickly create or drop existing objects from the Object Explorer using templates
- Preview Top 100 rows from tables and views with a click of a button
- Easy to clone tables in a warehouse using context menu option in Object explorer.



Add multiple warehouses

- Easy to add additional warehouses within current workspace to perform cross database querying.
- Can write a T-SQL query with three-part naming convention to refer to objects and join them across warehouses.



S: 1 min

Writing a SQL query

Templates to create new objects

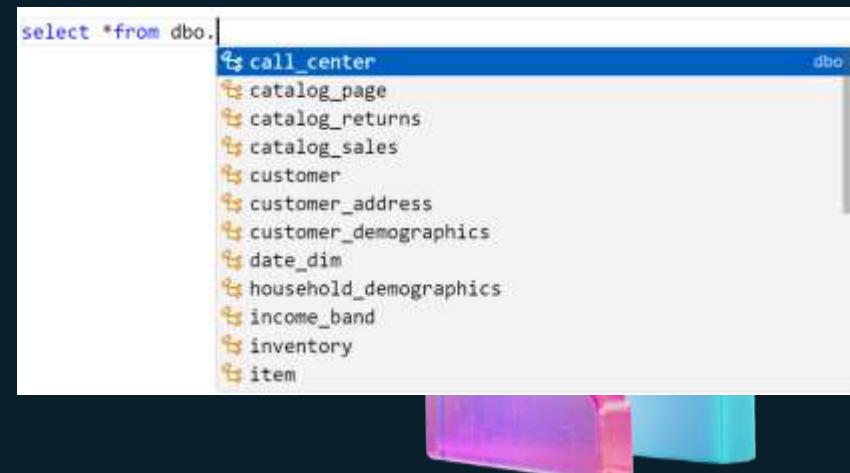
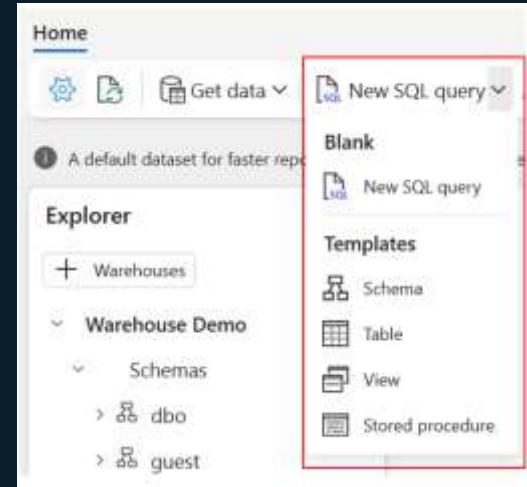
- Easy to create a new query using New SQL query -> Templates.
- Create T-SQL objects with code templates that populate in SQL query window.

IntelliSense and Keyword highlights

- SQL query editor provides support for IntelliSense, code completion, syntax highlighting

Keyboard Shortcuts

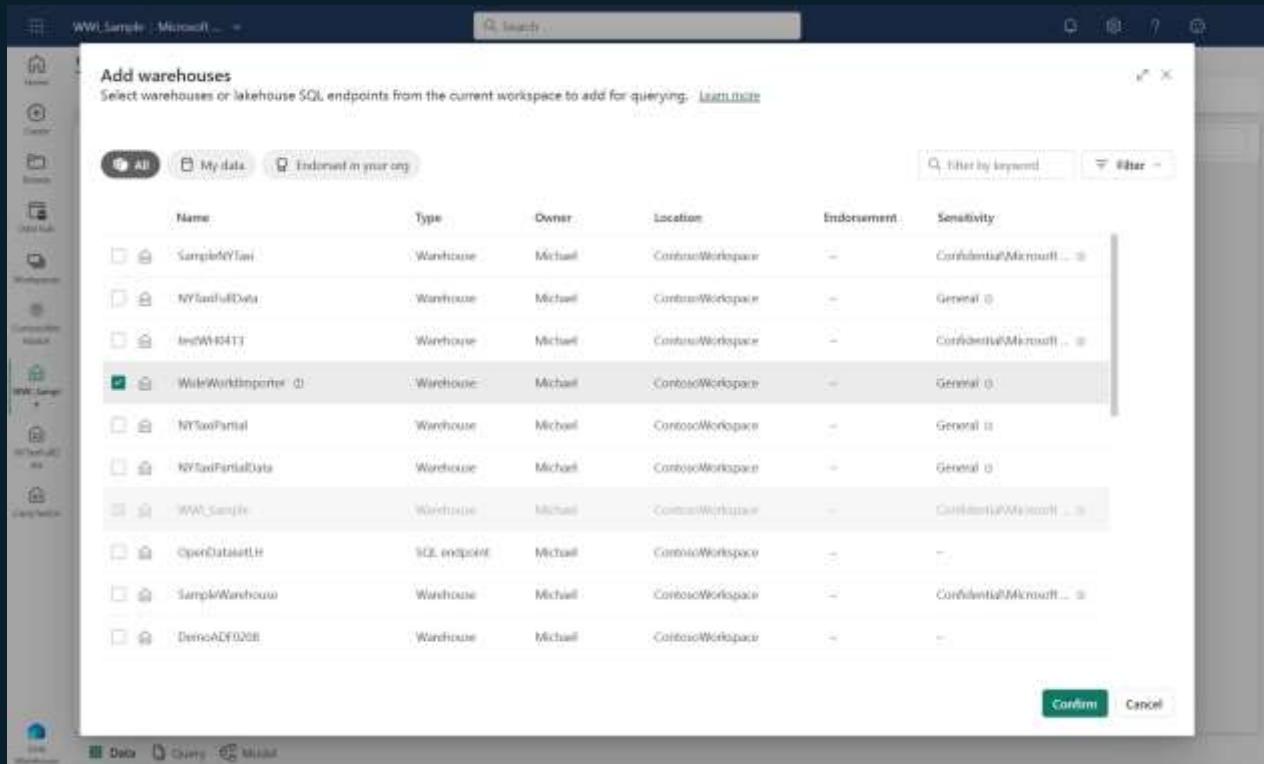
- Keyboard shortcuts provide a quick way to navigate and allow users to work more efficiently in SQL query editor. (Give a link). Different from SSMS.



S: 1 min

Cross database querying

- Easy to add additional warehouses within current workspace to perform cross database querying
- Create cross-warehouse queries via SQL query editor using 3-part naming syntax: [Warehouse].[Schema].[Table]
- Cross-warehouse queries can be used across SQL analytics endpoint and Warehouses in the current workspace



Viewing and analyzing your query results

Data grid

- View your SQL query results in an interactive data grid right within the query editor

	123 ws_sold_date_sk	123 ws_sold_time_sk	123 ws_ship_date_sk	123 ws_ship_time_sk	123 ws_bill_customer_sk
1	2451383	73313	2451482	4591	83074
2	2451383	73313	2451411	1566	83074
3	2451383	73313	2451413	7886	83074
4	2451383	73313	2451393	2755	83074
5	2451383	73313	2451502	2516	83074
6	2451383	73313	2451421	16966	83074
7	2451383	73313	2451457	10402	83074
8	2451383	73313	2451430	1735	83074
9	2451383	73313	2451458	15464	83074

Succeeded (17 sec 340 ms)

Open in Excel

- Analyze and refresh data from your SQL query directly inside Excel to ensure you are working with up-to-date data

Explore your data (preview)

- Perform ad-hoc exploration of your query results before building a report

Visualize Results

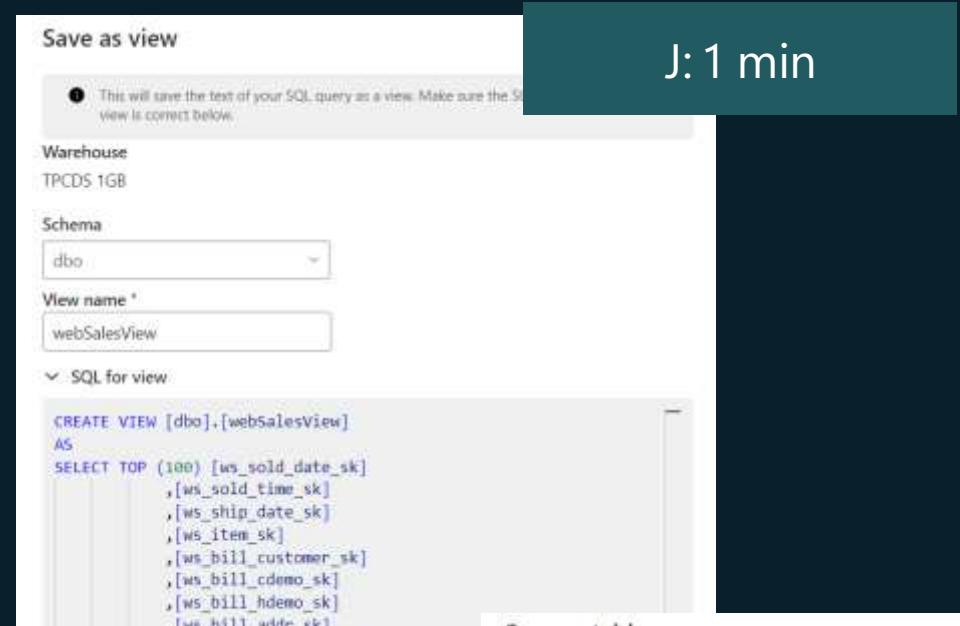
- Produce visual Power BI reports on top of your query results



Save as...

Save as View

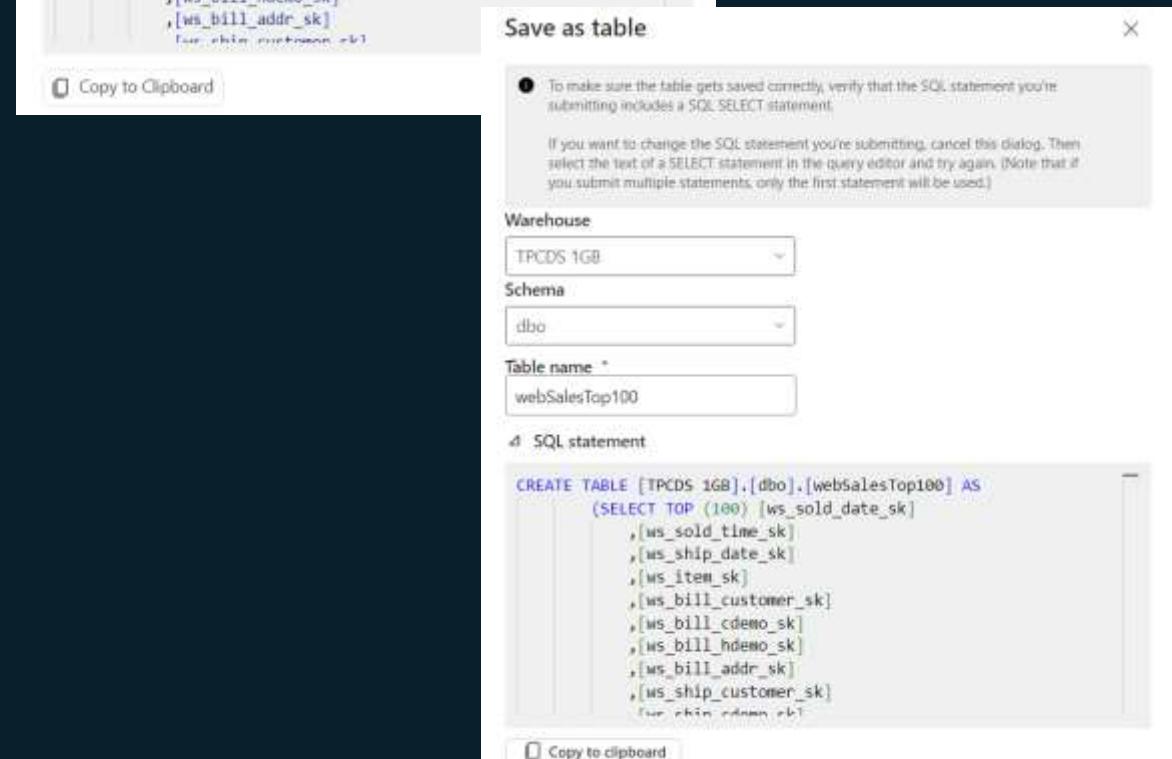
- Easy to create a view from selected SQL query text
- For the selected query, choose the desired schema and provide a name for the view



J: 1 min

Save as table

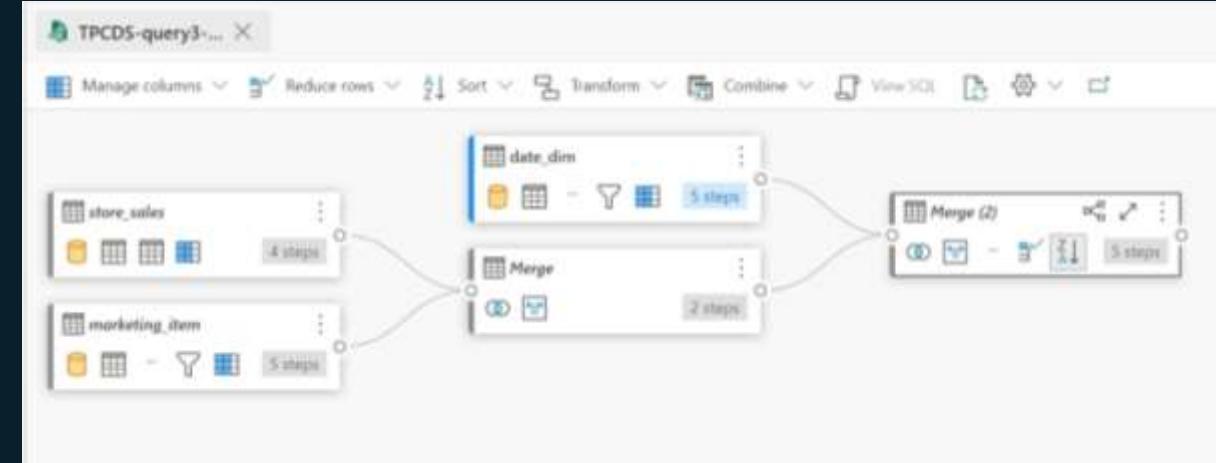
- Quickly save results of your SQL query into the table (CTAS) with just a simple click
- For the selected query, specify the warehouse, schema and table name to save results into the table



J: 1 min

Visual query editor

- No-code experience for citizen developers to analyze the data (leverages the familiar Power Query Diagram view)
- Simply drag-drop tables into the Visual query editor and apply steps to get results for your analysis
- Create cross-warehouse queries through drag/drop of tables from respective Warehouse/SQL analytics endpoint and apply merge condition



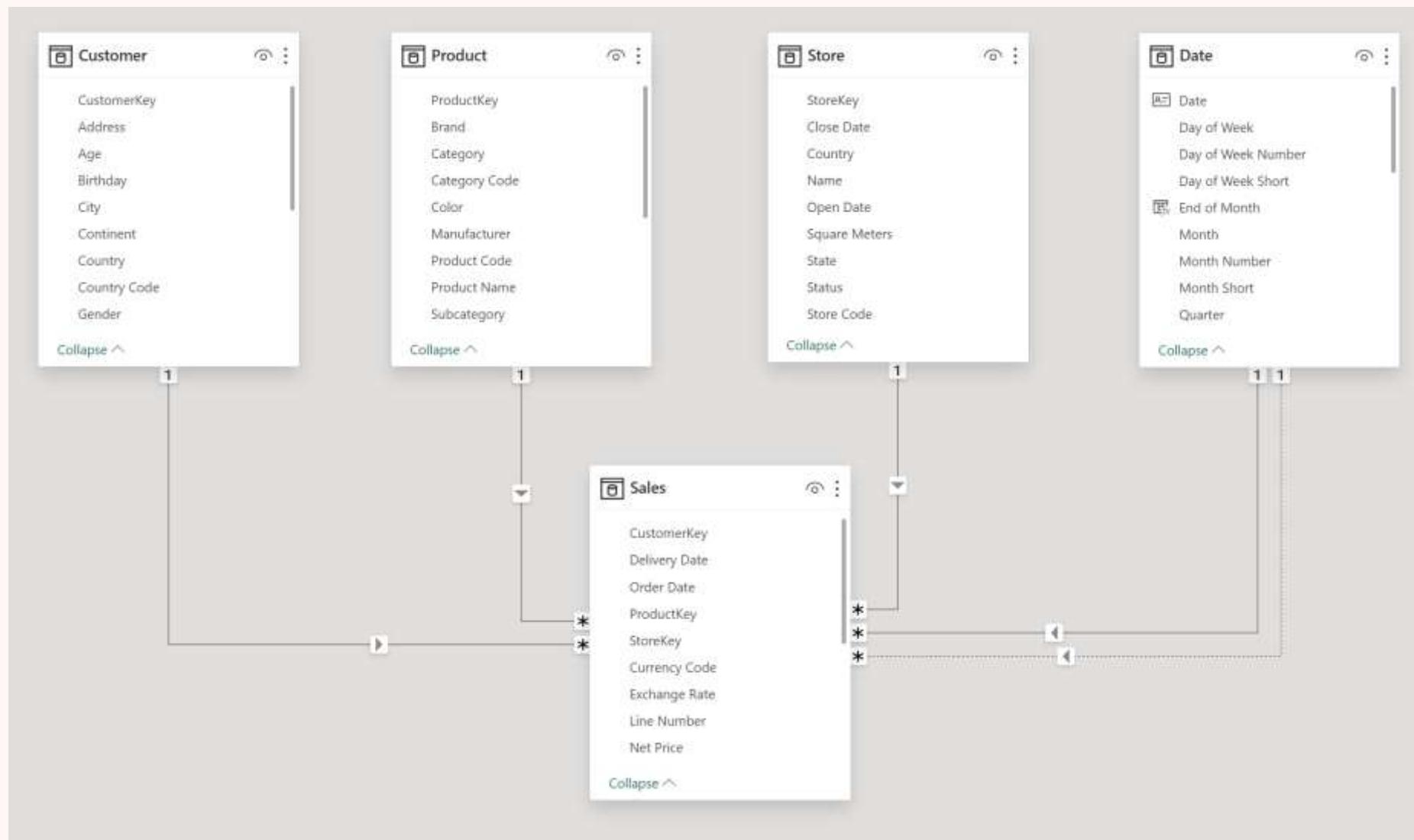
Semantic model, Not just a dataset

Data: only tables

The screenshot shows the Power BI Data view with five tables listed:

- Customer**: Contains fields like Address, Age, Birthday, City, Continent, Country, Country Code, CustomerKey, Gender, and a Collapse button.
- Product**: Contains fields like Brand, Category, Category Code, Color, Manufacturer, Product Code, Product Name, ProductKey, and Subcategory, along with a Collapse button.
- Store**: Contains fields like Close Date, Country, Name, Open Date, Square Meters, State, Status, Store Code, StoreKey, and a Collapse button.
- Date**: Contains fields like Date, Day of Week, Day of Week Number, Day of Week Short, End of Month, Month, Month Number, Month Short, and Quarter, along with a Collapse button.
- Sales**: Contains fields like Currency Code, CustomerKey, Delivery Date, Exchange Rate, Line Number, Net Price, Order Date, Order Number, ProductKey, and a Collapse button.

Dataset: tables + relationships + more...



Semantic model

Tables Model

Search

▼ Semantic model

- Calculation groups (0)
- ▼ Cultures (1)
 - en-US
- ▼ Measures (5)
 - Margin
 - Margin %
 - Sales Amount

▼ Relationships (5)

- Sales[CustomerKey] ←
- Sales[Delivery Date] ← Date[Date]
- Sales[Order Date] ← Date[Date]
- Sales[ProductKey] ← Product[ProductKey]
- Sales[StoreKey] ← Store[StoreKey]

Roles (0)

File Home Help External tools Table tools Measure tools

Name: Margin % Format: \$% Data category: Uncategorized

Home table: Sales \$ % , .00 2

Structure Formatting Properties

20 X ✓ 1 Margin % = DIVIDE ([Margin], [Sales Amount])

Properties

General

Formatting

Advanced

Sort by column

Month Number

Tables (5)

- Customer
- Columns (13)
- Hierarchies (1)
- Geography
 - City
 - Continent
 - Country
 - State

Measures (0)

- Partitions (1)
- Date
- Columns (17)
- Hierarchies (2)
- Year-Month
- Year-Quarter-Month

Measures (0)

- Partitions (1)
- Product
- Sales
- Store

Dynamic calculations embedded in the model

- Model-level semantics:
 - Calculated columns
 - Calculated tables
- Report-level semantics:
 - Measures
 - Calculation groups
- Composable language: DAX

Layout features

Layout features

- Format string
 - Static on columns
 - Static or dynamic on measures
- Hierarchies
- Sort column by
 - Month name
 - Day name
 - Account name by section
- Folders
- KPI

SQL vs. DAX

Sales by Month (Order Date)

DAX

EVALUATE

```
SUMMARIZECOLUMNS (
    'Date'[Calendar Year Month],
    'Date'[Calendar Year Month Number],
    "Sales Amount", [Sales Amount],
    "Margin", [Margin],
    "Margin %", [Margin %]
```

)

ORDER BY

```
'Date'[Calendar Year Month Number]
```

SQL

SELECT

```
d.[Year Month],
d.[Year Month Number],
SUM ( s.Quantity * s.[Net Price] ) AS SalesAmount,
SUM ( s.Quantity * (s.[Net Price] - s.[Unit Cost]) )
     AS Margin,
SUM ( s.Quantity * (s.[Net Price] - s.[Unit Cost]) )
     / SUM ( s.Quantity * s.[Net Price] ) AS [Margin %]
FROM Sales s
LEFT JOIN [Date] d
    ON d.[Date] = s.[Order Date]
GROUP BY d.[Year Month Number], d.[Year Month]
ORDER BY d.[Year Month Number]
```

Commission by Price Range

DAX

```
EVALUATE  
SUMMARIZECOLUMNS (  
    'Price Ranges'[Price Range],  
    'Price Ranges'[PriceRangeKey],  
    "Margin", [Margin],  
    "Margin %", [Margin %],  
    "% of Margin", [% of Margin]  
)  
ORDER BY 'Price Ranges'[PriceRangeKey]
```

SQL

```
),  
CommissionByPriceRange AS (  
    SELECT  
        pr.PriceRangeKey,  
        pr.PriceRange,  
        SUM ( s.Quantity * s.[Net Price] ) AS SalesAmount,  
        SUM ( s.Quantity * (s.[Net Price] - s.[Unit Cost]) ) AS Profit  
    FROM Sales s  
    LEFT JOIN PriceRange pr  
        ON pr.[Min] <= s.[Net Price] AND pr.[Max] > s.[Net Price]  
    GROUP BY pr.PriceRangeKey, pr.PriceRange  
)  
SELECT  
    PriceRange,  
    Margin,  
    Margin / SalesAmount AS [Margin %],  
    Margin / (SELECT SUM ( Margin ) FROM CommissionByPriceRange )  
FROM CommissionByPriceRange  
ORDER BY PriceRangeKey;
```

Security features

Row-level security (RLS)

- Static
 - All the role members have access to the same data
 - The security perimeter does not depend on the user, only on the role
- Dynamic
 - DAX expression based on connected user
 - Can be data-driven, based on specific configuration tables in the model

Object-level security (OLS)

- Define tables/columns that are not visible (restricted)
- Restrictions propagate to any calculation
- Example: hides sensitive data (SSN)

Modeling features

Different types of many-to-many relationships (M2M)

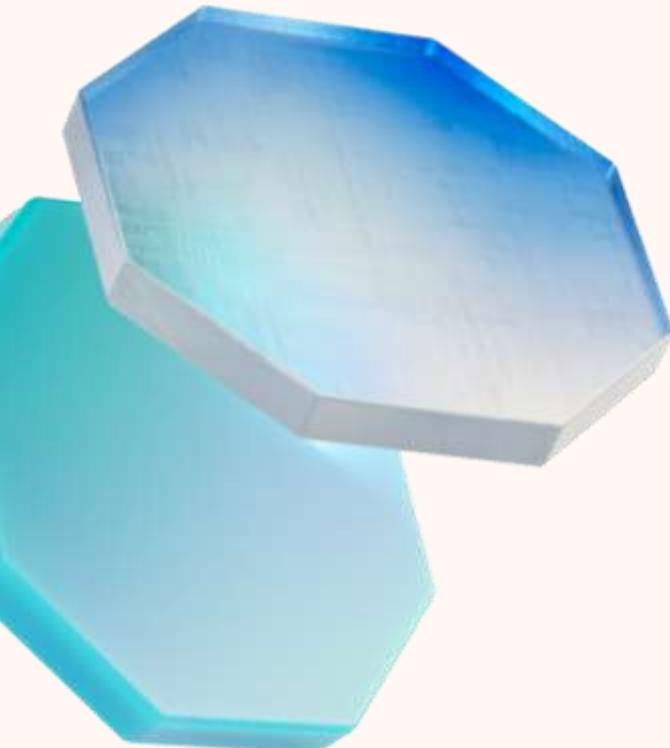
- M2M relationship between business entities (dimensions)
 - Use a bridge table from the data source
 - Apply a bidirectional filter on one relationship to enable non-additive calculations through M2M relationship
 - Example: bank accounts and customers who are account owners)
- M2M cardinality relationship
 - Propagate a relationship at a different granularity
 - Specify single direction for filter propagation (do not use bidirectional filters)
 - Example: budget by brand / country, dimensions products / customers

Composite models

Composite models

- Create a semantic model based on an existing one
 - No data duplication
 - Model artifacts are just a reference (proxy) to the remote model
- The local model can:
 - Add local tables
 - Add relationships, also between local and remote tables
 - Add measures, with automatic remote execution whenever possible
 - Add calculation groups
 - Rename existing objects (tables, columns, measures)
- The composite model can be published
 - Refresh needed for local tables only - including calculated tables

Conclusions

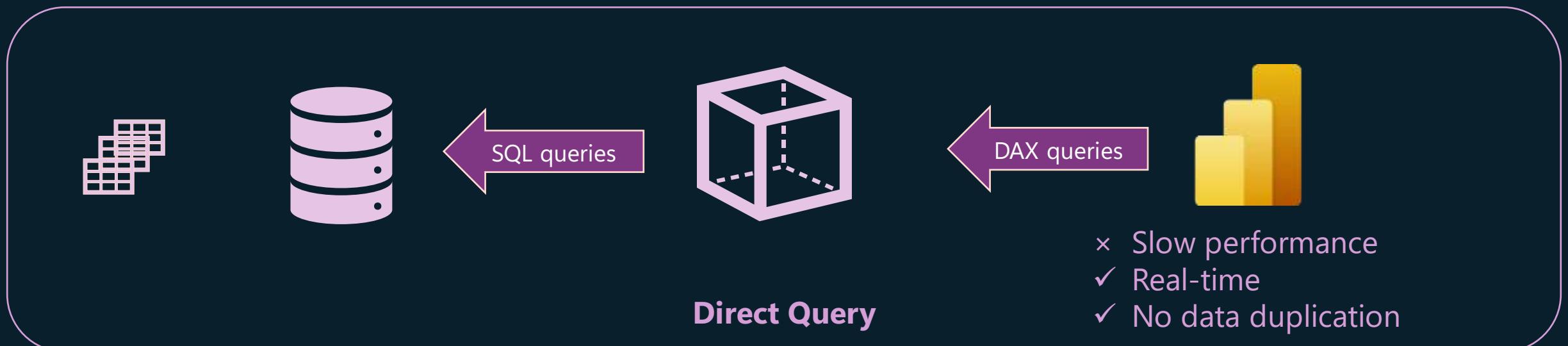
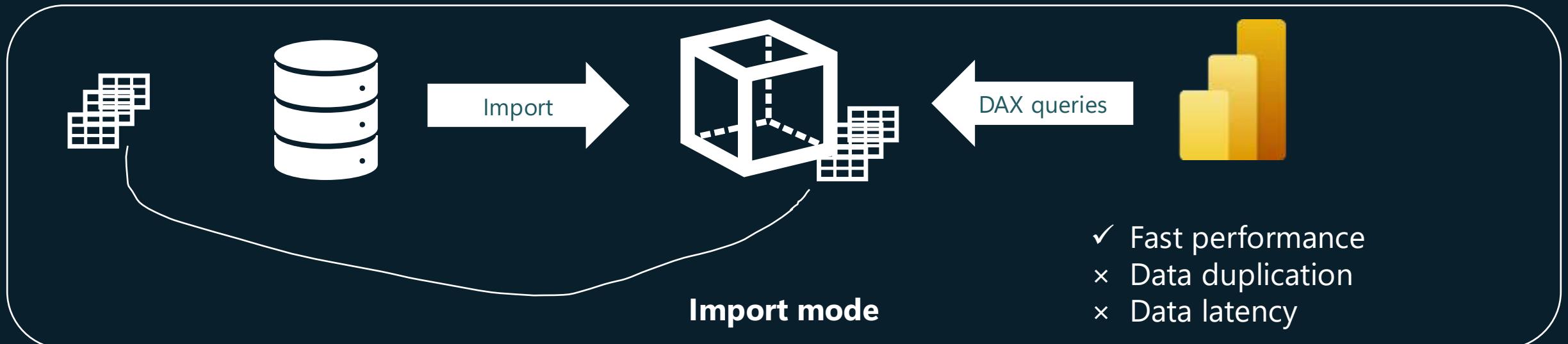


Adopt semantic models

- Control data quality
- Simplify data consumption
- Improve the productivity of report authors
- Extend certified models for task-specific specialized models
 - Leverage self-service BI by reusing corporate BI assets
- It was called dataset,
but it has always been a semantic model

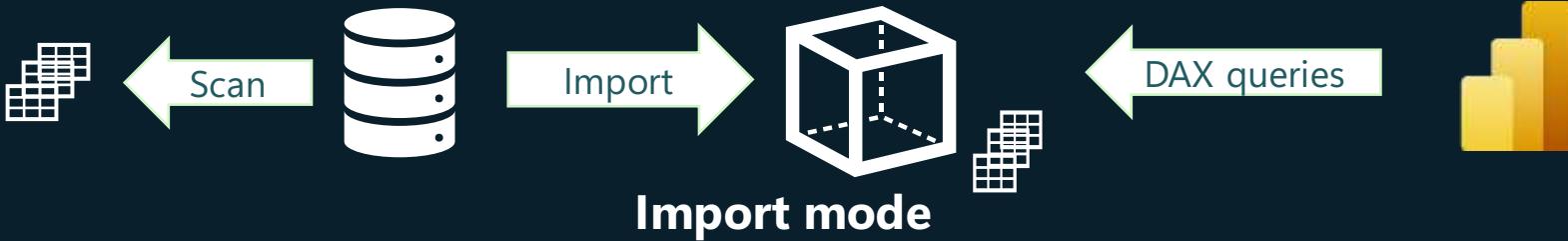
Power BI Semantic models: Storage modes

Power BI Architecture – Pre-Fabric

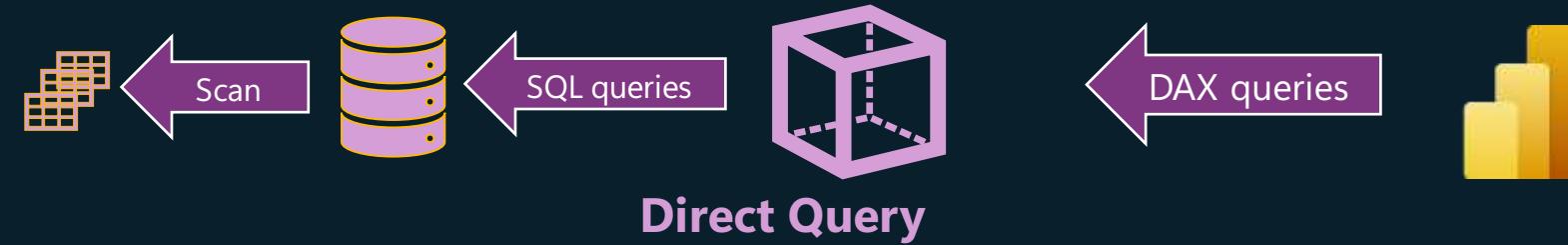


Power BI Architecture – Fabric

- ✓ Fast performance
- ✗ Data duplication
- ✗ Data latency



- ✗ Slow performance
- ✓ Real-time
- ✓ No data duplication



Direct Lake

Direct Lake mode

Revolutionary feature!

✓ Prerequisites

- ✓ Fabric F Capacity/Power BI Premium
- ✓ Lakehouse + SQL Endpoint (for DQ fallback)/Warehouse
- ✓ Delta tables
- ✓ V-Ordering*

* V-Ordering

Fabric-specific way of additionally optimizing Parquet files when writing data

Diving directly into a lake!



This Photo by Unknown Author is licensed under CC BY-NC

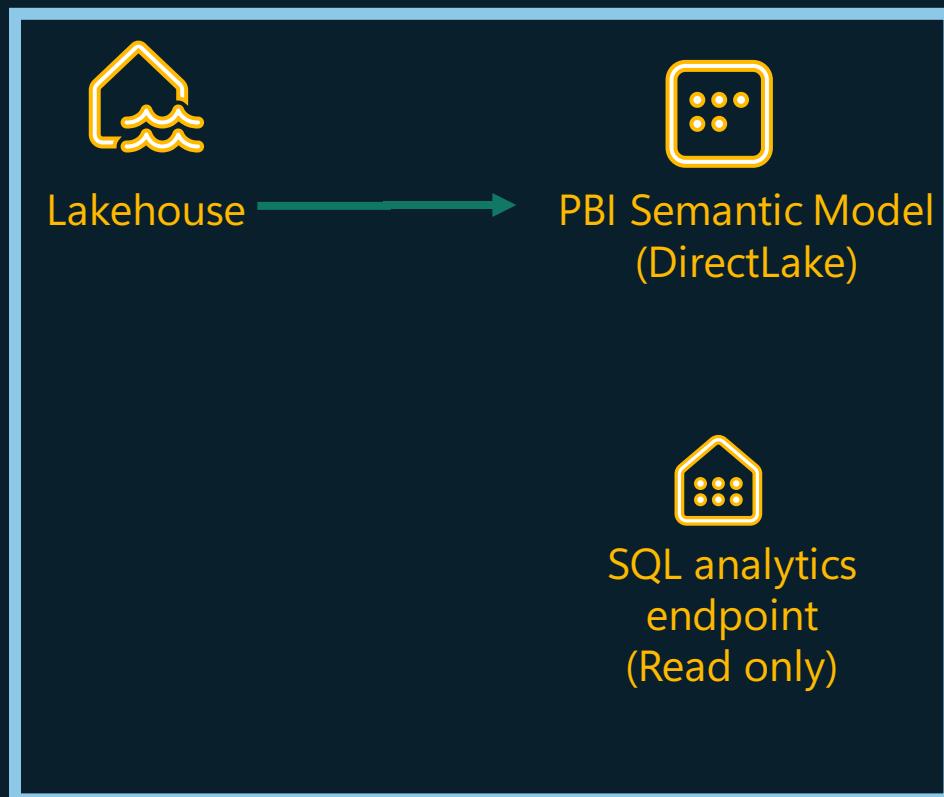
All engines read/write Delta files!



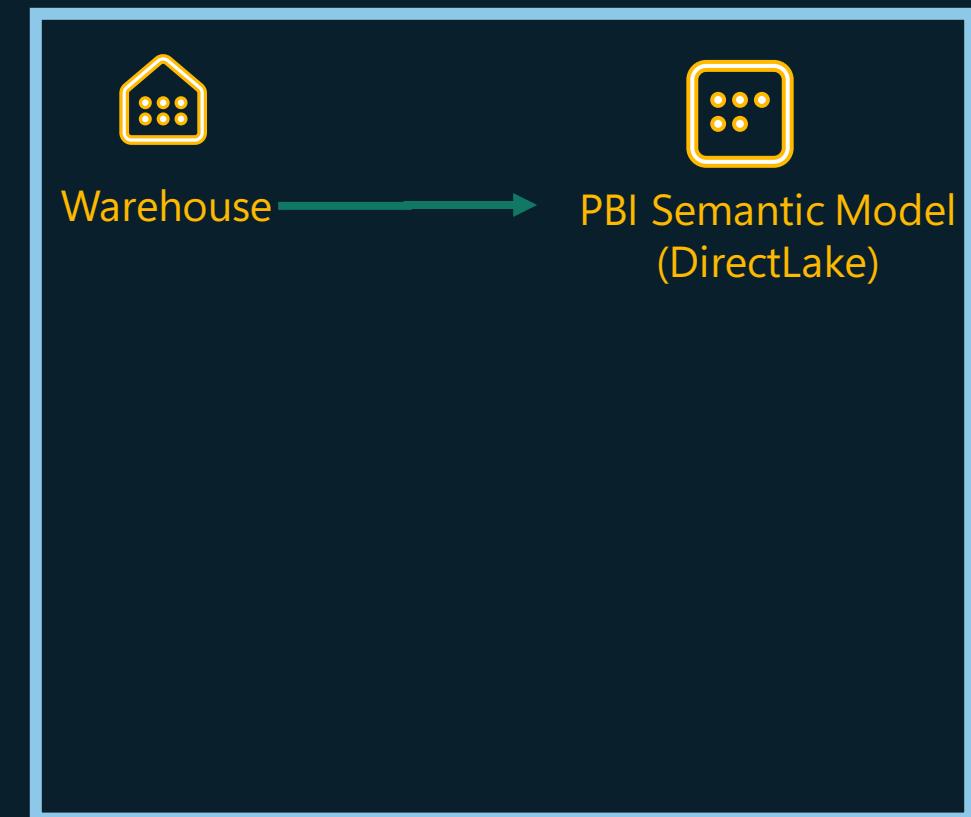
[Explore end-to-end analytics with Microsoft Fabric - Training | Microsoft Learn](#)

Default Power BI semantic model in DirectLake!

Lakehouse

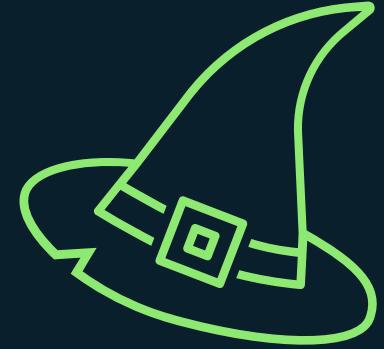


Warehouse



How does this magic work?

- Semantic model is initially empty
- Loads data into memory on-demand quickly
- Parquet is very similar to how Power BI stores its data in import mode (in VertiPaq)
- Power BI engine (Analysis Services) reads from Delta tables directly
- Refreshes just the parts it needs on-demand
- Direct Lake uses the SQL endpoint to discover schema and security information



Paging

- Use DMVs to see data paging / temperature:

```
select * from $SYSTEM.DISCOVER_STORAGE_TABLE_COLUMNS  
Order by dictionary_Last_Accessed Desc
```

```
select * from $SYSTEM.DISCOVER_STORAGE_TABLE_COLUMN_SEGMENTS  
Order by Last_Accessed Desc
```

TODAY(): Only works against the custom semantic model!

Semantic Model Refresh

Refreshing the data model

- Framing!
- Reads the Delta Log
- Metadata only

The screenshot shows a data visualization interface with a code editor at the top and a table view below. The code editor contains the following SQL command:

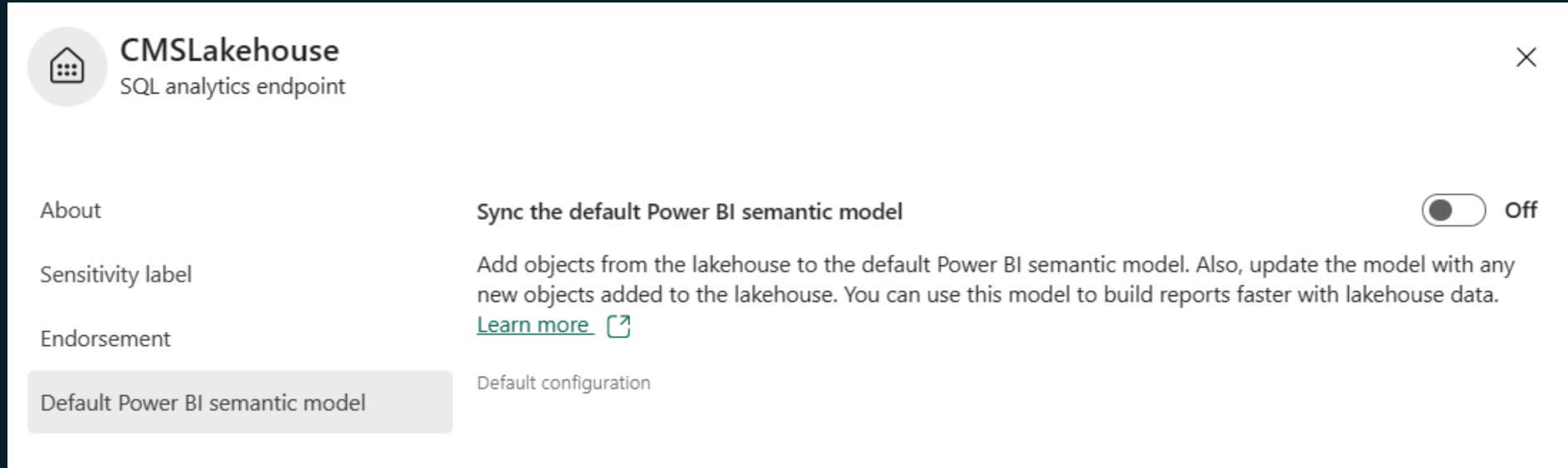
```
1 %%sql
2
3 DESCRIBE HISTORY cms_provider_drug_costs
```

A green checkmark indicates the command was executed successfully in 2 seconds by shabnam watson on 7:18:58 PM, 3/09/24. The interface includes tabs for 'Spark jobs (1 of 1 succeeded)' and 'Log'. Below the table, there are navigation controls for rows and columns.

Table View:

	12L version	⌚ timestamp	ABC userId	ABC userName	ABC operation
1	8	2024-03-10T00:11:01Z	NULL	NULL	WRITE
2	7	2024-03-10T00:08:03Z	NULL	NULL	WRITE
3	6	2024-03-10T00:05:03Z	NULL	NULL	WRITE
4	5	2024-03-10T00:02:01Z	NULL	NULL	WRITE
5	4	2024-03-09T23:59:00Z	NULL	NULL	WRITE
6	3	2024-03-09T23:56:02Z	NULL	NULL	WRITE
7	2	2024-03-09T23:53:06Z	NULL	NULL	WRITE
8	1	2024-03-09T23:50:13Z	NULL	NULL	WRITE
9	0	2024-03-09T23:47:25Z	NULL	NULL	WRITE

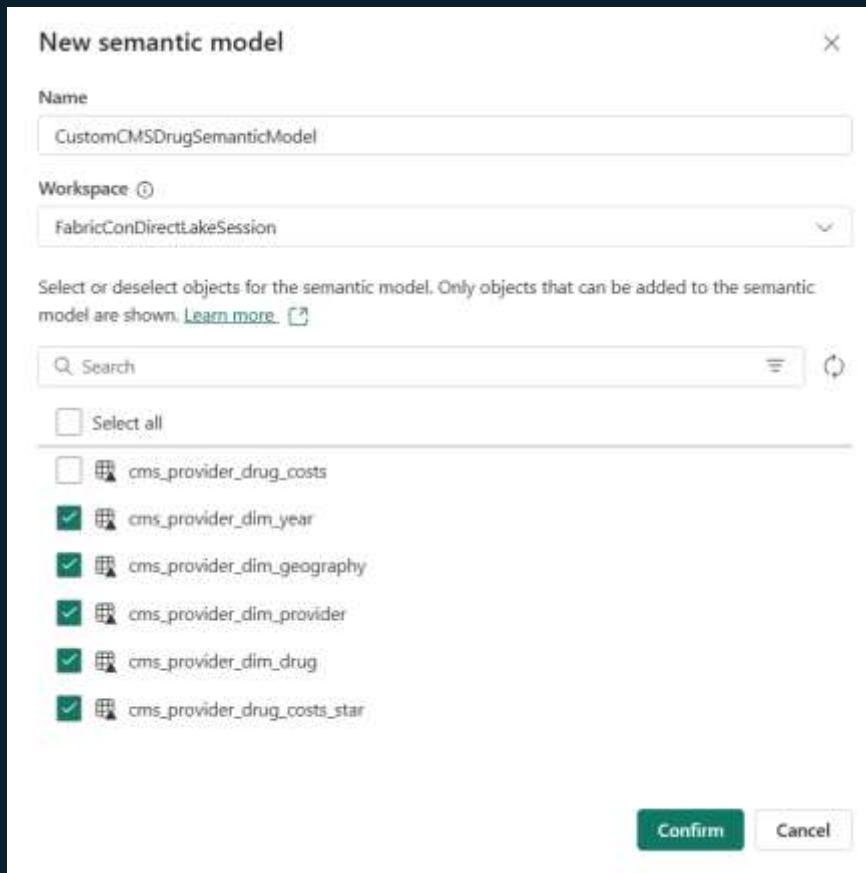
Syncing from lakehouse for default semantic model



- If set to off (default), you need a refresh to add new tables.

Custom Semantic Model

Custom Semantic Model



Not based on relationships in the SQL Endpoint

The screenshot shows the 'FabricConDirectLakeSession' dashboard. At the top, there are navigation links: '+ New', 'Upload', 'Create deployment pipeline', 'Create app', and 'More'. The main area displays a table of semantic models:

Name	Type
CMSLakehouse	Lakehouse
CMSLakehouse	Semantic model (default)
CMSLakehouse	SQL analytics endpoint
CustomCMSDrugSemanticModel	Semantic model

Refresh Options for default semantic model

Settings for CMSLakehouse

[View semantic model](#)

[Refresh history](#)

▷ Query Caching

▫ Refresh

Keep your Direct Lake data up to date

Configure Power BI to detect changes to the data in OneLake and automatically update the Direct Lake tables that are included in this semantic model. [Learn more](#)

 On

[Apply](#) [Discard](#)

Custom Semantic Model

- Refresh can be scheduled
- Schema changes are not updated automatically
- Spark notebooks (SemPy) or Dataflows Gen2

Fallback to Direct Query

Fallback to DirectQuery

RLS/OLS in Warehouse
Views



DirectLake Guardrails



Fabric/Power BI SKUs	Parquet files per table	Row groups per table	Rows per table (millions)	Max model size on disk/OneLake ¹ (GB)	Max memory (GB)
F2	1,000	1,000	300	10	3
F4	1,000	1,000	300	10	3
F8	1,000	1,000	300	10	3
F16	1,000	1,000	300	20	5
F32	1,000	1,000	300	40	10
F64/FT1/P1	5,000	5,000	1,500	Unlimited	25
F128/P2	5,000	5,000	3,000	Unlimited	50
F256/P3	5,000	5,000	6,000	Unlimited	100
F512/P4	10,000	10,000	12,000	Unlimited	200
F1024/P5	10,000	10,000	24,000	Unlimited	400
F2048	10,000	10,000	24,000	Unlimited	400

Max Memory = memory resource limit for how much data can be paged in for each query

Max model size on disk/Onelake = limit beyond which all queries fall back to DirectQuery

DMV for DirectQuery fallback reason

- \$SYSTEM.TMSCHEMA_DELTA_TABLE_METADATA_STORAGES
- Zero is good, anything else means fallback to DirectQuery

The screenshot shows a database query results grid. At the top, there is a SQL query: `1 SELECT * FROM $SYSTEM.TMSCHEMA_DELTA_TABLE_METADATA_STORAGES`. Below the query, there are three tabs: Log, Results (which is selected), and History. The Results tab displays a table with six columns: otLocation, CurrentVersion, TableObjectID, DatamartObjectID, FramedSchemaName, and FallbackReason. There are five rows of data, each corresponding to a different table location. All rows have a CurrentVersion of 0 and a FallbackReason of 0.

otLocation	CurrentVersion	TableObjectID	DatamartObjectID	FramedSchemaName	FallbackReason
'97f85e-dbff-4070-a2c5-a0083e2dce29/9fc14bda-c43a-448f-ae33-2f73857a9f88/Tables/cms_provider_dim_year/	0	06ce4911-751f-4fe5-add7-d4d9ac3b4228	e541e52f-ef04-4637-8902-b1f3d1d7d005	dbo	0
'97f85e-dbff-4070-a2c5-a0083e2dce29/9fc14bda-c43a-448f-ae33-2f73857a9f88/Tables/cms_provider_dim_geography/	0	6c322592-39dc-4fdd-b729-dc8a59e7af3f	e541e52f-ef04-4637-8902-b1f3d1d7d005	dbo	0
'97f85e-dbff-4070-a2c5-a0083e2dce29/9fc14bda-c43a-448f-ae33-2f73857a9f88/Tables/cms_provider_dim_provider/	0	e2e600c3-dfdb-4cbd-938b-49c02c43af87	e541e52f-ef04-4637-8902-b1f3d1d7d005	dbo	0
'97f85e-dbff-4070-a2c5-a0083e2dce29/9fc14bda-c43a-448f-ae33-2f73857a9f88/Tables/cms_provider_dim_drug/	0	1ecadce2-26d5-4a9f-9e6e-c07a84387aa7	e541e52f-ef04-4637-8902-b1f3d1d7d005	dbo	0
'97f85e-dbff-4070-a2c5-a0083e2dce29/9fc14bda-c43a-448f-ae33-2f73857a9f88/Tables/cms_provider_drug_costs_star/	0	7abceec4-017e-449f-8329-6ee161e01114	e541e52f-ef04-4637-8902-b1f3d1d7d005	dbo	0

DirectLakeBehavior property



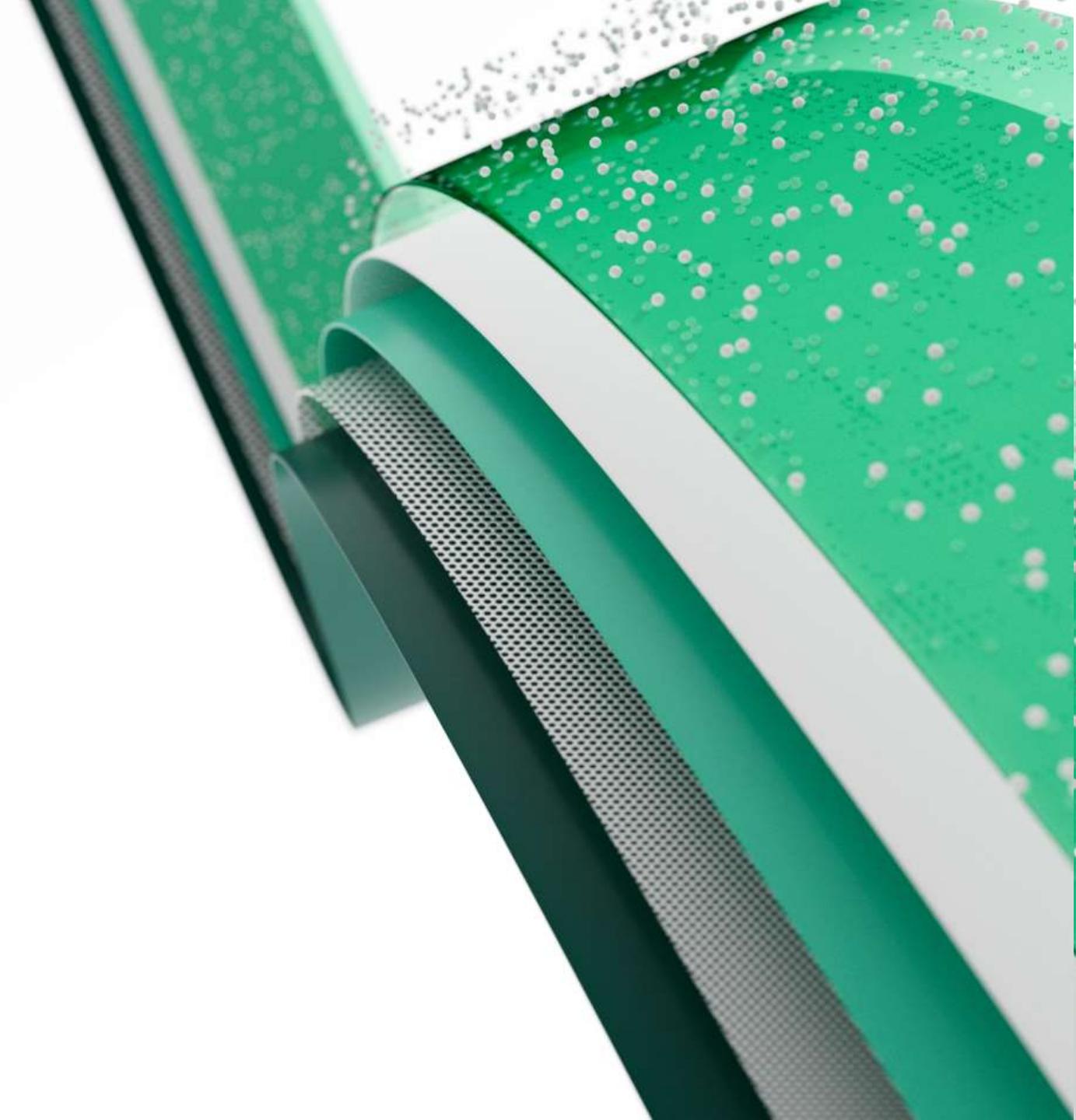
Can be configured by using Tabular Object Model (TOM) or Tabular Model Scripting Language (TMSL)

Is a visual using DirectLake or DirectQuery?

Performance analyzer in Power BI Desktop, SQL Server Profiler

If you see DirectQuery in Performance Analyzer

Practice test





Live Practice test

Access Wifi

SSID:

Event Code:

Access the test
Aka.ms/fabric-cert-day

Prepare for exam day



What to expect?

40-60 questions

Plan for 180 minutes

Question types include more than multiple choice

Case studies require complex decision making

Tip: You can check your technical setup in advance

Checkin

Plan for 20-30 minutes for this step

Have an official ID nearby

Nothing on your desk

Unplug all second monitors

Have your smartphone available for checkin

Choose a calm place

No glassed windows if possible

Let's take an exam

Exam scoring

Exam cut score (700 on a scale of 0-1000). != **70%**

No points deducted for wrong answers

Mark questions for review

Don't overthink it

Some questions are worth more than one point

Answer every question!

Use Microsoft Learn during exam

Question 1 (of 10)

Review Leave Feedback (after answering all questions)

This question type is known as a multiple choice; select one. This question type contains multiple answer choices and has one correct answer.

There are several varieties of this question type. The number of answer choices may vary, and questions may appear as part of a series or set.

In this question type, you respond to the question by selecting the radio or option button next to your answer.

Here is a sample question:

What is your favorite sound?

A. Bell
 B. Bed
 C. River
 D. Whistle

?

Help

Calculator

Color Scheme

Reset Answer

Take a Break

Microsoft Learn

Previous

Next

Time remaining: 07:52:35

Overall exam progress: 1 of 10 total questions

Microsoft Learn: x

Microsoft | Learn Documentation Training Certifications Q&A Code Samples More Search Sign in

Microsoft Learn. Spark possibility.

Build skills that open doors. See all you can do with documentation, hands-on training, and certifications to help you get the most from Microsoft products.

Search Microsoft Learn

Search

Learn by doing

Gain the skills you can apply to everyday situations through hands-on training personalized to your needs, at your own pace or with our global network of learning partners.

Take training

Find technical documentation

Get tools and step-by-step guidance to help you get the most from Microsoft products such as Azure, Windows, Office, Dynamics, Power Apps, Teams, and more.

View documentation

Showcase your skills

Advance in your career by completing challenges that demonstrate your expertise. Earn globally recognized and industry-endorsed certifications, and showcase them to your network.

Get certified

Ask questions and get help

Connect with other learners and experts, ask and answer questions, share resources, and learn together.

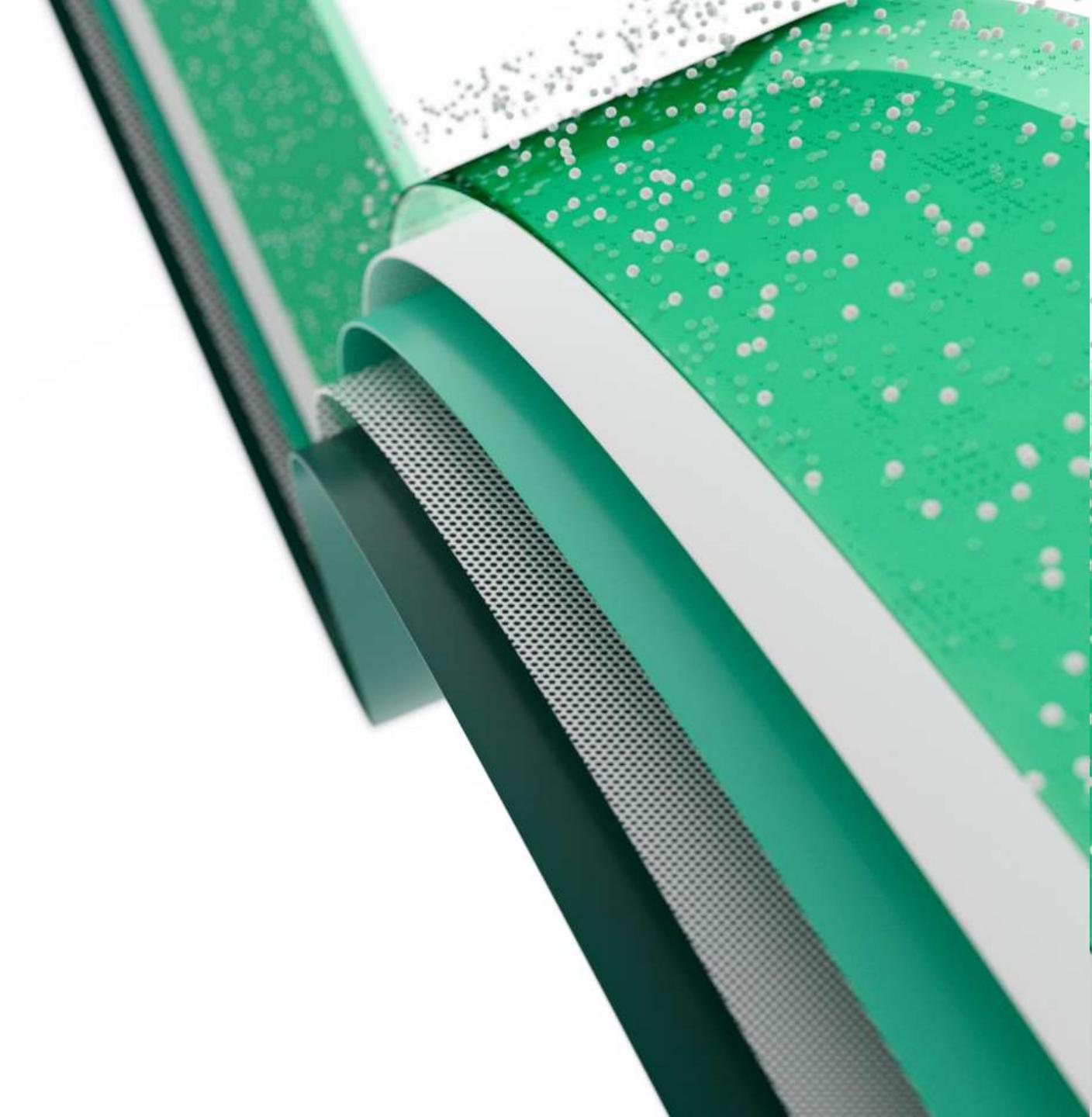
Try code samples

Test out new capabilities in your own projects faster and easier with code samples that bring Microsoft technology to life.

See new ways to innovate

Discover thousands of fun, authentic, and informative videos by Microsoft and community experts that help you and your team find inventive ways to use technology.

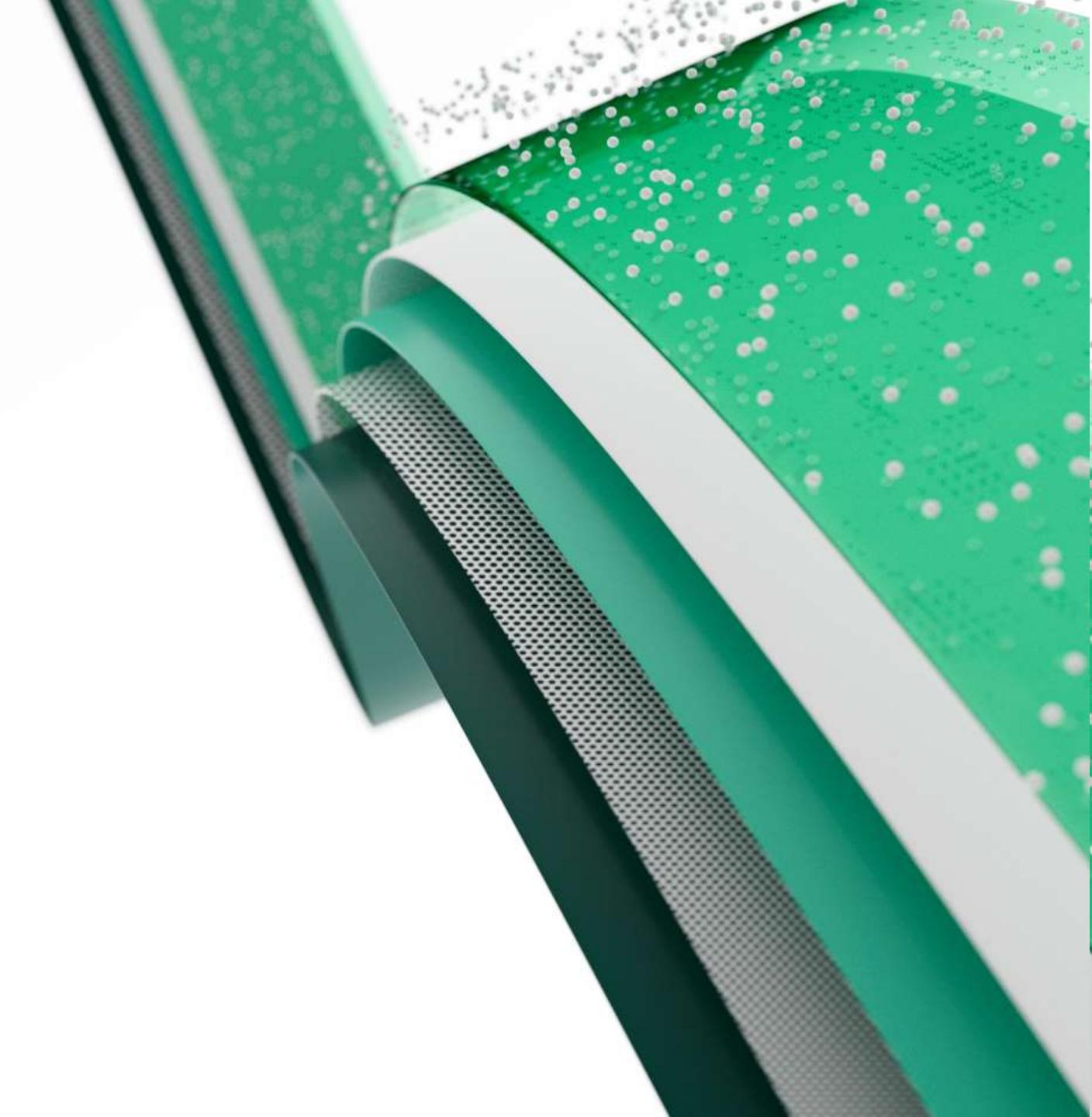
Last words



If you pass the exam

Do not hesitate to share your success on LinkedIn and tag us (Frédéric Gisbert and Christopher Maneu) ☺

Q&A



Thank you

<Presenter
Company name>