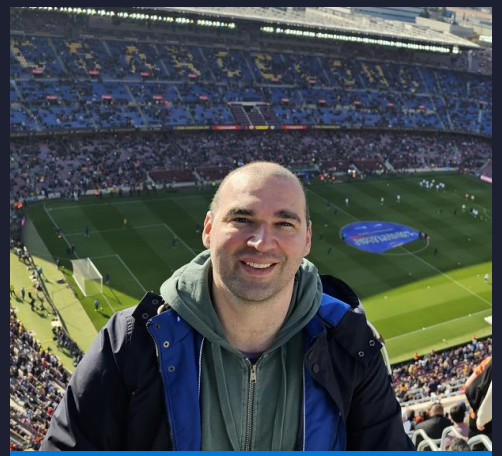
DP-600 Fabric Analytics Engineer

Nikola Ilic

Data Mozart, Microsoft Data Platform MVP



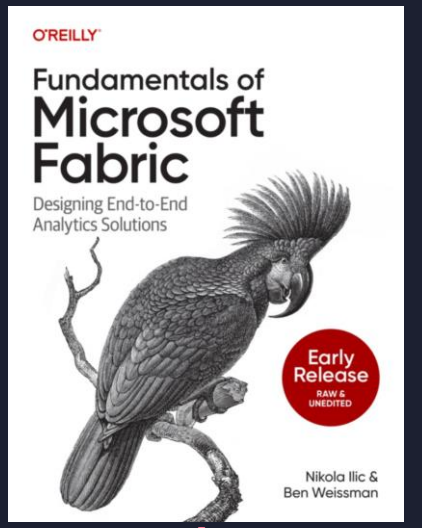
Nikola Ilic

Consultant & Trainer

data-mozart.com

learn.data-mozart.com

- *I'm making music from the data*!
- Power BI and SQL addict, blogger, speaker…
- Father of 2, Barca & Leo Messi fan…



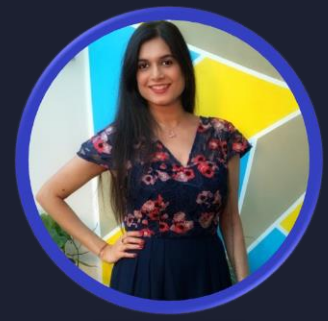
@DataMozart

Parvinder Chana

Johnny Winter

Govindarajan D

Thomas Martens



Pragati Jain

Eugene Meidinger

Miguel Felix

Mst. Rubayat Yasmin

Andy Cutler

Shannon Lindsay

@DataMozart

Session Goals

Learn

Have fun!

Introduction and Overview

What is DP-600?

1. One exam: Implementing Analytics Solutions Using MS Fabric
2. Data Analysis/Data Engineering
3. Understand features and services in MS Fabric

Candidate Profile

Expertise in designing, creating, and deploying enterprise-scale data analytics solutions

Responsibilities include

- Prepare and enrich data for analysis
- Secure and maintain analytics assets
- Implement and manage semantic models

These professionals help

- Collect enterprise-level requirements
- Data governance and configurations
- Monitor data usage
- Optimize performance

Candidates should have

- Advanced Power BI skills
- Power Query and DAX
- T-SQL & KQL
- Data modeling

Skills Measured Breakdown

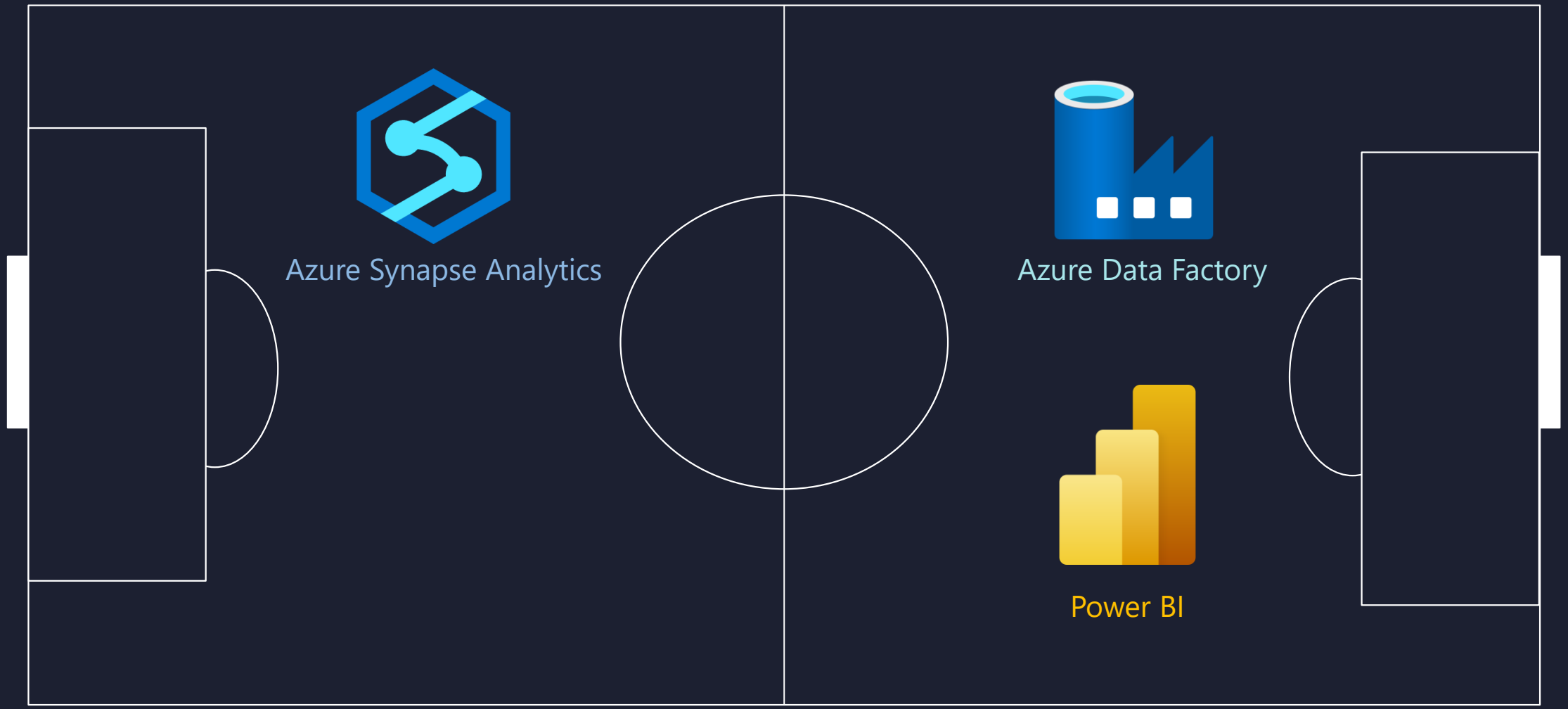
Maintain a data analytics solution (25-30%)

Prepare data (45-50%)

Implement and manage semantic models (25-30%)

What is a Microsoft Fabric?

What is a Microsoft Fabric?



Players in Microsoft Fabric

Microsoft Fabric Terminology

Tenant - organization's fabric instance

Capacity – compute power

Domain – logical grouping of workspaces

Workspace - collaborative container

Items - units of experience

Maintain a Data Analytics Solution 25-30%

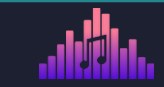
@DataMozart

Implement security and governance

- Workspace-level access controls
- Item-level access controls
- Row-level, column-level, object-level, and file-level access control
- Sensitivity labels
- Endorse items

Maintain the analytics development lifecycle

- Version Control
- Power BI Desktop project (.pbip)
- Deployment pipelines
- Impact analysis
- Semantic model deployment (XMLA endpoint)
- Reusable Power BI assets

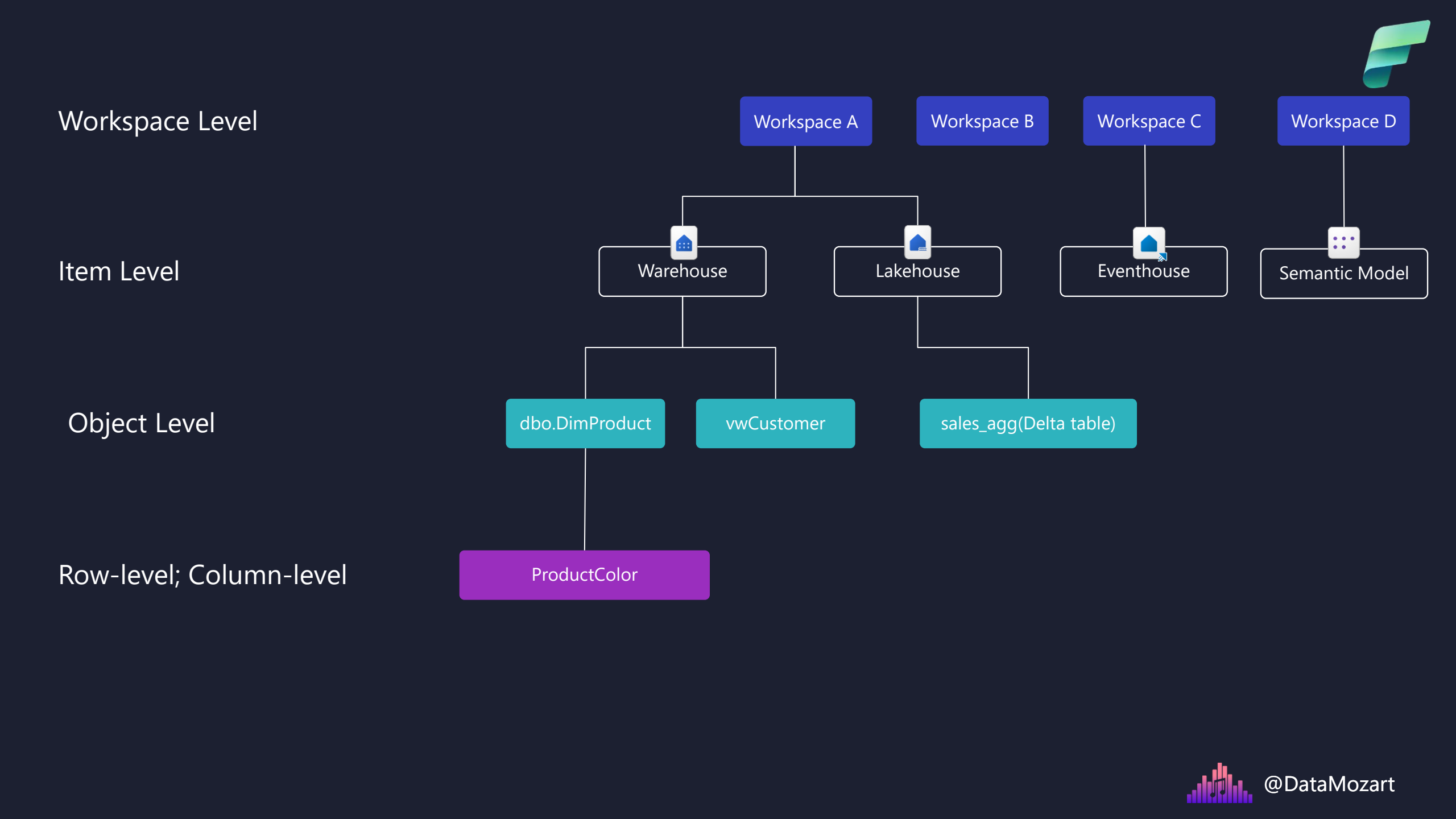


@DataMozart

Implement security and governance

- Workspace-level access controls
- Item-level access controls
- Row-level, column-level, object-level, and file-level access control
- Sensitivity labels
- Endorse items

@DataMozart


Workspace Level
Item Level
Object Level
Row-level; Column-level
Workspace A
Workspace B
Workspace C
Workspace D
Warehouse
Lakehouse
Eventhouse
Semantic Model
dbo.DimProduct
vwCustomer
sales_agg(Delta table)
ProductColor

@DataMozart

Access Control in Fabric

Workspace A

Lakehouse

Workspace-level

Access to all items in the workspace!

Item-level

Only access items shared with them!

Workspace Access Control

- **4 roles**
  - Admin
  - Member
  - Contributor
  - Viewer
- **Applies to ALL the items in the workspace!**
- **Individual user vs. Entra ID/M365 group**

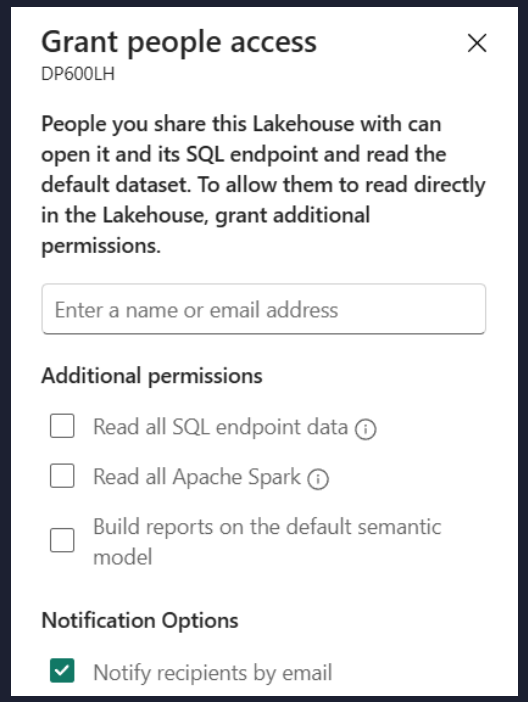
Capability	Admin	Member	Contributor	Viewer
Update and delete the workspace.	✓			
Add or remove people, including other admins.	✓			
Add members or others with lower permissions.	✓	✓		
Allow others to reshare items.[1]	✓	✓		
View and read content of data pipelines, notebooks, Spark job definitions, ML models and experiments, and Event streams.	✓	✓	✓	✓
View and read content of KQL databases, KQL query-sets, and real-time dashboards.	✓	✓	✓	✓
Connect to SQL analytics endpoint of Lakehouse or the Warehouse	✓	✓	✓	✓
Read Lakehouse and Data warehouse data and shortcuts[2] with T-SQL through TDS endpoint.	✓	✓	✓	✓
Read Lakehouse and Data warehouse data and shortcuts[2] through OneLake APIs and Spark.	✓	✓	✓	
Read Lakehouse data through Lakehouse explorer.	✓	✓	✓	
Write or delete data pipelines, notebooks, Spark job definitions, ML models and experiments, and Event streams.	✓	✓	✓	
Write or delete KQL query-sets, real-time dashboards, and schema and data of KQL databases, Lakehouses, data warehouses, and shortcuts.	✓	✓	✓	
Execute or cancel execution of notebooks, Spark job definitions, ML models and experiments.	✓	✓	✓	
Execute or cancel execution of data pipelines.	✓	✓	✓	
View execution output of data pipelines, notebooks, ML models and experiments.	✓	✓	✓	✓
Schedule data refreshes via the on-premises gateway.[3]	✓	✓	✓	
Modify gateway connection settings.[3]	✓	✓	✓	

[1] Contributors and Viewers can also share items in a workspace, if they have Reshare permissions.

[2] Additional permissions are needed to read data from shortcut destination. Learn more about shortcut security model.

[3] Keep in mind that you also need permissions on the gateway. Those permissions are managed elsewhere, independent of workspace roles and permissions.

Item-level Access Control



Permission granted while sharing	Effect
Read	Recipient can discover the item in the data hub and open it. Connect to the Warehouse or SQL analytics endpoint of the Lakehouse.
Edit	Recipient can edit the item or its content.
Share	Recipient can share the item and grant permissions up to the permissions that they have. For example, if the original recipient has *Share*, *Edit*, and *Read* permissions, they can at most grant *Share*, *Edit*, and *Read* permissions to the next recipient.
Read All with SQL analytics endpoint	Read data from the SQL analytics endpoint of the Lakehouse or Warehouse data through TDS endpoints.
Read all with Apache Spark	Read Lakehouse or Data warehouse data through OneLake APIs and Spark. Read Lakehouse data through Lakehouse explorer.
Build	Build new content on the semantic model.
Execute	Execute or cancel execution of the item.

[Item permission model]

- **Nothing checked –** access LH from OneLake hub, but none of the tables (suitable for OLS/CLS)
- **Read all SQL endpoint data**
- **Read all Apache Spark**
- **Build reports on the default semantic model**

@DataMozart

Row-level Security

```sql
-- Creating schema for Security
CREATE SCHEMA Security;
GO
```

```sql
-- Creating a function for the SalesRep evaluation
CREATE FUNCTION Security.tvf_securitypredicate(@UserName AS varchar(50))
    RETURNS TABLE
WITH SCHEMABINDING
AS
    RETURN SELECT 1 AS tvf_securitypredicate_result
WHERE @UserName = USER_NAME();
GO
```

```sql
-- Using the function to create a Security Policy
CREATE SECURITY POLICY YourSecurityPolicy
ADD FILTER PREDICATE Security.tvf_securitypredicate(UserName_column)
ON sampleschema.sampletable
WITH (STATE = ON);
GO
```

Admin, Member, or Contributor on the workspace OR elevated permissions on the Warehouse/SQL endpoint

Object-level and Column-level Security

Object-level security

```
GRANT SELECT ON Sales.FactResellerSales TO [SalesReps];
```

Column-level security

```
GRANT SELECT ON dbo.Customer (CustomerName, Email, PhoneNumber) TO [SalesReps];
```

Admin, Member, or Contributor on the workspace OR elevated permissions on the Warehouse/SQL endpoint

Folder-level Access Control

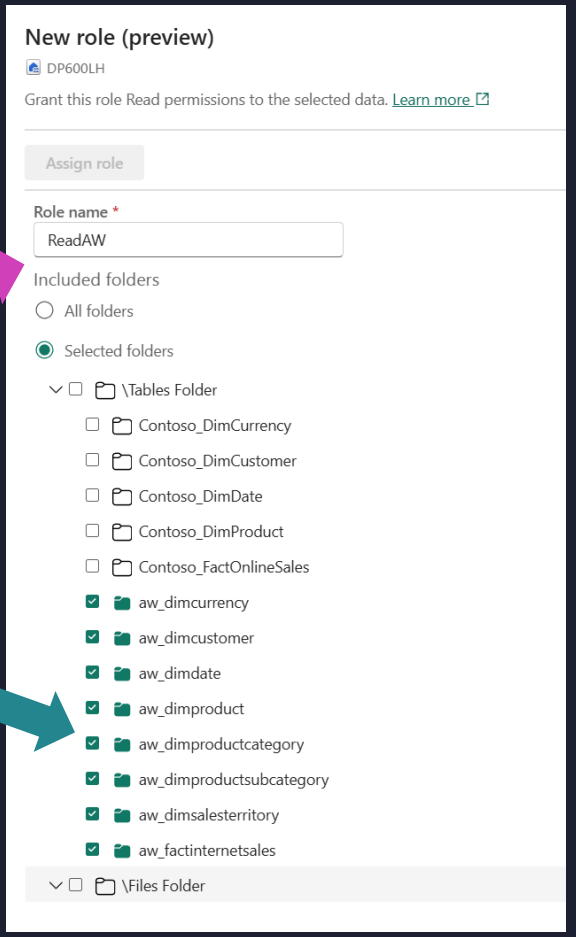
Read access to specific folders

- Data access role security ONLY applies to users accessing OneLake directly!
- Currently supported only for the Lakehouse item

1. Create a role

2. Choose tables/files from Lakehouse

3. Assign role to user/group



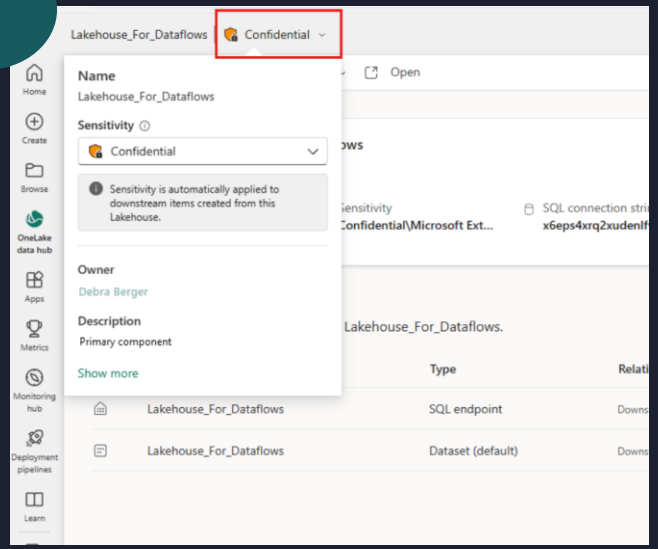
Sensitivity Labels

First created in Purview

Microsoft Purview Information Protection

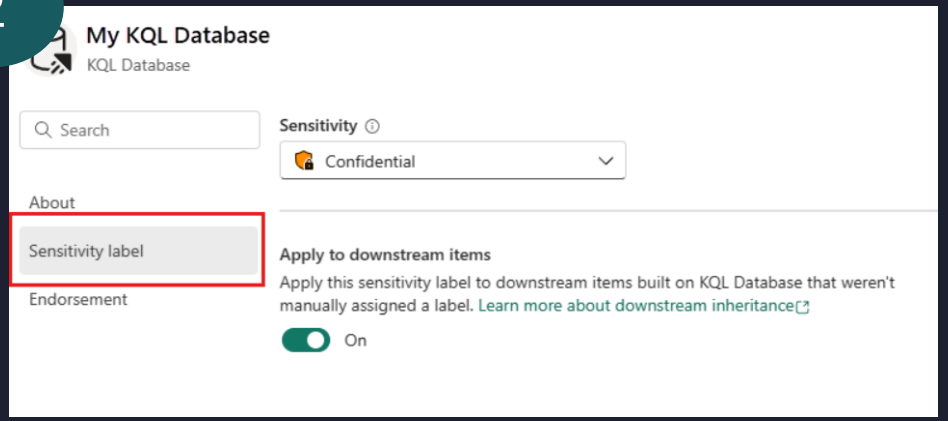
- Meet governance and compliance requirements
- Only authorized people can access the data
- 2 ways to apply

1



From the flyout menu

2

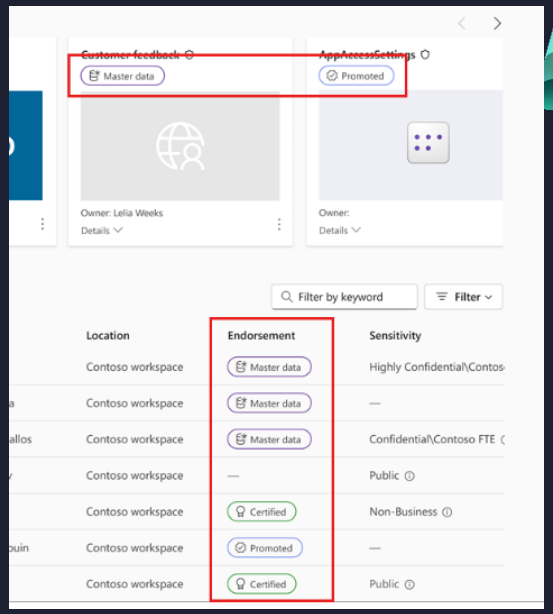


In item settings

@DataMozart

Endorsement

- Makes it easier for users to find high-quality, trustworthy data
- Labeled with a badge in the UI
- 3 endorsement badges



1

Promoted

- ✓ Item ready for sharing and reuse
- ✓ All items except dashboards
- ✓ Any user with write permission on item can promote it

2

Certified

- ✓ Authorized reviewer
- ✓ Item meets org's quality standards
- ✓ All items except dashboards
- ✓ Any user can ***request***, only users specified by Fabric Admin can certify items

3

Master data

- ✓ Core source of org data
- ✓ Single source of truth
- ✓ Items that contain data (lakehouses, semantic models)
- ✓ Only users specified by Fabric Admin can label Master data

Implement security and governance

- Workspace-level access controls
- Item-level access controls
- Row-level, column-level, object-level, and file-level access control
- Sensitivity labels
- Endorse items

Maintain the analytics development lifecycle

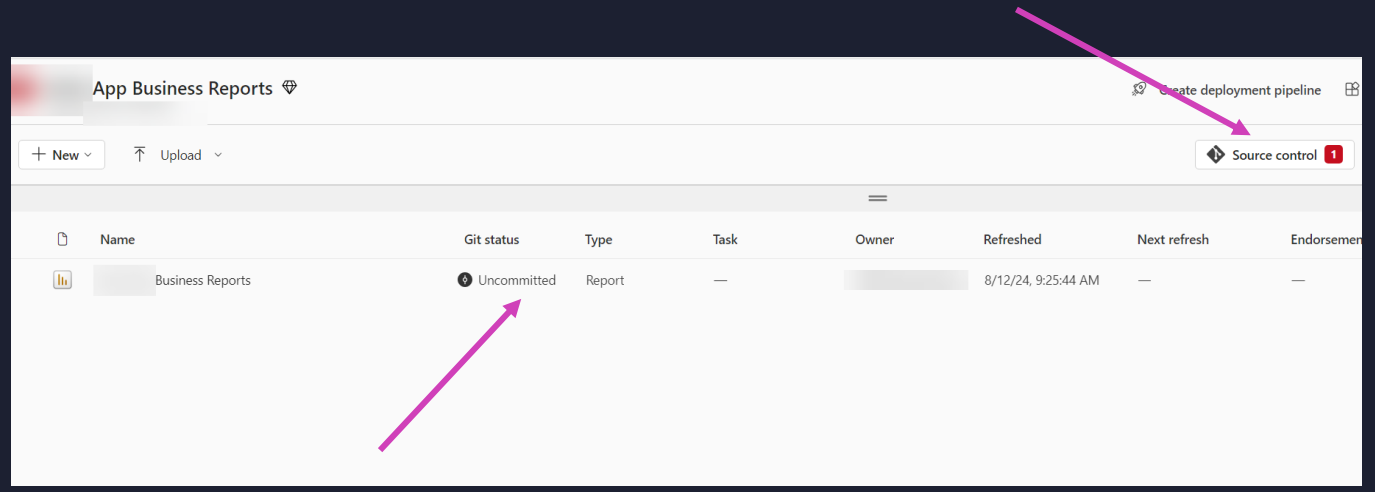
- Version Control
- Power BI Desktop project (.pbip)
- Deployment pipelines
- Impact analysis
- Semantic model deployment (XMLA endpoint)
- Reusable Power BI assets

Maintain the analytics development lifecycle

- Version Control
- Power BI Desktop project (.pbip)
- Deployment pipelines
- Impact analysis
- Semantic model deployment (XMLA endpoint)
- Reusable Power BI assets

Version Control for a Workspace

- Git concepts – branches, commits, pull requests, merging
- Supported providers: Git in Azure Repos (the same tenant as the Fabric tenant), GitHub, GitHub Enterprise
- What can be done?
  - Sync content from Git – Overwrite!
  - Commit changes to Git
  - Branch out to a new workspace



- ✓ Almost all Fabric items are supported
- ✓ Unsupported items will be ignored
- ✓ Only workspace Admin can manage connections

Power BI Desktop Project (.pbip)

.pbip

- ✓ Report and semantic model definitions saved separately as plain text files
- ✓ Folder structure
  - ✓ .**Report** folder
  - ✓ .**SemanticModel** folder

- Text editor support
- Scripting and editing definitions
- Source control
- CI/CD

Deployment Pipelines

Development

- ✓ Design, review, play around
- ✓ Start small

All vs Selective deployment

Test

- ✓ Test and verify that content meets the criteria
- ✓ Larger, realistic data volumes
- ✓ Test the App

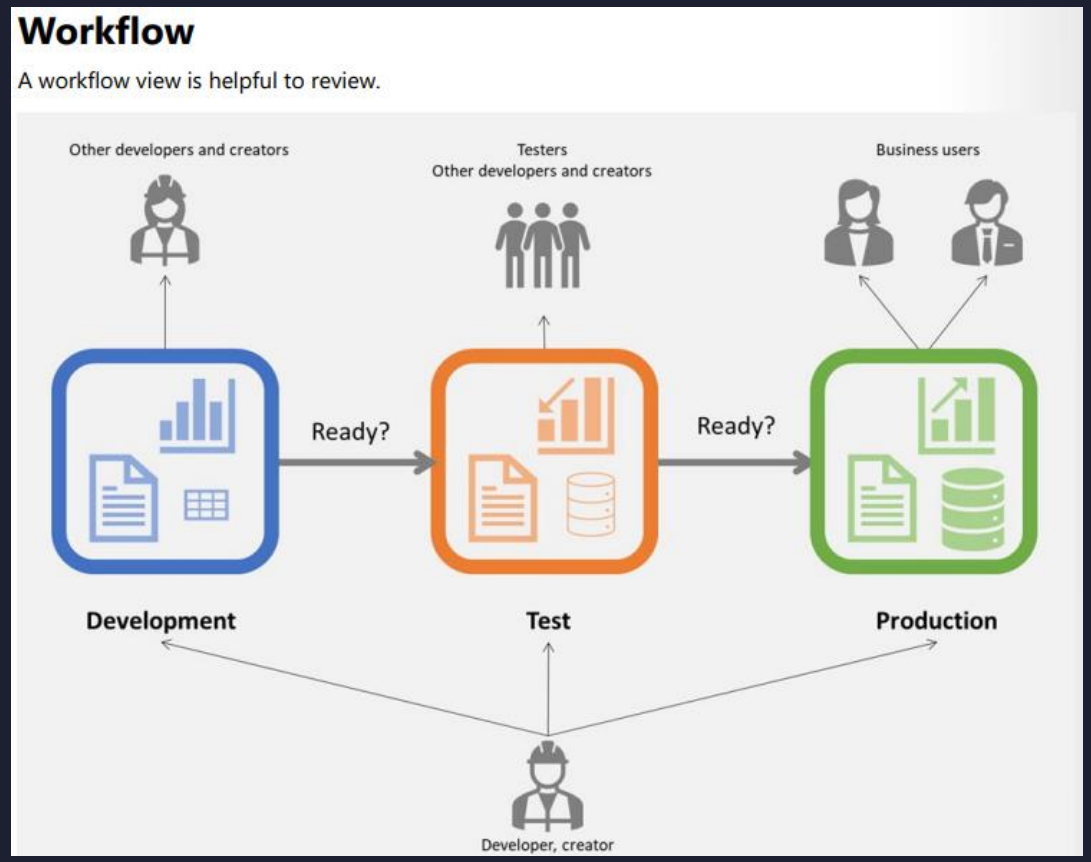
All vs Selective deployment

Production

- ✓ Highest possible quality

@DataMozart

Deployment Pipelines Workflow



- ✓ Content and workspace can be different in each stage
- ✓ Size of the data source
- ✓ Separate workspaces for each stage
- ✓ Automate with REST APIs and DevOps
- ✓ Configure rules to allow changes

The content will be overridden!

@DataMozart

Deployment Pipelines Supported Items

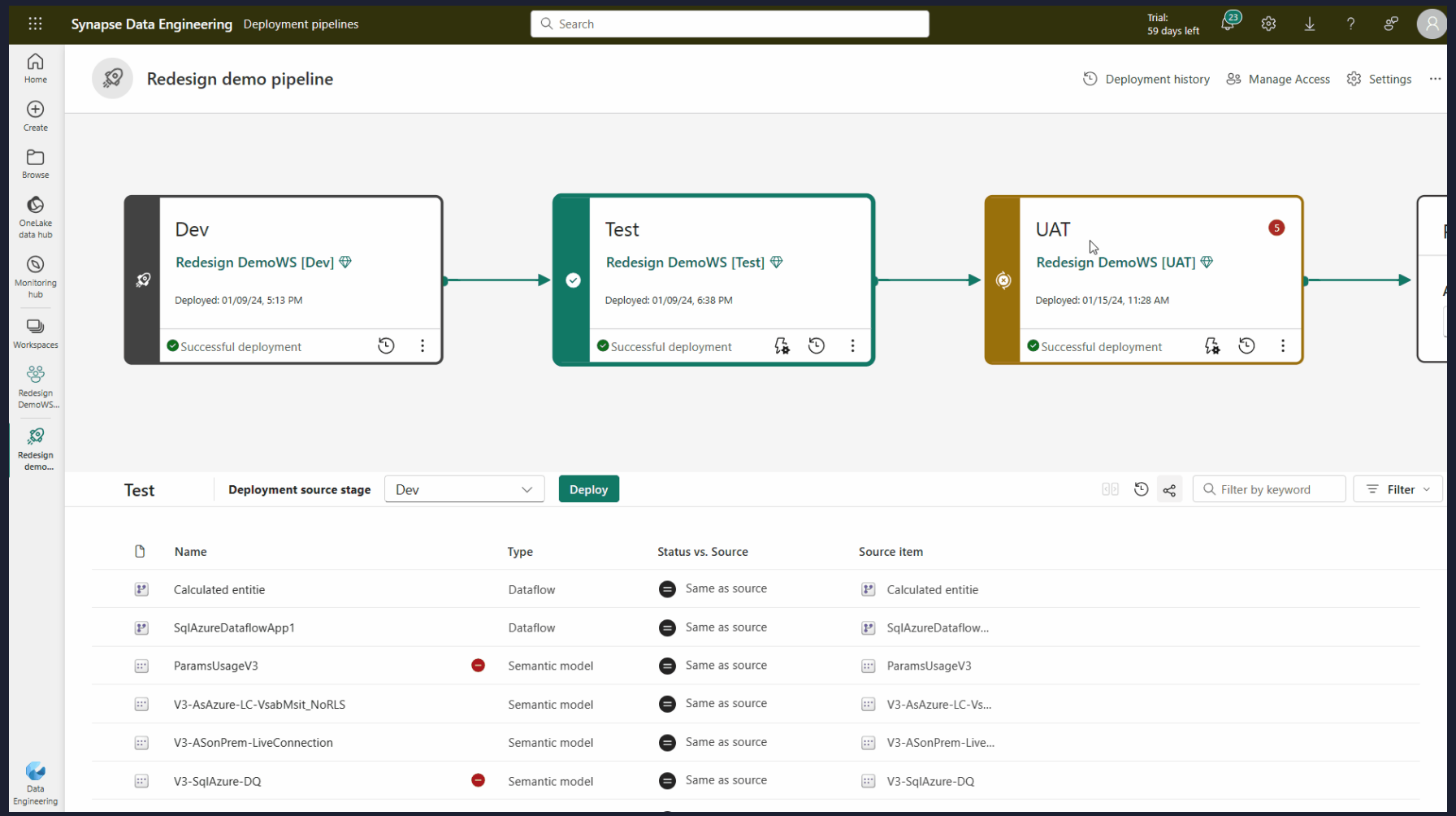
Supported items

When you deploy content from one pipeline stage to another, the copied content can contain the following items:

- Dashboards
- Data pipelines *(preview)*
- Dataflows gen2 *(preview)*
- Datamarts *(preview)*
- EventHouse *(preview)*
- EventStream *(preview)*
- Lakehouse *(preview)*
- Eventhouse and KQL database *(preview)*
- Notebooks
- Organizational apps *(preview)*
- Paginated reports
- Power BI Dataflows
- Reflex *(preview)*
- Reports (based on supported semantic models)
- Spark environment *(preview)*
- Semantic models (that originate from .pbix files and aren't PUSH datasets)
- SQL database *(preview)*
- Warehouses *(preview)*

Always check the list of supported items

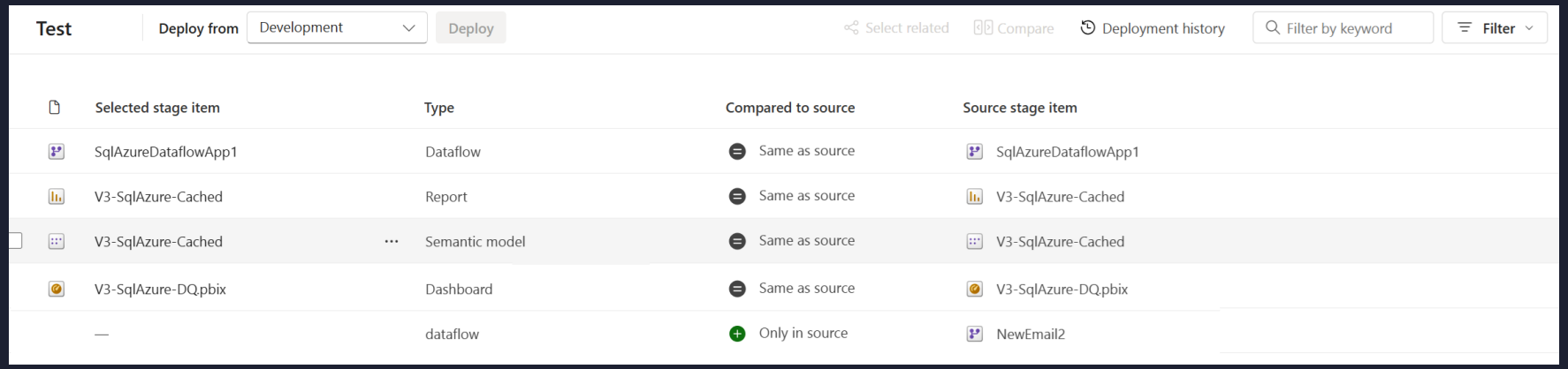
Deployment Pipelines Workflow



@DataMozart

Deployment Pipelines – Item Pairing

- Item in one stage of the deployment pipeline is associated with the same item in the previous/next stage



- Paired items remain paired even if you change their names => Paired items can have different names

- Items added after the workspace is assigned to a pipeline aren't automatically paired => You can have identical items in adjacent workspaces that aren't paired

Deployment Pipelines – Item Pairing

Scenario	Stage A (for example, Dev)	Stage B (for example, Test)	Comment
1	Name: *PBI Report* Type: *Report*	Name: *PBI Report* Type: *Report*	✅ Pairing occurs
2	Name: *PBI Report* Type: *Report*	Name: *PBI Report* Type: *Report*	❌ Pairing doesn't occur (duplicates). ❌ Deployment fails.
		Name: *PBI Report* Type: *Report*	❌ Pairing doesn't occur (duplicates). ❌ Deployment fails.
3	Name: *PBI Report* Type: *Report* *Folder A*	Name: *PBI Report* Type: *Report* *Folder B*	✅ Deployment succeeds but ❌ this report isn't paired with dev
		Name: *PBI Report* Type: *Report* *Folder A*	✅ Pairing occurs using folder as a tie breaker for duplicates
		Name: *PBI Report* Type: *Report* *No folder*	✅ Deployment succeeds but ❌ this report isn't paired with dev

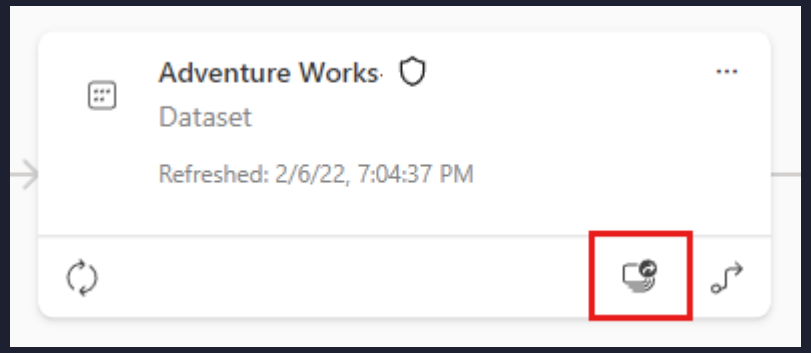
Deployment Pipelines – Deployment Rules

- Different stages can have different configurations (different databases, query parameters…)

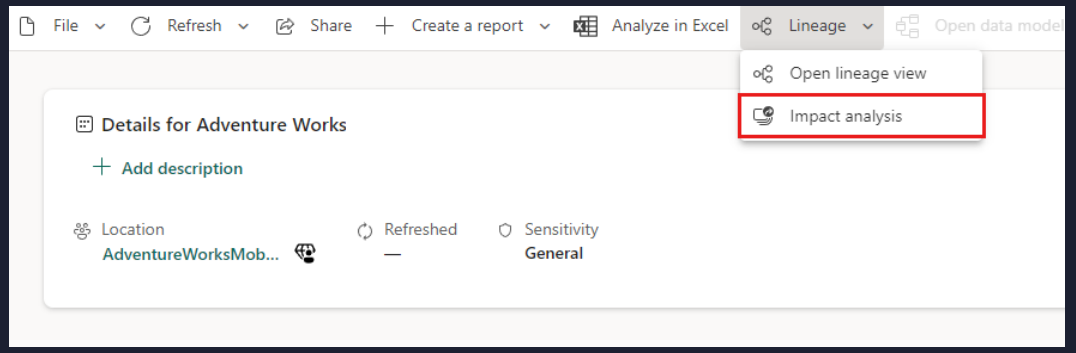
Item	Data source rule	Parameter rule	Default lakehouse rule	Details
Dataflow	✅	✅	❌	Use to determine the values of the data sources or parameters for a specific dataflow.
Semantic model	✅	✅	❌	Use to determine the values of the data sources or parameters for a specific semantic model.
Datamart	✅	✅	❌	Use to determine the values of the data sources or parameters for a specific datamart.
Paginated report	✅	❌	❌	Defined for the data sources of each paginated report. Use to determine the data sources of the paginated report.
Notebook	❌	❌	✅	Use to determine the default lakehouse for a specific notebook.

Perform Impact Analysis

- Potential impact of changes to Fabric item
- Lineage view in the workspace



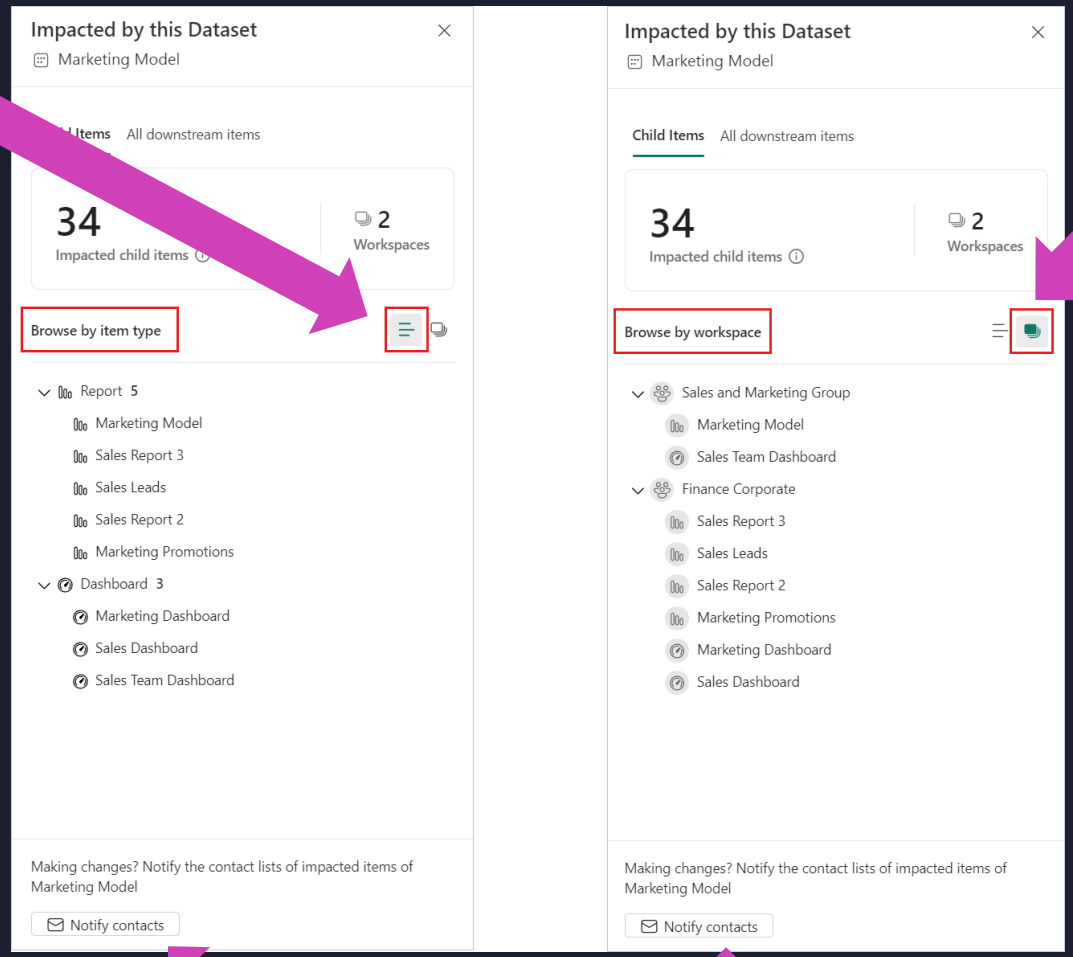
- Item details page



Perform Impact Analysis

By item type

By workspace



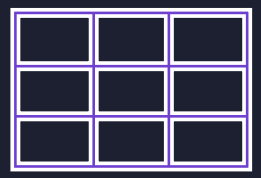
Notify contacts

@DataMozart

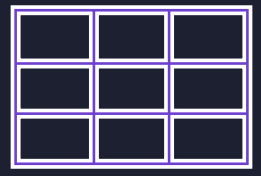
Semantic Model Deployment via XMLA

- Allows “communication” with a semantic model
- Read-only by default
- Read-write enables more options for model management, configuration, advanced modeling…

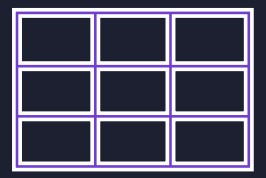
Creating Reusable Power BI Assets



Product

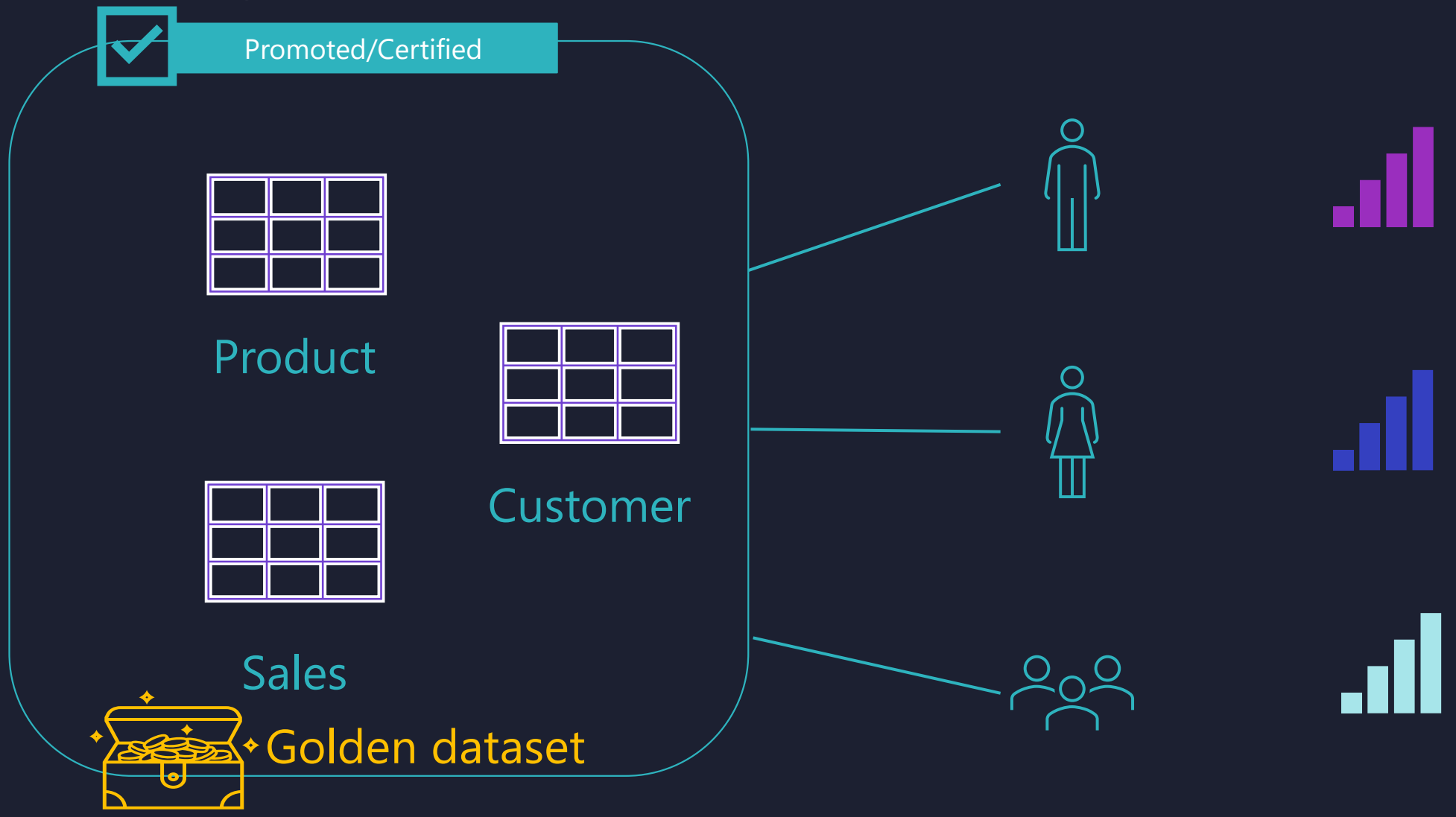


Sales



Customer

Creating Reusable Power BI Assets



Creating Reusable Power BI Assets

PBI Template

- Initial point for new report layout and/or model
- Export the existing pbix file as a template (pbit)

PBI DS

- Data source file
- Includes connection details
- Quickly move connections from one solution to another
- Export from the existing pbix

Shared models

- Multiple users leverage models that were already created
- Various permissions: Can modify, share onwards…

@DataMozart

Implement security and governance

- Workspace-level access controls
- Item-level access controls
- Row-level, column-level, object-level, and file-level access control
- Sensitivity labels
- Endorse items

Maintain the analytics development lifecycle

- Version control for workspace
- Power BI Desktop project (.pbip)
- Deployment pipelines
- Impact analysis
- Semantic model deployment via XMLA endpoint
- Reusable Power BI assets

Prepare Data
45-50%

Get data

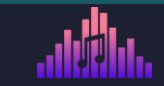
- Create a data connection
- Discover data by using OneLake data hub and real-time hub
- Ingest or access data as needed (shortcuts, dataflows, pipelines, notebooks)
- Choose between a lakehouse, warehouse, or eventhouse
- Implement OneLake integration for eventhouse and semantic models

Transform data

- Create views, functions, and stored procedures
- Enrich data by adding new columns or tables
- Implement a star schema for a lakehouse or warehouse
- Denormalize data
- Aggregate data
- Merge or join data
- Identify and resolve duplicate data, missing data, or null values
- Convert column data types
- Filter data

Query and analyze data

- Select, filter, and aggregate data by using the Visual Query Editor
- Select, filter, and aggregate data by using SQL
- Select, filter, and aggregate data by using KQL



@DataMozart

Get data

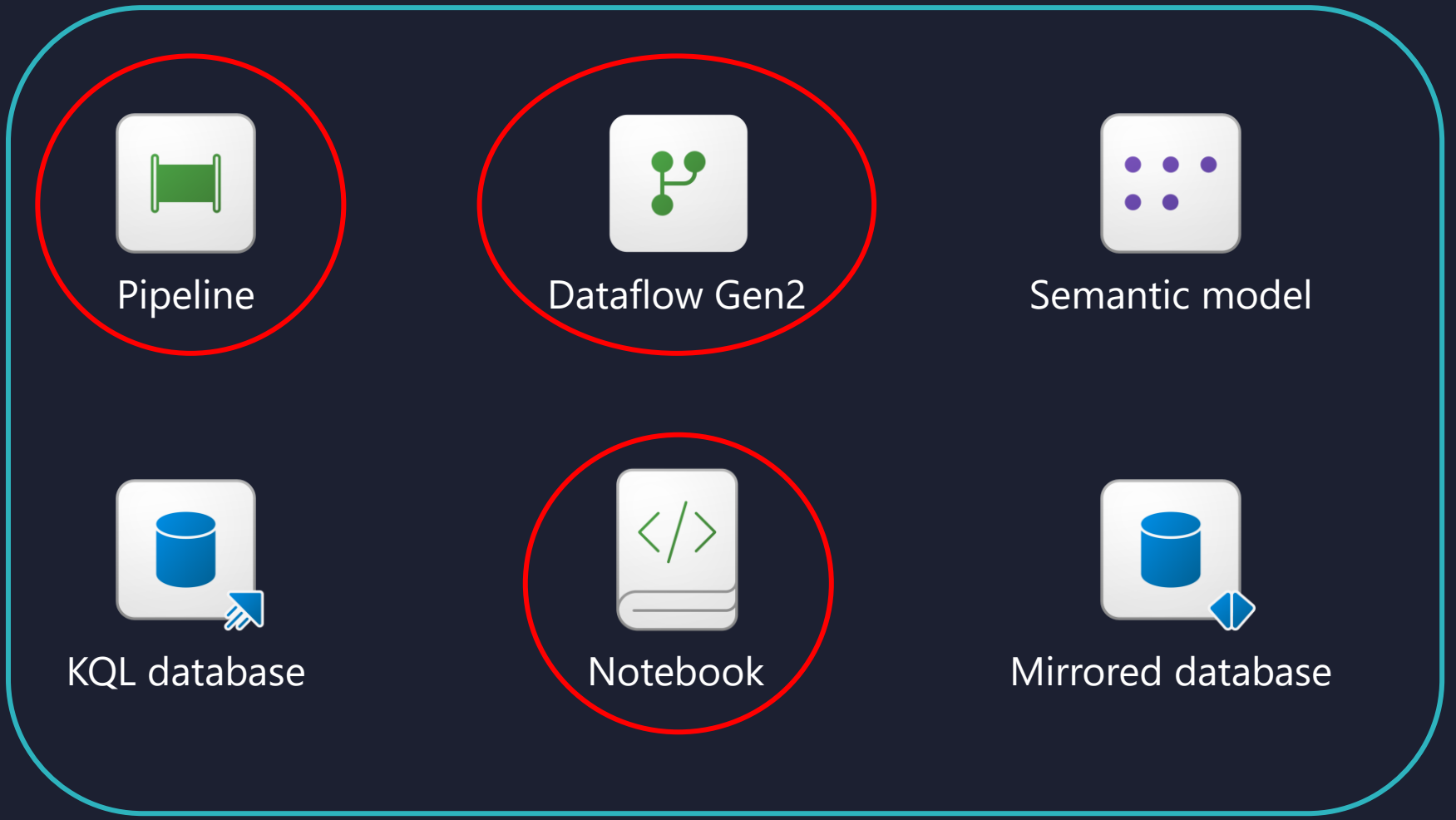
- Create a data connection
- Discover data by using OneLake data hub and real-time hub
- Ingest or access data as needed (shortcuts, dataflows, pipelines, notebooks)
- Choose between a lakehouse, warehouse, or eventhouse
- Implement OneLake integration for eventhouse and semantic models

Create a Data Connection

Cloud sources

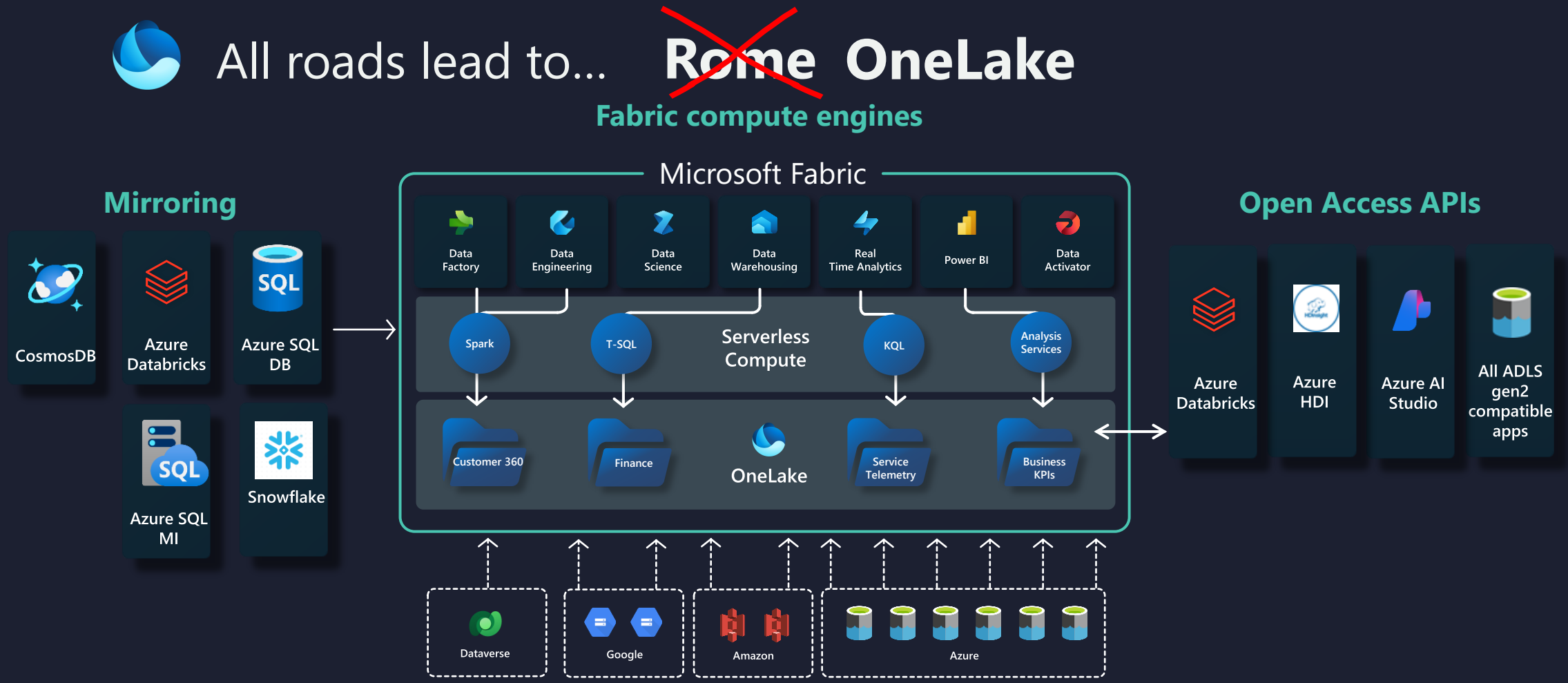
On-prem sources

Create a Data Connection



Ingest or Access Data

All roads lead to... ~~Rome~~ OneLake

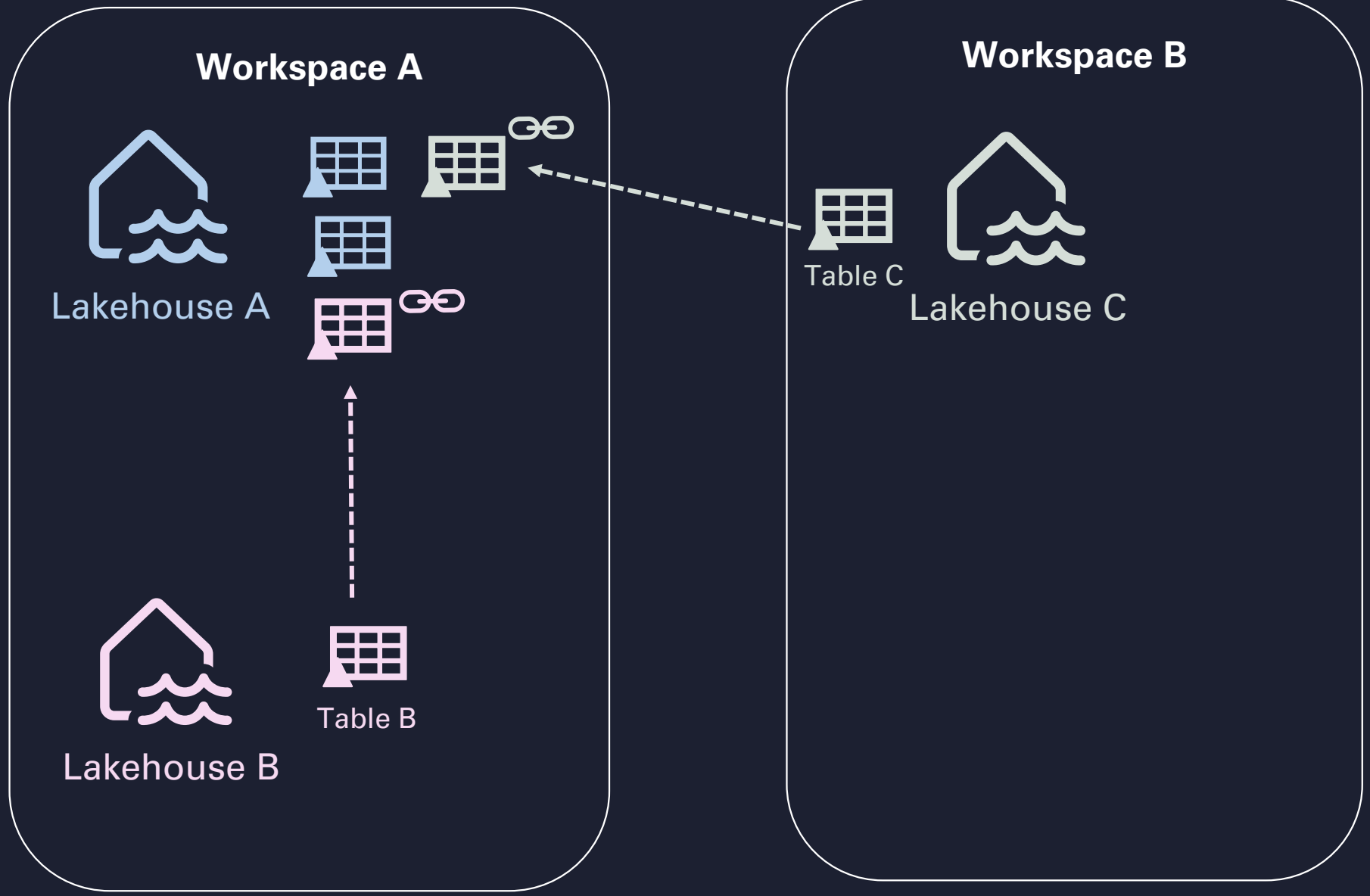


@DataMozart

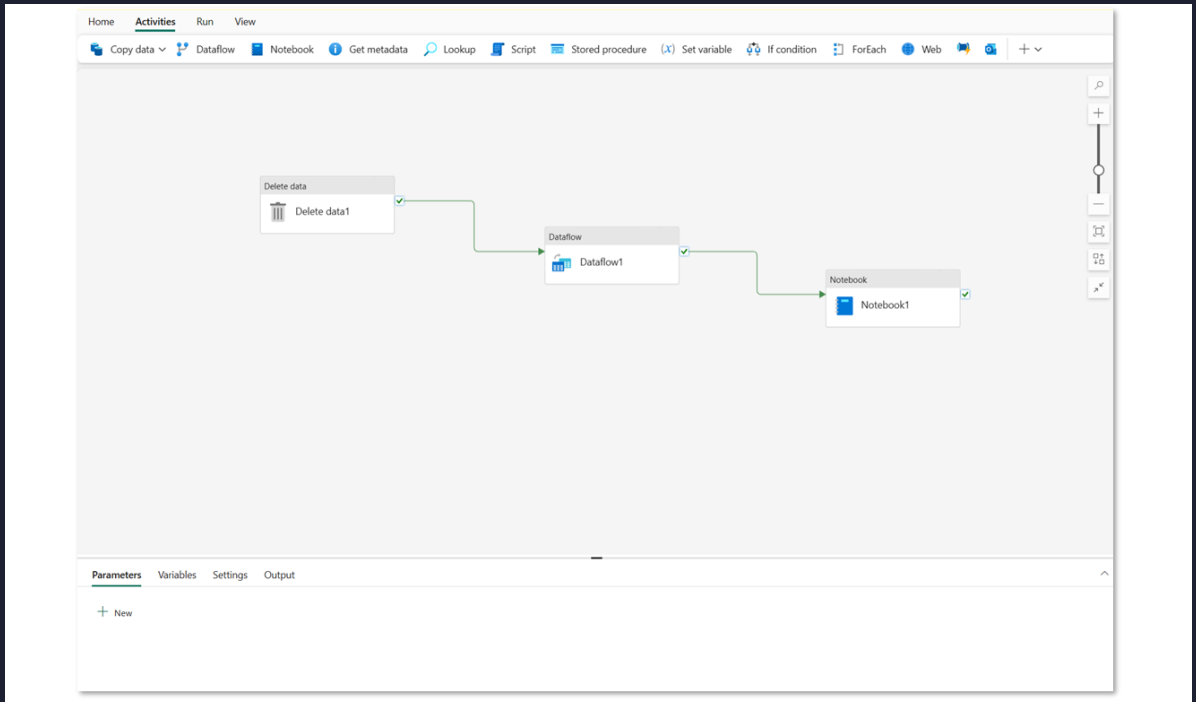
Shortcuts

Shortcuts in Windows Explorer?

Shortcuts



Pipelines

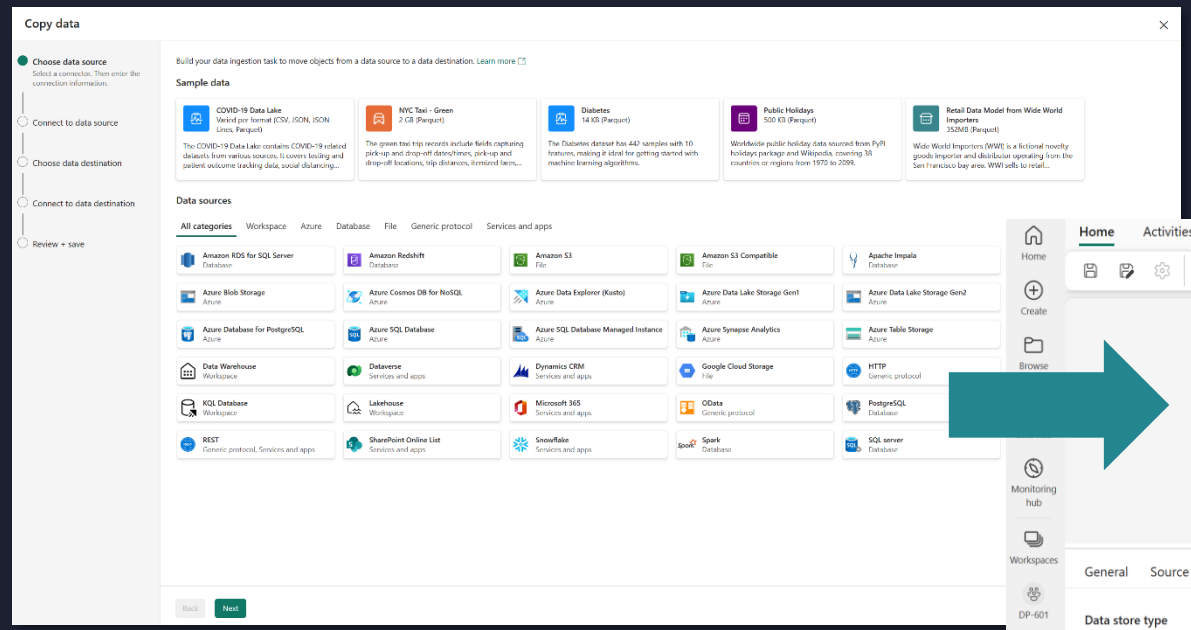


Pipeline concepts:

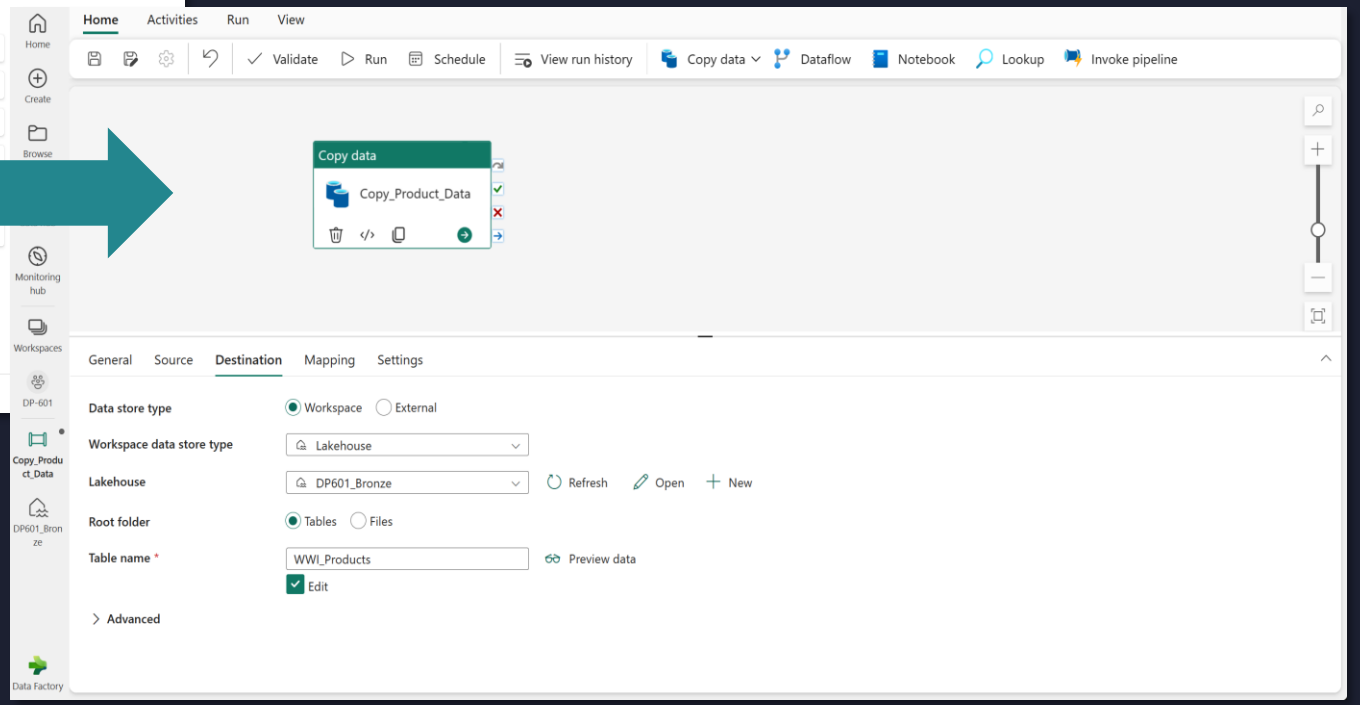
- Activities
  - Data transformation
  - Control flow
- Parameters
- Schedule runs

@DataMozart

Common Activities – Copy Data



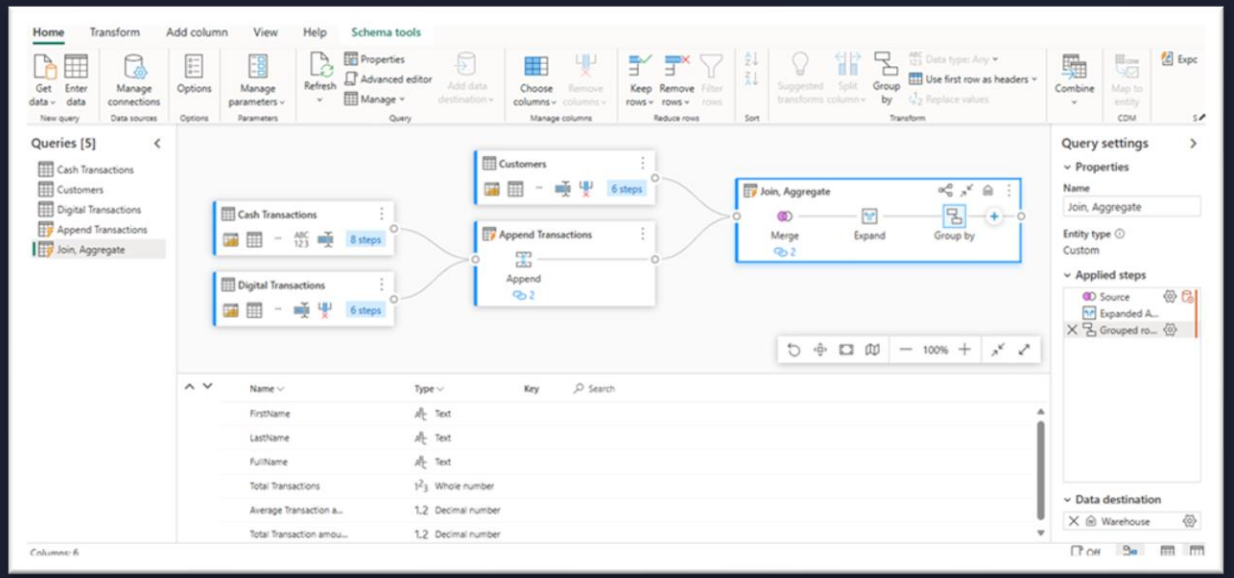
1. Use the copy data tool



2. Edit the settings below the pipeline canvas

Dataflows Gen2

- Low-code graphical environment for defining ETL solutions
- Extract data from multiple sources, transform it, and load it into a destination
- Run dataflows independently or as an activity in a Pipeline



Dataflow Gen2 vs Pipeline

- ✓ Data transformations
- ✓ Data profiling
- ✓ Familiar Power Query experience
- ✓ Many connectors (150+)

- ✓ Orchestration
- ✓ Copy data = simple transformation
- ✓ DF G2 can be part of the orchestration process

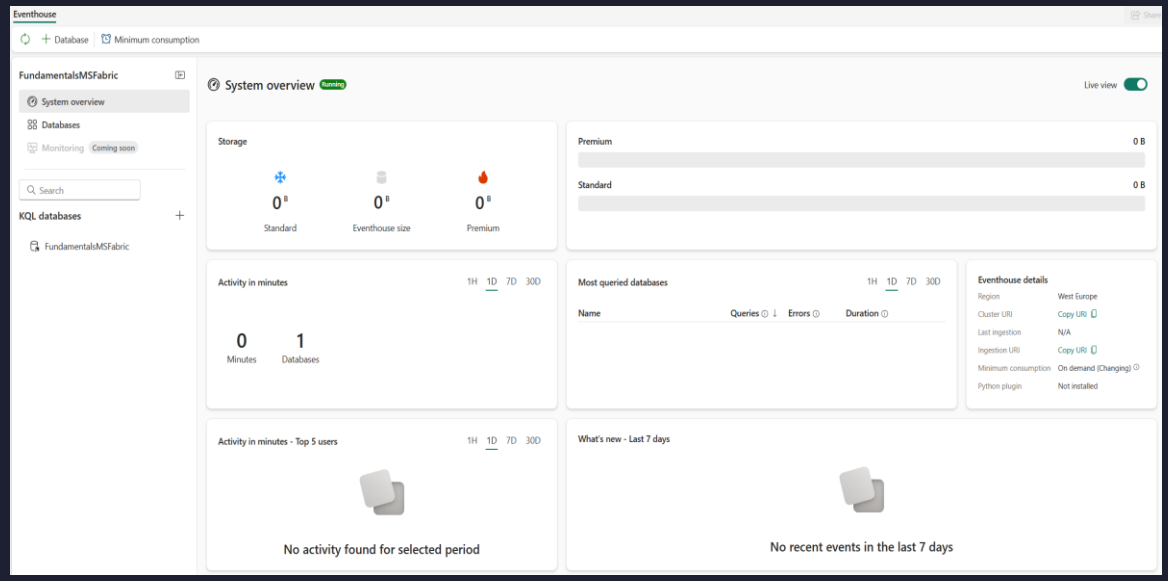
Lakehouse, Warehouse or Eventhouse

When to choose what?!

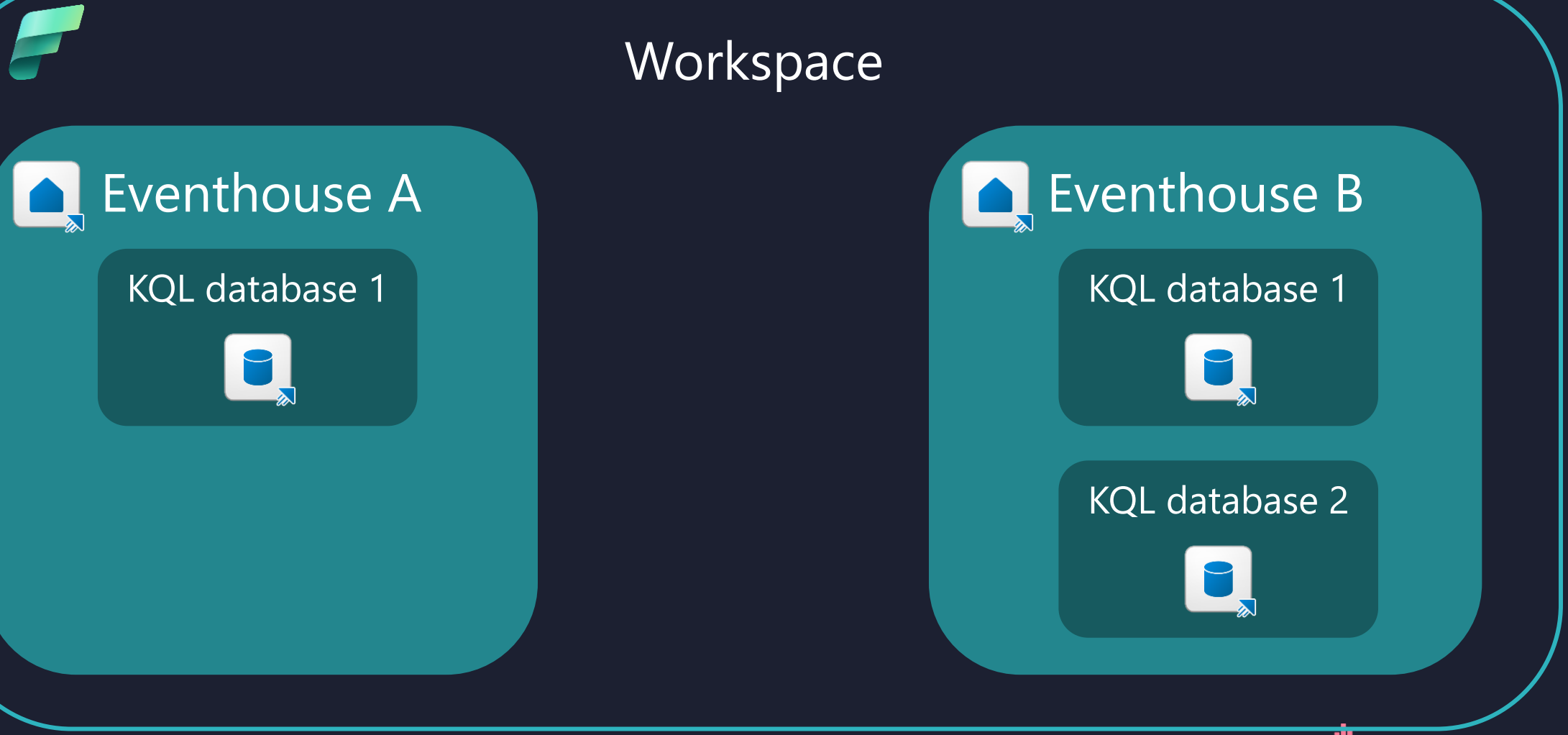
Handling streaming data!

Eventhouse

- Container for KQL databases
- Unified monitoring and management across all databases
- Data automatically indexed and partitioned



KQL Database



Fabric “Houses”

+

Data Lake**house**

- Scalable, distributed file storage
- Flexible schema-on-read semantics
- Unstructured, semi-structured, and structured data

Data Ware**house**

- Relational schema modeling
- SQL-based querying
- Structured data

Lakehouse

- ✓ Files -> Unmanaged area -> store any format
- ✓ Tables -> Managed by Spark -> preferably Delta

- ✓ Read-only for T-SQL engine
- ✓ Only delta tables are exposed
- ✓ Needed for DirectQuery fallback

Working With Fabric Lakehouse

Tools and techniques to explore and transform data

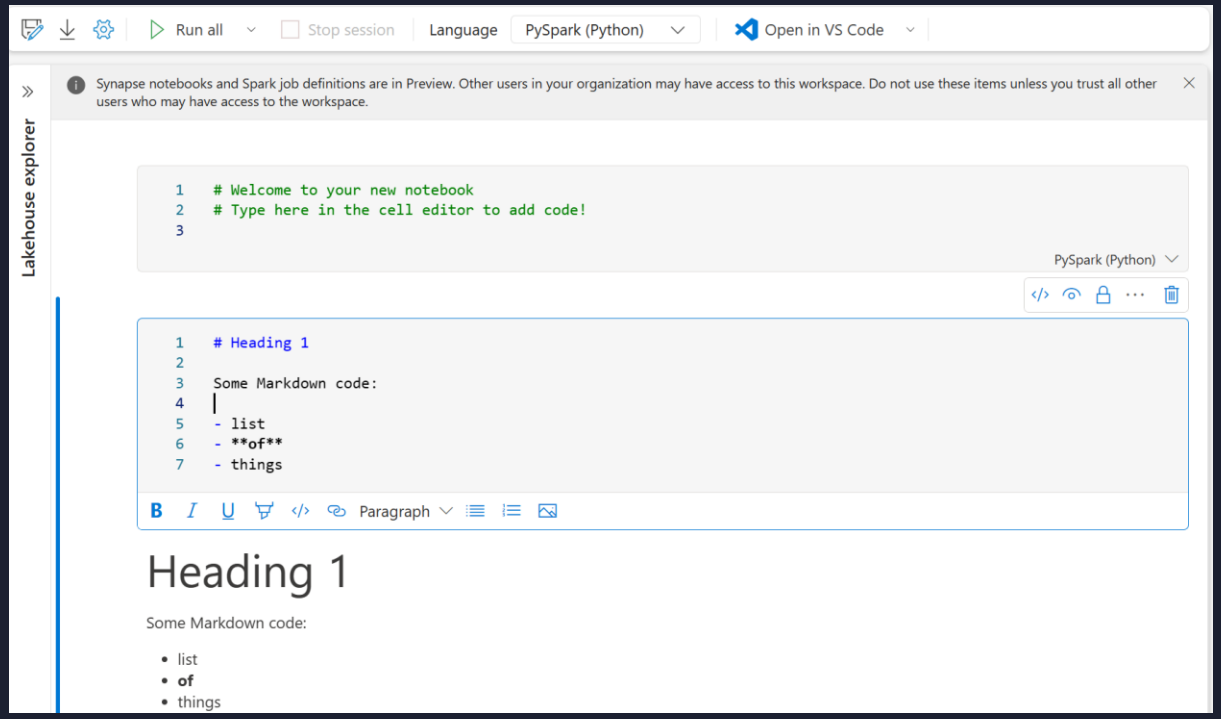
- Apache Spark
  - Notebooks
  - Spark Job Definitions
- SQL analytics endpoint
- Dataflows (Gen2)
- Data Pipelines

Visualize with Power BI

@DataMozart

Fabric Notebooks

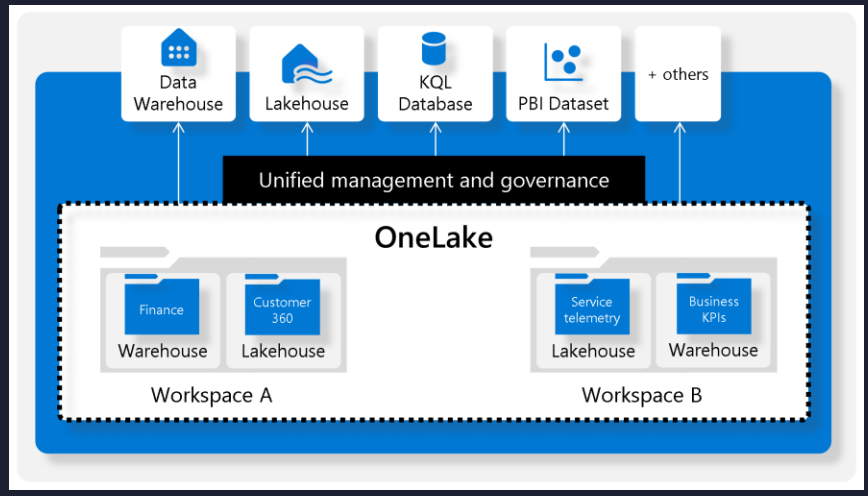
- Code (PySpark, Scala, R, Spark SQL)
- Markdown (comments)
- Run or freeze individual or multiple cells
- Ingest and transform
- Support automation



@DataMozart

Warehouse

- Centered on the single data lake (OneLake)
- Powered by Synapse Analytics
- (Almost) Fully supports T-SQL
- Parquet file format



Working With Fabric Warehouse

Tools and techniques to explore and transform data

- SQL endpoint
- Dataflows (Gen2)
- Data Pipelines

Visualize with Power BI

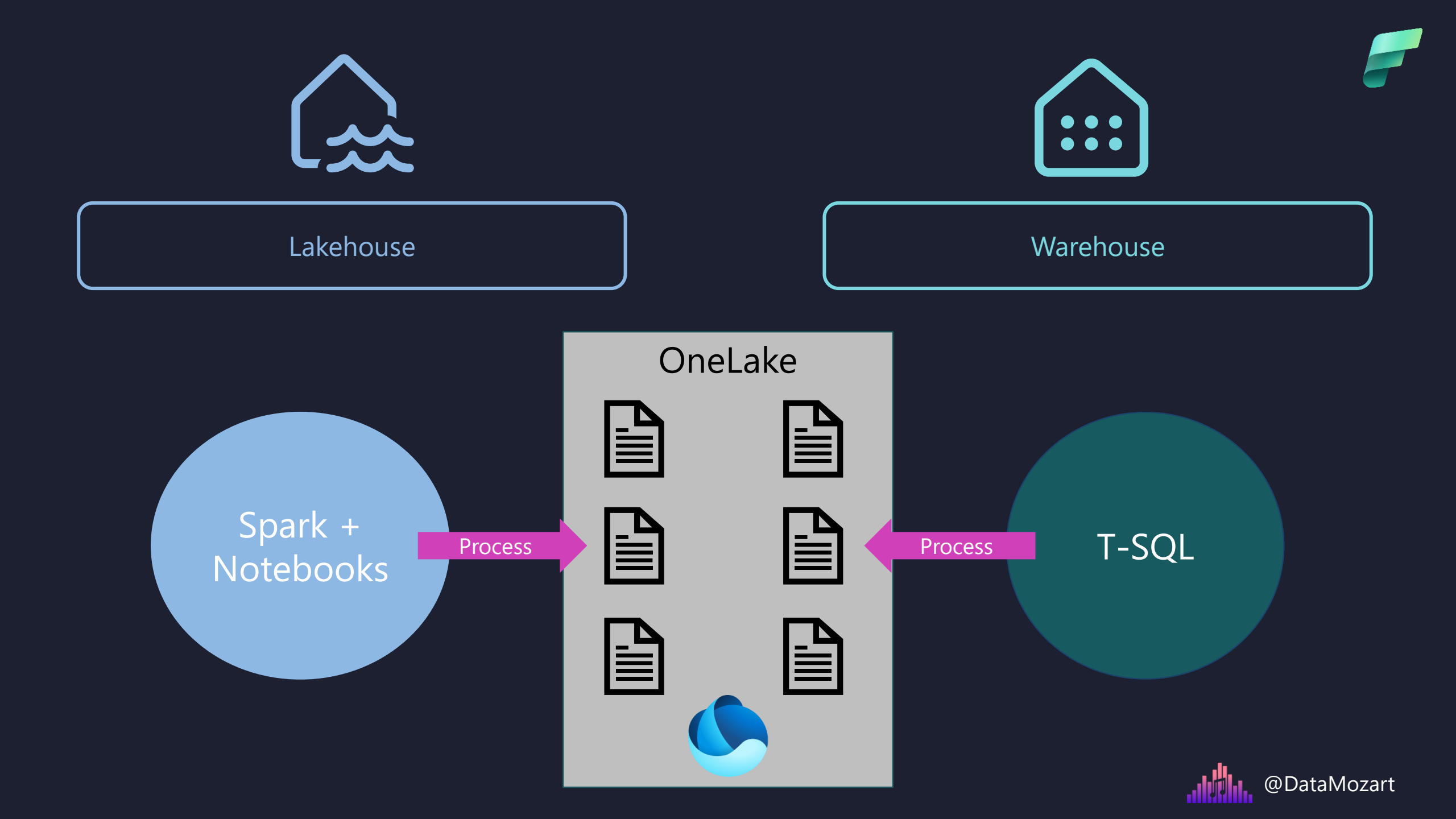
Two Types of SQL-"Houses" in Fabric

SQL Endpoint of the Lakehouse

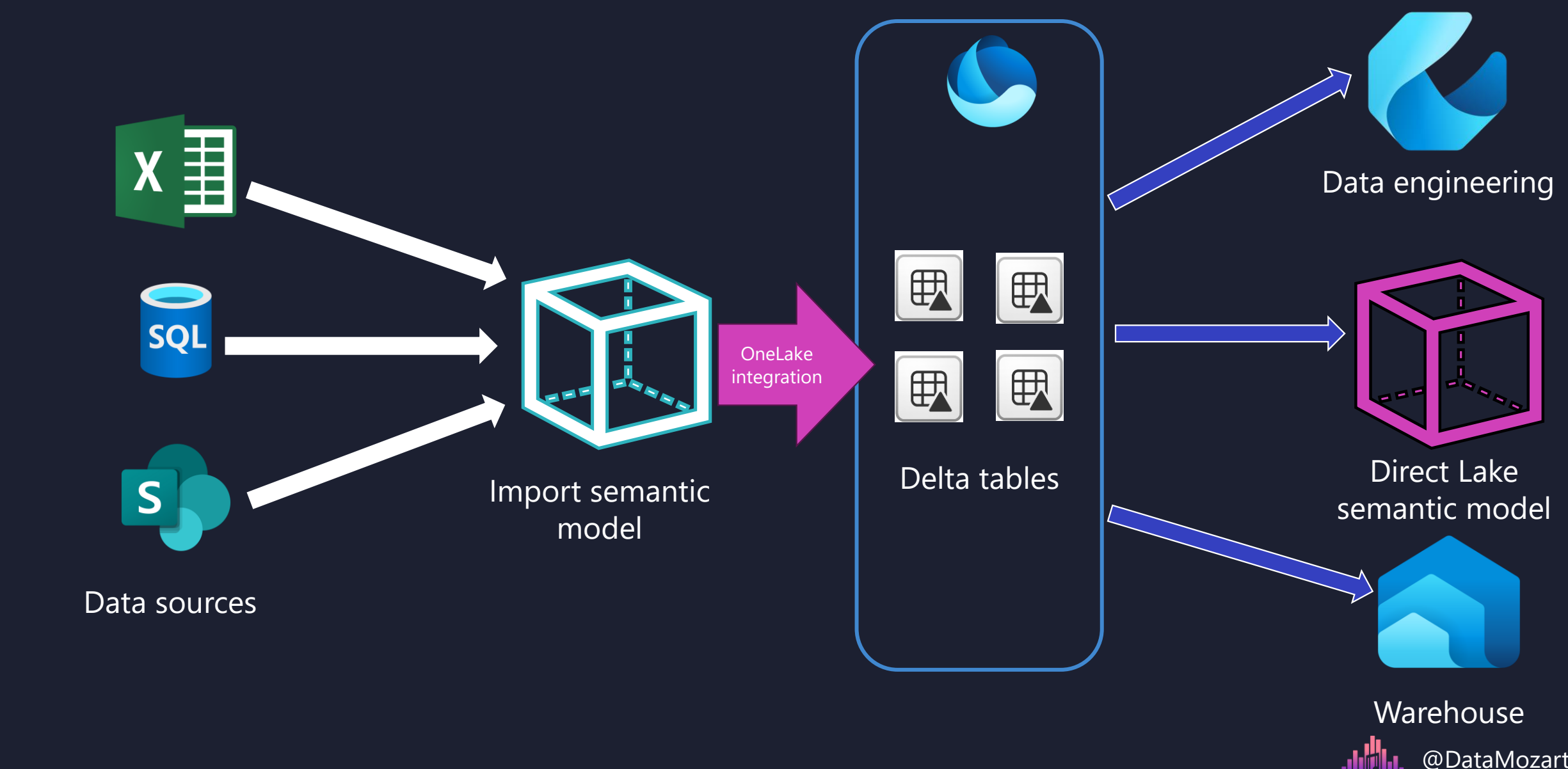
- ✓ Automatically generated
- ✓ Supports ONLY read operations
- ✓ Views, inline TVFs, procs…
- ✓ Manage permissions

Synapse Data Warehouse

- ✓ Full transactional support
- ✓ DDL/DML operations
- ✓ Traditional data warehousing workloads

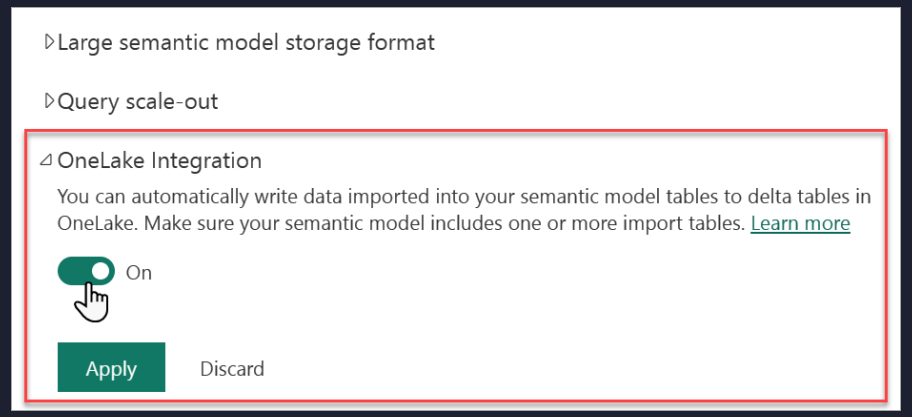

Lakehouse
Warehouse
OneLake
Spark + Notebooks
Process
Process
T-SQL
@DataMozart

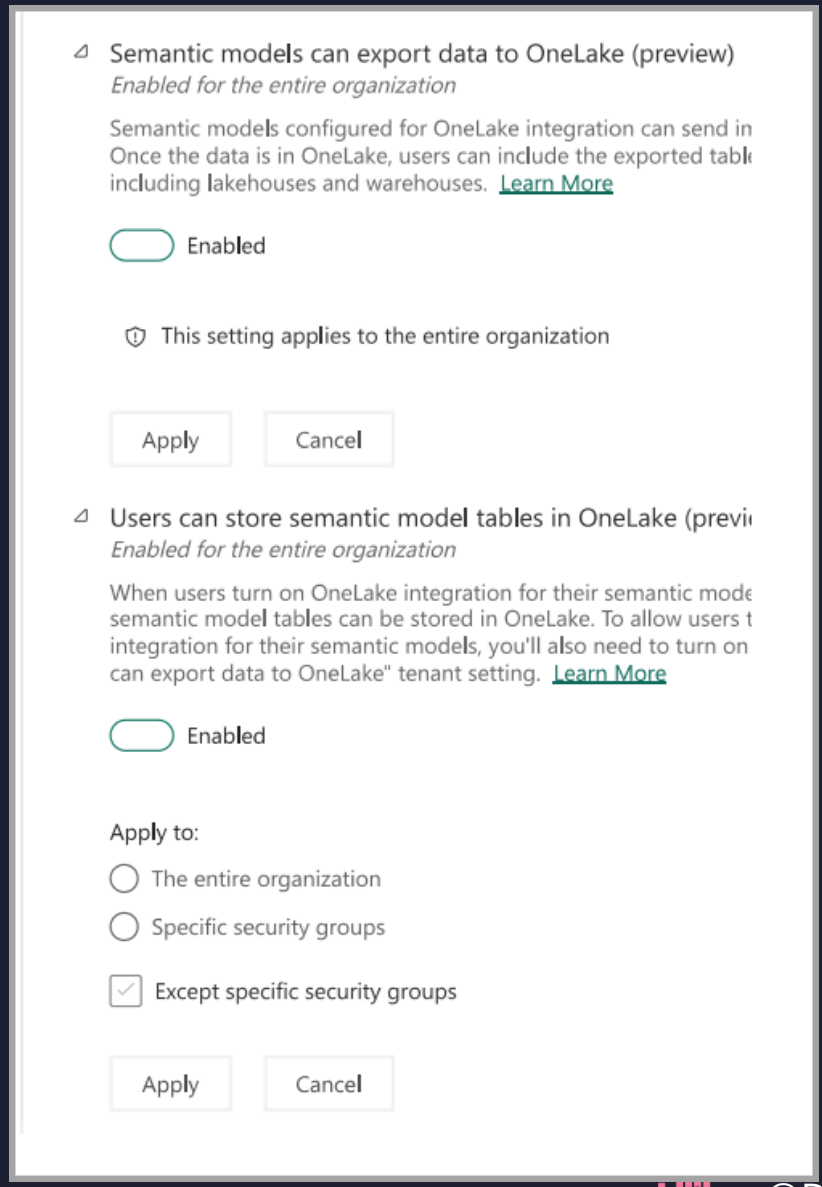
OneLake Integration for Semantic Models and Eventhouse



@DataMozart

OneLake Integration for Semantic Models





@DataMozart

OneLake Integration for Eventhouse

	Turned on	Turned off
KQL Database	- Existing tables aren't affected. New tables are made available in OneLake. - The Data retention policy of your KQL database is also applied to the data in OneLake. Data removed from your KQL database at the end of the retention period is also removed from OneLake.	- Existing tables aren't affected. New tables won't be available in OneLake.
A table in KQL Database	- New data is made available in OneLake. - Existing data isn't backfilled. - Data can't be deleted, truncated, or purged. - Table schema can't be altered and the table can't be renamed. - Row Level Security can't be applied to the table.	- New data isn't made available in OneLake. - Data can be deleted, truncated, or purged. - Table schema can be altered and the table can be renamed. - Data is soft deleted from OneLake.

- Turn on OneLake availability on a KQL database or a table level

Get data

- Create a data connection
- Discover data by using OneLake data hub and real-time hub
- Ingest or access data as needed (shortcuts, dataflows, pipelines, notebooks)
- Choose between a lakehouse, warehouse, or eventhouse
- Implement OneLake integration for eventhouse and semantic models

Transform data

- Create views, functions, and stored procedures
- Enrich data by adding new columns or tables
- Implement a star schema for a lakehouse or warehouse
- Denormalize data
- Aggregate data
- Merge or join data
- Identify and resolve duplicate data, missing data, or null values
- Convert column data types
- Filter data

Query and analyze data

- Select, filter, and aggregate data by using the Visual Query Editor
- Select, filter, and aggregate data by using SQL
- Select, filter, and aggregate data by using KQL

Transform data

- Create views, functions, and stored procedures
- Enrich data by adding new columns or tables
- Implement a star schema for a lakehouse or warehouse
- Denormalize data
- Aggregate data
- Merge or join data
- Identify and resolve duplicate data, missing data, or null values
- Convert column data types
- Filter data

Organizing Data in a Lakehouse

Bronze

Raw

Silver

Validated

Gold

Enriched/Curated

Organizing Data in a Lakehouse

- ✓ Land data from external sources in its original state
- ✓ Serve as a repository of the historical archive of source data
- ✓ Contains unvalidated data
- ✓ Stores the data in the original format

Organizing Data in a Lakehouse

- ✓ Conformed and cleaned data from the bronze layer
- ✓ Ad-hoc analysis, machine learning workloads
- ✓ Contains enriched and validated data
- ✓ Data model normalized to a 3rd normal form
- ✓ Stores the data in Delta/Parquet

Organizing Data in a Lakehouse

- ✓ Structured and organized data for specific project requirements
- ✓ Data additionally cleaned and refined
- ✓ Complex business logic and specific calculations
- ✓ Data model is a Kimball-style star schema
- ✓ Stores the data preferably in Delta

Create Views, Functions, and Stored Procedures

- Warehouse – the same as in the traditional T-SQL workloads
- Lakehouse – with PySpark or SparkSQL
- Eventhouse – supports materialized views

Views don’t work with Direct Lake mode!

Enrich Data by Adding New Columns or Tables

Lakehouse

Warehouse

Eventhouse

Enrich Data by Adding New Columns or Tables

- Use notebooks or Spark jobs
- Operating on a dataframe
- Use withColumn() and select() functions

Lakehouse

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

# Create customer dataframe

dfdimCustomer = df.dropDuplicates(["CustomerName","Email"]).select(col("CustomerName"),col("Email")) \
    .withColumn("First",split(col("CustomerName"), " ").getItem(0)) \
    .withColumn("Last",split(col("CustomerName"), " ").getItem(1))
```

Enrich Data by Adding New Columns or Tables

Warehouse

- ALTER TABLE…ADD COLUMN
- ALTER TABLE…ALTER COLUMN
- ALTER TABLE…DROP COLUMN

```
1    ALTER TABLE dbo.DimProduct
2    ADD ProductSubCategory VARCHAR(255);
```

Enrich Data by Adding New Columns or Tables

Eventhouse

- .alter table command
- Existing non-specified columns will be dropped
- Use .show table [myTable] cslschema to get the existing table schema before you alter it
- Adds nullable column to the end of the schema

```
.alter table MyTable (ColumnX:string, ColumnY:int)
.alter table MyTable (ColumnX:string, ColumnY:int) with (docstring = "Some documentation", folder = "Folder1
```

Enrich Data by Adding New Columns or Tables

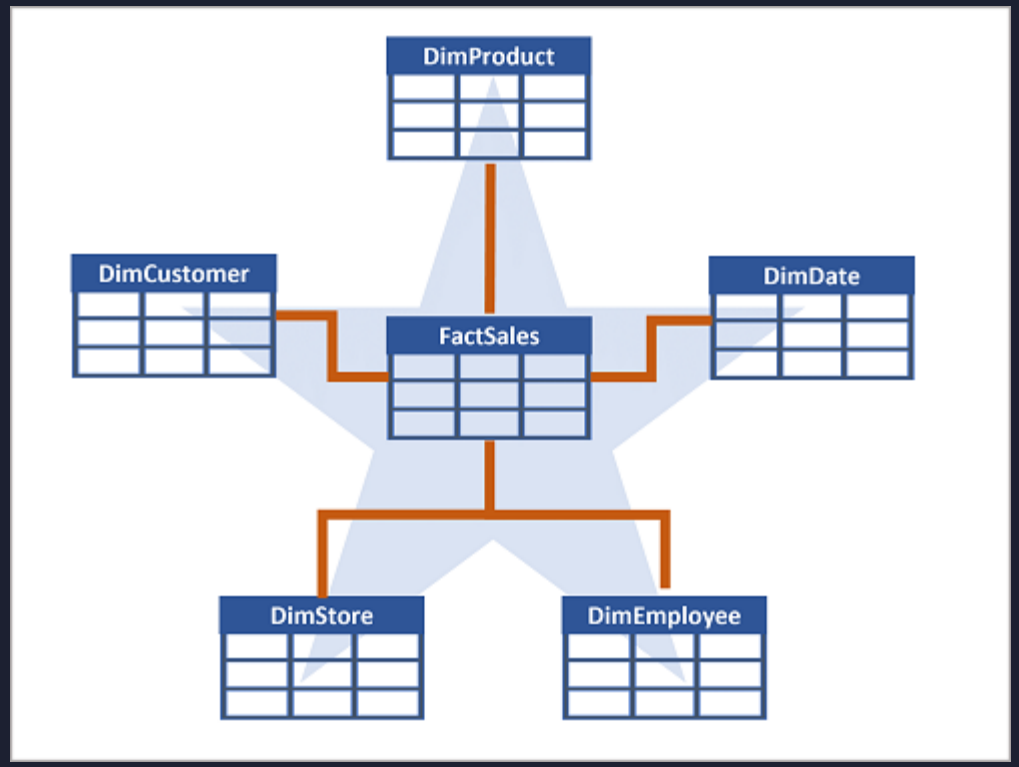
- ***extend*** operator
- Creates calculated column and appends to the end of the result set

Eventhouse

```
StormEvents
| project EndTime, StartTime
| extend Duration = EndTime - StartTime
```

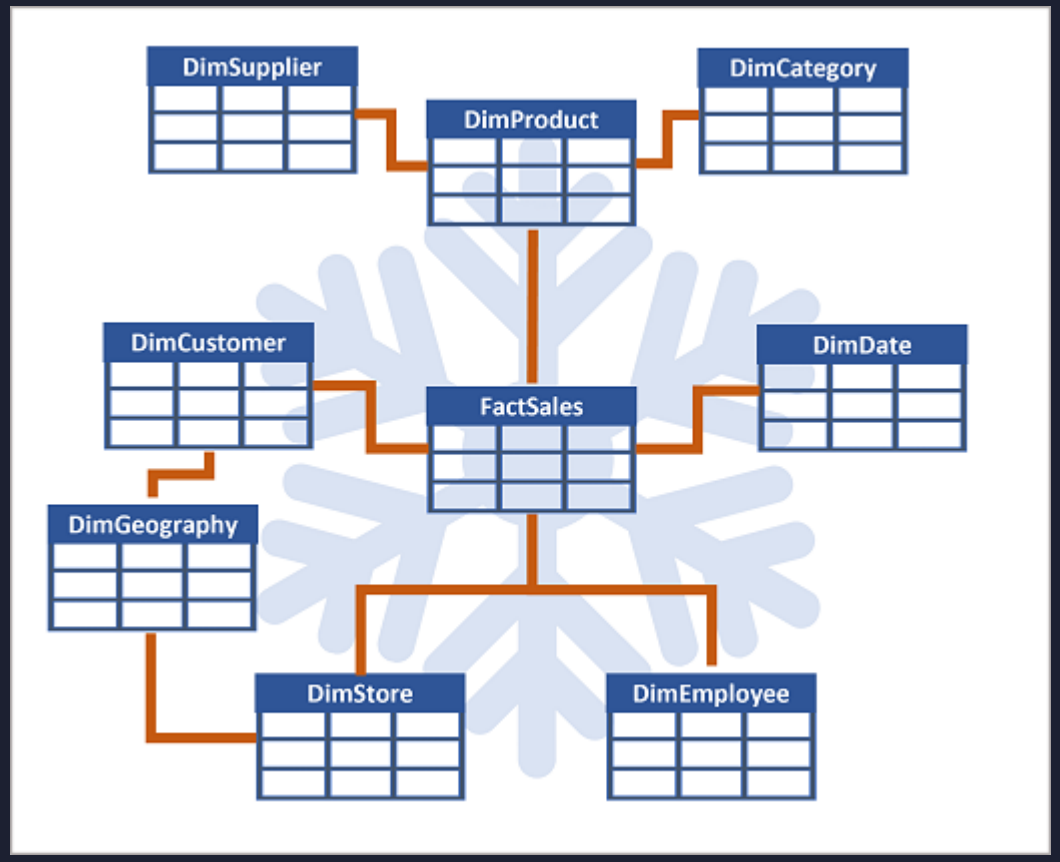
Implement a Star Schema for a Lakehouse

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



Dimensional Modeling Beyond Star Schema

- Star schema, further normalized
- More granular dimensions



Dimension Tables - Extended

Calendar dimension

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

Slowly changing dimension (SCD)

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

Normalization vs. Denormalization

Normalization

Process of organizing the data in a database

- In most cases, 3rd normal form is optimal
- Data writing speed

Denormalization

Creates redundant data in the table

- Data reading speed

@DataMozart

Denormalization

Fact Sales

SalesKey	ProductKey	SalesAmount
123	1	100
456	2	200
789	1	300
357	1	400

Dim Product

ProductKey	ProductSubcategoryKey	ProductName
1	11	Shirt
2	11	Hoodie
3	12	Mug

Dim ProductSubcategory

ProductSubcategoryKey	ProductCategoryKey	SubcategoryName
11	111	Women
12	122	Kitchen

Dim ProductCategory

ProductCategoryKey	CategoryName
111	Clothes
122	House

Denormalization

Fact Sales

SalesKey	ProductKey	SalesAmount
123	1	100
456	2	200
789	1	300
357	1	400

Dim Product

ProductKey	SubcategoryName	CategoryName	ProductName
1	Women	Clothes	Shirt
2	Women	Clothes	Hoodie
3	Kitchen	House	Mug

@DataMozart

Aggregations

Large Fact Table

Date	Customer ID	Product ID	Sales Amount
2021-10-12	123	11	10
2021-10-12	456	12	20
2021-10-12	789	12	50
2021-10-13	123	13	30
2021-10-13	456	11	10

Create Aggregated Tables

Date	Sales Amount
2021-10-12	80
2021-10-13	40

Product ID	Sales Amount
11	20
12	70
13	30

Customer ID	Sales Amount
123	40
456	30
789	50

Date	Customer ID	Product ID	Sales Amount
2021-10-12	123	11	10
2021-10-12	456	12	20
2021-10-12	789	12	50
2021-10-13	123	13	30
2021-10-13	456	11	10

Aggregations in Microsoft Fabric

Lakehouse

Warehouse

Eventhouse

✓ PySpark/SparkSQL

✓ T-SQL

✓ KQL

Merge or Join Data

Lakehouse

✓ PySpark/SparkSQL

Warehouse

✓ T-SQL

Eventhouse

✓ KQL

Dataflow Gen2

✓ Low-code/No-code

Merge or Join Data

Table A

Fruit
Apple
Orange
Banana

Fruit
Apple
Orange
Mango

Table B

Left Outer

Fruit	Fruit
Apple	Apple
Orange	Orange
Banana	

Right Outer

Fruit	Fruit
Apple	Apple
Orange	Orange
	Mango

Inner

Fruit	Fruit
Apple	Apple
Orange	Orange

Full Outer

Fruit	Fruit
Apple	Apple
Orange	Orange
Banana	
	Mango

@DataMozart

Identify and Resolve Duplicate Data

Lakehouse

Warehouse

Eventhouse

✓ PySpark/SparkSQL

✓ T-SQL

✓ KQL

```
from |
# Cre
dfdimCustomer = df.dropDuplica
```

```
DeviceEventsAll
| where EventDateTime > ago(90d)
| summarize hint.strategy=shuffle arg_max(EventDateTime, *) by DeviceId, EventId, StationId
```

```
WITH CTE AS (
    SELECT
    DELETE FROM CTE
    WHERE RowNum > 1;
```

```
ne"),col("Email")) \
```

@DataMozart

Identify and Resolve Missing Data

Lakehouse

Warehouse

Eventhouse

✓ PySpark/SparkSQL

✓ T-SQL

✓ KQL

Identify and Resolve Missing Data

Lakehouse

- ✓ PySpark/SparkSQL

- ✓ *fillna()* and *fill()*
- ✓ Return same results

```
#Replace 0 for null for all integer columns
df.na.fill(value=0).show()

#Replace 0 for null on only population column
df.na.fill(value=0,subset=["population"]).show()


df.na.fill("").show(false)


df.na.fill("unknown",["city"]) \
    .na.fill("",["type"]).show()
```

Identify and Resolve Missing Data

Warehouse

✓ T-SQL

✓ ***COALESCE()*** and ***ISNULL()***

First non-null expression

```
SELECT Name, Class, Color, ProductNumber,
COALESCE(Class, Color, ProductNumber) AS FirstNotNull
FROM Production.Product;
```

Replace NULL with the specified value

```
SELECT Description, DiscountPct, MinQty, ISNULL(MaxQty, 0.00) AS 'Max Quantity'
FROM Sales.SpecialOffer;
```

Identify and Resolve Missing Data

Eventhouse

✓ KQL

✓ ***isnull()*** – returns a Boolean result for non-string columns

✓ ***isempty()*** – returns a Boolean result for string columns

a	b	isnull_a	isempty_a	strlen_a	isnull_b
		false	true	0	true
		false	false	1	true
a	1	false	false	1	false

✓ ***series_fill_const()***– replaces missing values in a series with a specified constant value

✓ ***coalesce()***

@DataMozart

Convert Column Data Types

Lakehouse

✓ PySpark/SparkSQL

Warehouse

✓ T-SQL

Eventhouse

✓ KQL

Dataflow Gen2

✓ Low-code/No-code

Pipeline

✓ Copy data

Convert Column Data Types

Lakehouse

- ✓ *cast()*

Warehouse

- ✓ ALTER TABLE not supported!
- ✓ Workaround: *sp_rename* table -> CTAS with new data types

Eventhouse

- ✓ *.alter column*

```
.alter column ['Table'].['ColumnX'] type=string
```

- ✓ *.set-or-append* – creates a new table and preserves existing data while changing the column type

```
.set-or-append NewTable <| OriginalTable | extend Col1=tostring(Col1)
```

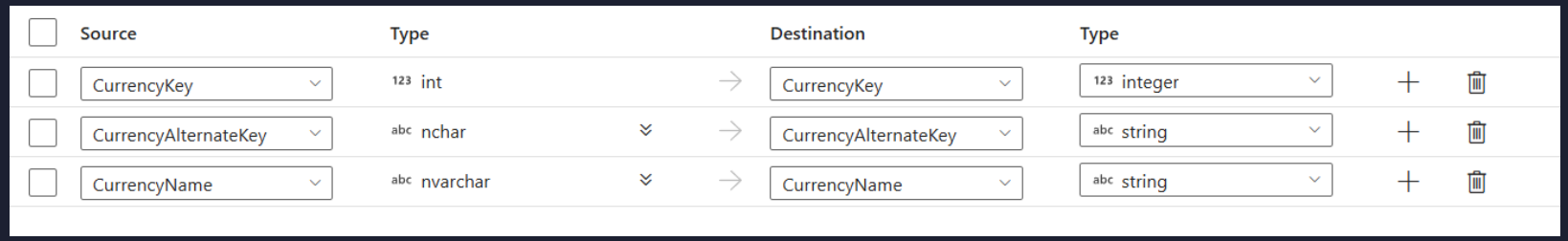
Convert Column Data Types

Dataflow Gen2

- ✓ Low-code/No-code

Pipeline

- ✓ Copy data



	Source	Type		Destination	Type
☐	CurrencyKey	123 int	→	CurrencyKey	123 integer
☐	CurrencyAlternateKey	abc nchar	→	CurrencyAlternateKey	abc string
☐	CurrencyName	abc nvarchar	→	CurrencyName	abc string

@DataMozart

Filter Data

Lakehouse

- ✓ PySpark/SparkSQL
- ✓ *select()* – subset of columns
- ✓ *filter()* – subset of rows

Warehouse

- ✓ T-SQL
- ✓ *select()* – subset of columns
- ✓ *where()* – subset of rows

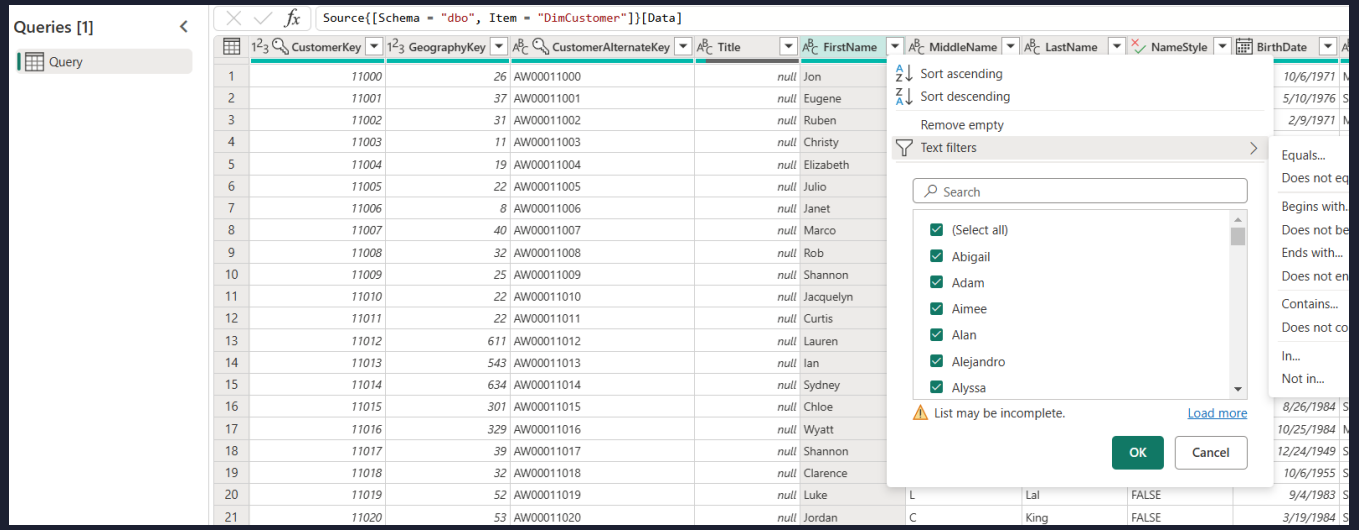
Eventhouse

- ✓ KQL
- ✓ *project()* – subset of columns
- ✓ *where()* – subset of rows

Filter Data

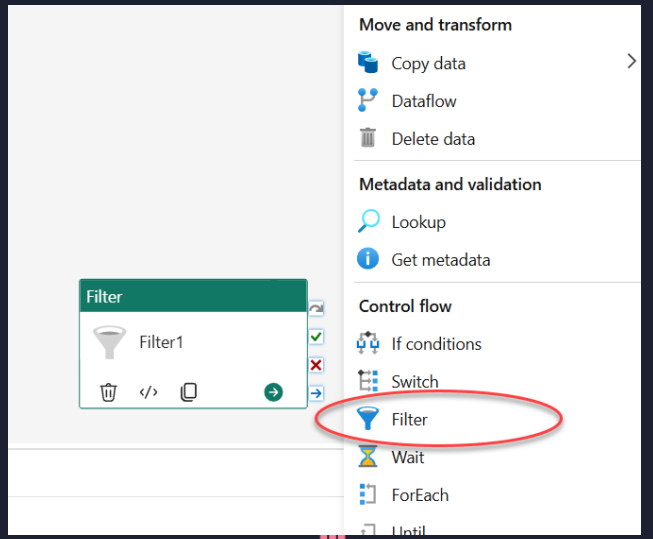
Dataflow Gen2

✓ Low-code/No-code



Pipeline

✓ Filter activity



@DataMozart

Get data

- Create a data connection
- Discover data by using OneLake data hub and real-time hub
- Ingest or access data as needed (shortcuts, dataflows, pipelines, notebooks)
- Choose between a lakehouse, warehouse, or eventhouse
- Implement OneLake integration for eventhouse and semantic models

Transform data

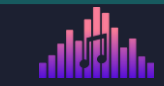
- Create views, functions, and stored procedures
- Enrich data by adding new columns or tables
- Implement a star schema for a lakehouse or warehouse
- Denormalize data
- Aggregate data
- Merge or join data
- Identify and resolve duplicate data, missing data, or null values
- Convert column data types
- Filter data

Query and analyze data

- Select, filter, and aggregate data by using the Visual Query Editor
- Select, filter, and aggregate data by using SQL
- Select, filter, and aggregate data by using KQL

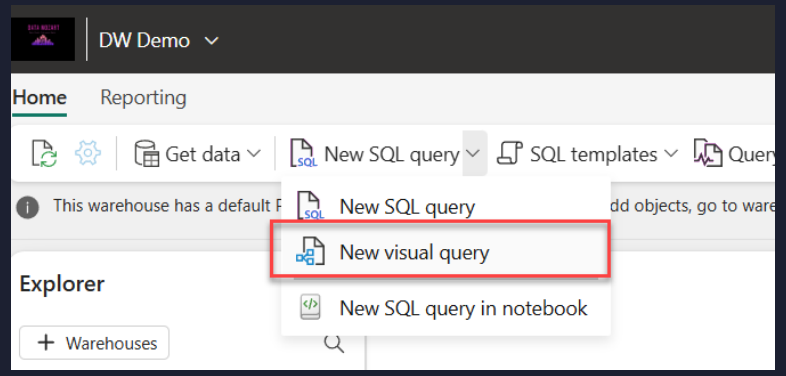
Query and analyze data

- Select, filter, and aggregate data by using the Visual Query Editor
- Select, filter, and aggregate data by using SQL
- Select, filter, and aggregate data by using KQL

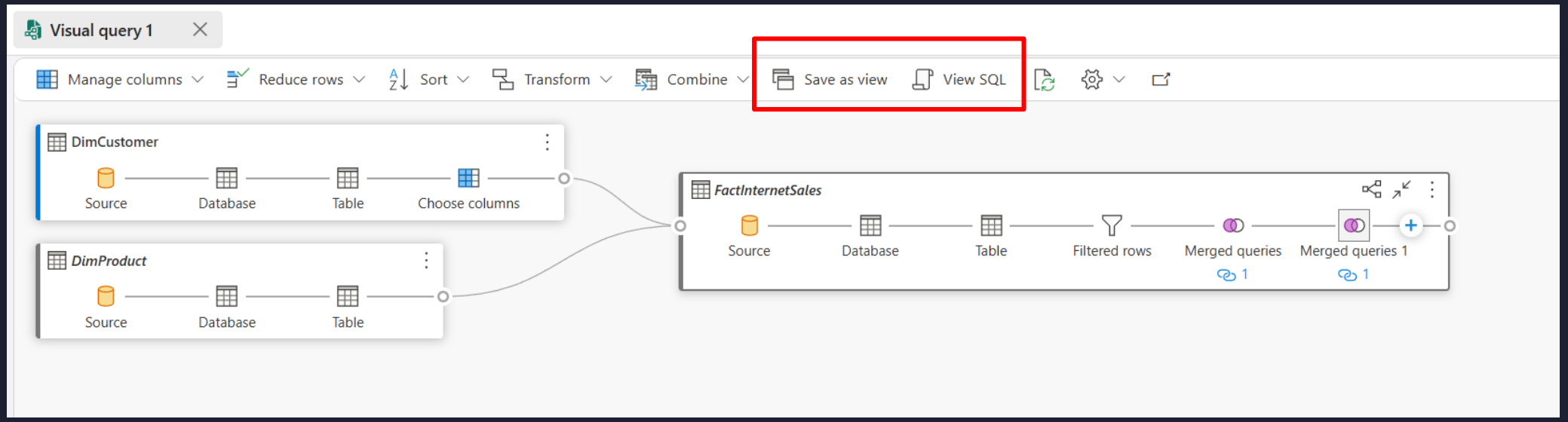


Query Data with Visual Query Editor

1



2



@DataMozart

Query Data with SQL

Window functions

OVER (defines a window)

PARTITION BY (optional, breaks the rows into smaller sets)

ORDER BY (required depending on the function)

```
1   --ROW_NUMBER()
2   SELECT SalesOrderNumber
3       , OrderDate
4       , CustomerKey
5       , ROW_NUMBER() OVER (PARTITION BY CustomerKey ORDER BY OrderDate) AS rnm
6   FROM aw_factinternetsales
```

@DataMozart

Query Data with SQL

Ranking functions

ROW_NUMBER

Results Messages

	SalesOrderID	OrderDate	CustomerID	RowNum
1	52634	2013-07-15 00:00:00.000	11330	1
2	53205	2013-07-26 00:00:00.000	11330	2
3	54112	2013-08-09 00:00:00.000	11330	3
4	57792	2013-10-11 00:00:00.000	11330	4
5	58084	2013-10-16 00:00:00.000	11330	5
6	58556	2013-10-24 00:00:00.000	11330	6
7	58557	2013-10-24 00:00:00.000	11330	7
8	59422	2013-11-04 00:00:00.000	11330	8
9	59510	2013-11-05 00:00:00.000	11330	9
10	60200	2013-11-15 00:00:00.000	11330	10
11	60337	2013-11-17 00:00:00.000	11330	11
12	61555	2013-12-05 00:00:00.000	11330	12
13	61800	2013-12-09 00:00:00.000	11330	13
14	62542	2013-12-21 00:00:00.000	11330	14
15	62560	2013-12-21 00:00:00.000	11330	15
16	64950	2014-01-26 00:00:00.000	11330	16

RANK

Results Messages

	SalesOrderID	OrderDate	CustomerID	RowNum	Rnk
1	52634	2013-07-15 00:00:00.000	11330	1	1
2	53205	2013-07-26 00:00:00.000	11330	2	2
3	54112	2013-08-09 00:00:00.000	11330	3	3
4	57792	2013-10-11 00:00:00.000	11330	4	4
5	58084	2013-10-16 00:00:00.000	11330	5	5
6	58556	2013-10-24 00:00:00.000	11330	6	6
7	58557	2013-10-24 00:00:00.000	11330	7	6
8	59422	2013-11-04 00:00:00.000	11330	8	8
9	59510	2013-11-05 00:00:00.000	11330	9	9
10	60200	2013-11-15 00:00:00.000	11330	10	10
11	60337	2013-11-17 00:00:00.000	11330	11	11
12	61555	2013-12-05 00:00:00.000	11330	12	12
13	61800	2013-12-09 00:00:00.000	11330	13	13
14	62542	2013-12-21 00:00:00.000	11330	14	14
15	62560	2013-12-21 00:00:00.000	11330	15	14
16	64950	2014-01-26 00:00:00.000	11330	16	16

DENSE_RANK

Results Messages

	SalesOrderID	OrderDate	CustomerID	RowNum	Rnk	DenseRnk
1	52634	2013-07-15 00:00:00.000	11330	1	1	1
2	53205	2013-07-26 00:00:00.000	11330	2	2	2
3	54112	2013-08-09 00:00:00.000	11330	3	3	3
4	57792	2013-10-11 00:00:00.000	11330	4	4	4
5	58084	2013-10-16 00:00:00.000	11330	5	5	5
6	58556	2013-10-24 00:00:00.000	11330	6	6	6
7	58557	2013-10-24 00:00:00.000	11330	7	6	6
8	59422	2013-11-04 00:00:00.000	11330	8	8	7
9	59510	2013-11-05 00:00:00.000	11330	9	9	8
10	60200	2013-11-15 00:00:00.000	11330	10	10	9
11	60337	2013-11-17 00:00:00.000	11330	11	11	10
12	61555	2013-12-05 00:00:00.000	11330	12	12	11
13	61800	2013-12-09 00:00:00.000	11330	13	13	12
14	62542	2013-12-21 00:00:00.000	11330	14	14	13
15	62560	2013-12-21 00:00:00.000	11330	15	14	13
16	64950	2014-01-26 00:00:00.000	11330	16	16	14

Query Data with SQL

Offset functions

LAG

Results | Messages

	CustomerID	OrderDate	SalesOrderID	PrevOrder
1	11000	2011-06-21 00:00:00.000	43793	NULL
2	11000	2013-06-20 00:00:00.000	51522	43793
3	11000	2013-10-03 00:00:00.000	57418	51522
4	11001	2011-06-17 00:00:00.000	43767	NULL
5	11001	2013-06-18 00:00:00.000	51493	43767
6	11001	2014-05-12 00:00:00.000	72773	51493
7	11002	2011-06-09 00:00:00.000	43736	NULL
8	11002	2013-06-02 00:00:00.000	51238	43736
9	11002	2013-07-26 00:00:00.000	53237	51238
10	11003	2011-05-31 00:00:00.000	43701	NULL
11	11003	2013-06-07 00:00:00.000	51315	43701
12	11003	2013-10-10 00:00:00.000	57783	51315
13	11004	2011-06-25 00:00:00.000	43810	NULL
14	11004	2013-06-24 00:00:00.000	51595	43810
15	11004	2013-10-01 00:00:00.000	57293	51595
16	11005	2011-06-01 00:00:00.000	43704	NULL

LEAD

Results | Messages

	CustomerID	OrderDate	SalesOrderID	NextOrder
1	11000	2011-06-21 00:00:00.000	43793	51522
2	11000	2013-06-20 00:00:00.000	51522	57418
3	11000	2013-10-03 00:00:00.000	57418	NULL
4	11001	2011-06-17 00:00:00.000	43767	51493
5	11001	2013-06-18 00:00:00.000	51493	72773
6	11001	2014-05-12 00:00:00.000	72773	NULL
7	11002	2011-06-09 00:00:00.000	43736	51238
8	11002	2013-06-02 00:00:00.000	51238	53237
9	11002	2013-07-26 00:00:00.000	53237	NULL
10	11003	2011-05-31 00:00:00.000	43701	51315
11	11003	2013-06-07 00:00:00.000	51315	57783
12	11003	2013-10-10 00:00:00.000	57783	NULL
13	11004	2011-06-25 00:00:00.000	43810	51595
14	11004	2013-06-24 00:00:00.000	51595	57293
15	11004	2013-10-01 00:00:00.000	57293	NULL
16	11005	2011-06-01 00:00:00.000	43704	51612

FIRST_VALUE/LAST_VALUE

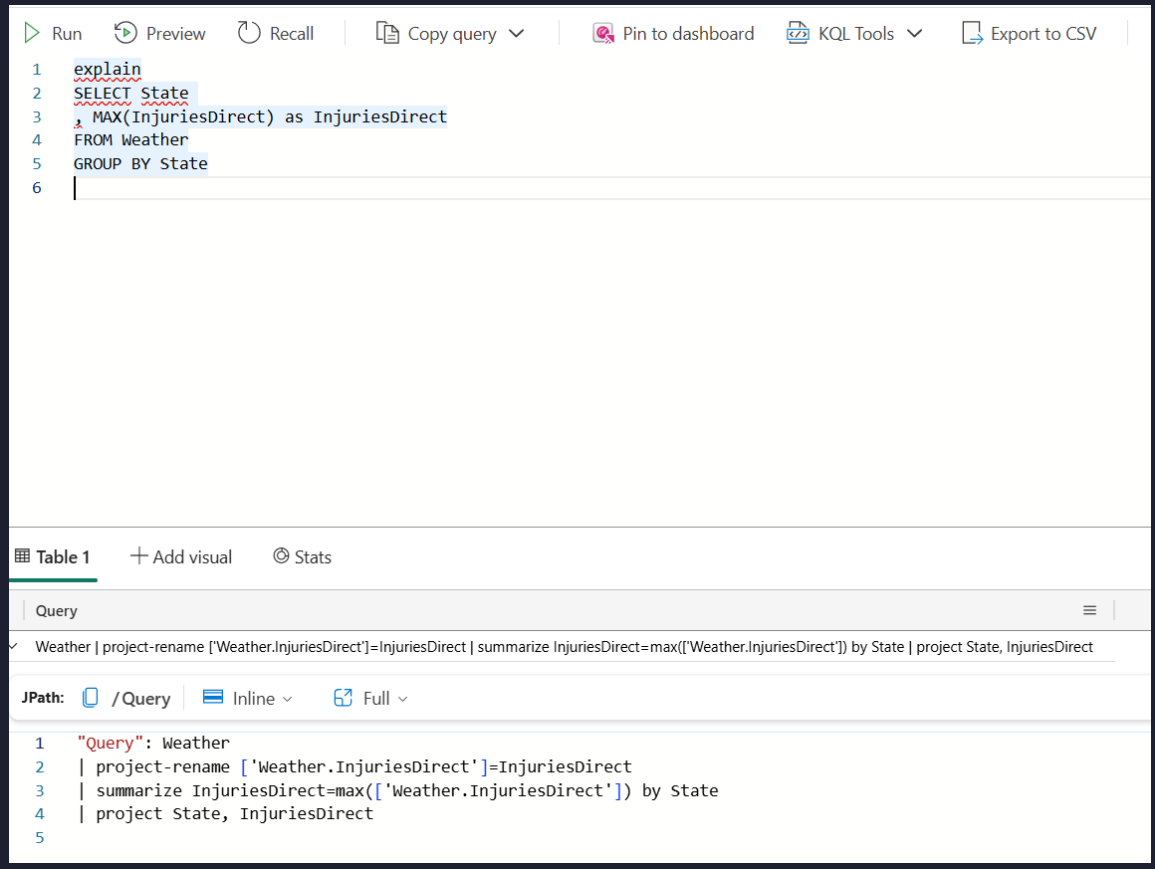
Results | Messages

	CustomerID	OrderDate	SalesOrderID	FirstOrder
1	11000	2011-06-21 00:00:00.000	43793	43793
2	11000	2013-06-20 00:00:00.000	51522	43793
3	11000	2013-10-03 00:00:00.000	57418	43793
4	11001	2011-06-17 00:00:00.000	43767	43767
5	11001	2013-06-18 00:00:00.000	51493	43767
6	11001	2014-05-12 00:00:00.000	72773	43767
7	11002	2011-06-09 00:00:00.000	43736	43736
8	11002	2013-06-02 00:00:00.000	51238	43736
9	11002	2013-07-26 00:00:00.000	53237	43736
10	11003	2011-05-31 00:00:00.000	43701	43701
11	11003	2013-06-07 00:00:00.000	51315	43701
12	11003	2013-10-10 00:00:00.000	57783	43701
13	11004	2011-06-25 00:00:00.000	43810	43810
14	11004	2013-06-24 00:00:00.000	51595	43810
15	11004	2013-10-01 00:00:00.000	57293	43810
16	11005	2011-06-01 00:00:00.000	43704	43704

Query Data with KQL

KQL Queryset

- Collection of one or more KQL queries
- Sharing queries with others
- *explain* -> converts SQL to KQL
- You can also write T-SQL! But…



Get data

- Create a data connection
- Discover data by using OneLake data hub and real-time hub
- Ingest or access data as needed (shortcuts, dataflows, pipelines, notebooks)
- Choose between a lakehouse, warehouse, or eventhouse
- Implement OneLake integration for eventhouse and semantic models

Transform data

- Create views, functions, and stored procedures
- Enrich data by adding new columns or tables
- Implement a star schema for a lakehouse or warehouse
- Denormalize data
- Aggregate data
- Merge or join data
- Identify and resolve duplicate data, missing data, or null values
- Convert column data types
- Filter data

Query and analyze data

- Select, filter, and aggregate data by using the Visual Query Editor
- Select, filter, and aggregate data by using SQL
- Select, filter, and aggregate data by using KQL

@DataMozart

Implement and Manage Semantic Models

25-30%

Design and build semantic models

- Choose a storage mode
- Implement star schema for semantic model
- Implement relationships (bridge tables and many-to-many)
- DAX calculations with variables and functions (iterator, filtering, window, information)
- Implement calculation groups, dynamic format strings, and field parameters
- Identify and configure large semantic model storage format
- Design composite models

Optimize enterprise-scale semantic models

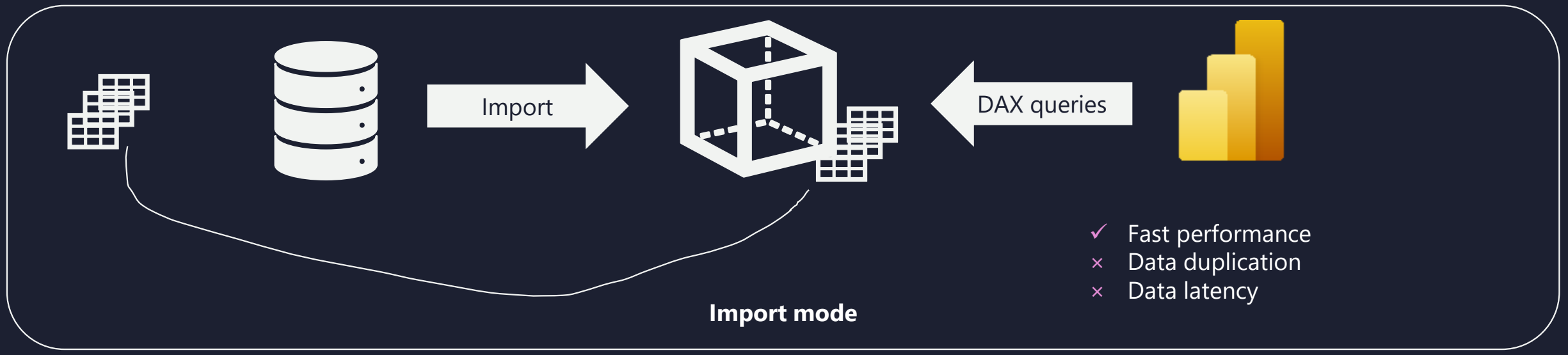
- Implement performance improvements in queries and visuals
- Improve DAX performance
- Configure Direct Lake (default fallback and refresh behavior)
- Implement incremental refresh

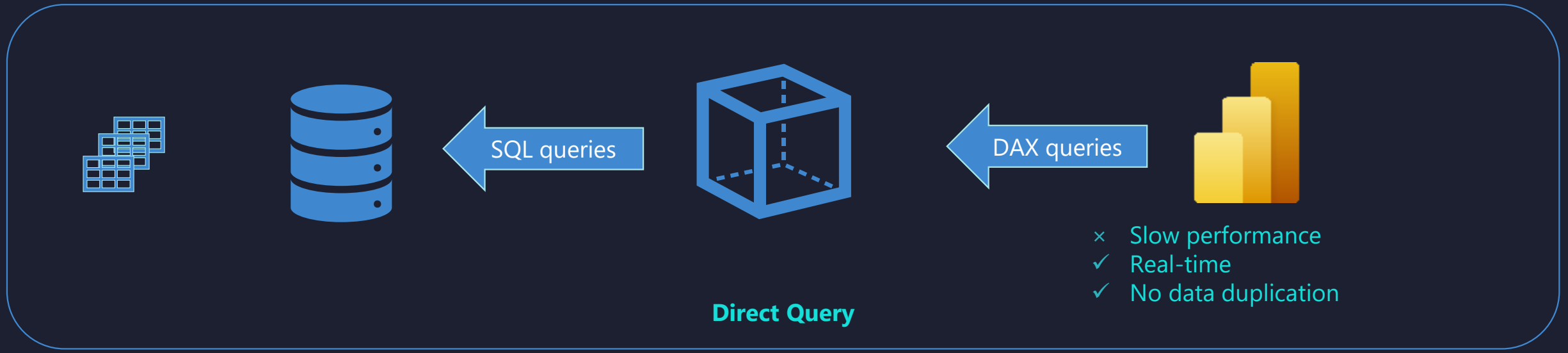
@DataMozart

Design and build semantic models

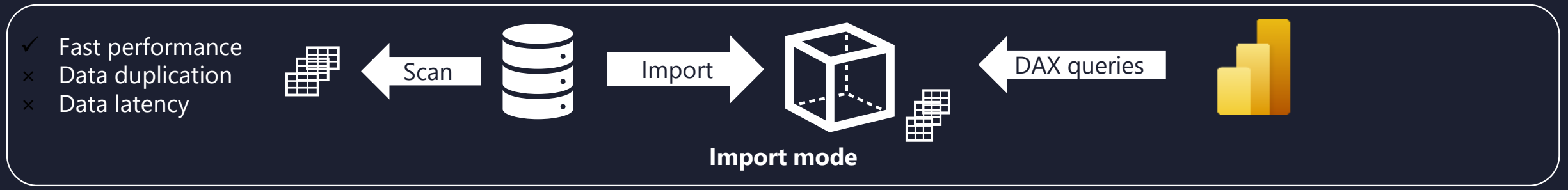
- Choose a storage mode
- Implement star schema for semantic model
- Implement relationships (bridge tables and many-to-many)
- DAX calculations with variables and functions (iterator, filtering, window, information)
- Implement calculation groups, dynamic format strings, and field parameters
- Identify and configure large semantic model storage format
- Design composite models

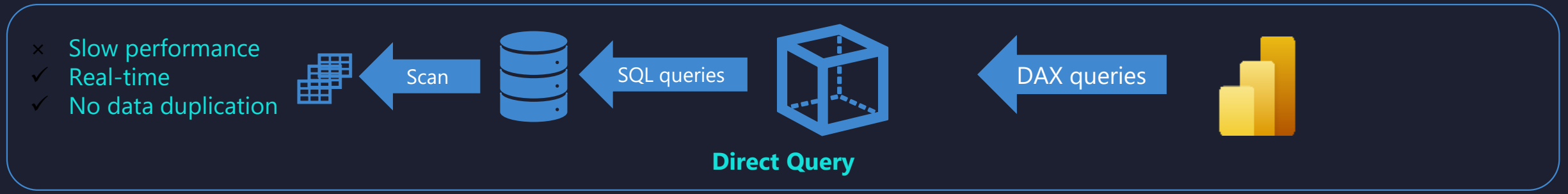
Power BI Architecture – Pre-Fabric

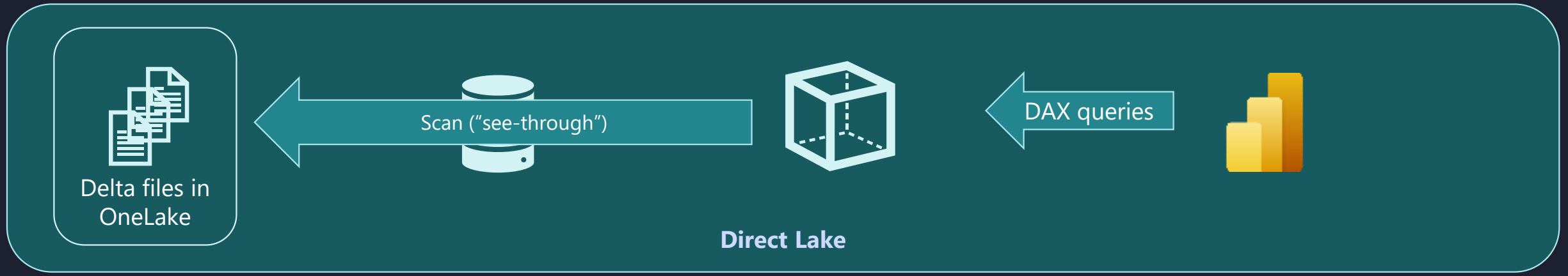




Power BI Architecture – Fabric







Direct Lake for Power BI

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

@DataMozart

Limitations of Direct Lake

- Querying one single Lakehouse or Warehouse
- T-SQL Views are not supported (will fall back to DirectQuery)
- DAX queries exceeding limits or using unsupported features fall back to DirectQuery mode
- No DAX calculated columns
- No composite model
- No DateTime relationships

Always check the list of current limitations!

Implement Relationships

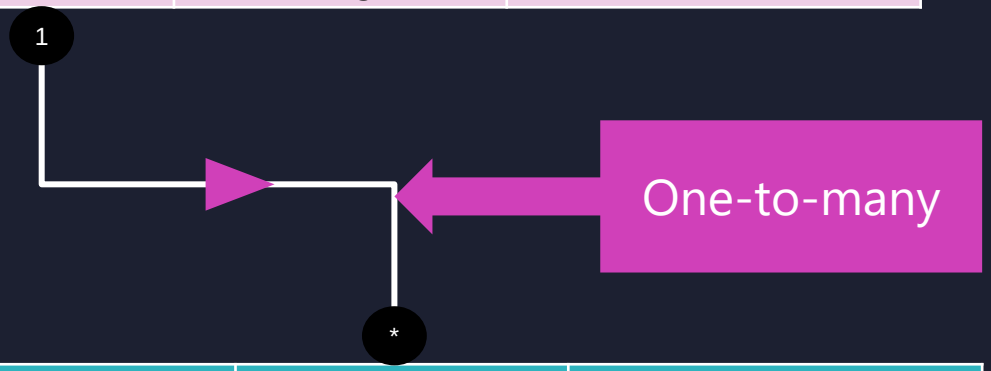
Relationships serve as filters!

Cardinality

- One-to-many (1:*)
- Many-to-one (*:1)
- One-to-one (1:1)
- Many-to-many (*:*)

Relationship Cardinality (1:*, *:1)

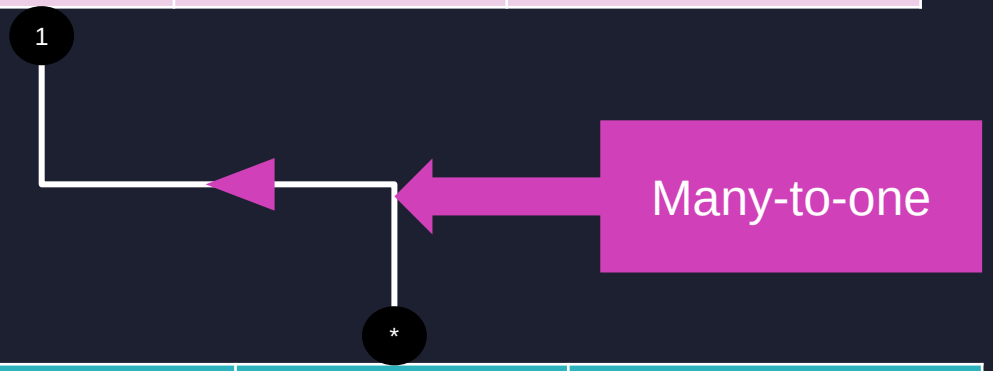
Product Key	Product Name	Product Category
1	T-Shirt	Clothes
2	Socks	Clothes
3	Mug	Accessories



Sales Key	Product Key	Sales Amount
1	1	100
2	1	200
3	1	300
4	2	50
5	2	75
6	3	100
7	3	150

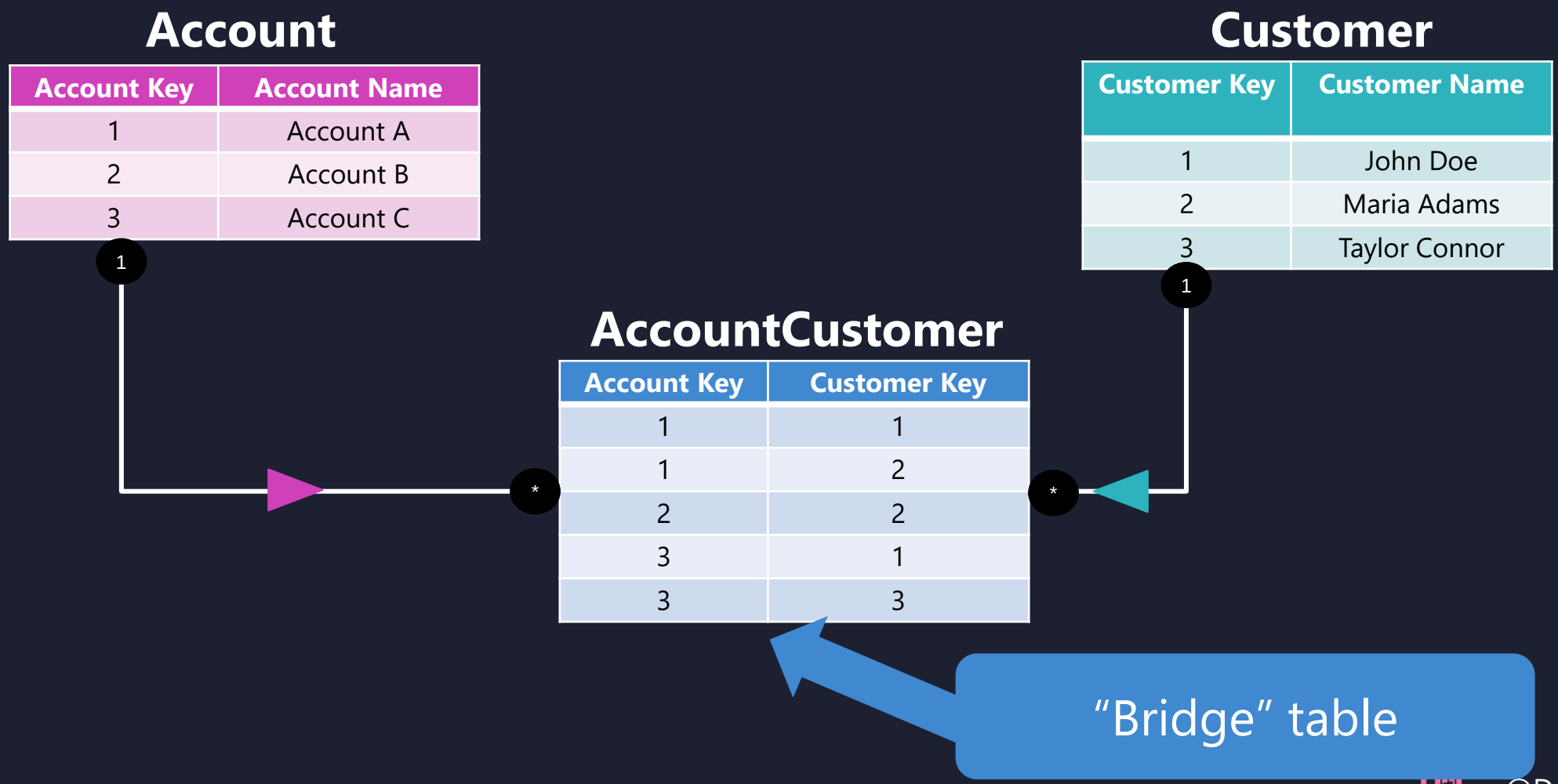
Relationship Cardinality (1:*, *:1)

Product Key	Product Name	Product Category
1	T-Shirt	Clothes
2	Socks	Clothes
3	Mug	Accessories



Sales Key	Product Key	Sales Amount
1	1	100
2	1	200
3	1	300
4	2	50
5	2	75
6	3	100
7	3	150

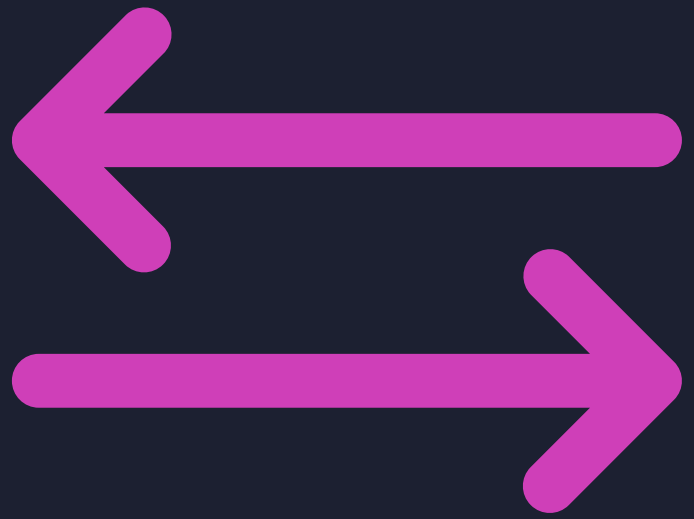
Relationship Cardinality (*:*)



Relationship Filter Direction

Single Direction

Recommended practice



Both Directions

Implement with care

Relationship Filter Direction (Single)

Product Category Key	Product Category
1	Clothes
2	Accessories

1

*

Product Key	Product Name	Product Category Key
1	T-Shirt	1
2	Socks	1
3	Mug	2

1

Sales Key	Product Key	Sales Amount
1	1	100
2	1	200
3	1	300
4	2	50
5	2	75
6	3	100
7	3	150

*

@DataMozart

Relationship Filter Direction (Both)

Product Category Key	Product Category
1	Clothes
2	Accessories

1

*

Product Key	Product Name	Product Category Key
1	T-Shirt	1
2	Socks	1
3	Mug	2

1

Sales Key	Product Key	Sales Amount
1	1	100
2	1	200
3	1	300
4	2	50
5	2	75
6	3	100
7	3	150

*

CROSSFILTER()

@DataMozart

DAX Variables

- Reuse expression logic
- Improve performance
- Easy debug

```
YoY Variance % =
    var _currentSales = SUM(FactResellerSales[SalesAmount])
    var _prevYearSales = CALCULATE(
                            SUM(FactResellerSales[SalesAmount]),
                            SAMEPERIODLASTYEAR(DimDate[FullDateAlternateKey])
                        )
    var _result = DIVIDE(_currentSales - _prevYearSales,_prevYearSales,0)
    RETURN
    _result
```

```
YoY Variance % NO VARs =
    DIVIDE(
        SUM(FactResellerSales[SalesAmount]) -
        CALCULATE(
                    SUM(FactResellerSales[SalesAmount]),
                    SAMEPERIODLASTYEAR(DimDate[FullDateAlternateKey])
                    )
        ,CALCULATE(
                    SUM(FactResellerSales[SalesAmount]),
                    SAMEPERIODLASTYEAR(DimDate[FullDateAlternateKey])
                    )
        ,0
                    )
```

DAX Iterator Functions

- Iterate over a table, evaluate the expression for each row, and then aggregate result
- Ends with X
- You must use iterators whenever more than 1 column is used in the expression

AVERAGEX()
COUNTX()
SUMX()
MINX()
MAXX()
CONCATENATEX()

```
Total Sales = SUM(Table[Column])
```

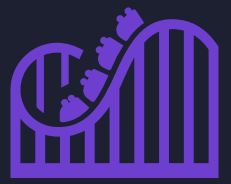
CORRESPONDS TO

```
Total Sales = SUMX(Table,
                   [Column]
              )
```

INDEX(), OFFSET(), WINDOW()

Sort customers by the total amount spent

Compare the sales from the current year and previous year



Calculate moving averages or running totals in N quarters, months...

PARTITIONBY(), ORDERBY()

PARTITIONBY()

Determines the “window” as a subset of rows

ORDERBY()

Defines the sorting order within the “window”

INDEX()

Returns a particular table row based on its position

```
.....
RETURN
        INDEX(
                1,
                [TableName],
                ORDERBY ([SalesAmount], DESC)
        )
```

Brand	SalesAmount
Brand1	5000
Brand2	4000
Brand3	3000
Brand4	2000

INDEX with PARTITIONBY()

Returns a particular table row based on its position

```
.....
RETURN
        INDEX(
                1,
                [TableName],
                ORDERBY ([SalesAmount],
DESC),
                PARTITIONBY([Category])
        )
```

Category	Brand	SalesAmount
Audio	Brand3	2000
Computers	Brand2	1000
Accessories	Brand1	1500
Video	Brand2	1000

OFFSET()

Year	SalesAmount	SalesAmountOffset -1
2020	2000	
2021	3000	2000
2022	4000	3000
2023	3000	4000

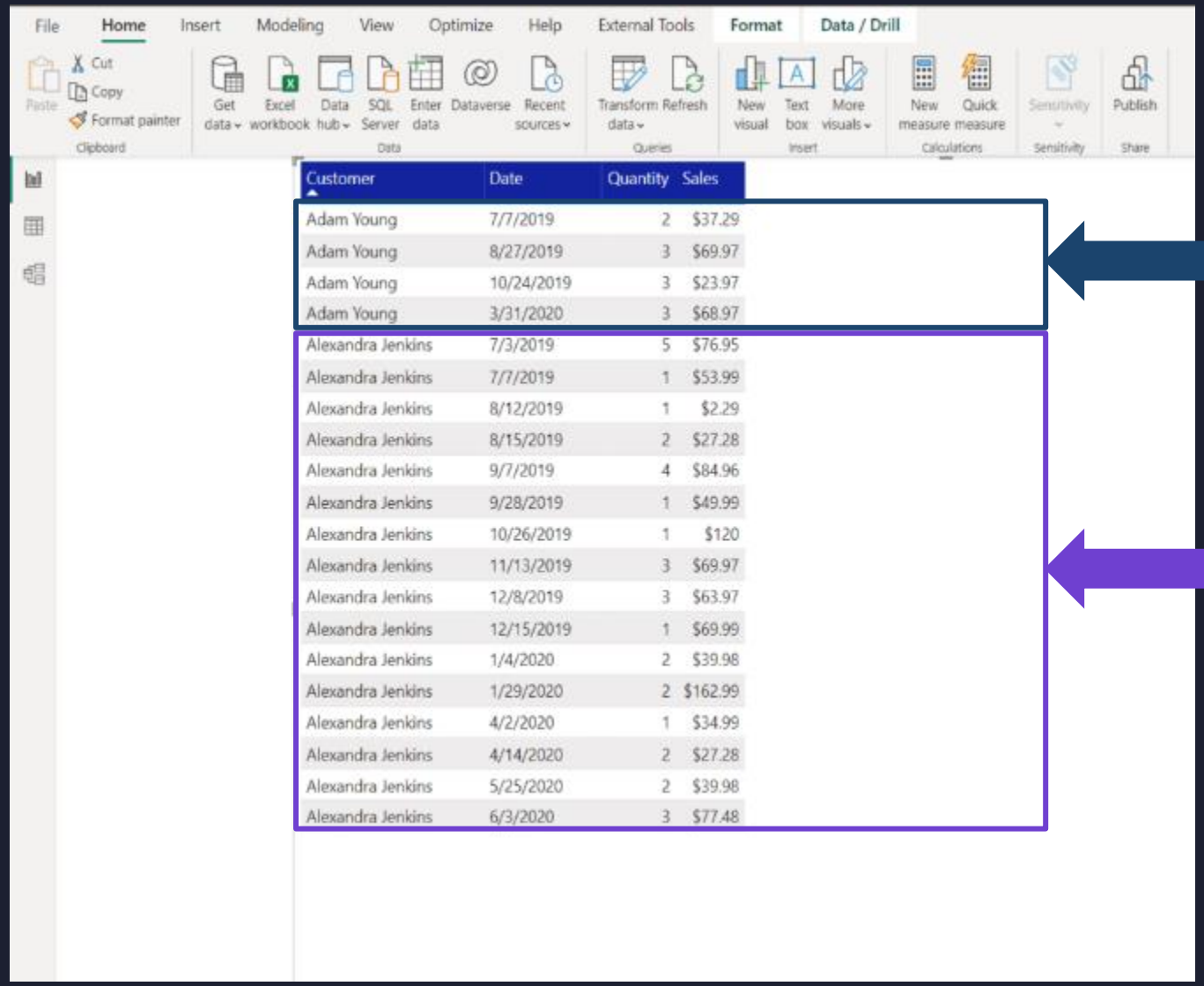
```
RETURN
        OFFSET(
                -1,
                ORDERBY ([YEAR], DESC)
        )
```

OFFSET()

Year	SalesAmount	SalesAmountOffset -1
2020	2000	
2021	3000	
2022	4000	2000
2023	3000	3000

```
RETURN
        OFFSET(
                -2,
                ORDERBY ([YEAR], DESC)
        )
```

WINDOW()



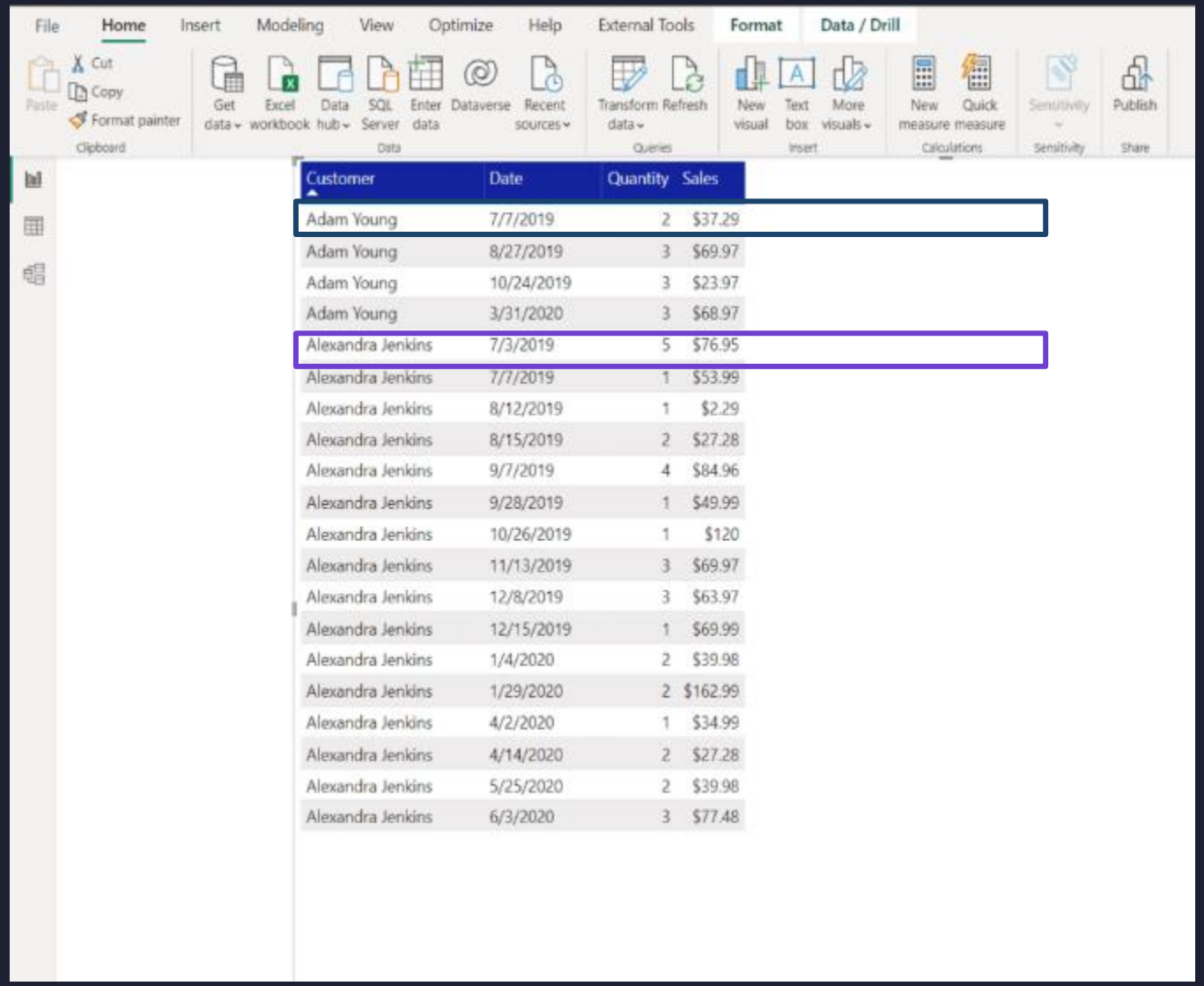
Customer	Date	Quantity	Sales
Adam Young	7/7/2019	2	$37.29
Adam Young	8/27/2019	3	$69.97
Adam Young	10/24/2019	3	$23.97
Adam Young	3/31/2020	3	$68.97
Alexandra Jenkins	7/3/2019	5	$76.95
Alexandra Jenkins	7/7/2019	1	$53.99
Alexandra Jenkins	8/12/2019	1	$2.29
Alexandra Jenkins	8/15/2019	2	$27.28
Alexandra Jenkins	9/7/2019	4	$84.96
Alexandra Jenkins	9/28/2019	1	$49.99
Alexandra Jenkins	10/26/2019	1	$120
Alexandra Jenkins	11/13/2019	3	$69.97
Alexandra Jenkins	12/8/2019	3	$63.97
Alexandra Jenkins	12/15/2019	1	$69.99
Alexandra Jenkins	1/4/2020	2	$39.98
Alexandra Jenkins	1/29/2020	2	$162.99
Alexandra Jenkins	4/2/2020	1	$34.99
Alexandra Jenkins	4/14/2020	2	$27.28
Alexandra Jenkins	5/25/2020	2	$39.98
Alexandra Jenkins	6/3/2020	3	$77.48

Window 1

Window 2

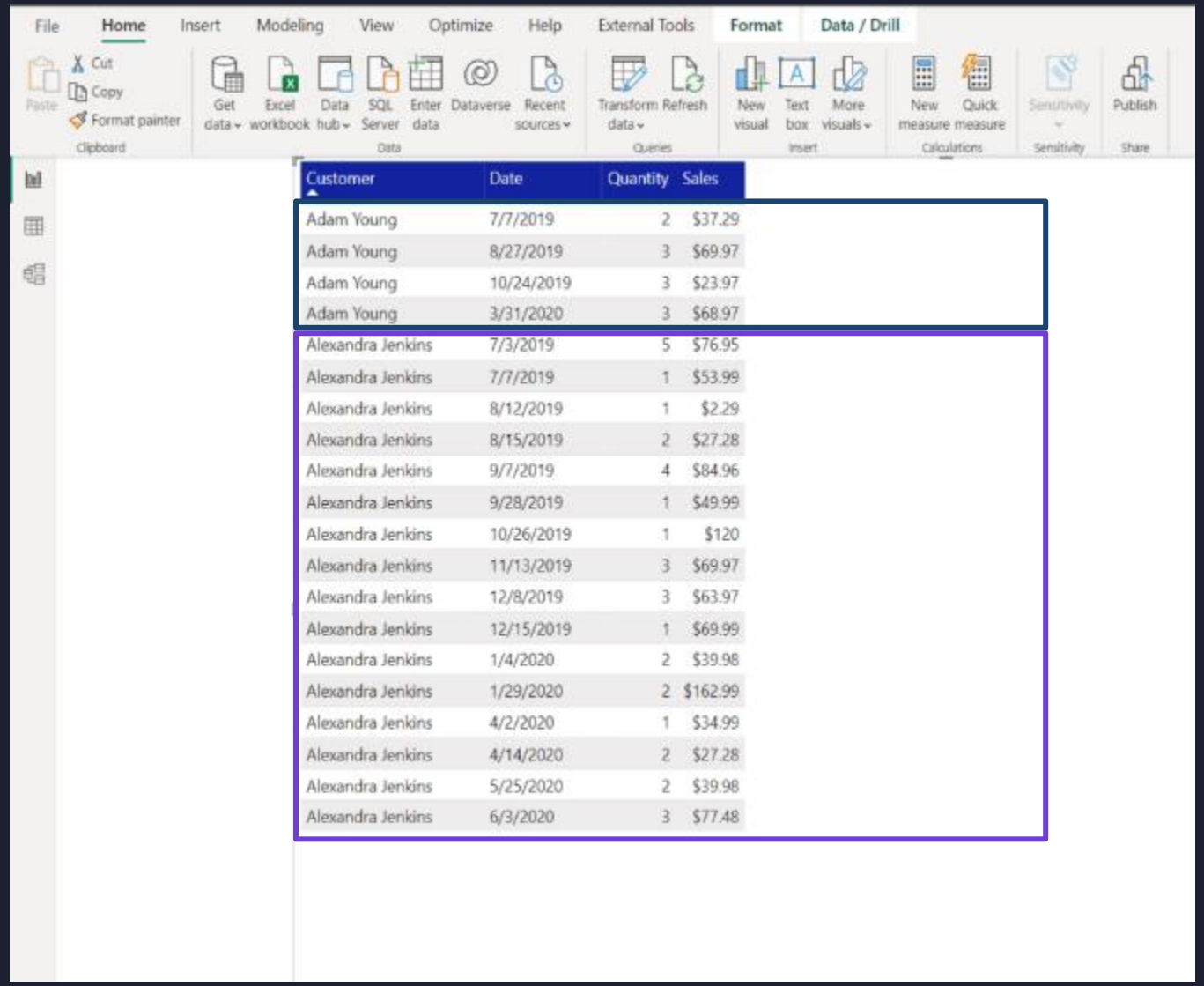
@DataMozart

WINDOW()



Customer	Date	Quantity	Sales
Adam Young	7/7/2019	2	$37.29
Adam Young	8/27/2019	3	$69.97
Adam Young	10/24/2019	3	$23.97
Adam Young	3/31/2020	3	$68.97
Alexandra Jenkins	7/3/2019	5	$76.95
Alexandra Jenkins	7/7/2019	1	$53.99
Alexandra Jenkins	8/12/2019	1	$2.29
Alexandra Jenkins	8/15/2019	2	$27.28
Alexandra Jenkins	9/7/2019	4	$84.96
Alexandra Jenkins	9/28/2019	1	$49.99
Alexandra Jenkins	10/26/2019	1	$120
Alexandra Jenkins	11/13/2019	3	$69.97
Alexandra Jenkins	12/8/2019	3	$63.97
Alexandra Jenkins	12/15/2019	1	$69.99
Alexandra Jenkins	1/4/2020	2	$39.98
Alexandra Jenkins	1/29/2020	2	$162.99
Alexandra Jenkins	4/2/2020	1	$34.99
Alexandra Jenkins	4/14/2020	2	$27.28
Alexandra Jenkins	5/25/2020	2	$39.98
Alexandra Jenkins	6/3/2020	3	$77.48

WINDOW()



Customer	Date	Quantity	Sales
Adam Young	7/7/2019	2	$37.29
Adam Young	8/27/2019	3	$69.97
Adam Young	10/24/2019	3	$23.97
Adam Young	3/31/2020	3	$68.97
Alexandra Jenkins	7/3/2019	5	$76.95
Alexandra Jenkins	7/7/2019	1	$53.99
Alexandra Jenkins	8/12/2019	1	$2.29
Alexandra Jenkins	8/15/2019	2	$27.28
Alexandra Jenkins	9/7/2019	4	$84.96
Alexandra Jenkins	9/28/2019	1	$49.99
Alexandra Jenkins	10/26/2019	1	$120
Alexandra Jenkins	11/13/2019	3	$69.97
Alexandra Jenkins	12/8/2019	3	$63.97
Alexandra Jenkins	12/15/2019	1	$69.99
Alexandra Jenkins	1/4/2020	2	$39.98
Alexandra Jenkins	1/29/2020	2	$162.99
Alexandra Jenkins	4/2/2020	1	$34.99
Alexandra Jenkins	4/14/2020	2	$27.28
Alexandra Jenkins	5/25/2020	2	$39.98
Alexandra Jenkins	6/3/2020	3	$77.48

WINDOW() With PARTITIONBY()

Create a measure to calculate a running total of the sales amount for every partition

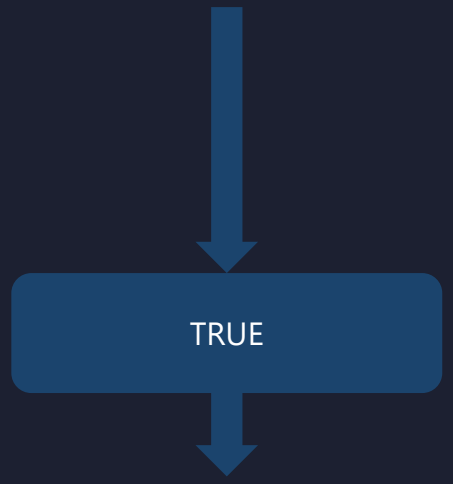
WINDOW

```
Window Sales Amount = CALCULATE([Total Sales],
                                WINDOW(
                        1,ABS,
                        -1,ABS,
                        SUMMARIZE(ALLSELECTED(Sales),Customer[Customer],'Date'[Date]),
                        ORDERBY ('Date'[Date]),
                        KEEP,
                        PARTITIONBY(Customer[Customer])
                        )
                )
```

DAX Information Functions

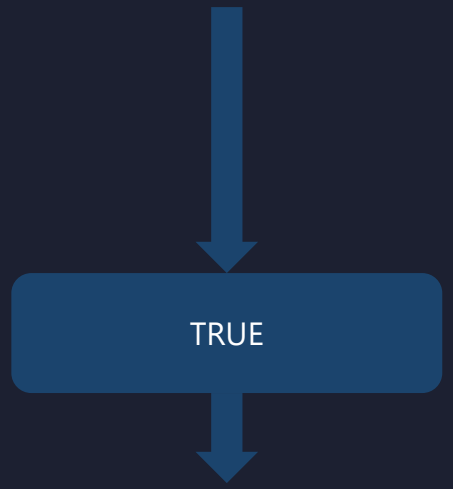
Obtain info about data type or filter context

CONTAINSSTRING()



CONTAINSSTRING(ProductColor, “Blue”)

HASONEVALUE ()



Only one value in the specified column

INFO.

Useful information about the semantic model

INFO.TABLES

INFO.COLUMNS

INFO.MEASURES

INFO.CALCULATIONGROUPS

INFO.HIERARCHIES

Calculation Groups in a Nutshell

Explicit measures

- ✓ Sales Amount
- ✓ Sales Quantity
- ✓ Discount Amount
- ✓ Return Quantity
- ✓ ….

Business Logic

- ✓ Sales Amount in the previous quarter
- ✓ Sales Quantity YoY
- ✓ Running total of Return Quantity
- ✓ ….

50 measures!

How Calculation Groups Work?

SELECTEDMEASURE()

Reference a measure that is currently in context: Sales Amount, Total Quantity...

```
YTD = CALCULATE(
            SELECTEDMEASURE(),
            DATESYTD(Date)
)
```

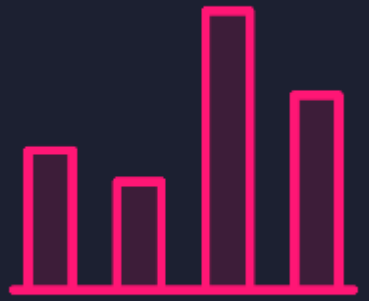
```
MTD = CALCULATE(
            SELECTEDMEASURE(),
            DATESMTD(Date)
)
```

Time Calculation	Current		YOY		YOY%	
CalendarYear	Sales	Orders	Sales	Orders	Sales	Orders
2012						
January	$495,364	252	$25,540	108	5.44%	75.00%
February	$506,994	260	$40,659	116	8.72%	80.56%
March	$373,483	212	($111,716)	62	-23.02%	41.33%
April	$400,336	219	($101,738)	62	-20.26%	39.49%
May	$358,070	207	($202,804)	33	-36.11%	18.97%
June	$555,160	319	($182,680)	88	-24.76%	38.26%
July	$444,558	246	($152,180)	58	-25.50%	30.85%
August	$523,917	294	($90,641)	101	14.75%	52.33%
September	$486,177	269	($116,906)	84	-19.38%	45.41%
October	$535,159	315	($173,049)	92	-24.43%	41.68%
November	$537,956	324	($122,590)	116	-18.56%	55.77%
December	$624,502	487	($44,929)	265	-6.71%	119.37%
2013						
January	$857,690	1662	$362,326	1410	73.14%	559.52%
February	$771,349	3453	$264,355	3193	52.14%	1228.08%
March	$1,049,907	4087	$676,424	3875	181.11%	1827.83%
April	$1,046,023	3979	$645,687	3760	161.29%	1716.89%
May	$1,284,593	4399	$925,715	4192	257.95%	2025.12%
June	$1,643,178	5025	$1,088,018	4787	195.98%	1480.19%
July	$1,371,676	4671	$927,118	4425	208.55%	1798.78%
August	$1,551,066	4865	$1,027,148	4571	196.05%	1554.76%
September	$1,447,496	4616	$961,318	4347	197.73%	1615.99%
October	$1,673,293	5300	$1,138,134	4987	212.67%	1593.29%
November	$1,780,920	5224	$1,242,965	4900	231.05%	1512.35%
December	$1,874,360	6231	$1,249,858	5744	200.14%	1179.47%

What can you do with Field Parameters?

Without a single line of DAX!

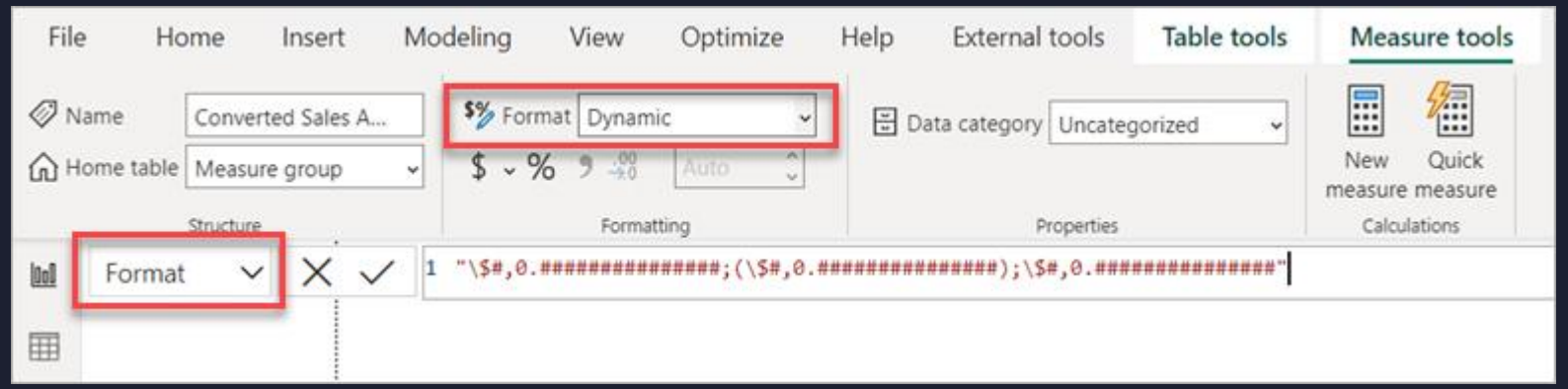
Dynamically change the attribute



Dynamically change the metrics

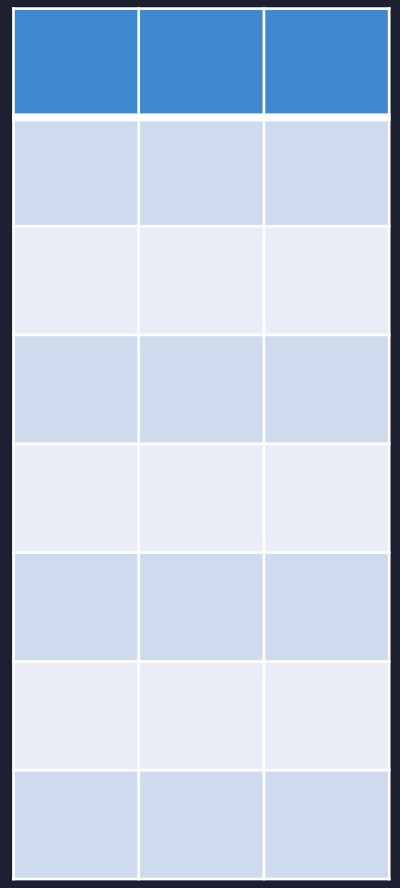
Dynamic Format Strings

- Determine how measures appear in the visual
- Conditionally applying a format string with a separate DAX expression
- Overcomes the limitations of the FORMAT() function

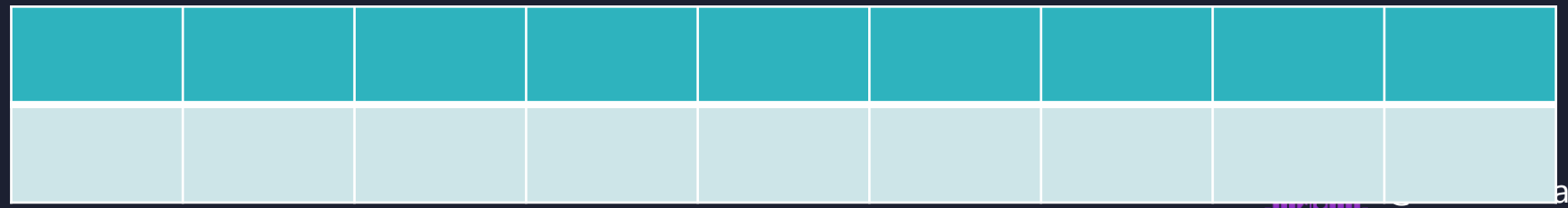


Large Semantic Models in Power BI

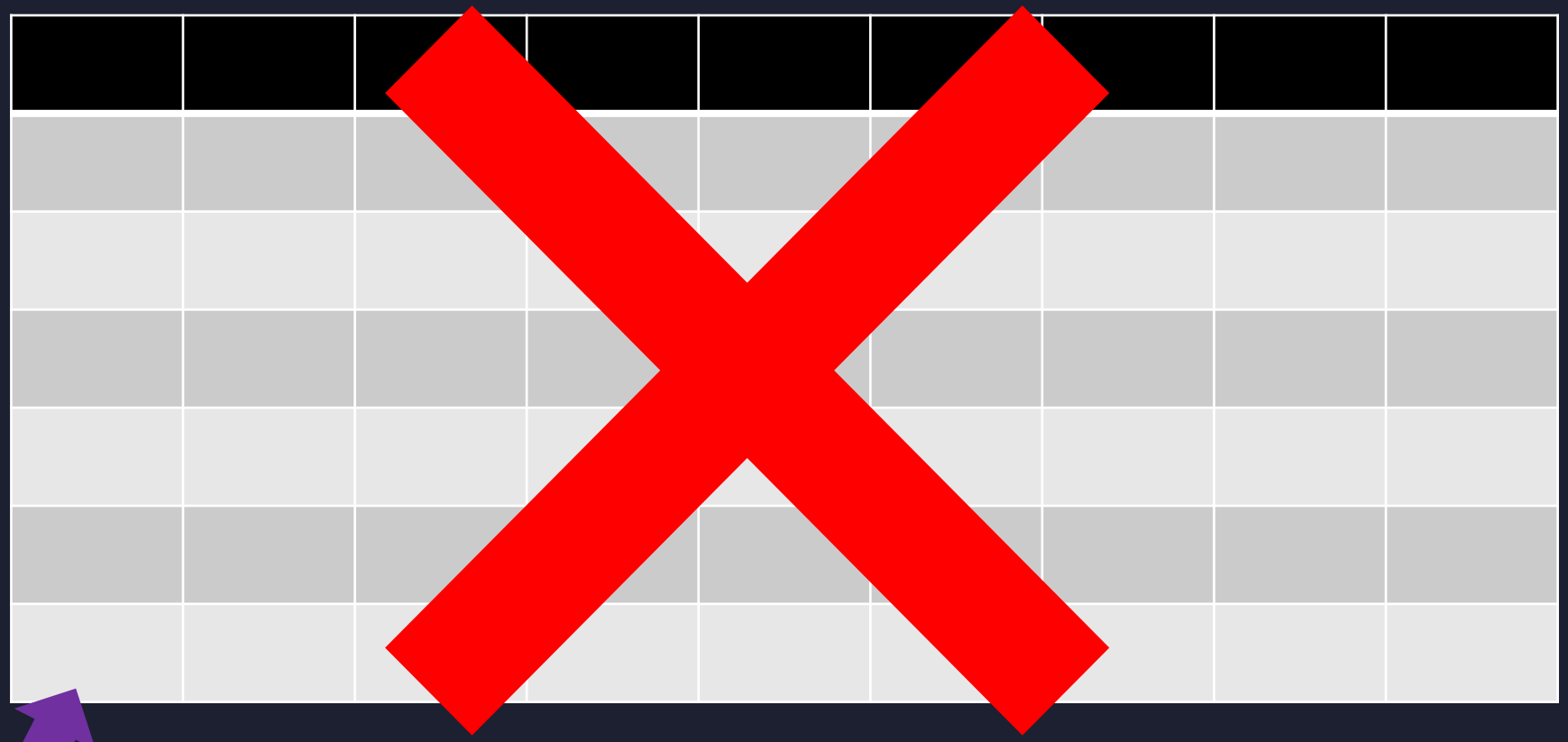
I'm a tall table with many rows and less columns



I'm a wide table with many columns and less rows



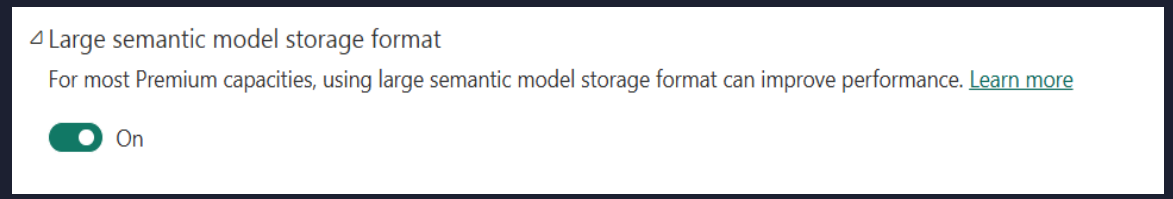
Large Semantic Models in Power BI



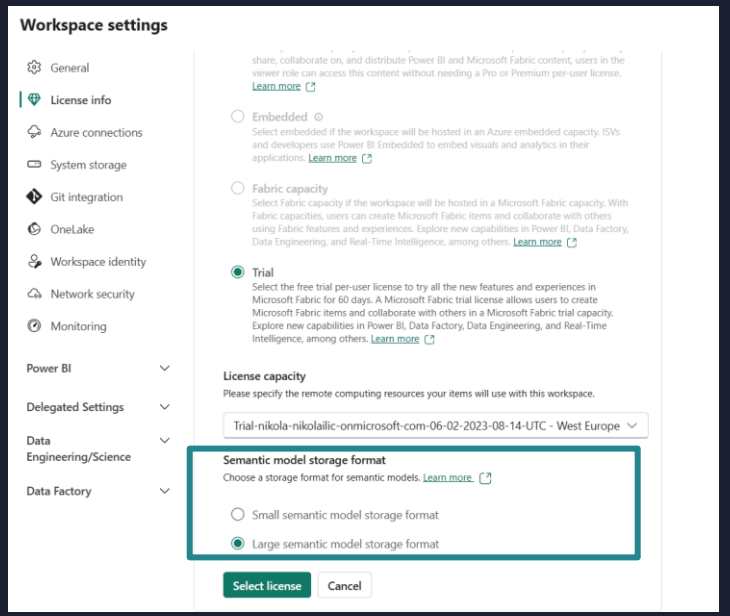
I'm a tall and wide table, with many rows and many columns

Enable Large Semantic Model Storage Format

- ✓ Beyond the default model limit
- ✓ Limit = max capacity size (i.e. F64 = 25 GB, F128 = 50 GB…F2 = 3 GB)
- ✓ Limit is *per model*, not the sum of all models
- ✓ Improves XMLA write operations performance



Semantic model settings



Workspace settings

Semantic model eviction and On-demand load

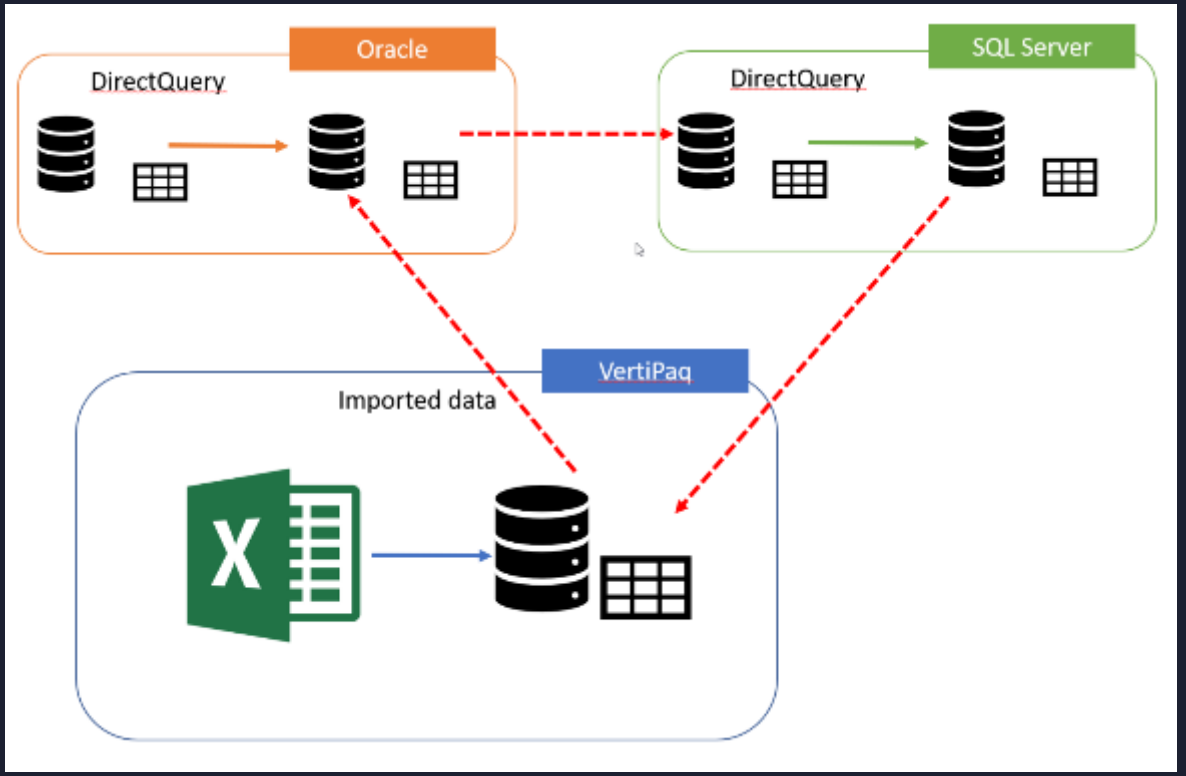
@DataMozart

Composite Models

- ✓ Combine 2 or more DirectQuery sources
- ✓ Combine 1 or more DirectQuery sources AND Import mode

Composite Models

- ✓ All Imported data = 1 source
- ✓ Regular vs Limited relationships

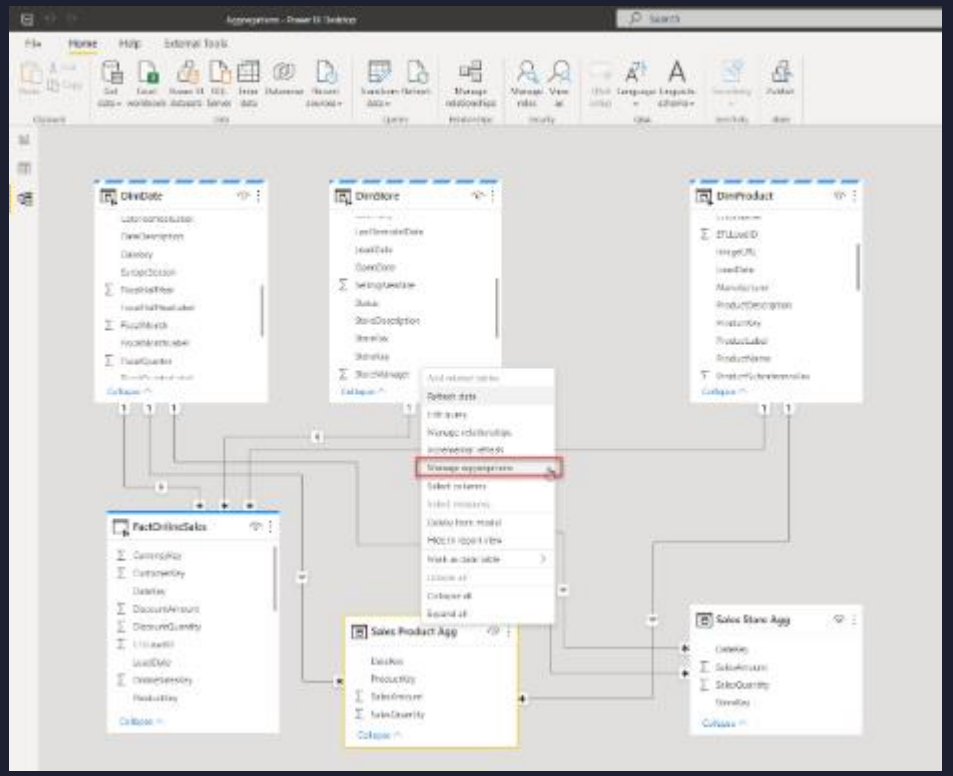


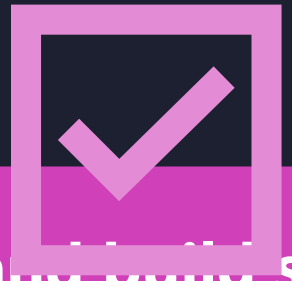
Aggregations

- ✓ User-defined vs Automatic
- ✓ Reducing the amount of data
- ✓ Make Power BI "aware" of aggregated tables!

Aggregations

- ✓ Original table = DirectQuery
- ✓ Dim tables = Dual
- ✓ Agg tables = Import





Design and build semantic models

- Choose a storage mode
- Implement star schema for semantic model
- Implement relationships (bridge tables and many-to-many)
- DAX calculations with variables and functions (iterator, filtering, window, information)
- Implement calculation groups, dynamic format strings, and field parameters
- Identify and configure large semantic model storage format
- Design composite models

Optimize enterprise-scale semantic models

- Implement performance improvements in queries and visuals
- Improve DAX performance
- Configure Direct Lake (default fallback and refresh behavior)
- Implement incremental refresh

@DataMozart

Optimize enterprise-scale semantic models

- Implement performance improvements in queries and visuals
- Improve DAX performance
- Configure Direct Lake (default fallback and refresh behavior)
- Implement incremental refresh

@DataMozart

Who Complains about performance?

DBA

Background

- Data model size
- Data refresh process

Consumer

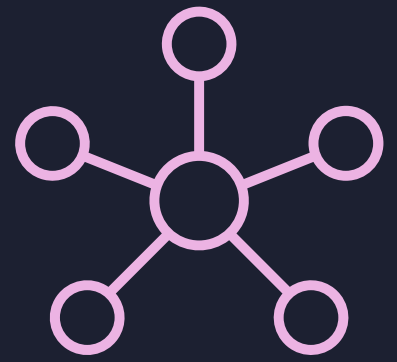
Front End

- DAX measures
- Visuals rendering

Performance Analyzer captures key metrics of the report performance

@DataMozart

Performance Analyzer Results



DAX query	Visual display	Other
Time needed to return query results	Time elapsed for visual rendering	Waiting for other processes/queries

Fewer visuals on the page = better performance!

Bottlenecks

Visuals

Speeds up the report and removes the clutter

```
1  Sales Amount YTD NO TI =
2                  var EoMonthCurrDate = EOMONTH(CALCULATE(MAX('Flat table'[Sales Date])),0)
3                  var _Year = YEAR(MAX('Flat table'[Sales Date]))
4                  var _theDates = DATESBETWEEN('Flat table'[Sales Date], DATE(_Year,1,1), EoMonthCurrDate)
5                  return
6                  IF( ISBLANK( CALCULATE( SUM( 'Flat table'[Sales Amount] ) ) )
7      , BLANK()
8      , CALCULATE(
9          SUM('Flat table'[Sales Amount] )
10         , ALLSELECTED( 'Flat table')
11         , _theDates
12     )
13 )
14
```

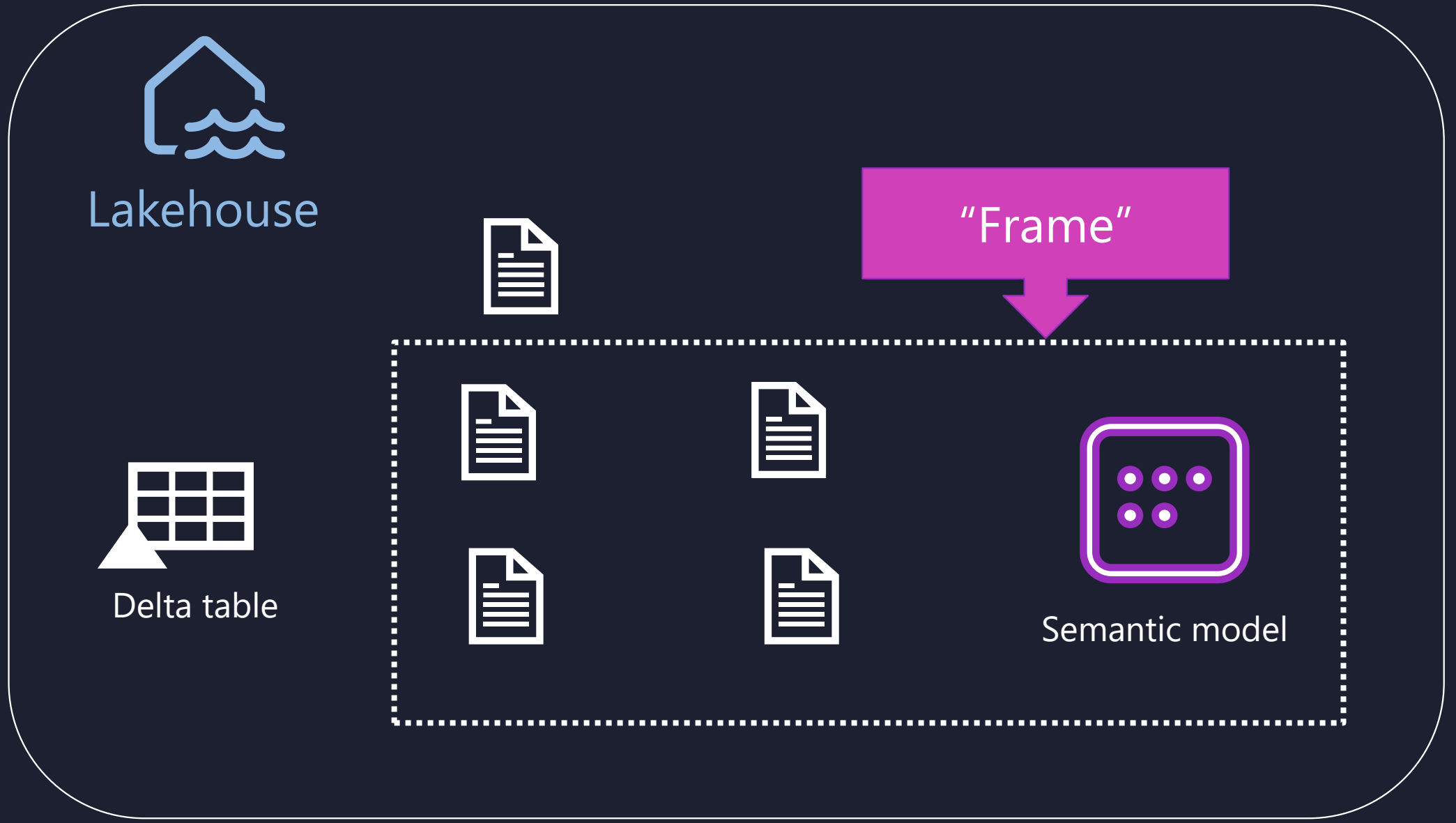
DAX queries

Learn and understand DAX!

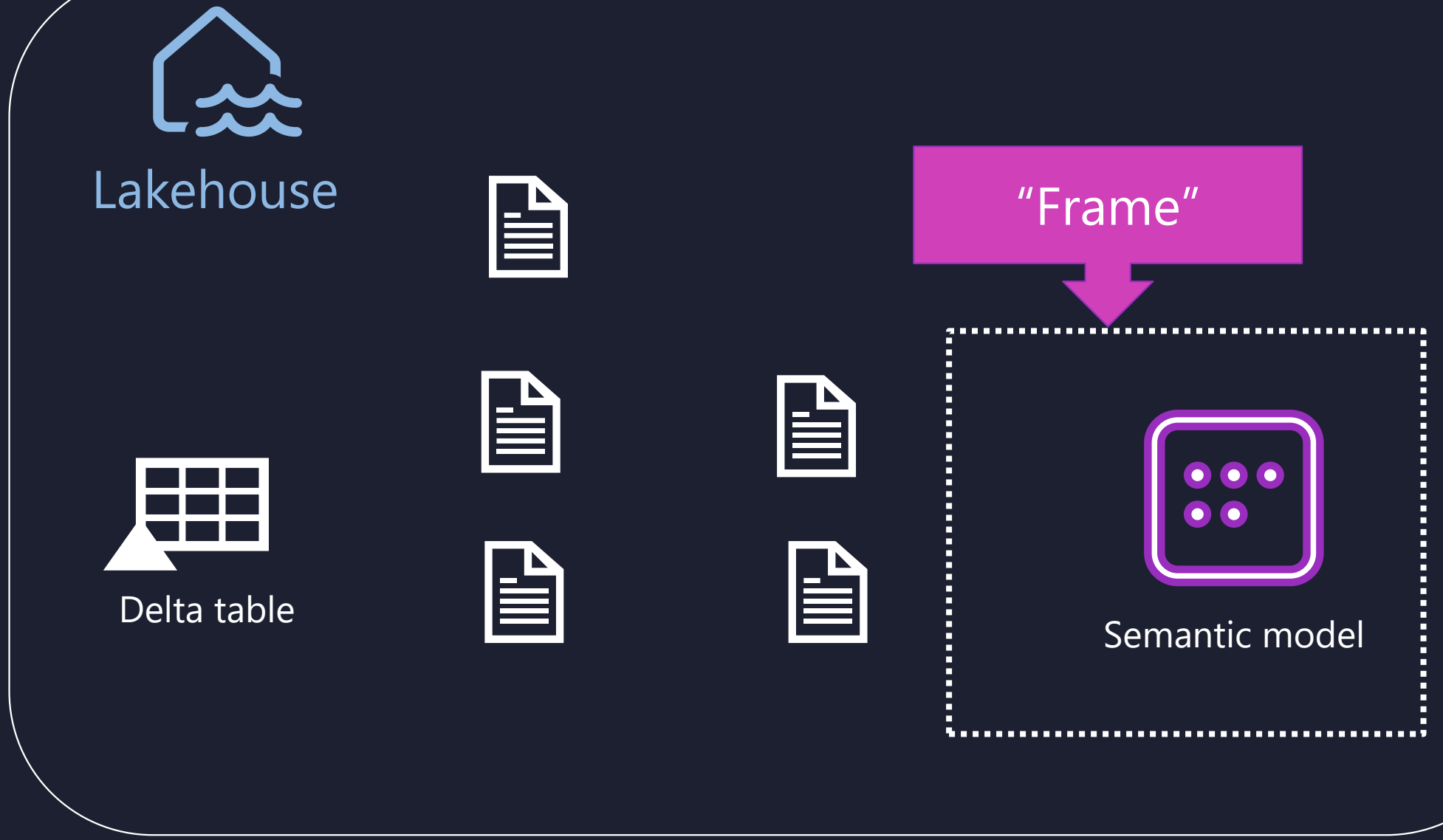
Direct Lake – Data Refresh

- Called internally as “framing”
- A metadata operation to read the latest parquet schema
- It doesn’t copy (or refresh) any actual data!

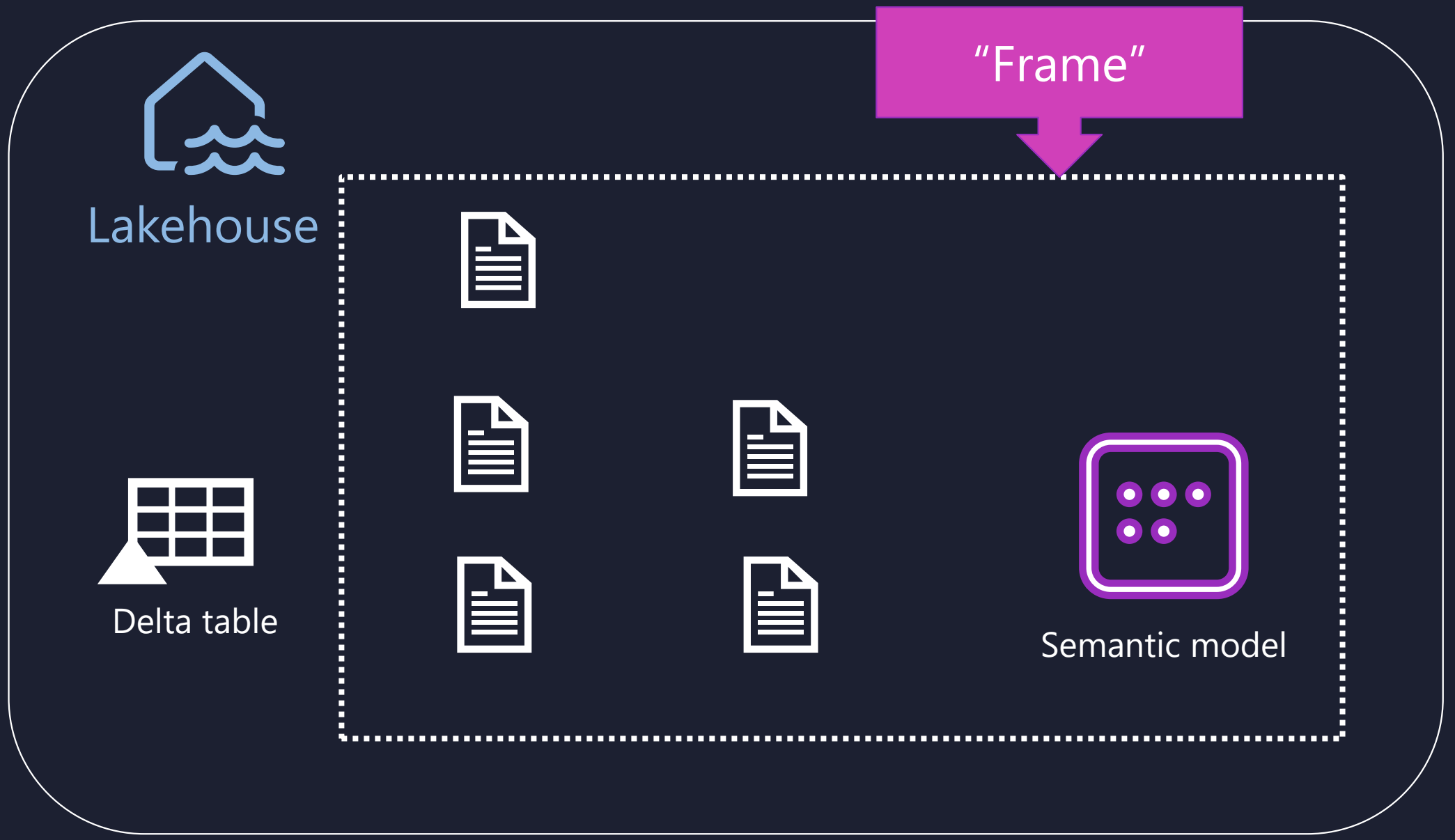
Direct Lake Refresh (AKA “Framing”)



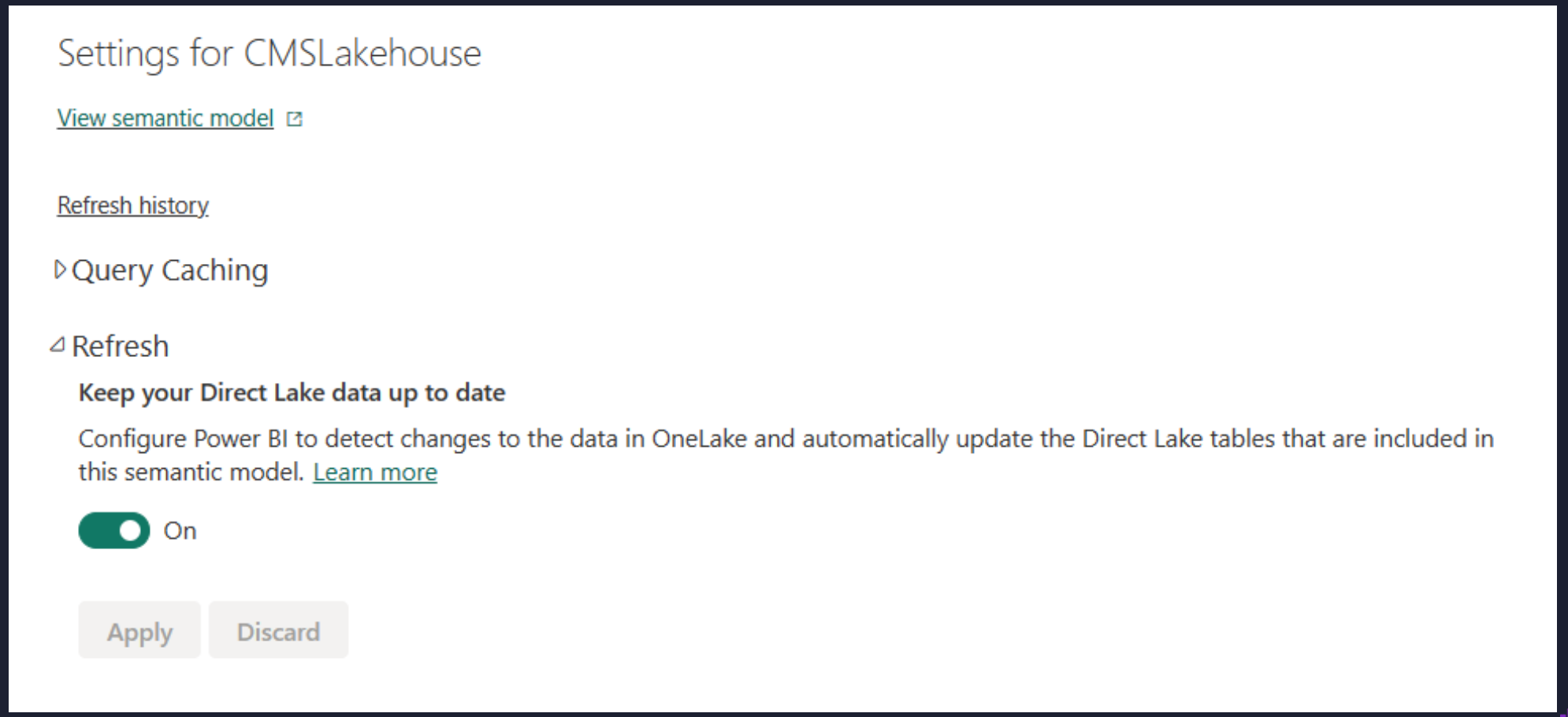
Direct Lake Refresh (AKA “Framing”)



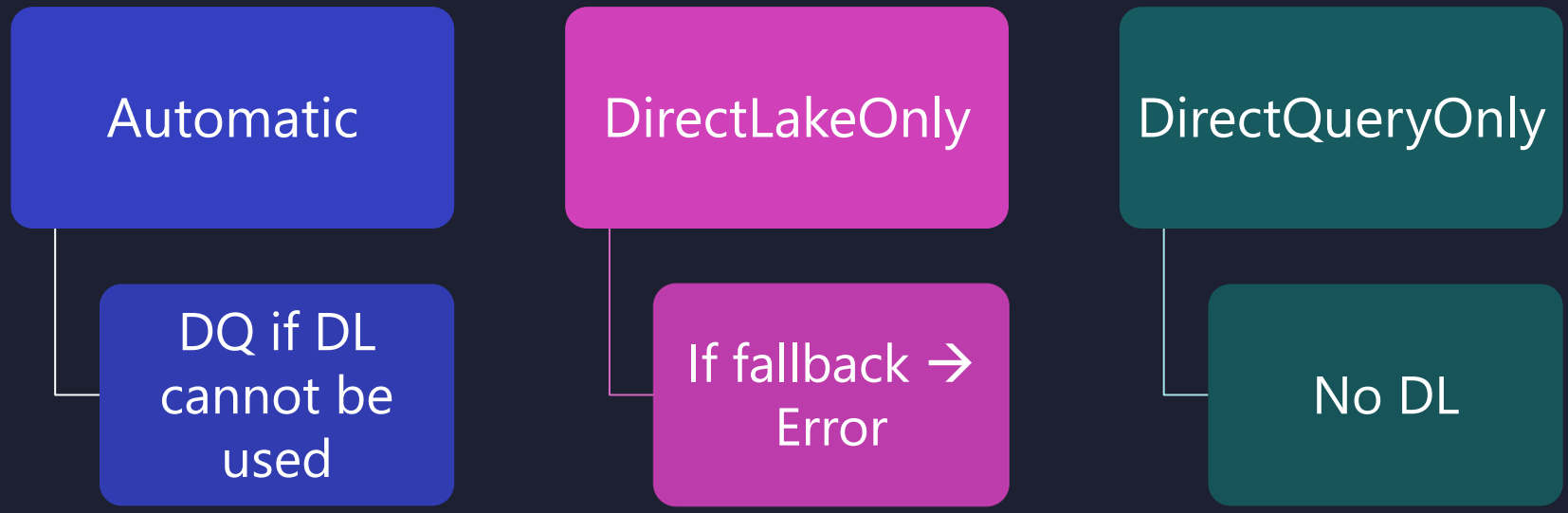
Direct Lake Refresh (AKA “Framing”)



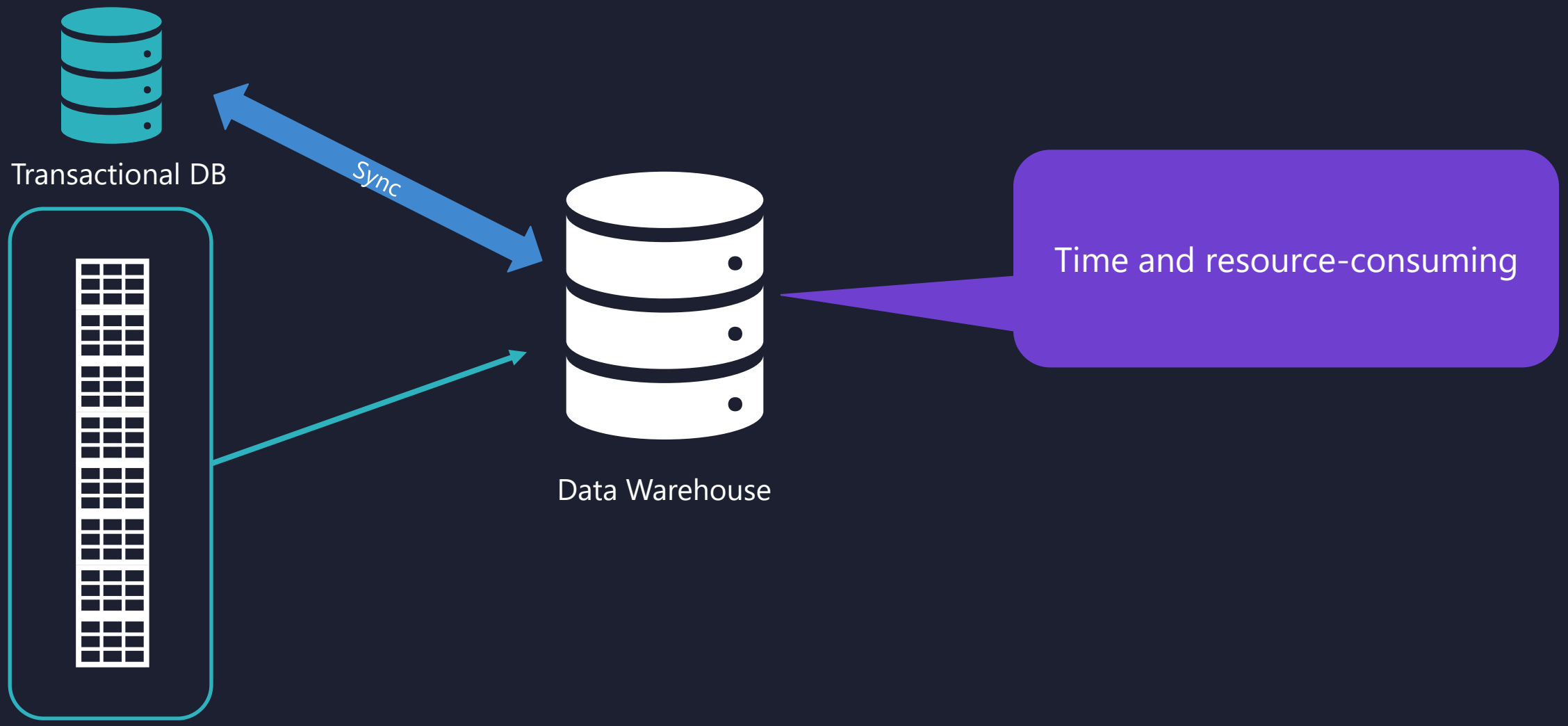
Refresh option for semantic models



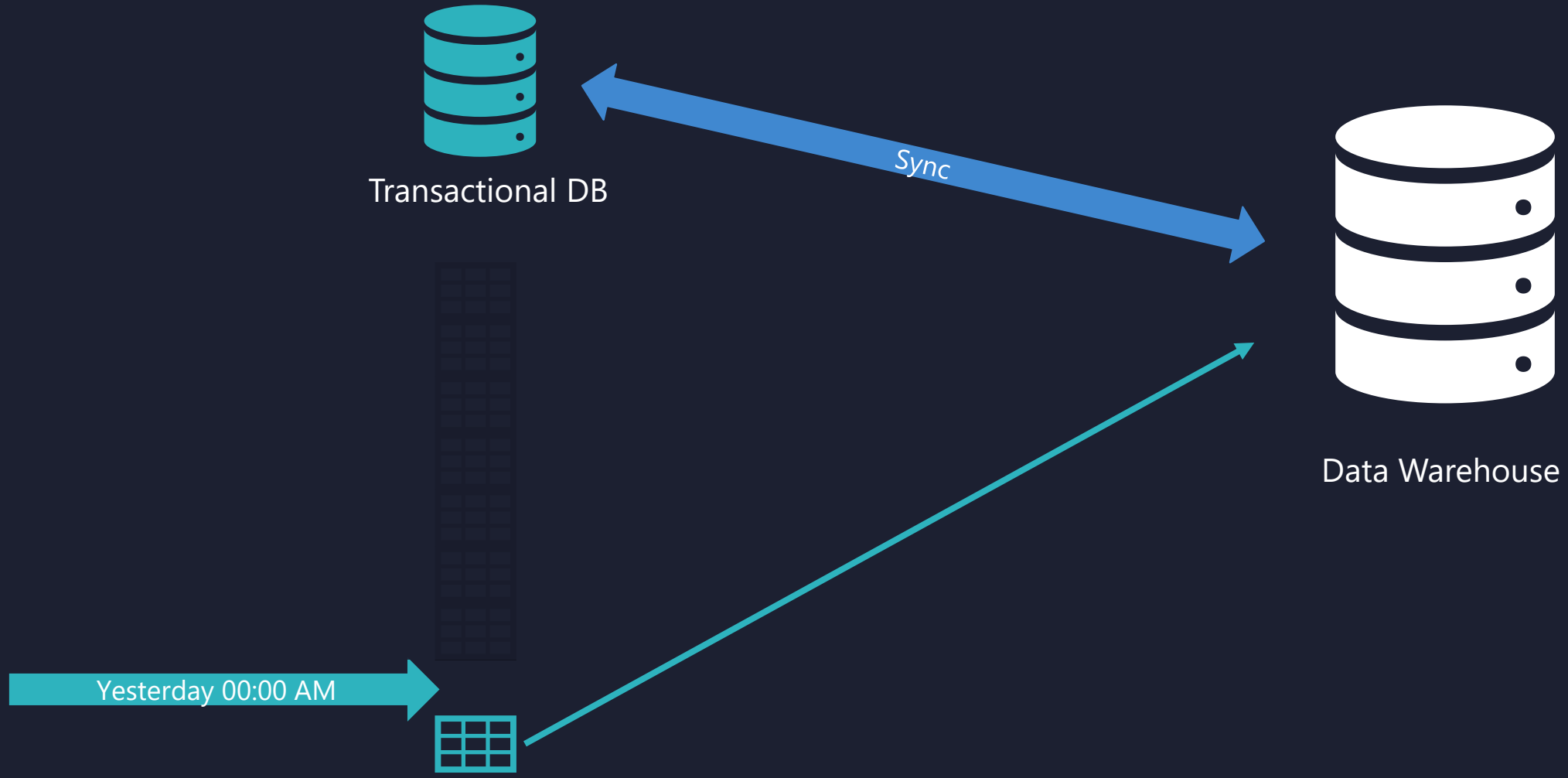
DirectLakeBehavior Property



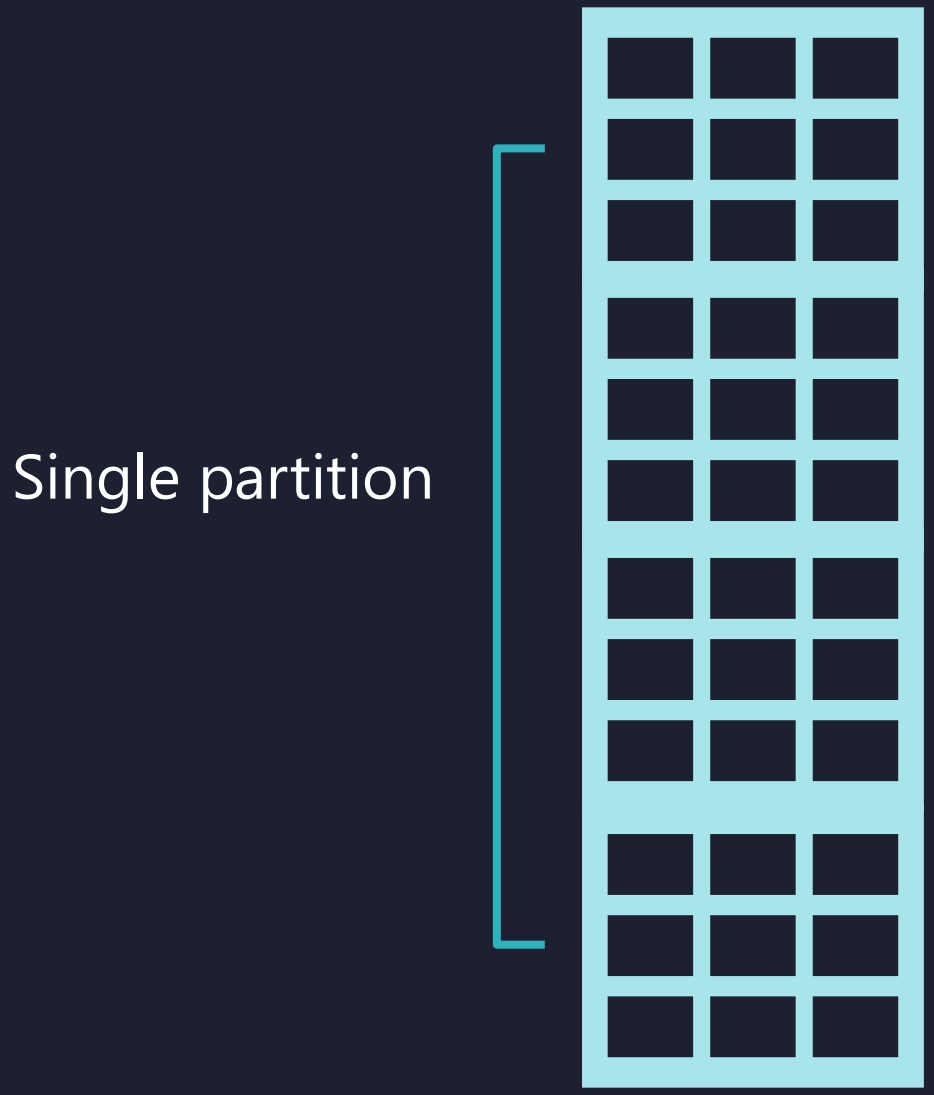
Why Incremental Refresh?



Why Incremental Refresh?



Incremental Refresh in Power BI

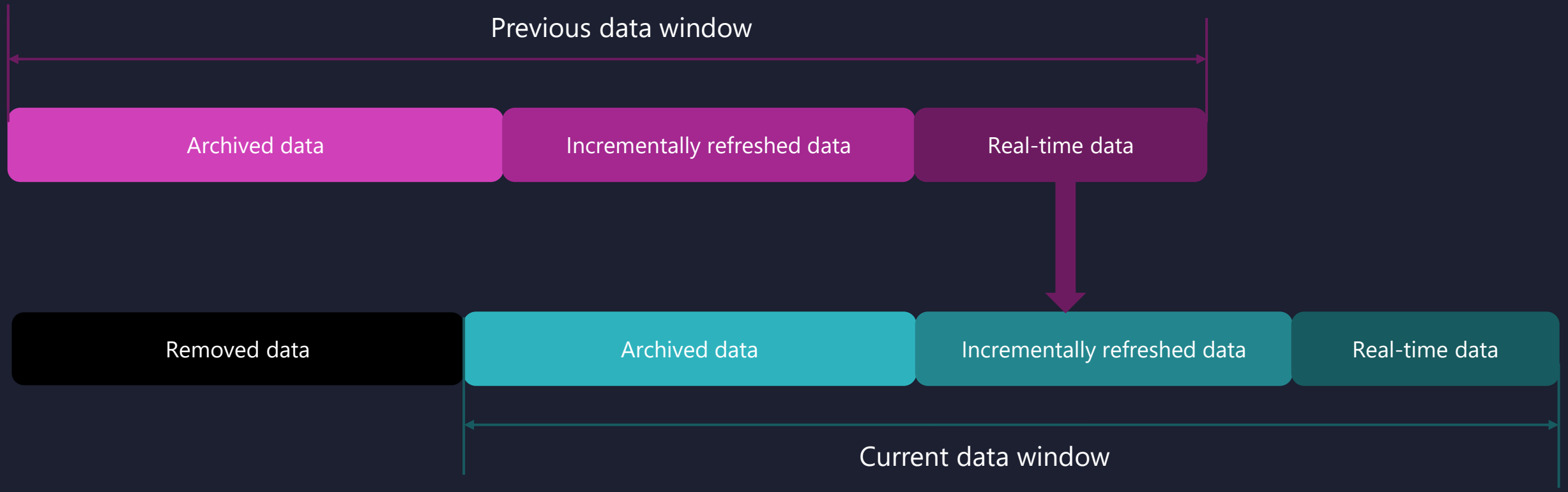


Incremental Refresh in Power BI

No changing rows partition

Frequently changing rows partition

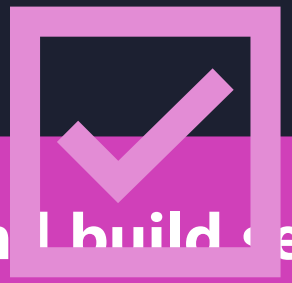
Incremental Refresh Workflow



"Rolling Window Pattern"

Incremental Refresh Prerequisites

- ✓ Date column
  - Date/Time or Integer type
- ✓ Query folding
  - Date range parameters -> WHERE clause
- ✓ Single data source



Design and build semantic models

- Choose a storage mode
- Implement star schema for semantic model
- Implement relationships (bridge tables and many-to-many)
- DAX calculations with variables and functions (iterator, filtering, window, information)
- Implement calculation groups, dynamic format strings, and field parameters
- Identify and configure large semantic model storage format
- Design composite models

Optimize enterprise-scale semantic models

- Implement performance improvements in queries and visuals
- Improve DAX performance
- Configure Direct Lake (default fallback and refresh behavior)
- Implement incremental refresh

@DataMozart

Let’s Play!

Quiz

Kahoot.it

Thanks to…

Ricardo Rincón

Practical Exam Tips

- ✓ Make sure to prepare the room (for online takers)
- ✓ You can use Microsoft Learn during the exam☺…
- ✓ …but, don’t let this waste too much of your time!
- ✓ Focus on specific functions (DAX, T-SQL, KQL) and their ORDER of execution
- ✓ Understand WHEN to complete a specific task in a certain way (i.e. ingest data with pipeline vs notebook vs Dataflow Gen2 vs Shortcuts)

Where to go from here?

- Set up your Fabric Trial account and try it for 60 days for free:
  - Fabric Trial Capacity
- Microsoft Fabric official documentation
- Microsoft Fabric – The Ultimate Guide (8-hour live training with hands-on labs)
  - February 3rd/4th, 5 PM CET/11 AM EST/8 AM PST

Where to go from here?

Free learning resources

Data-Mozart.com

Serverlesssql.com

kevinrchant.com

Learn MS Fabric YT channel

@DataMozart

Thank you

Nikola Ilic

@DataMozart

www.data-mozart.com

www.learn.data-mozart.com