

# *DP-600 Fabric Analytics Engineer*



Nikola Ilic

Data Mozart, Microsoft Data Platform MVP



# Nikola Ilic

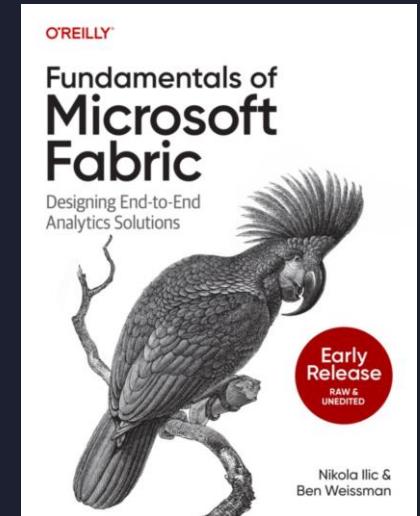
*Consultant & Trainer*



[data-mozart.com](http://data-mozart.com)

[learn.data-mozart.com](http://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!



@DataMozart



# Introduction and Overview



@DataMozart



# 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



@DataMozart



# 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?



@DataMozart



# What is a Microsoft Fabric?



Azure Synapse Analytics



Azure Data Factory



Power BI



@DataMozart



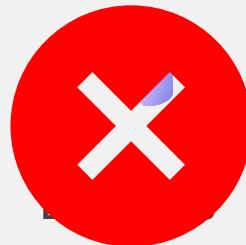
# Players in Microsoft Fabric



Data  
Factory



Real-Time  
Intelligence



Analytics



Analytics



Power BI



Analytics



Fabric



OneLake



Purview



@DataMozart



# Microsoft Fabric Terminology

**Tenant** - organization's fabric instance

**Capacity** – compute power

**Domain** – logical grouping of workspaces

**Workspace** - collaborative container

**Items** - units of experience



@DataMozart



# 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





# 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



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!



@DataMozart



# 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. <sup>1</sup>	✓	✓		
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 <sup>2</sup> with T-SQL through TDS endpoint.	✓	✓	✓	✓
Read Lakehouse and Data warehouse data and shortcuts <sup>2</sup> 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. <sup>3</sup>	✓	✓	✓	
Modify gateway connection settings. <sup>3</sup>	✓	✓	✓	

<sup>1</sup> Contributors and Viewers can also share items in a workspace, if they have Reshare permissions.

<sup>2</sup> Additional permissions are needed to read data from shortcut destination. Learn more about [shortcut security model](#).

<sup>3</sup> 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

Grant people access X

DP600LH

People you share this Lakehouse with can open it and its SQL endpoint and read the default dataset. To allow them to read directly in the Lakehouse, grant additional permissions.

Enter a name or email address

Additional permissions

Read all SQL endpoint data ⓘ

Read all Apache Spark ⓘ

Build reports on the default semantic model

Notification Options

Notify recipients by email

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 <i>Share</i> , <i>Edit</i> , and <i>Read</i> permissions, they can at most grant <i>Share</i> , <i>Edit</i> , and <i>Read</i> 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.

- **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**

[Item permission model](#)



@DataMozart



# Row-level Security

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

```
-- 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
```

```
-- 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



@DataMozart



# 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



@DataMozart



# 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

New role (preview)

DP600LH

Grant this role Read permissions to the selected data. [Learn more](#)

Assign role

Role name \*

ReadAW

Included folders

All folders

Selected folders

\Tables Folder

- Contoso\_DimCurrency
- Contoso\_DimCustomer
- Contoso\_DimDate
- Contoso\_DimProduct
- Contoso\_FactOnlineSales
- aw\_dimcurrency
- aw\_dimcustomer
- aw\_dimdate
- aw\_dimproduct
- aw\_dimproductcategory
- aw\_dimproductsubcategory
- aw\_dimsalesterritory
- aw\_factinternetsales

\Files Folder



@DataMozart



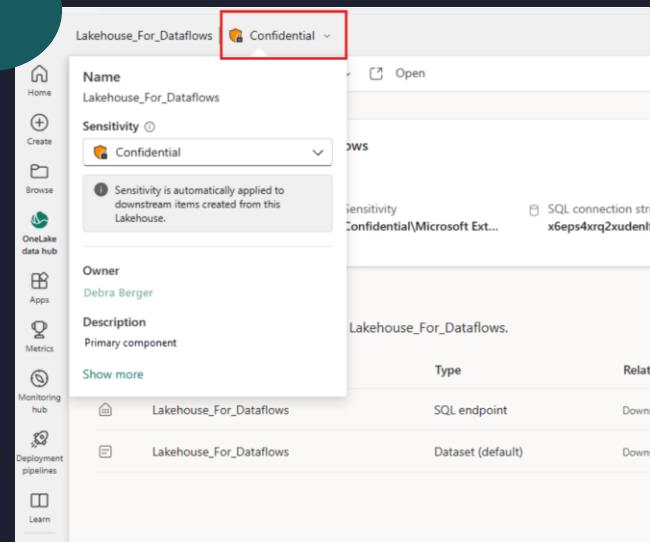
# Sensitivity Labels

First created in Purview

## Microsoft Purview Information Protection

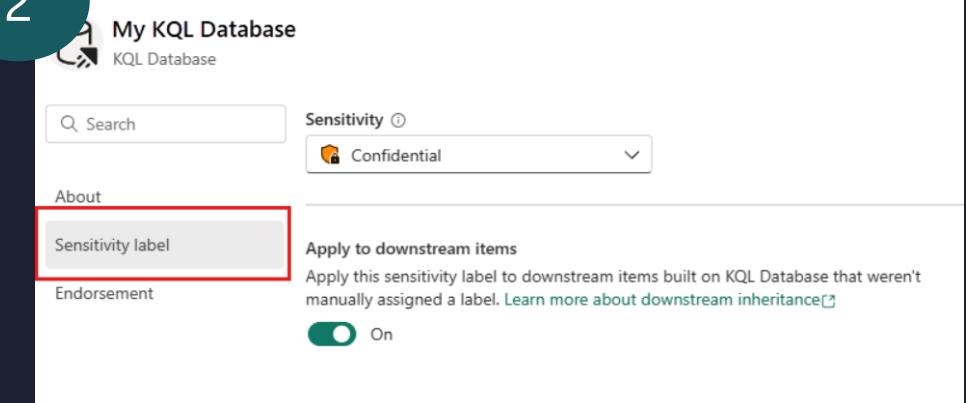
- Meet governance and compliance requirements
- Only authorized people can access the data
- 2 ways to apply

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

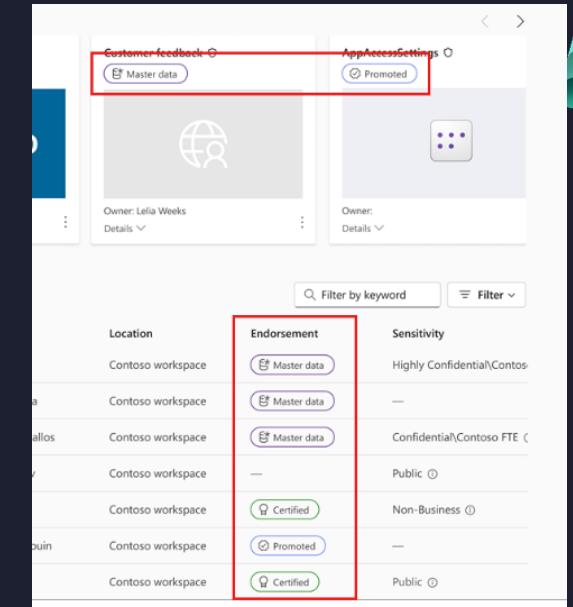
## Certified

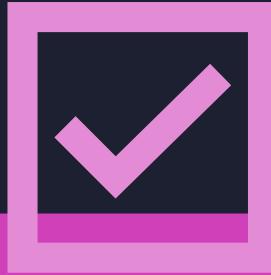
- ✓ Authorized reviewer
- ✓ Item meets org's quality standards
- ✓ All items except dashboards
- ✓ Any user can **request**, only users specified by Fabric Admin can certify items

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

The screenshot shows a Microsoft Fabric workspace titled "App Business Reports". A table lists a single item: "Business Reports" with a "Report" type. The "Git status" column shows "Uncommitted". In the top right, there are buttons for "Create deployment pipeline" and "Source control". A pink arrow points to the "Source control" button, and another pink arrow points to the "Uncommitted" status in the table.

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



@DataMozart



# 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



@DataMozart



# Deployment Pipelines

## Development

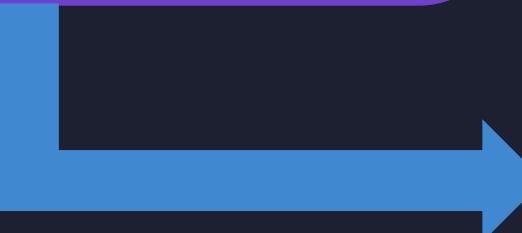
- ✓ Design, review, play around
- ✓ Start small



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

## Workflow

A workflow view is helpful to review.



- ✓ 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



@DataMozart



# Deployment Pipelines Workflow

The screenshot shows the Synapse Data Engineering Deployment pipelines interface. The top navigation bar includes 'Synapse Data Engineering' and 'Deployment pipelines'. The left sidebar lists various workspace options, with 'Redesign demo...' currently selected. The main area displays a deployment pipeline named 'Redesign demo pipeline' with three stages: Dev, Test, and UAT.

- Dev Stage:** Status is green, labeled 'Successful deployment'. Deployment date: 01/09/24, 5:13 PM.
- Test Stage:** Status is green, labeled 'Successful deployment'. Deployment date: 01/09/24, 6:38 PM.
- UAT Stage:** Status is green, labeled 'Successful deployment'. Deployment date: 01/15/24, 11:28 AM. A red notification bubble indicates 5 pending tasks.

The pipeline interface includes a 'Deployment source stage' dropdown set to 'Dev' and a 'Deploy' button. Below the stages, a table lists deployment details for each stage:

Name	Type	Status vs. Source	Source item
Calculated entitie	Dataflow	Same as source	Calculated entitie
SqlAzureDataflowApp1	Dataflow	Same as source	SqlAzureDataflow...
ParamsUsageV3	Semantic model	Same as source	ParamsUsageV3
V3-AsAzure-LC-VsabMsit_NoRLS	Semantic model	Same as source	V3-AsAzure-LC-Vs...
V3-ASonPrem-LiveConnection	Semantic model	Same as source	V3-ASonPrem-Live...
V3-SqlAzure-DQ	Semantic model	Same as source	V3-SqlAzure-DQ



@DataMozart



# Deployment Pipelines – Item Pairing

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

Test		Deploy from	Development	Deploy	Select related	Compare	Deployment history	Filter by keyword	Filter
	Selected stage item	Type	Compared to source	Source stage item					
	SqlAzureDataflowApp1	Dataflow	Same as source	SqlAzureDataflowApp1					
	V3-SqlAzure-Cached	Report	Same as source	V3-SqlAzure-Cached					
	V3-SqlAzure-Cached	Semantic model	Same as source	V3-SqlAzure-Cached					
	V3-SqlAzure-DQ.pbix	Dashboard	Same as source	V3-SqlAzure-DQ.pbix					
—		dataflow	Only in source	NewEmail2					

- 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: <i>PBI Report</i> Type: <i>Report</i>	Name: <i>PBI Report</i> Type: <i>Report</i>	<input checked="" type="checkbox"/> Pairing occurs
2	Name: <i>PBI Report</i> Type: <i>Report</i>	Name: <i>PBI Report</i> Type: <i>Report</i>	<input checked="" type="checkbox"/> Pairing doesn't occur (duplicates). <input checked="" type="checkbox"/> Deployment fails.
		Name: <i>PBI Report</i> Type: <i>Report</i>	<input checked="" type="checkbox"/> Pairing doesn't occur (duplicates). <input checked="" type="checkbox"/> Deployment fails.
3	Name: <i>PBI Report</i> Type: <i>Report</i> <i>Folder A</i>	Name: <i>PBI Report</i> Type: <i>Report</i> <i>Folder B</i>	<input checked="" type="checkbox"/> Deployment succeeds but <input checked="" type="checkbox"/> this report isn't paired with dev
		Name: <i>PBI Report</i> Type: <i>Report</i> <i>Folder A</i>	<input checked="" type="checkbox"/> Pairing occurs using folder as a tie breaker for duplicates
		Name: <i>PBI Report</i> Type: <i>Report</i> <i>No folder</i>	<input checked="" type="checkbox"/> Deployment succeeds but <input checked="" type="checkbox"/> 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.



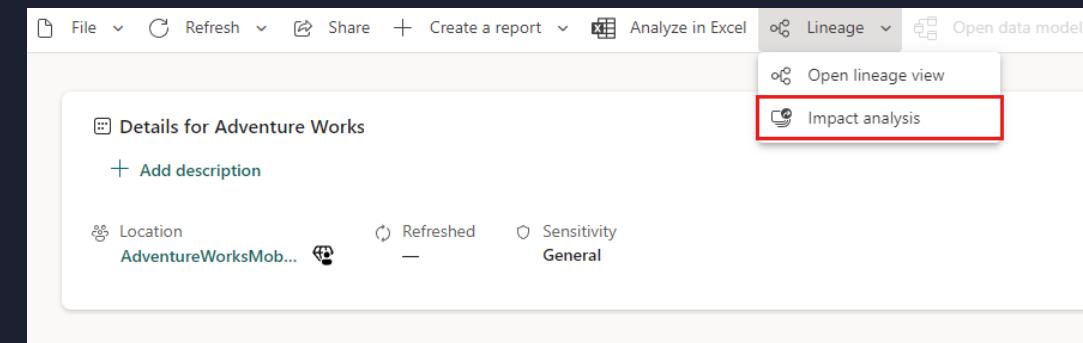
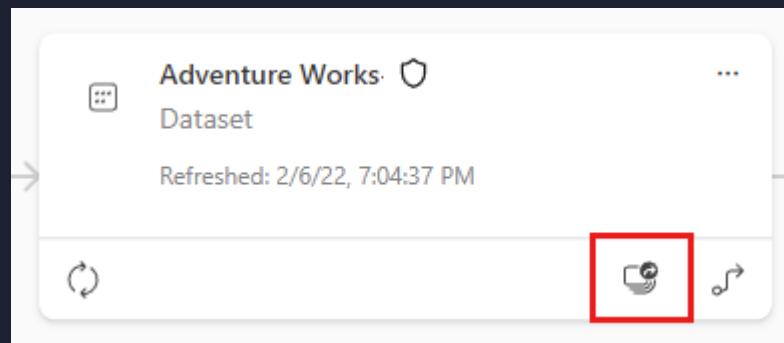


# Perform Impact Analysis

➤ Potential impact of changes to Fabric item

➤ Lineage view in the workspace

➤ Item details page





# Perform Impact Analysis

By item type

Impacted by this Dataset  
Marketing Model

All Items All downstream items

34 Impacted child items 2 Workspaces

Browse by item type

Report 5 Marketing Model Sales Report 3 Sales Leads Sales Report 2 Marketing Promotions

Dashboard 3 Marketing Dashboard Sales Dashboard Sales Team Dashboard

Making changes? Notify the contact lists of impacted items of Marketing Model

Notify contacts

Impacted by this Dataset  
Marketing Model

Child Items All downstream items

34 Impacted child items 2 Workspaces

Browse by workspace

Sales and Marketing Group Marketing Model Sales Team Dashboard

Finance Corporate Sales Report 3 Sales Leads Sales Report 2 Marketing Promotions Marketing Dashboard Sales Dashboard

Making changes? Notify the contact lists of impacted items of Marketing Model

Notify contacts

By workspace

Notify contacts



@DataMozart



# Semantic Model Deployment via XMLA

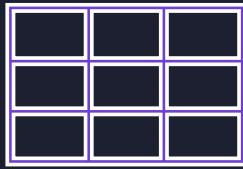
- Allows “communication” with a semantic model
- Read-only by default
- Read-write enables more options for model management, configuration, advanced modeling...



@DataMozart



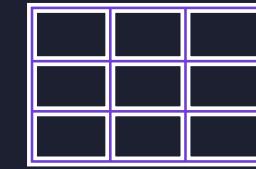
# Creating Reusable Power BI Assets



Product



Sales



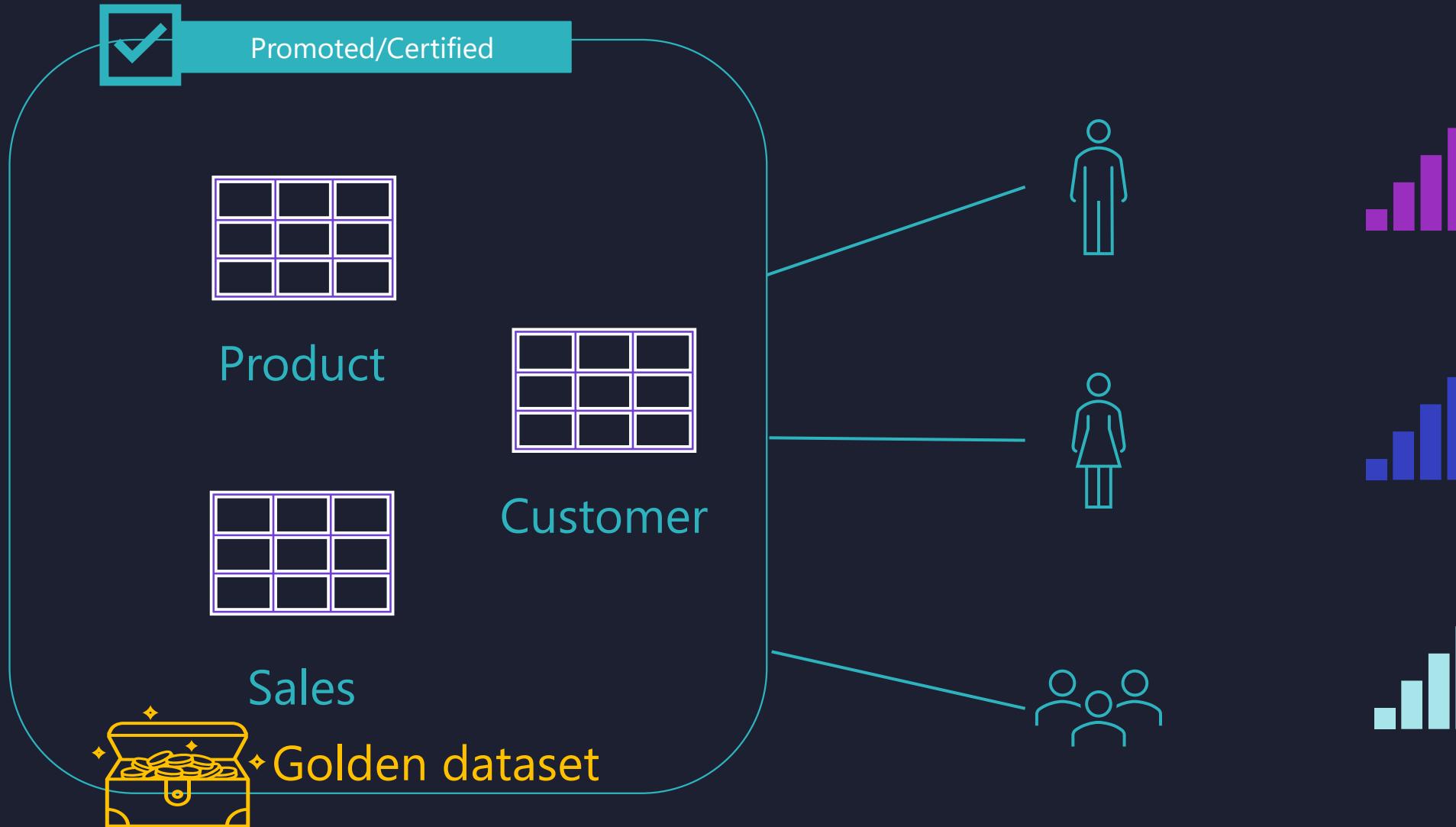
Customer



@DataMozart



# 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

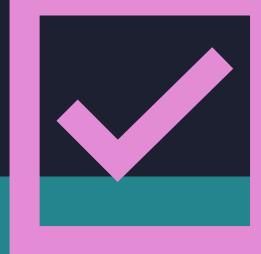
- Multiple users leverage models that were already created
- Various permissions: Can modify, share onwards...





## 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%



@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



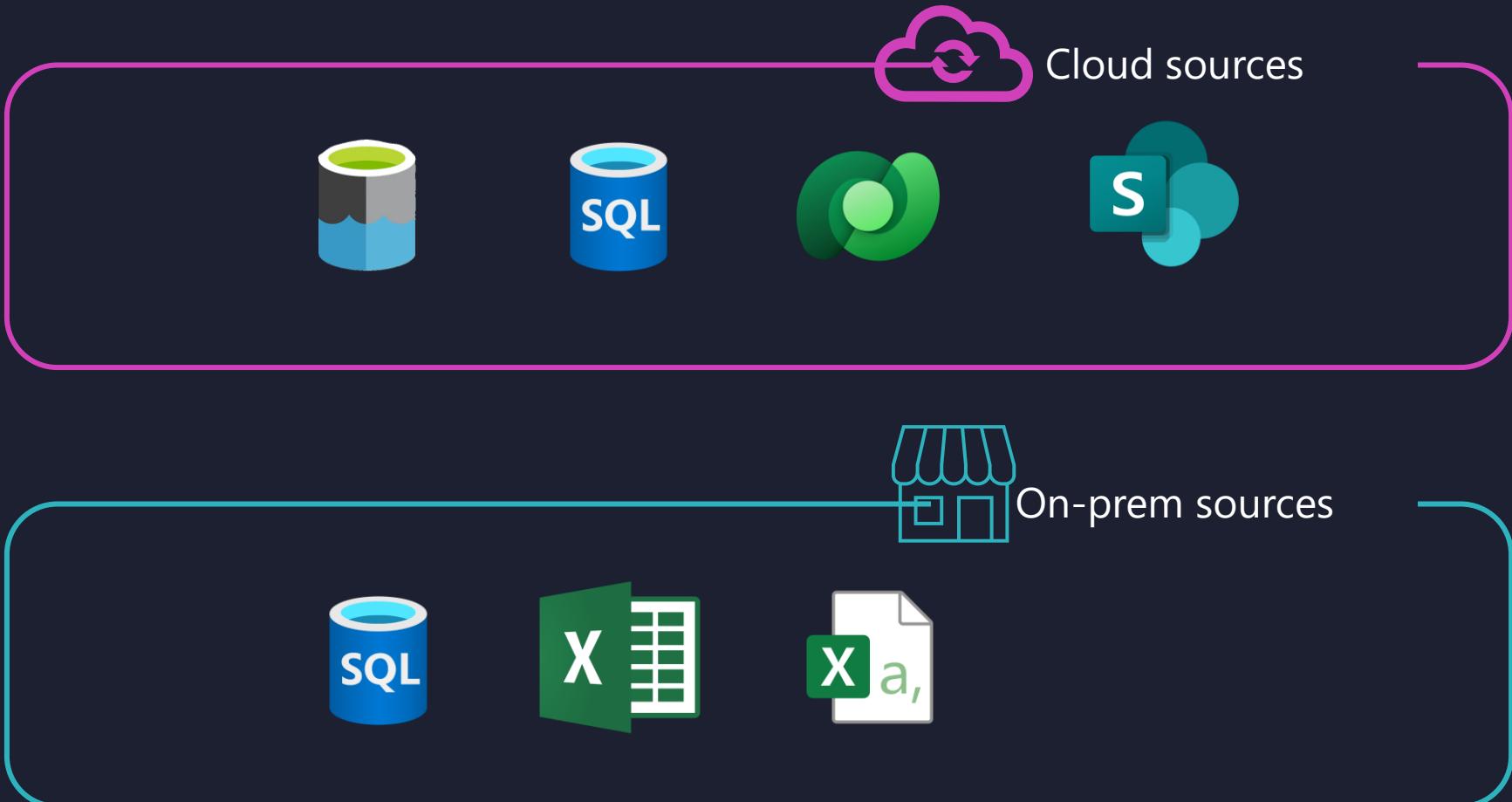


## Get data

- 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



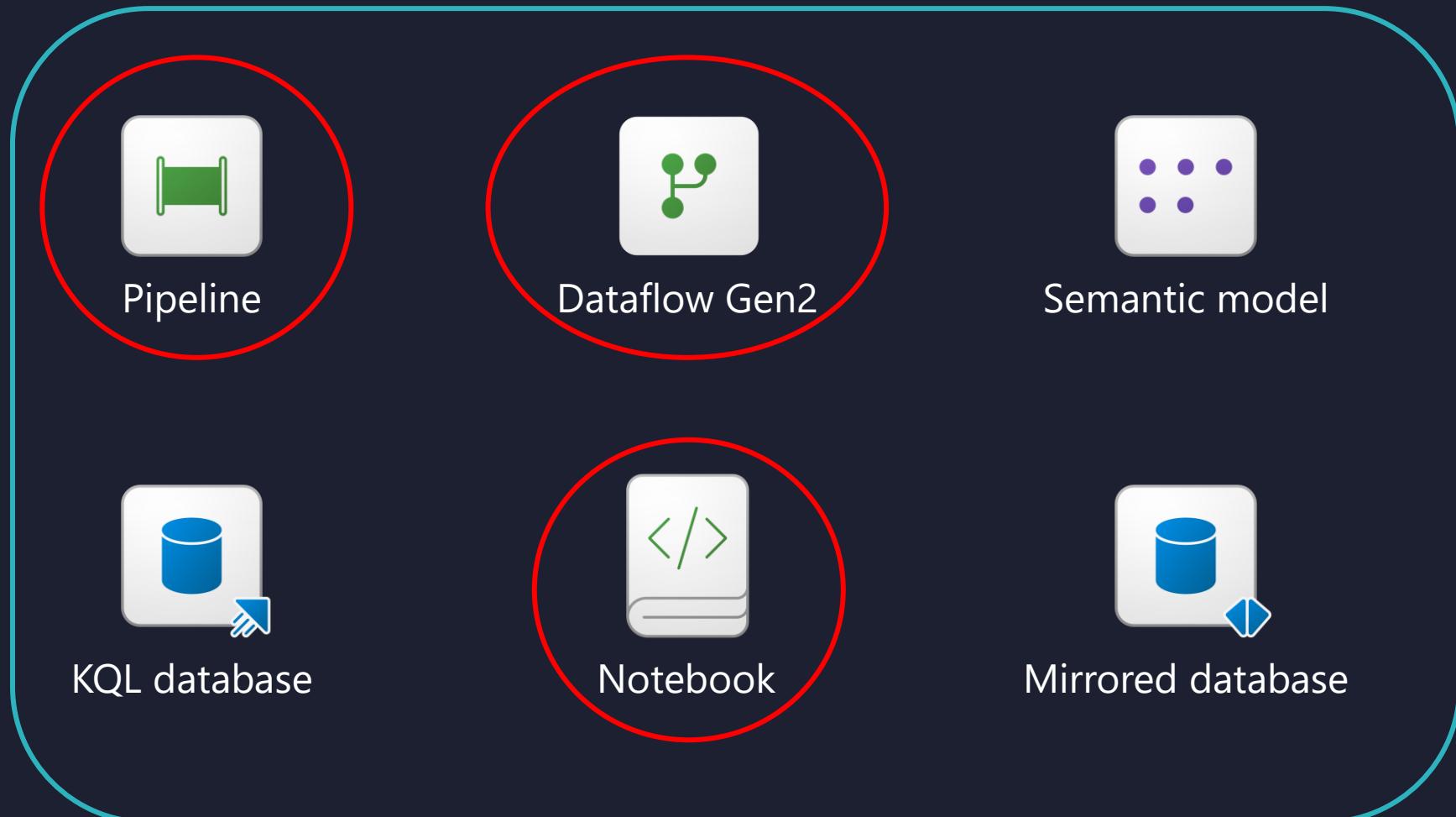
# Create a Data Connection



@DataMozart



# Create a Data Connection



@DataMozart



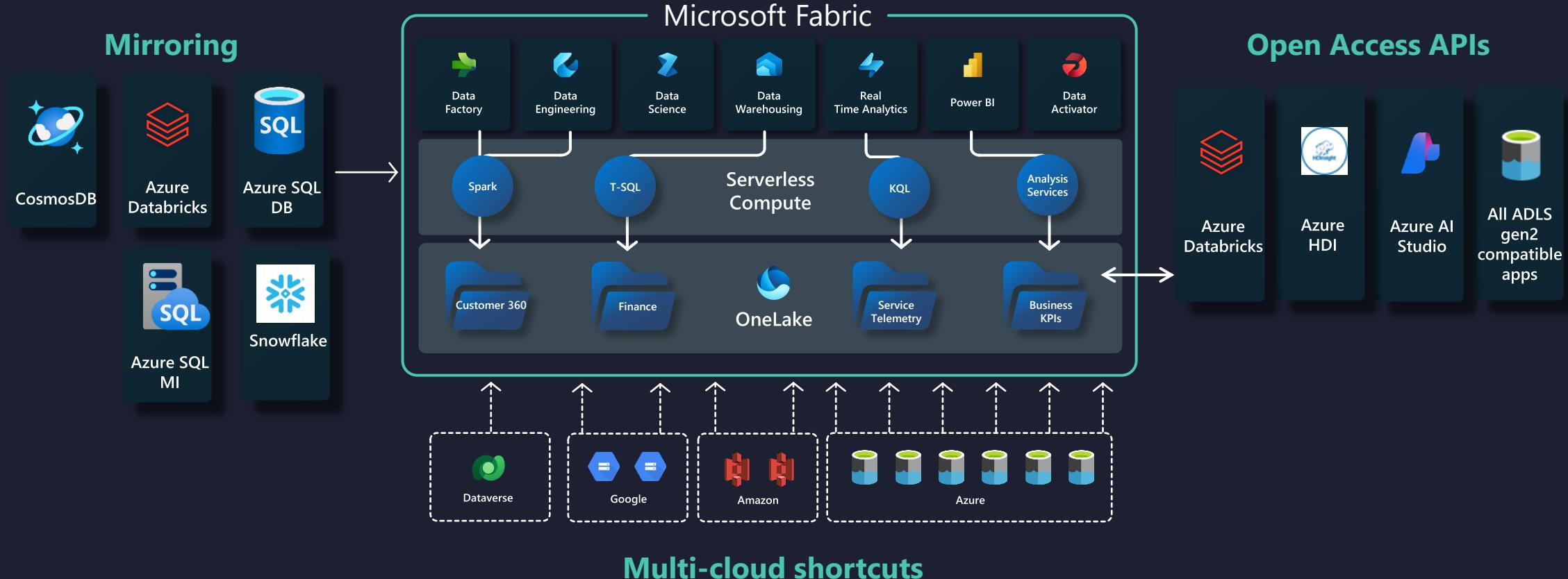
# Ingest or Access Data



All roads lead to...

~~Rome OneLake~~

Fabric compute engines



@DataMozart



# Shortcuts



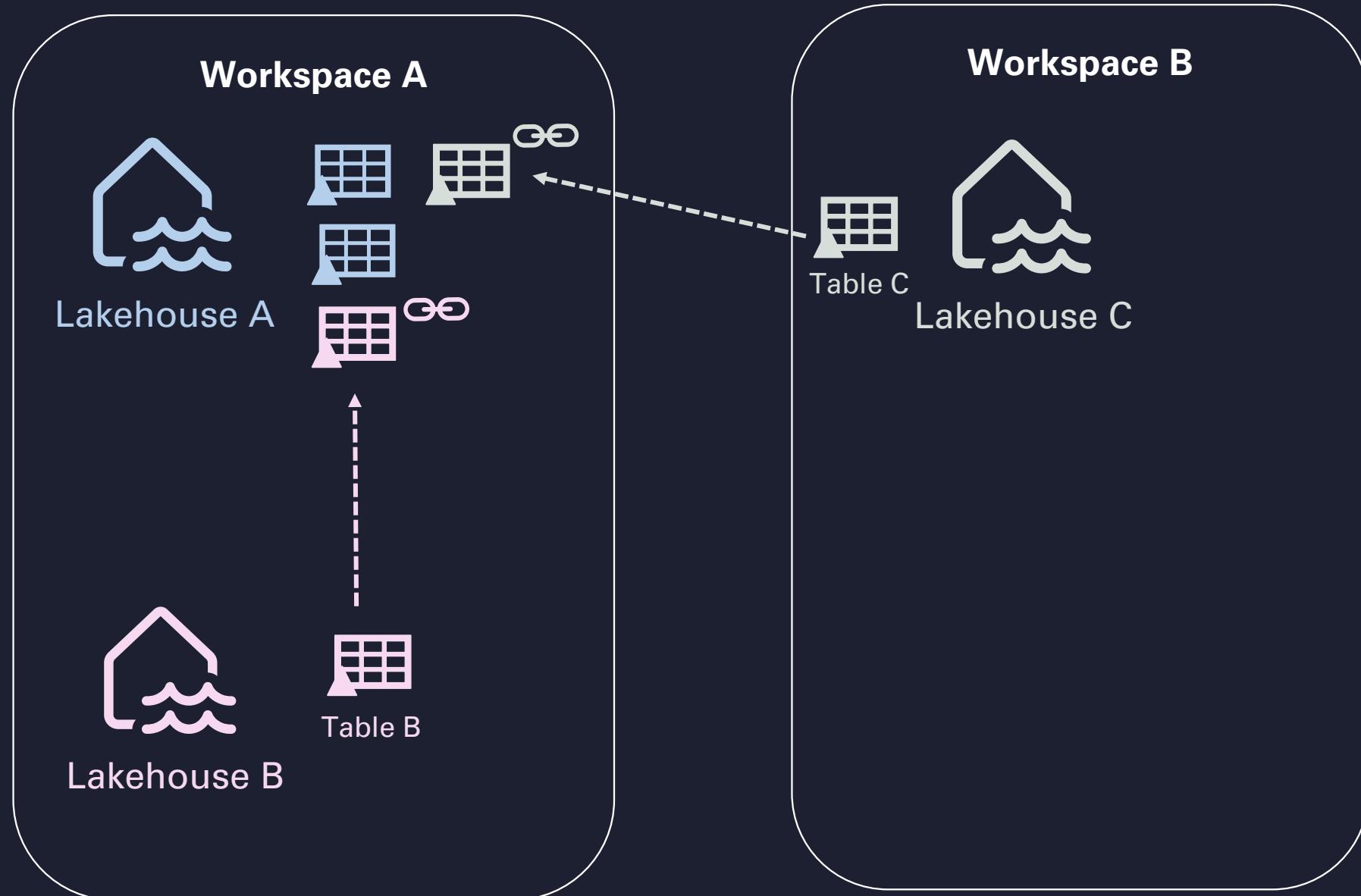
**Shortcuts in  
Windows Explorer?**



@DataMozart



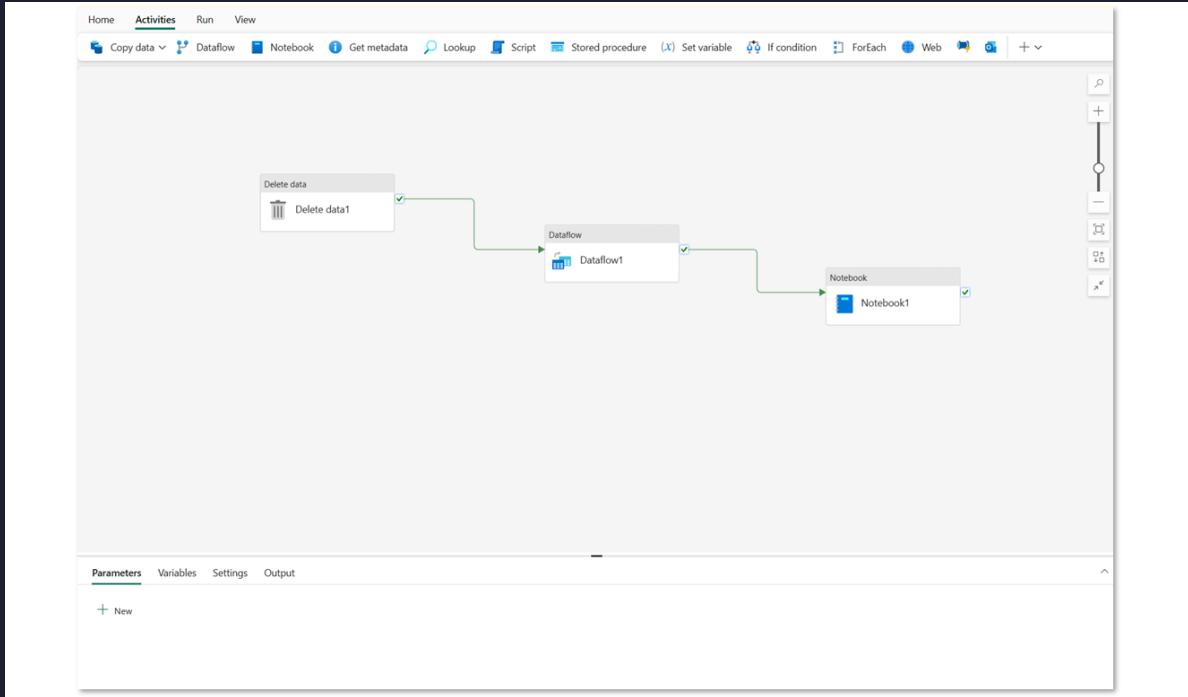
# Shortcuts



@DataMozart



# Pipelines



## Pipeline concepts:

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



@DataMozart



# Common Activities – Copy Data



The screenshot shows the Azure Data Factory 'Copy data' tool. On the left, there's a 'Choose data source' step with options like 'COVID-19 Data Lake', 'NYC Taxi - Green', 'Diabetes', 'Public Holidays', and 'Retail Data Model from Wide World Importers'. Below it is a 'Data sources' section with categories like All categories, Workspace, Database, File, Generic protocol, and Services and apps, listing various cloud services. On the right, the main canvas shows a pipeline named 'Copy\_Produc\_Data' with a single activity. The pipeline settings show 'Destination' tab selected, with 'Workspace' data store type set to 'Lakehouse' in 'DP601\_Bronze' folder, and 'Table name' set to 'WWI\_Products'. A teal arrow points from the left side of the interface towards the pipeline canvas on the right.

1. Use the copy data tool

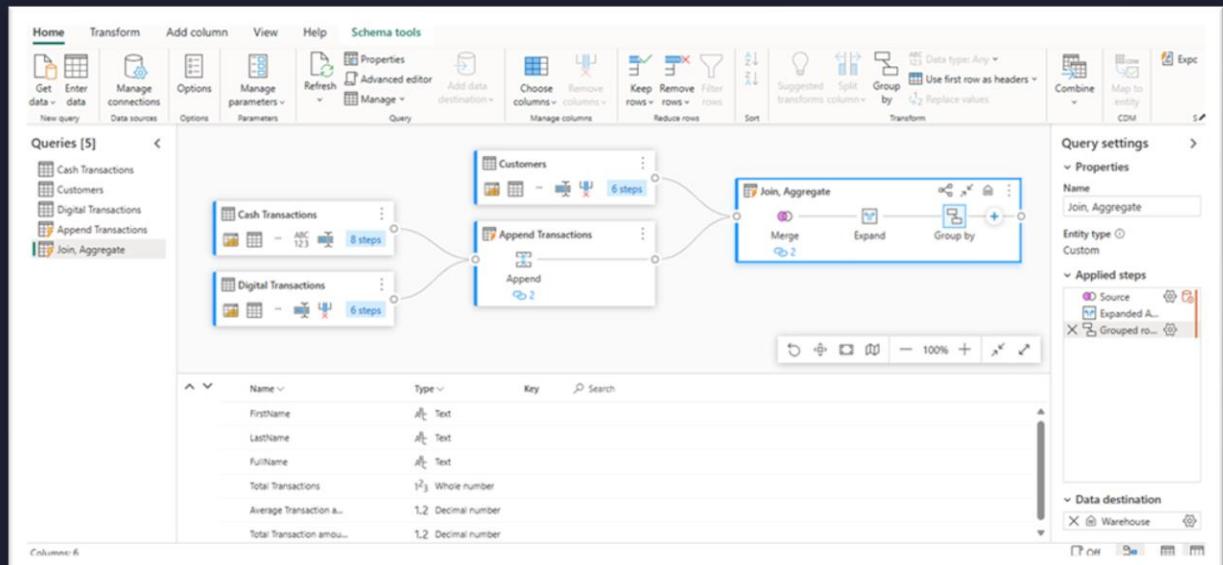
2. Edit the settings below the pipeline canvas



# 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



@DataMozart



# Lakehouse, Warehouse or Eventhouse



**When to choose what?!**



@DataMozart



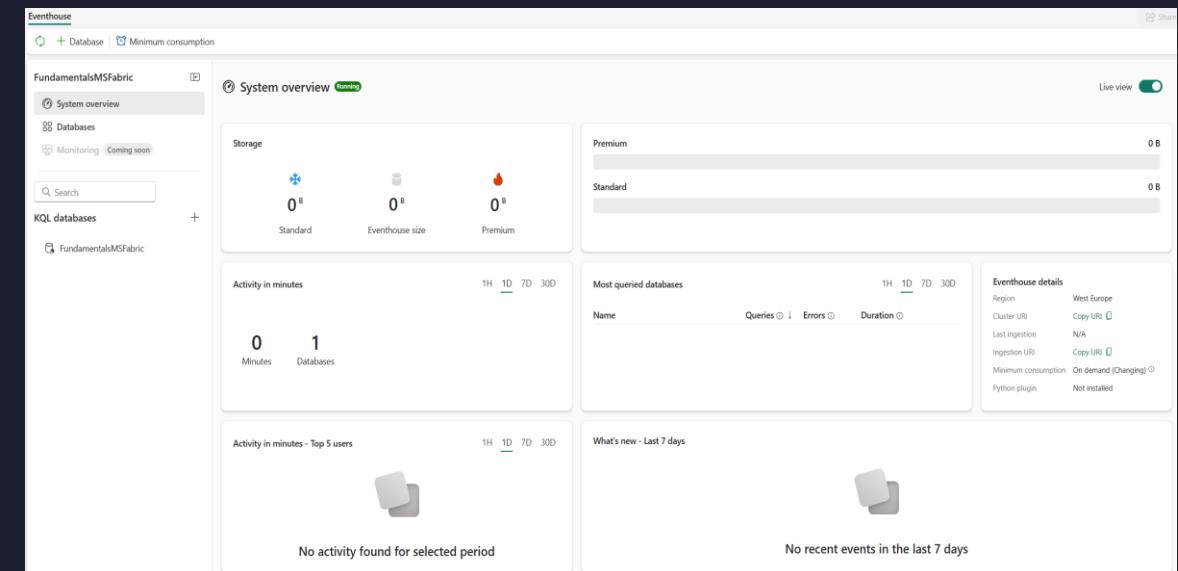
# Handling streaming data!



# Eventhouse



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



@DataMozart



# KQL Database



Workspace



Eventhouse A

KQL database 1



Eventhouse B

KQL database 1



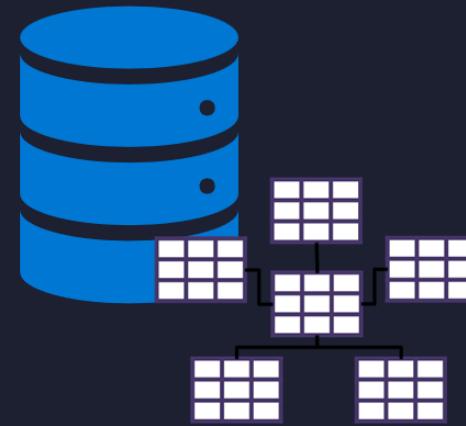
KQL database 2



@DataMozart



# Fabric “Houses”



## Data Lake**house**

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

## Data **Warehouse**

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



@DataMozart



# 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

The screenshot shows a Fabric Notebook interface. At the top, there's a toolbar with icons for file operations, a progress bar, and tabs for 'Run all', 'Stop session', 'Language' (set to 'PySpark (Python)'), and 'Open in VS Code'. Below the toolbar, a message states: 'Synapse notebooks and Spark job definitions are in Preview. Other users in your organization may have access to this workspace. Do not use these items unless you trust all other users who may have access to the workspace.' The main area is divided into two sections: 'Lakehouse explorer' on the left and a notebook editor on the right. The notebook editor contains two cells. The first cell is a code cell with the following content:1 # Welcome to your new notebook  
2 # Type here in the cell editor to add code!  
3The second cell is a markdown cell with the following content:1 # Heading 1  
2  
3 Some Markdown code:  
4 |  
5 - list  
6 - \*\*of\*\*  
7 - thingsBelow the cells, there are rich text editing tools (bold, italic, underline, etc.) and a preview section. The preview shows the heading 'Heading 1' and the text 'Some Markdown code:' followed by a bulleted list: '• list', '• of', and '• things'.

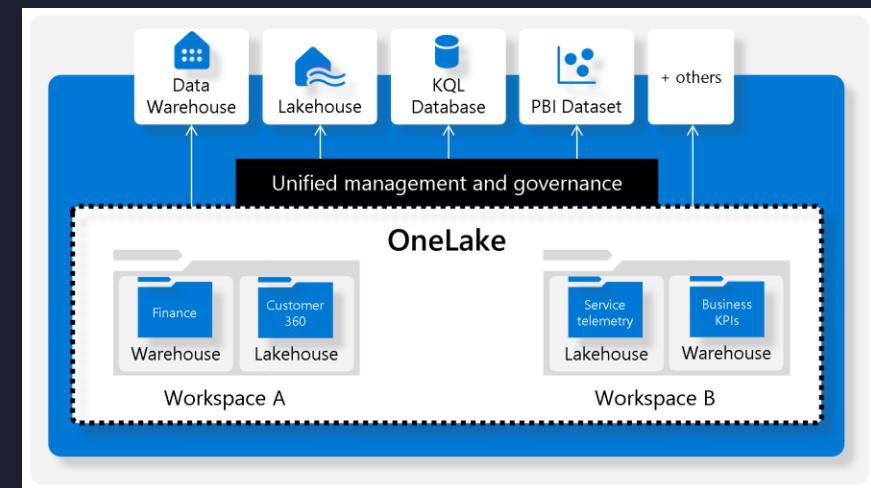


@DataMozart



# Warehouse

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



@DataMozart



# Working With Fabric Warehouse

Tools and techniques to explore and transform data



**SQL endpoint**



**Dataflows (Gen2)**



**Data Pipelines**

**Visualize with Power BI**



@DataMozart



# Two Types of SQL-“Houses” in Fabric



SQL Endpoint of the Lakehouse



Synapse Data Warehouse

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

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





Lakehouse



Warehouse



Spark +  
Notebooks

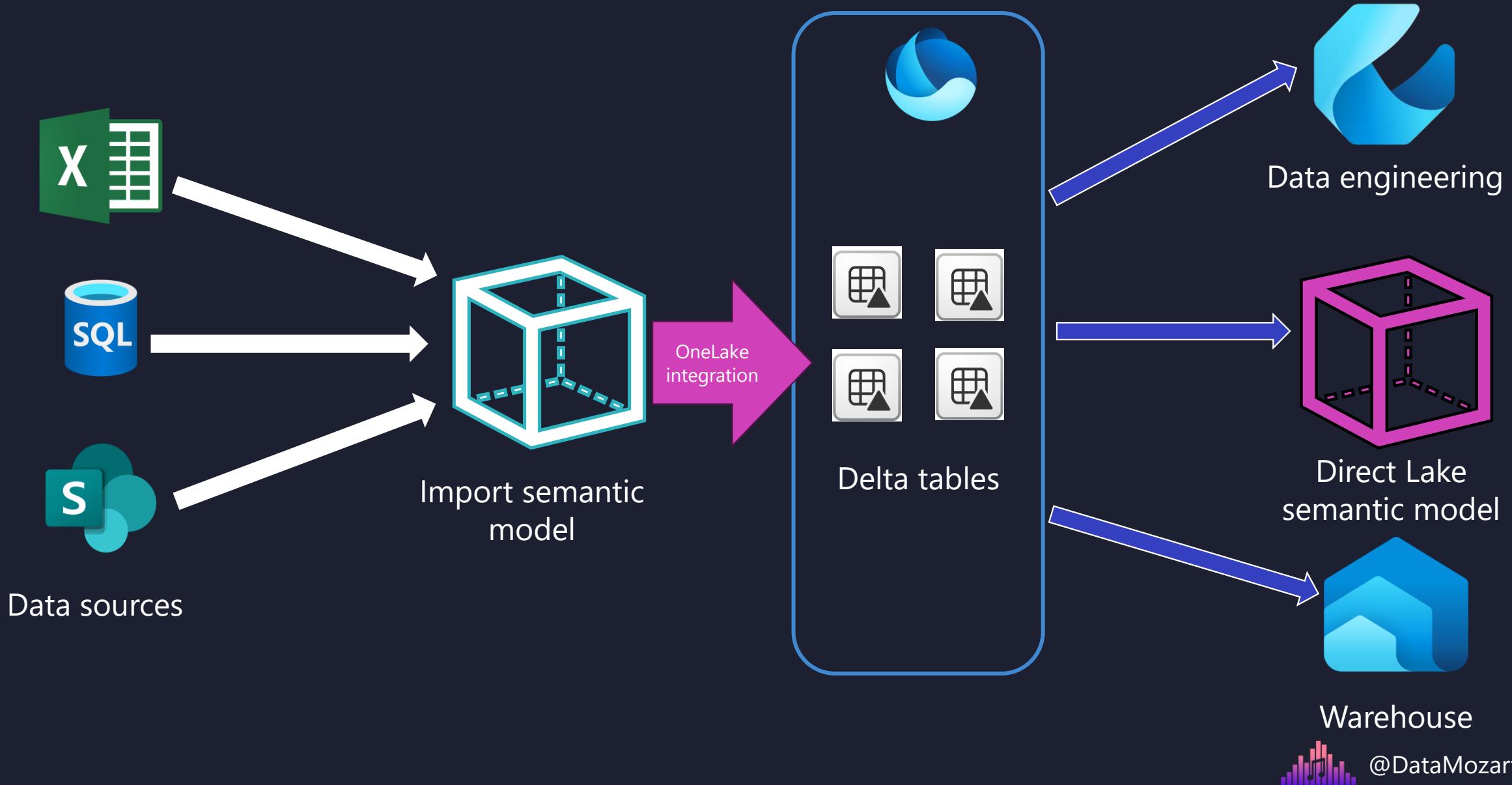
Process



T-SQL

Process

# OneLake Integration for Semantic Models and Eventhouse



# OneLake Integration for Semantic Models



- ▷ Large semantic model storage format
- ▷ Query scale-out
- ▷ OneLake Integration
  - You can automatically write data imported into your semantic model tables to delta tables in OneLake. Make sure your semantic model includes one or more import tables. [Learn more](#)
  - On
  - Apply
  - Discard

- ▷ Semantic models can export data to OneLake (preview)  
*Enabled for the entire organization*

Semantic models configured for OneLake integration can send in Once the data is in OneLake, users can include the exported table including lakehouses and warehouses. [Learn More](#)

Enabled

This setting applies to the entire organization

Apply     Cancel
- ▷ Users can store semantic model tables in OneLake (preview)  
*Enabled for the entire organization*

When users turn on OneLake integration for their semantic model semantic model tables can be stored in OneLake. To allow users to integration for their semantic models, you'll also need to turn on can export data to OneLake" tenant setting. [Learn More](#)

Enabled

Apply to:  
 The entire organization  
 Specific security groups  
 Except specific security groups

Apply     Cancel



@DataMozart



# OneLake Integration for Eventhouse

	Turned on	Turned off
<b>KQL Database</b>	<ul style="list-style-type: none"><li>- Existing tables aren't affected. New tables are made available in OneLake.</li><li>- The <a href="#">Data retention policy</a> 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.</li></ul>	<ul style="list-style-type: none"><li>- Existing tables aren't affected. New tables won't be available in OneLake.</li></ul>
<b>A table in KQL Database</b>	<ul style="list-style-type: none"><li>- New data is made available in OneLake.</li><li>- Existing data isn't backfilled.</li><li>- Data can't be deleted, truncated, or purged.</li><li>- Table schema can't be altered and the table can't be renamed.</li><li>- Row Level Security can't be applied to the table.</li></ul>	<ul style="list-style-type: none"><li>- New data isn't made available in OneLake.</li><li>- Data can be deleted, truncated, or purged.</li><li>- Table schema can be altered and the table can be renamed.</li><li>- Data is soft deleted from OneLake.</li></ul>

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



@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





## 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



@DataMozart



# Organizing Data in a Lakehouse



Bronze

- ✓ 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



@DataMozart



# Organizing Data in a Lakehouse



Silver

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



@DataMozart



# Organizing Data in a Lakehouse



Gold

- ✓ 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



@DataMozart



# 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!



@DataMozart

# Enrich Data by Adding New Columns or Tables



Lakehouse



Warehouse



Eventhouse



@DataMozart

# Enrich Data by Adding New Columns or Tables



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

## Lakehouse

```
1 from pyspark.sql.functions import col, split
2
3 # Create customer dataframe
4
5 dfdimCustomer = df.dropDuplicates(["CustomerName", "Email"]).select(col("CustomerName"), col("Email")) \
6     .withColumn("First", split(col("CustomerName"), " ").getItem(0)) \
7     .withColumn("Last", split(col("CustomerName"), " ").getItem(1))
```



@DataMozart

# 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")
```



@DataMozart

# 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
```

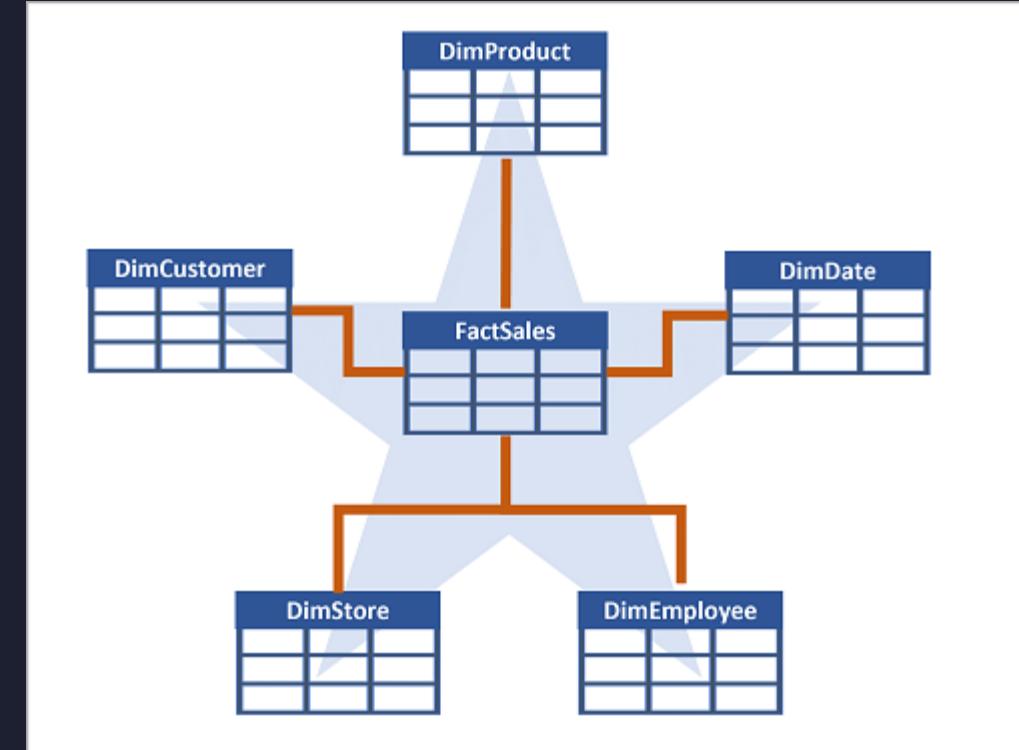


@DataMozart



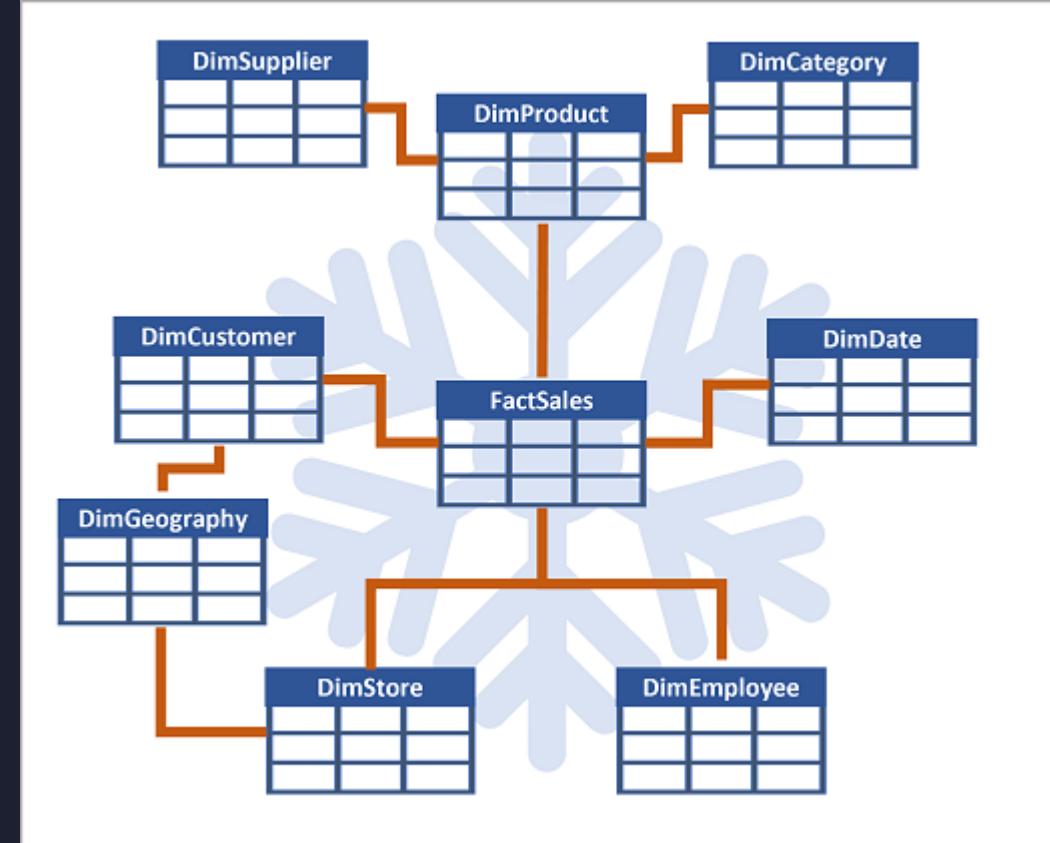
# Implement a Star Schema for a Lakehouse

- 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, 3<sup>rd</sup> normal form is optimal
- Data writing speed

## Denormalization

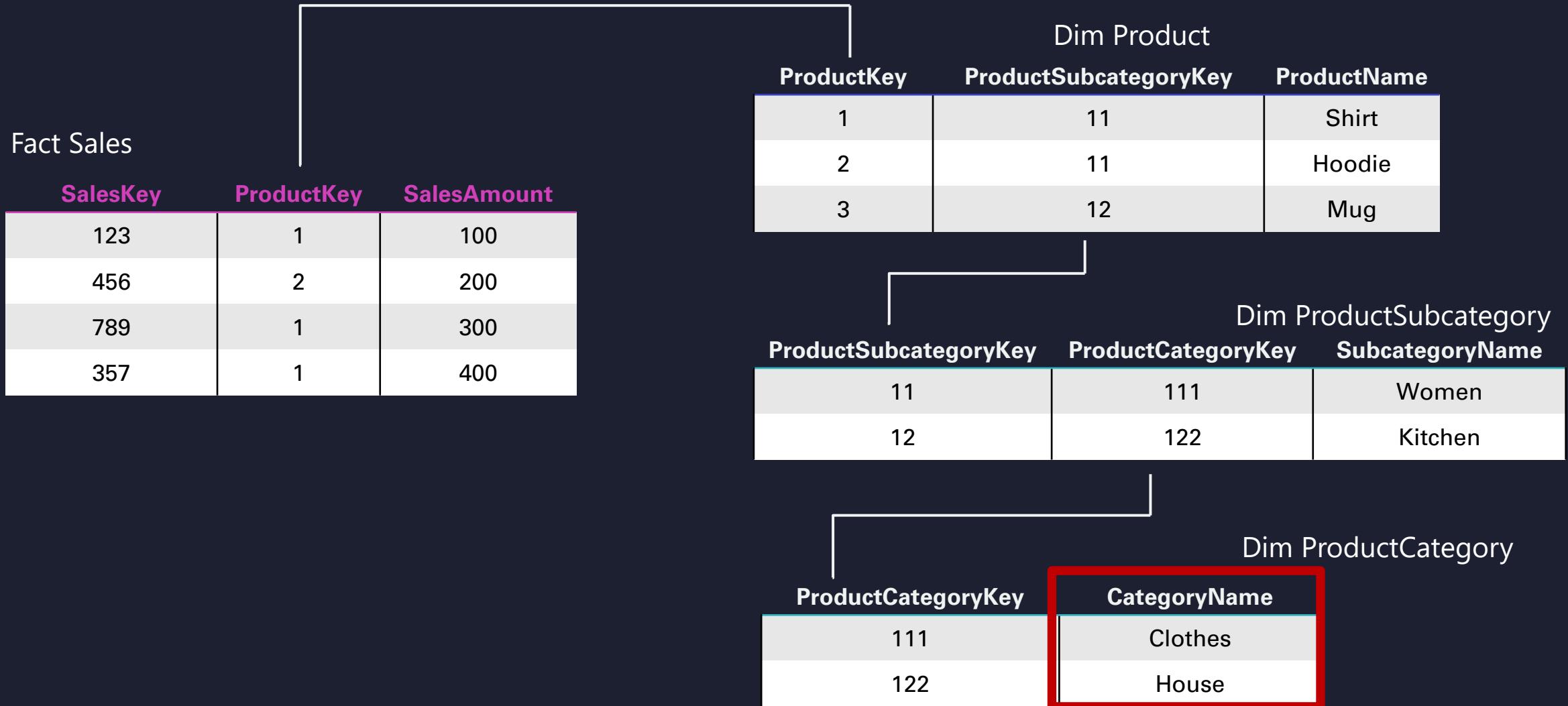
Creates redundant data in the table

- Data reading speed





# Denormalization





# 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





# 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



@DataMozart



# 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

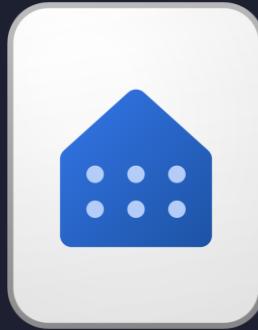




# Aggregations in Microsoft Fabric



Lakehouse



Warehouse



Eventhouse

✓ PySpark/SparkSQL

✓ T-SQL

✓ KQL



@DataMozart



# Merge or Join Data



Lakehouse



Warehouse



Eventhouse



Dataflow Gen2

✓ PySpark/SparkSQL

✓ T-SQL

✓ KQL

✓ Low-code/No-code



@DataMozart



# Merge or Join Data

Table A

Fruit
Apple
Orange
Banana

Table B

Fruit
Apple
Orange
Mango

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





# Identify and Resolve Duplicate Data



Lakehouse



Warehouse



Eventhouse

✓ PySpark/SparkSQL

```
1 | from fivetran import DeviceEventsAll
2 | # Create CTE
3 | # Create CTE
4 | # Create CTE
5 | dfdimCustomer = df.dropDuplicates()
```

```
1 | WITH CTE AS (
2 |     SELECT
3 |         DeviceId,
4 |         EventId,
5 |         StationId,
6 |         EventDateTime,
7 |         DeviceName,
8 |         Email,
9 |         RowNum
10 |     FROM DeviceEventsAll
11 |     WHERE EventDateTime > ago(90d)
12 |     GROUP BY DeviceId, EventId, StationId, EventDateTime, DeviceName, Email
13 |     HAVING COUNT(*) > 1
14 | )
15 | DELETE FROM CTE
16 | WHERE RowNum > 1;
```

✓ T-SQL

✓ KQL

@DataMozart



# Identify and Resolve Missing Data



Lakehouse



Warehouse



Eventhouse

✓ PySpark/SparkSQL

✓ T-SQL

✓ KQL



@DataMozart



# 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()
```



@DataMozart



# Identify and Resolve Missing Data



Warehouse

✓ T-SQL

✓ ***COALESCE()*** and ***ISNULL()***

First non-null expression

Replace NULL with the specified value

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

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



@DataMozart



# 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



@DataMozart



# 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)
```



@DataMozart



# Convert Column Data Types



Dataflow Gen2

- ✓ Low-code/No-code



Pipeline

- ✓ Copy data

Source	Type	Destination	Type	+ <span style="font-size: small;">Delete</span>
CurrencyKey	int	CurrencyKey	integer	+ <span style="font-size: small;">Delete</span>
CurrencyAlternateKey	nchar	CurrencyAlternateKey	string	+ <span style="font-size: small;">Delete</span>
CurrencyName	nvarchar	CurrencyName	string	+ <span style="font-size: small;">Delete</span>



@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



@DataMozart



# Filter Data



Dataflow Gen2

✓ Low-code/No-code

The screenshot shows the Microsoft Power BI Dataflow Gen2 interface. On the left, there's a 'Queries [1]' pane with a single 'Query' item. The main area displays a table titled 'Source{[Schema = "dbo", Item = "DimCustomer"]}[Data]'. The table has columns: CustomerKey, GeographyKey, CustomerAlternateKey, Title, FirstName, MiddleName, LastName, NameStyle, BirthDate, and several date columns. A filter dialog is open over the table, specifically for the 'FirstName' column. The dialog title is 'Text filters' and includes a search bar and a list of names: (Select all), Abigail, Adam, Aimee, Alan, Alejandro, Alyssa. Below the list is a note: 'List may be incomplete.' At the bottom of the dialog are 'OK' and 'Cancel' buttons.



Pipeline

✓ Filter activity

The screenshot shows the Azure Data Factory pipeline editor. On the right, there's a sidebar with various activities: Move and transform (Copy data, Dataflow, Delete data), Metadata and validation (Lookup, Get metadata), Control flow (If conditions, Switch, Filter, Wait, ForEach, Until). The 'Filter' activity is highlighted with a red circle. In the center, there's a 'Filter1' activity in a green rounded rectangle, connected to other components in a flow.

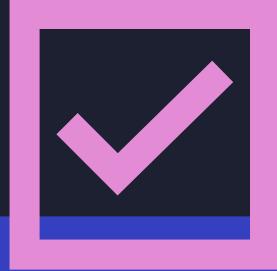


@DataMozart



## 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



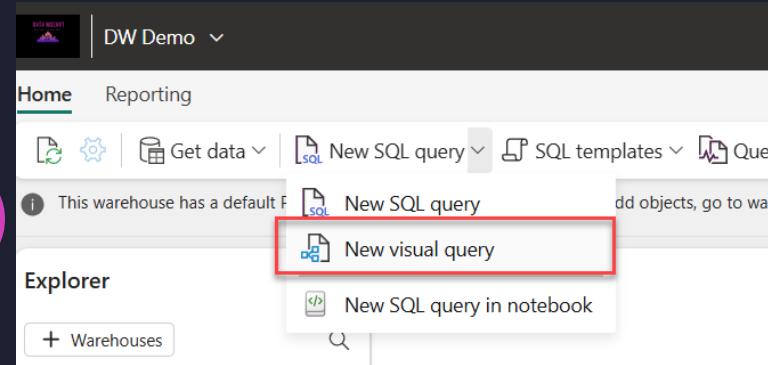


# Query and analyze data

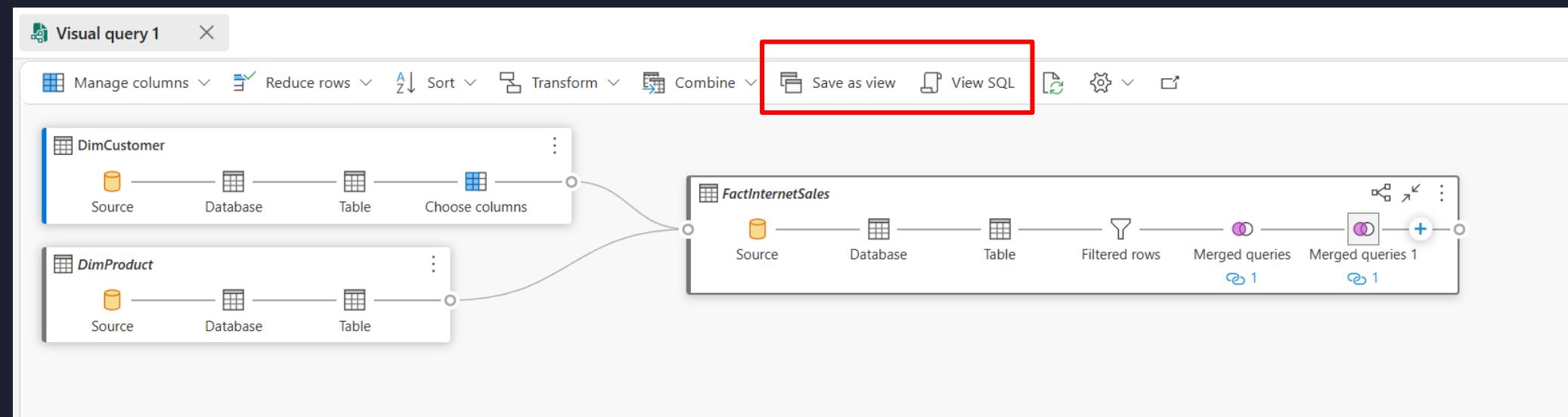
- Select, filter, and aggregate data by using the Visual Query Editor
- Select, filter, and aggregate data by using SQL
- Select, filter, and aggregate data by using KQL



# Query Data with Visual Query Editor



1



2



# Query Data with SQL

## Window functions

***OVER (defines a window)***

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

```
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
```

***ORDER BY (required depending on the function)***



@DataMozart



# Query Data with SQL

## Ranking functions

### ROW\_NUMBER

	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

	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

	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

	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

	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

	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...

The screenshot shows a KQL editor interface with the following components:

- Top Bar:** Includes "Run", "Preview", "Recall", "Copy query", "Pin to dashboard", "KQL Tools", and "Export to CSV".
- Query Editor:** Displays the following KQL code:

```
1 explain
2 SELECT State
3 , MAX(InjuriesDirect) as InjuriesDirect
4 FROM Weather
5 GROUP BY State
6 |
```
- Table View:** Shows "Table 1" with a "Query" section containing:

```
Weather | project-rename ['Weather.InjuriesDirect']=InjuriesDirect | summarize InjuriesDirect=max(['Weather.InjuriesDirect']) by State | project State, InjuriesDirect
```
- Bottom Bar:** Includes "JPath:" dropdown with options "Query", "Inline", and "Full".

Below the table view, there is a T-SQL translation of the KQL query:

```
1 "Query": Weather
2 | project-rename ['Weather.InjuriesDirect']=InjuriesDirect
3 | summarize InjuriesDirect=max(['Weather.InjuriesDirect']) by State
4 | project State, InjuriesDirect
5 |
```

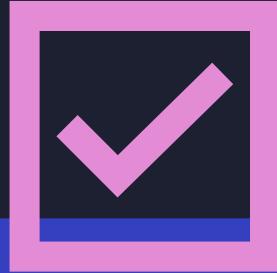


@DataMozart



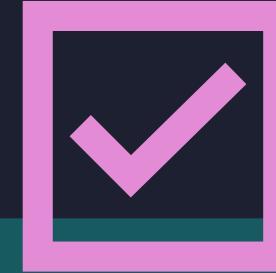
## Get data

- 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





# Implement and Manage Semantic Models

25-30%



@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

## 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





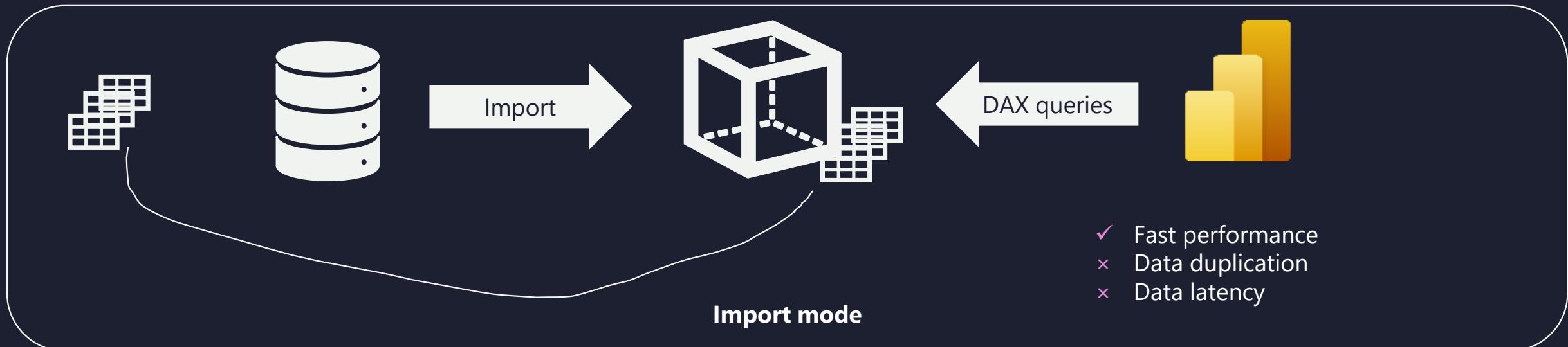
## Design and build semantic models

- 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





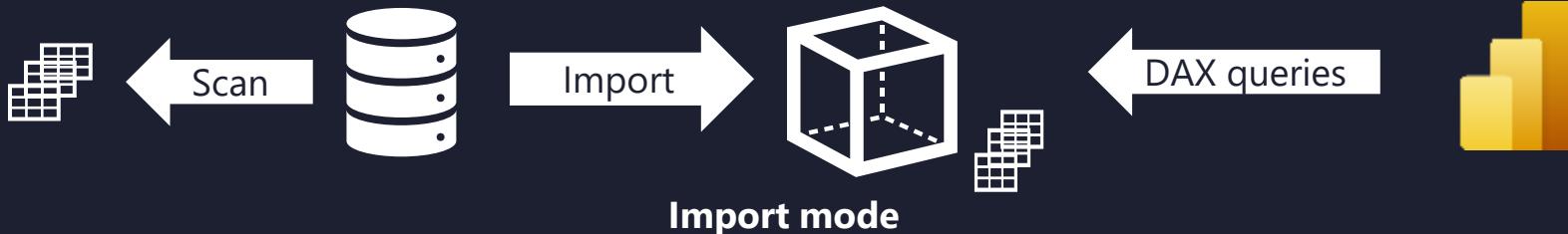
# Power BI Architecture – Pre-Fabric



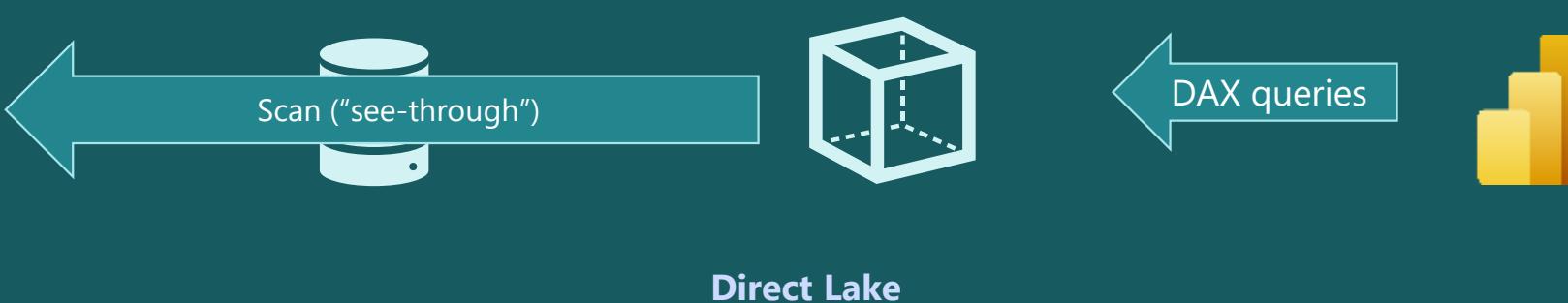
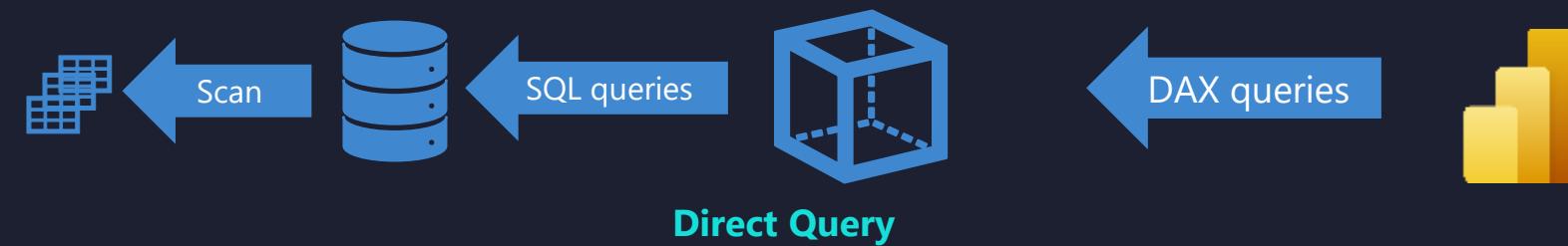


# Power BI Architecture – Fabric

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



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





# 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!**



@DataMozart



# Implement Relationships



Relationships serve as filters!

## Cardinality

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

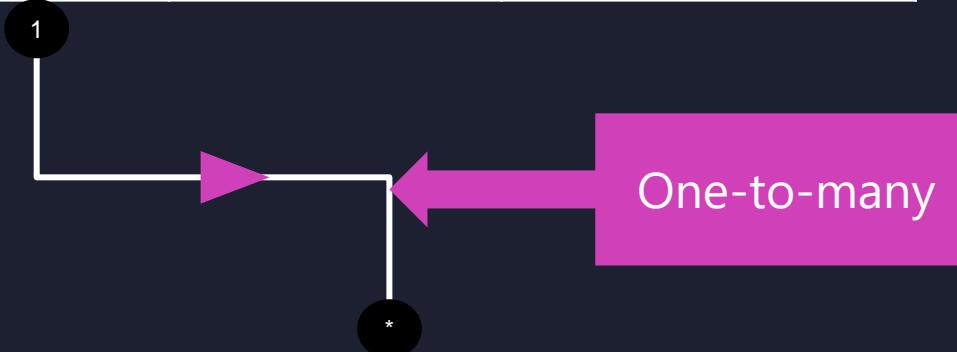


@DataMozart



# Relationship Cardinality (1:\*, \*:1)

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



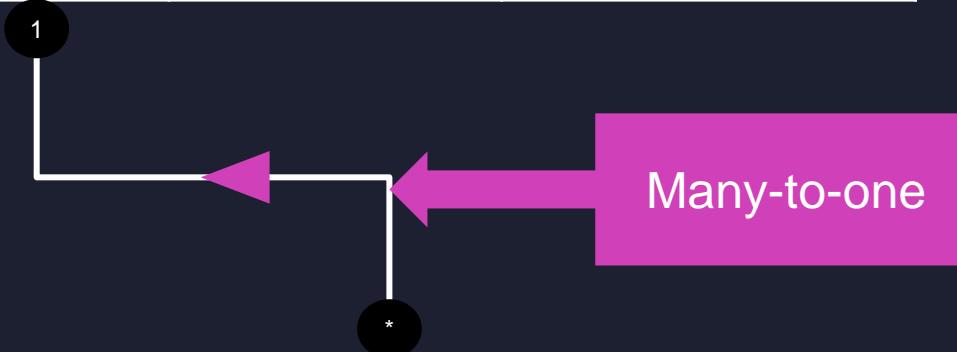
Sales Key	Product Key	Sales Amount
1	1	100
2	1	200
3	1	300
4	2	50
5	2	75
6	3	100
7	3	150





# Relationship Cardinality (1:\*, \*:1)

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

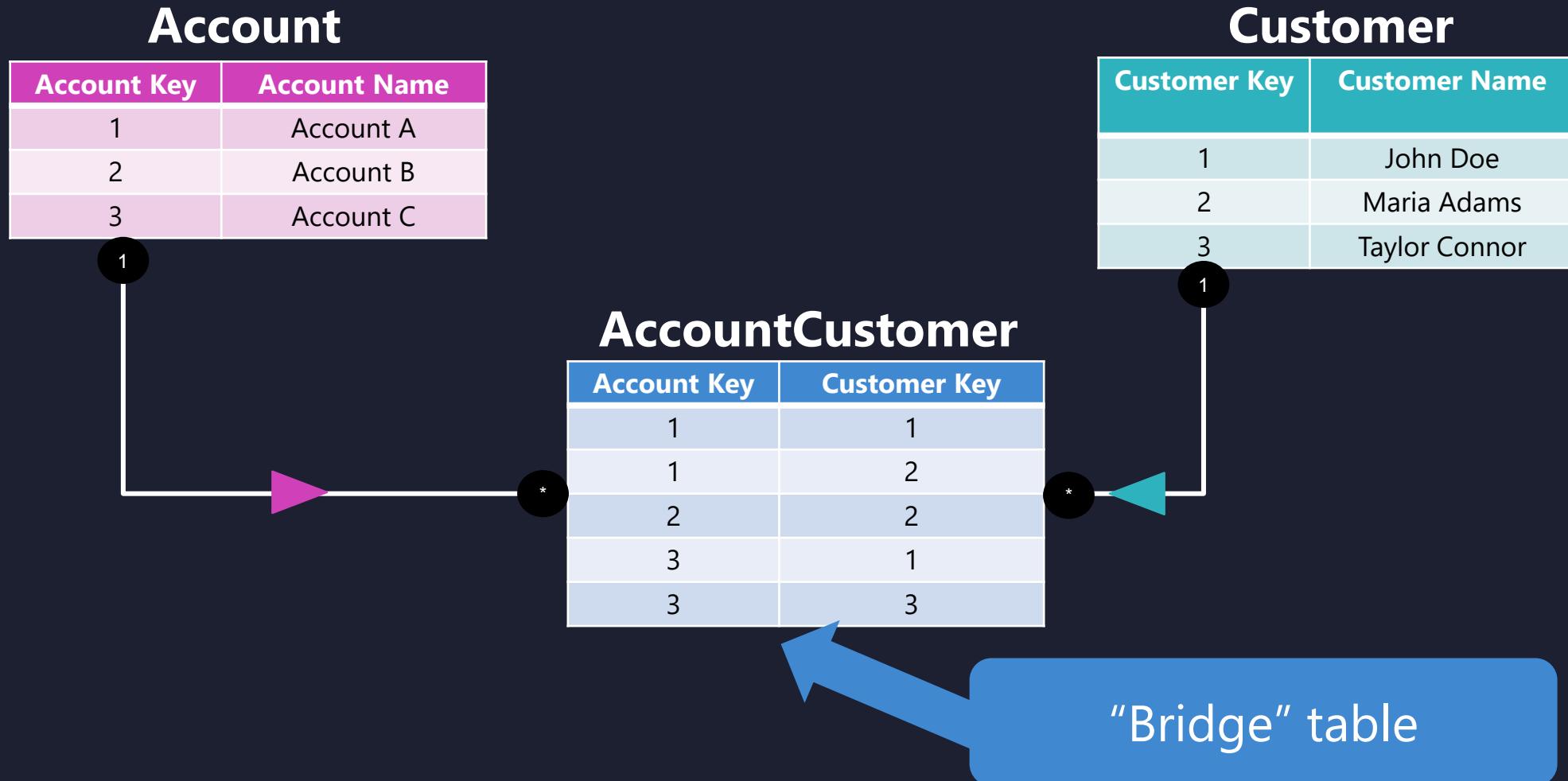


Sales Key	Product Key	Sales Amount
1	1	100
2	1	200
3	1	300
4	2	50
5	2	75
6	3	100
7	3	150





# Relationship Cardinality (\*:\*)



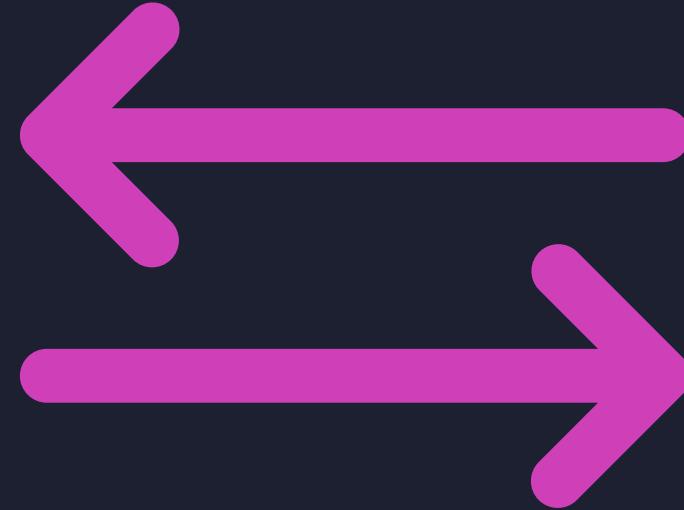


# Relationship Filter Direction



Single Direction

Recommended practice



Both Directions

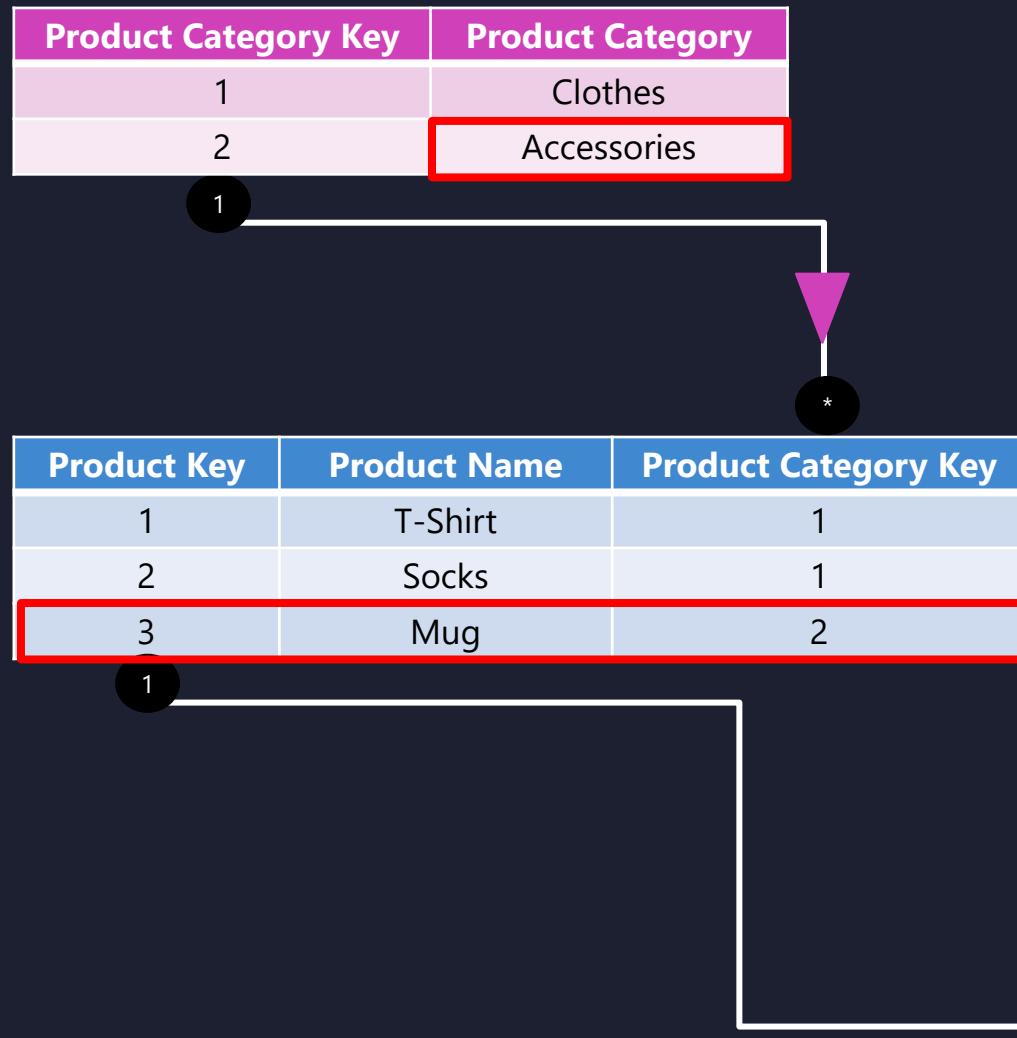
Implement with care



@DataMozart



# Relationship Filter Direction (Single)



The diagram illustrates a relationship between the **Sales** and **Product** tables. A line connects the **Sales Key** in the **Sales** table to the **Product Key** in the **Product** table. A black circle with an asterisk (\*) is at the start of the line, and a black circle with the number '1' is at the end, indicating a many-to-1 relationship.

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

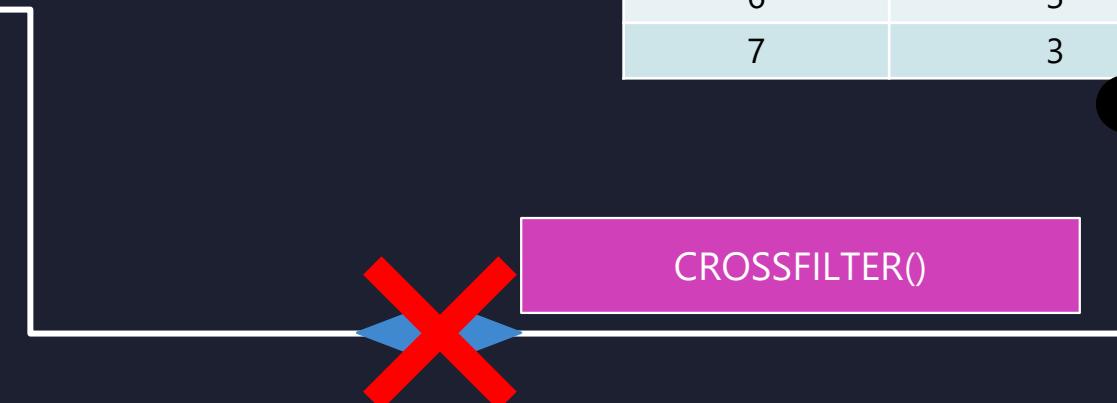


# Relationship Filter Direction (Both)

Product Category Key	Product Category
1	Clothes
2	Accessories



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



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



# DAX Variables

- Reuse expression logic
- Improve performance
- Easy debug

```
1 YoY Variance % =
2     var _currentSales = SUM(FactResellerSales[SalesAmount])
3     var _prevYearSales = CALCULATE(
4         SUM(FactResellerSales[SalesAmount]),
5             SAMEPERIODLASTYEAR(DimDate[FullDateAlternateKey])
6     )
7     var _result = DIVIDE(_currentSales - _prevYearSales,_prevYearSales,0)
8     RETURN
9     _result
```

```
1 YoY Variance % NO VARs =
2     DIVIDE(
3         SUM(FactResellerSales[SalesAmount]) -
4             CALCULATE(
5                 SUM(FactResellerSales[SalesAmount]),
6                     SAMEPERIODLASTYEAR(DimDate[FullDateAlternateKey])
7                 )
8             ,CALCULATE(
9                 SUM(FactResellerSales[SalesAmount]),
10                SAMEPERIODLASTYEAR(DimDate[FullDateAlternateKey])
11                )
12            ,0
13        )
14
15
```





# 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...



@DataMozart



# PARTITIONBY(), ORDERBY()

## PARTITIONBY()

Determines the “window” as a subset of rows

## ORDERBY()

Defines the sorting order within the “window”



@DataMozart



# 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



@DataMozart



# 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



@DataMozart



# OFFSET()

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

RETURN

OFFSET(

-1,  
ORDERBY ([YEAR], DESC)

)



@DataMozart



# OFFSET()

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

RETURN

OFFSET(

-2,  
ORDERBY ([YEAR], DESC)

)



@DataMozart



# WINDOW()

The screenshot shows a Microsoft Power BI interface with a table of sales data. The table has columns: Customer, Date, Quantity, and Sales. The data is grouped by Customer. For 'Adam Young', there are four rows: 7/7/2019, 8/27/2019, 10/24/2019, and 3/31/2020. For 'Alexandra Jenkins', there are 16 rows from 7/3/2019 to 6/3/2020. Two arrows point to specific groups of rows: a blue arrow labeled 'Window 1' points to the first four rows of 'Adam Young', and a purple arrow labeled 'Window 2' points to the first four rows of 'Alexandra Jenkins'.

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()

Screenshot of Microsoft Power BI desktop showing a table visual with window functions applied.

The table has four columns: Customer, Date, Quantity, and Sales.

Rows 1 through 4 are highlighted with a dark blue border, representing a window frame starting from the current row (Alexandra Jenkins) and extending back to Adam Young.

Rows 5 through 18 are highlighted with a purple border, representing a window frame starting from the current row (Alexandra Jenkins) and extending forward to Alexandra Jenkins.

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



@DataMozart



# WINDOW()

Screenshot of Microsoft Power BI desktop showing a table visual with a window function applied.

The table has columns: Customer, Date, Quantity, and Sales.

The data shows two groups of purchases:

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



@DataMozart



# 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])  
    )  
)
```



@DataMozart



# DAX Information Functions

Obtain info about data type or filter context

**CONTAINSSTRING()**



TRUE

CONTAINSSTRING(ProductColor, "Blue")

**HASONEVALUE ()**



TRUE

Only one value in the specified column

INFO.

**Useful information about the semantic model**

**INFO.TABLES**

**INFO.COLUMNS**

**INFO.MEASURES**

**INFO.CALCULATIONGROUPS**

**INFO.HIERARCHIES**



@DataMozart



# 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!



@DataMozart



# 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)  
)**

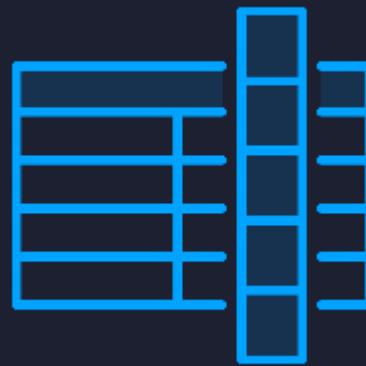
Time Calculation Calendar Year	Current		YOY		YOY%	
	Sales	Orders	Sales	Orders	Sales	Orders
<b>2012</b>						
January	\$495,364	252	\$25,540	108	5.44%	75.00%
February	\$506,994	260	\$40,659	116	8.72%	80.56%
March	\$373,483	212	(\$111,716)	62	-23.03%	41.35%
April	\$400,336	219	(\$101,738)	62	-20.28%	39.49%
May	\$358,878	207	(\$202,804)	33	-36.11%	18.97%
June	\$355,160	310	(\$102,600)	88	-24.75%	30.26%
July	\$444,558	246	(\$152,180)	56	-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	313	(\$173,049)	92	-24.45%	41.65%
November	\$537,956	324	(\$122,590)	116	-18.58%	55.77%
December	\$624,502	487	(\$44,429)	265	-6.71%	119.37%
<b>2013</b>						
January	\$857,690	1602	\$362,326	1410	73.14%	559.52%
February	\$771,349	3453	\$264,355	3193	52.14%	1228.09%
March	\$1,048,907	4087	\$676,424	3875	181.11%	1827.83%
April	\$1,046,073	3979	\$645,687	3760	161.29%	1716.89%
May	\$1,284,593	4199	\$925,715	4192	257.95%	2025.12%
June	\$1,643,178	5025	\$1,088,018	4707	195.98%	1480.19%
July	\$1,371,676	4671	\$927,118	4425	208.55%	1798.78%
August	\$1,551,056	4865	\$1,027,148	4571	196.05%	1554.75%
September	\$1,447,498	4616	\$961,318	4347	197.73%	1615.99%
October	\$1,673,293	5300	\$1,138,134	4887	212.67%	1593.29%
November	\$1,780,920	5224	\$1,242,965	4902	231.05%	1512.35%
December	\$1,874,360	6231	\$1,249,858	5744	200.14%	1179.41%



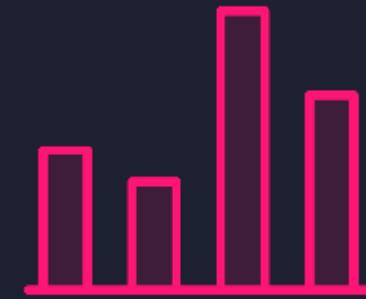


# What can you do with Field Parameters?

Without a single line of DAX!



Dynamically change the attribute



Dynamically change the metrics

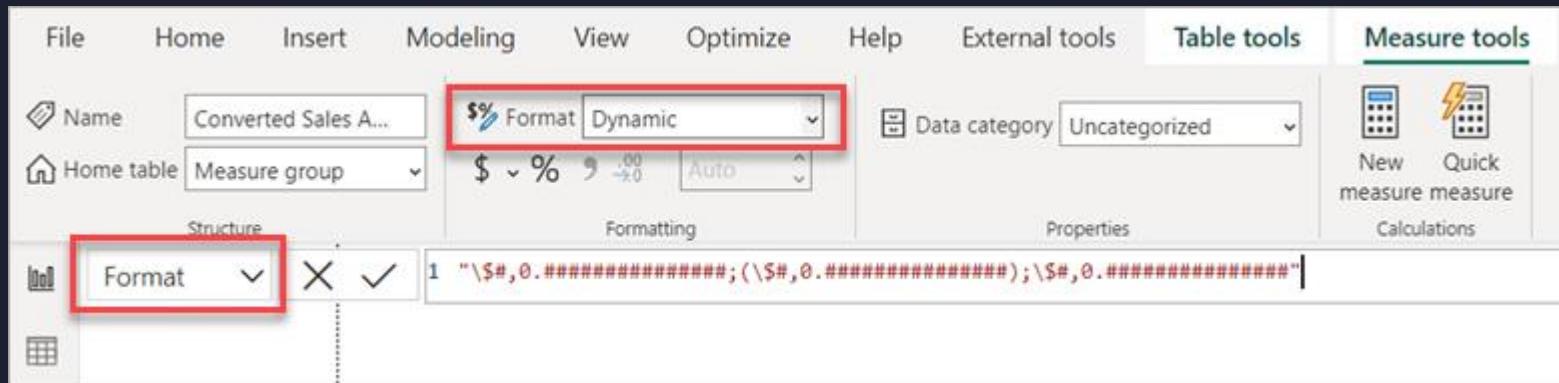


@DataMozart



# 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



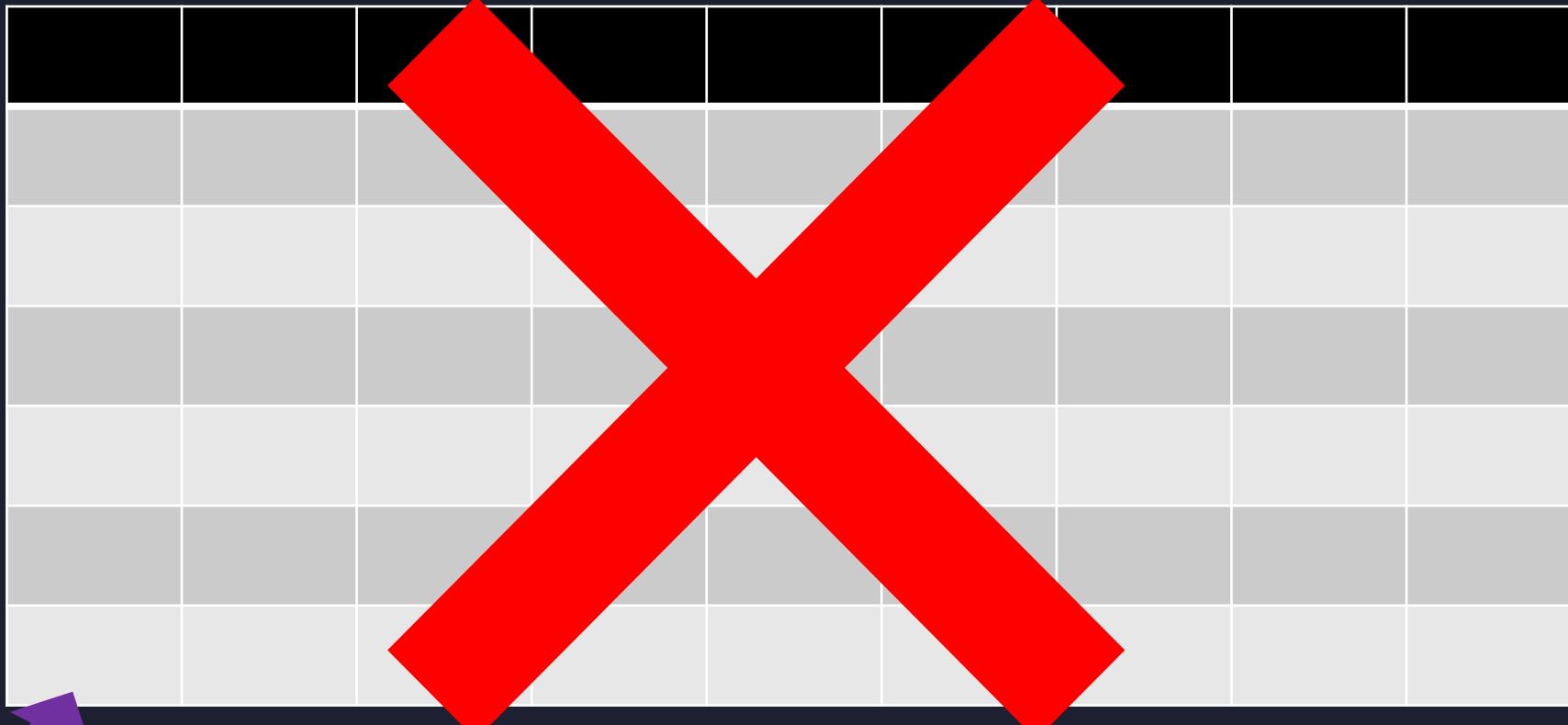

I'm a wide table  
with many  
columns and less  
rows


art





# Large Semantic Models in Power BI



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



@DataMozart



# 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

⚠ Large semantic model storage format  
For most Premium capacities, using large semantic model storage format can improve performance. [Learn more](#)

On

## Semantic model settings



Semantic model eviction and On-demand load

Workspace settings

General

License info

Azure connections

System storage

Git integration

OneLake

Workspace identity

Network security

Monitoring

Power BI

Delegated Settings

Data Engineering/Science

Data Factory

share, collaborate on, and distribute Power BI and Microsoft Fabric content, users in the viewer role can access this content without needing a Pro or Premium per-user license. [Learn more](#)

Embedded

Select embedded if the workspace will be hosted in an Azure embedded capacity. ISVs and developers use Power BI Embedded to embed visuals and analytics in their applications. [Learn more](#)

Fabric capacity

Select Fabric capacity if the workspace will be hosted in a Microsoft Fabric capacity. With Fabric capacities, users can create Microsoft Fabric items and collaborate with others using Fabric features and experiences. Explore new capabilities in Power BI, Data Factory, Data Engineering, and Real-Time Intelligence, among others. [Learn more](#)

Trial

Select the free trial per-user license to try all the new features and experiences in Microsoft Fabric for 60 days. A Microsoft Fabric trial license allows users to create Microsoft Fabric items and collaborate with others in a Microsoft Fabric trial capacity. Explore new capabilities in Power BI, Data Factory, Data Engineering, and Real-Time Intelligence, among others. [Learn more](#)

License capacity

Please specify the remote computing resources your items will use with this workspace.

Trial-nikola-nikolic-onmicrosoft.com-06-02-2023-08-14-UTC - West Europe

Semantic model storage format

Choose a storage format for semantic models. [Learn more](#)

Small semantic model storage format

Large semantic model storage format

Select license Cancel

## Workspace settings



@DataMozart



# Composite Models

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

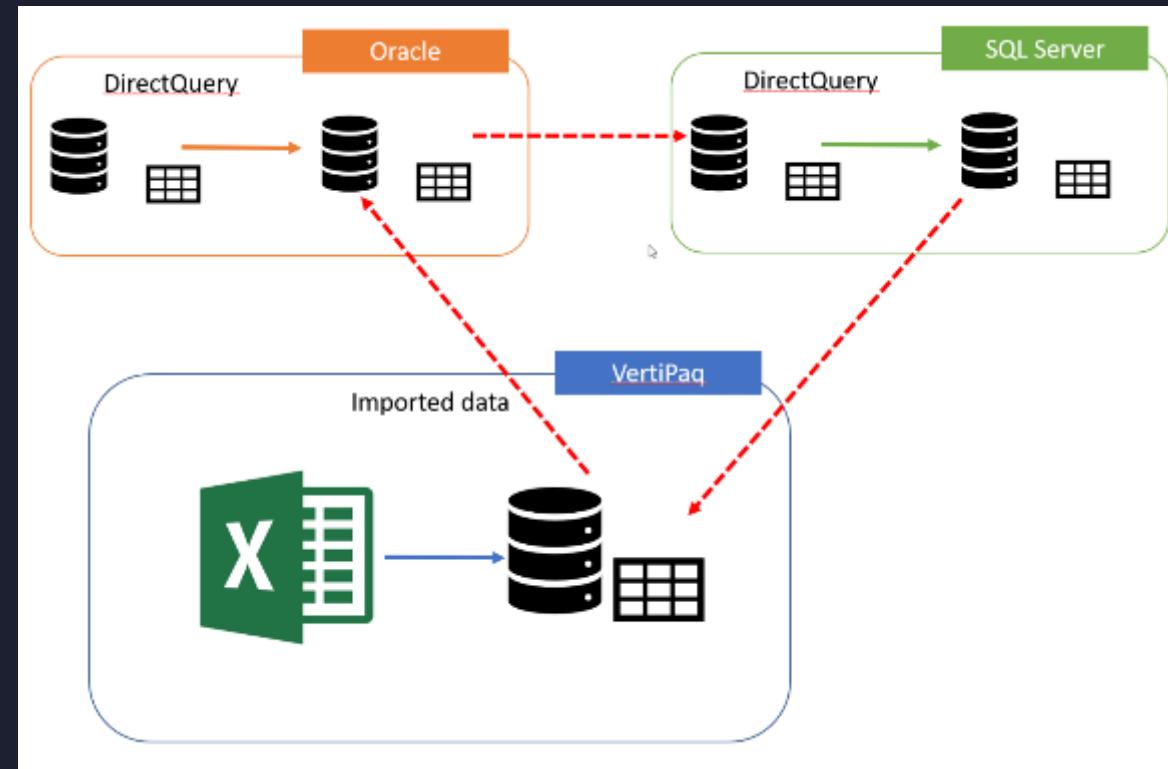


@DataMozart



# 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!

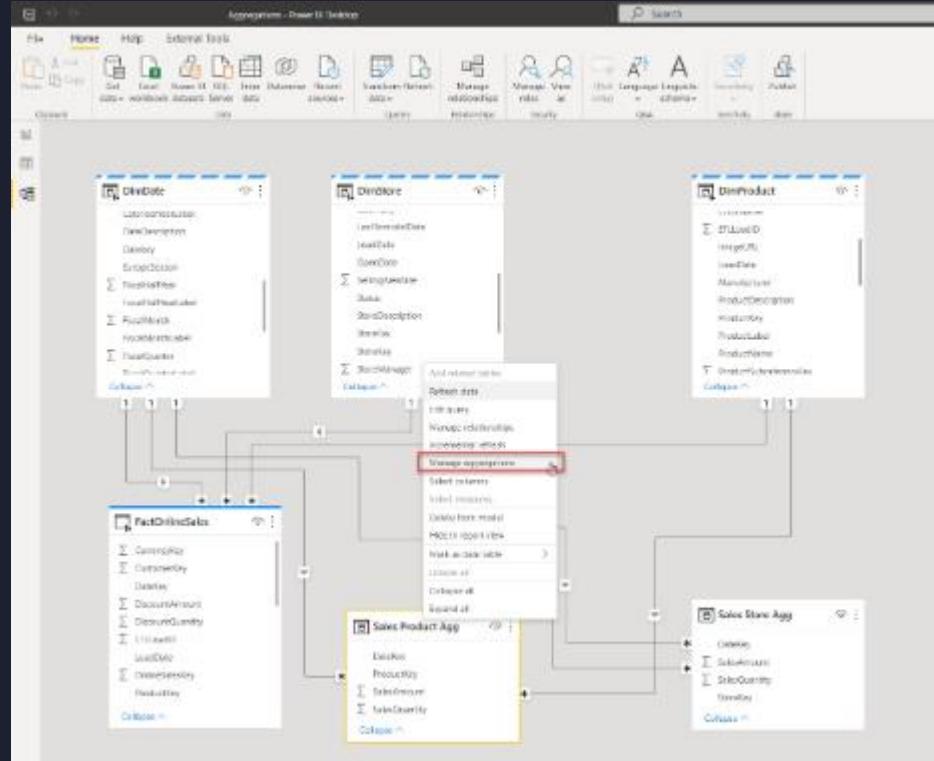


@DataMozart



# 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





## 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



# Who Complains about performance?



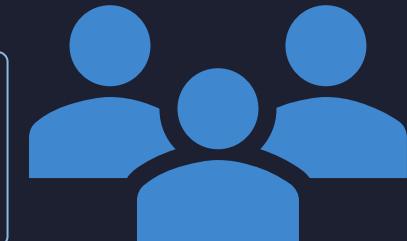
DBA

Background

- Data model size
- Data refresh process

Consumer

Front End



- DAX measures
- Visuals rendering



@DataMozart



*Performance Analyzer captures key metrics of the report performance*



# Performance Analyzer Results



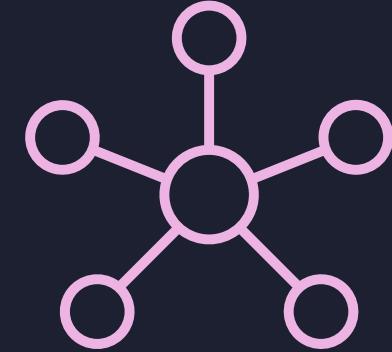
DAX query

Time needed to return  
query results



Visual display

Time elapsed for visual  
rendering



Other

Waiting for other  
processes/queries



@DataMozart



@DataMozart



# Fewer visuals on the page = better performance!



@DataMozart



# Bottlenecks



```
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
```

## Visuals

Speeds up the report  
and removes the clutter

## DAX queries

Learn and understand  
DAX!



@DataMozart



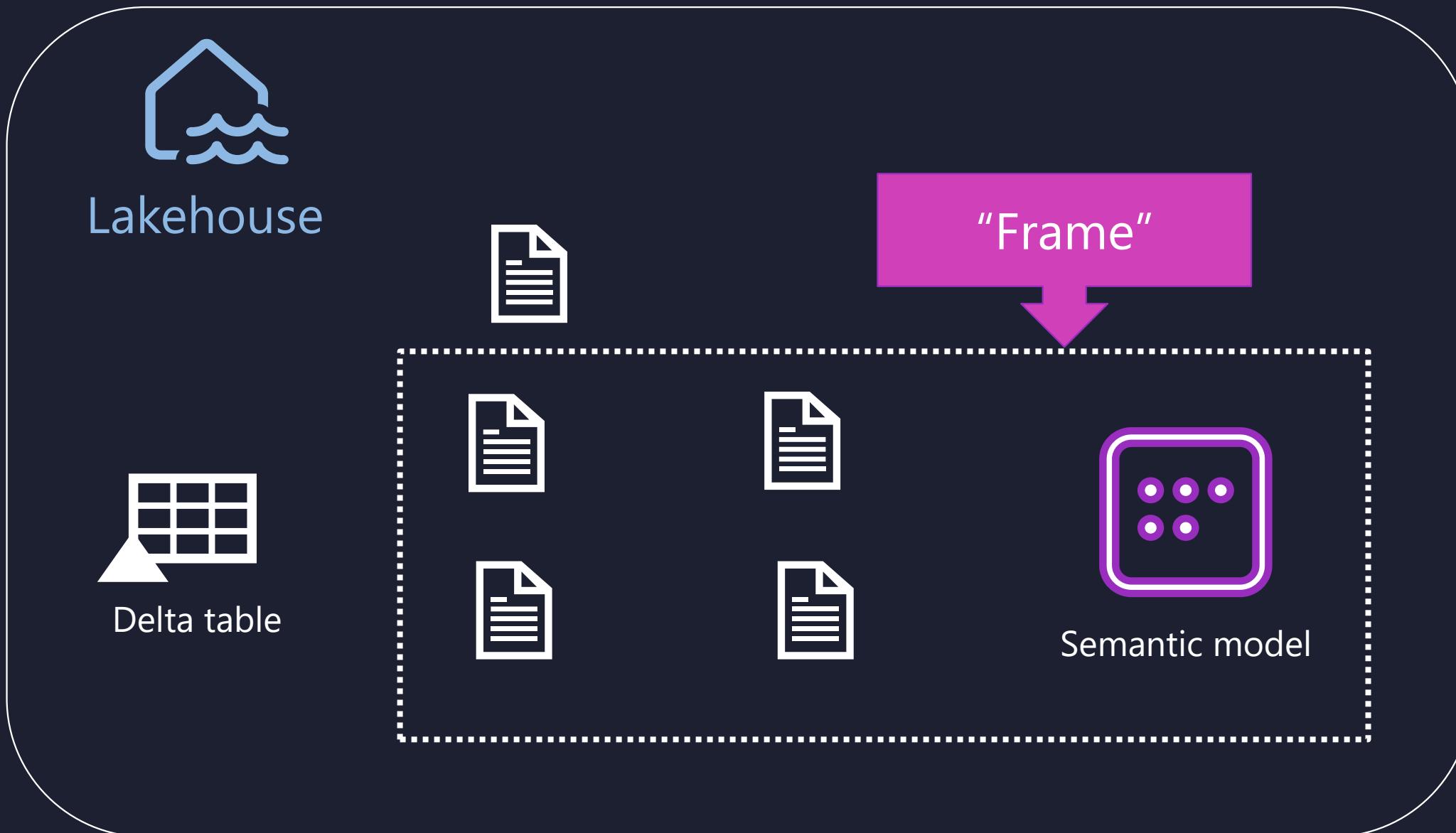
# Direct Lake – Data Refresh

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

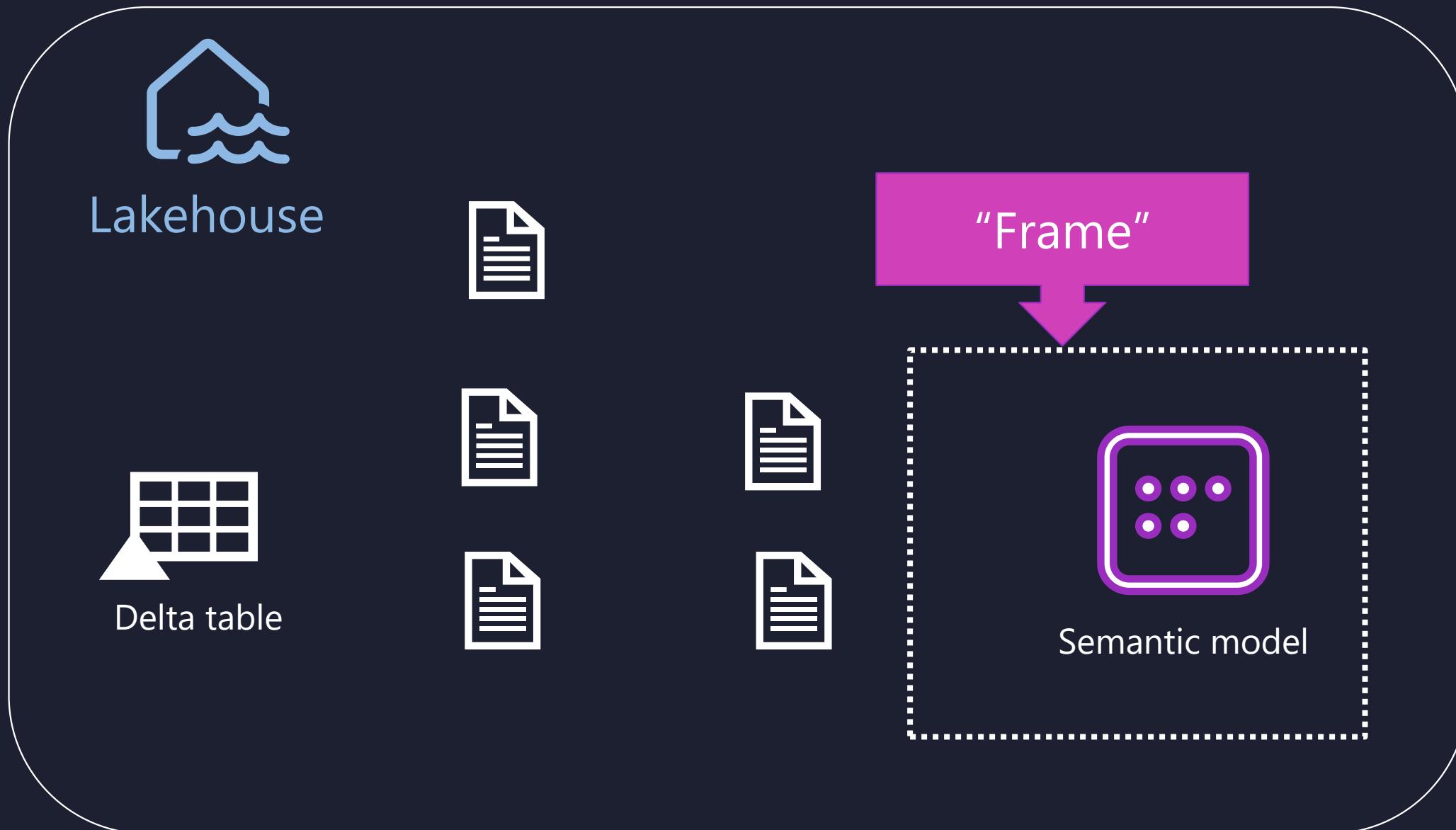


@DataMozart

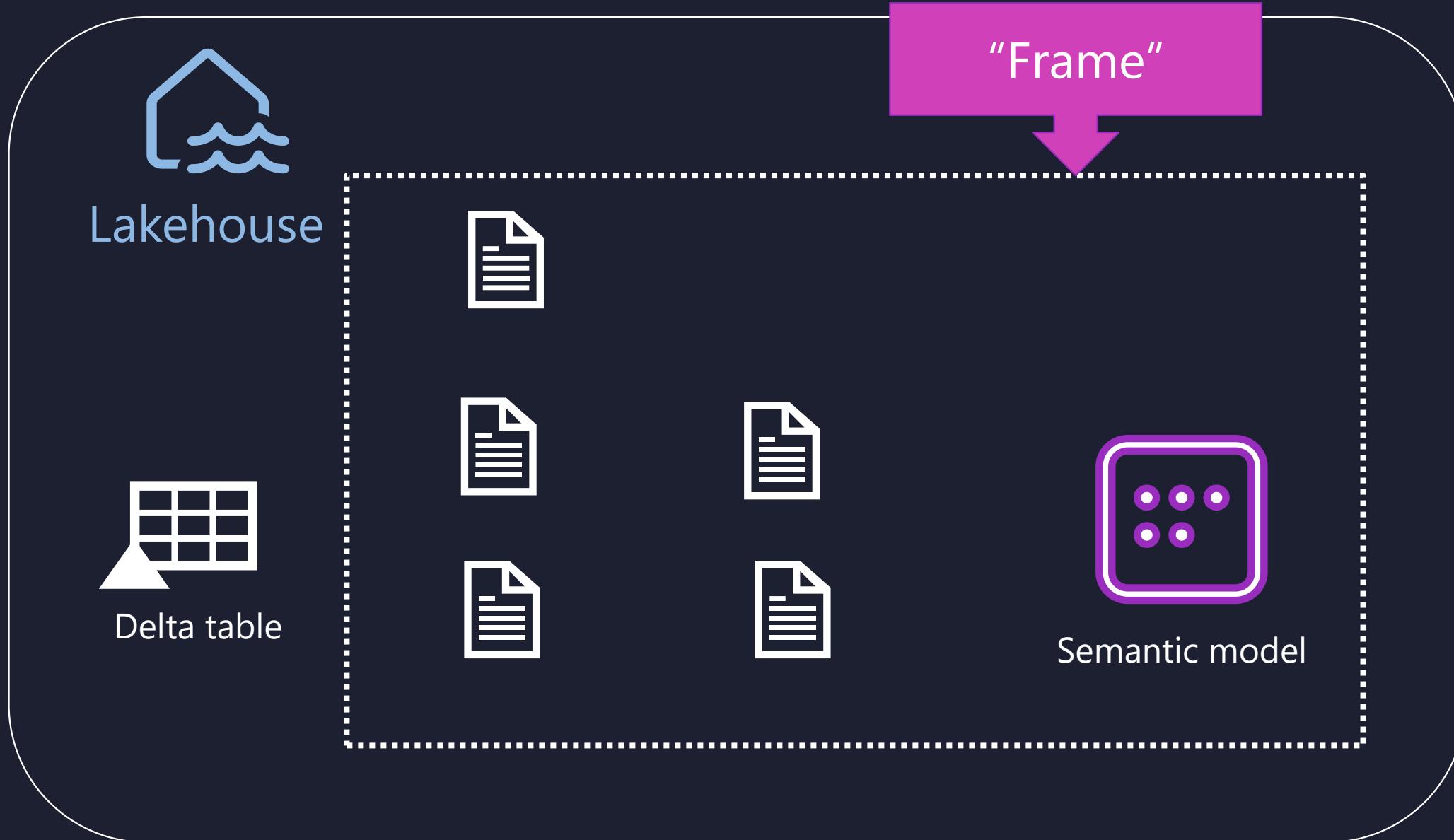
# Direct Lake Refresh (AKA “Framing”)



# Direct Lake Refresh (AKA “Framing”)



# Direct Lake Refresh (AKA “Framing”)





# Refresh option for semantic models

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](#)



@DataMozart



# DirectLakeBehavior Property

Automatic

DQ if DL  
cannot be  
used

DirectLakeOnly

If fallback →  
Error

DirectQueryOnly

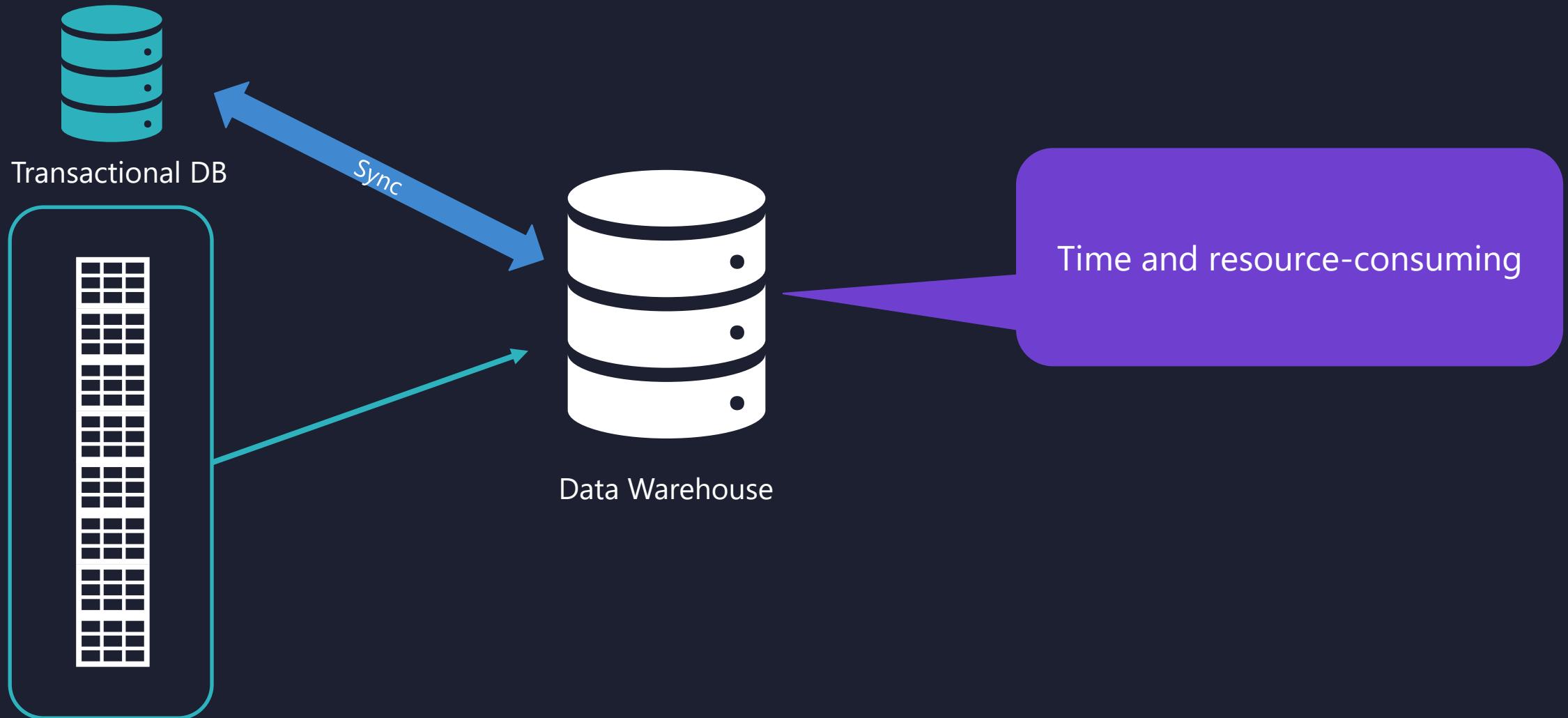
No DL



@DataMozart



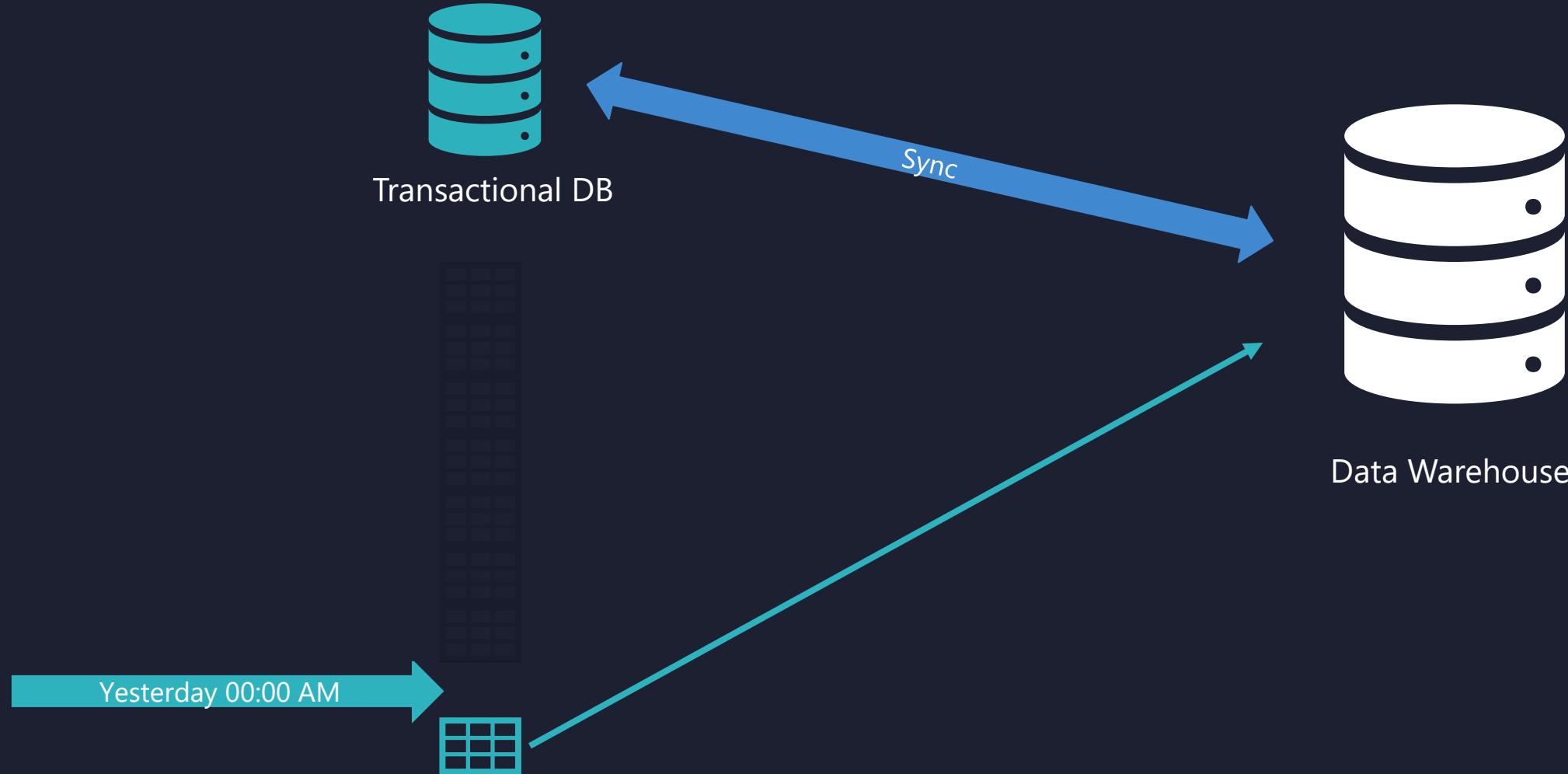
# Why Incremental Refresh?



@DataMozart



# Why Incremental Refresh?

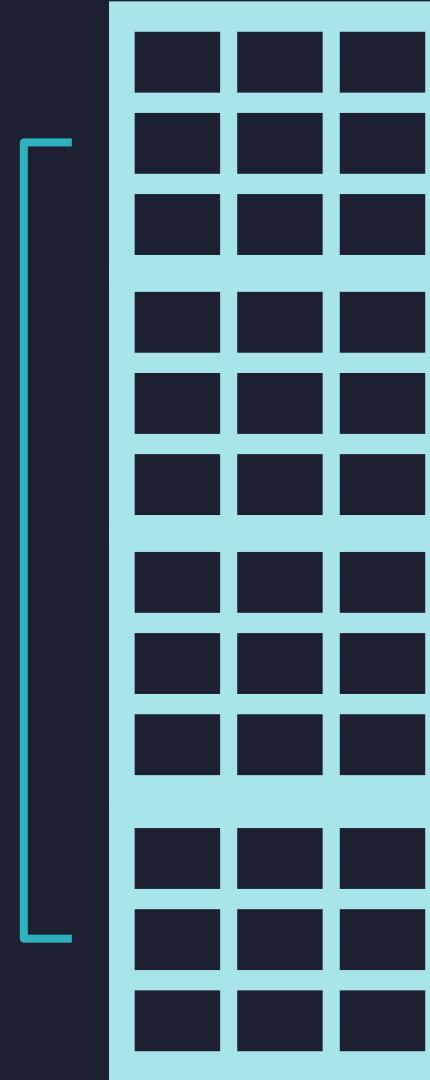


@DataMozart



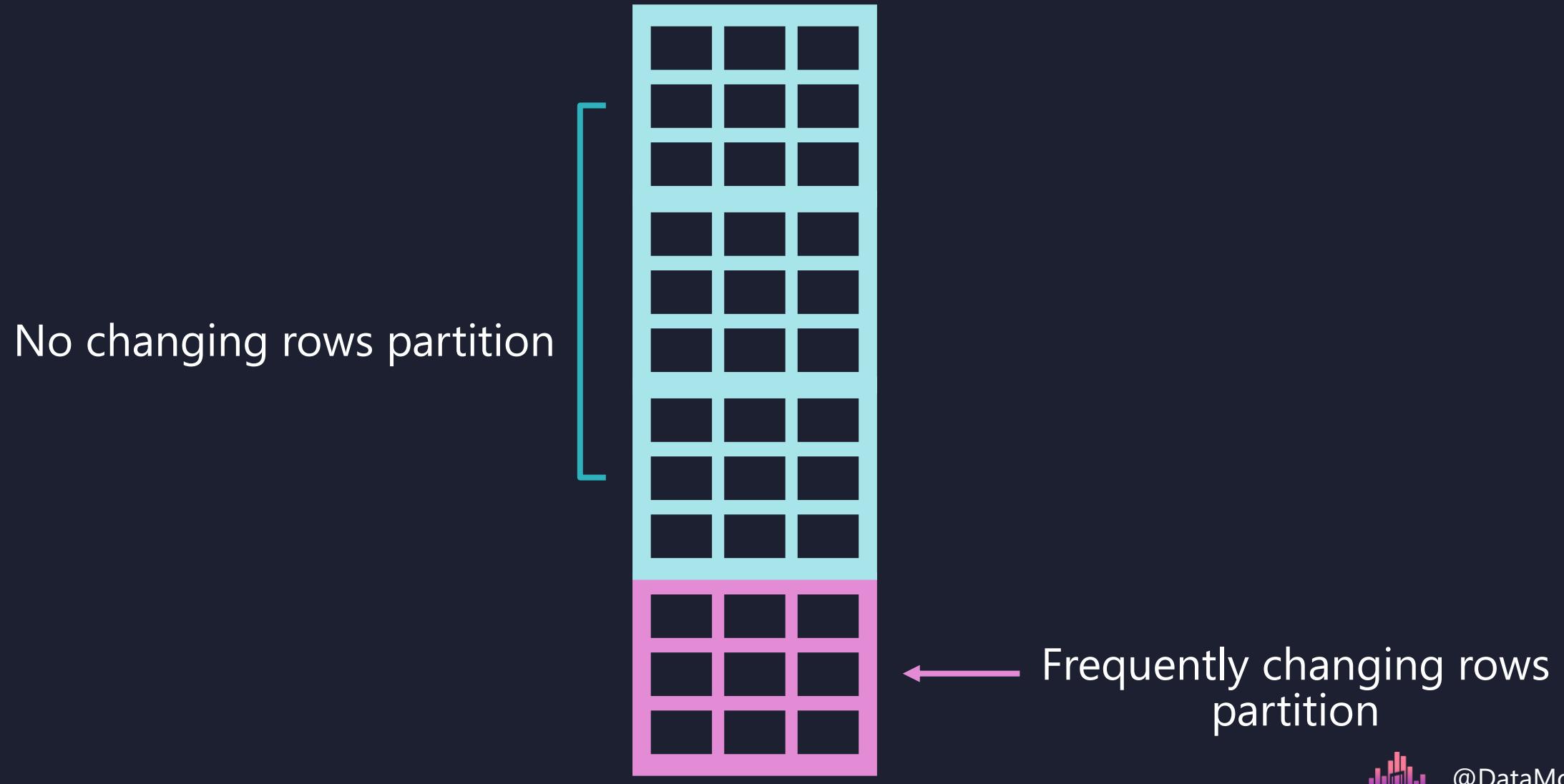
# Incremental Refresh in Power BI

Single partition





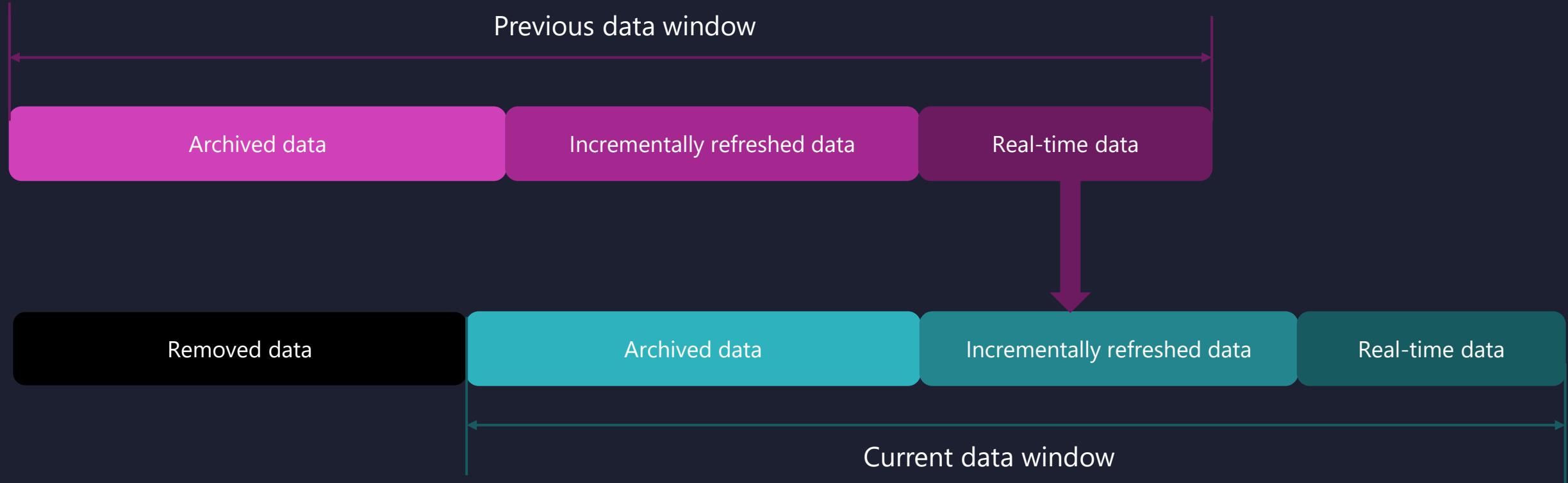
# Incremental Refresh in Power BI



@DataMozart



# Incremental Refresh Workflow



“Rolling Window Pattern”



@DataMozart



# Incremental Refresh Prerequisites

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



@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

## 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



# Let's Play!

# Quiz

# Kahoot.it

Thanks to...



Ricardo Rincón



@DataMozart



# 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 3<sup>rd</sup>/4th, 5 PM CET/11 AM EST/8 AM PST



@DataMozart



# 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](http://www.data-mozart.com)

[www.learn.data-mozart.com](http://www.learn.data-mozart.com)

