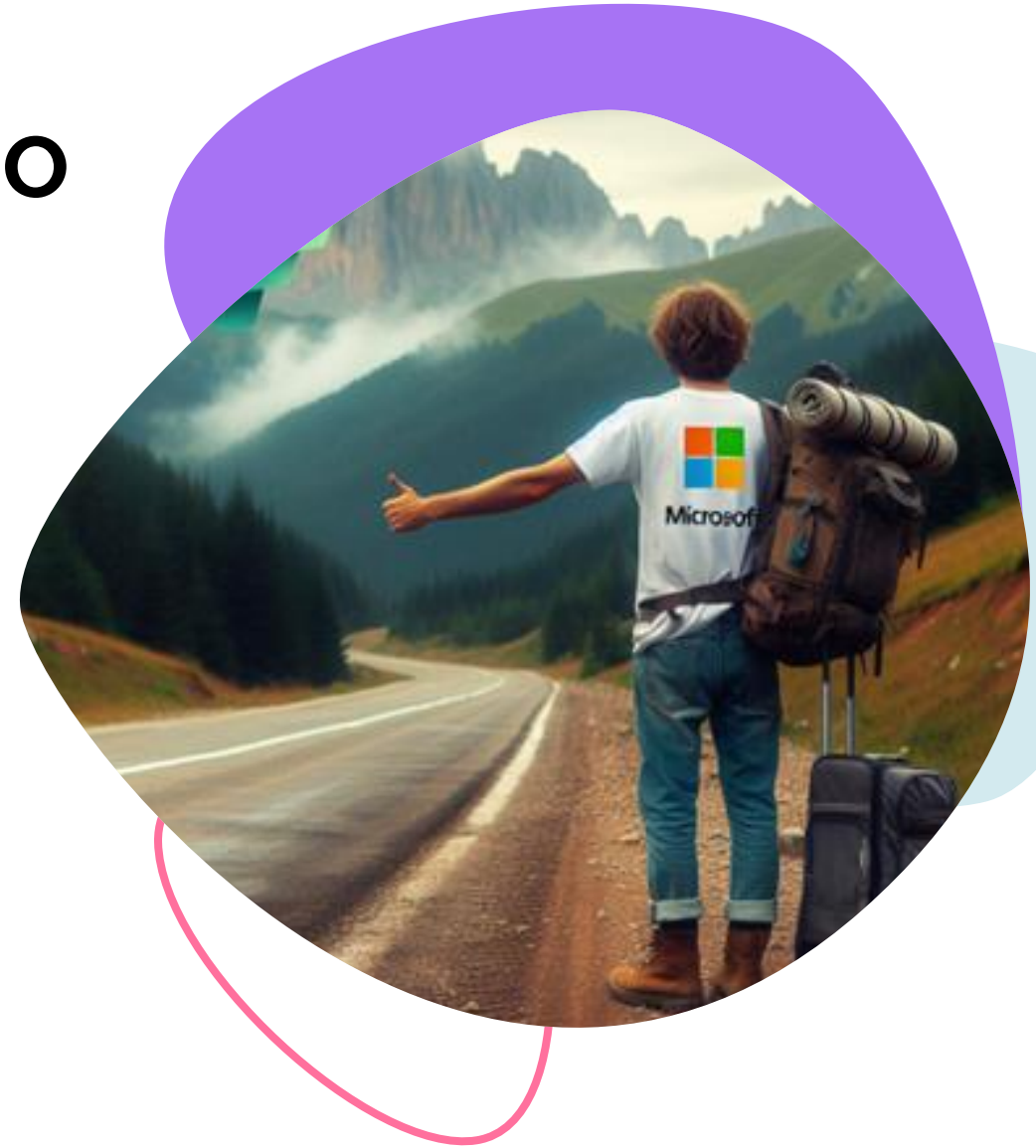
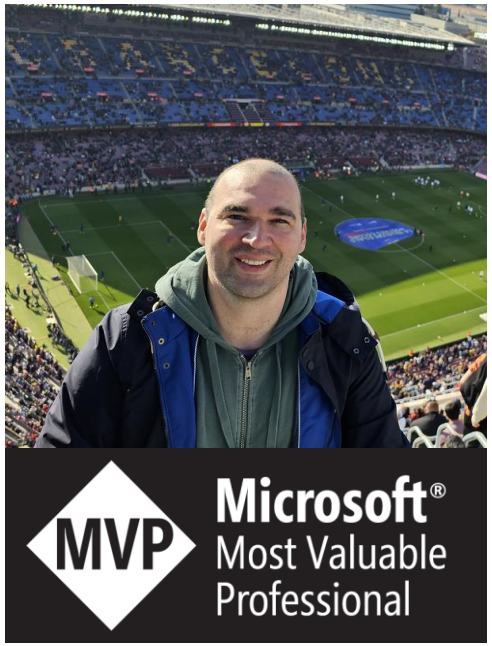
A Hitchhikers Guide to Mastering Fabric Analytics Engineer Certificate (DP-600)



Nikola Ilic

Data Mozart, Microsoft Data Platform MVP



Nikola Ilic

Consultant & Trainer

data-mozart.com

@DataMozart

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…

Agenda

- What is DP-600?
- Skills measured
- Core concepts/features to learn
- Learning resources
- Q&A

What is DP-600?

Microsoft has released a new certification…

Fabric Analytics Engineer Associate

- ✓ One exam: **DP-600** Implementing Analytics Solutions Using Microsoft Fabric
- ✓ If you work in a space involving **Data Analysis** and **Data Engineering**, then this certification is for you.
- ✓ After this session, you'll understand which features and services in Microsoft Fabric you need to know when approaching the DP-600 exam with confidence.

Candidate Profile

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

Responsibilities include

- Cleaning and transforming data
- Building enterprise semantic models
- Incorporating advanced analytics
- Applying development lifecycles

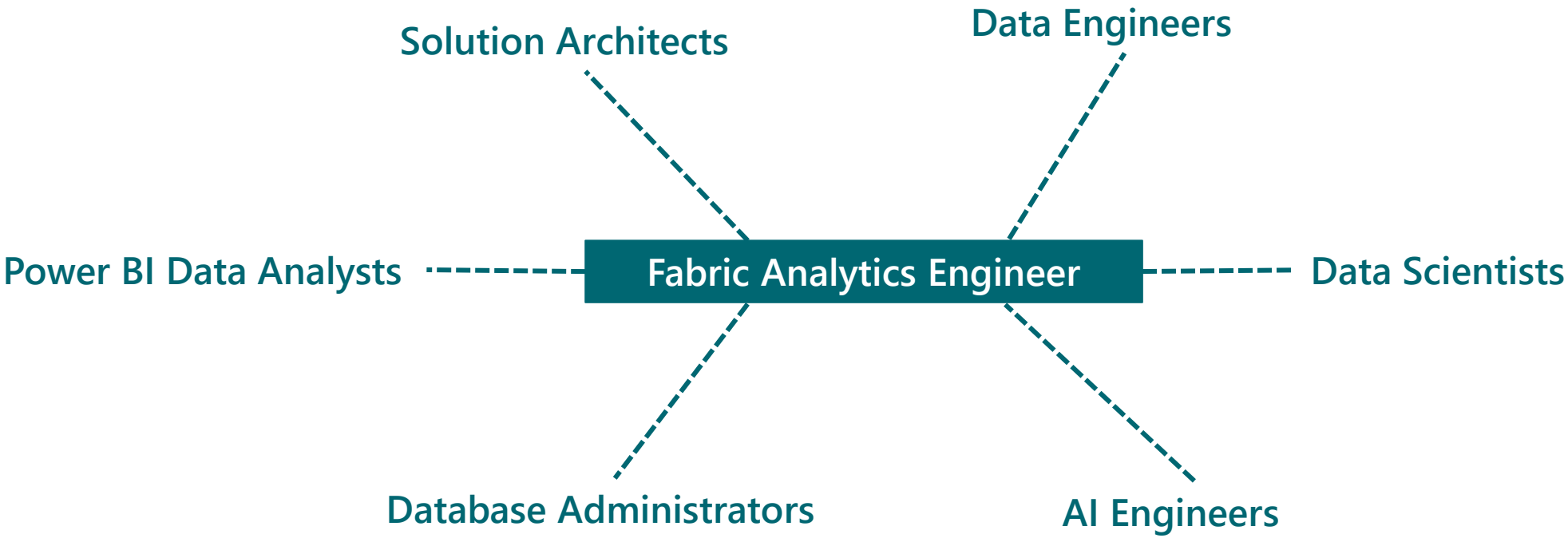
These professionals help

- 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 & PySpark
- Data modeling

Organisational Role Collaboration



Skills Measured

Skills Measured Breakdown

- ✓ Plan, implement and manage a solution for data analytics (10–15%)
- ✓ Prepare and serve data (40-45%)
- ✓ Implement and manage semantic models (20–25%)
- ✓ Explore and analyze data (20–25%)

57

Total skills measured

Session Content Breakdown

Data Engineering

Power BI

Data Warehouse

Data Factory

Session Content Breakdown

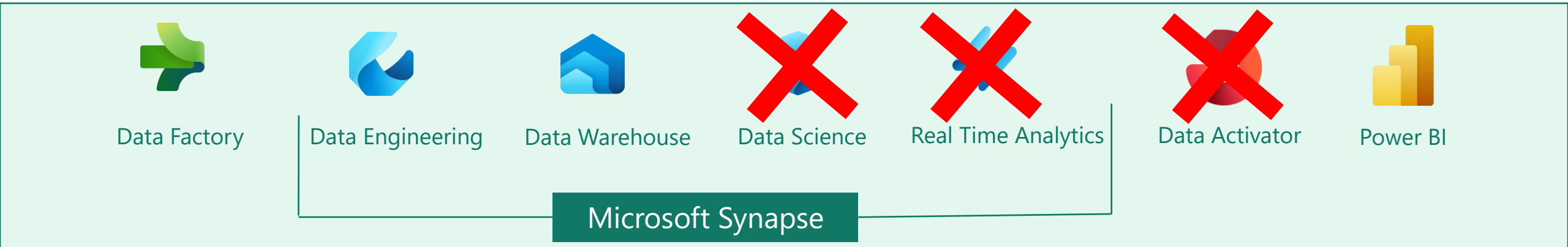
Prepare and serve data

Implement and manage semantic models

Explore and analyze data

Core Concepts/Features to Learn

“Players” in Microsoft Fabric



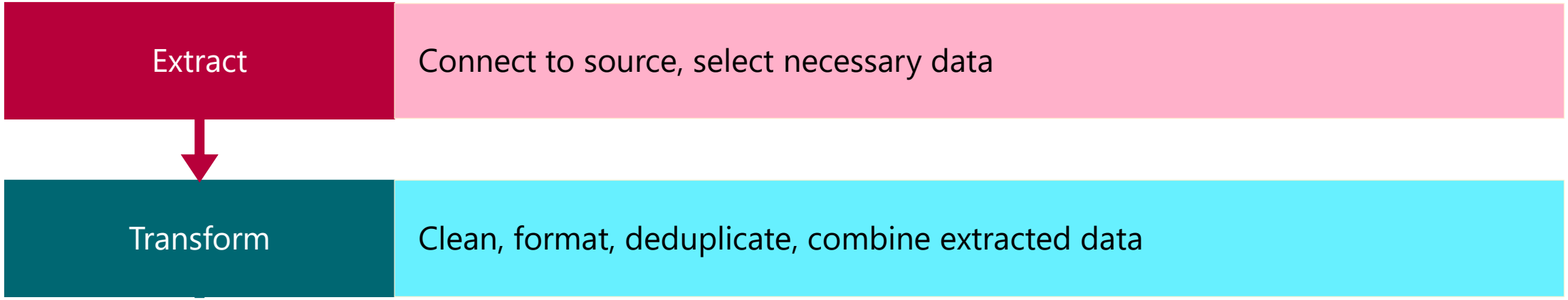
OneLake

Prepare and serve data

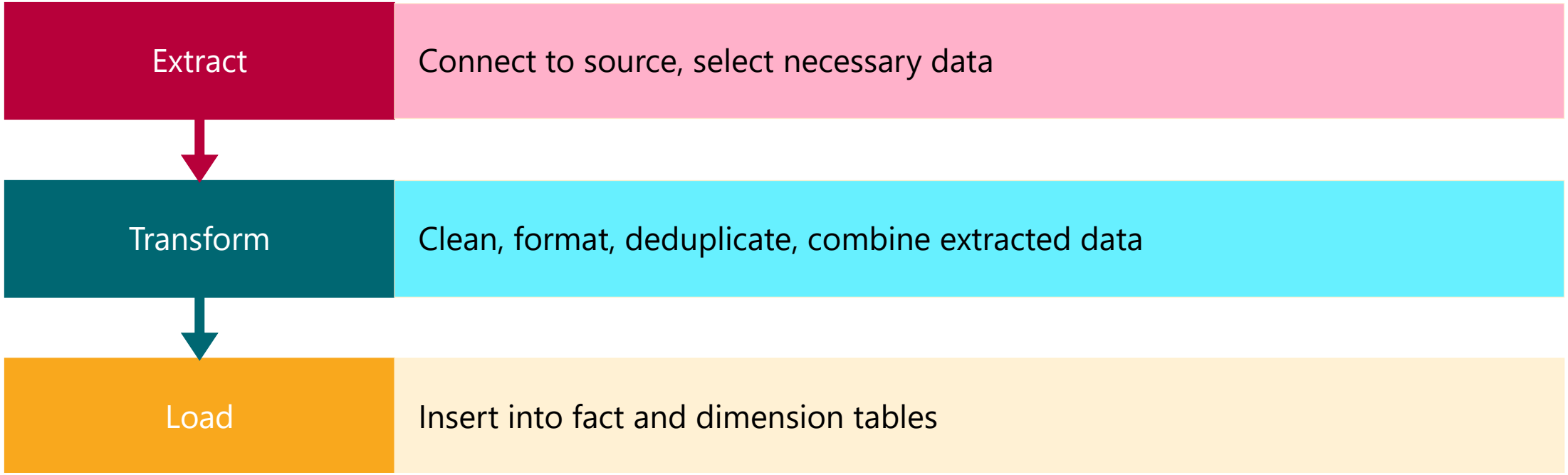
Understand ETL (Extract, Transform and Load)

Extract	Connect to source, select necessary data

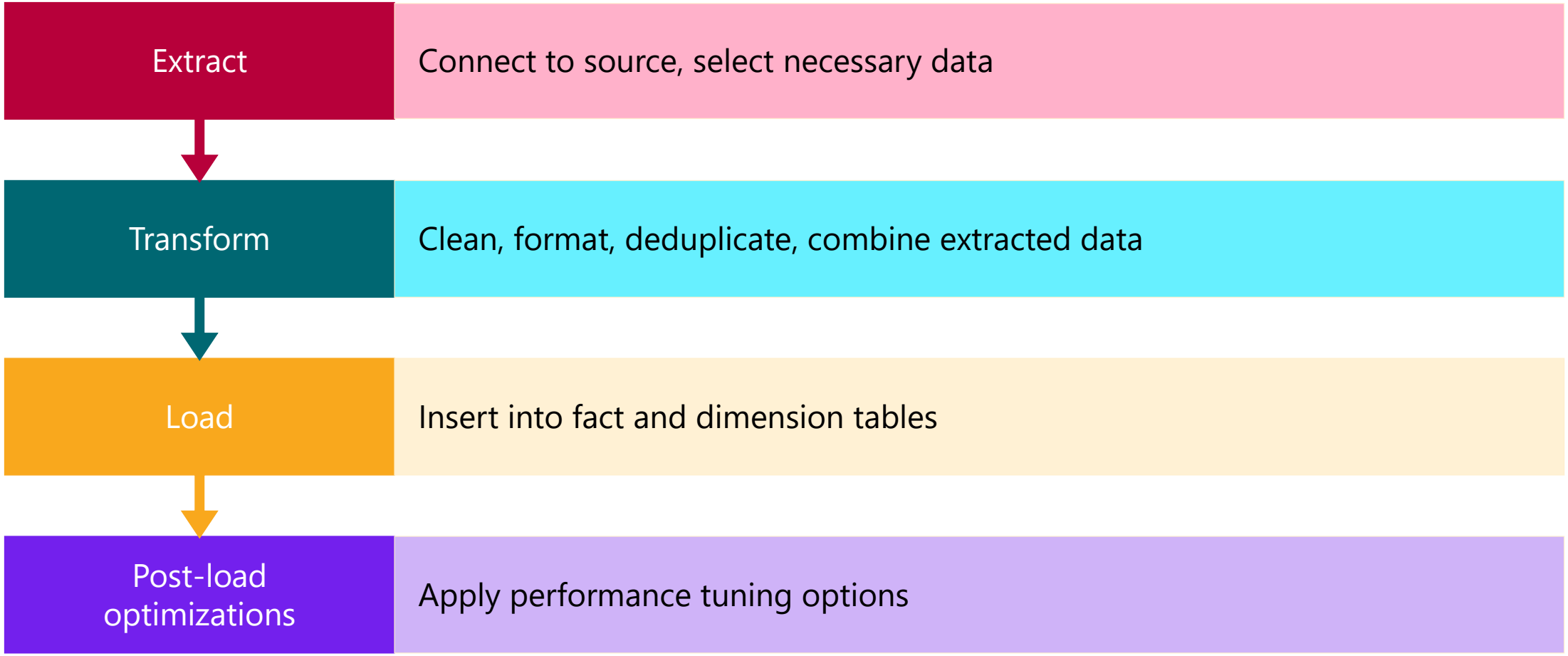
Understand ETL (Extract, Transform and Load)



Understand ETL (Extract, Transform and Load)



Understand ETL (Extract, Transform and Load)



Data Loading Process

Full loading

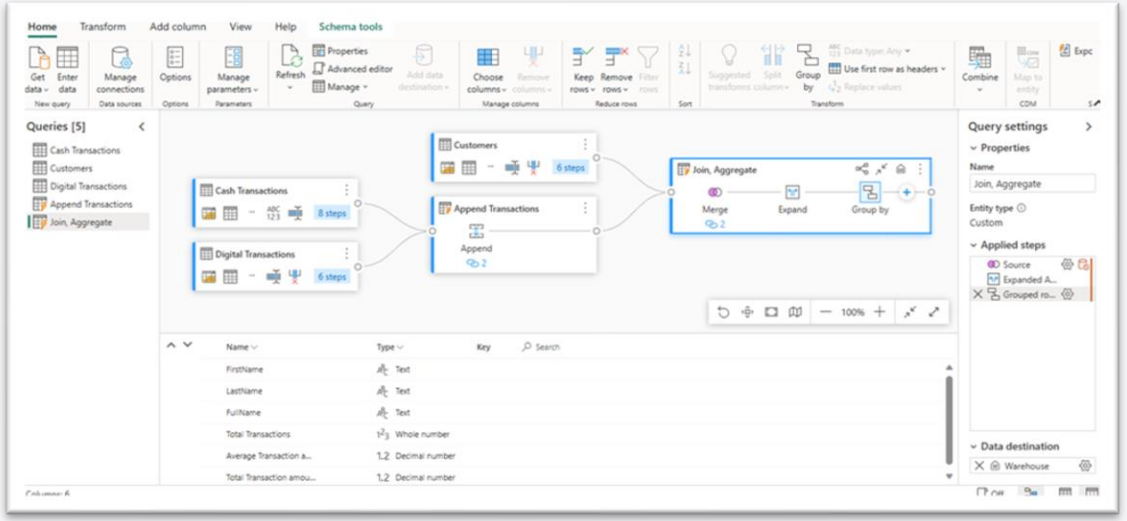
- ✓Truncate table and load data
- ✓Longer load time
- ✓No history stored
- ✓Best for initial load or refresh

Incremental loading

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

Dataflows (Gen2)

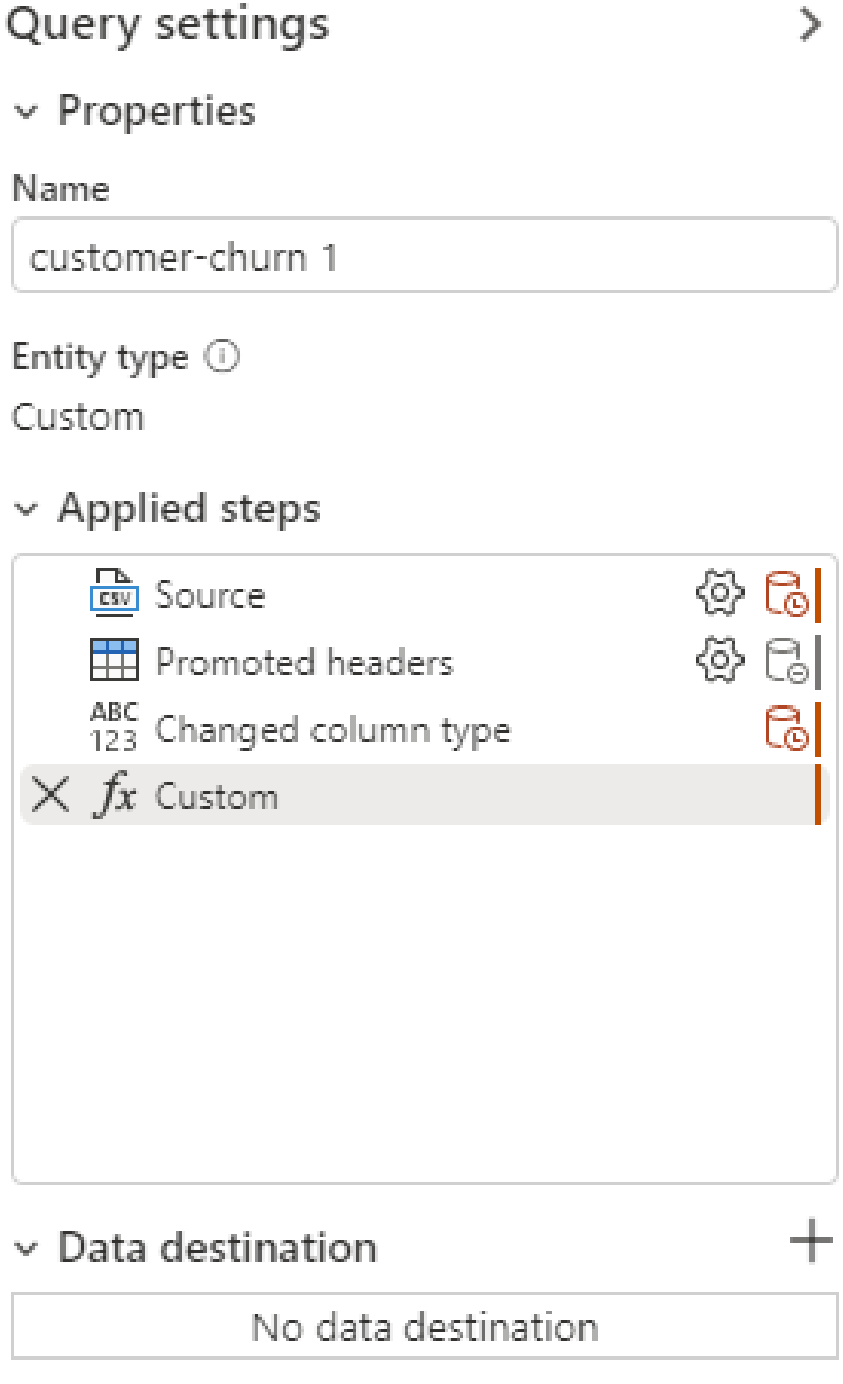
- Low-code graphical environment for defining ETL solutions
- Extract data from multiple sources, transform it, and load it into a destination
- Run dataflows independently or as an activity in a Pipeline



Load data with Dataflows Gen2

Ingest and transform with Power Query

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



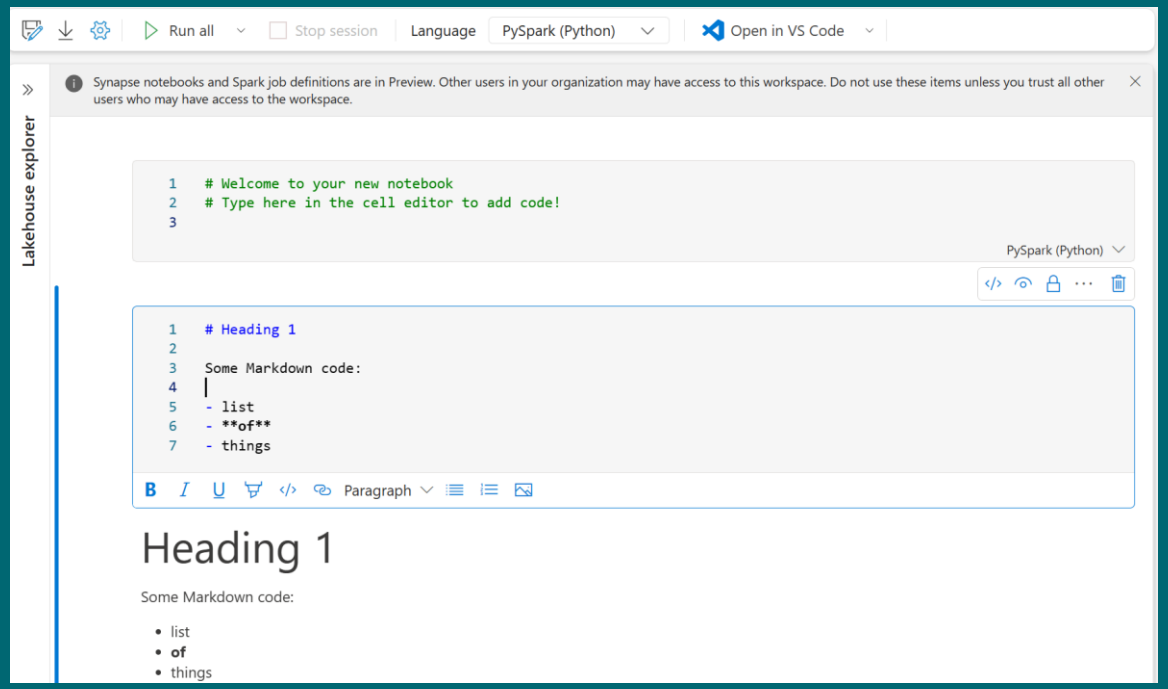
Dataflow Gen2 Pros and Cons

- Allow self-service users access to a subset of data warehouse separately
- Optimize performance – extract once, reuse multiple times
- Reduce complexity – expose only dataflows to larger analyst groups
- Ensure consistency and quality of data by enabling users to clean and transform data before loading it to a destination

- Row-level security isn't supported
- Fabric capacity workspace is required

Fabric Notebooks

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



Connect to external data sources

- Fabric shortcuts offer easy connections to external sources
- When not available, notebooks can programmatically connect to and ingest data from an **external** source

```
# Azure Blob Storage access info

blob_account_name = "azureopendatastorage"
blob_container_name = "nyctlc"
blob_relative_path = "yellow"
blob_sas_token = “<yourSAStoken>"

# Construct the path for connection

wasbs_path =
f'wasbs://{blob_container_name}@{blob_account_n
ame}.blob.core.windows.net/{blob_relative_path}
?{blob_sas_token}'

# Read parquet data from Azure Blob Storage
path

blob_df = spark.read.parquet(wasbs_path)

# Show the Azure Blob DataFrame
blob_df.show()
```

Write data in a lakehouse

Write to a Parquet file

Parquet format allows the lakehouse to distribute and optimize performance in the Spark engine

```
# Write DataFrame to Parquet file format

parquet_output_path =
"dbfs:/FileStore/your_folder/your_file_name"
df.write.mode("overwrite").parquet(parquet_o
utput_path)
print(f"DataFrame has been written to
Parquet file: {parquet_output_path}")

# Write DataFrame to Delta table

delta_table_name = "your_delta_table_name"
df.write.format("delta").mode("overwrite").s
aveAsTable(delta_table_name)
print(f"DataFrame has been written to Delta
table: {delta_table_name}")
```

Write to a Delta table

Optimize Delta table writes

- Use **Delta format** for durability and scale.
- Optimize read and write with **V-Order** and **optimized write** options.

```
# Use format and save to load as a Delta
table
table_name = "nyctaxi_raw"
filtered_df.write.mode("overwrite").format("
delta").save(f"Tables/{table_name}")

# Confirm load as Delta table
print(f"Spark DataFrame saved to Delta
table: {table_name}")

# Enable V-Order
spark.conf.set("spark.sql.parquet.vorder.ena
bled", "true")

# Enable automatic Delta optimized write
spark.conf.set("spark.microsoft.delta.optimi
zeWrite.enabled", "true")
```

Load data with T-SQL

Use COPY statement

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

```
-- Load as CSV
COPY my_table
FROM
'https://myaccount.blob.core.windows.net/myblobcontainer/folder0/*.csv,
https://myaccount.blob.core.windows.net/myblobcontainer/folder1/'
WITH (FILE_TYPE = 'CSV',
    CREDENTIAL=(IDENTITY= 'Shared Access Signature',
SECRET='<Your_SAS_Token>')
    FIELDTERMINATOR = '|’)

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

Load from other items

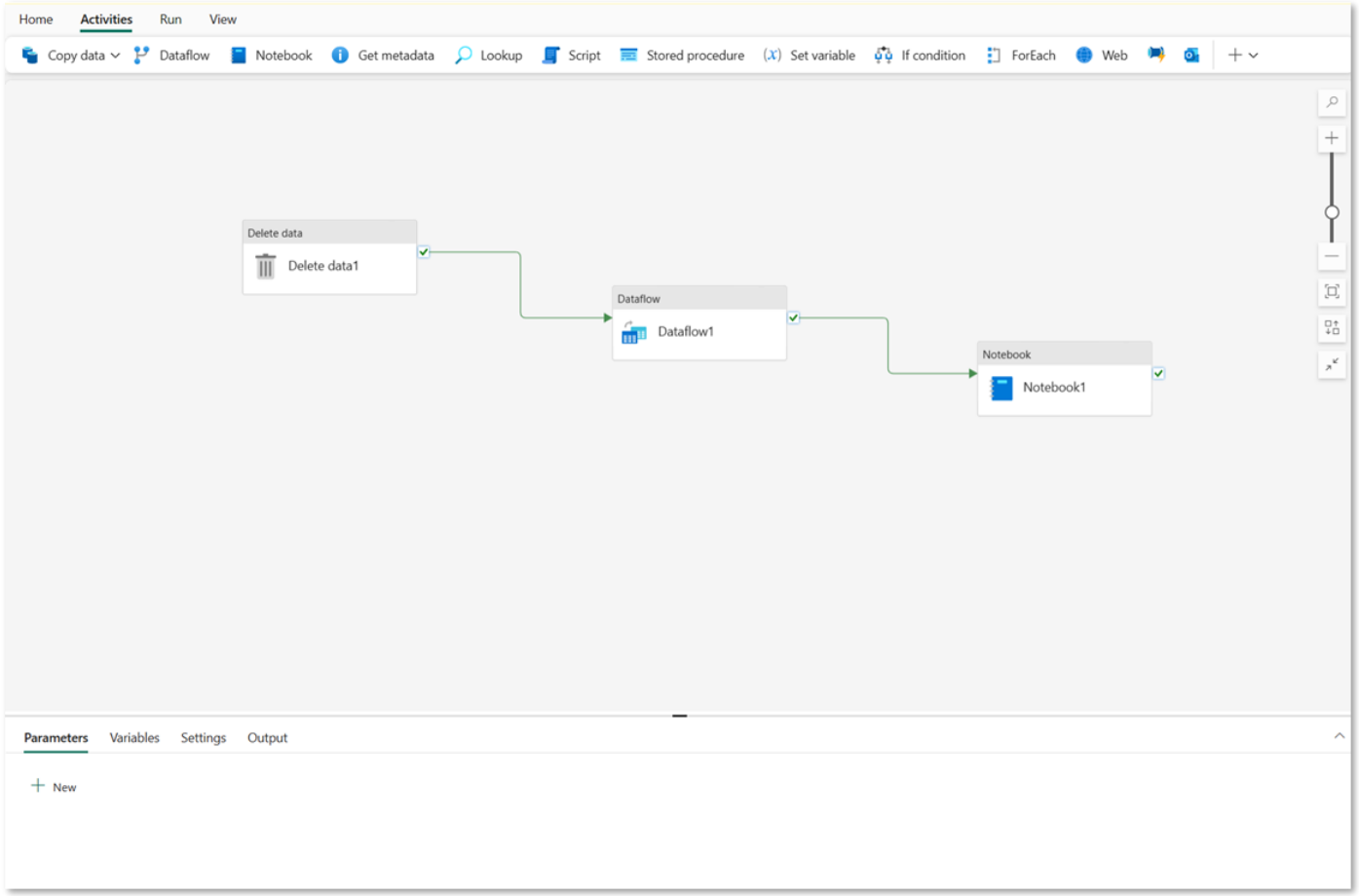
Combine into one warehouse

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

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

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

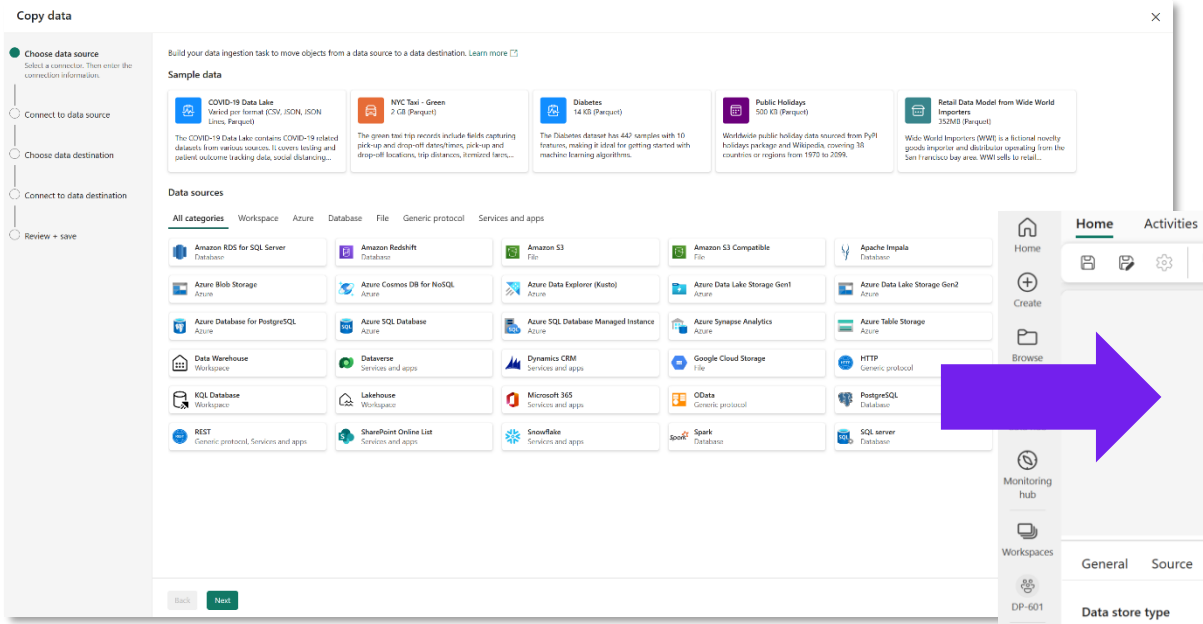
Fabric Pipelines



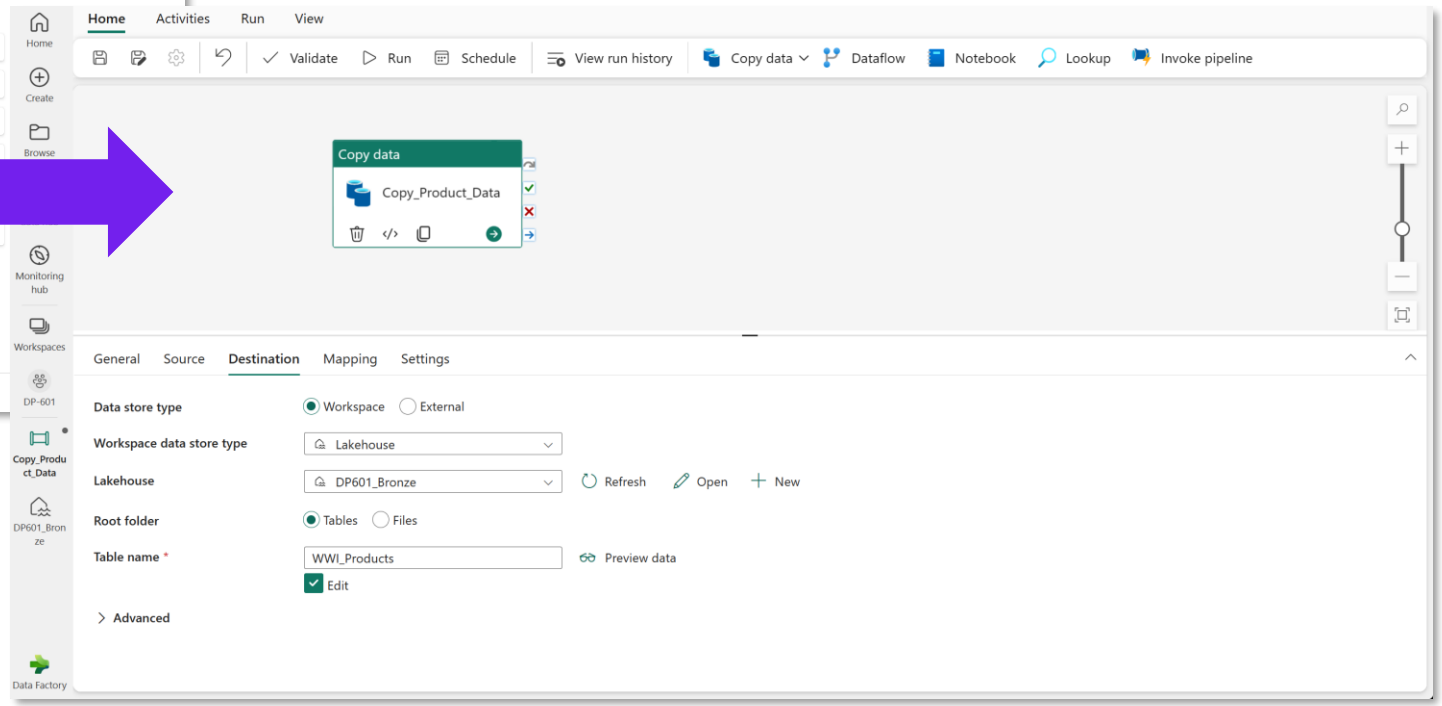
Pipeline concepts:

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

Common Activities – Copy Data



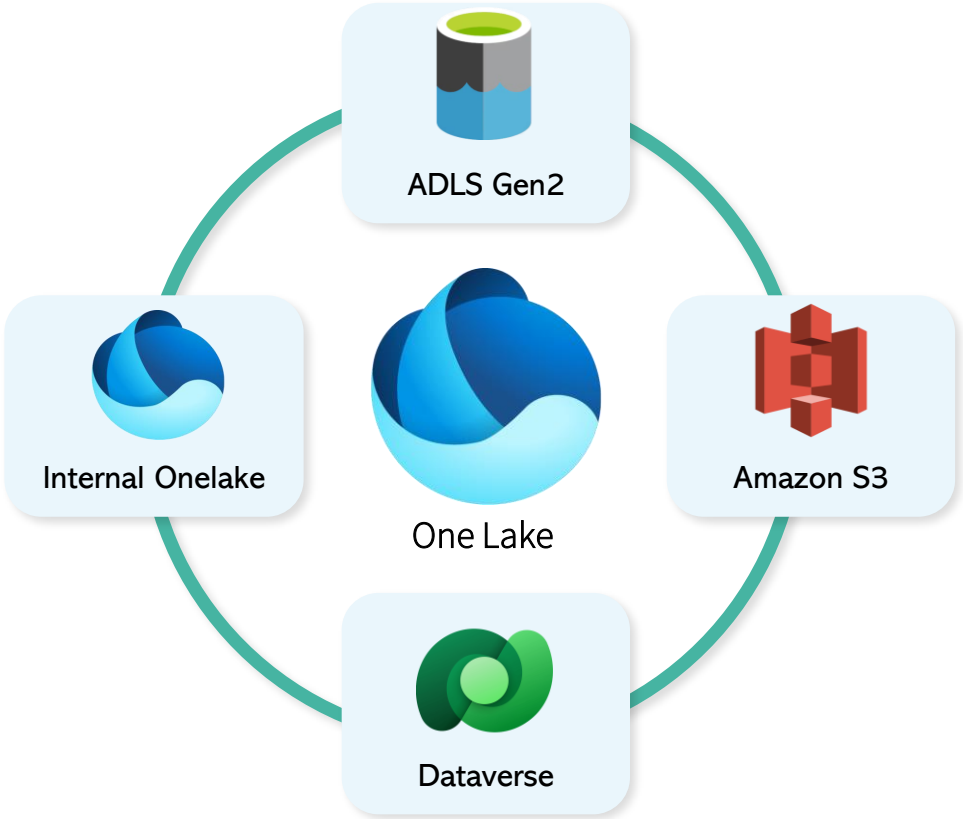
1. Use the copy data tool



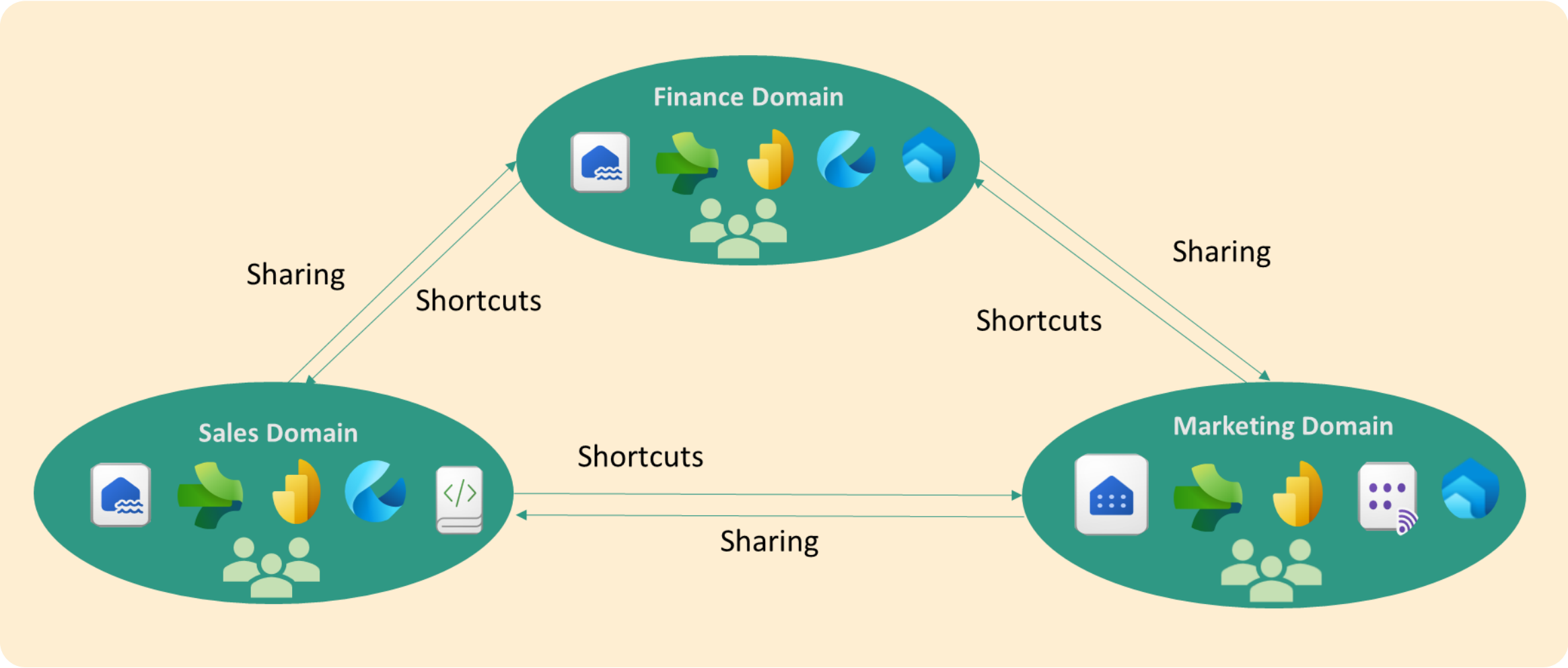
2. Edit the settings below the pipeline canvas

OneLake Shortcuts

Shortcuts –
Like Windows Explorer?



Life is easier with Shortcuts



Benefits of using Shortcuts

- OneLake for all Domains
- Seamlessly query multiple data sources
- Data sharing capabilities made easy
- Unlocking the potential of Data Mesh
- Effective governance and data security

Organizing the Fabric Lakehouse

Bronze

Raw

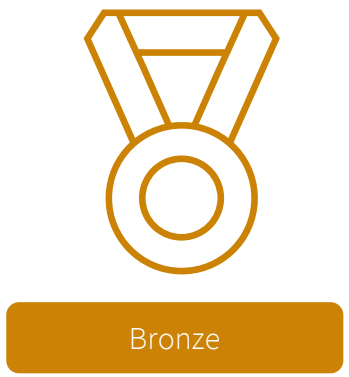
Silver

Validated

Gold

Enriched/Curated

Medallion Architecture Fundamentals

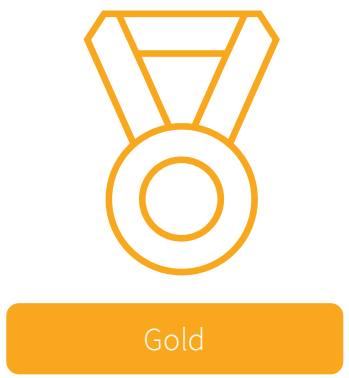


- ✓ Land data from external sources in its original state
- ✓ Serve as a repository of the historical archive of source data
- ✓ Contains unvalidated data
- ✓ Stores the data in Parquet/Delta

Medallion Architecture Fundamentals

- ✓ 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

Medallion Architecture Fundamentals



- ✓ 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, alternatively in Parquet

Organizing the Fabric Lakehouse

- How much data are you working with?
- How complex are the transformations you need to make?
- How often will you need to move data between layers?
- What tools are you most comfortable with?

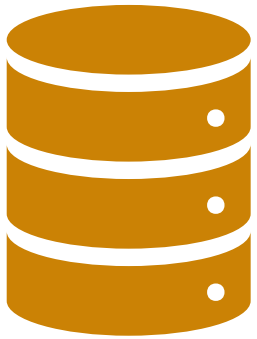
Point to remember!

Dataflows transform data, pipelines orchestrate data, notebooks can do both

Implement a Medallion Architecture in Fabric

Bronze

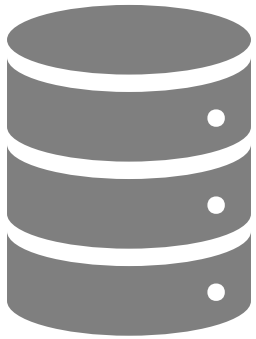
Ingest raw data



Pipelines, dataflows, notebooks

Silver

Cleanse and validate data



Dataflows or notebooks

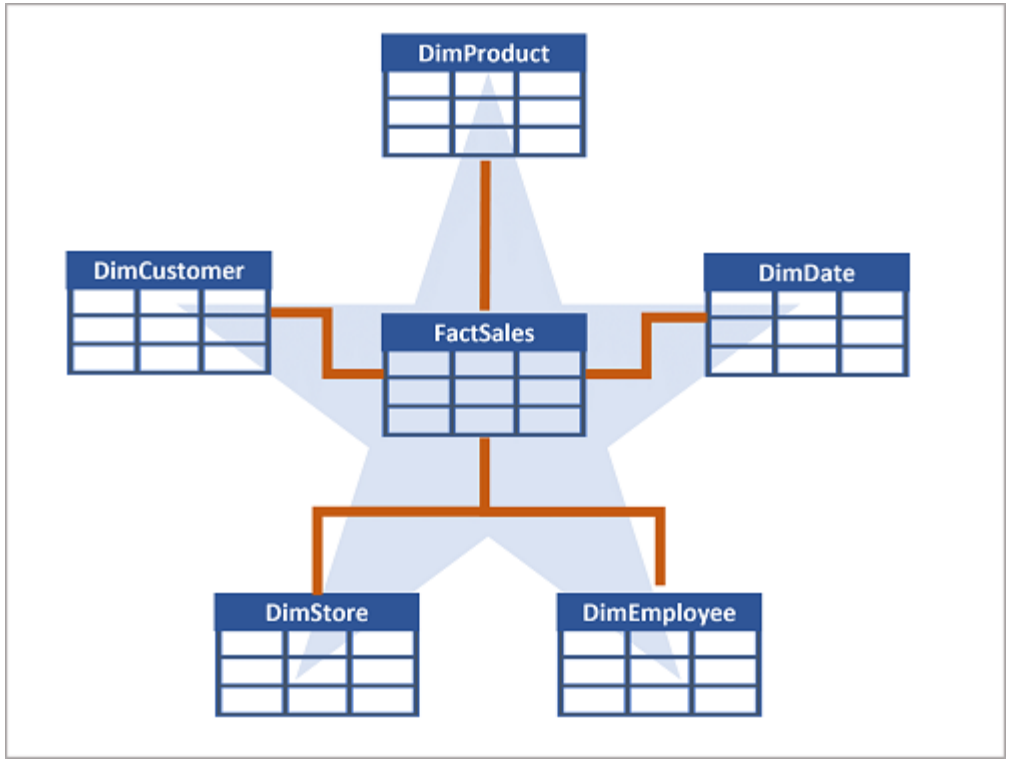
Gold

Additional transformations and modeling

SQL analytics endpoint or semantic model

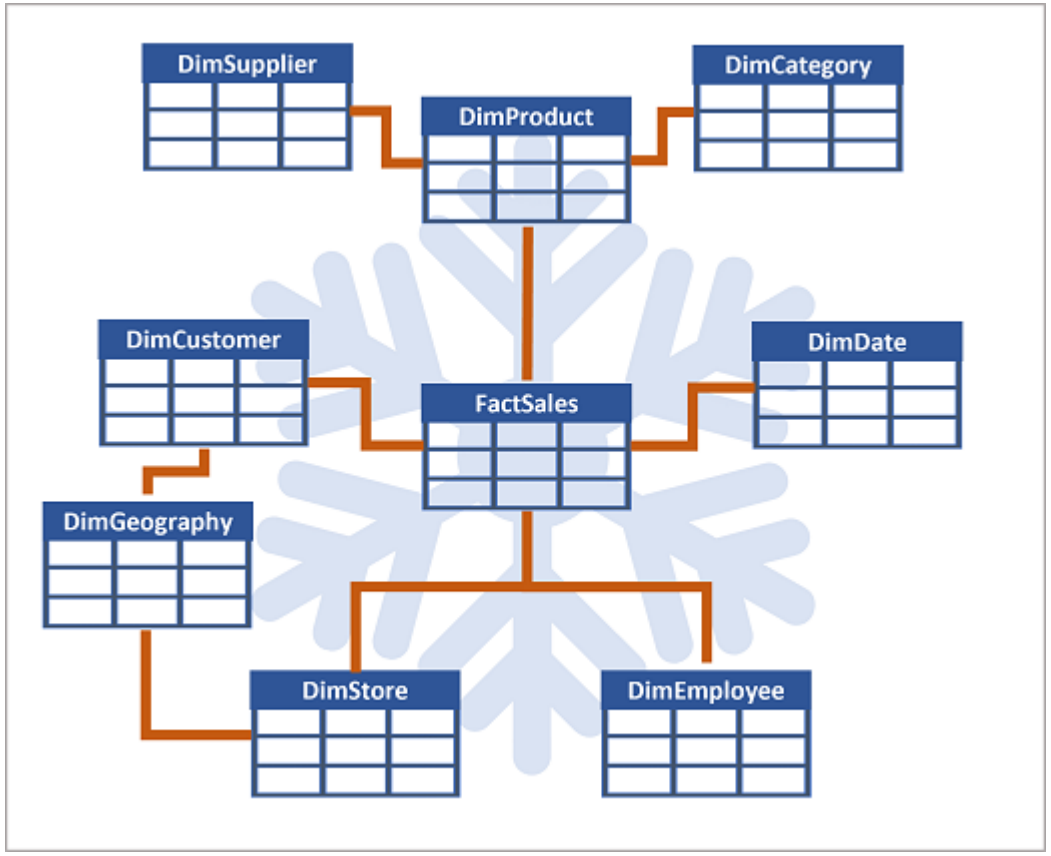
Dimensional Modeling Fundamentals

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



Dimensional Modeling Beyond Star Schema

- Star schema, further denormalized
- 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

Other Concepts/Features to Learn…

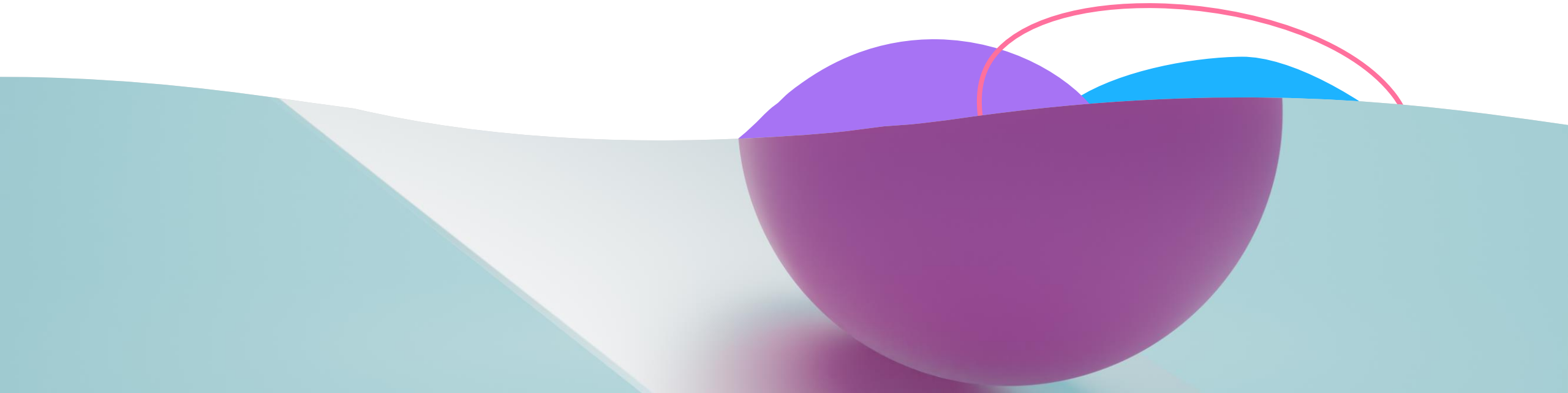
Data modeling/transformation

- Bridge tables
- Denormalization
- Merging/joining
- Data cleansing

Performance optimization

- Identify bottlenecks
- Resolve performance issues
  - Query folding

Implement and manage semantic models



Direct Lake for Power BI

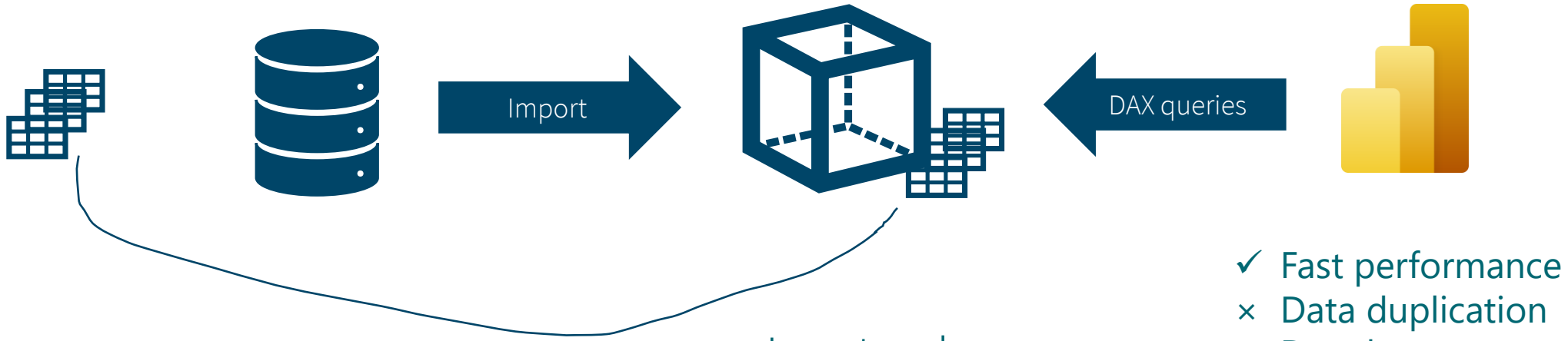
Revolutionary feature!

✓ Prerequisites

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

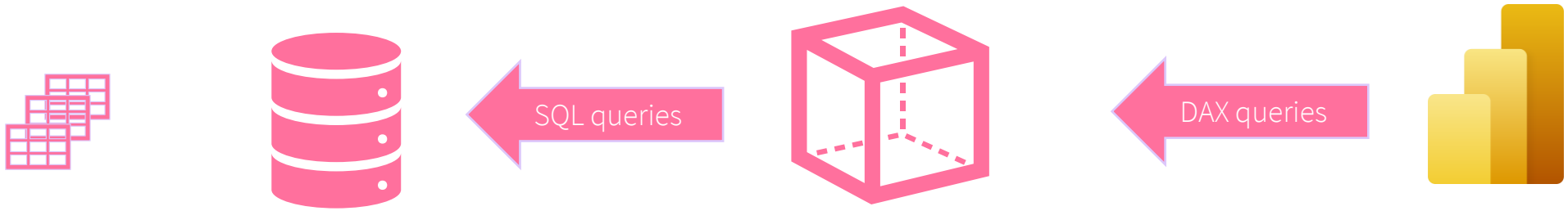
* V-Ordering
Fabric-specific way of additionally optimizing Parquet files when writing data

Power BI Architecture – Pre-Fabric



Import mode

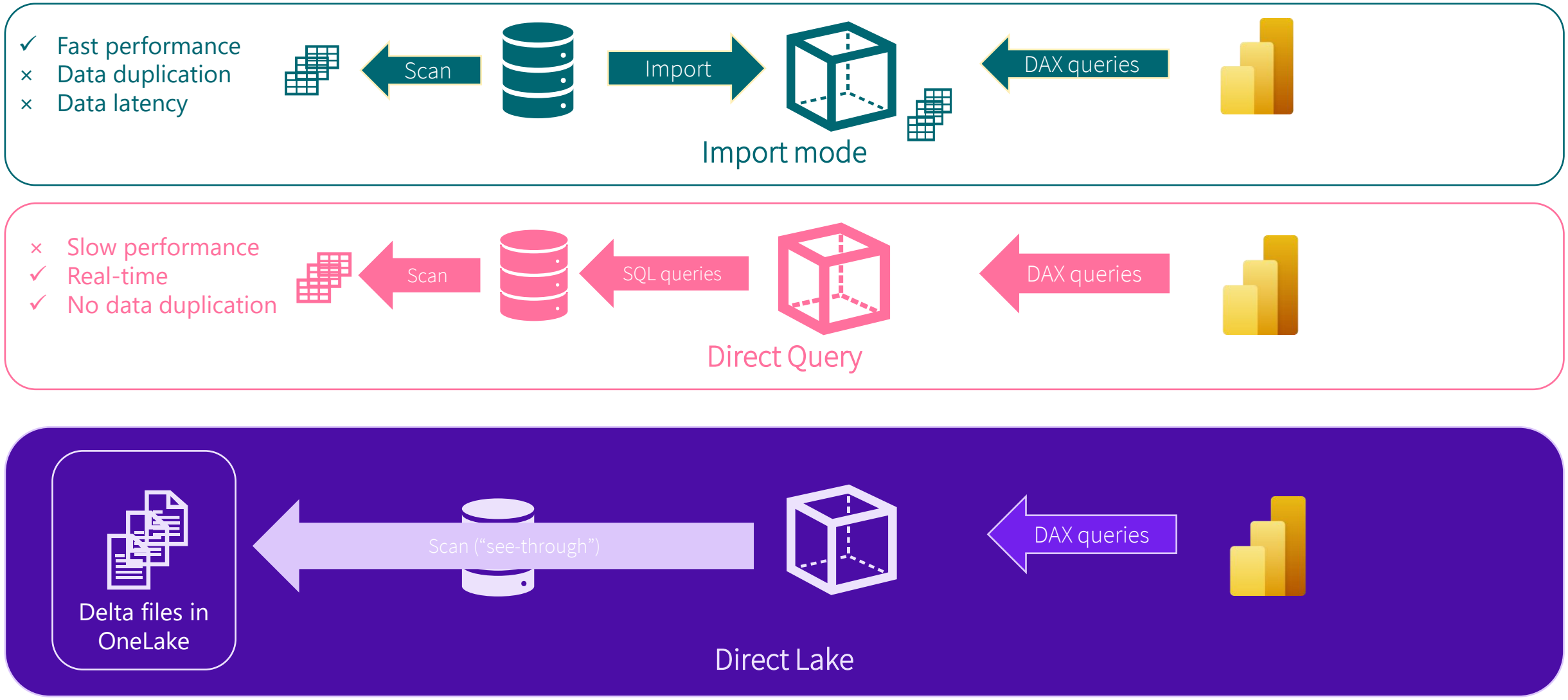
- ✓ Fast performance
- × Data duplication
- × Data latency



Direct Query

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

Power BI Architecture – Fabric



Limitations of Directlake

- 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/calculated tables
- No composite model
- No DateTime relationships

Always check the list of current limitations!

DAX Studio

- ✓ Write DAX
- ✓ Performance tuning
- ✓ Understand query plans
- ✓ Server timings – FE vs SE

Tabular Editor

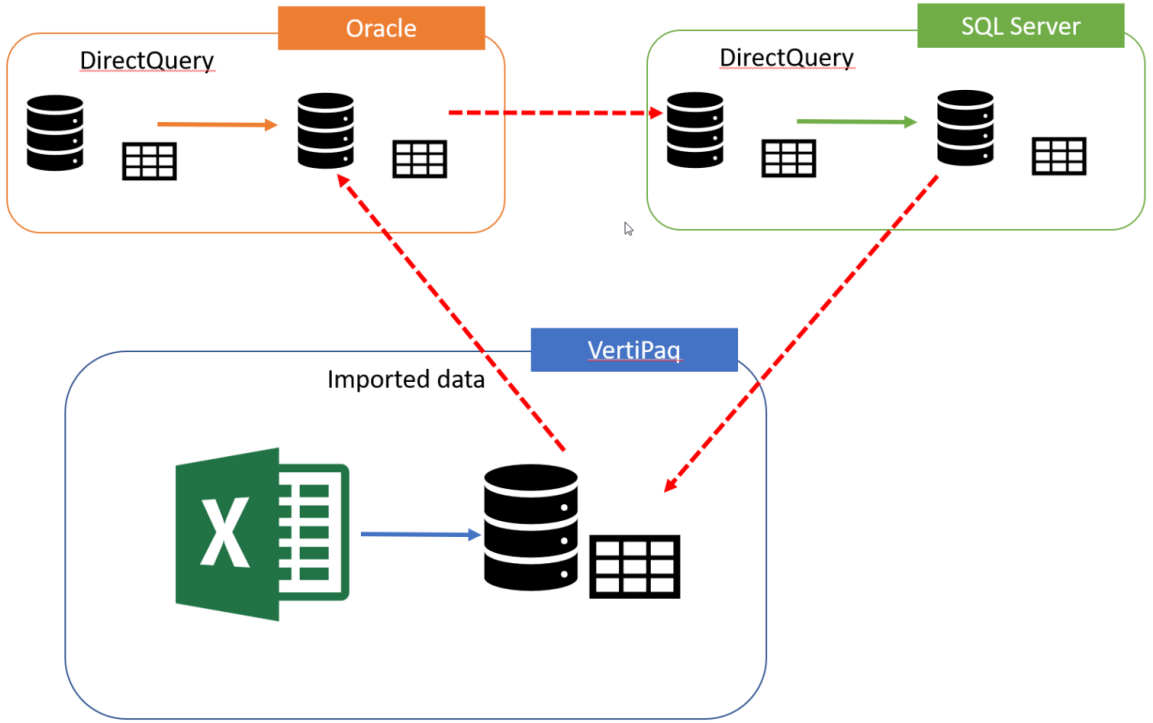
- ✓ Manipulate TOM
- ✓ Object-level security
- ✓ Custom KPIs

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



Composite models Best Practices

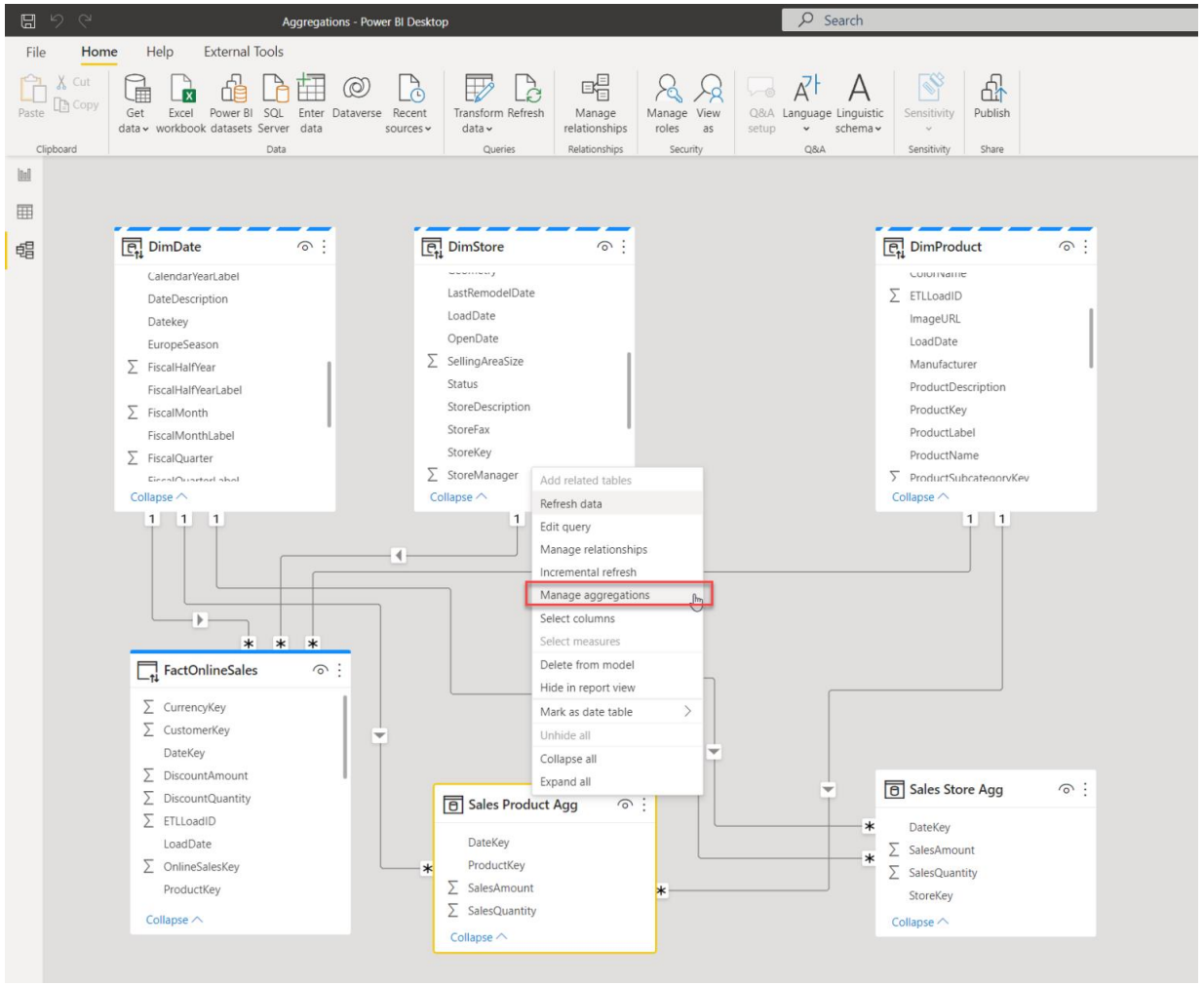
- ✓ Use only when pure Import mode is not an option
- ✓ Set dimension tables storage mode to Dual
- ✓ Identify the appropriate refresh rate
- ✓ Stick with general recommendations for optimizing DirectQuery scenarios

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

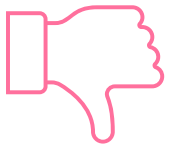


RLS in a Nutshell

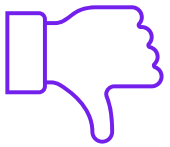
Limit access to specific attributes (country, product category…)

Static vs Dynamic RLS

Static

- ✓ Basic scenarios
- ✓ Rules maintained from within PBIX file
- ✓ Easy and straightforward setup
- 
- ✓ Maintenance, lot of manual work, not reusable

Dynamic

- ✓ Control access on a granular level
- ✓ Access control via data model
- ✓ Reusable and less maintenance
- 
- ✓ Adds to model complexity

RLS Considerations

- ✓ One user in multiple roles – filters work in additive way
- ✓ Performance impact – enforce RLS filters on dimension tables
- ✓ Assign AAD groups instead of individual accounts

OLS in a Nutshell

The underlying object is completely obscured from the data model (unlike with RLS)!

Other Concepts/Features to Learn…

DAX

- Variables
- Iterator functions (SUMX…)
- Window functions (OFFSET, ROWNUMBER…)
- Information functions (ISBLANK, HASONEVALUE…)

Power BI “stuff”

- Calculation groups
- Field parameters
- Dynamic format strings

Performance tuning

- DAX Studio
- Tabular Editor 2
- Incremental refresh

Explore and Analyze Data

Exploratory Analytics with PySpark

- *describe()* method on the DataFrame object
  - Count
  - Mean
  - Standard deviation
  - Minimum
  - Maximum

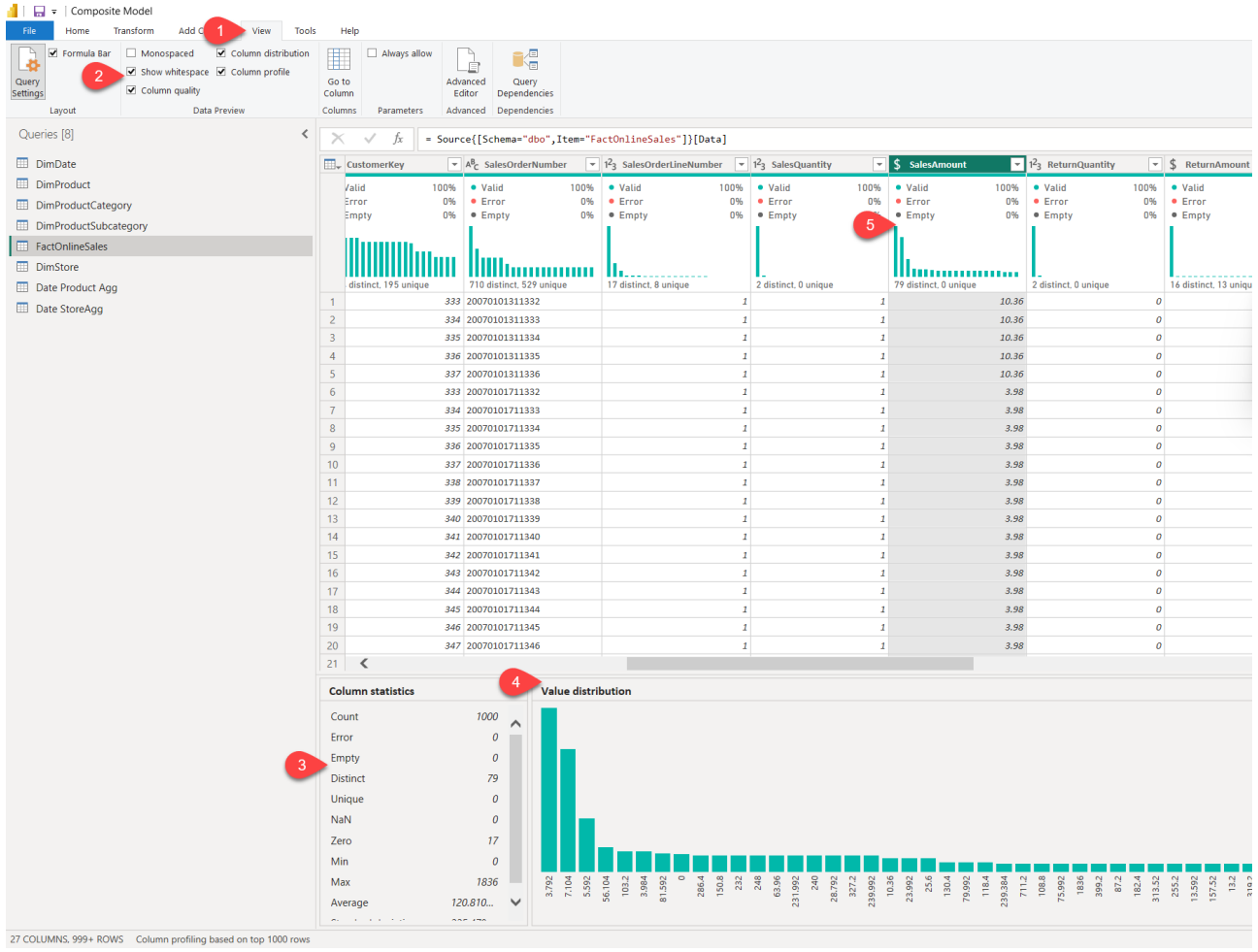
```
df = df.describe()
```

- *Grouping and aggregating the data*
  - groupBy()
  - agg

```
df = df.groupBy("gender").agg({"age": "avg",
"salary": "sum"})
```

Exploratory Analytics with Power Query

- *Column quality (5)*
  - Valid
  - Error
  - Empty
- *Column distribution (5)*
  - Distinct vs Unique
- *Column profile (3, 4)*
  - Value distribution
  - Column statistics



Query Data by Using SQL

Window functions

"Looking through the window" – perform calculations over a set of rows

OVER (defines a window)

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

ORDER BY (required depending on the function)

T-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

T-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

Other Concepts/Features to Learn…

1. **Prescriptive/Predictive analytics**
2. **“Classic” T-SQL**
3. **Query XMLA endpoint (DAX Studio)**

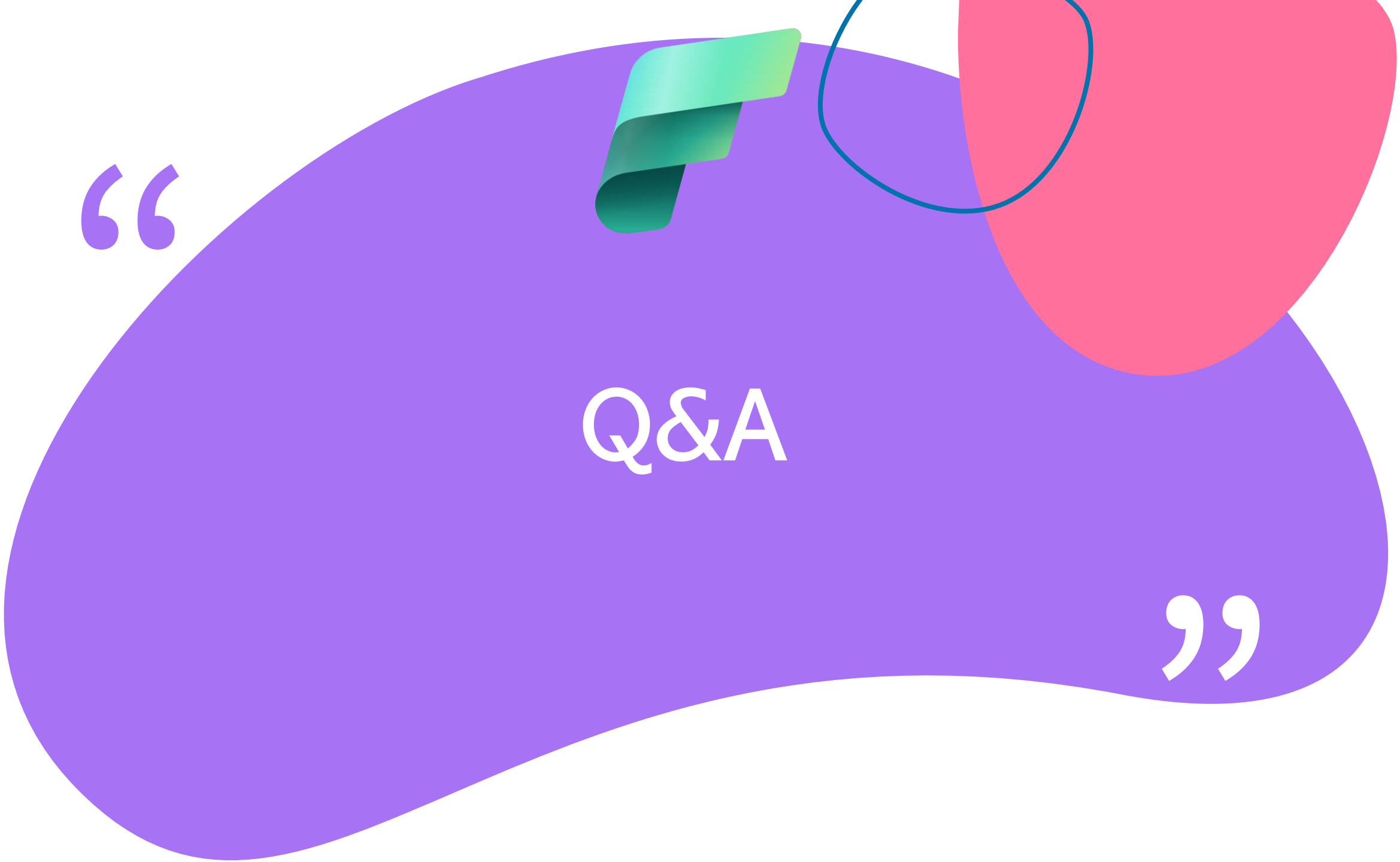
Learning Resources

Learning Resources

Data-Mozart.com

Serverlesssql.com

kevinrchant.com


“
Q&A
”

Thank you

Nikola Ilic
nikola@data-mozart.com
www.data-mozart.com