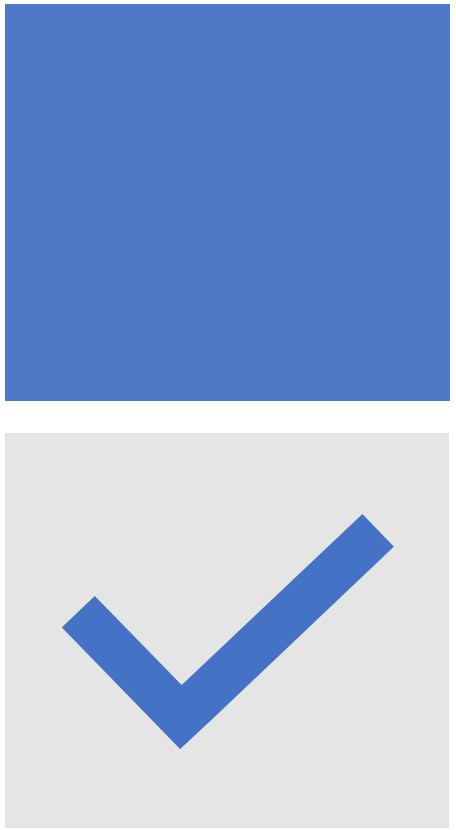
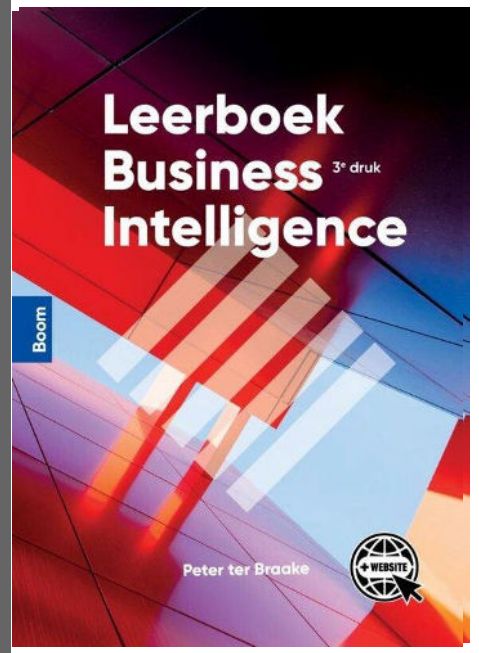
Microsoft Fabric

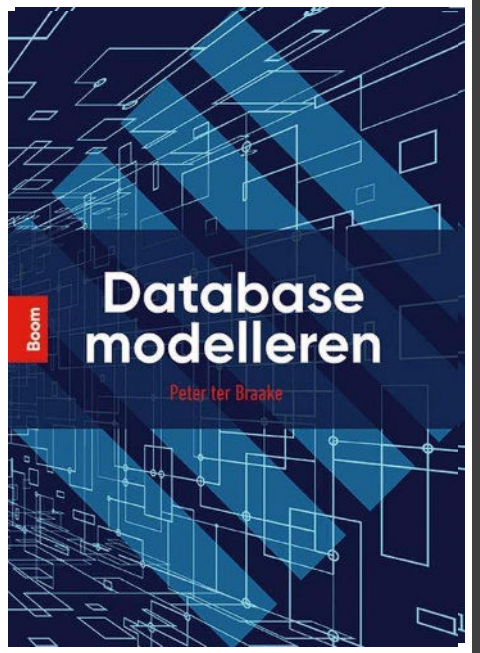


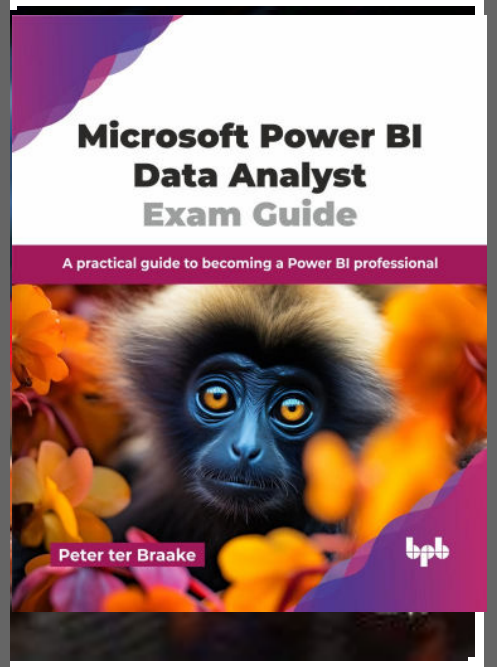
Peter ter Braake

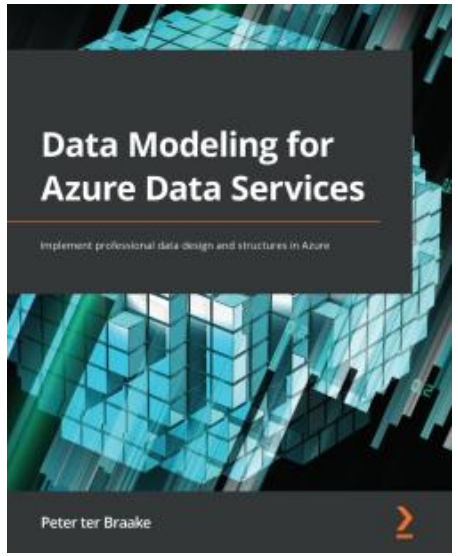
Introducing myself

- Peter ter Braake
  - trainsql@live.nl
  - Independent contractor
  - MCT since 2002
- Microsoft Data Platform
  - SQL Server
    - Incl. SSRS
  - Business Intelligence
  - Power BI
- Author







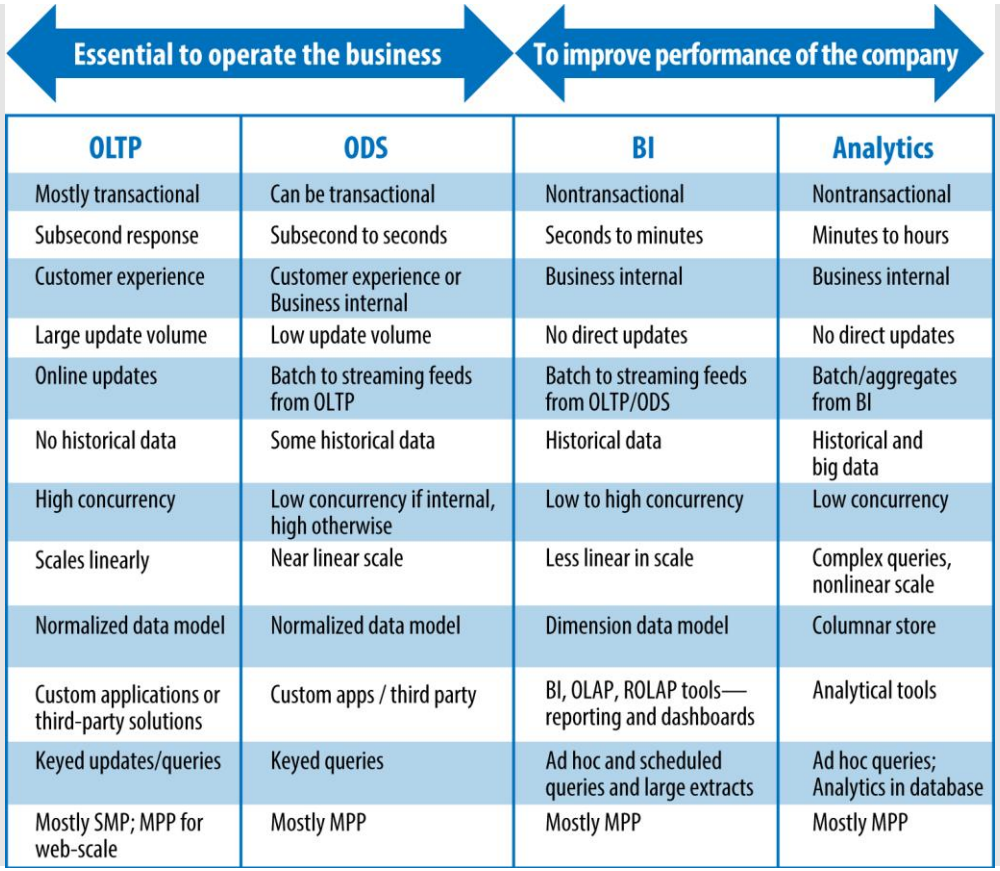


Agenda

- Medallion architecture
- Lakehouse
- Pipeline
- (py)Spark / Delta
- Dataflow Gen2
- Warehouse
- Semantic model
- Power BI
- Security
- Eventhouse
- Machine Learning

Medallion architecture

Operational versus Analytical

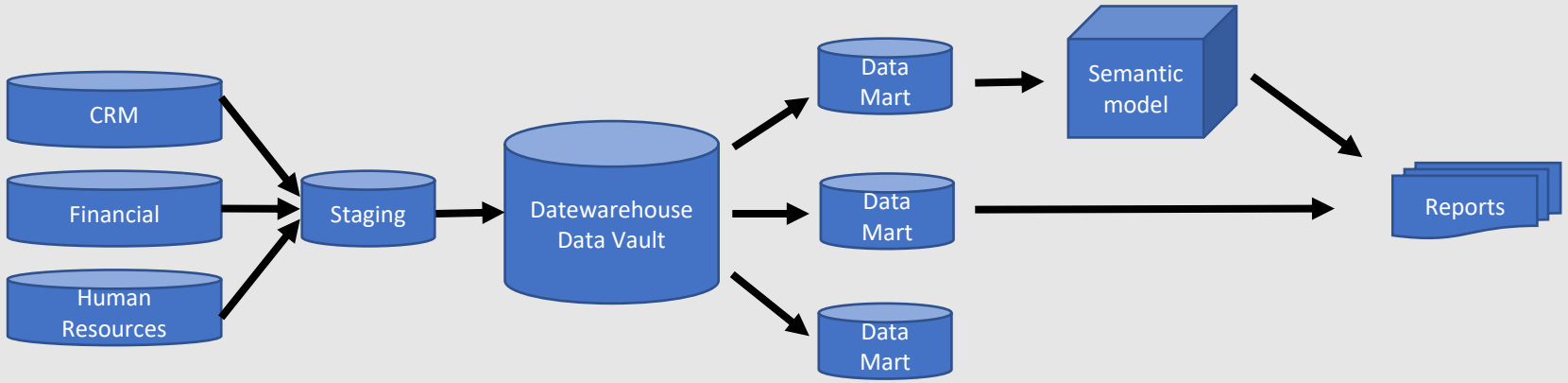


Essential to operate the business		To improve performance of the company	
OLTP	**ODS**	**BI**	**Analytics**
Mostly transactional	Can be transactional	Nontransactional	Nontransactional
Subsecond response	Subsecond to seconds	Seconds to minutes	Minutes to hours
Customer experience	Customer experience or Business internal	Business internal	Business internal
Large update volume	Low update volume	No direct updates	No direct updates
Online updates	Batch to streaming feeds from OLTP	Batch to streaming feeds from OLTP/ODS	Batch/aggregates from BI
No historical data	Some historical data	Historical data	Historical and big data
High concurrency	Low concurrency if internal, high otherwise	Low to high concurrency	Low concurrency
Scales linearly	Near linear scale	Less linear in scale	Complex queries, nonlinear scale
Normalized data model	Normalized data model	Dimension data model	Columnar store
Custom applications or third-party solutions	Custom apps / third party	BI, OLAP, ROLAP tools—reporting and dashboards	Analytical tools
Keyed updates/queries	Keyed queries	Ad hoc and scheduled queries and large extracts	Ad hoc queries; Analytics in database
Mostly SMP; MPP for web-scale	Mostly MPP	Mostly MPP	Mostly MPP

Performing analytics on operational databases might suffer from:

- Bad performance
  - Of LOB application
  - Of analytics itself
- Complexity of operational databases
  - Error prone
  - Makes BI developers in-productive
- Lack of consistency / 360 degree view
- Bad Data Quality
- Lack of historical data

Where we come from



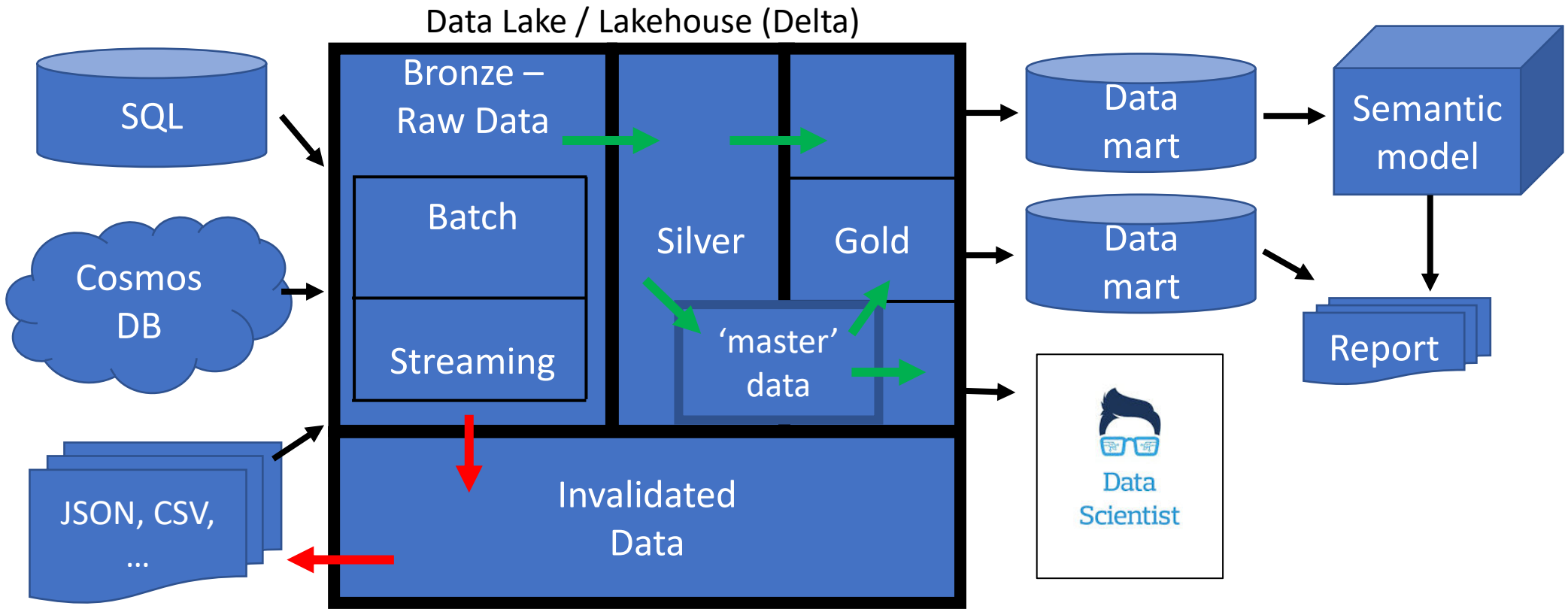
Source → ETL → BISM → Report

Power BI:

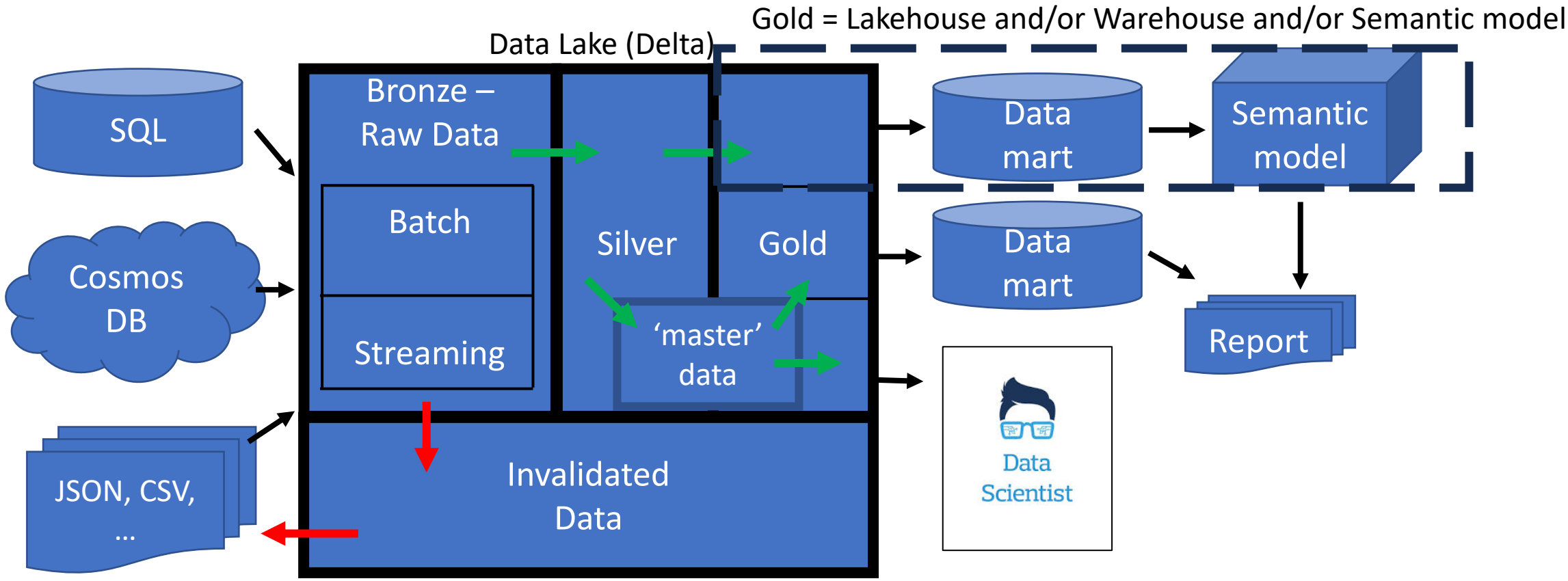
Source → Get Data → Power Query → DAX → Report

- DWH architecture:
  - Kimball
  - Inmon
  - Data Vault
- Same architecture, same best practices
- Implement solutions as far to the left as possible

Architecture Overview



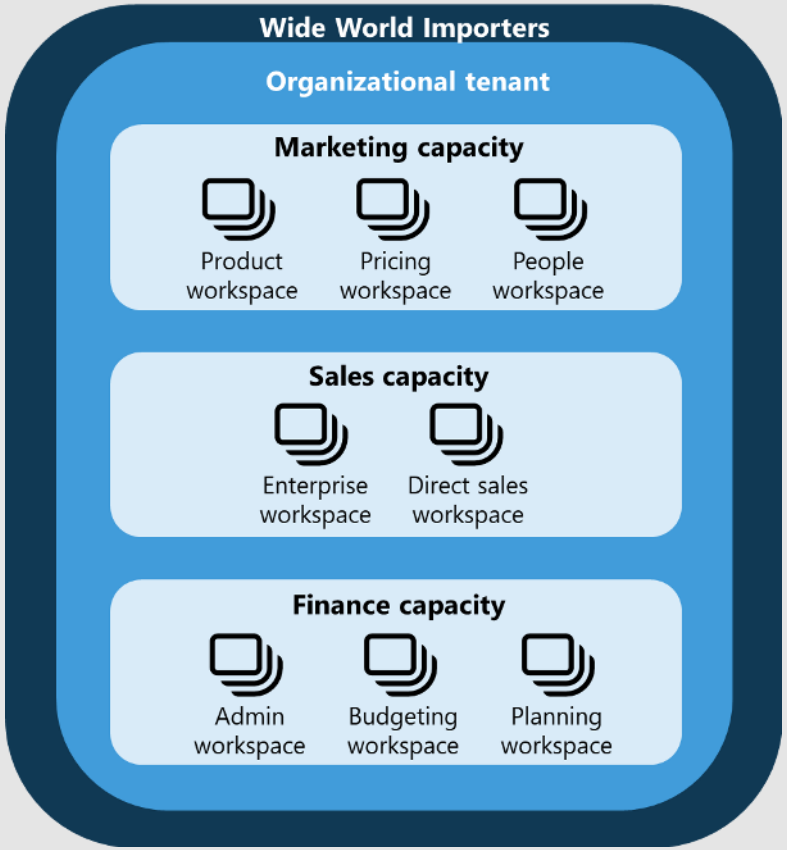
Architecture Overview



Consider virtual layers (data virtualization) whenever possible

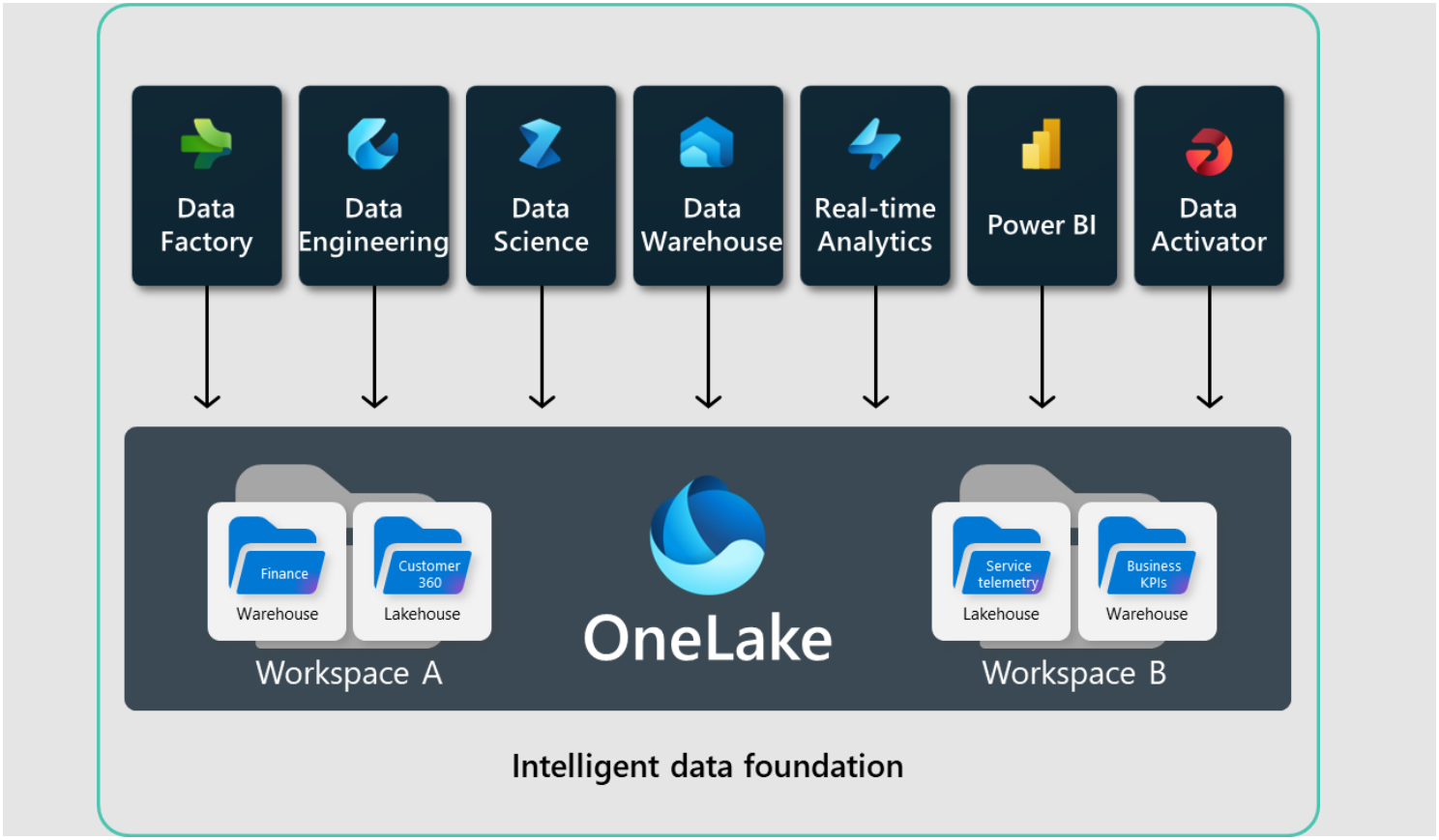
Get started: Fabric Lakehouse

Fabric concepts



- Tenant
  - Dedicated space to create, store, and manage Fabric items
  - Aligned with Entra
  - Maps to root of OneLake
  - Top level of hierarchy
- Capacity
  - Dedicated set of resources available for use
  - One or more per tenant
  - Offered through Fabric SKU (and Trials)
- Domain
  - Logical grouping of workspaces
  - Used to organize items in a way that makes sense to organization
  - Can contain subdomains + contains one or more workspaces
  - Roles: Domain admin + Domain contributor
- Workspace
  - Collection of items
    - Access control
  - Can contain folders
- Items
  - Building blocks of Fabric platform
  - Notebooks, Semantic models, Reports, ...

Fabric architecture



- Single SaaS lake
  - Provisioned automatically with tenant
  - Build on ADLS Gen2
- All data
  - Stored in OneLake
  - Organized in hierarchical namespace
- Data is automatically indexed for
  - Discovery
  - MIP labels
    - Microsoft Information Protection Sensitivity labels
  - Lineage
  - PII scans
    - personally identifiable information
  - Sharing
  - Governance and compliance

Fabric workspace

Consider for example:

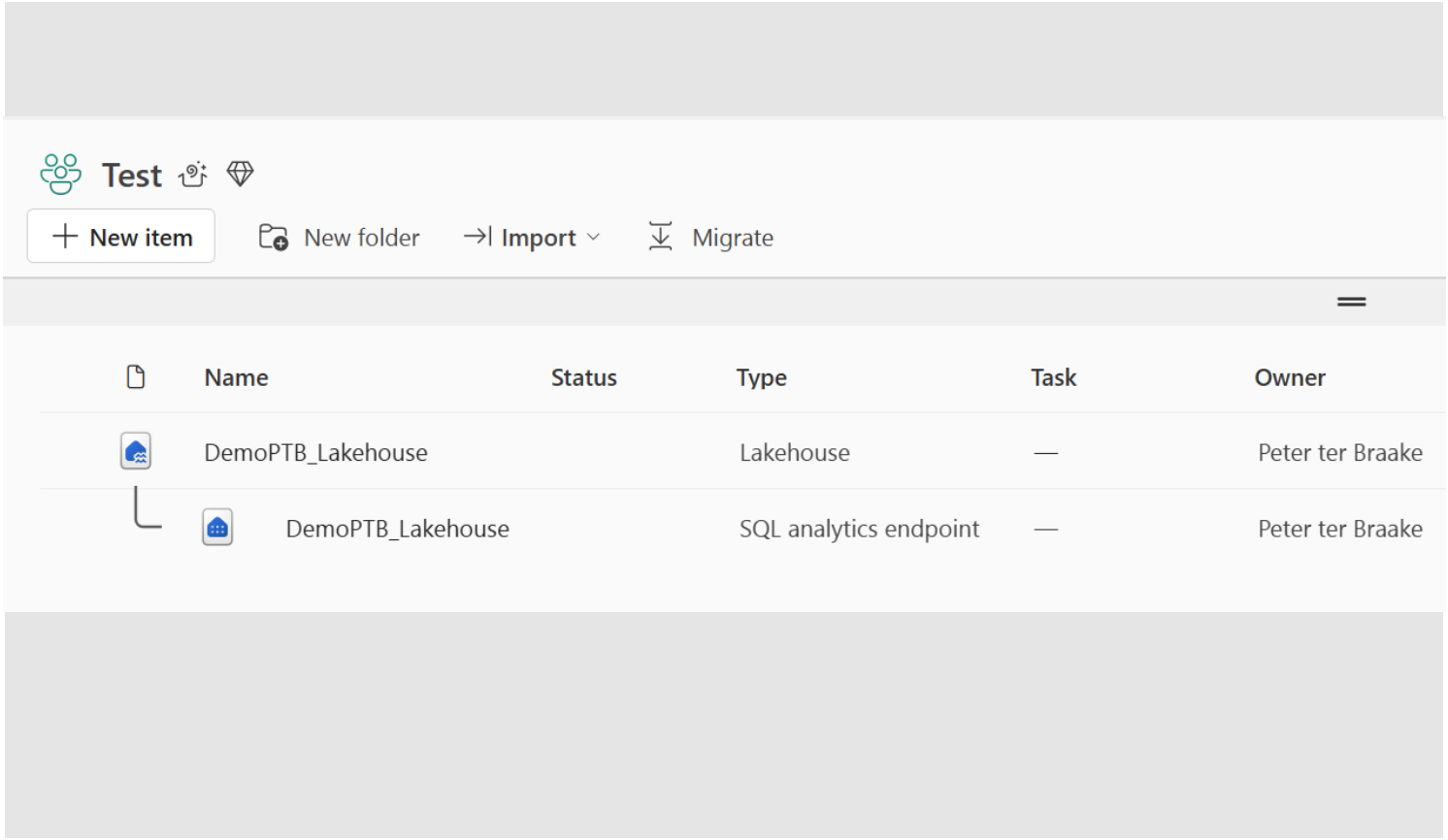
Workspace	Purpose	Artifacts
1	Storage	Bronze, Silver: Lakehouse
2	ETL	Pipelines, notebooks, variable library, dataflows
3	Storage	Gold: Lakehouse, Warehouse, Semantic model
4	Reporting	Reports, Dashboards
5	Monitor, logging	SQL Database

- Grouping of artifacts
- Can be monitored and shared as a whole
- Can be connected to Git repository
- Can be part of deployment pipeline
- Can be part of domain
- Has folders

Demo

Create Lakehouse

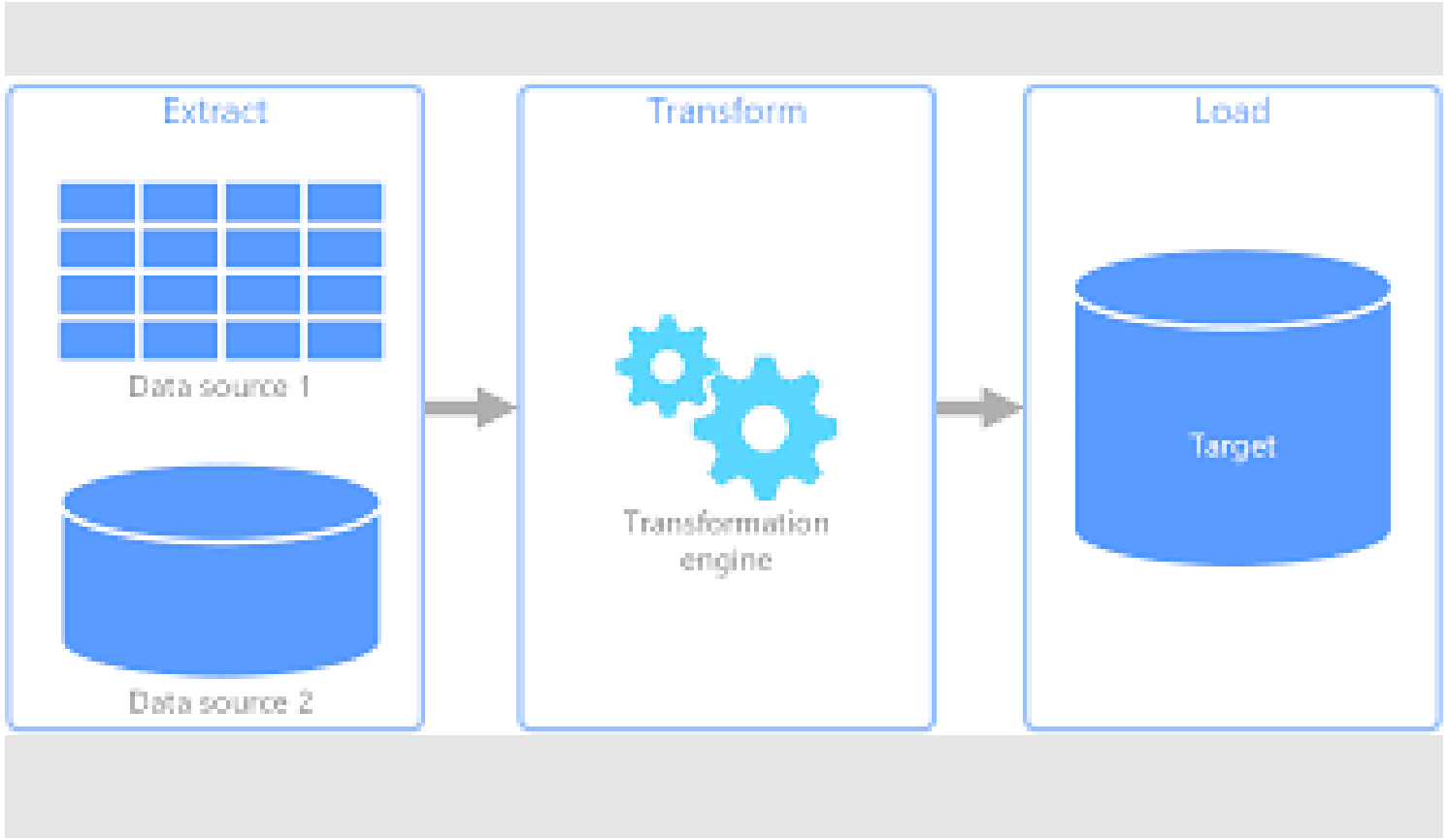
Create Lakehouse



- Create Lakehouse creates 3 artifacts
  - Lakehouse
    - Data in files
    - Data in tables
  - Semantic model
    - Formerly Power BI dataset
    - Based on Lakehouse tables
  - SQL Analytics endpoint
    - Use T-SQL to work with data
    - Use tools like
      - Azure Data Studio
      - SSMS

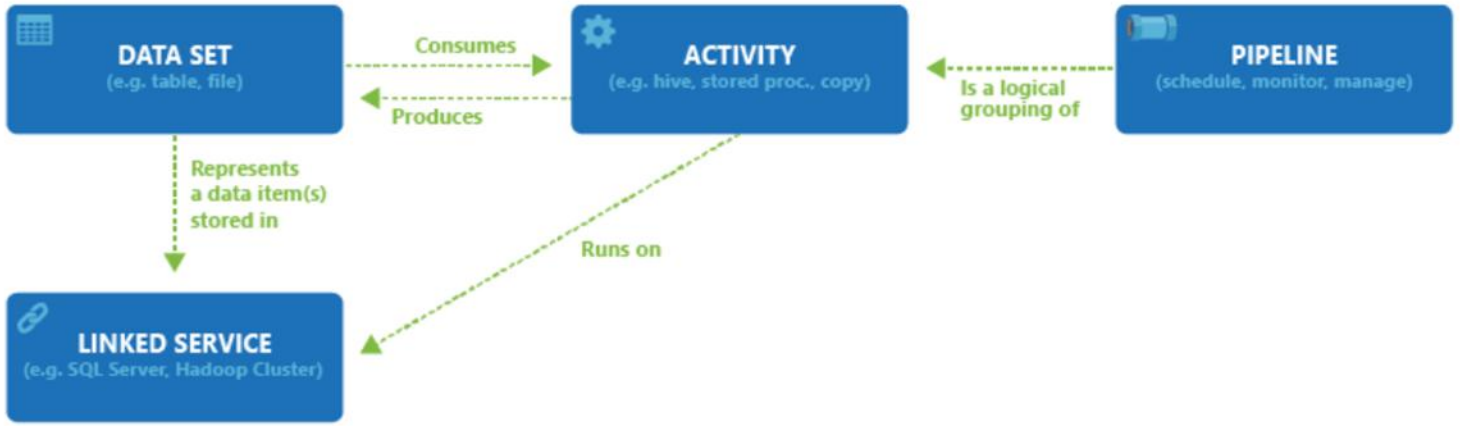
Pipeline

Extract *Transform* Load



- Cleanse data
  - Change data type
  - Replace errors
  - Replace nulls
  - Add (calculated) columns
  - …
- Transform
  - Combine tables
  - Combine sources
  - Split tables
- Apply business rules (when appropriate)

Pipeline



- Encapsulate flow of activities orchestrated by integration runtime
- Activities can include:
  - Data movement /transformation
  - External processing activities
  - Control flow activities
    - Manage variables
    - Contains processing logic
- Linked service
  - Provides access to
    - Storage (Data stores)
    - Compute
- Dataset
  - Reference to data
    - Table, file, …
  - Accessed through linked services
- Parameters

Demo

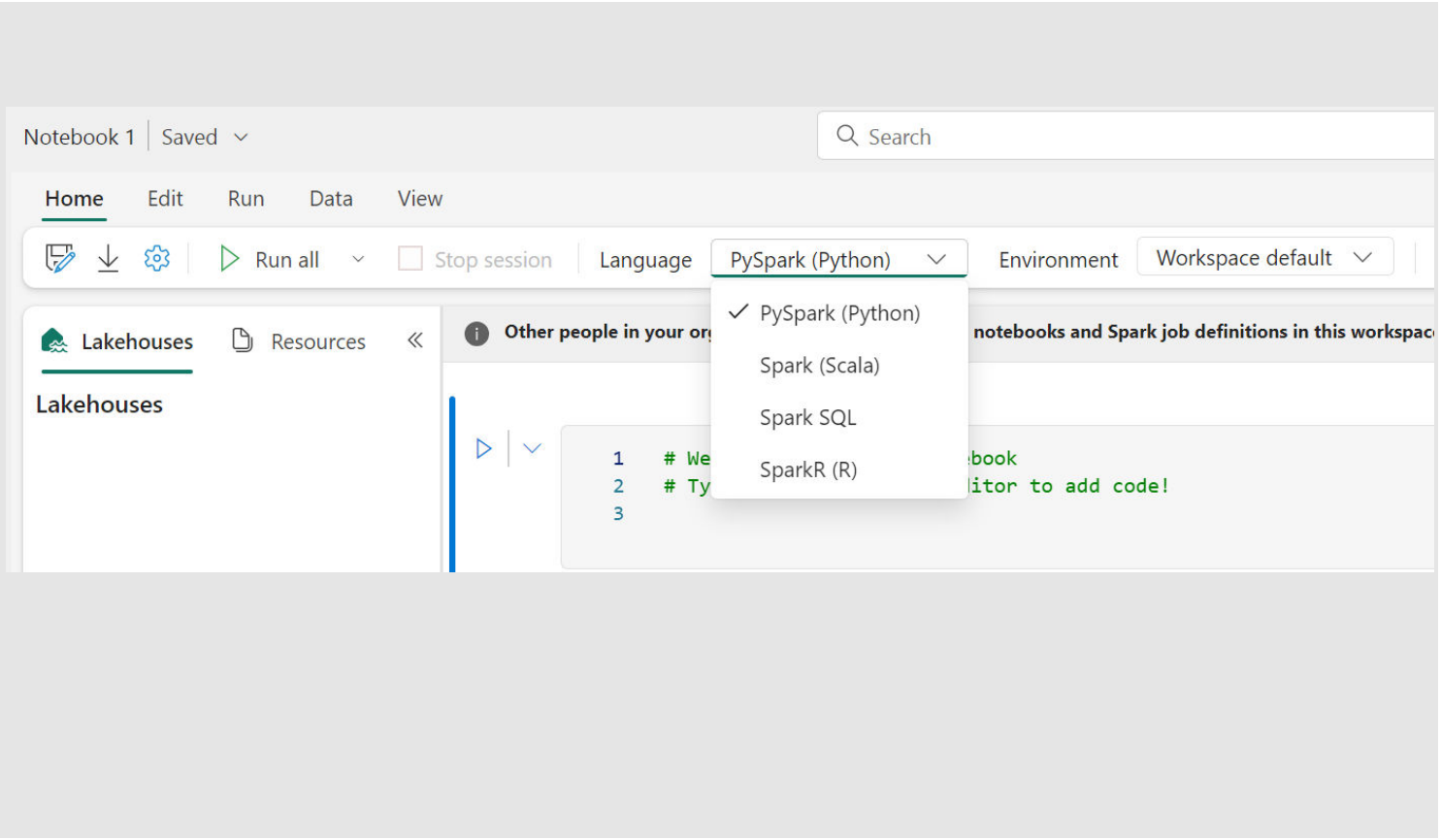
Create pipeline

(py)spark

Cluster

- Use more servers instead of bigger servers
- Head node / master node:
  - Where you connect to / what you see
  - Divides the work / assembles intermediate results
- Worker node:
  - Stores part of the data
  - Executes part of the query

Spark Pools



- Apache open-source Spark technology
- Backed by (configurable) clusters
  - Highly scalable
- Notebook oriented
  - Multiple languages
    - Python
    - Scala
    - SQL
    - …
  - Multiple libraries

Demo

Create notebook

spark.read

```
File = "dbfs:/FileStore/shared_uploads/trainsql@live.nl/Employee/Employee.csv"
Wildcard = "dbfs:/FileStore/shared_uploads/trainsql@live.nl/Employee/*.csv"
Folder = "dbfs:/FileStore/shared_uploads/trainsql@live.nl/Employee/"

# read single file
df1 = (spark.read
        .format("csv")
        .option("header", "true")
        .load(File)
)

# read multiple file2
df2 = (spark.read
        .format("csv")
        .option("header", "true")
        .load(Wildcard)
)

# read all files from folder
df3 = (spark.read
        .format("csv")
        .option("header", "true")
        .load(Folder)
)
```

- Read:
  - File
  - Multiple files using wildcards
  - All files from folder
- Use `.option("recursiveFileLookup", "true")`
- Infer schema or specify explicitly

dataframe

```
1   df_employee_filtered = (df_employee
2       .filter(df_employee.Salary >= '3000')
3       .select("Name", "Salary")
4   )
5
6   display(df_employee_filtered)
```

- Read data
  - `spark.read`
- Transform data
  - Select columns
  - Filter rows
  - Add calculated columns
  - Join data
  - Aggregate data
- Write data
  - `Dataframe.write`

spark.write

```
1   NewFile = 'Files/Silver/Northwind/Customer/'
2
3   (df_customer.write
4       .format('parquet')
5       .mode('overwrite')
6       .save(NewFile)
7
8       )
```

- Write:
  - Multiple file formats
    - Parquet
    - Delta
    - …
- Write to
  - File section: save
  - Table section: saveAsTable
    - Allows use of SQL
- Creates multiple files
  - Depending on cluster
- 4 modes
  - Append
  - Overwrite
  - Ignore
  - Error (errorifexists)

Delta Lake

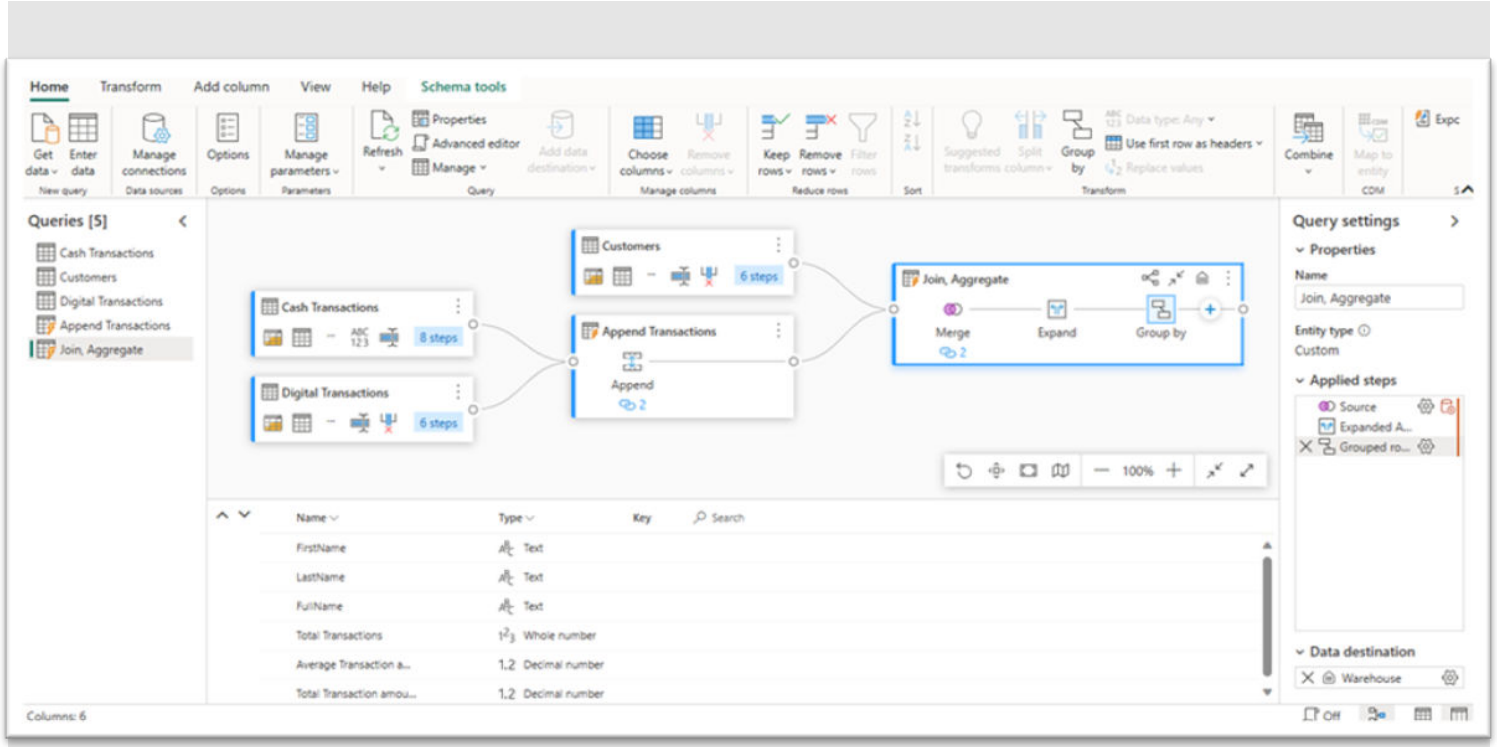
- Open-source storage layer that adds relational database semantics to Spark
- Relational tables that support querying and data modification
- Support for ACID transactions
- Data versioning and time travel
- Support for batch and streaming data
- Standard formats and interoperability

Dataflow Gen2

Demo

Create Dataflow

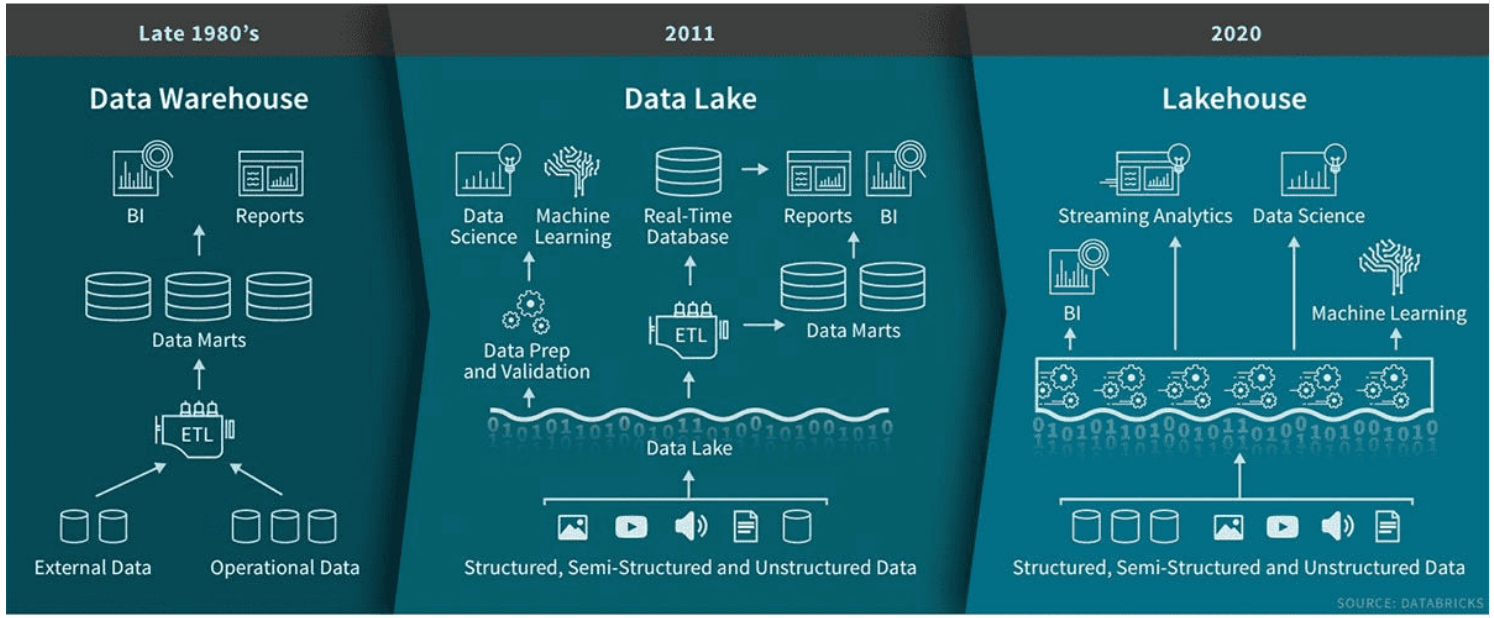
Dataflow Gen2



- Low-code graphical ETL environment
  - Extract data
    - Many source types
  - Transform data
    - 300+ transformations
  - Load data
    - Fabric destination
    - NO DESTINATION !
- Power Query online
  - Enhanced compute engine
- Run
  - Independently
    - Manual
    - Scheduled refresh
  - Pipeline activity

Fabric Warehouse

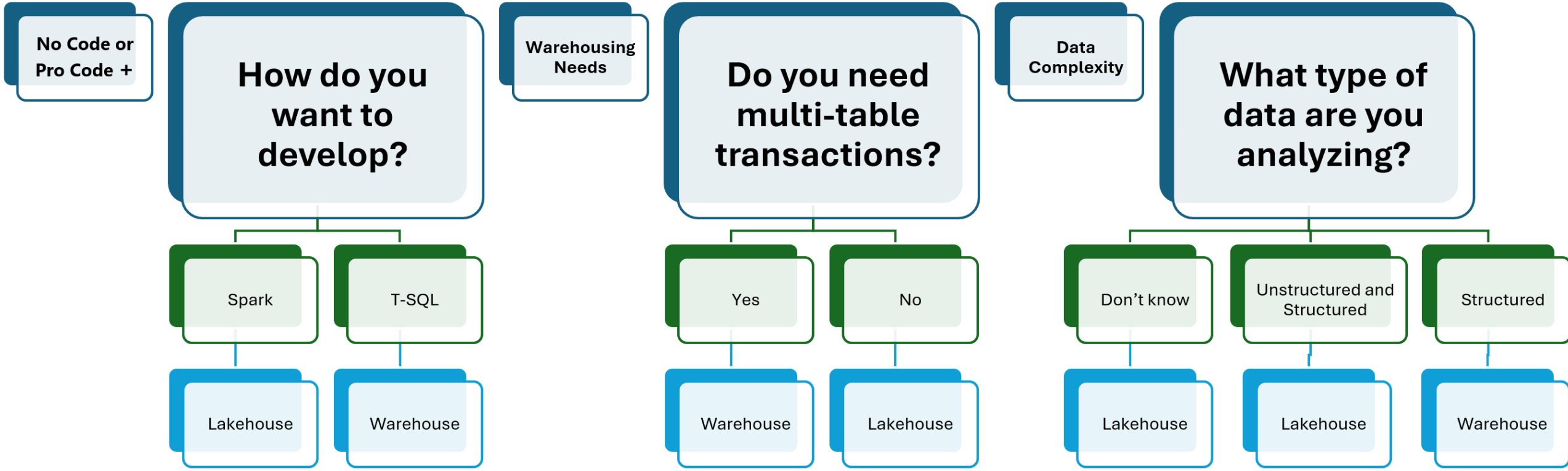
Warehouse versus Lakehouse



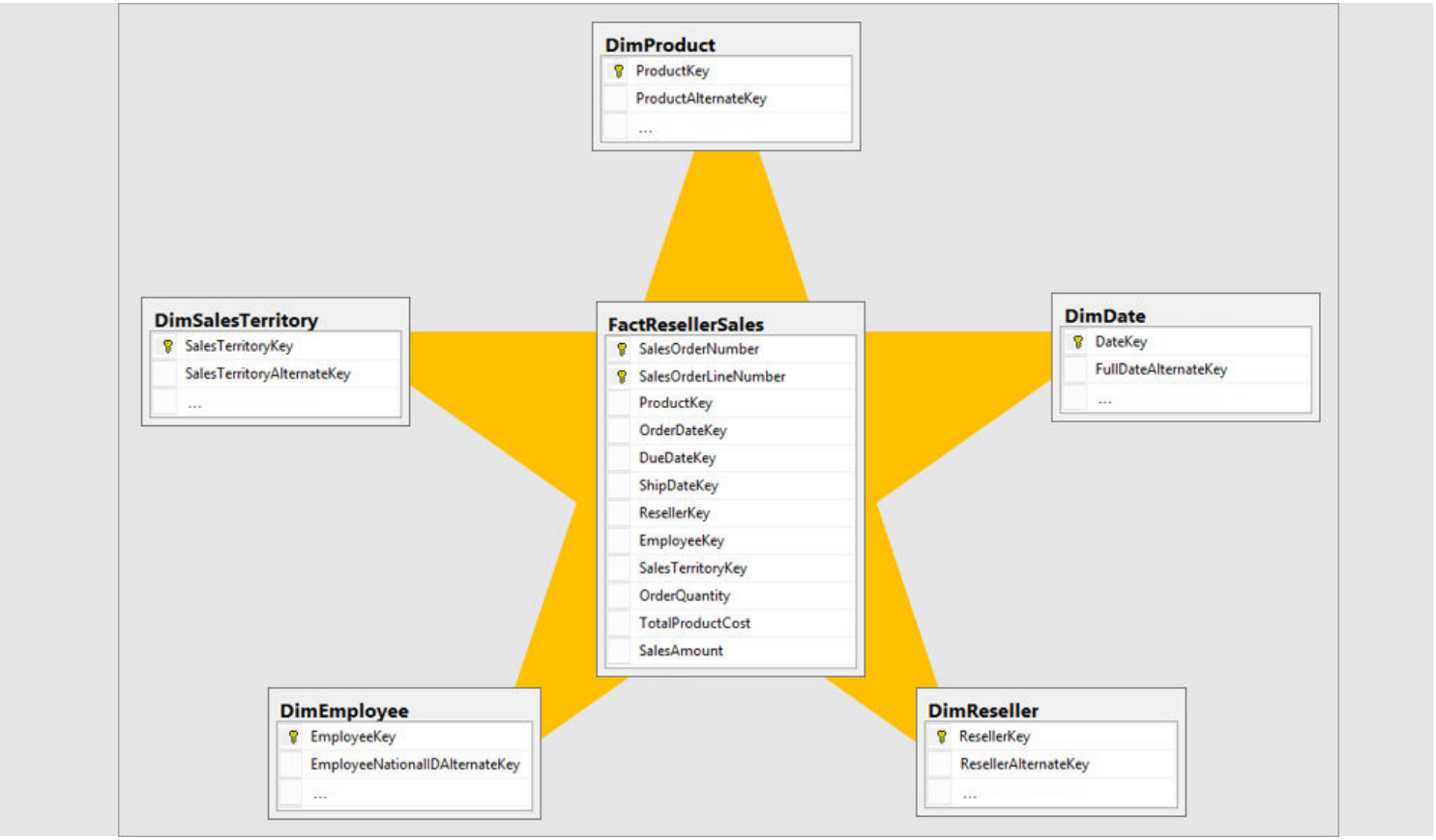
- Warehouse
  - Structured data
  - Multi-table transactions
  - High performance
  - Expansive security
    - Object/column/row-level, Dynamic Data Masking
  - T-SQL
    - DDL, DML
- Lakehouse
  - Semi-structured or unstructured data
  - Scalable and cost-effective
  - Supports Delta Lake features
  - T-SQL security (row/table level)
  - T-SQL, Spark (Scala, PySpark, Spark SQL, R)

Lakehouse versus Warehouse

https://learn.microsoft.com/en-us/fabric/fundamentals/decision-guide-lakehouse-warehouse



Why star schema?

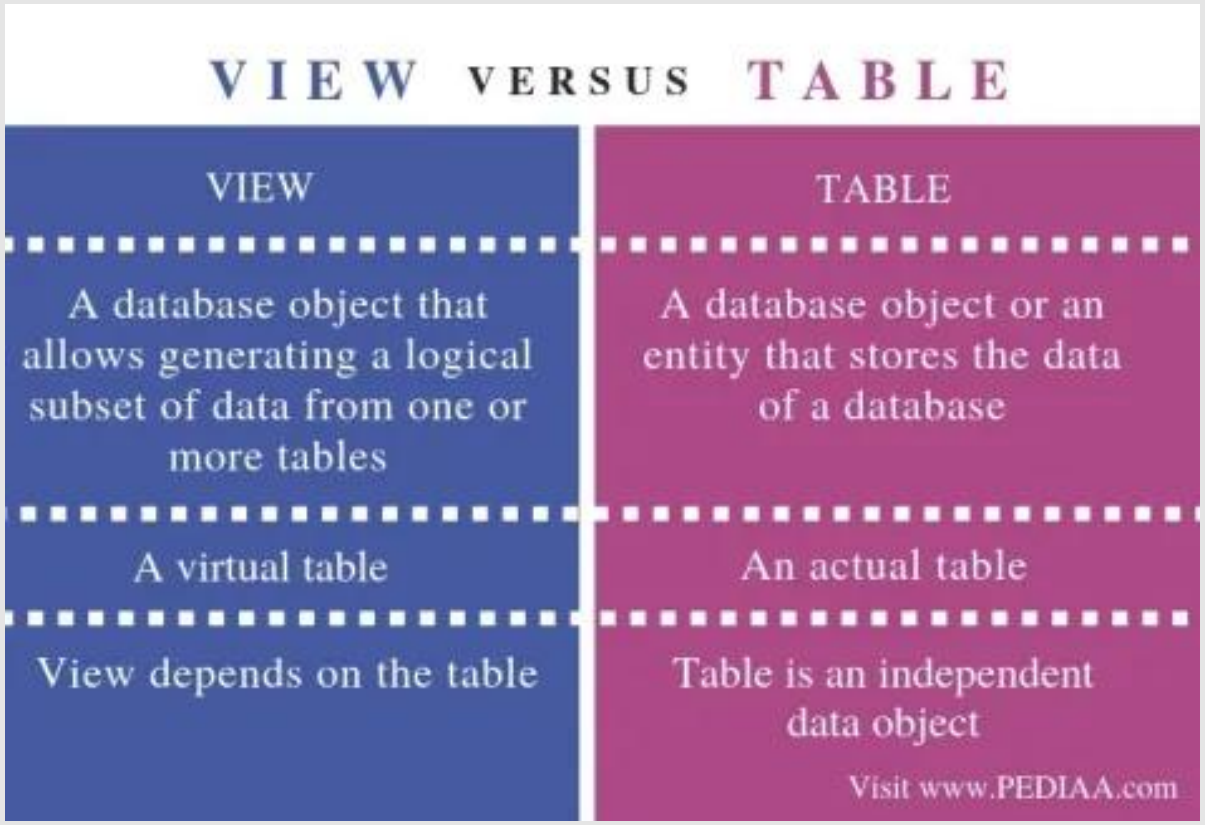


- Warehouse used for data mart (NOT data warehouse)
- Intuitive / simple
  - Less error prone
  - Rapid report development
- Better performance
  - Less ‘joins’
- Power BI
  - More consistent experience
  - Easier DAX
  - ‘auto exist’
  - Effect cross filter both ways
  - Better performance

Demo

Create warehouse

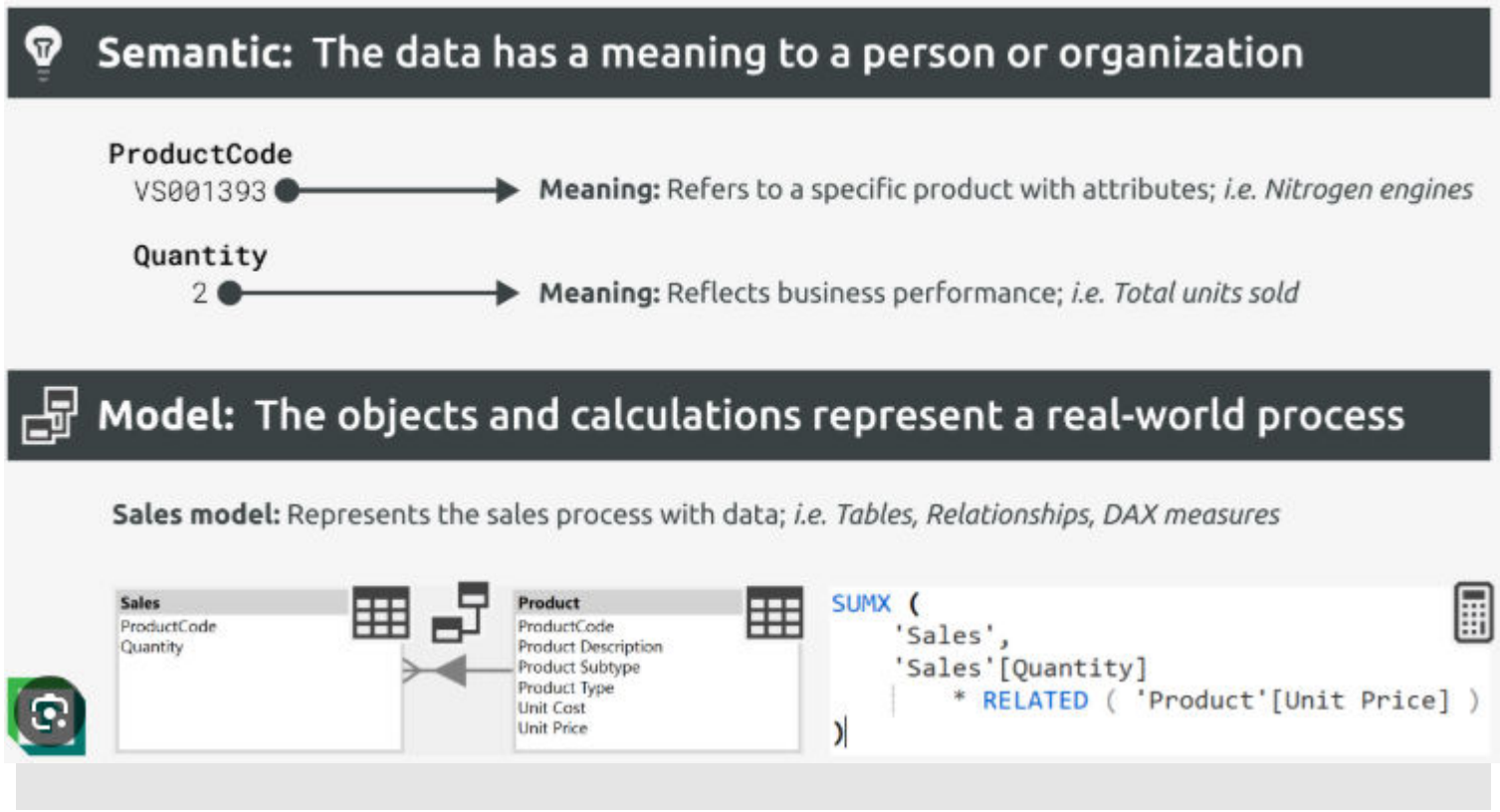
Warehouse view versus table



- Table
  - Stored physically:
    - storage cost overhead
    - ETL load time and cost
  - Supports:
    - Row level security
    - Semantic model DirectLake access
- View
  - Abstraction layer over tables
    - No storage
    - Compute cost when used
  - Supports:
    - Querying using SQL Analytics endpoint
    - No RLS, no DirectLake

Semantic model

What is a semantic model?

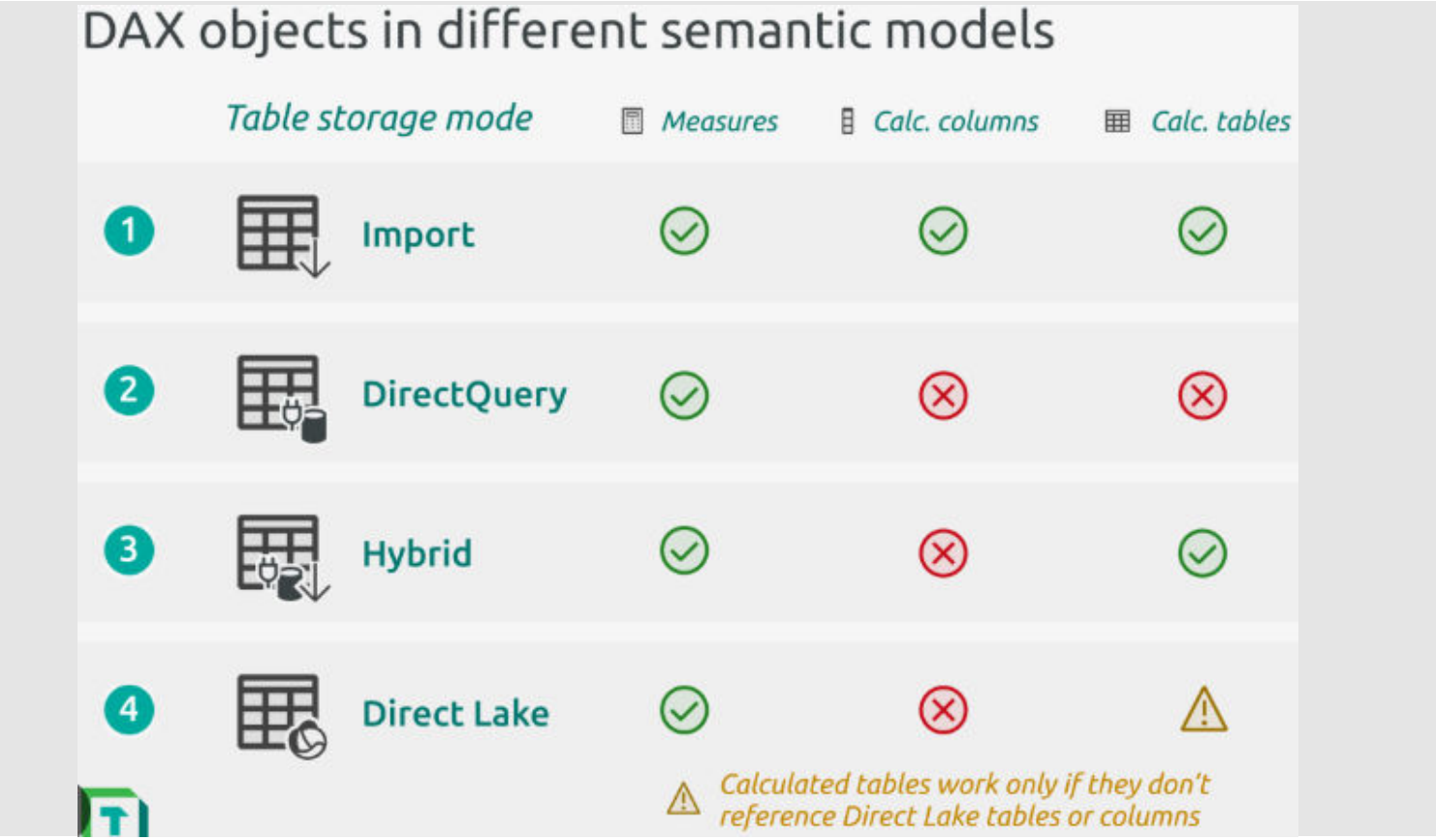


- Translates
  - Technical DB implementation
    - Table, PK, FK
    - Query, Join, Predicate
  - Into
    - Human terminology
- Abstraction over database
  - Allows for drag and drop reporting
  - Enables
    - Self service reporting
    - Faster reporting development
    - Central store for Business Rules

Demo

Create semantic model

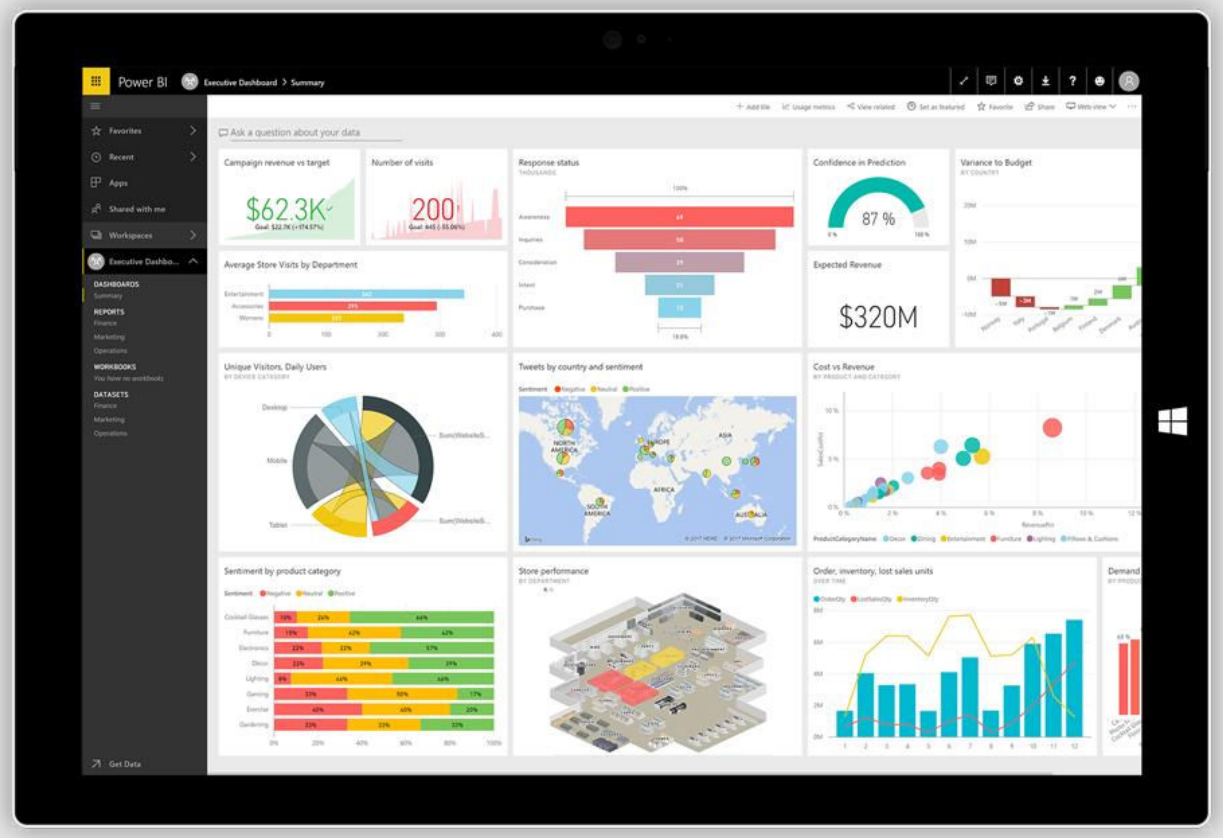
Storage modes



- Import
- DirectQuery
- Direct Lake
- Hybrid / Composite models

Power BI

Power BI



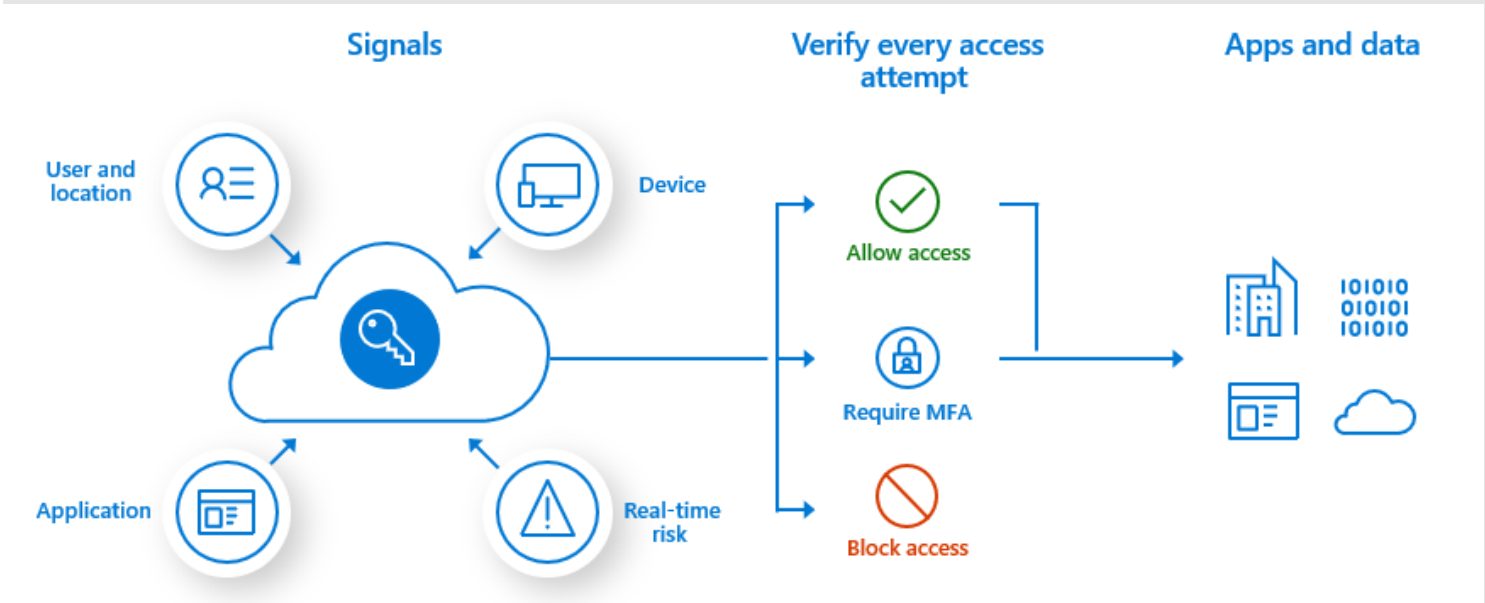
- Data visualization tool
- Use cases:
  - Corporate reporting
  - Exploratory reporting
  - Everything in between
- Parts:
  - Power BI Desktop
  - Power BI Service (Power BI Report Server)
  - Power BI Mobile app
- Power BI service
  - Collaboration
  - Workspaces
    - => Apps
  - Dashboards

Demo

Show report

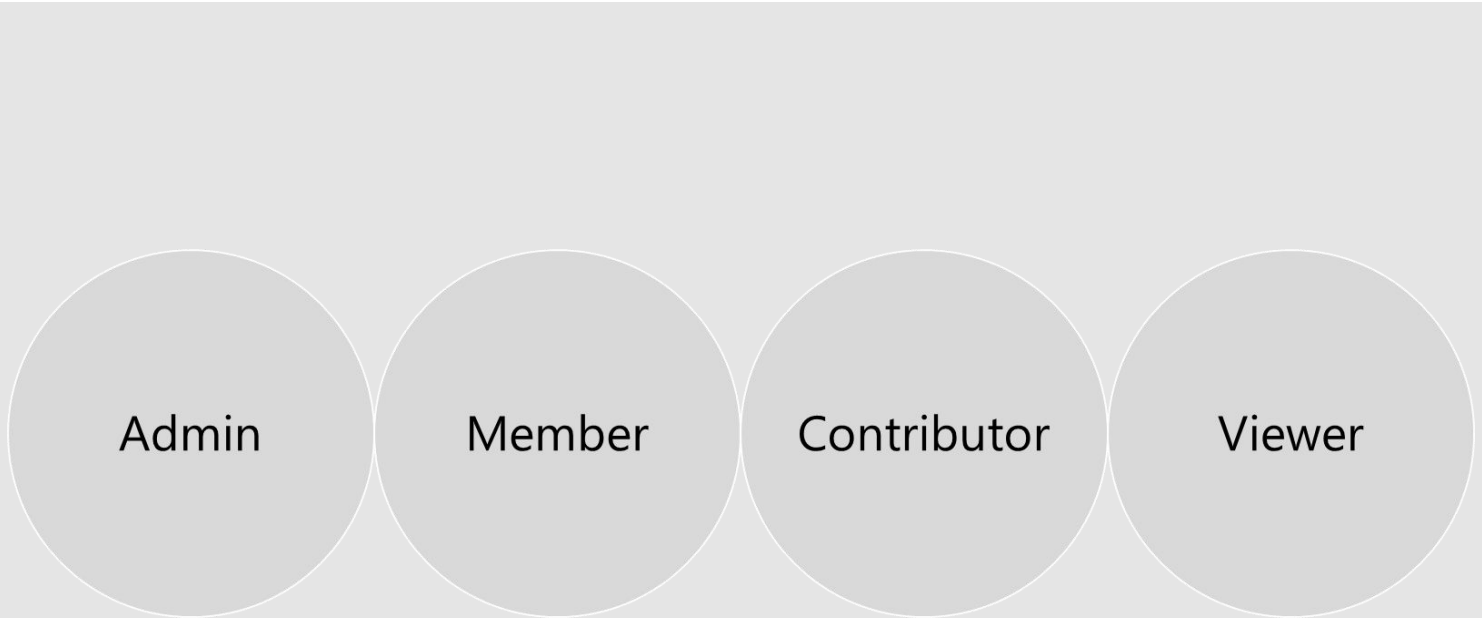
Security

Fabric security model



- Principle of least privilege:
  - Restrict user permissions to only what's needed
- Three layers:
  - Entra ID authentication
    - Who are you
    - Single sign on
    - MFA
  - Fabric access
    - Checks if user can access Fabric
  - Data security
    - Permissions on actions and data

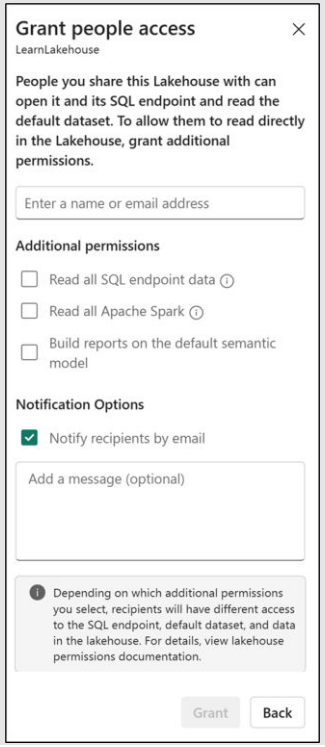
Workspace roles



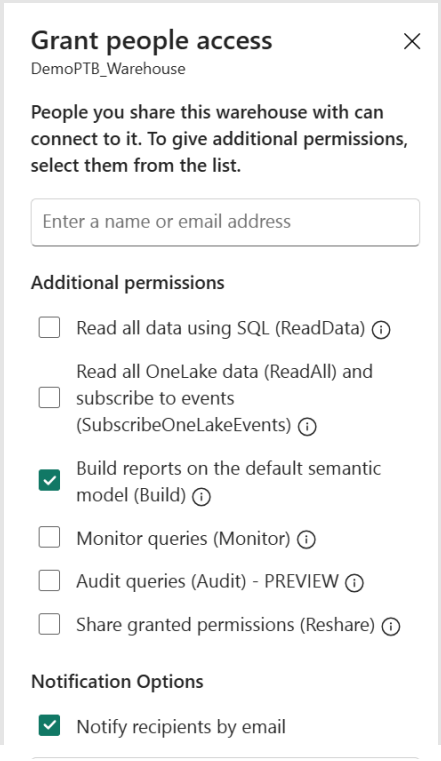
- Workspace role:
  - Preconfigured set of permissions
  - Hierarchical: share ALL artifacts
  - Assigned to
    - Individuals
    - Security groups
    - Microsoft 365 groups
    - Distribution lists
- Workspace roles
  - Viewer
    - View all content and data in workspace
    - Cannot modify anything
  - Contributor
    - Viewer +
    - Modify all content and data in the workspace
  - Member
    - Contributor +
    - Share all content and data in the workspace
  - Admin
    - Member +
    - Manage permissions

Share individual items

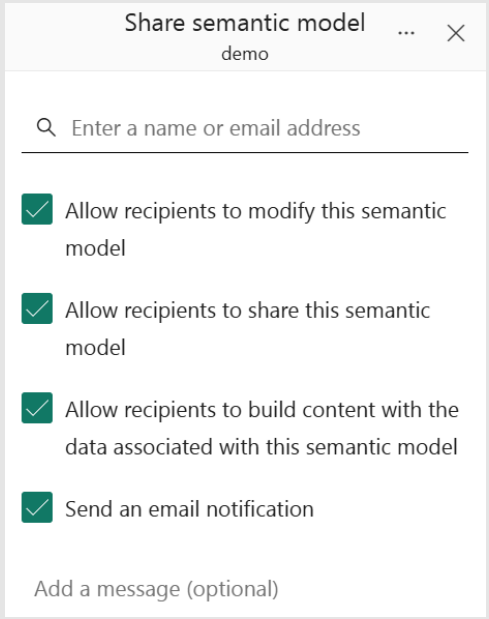
Lakehouse:



Warehouse:



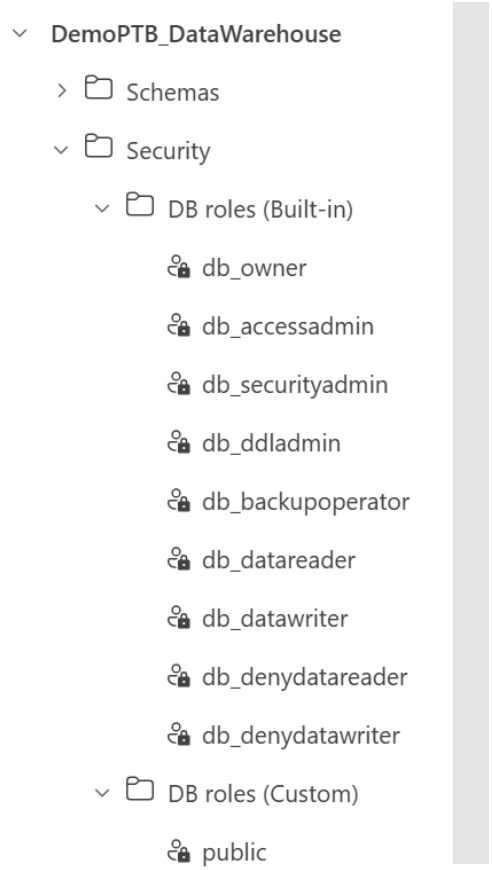
Semantic model:



- Item sharing = connect permission
  - + read for Semantic model
- Extra permissions often too much (equal to db_datareader)
  - Use fine grained security of SQL Server / Semantic model

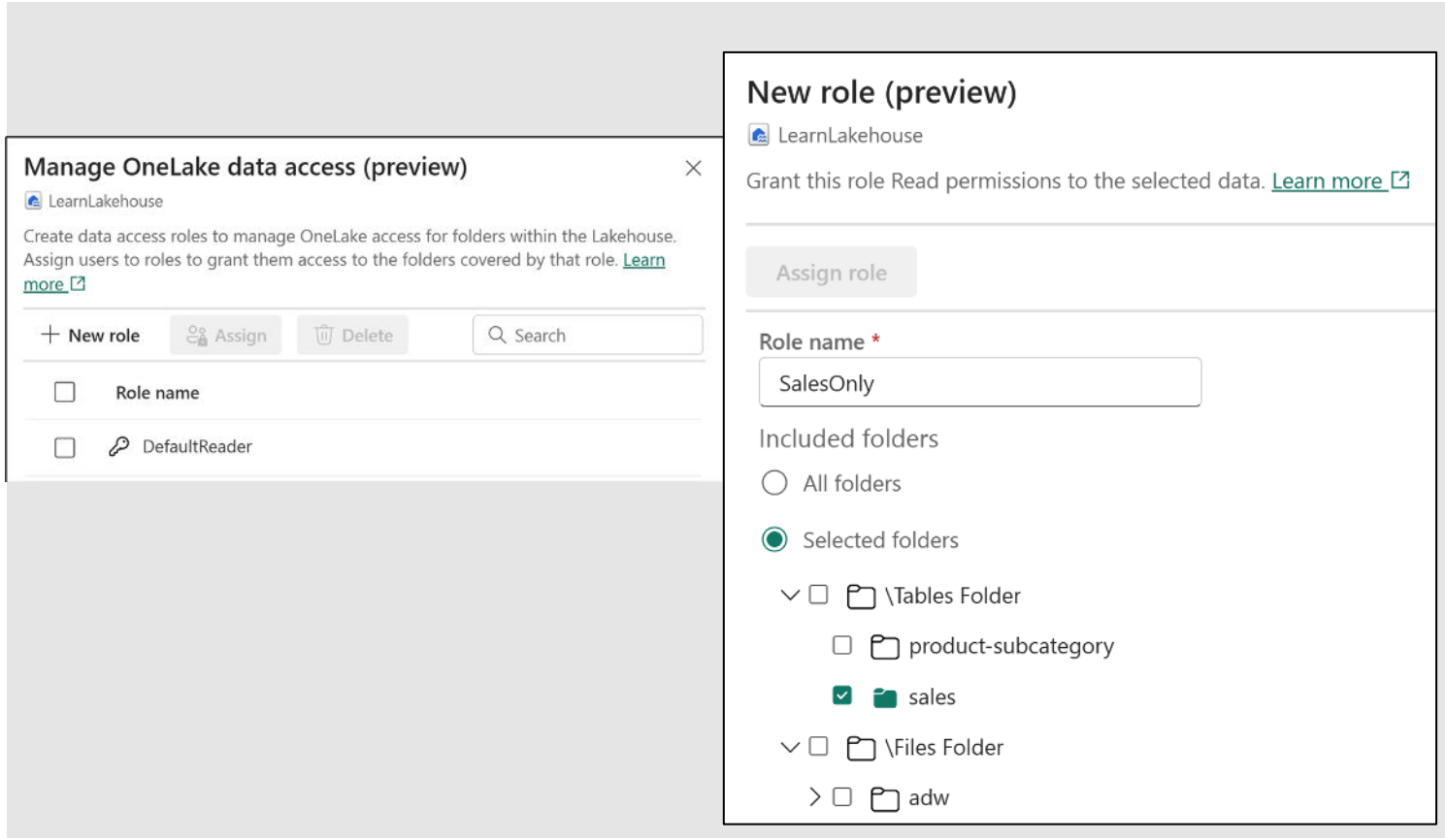
Granular permissions

```sql
-- Create a custom role
CREATE ROLE TestRole


-- Grant permission to role
GRANT SELECT ON Sales.Product
TO TestRole


-- Add user to role
ALTER ROLE TestRole
ADD MEMBER [peter@trainsql.onmicrosoft.com]


-- Remove user from role
ALTER ROLE TestRole
DROP MEMBER [peter@trainsql.onmicrosoft.com]
```



- Assign only Read permission in Fabric
- Add T-SQL
  - GRANT
  - DENY
  - REVOKE
- No explicit CREATE USER
- Database roles
  - Collection of permissions
  - Built-in roles
  - Custom roles
- Row Level Security
  - Warehouse, Semantic model

OneLake data access roles



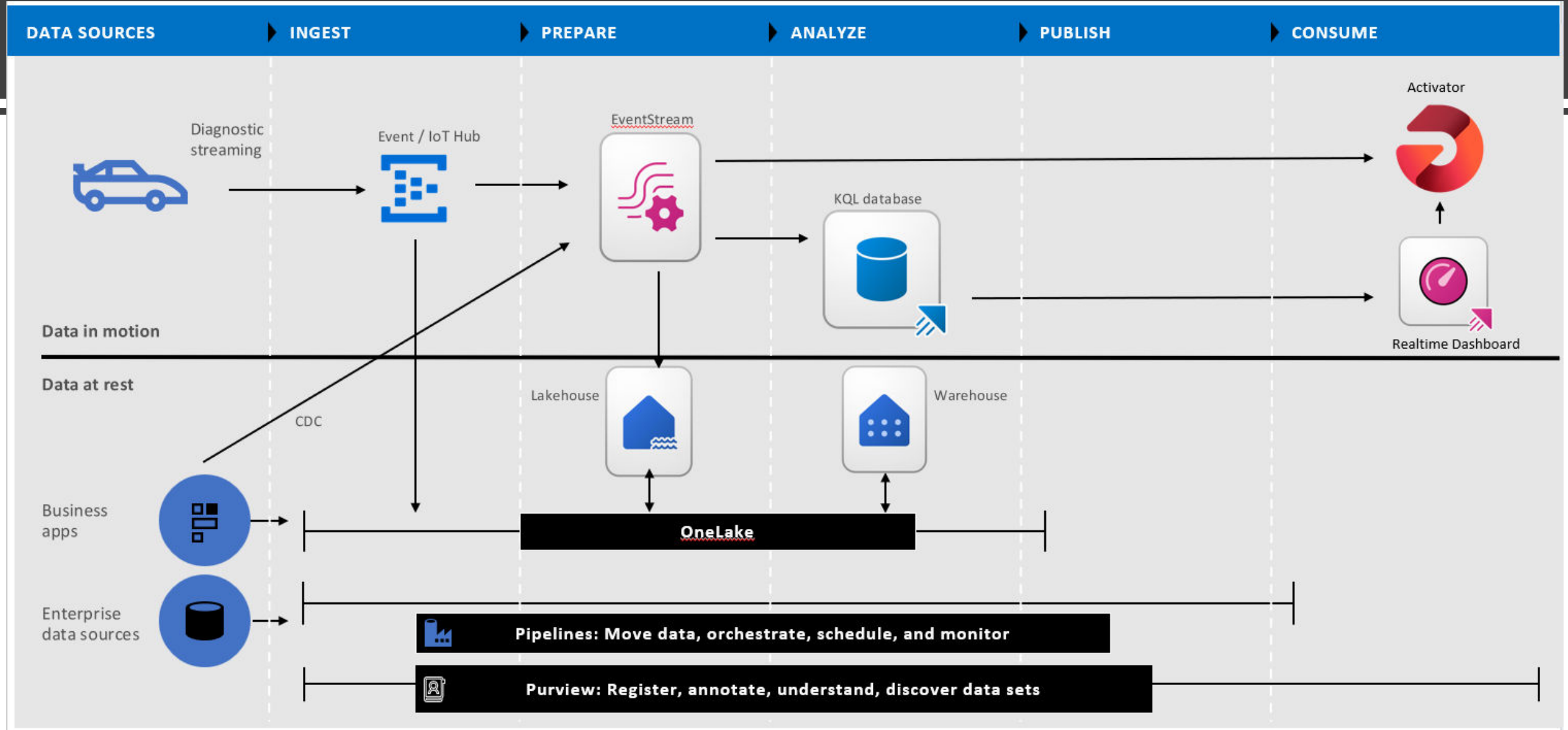
- Lake view of lakehouse is used to read data in Tables and Files folder
- Workspace / item permissions
  - Coarse access to data in lakehouse
- OneLake data access roles (preview)
  - Fine grained access
  - Create custom roles within lakehouse
  - Grant read permissions only to specific folders in OneLake
  - Folder security is inheritable to all subfolders
- Create custom OneLake role:
  - Select Manage OneLake data access (preview)
    - Toolbar in lake view of lakehouse
  - In New Role window
    - Provide role name
    - Select folders
  - Assign user / group to role
  - Select permissions:
    - Read
    - Write
    - Reshare
    - Execute
    - ReadAll

Demo – If Time Permits

Row Level Security

Eventhouse

Streaming Data Architecture Overview



Demo – If Time Permits

Stock eventstream

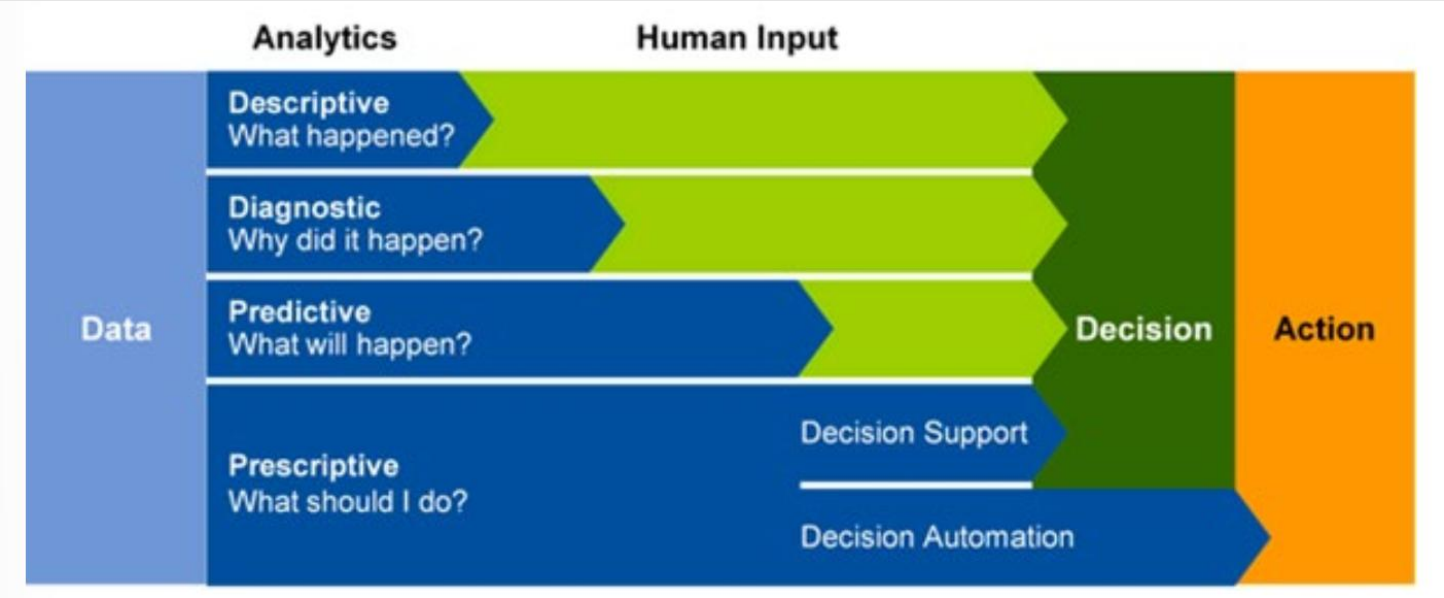
Fabric Machine Learning

Artificial Intelligence?

Software that imitates human capabilities

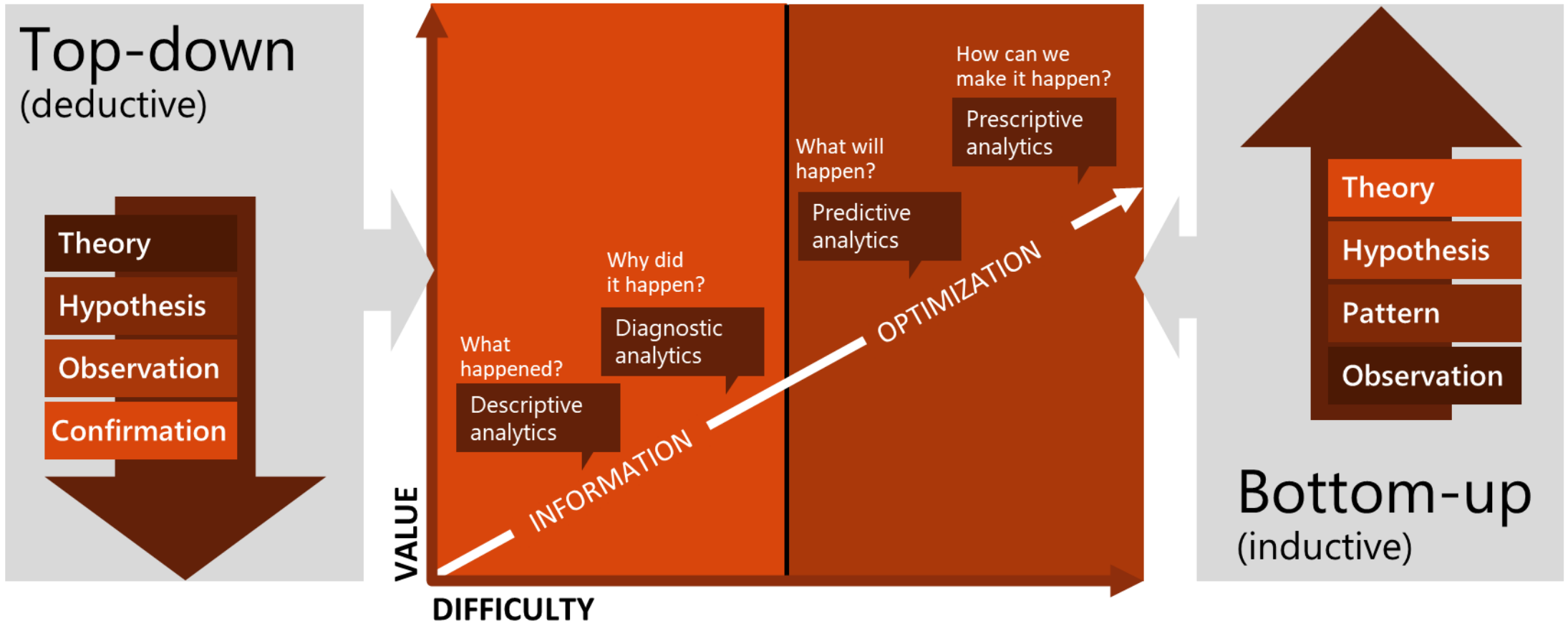
- Predicting outcomes and recognizing patterns based on historic data
- Recognizing abnormal events and making decisions
- Interpreting visual input
- Understanding language, engaging in conversations
- Extracting information from sources to gain knowledge

Advanced Analytics

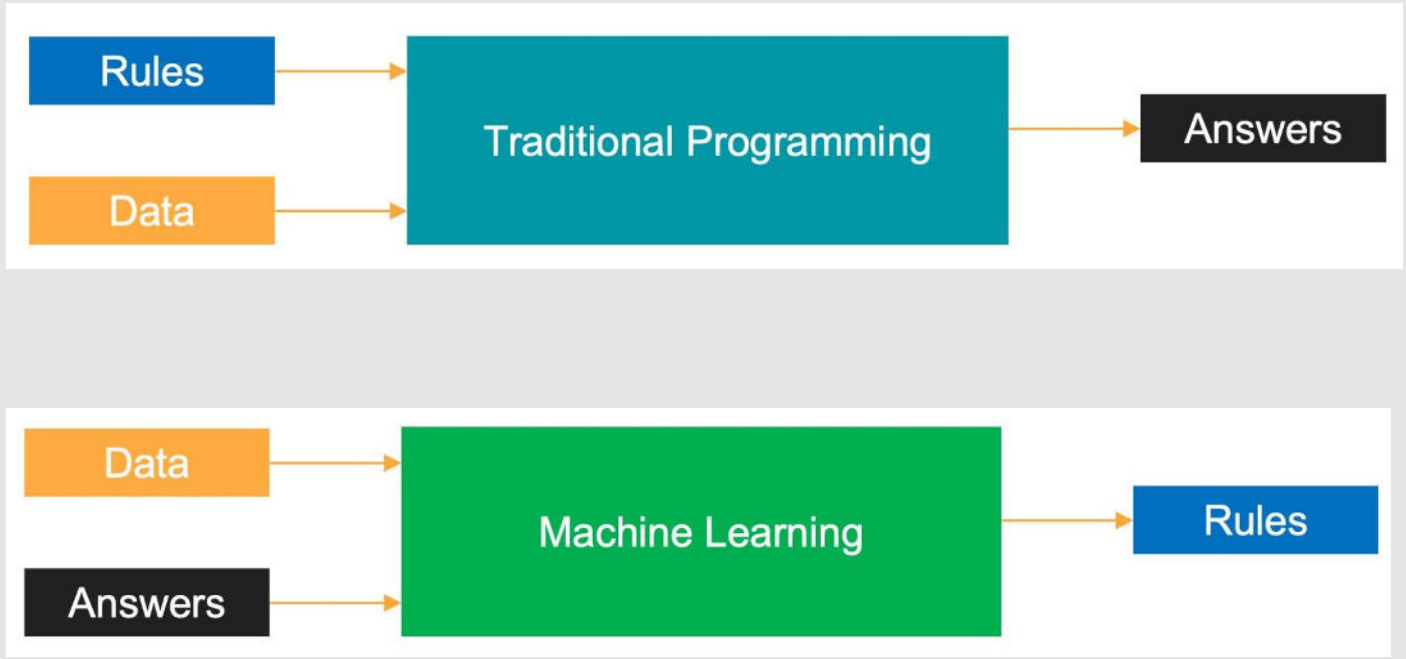


- Machine learning is a type of artificial intelligence that provides computers with the ability to learn without being explicitly programmed
  - They do not follow strictly static program instructions
- Formal definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" - *Tom M. Mitchell*
- "The goal of machine learning is to program computers to use example data or past experience to solve a given problem." – *Introduction to Machine Learning, 2nd Edition, MIT Press*

Top-down versus bottom-up



Machine Learning

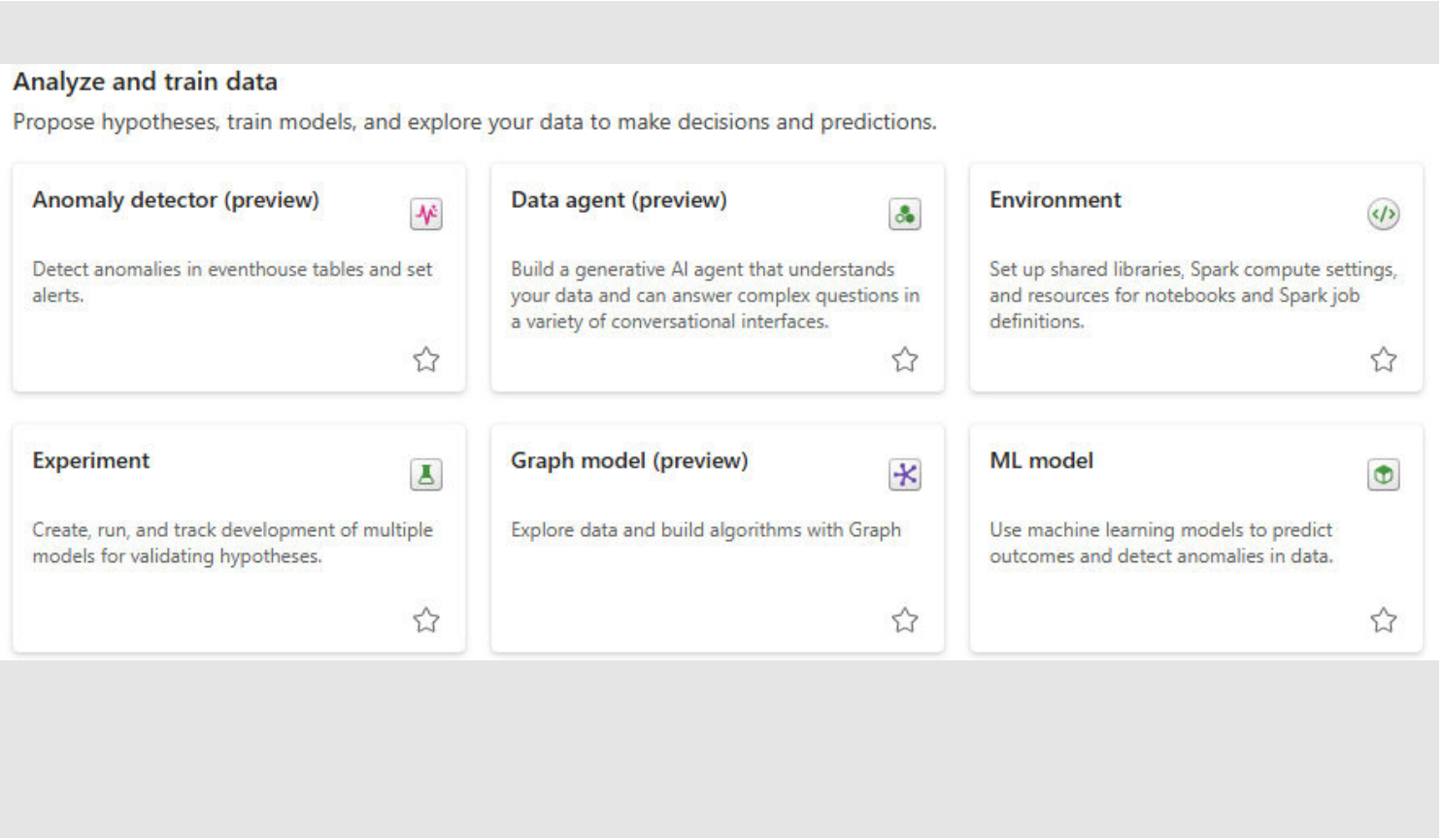


- Supervised learning
- Unsupervised learning
- Semi-supervised learning

- With tabular data
  - 1 row = 1 observation
  - 1 column =
    - 1 feature
    - Label to predict

- Columns/features can be
  - Continuous (numerical)
  - Discrete (categorical)
  - Datetime
  - Text
- Labels:
  - Continuous: regression
  - Discrete: classification

Analyze and train data

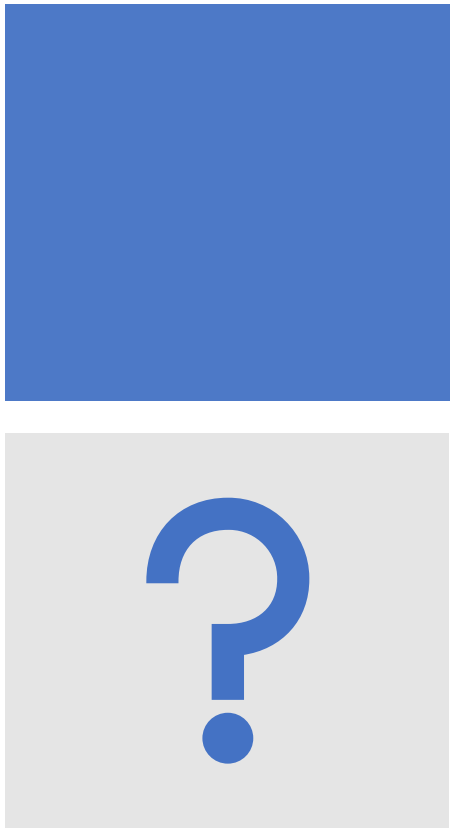


- Lake view of lakehouse is used to read data in Tables and Files folder
- Workspace / item permissions
  - Coarse access to data in lakehouse
- OneLake data access roles (preview)
  - Fine grained access
  - Create custom roles within lakehouse
  - Grant read permissions only to specific folders in OneLake
  - Folder security is inheritable to all subfolders
- Create custom OneLake role:
  - Select Manage OneLake data access (preview)
    - Toolbar in lake view of lakehouse
  - In New Role window
    - Provide role name
    - Select folders
  - Assign user / group to role
  - Select permissions:
    - Read
    - Write
    - Reshare
    - Execute
    - ReadAll

Demo – If Time Permits

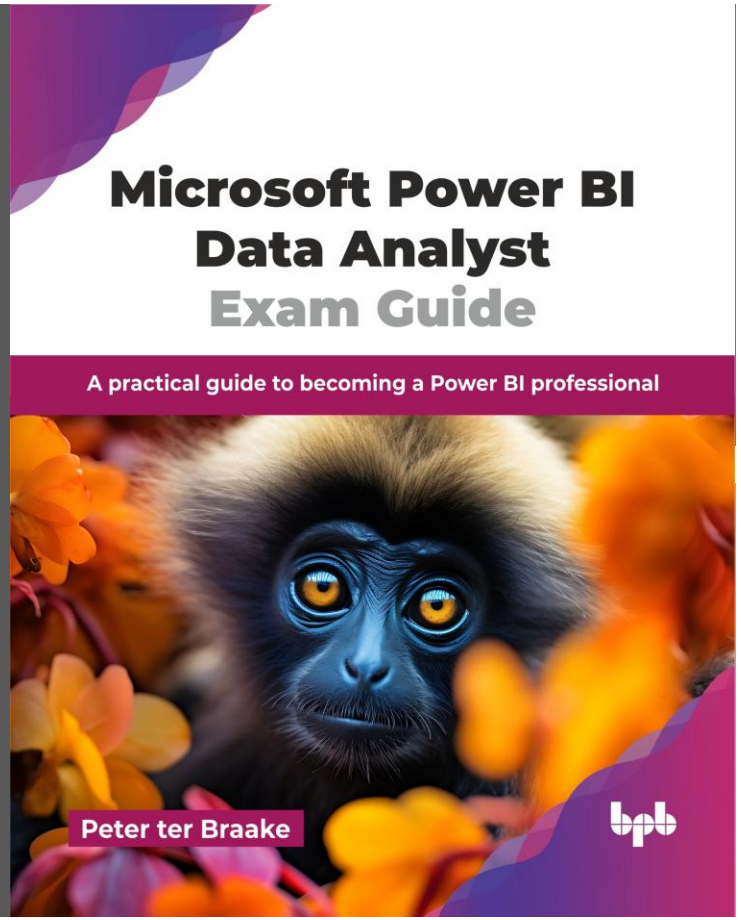
Predict Loan

Questions ?



Thank you!

Download slides:
<add tiny url here>



Peter ter Braake
trainsql@live.nl

15% Discount on PowerBI exam guide:
https://tinyurl.com/39cs9ync + code POWERBI15