



Analytics at scale with Power BI and Azure Synapse



Premium
sponsors

Fellowwind



KONICA MINOLTA

NEXTAGENDA[»]
ACHIEVE MORE WITH YOUR DATA
DIGITAL WORKPLACE SOLUTIONS
FROM KONICA MINOLTA

kapacity

unit^{it}



[stoltze][it]

Initiate



Standard
sponsors



Dave Ruijter

Solution Architect Data & Analytics
Blue Rocket Data Consulting & Solutions

✉ dave@blue-rocket.it

🐦 @DaveRuijter

linkedin.com/in/DaveRuijter

🌐 ModernData.ai



Marc Lelijveld

Data & Analytics consultant
Macaw Netherlands



✉ marc.lelijveld@outlook.com

🐦 @MarcLelijveld

linkedin.com/in/MarcLelijveld

🌐 Data-Marc.com





Break times

Tea and coffee available all day

- 10:30 to 11:00 Refreshments with snacks
- 12:30 to 13:30 Lunch
- 15:30 to 16:00 Afternoon break with snacks



Agenda

- Challenges
- Deep Dive
- Lunch
- Hybrid Tables & Incremental Refresh
- Refresh Challenges & Orchestration
- Backup & Restore
- Scaling
- Monitoring
- Wrap-up



After this session

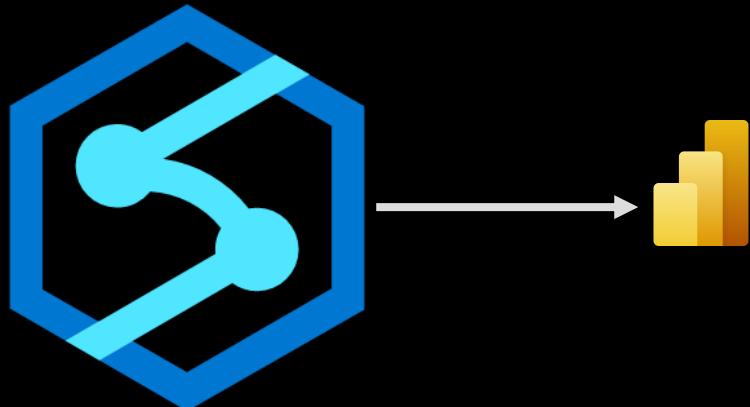
Design and implement	Orchestrate	Performance	Cost
Better design and implement complex data models, including hybrid tables, aggregations, and combined storage modes (import, DirectQuery , dual).	Orchestrate the end-to-end data processing, with a pipeline chain from data ingest in the data lake house to the incremental Power BI dataset refresh.	Use different techniques to identify performance bottlenecks in your solutions and how to solve those ("does it fold"?).	Implement a cost-efficient solution, that still meets the scalability demands.



Solution challenges

two separate worlds

Data Platform



Power BI







Data Platform Solution Challenges

- Power BI report can't handle the volume of data
- Showing near real-time data in Power BI report



Power BI report can't handle the volume

The screenshot shows the Power BI Desktop interface with the title bar "WWI Sales Model - full IMPORT - Power BI Desktop". The Home tab is selected in the ribbon. A central message box says "Unable to save document" and "Power BI Desktop ran out of memory trying to save the data model". The ribbon also displays "1bn" and "1 Sales". To the right, there are sections for Filters, Visualizations, Fields, and a list of fields including City, Customer, Date, Employee, Movements, Orders, Payment Method, Purchases, Sales, Stock Holdings, and Stock Item. The status bar at the bottom right shows "Storage Mode: Mixed".

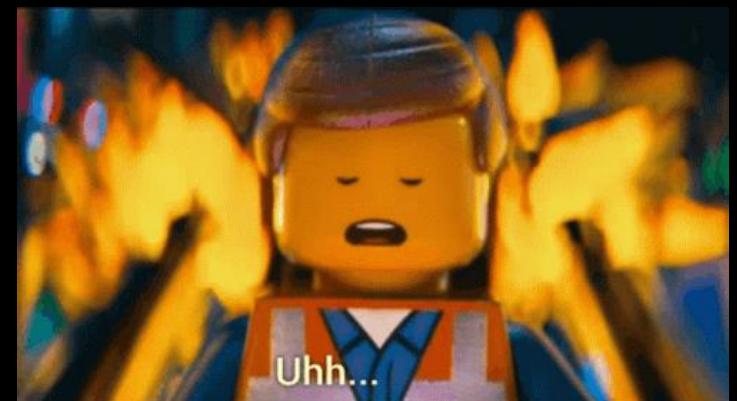


Demo

Power BI report can't handle the volume of data



Should I put everything on DirectQuery instead?





DirectQuery limitations

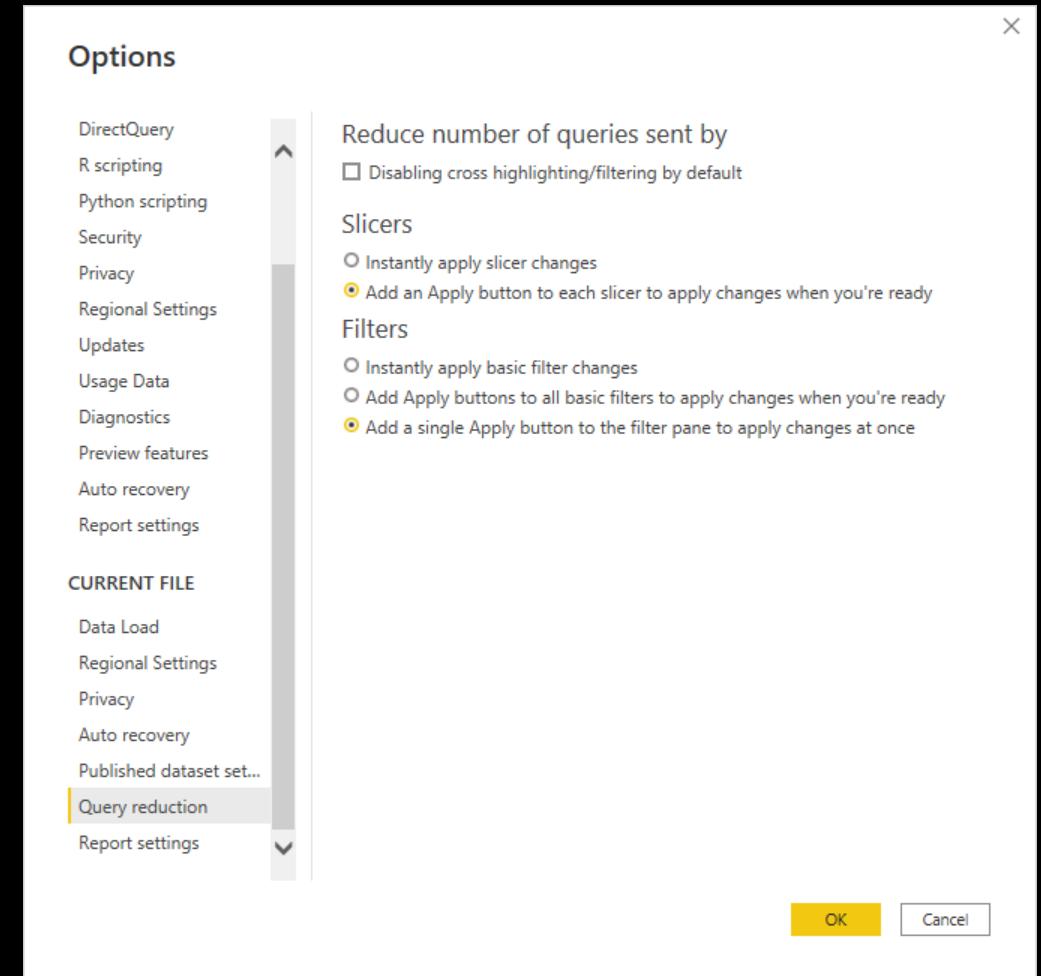
- Limited Power Query capabilities
- DirectQuery != streaming / live! Front-end still caches data
- No built-in date hierarchy (year/quarter/month/day)
- Lowest granularity data is seconds (no milliseconds)
- No parent-child support in DAX with *PATH()*
- Slower end user performance 
- 1M row per query
- DAX limitations, only simple calculations possible

DirectQuery query reduction

Consider requesting to click **Apply** before queries are executed to the source

Applies to

- Slicers
- Filters (filter pane)





Showing near real-time data in
Power BI



Showing near real-time data in Power BI

- Refresh takes too long
- Poor end-user performance on DirectQuery
- Streaming datasets only allow one table
- Potentially queries are not foldable, therefore incremental refresh does not work (depending on source)



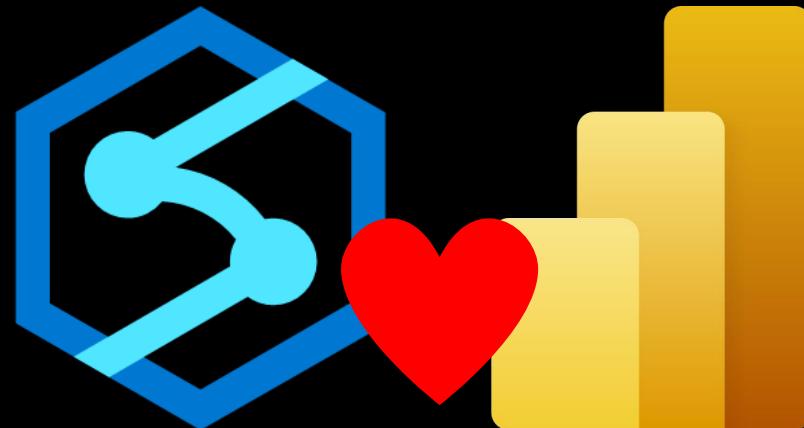
Demo

Showing near real-time data in Power BI report



Data Platform Solution Challenges

- Optimize Power BI model
- Use Aggregations
- Use Hybrid Table





Power BI Solution Challenges

- Loading data from various sources (flat files, databases, APIs)
- Some sources are manually maintained (like mapping tables)
- Data stored on decentralized storages, like SharePoint pages.
- Data from source systems are exported, rather than connected to analytical systems
- Store historical data in Power BI for trend analysis





OrderDate
Last ▾ 1 Select ▾
No filters applied

Product Category
All ▾

CountryRegion
 Canada
 United Kingdom
 United States

StateProvince, City
 Alberta
 Arizona
 British Columbia
 Brunswick

\$708,69K

\$ Sales

2.087

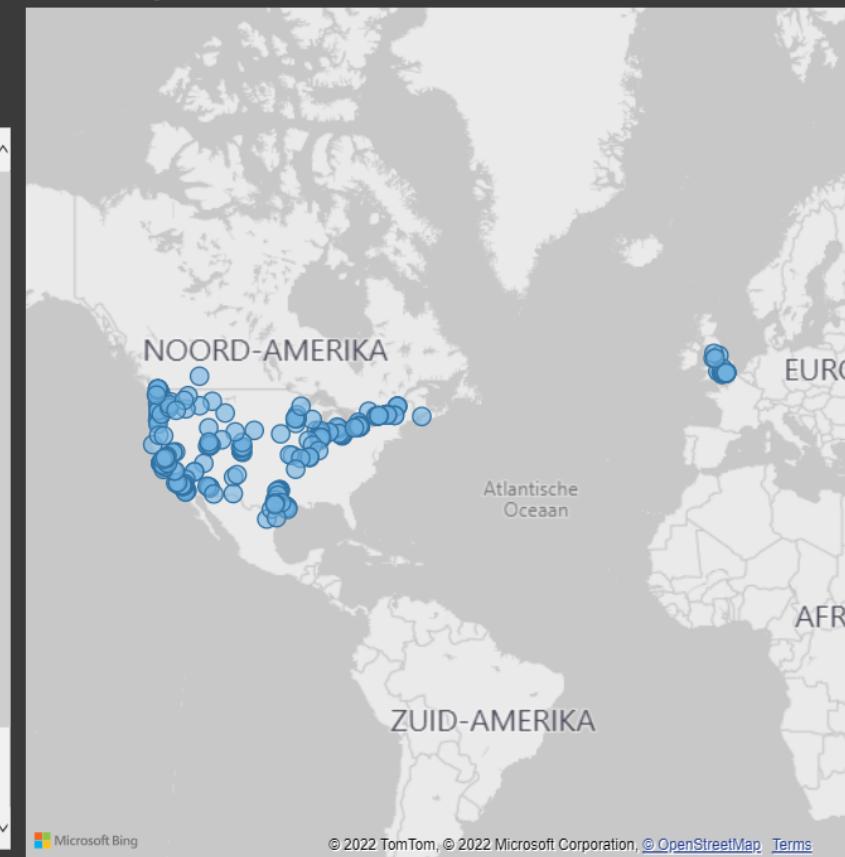
Order quantity

32

Orders

Product Category	# Order quantity	\$ Sales	% GT \$ Sales	\$ Price Avg per product
Touring Bikes	252	\$220.655,38	31,14%	\$875,62
Road Bikes	222	\$183.130,30	25,84%	\$824,91
Mountain Bikes	209	\$170.825,89	24,10%	\$817,35
Mountain Frames	128	\$54.949,60	7,75%	\$429,29
Road Frames	60	\$24.346,58	3,44%	\$405,78
Touring Frames	39	\$19.066,26	2,69%	\$488,88
Jerseys	230	\$7.017,88	0,99%	\$30,51
Vests	121	\$4.309,90	0,61%	\$35,62
Cranksets	22	\$3.968,87	0,56%	\$180,40
Shorts	80	\$3.299,80	0,47%	\$41,25
Pedals	84	\$2.996,50	0,42%	\$35,67
Helmets	124	\$2.523,88	0,36%	\$20,35
Bike Racks	32	\$2.304,00	0,33%	\$72,00
Hydration Packs	50	\$1.649,70	0,23%	\$32,99
Bottom Brackets	22	\$1.320,17	0,19%	\$60,01
Deraileurs	21	\$1.296,63	0,18%	\$61,74
Handlebars	27	\$1.192,97	0,17%	\$44,18
Saddles	39	\$1.010,30	0,14%	\$25,91
Gloves	57	\$837,56	0,12%	\$14,69
Brakes	13	\$830,70	0,12%	\$63,90
Socks	66	\$345,21	0,05%	\$5,23
Caps	52	\$277,36	0,04%	\$5,33
Cleaners	55	\$251,88	0,04%	\$4,58
Total	2.087	\$708.690,15	100,00%	\$339,57

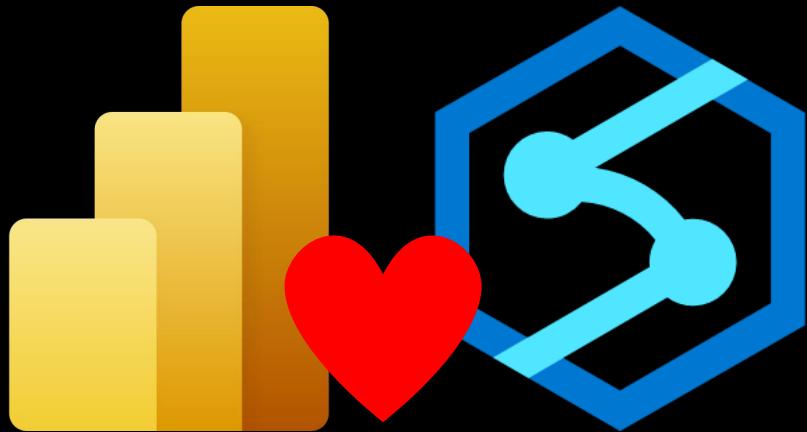
\$ Sales by PostalCode





Power BI Solution Challenges

- Use Synapse Analytics to ingest into a data lakehouse
- Use Layered approach: Bronze, Silver & Gold



Better together

Data platform deep dive





Improvement areas

- Ingesting data from APIs using Synapse Analytics
- Store (historical!) data in the delta lakehouse architecture

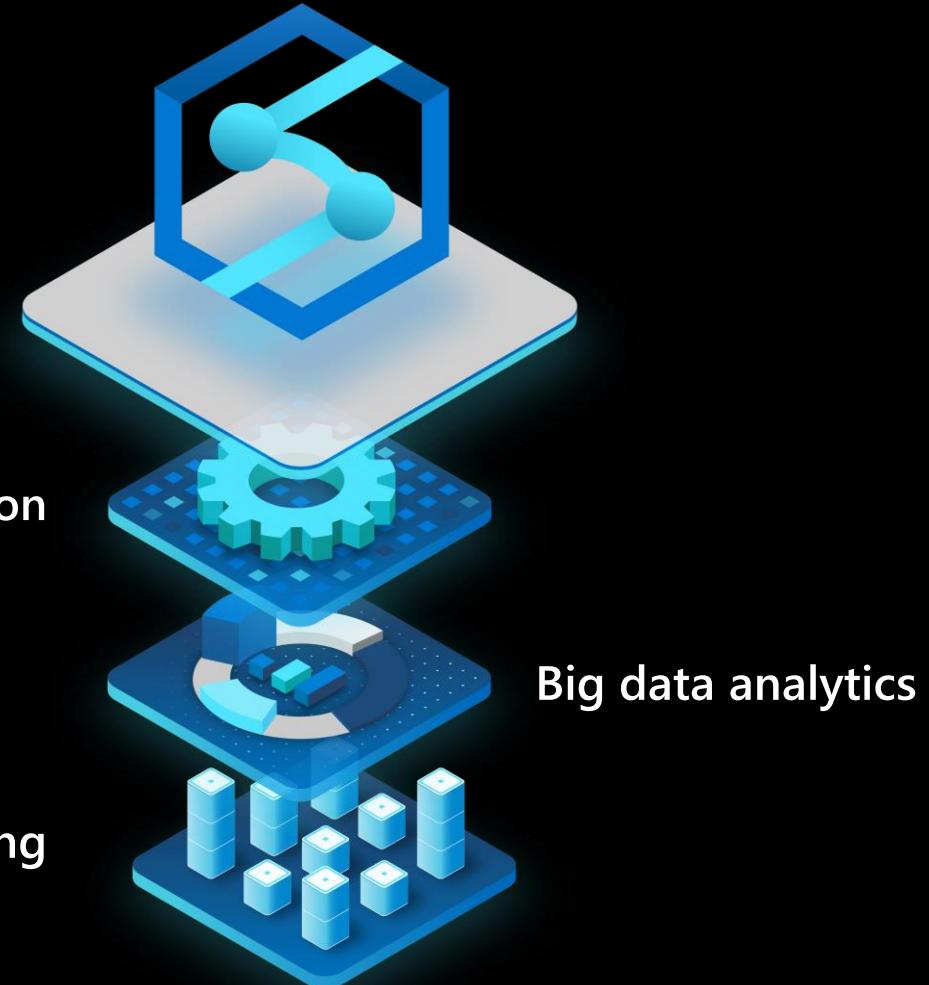
Azure Synapse Analytics

The first unified, cloud native platform for converged analytics

Azure Synapse is the only unified platform for analytics, blending big data, data warehousing, and data integration into a single cloud native service for end-to-end analytics at cloud scale.

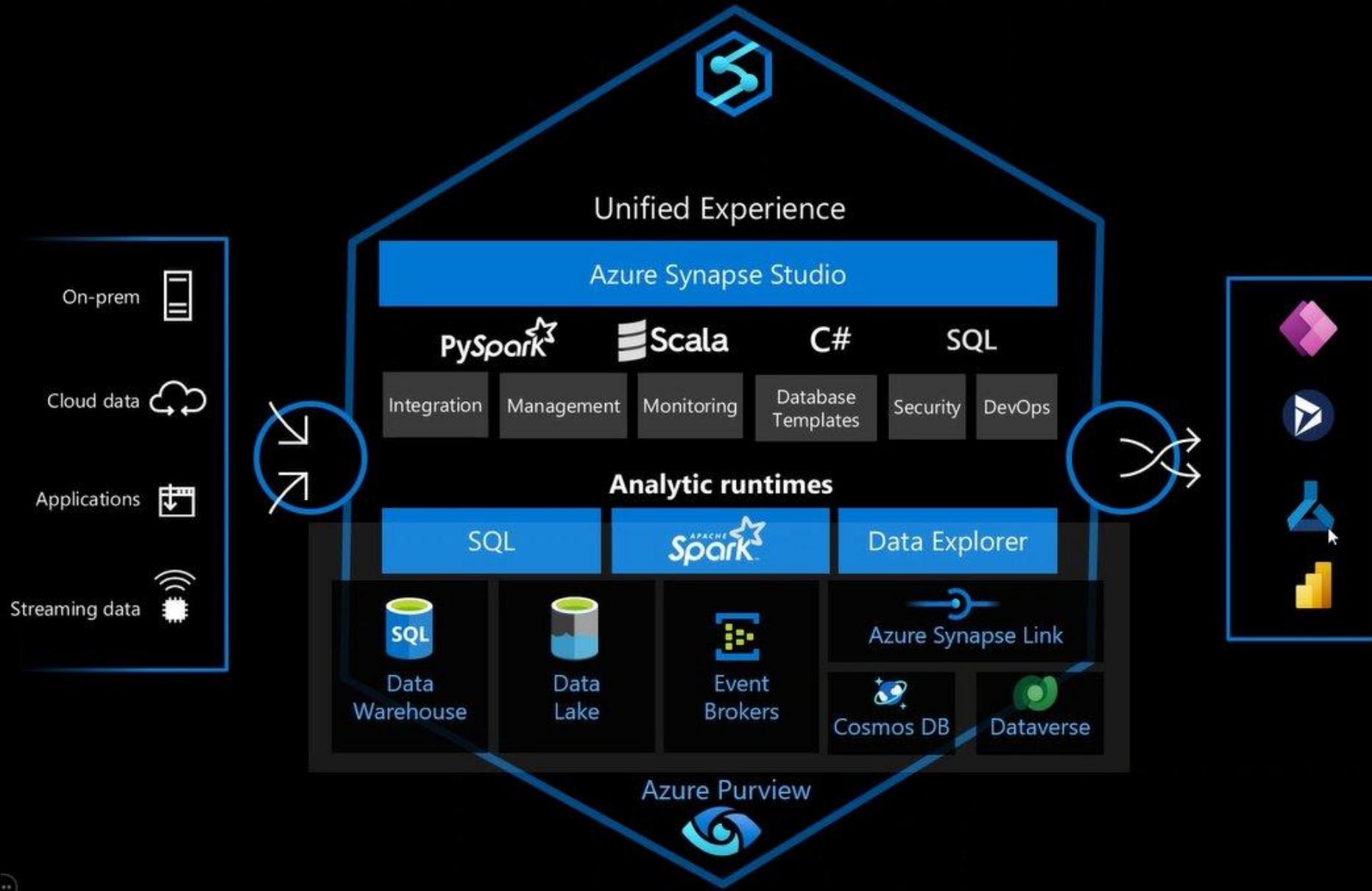
Data integration

Data warehousing



Azure Synapse Analytics

The first unified, cloud native platform for converged analytics





Ingesting data from APIs using Synapse Analytics

- Pipeline (same as ADF)
- Data Flow (same as ADF) *not to be confused with dataflows in PBI*
- Spark Notebook (4 languages available)
- Wrangling Dataflows (Same as ADF)



Hybrid data integration

Cloud native ETL/ELT

95+ connectors available

Secure connectivity to on-premise data sources, other clouds, and SaaS applications

Code-first and low/no code design interfaces

Schedule and Event based triggering



Code-free



Code-free data wrangling

No/low-code data transformation

Excel-like interface is familiar to users

Transform data to desired shape completely visually

Operationalize into pipelines

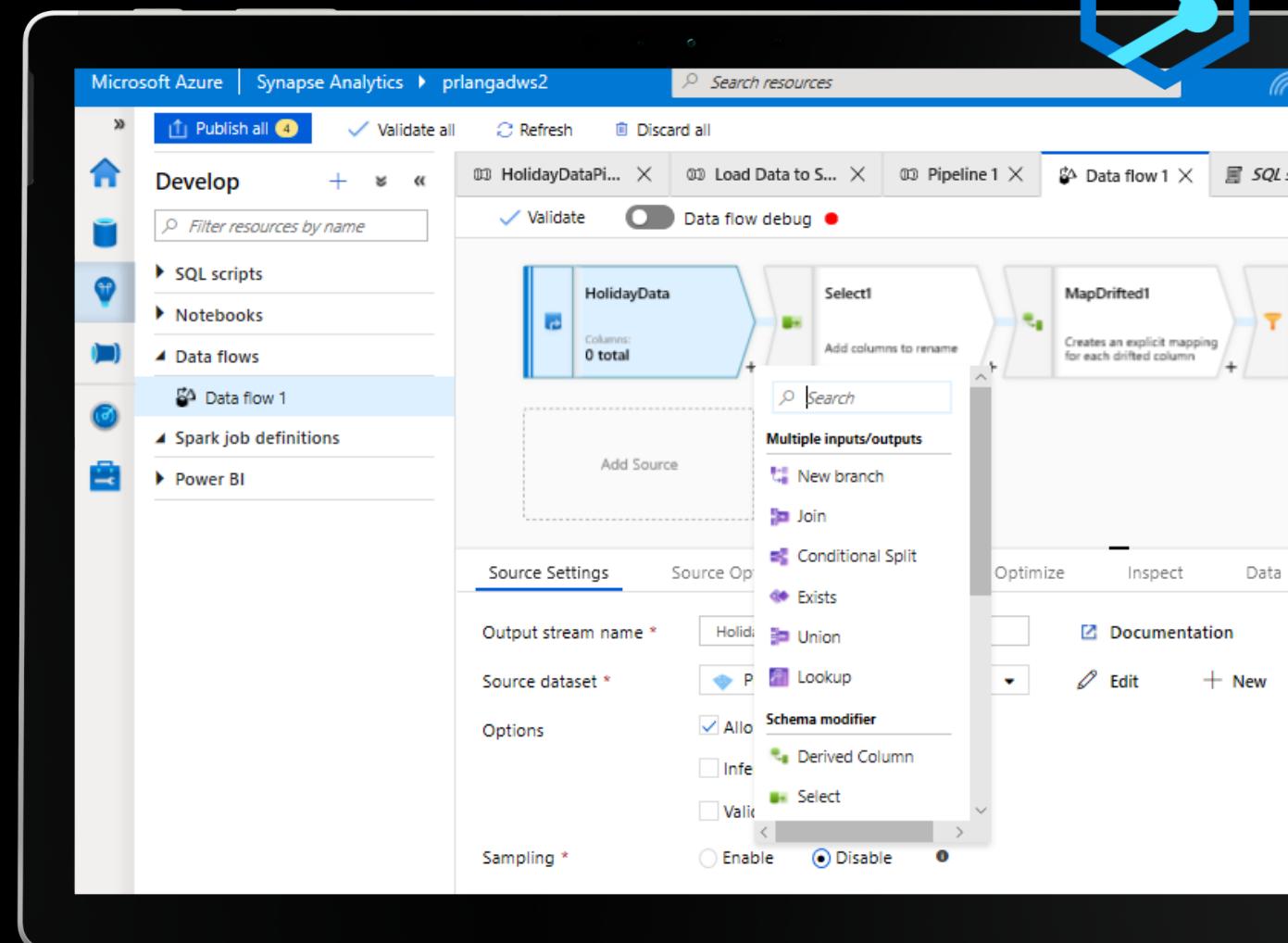
The screenshot shows the Microsoft Azure Synapse Analytics Power Query Editor interface. The top navigation bar includes 'Microsoft Azure', 'Synapse Analytics', 'wsazuresynapseanalytics', and a search bar. Below the navigation is a toolbar with various icons for validation, publishing, and discarding. The main workspace is titled 'PQSalesPrep' and shows a 'Settings' section with a note about Power Query M functions. The 'Home' tab is selected, displaying a ribbon of tools: Enter data, Options, Manage parameters, Refresh, Advanced editor, Properties, Choose columns, Remove columns, Manage columns, Keep rows, Remove rows, Reduce rows, Sort, Split column, Group by, Replace values, and Transform. To the right of the ribbon is a data preview pane showing a table with 17 rows and 8 columns. The columns are labeled: ab storeId, ab productCode, 12 quantity, 1.2 logQuantity, ab advertising, ab price, ab weekStarting, and ab id. The data consists of various surface.go entries with different quantities and prices. At the bottom of the preview pane, it says 'Columns: 8 Rows: 99+'. The overall interface is clean and modern, designed for data wrangling and transformation.



Hybrid data integration

Data Flows

not to be confused with dataflows in Power BI





Demo API ingest



Store (historical!) data in the lake

- The Layered approach: Bronze, Silver & Gold
- Keep original raw data, build up history -> **bronze**
- Cleanse and refine data, standard file format -> **silver**
- Aggregate, prepare, transform, merge, make start schema -> **gold**



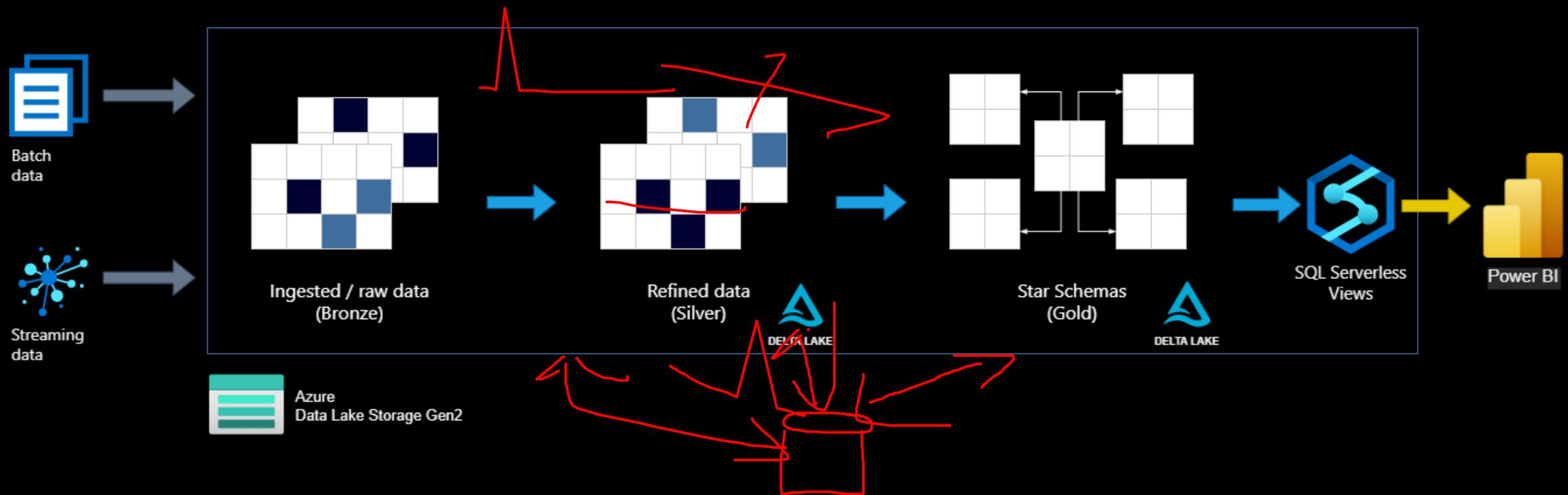
Store (historical!) data in the lake

- The Layered approach:
Bronze, Silver & Gold
- Keep original raw data, build up
history -> **bronze**
- Cleanse and refine data, standard file
format -> **silver**
- Aggregate, prepare, transform, merge,
make start schema -> **gold**

Bronze	Silver	Gold
Raw data	Apply metadata	Implement business rules
All history, system replayable	Protect data (GDPR)	Fit for purpose
	Current & historical view	



Using Delta Lakehouse with layered approach





Warm-up time of Serverless SQL pools

SQL requests

Refresh Edit columns

Filter by keyword

Local time : **Last 24 hours**

Status : **All**

Pool : **Built-in**

Add filter

Showing 1 - 4 of 4 items

Request ID ↑	Request content ↑	Submit time ↑	Duration	Data processed
44196195	SELECT TOP (1000... More	3/6/22, 10:45:14 AM	14 sec	5.12 GB
44190548	SELECT TOP (1000... More	3/6/22, 10:43:13 AM	16 sec	4.34 GB
44187683	SELECT TOP (1000... More	3/6/22, 10:42:32 AM	14 sec	6.12 GB
44178591	SELECT TOP (1000... More	3/6/22, 10:39:55 AM	20 sec	6.11 GB

Sales Territory	\$ Sales
External	\$12,540,565,587.63
Far West	\$130,724,597,536.11
Great Lakes	\$137,062,180,086.18
Mideast	\$170,472,095,945.05
New England	\$58,853,530,841.09
Plains	\$154,405,133,753.92
Rocky Mountain	\$72,387,209,160.70
Southeast	\$258,110,199,285.25
Southwest	\$141,628,285,606.20
Total	\$1,136,183,797,802.13



Best practices for serverless SQL pools

- Azure AD Pass-through Authentication performance <= shared access signature credentials
- Colocate
- Same region
- Convert large CSV and JSON files to Parquet
- Try to optimize storage layout by using partitioning and keeping your files in the range between 100 MB and 10 GB
- Use appropriate data types (smallest, integer-based, varchar)
- Use filename and filepath functions to target specific partitions



Demo Data Platform Lakehouse architecture



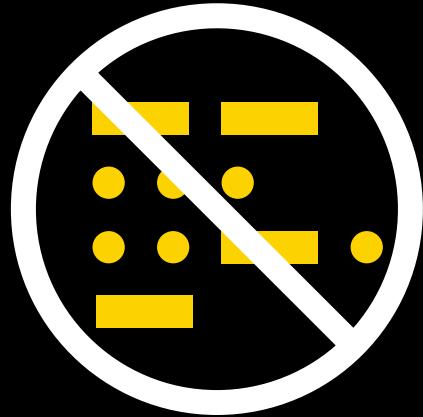
What about Datamarts?



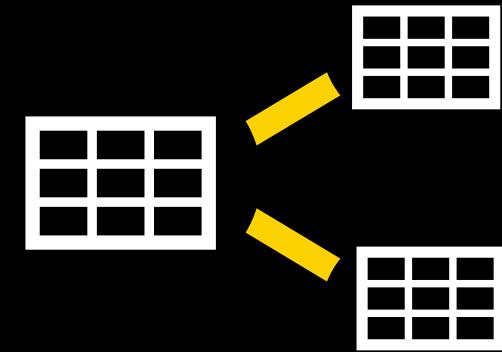
What are Datamarts?



Self service relational database capabilities without IT involvement



No-code experience for data ingestion, preparation and transformation

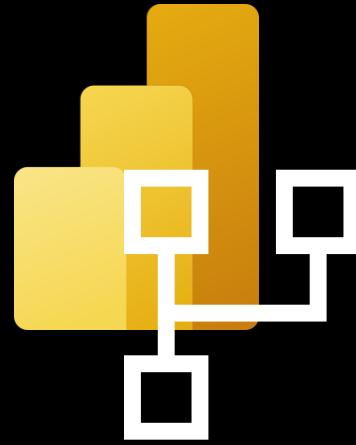


Build semantic models in a holistic cloud experience

What is it NOT?



Replacement for
Power BI datasets



Replacement for
Power BI dataflows



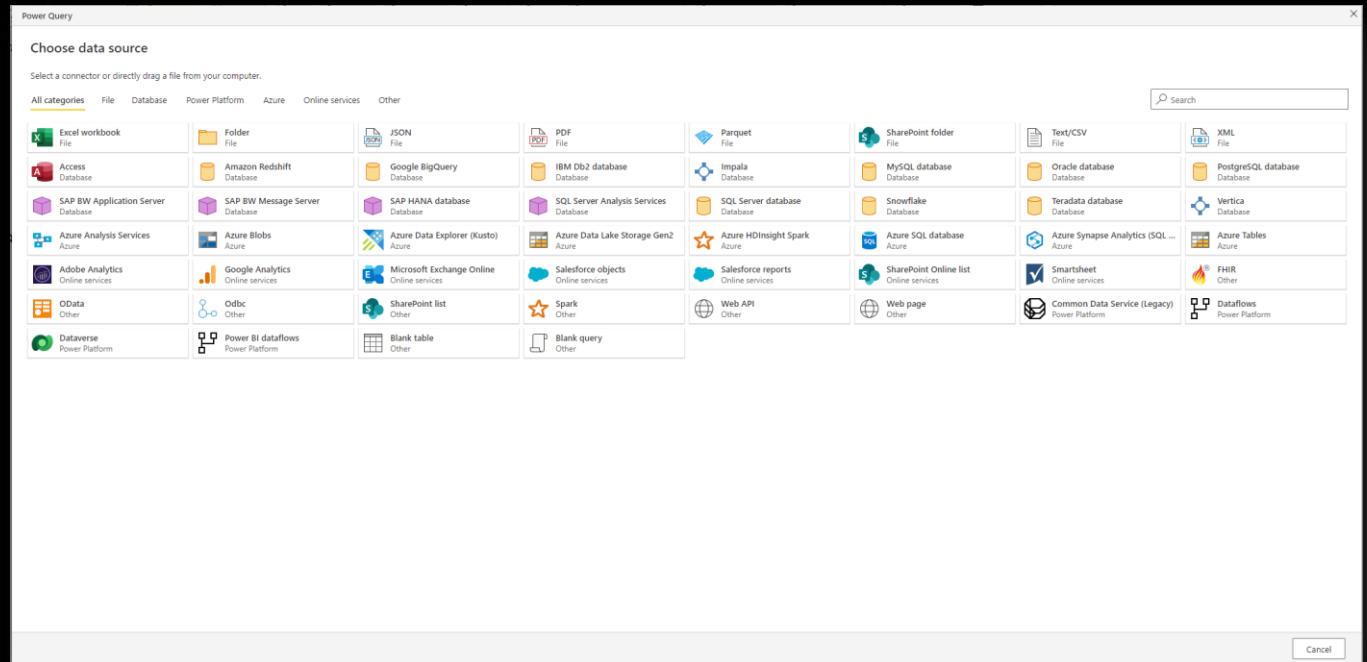
Replacement for
Power BI Desktop



Replacement your
data platform / DWH

Datamarts or dataflows?

Ingest new sources, all the Power Query connectors are available!



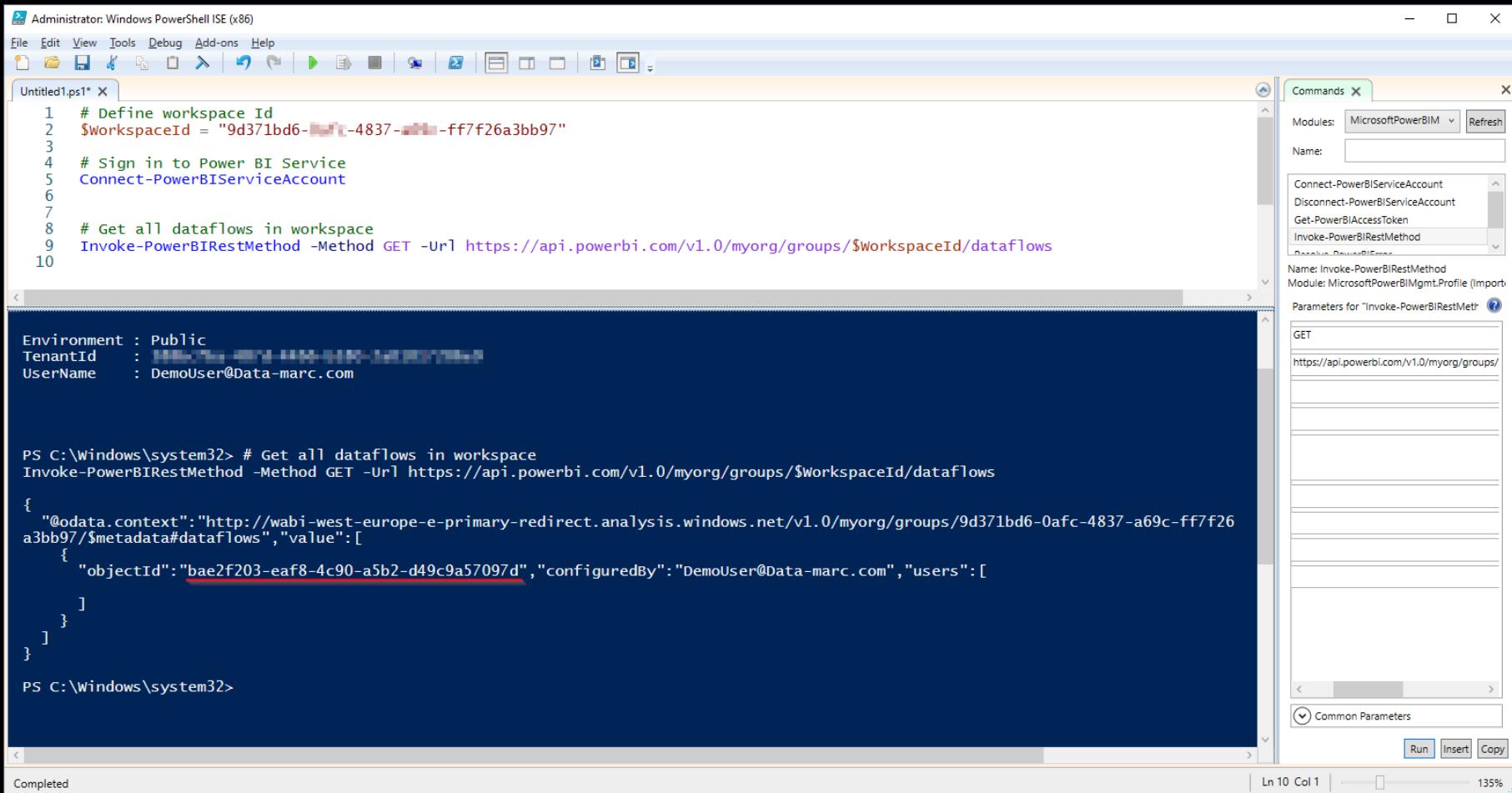
Datamarts or dataflows?

The web-based experience we all know from dataflows to transform data. But now data is loaded in a fully managed Azure SQL Database.

The screenshot shows the Microsoft Power Query Editor interface. The top menu bar includes Home, Transform, Add column, View, and Help. The ribbon below the menu has tabs for Get data, Enter data, Options, Manage parameters, Refresh, Advanced editor, Query, Manage columns, and Transform. The main area displays a table titled 'Table.RemoveColumns(#"Navigation 2", "rowguid")' with 99 rows and 19 columns. The columns include ProductID, Name, Color, StandardCost, ListPrice, Size, Weight, ProductCategoryID, ProductModelID, SellStartDate, SellEndDate, DiscontinuedDate, and ThumbnailPhoto. The 'Transform' tab is selected. On the right side, there are sections for 'Query settings', 'Properties', 'Name', 'Product', and 'Applied steps'. The 'Applied steps' section shows a history of transformations: 'Source', 'Navigation 1', 'Navigation 2', and 'Removed ...'. At the bottom, buttons for 'Step' and 'Save' are visible.

Magic uncovered?

Each table, is first saved as dataflow before being part of the Datamart.



The screenshot shows a Windows PowerShell ISE window with the following content:

```
Administrator: Windows PowerShell ISE (x86)
File Edit View Tools Debug Add-ons Help
Untitled1.ps1* X
1 # Define workspace Id
2 $workspaceId = "9d371bd6-[REDACTED]-4837-[REDACTED]-ff7f26a3bb97"
3
4 # Sign in to Power BI Service
5 Connect-PowerBIServiceAccount
6
7
8 # Get all dataflows in workspace
9 Invoke-PowerBIRestMethod -Method GET -Url https://api.powerbi.com/v1.0/myorg/groups/$workspaceId/dataflows
10
```

Output pane:

```
Environment : Public
TenantId    : [REDACTED]
UserName    : DemoUser@Data-marc.com

PS C:\Windows\system32> # Get all dataflows in workspace
Invoke-PowerBIRestMethod -Method GET -Url https://api.powerbi.com/v1.0/myorg/groups/$workspaceId/dataflows
{
  "@odata.context": "http://wabi-west-europe-e-primary-redirect.analysis.windows.net/v1.0/myorg/groups/9d371bd6-0afc-4837-a69c-ff7f26a3bb97/$metadata#dataflows", "value": [
    {
      "objectId": "bae2f203-eaf8-4c90-a5b2-d49c9a57097d", "configuredBy": "DemoUser@Data-marc.com", "users": [
        ]
    }
  ]
}

PS C:\Windows\system32>
```

A Commands pane on the right shows the `Invoke-PowerBIRestMethod` command with its parameters set to GET and URL `https://api.powerbi.com/v1.0/myorg/groups/`.

Magic uncovered?

Sources of the dataflows listed easily

```
PS C:\Windows\system32> # Get all dataflows in workspace
$listDataflows = Invoke-PowerBIRestMethod -Method GET -Url https://api.powerbi.com/v1.0/myorg/groups/$WorkspaceId/dataflows | ConvertFrom-Json
$listDataflows.value

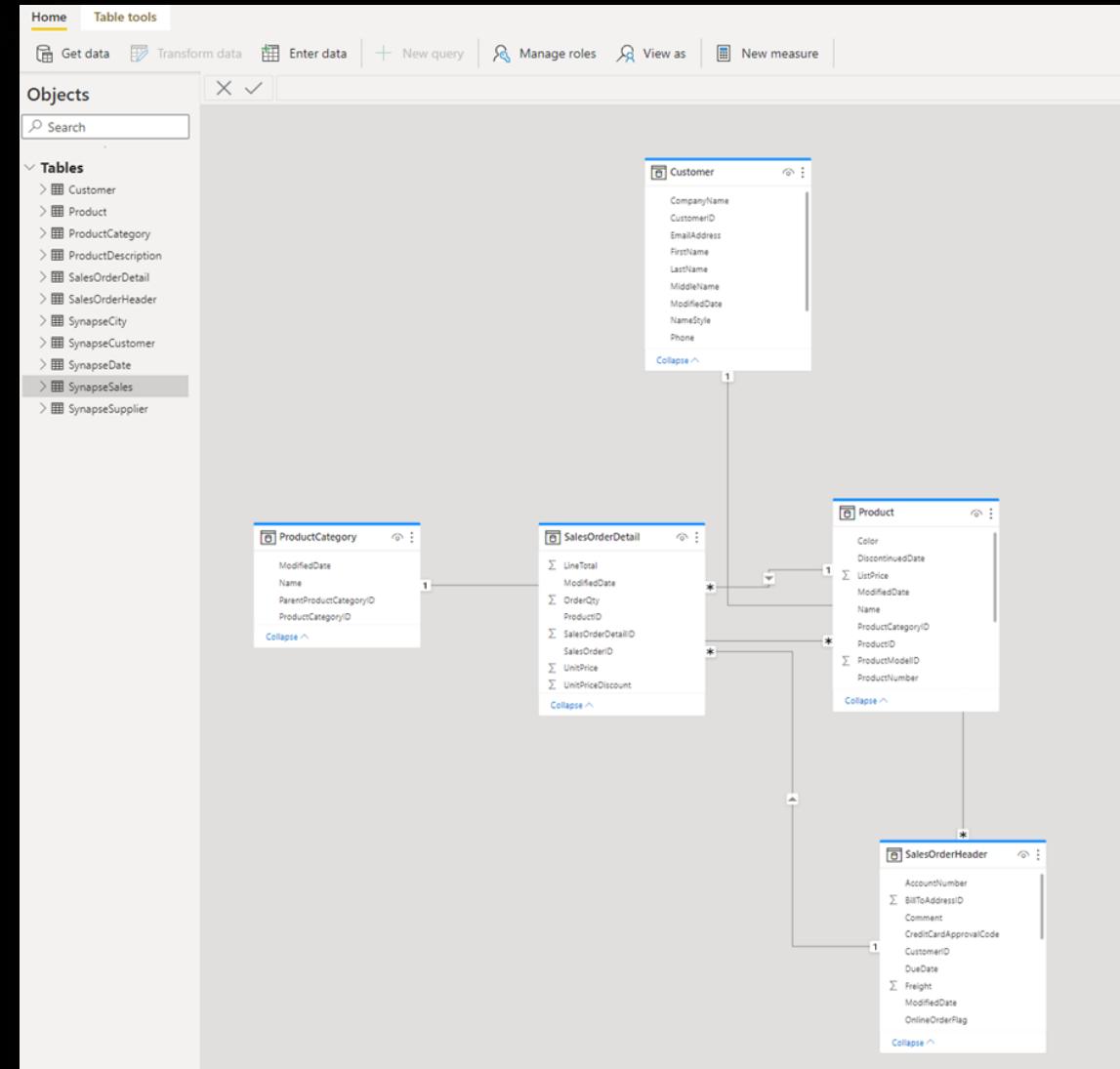
objectID          configuredBy      users
-----          -----
bae2f203-[REDACTED]-a5b2-d49c9a57097d DemoUser@Data-marc.com {}

PS C:\Windows\system32> # Get Dataflow sources
$dataFlowObjectId = $listDataflows.value.objectId
Invoke-PowerBIRestMethod -Method GET -Url https://api.powerbi.com/v1.0/myorg/groups/$WorkspaceId/dataflows/$dataFlowObjectId/datasources

{
  "@odata.context": "http://wabi-west-europe-e-primary-redirect.analysis.windows.net/v1.0/myorg/groups/9d371bd6-[REDACTED]-a69c-ff7f26a3bb97/$meta
data#datasources", "value": [
  {
    "datasourceType": "Sql", "connectionDetails": {
      "server": "[REDACTED].database.windows.net"
    }, "datasourceId": "02bacc67-4570-4c7c-a801-2d47ae9a209c", "gatewayId": "c1d863df-d137-453e-9504-0d1d99268906"
  }, {
    "datasourceType": "Sql", "connectionDetails": {
      "server": "[REDACTED].database.windows.net", "database": "PowerBICheatSheet"
    }, "datasourceId": "dc69e506-3d08-4f50-ba5f-daf90eb1f700", "gatewayId": "c1d863df-d137-453e-9504-0d1d99268906"
  }, {
    "datasourceType": "Sql", "connectionDetails": {
      "server": "[REDACTED]-ondemand.sql.azuresynapse.net"
    }, "datasourceId": "318de039-d12b-48f7-b676-963f2e077a75", "gatewayId": "c1d863df-d137-453e-9504-0d1d99268906"
  }, {
    "datasourceType": "Sql", "connectionDetails": {
      "server": "[REDACTED]-ondemand.sql.azuresynapse.net ", "database": "lake"
    }, "datasourceId": "104ee253-d4d0-4692-90d1-283dfd905642", "gatewayId": "c1d863df-d137-453e-9504-0d1d99268906"
  }
}
```

Modeling experience

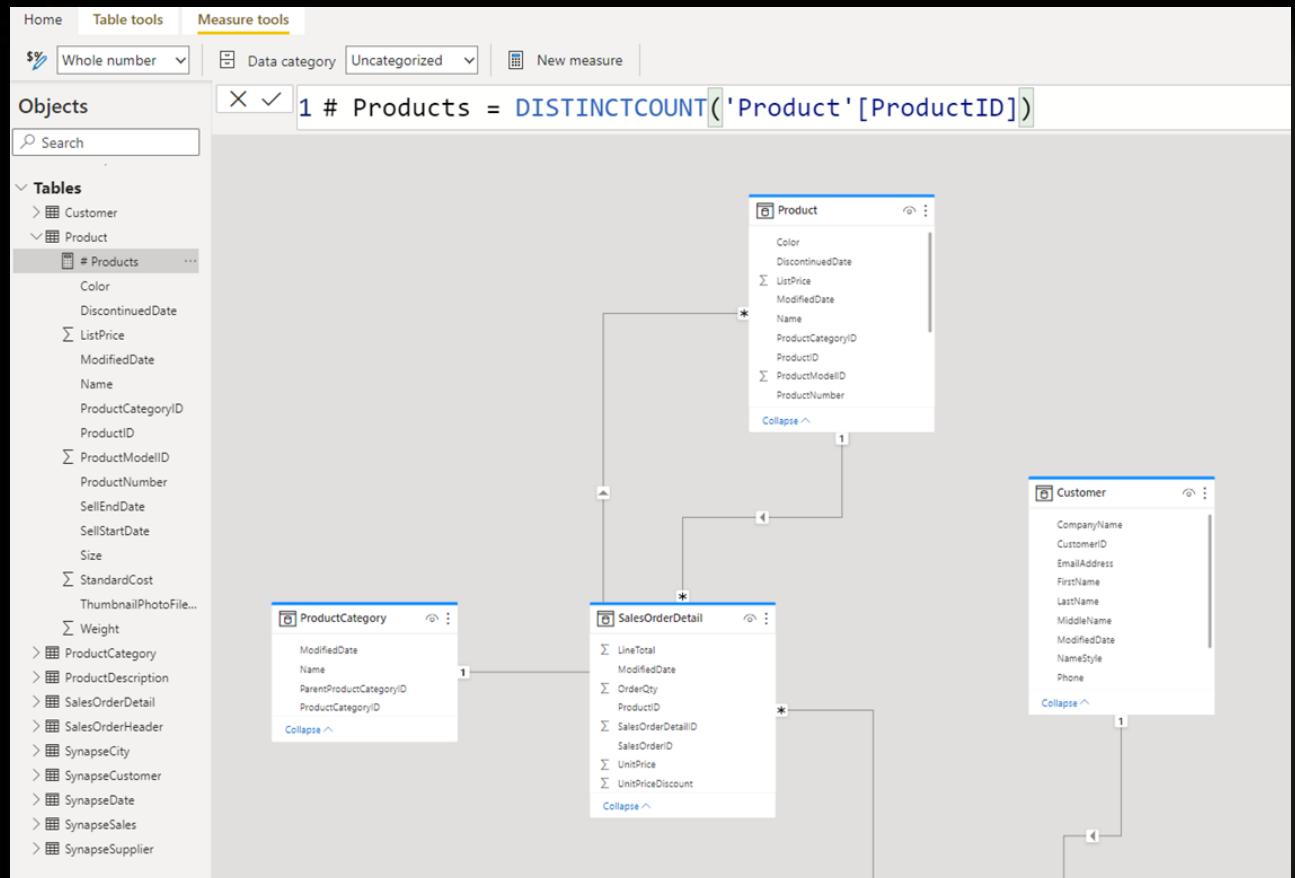
Modeling experience in the service, which allows you to build new relationships, define measure and more!



Measures

Datamarts allow you to build measures directly, as part of the modeling experience.

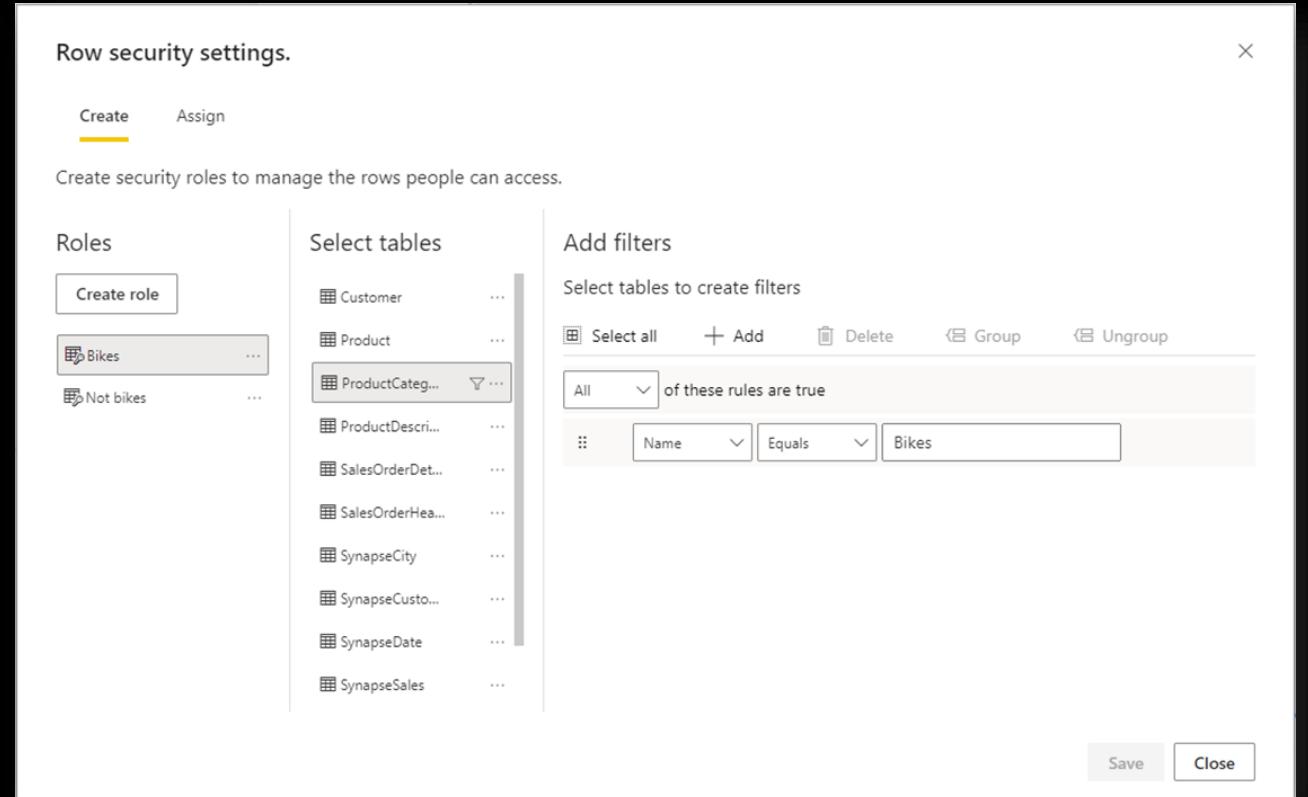
Measures results cannot be displayed inside the datamart.



Security

Simplified experience to setup row level security, without the need to understand DAX.

Object level security is not possible (yet?).



The screenshot shows the 'Row security settings' dialog box. At the top, there are 'Create' and 'Assign' tabs, with 'Create' being the active tab. Below the tabs, a message says 'Create security roles to manage the rows people can access.' On the left, under 'Roles', there is a 'Create role' button and two existing roles: 'Bikes' (selected) and 'Not bikes'. In the center, under 'Select tables', a list of tables is shown: Customer, Product, ProductCateg..., ProductDescri..., SalesOrderDet..., SalesOrderHea..., SynapseCity, SynapseCusto..., SynapseDate, and SynapseSales. To the right, under 'Add filters', there is a section titled 'Select tables to create filters' with a 'Select all' button and a 'Group' button. Below this, a dropdown menu says 'All' and a condition 'Name Equals Bikes' is defined. At the bottom right of the dialog are 'Save' and 'Close' buttons.

Auto-generated dataset

A Power BI dataset is automatically generated in the same workspace, using a DirectQuery connection to your datamart.

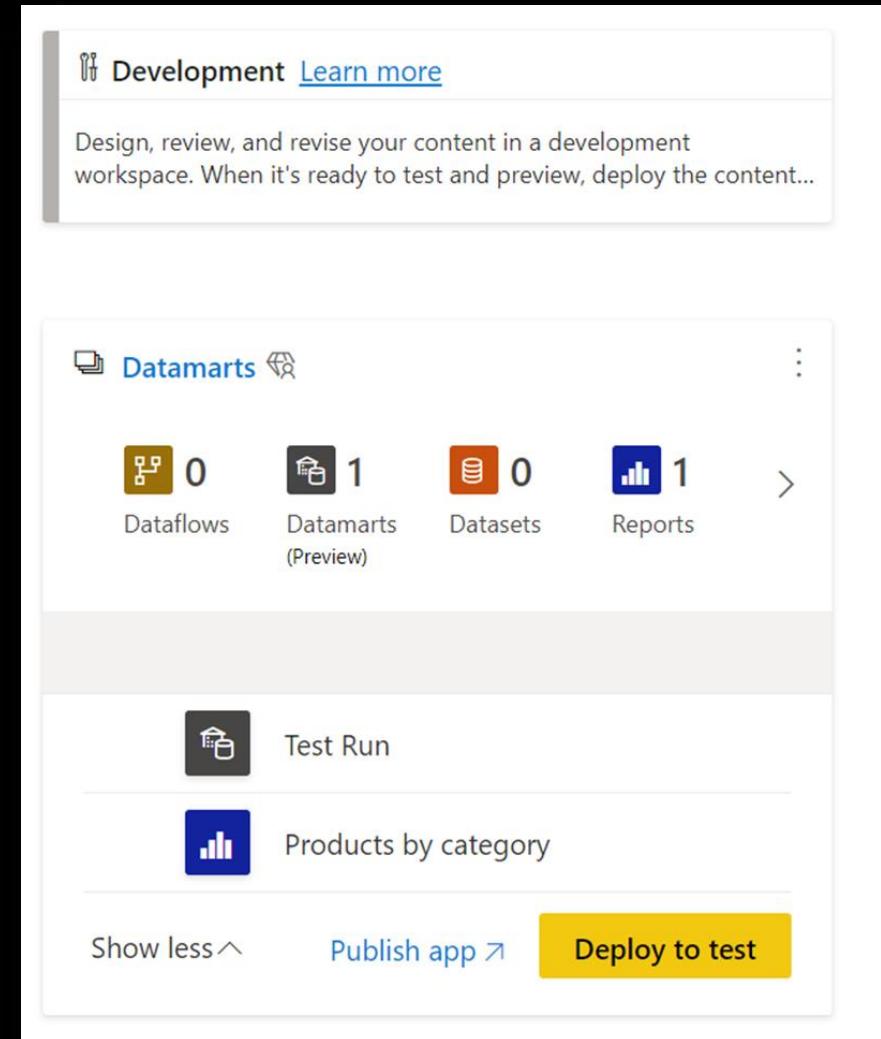
The screenshot shows the Microsoft Power BI Data Mart interface. At the top, there's a circular icon with a person symbol and the text "Datamarts". Below it is a "Create a pipeline" button. A navigation bar includes "All" (which is selected), "Content", "Datasets + dataflows", and "Datamarts (Preview)". The main area displays a table with three columns: "Name", "Type", and "Owner". There are two entries:

Name	Type	Owner
Test Run	Dataset	Datamarts
Test Run	Datamart	Demo User

ALM for datamarts

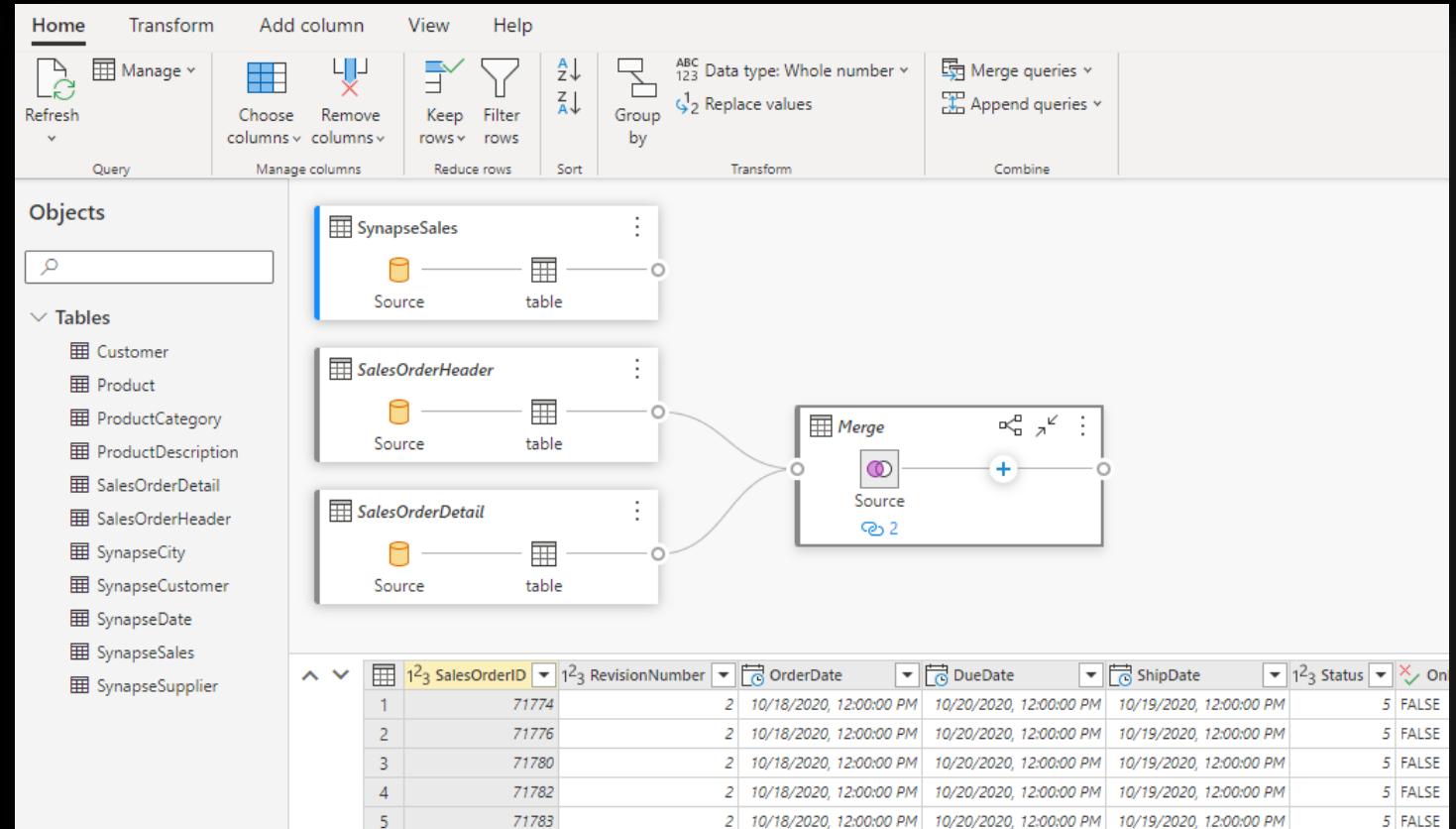
Datamarts can be deployed to different stages using Power BI deployment pipelines.

Notice the auto-generated dataset is not visible in the pipeline.



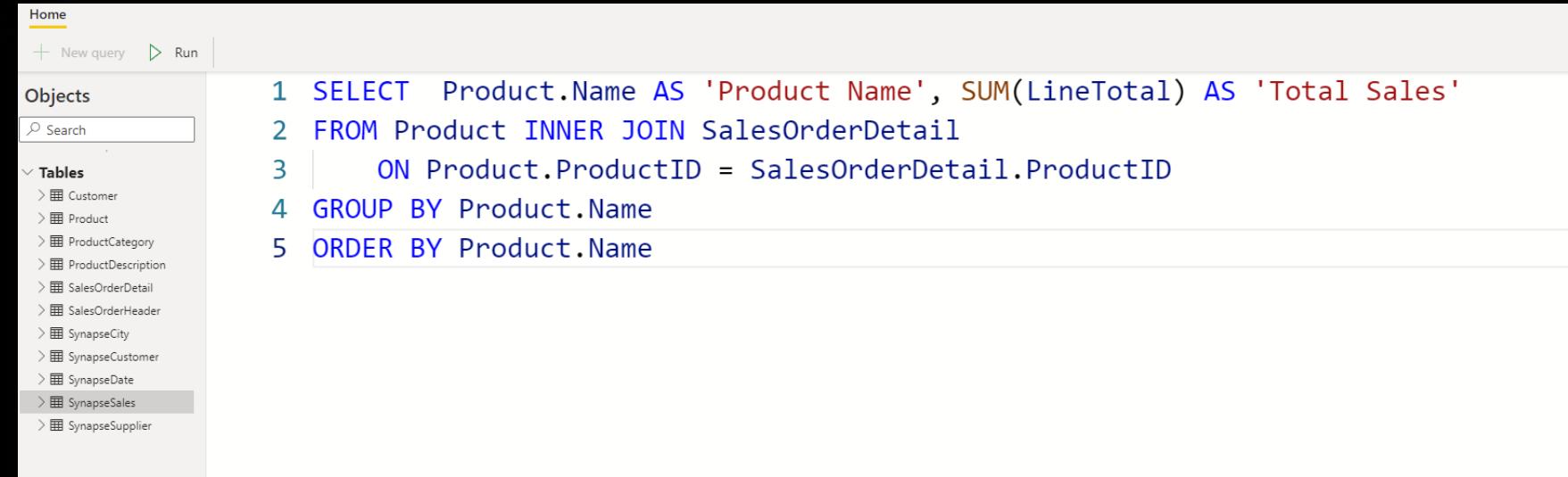
Query experience

Ad-hoc M-queries
(PowerQuery) in a visual
experience.



Query experience

Write on-demand SQL queries in
the same datamart interface.



The screenshot shows a query editor interface. On the left, there's a sidebar titled 'Objects' with a 'Tables' section containing a list of tables: Customer, Product, ProductCategory, ProductDescription, SalesOrderDetail, SalesOrderHeader, SynapseCity, SynapseCustomer, SynapseDate, SynapseSales (which is highlighted), and SynapseSupplier. At the top, there are buttons for 'Home', 'New query', and 'Run'. The main area contains a numbered SQL query:

```
1 SELECT Product.Name AS 'Product Name', SUM(LineTotal) AS 'Total Sales'  
2 FROM Product INNER JOIN SalesOrderDetail  
3 ON Product.ProductID = SalesOrderDetail.ProductID  
4 GROUP BY Product.Name  
5 ORDER BY Product.Name
```



Datamarts

Better together

Power BI deep dive





Improvement areas

- Data model
- Query Folding
- Aggregations
- Storage modes
- Hybrid tables



But before we start changing our solution, let's measure...

- Refresh durations
- Model Size
- Vertipaq Analyzer
- Performance Analyzer
- Query folding applied?

Perfect E2E.pbix			
Total Size 74,89 MB	Last Data Refresh 1-3-2022 20:05:54 +01:00	Analysis Date 1-3-2022 20:05:55 +01:00	
Compatibility 1550	Tables 7	Columns 124	Server localhost:60032



Performance analyzer in Power BI Desktop

The screenshot shows the 'Performance analyzer' window in Power BI Desktop. It lists various tasks and their execution times in milliseconds (ms). The tasks include '0.0%', 'Simple Image', 'Net Sales vs "What If" Analysis', 'OneNote', 'What If...', 'Return Rate', 'Net Sales (Forecast)', 'Extra Profit', 'Card', 'Returns', 'OneNote', 'Button', 'Last Refresh: Jun 30th, 2019 / ...', '\$30,772', '+17.1%', '1715', and 'What If" Analysis Forecast'. The 'Duration (ms)' column is sorted in descending order, with the first few items being the longest. A red box highlights the 'Duration (ms)' column header.

Name	Duration (ms)
0.0%	2279
Simple Image	1440
Net Sales vs "What If" Analysis	4331
OneNote	2391
"What If" Analysis Forecast	2543
Changed a slicer	50
What If...	-
What If...	128
Return Rate	1028
Net Sales (Forecast)	1618
Extra Profit	2046
Card	1425
"What If" Analysis Forecast	1890
Returns	1723
OneNote	2249
	1155
	1723
	1722
	1723
Button	398
Last Refresh: Jun 30th, 2019 / ...	397
\$30,772	2107
+17.1%	1528
	1715
"What If" Analysis Forecast	388
Simple Image	2721

Learn more about optimizing your report's performance on our [support site](#). Find help tuning your report from specialist Power BI partners on [AppSource](#).

- DAX Query
- Visual Display
- Other
 - Preparing queries
 - Waiting for other visuals to complete
 - Other background processes



Vertipaq analyzer

See where your data volume is

VertiPaq Analyzer																		
Tables	Name	Cardinality	Total Size ↓	Data	Dictionary	Hier Size	Encoding	Data Type	RI Violations	User Hier Size	Rel Size	% Table	% DB	Columns	Partitions	Segments	Pageable	Re
Columns	fact_sales	95.698.437	3.550.478.095	1.593.067.7...	1.623.142.9...	333.924.200	Many	-	-	0	343.152	93,49%	23	37	1.058	1.012		
Relationships	dim_product	702.325	148.212.536	11.674.288	120.676.304	15.861.896	Many	-	5	0	48	3,90%	29	1	29	28		
Partitions	fact_orders	786.607	31.441.169	8.799.896	19.509.521	2.906.040	Many	-	-	0	225.712	0,83%	20	8	160	152		
Summary	fact_sales_delivery	835.125	19.332.293	12.489.752	5.919.589	790.408	Many	-	-	0	132.544	0,51%	26	8	208	200		
	dim_calendar	1.126	15.136.510	29.064	15.065.662	41.784	Many	-	13	0	0	0,40%	39	1	39	38		
	fact_sales_consumer...	355.266	11.410.007	4.168.504	6.912.887	304.448	Many	-	-	0	24.168	0,30%	28	8	224	216		
	fact_sales_budget_w...	292.939	5.205.688	2.160.552	2.172.840	870.632	Many	-	-	0	1.664	0,14%	22	8	176	168		
	fact_sales_hd_budg...	436.324	4.052.572	994.688	2.688.172	366.240	Many	-	-	0	3.472	0,11%	12	8	96	88		
	fact_ga_product_vie...	537.616	3.287.830	1.575.408	1.281.262	327.008	Many	-	-	0	104.152	0,09%	7	8	56	48		
	dim_product_ecateg...	39	2.133.436	64	2.132.956	416	Many	-	2	0	0	0,06%	4	1	4	3		
	fact_sales_budget_d...	257.436	1.770.948	1.005.456	193.980	570.520	Many	-	-	0	992	0,05%	12	8	96	88		
	dim_store	1.045	1.692.785	19.744	1.614.505	58.528	Many	-	4	0	8	0,04%	31	1	31	30		
	dim_countries	8	1.065.912	8	1.065.824	80	Many	-	4	0	0	0,03%	2	1	2	1		
	fact_store_traffic	284.585	898.358	774.544	87.422	34.144	Many	-	-	0	2.248	0,02%	5	8	40	32		
	fact_ga_sessions	325.171	800.778	570.200	160.906	65.440	Many	-	-	0	4.232	0,02%	6	8	48	40		
	fact_ecommerce_BU...	37.944	449.848	204.440	133.104	111.872	Many	-	-	0	432	0,01%	11	8	88	80		
	fact_ecommerce_for...	4.464	318.524	16.120	268.396	33.600	Many	-	-	0	408	0,01%	10	8	80	72		
	fact_sales_budget_d...	30.132	221.814	103.048	103.646	15.048	Many	-	-	0	72	0,01%	14	8	112	104		

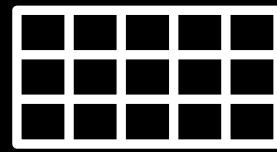
Tables	Name	Cardinality	Total Size ↓	Data	Dictionary	Hier Size	Encoding	Data Type	RI Violations	User Hier Size	Rel Size	% Table	% DB	Co
Columns	fact_sales	95.698.437	3.550.478.095	1.593.067.7...	1.623.142.9...	333.924.200	Many	-	-	0	343.152	93,49%		
Relationships	transaction_id	35.848.228	2.097.266.232	378.951.376	1.431.529.016	286.785.840	HASH	Double	-	-	-	59,08%	55,22%	
Partitions	receipt_number	2.889.572	391.942.672	380.384.240	136	11.558.296	VALUE	Int64	-	-	-	11,04%	10,32%	
Summary	product_id	127.796	217.518.748	213.841.864	2.654.500	1.022.384	HASH	Int64	-	-	-	6,13%	5,73%	
	[REDACTED]	2.147.941	188.167.733	76.707.760	94.276.437	17.183.536	HASH	String	-	-	-	5,30%	4,95%	
	gross_margin	29.248	149.055.816	147.460.408	1.361.408	234.000	HASH	Double	-	-	-	4,20%	3,92%	
	store_id	745	114.206.090	114.169.472	30.650	5.968	HASH	String	-	-	-	3,22%	3,01%	
	order_id	1.517.113	92.295.229	24.397.824	55.760.493	12.136.912	HASH	String	-	-	-	2,60%	2,43%	
	net_revenue	21.058	63.140.880	61.728.376	1.244.024	168.480	HASH	Double	-	-	-	1,78%	1,66%	
	gross_revenue	21.783	62.888.240	61.456.240	1.257.728	174.272	HASH	Double	-	-	-	1,77%	1,66%	
	vat_amount	9.281	59.433.856	58.744.256	615.344	74.256	HASH	Double	-	-	-	1,67%	1,56%	
	customer_email_id	558.750	51.952.377	13.888.256	33.594.105	4.470.016	HASH	String	-	-	-	1,46%	1,37%	
	gross_total_markdown	10.958	28.449.208	27.723.848	637.680	87.680	HASH	Double	-	-	-	0,80%	0,75%	
	transaction_date	355	21.011.840	20.989.424	19.568	2.848	HASH	DateTime	-	-	-	0,59%	0,55%	
	product_line_number	887	5.982.752	5.955.408	20.240	7.104	HASH	Int64	-	-	-	0,17%	0,16%	
	vat_percentage	1.094	4.264.760	4.212.584	43.408	8.768	HASH	Double	-	-	-	0,12%	0,11%	
	delivery_method	4	888.426	871.024	17.354	48	HASH	String	-	-	-	0,03%	0,02%	
	loyalty_indicator	2	791.592	790.216	1.344	32	HASH	Boolean	-	-	-	0,02%	0,02%	
	quantity	429	300.192	286.376	10.376	3.440	HASH	Int64	-	-	-	0,01%	0,01%	
	sales_channel	3	276.956	259.824	17.100	32	HASH	String	-	-	-	0,01%	0,01%	
	country_id	8	179.128	161.720	17.328	80	HASH	String	-	-	-	0,01%	0,00%	
	return_code	20	85.060	67.384	17.500	176	HASH	String	-	-	-	0,00%	0,00%	
	transaction_indicator	3	37.020	19.904	17.084	32	HASH	String	-	-	-	0,00%	0,00%	



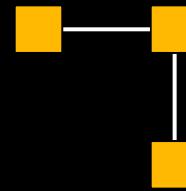
— Performance & Vertipaq Analyzer



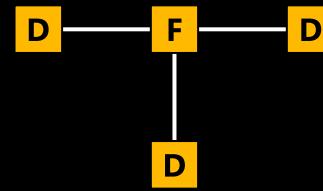
Care about the data model!



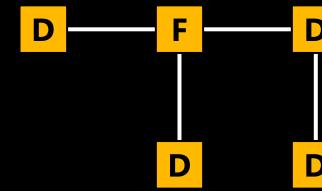
Flat



Normalized
(nth normal form)



Star



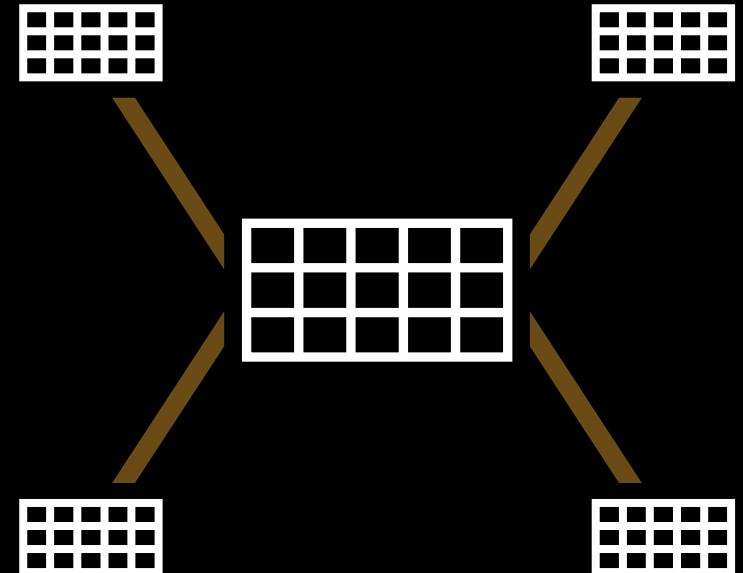
Snowflake



Star schema all the things!

Facts

- Contains **numerical information** about a business process or items to be aggregated
- Aggregations provide totals, averages, etc.
Power BI implements these using **Measures**
- Usefulness limited without context
Context is provided by **dimensions** that slice the data



Dimensions

- Contains **descriptive information** that define how a fact should roll up.
- Examples: Date, Month, Customer, Geography, Product, Payment type.
- Without dimensions there is no context.
- Also called: Lookup table on steroids.



Care about the data model

	Flat table	Normalized	Star schema	Snowflake
Performance for analytics	Low	Medium	High	High
Development effort	Low	High	High	High
Query volume and complexity	Low volume Low complexity	High volume Low complexity	Low volume High complexity	Low volume High complexity
Intended for	No database	CRM / ERP / Applications	Analytical systems / data warehouses	Analytical systems / data warehouses
Compression	Row	Row	Column	Column



Aggregations

Benefits

- Report visualizations are faster
- Balanced architecture by combining DirectQuery and Import storage modes

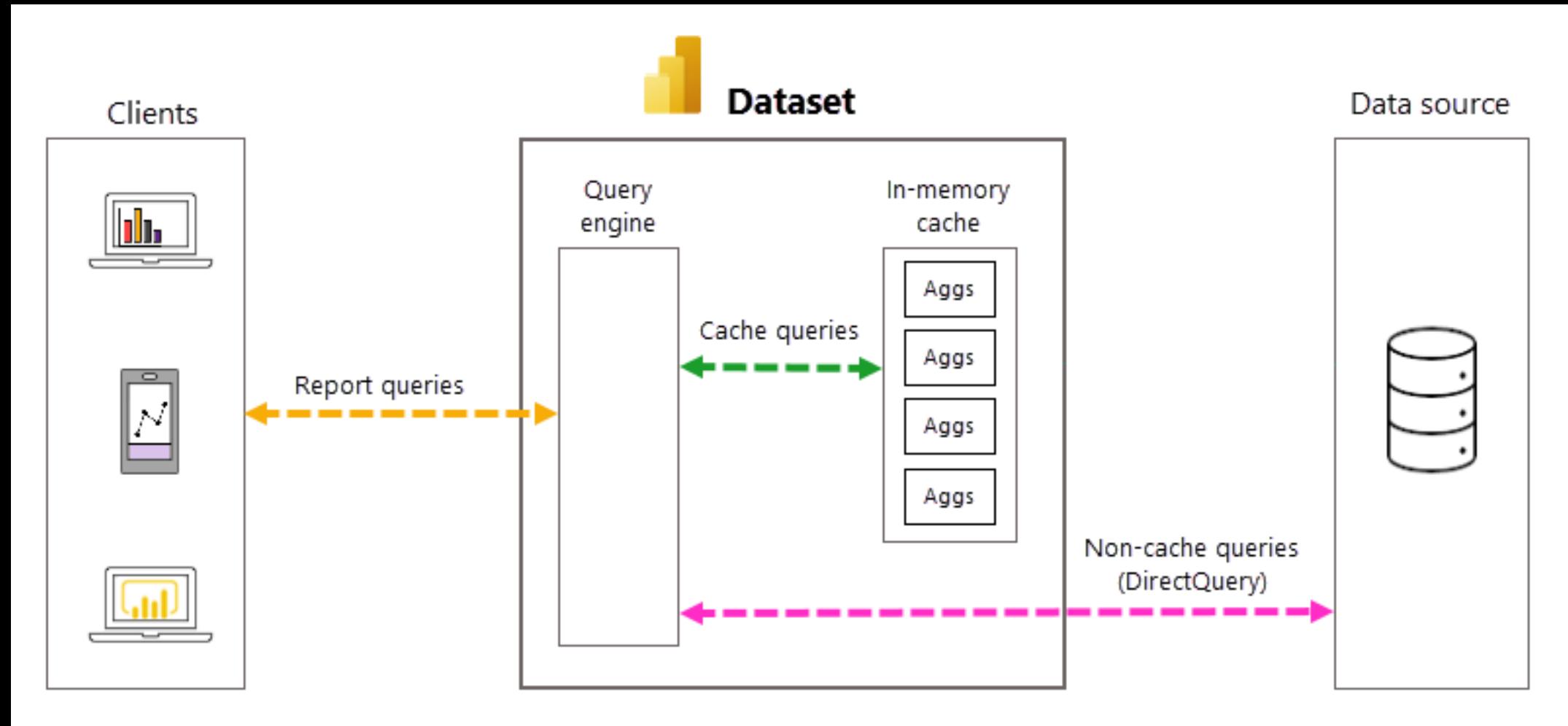
Store data at a higher level of granularity than the original table

The following aggregations are available:
count, groupby, max, min, sum, and count table rows

- Aggregated data is stored in-memory (imported), details are accessed through DirectQuery
- You can create the aggregated table in the Data Transformations (Power Query) or in your source (preferred)



Aggregations





Aggregations

Aggregation, that hit based on relationships, require *regular* relationships.

Regular relationships include the following storage mode combinations, where both tables are from a single source group:

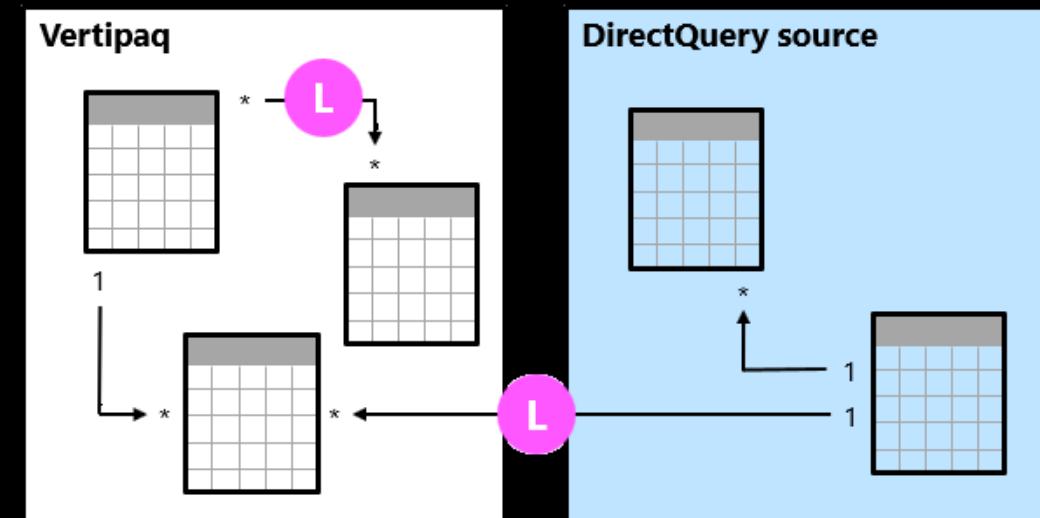
Table on the many side	Table on the 1 side
Dual	Dual
Import	Import or Dual
DirectQuery	DirectQuery or Dual



Relationships and storage modes

A model relationship is **limited** when there's no guaranteed "one" side. It can be the case for three reasons:

- The relationship uses a Many-to-many cardinality type (even if one or both columns contain unique values)
- The storage mode combination is Import and DirectQuery
- The relationship is cross source group





Impact of limited relationships

Cross source group relationships have performance implications.

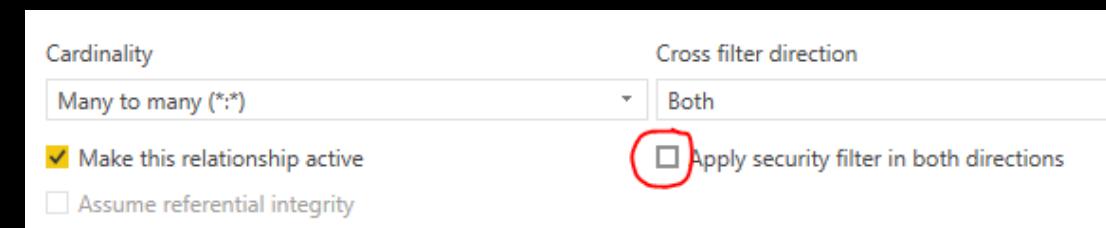
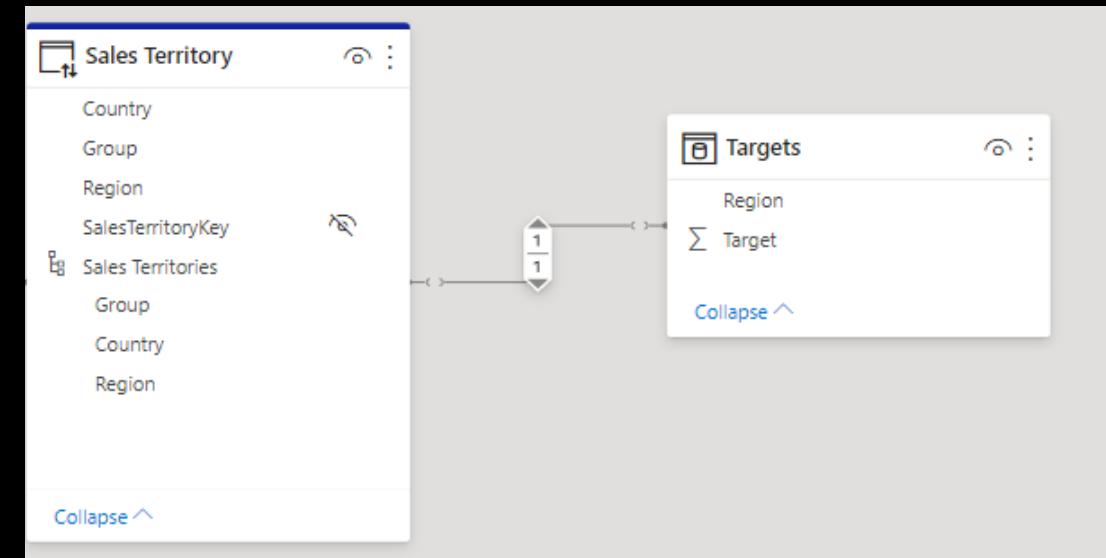
Limited optimization: joins are done on-demand for DirectQuery.

No blank rows: table joins are achieved by using INNER JOIN.

- Blank rows are not added for referential integrity violations

Additional restrictions:

- RELATED DAX function cannot be used to retrieve the 'one' side of the relationship
- Enforcing RLS requires you to check the following checkbox





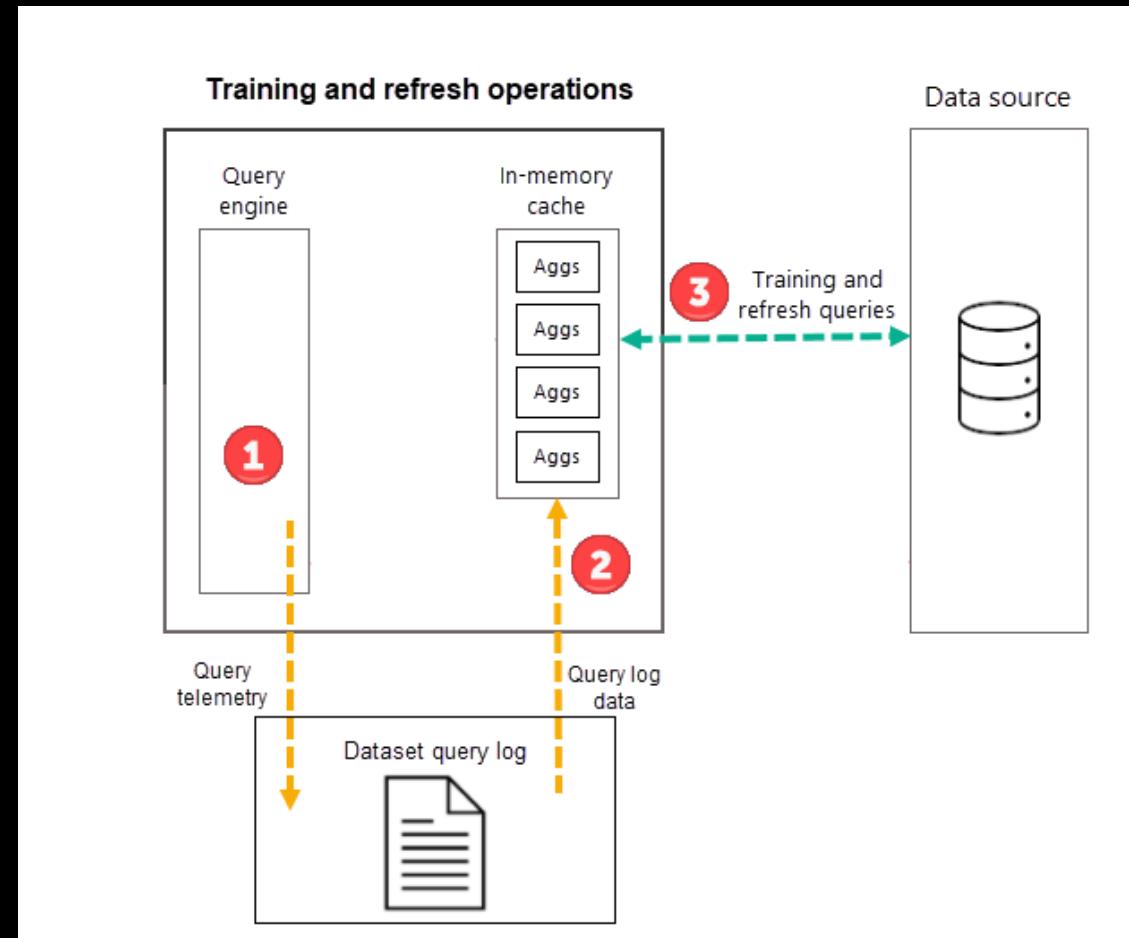
Automatic Aggregations

Power BI Premium per User, Premium Capacity and Embedded datasets

Automatic aggregations based on Query logs (7 days)

Supported sources:

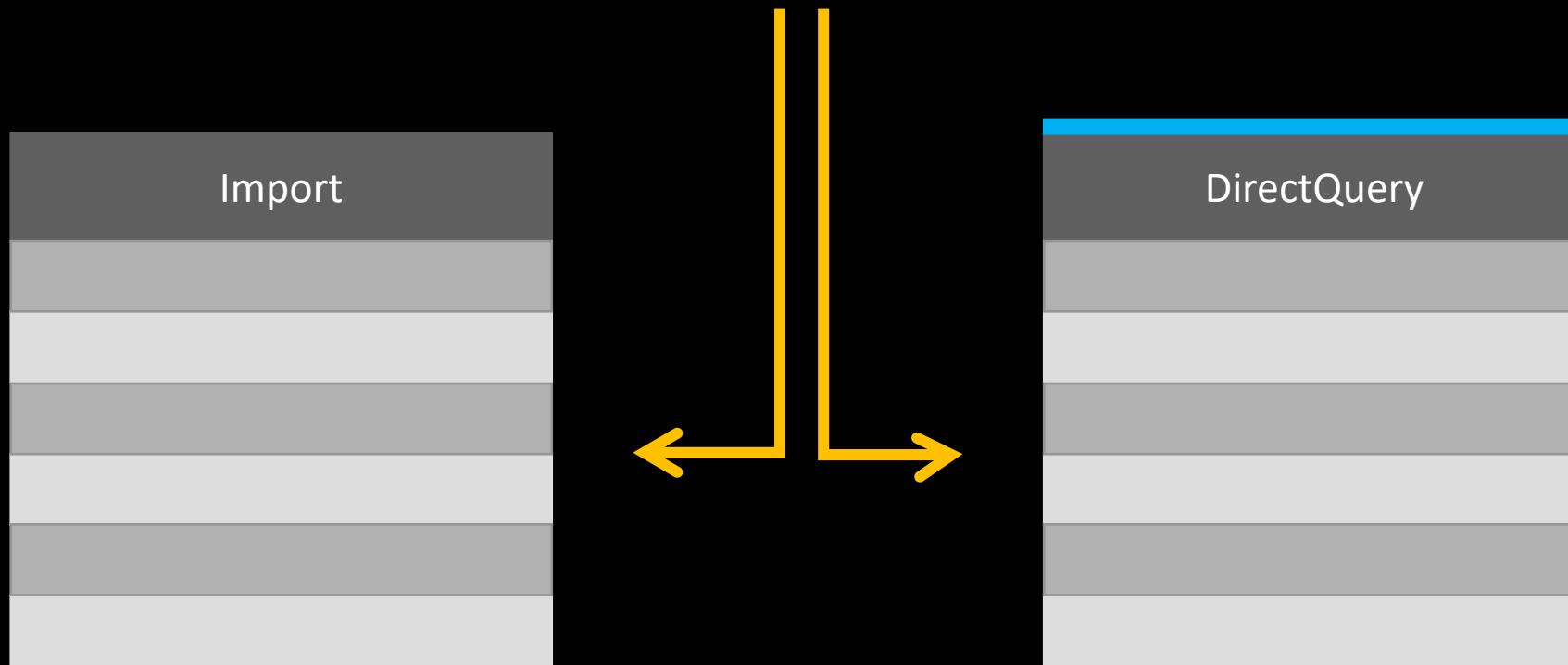
- Azure SQL Database
- Azure Synapse Dedicated SQL pool
- Databricks
- Google BigQuery
- Snowflake
- Amazon Redshift



Aggregated data

Detailed data

{ *Query* }





Demo Aggregations



Query Folding



Generate SQL queries



Push queries back to
the source



Improve performance
in Power Query



Query folding – supported sources

- Relational data sources like SQL Server, Oracle...
- OData sources (SharePoint lists...)
- Active Directory
- Exchange
- ...



Query folding – supported operations

- Filtering (both rows and columns)
- Joins
- Aggregation (Group By)
- Pivot and Unpivot
- Numeric Calculations
- Simple transformations (Uppercase / Lowercase)

« «

SQL requests

 Refresh  Edit columns

 Filter by keyword

Local time : Last 24 hours

Status : All

Pool : Built-in

 Add filter

Showing 1 - 94 of 94 items

Request ID ↑	Request content ↑↓	Submit time ↑↓	Duration	Data processed	Submitter ↑↓
11198235	*** Global stats qu... More	3/1/22, 9:56:40 PM	2 sec	10.00 MB	Marc@Data-marc.com
11198526	SELECT TOP (1000... More	3/1/22, 9:56:40 PM	4 sec	3.29 GB	Marc@Data-marc.com
11196299	SELECT TOP (1000... More	3/1/22, 9:56:09 PM	18 sec	6.64 GB	Marc@Data-marc.com
11033321	SELECT TOP (1000... More	3/1/22, 8:51:57 PM	3 sec	3.29 GB	Marc@Data-marc.com
11031891	*** Global stats qu... More	3/1/22, 8:51:36 PM	2 sec	10.00 MB	Marc@Data-marc.com
11032227	SELECT TOP (1000... More	3/1/22, 8:51:36 PM	5 sec	3.41 GB	Marc@Data-marc.com
11028238	*** Global stats qu... More	3/1/22, 8:51:00 PM	22 sec	1.85 GB	Marc@Data-marc.com
11029080	*** Global stats qu... More	3/1/22, 8:51:00 PM	30 sec	1.62 GB	Marc@Data-marc.com
11030543	*** Global stats qu... More	3/1/22, 8:51:00 PM	30 sec	10.00 MB	Marc@Data-marc.com
11030806	SELECT TOP (1000... More	3/1/22, 8:51:00 PM	36 sec	10.00 MB	Marc@Data-marc.com
11026584	*** Global stats qu... More	3/1/22, 8:50:57 PM	16 sec	351.00 MB	Marc@Data-marc.com
11025192	*** Global stats qu... More	3/1/22, 8:50:47 PM	2 sec	11.00 MB	Marc@Data-marc.com
11025794	SELECT TOP (3502)... More	3/1/22, 8:50:47 PM	3 sec	10.00 MB	Marc@Data-marc.com
11004240	select [].[Customer... More	3/1/22, 8:44:08 PM	1 sec	10.00 MB	Marc@Data-marc.com
11004313	select [].[Employee... More	3/1/22, 8:44:08 PM	1 sec	10.00 MB	Marc@Data-marc.com
11003771	select [rows].[Invoi... More	3/1/22, 8:44:05 PM	15 sec	37.47 GB	Marc@Data-marc.com
10988500	select top 1000 [ro... More	3/1/22, 8:38:51 PM	15 sec	37.41 GB	Marc@Data-marc.com
10987176	select top 1000 [ro... More	3/1/22, 8:38:33 PM	15 sec	37.41 GB	Marc@Data-marc.com
10982220	*** Global stats qu... More	3/1/22, 8:37:42 PM	7 sec	83.00 MB	Marc@Data-marc.com
10982965	*** Global stats qu... More	3/1/22, 8:37:42 PM	15 sec	910.00 MB	Marc@Data-marc.com

Home

Data

Develop

Integrate

Monitor

Manage

Analytics pools

- SQL pools
- Apache Spark pools
- Data Explorer pools (preview)

Activities

- SQL requests
- KQL requests
- Apache Spark applications
- Data flow debug

Integration

- Pipeline runs
- Trigger runs
- Integration runtimes

SQL requests

 Refresh  Edit columns

Filter by keyword

Local time : Last 24 hours

Status : All

Showing 1 - 94 of 94 items

Request ID ↑	Request content ↑	Submit time ↑
11198235	*** Global stats qu... More	3/1/22, 9:56:40 PM
11198526	SELECT TOP (1000... More	3/1/22, 9:56:40 PM
11196299	SELECT TOP (1000... More	3/1/22, 9:56:09 PM
11033321	SELECT TOP (1000... More	3/1/22, 8:51:57 PM
11031891	*** Global stats qu... More	3/1/22, 8:51:36 PM
11032227	SELECT TOP (1000... More	3/1/22, 8:51:36 PM
11028238	*** Global stats qu... More	3/1/22, 8:51:00 PM
11029080	*** Global stats qu... More	3/1/22, 8:51:00 PM
11030543	*** Global stats qu... More	3/1/22, 8:51:00 PM
11030806	SELECT TOP (1000... More	3/1/22, 8:51:00 PM
11026584	*** Global stats qu... More	3/1/22, 8:50:57 PM
11025192	*** Global stats qu... More	3/1/22, 8:50:47 PM
11025794	SELECT TOP (3502)... More	3/1/22, 8:50:47 PM
11004240	select [__].[CustomerKey] as [Customer Key], [__].[CityKey] as [City Key], [__].[CustomerName] as [Customer Name], [__].[Address] as [Address], [__].[PostalCode] as [Postal Code], [__].[Phone] as [Phone], [__].[Fax] as [Fax], [__].[Email] as [Email], [__].[SalesPersonKey] as [Salesperson Key], [__].[SalesTerritoryKey] as [Sales Territory], [__].[SalesTerritoryName] as [Sales Territory Name], [__].[SalesQuota] as [Sales Quota], [__].[CommissionRate] as [Commission Rate], [__].[LastModifiedBy] as [Last Modified By], [__].[LastModifiedDate] as [Last Modified Date], [__].[Rowguid] as [Rowguid], [__].[ModifiedDate] as [Modified Date]	3/1/22, 8:44:08 PM
11004313	select [__].[EmployeeID] as [Employee ID], [__].[EmployeeName] as [Employee Name], [__].[Title] as [Title], [__].[TitleOfCourtesy] as [Title Of Courtesy], [__].[BirthDate] as [Birth Date], [__].[HireDate] as [Hire Date], [__].[Address] as [Address], [__].[PostalCode] as [Postal Code], [__].[Phone] as [Phone], [__].[Fax] as [Fax], [__].[Email] as [Email], [__].[ReportTo] as [Report To], [__].[CommissionRate] as [Commission Rate], [__].[LastModifiedBy] as [Last Modified By], [__].[LastModifiedDate] as [Last Modified Date], [__].[Rowguid] as [Rowguid], [__].[ModifiedDate] as [Modified Date]	3/1/22, 8:44:08 PM
11003771	select [__].[rows].[InvoiceDateKey] as [Invoice Date Key], [__].[rows].[InvoiceDate] as [Invoice Date], [__].[rows].[InvoiceLineCount] as [Invoice Line Count], [__].[rows].[InvoiceTotalAmount] as [Invoice Total Amount], [__].[rows].[InvoiceTotalExcludingTax] as [Invoice Total Excluding Tax], [__].[rows].[InvoiceTotalIncludingTax] as [Invoice Total Including Tax], [__].[rows].[InvoiceType] as [Invoice Type], [__].[rows].[LastModifiedBy] as [Last Modified By], [__].[rows].[LastModifiedDate] as [Last Modified Date], [__].[rows].[Rowguid] as [Rowguid], [__].[rows].[ModifiedDate] as [Modified Date]	3/1/22, 8:44:05 PM
10988500	select top 1000 [row].[InvoiceDateKey] as [Invoice Date Key], [row].[InvoiceDate] as [Invoice Date], [row].[InvoiceLineCount] as [Invoice Line Count], [row].[InvoiceTotalAmount] as [Invoice Total Amount], [row].[InvoiceTotalExcludingTax] as [Invoice Total Excluding Tax], [row].[InvoiceTotalIncludingTax] as [Invoice Total Including Tax], [row].[InvoiceType] as [Invoice Type], [row].[LastModifiedBy] as [Last Modified By], [row].[LastModifiedDate] as [Last Modified Date], [row].[Rowguid] as [Rowguid], [row].[ModifiedDate] as [Modified Date]	3/1/22, 8:38:51 PM
10987176	select top 1000 [row].[InvoiceDateKey] as [Invoice Date Key], [row].[InvoiceDate] as [Invoice Date], [row].[InvoiceLineCount] as [Invoice Line Count], [row].[InvoiceTotalAmount] as [Invoice Total Amount], [row].[InvoiceTotalExcludingTax] as [Invoice Total Excluding Tax], [row].[InvoiceTotalIncludingTax] as [Invoice Total Including Tax], [row].[InvoiceType] as [Invoice Type], [row].[LastModifiedBy] as [Last Modified By], [row].[LastModifiedDate] as [Last Modified Date], [row].[Rowguid] as [Rowguid], [row].[ModifiedDate] as [Modified Date]	3/1/22, 8:38:33 PM
10982220	*** Global stats qu... More	3/1/22, 8:37:42 PM
10982965	*** Global stats qu... More	3/1/22, 8:37:42 PM

Request content

11198526

```
SELECT
TOP (1000001) *
FROM
(
SELECT [t5].[Sales Territory],SUM([t4].[Profit])
AS [a0]
FROM
(
select [__].[SaleKey] as [Sale Key],
[__].[CityKey] as [City Key],
[__].[CustomerKey] as [Customer Key],
[__].[BillToCustomerKey] as [Bill To Customer Key],
[__].[StockItemKey] as [Stock Item Key],
[__].[InvoiceDateKey] as [Invoice Date Key],
[__].[DeliveryDateKey] as [Delivery Date Key],
[__].[SalespersonKey] as [Salesperson Key],
[__].[WWIInvoiceID] as [WWIInvoice ID],
[__].[Description] as [Description],
[__].[Package] as [Package],
[__].[Quantity] as [Quantity],
[__].[UnitPrice] as [Unit Price],
[__].[TaxRate] as [Tax Rate],
[__].[TotalExcludingTax] as [Total Excluding Tax],
[__].[TaxAmount] as [Tax Amount],
[__].[Profit] as [Profit],
[__].[TotalIncludingTax] as [Total Including Tax],
[__].[TotalDryItems] as [Total Dry Items],
[__].[TotalChillerItems] as [Total Chiller Items],
[__].[LineageKey] as [Lineage Key]
from [sales_model].[Sales] as [__]
where [__].[InvoiceDateKey] >= convert(date, '2013-01-01') and [__].[InvoiceDateKey]
) AS [t4]
INNER JOIN

(
select [__].[CityKey] as [City Key],
[__].[WWICityID] as [WWICity ID],
[__].[City] as [City],
```

DAX Studio trace

The screenshot shows the DaxStudio interface version 2.17.3. The top ribbon has tabs for File, Home, Advanced, Help, Layout, and Traces. The Traces tab is selected. The toolbar includes Run, Cancel, Clear Cache, Output, Query Builder, View, Cut, Undo, Copy, Redo, Format Query, To Upper, To Lower, Comment, Uncomment, Merge XML, Find, Replace, Load Perf Data, All Queries, Query Plan, Server Timings, Connect, Refresh Metadata, Power BI, and Traces. The main window has a left sidebar with a tree view of tables: City, Customer, Date, Employee, Sales, Sales Agg, Sales Hybrid, and StockItem. Below the sidebar is a large code editor window displaying XSD schema code. At the bottom is a trace table with columns: StartTime, Type, Duration, User, Database, and Query. The trace table lists numerous entries from different users (Power BI Service, DemoUser@Data-marc.com) performing various operations like XMLA requests, DAX queries, and XML schema refreshes. The status bar at the bottom shows 'Ln 22, Col 19' and other system information.

StartTime	Type	Duration	User	Database	Query
10:11:51	Xmla	9.262	Power BI Service	Test with partitions	<Batch Transaction="true" xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
10:11:42	Xmla	16	Power BI Service	Test with partitions	<Refresh xmlns="http://schemas.microsoft.com/analysisservices/2014/engine">
10:10:55	Xmla	527	DemoUser@Data-marc.com	Test with partitions	<DatabaseID>3771129f-bce3-4f2e-b0e4-91af44829145</DatabaseID>
10:09:22	DAX	2.127	DemoUser@Data-marc.com	Test with partitions	<MaxParallelism>2</MaxParallelism>
10:08:49	DAX	9.957	DemoUser@Data-marc.com	Test with partitions	<Partitions>
10:08:49	DAX	16	DemoUser@Data-marc.com	Test with partitions	<xss:schema xmlns:xss="http://www.w3.org/2001/XMLSchema" xmlns:sql="urn:schemas-microsoft-com:xml-sql">
10:07:47	DAX	2.236	DemoUser@Data-marc.com	Test with partitions	<xss:element>
10:07:10	DAX	14.613	DemoUser@Data-marc.com	Test with partitions	<xss:complexType>
10:05:58	Xmla	11.027	Power BI Service	Test with partitions	<xss:sequence>
10:05:47	Xmla	31	Power BI Service	Test with partitions	<xss:element name="ID" type="xs:unsignedLong" sql:field="ID" minOccurs="0"/>
10:04:16	Xmla	515	DemoUser@Data-marc.com	Test with partitions	<xss:element name="ID.Table" type="xs:string" sql:field="ID_Table" minOccurs="0"/>
09:57:45	Xmla	47	DemoUser@Data-marc.com	Test with partitions	<xss:element name="ID.Partition" type="xs:string" sql:field="ID_Partition" minOccurs="0"/>
09:51:41	DAX	10.685	DemoUser@Data-marc.com	Test with partitions	<xss:element name="RefreshType" type="xs:long" sql:field="RefreshType" minOccurs="0"/>
09:51:34	DAX	4.215	DemoUser@Data-marc.com	Test with partitions	</xss:sequence>
09:50:41	DAX	4.603	DemoUser@Data-marc.com	Test with partitions	</xss:complexType>
09:50:36	Xmla	0	Power BI Service (DemoUser@Data...)	Test with partitions	</xss:schema>
09:50:28	Xmla	8.376	DemoUser@Data-marc.com	Test with partitions	<row xmlns="urn:schemas-microsoft-com:xml-analysis:rowset">
09:50:32	DAX	16	DemoUser@Data-marc.com	Test with partitions	<ID>123</ID>
09:49:58	DAX	2.345	DemoUser@Data-marc.com	Test with partitions	<RefreshTimes>1</RefreshTimes>



Demo Query Folding

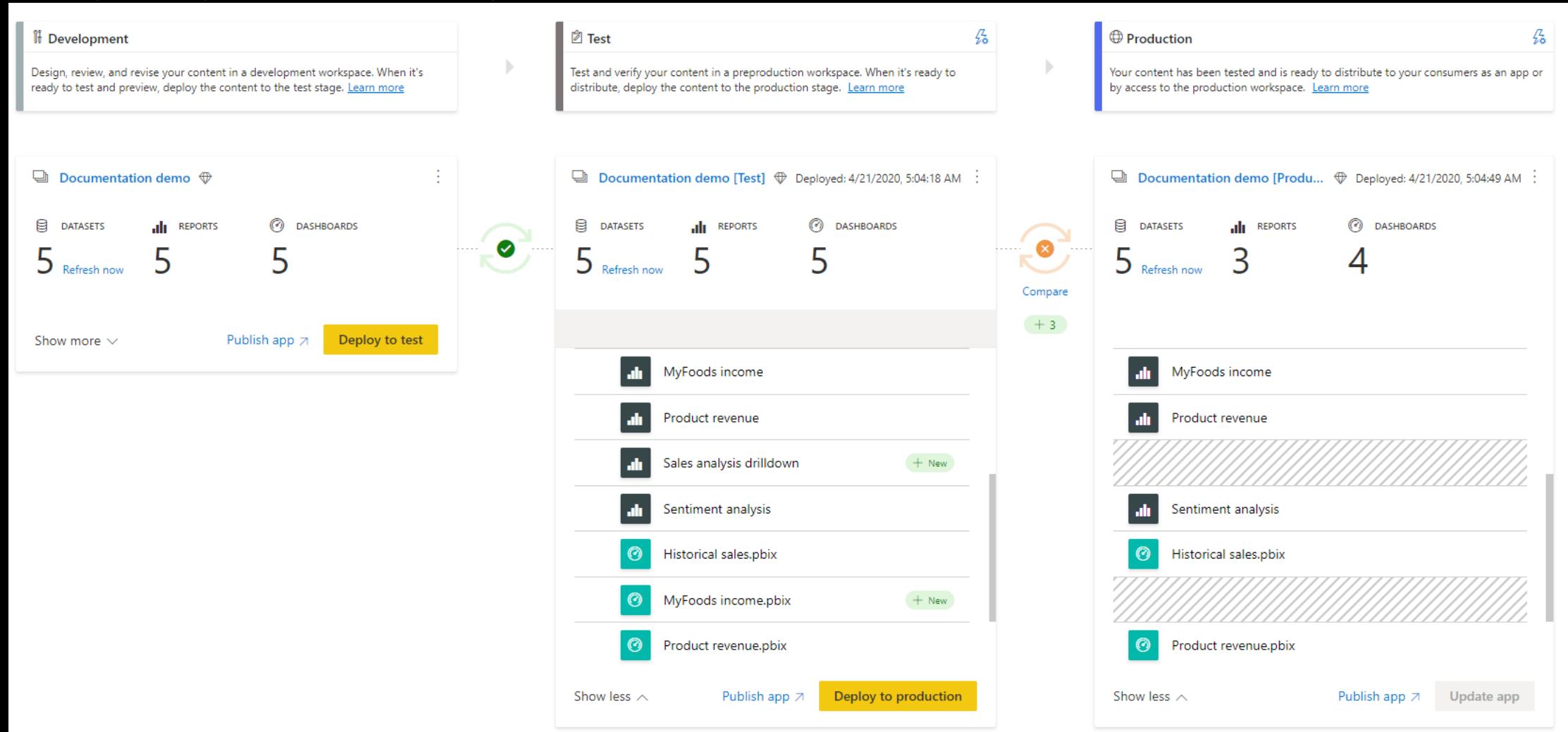


Premium specific features

- Paginated Reports
- Share with free users
(P sku only! Not in A sku!)
- Auto-scale
- Support for larger dataset sizes
- 48x daily refresh
(and Automatic Page Refresh)
- Extra dataflows features
- Enhanced embedding scenarios
- Bring your own key (BYOK)
- Hybrid tables
- Datamarts
- Multi-geo support
- XMLA endpoints
- Deployment Pipelines
- AI workloads

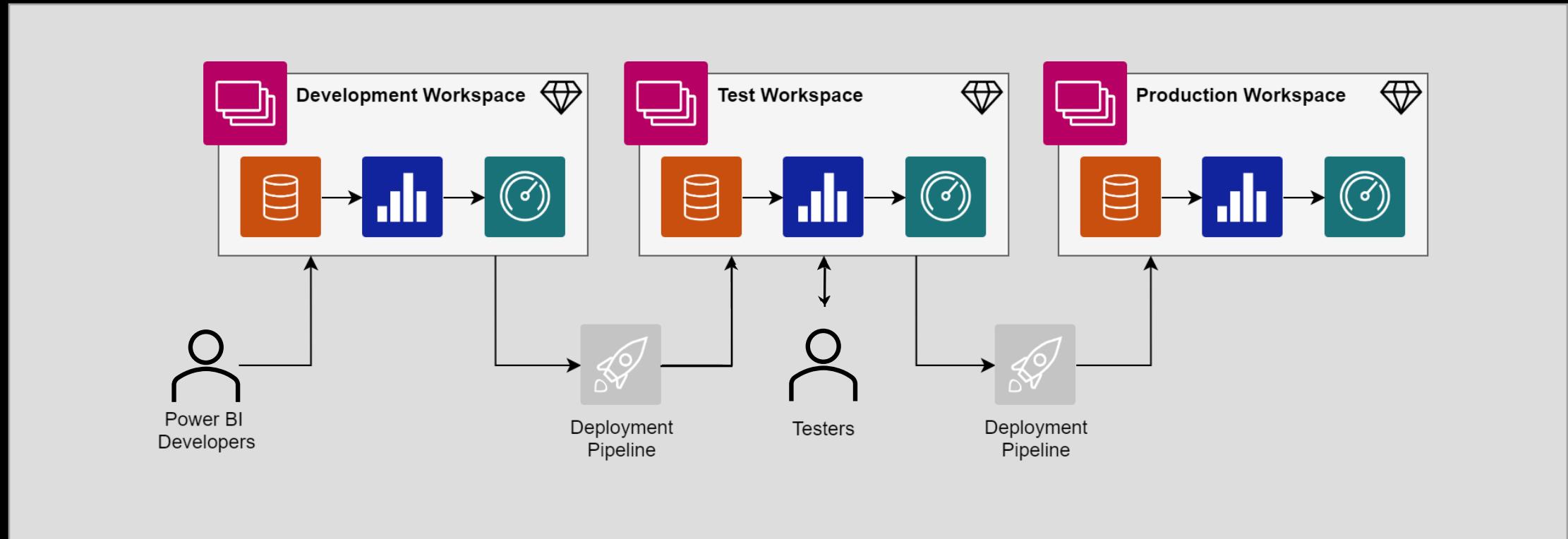


Deployment Pipelines



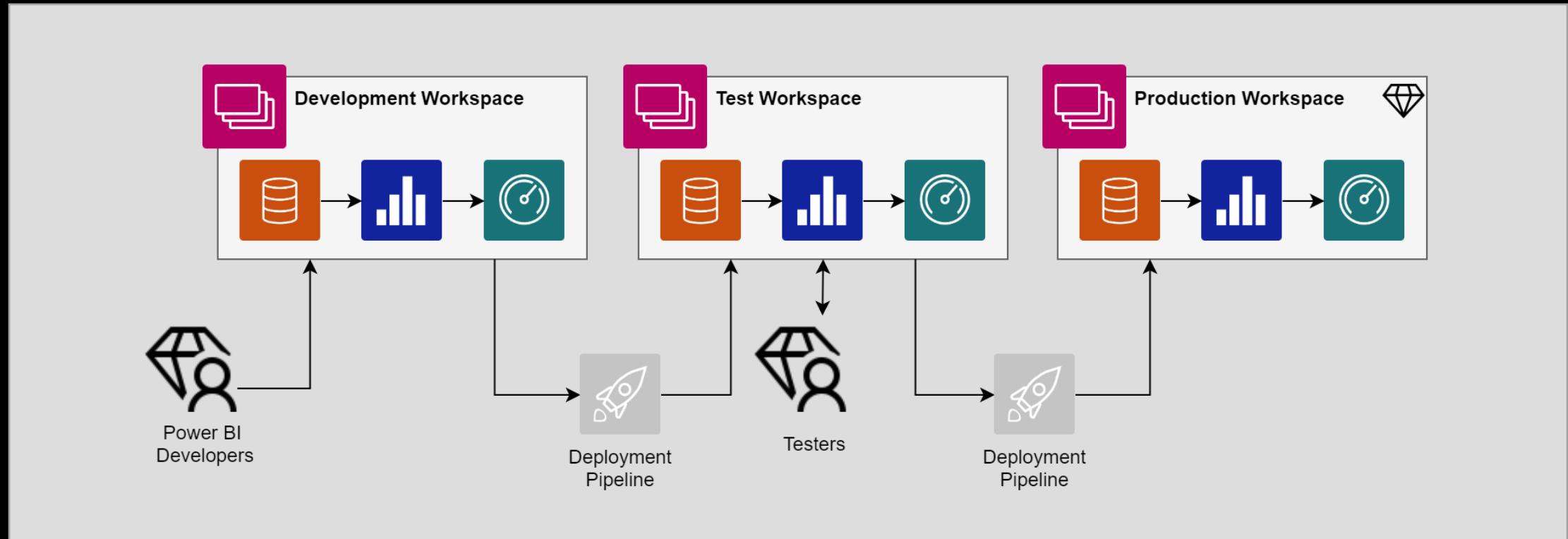


Lower capacity utilization with DTAP





Lower capacity utilization with DTAP



LUNCH BREAK





Hybrid tables & incremental refresh

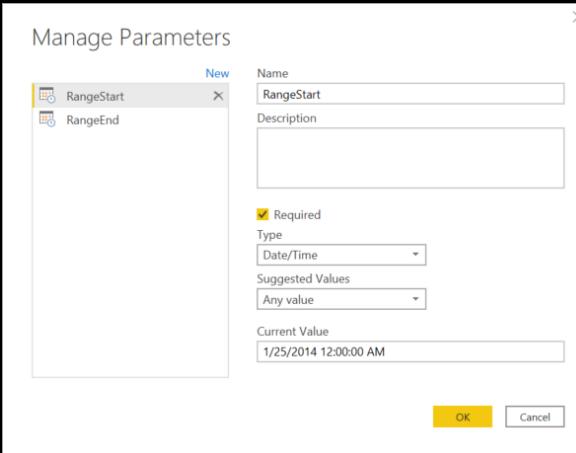
Incremental refresh

- Fewer refresh cycles for fast-changing data – DirectQuery mode gets the latest data updates as queries are processed without requiring a high refresh cadence.
- Refreshes are faster - Only the most recent data that has changed needs to be refreshed.
- Refreshes are more reliable - Long-running connections to volatile data sources aren't necessary. Queries to source data run faster, reducing potential for network problems to interfere.
- Resource consumption is reduced - Less data to refresh reduces overall consumption of memory and other resources in both Power BI and data source systems.
- Enables large datasets - Datasets with potentially billions of rows can grow without the need to fully refresh the entire dataset with each refresh operation.
- Easy setup - Incremental refresh policies are defined in Power BI Desktop with just a few tasks. When published, the service automatically applies those policies with each refresh.

Incremental refresh

- Incremental refresh is supported for Power BI Premium, Premium per user, **Power BI Pro**, and Power BI Embedded datasets.
- Getting the latest data in **real time** with DirectQuery is **only supported for Power BI Premium**, Premium per user, and Power BI Embedded datasets.

Incremental refresh config



CustomerPONumber

OrderDate

DueDate

ShipDate

RangeStart

RangeEnd

Name: RangeStart

Description:

Required

Type: Date/Time

Suggested Values

Any value

Current Value: 1/25/2014 12:00:00 AM

OK Cancel

Sort Ascending

Sort Descending

Clear Sort

Clear Filter

Remove Empty

Date/Time Filters

Search

(Select All)

12/29/2010 12:00:00 AM

12/30/2010 12:00:00 AM

12/31/2010 12:00:00 AM

1/1/2011 12:00:00 AM

1/2/2011 12:00:00 AM

1/3/2011 12:00:00 AM

1/4/2011 12:00:00 AM

1/5/2011 12:00:00 AM

1/6/2011 12:00:00 AM

1/7/2011 12:00:00 AM

1/8/2011 12:00:00 AM

1/9/2011 12:00:00 AM

1/10/2011 12:00:00 AM

1/11/2011 12:00:00 AM

1/12/2011 12:00:00 AM

1/13/2011 12:00:00 AM

1/14/2011 12:00:00 AM

List may be incomplete. Load more

OK Cancel Custom Filter...

Incremental refresh and real-time data

Refresh large tables faster with incremental refresh. Plus, get the latest data in real time with DirectQuery (Premium only). [Learn more](#)

These settings will apply when you publish the dataset to the Power BI service. Once you do that, you won't be able to download it back to Power BI Desktop. [Learn more](#)

- 1. Select table**
FactInternetSales
- 2. Set import and refresh ranges**

Incrementally refresh this table

Archive data starting 5 Years before refresh date
Data imported from 12/21/2016 to 12/18/2021.

Incrementally refresh data starting 3 Days before refresh date
Data will be incrementally refreshed from 12/18/2021 to 12/21/2021.
- 3. Choose optional settings**

Get the latest data in real time with DirectQuery (Premium only) [Learn more](#)

Only refresh complete days [Learn more](#)

Detect data changes [Learn more](#)
- 4. Review and apply**

Archived Incremental Refresh Real time

5 years before refresh date 3 days before refresh date Refresh date

Apply Cancel



Hybrid tables

- Live / Realtime data in Power BI
- Combines different storage modes on partition level in a single table
- Goes hand-in-hand with Incremental Refresh

Granularity	Name	Row Count
Year	2011	295,489,717
Year	2012	297,678,498
Year	2013	295,575,442
Year	2014	292,477,875
Year	2015	297,780,469
Year	2016	294,060,081
Year	2017	300,419,682
Year	2018	296,541,108
Year	2019	292,787,420
Year	2020	299,273,979
Quarter	2021Q1	74,135,277
Month	2021Q104	24,939,498
Day	2021Q10501	820,805
Day	2021Q10502	826,885
Day	2021Q10503	821,043
Day-DirectQuery	2021Q10504-DQ	271,110
Total		3,063,898,887

Archived: **Import**

Incremental refresh: **Import**

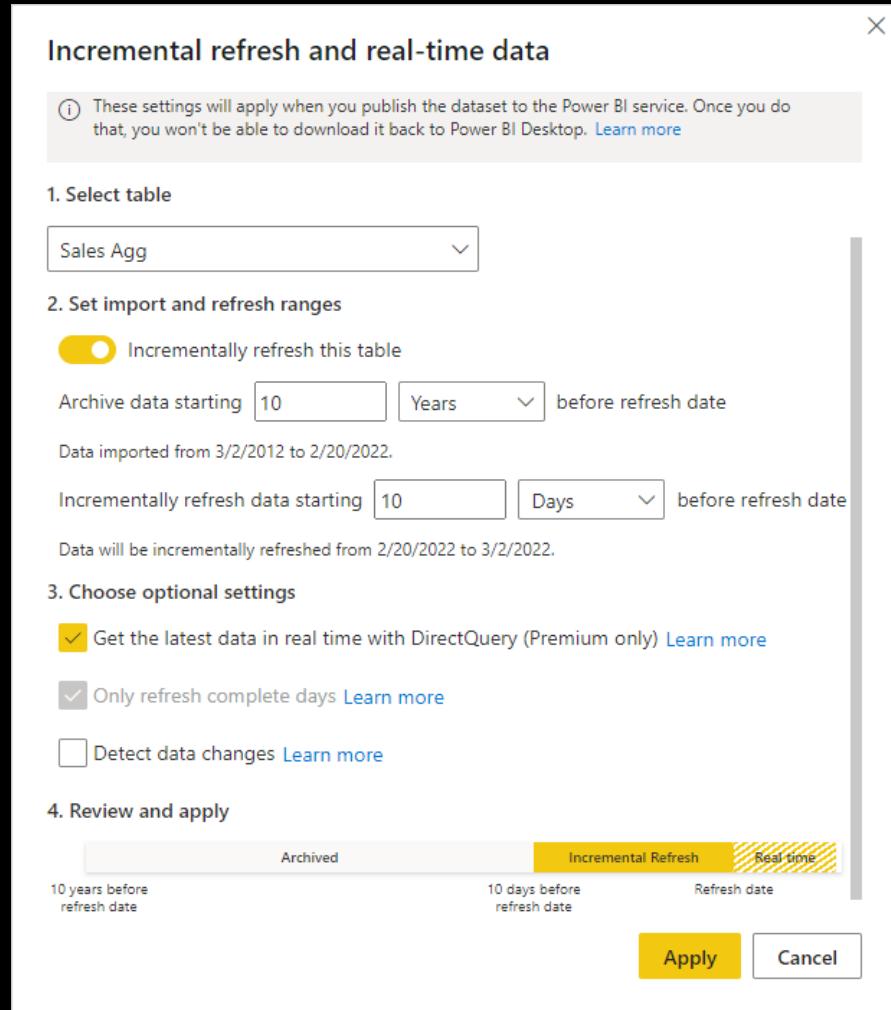
Real time: **DirectQuery**



Hybrid tables

- Implementation with Incremental Refresh
- Customizable via 3rd party tooling like Tabular Editor

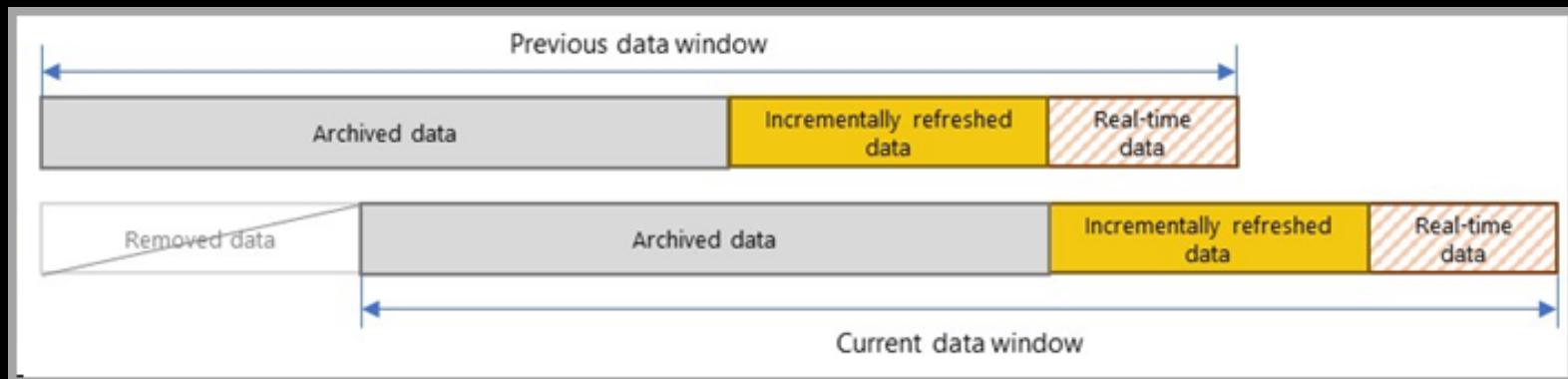
>> Limitation: Only 1 DQ partition per table allowed at the moment.





Hybrid tables – what challenge does it solve?

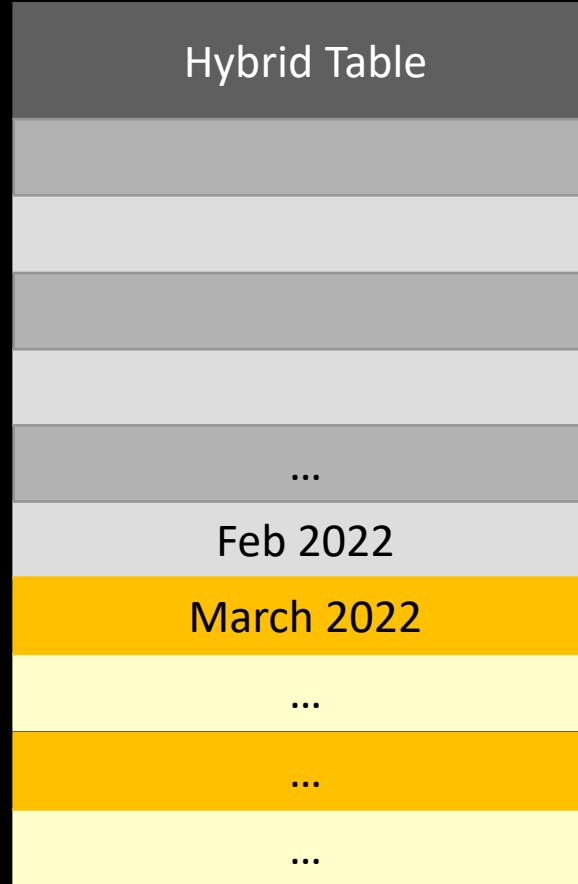
- Realtime scenarios without full tables on DQ mode
- No complex refresh mechanisms needed with partition refresh and queries over XMLA
- No more multiple tables and complex DAX to combine to achieve the same goal





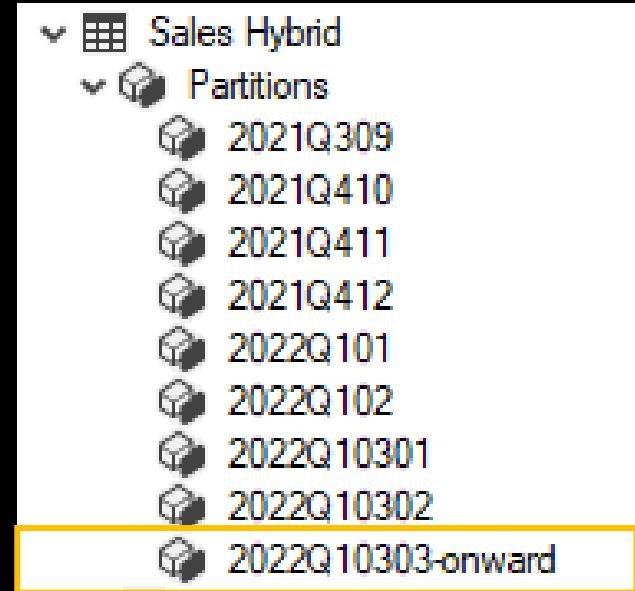
Hybrid tables – Keep in mind that...

- Premium feature
- DAX restrictions for DirectQuery apply
- Limited Power Query capabilities (due to DQ)
- Requires Large Dataset Format (storage) in workspace
- Performance hit on upstream data sources



} Import

} DirectQuery →





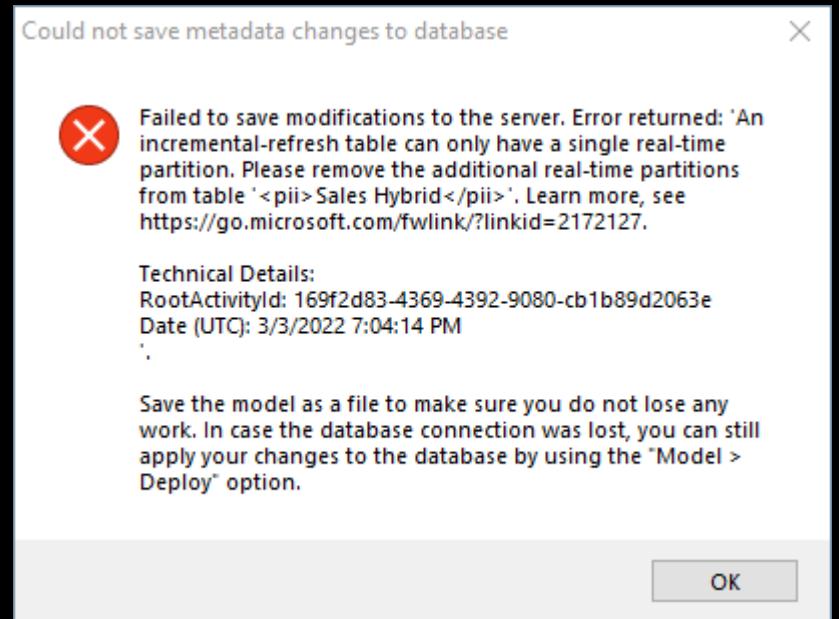
Demo Hybrid Tables
Latest data in real time



Can I change partition storage modes?

You cannot adjust tables with incremental refresh policies applied. However, there are other options to consider.

Options	Default
Data View	DirectQuery
Mode	PolicyRange
Source Type	





Manually setup partitioning

The screenshot shows the Tabular Editor interface with the following details:

- Left pane (Model View):** Shows the project structure with nodes like Model, Data Sources, Perspectives, Relationships, Roles, Shared Expressions, Tables, and Partitions.
- Middle pane (Expression Editor):** Displays DAX code for a partition:

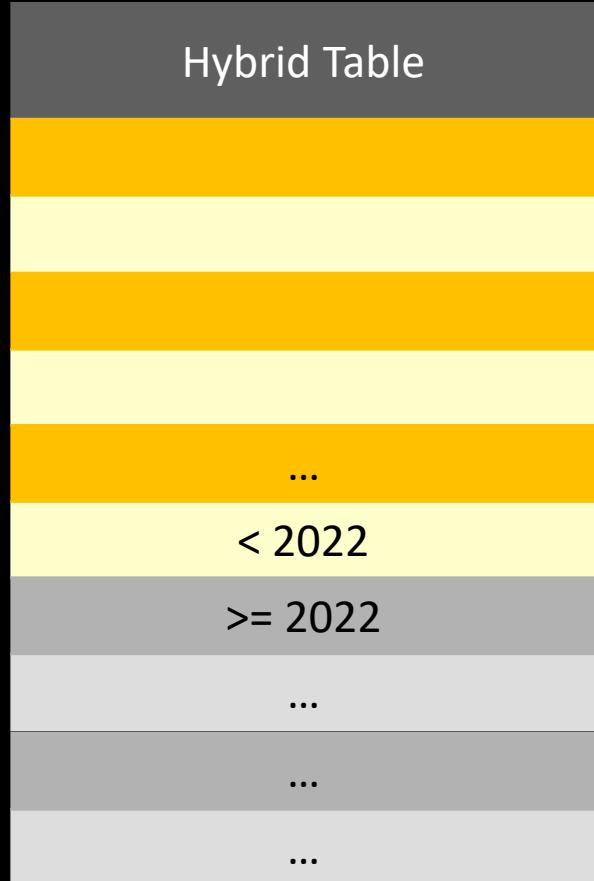
```
let
    Source = Sql.Databases((SynapseEndpoint)),
    #"WideWorldImportersDW-Standard" = Source[[Name=(DatabaseName)]][Data],
    Fact_Sale = #"WideWorldImportersDW-Standard"[[Schema=(SynapseSchema), Item="Sales"]][Data],
    #"Filtered Rows" = Table.SelectRows(#"Fact_Sale", each [InvoiceDateKey] < #date(2022, 01, 01) ),
    SplitByCharacter = Table.TransformColumnNames(#"Filtered Rows", each
        Text.Combine(
            Splitter.SplitTextByCharacterTransition({{"a".."z"}, {"A".."Z"}})(_)
            , " ")
        )
    in
        SplitByCharacter
```

A red box highlights the part of the code where rows are filtered based on the date key.
- Bottom pane (Properties):** Shows the properties for the selected partition "Sales Hybrid-DQ".

Basic	let Source = Sql.Databases((SynapseEndpoint)). #"WideWorldImportersDW
Metadata	Sales Hybrid-DQ
Annotations	0 annotations
Last Processed	31-12-1699
Object Type	Partition (M - DirectQuery)
State	Ready
Options	
Attributes	
Data View	Default
Mode	DirectQuery
Source Type	M

A red box highlights the "Mode" field set to "DirectQuery".
- Bottom status bar:** Shows "1 partition (m - directquery) selected." and "0 BP issues".

Create your own hybrid table setup, using 3rd party tooling like Tabular Editor



} DirectQuery
}

} Import

DirectQuery Partition definition

```
let
    Source = Sql.Databases((SynapseEndpoint)),
    #"WideWorldImportersDW-Standard" = Source{[Name=(DatabaseName)]}[Data],
    Fact_Sale = #"WideWorldImportersDW-
Standard"[{[Schema=(SynapseSchema),Item="Sales"]}] [Data],
    #"Filtered Rows" = Table.SelectRows(#"Fact_Sale", each
        [InvoiceDateKey] < #date(2022, 01, 01) and
        [InvoiceDateKey] >= #date(2013, 01, 01)),
        SplitByCharacter = Table.TransformColumnNames(#"Filtered Rows", each
            Text.Combine(
                Splitter.SplitTextByCharacterTransition({"a".."z"}, {"A".."Z"})(_)
                , " ")
            )
        in
            SplitByCharacter
```

Import Partition definition

```
let
    Source = Sql.Databases((SynapseEndpoint)),
    #"WideWorldImportersDW-Standard" = Source{[Name=(DatabaseName)]}[Data],
    Fact_Sale = #"WideWorldImportersDW-
Standard"[{[Schema=(SynapseSchema),Item="Sales"]}] [Data],
    #"Filtered Rows" = Table.SelectRows(#"Fact_Sale", each
        [InvoiceDateKey] >= #date(2022, 01, 01) and
        [InvoiceDateKey] < #date(2022, 03, 31)),
        SplitByCharacter = Table.TransformColumnNames(#"Filtered Rows", each
            Text.Combine(
                Splitter.SplitTextByCharacterTransition({"a".."z"}, {"A".."Z"})(_)
                , " ")
            )
        in
            SplitByCharacter
```



Demo Hybrid Tables
Historic data via DirectQuery

Refresh challenges







Refresh options

- Scheduled in the service
- Manual trigger
- Power Automate
- PowerShell
- API



Large datasets & high refresh rate

	Pro (shared capacity)	Premium (dedicated capacity)
Total storage size	10 GB	100 TB
Refresh rate	8/day	48/day
Max imported dataset size	1 GB	Depending on SKU size
Isolation with dedicated resources	✗	✓ ¹
Enterprise distribution to all users		
Apps & sharing	✗	✓ ²
Email subscriptions	✗	✓
Embed APIs and controls	✗	✓ ³
Power BI reports on-premises	✗	✓

¹ EM3 SKU or higher.

² Basic user consumption in apps includes viewing content in web and mobile; using Q&A, Quick Insights, and Cortana; and exporting to Excel, CSV, and PowerPoint.

³ Embedded SKUs only allow consumption through embedded APIs and controls, NOT apps or email subscriptions.



Effective refreshing

Considerations

- Refreshing the entire model takes too long with high load on sources
- Can we only refresh certain tables?
- Can we only refresh certain partitions?
- Can we use DQ tables/partitions (Hybrid Tables)

What do we need?

- Enhanced refresh API
- XMLA Endpoints



Enhanced refresh API

- Specify the objects to refresh

POST

https://api.powerbi.com/v1.0/myorg/groups/f089354e-8366-4e18-aea3-4cb4a3a50b48/datasets/cfafbeb1-8037-4d0c-896e-a46fb27ff229/refreshes

```
{  
  "type": "Full",  
  "commitMode": "transactional",  
  "maxParallelism": 2,  
  "retryCount": 2,  
  "objects": [  
    {  
      "table": "DimCustomer",  
      "partition": "DimCustomer"  
    },  
    {  
      "table": "DimDate"  
    }  
  ]  
}
```

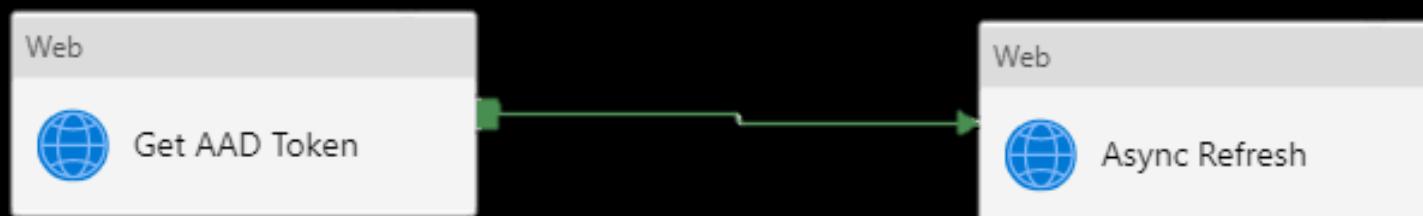


— Demo Enhanced Refresh API



Central E2E orchestration

- Combine pipelines from Data Platform with Power BI
- Lowest latency between source and report
- Consider including backup operations for Power BI
- Incremental loading where possible





— Demo end-2-end orchestration

Scaling





Scaling data platform

- Spark Cluster:
 - Use multiple cluster configs
 - Autoscale -> *It can take 1 to 5 minutes for a scaling operation to complete*
 - Dynamic allocation of executors
 - Automatic pause

Size	vCore	Memory
Small	4	32 GB
Medium	8	64 GB
Large	16	128 GB
XLarge	32	256 GB
XXLarge	64	512 GB
XXX Large (Isolated Compute)	80	504 GB



Scaling data platform

If you are unsure what size cluster to utilize, start with a Medium Spark pool with three nodes and the ability to scale to ten nodes. After executing your spark jobs, view the Apache Spark history server to check performance.

- There are a few areas to monitor:
 - **Disk Spillage** – If there are jobs where data spilled to disk, consider increasing the size of your Spark instance.
 - **Number of Tasks vs. Number of Executors** – Utilizing larger clusters can cause high Garbage Collection (GC). To prevent this, it is best to have less than 5 tasks executing per core when running Spark jobs. If you have exceeded this, consider scaling up your Spark instance.
 - **Executor Usage** – In the Diagnostics tab, the Executor Usage Graph will show you the executor usage over time. If executor usage is very low, consider scaling down your Spark instance.

Best practices for serverless views/pools

- Collocated/region (Azure Storage account vs Synapse)
- Convert large CSV and JSON files to Parquet Delta Tables
- Use appropriate data types
- Try to optimize storage layout by using partitioning and keeping your files in the range between 100 MB and 10 GB
- If you're filtering results by string column, try to use a BIN2_UTF8 collation
- Consider caching the results on the client side by using Power BI import mode
- Push wildcards to lower levels in the path

<https://learn.microsoft.com/en-us/azure/synapse-analytics/sql/best-practices-serverless-sql-pool>



Auto-scale (gen2 only)

Auto-scale adds:

- Additional vCores
- Applies for at least 24h

Configured through

- Max. number of scalable vCores
- Azure subscription – Pay as you go

Power BI Premium > AutoScale test

Premium Generation 2 (preview)
Improve performance and easily track your usage with Premium Generation 2. Enable the preview today. Note: Once enabled, the capacity stops emitting metrics to the metrics app. [Learn more](#)

Enabled

Size | P1
8
Base v-cores

Auto scale | On
0
Additional v-cores in use
Max = 2

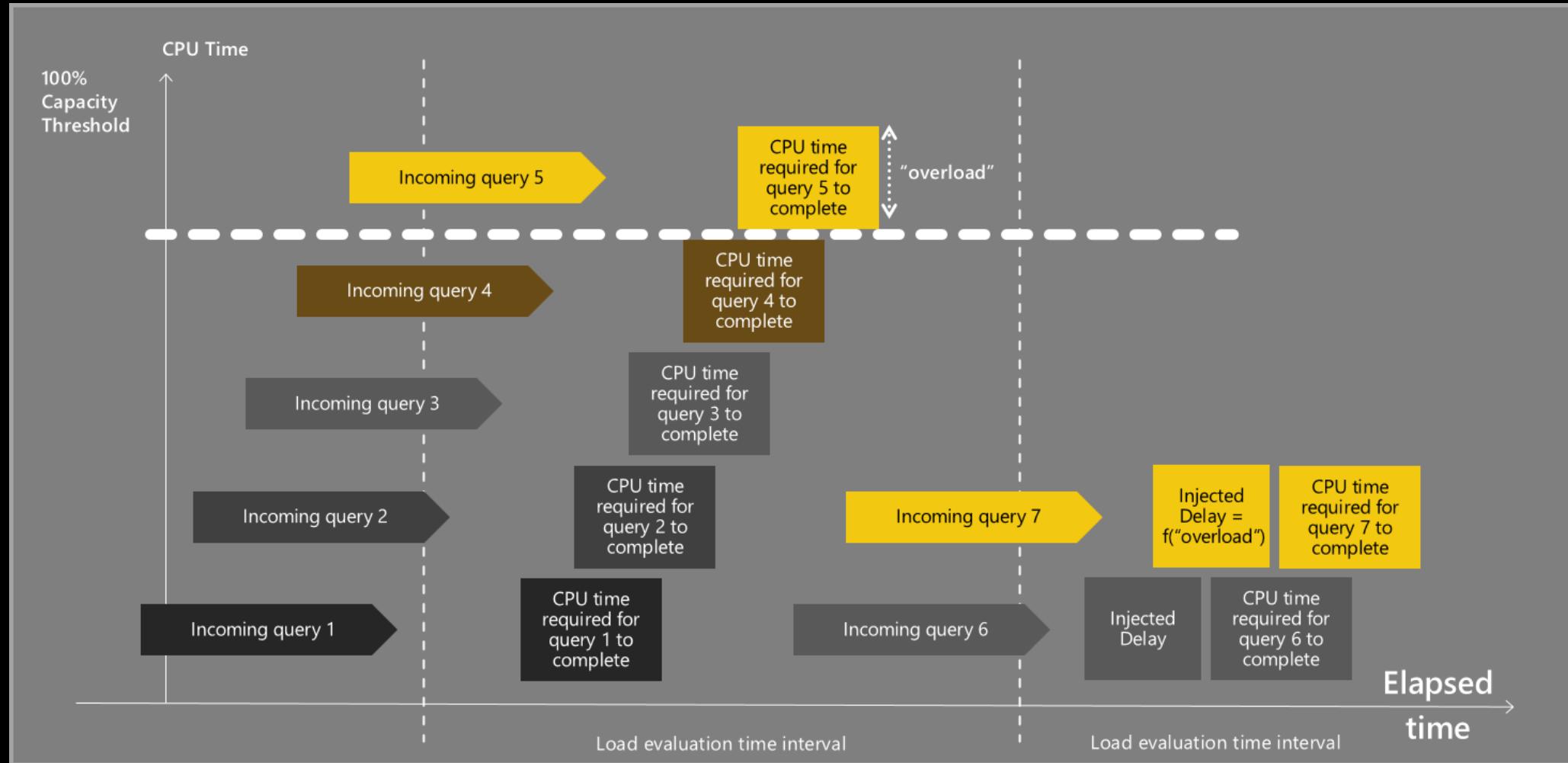
The Premium SKU size you purchased is a P1, which gives you access to 8 v-cores.

[Change size](#) [Manage auto-scale](#)

Auto-scale gives the flexibility to use as much capacity as you need, when you need it. [Learn more](#)

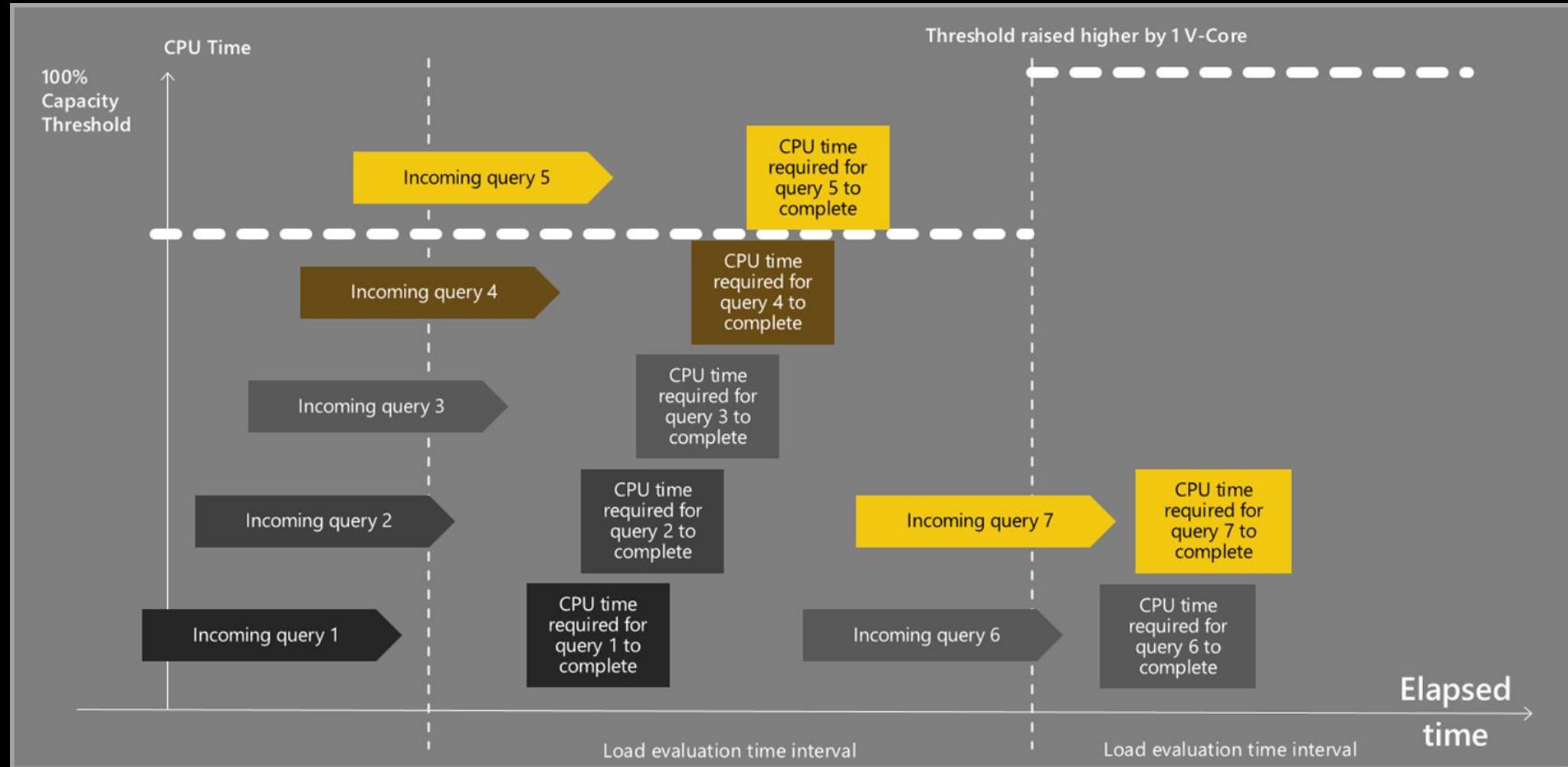


Power BI Premium – Auto scale





Power BI Premium – Auto scale



Monitoring

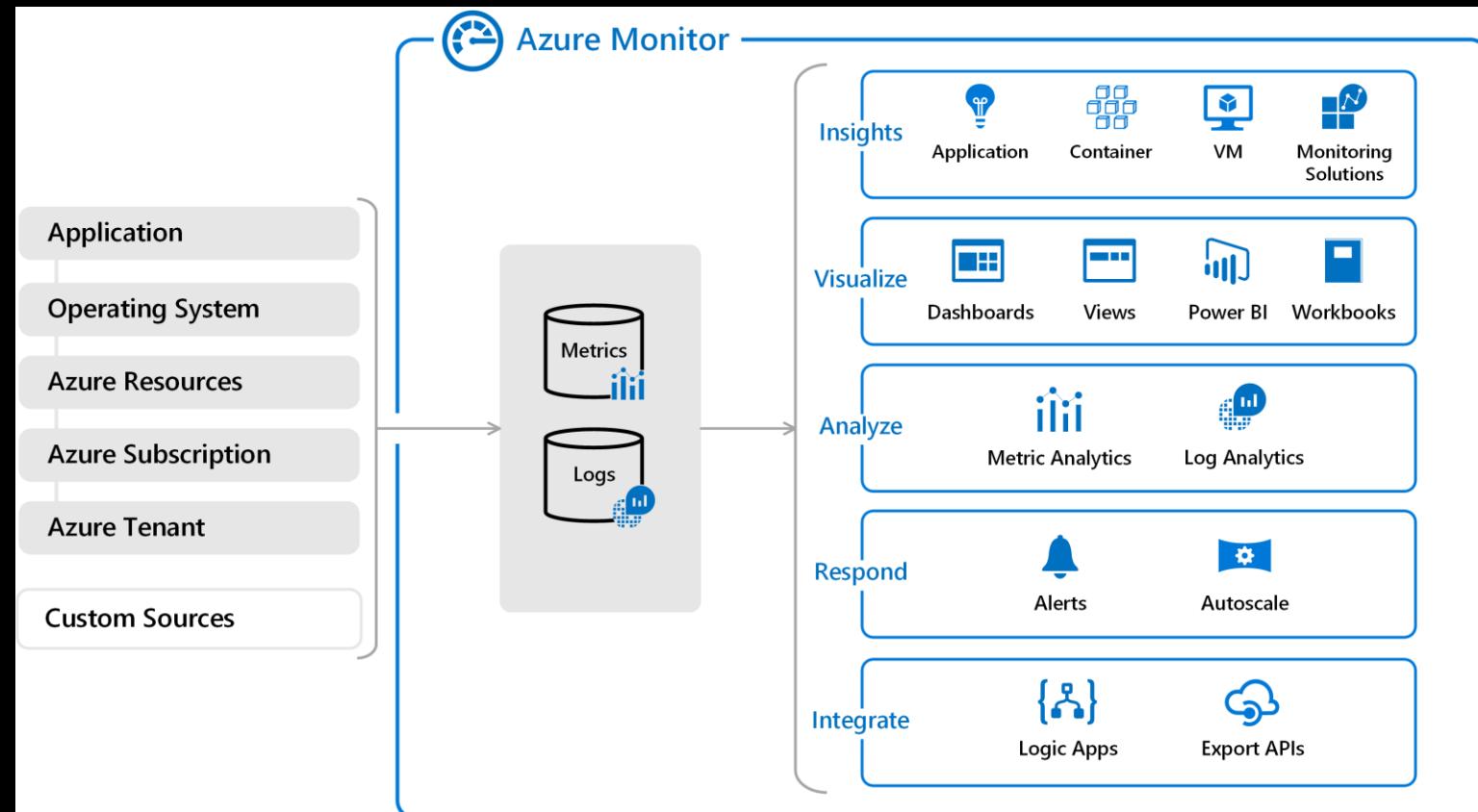


Data platform monitoring



Log Analytics Workspace / Azure Monitor

Azure Monitor delivers a comprehensive solution for collecting, analyzing, and acting on telemetry from your cloud and on-premises environments. It helps you understand how your applications are performing and proactively identifies issues affecting them and the resources they depend on.





Scenarios

- Pipelines:
 - Identify (unusual) long durations
 - Identify unusual amount of runs
 - Trend analysis for durations
- Spark cluster:
 - Analyse utilization of driver and worker nodes
- SQL Serverless:
 - Data scanned by DirectQuery & import to datasets
- Storage:
 - Analyse trend in amount of data stored
 - Analyse trend in amount/duration of reads and writes
- Cost:
 - Analyse Resource Costs



Azure Synapse serverless SQL pool

The screenshot shows the 'Manage' blade of an Azure Synapse Analytics workspace. The left sidebar includes links for Home, Data, Develop, Integrate, Monitor, and Manage, with 'Manage' highlighted by a red box. The main area has tabs for Validate all, Publish all, and Discard all. Under 'Analytics pools', the 'SQL pools' tab is selected and highlighted by a red box. The 'SQL pools' section displays two items: 'Built-in' (Serverless) and 'vvtest' (Dedicated). A 'New' button and a 'Refresh' button are available at the top of the list.

Name	Type
Built-in	Serverless
vvtest	Dedicated

Cost Control

Workspace Budget limit for a period. [Learn more](#)

Daily limit [?](#)

Enable Disable

Data used today

0 MB

10

TB

Weekly limit [?](#)

Enable Disable

Data used this week

5 MB

Enter Weekly limit in TB

TB

Monthly limit [?](#)

Enable Disable

Data used this month

26 GB

200

TB

Azure Synapse serverless SQL pool

The screenshot shows the Azure Synapse Analytics workspace management interface. The left sidebar has several navigation items: Home, Data, Develop, Integrate, Monitor, and Manage, with 'Manage' highlighted by a red box. The main area has tabs for Analytics pools, SQL pools (which is selected and highlighted by a red box), Apache Spark pools, External connections, Linked services, Integration, Triggers, Integration runtimes, Security, Access control, Credentials, and Managed private endpoints. The SQL pools tab displays a summary message: "Serverless SQL pool is immediately available for your workspace. Dec 1". It includes a "New" button, a "Refresh" button, and a toggle for "System assigned managed identity". Below this, it says "Showing 1-2 of 2 items (1 Serverless, 1 Dedicated)". A table lists two items: "Built-in" and "vvtest". The "Built-in" row has a red box around its "Type" column icon (a dollar sign). The "vvtest" row has a red box around its three-dot ellipsis column. At the bottom right are "Apply" and "Cancel" buttons.

Name	Type	... More Options	Se Security	De Delete
Built-in	\$...		
vvtest		...		



Apply

Cancel



Azure Synapse monitoring

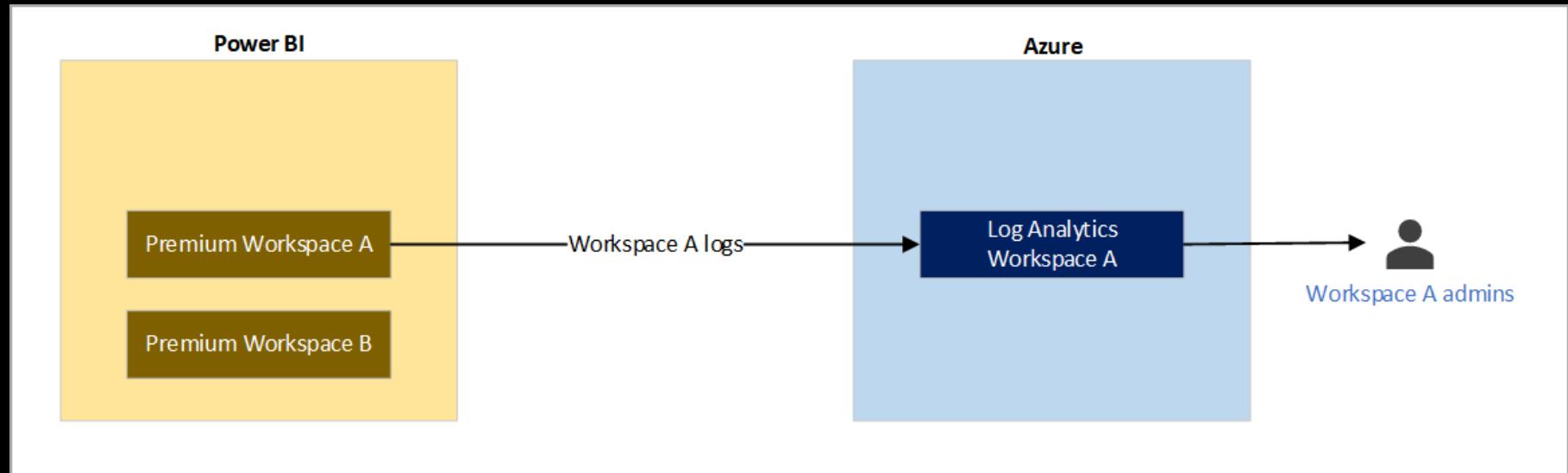
Pipeline runs
Spark cluster
Log Analytics

Power BI monitoring



Power BI Azure Log Analytics (ALA) integration

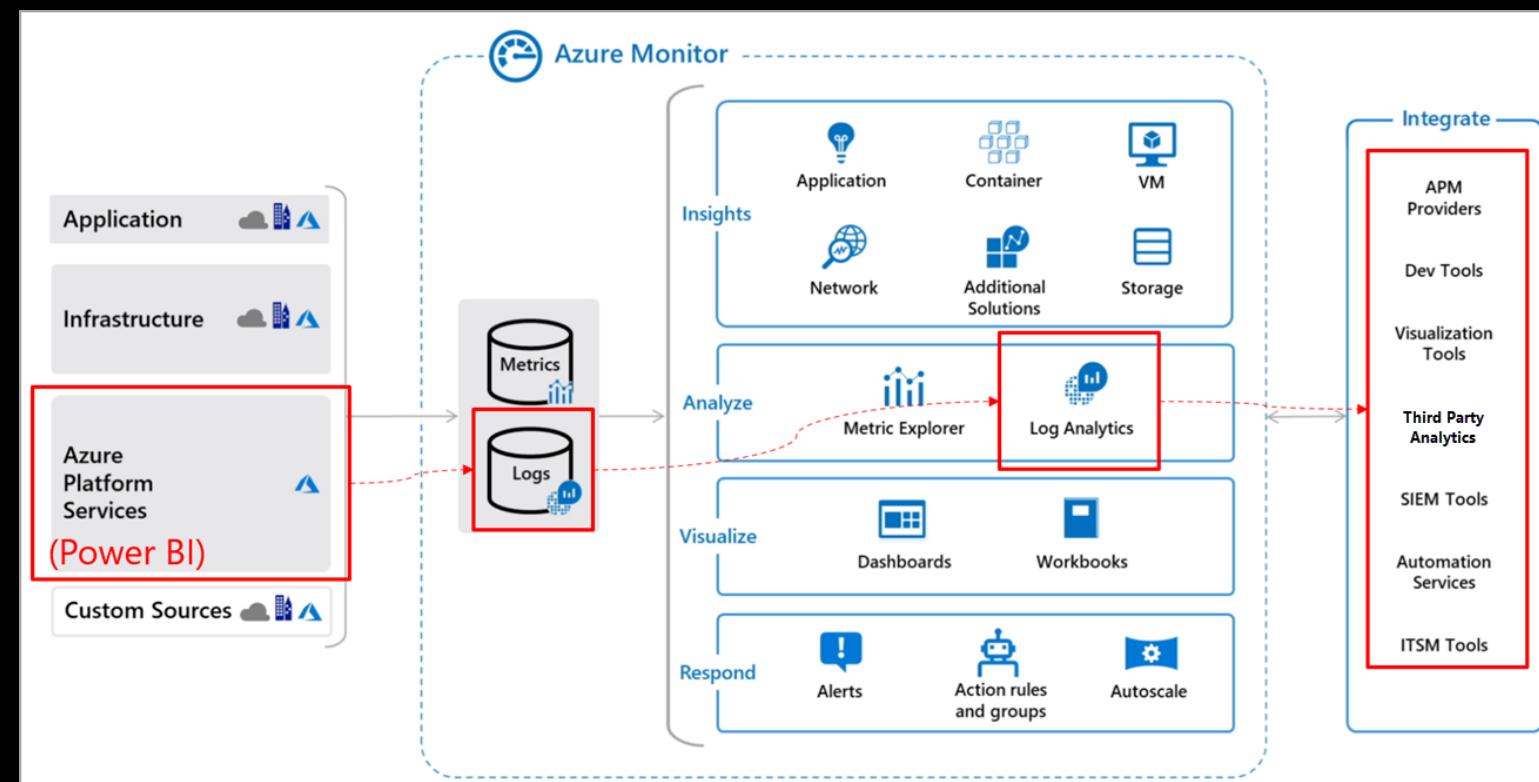
Configure Azure Monitor integration on Power BI workspace level to allow workspace administrators to collect, analyze and act on telemetry data. PBI integration with ALA exposes events from the Analysis Services engine.





Log Analytics Workspace / Azure Monitor

Azure Monitor delivers a comprehensive solution for collecting, analyzing, and acting on telemetry from your cloud and on-premises environments. It helps you understand how your applications are performing and proactively identifies issues affecting them and the resources they depend on.





Scenarios

- Identify high or unusual engine activity by capacity / ws / user
- Analyze query performance
- Analyze dataset refresh durations and engine operations
- Investigate impact of custom operations via Premium XMLA



Example KQLs

```
// log count per day for last 30d  
PowerBIDatasetsWorkspace  
| where TimeGenerated > ago(30d)  
| summarize count() by format_datetime(  
TimeGenerated, 'yyyy-MM-dd')
```

```
// average query duration by day for last 30d  
PowerBIDatasetsWorkspace  
| where TimeGenerated > ago(30d)  
| where OperationName == 'QueryEnd'  
| summarize avg(DurationMs) by  
format_datetime(TimeGenerated, 'yyyy-MM-dd')
```

```
//query duration percentiles for a single day in 1 hour bins  
PowerBIDatasetsWorkspace  
| where TimeGenerated >= todatetime('2021-04-28') and  
TimeGenerated <= todatetime('2021-04-29')  
| where OperationName == 'QueryEnd'  
| summarize percentiles(DurationMs, 0.5, 0.9) by  
bin(TimeGenerated, 1h)
```

```
// refresh durations by workspace and dataset for last 7d  
PowerBIDatasetsWorkspace  
| where TimeGenerated > ago(30d)  
| where OperationName == 'CommandEnd'  
| where ExecutingUser contains 'system'  
| where EventText contains 'refresh'  
| project WorkspaceName, DatasetName = ArtifactName,  
DurationMs
```



Ad-blocker?

✉️ 📈 🔔 ⚡ 🌐 🌐 ? 🔍 inbox@daveruijter.nl MARC LELIUVELD

Notifications ×

More events in the activity log → Dismiss all ▼

! Failed to retrieve schema. Please try to refresh the page. ×

We have experienced a connection issue while retrieving data. This is usually an indication that the network is down or a firewall or browser extension (such as an ad blocker) is mistakenly preventing access.

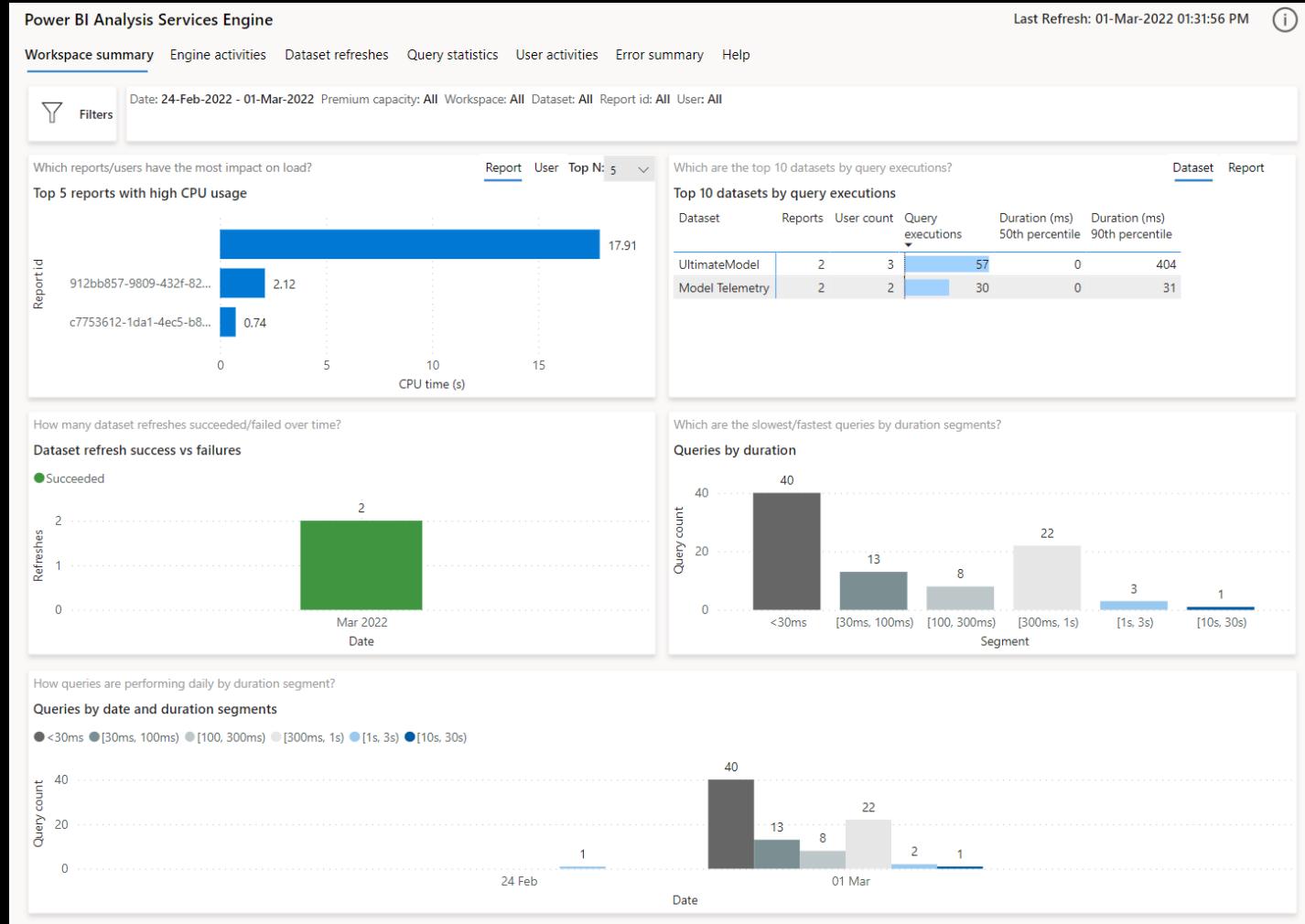
Connection Error

If issue persists, please open a support ticket. Request id: 256ea54b-65a6-4257-b248-a000e699c24e

a few seconds ago



Template report to get insights



- Workspace summary
- Engine activities
- Dataset refreshes
- Query statistics
- Error summary



Power BI Analysis Services Engine

Power BI Analysis Services Engine

Last Refresh: 01-Oct-2021 01:30:46 PM

Workspace summary Engine activities Dataset refreshes Query statistics **User activities** Error summary Help

Date: 26-Sep-2021 - 01-Oct-2021 Premium capacity: All Wokspace: All Dataset: All Report id: All User: All

Operations Queries Top N: 5

Top 5 users by operation

User	Operations
3f7ecc014cc48b5cf2...	222
ea1a7a2ca8ae278b6...	198
335989582a9c58801...	110
27f4484675f0aaaf256...	107
d661cf332ee8e0cbd...	107

Queries vs CPU time by users

46 Users

2,454 Operations

Daily user and operation count

Date	Users	Operations
Sep 26	0.10K	0.00K
Sep 27	0.46K	0.01K
Sep 28	0.71K	0.02K
Sep 29	0.01K	0.47K
Sep 30	0.68K	0.01K
Oct 01	0.00K	0.02K

Hourly user and operation count

Hour	Users	Operations
0	0.00K	0.00K
1	0.02K	0.00K
2	0.00K	0.00K
3	0.00K	0.00K
4	0.00K	0.00K
5	0.19K	0.00K
6	0.00K	0.00K
7	0.09K	0.00K
8	0.23K	0.00K
9	0.35K	0.00K
10	0.00K	0.00K
11	0.00K	0.00K
12	0.37K	0.00K
13	0.00K	0.00K
14	0.00K	0.00K
15	0.00K	0.00K
16	0.00K	0.00K
17	0.00K	0.00K
18	0.00K	0.00K
19	0.00K	0.00K
20	0.00K	0.00K
21	0.00K	0.00K
22	0.00K	0.00K
23	0.00K	0.00K
24	0.00K	0.00K

User details

User	Last active date	Dataset	Report	Operations	Refreshes	Queries	Avg CPU time	Avg duration (ms)
3f7ecc014cc48b5cf2...	01-Oct-2021	2	2	222		115	219	228
ea1a7a2ca8ae278b6...	01-Oct-2021	2	2	198		43	133	132
335989582a9c58801...	01-Oct-2021	2	2	110		58	72	104
27f4484675f0aaaf256...	01-Oct-2021	2	3	107		32	316	340
d661cf332ee8e0cbd...	01-Oct-2021	2	2	107		49	75	70
f2a2a1f84d3a9010ba8302...	01-Oct-2021	2	2	106		77	156	166
f60aef70097735754711...	01-Oct-2021	2	2	90		64	186	181
Total		4	6	2,454		959	196	178



GitHub location for .pbix

<https://github.com/microsoft/PowerBI-LogAnalytics-Template-Reports>



Customizations

- Visual ID
- Report Name



Demo Log Analytics Integration

SQL Database	S3 DTUs	\$88.91
Azure Synapse Analytics	vCore	\$63.06
Azure Synapse Analytics	Standard Data Processed	\$14.69
Azure Synapse Analytics	Standard LRS Data Stored	\$13.63
Azure Synapse Analytics	Azure Hosted IR Data Movement	\$7.6
Storage	S10 Disks	\$3.62
SQL Database	B DTU	\$2.96
Storage	Hot LRS Data Stored	\$2.51
Storage	Hot Read Operations	\$1.03
Storage	Hot Iterative Read Operations	\$0.62
Storage	Hot Write Operations	\$0.43
Storage	Hot Other Operations	\$0.39
Storage	Hot LRS Data Stored	\$0.2
Azure Synapse Analytics	Standard ZRS Data Stored	\$0.11
Azure Synapse Analytics	Azure Hosted IR Orchestration Activity Run	\$0.09
Log Analytics	Pay-as-you-go Data Ingestion	\$0.05
Log Analytics	Pay-as-you-go Data Retention	\$0
Storage	Hot LRS Index	\$0
Azure Synapse Analytics	Azure Hosted IR Pipeline Activity	\$0
Azure Synapse Analytics	Azure Hosted IR External Pipeline Activity	\$0
Bandwidth	Standard Data Transfer Out	\$0
Storage	Hot Other Operations	\$0
Storage	All Other Operations	\$0
Storage	Hot Write Operations	\$0
Storage	LRS List and Create Container Operations	\$0
Storage	Hot Read Operations	\$0
Storage	Delete Operations	\$0
Storage	Cool LRS Data Stored	\$0
Bandwidth	Standard Data Transfer In	\$0
Storage	Batch Write Operations	\$0
Storage	LRS Data Stored	\$0

Wrap-up

Design and implement	Orchestrate	Performance	Cost
Better design and implement complex data models, including hybrid tables, aggregations, and combined storage modes (import, DirectQuery , dual).	Orchestrate the end-to-end data processing, with a pipeline chain from data ingest in the data lake house to the incremental Power BI dataset refresh.	Use different techniques to identify performance bottlenecks in your solutions and how to solve those ("does it fold"?).	Implement a cost-efficient solution, that still meets the scalability demands.

Any bonus questions?



Please provide us feedback!