

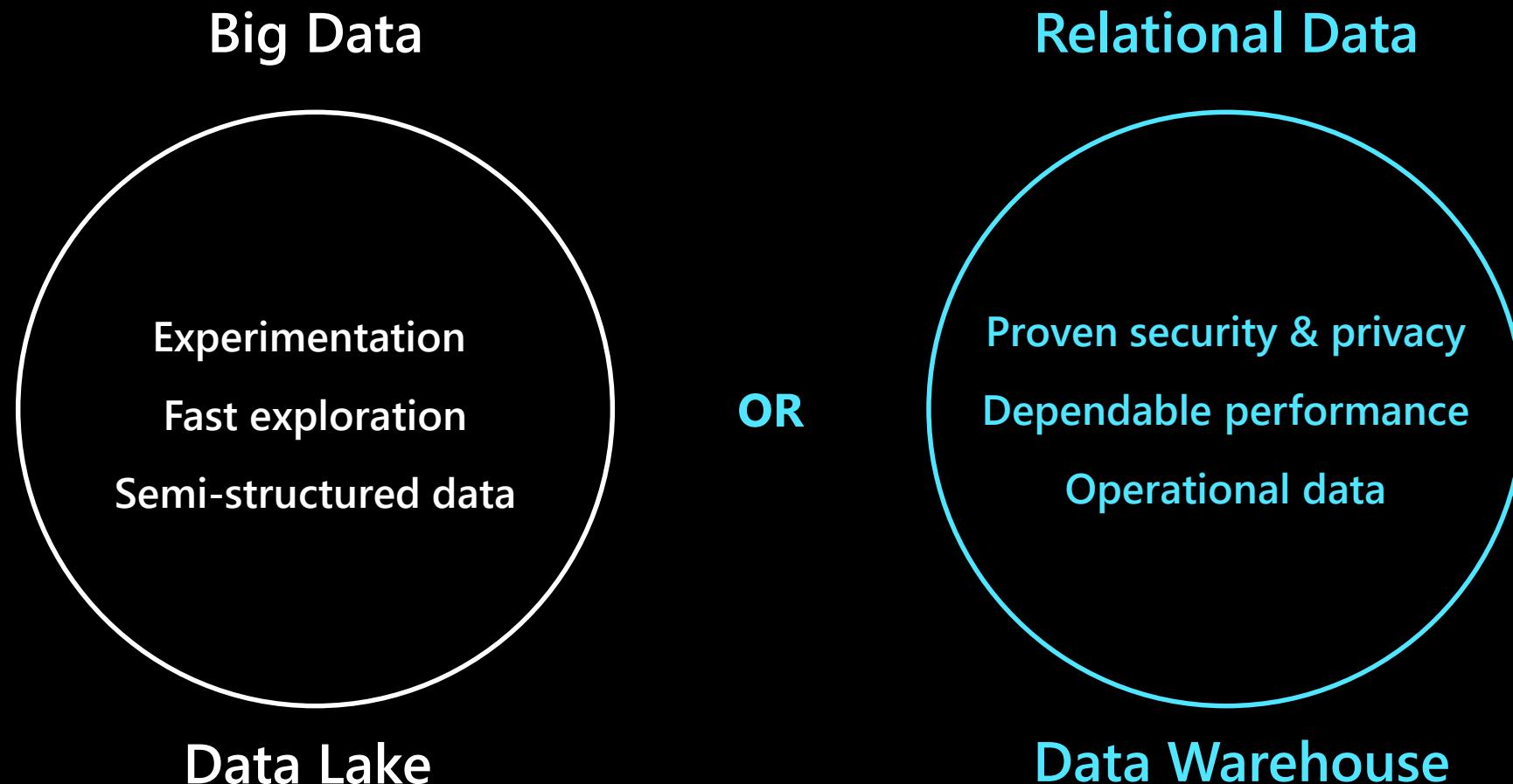


# Azure Synapse Analytics

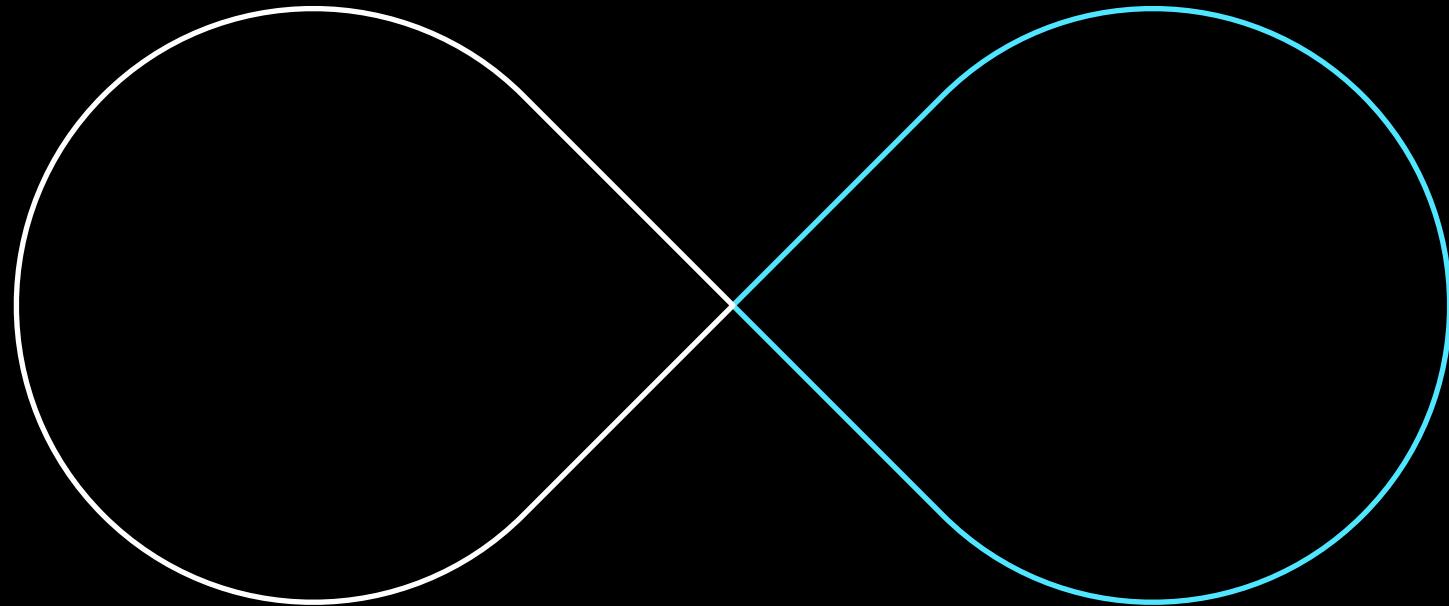
Andrea Benedetti

Sr. Cloud Architect / Data & AI Engineer | Microsoft Italia

This is a result of businesses being forced to maintain two critical, yet independent analytics systems



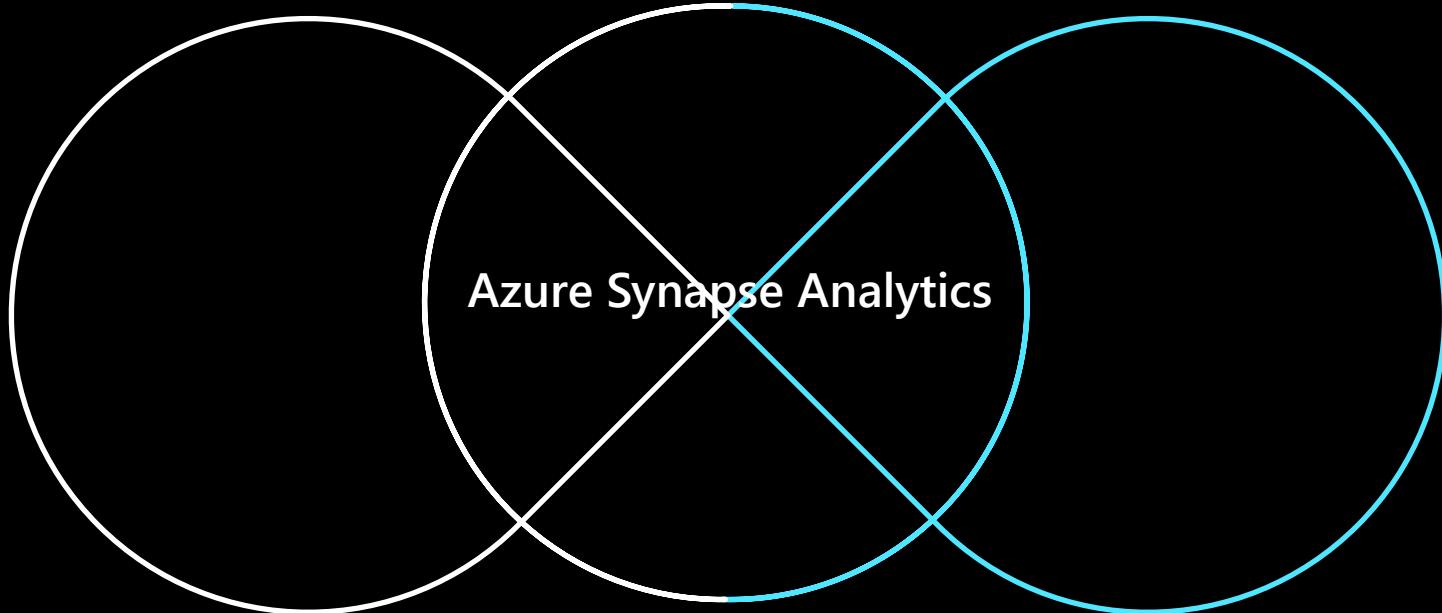
Azure brings these two worlds together, in a single service,  
to provide limitless analytics



Welcome to **limitless**

Data warehousing & big data analytics—all in one service

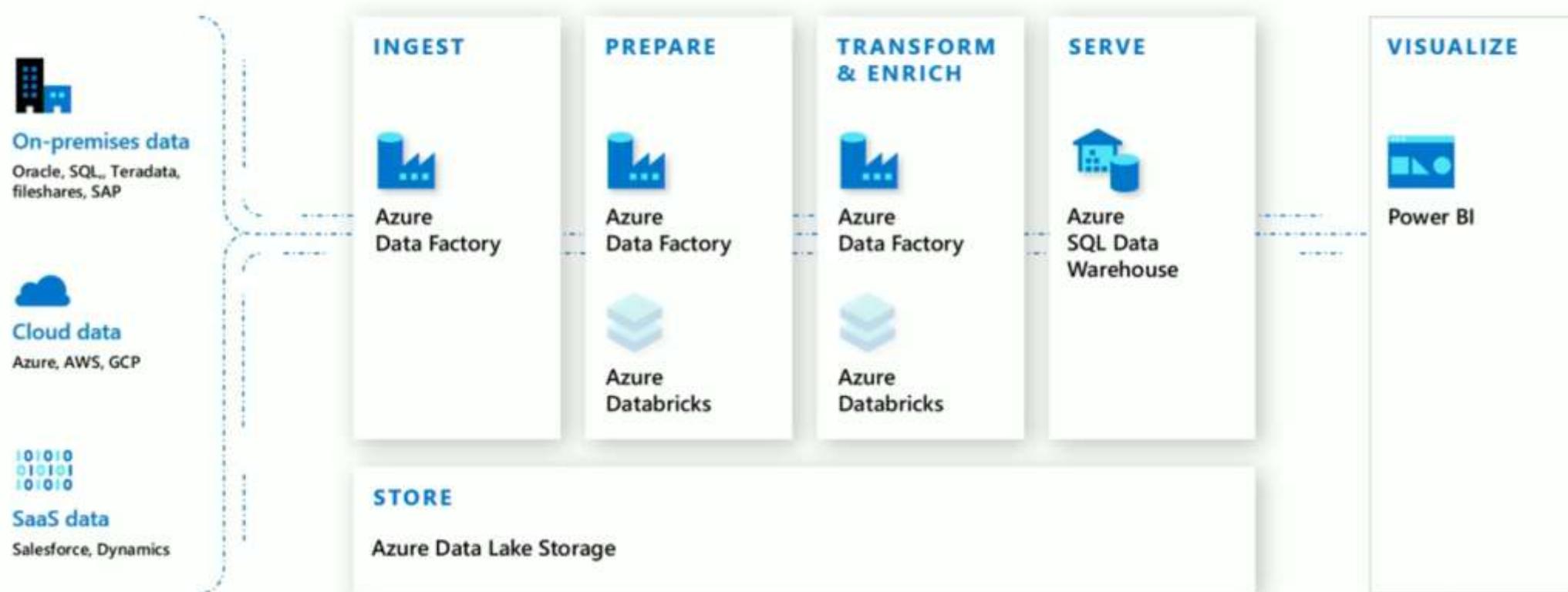
Azure brings these two worlds together, in a single service,  
to provide limitless analytics



Welcome to **limitless**

Data warehousing & big data analytics—all in one service

# Modern Data Warehouse

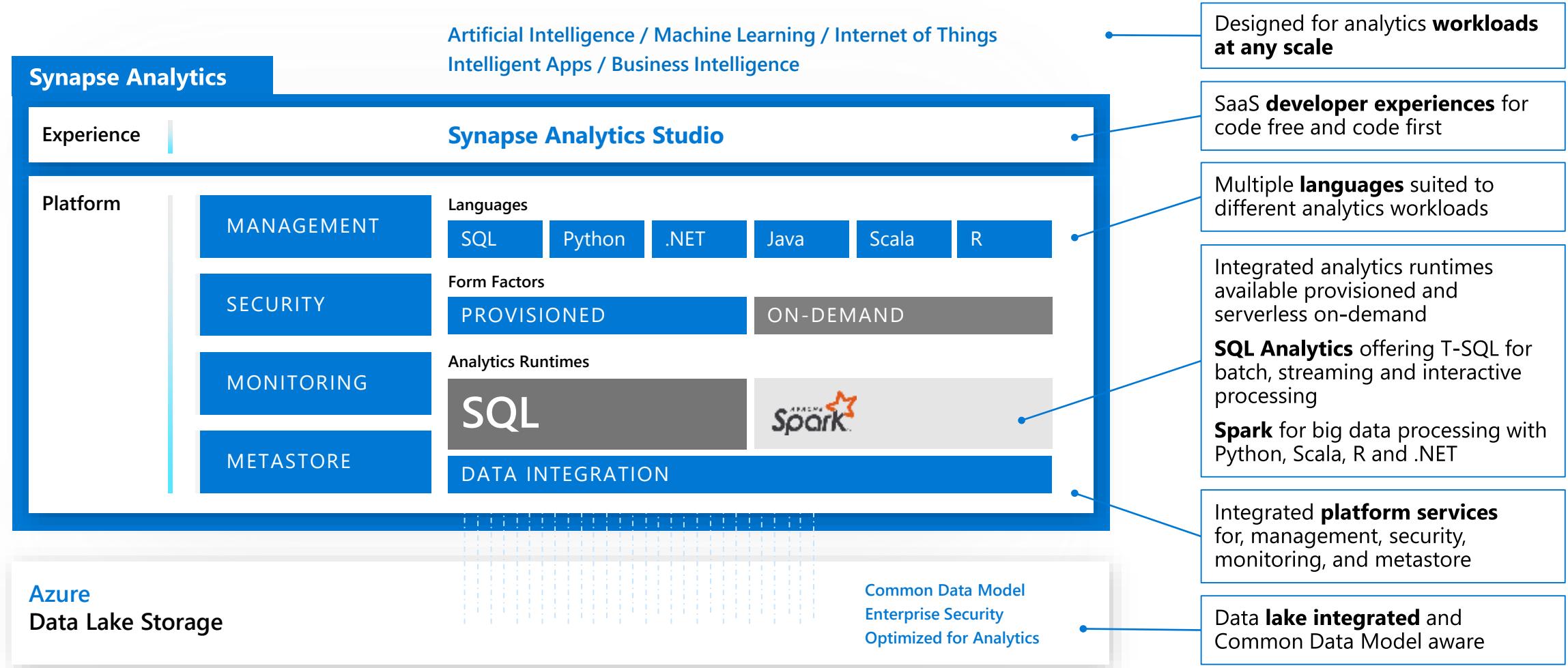


# Azure Synapse Analytics



# Azure Synapse Analytics

Limitless analytics service with unmatched time to insight

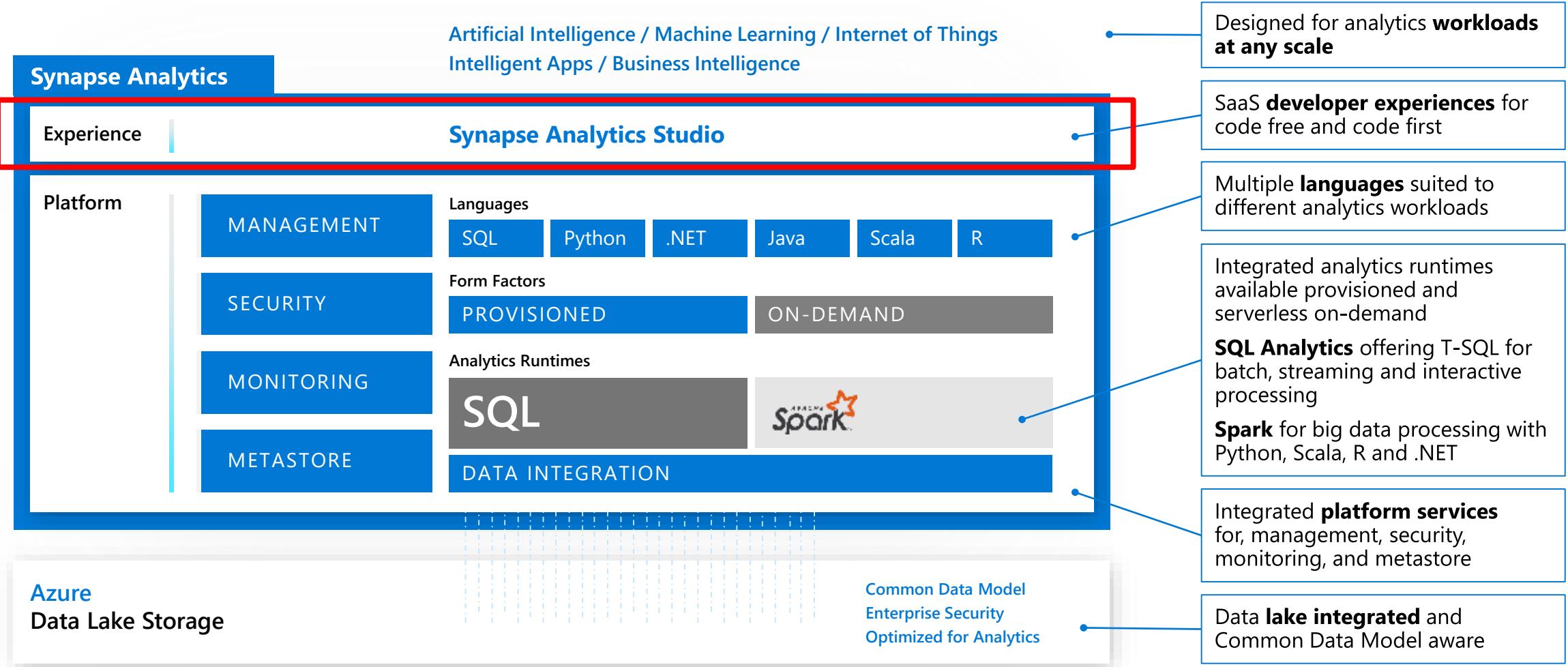




# Azure Synapse Analytics Studio

# Azure Synapse Analytics

Limitless analytics service with unmatched time to insight



# Studio

A single place for Data Engineers, Data Scientists, and IT Pros to collaborate on enterprise analytics

Microsoft Azure | Synapse Analytics ▶ prlangadws2

Overview Data Develop Orchestrate Monitor Manage

Synapse workspace  
prlangadws2

New ▾

Ingest Explore Analyze Visualize

Use the copy data tool to import data once or on a schedule.

Learn how to navigate and interact with your data.

Learn how to use SQL or Spark to get insights from your data.

Build interactive reports with integrated Power BI capabilities.

Resources

Recent Pinned

NAME	LAST OPENED BY YOU
GreenCabTransformation	a day ago
EXE2 StoredProceduresCabs	a day ago
EXE3 Query Market Share SQL Pool	a day ago
EXE5 Query SQL OD Views	a day ago
EXE5 Create SQL OD Views	a day ago

Show more ▾

Name: prlangadws2  
Region: West US 2  
Resource group: prlangadrg  
Subscription ID: 58f8824d-32b0-4825-9825-02fa6a801546  
[Select another workspace](#)

Useful links

[Synapse Analytics overview](#)  
Discover the capabilities offered by Synapse and learn how to make the most of them.

[Pricing](#)  
Learn about pricing details for Synapse capabilities.

[Documentation](#)  
Visit the documentation center for quickstarts, how-to guides, and references for PowerShell, APIs, etc.

[Give feedback](#)  
Share your comments or suggestions with us to improve Synapse.

# Synapse Studio

Synapse Studio divided into **Activity hubs**.

These organize the tasks needed for building analytics solution.

The screenshot shows the Microsoft Azure Synapse Studio interface. On the left, there's a sidebar with a red box highlighting the 'Overview' section. Below it, under 'Resources', are sections for 'Recent' and 'Pinned' items, showing entries like 'NYCTaxiFinalWUS2\_JB', 'NYCTaxiFinalWUS2\_R', 'HolidayDataPipeline', 'create notebook on s...', and 'Data flow 1'. At the bottom of the sidebar, it says 'Name: prlangadws2', 'Region: West US 2', 'Resource group: prlangadry', 'Subscription ID: 50000000-0000-0000-0000-000000000000', and 'Show more'. The main area is titled 'Synapse workspace prlangadws2' and contains six activity hubs: 'Overview', 'Data', 'Develop', 'Monitor', 'Orchestrate', and 'Manage'. Each hub has a corresponding icon and a brief description.

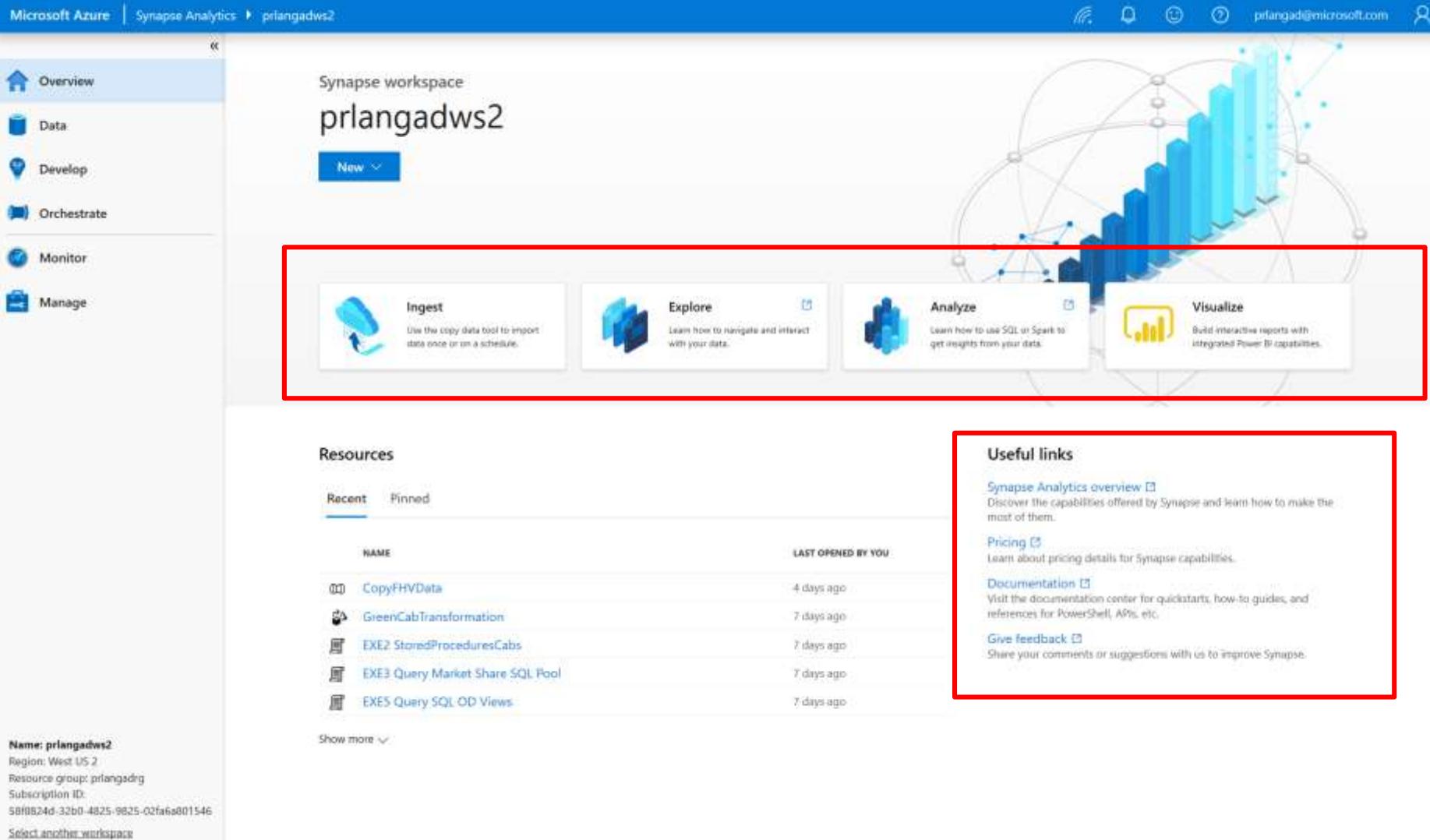
- Overview**: Quick-access to common gestures, most-recently used items, and links to tutorials and documentation.
- Data**: Explore structured and unstructured data.
- Develop**: Write code and define business logic of the pipeline via notebooks, SQL scripts, Data flows, etc.
- Monitor**: Centralized view of all resource usage and activities in the workspace.
- Manage**: Configure the workspace, pool, access to artifacts.
- Orchestrate**: Design pipelines that move and transform data.



# Synapse Studio Overview hub

# Overview Hub

It is a starting point for the activities with key links to tasks, artifacts and documentation



The screenshot shows the Microsoft Azure Synapse Analytics Overview Hub for the workspace "prlangadws2". The left sidebar includes links for Overview, Data, Develop, Orchestrate, Monitor, and Manage. The main area features a "Synapse workspace" header with the name "prlangadws2" and a "New" button. Below this is a large circular graphic illustrating data flow and analysis. Four cards are displayed: "Ingest" (use the copy data tool to import data once or on a schedule), "Explore" (learn how to navigate and interact with your data), "Analyze" (learn how to use SQL or Spark to get insights from your data), and "Visualize" (build interactive reports with integrated Power BI capabilities). A red box highlights these four cards. Further down, there are sections for "Resources" (Recent and Pinned items like CopyPHVData, GreenCapTransformation, EXE2\_StoredProcedureCabs, EXE3\_Query Market Share SQL Pool, and EXE5\_Query SQL OD Views) and "Useful links" (Synapse Analytics overview, Pricing, Documentation, and Give feedback). At the bottom, workspace details are listed: Name: prlangadws2, Region: West US 2, Resource group: prlangadrg, Subscription ID: 58f0824d-32b0-4825-9825-02fa6a801546, and a link to Select another workspace.

**Ingest**  
Use the copy data tool to import data once or on a schedule.

**Explore**  
Learn how to navigate and interact with your data.

**Analyze**  
Learn how to use SQL or Spark to get insights from your data.

**Visualize**  
Build interactive reports with integrated Power BI capabilities.

**Resources**

**Recent**

NAME	LAST OPENED BY YOU
CopyPHVData	4 days ago
GreenCapTransformation	7 days ago
EXE2_StoredProcedureCabs	7 days ago
EXE3_Query Market Share SQL Pool	7 days ago
EXE5_Query SQL OD Views	7 days ago

Show more ↴

**Useful links**

- Synapse Analytics overview** ⓘ  
Discover the capabilities offered by Synapse and learn how to make the most of them.
- Pricing** ⓘ  
Learn about pricing details for Synapse capabilities.
- Documentation** ⓘ  
Visit the documentation center for quickstarts, how-to guides, and references for PowerShell, APIs, etc.
- Give feedback** ⓘ  
Share your comments or suggestions with us to improve Synapse.

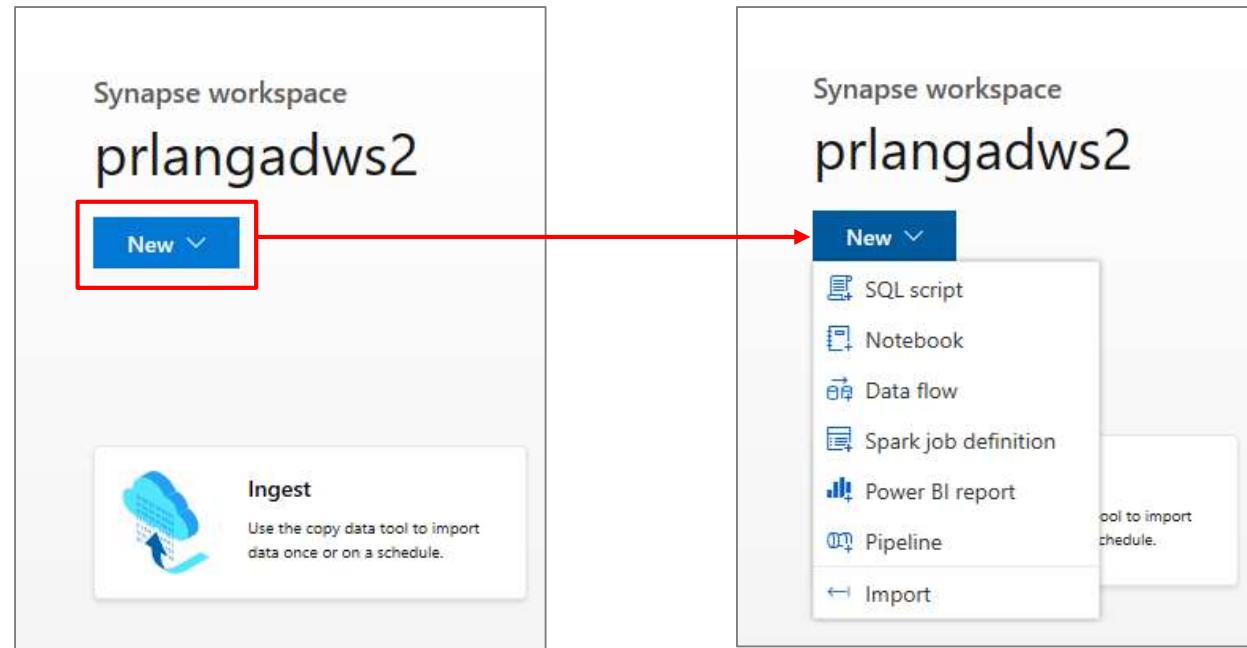
Name: prlangadws2  
Region: West US 2  
Resource group: prlangadrg  
Subscription ID: 58f0824d-32b0-4825-9825-02fa6a801546  
Select another workspace

# Overview Hub

## Overview

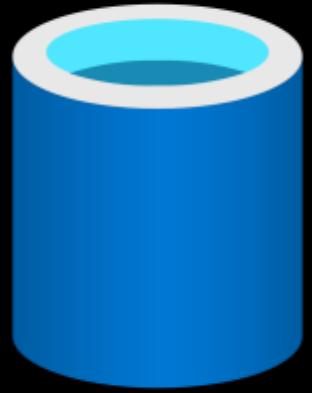
**New** dropdown – offers quickly start work item

**Recent & Pinned** – Lists recently opened code artifacts. Pin selected ones for quick access



Recent	Pinned
NAME	
BOOT_AMLautoMLPredict	LAST OPENED BY YOU 6 hours ago
SQLConnector	6 hours ago
TaxiCreateSparkTable	6 hours ago
Notebook 1	6 hours ago
NYCTAx1	6 hours ago
Show more ▾	

Recent	Pinned
NAME	
NYCTAx1	LAST OPENED BY YOU 6 hours ago



# Synapse Studio **Data hub**

# Data Hub

Explore data inside the workspace and in linked storage accounts

Microsoft Azure | Synapse Analytics > prlangadws2 |  | | prlangad@microsoft.com |

Overview | Data (selected) | Develop | Orchestrate | Monitor | Manage

Publish all Validate all Refresh Discard all

**Data**

Storage accounts

Databases

Datasets

Select an item from the resource explorer or create a new item

Name: prlangadws2  
Region: West US 2  
Resource group: prlangadrg  
Subscription ID:  
58f8824d-32b0-4825-9825-02fa6a801546  
[Select another workspace](#)

# Data Hub – Storage accounts

Browse Azure Data Lake Storage Gen2 accounts and filesystems – navigate through folders to see data

ADLS Gen2 Account

Container (filesystem)

The screenshot shows the Microsoft Azure Data Hub interface for managing storage accounts. On the left, a sidebar lists navigation options: Overview, Data (selected), Develop, Orchestrate, Monitor, and Manage. Below this, account details are shown: Name: prlangadws2, Region: West US 2, Resource group: prlangadrg, Subscription ID: 58f8824d-32b0-4825-9825-02fa6a801546. The main area is titled 'Data' and shows a 'Storage accounts' section. A red box highlights the 'prlangaddemosa (Primary)' storage account. Another red box highlights the 'nyctlc' container within it. The 'Filepath' bar shows the path 'nyctlc > yellow'. The right side displays a file browser with a list of folders named 'puYear=2001' through 'puYear=2026', each with a timestamp and 'Folder' type.

NAME	LAST MODIFIED	CONTENT TYPE	SIZE
puYear=2001	10/25/2019, 2:25:03 PM	Folder	
puYear=2002	10/25/2019, 2:25:21 PM	Folder	
puYear=2003	10/25/2019, 2:25:03 PM	Folder	
puYear=2008	10/25/2019, 2:20:38 PM	Folder	
puYear=2009	10/25/2019, 2:19:33 PM	Folder	
puYear=2010	10/25/2019, 2:19:24 PM	Folder	
puYear=2011	10/25/2019, 2:23:56 PM	Folder	
puYear=2012	10/25/2019, 2:20:01 PM	Folder	
puYear=2013	10/25/2019, 2:19:52 PM	Folder	
puYear=2014	10/25/2019, 2:24:06 PM	Folder	
puYear=2015	10/25/2019, 2:20:12 PM	Folder	
puYear=2016	10/25/2019, 2:19:21 PM	Folder	
puYear=2017	10/25/2019, 2:20:28 PM	Folder	
puYear=2018	10/25/2019, 2:24:38 PM	Folder	
puYear=2019	10/25/2019, 2:20:33 PM	Folder	
puYear=2020	10/25/2019, 2:24:47 PM	Folder	
puYear=2021	10/25/2019, 2:28:34 PM	Folder	
puYear=2026	10/25/2019, 2:20:20 PM	Folder	

# Data Hub – Storage accounts

Preview a sample of your data

The screenshot shows the Azure Synapse Studio Data Hub interface. On the left, there's a sidebar with sections for Data, Storage accounts, Databases, and Datasets. Under Storage accounts, 'nyctlc' is selected, and under it, 'filesystem' is also selected. The main area shows a file system structure with folders like 'synapse', 'temp', 'tmp', and files like 'dbo.StoreSales.parquet', 'dbo.StoreSales.txt', 'employee.json', and 'SampleCSVFile\_2kb.csv'. A context menu is open over the 'SampleCSVFile\_2kb.csv' file, with the 'Preview' option highlighted and a red arrow pointing from the menu to the preview pane on the right.

**SampleCSVFile\_2kb.csv**

**Path** [https://prlangaddemosa.dfs.core.windows.net/filesystem/SampleCSVFile\\_2kb.csv](https://prlangaddemosa.dfs.core.windows.net/filesystem/SampleCSVFile_2kb.csv)  
**Modified** 10/29/2019, 1:30:21 PM

**With column header**  On

USER_ID	USERNAME	FIRST_NAME	LAST_NAME	GENDER	PASSWORD
1	rogers63	david	john	Female	e6a33eee18
2	mike28	rogers	paul	Male	2e7dc6b8a1
3	rivera92	david	john	Male	1c3a8e03f4
4	ross95	maria	sanders	Male	62f0a68a41
5	paul85	morris	miller	Female	61bd060b07
6	smith34	daniel	michael	Female	7055b3d9f5
7	james84	sanders	paul	Female	b7f72d6eb9
8	daniel53	mark	mike	Male	299cbf7171
9	brooks80	morgan	maria	Female	aa736a35dc
10	morgan65	paul	miller	Female	a28dca31f5

**OK**

# Data Hub – Storage accounts

See basic file properties

The screenshot shows the Azure Synapse Studio Data Hub interface. On the left, there's a navigation sidebar with sections for Data, Storage accounts, Databases, and Datasets. Under Storage accounts, 'prlangaddemosa (Primary)' is expanded, showing its contents: filesystem, holidaydatacontainer, isdweatherdatacontainer, nyctlc, prlangaddemosa, tmpcontainer, and wwidporters. The 'filesystem' item is currently selected. In the main area, there are tabs for 'nyctlc' and 'filesystem'. The 'filesystem' tab is active, showing a list of files and folders. One file, 'SampleCSVFile\_2kb.csv', is selected. A context menu is open over this file, with options including Preview, New notebook, Copy ABFSS path, Manage Access..., Rename..., Download, Delete, and Properties... (which is highlighted with a red box and arrow).

The screenshot shows the 'Properties' dialog box for the file 'SampleCSVFile\_2kb.csv'. The 'System Properties' section contains the following information:

Name	SampleCSVFile_2kb.csv
URL	https://prlangaddemosa.dfs.core.windows.net/filesystem/SampleCSVFile_2kb.csv
LastModified	Tue, 29 Oct 2019 20:30:21 GMT
CacheControl	(empty)
ContentType	application/vnd.ms-excel
ContentDisposition	(empty)
ContentEncoding	(empty)
ContentLanguage	(empty)

The 'User Properties' section is currently empty. At the bottom of the dialog are 'Save' and 'Cancel' buttons.

# Data Hub – Storage accounts

Manage Access - Configure standard POSIX ACLs on files and folders

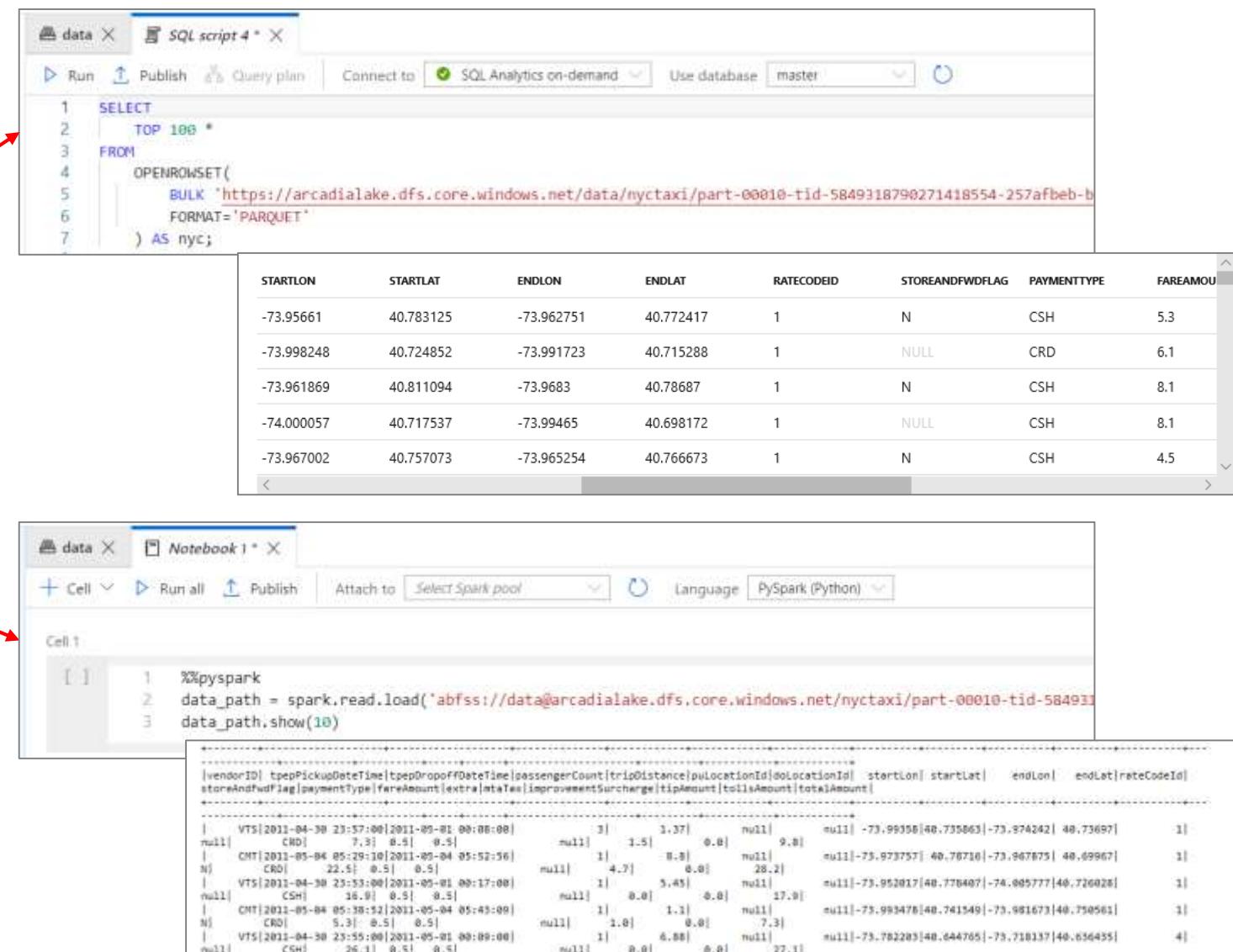
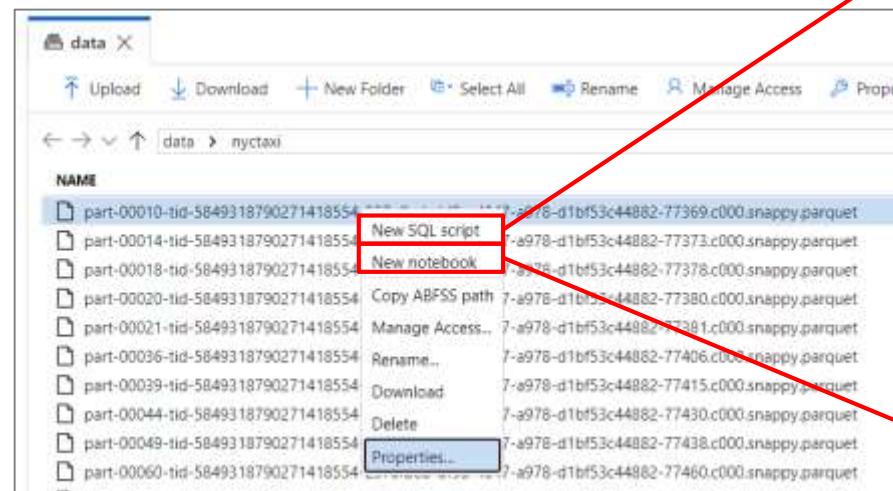
The screenshot shows the Azure Synapse Studio Data Hub interface. On the left, there's a navigation sidebar with sections for Data, Storage accounts, Databases, and Datasets. Under Storage accounts, 'prlangaddemo (Primary)' is selected, showing its contents: filesystem, holidaydatacontainer, isdweatherdatacontainer, nyctlc, prlangaddemo, tmpcontainer, and wwimporters. The 'filesystem' item is also selected. In the main workspace, there are two tabs: 'nyctlc' and 'filesystem'. The 'filesystem' tab is active, displaying a list of files and folders: synapse, temp, tmp, dbo.StoreSales.parquet, dbo.StoreSales.txt, employee.json, and SampleCSVFile\_2kb.csv. A context menu is open over the 'SampleCSVFile\_2kb.csv' file, listing options: Preview, New notebook, Copy ABFSS path, Manage Access..., Rename..., Download, Delete, and Properties... . The 'Manage Access...' option is highlighted with a red box and a red arrow pointing to the corresponding window on the right.

The screenshot shows the 'Manage Access' dialog box for the file 'SampleCSVFile\_2kb.csv'. The title bar says 'Manage Access' and the sub-header says 'Managing permissions for: filesystem/SampleCSVFile\_2kb.csv'. It lists 'Users and groups': '\$superuser (Owner)' and '\$superuser (Owning Group)'. Below that are sections for 'Other' and 'Mask'. Under 'Permissions for: \$superuser', there are checkboxes for 'Read', 'Write', and 'Execute', all of which are checked. There's also a section to 'Add user or group' with a text input field 'Enter a UPN or Object ID' and a 'Add' button. At the bottom are 'Save' and 'Cancel' buttons.

# Data Hub – Storage accounts

Two simple gestures to start analyzing with SQL scripts or with notebooks

T-SQL or PySpark auto-generated.



# Data Hub – Storage accounts

SQL Script from Multiple files

Multi-select of files generates a SQL script that analyzes all those files together

The screenshot shows the Azure Synapse Analytics Studio interface. On the left, there is a file browser window titled 'isdweatherdatacontainer > ISDWeathercurated'. It lists several Parquet files, including '\_SUCCESS' and multiple part files. A red arrow points from the 'New SQL script' option in a context menu (which is highlighted with a red box) to the query editor on the right. The query editor contains a T-SQL script for reading multiple Parquet files:

```
1 -- Read multiple parquet files with same schema
2 SELECT
3     TOP 100 *
4 FROM
5     OPENROWSET(
6         BULK 'https://prlangaddemoa.dfs.core.windows.net/isdweatherdatacontainer/ISDWeathercurated/+',
7         FORMAT = 'Parquet'
8     ) AS [r]
9 WHERE
10    r.filepath() in (
11        'https://prlangaddemoa.dfs.core.windows.net/isdweatherdatacontainer/ISDWeathercurated/part-00000'
12        'https://prlangaddemoa.dfs.core.windows.net/isdweatherdatacontainer/ISDWeathercurated/part-00001'
13        'https://prlangaddemoa.dfs.core.windows.net/isdweatherdatacontainer/ISDWeathercurated/part-00002'
14        'https://prlangaddemoa.dfs.core.windows.net/isdweatherdatacontainer/ISDWeathercurated/part-00003'
15        'https://prlangaddemoa.dfs.core.windows.net/isdweatherdatacontainer/ISDWeathercurated/part-00004'
16        'https://prlangaddemoa.dfs.core.windows.net/isdweatherdatacontainer/ISDWeathercurated/part-00005'
17        'https://prlangaddemoa.dfs.core.windows.net/isdweatherdatacontainer/ISDWeathercurated/part-00006'
18    )
```

# Data Hub – Databases

Explore the different kinds of databases that exist in a workspace.

The diagram illustrates the structure of databases in a workspace, comparing two views: a smaller left pane and a larger right pane. Red arrows point from specific items in the left pane to their corresponding detailed views in the right pane.

**Left Pane (Summary View):**

- Storage accounts:** 2 items
- Databases:** 3 items
  - sql1 (SQL pool)**
  - sample (SQL on-demand)**
  - default (Spark)**
    - Tables:** 2 items
      - nytaxiyellow7days**
      - searchlogtable**
- Datasets:** 2 items

**Right Pane (Detailed View):**

- Storage accounts:** 2 items
- Databases:** 3 items
  - sql1 (SQL pool)**
    - Tables**
    - External tables**
    - External resources**
    - Views**
    - Programmability**
    - Schemas**
    - Security**
  - sample (SQL on-demand)**
    - External tables**
    - External resources**
    - Views**
    - Schemas**
    - Security**
  - default (Spark)**
    - Tables**
- Datasets:** 2 items

SQL pool

SQL on-demand

Spark

# Data Hub – Databases

Familiar gesture to generate T-SQL scripts from SQL metadata objects such as tables.

A screenshot of the Azure Synapse Studio interface. On the left, there's a tree view labeled 'Databases' with one item under 'sql1 (SQL pool)'. Under 'sql1', there's a 'Tables' node which contains 'dbo.SearchLogTable' and 'dbo.NycTaxiPredict'. The 'dbo.NycTaxiPredict' node has a 'Columns' node underneath it. A context menu is open over the 'Columns' node, listing options: 'New SQL script', 'Select TOP 1000 rows', 'CREATE', 'DROP', and 'DROP and CREATE'.

Starting from a table, auto-generate a single line of PySpark code that makes it easy to load a SQL table into a Spark dataframe

A screenshot of the Azure Synapse Studio interface. The 'Databases' tree view shows 'sql1 (SQL pool)' with 'Tables' and 'dbo.NycTaxiPredict'. Under 'dbo.NycTaxiPredict', there's a 'Columns' node. A context menu is open over the 'Columns' node, with 'Load to DataFrame' highlighted by a red box. Other options in the menu include 'New SQL script', 'New notebook', and 'Refresh'. Below the interface, a 'Notebook 1' window is open, showing a single cell with the PySpark code: 'val df = spark.read.sqlanalytics("sql1 dbo.NycTaxiPredict")'.

# Data Hub – Datasets

Orchestration datasets describe data that is persisted. Once a dataset is defined, it can be used in pipelines and sources of data or as sinks of data.

The screenshot shows the Azure Synapse Analytics Studio interface for managing datasets. On the left, a sidebar titled 'Data' lists resources: Storage accounts (2), Databases (3), and Datasets (2). The 'NYCTaxiParquet' dataset is highlighted with a red box and has a red arrow pointing to its configuration page on the right. The main area displays the 'NYCTaxiParquet' dataset details, including its Parquet file icon and name. The configuration page includes tabs for General, Connection, Schema, and Parameters. Under the Connection tab, the linked service is set to 'Lake\_ArcadiaLake'. The File path is specified as 'data / nyctaxi / File'. The Compression type is set to 'snappy'. There are also buttons for Test connection, Open, New, Browse, and Preview data.



# Synapse Studio

## Develop hub

# Develop Hub

## Overview

It provides development experience to query, analyze, model data

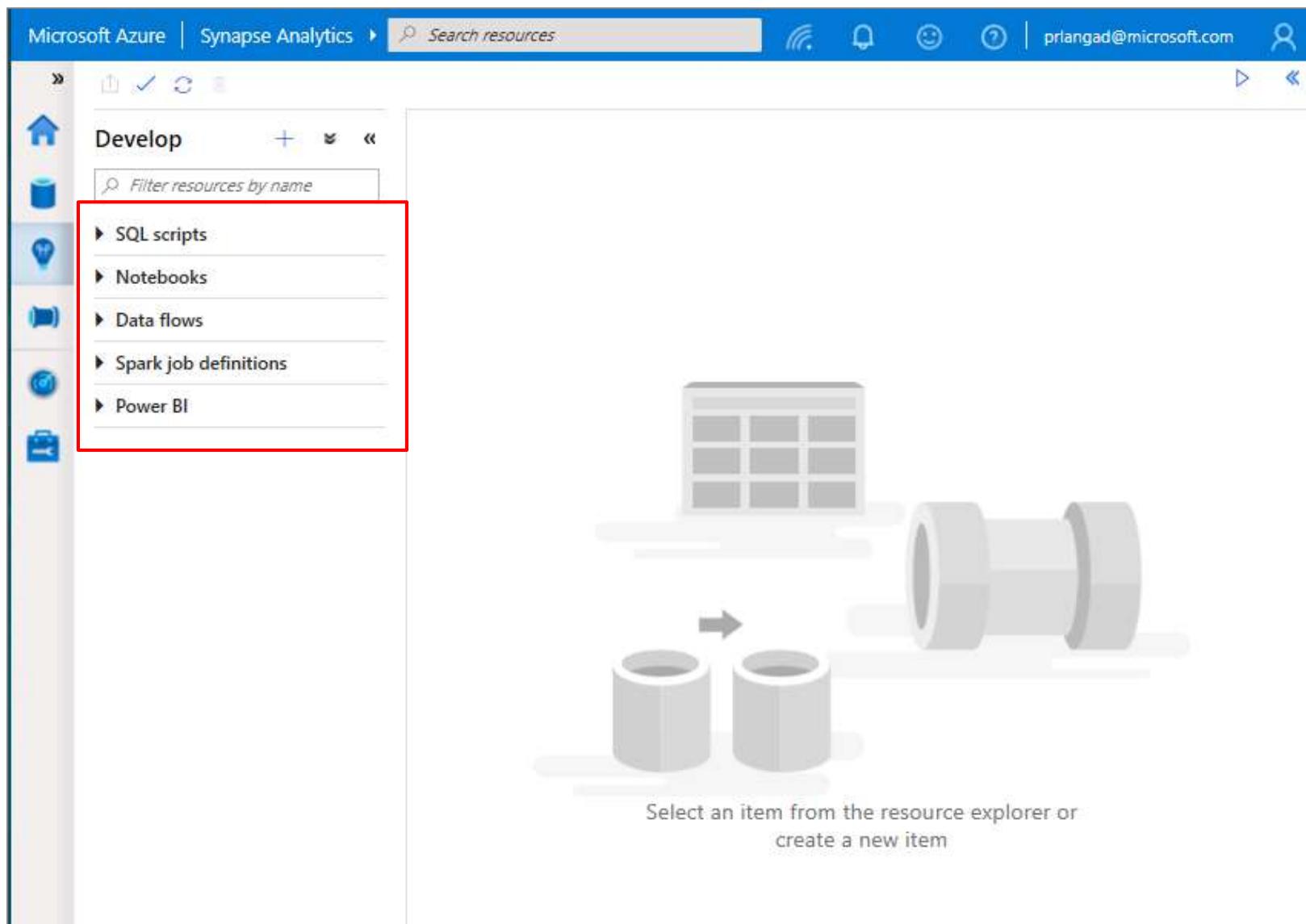
## Benefits

Multiple languages to analyze data under one umbrella

Switch over notebooks and scripts without loosing content

Code intellisense offers reliable code development

Create insightful visualizations



# Develop Hub - SQL scripts

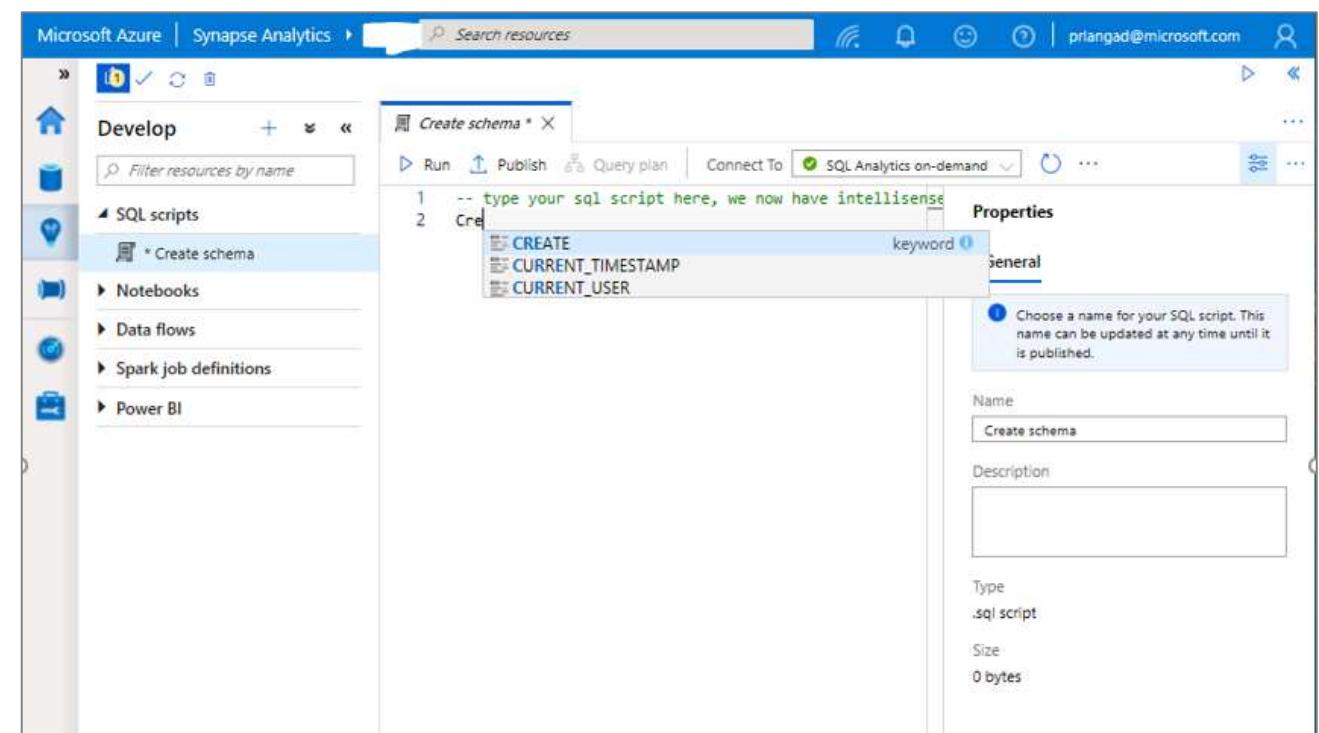
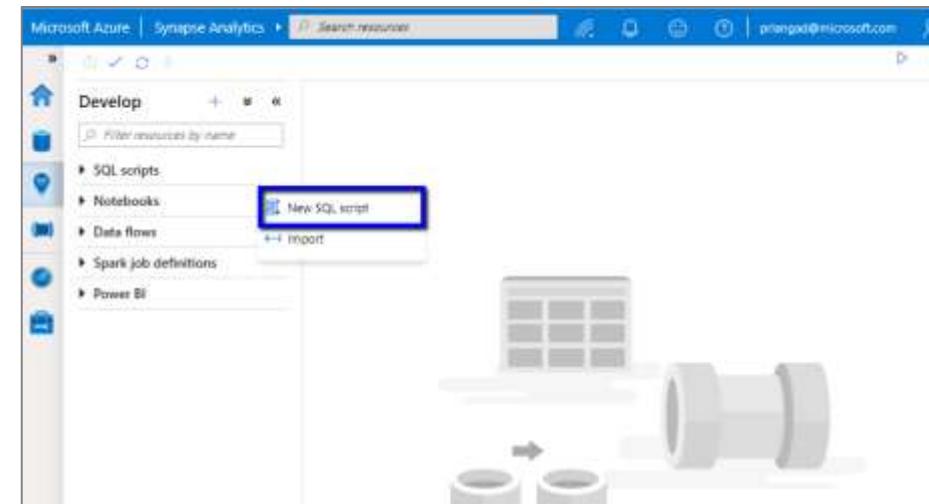
## SQL Script

### Authoring SQL Scripts

Execute SQL script on provisioned SQL Pool or SQL On-demand

Publish individual SQL script or multiple SQL scripts through Publish all feature

Language support and intellisense



# Develop Hub - SQL scripts

## SQL Script

View results in Table or Chart form and export results in several popular formats

The screenshot shows the Azure Synapse Studio interface. On the left, a query editor window titled "SearchLog\_que..." displays a T-SQL script for reading data from a CSV file into a temporary table named "searchlog". The script uses OPENROWSET and BULK options to import data from a URL. Below the script, the "Results" tab is selected, showing a table with columns ID, TIME, and REGION. The table contains four rows of data. At the bottom of the results pane, there are buttons for "Table" (selected), "Chart", and "Export results". A red box highlights the "Export results" button, and a red arrow points from it to a larger callout box on the right.

**Results**

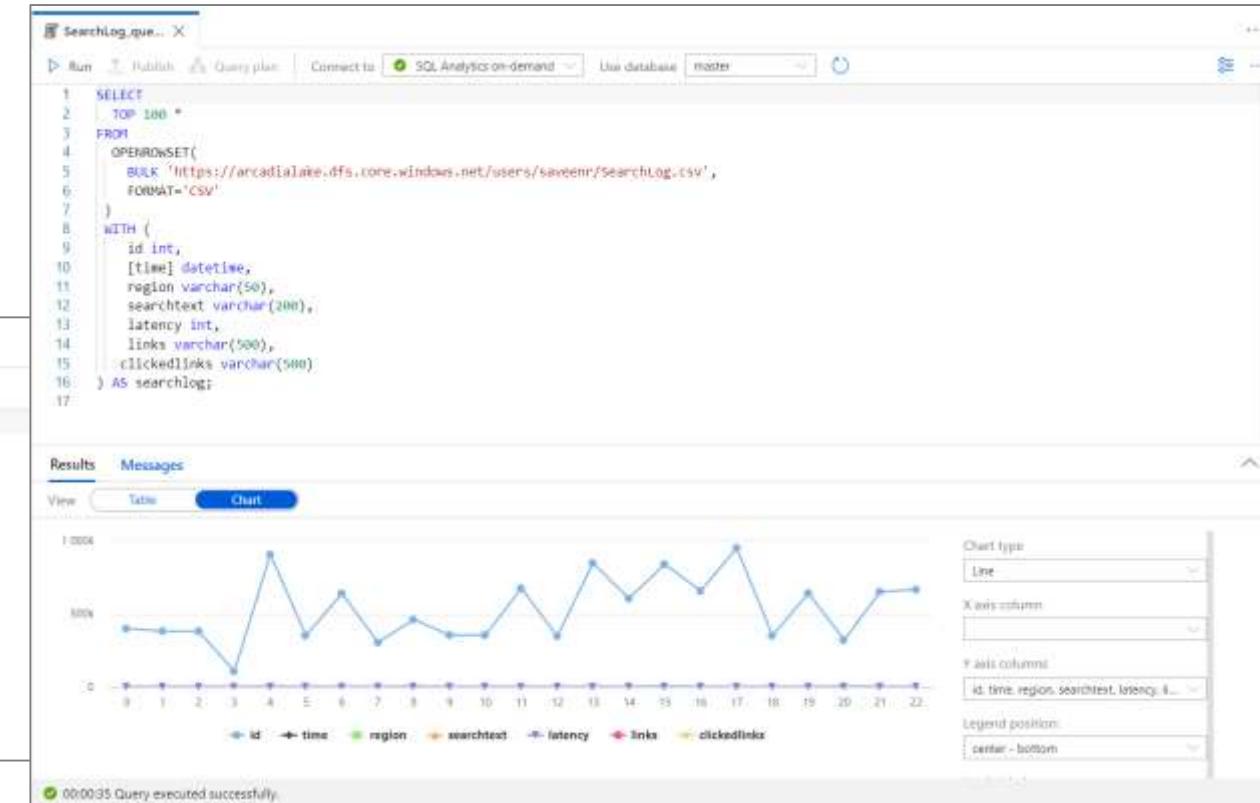
ID	TIME	REGION
399266	2019-10-15T11:53:04.0000000	en-us
382045	2019-10-15T11:53:25.0000000	en-gb
382045	2019-10-16T11:53:42.0000000	en-gb
106479	2019-10-16T11:53:10.0000000	en-ca
906441	2019-10-16T11:54:18.0000000	en-us

**Messages**

00:00:35 Query executed successfully.

**Export results**

- CSV
- Excel
- JSON
- XML



# Develop Hub - Notebooks

## Notebooks

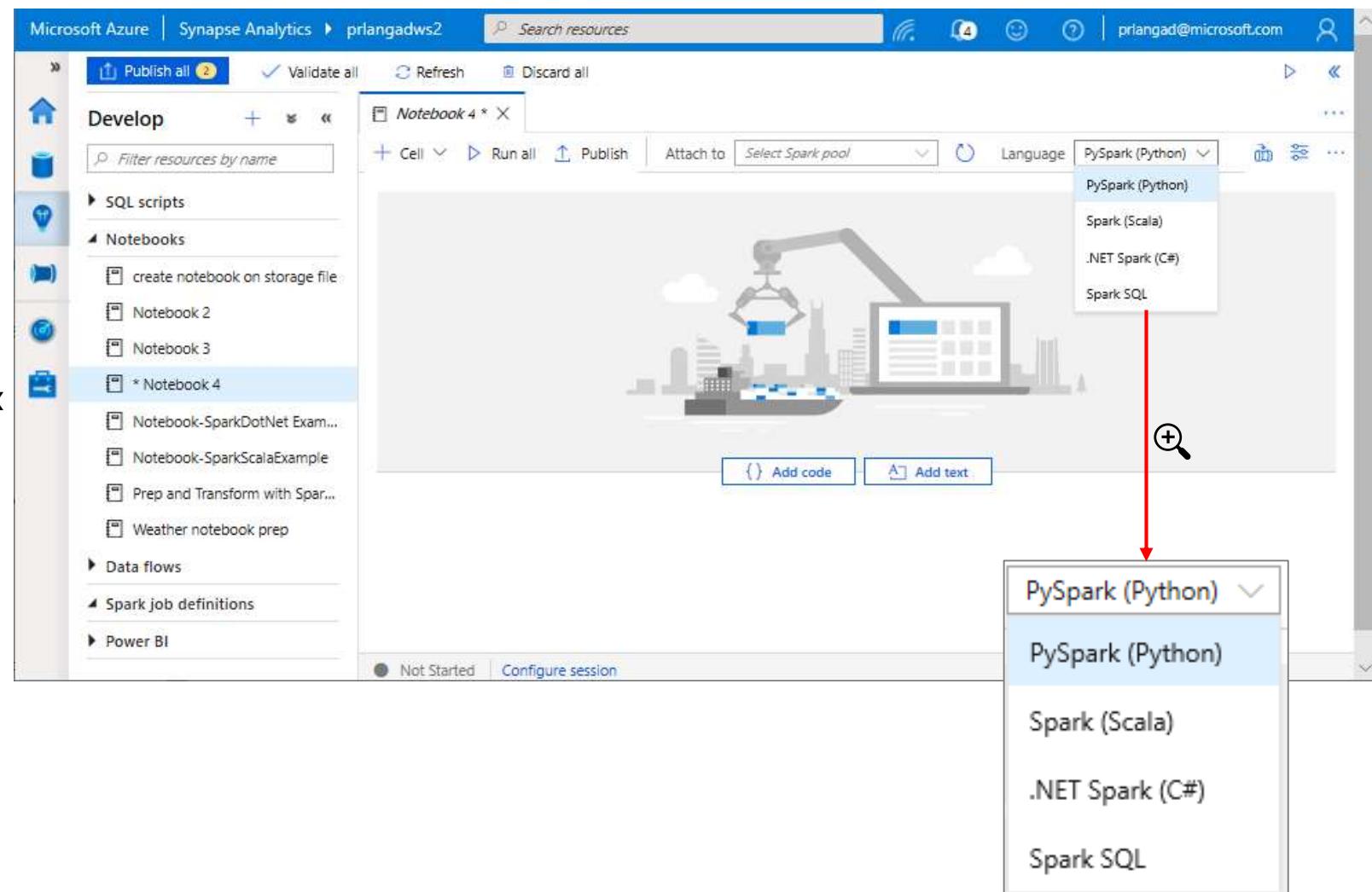
Allows to write multiple languages in one notebook

`%%<Name of language>`

Offers use of temporary tables across languages

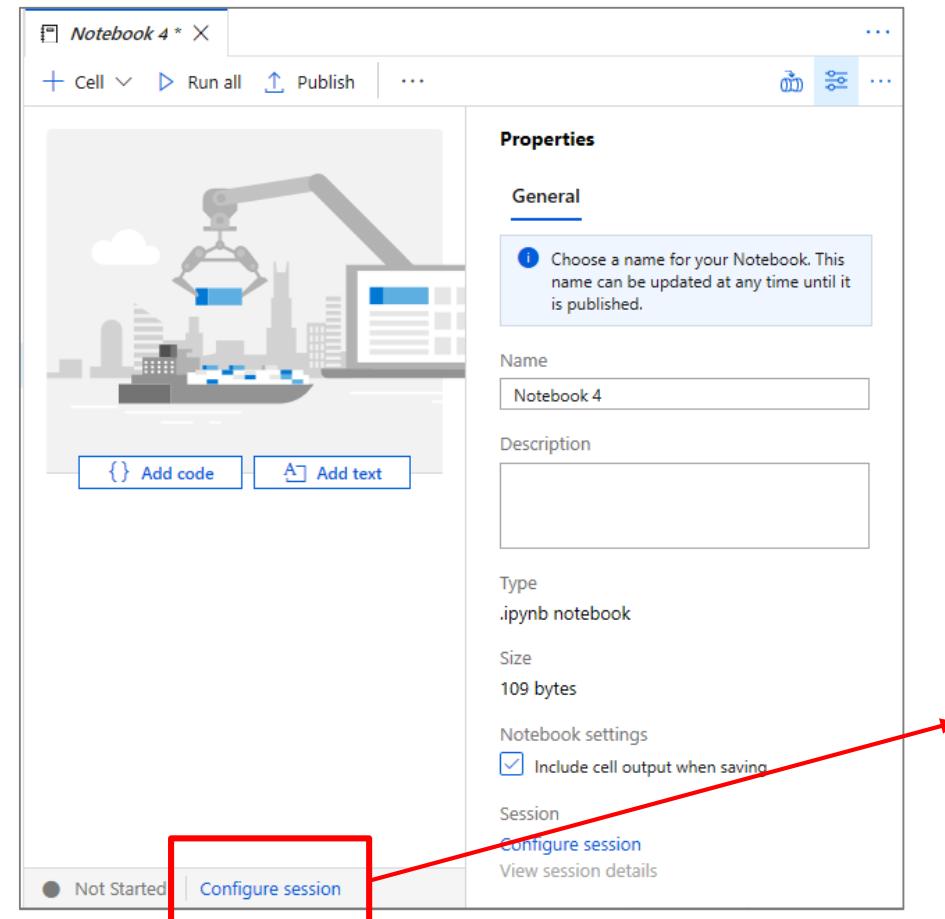
Language support for Syntax highlight, syntax error, syntax code completion, smart indent, code folding

Export results



# Develop Hub - Notebooks

Configure session allows developers to control how many resources are devoted to running their notebook.



### Configure session

**Notebook 4**

\* Session timeout

\* Executors

\* Executor size

\* Driver size

**Apply** **Cancel**

# Develop Hub - Notebooks

As notebook cells run, the underlying Spark application status is shown. Providing immediate feedback and progress tracking.

The screenshot shows the Azure Synapse Studio interface for developing notebooks. At the top, there's a toolbar with options like Cell, Run all, Publish, Attach to, Language (set to PySpark (Python)), and a session dropdown (spark1). Below the toolbar, a cell titled "Cell 1" contains the following PySpark code:

```

1 %%pyspark
2 data_path = spark.read.load('abfss://data@arcadialake.dfs.core.windows.net/nyctaxi/part-00010-tid-584931879')
3 data_path.show(10)

```

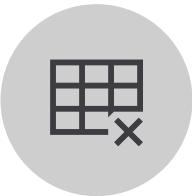
Below the code, a message indicates the command was executed in 4mins 33s 99ms by saveenr on 11-16-2019 09:36:31.944 -08:00. A section titled "Job execution Succeeded" shows three tasks (Job 0, Job 1, Job 2) completed on Spark 2 executors with 8 cores. Each task has a green checkmark and a duration of 14s, 21s, and 4s respectively. To the right, there are links to "View in monitoring" and "Spark history server".

At the bottom, a preview of the DataFrame is shown with columns: vendorID, tpepPickupDateTime, tpepDropoffDateTime, passengerCount, tripDistance, puLocationId, doLocationId, startLon, startLat, endLon, endLat, rateCodeId, and lpepDropoffDateTime. The data is presented as a table of rows, with the first few rows visible:

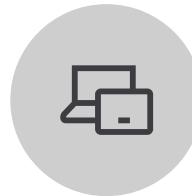
	tpepPickupDateTime	tpepDropoffDateTime	passengerCount	tripDistance	puLocationId	doLocationId	startLon	startLat	endLon	endLat	rateCodeId	lpepDropoffDateTime
null	2 2017-03-09 21:30:11 2017-03-09 21:44:20		1	14.0	0.5	0.5		4.06	148	48	null	null
null	1	N	1	14.0	0.5	0.5		0.3	3.06	0.0	18.36	
null	2 2017-03-09 21:47:00 2017-03-09 21:58:01		1		2.73			48	107	null	null	
null	1	N	2	11.5	0.5	0.5		0.3	0.0	0.0	12.8	
null	2 2017-03-09 22:01:08 2017-03-09 22:11:16		1		2.27			79	162	null	null	
null	1	N	1	10.0	0.5	0.5		0.3	2.82	0.0	14.12	
null	2 2017-03-09 22:16:05 2017-03-10 06:26:11		1		3.86			237	41	null	null	
null	1	N	1	12.0	0.5	0.5		0.3	3.99	0.0	17.29	
null	2 2017-03-31 06:31:53 2017-03-31 06:41:48		1		3.45			41	162	null	null	
null	1	N	2	12.0	0.5	0.5		0.3	0.0	0.0	13.3	
null	1 2017-03-01 00:00:00 2017-03-01 00:14:22		1		2.8			261	79	null	null	
null	1	N	1	12.5	0.5	0.5		0.3	1.0	0.0	14.8	
null	1 2017-03-01 00:00:00 2017-03-01 00:19:30		1		6.0			87	142	null	null	
null	1	N	1	19.5	0.5	0.5		0.3	3.5	0.0	24.3	

At the bottom of the notebook cell, there are buttons for Ready, Stop session, Spark history server, and Configure session.

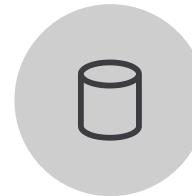
# Dataflow Capabilities



Handle upserts, updates, deletes on sql sinks



Add new partition methods



Add schema drift support



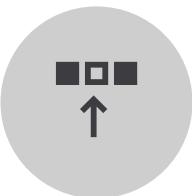
Add file handling (move files after read, write files to file names described in rows etc)



New inventory of functions (for e.g Hash functions for row comparison)



Commonly used ETL patterns(Sequence generator/Lookup transformation/SCD...)



Data lineage – Capturing sink column lineage & impact analysis(invaluable if this is for enterprise deployment)

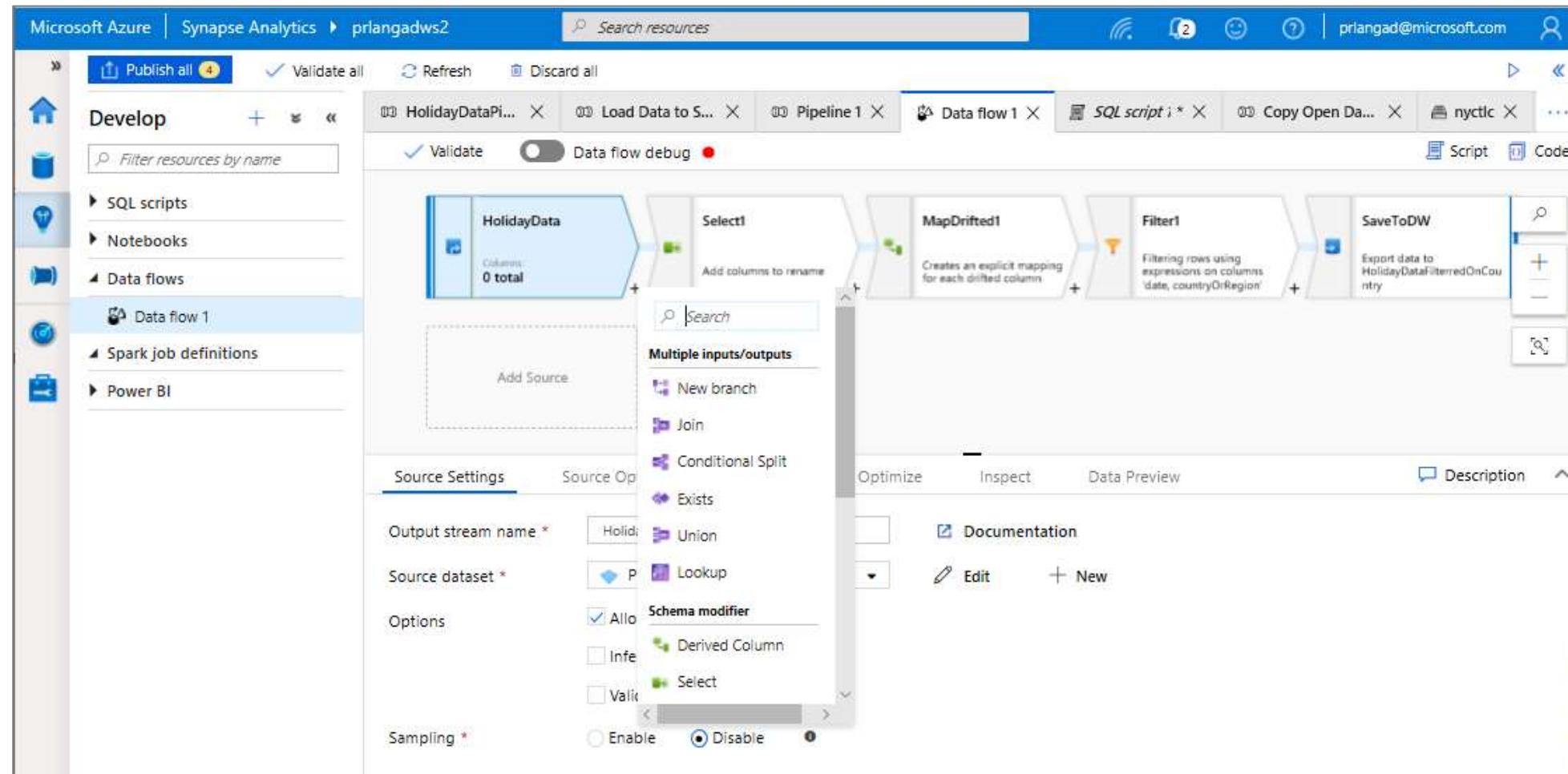


Implement commonly used ETL patterns as templates(SCD Type1, Type2, Data Vault)

# Develop Hub - Data Flows

Data flows are a visual way of specifying how to transform data.

Provides a code-free experience.



# Develop Hub – Power BI

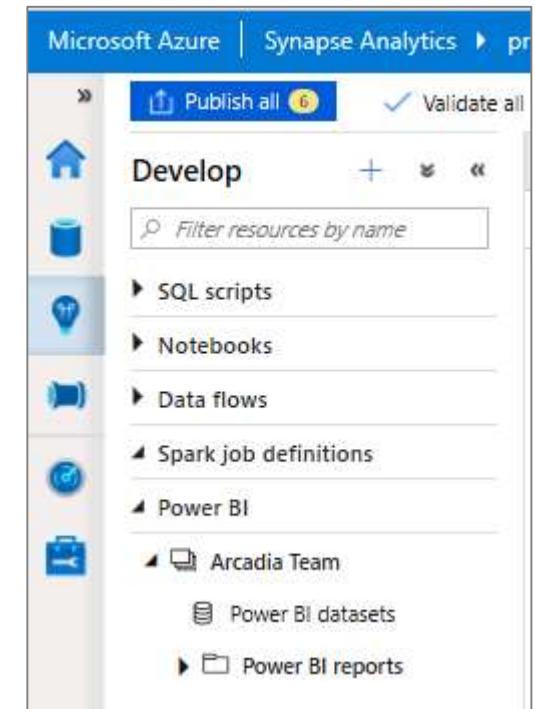
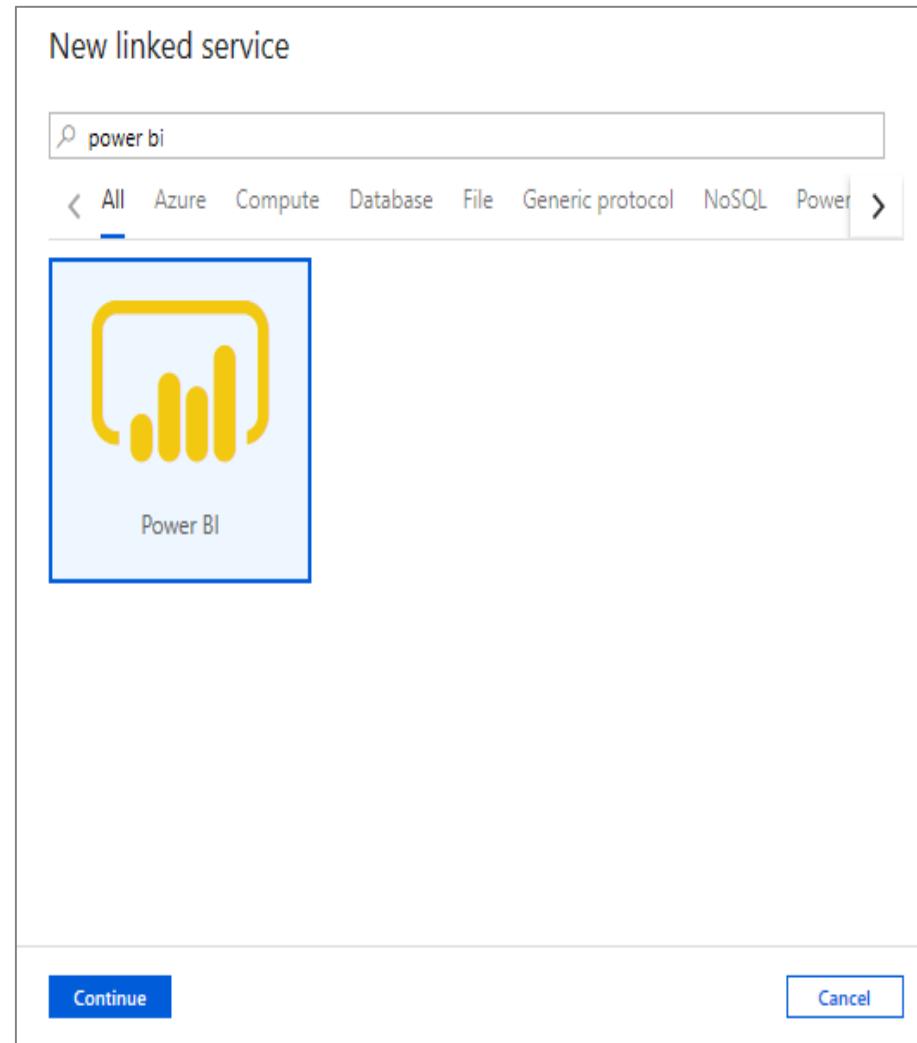
## Overview

Create Power BI reports in the workspace

Provides access to published reports in the workspace

Update reports real time from Synapse workspace to get it reflected on Power BI service

Visually explore and analyze data



# Develop Hub – Power BI

View published reports in Power BI workspace

The screenshot shows the Azure Synapse Analytics Develop Hub interface for a Power BI workspace. The left sidebar lists resources under 'Develop', including Notebooks (yellowcabprep, YellowCabExploration\_sqld), Data flows (PrepareCabDataFlow), Spark job definitions, and Power BI (SynapseNYTaxiInsights, Power BI Datasets, Power BI Reports). A report titled 'SynapseNYIgnite2019' is currently selected. The main area displays a line chart with two data series: a blue line showing a general upward trend and a yellow line showing more volatile fluctuations. The top navigation bar includes 'Publish all' (with 2 pending), 'Validate all', 'Refresh', and 'Discard all'. The right side features sections for 'VISUALIZATIONS' (containing icons for various chart types) and 'FIELDS' (listing data fields like dimHoliday, dimNYCLocations, Fhv, GreenCab, PredictedValues, wwFhvMarketShare, wwGrnCabMarketS..., wwMarketShare..., wwPredictedValues, wwYelCabMarketSh..., weather, YellowCab, YellowCabTripsHoli...), along with 'Filters' and 'Drillthrough' settings.

# Develop Hub – Power BI

Edit reports in Synapse workspace

The screenshot shows the Azure Synapse Analytics Develop Hub interface for Power BI. On the left, the sidebar lists various resources: Notebooks (e.g., AMLAutoMLPredict, AutoML, Data Download\_Weather, PrepareCabDataFlow, yellowcabprep, YellowCabPrepare), Data flows (e.g., PrepareCabDataFlow), Spark job definitions, and Power BI datasets (e.g., SynapseNYTaxiInsights, SynapseNYIgnite2019). The main area displays a Power BI report titled "SynapseNYIgnite2019". The report contains two visualizations: a line chart showing trip counts over time and a bar chart showing trip counts by holiday name. The Power BI ribbon at the top includes Publish all, Validate all, Refresh, and Discard all buttons. The ribbon tabs show the current report and other open items like yellowcabprep, PrepareTaxiData, 1 Marketshare, 2 MostTripsHo..., AutoML, and SynapseNYIgnite2019 (1). The right side of the screen shows the Power BI visualizations pane, which includes sections for Filters, Visualizations, Fields, Axis, Legend, Value, Tooltips, DRILLTHROUGH, and Cross-report. The "Value" section is currently selected, showing fields like numTrips, date, holidayName, and year.

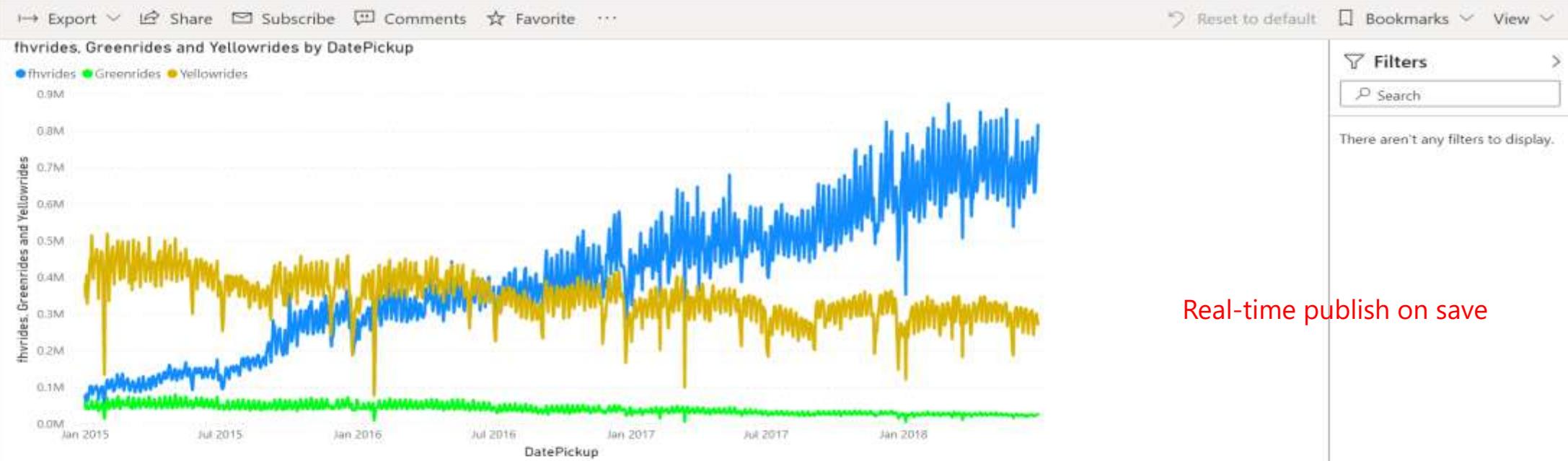
# Develop Hub – Power BI

Publish edited reports in Synapse workspace to Power BI workspace

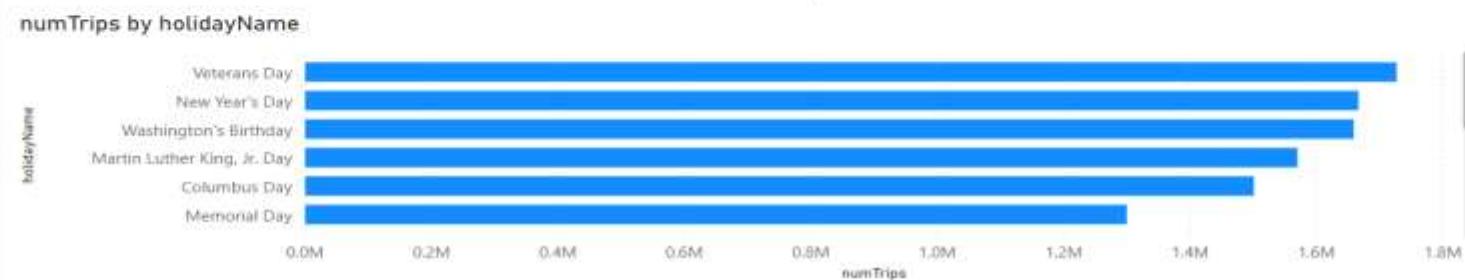
The screenshot shows the Azure Synapse Analytics Develop Hub interface. On the left, there's a sidebar with icons for Home, Datasets, Notebooks, Data Flows, Spark job definitions, Power BI, and a specific workspace named 'SynapseNYIgnite2019'. Below that, under 'Power BI', is a folder 'Power BI Reports' containing 'SynapseNYIgnite2019' and 'SynapseNYIgnite2019 (1)'. The main area displays a Power BI report with a red box highlighting the 'Save' button in the top-left corner of the report view. A red arrow points from this highlighted button to the text 'Publish changes by simple save report in workspace' located below the report preview. The report itself contains a line chart and a bar chart. To the right of the report, there are several panes: 'VISUALIZATIONS' showing various chart and table icons, 'FIELDS' listing columns like 'dimHoliday', 'dimNYCLocations', etc., and 'FILTERS' which show filters for 'holidayName' and 'numTrips'. The bottom of the screen shows navigation buttons for 'Page 1' and a '+' sign.

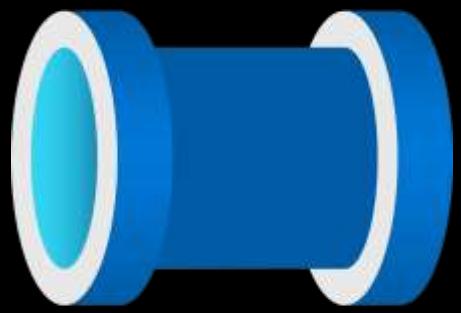
Publish changes by simple save report in workspace

- Home
- Favorites
- Recent
- Apps
- Shared with me
- Workspaces
- SynapseNYTaxiInsi...



Real-time publish on save



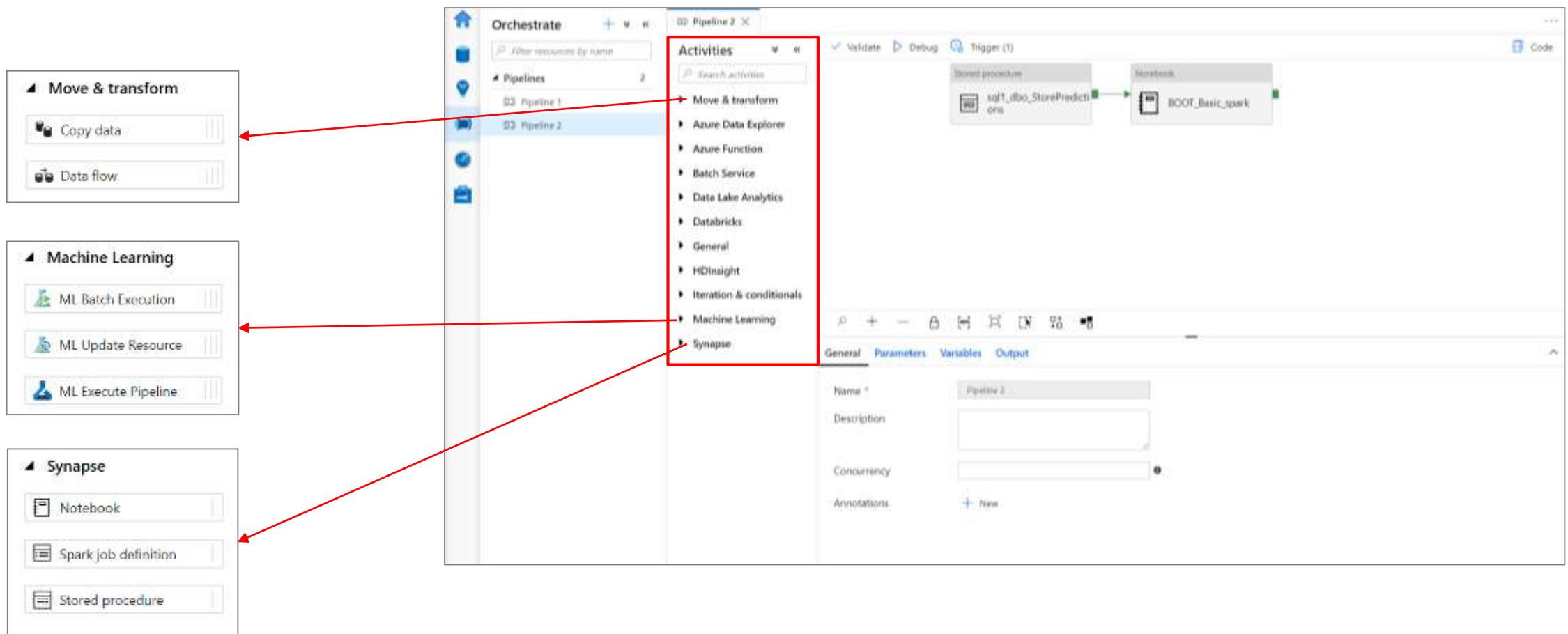


# Synapse Studio Orchestrate hub

# Orchestrate Hub

It provides ability to create pipelines to ingest, transform and load data with 90+ inbuilt connectors.

Offers a wide range of activities that a pipeline can perform.





# Synapse Studio Monitor hub

# Monitor Hub

## Overview

This feature provides ability to monitor orchestration, activities and compute resources.

The screenshot shows the Microsoft Azure Synapse Analytics Monitor Hub interface. The left sidebar has categories: Orchestration (selected), Pipeline runs, Trigger runs, Integration runtimes, Activities, Spark applications, Computes, and SQL Pools. The main area is titled "Pipeline runs" with filters: Time : 24 hours (default), Time zone : Pacific Time (US & Canada) (UT...), Runs : Latest runs, List (selected), Gantt, All status, Rerun, Cancel, Refresh, and Edit columns. It lists two pipeline runs:

Pipeline Name	Run Start	Duration	Triggered By	Status
Load Data to SQLDW	10/25/2019, 3:49:42 PM	00:10:55	Manual trigger	Succeeded
Copy Open Dataset	10/25/2019, 2:17:54 PM	00:14:12	Manual trigger	Succeeded

# Monitoring Hub - Orchestration

## Overview

Monitor orchestration in the Synapse workspace for the progress and status of pipeline

## Benefits

Track all/specific pipelines

Monitor pipeline run and activity run details

Find the root cause of pipeline failure or activity failure

Pipeline runs					
Time : Last week (10/24/2019 9:44 AM - 10/31/2019 9:44 AM)		Time zone : Pacific Time (US & Canada) (UT...)		Runs : Latest runs	
All status	Rerun	Cancel	Refresh	Edit columns	
<input type="checkbox"/> PIPELINE NAME	RUN START		DURATION	TRIGGERED BY	STATUS
<a href="#">Load Data to SQLDW</a>	10/25/2019, 3:49:42 PM		00:10:55	Manual trigger	<span style="color: green;">✓ Succeeded</span>
<a href="#">Copy Open Dataset</a>	10/25/2019, 2:17:54 PM		00:14:12	Manual trigger	<span style="color: green;">✓ Succeeded</span>
<a href="#">Pipeline 1</a>	10/24/2019, 1:23:43 PM		00:00:08	Manual trigger	<span style="color: green;">✓ Succeeded</span>

# Monitoring Hub - Spark applications

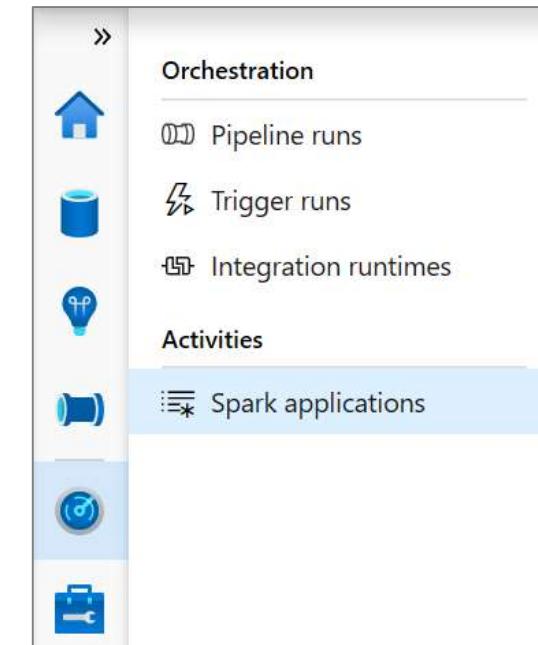
## Overview

Monitor Spark pools, Spark applications for the progress and status of activities

## Benefits

Monitor Spark pools for the status as paused, active, resume, scaling and upgrading

Track the usage of resources



Spark applications						
Submit time : 24 hours (default) (10/30/2019 9:52 AM - 10/31/2019 9:52 AM)		Time zone : Pacific Time (US & Canada) (UT...		<a href="#">List</a> <span style="margin-left: 10px;"><a href="#">Chart</a></span>		
All types	Cancel	Refresh	Edit columns			
APPLICATION NAME	SUBMITTER	SUBMIT TIME	STATUS	POOL	TYPE	
<a href="#">Synapse_prlang-syntax...</a>	prlangad@microsoft.com	10/30/2019 1:21 PM	Cancelled	prlang-syntaxcheck	Spark session	
<a href="#">Synapse_prlSpark_1572...</a>	prlangad@microsoft.com	10/30/2019 1:06 PM	Cancelled	prlSpark	Spark session	



# Synapse Studio

## Manage hub

# Manage Hub

## Overview

This feature provides ability to manage Linked Services, Orchestration and Security.

The screenshot shows the Microsoft Azure Synapse Analytics Studio Manage Hub. The left sidebar has icons for External connections, Linked services (which is selected), Orchestration, Triggers, Integration runtimes, Security, and Access control. The top navigation bar includes Publish all, Validate all, Refresh, Discard all, and user information (prlangad@microsoft.com). The main area is titled "Linked services" and contains a description: "Linked services are much like connection strings, which define the connection information needed for Arcadia to connect to external resources." Below this is a table with columns: NAME, TYPE, and ANNOTATIONS. The table lists the following linked services:

NAME	TYPE	ANNOTATIONS
ADLSG2OpenDataSetSink	Azure Data Lake Storage Gen2	
AzureBlobStorage1	Azure Blob Storage	
AzureDataLakeStorage1	Azure Data Lake Storage Gen2	
AzureDataLakeStorage2Source	Azure Data Lake Storage Gen2	
AzureOpenDataset	Azure Blob Storage	
AzureOpenDataSet2	Azure Blob Storage	
AzureSqlDW1	Azure Synapse Analytics (formerly SQL DW)	
AzureSynapseAnalytics1	Azure Synapse Analytics (formerly SQL DW)	
AzureSynapseAnalytics2	Azure Synapse Analytics (formerly SQL DW)	
PowerBIWorkspace1	Power BI	

# Manage – Linked services

## Overview

It defines the connection information needed to connect to external resources.

## Benefits

Offers pre-build 90+ connectors

Easy cross platform data migration

Represents data store or compute resources

Microsoft Azure | Synapse Analytics

External connections

Linked services

Linked services are much like connection strings, which define the connection information needed for Arcadia to connect to external resources.

+ New

Search to filter items...

NAME	TYPE	ANNOTATIONS
ADLSG2OpenDataSetSink	Azure Data Lake Storage Gen2	
AzureBlobStorage1	Azure Blob Storage	
AzureDataLakeStorage1	Azure Data Lake Storage Gen2	
AzureDataLakeStorage2Source		
AzureOpenDataset		
AzureOpenDataSet2		
AzureSqlDW1		

New linked service:

Power BI	Presto (Preview)	QuickBooks (Preview)
REST	SAP BW Open Hub	SAP BW via MDX
SAP Cloud for Customer	SAP ECC	SAP HANA
SAP	SAP	SAP

Continue Cancel

# Manage – Access Control

## Overview

It provides access control management to workspace resources and artifacts for admin and users

## Benefits

Share workspace with the team

Increases productivity

Manage permissions on code artifacts and Spark pools

Microsoft Azure | Synapse Analytics > prlangad

External connections  
Linked services  
Orchestration  
Triggers  
Integration runtimes  
Security  
**Access control**

**Add admin** Refresh Remove Search to filter items...

Showing 1 of 1 items

NAME	PERMISSIONS
Priyanka Langade Priyanka.Langade@microsoft.com	Full permissions on code artifacts and Spark pools. Learn more about SQ

Add admin

An admin has full control over code artifacts, can attach to Spark pools, and can schedule pipelines. Permissions to Storage accounts and SQL pool databases are managed on the resources directly. [Learn more](#)

\* Select user

Search by name or email address

Selected individual, groups or apps  
No individual, groups, or apps selected.

Apply Cancel

# Manage – Triggers

## Overview

It defines a unit of processing that determines when a pipeline execution needs to be kicked off.

## Benefits

Create and manage

- Schedule trigger
- Tumbling window trigger
- Event trigger

Control pipeline execution

The screenshot shows the Azure Synapse Analytics portal with the 'Triggers' section selected. A red box highlights the '+ New' button in the top-left corner of the trigger list. A red arrow points from this button to a detailed configuration dialog box titled 'New trigger' on the right side of the screen.

**New trigger**

Choose a name for your trigger. This name can be updated at any time until it is published.

**Name \***  
Trigger 2

**Description**

**Type \***  
 Schedule    Tumbling window    Event

**Start Date (UTC) \***  
10/29/2019 9:46 PM

**Recurrence \***  
Every 1 Minute(s)

**End \***  
 No End    On Date

**Annotations**  
+ New

**Activated \***  
 Yes    No

**OK**   **Cancel**

NAME ↑	TYPE ↑	STATUS ↑	NUMBER OF PIPELINES ↑	ANNOTATIONS ↑
* CopyParquetDataTrigger	Schedule	Started	1	
* Trigger 1	Schedule	Stopped	0	

# Manage – Integration runtimes

## Overview

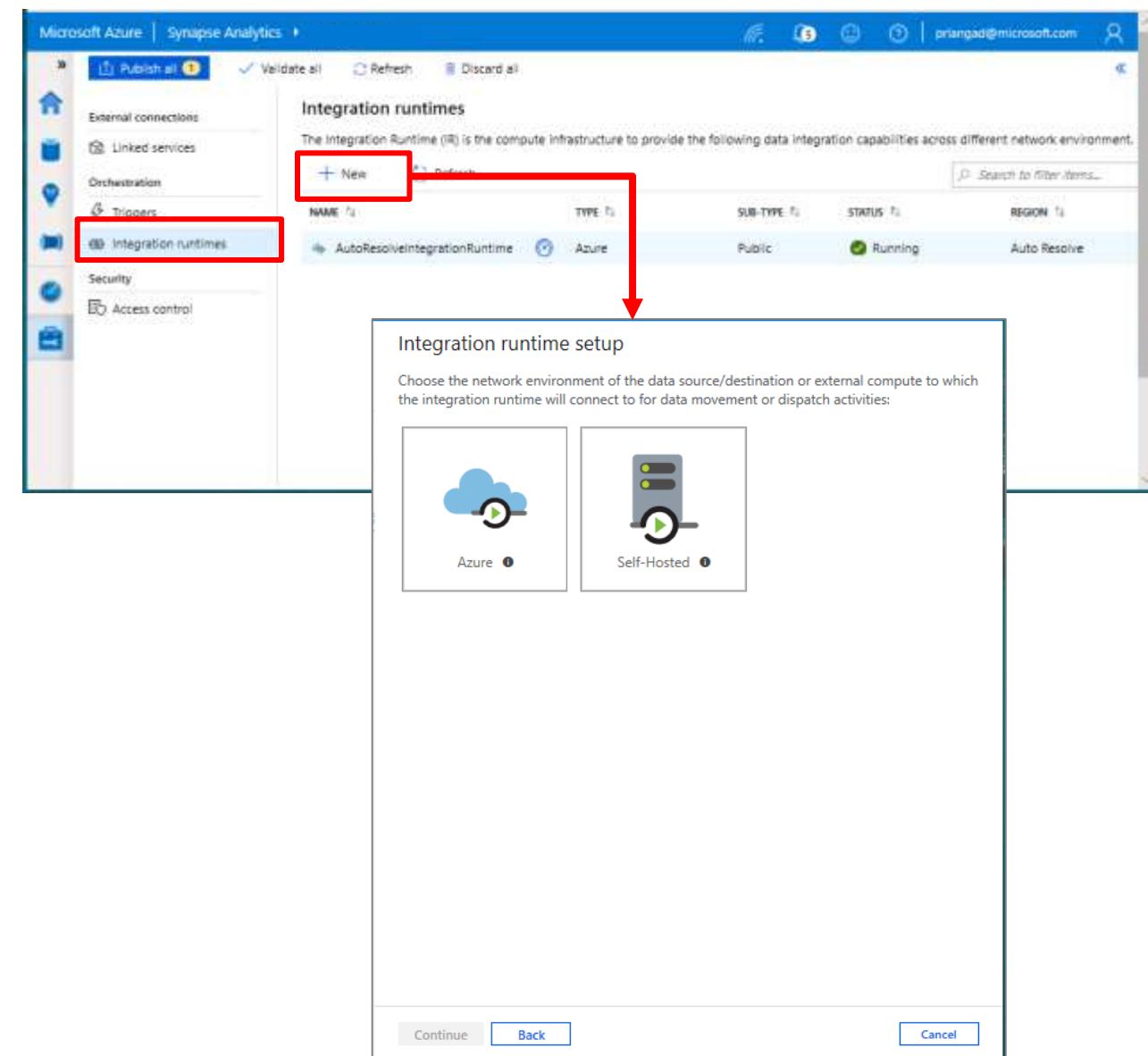
Integration runtimes are the compute infrastructure used by Pipelines to provide the data integration capabilities across different network environments. An integration runtime provides the bridge between the activity and linked services.

## Benefits

Offers Azure Integration Runtime or Self-Hosted Integration Runtime

Azure Integration Runtime – provides fully managed, serverless compute in Azure

Self-Hosted Integration Runtime – use compute resources in on-premises machine or a VM inside private network



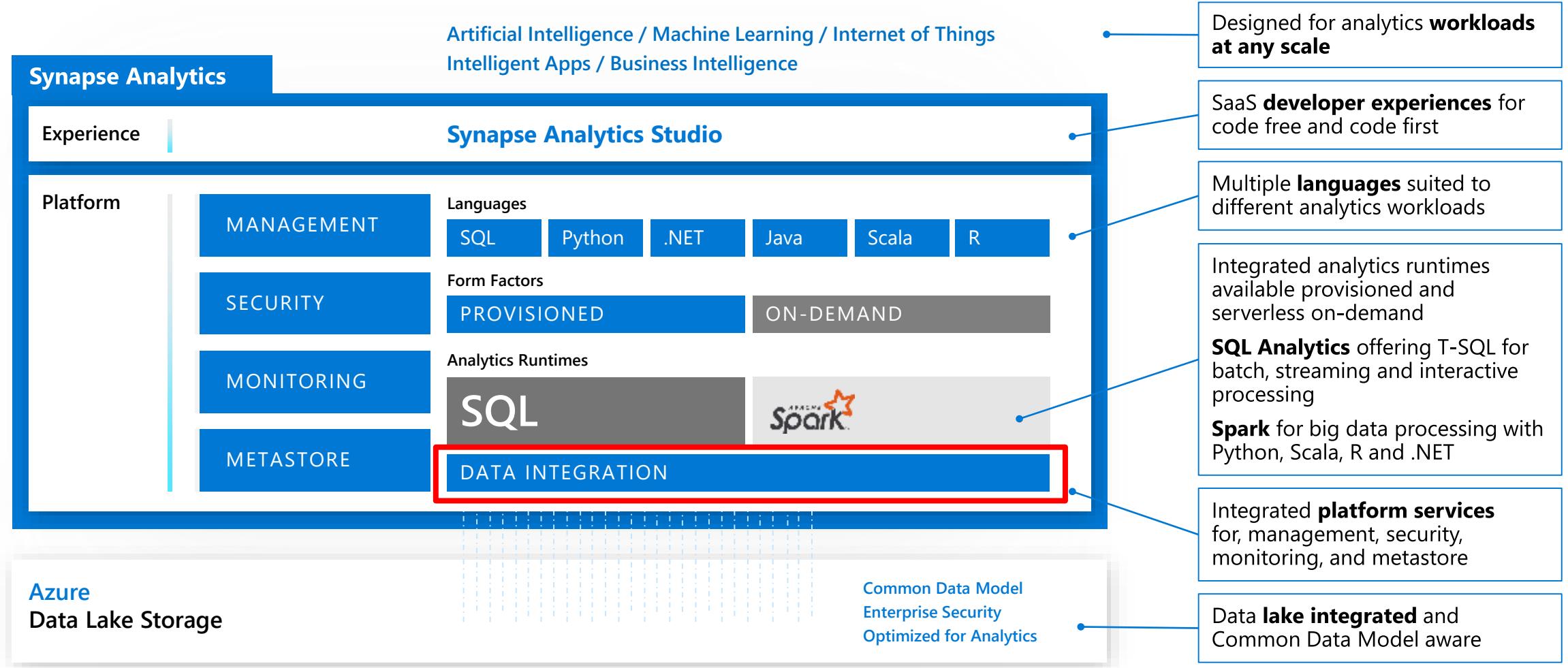


# Azure Synapse Analytics

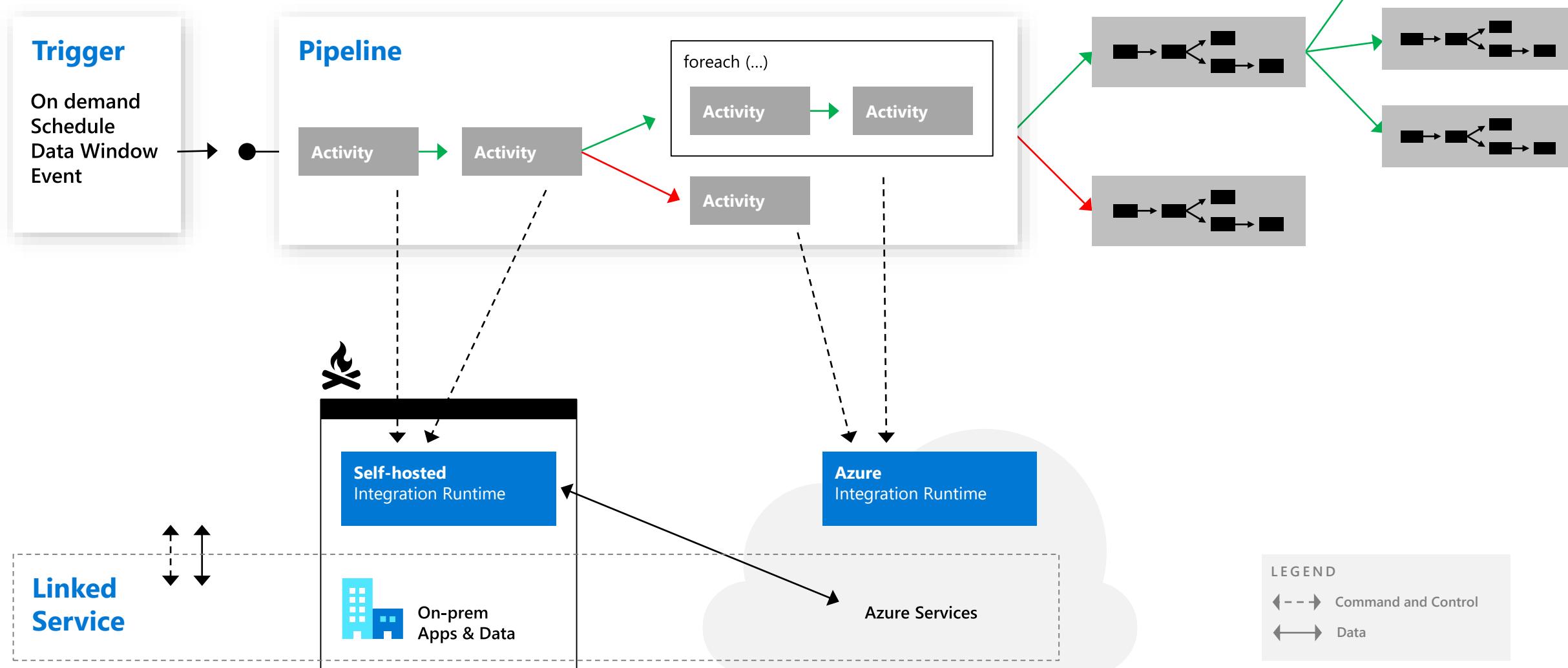
## Data Integration

# Azure Synapse Analytics

Limitless analytics service with unmatched time to insight



# Orchestration @ Scale



# Data Movement

## Scalable

per job elasticity

Up to 4 GB/s

## Simple

Visually author or via code (Python, .Net, etc.)

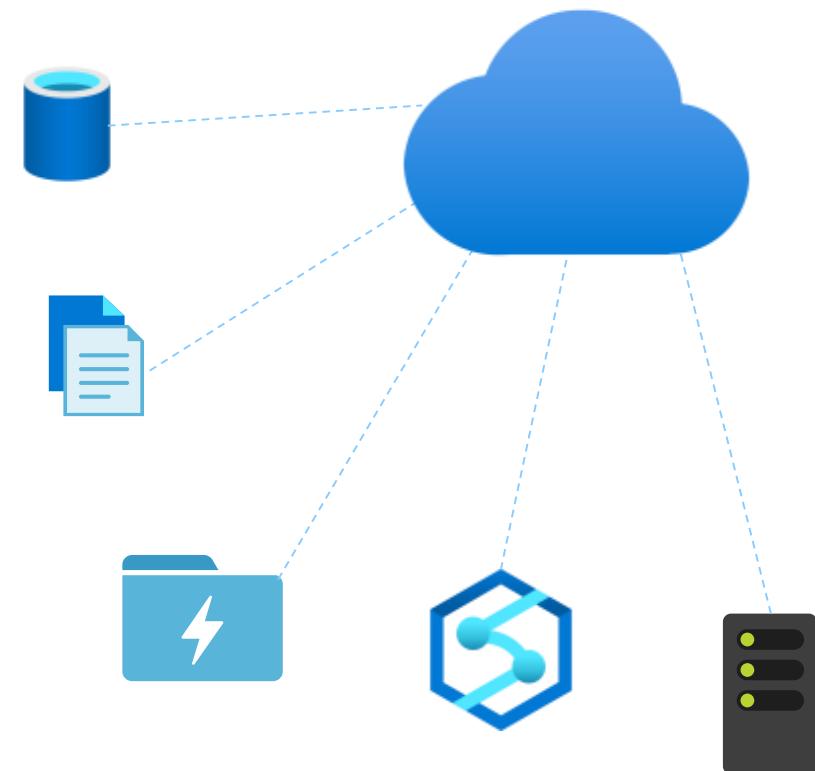
Serverless, no infrastructure to manage

## Access all your data

90+ connectors provided and growing (cloud, on premises, SaaS)

Data Movement as a Service: 25 points of presence worldwide

Self-hostable Integration Runtime for hybrid movement



# 90+ Connectors out of the box

# Pipelines

## Overview

It provides ability to load data from storage account to desired linked service. Load data by manual execution of pipeline or by orchestration

## Benefits

Supports common loading patterns

Fully parallel loading into data lake or SQL tables

Graphical development experience

The screenshot displays two views of the Azure Synapse Analytics Pipelines interface.

**Top View:** Shows the main Pipelines page with a search bar and a 'New pipeline' button. The sidebar includes links for Synapse, Data, Develop, Orchestrate, and Monitor.

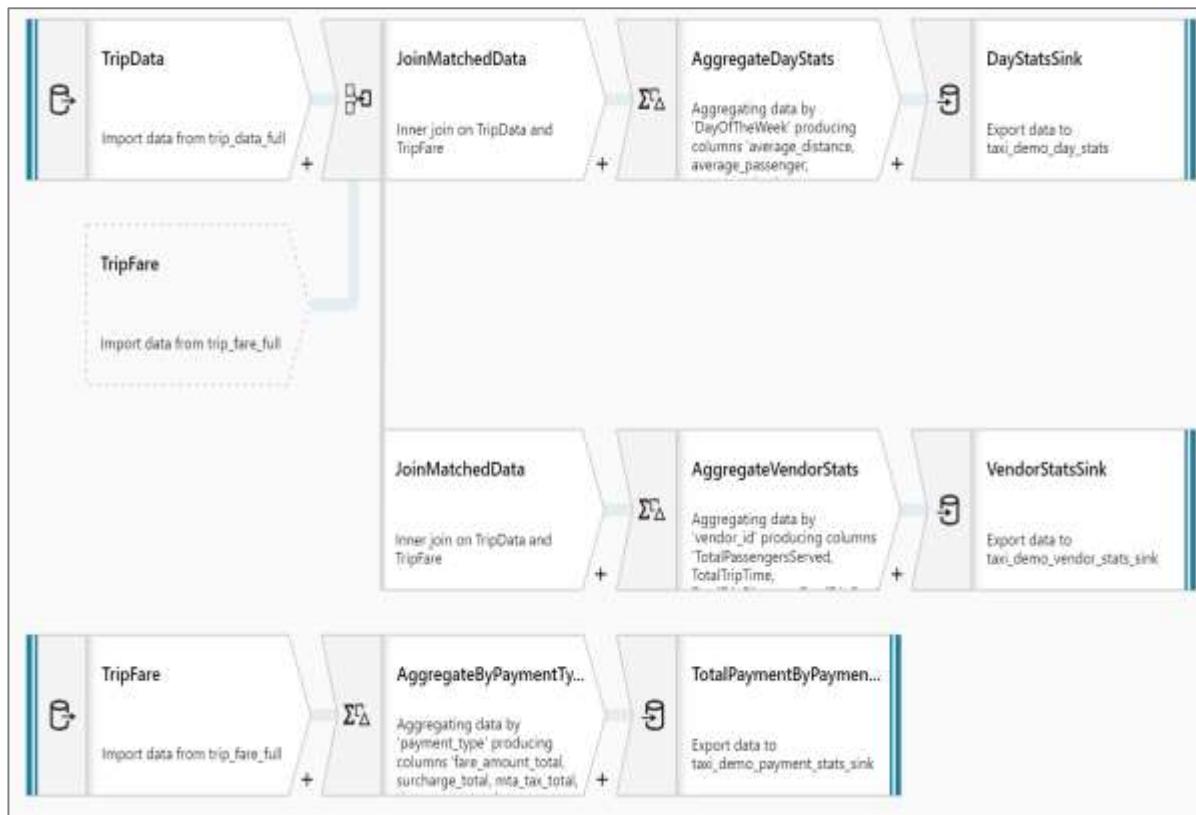
**Bottom View:** Shows a detailed view of a pipeline named 'Load Data to SQLDW'. The pipeline contains a single 'Copy data' activity named 'WeatherData'. The activity configuration pane shows the source dataset as 'ADLSGen2so' and the sink as 'Azure SQL Database Managed Instance'. A large grid of linked services is visible on the right, including:

Azure Cosmos DB (SQL API)	Azure Data Explorer (Kusto)	Azure Data Lake Storage Gen1
Azure Data Lake Storage Gen2	Azure Database for MariaDB	Azure Database for MySQL
Azure Database for PostgreSQL	Azure File Storage	Azure SQL Database
Azure SQL Database Managed Instance	Azure Synapse Analytics (formerly SQL DW)	Azure Table Storage

# Prep & Transform Data

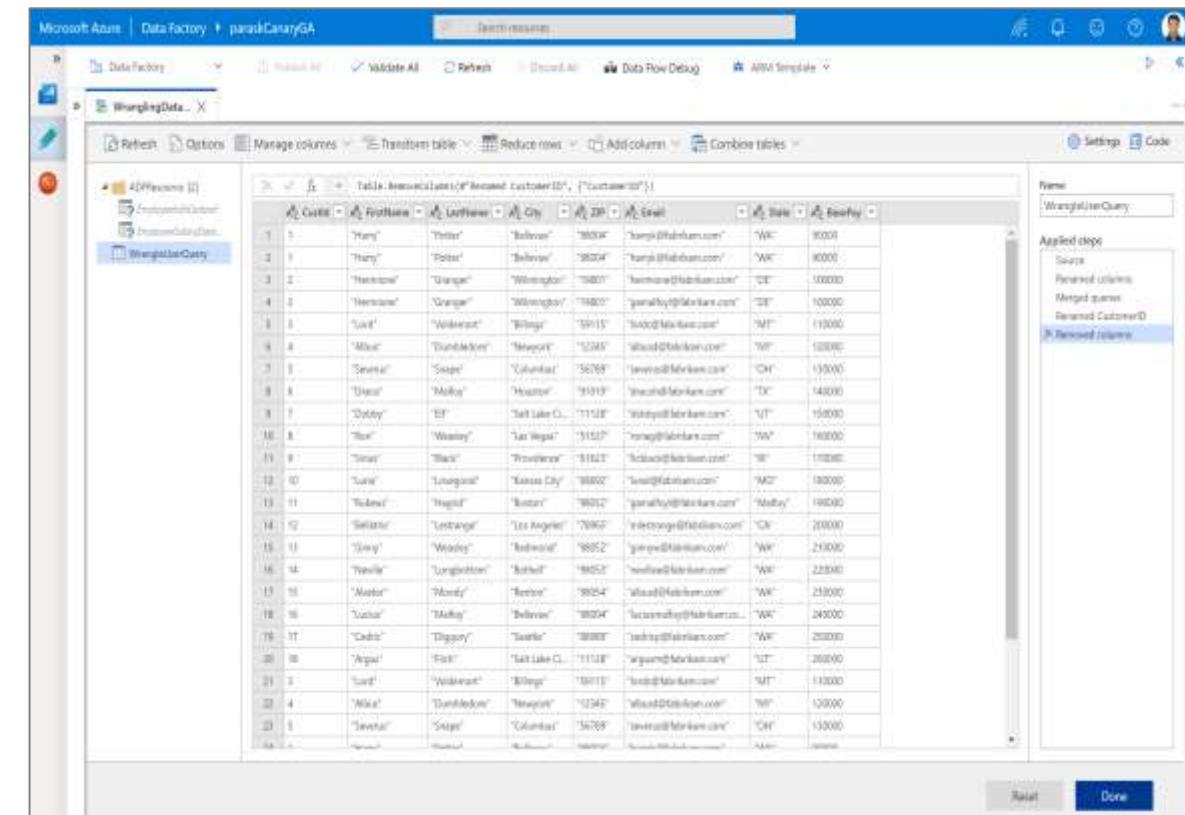
## Mapping Dataflow

Code free data transformation @scale



## Wrangling Dataflow

Code free data preparation @scale



# Triggers

## Overview

Triggers represent a unit of processing that determines when a pipeline execution needs to be kicked off.

Data Integration offers 3 trigger types as –

1. Schedule – gets fired at a schedule with information of start date, recurrence, end date
2. Event – gets fired on specified event
3. Tumbling window – gets fired at a periodic time interval from a specified start date, while retaining state

It also provides ability to monitor pipeline runs and control trigger execution.

The screenshot shows the Azure Synapse Analytics Data Integration Orchestration interface. On the left, there is a navigation sidebar with icons for External connections, Linked services, Orchestration, Triggers (which is selected and highlighted in blue), Integration runtimes, Security, and Access control. Above the sidebar, there are buttons for Publish all (with a count of 4), Validate all, Refresh, and Discard all. To the right of the sidebar, the main area is titled 'Triggers' with the sub-instruction: 'To execute a pipeline set the trigger. Triggers represent a unit of processing that determines when a pipeline runs.' Below this, there is a table with columns for NAME, TYPE, and STATUS. Two triggers are listed: one named 'CopyParquetDataTrigger' of type 'Schedule' and status 'Started', and another named 'Trigger 1' of type 'Schedule' and status 'Stopped'. At the bottom of the main area, there is a 'New' button. A large modal dialog box is open over the main content, titled 'New trigger'. It contains fields for 'Name' (set to 'Trigger 1'), 'Description' (empty), 'Type' (set to 'Schedule'), 'Start Date (UTC)' (set to '10/30/2019 11:20 PM'), 'Recurrence' (set to 'Every 1 Minute(s)'), 'End' (set to 'No End'), and an 'Annotations' section with a '+ New' button. There is also an 'Activated' section with radio buttons for 'Yes' (selected) and 'No'. At the bottom of the modal is an 'OK' button.

NAME	TYPE	STATUS
* CopyParquetDataTrigger	Schedule	Started
* Trigger 1	Schedule	Stopped

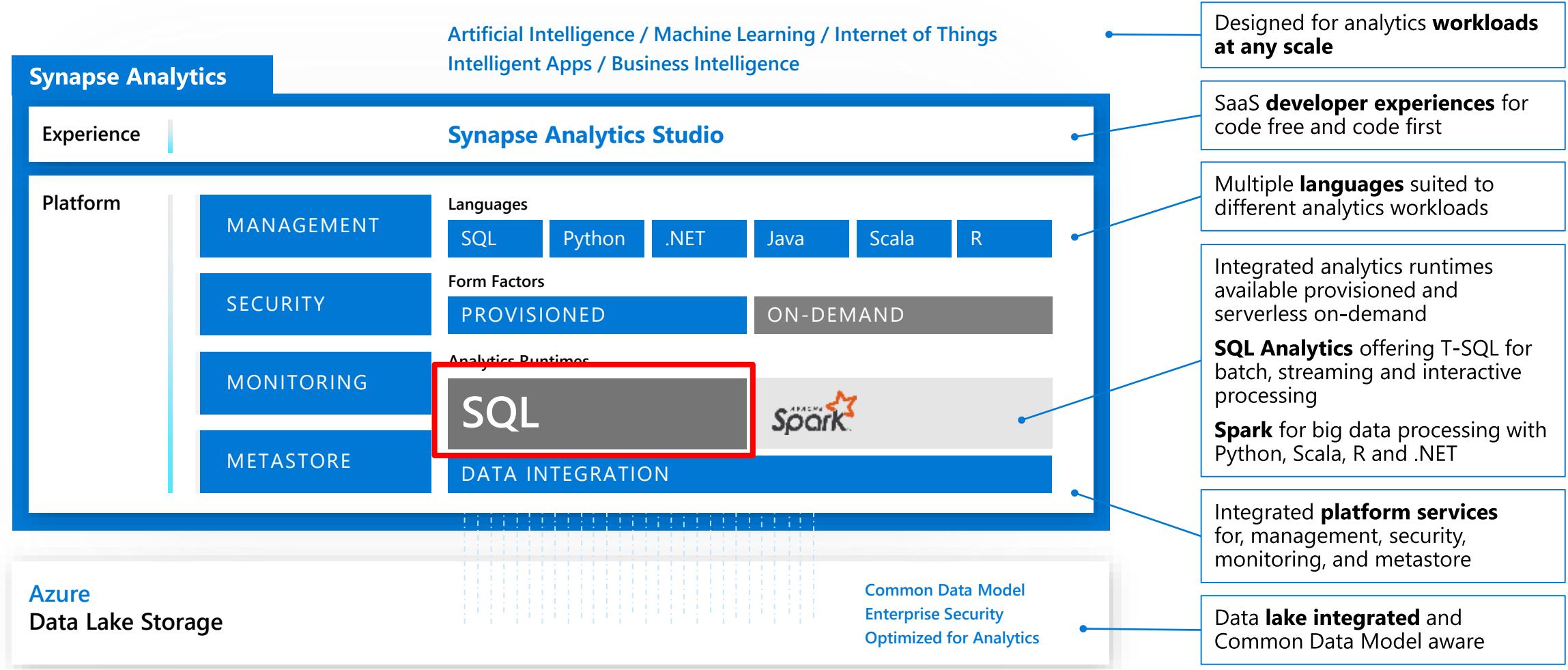


# Azure Synapse Analytics

## SQL Analytics

# Azure Synapse Analytics

Limitless analytics service with unmatched time to insight



# Comprehensive SQL functionality



## Advanced storage system

- Columnstore Indexes
- Table partitions
- Distributed tables
- Isolation modes
- Materialized Views
- Nonclustered Indexes
- Result-set caching

## T-SQL Querying

- Windowing aggregates
- Approximate execution (Hyperloglog)
- JSON data support

## Complete SQL object model

- Tables
- Views
- Stored procedures
- Functions

# Windowing functions

## OVER clause

Defines a window or specified set of rows within a query result set

Computes a value for each row in the window

## Aggregate functions

COUNT, MAX, AVG, SUM, APPROX\_COUNT\_DISTINCT, MIN, STDEV, STDEVP, STRING\_AGG, VAR, VARP, GROUPING, GROUPING\_ID, COUNT\_BIG, CHECKSUM\_AGG

## Ranking functions

RANK, NTILE, DENSE\_RANK, ROW\_NUMBER

## Analytical functions

LAG, LEAD, FIRST\_VALUE, LAST\_VALUE, CUME\_DIST, PERCENTILE\_CONT, PERCENTILE\_DISC, PERCENT\_RANK

## ROWS | RANGE

PRECEDING, UNBOUNDED PRECEDING, CURRENT ROW, BETWEEN, FOLLOWING, UNBOUNDED FOLLOWING

`SELECT`

```
ROW_NUMBER() OVER(PARTITION BY PostalCode ORDER BY SalesYTD DESC) AS "Row Number",
LastName,
SalesYTD,
PostalCode
FROM Sales
WHERE SalesYTD <> 0
ORDER BY PostalCode;
```

Row Number	LastName	SalesYTD	PostalCode
1	Mitchell	4251368.5497	98027
2	Blythe	3763178.1787	98027
3	Carson	3189418.3662	98027
4	Reiter	2315185.611	98027
5	Vargas	1453719.4653	98027
6	Ansman-Wolfe	1352577.1325	98027
1	Pak	4116870.2277	98055
2	Varkey Chudukaktil	3121616.3202	98055
3	Saraiva	2604540.7172	98055
4	Ito	2458535.6169	98055
5	Valdez	1827066.7118	98055
6	Mensa-Annan	1576562.1966	98055
7	Campbell	1573012.9383	98055
8	Tsoflias	1421810.9242	98055

# Windowing Functions (continued)

## Analytical functions

LAG, LEAD, FIRST\_VALUE, LAST\_VALUE, CUME\_DIST, PERCENTILE\_CONT, PERCENTILE\_DISC, PERCENT\_RANK

-- PERCENTILE\_CONT, PERCENTILE\_DISC

```
SELECT DISTINCT Name AS DepartmentName
,PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY ph.Rate)
    OVER (PARTITION BY Name) AS MedianCont
,PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY ph.Rate)
    OVER (PARTITION BY Name) AS MedianDisc
FROM HumanResources.Department AS d
INNER JOIN HumanResources.EmployeeDepartmentHistory AS dh
    ON dh.DepartmentID = d.DepartmentID
INNER JOIN HumanResources.EmployeePayHistory AS ph
    ON ph.BusinessEntityID = dh.BusinessEntityID
WHERE dh.EndDate IS NULL;
```

DepartmentName	MedianCont	MedianDisc
Document Control	16.8269	16.8269
Engineering	34.375	32.6923
Executive	54.32695	48.5577
Human Resources	17.427850	16.5865

--LAG Function

```
SELECT BusinessEntityID,
YEAR(QuotaDate) AS SalesYear,
SalesQuota AS CurrentQuota,
LAG(SalesQuota, 1,0) OVER (ORDER BY YEAR(QuotaDate)) AS PreviousQuota
FROM Sales.SalesPersonQuotaHistory
WHERE BusinessEntityID = 275 and YEAR(QuotaDate) IN ('2005','2006');
```

BusinessEntityID	SalesYear	CurrentQuota	PreviousQuota
275	2005	367000.00	0.00
275	2005	556000.00	367000.00
275	2006	502000.00	556000.00
275	2006	550000.00	502000.00
275	2006	1429000.00	550000.00
275	2006	1324000.00	1429000.00

# Windowing Functions (continued)

## ROWS | RANGE

PRECEDING, UNBOUNDING PRECEDING, CURRENT ROW, BETWEEN, FOLLOWING, UNBOUNDED FOLLOWING

```
-- First_Value  
  
SELECT JobTitle, LastName, VacationHours AS VacHours,  
FIRST_VALUE(LastName) OVER (PARTITION BY JobTitle  
ORDER BY VacationHours ASC ROWS UNBOUNDED PRECEDING ) AS FewestVacHours  
FROM HumanResources.Employee AS e  
INNER JOIN Person.Person AS p  
ON e.BusinessEntityID = p.BusinessEntityID  
ORDER BY JobTitle;
```



JobTitle	LastName	VacHours	FewestVacHours
Accountant	Moreland	58	Moreland
Accountant	Seamans	59	Moreland
Accounts Manager	Liu	57	Liu
Accounts Payable Specialist	Tomic	63	Tomic
Accounts Payable Specialist	Sheperdigian	64	Tomic
Accounts Receivable Specialist	Poe	60	Poe
Accounts Receivable Specialist	Spoon	61	Poe
Accounts Receivable Specialist	Walton	62	Poe

# Approximate execution

## HyperLogLog accuracy

Will return a result with a 2% accuracy of true cardinality on average.

e.g. COUNT (DISTINCT) returns 1,000,000, HyperLogLog will return a value in the range of 999,736 to 1,016,234.

## APPROX\_COUNT\_DISTINCT

Returns the approximate number of unique non-null values in a group.

## Use Case: Approximating web usage trend behavior

-- Syntax

```
APPROX_COUNT_DISTINCT( expression )
```

-- The approximate number of different order keys by order status from the orders table.

```
SELECT O_OrderStatus, APPROX_COUNT_DISTINCT(O_OrderKey) AS Approx_Distinct_OrderKey  
FROM dbo.Orders  
GROUP BY O_OrderStatus  
ORDER BY O_OrderStatus;
```

# Approximate execution

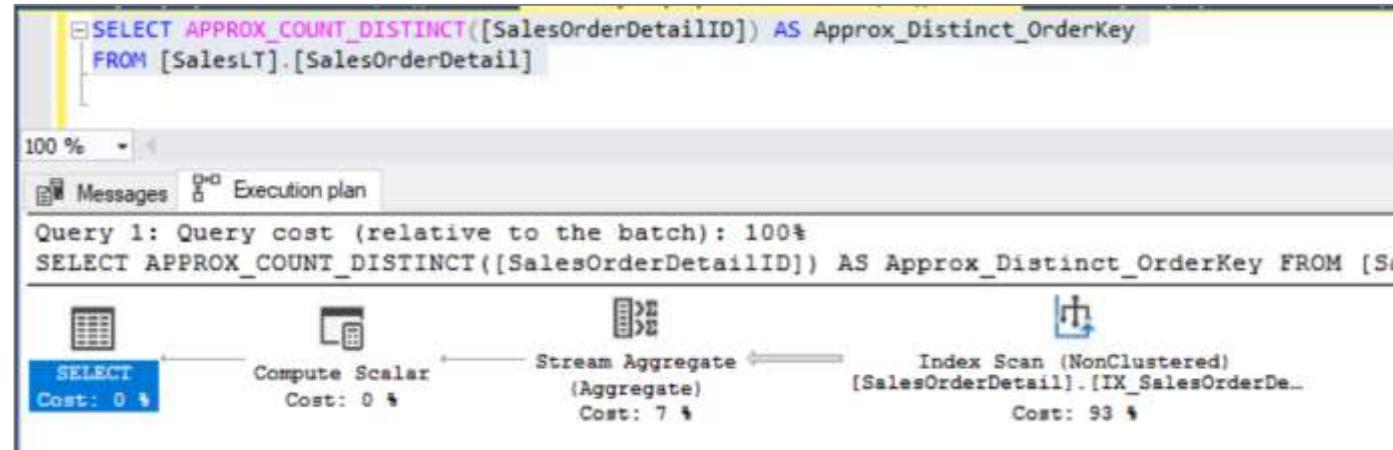
## APPROX\_COUNT\_DISTINCT

```
SELECT APPROX_COUNT_DISTINCT([SalesOrderDetailID]) AS Approx_Distinct_OrderKey
FROM [SalesLT].[SalesOrderDetail]
```

100 %

Results Messages

Approx_Distinct_OrderKey
540



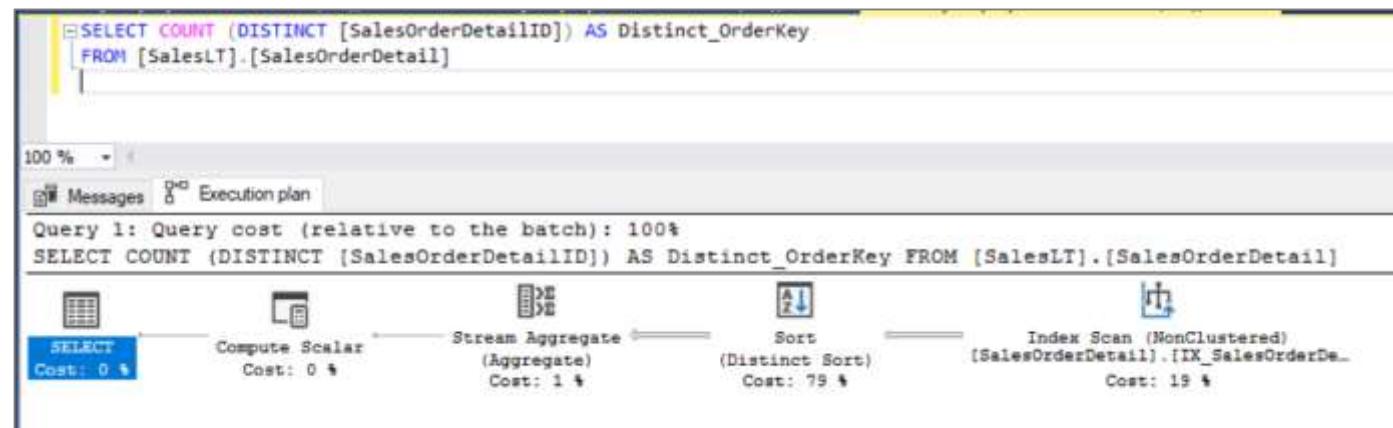
## COUNT DISTINCT

```
SELECT COUNT (DISTINCT [SalesOrderDetailID]) AS Distinct_OrderKey
FROM [SalesLT].[SalesOrderDetail]
```

100 %

Results Messages

Distinct_OrderKey
542



# Group by options

## Group by with rollup

Creates a group for each combination of column expressions.

Rolls up the results into subtotals and grand totals

Calculate the aggregates of hierarchical data

-- GROUP BY ROLLUP Example --

```
SELECT Country,
Region,
SUM(Sales) AS TotalSales
FROM Sales
GROUP BY ROLLUP (Country, Region);
```

-- Results --

## Grouping sets

Combine multiple GROUP BY clauses into one GROUP BY CLAUSE.

Equivalent of UNION ALL of specified groups.

-- GROUP BY SETS Example --

```
SELECT Country,
SUM(Sales) AS TotalSales
FROM Sales
GROUP BY GROUPING SETS ( Country, () );
```



Country	Region	TotalSales
Canada	Alberta	100
Canada	British Columbia	500
Canada	NULL	600
United States	Montana	100
United States	NULL	100
NULL	NULL	700

# Snapshot isolation

## Overview

Specifies that statements cannot read data that has been modified but not committed by other transactions.

This prevents dirty reads.

```
ALTER DATABASE MyDatabase  
SET ALLOW_SNAPSHOT_ISOLATION ON
```

```
ALTER DATABASE MyDatabase SET  
READ_COMMITTED_SNAPSHOT ON
```

## Isolation level

- READ COMMITTED
- REPEATABLE READ
- SNAPSHOT
- READ UNCOMMITTED
- SERIALIZABLE

## **READ\_COMMITTED\_SNAPSHOT**

**OFF** (Default) – Uses shared locks to prevent other transactions from modifying rows while running a read operation

**ON** – Uses row versioning to present each statement with a transactionally consistent snapshot of the data as it existed at the start of the statement. Locks are not used to protect the data from updates.

# JSON data support – insert JSON data

## Overview

The JSON format enables representation of complex or hierarchical data structures in tables.

JSON data is stored using standard NVARCHAR table columns.

## Benefits

Transform arrays of JSON objects into table format

Performance optimization using clustered columnstore indexes and memory optimized tables

```
-- Create Table with column for JSON string
CREATE TABLE CustomerOrders
(
    CustomerId BIGINT NOT NULL,
    Country NVARCHAR(150) NOT NULL,
    OrderDetails NVARCHAR(3000) NOT NULL -- NVARCHAR column for JSON
) WITH (DISTRIBUTION = ROUND_ROBIN)

-- Populate table with semi-structured data
INSERT INTO CustomerOrders
VALUES
( 101, -- CustomerId
'Bahrain', -- Country
N'[{ "StoreId": "AW73565",
        "Order": { "Number": "SO43659",
                  "Date": "2011-05-31T00:00:00"
                },
        "Item": { "Price": 2024.40, "Quantity": 1 }
      }]' -- OrderDetails
)
```

# JSON data support – read JSON data

## Overview

Read JSON data stored in a string column with the following:

- **ISJSON** – verify if text is valid JSON
- **JSON\_VALUE** – extract a scalar value from a JSON string
- **JSON\_QUERY** – extract a JSON object or array from a JSON string

## Benefits

Ability to get standard columns as well as JSON column

Perform aggregation and filter on JSON values

-- Return all rows with valid JSON data

```
SELECT CustomerId, OrderDetails
FROM CustomerOrders
WHERE ISJSON(OrderDetails) > 0;
```

CustomerId	OrderDetails
101	N'[{ StoreId": "AW73565", "Order": { "Number": "SO43659", "Date": "2011-05-31T00:00:00" }, "Item": { "Price": 2024.40, "Quantity": 1 }}]'

-- Extract values from JSON string

```
SELECT CustomerId,
Country,
JSON_VALUE(OrderDetails,'$.StoreId') AS StoreId,
JSON_QUERY(OrderDetails,'$.Item') AS ItemDetails
FROM CustomerOrders;
```

CustomerId	Country	StoreId	ItemDetails
101	Bahrain	AW73565	{ "Price": 2024.40, "Quantity": 1 }

# JSON data support – modify and operate on JSON data

## Overview

Use standard table columns and values from JSON text in the same analytical query.

Modify JSON data with the following:

- **JSON\_MODIFY** – modifies a value in a JSON string
- **OPENJSON** – convert JSON collection to a set of rows and columns

## Benefits

Flexibility to update JSON string using T-SQL

Convert hierarchical data into flat tabular structure

```
-- Modify Item Quantity value
UPDATE CustomerOrders SET OrderDetails =
JSON_MODIFY(OrderDetails, '$.OrderDetails.Item.Quantity', 2)
```

### OrderDetails

```
N'[{ StoreId: "AW73565", "Order": { "Number": "SO43659",
"Date": "2011-05-31T00:00:00" }, "Item": { "Price": 2024.40, "Quantity": 2 }}]'
```

```
-- Convert JSON collection to rows and columns
SELECT CustomerId,
StoreId,
OrderDetails.OrderDate,
OrderDetails.OrderPrice
FROM CustomerOrders
CROSS APPLY OPENJSON (CustomerOrders.OrderDetails)
WITH ( StoreId    VARCHAR(50) '$.StoreId',
OrderNumber  VARCHAR(100) '$.Order.Date',
OrderDate    DATETIME   '$.Order.Date',
OrderPrice   DECIMAL    '$.Item.Price',
OrderQuantity INT       '$.Item.Quantity'
) AS OrderDetails
```

CustomerId	StoreId	OrderDate	OrderPrice
101	AW73565	2011-05-31T00:00:00	2024.40

# Stored Procedures

## Overview

It is a group of one or more SQL statements or a reference to a Microsoft .NET Framework common language runtime (CLR) method.

Promotes flexibility and modularity.

Supports parameters and nesting.

## Benefits

Reduced server/client network traffic, improved performance

Stronger security

Easy maintenance

```
CREATE PROCEDURE HumanResources.uspGetAllEmployees
AS
    SET NOCOUNT ON;
    SELECT LastName, FirstName, JobTitle, Department
    FROM HumanResources.vEmployeeDepartment;
GO

-- Execute a stored procedures
EXECUTE HumanResources.uspGetAllEmployees;
GO
-- Or
EXEC HumanResources.uspGetAllEmployees;
GO
-- Or, if this procedure is the first statement within a batch:
HumanResources.uspGetAllEmployees;
```

# Tables – Indexes

## Clustered Columnstore index (Default Primary)

Highest level of data compression

Best overall query performance

## Clustered index (Primary)

Performant for looking up a single to few rows

## Heap (Primary)

Faster loading and landing temporary data

Best for small lookup tables

## Nonclustered indexes (Secondary)

Enable ordering of multiple columns in a table

Allows multiple nonclustered on a single table

Can be created on any of the above primary indexes

More performant lookup queries

-- Create table with index

```
CREATE TABLE orderTable
```

```
(
```

```
    OrderId INT NOT NULL,
```

```
    Date DATE NOT NULL,
```

```
    Name VARCHAR(2),
```

```
    Country VARCHAR(2)
```

```
)
```

```
WITH
```

```
(
```

```
    CLUSTERED COLUMNSTORE INDEX |
```

```
    HEAP |
```

```
    CLUSTERED INDEX (OrderId)
```

```
);
```

-- Add non-clustered index to table

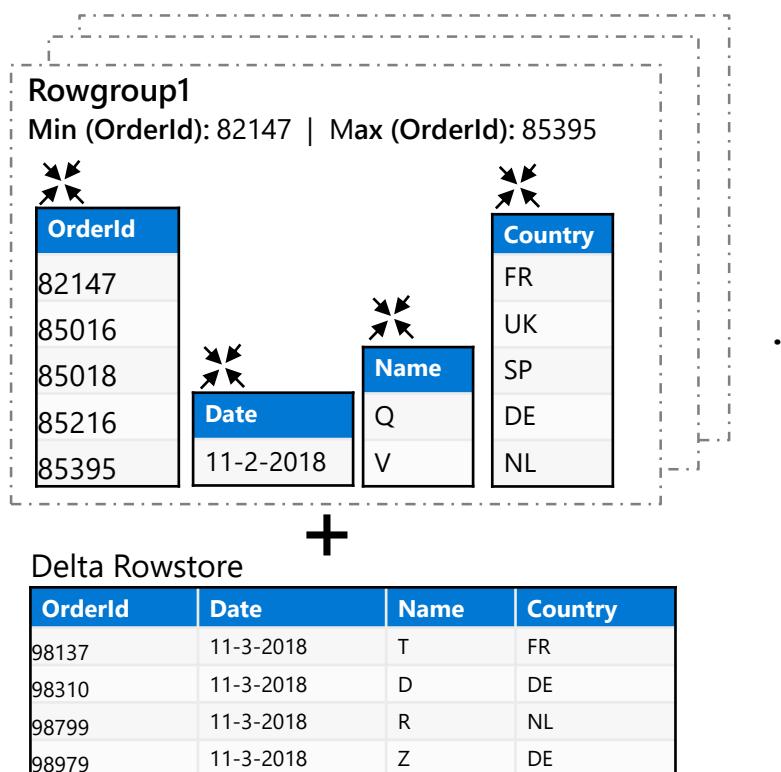
```
CREATE INDEX NameIndex ON orderTable (Name);
```

# SQL Analytics Columnstore Tables

## Logical table structure

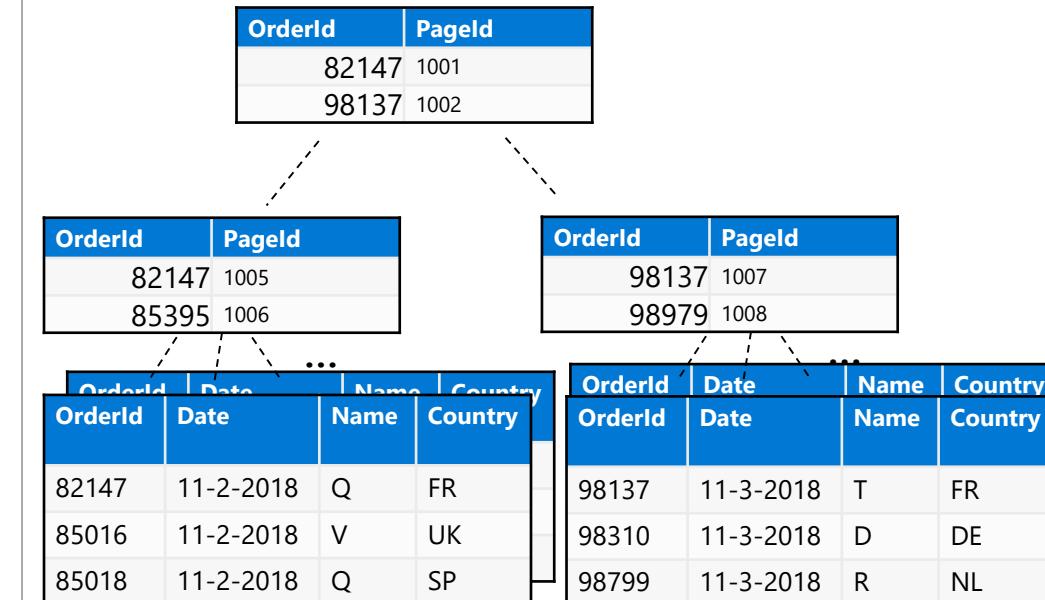
OrderId	Date	Name	Country
85016	11-2-2018	V	UK
85018	11-2-2018	Q	SP
85216	11-2-2018	Q	DE
85395	11-2-2018	V	NL
82147	11-2-2018	Q	FR
86881	11-2-2018	D	UK
93080	11-3-2018	R	UK
94156	11-3-2018	S	FR
96250	11-3-2018	Q	NL
98799	11-3-2018	R	NL
98015	11-3-2018	T	UK
98310	11-3-2018	D	DE
98979	11-3-2018	Z	DE
98137	11-3-2018	T	FR
...	...	...	...

## Clustered columnstore index (OrderId)



- Data stored in compressed columnstore segments after being sliced into groups of rows (rowgroups/micro-partitions) for maximum compression
- Rows are stored in the delta rowstore until the number of rows is large enough to be compressed into a columnstore

## Clustered/Non-clustered rowstore index (OrderId)



- Data is stored in a B-tree index structure for performant lookup queries for particular rows.
- Clustered rowstore index: The leaf nodes in the structure store the data values in a row (as pictured above)
- Non-clustered (secondary) rowstore index: The leaf nodes store pointers to the data values, not the values themselves

# Ordered Clustered Columnstore Indexes

## Overview

Queries against tables with ordered columnstore segments can take advantage of improved segment elimination to drastically reduce the time needed to service a query.

### -- Create Table with Ordered Columnstore Index

```
CREATE TABLE sortedOrderTable
```

```
(  
    OrderId INT NOT NULL,  
    Date DATE NOT NULL,  
    Name VARCHAR(2),  
    Country VARCHAR(2)
```

```
)  
WITH  
(  
    CLUSTERED COLUMNSTORE INDEX ORDER (OrderId)
```

### -- Create Clustered Columnstore Index on existing table

```
CREATE CLUSTERED COLUMNSTORE INDEX cciOrderId  
ON dbo.OrderTable ORDER (OrderId)
```

### -- Insert data into table with ordered columnstore index

```
INSERT INTO sortedOrderTable
```

```
VALUES (1, '01-01-2019','Dave', 'UK')
```

# Tables – Distributions

## Round-robin distributed

Distributes table rows evenly across all distributions at random.

## Hash distributed

Distributes table rows across the Compute nodes by using a deterministic hash function to assign each row to one distribution.

## Replicated

Full copy of table accessible on each Compute node.

```
CREATE TABLE dbo.OrderTable
(
    OrderId INT NOT NULL,
    Date DATE NOT NULL,
    Name VARCHAR(2),
    Country VARCHAR(2)
)
WITH
(
    CLUSTERED COLUMNSTORE INDEX,
    DISTRIBUTION = HASH([OrderId]) |
        ROUND ROBIN |
        REPLICATED
);
```

# Tables – Partitions

## Overview

Table partitions divide data into smaller groups

In most cases, partitions are created on a date column

Supported on all table types

RANGE RIGHT – Used for time partitions

RANGE LEFT – Used for number partitions

## Benefits

Improves efficiency and performance of loading and querying by limiting the scope to subset of data.

Offers significant query performance enhancements where filtering on the partition key can eliminate unnecessary scans and eliminate IO.

```
CREATE TABLE partitionedOrderTable
(
    OrderId INT NOT NULL,
    Date DATE NOT NULL,
    Name VARCHAR(2),
    Country VARCHAR(2)
)
WITH
(
    CLUSTERED COLUMNSTORE INDEX,
    DISTRIBUTION = HASH([OrderId]),
    PARTITION (
        [Date] RANGE RIGHT FOR VALUES (
            '2000-01-01', '2001-01-01', '2002-01-01',
            '2003-01-01', '2004-01-01', '2005-01-01'
        )
    )
);
```

# Tables – Distributions & Partitions

## Logical table structure

OrderId	Date	Name	Country
85016	11-2-2018	V	UK
85018	11-2-2018	Q	SP
85216	11-2-2018	Q	DE
85395	11-2-2018	V	NL
82147	11-2-2018	Q	FR
86881	11-2-2018	D	UK
93080	11-3-2018	R	UK
94156	11-3-2018	S	FR
96250	11-3-2018	Q	NL
98799	11-3-2018	R	NL
98015	11-3-2018	T	UK
98310	11-3-2018	D	DE
98979	11-3-2018	Z	DE
98137	11-3-2018	T	FR
...	...	...	...

## Physical data distribution

( Hash distribution (OrderId), Date partitions )

### Distribution1

(OrderId 80,000 – 100,000)

#### 11-2-2018 partition

OrderId	Date	Name	Country
85016	11-2-2018	V	UK
85018	11-2-2018	Q	SP
85216	11-2-2018	Q	DE
85395	11-2-2018	V	NL
82147	11-2-2018	Q	FR
86881	11-2-2018	D	UK
...	...	...	...

#### 11-3-2018 partition

OrderId	Date	Name	Country
93080	11-3-2018	R	UK
94156	11-3-2018	S	FR
96250	11-3-2018	Q	NL
98799	11-3-2018	R	NL
98015	11-3-2018	T	UK
98310	11-3-2018	D	DE
98979	11-3-2018	Z	DE
98137	11-3-2018	T	FR
...	...	...	...

...

x 60 distributions (shards)

- Each shard is partitioned with the same date partitions
- A minimum of 1 million rows per distribution and partition is needed for optimal compression and performance of clustered Columnstore tables

# Common table distribution methods

Table Category	Recommended Distribution Option
Fact	<p>Use hash-distribution with clustered columnstore index. Performance improves because hashing enables the platform to localize certain operations within the node itself during query execution.</p> <p>Operations that benefit:</p> <p>COUNT(DISTINCT( &lt;hashed_key&gt; )) OVER PARTITION BY &lt;hashed_key&gt; most JOIN &lt;table_name&gt; ON &lt;hashed_key&gt; GROUP BY &lt;hashed_key&gt;</p>
Dimension	Use replicated for smaller tables. If tables are too large to store on each Compute node, use hash-distributed.
Staging	Use round-robin for the staging table. The load with CTAS is faster. Once the data is in the staging table, use INSERT...SELECT to move the data to production tables.

# Materialized views

## Overview

A materialized view pre-computes, stores, and maintains its data like a table.

Materialized views are automatically updated when data in underlying tables are changed. This is a synchronous operation that occurs as soon as the data is changed.

The auto caching functionality allows Azure Synapse Analytics Query Optimizer to consider using indexed view even if the view is not referenced in the query.

Supported aggregations: MAX, MIN, AVG, COUNT, COUNT\_BIG, SUM, VAR, STDEV

## Benefits

Automatic and synchronous data refresh with data changes in base tables. No user action is required.

High availability and resiliency as regular tables

```
-- Create indexed view
CREATE MATERIALIZED VIEW Sales.vw_Orders
WITH
(
    DISTRIBUTION = ROUND_ROBIN |
    HASH(ProductID)
)
AS
    SELECT SUM(UnitPrice*OrderQty) AS Revenue,
        OrderDate,
        ProductID,
        COUNT_BIG(*) AS OrderCount
    FROM Sales.SalesOrderDetail
    GROUP BY OrderDate, ProductID;
GO

-- Disable index view and put it in suspended mode
ALTER INDEX ALL ON Sales.vw_Orders DISABLE;

-- Re-enable index view by rebuilding it
ALTER INDEX ALL ON Sales.vw_Orders REBUILD;
```

# Materialized views - example

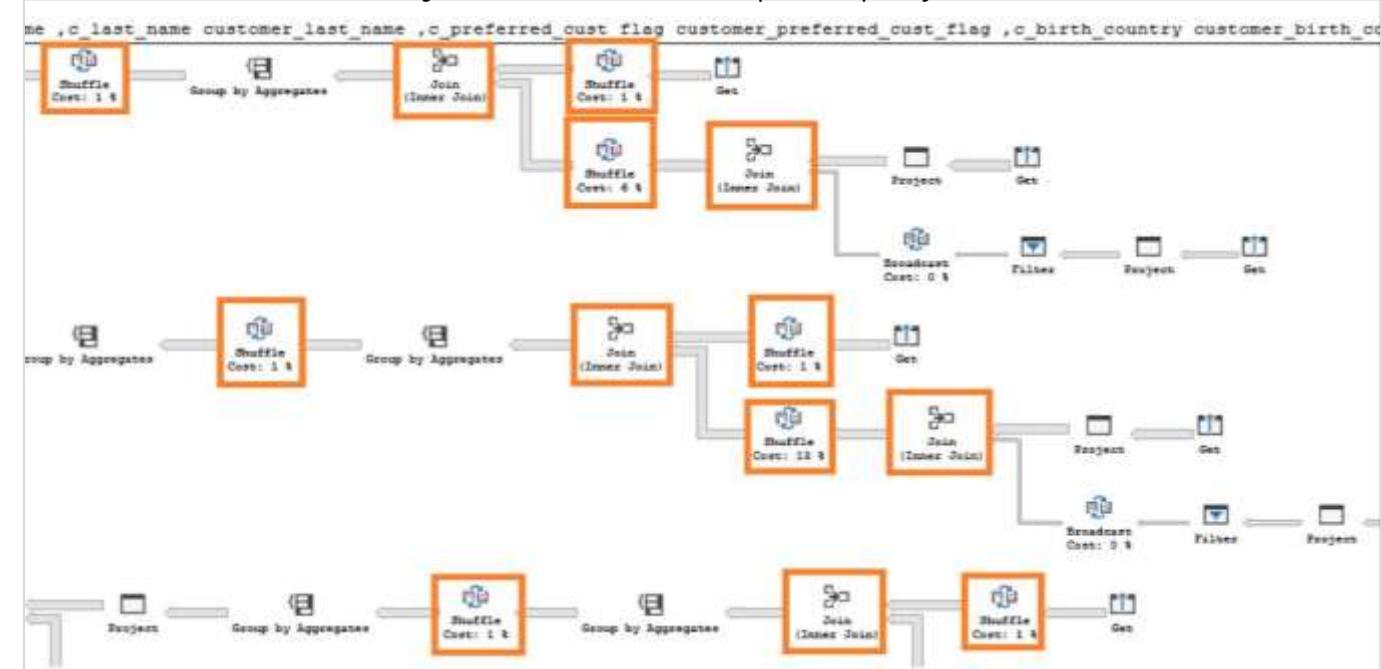
In this example, a query to get the year total sales per customer is shown to have a lot of data shuffles and joins that contribute to slow performance:

No relevant indexed views created on the data warehouse

```
-- Get year total sales per customer
(WITH year_total AS
    SELECT customer_id,
        first_name,
        last_name,
        birth_country,
        login,
        email_address,
        d_year,
        SUM(ISNULL(list_price - wholesale_cost -
        discount_amt + sales_price, 0)/2)year_total
    FROM customer cust
    JOIN catalog_sales sales ON cust.sk = sales.sk
    JOIN date_dim ON sales.sold_date = date_dim.date
    GROUP BY customer_id, first_name,
        last_name,birth_country,
        login,email_address ,d_year
)
SELECT TOP 100 ...
FROM year_total ...
WHERE ...
ORDER BY ...
```

**Execution time:** 103 seconds

Lots of data shuffles and joins needed to complete query



# Materialized views - example

Now, we add an indexed view to the data warehouse to increase the performance of the previous query. This view can be leveraged by the query even though it is not directly referenced.

Original query – get year total sales per customer

```
-- Get year total sales per customer
(WITH year_total AS
    SELECT customer_id,
           first_name,
           last_name,
           birth_country,
           login,
           email_address,
           d_year,
           SUM(ISNULL(list_price - wholesale_cost -
           discount_amt + sales_price, 0)/2)year_total
      FROM customer cust
     JOIN catalog_sales sales ON cust.sk = sales.sk
     JOIN date_dim ON sales.sold_date = date_dim.date
    GROUP BY customer_id, first_name,
             last_name,birth_country,
             login,email_address ,d_year
)
SELECT TOP 100 ...
   FROM year_total ...
  WHERE ...
 ORDER BY ...
```

Create indexed view with hash distribution on customer\_id column

```
-- Create indexed view for query
CREATE INDEXED VIEW nbViewCS WITH (DISTRIBUTION=HASH(customer_id)) AS
SELECT customer_id,
       first_name,
       last_name,
       birth_country,
       login,
       email_address,
       d_year,
       SUM(ISNULL(list_price - wholesale_cost - discount_amt +
       sales_price, 0)/2) AS year_total
  FROM customer cust
 JOIN catalog_sales sales ON cust.sk = sales.sk
 JOIN date_dim ON sales.sold_date = date_dim.date
 GROUP BY customer_id, first_name,
          last_name,birth_country,
          login, email_address, d_year
```

# Indexed (materialized) views - example

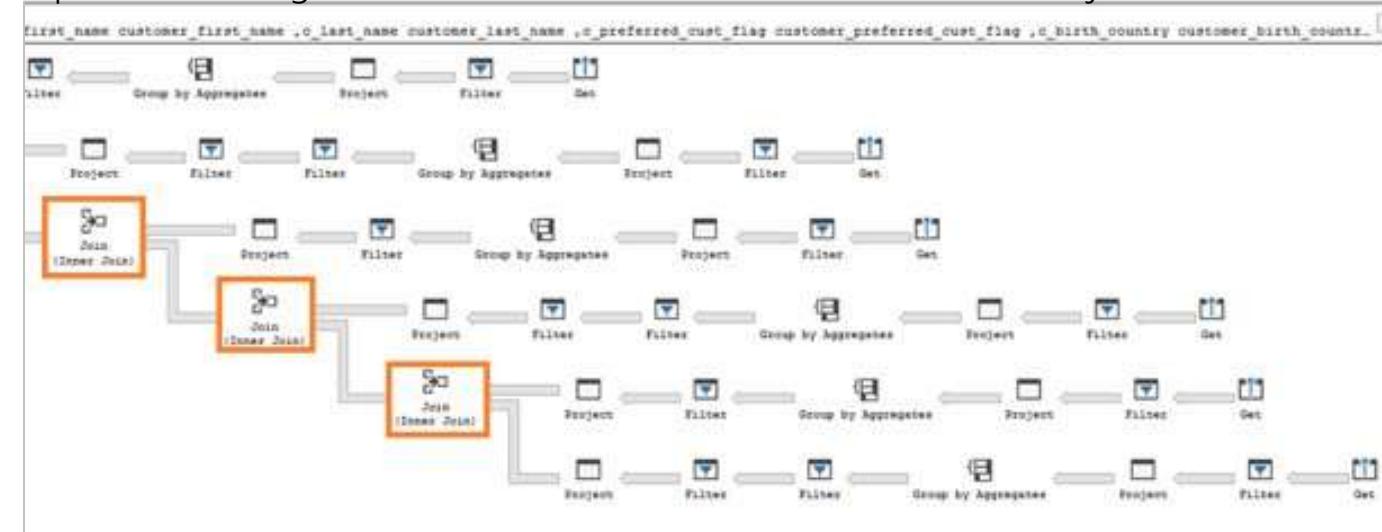
SQL pool query optimizer automatically leverages the indexed view to speed up the same query. Notice that the query does not need to reference the view directly

Original query – no changes have been made to query

```
-- Get year total sales per customer
(WITH year_total AS
    SELECT customer_id,
           first_name,
           last_name,
           birth_country,
           login,
           email_address,
           d_year,
           SUM(ISNULL(list_price - wholesale_cost -
           discount_amt + sales_price, 0)/2)year_total
      FROM customer cust
     JOIN catalog_sales sales ON cust.sk = sales.sk
     JOIN date_dim ON sales.sold_date = date_dim.date
   GROUP BY customer_id,first_name,
           last_name,birth_country,
           login,email_address ,d_year
)
SELECT TOP 100 ...
   FROM year_total ...
  WHERE ...
 ORDER BY ...
```

**Execution time:** 6 seconds

Optimizer leverages materialized view to reduce data shuffles and joins needed



# Materialized views- Recommendations

[EXPLAIN](#) - provides query plan for SQL statement without running the statement; view estimated cost of the query operations.

[EXPLAIN WITH\\_RECOMMENDATIONS](#) - provides query plan with recommendations to optimize the SQL statement performance.

```
EXPLAIN WITH_RECOMMENDATIONS
select count(*)
from (
    select distinct c_last_name, c_first_name, d_date
    from store_sales, date_dim, customer
    where store_sales.ss_sold_date_sk = date_dim.d_date_sk
    and store_sales.ss_customer_sk = customer.c_customer_sk
    and d_month_seq between 1194 and 1194+11)
except
    (select distinct c_last_name, c_first_name, d_date
    from catalog_sales, date_dim, customer
    where catalog_sales.cs_sold_date_sk = date_dim.d_date_sk
    and catalog_sales.cs_bill_customer_sk = customer.c_customer_sk and
    d_month_seq between 1194 and 1194+11)
) top_customers
```

# COPY command

## Overview

Copies data from source to destination

## Benefits

Retrieves data from all files from the folder and all its subfolders.

Supports multiple locations from the same storage account, separated by comma

Supports Azure Data Lake Storage (ADLS) Gen 2 and Azure Blob Storage.

Supports CSV, PARQUET, ORC file formats

```
COPY INTO test_1
FROM 'https://XXX.blob.core.windows.net/customerdatasets/test_1.txt'
WITH (
    FILE_TYPE = 'CSV',
    CREDENTIAL=(IDENTITY= 'Shared Access Signature',
    SECRET='<Your_SAS_Token>'),
    FIELDQUOTE = """",
    FIELDTERMINATOR=';',
    ROWTERMINATOR='0XA',
    ENCODING = 'UTF8',
    DATEFORMAT = 'ymd',
    MAXERRORS = 10,
    ERRORFILE = '/errorsfolder/'--path starting from the storage container,
    IDENTITY_INSERT
)
```

```
COPY INTO test_parquet
FROM 'https://XXX.blob.core.windows.net/customerdatasets/test.parquet'
WITH (
    FILE_FORMAT = myFileFormat
    CREDENTIAL=(IDENTITY= 'Shared Access Signature',
    SECRET='<Your_SAS_Token>')
)
```

# Result-set caching

## Overview

Cache the results of a query in SQL pool storage. This enables interactive response times for repetitive queries against tables with infrequent data changes.

The result-set cache persists even if SQL pool is paused and resumed later.

Query cache is invalidated and refreshed when underlying table data or query code changes.

Result cache is evicted regularly based on a time-aware least recently used algorithm (TLRU).

## Benefits

Enhances performance when same result is requested repetitively

Reduced load on server for repeated queries

Offers monitoring of query execution with a result cache hit or miss

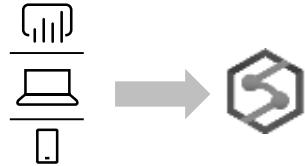
```
-- Turn on/off result-set caching for a database  
-- Must be run on the MASTER database  
ALTER DATABASE {database_name}  
SET RESULT_SET_CACHING { ON | OFF }
```

```
-- Turn on/off result-set caching for a client session  
-- Run on target Azure Synapse Analytics  
SET RESULT_SET_CACHING {ON | OFF}
```

```
-- Check result-set caching setting for a database  
-- Run on target Azure Synapse Analytics  
SELECT is_result_set_caching_on  
FROM sys.databases  
WHERE name = {database_name}
```

```
-- Return all query requests with cache hits  
-- Run on target data warehouse  
SELECT *  
FROM sys.dm_pdw_request_steps  
WHERE command like '%DWResultCacheDb%'  
    AND step_index = 0
```

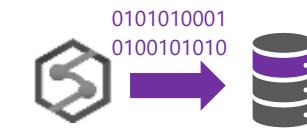
# Result-set caching flow



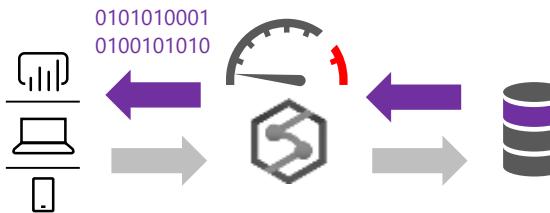
- 1 Client sends query to SQL pool



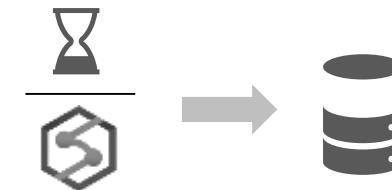
- 2 Query is processed using compute nodes which pull data from remote storage, process query and output back to client app



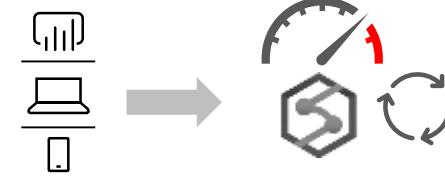
+ Query results are cached in remote storage so subsequent requests can be served immediately



- 3 Subsequent executions for the same query bypass compute nodes and can be fetched instantly from persistent cache in remote storage



- 4 Remote storage cache is evicted regularly based on time, cache usage, and any modifications to underlying table data.



- 5 Cache will need to be regenerated if query results have been evicted from cache

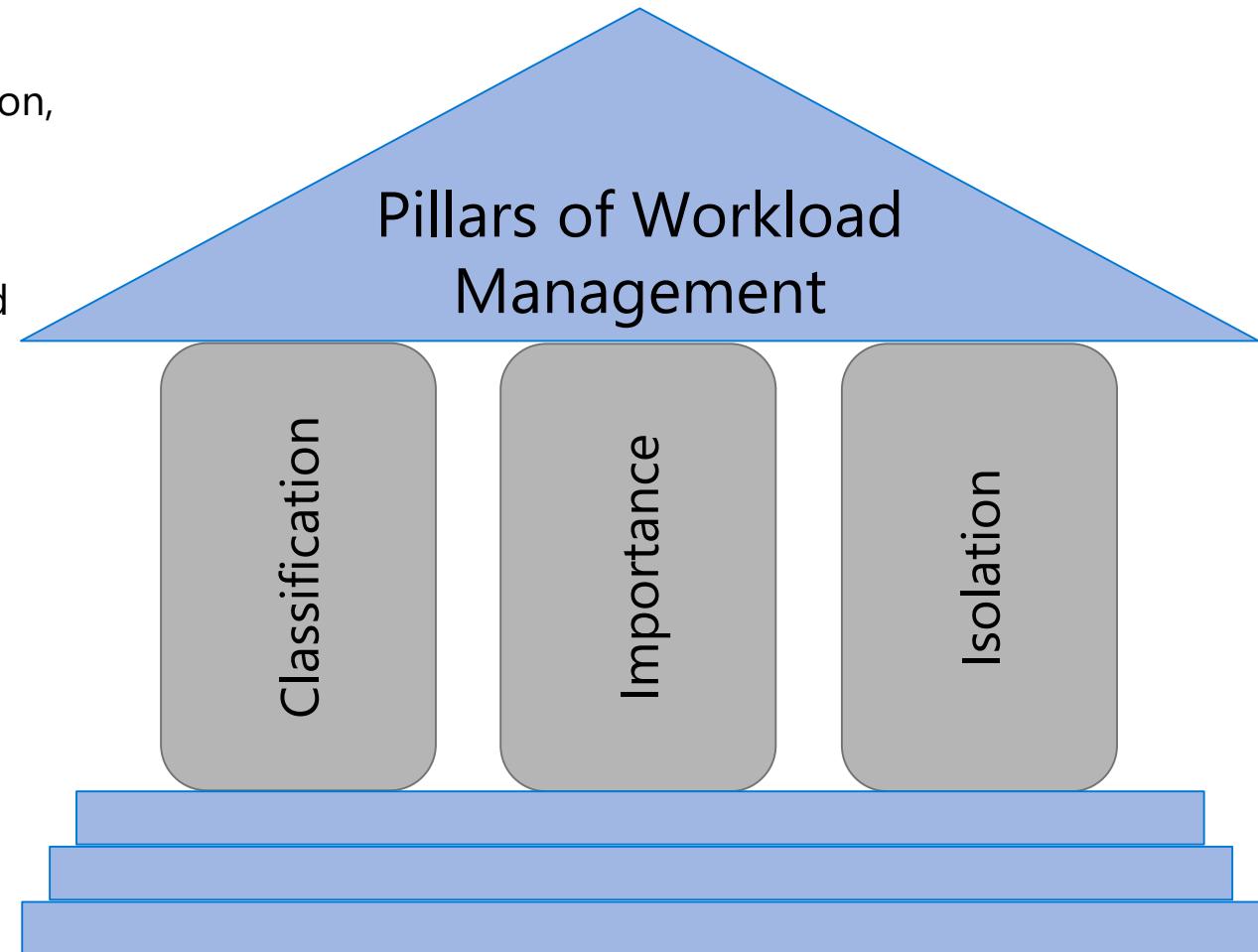
# Workload Management

## Overview

It manages resources, ensures highly efficient resource utilization, and maximizes return on investment (ROI).

The three pillars of workload management are

1. Workload Classification – To assign a request to a workload group and setting importance levels.
2. Workload Importance – To influence the order in which a request gets access to resources.
3. Workload Isolation – To reserve resources for a workload group.



# Workload classification

## Overview

Map queries to allocations of resources via pre-determined rules.

Use with workload importance to effectively share resources across different workload types.

If a query request is not matched to a classifier, it is assigned to the default workload group.

## Benefits

Map queries to both Resource Management and Workload Isolation concepts.

## Monitoring DMVs

[sys.workload\\_management\\_workload\\_classifiers](#)

[sys.workload\\_management\\_workload\\_classifier\\_details](#)

Query DMVs to view details about all active workload classifiers.

```
CREATE WORKLOAD CLASSIFIER classifier_name
WITH
(
    WORKLOAD_GROUP = 'name'
    , MEMBERNAME = 'security_account'
    [ [,] IMPORTANCE = {LOW|BELOW_NORMAL|NORMAL|ABOVE_NORMAL|HIGH} ]
    [ [,] WLM_LABEL = 'label' ]
    [ [,] WLM_CONTEXT = 'name' ]
    [ [,] START_TIME = 'start_time' ]
    [ [,] END_TIME = 'end_time' ]
)[ ; ]
```

*WORKLOAD\_GROUP: maps to an existing resource class*

*IMPORTANCE: specifies relative importance of request*

*MEMBERNAME: database user, role, AAD login or AAD group*

# Workload importance

## Overview

Queries past the concurrency limit enter a FiFo queue

By default, queries are released from the queue on a first-in, first-out basis as resources become available

Workload importance allows higher priority queries to receive resources immediately regardless of queue

## Example Video

State analysts have normal importance.

National analyst is assigned high importance.

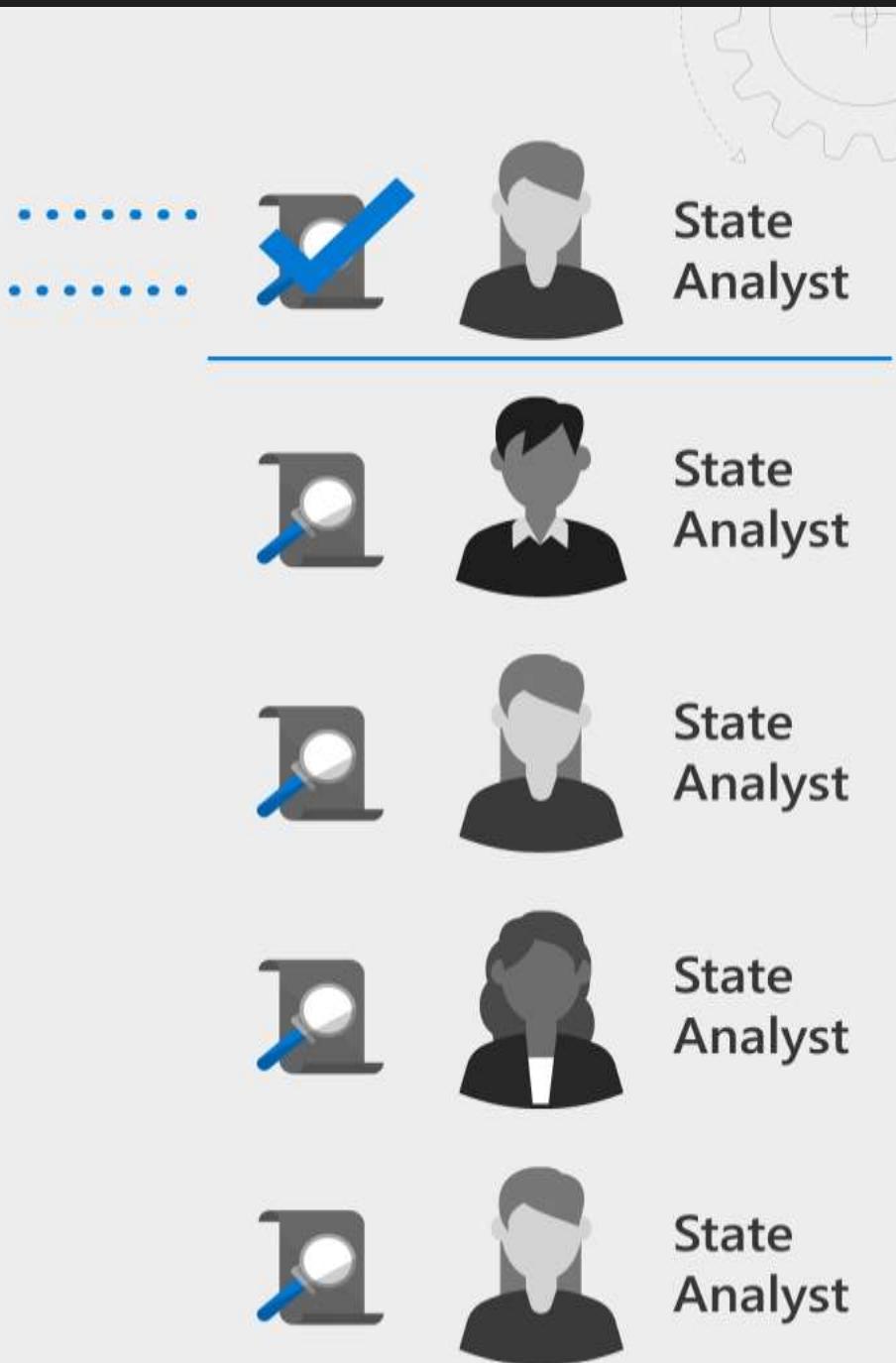
State analyst queries execute in order of arrival

When the national analyst's query arrives, it jumps to the top of the queue

```
CREATE WORKLOAD CLASSIFIER National_Analyst
WITH
(
    WORKLOAD_GROUP = 'analyst'
    ,IMPORTANCE = HIGH
    ,MEMBERNAME = 'National_Analyst_Login')
```



Azure Synapse  
Analytics



# Workload Isolation

## Overview

Allocate fixed resources to workload group.

Assign maximum and minimum usage for varying resources under load. These adjustments can be done live without having to SQL Analytics offline.

## Benefits

Reserve resources for a group of requests

Limit the amount of resources a group of requests can consume

Shared resources accessed based on importance level

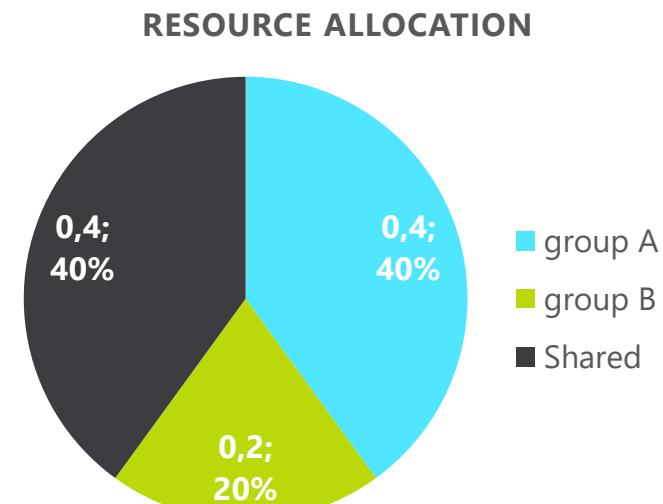
Set Query timeout value. Get DBAs out of the business of killing runaway queries

## Monitoring DMVs

[sys.workload\\_management\\_workload\\_groups](#)

Query to view configured workload group.

```
CREATE WORKLOAD GROUP group_name
WITH
(
    MIN_PERCENTAGE_RESOURCE = value
    , CAP_PERCENTAGE_RESOURCE = value
    , REQUEST_MIN_RESOURCE_GRANT_PERCENT = value
    [[,] REQUEST_MAX_RESOURCE_GRANT_PERCENT = value ]
    [[,] IMPORTANCE = {LOW | BELOW_NORMAL | NORMAL | ABOVE_NORMAL | HIGH} ]
    [[,] QUERY_EXECUTION_TIMEOUT_SEC = value ]
)[;]
```



# Dynamic Management Views (DMVs)

## Overview

Dynamic Management Views (DMV) are queries that return information about model objects, server operations, and server health.

## Benefits:

Simple SQL syntax

Returns result in table format

Easier to read and copy result

# SQL Monitor with DMVs

## Overview

Offers monitoring of

- all open, closed sessions
- count sessions by user
- count completed queries by user
- all active, complete queries
- longest running queries
- memory consumption

Count sessions by user

--count sessions by user

```
SELECT login_name, COUNT(*) as session_count FROM sys.dm_pdw_exec_sessions  
where status = 'Closed' and session_id <> session_id() GROUP BY login_name;
```

List all open sessions

-- List all open sessions

```
SELECT * FROM sys.dm_pdw_exec_sessions where status <> 'Closed' and session_id <>  
session_id();
```

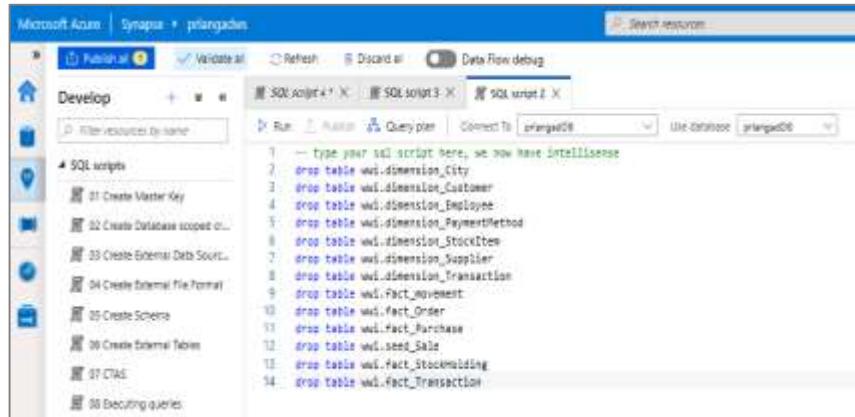
List all active queries

-- List all active queries

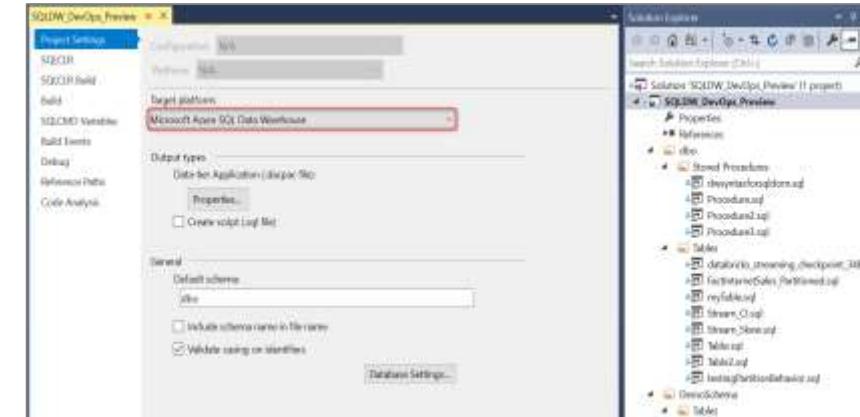
```
SELECT * FROM sys.dm_pdw_exec_requests WHERE status not in  
('Completed', 'Failed', 'Cancelled') AND session_id <> session_id() ORDER BY submit_time  
DESC;
```

# Developer Tools

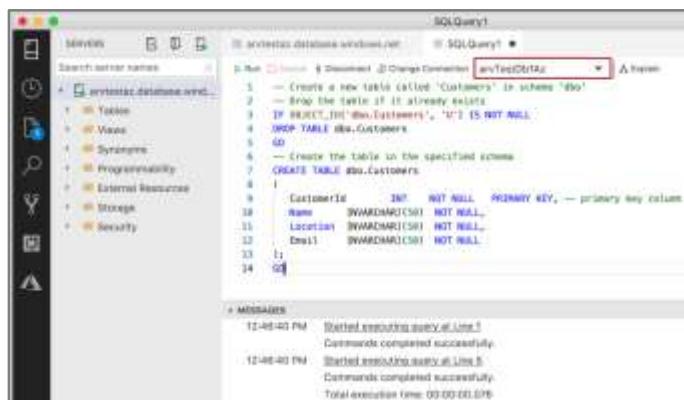
Azure Synapse Analytics



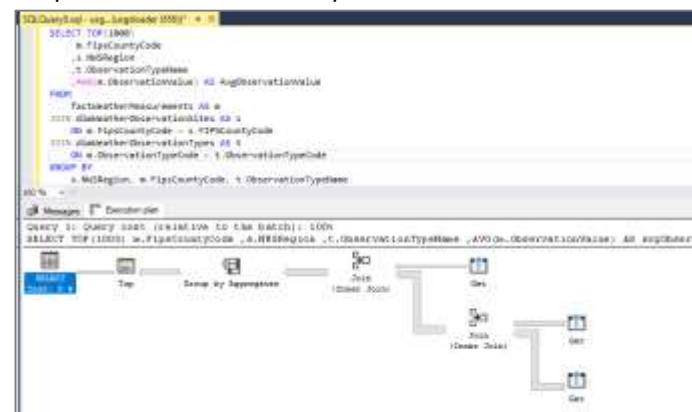
Visual Studio - SSDT database projects



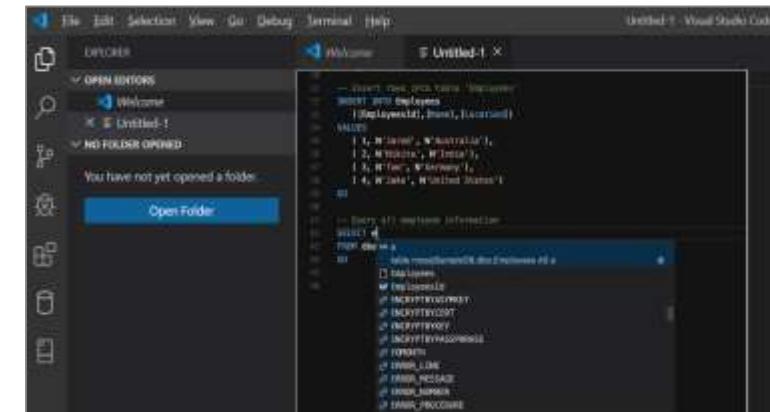
Azure Data Studio (queries, extensions etc.)



SQL Server Management Studio (queries, execution plans etc.)

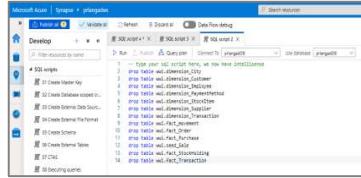


Visual Studio Code



# Developer Tools

## Azure Synapse Analytics

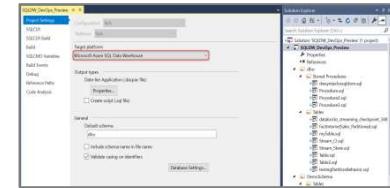


Azure Cloud Service

Offers end-to-end lifecycle for analytics

Connects to multiple services

## Visual Studio - SSDT database projects



Runs on Windows

Create, maintain database code, compile, code refactoring

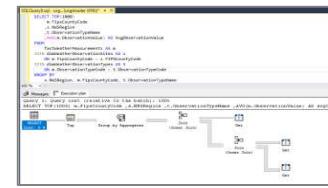
## Azure Data Studio



Runs on Windows, Linux, macOS

Light weight editor, (queries and extensions)

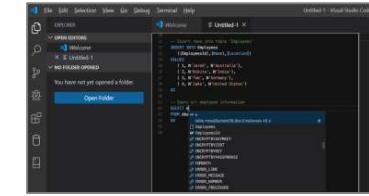
## SQL Server Management Studio



Runs on Windows

Offers GUI support to query, design and manage

## Visual Studio Code



Runs on Windows, Linux, macOS

Offers development experience with light-weight code editor

# Continuous integration and delivery (CI/CD)

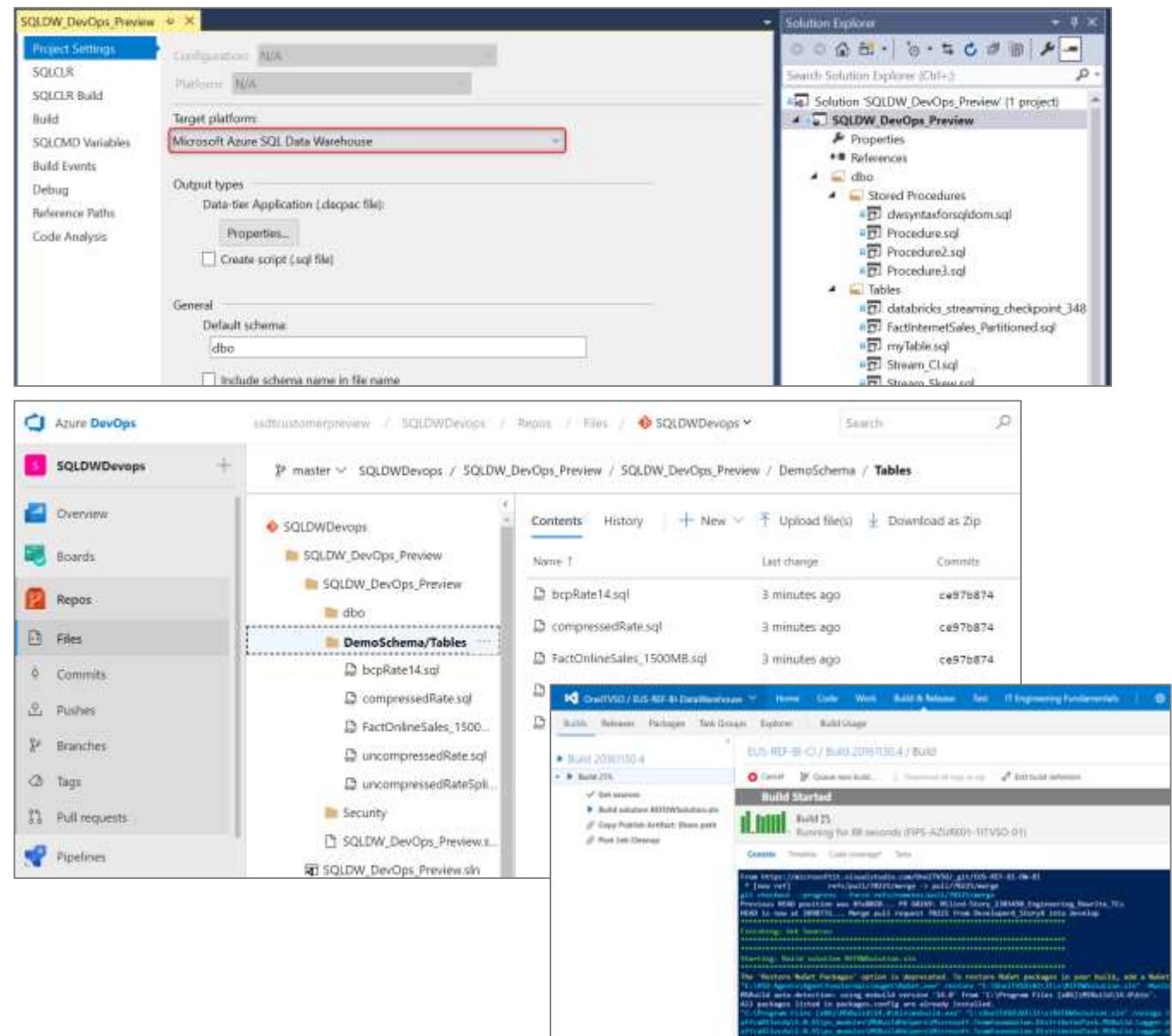
## Overview

Database project support in SQL Server Data Tools (SSDT) allows teams of developers to collaborate over a version-controlled Azure Synapse Analytics, and track, deploy and test schema changes.

## Benefits

Database project support includes first-class integration with Azure DevOps. This adds support for:

- **Azure Pipelines** to run CI/CD workflows for any platform (Linux, macOS, and Windows)
- **Azure Repos** to store project files in source control
- **Azure Test Plans** to run automated check-in tests to verify schema updates and modifications
- Growing ecosystem of third-party integrations that can be used to complement existing workflows (Timetracker, Microsoft Teams, Slack, Jenkins, etc.)



# Azure Advisor recommendations

## Suboptimal Table Distribution

Reduce data movement by replicating tables

## Data Skew

Choose new hash-distribution key

Slowest distribution limits performance

## Cache Misses

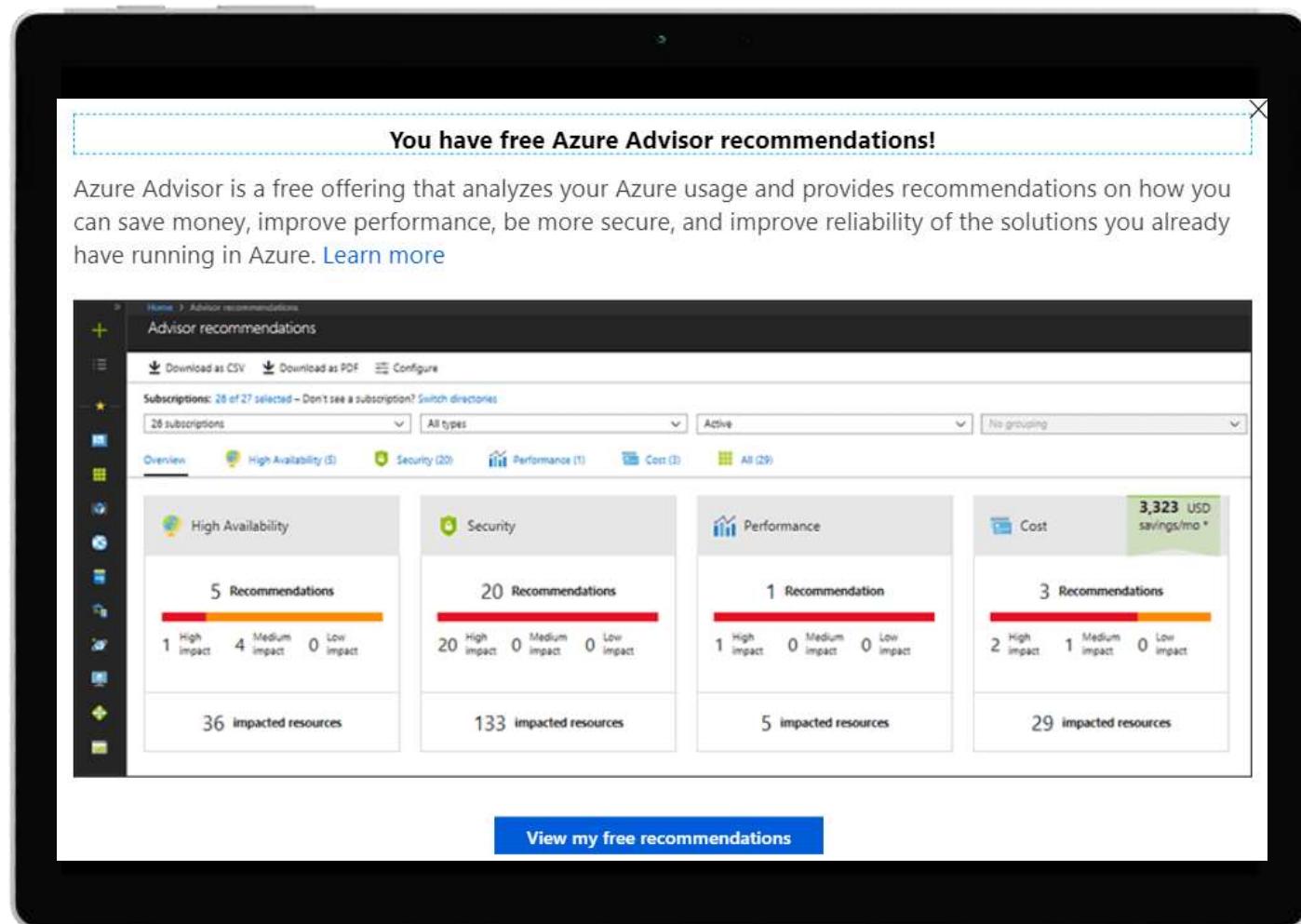
Provision additional capacity

## Tempdb Contention

Scale or update user resource class

## Suboptimal Plan Selection

Create or update table statistics



# Maintenance windows

## Overview

Choose a time window for your upgrades.

Select a primary and secondary window within a seven-day period.

Windows can be from 3 to 8 hours.

24-hour advance notification for maintenance events.

## Benefits

Ensure upgrades happen on your schedule.

Predictable planning for long-running jobs.

Stay informed of start and end of maintenance.

The screenshot shows the 'Maintenance Schedule (preview)' page in the Azure portal. The top navigation bar includes 'Home', 'maintenanceexamples', and 'Maintenance Schedule (preview)'. Below the navigation is a toolbar with 'Save', 'Discard', and 'Feedback' buttons. A sidebar on the left contains various icons for different Azure services. The main content area has an informational box with an info icon stating: 'Maintenance on your data warehouse could occur once a week within one of two maintenance windows. Choose the primary and secondary windows that best suit your operational needs. If you would like to use the maintenance windows already defined, no action is required.' It also notes that 'Maintenance will not take place outside these windows unless we notify you in advance.' Below this is a section titled 'Choose primary window' with radio buttons for 'Saturday - Sunday' (selected) and 'Tuesday - Thursday'. The 'Primary maintenance window' section shows 'Day' set to 'Saturday', 'Start time' at '03:00 UTC', and 'Time window' at '8 hours'. The 'Secondary maintenance window' section shows 'Day' set to 'Tuesday', 'Start time' at '13:00 UTC', and 'Time window' at '8 hours'. At the bottom is a 'Schedule summary' section with the text: 'Primary maintenance window Saturday 03:00 UTC (8 hours)' and 'Secondary maintenance window Tuesday 13:00 UTC (8 hours)'.

# Automatic statistics management

## Overview

Statistics are automatically created and maintained for SQL pool. Incoming queries are analyzed, and individual column statistics are generated on the columns that improve cardinality estimates to enhance query performance.

Statistics are automatically updated as data modifications occur in underlying tables. By default, these updates are synchronous but can be configured to be asynchronous.

Statistics are considered out of date when:

- There was a data change on an empty table
- The number of rows in the table at time of statistics creation was 500 or less, and more than 500 rows have been updated
- The number of rows in the table at time of statistics creation was more than 500, and more than  $500 + 20\%$  of rows have been updated

-- Turn on/off auto-create statistics settings

```
ALTER DATABASE {database_name}
```

```
SET AUTO_CREATE_STATISTICS { ON | OFF }
```

-- Turn on/off auto-update statistics settings

```
ALTER DATABASE {database_name}
```

```
SET AUTO_UPDATE_STATISTICS { ON | OFF }
```

-- Configure synchronous/asynchronous update

```
ALTER DATABASE {database_name}
```

```
SET AUTO_UPDATE_STATISTICS_ASYNC { ON | OFF }
```

-- Check statistics settings for a database

```
SELECT      is_auto_create_stats_on,  
            is_auto_update_stats_on,  
            is_auto_update_stats_async_on  
FROM        sys.databases
```

SQL On-Demand  
SQL On Demand

# SQL On-Demand

## Overview

An interactive query service that provides T-SQL queries over high scale data in Azure Storage.

## Benefits

Serverless

No infrastructure

Pay only for query execution

No ETL

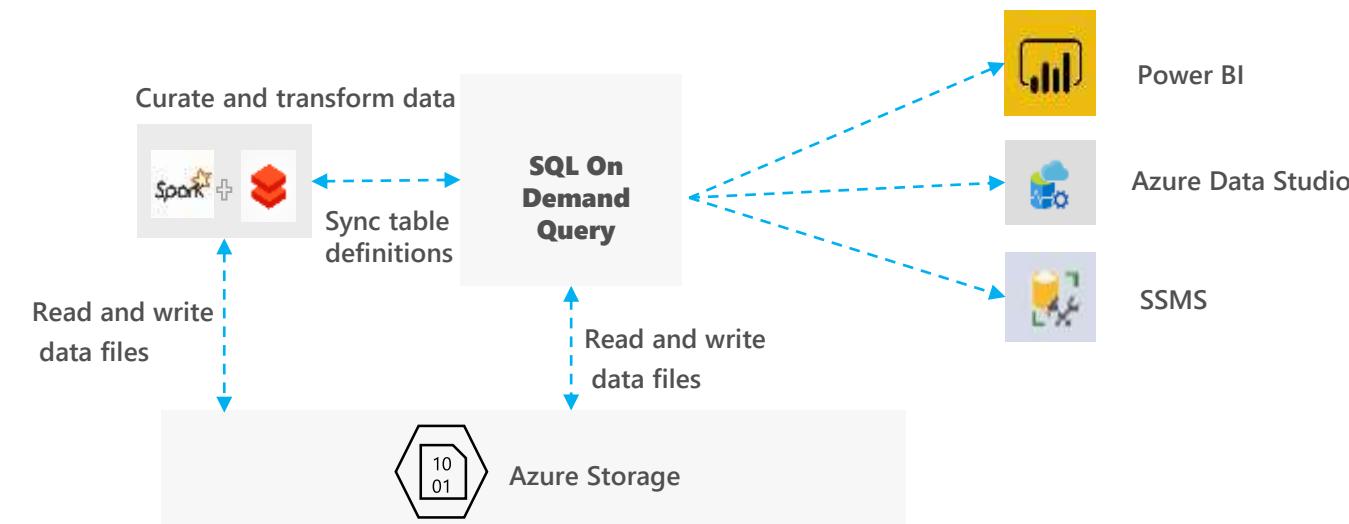
Offers security

Data integration with Databricks, HDInsight

T-SQL syntax to query data

Supports data in various formats (Parquet, CSV, JSON)

Support for BI ecosystem



# SQL On Demand – Querying on storage

The screenshot displays the Microsoft Azure Synapse Analytics interface, specifically the "SQL On Demand" feature for querying storage.

**Left Tab (File System View):**

- Shows a hierarchical file structure under "Storage accounts" for "prlangademos (Primary)".
  - filesystem
  - holidaydatacontainer
  - isweathermapdatacontainer
  - nytic** (selected)
  - prlangaddemos
  - tmpcontainer
  - wwimporters
- Shows a "New SQL script and open in new tab" context menu for a file named "part-00133".
  - New SQL script
  - New notebook
  - Copy ABFS path
  - Manage Access...
  - Rename...
  - Download
  - Delete
  - Properties...

**Right Tab (Query Editor):**

- Shows the "SQL Analytics on-demand" connection selected in the "Connect to" dropdown.
- The query window contains the following T-SQL code:
 

```
1 SELECT
2     TOP 100 *
3     FROM
4     OPENROWSET(
5         Bulk 'https://prlangadidemoa.blob.core.windows.net/nytic/c/yellow/puYear=2015/puMonth=3/part-00133-11d21803864719830543-aea5b543-5e83-4a7d-8d31-69f72c500050-15253-1.c000.snappy.parquet'
6         FORMAT='PARQUET'
7     ) AS nyte
```
- The "Results" pane displays the query results as a table:

VENDORID	TPEPICKUPDATETIME	TPEPDISPATCHDATE_	PASSENGERCOUNT	TRIPDISTANCE	PILOCATIONID	DOLocationID	STARTSTATION	ENDSTATION	TRIPID
2	2015-02-28T23:5...	2015-03-01T00:0...	6	1.63	40011	40011	-74.000540662793	40.7306938171387	-73
1	2015-03-28T19:2...	2015-03-28T19:2...	1	2.2	40011	40011	-73.977653503418	40.7621607035664	-73
2	2015-02-28T23:5...	2015-03-01T00:1...	5	3.23	40011	40011	-73.96012879417...	40.7621574462185	-73
1	2015-03-28T19:2...	2015-03-28T19:3...	1	2.1	40011	40011	-73.9814305371...	40.7815955847368	-74
2	2015-02-28T23:5...	2015-03-01T00:1...	1	3.52	40011	40011	-73.98373413085...	40.7497062683105	-74
5	2015-02-28T23:5...	2015-03-01T00:1...	6	1.63	40011	40011	-73.9814305371...	40.7815955847368	-74

- A message at the bottom indicates: "00:01:00-Query executed successfully."

# SQL On Demand – Querying CSV File

## Overview

Uses OPENROWSET function to access data

## Benefits

Ability to read CSV File with

- no header row, Windows style new line
- no header row, Unix-style new line
- header row, Unix-style new line
- header row, Unix-style new line, quoted
- header row, Unix-style new line, escape
- header row, Unix-style new line, tab-delimited
- without specifying all columns

```
SELECT *
FROM OPENROWSET(
    BULK 'https://XXX.blob.core.windows.net/csv/population/population.csv',
    FORMAT = 'CSV',
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n'
)
WITH (
    [country_code] VARCHAR (5) COLLATE Latin1_General_BIN2,
    [country_name] VARCHAR (100) COLLATE Latin1_General_BIN2,
    [year] smallint,
    [population] bigint
) AS [r]
WHERE
    country_name = 'Luxembourg'
    AND year = 2017
```

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

# SQL On Demand – Querying CSV File

Read CSV file - header row, Unix-style new line

```

SELECT *
FROM OPENROWSET(
    BULK 'https://XXX.blob.core.windows.net/csv/population-unix-hdr/population.csv',
    FORMAT = 'CSV',
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '0x0a',
    FIRSTROW = 2
)
WITH (
    [country_code] VARCHAR (5) COLLATE Latin1_General_BIN2,
    [country_name] VARCHAR (100) COLLATE Latin1_General_BIN2,
    [year] smallint,
    [population] bigint
) AS [r]
WHERE
    country_name = 'Luxembourg'
    AND year = 2017

```

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

Read CSV file - without specifying all columns

```

SELECT
    COUNT(DISTINCT country_name) AS countries
FROM OPENROWSET(
    BULK 'https://XXX.blob.core.windows.net/csv/population/population.csv',
    FORMAT = 'CSV',
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n'
)
WITH (
    [country_name] VARCHAR (100) COLLATE Latin1_General_BIN2 2
) AS [r]

```

	countries
1	228

# SQL On Demand – Querying folders

## Overview

Uses OPENROWSET function to access data from multiple files or folders

## Benefits

Offers reading multiple files/folders through usage of wildcards

Offers reading specific file/folder

Supports use of multiple wildcards

```

SELECT YEAR(pickup_datetime) AS [year], SUM(passenger_count) AS passengers_total,
COUNT(*) AS [rides_total]
FROM OPENROWSET(
BULK 'https://XXX.blob.core.windows.net/csv/taxi/*.csv',
FORMAT = 'CSV'
, FIRSTROW = 2 )
WITH (
    vendor_id VARCHAR(100) COLLATE Latin1_General_BIN2,
    pickup_datetime DATETIME2,
    dropoff_datetime DATETIME2,
    passenger_count INT,
    trip_distance FLOAT,
    rate_code INT,
    store_and_fwd_flag VARCHAR(100) COLLATE Latin1_General_BIN2,
    pickup_location_id INT,
    dropoff_location_id INT,
    payment_type INT,
    fare_amount FLOAT,
    extra FLOAT, mta_tax FLOAT,
    tip_amount FLOAT,
    tolls_amount FLOAT,
    improvement_surcharge FLOAT,
    total_amount FLOAT
) AS nyc
GROUP BY YEAR(pickup_datetime)
ORDER BY YEAR(pickup_datetime)

```

	year	passengers_total	rides_total
1	2001	14	10
2	2002	29	16
3	2003	22	16
4	2008	378	188
5	2009	594	353
6	2016	102093687	61758523
7	2017	184464988	113496932
8	2018	86272771	53925040
9	2019	37	29
...	2020	6	6

# SQL On Demand – Querying folders

Read all files from multiple folders

```
SELECT YEAR(pickup_datetime) AS [year],
       SUM(passenger_count) AS passengers_total,
       COUNT(*) AS [rides_total]
  FROM OPENROWSET(
    BULK 'https://XXX.blob.core.windows.net/csv/t*/*/',
    FORMAT = 'CSV',
    FIRSTROW = 2
  )
  WITH (
    vendor_id VARCHAR(100) COLLATE Latin1_General_BIN2,
    pickup_datetime DATETIME2,
    dropoff_datetime DATETIME2,
    passenger_count INT,
    trip_distance FLOAT,
    <... columns>
  ) AS nyc
 GROUP BY YEAR(pickup_datetime)
 ORDER BY YEAR(pickup_datetime)
```

	year	passengers_total	rides_total
1	2001	14	10
2	2002	29	16
3	2003	22	16
4	2008	378	188
5	2009	594	353
6	2016	102093687	61758523
7	2017	184464988	113496932
8	2018	86272771	53925040
9	2019	37	29
...	2020	6	6

Read subset of files in folder

```
SELECT
    payment_type,
    SUM(fare_amount) AS fare_total
  FROM OPENROWSET(
    BULK 'https://XXX.blob.core.windows.net/csv/taxi/yellow_tripdata_2017-*.*',
    FORMAT = 'CSV',
    FIRSTROW = 2
  )
  WITH (
    vendor_id VARCHAR(100) COLLATE Latin1_General_BIN2,
    pickup_datetime DATETIME2,
    dropoff_datetime DATETIME2,
    passenger_count INT,
    trip_distance FLOAT,
    <... columns>
  ) AS nyc
 GROUP BY payment_type
 ORDER BY payment_type
```

	payment_type	fare_total
1	1	1026072325.579...
2	2	441093322.8000...
3	3	10435183.04
4	4	3304550.99
5	5	14

# SQL On Demand – Querying specific files

## Overview

`filename` – Provides file name that originates row result

`filepath` – Provides full path when no parameter is passed or part of path when parameter is passed that originates result

## Benefits

Provides source name/path of file/folder for row result set

### Example of filename function

```
SELECT
    r.filename() AS [filename]
    ,COUNT_BIG(*) AS [rows]
FROM OPENROWSET(
    BULK 'https://XXX.blob.core.windows.net/csv/taxi/yellow_tripdata_2017-1*.csv',
    FORMAT = 'CSV',
    FIRSTROW = 2
)
WITH (
    vendor_id INT,
    pickup_datetime DATETIME2,
    dropoff_datetime DATETIME2,
    passenger_count SMALLINT,
    trip_distance FLOAT,
    <...columns>
) AS [r]
```

GROUP BY r.filename()

ORDER BY [filename]

	filename	rows
1	yellow_tripdata_2017-10.csv	9768815
2	yellow_tripdata_2017-11.csv	9284803
3	yellow_tripdata_2017-12.csv	9508276

# SQL On Demand – Querying specific files

Example of filepath function

```

SELECT
    r.filepath() AS filepath
    ,r.filepath(1) AS [year]
    ,r.filepath(2) AS [month]
    ,COUNT_BIG(*) AS [rows]
FROM OPENROWSET(
    BULK 'https://XXX.blob.core.windows.net/csv/taxi/yellow_tripdata_*-* .csv',
    FORMAT = 'CSV',
    FIRSTROW = 2
)
WITH (
    vendor_id INT,
    pickup_datetime DATETIME2,
    dropoff_datetime DATETIME2,
    passenger_count SMALLINT,
    trip_distance FLOAT,
    <... columns>
) AS [r]

```

```

WHERE r.filepath(1) IN ('2017')
    AND r.filepath(2) IN ('10', '11', '12')

```

```

GROUP BY r.filepath(),r.filepath(1),r.filepath(2)
ORDER BY filepath

```

filepath	year	month	rows
https://XXX.blob.core.windows.net/csv/taxi/yellow_tripdata_2017-10.csv	2017	10	9768815
https://XXX.blob.core.windows.net/csv/taxi/yellow_tripdata_2017-11.csv	2017	11	9284803
https://XXX.blob.core.windows.net/csv/taxi/yellow_tripdata_2017-12.csv	2017	12	9508276

# SQL On Demand – Querying Parquet files

## Overview

Uses OPENROWSET function to access data

## Benefits

Ability to specify column names of interest

Offers auto reading of column names and data types

Provides target specific partitions using filepath function

```

SELECT
    YEAR(pickup_datetime),
    passenger_count,
    COUNT(*) AS cnt
FROM
    OPENROWSET(
        BULK 'https://XXX.blob.core.windows.net/parquet/taxi/\*/\*/\*',
        FORMAT='PARQUET'
    ) WITH (
        pickup_datetime DATETIME2,
        passenger_count INT
    ) AS nyc
GROUP BY
    passenger_count,
    YEAR(pickup_datetime)
ORDER BY
    YEAR(pickup_datetime),
    passenger_count
  
```

	(No column name)	passenger_count	cnt
1	2016	0	2557
2	2016	1	43735845
3	2016	2	9056714
4	2016	3	2610541
5	2016	4	1309639
6	2016	5	3086097
7	2016	6	1956607

# SQL On Demand – Creating views

## Overview

Create views using SQL On Demand queries

## Benefits

Works same as standard views

```
USE [mydbname]
GO

IF EXISTS(select * FROM sys.views where name = 'populationView')
DROP VIEW populationView
GO

CREATE VIEW populationView AS
SELECT *
FROM OPENROWSET(
    BULK 'https://XXX.blob.core.windows.net/csv/population/population.csv',
    FORMAT = 'CSV',
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n'
)
WITH (
    [country_code] VARCHAR (5) COLLATE Latin1_General_BIN2,
    [country_name] VARCHAR (100) COLLATE Latin1_General_BIN2,
    [year] smallint,
    [population] bigint
) AS [r]
```

```
SELECT
    country_name, population
FROM populationView
WHERE
    [year] = 2019
ORDER BY
    [population] DESC
```

	country_name	population
1	China	1389618778
2	India	1311559204
3	United States	331883986
4	Indonesia	264935824
5	Pakistan	210797836
6	Brazil	210301591
7	Nigeria	208679114
8	Bangladesh	161062905
9	Russia	141944641
10	Mexico	127318112

# SQL On Demand – Creating views

The screenshot shows three windows from the Microsoft Azure Synapse Analytics studio:

- Top Left Window:** A query editor titled "SQL script i \*". It contains the following T-SQL code:
 

```

1 -- type your sql script here, we now have intellisense
2 CREATE VIEW yellow_2017 AS
3 SELECT *
4 FROM
5 OPENROWSET(
6   BULK 'https://prlangaddemosa.dfs.core.windows.net/nyctlc/yellow/puYear=2017/*/*',
7   FORMAT='PARQUET'
8 ) AS nyc
9 
```
- Bottom Left Window:** A query editor titled "SQL script i \*". It contains the same T-SQL code as the top window. Below the code, the status message "00:00:17 Query executed successfully." is displayed.
- Bottom Right Window:** A chart viewer titled "Chart". It displays a line chart with "passengerCount" on the X-axis (ranging from 0 to 10) and "TBL" on the Y-axis (ranging from 0 to 10M). The chart shows a single data series represented by green dots connected by a line. The legend indicates "passengerCount" (blue dot) and "tbl" (green dot).

The screenshot shows a Microsoft Azure Synapse Analytics studio interface with the following details:

- Top Bar:** Shows "Microsoft Azure | Synapse Analytics > prlangadws2". The "Connect to" dropdown is highlighted with a red box and set to "SQL Analytics on-demand".
- Query Editor:** Displays the same T-SQL code as the other windows:
 

```

1 SELECT
2   YEAR(tpepPickupDateTime),
3   passengerCount,
4   COUNT(*) AS cnt
5 FROM
6   yellow_2017
7 GROUP BY
8   passengerCount,
9   YEAR(tpepPickupDateTime)
10 ORDER BY
11   YEAR(tpepPickupDateTime),
12   passengerCount

```
- Results View:** Shows the results of the query in a "Table" view. The data is as follows:
 

(NO COLUMN NAME)	PASSENGERCOUNT	CNT
2017	0	166086
2017	1	81034075
2017	2	16545571
2017	3	4748869
2017	4	2257813
2017	5	5407319
- Message:** Below the results, the message "00:02:19 Query executed successfully." is shown.

# SQL On Demand – Querying JSON files

## Overview

Read JSON files and provides data in tabular format

## Benefits

Supports OPENJSON, JSON\_VALUE and JSON\_QUERY functions

```
SELECT *
FROM
OPENROWSET(
    BULK 'https://XXX.blob.core.windows.net/json/books/book1.json',
    FORMAT='CSV',
    FIELDTERMINATOR = '0x0b',
    FIELDQUOTE = '0x0b',
    ROWTERMINATOR = '0x0b'
)
WITH (
    jsonContent varchar(8000)
) AS [r]
```

jsonContent
1 {"_id": "kim95", "type": "Book", "title": "Modern Databas...

# SQL On Demand – Querying JSON files

## Example of JSON\_VALUE function

```

SELECT
    JSON_VALUE(jsonContent, '$.title') AS title,
    JSON_VALUE(jsonContent, '$.publisher') AS publisher,
    jsonContent
FROM
    OPENROWSET(
        BULK 'https://XXX.blob.core.windows.net/json/books/*.json',
        FORMAT='CSV',
        FIELDTERMINATOR = '0x0b',
        FIELDQUOTE = '0x0b',
        ROWTERMINATOR = '0x0b'
    )
    WITH (
        jsonContent varchar(8000)
    ) AS [r]
WHERE
    JSON_VALUE(jsonContent, '$.title') = 'Probabilistic and Statistical Methods in Cryptology,
    An Introduction by Selected Topics'

```

	title	publisher	jsonContent
1	Probabilistic and Statistical Methods in Cryptology, An Int...	Springer	{"_id": "neuen...",

## Example of JSON\_QUERY function

```

SELECT
    JSON_QUERY(jsonContent, '$.authors') AS authors,
    jsonContent
FROM
    OPENROWSET(
        BULK 'https://XXX.blob.core.windows.net/json/books/*.json',
        FORMAT='CSV',
        FIELDTERMINATOR = '0x0b',
        FIELDQUOTE = '0x0b',
        ROWTERMINATOR = '0x0b'
    )
    WITH (
        jsonContent varchar(8000)
    ) AS [r]
WHERE
    JSON_VALUE(jsonContent, '$.title') = 'Probabilistic and Statistical Methods in Cryptology,
    An Introduction by Selected Topics'

```

	authors	jsonContent
1	["Daniel Neuenschwander"]	{"_id": "neuenschwander04", "type": "Book", "title": "Probabi..."

# Create External Table As Select

## Overview

Creates an external table and then exports results of the Select statement. These operations will import data into the database for the duration of the query

### Steps:

1. Create Master Key
2. Create Credentials
3. Create External Data Source
4. Create External Data Format
5. Create External Table

```
-- Create a database master key if one does not already exist
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'S0me!Info'
;

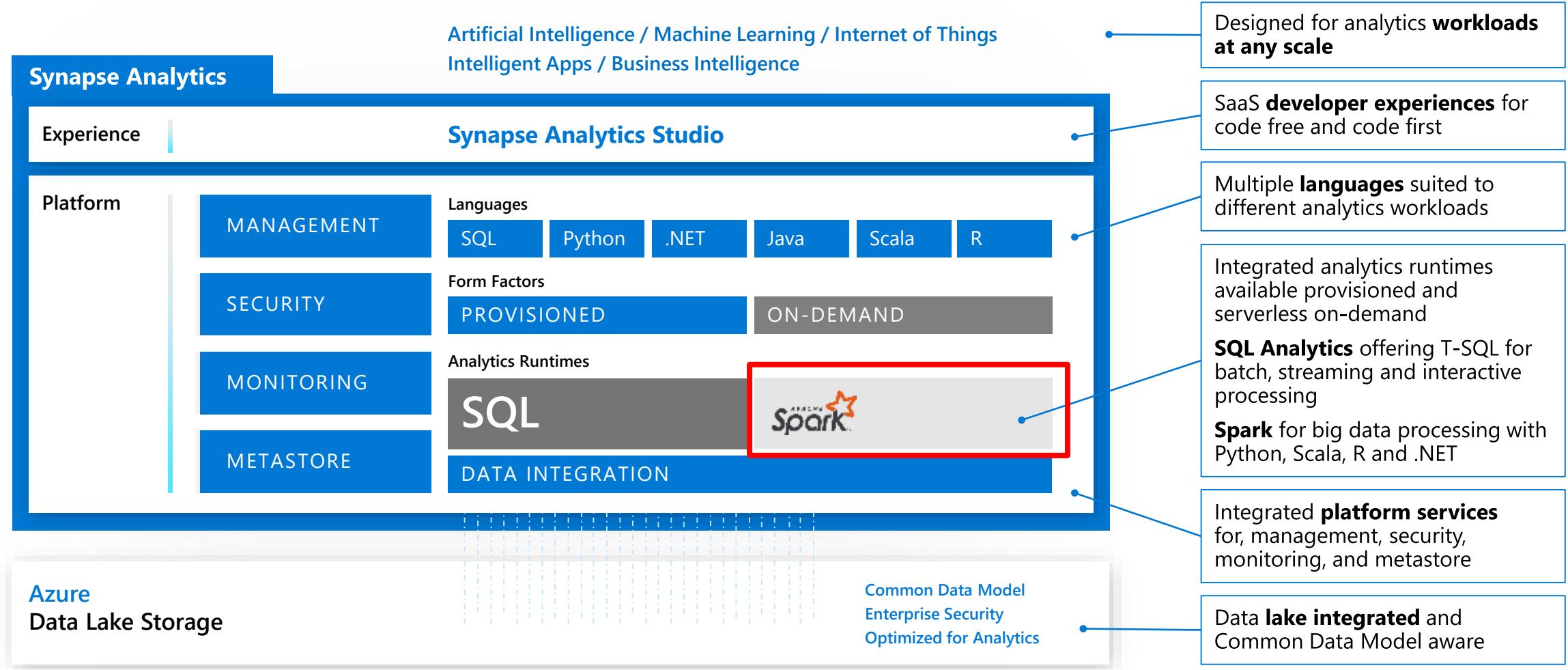
-- Create a database scoped credential with Azure storage account key as the secret.
CREATE DATABASE SCOPED CREDENTIAL AzureStorageCredential
WITH
    IDENTITY = '<my_account>'
, SECRET   = '<azure_storage_account_key>'
;
-- Create an external data source with CREDENTIAL option.
CREATE EXTERNAL DATA SOURCE MyAzureStorage
WITH
(
    LOCATION  = 'wasbs://daily@logs.blob.core.windows.net/'
, CREDENTIAL = AzureStorageCredential
, TYPE      = HADOOP
)
-- Create an external file format
CREATE EXTERNAL FILE FORMAT MyAzureCSVFormat
WITH (FORMAT_TYPE = DELIMITEDTEXT,
      FORMAT_OPTIONS(
          FIELD_TERMINATOR = ',',
          FIRST_ROW = 2))
--Create an external table
CREATE EXTERNAL TABLE dbo.FactInternetSalesNew
WITH(
    LOCATION = '/files/Customer',
    DATA_SOURCE = MyAzureStorage,
    FILE_FORMAT = MyAzureCSVFormat
)
AS SELECT T1.* FROM dbo.FactInternetSales T1 JOIN dbo.DimCustomer T2
ON ( T1.CustomerKey = T2.CustomerKey )
OPTION ( HASH JOIN );
```



# Azure Synapse Analytics Spark

# Azure Synapse Analytics

Limitless analytics service with unmatched time to insight





# Azure Synapse Apache Spark - Summary

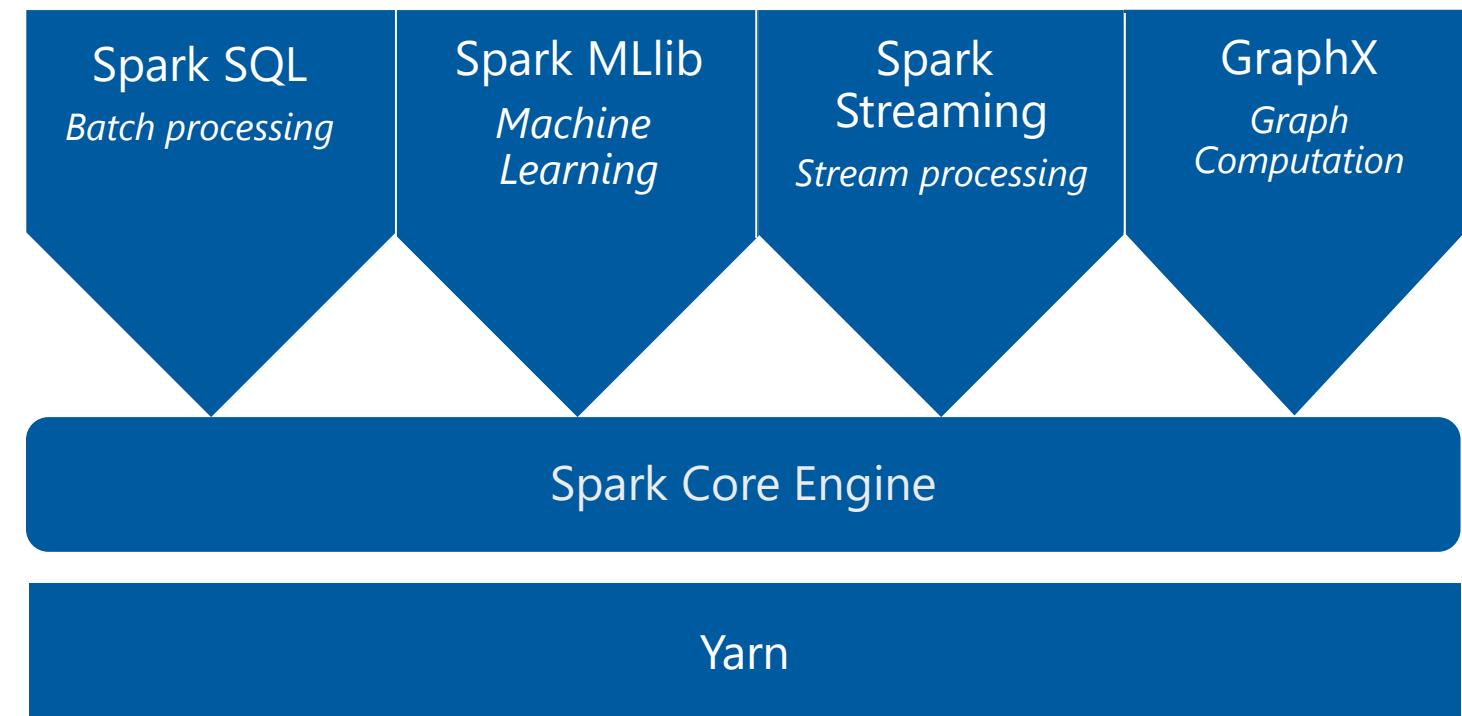
- **Apache Spark 2.4 derivation**
  - Linux Foundation Delta Lake 0.4 support
  - .Net Core 3.0 support
  - Python 3.6 + Anacondas support
- **Tightly coupled to other Azure Synapse services**
  - Integrated security and sign on
  - Integrated Metadata
  - Integrated and simplified provisioning
  - Integrated UX including nteract based notebooks
  - Fast load of SQL Analytics pools
- **Core scenarios**
  - Data Prep/Data Engineering/ETL
  - Machine Learning via Spark ML and Azure ML integration
  - Extensible through library management
- **Efficient resource utilization**
  - Fast Start
  - Auto scale (up and down)
  - Auto pause
  - Min cluster size of 3 nodes
- **Multi Language Support**
  - .Net (C#), PySpark, Scala, Spark SQL, Java

# Apache Spark

An unified, open source, parallel, data processing framework for Big Data Analytics

## Spark Unifies:

- Batch Processing
- Interactive SQL
- Real-time processing
- Machine Learning
- Deep Learning
- Graph Processing



<http://spark.apache.org>

# Motivation for Apache Spark

Traditional Approach: MapReduce jobs for complex jobs, interactive query, and online event-hub processing involves lots of (slow) disk I/O

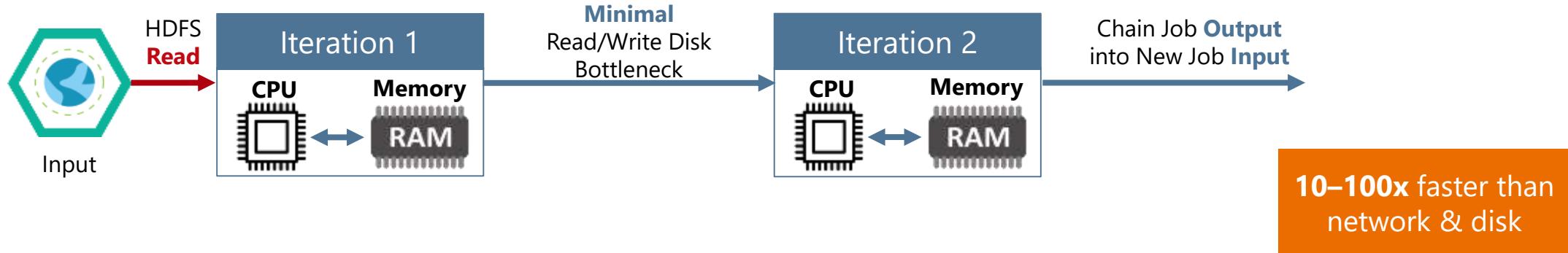


# Motivation for Apache Spark

Traditional Approach: MapReduce jobs for complex jobs, interactive query, and online event-hub processing involves lots of **(slow) disk I/O**

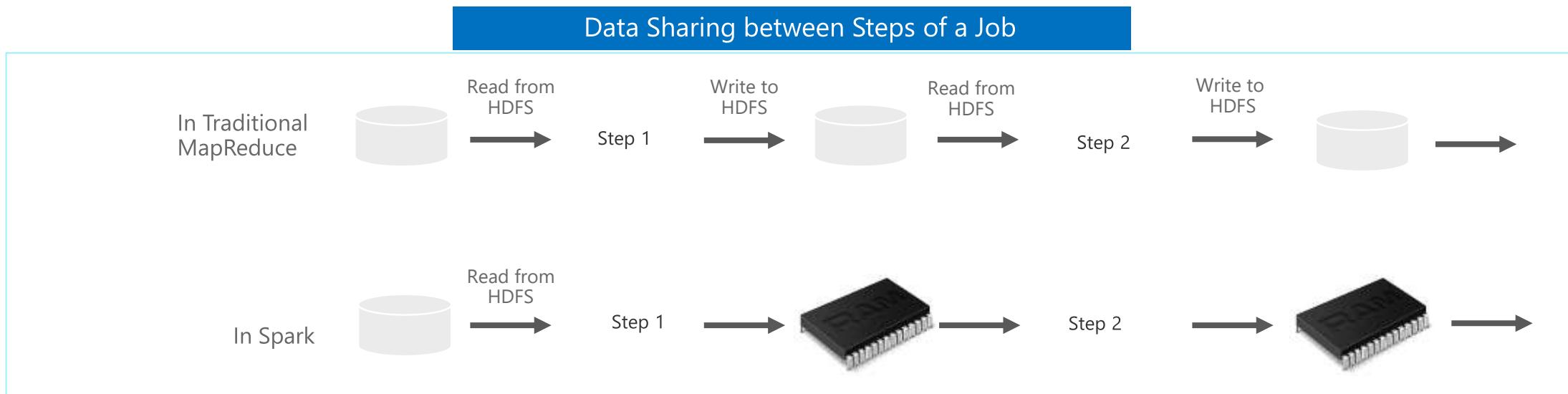


Solution: Keep data **in-memory** with a new distributed execution engine



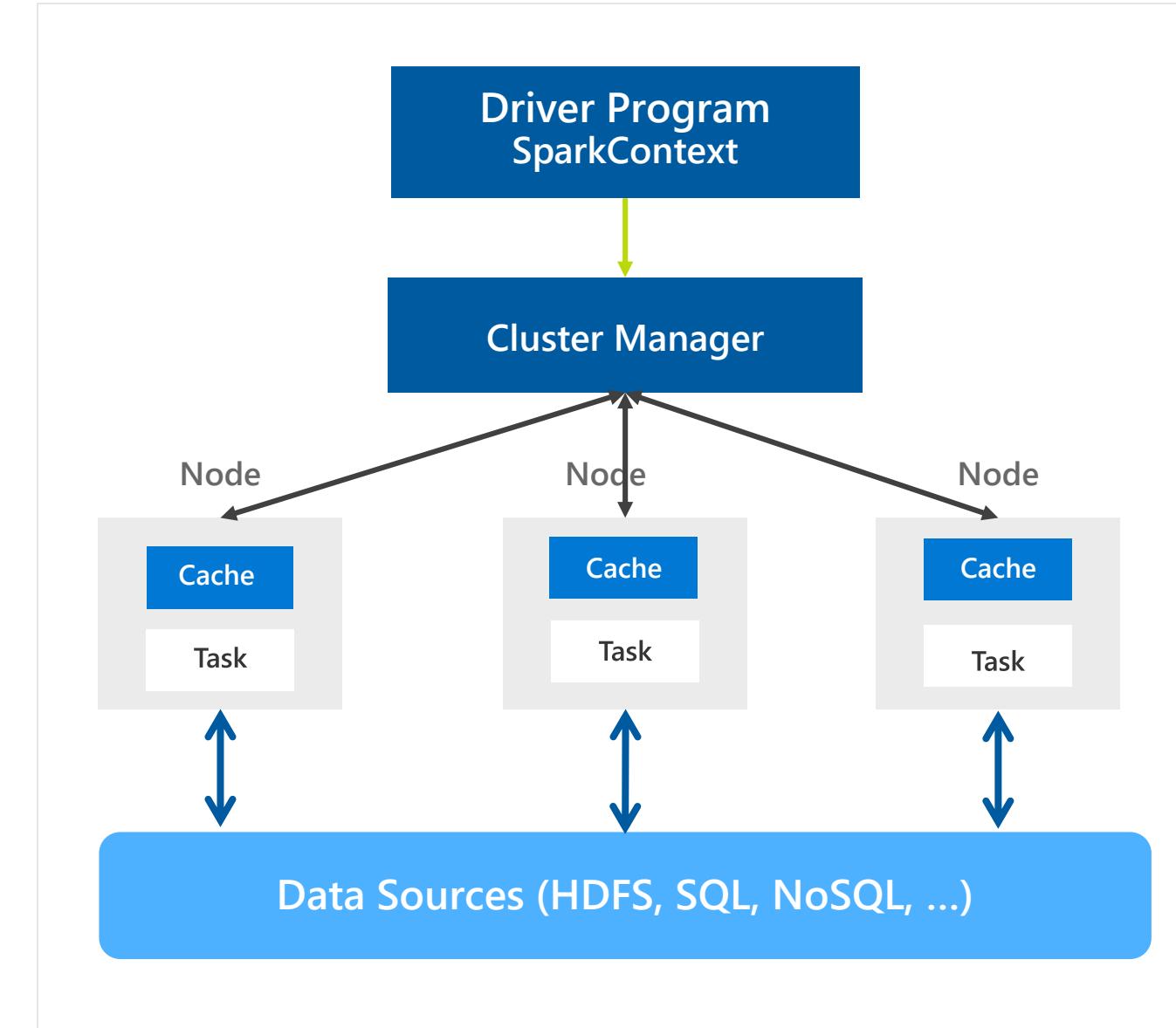
# What makes Spark fast

- **In-memory cluster computing:** Spark provides primitives for *in-memory* cluster computing. A Spark job can *load and cache* data into memory and query it repeatedly (iteratively) much quicker than disk-based systems.
- **Scala Integration:** Spark integrates into the Scala programming language, letting you manipulate distributed datasets like local collections. No need to structure everything as map and reduce operations
- **Faster Data-sharing:** Data-sharing between operations is faster as data is in-memory:
  - In (traditional) Hadoop data is shared through HDFS which is expensive. HDFS maintains three replicas.
  - Spark stores data in-memory *without any replication*.



# General Spark Cluster Architecture

- 'Driver' runs the user's 'main' function and executes the various parallel operations on the worker nodes.
- The results of the operations are collected by the driver
- The worker nodes read and write data from/to Data Sources including HDFS.
- Worker node also cache transformed data in memory as RDDs (Resilient Data Sets).
- Worker nodes and the Driver Node execute as VMs in public clouds (AWS, Google and Azure).



# Spark Component Features

## Spark SQL

- Unified data access: Query structured data sets with SQL or DataFrame APIs
- Fast, familiar query language across all your enterprise data
- Use BI tools to connect and query via JDBC or ODBC drivers

## Mlib/SparkML

- Predictive and prescriptive analytics
- Machine learning algorithms for:
  - Clustering
  - Classification
  - Regression
  - etc.
- Smart application design from pre-built, out-of-the-box statistical and algorithmic models

## Spark Streaming

- Micro-batch event processing for near-real time analytics
- e.g. Internet of Things (IoT) devices, Twitter feeds, Kafka (event hub), etc.
- Spark's engine drives some action or outputs data in batches to various data stores

## GraphX

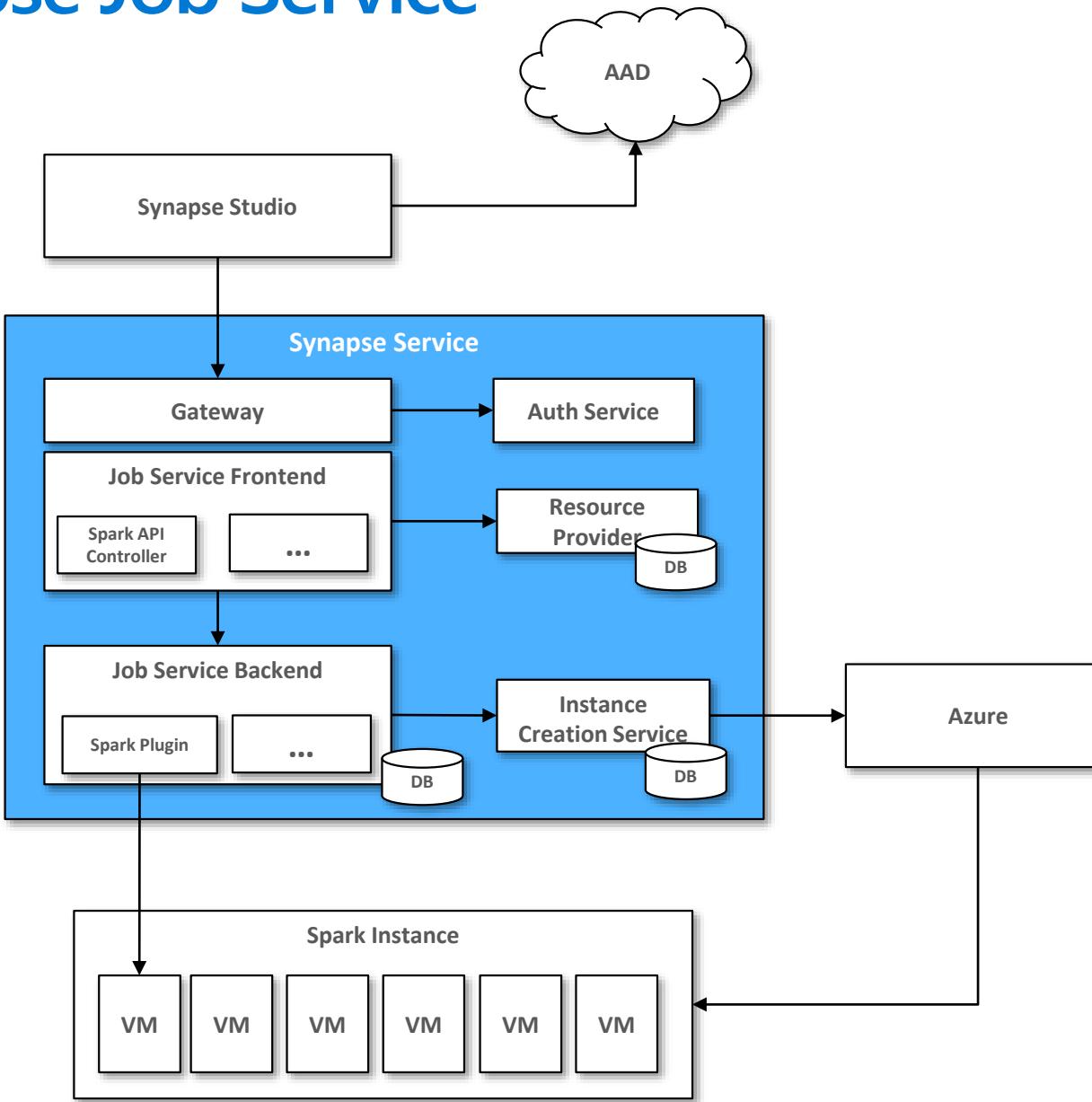
- Represent and analyze systems represented by graph nodes
- Trace interconnections between graph nodes
- Applicable to use cases in transportation, telecommunications, road networks, modeling personal relationships, social media, etc.



# Azure Synapse Apache Spark

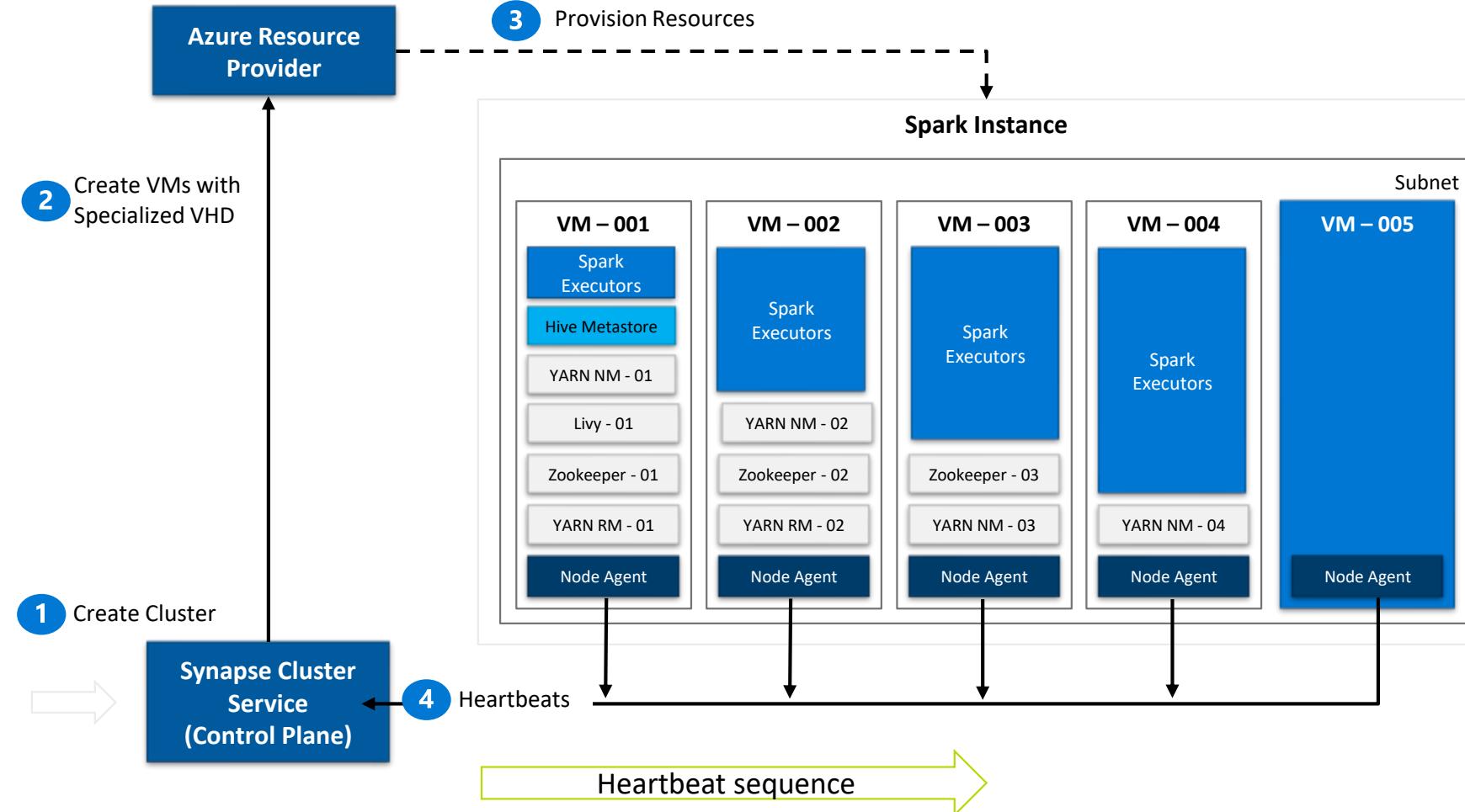
## Architecture Overview

# Synapse Job Service



- User creates Synapse Workspace and Spark pool and launches Synapse Studio.
- User attaches Notebook to Spark pool and enters one or more Spark statements (code blocks).
- The Notebook client gets user token from AAD and sends a Spark session create request to Synapse Gateway.
- Synapse Gateway authenticates the request and validates authorizations on the Workspace and Spark pool and forwards it to the Spark (Livy) controller hosted in Synapse Job Service frontend.
- The Job Service frontend forwards the request to Job Service backend that creates two jobs – one for creating the cluster and the other for creating the Spark session.
- The Job service backend contacts Synapse Resource Provider to obtain Workspace and Spark pool details and delegates the cluster creation request to Synapse Instance Service.
- Once the instance is created, the Job Service backend forwards the Spark session creation request to the Livy endpoint in the cluster.
- Once the Spark session is created the Notebook client sends Spark statements to the Job Service frontend.
- Job Service frontend obtains the actual Livy endpoint for the cluster created for the particular user from the backend and sends the statement directly to Livy for execution.

# Synapse Spark Instances



1. Synapse Job Service sends request to Cluster Service for creating BBC clusters per the description in the associated Spark pool.
2. Cluster Service sends request to Azure using Azure SDK to create VMs (required plus additional) with specialized VHD.
3. The specialized VHD contains bits for all the services that are required by the Cluster type (for e.g. Spark) with prefetch instrumentation.
4. Once VM boots up, the Node Agent sends heartbeat to Cluster Service for getting node configuration.
5. The nodes are initialized and assigned roles based on their first heartbeat.
6. Extra nodes get deleted on first heartbeat.
7. After Cluster Service considers the cluster ready, it returns the Livy endpoint to the Job Service.

# Creating a Spark pool (1 of 2)

Provision Spark Pool through Azure Portal with default settings or per requirements

Basic Settings – Minimum details required from user

Home > Synapse workspaces > euang-synapse-nov-ws - Apache Spark pools > Create Apache Spark pool

## Create Apache Spark pool

Basics \* Additional settings \* Tags Summary

Create a Synapse Analytics Apache Spark pool with your preferred configurations. Complete the Basics tab then go to Review + create to provision with smart defaults, or visit each tab to customize.

Apache Spark pool details

Name your Apache Spark pool and choose its initial settings.

Apache Spark pool name \*

Enter Apache Spark pool name

Node size family

MemoryOptimized

Node size \*

Medium (8 vCPU / 64 GB)

Autoscale \* ⓘ

Enabled  Disabled

Number of nodes \*

3  40

Only required field from user

Default Settings

# Creating a Spark pool (2 of 2) - optional

Additional Settings offer optional settings to customize Spark pool

Customize component versions, auto-pause

Import libraries by providing text file containing library name and version

Home > Synapse workspaces > euang-synapse-nov-ws - Apache Spark pools > Create Apache Spark pool

Create Apache Spark pool

Basics \* Additional settings \* Tags Summary

Customize additional configuration parameters including autoscale and component versions.

Auto-pause

Enter required settings for this Apache Spark pool, including setting auto-pause and picking versions.

Auto-pause \* ⓘ Enabled Disabled

Number of minutes idle \* 15

Component versions

Select the Apache Spark version for your Apache Spark pool.

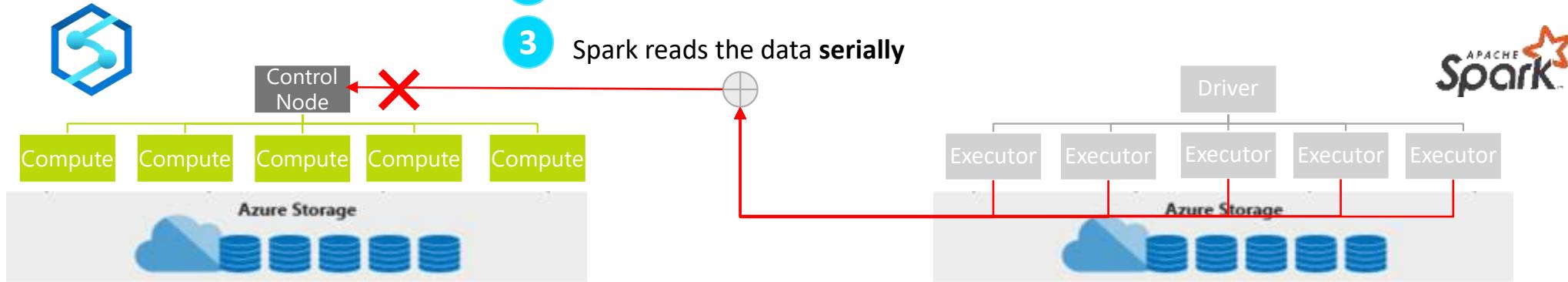
Apache Spark *	2.4
Python	3.6.1
Scala	2.11.12
Java	1.8.0_222
.NET Core	3.0
.NET for Apache Spark	0.6.0
Delta Lake	0.4.0

Packages

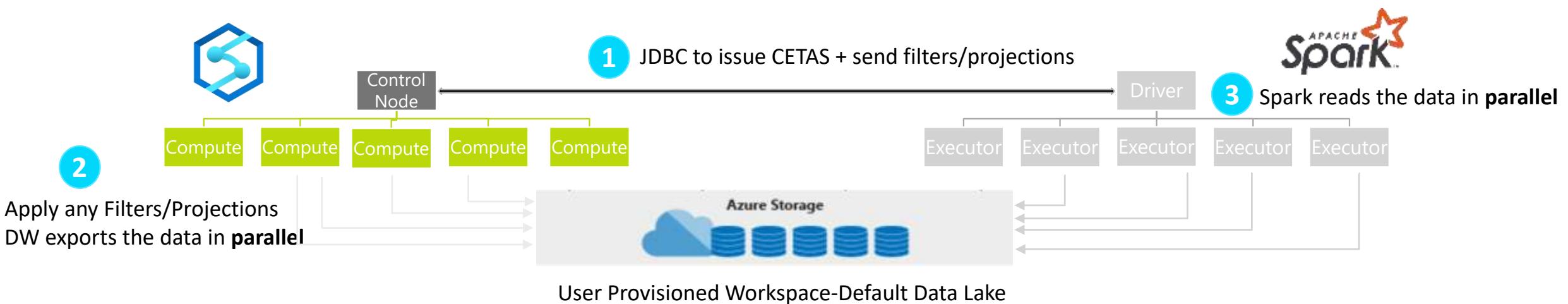
Upload environment configuration file ("PIP freeze" output).

File upload  Upload

## Existing Approach: JDBC



## New Approach: JDBC and Polybase



# Code-Behind Experience

## Existing Approach

```
val jdbcUsername = "<SQL DB ADMIN USER>"  
val jdbcPwd = "<SQL DB ADMIN PWD>"  
val jdbcHostname = "servername.database.windows.net"  
val jdbcPort = 1433  
val jdbcDatabase = "<AZURE SQL DB NAME>"  
  
val jdbc_url =  
  s"jdbc:sqlserver://${jdbcHostname}:${jdbcPort};database=${jdbcDatabase};"  
  encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.databas  
e.windows.net;loginTimeout=60;"  
  
val connectionProperties = new Properties()  
  
connectionProperties.put("user", s"${jdbcUsername}")  
connectionProperties.put("password", s"${jdbcPwd}")  
  
val sqlTableDf = spark.read.jdbc(jdbc_url, "dbo.Tbl1", connectionProperties)
```

## New Approach

```
// Construct a Spark DataFrame from SQL Pool  
var df = spark.read.sqlAnalytics("sql1.dbo.Tbl1")  
  
// Write the Spark DataFrame into SQL Pool  
df.write.sqlAnalytics("sql1.dbo.Tbl2")
```

# Create Notebook on files in storage

The screenshot illustrates the process of creating a new notebook from data stored in Azure Storage.

**Left Panel (Storage View):**

- Shows a list of Storage accounts under the "Data" category.
- The "nyctic" account is selected and highlighted in blue.
- A red box highlights the "New notebook" option in the context menu for a specific parquet file named "part-00133-tid-216-69f72c50b05d-15253-1.c000.snappy.parquet".

**Bottom Panel (Spark History Server):**

- The "Notebook 4" tab is active.
- The code cell contains the following PySpark command to read the data:

```
$ pyspark  
data_path = spark.read.load('abfss://nyctic@priangaddemosa.dfs.core.windows.net/yellow/puYear=2015/puMonth=3/part-00133-tid-216-69f72c50b05d-15253-1.c000.snappy.parquet')
```

- The cell status is "Succeeded".
- The "Job execution" section shows three tasks (Job 0, Job 1, Job 2) all completed successfully.
- The data preview shows a schema with columns: vendorId, tpepPickupDateTime, tpepDropoffDateTime, passengerCount, tripDistance, puLocationId, doLocationId, ratecodeId, storeAndFlag, paymentType, fareAmount, extra, surtax, tipAmount, tollsAmount, totalAmount.
- The data preview displays several rows of taxi trip data.

Microsoft Azure Synapse Analytics > euang-synapse-now-ws

Develop    Publish all    Validate all    Refresh    Discard all

Data Download \*    NYCTaxi\_Docs... \*    SeattleSafetyD... \*    Repo \*   

Cell 1

```

1 # Azure storage access info
2 blob_account_name = "azuresynapsedatastorage"
3 blob_container_name = "citydatacontainer"
4 blob_relative_path = "Safety/Release/city=Seattle"
5 blob_sas_token = r""

6 # Allow SPARK to read from Blob remotely
7 wasbs_path = 'wasbs://'+blob_account_name+'.blob.core.windows.net/'+blob_container_name+'/'+blob_relative_path
8 spark.conf.set('fs.azure.sas.%s.blob.core.windows.net' % (blob_container_name), blob_sas_token)
9
10 # SPARK read parquet, note that it won't load any data yet
11 seattlesafety_df = spark.read.parquet(wasbs_path)

```

Command executed in 2mins 18s 412ms by euang on 11-22-2019 00:44:52.415 -08:00

Job execution in progress Spark 1 executors 4 cores

ID	DESCRIPTION	STATUS	STAGES	TASKS	SUBMISSION TIME	DURATION
Job 0	parquet at NativeMethodAccessImpl.java:0	In progress	0/1 (active)		11/22/2019 12:44:46 AM	9m54s

View in monitoring    Spark history server

Cell 2

```

1 seattlesafety_df.createOrReplaceTempView('seattlesafety')

```

Command executed in 2s 830ms by euang on 11-22-2019 00:44:57.321 -08:00

Cell 3

```

[6] 1 display(spark.sql("SELECT * FROM seattlesafety LIMIT 10"))

```

Command executed in 12s 907ms by euang on 11-22-2019 00:45:07.212 -08:00

View    Table    Chart

datatype	dataSubtype	dateTime	category	address	latitude	longitude
Safety	911_Fire	2011-03-04T10:00:26.000Z	Aid Response	517 3rd Av	47.602172	-122.330863
Safety	911_Fire	2015-06-08T02:39:35.000Z	Trans to AMR	10044 65th Av S	47.511314	-122.292346
Safety	911_Fire	2015-06-08T21:10:52.000Z	Aid Response	Aurora Av N / N 125th St	47.719572	-122.344057
Safety	911_Fire	2007-09-17T13:03:34.000Z	Medic Response	1st Av N / Republican St	47.623272	-122.355415
Safety	911_Fire	2007-11-19T17:46:57.000Z	Aid Response	7724 Ridge Dr Ne	47.684393	-122.275254
Safety	911_Fire	2008-06-15T14:32:33.000Z	Medic Response	6940 62nd Av Ne	47.678769	-122.262227
Safety	911_Fire	2007-06-18T23:05:58.000Z	Medic Response	5107 S Myrtle St	47.538902	-122.268829
Safety	911_Fire	2005-06-08T19:25:10.000Z	Aid Response	532 Belmont Av E	47.623505	-122.324033
Safety	911_Fire	2017-03-06T19:45:36.000Z	Trans to AMR	610 1st Av N	47.624659	-122.355403
Safety	911_Fire	2017-06-25T16:21:21.000Z	Automatic Fire Alarm Read	7711 8th Av Ne	47.685157	-122.368006

Cell 4

```

[7] 1 seattlesafety_df.coalesce(1).write.csv('abfss://default@euangsynapsestorage.dfs.core.windows.net/demodata/seattlesafety', mode='overwrite')

```

View results in  
table format



Microsoft Azure Synapse Analytics > euang-synapse-nov-ws

Develop

Language: PySpark (Python)

Cell 1

```
[3] 1 # Azure storage access info
2 blob_account_name = "azuresynapsesas"
3 blob_container_name = "citydatacontainer"
4 blob_relative_path = "Safety/Release/city=Seattle"
5 blob_sas_token = r""

6 # Allow SPARK to read from Blob remotely
7 wasbs_path = "wasbs://<blob_container_name>.<blob_account_name>.blob.core.windows.net/<blob_relative_path>"
8 spark.conf.set( 'fs.azure.sas.<blob_container_name>.<blob_account_name>', blob_sas_token)
9
10 # SPARK read parquet, note that it won't load any data yet
11 seasafety_df = spark.read.parquet(wasbs_path)

Command executed in 2m 16s 412ms by euang on 11-22-2019 00:14:32.415 -08:00
```

Job execution in progress: Spark 1 executors 4 cores

ID	DESCRIPTION	STATUS	STAGES	TAGS	SUBMISSION TIME	DURATION
Job 0	parquet at NativeMethodAccessorsImpl.java:0	In progress	0/1 (1 active)		11/22/2019, 12:44:46 AM	13m43s

Cell 2

```
[4] 1 seasafety_df.createOrReplaceTempView("seattlesafety")
```

Command executed in 2s 839ms by euang on 11-22-2019 00:16:37.321 -08:00

Cell 3

```
[5] 1 display(spark.sql('SELECT * FROM seattlesafety'))
```

Command executed in 11s 320ms by euang on 11-22-2019 00:38:21.241 -08:00

View Table Chart

SQL support

View results in chart format

A red arrow points from the text "View results in chart format" to the "Chart" tab in the interface.

A pie chart titled "Aid Response" is displayed, showing the distribution of various incident types. The chart includes labels for "Medic Response", "Tran to AMR", "longitude", and "Aid Response". A legend on the left lists categories such as "Automatic Fire Alarm False", "Medic Response, 7 per Rule", "Aid Response Yellow", "AVVI - Motor Vehicle Incident", "Medic Response, 6 per Rule", "Motor Vehicle Accident", "Automatic Medical Alarm", "TRED 1 Unit", "Auto Fire Alarm", "Automatic Fire Alarm Red", and "Medic Response".

Chart type: pie chart  
X axis column: category  
Y axis column: longitude  
Aggregation: COUNT  
Y axis label: Time  
X axis label: category

Cell 4

```
[6] 1 seasafety_df.coalesce(1).write.csv('abfss://default@euangsynapse-novstorage.dfs.core.windows.net/desodata/seattlesafety', mode='overwrite')
```

Microsoft Azure | Synapse Analytics | euang-synapse-nov-2023 | Search resources

Develop + ×

Data Download... NYCTaxi\_Docs\_...

Cell 11 Run all Publish Attach to Enter Spark session Language PySpark (Python)

```
11 # Creating a temp table allows easier manipulation during the session, they are not persisted between sessions;
12 # for that write the data to storage like above;
13 sampled_taxi_df.createOrReplaceTempView("nytaxi")
```

**Exploratory Data Analysis**

Look at the data and evaluate its suitability for use in a model. do this via some basic charts focused on tip values and relationships.

Cell 8

```
1 #The charting package needs a Pandas dataframe or numpy array do the conversion
2 sampled_taxi_pd_df = sampled_taxi_df.toPandas()
3
4 # Look at tips by amount count histogram
5 ax1 = sampled_taxi_pd_df['tipAmount'].plot(kind='hist', bins=25, facecolor='lightblue')
6 ax1.set_title('Tip amount distribution')
7 ax1.set_xlabel('Tip Amount ($)')
8 ax1.set_ylabel('Counts')
9 plt.subtitle('')
10 plt.show()
11
12 # How many passengers tip'd by various amounts
13 ax2 = sampled_taxi_pd_df.boxplot(column=['tipAmount'], by=['passengerCount'])
14 ax2.set_title('Tip amount by Passenger count')
15 ax2.set_xlabel('Passenger count')
16 ax2.set_ylabel('Tip Amount ($)')
17 plt.subtitle('')
18 plt.show()
19
20 # Look at the relationship between fare and tip amounts
21 ax = sampled_taxi_pd_df.plot(kind='scatter', x='fareAmount', y='tipAmount', c='blue', alpha = 0.1, s=2.5*(sampled_taxi_pd_df['passengerCount']))
22 ax.set_xlabel('Fare Amount ($)')
23 ax.set_ylabel('Tip Amount ($)')
24 plt.axis([-2, 80, -2, 20])
25 plt.subtitle('')
26 plt.show()
```

Tip amount distribution

Tip amount by Passenger count

Exploratory data analysis with graphs – histogram, boxplot etc

# Library Management - Python

## Overview

Customers can add new python libraries at Spark pool level

## Benefits

Input requirements.txt in simple pip freeze format

Add new libraries to your cluster

Update versions of existing libraries on your cluster

Libraries will get installed for your Spark pool during cluster creation

Ability to specify different requirements file for different pools within the same workspace

## Constraints

The library version must exist on PyPI repository

Version downgrade of an existing library not allowed

## In the Portal

Specify the new requirements while creating Spark Pool in Additional Settings blade

Microsoft Azure (Preview)    Restore default configuration    Report a bug    Search resources, services, and data

Home > nushuklasynapsewestus2 > Create Apache Spark pool

Create Apache Spark pool

Enter required settings for this Apache Spark pool, including setting auto-pause and picking versions.

Auto-pause \* ⓘ

Enabled    Disabled

Number of minutes idle \*

15

Component versions

Select the Apache Spark version for your Apache Spark pool.

Apache Spark *	2.4
Python	3.6.1
Scala	2.11.12
Java	1.8.0_222
.NET Core	3.0
.NET for Apache Spark	0.6.0
Delta Lake	0.4.0

Packages

Upload environment configuration file ("PIP freeze" output).

File upload

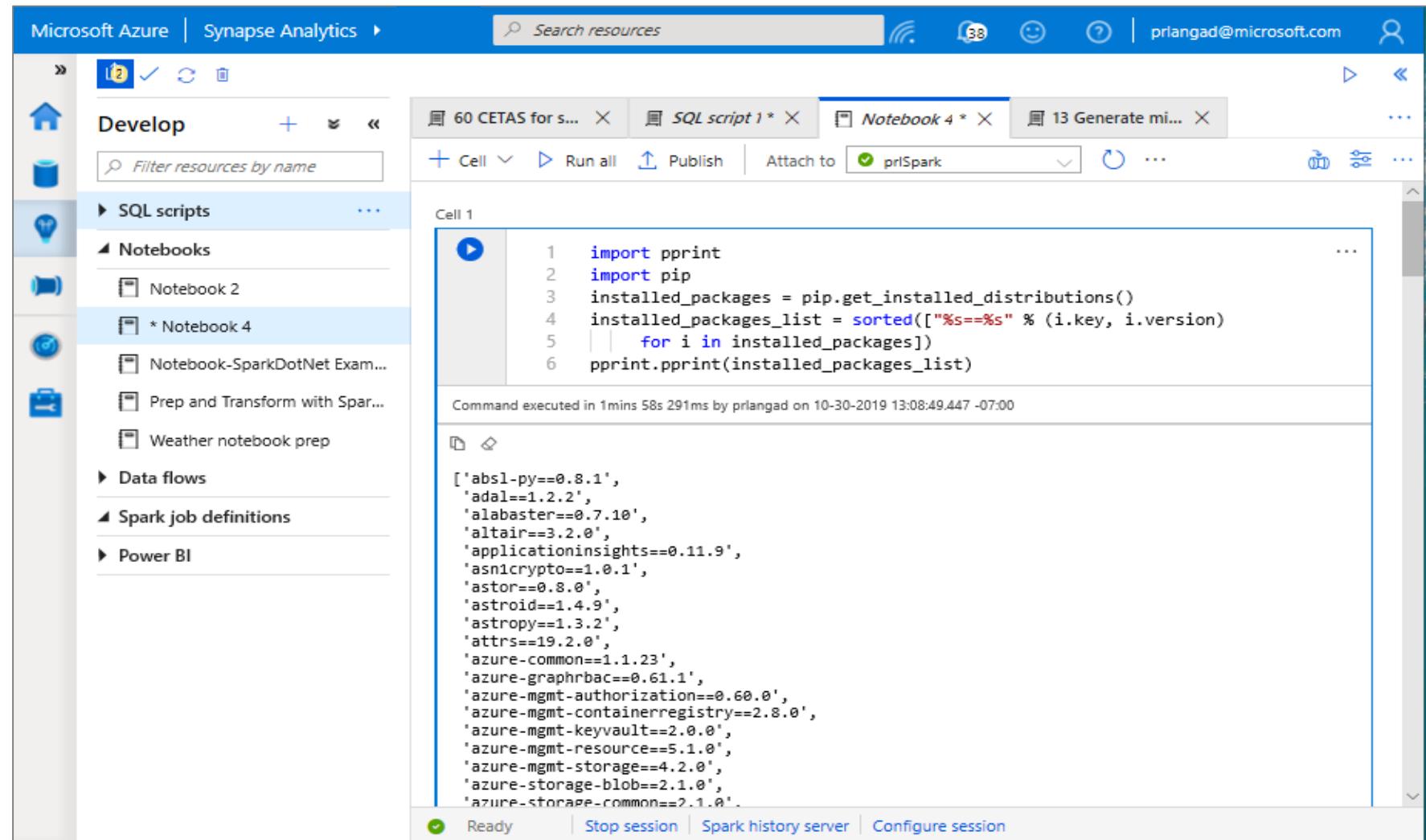
"requirements.txt"

Upload

Review + create    < Previous    Next: Tags >

# Library Management - Python

Get list of installed libraries with version information



The screenshot shows the Microsoft Azure Synapse Analytics interface. On the left, the 'Develop' sidebar is open, showing a list of resources including 'SQL scripts', 'Notebooks', 'Data flows', 'Spark job definitions', and 'Power BI'. A 'Notebook 4' item is selected. In the main workspace, there are four tabs at the top: '60 CETAS for s...', 'SQL script 1 \* X', 'Notebook 4 \* X' (which is active), and '13 Generate mi... X'. Below the tabs, a toolbar includes 'Cell', 'Run all', 'Publish', 'Attach to', and session controls. A session named 'priSpark' is attached. A code cell labeled 'Cell 1' contains the following Python script:

```
1 import pprint
2 import pip
3 installed_packages = pip.get_installed_distributions()
4 installed_packages_list = sorted(["%s==%s" % (i.key, i.version)
5 | | for i in installed_packages])
6 pprint.pprint(installed_packages_list)
```

The output of the cell shows a long list of installed Python packages and their versions:

```
['absl-py==0.8.1',
 'adal==1.2.2',
 'alabaster==0.7.10',
 'altair==3.2.0',
 'applicationinsights==0.11.9',
 'asn1crypto==1.0.1',
 'astor==0.8.0',
 'astroid==1.4.9',
 'astropy==1.3.2',
 'attrs==19.2.0',
 'azure-common==1.1.23',
 'azure-graphrbac==0.61.1',
 'azure-mgmt-authorization==0.60.0',
 'azure-mgmt-containerregistry==2.8.0',
 'azure-mgmt-keyvault==2.0.0',
 'azure-mgmt-resource==5.1.0',
 'azure-mgmt-storage==4.2.0',
 'azure-storage-blob==2.1.0',
 'azure-storage-common==2.1.0']
```

At the bottom of the cell, it says 'Command executed in 1mins 58s 291ms by prlangad on 10-30-2019 13:08:49.447 -07:00'. The status bar at the bottom of the interface shows 'Ready'.

# Spark ML Algorithms

## Spark ML Algorithms

Classification and Regression	<ul style="list-style-type: none"><li>• Linear Models (SVMs, logistic regression, linear regression)</li><li>• Naïve Bayes</li><li>• Decision Trees</li><li>• Ensembles of trees (Random Forest, Gradient-Boosted Trees)</li><li>• Isotonic regression</li></ul>
Clustering	<ul style="list-style-type: none"><li>• k-means and streaming k-means</li><li>• Gaussian mixture</li><li>• Power iteration clustering (PIC)</li><li>• Latent Dirichlet allocation (LDA)</li></ul>
Collaborative Filtering	<ul style="list-style-type: none"><li>• Alternating least squares (ALS)</li></ul>
Dimensionality Reduction	<ul style="list-style-type: none"><li>• SVD</li><li>• PCA</li></ul>
Frequent Pattern Mining	<ul style="list-style-type: none"><li>• FP-growth</li><li>• Association rules</li></ul>
Basic Statistics	<ul style="list-style-type: none"><li>• Summary statistics</li><li>• Correlations</li><li>• Stratified sampling</li><li>• Hypothesis testing</li><li>• Random data generation</li></ul>

# Microsoft Machine Learning for Apache Spark

v1.0-rc

Microsoft's Open Source  
Contributions to Apache Spark



Distributed  
Machine Learning



Fast Model  
Deployment



Microservice  
Orchestration



Multilingual Binding  
Generation

[www.aka.ms/spark](http://www.aka.ms/spark)

 Azure/mmlspark

## Synapse Notebook: Connect to AML workspace

The screenshot shows a Microsoft Azure Synapse Analytics notebook interface. The left sidebar shows 'Develop' resources: SQL scripts, Notebooks, Data flows, Spark job definitions, and Power BI. The main area has a title bar with 'Search resources' and user info 'balapv@microsoft.com'. Below is a toolbar with 'Cell', 'Run all', 'Publish', 'Attach to', 'Language' set to 'PySpark Python', and other controls.

**Check the Azure ML Core SDK Version to Validate Your Installation**

**Cell 3**

```
[9] 1 import azureml.core  
2 print("SDK Version:", azureml.core.VERSION)
```

Command executed in 1s 258ms by balapv on 11-12-2019 14:41:52.805 -00:00

SDK Version: 1.0.69

**Connect to Azure Workspace**

**Cell 5**

```
[6] 1 ## Import the Workspace class and check the Azure ML SDK version.  
2 from azureml.core import Workspace  
3  
4 ws = ws = Workspace(subscription_id = "6568575d-fa96-4e7d-95fb-f962e74efd7a",  
5                      resource_group = "balapv-synapse-rg", workspace_name = "AML-WS-synapse")  
6  
7 print(ws.name, ws.location, ws.resource_group, sep='\t')
```

Command executed in 3s 909ms by balapv on 11-12-2019 14:41:55.491 -00:00

AML-WS-synapse westus2 balapv-synapse-rg

**Cell 6**

```
[7] 1 # import modules  
2 import azureml.core  
3 import pandas as pd  
4 from azureml.core.authentication import ServicePrincipalAuthentication  
5 from azureml.core.workspace import Workspace  
6 from azureml.core.experiment import Experiment
```

Running Stop session Spark history server Configure session

**Simple code to connect workspace** →

## Synapse Notebook: Configure AML job to run on Synapse

The screenshot shows the Microsoft Azure Synapse Analytics notebook interface. On the left, the sidebar lists resources: Develop, SQL scripts, Notebooks (with 'automl\_synapse\_local\_regr...' selected), Data flows, Spark job definitions, and Power BI. The main area has tabs for Train, Cell, Run all, Publish, Attach to (set to sparkcompute), Language (PySpark (Python)), and PySpark (Python). The 'Train' tab is active, displaying configuration parameters for an AutoMLConfig object:

Property	Description
<code>task</code>	classification or regression
<code>primary_metric</code>	This is the metric that you want to optimize.
<code>iteration_timeout_minutes</code>	Time limit in minutes for each iteration.
<code>iterations</code>	Number of iterations. In each iteration AutoML trains a specific pipeline with the data.
<code>X</code>	(sparse) array-like, shape = <code>n_samples</code> , <code>n_features</code>
<code>y</code>	(sparse) array-like, shape = <code>n_samples</code> , Multi-class targets.
<code>enable_onnx_compatible_models</code>	Enable the ONNX compatible models in the experiment.
<code>path</code>	Relative path to the project folder. AutoML stores configuration files for the experiment under this folder. You can specify a new empty folder.

Below this, 'Cell 13' contains the following Python code:

```

1  automl_config = AutoMLConfig(task = 'regression',
2                                debug_log = 'automl_errors.log',
3                                primary_metric = 'normalized_root_mean_squared_error',
4                                iteration_timeout_minutes = 10,
5                                iterations = 20,
6                                preprocess = True,
7                                n_cross_validations = 2,
8                                max_concurrent_iterations = 2, #spark compute size
9                                verbosity = logging.INFO,
10                               spark_context=sc, #spark related
11                               enable_onnx_compatible_models=True, # This will generate ONNX compatible models.
12                               cache_store=True,
13                               X = X_train,
14                               y = y_train)

```

A red arrow points from the text 'Configuration parameters' to the code block. At the bottom, a note says: 'Call the `submit` method on the experiment object and pass the run configuration. Execution of local runs is synchronous. Depending on the data and the number of iterations this can run for a while. In this example, we specify `show_output = True` to print currently running iterations to the console.'

## Synapse Notebook: Run AML job

Microsoft Azure | Synapse Analytics > synapsews4aml | Search resources | Show notifications | balapv@microsoft.com

Publish all | Validate all | Refresh | Discard all

Develop | Filter resources by name

SQL scripts | Notebooks | automl\_synapse\_local\_regression.ipynb | Data flows | Spark job definitions | Power BI

Run AutoML job

Cell 15

```
local_run = experiment.submit(automl_config, show_output = True)
```

Command executed in 12mins 34s 972ms by balapv on 11-12-2019 15:17:53.089 -08:00

Running an experiment on spark cluster: automl-local-regression-Synapse.  
Parent Run ID: AutoML\_ad8600ab-a1ab-4b6b-b233-059d969e0a0e

\*\*\*\*\*  
ITERATION: The iteration being evaluated.  
PIPELINE: A summary description of the pipeline being evaluated.  
DURATION: Time taken for the current iteration.  
METRIC: The result of computing score on the fitted pipeline.  
BEST: The best observed score thus far.  
\*\*\*\*\*

ITERATION	PIPELINE	DURATION	METRIC	BEST
1	StandardScalerWrapper ElasticNet	0:00:38	0.0021	0.0021
2	StandardScalerWrapper ElasticNet	0:00:32	0.0054	0.0021
0	StandardScalerWrapper ElasticNet	0:01:20	0.0004	0.0004
4	StandardScalerWrapper RandomForest	0:00:33	0.0179	0.0004
3	StandardScalerWrapper ElasticNet	0:00:36	0.0036	0.0004
5	StandardScalerWrapper LightGBM	0:00:28	0.0109	0.0004
6	MaxAbsScaler DecisionTree	0:00:34	0.0168	0.0004
7	MaxAbsScaler RandomForest	0:00:41	0.0184	0.0004
8	MaxAbsScaler DecisionTree	0:01:05	0.0077	0.0004
9	MaxAbsScaler DecisionTree	0:00:48	0.0086	0.0004
10	StandardScalerWrapper DecisionTree	0:00:39	0.0058	0.0004
11	MaxAbsScaler DecisionTree	0:00:45	0.0096	0.0004
13	MaxAbsScaler ExtremeRandomTrees	0:00:47	0.0147	0.0004
12	MaxAbsScaler ExtremeRandomTrees	0:01:54	0.0096	0.0004
14	StandardScalerWrapper ElasticNet	0:00:39	0.0027	0.0004
15	StandardScalerWrapper ElasticNet	0:00:54	0.0010	0.0004
16	StandardScalerWrapper ElasticNet	0:00:48	0.0023	0.0004
17	MaxAbsScaler ElasticNet	0:00:31	0.0239	0.0004
18	StandardScalerWrapper ElasticNet	0:00:53	0.0014	0.0004
19	VotingEnsemble	0:01:59	0.0004	0.0004

Get Azure Portal URL for Monitoring Runs

Running | Stop session | Spark history server | Configure session

ML job execution result

**Industry-leading security  
and compliance**

# Enterprise-grade security



Defense-in-Depth

# Industry-leading compliance



ISO 27001



SOC 1 Type 2



SOC 2 Type 2



PCI DSS Level 1



Cloud Controls Matrix



ISO 27018



Content Delivery and Security Association



Shared Assessments



FedRAMP JAB P-ATO



HIPAA / HITECH



FIPS 140-2



21 CFR Part 11



FERPA



DISA Level 2



CJIS



IRS 1075



ITAR-ready



Section 508 VPAT



European Union Model Clauses



EU Safe Harbor



United Kingdom G-Cloud



China Multi Layer Protection Scheme



China GB 18030



China CCCPPF



Singapore MTCS Level 3



Australian Signals Directorate



New Zealand GCIO



Japan Financial Services



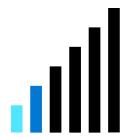
ENISA IAF

# Threat Protection - Business requirements



**How do we enumerate and track potential SQL vulnerabilities?**

To mitigate any security misconfigurations before they become a serious issue.



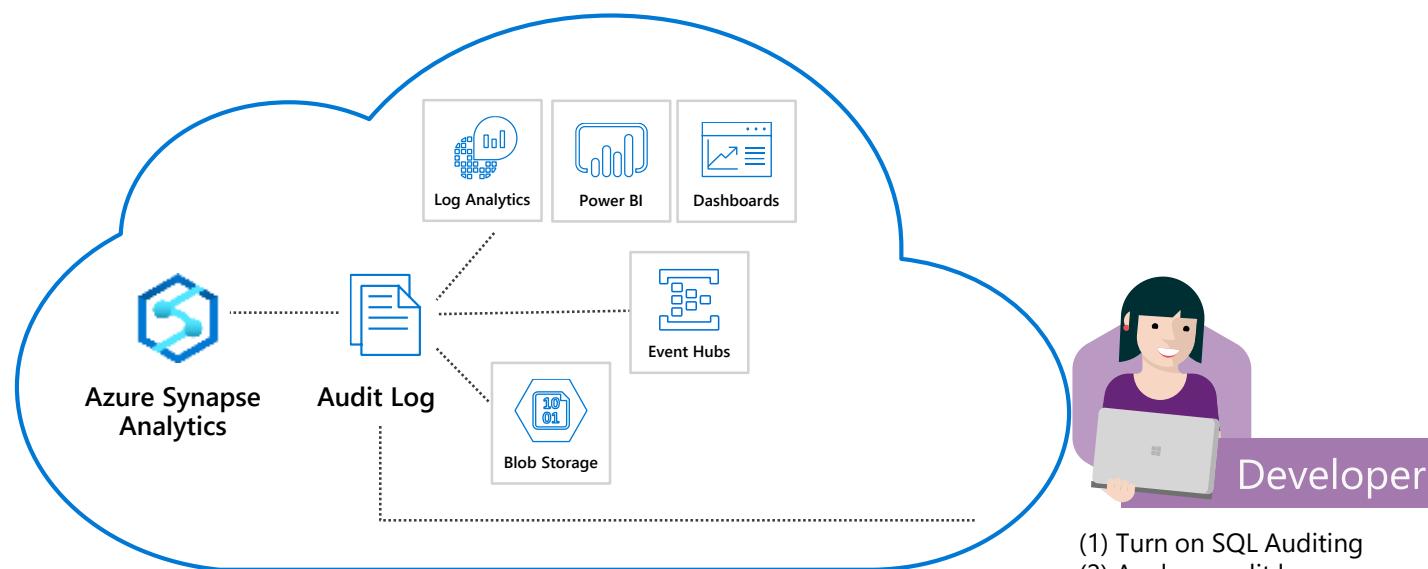
**How do we discover and alert on suspicious database activity?**

To detect and resolve any data exfiltration or SQL injection attacks.



# SQL auditing in Azure Log Analytics and Event Hubs

## Gain insight into database audit log

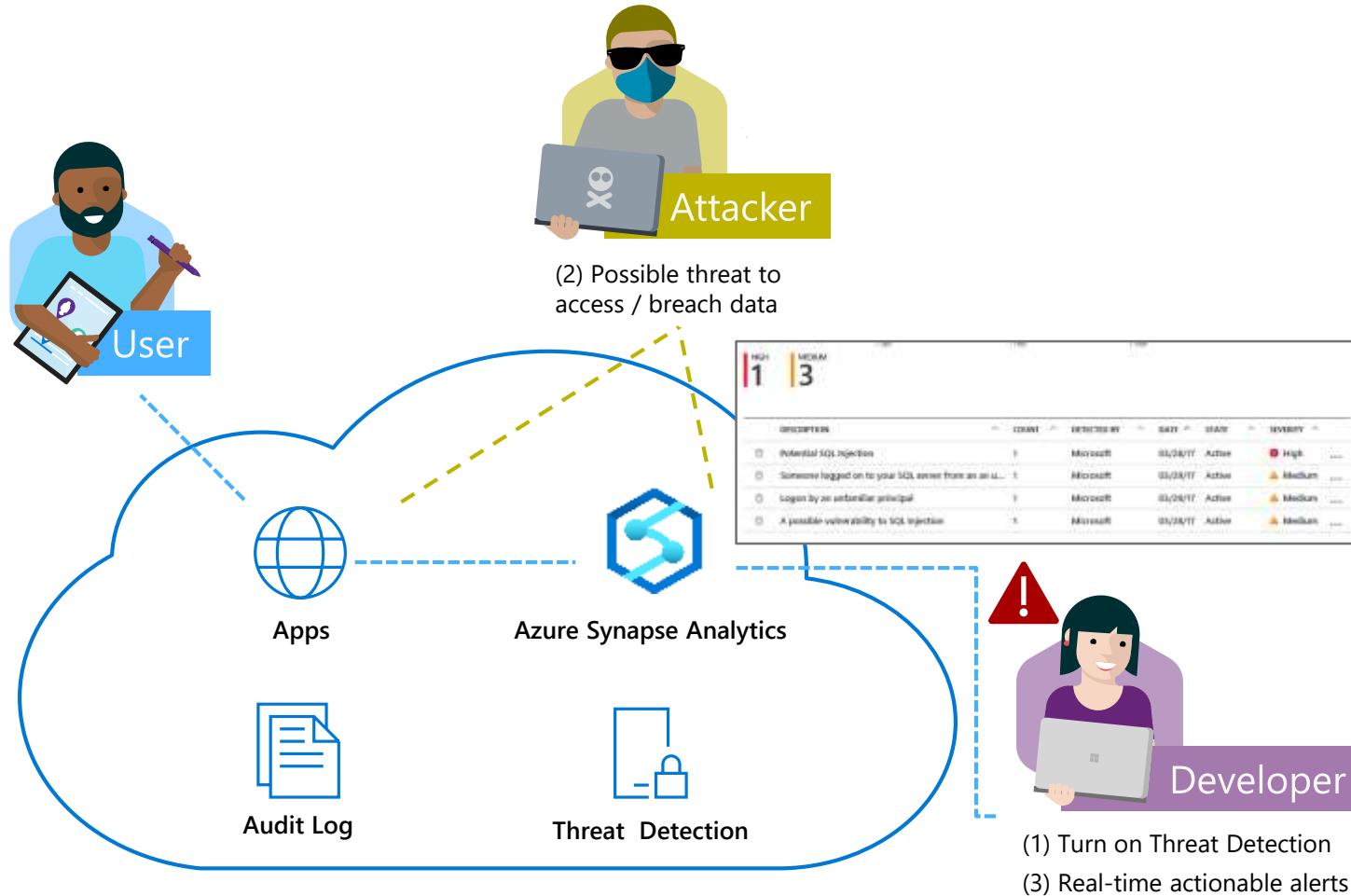


A screenshot of the Azure Log Analytics workspace titled 'Logs'. It displays a table of audit log entries. The columns include TimeGenerated, RemoteComputerName, Statement, and others. One row is highlighted in blue, showing a query: 'SELECT \* FROM master..syscolumns WHERE name = 'password''. The interface includes various filters and search options at the top.

- ✓ Configurable via audit policy
- ✓ SQL audit logs can reside in
  - Azure [Storage account](#)
  - Azure Log Analytics
  - Azure Event Hubs
- ✓ Rich set of tools for
  - [Investigating](#) security alerts
  - [Tracking](#) access to sensitive data

# SQL threat detection

## Detect and investigate anomalous database activity



- ✓ Detects potential SQL injection attacks
- ✓ Detects unusual access & data exfiltration activities
- ✓ Actionable alerts to investigate & remediate
- ✓ View alerts for your entire Azure tenant using Azure Security Center

# SQL Data Discovery & Classification

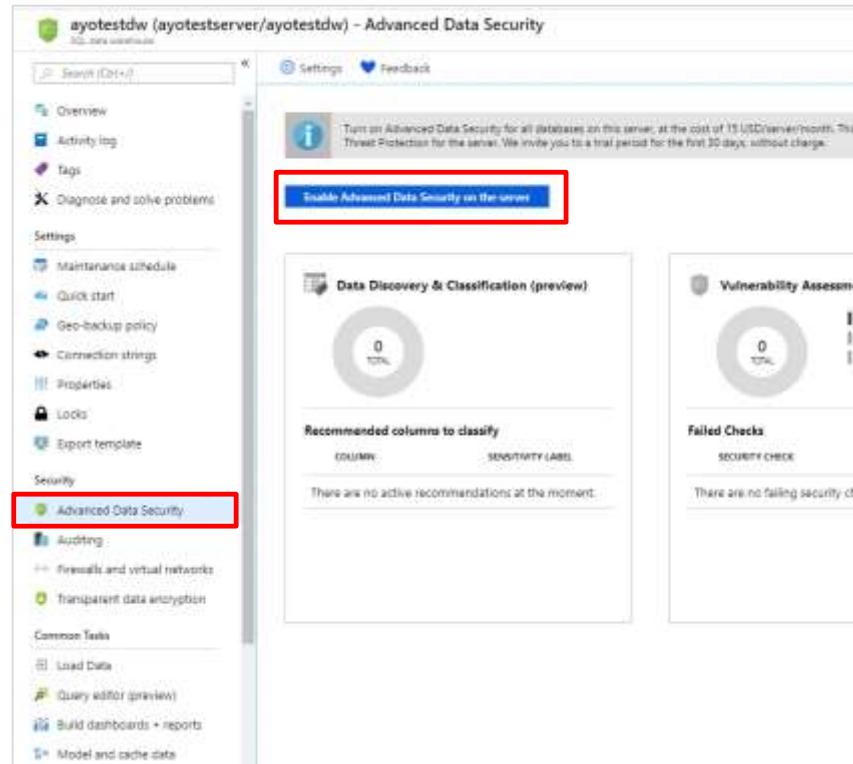
## Discover, classify, protect and track access to sensitive data

The screenshot shows the Azure portal interface for SQL Data Discovery & Classification. The main dashboard displays two donut charts: one for 'Label distribution' (10 columns) and another for 'Information type distribution' (10 columns). Below these charts, there's a table with columns: Schema, Table, Column, Information Type, and Sensitivity Label. A separate window titled 'Settings - Information protection' is open, showing a list of sensitivity labels with their descriptions. The labels include 'General', 'Confidential', 'Confidential - GDPR', 'Highly confidential', and 'Highly confidential - GDPR'. The 'Confidential' label is selected.

- ✓ Automatic **discovery** of columns with sensitive data
- ✓ Add **persistent sensitive data labels**
- ✓ Audit and detect access to the sensitive data
- ✓ Manage labels for your entire Azure tenant using Azure Security Center

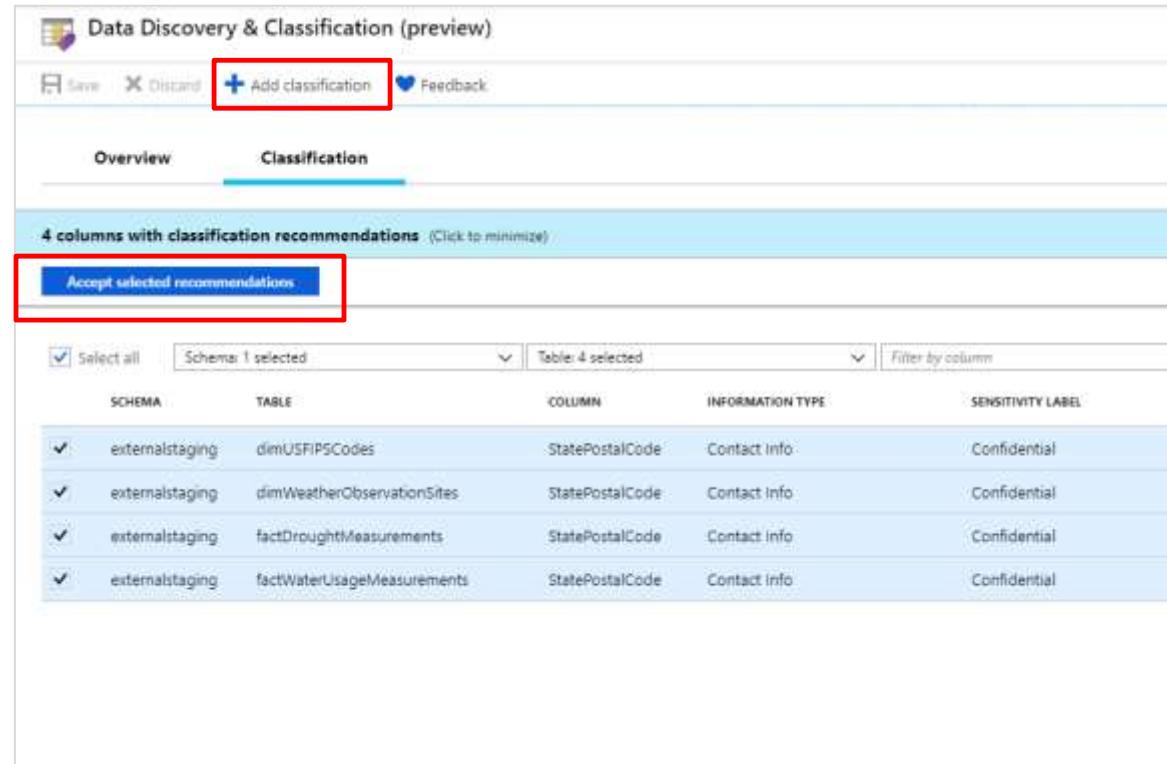
# SQL Data Discovery & Classification - setup

**Step 1:** Enable Advanced Data Security on the logical SQL Server



The screenshot shows the 'ayotestdw (ayotestserver/ayotestdw) - Advanced Data Security' blade. On the left, there's a sidebar with various navigation items like Overview, Activity log, and Diagnose and solve problems. Under the 'Security' section, 'Advanced Data Security' is selected and highlighted with a red box. In the main area, there's a button labeled 'Enable Advanced Data Security on the server' which is also highlighted with a red box. Below it, there are sections for 'Data Discovery & Classification (preview)' and 'Vulnerability Assessment', both showing zero results.

**Step 2:** Use recommendations and/or manual classification to classify all the sensitive columns in your tables



The screenshot shows the 'Data Discovery & Classification (preview)' blade. At the top, there are buttons for Save, Discard, and Add classification (the latter is highlighted with a red box). Below that, there are tabs for Overview and Classification, with Classification selected. A message indicates '4 columns with classification recommendations'. A button labeled 'Accept selected recommendations' is highlighted with a red box. Below this, there's a table showing four rows of recommendations. The columns are SCHEMA, TABLE, COLUMN, INFORMATION TYPE, and SENSITIVITY LABEL. All four rows show 'externalstaging' as the schema, 'dimUSPostalCodes' or 'factDroughtMeasurements' as the table, 'StatePostalCode' as the column, 'Contact Info' as the information type, and 'Confidential' as the sensitivity label. There are checkboxes for 'Select all' and dropdowns for 'Schema 1 selected' and 'Table 4 selected'.

SCHEMA	TABLE	COLUMN	INFORMATION TYPE	SENSITIVITY LABEL
✓ externalstaging	dimUSPostalCodes	StatePostalCode	Contact Info	Confidential
✓ externalstaging	dimWeatherObservationSites	StatePostalCode	Contact Info	Confidential
✓ externalstaging	factDroughtMeasurements	StatePostalCode	Contact Info	Confidential
✓ externalstaging	factWaterUsageMeasurements	StatePostalCode	Contact Info	Confidential

# SQL Data Discovery & Classification – audit sensitive data access

**Step 1:** Configure auditing for your target Data warehouse. This can be configured for just a single data warehouse or all databases on a server.

The screenshot shows the Azure portal interface for configuring auditing. In the left sidebar, under 'Security', the 'Auditing' option is selected and highlighted with a red box. On the main page, the 'Audit' switch is turned 'ON'. Below it, there are sections for 'Audit log destination' (set to 'Storage'), 'Storage details' (showing 'azurite://localhost:10000/auditlog'), and 'Event hub' and 'Log Analytics' options. A note at the top states: 'If both Auditing is enabled on the server, it will always apply to the database, regardless of the database settings.'

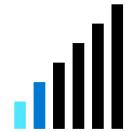
**Step 2:** Navigate to audit logs in storage account and download 'xel' log files to local machine.

The screenshot shows the Azure portal's storage account interface. Under the 'sqldbauditlogs' container, a blob named '01\_34\_30\_090.xel' is listed. The blob was uploaded on April 1, 2019, at 6:34:31 PM. The blob type is 'Append blob' and its size is 7.5 KB. The 'LEASE STATE' column shows 'Available'.

**Step 3:** Open logs using extended events viewer in SSMS. Configure viewer to include 'data\_sensitivity\_information' column

The screenshot shows the Extended Events Viewer in SQL Server Management Studio (SSMS). The table '01\_34\_30\_090.xel' contains 2475 events. The columns are: name, timestamp, affected\_rows, application\_name, client\_ip, data\_sensitivity\_information, and database\_name. A red box highlights the 'data\_sensitivity\_information' column in the table header. Below, a specific event row is expanded, showing detailed fields like action\_id, additional\_information, affected\_rows, application\_name, auth\_scheme\_version, client\_ip, connection\_id, database\_name, database\_principal\_id, database\_principal\_name, duration\_milliseconds, event\_time, host\_name, is\_column\_permission, object\_id, object\_name, and user\_name. The 'data\_sensitivity\_information' field is also highlighted with a red box in this expanded view.

# Network Security - Business requirements

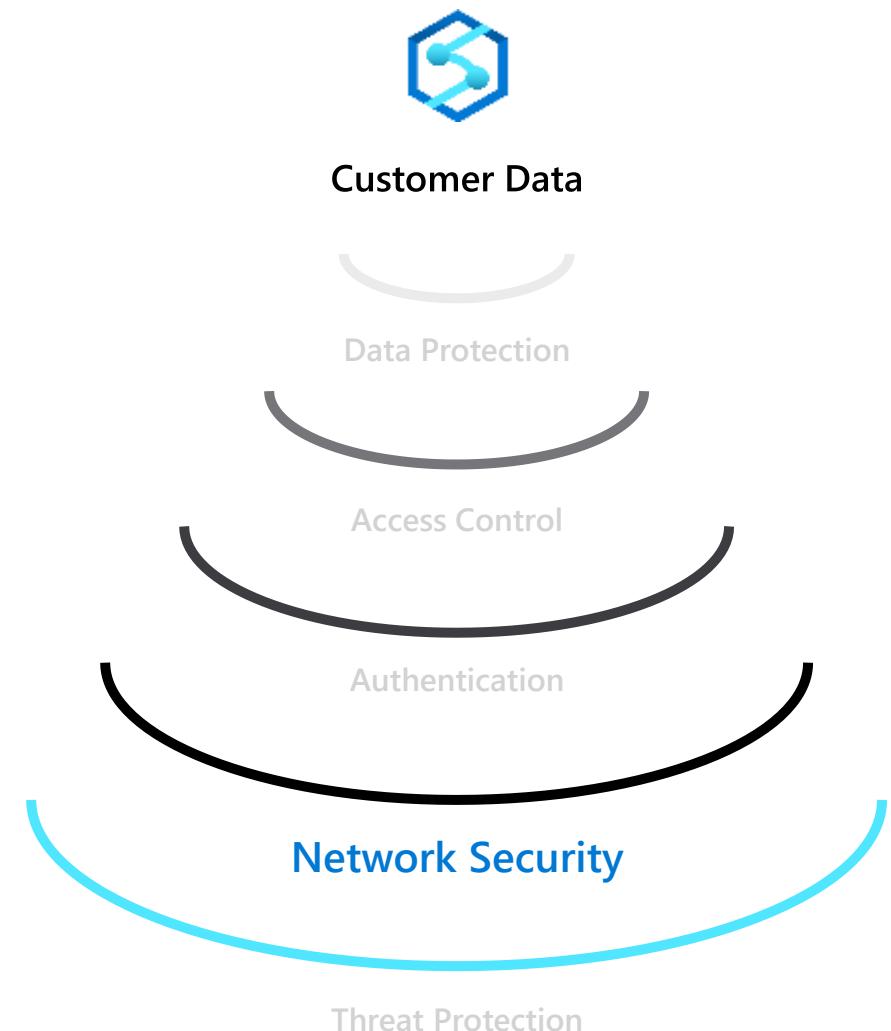


## How do we implement network isolation?

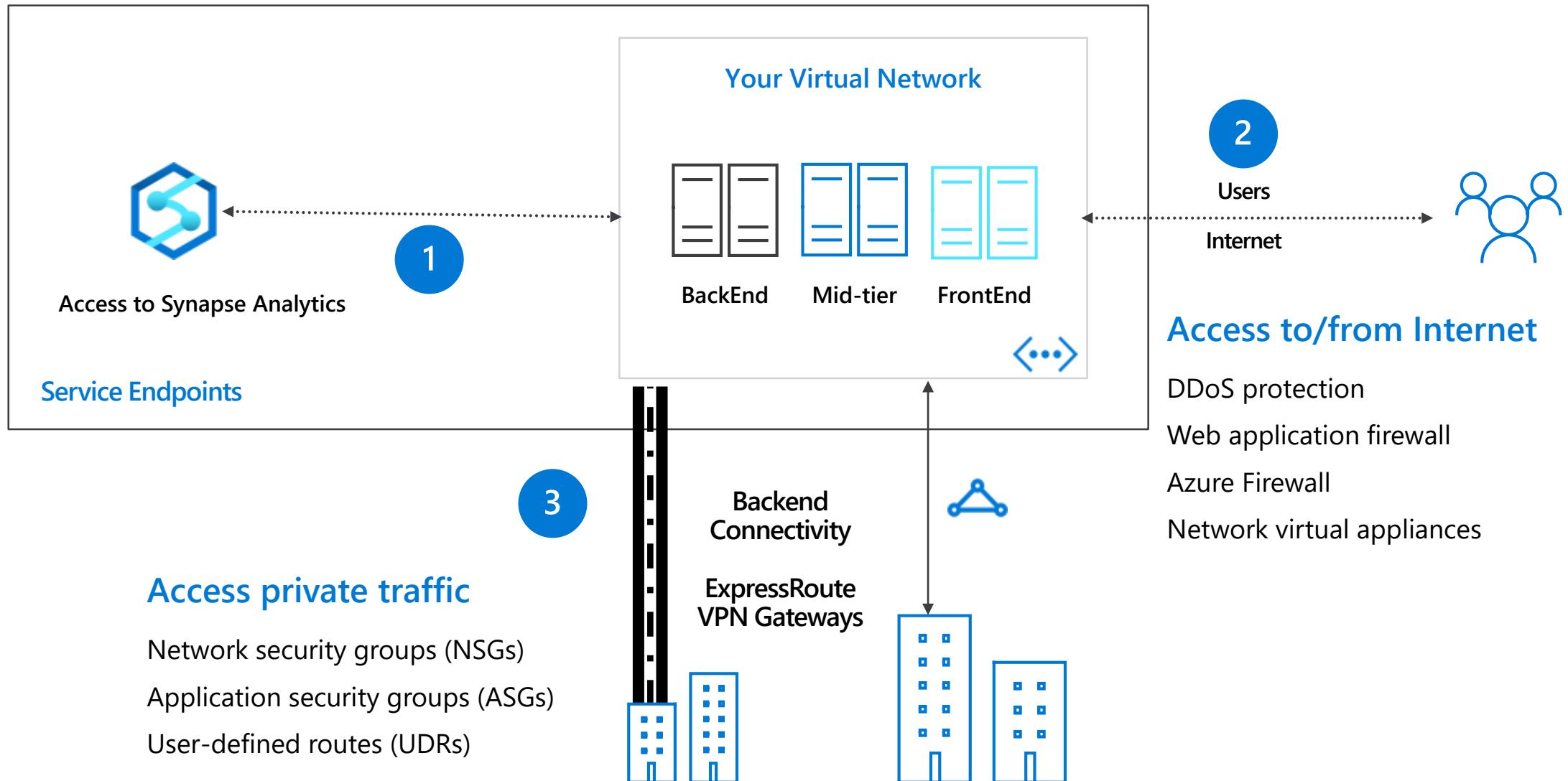
Data at different levels of security needs to be accessed from different locations.

## How do we achieve separation?

Disallowing access to entities outside the company's network security boundary.



# Azure networking: application-access patterns

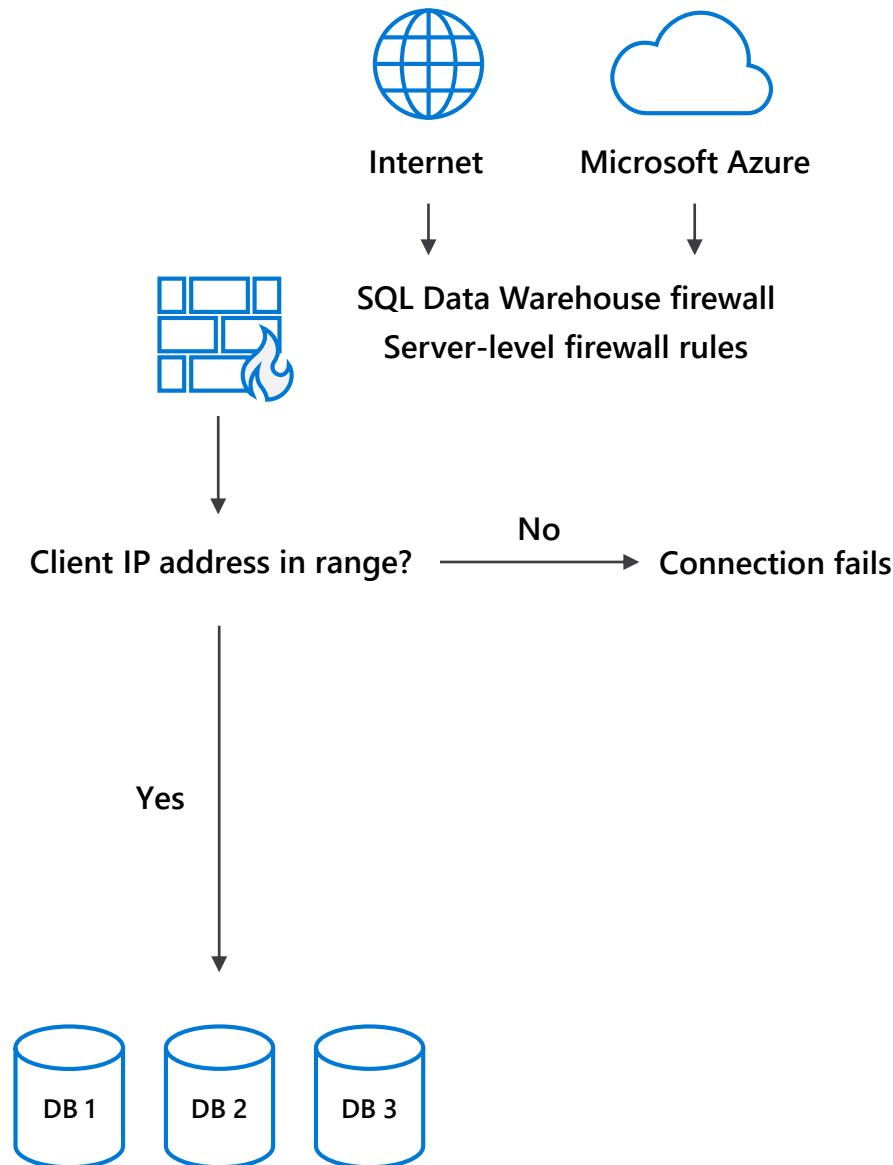


# Securing with firewalls

## Overview

By default, all access to your Azure Synapse Analytics is blocked by the firewall.

Firewall also manages virtual network rules that are based on virtual network service endpoints.

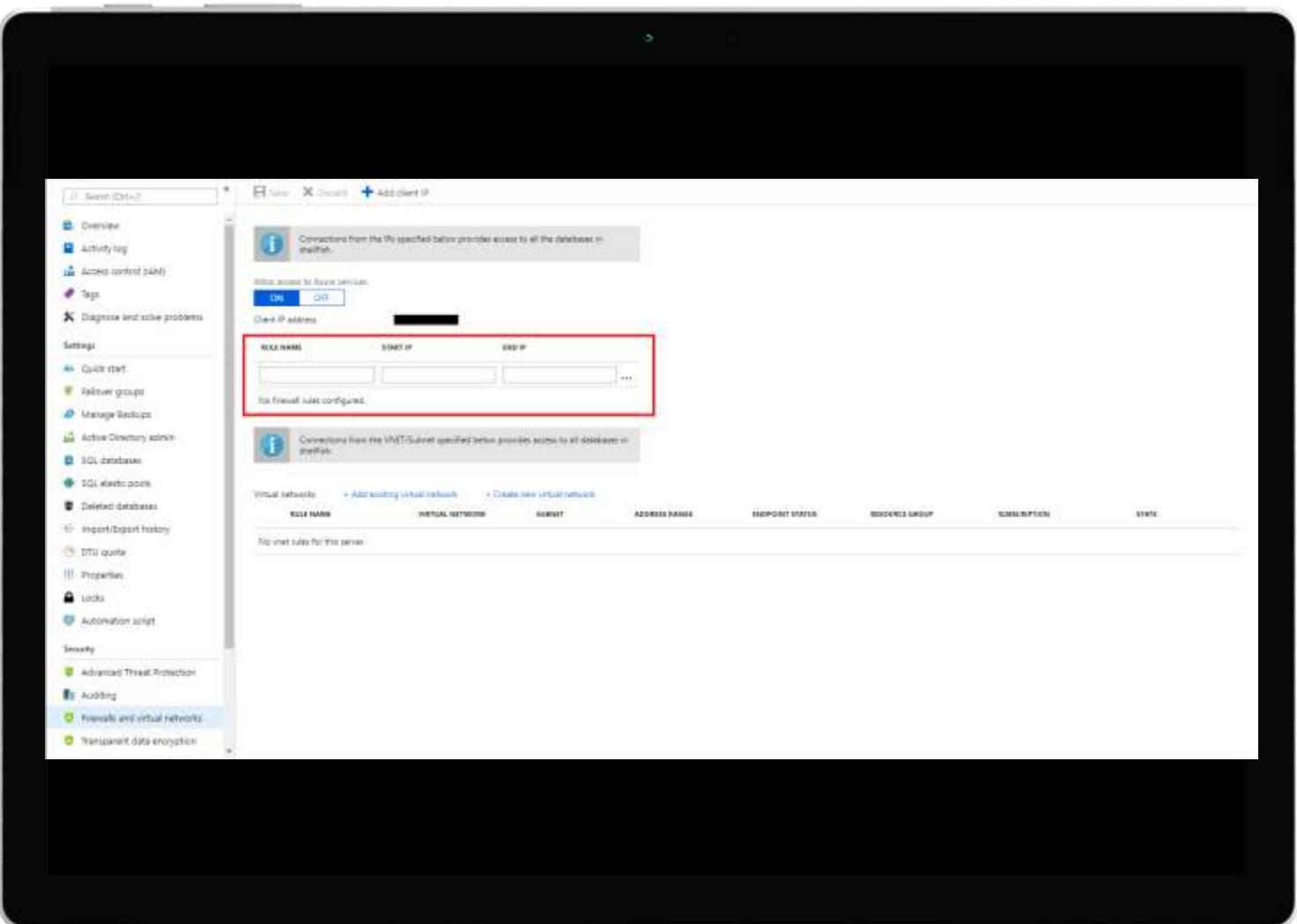


# Firewall configuration on the portal

By default, Azure blocks all external connections to port 1433

Configure with the following steps:

Azure Synapse Analytics Resource:  
Server name > Firewalls and virtual networks



# Firewall configuration using REST API

Managing firewall rules through REST API must be authenticated.

For information, see [Authenticating Service Management Requests](#).

Server-level rules can be created, updated, or deleted using [REST API](#).

To create or update a server-level firewall rule, execute the [PUT](#) method.

To remove an existing server-level firewall rule, execute the [DELETE](#) method.

To list firewall rules, execute the [GET](#).

PUT

```
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Sql/servers/{serverName}/firewallRules/{firewallRuleName}?api-version=2014-04-01REQUEST BODY
{
  "properties": {
    "startIpAddress": "0.0.0.3",
    "endIpAddress": "0.0.0.3"
  }
}
```

DELETE

```
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Sql/servers/{serverName}/firewallRules/{firewallRuleName}?api-version=2014-04-01
```

GET

```
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Sql/servers/{serverName}/firewallRules/{firewallRuleName}?api-version=2014-04-01
```

# Firewall configuration using PowerShell/T-SQL

## Windows PowerShell Azure cmdlets

```
New-AzureRmSqlServerFirewallRule
```

```
Get-AzureRmSqlServerFirewallRule
```

```
Set-AzureRmSqlServerFirewallRule
```

## Transact SQL

```
sp_set_firewall_rule
```

```
sp_delete_firewall_rule
```

```
# PS Allow external IP access to SQL DW  
PS C:\> New-AzureRmSqlServerFirewallRule  
    -ResourceGroupName "myResourceGroup" `  
    -ServerName $servername `  
    -FirewallRuleName "AllowSome" `  
    -StartIpAddress "0.0.0.0" `  
    -EndIpAddress "0.0.0.0"  
  
-- T-SQL Allow external IP access to SQL DW  
EXECUTE sp_set_firewall_rule  
    @name = N'ContosoFirewallRule',  
    @start_ip_address = '192.168.1.1',  
    @end_ip_address = '192.168.1.10'
```

# VNET configuration on Azure portal

## Configure with the following steps:

Azure Synapse Analytics Resource:

Server name > Firewalls and virtual networks

REST API and PowerShell alternatives available

### Note:

By default, VMs on your subnets cannot communicate with your SQL Data Warehouse.

There must first be a virtual network service endpoint for the rule to reference.

The screenshot shows the 'Firewall / Virtual Networks' settings for a SQL server named 'gm-sql-db-server-svr1'. At the top, there are 'Save' and 'Discard' buttons, and a '+ Add client IP' button which is highlighted with a red box. Below this is an information icon with the text: 'Connections from the IPs specified below provides access to all the databases in gm-sql-db-server-svr1.' Underneath is a section for 'Allow access to Azure services' with an 'ON' switch. The 'Client IP address' is listed as 73.118.201.137. The main table lists two IP rules:

RULE NAME	START IP	END IP	Actions
gm-ip-rule-ir1	172.27.26.0	172.27.26.255	...
gm-ip-rule-ir2	73.118.201.0	73.118.201.255	...

Below the table is another information icon with the text: 'Connections from the VNET/Subnet specified below provides access to all databases in gm-sql-db-server-svr1.' At the bottom, there are buttons for 'Virtual networks' (disabled), '+ Add existing' (highlighted with a red box), and '+ Create new'. The table headers for the bottom section are 'RULE NAME', 'RESOURCE GROUP/VNET NAME', and 'SUBNET'.

# Authentication - Business requirements

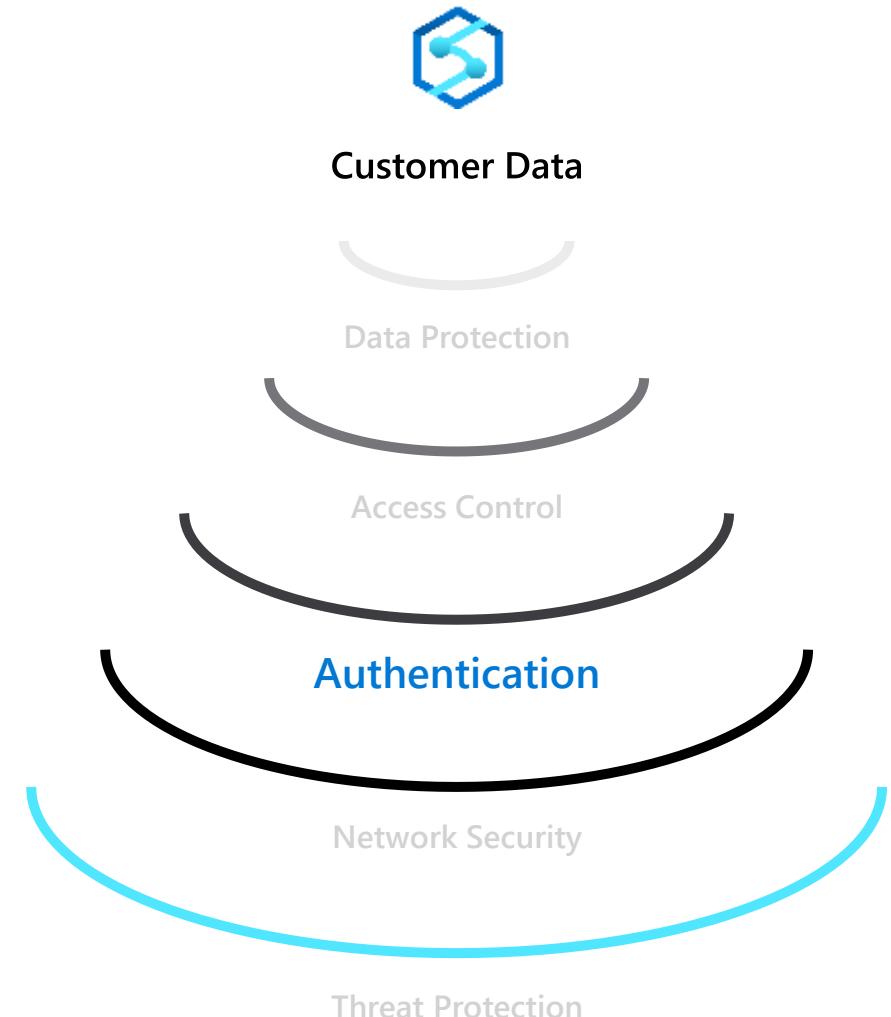


**How do I configure Azure Active Directory with Azure Synapse Analytics?**

I want additional control in the form of multi-factor authentication



**How do I allow non-Microsoft accounts to be able to authenticate?**



# Azure Active Directory authentication

## Overview

Manage user identities in one location.

Enable access to Azure Synapse Analytics and other Microsoft services with Azure Active Directory user identities and groups.

## Azure Synapse Analytics

## Benefits

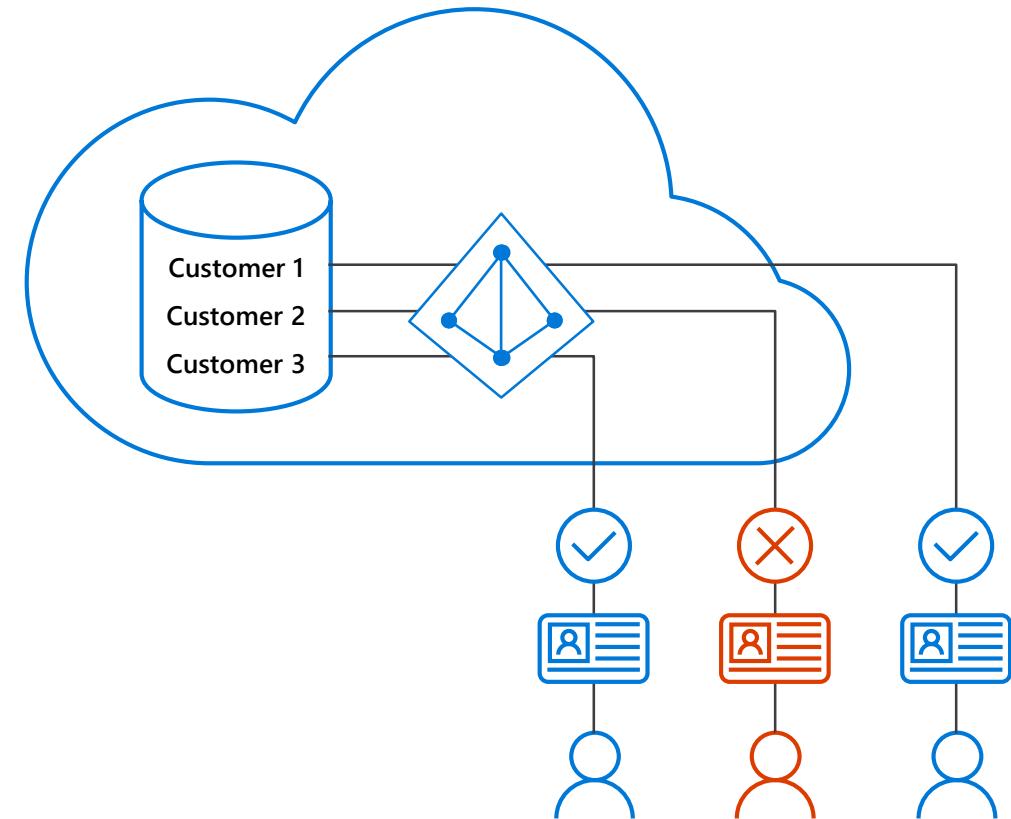
Alternative to SQL Server authentication

Limits proliferation of user identities across databases

Allows password rotation in a single place

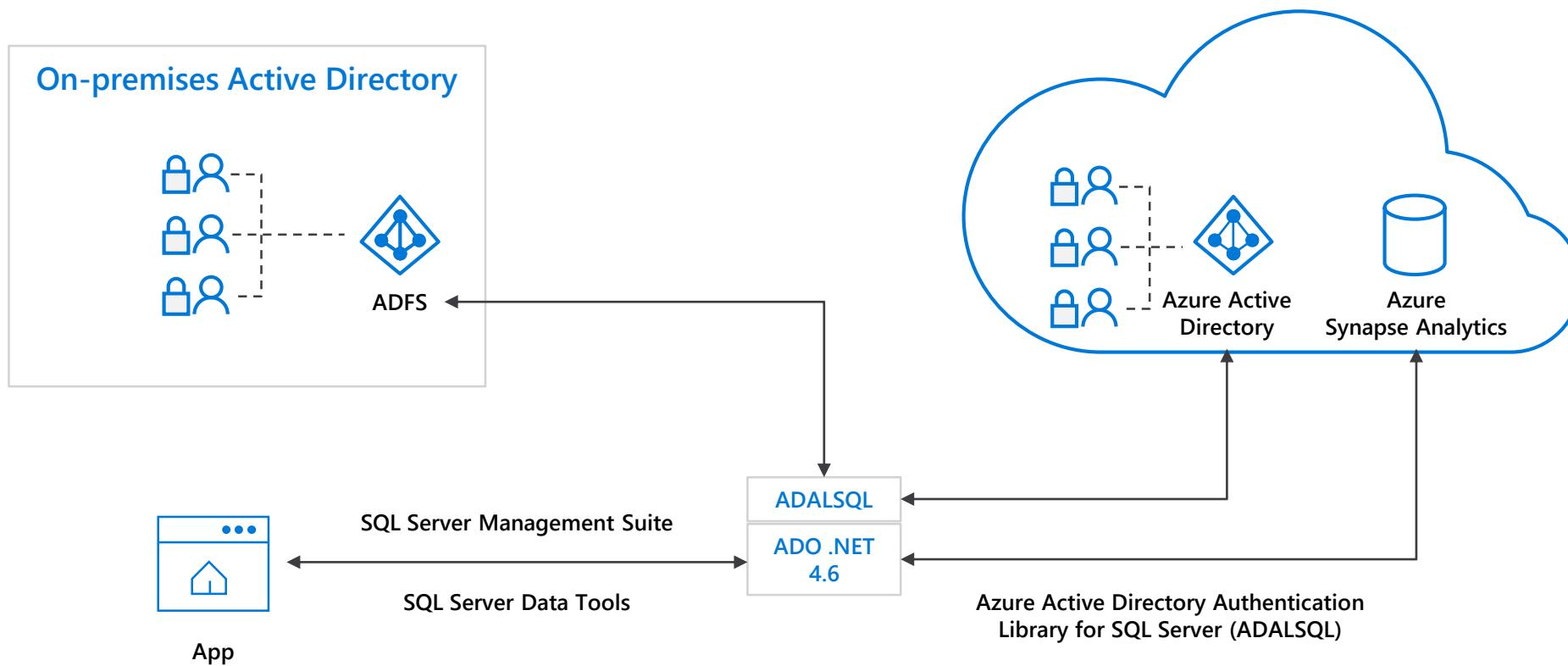
Enables management of database permissions by using external Azure Active Directory groups

Eliminates the need to store passwords



# Azure Active Directory trust architecture

## Azure Active Directory and Azure Synapse Analytics



# SQL authentication

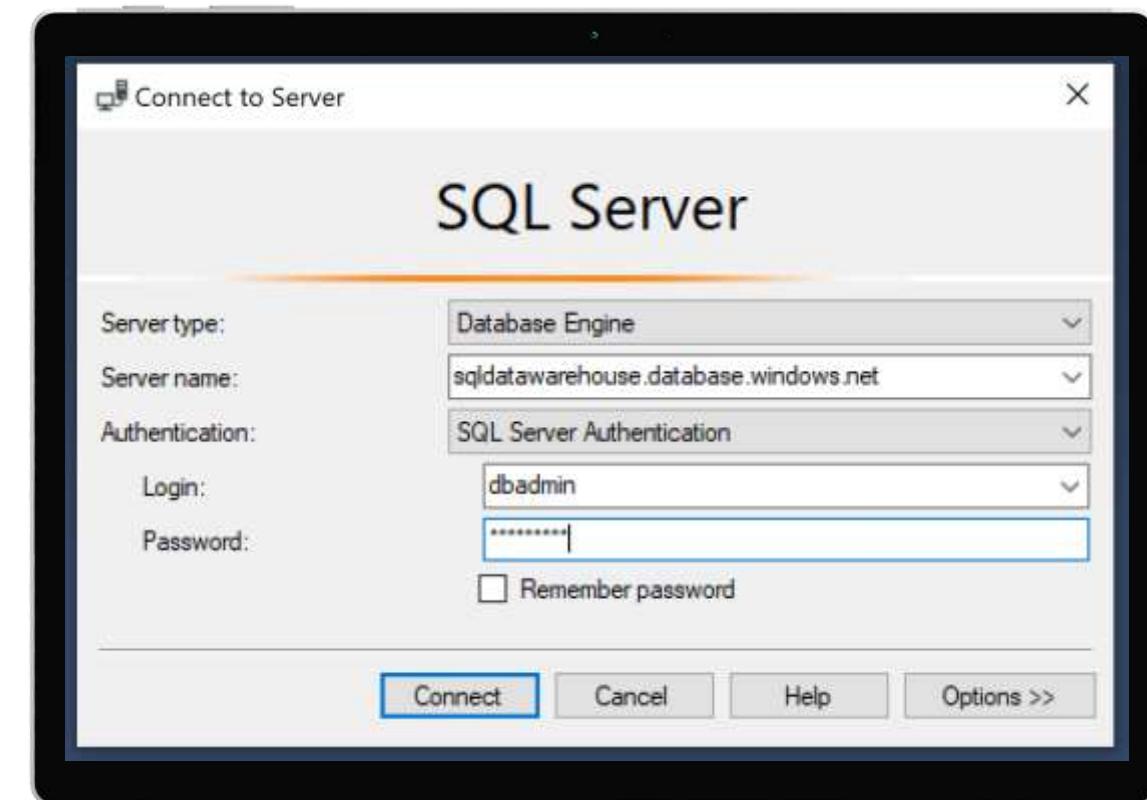
## Overview

This authentication method uses a username and password.

When you created the logical server for your data warehouse, you specified a "server admin" login with a username and password.

Using these credentials, you can authenticate to any database on that server as the database owner.

Furthermore, you can create user logins and roles with familiar SQL Syntax.



```
-- Connect to master database and create a login  
CREATE LOGIN ApplicationLogin WITH PASSWORD = 'Str0ng_password';  
CREATE USER ApplicationUser FOR LOGIN ApplicationLogin;
```

```
-- Connect to SQL DW database and create a database user  
CREATE USER DatabaseUser FOR LOGIN ApplicationLogin;
```

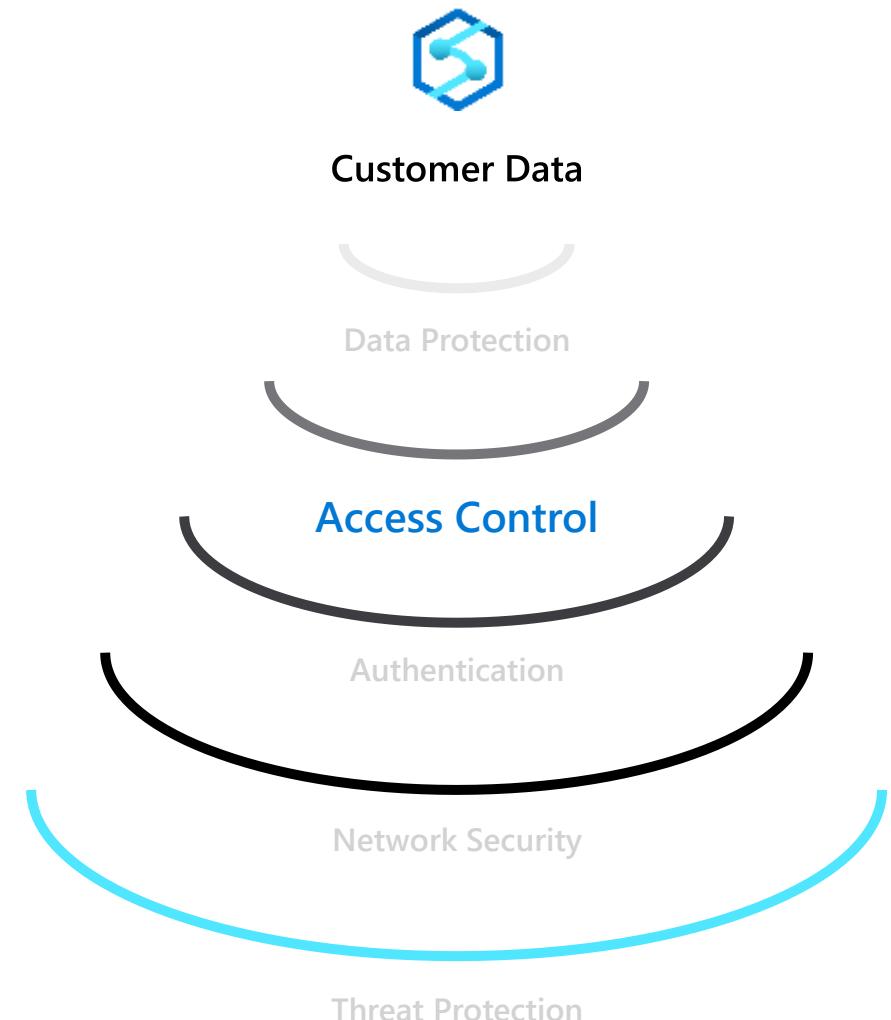
# Access Control - Business requirements



How do I restrict access to sensitive data to specific database users?

How do I ensure users only have access to relevant data?

For example, in a hospital only medical staff should be allowed to see patient data that is relevant to them—and not every patient's data.



# Object-level security (tables, views, and more)

## Overview

GRANT controls permissions on designated tables, views, stored procedures, and functions.

Prevent unauthorized queries against certain tables.

Simplifies design and implementation of security at the database level as opposed to application level.

```
-- Grant SELECT permission to user RosaQdM on table Person.Address in the AdventureWorks2012 database
GRANT SELECT ON OBJECT::Person.Address TO RosaQdM;
GO

-- Grant REFERENCES permission on column BusinessEntityID in view HumanResources.vEmployee to user Wanida
GRANT REFERENCES(BusinessEntityID) ON OBJECT::HumanResources.vEmployee TO Wanida WITH GRANT OPTION;
GO

-- Grant EXECUTE permission on stored procedure HumanResources.uspUpdateEmployeeHireInfo to an application role called Recruiting11
USE AdventureWorks2012;
GRANT EXECUTE ON OBJECT::HumanResources.uspUpdateEmployeeHireInfo TO RECRUITING 11;
GO
```

# Row-level security (RLS)

## Overview

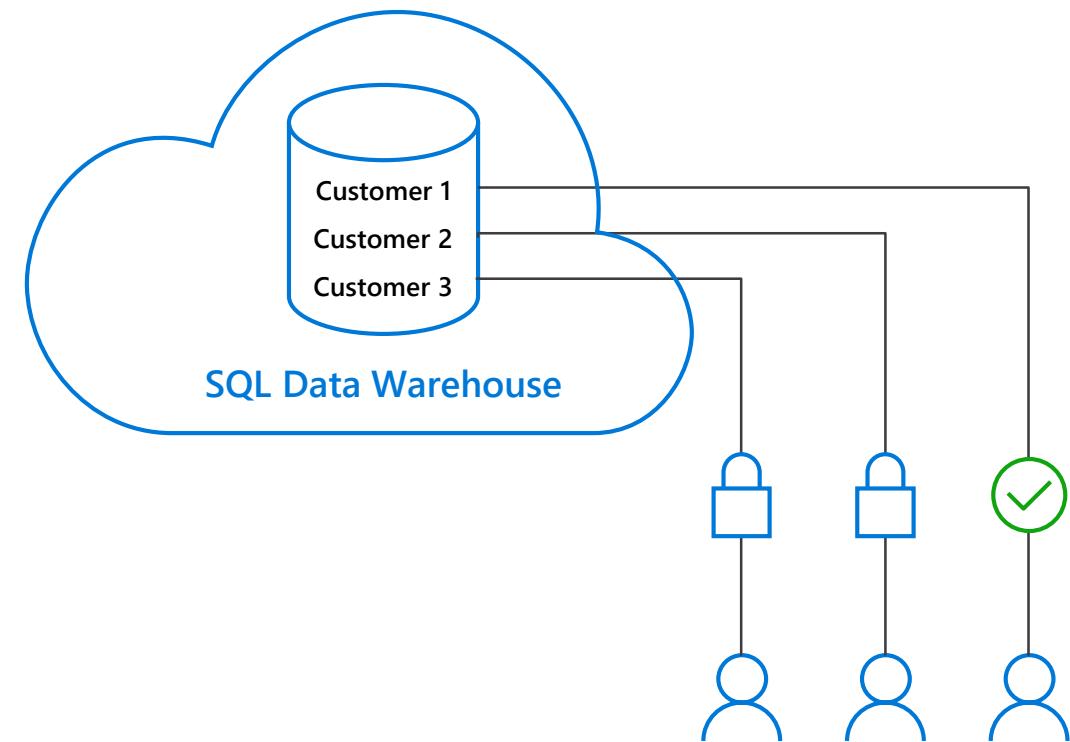
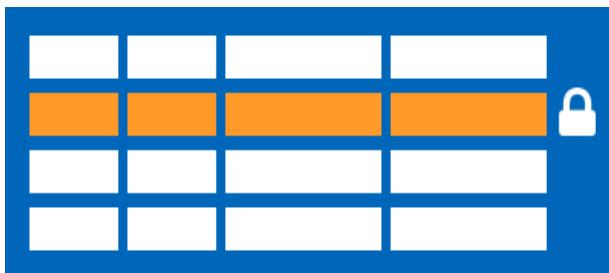
Fine grained access control of specific rows in a database table.

Help prevent unauthorized access when multiple users share the same tables.

Eliminates need to implement connection filtering in multi-tenant applications.

Administer via SQL Server Management Studio or SQL Server Data Tools.

Easily locate enforcement logic inside the database and schema bound to the table.



# Row-level security

## Creating policies

Filter predicates silently filter the rows available to read operations (SELECT, UPDATE, and DELETE).

The following examples demonstrate the use of the CREATE SECURITY POLICY syntax

```
-- The following syntax creates a security policy with a filter predicate for the Customer table
CREATE SECURITY POLICY [FederatedSecurityPolicy]
ADD FILTER PREDICATE [rls].[fn_securitypredicate]([CustomerId])
ON [dbo].[Customer];

-- Create a new schema and predicate function, which will use the application user ID stored in CONTEXT_INFO to filter rows.
CREATE FUNCTION rls.fn_securitypredicate (@AppUserId int)
RETURNS TABLE
WITH SCHEMABINDING
AS
RETURN (
SELECT 1 AS fn_securitypredicate_result
WHERE
DATABASE_PRINCIPAL_ID() = DATABASE_PRINCIPAL_ID('dbo') -- application context
AND CONTEXT_INFO() = CONVERT(VARBINARY(128), @AppUserId));
GO
```

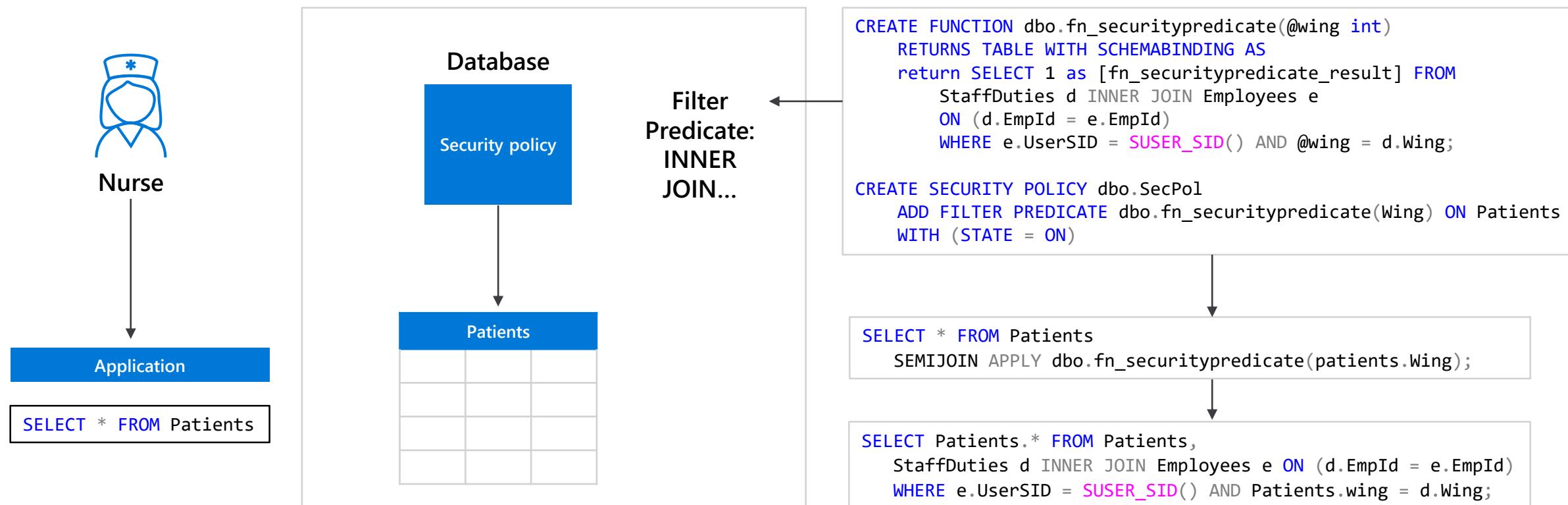
# Row-level security

## Three steps:

1. Policy manager creates filter predicate and security policy in T-SQL, binding the predicate to the patients table.
2. App user (e.g., nurse) selects from Patients table.
3. Security policy transparently rewrites query to apply filter predicate.



Policy manager



# Column-level security

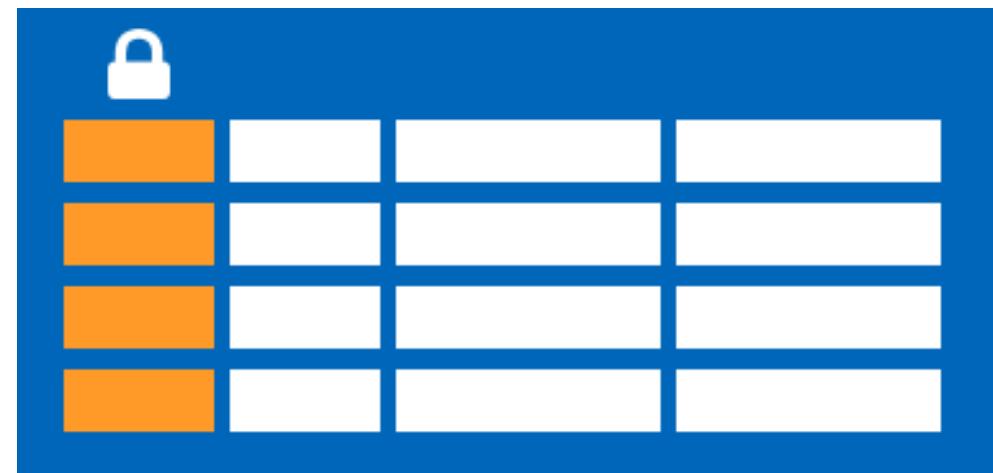
## Overview

Control access of specific columns in a database table based on customer's group membership or execution context.

Simplifies the design and implementation of security by putting restriction logic in database tier as opposed to application tier.

Administer via GRANT T-SQL statement.

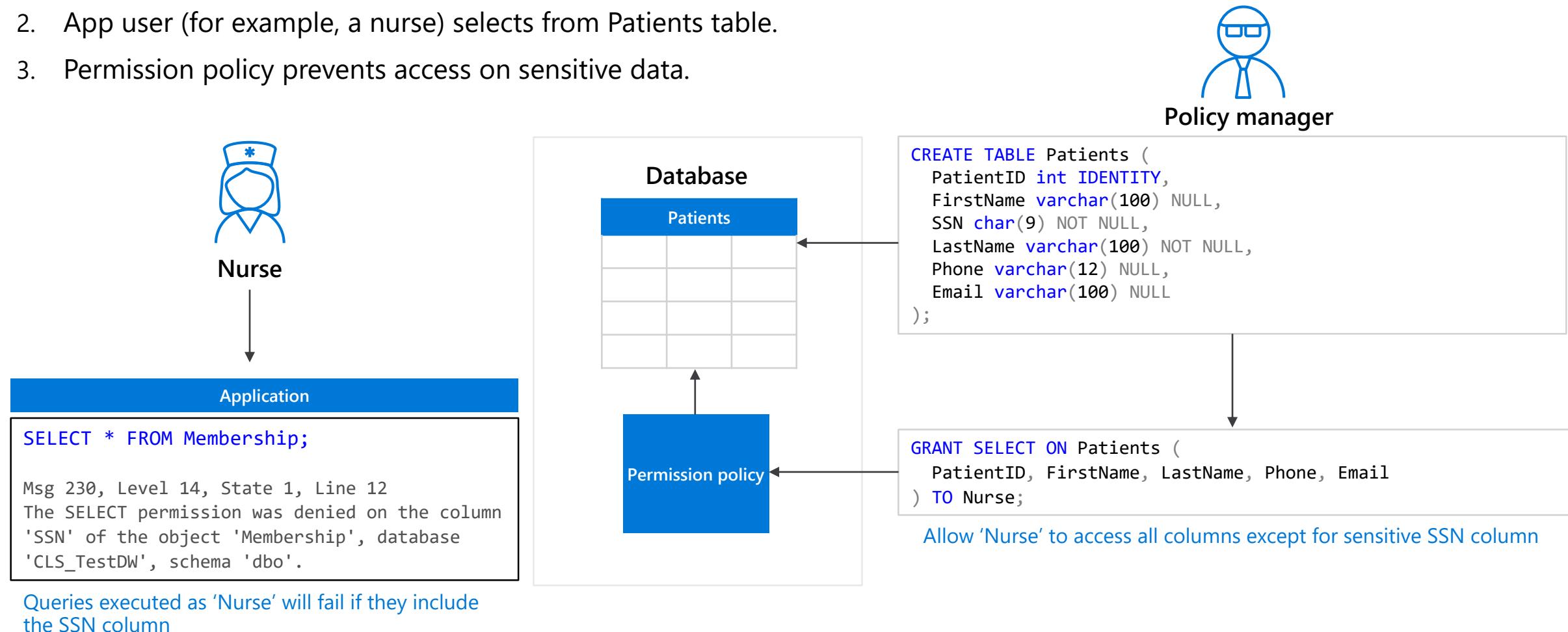
Both Azure Active Directory (AAD) and SQL authentication are supported.



# Column-level security

## Three steps:

1. Policy manager creates permission policy in T-SQL, binding the policy to the Patients table on a specific group.
2. App user (for example, a nurse) selects from Patients table.
3. Permission policy prevents access on sensitive data.

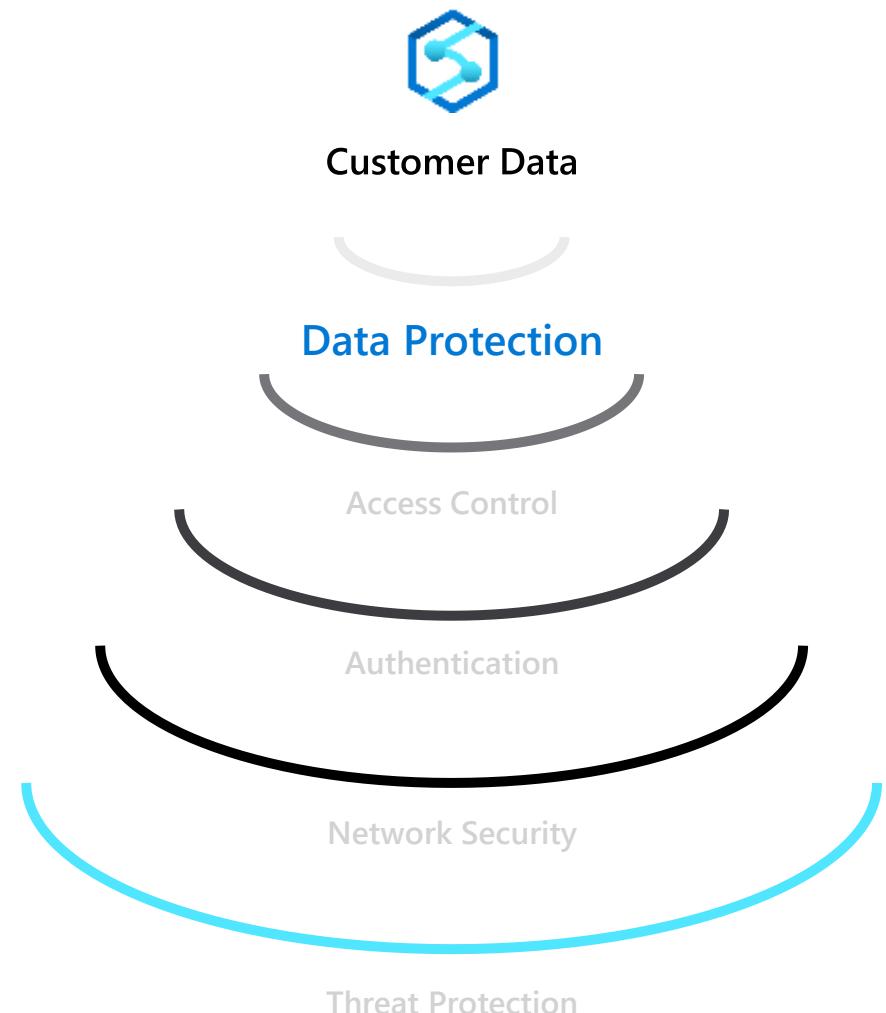


# Data Protection - Business requirements



How do I protect sensitive data against unauthorized (high-privileged) users?

What key management options do I have?



# Dynamic Data Masking

## Overview

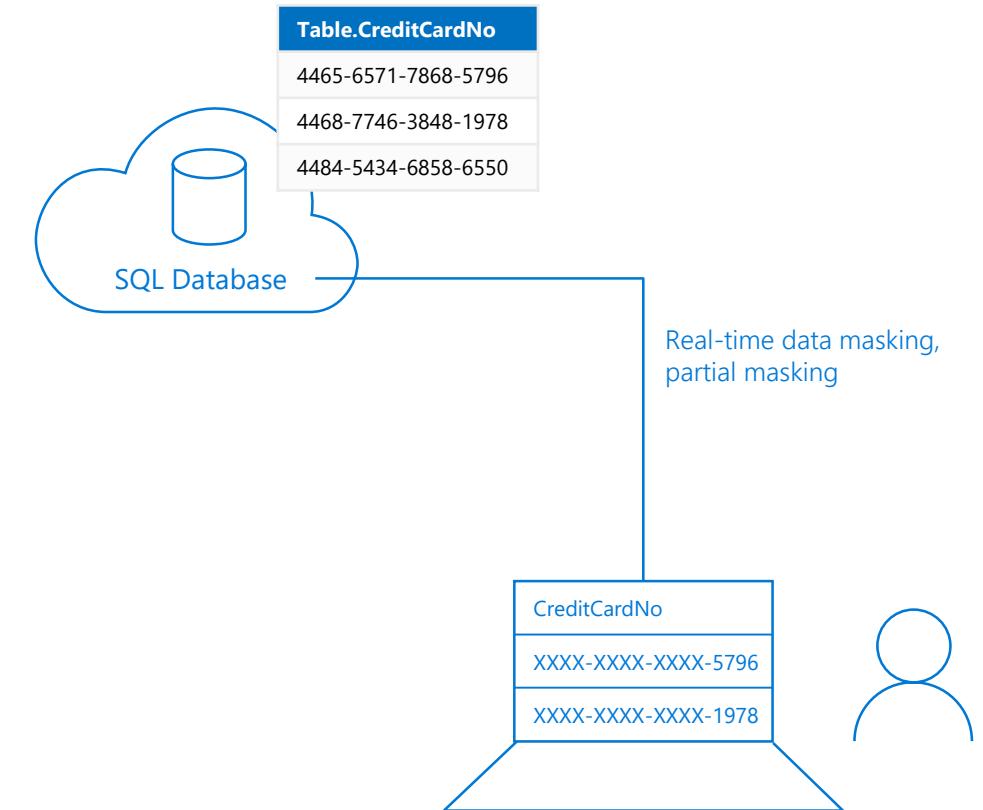
Prevent abuse of sensitive data by hiding it from users

Easy configuration in new Azure Portal

Policy-driven at table and column level, for a defined set of users

Data masking applied in real-time to query results based on policy

Multiple masking functions available, such as full or partial, for various sensitive data categories  
(credit card numbers, SSN, etc.)



# Dynamic Data Masking

## Three steps

1. Security officer defines dynamic data masking policy in T-SQL over sensitive data in the Employee table. The security officer uses the built-in masking functions (default, email, random)
2. The app-user selects from the Employee table
3. The dynamic data masking policy obfuscates the sensitive data in the query results for non-privileged users



Security officer

```

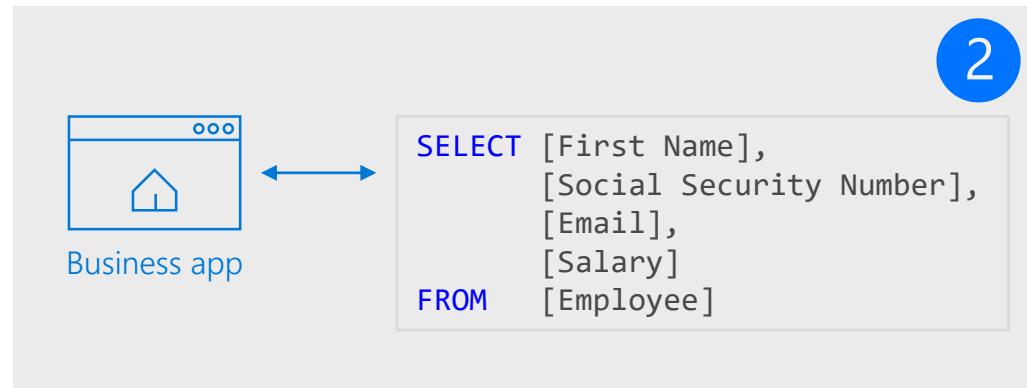
ALTER TABLE [Employee]
ALTER COLUMN [SocialSecurityNumber]
ADD MASKED WITH (FUNCTION = 'DEFAULT()')

ALTER TABLE [Employee]
ALTER COLUMN [Email]
ADD MASKED WITH (FUNCTION = 'EMAIL()')

ALTER TABLE [Employee]
ALTER COLUMN [Salary]
ADD MASKED WITH (FUNCTION = 'RANDOM(1,20000)')

GRANT UNMASK to admin1
    
```

1



2

Diagram illustrating Step 3:

	First Name	Social Security Num...	Email	Salary
1	LILA	758-10-9637	lila.barnett@comcast.net	1012794
2	JAMIE	113-29-4314	jamie.brown@ntlworld.com	1025713
3	SHELLEY	550-72-2028	shelley.lynn@charter.net	1040131
4	MARCELLA	903-94-5665	marcella.estrada@comcast.net	1040753
5	GILBERT	376-79-4787	gilbert.juarez@verizon.net	1041308

Non-masked data (admin login)

	First Name	Social Security Number	Email	Salary
1	LILA	758-10-9637	lila.barnett@comcast.net	1012794
2	JAMIE	113-29-4314	jamie.brown@ntlworld.com	1025713
3	SHELLEY	550-72-2028	shelley.lynn@charter.net	1040131
4	MARCELLA	903-94-5665	marcella.estrada@comcast.net	1040753
5	GILBERT	376-79-4787	gilbert.juarez@verizon.net	1041308

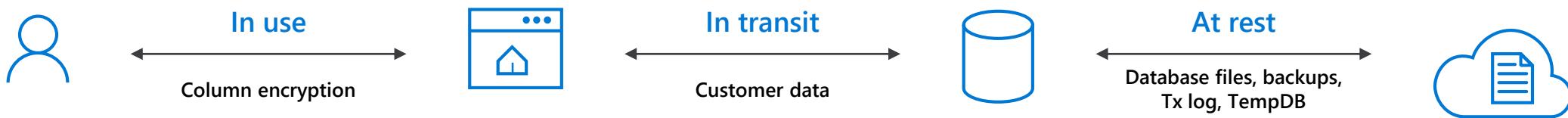
Masked data (admin1 login)

	First Name	Social Security Number	Email	Salary
1	LILA	XXX-XX-XX37	IXX@XXXX.net	8940
2	JAMIE	XXX-XX-XX14	jXX@XXXX.com	19582
3	SHELLEY	XXX-XX-XX28	sXX@XXXX.net	3713
4	MARCELLA	XXX-XX-XX65	mXX@XXXX.net	11572
5	GILBERT	XXX-XX-XX87	gXX@XXXX.net	4487

3

# Types of data encryption

Data Encryption	Encryption Technology	Customer Value
In transit	Transport Layer Security (TLS) from the client to the server TLS 1.2	Protects data between client and server against snooping and man-in-the-middle attacks
At rest	Transparent Data Encryption (TDE) for Azure Synapse Analytics	Protects data on the disk User or Service Managed key management is handled by Azure, which makes it easier to obtain compliance



# Transparent data encryption (TDE)

## Overview

All customer data encrypted at rest

TDE performs real-time I/O encryption and decryption of the data and log files.

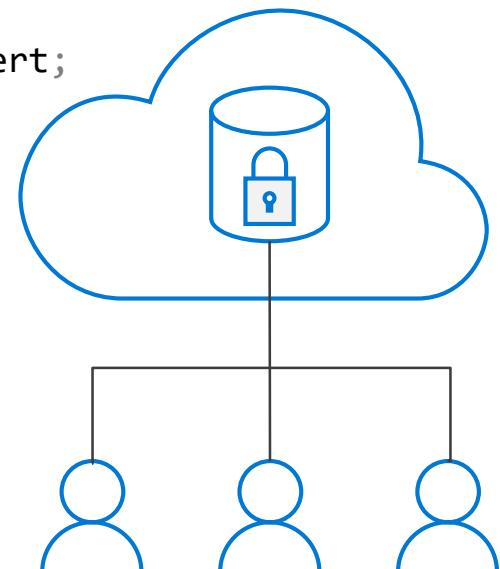
Service OR User managed keys.

Application changes kept to a minimum.

Transparent encryption/decryption of data in a TDE-enabled client driver.

Compliant with many laws, regulations, and guidelines established across various industries.

```
USE master;
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD = '<UseStrongPasswordHere>';
go
CREATE CERTIFICATE MyServerCert WITH SUBJECT = 'My DEK Certificate';
go
USE MyDatabase;
GO
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER CERTIFICATE MyServerCert;
GO
ALTER DATABASE MyDatabase
SET ENCRYPTION ON;
GO
```



# Transparent data encryption (TDE)

## Key Vault

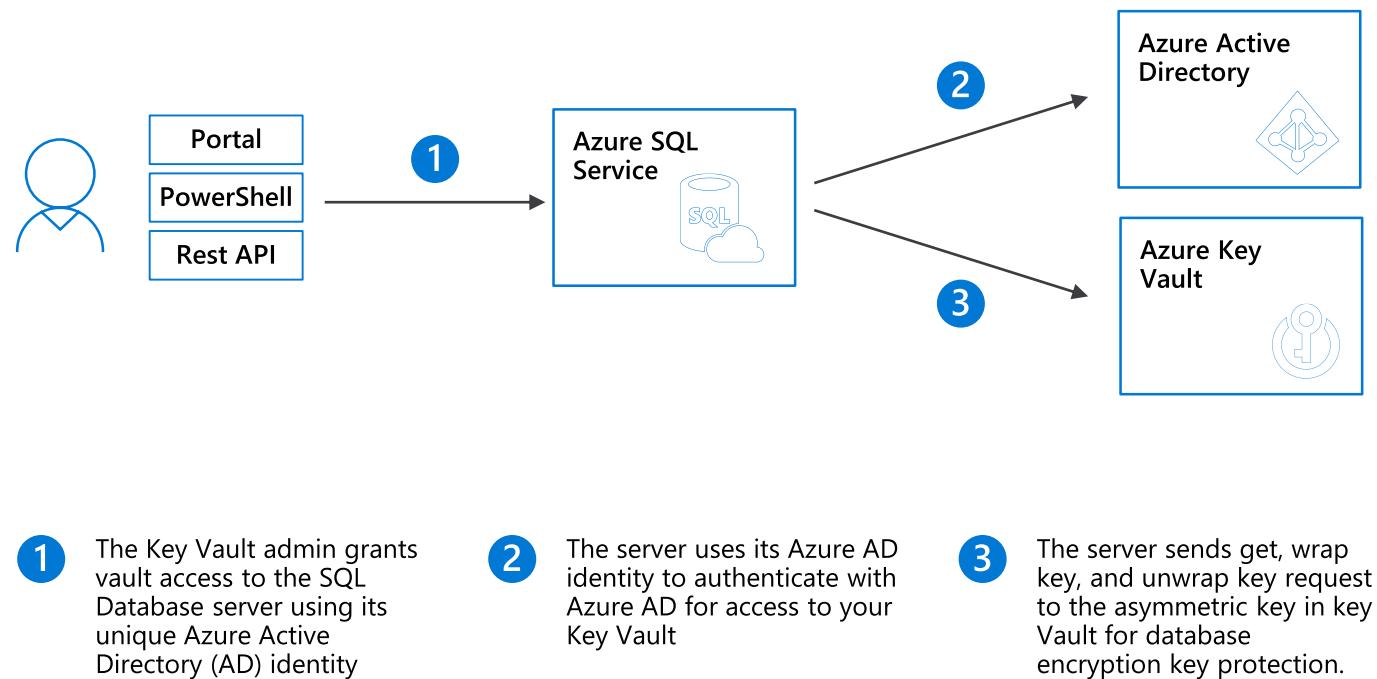
### Benefits with User Managed Keys

Assume more control over who has access to your data and when.

Highly available and scalable cloud-based key store.

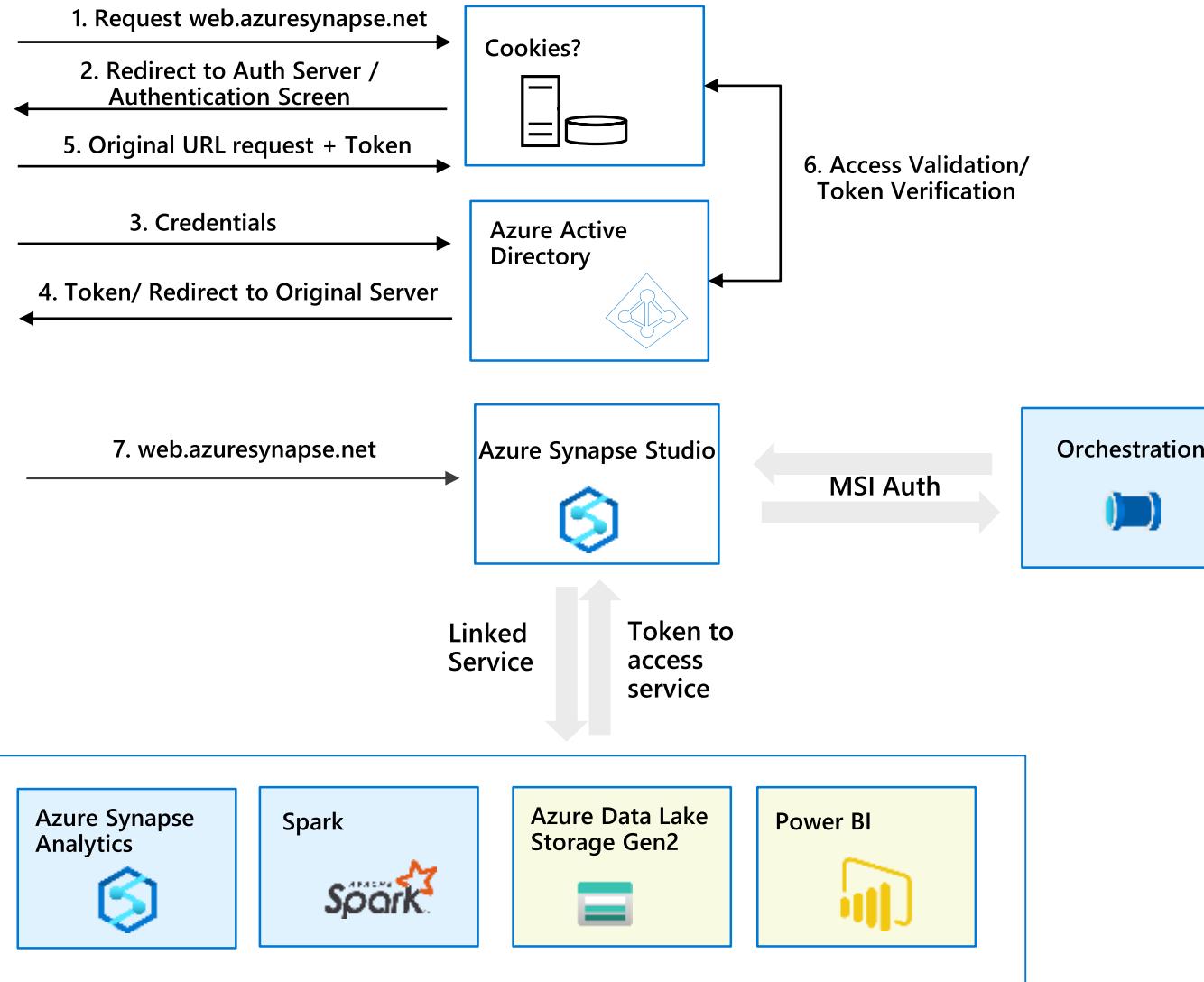
Central key management that allows separation of key management and data.

Configurable via Azure Portal, PowerShell, and REST API.



# Single Sign-On

Synapse Foundation Components  
 Synapse Linked Services



**Implicit authentication** - User provides login credentials once to access Azure Synapse Workspace

**AAD authentication** - Azure Synapse Studio will request token to access each linked services as user. A separate token is acquired for each of the below services:

1. ADLS Gen2
2. Azure Synapse Analytics
3. Power BI
4. Spark – Spark Livy API
5. management.azure.com – resource provisioning
6. Develop artifacts – dev.workspace.net
7. Graph endpoints

**MSI authentication** - Orchestration uses MSI auth for automation



End