DP-600 exam preparation notes

Note: The purpose of these notes is to help people in the data community prepare for the Microsoft DP-600 certification exam. This is a personal collection of notes, remarks and code snippets meant for educational purposes only.

Includes links, pictures and text from freely available online sources

- **Microsoft Learn website**
- **https://www.skool.com/microsoft-fabric/**
- **https://data-mozart.com/dp-600-certification/**

Plan, implement, and manage a solution for data analytics (10–15%)

Topic 1: Identify requirements for a solution, including components, features, performance, and capacity stock-keeping units (SKUs)

- To share content and collaborate in Microsoft Fabric, an organization needs to have an **F or P capacity license**, and at least **one per-user license**.
- A Microsoft Fabric **capacity** resides on a **tenant**.
- **Domain** is a logical grouping of workspaces. To group data into domains, workspaces are associated with domains. When a workspace is assigned to a domain, all the items in the workspace are associated with the domain.
- **Workspaces** reside within capacities and are used as containers for Microsoft Fabric items.
- **Items** are artifacts that are created within workspaces and are part of different **Fabric experiences**.

- Capacity licenses, are split into Stock Keeping Units (SKUs). Each SKU provides a set of Fabric resources for the organization.
- For Fabric capacities – 2 CUs is minimum and 2048 CUs is maximum. Power BI P1 equivalent is F64
- Capacity units (CUs) are units of measure that represent a pool of compute power needed. Compute power is required to run queries, jobs, or tasks.
- Pricing for Compute
    - Pay as you Go
    - One year reservation
- Pricing for Storage
    - Pay as you go ($ per GB / month)
- Factors that determine capacity requirements
    - Compliance with data residency requirements
    - Billing preferences
    - Segregation by workload requirements (DE vs BI)
    - Segregation by departments
- Factors that determine storage requirements
    - Data structure and type (structured, semi-structured or un-structured)
    - Structured / Relational -> Warehouse
    - All types -> Lakehouse

- Real time streaming -> KQL database (Eventhouse)

Topic 2: Choose a data gateway type

There are 2 types of data connections

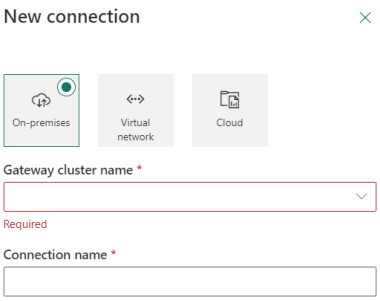
- Data Gateway connections
  - Personal data gateway
  - Enterprise data gateway
  - V-Net data gateway
- Direct cloud connections
  - Personal cloud connections
  - Shareable cloud connections

Data Gateway connections

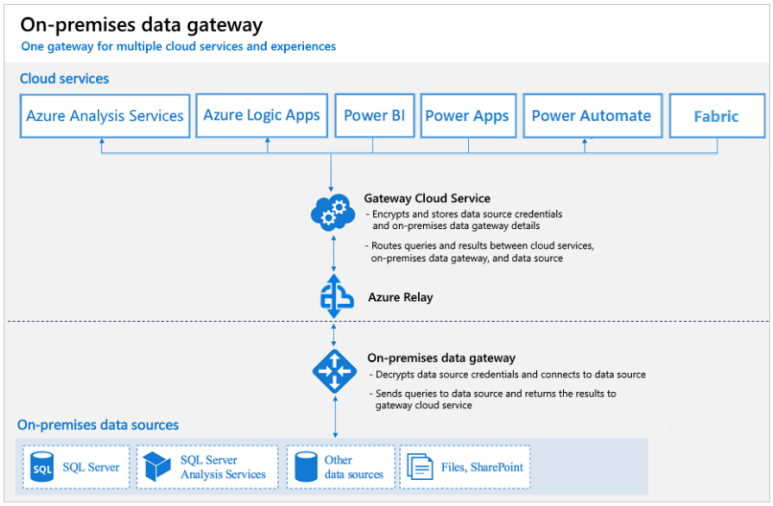
To connect to on-premises data sources or a data source that might be protected by a firewall or a virtual network, the following options are available.

On-premises data gateway

- The gateway acts as a bridge between your on-premises data sources and Fabric. The gateway is installed on a server within your network, and it allows Fabric to connect to your data sources through a secure channel without the need to open ports or make changes to your network.

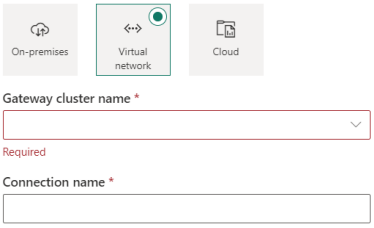


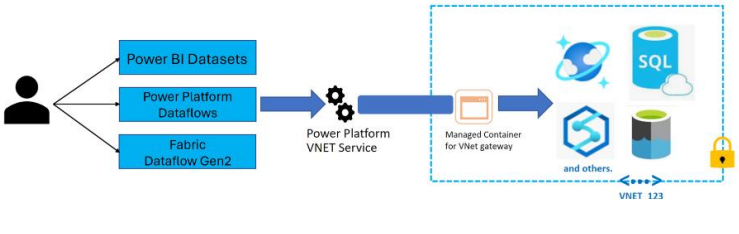
- The on-premises data gateway provides quick and secure data transfer between on-premises data and several Microsoft cloud services, such as Power BI, Power Apps, Power Automate, Azure Analysis Services, and Azure Logic Apps.
- Accessible through dataflows and data pipelines.



Virtual network (VNet) data gateway

- The VNet gateway allows you to connect from Microsoft Cloud services to your Azure data services within a VNet, without the need of an on-premises data gateway.





- Currently, this feature is available only for Fabric Dataflow Gen2, Power BI semantic models, Power Platform dataflows, and Power BI paginated reports. Power BI dataflows and datamarts are not supported.

In order to use the VNet data gateway, you need to do the fol

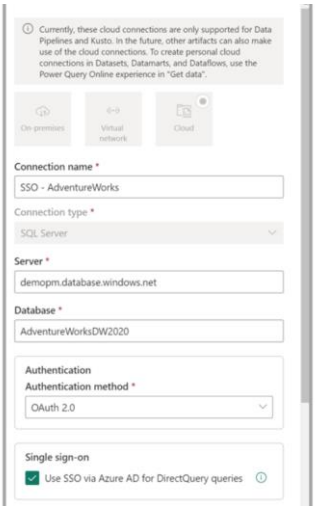
1. Register Power Platform resources provider.
2. Create a private endpoint on your Azure object
3. Create a subnet
4. In Fabric, create a new virtual gateway connection.
5. Use a dataflow Gen2 or data pipeline to get the data into Fabric

Features	Personal Mode	Standard Mode	VNet
Numbers of users supported	One	Multiple	Multiple
Managed by	Customer (User)	Customer (Admin)	Microsoft
Support of Data Refresh	Yes	Yes	Yes
Support for Azure Active Directory (Single Sign-On) DirectQuery PBI dataset	No	No	Yes
Support for standard DirectQuery + source Single Sign-On	No	Yes	Yes
Live Connection	No	Yes	Yes
Python & R visuals	Yes	No	No

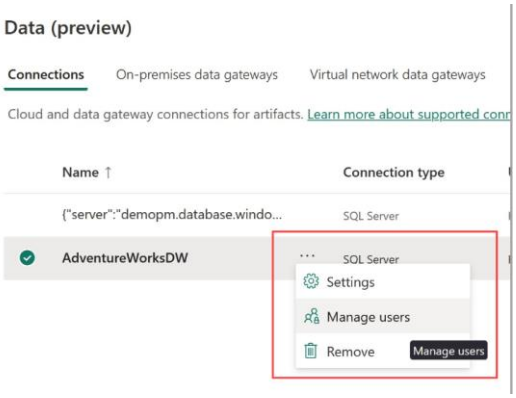
Direct Cloud Connections

Fabric or Power BI Services can generally connect to cloud data sources without a gateway. However, you might need a data gateway if your data sources are behind a firewall, require a VPN, or are on virtual networks.

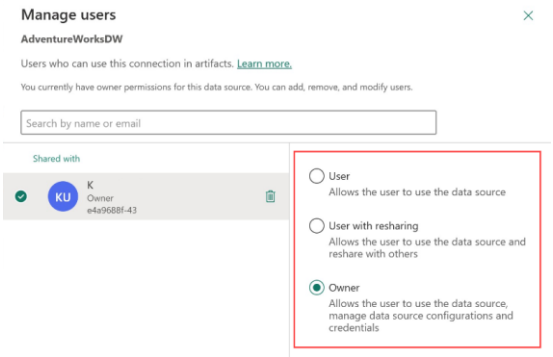
By default, when you create a Power BI Desktop report that connects to a cloud data source, then upload it into a workspace in the Power BI service, Power BI creates a personal cloud connection and binds it to your semantic model, for which you must provide credentials. If an existing personal cloud connection is available, you likely provided the credentials previously.



To share a shareable cloud connection that you've already created, go to your **Connections** settings in the Power BI service, select the **More** menu (the ellipses) for the connection you want to share, and select **Manage users**.



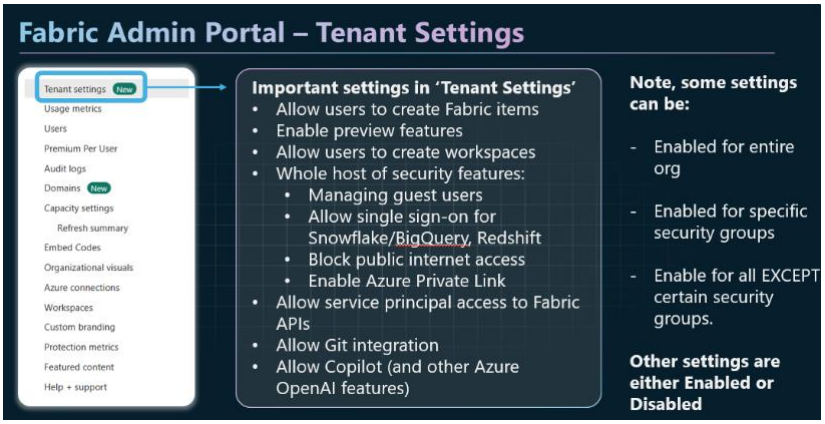
The **Manage users** window appears, where you can search users by name or by their email address, and then grant them the permission level you want them to have. You must at least grant User permission to allow users to connect their artifacts to the connection's data source.



Topic 3: Recommend settings in the Fabric admin portal

The admin portal can be accessed by admins with the following roles:

- Global administrator
- Power Platform administrator
- Fabric administrator



User Metrics is overview of items in workspace, user statistics and consumption by users. Important KPIs for both users and groups include

- Number of reports
- Number of dashboards
- Number of datasets
- Most consumed dashboards
- Top Users / Groups with most dashboards
- Top Users / Groups with most reports
- Most consumed packages by users / groups

Domains can be created based on requirements within the Fabric admin portal.

Embed codes and **Organizational visuals** are managed within the Fabric admin portal. Administrator can view, disable or delete the embed codes that are used for sharing reports publicly.

Azure connections is where you define where you can store your dataflows in your organizations's ADLS Gen2 (Azure Data Lake Storage Gen2) account. You can set the Tenant-level storage or Workspace-level storage. Data used in Power BI is stored in internal tenant-level storage, provided by Power BI. You can store your dataflows in ADLS Gen2 tenant-storage and access your dataflows using Azure portal, Azure Storage Explorer, and Azure APIs.

Workspaces can be viewed with ID, name and activation status.

Custom branding -> change logo, cover image and theme color.

Disabling **Publish to Web** disables the ability to publish any unsecure (no login required) reports to any embedded location.

Capacity settings

- Create new capacities
- Delete capacities
- Manage capacity permissions
- Change the size of the capacity

Topic 4: Create a custom Power BI report theme

Create and customize a theme in Power BI Desktop

- Name and colors: Theme name and color settings include theme colors, sentiment colors, divergent colors, and structural colors (Advanced).
- Text: Text settings include font family, size, and color, which sets the primary text class defaults for labels, titles, cards and KPIs, and tab headers.
- Visuals: Visual settings include background, border, header, and tooltips.
- Page: Page element settings include wallpaper and background.
- Filter pane: Filter pane settings include background color, transparency, font and icon color, size, and filter cards.
- Customize theme -> Options to change background color and wallpaper
- Report theme is nothing but a JSON file which can be saved and edited

Create and customize a report theme JSON file

- Name: The report theme name. This field is the only required field.
- dataColors: The list of color hexadecimal codes to use for data in Power BI Desktop visuals. This list can have as many or as few colors as you want.
- background, foreground, tableAccent: Color classes that enable to set many structural colors in your report at once.

Some other points to remember

- Custom colors do not stick when a report theme is changed
- A background image can also be included as part of the JSON file
- Image URL needs to be converted into a Base64 scheme, which is just an encoded string
- .PBIT file is another way (perhaps better) to share the custom themes developed for any organization

Topic 5: Implement workspace and item-level access controls for Fabric items

Workspace access control

- You can share a workspace with a user or security group (or M365 group). This will give access to everything in the workspace.
- When you share, you give the person/ group a role in the workspace. One of:
    - Admin
    - Member
    - Contributor
    - Viewer
- Security is derived from Microsoft Entra authentication and is compatible with user identities, service principals, and managed identities
- **Viewer** is the read-only role, which allows users to query data from SQL or Power BI reports but not create items or write data.
- **Admin, Member**, and **Contributor** are the read-write roles that can view data directly in OneLake, write data to OneLake, and create and manage items.
- You can simplify the management of Fabric workspace roles by assigning them to security groups. This method lets you control access by adding or removing members from the security group.
- Everyone in a user group gets the role that you've assigned. If someone is in several user groups, they get the highest level of permission that's provided by the roles that they're assigned. If you nest user groups and assign a role to a group, all the contained users have permissions.

Capability	Admin	Member	Contributor	Viewer
Update and delete the workspace.	✓			
Add or remove people, including other admins.	✓			
Add members or others with lower permissions.	✓	✓		
Allow others to reshare items.[1]	✓	✓		
View and read content of data pipelines, notebooks, Spark job definitions, ML models and experiments, and Event streams.	✓	✓	✓	✓
View and read content of KQL databases, KQL query-sets, and real-time dashboards.	✓	✓	✓	✓
Connect to SQL analytics endpoint of Lakehouse or the Warehouse	✓	✓	✓	✓
Read Lakehouse and Data warehouse data and shortcuts[2] with T-SQL through TDS endpoint.	✓	✓	✓	-[3]
Read Lakehouse and Data warehouse data and shortcuts[2] through OneLake APIs and Spark.	✓	✓	✓	-
Read Lakehouse data through Lakehouse explorer.	✓	✓	✓	-
Write or delete data pipelines, notebooks, Spark job definitions, ML models and experiments, and Event streams.	✓	✓	✓	-
Write or delete KQL query-sets, real-time dashboards, and schema and data of KQL databases, Lakehouses, data warehouses, and shortcuts.	✓	✓	✓	-
Execute or cancel execution of notebooks, Spark job definitions, ML models and experiments.	✓	✓	✓	-
Execute or cancel execution of data pipelines.	✓	✓	✓	✓
View execution output of data pipelines, notebooks, ML models and experiments.	✓	✓	✓	✓
Schedule data refreshes via the on-premises gateway.[4]	✓	✓	✓	
Modify gateway connection settings.[4]	✓	✓	✓	

[1] Contributors and Viewers can also share items in a workspace, if they have Reshare permissions.

[2] Additional permissions are needed to read data from shortcut destination.

[3] Admins, Members, and Contributors can grant viewers granular SQL permissions to query the SQL analytics endpoint of the Lakehouse and the Warehouse via TDS endpoints for SQL connections.

[4] Keep in mind that you also need permissions on the gateway. Those permissions are managed elsewhere, independent of workspace roles and permissions.

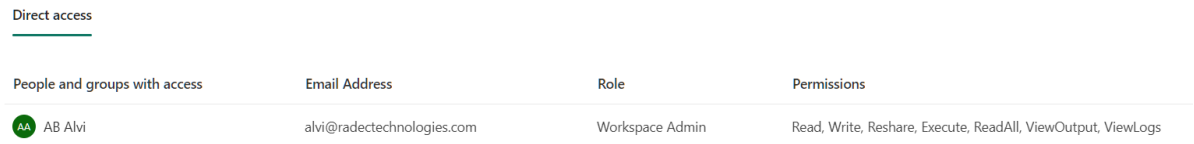
Item access control

Item: a set of capabilities bundled together into a single component. A data item or object is a subtype of item that allows data to be stored within it using OneLake. For example, Lakehouse is an item and tables / folders are data items or objects

- All Fabric items (lakehouses, notebooks, pipelines, etc.) are stored in OneLake and accessed via Fabric workspaces.
- Items always live within workspaces and workspaces always live directly under the OneLake namespace.
- Within a workspace, Fabric items can have permissions configured separately from the workspace roles. You can configure permissions either through **sharing an item** or by **managing the permissions of an item.**

Manage Permissions

- With the sharing feature, you can give a user **direct access to an item.**
- The user can only see that item in the workspace and isn't a member of any workspace roles.



Sharing

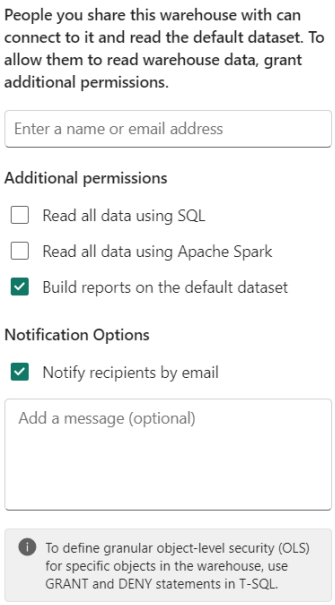
- This page allows you to add or remove permissions for ReadAll access for a user or group.
- This approach is ideal if you have workspace viewers that need OneLake access.

- Alternatively, you can use this page to adjust permissions after sharing the item.

Permission Name	Sharing text	Can view files in OneLake?	Can write files in OneLake?	Can read data through SQL analytics endpoint?
Read	*No share boxes selected*	No	No	No
ReadData	Read all SQL endpoint data	No	No	Yes
ReadAll	Read all Apache Spark	Yes	No	No
Write	*N/A, only available through workspace roles*	Yes	Yes	Yes

Topic 6: Implement data sharing for workspaces, warehouses, and lakehouses

Warehouse



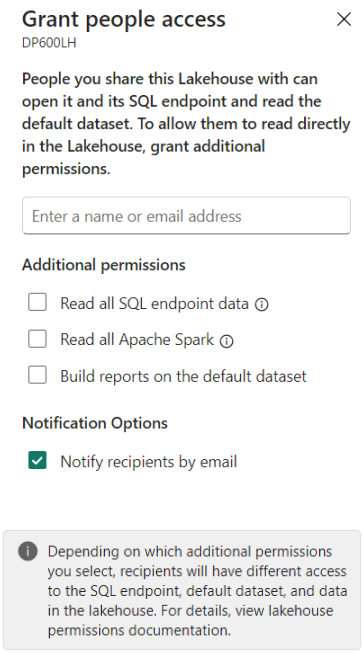
Read all data using SQL" is selected ("ReadData" permissions)- The shared recipient can **read all the database objects within the Warehouse**. ReadData is the equivalent of db_datareader role in SQL Server. The shared recipient can read data from all tables and views within the Warehouse. If you want to further restrict and provide granular access to some objects within the Warehouse, you can do this using T-SQL GRANT/REVOKE/DENY statements.

Read all data using Apache Spark" is selected ("ReadAll" permissions)- The shared recipient has **read access to the underlying parquet files in OneLake**, which can be consumed using Spark. ReadAll should be provided only if the shared recipient wants complete access to your warehouse's files using the Spark engine.

Build reports on the default semantic model" checkbox is selected ("Build" permissions)- The shared recipient can build reports on top of the default semantic model that is connected to your Warehouse. Build should be provided if the shared recipient wants Build permissions on the default semantic model, to create Power BI reports on this data. The Build checkbox is selected by default, but can be unchecked.

If no additional permissions are selected - The shared recipient by default receives "Read" permission, which **only allows the recipient to *connect* to the SQL analytics endpoint**, the equivalent of CONNECT permissions in SQL Server. **The shared recipient will not be able to query any table or view or execute any function or stored procedure unless they are provided access to objects within the Warehouse using T-SQL GRANT statement.**

Lakehouse



- In a **lakehouse** the users with Admin, Member, and Contributor roles can perform all CRUD operations on all data.
- When someone shares a lakehouse, they also **grant access to the SQL endpoint and read the associated default semantic model**

Read all SQL endpoint data - In the SQL analytics endpoint of the Lakehouse, "Read all SQL Endpoint data" is equivalent to "Read all data using SQL" for the warehouse

Read all data using Apache Spark - Same as warehouse

Build reports on the default semantic model - Same as warehouse

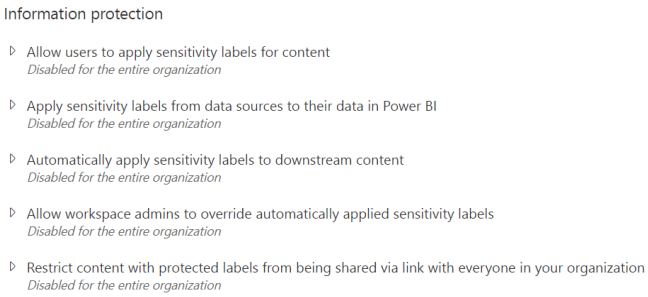
General guidance

- Write access: Users that need write access must be part of a workspace role that grants write access. This rule applies to all data items, so scope workspaces to a single team of data engineers.
- Lake access: To give users direct read access to data in OneLake, make them part of the Admin, Member, or Contributor workspace roles, or share the item with ReadAll access.
- General data access: Any user with Viewer permissions can access data through the warehouses, semantic models, or the SQL analytics endpoint for the Lakehouse.
- Object level security: To protect sensitive data, give users access to a warehouse or lakehouse SQL analytics endpoint through the Viewer role and use SQL DENY statements to restrict access to certain tables.

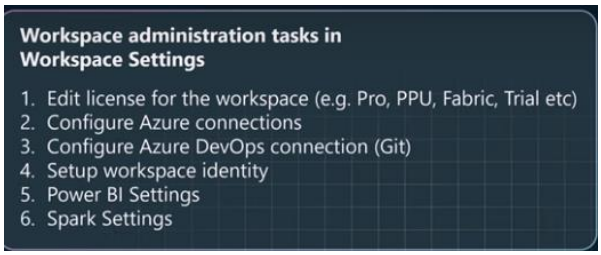
Topic 7: Manage sensitivity labels in semantic models and lakehouses

- Sensitivity labels are a data governance feature.
- Labels are created and managed in Microsoft Purview.
- Fabric items, such as a sematic model, can be given a sensitivity label such as 'Confidential'. This is for information protection.
- In some industries, labelling data and information with a sensitivity label is necessary for compliance with information protection regulations.
- To be able to apply labels to Power BI Fabric items, a user must have a Power BI Pro or Premium Per User (PPU) license in addition to one of the Azure Information Protection license P1 or P2
- Sensitivity labels must be enabled in your tenant before they can be used both in the Fabric service and in Power BI Desktop.
- When sensitivity labels are enabled:
    - Specified users and security groups in the organization can apply sensitivity labels to their Fabric content. In the Fabric service, this means any Fabric item. In Power BI Desktop, it means their *.pbix* files.

    - In the service, all members of the organization can see those labels. In Desktop, only members of the organization who have the labels published to them can see the labels.
- A protection metrics report available in the Power BI admin portal gives Power BI admins full visibility over the sensitive data in the Power BI tenant.
- In addition, the Power BI audit logs include sensitivity label information about activities such as applying, removing, and changing labels, as well as about activities such as viewing reports, dashboards, etc.
- Before enabling sensitivity labels on your tenant, make sure that sensitivity labels have been defined and published for relevant users and groups.
- Power BI semantic models that connect to sensitivity-labeled data in supported data sources can inherit those labels, so that the data remains classified and secure when brought into Power BI.



Topic 8: Configure Fabric-enabled workspace settings



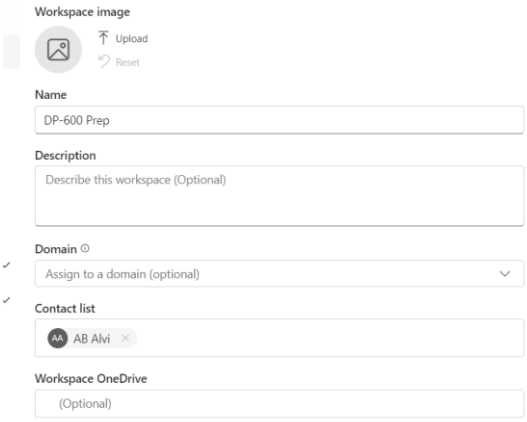
Workspace States

State	Description
Active	A normal workspace. It doesn't indicate anything about usage or what's inside, only that the workspace itself is "normal".
Orphaned	A workspace with no admin user. You need to assign an admin.
Deleted	A deleted workspace. When a workspace is deleted, it enters a retention period. During the retention period, a Microsoft Fabric administrator can restore the workspace. See Workspace retention for detail. When the retention period ends, the workspace enters the *Removing* state.
Removing	At the end of a deleted workspace's retention period, it moves into the *Removing* state. During this state, the workspace is permanently removed. Permanently removing a workspace takes a short while, and depends on the service and folder content.
Not found	If the customer's API request includes a workspace ID for a workspace that doesn't belong to the customer's tenant, "Not found" is returned as the status for that ID.

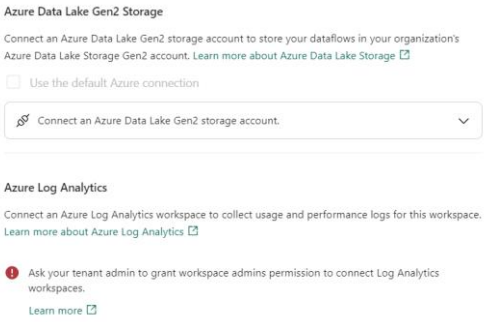
- The retention period for personal workspaces (*My workspaces*) is 30 days.
- The retention period for collaborative workspaces is configurable. The default retention period is seven days. However, Fabric administrators can change the length of the retention period by turning on the **Define workspace retention period** setting in the admin portal and specifying the desired retention period (from 7 to 90 days).
- During the retention period, Fabric administrators can restore the workspace. At the end of the retention period, the workspace is deleted permanently and it and its contents are irretrievably lost.
- While a workspace is in the retention period, Fabric administrators can permanently delete it before the end of the retention period.

About

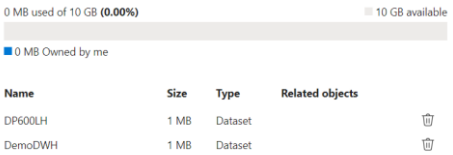
- Domains help teams in your org find data they need, and make governing your tenant easier like 'Finance team', or 'HR', then assign relevant workspaces to them.



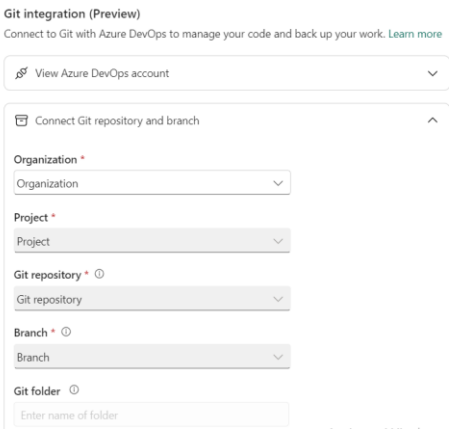
Azure Connections



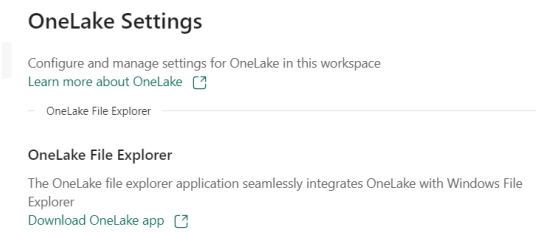
System Storage



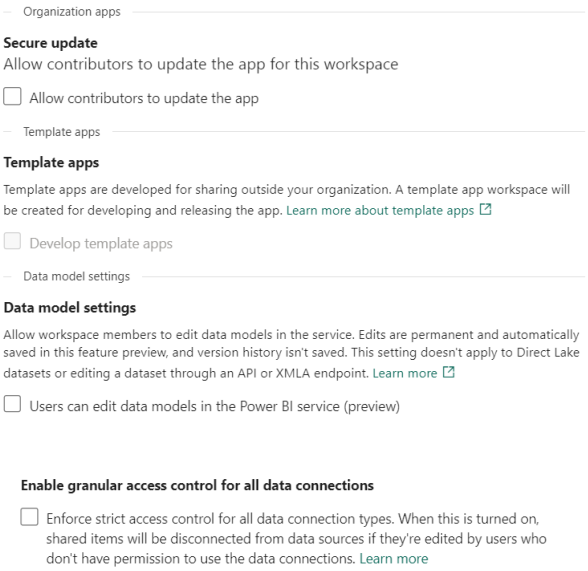
Git Integration



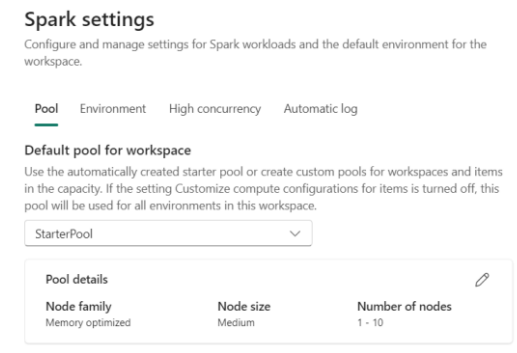
One Lake

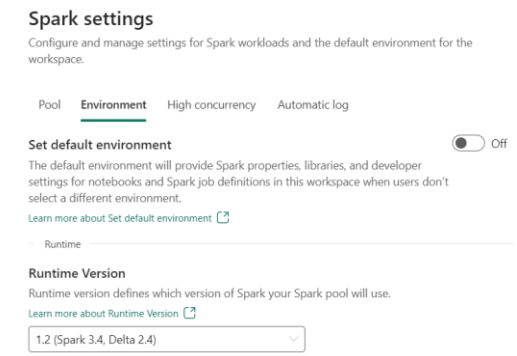


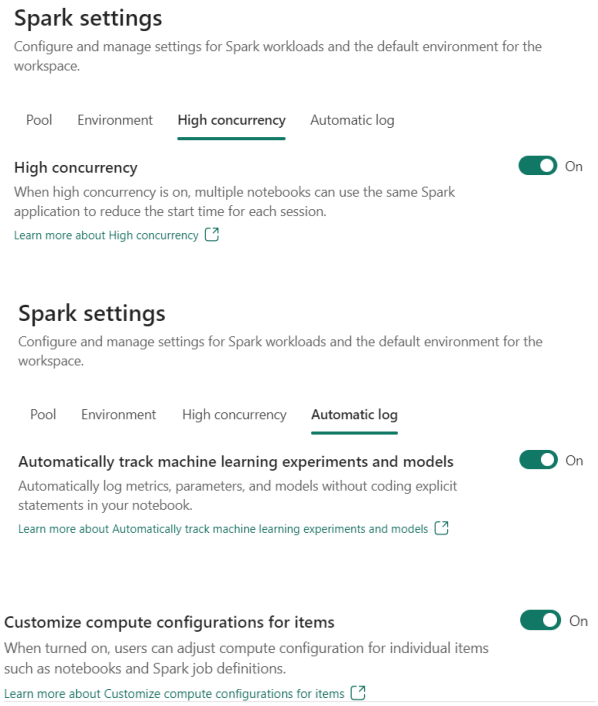
Power BI



Data Engineering / Science (Spark)

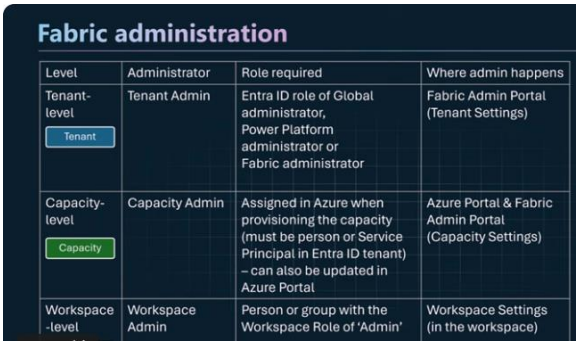






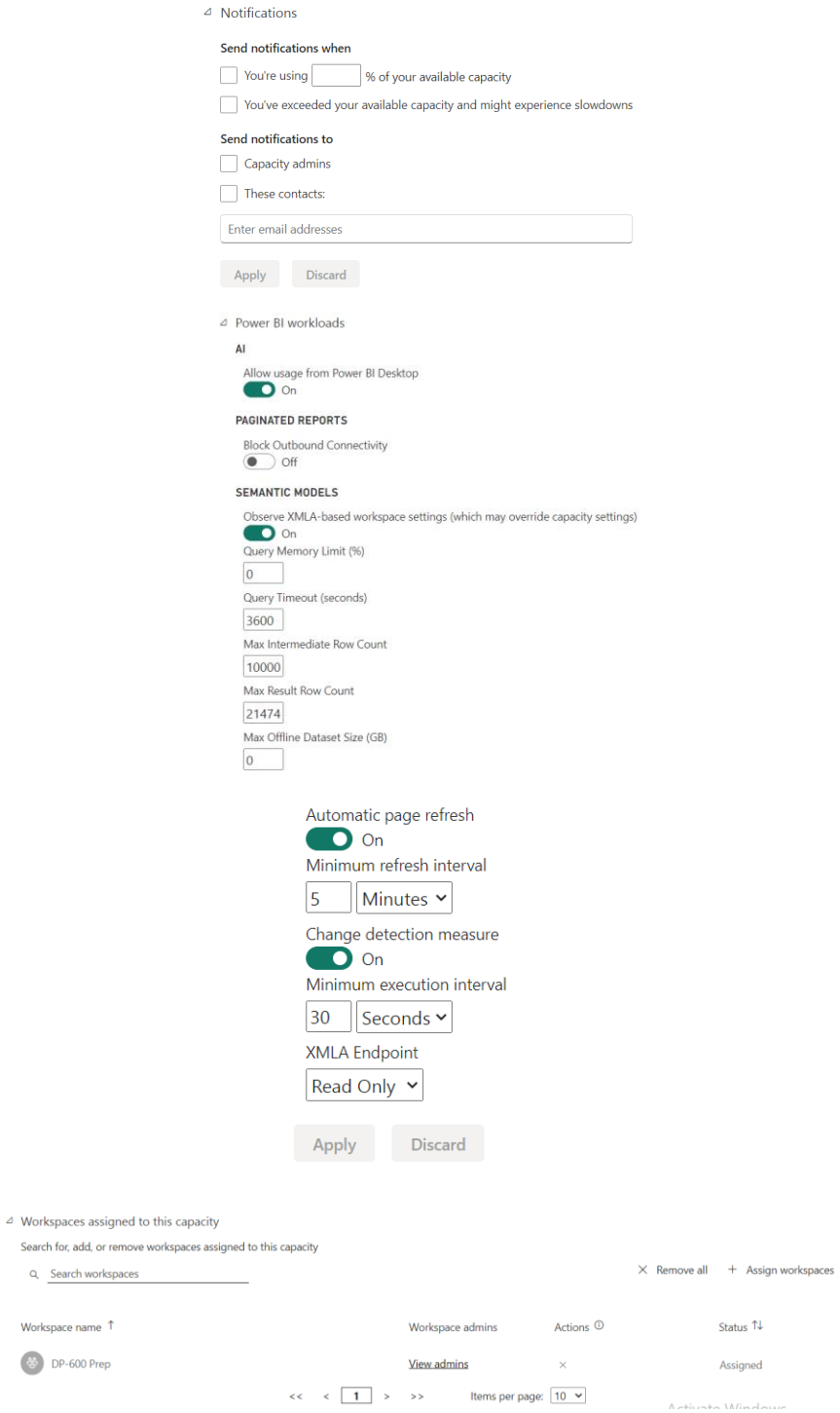
Topic 9: Manage Fabric capacity

Capacity Settings



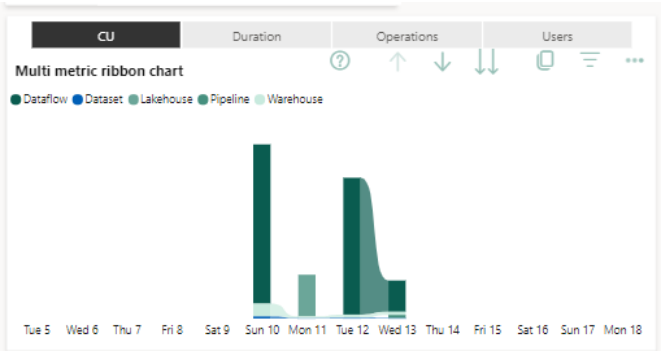
- **Azure SKUs –** F SKUs
- **Microsoft 365 SKUs –** P and EM SKUs. EM SKUs do not support Fabric
- Scaling allows you to increase or decrease the size of your capacity.
- Throttling occurs when a tenant's capacity consumes more capacity resources than it has purchased.
- Throttling is applied at the capacity level, meaning that while one capacity, or set of workspaces, may be experiencing reduced performance due to being overloaded, other capacities may continue running normally

Capacity Settings



Fabric capacity metrics app

Compute

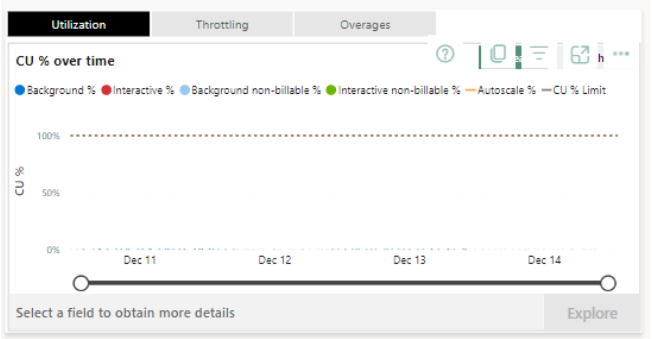


CU value is in seconds (CU processing time). Can be drilled down to the hour level

Utilization

Background operations cover backend processes that are not directly triggered by users, such as data refreshes.

Interactive operations cover a wide range of resources triggered by users. These operations are associated with interactive page loads.



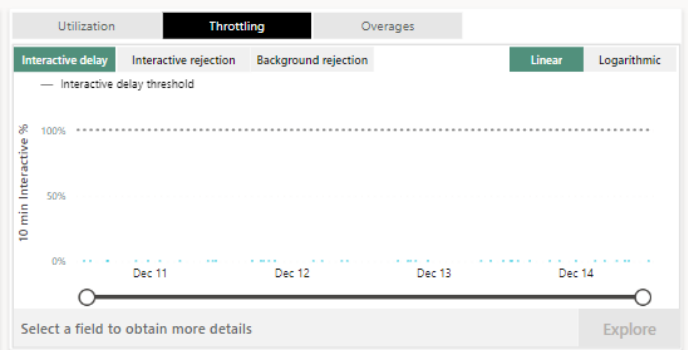
Throttling

Displays delay and rejection over time. Throttling is based on the **amount of future capacity consumption** resulting from the following smoothing policies.

Policy	Consumption	Impact
Overage protection	Usage <= 10 minutes	Jobs can consume 10 minutes of future capacity use without throttling.
Interactive delay	10 minutes < usage <= 60 minutes	User requested interactive jobs are throttled.
Interactive rejection	60 minutes < usage <= 24 hours	User requested interactive jobs are rejected.
Background rejection	Usage > 24 hours	User scheduled background jobs are rejected and not executed.

The throttling chart displays the following elements:

- **Interactive delay** - Interactive operations get delayed when *10 min Interactive %* smoothing crosses the *Interactive delay* threshold.
- **Interactive rejection** - Interactive operations get rejected when *60 min Interactive %* smoothing crosses the *Interactive rejection* threshold.
- **Background rejection** - Background operations get rejected when *24 hours Background %* smoothing crosses the *Background rejection* threshold



Overages

Displays the *add*, *burndown* and *cumulative* carryforward over time. Carryforward only takes into account billable operations.

The overages chart displays the following elements:

- **Add %** - Green columns represent the carryforward percent added during the current 30 second period.
- **Burndown %** - Blue columns represent the carryforward percent burned down for the current 30 second period.
- **Cumulative %** - Red line represent the cumulative carryforward percent for the current 30 second period. Cumulative percent is displayed on the secondary axis located on the right side of the visual.

Matrix by items and operation

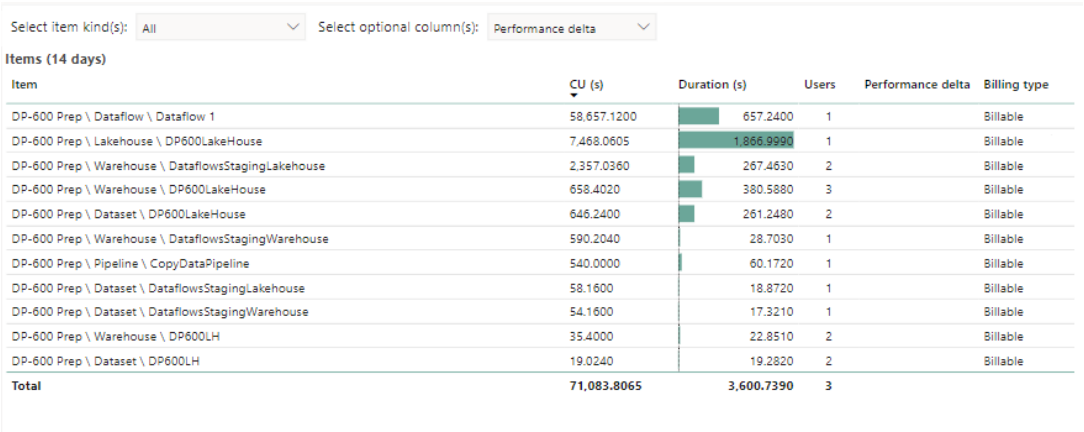
Performance delta - Displays the performance effect on the items. The number represents the percent of change from seven days ago. For example, 20 suggests that there's a 20% improvement today, compared with the same metric taken a week ago.

The colors in the matrix represent your performance delta:

- No color - A value higher than -10
- Orange - A value between -10 and -25
- Red - A value lower than -25

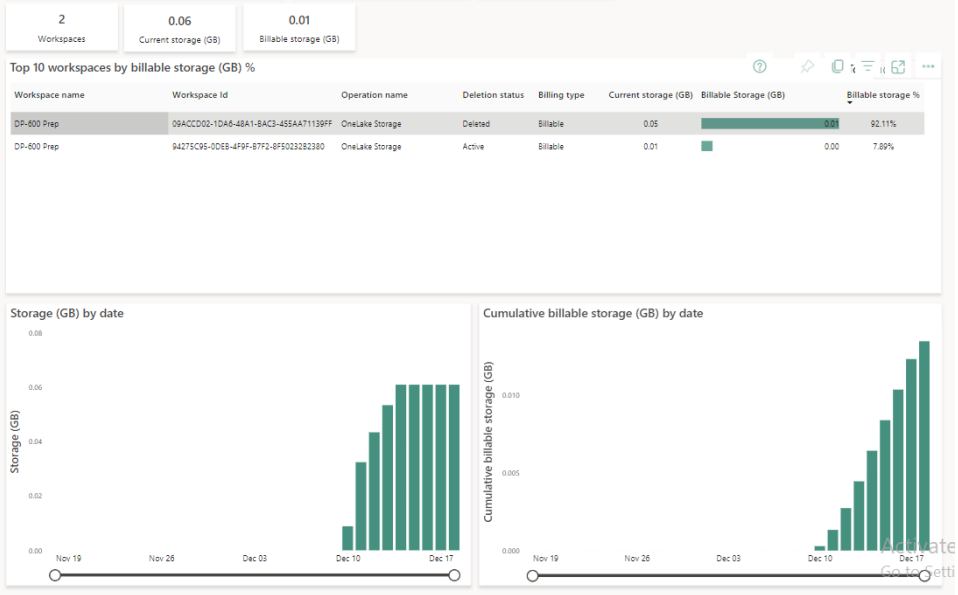
To create the performance delta Microsoft Fabric calculates an hourly average for all the fast operations that take under 200 milliseconds to complete. The hourly value is used as a slow moving average over the last seven days (168 hours). The slow moving average is then compared to the average between the most recent data point, and a data point from seven days ago. The performance delta indicates the difference between these two averages.

You can use the performance delta value to assess whether the average performance of your items improved or worsened over the past week. The higher the value is, the better the performance is likely to be. A value close to zero indicates that not much has changed, and a negative value suggests that the average performance of your items got worse over the past week.



Item	CU (s)	Duration (s)	Users	Performance delta	Billing type
DP-600 Prep \ Dataflow \ Dataflow 1	58,657.1200	657.2400	1		Billable
DP-600 Prep \ Lakehouse \ DP600LakeHouse	7,468.0605	1,866.9990	1		Billable
DP-600 Prep \ Warehouse \ DataflowsStagingLakehouse	2,357.0360	267.4630	2		Billable
DP-600 Prep \ Warehouse \ DP600LakeHouse	658.4020	380.5880	3		Billable
DP-600 Prep \ Dataset \ DP600LakeHouse	646.2400	261.2480	2		Billable
DP-600 Prep \ Warehouse \ DataflowsStagingWarehouse	590.2040	28.7030	1		Billable
DP-600 Prep \ Pipeline \ CopyDataPipeline	540.0000	60.1720	1		Billable
DP-600 Prep \ Dataset \ DataflowsStagingLakehouse	58.1600	18.8720	1		Billable
DP-600 Prep \ Dataset \ DataflowsStagingWarehouse	54.1600	17.3210	1		Billable
DP-600 Prep \ Warehouse \ DP600LH	35.4000	22.8510	2		Billable
DP-600 Prep \ Dataset \ DP600LH	19.0240	19.2820	2		Billable
Total	**71,083.8065**	**3,600.7390**	**3**		

Storage

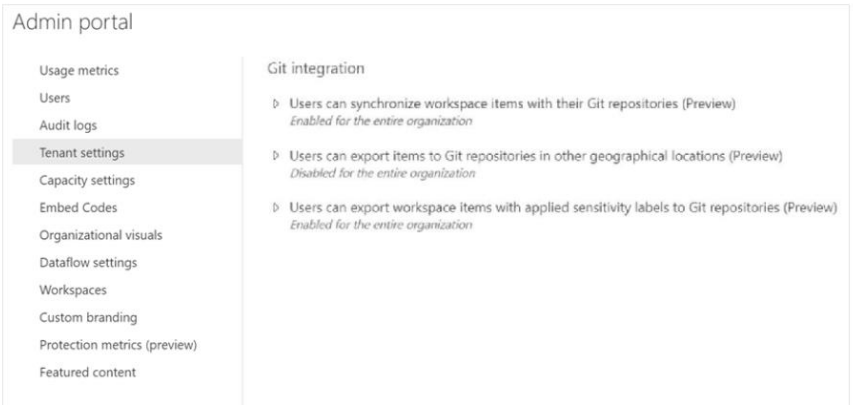


Topic 10: Implement version control for a workspace

Version control with Git allows:

- track changes made to Fabric items
- revert to older versions of an item
- Multiple users can collaborate on the same Fabric item (for example a Power BI report), at the same time, and their changes can be merged together (assuming no conflicts).
- Implement a check and approval process for approving changes made to Fabric items.

Admin portal -> Tenant settings -> Git integration



- Users can synchronize workspace items with their Git repositories (Preview)
  - Users can synchronize a workspace with a git repository, edit their workspace, and update their git repos using the git integration tool.
- Users can export items to Git repositories in other geographical locations (Preview)
  - If a workspace capacity is in one geographic location (for example, Central US) while the Azure DevOps repo is in another location (for example, West Europe), the Fabric admin can decide whether to allow users to commit metadata (or perform other git actions) to another geographical location. Only the metadata of the item is exported. Item data and user related information are not exported.
- Users can export workspace items with applied sensitivity labels to Git repositories (Preview)
  - Sensitivity labels aren't included when exporting an item. Therefore, the Fabric admin can choose whether to block the export of items that have sensitivity labels, or to allow it even though the sensitivity label won't be included.

Setting up Azure DevOps

- **Organization** - An organization in Azure DevOps is a mechanism for organizing and connecting groups of related projects.
- **Project** - A project in Azure DevOps contains the following set of features:
    - Boards and backlogs for agile planning
    - Pipelines for continuous integration and deployment
    - Repos for version control and management of source code and artifacts
    - Continuous test integration throughout the project life cycle. Each organization contains one or more projects
- The goal is
    - To link the Azure DevOps Git repository with a local folder that has the Power BI project files (PBIP).
    - To link the Fabric workspace with the Git repository in Azure DevOps

How to setup Git integration for local PBIP files

1. Create an Azure DevOps project.
2. Add a readme.md file to initialize the **main branch**
3. Copy .git url and clone it locally using GitHub Desktop or VS code.
4. Open the local git repo (folder) and save the PBIP files in this folder.
5. Create a new **feature branch**.
6. Commit and add a **pull request**.
7. Through an approval process merge the changes in the main branch.
8. Process from serial 5 to 7 is repeated every time changes are made to the PBIX file.

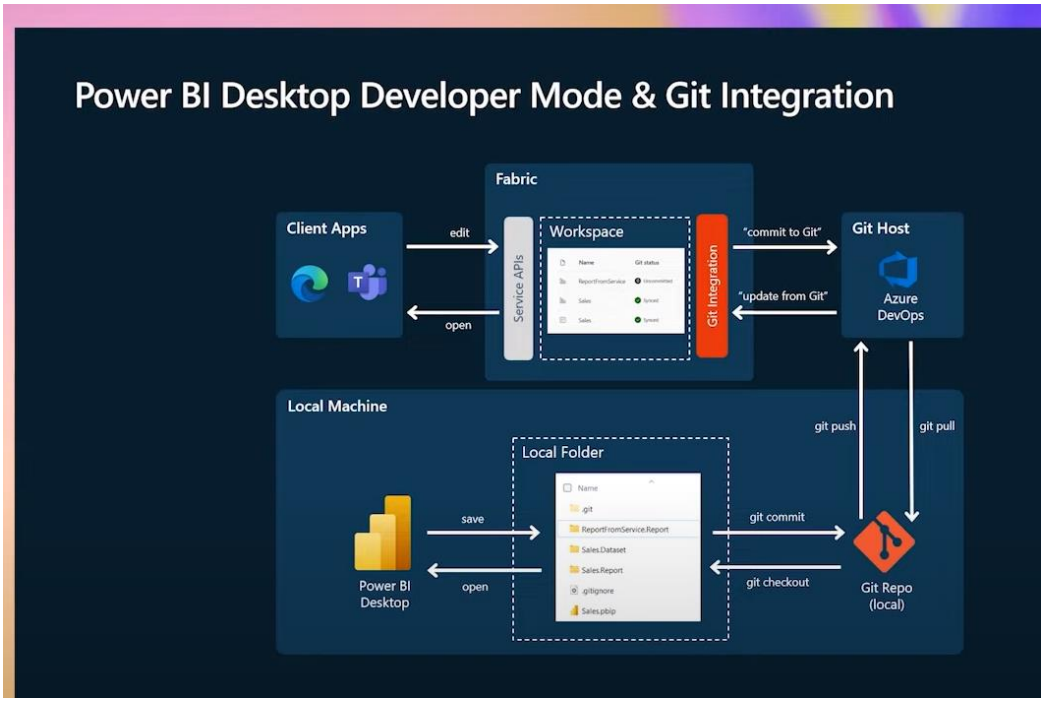
How to set up Git Integration for a workspace

1. Go to workspace settings -> Git Integration
2. Sync Azure DevOps account
3. Connect / sync Git repository and branch
4. Create a new item in workspace (like pipeline)
5. Create a **new checkout branch** and commit to merge with Azure DevOps.
6. For any uncommitted changes, the source control icon can be used to commit the changes.

2. Select the **Source control** icon. This icon shows the number of uncommitted changes.

3. Select the **Changes** tab of the **Source control** pane. A list appears with all the items you changed, and an icon indicating if the item is *new*, *modified*, *conflict*, or *deleted*.

- Synced (the item is the same in the workspace and Git branch)
- Conflict (the item was changed in both the workspace and Git branch)
- Unsupported item
- Uncommitted changes in the workspace
- Update required from Git
- Item is identical in both places but needs to be updated to the last commit



- The integration with source control is on a workspace level. Developers can version items they develop within a workspace in a single process, with full visibility to all their items.
- The following items are currently supported:
    - Lakehouse
    - Notebooks
    - Paginated reports
    - Reports (except reports connected to semantic models hosted in Azure Analysis Services, SQL Server Analysis Services or reports exported by Power BI Desktop that depend on semantic models hosted in MyWorkspace)
    - Semantic models (except push semantic models, live connections, model v1, and semantic models created from the Data warehouse/lakehouse.)
- **Only a workspace admin can connect a workspace to an Azure Repo, but once connected, anyone with permission can work in the workspace.**
- **You can only connect a workspace to one branch and one folder at a time.**
- **Only a workspace admin can disconnect a workspace from an Azure Repo**.

Topic 11: Create and manage a Power BI Desktop project (.pbip)

- Power BI Desktop introduces a new way to author, collaborate, and save your projects. You can now save your work as a **Power BI Project** (PBIP). As a project, report and semantic model *artifact* definitions are saved as individual plain text files in a simple, intuitive folder structure.
- **Text editor support** - Artifact definition files are JSON formatted text files containing model semantic model and report metadata. They're publicly documented and human readable.
- **Programmatic generation and editing artifact definitions** - You can create scripts using the popular and easy to use **Tabular Model Scripting Language (TMSL)** , or create your own custom applications to make changes to your artifact definitions. Applications can be based on public documentation of the artifact definition schemas and/or client libraries.
- **Source control** - Power BI semantic model and report artifact definitions can be stored in a source control system, like Git. With Git, you can track version history, compare revisions (diff), and revert to previous versions. Source control can also unblock collaboration when using Power BI Desktop by using familiar collaboration mechanisms for resolving conflicts (merge) and reviewing changes (pull requests).
- **Continuous Integration and Continuous Delivery (CI/CD)** - You can use systems where developers in your organization submit a proposed change to the CI/CD system. The system then validates the change with a series of *quality gates* before applying the change to the production system. These quality gates can include code reviews by other developers, automated testing, and automated build to validate the integrity of the changes. CI/CD systems are typically built on top of existing source control systems.

The PBIP structure has the following components

- **<project name>.pbip**
  - The PBIP file contains a pointer to a report folder, opening a PBIP opens the targeted report and model for authoring.

```
├── project
│   ├── AdventureWorks-Sales.Report
│   │   └── definition.pbir
│   ├── AdventureWorks-Stocks.Report
│   │   └── definition.pbir
│   ├── AdventureWorks.SemanticModel
│   │   └── *.*
│   .gitignore
└──  AdventureWorks.pbip
```

- **<project name>.Semantic model**
  - A collection of files and folders that represent a Power BI semantic model. It contains some of the most important files you're likely to work on, like model.bim.
- **<project name>.Report**
  - A collection of files and folders that represent a Power BI report.
- **.gitIgnore**
  - Specifies intentionally untracked files Git should ignore. Power BI Desktop creates the .gitignore file in the root folder when saving if it doesn't already exist. Semantic model and report subfolders each have default git ignored files specified in .gitIgnore:

```
├── project
│   ├── *.SemanticModel
│   │   ├── .pbi
│   │   │   ├── localSettings.json
│   │   │   └── cache.abf
│   ├── *.Report
│   │   ├── .pbi
│   │   │   └── localSettings.json
```

Topic 12: Plan and implement deployment solutions

Deployment pipelines enable creators to develop and test content in the service before it reaches the users

Development

- The first stage in deployment pipelines where you upload new content with your fellow creators. You can design build, and develop here, or in a different stage.

Test

- After you've made all the needed changes to your content, you're ready to enter the test stage. Upload the modified content so it can be moved to this test stage. Here are three examples of what can be done in the test environment:
  - Share content with testers and reviewers
  - Load and run tests with larger volumes of data
  - Test your app to see how it will look for your end users

Production

- After testing the content, use the production stage to share the final version of your content with business users across the organization.

You must an admin of a workspace to use deployment pipelines

Step 1: Create a deployment pipeline

1. A pipeline can be created from the deployment pipeline tab or from within a workspace

Step 2: Assign a workspace

- After creating a pipeline, you need to add the content you want to manage to the pipeline. Adding content to the pipeline is done by assigning a workspace to the pipeline stage. You can assign a workspace to any stage.

Step 3: Deploy to an empty stage

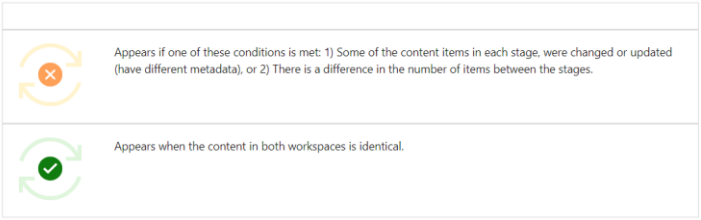
- Any Pro user that's a member or admin in the source workspace, can deploy content to an empty stage (a stage that doesn't contain content). The workspace must reside on a capacity for the deployment to be completed.
- You can also use the deployment pipelines REST APIs to programmatically perform deployments.
- Deployment pipelines offers three options when it comes to deploying your Power BI content:
    i. Deploying all content - Deploy all your Power BI content to the target stage.
    ii. Selective deployment - Select which Power BI content to deploy to the target stage. Dependency of items is critical in this operation
    iii. Backwards deployment - Deploy your content to a previous stage in the pipeline which must be empty. Selective deployment cannot be done in this scenario

Step 4: Create deployment rules

- Data sources rules and parameter rules can be configured in test and deployment workspace only like changing source datasets or changing lakehouses etc.

Step 5: Deploy content from one stage to another

- When reviewing the test and production stage cards, you can see the last deployment time.
- **Deployment time** is useful for establishing when a stage was last updated. It can also be helpful if you want to track time between test and production deployments.
- When two sequential stages have content, the content is compared based on the content item's metadata.
- Using the Compare view option helps you identify and track changes between items in each pipeline stage.



As well as the Deployment Pipelines functionality there are other ways to manage deployment:

- Can be managed through Branching
- Can be managed through Azure DevOps Pipelines (YAML templates)
- For semantic models, you can do it using the XMLA endpoint

Topic 13: Deploy and manage semantic models by using the XMLA endpoint

- The XMLA endpoint is a way to connect your fabric workspace with third-party tools. You can find the XMLA endpoint address in the **Workspace Settings**.
- You can use the XMLA endpoint to deploy semantic models from third-party tools, into your Fabric workspace.
- All Fabric workspaces have an XMLA endpoint, regardless of your capacity (this is unlike the old Power BI model where only Power BI Premium workspaces had an XMLA endpoint connection).
- Direct Lake models support write operations through the XMLA endpoint by using tools such as SQL Server Management Studio (19.1 and higher) , and the latest versions of external BI tools like Tabular Editor and DAX studio. Model write operations through the XMLA endpoint support:
    - Customizing, merging, scripting, debugging, and testing Direct Lake model metadata.
    - Source and version control, continuous integration and continuous deployment (CI/CD) with Azure DevOps and GitHub.
    - Automation tasks like refreshing, and applying changes to Direct Lake models by using PowerShell and REST APIs.
- Note that **XMLA Write operations are not supported with default semantic models**. You must create a separate Direct Lake semantic model, sometimes called standalone or **custom semantic model**.
- By default, endpoint connectivity is read only. This gives access to Model data, metadata, events and schema.

- Within Tenant settings, disabling XMLA endpoints ensure that semantic models can be connected to but not edited directly in workspaces.
- *Read write* settings can be enabled in Tenant settings.

Topic 14: Create and update reusable assets, including Power BI template (.pbit) files, Power BI data source (.pbids) files, and shared semantic models

Power BI Template (.PBIT)

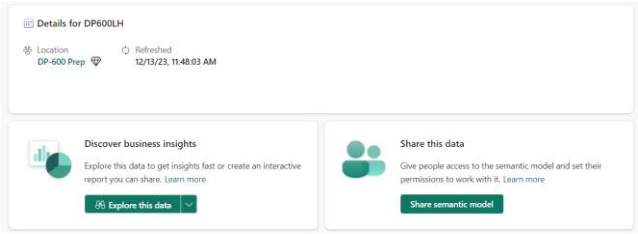
- Export -> Power BI Template
- Power BI report templates contain the following information from the report from which they were generated:
    - Report pages, visuals, and other visual elements
    - The data model definition, including the schema, relationships, measures, and other model definition items
    - All query definitions, such as queries, Query Parameters, and other query elements
- What is not included in templates is the report's data.

Power BI Data Source (.PBIDS)

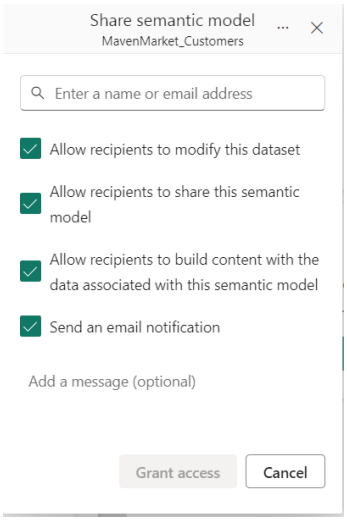
- PBIDS files are Power BI Desktop files that have a specific structure, and they have a .PBIDS extension to identify it is a Power BI data source file.
- Currently, PBIDS files only support a single data source in one file. Specifying more than one data source results in an error.
- To create the PBIDS file, select File > Options and settings > Data source settings:
- In the dialog that appears, select the data source you want to export as a PBIDS, and then select Export PBIDS.
- When an author opens a PBIDS file, Power BI Desktop opens and prompts the user for credentials to authenticate and connect to the data source that's specified in the file

Shared Semantic Models

- Default semantic models in Fabric items like Lakehouse and Warehouse cannot be shared



- The following options are available on the semantic model



    - Allow allow recipients to modify this semantic model
    - Allow recipients to share this semantic model - This option allows the recipients to grant access to other users via sharing.
    - Allow recipients to build content with the data associated with this semantic model - This option grants the recipients Build permission on the semantic model, which enables them to build new reports and dashboards based on the data associated it. If you clear this checkbox, the user will get read-only permission on the

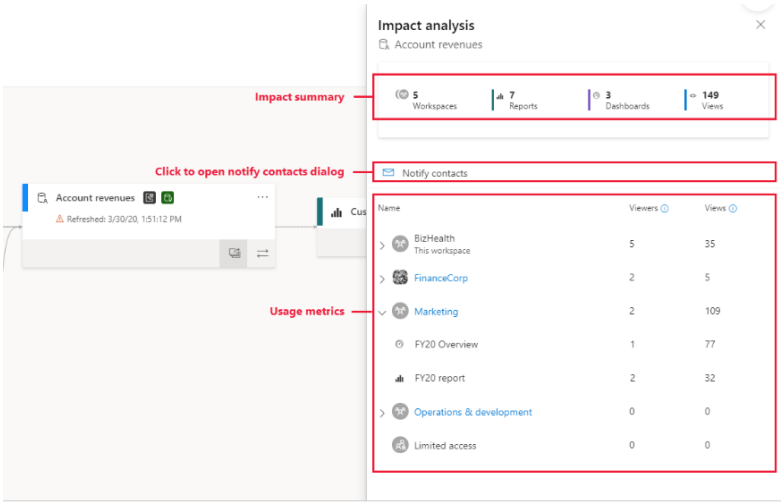
semantic model. Read-only permission allows them to explore the semantic model on the semantic model's info page but doesn't allow them to build new content based on the semantic model.

- Send an email notification to recipients.

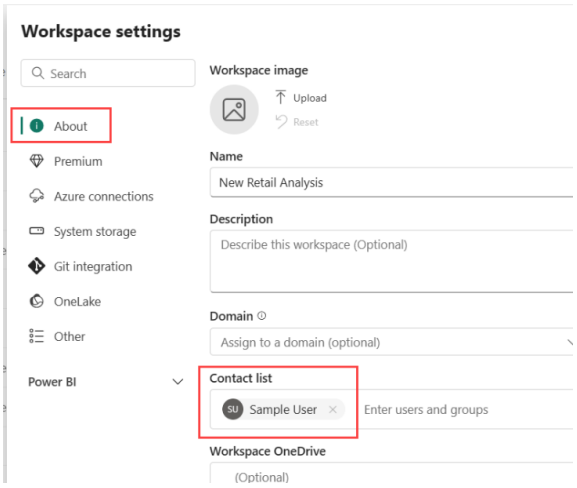
Topic 15: Perform impact analysis of downstream dependencies from lakehouses, data warehouses, dataflows, and semantic models

- Semantic model impact analysis is easily launched from within data lineage view

- You **cannot perform impact analysis on external semantic models** that are displayed in lineage view but are in fact located in another workspace.



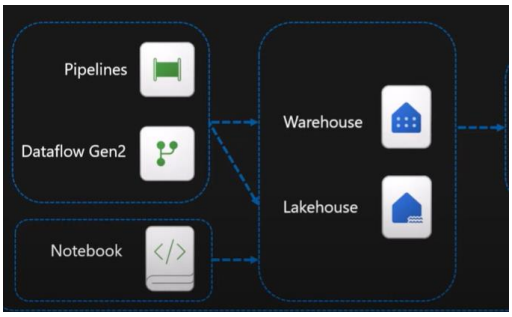
- In order to perform impact analysis on an item, you **must have write permissions** to it.
- Notify contact sends an email to the contact lists of all the impacted workspaces.
- The pane may say that there are more items with limited access. This occurs when there are affected items that you don't have access permissions for.
- Items that you don't have access to are listed as **Limited access**.
- Even if you don't have access to some workspaces, **your notify contacts messages will still reach the contact lists of those workspaces.**



Prepare and serve data (40–45%)

Topic 16: Ingest data by using a data pipeline, dataflow, or notebook

Topic 17: Copy data by using a data pipeline, dataflow or notebook



Lakehouse

- Data Lake + Data warehouse
- (Files) + (Tables) -> Parquet partitioned + Delta partitioned
- Delta Tables are Parquet data files + delta_log_folder. Delta_log folder contains details of transactions on the table and contains json files
- Named items created
    - Lakehouse
    - Semantic Model
        - Can be analysed with SQL Server profiler
        - Scripting can be done via Object explorer in SSMS using XMLA endpoint
    - SQL endpoint (read only)

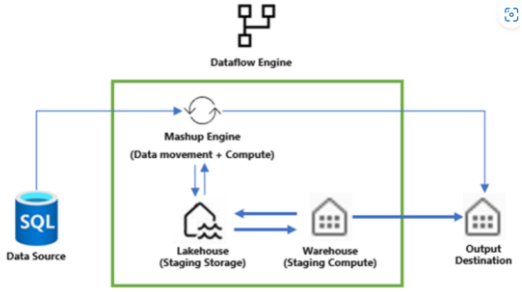
Dataflows Gen 2

Export template

- Power Query Template (PQT) can be used to export Dataflow Gen 1 to Dataflow Gen 2

Staging Queries

- Blue accent bar -> Staging enabled
- Grey accent bar -> Staging disabled
- Lighting icon -> Both queries are staged and enhanced compute engine will be used for refresh.



Additional tables created in Model View

- Exec_requests_history
- Frequently_run_queries
- Long_running_queries

Data Destinations

- Azure SQL DB
- Azure Data explorer (Kusto)
- Fabric LH
- Fabric WH
- Fabric KQL

Update method

- Append
- Replace
    - Dynamic schema -> tables dropped & recreated on refresh
    - Fixed schema -> Fixed schema
- For DWH only fixed schema is supported

Limitations

- Incremental refresh is not available

Data pipeline

Data can land into tables or files area

Activities

- Data Transformation
    - Copy Data
    - Dataflow Activities
    - Notebook activities
    - Stored procedure activities
    - Delete data activities
    - SQL script activities
- Control
    - Implement loops
    - Conditional formatting
    - Manage variables and parameter values

Copy data activity

Type conversion settings

Setting	Description
Intelligent throughput optimization	Specify to optimize the throughput. You can choose from: • **Auto** • **Standard** • **Balanced** • **Maximum** When you choose **Auto**, the optimal setting is dynamically applied based on your source-destination pair and data pattern. You can also customize your throughput, and custom value can be 2-256 while higher value implies more gains.
Degree of copy parallelism	Specify the degree of parallelism that data loading would use.
Fault tolerance	When selecting this option, you can ignore some errors occurred in the middle of copy process. For example, incompatible rows between source and destination store, file being deleted during data movement, etc.
Enable logging	When selecting this option, you can log copied files, skipped files and rows
Enable staging	Specify whether to copy data via an interim staging store. Enable staging only for the beneficial scenarios.
Staging account connection	When selecting **Enable staging**, specify the connection of an Azure storage data source as an interim staging store. Select + **New** to create a staging connection if you don't have it.

Features

Ability to create parameters, variables, functions and build expressions using expression builder.

Limitations

For data ingestion, **no cross-workspace capability**.

Notebook

Use SQL or Python to perform ETL operation on very large datasets

SQL

```
--Copy data from the public Azure storage account to the dbo.dimension_city table.
COPY INTO [dbo].[dimension_city]
FROM
'https://azuresynapsestorage.blob.core.windows.net/sampledata/WideWorldImportersDW/tables/
dimension_city.parquet'
WITH (FILE_TYPE = 'PARQUET');

--Copy data from the public Azure storage account to the dbo.fact_sale table.
COPY INTO [dbo].[fact_sale]
FROM
'https://azuresynapsestorage.blob.core.windows.net/sampledata/WideWorldImportersDW/tables/
fact_sale.parquet'
WITH (FILE_TYPE = 'PARQUET');

CREATE TABLE dbo.DimProduct
(
    ProductKey INTEGER NOT NULL,
    ProductAltKey VARCHAR(25) NULL,
    ProductName VARCHAR(50) NOT NULL,
    Category VARCHAR(50) NULL,
    ListPrice DECIMAL(5,2) NULL
);
GO

INSERT INTO dbo.DimProduct
VALUES
(1, 'RING1', 'Bicycle bell', 'Accessories', 5.99),
(2, 'BRITE1', 'Front light', 'Accessories', 15.49),
(3, 'BRITE2', 'Rear light', 'Accessories', 15.49);
GO
```

PySpark

Steps

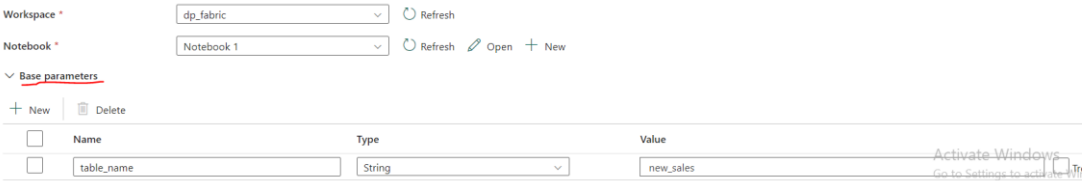
- Connect to Data source
- Perform authentication
- Read data into DF
- Perform ETL transformations on DF
- Either write DF to Parquet File format or Write DF to Delta Table

```
# Azure Blob Storage access info
blob_account_name = "azureopendatastorage"
blob_container_name = "nyctlc"
blob_relative_path = "yellow"

# Construct connection path
wasbs_path =
f'wasbs://{blob_container_name}@{blob_account_name}.blob.core.windows.net/{blob_relative_p
ath}'
print(wasbs_path)

# Read parquet data from Azure Blob Storage path
blob_df = spark.read.parquet(wasbs_path)

# Declare file name
file_name = "yellow_taxi"

# Construct destination path ABFS
```

```
output_parquet_path = f"**InsertABFSPathHere**/{file_name}"
print(output_parquet_path)

# Load the first 1000 rows as a Parquet file
blob_df.limit(1000).write.mode("overwrite").parquet(output_parquet_path)

# writing dataframes as CSV
df_csv.write.mode("overwrite").format("csv").save("Files/ " + csv_table_name)

# writing dataframes as json
df_csv.write.json("Files/json/property-sales.json", mode='overwrite')

# 4 different write modes
# append the new dataframe to the existing Table
df.write.mode("append").format("delta").saveAsTable(delta_table_name)

# overwrite existing Table with new DataFrame
df.write.mode("overwrite").format("delta").saveAsTable(delta_table_name)

# Throw error if data already exists
df.write.mode("error").format("delta").saveAsTable(delta_table_name)

# Fail silently if data already exists
df.write.mode("ignore").format("delta").saveAsTable(delta_table_name)
```

Toggle parameter cell -> This configures the cell so that the variables declared in it are treated as parameters when running the notebook from a pipeline.

Passing parameters



Limitation

- Using notebooks, currently **cannot write to a data warehouse**.

Topic 18: Choose an appropriate method for copying data from a Fabric data source to a lakehouse or warehouse

Use case	Recommendation
Small file upload from local machine	Use Local file upload
Small data or specific connector	Use Dataflows
Large data source	Use Copy tool in pipelines
Complex data transformations	Use Notebook code

Copy activity is the best low-code and no-code choice to move petabytes of data to lakehouses and warehouses from varieties of sources, either ad-hoc or via a schedule.

Topic 19: Create and manage shortcuts

- A shortcut enables you to create a live link to data stored either in another part of Fabric (internal shortcut) or in external storage locations:
    - ADLS Gen2
    - Amazon S3
    - Other services that USE Amazon S3 for storage (like Cloudflare)
    - Google Cloud Storage
    - Dataverse
- Shortcuts can be setup for individual files, but more commonly to a folder. If you create a shortcut to a 'base' folder, it will monitor and sync all files in subfolders.
- Within a Lakehouse, shortcuts can be created in both tables and files. In the **Tables** folder, you can only create shortcuts at the top level. Shortcuts aren't supported in other subdirectories of the **Tables** folder. No such restriction for folders.
- **A shortcut cannot be created from a Warehouse**.
- For external shortcuts, egress fee is a factor to consider for cross region data movement.
- Shortcut permissions

Capability	Admin	Member	Contributor	Viewer
Create a shortcut	Yes[1]	Yes[1]	Yes[1]	-
Read file/folder content of shortcut	Yes[2]	Yes[2]	Yes[2]	-
Write to shortcut target location	Yes[3]	Yes[3]	Yes[3]	-
Read data from shortcuts in table section of the lakehouse via TDS endpoint	Yes	Yes	Yes	Yes

[1] Users must have a role that provides write permission the shortcut location and at least read permission in the target location.

[2] Users must have a role that provides read permission both in the shortcut location and the target location.

[3] Users must have a role that provides write permission both in the shortcut location and the target location.

- Apache Spark notebooks and Apache Spark jobs can use shortcuts that you create in OneLake. Relative file paths can be used to directly read data from shortcuts. Additionally, if you create a shortcut in the **Tables** section of the lakehouse and it is in the Delta format, you can read it as a managed table using Apache Spark SQL syntax.

```
df = spark.read.format("delta").load("Tables/MyShortcut")
display(df)

df = spark.sql("SELECT * FROM MyLakehouse.MyShortcut LIMIT 1000")
display(df)
```

- You can also read shortcuts in the **Tables** section of a lakehouse through the SQL analytics endpoint for the lakehouse. You can access the SQL analytics endpoint through the mode selector of the lakehouse or through SQL Server Management Studio (SSMS).

```
SELECT TOP (100) *
FROM [MyLakehouse].[dbo].[MyShortcut]
```

Topic 20: Create views, functions, and stored procedures

VIEWS

- Transformed data is not stored in a view, the result is calculated at query time.
- You can create a View from another View. This can lead to poor performance, and difficult to understand/ maintain
- You cannot specify parameters for a View (this is possible with functions and stored procedures)
- Reading a SQL View into a semantic model will fallback to DirectQuery (Direct Lake not possible because Delta tables do not exist for such a scenario).

```
CREATE VIEW  dbo.vw_Employee_GetJack AS
SELECT
    EmployeeId
    , Name
```

```
    , Age
from dbo.Employees
WHERE Name like 'Jack%';

select * from dbo.vw_Employee_GetJack
```

FUNCTIONS

- Useful for packaging up logic, which can be parameterized
- **Only SELECT statements, can never edit the underlying data (using UPDATE/ INSERT etc)**
- Can be called from DDL statements, or from within Stored Procedures, or a View
- Can't be orchestrated (directly) from a data pipeline (although can embed in a stored proc).
- Allows one or many input parameters
- Output type should also be a Table for T_SQL in Fabric

```
CREATE FUNCTION dbo.fn_Employee_FirstNameSearch (  @firstname varchar(20) = ''  )
RETURNS TABLE
AS
RETURN
(
    SELECT
        EmployeeId
        , Name
        , Age
    from dbo.Employees
    WHERE Name like @firstname + '%'
) ;

-- Test the function
select * from dbo.fn_Employee_FirstNameSearch('Sarah')
```

STORED PROCEDURES

- Ability to define input AND OUTPUT parameters
- Are called using EXEC (not as part of a select statement)
- Can call other stored procedures
- Can also give access to people to the Stored Procedure (and not the underlying dataset)
- Can be embedded in a data pipeline (with parameters passed)
- Ability to do INSERTS, UPDATES, DELETES

```
-- stored procedure with input parameter
CREATE PROCEDURE dbo.sp_Employee_GetByFirstName @firstname varchar(20)
as
select * from dbo.Employees
where Name like @firstname + '%'

exec dbo.sp_Employee_GetByFirstName 'Jack'

--Drop the stored procedure if it already exists.
DROP PROCEDURE IF EXISTS [dbo].[populate_aggregate_sale_by_city]
GO

--Create the populate_aggregate_sale_by_city stored procedure.
CREATE PROCEDURE [dbo].[populate_aggregate_sale_by_city]
AS
BEGIN
    --If the aggregate table already exists, drop it. Then create the table.
    DROP TABLE IF EXISTS [dbo].[aggregate_sale_by_date_city];
    CREATE TABLE [dbo].[aggregate_sale_by_date_city]
        (
            [Date] [DATETIME2](6),
            [City] [VARCHAR](8000),
            [StateProvince] [VARCHAR](8000),
            [SalesTerritory] [VARCHAR](8000),
            [SumOfTotalExcludingTax] [DECIMAL](38,2),
            [SumOfTaxAmount] [DECIMAL](38,6),
            [SumOfTotalIncludingTax] [DECIMAL](38,6),
```

```
            [SumOfProfit] [DECIMAL](38,2)
        );

    --Reload the aggregated dataset to the table.
    INSERT INTO [dbo].[aggregate_sale_by_date_city]
    SELECT
        FS.[InvoiceDateKey] AS [Date],
        DC.[City],
        DC.[StateProvince],
        DC.[SalesTerritory],
        SUM(FS.[TotalExcludingTax]) AS [SumOfTotalExcludingTax],
        SUM(FS.[TaxAmount]) AS [SumOfTaxAmount],
        SUM(FS.[TotalIncludingTax]) AS [SumOfTotalIncludingTax],
        SUM(FS.[Profit]) AS [SumOfProfit]
    FROM [dbo].[fact_sale] AS FS
    INNER JOIN [dbo].[dimension_city] AS DC
        ON FS.[CityKey] = DC.[CityKey]
    GROUP BY
        FS.[InvoiceDateKey],
        DC.[City],
        DC.[StateProvince],
        DC.[SalesTerritory]
    ORDER BY
        FS.[InvoiceDateKey],
        DC.[StateProvince],
        DC.[City];
END
```

Topic 21: Add stored procedures, notebooks, and dataflows to a data pipeline

Stored Procedure

- Add SP activity to data pipeline.
- Specify the source, workspace or external (Azure SQL database and Azure SQL DB Managed instance supported)
- Specify the stored procedure
- Configure parameters or import these parameters. There are some advanced settings on timeout or logging the input / output results

Notebooks

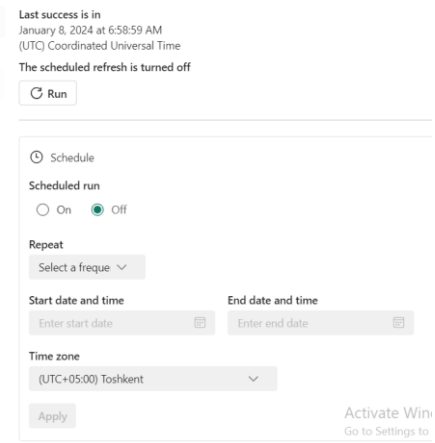
- Create notebook activity
- Specify the notebook to be used
- Specify the base parameters, if used and coming from other activities in the pipeline.

Dataflows

- Create dataflow activity
- Specify the workspace and dataflow

Topic 22: Schedule data pipelines

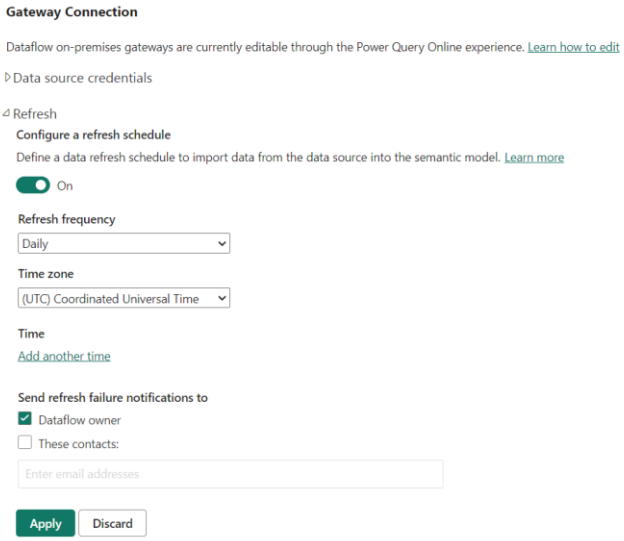
- A data pipeline run can be triggered one of two ways, either on-demand or by setting up a schedule.
- A scheduled pipeline will be able to run based on the time and frequency that you set. No limit on the frequency of scheduling just like a dataflow.
- Start datetime and End datetime is available for all the option settings within Repeat
- Repeat
    - By the minute (every xx minutes)
    - Hourly (every xx hours)
    - Daily (can specify multiple times here)
    - Weekly (Can select the days of the week as well as specify multiple times here)



Topic 23: Schedule dataflows and notebooks

Dataflows

- Incremental Refresh is not yet supported for Dataflows Gen 2, however, there are workarounds to get things done.
- Max number of refreshes per day are 48.



Notebooks

- Same options as for a data pipeline.
- **No refresh limit as in the case of dataflows**.

Topic 24: Implement a data cleansing process

- Creating, altering, and dropping tables, and insert, update, and delete are only supported in Warehouse in Microsoft Fabric, not in the SQL analytics endpoint of the Lakehouse.
- You can create your own T-SQL views, functions, and procedures on top of the tables that reference your Delta Lake data in the SQL analytics endpoint of the Lakehouse.
- Within Fabric, the data cleansing process is normally planned for the Silver layer, where data is cleaned, de-duplicated and ready for use.
- If you are using Dataflows and M script for data cleaning and transformation, here are some of the important functions.
    - Table.AddColumn -> Adds a new column to a table.
    - Table.TransformColumns -> Perform transformations on one or more columns in a table.
    - Table.SelectRows -> Used to filter rows from table based on a condition

- Table.Combine -> Used to append tables (union)
- Table.Join -> Used to merge tables (join) based on join type. Default is inner join
- Table.Group -> Used to groups the rows of a table by values in one or more columns and aggregate the data in the remaining columns.
- Value.NativeQuery -> To specify an SQL query for the database while connecting

Topic 25: Identify and resolve duplicate data, missing data, or null values

SQL

```
-- identifying duplicates
SELECT [Dealer_ID]
    ,[Date_ID]
    ,count([Revenue]) 'NumberOfRows'
FROM [LH_BRONZE].[dbo].[revenue]
GROUP BY [Dealer_ID],[Date_ID]
HAVING count([Revenue]) > 1

-- deduplication
SELECT [Dealer_ID]
    , [Date_ID]
    , max([Revenue]) 'Revenue'
FROM [LH_BRONZE].[dbo].[revenue]
GROUP BY [Dealer_ID] ,[Date_ID]

-- missing data/ nulls
-- identifying/ removing nulls / filtering with WHERE

with removed_nulls as (
SELECT [Dealer_ID]
    ,[Model_ID]
    ,[Branch_ID]
    ,[Date_ID]
    ,[Units_Sold]
    ,[Revenue]
FROM [LH_BRONZE].[dbo].[revenue]
WHERE Revenue IS NOT NULL
)

select count(*) from removed_nulls
select count(*) from [LH_BRONZE].[dbo].[revenue]
```

PySpark

```
#renaming existing columns
df = df.withColumnRenamed('Address ', 'Address')
df.select('Address').show()

# selecting a few columns
df.select(['Address','Type']).show()

#most basic/drastic drop NAs
df.na.drop().show()

# fill null values
df.fillna(value='Unknown Address',subset=["Address"]).show()

# drop duplicates
df_deduplicated = df.dropDuplicates()

# De-duplicate the DataFrame based on the "Name" column df_deduplicated =
df.dropDuplicates(["Name"])

# using cast, create a new column
type_conv = df.withColumn('UnitsSoldConverted', df.Units_Sold.cast("string"))
type_conv.printSchema()
display(type_conv)
```

Topic 26: Filter data

SQL

```
with removed_nulls as (
SELECT [Dealer_ID]
    ,[Model_ID]
    ,[Branch_ID]
    ,[Date_ID]
    ,[Units_Sold]
    ,[Revenue]
FROM [LH_BRONZE].[dbo].[revenue]
WHERE Revenue IS NOT NULL
)
```

PySpark

```
# simple filter condition (pythonic)
df.filter(df['City'] == "New York").show()

# not equal to
df.filter(df['City'] != "New York").show()

# De-duplicate the DataFrame based on the "Name" column
df_deduplicated = df.dropDuplicates(["Name"])

# Create temp SQL table
table_df.createOrReplaceTempView("yellow_taxi_temp")

# SQL Query
table_df = spark.sql('SELECT * FROM yellow_taxi_temp')

# Filter a dataframes
customers = df['CustomerName', 'Email']

# Filter a dataframe using select and where clause
customers = df.select("CustomerName", "Email").where(df['Item']=='Road-250 Red, 52')

# df.filter()
filtered_df = df.filter(col("Revenue") > 10000000)
print(f"Proces removed {df.count() - filtered_df.count()} rows from the dataset" )
```

Topic 27: Convert data types by using SQL or PySpark

SQL

```
-- type conversion / adding new columns
SELECT [Dealer_ID]
    , [Revenue]
    , CAST([Revenue] as FLOAT) as RevFloat
    , [Revenue] / 2 as HalfRevenue

FROM [LH_BRONZE].[dbo].[revenue]
```

PySpark

```
# Change the data type of the "Id" column from String to Integer
df = df.withColumn("Id", df["Id"].cast(IntegerType()))

from pyspark.sql.types import *

orderSchema = StructType([
   StructField("SalesOrderNumber", StringType()),
   StructField("SalesOrderLineNumber", IntegerType()),
   StructField("OrderDate", DateType()),
   StructField("CustomerName", StringType()),
   StructField("Email", StringType()),
   StructField("Item", StringType()),
   StructField("Quantity", IntegerType()),
   StructField("UnitPrice", FloatType()),
```

```
    StructField("Tax", FloatType())
    ])

df = spark.read.format("csv").schema(orderSchema).load("Files/orders/*.csv")
display(df)
```

Topic 28: Enrich data by adding new columns or tables

- Transaction history for delta tables is stored in JSON files in the **delta_log** folder. You can use this transaction log to manage data versioning.
- PRIMARY KEY is only supported when NONCLUSTERED and NOT ENFORCED are both used.
- FOREIGN KEY is only supported when NOT ENFORCED is used
- UNIQUE constraint is only supported when NONCLUSTERED and NOT ENFORCED is used.
- Identity columns are not supported for Fabric Warehouses. Instead a workaround with ROW_NUMBER() and max of the ID column is used.

SQL

```
-- create clones from a part point in time
CREATE TABLE [dbo].[fact_sale2] AS CLONE OF [dbo].[fact_sale] AT '2024-04-
29T23:51:48.923';

CREATE TABLE [dbo].[dimension_city2] AS CLONE OF [dbo].[dimension_city] AT '2024-04-
29T23:51:48.923';

-- create delta tables
CREATE TABLE products
USING DELTA
LOCATION 'Files/external_products';

-- use history for delta tables
UPDATE products
SET ListPrice = ListPrice * 0.9
WHERE Category = 'Mountain Bikes';

DESCRIBE HISTORY products;
```

PySpark

```
# adding new column
df = df.withColumn('2x_SalePrice', df['SalePrice ($)'] * 2)

## Add month and year columns
df = df.withColumn("Year", year(col("OrderDate"))).withColumn("Month",
month(col("OrderDate")))

## Create Year and Month columns
transformed_df = df.withColumn("Year", year(col("OrderDate"))).withColumn("Month",
month(col("OrderDate")))

# Create the new FirstName and LastName fields
transformed_df = transformed_df.withColumn("FirstName", split(col("CustomerName"), "
").getItem(0)).withColumn("LastName", split(col("CustomerName"), " ").getItem(1))
```

Topic 29: Merge or join data

SQL

```
-- using case statements
-- windows functions
WITH CategorizedSales AS (
 SELECT
     CASE
         WHEN i.ItemName LIKE '%Helmet%' THEN 'Helmet'
         WHEN i.ItemName LIKE '%Bike%' THEN 'Bike'
         WHEN i.ItemName LIKE '%Gloves%' THEN 'Gloves'
         ELSE 'Other'
```

```
    END AS Category,
    c.CustomerName,
    s.UnitPrice * s.Quantity AS Sales
FROM Sales.Fact_Sales s
JOIN Sales.Dim_Customer c
ON s.CustomerID = c.CustomerID
JOIN Sales.Dim_Item i
ON s.ItemID = i.ItemID
WHERE YEAR(s.OrderDate) = 2021
),
RankedSales AS (
    SELECT
        Category,
        CustomerName,
        SUM(Sales) AS TotalSales,
        ROW_NUMBER() OVER (PARTITION BY Category ORDER BY SUM(Sales) DESC) AS SalesRank
    FROM CategorizedSales
    WHERE Category IN ('Helmet', 'Bike', 'Gloves')
    GROUP BY Category, CustomerName
)
SELECT Category, CustomerName, TotalSales
FROM RankedSales
WHERE SalesRank = 1
ORDER BY TotalSales DESC;
```

Python

```
# Perform an inner join on the "Id" column
df_joined = df1.join(df2, on="Id", how="inner")

joined_df = (
    dealers_df
        .join(countries_df, dealers_df.Country_ID == countries_df.Country_ID)
        .select(dealers_df.Dealer_ID, dealers_df.Country_ID, countries_df.Country_Name)
    )
display(joined_df)
```

Topic 30: Implement a star schema for a lakehouse or warehouse, including Type 1 and Type 2 slowly changing dimensions

Type 1 SCD

A **Type 1 SCD** always reflects the latest values, and when changes in source data are detected, the dimension table data is overwritten.

CustomerID	FirstName	LastName	EmailAddress	CompanyName	InsertedDate	ModifiedDate
2	Keith	Harris	keith0@aw.com	Progressive Sports	2021-03-20	2021-03-20
3	Donna	Carreras	donna0@aw.com	A Bike Store	2021-03-20	2021-03-20

CustomerID	FirstName	LastName	EmailAddress	CompanyName	InsertedDate	ModifiedDate
2	Keith	Harris	keith0@aw.com	Progressive Sports	2021-03-20	2021-03-20
3	Donna	Carreras	donna0@aw.com	Bikes, Bikes, Bikes	2021-03-20	2021-03-22

Type 2 SCD

A **Type 2 SCD** supports versioning of dimension members. Often the source system doesn't store versions, so the data warehouse load process detects and manages changes in a dimension table. In this case, the dimension table must use a surrogate key to provide a unique reference to a version of the dimension member. It also includes columns that define the date range validity of the version.

SalesRepID	RepSourceId	FirstName	LastName	Region	StartDate	EndDate	IsCurrent
1	312	Jun	Cao	Southwest	2021-03-20	9999-12-31	True
2	331	Susan	Eaton	Southcentral	2021-03-20	9999-12-31	True

SalesRepID	RepSourceId	FirstName	LastName	Region	StartDate	EndDate	IsCurrent
1	312	Jun	Cao	Southwest	2021-03-20	9999-12-31	True
2	331	Susan	Eaton	Southcentral	2021-03-20	2021-03-21	False
3	331	Susan	Eaton	Southeast	2021-03-22	9999-12-31	True

Implementation of Type 1 SCD

1. Update records where id is present in dim table
2. Only insert records in dim table where id is not already present

Implementation of Type 2 SCD

1. Update records for end date where id is already present in dim table AND EndDate has high value or IsCurrent = True.
2. Insert records

SQL

Lakehouse

- Delta log property can be used to capture history at any point in time.
- MERGE INTO can be used to update the Delta tables

```
MERGE INTO people10m
USING people10mupdates
ON people10m.id = people10mupdates.id
WHEN MATCHED THEN
  UPDATE SET
    id = people10mupdates.id,
    firstName = people10mupdates.firstName,
    middleName = people10mupdates.middleName,
    lastName = people10mupdates.lastName,
    gender = people10mupdates.gender,
    birthDate = people10mupdates.birthDate,
    ssn = people10mupdates.ssn,
    salary = people10mupdates.salary
WHEN NOT MATCHED
  THEN INSERT (
    id,
    firstName,
    middleName,
    lastName,
    gender,
    birthDate,
    ssn,
    salary
  )
  VALUES (
    people10mupdates.id,
    people10mupdates.firstName,
    people10mupdates.middleName,
    people10mupdates.lastName,
    people10mupdates.gender,
    people10mupdates.birthDate,
    people10mupdates.ssn,
    people10mupdates.salary
  )
```

Warehouse

- MERGE operation is not supported for warehouse currently.

- Best method is to use stored procedures for implementing the SCD Type 1 and Type 2 operations
- Another option is to use dataflows for implementing this.

Topic 31: Implement bridge tables for a lakehouse or a warehouse

```
insert into projects VALUES (1,'First Project', '100,101'), (2, 'Second Project',
'101,102,103');

insert into employees VALUES (100,'Edwin Monty'), (101,'John Smith'),
(102, 'Sarah Hardy'),
(103, 'Jimmy Zhang')

-   Create a bridge table
-   The string_split is going to split the comma separated values and cross apply will
-   Create a one to one mapping against project_id and employee_id
create view dbo.vw_BRIDGE_project_participants as
SELECT project_id, CAST(value as INT) as employee_id
FROM dbo.projects
    CROSS APPLY STRING_SPLIT(project_participants, ',');
```

Topic 32: Aggregate or de-aggregate data

SQL

```
-- aggregate data using group by
SELECT
        FS.[InvoiceDateKey] AS [Date],
        DC.[City],
        DC.[StateProvince],
        DC.[SalesTerritory],
        SUM(FS.[TotalExcludingTax]) AS [SumOfTotalExcludingTax],
        SUM(FS.[TaxAmount]) AS [SumOfTaxAmount],
        SUM(FS.[TotalIncludingTax]) AS [SumOfTotalIncludingTax],
        SUM(FS.[Profit]) AS [SumOfProfit]
    FROM [dbo].[fact_sale] AS FS
    INNER JOIN [dbo].[dimension_city] AS DC
        ON FS.[CityKey] = DC.[CityKey]
    GROUP BY
        FS.[InvoiceDateKey],
        DC.[City],
        DC.[StateProvince],
        DC.[SalesTerritory]
    ORDER BY
        FS.[InvoiceDateKey],
        DC.[StateProvince],
        DC.[City];
```

PySpark

```
# Perform aggregation: Sum of salaries by department
df_grouped = df.groupBy("Department").agg(sum("Salary").alias("Total_Salary"))

# Order the result by Total_Salary in descending order
df_ordered = df_grouped.orderBy(df_grouped["Total_Salary"].desc())

# maximum sales price for each agent
df.groupBy('Agent')\
  .max('SalePrice_USD')\
  .show()

# method 2: using agg() and then alias()
df.groupBy("Agent") \
  .agg(max('SalePrice_USD').alias('max_sales_price'))\
  .show()

# group by multiple columns
df.groupBy(['City', 'Agent']).avg('SalePrice_USD').show()
```

Topic 33: De-normalize data

```
CREATE OR ALTER PROCEDURE Sales.LoadDataFromStaging (@OrderYear INT)
 AS
 BEGIN
      -- Load data into the Customer dimension table
     INSERT INTO Sales.Dim_Customer (CustomerID, CustomerName, EmailAddress)
     SELECT DISTINCT CustomerName, CustomerName, EmailAddress
     FROM [Sales].[Staging_Sales]
     WHERE YEAR(OrderDate) = @OrderYear
     AND NOT EXISTS (
         SELECT 1
         FROM Sales.Dim_Customer
         WHERE Sales.Dim_Customer.CustomerName = Sales.Staging_Sales.CustomerName
         AND Sales.Dim_Customer.EmailAddress = Sales.Staging_Sales.EmailAddress
     );
```

Topic 34: Identify and resolve data loading performance bottlenecks in dataflows, notebooks, and SQL queries

	Identify performance issues	Possible methods to resolve issues
Dataflow	Refresh History Monitoring Hub Capacity Metrics App	Refactoring Staging Fast copy
SQL Data Warehouse	Query Insights DMVs Capacity Metrics App Statistics & Query Plan (in future)	Refactoring/ Query optimization
Notebook (Spark)	Spark History Server Monitoring Hub Capacity Metrics App	Refactoring/ Query optimization
Delta files	DESCRIBE	V-Order Optimization File partitioning

Topic 35: Implement performance improvements in dataflows, notebooks, and SQL queries

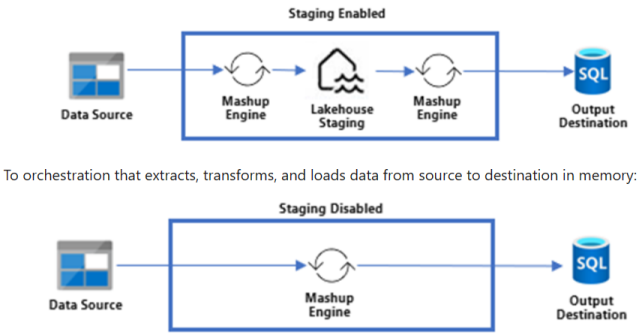
Dataflows

Scenarios where staging not recommended

1. If your data source does not contain big data volumes
2. If you are not joining data from different data sources
3. If you are not performing compute/memory intensive transformations such joining or aggregating large data volumes
4. Query folding is being used for transformations

Scenarios where staging recommended

1. If data destination is a data warehouse
2. Large amount of data is contained in data source



To orchestration that extracts, transforms, and loads data from source to destination in memory:

Notebooks

- The monitoring hub is the central place to monitor Spark jobs and notebook sessions.
- The Spark History Server is not a Fabric feature, but a Spark feature within the Monitoring Hub.
- You can find details about
    - Event Timelines
    - Detailed metrics about each specific spark job

SQL Queries

There are three dynamic management views (DMVs) provided for you to receive live SQL query lifecycle insights.

- sys.dm_exec_connections
    - Returns information about each connection established between the warehouse and the engine.
- sys.dm_exec_sessions
    - Returns information about each session authenticated between the item and engine.
- sys.dm_exec_requests
    - Returns information about each active request in a session.
- If singleton INSERT statements were used for data ingestion consecutively, we recommend creating a new table by using CREATE TABLE AS SELECT (CTAS) or INSERT...SELECT patterns, dropping the original table, and then creating your table again from the table you created using CREATE TABLE AS SELECT (CTAS) or INSERT...SELECT.
- If you roll back a transaction with SELECTs after a large INSERT, update statistics for the columns mentioned in your SELECT.
- The CREATE TABLE AS SELECT (CTAS) statement allows you to create a new table in your warehouse from the output of a SELECT statement. It runs the ingestion operation into the new table in parallel, making it highly efficient for data transformation and creation of new tables in your workspace.
- To achieve optimal query performance, it is important to have accurate statistics.
- If creating statistics manually, consider focusing on those heavily used in your query workload **(specifically in GROUP BYs, ORDER BYs, filters, and JOINs).**
- Consider updating **column-level statistics regularly after data changes that significantly change rowcount or distribution of the data.**

```
CREATE STATISTICS DimCustomer_CustomerKey_FullScan
ON dbo.DimCustomer (CustomerKey) WITH FULLSCAN;

UPDATE STATISTICS dbo.DimCustomer (DimCustomer_CustomerKey_FullScan) WITH FULLSCAN;
```

Queryinsights is a schema exposed in every Data Warehouse, it consists of four main views, all of which are useful for performance monitoring:

- exec_requests_history: Returns information about each completed SQL request/query.
- frequently_run_queries: Returns information about frequently run queries.
- long_running_queries: Returns the information about queries by query execution time.

Topic 36: Identify and resolve issues with Delta table file sizes

The delta file format is great, but it can lead to poor performance and bloated storage sizes, if not managed correctly.

File partitioning for delta tables

- If the column cardinality for a column is very high, do not use that column for partitioning.
- You can partition by a column if the data within that partition is at least 1 GB.
- Most commonly used partition column is date.

Delta table maintenance

As Delta tables change, performance and storage cost efficiency tend to degrade for the following reasons:

- New data added to the table might skew data
- Batch and streaming data ingestion rates might bring in many small files

- Update and delete operations eventually create read overhead; parquet files are immutable by design, so Delta tables adds new parquet files which the changeset, further amplifying the issues imposed by the first two items.
- No longer needed data files and log files available in the storage.

In order to keep the tables at the best state for best performance, perform bin-compaction and vacuuming operations in the Delta tables.

- Bin-compaction is achieved by the OPTIMIZE command; it merges all changes into bigger, consolidated parquet files.
- Dereferenced storage clean-up is achieved by the VACUUM command. It involves removal of files no longer referenced by a Delta table.
- Coalesce() is a Spark method for reducing the amount of partitions in a delta table. Say you have 100 partitions, you can 'coalesce' the delta table into 10 partitions. Importantly, this is quite an efficient operation, because it doesn't require a shuffle of your data.
- Repartition() involves the breaking of existing partitions to create new partitions, these can either be more or less than the original number of partitions. Repartitioning is an expensive operation because it involves shuffling (unlike `coalesce`)

V-Order Optimization

- V-Order is a write time optimization to the parquet file format that enables lightning-fast reads under the Microsoft Fabric compute engines, such as Power BI, SQL, Spark and others.
- V-Order works by applying special sorting, row group distribution, dictionary encoding and compression on parquet files, thus requiring less network, disk, and CPU resources in compute engines to read it, providing cost efficiency and performance.
- V-Order sorting has a 15% impact on average write times but provides up to 50% more compression.
- Fabric provides the V-order capability to write optimized delta lake files. V-order often improves compression by three to four times and up to 10 times performance acceleration over the Delta Lake files that aren't optimized.

Check V-order configuration

```
%%sql
SET spark.sql.parquet.vorder.enabled

%%pyspark
spark.conf.get('spark.sql.parquet.vorder.enabled')
```

Enable / Disable

```
%%pyspark
spark.conf.set('spark.sql.parquet.vorder.enabled', 'false')

%%pyspark
spark.conf.set('spark.sql.parquet.vorder.enabled', 'true')

%%sql
ALTER TABLE person SET TBLPROPERTIES("delta.parquet.vorder.enabled" = "true");

ALTER TABLE person SET TBLPROPERTIES("delta.parquet.vorder.enabled" = "false");
```

Optimized Write

- Optimize Write is a Delta Lake on Synapse feature that reduces the number of files written and aims to increase individual file size of the written data. It dynamically optimizes partitions while generating files with a default 128 MB size. The target file size may be changed per workload requirements using configurations.
- This feature achieves the file size by using an extra data shuffle phase over partitions, causing an extra processing cost while writing the data. The small write penalty should be outweighed by read efficiency on the tables.

When to use it

- Delta lake partitioned tables subject to write patterns that generate suboptimal (less than 128 MB) or non-standardized files sizes (files with different sizes between itself).
- Repartitioned data frames that will be written to disk with suboptimal files size.
- Delta lake partitioned tables targeted by small batch SQL commands like UPDATE, DELETE, MERGE, CREATE TABLE AS SELECT, INSERT INTO, etc.
- Streaming ingestion scenarios with append data patterns to Delta lake partitioned tables where the extra write latency is tolerable.

When to avoid it

- Non partitioned tables.
- Use cases where extra write latency isn't acceptable.
- Large tables with well defined optimization schedules and read patterns.

Enable / Disable the Optimize Write feature

PySpark

```
spark.conf.set("spark.microsoft.delta.optimizeWrite.enabled", "true")
```

SQL

```
SET `spark.microsoft.delta.optimizeWrite.enabled` = true
```

```
%%sql
OPTIMIZE <table|fileOrFolderPath> VORDER;

OPTIMIZE <table|fileOrFolderPath> WHERE <predicate> VORDER;

OPTIMIZE <table|fileOrFolderPath> WHERE <predicate> [ZORDER  BY (col_name1, col_name2, ...)]
VORDER;
```

Topic 37: Implement file partitioning for analytics workloads in a lakehouse

Pipelines

- Enable partitions option in the **Destination** mapping
- This selection allows you to create partitions in a folder structure based on one or multiple columns. Each distinct column value (pair) is a new partition. For example, "year=2000/month=01/file".
- This selection supports insert-only mode and requires an empty directory in the destination.
- Select from the destination columns in schemas mapping. Supported data types are string, integer, boolean, and datetime.
- Format respects type conversion settings under the **Mapping** tab.

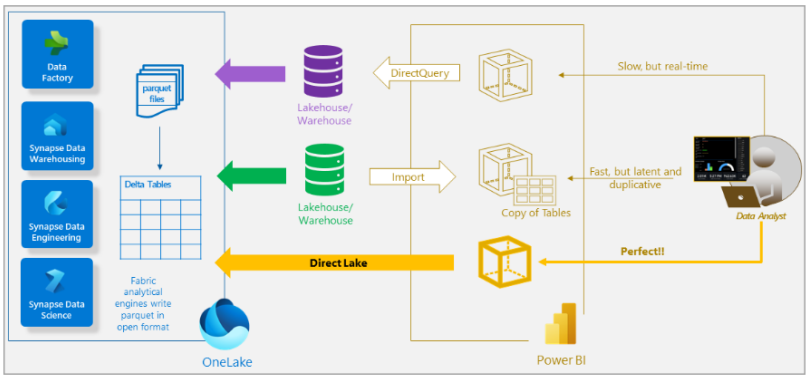
PySpark

```
# write data using partition by
orders_df.write.partitionBy("Year","Month").mode("overwrite").parquet("Files/partitioned_d
ata")

#write data into lakehouse using partitioning
df_output.write.mode("overwrite").partitionBy('year','month','day').parquet(output_path)


# Read data in dataframe from partitioned folder
orders_2021_df =
spark.read.format("parquet").load("Files/partitioned_data/Year=2021/Month=*")
```

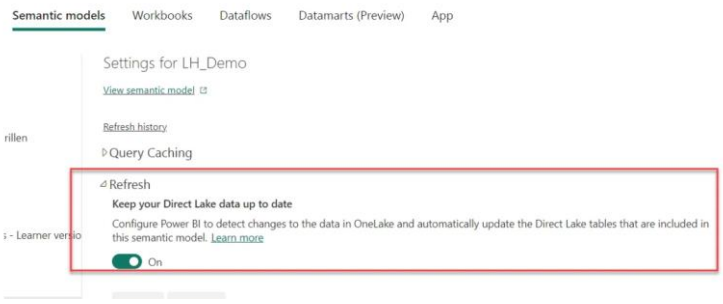
Implement and manage semantic models (20–25%)

Topic 38: Choose a storage mode, including Direct Lake

- Direct Lake is based on loading parquet-formatted files directly from a data lake without having to query a Lakehouse endpoint, and without having to import or duplicate data into a Power BI model.
- Direct Lake is a fast-path to load the data from the lake straight into the Power BI engine, ready for analysis. In other words, this is on-demand loading (returns only the required data for the underlying query)
- The following diagram shows how classic import and DirectQuery modes compare with Direct Lake mode.



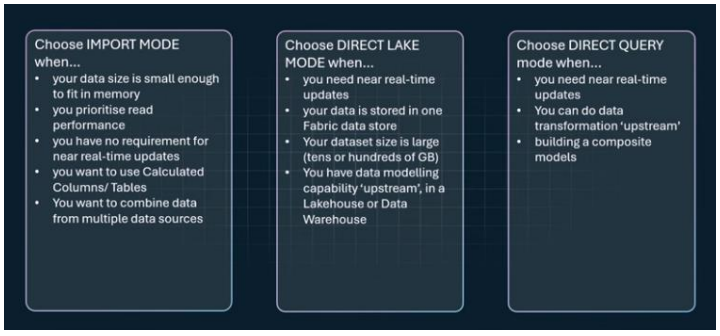
- Because there's no explicit import process, it's possible to pick up any changes at the data source as they occur, combining the advantages of both DirectQuery and import modes while avoiding their disadvantages.
- Whereas in Import mode it is necessary to ensure that the data is in sync with the data source by configuring a scheduled refresh, in Direct Lake mode (unless explicitly disabled), Semantic Model will automatically apply the latest changes from OneLake.
- Power BI semantic models in Direct Lake mode read delta tables directly from OneLake. However, if a DAX query on a Direct Lake model exceeds limits for the SKU, or uses features that don't support Direct Lake mode, like SQL views in a Warehouse, the query can **fall back** to DirectQuery mode.
- Guardrails define resource limits for Direct Lake mode beyond which a fallback to DirectQuery mode is necessary to process DAX queries.
- For Direct Lake semantic models, **Max Memory** represents the upper memory resource limit for how much data can be paged in. In effect, it's not a guardrail because exceeding it does not cause a fallback to DirectQuery; however, it can have a performance impact if the amount of data is large enough to cause paging in and out of the model data from the OneLake data.
- Direct Lake models include the DirectLakeBehavior property, which has three options:
    - Automatic - (Default) Specifies queries fall back to DirectQuery mode if data can't be efficiently loaded into memory.
    - DirectLakeOnly - Specifies all queries use Direct Lake mode only. Fallback to DirectQuery mode is disabled. If data can't be loaded into memory, an error is returned. Use this setting to determine if DAX queries fail to load data into memory, forcing an error to be returned.
    - DirectQueryOnly - Specifies all queries use DirectQuery mode only. Use this setting to test fallback performance.
- The DirectLakeBehavior property can be configured by using Tabular Object Model (TOM) or Tabular Model Scripting Language (TMSL). This property can be configured in the Semantic Model settings:



Fabric/Power BI SKUs	Parquet files per table	Row groups per table	Rows per table (millions)	Max model size on disk/OneLake[1] (GB)	Max memory (GB)
F2	1,000	1,000	300	10	3
F4	1,000	1,000	300	10	3
F8	1,000	1,000	300	10	3
F16	1,000	1,000	300	20	5
F32	1,000	1,000	300	40	10
F64/FT1/P1	5,000	5,000	1,500	Unlimited	25
F128/P2	5,000	5,000	3,000	Unlimited	50
F256/P3	5,000	5,000	6,000	Unlimited	100
F512/P4	10,000	10,000	12,000	Unlimited	200
F1024/P5	10,000	10,000	24,000	Unlimited	400
F2048	10,000	10,000	24,000	Unlimited	400

Limitations

- Currently, Direct Lake models can only contain tables and views from a single Lakehouse or Data Warehouse.
- Direct Lake models created or modified by using XMLA-based tools cannot be opened in the Web modelling feature.
- Direct Lake tables cannot currently be mixed with other table types, such as Import, DirectQuery, or Dual, in the same model. Composite models are not yet supported.
- DateTime relationships are not supported in Direct Lake models.
- Calculated columns and calculated tables are not yet supported.
- Some data types may not be supported.
- Direct Lake tables do not support complex delta table column types. Binary and Guid semantic types are also unsupported. You must convert these data types into strings or other supported data types.
- Table relationships require the data types of their key columns to coincide. Primary key columns must contain unique values. DAX queries will fail if duplicate primary key values are detected.
- Tables based on **T-SQL-based views** cannot be queried in Direct Lake mode. DAX queries that use these model tables fallback to DirectQuery mode.
- Validation is limited for Direct Lake models. User selections are assumed correct and no queries will validate cardinality and cross filter selections for relationships, or for the selected date column in a date table.
- Currently, when creating a new Direct Lake semantic model from the Lakehouse or SQL analytics endpoint in the Power BI service, any model items created in the default model are also added to the new model. These items can include measures, relationships, and properties such a format strings and data category. While these items can be changed in the new model, those changes aren't reflected back to the default model.
- *SQL Server Profiler* traces can be used to check if the query has fallen back to DirectQuery mode or not.
- While Direct Lake mode doesn't query the SQL endpoint when loading data directly from OneLake, it's required when a Direct Lake model must seamlessly fall back to DirectQuery mode, such as when the data source uses specific features like advanced security or views that can't be read through Direct Lake.



Topic 39: Design and build composite models that include aggregations

Composite Models

It allows to create relationships between the data coming from different storage mode sources

Storage Mode options

- Import
- Direct Query
- Dual mode - data is stored in cache memory, but can also be retrieved from the original data source. It's a common scenario to configure dimension tables in Composite models to be in Dual mode, so they can be queried with fact tables from the same source
- Hybrid – Hybrid tables can have both DirectQuery and Import mode within a single table! You configure one table partition to be DirectQuery, while the remaining data stays in Import mode

Regular vs Limited Relationship

A relationship is limited if it has a many-to-many cardinality or if it is a cross-island relationship (import-DQ mode)

4 options (Order of performance)

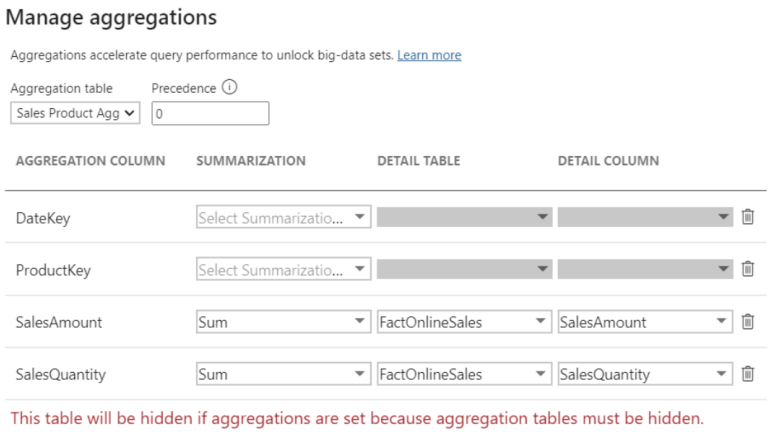
- Import or Dual
- Dual or DirectQuery tables from the same source -> still good as relationships are regular
- Dual or Hybrid tables from the same source -> still regular relationships
- For all other scenarios, sources are different, and relationships are limited.

Composite Model Best practices

- Use composite models ONLY when pure Import mode is not an option
- Using fact tables from the same source group in both DirectQuery and Import (or Hybrid) mode, set the storage mode of dimension tables to **Dual**
- Identify the appropriate refresh rate for the Dual and Import (and Hybrid) tables, to keep the data as much as possible synchronized with the data coming from the DirectQuery sources

Aggregations

- User defined – Work for both Pro and Premium
- Automatic – Work only with Premium
- Power BI needs to be aware of aggregations and that only works for **Direct Query mode**
- Aggregation tables need to be created for the **Fact table** and then Power BI needs to be made aware of this aggregation



- In order for aggregations to work, *data types between the columns from the original fact table and aggregated table must match*
- Aggregation Precedence - This value "instructs" Power BI on which aggregated table to use in case the query can be satisfied from multiple different aggregations! By default, it's set to 0, but you can change the value. The higher the number, the higher the precedence of that aggregation.
- Set dimensions as Dual, Fact table as Direct Query and Aggregated tables in Import mode.
- Aggregated tables are hidden by default.

Topic 40: Design and build a large format dataset

- Large format semantic models provide a highly compressed in-memory cache for optimized query performance, enabling fast user interactivity
- Normally used in your semantic models are > 10GB in size.
- Available on Premium P SKUs, Embedded A SKUs, and with Premium Per User (PPU)
- Used commonly when connecting third-party tools via the XMLA endpoint
- On-demand loading (same feature as Direct Lake)

Topic 41: Implement a star schema for a semantic model

Topic 42: Implement relationships, such as bridge tables and many-to-many relationships

- You can't use the RELATED() function, because more than one row could be related.
- Using the ALL() function on a table doesn't remove filters that are applied to other, related tables by a many-to-many relationship.
- When connecting dimension tables with Facts that are at a lower granularity, M-to-M relationships can be used and with some DAX, the summarization of the Fact table can be controlled

```
Target Quantity =
IF(
    NOT ISFILTERED('Product'[ProductID])
        && NOT ISFILTERED('Product'[Product])
        && NOT ISFILTERED('Product'[Color]),
    SUM(Target[TargetQuantity])
)
```

Topic 43: Write calculations that use DAX variables and functions, such as iterators, table filtering, windowing, and information functions

Windowing Functions

INDEX

- INDEX returns the nth row of a table. INDEX requires as its first argument, the position to find; the second argument is the source table, and the third argument is the sorting to use to determine the position.
- INDEX requires each row in the input table to be unique. If there are not enough columns in the ORDERBY to guarantee each row is unique, then INDEX automatically adds other columns to the ORDERBY clause to guarantee each row is unique.
- The last argument of INDEX defines how to partition the table

```
INDEX (
    1,
    BrandsAndSales,
    ORDERBY (
        [@Sales],
        DESC
    )
)
```

```
INDEX (
        1,
        BrandsAndSalesByCategory,
        ORDERBY ( [@Sales], DESC ),
        KEEP,
        PARTITIONBY ( 'Product'[Category] )
    )
```

OFFSET

- OFFSET returns a row relative to the current row.
- OFFSET receives as the first argument the number of rows to move back or forth, then it receives the source table, and then the usual ORDERBY and PARTITIONBY sections to define the sorting of rows.

```
OFFSET (
                -1,
```

```
                ALL ( 'Date'[Year] ),
                ORDERBY ( 'Date'[Year], ASC )
            )
```

WINDOW

- Window functions allow you to perform calculations within a specific window of your table data.
- This window can either be:
    - time-based (e.g. a 3 month moving average),
    - window of a categorical variables (e.g. average revenue for each department in a company).

```
6 Months Avg =
AVERAGEX (
    WINDOW (
        -5, REL,
         0, REL, // current position
        ORDERBY ( 'Date'[Year Month Number], ASC, 'Date'[Year Month], ASC )
    ),
    [Sales Amount]
)
```

Information Functions

- INFO.TABLES contains information about the tables in the model, such as the table name, description, and whether it is hidden or not.
- INFO.COLUMNS contains information about the columns in a model, such as the column name, data type, and whether it is hidden or not.
- INFO.MEASURES contains information about the measures in the model, such as the measure name, expression, and format string.
- CONTAINS Returns true if values for all referred columns exist, or are contained, in those columns; otherwise, the function returns false.
- CONTAINSSTRING Returns TRUE or FALSE indicating whether one string contains another string.
- HASONEVALUE Returns TRUE when the context for *columnName* has been filtered down to one distinct value only. Otherwise is FALSE.
- ISBLANK Checks whether a value is blank, and returns TRUE or FALSE.
- ISERROR Checks whether a value is an error, and returns TRUE or FALSE.
- SELECTEDMEASURE Used by expressions for calculation items to reference the measure that is in context.
- USERPRINCIPALNAME Returns the user principal name.

Topic 44: Implement calculation groups, dynamic strings, and field parameters

Calculation Groups

- Calculation groups are a simple way to reduce the number of measures in a model by grouping common measure expressions
- Calculation groups are made up of *calculation items*, which are simply DAX statements containing a substitute or placeholder for existing explicit measures in your model.
- *Precedence* is a property defined for a calculation group. When a data model contains more than one calculation group it's essential to define the *precedence*, or the order of evaluation.
- *The* ordinal value *is the sort order of the calculation item. The order in which calculation items appear in a report can be changed by specifying the Ordinal property.*
- SELECTEDMEASURE() – evaluates the measure that is currently in the context (for example, sales amount)

Dynamic Format String

- With dynamic format strings for measures, you can determine how measures appear in visuals by conditionally applying a format string with a separate DAX expression.

Field parameters

- Field parameters allow the report user to select a categorical variable or measure to view in a particular visual.

Topic 45: Implement dynamic row-level security and object-level security

Topic 46: Validate row-level security and object-level security

Row level security

- Think about Row-Level security as limiting access to specific attribute(s) – for example, country (like Canada only), product category (i.e. Economy, Regular, etc.), customer's location, and so on. Dynamic RLS is a method of applying Row-Level Security using the UserPrincipalName() Information Function.
- You can configure row-level security in the semantic model, the data warehouse and the T-SQL endpoint of the Lakehouse in Fabric.
- If you're using Direct Lake mode, you need to configure RLS in the semantic model (otherwise it will fallback to Direct Query).
- RLS only restricts data access for users with Viewer permissions. It doesn't apply to Admins, Members, or Contributors.
- By default, row-level security filtering uses single-directional filters, whether the relationships are set to single direction or bi-directional. You have to explicitly specify this part in PBI Desktop for bi-directional use.
- A user can belong to multiple roles, and the roles are additive. For example, if a user belongs to both the "Sales" and "Marketing" roles, they can see data for both these roles.
- Process has the fol steps
    - Create security role in PBI Desktop
    - Apply DAX filter expressions with *userprincipalname()*
    - Test the role in PBI Desktop.
    - Assign users or security groups to security roles in PBI Service.
    - Test the role in PBI Service
- Row-level security in Fabric Synapse Data Warehouse supports predicate-based security. Filter predicates silently filter the rows available to read operations.
- Access to row-level data in a table is restricted by a security predicate defined as an inline table-valued function.

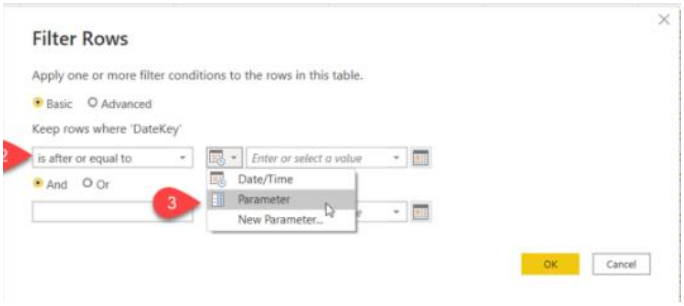
```
-- Creating schema for Security
CREATE SCHEMA Security;
GO

-- Creating a function for the SalesRep evaluation
CREATE FUNCTION Security.tvf_securitypredicate(@SalesRep AS nvarchar(50))
    RETURNS TABLE
WITH SCHEMABINDING
AS
    RETURN SELECT 1 AS tvf_securitypredicate_result
WHERE @SalesRep = USER_NAME() OR USER_NAME() = 'manager@contoso.com';
GO

-- Using the function to create a Security Policy
CREATE SECURITY POLICY SalesFilter
ADD FILTER PREDICATE Security.tvf_securitypredicate(SalesRep)
ON sales.Orders
WITH (STATE = ON);
GO
```

Object Level security

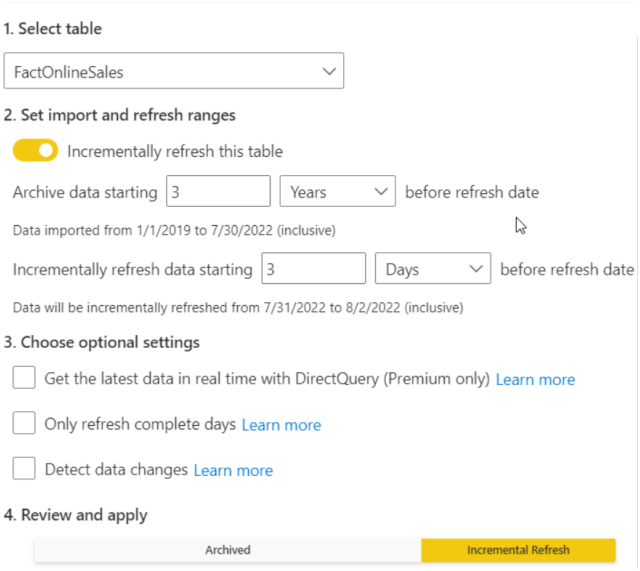
- Think about Object-Level security as limiting access to numbers (measures), such as Unit Cost, or Company Expenses, or complete data objects – for example, whole tables, or whole column(s) of the table!
- With object-level security, you (as a model author) can configure who can view tables and specific columns within a Power BI report.
- Can only be done through Tabular Editor
- Works on roles created for RLS
- With RLS, objects still exist in the model, while with OLS they are completely omitted!
- For the Role, use the Table Properties and set the permission on the column or Table to None.

Topic 47: Implement incremental refresh

- Currently, incremental refresh is only possible within Power BI (not in any of the Fabric ETL items like a Dataflow Gen2 or Data Pipeline).
- Incremental refresh policies are defined in Power BI Desktop
- Pre-requisite
    - Table must contain a Date column
    - Query folding – The date range parameters need to be translated to a single query with a WHERE clause and that is not possible without query folding
    - Single data source – All partitions must query data from a single source
- How to set up
    - Create RangeStart and RangeEnd names for the parameters
    - Filter the Date column of the Fact table using these parameters



    - Select incremental refresh option for the Fact table



    - Under the optional settings, with the *Get the latest data in real time with DirectQuery, one* can take advantage of the Hybrid tables feature
    - *Only refresh complete days* option comes in very handy if only the records for the completed days will be uploaded.
    - *Detect data changes may bring an* even greater performance boost to the data refresh process because it enables the processing of only those records that were changed since the previous run. For this option to work, a specific column in the data source containing the information when the specific record was last time updated (for example, LastUpdated, LoadDate, and so on). This column must be the date/time data type and *should not be the same column that you use for partitioning the fact table*

Topic 48: Implement performance improvements in queries and report visuals

- Use Performance Analyzer or DAX Studio to determine if the report is slow and some additional steps can be applied to improve the report
- Reduce the number of visuals on your report page
- If there is a possibility to satisfy the business request by generating one DAX query instead of 5, you should tend to do it whenever possible
- Try to display data at a low level of granularity. So, display aggregated data in the table by default, and then give the user a possibility to drill through to a specific row if necessary
- Disable cross-filtering option between visuals

- To achieve optimal query performance, it is important to have accurate statistics.
- The traditional option of maintaining statistics health is available in Microsoft Fabric. Users can create, update, and drop histogram-based single-column statistics with CREATE STATISTICS, UPDATE STATISTICS, and DROP STATISTICS, respectively.
- Users can also view the contents of histogram-based single-column statistics with DBCC SHOW_STATISTICS. Currently, a limited version of these statements is supported.
- If creating statistics manually, consider focusing on those heavily used in your query workload (specifically in GROUP BYs, ORDER BYs, filters, and JOINs).
- Consider updating column-level statistics regularly after data changes that significantly change rowcount or distribution of the data.

```
CREATE STATISTICS DimCustomer_CustomerKey_FullScan
ON dbo.DimCustomer (CustomerKey) WITH FULLSCAN;

UPDATE STATISTICS dbo.DimCustomer (DimCustomer_CustomerKey_FullScan) WITH FULLSCAN;
```

Topic 49: Identify use cases for DAX Studio and Tabular Editor 2

DAX Studio

Write DAX, perform different kind of diagnosis, obtain query plans, understand time spent within the VertiPaq Formula engine and Storage engine, troubleshoot performance, format your DAX code

Tabular Editor 2

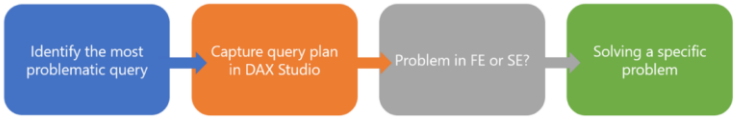
- Data Modeling, Creating and managing Calculation groups and setting the **Object-level security**.
- **Apply Refresh policy** can also be updated for initial load of a large Fact table

Topic 50: Improve DAX performance by using DAX Studio

- All Queries trace: captures query events from client tools (like Power BI)
- Query Plan trace: query plan trace events from a SSAS Tabular server

Vertipaq Performance Analyzer and DAX Studio

- Using Vertipaq performance analyzer with DAX studio can help which part of the query is the slowest. The 3 steps are
    - Run Vertipaq performance analyzer
    - Copy DAX query to DAX Studio
    - Troubleshoot the query in DAX Studio
        - Turn ON query plan and server timings
        - Run the query and observe the FE and SE timings plus number of queries that are run by SE



Server Timings

- Total – shows the total query duration in milliseconds. In other words, you see that my query took 11.3 seconds to execute

- SE CPU – Amount of CPU time spent on Storage Engine queries. As Storage Engine works in a multi-threaded way, it's able to achieve a certain degree of parallelism when running the queries. In our case, the query spent a total of 75 seconds of CPU time
- SE and FE – time split between Storage Engine and Formula Engine. Displayed both as a number value, and as a percentage
- SE Queries – number of Storage Engine queries

Query plan

- Logical plan
- Physical plan

Vertipaq Analyzer in DAX Studio

- DAX Studio -> Advanced -> View Metrics
- **VertiPaq Analyzer collects the data from various DMVs (Dynamic Management Views) in Analysis Services** (don't forget, Power BI stores the data in the instance of the Analysis Services). DMVs are queries that return information about model objects, various server operations, connections, active sessions, and so on.
- Data Model Size Optimization
    - Don't retrieve the columns if you don't need those
    - High cardinality columns have issues with compression so the cardinality of these columns should be reduced
    - Keep only the rows that are required
    - Avoid calculated columns in DAX and try and use Power Query instead
    - Disable Auto Date / Time option as it creates a hidden date table hierarchy
    - Use proper data types
- Formula Engine (FE) accepts the request, process it, generates the query plan and finally executes it
- Storage Engine (SE) pulls the data out of Tabular model to satisfy the request issued within the query generated by the Formula Engine

Topic 51: Optimize a semantic model by using Tabular Editor 2

Best Practice Analyzer

- This is a tool that performs a scan of your semantic model and checks for common issues.
- The list of rules can be downloaded from GitHub, and are organized into the following categories:
    - Performance
    - DAX Expressions
    - Error Prevention
    - Formatting
    - Maintenance
- The checks can also be run from the Tabular Editor CLI as part of a CI/CD process.
- Level 3 and above rule violations are errors.

Explore and analyze data (20–25%)

Topic 52: Implement descriptive and diagnostic analytics

- Descriptive analytics: These analytics interpret past data and KPIs to identify trends and patterns.
- Diagnostic analytics: By focusing on past performance data, these analytics decide which data element will influence specific trends and the possibility of any future events—created from techniques like data mining and correlation.
- Outliers are best visualized through the scatter plots. Use DAX functions to calculate measures that can help identify outliers and anomalies
- Grouping is done for categorical columns and binning is done for numerical columns through specifying either bin size or number of bins. Binning helps to highlight trends and distribution patterns
- Clustering uses machine learning to reveal patterns and insights that may not be immediately apparent. It is best visualized in a scatter plot.

- The 'Analyze' feature in Power BI can automatically generate insights and show correlations (if these exist). Using the 'Explain increase / decrease' feature helps understand the factors contributing to change
- Reference lines can be used to highlight specific values or ranges in visuals to help in comparison and trend identification.
- Error bars provide a visual representation of variability in data, which can be important in a statistical context.
- For creating scorecards, the most relevant metrics that align with the business goals are included. Both hard coded values and measures in the reports can be used to track the progress.

Topic 53: Integrate prescriptive and predictive analytics into a visual or report

- Predictive analytics: By using statistics to forecast future outcomes with statistical models and machine learning techniques, these analytics provide context and clarity for future decisions.
- Prescriptive analytics: These analytics build on descriptive and predictive analytics to recommend specific actions that ensure the best or most profitable customer reactions and business outcomes possible.
- In the Key Influencers AI visual, fields that are considered as influencers for 'Analyze' are placed in 'Explain by'. Fields placed in 'Expand by' are never considered as influencers for 'Analyze'. 'Expand by' fields specify the level of detail for 'Analyze' when a measure or summarized column is placed.
- Forecasting feature available in line charts can predict future trends based on historical date and time data
- AI visuals can be used to uncover hidden insights, predict trends, or simplify complex data analysis tasks. Interpreting the results of the AI visuals is very important as the quality of underlying data can highly impact the accuracy and outcome of the AI visuals

Topic 54: Profile data

- Column Quality focuses on Valid values, Errors and Empty rows for all the columns of a query.
- Column Distribution focuses on distinct values, unique values (which appear only once) for all the columns of a query
- Column Profile returns the minimum, maximum, average, standard deviation, count, null count, distinct count of a column
- Table.schema() and Table.profile() M functions also provide similar information as provided by the Column Quality, Column Distribution and Column Profiling options in Power Query Editor

Topic 55: Query a lakehouse in Fabric by using SQL queries or the visual query editor

Topic 56: Query a warehouse in Fabric by using SQL queries or the visual query editor

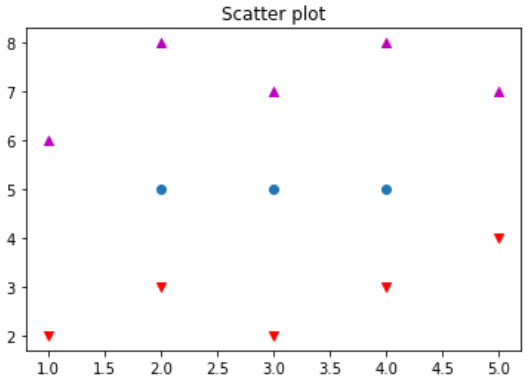
```
-- CTAS to get us started
CREATE TABLE dbo.FactRevenue AS
SELECT * FROM [LH_Bronze].[dbo].[revenue]

CREATE TABLE dbo.DimDate AS
SELECT * FROM [LH_Bronze].[dbo].[date]

CREATE TABLE dbo.DimBranch AS
SELECT * FROM [LH_Bronze].[dbo].[branches]

-- let's just start by visualizing our data (Branch_ID, Date_ID, Revenue)
SELECT top 10 Branch_ID, Date_ID, Revenue FROM dbo.FactRevenue

-- calculate the top 5 branches, by total revenue?
SELECT TOP 5 Branch_ID, SUM(Revenue) AS TotalRevenue FROM dbo.FactRevenue
GROUP BY Branch_ID
ORDER BY SUM(Revenue) DESC

-- change the above query to only return the TOP 3 Branches in Spain
SELECT TOP 3 rev.Branch_ID, B.Branch_NM, B.Country_Name, SUM(Revenue) AS TotalRevenue FROM
dbo.FactRevenue rev
INNER JOIN dbo.DimBranch B on rev.Branch_ID = B.Branch_ID
WHERE B.Country_Name = ' Spain'
GROUP BY rev.Branch_ID, B.Country_Name, B.Branch_NM
ORDER BY SUM(rev.Revenue) DESC

-- what about if we want to filter after the aggregate? I.e. Give me all the Branches that
-- has a Revenue > some amount X
SELECT B.Branch_ID, B.Branch_NM, SUM(Revenue) AS TotalRevenue FROM dbo.FactRevenue rev
INNER JOIN dbo.DimBranch B on rev.Branch_ID = B.Branch_ID
```

```
GROUP BY B.Branch_ID, B.Branch_NM
HAVING SUM(Revenue) > 50000000

-- Common Table Expressions
WITH top_5_rev as (
      SELECT TOP 5 Branch_ID, SUM(Revenue) AS TotalRevenue FROM dbo.FactRevenue
      GROUP BY Branch_ID
      ORDER BY SUM(Revenue) DESC
),
branches as (
      SELECT Branch_ID, Branch_NM from dbo.DimBranch
)
SELECT t5.Branch_ID, b.Branch_NM, t5.TotalRevenue  FROM top_5_rev t5
LEFT JOIN branches b on t5.Branch_ID = b.Branch_ID

-- LAG and LEAD
-- LAG -> null values at the top
-- LEAD -> null values at the end
-- offset parameter cannot be a negative value

with revT as (
      SELECT TOP 5 Branch_ID
                , floor(Revenue/1000000) as Rev_M
      FROM dbo.FactRevenue
)

SELECT Branch_ID
      , Rev_M
      , LEAD(Rev_M, 2) OVER(ORDER BY Rev_M) as lead_col
from revT

-- ROW Number
with parts as (

      SELECT Dealer_ID
                , Revenue
                , ROW_NUMBER()  OVER(PARTITION BY Dealer_ID ORDER BY Revenue) as row_num
      from dbo.FactRevenue
)

select * from parts
WHERE Dealer_ID in ('DLR0001', 'DLR0017')

-- Subqueries
SELECT * from (
      SELECT * from dbo.FactRevenue
      WHERE Dealer_ID in ('DLR0001', 'DLR0017')
) sub

-- Cross database queries
SELECT *
FROM ContosoLakehouse.dbo.ContosoSalesTable AS Contoso
INNER JOIN My_lakehouse.dbo.Affiliation as MyAffiliation
ON MyAffiliation.AffiliationId = Contoso.RecordTypeID;
```

PySpark

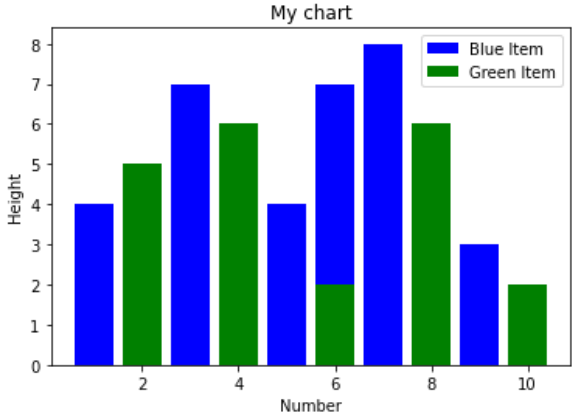
- Scatter plot

```
import matplotlib.pyplot as plt
x1 = [2, 3, 4]
y1 = [5, 5, 5]
x2 = [1, 2, 3, 4, 5]
y2 = [2, 3, 2, 3, 4]
y3 = [6, 8, 7, 8, 7]
plt.scatter(x1, y1)
plt.scatter(x2, y2, marker='v', color = 'r')
plt.scatter(x2, y3, marker='^', color = 'm')
plt.title("Scatter plot")
plt.show()
```



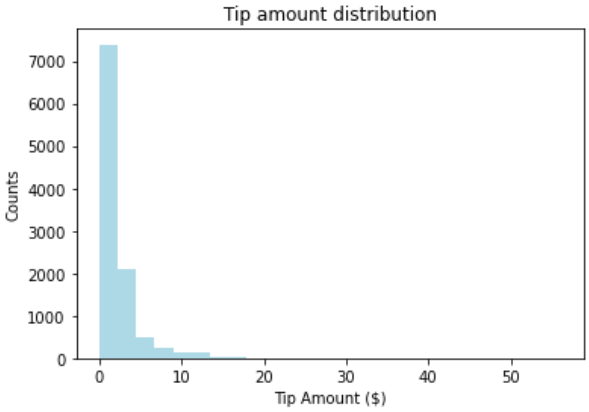
- Bar chart (stacked)

```
x1 = [1, 3, 4, 5, 6, 7, 9]
y1 = [4, 7, 2, 4, 7, 8, 3]
x2 = [2, 4, 6, 8, 10]
y2 = [5, 6, 2, 6, 2]
plt.bar(x1, y1, label = "Blue Item", color = 'b')
plt.bar(x2, y2, label = "Green Item", color = 'g')
plt.plot()
plt.xlabel("Number")
plt.ylabel("Height")
plt.title("My chart")
plt.legend()
plt.show()
```



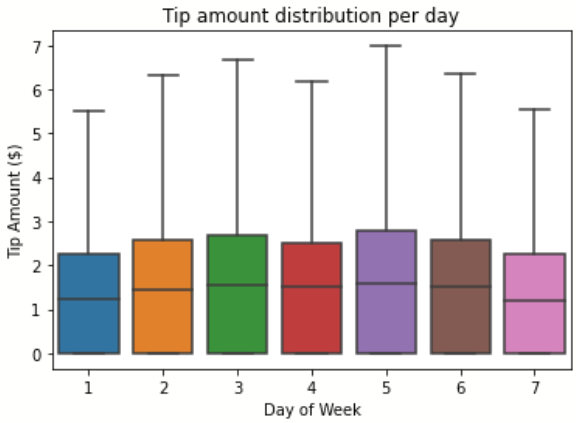
- Histogram

```
ax1 = sampled_taxi_pd_df['tipAmount'].plot(kind='hist', bins=25, facecolor='lightblue')
ax1.set_title('Tip amount distribution')
ax1.set_xlabel('Tip Amount ($)')
ax1.set_ylabel('Counts')
plt.suptitle('')
plt.show()
```

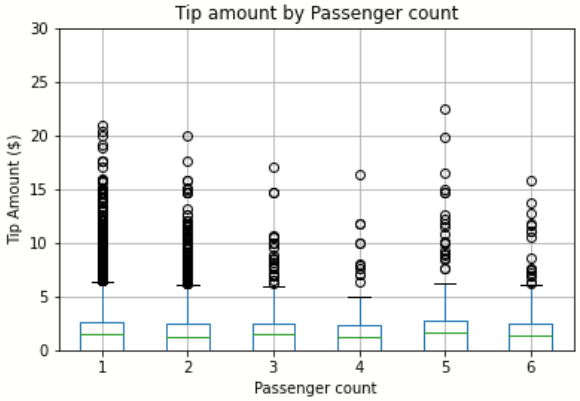


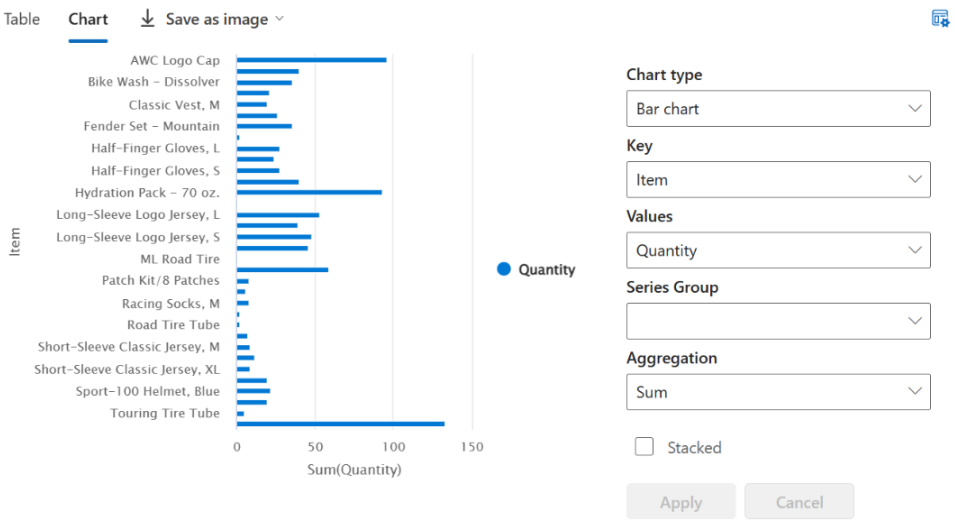
- Box plot

```
ax = sns.boxplot(x="day_of_week", y="tipAmount",data=sampled_taxi_pd_df, showfliers = False)
ax.set_title('Tip amount distribution per day')
ax.set_xlabel('Day of Week')
ax.set_ylabel('Tip Amount ($)')
plt.show()
```




```
# How many passengers tipped by various amounts
ax2 = sampled_taxi_pd_df.boxplot(column=['tipAmount'], by=['passengerCount'])
ax2.set_title('Tip amount by Passenger count')
ax2.set_xlabel('Passenger count')
ax2.set_ylabel('Tip Amount ($)')
ax2.set_ylim(0,30)
plt.suptitle('')
plt.show()
```





Run the code to render an existing Power BI report in a notebook

```
from powerbiclient import Report

report_id="Your report id"
report = Report(group_id=None, report_id=report_id)

report
```

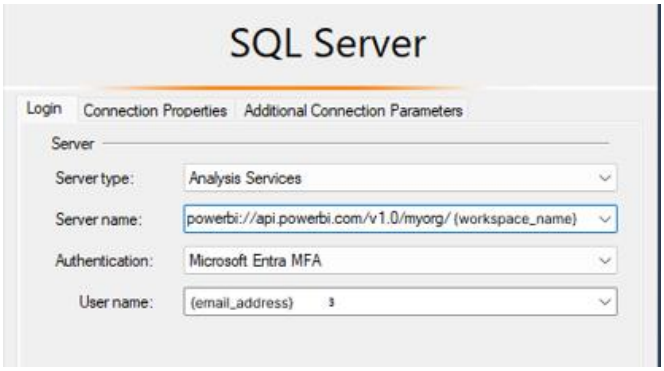
You can use a Spark DataFrame in your notebook to quickly generate insightful Power BI visualizations

```
# Create a spark dataframe from a Lakehouse parquet table
sdf = spark.sql("SELECT * FROM testlakehouse.table LIMIT 1000")

# Create a Power BI report object from spark data frame
from powerbiclient import QuickVisualize, get_dataset_config
PBI_visualize = QuickVisualize(get_dataset_config(sdf))

# Render new report
PBI_visualize
```

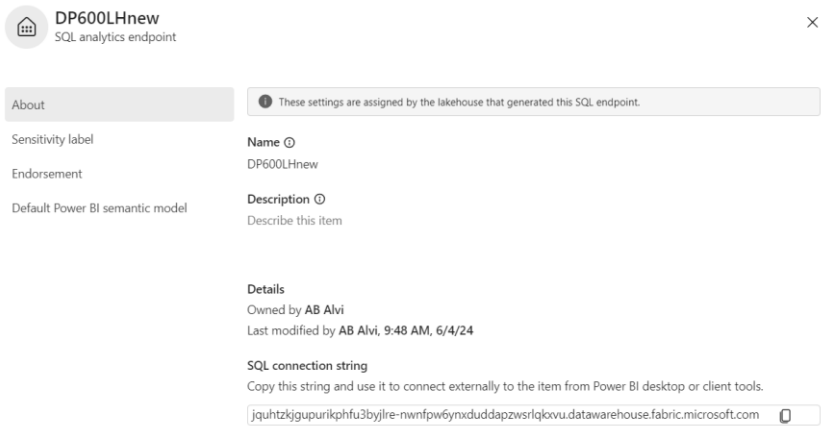
Topic 57: Connect to and query datasets by using the XMLA endpoint

To connect to the XMLA Endpoint using SSMS:

1. Grab the XMLA endpoint address from your Workspace Settings in Fabric -> License Info
2. In SSMS create a new connection like so:



3. To connect with SQL endpoint, go to the settings in Lakehouse or Warehouse



Important T-SQL functions for the exam

- SELECT
- FROM
- WHERE
- GROUP BY
- HAVING
- ORDER BY
- CASE
- PARTITION BY
- ROW_NUMBER
- RANK
- DENSE_RANK
- LAST_VALUE
- NTILE
- LEAD
- LAG
- LEAST
- GREATEST
- DATETRUNC
- DATEPART
- COALESCE

Important PySpark methods for the exam

- .display()
- .show()
- .summary()
- .describe()
- .dropna()
- .fillna()
- .dropduplicates()
- .dtypes()
- .loc()
- .iloc()
- .query()
- .isnull()
- .columns()
- .get()
- .select()
- .selectExpr()
- .withColumn()
- .withcolumnRenamed()
- .inferschema()
- F.broadcast()
- .transform()