Fabric

Fabric Data Engineer Associate – DP700

Fabric Analytics Engineer

Implementing Analytics Solution Using Microsoft Fabric – DP 600

Study guide for Exam DP-600: Implementing Analytics Solutions Using Microsoft Fabric | Microsoft Learn

Online course

Microsoft Certified: Fabric Analytics Engineer Associate - Certifications | Microsoft Learn

Voucher

Get certified in Microsoft Fabric—for free! - Microsoft Fabric Community

✔To be eligible for this limited-time offer, you must:

1. Join the Fabric Community if you haven't already.
2. Not already be a Microsoft Certified: Fabric Analytics Engineer Associate (DP-600).
3. Register for and complete all modules in the Microsoft Learn Challenge | Ignite Edition: Fabric Challenge.
    - Do not submit your request form before completing the challenge your or request will be denied.
4. Be confident that you can *take and pass* exam DP-600 by December 31, 2024.
5. Agree to these terms and conditions.

Challenge

Collections | Microsoft Learn

Course Syllabus

Get Started with Microsoft Fabric (11 modules)

Implement a data warehouse with Microsoft Fabric (5 modules)

Work with Semantic models in Microsoft Fabric (6 modules)

Administer and govern Microsoft Fabric (3 modules)

Get Started with Microsoft Fabric (11 modules)

OneCopy: Is a component of OneLake, allows to read data from a single copy, without moving or duplicating data.

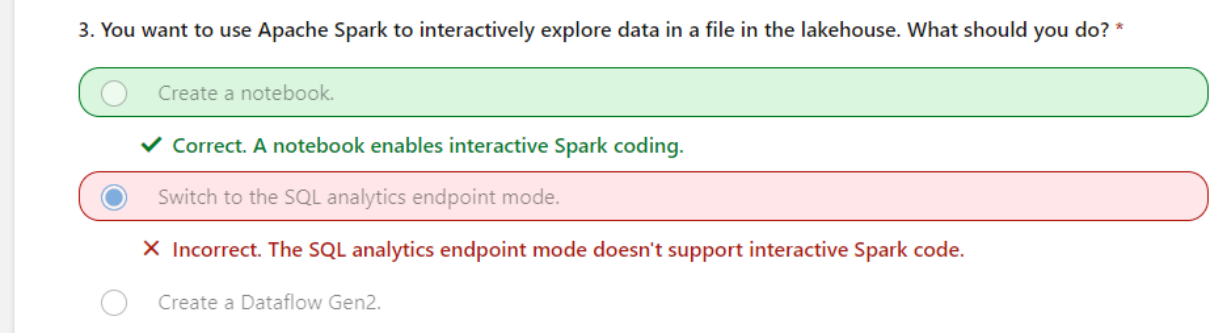
OneLake: All of the compute engines in Fabric automatically store their data in OneLake. For tabular data, the analytical engines in Fabric write data in delta-parquet format.

Shortcuts: Embedded references withing OneLake that point to other files or storage locations.

Workspaces: Logical containers to organize and manage data, reports and other assets. Provide separation of resources, helping with access control and security. Some features of workspaces are data lineage and impact analysis.

Lakehouses provide data engineers and analysts with the combined benefit of data lake storage and relational data warehouse.

Hints



Unit 3 – Use Apache Spark

Fabric provides support for Spark clusters.

Objectives of module:

Configure Spark in a Fabric workspace

Use spark to connect to data sources

Use spark dataframes to analyze and transform data.

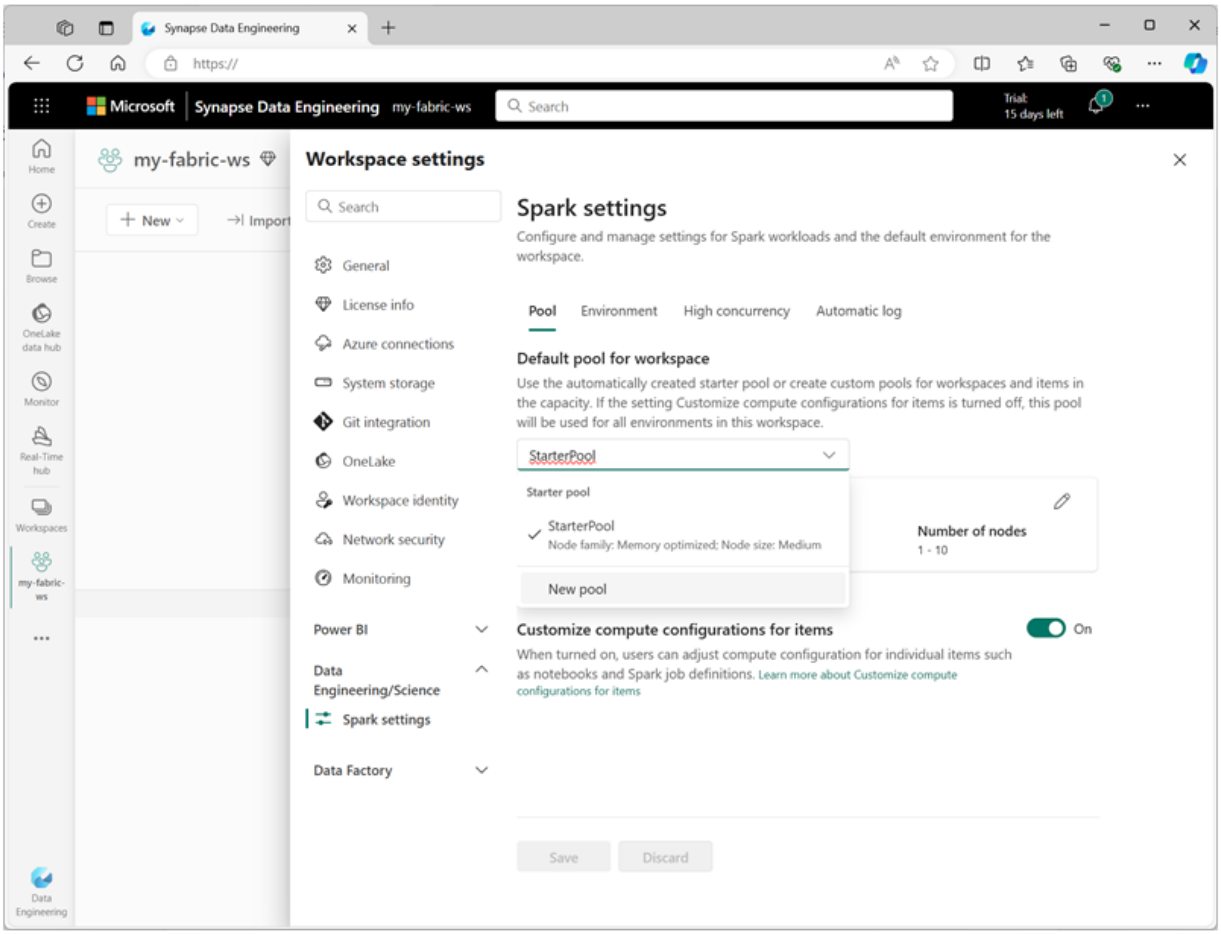
Use spar SQL to query data

Visualize data in a Spark notebook

Spark pools consist of compute nodes that distribute data processing tasks. A spark pool contains two kinds of node: 1) A head node, coordinates distributed processes through a driver program. 2) worker nodes on which executer processes perform the actual data processing tasks.

Fabric provides a starter pool in each workspace, enabling Spark jobs to be started and run quickly with minimal setup and configuration. Additionally, you can create custom Spark pools with specific node configurations that support your data processing needs. Note: The ability to customize Spark pool settings can be disabled by Fabric administrators at the Fabric Capacity level.

You can manage settings for the starter pool and create new Spark pools in the Data Engineering/Science section of the workspace settings.



Specific configuration settings for Spark pools: Node Family (type of virtual machines, memory optimized provide optimal performance), Autoscale (to automatically provision nodes as needed, if so, the maximum number of nodes), Dynamic allocation (to allocate executor processes on the worker nodes based on data volumes.
Yes, you've got it right!

- **Dynamic allocation** adjusts the number of executors on the worker nodes based on the workload.

- **Autoscale** adjusts the number of worker nodes in the Spark pool to manage overall capacity.

Together, they help optimize resource usage and performance in Spark jobs.

Environments

When creating a environment you can: 1) specify the Spark runtime, 2) View the built-in libraries that are installed in every environment. 3) Install specific libraries from the Python Package Index (PyPI) 4) Install custom libraries by uploading a package file 5) Specify the Spark pool that the environment should use 6) Specify Spark configuration properties to override default behavior 6) Upload resource files that need to be available in the environment

Additional options

Native execution engine: A vectorized processing engine that runs Spark operations directly on lakehouse infrastructure. Using it, can improve the performance of queries with large data sets in Parquet or Delta file formats.

The native execution engine can be enabled at the environment level or within an individual notebook. To enable it at the environment level, set the following Spark properties in the environment configuration

- **spark.native.enabled**: true
- **spark.shuffle.manager**: org.apache.spark.shuffle.sort.ColumnarShuffleManager

To enable it for a specific notebook, you can set these configuration properties at the beginning of your code:

```
%%configure

{

  "conf": {

     "spark.native.enabled": "true",

     "spark.shuffle.manager": "org.apache.spark.shuffle.sort.ColumnarShuffleManager"

  }

}
```

High concurrency mode

To share Spark sessions across multiple concurrent users or processes, ensuring isolation of code to avoid variables in one notebook being affected by code in another notebook. It is enabled in the **Data Engineering/Science** section of the workspace settings interface.

Scheduling notebooks and job execution

When you run code in a notebook or a Spark job definition interactively, the code runs in the security context of the currently logged in user. When run using a schedule, the code assumes the identity of the user who created the schedule. When run in a pipeline, the code uses the identity of the user who owns the pipeline. Regardless of how the code is run, you must ensure that the identity used in the security context has sufficient permissions to access any resources references by the code, including tables or files in a lakehouse.

Unit 4 – Work with Delta lake Tables in Fabric

To create a table, save the dataframe with delta format

```
# Load a file into a dataframe

df = spark.read.load('Files/mydata.csv', format='csv', header=True)


# Save the dataframe as a delta table

df.write.format("delta").saveAsTable("mytable")
```

Managed vs external tables

Managed: The table definition and the data files are both managed by the Spark runtime for the Fabric Lakehouse.

External table: the table definition in the metastore is mapped to an alternative file storage location.

To create an external table, we must specify the path

```
df.write.format("delta").saveAsTable("myexternaltable", path="Files/myexternaltable")
```

We can also specify a fully qualifitied path for a storage location

```
df.write.format("delta").saveAsTable("myexternaltable",
path="abfss://my_store_url..../myexternaltable")
```

Note: Deleting an external table from the lakehouse metastore doesn't delete the associated data files.

Question: How to delete items from the metastore?

Creating table metadata

There are multiple ways to crate a table definition in the metastore

1) DeltaTableBuilder API

```python
from delta.tables import *

DeltaTable.create(spark) \
  .tableName("products") \
  .addColumn("Productid", "INT") \
  .addColumn("ProductName", "STRING") \
  .addColumn("Category", "STRING") \
  .addColumn("Price", "FLOAT") \
  .execute()
```

2) Spark SQL

```sql
%%sql

CREATE TABLE salesorders
(
    Orderid INT NOT NULL,
    OrderDate TIMESTAMP NOT NULL,
    CustomerName STRING,
    SalesTotal FLOAT NOT NULL
)
USING DELTA
```

3) Saving data in delta format

   To save the data in delta format without creating a table definition in the metastore.

   Note: Fabric uses an automatic table discovery capability to create the corresponding table metadatada in the metastore.

```python
delta_path = "Files/mydatatable"
df.write.format("delta").save(delta_path)
```

Note: To create an external table, the location must be specified in the code, for example:

```
%%sql

CREATE TABLE MyExternalTable
USING DELTA
LOCATION 'Files/mydata'
```

OptimizeWrite

Parquet files are immutable, so new files are written for every update or delete. This process results in storing data in many small files (small file problem). To prevent it, OptimizeWrite reduces the number of files as they are written.

In Fabric, OptimizeWrite is enabled by default. It can be enabled/disabled at the Spark session level

Disable Optimize Write at the Spark session level

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

Enable Optimize Write at the Spark session level

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

print(spark.conf.get("spark.microsoft.delta.optimizeWrite.enabled"))

V-Order function

Unit 5 – Orchestrate processes and data movements with Microsoft Fabric

Fabric includes pipelines, they are like ADF pipelines. Below are some pipeline concepts

Activities: are executable tasks, they are two categories – data transformation activities, control flow activities.

Parameters: To allow reusability of the pipelines.

Pipeline run: When a pipeline is executed, a data pipeline run is initiated. It can be on-demand or scheduled.

Copy data activity: When you need to copy data directly between a supported source and destination without transformation, or when transformations are going to occur later.

If you need to apply transformations, then use Data Flow activity to run a dataflow (Gen2).

Unit 6 – Ingest Data with Dataflows Gen2 in Microsoft Fabric

Dataflows Gen 2 use Power Query Online

Dataflows can be horizontally partitioned. Once you create a global dataflow, data analysts can use dataflows to create specialized semantic models for specific needs.

Limitations of dataflows

1. Dataflows are not a replacement for a data warehouse
2. Row-level security isn't supported.
3. Fabric capacity workspace is required.

Unit 7 – Get started with data warehouse in Microsoft Fabric

Fabric's data warehouse supports fully transactional T-SQL and can be used to store and query data in the Lakehouse.

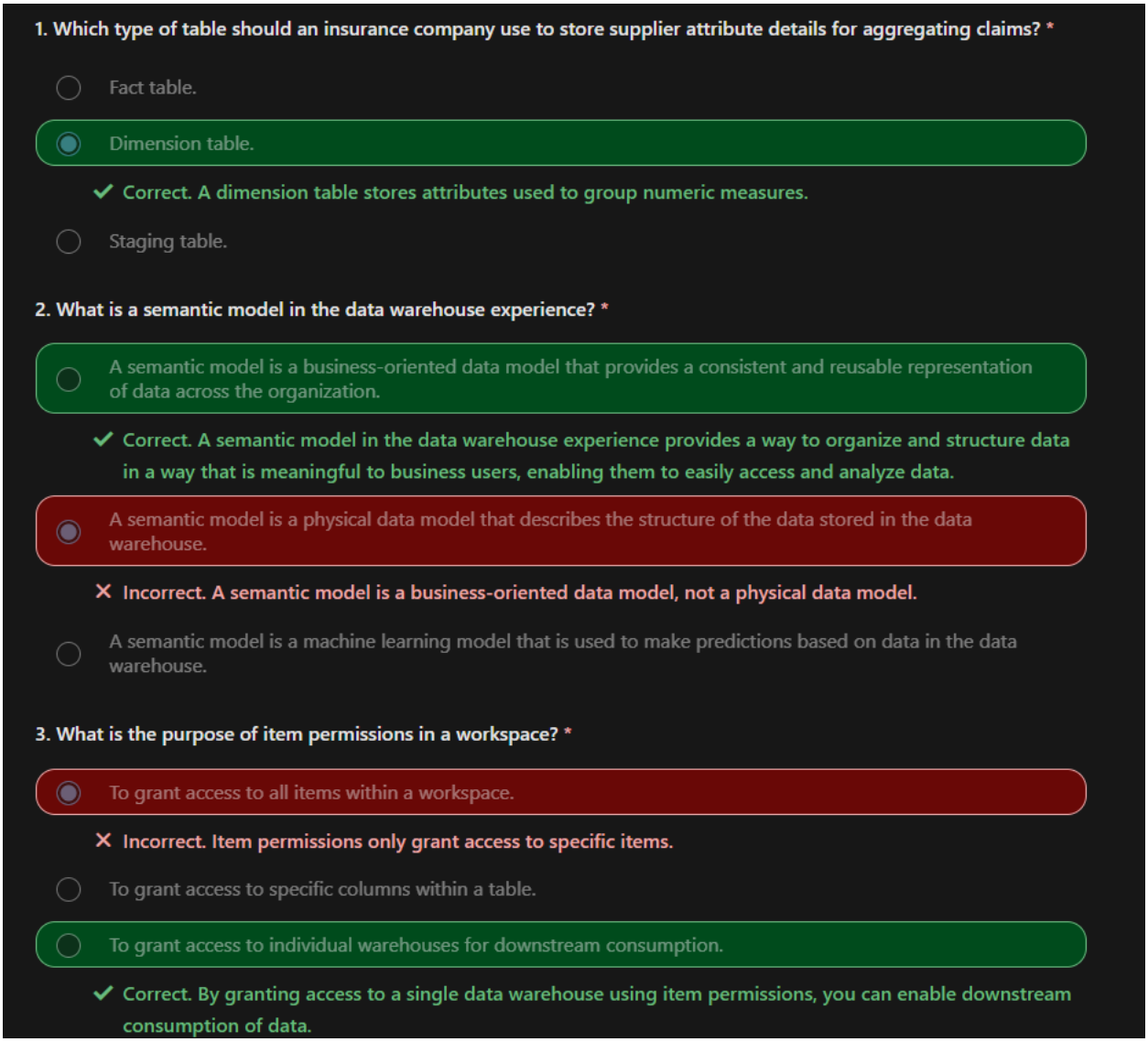
Clone tables are replicas of table created by copying the metadata while still referencing the same data files in OneLake.

There are two ways to query data: The Visual query editor, and the SQL query editor.

There is also a SQL analytics endpoint, where you can connect from any tool.

The default semantic model

There is a default semantic model automatically created in Fabric. It inherits business logic from the parent lakehouse or warehouse, which initiates



Challenge Module – Load data into a Microsoft Fabric data warehouse

Copy statement: offers flexibility as the format is specified, designate a location for rejected rows, skip header rows and other configurable options.

Challenge Module – Secure a data warehouse in Fabric

Three measures: Workspace roles (admin, member, contributor and viewer), item permissions, data protection security.

Dynamic data masking

Limits data exposure to nonprivileged users by obscuring sensitive information. It performs real-time masking, the actual data is never exposed to unauthorized users. It doesn't require complex coding. The data is never changed, a masking rule is applied to the query results.

Dynamic data masking (DDM) is set up at the column level. A function defines the masking type (default, email, partial and random)

```
-- For Email

ALTER TABLE Customers

ALTER COLUMN Email ADD MASKED WITH (FUNCTION = 'email()');


-- For PhoneNumber

ALTER TABLE Customers

ALTER COLUMN PhoneNumber ADD MASKED WITH (FUNCTION = 'partial(3,"XXX-XXX-",4)');


-- For CreditCardNumber

ALTER TABLE Customers

ALTER COLUMN CreditCardNumber ADD MASKED WITH (FUNCTION = 'partial(4,"XXXX-XXXX-XXXX-",4)');
```

Row-level security

RLS works by associating a function (a security predicate) to a table. The function is defined to return true or false based on certain conditions. When a user attempts to access data in the table, the security predicate function is invoked.

RLS is implemented in two main steps:

1. Filter predicates : It is an inline table-valued function that filters the results based on the predicate defined.
2. Security policy: A security policy that invokes an inline table-valued function to protect access to the rows in a table.

SQL Copy

```sql
-- Create supporting objects for this example
CREATE TABLE [Sales] (SalesID INT,
    ProductID INT,
    TenantName NVARCHAR(10),
    OrderQty INT,
    UnitPrice MONEY)
GO

INSERT INTO [Sales]  VALUES (1, 3, 'tenant1@contoso.com', 5, 10.00);
INSERT INTO [Sales]  VALUES (2, 4, 'tenant2@contoso.com', 2, 57.00);
INSERT INTO [Sales]  VALUES (3, 7, 'tenant3@contoso.com', 4, 23.00);
INSERT INTO [Sales]  VALUES (4, 2, 'tenant4@contoso.com', 2, 91.00);
INSERT INTO [Sales]  VALUES (5, 9, 'tenant5@contoso.com', 5, 80.00);

-- View all the rows in the table
SELECT * FROM Sales;
```

Next, we create a new schema, an inline table-valued function, and grant user access to the new function. The `WHERE @TenantName = USER_NAME() OR USER_NAME() = 'TenantAdmin'` predicate evaluates if the user name executing the query matches the *TenantName* column values.

SQL Copy

```sql
--Create a schema
CREATE SCHEMA [Sec];
GO

--Create the filter predicate
CREATE FUNCTION sec.tvf_SecurityPredicatebyTenant(@TenantName AS NVARCHAR(10))
    RETURNS TABLE
WITH SCHEMABINDING
AS
    RETURN  SELECT 1 AS result
            WHERE @TenantName = USER_NAME() OR USER_NAME() = 'tenantAdmin@contoso.com';
GO

--Create security policy and add the filter predicate
CREATE SECURITY POLICY sec.SalesPolicy
ADD FILTER PREDICATE sec.tvf_SecurityPredicatebyTenant(TenantName) ON [dbo].[Sales]
WITH (STATE = ON);
GO
```

If you alter the security policy with WITH (STATE = OFF);, you notice that users see all the rows.

Best practices to consider when implementing RLS

- Create a separate schema for predicate functions, and security policies.
- Whenever possible, avoid type conversions in predicate functions.
- To maximize performance, avoid using excessive table joins and recursion in predicate functions.

Column-level security

CLS restricts column access. CLS is assigned to roles.

Configure column-level security

In the scenario we've recently explored, the syntax to use for implementing column-level security might look as follows:

SQL Copy

```
-- Create roles
CREATE ROLE Doctor AUTHORIZATION dbo;
CREATE ROLE Nurse AUTHORIZATION dbo;
CREATE ROLE Receptionist AUTHORIZATION dbo;
CREATE ROLE Patient AUTHORIZATION dbo;
GO

-- Grant SELECT on all columns to all roles
GRANT SELECT ON dbo.Patients TO Doctor;
GRANT SELECT ON dbo.Patients TO Nurse;
GRANT SELECT ON dbo.Patients TO Receptionist;
GRANT SELECT ON dbo.Patients TO Patient;
GO

-- Deny SELECT on the MedicalHistory column to the Receptionist and Patient roles
DENY SELECT ON dbo.Patients (MedicalHistory) TO Receptionist;
DENY SELECT ON dbo.Patients (MedicalHistory) TO Patient;
GO
```

In this example, we first create the roles `Doctor`, `Nurse`, `Receptionist`, and `Patient`. We then grant `SELECT` permissions on all columns in the `Patients` table to all roles. Finally, we deny `SELECT` permissions on the `MedicalHistory` column to the `Receptionist` and `Patient` roles. This ensures that only users with the `Doctor` or `Nurse` role can access the `MedicalHistory` column.

We can also apply views to restrict a column. There are some trade-offs between CLS and views.

Granular permissions

GRANT/DENY DML (SELECT, INSERT, UPDATE, DELETE)

Principle of least privilege: users and applications should only be given permissions needed for them to complete the task.

Dynamic SQL

Dynamic SQL is a concept where a query is built programmatically. Dynamic SQL allows T-SQL statements to be generated within a stored procedure or a query itself. A simple example is shown below.

SQL Copy

```sql
CREATE PROCEDURE sp_TopTenRows @tableName NVARCHAR(128)
AS
BEGIN
    DECLARE @query NVARCHAR(MAX);
    SET @query = N'SELECT TOP 10 * FROM ' + QUOTENAME(@tableName);
    EXEC sp_executesql @query;
END;
```

This example demonstrates a stored procedure that accepts a table name as a parameter and returns the top 10 rows from that table. This could be useful for quickly inspecting tables in a data warehouse.

SQL Copy

```sql
CREATE PROCEDURE sp_TopTenRows @tableName NVARCHAR(128)
AS
BEGIN
    DECLARE @query NVARCHAR(MAX);
    SET @query = N'SELECT TOP 10 * FROM ' + QUOTENAME(@tableName);
    EXEC sp_executesql @query;
END;
```

In this example, `@tableName` is the parameter that you can replace with the name of the table you want to inspect. The `QUOTENAME` function is used to safely quote the table name, preventing SQL injection attacks. The `sp_executesql` stored procedure is then used to execute the dynamically built query.

Please note that this is a simple example and real-world data warehouse tasks might require more complex dynamic SQL queries. Always be cautious when using dynamic SQL due to the risk of SQL injection attacks. Always use parameterization methods like `sp_executesql` or `QUOTENAME` to sanitize inputs.

Unit 8 – Real-Time intelligence in Microsoft Fabric

Stream processing solutions have these:

1. Unbound: Data is added to the stream perpetually
2. Add temporal data to the data in the stream: To indicate when the event occurred or was recorded.
3. Aggregations are usually performed over temporal windows. Data per second, per minute.
4. Use streaming data for visualization, automation or persist for historical analysis.

Activator

Operated based on four concepts: Events, Objects, Triggers and Properties.

Events:

Challenge Module – Create PBI model relationships.

Understand how model relationships work

Set up relationships.

Use DAX relationship functions.

Understand relationship evaluation.

Model relationships don't enforce data integrity.

When multiple filters are applied to a table, it's always an AND operation, requiring that all conditions must be true.

A relationship relates one column in one table to one column in another table. Special case:

DirectQuery models can have a multi-column relationship.

Cardinality note: If a data refresh operation attempts to load duplicate values into a "one" side column, the entire data refresh will fail.

Cross filter directions

Cardinality Type	Cross filter options
One-to-many	Single, both
One-to-one	Both
Many-to-many	Single (Table1 to Table2) Single (Table2 to Table1)

	Both

The cross filter direction can be modified by using a model calculation CROSSFILTER.

Use Bi-directional filter as a last resort.

Active vs Inactive relationships

There can only be one active filter propagation path between two model tables. Inactive relationships can only be made active during the evaluation of a mode calculation using USERELATIONSHIP DAX function.

Set assume referential integrity

Assume referential integrity property is available for one-to-many and one-to-one relationships between DirectQuery storage mode tables that belong to the same source group. Note: This property can be enabled only of the “may” side column doesn’t contain NULLs

When enabled, native queries sent to the data source will join by using INNER JOIN rather than OUTER join. Enabling this property generally improves performance. Note: Always enable this property when a single-column database foreign key constrain **exists** between the two tables.

RELATED

It retrieves the value from “one” side of a relationship.

RELATEDTABLE

Retrieves a table of rows from the “may” side of a relationship.

USERELATIONSHIP

Forces the use of a specific inactive model relationship. Useful for role-playing dimension table.

CROSSFILTER

Either modifies the relationship cross filter direction (to one or both), or it disables filter propagation. It’s useful when you need to change or ignore model relationships during the evaluation of a specific calculation.

COMBINEVALUES

Joins two or more text strings into one text string. To support multi-column relationships in DirectQuery models when table belong to the same source group.

TREATAS

Applies the result of a table expression as filter to columns from an unrelated table. It’s helpful in advances scenarios when you want to create a virtual relationship during the evaluation of a specific calculation.

Relationship evaluation

Classified as either regular or limited. It’s not a configurable property and it is inferred from the cardinality type and the data source of the two related tables.

Challenge Module – Administer a Microsoft Fabric environment.

Implement a data warehouse with Microsoft Fabric

5 units

Unit 1 -

Work with semantic models in Fabric

6 units

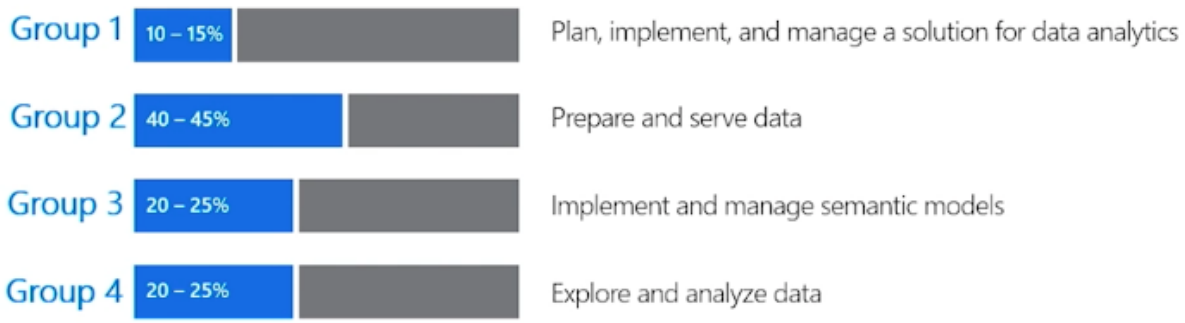
Administer and govern Microsoft Fabric

4 units

Tips from Video – Preparing for DP-600. Plan, implement and manage a solution for data analytics

[Preparing for DP-600: Plan, implement, and manage a solution for data analytics (Part 1 of 4) | Microsoft Learn]

Weight of skills covered by the exam



Group 1

What permissions are needed for user accessing a spark cluster vs a sql endpoint vs semantic model?

Fabric admin portal is the place to managing data and access based on security rules. Managing access to workspace, assigning the right access level depending on users needs.

How and when to scale capacity?

Version control for a workspace. Which source controls work with Fabric?

PBI projects. Advantages or using a Power BI Project PBIP file; PBI template (pbit), PBI data source (pbids).

Plan and implement deployment solutions. How to create a deployment pipeline and assign permissions. Perform changes to deployment pipelines via API.

Impact analysis. What are the steps to do impact analysis on an item.

Deploy semantic models using XMLA endpoint.

Create a custom PBI report theme, manage sensitivity lables.