SQL Saturday Atlanta 2025 - AI & BI (#1102)

08 March 2025

Fabric Pipelines

Metadata-driven ETL Patterns!

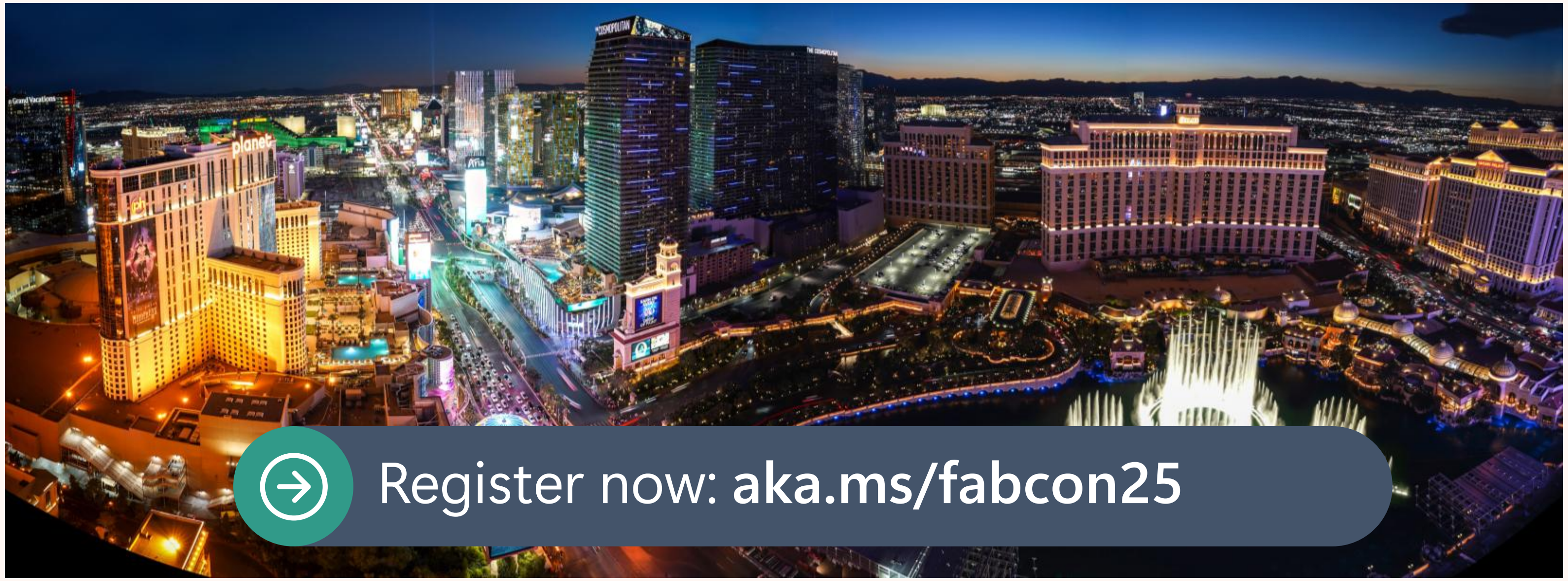
Mike Diehl Improving Winnipeg

Join us at the Fabric Community Conference

MGM GRAND, Las Vegas, NV

March 31 - April 2, 2025

Workshops: March 29, 30, and April 3



Join us at the second annual Microsoft Fabric Community Conference and get up close with the latest data, analytics, and AI developments—plus network with community leaders and other technical experts from around the world.

~~**Use code FABINSIDER for a $400 discount***~~

Use code DIEHL for a $200 discount.

*Discount must be redeemed by March 18

Thanks to our sponsors

Please visit our sponsors!

CloudStaff
MORE. HUMAN.

improving
It's what we do. ™

KEY 2 Consulting

POWER BI SENTINEL
Governance, Disaster Recovery and Auditing for Power BI & Fabric

COZYROC

VISCOSITY
PROFESSIONAL SERVICES

ROYAL BLUE
ANALYTICS

Closing Ceremony

Please join us in the in the **auditorium** for the closing ceremony **right after the last session** of the day.

This is where sponsors will give **raffle prizes**.

Fabric Pipelines
Metadata-driven ETL Patterns

Mike Diehl
Director of Data Engineering and Business Intelligence
Mike.Diehl@improving.com

/mikediehlsqlbi

1997 imaginet

improving

18 OFFICES

Why use metadata for Fabric Pipelines?

- Reduces ETL development, increases velocity
- Compared to….SSIS
- Lakehouse tables - schema evolution (vs SQL database)
- Code and slides:
  https://github.com/xhead/SqlSaturdayATL-2025

Survey:
- SQL Server Integration Services
- Azure Data Factory
- Fabric Pipelines

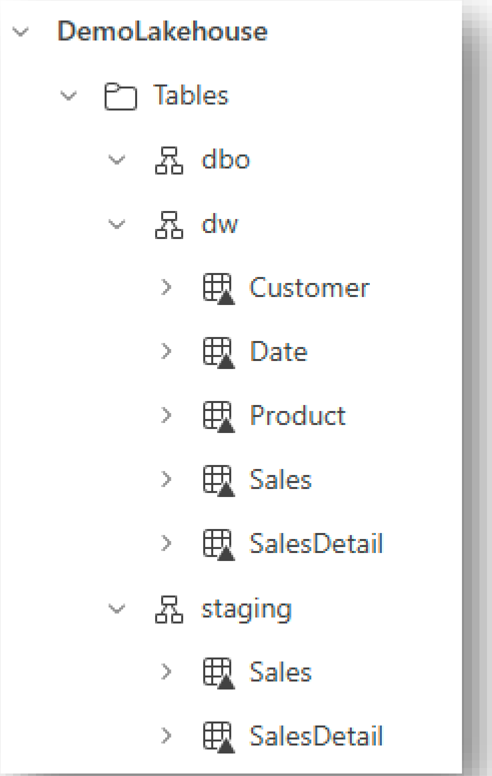
Jens Vestergaard
ETL Orchestration: Air Traffic Control for Data

https://t-sql.dk/2025/02/etl-orchestration-air-traffic-control-for-data/

Scenario: Metadata-driven Approach

Target: Lakehouse in OneLake

- Set of tasks:
  - Get data from source
    - Load all data, Overwrite all data
    - Load some data in staging area, merge it into target (incremental)
- Dependencies
  - Load dimensions first, then facts that depend on dimensions
- Develop Pipelines at the Data Source type level
  - SQL server, Oracle server, plus authentication type
  - File source (File System, SharePoint, Azure) and type (XML, JSON, CSV)



Data Warehouses, Data Lakes, and Data Lakehouses

Data Warehouse

- Structured data
- Great for BI
- Quickly becomes expensive at larger data sets

Data Lake

- Unstructured data
- Great for BI and ML/AI
- Better economics for large data sets

Lakehouse

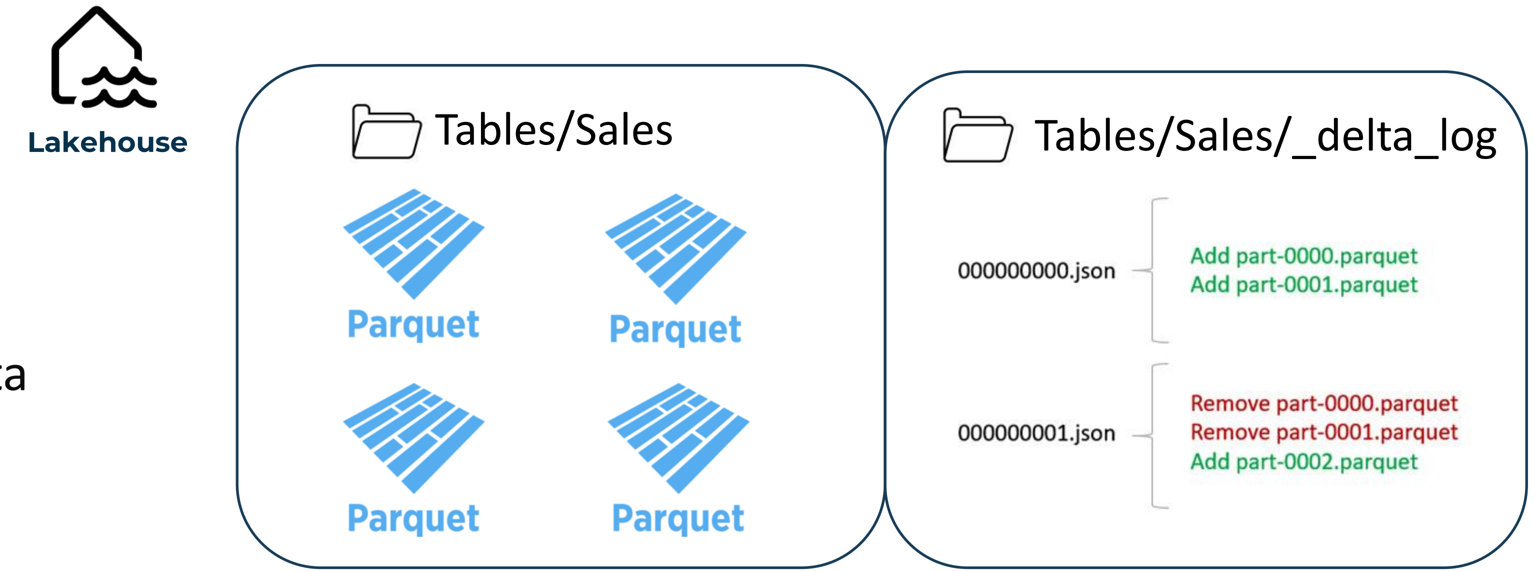
- Both structured and unstructured data in one place
- Blazing-fast
- Great for BI and ML/AI
- Great economics

iomete

improving

Delta tables

OneLake Lakehouse



- Parquet files
  - Compressed, column-store data
  - Fast queries (like Vertipaq)
- Transaction log (json)
  - Time travel
- Easy schema evolution
- Optimize/vacuum
  - Rewrites/removes parquet files
- Easier/faster to drop/overwrite than to update
  - Unlike RDBMS: use change detection to minimize changes from staging to dw

Viewer Discretion Advised

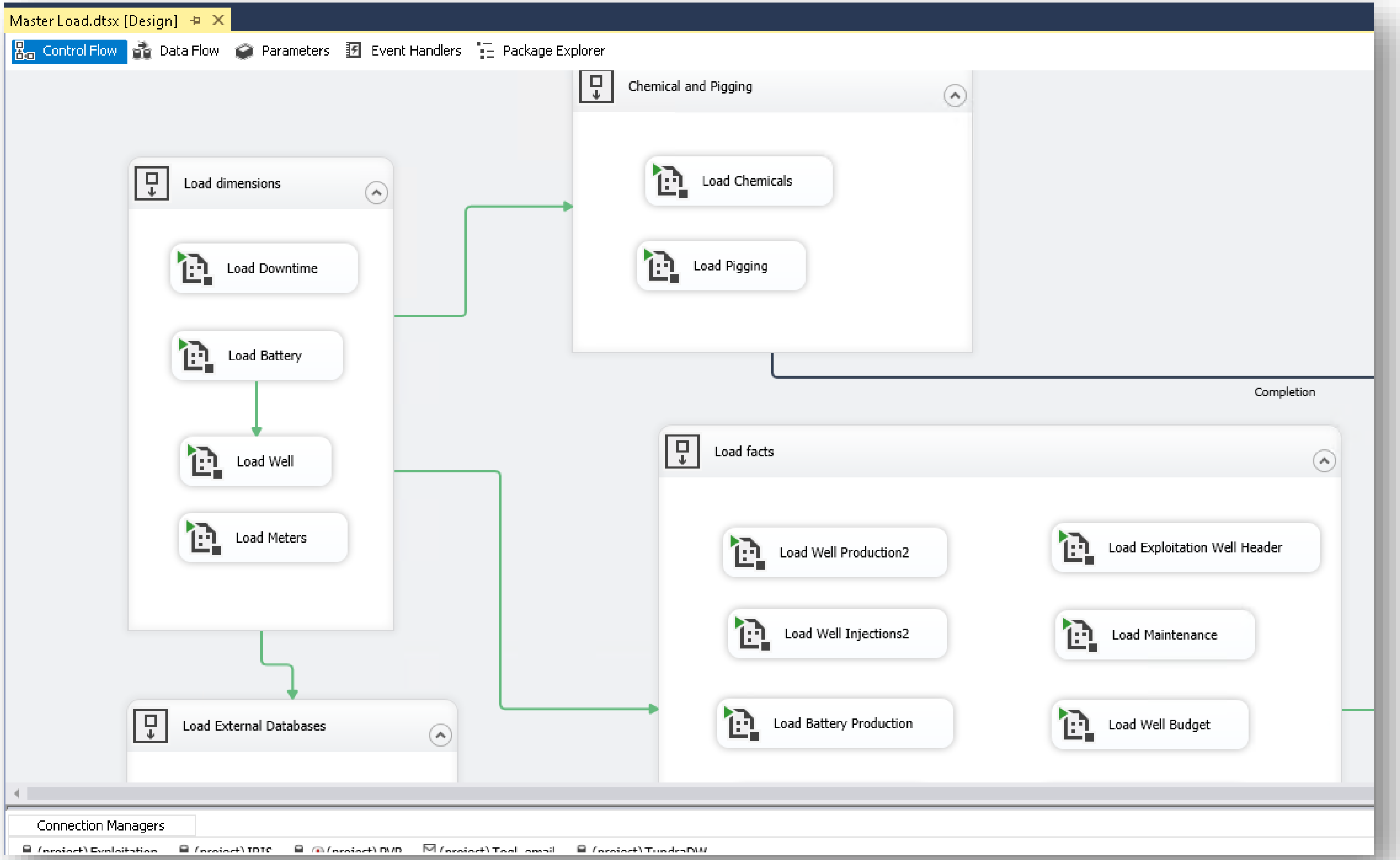
These slides may contain screenshots that are disturbing.

Those that have bad memories of SSIS development may be triggered.

SSIS Approach

Develop at the COLUMN Level

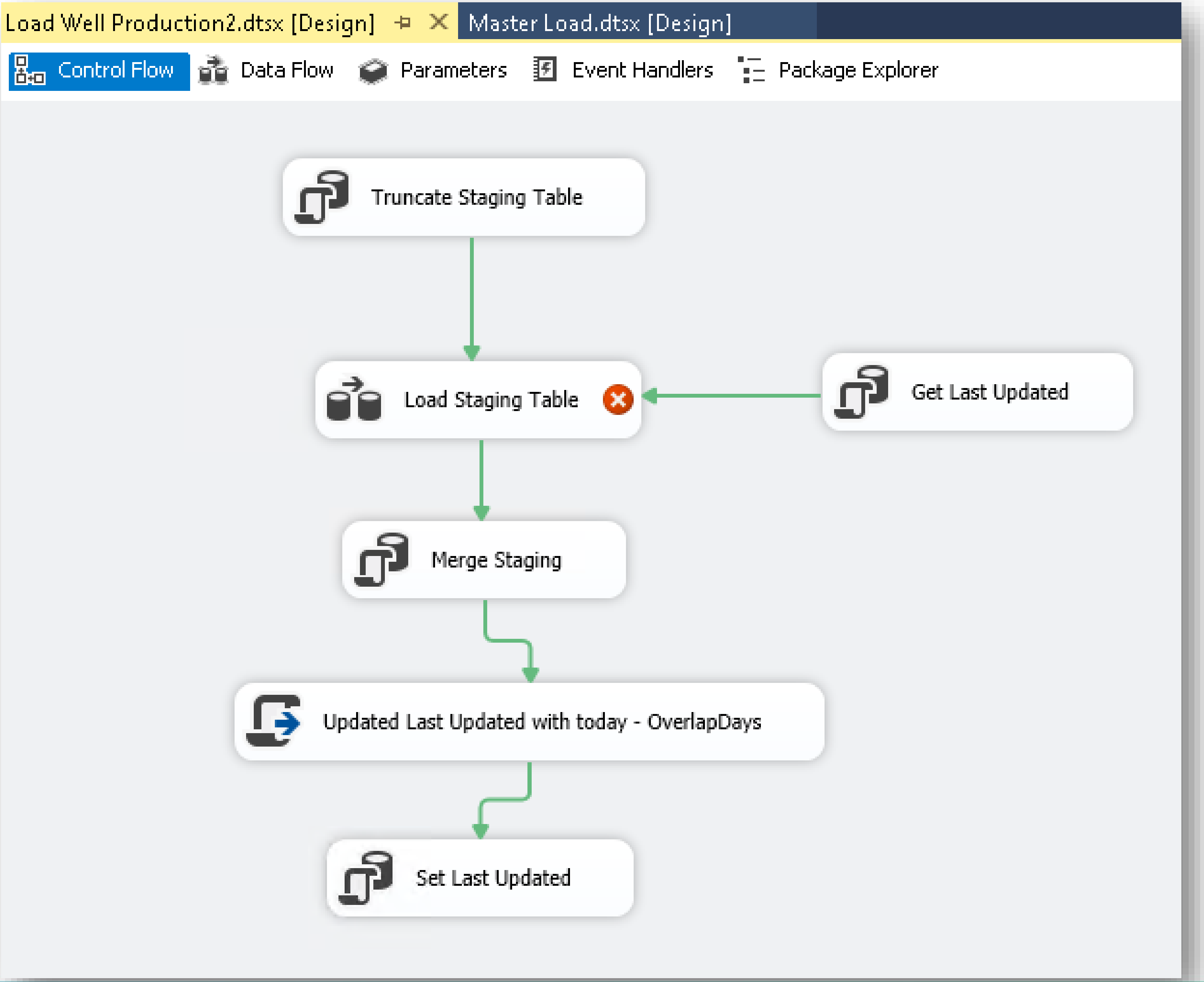
- Column changes require updating of mappings in Data flow tasks
- Redeploy updated SSIS packages



SSIS Approach

Develop at the COLUMN Level

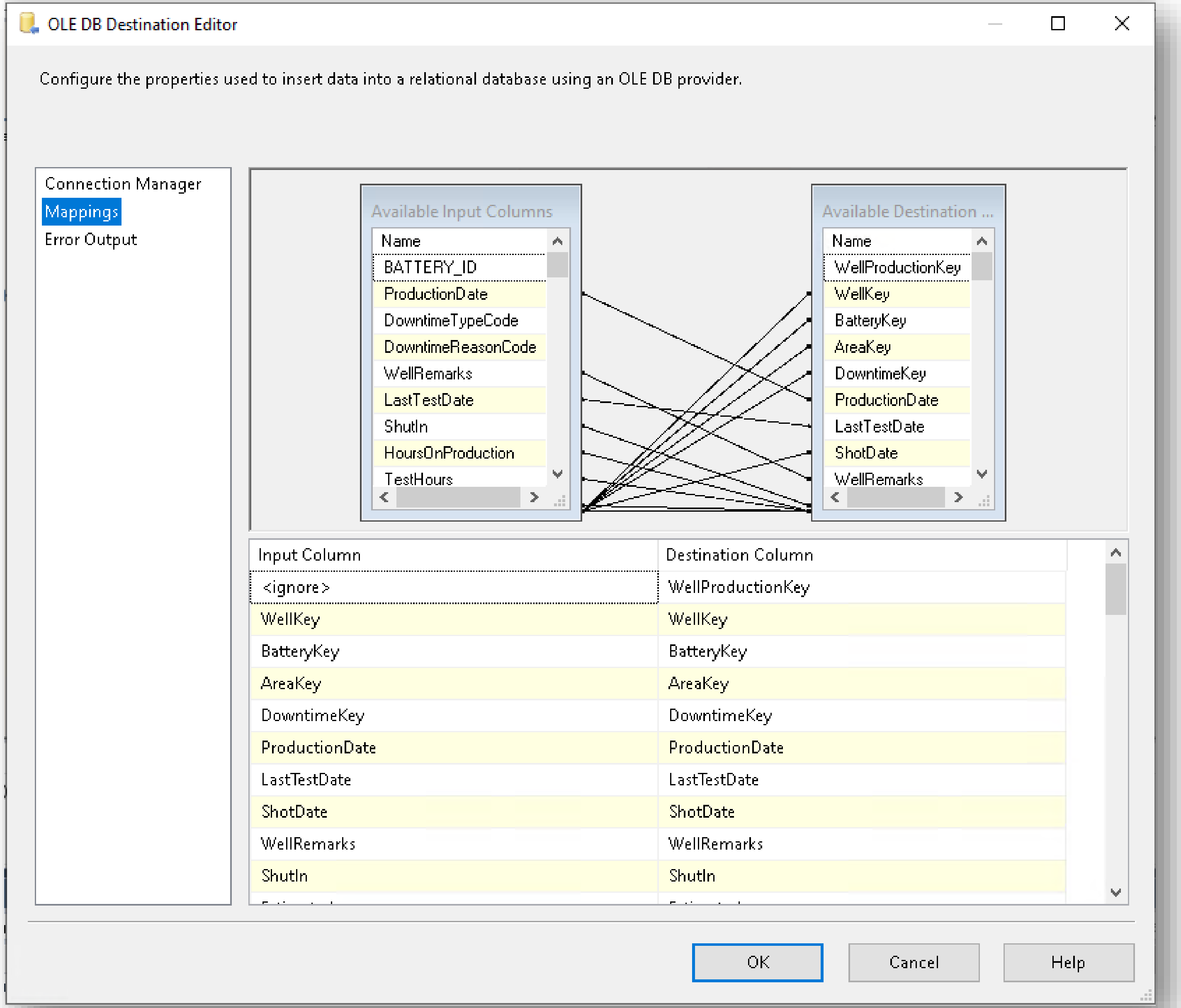
- Column changes require updating of mappings in Data flow tasks
- Redeploy updated SSIS packages



SSIS Approach

Develop at the COLUMN Level

- Column changes require updating of mappings in Data flow tasks
- Redeploy updated SSIS packages



improving

Advantages of Metadata-driven Approach

ETL Change	SSIS	Deploy SSIS?	Fabric
New data from new source	New data connection, new package or dataflow,	Yes	New service connection, new pipeline, new metadata files
New data from existing source	New package or dataflow	Yes	Add task metadata files
Add/change/drop column in existing source	Refresh column metadata in source & target	Yes	Update task metadata files
Change dependency	Update package	Yes	Update task metadata files, update pipeline (sometimes)
Stop loading existing data	Remove or disable package	Yes	Update task metadata files

improving

Fabric Pipeline features

Parameters

Parameters | Variables | Settings | Output

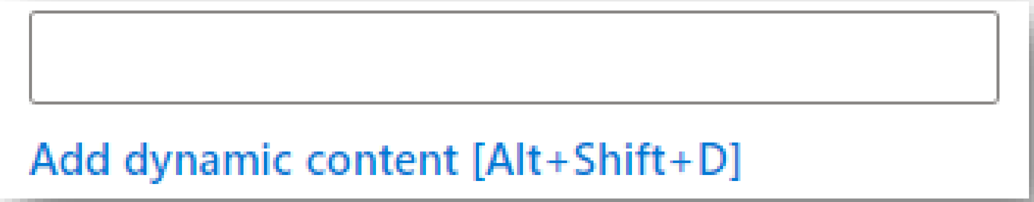
+ New | Delete

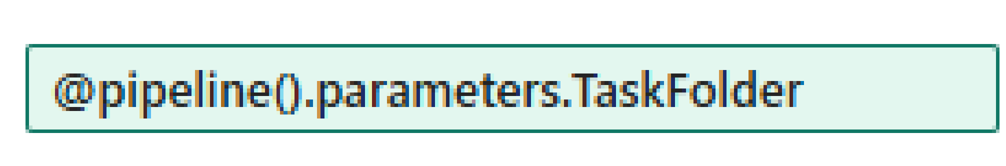
	Name	Type	Default value
☐	TaskFolder	String	PipelineTasks
☐	TaskFile	String	SampleDBDaily.json
☐	SourceQueryFilter	String	***ALL***

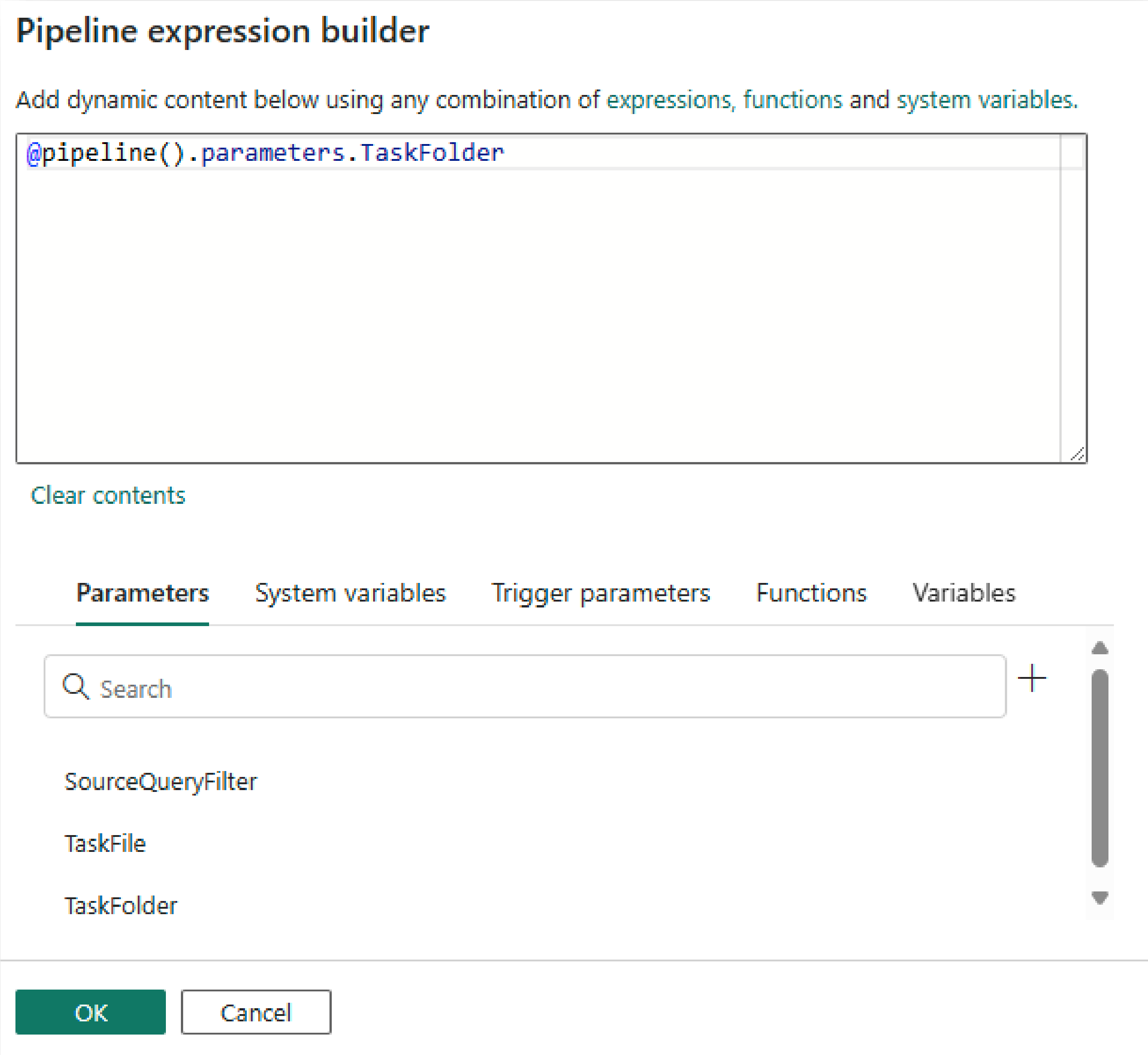
improving

Fabric Pipeline features

Dynamic Content





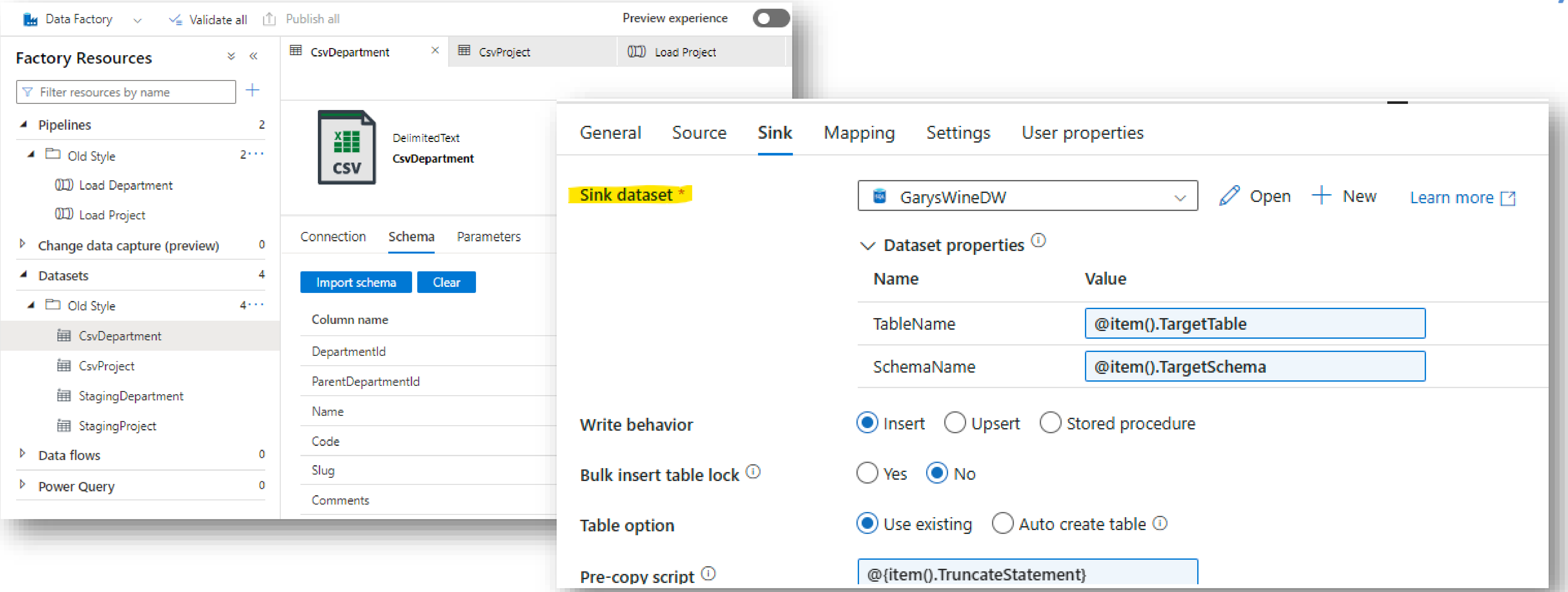


improving

Azure Data Factory → Fabric Pipelines

No more Datasets!

Azure Data Factory



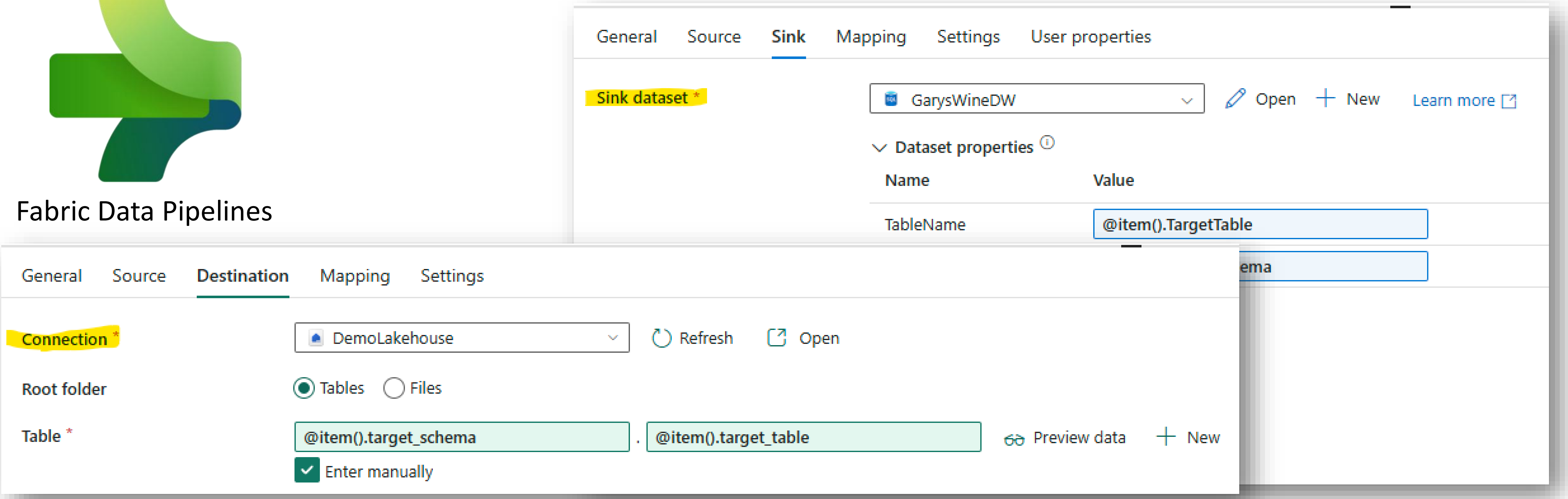
improving

Azure Data Factory → Fabric Pipelines

No more Datasets!

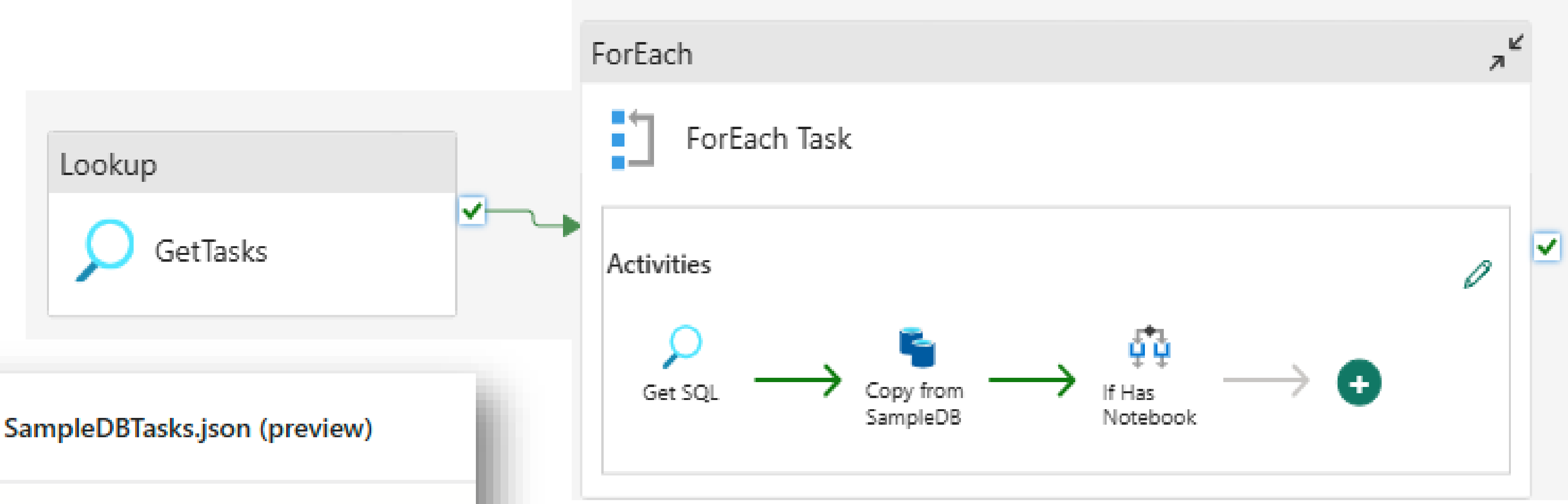
Azure Data Factory

Fabric Data Pipelines



Typical Metadata-driven Pipeline Pattern

Fabric Pipeline



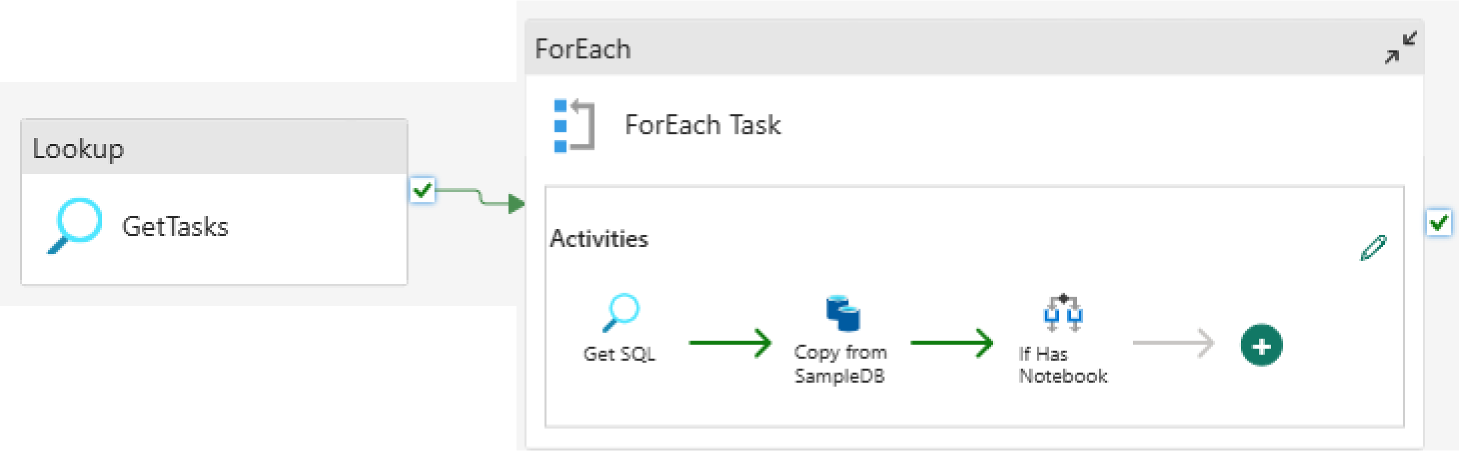
Files > PipelineTasks > **SampleDBTasks.json (preview)**

```json
[
    {
        "source_query": "Products.sql",
        "target_schema": "dw",
        "target_table": "Product",
        "notebook": ""
    },
    {
        "source_query": "Customers.sql",
        "target_schema": "dw",
        "target_table": "Customer",
        "notebook": ""
    },
```

Lakehouse

Fabric: Typical Pipeline
(per data source type)

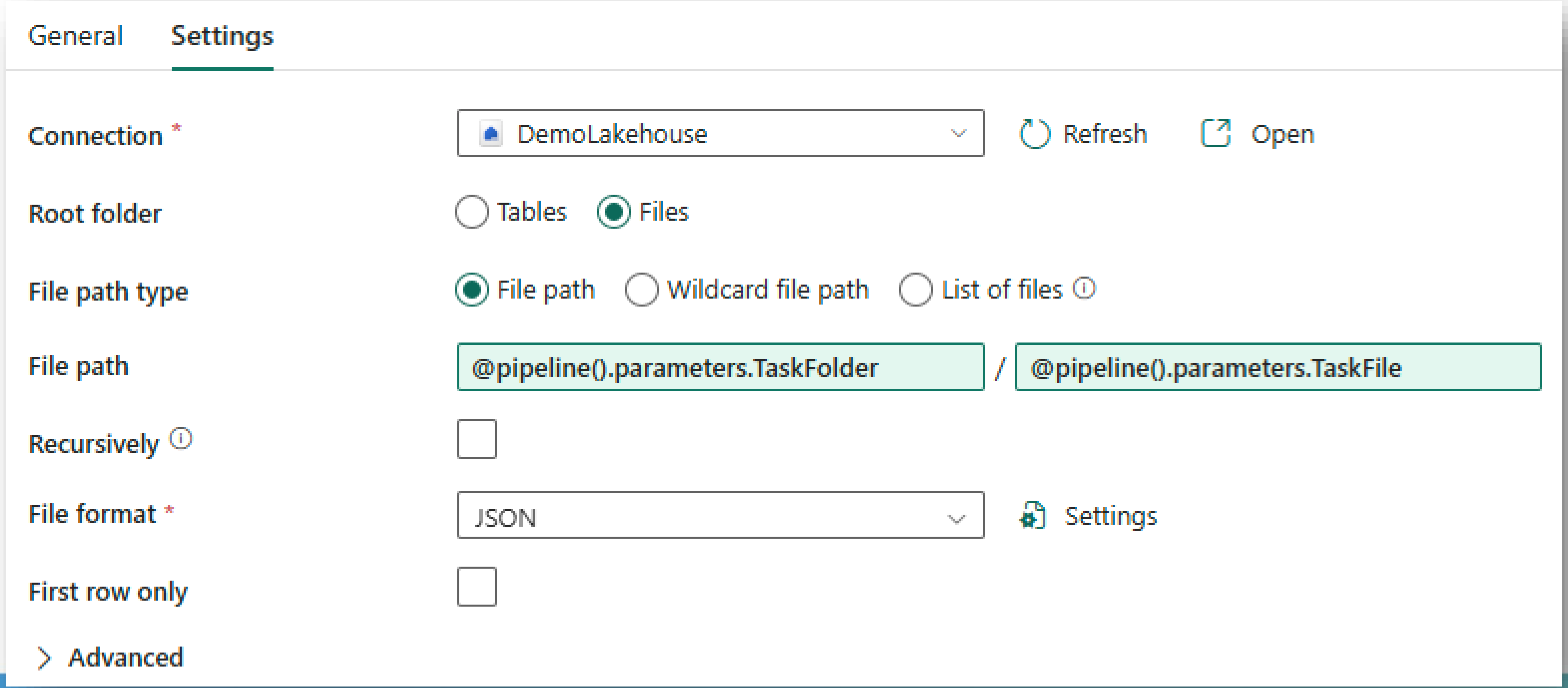
- Get Tasks (Lookup)
  - Gets a json file from Lakehouse files
  - (filter for debug purposes)
- For Each Task
  - Get SQL query from Lakehouse file (Lookup)
  - Copy Data Task
  - Execute Notebook



Get Tasks (Lookup)

Gets a JSON file from Lakehouse

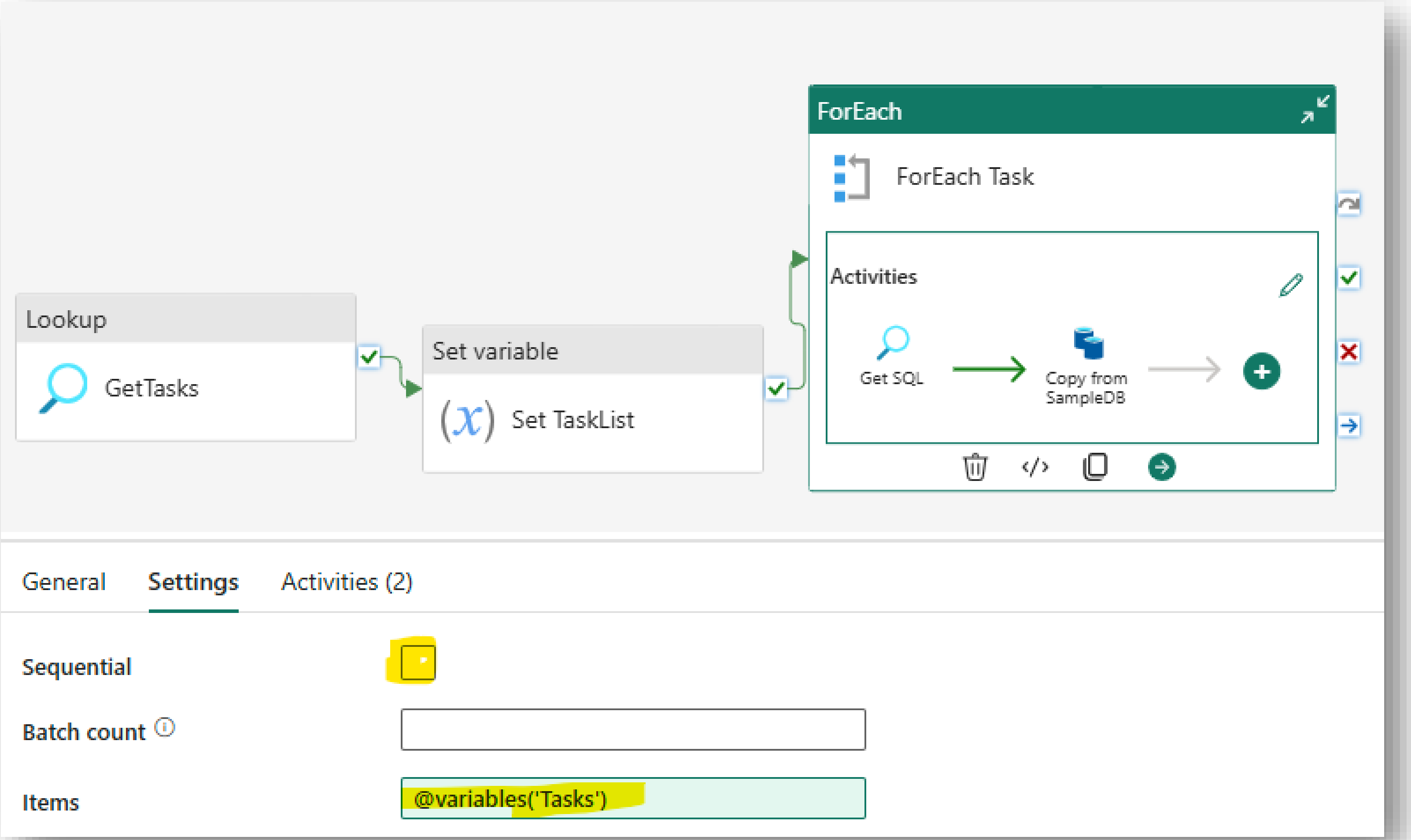
- Task folder and file name parameters
- Contains JSON array



For Each Task

Iterates through item array

- Output of GetTasks
- Item() object
- Parallel or serial execution

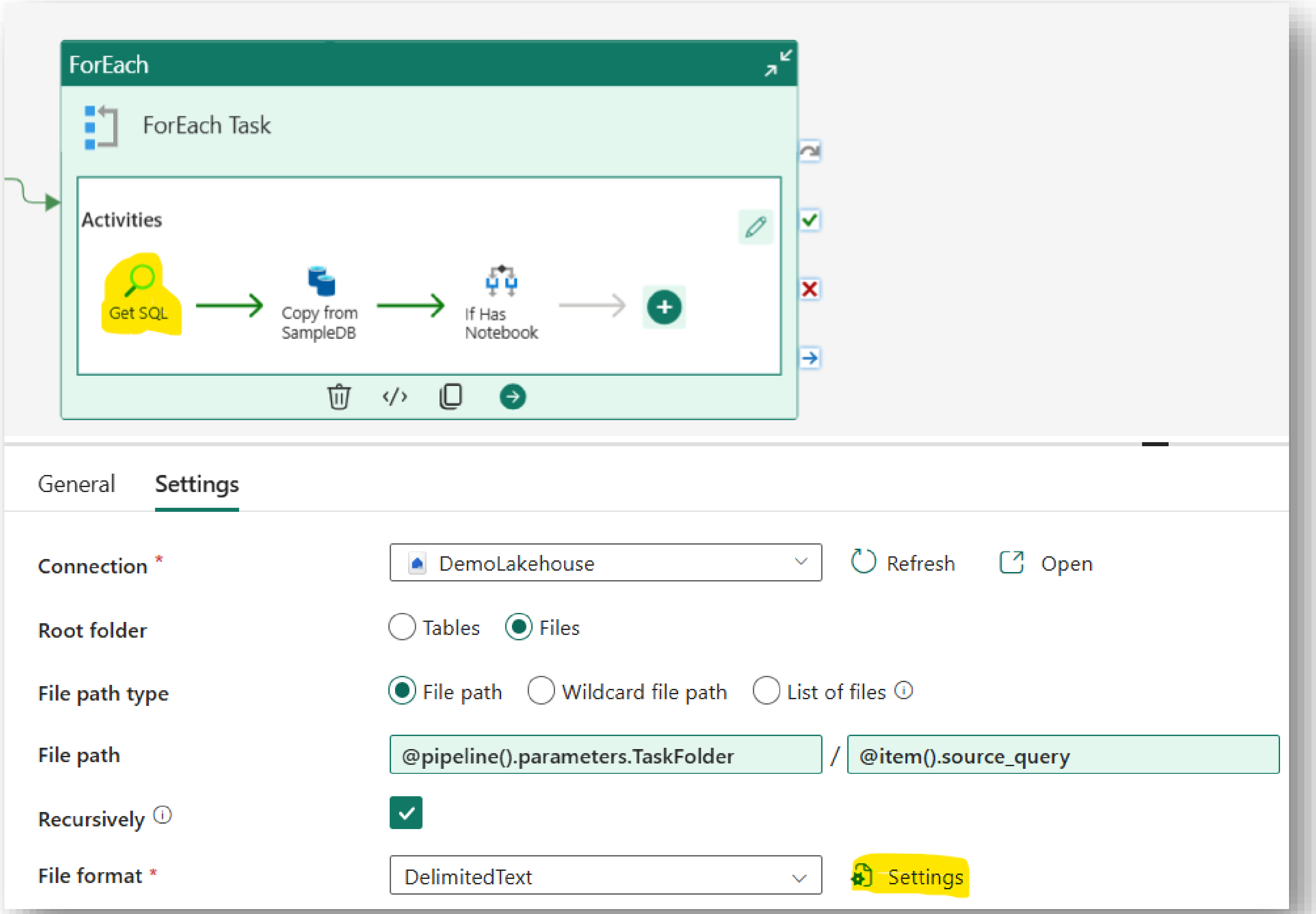


improving

For Each – Read SQL

Lookup task to get query text

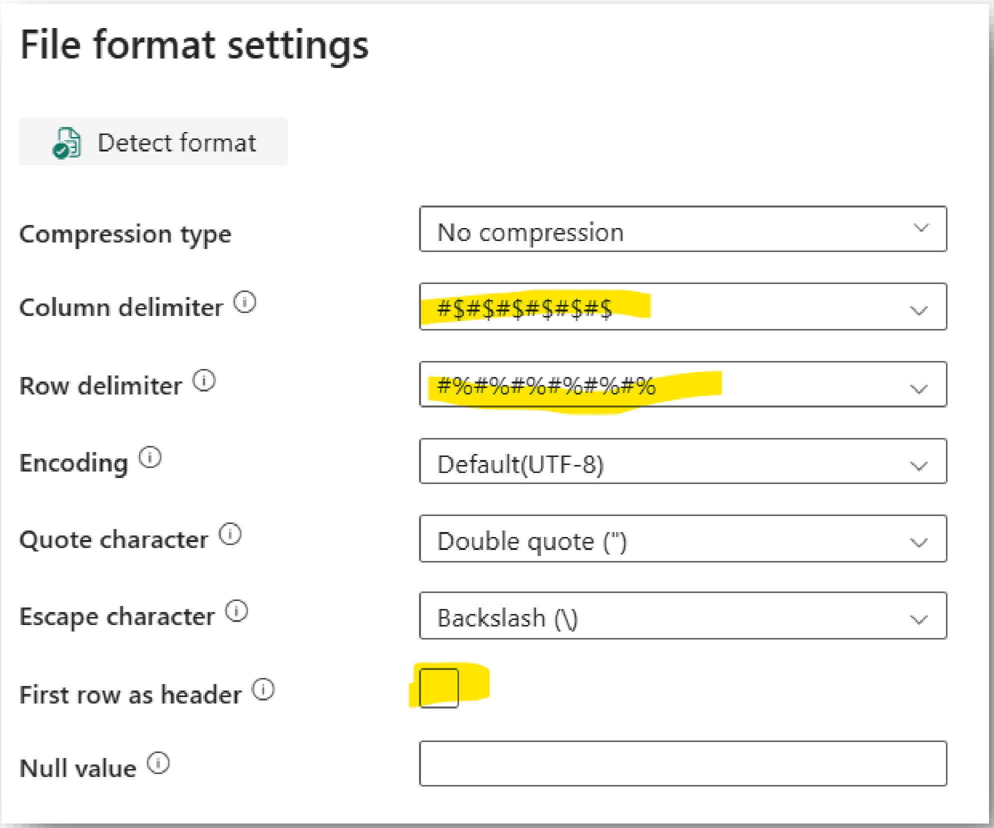
- Read file as DelimitedText
- Set delimiters so only one row, one column is found
- No Header



For Each – Read SQL

Lookup task to get query text

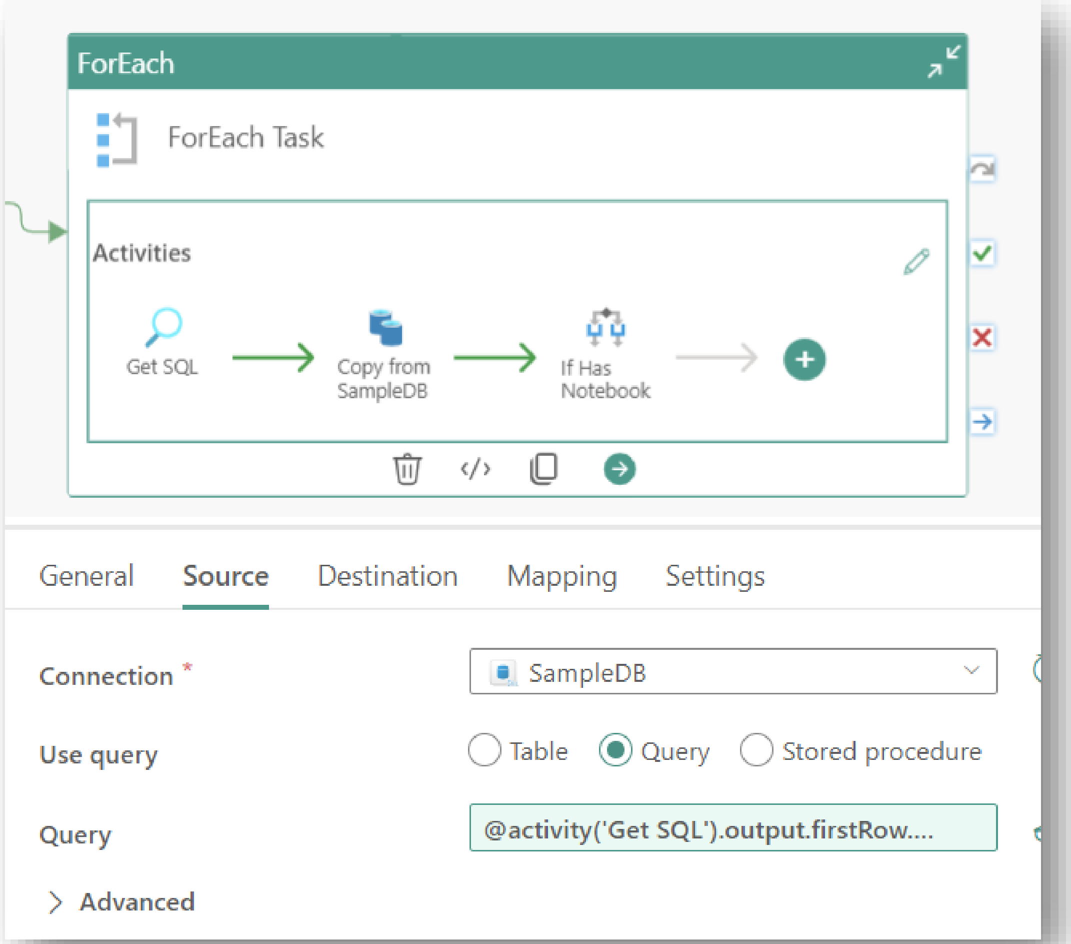
- Read file as DelimitedText
- Set delimiters so only one row, one column is found
- No Header



For Each – Copy Data Task

Source

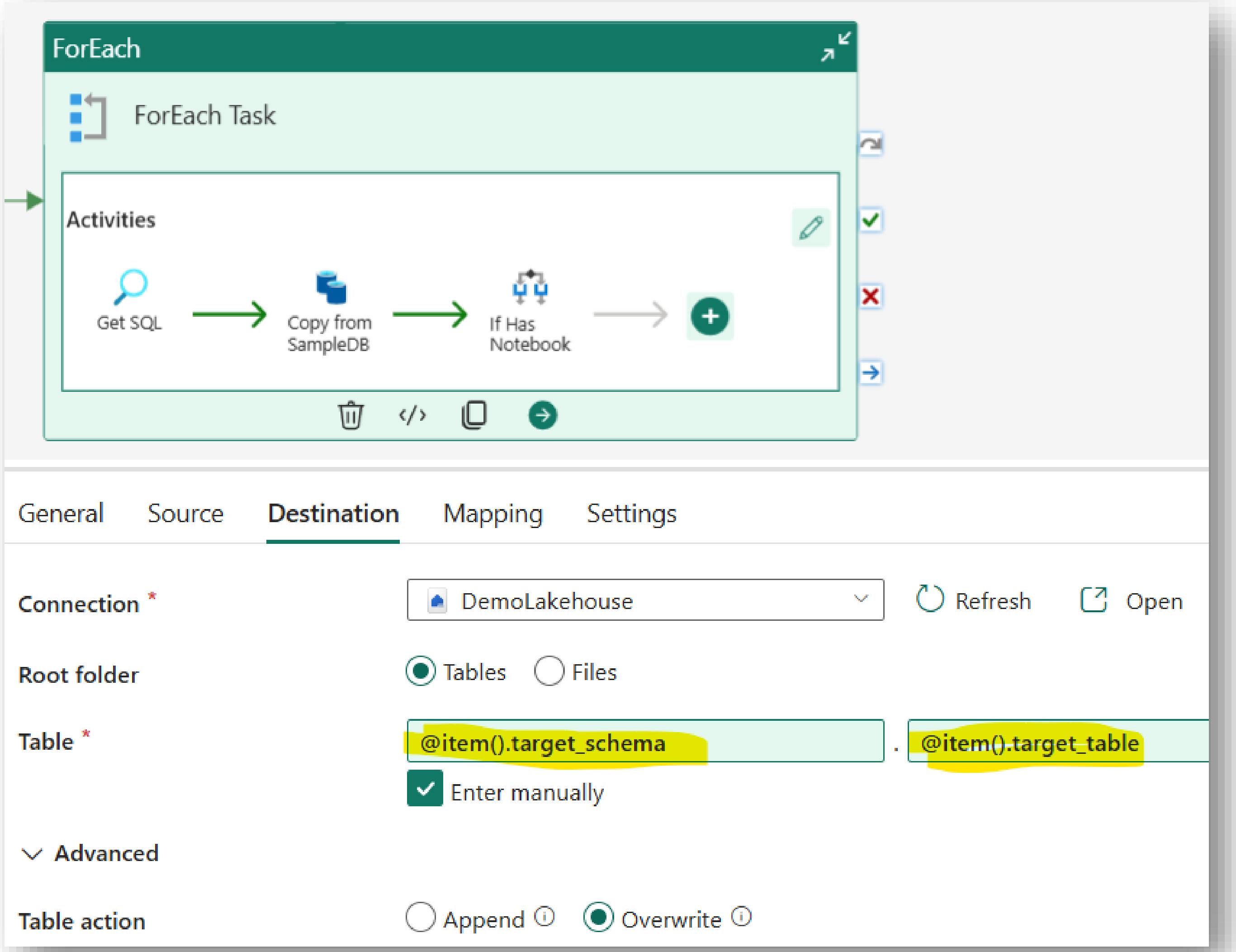
- Dynamic property for Source Query
- `@activity('Get SQL’).output .firstRow.prop_0`



For Each – Copy Data Task

Destination

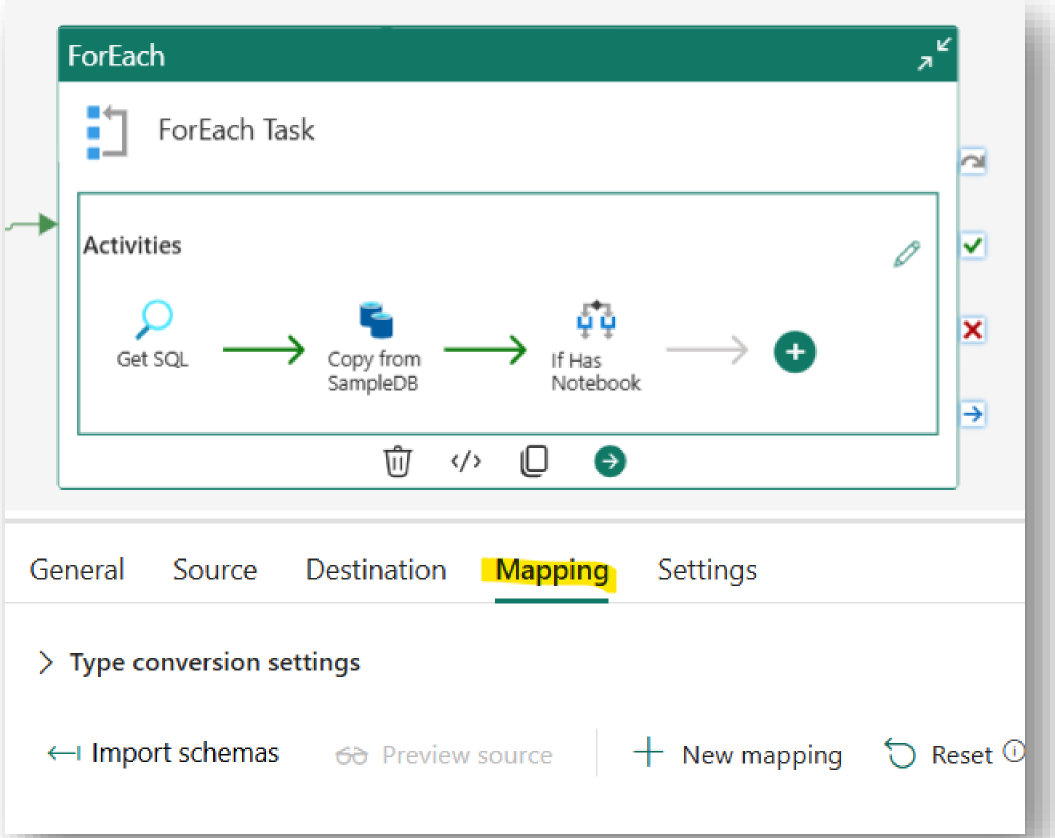
- Dynamic properties for Target Schema & Table
- Overwrite
- Target table may not exist
  - But schema needs to exist



For Each – Copy Data Task

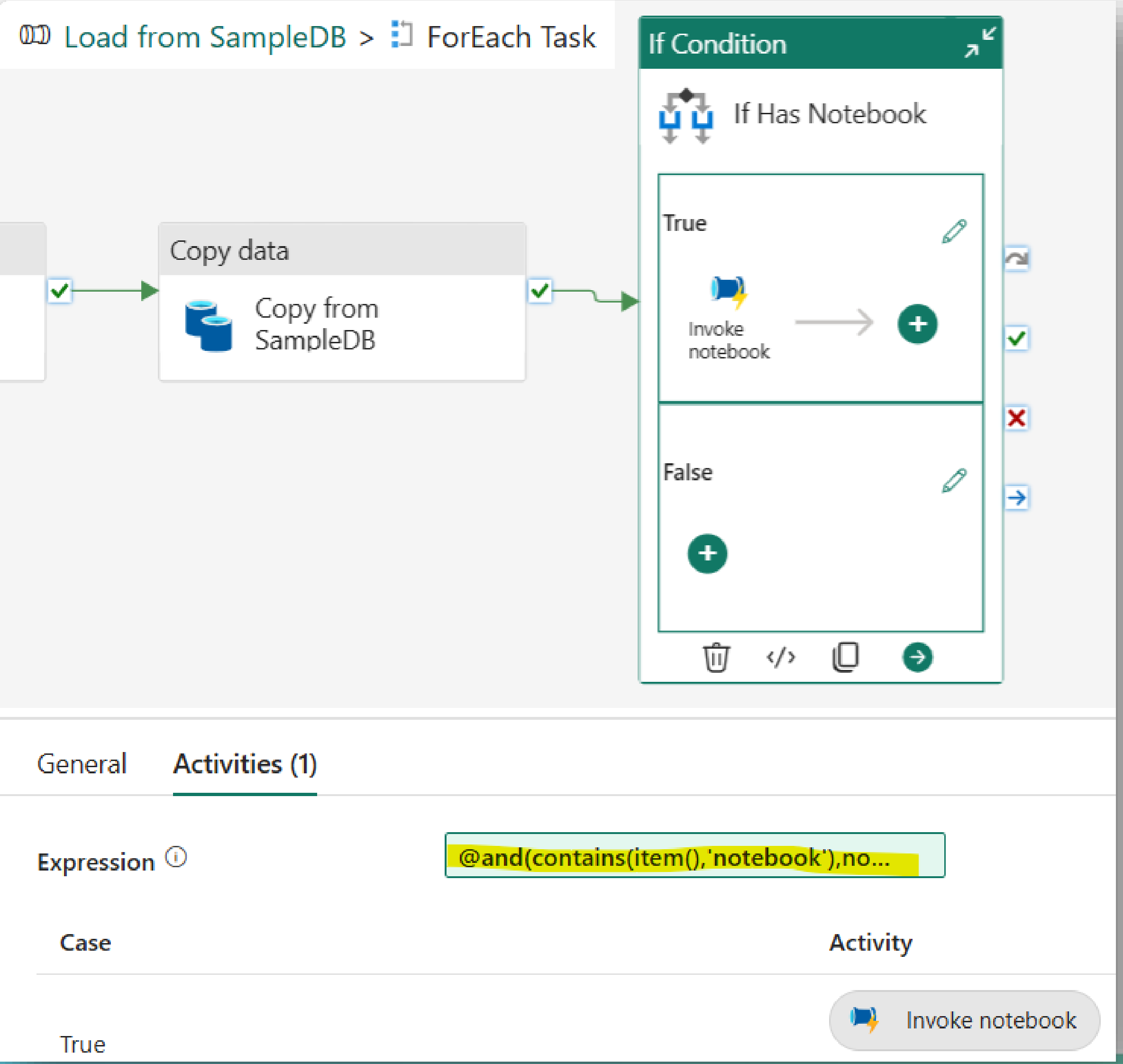
Mapping

- **Zero** column mapping (very different from SSIS)



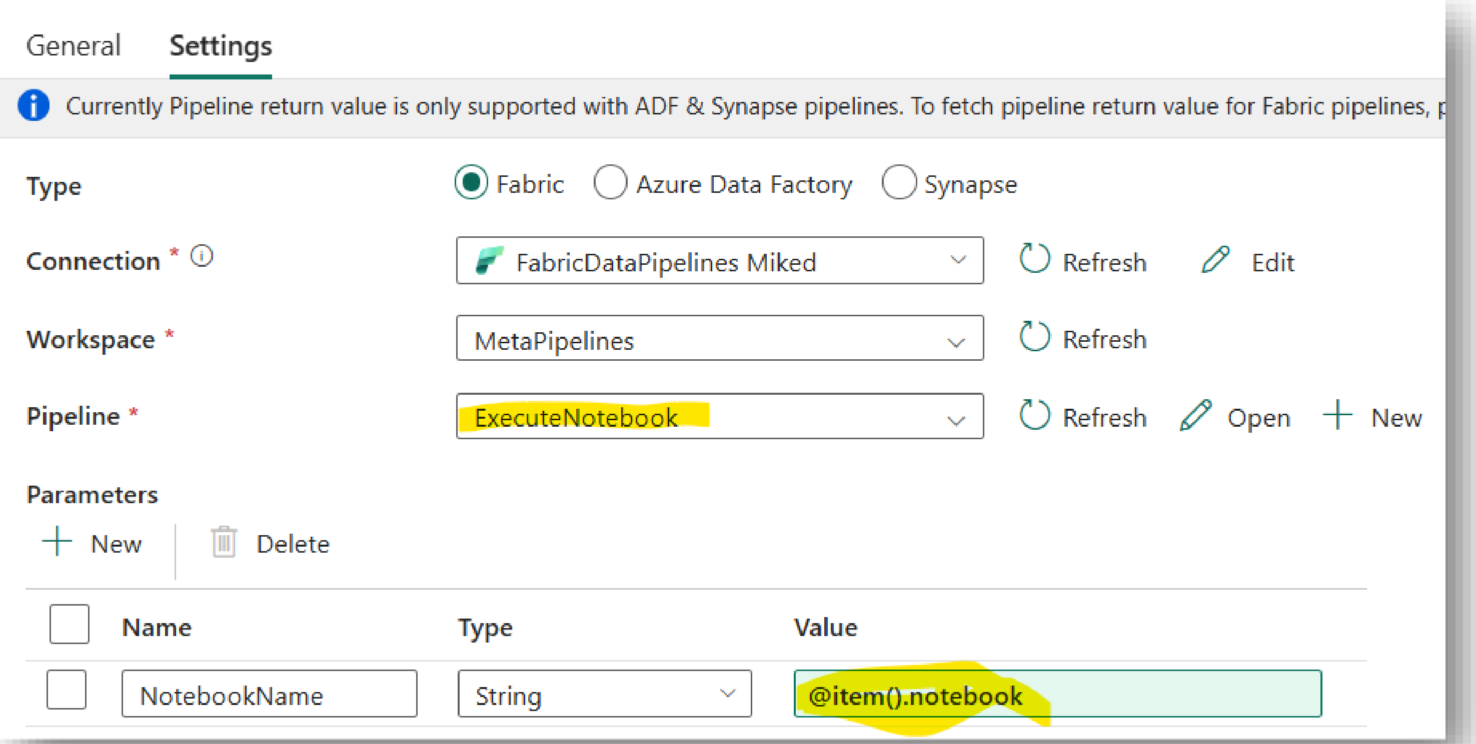
For Each – Execute notebook

- Execute notebook to process staging data
- `@and(contains(item(),'notebook'), not(empty(item().notebook)))`



For Each – Execute notebook

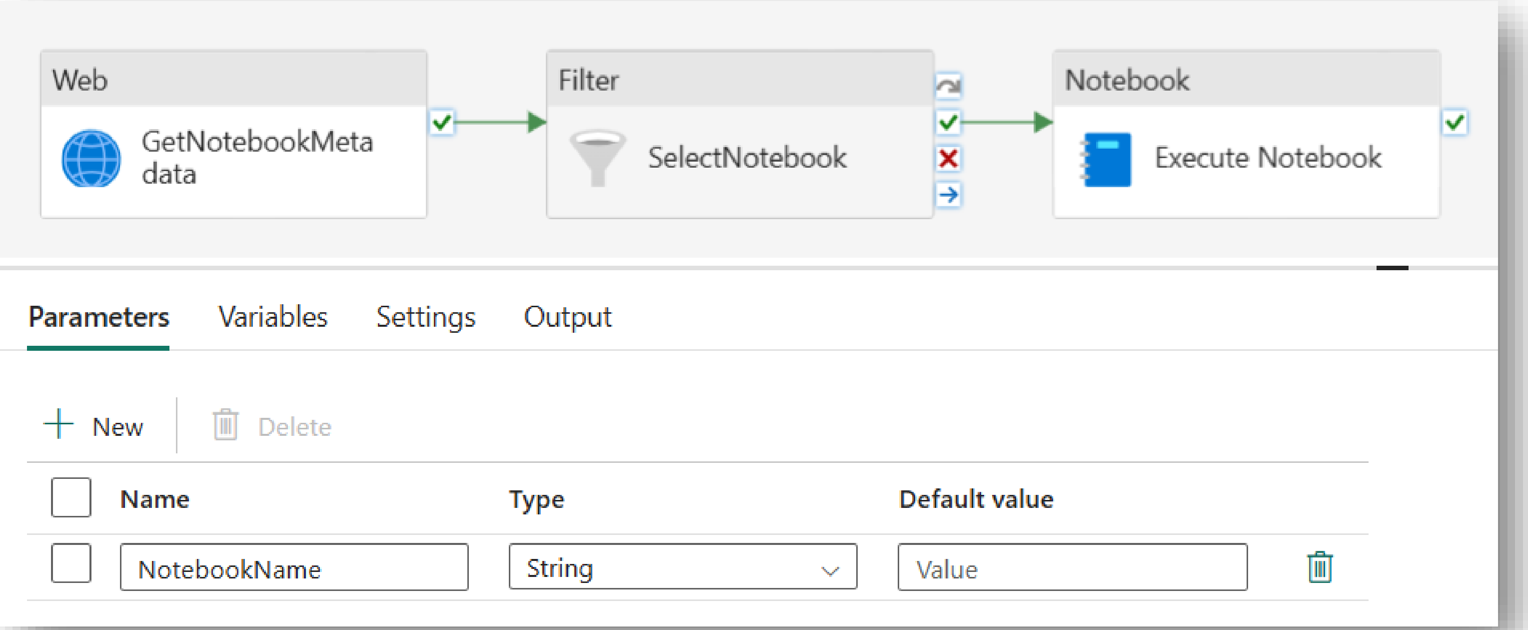
- Calls another pipeline to execute the notebook



For Each – Execute notebook

Invoke Notebook pipeline

- Need the resource ID of the notebook, not the name
- REST API gets all notebook metadata
- Filter selects by name
- Dynamic properties for notebook execution task



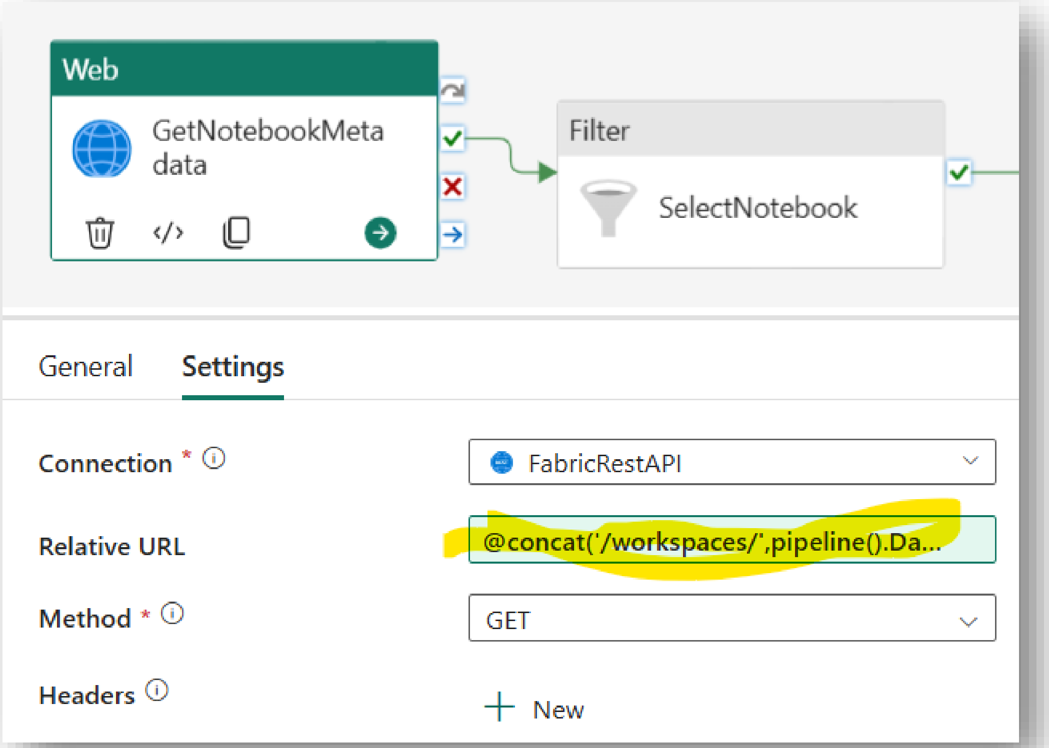
For Each – Execute notebook

Invoke Notebook pipeline

- Need the **resource ID** of the notebook, **not the name**
- REST API gets all notebook metadata
- Base URL: https://api.fabric.microsoft.com/v1/
- Relative URL:

```
@concat('/workspaces/',pipeline()
.DataFactory,'/items?type=Notebook')
```

  - (all notebooks in this workspace)

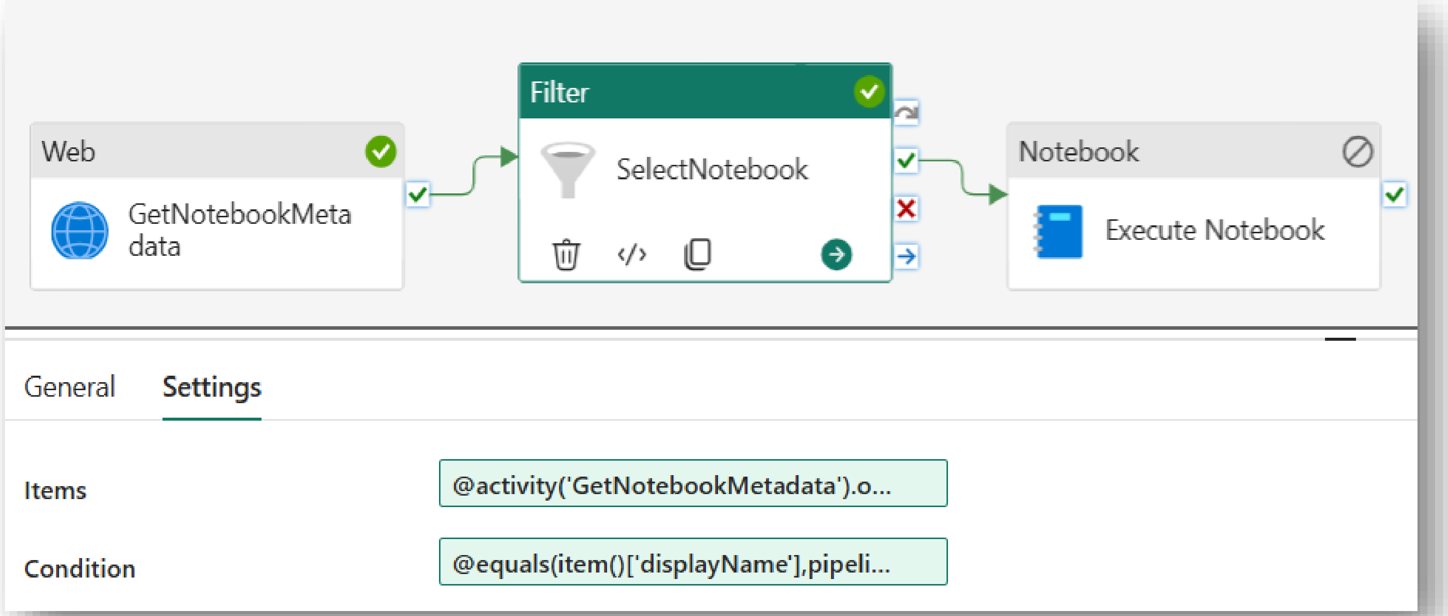


improving

For Each – Execute notebook

Invoke Notebook pipeline

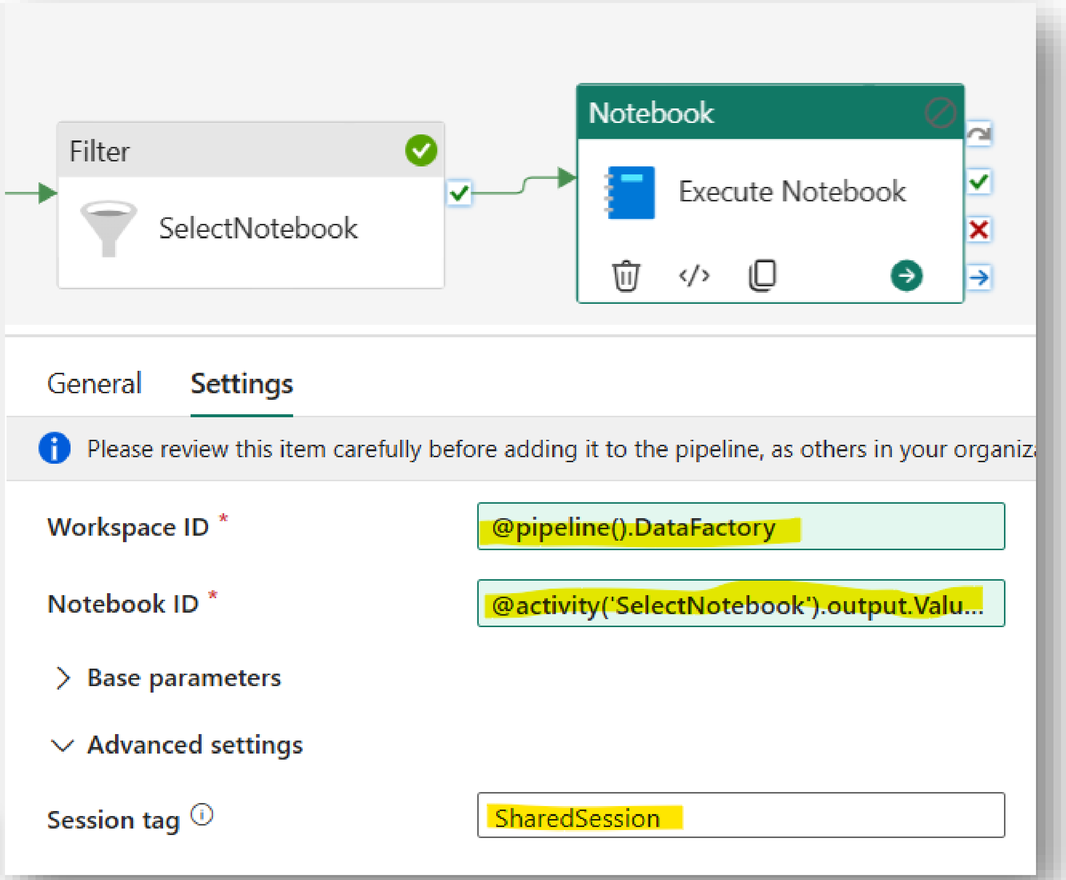
- Filter selects by name
- Items: `@activity('GetNotebookMetadata').output.value`
- Condition: `@equals(item()['displayName'],pipeline().parameters.NotebookName)`



For Each – Execute notebook

Invoke Notebook pipeline

- Workspace ID:
  `@pipeline().DataFactory`
  ("this workspace")
- Notebook ID:
  `@activity('SelectNotebook')`
  `.output.Value[0].id`
- Session tag

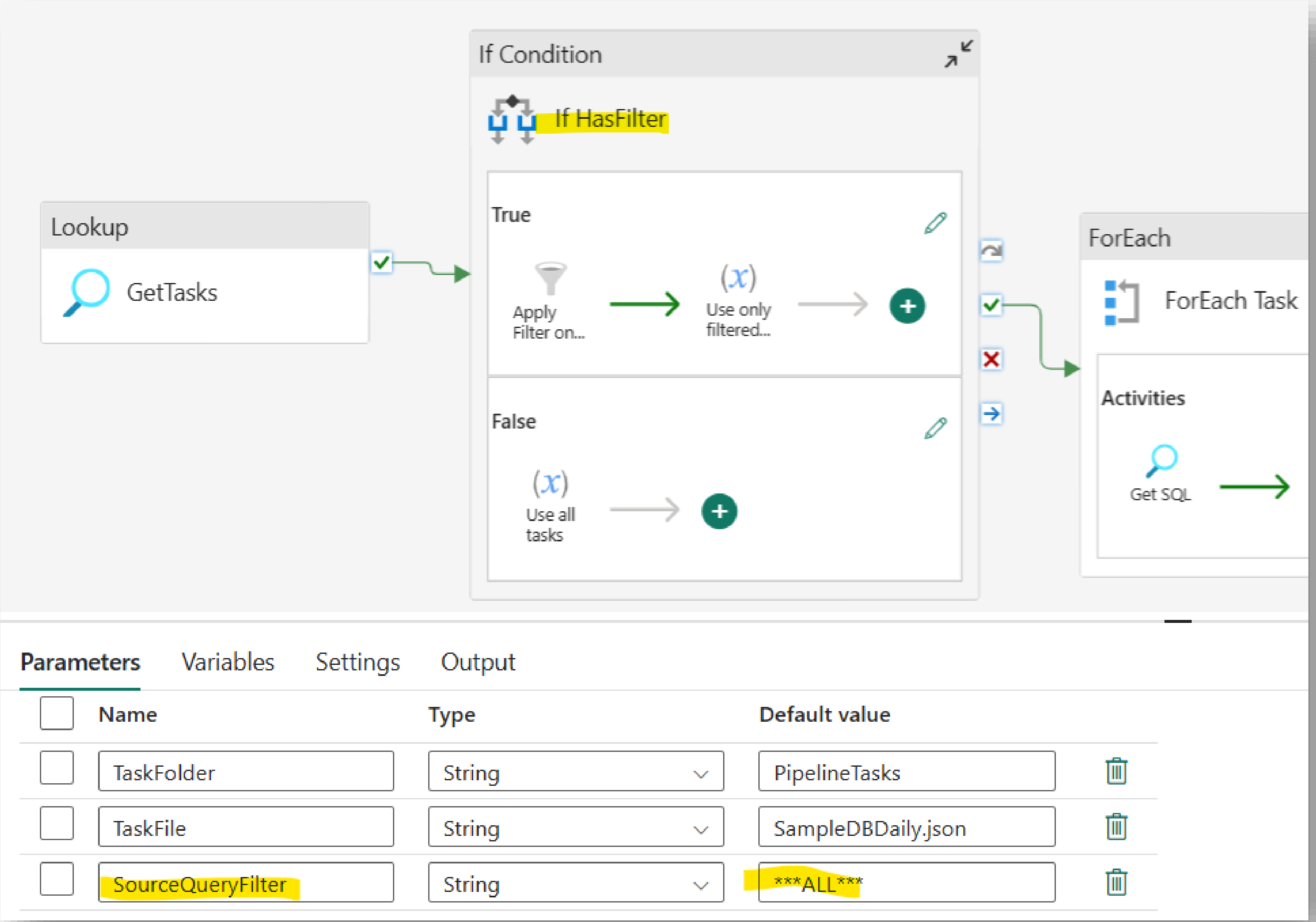


Debugging tasks

Filter by task name

- If HasFilter:

```
@not(equals(pipeline()
.parameters.SourceQueryFilter
,'***ALL***'))
```

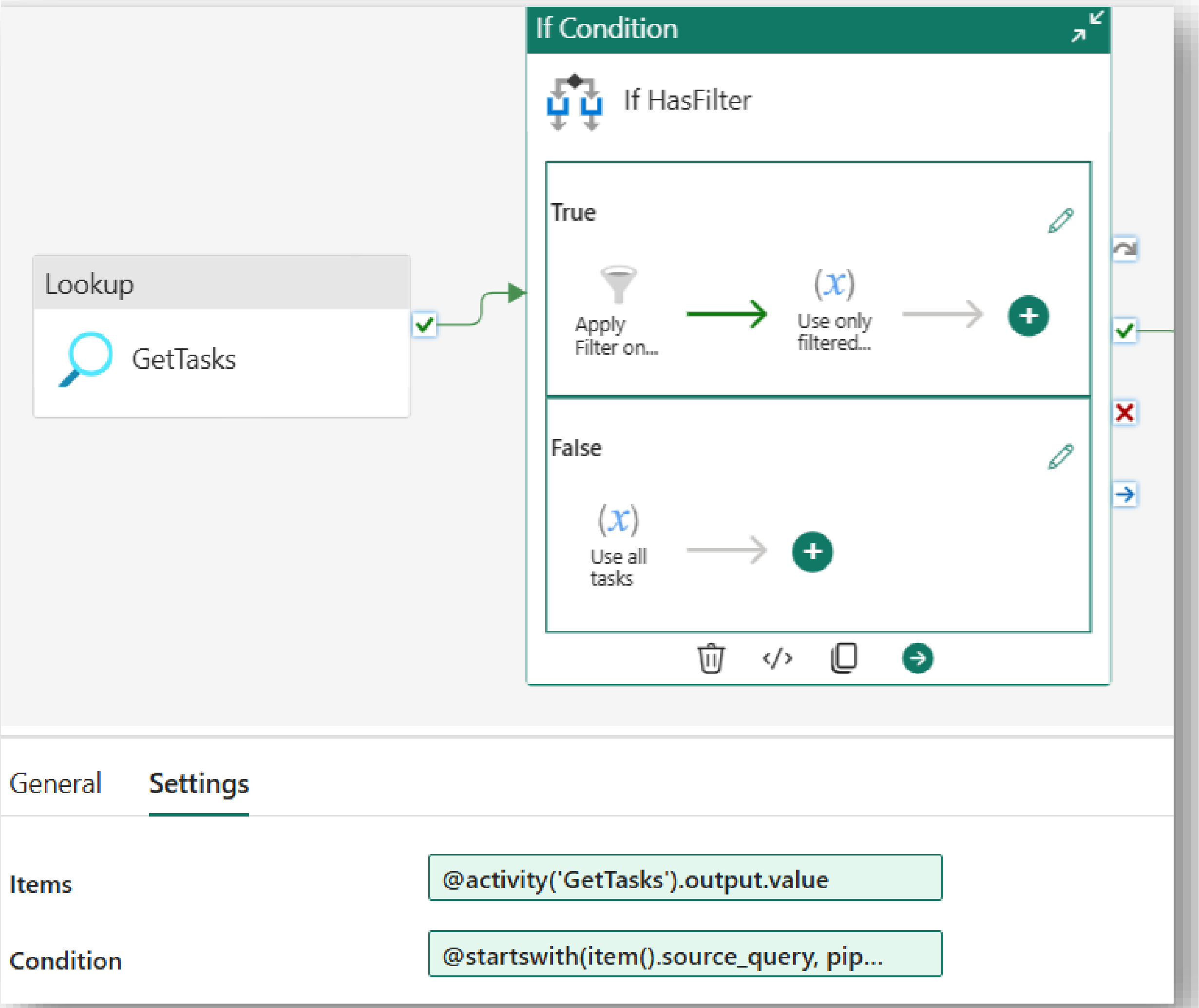


improving

Debugging tasks

Filter by task name

- Filter Condition:
  `@startswith(item().source_query ,pipeline().parameters.SourceQueryFilter)`
- Set tasks variable



How Many Pipelines?

- Usually One per Data Source, plus one Main pipeline
- Most pipelines use the Lookup/ForEach/Copy Data/Exec notebook pattern

Main Pipeline

- Calls Task-driven pipelines with parameter for task file
- Main pipeline handles dependencies
  - "Level 1" pipelines – loading dimensions
  - "Level 2" pipelines – loading facts
  - Additional levels as needed, based on dependency tree
- Advanced: Main pipeline can use the Lookup/ForEach pattern to get a list of pipelines (and their parameters) to execute
- Could arrange pipelines in medallion order

Notebooks

Incremental loading

- Staging table contains complete data for a range of dates
  - CurrentYearSales.sql (from Jan 1 of this year)
  - PastYearSales.sql (before Jan 1 of this year)
- Notebook: MergeSales

```
%run CommonLakehouseMerge
MergeByDateRange("staging.Sales", "dw.Sales", "OrderDate")
```

```
def MergeByDateRange(staging_table,target_table, date_range_replace_column):
```


improving

Notebooks

Incremental loading

- Staging table contains data identified by natural key
  - CurrentYearSalesDetails.sql (no Order Date)
- Notebook: MergeSalesDetails

```
%run CommonLakehouseMerge
MergeUpsert("staging.SalesDetail", "dw.SaleDetail",
"SalesOrderID")
```

```
def MergeUpsert(staging_table,target_table, key_column):
```

CommonLakehouseMerge

MergeByDateRange

- Check that target table exists
- Gets Min and Max values
- Builds ReplaceWhere clause
- Merge Schema

```python
def MergeByDateRange(staging_table,target_table, date_range_replace_column):

    target_df = spark.sql(f"select * from {target_table}")
    source_df = spark.sql(f"select * from {staging_table}")

    #check if table has any columns or has already columns but they are empty
    if len(target_df.columns)==0 or target_df.isEmpty():
        # first time insert when nothing to delete
        (source_df
        .write
        .format("delta")
        .mode("overwrite")
        .option("mergeSchema", "true")
        .saveAsTable(target_table)
        )
    else:
        # insert with replacement by 'date_range_replace_column'
        min_date = source_df.agg(min(col(date_range_replace_column))).collect()[0][0]
        max_date = source_df.agg(max(col(date_range_replace_column))).collect()[0][0]
        replace_where = f"`{date_range_replace_column}` between '{min_date}' and '{max_date}'"
        (source_df
        .write
        .format("delta")
        .mode("overwrite")
        .option("replaceWhere", replace_where)
        .option("mergeSchema", "true")
        .saveAsTable(target_table)
        )
```

CommonLakehouseMerge

MergeUpsert

- Opens native DeltaTable from Lakehouse path
- Merge with match clause
- Match – update
- No match – insert
- Schema evolution

```python
def MergeUpsert(staging_table,target_table, key_column):

    #read from delta tables to spark dataframes
    target_df = spark.sql(f"select * from {target_table}")
    source_df = spark.sql(f"select * from {staging_table}")

    #check if table has any columns or has already columns but they are empty
    if len(target_df.columns)==0 or target_df.isEmpty():

        )
    else:
        #read DeltaTable from Lakehouse path
        lakehouse_table_path = f"{get_lakehouse_path()}/Tables/{target_table.replace('.','/')}"
        target_dt = DeltaTable.forPath(spark, lakehouse_table_path)

        # Alias the columns to avoid name conflicts
        target_dt = target_dt.alias("target")
        source_df = source_df.alias("source")

        (target_dt
            .merge(source_df, f"source.`{key_column}` = target.`{key_column}`")
            .whenMatchedUpdateAll()
            .whenNotMatchedInsertAll()
            # .whenNotMatchedBySourceDelete()
            .withSchemaEvolution()
            .execute()
        )
```

Demo

Changing ETL Requirements

- New table in target from same source?

  ~~Add Staging and dw tables,~~

  Add an item to task file

  Add query file

  No pipeline changes required

  Merge Notebook (maybe)

- Change in target table? (Add column, remove column, change column)

  ~~Change Staging and dw tables,~~
  Update query file

  Merge Notebook (maybe)

- Incorrect logic in SourceQuery?

  Update query file

- Target table no longer needed?

  Remove item from task file

Advantages of Metadata-driven Approach

ETL Change	SSIS	Deploy SSIS?	Fabric
New data from new source	New data connection, new package or dataflow,	Yes	New service connection, new pipeline, new metadata files
New data from existing source	New package or dataflow	Yes	Add task metadata files
Add/change/drop column in existing source	Refresh column metadata in source & target	Yes	Update task metadata files
Change dependency	Update package	Yes	Update task metadata files, update pipeline (sometimes)
Stop loading existing data	Remove or disable package	Yes	Update task metadata files

improving

Questions?

Session Evaluation

Code and slides:

https://github.com/xhead/SqlSaturdayATL-2025

Mike Diehl, Mike.Diehl@improving.com
Director of Data Engineering and Business Intelligence

/mikediehlsqlbi