1. What is ADF?

a) Azure Data Factory is **Azure's cloud ETL service** for scale-out server-less data integration and data transformation.

b) It offers a **code-free UI** for intuitive authoring & single-pane-of-glass monitoring & management.

c) You can also **lift and shift existing SQL Server Integration Service packages** to Azure and run them with full compatibility in Azure Data Factory.

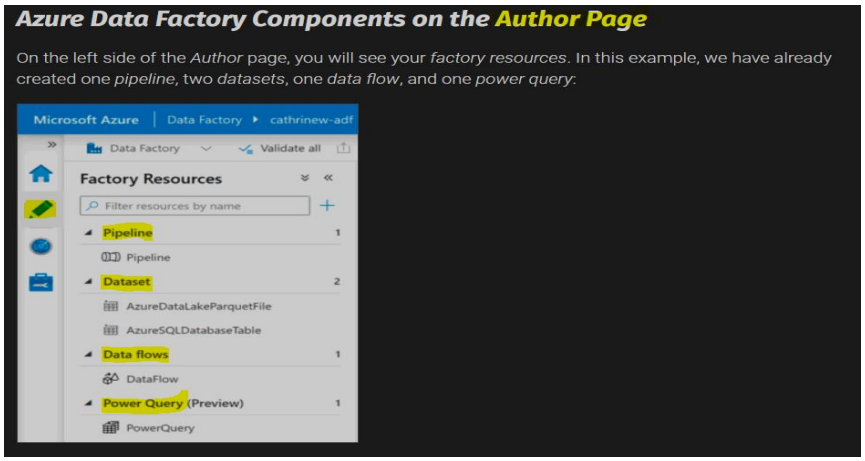
d) https://www.techbrothersit.com/2021/10/azure-data-factory-tutorial-step-by-step_23.html

2. Components of ADF:

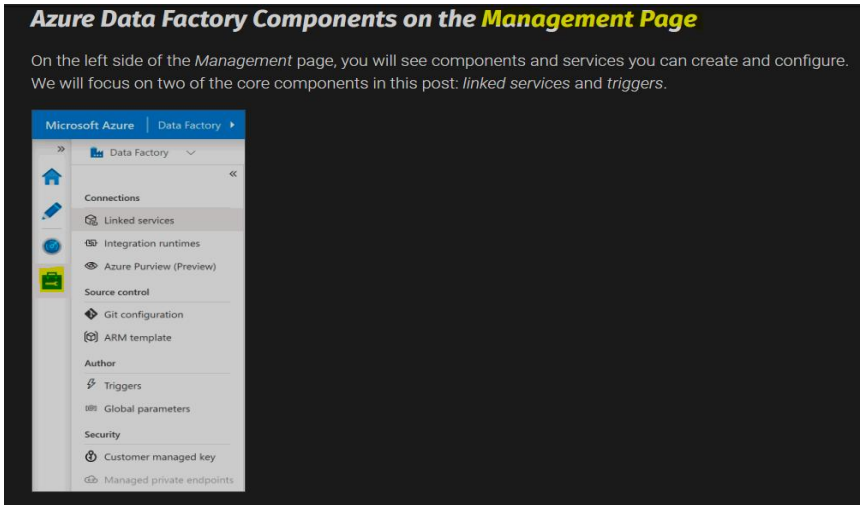
a) https://www.liktorius.com/wp-content/uploads/2020/01/Azure-Data-Factory-for-Beginners.pdf

i. 

b) https://www.cathrinewilhelmsen.net/overview-azure-data-factory-components/

i. 

ii.



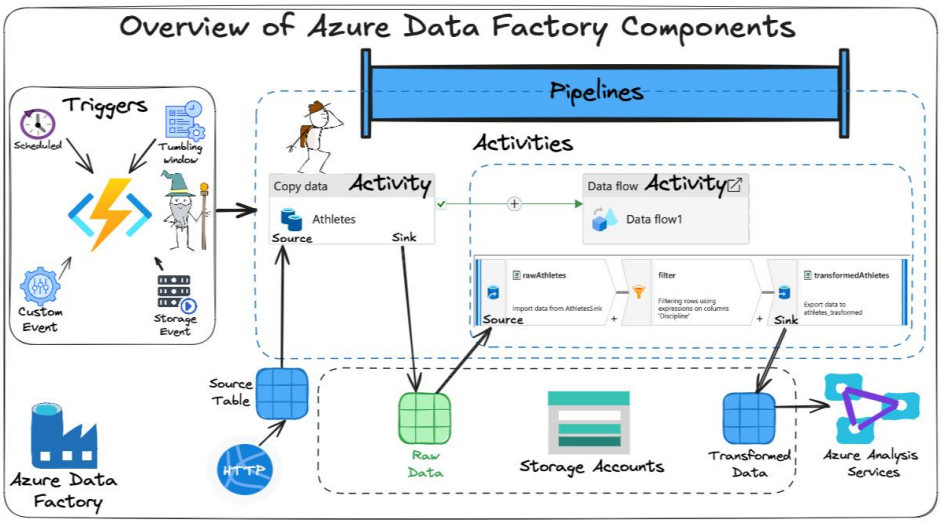
Azure Data Factory Components

ADF has key components that work together to provide the platform on which you can compose data-driven workflows with steps to move and transform data:

- **Pipelines**: A data factory can have one or more pipelines, which are a logical grouping of activities performed by a work unit. This allows activities to be managed as a set instead of managing each one individually, and they can be chained to work sequentially or independently.
- **Data flow mapping**: allows you to create and manage transformation logic graphs that can be used to transform data of any size. You can also create a reusable library of transformation routines and run those processes in a scalable manner from the service pipelines in an automated fashion.
- **Activity**: represents a processing step in a pipeline and three types are supported: data movement activities, data transformation activities, and control activities.
- **Data sets**: represent data structures within data stores that point to or reference data to be used in activities.
- **Linked services**: define the connection information required for ADF to connect to external resources. They are often used to represent a data store or represent a computer course that can host the execution of an activity.
- **Triggers**: represent the processing unit that determines when to start the execution of a process.
- **Parameters**: are defined in the pipeline and are passed during the execution created by a trigger or a manually executed pipeline. Activities within the pipeline consume parameter values or a set of data also represent a parameter.
- **Control flow**: is an orchestration of pipeline activities that includes chaining activities in a sequence, branching, defining parameters at the pipeline level, and passing arguments while invoking the pipeline or from a trigger.
- **Variables**: can be used within pipelines to store temporary values or to be used together with other parameters to allow passing values between pipelines, data flows, and other activities.

c)

d)

e) 

3. Notes on Azure Storage Account:

a) An Azure storage account contains all of your Azure Storage data objects: blobs, files, queues, and tables.

b) The storage account provides a unique namespace for your Azure Storage data that's accessible from anywhere in the world over HTTP or HTTPS.

c) Data in your storage account is durable and highly available, secure, and massively scalable.

d) Blob Storage vs Data Lake:

   i. **Azure Blob Storage** is one of the most common Azure storage types. It's an **object storage** service for workloads that need high-capacity storage.

   ii. **Azure Data Lake** is intended primarily for **big data analytics** workloads.

   iii. **Blob** -- which is shorthand for binary large object -- is ideal for large amounts of unstructured data, such as text, videos, photos, application back-end data and backup data. It's a general-purpose object store for unstructured data in a single hierarchy and a flat namespace.

      1. Common uses for Azure Blob Storage include the following:

         a) Storing files for distributed access, such as installation or upgrades.

         b) Streaming video and audio.

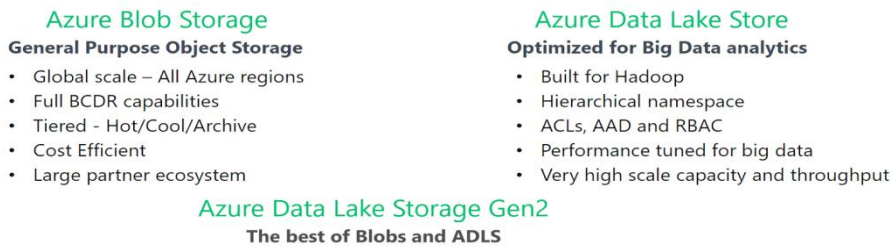
         c) Storing backups for DR and archiving.

         d) Storing binary data, such as application back-end files & general-purpose data.

   iv. **Azure Data Lake storage is currently separated into Gen1 and Gen2 options**. Microsoft will retire Data Lake Gen1 storage in February 2024, and all customers using it must migrate to Gen2 before this date.

1. Azure Data Lake Gen1 is a storage service that's optimized for big data analytics workloads. Its hierarchical file system can store machine learning data, including log files, as well as interactive streaming analytics. It is performance-tuned to run large-scale analytics systems that require massive throughput and bandwidth to query and analyze large amounts of data.
2. Azure Data Lake Gen2 converges the features and capabilities of Data Lake Gen1 with Blob Storage. It inherits the file system semantics, file-level security and scaling features of Gen1 and builds them on Blob Storage. This results in a low-cost, tiered-access, high-security and high availability big data storage option.

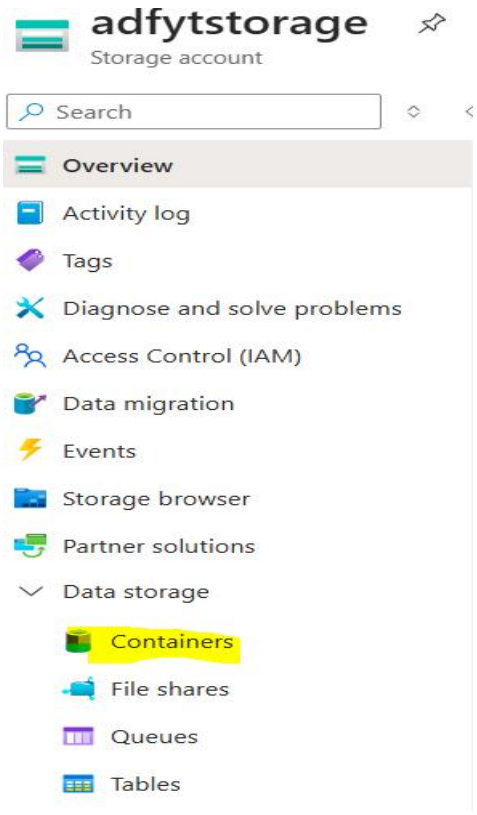
v.



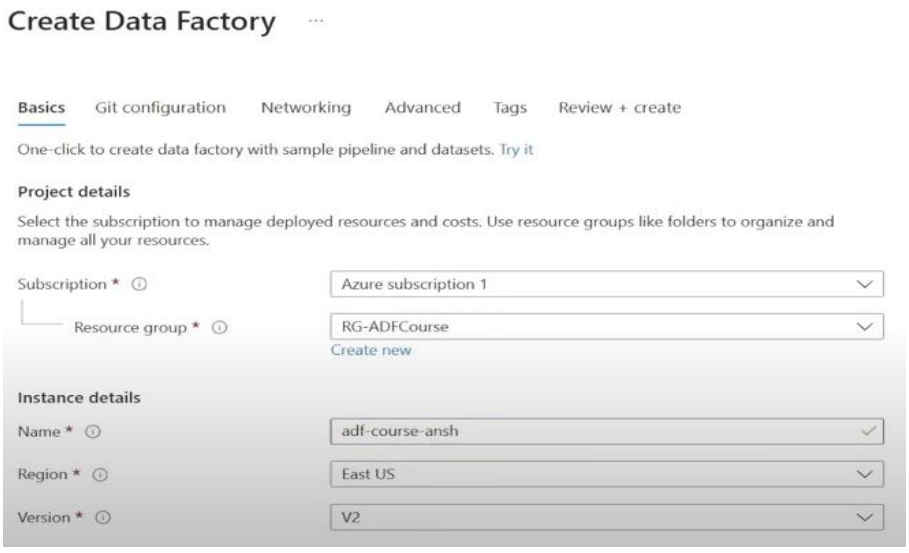
vi.

	Data Lake	Blob Storage	Data Lake Gen 2
Hot/Cold Storage Tiers	NO	YES	YES
Redundant Storage	NO	YES	YES
AD Security	YES	NO	YES
HDFS Compatible	YES	NO	YES

4. Create a ADLS gen2 storage
   a) Enable Hierarchical namespace*** >> “Containers” are Data lakes.

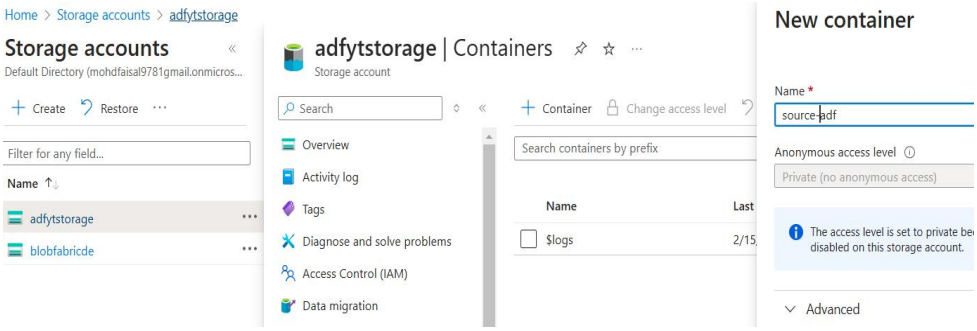
b) 

5. Create ADF

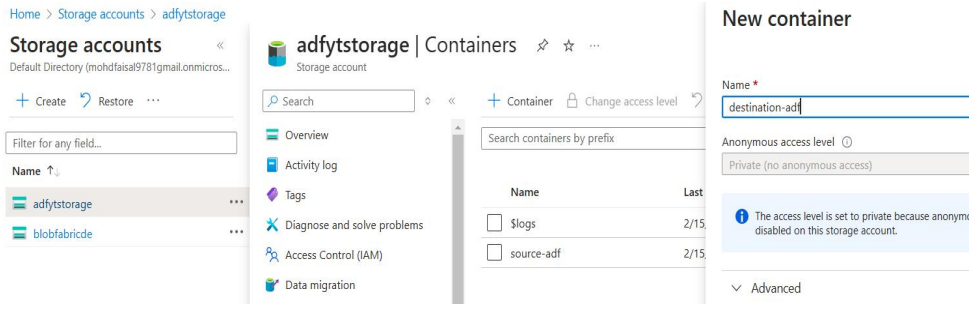
a) 

6. Create new Container(s) in Data Lake storage account

a)

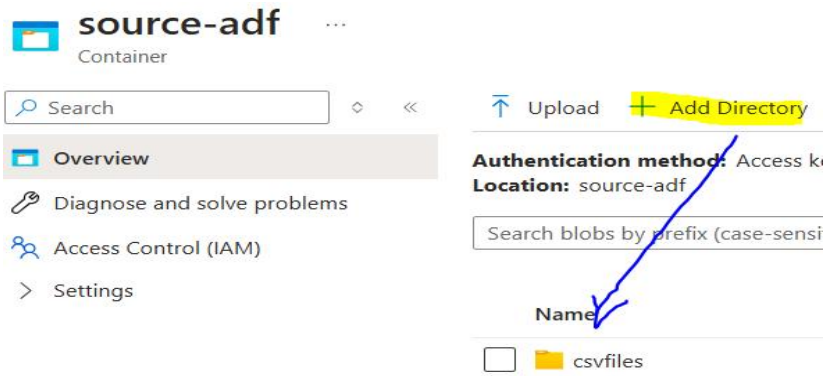


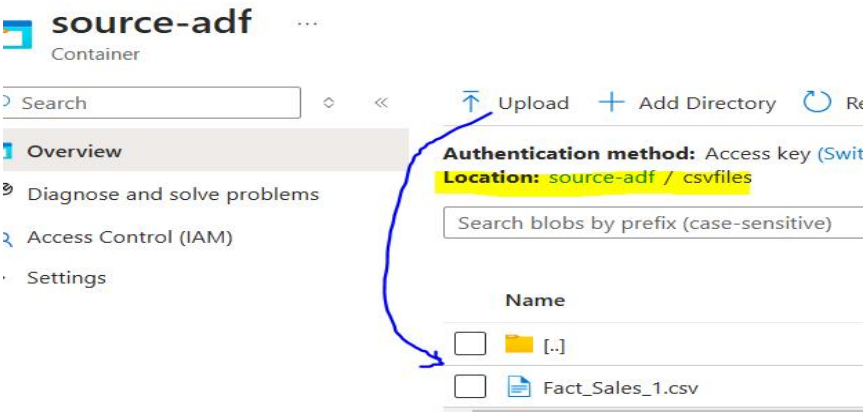
b)



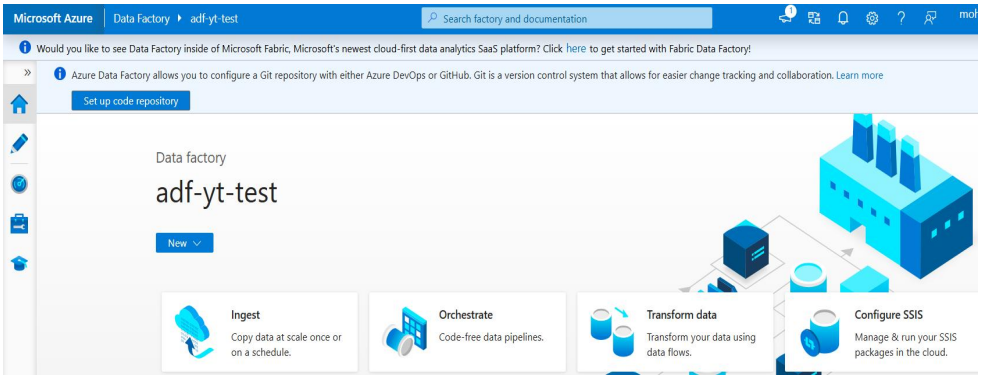
7. Upload source file 'fact_sales_1.csv' to 'csvfiles' directory of source container

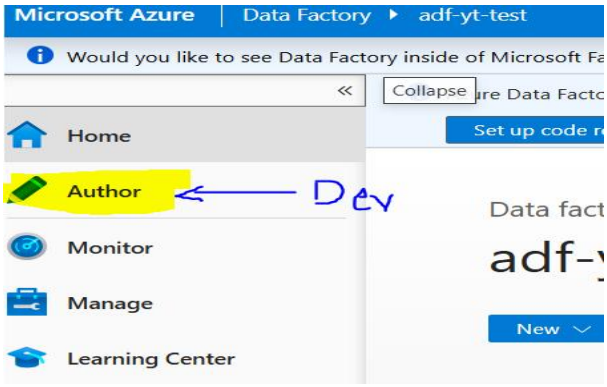
a)



b) 

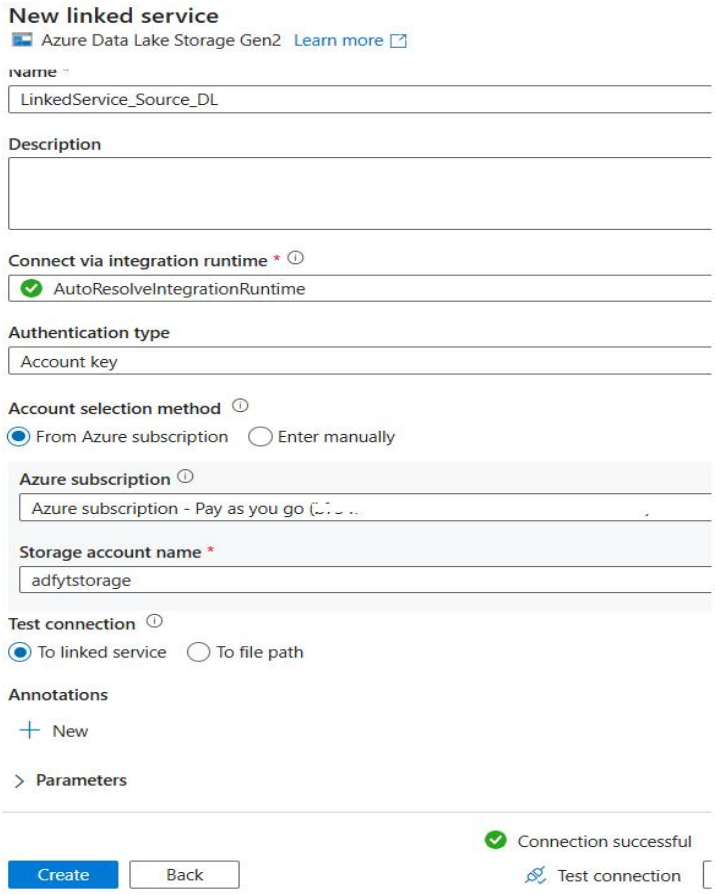
8. ADF > Launch studio from resource we created in step-5

a) 

b) 

c) Create a connection to source

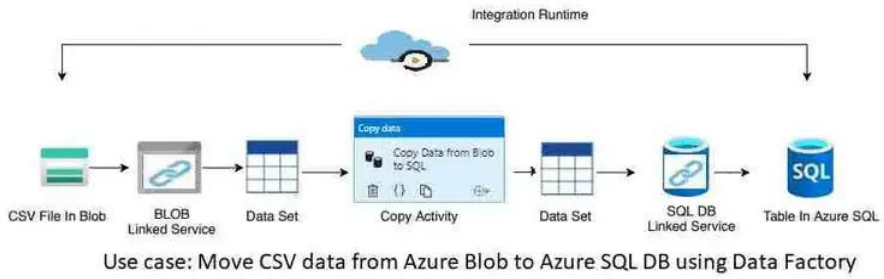
i. ADF > Manage > Linked Service > New >

1. 

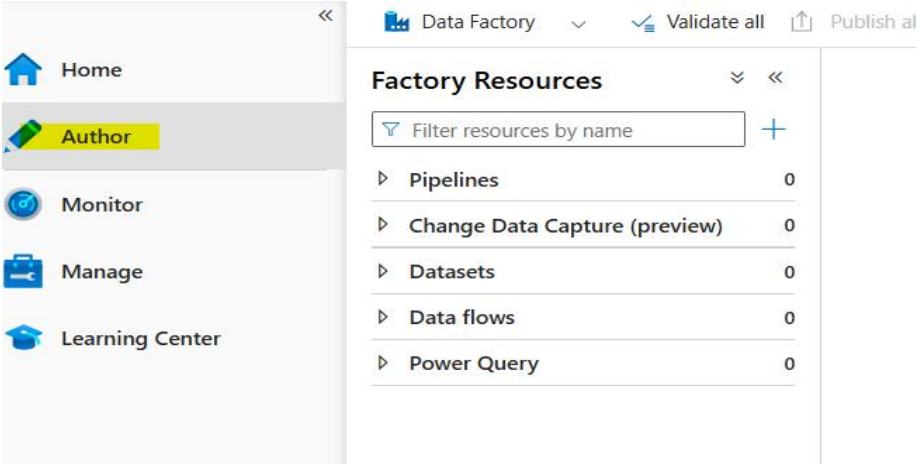
d) COPY ACTIVITY:

i. Copy activity is basically used for ETL purpose or lift and shift where you want to move the data from one data source to the other data source. While you copy the data you can also do the transformation.

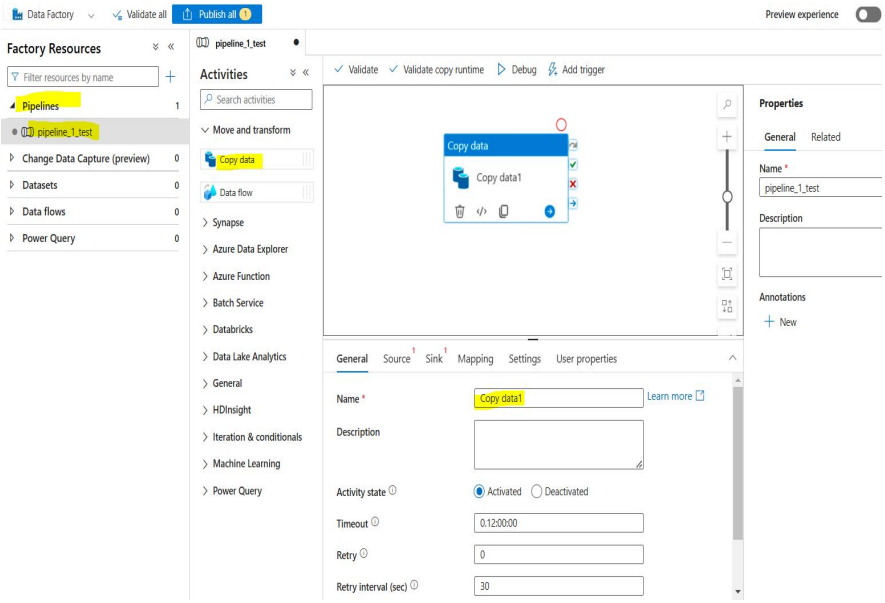
ii. https://azurelib.com/azure-data-factory-copy-activity/

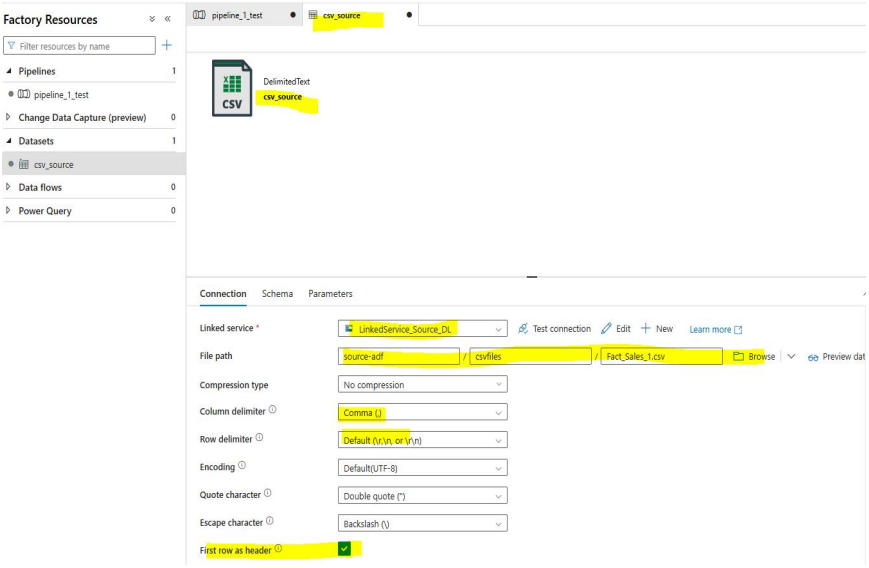
iii. 

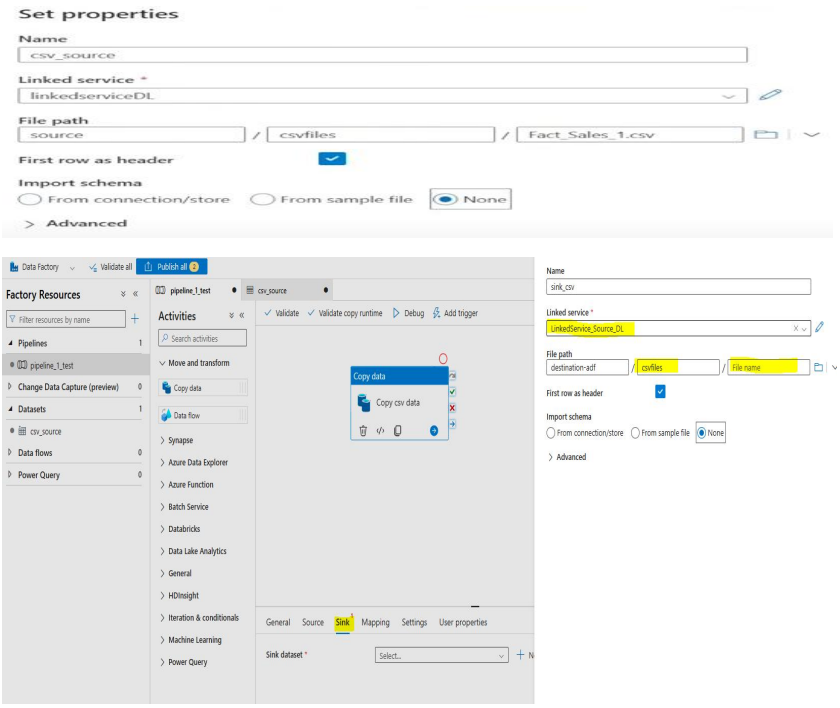
iv. Go to Author >



v. Create Pipeline > Add Copy Activity > Specify source > Sink >

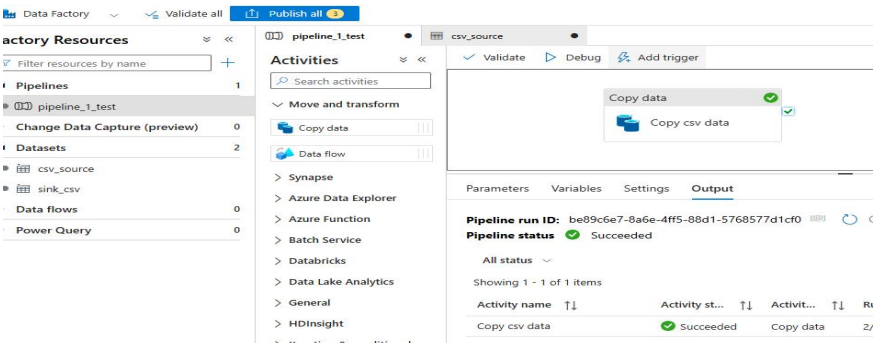
1. 

2. 

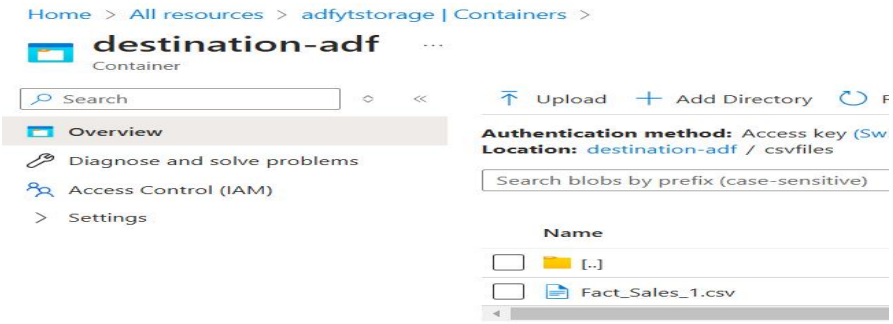
3. 

Note: Here we are using same Linked service as target is ADLS G2.. creating new directory in sink config… and file name will be set be ADF..

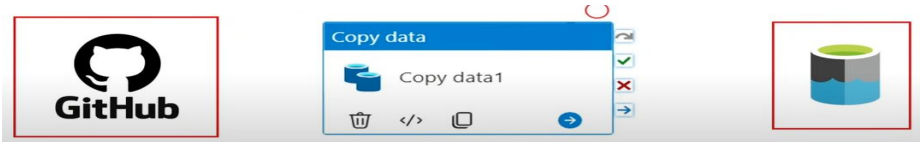
4. Debug the pipeline



5. Validate in destination container



e) COPY DATA using REST API:

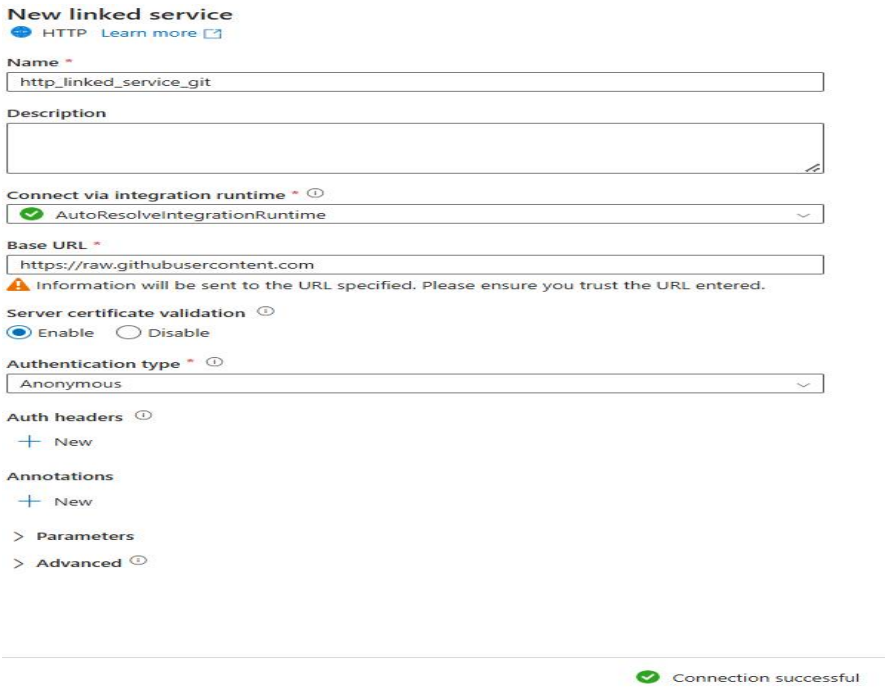
i. 

ii. Create new pipeline

iii. Create new Copy Activity

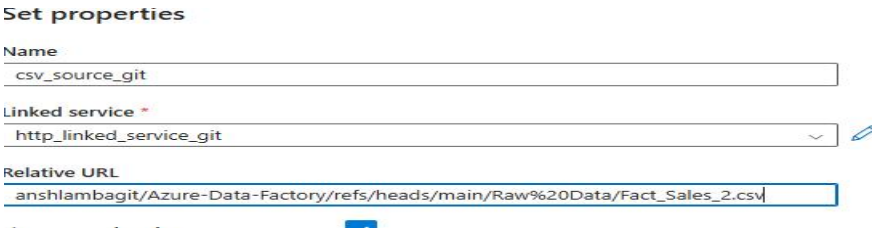
1. Source: HTTP > New Linked Service

   Note: For Linked Service, base url is parsed "Raw" button url of file on git hub

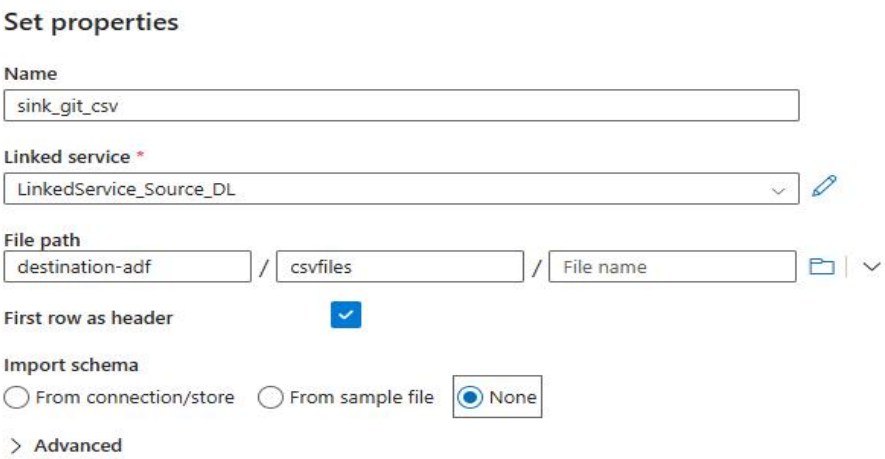


   Then in the source properties, it'll ask the Relative URL, get same from parsed

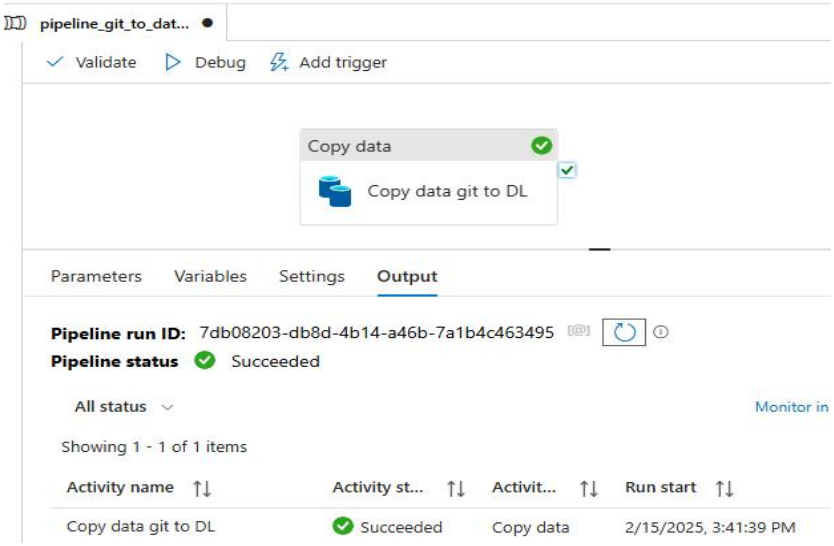
“Raw” button url of file on git hub



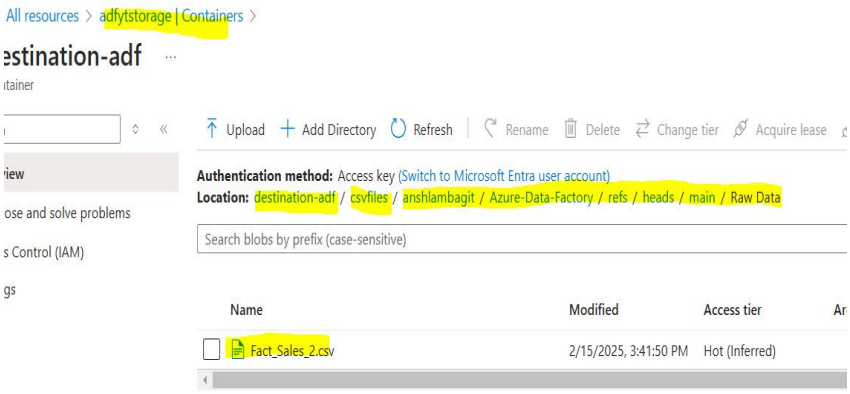
2. Sink >



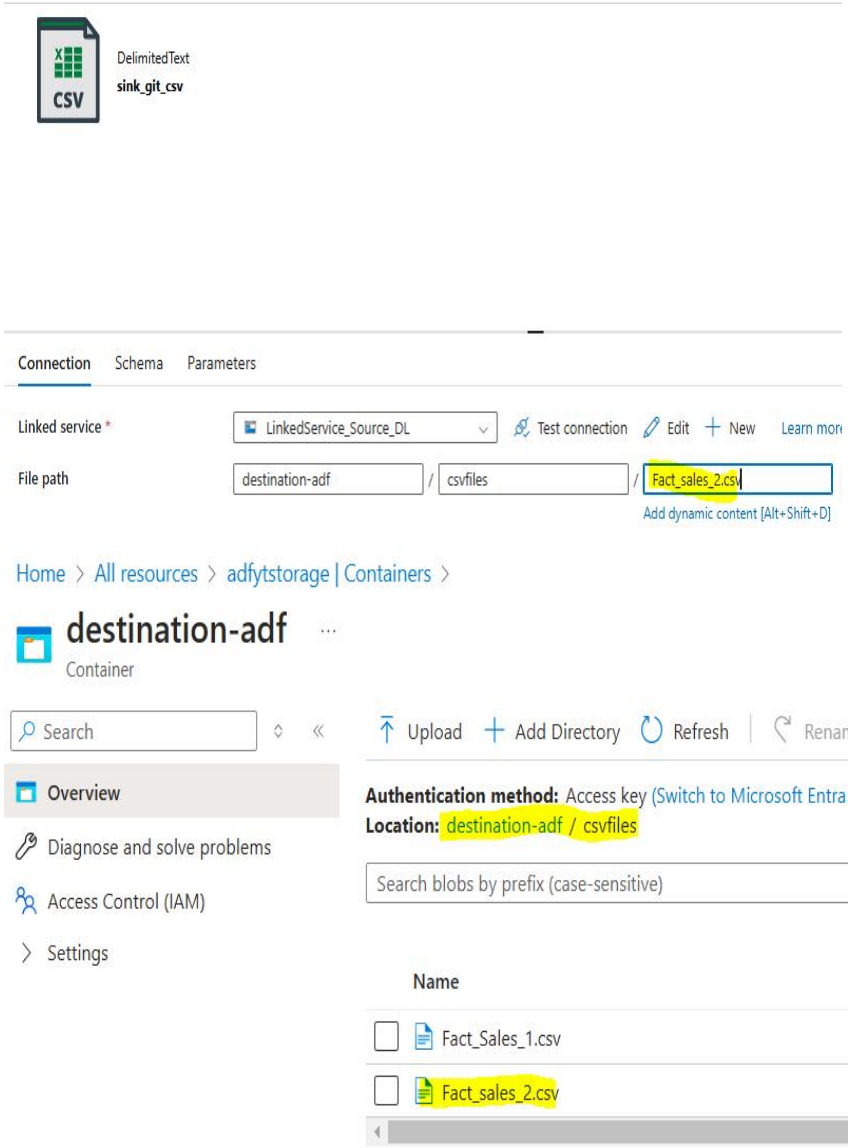
iv. Debug & Validate

1. 

2. Note: As we didn't specify the file name in Sink properties, ADF has taken full path from git-hub



3. To fix this File path, edit sink properties and re-run & validate



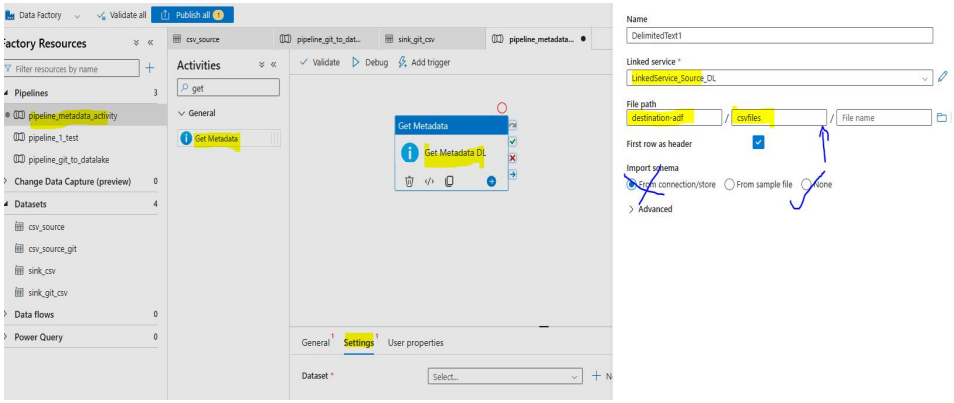
f) Get Metadata Activity:

i. Suppose we have a data governance policy where only 'Fact_sales_1.csv' (out of multiple files in source folder) has to be be ingested / copied to destination.

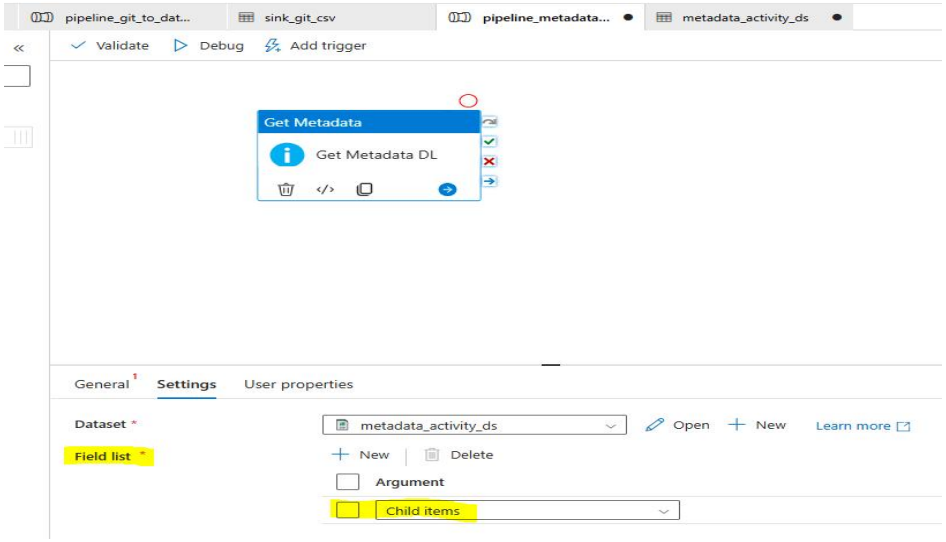
ii. While working in Azure Data Factory, sometimes we need to retrieve metadata information, like the file name, file size, file existence, etc. We can use the Get Metadata activity to retrieve metadata information from the data set and then we can use that metadata information in subsequent activities. Refer https://www.sqlservercentral.com/articles/working-with-get-metadata-activity-in-azure-data-factory for more details.

iii. https://www.mssqltips.com/sqlservertip/6246/azure-data-factory-get-metadata-example/

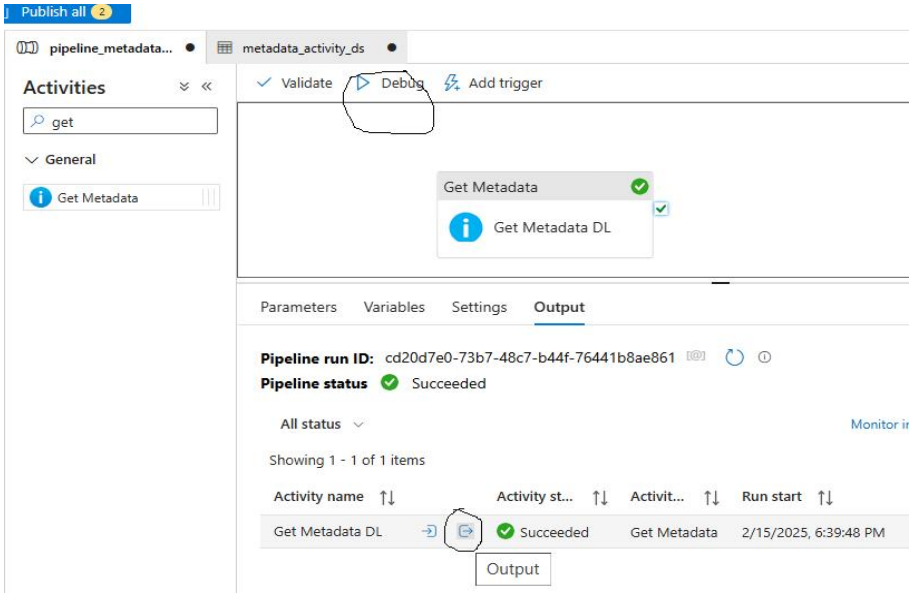
iv. New Pipeline > Add 'Get Metadata' activity > Set properties > Debug >> This will give array of files in folder..



Note: We need metadata of folder.. so file name is not given..



Below is snip of successful run > see output >



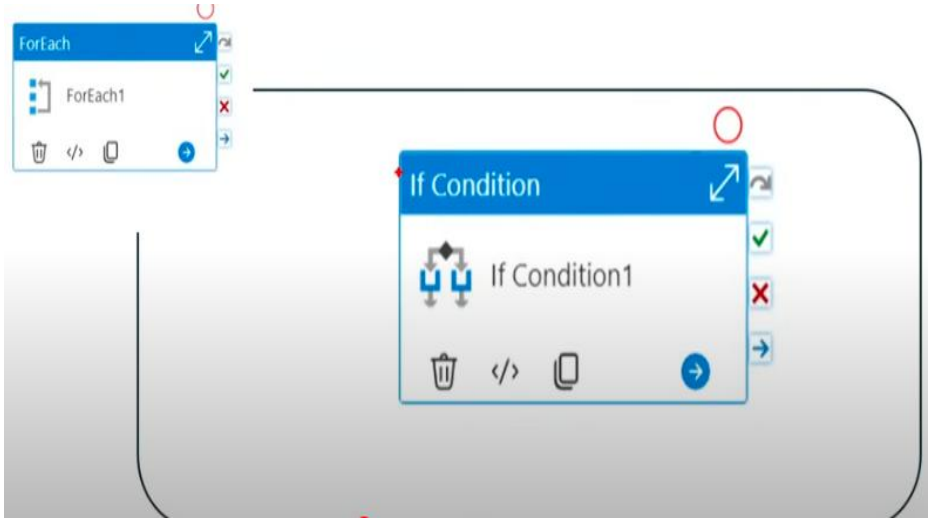
This'll be the output >

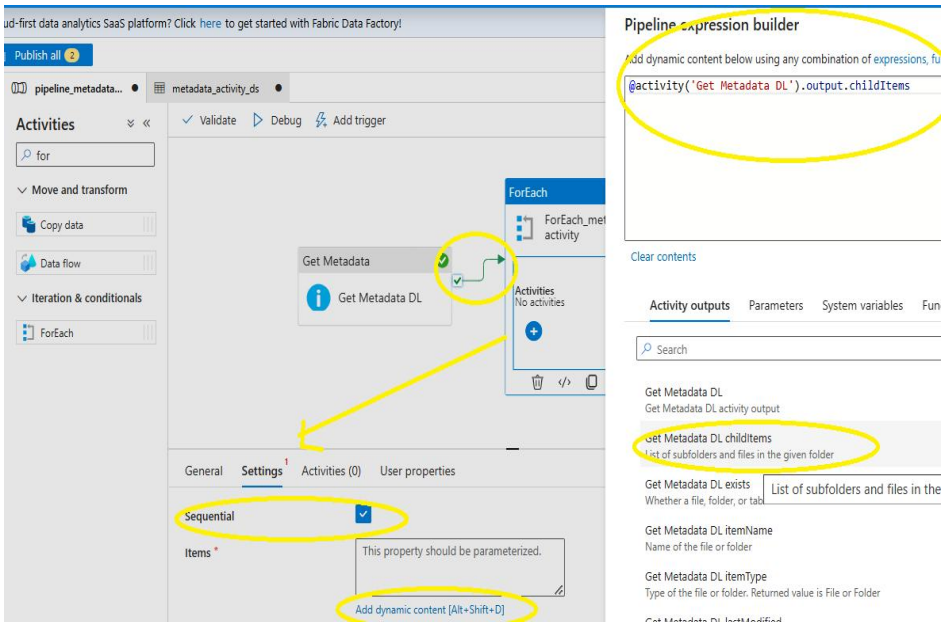
Output

Copy to clipboard

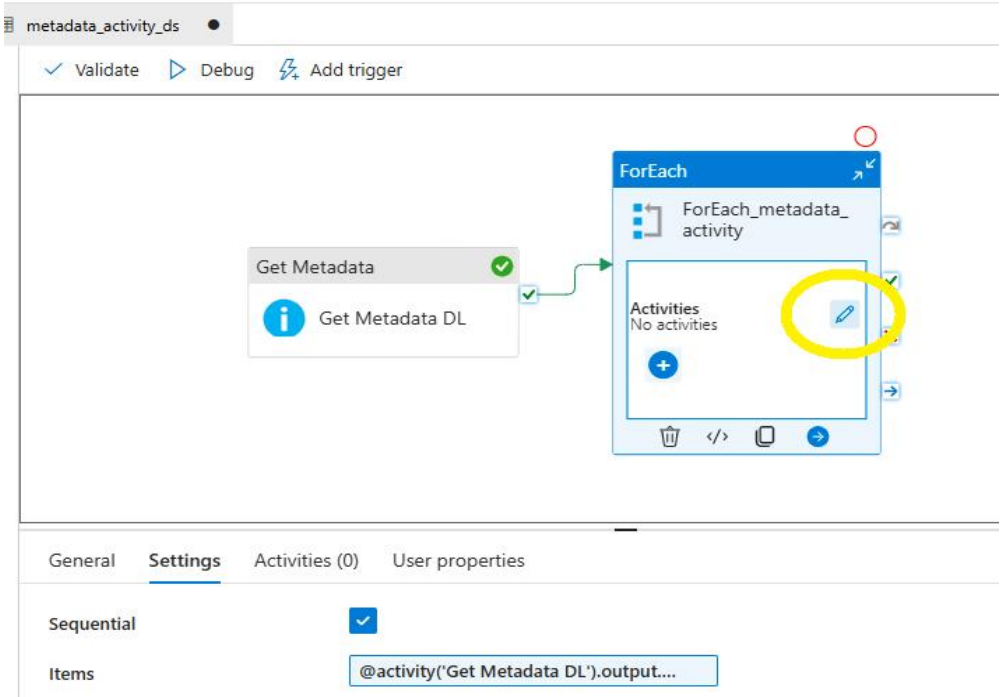

```
{
    "childItems": [
        {
            "name": "Fact_Sales_1.csv",
            "type": "File"
        },
        {
            "name": "Fact_sales_2.csv",
            "type": "File"
        }
    ],
    "effectiveIntegrationRuntime": "AutoResolveIntegrationRuntime (Canada Central)",
    "executionDuration": 2,
    "durationInQueue": {
        "integrationRuntimeQueue": 0
    },
    "billingReference": {
        "activityType": "PipelineActivity",
        "billableDuration": [
            {
                "meterType": "AzureIR",
                "duration": 0.016666666666666666,
                "unit": "Hours"
            }
        ]
    }
}
```


v. ForEach file (using **ForEach activity** as we have array of filenames above) > Now give the filename to **IF activity** >> i.e if the file_name matches > load to target



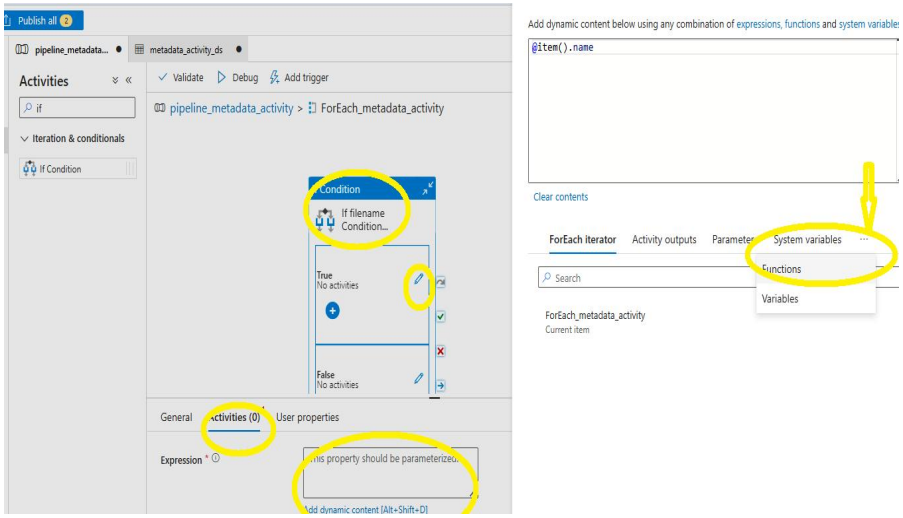


Then go Inside the FOREACH >>

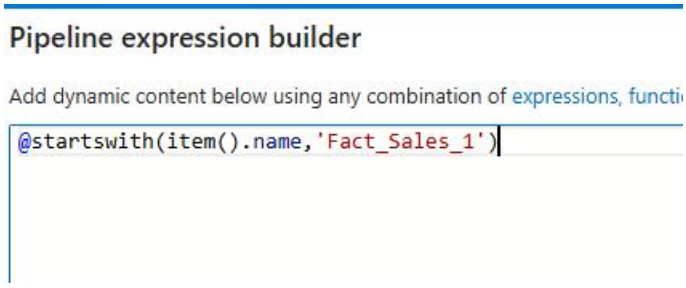


Add IF Activity >>

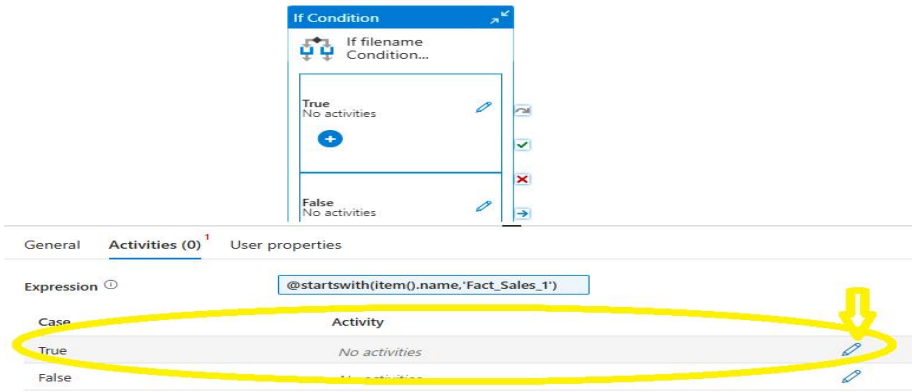
Syntax we refer as per output above



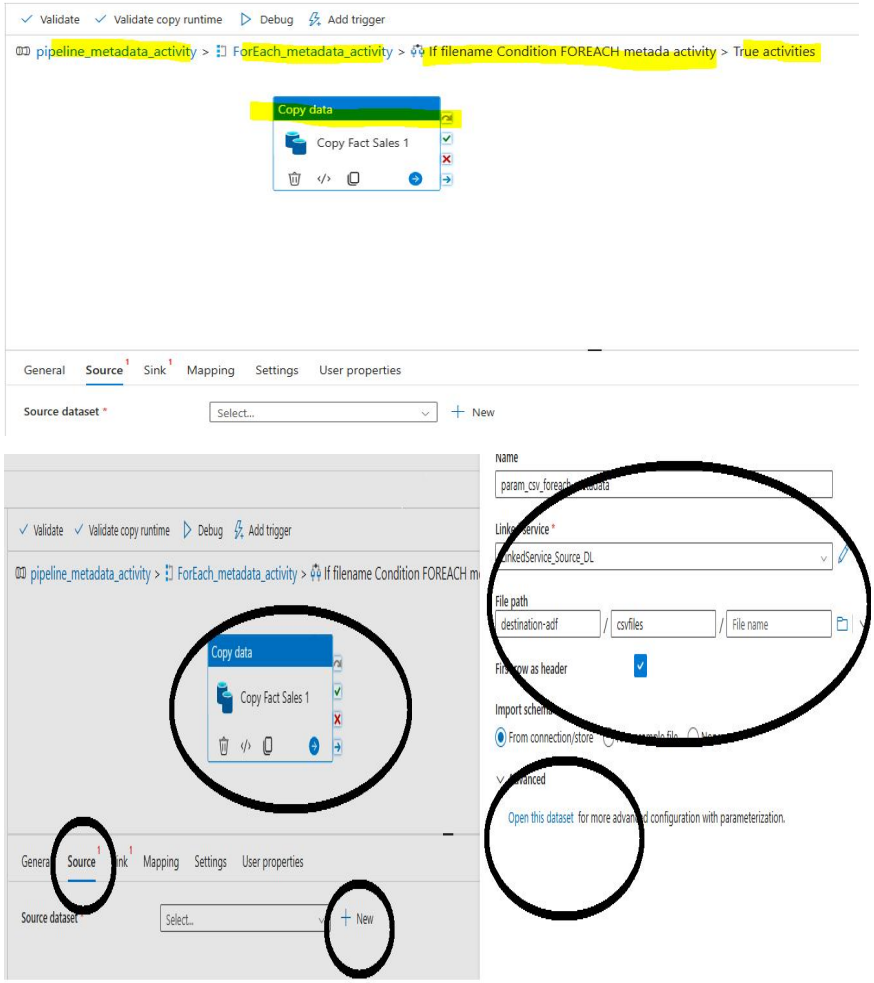
>



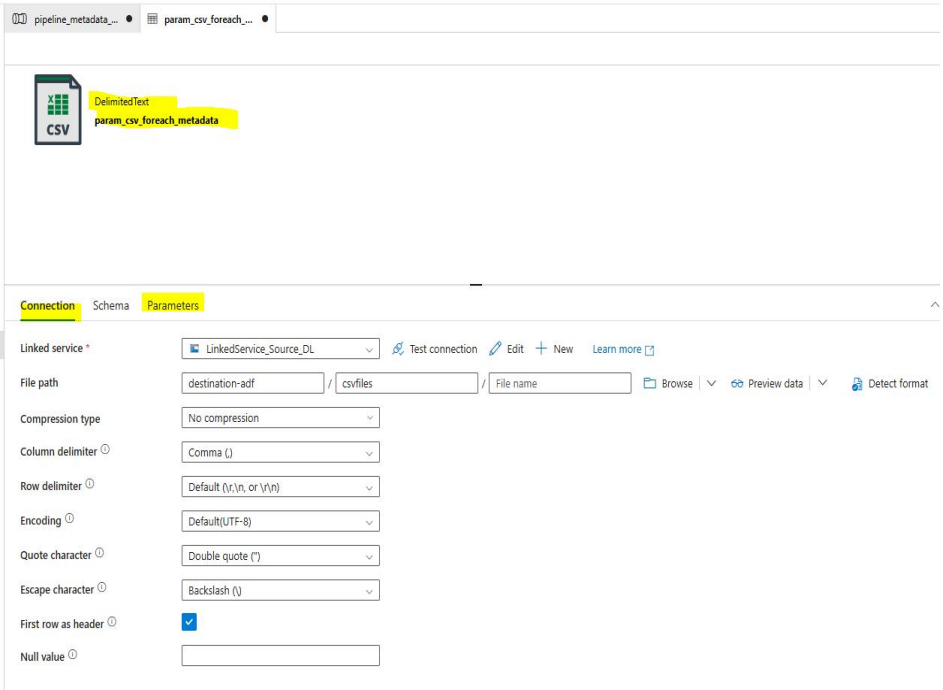
> If above expression is TRUE > Copy output file > Edit TRUE condition >



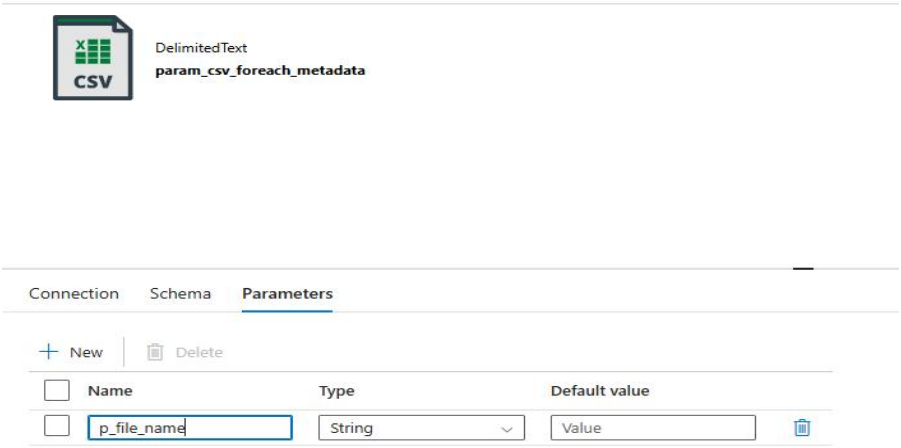
>> Edit TRUE condition > Add Copy Activity >> Here we'll add PARAMs as File name >>



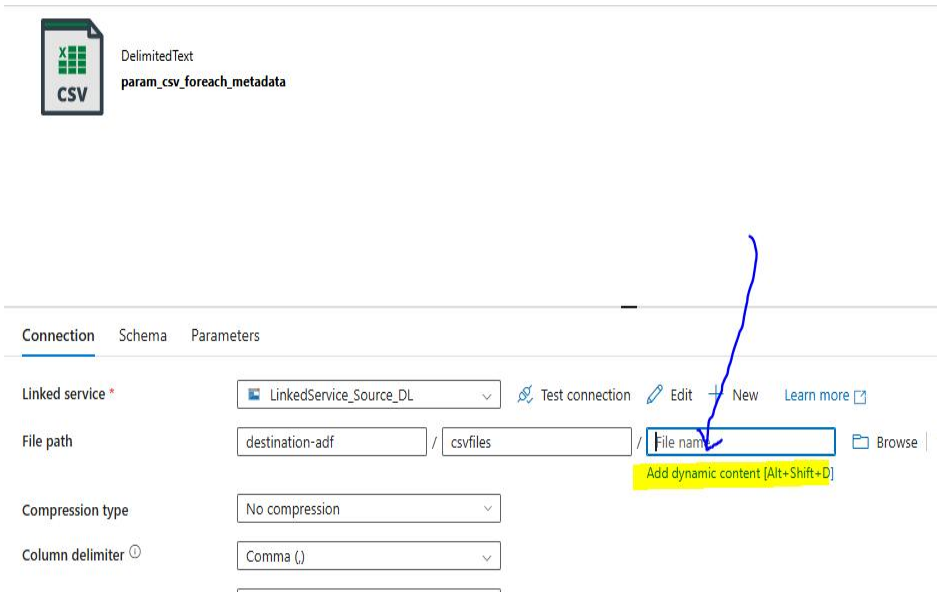
>> On selecting “Open the dataset” from Advance > we get >>



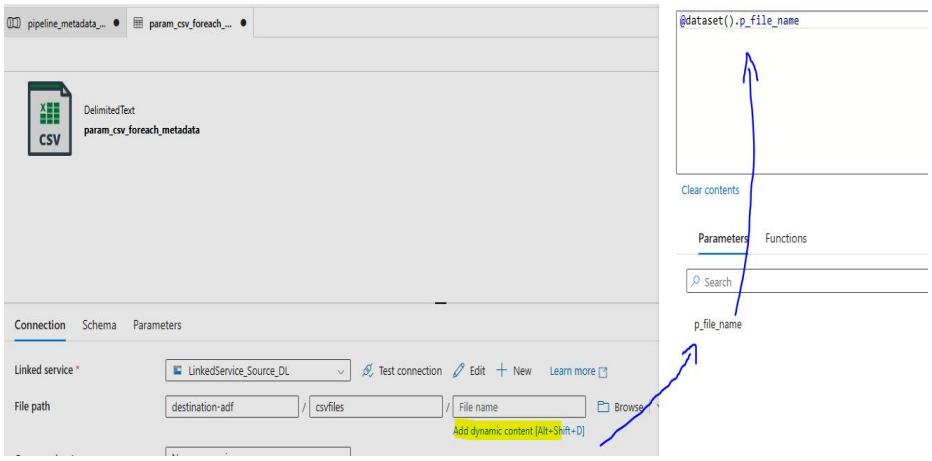
>> Create parameter >



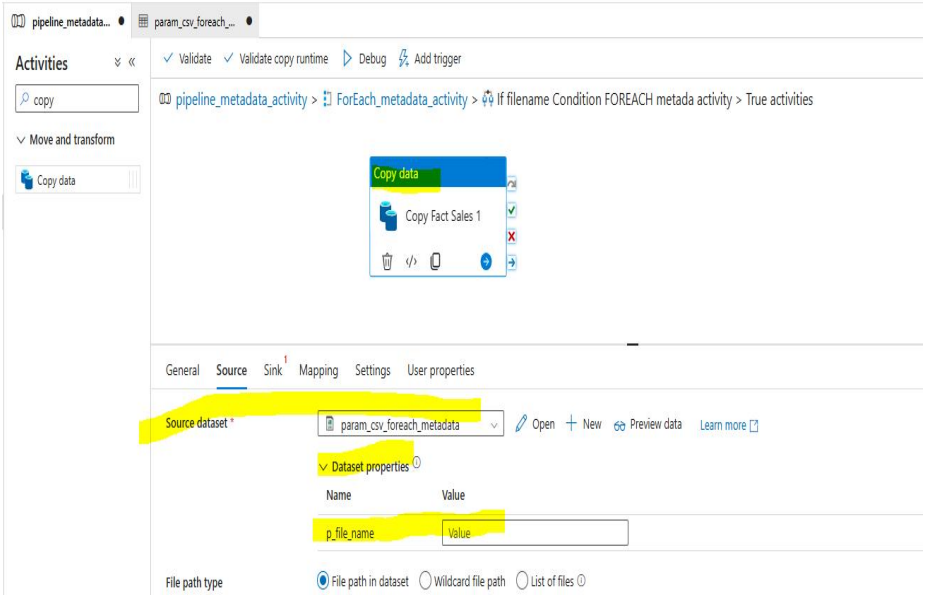
> Use the parameter >



>

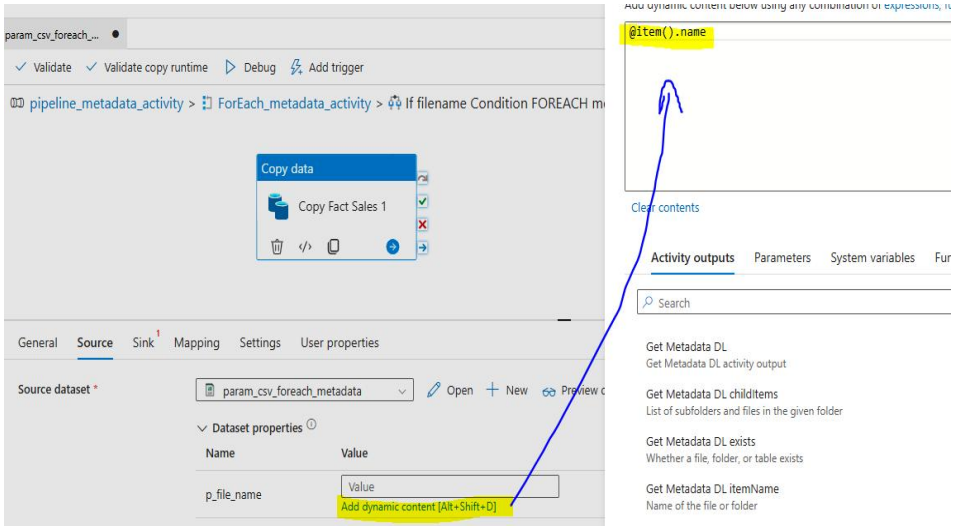


>> Now Go to Copy Activity >



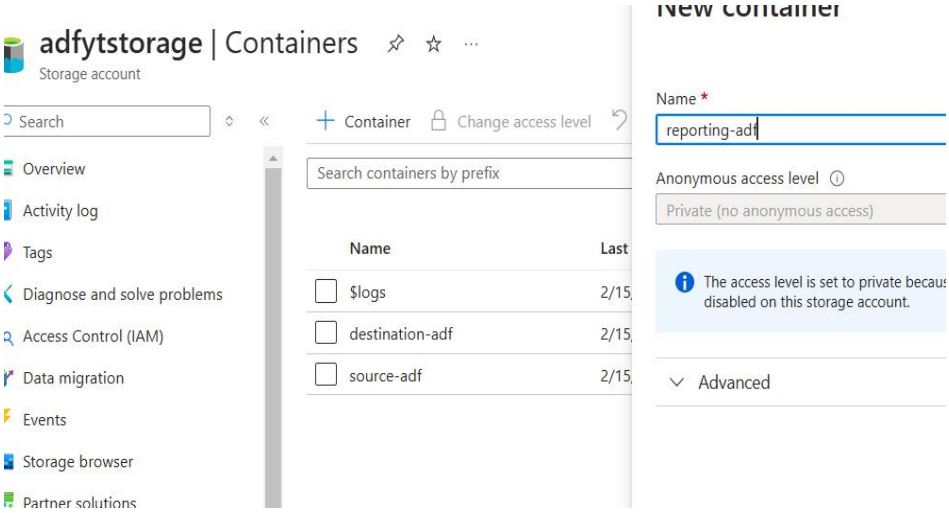
>> Here we pass the value to Parameter >>

This Filename is coming from ForEach , i.e. >> Here it'll olny COPY the file whose condition is satisfied in each iteration of ForEach >>

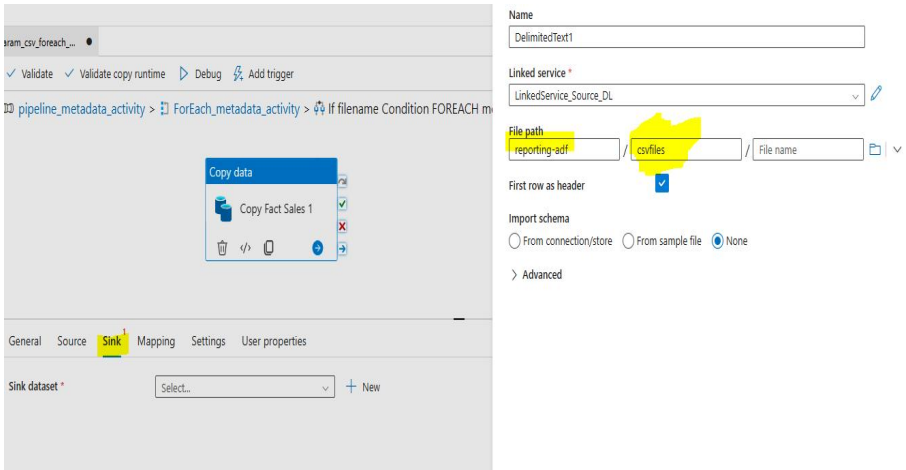


>>

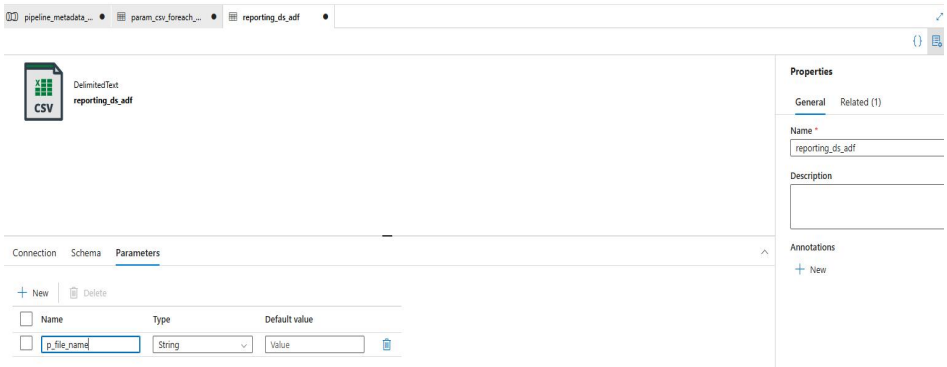
Create New container 'reporing-adf' > This will be used as sink >



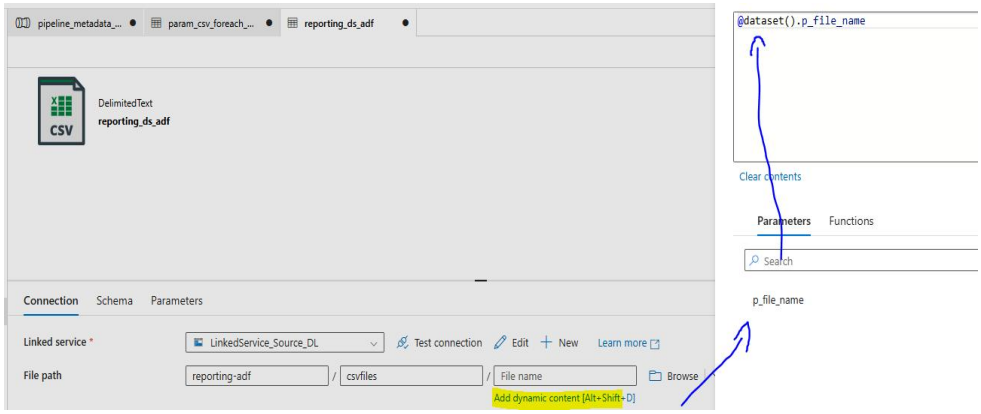
>>



>> Add PARAM to this sink dataset >

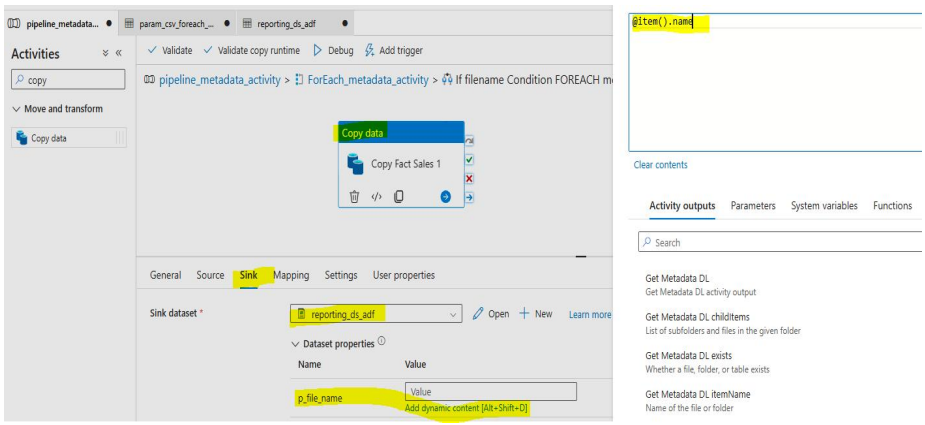


>>



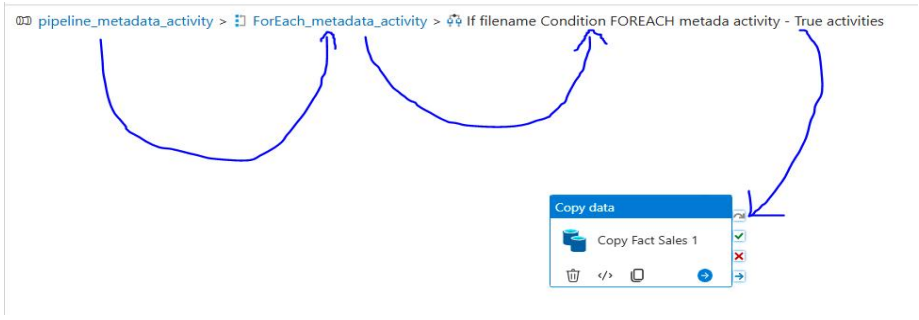
>>

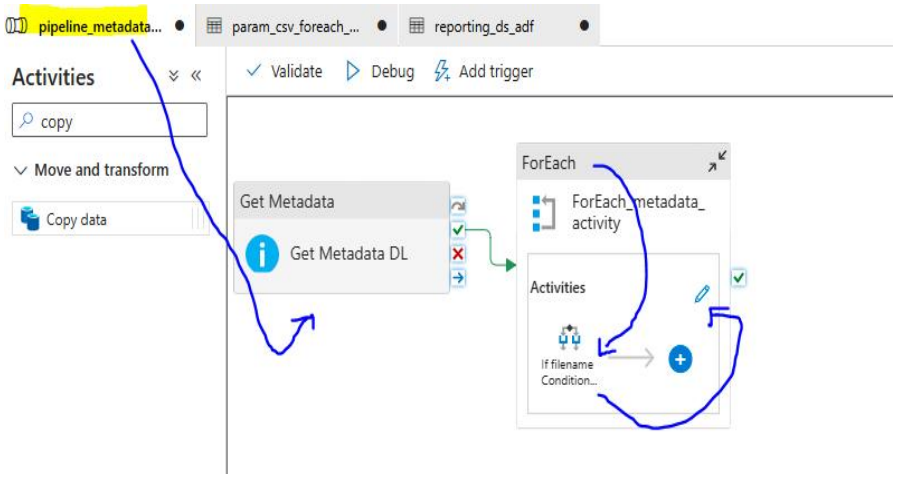
Use that PARAM in COPY sink properties >



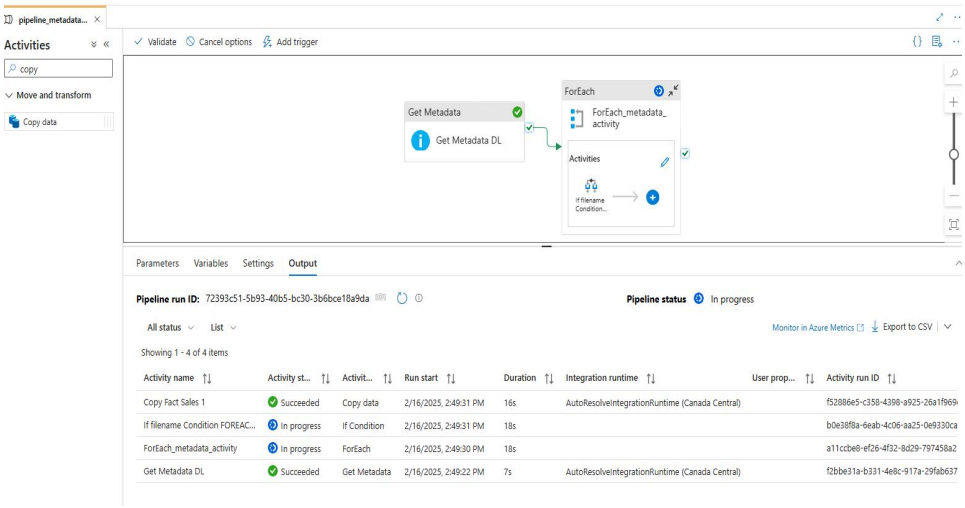
>>

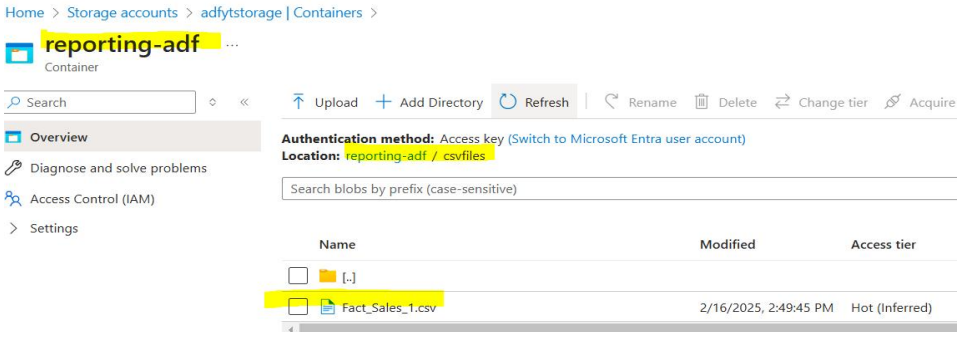
NOW THIS IS THE END to END view of this exercise >>





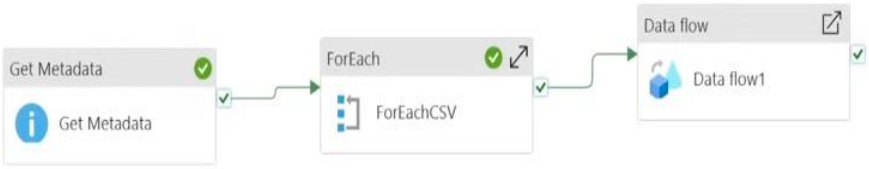
>>> DEBUG & VALIDATE >> ONLY FACT_SALES_1 file is copied!!!





g) DATAFLOWs in ADF:

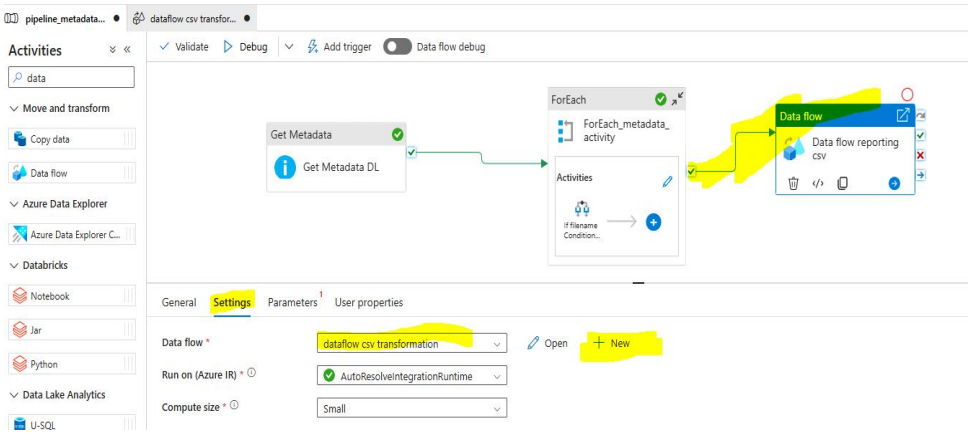
i. Once we have the data in reporting container, what if we want to **TRSANSFORM** the data??
ii. Then we will use **DataFlows (Refer tutorial on git-hub)**
iii. https://www.sqlshack.com/data-flow-transformations-in-azure-data-factory/
iv. Alter Row Transformation is used for UPSERTs in data flow.



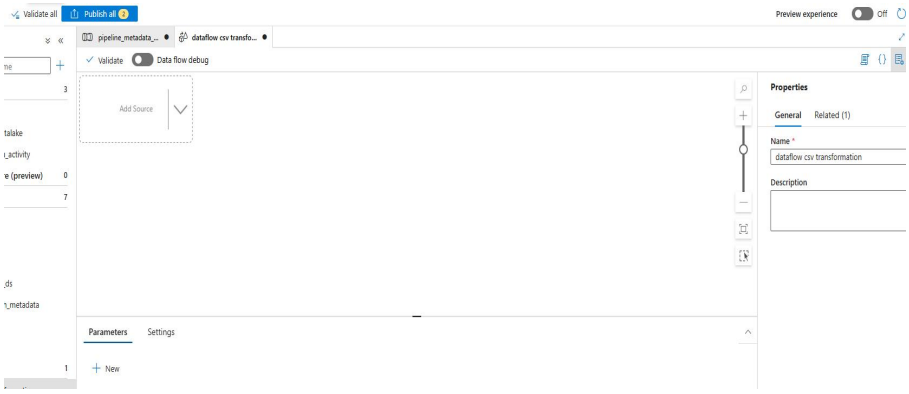
v.

vi. **DATAFLOW uses SPARK** for transformations.

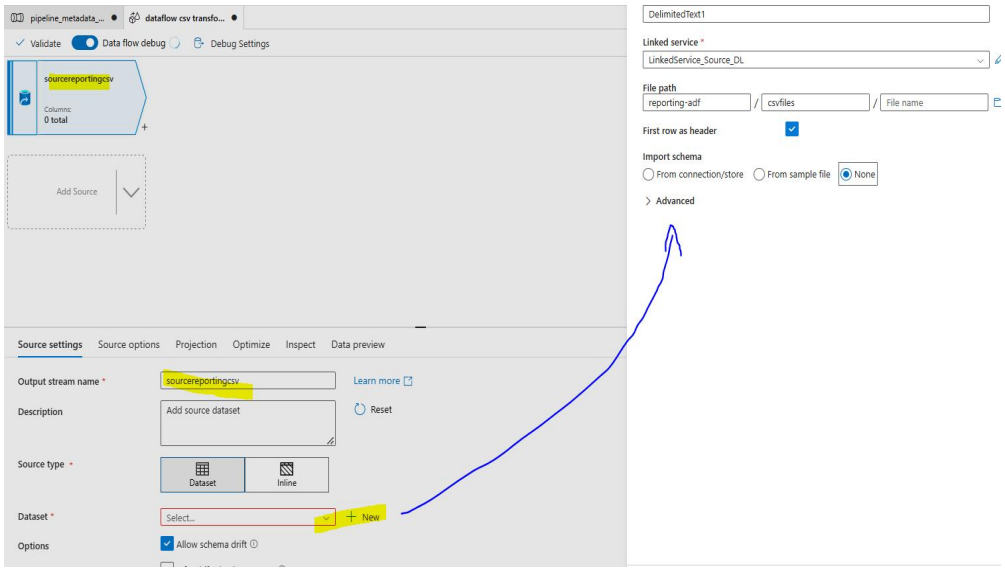
vii. Now add dataflow > add source > turn on Data Flow Debug (spark comoute and Time to Live) >



>>>>

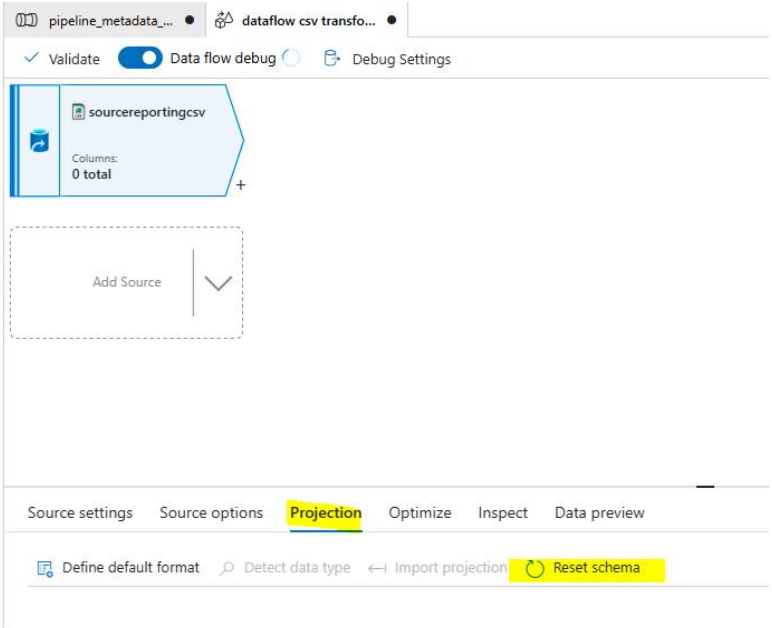


>>

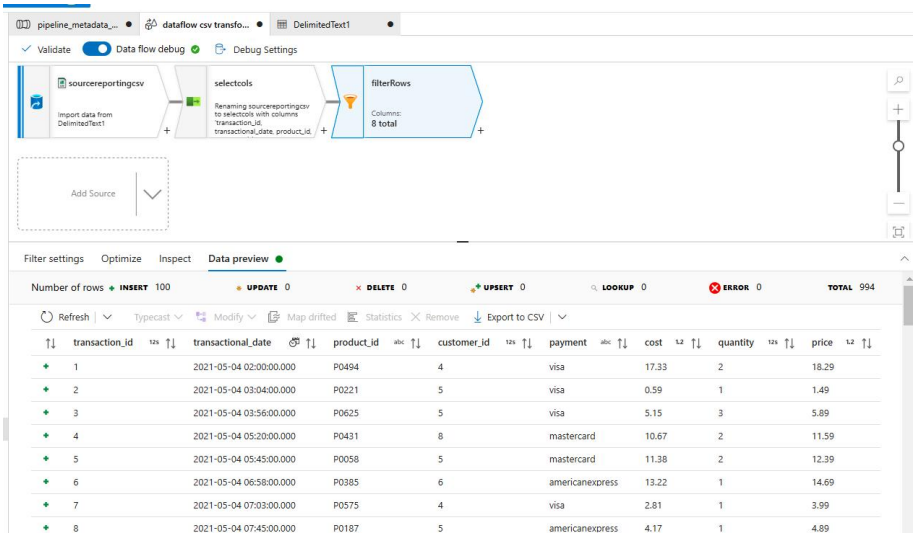


>>

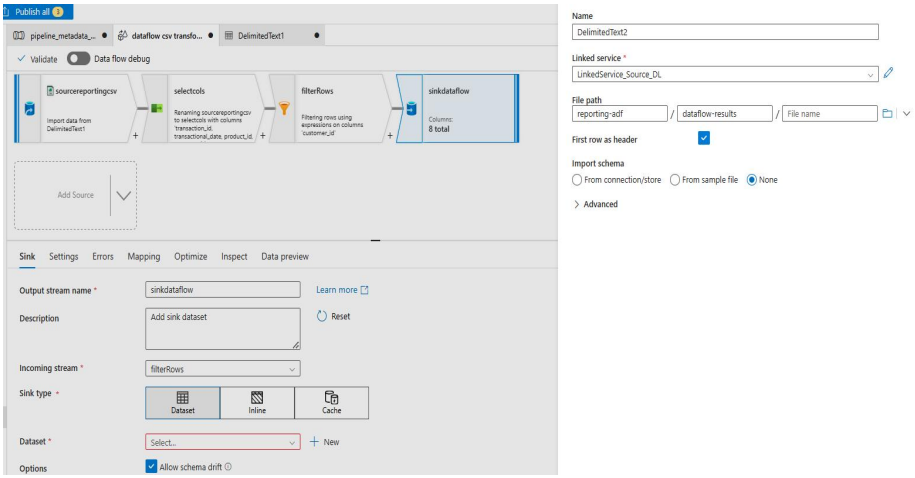
>> Now IMPORT PROJECTION (i.e. Schema) >



>>> Add some tranformations (SELECT & FILTER for example) >>



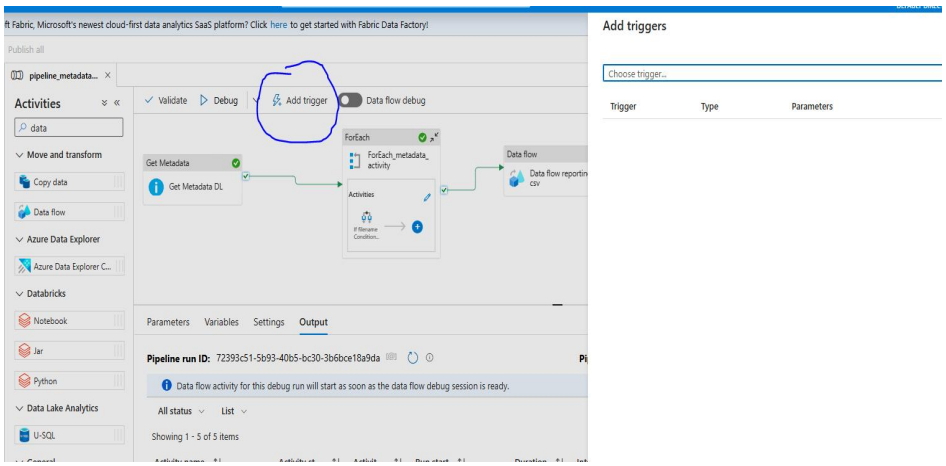
>> Add sink to data flow >>

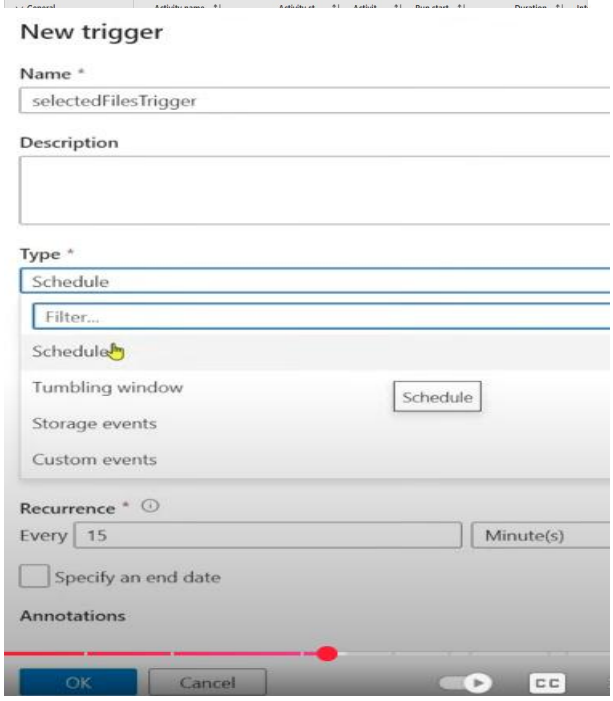


Note: The format of output file is shown in Triggers below.

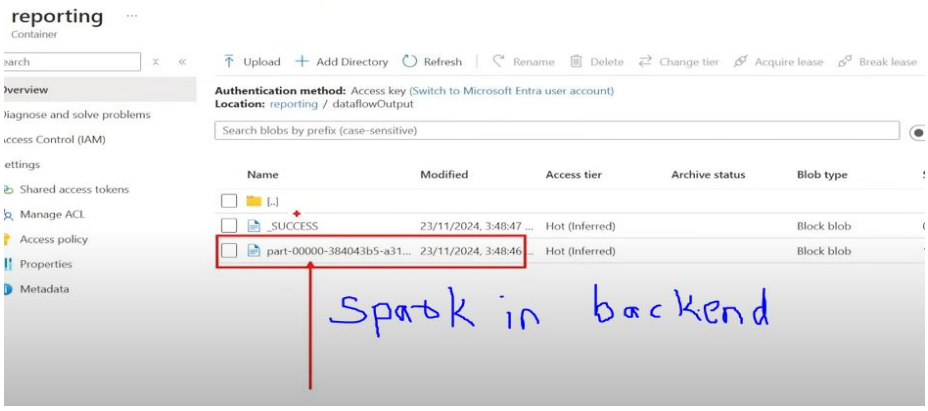
h) Schedule Trigger in ADF:

i. https://aspinfo.medium.com/what-are-the-triggers-in-adf-14ec208fdbdc
ii. https://medium.com/@hasninemirza/azure-data-factory-triggers-cdb193142c9e

iii. 

iv. 

v. When Trigger is executed > Go to Monitor > Also validate >



i) Set Variable Activity:

i. "Set variable" Activity is used to set the value of an existing variable of type String, Bool, or, Array defined in a Data Factory Pipeline.
ii. https://medium.com/codex/introduction-to-set-variable-activity-73bcddcf0318
iii. **Set Variable vs Pipeline Variable**: https://medium.com/@rganesh0203/set-variable-vs-pipeline-variable-in-adf-21bf2eaec44b
iv. **Set Pipeline Return Value:**
    1. pass values between two ADF pipelines
    2. https://www.techbrothersit.com/2023/05/set-pipeline-return-value-in-azure-data.html

j) Storage Events Trigger:

i. Similar to event driven in AWS S3
ii. https://pragmaticworks.com/blog/azure-data-factory-event-triggers
iii. https://www.skynorthsoftware.com/blog/posts/copy-blob-event-trigger/

k) Execute Pipeline Activity:

i. to call a pipeline from another pipeline or to pass parameter from one pipeline to another pipeline.
ii. https://azurelib.com/execute-pipeline-activity-in-adf/

What could be the business use cases or scenarios where you need to call a pipeline from another pipeline

- You wanted to chain two different pipeline flow together. For example one pipeline is pulling the master data and other pipeline is pulling the entity data and you want to run both of them in sequence.
- You wanted to split the existing pipeline into two or more pipeline, because keeping all the logic in one pipeline making it very big to handle and maintain.
- There could be scenario where you wanted to do the nesting of foreach activity, but unfortunately it isn't allowed in the Azure data factory to have nested foreach activity. Hence to solve this problem what you can do is you split the pipeline into two. Both the pipeline keeps one foreach activity. Now call the second pipeline from the first pipeline using the execute pipeline activity.

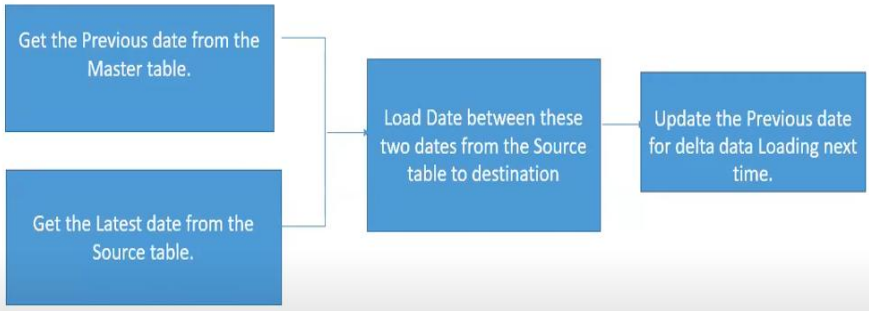
iii.

l) Real time- Incremental Load:

i. https://www.youtube.com/watch?v=z5frQ3RyFmY

ii.

Incremental Load Flow



iii.

Pipeline Flow

Create two Lookup activities.

- Use the first Lookup activity to retrieve the Previous date.
- Use the second Lookup activity to retrieve the Latest Date. These Date values are passed to the Copy activity in the Flow.

>

Create a Copy activity that copies rows from the source data store with the value of the Lastmodify column greater than the Previous date value and less than the Latest Date value. Then, it copies the delta data from the source data store to Blob storage as a new file.

>

Create a StoredProcedure activity that updates the Previous date value for the pipeline that runs next time.

>