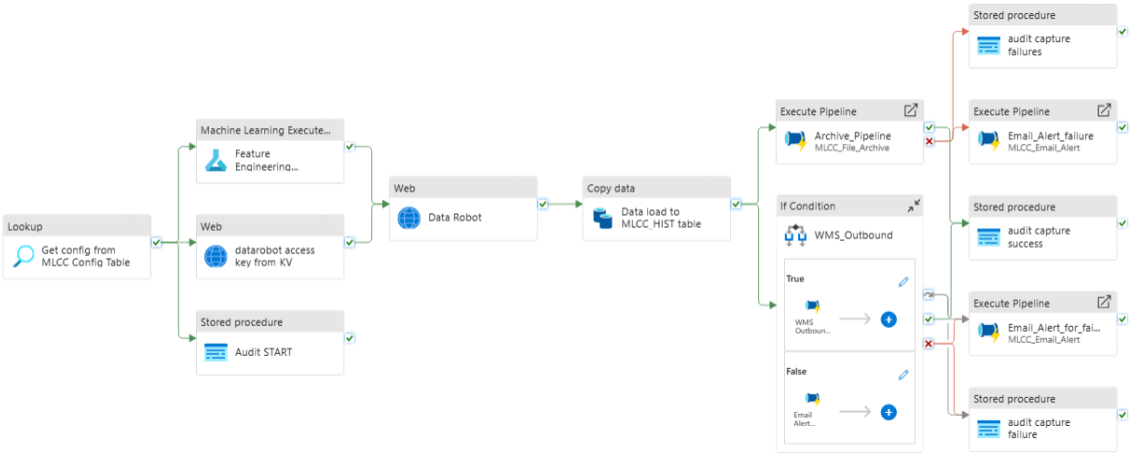
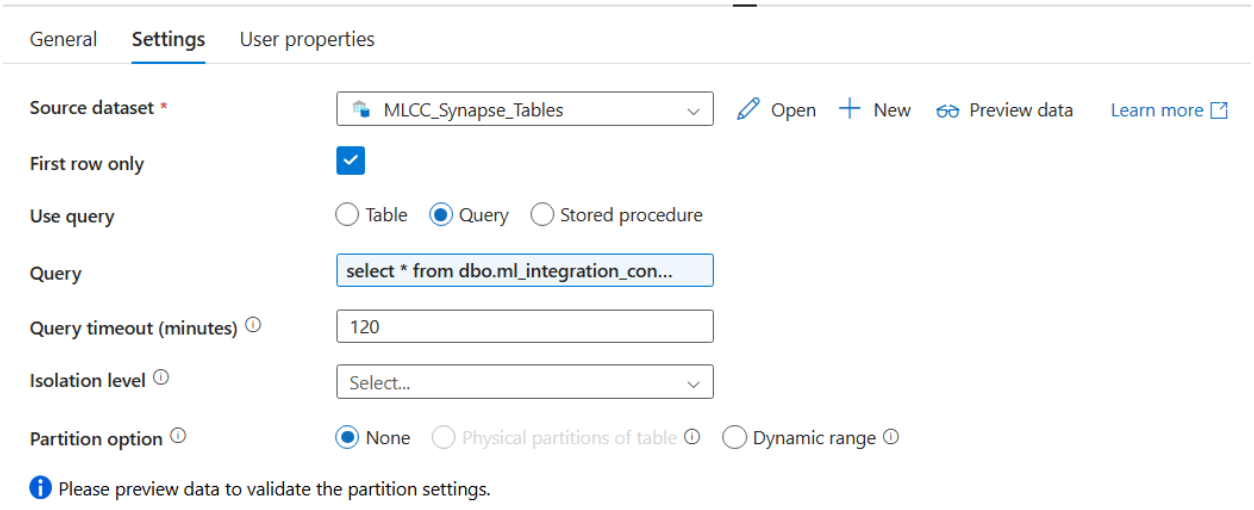
--MLCC ADF pipeline



As a first activity in this pipeline, it checks if a row of data with the given process identifier exists on the ml_config table. Then feature engineering notebook(ML studio pipeline) is triggered and alongside Datarobot key is accessed from key vault. When both these activities completed datarobot job definition is triggered for prediction using web activity in ADF. Also, the audit activity creates a record of the pipeline start time, process identifier etc. Once the datarobot prediction is complete the copy activity loads data to mlcc hist table.

Activity 1 - Get config from MLCC Config Table



For this we need to create a datastore. Go to pencil icon > Factory resources > datasets > plus icon for new dataset and choose azure synapse.

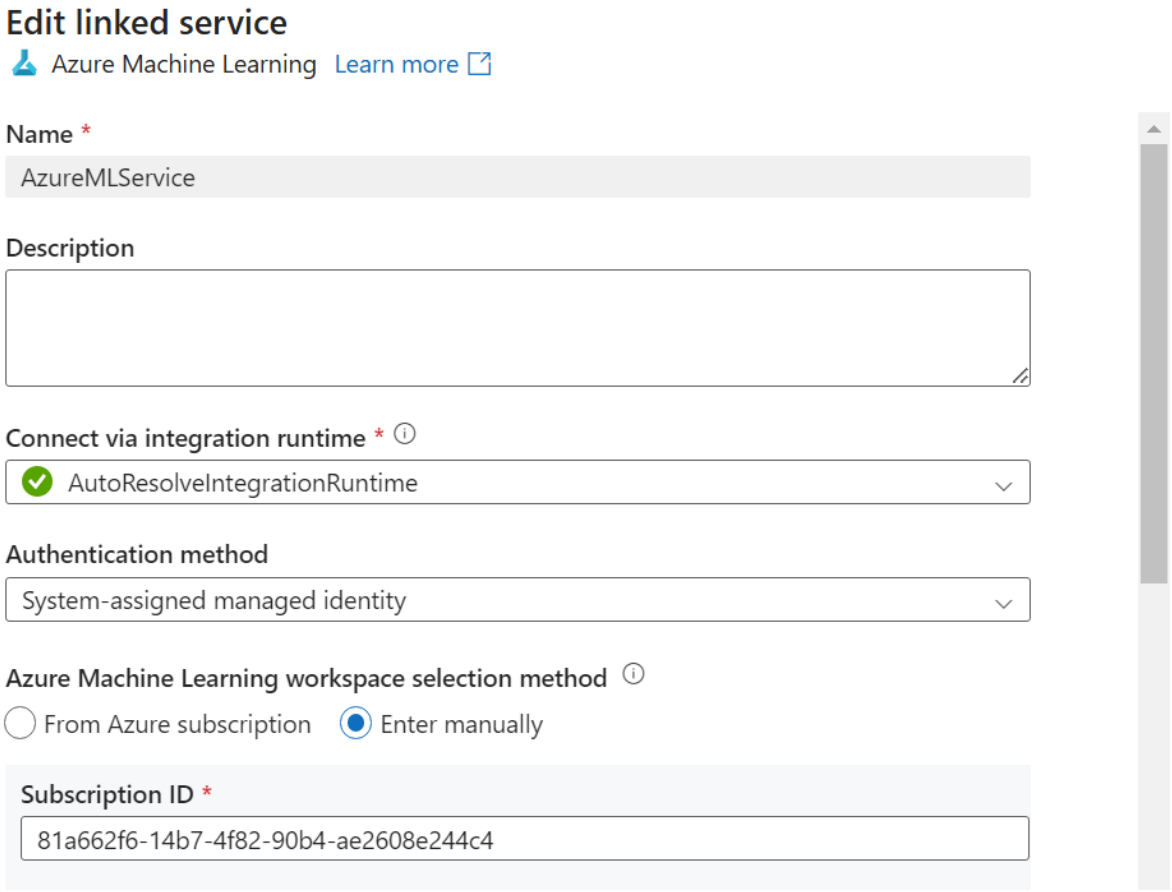
Select the datastore in the settings tab of the activity.

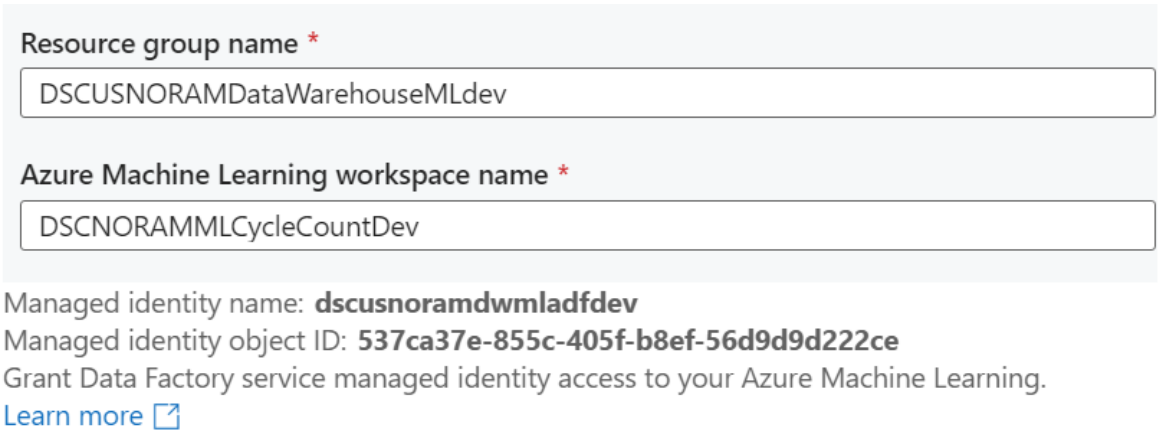
Then add query to check if the process identifier is there in the ml config table

```
select * from dbo.ml_integration_config_table where
process_identifier='@{pipeline().parameters.process_identifier}'
```

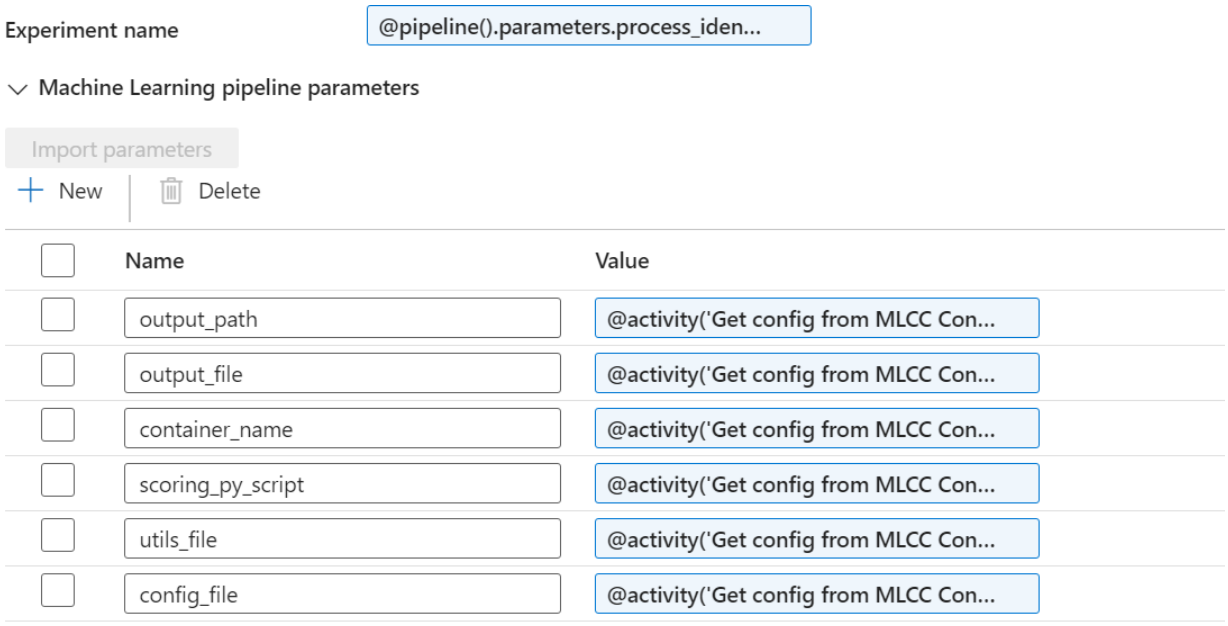
Activity 2 - Feature Engineering Notebook Execution

On the on-success button of activity 1 add this. In the general set retry and retry interval. In the **Azure Machine Learning linked service** we need to add a linked service Azure ML.





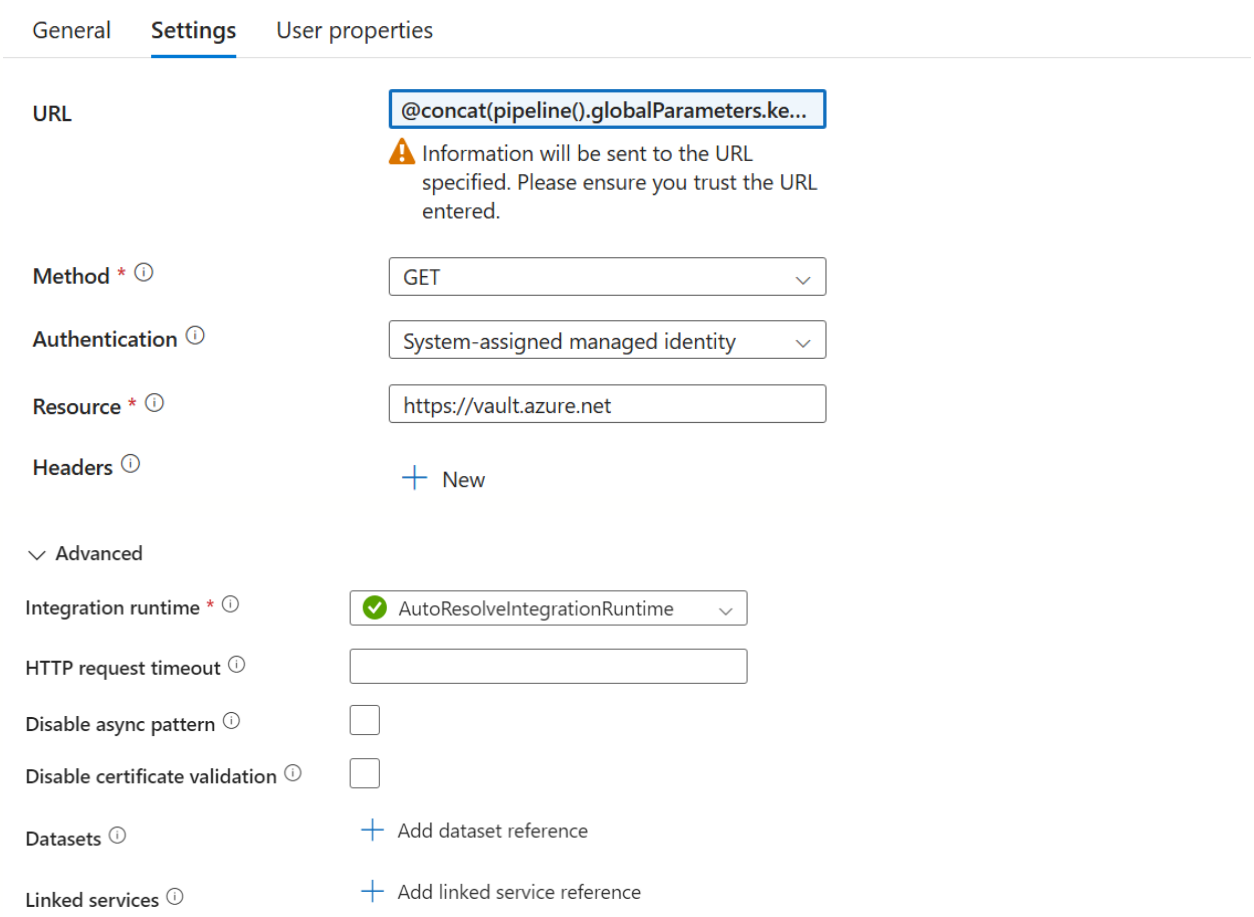
In the settings tab we use - @activity('Get config from MLCC Config Table').output.firstRow.ML_pipeline_id - to get the mlcc_id as input for this activity. To get the experiment name we use the query - @pipeline().parameters.process_identifier. in the below section Machine Learning pipeline parameters, we have added parameters as shown below.



All the details like container name, scripts for scoring config and utils are given in the ml_config table.

Activity 3 – Parallel to activity 2 – Datarobot web

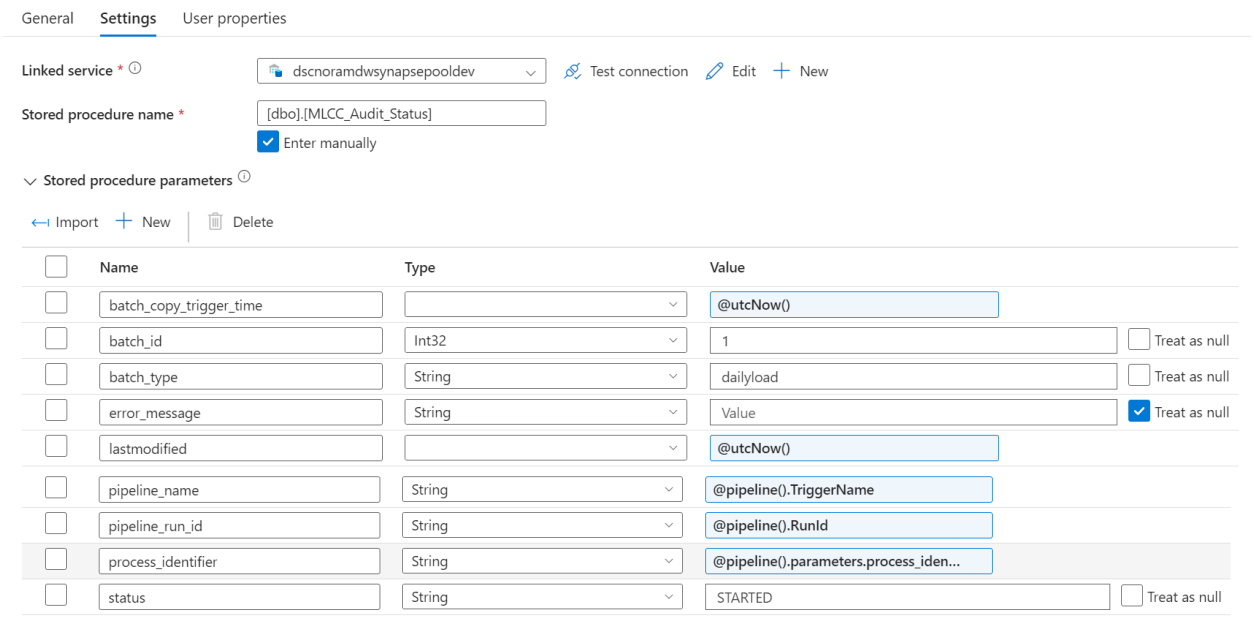
datarobot access key from KV – This activity is used to get the datarobot secret from key vault. We are using GET method. The authentication method used is managed identity.




```
@concat(pipeline().globalParameters.keyvault_url,activity('Get config from MLCC Config
Table').output.firstRow.DR_KV_secret,pipeline().globalParameters.keyvault_api_version)
```

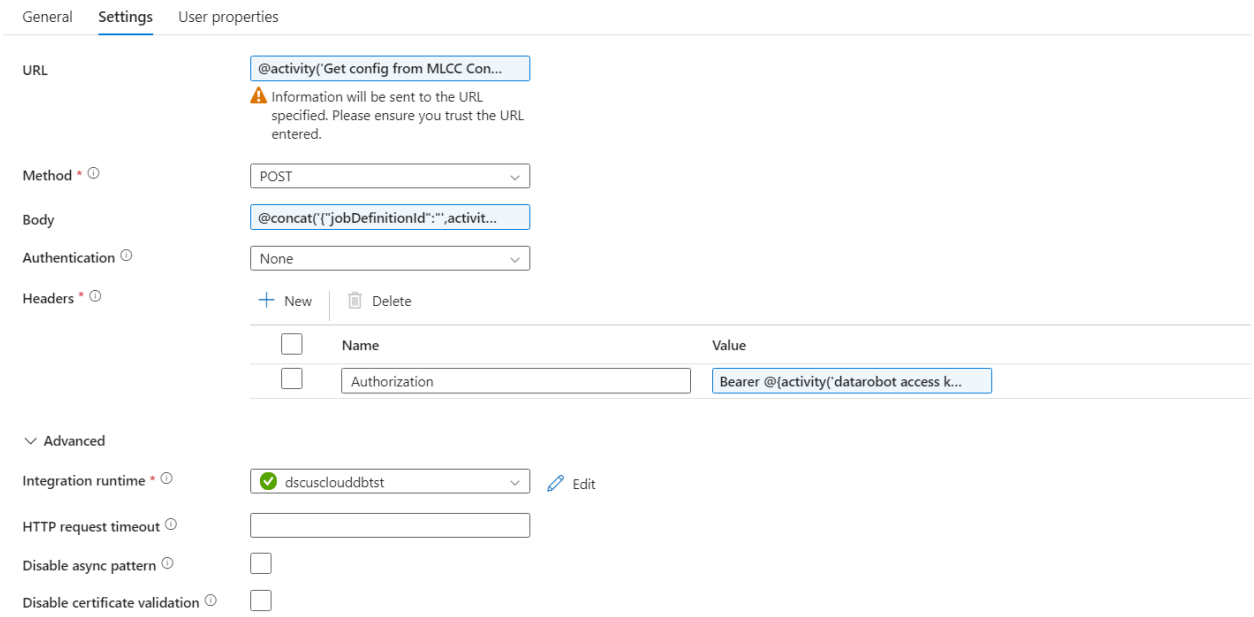
Here the `DR_KV_secret` is the secret name which is 'datarobot-access-key'. Here this link will be used in a Get request to get the secret from the key vault. The key vault url given is https://dscusnoramamdwmlvaultdev.vault.azure.net/secrets/ which is saved as a global parameter. The API version is ?api-version=7.0. So the final url is https://dscusnoramamdwmlvaultdev.vault.azure.net/secrets/datarobot-access-key/?api-version=7.0

Activity 4 - Audit START – Parallel to Activity 3



[dbo].[MLCC_Audit_Status] – This stored procedure gets executed and it creates an entry in the audit table with the pipeline name run id, run date etc.

Activity 5 – Web activity - Data Robot – Starts after activity 2 and 3



The URL part in this activity gets URL from ml config table using the first activity output using query below

```
@activity('Get config from MLCC Config Table').output.firstRow.datarobot_URL
```

This activity uses a post method to pass the job definition id of the job definition defined in the data robot using the body format of the request given below

```
@concat('{"jobDefinitionId":"',activity('Get config from MLCC Config Table').output.firstRow.jobdefinitionid,'"}')
```

For authentication to DR the below expression is used to get the access key

Here the Bearer is a header

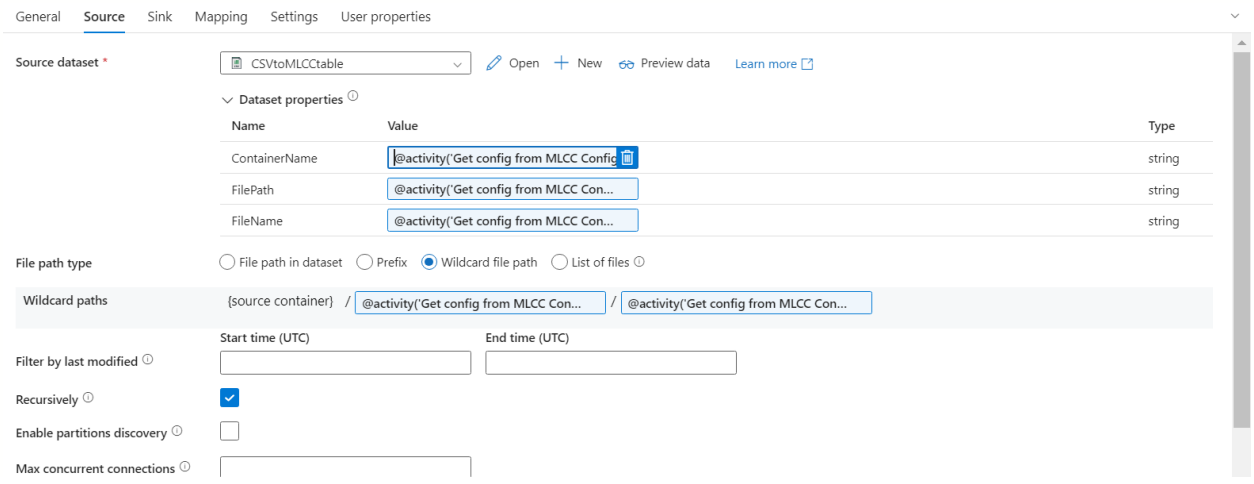
Headers that are sent to the request. For example, to set the language and type on a request: "headers" : { "Accept-Language": "en-us", "Content-Type": "application/json" }.

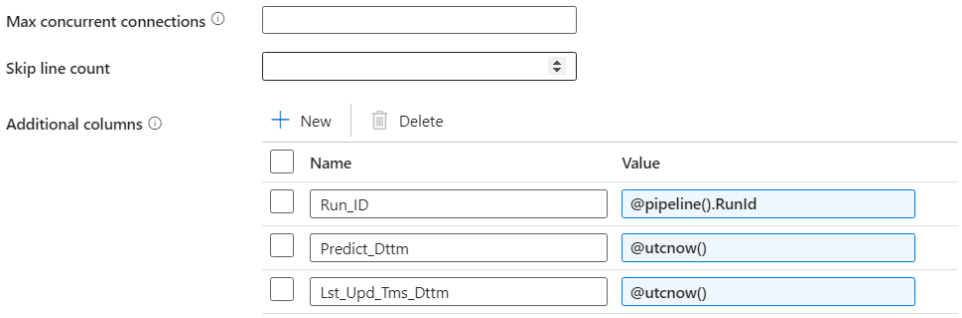
```
Bearer @{activity('datarobot access key from KV').output.value}
```

Also, we are using a specific integration runtime named dscuscloudbtst for this activity to run

Activity 6 - Copy activity - Data load to MLCC_HIST table

Source properties





Container name value from ML config table

```
@activity('Get config from MLCC Config Table').output.firstRow.Container_Name
```

File path value from ML config table

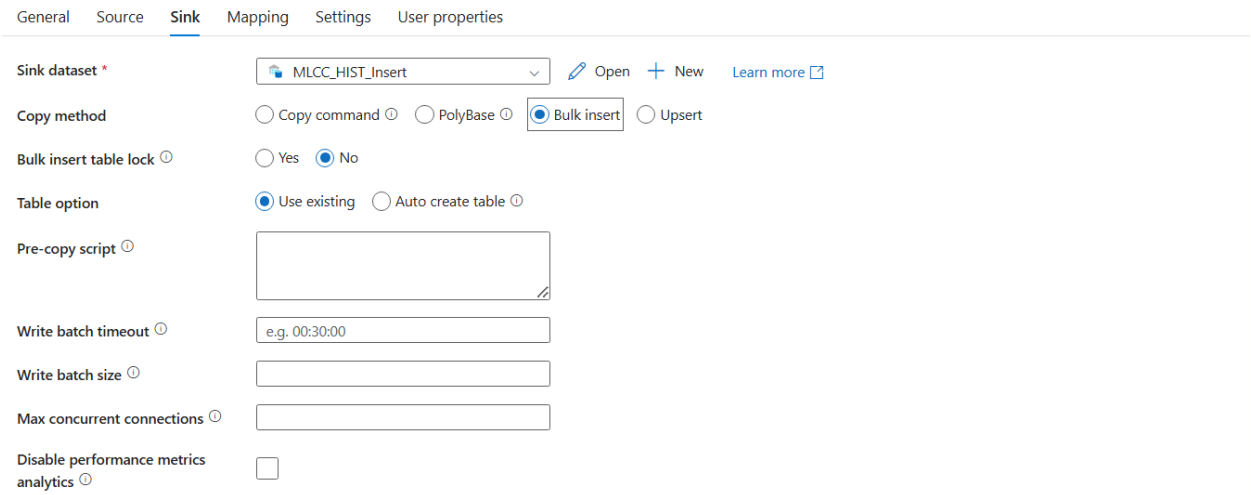
```
@activity('Get config from MLCC Config Table').output.firstRow.Container_Name
```

File name value from ML config table

```
@activity('Get config from MLCC Config Table').output.firstRow.datarobot_outputFilename
```

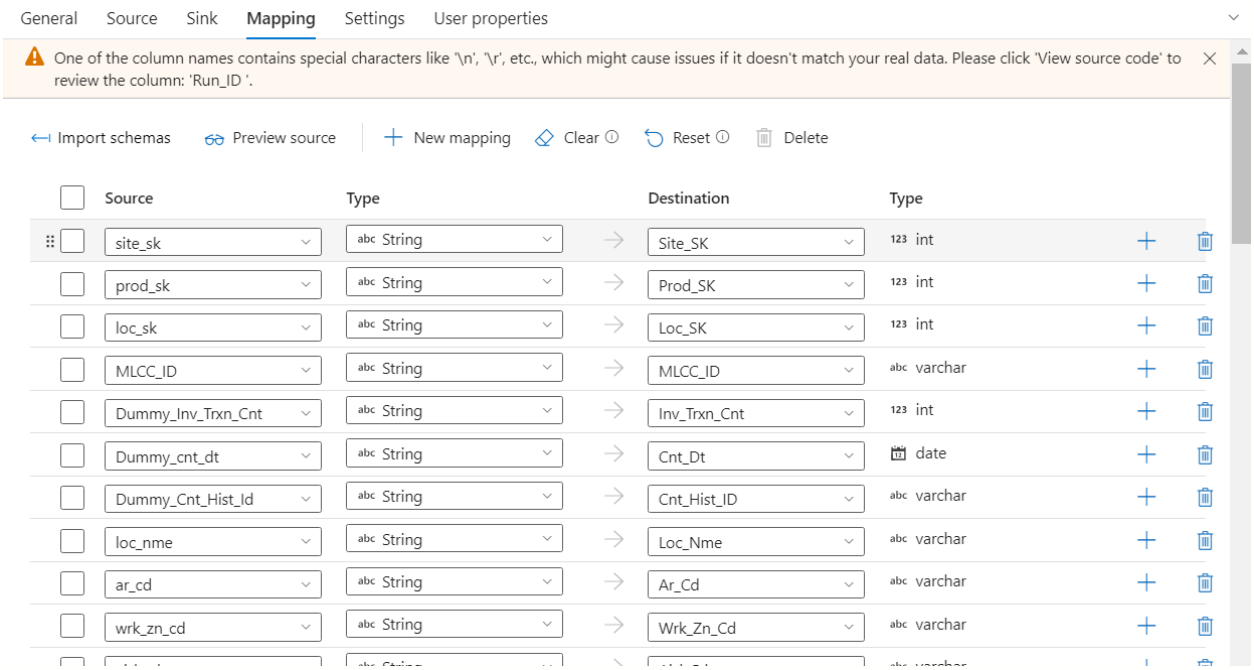
Sink properties

On the sink side we are providing the dataset(manually created) details and the copy method wen need to use. Here we are using bulk insert for copying large amount of data

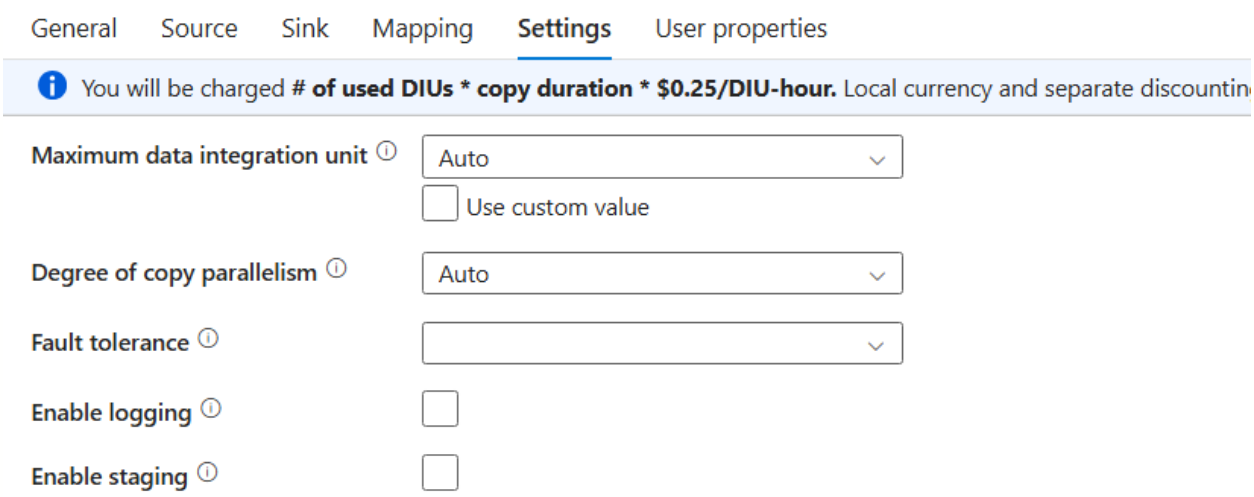


Mapping

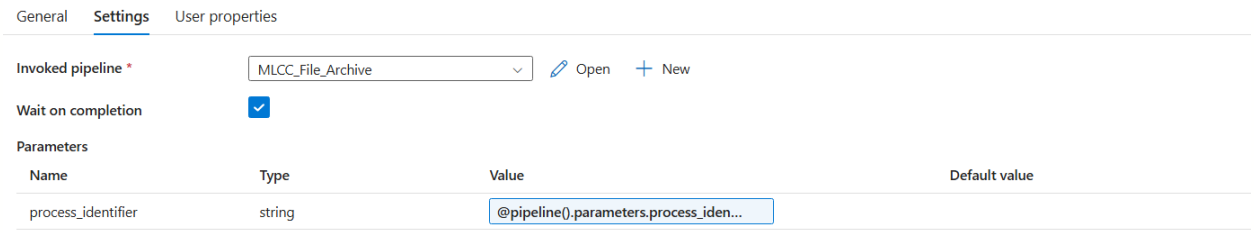
Mapping gets created automatically usually and we can verify the data type if needed



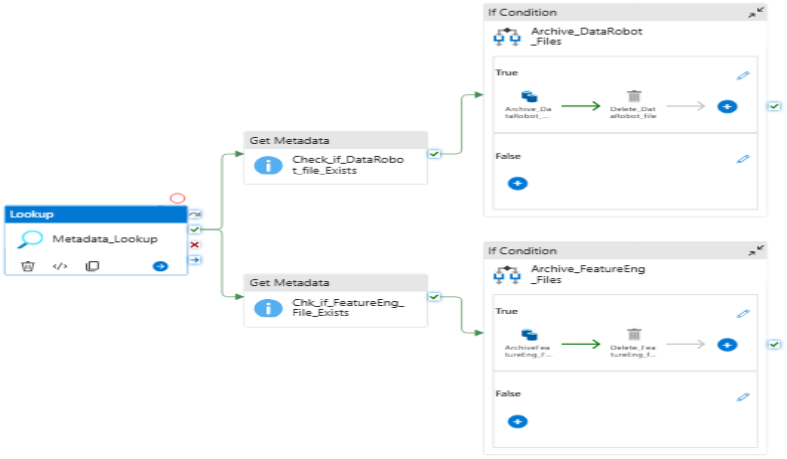
Settings



Activity 7 – Execute Pipeline - Archive_Pipeline



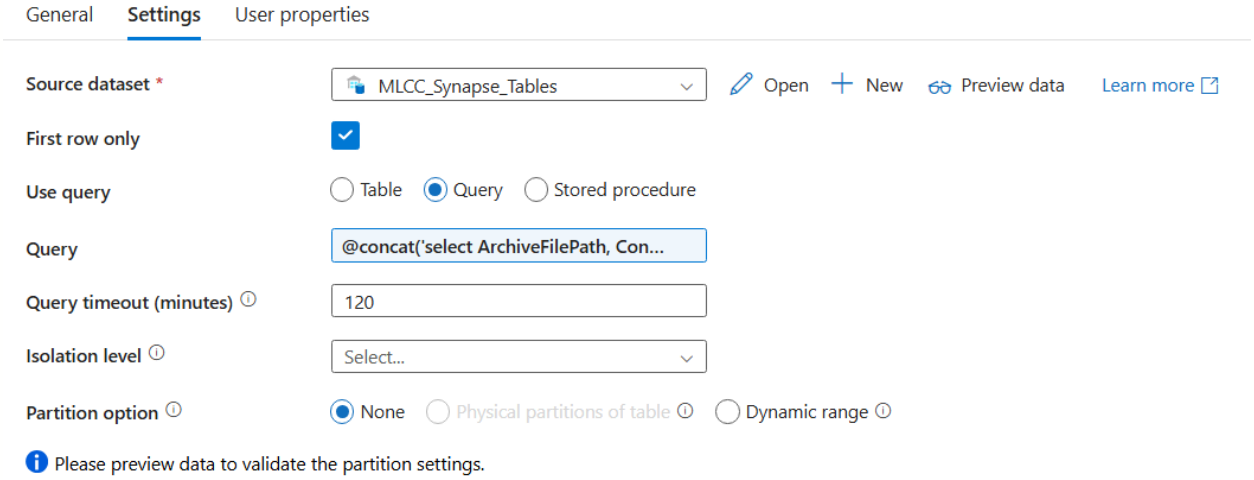
Here a pipeline MLCC_File_Archive is invoked, and process identifier is the same parameter that we pass while debugging pipeline. The process identifier is needed as a parameter to run this pipeline also.



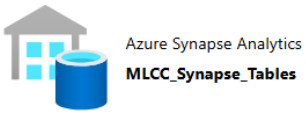
Here in the first lookup activity we are getting values of the output file name file paths etc by running a query as shown below in the dataset MLCC_Synapse_tables

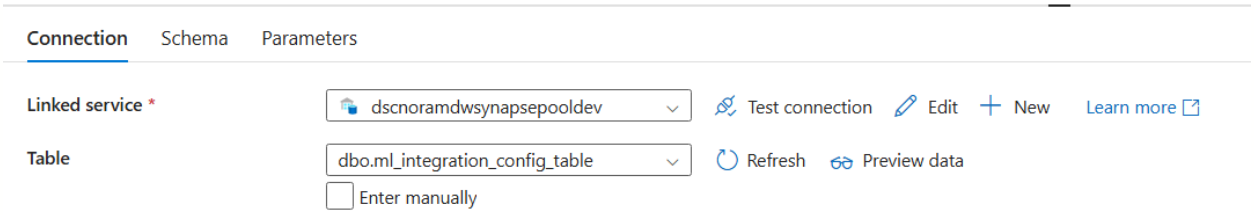
```
@concat('select ArchiveFilePath, Container_name,datarobot_outputFilename,
datarobot_outputFilePath,FeatureEngOutputFilePath, FeatureEngOutputFileName,
WMSOutputFilePath,WMSOutputFileName from dbo.ml_integration_config_table  where
process_identifier=''',pipeline().parameters.process_identifier,'''')
```

Lookup activity



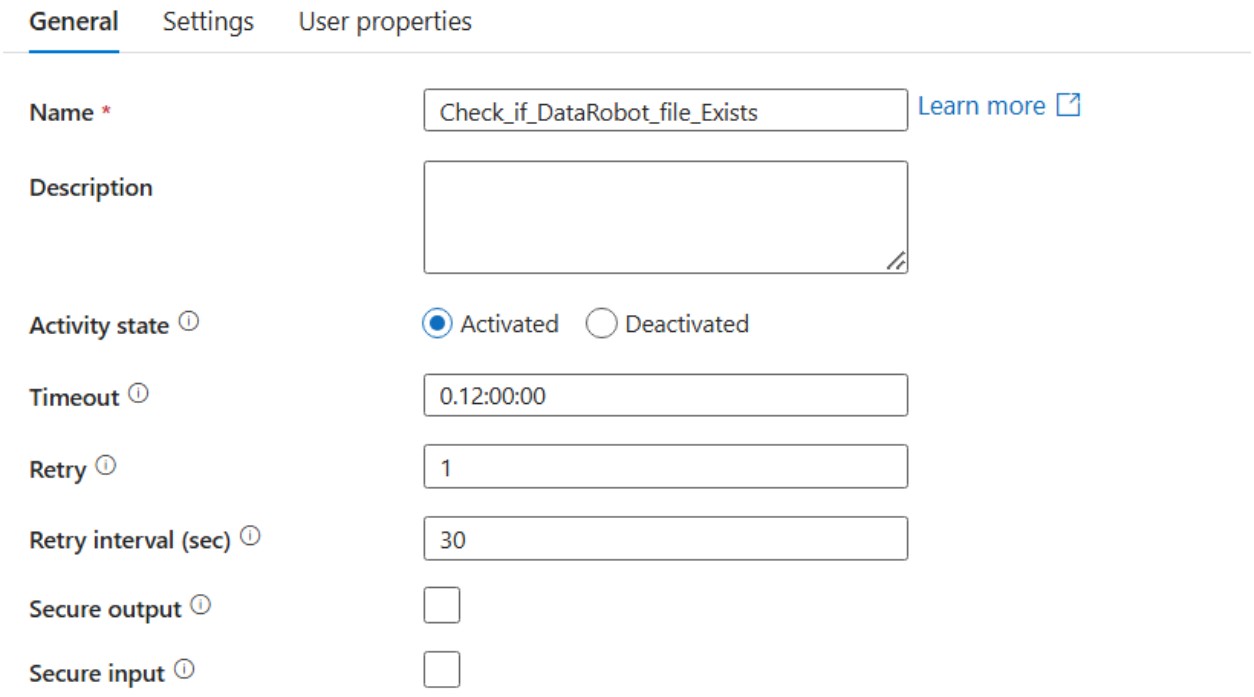
MLCC_Synapse_tables

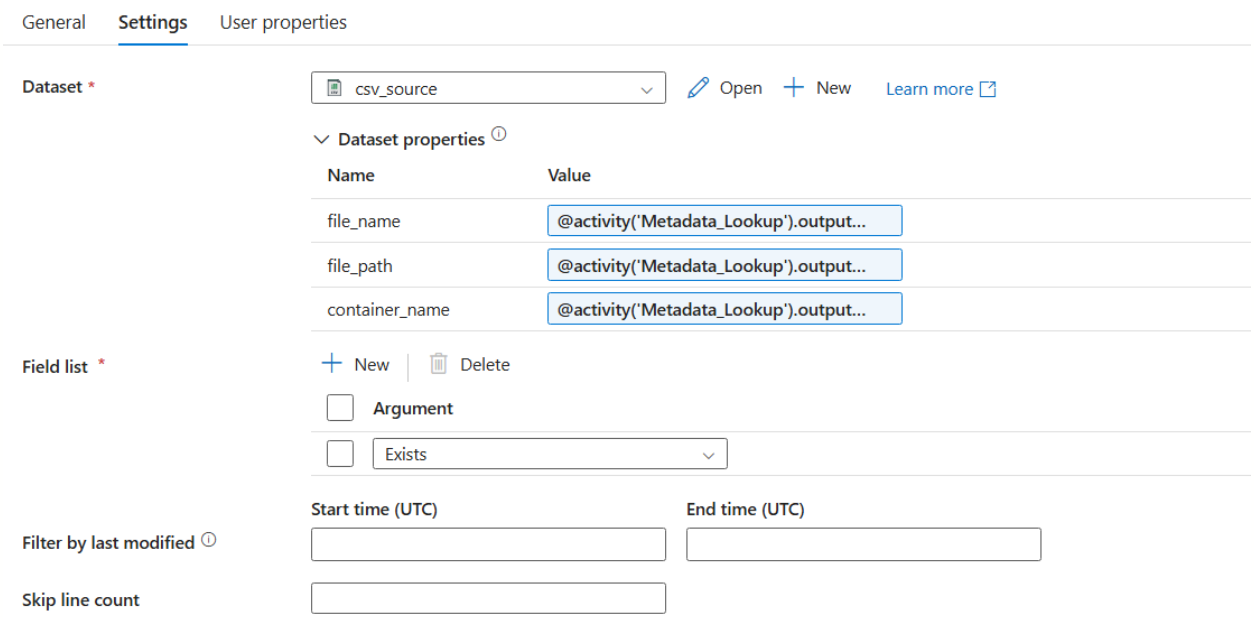




In the next activity we are checking if the Feature Engineering output file(features.csv) and Data robot output file is present in the specified blob storage which we connect using a linked service. Here the linked service is dscnoramdwmlstoragedev which is defined inside the Dataset part of the Get Metadata activity settings

Get Metadata activity



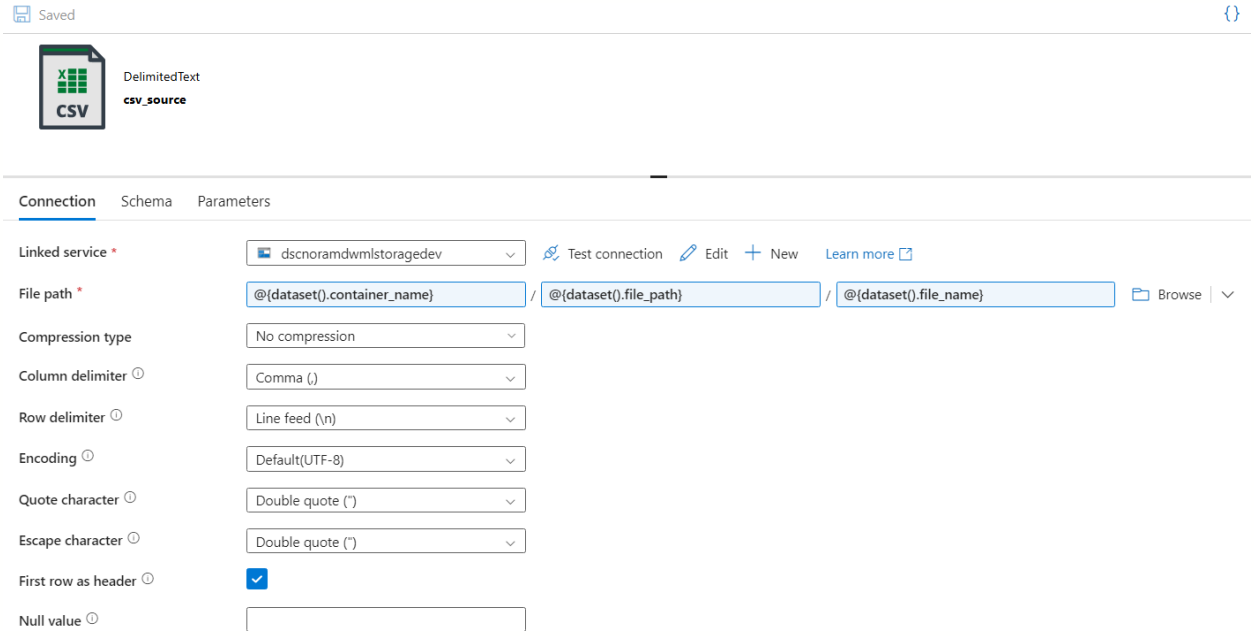



```
file name - @activity('Metadata_Lookup').output.firstRow.datarobot_outputFilename

file path - @activity('Metadata_Lookup').output.firstRow.datarobot_outputFilePath

Container_name - @activity('Metadata_Lookup').output.firstRow.Container_name
```

The csv_source is defined as

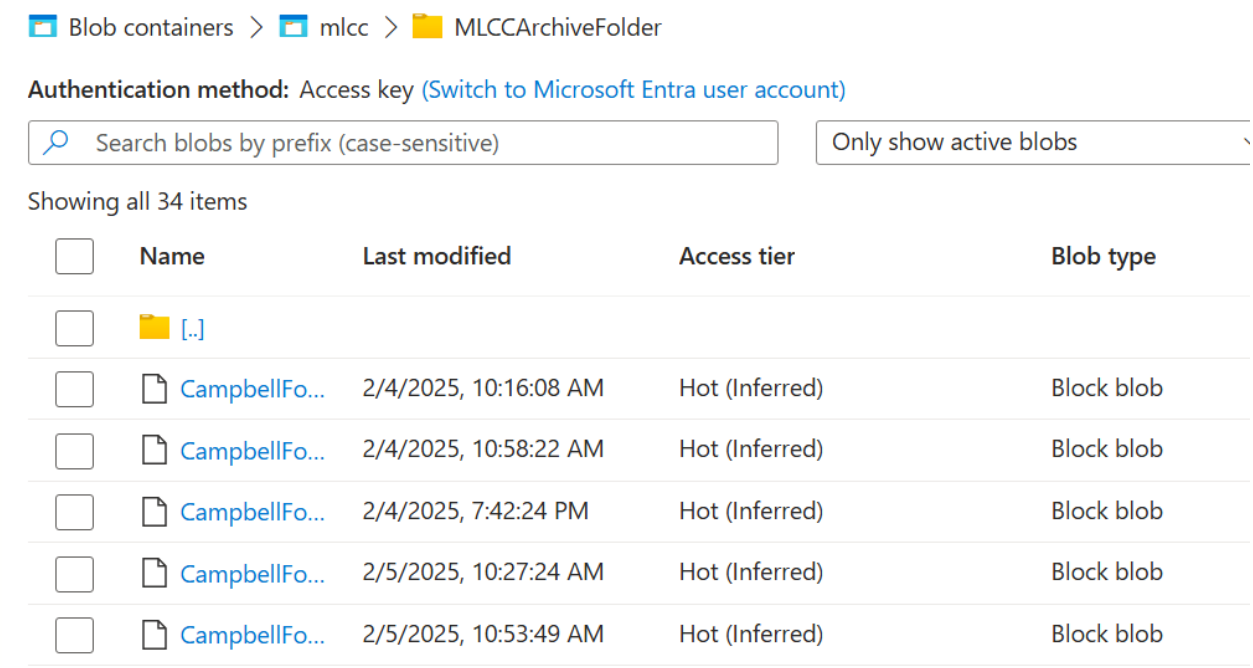


Here the file name is taken like this from the first activity of MLCC_Archive_pipeline

```
@activity('Metadata_Lookup').output.firstRow.datarobot_outputFilename
```

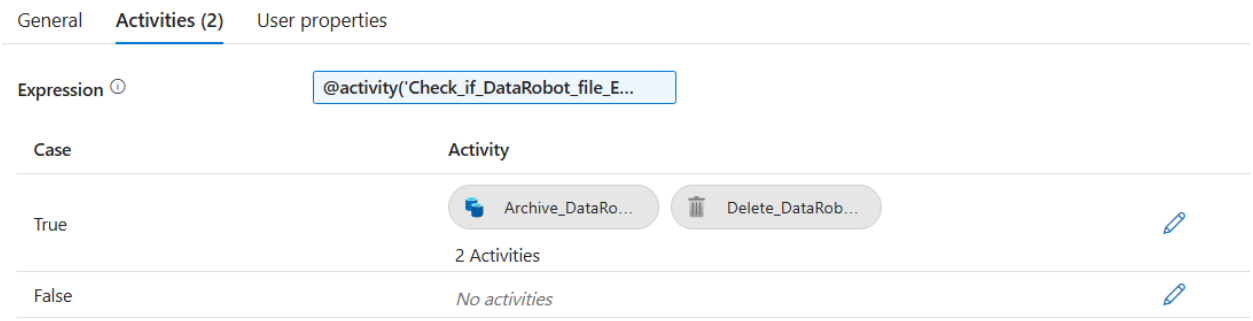
Similarly, container_name and file_path. All these values vary based on the process_identifier used in the metadata lookup activity

The next activity that happens if the files exist is a copy data activity and a delete activity which archives the files created to an archive folder named MLCCArchiveFolder within the blob storage



Here the if condition works on the basis of true or false value returned for the below Expression

```
@activity('Check_if_DataRobot_file_Exists').output.exists
```



Once the copy data(archive) activity is done the delete activity is performed.

General | Source | Sink | Mapping | Settings | User properties

Name * Archive_DataRobot_output_File Learn more

Description

Activity state ⓘ (•) Activated ( ) Deactivated

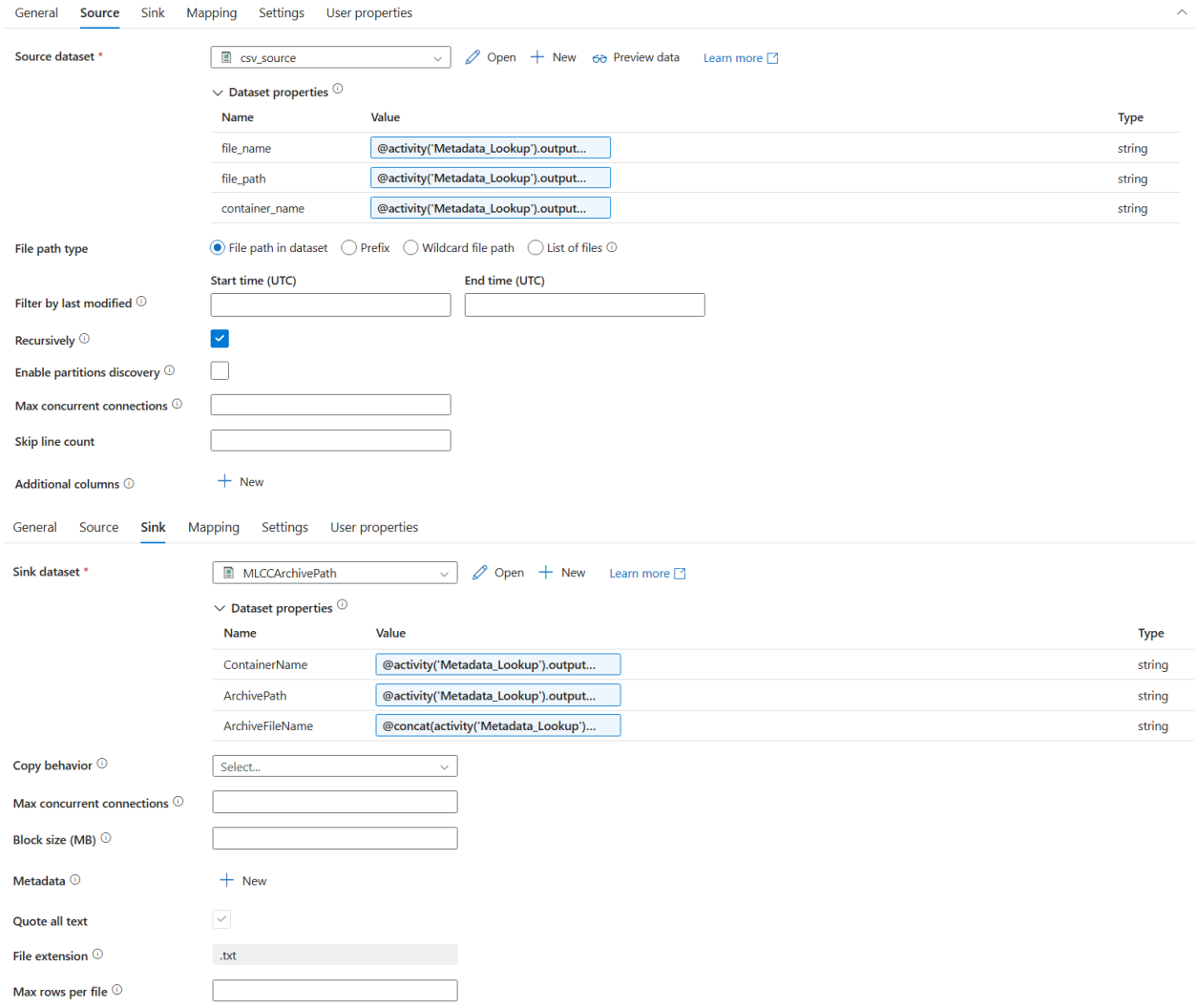
Timeout ⓘ 0.12:00:00

Retry ⓘ 1

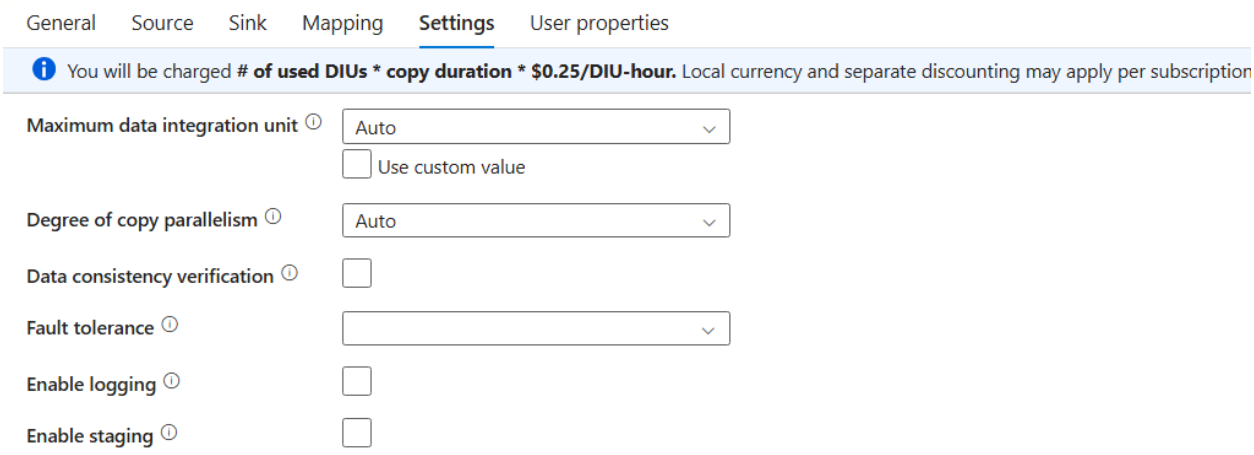
Retry interval (sec) ⓘ 30

Secure output ⓘ [ ]

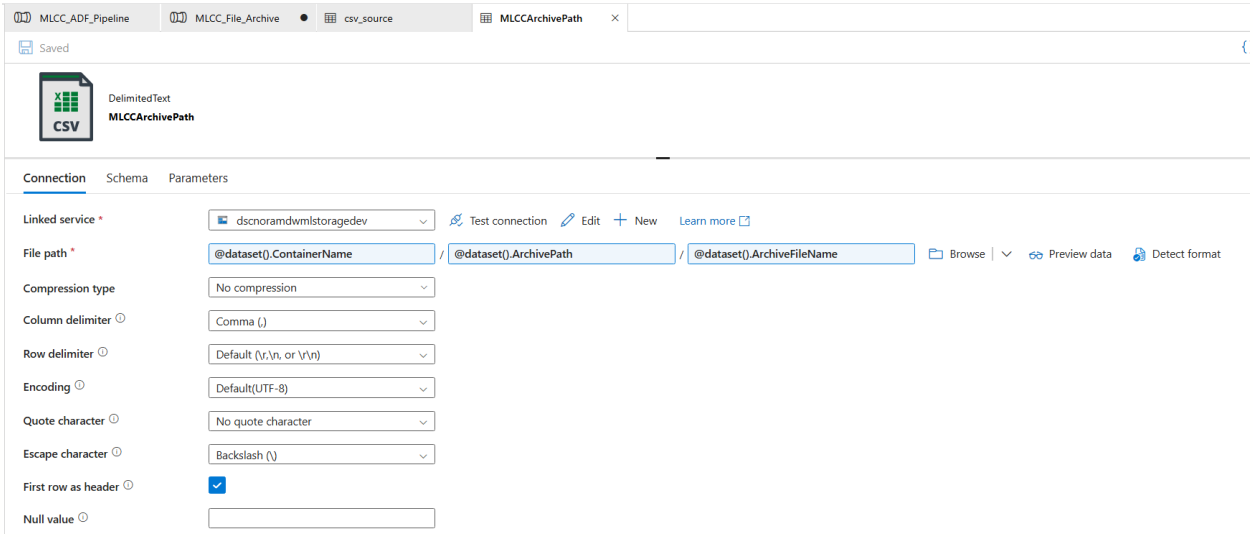
Secure input ⓘ [ ]



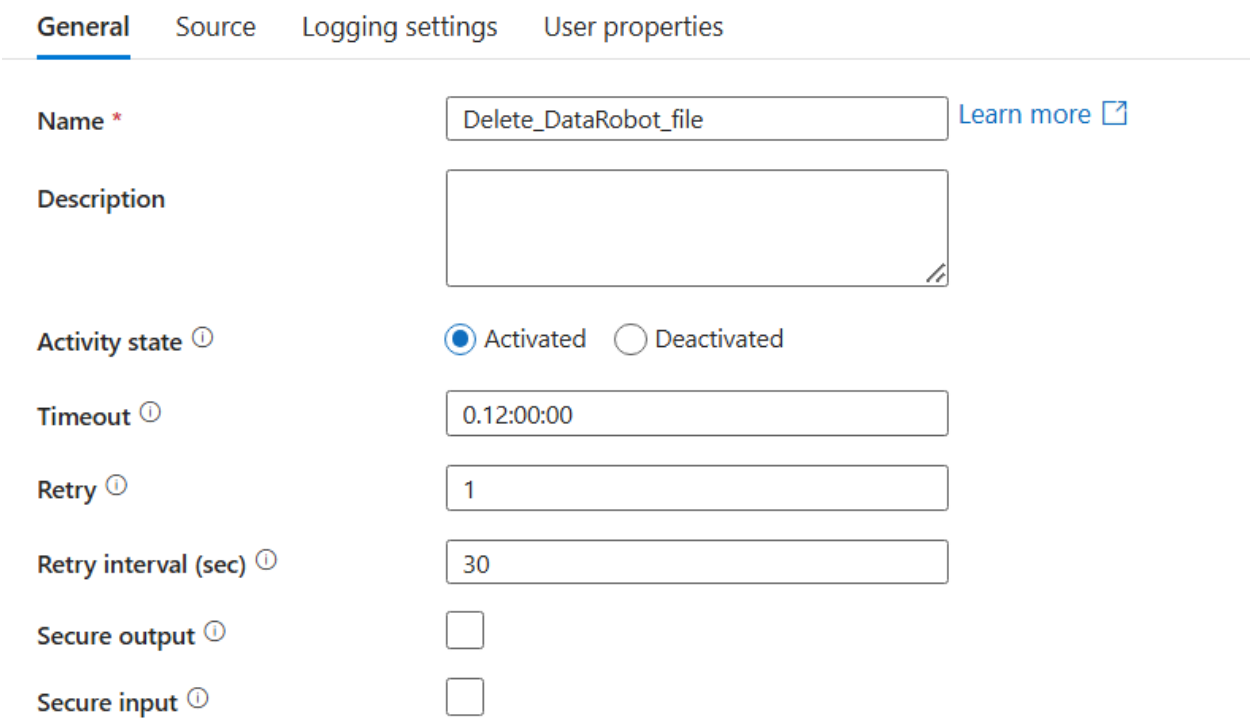
Mapping is blank

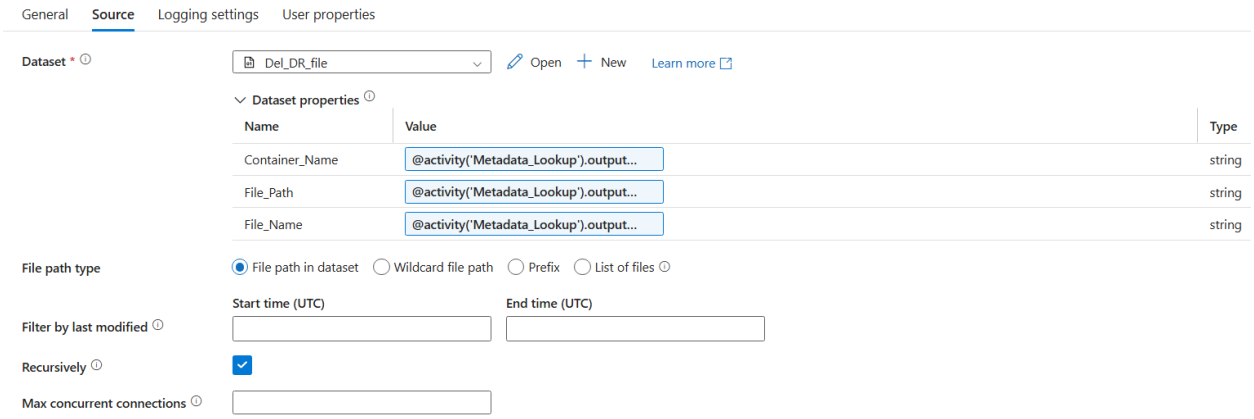


The MLCCArchivePath is a datastore as shown below



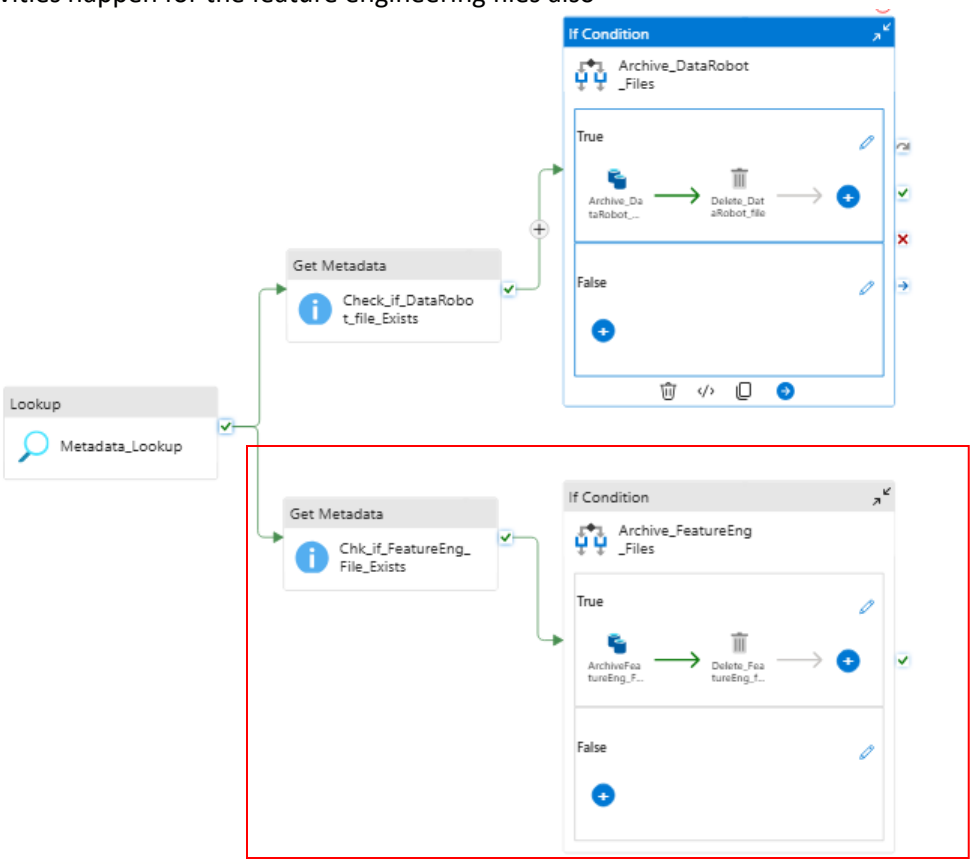
Now the delete activity





Enable logging and User properties are blank

The exact activities happen for the feature engineering files also



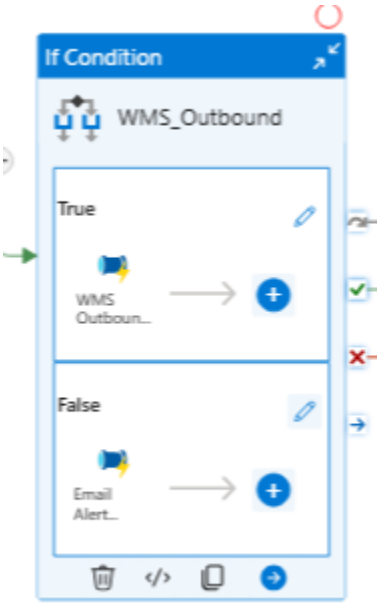
Activity 8 – WMS_Outbound - If condition

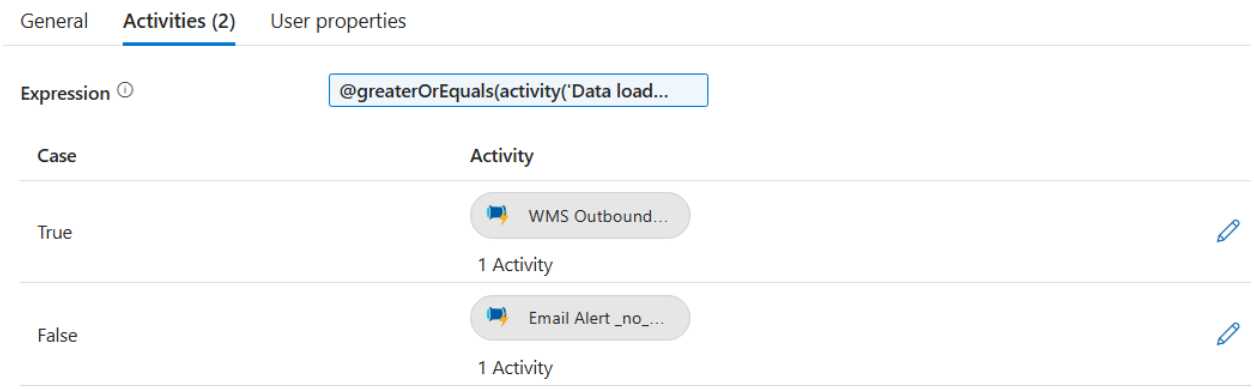
In this activity if data load has happened to MLCC_hist_table the true part which is the WMS outbound pipeline gets executed and if false an email alert pipeline gets executed

Expression used

```
@greaterOrEquals(activity('Data load to MLCC_HIST table').output.rowsCopied,1)
```

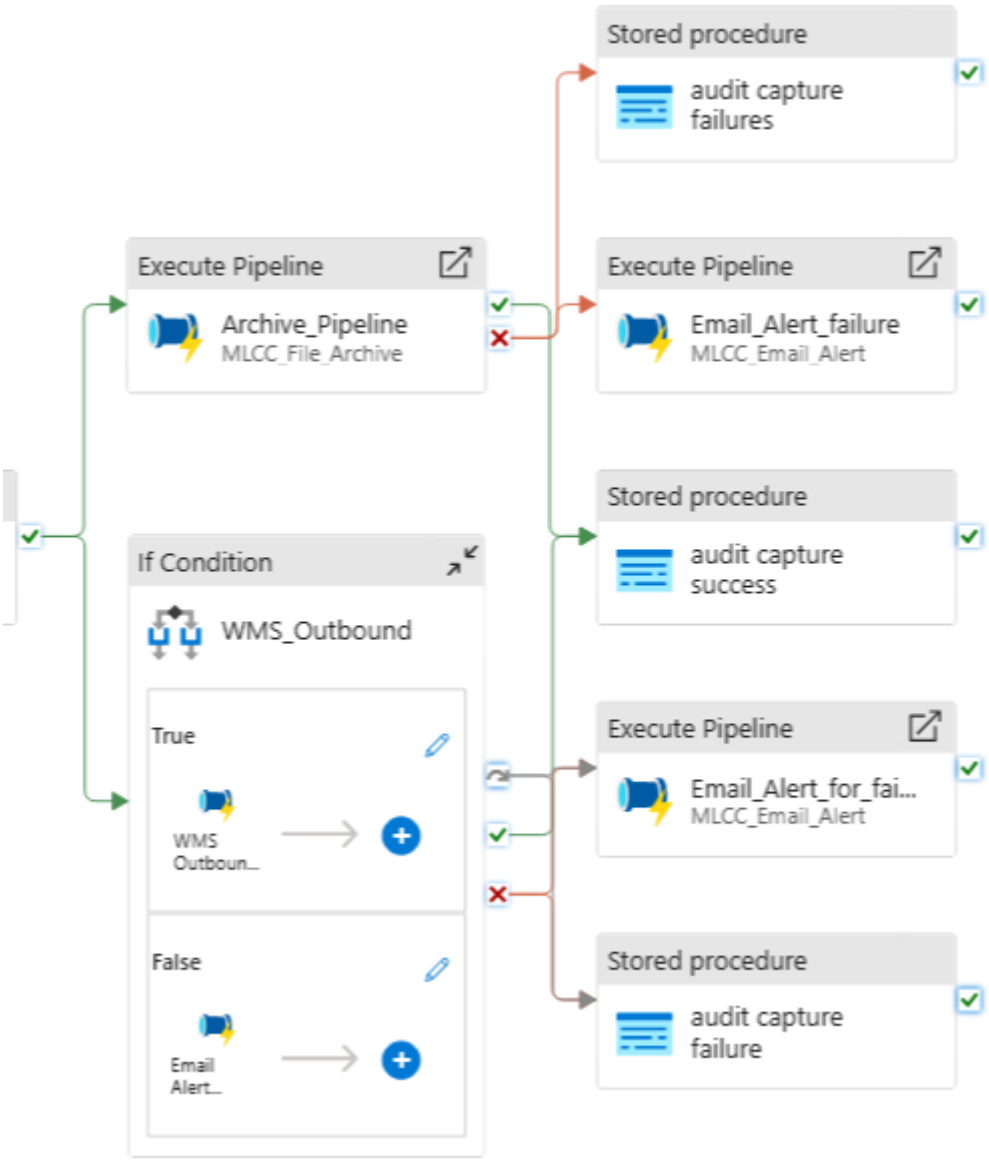
This expression is a condition that checks if the number of rows copied during the execution of an activity called **'Data load to MLCC_HIST table'** is greater than or equal to 1.





The WMS outbound pipeline loads data to the final oracle DB and if it fails an email alert is sent by triggering email alert pipeline

The last part of the MLCC_ADF_pipeline is shown zoomed below


Stored procedure
audit capture failures
Execute Pipeline
Archive_Pipeline
MLCC_File_Archive
Execute Pipeline
Email_Alert_failure
MLCC_Email_Alert
Stored procedure
audit capture success
If Condition
WMS_Outbound
True
WMS Outboun...
Execute Pipeline
Email_Alert_for_fai...
MLCC_Email_Alert
False
Email Alert...
Stored procedure
audit capture failure

Triggers

MLCC_Trigger_CarharttDC4_SMRU73201_0240	Schedule	Stopped	1
MLCC_Trigger_CarharttDC4_SMRU73201_0600	Schedule	Stopped	1
MLCC_Trigger_Clorox_SMRU1846_0545	Schedule	Started	1
MLCC_Trigger_Clorox_SMRU1846_0700	Schedule	Stopped	1
MLCC_Trigger_CloroxAberdeen_SMRU1042_0545	Schedule	Started	1
MLCC_Trigger_Danone_SMRU0085_0345	Schedule	Started	1
MLCC_Trigger_Duracell_SMRU12951_0545	Schedule	Started	1
MLCC_Trigger_Energizer_Monroe_SMRU80700_0545	Schedule	Started	1
MLCC_Trigger_EnergizerFranklin_SMRU80828_0545	Schedule	Started	1
MLCC_Trigger_EnergizerFranklin_SMRU80828_1745	Schedule	Started	1
MLCC_Trigger_Ferrero_SMRU72129_0545	Schedule	Started	1
MLCC_Trigger_FerreroGoodyear_SMRU80604_0545	Schedule	Started	1
MLCC_Trigger_General_Mills_Fort_Wayne_SMRU1856_0545	Schedule	Started	1
MLCC_Trigger_GM_Palmyra_SMRU1857_0545	Schedule	Started	1