Korn Ferry Data Lake

Ver 1.0

June 2021

Contents

[1. Introduction 4](#_Toc75189275)

[1.1. Purpose 4](#_Toc75189276)

[1.2. Scope 4](#_Toc75189277)

[1.3. Out of Scope 4](#_Toc75189278)

[2. Target State Architecture 4](#_Toc75189279)

[2.1. Key Design Considerations 4](#_Toc75189280)

[2.2. Conceptual Architecture 4](#_Toc75189281)

[2.3. Data Flow Diagram 4](#_Toc75189282)

[3. KF Framework 4](#_Toc75189283)

[3.1. Data Ingestion Framework 4](#_Toc75189284)

[3.2. Key components 4](#_Toc75189285)

[3.3. ER Diagram 4](#_Toc75189286)

[3.4. Process Flow 4](#_Toc75189287)

[3.5. Data Validation 4](#_Toc75189289)

[3.6. Modularized Functions 4](#_Toc75189290)

[3.7. Restart ability 4](#_Toc75189291)

[3.8. Retry 4](#_Toc75189292)

[3.9. Logging Mechanism 4](#_Toc75189293)

[3.10. Alert and Notification 4](#_Toc75189294)

[3.11. Exception Handling 4](#_Toc75189295)

[4. Source System 4](#_Toc75189296)

[5. Data Loading Strategy 4](#_Toc75189297)

[5.1. Full Load 4](#_Toc75189298)

[5.1.1. ADF Job name: 4](#_Toc75189299)

[5.1.2. Command to run: 4](#_Toc75189300)

[5.1.3. Steps for Full Load: 4](#_Toc75189301)

[5.2. Delta Load (Incremental load) 4](#_Toc75189302)

[5.2.1. JOB NAME: 4](#_Toc75189303)

[5.2.2. Command to run: 4](#_Toc75189304)

[5.2.3. Steps For Delta Load: 4](#_Toc75189305)

[6. Naming & Coding Standards 5](#_Toc75189306)

[6.1. Coding Standards 5](#_Toc75189307)

[6.2. Table Names 5](#_Toc75189308)

[6.2.1. Data Ingestion Framework Tables 5](#_Toc75189309)

[6.2.2. Data Lake Layer Tables 5](#_Toc75189310)

[6.2.3. Delta Layer Tables 5](#_Toc75189311)

[6.3. Variable Names 5](#_Toc75189312)

[6.4. Function Names 5](#_Toc75189313)

[6.5. Script Names 5](#_Toc75189314)

[7. Retention Strategy 5](#_Toc75189315)

[8. Capacity Plan 5](#_Toc75189316)

[8.1. Hardware 5](#_Toc75189317)

[8.2. Software 5](#_Toc75189318)

[9. Git hub Process 5](#_Toc75189319)

[10. Job Orchestration (ADF) 5](#_Toc75189320)

[11. Spark Process 5](#_Toc75189321)

[11.1. Requirement 5](#_Toc75189322)

[11.2. Process Flow 5](#_Toc75189323)

[11.3. Data Used 5](#_Toc75189324)

[11.4. Step by step Solution 5](#_Toc75189325)

[12. Go Live Process (Dev-Ops) 5](#_Toc75189326)

[13. Run Book 5](#_Toc75189327)

[14. Maintenance 5](#_Toc75189328)

[15. Data Governance 5](#_Toc75189329)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Version No.*** | ***Date*** | ***Description*** | ***Author*** | ***Reviewed By*** |
| 1.0 | 21/06/2021 | Initial document | Prashant Rajguru |  |

# Introduction

## Purpose

## Scope

## Out of Scope

# Target State Architecture

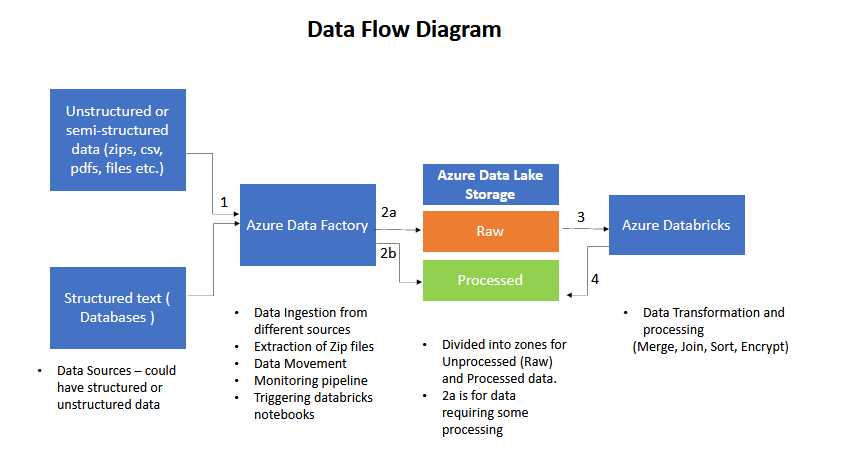
## Key Design Considerations

## Conceptual Architecture

## Data Flow Diagram

## A high-level architecture with data flow diagram is provided below. In the case of Linkup data, the source data is present in an external SFTP Server which contains zipped xml files. It is fed into the Azure Data Factory after configuring linked services to connect to SFTP Server.

## It is loaded into ADLS gen2 via ADF into a staging directory. Databricks notebook is later run on the extracted xml files and it creates delta tables and saves them to ADLS Gen 2 Storage account.



# KF Framework

## Data Ingestion Framework

## Key components

## ER Diagram

## Process Flow



## Data Validation

## Modularized Functions

## Restart ability

## Retry

## Logging Mechanism

## Alert and Notification

## Exception Handling

# Source System

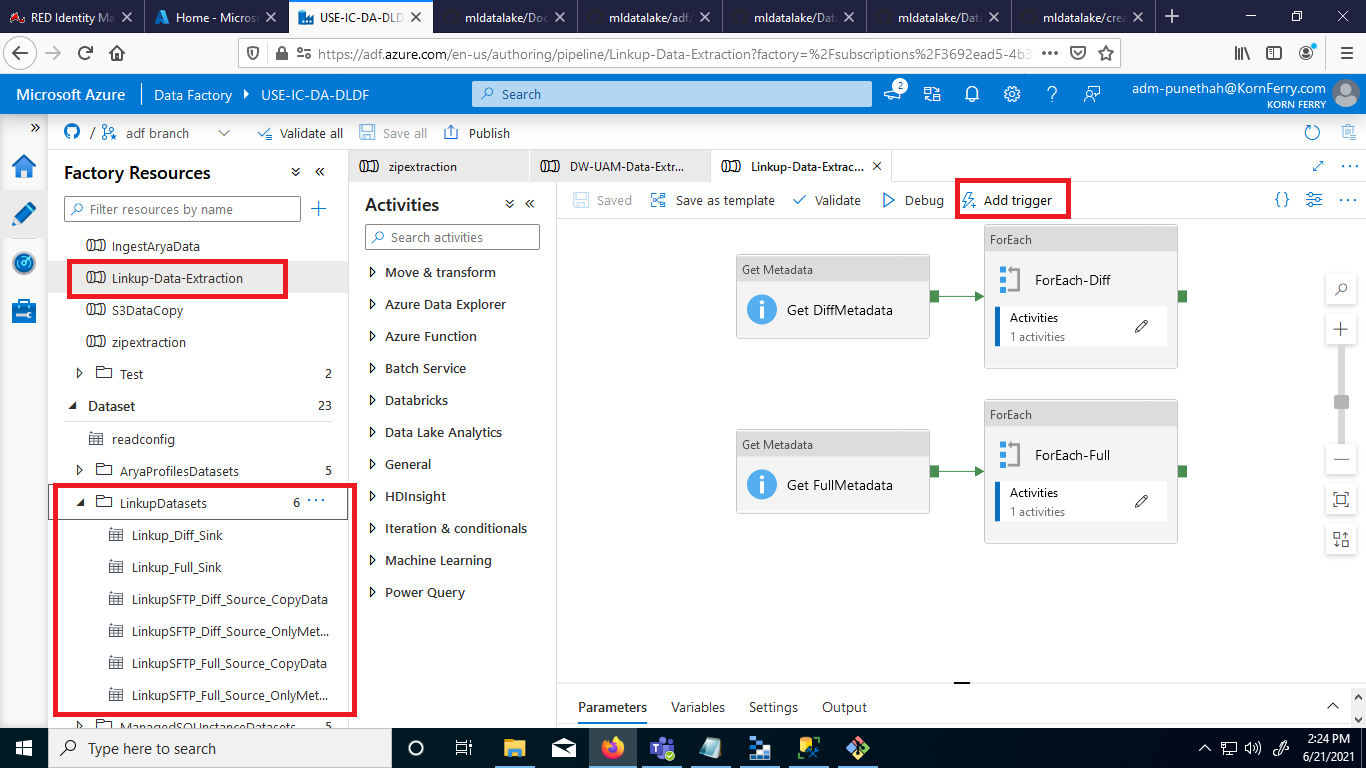
# Data Loading Strategy

## Full Load

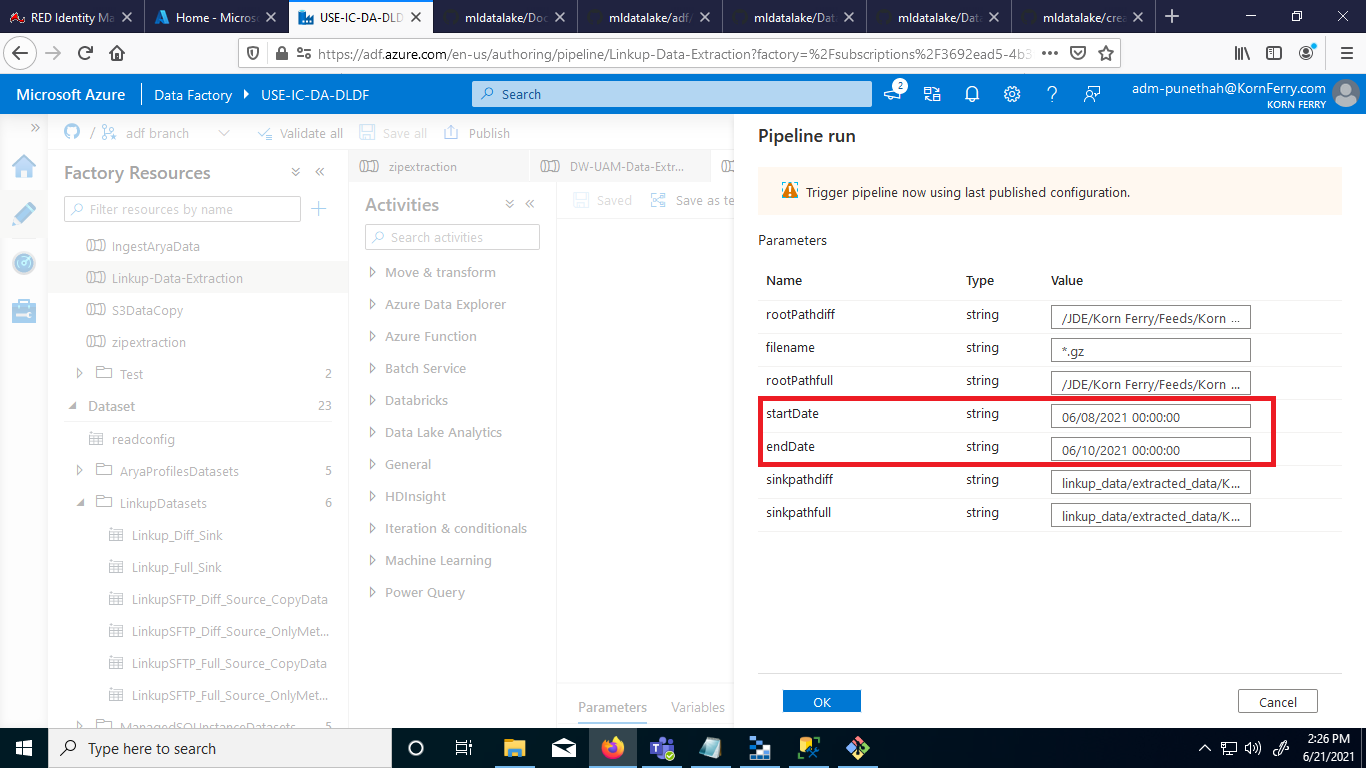
## ADF Job name: Linkup-Data-extraction

## Command to run:

* + The pipeline to be executed from ADF is Linkup-Data-Extraction. Once triggered, it will take some input parameters (for example start and end date). The pipeline will extract and copy only those files which are modified in the specified range.



* On triggering the Linkup-Data-Extraction pipeline, the parameters startDate and endDate must be changed to specify the period for which the copy command is to be run.



## Steps for Full Load:

For running the pipeline for full load, change the parameters startDate and endDate to a very long range or leave them blank. The parameters are set on the second screenshot shown in the commands to run section.

## Delta Load (Incremental load)

## JOB NAME: Linkup-Data-extraction

## Command to run: Same as in Full Load.

## Steps For Delta Load: For running the pipeline for delta load, specify the startDate and endDate for which the files are to be copied and extracted.

# Naming & Coding Standards

## Coding Standards

## Table Names

## Data Ingestion Framework Tables

## Data Lake Layer Tables

## Delta Layer Tables

## Variable Names

## Function Names

## Script Names

# Retention Strategy

# Capacity Plan

## Hardware -

## Software - Azure Data Factory, Azure Databricks

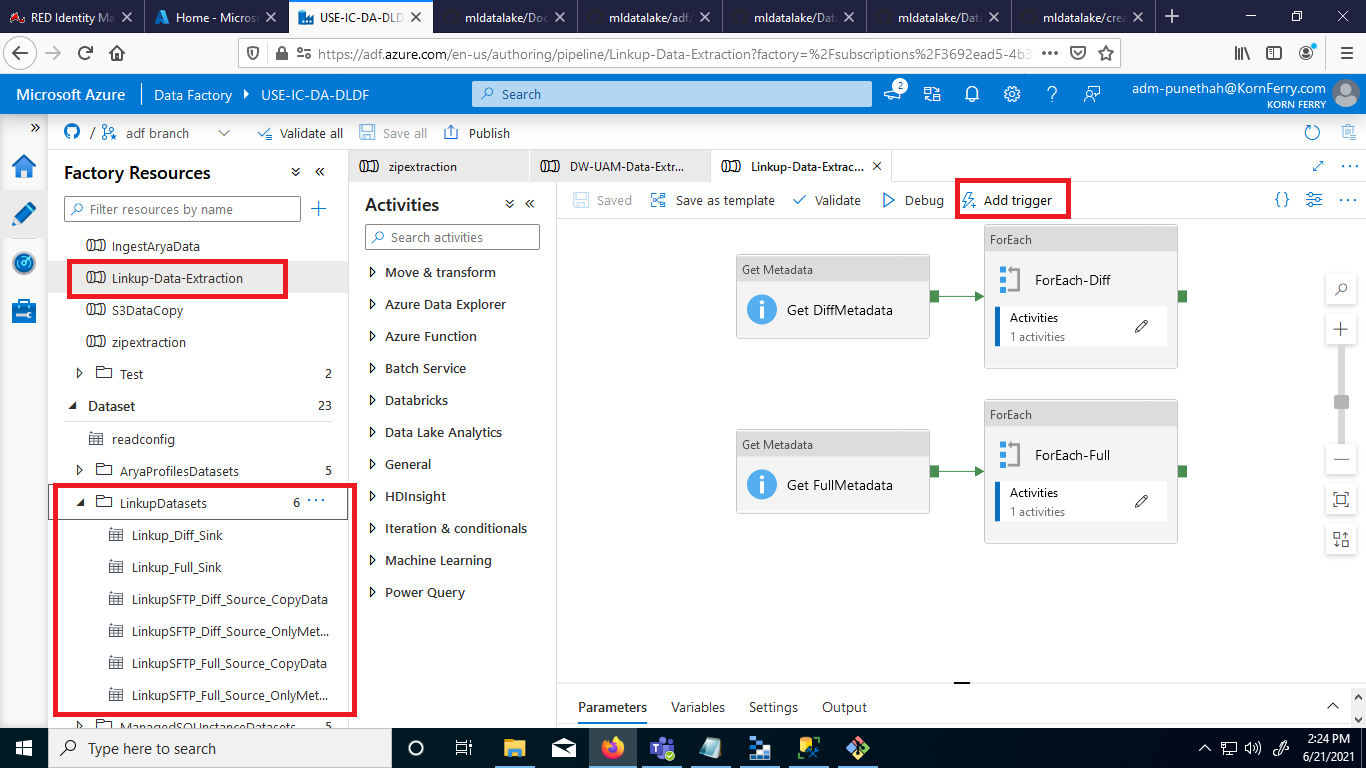
# Git hub Process

# Job Orchestration (ADF)

**ADF - Pipeline Data Sources**

There are 6 datasets defined for Linkup Data (two of these data sources are only used for getting metadata from the directory)

* + LinkupSFTP\_Diff\_Source\_OnlyMetadataLookup - only used for getting the folde listings inside the Diff directory
  + LinkupSFTP\_Full\_Source\_OnlyMetadataLookup - only used for getting folder listings inside the full directory.
  + LinkupSFTP\_Diff\_Source\_CopyData – used for copying data
  + LinkupSFTP\_Full\_Source\_CopyData
  + Linkup\_Diff\_Sink
  + Linkup\_Full\_Sink



**ADF - Pipeline Activities**

There are 7 activities in the pipeline out of which 4 activities run for UAM tables and 4 activities run for Datawarehouse tables.

* + Configlookup – This activity performs a lookup on the config table which is created for orchestration purposes. It gets the entries added in the config table and passes only those table entries further for which Enabled flag is set to true.
  + DWFilter – Used for filtering the table entries which are present in DataWarehouse database and only allowing those table entries to pass further to loop
  + ForEach (following DWFilter) – This activity is run for all the entries which were obtained after DWFilter activity
  + If Activity inside ForEach (DWFilter) - If activity checks if incremental flag is set to true or false. For full data copy it runs the false portion of if activity.

Merging Algorithm.

* Load first day full data into a delta table (final\_table) and load it with name Final DF
* Add columns for below:-
  + LoadDate (Same as the source folder date),
  + UpdateDate (Default value – Null) - to be updated only when theres an updated record
  + DeletedDate (Default value – Null) - to be filled only whena record is deleted
  + DeletedFlag (True or False) True – Record has been deleted, False- default value
* Perform in loop for all days one by one :-
  + Perform Merging and deletion operations of same day diff data-
    - Load Diff data for the day
      * Split diff data into three separate dataframes based on status value as I, U, and D where I refers to insertion, U represents Update and D refers to deletion
      * For Insert dataframe – add four columns
        + LoadDate – same as folder name
        + UpdateDate – Null
        + DeletedDate – Null
        + DeletedFlag – False
      * For Update Dataframe, add four columns
        + LoadDate – same as folder name
        + UpdateDate – same as folder name for diff being processed
        + DeletedDate – Null
        + DeletedFlag- False
      * For Delete Dataframe, add four columns
        + LoadDate – same as foldername
        + UpdateDate – Null
        + Deleteddate = same as foldername
        + Deletedflag = True
      * Create Merge df by taking union of Update DF and Insert DF and drop duplicate job ids
      * Perform merge of Merge df into the delta table (final\_table)
      * Perform merge of delete df into delta table such that only deletedflag and deleteddate column is updated.
  + Perform merging of full data from next such that only those job ids are added which were absent from final data of previous day :-
    - Load Next day data in df and obtain left anti join of Next day df and final df
    - Merge the leftantijoin df data into the delta table (final\_table)
  + Drop duplicate job ids from data.

# Spark Process

### 11.0.1 Requirement

1 - Microsoft Azure Subscriptions

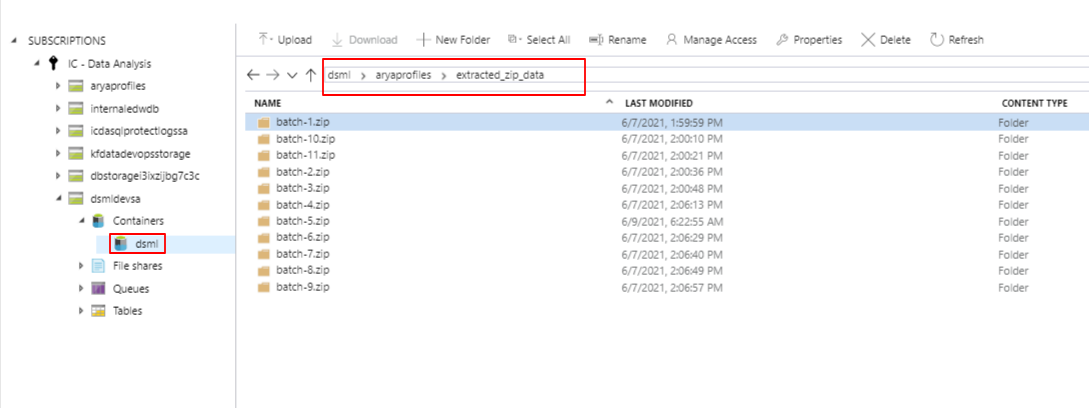
2 - Azure Databricks Instance/Subscription

3 - Databricks Cluster with Scala/SQL/Python language configured.

4 - Notebook that contain multiple commands in sequence for data pipeline to perform ETL

### 11.0.2 Process Flow

Data Location where we stored in the ADLS

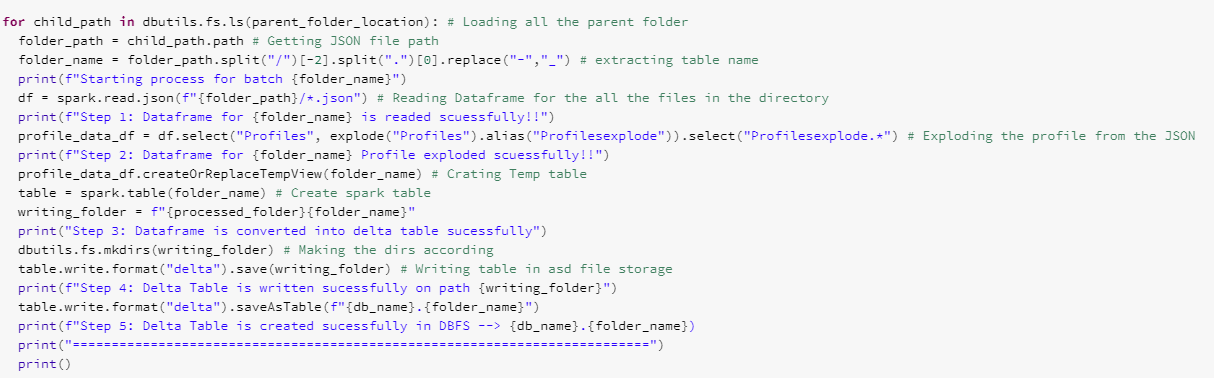


GitHub Link of the [Notebook: https://github.com/HayGroup/mldatalake/blob/databricks/source/notebooks/arya\_profile/read\_data.py](https://github.com/HayGroup/mldatalake/blob/databricks/source/notebooks/arya_profile/read_data.py)

Step 1: Initialize all the required variables in the top of the notebooks so you case use them in the entire notebook



Step 2: Now navigate using dbutils within ADLs GEN2 instance container to the desired files.



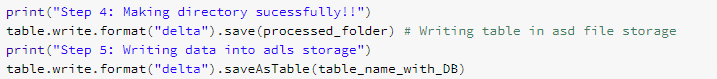
Step 3: Now read that file according to their format in spark data frame.



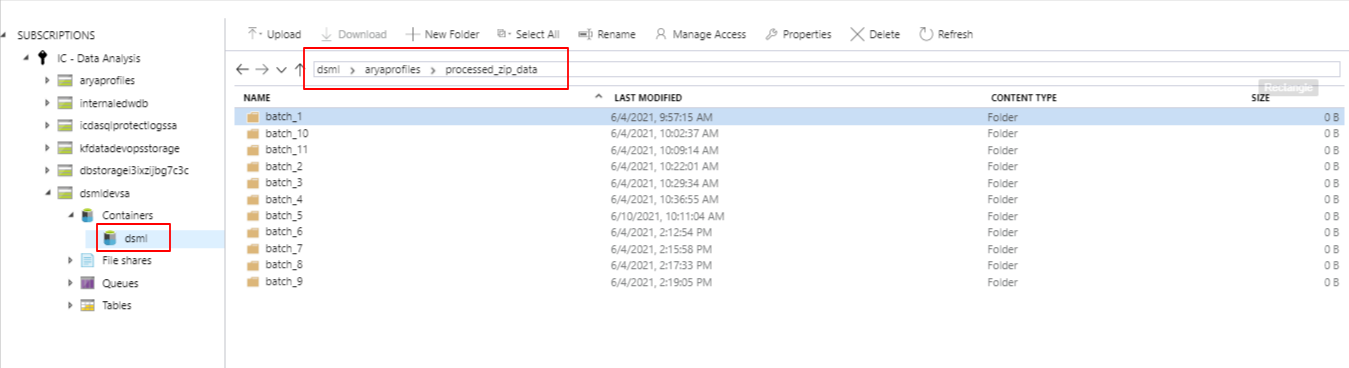
Step 4: Now make temp table in memory form that data frame to check that is compatible to write in delta table.



Step 5 : Once the step 4 is verified now write that temporary table in DBFS in delta format.



Location of ADLS GEN2 where we Dump all the delta tables data.



### 11.0.3 Data Used

1 – We can read that data in pyspark in data frame format

2 – We can use that in SQL format also

3 – That Data strictly maintains the ACID properties

4 – We can use in various place for analysis or for model training, The whole fetch on the tip of the finger

# Go Live Process (Dev-Ops)

# Run Book

# Maintenance

# Data Governance