I hope that I’m not getting this wrong cause the title of assignment says, “Coding Assignment” but, this assignment is architectural and conceptual task rather than coding. Maybe I got it wrong ☺

Assignment 1)

Here are some of my key take aways from the task’s definition:

1. The solution has to support daily ingestions. So it will be a batch one.
2. Solution has to find new files that are added to the centralized ADLS and then load to MS team’s corresponding storage account.
3. Solution has to only operate in case a new file is loaded to MS team’s storage account. If there is no new file, nothing needs to be processed.
4. After the file is loaded to the target MS team’s storage account, solution has to read the file, does the required transformation and implement the required business logic on top of it to propagate the dashboard.

Considering the above mentioned keys, my suggestion would be to create a ADF pipeline with all the required linked services defined. These Linked services will define the connection to source and sink which in our case, is ADLS details.   
The ADF will have some “GetMetaData”, “ForEach”and “CopyActivity” steps to take care of moving the daily ingested files to the MS team’s storage account. The important part here is to define a solution to understand which files are already processed and which files are new…so that ADF processes new files only.

First approach)

When loading files from different sources to centralized ADLS, we can use partitioning approach in the ADLS. Meaning that every day’s files would be stored in containers with following hierarchy name:  
Year/Month/Day

This would essentially mean that every day’s file will have its own folder. So when the ADF pipeline is running for a day(let’s say today), it will only process files which exist in the specific day’s folder. An example would be that if we run the ADF pipeline today, it will only look into files in the following folder: 2024/06/30.   
If for any reason, there is no file for today or if there has been a delay in providing the file to ADLS, then we can set email notification to inform relevant parties that there has been no data for today.

Second approach)

We create and make use of a log file or a database like Postgres DB. In both cases, we can design our ADF pipeline in a way that at its very first stages, it looks into either the logfile or the database to retrieve the latest date for which the ADF was ran successfully. If the current run of the ADF pipeline has a newer date, then the pipeline will run and once it finishes processing of data for that specific date, it inserts a new line to the logfile or a new record to the database. This line will record what is the latest date for which the date has been processed. Based on my experience, using log file has been more welcomed than using another database.

Once the above mentioned ADF is designed, we can add another step to connect to a Databricks notebook. This specific Databricks notebook, will contain the business logic and transformation logics which are needed to be applied on top of the day’s data. Obviously, it can also be done using DataFlow of ADF…but, that’s something we always try to avoid due to performance and costs. Achieving the same result in Databricks would be way easier,

Please note that the same pipeline can also be built in Synapse and then linked to PoweBI or any other visualization tool used within the team.   
  
I would personally prefer to go with first approach. It’s a pretty standard architecture across different companies and provides a great way of organizing data.