**All the artifacts are available at github -** <https://github.com/ajmistry007/mm-assessment>

In order to finish this assessment I have followed below steps :

**Environment:** I haveused Databricks to complete this assignment and used serverless compute for Notebooks and Serverless Starter Warehouse for SQL.

**Volumes/Storage:** I have used DBFS file system provided by Databricks and created below volumes to process source files:

create volume raw\_xml; - Uploaded all XML files

create volume csv\_files; - Uploaded both Claim csv files

create volume unit\_test; - Created for unit testing with 2 XML files

A screenshot of a computer

AI-generated content may be incorrect.

**Notebooks:** I have created 2 separate notebooks to ingest XML data and Claims data.

1. clinical\_data\_parsing : This notebook reads XNML data from raw\_xml volume and parse it using pyspark. There are 4 dataframes created
   1. df - first dataframe with all XML data and file path where path has patientid and documented
   2. df\_patient – this dataframe has parsed patient data with patientid
   3. df\_medications – this dataframe has parsed medications data with patientid
   4. df\_problems – this dataframe has parsed problem list data with patientid

This notebook creates 3 tables each for Patient, Medications and ProblemList data.

1. load\_claims\_data : This notebook reads claims csv files from csv\_files volume and creates 2 tables claims and rx each for one file.

A screenshot of a computer

AI-generated content may be incorrect.

**SQL for combined data :** In order to combine the data which is based on patientid/memberid in all 5 tables. I am getting counts of Medications, Problems, MedClaims and RxClaims each by patientid and joined them using patientid/memberid.

**A screenshot of a computer

AI-generated content may be incorrect.**

**Further Solution options**

Since this solution is just one time and not following ETL standards, If I would have to make it operationalize I would make below updates.

1. For XML file parsing and ingestion
   1. One option is to use Streaming (Autoloader) where these files will be copied in cloud folder (Azure DL/BLOB / AWS S3 etc) and Autoloader will keep listening to cloud folder and ingest delta files to some table with metadata information as well (FileName, Path, ModifiedDate etc.)
   2. Another option is to keep watermark in some control table (timestamp field) where the pipeline will read files from last read timestamp and update the watermark once done
2. For CSV parsing would implement similar option of keeping watermark info and loading only new files
3. Instead of using managed tables, I would leverage Delta Lake and would create external tables (Delta Tables) in **bronze** catalog. These tables will be loaded using upsert and being Delta Tables the history will be tracked by Delta log
4. In order to clean and model the data I would use **silver** catalog where this data will follow data model which can then be utilized to prepare FHIR bundles.