**Introduction**

* Problem statement: The project aims to ingest and analyse Parquet files to provide movie recommendations to users. This involves building a data pipeline to ingest, process the data, performing EDA and analysis on the processed data, building a recommendation engine based on user preferences, and providing a filter feature to sort movies based on ratings and date range. And allowing user to access all these using the Flask app.
* Technologies used: Python Flask is used as the web framework for the app, Azure Data Factory and Azure Databricks for the data pipeline and analysis. PySpark is also used for processing and analysing the data.
* Resulting Feature: User will be able to provide input files and querying the processed data, the recommendation engine based on user preferences, and the filter feature for sorting movies based on ratings and date range by using the Flask app.

**IT Stack**

1. Python Flask: Flask is a lightweight web application framework written in Python. It is designed to be easy to use and flexible, allowing developers to quickly create web applications with minimal overhead. Flask provides a simple and intuitive API for handling HTTP requests and responses, as well as support for templating, authentication, and other common web development tasks.
2. Azure Data Factory: Azure Data Factory is a cloud-based data integration service that allows users to create, schedule, and manage data pipelines that move and transform data from various sources to various destinations. It provides a wide range of connectors and integration options, as well as support for complex data transformations and processing.
3. Azure Databricks: Azure Databricks is a cloud-based data analytics platform that provides a unified analytics platform for data engineering, machine learning, and analytics. It provides a collaborative workspace for data scientists, engineers, and analysts to work together on data-driven projects. It also includes a variety of tools and libraries for data processing, machine learning, and data visualization.
4. PySpark: PySpark is the Python API for Apache Spark, an open-source big data processing framework. It provides a simple and intuitive API for working with large datasets, as well as support for distributed computing and parallel processing. PySpark allows users to work with data in various formats, including CSV, JSON, Parquet, and more, and provides a variety of tools and libraries for data processing and analysis.

Top of Form

**Data Pipeline**

* The Parquet files which are provided as input are stored in the ‘raw’ container of the Azure Blob Storage, which triggers the Azure Data Factory Pipeline. Then, the data is copied and stored in other container named ‘validated’. After the completion of the previous process, Databricks Notebook gets executed. It consists the Python code required for processing and analysing data.
* The data is read and converted into DataFrame and finally saved into the Delta Table. Delta Table is much more efficient and fast for processing large data. ACID transactions, scalable metadata management, and automatic file optimization are some of its other benefits.
* Converting Genre column: You converted the Genre column to separate columns with binary flags to enable efficient querying and analysis.

**EDA and Analysis**

* Exploratory data analysis: I used EDA techniques to analyse the processed data, such as finding the top-rated and most-watched movies in each genre and much more.
* Azure Databricks: I used Azure Databricks to perform analysis and transformation on the processed data, including the algorithms you used and the data sources you drew from.
* Queries on processed data: Users are enabled to perform common queries on the processed data using the Flask app.

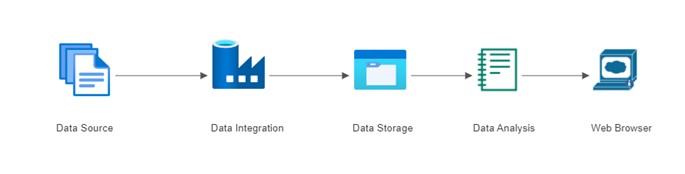
**Recommendation Engine**

* Algorithm used: Recommendation engine is based on finding out the favourite Genre of the provided UserId and based on that using data sources drawn from the processed data to recommend user top movies of that genre based on previous analysis performed i.e. of personalized movie recommendations that users can receive based on their favourite genre.

**Filter Feature**

* Sorting movies: I provided users with the ability to sort movies based on top or least ratings, watched for a specific date range.
* Examples: Suppose if user want to watch top movie of his favourite genre of year 2019, he/she can select filter as ‘Top’, criteria as ‘Watched Movie’ , starting date as ‘2019-01-01’ and ending date as ‘2019-12-31’. The user will get the top watched movie of each genre and can select the movie of his/her favourite genre.

**Process of Implementation**



* A Flask app is created and configured locally which consists of different route like Upload-files from where the user can upload the Parquet files, Download route where the processed data which is stored in the form of csv files in Azure Blob Storage gets automatically downloaded, Query route from where the user can query on the data and a display route where the queried data is displayed. The Query route contains three section, one is by querying through already made up questions, second is the recommendation section where the user can get recommendation for the specified the user and third is querying by filter, criteria and date range like – Top Rated Movie between 2018-09-21 to 2029-12-24.
* The input data which is used for this project is taken from Movielens. There are two files – ratings.csv which contains userId, movieId, ratings and timestamp. And movies.csv which consists of movieId, title and genre. The Genre column is made up by the combination of multiple single genre separated by ‘|’ as the movie may belong to more than one.
* In the Azure cloud, Azure Data Factory and Databricks is used for further processes. Once the file is uploaded on the Flask app, the files are uploaded to Azure Blob Storage through API Gateway. As the files are uploaded, the Data Factory Pipeline gets triggered. The first process is the Metadata - it checks out the schema and other details of the file. After that Copy data gets executed which copies data from the raw container of Blob Storage to validated container of Blob Storage. Once the data is copied, the Databricks Notebook which consists of the Python Script for loading, processing, analysing, and performing EDA on data.
* The parquet files are read and loaded as Spark Dataframe using PySpark. Then both the ratings dataframe and movies dataframe are joined together using the common column movieId. Then, the dataframe is saved as Delta Table in the DBFS (DataBricks File System). The saved Delta Table then loaded again as the processing of data is much fast in Delta Tables. The data is still not ready to be used for analysis as the Genre column consists combination of multiple single genre combined together by ‘|’. So, I used the pyspark functions like split and explode to first split the multiple into genres into one and then convert each genre value into row with same data of the other columns.
* After this where clause is used to flag binary data in genre columns, as if the Genre column consist that specific genre then 1 else 0.Then, duplicate values are dropped from the DataFrame and again this DataFrame is saved as Delta Table in DBFS.
* The new Delta Table is then loaded and further used for querying the data to find top or least, rated or watched, movie or genre. The insights are then saved as csv files in Blob storage, which then are downloaded through the Flask App automatically and thus provides the data for the pre-written query questions.
* For recommendation and filter query by date, separate Databricks Jobs are created which gets triggered once the form is submitted for either of them. The data provided through the HTML Form is sent as parameters in the API call and the job starts running to process the data. After processing the output is fetched using another API endpoint and displayed to the user in the web browser as HTML file using Flask.

**Conclusion**

* Flask app for querying the processed data: This allows users to perform queries on the processed data, such as finding the top-rated movies in a specific genre or finding the most-watched movies in a given date range.
* Recommendation engine based on user preferences: This allows users to input their user ID to get personalized movie recommendations based on their favourite genre.
* Filter feature for sorting movies based on ratings and date range: This allows users to filter movies based on top or least ratings, watched, or date range.