**Building an ETL Pipeline using Azure Data Services**

**Prerequisites:**

1. Create a blob storage, then create a container named “data1” and put the dataset (movies.csv) into the container as below:

Graphical user interface, text, application, email

Description automatically generated

1. Create the SQL database “appdb” and create a table named “movies” using the below query:

CREATE TABLE movies

(

film varchar(200) NULL,

genre varchar(200) NULL,

lead\_studio varchar(200) NULL,

audience\_score int NULL,

profitability real NULL,

rotten\_tomatoes int NULL,

worldwide\_gross varchar(20) NULL,

year varchar(4) NULL

);

1. Create a table named “agg\_movies” using the below query which is used to store the aggregated cleaned data:

CREATE TABLE agg\_movies

(

film varchar(200) NULL,

genre varchar(200) NULL,

lead\_studio varchar(200) NULL,

audience\_score int NULL,

profitability real NULL,

rotten\_tomatoes int NULL,

worldwide\_gross real NULL,

year varchar(4) NULL,

film\_count int NULL

);

1. Create a Data Lake Gen 2 Storage, then create a container named “movies” to store the processed data after cleaning and aggregating as shown below:

Graphical user interface, text, application

Description automatically generated

1. Create a Synapse Analytics, the create a SQL pool named “Appdemosqlpoolwithserver” and create a table named “final\_movies” using the below query:

CREATE TABLE final\_movies

(

film varchar(200) NULL,

genre varchar(200) NULL,

lead\_studio varchar(200) NULL,

audience\_score int NULL,

profitability real NULL,

rotten\_tomatoes int NULL,

worldwide\_gross real NULL,

year varchar(4) NULL,

film\_count int NULL,

rank int NULL

);

1. Create a Databricks as shown below:

Graphical user interface

Description automatically generated with medium confidence

**Data Pipeline:**

1. Create a Data Factories and click on “open Azure Data Factory Studio” as shown below:

Graphical user interface, application

Description automatically generated

1. Click on “Author” and create a pipeline named “Final project”
2. Drag and drop “Copy data” and name it as “Blob to sql”
3. Genarate a source dataset (Blob storage), then click the file format as “csv” and create a linked service for the blob storage as below

Graphical user interface, text, application, email

Description automatically generated

1. Select the file (movies.csv) and enable “First row as Header” as our data contains header, then click on OK and the source looks as below.

Graphical user interface, text, application, email

Description automatically generated

1. Generate a sink dataset (sql database appdb), then create a linked service for SQL database credentials as shown below:

Graphical user interface, application

Description automatically generated

1. Select the table which we created (movies in appdb) and click on OK. The sink dataset will be shown as below:

Graphical user interface, text, email

Description automatically generated

1. Click on “Import schemas” in mapping section. It will import schemas as below:

A screenshot of a computer

Description automatically generated with medium confidence

1. In User Properties section, click on “Auto generate”. The source and destination location will be shown as below:

Graphical user interface, text, application

Description automatically generated

1. Create a Dataflow named “CleaningAndJoiningDataToSQL”
2. Create a source named “MoviesTable” and select the Dataset as “movies” table from SQL database (appdb) using same linked service as shown below and then click on import projection in projection section:

Graphical user interface, text, application, email

Description automatically generated

1. In source option section, click on query and use the below query to remove the dollar ($) symbol and cast the worldwide\_gross column as float:

SELECT film, genre,

lead\_studio,

audience\_score,

profitability,

rotten\_tomatoes,

CAST(TRIM('$' FROM worldwide\_gross) AS real) AS wordwide\_gross,

year

from movies

1. Then, create a “Derived column” named “CleaningMoviesTable”, make Incoming stream as “MoviesTable” and create a column named “genre\_crted” and give the expression as “initCap(case(lower(genre) == 'romence','romance',case(lower(genre) == 'comdy', 'comedy', lower(genre))))” as below:

Graphical user interface, text, application, email

Description automatically generated

1. Then, create a “aggregate” named “CountofFilms”, make Incoming stream as “CleaningMoviesTable” and Group by column “genre\_crted” aggregated by count(genre\_crted) and make the column name as “filmcount” as below:

Graphical user interface, text, application, email

Description automatically generated

1. Create a filter named “MoviesWithCountofFilms”, use inner join and specify join conditions as below:

A screenshot of a computer

Description automatically generated

1. Create a select option named “FilterRequiredColumnsUsingSelect” and select the required columns to remove one of two genre column happened while joining as shown below:

Graphical user interface, text, application, email

Description automatically generated

1. Create a sink named “tosqlaggmovies” and input stream as “FilterRequiredColumnsUsingSelect” and select the SQL table “agg\_movies” using same linked service. The mapping is shown as below:

Graphical user interface, application

Description automatically generated

1. The final dataflow will look like below:

Chart

Description automatically generated

1. In pipeline, Drag and drop dataflow named “Aggregation and Moving to SQL” and this should work after the completion (success) of “Blob to sql”.
2. Drag and drop “Copy data” and name it as “SQL to ADLS”. This will be started after the completion of dataflow “Aggregation and Moving to SQL”.
3. Create a source using same linked service as shown below and use query “select distinct \* from agg\_movies” to remove the duplicate columns and to get data from table.

Graphical user interface, text, application, email

Description automatically generated

1. Create a sink dataset, then select “Azure Data Lake Storage Gen2” and select parquet format and click on continue.
2. Create a new linked service for Data Lake Gen2 as below:

Graphical user interface

Description automatically generated

1. Select the file path as container “movies” and click on OK.
2. Click on “Import schemas” and the mapping will look as below:

A screenshot of a computer

Description automatically generated with medium confidence

1. In User properties section, click on “ Auto generate” as shown below:

Graphical user interface, text, application, email

Description automatically generated

1. In Azure Databricks, click on “Launch workspace”.
2. Create a Cluster using Cluster mode as “Singel Node” and click on create as shown below and click on start cluster:

Graphical user interface, text, application

Description automatically generated

1. Create a python notebook and create a key vault for the databricks to interact with other services.
2. In notebook named “Final project”, run the below command to give access to datalake storage:

from pyspark.sql import functions as f

spark.conf.set(

"fs.azure.account.key.barathstoragedatalake.dfs.core.windows.net",

dbutils.secrets.get(scope="adlsaccess",key="barathstoragedatalake"))

1. Run this command to read data from datalake storage:

df = spark.read.format("parquet").option("header","true").load("abfss://movies@barathstoragedatalake.dfs.core.windows.net/")

1. Run this command to create a temp view and perform rank operation based on genre and store the result in rank\_df:

df.createOrReplaceTempView("agg\_movies")

rank\_df = spark.sql("SELECT distinct \*, DENSE\_RANK() OVER (PARTITION BY genre ORDER BY profitability DESC) AS rank from agg\_movies order by genre desc,rank;")

1. Run this command to write the data in Datalake storage inside movies container by creating a new folder named “final\_data”:

rank\_df.write.mode("overwrite").format("com.databricks.spark.parquet").option("header", "true").parquet("abfss://movies@barathstoragedatalake.dfs.core.windows.net/final\_data/")

1. Run this command to read data from datalake storage folder named “final data” and store the dataframe in final\_df:

final\_df = spark.read.format("parquet").option("inferSchema","true").load("abfss://movies@barathstoragedatalake.dfs.core.windows.net/final\_data/")

1. Run this command to send the data from final\_df to synapse analytics sql pool table named “final\_movies”:

tablename="final\_movies"

tmpdir="abfss://tmpdir@barathstoragedatalake.dfs.core.windows.net/movies"

# This is the connection to our Azure Synapse dedicated SQL pool

connection = "jdbc:sqlserver://dataworkspace103.sql.azuresynapse.net:1433;database=Appdemosqlpoolwithserver;user={user\_name};password={password};encrypt=true;trustServerCertificate=false;"

# We can use the write function to write to an external data store

final\_df.write.mode("append").format("com.databricks.spark.sqldw").option("url", connection).option("tempDir", tmpdir).option("forwardSparkAzureStorageCredentials", "true").option("dbTable", tablename).save()

1. Store all the commands in a single notebook and create a databricks notebook in the pipeline
2. Create a role assignment in Access control in databricks to make synapse analytics interact with each other (provide contributer role).
3. Create a linked service as below and select the existing cluster:

Graphical user interface, text, application, email

Description automatically generated

1. Select the python notebook “FinalProject”path as below:

Graphical user interface, text, application, email

Description automatically generated

1. This databricks notebook should execute after the success of the “SQL to ADLS” copy data.
2. The final pipeline will look as below:

Graphical user interface, text, application

Description automatically generated

1. Click on validate and check if there is any error and if no error occurs, then click on debug pipeline.
2. The pipeline will be succeeded as below:

Graphical user interface

Description automatically generated

1. The Final data is shown as below:
2. In SQL database, movies table:

Graphical user interface, application, table, Excel

Description automatically generated

1. In SQL database, agg\_movie table:

Graphical user interface, application, table

Description automatically generated

1. In DataLake Gen2, the files inside container look like:

Graphical user interface, text, application, email

Description automatically generated

1. In Synapse Analytics SQL pool, final\_movies table:

Table

Description automatically generated