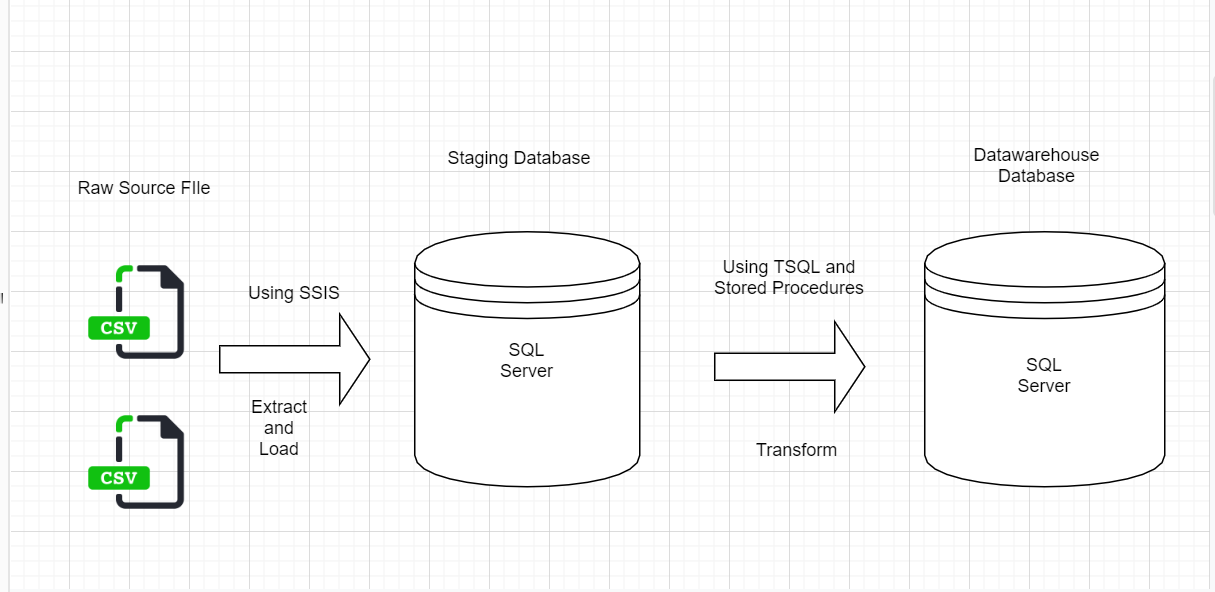
Summary of approach

I initially reviewed the dataset documentation on Kaggle to understand the business significance. I subsequently ingested the data into a staging table in a SQL Server Database using Microsoft SQL server built in ETL tool. I performed exploratory analysis on the data to determine the relationships between table, cardinality and data quality. Then I designed the data model for the Datawarehouse and populated the tables with the staging data using some transformation logic when needed. I also fine tuned the integrity constraints needed to main relational model for the Datawarehouse. Also applied performance and optimization best practice when needed to the tables like added indexes. Finally, I would create a Source to Target mapping documentation on the data flow and transformation for the support team and data lineage analysis.

ETL Architecture



STEP 1: Review Kaggle Documentation about the dataset

“The movies dataset includes 81,273 movies with attributes such as movie description, average rating, number of votes, genre, etc.

The ratings dataset includes 81,273 rating details from demographic perspective.

The names dataset includes 175,719 cast members with personal attributes such as birth details, death details, height, spouses, children, etc.

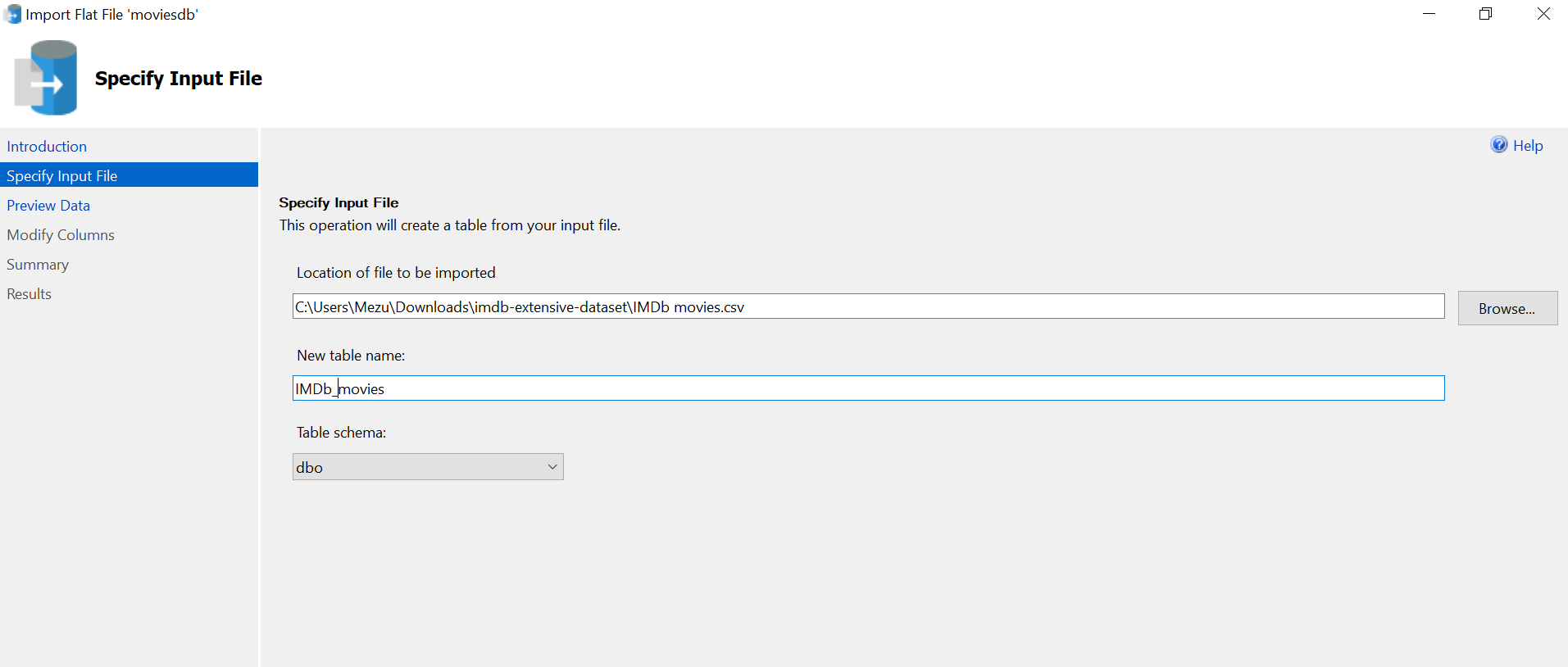
The title principals dataset includes 377,848 cast members roles in movies with attributes such as IMDb title id, IMDb name id, order of importance in the movie, role, and characters played.

”

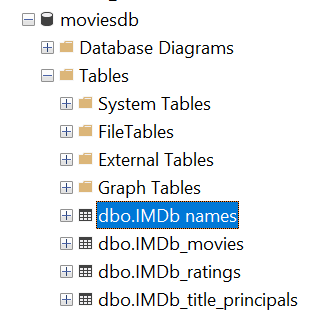
This will go into the Source Analysis documentation for the project

STEP 2: RAW SOURCE DATA INGESTION PIPELINE INTO STAGING TABLES in SQL Server Database

I would create an ELT pipeline using any available ETL tool like Microsoft SSIS, IBM Datastage or Nifi, to ingest the data into a persistent storage like a SQL Server Database since that data size is small and it’s in a structured format (csv). In this case I decide to use a built in Data Import Utility provided by MS SQL Server to import the csv files into a staging Table in a Staging database.



The goal of this step is to ingest the data in it’s raw form without transformation into a Database so I can perform detailed exploratory analysis before modeling.

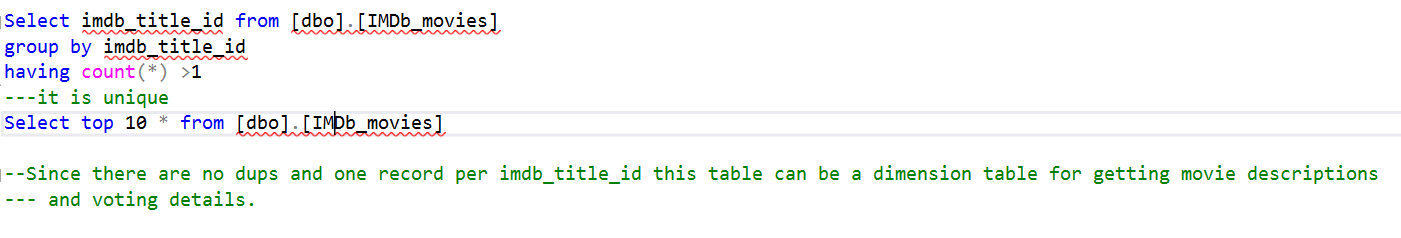


During the import, I had to ensure proper data type mapping from the csv file to Database table. Since it’s a staging table I defaulted all columns to varchar(max) (large string object) to ensure no data truncation during ingestion.

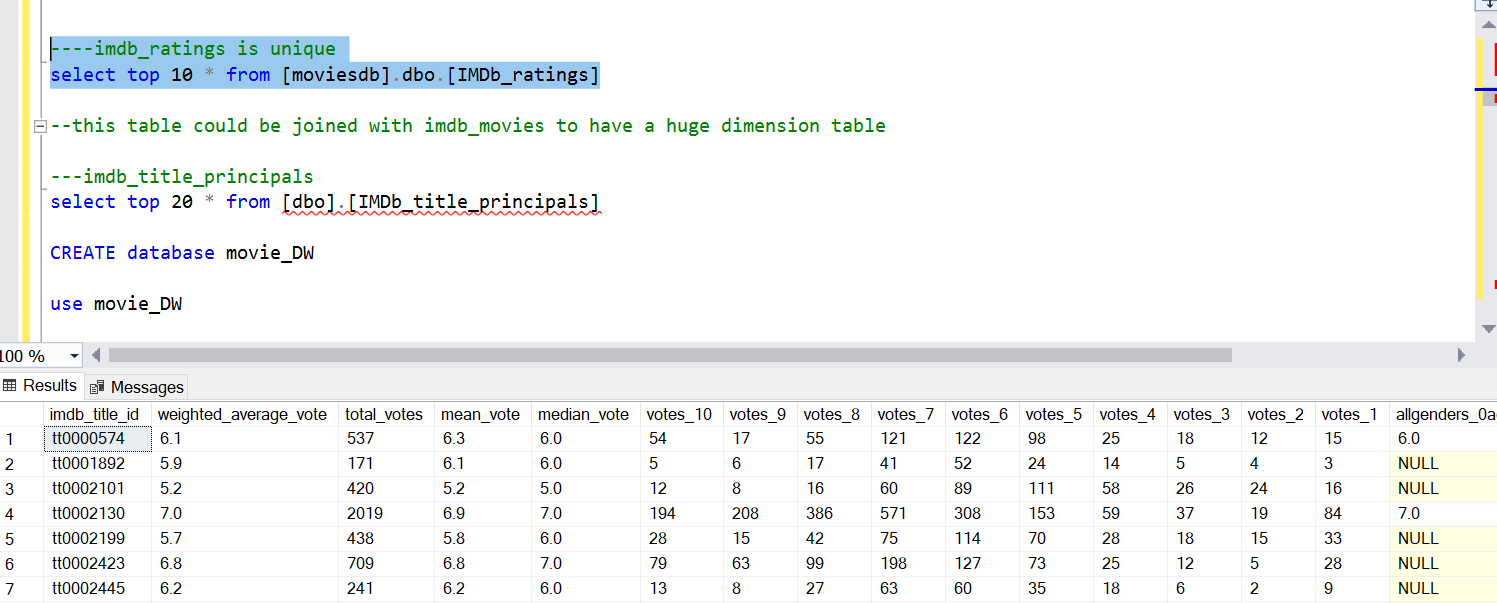
STEP 2: Exploratory Analysis

Review the top 10 records of each dataset to understand the fields and data structure.

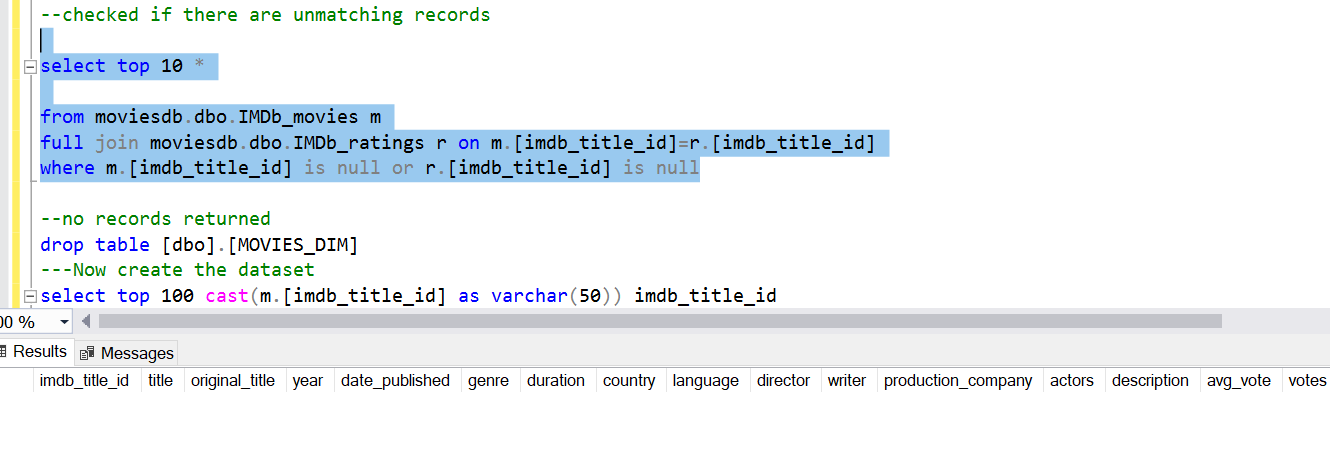
Duplicate check of all tables to determine the granuality, and data quality



Analysis of the IMDB movies dataset shows that there is only one record per imdb\_title\_id. Therefore, This table can be treated as descriptive table of the IMDB titles. Additionally, the Imdb ratings data set was analyzed and I found out it also has one record per imdb\_title\_id. This means that we can denormalize this two datasets to form one dimension table.

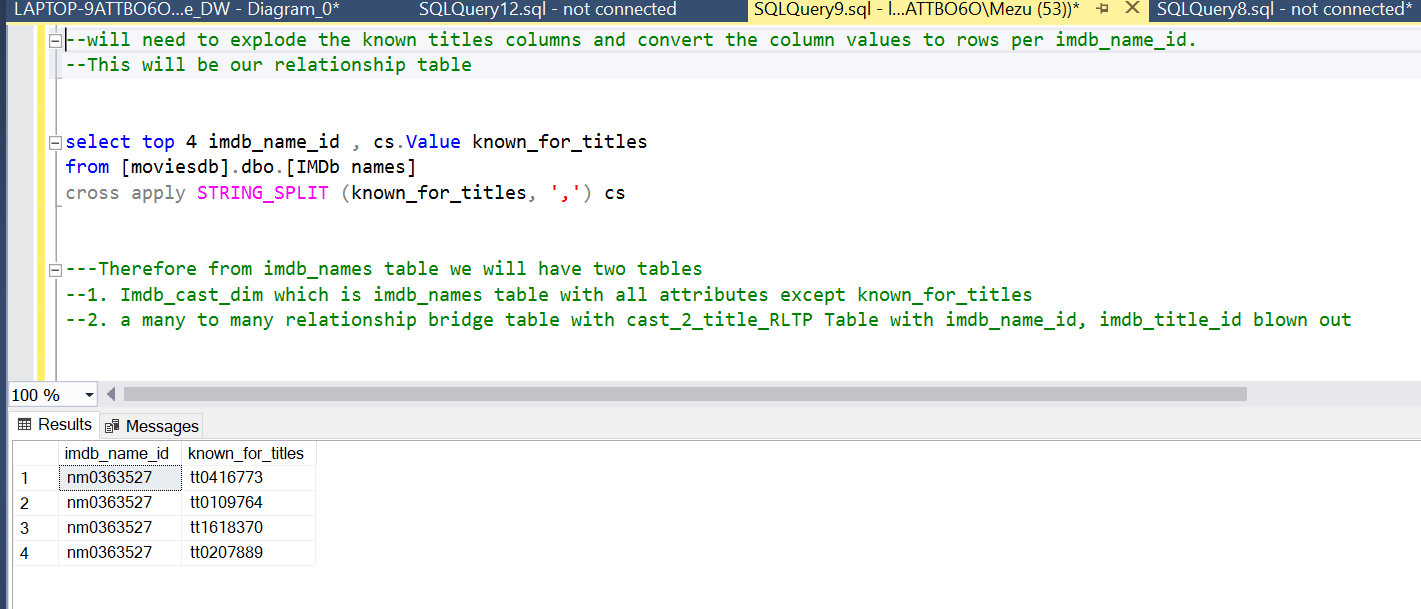


Before doing that I checked that data quality to ensure no orphan records between the two datasets. Which resulted in no records



I then analyzed the IMDB names dataset, noticed that there are no duplicate records per imdb\_name\_id

After reviewing the top 10 records, I noticed a field that ‘**known\_titles’** This field contains a comma separated values of the titles a cast member (imdb\_name\_id) featured in. This field can be “exploded” into rows and formatted for easier better analysis. The new table formed will be appended into the title principals table.



Last dataset analyzed was the imdb\_title\_principals which had a many to many relationship of the imdb\_title to the imdb\_names.

STEP 3: Transformation and Dimensional model loading

After analysis, I decided that three tables will be all that is needed to create a dimensional model for easy analysis of the data. A new datawarehouse database was created. As previously mentioned the Imdb title and ratings data will be combined to form a Dimensional table called [dbo].[MOVIES\_DIM]

Select m.\* , r. \* into movie\_DW.dbo.MOVIES\_DIM

from moviesdb.dbo.IMDb\_movies m

join moviesdb.dbo.IMDb\_ratings r on m.[imdb\_title\_id]=r.[imdb\_title\_id]

Secondly a Cast Members Dimension table is also created from the IMDb names staging table with some datatype transformation

Select top 10 cast ([imdb\_name\_id] as varchar(50)) as imdb\_name\_id

,[name]

,[birth\_name]

,[height]

,[bio]

,[birth\_details]

,[birth\_year]

,[date\_of\_birth]

,[place\_of\_birth]

,[death\_details]

,[death\_year]

,[date\_of\_death]

,[place\_of\_death]

,[reason\_of\_death]

,[spouses]

,[divorces]

,[spouses\_with\_children]

,[children]

,[primary\_profession]

,[known\_for\_titles]

into movie\_DW.dbo.Cast\_Names\_DIM

from [moviesdb].dbo.[IMDb names]

Thirdly, I added a table that is bridge table between the cast and the movie titles. This table will contain

1. Data from the IMDb\_title\_principals
2. A remainder set of imdb title ids and imdb names id that were extracted from the *exploded* imdb\_names tables that are not already in the IMDB\_title\_principals table

select cast( [imdb\_title\_id] as varchar(50)) [imdb\_title\_id]

,[ordering]

,cast([imdb\_name\_id] as varchar(50)) imdb\_name\_id

,[category]

,[job]

,[characters]

into Movies\_title\_2\_Cast\_name\_Bridge

from moviesdb.[dbo].[IMDb\_title\_principals]

union all

select cast( known\_for\_titles as varchar(50)) [imdb\_title\_id]

,null [ordering]

,cast([imdb\_name\_id] as varchar(50)) imdb\_name\_id

,null [category]

,null [job]

,null [characters]

from [moviesdb].dbo.[title\_2\_names\_brg\_remainder]

Dimensional Model

1. MOVIES\_DIM
2. Cast\_names\_DIM
3. Movies\_Title\_2\_Cast\_name\_Bridge

