Sri Lanka Institute of Information Technology

Data Warehousing and Business Intelligence

Assignment – 01

2022

IT20203412

Dinoja.N

Table of Content

DATA SET SELECTION

Data set Name	: MovieLens
Provided by	: kaggle.com
Source Link	: https://www.kaggle.com/datasets/jneupane12/movielens

About Dataset **:** The selected dataset is a collection of rating of movies.
This database describes 5-Star rating and free tagging activity from movieLens, a movie recommendation service. The dataset consists of movies released on or before 2022.This dataset also has files containing 10000 ratings from 9999 users for all 9999 movies. Ratings are on a scale of 1-5 and have been obtained from the official MovieLens website.
Users were selected at random for inclusion. All selected users had rated at least 1 movie. No demographic information is included. Each user is represented by an id, and no other information is provided.

The data contained in the files

- **movies.csv** : The main movies file. Contains information on 10000 movies feature in the full MovieLens dataset. Features include movieID, Title, Genres

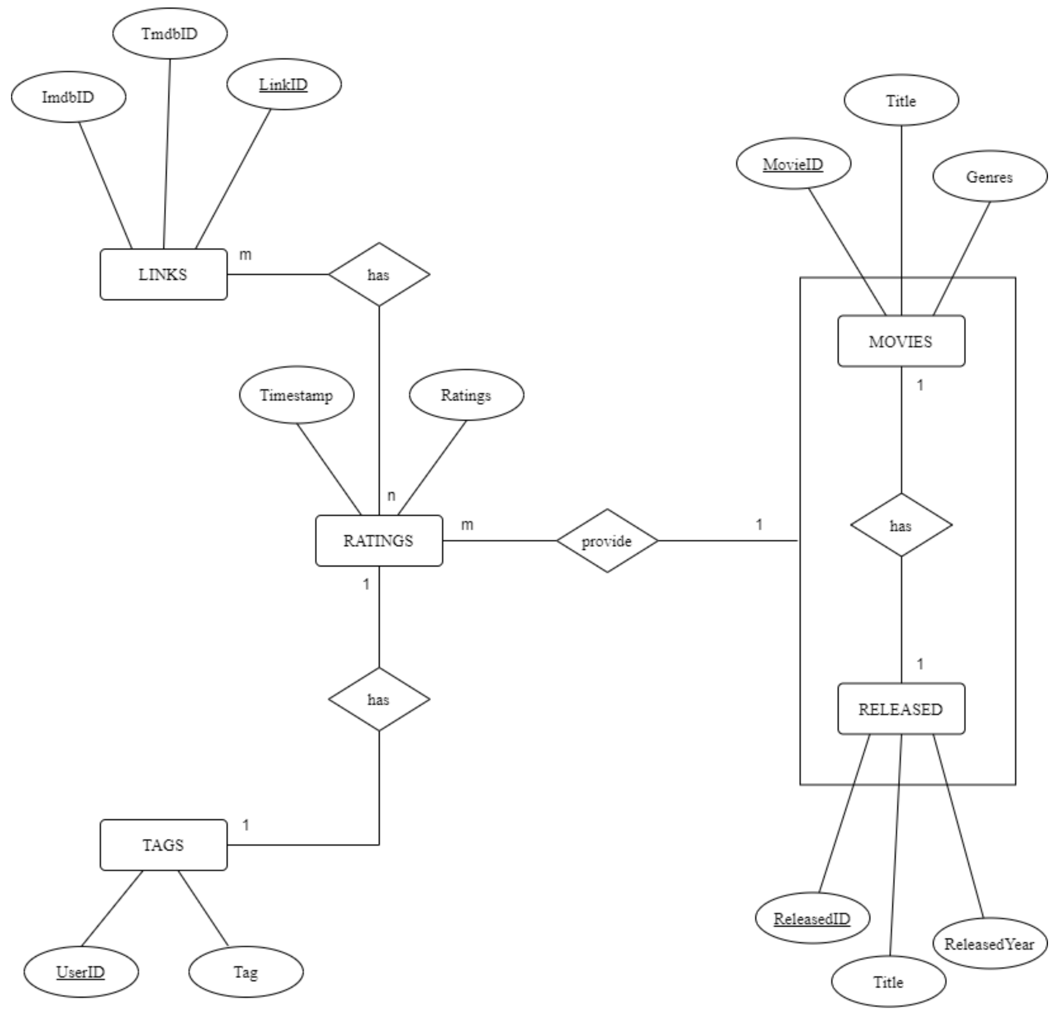
- **links.csv** : Contains the TMDB and IMDB IDs of all the movies featured in the full MovieLens dataset.

- **tags.csv** : Contains the userID and tagging of the all the users in the full MovieLens dataset.

- **ratings.csv** : The subset 10000 ratings from 9999 users on 9999 movies of the full movie dataset.

- **released.txt** : Contains the ReleasedID, Year and title features of the all movies in the full MovieLens dataset.

ER DIAGRAM



The above diagram shows the connection between the entities in the data set

- The particular transaction includes only a single item
- One movie rating can be provided by many users
- There should be one released year for a movie

PREPARATION OF DATA SOURCES

The website provides all of the data sources in csv format. Some modifications were made to the source format (some columns were added, segregated into another table) of the provided files during the preparation of data sources, such as converting text files and importing csv files into a source database.

Final state of preparation of the source data formats before transforming data

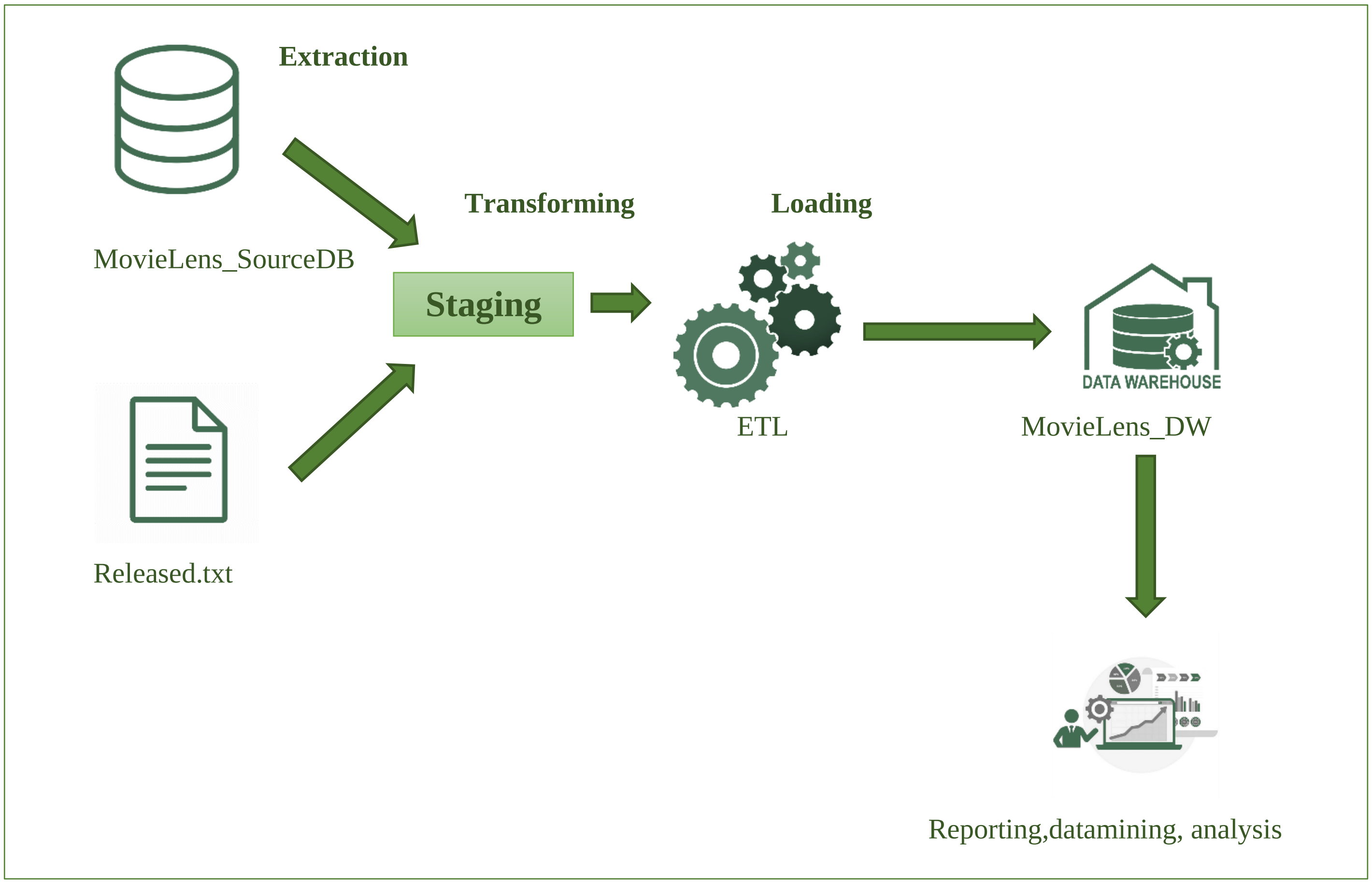
- released.txt

- MovieLens_SourceDB(Source Database) Tables:
  - dbo.links
  - dbo.movies
  - dbo.ratings
  - dbo.tag

DESCRIPTION OF THE DATA

Source Type	Table Name	Include		Description
Released.txt	Released	ReleasedID	int	Details of the movies and released details
		Title	nvarchar	
		MovieID	int	
		ReleasedYear	int	
MovieLens_SourceDB	Links	LinkID	int	Summary of the Tmdb, Imdb, link details
		ImdbID	int	
		TmdbID	int	
	Movies	MovieID	int	Summary of the movie title, genre details
		Title	varchar	
		Genres	varchar	
	Ratings	LinkID	int	Ratings summary
		UserID	int	
		MovieID	int	
		Timestamp	int	
		Rating	float	
	Tag	UserID	int	Details of the tags
		Tag	nvarchar	

SOLUTION ARCHITECTURE



As shown in the figure, MovieLens_SourceDB , Released.txt used for data extraction to the staging destination.

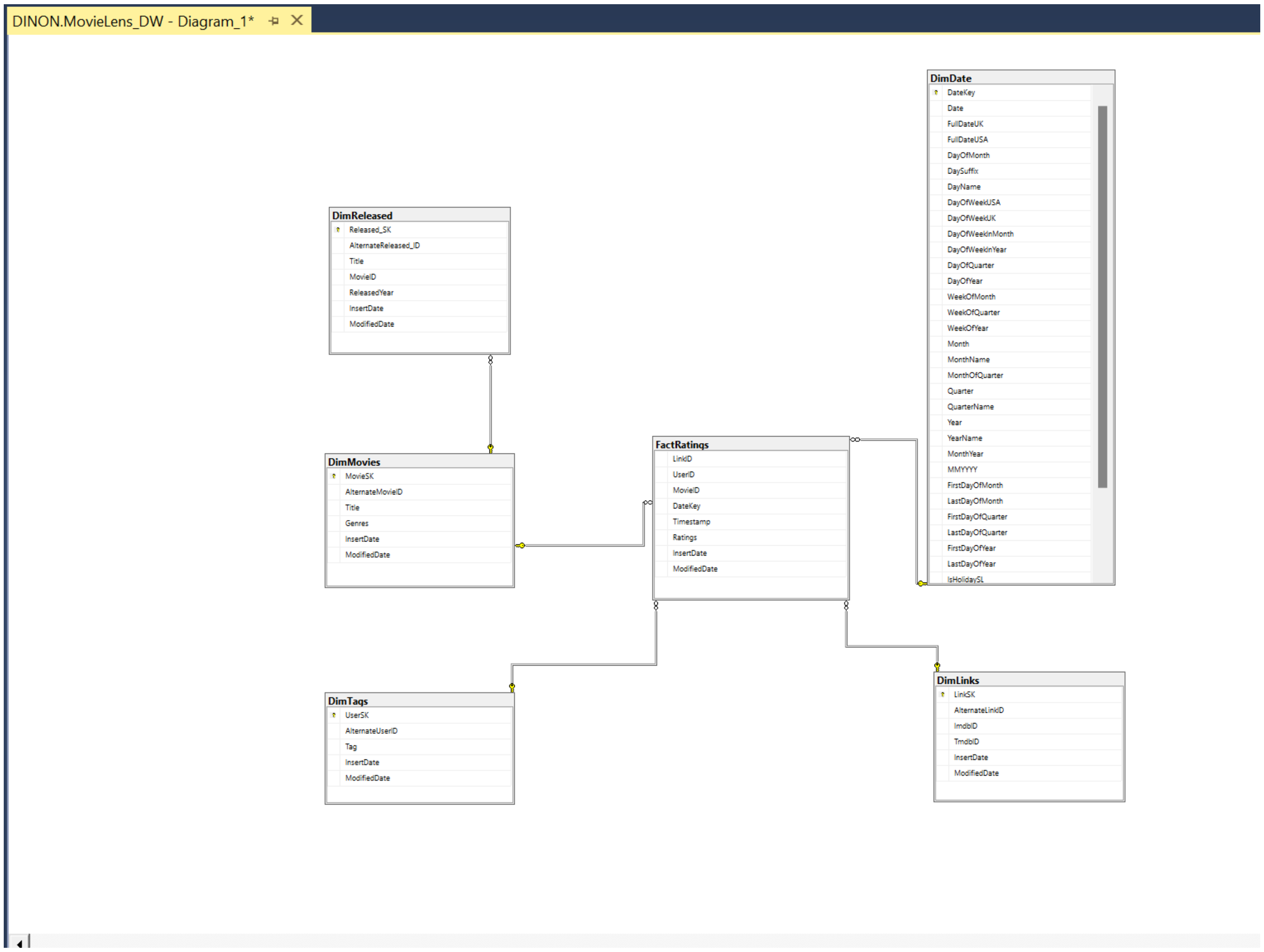
After the staging layer the below mentioned staging tables are created

- Stg.Links
- Stg.Movies
- Stg.Ratings
- Stg.Released
- Stg.Tags

Next staged tables are profiled, and aggregations are performed when necessary. As the next step data is transformed and loaded. After completing the described stages, data is tested and validated and the Datawarehouse is created.
After the warehouse is created BI results such as OLAP analysis, Reports, Data visualization, Data mining can be obtained as results after further modifications.

DATAWAREHOUSE DESIGN & DEVELOPMENT

The MovieLens_DW (ware house) is designed according to the given below snowflake schema with one fact table (dbo.FactRatings) and four dimension tables



Snowflake schema is used to design the Datawarehouse design. There is one fact table as transactions and 4 dimensions.

Slowly changing dimensions

- Movies were considered as a slowly changing dimensions

Dimension Table	Attributes
DimMovies	MovieID Title Title

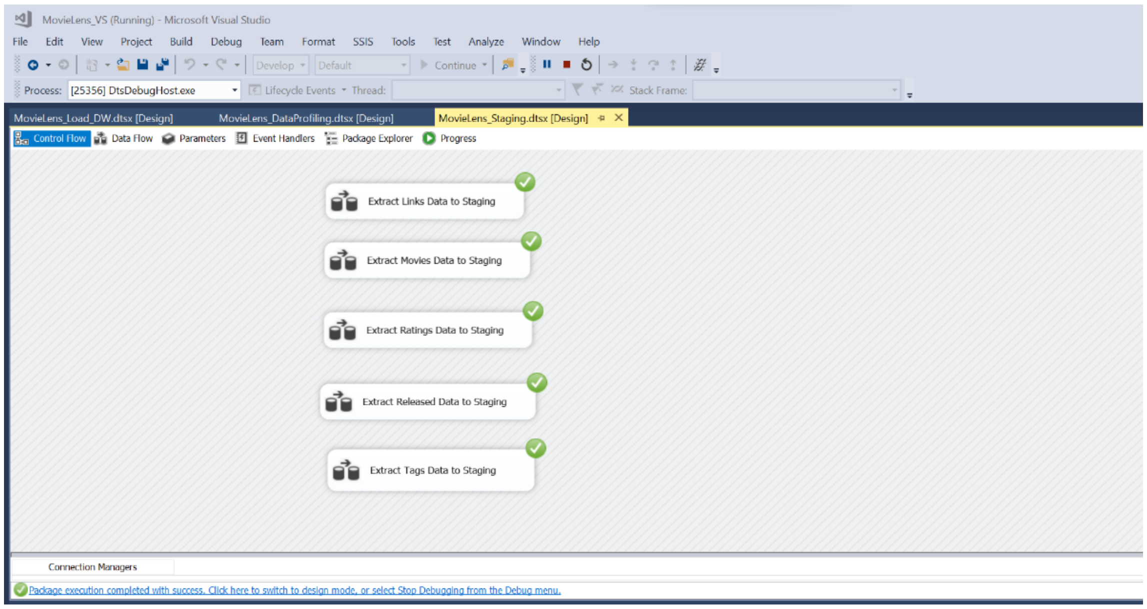
ETL DEVELOPMENT

Data extraction & load into staging table

- Data Extraction is done by using the provided data sources mentioned above in Visual Studio 2015 (Data Tool) development environment. The text file and the source database were used here.

- Initially, **OLE DB SOURCE** (for source database) or **FLAT FILE SOURCE** (for flat files txt) is used to extract data for the Staging criteria. In this step developer is able to select the columns what would be included in the Staging from available data columns. As the next step of Staging, **OLE DB DESTINATION** has applied here to storing data in the Staging tables of **AmExpert_Staging**.

- As the first step data was extracted from the sources (DB source & text file). For every extraction, data flow task was used, and data was extracted from the source to the staging table. Then for every staging table a truncate table was created. All the data flow tasks were joined as shown below at the end


MovieLens_VS (Running) - Microsoft Visual Studio
File Edit View Project Build Debug Team Format SSIS Tools Test Analyze Window Help
Develop Default Continue
Process: [25356] DtsDebugHost.exe Lifecycle Events Thread: Stack Frame:
MovieLens_Load_DW.dtsx [Design]
MovieLens_DataProfiling.dtsx [Design]
MovieLens_Staging.dtsx [Design]
Control Flow Data Flow Parameters Event Handlers Package Explorer Progress
Connection Managers

Extract Links Data to Staging

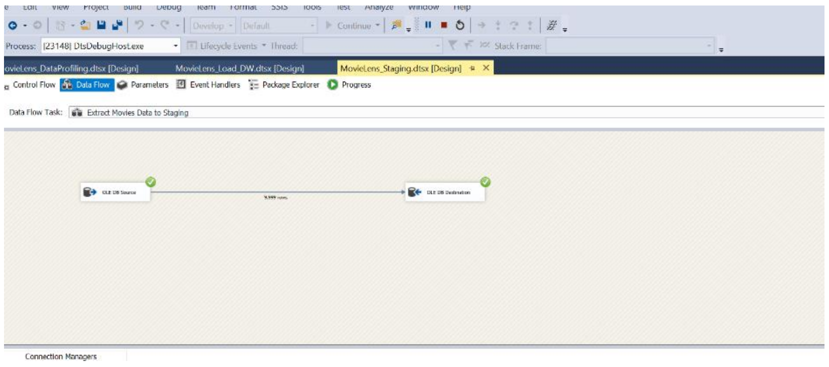
Extract Movies Data to Staging

Extract Ratings Data to Staging

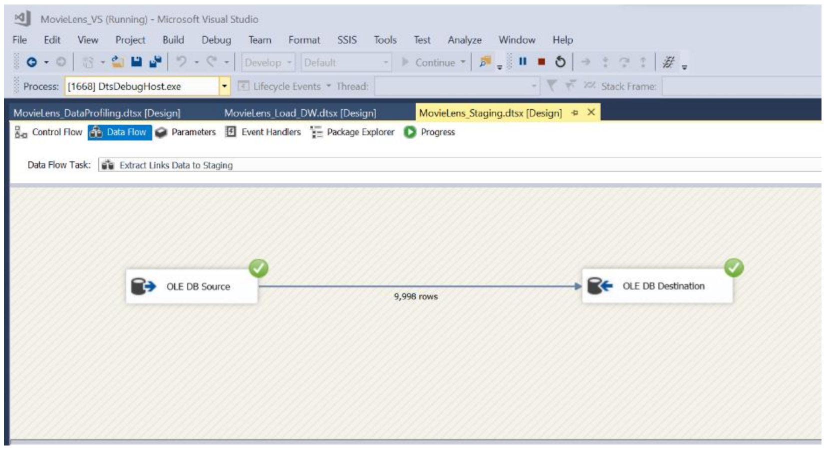
Extract Released Data to Staging

Extract Tags Data to Staging

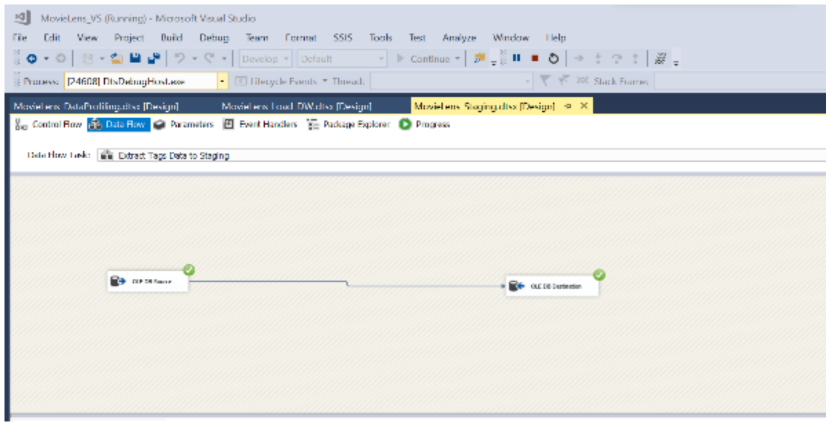
Package execution completed with success. Click here to switch to design mode, or select Stop Debugging from the Debug menu.



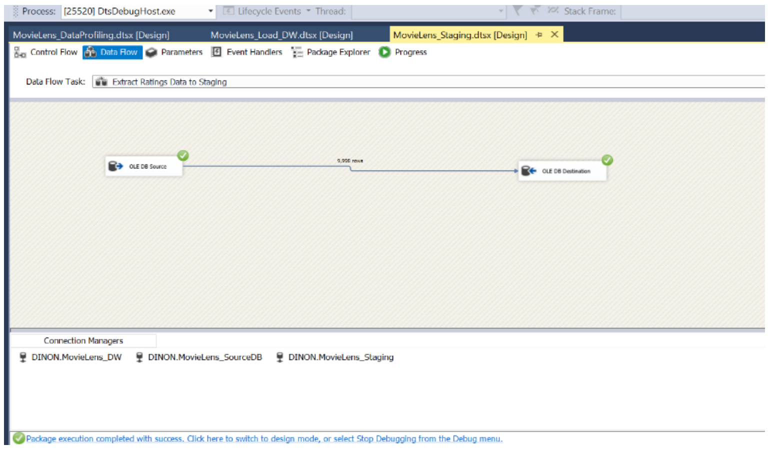
Movies data is extracted from the movie table in the source database and inserted to the StgMovies table


MovieLens_VS (Running) - Microsoft Visual Studio
File
Edit
View
Project
Build
Debug
Team
Format
SSIS
Tools
Test
Analyze
Window
Help
Process: [1668] DtsDebugHost.exe
MovieLens_DataProfiling.dtsx [Design]
MovieLens_Load_DW.dtsx [Design]
MovieLens_Staging.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task:
Extract Links Data to Staging
OLE DB Source
9,998 rows
OLE DB Destination

Links data is extracted from the movie table in the source database and inserted to the StgLinks table

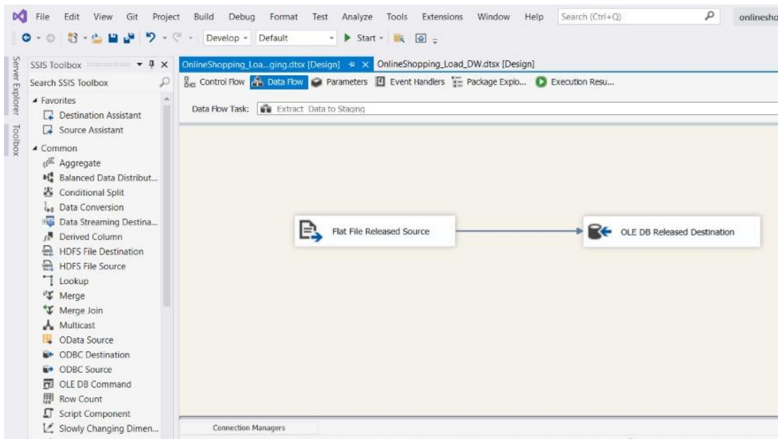

MovieLens_VS (Running) - Microsoft Visual Studio
Process: [24608] DtsDebugHost.exe
MovieLens_DataProfiling.dtsx [Design]
MovieLens_Load_DW.dtsx [Design]
MovieLens_Staging.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task:
Extract Tags Data to Staging
OLE DB Source
OLE DB Destination

Tags data is extracted from the movie table in the source database and inserted to the StgTags table

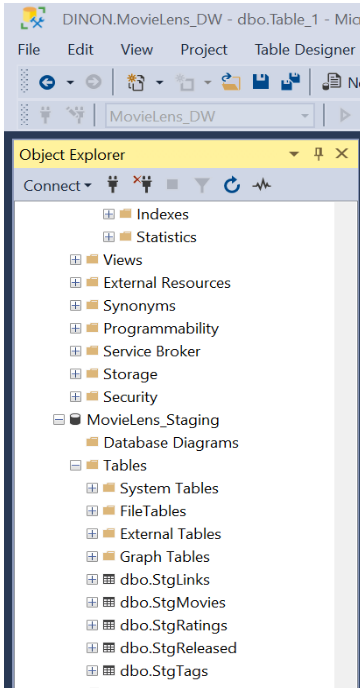


Package execution completed with success. Click here to switch to design mode, or select Stop Debugging from the Debug menu.

Ratings data is extracted from the movie table in the source database and inserted to the StgRatings table



Released data is extracted from the movie table in the source database and inserted to the StgReleased table

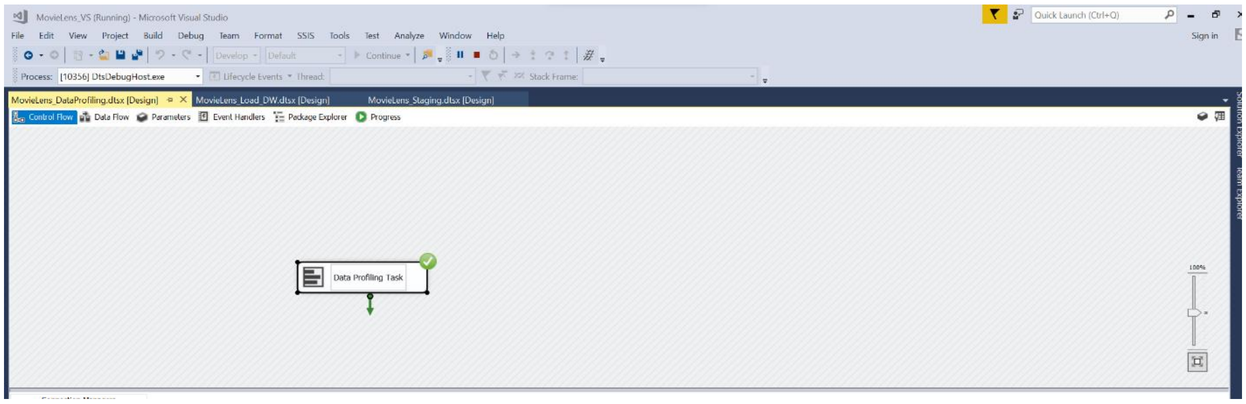


Staging Tables created and values inserted

Data profiling

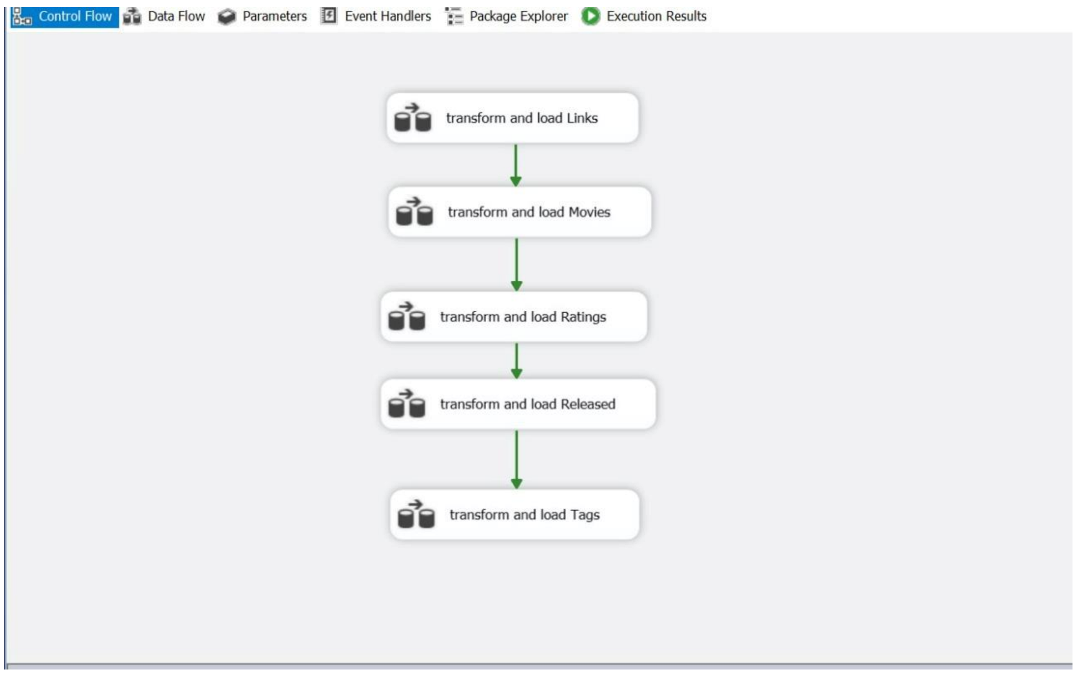
Data Profiling provides the means of analyzing large amount of data using different kind of processes. In this step, null values, repeated values and quality of the data is checked.

- Every staging table is profiled and saved in a selected location.
- As the figure shows, after the Staging step doing this task shows the things what the developer has to consider about the data which are stored in staging table and the developer is able to identify the issues with staging data by data profiling (such as null values).
- The given figure illustrated the complete part of Data Profiling relevant to the Staging.

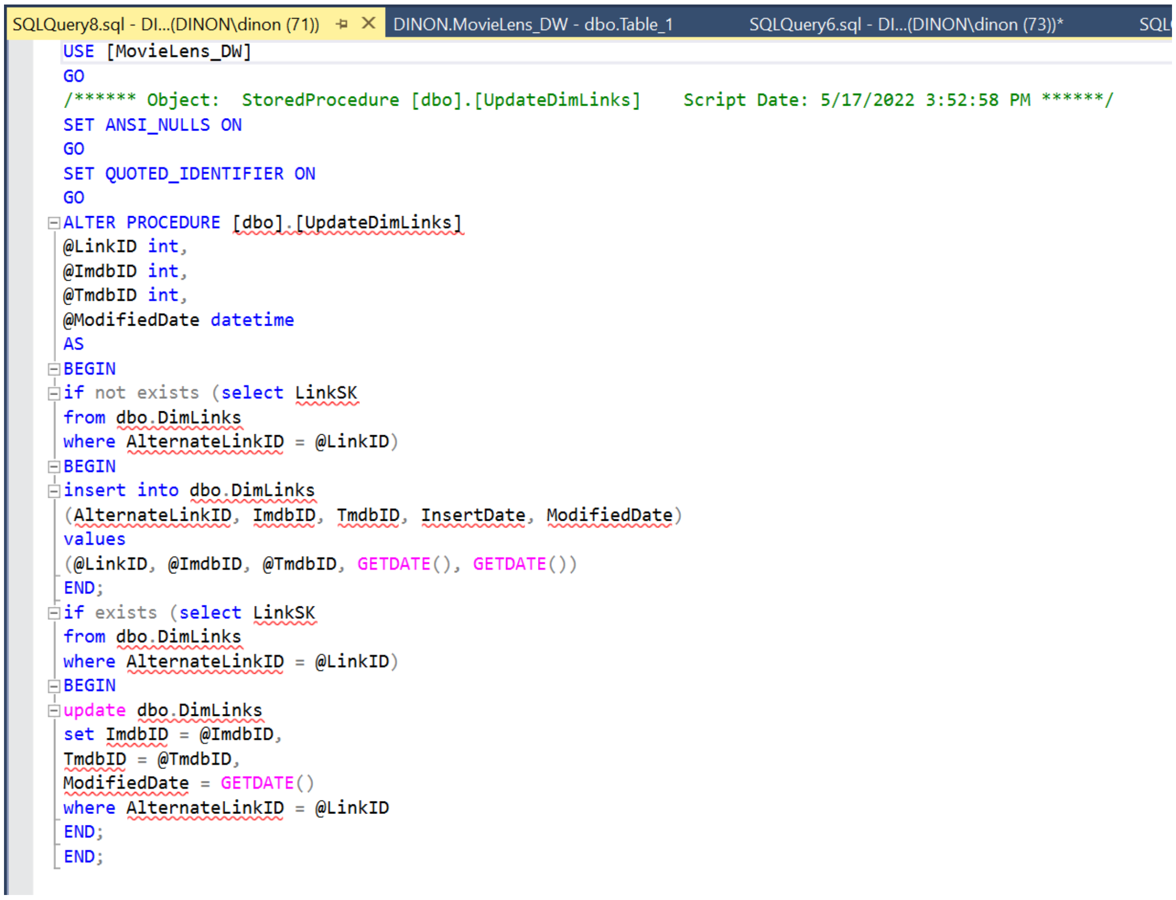

MovieLens_VS (Running) - Microsoft Visual Studio
File Edit View Project Build Debug Team Format SSIS Tools Test Analyze Window Help
Process: [10356] DtsDebugHost.exe
MovieLens_DataProfiling.dtsx [Design]
MovieLens_Load_DW.dtsx [Design]
MovieLens_Staging.dtsx [Design]
Control Flow Data Flow Parameters Event Handlers Package Explorer Progress
Data Profiling Task

Data transformation Loading

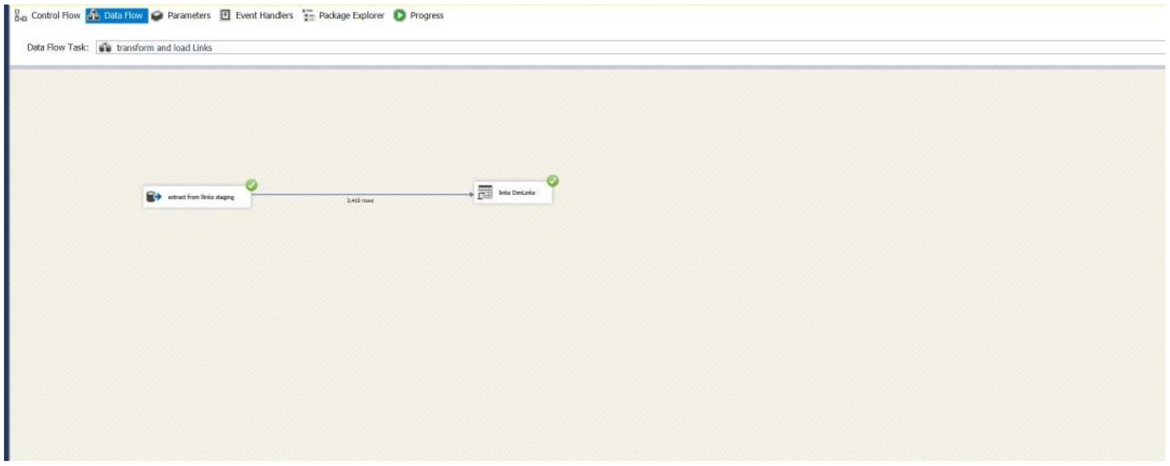
Data Transformation is developed according to the dimensional modeling designed above.



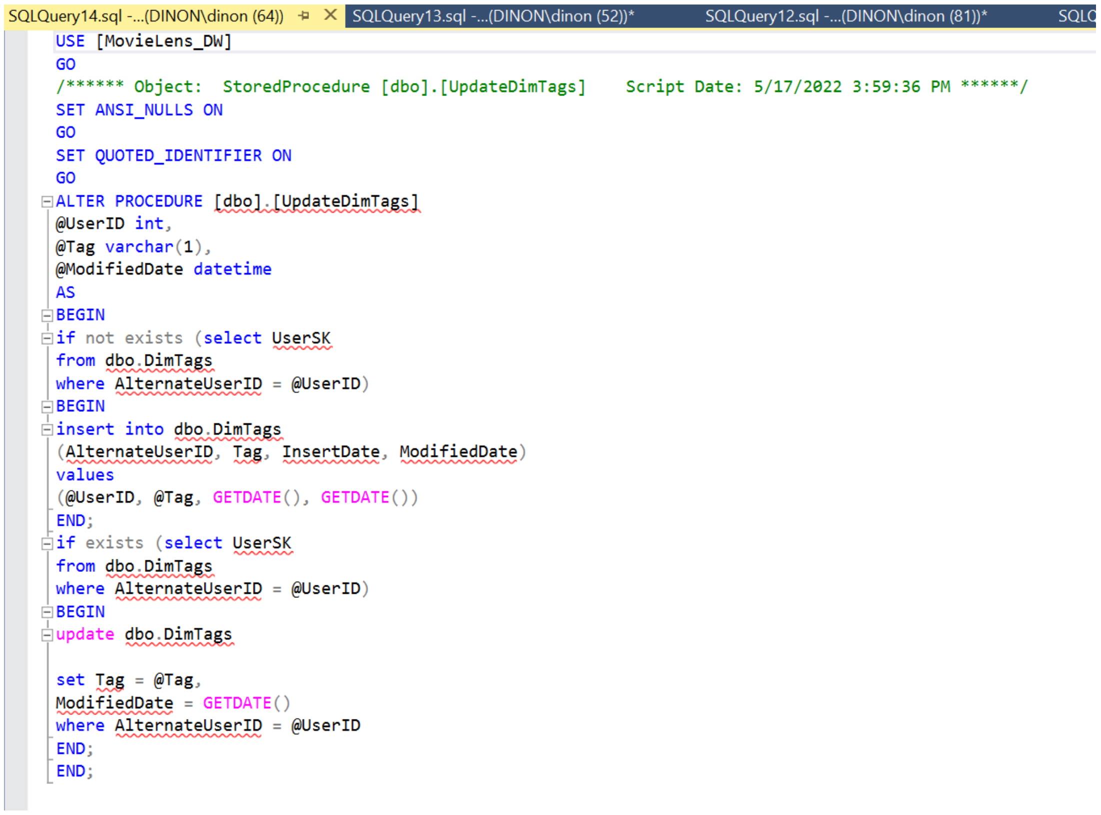
In this step, the Dimension Tables created in MovieLens_DW are loaded with the data of relevant staging tables.


SQLQuery8.sql - DI...(DINON\dinon (71))
DINON.MovieLens_DW - dbo.Table_1
SQLQuery6.sql - DI...(DINON\dinon (73))*
SQL

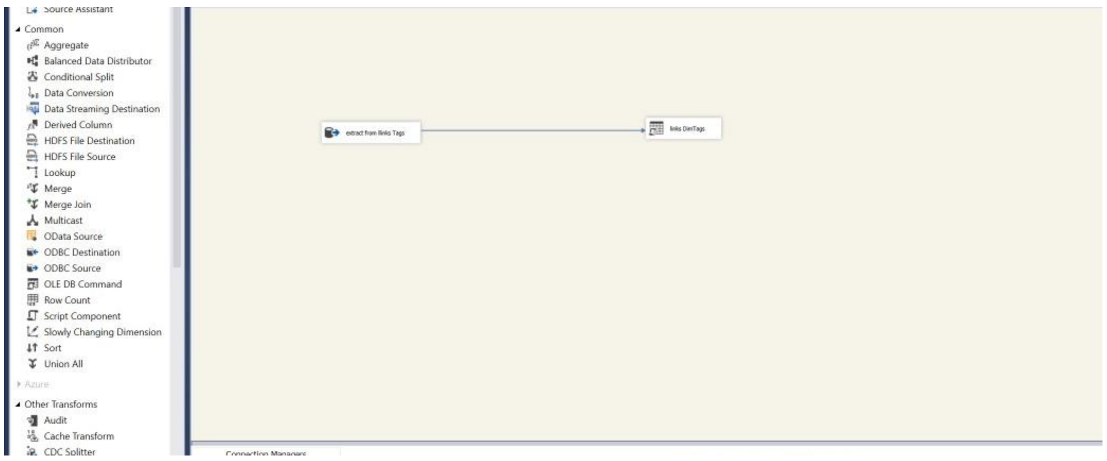
```sql
USE [MovieLens_DW]
GO
/****** Object:  StoredProcedure [dbo].[UpdateDimLinks]    Script Date: 5/17/2022 3:52:58 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdateDimLinks]
@LinkID int,
@ImdbID int,
@TmdbID int,
@ModifiedDate datetime
AS
BEGIN
if not exists (select LinkSK
from dbo.DimLinks
where AlternateLinkID = @LinkID)
BEGIN
insert into dbo.DimLinks
(AlternateLinkID, ImdbID, TmdbID, InsertDate, ModifiedDate)
values
(@LinkID, @ImdbID, @TmdbID, GETDATE(), GETDATE())
END;
if exists (select LinkSK
from dbo.DimLinks
where AlternateLinkID = @LinkID)
BEGIN
update dbo.DimLinks
set ImdbID = @ImdbID,
TmdbID = @TmdbID,
ModifiedDate = GETDATE()
where AlternateLinkID = @LinkID
END;
END;
```

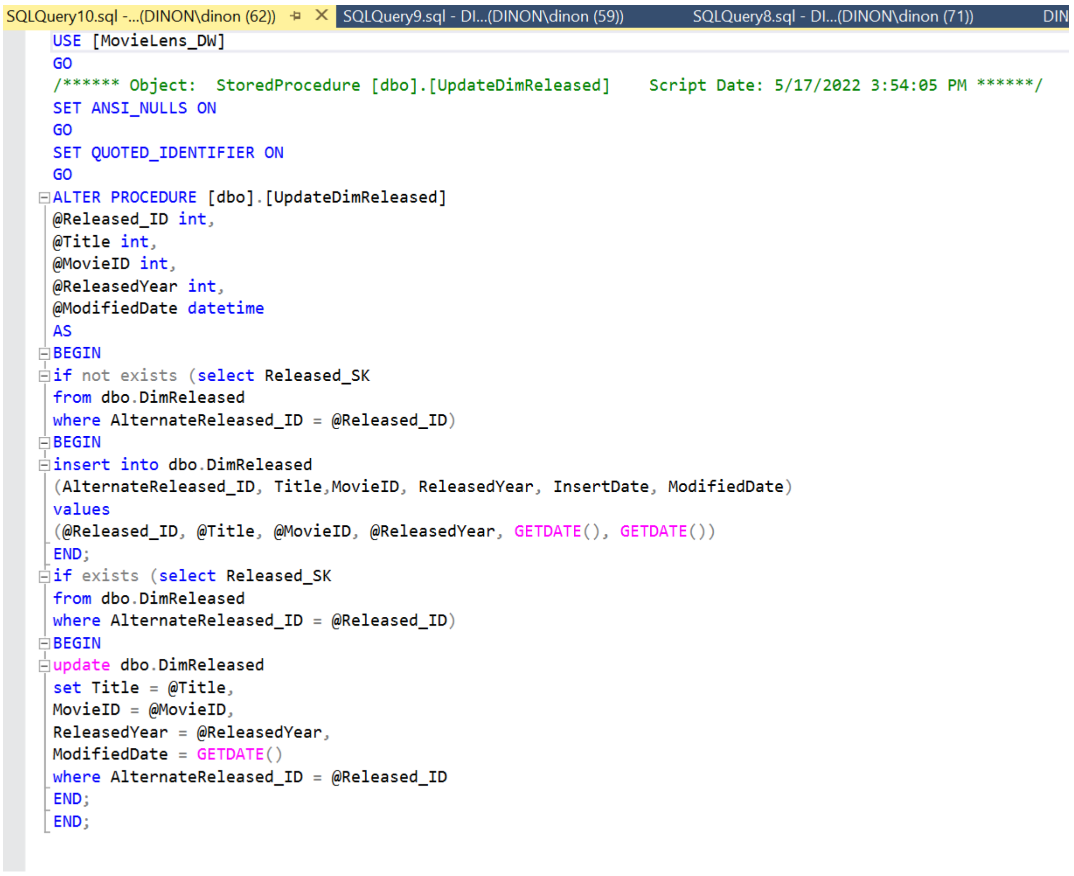


- Links data is loaded to the DimLinks
- UpdateDimLinks procedure is used to check whether the data inserted or not.

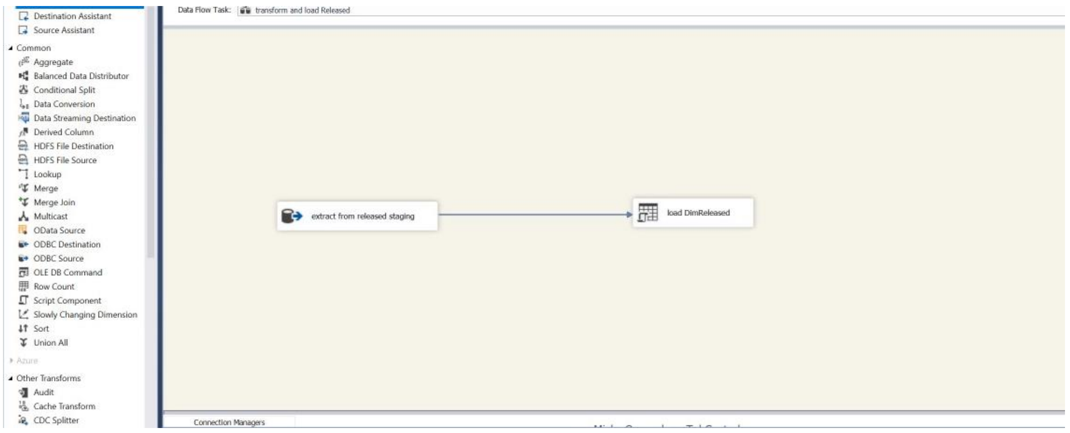

SQLQuery14.sql -...(DINON\dinon (64))
SQLQuery13.sql -...(DINON\dinon (52))*
SQLQuery12.sql -...(DINON\dinon (81))*
SQLQ

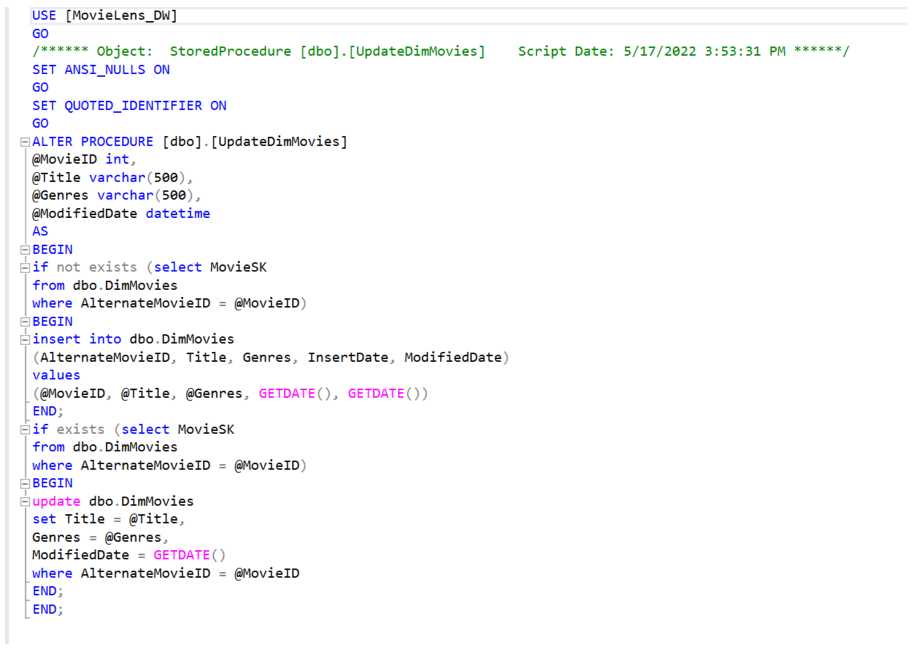
```
USE [MovieLens_DW]
GO
/****** Object:  StoredProcedure [dbo].[UpdateDimTags]    Script Date: 5/17/2022 3:59:36 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdateDimTags]
@UserID int,
@Tag varchar(1),
@ModifiedDate datetime
AS
BEGIN
if not exists (select UserSK
from dbo.DimTags
where AlternateUserID = @UserID)
BEGIN
insert into dbo.DimTags
(AlternateUserID, Tag, InsertDate, ModifiedDate)
values
(@UserID, @Tag, GETDATE(), GETDATE())
END;
if exists (select UserSK
from dbo.DimTags
where AlternateUserID = @UserID)
BEGIN
update dbo.DimTags

set Tag = @Tag,
ModifiedDate = GETDATE()
where AlternateUserID = @UserID
END;
END;
```



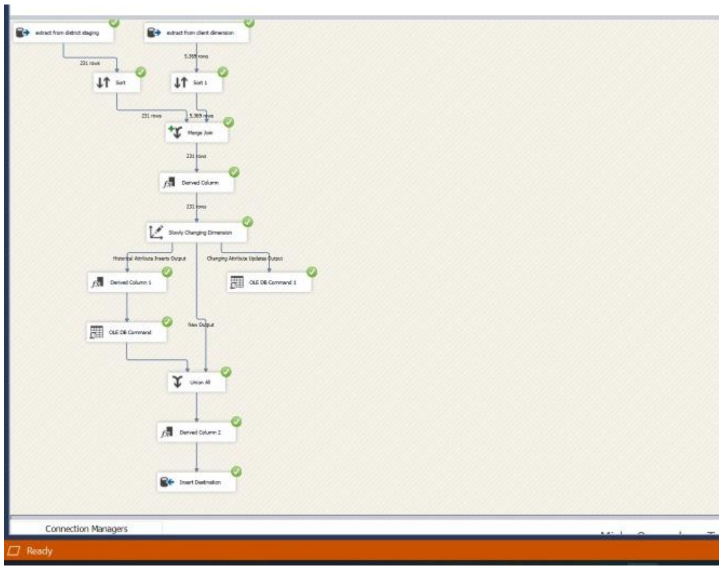


```
SQLQuery10.sql -...(DINON\dinon (62))    SQLQuery9.sql - DI...(DINON\dinon (59))    SQLQuery8.sql - DI...(DINON\dinon (71))    DIN
USE [MovieLens_DW]
GO
/****** Object:  StoredProcedure [dbo].[UpdateDimReleased]    Script Date: 5/17/2022 3:54:05 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdateDimReleased]
@Released_ID int,
@Title int,
@MovieID int,
@ReleasedYear int,
@ModifiedDate datetime
AS
BEGIN
if not exists (select Released_SK
from dbo.DimReleased
where AlternateReleased_ID = @Released_ID)
BEGIN
insert into dbo.DimReleased
(AlternateReleased_ID, Title,MovieID, ReleasedYear, InsertDate, ModifiedDate)
values
(@Released_ID, @Title, @MovieID, @ReleasedYear, GETDATE(), GETDATE())
END;
if exists (select Released_SK
from dbo.DimReleased
where AlternateReleased_ID = @Released_ID)
BEGIN
update dbo.DimReleased
set Title = @Title,
MovieID = @MovieID,
ReleasedYear = @ReleasedYear,
ModifiedDate = GETDATE()
where AlternateReleased_ID = @Released_ID
END;
END;
```





```sql
USE [MovieLens_DW]
GO
/****** Object:  StoredProcedure [dbo].[UpdateDimMovies]    Script Date: 5/17/2022 3:53:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdateDimMovies]
@MovieID int,
@Title varchar(500),
@Genres varchar(500),
@ModifiedDate datetime
AS
BEGIN
if not exists (select MovieSK
from dbo.DimMovies
where AlternateMovieID = @MovieID)
BEGIN
insert into dbo.DimMovies
(AlternateMovieID, Title, Genres, InsertDate, ModifiedDate)
values
(@MovieID, @Title, @Genres, GETDATE(), GETDATE())
END;
if exists (select MovieSK
from dbo.DimMovies
where AlternateMovieID = @MovieID)
BEGIN
update dbo.DimMovies
set Title = @Title,
Genres = @Genres,
ModifiedDate = GETDATE()
where AlternateMovieID = @MovieID
END;
END;
```



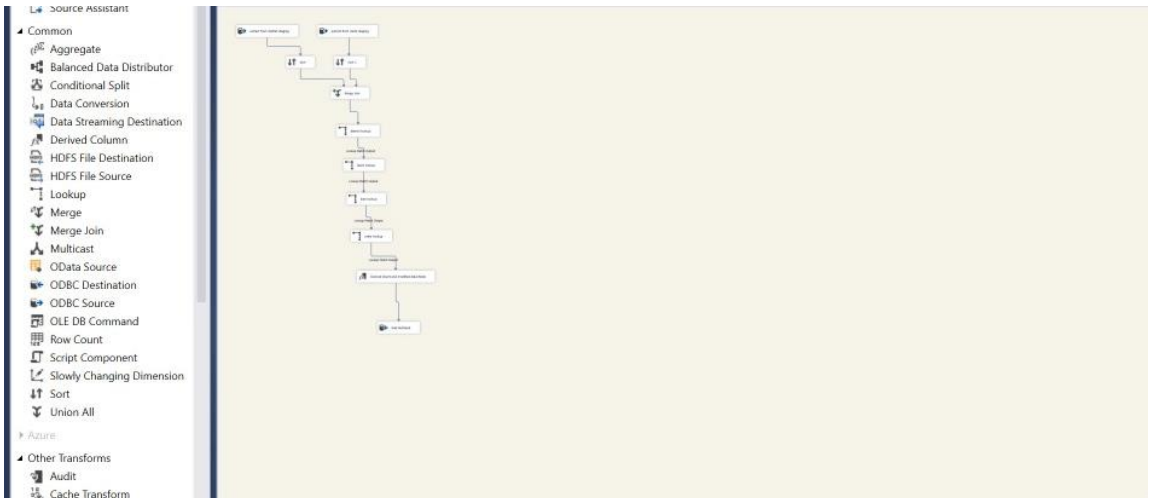
Loading slowly changing dimension

- DimMovies is the slowly changing dimension in this dimensional modeling.
- In order to load data to Dimension table, the slowly changing dimensions (historical) have two specific columns as StartDate & EndDate to ensure that the data is valid at the moment.
- slowly changing dimension wizard let the developer to select the Dimension table, Business keys of the dimension and what would be the slowly changing attributes.
- Initially data cleansing is done in order to remove null values from the data source table.

Load data to fact table

- The final step of Transformation & Loading is load data to fact table. According to the dimensional model, TransactionStaging table is used to insert values into DimTransaction table.
- FactTransaction table has one date key which are related to Date Dimension as TransactionDateKey.
- After loading to all the dimensions, lastly data was loaded to the fact table. The below steps were followed:

1. Data extracted from the Ratings transaction staging
2. Join operation is done for the date using look up.
3. Join operation is done for the ratings using look up.
4. Join operation is done for the titlemovie using look up.
5. Join operation is done for the tagdate using look up.
6. insert and modified date were derived.
7. Fact details loaded to the DimRatings table.


Source Assistant
Common
Aggregate
Balanced Data Distributor
Conditional Split
Data Conversion
Data Streaming Destination
Derived Column
HDFS File Destination
HDFS File Source
Lookup
Merge
Merge Join
Multicast
OData Source
ODBC Destination
ODBC Source
OLE DB Command
Row Count
Script Component
Slowly Changing Dimension
Sort
Union All
Azure
Other Transforms
Audit
Cache Transform