Pemba Gurung
SEIS 630, Spring 2024
Final Project – Final Draft

Final Draft

Introduction

My project topic is to work on a IMDB dataset. For this project, I am choosing at least two types of IMDB datasets (i.e., Title and rating). I am working on understanding the schema, keys, relationship, and normalization issues. If possible, I would like to use my observation regarding this dataset to achieve my pessimistic goal by implementing it to build a simple movie recommendation application.

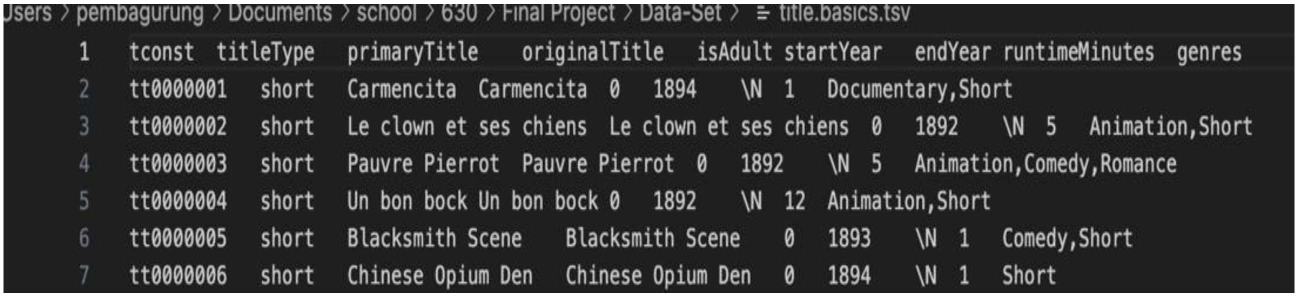
Exploring

I decided to work on the movie dataset because last winter break, I was learning and working on building a backend application where I was introduced to the movie dataset briefly. I am interested in diving deeper to understand the dataset with what I have learned in this course. One of the alternate projects I considered working on was a book dataset using a source like goodreads.com. For a project idea, I used Google and chatGPT and picked my family members' brains on some topics. The book dataset project idea was one of them.

Building

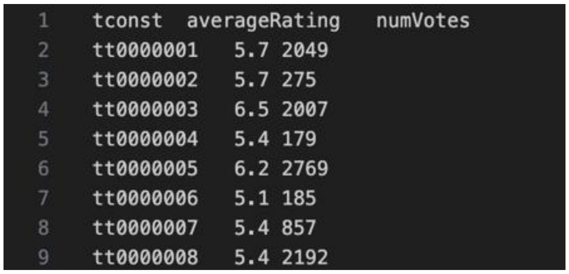
- **Findings/Observations**

Schema: I downloaded a couple of datasets from the IMDB website to study the data schema. I have used a text editor to understand the schema and relationship among the tables. The images below show data and metadata of the ‘title.tsv’ and ‘rating.tsv’ (tsv: tab separate value). Files title.tsv and rating.tsv represents each table as the title and rating table.



```
Jsers > pembagurung > Documents > school > 630 > Final Project > Data-Set > ≡ title.basics.tsv
tconst  titleType   primaryTitle   originalTitle   isAdult startYear   endYear runtimeMinutes  genres
tt0000001   short   Carmencita  Carmencita  0   1894    \N  1   Documentary,Short
tt0000002   short   Le clown et ses chiens  Le clown et ses chiens  0   1892    \N  5   Animation,Short
tt0000003   short   Pauvre Pierrot  Pauvre Pierrot  0   1892    \N  5   Animation,Comedy,Romance
tt0000004   short   Un bon bock Un bon bock 0   1892    \N  12  Animation,Short
tt0000005   short   Blacksmith Scene    Blacksmith Scene    0   1893    \N  1   Comedy,Short
tt0000006   short   Chinese Opium Den   Chinese Opium Den   0   1894    \N  1   Short
```

Snapshot: title.tsv



```
tconst	averageRating	numVotes
tt0000001	5.7	2049
tt0000002	5.7	275
tt0000003	6.5	2007
tt0000004	5.4	179
tt0000005	6.2	2769
tt0000006	5.1	185
tt0000007	5.4	857
tt0000008	5.4	2192
```

Snapshot: rating.tsv

Tables: **Title** table has columns as shown.

tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres

Data Type:

String - *tconst*, *titleType*, *primaryTitle*, *originalTitle*, *genre,*

Number - *startYear*, *endYear*, *runtimeMinutes,*

Boolean – *isAdult*

Rating table's columns as shown.

tconst	averageRating	numVotes

Data Type:

String – *tconst,*

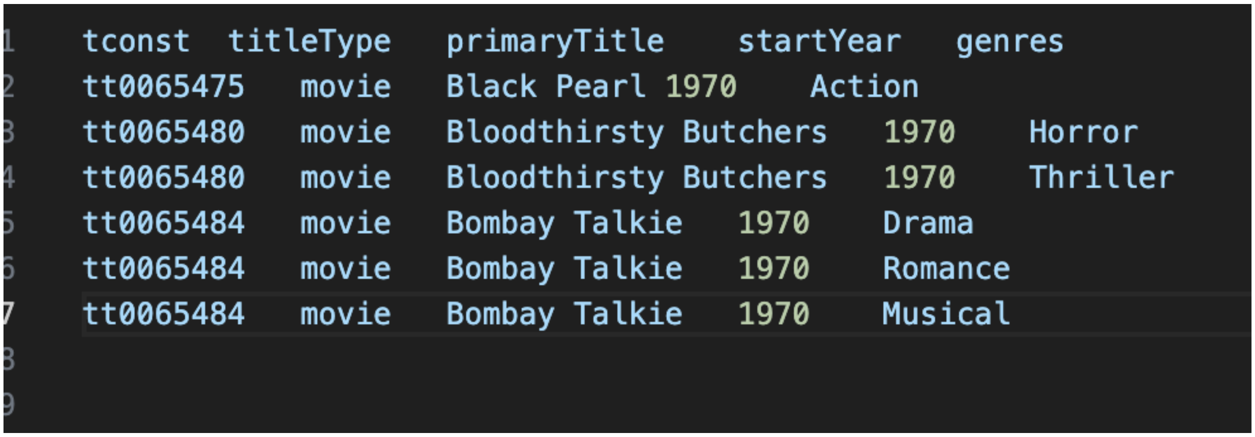
Number – *averageRating*, *numVotes*

Keys:

table	**Primary key**	**Foreign key**
title	tconst	
rating	tconst	tconst

Relationship: These two tables bear a one-to-one relationship with 'tconst' as a primary key. A record of the title table is related to either one or none of the records of the rating table and a record of the rating table is related to only one record of the title table. In this relationship, we can see that the title table serves as a parent table and the rating table serves as a child. A record in the rating table can exist only when a corresponding parent record in the title table exists. Also noticed that the table title shares the primary key with the rating table.

Normalization: In the title table, the genre column has multiple values in a cell. Multiple values in a cell breaks a rule of 1NF. According to 1NF, the table should have a single-valued cell. To normalize the data to 1NF, I must separate values from the genre column to avoid multi-valued attributes in a cell. Before I began to work on queries to modify tables to the first normal forms, to visualize the data, I formatted the data of title.tsv in a new file manually in 1NF state. I realized that if the table is in a 1NF state, there will be duplicate rows with repeated keys and non-key attributes. The table would look like shown in the snapshot below.



```
tconst  titleType   primaryTitle    startYear   genres
tt0065475   movie   Black Pearl 1970    Action
tt0065480   movie   Bloodthirsty Butchers   1970    Horror
tt0065480   movie   Bloodthirsty Butchers   1970    Thriller
tt0065484   movie   Bombay Talkie   1970    Drama
tt0065484   movie   Bombay Talkie   1970    Romance
tt0065484   movie   Bombay Talkie   1970    Musical
```

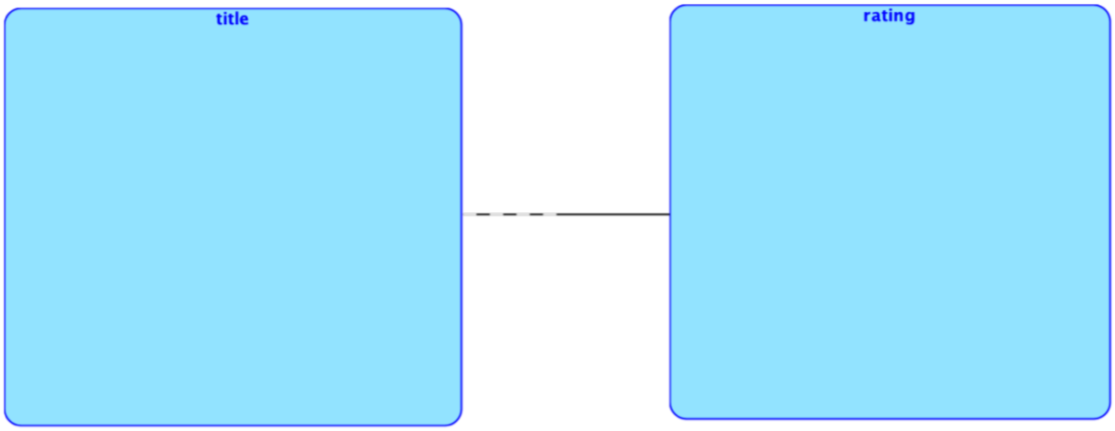
NOTE: The snapshot below does not contain all the columns from the original data file. As I mentioned, the data is filtered to achieve my ambitious goal.

This table now complies with the 1NF through 3NF definition. However, resolving multi-valued attribute issues raises multi-valued dependency issues. To fix the issue, the table needs to be in 4NF.

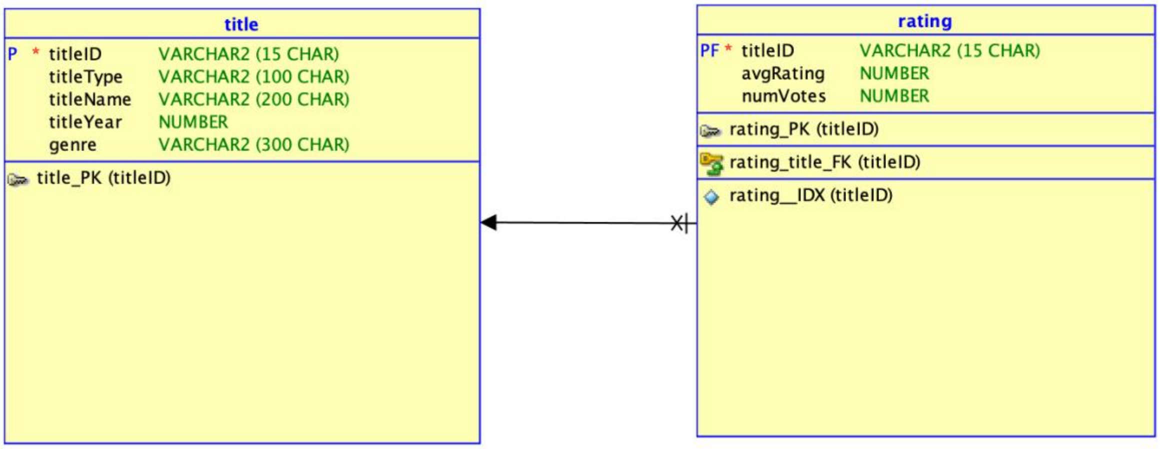
- **Building Database for IDMB dataset:**

After understanding the schema and relationship of these two tables, I used Oracle Data Modeler to create a DDL to create tables for the database. After the tables were created, I used import wizards to import the data from ‘title.tsv’ and ‘rating.tsv’.

ER-Diagram with Barker notation.



Logical Diagram



title

```
P  * titleID      VARCHAR2 (15 CHAR)
     titleType    VARCHAR2 (100 CHAR)
     titleName    VARCHAR2 (200 CHAR)
     titleYear    NUMBER
     genre        VARCHAR2 (300 CHAR)
title_PK (titleID)
```

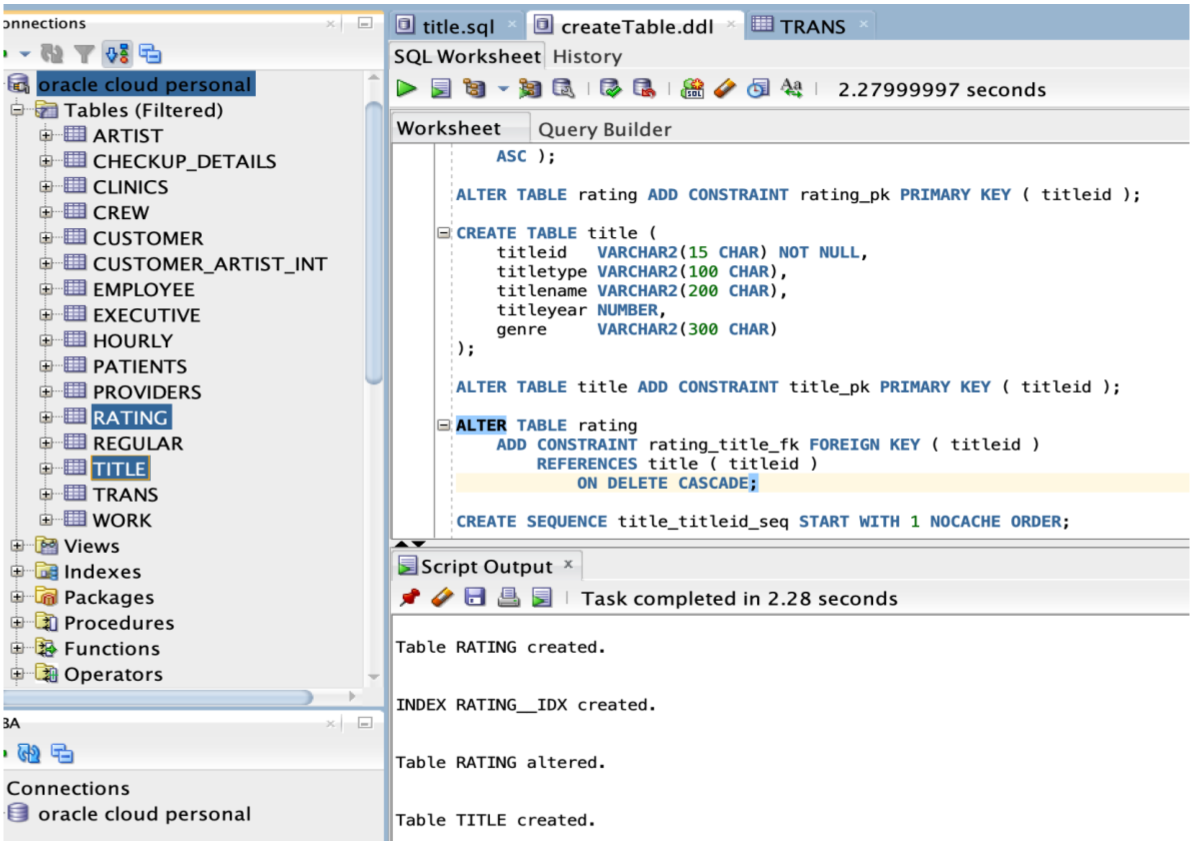
rating

```
PF * titleID      VARCHAR2 (15 CHAR)
     avgRating    NUMBER
     numVotes     NUMBER
rating_PK (titleID)
rating_title_FK (titleID)
rating__IDX (titleID)
```

Relational Diagram

The index constraint in the rating table is to prevent duplication of combination values. In addition, the index speeds up the querying performance.

Creating tables using the generated .ddl file



```
    ASC );

ALTER TABLE rating ADD CONSTRAINT rating_pk PRIMARY KEY ( titleid );

CREATE TABLE title (
    titleid   VARCHAR2(15 CHAR) NOT NULL,
    titletype VARCHAR2(100 CHAR),
    titlename VARCHAR2(200 CHAR),
    titleyear NUMBER,
    genre     VARCHAR2(300 CHAR)
);

ALTER TABLE title ADD CONSTRAINT title_pk PRIMARY KEY ( titleid );

ALTER TABLE rating
    ADD CONSTRAINT rating_title_fk FOREIGN KEY ( titleid )
        REFERENCES title ( titleid )
            ON DELETE CASCADE;

CREATE SEQUENCE title_titleid_seq START WITH 1 NOCACHE ORDER;
```

Task completed in 2.28 seconds

```
Table RATING created.


INDEX RATING__IDX created.


Table RATING altered.


Table TITLE created.
```

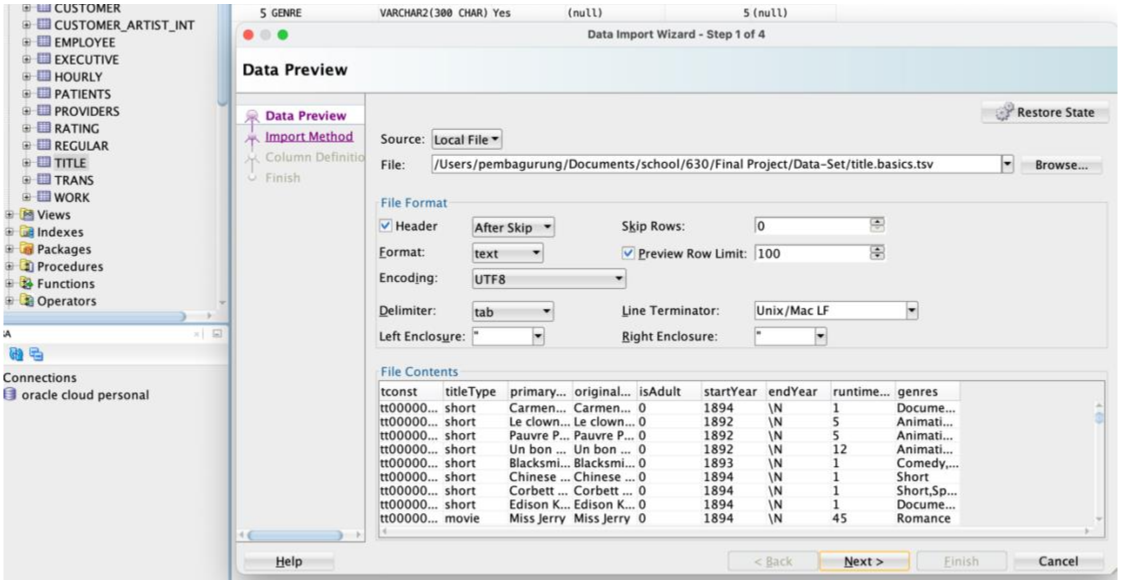
DDL script

- Database construction for IMDB dataset

```
CREATE TABLE rating (
        titleid   VARCHAR2(15 CHAR) NOT NULL,
        avgrating NUMBER,
        numvotes  NUMBER
);

CREATE UNIQUE INDEX rating__idx ON
        rating (
        titleid
ASC );

ALTER TABLE rating ADD CONSTRAINT rating_pk PRIMARY KEY ( titleid );

CREATE TABLE title (
        titleid   VARCHAR2(15 CHAR) NOT NULL,
        titletype VARCHAR2(100 CHAR),
        titlename VARCHAR2(200 CHAR),
        titleyear NUMBER,
        genre     VARCHAR2(300 CHAR)
        );

        ALTER TABLE title ADD CONSTRAINT title_pk PRIMARY KEY ( titleid );

        ALTER TABLE rating
                ADD CONSTRAINT rating_title_fk FOREIGN KEY ( titleid )
                REFERENCES title ( titleid )
                ON DELETE CASCADE;

        CREATE SEQUENCE title_titleid_seq START WITH 1 NOCACHE ORDER.

        CREATE OR REPLACE TRIGGER title_titleid_trg BEFORE
                INSERT ON title
                FOR EACH ROW
                WHEN ( new.titleid IS NULL )
        BEGIN
                :new.titleid := title_titleid_seq.nextval;
        END;
/
```

Import Data

To import data from a file, select the table where the data needs to be entered. Right-click/secondary click depending on Windows or macOS. Select import. The Data import wizards will pop up. Check the file format values are correct. For instance, header section, delimiter, etc.



5 GENRE VARCHAR2(300 CHAR) Yes (null) 5 (null)

Data Import Wizard - Step 1 of 4

Data Preview

Source: Local File

File: /Users/pembagurung/Documents/school/630/Final Project/Data-Set/title.basics.tsv

File Format

☑ Header After Skip

Skip Rows: 0

Format: text

☑ Preview Row Limit: 100

Encoding: UTF8

Delimiter: tab

Line Terminator: Unix/Mac LF

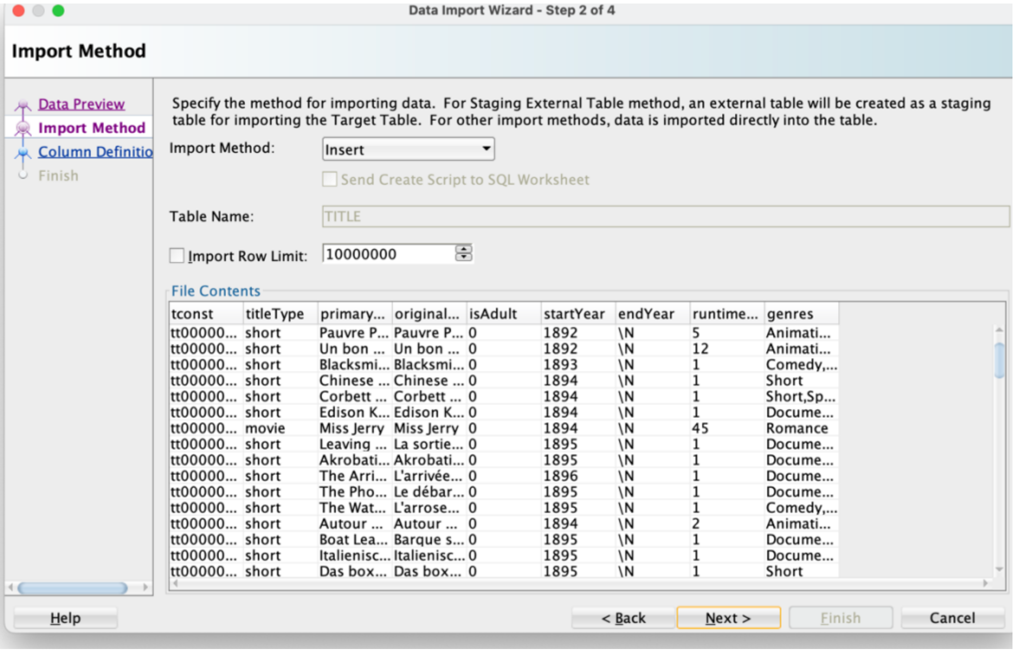
Left Enclosure: "

Right Enclosure: "

File Contents

tconst	titleType	primary...	original...	isAdult	startYear	endYear	runtime...	genres
tt00000...	short	Carmen...	Carmen...	0	1894	\N	1	Docume...
tt00000...	short	Le clown...	Le clown...	0	1892	\N	5	Animati...
tt00000...	short	Pauvre P...	Pauvre P...	0	1892	\N	5	Animati...
tt00000...	short	Un bon ...	Un bon ...	0	1892	\N	12	Animati...
tt00000...	short	Blacksmi...	Blacksmi...	0	1893	\N	1	Comedy,...
tt00000...	short	Chinese ...	Chinese ...	0	1894	\N	1	Short
tt00000...	short	Corbett ...	Corbett ...	0	1894	\N	1	Short,Sp...
tt00000...	short	Edison K...	Edison K...	0	1894	\N	1	Docume...
tt00000...	movie	Miss Jerry	Miss Jerry	0	1894	\N	45	Romance

Click next and then select import method and import row limit.



Data Import Wizard - Step 2 of 4

Import Method

Specify the method for importing data. For Staging External Table method, an external table will be created as a staging table for importing the Target Table. For other import methods, data is imported directly into the table.

Import Method: Insert

☐ Send Create Script to SQL Worksheet

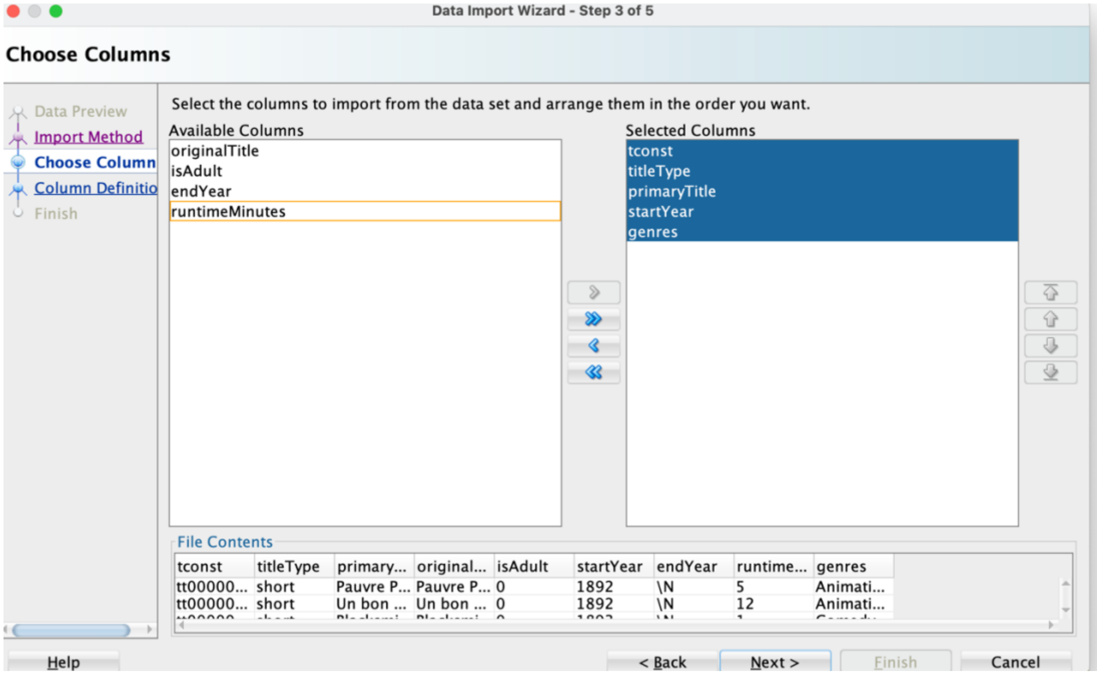
Table Name: TITLE

☐ Import Row Limit: 10000000

File Contents

tconst	titleType	primary...	original...	isAdult	startYear	endYear	runtime...	genres
tt00000...	short	Pauvre P...	Pauvre P...	0	1892	\N	5	Animati...
tt00000...	short	Un bon ...	Un bon ...	0	1892	\N	12	Animati...
tt00000...	short	Blacksmi...	Blacksmi...	0	1893	\N	1	Comedy,...
tt00000...	short	Chinese ...	Chinese ...	0	1894	\N	1	Short
tt00000...	short	Corbett ...	Corbett ...	0	1894	\N	1	Short,Sp...
tt00000...	short	Edison K...	Edison K...	0	1894	\N	1	Docume...
tt00000...	movie	Miss Jerry	Miss Jerry	0	1894	\N	45	Romance
tt00000...	short	Leaving ...	La sortie...	0	1895	\N	1	Docume...
tt00000...	short	Akrobati...	Akrobati...	0	1895	\N	1	Docume...
tt00000...	short	The Arri...	L'arrivée...	0	1896	\N	1	Docume...
tt00000...	short	The Pho...	Le débar...	0	1895	\N	1	Docume...
tt00000...	short	The Wat...	L'arrose...	0	1895	\N	1	Comedy,...
tt00000...	short	Autour ...	Autour ...	0	1894	\N	2	Animati...
tt00000...	short	Boat Lea...	Barque s...	0	1895	\N	1	Docume...
tt00000...	short	Italienisc...	Italienisc...	0	1895	\N	1	Docume...
tt00000...	short	Das box...	Das box...	0	1895	\N	1	Short

Click next and select fields to enter data into the appropriate fields. Box on the right side are the selected field's names of the data file. For my "ambitious goal" for this project, I would like to filter columns for title table down to five columns from nine column of original data file.



Data Import Wizard - Step 3 of 5

Choose Columns

Select the columns to import from the data set and arrange them in the order you want.

Available Columns:
originalTitle
isAdult
endYear
runtimeMinutes

Selected Columns:
tconst
titleType
primaryTitle
startYear
genres

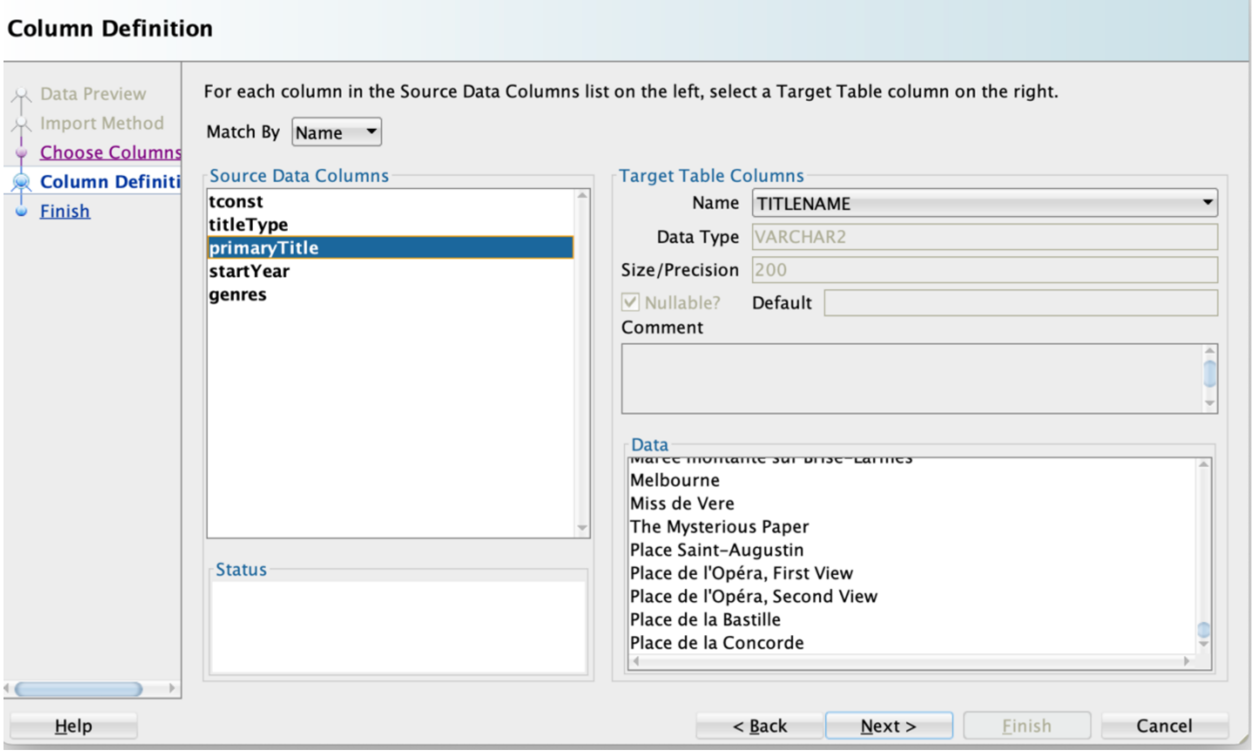
File Contents

tconst	titleType	primary...	original...	isAdult	startYear	endYear	runtime...	genres
tt00000...	short	Pauvre P...	Pauvre P...	0	1892	\N	5	Animati...
tt00000...	short	Un bon ...	Un bon ...	0	1892	\N	12	Animati...

Click next and map the corresponding columns from data file to table fields. Select each column in source data column and match with appropriate target table column.

If there is red or yellow mark on a source data column, check the specific column in the target column and map the columns again.

If that doesn't work, click on back button, and click next button again.



Column Definition

For each column in the Source Data Columns list on the left, select a Target Table column on the right.

Match By: Name

Source Data Columns:
tconst
titleType
primaryTitle
startYear
genres

Status

Target Table Columns
Name: TITLENAME
Data Type: VARCHAR2
Size/Precision: 200
☑ Nullable? Default:
Comment:

Data
Melbourne
Miss de Vere
The Mysterious Paper
Place Saint-Augustin
Place de l'Opéra, First View
Place de l'Opéra, Second View
Place de la Bastille
Place de la Concorde

Click next and finish. If there's an error import wizard will show errors message and action to be taken to resolve error. Import wizard will import the data otherwise.

Running the SQL commands in SQLDeveloper:

I used a data modeler application to engineer the DDL, but to allow the import wizard to insert data straight from the file (csv, tsv, etc.) in the correct columns in SQLDeveloper and to prevent the complication, I had to write DDL manually.

I tried running queries like create, insert, select, alter, and delete against the imported data. Some of the queries I ran against the database are listed below.

1)
```
/* Display all the column from title */
select * from title;
```

2)
```
/* Display title id, name and year of movies released in 2018 */
select titleid, titlename, titleyear from title
where titletype = 'movie' AND titleyear = 2018;
```

3)
```
/* Find the movie 'The Shashank Redemption' and display single column of title_name*/
select titlename from title
where titlename ='The Shawshank Redemption';
```

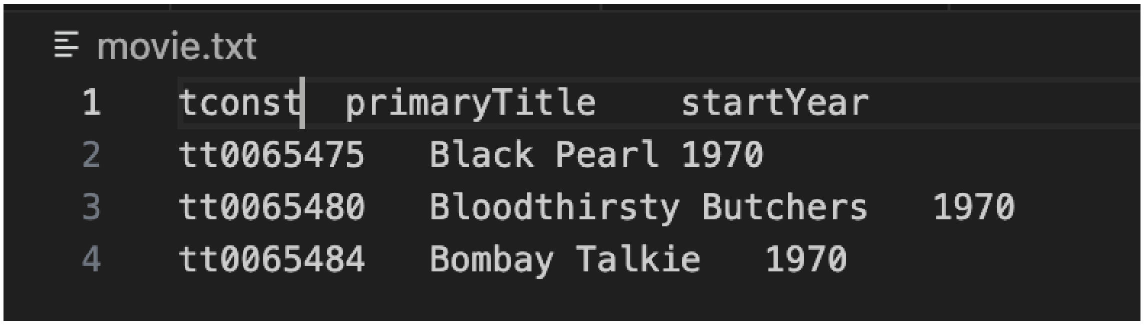
4)
```
/* Display all the column from rating*/
select * from rating;
```

5)
```
/* Show list of comedy movies that has IMDB scores greater than 8*/
select titlename from title
inner join rating
on title.titleid = rating.titleid
where title.titletype='movie' and rating.avgRating > 8 and title.genre = 'Comedy';
```

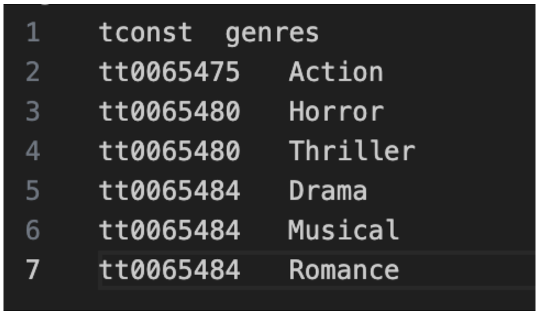
6)
```
/* Show list of movies name and year in ascending order that has more than 50000 votes and
rating of more than 8*/
select titlename, titleyear from title
inner join rating
on title.titleid = rating.titleid
where title.titletype='movie' and rating.avgRating >8 and rating.numVotes > 50000
order by
title.titleyear ASC;
```

- **Normal Form (**1NF and 4NF)

To resolve the issue, the title table can be decomposed into movie and genre tables as shown in snapshots.


movie.txt

```
tconst  primaryTitle    startYear
tt0065475   Black Pearl 1970
tt0065480   Bloodthirsty Butchers   1970
tt0065484   Bombay Talkie   1970
```



```
tconst  genres
tt0065475   Action
tt0065480   Horror
tt0065480   Thriller
tt0065484   Drama
tt0065484   Musical
tt0065484   Romance
```

Designed database schema using data modeler.

To bring the old database (title table) to 1NF and then 4NF, I decomposed the title table to movie and genre tables. The information engineering notation is used for ER diagram.

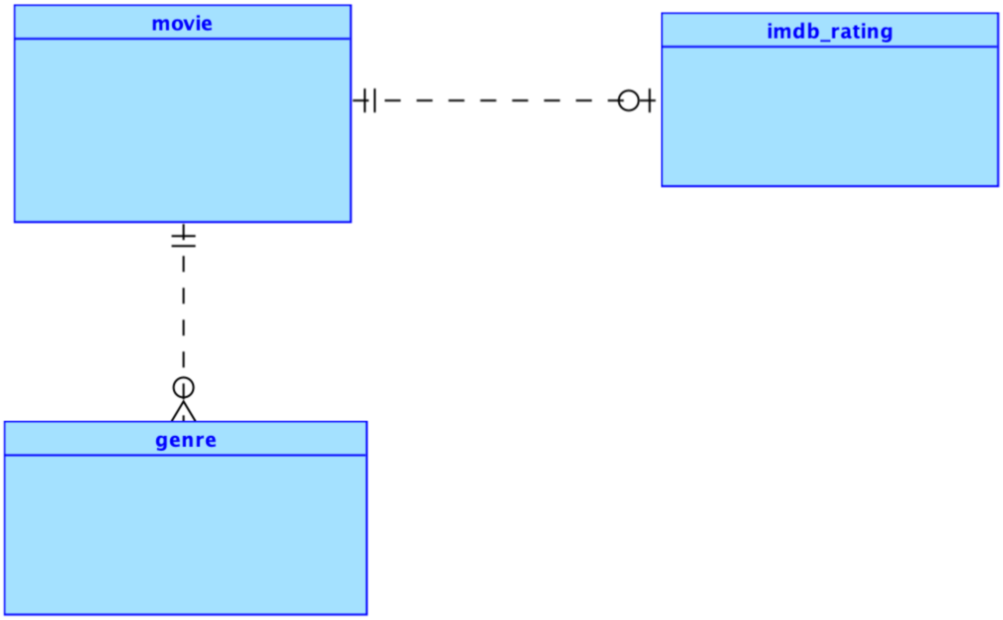


Figure: Logical Model

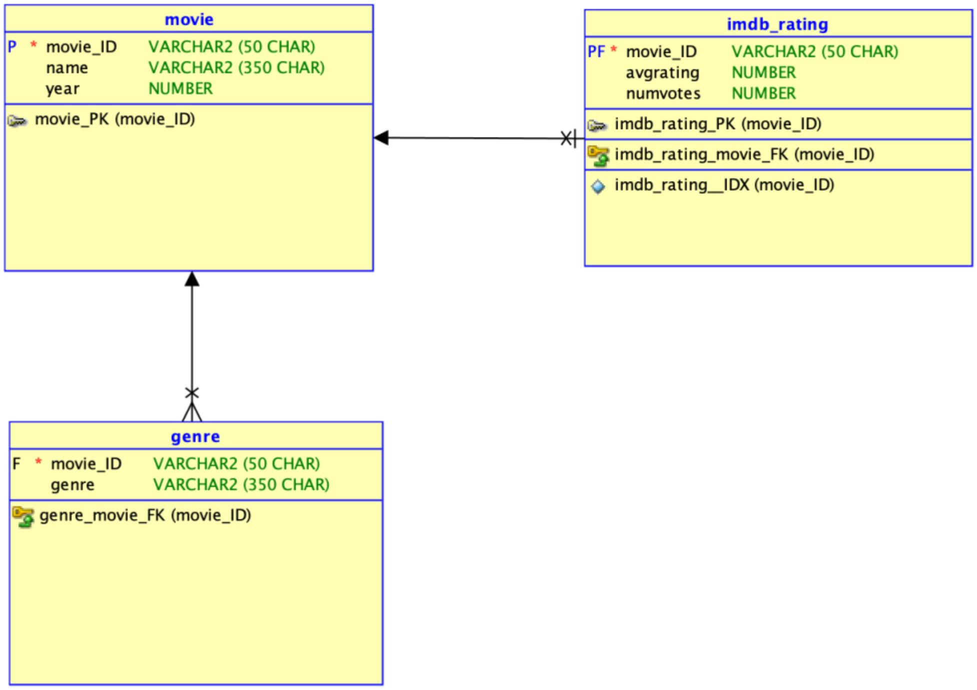

movie
P * movie_ID VARCHAR2 (50 CHAR)
name VARCHAR2 (350 CHAR)
year NUMBER
movie_PK (movie_ID)
imdb_rating
PF * movie_ID VARCHAR2 (50 CHAR)
avgrating NUMBER
numvotes NUMBER
imdb_rating_PK (movie_ID)
imdb_rating_movie_FK (movie_ID)
imdb_rating__IDX (movie_ID)
genre
F * movie_ID VARCHAR2 (50 CHAR)
genre VARCHAR2 (350 CHAR)
genre_movie_FK (movie_ID)

Figure: Relation Model

Relationship

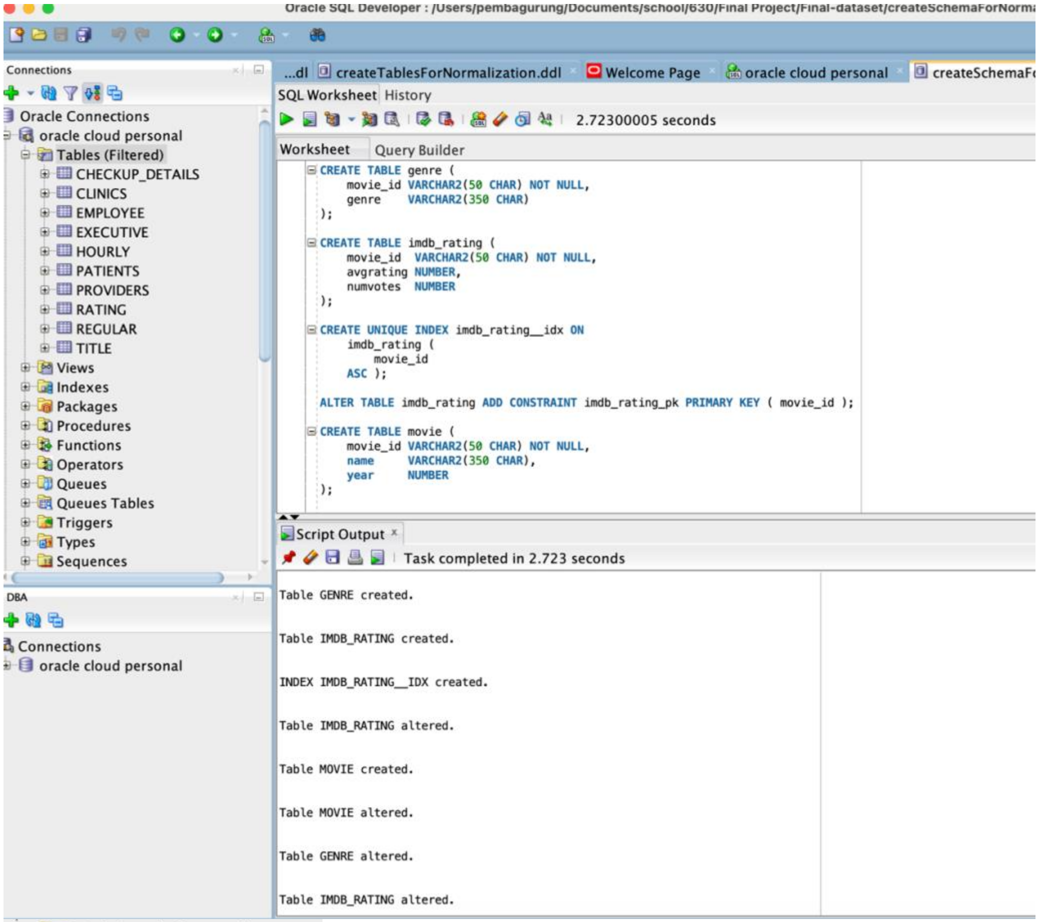
Relationship between imdb_rating and movie table remains same as relationship between title and rating tables which one-to-one or none. The relationship of movie and genre is one to many since one movie has many genres.

DDL

```
/* genre table */
CREATE TABLE genre (
   movie_id VARCHAR2(50 CHAR) NOT NULL,
   genres   VARCHAR2(350 CHAR)
);

/* imdbratings table */
CREATE TABLE imdb_ratings (
   movie_id VARCHAR2(50 CHAR) NOT NULL,
   avgrating NUMBER,
   numvotes NUMBER
);

/* indexing on imdbrating and alter the table to add PK*/
CREATE UNIQUE INDEX imdb_ratings__idx ON
   imdb_ratings (
      movie_id
   ASC);

ALTER TABLE imdb_ratings ADD CONSTRAINT imdb_ratings_pk PRIMARY KEY (movie_id);

/* movie table with PK */
CREATE TABLE movie (
   movie_id      VARCHAR2(50 CHAR) NOT NULL,
   name          VARCHAR2(350 CHAR),
   year   NUMBER
);

ALTER TABLE movie ADD CONSTRAINT movie_pk PRIMARY KEY (movie_id);

/* alter to add FK and Cascade delete rule */
ALTER TABLE genre
   ADD CONSTRAINT genre_movie_fk FOREIGN KEY (movie_id)
      REFERENCES movie (movie_id)
         ON DELETE CASCADE;

ALTER TABLE imdb_ratings
   ADD CONSTRAINT imdb_rating_movie_fk FOREIGN KEY (movie_id)
      REFERENCES movie (movie_id)
         ON DELETE CASCADE;

CREATE SEQUENCE movie_movie_id_seq START WITH 1 NOCACHE ORDER;
```

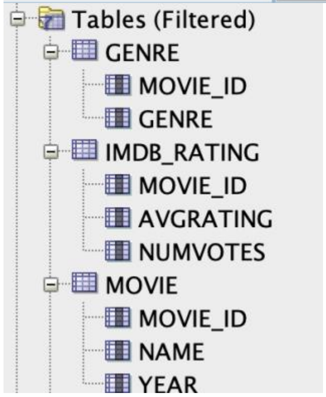
```
/* Trigger */
CREATE OR REPLACE TRIGGER movie_movie_id_trg BEFORE
    INSERT ON movie
    FOR EACH ROW
    WHEN (new.movie_id IS NULL )
BEGIN
    :new.movie_id := movie_movie_id_seq.nextval;
END;
```

Results of running DDL



```
CREATE TABLE genre (
    movie_id VARCHAR2(50 CHAR) NOT NULL,
    genre    VARCHAR2(350 CHAR)
);

CREATE TABLE imdb_rating (
    movie_id  VARCHAR2(50 CHAR) NOT NULL,
    avgrating NUMBER,
    numvotes  NUMBER
);

CREATE UNIQUE INDEX imdb_rating__idx ON
    imdb_rating (
        movie_id
    ASC );

ALTER TABLE imdb_rating ADD CONSTRAINT imdb_rating_pk PRIMARY KEY ( movie_id );

CREATE TABLE movie (
    movie_id VARCHAR2(50 CHAR) NOT NULL,
    name     VARCHAR2(350 CHAR),
    year     NUMBER
);
```

```
Table GENRE created.


Table IMDB_RATING created.


INDEX IMDB_RATING__IDX created.


Table IMDB_RATING altered.


Table MOVIE created.


Table MOVIE altered.


Table GENRE altered.


Table IMDB_RATING altered.
```



Data insertion (DML/DQL) with select statement.

```
/* inserts data from title table into movie table
   where titletype is movie from 1970 or later with IMDB rating of 8+ score */

INSERT INTO movie (movie_id, name, year)
SELECT title.titleid, title.titleName, title.titleyear FROM title
INNER JOIN rating
ON title.titleId = rating.titleId
WHERE title.titletype='movie' AND rating.avgRating >=8 AND title.titleyear >= 1970;

-- Inserts data from rating into imdb_rating

INSERT INTO imdb_rating (movie_id, avgrating, numvotes)
SELECT rating.titleid, rating.avgrating, rating.numvotes FROM rating
INNER JOIN movie
ON movie_id = rating.titleId
WHERE rating.avgRating >=8;

/* Inserts data from title table to genre table
   by splitting the multi valued column from title table
   and enters each value into a cell from genre tables */

INSERT INTO genre (movie_id, genre)
SELECT titleid,
      TRIM(REGEXP_SUBSTR(title.genre, '[^,]+', 1, LEVEL)) AS genre
FROM title
INNER JOIN movie ON movie.movie_id = titleid
CONNECT BY LEVEL <= REGEXP_COUNT(title.genre, ',') + 1
AND PRIOR titleid = titleid
AND PRIOR DBMS_RANDOM.VALUE IS NOT NULL;
```

NOTE: Inserting data for genre is a complicated task. To separate values from multi valued column in original table and store each of those value into a column of new table, I had to use combination of string manipulation functions and queries statement.

The bolded command such as follows are defined,

REGEXP_SUBSTR:

This function extracts each individual value from the *genre* column of the *title* table. The regular expression '[^,]+' matches any sequence of characters that are not commas.

CONNECT BY LEVEL:

This is used to generate rows for each individual value extracted from the *genre* column.

TRIM:

This removes any leading or trailing spaces from the extracted values.

```
-- Deleting columns to create table space in db
alter table title
drop column titlename;
alter table title
drop column titleyear;
alter table title
drop column titletype;
```

Discovering

I have learned to use different features of applications, oracle data modeler, and SQLDeveloper platform such as import wizard features and how to tweak the memory limits for importing data to SQLDeveloper. I have improved at creating a DDL using Data Modeler and implementing it in SQLDeveloper.

My analysis skills to comprehend existing datasets have progressed with the help of database concepts like relational database models, design, ER-diagrams, relationships, keys, and normal forms. In addition, I learned how to modify tables using string manipulation functions and select queries.

Topics from class

1. DML/DQL

   The project involves DML and DQL implementation to build database. In the ***Building Database*** and ***normalization*** sections, DML likes to insert and delete commands, and DQL which is a select command is used to create the database. Further details are included under the ***Building Database*** and ***Running the SQL commands in SQLDeveloper.***

2. Relationships among tables

   My observation includes the study of the relationship the tables for the IMDB dataset and how the relationship of those tables changes after normalizations. The sections ***Relationship.***

3. DB schema, keys, and design pattern:

   Understanding a DB schema is key to creating a successful database management system. My notes for the IMDB data schema are under the ***Finding/Observation*** section. Moreover, I went over the keys and designs of the title and rating tables mentioned in the ***Finding/Observation*** section.

4. Database construction/DDL:
   Database design and data modeling concepts help create and organize database systems efficiently and effectively. I have used Oracle data modeler to replicate the design of the IMDB dataset tile and rating. I constructed a database to bring the title table to 1NF and 4NF. The details are mentioned in **Building Database for IDMB dataset and Designed database schema using data modeler.**

5. Normalizations issues:
   The IMDB dataset is well organized and designed thoroughly. However, the genre field of the title table has multi-valued attributes. The column could be in 1NF by separating all the values into its rows. However, modifying the title table to 1NF creates multi-value dependencies and breaks the 4NF rule. The details of the normalization processes for this project are mentioned in ***Normalization***.

Challenges

Technical issues such as

- Oracle SQL developer memory issue - SQLDeveloper not being able to insert all the data and must keep deleting the columns or table to provide space for new tables.
- Sometimes after data insertion is completed the data gets erased or are not saved.
- Inserting data into genre table from title table was challenging.

Optimistic Objective:

1) Connect the oracle.db with node.js to create a simple movie recommendation application.

Reference

IMDB dataset, https://datasets.imdbws.com