DASC 5300/CSE 5300
Foundations of Computing
Instructor: Sharma Chakravarthy
Project III: DBMS Data Analysis

Instructor: Sharma Chakravarthy
Description of the IMDb Database and Analysis Questions

Made available on:	**11/14/2021**
Project Due on:	**12/7/2021 (last day of classes) Hence, no extension or delayed submission**
Submit by:	**Canvas (1 zipped folder containing a file with English questions, SQL queries, and answers obtained)**
Weight:	**15% of total**
Total Points:	**100**

We have created a large database and populated it with Millions of rows of International Movies and TV episodes information. It is known as the IMDb database by the community (publicly available data set, but not as a relational DBMS) and used by researchers in databases and other fields. The details of the tables are given at the end. This has all movie and TV episode information from the beginning (1925) until 2018 for US and international movies and TV episodes. You can query this database for looking up certain information of interest, finding aggregate and statistical information that you are interested in, and OLAP analysis queries as well to the extent possible using SQL.

The IMDb database includes the following information: movie title, produced year, genres a movie belongs to, actors, writers, directors, runtime, adult or non-adult classification, reviews in terms of votes on the movie, average rating, region, language etc. Similarly for TV series.

The purpose of setting up this database and this exercise is to provide you with an understanding of the differences between analyzing data available in files vs. analyzing data stored in a DBMS. This is a large real-world DBMS and hence you will be writing your analysis queries in SQL (or Structured Query Language). Although the database is large, the response time is good which will allow you to appreciate the technology behind a DBMS (query optimization, concurrency control, simple relational abstraction, easy-to-use, non-procedural query language etc.)

Please make sure you do not write queries that produces large amounts of output. You need to think in terms of aggregate queries so you can extract the sliver of information that you are interested in. In addition, as many fields contain strings with some delimiter, you need to use the LIKE operator with % and _ for selecting the correct string of interest (can also use

string matching). For example, genres can be matched using LIKE ‘Comedy’ or LIKE ‘Drama’. Note the first letter is capitalized. Here is a complete list of genres: Action, Adult, Adventure, Animation, Biography, Comedy, Crime, Documentary, Drama, Family, Fantasy, Game-Show, History, Horror, Music, Musical, Mystery, Short, Sci-Fi, Thriller, Sci-Fi. Similarly, there are number of title types, such as tvSpecial, tvMovie, tvShort, short, tvEpisode, videogame, movie, tvSeries, tvMiniSeries, video. For years, use LIKE ‘200%’ to get values in the range 2000 to 2009. Similarly for others. Some populated field values have a \N as their value. So it is useful to have NOT LIKE ‘\N’ to exclude those.

I. Analysis 1

For this analysis choose 3 genres and 1 title type and write SQL queries to output the number of movies produced for each year (2000 to 2015) for each genre of that titletype. A sample is shown below for the genre Comedy and titletype movie

```
Start
Year          Genres       Movies_produced
-------       ---------    ------------
2000          Comedy       325
2001          Comedy       312
2002          Comedy       314
2003          Comedy       398
2004          Comedy       443
2005          Comedy       539
2006          Comedy       567
2007          Comedy       562
2008          Comedy       689
2009          Comedy       751
```

Analyze the result by plotting them with year on the x-axis and number of movies produced on the Y-axis. See whether you can relate these trends or changes to anything else that we know of. You are welcome to do additional analysis as well.

II. Analysis 2

There are many instances when a person directs a movie, TV series etc., in which s/he also acts. For example, *Ben Affleck directed and starred in the 2012 movie Argo*. Write an SQL query to list actors who are also producers of the same move. You need to do this for 1 title type and 1 genre chosen by you. There are about 20+ genres altogether. Choose titletype and genre that produce some interesting (non-empty) results. *This problem corresponds to a typical SQL query which has joins, group by and having clauses.*

If possible, try to produce an output shown below.

title_type, title name, person name (both acted/directed), year, genre type (optional)

Answers to verify
movie;27511
tvMovie;2817
tvSeries;12648

III. Analysis 3

Choose 1 actor and 1 actress and compute the number of movies they were in for each year (in a 10 year range they have acted because the years depend on the actor/actress you choose) to analyze their popularity, peak year, celebrity status, and their sustained career. You are welcome to do this for more than 1 actor/actress that you are interested in.

IV. Grading Scheme

1.	Analysis 1	(query and output)	25
2.	Analysis 2	(query and output)	25
3.	Analysis 3	(query and output)	25
4.	Report (analysis quality, coverage, clarity)		25
	Total		**100**

I am providing a .txt data file (IMDb_Actors.txt) from which you can search and locate actors/actresses and their unique code starting with nm
This should be useful for analysis 2 and 3. You can load this into an editor and search.

IMDB_ACTORS.txt

This dataset contains the information about the **actors from each IMDB title**. Each field in this dataset is separated by a semi-colon. A random sample from this file has been shown below

```
tt1410063;nm0000288;Christian Bale
tt1429751;nm0004266;Anne Hathaway
tt1872194;nm0000375;Robert Downey Jr.
```

The description of the fields is given below

Field Number	**Field Name**	**Field Description**

1	TITLE ID	The 9-digit unique IMDB title identifier attached to every entry, example: *tt0000091*
2	ACTOR ID	The 9-digit unique actor identifier, example: *nm0000288*
3	ACTOR NAME	The name of the actor, example: *Christian Bale*

IMDb Database Description

In the following table descriptions, varchar2(10) is a single string value whereas varchar2(nnn) where nnn is more than 10 are strings with multiple values with ; as the separator

I. The following tables are populated in the database:

Total Number of Tables: 9
Maximum Number of rows in a table: 27 million rows
Maximum Number of attributes in a table: 9; they are self-explanatory.

1. TITLE_BASICS table

```
SQL> describe TITLE_BASICS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 TCONST                                    NOT NULL VARCHAR2(10)
 TITLETYPE                                 NVARCHAR2(500)
 PRIMARYTITLE                              NVARCHAR2(950)
 ORIGINALTITLE                             NVARCHAR2(950)
 ISADULT                                   NUMBER(1)
 STARTYEAR                                 NUMBER(4)
 ENDYEAR                                   NUMBER(4)
 RUNTIMEMINUTES                            NUMBER(10)
 GENRES                                    NVARCHAR2(350)

 SQL> select count(*) from TITLE_BASICS;

  COUNT(*)
----------
   Total number of row: 4809386 ( 4.8 million rows)
*************************************************************
```

2. TITLE_CREW_WRITER table

```
SQL> describe TITLE_CREW_WRITER
 Name                                      Null?    Type
 ---------------------------------------- -------- ----------------------------

 TCONST                             NOT NULL     VARCHAR2(10)
 WRITERS                            NOT NULL     VARCHAR2(10)

 SQL> select count(*) from TITLE_CREW_WRITER;

  COUNT(*)
----------
  Total number of rows: 5297540 ( 5.2 million rows)
```

3. TITLE_CREW_DIR table

```
SQL> describe TITLE_CREW_DIR
 Name                                      Null?          Type
 ---------------------------------------- -------- ----------------------------

 TCONST                                 NOT NULL     VARCHAR2(10)
 DIRECTORS                              NOT NULL     VARCHAR2(10)

 SQL> select count(*) from TITLE_CREW_DIR;

  COUNT(*)
----------
  Total number of rows: 3408484 (3.4 million rows)
**************************************************************
```

4. TITLE_EPISODE table

```
SQL> describe TITLE_EPISODE
 Name                                      Null?          Type
 ---------------------------------------- -------- ----------------------------

 TCONST                                 NOT NULL     VARCHAR2(10)
 PARENTTCONST                           NOT NULL     VARCHAR2(10)
 SEASONNUMBER                                        NUMBER(9)
 EPISODENUMBER                                       NUMBER(9)

 SQL> select count(*) from TITLE_EPISODE;
```

```
  COUNT(*)
----------
   Total number of rows:3206322 (3.2 million rows)
*******************************************************************
```

5. TITLE_PRINCIPALS table

```
SQL> describe TITLE_PRINCIPALS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 TCONST                                    NOT NULL VARCHAR2(10)
 ORDERING                                           NUMBER(4)
 NCONST                                    NOT NULL VARCHAR2(10)
 CATEGORY                                           VARCHAR2(550)
 JOB                                                VARCHAR2(500)
 CHARACTERS                                         NVARCHAR2(800)

 SQL> select count(*) from TITLE_PRINCIPALS;

  COUNT(*)
----------
 Total number of row: 27054380 ( 27 million rows)
*****************************************************************
```

6. TITLE_RATINGS tables

```
SQL> describe TITLE_RATINGS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 TCONST                                    NOT NULL VARCHAR2(10)
 AVERAGERATING                             NOT NULL NUMBER(5,2)
 NUMVOTES                                  NOT NULL NUMBER(15)

 SQL> select count(*) from TITLE_RATINGS;

  COUNT(*)
----------
 Total number of rows:  805011 ( 0.8 million rows)
*******************************************************************
```

7. TITLE_AKAS table

```
SQL> describe TITLE_AKAS
 Name                                      Null?         Type
 ----------------------------------------- -------- ----------------------------

 TITLEID                                   NOT NULL VARCHAR2(10)
 ORDERING                                           NUMBER(10)
 TITLE                                              NVARCHAR2(950)
 REGION                                             NVARCHAR2(550)
 LANGUAGE                                           NVARCHAR2(550)
 TYPES                                              NVARCHAR2(550)
 ATTRIBUTES                                         NVARCHAR2(500)
 ISORIGINALTITLE                                    NUMBER(2)

 SQL> select count(*) from TITLE_AKAS;

  COUNT(*)
----------
   3563547 (3.5 million rows)
************************************************************
```

8. NAME_TITLE_MAPPING table

```
SQL> describe NAME_TITLE_MAPPING
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 NCONST                                    NOT NULL VARCHAR2(10)
 TCONST                                    NOT NULL VARCHAR2(10)

SQL> select count(*) from NAME_TITLE_MAPPING;

  COUNT(*)
----------
  14144524 (14 million rows)
*************************************************************
```

9. NAME_BASICS table

```
SQL> describe NAME_BASICS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 NCONST                                    NOT NULL VARCHAR2(10)
 PRIMARYNAME                               NOT NULL NVARCHAR2(950)
 BIRTHYEAR                                          NUMBER(4)
 DEATHYEAR                                          NUMBER(4)
 PRIMARYPROFESSION                                  VARCHAR2(900)
```

```
SQL> select count(*) from NAME_BASICS;

  COUNT(*)
----------
   8424762 (8.4 million rows)
```