1 Overview

In this assignment you will practice using SQL to create a schema for video store relations as well as writing basic SQL queries on a given movie database. The database schema you create will be managed using the Oracle DBMS. You may download and install Oracle server at your local machine. Feel free to explore. In this document, I provide instructions below for you to use an online tool to practice.

1.1 Use Oracle Live SQL

- Open your browser and go to *https://livesql.oracle.com*
- Click **Start Coding Now**
- If you don't have an Oracle Account, click **Create Account** to create an account. Otherwise, provide your username and password to Sign in.
- You can use the *SQL Worksheet* window to write your SQL statements. To separate them, you may use forward slash or semicolon. For example,

```
--Movie tuples
SELECT *
```

```
FROM Movie
/
--Actor IDs
SELECT actorid
FROM Actor
/
```

You can run the SQL statements you wrote by clicking **Run** at the top right corner. You can save your statements as script using **Save** that is left to **Run** at the top right corner.

- If you need to upload a script file such as *tuples.sql*, you can click *My Scripts* at the left column. At the *My Scripts* page, click **upload Script** at the top of the right column to load a SQL script file from your local computer. After you upload each script, you need to run the script by clicking Run Script at the uploaded script page.
- If you want to check the data schemas you created at the session, you may click **Schema** at the left column.

You need some time to practice and be familiar with the tool. Have fun!

2 What you need to turn in

The required script files described below could be created based on your saved session scripts using Oracle live SQL.

2.1 Use SQL to create three tables

In this part of your assignment, you need to create a file called *mytables.sql*, which is a text file containing your SQL statements that can be used to

create three tables. Note that you can design your own column and table constraints. You need state your assumptions in your turn-in work.

The subset of video store relations includes the following relations (40 points):

Customer (CustomerID, Name, Street, City, State, Zipcode)

Film (FilmID, Title, RentalPrice, Kind)

Reserved (CustomerID, FilmID, ResDate)

The *mytables.sql* file should have the following format (note, two dashes '–' at the start of a line indicates a comment). Please add other comments as needed.

```
-- Table 1
-- Customer (CustomerID, Name, Street, City, State, Zipcode)
-- Other comments if you want
CREATE ...


-- Table 2
-- Film(...
--
```

If the statement does not work or is incomplete, please comment out the SQL text and put in appropriate comments as to why the statement does not work.

Note that good programming style is always important. Your code should be clear, concise, and have appropriate comments.

2.2 Use SQL to write basic queries

In this part of your assignment, you are going to practice writing SQL queries. On D2L, I will provide you three script files including *createta-*

bles.sql, *tuples.sql*, and *droptables.sql* so that you can construct database tables, load data into the tables, and remove the tables.

2.2.1 Creating the movie database

An SQL script to create the tables is available in the file *createtables.sql*. If you ever need to remove the tables use the script in the file *droptables.sql*. The database that will be created has three relations, with the schema listed below.

Movie (MovieID, Title, Year, Score, Votes)

Actor(ActorID, Name)

Casting (MovieID, ActorID, Ordinal)

In the Movie table, the Score is a measure of a movie's popularity as voted on by internet users. Votes is the number of votes cast. The Casting table relates actors with the movies in which they are cast. The Ordinal column is the cast 'billing order', e.g., the star actor in a movie has an ordinal of 1, the second leading star has an ordinal of 2, a bit player would have an ordinal of 80 (assuming at least 80 actors were in the movie).

2.2.2 Loading the database

To load data into the three tables in the movie database, you can use the given SQL script *tuples.sql*. The script is a sequence of insertion operations. The data is from the Internet Movie Database. There are 910 tuples in the Casting table, 807 Actors, and 58 movies.

2.2.3 Queries to write

You need to create a file called *queries.sql*, which is a text file containing your queries for each of the questions given below. Each query will be (at least one) SELECT statement.

1. (10 points) List the titles of all movies.
2. (10 points) List the actor table.
3. (10 points) list movies that have scores greater than 7.0.
4. (10 points) List movies that have scores greater than 7.0 or years greater than 1990.
5. (10 points) List movies that have scores greater than 7.0 and years greater than 1990.
6. (10 points) List movies that have votes between 3000 and 4000.

The *queries.sql* file should have the following format (note, two dashes '–' at the start of a line indicates a comment). Please add other comments as needed.

```
-- Query 1
-- List the titles of all movies
-- Other comments if you want
SELECT ...
FROM ...

-- Query 2
-- List the actor table
--
```

Be sure to include the number for each query, the English text describing the query and the SQL text for the query. If the query does not work or

is incomplete, please comment out the SQL text and put in appropriate comments as to why the query does not work.

Note that good programming style is always important. Your code should be clear, concise, and have appropriate comments.

4 APPENDIX: Sample SQL DDL Code

Sample SQL DDL statements are provided in the following script, which is used to create a database schema for the movie database used in the assignment. You can use the script as your reference to write your own. The movie database has three relations, with the schema listed below.

Movie (MovieID, Title, Year, Score, Votes)

Actor(ActorID, Name)

Casting (MovieID, ActorID, Ordinal)

Below is the script code that creates the schema.

```
-- ============================================================
--   Table: MOVIE
-- ============================================================
create table MOVIE
(
    MOVIEID INTEGER                not null,
    TITLE   VARCHAR2(70)           not null,
    YEAR    NUMBER(4)              not null,
    SCORE   FLOAT                  null    ,
    VOTES   INTEGER                null    ,
    constraint PK_MOVIE primary key (MOVIEID)
)
/

-- ============================================================
--   Table: ACTOR
-- ============================================================
create table ACTOR
(
    ACTORID INTEGER                not null,
    NAME    VARCHAR2(35)           not null,
```

```
    constraint PK_ACTOR primary key (ACTORID)
)
/

-- ============================================================
--   Table: CASTING
-- ============================================================
create table CASTING
(
    MOVIEID  INTEGER                not null,
    ACTORID  INTEGER                not null,
    ORDINAL  INTEGER                null    ,
    constraint PK_CASTING primary key (MOVIEID, ACTORID)
)
/

-- ============================================================
--   Index: APPEARS_IN_FK
-- ============================================================
create index APPEARS_IN_FK on CASTING (MOVIEID asc)
/

-- ============================================================
--   Index: STARS_FK
-- ============================================================
create index STARS_FK on CASTING (ACTORID asc)
/

alter table CASTING
    add constraint FK_CASTING_APPEARS_IN_MOVIE foreign key  (MOVIEID)
       references MOVIE (MOVIEID)
/

alter table CASTING
    add constraint FK_CASTING_STARS_ACTOR foreign key  (ACTORID)
       references ACTOR (ACTORID)
/
```

Note that SQL is case insensitive. You are free to choose upper- or lower-case characters in your code.