IS480 – Final Exam Part 2

Academic Honor Code
Graduate Business Programs
College of Business
California State University Long Beach

Insert Your Name Here:
Jon Aldrete
Insert Your Beach ID Here:
026774967

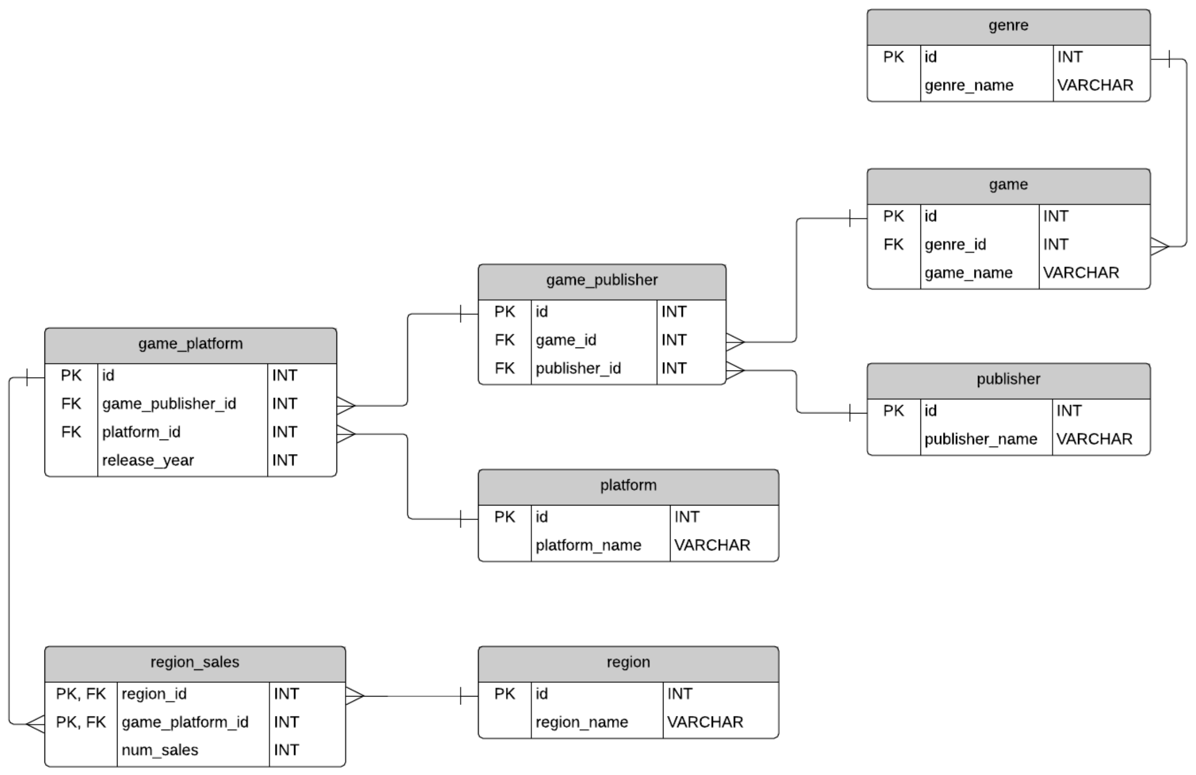
Instructions

1. For the 2nd part of this final, we are going to play with 2 Databases:
    a. **Movies DB**: this DB includes data about multiple movies similar to what we could expect from a site like www.IMDB.com, and
    b. **Videogames DB**: this DB contains data about videogames including their publishers, availability in different platforms and sales records.

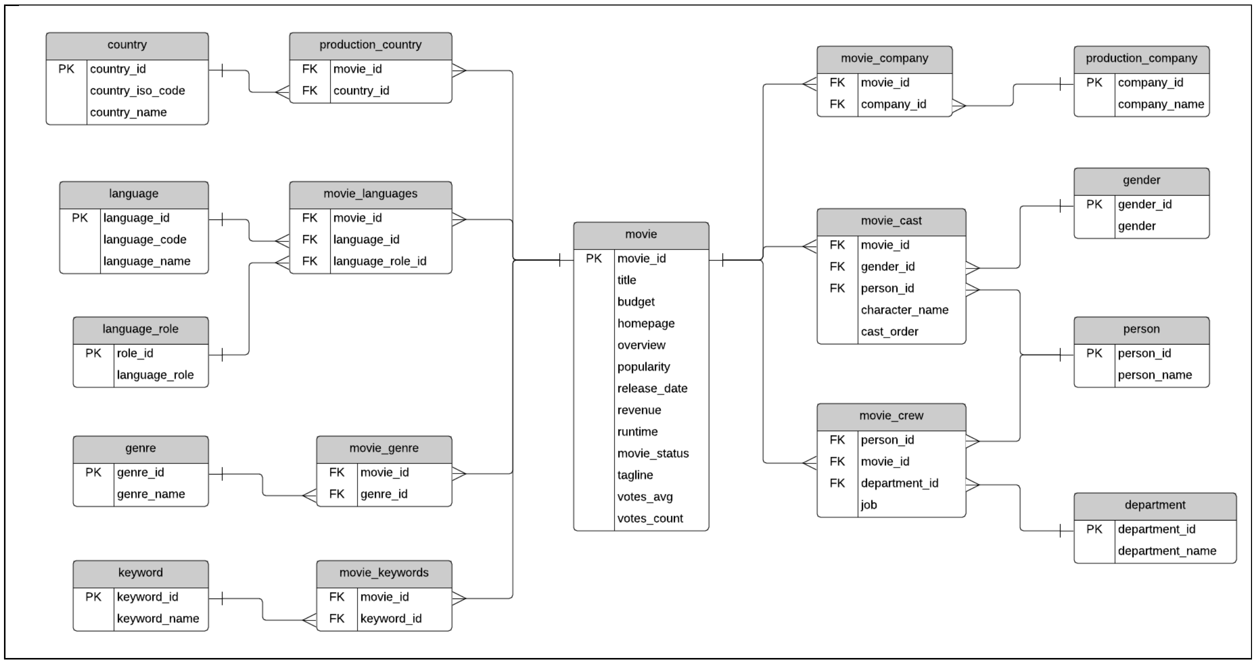
2. PKs and FKs are properly identified in the models below. While the models shown are mixing the elements of a conceptual and a logical model rather than keeping them separated, I decided to show it as is since this is the way the original developers of the databases presented them. Please assume this is already a fully normalized (3rd normal form) logical model.

3. Each question presents a slightly different challenge based on the same exercises we have been working for in the last few weeks. For any questions or technical support please come forth in class and let me know. I'll be happy to support you all.

Video Games DB:



Movies DB:



Exercises:

We will start by looking at the video games data. We will take the role of a Gamestop-like company that keeps tracks of its global sales of video games as well as their backend information such as publication records and sales data.

1. 15pts. In this problem we aim to update the sales figures for each video game based on the console they belong too. Specifically, the task is **to create a named procedure** that will look at the Videogames data and:

Updating Sales Figures:

a) Define and use a cursor to retrieve information about the video games' names, the console where it was published, and their global_sales.
b) Now we will update the sales figures to the latest values. For games sold in the Wii console, increase the total sales by 20% and update it in the DB.
c) For games sold in the PS3 console, increase the sales figures by 15% and update them in the DB.
d) For any other game, increase the sales figures by 10%, and update the value in the DB.

Place your answer within this Box:

```
SET SERVEROUTPUT ON
SET ECHO ON

CREATE OR REPLACE PROCEDURE Q1 AS
   v_gname game.game_name%type;
   v_consolename platform.platform_name%type;
   v_gbl_sales NUMBER;

CURSOR cVideoGames IS
   SELECT game.game_name, platform.platform_name, region_sales.num_sales
   FROM game, game_publisher, game_platform, platform, region_sales
   WHERE game.id = game_publisher.game_id
   AND game_platform.game_publisher_id = game_publisher.id
   AND platform.id = game_platform.platform_id
   AND platform.id = region_sales.region_id
   ORDER BY game_name;

BEGIN

OPEN cVideoGames;
LOOP
   FETCH cVideoGames INTO v_gname, v_consolename, v_gbl_sales;

   IF v_consolename = 'Wii' THEN
   UPDATE region_sales
   SET num_sales = num_sales * 1.20;

   ELSIF v_consolename = 'PS3' THEN
   UPDATE region_sales
   SET num_sales = num_sales * 1.15;

   ELSE
   UPDATE region_sales
   SET num_sales = num_sales * 1.10;
   EXIT WHEN cVideoGames%NOTFOUND;
END IF;
END LOOP;
CLOSE cVideoGames;
END;
```

Registering a video game release in a new console:

2. Goal:
   Register the release of a game in a new console. Recently a former AAA game was re-released into a new generation console. To simplify the activity of recording this release into our systems,

we aim to create a procedure that will validate the game's released had not already been recorded, and, if not, proceeds to record it.

Task: 20pts.
First, the procedure must collect the games' name from a parameter sent to the DB. From this value, **pass it to a function to** obtain the ID of the game, and confirm that the database does not already have a record of the game having been released to this console.

- If it does, stop the process and print a statement stating the information is already there.
- If it does not appear in the DB, return the information back to the procedure where you must record the game's ID and publication console into our DB.i

Deliverables:

1) Overarching procedure that records the game's release into a new console.
2) The function that validates the game had not been previously released into the console.

HINT: Look at the DB model and consider what information (parameters) needs to be passed to the procedure in the first place.

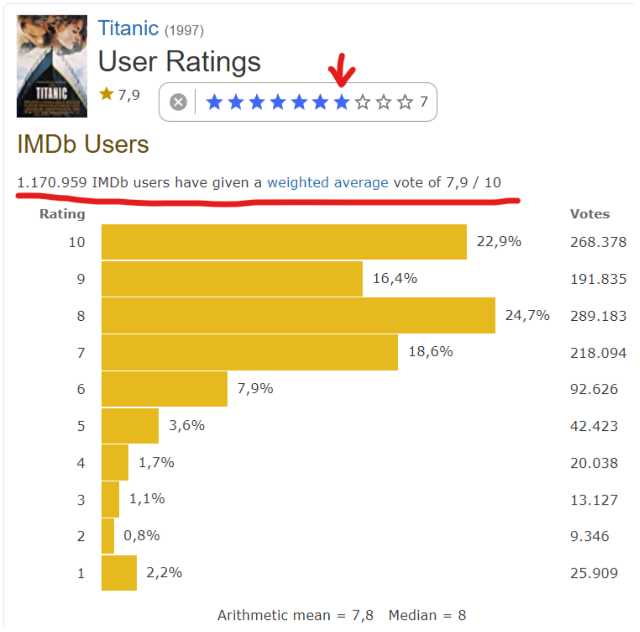
Place your answer within this Box:

```
 create or replace FUNCTION Validate_New_Release(
   p_game_name VARCHAR2,
   p_consoleID INT
) RETURN Boolean AS
v_game_id INT;
v_count INT;

BEGIN
   SELECT game.id INTO v_game_id
   FROM game
   WHERE game_name = p_game_name;

   SELECT count(*) INTO v_count
   FROM game, game_publisher, game_platform, platform
   WHERE platform.id = game_platform.platform_id
   AND game_publisher.game_id = game.id
   AND platform.id = p_consoleID
   AND game.id = v_game_id;

   IF v_count = 0 THEN
      RETURN FALSE;
   ELSE
      RETURN TRUE;
END IF;
END;

SET SERVEROUTPUT ON
SET ECHO ON

CREATE OR REPLACE PROCEDURE New_Game_Release(
   p_game_name VARCHAR2,
   p_consoleID INT
   ) AS
   v_newGameID game.id%TYPE;
   BEGIN
   IF Validate_New_Release(p_game_name, p_consoleID) = TRUE THEN
      dbms_output.put_line('Information is already there!!!');
   ELSE
      INSERT INTO game (id, genre_id, game_name)
      VALUES (v_newGameID, null, p_game_name);

   END IF;
   END;
```

Registering new votes (ratings by users) into our records and recalculate the average ranking:

3. Goal:
   Imagine we are managing the servers for an IMDB like company. We would like to create a procedure that allow us to update the average rating (user_avg) of a movie in our website whenever a user submits a vote.

   In essence, every time someone submits a vote, the average rating gets recalculated and published automatically. See below:



Titanic (1997)

User Ratings

★ 7,9

★★★★★★★☆☆☆ 7

IMDb Users

1.170.959 IMDb users have given a weighted average vote of 7,9 / 10

Rating		Votes
10	22,9%	268.378
9	16,4%	191.835
8	24,7%	289.183
7	18,6%	218.094
6	7,9%	92.626
5	3,6%	42.423
4	1,7%	20.038
3	1,1%	13.127
2	0,8%	9.346
1	2,2%	25.909

Arithmetic mean = 7,8 Median = 8

Place your answer within this Box:

```
 create or replace PROCEDURE Q3(
p_vote NUMBER,
p_movie movie.title%TYPE) AS

v_vote_avg NUMBER;
v_newVoteAvg NUMBER;
v_vote_count movie.vote_count%TYPE;

BEGIN

SELECT VOTE_AVERAGE
INTO v_vote_avg
FROM MOVIE
WHERE movie.title = p_movie;

SELECT VOTE_COUNT
INTO v_vote_count
FROM MOVIE
WHERE movie.title = p_movie;

v_newVoteAvg := (v_vote_avg + p_vote)/v_vote_count + 1;

UPDATE MOVIE
SET VOTE_AVERAGE = v_newVoteAvg
WHERE movie.title = p_movie;

END;
```

Registering new actor in a movie:

4. **15pts. Goal:**
   Since not all movies we keep track of have been released, sometimes the cast may change. In those situations, we would like to update our DB by registering the new person (if they have not been registered in our database before), and then registering their role in the movie.

   For this problem, please answer the following questions:
   (I want you to think carefully into the process. What will happen if we put the data into the wrong order, If the person is new how to do record their role in the movie within the movie_cast table?

   I. What input data would the DB need to be able to do its job? Why? (that is, what parameters should we record and why?)

   To register an new actor into the movie they would enter there person_id, and gender id, and the movie they are adding there role into the movie cast table, after registering there information in the person table and gender table.

   II. Very briefly what should the application be able to do and in what order?

   The application should ask the user their name, person id, and gender, as parameters, this would inserted into the two separate tables, from those tables, will connect the foreign keys to the role of the characters, and the person would enter there role into the cast. I would also use a validating function, to validate if they in the system, and I will follow the foreign keys and primary to make sure the order is right.

Place your answer within this Box:

```
 create or replace PROCEDURE Q4(
   p_actor_name person.person_name%type,
   p_gender movie_cast.gender_id%TYPE,
   p_movie_title movie.title%type
)AS

v_movie_id NUMBER;
v_new_id_num NUMBER;
v_cast_order NUMBER;
v_old_id_num NUMBER;

BEGIN

SELECT movie_id into v_movie_id
FROM movie
WHERE title = p_movie_title;

select count(*) into v_new_id_num
from person;

Select MAX(cast_order)into v_cast_order
FROM movie_cast;

v_new_id_num:= v_new_id_num +1;
v_cast_order:= v_cast_order + 1;

IF f_Exist(p_actor_name) = FALSE THEN

INSERT INTO person VALUES(v_new_id_num, p_actor_name);
INSERT INTO MOVIE_CAST VALUES(v_movie_id,v_new_id_num,
p_actor_name,p_gender,v_cast_order);

ELSE
   SELECT person_ID into v_old_id_num
   FROM person
   where person_name = p_actor_name;
   INSERT INTO MOVIE_CAST VALUES(v_movie_id, v_old_id_num , p_actor_name,
p_gender,v_cast_order);

END IF;
END;
```