Islamic University of Technology

CSE-4410
Database Management Systems - II Lab

Lab Report - 3

Name: Mukit Mahdin
ID: 200042170
Department: CSE
Program: SWE

Task - 1:
Write a procedure that will take a mov_title and show the required time (–hour –minute) to play that movie in a cinema hall. Let's say, there will be an intermission of 15 minutes after every 70 minutes only if the remaining time of the movie is greater than 30 minutes.

```
CREATE OR REPLACE PROCEDURE calculate_movie_time (mov_title IN
VARCHAR2)
AS
    mov_length NUMBER;
BEGIN
    SELECT mov_time INTO mov_length FROM movie WHERE mov_title =
mov_title;

    intermission NUMBER := CEIL(mov_length / 70);

    IF intermission > 0 AND (mov_length - (70 * (intermission -
1))) > 30 THEN
        total_time NUMBER := mov_length + (intermission * 15);
    ELSE
        total_time NUMBER := mov_length;
    END IF;

    hours NUMBER := TRUNC(total_time / 60);
    minutes NUMBER := MOD(total_time, 60);

    DBMS_OUTPUT.PUT_LINE('Movie: ' || mov_title);
    DBMS_OUTPUT.PUT_LINE('Total Runtime: ' || hours || ' hours '
|| minutes || ' minutes');
END;
```

Description:
This procedure selects the movie length (in minutes) from the "movie" table based on the given movie title. It then calculates the number of intermissions required by dividing the movie length by 70 and rounding it up using the CEIL function. If the intermission is necessary (the intermission count is greater than 0 and the remaining time of the movie after all intermissions is greater than 30 minutes), it adds 15 minutes to the total time for each intermission. Finally, it calculates the number of hours and minutes from the total time and prints the results.

Task - 2:

Write a procedure to find the N top-rated movies (the average rev_stars of a movie is higher than other movies). The procedure will take N as input and print the mov_title up to N movies. If N is greater than the number of movies, then it will print an error message.

```
CREATE OR REPLACE PROCEDURE top_rated_movies (N IN NUMBER)
AS
    CURSOR top_rated_movies IS
        SELECT mov_id, mov_title, AVG(rev_stars) avg_rating
        FROM movie m
        JOIN rating r ON m.mov_id = r.mov_id
        GROUP BY mov_id, mov_title
        ORDER BY avg_rating DESC;

    movie_count NUMBER := 0;
BEGIN
    FOR movie IN top_rated_movies LOOP
        movie_count := movie_count + 1;
        DBMS_OUTPUT.PUT_LINE(movie_count || '. ' ||
movie.mov_title || ' (' || movie.avg_rating || ')');
        IF movie_count = N THEN
            EXIT;
        END IF;
    END LOOP;

    IF movie_count < N THEN
        DBMS_OUTPUT.PUT_LINE('Error');
    END IF;
END;
```

Description:

This procedure defines a cursor top_rated_movies that selects the movie ID, movie title, and average rating (calculated using the AVG function) from the "movie" and "rating" tables, grouping by movie ID and title, and ordering by average rating in descending order. It then loops through the cursor and prints the movie title and average rating for each movie until N movies are found or the cursor is exhausted. If N is greater than the number of movies, it prints an error message indicating the number of movies found.

Task - 3:
Suppose, there is a scheme that for each rev_stars greater than or equal to 6, a movie will receive $10. Now write a function to calculate the yearly earnings (total earnings/year between the current date and release date) of a movie that is obtained from user reviews.

```
CREATE OR REPLACE FUNCTION calculate_movie_earnings (p_mov_title
VARCHAR2)
RETURN NUMBER
AS
    temp_mov_id NUMBER;
    temp_mov_year NUMBER;
    temp_mov_release_date DATE;
    temp_total_earnings NUMBER:= 0;
    temp_num_of_ratings NUMBER;
    temp_num_of_years NUMBER;
BEGIN
    SELECT mov_id, mov_year, mov_release_date INTO temp_mov_id,
temp_mov_year, temp_mov_release_date
    FROM movie
    WHERE mov_title = p_mov_title;

    SELECT COUNT(*) INTO temp_num_of_ratings
    FROM rating
    WHERE mov_id = temp_mov_id AND rev_stars >= 6;

    temp_num_of_years := TRUNC(MONTHS_BETWEEN(SYSDATE,
temp_mov_release_date) / 12);

    temp_total_earnings := temp_num_of_ratings * 10 *
temp_num_of_years;

    RETURN temp_total_earnings;
END;
```

The function calculates the yearly earnings of a movie based on the number of reviews it receives. The function takes the movie ID as input and retrieves the relevant details about the movie from the movie schema, such as the title, release date, and release year. The number of reviews is determined by counting the number of reviews with a rating greater than or equal to 6 from the rating schema. For each review with a rating greater than or equal to 6, the movie earns $10. The total earnings are then calculated by multiplying the number of reviews by $10. The number of years between the release date and the current date is then calculated, and the total earnings is divided by this value to determine the yearly earnings. The function finally returns the yearly earnings of the movie.

Task - 4:

Write a function in PL/SQL, that given a genre (gen_id) will return genre status, additionally the review count and average rating of that genre.

```
CREATE OR REPLACE FUNCTION get_genre_status (gen_id NUMBER)
RETURN VARCHAR2
IS
  v_review_count NUMBER;
  v_avg_rating NUMBER;
  v_gen_avg_review_count NUMBER;
  v_gen_avg_rating NUMBER;
  v_result VARCHAR2(30);
BEGIN

  SELECT COUNT(*), AVG(r.rev_stars)
  INTO v_review_count, v_avg_rating
  FROM ratings r
  JOIN movies m ON r.mov_id = m.mov_id
  JOIN mtype mt ON m.mov_id = mt.mov_id
  WHERE mt.gen_id = gen_id;

  SELECT AVG(cnt), AVG(avg_rating)
  INTO v_gen_avg_review_count, v_gen_avg_rating
  FROM
  (
    SELECT COUNT(*) cnt, AVG(r.rev_stars) avg_rating
    FROM ratings r
    JOIN movies m ON r.mov_id = m.mov_id
    JOIN mtype mt ON m.mov_id = mt.mov_id
    GROUP BY mt.gen_id
  );
  IF v_review_count > v_gen_avg_review_count AND v_avg_rating >
v_gen_avg_rating THEN
    v_result := 'People''s Favorite';
  ELSIF v_review_count > v_gen_avg_review_count THEN
    v_result := 'Widely Watched';
  ELSIF v_avg_rating > v_gen_avg_rating THEN
    v_result := 'Highly Rated';
  ELSE
    v_result := 'So So';
  END IF;

  RETURN v_result;
END get_genre_status;
```

Description:
This function, get_genre_status, takes a genre ID as input and returns the genre status based on the review count and average rating of the genre. The function starts by finding the review count and average rating of the genre specified in the input. Then, it finds the average review count and the average rating of all genres. Finally, it determines the genre status based on the review count and average rating of the genre and the average of all genres. The genre status can be one of the following: "People's Favorite", "Widely Watched", "Highly Rated", or "So So". The function returns the genre status as a string value.

Task - 5:
Write a function, that given two dates will return the most frequent genre of that time (according to movie count) along with the count of movies under that genre that had been released in the given time range.

```
CREATE OR REPLACE FUNCTION get_most_frequent_genre(start_date
DATE, end_date DATE)
RETURN VARCHAR2
IS
   l_gen_title VARCHAR2(50);
   l_mov_count NUMBER;
```

```
BEGIN
   SELECT gen_title, COUNT(mt.mov_id)
   INTO l_gen_title, l_mov_count
   FROM genres g
   JOIN mtype mt ON g.gen_id = mt.gen_id
   JOIN movie m ON mt.mov_id = m.mov_id
   WHERE m.mov_releasedate BETWEEN start_date AND end_date
   GROUP BY gen_title
   ORDER BY l_mov_count DESC
   FETCH FIRST 1 ROW ONLY;

   RETURN l_gen_title || ' ' || l_mov_count || ' ';
END;
```

Description:

This function in will determine the most frequently occurring genre of movies within a given time range. The input to the function will be two dates, which specify the start and end of the time range. The function will retrieve data from three tables: "Movie", "Genres", and "MType". The "Movie" table contains information about each movie, including its ID, title, year of release, and release date. The "Genres" table contains information about different genres of movies, including their ID and title. The "MType" table maps movies to their corresponding genres using the movie ID and genre ID.

The function will first filter the movies in the "Movie" table that were released within the given time range. Then it will count the number of movies belonging to each genre by joining the filtered movie data with the "MType" and "Genres" tables. Finally, the function will return the genre with the most number of movies and the count of movies within that genre.