Name: K.M. Tahlil Mahfuz Faruk

Student ID: 200042158

Department: CSE(SWE)

Course: CSE 4410

Database Management System II LAB 3

K.M. Tahlil Mahfuz Faruk
Islamic University of Technology

SQL Commands:

In task 1,

```
set SERVEROUTPUT on;
--1--
create or replace Procedure
requiredtime(title in varchar)
As
    time number;
    hour number;
    minute number;
    intermission number;
    c number;
begin
    select mov_time into time
    from movie
    where mov_title=title;

    hour:=round(time/60);
    minute:=time-hour*60;
    c:=round(time/70);
    if((time-(c-1)*70)>=30) THEN
        intermission:=c*15;
    else
        intermission:=(c-1)*15;
    end if;
    if(hour<1 and time>70)then
        DBMS_OUTPUT.PUT_LINE( A: '1HOUR:' || hour || ' ');
    elsif(hour>1 and time<70) then
        DBMS_OUTPUT.PUT_LINE( A: '2HOUR:' || hour || ' ' || 'Minute:' || Minute || ' ');
    else
        DBMS_OUTPUT.PUT_LINE( A: '3HOUR:' || hour || ' ' || 'Minute:' || Minute || ' ' || 'Intermission(mins):' ||intermission);
    end if;
end;
/

begin
    requiredtime( title: 'Vertigo');
end;
/
```

Explanation:

- A mov title will be passed in the function parameter.
- Find the title's time. Divide it by 60 and round to find the hour. Then find the minute by subtracting the hour*60 from the time.
- Now divide the time by 70 and round to find the interval time. Interval would be valid if remaining time is greater than 30 mins. Check that condition by (c-1)*70>=30.

Difficulties:

- No significant difficulty appeared.

In task 2,

```
--2--
create or replace Procedure
topratedmovies(n in number)
As
    i number;
    c number;
    average number;
    track number;
begin
    select avg(rev_stars) into average
    from movie natural join rating natural join reviewer;

    i:=0;
    c:=0;

    for row in (select mov_title,rev_stars from movie natural join rating natural join reviewer) loop
        if(row.rev_stars>average)then
            c:=c+1;
        end if;
    end loop;
    if(c>n)then
        DBMS_OUTPUT.PUT_LINE( A: 'Error');
    else
        for row in (select mov_title,rev_stars from movie natural join rating natural join reviewer) loop
        if(row.rev_stars>average and c<=n)then
            DBMS_OUTPUT.PUT_LINE( A: row.mov_title);
            track:=track+1;
        end if;
    end loop;
    end if;
end;
/

begin
    topratedmovies( n: 200);
end;
/
```

Explanation:

- First find the average rev_stars and save it tin average variable.
- Now count the number of movies that has greater review than average.
- If it's greater than n then print error.
- Else print those movies.

Difficulties:

- No such difficulties were faced during this task.

In task 3,

```
--3--
create or replace
function yearlyearnings(movieid number)
return number
is
earnings number;
yearly number;
rd date;
n number;
begin
    earnings:=1;

    select mov_releasedate into rd
    from  movie
    where mov_id=movieid;

    select count(mov_id) into n
    from  movie natural join rating natural join reviewer
    where mov_id=movieid and rev_stars>=6
    group by mov_id;

    earnings:=(10*n);
    yearly:=earnings/((sysdate-rd)/365);

    return yearly;
end;
/

begin
    DBMS_OUTPUT.PUT_LINE( A: 'Yearly Income: ' || yearlyearnings( movieid: 902));
end;
/
```

Explanation:

- First find the release data of the movie and also find the number of reviews that movie has got which is greater than 6.
- Now multiply by 10 with the number of reviews and then divide it by the year.
- That will provide the yearly income.

Difficulties:

- To find the year from the release data was the challenge.

In task 4,

```
--4--
create or replace
function genrestatusshow(genreid in number)
return varchar2
is
avg_review_of_all_genre number;
avg_rating_of_all_genre number;
val varchar2(150);

    cursor dummytable
    is
        select gen_id,gen_title,avgstars,avg_reviews_per_genre
        from (select GENRES.GEN_ID,avg(REV_ID) as avg_reviews_per_genre
            from genres natural join REVIEWER
            group by genres.gen_id) t1
            natural join
            (select GENRES.GEN_ID,GENRES.GEN_TITLE, avg(rating.rev_stars) as avgstars
            from genres natural join rating
            group by genres.gen_id, GENRES.GEN_TITLE
            order by gen_id) t2;
begin
    select avg(M) into avg_review_of_all_genre
    from(select GENRES.GEN_ID,avg(REV_ID) M
        from genres natural join REVIEWER
        group by genres.gen_id);

    select avg(x) into avg_rating_of_all_genre
    from(select GENRES.GEN_ID,GENRES.GEN_TITLE, avg(rating.rev_stars) as x
        from genres natural join rating
        group by genres.gen_id, GENRES.GEN_TITLE
        order by gen_id);

    FOR row IN dummytable
    LOOP
        if(row.GEN_ID=genreid) then
            if(row.avg_reviews_per_genre>avg_review_of_all_genre and row.avgstars<avg_review_of_all_genre) then
                val:='Widely Watched';
                return val;
            elsif(row.avg_reviews_per_genre<avg_review_of_all_genre and row.avgstars>avg_review_of_all_genre) then
                val:='Highly rated';
                return val;
            elsif(row.avg_reviews_per_genre>avg_review_of_all_genre and row.avgstars>avg_review_of_all_genre) then
                val:='People favourite';
                return val;
            end if;
        end if;
    END LOOP ;
    val:='so so';
    return 'so so';
end;
/

begin
    DBMS_OUTPUT.PUT_LINE( A: GENRESTATUSSHOW( genreid: 1012));
end;
/
```

Explanation:

- First find the average review stars. To find it do a subquery.
- Also do a subquery to find the average review count.
- Now natural join both of these queries using the gen_id that will provide the average stars and reviews and save the result in a cursor called dummytable.
- Now Find the average review and stars of all the genres using two queries and save them to variable avg_review_of_all_genres and avg_stars_of_all_genres respectively.
- Now compare according to condition given and return the genre status.

Difficulties:

- To do the subquery and get them to a table was the challenge.

In task 5,

```
--5--
create or replace
type freq_genre_and_movie_count as object
(
    genre_id number,
    genre_title varchar(150),
    mov_count number
);

create or replace
function frequent_genre(starting varchar,ending varchar)
return freq_genre_and_movie_count
is
    data freq_genre_and_movie_count;
    genre_id GENRES.GEN_ID%type;
    genre_title GENRES.GEN_TITLE%type;
    countfreq number;

begin
    select GEN_ID,GEN_TITLE,num_of_movies into genre_id,genre_title,countfreq
    from(select GEN_ID,GEN_TITLE,MOV_RELEASEDATE
        from MOVIE natural join GENRES
        where to_date(starting,'DD-MON-YY')<MOV_RELEASEDATE
          and MOV_RELEASEDATE<to_date(ending,'DD-MON-YY') and rownum<=1
        order by MOV_RELEASEDATE)
        natural join
        (select GEN_ID,count(MOV_ID) num_of_movies
        from MOVIE natural join GENRES
        where to_date(starting,'DD-MON-YY')<MOV_RELEASEDATE
          and MOV_RELEASEDATE<to_date(ending,'DD-MON-YY')
        group by GEN_ID);

    data:= freq_genre_and_movie_count(genre_id, genre_title, countfreq);

    return data;
end;
/

declare
    data freq_genre_and_movie_count;

begin
    data:=frequent_genre( starting: '31-DEC-1940', ending: '31-DEC-1998');
    DBMS_OUTPUT.PUT_LINE( A: data.GENRE_ID || ' ' || data.GENRE_TITLE || ' ' || data.MOV_COUNT);
end;
```

Explanation:

- First create or replace a composite data type that will contain gen_id, gen_title and frequent movie counts.
- Now set that data type as the return value of the function.
- Do a subquery of finding the number of movies between the starting date and ending date and sort them according to release data. First row will contain the most frequent genre.
- Now to find the number of movies of that genre, just do a subquery to find the number of movies of each genre and in the range of given date and natural join with the previous subquery.
- Now save them is the composite datatype variable data.
- Return data.

Difficulties:

- The composite data type was showing an error that the “Reference to uninitialized composite”.
- To solve this we have to declare the data variable like this.

```
data:= freq_genre_and_movie_count(genre_id, genre_title, countfreq);
```