CSCE 5350.001

Fundamentals of Database Systems

Project Part 4

Shaik Mohammad Afroz

11660362

shaikmohammadafroz@my.unt.edu

Latest Schema

Movie (movie_id, title, avg_rating, date_of_release, duration, script_inventory_id, director, in_production)

Songs (song_id, song_name, singer_name, movie_id)

Genre (genre_id, genre_name)

TaggedWith (movie_id, genre_id)

SiteLocation (location_id, location_name, address)

ShotAt (location_id, movie_id)

Building (building_id, building_name, purpose, location_id)

PostProductionDoneIn (movie_id, building_id)

Employees (employee_id, employee_name, designation, phone_number, hourly_pay, employee_level)

Manages (employee_id, location_id)

Payroll (payroll_id, employee_id, hours_worked, date)

SponsoringCompany (company_id, company_name, num_movies_produced)

getsPaidBy (artist_id, company_id)

Produces (company_id, movie_id)

Artist (artist_id, artist_name, date_of_birth, gender, address)

MovieScriptInventory (script_inventory_id, script_inventory_name)

ActsIn(movie_id, artist_id)

Rating(rating_id, movie_id, rating)

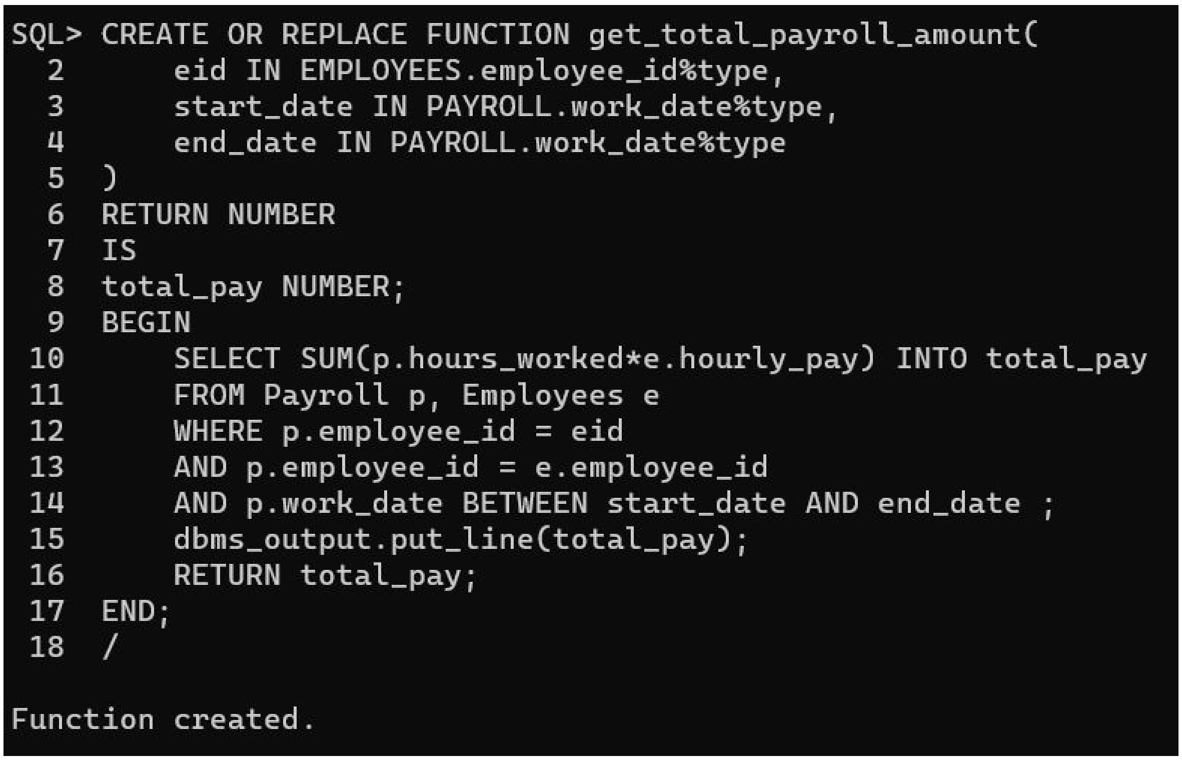
Additional Assumptions:

- Movie has avg_rating which we calculate from Rating table after insertion in Rating table.
- Employee has employee level which determines the hourly pay, every time an update on employee triggers check of employee level to change hourly pay.
- Sponsoring company table has num_movies_produced attribute that tracks number of movies produced by that company. This is going to be updated whenever a record is inserted in Produces table.

Stored Functions:

1. Get total payroll amount: Calculates employee payroll amount by multiplying hours worked with hourly pay.

```
drop function get_total_payroll_amount;

CREATE OR REPLACE FUNCTION get_total_payroll_amount(

   eid IN EMPLOYEES.employee_id%type,

   start_date IN PAYROLL.work_date%type,

   end_date IN PAYROLL.work_date%type

)

RETURN NUMBER

IS

total_pay NUMBER;

BEGIN

   SELECT SUM(p.hours_worked*e.hourly_pay) INTO total_pay

   FROM Payroll p, Employees e

   WHERE p.employee_id = eid

   AND p.employee_id = e.employee_id

   AND p.work_date BETWEEN start_date AND end_date ;

   dbms_output.put_line(total_pay);

   RETURN total_pay;

END;

/

VAR TOTAL NUMBER;

EXECUTE :TOTAL:=get_total_payroll_amount(109, '01-MAR-22', '03-NOV-22');

print TOTAL;
```

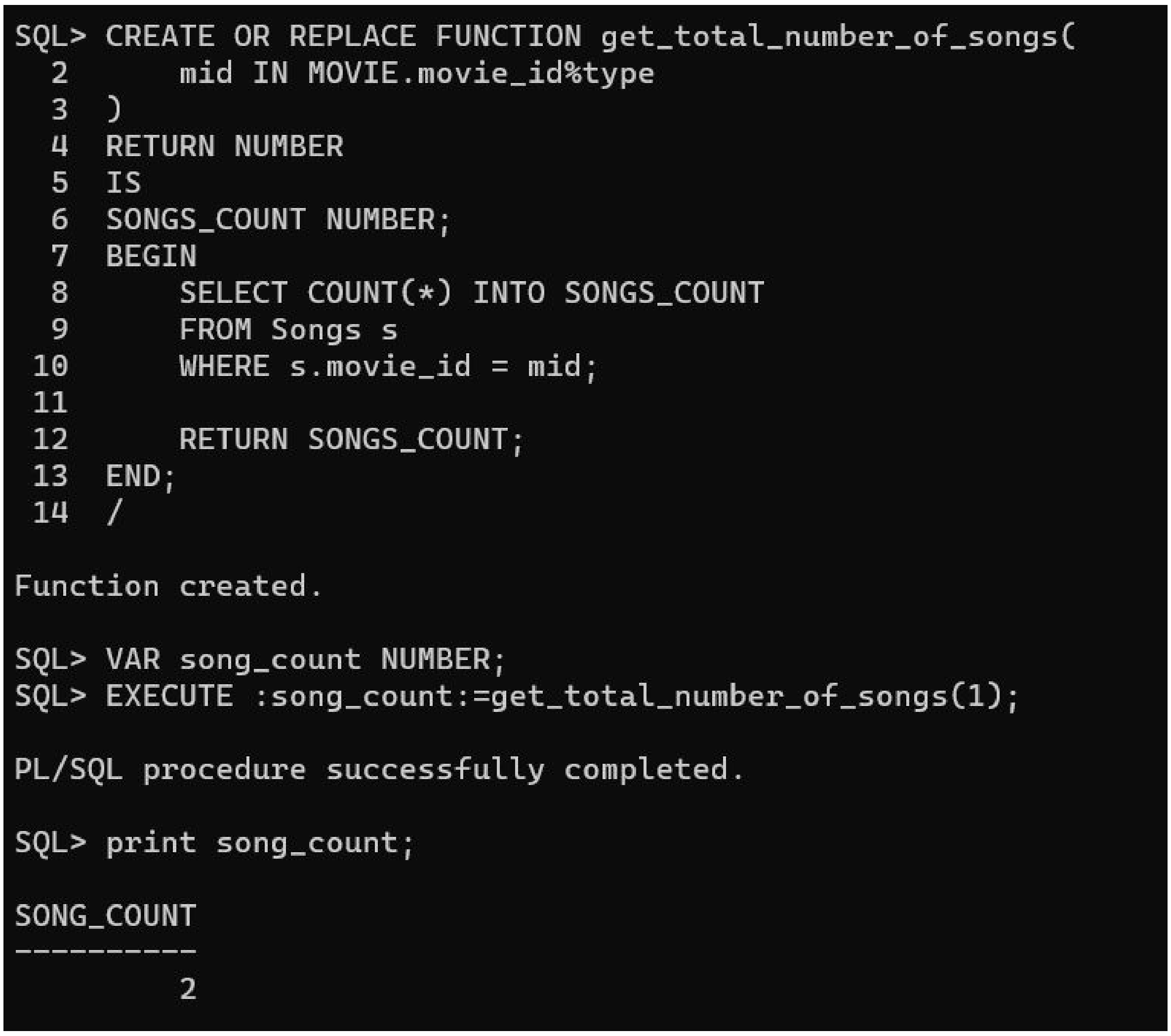


```
SQL> CREATE OR REPLACE FUNCTION get_total_payroll_amount(
         eid IN EMPLOYEES.employee_id%type,
         start_date IN PAYROLL.work_date%type,
         end_date IN PAYROLL.work_date%type
     )
     RETURN NUMBER
     IS
     total_pay NUMBER;
     BEGIN
         SELECT SUM(p.hours_worked*e.hourly_pay) INTO total_pay
         FROM Payroll p, Employees e
         WHERE p.employee_id = eid
         AND p.employee_id = e.employee_id
         AND p.work_date BETWEEN start_date AND end_date ;
         dbms_output.put_line(total_pay);
         RETURN total_pay;
     END;
     /

Function created.
```

2. Get total number of songs: Returns total number of songs using the movie id sent in function call.

```
DROP FUNCTION GET_TOTAL_NUMBER_OF_SONGS;

CREATE OR REPLACE FUNCTION get_total_number_of_songs(

   mid IN MOVIE.movie_id%type

)

RETURN NUMBER

IS

SONGS_COUNT NUMBER;

BEGIN

   SELECT COUNT(*) INTO SONGS_COUNT

   FROM Songs s

   WHERE s.movie_id = mid;

   RETURN SONGS_COUNT;

END;

/

VAR song_count NUMBER;

EXECUTE :song_count:=get_total_number_of_songs(1);
```

print song_count;

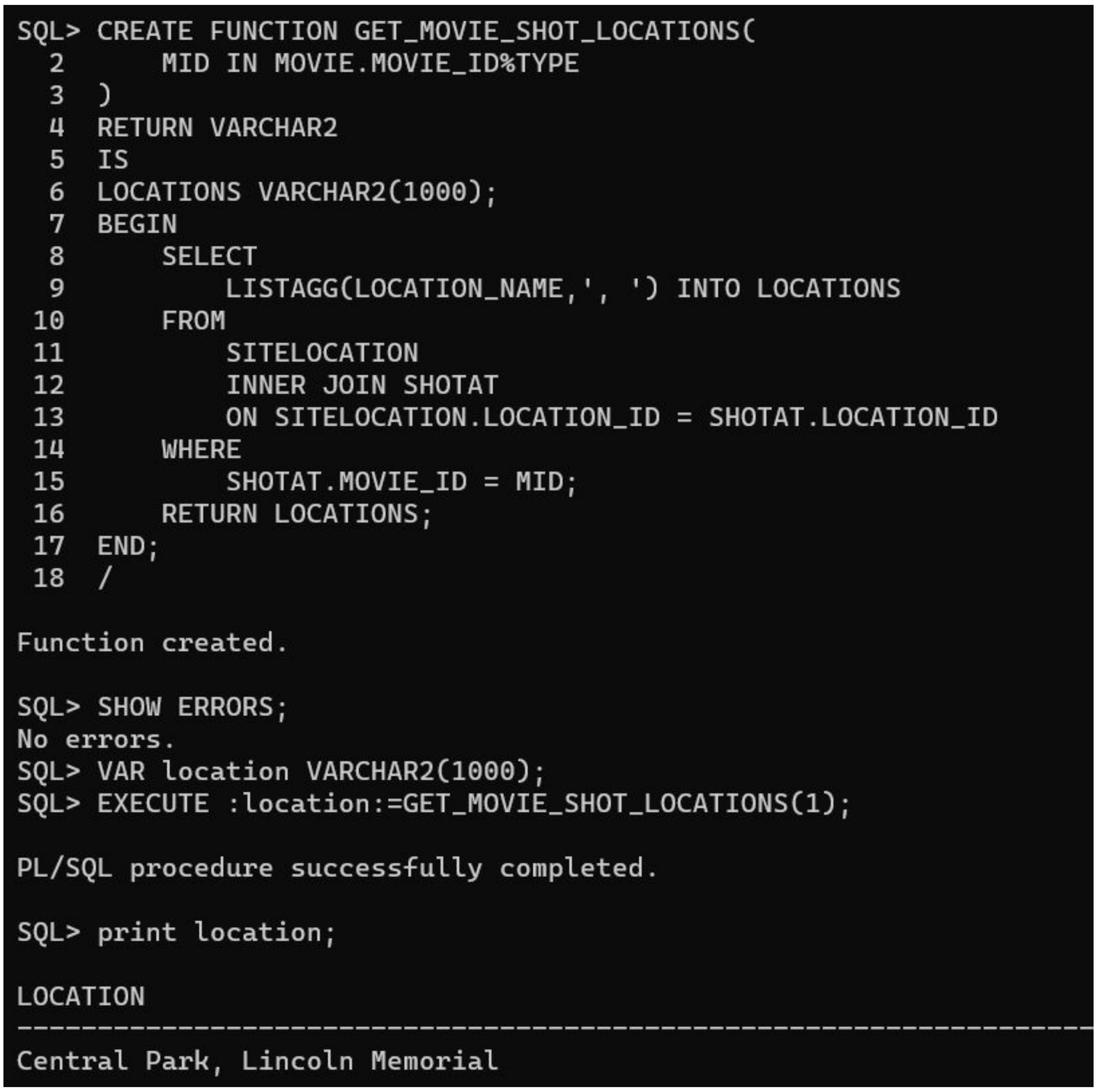


```
SQL> CREATE OR REPLACE FUNCTION get_total_number_of_songs(
        mid IN MOVIE.movie_id%type
     )
     RETURN NUMBER
     IS
     SONGS_COUNT NUMBER;
     BEGIN
         SELECT COUNT(*) INTO SONGS_COUNT
         FROM Songs s
         WHERE s.movie_id = mid;

         RETURN SONGS_COUNT;
     END;
     /

Function created.

SQL> VAR song_count NUMBER;
SQL> EXECUTE :song_count:=get_total_number_of_songs(1);

PL/SQL procedure successfully completed.

SQL> print song_count;

SONG_COUNT
----------
         2
```

3. Get movie shot locations: Returns shot locations concatenated into single string using comma separator with the help of movie id sent in function call.

```
DROP FUNCTION GET_MOVIE_SHOT_LOCATIONS;

CREATE FUNCTION GET_MOVIE_SHOT_LOCATIONS(

    MID IN MOVIE.MOVIE_ID%TYPE

)

RETURN VARCHAR2

IS

LOCATIONS VARCHAR2(1000);

BEGIN

    SELECT
```

```
        LISTAGG(LOCATION_NAME,', ') INTO LOCATIONS
    FROM
        SITELOCATION
        INNER JOIN SHOTAT
        ON SITELOCATION.LOCATION_ID = SHOTAT.LOCATION_ID
    WHERE
        SHOTAT.MOVIE_ID = MID;
    RETURN LOCATIONS;
END;
/
VAR location VARCHAR2(1000);
EXECUTE :location:=GET_MOVIE_SHOT_LOCATIONS(1);
print location;
```

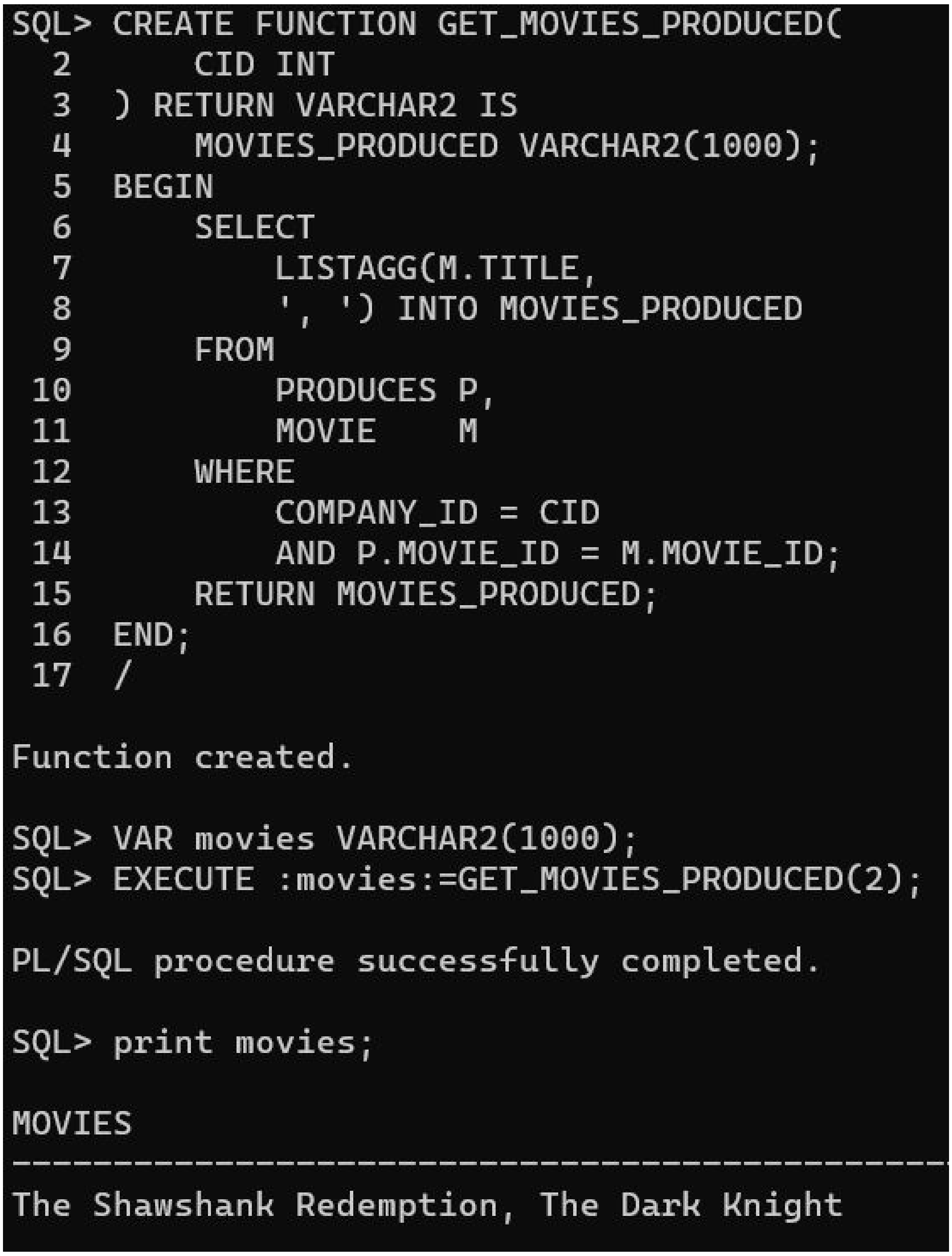


```
SQL> CREATE FUNCTION GET_MOVIE_SHOT_LOCATIONS(
         MID IN MOVIE.MOVIE_ID%TYPE
     )
     RETURN VARCHAR2
     IS
     LOCATIONS VARCHAR2(1000);
     BEGIN
         SELECT
             LISTAGG(LOCATION_NAME,', ') INTO LOCATIONS
         FROM
             SITELOCATION
             INNER JOIN SHOTAT
             ON SITELOCATION.LOCATION_ID = SHOTAT.LOCATION_ID
         WHERE
             SHOTAT.MOVIE_ID = MID;
         RETURN LOCATIONS;
     END;
     /

Function created.

SQL> SHOW ERRORS;
No errors.
SQL> VAR location VARCHAR2(1000);
SQL> EXECUTE :location:=GET_MOVIE_SHOT_LOCATIONS(1);

PL/SQL procedure successfully completed.

SQL> print location;

LOCATION
--------------------------------------------------------------------------------
Central Park, Lincoln Memorial
```

4. Get movies produced: returns movies produces concatenated into single string using the company id sent in function call.

```
DROP FUNCTION get_movies_produced;

CREATE FUNCTION GET_MOVIES_PRODUCED(

   CID INT

) RETURN VARCHAR2 IS

   MOVIES_PRODUCED VARCHAR2(1000);

BEGIN

   SELECT

      LISTAGG(M.TITLE,

      ', ') INTO MOVIES_PRODUCED
```

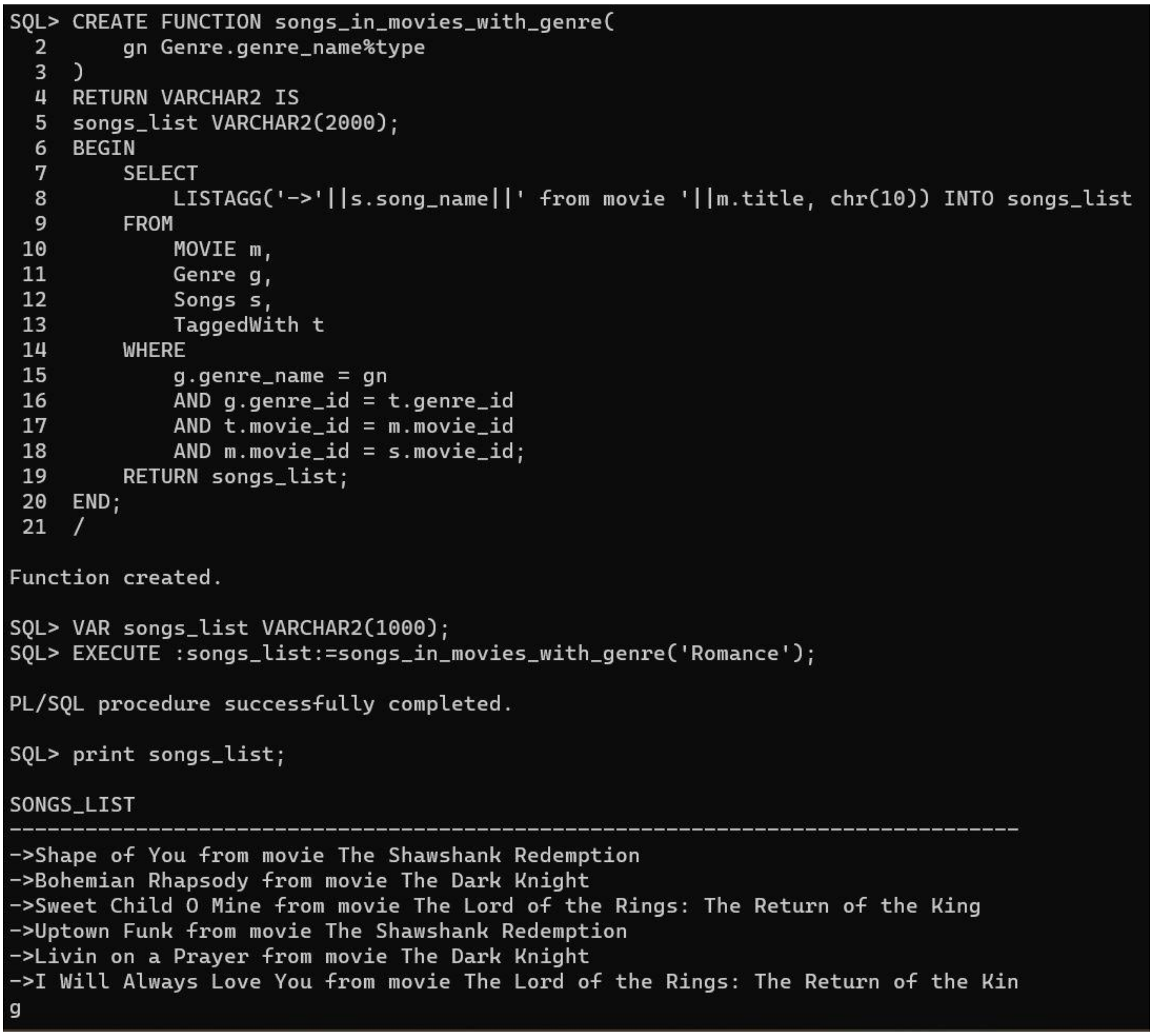
```
    FROM
        PRODUCES P,
        MOVIE    M
    WHERE
        COMPANY_ID = CID
        AND P.MOVIE_ID = M.MOVIE_ID;
    RETURN MOVIES_PRODUCED;
END;
/
VAR movies VARCHAR2(1000);
EXECUTE :movies:=GET_MOVIES_PRODUCED(2);
print movies;
```



```
SQL> CREATE FUNCTION GET_MOVIES_PRODUCED(
         CID INT
     ) RETURN VARCHAR2 IS
         MOVIES_PRODUCED VARCHAR2(1000);
     BEGIN
         SELECT
             LISTAGG(M.TITLE,
             ', ') INTO MOVIES_PRODUCED
         FROM
             PRODUCES P,
             MOVIE    M
         WHERE
             COMPANY_ID = CID
             AND P.MOVIE_ID = M.MOVIE_ID;
         RETURN MOVIES_PRODUCED;
     END;
     /

Function created.

SQL> VAR movies VARCHAR2(1000);
SQL> EXECUTE :movies:=GET_MOVIES_PRODUCED(2);

PL/SQL procedure successfully completed.

SQL> print movies;

MOVIES
--------------------------------------------------
The Shawshank Redemption, The Dark Knight
```

5. Songs in movies with genre: Returns songs list concatenated using the genre name sent to the function. Uses genre name to gather movies tagged to it. And using the movie ids forms a list to concatenate and return as result.

```
DROP FUNCTION songs_in_movies_with_genre;

-- Return songs that are present in movies with genre 'gn'

CREATE FUNCTION songs_in_movies_with_genre(

    gn Genre.genre_name%type
)
RETURN VARCHAR2 IS
songs_list VARCHAR2(2000);
BEGIN
   SELECT
      LISTAGG('->'||s.song_name||' from movie '||m.title, chr(10)) INTO songs_list
   FROM
      MOVIE m,
      Genre g,
      Songs s,
      TaggedWith t
   WHERE
      g.genre_name = gn
      AND g.genre_id = t.genre_id
      AND t.movie_id = m.movie_id
      AND m.movie_id = s.movie_id;
   RETURN songs_list;
END;
/
VAR songs_list VARCHAR2(1000);
EXECUTE :songs_list:=songs_in_movies_with_genre('Romance');
print songs_list;
```



```
SQL> CREATE FUNCTION songs_in_movies_with_genre(
        gn Genre.genre_name%type
     )
     RETURN VARCHAR2 IS
     songs_list VARCHAR2(2000);
     BEGIN
        SELECT
            LISTAGG('->'||s.song_name||' from movie '||m.title, chr(10)) INTO songs_list
        FROM
            MOVIE m,
            Genre g,
            Songs s,
            TaggedWith t
        WHERE
            g.genre_name = gn
            AND g.genre_id = t.genre_id
            AND t.movie_id = m.movie_id
            AND m.movie_id = s.movie_id;
        RETURN songs_list;
     END;
     /

Function created.

SQL> VAR songs_list VARCHAR2(1000);
SQL> EXECUTE :songs_list:=songs_in_movies_with_genre('Romance');

PL/SQL procedure successfully completed.

SQL> print songs_list;

SONGS_LIST
--------------------------------------------------------------------------------
->Shape of You from movie The Shawshank Redemption
->Bohemian Rhapsody from movie The Dark Knight
->Sweet Child O Mine from movie The Lord of the Rings: The Return of the King
->Uptown Funk from movie The Shawshank Redemption
->Livin on a Prayer from movie The Dark Knight
->I Will Always Love You from movie The Lord of the Rings: The Return of the Kin
g
```

Stored Procedures:

1) Get artist by movie: This procedure takes the movie id and gives the related artist names that are associated with that movie.

Create or replace procedure get_artist_by_movie(

v_movie_id in Movie.movie_id%type

)

Is

artist_names varchar2(2000);

Begin

Select listagg(a.artist_name,',') into artist_names

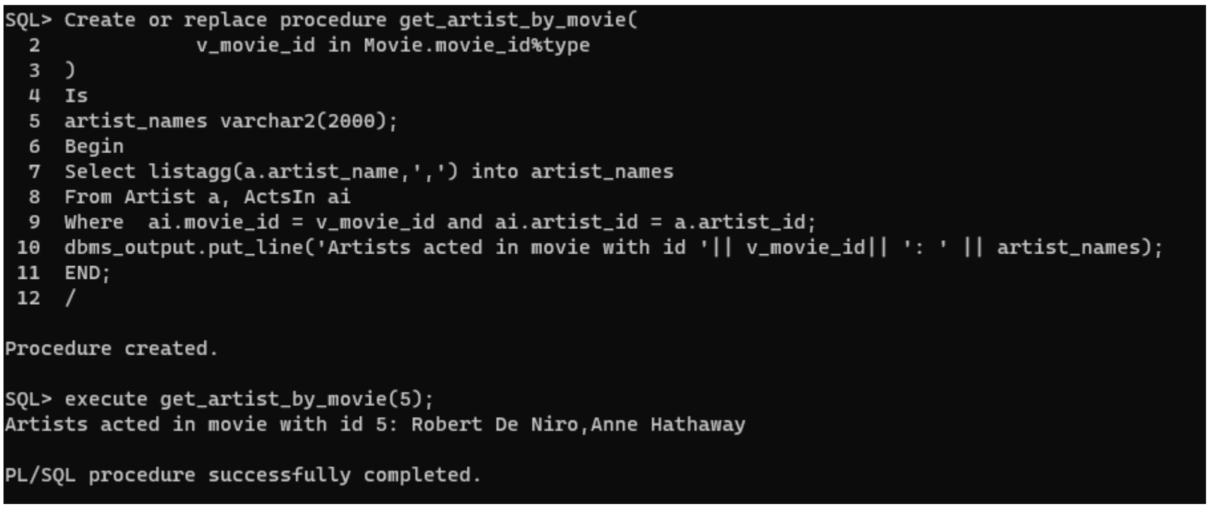
From Artist a, ActsIn ai

Where ai.movie_id = v_movie_id and ai.artist_id = a.artist_id;

dbms_output.put_line('Artists acted in movie with id '|| v_movie_id|| ': ' || artist_names);

END;

/



```
SQL> Create or replace procedure get_artist_by_movie(
             v_movie_id in Movie.movie_id%type
)
Is
artist_names varchar2(2000);
Begin
Select listagg(a.artist_name,',') into artist_names
From Artist a, ActsIn ai
Where  ai.movie_id = v_movie_id and ai.artist_id = a.artist_id;
dbms_output.put_line('Artists acted in movie with id '|| v_movie_id|| ': ' || artist_names);
END;
/

Procedure created.

SQL> execute get_artist_by_movie(5);
Artists acted in movie with id 5: Robert De Niro,Anne Hathaway

PL/SQL procedure successfully completed.
```

2) Get building by location: This stored procedure takes the input as location id and gives the names of the buildings associated with that location id.

Create or replace procedure get_building_by_location(

v_location_id in SiteLocation.location_id%type

)

is

v_building_name varchar2(255);

Begin

Select listagg(building_name,',') into v_building_name

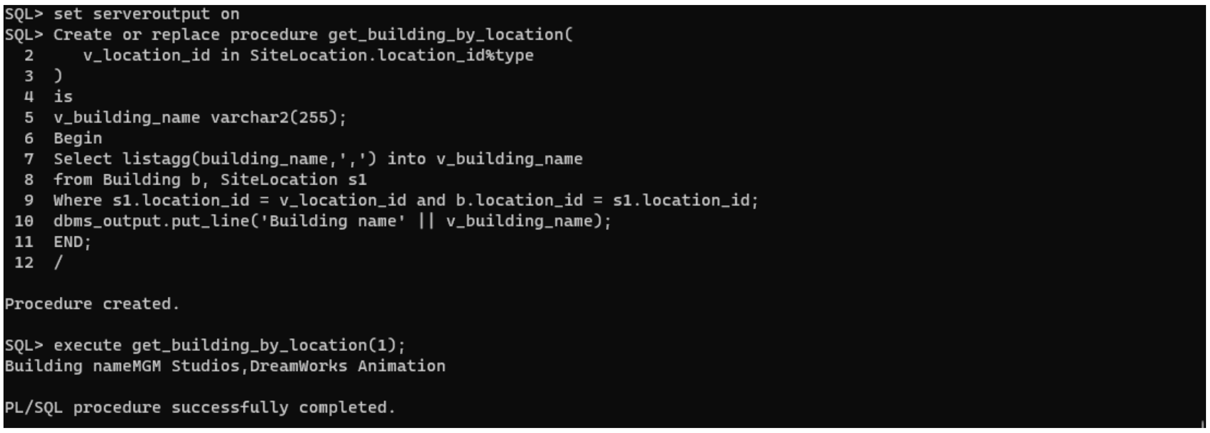
from Building b, SiteLocation s1

Where s1.location_id = v_location_id and b.location_id = s1.location_id;

dbms_output.put_line('Building name' || v_building_name);

END;

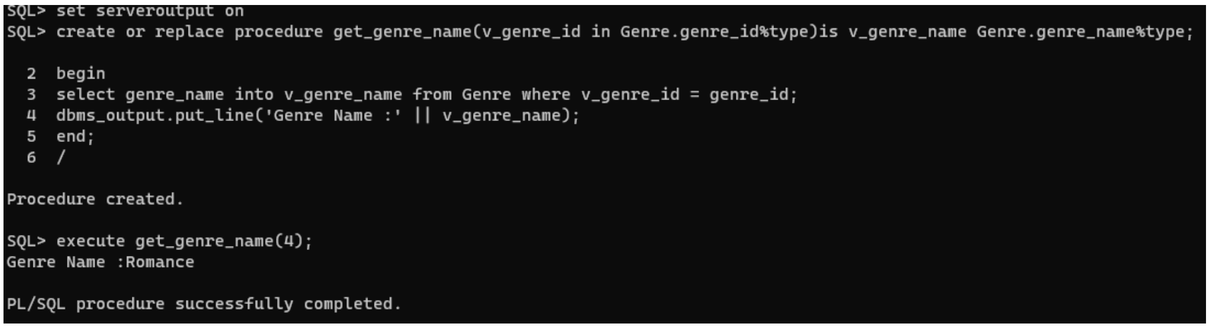
/



```
SQL> set serveroutput on
SQL> Create or replace procedure get_building_by_location(
    v_location_id in SiteLocation.location_id%type
 )
 is
 v_building_name varchar2(255);
 Begin
 Select listagg(building_name,',') into v_building_name
 from Building b, SiteLocation s1
 Where s1.location_id = v_location_id and b.location_id = s1.location_id;
 dbms_output.put_line('Building name' || v_building_name);
 END;
 /

Procedure created.

SQL> execute get_building_by_location(1);
Building nameMGM Studios,DreamWorks Animation

PL/SQL procedure successfully completed.
```

3) Get genre name: This stored procedure takes the genre id and gives the names of the genre associated with it.

```
create or replace procedure get_genre_name(

v_genre_id in Genre.genre_id%type

)

Is

v_genre_name Genre.genre_name%type;

Begin

select genre_name into v_genre_name

from Genre

where v_genre_id = genre_id;

dbms_output.put_line('Genre Name :' || v_genre_name);

END;

/
```



```
SQL> set serveroutput on
SQL> create or replace procedure get_genre_name(v_genre_id in Genre.genre_id%type)is v_genre_name Genre.genre_name%type;

 begin
 select genre_name into v_genre_name from Genre where v_genre_id = genre_id;
 dbms_output.put_line('Genre Name :' || v_genre_name);
 end;
 /

Procedure created.

SQL> execute get_genre_name(4);
Genre Name :Romance

PL/SQL procedure successfully completed.
```

4) Get Song Name: This procedure takes input as song id and gives the songs names that are related to it.

```
create or replace procedure get_song_name(

v_song_id in Songs.song_id%type
```

)

Is

 v_song_name songs.song_name%type;

BEGIN

select song_name into v_song_name

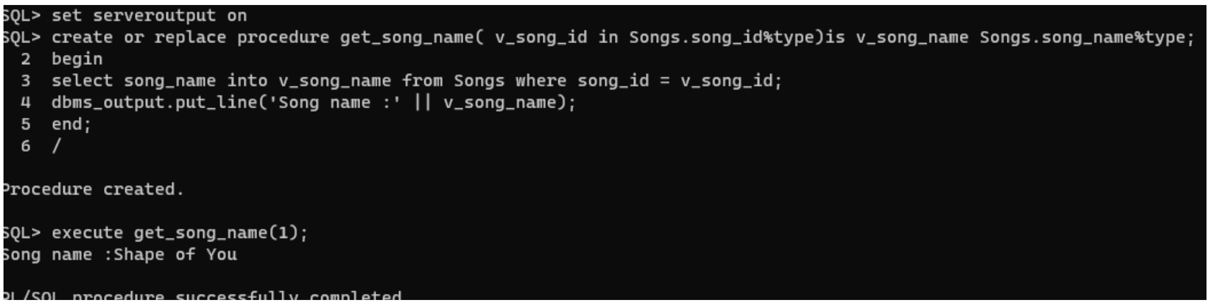
from Songs

where song_id = v_song_id;

dbms_output.put_line('Song Name :' || v_song_name);

END;

/



```
SQL> set serveroutput on
SQL> create or replace procedure get_song_name( v_song_id in Songs.song_id%type)is v_song_name Songs.song_name%type;
     begin
     select song_name into v_song_name from Songs where song_id = v_song_id;
     dbms_output.put_line('Song name :' || v_song_name);
     end;
     /

Procedure created.

SQL> execute get_song_name(1);
Song name :Shape of You

PL/SQL procedure successfully completed
```

5) Get payroll by employee: This procedure takes the employee id as input from the payroll table and gives the sum of the hours that employee worked.

Create or replace procedure get_payroll_by_employee(

v_employee_id in Employees.employee_id%type

)

Is

v_hours_worked number;

Begin

Select sum(hourly_worked) into v_hours_worked

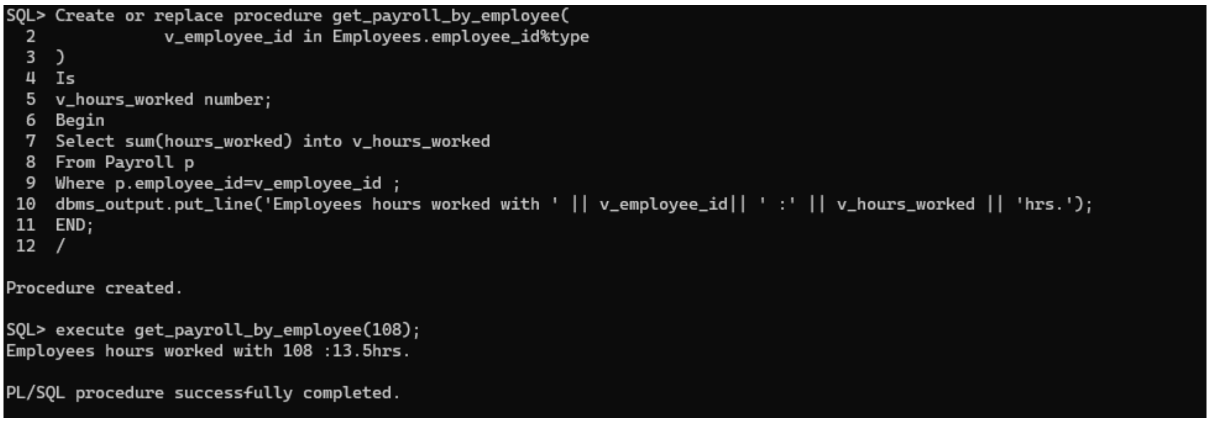
From Payroll p

Where p.employee_id = v_employee_id;

dbms_output.put_line(' Employees hours worked with' || v_employee_id|| ': ' || v_hours_worked || 'hrs.');

END;

/

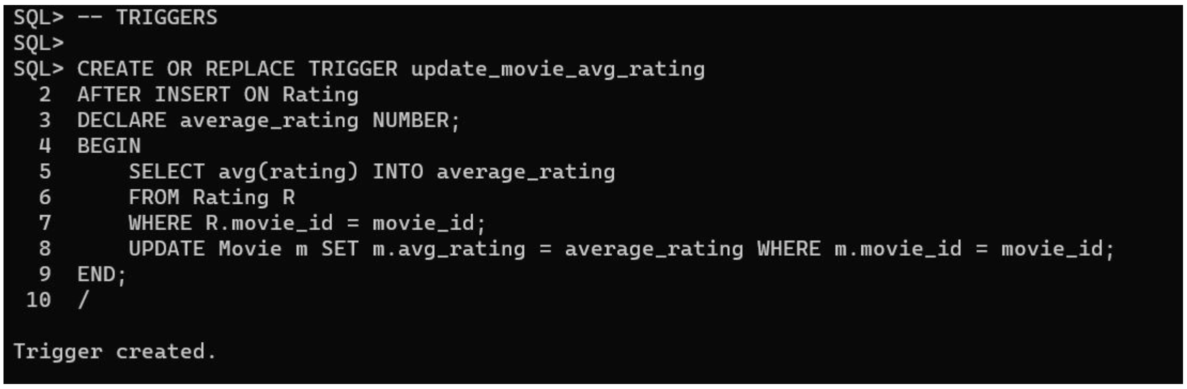


```
SQL> Create or replace procedure get_payroll_by_employee(
           v_employee_id in Employees.employee_id%type
)
Is
v_hours_worked number;
Begin
Select sum(hours_worked) into v_hours_worked
From Payroll p
Where p.employee_id=v_employee_id ;
dbms_output.put_line('Employees hours worked with ' || v_employee_id|| ' :' || v_hours_worked || 'hrs.');
END;
/

Procedure created.

SQL> execute get_payroll_by_employee(108);
Employees hours worked with 108 :13.5hrs.

PL/SQL procedure successfully completed.
```

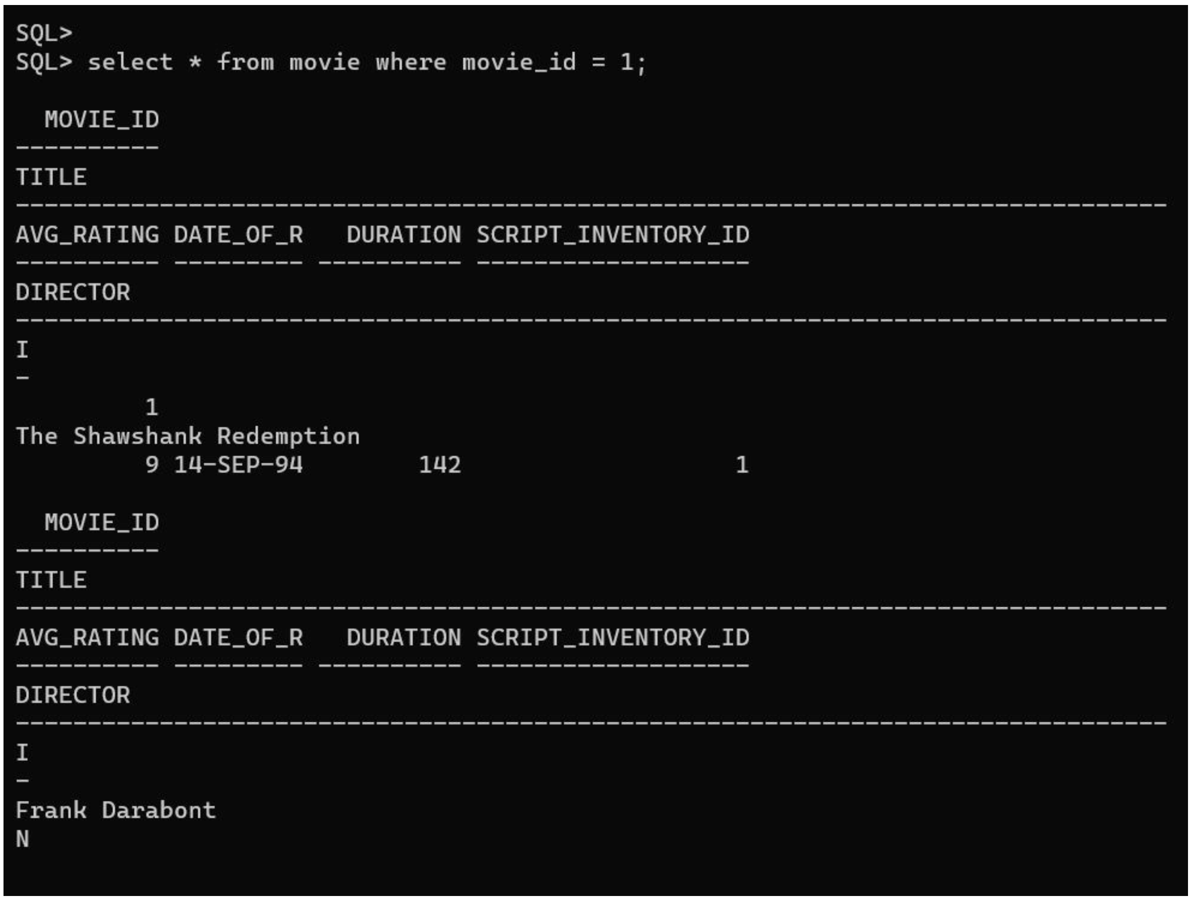
Triggers:

1) Update movie average rating: This trigger will update the average rating of the movie and if we give the movie id then it will display the details of movie.

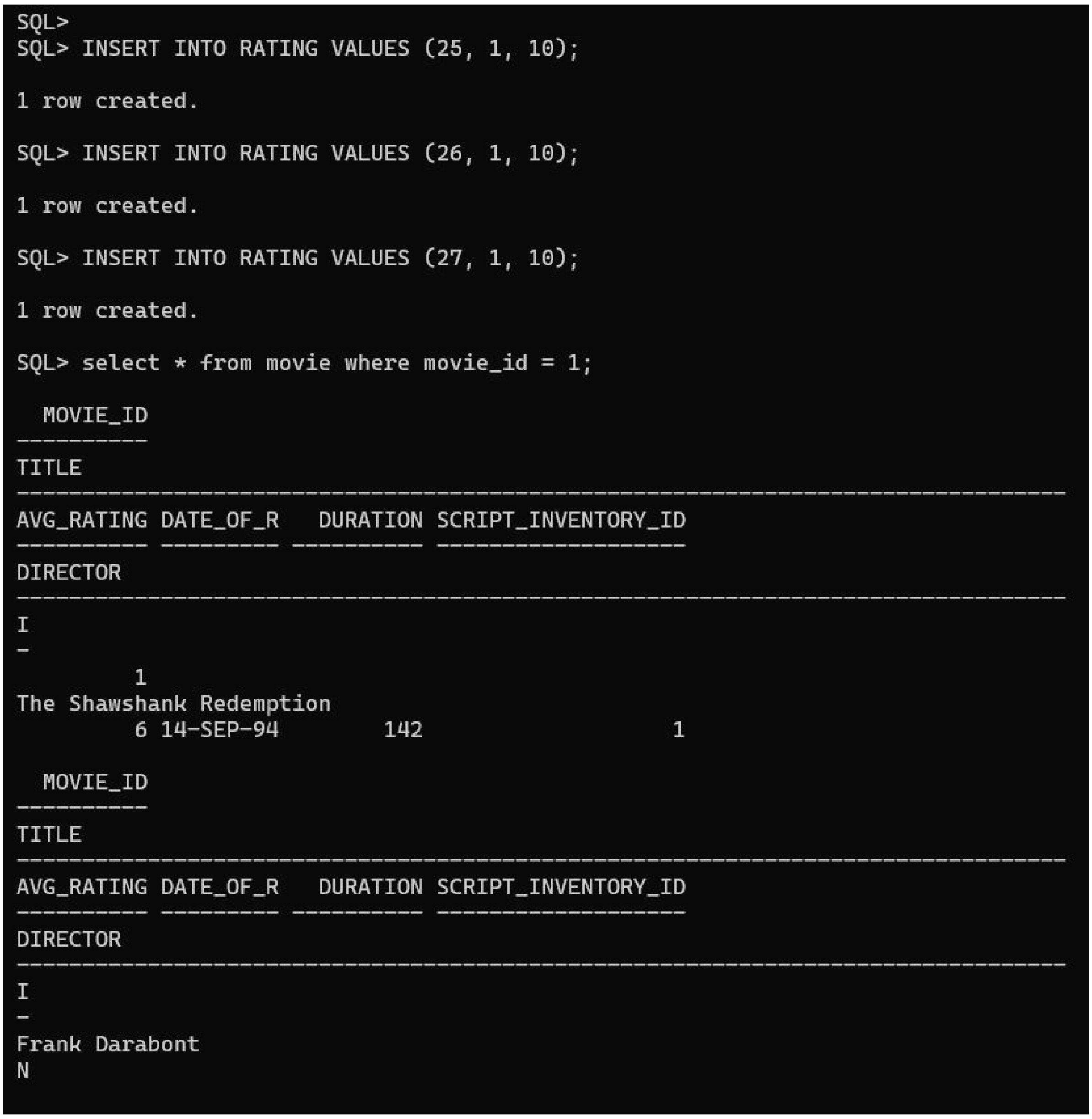
```
CREATE OR REPLACE TRIGGER update_movie_avg_rating

AFTER INSERT ON Rating

DECLARE average_rating NUMBER;

BEGIN

    SELECT avg(rating) INTO average_rating

    FROM Rating R

    WHERE R.movie_id = movie_id;

    UPDATE Movie m SET m.avg_rating = average_rating WHERE m.movie_id = movie_id;

END;

/
```



```
SQL> -- TRIGGERS
SQL>
SQL> CREATE OR REPLACE TRIGGER update_movie_avg_rating
AFTER INSERT ON Rating
DECLARE average_rating NUMBER;
BEGIN
    SELECT avg(rating) INTO average_rating
    FROM Rating R
    WHERE R.movie_id = movie_id;
    UPDATE Movie m SET m.avg_rating = average_rating WHERE m.movie_id = movie_id;
END;
/

Trigger created.
```



```
SQL>
SQL> select * from movie where movie_id = 1;

  MOVIE_ID
----------
TITLE
--------------------------------------------------------------------------------
AVG_RATING DATE_OF_R   DURATION SCRIPT_INVENTORY_ID
---------- --------- ---------- -------------------
DIRECTOR
--------------------------------------------------------------------------------
I
-
         1
The Shawshank Redemption
         9 14-SEP-94        142                   1

  MOVIE_ID
----------
TITLE
--------------------------------------------------------------------------------
AVG_RATING DATE_OF_R   DURATION SCRIPT_INVENTORY_ID
---------- --------- ---------- -------------------
DIRECTOR
--------------------------------------------------------------------------------
I
-
Frank Darabont
N
```



```
SQL>
SQL> INSERT INTO RATING VALUES (25, 1, 10);

1 row created.

SQL> INSERT INTO RATING VALUES (26, 1, 10);

1 row created.

SQL> INSERT INTO RATING VALUES (27, 1, 10);

1 row created.

SQL> select * from movie where movie_id = 1;

  MOVIE_ID
----------
TITLE
--------------------------------------------------------------------------------
AVG_RATING DATE_OF_R   DURATION SCRIPT_INVENTORY_ID
---------- --------- ---------- -------------------
DIRECTOR
--------------------------------------------------------------------------------
I
-
         1
The Shawshank Redemption
         6 14-SEP-94        142                   1

  MOVIE_ID
----------
TITLE
--------------------------------------------------------------------------------
AVG_RATING DATE_OF_R   DURATION SCRIPT_INVENTORY_ID
---------- --------- ---------- -------------------
DIRECTOR
--------------------------------------------------------------------------------
I
-
Frank Darabont
N
```

2) Update employee hourly pay: This trigger will update the employee level and the hourly pay of the employee and gives the updates details of employees.

SELECT EMPLOYEE_LEVEL, HOURLY_PAY FROM EMPLOYEES WHERE EMPLOYEE_ID = 101;

CREATE OR REPLACE TRIGGER update_employee_hourly_pay

BEFORE UPDATE ON Employees

FOR EACH ROW

BEGIN

IF :NEW.employee_level = 'Manager' THEN

:NEW.hourly_pay := 50;

ELSIF :NEW.employee_level = 'Assistant Manager' THEN

:NEW.hourly_pay := 30;

ELSIF :NEW.employee_level = 'Production Assistant' THEN

:NEW.hourly_pay := 15;

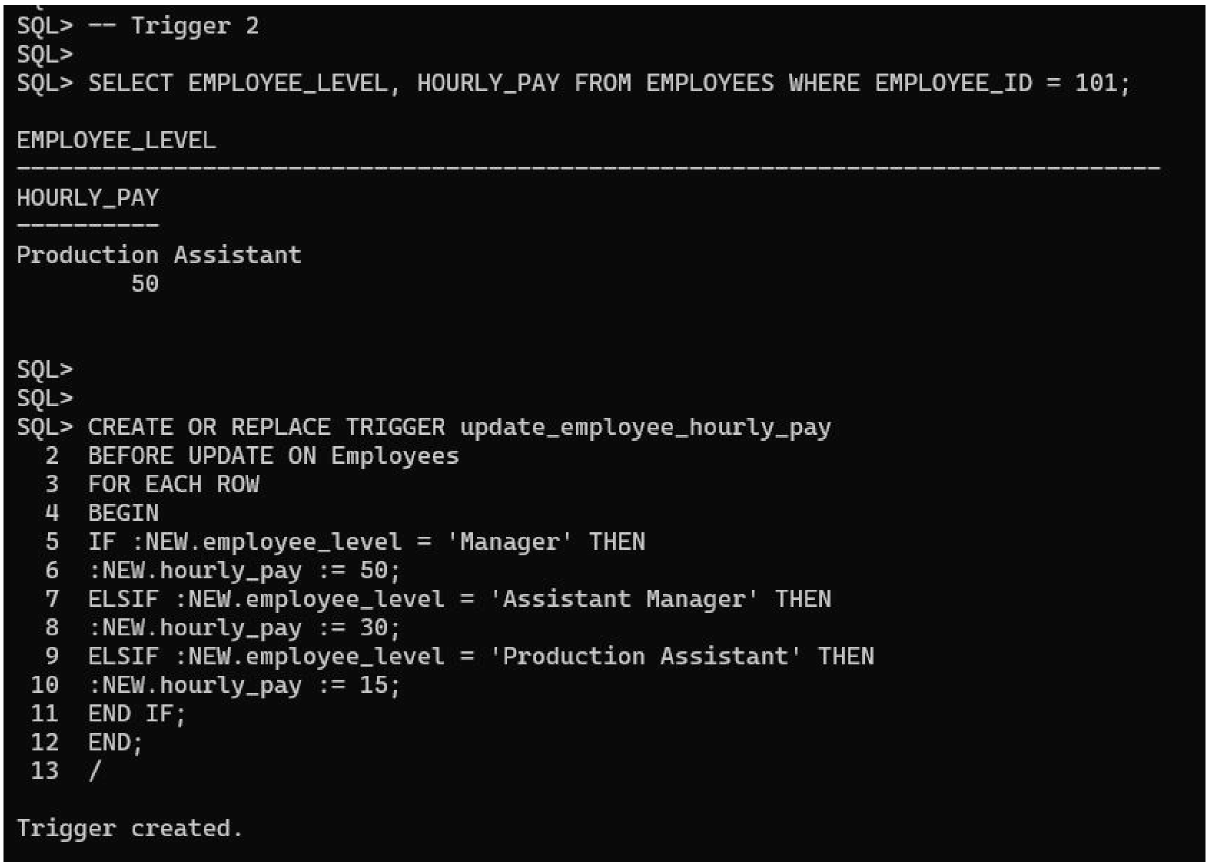
END IF;

END;

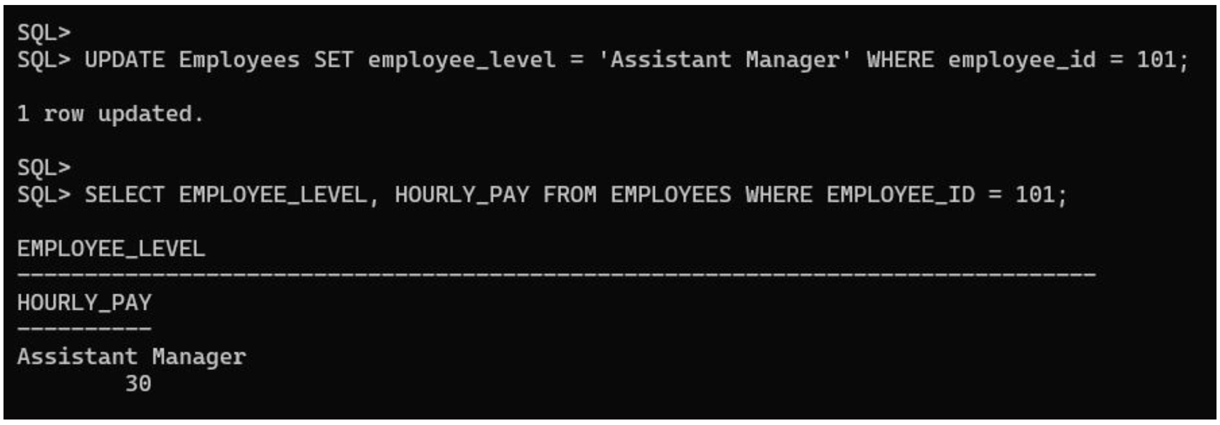
/

UPDATE Employees SET employee_level = 'Assistant Manager' WHERE employee_id = 101;

SELECT EMPLOYEE_LEVEL, HOURLY_PAY FROM EMPLOYEES WHERE EMPLOYEE_ID = 101;



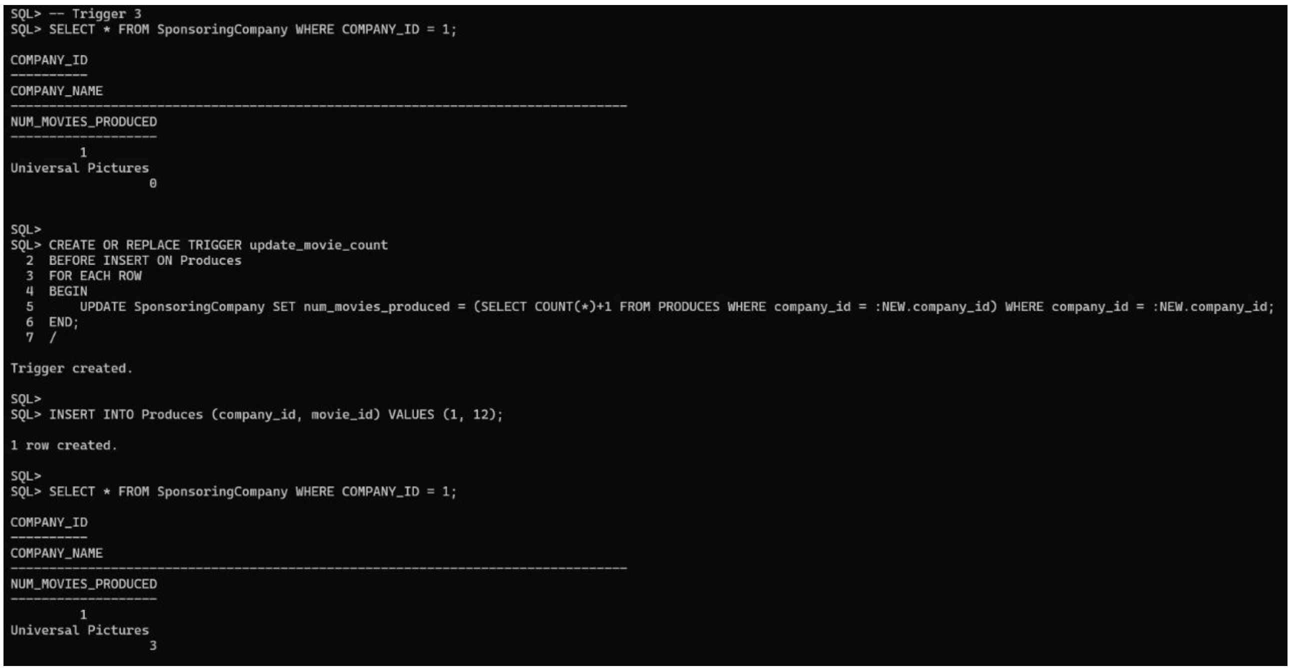
```
SQL> -- Trigger 2
SQL>
SQL> SELECT EMPLOYEE_LEVEL, HOURLY_PAY FROM EMPLOYEES WHERE EMPLOYEE_ID = 101;

EMPLOYEE_LEVEL
--------------------------------------------------------------------------------
HOURLY_PAY
----------
Production Assistant
        50


SQL>
SQL>
SQL> CREATE OR REPLACE TRIGGER update_employee_hourly_pay
  2  BEFORE UPDATE ON Employees
  3  FOR EACH ROW
  4  BEGIN
  5  IF :NEW.employee_level = 'Manager' THEN
  6  :NEW.hourly_pay := 50;
  7  ELSIF :NEW.employee_level = 'Assistant Manager' THEN
  8  :NEW.hourly_pay := 30;
  9  ELSIF :NEW.employee_level = 'Production Assistant' THEN
 10  :NEW.hourly_pay := 15;
 11  END IF;
 12  END;
 13  /

Trigger created.
```



```
SQL>
SQL> UPDATE Employees SET employee_level = 'Assistant Manager' WHERE employee_id = 101;

1 row updated.

SQL>
SQL> SELECT EMPLOYEE_LEVEL, HOURLY_PAY FROM EMPLOYEES WHERE EMPLOYEE_ID = 101;

EMPLOYEE_LEVEL
--------------------------------------------------------------------------------
HOURLY_PAY
----------
Assistant Manager
        30
```

3) Update movie count: This trigger will update the movie count based on company id from the sponsoring company table and displays all the details of it.

```
SELECT * FROM SponsoringCompany WHERE COMPANY_ID = 1;


CREATE OR REPLACE TRIGGER update_movie_count

BEFORE INSERT ON Produces

FOR EACH ROW

BEGIN

    UPDATE SponsoringCompany SET num_movies_produced =

             (SELECT COUNT(*)+1

               FROM PRODUCES

             WHERE company_id = :NEW.company_id)

   WHERE company_id = :NEW.company_id;

END;

/


INSERT INTO Produces (company_id, movie_id) VALUES (1, 12);


SELECT * FROM SponsoringCompany WHERE COMPANY_ID = 1;
```



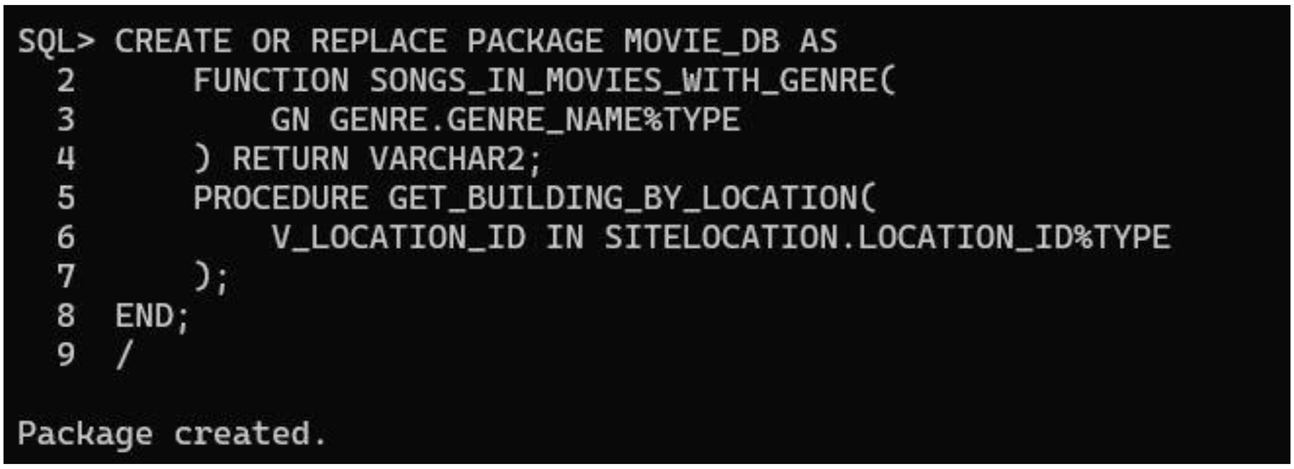
```
SQL> -- Trigger 3
SQL> SELECT * FROM SponsoringCompany WHERE COMPANY_ID = 1;

COMPANY_ID
----------
COMPANY_NAME
--------------------------------------------------------------------------------
NUM_MOVIES_PRODUCED
-------------------
         1
Universal Pictures
                  0


SQL>
SQL> CREATE OR REPLACE TRIGGER update_movie_count
  BEFORE INSERT ON Produces
  FOR EACH ROW
  BEGIN
      UPDATE SponsoringCompany SET num_movies_produced = (SELECT COUNT(*)+1 FROM PRODUCES WHERE company_id = :NEW.company_id) WHERE company_id = :NEW.company_id;
  END;
  /

Trigger created.

SQL>
SQL> INSERT INTO Produces (company_id, movie_id) VALUES (1, 12);

1 row created.

SQL>
SQL> SELECT * FROM SponsoringCompany WHERE COMPANY_ID = 1;

COMPANY_ID
----------
COMPANY_NAME
--------------------------------------------------------------------------------
NUM_MOVIES_PRODUCED
-------------------
         1
Universal Pictures
                  3
```

Package:

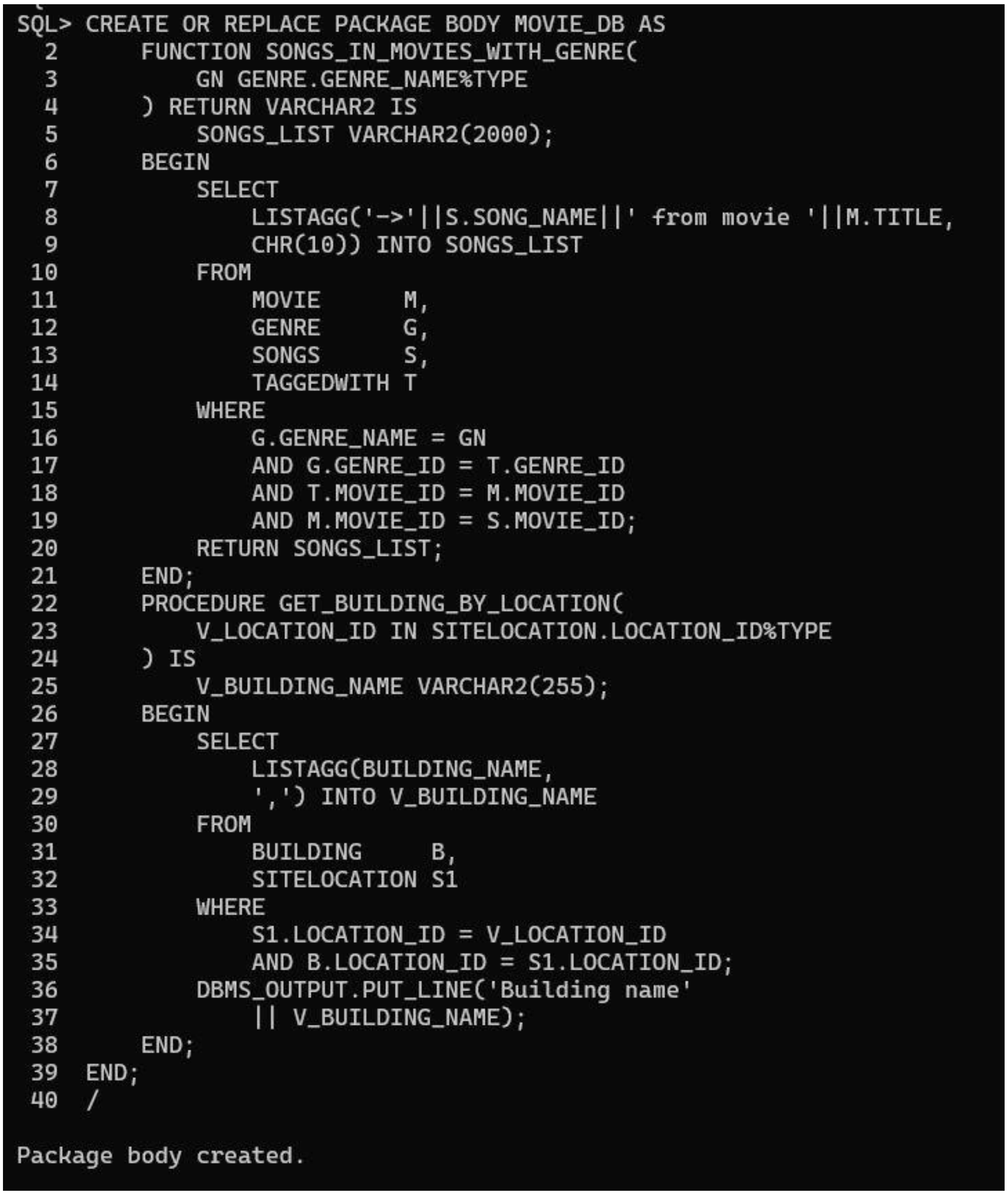
1)

```
CREATE OR REPLACE PACKAGE MOVIE_DB AS
   FUNCTION SONGS_IN_MOVIES_WITH_GENRE(
      GN GENRE.GENRE_NAME%TYPE
   ) RETURN VARCHAR2;
   PROCEDURE GET_BUILDING_BY_LOCATION(
      V_LOCATION_ID IN SITELOCATION.LOCATION_ID%TYPE
   );
END;
/
SHOW ERRORS;

CREATE OR REPLACE PACKAGE BODY MOVIE_DB AS
   FUNCTION SONGS_IN_MOVIES_WITH_GENRE(
      GN GENRE.GENRE_NAME%TYPE
   ) RETURN VARCHAR2 IS
      SONGS_LIST VARCHAR2(2000);
   BEGIN
      SELECT
         LISTAGG('->'||S.SONG_NAME||' from movie '||M.TITLE,
         CHR(10)) INTO SONGS_LIST
      FROM
         MOVIE     M,
         GENRE     G,
         SONGS     S,
         TAGGEDWITH T
      WHERE
         G.GENRE_NAME = GN
         AND G.GENRE_ID = T.GENRE_ID
         AND T.MOVIE_ID = M.MOVIE_ID
         AND M.MOVIE_ID = S.MOVIE_ID;
```

```
        RETURN SONGS_LIST;
    END;
    PROCEDURE GET_BUILDING_BY_LOCATION(
        V_LOCATION_ID IN SITELOCATION.LOCATION_ID%TYPE
    ) IS
        V_BUILDING_NAME VARCHAR2(255);
    BEGIN
        SELECT
            LISTAGG(BUILDING_NAME,
            ',') INTO V_BUILDING_NAME
        FROM
            BUILDING     B,
            SITELOCATION S1
        WHERE
            S1.LOCATION_ID = V_LOCATION_ID
            AND B.LOCATION_ID = S1.LOCATION_ID;
        DBMS_OUTPUT.PUT_LINE('Building name'
            || V_BUILDING_NAME);
    END;
END;
/
SHOW ERRORS;
```



```
SQL> CREATE OR REPLACE PACKAGE MOVIE_DB AS
         FUNCTION SONGS_IN_MOVIES_WITH_GENRE(
             GN GENRE.GENRE_NAME%TYPE
         ) RETURN VARCHAR2;
         PROCEDURE GET_BUILDING_BY_LOCATION(
             V_LOCATION_ID IN SITELOCATION.LOCATION_ID%TYPE
         );
     END;
     /

Package created.
```



```
SQL> CREATE OR REPLACE PACKAGE BODY MOVIE_DB AS
        FUNCTION SONGS_IN_MOVIES_WITH_GENRE(
            GN GENRE.GENRE_NAME%TYPE
        ) RETURN VARCHAR2 IS
            SONGS_LIST VARCHAR2(2000);
        BEGIN
            SELECT
                LISTAGG('->'||S.SONG_NAME||' from movie '||M.TITLE,
                CHR(10)) INTO SONGS_LIST
            FROM
                MOVIE      M,
                GENRE      G,
                SONGS      S,
                TAGGEDWITH T
            WHERE
                G.GENRE_NAME = GN
                AND G.GENRE_ID = T.GENRE_ID
                AND T.MOVIE_ID = M.MOVIE_ID
                AND M.MOVIE_ID = S.MOVIE_ID;
            RETURN SONGS_LIST;
        END;
        PROCEDURE GET_BUILDING_BY_LOCATION(
            V_LOCATION_ID IN SITELOCATION.LOCATION_ID%TYPE
        ) IS
            V_BUILDING_NAME VARCHAR2(255);
        BEGIN
            SELECT
                LISTAGG(BUILDING_NAME,
                ',') INTO V_BUILDING_NAME
            FROM
                BUILDING     B,
                SITELOCATION S1
            WHERE
                S1.LOCATION_ID = V_LOCATION_ID
                AND B.LOCATION_ID = S1.LOCATION_ID;
            DBMS_OUTPUT.PUT_LINE('Building name'
                || V_BUILDING_NAME);
        END;
    END;
    /

Package body created.
```

Individual Contribution:

Created package and package body.

Created Function to return payroll amount of employee based on employee id, start_date, and end_date.

Create Procedure to print artists in movie using movie id.

Created Trigger to calculate average rating after inserting into Rating table and assigning movie with average rating.

Created Rating table schema to accommodate ratings for movies.