Report on Lab-10

DATABASE MANAGEMENT SYSTEMS LAB

Submitted by

Adid-Al-Mahamud Shazid

Student Id: 210042172

Department: CSE

Programme: SWE

Course Title: CSE 4308

Submitted to

Zannatun Naim Sristy

Lecturer, Department of CSE

November 15, 2023

Introduction

In the lab class, we were given task to perform command on PL/SQL.

Task 1

(a) Print your name.

(b) Take your student ID as input and print its length.

(c) Take two numbers as input and print their product.

(d) Print the current system time in 12-hour format.

(e) Take a number as input and print whether it is a whole number or a fraction.

(f) Write a procedure that takes a number as an argument and prints whether it is a composite number or not.

1.1 Solution

```
SET SERVEROUTPUT ON SIZE 1000000;
SET VERIFY OFF;

--a--
BEGIN
DBMS_OUTPUT.PUT_LINE('Adid Al Mahamud Shazid');
END;
/

--b--
DECLARE
ID VARCHAR2 (20);
BEGIN
ID := '&Student_Id';
DBMS_OUTPUT.PUT_LINE('Student Id Length: ' || LENGTH(ID));
END ;
/
```

```
--c--
DECLARE
Num1 NUMBER;
Num2 NUMBER;
Numsum NUMBER;
BEGIN
Num1 := '&Num1';
Num2 := '&Num2';
Numsum := Num1+ Num2;
DBMS_OUTPUT.PUT_LINE( 'Sum= ' || Numsum);
END ;
/

---d--
DECLARE
nowTime TIMESTAMP;
BEGIN
nowTime:= SYSTIMESTAMP;
DBMS_OUTPUT . PUT_LINE ('Current Time: ' || TO_CHAR ( nowTime, 'HH :MI:SS AM'));
END ;
/

--e--
DECLARE
Num1 NUMBER;
BEGIN
Num1 := '&Number';
IF MOD(num1,1) = 0
THEN DBMS_OUTPUT.PUT_LINE( Num1 || ' is a Whole Number');
ELSE
DBMS_OUTPUT.PUT_LINE( Num1 || ' is a Fraction');
END IF;
END ;
/

--f--
CREATE OR REPLACE
FUNCTION Check_Composite (num NUMBER)
RETURN BOOLEAN
IS
BEGIN
IF (num <= 1)
THEN RETURN FALSE;
END IF;
FOR i IN 2..ROUND(SQRT(num)) LOOP
```

```
        IF MOD(num, i) = 0 THEN
            RETURN TRUE;
        END IF;
    END LOOP;
RETURN FALSE;
END ;
/
-- Call it from an anonymous block
DECLARE
num NUMBER;
res BOOLEAN;
BEGIN
num:= '&number';
res:= Check_Composite(num);
IF res
THEN DBMS_OUTPUT.PUT_LINE(num || ' is Composite');
ELSE DBMS_OUTPUT.PUT_LINE(num || ' is Prime');
END IF;
END ;
/
```

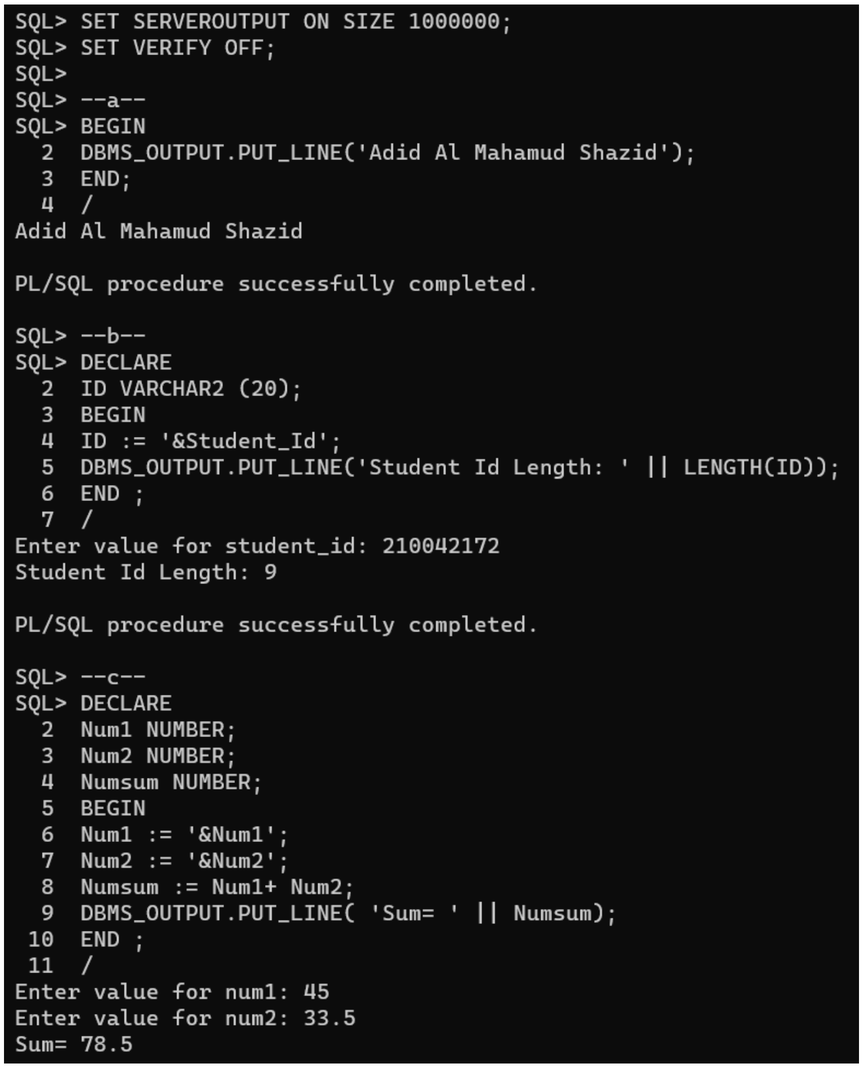
1.2 Analysis and Explanation

The task was given to getting familiar with the PL/SQL basic commands.

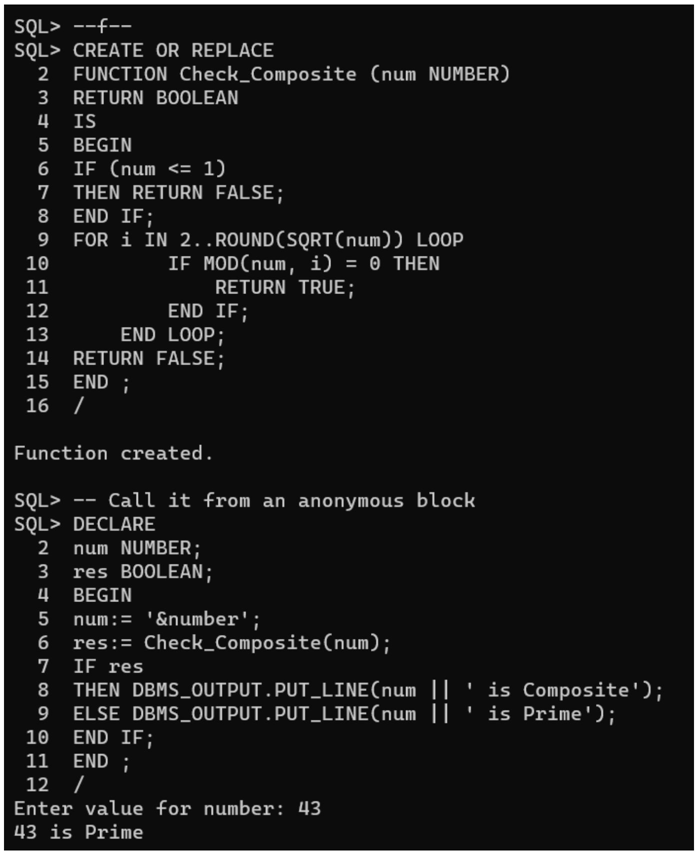
1.3 Difficulties

I did not face any difficulties when doing this task as all of the guidelines I got from the PL/SQL notes file.

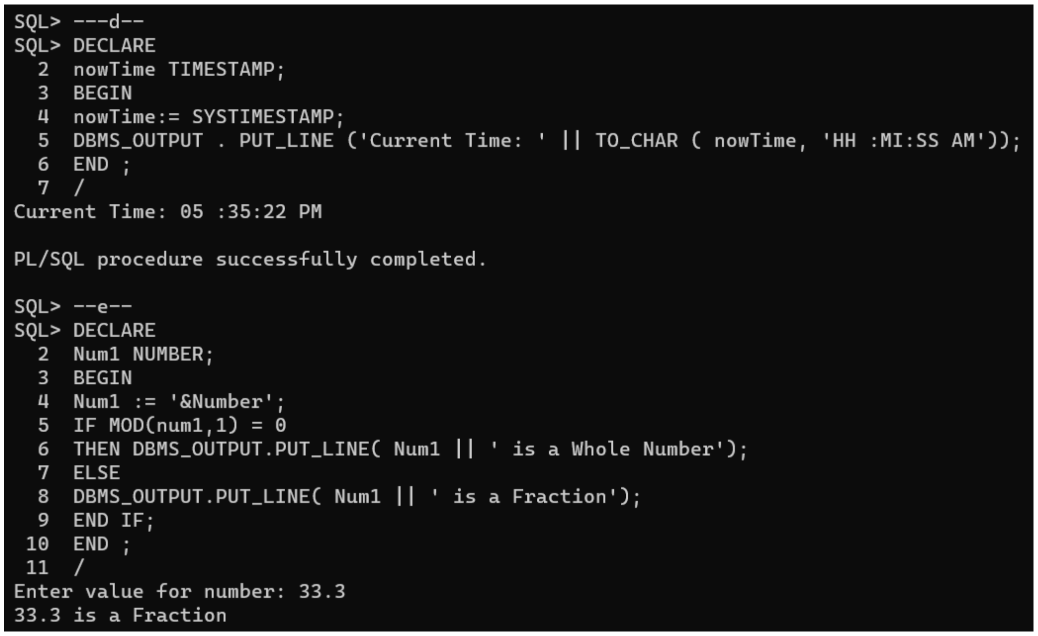
1.4 Output



```
SQL> SET SERVEROUTPUT ON SIZE 1000000;
SQL> SET VERIFY OFF;
SQL>
SQL> --a--
SQL> BEGIN
DBMS_OUTPUT.PUT_LINE('Adid Al Mahamud Shazid');
END;
/
Adid Al Mahamud Shazid

PL/SQL procedure successfully completed.

SQL> --b--
SQL> DECLARE
ID VARCHAR2 (20);
BEGIN
ID := '&Student_Id';
DBMS_OUTPUT.PUT_LINE('Student Id Length: ' || LENGTH(ID));
END ;
/
Enter value for student_id: 210042172
Student Id Length: 9

PL/SQL procedure successfully completed.

SQL> --c--
SQL> DECLARE
Num1 NUMBER;
Num2 NUMBER;
Numsum NUMBER;
BEGIN
Num1 := '&Num1';
Num2 := '&Num2';
Numsum := Num1+ Num2;
DBMS_OUTPUT.PUT_LINE( 'Sum= ' || Numsum);
END ;
/
Enter value for num1: 45
Enter value for num2: 33.5
Sum= 78.5
```



```
SQL> --f--
SQL> CREATE OR REPLACE
FUNCTION Check_Composite (num NUMBER)
RETURN BOOLEAN
IS
BEGIN
IF (num <= 1)
THEN RETURN FALSE;
END IF;
FOR i IN 2..ROUND(SQRT(num)) LOOP
        IF MOD(num, i) = 0 THEN
            RETURN TRUE;
        END IF;
    END LOOP;
RETURN FALSE;
END ;
/

Function created.

SQL> -- Call it from an anonymous block
SQL> DECLARE
num NUMBER;
res BOOLEAN;
BEGIN
num:= '&number';
res:= Check_Composite(num);
IF res
THEN DBMS_OUTPUT.PUT_LINE(num || ' is Composite');
ELSE DBMS_OUTPUT.PUT_LINE(num || ' is Prime');
END IF;
END ;
/
Enter value for number: 43
43 is Prime
```



```
SQL> ---d--
SQL> DECLARE
nowTime TIMESTAMP;
BEGIN
nowTime:= SYSTIMESTAMP;
DBMS_OUTPUT . PUT_LINE ('Current Time: ' || TO_CHAR ( nowTime, 'HH :MI:SS AM'));
END ;
/
Current Time: 05 :35:22 PM

PL/SQL procedure successfully completed.

SQL> --e--
SQL> DECLARE
Num1 NUMBER;
BEGIN
Num1 := '&Number';
IF MOD(num1,1) = 0
THEN DBMS_OUTPUT.PUT_LINE( Num1 || ' is a Whole Number');
ELSE
DBMS_OUTPUT.PUT_LINE( Num1 || ' is a Fraction');
END IF;
END ;
/
Enter value for number: 33.3
33.3 is a Fraction
```

Task 2

Execute the provided “movie.sql” file and answer the following questions:

(a) Write a procedure to find the N top-rated movies and their details (Top-rated means top highest average rating). The procedure will take N as input and print the details up to N movies. If N is greater than the number of movies, then it will print an error message.

(b) Write a function to find the movie status (“Solo”, “Ensemble”). If the total number of actors/actresses in a movie is 1, then the status should be “Solo”, else it should be “Ensemble”. The function will take the title of the movie as input as input and return the status.

(c) Write a procedure to find the possible nominees for the Oscars. A director is eligible for an Oscar if at least one of their movies has an average rating of at least 7. Also, the movie should be reviewed by more than 10 reviewers.

(d) Write a function that will take the title of the movie as input and find the movie category based on Table 1.

2.1 Solution

```
--a--

-- Procedure to Find N Top movie rating
CREATE OR REPLACE PROCEDURE Top_Movies(N IN NUMBER) IS
    movie_count NUMBER;
  BEGIN
    SELECT COUNT(*)
    INTO movie_count
    FROM Movie;

    IF N > movie_count THEN
        DBMS_OUTPUT.PUT_LINE('Error: Overflow');
        RETURN;
    END IF;

    movie_count := 0;
    FOR movie_rating IN (
        SELECT m.mov_id, m.mov_title, NVL(AVG(r.rev_stars),0) as avg_rating
        FROM Movie m LEFT JOIN Rating r ON m.mov_id = r.mov_id
        GROUP BY m.mov_id, m.mov_title
        ORDER BY avg_rating DESC
    )
    LOOP
        DBMS_OUTPUT.PUT_LINE('Movie ID: ' || movie_rating.mov_id);
        DBMS_OUTPUT.PUT_LINE('Title: ' || movie_rating.mov_title);
        DBMS_OUTPUT.PUT_LINE('Average Rating: ' || ROUND(movie_rating.avg_rating,
2));
        DBMS_OUTPUT.PUT_LINE('------------------------');
        movie_count:= movie_count + 1;
        EXIT WHEN movie_count>= n;
    END LOOP;
END;
/

-- Call it from an anonymous block
DECLARE
    N NUMBER;
BEGIN
    N:= '&N';
    Top_Movies(N);
END;
/
```

```sql
-- b--
CREATE OR REPLACE
FUNCTION movie_Status (movie_title VARCHAR2)
RETURN VARCHAR2
IS
act_count NUMBER;
BEGIN
SELECT COUNT(c.ACT_ID) INTO act_count
FROM MOVIE m LEFT JOIN CASTS c ON m.MOV_ID= c.MOV_ID
GROUP BY m.MOV_ID, m.MOV_TITLE
HAVING m.MOV_TITLE = movie_title;

IF act_count= 0 THEN
    RETURN 'No actor Found';
ELSIF act_count= 1 THEN
    RETURN 'Solo';
ELSE
    RETURN 'Ensemble';
END IF;

END ;
/

DECLARE
    title VARCHAR2(100);
BEGIN
    title:= '&title';
    DBMS_OUTPUT.PUT_LINE(movie_Status(title));
END;
/
```

```
--c--

-- Procedure for Oscar Nominees

CREATE OR REPLACE PROCEDURE Find_Oscar_Nominees
IS
    nominee_count NUMBER := 1;
BEGIN
    FOR nominee_names IN (
        SELECT DISTINCT d.DIR_FIRSTNAME || ' ' || d.DIR_LASTNAME AS NOMINEE_NAME
        FROM DIRECTOR d
        LEFT JOIN DIRECTION dir ON d.DIR_ID = dir.DIR_ID
        WHERE dir.MOV_ID IN (
            SELECT m.MOV_ID
            FROM MOVIE m
            LEFT JOIN RATING r ON m.MOV_ID = r.MOV_ID
            GROUP BY m.MOV_ID
            HAVING COUNT(r.REV_ID) > 10 AND AVG(r.REV_STARS) >= 7
        )
    )
    LOOP
        DBMS_OUTPUT.PUT_LINE('Nominee ' || nominee_count || ': ' ||
nominee_names.NOMINEE_NAME);
        nominee_count := nominee_count + 1;
    END LOOP;
END Find_Oscar_Nominees;
/


-- Call it from an anonymous block

BEGIN
    Find_Oscar_Nominees;
END;
/
```

```
--d--
CREATE OR REPLACE FUNCTION movie_Category (movie_title VARCHAR2)
RETURN VARCHAR2
IS
    release_year NUMBER;
    avg_rating NUMBER;
BEGIN
    SELECT m.mov_year, NVL(AVG(r.rev_stars), 0)
    INTO release_year, avg_rating
    FROM Movie m LEFT JOIN Rating r ON m.mov_id = r.mov_id
    WHERE m.MOV_TITLE = movie_title
    GROUP BY m.mov_id, m.mov_title, m.mov_year;

    IF (release_year >= 1950 AND release_year <= 1959 AND avg_rating > 6.5) THEN
        RETURN 'Fantastic Fifties';
    ELSIF (release_year >= 1960 AND release_year <= 1969 AND avg_rating > 6.7)
THEN
        RETURN 'Sweet Sixties';
    ELSIF (release_year >= 1970 AND release_year <= 1979 AND avg_rating > 6.9)
THEN
        RETURN 'Super Seventies';
    ELSIF (release_year >= 1980 AND release_year <= 1989 AND avg_rating > 7.1)
THEN
        RETURN 'Ecstatic Eighties';
    ELSIF (release_year >= 1990 AND release_year <= 1999 AND avg_rating > 7.3)
THEN
        RETURN 'Neat Nineties';
    ELSE
        RETURN 'Garbage';
    END IF;
END;
/

DECLARE
    title VARCHAR2(100);
BEGIN
    title:= '&title';
    DBMS_OUTPUT.PUT_LINE(movie_Category(title));
END;
/
```

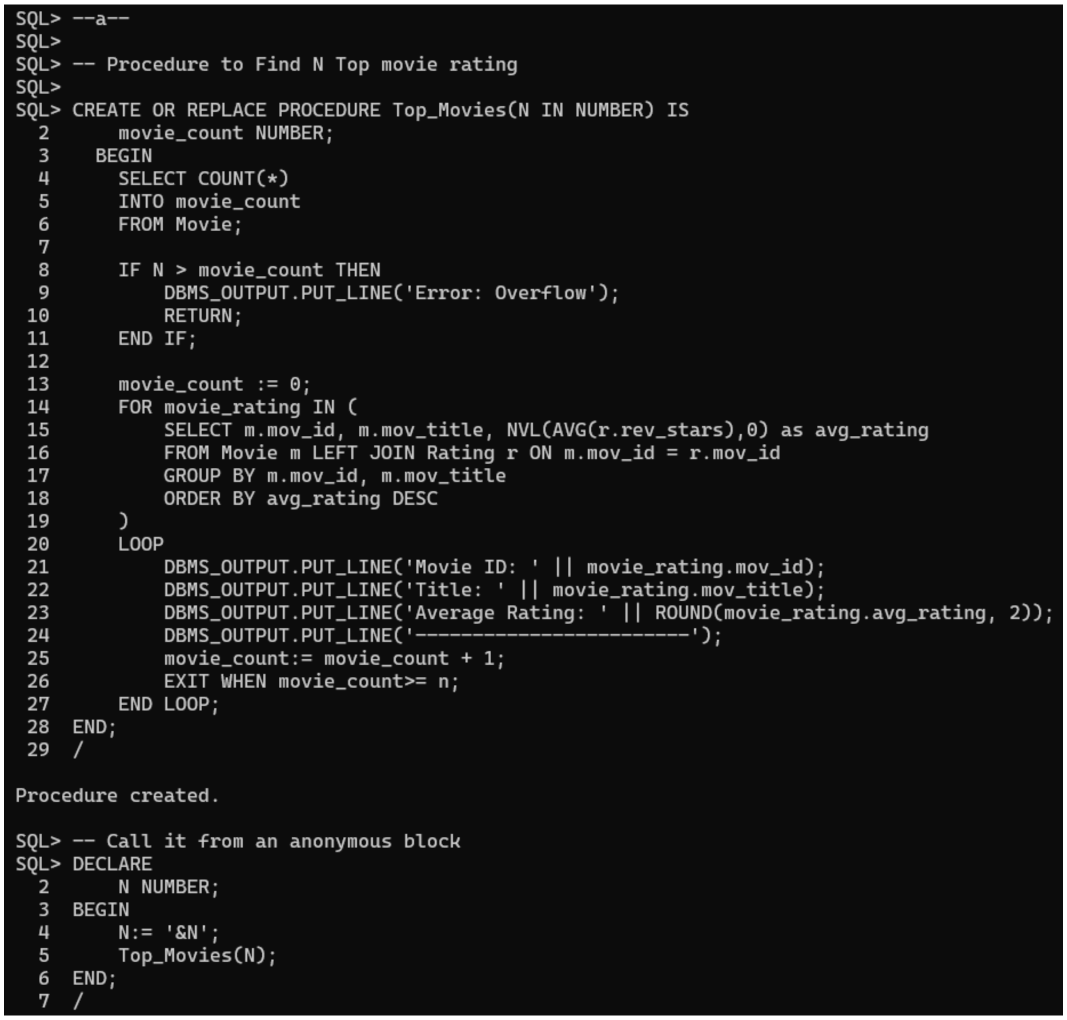
2.2 Analysis and Explanation

Here I had to run advance queries to make procedures and Functions for PL/SQL.

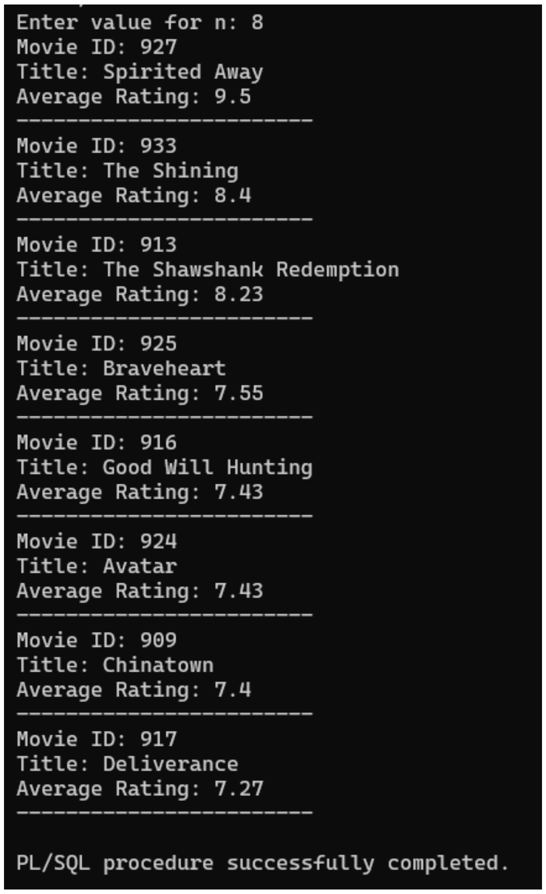
2.3 Difficulties

The major difficulty I faced is I got compilation error so many times and it was so hard to identify the problem as I had to give input in a whole block which contains several lines.

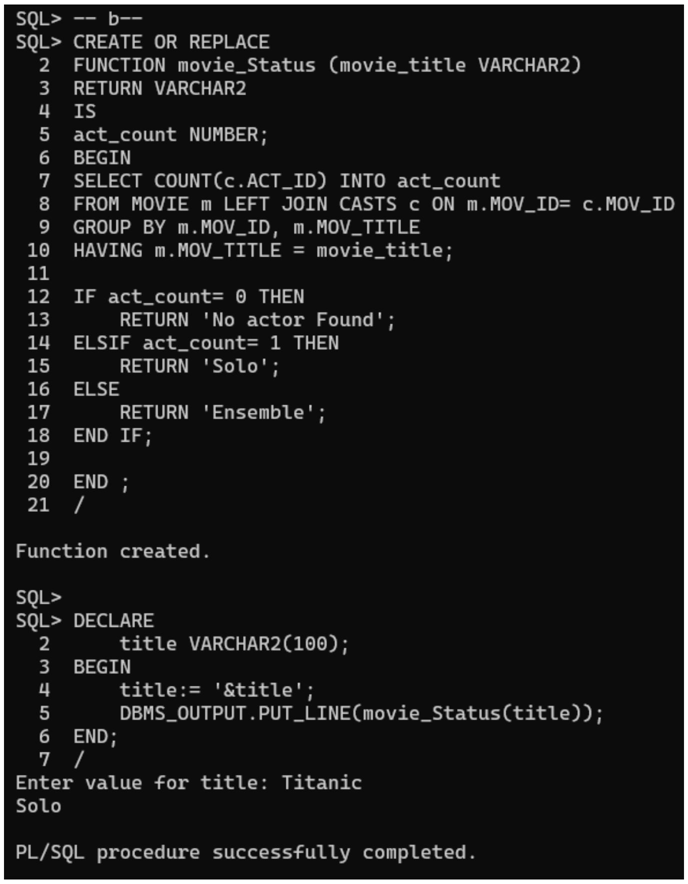
2.4 Output



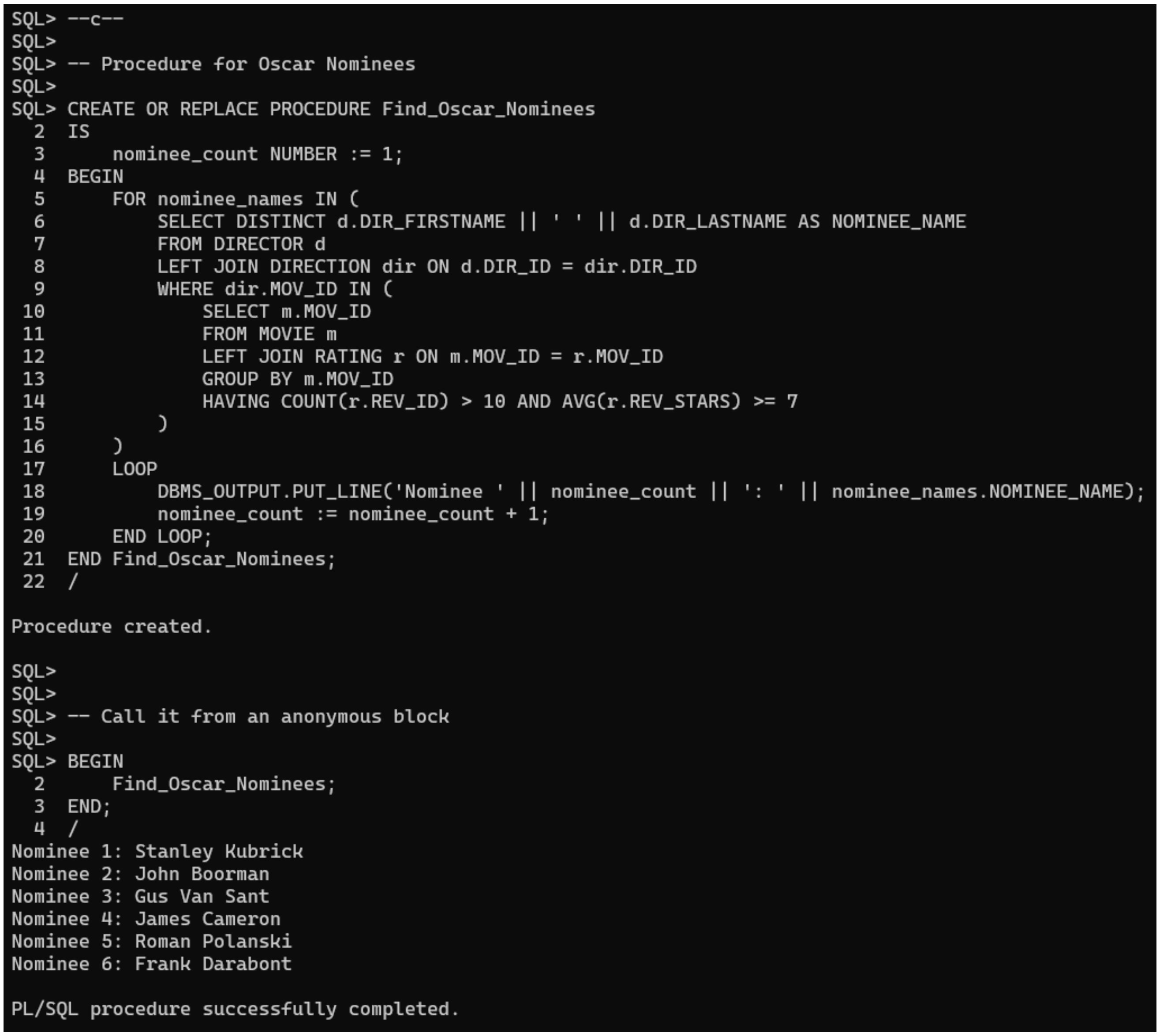
```
SQL> --a--
SQL>
SQL> -- Procedure to Find N Top movie rating
SQL>
SQL> CREATE OR REPLACE PROCEDURE Top_Movies(N IN NUMBER) IS
        movie_count NUMBER;
      BEGIN
        SELECT COUNT(*)
        INTO movie_count
        FROM Movie;

        IF N > movie_count THEN
            DBMS_OUTPUT.PUT_LINE('Error: Overflow');
            RETURN;
        END IF;

        movie_count := 0;
        FOR movie_rating IN (
            SELECT m.mov_id, m.mov_title, NVL(AVG(r.rev_stars),0) as avg_rating
            FROM Movie m LEFT JOIN Rating r ON m.mov_id = r.mov_id
            GROUP BY m.mov_id, m.mov_title
            ORDER BY avg_rating DESC
        )
        LOOP
            DBMS_OUTPUT.PUT_LINE('Movie ID: ' || movie_rating.mov_id);
            DBMS_OUTPUT.PUT_LINE('Title: ' || movie_rating.mov_title);
            DBMS_OUTPUT.PUT_LINE('Average Rating: ' || ROUND(movie_rating.avg_rating, 2));
            DBMS_OUTPUT.PUT_LINE('-------------------------');
            movie_count:= movie_count + 1;
            EXIT WHEN movie_count>= n;
        END LOOP;
    END;
    /

Procedure created.

SQL> -- Call it from an anonymous block
SQL> DECLARE
        N NUMBER;
    BEGIN
        N:= '&N';
        Top_Movies(N);
    END;
    /
```



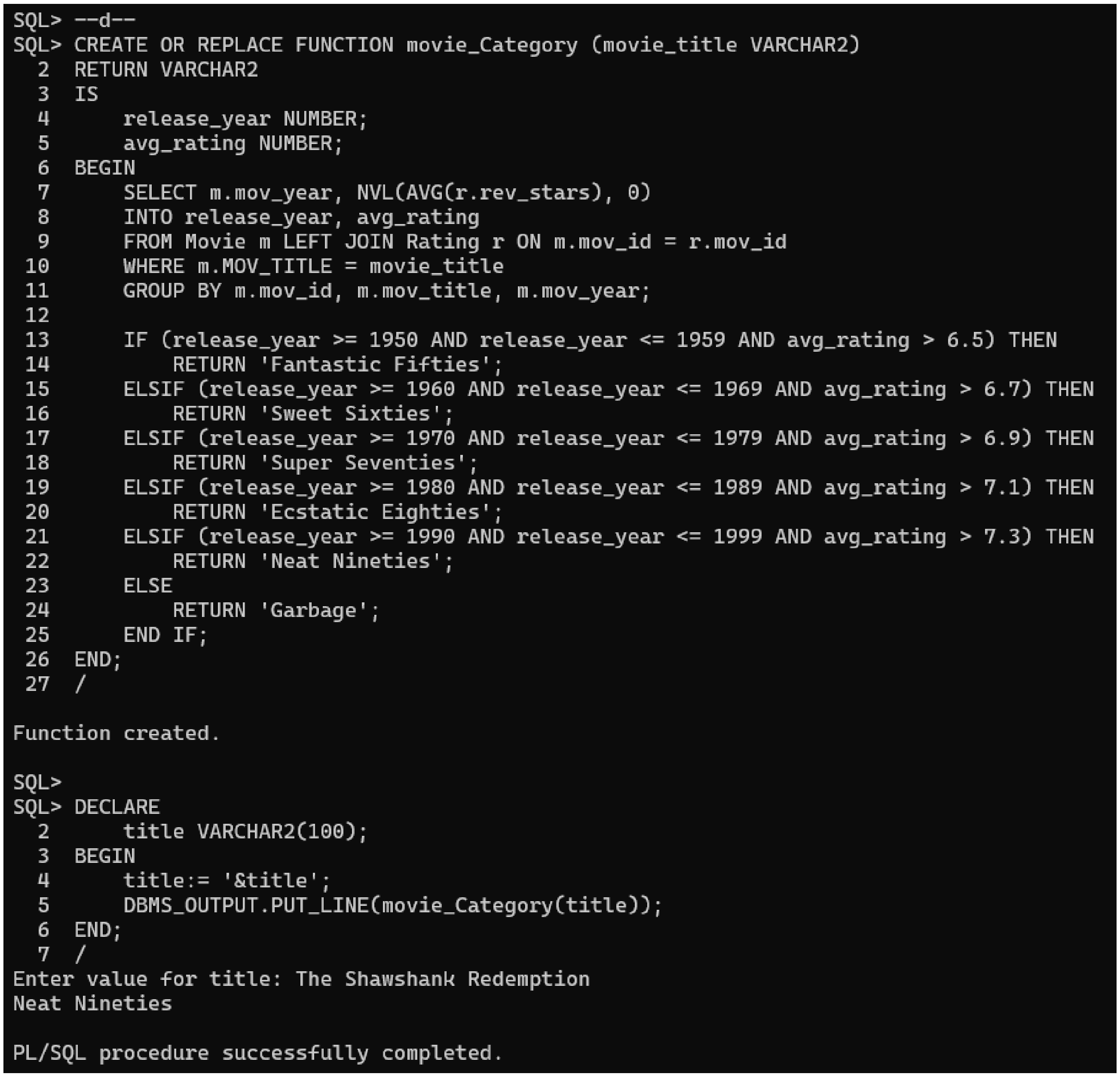
```
Enter value for n: 8
Movie ID: 927
Title: Spirited Away
Average Rating: 9.5
-------------------------
Movie ID: 933
Title: The Shining
Average Rating: 8.4
-------------------------
Movie ID: 913
Title: The Shawshank Redemption
Average Rating: 8.23
-------------------------
Movie ID: 925
Title: Braveheart
Average Rating: 7.55
-------------------------
Movie ID: 916
Title: Good Will Hunting
Average Rating: 7.43
-------------------------
Movie ID: 924
Title: Avatar
Average Rating: 7.43
-------------------------
Movie ID: 909
Title: Chinatown
Average Rating: 7.4
-------------------------
Movie ID: 917
Title: Deliverance
Average Rating: 7.27
-------------------------

PL/SQL procedure successfully completed.
```



```
SQL> -- b--
SQL> CREATE OR REPLACE
     FUNCTION movie_Status (movie_title VARCHAR2)
     RETURN VARCHAR2
     IS
     act_count NUMBER;
     BEGIN
     SELECT COUNT(c.ACT_ID) INTO act_count
     FROM MOVIE m LEFT JOIN CASTS c ON m.MOV_ID= c.MOV_ID
     GROUP BY m.MOV_ID, m.MOV_TITLE
     HAVING m.MOV_TITLE = movie_title;

     IF act_count= 0 THEN
         RETURN 'No actor Found';
     ELSIF act_count= 1 THEN
         RETURN 'Solo';
     ELSE
         RETURN 'Ensemble';
     END IF;

     END ;
     /

Function created.

SQL>
SQL> DECLARE
         title VARCHAR2(100);
     BEGIN
         title:= '&title';
         DBMS_OUTPUT.PUT_LINE(movie_Status(title));
     END;
     /
Enter value for title: Titanic
Solo

PL/SQL procedure successfully completed.
```



```
SQL> --c--
SQL>
SQL> -- Procedure for Oscar Nominees
SQL>
SQL> CREATE OR REPLACE PROCEDURE Find_Oscar_Nominees
     IS
         nominee_count NUMBER := 1;
     BEGIN
         FOR nominee_names IN (
             SELECT DISTINCT d.DIR_FIRSTNAME || ' ' || d.DIR_LASTNAME AS NOMINEE_NAME
             FROM DIRECTOR d
             LEFT JOIN DIRECTION dir ON d.DIR_ID = dir.DIR_ID
             WHERE dir.MOV_ID IN (
                 SELECT m.MOV_ID
                 FROM MOVIE m
                 LEFT JOIN RATING r ON m.MOV_ID = r.MOV_ID
                 GROUP BY m.MOV_ID
                 HAVING COUNT(r.REV_ID) > 10 AND AVG(r.REV_STARS) >= 7
             )
         )
         LOOP
             DBMS_OUTPUT.PUT_LINE('Nominee ' || nominee_count || ': ' || nominee_names.NOMINEE_NAME);
             nominee_count := nominee_count + 1;
         END LOOP;
     END Find_Oscar_Nominees;
     /

Procedure created.

SQL>
SQL>
SQL> -- Call it from an anonymous block
SQL>
SQL> BEGIN
         Find_Oscar_Nominees;
     END;
     /
Nominee 1: Stanley Kubrick
Nominee 2: John Boorman
Nominee 3: Gus Van Sant
Nominee 4: James Cameron
Nominee 5: Roman Polanski
Nominee 6: Frank Darabont

PL/SQL procedure successfully completed.
```



```
SQL> --d--
SQL> CREATE OR REPLACE FUNCTION movie_Category (movie_title VARCHAR2)
     RETURN VARCHAR2
     IS
         release_year NUMBER;
         avg_rating NUMBER;
     BEGIN
         SELECT m.mov_year, NVL(AVG(r.rev_stars), 0)
         INTO release_year, avg_rating
         FROM Movie m LEFT JOIN Rating r ON m.mov_id = r.mov_id
         WHERE m.MOV_TITLE = movie_title
         GROUP BY m.mov_id, m.mov_title, m.mov_year;

         IF (release_year >= 1950 AND release_year <= 1959 AND avg_rating > 6.5) THEN
             RETURN 'Fantastic Fifties';
         ELSIF (release_year >= 1960 AND release_year <= 1969 AND avg_rating > 6.7) THEN
             RETURN 'Sweet Sixties';
         ELSIF (release_year >= 1970 AND release_year <= 1979 AND avg_rating > 6.9) THEN
             RETURN 'Super Seventies';
         ELSIF (release_year >= 1980 AND release_year <= 1989 AND avg_rating > 7.1) THEN
             RETURN 'Ecstatic Eighties';
         ELSIF (release_year >= 1990 AND release_year <= 1999 AND avg_rating > 7.3) THEN
             RETURN 'Neat Nineties';
         ELSE
             RETURN 'Garbage';
         END IF;
     END;
     /

Function created.

SQL>
SQL> DECLARE
         title VARCHAR2(100);
     BEGIN
         title:= '&title';
         DBMS_OUTPUT.PUT_LINE(movie_Category(title));
     END;
     /
Enter value for title: The Shawshank Redemption
Neat Nineties

PL/SQL procedure successfully completed.
```