|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | |
|  | 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. **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 ;

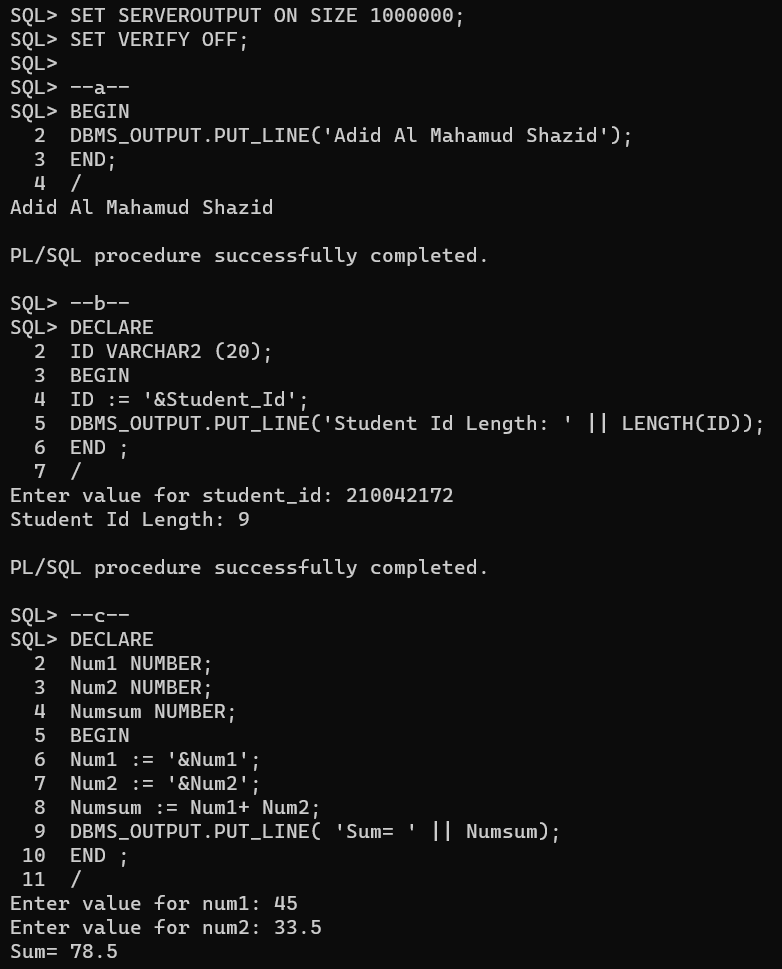
/

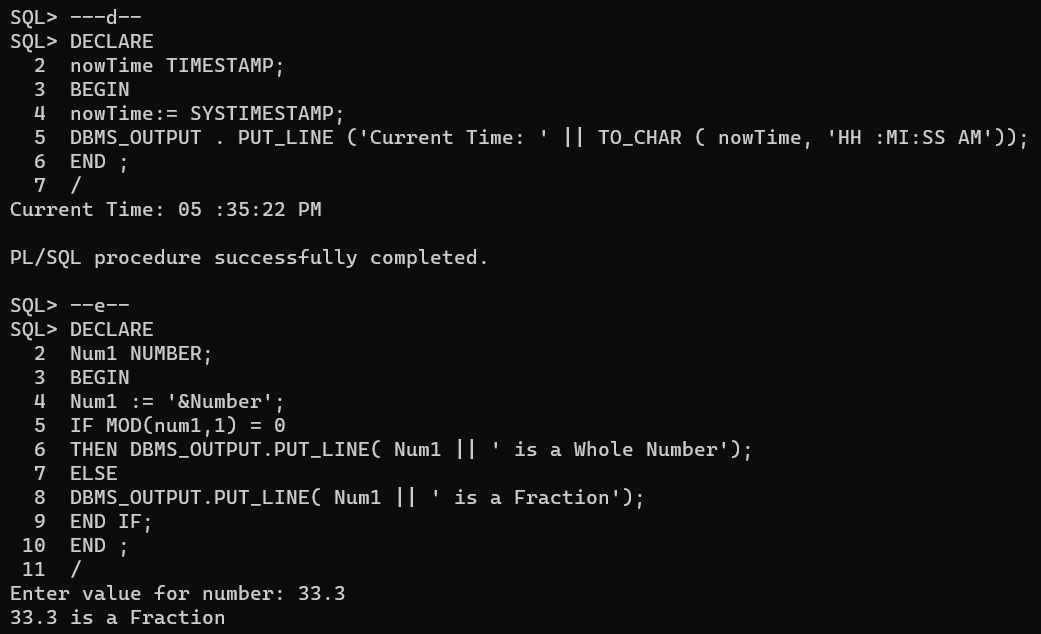
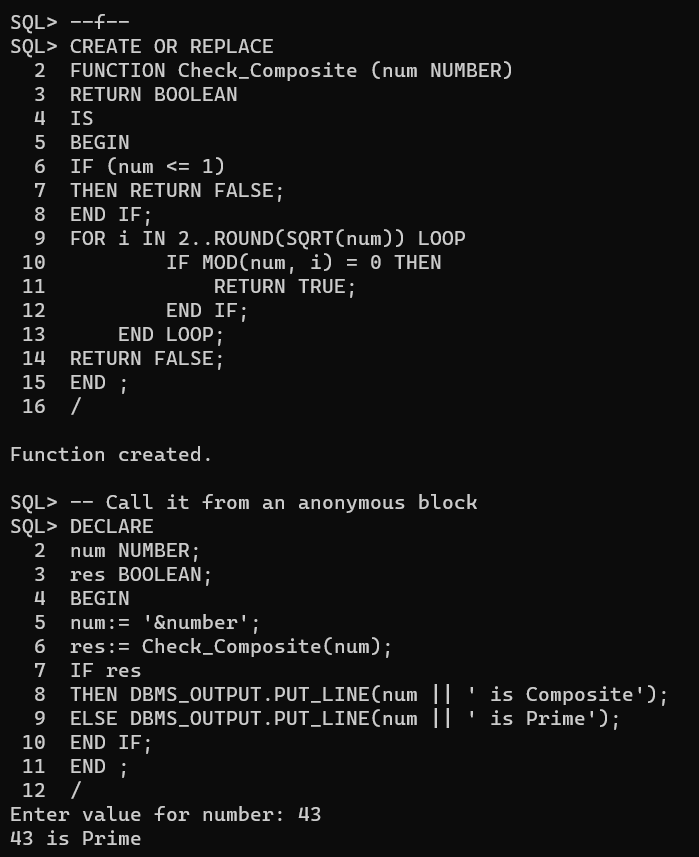
* 1. **Analysis and Explanation**

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

* 1. **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. **Output**

****

**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.

* 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;

/

-- 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;

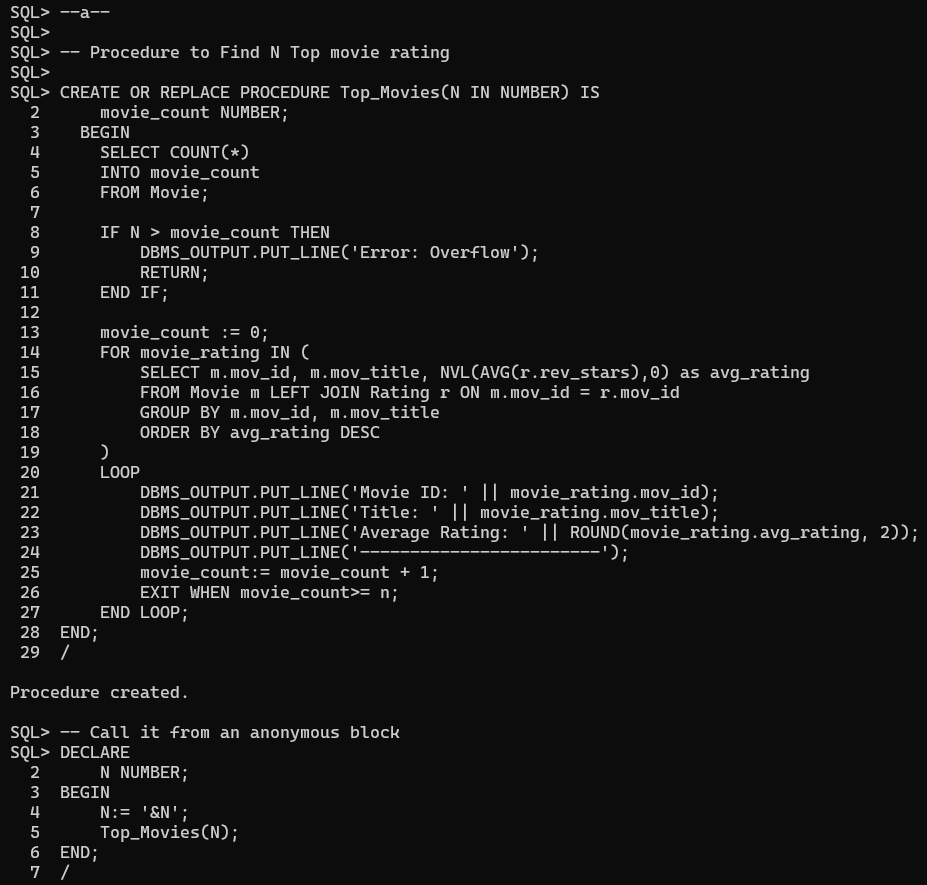
/

* 1. **Analysis and Explanation**

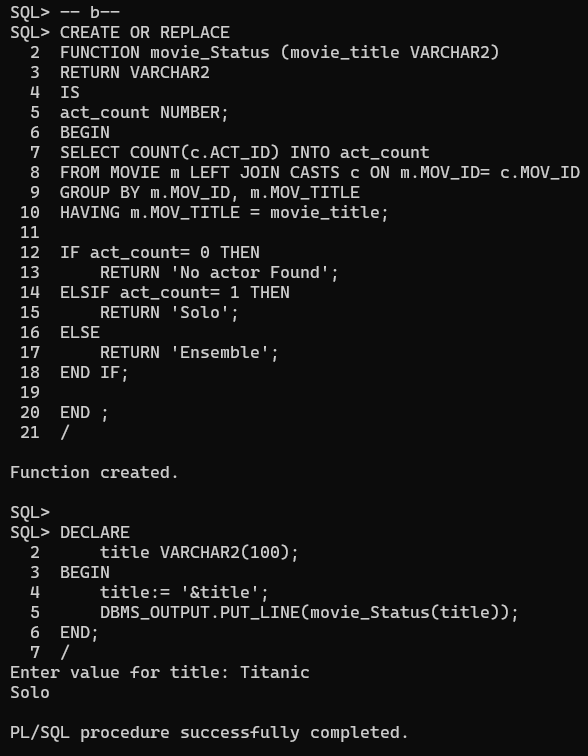
Here I had to run advance queries to make procedures and Functions for PL/SQL.

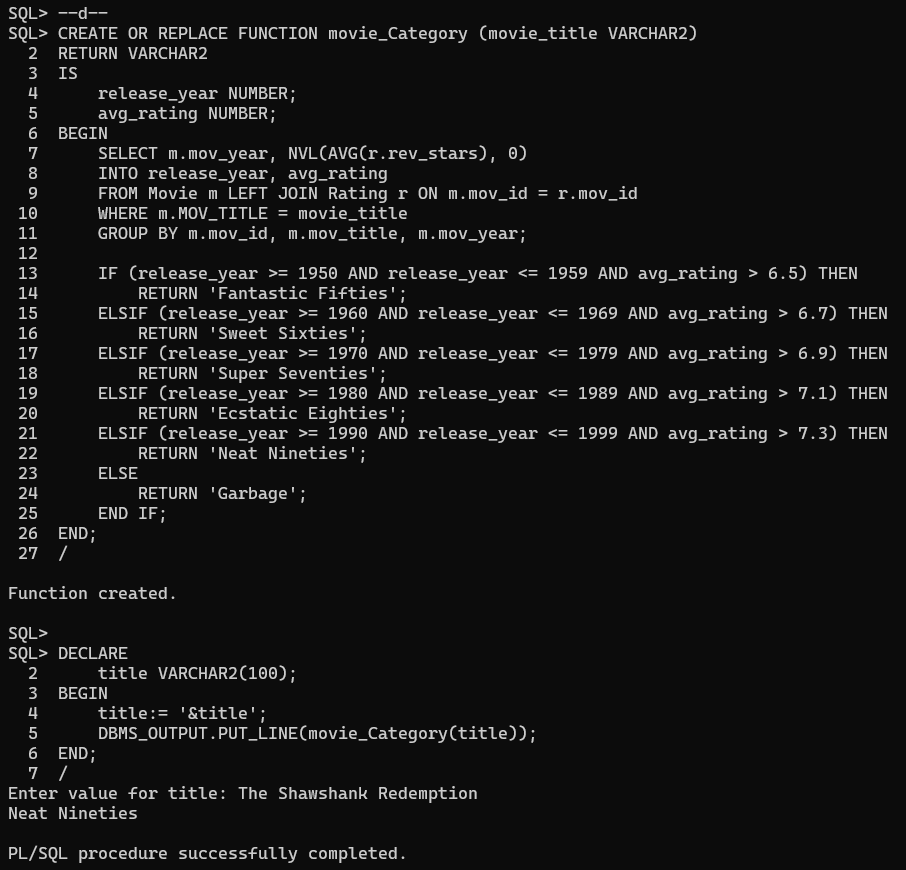
* 1. **Difficulties**

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.

* 1. **Output**

****

****

****