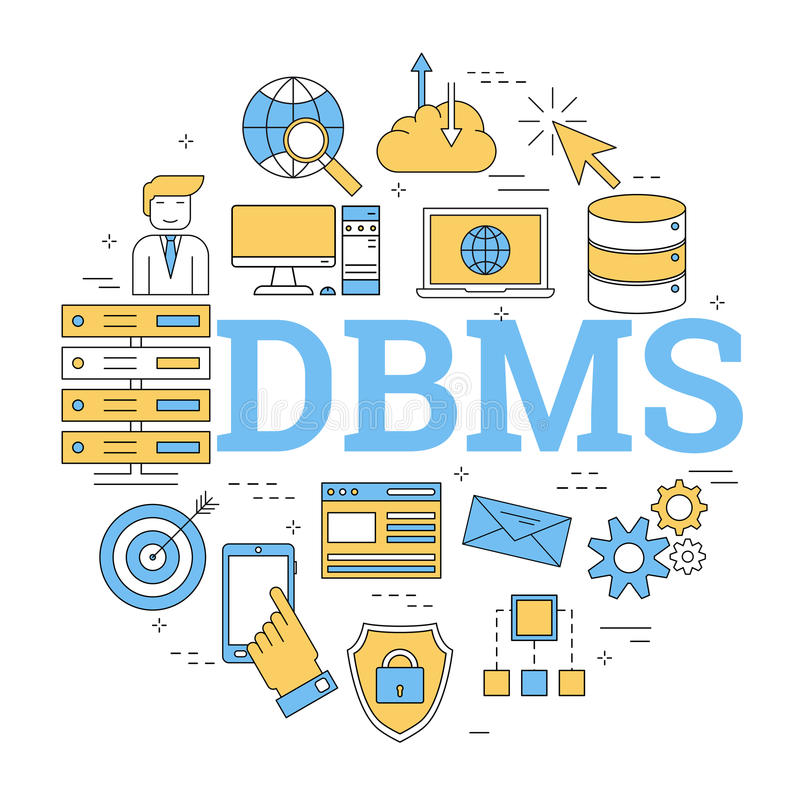
 Islamic University of Technology (IUT)

CSE 4308: Database Management Systems Lab

Lab Report # 9

**Submitted to:**

Md. Bakhtiar Hasan,

Asst. Professor, CSE.

Zannatun Naim Srsity,

Lecturer, CSE.

**Submitted by:**

M M Nazmul Hossain  
ID 200042118  
CSE (SWE)

**Submission Date:**

7.11.2022

**Introduction**

The ninth Database Management Systems Lab was about Advanced SQL. PL/SQL or Procedural Language extension to Structured Query Language, can simplify application development and optimize its execution. It removes the dependency on OJDBC, rather is tightly integrated with SQL itself.  
  
**Method**

At first, the provided movie.sql file is imported into the Database to create the tables and insert the values in the tables, which will be required for this lab. Before completing any tasks, the server output size has to be set which can be done by writing:

SET SERVEROUTPUT ON SIZE 1000000

Analysis of the problem of Task 1 A

This warm-up task was to simply print a name.

The Code

-- WARMUP TASK A

BEGIN

    DBMS\_OUTPUT.PUT\_LINE('My Name is M M Nazmul Hossain');

END;

/

Explanation of Solution

* Simply output a static string that includes a name.

Analysis of the problem of Task 1 B

The warm-up task B was to take student ID as input and then print its length.

The Code

-- WARMUP TASK B

DECLARE

    USER\_ID VARCHAR2(100);

BEGIN

    USER\_ID := '&userid';

    DBMS\_OUTPUT.PUT\_LINE('The length of the user ID is '

        ||LENGTH(USER\_ID));

END;

/

Explanation of Solution

* A string variable called USER\_ID is declared.
* Then the user id as input is taken from the user.
* The Length(String) function is used to print out the length of the USER\_ID string.

Analysis of the problem of Task 1 C

The warm-up task C was to take two numbers as input and output their product.

The Code

-- WARMUP TASK C

DECLARE

    VARA    NUMBER;

    VARB    NUMBER;

    PRODUCT NUMBER;

BEGIN

    VARA :='&VarA';

    VARB :='&VarB';

    PRODUCT := VARA\*VARB;

    DBMS\_OUTPUT.PUT\_LINE('The Product of the two input variables are : '

        ||PRODUCT);

END;

/

Explanation of Solution

* Two number variables are declared. Another variable, product, is declared to store their product.
* Input is taken in both of the numbers.
* Their product is calculated and the value is stored in the third variable.
* The product variable is displayed as the output.

Analysis of the problem of Task 1 D

The warm-up task D was to print the current system time but in 12-hour format.

The Code

-- WAMRUP TASK D

DECLARE

    D    DATE;

BEGIN

    D := SYSDATE();

    DBMS\_OUTPUT.PUT\_LINE(TO\_CHAR(D, 'DD-MON-YYYY HH12:MI:SS'));

END;

/

Explanation of Solution

* A date variable is declared.
* Sysdate() is assigned to that variable.
* The date variable is displayed as output in the 12-hour format via the To\_Char() function.

Analysis of the problem of Task 1 E

The warm-up task e was to take a number as input and determine whether or not it is a whole number.

The Code

-- WARMUP TASK E

-- WITHOUT CASE

DECLARE

    VAR NUMBER(10,2);

BEGIN

    VAR := '&number';

    IF VAR = ROUND(VAR) THEN

        DBMS\_OUTPUT.PUT\_LINE('It is a Whole Number');

    ELSE

        DBMS\_OUTPUT.PUT\_LINE ('It is a Fraction.');

    END IF;

END;

/

-- WITH CASE

DECLARE

    VAR NUMBER(10,2);

BEGIN

    VAR := '&NUMBER';

    CASE TRUE

        WHEN VAR = ROUND(VAR) THEN

            DBMS\_OUTPUT.PUT\_LINE ('It is a Whole Number.');

        ELSE

            DBMS\_OUTPUT.PUT\_LINE ('It is a Fraction.');

    END CASE;

END;

/

Explanation of Solution

* A number variable is declared, that has a precision of up to 2 decimal points to allow for fraction values.
* Input is taken from the user.
* Round() function is used to check if the value of the variable changes. If it changes, then the variable was a fraction, not a whole number.
* The above-stated logic is implemented using both IF conditional and Case conditional.

Analysis of the problem of Task 1 F

The warm-up task f was to take a number as an input. Then send it to a procedure that will return whether the number is a composite number or not.

The Code

-- WARMUP TASK F

CREATE OR REPLACE

PROCEDURE FIND\_COMPOSITE( NUM IN NUMBER , VERDICT OUT VARCHAR2 )

AS

BEGIN

    VERDICT := 'Number is not Composite';

    FOR i IN 2 .. (NUM/2)

        LOOP

            IF MOD(NUM, i) = 0 THEN

                VERDICT := 'Number is Composite';

                EXIT;

            END IF;

        END LOOP;

END;

/

DECLARE

    NUM NUMBER(5);

    VERIFY VARCHAR2(30);

BEGIN

    NUM := '&number';

    FIND\_COMPOSITE(NUM,VERIFY);

    DBMS\_OUTPUT.PUT\_LINE (VERIFY);

END;

/

Explanation of Solution

* A procedure is created that will take in a number variable and give out a string variable, verdict. Using a for loop, a modulus operation is run until half the provided number’s value. If it returns 0 at any point, then the number would be a composite number. Else the number would be a prime number, ie: not a composite number. This closes the procedure block.
* In another block, a number variable and a string variable is declared.
* Input is taken into the number variable.
* The number variable and the string variable are sent through the procedure and then output the string variable which should print the result found from the procedure.
* The result is displayed using the output function.

Analysis of the problem of Task 2 A

Task 2 A was to write a procedure, which will take in a Number variable. Then check if the number is greater than the number of movies in the database. If it is greater, then it will output an error message. If it is not, then it will print the details of the First N top-rated movies.

The Code

-- TASK 2 A

CREATE OR REPLACE

PROCEDURE FIND\_MOVIES(NUM IN NUMBER)

AS

ROWNUMS NUMBER(10);

CURSOR MOV\_INFO IS

    SELECT \*

    FROM (SELECT MOV\_ID,MOV\_TITLE,MAX(MOV\_YEAR) AS MOV\_YEAR,MAX(MOV\_LANGUAGE) AS MOV\_LANGUAGE,MAX(MOV\_RELEASEDATE) AS MOV\_RELEASEDATE,MAX(MOV\_COUNTRY) AS MOV\_COUNTRY, AVG(REV\_STARS) AS AVG\_RATING

    FROM MOVIE NATURAL JOIN RATING

    GROUP BY MOV\_ID,MOV\_TITLE

    ORDER BY AVG\_RATING DESC)

    WHERE ROWNUM<=NUM;

BEGIN

    SELECT MAX(ROWNUM) INTO ROWNUMS

    FROM (SELECT MOV\_ID,MOV\_TITLE,MAX(MOV\_YEAR) AS MOV\_YEAR,MAX(MOV\_LANGUAGE) AS MOV\_LANGUAGE,MAX(MOV\_RELEASEDATE) AS MOV\_RELEASEDATE,MAX(MOV\_COUNTRY) AS MOV\_COUNTRY, AVG(REV\_STARS) AS AVG\_RATING FROM MOVIE NATURAL JOIN RATING GROUP BY MOV\_ID,MOV\_TITLE ORDER BY AVG\_RATING DESC);

    CASE TRUE

    WHEN (NUM>ROWNUMS) THEN

        DBMS\_OUTPUT.PUT\_LINE ('Input exceeds number of entries');

    ELSE

        DBMS\_OUTPUT.PUT\_LINE ('MOV\_TITLE' || chr(9) || 'MOV\_YEAR' || chr(9) || 'MOV\_LANGUAGE' || chr(9)  || 'MOV\_RELEASEDATE' || chr(9)  || 'MOV\_COUNTRY');

        FOR i IN MOV\_INFO LOOP

        DBMS\_OUTPUT.PUT\_LINE (i.MOV\_TITLE || chr(9) || i.MOV\_YEAR || chr(9) || i.MOV\_LANGUAGE || chr(9)  || i.MOV\_RELEASEDATE || chr(9)  || i.MOV\_COUNTRY);

        END LOOP;

    END CASE;

END;

/

DECLARE

    NUM NUMBER(10);

BEGIN

    NUM := '&number';

    FIND\_MOVIES(NUM);

END;

/

Explanation of Solution

* A procedure, Find\_Movies is created which will take in a Number variable. The procedure declares a Rownums number variable and a Cursor, Mov\_Info which stores all the movie information sorted based on their average rating up to the first provided number variable which is determined via an SQL query.
* At the beginning of the procedure, the number of rows of movie information is assigned to the rownums variable.
* Then a conditional statement is used to check if the provided number is larger than the rownums variable. If it is, then it will output an error. If it isn’t, then it will use a for loop to traverse the Mov\_Info Cursor and output the movie details. Close the loop and this closes the procedure block.
* Finally, in another block, a number variable is declared.
* User input is taken into that variable and the Find\_Movies procedure is called.

Analysis of the problem of Task 2 B

The

The Code

-- TASK 2 B

CREATE OR REPLACE

PROCEDURE INSERT\_STATUS(MOVIE\_TITLE IN VARCHAR2)

AS

CURSOR MOV\_INFO IS

    SELECT MOV\_ID, MOV\_TITLE, COUNT(ACT\_ID) AS NO\_OF\_ACTORS

    FROM MOVIE NATURAL JOIN CASTS

    WHERE MOV\_TITLE = MOVIE\_TITLE

    GROUP BY MOV\_ID, MOV\_TITLE;

BEGIN

    FOR i in MOV\_INFO LOOP

        CASE TRUE

        WHEN (i.NO\_OF\_ACTORS<=1) THEN

            DBMS\_OUTPUT.PUT\_LINE('SOLO');

            DBMS\_OUTPUT.PUT\_LINE(i.NO\_OF\_ACTORS);

        ELSE

            DBMS\_OUTPUT.PUT\_LINE('ENSEMBLE');

            DBMS\_OUTPUT.PUT\_LINE(i.NO\_OF\_ACTORS);

        END CASE;

    END LOOP;

END;

/

DECLARE

    MOVIE\_TITLE VARCHAR2(20);

BEGIN

    MOVIE\_TITLE := '&title';

    INSERT\_STATUS(MOVIE\_TITLE);

END;

/

Explanation of Solution

* Asd

Analysis of the problem of Task 2 C

The

The Code

-- TASK 2 C

CREATE OR REPLACE

PROCEDURE GIVE\_OSCAR

AS

CURSOR MOV\_INFO IS

    SELECT DIR\_ID, MAX(DIR\_FIRSTNAME) AS DIR\_FIRSTNAME, MAX(DIR\_LASTNAME) AS DIR\_LASTNAME

    FROM (SELECT MOV\_ID,MOV\_TITLE, AVG(REV\_STARS) AS AVG\_RATING, COUNT(REV\_STARS) AS NO\_OF\_REVIEWS

    FROM MOVIE NATURAL JOIN RATING

    GROUP BY MOV\_ID,MOV\_TITLE

    ORDER BY AVG\_RATING DESC

    ) MOV NATURAL JOIN DIRECTOR

    WHERE AVG\_RATING >=7 AND NO\_OF\_REVIEWS >=10

    GROUP BY DIR\_ID

    ORDER BY DIR\_ID;

BEGIN

    DBMS\_OUTPUT.PUT\_LINE('DIR\_ID'||chr(9)||'DIR\_FIRSTNAME'||chr(9)||'DIR\_LASTNAME');

    FOR i in MOV\_INFO LOOP

        DBMS\_OUTPUT.PUT\_LINE(i.DIR\_ID || chr(9)|| i.DIR\_FIRSTNAME || chr(9) || i.DIR\_LASTNAME);

    END LOOP;

END;

/

BEGIN

GIVE\_OSCAR();

END;

/

Explanation of Solution

* asd

Analysis of the problem of Task 2 D

The

The Code

-- TASK 2 D

CREATE OR REPLACE

PROCEDURE MOVIE\_CATEGORY(MOVIE\_TITLE IN VARCHAR2)

AS

YEAR VARCHAR2(20);

CURSOR MOV\_INFO IS

    SELECT MOV\_ID,MOV\_TITLE, MAX(MOV\_RELEASEDATE) AS MOV\_RELEASEDATE, AVG(REV\_STARS) AS AVG\_RATING

    FROM MOVIE NATURAL JOIN RATING

    WHERE MOV\_TITLE = MOVIE\_TITLE

    GROUP BY MOV\_ID,MOV\_TITLE

    ORDER BY AVG\_RATING DESC;

BEGIN

    FOR i IN MOV\_INFO LOOP

        YEAR := TO\_CHAR(i.MOV\_RELEASEDATE, 'YYYY');

        IF i.AVG\_RATING>6.5 AND YEAR>=1950 AND YEAR<=1959 THEN

            DBMS\_OUTPUT.PUT\_LINE('The Movie is From the Fantastic Fifties');

        ELSIF i.AVG\_RATING>6.7 AND YEAR>=1960 AND YEAR<=1969 THEN

            DBMS\_OUTPUT.PUT\_LINE('The Movie is From the Sweet Sixties');

        ELSIF i.AVG\_RATING>6.9 AND YEAR>=1970 AND YEAR<=1979 THEN

            DBMS\_OUTPUT.PUT\_LINE('The Movie is From the Super Seventies');

        ELSIF i.AVG\_RATING>7.1 AND YEAR>=1980 AND YEAR<=1989 THEN

            DBMS\_OUTPUT.PUT\_LINE('The Movie is From the Ecstatic Eighties');

        ELSIF i.AVG\_RATING>7.3 AND YEAR>=1990 AND YEAR<=1999 THEN

            DBMS\_OUTPUT.PUT\_LINE('The Movie is From the Neat Nineties');

        ELSE

            DBMS\_OUTPUT.PUT\_LINE('The Movie is Garbage');

        END IF;

    END LOOP;

END;

/

DECLARE

    MOVIE\_TITLE VARCHAR2(20);

BEGIN

MOVIE\_TITLE := '&MOVIE\_TITLE';

MOVIE\_CATEGORY(MOVIE\_TITLE);

END;

/

Explanation of Solution

* asd

**Problems**The

Spacing

If

Cursor

Syntax

**Findings**

Using

**Conclusion**

Overall