Course Number and Name: CSE 4308 Database Management Systems Lab	
Student Name: Nafisa Maliyat	**Student ID:** 200042133
Report Submission Date: 03 November, 2022	**Name of Lab Instructor:** Md. Bakhtiar Hasan, Lecturer, CSE Zannatun Naim Sristy, Lecturer, CSE

Lab 8: PL/SQL

Overview:

This lab provided us a manual for the basics of PL/SQL. Using this knowledge, different queries had to be performed on the table banking.sql given with the lab task.

On the next pages, I have mentioned the following :

- the problem statement
- analysis of the problem,
- SQL written to solve the problem,
- problems faced (if any) during solution of the tasks.

Task 1(a, b, c):

Problem Statement:

(a) Print your student ID.
(b) Take your name as input and print its length.
(c) Take two numbers as input and print their sum.
(d) Print the current system time in 24-hour format.

Analysis of the problem:

For 1(a), serveroutput variable is enabled at the beginning and DBMS_OUTPUT.PUT_LINE is used to print the ID.
For 1(b), a varchar2 variable is declared beforehand and input is taken. DBMS_OUTPUT.PUT_LINE is then used to print the variable containing the name.
For 1(c), two numbers are declared as variables. After taking user input of the numbers, they are added and converted to char before printing.
For 1(d), a date variable is assigned the value of current system date. When printing, it is converted to the desired form by 'to char' where the format is specified.

Any problems faced and how it was solved:

There were slight problems with the syntax since this is an entirely new language for example, how to convert sysdate to the required format.

Code:

```
--1(a)
SET SERVEROUTPUT ON SIZE 1000000
BEGIN
DBMS_OUTPUT.PUT_LINE('200042133');
END ;
/
```

```
--1(b)
DECLARE
NAME VARCHAR2 (20);
BEGIN
NAME := '& name ';
DBMS_OUTPUT.PUT_LINE('Length of my name is ' || Length(NAME) );
END ;
/
```

```
--1(c)
DECLARE
NUMBER1 NUMBER;
NUMBER2 NUMBER;
BEGIN
NUMBER1 := '& number1 ';
NUMBER2 := '& number2 ';
DBMS_OUTPUT.PUT_LINE('Sum of the numbers is ' || TO_CHAR(NUMBER1 +
NUMBER2) );
END ;
/
```

```
--1(d)
DECLARE
D DATE := SYSDATE ;
BEGIN
DBMS_OUTPUT . PUT_LINE (TO_CHAR(D, 'HH24:MI:SS'));
END ;
/
```

Results:

```
(a)
200042133
PL/SQL procedure successfully completed.

(b)
Length of my name is 15
PL/SQL procedure successfully completed.

(c)
Enter value for number1: 3
old   5: NUMBER1 := '& number1 ';
new   5: NUMBER1 := '3 ';
Enter value for number2: 4
old   6: NUMBER2 := '& number2 ';
new   6: NUMBER2 := '4 ';
Sum of the numbers is 7
PL/SQL procedure successfully completed.

(d)
00:48:21
PL/SQL procedure successfully completed.
```

Task 1(e):

Problem Statement:

Take a number as input and print whether it is odd or even (with and without CASE statement).

Analysis of the problem:

For solving the problem with CASE statement, CASE was used and in without CASE statement, if-else was used to check conditions. If the number is divisible by 2, it is even. Otherwise, it is even.

Any problems faced and how it was solved:

There were no problems faced since the query was straightforward and simple.

Code:

```
--1(e) WITHOUT CASE
DECLARE
X NUMBER ;
BEGIN
X := '& number';
X := MOD(X, 2);
IF (X = 0) THEN
DBMS_OUTPUT . PUT_LINE ( 'X is even');
ELSE
DBMS_OUTPUT . PUT_LINE ( 'X is odd');
END IF;
END ;
/

--1(e) WITH CASE
DECLARE
X NUMBER ;
BEGIN
X := '& x';
X := MOD(X, 2);
CASE X
WHEN 0 THEN
DBMS_OUTPUT . PUT_LINE ( 'X is even');
ELSE
DBMS_OUTPUT . PUT_LINE ( 'X is odd');
END CASE ;
END ;
```

```
/
```

Results:

```
(e)  WITH/WITHOUT CASE (same output)
Enter value for number: 5
old   4: X := '& number';
new   4: X := '5';
X is odd
PL/SQL procedure successfully completed.
```

Task 1(f):

Problem Statement:

Write a procedure that takes a number as argument and prints whether it is a prime number or not.

Analysis of the problem:

First, a variable called is_prime is taken as out parameter that is initially considered to be 1. A loop was used to check if the number y is is divisible by any number from 2 to y/2. If any point in loop, this turns out to be true, the is_prime's value is changed to 0 and loop is broken.

In the block calling the procedure, a prime number is taken as input and passed in the function along with is_prime variable as parameters. The value of is_prime is checked. If it is 1, the output will be 'prime', and if 0, output will be 'not prime'.

A more efficient way would have been to put the print lines inside the procedure so upon passing a number, condition checking and printing is done within the procedure.

Any problems faced and how it was solved:

There were problems when creating the procedure - a lot of compilation errors. The errors were difficult to understand and vague sometimes thus it was a bit of a challenge. The line number was used to find where the error was and internet resources were utilized to identify what went wrong.

Code:

```
CREATE OR REPLACE
PROCEDURE CHECK_PRIME (Y IN NUMBER, IS_PRIME OUT NUMBER)
AS
BEGIN
IS_PRIME:=1;
  for i in 2..Y/2
    loop
        if (mod(y, i) = 0) then
        IS_PRIME:=0;
        exit;
        end if;
    end loop;
END ;
/
```

```
DECLARE
IS_PRIME NUMBER ;
Y NUMBER;
BEGIN
Y := '& y';
CHECK_PRIME (y , IS_PRIME );
IF(IS_PRIME = 0) THEN
    DBMS_OUTPUT . PUT_LINE ( 'NOT PRIME' );
ELSE
    DBMS_OUTPUT . PUT_LINE ( 'IS PRIME' );
END IF;
END;
/
```

Results:

```
(f)
Enter value for y: 33
old   5: Y := '& y';
new   5: Y := '33';
NOT PRIME
PL/SQL procedure successfully completed.
```

Task 2(a):

Problem Statement:

Write a procedure to find the N richest branches and their details. The procedure will take N as input and print the details upto N branches. If N is greater then the number of branches, then it will print an error message.

Analysis of the problem:

At first, a count of the number of maximum record MAX_ROW was chosen for following the condition that N must not be larger than numbers of branches. The condition was checked by comparing the input variable N with MAX_ROW to see if it exceeds. If so, a message is printed, and the procedure is exited using a return statement.
Next the top N rows are selected and the results are iterated one by one to show the required information of the branch table.
Finally, the procedure is called using 2 values - one valid and the other invalid, for demonstration purposes.

Any problems faced and how it was solved:

Since the if statement does not allow queries, a different variable had to be taken for storing the value MAX_ROW.
At first, the condition was written as `where rownum = N` but it became obvious that the entries to be selected were the ones with row number less or equal to N.
There were a few minor mistakes that were difficult to identify due to the way PL/SQL shows errors. This was solved by debugging at certain points by printing values of the variables to identify which part of the procedure is not giving expected results.

Code:

```
CREATE OR REPLACE
PROCEDURE N_RICHEST_BRANCHES(N IN NUMBER)
AS
MAX_ROW NUMBER;
CHECK_BRANCH NUMBER;
BEGIN
SELECT COUNT(BRANCH_NAME) INTO MAX_ROW
FROM BRANCH;

IF(N > MAX_ROW) THEN
    DBMS_OUTPUT . PUT_LINE ('N is too large!');
    RETURN;
END IF;
FOR i IN (SELECT *
        FROM (SELECT *
            FROM BRANCH
            ORDER BY ASSETS DESC)
        WHERE ROWNUM<= N) loop
    DBMS_OUTPUT . PUT_LINE ('Branch name: ' || i.branch_name || ', Branch
city: ' || i.branch_city || ', Assets: ' || i.assets);
END LOOP;
END;
/
```

```
BEGIN
    N_RICHEST_BRANCHES(5);
    N_RICHEST_BRANCHES(1000);
END;
/
```

Results:

```
Branch name: Round Hill, Branch city: Horseneck, Assets: 8000000
Branch name: Brighton, Branch city: Brooklyn, Assets: 7000000
Branch name: North Town, Branch city: Rye, Assets: 3700000
Branch name: Redwood, Branch city: Palo Alto, Assets: 2100000
Branch name: Perryridge, Branch city: Horseneck, Assets: 1700000
N is too large!

PL/SQL procedure successfully completed.
```

Task 2(b):

Problem Statement:

Write a procedure to find the customer status (“Green zone”, “Red zone”). If net loan > net balance, then the status should be “Red zone”, else it should be “Green zone”. The procedure will take the name of the customer as input as input and print the status.

Analysis of the problem:

Two number variables are taken to store the results of total balance and total loan amount from queries. Total balance was found by joining depositor and account using appropriate conditions and matching with the parameter of customer name. Total loan was found by joining borrower and loan tables similarly.
Next, an if block is used to check to see if total loan is greater than total balance and if so, status is printed as red zone and otherwise status is printed as green zone.

Any problems faced and how it was solved:

There were a number of problems in this procedure.
Firstly, there were problems in comparing NET_LOAN and NET_BALANCE because for some customer, there might not be any entries in depositor and/or borrower table.
Thus two more number variables was used to count the number of entries of the customer in the depositor and borrower table. If the count in borrower table is 0, NET_LOAN is initiated as zero. Similarly, this is repeated for NET_BALANCE. If count is not 0, the blocks with query is executed.
If the customer name taken as input does not exist in the first place, the procedure gives the wrong answer ‘green zone’. Thus an easy solution would be to check the number of entries in customer table and return from the procedure with a printed message if customer does not exist.
Using natural join gave errors, so instead where conditions were used to join the tables which was much lengthier.

Code:

```
CREATE OR REPLACE
PROCEDURE FIND_CUSTOMER_STATUS(C_NAME IN VARCHAR2)
AS
    NET_LOAN NUMBER;
    NET_BALANCE NUMBER;
```

```
    CHECK_CUSTOMER NUMBER;
    CHECK_DEPOSITOR NUMBER;
    CHECK_BORROWER NUMBER;
BEGIN
    SELECT COUNT(CUSTOMER_NAME) INTO CHECK_CUSTOMER
    FROM CUSTOMER
    WHERE CUSTOMER_NAME = C_NAME;
    IF(CHECK_CUSTOMER = 0) THEN
        DBMS_OUTPUT.PUT_LINE('User does not exist!');
        RETURN;
    END IF;

    SELECT COUNT(CUSTOMER_NAME) INTO CHECK_DEPOSITOR
    FROM DEPOSITOR
    WHERE CUSTOMER_NAME = C_NAME;
    IF(CHECK_DEPOSITOR = 0) THEN
        NET_BALANCE:=0;
    ELSE
        SELECT SUM(ACCOUNT.BALANCE) INTO NET_BALANCE
        FROM DEPOSITOR, ACCOUNT
        WHERE DEPOSITOR.ACCOUNT_NUMBER = ACCOUNT.ACCOUNT_NUMBER
        GROUP BY DEPOSITOR.CUSTOMER_NAME
        HAVING DEPOSITOR.CUSTOMER_NAME = C_NAME;
    END IF;

    SELECT COUNT(CUSTOMER_NAME) INTO CHECK_BORROWER
    FROM BORROWER
    WHERE CUSTOMER_NAME = C_NAME;
    IF(CHECK_BORROWER = 0) THEN
        NET_LOAN:=0;
    ELSE
        SELECT SUM(LOAN.AMOUNT) INTO NET_LOAN
        FROM BORROWER, LOAN
        WHERE BORROWER.LOAN_NUMBER = LOAN.LOAN_NUMBER
        GROUP BY BORROWER.CUSTOMER_NAME
        HAVING BORROWER.CUSTOMER_NAME = C_NAME;
    END IF;

    IF((NET_LOAN) > (NET_BALANCE)) THEN
        DBMS_OUTPUT . PUT_LINE('Red Zone');
    ELSE
        DBMS_OUTPUT . PUT_LINE('Green Zone');
    END IF;
END;
/
```

```
DECLARE
    CUSTOMER_NAME VARCHAR2(15);
BEGIN
    CUSTOMER_NAME:='&CUSTOMER_NAME';
    FIND_CUSTOMER_STATUS(CUSTOMER_NAME);
END;
/
```

Results:

```
Test 01
Enter value for customer_name: McBride
old   4:     CUSTOMER_NAME:='&CUSTOMER_NAME';
new   4:     CUSTOMER_NAME:='McBride';
Red Zone

PL/SQL procedure successfully completed.

Test 02
Enter value for customer_name: Glenn
old   4:     CUSTOMER_NAME:='&CUSTOMER_NAME';
new   4:     CUSTOMER_NAME:='Glenn';
Green Zone

PL/SQL procedure successfully completed.

Test 03
Enter value for customer_name: Nafisa Maliyat
old   4:     CUSTOMER_NAME:='&CUSTOMER_NAME';
new   4:     CUSTOMER_NAME:='Nafisa Maliyat';
User does not exist!

PL/SQL procedure successfully completed.
```

Task 2(c):

Problem Statement:

Write a function to find the tax amount for each customer. A customer is eligible for tax if their net balance is greater then or equal to 750 (do not consider the loan). And amount of tax for one is 8% of the net balance.

Analysis of the problem:

Approach to this problem was the same as 2(b). First it is checked if the customer does exists. If so, -1 was returned with a message. Next, it is checked if the customer exists in the depositor table. If not, the variable NET_BALANCE is initiated to zero. Otherwise, the NET_BALANCE will contain the results of the query calculating total balance of a customer.
The condition for eligibility is checked, and if NET_BALANCE fulfills the conditions, the tax is calculated and printed.Otherwise, -1 is returned with a message.

Any problems faced and how it was solved:

Since this is a function, even if no tax was calculated, something had to be returned. Thus this was solved by returning null at first but it caused problem during printing since it just showed up as blank. Thus another method was to return -1 to indicate tax was not calculated.

Code:

```
CREATE OR REPLACE
FUNCTION CALC_TAX(C_NAME VARCHAR2)
RETURN NUMBER
IS
    TAX NUMBER;
    NET_BALANCE NUMBER;
    CHECK_CUSTOMER NUMBER;
    CHECK_DEPOSITOR NUMBER;
BEGIN
    SELECT COUNT(CUSTOMER_NAME) INTO CHECK_CUSTOMER
    FROM CUSTOMER
    WHERE CUSTOMER_NAME = C_NAME;

    IF(CHECK_CUSTOMER = 0) THEN
        DBMS_OUTPUT.PUT_LINE('User does not exist!');
        RETURN -1;
    END IF;
```

```
    SELECT COUNT(CUSTOMER_NAME) INTO CHECK_DEPOSITOR
    FROM DEPOSITOR
    WHERE CUSTOMER_NAME = C_NAME;

    IF(CHECK_DEPOSITOR = 0) THEN
        NET_BALANCE:=0;
    ELSE
        SELECT SUM(ACCOUNT.BALANCE) INTO NET_BALANCE
        FROM DEPOSITOR, ACCOUNT
        WHERE DEPOSITOR.ACCOUNT_NUMBER = ACCOUNT.ACCOUNT_NUMBER
        GROUP BY DEPOSITOR.CUSTOMER_NAME
        HAVING DEPOSITOR.CUSTOMER_NAME = C_NAME;
    END IF;

    IF(NET_BALANCE <750) then
        DBMS_OUTPUT . PUT_LINE('User is not eligible!');
        RETURN -1;
    ELSE
        TAX := .08 * NET_BALANCE;
        RETURN TAX;
    END IF;

END;
/
```

```
DECLARE
    C_NAME VARCHAR2(15);
BEGIN
    C_NAME:='& C_NAME';
    DBMS_OUTPUT . PUT_LINE('Tax: ' || CALC_TAX(C_NAME));
END;
/
```

Results:

```
Test 01
Enter value for c_name: Hayes
old   4:     C_NAME:='& C_NAME';
new   4:     C_NAME:='Hayes';
Tax: 72

PL/SQL procedure successfully completed.

Test 02
Enter value for c_name: Williams
```

```
old   4:     C_NAME:='& C_NAME';
new   4:     C_NAME:='Williams';
User is not eligible!
Tax: -1

PL/SQL procedure successfully completed.
```

Test 03

```
Enter value for c_name: Nafisa
old   4:     C_NAME:='& C_NAME';
new   4:     C_NAME:='Nafisa';
User does not exist!
Tax: -1

PL/SQL procedure successfully completed.
```

Task 2(d):

Problem Statement:

Write a function to find the customer category based on Table 1. The function will take the name of the customer as input and return the category.

Analysis of the problem:

Similar to the previous two tasks, it is checked if customer exists. If so, the NET_BALANCE and NET_LOAN is calculated. Using the conditions provided, the category of the customer is returned using a varchar2 variable.
In the block calling the function, the customer name is taken as input and the variable returned is printed.

Any problems faced and how it was solved:

There were no problems faced since this is similar to the previous tasks.

Code:

```
CREATE OR REPLACE
FUNCTION CUSTOMER_CATEGORY(C_NAME VARCHAR2)
RETURN VARCHAR2
IS
    CATEGORY VARCHAR2(4);
    NET_BALANCE NUMBER;
    NET_LOAN NUMBER;
    CHECK_CUSTOMER NUMBER;
    CHECK_DEPOSITOR NUMBER;
    CHECK_BORROWER NUMBER;
BEGIN
    SELECT COUNT(CUSTOMER_NAME) INTO CHECK_CUSTOMER
    FROM CUSTOMER
    WHERE CUSTOMER_NAME = C_NAME;
    IF(CHECK_CUSTOMER = 0) THEN
        DBMS_OUTPUT.PUT_LINE('User does not exist!');
        RETURN 'N/A';
    END IF;

    SELECT COUNT(CUSTOMER_NAME) INTO CHECK_DEPOSITOR
    FROM DEPOSITOR
    WHERE CUSTOMER_NAME = C_NAME;
    IF(CHECK_DEPOSITOR = 0) THEN
        NET_BALANCE:=0;
    ELSE
```

```
        SELECT SUM(ACCOUNT.BALANCE) INTO NET_BALANCE
        FROM DEPOSITOR, ACCOUNT
        WHERE DEPOSITOR.ACCOUNT_NUMBER = ACCOUNT.ACCOUNT_NUMBER
        GROUP BY DEPOSITOR.CUSTOMER_NAME
        HAVING DEPOSITOR.CUSTOMER_NAME = C_NAME;
    END IF;
    SELECT COUNT(CUSTOMER_NAME) INTO CHECK_BORROWER
    FROM BORROWER
    WHERE CUSTOMER_NAME = C_NAME;
    IF(CHECK_BORROWER = 0) THEN
        NET_LOAN:=0;
    ELSE
        SELECT SUM(LOAN.AMOUNT) INTO NET_LOAN
        FROM BORROWER, LOAN
        WHERE BORROWER.LOAN_NUMBER = LOAN.LOAN_NUMBER
        GROUP BY BORROWER.CUSTOMER_NAME
        HAVING BORROWER.CUSTOMER_NAME = C_NAME;
    END IF;

    IF(NET_BALANCE>1000 AND NET_LOAN<1000) THEN
        CATEGORY:='C-A1';
    ELSIF(NET_BALANCE < 500 AND NET_LOAN > 2000) THEN
        CATEGORY:='C-C3';
    ELSE
        CATEGORY:='C-B1';
    END IF;

    RETURN CATEGORY;

END;
/
```

```
DECLARE
    C_NAME VARCHAR2(15);
BEGIN
    C_NAME:='& C_NAME';
    DBMS_OUTPUT . PUT_LINE('Category: ' || CUSTOMER_CATEGORY(C_NAME));
END;
/
```

Results:

```
Test 01
Enter value for c_name: Hayes
old   4:     C_NAME:='& C_NAME';
```

```
new   4:     C_NAME:='Hayes';
Category: C-B1

PL/SQL procedure successfully completed.
```

Test 02

```
Enter value for c_name: McBride
old   4:     C_NAME:='& C_NAME';
new   4:     C_NAME:='McBride';
Category: C-C3

PL/SQL procedure successfully completed.
```

Test 03

```
Enter value for c_name: Johnson
old   4:     C_NAME:='& C_NAME';
new   4:     C_NAME:='Johnson';
Category: C-A1

PL/SQL procedure successfully completed.
```

Test 04

```
Enter value for c_name: Nafisa
old   4:     C_NAME:='& C_NAME';
new   4:     C_NAME:='Nafisa';
User does not exist!
Category: N/A

PL/SQL procedure successfully completed.
```