Lab 9 Report

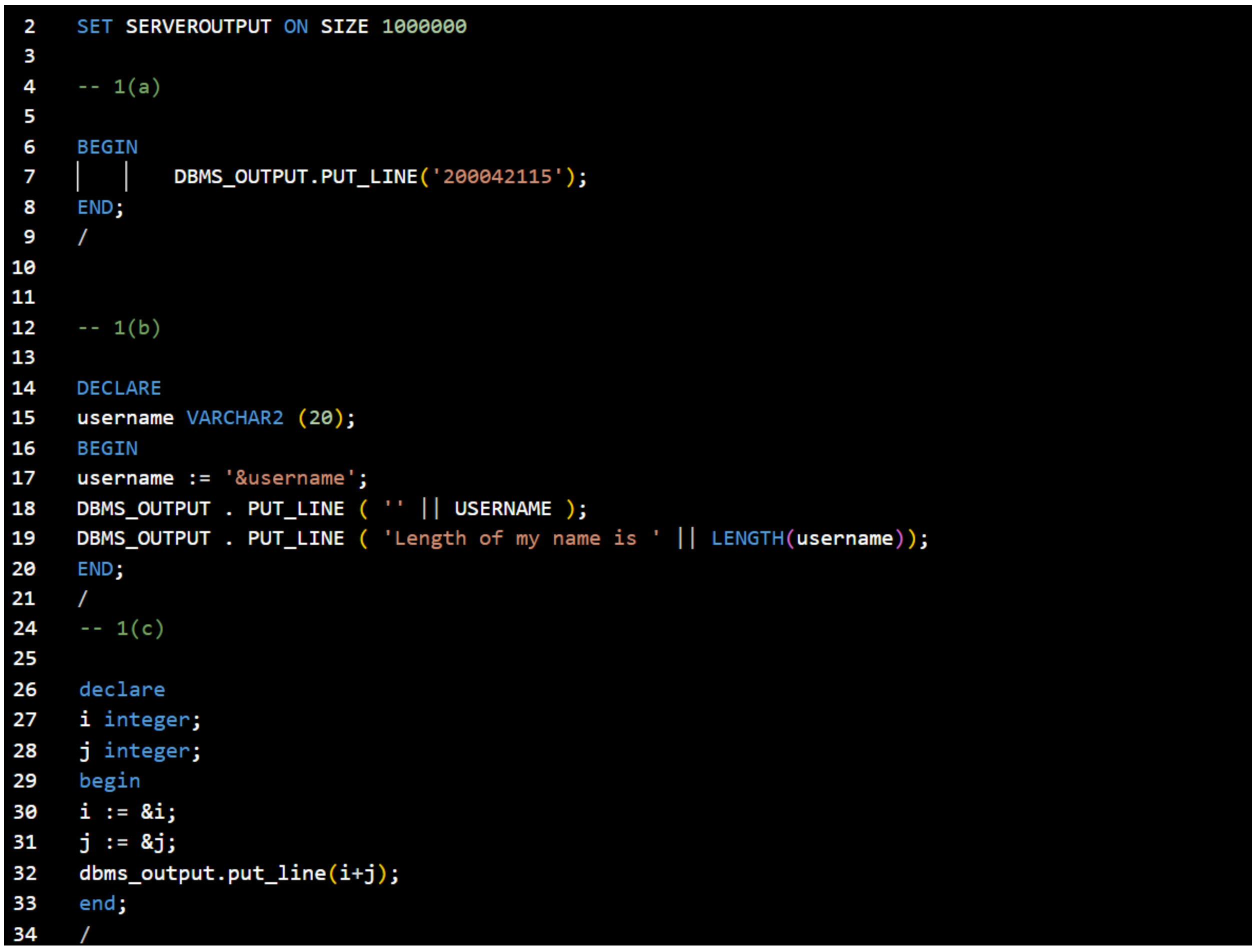
Name	: Mashrur Ahsan
ID	: 200042115
Program	: SWE
Department	: CSE
Course	: CSE 4308

Problem Statements:

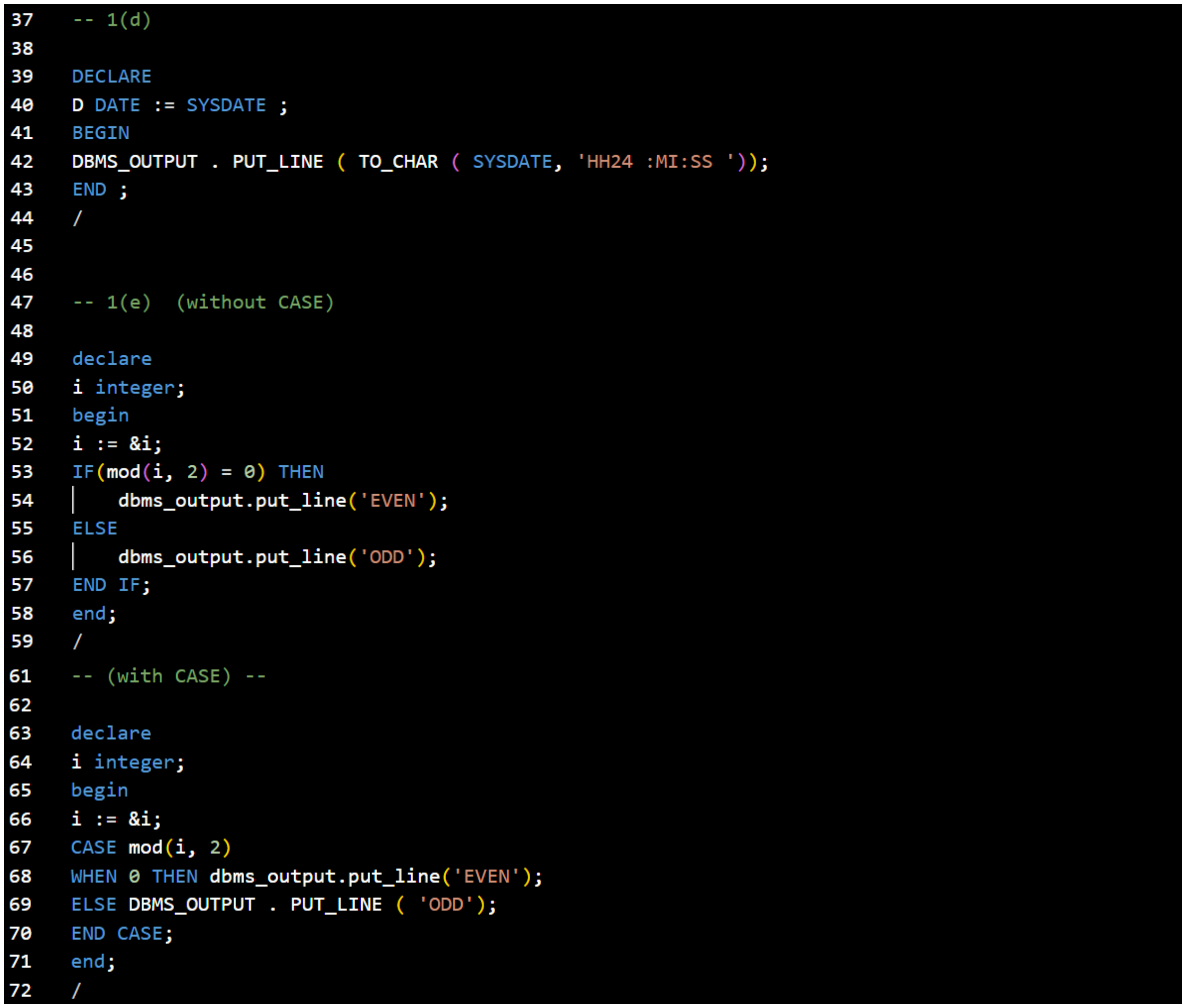
Write PL/SQL statements to perform the following tasks:

1. **Warm-up**:
   (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.
   (e) Take a number as input and print whether it is odd or even (with and without CASE statement).
   (f) Write a procedure that takes a number as argument and prints whether it is a prime number or not.

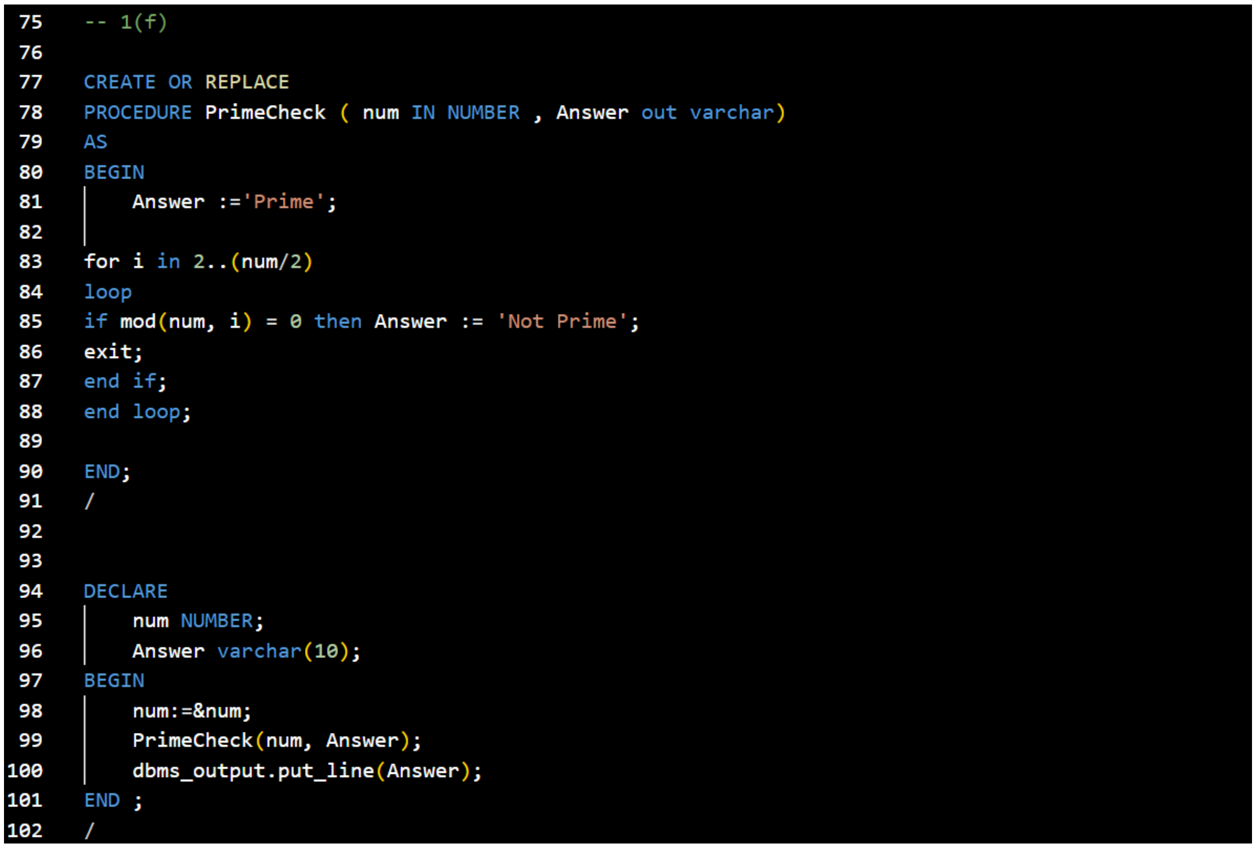
The Solution code(s) for the first task:



```
SET SERVEROUTPUT ON SIZE 1000000

-- 1(a)

BEGIN
        DBMS_OUTPUT.PUT_LINE('200042115');
END;
/


-- 1(b)

DECLARE
username VARCHAR2 (20);
BEGIN
username := '&username';
DBMS_OUTPUT . PUT_LINE ( '' || USERNAME );
DBMS_OUTPUT . PUT_LINE ( 'Length of my name is ' || LENGTH(username));
END;
/
-- 1(c)

declare
i integer;
j integer;
begin
i := &i;
j := &j;
dbms_output.put_line(i+j);
end;
/
```



```
-- 1(d)

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


-- 1(e)  (without CASE)

declare
i integer;
begin
i := &i;
IF(mod(i, 2) = 0) THEN
    dbms_output.put_line('EVEN');
ELSE
    dbms_output.put_line('ODD');
END IF;
end;
/
-- (with CASE) --

declare
i integer;
begin
i := &i;
CASE mod(i, 2)
WHEN 0 THEN dbms_output.put_line('EVEN');
ELSE DBMS_OUTPUT . PUT_LINE ( 'ODD');
END CASE;
end;
/
```



```
75    -- 1(f)
76
77    CREATE OR REPLACE
78    PROCEDURE PrimeCheck ( num IN NUMBER , Answer out varchar)
79    AS
80    BEGIN
81        Answer :='Prime';
82
83    for i in 2..(num/2)
84    loop
85    if mod(num, i) = 0 then Answer := 'Not Prime';
86    exit;
87    end if;
88    end loop;
89
90    END;
91    /
92
93
94    DECLARE
95        num NUMBER;
96        Answer varchar(10);
97    BEGIN
98        num:=&num;
99        PrimeCheck(num, Answer);
100       dbms_output.put_line(Answer);
101   END ;
102   /
```

Explanation:

Line 2: To show results onto the screen we need to SET Serveroutput ON.

Line 42: One way of showing the current system time.

Line 67: The keyword 'CASE' is equivalent to the 'Switch' conditional statement.

Line 78: The procedure takes in a number then returns if it was odd or even.

Line 99: Passing parameters into the procedure. Then Printing the answer.

Problems:

- Syntax error and compilation error was a common occurrence.
- Didn't know how to show current system date until I did the task.
- Took me quite a bit of time to get used to the input and output system.

Schema for the 2nd task:

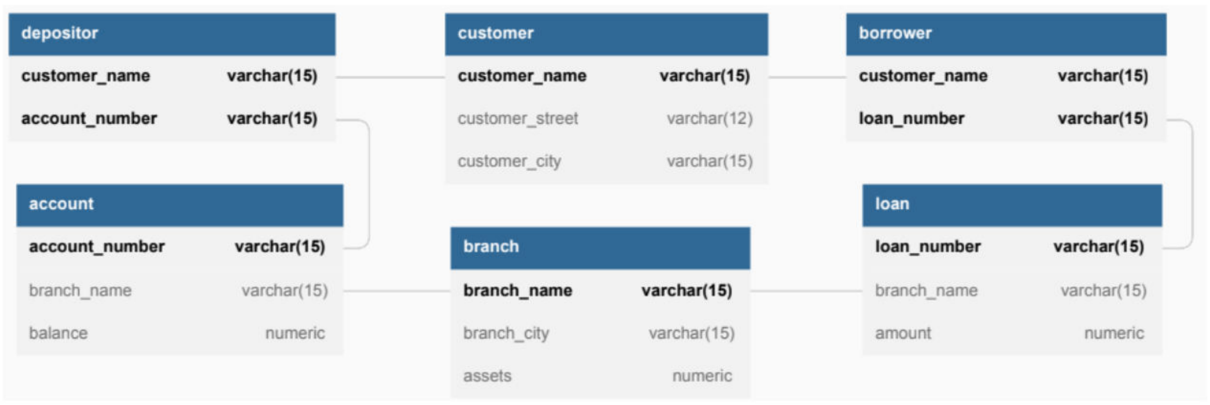
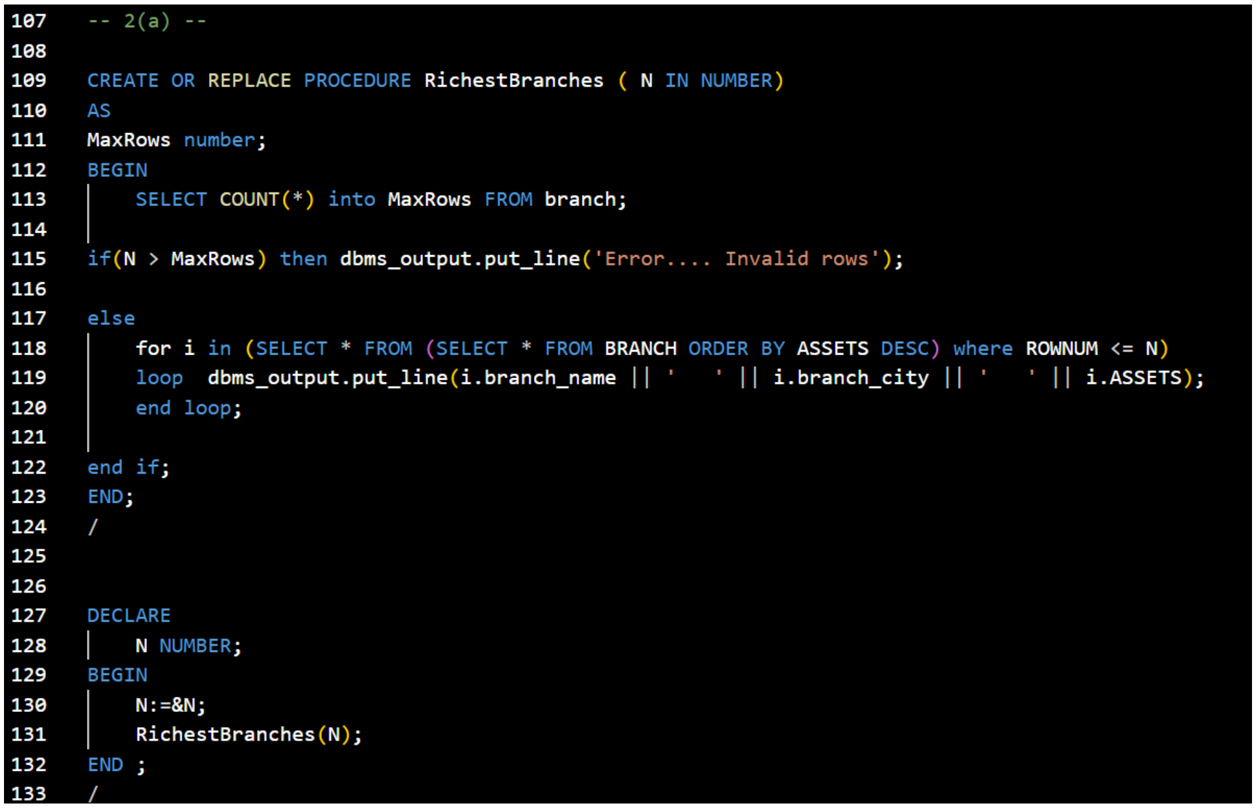


Figure 1: ER Diagram for a banking management system

Problem Statements:

(a) 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.

Solution code:



```
-- 2(a) --

CREATE OR REPLACE PROCEDURE RichestBranches ( N IN NUMBER)
AS
MaxRows number;
BEGIN
    SELECT COUNT(*) into MaxRows FROM branch;

if(N > MaxRows) then dbms_output.put_line('Error.... Invalid rows');

else
    for i in (SELECT * FROM (SELECT * FROM BRANCH ORDER BY ASSETS DESC) where ROWNUM <= N)
    loop  dbms_output.put_line(i.branch_name || '    ' || i.branch_city || '    ' || i.ASSETS);
    end loop;

end if;
END;
/


DECLARE
    N NUMBER;
BEGIN
    N:=&N;
    RichestBranches(N);
END ;
/
```

Explanation:

Line 113: Fetching the number of rows from the Branch table into a variable.

Line 117 to 122: If the passed number is less than or equal to the max rows then it will print out all the details.

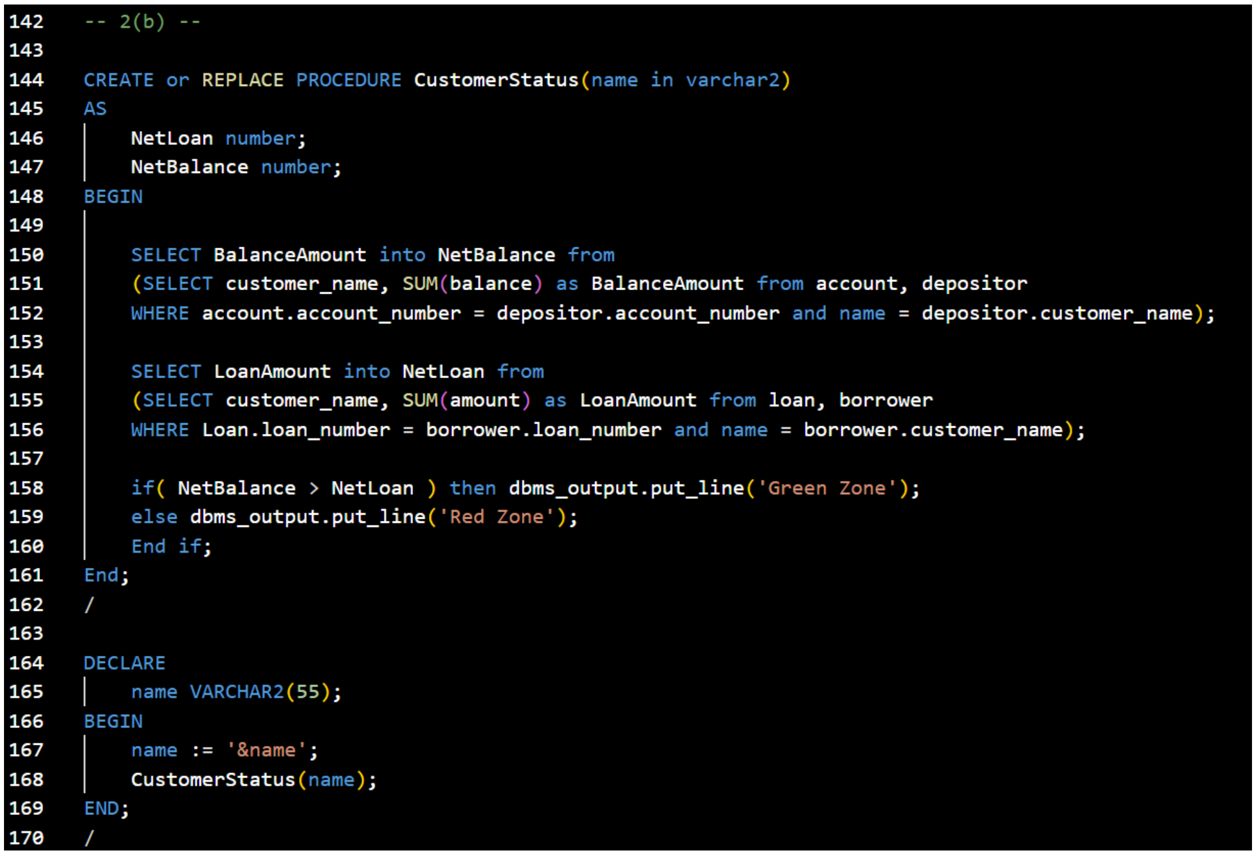
Line 118: Iterating each row.

Problems:

- Syntax error and compilation error was a common occurrence.
- Procedure was being created with compilation errors. Took quite a bit of time to solve the issues.

(b) 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.

Solution Code:



```
142   -- 2(b) --
143
144   CREATE or REPLACE PROCEDURE CustomerStatus(name in varchar2)
145   AS
146       NetLoan number;
147       NetBalance number;
148   BEGIN
149
150       SELECT BalanceAmount into NetBalance from
151       (SELECT customer_name, SUM(balance) as BalanceAmount from account, depositor
152       WHERE account.account_number = depositor.account_number and name = depositor.customer_name);
153
154       SELECT LoanAmount into NetLoan from
155       (SELECT customer_name, SUM(amount) as LoanAmount from loan, borrower
156       WHERE Loan.loan_number = borrower.loan_number and name = borrower.customer_name);
157
158       if( NetBalance > NetLoan ) then dbms_output.put_line('Green Zone');
159       else dbms_output.put_line('Red Zone');
160       End if;
161   End;
162   /
163
164   DECLARE
165       name VARCHAR2(55);
166   BEGIN
167       name := '&name';
168       CustomerStatus(name);
169   END;
170   /
```

Explanation:

Line 150 to 152: Storing the balance of a particular customer into a variable.

Line 154 to 156: Storing the loan of a particular customer into a variable.

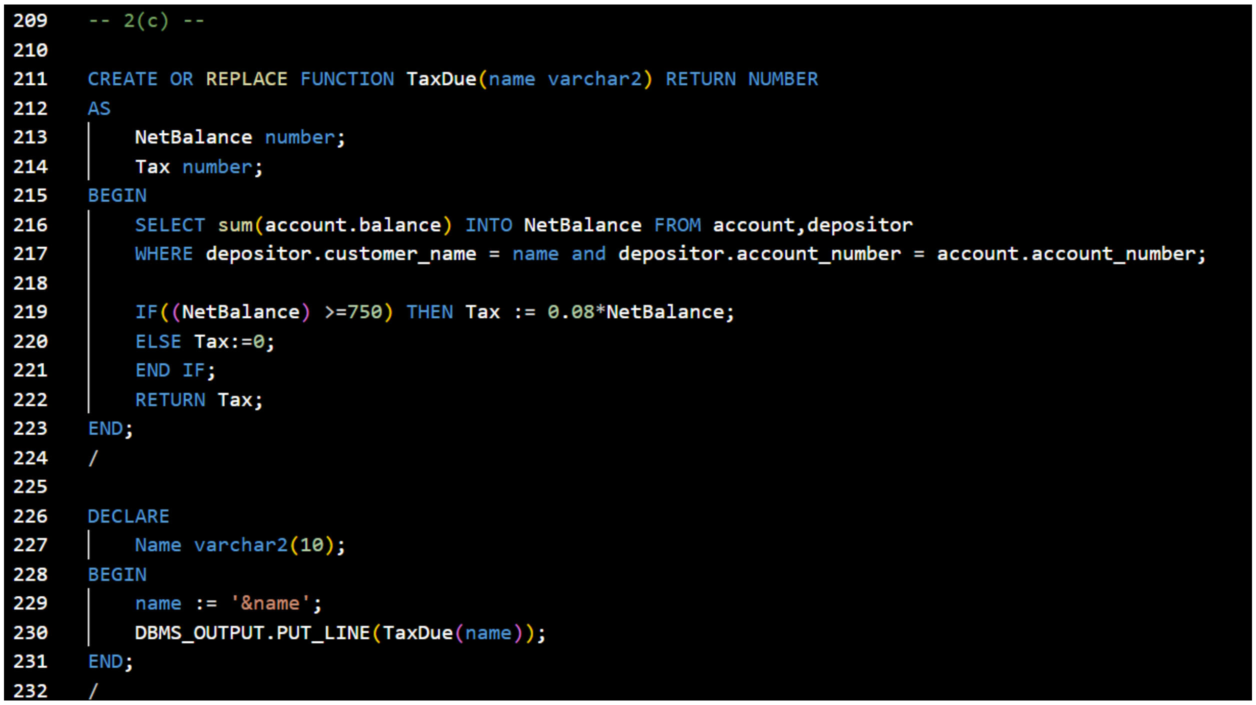
Line 158: Checking condition.

Problems:

- The process of fetching data and storing it into a variable was a bit harder than the previous task.

(c) 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.

Solution Code:



```
209   -- 2(c) --
210
211   CREATE OR REPLACE FUNCTION TaxDue(name varchar2) RETURN NUMBER
212   AS
213       NetBalance number;
214       Tax number;
215   BEGIN
216       SELECT sum(account.balance) INTO NetBalance FROM account,depositor
217       WHERE depositor.customer_name = name and depositor.account_number = account.account_number;
218
219       IF((NetBalance) >=750) THEN Tax := 0.08*NetBalance;
220       ELSE Tax:=0;
221       END IF;
222       RETURN Tax;
223   END;
224   /
225
226   DECLARE
227       Name varchar2(10);
228   BEGIN
229       name := '&name';
230       DBMS_OUTPUT.PUT_LINE(TaxDue(name));
231   END;
232   /
```

Explanation:

Line 216 to 217: Storing the balance of a particular customer into a variable.

Line 219 to 222: Checking and calculating taxes of a particular customer and then returning the value.

Line 230: In case of Functions, we put the function name and it's parameter(s) inside the DBMS_OUTPUT.PUT_LINE();

Problems:

- After doing the previous tasks, this task seemed pretty easy. Therefore, didn't face any problems.

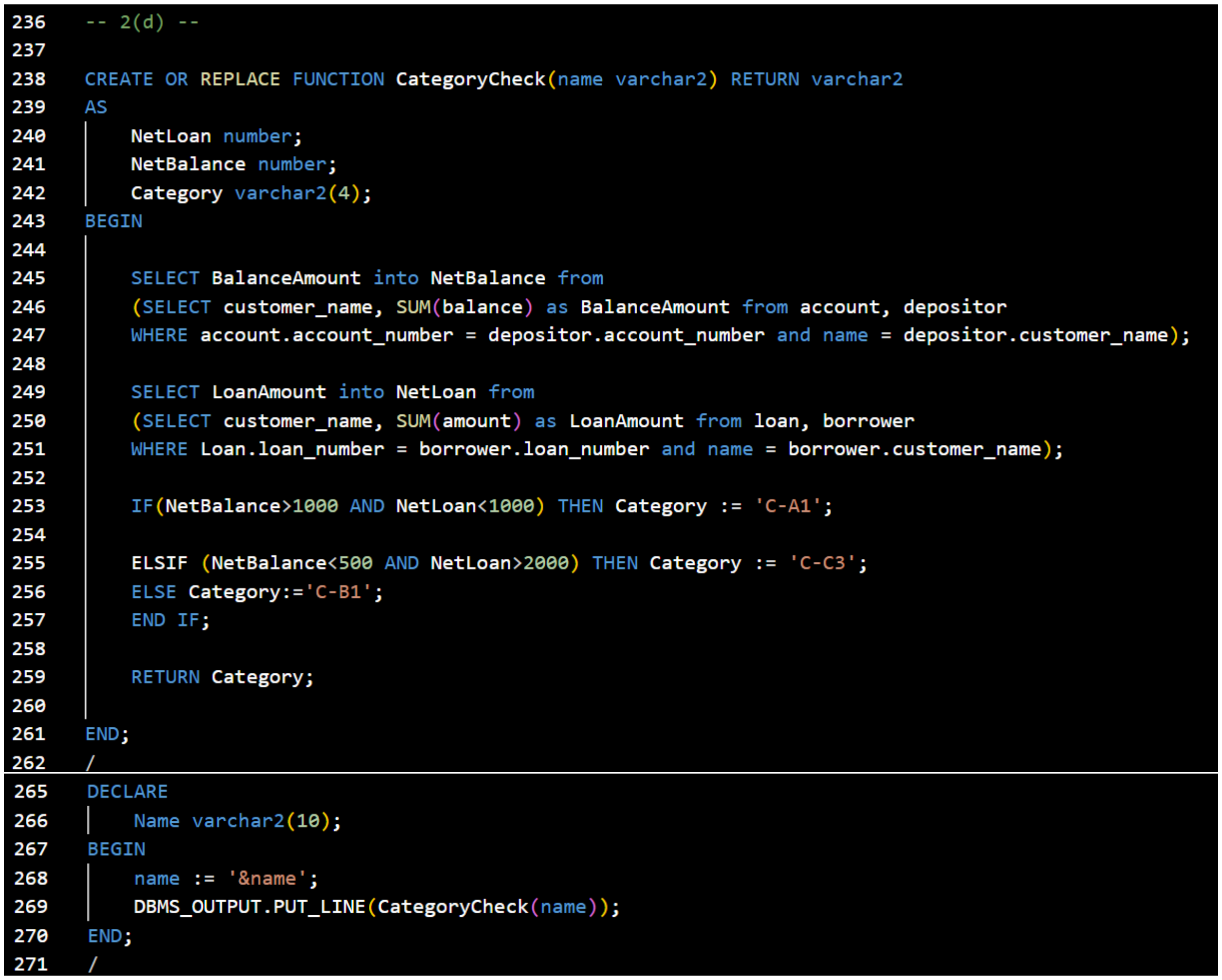
(d) Write a function to find the customer category based on Table 1.

Table 1: Customer Category Table for Question 2(d).

Customer Category	Total Balance	Total Loan
C-A1	>1000	<1000
C-C3	<500	>2000
C-B1	Neither C-A1 nor C-C3	

The function will take the name of the customer as input and return the category.

Solution Code:



```
-- 2(d) --

CREATE OR REPLACE FUNCTION CategoryCheck(name varchar2) RETURN varchar2
AS
    NetLoan number;
    NetBalance number;
    Category varchar2(4);
BEGIN

    SELECT BalanceAmount into NetBalance from
    (SELECT customer_name, SUM(balance) as BalanceAmount from account, depositor
    WHERE account.account_number = depositor.account_number and name = depositor.customer_name);

    SELECT LoanAmount into NetLoan from
    (SELECT customer_name, SUM(amount) as LoanAmount from loan, borrower
    WHERE Loan.loan_number = borrower.loan_number and name = borrower.customer_name);

    IF(NetBalance>1000 AND NetLoan<1000) THEN Category := 'C-A1';

    ELSIF (NetBalance<500 AND NetLoan>2000) THEN Category := 'C-C3';
    ELSE Category:='C-B1';
    END IF;

    RETURN Category;

END;
/
DECLARE
    Name varchar2(10);
BEGIN
    name := '&name';
    DBMS_OUTPUT.PUT_LINE(CategoryCheck(name));
END;
/
```

Explanation:

<u>Line 245 to 247</u>: Similar to the previous tasks.

<u>Line 249 to 251</u>: Similar to the previous tasks.

<u>Line 253 to 259</u>: Checking conditions and returning the Category type.

Problems:

- After doing the previous tasks, this task seemed pretty easy. Therefore, didn’t face any problems.