Midterm, part 2: Creating your PLSQL Application

Instructions

1. We are playing the role of the lead developer for our consulting firm that designs solutions for organizations of all sizes.
2. Each problem will contain several questions related to each other.
3. Read the introduction for each question to know more about the context of the problem.
4. The “GOAL” section includes a brief description of the problem that we need to solve. Refer to it when in doubt.

Problem #1:

1. 10pts. Write a function that takes a) a temperature, and b) whether it is Celsius or Fahrenheit. The function should **return** the equivalent temperature in the opposite scale (if the input was in Celsius it should convert it to Fahrenheit, if it was in Fahrenheit degrees, it should return it in Celsius degrees. Round the temperature before returning the temperature to the user (round up or down are both acceptable, the goal is simply to eliminate the fractions).

Fahrenheit to Celsius formula:

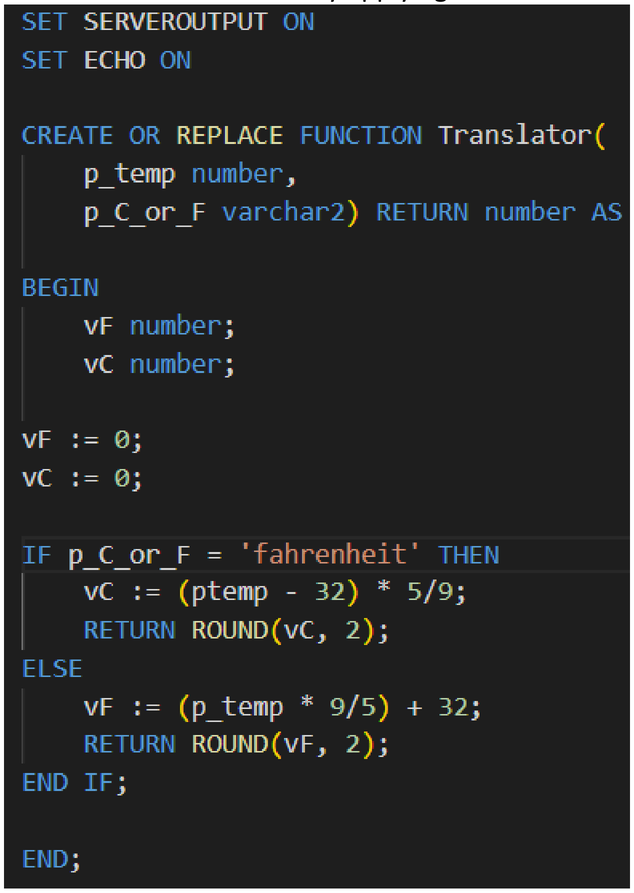
Fahrenheit = (Celsius_Degrees*9/5)+32

Celsius to Fahrenheit formula:

Celsius = (Fahrenheit_Degrees-32)*5/9

HINT:

We can round a number by applying the round function to it -> ROUND(number [, decimal_places])

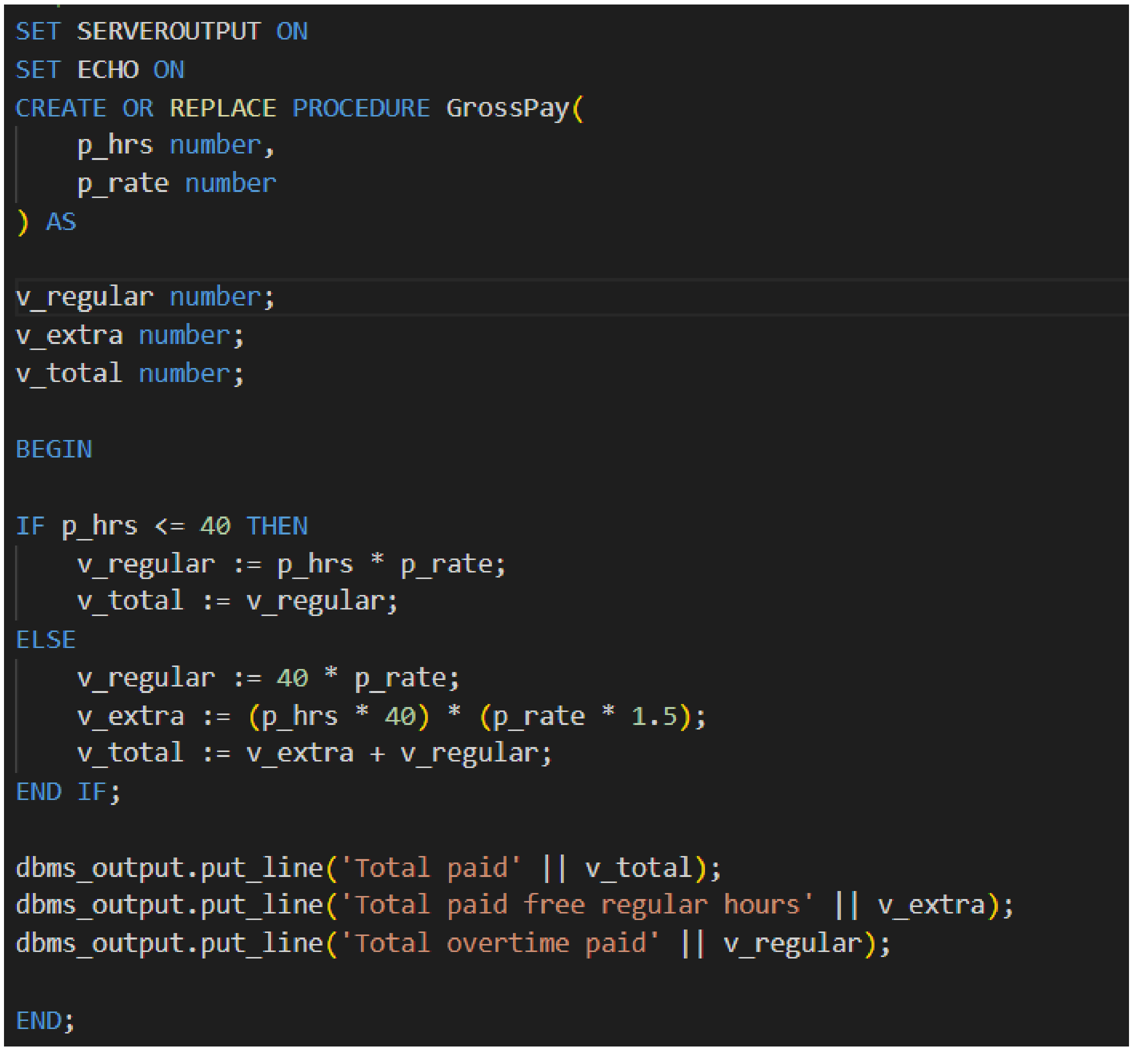


```
SET SERVEROUTPUT ON
SET ECHO ON

CREATE OR REPLACE FUNCTION Translator(
    p_temp number,
    p_C_or_F varchar2) RETURN number AS

BEGIN
    vF number;
    vC number;

vF := 0;
vC := 0;

IF p_C_or_F = 'fahrenheit' THEN
    vC := (ptemp - 32) * 5/9;
    RETURN ROUND(vC, 2);
ELSE
    vF := (p_temp * 9/5) + 32;
    RETURN ROUND(vF, 2);
END IF;

END;
```

2. 15pts. Write a procedure that takes an individuals' hours worked and the hourly rate paid per hour as inputs to calculate the individuals' gross pay. Use the hourly rate to calculate how much will be paid for up to the first 40 hours, and 1.5 times the hourly rate for every hour worked above 40. Finally, print/show a message letting the user know how much was paid for the regular hours worked, and how much was paid for the Extra hours.

Summary:

- Calculate pay for an individual based on how many hours that person worked and their hourly pay rate.
- Use regular pay rate to calculate how much was earned for the first 40 hours.
- Use 1.5 times the regular pay rate to calculate how much was earned from every hour above 40.
- Print/show a message letting the user know 1) how much she was paid in total and how much of that total was earned from 2) working the regular working hours (up to 40), and 3) from the extra pay.



```
SET SERVEROUTPUT ON
SET ECHO ON
CREATE OR REPLACE PROCEDURE GrossPay(
    p_hrs number,
    p_rate number
) AS

v_regular number;
v_extra number;
v_total number;

BEGIN

IF p_hrs <= 40 THEN
    v_regular := p_hrs * p_rate;
    v_total := v_regular;
ELSE
    v_regular := 40 * p_rate;
    v_extra := (p_hrs * 40) * (p_rate * 1.5);
    v_total := v_extra + v_regular;
END IF;

dbms_output.put_line('Total paid' || v_total);
dbms_output.put_line('Total paid free regular hours' || v_extra);
dbms_output.put_line('Total overtime paid' || v_regular);

END;
```

Problem #2: Calculate key financial metrics for a Real Estate Investing Organization

INTRODUCTION

For this section, we are playing the role of a Real Estate Investor looking to buy a new property. After seeing dozens of properties, we would like to request a loan from the bank but are unsure how much we can ask for. We will create a procedure that will tell us the maximum amount that we can safely request from the bank based on the 3 most common approaches for Maximum Loan (Debt) Calculation that banks use.

CONTEXT: (Optional, but good to know what we are doing)

When a real estate investor requests a loan from the bank, the bank personnel will evaluate how much that person may borrow based on 3 criteria:

NOTE: [The goal is to recreate these formulas as provided so that we can be run from a procedure. This will allow us to compare the numbers they provide. **The key for you will be to create 1) separate functions, 2) run them from a procedure, 3) choose the lowest value as the winner between the 3**]

- **Maximum Loan to Value Ratio**: The banker will look at what is the maximum loan the bank is willing to offer for any investment. For example, there may be a policy that says the bank can only finance up to 75% of the value of any project. Therefore, if a person wanted a loan for a property worth $100,000, the maximum loan allowed would be 75% of the $100,000 or $75,000 ($100,000*75% = $75,000)
- **Loan-to-Cost**: The banker may look at the expected costs (what comes out of the pocket) the investor will have rather than the expected value of the project. The bank may have a policy that they can only finance up to a certain percentage of the cost of the project
- **Debt Service Coverage Ratio (DSCR)**: A measure of whether you are likely to be able to repay the loan. It is a fixed number chosen by the bank that represents the ratio of your yearly income to the costs of the investment. In other words, it looks at the yearly income of the project to see if you can cover the costs of the loan and a little bit more.
  Example of DSCR:
  The yearly income of your project could be $130. Your bank has the policy that they only lend money up to the point where debt service coverage ratio is 1.25 (you can cover 100% of your loan and are left with an extra 25% to cover other expenses).
  Def.:

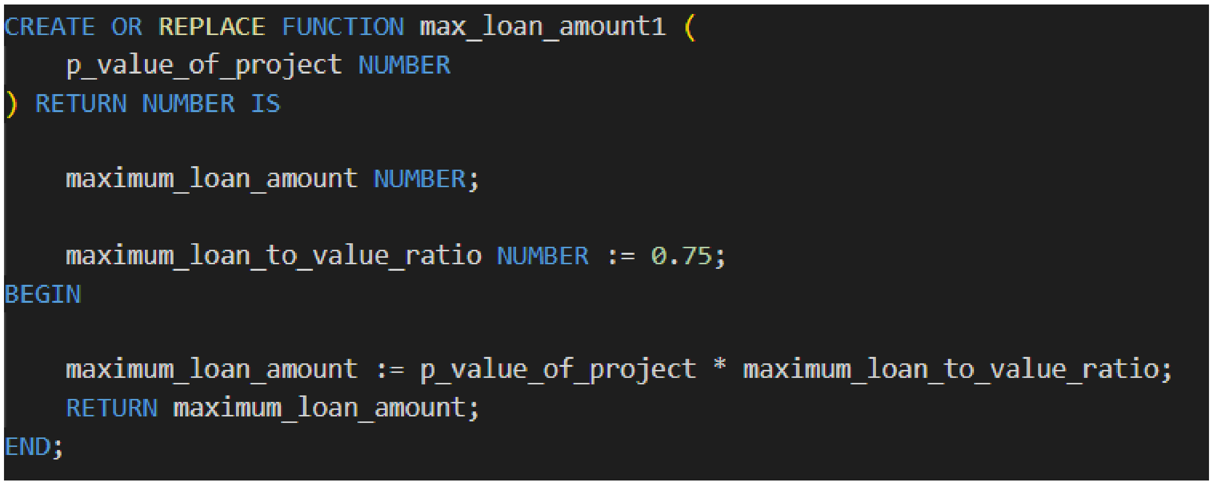
  Cost: What comes out of the pocket of an investor to acquire the project.
  Value: What the project is worth based on how much income it can generate.

3. 5pts. Create a function that calculates the Maximum Loan to Value Ratio and returns the maximum loan amount. Recreate this calculation based on the formula below:

   Maximum_Loan_Amount := Value_of_Project * Maximum_Loan_to_Value_Ratio

   GOAL:
   a) Set the "Maximum_Loan_to_Value_Ratio" in your function as a fixed 75% or .75.
   b) Get the Value of the project as an input for your calculation.
   c) Return the Maximum Loan Amount.

Paste your code below:



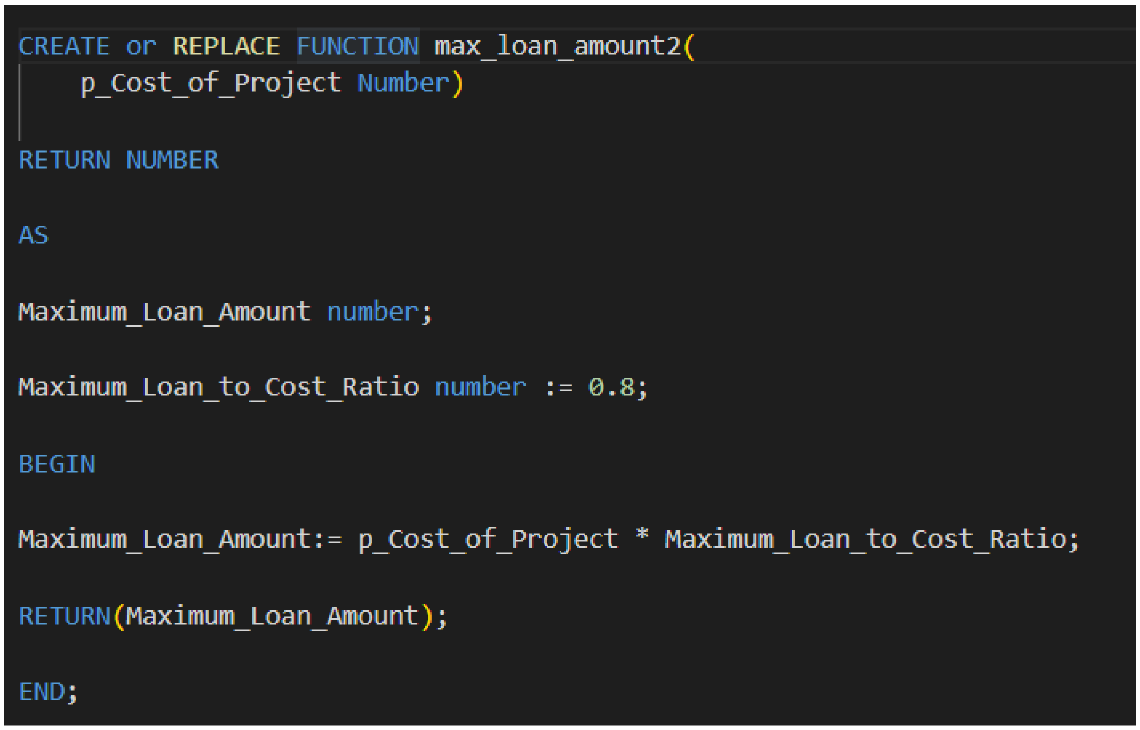
```
CREATE OR REPLACE FUNCTION max_loan_amount1 (
    p_value_of_project NUMBER
) RETURN NUMBER IS

    maximum_loan_amount NUMBER;

    maximum_loan_to_value_ratio NUMBER := 0.75;
BEGIN

    maximum_loan_amount := p_value_of_project * maximum_loan_to_value_ratio;
    RETURN maximum_loan_amount;
END;
```

4. 5pts. Create a function that calculates the Maximum Loan to Cost Ratio and returns the maximum loan amount. Recreate this calculation based on the formula below:

   Maximum_Loan_Amount := Cost_of_Project * Maximum_Loan_to_Cost_Ratio

   GOAL:
   d) Set the "Maximum_Loan_to_Cost_Ratio" in your function as a fixed 80% or .8.
   e) Get the **Cost** of the project as an input for your calculation.
   f) Return the Maximum Loan Amount.

Paste your code below:



```
CREATE or REPLACE FUNCTION max_loan_amount2(
    p_Cost_of_Project Number)

RETURN NUMBER

AS

Maximum_Loan_Amount number;

Maximum_Loan_to_Cost_Ratio number := 0.8;

BEGIN

Maximum_Loan_Amount:= p_Cost_of_Project * Maximum_Loan_to_Cost_Ratio;

RETURN(Maximum_Loan_Amount);

END;
```

5. pts. Create a function that calculates the Maximum Loan Amount to Cost Ratio and **returns** the **maximum loan amount**. Recreate this calculation based on the formula below:

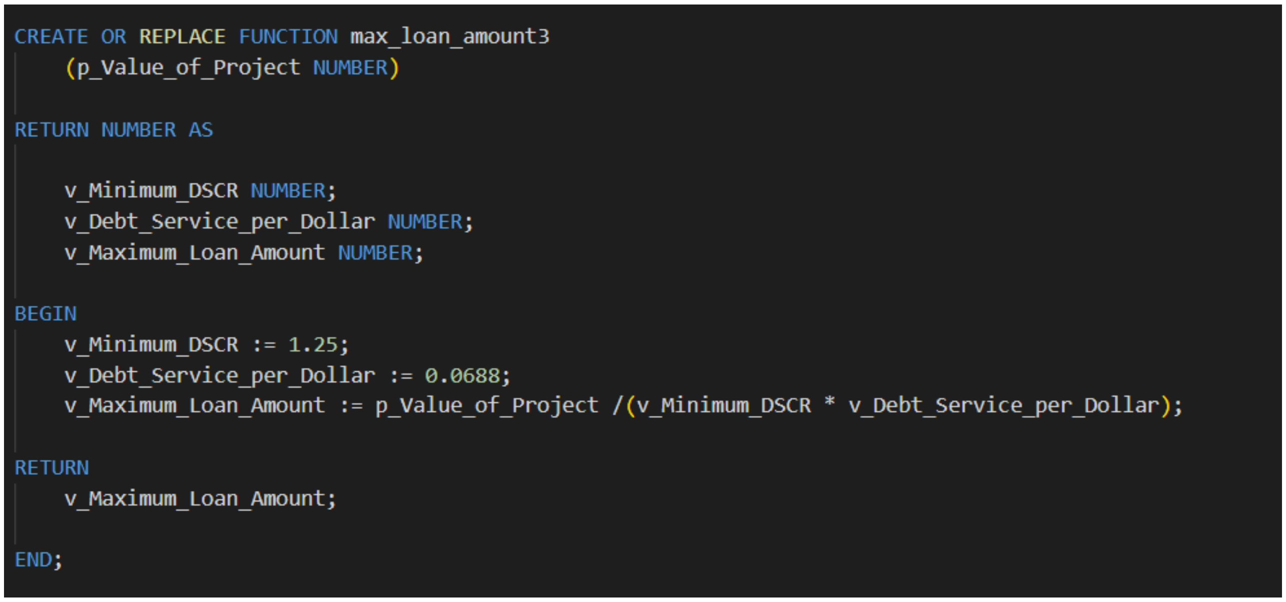
   Maximum_Loan_Amount := Value_Project/(Minimum_DSCR*Debt_Service_per_Dollar)

   GOAL:
   a) Set the "Minimum _DSCR" in your function as a fixed **1.25**.
   b) Get the **Value** of the project as an input for your calculation.
   c) Get the interest rate as an input for your calculation.
   d) Set the "Debt_Service_per_Dollar" in your function to a fixed **0.0688**.
   e) Return the Maximum Loan Amount.

Sample formula:

   Maximum_Loan_Amount := Value_Project/(1.25*0.0688)

Paste your code below:


```sql
CREATE OR REPLACE FUNCTION max_loan_amount3
    (p_Value_of_Project NUMBER)

RETURN NUMBER AS

    v_Minimum_DSCR NUMBER;
    v_Debt_Service_per_Dollar NUMBER;
    v_Maximum_Loan_Amount NUMBER;

BEGIN
    v_Minimum_DSCR := 1.25;
    v_Debt_Service_per_Dollar := 0.0688;
    v_Maximum_Loan_Amount := p_Value_of_Project /(v_Minimum_DSCR * v_Debt_Service_per_Dollar);

RETURN
    v_Maximum_Loan_Amount;

END;
```

6. 10pts. Now it's time to put these together. We will create a procedure that takes the Value of the project, and its costs as inputs. It must also call the functions mentioned above to get the 3 estimates of the Maximum loan amount. Finally, compare all 3 estimates to determine which one is the lowest. Print the message following 4 messages in that order:

"You maximum loan amount is [Lowest_Amount]!"

"Your Maximum loan based on the Loan to Value is: [Maximum_LoantoValue]."

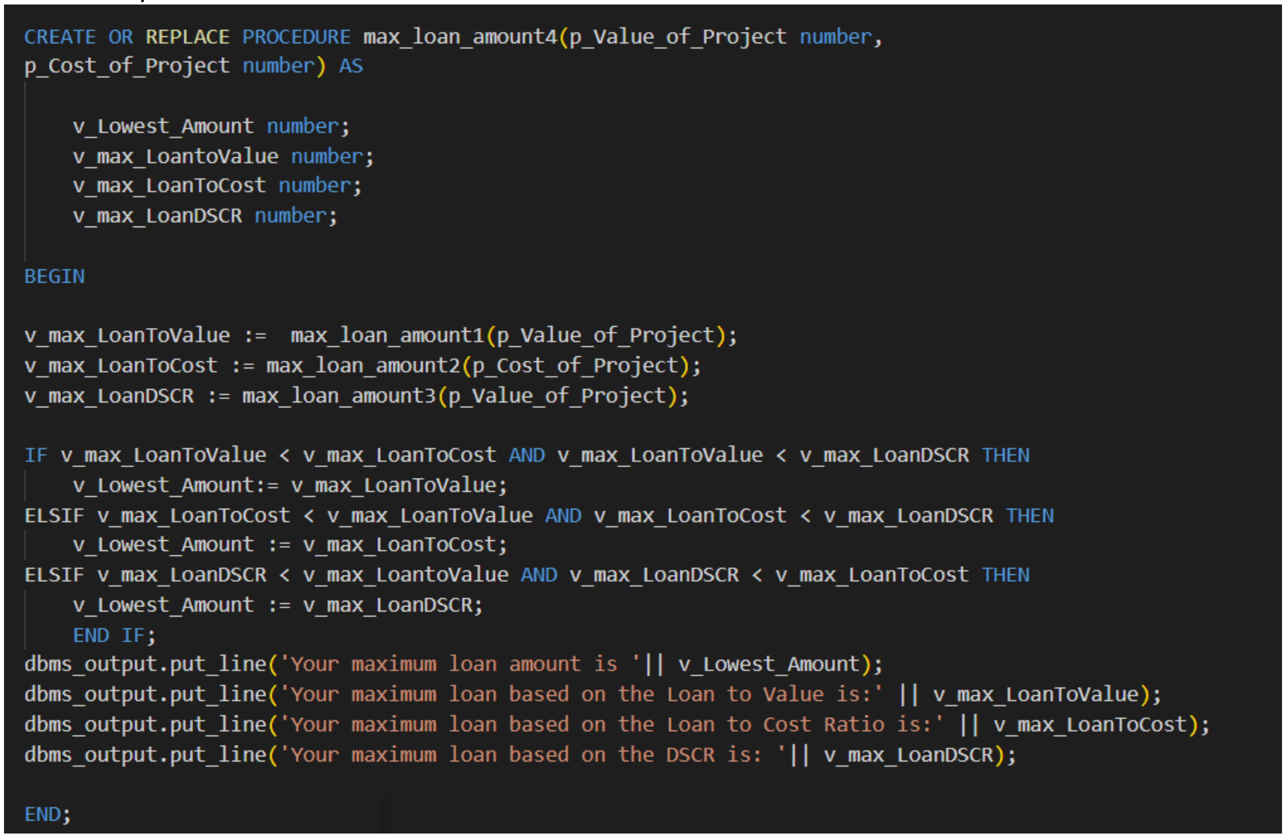
"Your Maximum loan based on the Loan to Cost Ratio is: [Maximum_LoanToCost]."

"Your Maximum loan based on the DSCR is: [Maximum_LoanDSCR]."

<u>Where:</u>

- [Lowest_Amount] is the lowest of the 3 maximum loans you got from the functions.
- [Maximum_LoantoValue] is the maximum loan from the Maximum Loan to Value Ratio function.
- [Maximum_LoanToCost] is the maximum loan from the Maximum Loan to Cost Ratio function.
- [Maximum_LoanDSCR] is the maximum loan from the Maximum Loan Based on DSCR function.

Paste your code below:



```
CREATE OR REPLACE PROCEDURE max_loan_amount4(p_Value_of_Project number,
p_Cost_of_Project number) AS

    v_Lowest_Amount number;
    v_max_LoantoValue number;
    v_max_LoanToCost number;
    v_max_LoanDSCR number;

BEGIN

v_max_LoanToValue :=  max_loan_amount1(p_Value_of_Project);
v_max_LoanToCost := max_loan_amount2(p_Cost_of_Project);
v_max_LoanDSCR := max_loan_amount3(p_Value_of_Project);

IF v_max_LoanToValue < v_max_LoanToCost AND v_max_LoanToValue < v_max_LoanDSCR THEN
    v_Lowest_Amount:= v_max_LoanToValue;
ELSIF v_max_LoanToCost < v_max_LoanToValue AND v_max_LoanToCost < v_max_LoanDSCR THEN
    v_Lowest_Amount := v_max_LoanToCost;
ELSIF v_max_LoanDSCR < v_max_LoantoValue AND v_max_LoanDSCR < v_max_LoanToCost THEN
    v_Lowest_Amount := v_max_LoanDSCR;
    END IF;
dbms_output.put_line('Your maximum loan amount is '|| v_Lowest_Amount);
dbms_output.put_line('Your maximum loan based on the Loan to Value is:' || v_max_LoanToValue);
dbms_output.put_line('Your maximum loan based on the Loan to Cost Ratio is:' || v_max_LoanToCost);
dbms_output.put_line('Your maximum loan based on the DSCR is: '|| v_max_LoanDSCR);

END;
```

7. 10pts. Finally, take the functions and procedures for questions 3, 4, 5, and 6 and create a package that contains all of them. Make sure to create the package definition in the first Box, and the Package Body in the 2nd Box below:

HINT: We are evaluating the creation of the Package components here, not the quality of the code. Therefore, you can correctly complete these exercises even if there are major errors in the code as long as the package is done properly.

INSERT Package Definition HERE:

```
---Question 7---

---Defitnion---

CREATE OR REPLACE Package Max_Loan AS

FUNCTION max_loan_amount1 (
    p_value_of_project NUMBER
) RETURN NUMBER;

FUNCTION max_loan_amount2 (
    p_Cost_of_Project Number
) RETURN NUMBER;


FUNCTION max_loan_amount3(
    p_Value_of_Project NUMBER
) RETURN NUMBER;

PROCEDURE max_loan_amount4(
    p_Value_of_Project number,
    p_Cost_of_Project number);

END Max_Loan;
```

INSERT Package Body HERE:

```
--Body----
SET SERVEROUTPUT ON
SET ECHO ON

---Package Body----
CREATE OR REPLACE PACKAGE BODY Max_Loan AS
--question 3--
FUNCTION max_loan_amount1 (
    p_value_of_project NUMBER
) RETURN NUMBER AS

   maximum_loan_amount NUMBER;

    maximum_loan_to_value_ratio NUMBER := 0.75;
BEGIN

    maximum_loan_amount := p_value_of_project * maximum_loan_to_value_ratio;
    RETURN maximum_loan_amount;
END;

--question 4--

FUNCTION max_loan_amount2 (
    p_Cost_of_Project Number)
    RETURN NUMBER

AS

Maximum_Loan_Amount number;

Maximum_Loan_to_Cost_Ratio number := 0.8;

BEGIN

Maximum_Loan_Amount:= p_Cost_of_Project * Maximum_Loan_to_Cost_Ratio;

RETURN(Maximum_Loan_Amount);

END;

--Question 5--
FUNCTION max_loan_amount3(p_Value_of_Project NUMBER)

RETURN NUMBER AS

    v_Minimum_DSCR NUMBER;
    v_Debt_Service_per_Dollar NUMBER;
    v_Maximum_Loan_Amount NUMBER;

BEGIN
    v_Minimum_DSCR := 1.25;
    v_Debt_Service_per_Dollar := 0.0688;
    v_Maximum_Loan_Amount := p_Value_of_Project /(v_Minimum_DSCR *
v_Debt_Service_per_Dollar);

RETURN
    v_Maximum_Loan_Amount;

END;

--Question 6--

PROCEDURE max_loan_amount4(p_Value_of_Project number,
p_Cost_of_Project number) AS

    v_Lowest_Amount number;
    v_max_LoantoValue number;
    v_max_LoanToCost number;
    v_max_LoanDSCR number;

BEGIN

v_max_LoanToValue :=  max_loan_amount1(p_Value_of_Project);
v_max_LoanToCost := max_loan_amount2(p_Cost_of_Project);
v_max_LoanDSCR := max_loan_amount3(p_Value_of_Project);

IF v_max_LoanToValue < v_max_LoanToCost AND v_max_LoanToValue < v_max_LoanDSCR
THEN
    v_Lowest_Amount:= v_max_LoanToValue;
ELSIF v_max_LoanToCost < v_max_LoanToValue AND v_max_LoanToCost < v_max_LoanDSCR
THEN
    v_Lowest_Amount := v_max_LoanToCost;
ELSIF v_max_LoanDSCR < v_max_LoantoValue AND v_max_LoanDSCR < v_max_LoanToCost
THEN
    v_Lowest_Amount := v_max_LoanDSCR;
    END IF;
dbms_output.put_line('Your maximum loan amount is '|| v_Lowest_Amount);
dbms_output.put_line('Your maximum loan based on the Loan to Value is:' ||
v_max_LoanToValue);
dbms_output.put_line('Your maximum loan based on the Loan to Cost Ratio is:' ||
v_max_LoanToCost);
dbms_output.put_line('Your maximum loan based on the DSCR is: '|| v_max_LoanDSCR);

END;
END Max_Loan;
```

Problem #3: Update the prices of products in our database

8. 10pts. Create a procedure that retrieves the MSRP of each product from the database. Then:
   a) Use a user input to specify which productline will be updated.
   b) Update the price of each product of the specified productline based on a percentage provided by the user of the function.

For example, a user may want to increase the price of “Vintage Cars” by 8% due to inflation in the sector.