**Week 2 – PLSQL SupersetId-6431499**

**Name : Balaji V**

**Exercise 1: Control Structures**

**Scenario1:**

The bank wants to apply a discount to loan interest rates for customers above 60 years old.

**Question:** Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

**Code:**

DECLARE

v\_customer\_id Customers.CustomerID%TYPE;

v\_name Customers.Name%TYPE;

v\_dob Customers.DOB%TYPE;

v\_age NUMBER;

BEGIN

FOR cust\_rec IN(SELECT CustomerID, Name, DOB FROM Customers) LOOP

v\_customer\_id:=cust\_rec.CustomerID;

v\_name:=cust\_rec.Name;

v\_dob:=cust\_rec.DOB;

v\_age:=TRUNC(MONTHS\_BETWEEN(SYSDATE,v\_dob)/12);

IF v\_age > 60 THEN

UPDATE Loans

SET InterestRate=InterestRate-1

WHERE CustomerID=v\_customer\_id;

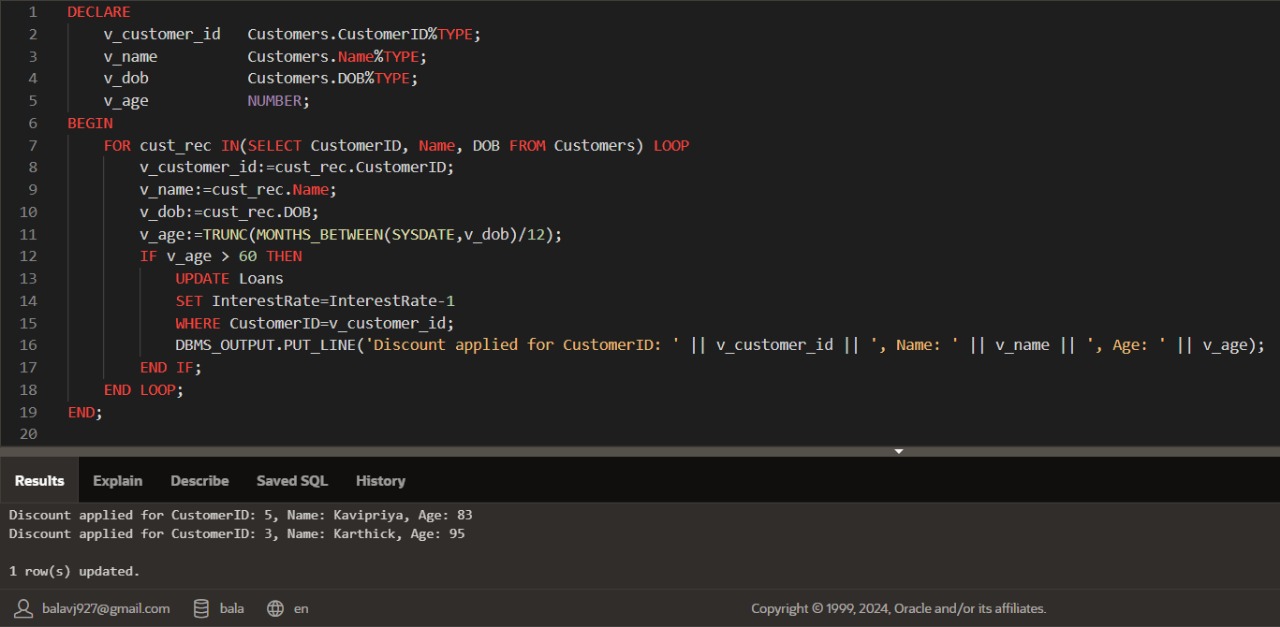
DBMS\_OUTPUT.PUT\_LINE('Discount applied for CustomerID: ' || v\_customer\_id || ', Name: ' || v\_name || ', Age: ' || v\_age);

END IF;

END LOOP;

END;

**Output:**



**Scenario 2:**

A customer can be promoted to VIP status based on their balance.

**Question:** Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

**Code:**

ALTER TABLE Customers ADD (IsVIP VARCHAR2(5));

BEGIN

FOR cust\_rec IN(SELECT CustomerID, Name, Balance FROM Customers) LOOP

IF cust\_rec.Balance > 10000 THEN

UPDATE Customers

SET IsVIP='TRUE'

WHERE CustomerID=cust\_rec.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('CustomerID: ' || cust\_rec.CustomerID || ', Name: ' || cust\_rec.Name || ' is now marked as VIP.');

ELSE

UPDATE Customers

SET IsVIP='FALSE'

WHERE CustomerID=cust\_rec.CustomerID;

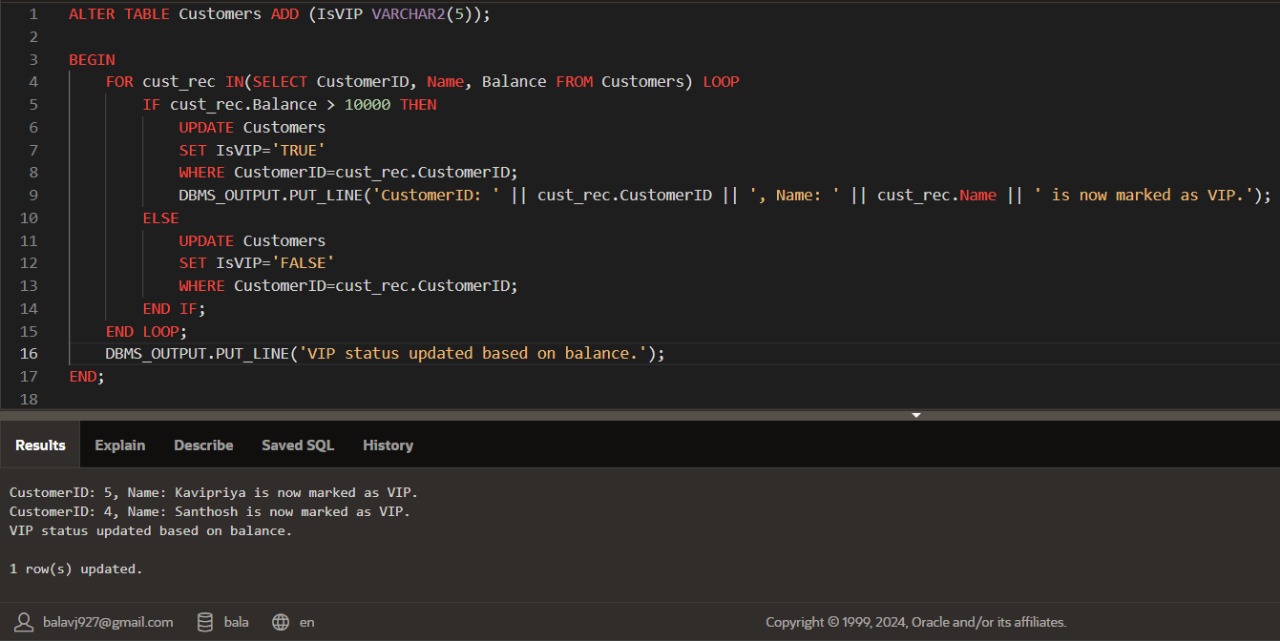
END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('VIP status updated based on balance.');

END;

**Output:**



**Scenario 3:**

The bank wants to send reminders to customers whose loans are due within the next 30 days.

**Question:** Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

**Code:**

DECLARE

v\_id Customers.CustomerID%TYPE;

v\_name Customers.Name%TYPE;

v\_due\_date Loans.EndDate%TYPE;

BEGIN

FOR loan\_rec IN(

SELECT c.CustomerID, c.Name, l.EndDate

FROM Loans l

JOIN Customers c ON l.CustomerID=c.CustomerID

WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

v\_id:=loan\_rec.CustomerID;

v\_name:=loan\_rec.Name;

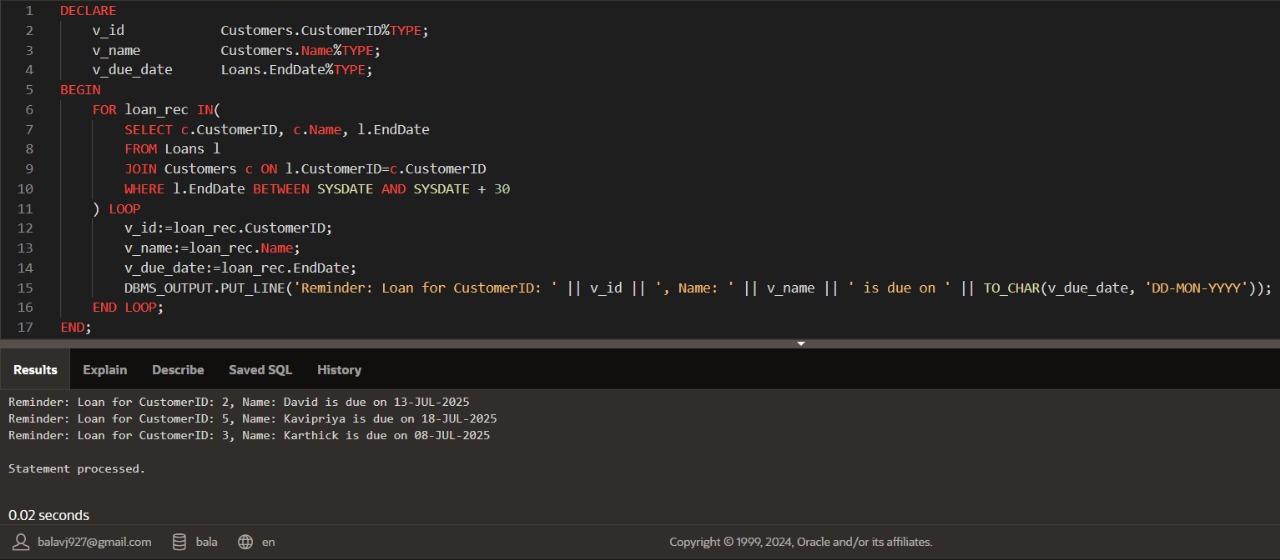
v\_due\_date:=loan\_rec.EndDate;

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan for CustomerID: ' || v\_id || ', Name: ' || v\_name || ' is due on ' || TO\_CHAR(v\_due\_date, 'DD-MON-YYYY'));

END LOOP;

END;

**Output:**



**Exercise 3 : Stored Procedures**

**Scenario 1:**

The bank needs to process monthly interest for all savings accounts.

**Question:** Write a stored procedure ProcessMonthlyInterest that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

**Code:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR acc\_rec IN(

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType='Savings'

)LOOP

UPDATE Accounts

SET Balance=Balance+(Balance \* 0.01),

LastModified=SYSDATE

WHERE AccountID=acc\_rec.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Interest applied to Account ID: ' || acc\_rec.AccountID);

END LOOP;

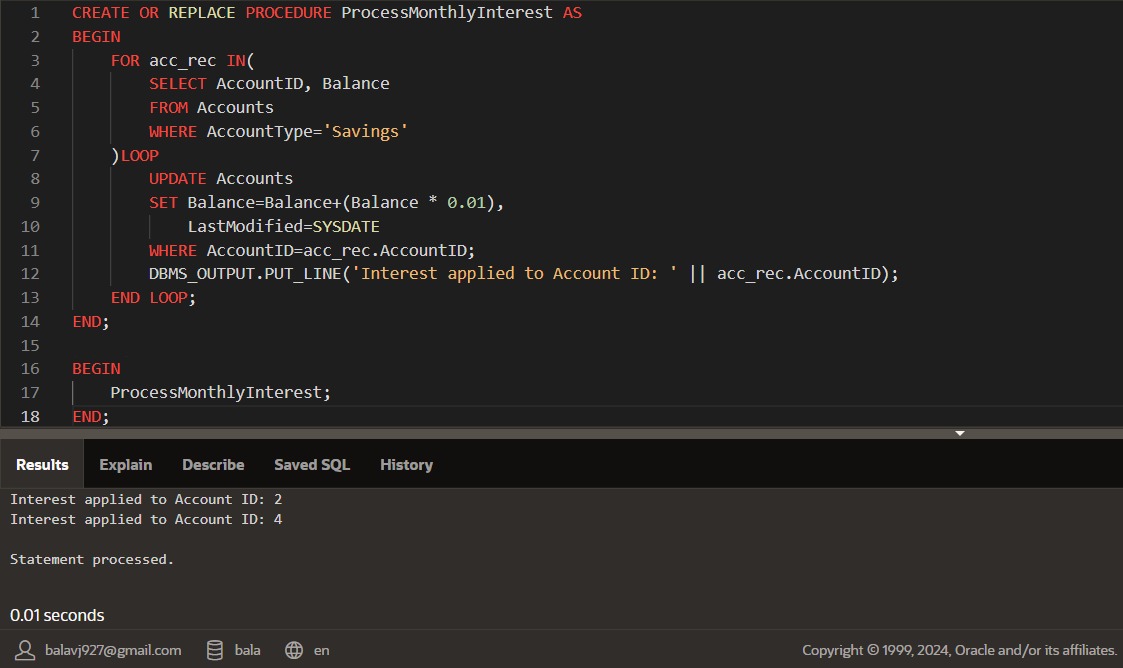
END;

BEGIN

ProcessMonthlyInterest;

END;

**Output:**



**Scenario 2:**

The bank wants to implement a bonus scheme for employees based on their performance.

**Question:** Write a stored procedure UpdateEmployeeBonus that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

**Code:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_department IN VARCHAR2,

p\_bonus\_percent IN NUMBER

)IS

BEGIN

FOR emp IN(

SELECT EmployeeID, Name, Salary

FROM Employees

WHERE Department=p\_department

)LOOP

UPDATE Employees

SET Salary=Salary+(emp.Salary\*p\_bonus\_percent / 100)

WHERE EmployeeID=emp.EmployeeID;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied to Employee ID: ' || emp.EmployeeID || ', Name: ' || emp.Name);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || p\_bonus\_percent || '% applied to department: ' || p\_department);

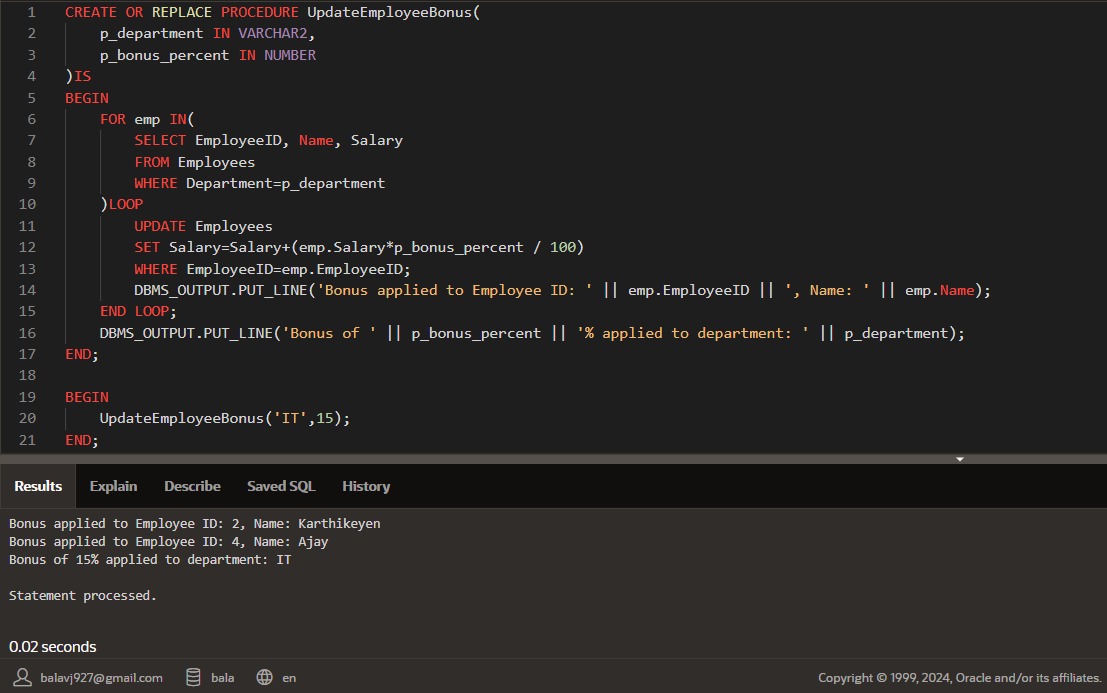
END;

BEGIN

UpdateEmployeeBonus('IT',15);

END;

**Output:**



**Scenario 3:**

Customers should be able to transfer funds between their accounts

**Question :** Write a stored procedure TransferFunds that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

**Code:**

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

)AS v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID=p\_from\_account

FOR UPDATE;

IF v\_balance < p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds in source account.');

END IF;

UPDATE Accounts

SET Balance=Balance-p\_amount,

LastModified=SYSDATE

WHERE AccountID=p\_from\_account;

UPDATE Accounts

SET Balance=Balance+p\_amount,

LastModified=SYSDATE

WHERE AccountID=p\_to\_account;

DBMS\_OUTPUT.PUT\_LINE('Transferred $' || p\_amount || ' from Account ' || p\_from\_account || ' to Account ' || p\_to\_account);

END;

BEGIN

TransferFunds(2,3,500);

END;

**Output:**

