|  |  |
| --- | --- |
| **WEEK 2 – PLSQL** | **Superset Id : 6429486**  **Name : Akshaya V** |

**Exercise 1: Control Structures**

**Scenario 1:** 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

CURSOR cust\_cursor IS

SELECT c.CustomerID,c.Name,c.DOB,l.LoanID,l.InterestRate

FROM Customers c

JOIN Loans l ON c.CustomerID=l.CustomerID;

v\_age NUMBER;

BEGIN

FOR cust\_rec IN cust\_cursor LOOP

v\_age:=FLOOR(MONTHS\_BETWEEN(SYSDATE,cust\_rec.DOB)/12);

IF v\_age>60 THEN

UPDATE Loans

SET InterestRate=InterestRate-1

WHERE LoanID=cust\_rec.LoanID;

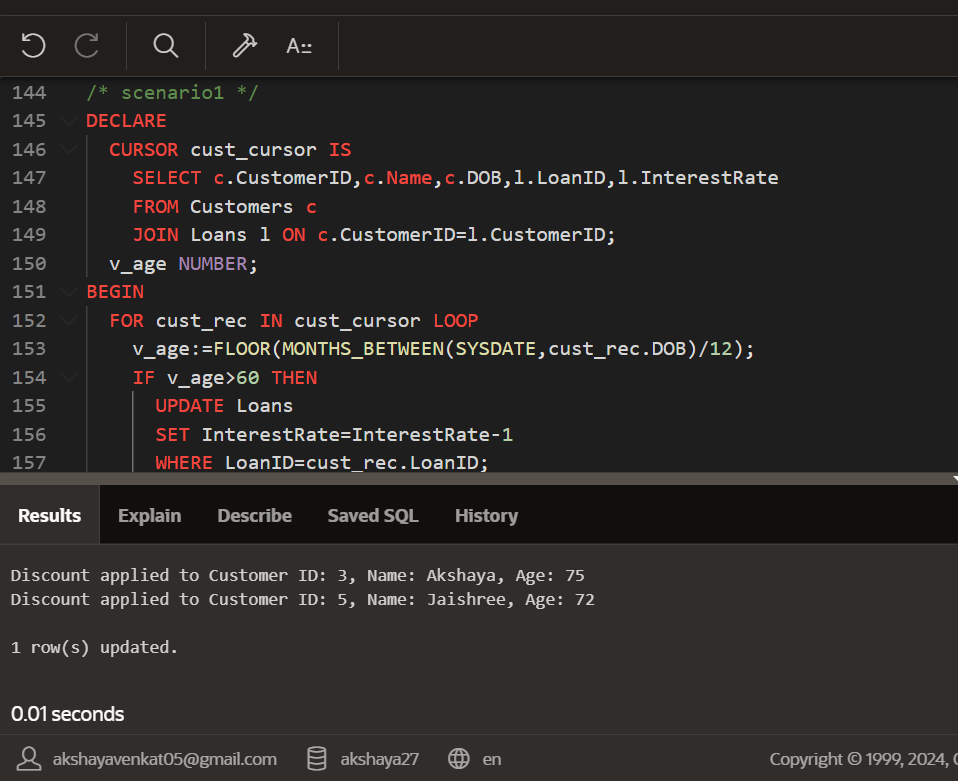
DBMS\_OUTPUT.PUT\_LINE('Discount applied to Customer ID: ' || cust\_rec.CustomerID ||', Name: ' || cust\_rec.Name ||', Age: ' || v\_age);

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 rec IN (SELECT CustomerID,Name, Balance FROM Customers) LOOP

IF rec.Balance>10000 THEN

UPDATE Customers

SET IsVIP='TRUE'

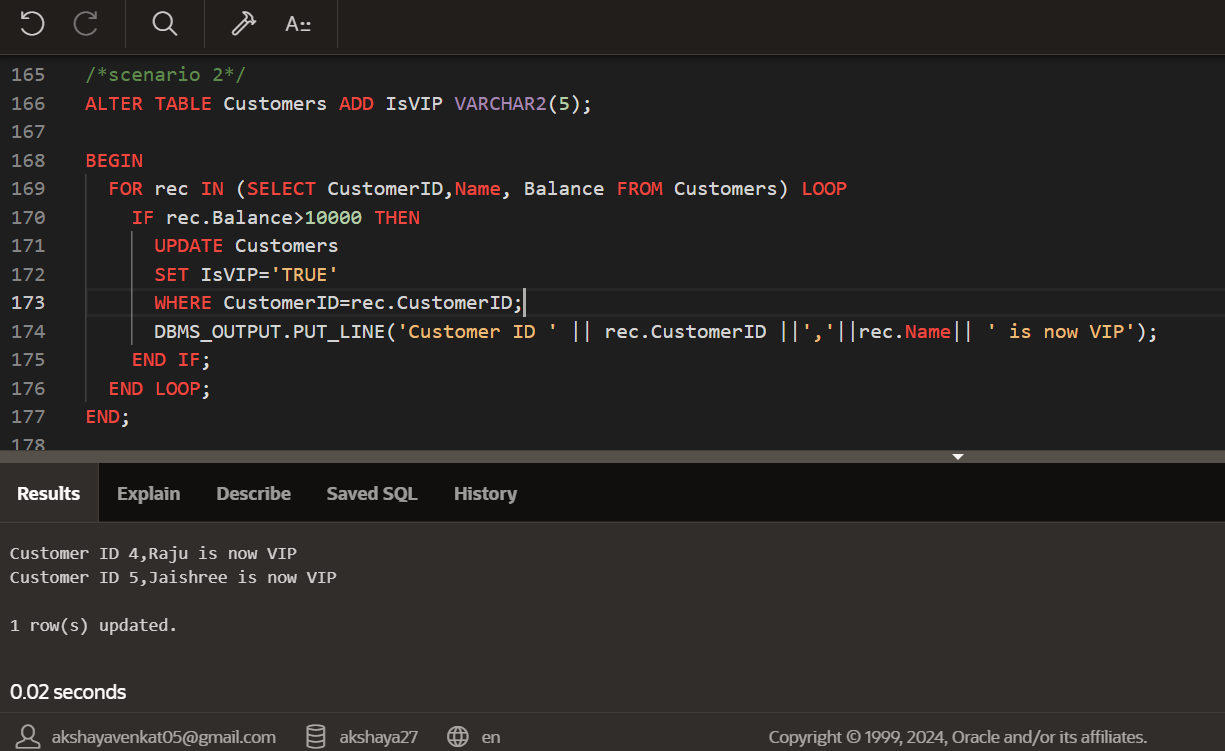
WHERE CustomerID=rec.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('Customer ID ' || rec.CustomerID ||','||rec.Name|| ' is now VIP');

END IF;

END LOOP;

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:**

BEGIN

FOR rec IN (

SELECT l.LoanID, c.Name, l.EndDate FROM Loans l

JOIN Customers c ON l.CustomerID=c.CustomerID

WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE+30

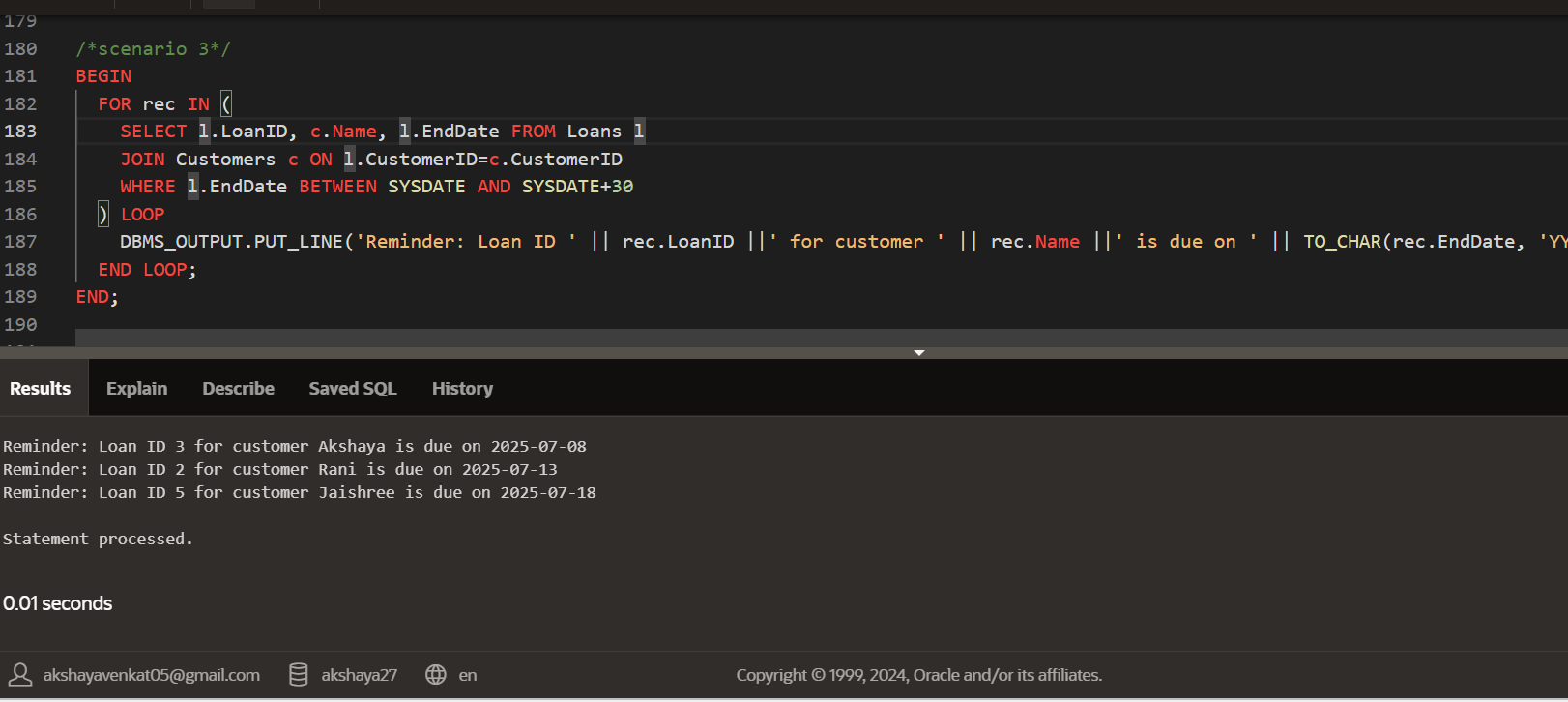
) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || rec.LoanID ||' for customer ' || rec.Name ||' is due on ' || TO\_CHAR(rec.EndDate, 'YYYY-MM-DD'));

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 IS

BEGIN

FOR acc IN(

SELECT AccountID, Balance FROM Accounts

WHERE AccountType='Savings'

) LOOP

UPDATE Accounts

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

LastModified=SYSDATE

WHERE AccountID=acc.AccountID;

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

END LOOP;

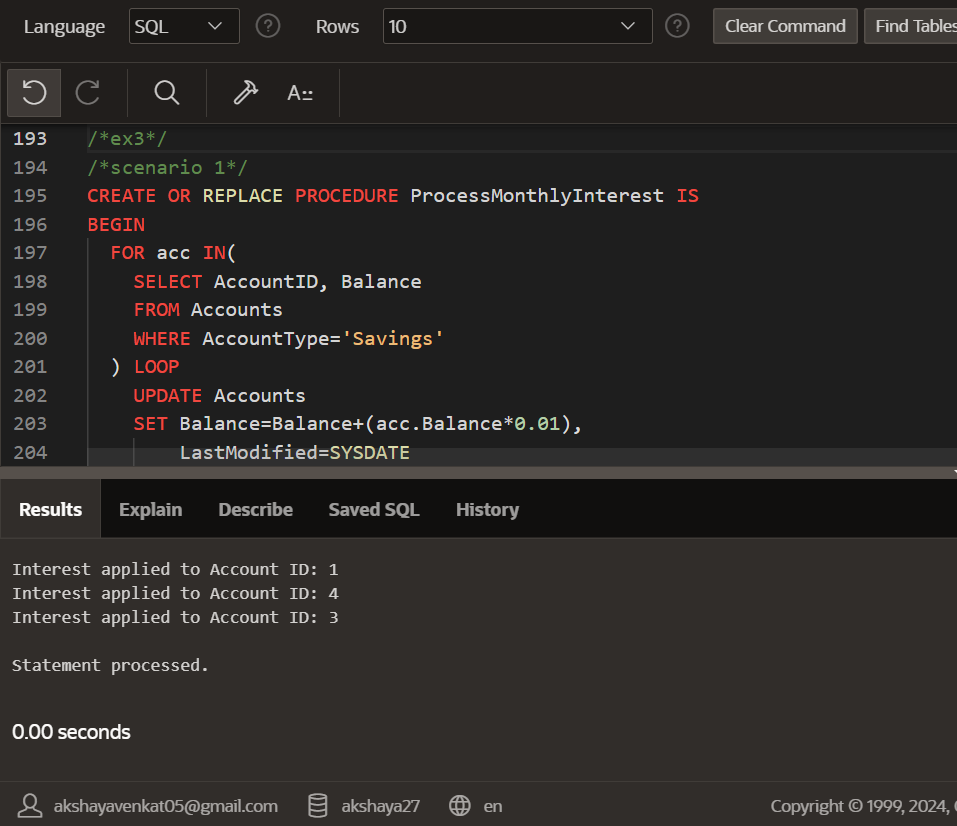
END;

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(

deptName IN VARCHAR2,bonusPercent IN NUMBER

) IS

BEGIN

FOR emp IN(

SELECT EmployeeID, Salary

FROM Employees

WHERE Department=deptName

) LOOP

UPDATE Employees

SET Salary=Salary+(emp.Salary\*bonusPercent/100)

WHERE EmployeeID=emp.EmployeeID;

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

END LOOP;

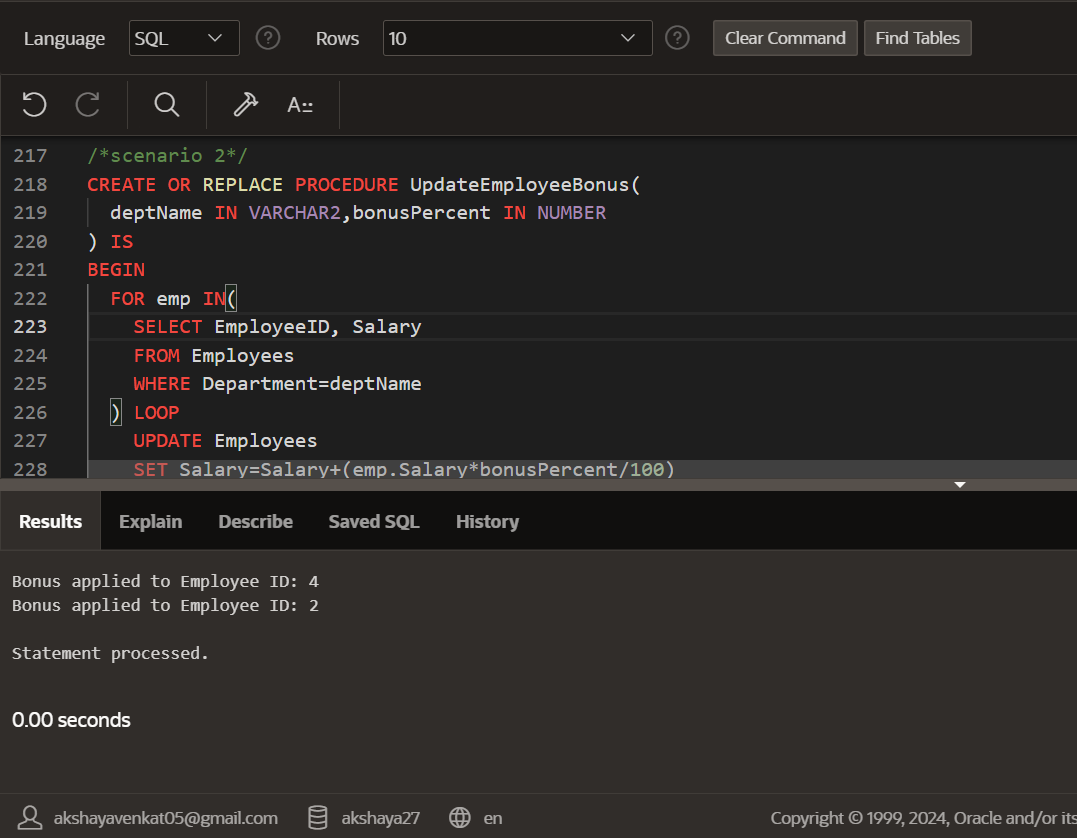
END;

BEGIN

UpdateEmployeeBonus('IT',10);

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(

fromAccountID IN NUMBER,toAccountID IN NUMBER,amount IN NUMBER

) IS

fromBalance NUMBER;

BEGIN

SELECT Balance INTO fromBalance FROM Accounts WHERE AccountID=fromAccountID FOR UPDATE;

IF fromBalance < amount THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance in Account ID: ' || fromAccountID);

ELSE

UPDATE Accounts

SET Balance=Balance-amount,

LastModified=SYSDATE

WHERE AccountID=fromAccountID;

UPDATE Accounts

SET Balance=Balance+amount,

LastModified=SYSDATE

WHERE AccountID=toAccountID;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful: ' || amount || ' transferred from Account ID ' || fromAccountID || ' to Account ID ' || toAccountID);

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: One of the Account IDs does not exist.');

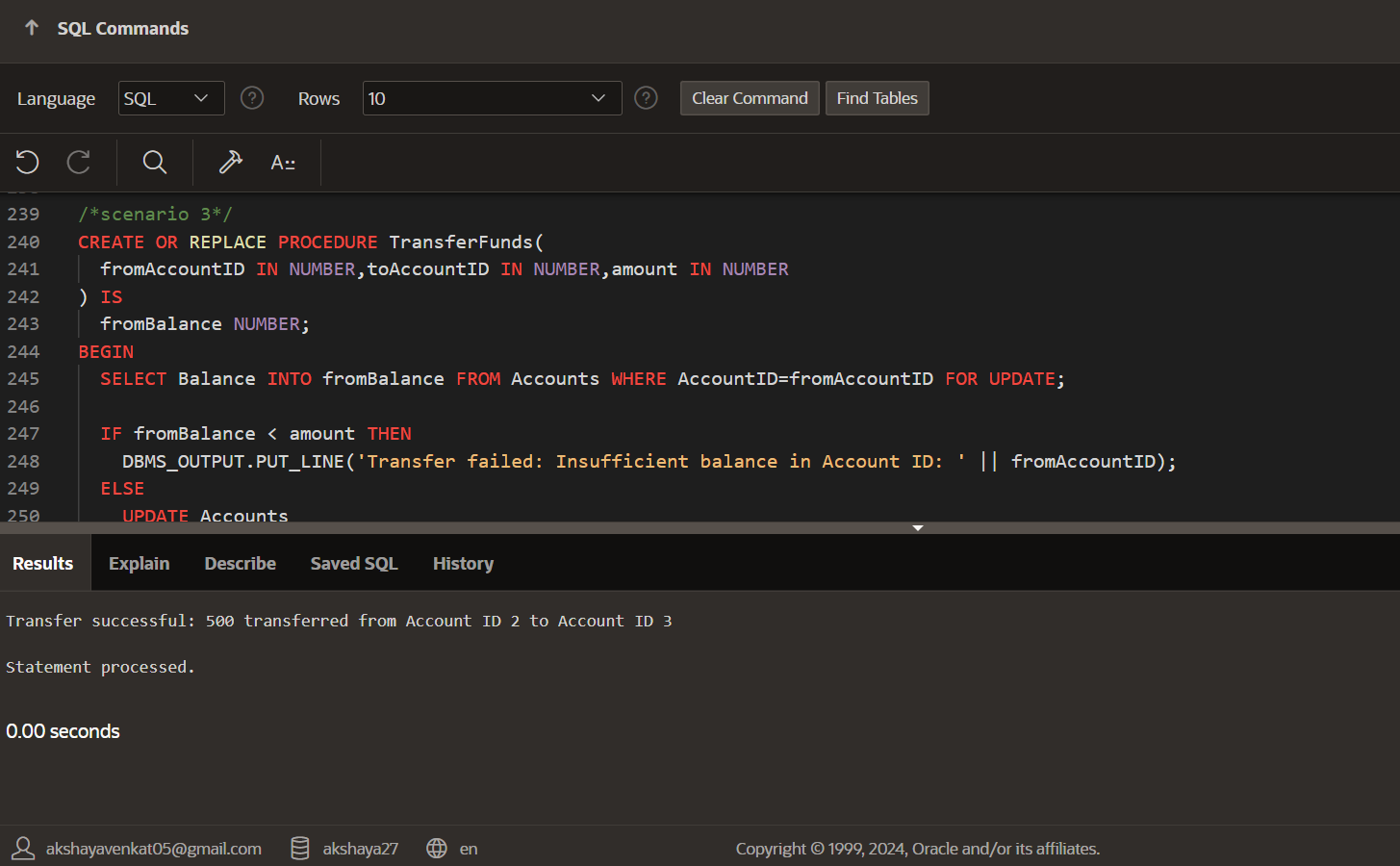
END;

BEGIN

TransferFunds(1,2,500);

END;

**Output:**

****