**Exercise 1: Control Structures**

**Scenario 1.** 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:**

BEGIN

FOR cust\_rec IN (

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

FROM Customers c

JOIN Loans l ON c.CustomerID = l.CustomerID

WHERE MONTHS\_BETWEEN(SYSDATE, c.DOB)/12 > 60

) LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE LoanID = cust\_rec.LoanID;

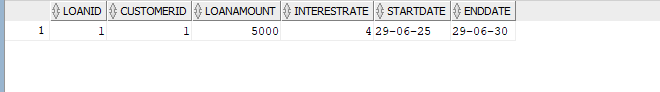
END LOOP;

COMMIT;

END;

**Output:**





**Scenario 2.** 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) DEFAULT 'FALSE';

BEGIN

FOR cust\_rec IN (

SELECT c.CustomerID

FROM Customers c

JOIN Accounts a ON c.CustomerID = a.CustomerID

GROUP BY c.CustomerID

HAVING SUM(a.Balance) > 10000

) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust\_rec.CustomerID;

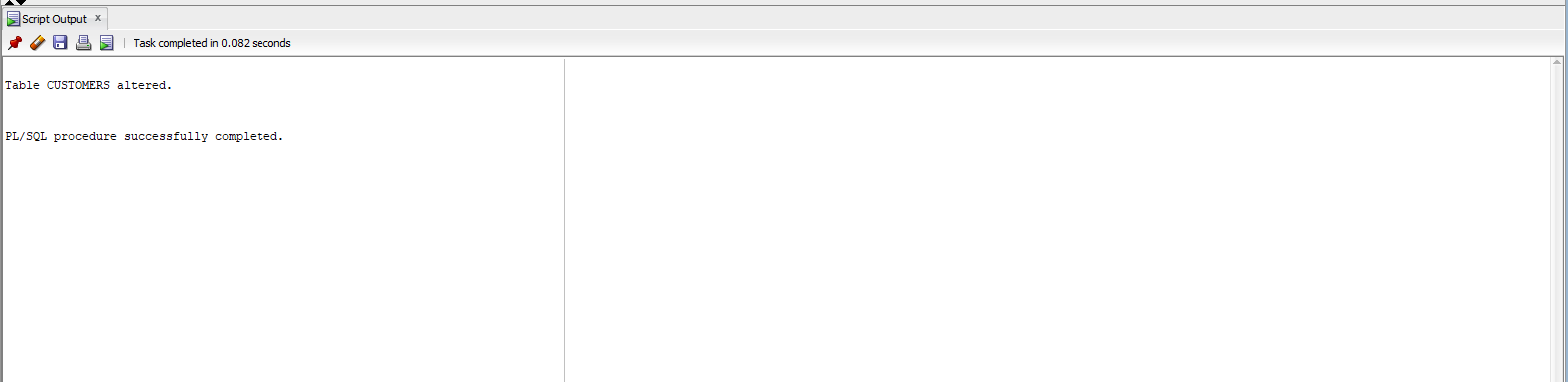
END LOOP;

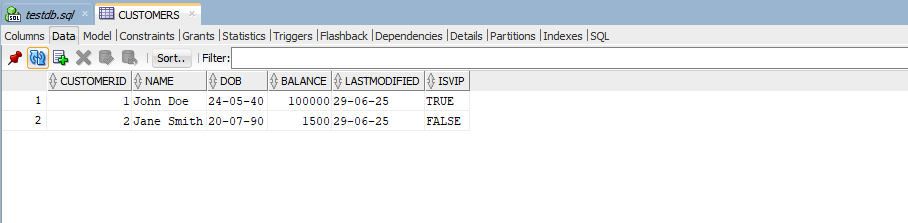
COMMIT;

END;

/

**Output:**





**Scenario 3.** Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

**Code:**

SET SERVEROUTPUT ON;

BEGIN

FOR loan\_rec IN (

SELECT l.LoanID,

c.Name AS CustomerName,

l.EndDate

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate <= SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE(' Reminder: Loan ' || loan\_rec.LoanID ||

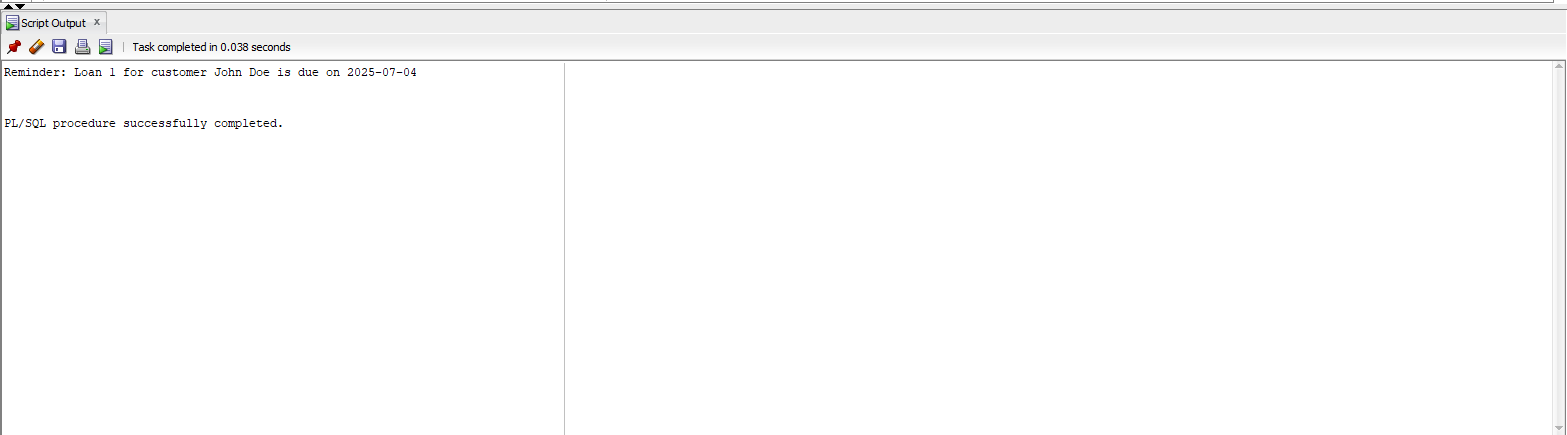
' for customer ' || loan\_rec.CustomerName ||

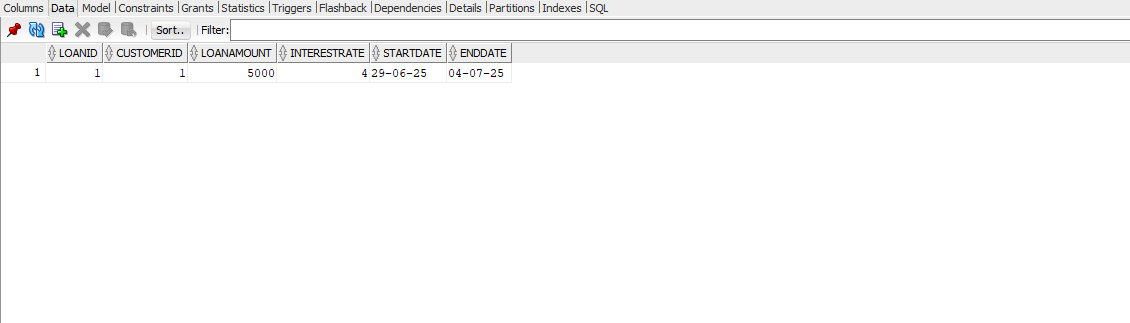
' is due on ' || TO\_CHAR(loan\_rec.EndDate, 'YYYY-MM-DD'));

END LOOP;

END;

**Output:**





**Exercise 3: Stored Procedures**

**Scenario 1.** 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:**

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

    UPDATE Accounts

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

        LastModified = SYSDATE

    WHERE AccountType = 'Savings';

    COMMIT;

END;

/

BEGIN

    ProcessMonthlyInterest;

END;

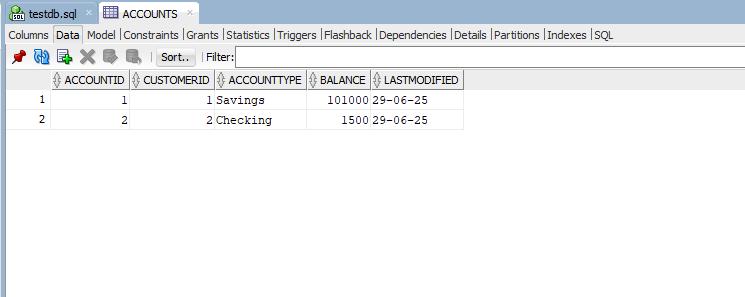
/

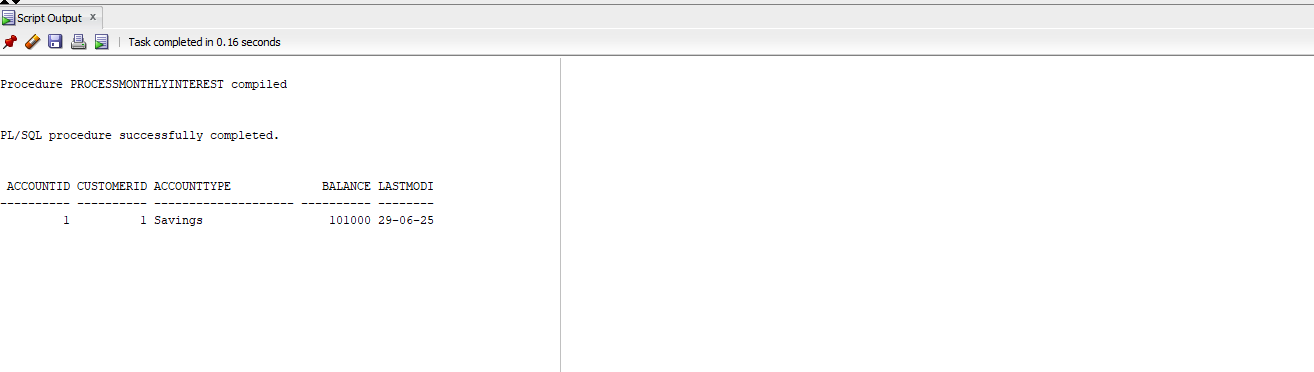
SELECT AccountID, CustomerID, AccountType, Balance, LastModified

FROM Accounts

WHERE AccountType = 'Savings';

**Output:**





**Scenario 2.** 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:**SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

    p\_dept IN VARCHAR2,

    p\_bonus\_pct IN NUMBER

) IS

BEGIN

    UPDATE Employees

    SET Salary = Salary + (Salary \* p\_bonus\_pct / 100)

    WHERE Department = p\_dept;

    COMMIT;

END;

/

BEGIN

    UpdateEmployeeBonus('IT', 10);

END;

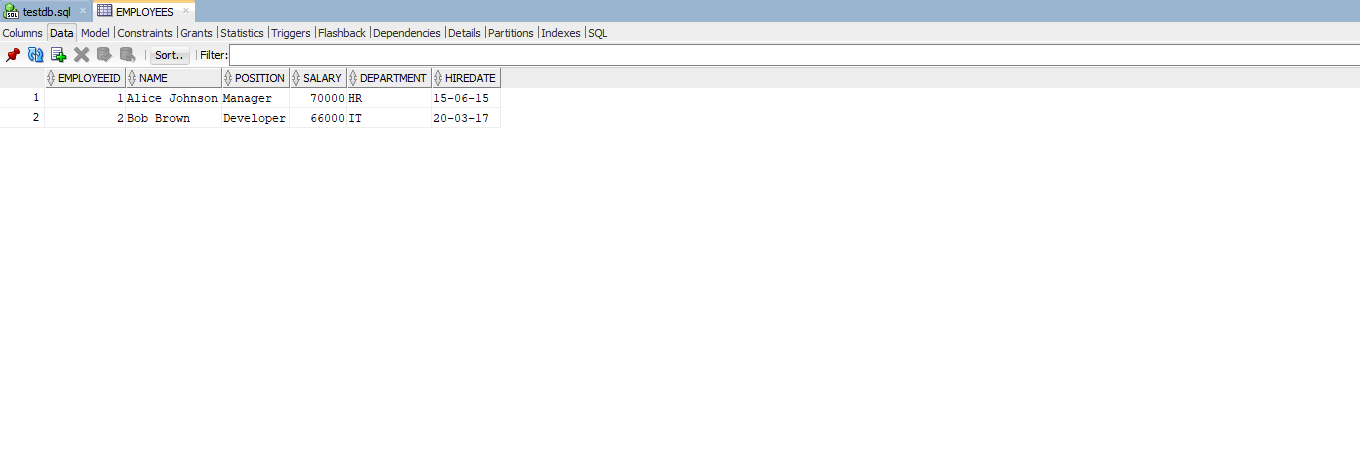
/

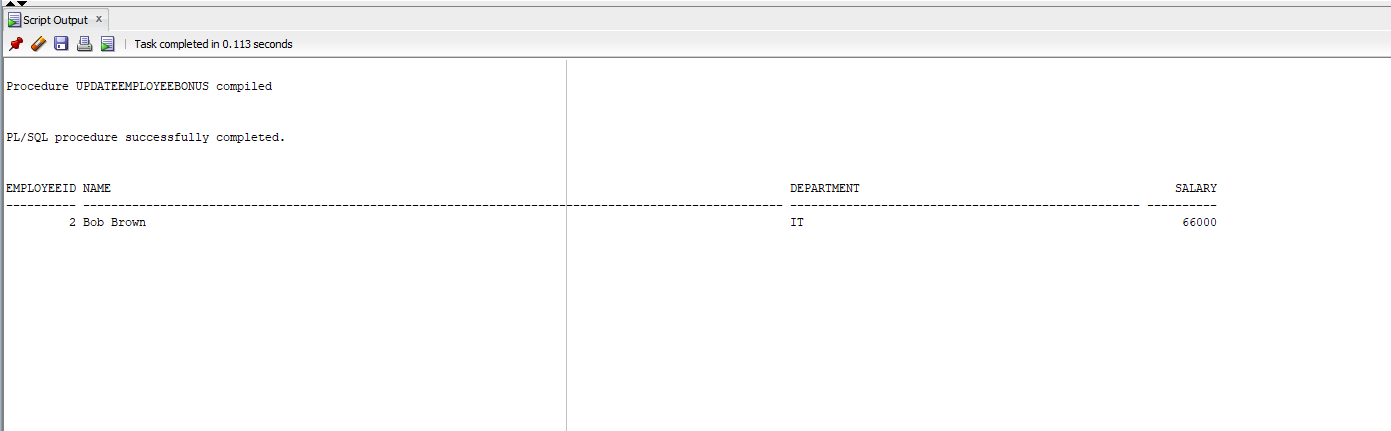
SELECT EmployeeID, Name, Department, Salary

FROM Employees

WHERE Department = 'IT';

**Output:**





**Scenario 3.** 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:**

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE TransferFunds (

    p\_from\_account IN NUMBER,

    p\_to\_account IN NUMBER,

    p\_amount IN NUMBER

) IS

    v\_balance NUMBER;

BEGIN

    SELECT Balance INTO v\_balance

    FROM Accounts

    WHERE AccountID = p\_from\_account;

    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;

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('Transfer successful.');

END;

/

BEGIN

    TransferFunds(1, 2, 500);

END;

/

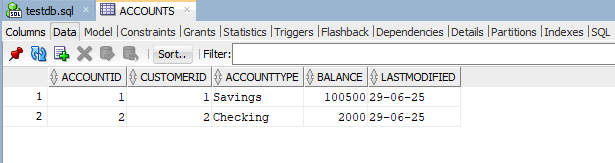
SELECT AccountID, CustomerID, AccountType, Balance

FROM Accounts

WHERE AccountID IN (1, 2);

**Output:**

**Before Transfer:**



**After Transfer:**

