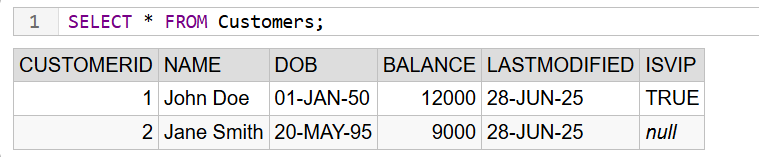
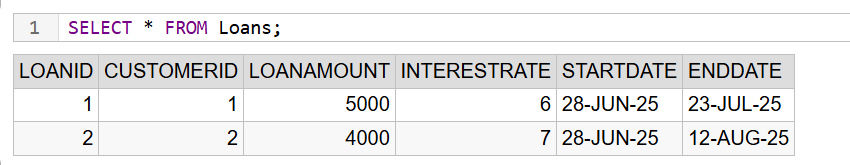
**Exercise 1: Control Structures**





**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.

BEGIN

FOR cust IN ( SELECT CustomerID, Balance FROM Customers

) LOOP

IF cust.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust.CustomerID;

END IF;

END LOOP;

COMMIT;

END;

/

**RESULT:**

1 rows affected

**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.

BEGIN

FOR cust IN (SELECT CustomerID, Balance FROM Customers

) LOOP

IF cust.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust.CustomerID;

END IF;

END LOOP;

COMMIT;

END;

/

**RESULT:**

1 rows affected

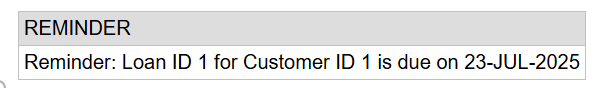
**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

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

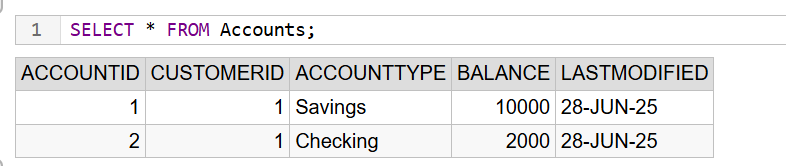
SELECT

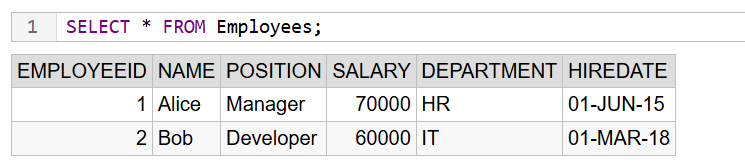
'Reminder: Loan ID ' || LoanID || ' for Customer ID ' || CustomerID || ' is due on ' || TO\_CHAR(EndDate, 'DD-MON-YYYY') AS Reminde FROM Loans WHERE EndDate <= SYSDATE + 30;

**RESULT:**



**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.

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountType = 'Savings';

COMMIT;

END;

/

**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.

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_Department IN VARCHAR2,

p\_BonusPercent IN NUMBER

) IS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* p\_BonusPercent / 100)

WHERE Department = p\_Department;

COMMIT;

END;

/

**Adds 10% bonus to IT dept salaries:**

BEGIN

UpdateEmployeeBonus('IT', 10);

END;

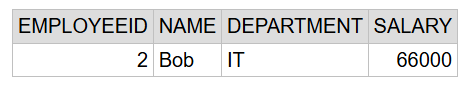
/

**Result:**

1 rows affected

**Check salaries again:**

SELECT EmployeeID, Name, Department, Salary FROM Employees WHERE Department = 'IT';



**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.

CREATE OR REPLACE PROCEDURE TransferFunds(p\_FromAccountID IN NUMBER, p\_ToAccountID IN NUMBER,p\_Amount IN NUMBER) IS v\_FromBalance NUMBER;

BEGIN

SELECT Balance INTO v\_FromBalance

FROM Accounts

WHERE AccountID = p\_FromAccountID;

IF v\_FromBalance < p\_Amount THEN

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

END IF;

UPDATE Accounts SET Balance = Balance - p\_Amount WHERE AccountID = p\_FromAccountID;

UPDATE Accounts SET Balance = Balance + p\_Amount WHERE AccountID = p\_ToAccountID;

COMMIT;

END;

/

**Transfer ₹100 from Account 1 to 2:**

BEGIN

TransferFunds(1, 2, 100);

END;

/

**Result:**

1 rows affected

**Check Balances After Transfer:**

SELECT AccountID, Balance FROM Accounts WHERE AccountID IN (1, 2);

