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.

QUERY:

```
DECLARE
CURSOR cust_cursor IS
SELECT c.CustomerID, l.LoanID, l.InterestRate
FROM Customers c
JOIN Loans l ON c.CustomerID = l.CustomerID
WHERE TRUNC(MONTHS_BETWEEN(SYSDATE, c.DOB) / 12) > 60;

BEGIN
FOR cust_rec IN cust_cursor LOOP
UPDATE Loans
SET InterestRate = cust_rec.InterestRate - 1
WHERE LoanID = cust_rec.LoanID;
END LOOP;
COMMIT;
END;
/
```

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.

QUERY:

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

DECLARE
  CURSOR vip_cursor IS
    SELECT CustomerID, Balance FROM Customers WHERE Balance > 10000;

BEGIN
  FOR vip_rec IN vip_cursor LOOP
    UPDATE Customers
    SET IsVIP = 'TRUE'
    WHERE CustomerID = vip_rec.CustomerID;
  END LOOP;

  COMMIT;
END;
/
```

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

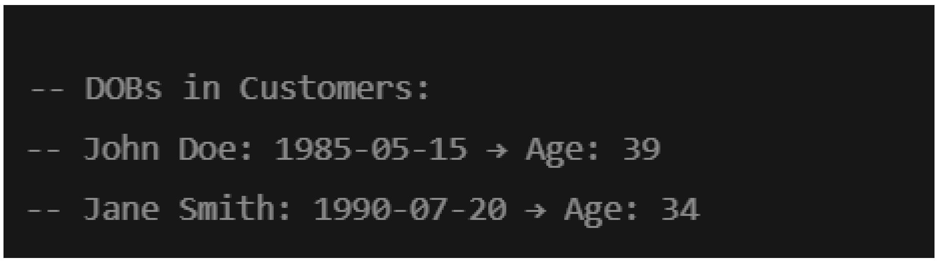
- **Question:** Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

ANS-

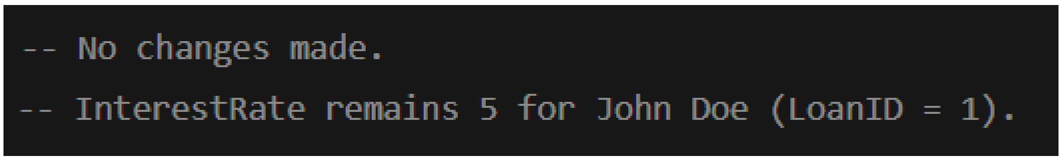
```
DECLARE
  CURSOR loan_cursor IS
    SELECT l.LoanID, l.CustomerID, c.Name, l.EndDate
    FROM Loans l
    JOIN Customers c ON c.CustomerID = l.CustomerID
    WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30;

BEGIN
  FOR rec IN loan_cursor LOOP
    DBMS_OUTPUT.PUT_LINE('Reminder: Loan ID ' || rec.LoanID ||
                         ' for Customer ' || rec.Name ||
                         ' is due on ' || TO_CHAR(rec.EndDate, 'DD-MON-YYYY'));
  END LOOP;
END;
/
```

OUTPUT:

SC1-

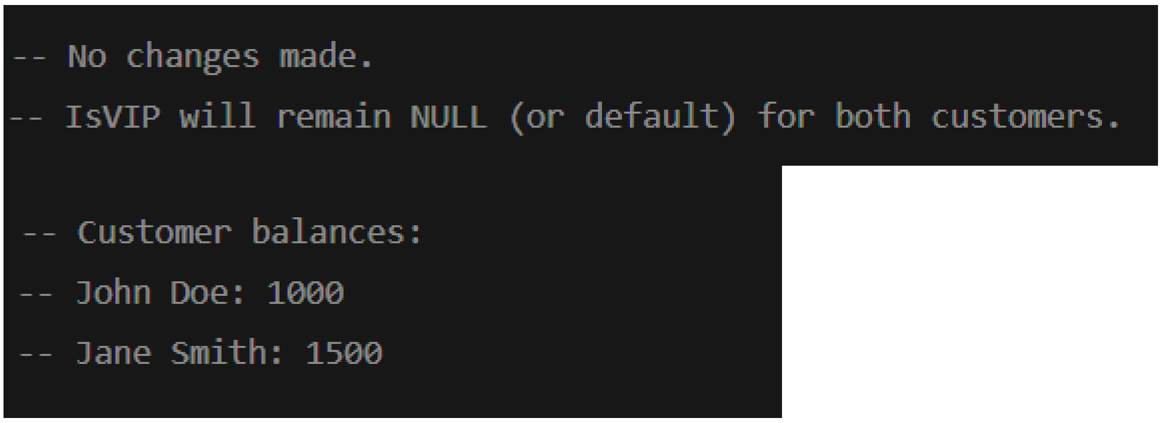


```
-- DOBs in Customers:
-- John Doe: 1985-05-15 → Age: 39
-- Jane Smith: 1990-07-20 → Age: 34
```



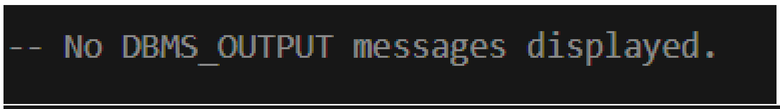
```
-- No changes made.
-- InterestRate remains 5 for John Doe (LoanID = 1).
```

SC2-

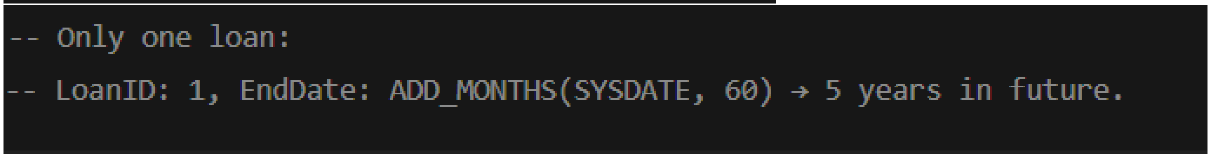


```
-- No changes made.
-- IsVIP will remain NULL (or default) for both customers.

-- Customer balances:
-- John Doe: 1000
-- Jane Smith: 1500
```

SC3-



```
-- No DBMS_OUTPUT messages displayed.
```



```
-- Only one loan:
-- LoanID: 1, EndDate: ADD_MONTHS(SYSDATE, 60) → 5 years in future.
```

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.

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.

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.

Ans->

```
Sc1- CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS
BEGIN
  UPDATE Accounts
  SET Balance = Balance + (Balance * 0.01)
  WHERE AccountType = 'Savings';
 COMMIT;
END;
/
```

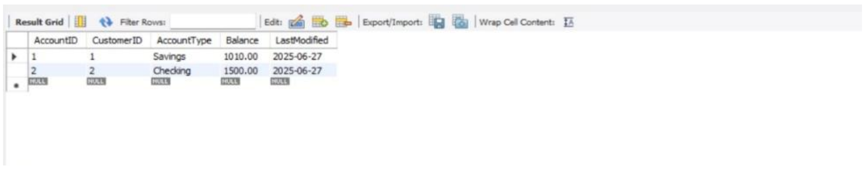
Sc2-

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
  p_Department IN VARCHAR2,
  p_BonusPercent IN NUMBER
) AS
BEGIN
  UPDATE Employees
  SET Salary = Salary + (Salary * p_BonusPercent / 100)
  WHERE Department = p_Department;
 COMMIT;
END;
/
```

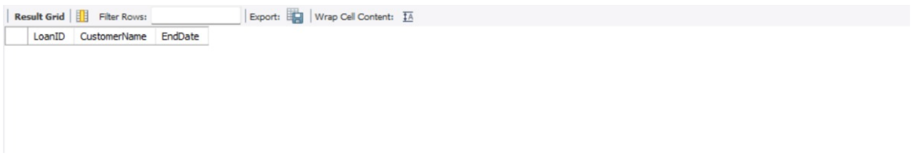
Sc3-

```
CREATE OR REPLACE PROCEDURE TransferFunds (
  p_FromAccountID IN NUMBER,
  p_ToAccountID IN NUMBER,
  p_Amount IN NUMBER
) AS
  v_FromBalance NUMBER;
BEGIN
  SELECT Balance INTO v_FromBalance
  FROM Accounts
  WHERE AccountID = p_FromAccountID
  FOR UPDATE;

  IF v_FromBalance < p_Amount THEN
    RAISE_APPLICATION_ERROR(-20001, 'Insufficient balance for transfer.');
  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;
/
```

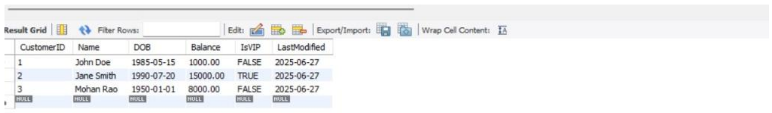
Output-



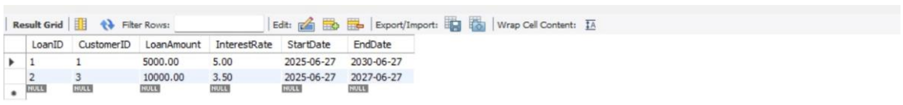
AccountID	CustomerID	AccountType	Balance	LastModified
1	1	Savings	1010.00	2025-06-27
2	2	Checking	1500.00	2025-06-27
NULL	NULL	NULL	NULL	NULL



LoanID	CustomerName	EndDate



CustomerID	Name	DOB	Balance	IsVIP	LastModified
1	John Doe	1985-05-15	1000.00	FALSE	2025-06-27
2	Jane Smith	1990-07-20	15000.00	TRUE	2025-06-27
3	Mohan Rao	1950-01-01	8000.00	FALSE	2025-06-27
NULL	NULL	NULL	NULL	NULL	NULL



LoanID	CustomerID	LoanAmount	InterestRate	StartDate	EndDate
1	1	5000.00	5.00	2025-06-27	2030-06-27
2	3	10000.00	3.50	2025-06-27	2027-06-27
NULL	NULL	NULL	NULL	NULL	NULL