**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.

**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.

**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.

**1.Create Table :**

CREATE TABLE customers (

    customer\_id        NUMBER PRIMARY KEY,

    name               VARCHAR2(100),

    birth\_date         DATE,

    balance            NUMBER(10,2),

    is\_vip             CHAR(1), -- 'Y' or NULL

    loan\_interest\_rate NUMBER(5,2)

);

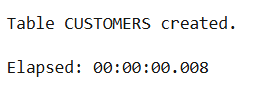
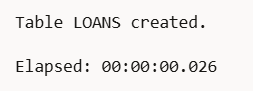
CREATE TABLE loans (

    loan\_id     NUMBER PRIMARY KEY,

    customer\_id NUMBER REFERENCES customers(customer\_id),

    due\_date    DATE

);

**2.Insert Sample Data:**

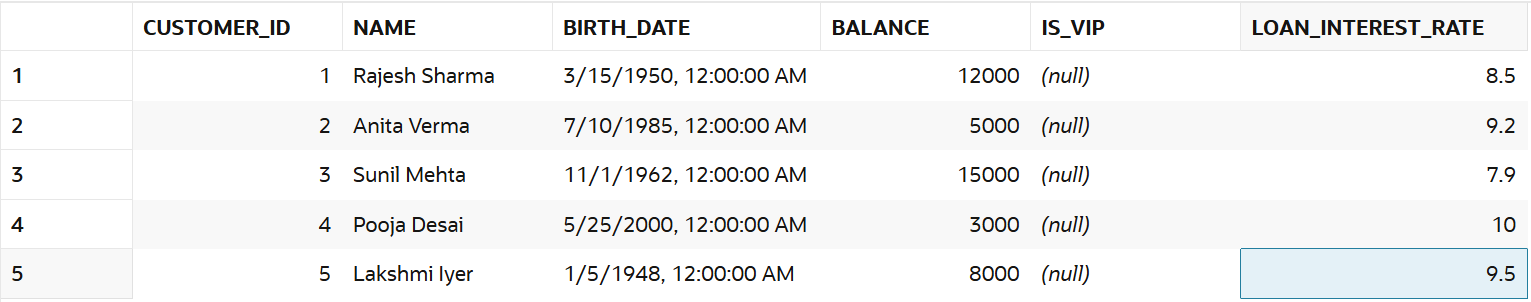
INSERT INTO customers VALUES (1, 'Rajesh Sharma', TO\_DATE('1950-03-15','YYYY-MM-DD'), 12000.00, NULL, 8.5);

INSERT INTO customers VALUES (2, 'Anita Verma', TO\_DATE('1985-07-10','YYYY-MM-DD'), 5000.00, NULL, 9.2);

INSERT INTO customers VALUES (3, 'Sunil Mehta', TO\_DATE('1962-11-01','YYYY-MM-DD'), 15000.00, NULL, 7.9);

INSERT INTO customers VALUES (4, 'Pooja Desai', TO\_DATE('2000-05-25','YYYY-MM-DD'), 3000.00, NULL, 10.0);

INSERT INTO customers VALUES (5, 'Lakshmi Iyer', TO\_DATE('1948-01-05','YYYY-MM-DD'), 8000.00, NULL, 9.5);



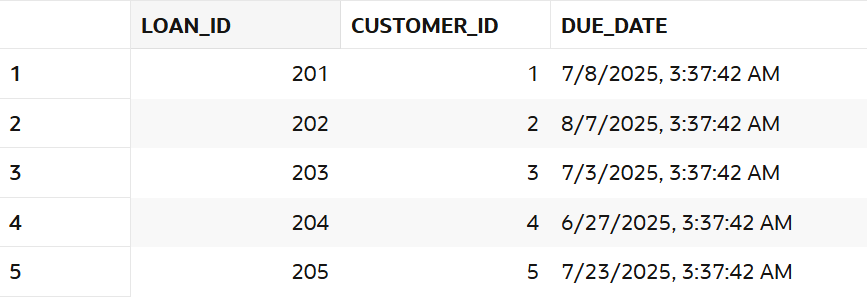
INSERT INTO loans VALUES (201, 1, SYSDATE + 10);

INSERT INTO loans VALUES (202, 2, SYSDATE + 40);

INSERT INTO loans VALUES (203, 3, SYSDATE + 5);

INSERT INTO loans VALUES (204, 4, SYSDATE - 1);

INSERT INTO loans VALUES (205, 5, SYSDATE + 25);



**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 customer\_id, birth\_date

FROM customers

) LOOP

IF TRUNC(MONTHS\_BETWEEN(SYSDATE, cust.birth\_date) / 12) > 60 THEN

UPDATE customers

SET loan\_interest\_rate = loan\_interest\_rate - 1

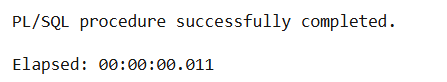
WHERE customer\_id = cust.customer\_id;

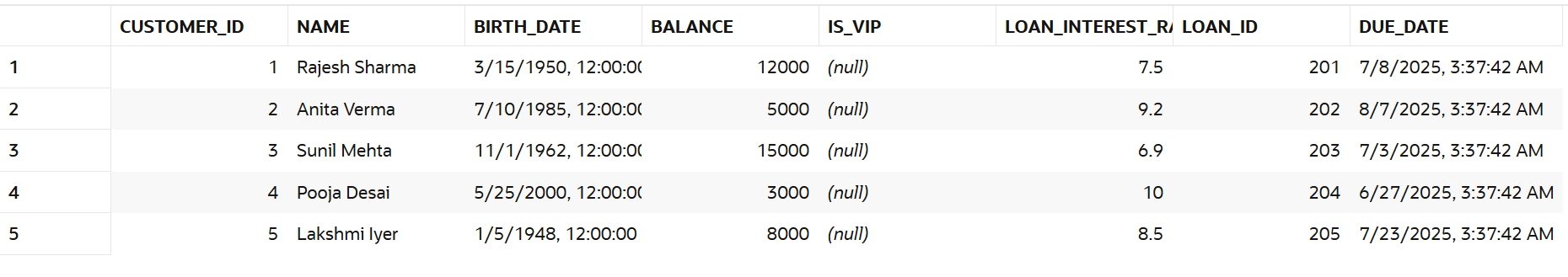
END IF;

END LOOP;

END;

/





**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 customer\_id, balance

FROM customers

) LOOP

IF cust.balance > 10000 THEN

UPDATE customers

SET is\_vip = 'Y'

WHERE customer\_id = cust.customer\_id;

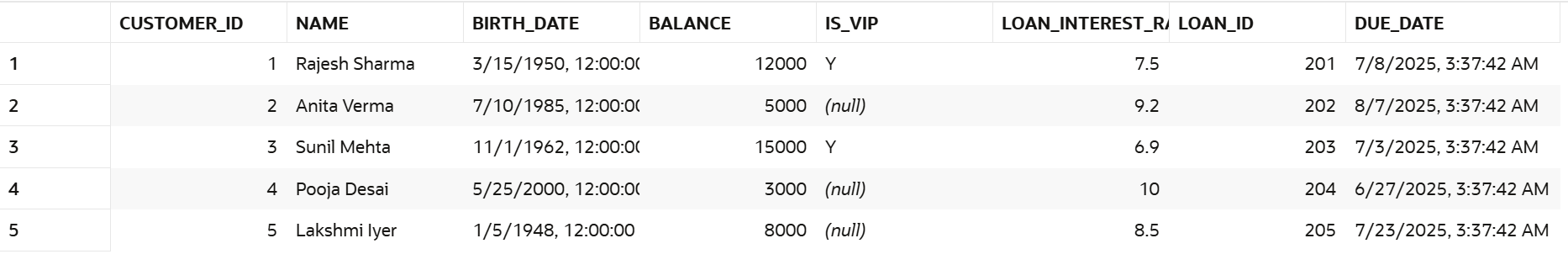
END IF;

END LOOP;

COMMIT;

END;

/



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

BEGIN

FOR loan\_rec IN (

SELECT l.loan\_id, l.due\_date, c.name

FROM loans l

JOIN customers c ON l.customer\_id = c.customer\_id

WHERE l.due\_date BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Reminder: ' || loan\_rec.name ||

', your loan (ID ' || loan\_rec.loan\_id ||

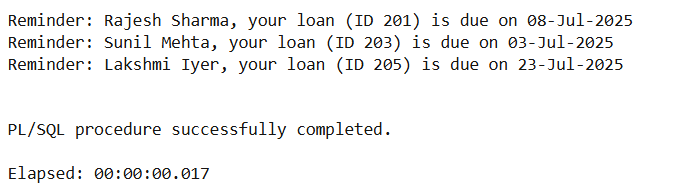
') is due on ' || TO\_CHAR(loan\_rec.due\_date, 'DD-Mon-YYYY')

);

END LOOP;

END;

/



**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.

**1.Create Table**  
CREATE TABLE accounts (

    account\_id     NUMBER PRIMARY KEY,

    customer\_id    NUMBER,

    account\_type   VARCHAR2(20), -- e.g., 'savings', 'current'

    balance        NUMBER(10,2)

);

CREATE TABLE employees (

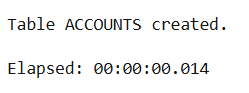
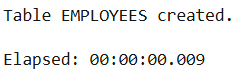
    emp\_id     NUMBER PRIMARY KEY,

    name       VARCHAR2(100),

    department VARCHAR2(50),

    salary     NUMBER(10,2)

);

**2.Add sample data to the table**

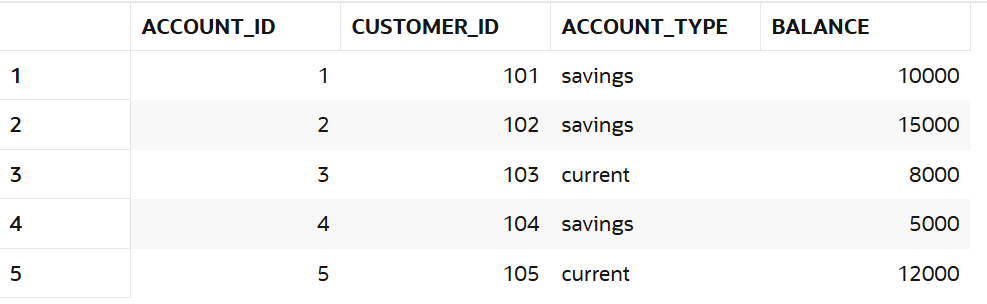
INSERT INTO accounts VALUES (1, 101, 'savings', 10000);

INSERT INTO accounts VALUES (2, 102, 'savings', 15000);

INSERT INTO accounts VALUES (3, 103, 'current', 8000);

INSERT INTO accounts VALUES (4, 104, 'savings', 5000);

INSERT INTO accounts VALUES (5, 105, 'current', 12000);

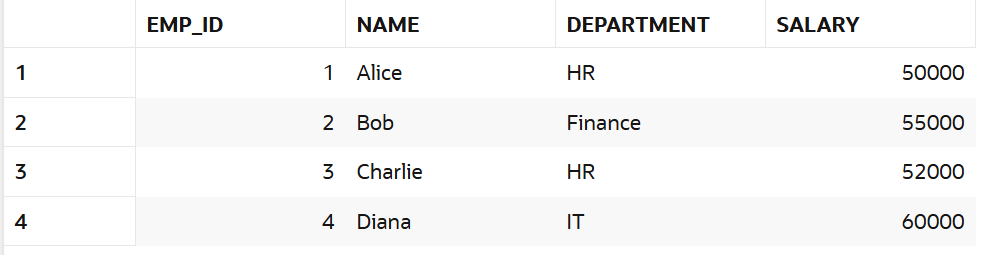


INSERT INTO employees VALUES (1, 'Alice', 'HR', 50000);

INSERT INTO employees VALUES (2, 'Bob', 'Finance', 55000);

INSERT INTO employees VALUES (3, 'Charlie', 'HR', 52000);

INSERT INTO employees VALUES (4, 'Diana', 'IT', 60000);



**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

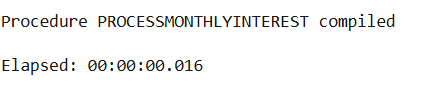
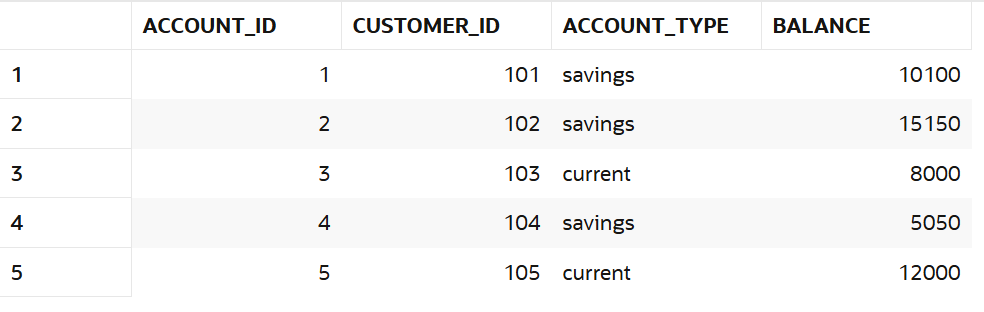
SET balance = balance \* 1.01

WHERE LOWER(account\_type) = 'savings';

COMMIT;

END;

/

**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 (

dept\_name IN VARCHAR2,

bonus\_pct IN NUMBER

) IS

BEGIN

UPDATE employees

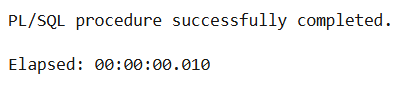
SET salary = salary + (salary \* bonus\_pct / 100)

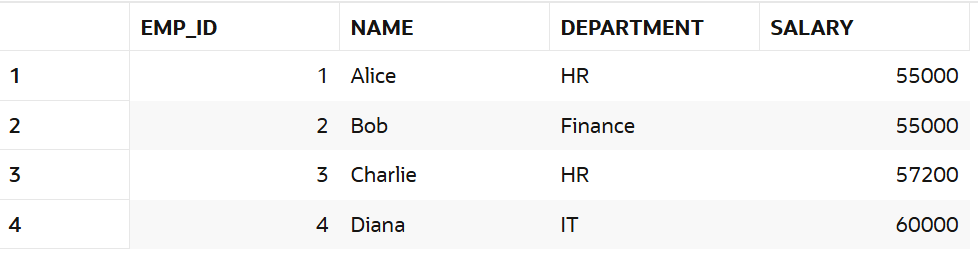
WHERE department = dept\_name;

COMMIT;

END;

/





**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 (

    from\_account IN NUMBER,

    to\_account   IN NUMBER,

    amount       IN NUMBER

) IS

    v\_balance NUMBER;

    insufficient\_balance EXCEPTION;

BEGIN

    -- Lock and get balance from source

    SELECT balance INTO v\_balance FROM accounts WHERE account\_id = from\_account FOR UPDATE;

    IF v\_balance < amount THEN

        RAISE insufficient\_balance;

    END IF;

    -- Deduct from source

    UPDATE accounts

    SET balance = balance - amount

    WHERE account\_id = from\_account;

    -- Add to destination

    UPDATE accounts

    SET balance = balance + amount

    WHERE account\_id = to\_account;

    COMMIT;

EXCEPTION

    WHEN insufficient\_balance THEN

        ROLLBACK;

        DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance.');

    WHEN NO\_DATA\_FOUND THEN

        ROLLBACK;

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

    WHEN OTHERS THEN

        ROLLBACK;

        DBMS\_OUTPUT.PUT\_LINE('Transfer failed: ' || SQLERRM);

END;

/

