**Exercise 1: Control Structures**

**SCENARIO 1**: Loan Interest Discount for Senior Citizens

**CODE:**

BEGIN

  FOR cust IN (SELECT customer\_id, age FROM customers) LOOP

    IF cust.age > 60 THEN

      UPDATE loans

      SET interest\_rate = interest\_rate \* 0.99

      WHERE customer\_id = cust.customer\_id;

    END IF;

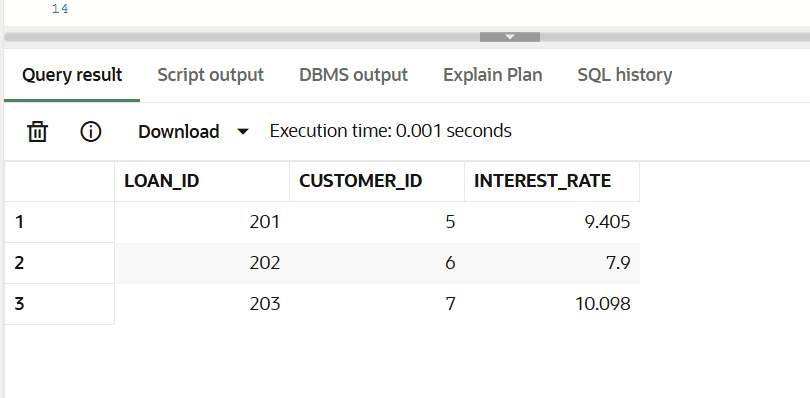
  END LOOP;

END;

/

SELECT \* FROM loans;

**OUTPUT:**

****

**Scenario 2:** A customer can be promoted to VIP status based on their balance.

**CODE:**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE customers';

EXCEPTION

WHEN OTHERS THEN

IF SQLCODE != -942 THEN

RAISE;

END IF;

END;

/

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(50),

age NUMBER,

balance NUMBER,

is\_vip CHAR(1) DEFAULT 'N'

);

BEGIN

INSERT INTO customers VALUES (1, 'John', 65, 9500, 'N');

INSERT INTO customers VALUES (2, 'Mary', 45, 15000, 'N');

INSERT INTO customers VALUES (3, 'Ravi', 70, 12000, 'N');

INSERT INTO customers VALUES (4, 'Anita', 30, 8000, 'N');

END;

/

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;

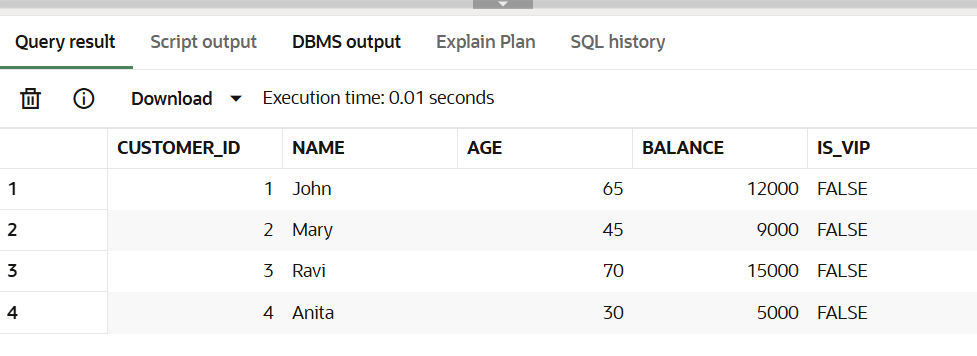
END LOOP;

END;

/

SELECT \* FROM customers;

**OUTPUT:**

****

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

**CODE:**

BEGIN

FOR rec IN (

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

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.loan\_id ||

' for ' || rec.name ||

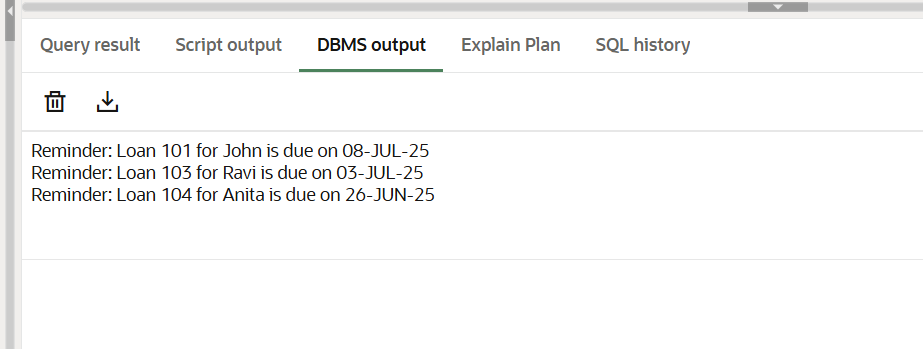
' is due on ' || TO\_CHAR(rec.due\_date, 'DD-MON-YY'));

END LOOP;

END;

/

**OUTPUT:**

****

**Exercise 3: Stored Procedures**

**Scenario 1:** The bank needs to process monthly interest for all savings accounts.

**CODE:**

**Step 1**: Create accounts table

CREATE TABLE accounts (

account\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

account\_type VARCHAR2(20),

balance NUMBER

);

**Step 2:** Insert sample data

BEGIN

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

INSERT INTO accounts VALUES (102, 2, 'Current', 8000);

INSERT INTO accounts VALUES (103, 3, 'Savings', 15000);

COMMIT;

END;

**Step 3:** Create stored procedure

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

UPDATE accounts

SET balance = balance \* 1.01

WHERE account\_type = 'Savings';

END;

/

**Step 4:** Execute procedure

BEGIN

ProcessMonthlyInterest;

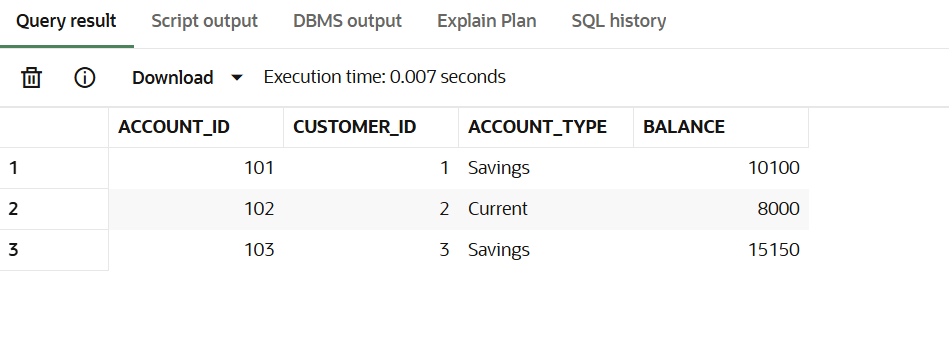
END;

/

**Step 5:** View result

SELECT \* FROM accounts;

**OUTPUT:**



**Scenario 2:** The bank wants to implement a bonus scheme for employees based on their performance.

**CODE:**

**Step 1:** Create employees table

CREATE TABLE employees (

emp\_id NUMBER PRIMARY KEY,

name VARCHAR2(50),

department VARCHAR2(50),

salary NUMBER

);

**Step 2:** Insert sample data

BEGIN

INSERT INTO employees VALUES (1, 'Alice', 'HR', 30000);

INSERT INTO employees VALUES (2, 'Bob', 'IT', 40000);

INSERT INTO employees VALUES (3, 'Charlie', 'IT', 45000);

COMMIT;

END;

/

**Step 3:** Create stored procedure

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE employees

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

WHERE department = dept\_name;

END;

/

**Step 4:** Execute procedure

BEGIN

UpdateEmployeeBonus('IT', 10);

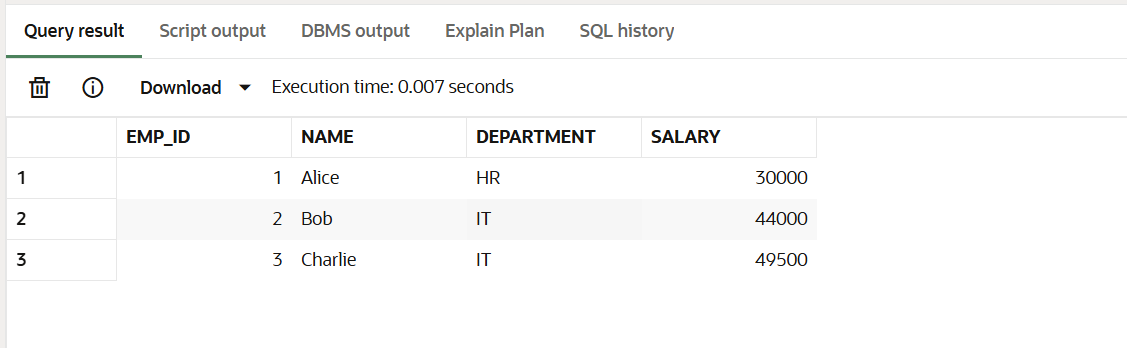
END;

/

**Step 5:** View result

SELECT \* FROM employees;

**OUTPUT:**

****

**Scenario 3:** Customers should be able to transfer funds between their accounts

**CODE:**

{Reusing accounts table from Scenario 1)

**Step 1:** Create stored procedure

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_account\_id IN NUMBER,

to\_account\_id IN NUMBER,

amount IN NUMBER

) AS

from\_balance NUMBER;

BEGIN

SELECT balance INTO from\_balance

FROM accounts

WHERE account\_id = from\_account\_id;

IF from\_balance < amount THEN

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

END IF;

UPDATE accounts

SET balance = balance - amount

WHERE account\_id = from\_account\_id;

UPDATE accounts

SET balance = balance + amount

WHERE account\_id = to\_account\_id;

END;

/

**Step 2:** Execute procedure

BEGIN

TransferFunds(101, 102, 1000);

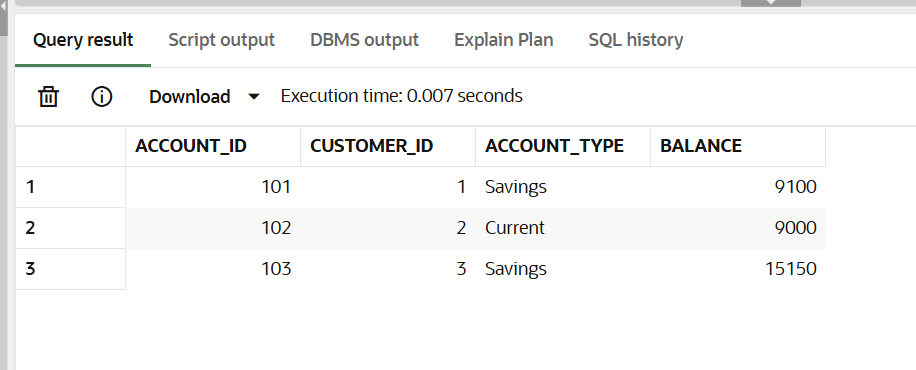
END;

/

**Step 3:** View result

SELECT \* FROM accounts;

**OUTPUT:**

****