Exercise 1: Control Structures

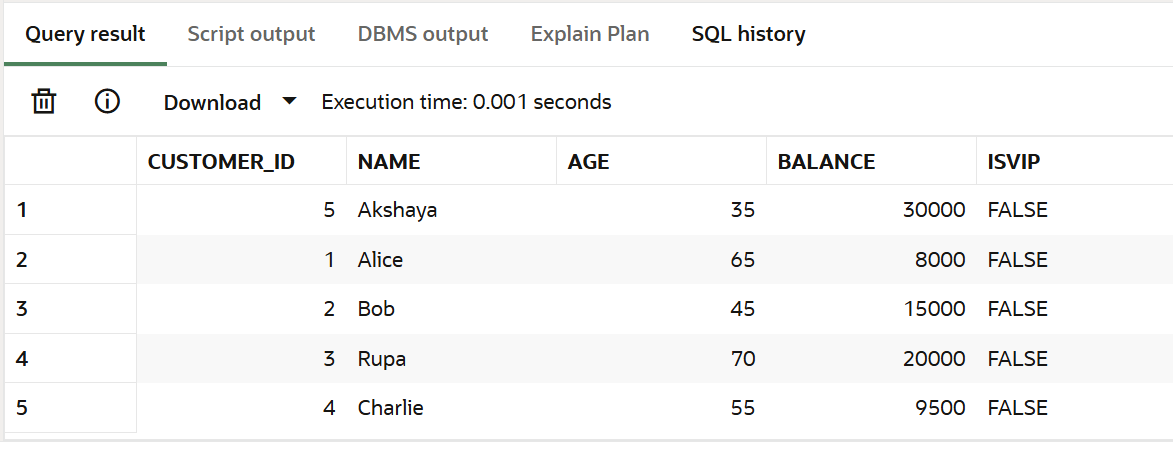
**Scenario 1: The bank wants to apply a discount to loan interest rates for customers above 60 years old.**

* **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.**

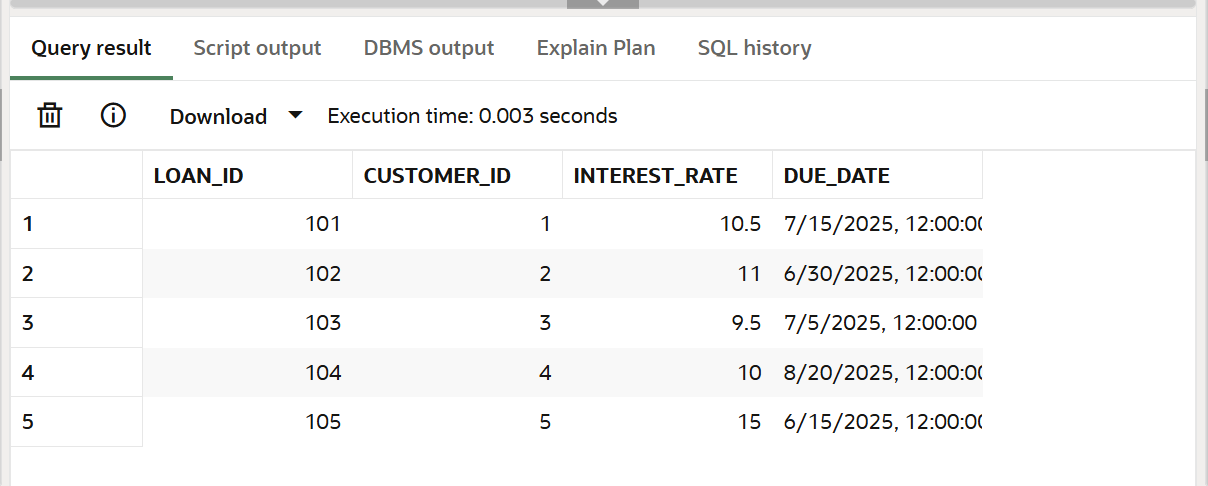
Solution:

Data:

SELECT \* FROM CUSTOMERS;



SELECT \* FROM loans;



Code:

DECLARE

CURSOR cust\_cursor IS

SELECT customer\_id, age FROM customers;

BEGIN

FOR cust IN cust\_cursor LOOP

IF cust.age > 60 THEN

UPDATE loans

SET interest\_rate = interest\_rate - 1

WHERE customer\_id = cust.customer\_id;

END IF;

END LOOP;

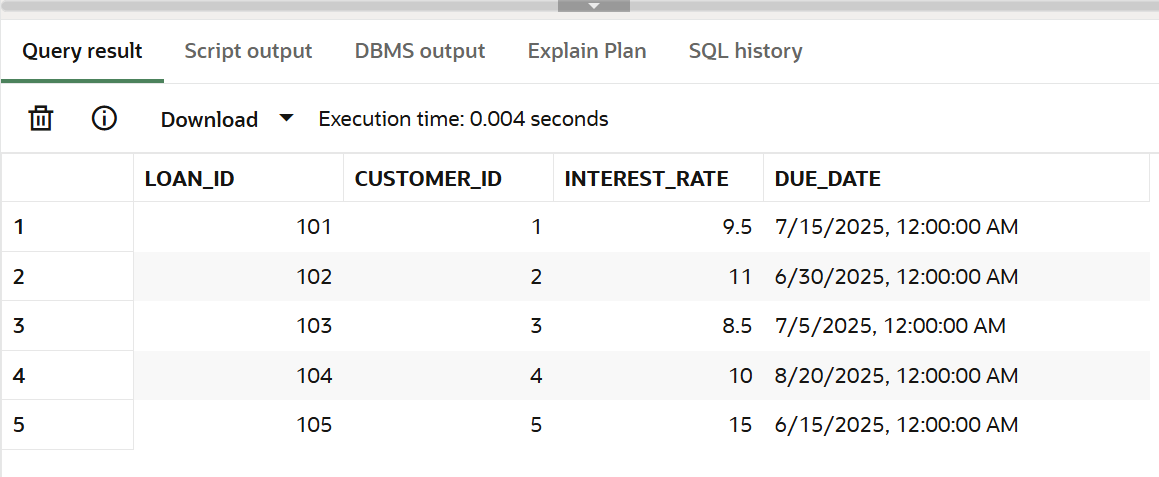
COMMIT;

END;

/

SELECT \* FROM LOANS;

Output:



**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.**

Code:

DECLARE

CURSOR cust\_cursor IS

SELECT customer\_id, balance FROM customers;

BEGIN

FOR cust IN cust\_cursor LOOP

IF cust.balance > 10000 THEN

UPDATE customers

SET IsVIP = 'TRUE'

WHERE customer\_id = cust.customer\_id;

END IF;

END LOOP;

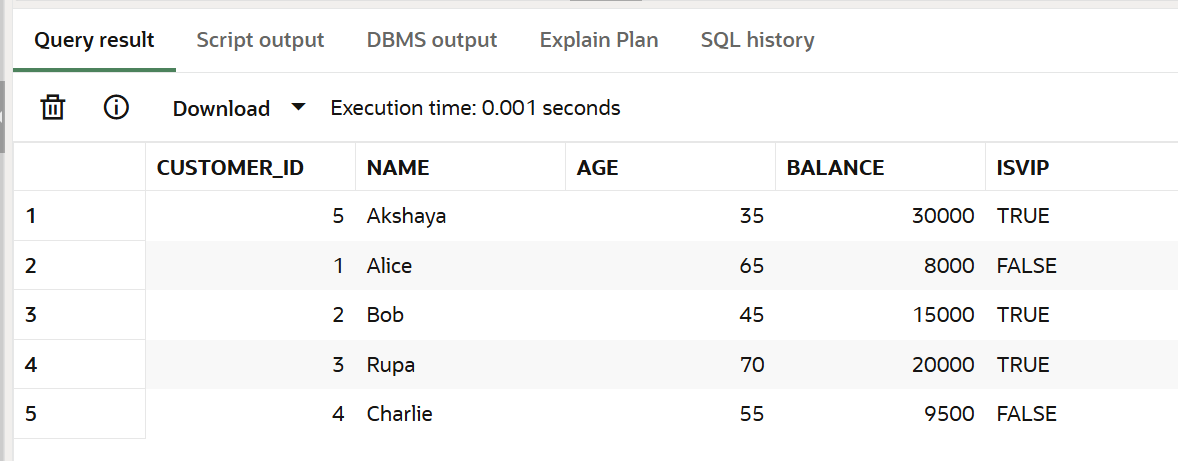
COMMIT;

END;

/

SELECT \* FROM CUSTOMERS;

Output:



**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.**

Code:

DECLARE

CURSOR loan\_cursor IS

SELECT customer\_id, due\_date

FROM loans

WHERE due\_date BETWEEN SYSDATE AND SYSDATE + 30;

BEGIN

FOR loan IN loan\_cursor LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Customer ' || loan.customer\_id ||

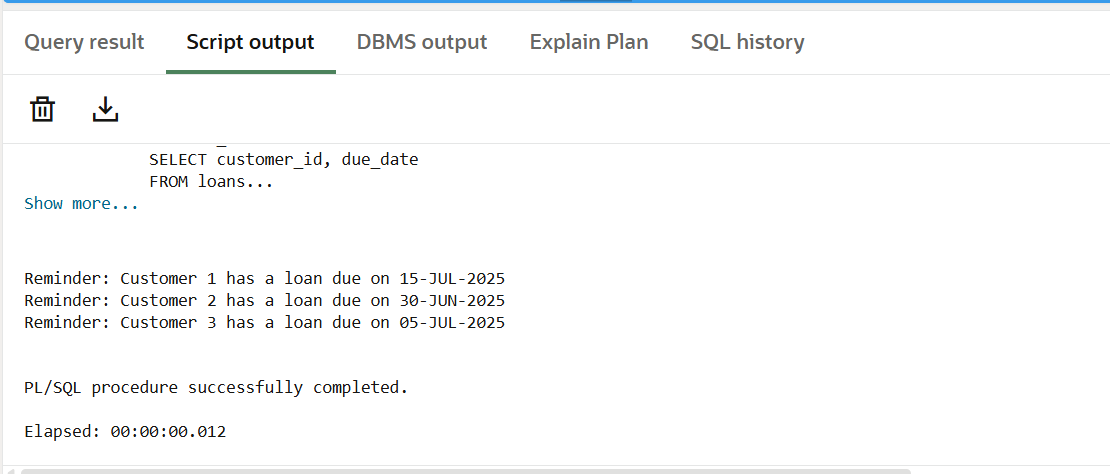
' has a loan due on ' || TO\_CHAR(loan.due\_date, 'DD-MON-YYYY'));

END LOOP;

END;

/

Output:



Exercise 3: Stored Procedures

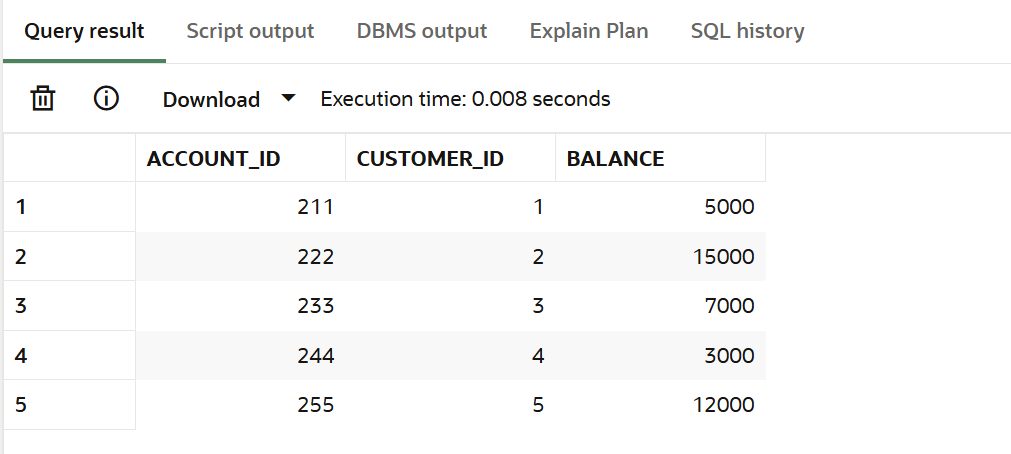
**Scenario 1: The bank needs to process monthly interest for all savings accounts.**

* **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.**

Solution:

Data:

select \* from SAVINGS\_ACCOUNTS;



Code:

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE savings\_accounts

SET balance = balance + (balance \* 0.01); -- Adds 1% interest

COMMIT;

END;

/

BEGIN

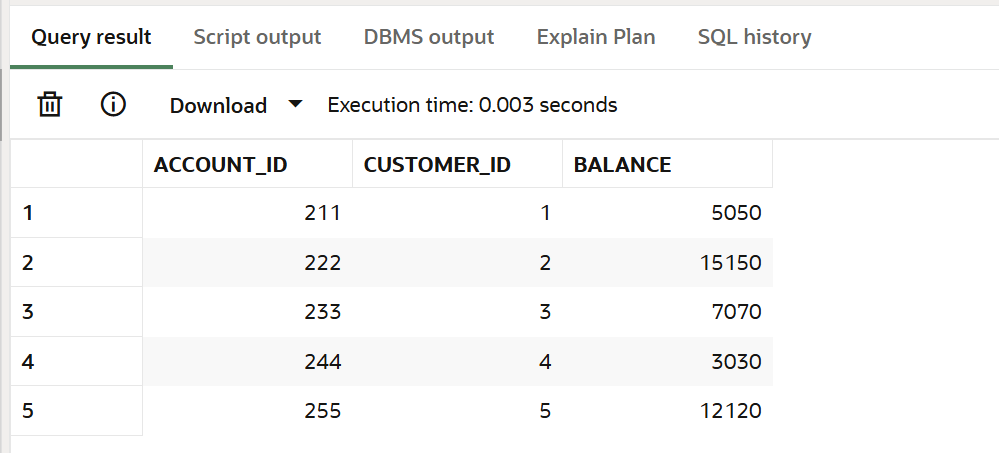
ProcessMonthlyInterest;

END;

/

SELECT \* FROM SAVINGS\_ACCOUNTS;

Output:

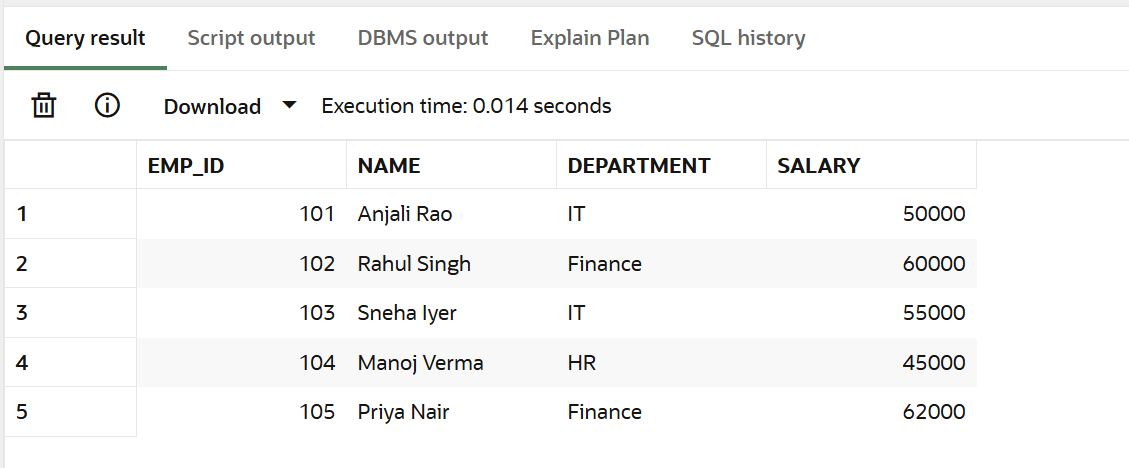


**Scenario 2: The bank wants to implement a bonus scheme for employees based on their performance.**

* **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.**

Data:

SELECT \* FROM EMPLOYEES;



Code:

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE employees

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

WHERE department = dept\_name;

COMMIT;

END;

/

BEGIN

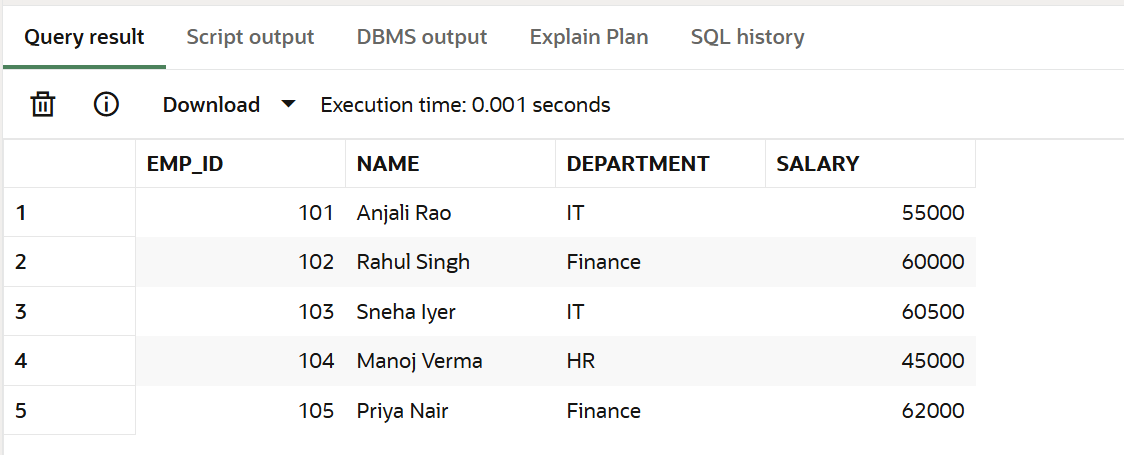
UpdateEmployeeBonus('IT', 10); -- for example: 10% bonus for IT department

END;

/

SELECT \* FROM EMPLOYEES;

Output:

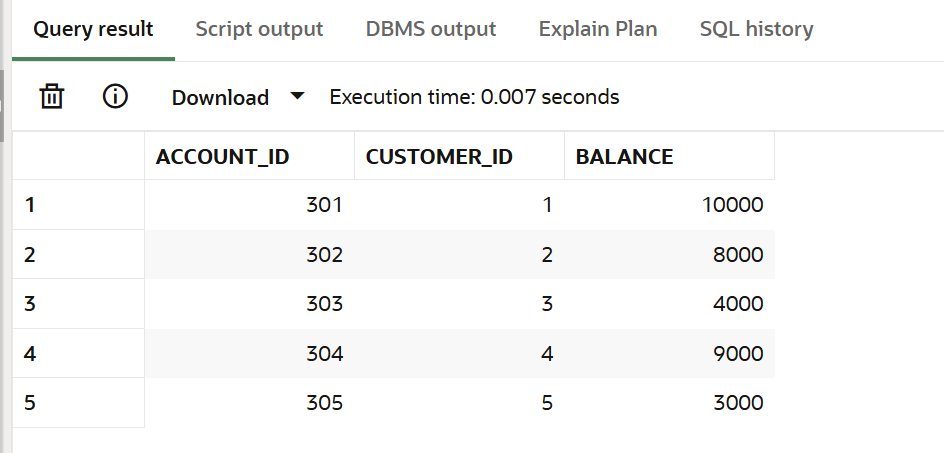


**Scenario 3: Customers should be able to transfer funds between their accounts.**

* **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.**

Data:

SELECT \* FROM ACCOUNTS;



Code:

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_account IN NUMBER,

to\_account IN NUMBER,

amount IN NUMBER

) IS

from\_balance NUMBER;

BEGIN

-- Step 1: Get the balance of the source account

SELECT balance INTO from\_balance

FROM accounts

WHERE account\_id = from\_account;

-- Step 2: Check if source has enough balance

IF from\_balance < amount THEN

DBMS\_OUTPUT.PUT\_LINE(' Transfer failed: Insufficient balance in account ' || from\_account);

ELSE

-- Step 3: Proceed with the transfer

UPDATE accounts

SET balance = balance - amount

WHERE account\_id = from\_account;

UPDATE accounts

SET balance = balance + amount

WHERE account\_id = to\_account;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(' Transfer successful: ₹' || amount ||

' transferred from account ' || from\_account ||

' to account ' || to\_account);

END IF;

END;

/

SET SERVEROUTPUT ON;

BEGIN

TransferFunds(301, 302, 2000); -- should work if account 301 has >= 2000

END;

/

Output:

