Exercise 1: Control Structures

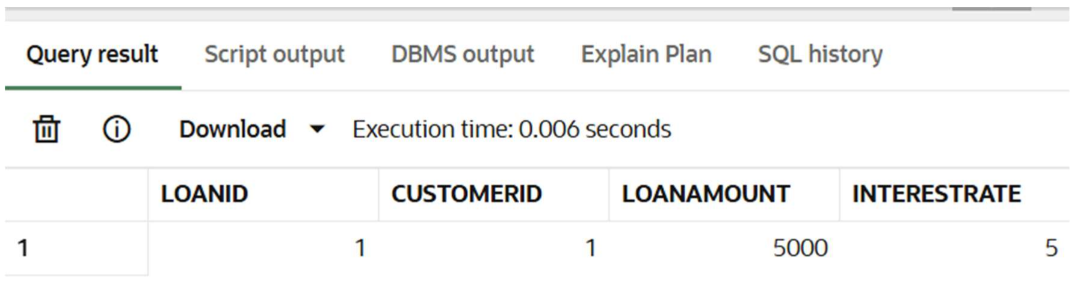
Scenario 1: The bank wants to apply a discount to loan interest rates for customers above 60 years old.

- **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.

Apply 1% Discount to interest Rates for customers Above 60,

```
BEGIN
 FOR rec IN (SELECT customer_id, interest_rate FROM customers WHERE age > 60) LOOP
  UPDATE customers
  SET interest_rate = interest_rate - (interest_rate * 0.01)
  WHERE customer_id = rec.customer_id;
 END LOOP;
 COMMIT;
END;
/

SELECT * FROM Loans;
```

Output:


Query result   Script output   DBMS output   Explain Plan   SQL history

Download   Execution time: 0.006 seconds

	LOANID	CUSTOMERID	LOANAMOUNT	INTERESTRATE
1	1	1	5000	5

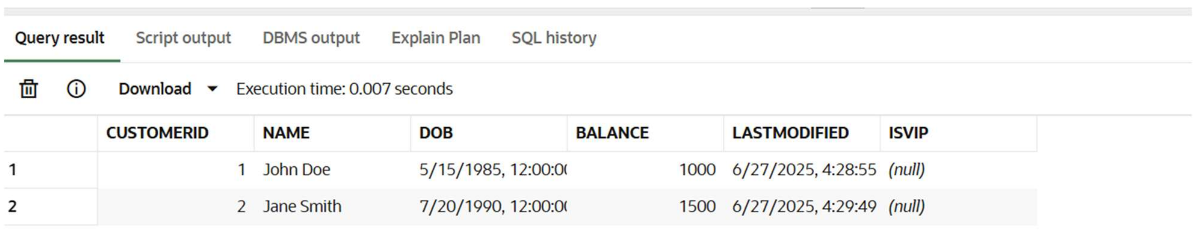
Scenario 2: A customer can be promoted to VIP status based on their balance.

- **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.

Set IsVIP to TRUE for Customers with Balance > 10,000,

```
BEGIN
 FOR rec IN (SELECT customer_id FROM customers WHERE balance > 10000) LOOP
  UPDATE customers
  SET IsVIP = 'TRUE'
  WHERE customer_id = rec.customer_id;
 END LOOP;
 COMMIT;
END;
/

SELECT * FROM Customer;
```

Output:



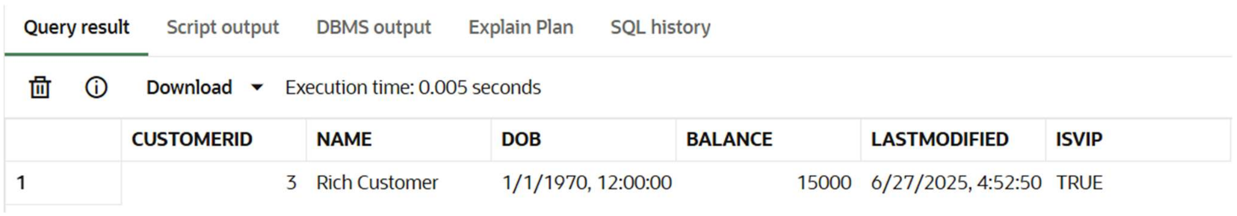
	CUSTOMERID	NAME	DOB	BALANCE	LASTMODIFIED	ISVIP
1	1	John Doe	5/15/1985, 12:00:0(	1000	6/27/2025, 4:28:55	(null)
2	2	Jane Smith	7/20/1990, 12:00:0(	1500	6/27/2025, 4:29:49	(null)

Both ISVIP values remain NULL since neither customer has a balance > $10,000.

Inserting a new customer,

```
INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)
VALUES (3, 'Rich Customer', TO_DATE('1970-01-01', 'YYYY-MM-DD'), 15000, SYSDATE);

SELECT * FROM CUSTOMERS WHERE ISVIP = TRUE;
```

Output:



	CUSTOMERID	NAME	DOB	BALANCE	LASTMODIFIED	ISVIP
1	3	Rich Customer	1/1/1970, 12:00:00	15000	6/27/2025, 4:52:50	TRUE

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.

Print Reminder Messages for Loans Due in Next 30 Days,

```
BEGIN
 FOR rec IN (
  SELECT CustomerID, LoanID, EndDate
  FROM Loans
  WHERE EndDate BETWEEN SYSDATE AND SYSDATE + 30
 ) LOOP
  DBMS_OUTPUT.PUT_LINE(
   'Reminder: Customer ID ' || rec.CustomerID ||
   ' has Loan ID ' || rec.LoanID ||
   ' due on ' || TO_CHAR(rec.EndDate, 'DD-MON-YYYY')
  );
 END LOOP;
END;
/
```

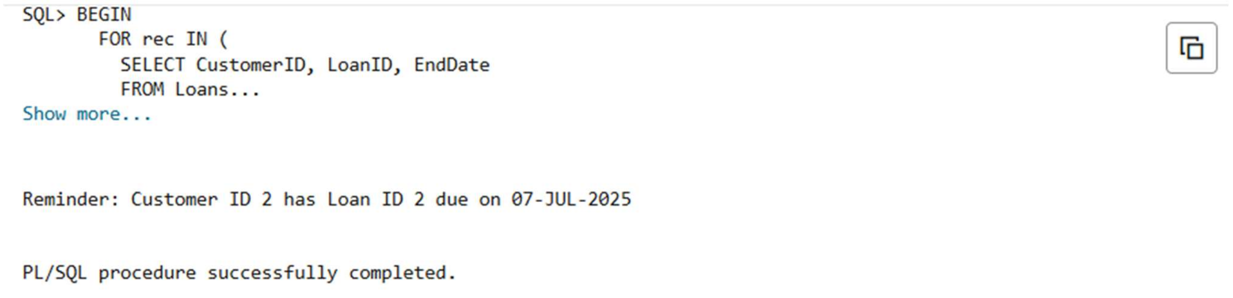
Based on the current data the output is,

```
PL/SQL procedure successfully completed.
```

Insert new loan that ends within 30 days,

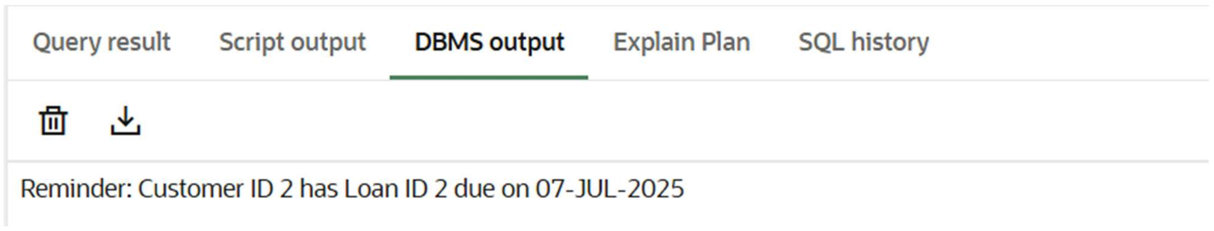
INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)
VALUES (2, 2, 3000, 6, SYSDATE, SYSDATE + 10);

OUTPUT after Inserting the new Loan,



```
SQL> BEGIN
       FOR rec IN (
         SELECT CustomerID, LoanID, EndDate
         FROM Loans...
Show more...


Reminder: Customer ID 2 has Loan ID 2 due on 07-JUL-2025


PL/SQL procedure successfully completed.
```

DBMS output,



```
Reminder: Customer ID 2 has Loan ID 2 due on 07-JUL-2025
```

Exercise 3: Stored Procedures

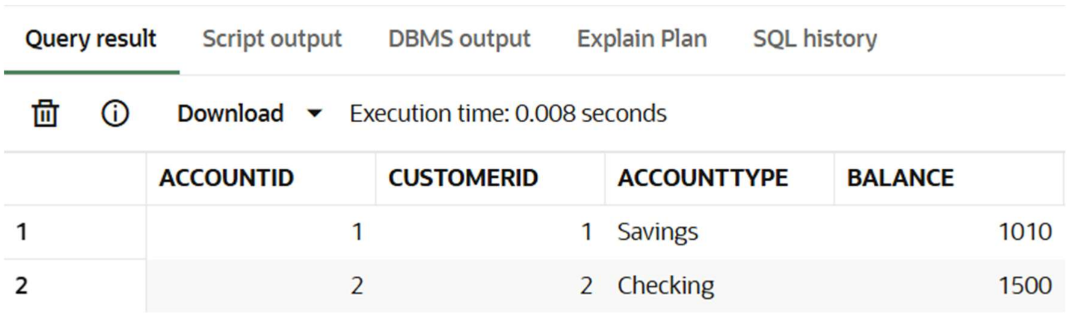
Scenario 1: The bank needs to process monthly interest for all savings accounts.

- **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.

Process Monthly Interest (1%) for Savings Accounts,

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
BEGIN
 FOR rec IN (
  SELECT AccountID, Balance
  FROM Accounts
  WHERE AccountType = 'Savings'
 ) LOOP
  UPDATE Accounts
  SET Balance = Balance + (rec.Balance * 0.01)
  WHERE AccountID = rec.AccountID;
 END LOOP;
 COMMIT;
END;
/

SELECT * FROM Accounts;
```

Output:
Procedure updates the savings by 1% (1000 to 1010),
Checking account balance should remain unchanged.



Execution time: 0.008 seconds

	ACCOUNTID	CUSTOMERID	ACCOUNTTYPE	BALANCE
1	1	1	Savings	1010
2	2	2	Checking	1500

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.

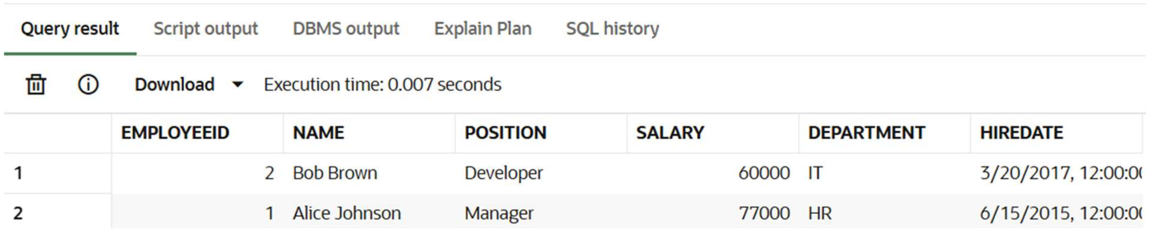
  Update Employee Bonus in a Department,

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
  p_dept IN VARCHAR2,
  p_bonus_percent IN NUMBER
) IS
BEGIN
  UPDATE Employees
  SET Salary = Salary + (Salary * (p_bonus_percent / 100))
  WHERE Department = p_dept;

  COMMIT;
END;
/
```

  For example, giving **10% bonus to HR department**:

```
BEGIN
  UpdateEmployeeBonus('HR', 10);
END;
/

SELECT * FROM Employees;
```

  Output:
  Employee in HR have increased salary (10% added[70000 to 77000])



	EMPLOYEEID	NAME	POSITION	SALARY	DEPARTMENT	HIREDATE
1	2	Bob Brown	Developer	60000	IT	3/20/2017, 12:00:0
2	1	Alice Johnson	Manager	77000	HR	6/15/2015, 12:00:0

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.

  Transfer Funds Between Two Accounts,

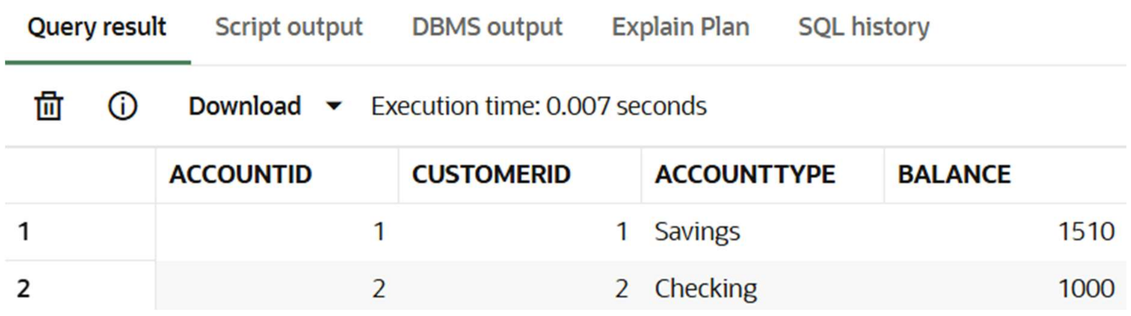
```
CREATE OR REPLACE PROCEDURE TransferFunds (
  p_from_account IN NUMBER,
  p_to_account   IN NUMBER,
  p_amount       IN NUMBER
) IS
  v_balance NUMBER;
BEGIN
  -- Get the current balance of the source account
  SELECT Balance INTO v_balance
  FROM Accounts
  WHERE AccountID = p_from_account;

  -- Check if sufficient balance exists
  IF v_balance >= p_amount THEN
    -- Deduct from source
    UPDATE Accounts
    SET Balance = Balance - p_amount
    WHERE AccountID = p_from_account;

    -- Add to destination
    UPDATE Accounts
    SET Balance = Balance + p_amount
    WHERE AccountID = p_to_account;

    COMMIT;
  ELSE
    -- Not enough balance → raise error
    RAISE_APPLICATION_ERROR(-20001, 'Insufficient balance in source account.');
  END IF;
END;
/
```

  For e.g., transfer ₹500 from AccountID = 2 to AccountID = 1:

```
BEGIN
  TransferFunds(2, 1, 500);
END;
/
```

SELECT * FROM Accounts;

Output:
Before transfer Account ID 1 Balance is 1010 after transfer 1510,



Query result | Script output | DBMS output | Explain Plan | SQL history

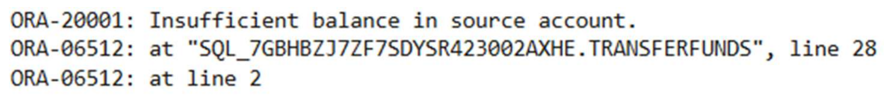
Download | Execution time: 0.007 seconds

	ACCOUNTID	CUSTOMERID	ACCOUNTTYPE	BALANCE
1	1	1	Savings	1510
2	2	2	Checking	1000

Trying to transfer Too Much

```
BEGIN
  TransferFunds(2, 1, 999999);
END;
/
```

Output:



```
ORA-20001: Insufficient balance in source account.
ORA-06512: at "SQL_7GBHBZJ7ZF7SDYSR423002AXHE.TRANSFERFUNDS", line 28
ORA-06512: at line 2
```