Exercise 1: Control Structures

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

Loan:

LOAN_ID	CUSTOMER_ID	LOAN_AMOUNT	LOAN_INTEREST_RATE	DUE_DATE
102	2	30000	10	8/7/2025
103	3	45000	8	7/8/2025
101	1	50000	9.5	7/18/2025
104	4	60000	11	7/3/2025

4 rows returned in 0.01 seconds Download

Customers:

CUSTOMER_ID	NAME	AGE	BALANCE	ISVIP
1	Alice	65	12000	FALSE
2	Bob	45	8000	FALSE
4	Diana	30	9500	FALSE
3	Charlie	70	15000	FALSE

4 rows returned in 0.02 seconds Download

1. **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:

```
BEGIN
   FOR cust IN (
      SELECT c.customer_id, l.loan_id
      FROM customers c
      JOIN loans l ON c.customer_id = l.customer_id
      WHERE c.age > 60
   ) LOOP
      UPDATE loans
      SET loan_interest_rate = loan_interest_rate - 1
      WHERE loan_id = cust.loan_id;
   END LOOP;

   COMMIT;
END;
```

CUSTOMER_ID	NAME	AGE	BALANCE	ISVIP
1	Alice	65	12000	FALSE
2	Bob	45	8000	FALSE
4	Diana	30	9500	FALSE
3	Charlie	70	15000	FALSE

4 rows returned in 0.01 seconds Download

Results Explain Describe Saved SQL History

LOAN_ID	CUSTOMER_ID	LOAN_AMOUNT	LOAN_INTEREST_RATE	DUE_DATE
102	2	30000	10	8/7/2025
103	3	45000	7	7/8/2025
101	1	50000	8.5	7/18/2025
104	4	60000	11	7/3/2025

4 rows returned in 0.00 seconds Download

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:

```
 BEGIN
FOR cust IN (
SELECT customer_id, balance
FROM customers
WHERE balance > 10000
) LOOP
UPDATE customers
SET isVIP = 'TRUE'
WHERE customer_id = cust.customer_id;
END LOOP;

COMMIT;
END;
```

CUSTOMER_ID	NAME	AGE	BALANCE	ISVIP
1	Alice	65	12000	TRUE
2	Bob	45	8000	FALSE
4	Diana	30	9500	FALSE
3	Charlie	70	15000	TRUE

4 rows returned in 0.01 seconds Download

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.

Query:

```
BEGIN
   FOR rec IN (
      SELECT l.loan_id, c.name, l.due_date
      FROM loans409 l
      JOIN customers409 c ON l.customer_id = c.customer_id
      WHERE l.due_date BETWEEN SYSDATE AND SYSDATE + 30
   ) LOOP
      DBMS_OUTPUT.PUT_LINE('Reminder: Dear ' || rec.name ||
                           ', your loan (ID: ' || rec.loan_id ||
                           ') is due on ' || TO_CHAR(rec.due_date, 'DD-Mon-YYYY'));
   END LOOP;
END;
```

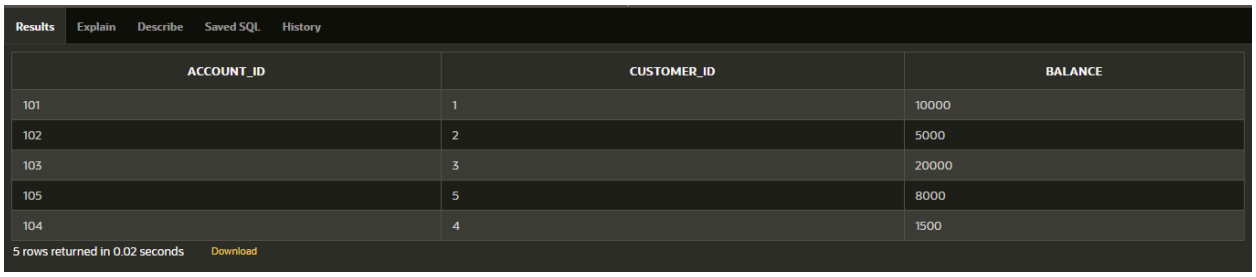
```
Reminder: Dear Charlie, your loan (ID: 103) is due on 08-Jul-2025
Reminder: Dear Alice, your loan (ID: 101) is due on 18-Jul-2025
Reminder: Dear Diana, your loan (ID: 104) is due on 03-Jul-2025

Statement processed.
```

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.

Table:



Results | Explain | Describe | Saved SQL | History

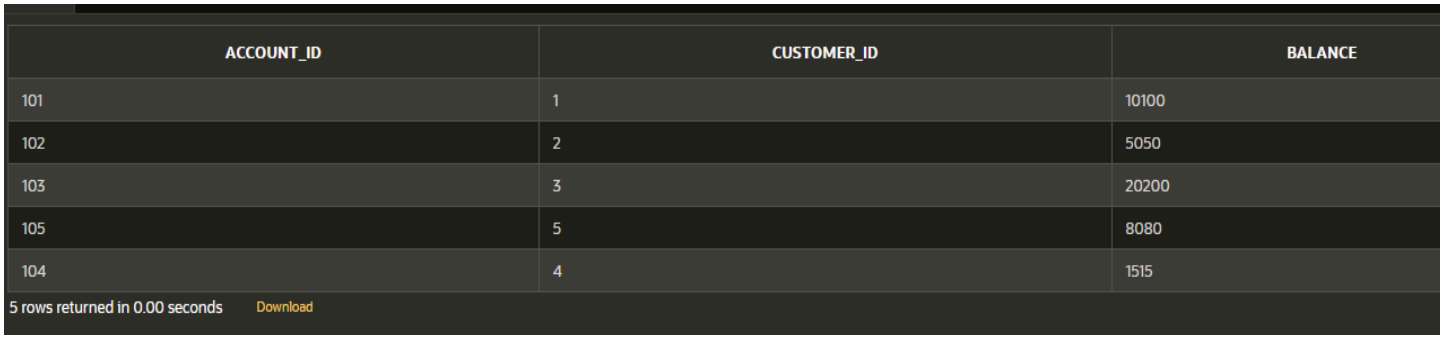
ACCOUNT_ID	CUSTOMER_ID	BALANCE
101	1	10000
102	2	5000
103	3	20000
105	5	8000
104	4	1500

5 rows returned in 0.02 seconds Download

Query:

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
BEGIN
   FOR acc IN (SELECT account_id, balance FROM savings_accounts) LOOP
      UPDATE savings_accounts409
      SET balance = balance + (balance * 0.01)
      WHERE account_id = acc.account_id;
   END LOOP;

   COMMIT;
END;
```

OUTPUT:



ACCOUNT_ID	CUSTOMER_ID	BALANCE
101	1	10100
102	2	5050
103	3	20200
105	5	8080
104	4	1515

5 rows returned in 0.00 seconds Download

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.
-

Table:

EMP_ID	NAME	DEPT_ID	SALARY
1	Arjun	101	50000
4	Divya	103	58000
5	Eesha	101	52000
2	Banu	102	60000
3	Chetan	101	55000

5 rows returned in 0.01 seconds Download

Query:

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
    p_dept_id IN NUMBER,
    p_bonus_percent IN NUMBER
) IS
BEGIN
    UPDATE employees
    SET salary = salary + (salary * (p_bonus_percent / 100))
    WHERE dept_id = p_dept_id;

    COMMIT;
END;
```

OUTPUT:

EMP_ID	NAME	DEPT_ID	SALARY
1	Arjun	101	55000
4	Divya	103	58000
5	Eesha	101	57200
2	Banu	102	60000
3	Chetan	101	60500

5 rows returned in 0.00 seconds Download

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.

TABLE:

ACCOUNT_ID	CUSTOMER_NAME	BALANCE
2	Bob	10000
3	Charlie	2000
1	Alice	15000
4	Diana	7000
5	Ethan	3000

5 rows returned in 0.02 seconds Download

QUERY:

```
CREATE OR REPLACE PROCEDURE TransferFunds (
   p_from_account IN NUMBER,
   p_to_account IN NUMBER,
   p_amount IN NUMBER
) IS
   v_balance NUMBER;
BEGIN
   -- Check if from_account has sufficient balance
   SELECT balance INTO v_balance
   FROM bank_accounts
   WHERE account_id = p_from_account;

   IF v_balance < p_amount THEN
      RAISE_APPLICATION_ERROR(-20001, 'Insufficient balance in source account.');
   END IF;

   -- Deduct from source
   UPDATE bank_accounts
   SET balance = balance - p_amount
   WHERE account_id = p_from_account;

   -- Add to target
   UPDATE bank_accounts
   SET balance = balance + p_amount
   WHERE account_id = p_to_account;
END;

BEGIN
   TransferFunds(1, 4, 2000);
END;
```

OUTPUT:

ACCOUNT_ID	CUSTOMER_NAME	BALANCE
2	Bob	10000
3	Charlie	2000
1	Alice	13000
4	Diana	9000
5	Ethan	3000

5 rows returned in 0.00 seconds Download