PL/SQL Programming

Exercise 1: Control Structures

Create Tables

Customers Table

```
CREATE TABLE customers (
  customer_id NUMBER PRIMARY KEY,
  name VARCHAR2(50),
  age NUMBER,
  balance NUMBER(10, 2),
  is_vip VARCHAR2(1)
);
```

Output:

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

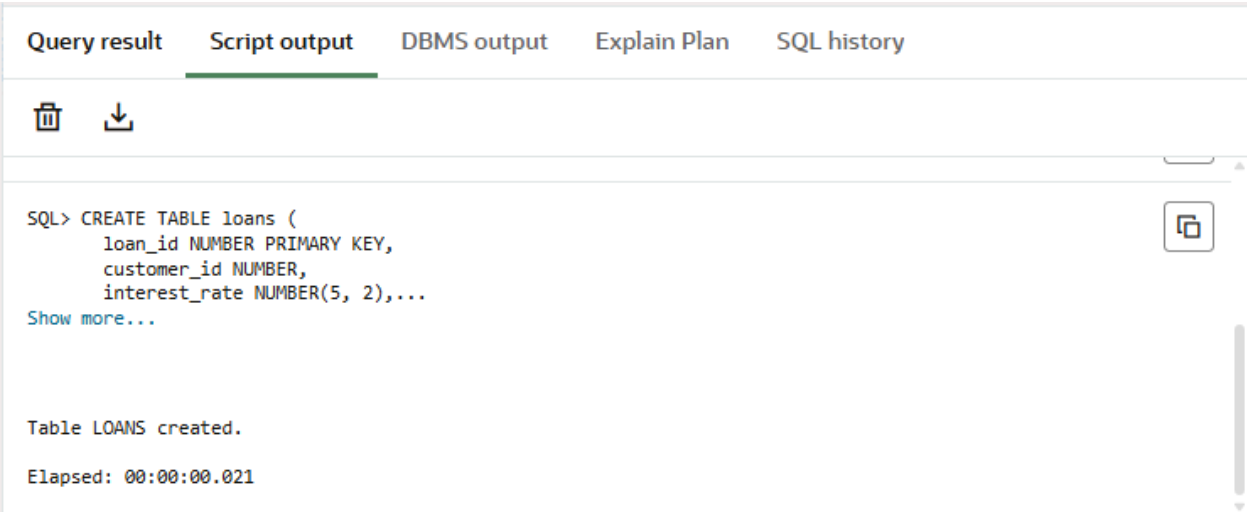
```
SQL> CREATE TABLE customers (
       customer_id NUMBER PRIMARY KEY,
       name VARCHAR2(50),
       age NUMBER,...
Show more...


Table CUSTOMERS created.

Elapsed: 00:00:00.016
```

Loans Table

```
CREATE TABLE loans (
  loan_id NUMBER PRIMARY KEY,
```

```
  customer_id NUMBER,
  interest_rate NUMBER(5, 2),
  due_date DATE,
  FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
```

Output:



Insert Sample Data

Insert Data into Customers

```
INSERT INTO customers VALUES (1, 'John', 65, 12000.00, 'N');
INSERT INTO customers VALUES (2, 'Alice', 45, 8000.00, 'N');
INSERT INTO customers VALUES (3, 'David', 70, 15000.00, 'N');
INSERT INTO customers VALUES (4, 'Mary', 30, 5000.00, 'N');
```

Output:

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

Download | Execution time: 0.014 seconds

	CUSTOMER_ID	NAME	AGE	BALANCE	IS_VIP
1	1	John	65	12000	N
2	2	Alice	45	8000	N
3	3	David	70	15000	N
4	4	Mary	30	5000	N

Insert Data into Loans

```
INSERT INTO loans VALUES (101, 1, 8.5, SYSDATE + 10);
INSERT INTO loans VALUES (102, 2, 9.0, SYSDATE + 40);
INSERT INTO loans VALUES (103, 3, 7.5, SYSDATE + 20);
INSERT INTO loans VALUES (104, 4, 10.0, SYSDATE + 15);
```

Output:

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

Download | Execution time: 0.008 seconds

	LOAN_ID	CUSTOMER_ID	INTEREST_RATE	DUE_DATE
1	101	1	8.5	7/8/2025, 4:51:00 PM
2	102	2	9	8/7/2025, 4:51:00 PM
3	103	3	7.5	7/18/2025, 4:51:00 PM
4	104	4	10	7/13/2025, 4:51:00 PM

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.

PL/SQL Block:

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

Output:

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

```
SQL> BEGIN
       FOR rec IN (
         SELECT l.loan_id
         FROM loans l...
Show more...


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.019
```

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

Download ▾ Execution time: 0.001 seconds

	LOAN_ID	CUSTOMER_ID	INTEREST_RATE	DUE_DATE
1	101	1	7.5	7/8/2025, 4:51:00 P
2	102	2	9	8/7/2025, 4:51:00 P
3	103	3	6.5	7/18/2025, 4:51:00
4	104	4	10	7/13/2025, 4:51:00

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.

PL/SQL Block:

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

Output:

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

```
SQL> BEGIN
       FOR rec IN (
         SELECT customer_id
         FROM customers...
Show more...


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.036
```

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

Download | Execution time: 0.006 seconds

	CUSTOMER_ID	NAME	AGE	BALANCE	IS_VIP
1	1	John	65	12000	Y
2	2	Alice	45	8000	N
3	3	David	70	15000	Y
4	4	Mary	30	5000	N

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.

Enable Output:

```
SET SERVEROUTPUT ON;
```

PL/SQL Block:

```
BEGIN
  FOR rec IN (
    SELECT c.name, l.loan_id, l.due_date
    FROM customers c
    JOIN loans l ON c.customer_id = l.customer_id
    WHERE l.due_date <= SYSDATE + 30
  ) LOOP
    DBMS_OUTPUT.PUT_LINE('Reminder: ' || rec.name ||
                         ', your loan #' || rec.loan_id ||
                         ' is due on ' || TO_CHAR(rec.due_date, 'DD-MON-YYYY'));
  END LOOP;
END;
```

Output:

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

```
SQL> BEGIN
       FOR rec IN (
         SELECT c.name, l.loan_id, l.due_date
         FROM customers c...
Show more...


Reminder: John, your loan #101 is due on 08-JUL-2025
Reminder: David, your loan #103 is due on 18-JUL-2025
Reminder: Mary, your loan #104 is due on 13-JUL-2025


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.015
```

Exercise 3: Stored Procedures

Create Tables

Accounts Table

```
CREATE TABLE accounts (
  account_id NUMBER PRIMARY KEY,
  customer_name VARCHAR2(50),
  balance NUMBER(10, 2)
);
```

Output:

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

```
SQL> CREATE TABLE accounts (
       account_id NUMBER PRIMARY KEY,
       customer_name VARCHAR2(50),
       balance NUMBER(10, 2)...
Show more...


Table ACCOUNTS created.

Elapsed: 00:00:00.017
```

Employees Table

```
CREATE TABLE employees (
  emp_id NUMBER PRIMARY KEY,
  name VARCHAR2(50),
  department VARCHAR2(50),
  salary NUMBER(10, 2)
);
```

Output:

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

```
SQL> CREATE TABLE employees (
       emp_id NUMBER PRIMARY KEY,
       name VARCHAR2(50),
       department VARCHAR2(50),...
Show more...


Table EMPLOYEES created.

Elapsed: 00:00:00.017
```

Insert Sample Data

Insert into accounts table

```
INSERT INTO accounts VALUES (101, 'John', 10000);
INSERT INTO accounts VALUES (102, 'Alice', 15000);
INSERT INTO accounts VALUES (103, 'David', 5000);
```

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

Download ▾ Execution time: 0.009 seconds

	ACCOUNT_ID	CUSTOMER_NAME	BALANCE
1	101	John	10000
2	102	Alice	15000
3	103	David	5000

Insert into employees table

```
INSERT INTO employees VALUES (1, 'Mark', 'HR', 50000);
INSERT INTO employees VALUES (2, 'Jane', 'IT', 60000);
INSERT INTO employees VALUES (3, 'Alex', 'HR', 55000);
```

Output:

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

Download | Execution time: 0.009 seconds

	EMP_ID	NAME	DEPARTMENT	SALARY
1	1	Mark	HR	50000
2	2	Jane	IT	60000
3	3	Alex	HR	55000

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.

Stored Procedure

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS
BEGIN
  FOR rec IN (SELECT account_id FROM accounts) LOOP
    UPDATE accounts
    SET balance = balance + (balance * 0.01)
    WHERE account_id = rec.account_id;
  END LOOP;
  COMMIT;
END;
```

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

```
SQL> CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS
     BEGIN
       FOR rec IN (SELECT account_id FROM accounts) LOOP
         UPDATE accounts...
Show more...


Procedure PROCESSMONTHLYINTEREST compiled

Elapsed: 00:00:00.016
```

Execute Procedure

```
BEGIN
  ProcessMonthlyInterest;
END;
```

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

```
SQL> BEGIN
       ProcessMonthlyInterest;
     END;


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.021
```

Output:

```
SELECT * FROM accounts;
```

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

Download ▾ Execution time: 0.004 seconds

	ACCOUNT_ID	CUSTOMER_NAME	BALANCE
1	101	John	10100
2	102	Alice	15150
3	103	David	5050

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.

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;

  COMMIT;
END;
```

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

```
SQL> CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(
       dept_name IN VARCHAR2,
       bonus_percent IN NUMBER
     ) AS...
Show more...


Procedure UPDATEEMPLOYEEBONUS compiled

Elapsed: 00:00:00.018
```

Execute Procedure

```
BEGIN
  UpdateEmployeeBonus('HR', 10); -- 10% bonus to HR
END;
```

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

```
SQL> BEGIN
       UpdateEmployeeBonus('HR', 10); -- 10% bonus to HR
     END;


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.024
```

Output:

```
SELECT * FROM employees;
```

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

Download ▾ Execution time: 0.005 seconds

	EMP_ID	NAME	DEPARTMENT	SALARY
1	1	Mark	HR	55000
2	2	Jane	IT	60000
3	3	Alex	HR	60500

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.

Stored Procedure

```
CREATE OR REPLACE PROCEDURE TransferFunds(
  from_acc IN NUMBER,
  to_acc IN NUMBER,
  amount IN NUMBER
) AS
  from_balance NUMBER;
BEGIN
  SELECT balance INTO from_balance FROM accounts WHERE account_id = fr
om_acc;

  IF from_balance >= amount THEN
    UPDATE accounts
    SET balance = balance - amount
    WHERE account_id = from_acc;

    UPDATE accounts
    SET balance = balance + amount
    WHERE account_id = to_acc;
```

```
    COMMIT;
  ELSE
    DBMS_OUTPUT.PUT_LINE('Insufficient balance in account ' || from_acc);
  END IF;
END;
```

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

```
SQL> CREATE OR REPLACE PROCEDURE TransferFunds(
       from_acc IN NUMBER,
       to_acc IN NUMBER,
       amount IN NUMBER...
Show more...


Procedure TRANSFERFUNDS compiled

Elapsed: 00:00:00.014
```

Enable Output & Execute Procedure

```
SET SERVEROUTPUT ON;

BEGIN
  TransferFunds(102, 103, 2000);
END;
```

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

```
SQL> BEGIN
       TransferFunds(102, 103, 2000);
     END;


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.011
```

Output:

```
SELECT * FROM accounts;
```

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

Download ▾ Execution time: 0.001 seconds

	ACCOUNT_ID	CUSTOMER_NAME	BALANCE
1	101	John	10100
2	102	Alice	13150
3	103	David	7050