**PL/SQL PROGRAMMING**

**Exercise 1: Control Structures**

**Step 1: Create the tables:**

**CODE:**

CREATE TABLE Customers (

  CustomerID NUMBER,

  Name VARCHAR2(50),

  Age NUMBER,

  Balance NUMBER,

  LoanInterestRate NUMBER,

  IsVIP VARCHAR2(5)

);

CREATE TABLE Loans (

  LoanID NUMBER,

  CustomerID NUMBER,

  DueDate DATE

);

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**Step 2: Insert the values:**

**CODE:**

-- Customers

INSERT INTO Customers VALUES (101, 'Kaviya', 65, 12000, 10, 'FALSE');

INSERT INTO Customers VALUES (102, 'Ravi', 45, 8000, 9, 'FALSE');

INSERT INTO Customers VALUES (103, 'Meena', 70, 15000, 11, 'FALSE');

INSERT INTO Customers VALUES (104, 'Rahul', 55, 9500, 8, 'FALSE');

-- Loans

INSERT INTO Loans VALUES (1, 101, SYSDATE + 10); -- Due soon

INSERT INTO Loans VALUES (2, 102, SYSDATE + 35); -- Not due

INSERT INTO Loans VALUES (3, 103, SYSDATE + 5);  -- Due soon

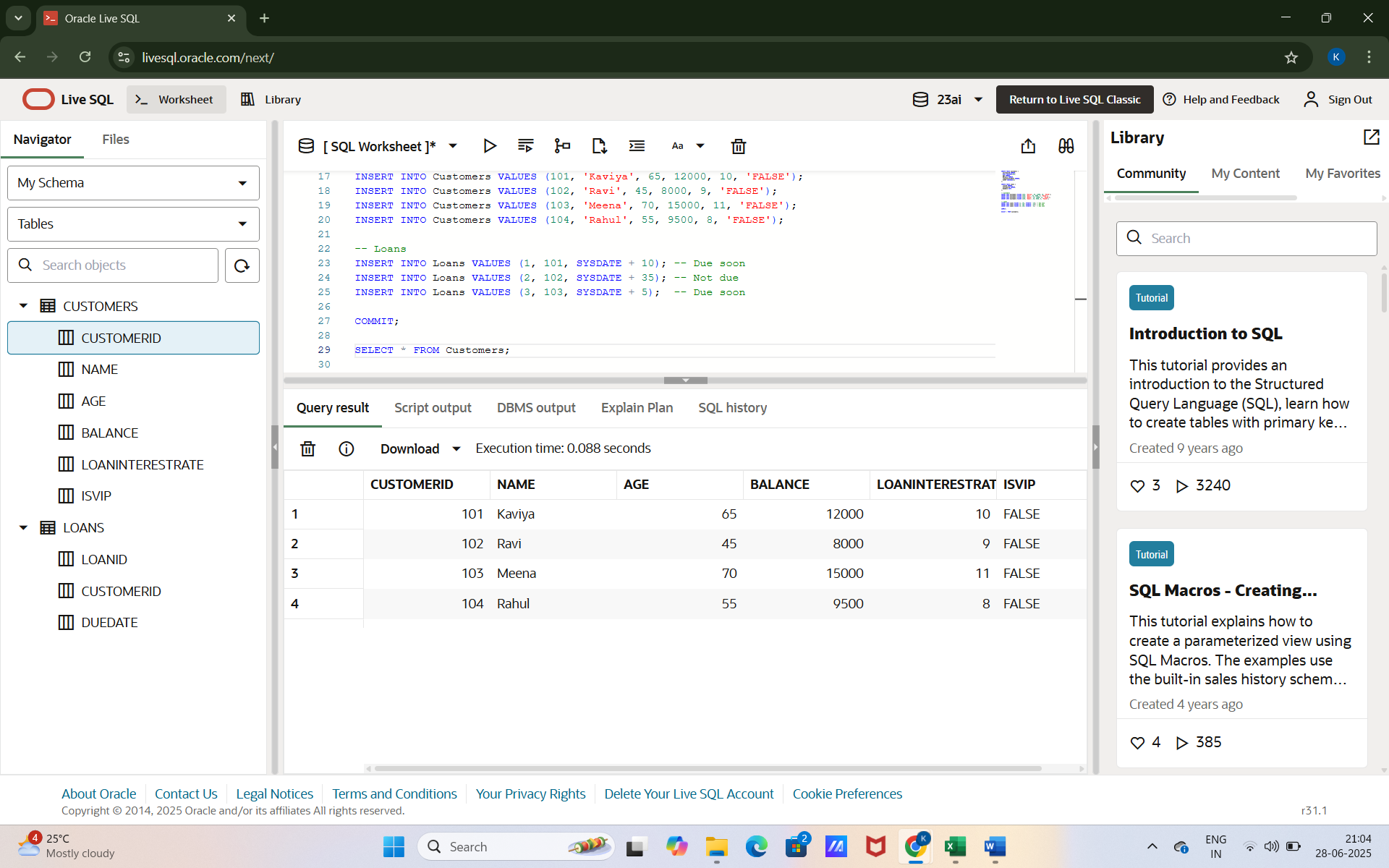
COMMIT;

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**Step 3: Tables created:**



A screenshot of a computer

AI-generated content may be incorrect.

**Scenario 1: Senior Citizen Loan Discount**

* Apply a 1% discount to loan interest rates for customers above 60 years old.

**CODE:**

BEGIN

  FOR cust IN (SELECT CustomerID, Age FROM Customers) LOOP

    IF cust.Age > 60 THEN

      UPDATE Customers

      SET LoanInterestRate = LoanInterestRate - 1

      WHERE CustomerID = cust.CustomerID;

    END IF;

  END LOOP;

  COMMIT;

END;

SELECT \* FROM Customers;

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**Scenario 2: VIP Status Based on Balance**

* Set IsVIP = TRUE for customers whose balance is over $10,000.

**CODE:**

BEGIN

  FOR cust IN (SELECT CustomerID, Balance FROM Customers) LOOP

    IF cust.Balance > 10000 THEN

      UPDATE Customers

      SET IsVIP = 'TRUE'

      WHERE CustomerID = cust.CustomerID;

    END IF;

  END LOOP;

  COMMIT;

END;

SELECT \* FROM Customers;

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**Scenario 3: Loan Reminder for Due Dates**

* Print reminders for loans due in the next 30 days.

**CODE:**

BEGIN

  FOR loan\_rec IN (

    SELECT l.LoanID, c.Name, l.DueDate

    FROM Loans l

    JOIN Customers c ON l.CustomerID = c.CustomerID

    WHERE l.DueDate BETWEEN SYSDATE AND SYSDATE + 30

  ) LOOP

    DBMS\_OUTPUT.PUT\_LINE('Reminder: Dear ' || loan\_rec.Name ||

                         ', your loan ID ' || loan\_rec.LoanID ||

                         ' is due on ' || TO\_CHAR(loan\_rec.DueDate, 'DD-MON-YYYY'));

  END LOOP;

END;

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**Exercise 3: Stored Procedures**

**Scenario 1: Monthly Interest for Savings Accounts**

* The bank needs to process monthly interest for all savings accounts. Apply 1% interest to each balance.

**Step 1: Create table**

**CODE:**

CREATE TABLE SavingsAccounts (

  AccountID NUMBER,

  CustomerName VARCHAR2(50),

  Balance NUMBER

);

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**Step 2 : Insert values**

**CODE:**

INSERT INTO SavingsAccounts VALUES (201, 'Kaviya', 10000);

INSERT INTO SavingsAccounts VALUES (202, 'Meena', 15000);

INSERT INTO SavingsAccounts VALUES (203, 'Ravi', 12000);

COMMIT;

SELECT \* FROM SavingsAccounts;

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**Step 3 : Create stored procedure**

**CODE:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

  FOR acc IN (SELECT AccountID, Balance FROM SavingsAccounts) LOOP

    UPDATE SavingsAccounts

    SET Balance = Balance + (Balance \* 0.01)

    WHERE AccountID = acc.AccountID;

  END LOOP;

  COMMIT;

END;

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**Step 4 : Execute the procedure**

**CODE:**

BEGIN

  ProcessMonthlyInterest;

END;

SELECT \* FROM SavingsAccounts;

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

A computer screen shot of a computer screen

AI-generated content may be incorrect.

**Scenario 2: Bonus Scheme for Employees**

* Add a bonus to all employees in a selected department. Bonus percentage is passed as a parameter.

**Step 1 : Create table**

**CODE:**

CREATE TABLE Employees (

  EmpID NUMBER,

  EmpName VARCHAR2(50),

  Department VARCHAR2(50),

  Salary NUMBER

);

**OUTPUT:**

A computer screen shot of a computer screen

AI-generated content may be incorrect.

**Step 2 : Insert values**

**CODE:**

INSERT INTO Employees VALUES (301, 'Anand', 'HR', 40000);

INSERT INTO Employees VALUES (302, 'Priya', 'Sales', 45000);

INSERT INTO Employees VALUES (303, 'Vikram', 'Sales', 47000);

COMMIT;

SELECT \* FROM Employees;

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**Step 3 : Create a stored procedure**

**CODE:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

  p\_Department IN VARCHAR2,

  p\_BonusPercent IN NUMBER

) IS

BEGIN

  UPDATE Employees

  SET Salary = Salary + (Salary \* p\_BonusPercent / 100)

  WHERE Department = p\_Department;

  COMMIT;

END;

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**Step 4 : Execute the procedure**

**CODE:**

BEGIN

  UpdateEmployeeBonus('Sales', 10);

END;

SELECT \* FROM Employees;

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Scenario 3: Transfer Funds Between Accounts**

* Allow customers to transfer funds between accounts after checking for sufficient balance.

**Step 1 : Create table**

**CODE:**

CREATE TABLE Accounts (

  AccountID NUMBER,

  CustomerName VARCHAR2(50),

  Balance NUMBER

);

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**Step 2 : Insert values**

**CODE:**

INSERT INTO Accounts VALUES (401, 'Kaviya', 15000);

INSERT INTO Accounts VALUES (402, 'Satheesh', 10000);

COMMIT;

SELECT \* FROM Accounts;

**OUTPUT:**

A computer screen shot of a computer screen

AI-generated content may be incorrect.

**Step 3 : Create a stored procedure**

**CODE:**

CREATE OR REPLACE PROCEDURE TransferFunds (

  p\_FromAccountID IN NUMBER,

  p\_ToAccountID IN NUMBER,

  p\_Amount IN NUMBER

) IS

  v\_FromBalance NUMBER;

BEGIN

  SELECT Balance INTO v\_FromBalance

  FROM Accounts

  WHERE AccountID = p\_FromAccountID;

  IF v\_FromBalance >= p\_Amount THEN

    UPDATE Accounts

    SET Balance = Balance - p\_Amount

    WHERE AccountID = p\_FromAccountID;

    UPDATE Accounts

    SET Balance = Balance + p\_Amount

    WHERE AccountID = p\_ToAccountID;

    COMMIT;

  ELSE

    DBMS\_OUTPUT.PUT\_LINE('Insufficient balance for transfer.');

  END IF;

END;

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**Step 4 : Execute the stored procedure**

**CODE:**

BEGIN

  TransferFunds(401, 402, 5000);

END;

SELECT \* FROM Accounts;

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.