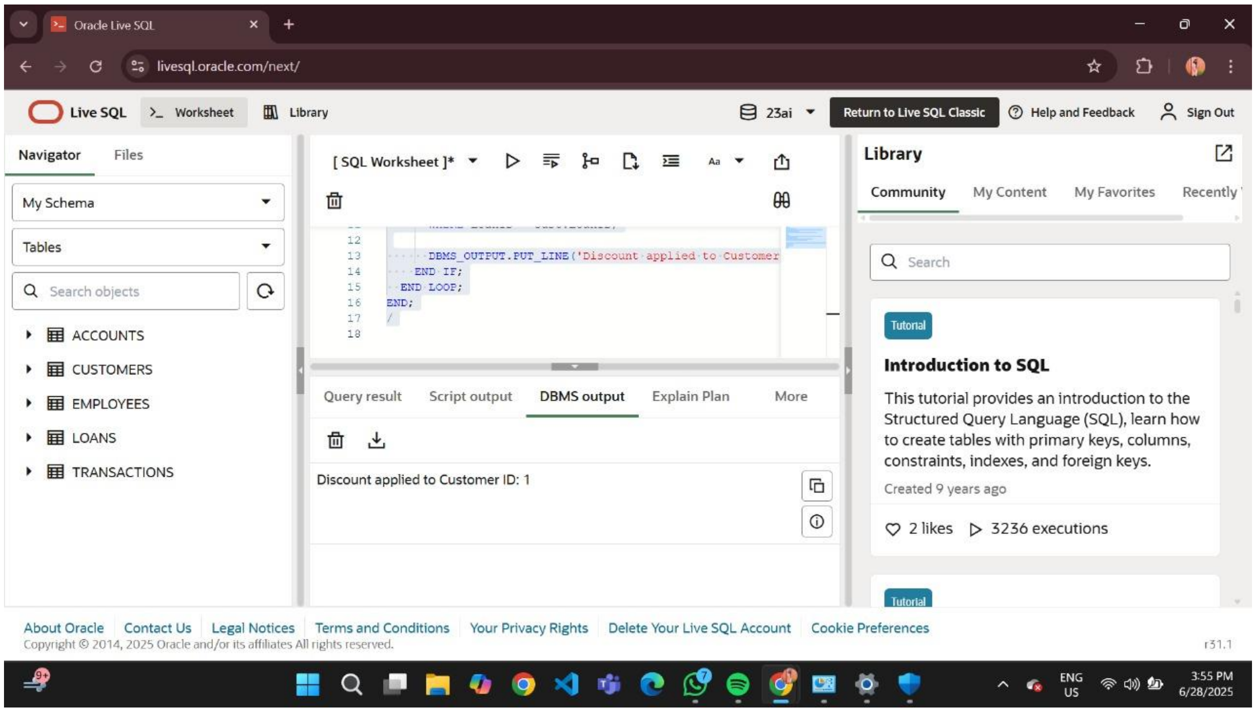
PL/SQL Programming:

Exercise 1: Control Structures

Scenario 1: Apply 1% Interest Discount for Customers Over 60

```
BEGIN
 FOR cust IN (
   SELECT c.CustomerID, l.LoanID, l.InterestRate, c.DOB
   FROM Customers c
   JOIN Loans l ON c.CustomerID = l.CustomerID
 )
 LOOP
   IF MONTHS_BETWEEN(SYSDATE, cust.DOB) / 12 > 60 THEN
     UPDATE Loans
     SET InterestRate = InterestRate - 1
     WHERE LoanID = cust.LoanID;

     DBMS_OUTPUT.PUT_LINE('Discount applied to Customer ID: ' || cust.CustomerID);
   END IF;
 END LOOP;
END;
/
```

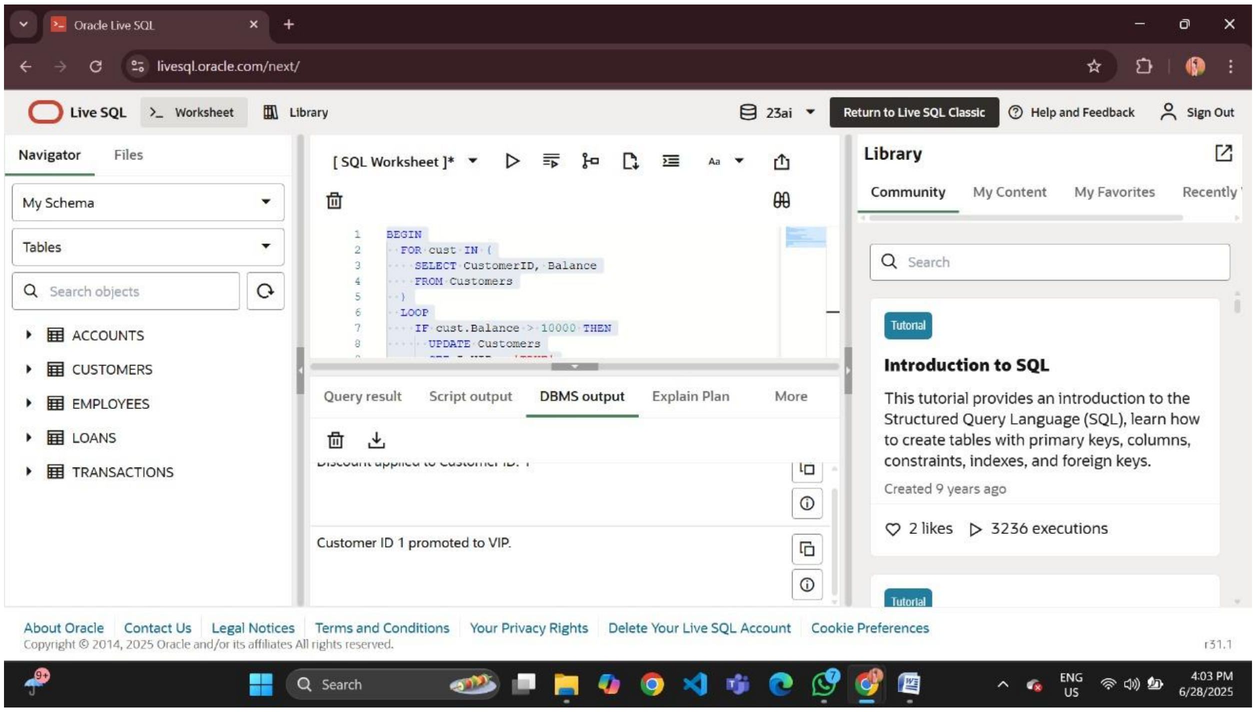


```
            DBMS_OUTPUT.PUT_LINE('Discount applied to Customer
        END IF;
    END LOOP;
END;
/
```

```
Discount applied to Customer ID: 1
```

Scenario 2: Promote Customers to VIP Based on Balance

```
BEGIN
  FOR cust IN (
    SELECT CustomerID, Balance
    FROM Customers
  )
  LOOP
    IF cust.Balance > 10000 THEN
      UPDATE Customers
      SET IsVIP = 'TRUE'
      WHERE CustomerID = cust.CustomerID;
      DBMS_OUTPUT.PUT_LINE('Customer ID ' || cust.CustomerID || ' promoted to VIP.');
    END IF;
  END LOOP;
END;
/
```

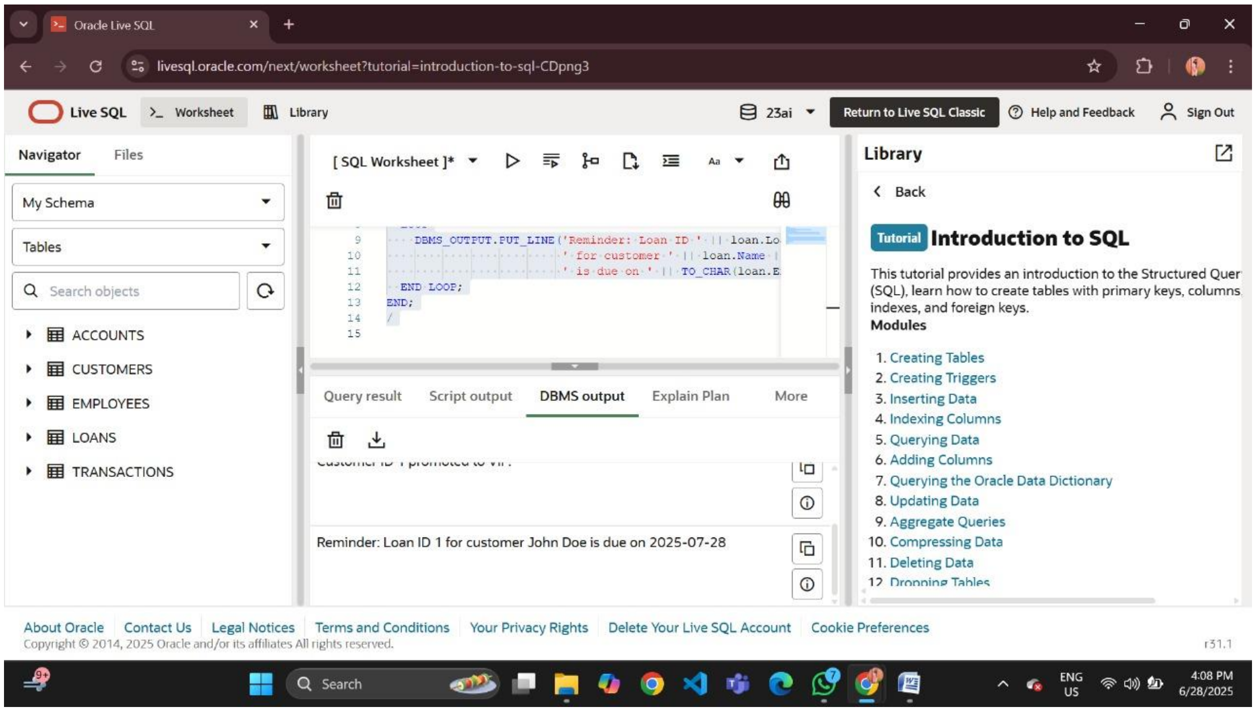


```
BEGIN
  FOR cust IN (
    SELECT CustomerID, Balance
    FROM Customers
  )
  LOOP
    IF cust.Balance > 10000 THEN
      UPDATE Customers
```

```
Discount applied to Customer ID: 1
Customer ID 1 promoted to VIP.
```

Scenario 3: Send Loan Due Reminders (Next 30 Days)

```
BEGIN
  FOR loan IN (
    SELECT c.Name, l.LoanID, l.EndDate
    FROM Loans l
    JOIN Customers c ON l.CustomerID = c.CustomerID
  WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30
 )
 LOOP
  DBMS_OUTPUT.PUT_LINE('Reminder: Loan ID ' || loan.LoanID ||
                       ' for customer ' || loan.Name ||
                       ' is due on ' || TO_CHAR(loan.EndDate, 'YYYY-MM-DD'));
 END LOOP;
END;
/
```



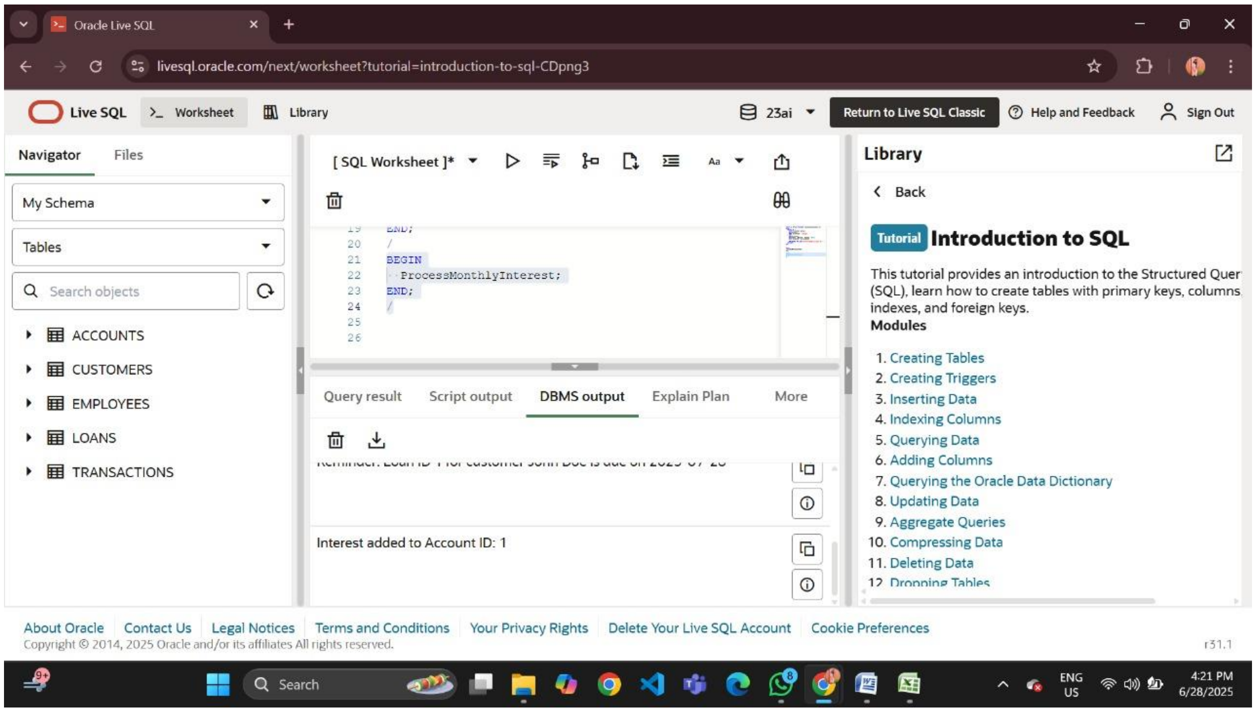
```
    DBMS_OUTPUT.PUT_LINE('Reminder: Loan ID ' || loan.Lo
                         ' for customer ' || loan.Name |
                         ' is due on ' || TO_CHAR(loan.E
  END LOOP;
END;
/
```

```
Reminder: Loan ID 1 for customer John Doe is due on 2025-07-28
```

Exercise 3: Stored Procedures

Scenario 1: Monthly Interest for Savings Accounts

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

    DBMS_OUTPUT.PUT_LINE('Interest added to Account ID: ' || acc.AccountID);
  END LOOP;
END;
/
BEGIN
  ProcessMonthlyInterest;
END;
/
```



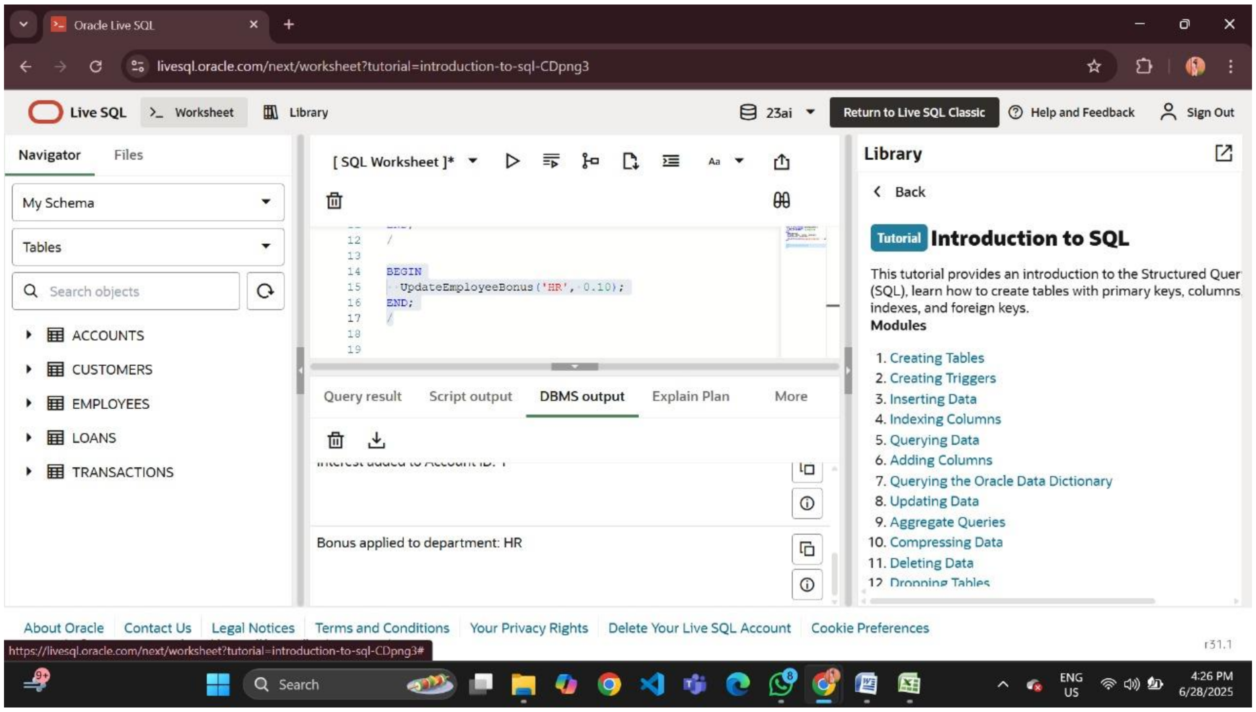
```
END;
/
BEGIN
  ProcessMonthlyInterest;
END;
/
```

```
Reminder: Loan ID 1 for customer John Doe is due on 2025-07-28

Interest added to Account ID: 1
```

Scenario 2: Bonus for Employees by Department

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
  p_department IN VARCHAR2,
  p_bonus_pct IN NUMBER  -- e.g. 0.10 for 10%
) IS
BEGIN
  UPDATE Employees
  SET Salary = Salary + (Salary * p_bonus_pct)
  WHERE Department = p_department;


  DBMS_OUTPUT.PUT_LINE('Bonus applied to department: ' || p_department);
```

```
END;
/
BEGIN
  UpdateEmployeeBonus('HR', 0.10);
END;
/
```



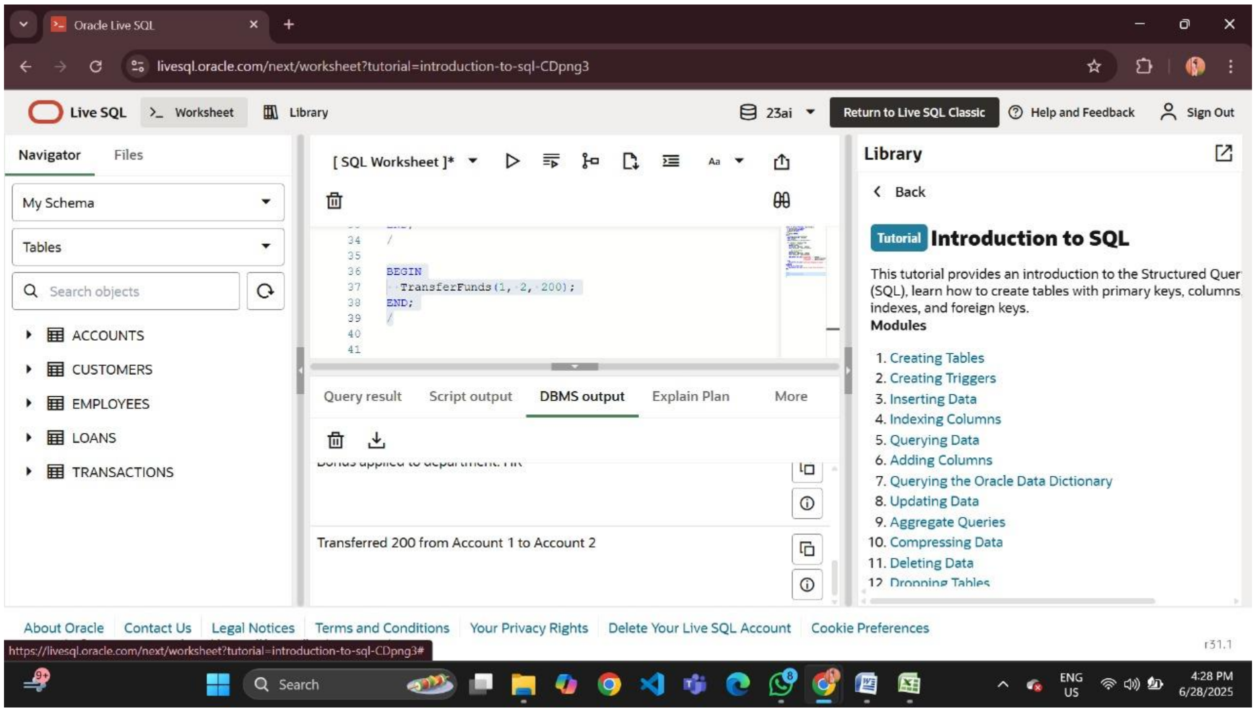
```
/

BEGIN
  UpdateEmployeeBonus('HR', 0.10);
END;
/
```

```
Bonus applied to department: HR
```

Scenario 3: Fund Transfer Between Accounts

```
CREATE OR REPLACE PROCEDURE TransferFunds (
  p_source_account IN NUMBER,
  p_dest_account IN NUMBER,
  p_amount IN NUMBER
) IS
  v_balance NUMBER;
BEGIN
  -- Check source account balance
  SELECT Balance INTO v_balance
  FROM Accounts
  WHERE AccountID = p_source_account;

  IF v_balance >= p_amount THEN
    -- Deduct from source
    UPDATE Accounts
    SET Balance = Balance - p_amount
    WHERE AccountID = p_source_account;

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

    DBMS_OUTPUT.PUT_LINE('Transferred ' || p_amount ||
                         ' from Account ' || p_source_account ||
                         ' to Account ' || p_dest_account);
  ELSE
```

```
    DBMS_OUTPUT.PUT_LINE('Insufficient balance in Account ' || p_source_account);
  END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Invalid Account ID(s) provided.');
END;
/
BEGIN
  TransferFunds(1, 2, 200);
END;
/
```



```
/

BEGIN
 TransferFunds(1, 2, 200);
END;
/
```

This tutorial provides an introduction to the Structured Quer (SQL), learn how to create tables with primary keys, columns indexes, and foreign keys.

```
Transferred 200 from Account 1 to Account 2
```