**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.

Solution:

SELECT c.CustomerID,

       c.Name,

       TRUNC(MONTHS\_BETWEEN(SYSDATE, c.DOB) / 12) AS Age,

       l.LoanID,

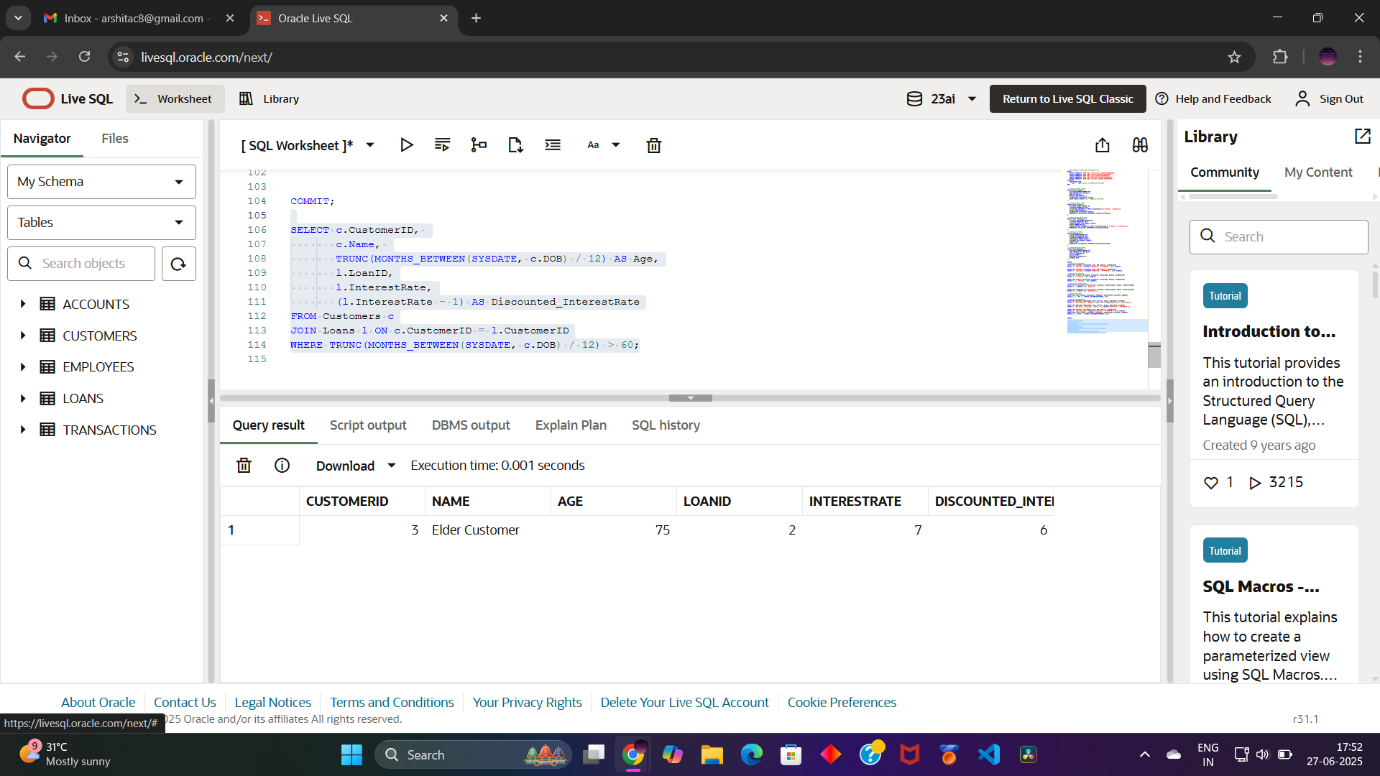
       l.InterestRate,

       (l.InterestRate - 1) AS Discounted\_InterestRate

FROM Customers c

JOIN Loans l ON c.CustomerID = l.CustomerID

WHERE TRUNC(MONTHS\_BETWEEN(SYSDATE, c.DOB) / 12) > 60;



* **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.

Solution:

SELECT CustomerID, Name, Balance,

       CASE

           WHEN Balance > 10000 THEN 'Y'

           ELSE 'N'

       END AS VIP\_Status

FROM Customers;

UPDATE Customers

SET IsVIP = CASE

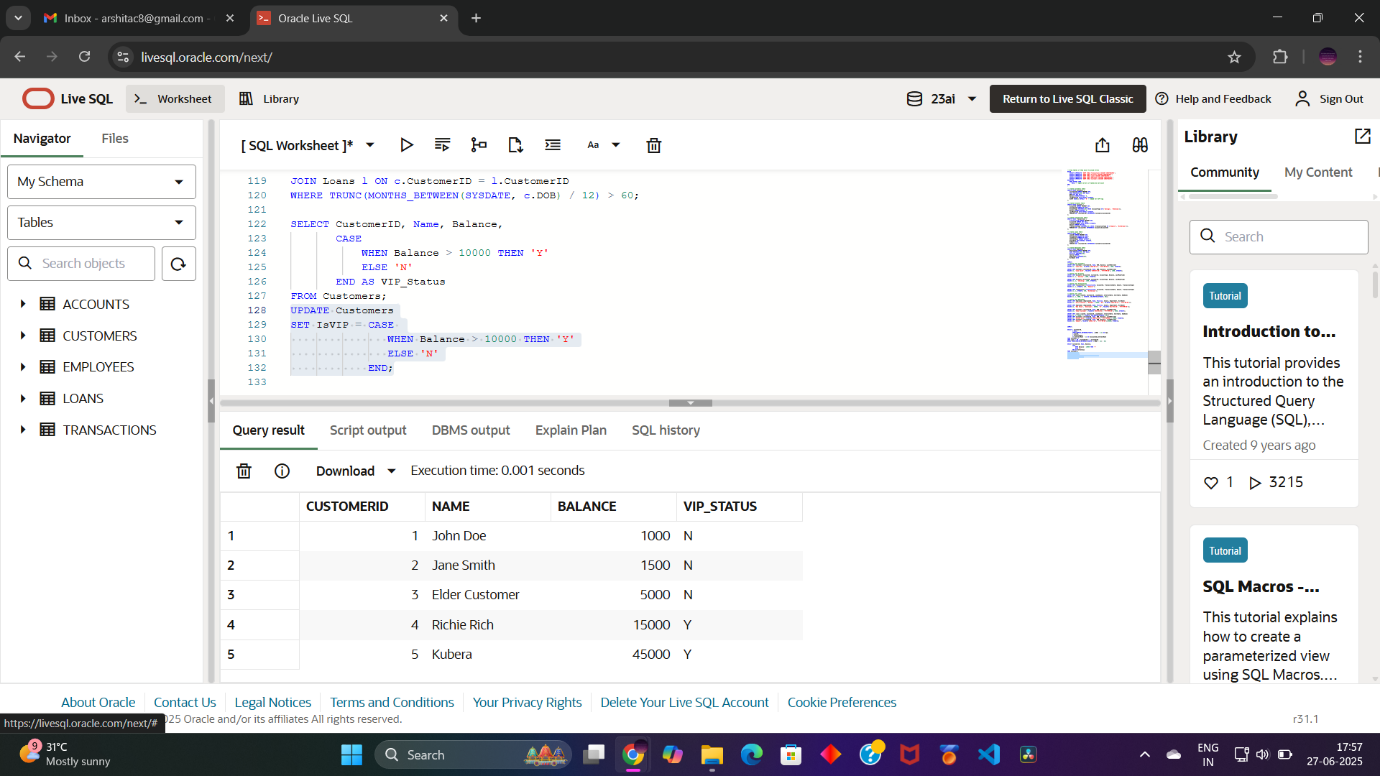
               WHEN Balance > 10000 THEN 'Y'

               ELSE 'N'

            END;

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (3, 1, 8000, 6, SYSDATE, SYSDATE + 15);



* **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.

Solution:

SELECT l.LoanID,

       l.CustomerID,

       c.Name,

       TO\_CHAR(l.EndDate, 'DD-MON-YYYY') AS Due\_Date,

       'Reminder: Loan ID ' || l.LoanID ||

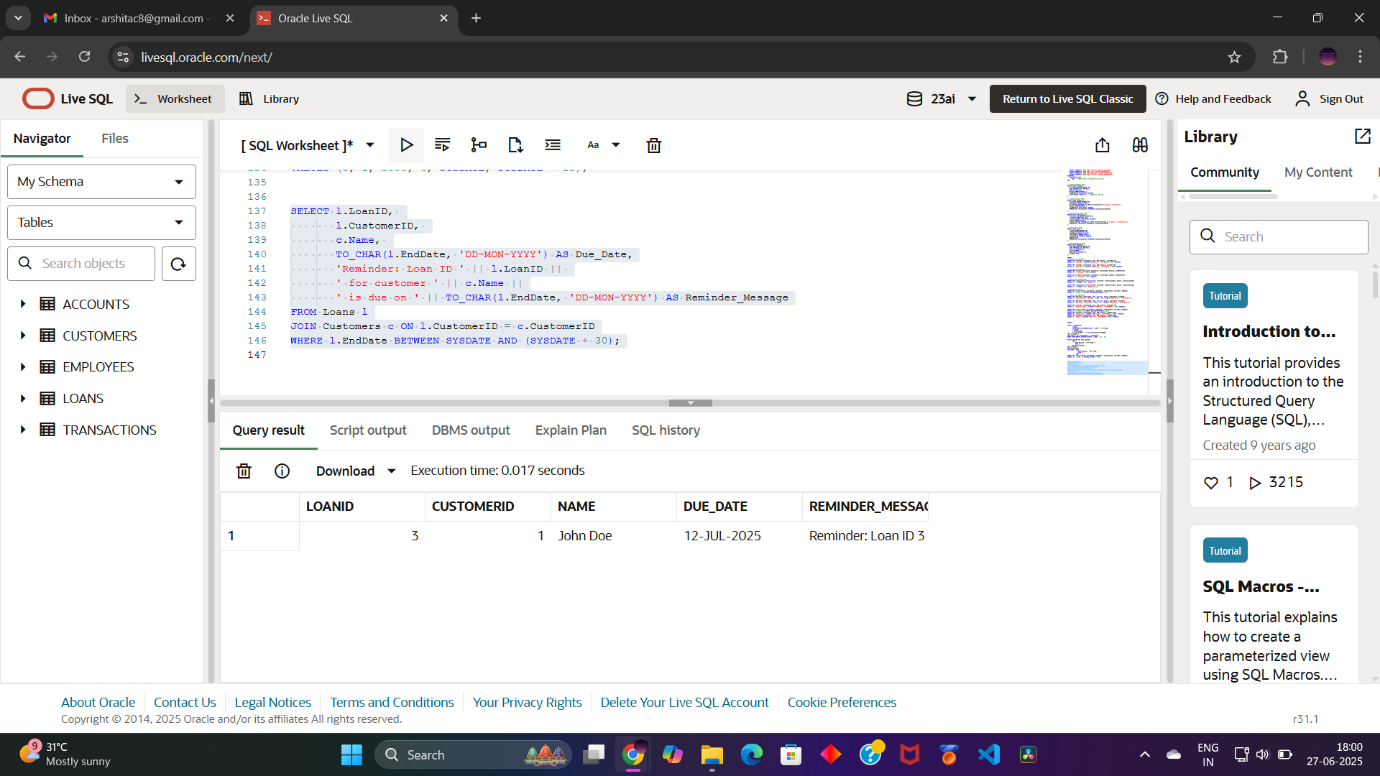
       ' for customer ' || c.Name ||

       ' is due on ' || TO\_CHAR(l.EndDate, 'DD-MON-YYYY') AS Reminder\_Message

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate BETWEEN SYSDATE AND (SYSDATE + 30);



**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.

Solution:

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

   UPDATE Accounts

   SET Balance = Balance + (Balance \* 0.01) -- Apply 1% interest

   WHERE AccountType = 'Savings';

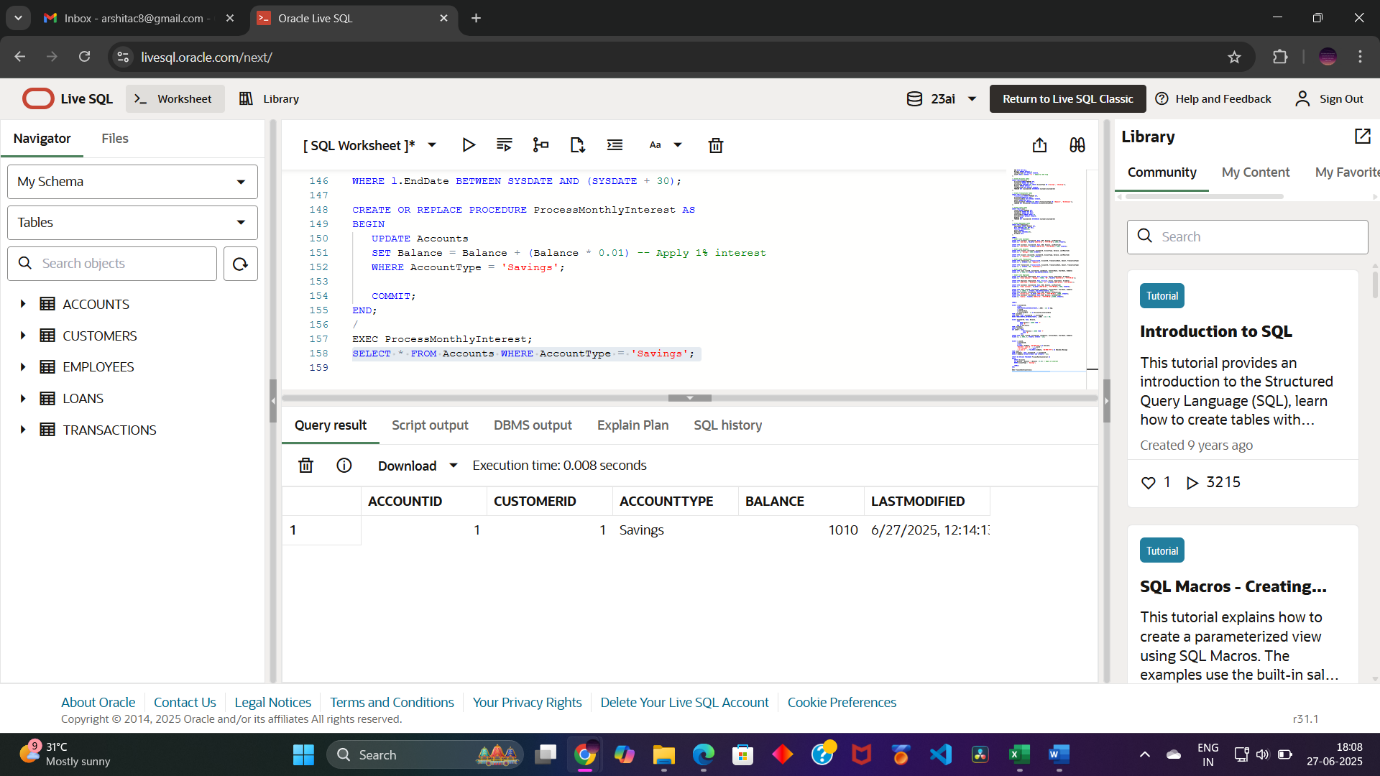
   COMMIT;

END;

/

EXEC ProcessMonthlyInterest;

SELECT \* FROM Accounts WHERE AccountType = 'Savings';



* **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.

Solution:

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

   p\_Department IN VARCHAR2,

   p\_BonusPercent IN NUMBER

) AS

BEGIN

   UPDATE Employees

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

   WHERE Department = p\_Department;

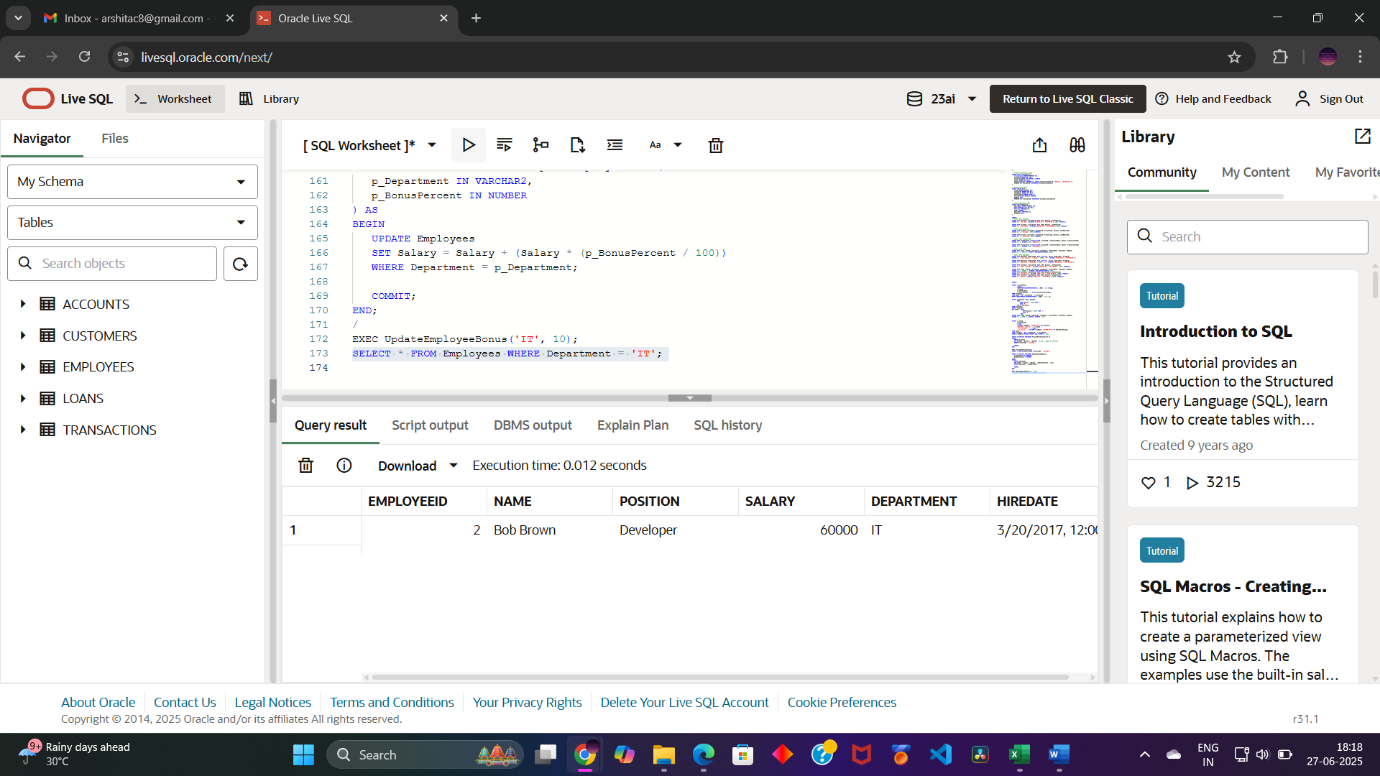
   COMMIT;

END;

/

EXEC UpdateEmployeeBonus('IT', 10);

SELECT \* FROM Employees WHERE Department = 'IT';



* **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.

Solution:

CREATE OR REPLACE PROCEDURE TransferFunds(

   p\_FromAccountID IN NUMBER,

   p\_ToAccountID IN NUMBER,

   p\_Amount IN NUMBER

) AS

   v\_FromBalance NUMBER;

BEGIN

   -- Check if source account has sufficient balance

   SELECT Balance INTO v\_FromBalance

   FROM Accounts

   WHERE AccountID = p\_FromAccountID;

   IF v\_FromBalance < p\_Amount THEN

      RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds in source account.');

   END IF;

   -- Deduct from source account

   UPDATE Accounts

   SET Balance = Balance - p\_Amount

   WHERE AccountID = p\_FromAccountID;

   -- Add to target account

   UPDATE Accounts

   SET Balance = Balance + p\_Amount

   WHERE AccountID = p\_ToAccountID;

   COMMIT;

END;

/

EXEC TransferFunds(1, 2, 500);

SELECT AccountID, CustomerID, AccountType, Balance FROM Accounts;

