**PL/SQL Exercises :**

**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.

**PL/SQL QUERY :**

BEGIN

FOR rec IN (

SELECT customer\_id

FROM customers

WHERE age > 60

) LOOP

UPDATE loans

SET interest\_rate = interest\_rate - 0.01

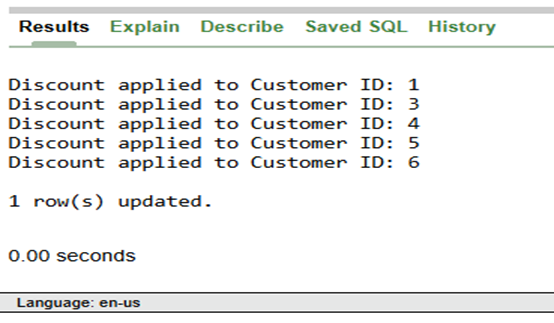
WHERE customer\_id = rec.customer\_id;

DBMS\_OUTPUT.PUT\_LINE('Discount applied to Customer ID: ' || rec.customer\_id);

END LOOP;

END;

**OUTPUT :**

****

**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 QUERY :**

BEGIN

FOR rec IN (

SELECT customer\_id

FROM customers

WHERE balance > 10000

) LOOP

UPDATE customers

SET IsVIP = 'TRUE'

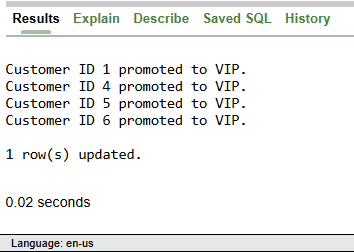
WHERE customer\_id = rec.customer\_id;

DBMS\_OUTPUT.PUT\_LINE('Customer ID ' || rec.customer\_id || ' promoted to VIP.');

END LOOP;

END;

**OUTPUT :**

****

**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.

**PL/SQL QUERY :**

BEGIN

FOR rec IN (

SELECT l.loan\_id, c.customer\_id, c.name, l.due\_date

FROM loans l

JOIN customers c ON l.customer\_id = c.customer\_id

WHERE l.due\_date BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || rec.loan\_id ||

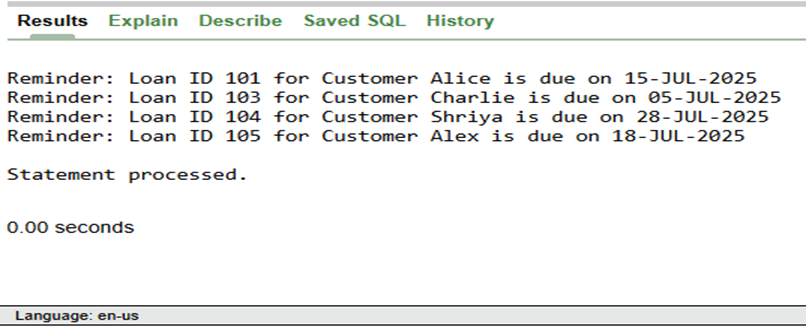
' for Customer ' || rec.name ||

' is due on ' || TO\_CHAR(rec.due\_date, 'DD-MON-YYYY'));

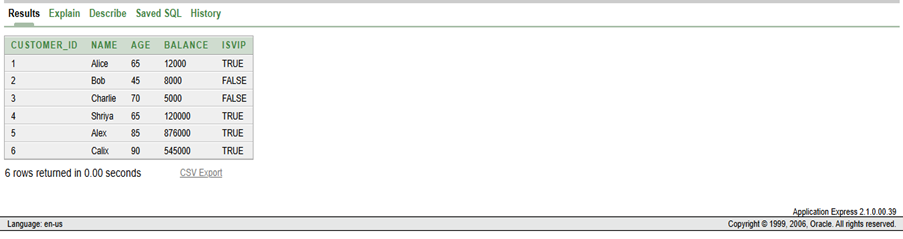
END LOOP;

END;

**OUTPUT :**

****

**SELECT \* FROM customers;**

****

**SELECT \* FROM loans;**

****

**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.

**PL/SQL Query :**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR rec IN (

SELECT AccountID, Balance

FROM Accounts1

WHERE AccountType = 'Savings'

) LOOP

UPDATE Accounts1

SET Balance = Balance + (Balance \* 0.01)

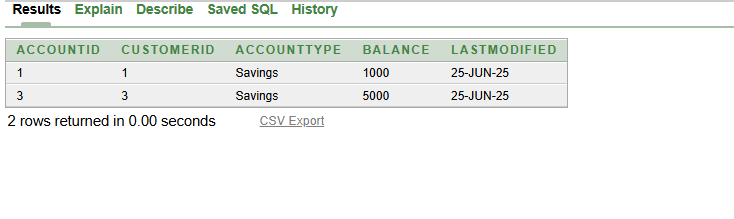
WHERE AccountID = rec.AccountID;

END LOOP;

END;

/

**OUTPUT :**

****

**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.

**PL/SQL Query :**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_Department IN VARCHAR2,

p\_BonusPercent IN NUMBER

) IS

BEGIN

UPDATE Employees2

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

WHERE Department = p\_Department;

END;

/

**Output :**



**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.

**PL/SQL Query :**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_SourceAccountID IN INT,

p\_TargetAccountID IN INT,

p\_Amount IN INT

) IS

v\_SourceBalance INT;

BEGIN

SELECT Balance INTO v\_SourceBalance

FROM Accounts1

WHERE AccountID = p\_SourceAccountID FOR UPDATE;

IF v\_SourceBalance >= p\_Amount THEN

UPDATE Accounts1

SET Balance = Balance - p\_Amount

WHERE AccountID = p\_SourceAccountID;

UPDATE Accounts1

SET Balance = Balance + p\_Amount

WHERE AccountID = p\_TargetAccountID;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Insufficient funds. Transfer failed.');

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('One of the accounts does not exist.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;

/

BEGIN

ProcessMonthlyInterest;

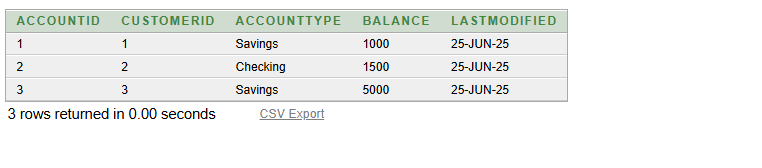
UpdateEmployeeBonus('IT', 10);

TransferFunds(1, 2, 300);

END;

/

**Output :**

****