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

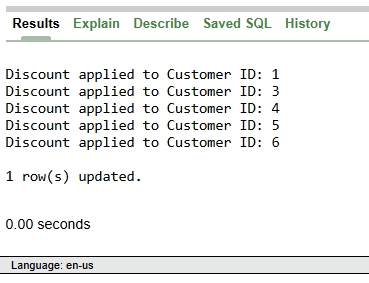
WHERE customer\_id = rec.customer\_id;

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'

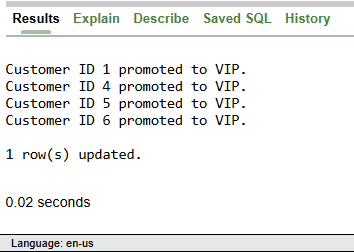
WHERE customer\_id = rec.customer\_id;

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 ||

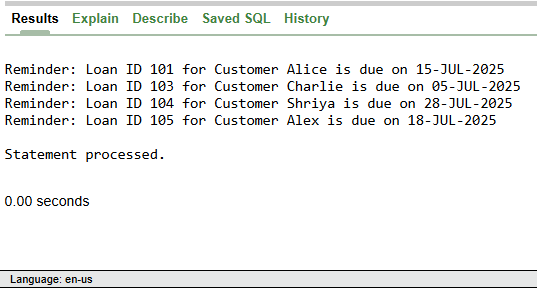
' for Customer ' || rec.name ||

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

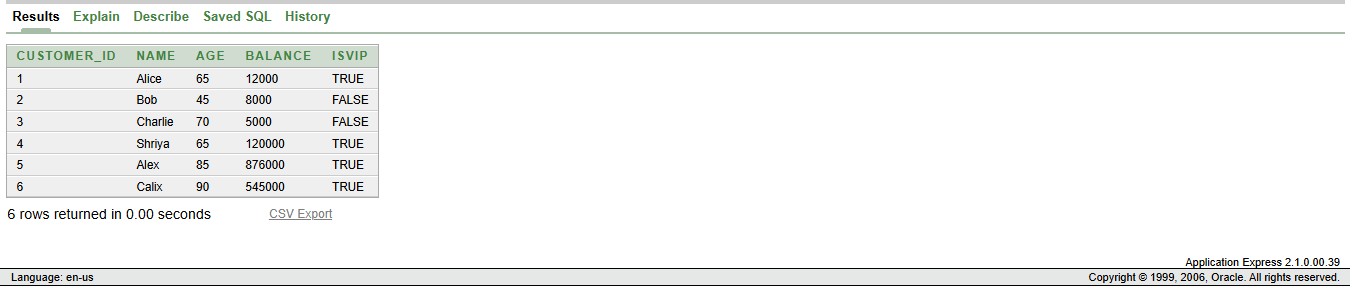
END LOOP;

END;

**OUTPUT :**

****

**SELECT \* FROM customers;**

****

**SELECT \* FROM loans;**

****

PL/SQL Exercises

**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

UPDATE savings\_account

SET balance = balance + (balance \* 0.01);

COMMIT;

END;

/

BEGIN

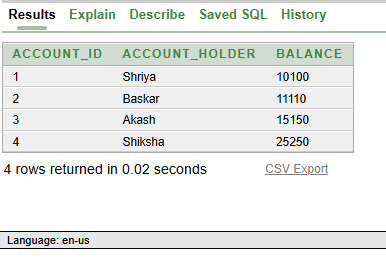
ProcessMonthlyInterest;

END;

/

SELECT \* FROM savings\_account;

**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\_id IN NUMBER,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE employees\_

SET salary = salary + (salary \* (p\_bonus\_percent / 100))

WHERE department\_id = p\_department\_id;

END;

/

BEGIN

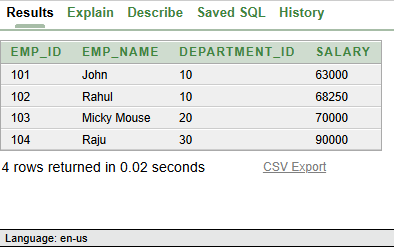
UpdateEmployeeBonus(10, 5); -- Example: Add 5% bonus to dept 10

END;

/

SELECT \* FROM employees\_;

**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\_from\_account\_id IN NUMBER,

p\_to\_account\_id IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

-- Get balance of source account

SELECT balance INTO v\_balance

FROM accounts\_

WHERE account\_id = p\_from\_account\_id;

-- Check sufficient balance

IF v\_balance < 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 account\_id = p\_from\_account\_id;

-- Add to target account

UPDATE accounts\_

SET balance = balance + p\_amount

WHERE account\_id = p\_to\_account\_id;

COMMIT;

END;

/

BEGIN

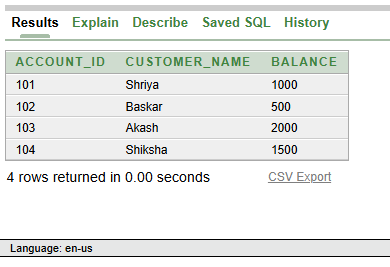
TransferFunds(101, 102, 500);

END;

/

SELECT \* FROM accounts\_\_;

**OUTPUT :**

****