Exercise 1: Control Structures

Scenario 1: The bank wants to apply a discount to loan interest rates for customers above 60 years old.

o 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.

Scenario 2: A customer can be promoted to VIP status based on their balance.

o 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.

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.

Answer :

Scenario 1: Apply Discount to Customers Above 60

-- Create customers table

CREATE TABLE customers (

customer\_id INT PRIMARY KEY,

name VARCHAR(100),

age INT,

balance DECIMAL(10,2),

is\_vip VARCHAR(5) DEFAULT 'FALSE'

);

-- Create loans table

CREATE TABLE loans (

loan\_id INT PRIMARY KEY,

customer\_id INT,

interest\_rate DECIMAL(5,2),

due\_date DATE,

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

-- Insert data into customers

INSERT INTO customers (customer\_id, name, age, balance) VALUES (1, 'John Doe', 65, 12000);

INSERT INTO customers (customer\_id, name, age, balance) VALUES (2, 'Jane Smith', 45, 8000);

INSERT INTO customers (customer\_id, name, age, balance) VALUES (3, 'Alice Johnson', 70, 15000);

INSERT INTO customers (customer\_id, name, age, balance) VALUES (4, 'Bob Brown', 30, 10500);

INSERT INTO customers (customer\_id, name, age, balance) VALUES (5, 'Carol White', 62, 9500);

-- Insert data into loans

INSERT INTO loans (loan\_id, customer\_id, interest\_rate, due\_date)

VALUES (101, 1, 8.5, DATE\_ADD(CURDATE(), INTERVAL 10 DAY));

INSERT INTO loans (loan\_id, customer\_id, interest\_rate, due\_date)

VALUES (102, 2, 9.0, DATE\_ADD(CURDATE(), INTERVAL 45 DAY));

INSERT INTO loans (loan\_id, customer\_id, interest\_rate, due\_date)

VALUES (103, 3, 7.5, DATE\_ADD(CURDATE(), INTERVAL 5 DAY));

INSERT INTO loans (loan\_id, customer\_id, interest\_rate, due\_date)

VALUES (104, 4, 6.8, DATE\_ADD(CURDATE(), INTERVAL 29 DAY));

INSERT INTO loans (loan\_id, customer\_id, interest\_rate, due\_date)

VALUES (105, 5, 8.0, DATE\_ADD(CURDATE(), INTERVAL -1 DAY));

UPDATE loans

JOIN customers ON loans.customer\_id = customers.customer\_id

SET loans.interest\_rate = loans.interest\_rate - 1

WHERE customers.age > 60;

SELECT c.name, c.age, l.loan\_id, l.interest\_rate

FROM customers c

JOIN loans l ON c.customer\_id = l.customer\_id;

A screenshot of a computer

AI-generated content may be incorrect.

**Scenario 2: Promote Customers to VIP Based on Balance**

UPDATE customers

SET is\_vip = 'TRUE'

WHERE balance > 10000;

SELECT customer\_id, name, balance, is\_vip FROM customers;

A screenshot of a computer

AI-generated content may be incorrect.

**Scenario 3: Send Reminders for Loans Due in Next 30 Days**

SELECT CONCAT(

'Reminder: Dear ', c.name,

', your loan (ID: ', l.loan\_id,

') is due on ', DATE\_FORMAT(l.due\_date, '%d-%b-%Y')

) AS Reminder\_Message

FROM loans l

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

WHERE l.due\_date <= DATE\_ADD(CURDATE(), INTERVAL 30 DAY);

A screenshot of a computer

AI-generated content may be incorrect.

**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.

**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.

**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.

**Scenario 1:** The bank needs to process monthly interest for all savings accounts.

-- Create savings\_accounts table

CREATE TABLE savings\_accounts (

account\_id INT PRIMARY KEY,

customer\_id INT,

balance DECIMAL(10,2)

);

-- Insert sample data

INSERT INTO savings\_accounts VALUES (101, 1, 1000.00);

INSERT INTO savings\_accounts VALUES (102, 2, 2000.00);

INSERT INTO savings\_accounts VALUES (103, 3, 3000.00);

-- Create employees table

CREATE TABLE employees (

emp\_id INT PRIMARY KEY,

name VARCHAR(100),

department\_id INT,

salary DECIMAL(10,2)

);

-- Insert sample data

INSERT INTO employees VALUES (1, 'Alice', 10, 50000.00);

INSERT INTO employees VALUES (2, 'Bob', 20, 55000.00);

INSERT INTO employees VALUES (3, 'Charlie', 10, 60000.00);

-- Create accounts table

CREATE TABLE accounts (

account\_id INT PRIMARY KEY,

customer\_id INT,

balance DECIMAL(10,2)

);

-- Insert sample data

INSERT INTO accounts VALUES (201, 1, 5000.00);

INSERT INTO accounts VALUES (202, 1, 1000.00);

INSERT INTO accounts VALUES (203, 2, 8000.00);

-- Create procedure to process monthly interest

DELIMITER //

CREATE PROCEDURE ProcessMonthlyInterest()

BEGIN

UPDATE savings\_accounts

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

END;

//

DELIMITER ;

-- Call the procedure

CALL ProcessMonthlyInterest();

-- View the updated balances

SELECT \* FROM savings\_accounts;

A white rectangular object with black lines

AI-generated content may be incorrect.

**Scenario 2:**

The bank wants to implement a bonus scheme for employees based on their performance.

**Code :**

DELIMITER //

CREATE PROCEDURE UpdateEmployeeBonus (

IN p\_dept\_id INT,

IN p\_bonus\_pct DECIMAL(5,2) -- For example, pass 10 for 10%

)

BEGIN

UPDATE employees

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

WHERE department\_id = p\_dept\_id;

END;

//

DELIMITER ;

CALL UpdateEmployeeBonus(10, 10); -- Give 10% bonus to department 10

SELECT \* FROM employees;

**Output :**

A screenshot of a computer

AI-generated content may be incorrect.

**Scenario 3:**

Customers should be able to transfer funds between their accounts.

**Code :**

CREATE PROCEDURE TransferFunds (

IN p\_from\_account INT,

IN p\_to\_account INT,

IN p\_amount DECIMAL(10,2)

)

BEGIN

DECLARE v\_balance DECIMAL(10,2);

-- Get the balance of the source account

SELECT balance INTO v\_balance

FROM accounts

WHERE account\_id = p\_from\_account;

-- Check for sufficient funds

IF v\_balance < p\_amount THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Insufficient funds';

END IF;

-- Deduct from source

UPDATE accounts

SET balance = balance - p\_amount

WHERE account\_id = p\_from\_account;

-- Add to destination

UPDATE accounts

SET balance = balance + p\_amount

WHERE account\_id = p\_to\_account;

END;

//

DELIMITER ;

CALL TransferFunds(201, 202, 1000.00);

SELECT \* FROM accounts;

**Output :**

A white rectangular object with black lines

AI-generated content may be incorrect.