**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.

**Code :**

DROP TABLE IF EXISTS loans;

DROP TABLE IF EXISTS customers;

CREATE TABLE customers (

customer\_id INT PRIMARY KEY,

name VARCHAR(50),

age INT,

balance DECIMAL(10,2),

loan\_interest\_rate DECIMAL(5,2),

IsVIP VARCHAR(5)

);

INSERT INTO customers VALUES

(1, 'Alice', 65, 12000.50, 8.5, 'FALSE'),

(2, 'Bob', 45, 9500.00, 9.0, 'FALSE'),

(3, 'Charlie', 70, 20000.00, 7.5, 'FALSE'),

(4, 'Diana', 30, 3000.00, 10.0, 'FALSE'),

(5, 'Edward', 62, 11000.00, 9.5, 'FALSE');

CREATE TABLE loans (

loan\_id INT PRIMARY KEY,

customer\_id INT,

due\_date DATE

);

INSERT INTO loans VALUES

(101, 1, CURDATE() + INTERVAL 10 DAY),

(102, 2, CURDATE() + INTERVAL 35 DAY),

(103, 3, CURDATE() + INTERVAL 5 DAY),

(104, 4, CURDATE() - INTERVAL 1 DAY),

(105, 5, CURDATE() + INTERVAL 20 DAY);

UPDATE customers

SET loan\_interest\_rate = loan\_interest\_rate - 1

WHERE age > 60;

SELECT name, age, loan\_interest\_rate FROM customers;

Output :

A screenshot of a computer

AI-generated content may be incorrect.

**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.

**Code :**

UPDATE customers

SET IsVIP = 'TRUE'

WHERE balance > 10000;

SELECT name, balance, IsVIP FROM customers;

**Output :**

**A screenshot of a computer

AI-generated content may be incorrect.**

**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.

**Code :**

SELECT

CONCAT('Reminder: Loan ID ', l.loan\_id, ' for customer ', c.name,

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

FROM loans l

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

WHERE l.due\_date BETWEEN CURDATE() AND CURDATE() + INTERVAL 30 DAY;

**Output :**

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.

**Code :**

DROP TABLE IF EXISTS accounts;

DROP TABLE IF EXISTS employees;

CREATE TABLE accounts (

account\_id INT PRIMARY KEY,

customer\_name VARCHAR(50),

balance DECIMAL(10,2)

);

INSERT INTO accounts VALUES

(1, 'Alice', 1000.00),

(2, 'Bob', 2000.00),

(3, 'Charlie', 3000.00);

CREATE TABLE employees (

emp\_id INT PRIMARY KEY,

name VARCHAR(50),

department VARCHAR(50),

salary DECIMAL(10,2)

);

INSERT INTO employees VALUES

(1, 'John', 'HR', 50000.00),

(2, 'Jane', 'Sales', 60000.00),

(3, 'Dave', 'Sales', 55000.00),

(4, 'Eve', 'IT', 65000.00);

DELIMITER $$

CREATE PROCEDURE ProcessMonthlyInterest()

BEGIN

UPDATE accounts

SET balance = balance \* 1.01;

END $$

DELIMITER ;

CALL ProcessMonthlyInterest();

SELECT \* FROM accounts;

**Output :**

A screenshot of a computer

AI-generated content may be incorrect.

**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.

**Code :**

DELIMITER $$

CREATE PROCEDURE UpdateEmployeeBonus(

IN dept\_name VARCHAR(50),

IN bonus\_percent DECIMAL(5,2)

)

BEGIN

UPDATE employees

SET salary = salary + (salary \* bonus\_percent / 100)

WHERE department = dept\_name;

END $$

DELIMITER ;

CALL UpdateEmployeeBonus('Sales', 10);

SELECT \* FROM employees;

**Output :**

**A close-up of a number

AI-generated content may be incorrect.**

**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.

**Code :**

DELIMITER $$

CREATE PROCEDURE TransferFunds(

IN from\_acc INT,

IN to\_acc INT,

IN amount DECIMAL(10,2)

)

BEGIN

DECLARE from\_balance DECIMAL(10,2);

SELECT balance INTO from\_balance

FROM accounts

WHERE account\_id = from\_acc;

IF from\_balance >= amount THEN

UPDATE accounts

SET balance = balance - amount

WHERE account\_id = from\_acc;

UPDATE accounts

SET balance = balance + amount

WHERE account\_id = to\_acc;

ELSE

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Insufficient balance in source account';

END IF;

END $$

DELIMITER ;

CALL TransferFunds(2, 1, 500.00);

SELECT \* FROM accounts;

**Output :**

**A screenshot of a computer code

AI-generated content may be incorrect.**