WEEK 2: PL/SQL programming

**Exercise 1: Control Structures**

**INIT**

CREATE TABLE customers (

customer\_id INT GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1),

Name VARCHAR2(100),

age Int,

isVip NUMBER(1), -- 1 for TRUE, 0 for FALSE

balance int

);

CREATE TABLE loans (

loan\_id INT GENERATED ALWAYS AS IDENTITY,

customer\_id INT,

interest\_rate NUMBER(5,2),

due\_date DATE

);

INSERT INTO customers (Name, age, isVip, balance) VALUES ('Alice Johnson', 65, 0, 12000);

INSERT INTO customers (Name, age, isVip, balance) VALUES ('Bob Smith', 45, 0, 8000);

INSERT INTO customers (Name, age, isVip, balance) VALUES ('Carol Lee', 70, 0, 15000);

INSERT INTO customers (Name, age, isVip, balance) VALUES ('David Kim', 30, 0, 9500);

INSERT INTO customers (Name, age, isVip, balance) VALUES ('Eva Green', 55, 0, 11000);

INSERT INTO loans (customer\_id, interest\_rate, due\_date) VALUES (1, 5.00, SYSDATE + 10);

INSERT INTO loans (customer\_id, interest\_rate, due\_date) VALUES (2, 6.50, SYSDATE + 40);

INSERT INTO loans (customer\_id, interest\_rate, due\_date) VALUES (3, 4.75, SYSDATE + 25);

INSERT INTO loans (customer\_id, interest\_rate, due\_date) VALUES (4, 7.00, SYSDATE + 5);

INSERT INTO loans (customer\_id, interest\_rate, due\_date) VALUES (5, 5.50, SYSDATE + 60);

1. **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 CODE:**

BEGIN

FOR cust\_rec IN (SELECT CUSTOMER\_ID, AGE FROM CUSTOMERS) LOOP

IF cust\_rec.AGE > 60 THEN

UPDATE LOANS

SET INTEREST\_RATE = INTEREST\_RATE - 0.01

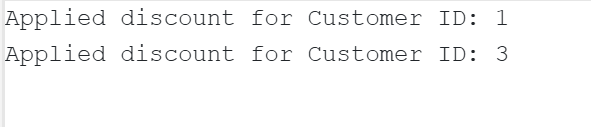
WHERE CUSTOMER\_ID = cust\_rec.CUSTOMER\_ID;

END IF;

END LOOP;

COMMIT;

END;



1. **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 CODE:**

BEGIN

FOR cust\_rec IN (SELECT customer\_id, balance FROM customers) LOOP

IF cust\_rec.balance > 10000 THEN

UPDATE customers

SET isVip = 1

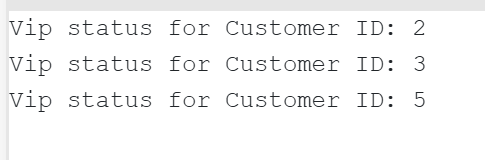
WHERE customer\_id = cust\_rec.customer\_id;

END IF;

END LOOP;

COMMIT;

END;



1. **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 CODE:**

BEGIN

FOR rec IN (

SELECT c.name, l.customer\_id, 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 for ' || rec.name ||

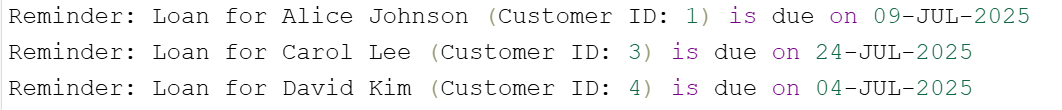
' (Customer ID: ' || rec.customer\_id ||

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

END LOOP;

END;

/



**Exercise 3: Stored Procedures**

**INIT**

-- Create and Insert Data into Tables

CREATE TABLE savings\_accounts (

account\_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

customer\_id INT,

balance NUMBER(12, 2) CHECK (balance >= 0)

);

INSERT INTO savings\_accounts (customer\_id, balance) VALUES (1, 1200.00);

INSERT INTO savings\_accounts (customer\_id, balance) VALUES (2, 2500.00);

INSERT INTO savings\_accounts (customer\_id, balance) VALUES (3, 3400.00);

INSERT INTO savings\_accounts (customer\_id, balance) VALUES (4, 1800.00);

INSERT INTO savings\_accounts (customer\_id, balance) VALUES (5, 2750.00);

CREATE TABLE employees (

employee\_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

name VARCHAR2(100),

department\_id INT,

salary NUMBER(10, 2) CHECK (salary >= 0)

);

INSERT INTO employees (name, department\_id, salary) VALUES ('Alice Manager', 101, 5000);

INSERT INTO employees (name, department\_id, salary) VALUES ('Bob Analyst', 101, 4000);

INSERT INTO employees (name, department\_id, salary) VALUES ('Carol Developer', 102, 4500);

CREATE TABLE accounts (

account\_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

customer\_id INT,

balance NUMBER(12, 2) CHECK (balance >= 0)

);

INSERT INTO accounts (customer\_id, balance) VALUES (1, 5000.00);

INSERT INTO accounts (customer\_id, balance) VALUES (2, 3000.00);

INSERT INTO accounts (customer\_id, balance) VALUES (3, 7000.00);

INSERT INTO accounts (customer\_id, balance) VALUES (4, 2000.00);

INSERT INTO accounts (customer\_id, balance) VALUES (5, 1000.00);

1. **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 CODE:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

UPDATE savings\_accounts

SET balance = balance + (balance \* 0.01); -- Apply 1% interest to each account

COMMIT;

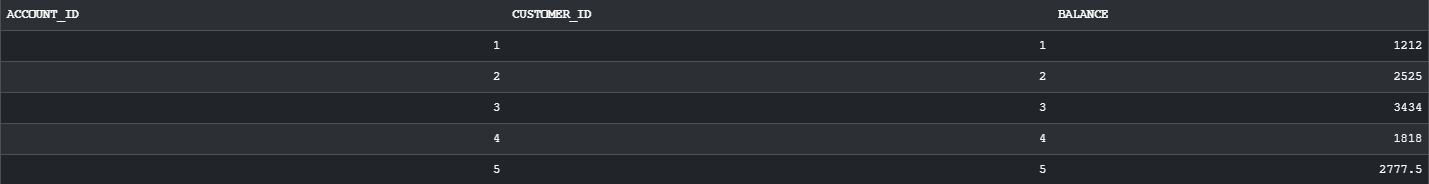
END;

/

EXEC ProcessMonthlyInterest;



Post execution: -



1. **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 CODE:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_department\_id IN NUMBER,

p\_bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE employees

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

WHERE department\_id = p\_department\_id;

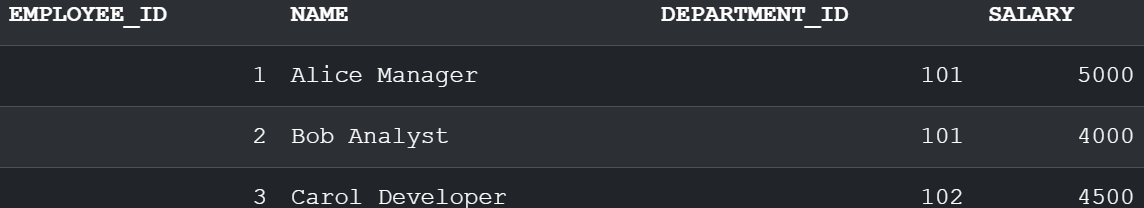
COMMIT;

END;

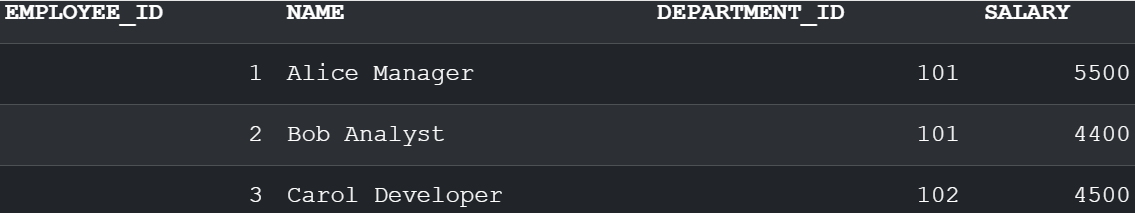
/

EXEC UpdateEmployeeBonus(101, 10);

SELECT \* FROM employees;



Post execution:-



1. **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 CODE:**

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_account\_id IN NUMBER,

p\_to\_account\_id IN NUMBER,

p\_amount IN NUMBER

) AS

v\_from\_balance NUMBER;

BEGIN

-- Check if source account has enough balance

SELECT balance INTO v\_from\_balance

FROM accounts

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

FOR UPDATE;

IF v\_from\_balance < p\_amount THEN

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

END IF;

-- Deduct the amount from source account

UPDATE accounts

SET balance = balance - p\_amount

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

-- Add the amount to destination account

UPDATE accounts

SET balance = balance + p\_amount

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

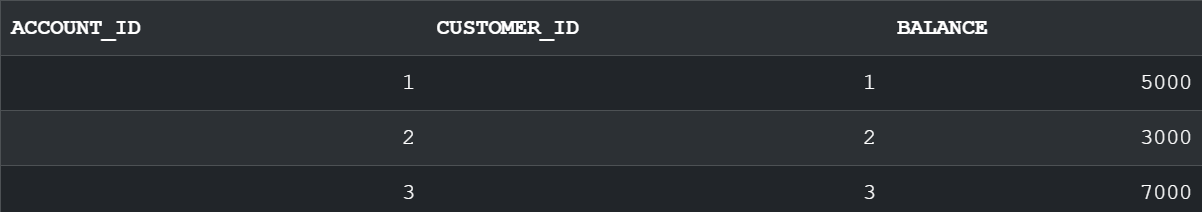
COMMIT;

END;

/

EXEC TransferFunds(1, 2, 500);

SELECT \* FROM accounts;



Post execution:-

