**PL/SQL Programming**

**Exercise 1: Control Structures**

**Tables:**

CREATE TABLE customer (

customer\_id NUMBER PRIMARY KEY,

full\_name VARCHAR2(100),

age NUMBER,

account\_balance NUMBER(10, 2),

vip\_status CHAR(1) DEFAULT 'N'

);

CREATE TABLE loan (

loan\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

rate\_of\_interest NUMBER(5, 2),

repayment\_date DATE,

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

);

**Inserting Values:**

INSERT INTO customer VALUES (101, 'Emma Watson', 65, 18000.50, 'N');

INSERT INTO customer VALUES (102, 'Daniel Craig', 58, 7000.00, 'N');

INSERT INTO customer VALUES (103, 'Olivia Wilde', 72, 12500.00, 'N');

INSERT INTO customer VALUES (104, 'Chris Evans', 30, 6000.75, 'N');

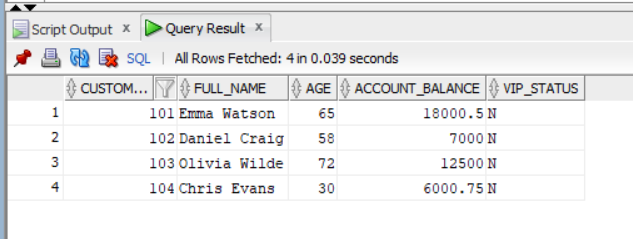
INSERT INTO loan VALUES (201, 101, 7.0, SYSDATE + 15);

INSERT INTO loan VALUES (202, 102, 8.5, SYSDATE + 45);

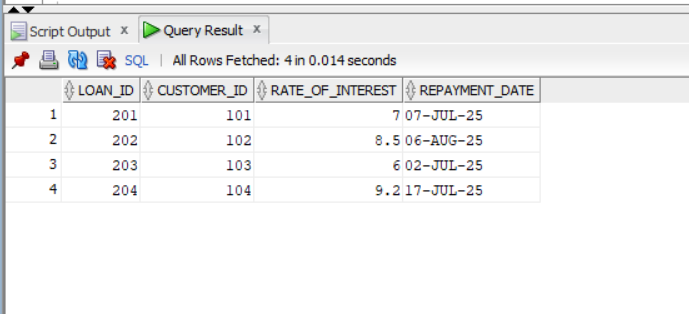
INSERT INTO loan VALUES (203, 103, 6.0, SYSDATE + 10);

INSERT INTO loan VALUES (204, 104, 9.2, SYSDATE + 25);

**Customer Table:**

****

**Loan Table:**

****

**Scenario 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.

**Code:**

BEGIN

FOR cust IN (SELECT customer\_id, age FROM customer) LOOP

IF cust.age > 60 THEN

UPDATE loan

SET rate\_of\_interest = rate\_of\_interest - 1

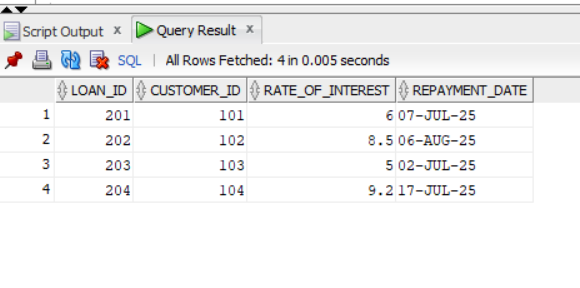
WHERE customer\_id = cust.customer\_id;

END IF;

END LOOP;

END;

**Output:**

**Scenario 2:**

**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:**

BEGIN

FOR cust\_info IN (SELECT customer\_id, account\_balance, vip\_status FROM customer) LOOP

IF cust\_info.account\_balance >= 10000 AND cust\_info.vip\_status = 'N' THEN

UPDATE customer

SET vip\_status = 'Y'

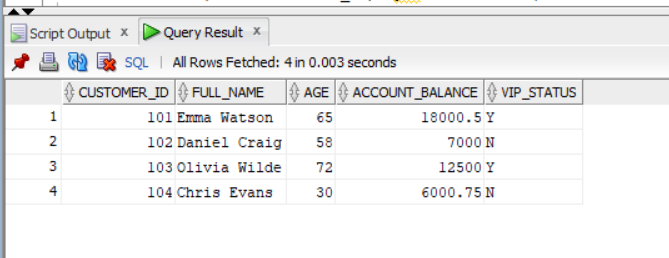
WHERE customer\_id = cust\_info.customer\_id;

END IF;

END LOOP;

END;

**Output:**



**Scenario 3:**

**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:**

SET SERVEROUTPUT ON;

BEGIN

FOR loan\_info IN (

SELECT l.loan\_id, l.repayment\_date, c.full\_name

FROM loan l

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

WHERE l.repayment\_date BETWEEN SYSDATE AND SYSDATE + 20

) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Alert: Dear ' || loan\_info.full\_name ||

', your loan #' || loan\_info.loan\_id ||

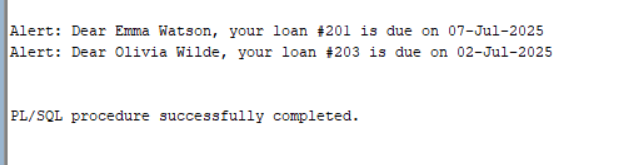
' is due on ' || TO\_CHAR(loan\_info.repayment\_date, 'DD-Mon-YYYY')

);

END LOOP;

END;

**Output:**



**Exercise 3: Stored Procedures**

**Tables:**

CREATE TABLE account (

account\_id NUMBER PRIMARY KEY,

customer\_name VARCHAR2(100),

account\_type VARCHAR2(20),

balance NUMBER(10, 2)

);

CREATE TABLE employee (

emp\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

department VARCHAR2(50),

salary NUMBER(10, 2)

);

**Inserting Values:**

INSERT INTO account VALUES (11, 'Ananya Roy', 'Savings', 12000);

INSERT INTO account VALUES (12, 'Raj Mehta', 'Current', 7000);

INSERT INTO account VALUES (13, 'Neha Verma', 'Savings', 18000);

INSERT INTO account VALUES (14, 'Kunal Das', 'Savings', 4000);

INSERT INTO employee VALUES (201, 'Reena Singh', 'HR', 52000);

INSERT INTO employee VALUES (202, 'Amit Joshi', 'IT', 65000);

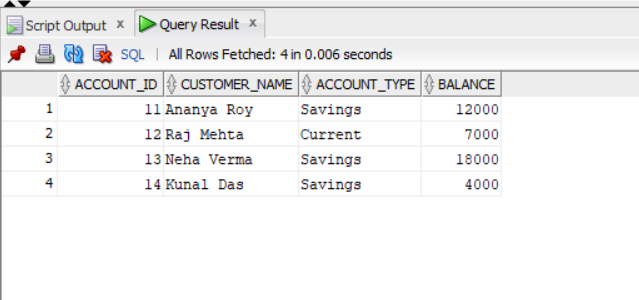
INSERT INTO employee VALUES (203, 'Tanya Nair', 'Finance', 58000);

INSERT INTO employee VALUES (204, 'Vikram Rao', 'HR', 47000);

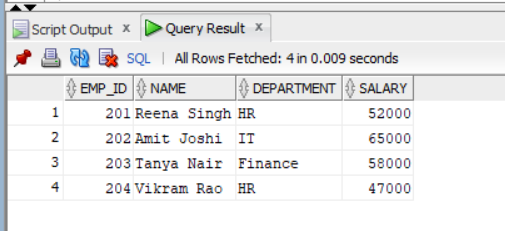
select \* from account;

select \* from employee;

**Account Table:**



**Employee Table:**



**Scenario 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.

**Code:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE account

SET balance = balance + (balance \* 0.01)

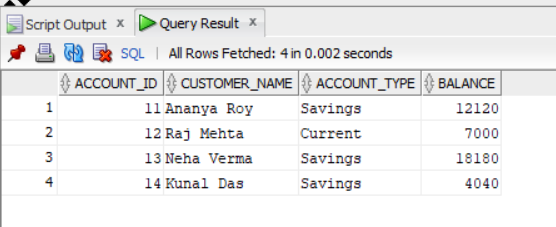
WHERE LOWER(account\_type) = 'savings';

END;

/

EXEC ProcessMonthlyInterest;

**Output:**



**Scenario 2:**

**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:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE employee

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

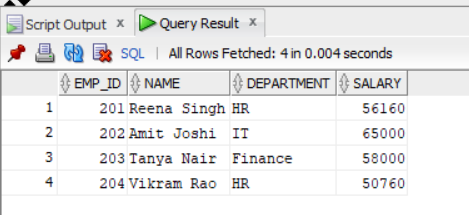
WHERE department = dept\_name;

END;

/

EXEC UpdateEmployeeBonus('HR', 8);

**Output:**



**Scenario 3:**

**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:**

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_account\_id IN NUMBER,

to\_account\_id IN NUMBER,

amount IN NUMBER

) IS

from\_balance NUMBER;

BEGIN

SELECT balance INTO from\_balance

FROM account

WHERE account\_id = from\_account\_id;

IF from\_balance < amount OR amount <= 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Invalid transfer amount or insufficient balance.');

END IF;

UPDATE account

SET balance = balance - amount

WHERE account\_id = from\_account\_id;

UPDATE account

SET balance = balance + amount

WHERE account\_id = to\_account\_id;

END;

/

EXEC TransferFunds(11, 12, 1500);

**Output:**

