**Exercise 1: Control Structures**

**Creating Tables:**

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

age NUMBER,

balance NUMBER(10, 2),

is\_vip CHAR(1) DEFAULT 'N'

);

CREATE TABLE loans (

loan\_id NUMBER PRIMARY KEY,

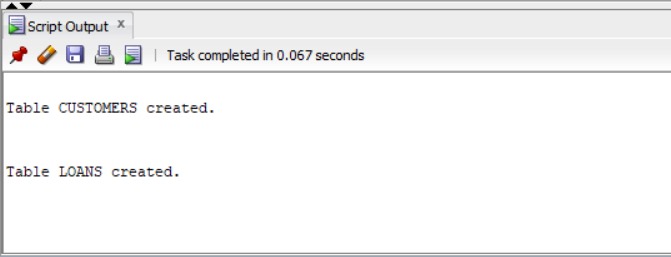
customer\_id NUMBER,

interest\_rate NUMBER(5, 2),

due\_date DATE,

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

);



**Inserting Values:**

INSERT INTO customers VALUES (1, 'John Doe', 69, 12000, 'N');

INSERT INTO customers VALUES (2, 'Alice Smith', 39, 9500, 'N');

INSERT INTO customers VALUES (3, 'Bob Johnson', 76, 15000, 'N');

INSERT INTO customers VALUES (4, 'Charlie Brown', 24, 8000, 'N');

INSERT INTO loans VALUES (101, 1, 7.5, SYSDATE + 10);

INSERT INTO loans VALUES (102, 2, 8.0, SYSDATE + 40);

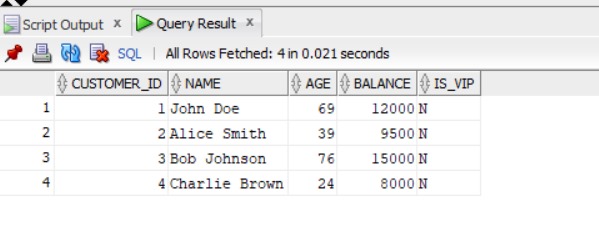
INSERT INTO loans VALUES (103, 3, 6.5, SYSDATE + 5);

INSERT INTO loans VALUES (104, 4, 9.0, SYSDATE + 20);

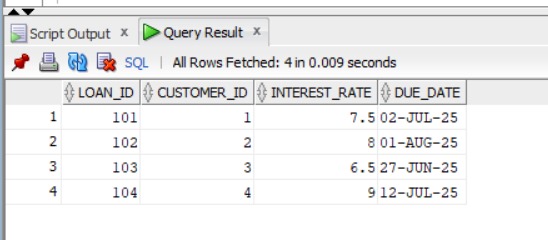
select \* from customers;

select \* from loans;

**Customers Table:**



**Loans Table:**



**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:**

BEGIN

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

IF cust.age > 60 THEN

UPDATE loans

SET interest\_rate = interest\_rate - 1

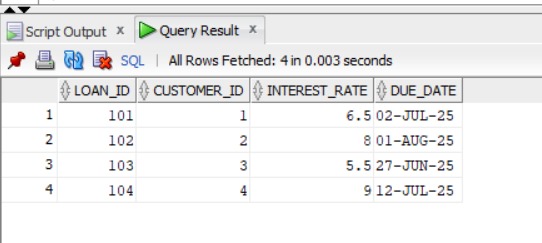
WHERE customer\_id = cust.customer\_id;

END IF;

END LOOP;

END;

**Output:** Loans Table



Interest\_Rate has been reduced for Customers having id 1 and 3 whose ages are 69 and 73 respectively.

**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:**

BEGIN

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

IF cust.balance > 10000 THEN

UPDATE customers

SET is\_vip = 'Y'

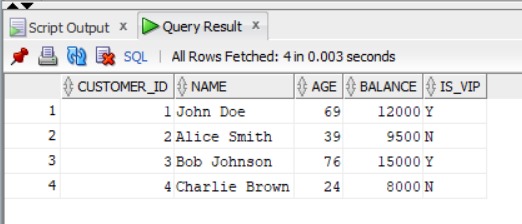
WHERE customer\_id = cust.customer\_id;

END IF;

END LOOP;

END;

**Output:** Customers Table



Customers 1 and 3 have been promoted to VIP since their Balance is >10000.

**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:**

SET SERVEROUTPUT ON;

BEGIN

FOR loan\_rec IN (

SELECT l.loan\_id, l.due\_date, c.name

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 ' || loan\_rec.loan\_id ||

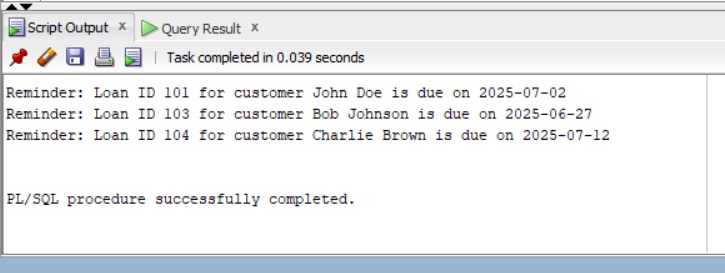
' for customer ' || loan\_rec.name ||

' is due on ' || TO\_CHAR(loan\_rec.due\_date, 'YYYY-MM-DD'));

END LOOP;

END;

**Output:**



Reminders have been sent for customers 1, 3, 4.

**Exercise 3: Stored Procedures**

**Creating Tables:**

CREATE TABLE accounts (

account\_id NUMBER PRIMARY KEY,

customer\_name VARCHAR2(100),

account\_type VARCHAR2(20),

balance NUMBER(10, 2)

);

CREATE TABLE employees (

emp\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

department VARCHAR2(50),

salary NUMBER(10, 2)

);

**Inserting Values:**

INSERT INTO accounts VALUES (1, 'John Doe', 'Savings', 10000);

INSERT INTO accounts VALUES (2, 'Alice Smith', 'Current', 8000);

INSERT INTO accounts VALUES (3, 'Bob Johnson', 'Savings', 15000);

INSERT INTO accounts VALUES (4, 'Charlie Brown', 'Savings', 5000);

INSERT INTO employees VALUES (101, 'Sarah Lee', 'HR', 50000);

INSERT INTO employees VALUES (102, 'David Kim', 'IT', 60000);

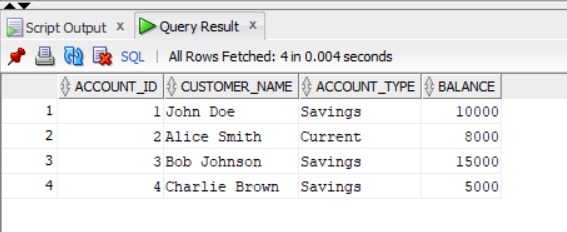
INSERT INTO employees VALUES (103, 'Priya Sharma', 'HR', 55000);

INSERT INTO employees VALUES (104, 'Mark Green', 'Sales', 45000);

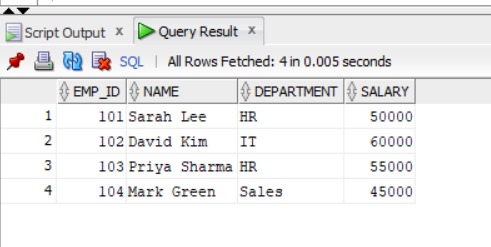
select \* from accounts;

select \* from employees;

**Accounts Table:**



**Employees Table:**



**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:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE accounts

SET balance = balance + (balance \* 0.01)

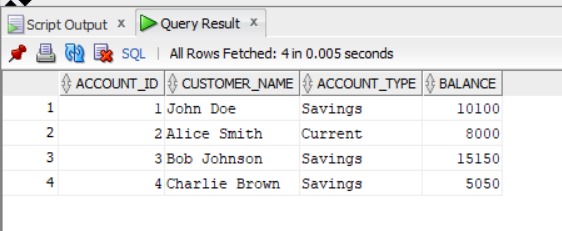
WHERE account\_type = 'Savings';

END;

/

EXEC ProcessMonthlyInterest;

**Output:** Accounts Table



Balance of Accounts 1 and 3 (accounts having type Savings) have been increased by 1%.

**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:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE employees

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

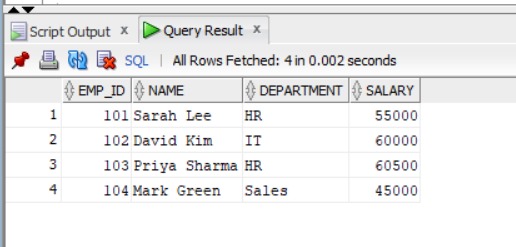
WHERE department = dept\_name;

END;

/

EXEC UpdateEmployeeBonus('HR', 10);

**Output:** Employees Table



Salary of Employees 1 and 3 (HR Department) have been incremented by 10%.

**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:**

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 accounts

WHERE account\_id = from\_account\_id;

IF from\_balance < amount THEN

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

END IF;

UPDATE accounts

SET balance = balance - amount

WHERE account\_id = from\_account\_id;

UPDATE accounts

SET balance = balance + amount

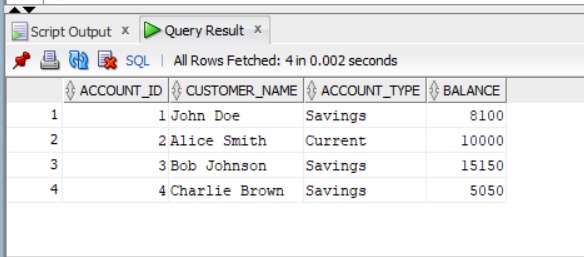
WHERE account\_id = to\_account\_id;

END;

/

EXEC TransferFunds(1, 2, 2000);

**Output:** Accounts Table



2000 Rupees have been transferred from Account 1 to Account 2.