**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:**

**SQL>** CREATE TABLE customers2 (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(50),

age NUMBER,

loan\_interest\_rate NUMBER(5,2)

);

**SQL>** INSERT INTO customers2 VALUES (301, 'Johnny', 62, 7.5);

**SQL>** INSERT INTO customers2 VALUES (302, 'Meera', 59, 8.0);

**SQL>** INSERT INTO customers2 VALUES (303, 'Sita', 67, 9.0);

**PL/SQL:**

DECLARE

CURSOR senior\_customers IS

SELECT customer\_id, loan\_interest\_rate

FROM customers2

WHERE age > 60;

new\_rate customers2.loan\_interest\_rate%TYPE;

BEGIN

FOR cust IN senior\_customers LOOP

new\_rate := cust.loan\_interest\_rate \* 0.99;

UPDATE customers2

SET loan\_interest\_rate = new\_rate

WHERE customer\_id = cust.customer\_id;

END LOOP;

COMMIT;

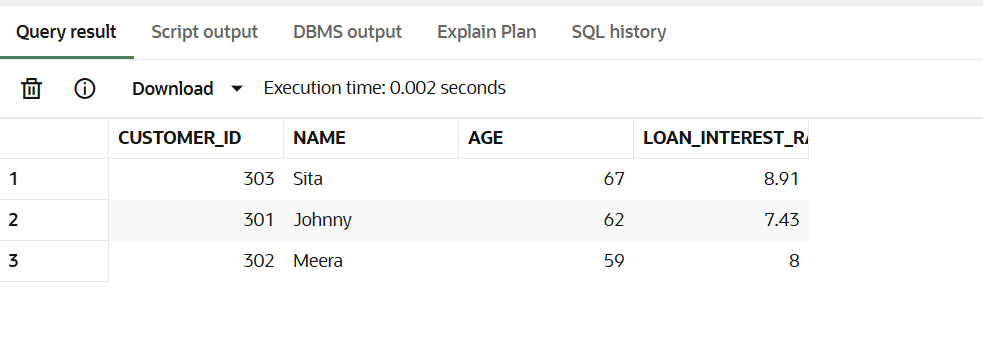
DBMS\_OUTPUT.PUT\_LINE('Senior customer loan discounts applied.');

END;

/

**SQL>** Select \* from customers2;

**Output:**



**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:**

**SQL>** ALTER TABLE customers2 ADD IsVIP VARCHAR2(5);

**SQL>** ALTER TABLE customers2 ADD balance NUMBER;

**SQL>** UPDATE customers2 SET balance = 12000 WHERE customer\_id = 303;

**SQL>** UPDATE customers2 SET balance = 9000 WHERE customer\_id = 302;

**SQL>** UPDATE customers2 SET balance = 15000 WHERE customer\_id = 301;

**PL/SQL:**

DECLARE

CURSOR rich\_customers IS

SELECT customer\_id, name, balance

FROM customers2

WHERE balance > 10000;

BEGIN

FOR cust IN rich\_customers LOOP

UPDATE customers2

SET IsVIP = 'TRUE'

WHERE customer\_id = cust.customer\_id;

END LOOP;

COMMIT;

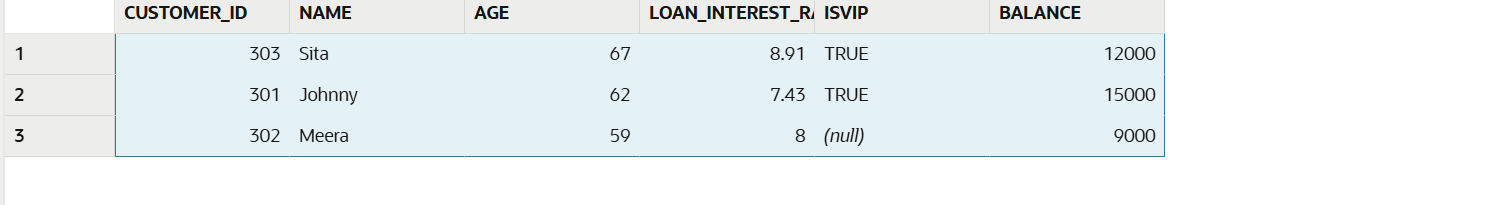
DBMS\_OUTPUT.PUT\_LINE(‘updated for customers.');

END;

/

**SQL>** select \* from customers2;

**Output:**



**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:**

**SQL>** ALTER TABLE customers2 ADD loan\_id NUMBER;

**SQL>** ALTER TABLE customers2 ADD loan\_due\_date DATE;

SQL> UPDATE customers2 SET loan\_id = 503, loan\_due\_date = SYSDATE + 5 WHERE customer\_id = 303;

**SQL>** UPDATE customers2 SET loan\_id = 502, loan\_due\_date = SYSDATE + 35 WHERE customer\_id = 302;

**SQL>** UPDATE customers2 SET loan\_id = 501, loan\_due\_date = SYSDATE + 10 WHERE customer\_id = 301;

**PL/SQL:**

DECLARE

CURSOR upcoming\_loans IS

SELECT customer\_id, loan\_id, loan\_due\_date

FROM customers2

WHERE loan\_due\_date BETWEEN SYSDATE AND SYSDATE + 30;

BEGIN

FOR rec IN upcoming\_loans LOOP

-- here you could do other processing if needed

-- for now, we do nothing

NULL;

END LOOP;

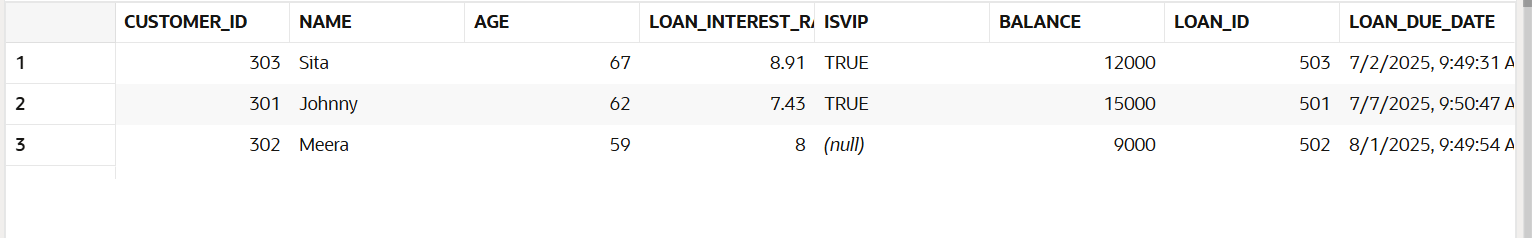
DBMS\_OUTPUT.PUT\_LINE('Loan reminders processing completed.');

END;

/

**SQL>** select \* from customers2;

**Output:**



**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:**

**SQL>** CREATE TABLE savings\_accounts (

account\_id NUMBER PRIMARY KEY,

balance NUMBER(12,2)

);

**SQL>** INSERT INTO savings\_accounts VALUES (1001, 5000);

**SQL>** INSERT INTO savings\_accounts VALUES (1002, 12000);

**SQL>** INSERT INTO savings\_accounts VALUES (1003, 8000);

**PL/SQL:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest

AS

BEGIN

FOR rec IN (

SELECT account\_id, balance

FROM savings\_accounts

)

LOOP

UPDATE savings\_accounts

SET balance = rec.balance \* 1.01

WHERE account\_id = rec.account\_id;

END LOOP;

COMMIT;

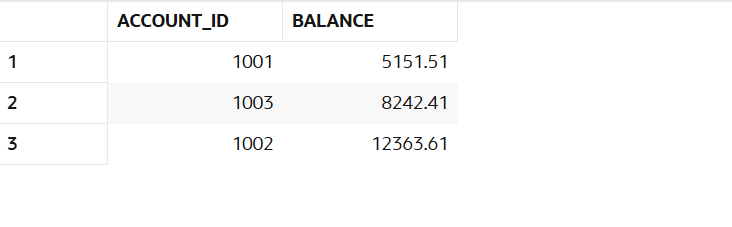
END;

/

**SQL>** EXEC ProcessMonthlyInterest;

**SQL>** SELECT \* FROM savings\_accounts;

**Output:**



**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:**

**SQL>** CREATE TABLE employees (

emp\_id NUMBER PRIMARY KEY,

emp\_name VARCHAR2(50),

department VARCHAR2(30),

salary NUMBER(10,2)

);

**SQL>** INSERT INTO employees VALUES (201, 'Ravi', 'HR', 40000);

**SQL>** INSERT INTO employees VALUES (202, 'Priya', 'HR', 45000);

**SQL>** INSERT INTO employees VALUES (203, 'John', 'Finance', 55000);

**PL/SQL:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

)

AS

BEGIN

FOR emp IN (

SELECT emp\_id, salary

FROM employees

WHERE department = dept\_name

)

LOOP

UPDATE employees

SET salary = emp.salary \* (1 + bonus\_percent / 100)

WHERE emp\_id = emp.emp\_id;

END LOOP;

COMMIT;

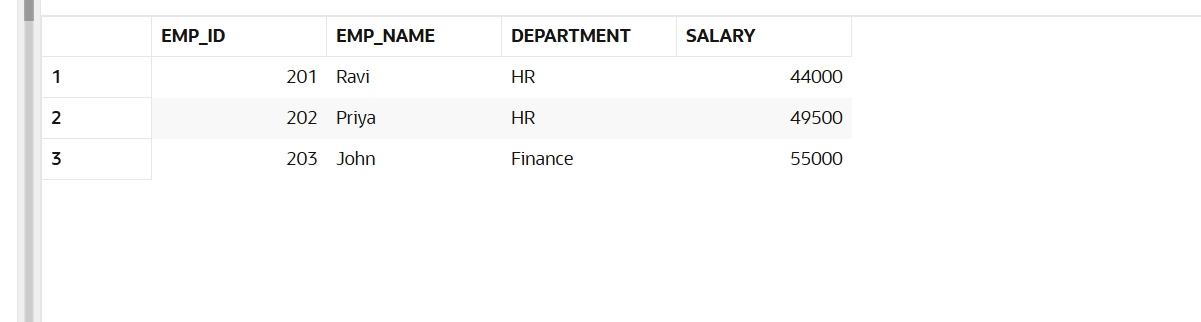
END;

/

**SQL>** EXEC UpdateEmployeeBonus('HR', 10);

**SQL>** SELECT \* FROM employees;

**Output:**



**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:**

SQL> CREATE TABLE customer\_accounts (

account\_number NUMBER PRIMARY KEY,

customer\_name VARCHAR2(50),

balance NUMBER(12,2)

);

**SQL>** INSERT INTO customer\_accounts VALUES (3001, 'Amit', 10000);

**SQL>** INSERT INTO customer\_accounts VALUES (3002, 'Meera', 15000);

**SQL>** INSERT INTO customer\_accounts VALUES (3003, 'Sita', 7000);

**PL/SQL:**

CREATE OR REPLACE PROCEDURE TransferFunds (

source\_acct IN NUMBER,

target\_acct IN NUMBER,

amount IN NUMBER

)

AS

source\_balance customer\_accounts.balance%TYPE;

BEGIN

SELECT balance

INTO source\_balance

FROM customer\_accounts

WHERE account\_number = source\_acct;

IF source\_balance < amount THEN

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

END IF;

UPDATE customer\_accounts

SET balance = balance - amount

WHERE account\_number = source\_acct;

UPDATE customer\_accounts

SET balance = balance + amount

WHERE account\_number = target\_acct;

COMMIT;

END;

/

**SQL>** EXEC TransferFunds(3002, 3003, 2000);

**SQL>** SELECT \* FROM customer\_accounts;

**Output:**

