1. **Control Structures**

**Code:**

**Tables-**

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

customer\_name VARCHAR2(100),

age NUMBER,

balance NUMBER(12, 2),

isVIP VARCHAR2(5) DEFAULT 'FALSE',

loan\_interest\_rate NUMBER(5, 2)

);

INSERT INTO customers (customer\_id,customer\_name,age,balance,loan\_interest\_rate) VALUES (1, 'Alice', 65, 15000, 8.5);

INSERT INTO customers (customer\_id,customer\_name,age,balance,loan\_interest\_rate) VALUES (2, 'Bob', 45, 7000, 9.0);

INSERT INTO customers (customer\_id,customer\_name,age,balance,loan\_interest\_rate) VALUES (3, 'Charlie', 61, 11000, 9.2);

INSERT INTO customers (customer\_id,customer\_name,age,balance,loan\_interest\_rate) VALUES (4, 'Diana', 30, 9500, 10.0);

**Scenario 1:**

SELECT customer\_id, age, loan\_interest\_rate from customers;

DECLARE

CURSOR cur\_senior\_customers IS

SELECT customer\_id, loan\_interest\_rate

FROM customers

WHERE age > 60;

v\_customer\_id customers.customer\_id%TYPE;

v\_loan\_interest\_rate customers.loan\_interest\_rate%TYPE;

BEGIN

OPEN cur\_senior\_customers;

LOOP

FETCH cur\_senior\_customers INTO v\_customer\_id, v\_loan\_interest\_rate;

EXIT WHEN cur\_senior\_customers%NOTFOUND;

UPDATE customers

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

WHERE customer\_id = v\_customer\_id;

END LOOP;

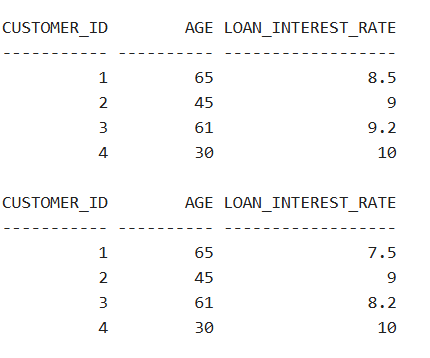
CLOSE cur\_senior\_customers;

END;

/

SELECT customer\_id, age, loan\_interest\_rate from customers;

**Output:**



**Scenario 2:**

SELECT customer\_id, age, loan\_interest\_rate from customers;

DECLARE

CURSOR cur\_vip\_candidates IS

SELECT customer\_id

FROM customers

WHERE balance > 10000;

v\_customer\_id customers.customer\_id%TYPE;

BEGIN

OPEN cur\_vip\_candidates;

LOOP

FETCH cur\_vip\_candidates INTO v\_customer\_id;

EXIT WHEN cur\_vip\_candidates%NOTFOUND;

UPDATE customers

SET isVIP = 'TRUE' -- use 'Y' or 1 based on your DB design

WHERE customer\_id = v\_customer\_id;

END LOOP;

CLOSE cur\_vip\_candidates;

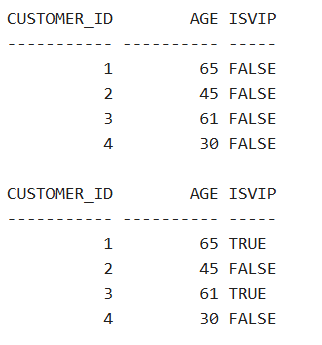
COMMIT;

END;

/

SELECT customer\_id, age, loan\_interest\_rate from customers;

**Output:**



**Scenario 3:**

DECLARE

CURSOR cur\_due\_loans IS

SELECT l.loan\_id, l.customer\_id, l.due\_date, c.customer\_name

FROM loans l

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

WHERE l.customer\_id = 1;

v\_loan\_id loans.loan\_id%TYPE;

v\_customer\_id customers.customer\_id%TYPE;

v\_due\_date loans.due\_date%TYPE;

v\_customer\_name customers.customer\_name%TYPE;

BEGIN

OPEN cur\_due\_loans;

LOOP

FETCH cur\_due\_loans INTO v\_loan\_id, v\_customer\_id, v\_due\_date, v\_customer\_name;

EXIT WHEN cur\_due\_loans%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Reminder: Dear ' || v\_customer\_name ||

', your loan (ID: ' || v\_loan\_id ||

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

END LOOP;

CLOSE cur\_due\_loans;

END;

/

**3) Stored Procedures**

**Scenario 1:**

CREATE TABLE accounts(

account\_id number PRIMARY KEY,

customer\_id number,

balance number(12, 2),

account\_type VARCHAR2(20)

);

INSERT INTO accounts VALUES (1, 101, 5000.00, 'SAVINGS');

INSERT INTO accounts VALUES (2, 102, 15000.50, 'CURRENT');

INSERT INTO accounts VALUES (3, 103, 8000.75, 'SAVINGS');

INSERT INTO accounts VALUES (4, 104, 25000.00, 'CURRENT');

SELECT \* from accounts;

CREATE or REPLACE PROCEDURE ProcessMonthlyInterest

IS

BEGIN

UPDATE accounts

SET balance=balance\*101/100

WHERE account\_type=’SAVINGS’;

END;

/

BEGIN

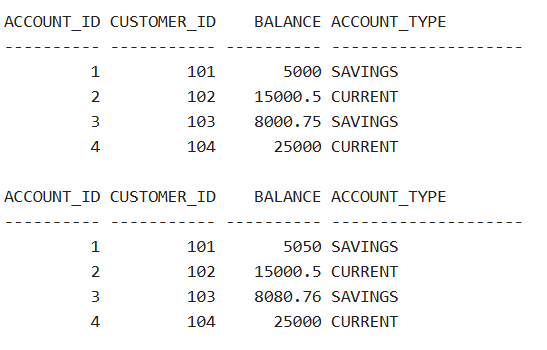
ProcessMonthlyInterest;

END;

/

SELECT \* from accounts;

**Output:**



**Scenario 2:**

CREATE TABLE employees (

employee\_id NUMBER PRIMARY KEY,

employee\_name VARCHAR2(100),

department\_id NUMBER,

salary NUMBER(10, 2)

);

INSERT INTO employees VALUES (1, 'Alice', 10, 50000);

INSERT INTO employees VALUES (2, 'Bob', 20, 60000);

INSERT INTO employees VALUES (3, 'Charlie', 10, 55000);

INSERT INTO employees VALUES (4, 'Diana', 30, 70000);

SELECT employee\_id, department\_id, salary FROM employees;

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;

END UpdateEmployeeBonus;

/

BEGIN

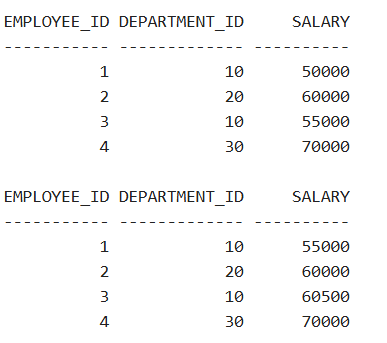
UpdateEmployeeBonus(10, 10);

END;

/

SELECT employee\_id, department\_id, salary FROM employees;

**Output:**



**Scenario 3:**

CREATE TABLE accounts(

account\_id number PRIMARY KEY,

customer\_id number,

balance number(12, 2)

);

INSERT INTO accounts VALUES (1, 101, 5000.00);

INSERT INTO accounts VALUES (2, 102, 15000.50);

INSERT INTO accounts VALUES (3, 103, 8000.75);

INSERT INTO accounts VALUES (4, 104, 25000.00);

SELECT account\_id, balance FROM accounts;

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_source\_account IN accounts.account\_id%TYPE,

p\_target\_account IN accounts.account\_id%TYPE,

p\_amount IN NUMBER

) AS

v\_source\_balance accounts.balance%TYPE;

BEGIN

SELECT balance INTO v\_source\_balance

FROM accounts

WHERE account\_id = p\_source\_account

FOR UPDATE;

IF v\_source\_balance < p\_amount THEN

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

END IF;

UPDATE accounts

SET balance = balance - p\_amount

WHERE account\_id = p\_source\_account;

UPDATE accounts

SET balance = balance + p\_amount

WHERE account\_id = p\_target\_account;

COMMIT;

END TransferFunds;

/

BEGIN

TransferFunds(2, 1, 10000);

END;

/

SELECT account\_id, balance FROM accounts;

**Output:**

