CSE-4508 Lab Task 02

Name: Rashikh Ahmad
Student ID: 210041255
Section: 2A
Semester: 5th
Academic Year: 2023-24

October 10, 2024

Task 1

Task 1

```
SET SERVEROUTPUT ON;

CREATE TABLE employee (
    ID NUMBER PRIMARY KEY,
    Name VARCHAR2(50),
    Salary NUMBER,
    Designation VARCHAR2(30)
);

INSERT INTO employee (ID, Name, Salary, Designation) VALUES (1, 'John
    Doe', 25000, 'manager');
-- Additional data...
DECLARE
    v_rows_updated NUMBER := 0;
BEGIN
    -- Update salary for managers with salary < 30000
    UPDATE employee
    SET salary = salary * 1.1
    WHERE designation = 'manager' AND salary < 30000;

    v_rows_updated := v_rows_updated + SQL%ROWCOUNT; -- Count affected
        rows
    DBMS_OUTPUT.PUT_LINE('Total rows affected: ' || v_rows_updated);
END;
/
```

Task 2

Task 2

```
CREATE TABLE transactions (
    User_ID NUMBER,
    Amount NUMBER,
    T_Date DATE
);

INSERT INTO transactions (User_ID, Amount, T_Date) VALUES (1, 500000,
    TO_DATE('2024-01-10', 'YYYY-MM-DD'));
-- Additional data...
CREATE OR REPLACE FUNCTION get_loan_scheme(p_user_id NUMBER)
RETURN NUMBER IS
    v_total_transactions NUMBER := 0;
    v_loan_scheme NUMBER := NULL;
    CURSOR c_loan IS
        SELECT Scheme, Min_Trans
        FROM loan_type
        ORDER BY Min_Trans DESC;
BEGIN
    -- Calculate total transactions for the given user
    SELECT SUM(Amount) INTO v_total_transactions
    FROM transactions
    WHERE User_ID = p_user_id;

    -- Iterate through the loan_type table using the cursor
    FOR r_loan IN c_loan LOOP
        IF v_total_transactions >= r_loan.Min_Trans THEN
            v_loan_scheme := r_loan.Scheme;
            EXIT; -- Exit the loop once a matching scheme is found
        END IF;
    END LOOP;

    RETURN v_loan_scheme;
END;
/
```

Task 3

Task 3

Task 3.1

```
-- Create CUSTOMER table
CREATE TABLE CUSTOMER (
    SSN NUMBER PRIMARY KEY,
    Name VARCHAR2(50),
    Surname VARCHAR2(50),
    PhoneNum VARCHAR2(15),
    Plan NUMBER
);

-- Insert data...
```

Task 3

Task 3.2

```
-- Create a trigger to update the bill based on phone calls
CREATE OR REPLACE TRIGGER update_bill_after_call
AFTER INSERT ON PHONECALL
FOR EACH ROW
DECLARE
    v_connection_fee NUMBER;
    v_price_per_second NUMBER;
    v_call_cost NUMBER;
BEGIN
    -- Get the pricing plan details for the customer's plan
    SELECT ConnectionFee, PricePerSecond
    INTO v_connection_fee, v_price_per_second
    FROM PRICINGPLAN
    WHERE Code = (SELECT Plan FROM CUSTOMER WHERE SSN = :NEW.SSN);

    -- Calculate the call cost
    v_call_cost := v_connection_fee + (v_price_per_second * :NEW.Seconds
        );

    -- Update the BILL table with the calculated cost
    UPDATE BILL
    SET Amount = Amount + v_call_cost
    WHERE SSN = :NEW.SSN
    AND Month = EXTRACT(MONTH FROM SYSDATE)
    AND Year = EXTRACT(YEAR FROM SYSDATE);
END;
/
```

Task 4

Task 4

Task 4.1

```
-- Create sequence for the XX part of the ID
CREATE SEQUENCE seq_student_id
START WITH 1
INCREMENT BY 1
NOCACHE;

-- Create the STUDENT table
CREATE TABLE STUDENT (
    ID VARCHAR2(10) PRIMARY KEY,
    Date_Of_Admission DATE,
    Department CHAR(1),
    Program CHAR(1),
    Section CHAR(1)
);

-- Create the Gen_ID function
CREATE OR REPLACE FUNCTION Gen_ID(p_date_of_admission DATE, p_department
     CHAR, p_program CHAR, p_section CHAR)
RETURN VARCHAR2
IS
    v_year VARCHAR2(2);
    v_number VARCHAR2(2);
    v_id VARCHAR2(10);
BEGIN
    -- Extract the last two digits of the year from the Date of
       Admission
    v_year := TO_CHAR(p_date_of_admission, 'YY');

    -- Get the next number in the sequence and format it to 2 digits
    v_number := LPAD(seq_student_id.NEXTVAL, 2, '0');

    -- Construct the ID in the format YY00DPSXX
    v_id := v_year || '00' || p_department || p_program || p_section ||
       v_number;

    RETURN v_id;
END;
/

-- Create the trigger to automatically generate ID when inserting a new
   student
CREATE OR REPLACE TRIGGER trg_generate_student_id
BEFORE INSERT ON STUDENT
FOR EACH ROW
BEGIN
    -- Call the Gen_ID function to generate the ID
    :NEW.ID := Gen_ID(:NEW.Date_Of_Admission, :NEW.Department, :NEW.
       Program, :NEW.Section);
END;
/
```

Task 4.2

```
CREATE TABLE Accounts (
    ID NUMBER PRIMARY KEY,
    Name VARCHAR2(100),
    AccCode VARCHAR2(20),
    Balance NUMBER,
    LastDateofInterest DATE
);

-- Additional table and procedure definitions...

CREATE OR REPLACE PROCEDURE UpdateAccountBalances IS
    CURSOR account_cursor IS
        SELECT a.ID, a.Balance, a.LastDateofInterest, ap.InterestRate,
            ap.GP
        FROM Accounts a
        JOIN AccountProperties ap ON a.ID = ap.ID;

    v_new_balance NUMBER;
    v_days_difference NUMBER;
    v_interest NUMBER;

BEGIN
    FOR account_record IN account_cursor LOOP
        -- Calculate the difference in days between the current date and
            the last date of interest
        v_days_difference := TRUNC(SYSDATE) - TRUNC(account_record.
            LastDateofInterest);

        -- Check if interest needs to be added based on GP (Growth
            Period)
        IF (account_record.GP = 1 AND v_days_difference >= 1) OR
           (account_record.GP = 2 AND v_days_difference >= 30) OR
           (account_record.GP = 3 AND v_days_difference >= 365) THEN

            -- Calculate the interest
            v_interest := account_record.Balance * (account_record.
                InterestRate / 100);

            -- Update the new balance
            v_new_balance := account_record.Balance + v_interest;

            -- Update the account with the new balance and set the
                LastDateofInterest to today
            UPDATE Accounts
            SET Balance = v_new_balance,
                LastDateofInterest = TRUNC(SYSDATE)
            WHERE ID = account_record.ID;
        END IF;
    END LOOP;

    COMMIT; -- Commit the changes to the database
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK; -- Rollback in case of any error
END;
/
```