Department of Computer Science and Engineering

Islamic University of Technology (IUT)

A subsidiary organ of OIC

Lab-8 Report

CSE 4508: RDBMS Lab Report

Name: Anm Zahid Hossain Milkan

Student ID: 200041202
Section: 2B
Semester: 5th
Academic Year: 2022-23

Date of Submission: 3/11/23

Task 1: Briefly mention the differences between a PL/SQL procedure and a function.

Certainly, let's present the differences between PL/SQL procedures and functions in a more concise format:

PL/SQL Procedure:

1. Purpose: Used for performing actions or tasks, often involving data modification.
2. Return Value: Does not return a value.
3. Usage: Suitable when you need to execute a series of statements or actions without expecting a result.
4. Call Syntax: Called using the `CALL` statement or by invoking the procedure's name.
5. Return Type: Does not have a return type.
6. DDL (Data Definition Language): Not directly involved in DDL operations.

PL/SQL Function:

1. Purpose: Used for computing and returning a specific result or value.
2. Return Value: Must return a value of a specified data type.
3. Usage: Suitable when you need to calculate and return a value, often used in SQL queries and expressions.
4. Call Syntax: Called within SQL queries or expressions using the function's name.
5. Return Type: Must have a specified return type indicating the data type of the returned value.
6. DDL (Data Definition Language): Can be used in DDL operations for data validation or retrieval of specific values.

TASK 2:

SQL:

i.

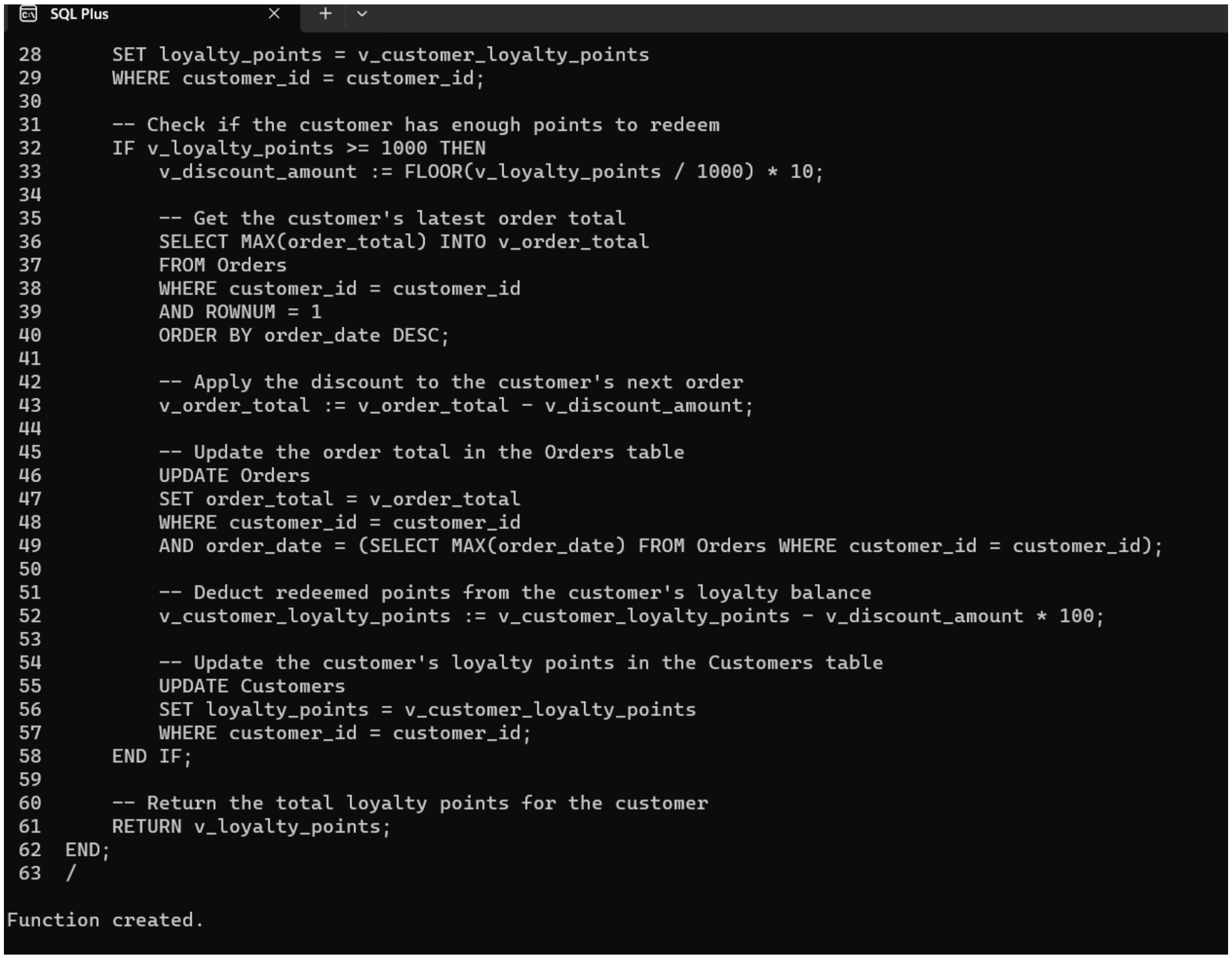
```
CREATE TABLE Authors (
    author_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50)
);
```

```sql
CREATE TABLE Books (
    book_id INT PRIMARY KEY,
    title VARCHAR(100),
    author_id INT,
    price DECIMAL(10, 2),
    publication_date DATE,
    FOREIGN KEY (author_id) REFERENCES Authors(author_id)
);

CREATE TABLE Customers (
    customer_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50)
);

CREATE TABLE Orders (
    order_id INT PRIMARY KEY,
    customer_id INT,
    order_date DATE,
    FOREIGN KEY (customer_id) REFERENCES
Customers(customer_id)
);

CREATE TABLE Order_Details (
    order_detail_id INT PRIMARY KEY,
    order_id INT,
    book_id INT,
    quantity INT,
    unit_price DECIMAL(10, 2),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (book_id) REFERENCES Books(book_id)
);
```

```sql
-- Insert authors
INSERT INTO Authors (author_id, first_name, last_name)
VALUES (1, 'Bibhutibhushan', 'Bandyopadhyay');
INSERT INTO Authors (author_id, first_name, last_name)
VALUES (2, 'Rabindranath', 'Tagore');

-- Insert books
INSERT INTO Books (title, author_id, price,
publication_date)
VALUES ('Pother Pachali', 1, 250.00,
TO_DATE('1955-01-01', 'YYYY-MM-DD'));

INSERT INTO Books (title, author_id, price,
publication_date)
VALUES ('Sesher Kobita', 2, 150.00, TO_DATE('1923-07-13',
'YYYY-MM-DD'));
-- Insert customers
INSERT INTO Customers (customer_id, first_name,
last_name) VALUES (1, 'Anm', 'Zahid');
INSERT INTO Customers (customer_id, first_name,
last_name) VALUES (2, 'Abu', 'Nowsad');
INSERT INTO Customers (customer_id, first_name,
last_name) VALUES (3, 'Jaber', 'Noman');

-- Insert orders
INSERT INTO Orders (order_id, customer_id, order_date)
VALUES (1, 1,  TO_DATE('2022-12-11', 'YYYY-MM-DD'));
INSERT INTO Orders (order_id, customer_id, order_date)
VALUES (2, 2,  TO_DATE('2023-5-11', 'YYYY-MM-DD'));
INSERT INTO Orders (order_id, customer_id, order_date)
VALUES (3, 2,  TO_DATE('2023-7-12', 'YYYY-MM-DD'));
```

```
-- Insert order details
INSERT INTO Order_Details (order_detail_id, order_id,
book_id, quantity, unit_price) VALUES (1, 1, 3, 2, 250.00);
INSERT INTO Order_Details (order_detail_id, order_id,
book_id, quantity, unit_price) VALUES (2, 2, 3, 1, 250.00);
INSERT INTO Order_Details (order_detail_id, order_id,
book_id, quantity, unit_price) VALUES (3, 2, 4 ,1, 150.00);\
```

Creation Snap:



```
        SET loyalty_points = v_customer_loyalty_points
        WHERE customer_id = customer_id;

        -- Check if the customer has enough points to redeem
        IF v_loyalty_points >= 1000 THEN
            v_discount_amount := FLOOR(v_loyalty_points / 1000) * 10;

            -- Get the customer's latest order total
            SELECT MAX(order_total) INTO v_order_total
            FROM Orders
            WHERE customer_id = customer_id
            AND ROWNUM = 1
            ORDER BY order_date DESC;

            -- Apply the discount to the customer's next order
            v_order_total := v_order_total - v_discount_amount;

            -- Update the order total in the Orders table
            UPDATE Orders
            SET order_total = v_order_total
            WHERE customer_id = customer_id
            AND order_date = (SELECT MAX(order_date) FROM Orders WHERE customer_id = customer_id);

            -- Deduct redeemed points from the customer's loyalty balance
            v_customer_loyalty_points := v_customer_loyalty_points - v_discount_amount * 100;

            -- Update the customer's loyalty points in the Customers table
            UPDATE Customers
            SET loyalty_points = v_customer_loyalty_points
            WHERE customer_id = customer_id;
        END IF;

        -- Return the total loyalty points for the customer
        RETURN v_loyalty_points;
    END;
    /

Function created.
```

Make Trigger To allow a customer to redeem points for a discount on their next order: here make a trigger instead

```
CREATE OR REPLACE TRIGGER RedeemLoyaltyPoints
BEFORE INSERT ON Orders
FOR EACH ROW
DECLARE
    v_customer_id NUMBER;
    v_discount_amount NUMBER;
BEGIN
    v_customer_id := :NEW.customer_id;
    SELECT loyalty_points / 1000 * 10
    INTO v_discount_amount
    FROM Customers
    WHERE customer_id = v_customer_id;
    IF v_discount_amount > 0 THEN
        :NEW.order_total := :NEW.order_total - v_discount_amount;
        UPDATE Customers
        SET loyalty_points = loyalty_points - (v_discount_amount * 
100)
        WHERE customer_id = v_customer_id;
    END IF;
END;
/
```

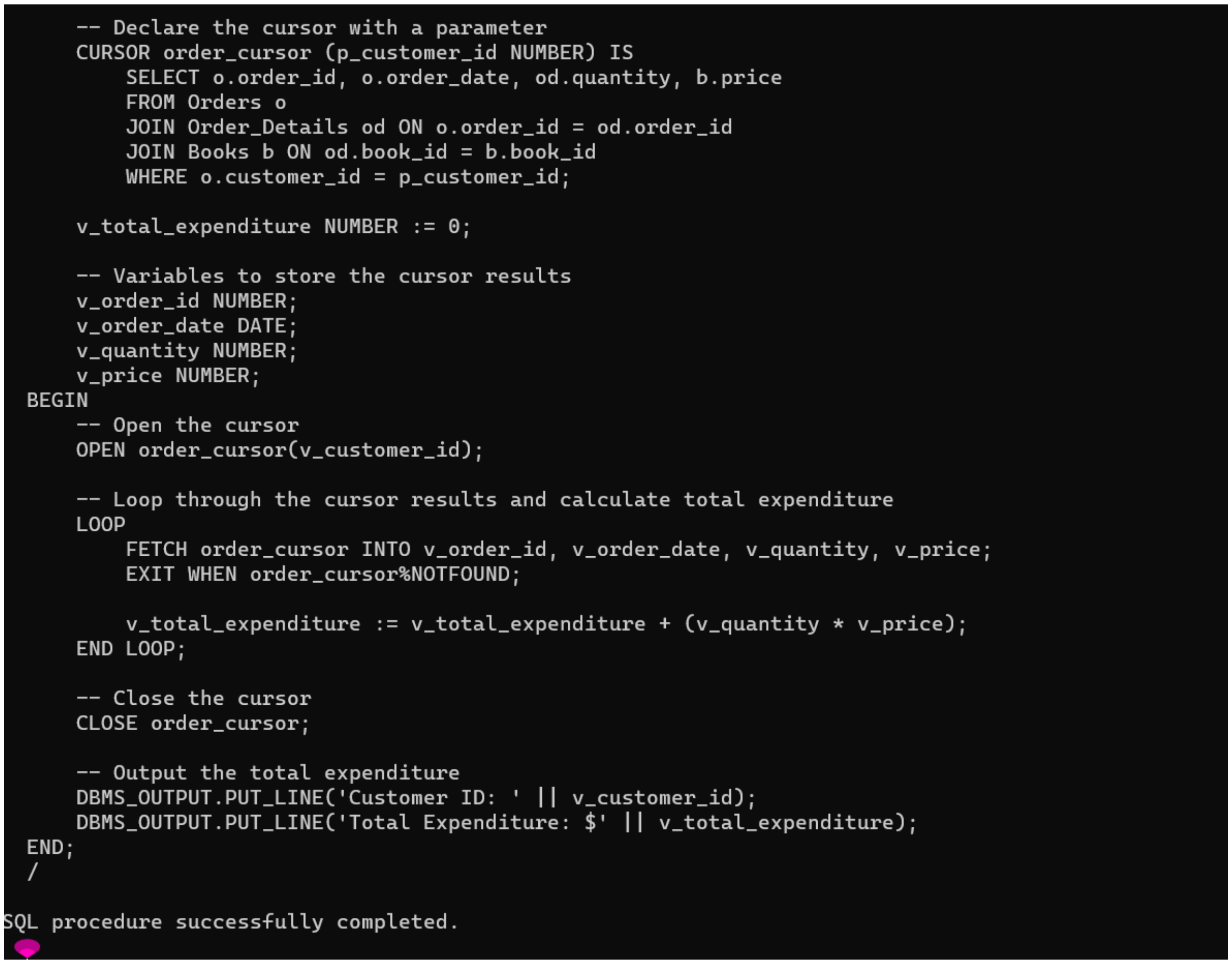
Task 2(iii): Write a PL/SQL block that uses an explicit cursor with parameters to retrieve all orders for a
specific customer and find the total expenditure of the customer based on the customer_id.

The cursor should accept the customer_id as a parameter and return the order details for that customer.

SQL:

```
DECLARE
    v_customer_id NUMBER := 1;
    CURSOR order_cursor (p_customer_id NUMBER) IS
        SELECT o.order_id, o.order_date, od.quantity,
b.price
        FROM Orders o
        JOIN Order_Details od ON o.order_id = od.order_id
        JOIN Books b ON od.book_id = b.book_id
        WHERE o.customer_id = p_customer_id;

    v_total_expenditure NUMBER := 0;
    v_order_id NUMBER;
    v_order_date DATE;
    v_quantity NUMBER;
    v_price NUMBER;
BEGIN
    OPEN order_cursor(v_customer_id);
    LOOP
        FETCH order_cursor INTO v_order_id, v_order_date,
v_quantity, v_price;
        EXIT WHEN order_cursor%NOTFOUND;

        v_total_expenditure := v_total_expenditure +
(v_quantity * v_price);
    END LOOP;
    CLOSE order_cursor;
    DBMS_OUTPUT.PUT_LINE('Customer ID: ' ||
v_customer_id);
    DBMS_OUTPUT.PUT_LINE('Total Expenditure: $' ||
v_total_expenditure);
END;
/
```

Creation Snap:



```
     -- Declare the cursor with a parameter
     CURSOR order_cursor (p_customer_id NUMBER) IS
         SELECT o.order_id, o.order_date, od.quantity, b.price
         FROM Orders o
         JOIN Order_Details od ON o.order_id = od.order_id
         JOIN Books b ON od.book_id = b.book_id
         WHERE o.customer_id = p_customer_id;

     v_total_expenditure NUMBER := 0;

     -- Variables to store the cursor results
     v_order_id NUMBER;
     v_order_date DATE;
     v_quantity NUMBER;
     v_price NUMBER;
 BEGIN
     -- Open the cursor
     OPEN order_cursor(v_customer_id);

     -- Loop through the cursor results and calculate total expenditure
     LOOP
         FETCH order_cursor INTO v_order_id, v_order_date, v_quantity, v_price;
         EXIT WHEN order_cursor%NOTFOUND;

         v_total_expenditure := v_total_expenditure + (v_quantity * v_price);
     END LOOP;

     -- Close the cursor
     CLOSE order_cursor;

     -- Output the total expenditure
     DBMS_OUTPUT.PUT_LINE('Customer ID: ' || v_customer_id);
     DBMS_OUTPUT.PUT_LINE('Total Expenditure: $' || v_total_expenditure);
 END;
 /

SQL procedure successfully completed.
```