**Department of Computer Science and Engineering**

**Islamic University of Technology (IUT)**

A subsidiary organ of OIC

# Laboratory Report

CSE 4508: RDBMS Lab

**Name: Abdullah**
**Student ID: 200041126**
**Section: 1B**
**Semester: 5th**
**Academic Year: 2022-2023**

**Date of Submission: 15/11/23**

## Task 1

### Differences between a PL/SQLprocedure and a function:

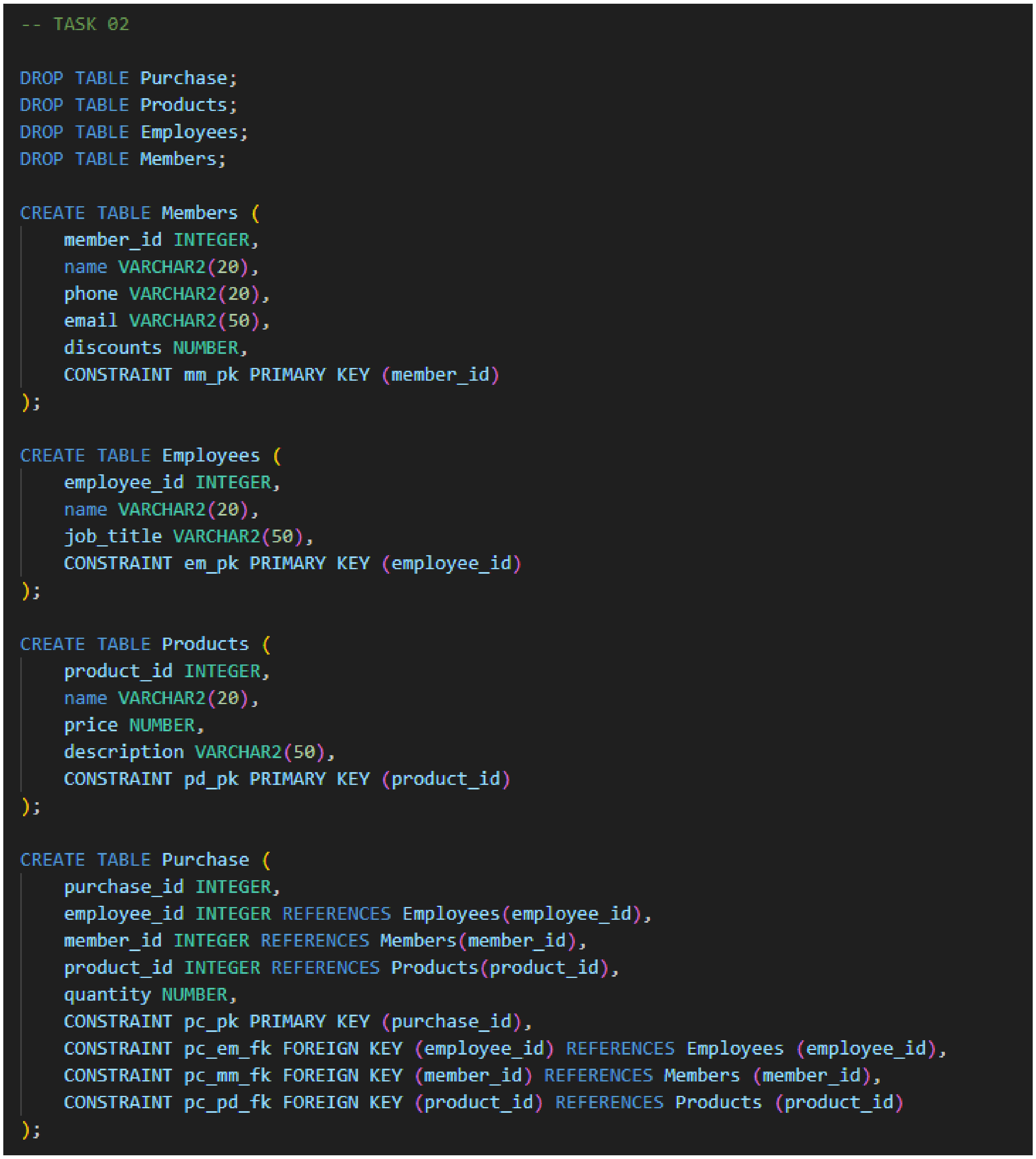
A procedure or function is similar to a miniature program. It has an optional declarative part, an executable part, and an optional exception-handling part.

- A procedure is a subprogram that performs a specific action. You specify the name of the procedure, its parameters, its local variables, and the BEGIN-END block that contains its code and handles any exceptions.

- A function is a subprogram that computes and returns a value. Functions and procedures are structured alike, except that functions return a value.

# Task 2

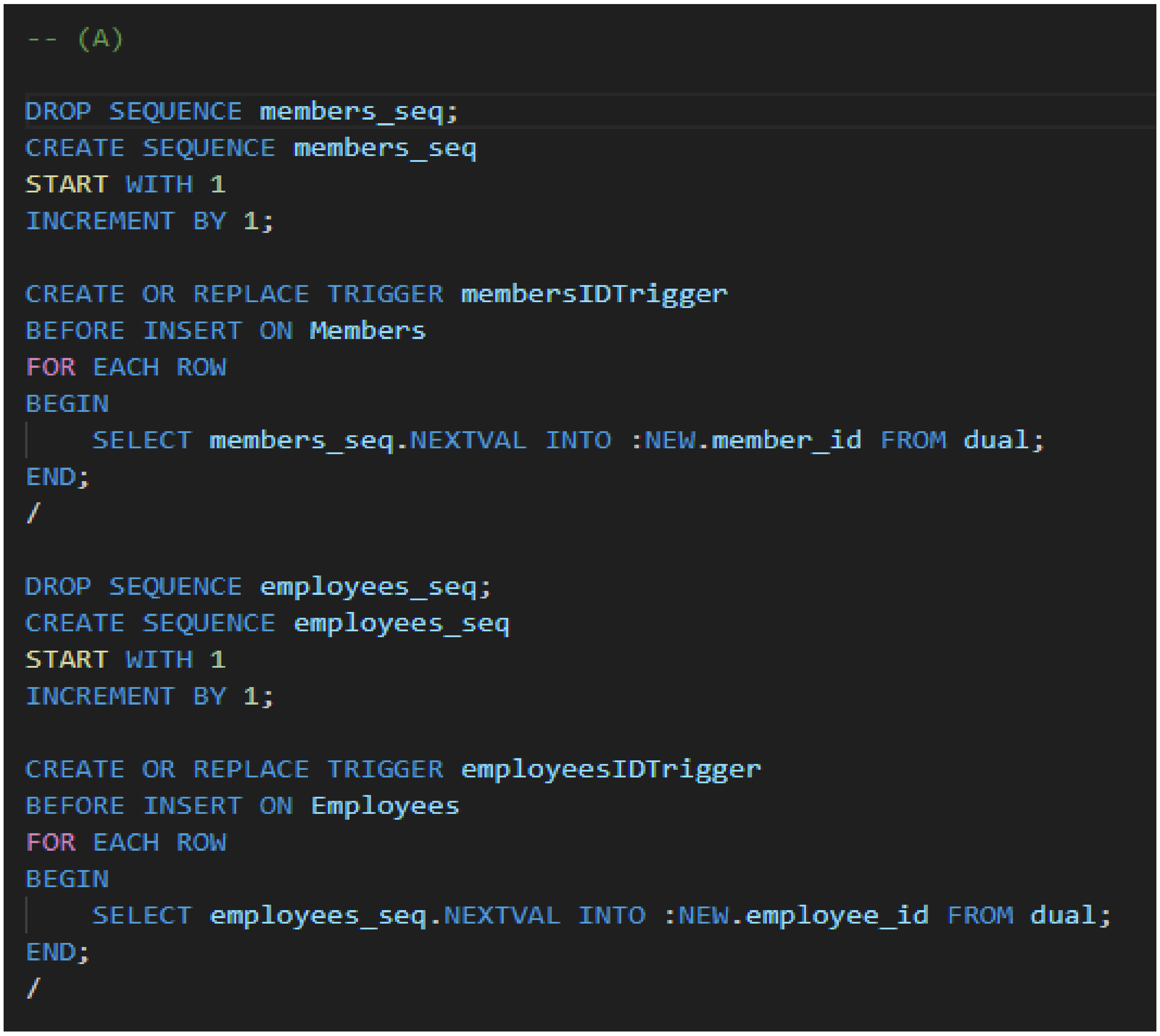
## Created Tables:



```
-- TASK 02

DROP TABLE Purchase;
DROP TABLE Products;
DROP TABLE Employees;
DROP TABLE Members;

CREATE TABLE Members (
    member_id INTEGER,
    name VARCHAR2(20),
    phone VARCHAR2(20),
    email VARCHAR2(50),
    discounts NUMBER,
    CONSTRAINT mm_pk PRIMARY KEY (member_id)
);

CREATE TABLE Employees (
    employee_id INTEGER,
    name VARCHAR2(20),
    job_title VARCHAR2(50),
    CONSTRAINT em_pk PRIMARY KEY (employee_id)
);

CREATE TABLE Products (
    product_id INTEGER,
    name VARCHAR2(20),
    price NUMBER,
    description VARCHAR2(50),
    CONSTRAINT pd_pk PRIMARY KEY (product_id)
);

CREATE TABLE Purchase (
    purchase_id INTEGER,
    employee_id INTEGER REFERENCES Employees(employee_id),
    member_id INTEGER REFERENCES Members(member_id),
    product_id INTEGER REFERENCES Products(product_id),
    quantity NUMBER,
    CONSTRAINT pc_pk PRIMARY KEY (purchase_id),
    CONSTRAINT pc_em_fk FOREIGN KEY (employee_id) REFERENCES Employees (employee_id),
    CONSTRAINT pc_mm_fk FOREIGN KEY (member_id) REFERENCES Members (member_id),
    CONSTRAINT pc_pd_fk FOREIGN KEY (product_id) REFERENCES Products (product_id)
);
```

## Task 2-A

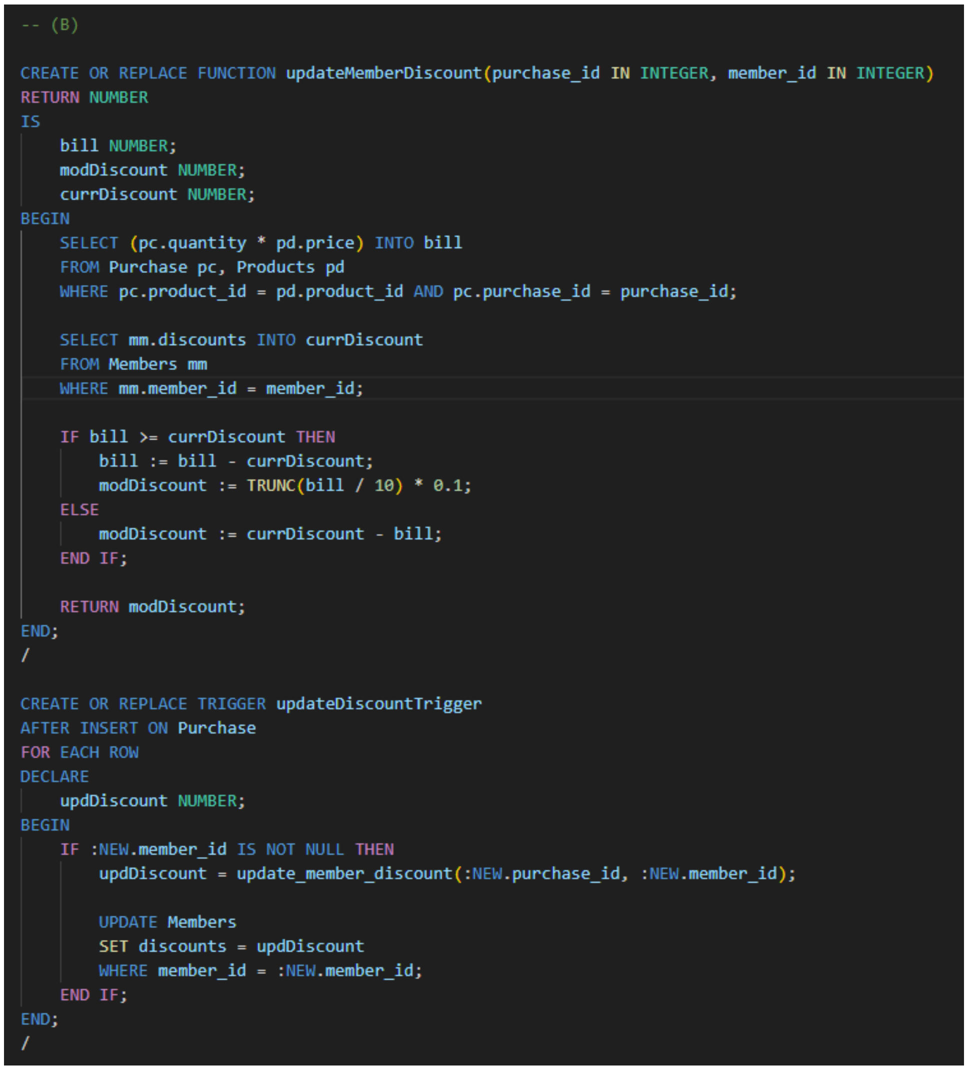
Here, I created two sequences, one for generating primary keys for Members table another for Employees table.



```
-- (A)

DROP SEQUENCE members_seq;
CREATE SEQUENCE members_seq
START WITH 1
INCREMENT BY 1;

CREATE OR REPLACE TRIGGER membersIDTrigger
BEFORE INSERT ON Members
FOR EACH ROW
BEGIN
    SELECT members_seq.NEXTVAL INTO :NEW.member_id FROM dual;
END;
/

DROP SEQUENCE employees_seq;
CREATE SEQUENCE employees_seq
START WITH 1
INCREMENT BY 1;

CREATE OR REPLACE TRIGGER employeesIDTrigger
BEFORE INSERT ON Employees
FOR EACH ROW
BEGIN
    SELECT employees_seq.NEXTVAL INTO :NEW.employee_id FROM dual;
END;
/
```

# Task 2-B

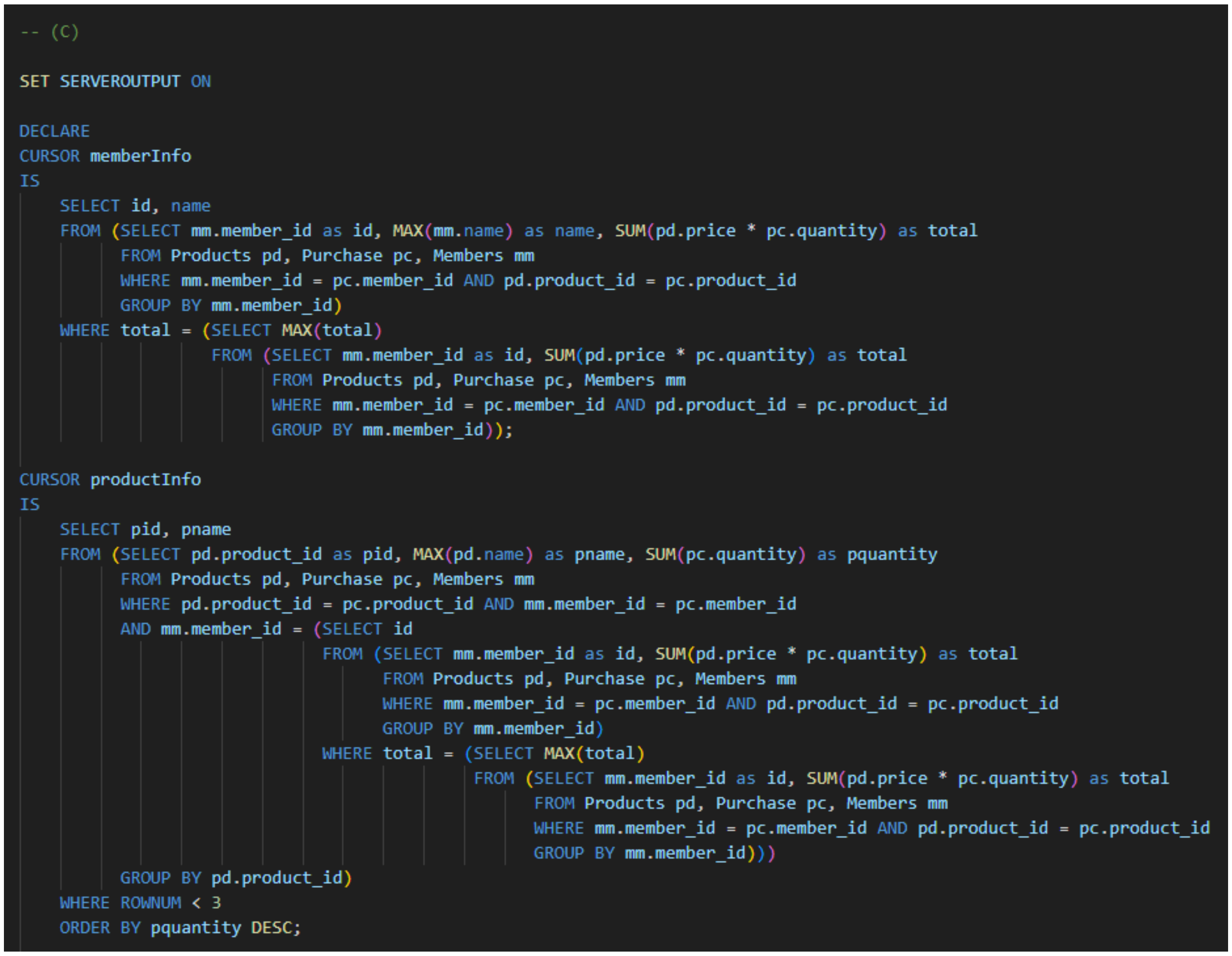
Here, I made a trigger updateDiscountTrigger which takes purchase id and member id and calls a function updateMemberDiscount. The function calculates the bill and current discount. Then I calculated new discount modify the column.



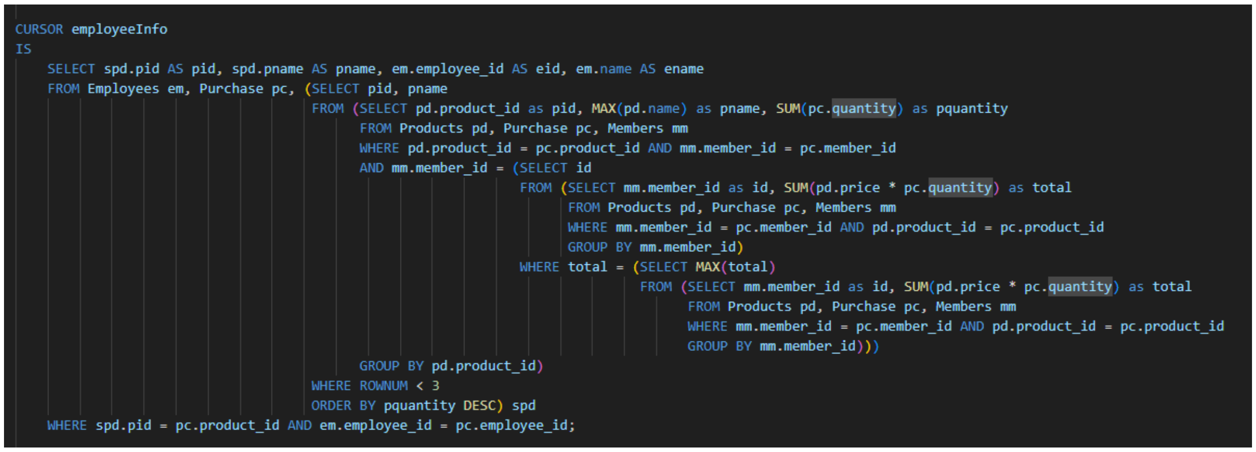
```sql
-- (B)

CREATE OR REPLACE FUNCTION updateMemberDiscount(purchase_id IN INTEGER, member_id IN INTEGER)
RETURN NUMBER
IS
    bill NUMBER;
    modDiscount NUMBER;
    currDiscount NUMBER;
BEGIN
    SELECT (pc.quantity * pd.price) INTO bill
    FROM Purchase pc, Products pd
    WHERE pc.product_id = pd.product_id AND pc.purchase_id = purchase_id;

    SELECT mm.discounts INTO currDiscount
    FROM Members mm
    WHERE mm.member_id = member_id;

    IF bill >= currDiscount THEN
        bill := bill - currDiscount;
        modDiscount := TRUNC(bill / 10) * 0.1;
    ELSE
        modDiscount := currDiscount - bill;
    END IF;

    RETURN modDiscount;
END;
/

CREATE OR REPLACE TRIGGER updateDiscountTrigger
AFTER INSERT ON Purchase
FOR EACH ROW
DECLARE
    updDiscount NUMBER;
BEGIN
    IF :NEW.member_id IS NOT NULL THEN
        updDiscount = update_member_discount(:NEW.purchase_id, :NEW.member_id);

        UPDATE Members
        SET discounts = updDiscount
        WHERE member_id = :NEW.member_id;
    END IF;
END;
/
```

## Task 2-C

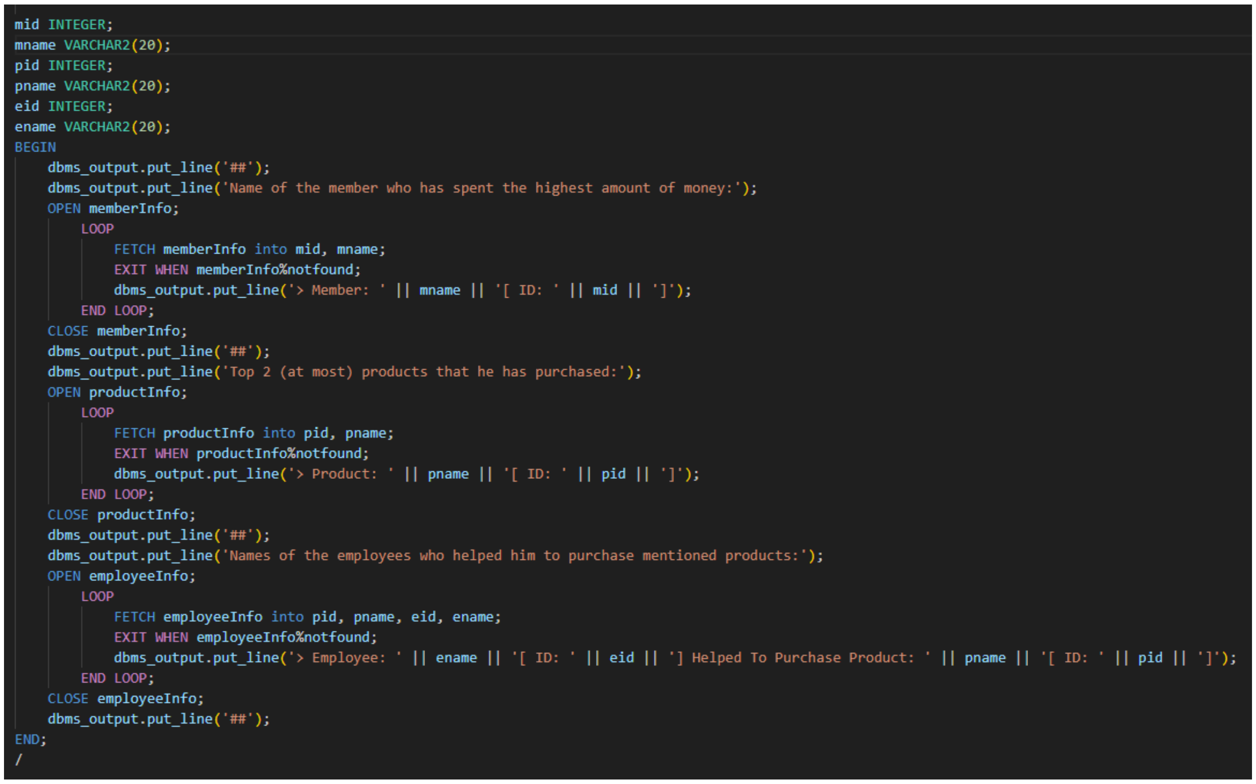
At first, I am creating a joint table of member, product and purchase and grouped the table by member id. Under one member id, there are multiple product rows. By multiplying product price and quantity and summing them all together, I got total cost of a member. Then I found who spent the maximum and extract id and name.
Now from the record of the member who spent the most, I grouped by the product id. Then I found out total quantity of a particular product. Then I found out top two maximum quantity's product.



```sql
-- (C)

SET SERVEROUTPUT ON

DECLARE
CURSOR memberInfo
IS
    SELECT id, name
    FROM (SELECT mm.member_id as id, MAX(mm.name) as name, SUM(pd.price * pc.quantity) as total
          FROM Products pd, Purchase pc, Members mm
          WHERE mm.member_id = pc.member_id AND pd.product_id = pc.product_id
          GROUP BY mm.member_id)
    WHERE total = (SELECT MAX(total)
                   FROM (SELECT mm.member_id as id, SUM(pd.price * pc.quantity) as total
                         FROM Products pd, Purchase pc, Members mm
                         WHERE mm.member_id = pc.member_id AND pd.product_id = pc.product_id
                         GROUP BY mm.member_id));

CURSOR productInfo
IS
    SELECT pid, pname
    FROM (SELECT pd.product_id as pid, MAX(pd.name) as pname, SUM(pc.quantity) as pquantity
          FROM Products pd, Purchase pc, Members mm
          WHERE pd.product_id = pc.product_id AND mm.member_id = pc.member_id
          AND mm.member_id = (SELECT id
                              FROM (SELECT mm.member_id as id, SUM(pd.price * pc.quantity) as total
                                    FROM Products pd, Purchase pc, Members mm
                                    WHERE mm.member_id = pc.member_id AND pd.product_id = pc.product_id
                                    GROUP BY mm.member_id)
                              WHERE total = (SELECT MAX(total)
                                             FROM (SELECT mm.member_id as id, SUM(pd.price * pc.quantity) as total
                                                   FROM Products pd, Purchase pc, Members mm
                                                   WHERE mm.member_id = pc.member_id AND pd.product_id = pc.product_id
                                                   GROUP BY mm.member_id)))
          GROUP BY pd.product_id)
    WHERE ROWNUM < 3
    ORDER BY pquantity DESC;
```



```sql
CURSOR employeeInfo
IS
    SELECT spd.pid AS pid, spd.pname AS pname, em.employee_id AS eid, em.name AS ename
    FROM Employees em, Purchase pc, (SELECT pid, pname
                                     FROM (SELECT pd.product_id as pid, MAX(pd.name) as pname, SUM(pc.quantity) as pquantity
                                           FROM Products pd, Purchase pc, Members mm
                                           WHERE pd.product_id = pc.product_id AND mm.member_id = pc.member_id
                                           AND mm.member_id = (SELECT id
                                                               FROM (SELECT mm.member_id as id, SUM(pd.price * pc.quantity) as total
                                                                     FROM Products pd, Purchase pc, Members mm
                                                                     WHERE mm.member_id = pc.member_id AND pd.product_id = pc.product_id
                                                                     GROUP BY mm.member_id)
                                                               WHERE total = (SELECT MAX(total)
                                                                              FROM (SELECT mm.member_id as id, SUM(pd.price * pc.quantity) as total
                                                                                    FROM Products pd, Purchase pc, Members mm
                                                                                    WHERE mm.member_id = pc.member_id AND pd.product_id = pc.product_id
                                                                                    GROUP BY mm.member_id)))
                                           GROUP BY pd.product_id)
                                     WHERE ROWNUM < 3
                                     ORDER BY pquantity DESC) spd
    WHERE spd.pid = pc.product_id AND em.employee_id = pc.employee_id;
```



```
mid INTEGER;
mname VARCHAR2(20);
pid INTEGER;
pname VARCHAR2(20);
eid INTEGER;
ename VARCHAR2(20);
BEGIN
    dbms_output.put_line('##');
    dbms_output.put_line('Name of the member who has spent the highest amount of money:');
    OPEN memberInfo;
        LOOP
            FETCH memberInfo into mid, mname;
            EXIT WHEN memberInfo%notfound;
            dbms_output.put_line('> Member: ' || mname || '[ ID: ' || mid || ']');
        END LOOP;
    CLOSE memberInfo;
    dbms_output.put_line('##');
    dbms_output.put_line('Top 2 (at most) products that he has purchased:');
    OPEN productInfo;
        LOOP
            FETCH productInfo into pid, pname;
            EXIT WHEN productInfo%notfound;
            dbms_output.put_line('> Product: ' || pname || '[ ID: ' || pid || ']');
        END LOOP;
    CLOSE productInfo;
    dbms_output.put_line('##');
    dbms_output.put_line('Names of the employees who helped him to purchase mentioned products:');
    OPEN employeeInfo;
        LOOP
            FETCH employeeInfo into pid, pname, eid, ename;
            EXIT WHEN employeeInfo%notfound;
            dbms_output.put_line('> Employee: ' || ename || '[ ID: ' || eid || '] Helped To Purchase Product: ' || pname || '[ ID: ' || pid || ']');
        END LOOP;
    CLOSE employeeInfo;
    dbms_output.put_line('##');
END;
/
```