SGBD 2022-2023

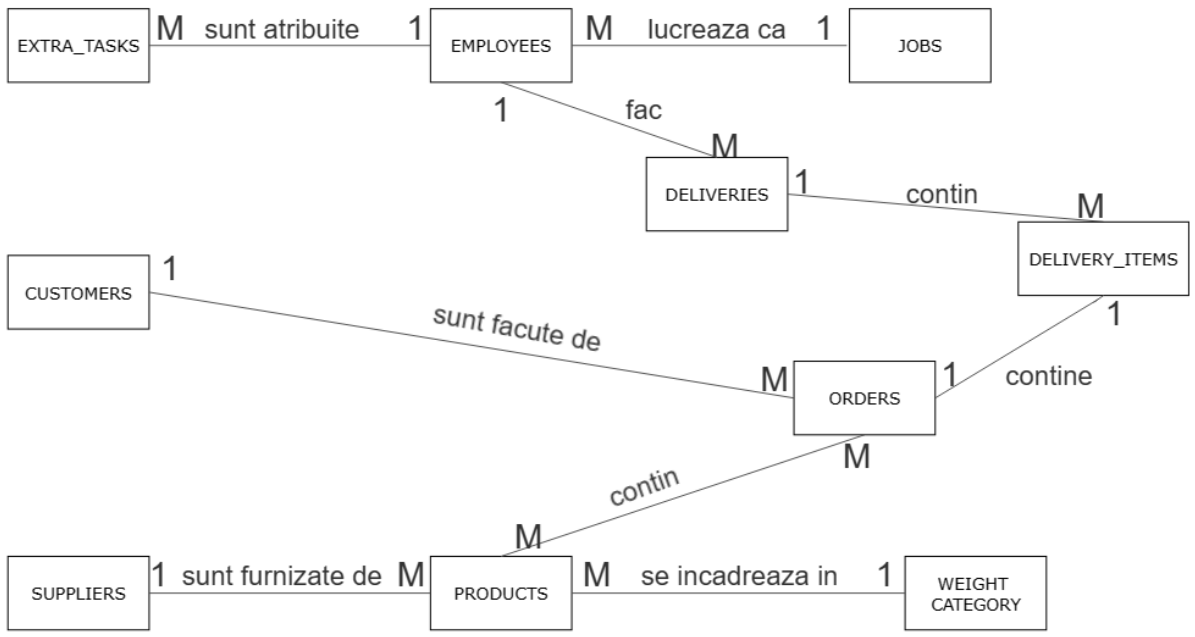
1. Utilitate

Prezentați pe scurt baza de date (utilitatea ei).

Baza de date gestioneaza activitatea unei companii mici care presteaza servicii de vanzari de produse, avand livratori proprii. Baza de date permite micro-managementul staffului avand un tabel de extra task-uri in care pot fi adaugate sarcini, care sunt acompaniate de bonusuri la salariu.

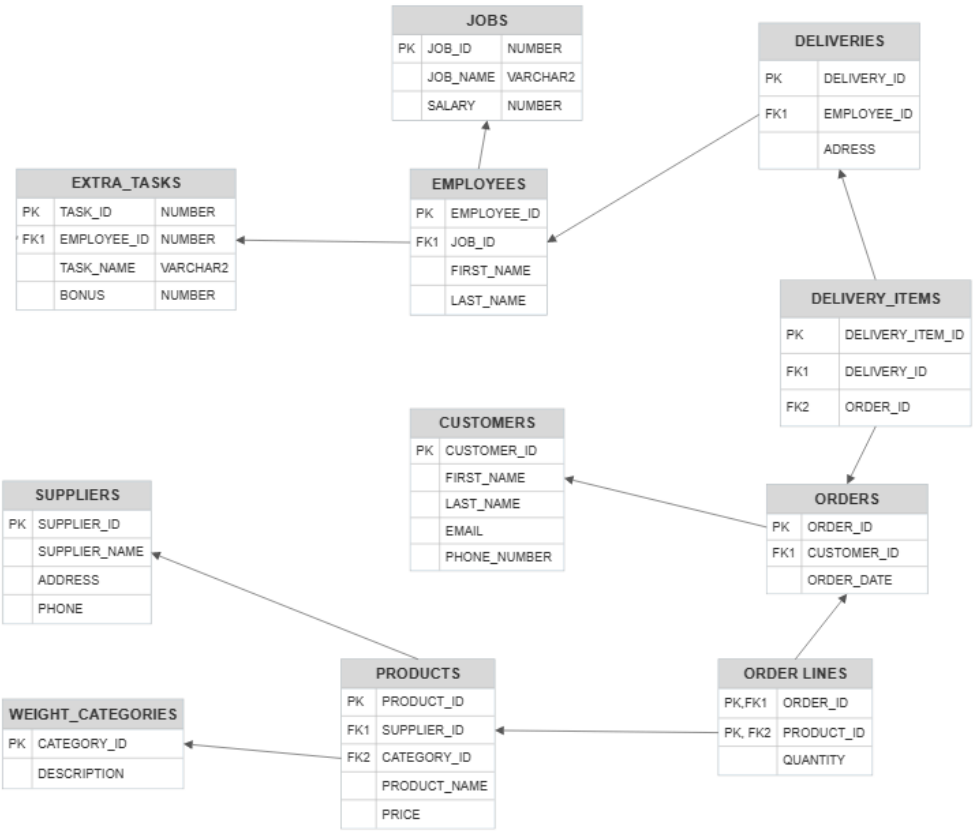
2. ERD

Realizați diagrama entitate-relație (ERD).



3. Diagrama conceptuală

Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare.



4. Implementați în Oracle diagrama

Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, implementând toate constrângerile de integritate necesare (chei primare, cheile externe etc).

```
CREATE TABLE CUSTOMERS (
    customer_ID INT PRIMARY KEY,
    first_name VARCHAR2(50),
    last_name VARCHAR2(50),
    phone_number VARCHAR2(20),
    email VARCHAR2(50),
    CONSTRAINT phone_number_unique UNIQUE (phone_number),
    CONSTRAINT email_unique UNIQUE (email)
);

CREATE TABLE ORDERS (
    order_ID INT PRIMARY KEY,
    customer_ID INT,
    order_date DATE,
    FOREIGN KEY (customer_ID) REFERENCES Customers(customer_ID)
);

CREATE TABLE SUPPLIERS (
```

```
    supplier_ID INT PRIMARY KEY,
    supplier_name VARCHAR2(100),
    address VARCHAR2(100),
    phone VARCHAR2(20)
);

CREATE TABLE WEIGHT_CATEGORY (
    category_ID INT PRIMARY KEY,
    DESCRIPTION VARCHAR2(200)
);

CREATE TABLE PRODUCTS (
    product_ID INT PRIMARY KEY,
    supplier_ID INT,
    category_ID INT,
    product_name VARCHAR2(100),
    price NUMBER(10, 2),
    FOREIGN KEY (supplier_ID) REFERENCES Suppliers(supplier_ID),
    FOREIGN KEY (category_ID) REFERENCES Weight_Category(category_ID)
);

CREATE TABLE JOBS (
    job_ID INT PRIMARY KEY,
    job_name VARCHAR2(50),
    salary NUMBER NOT NULL
);

CREATE TABLE EMPLOYEES (
    employee_ID INT PRIMARY KEY,
    first_name VARCHAR2(50),
    last_name VARCHAR2(50),
    job_id INT NOT NULL,
    FOREIGN KEY (job_id) REFERENCES Jobs(job_id)
);


CREATE TABLE EXTRA_TASKS (
    task_ID INT PRIMARY KEY,
    employee_ID INT,
    task_name VARCHAR2(100),
    bonus number(4, 3) NOT NULL,
    CONSTRAINT bonus_range check( bonus between 0 and 1 ),
    FOREIGN KEY (employee_ID) REFERENCES Employees(employee_ID)
);

CREATE TABLE ORDER_LINES (
    order_ID INT,
    product_ID INT,
    quantity INT,
    PRIMARY KEY (order_ID, product_ID),
    FOREIGN KEY (order_ID) REFERENCES Orders(order_ID),
    FOREIGN KEY (product_ID) REFERENCES Products(product_ID)
);
```

```
CREATE TABLE DELIVERIES (
    delivery_id INT PRIMARY KEY,
    employee_ID INT,
    adress VARCHAR2(100),
    FOREIGN KEY (employee_ID) REFERENCES Employees(employee_ID)
);

CREATE TABLE DELIVERY_ITEMS(
    delivery_item_id INT PRIMARY KEY,
    delivery_ID INT,
    order_ID INT UNIQUE,
    FOREIGN KEY (delivery_ID) REFERENCES Deliveries(delivery_ID),
    FOREIGN KEY (order_ID) REFERENCES Orders(order_ID)
);
```

5. Adăugați informații coerente

Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).

```
CREATE SEQUENCE customer_seq
START WITH     1
INCREMENT BY   1
NOCACHE
NOCYCLE;
INSERT INTO CUSTOMERS
(CUSTOMER_ID, FIRST_NAME, LAST_NAME, PHONE_NUMBER, EMAIL)
VALUES
(customer_seq.nextval, 'Miruna', 'Popa', '0712345678', 'miroonapupa@unidrep.ro');
INSERT INTO CUSTOMERS
(CUSTOMER_ID, FIRST_NAME, LAST_NAME, PHONE_NUMBER, EMAIL)
VALUES
(customer_seq.nextval, 'No', 'Purchases', '0712345268', 'brokeboy@s.unibuc.ro');
INSERT INTO CUSTOMERS
(CUSTOMER_ID, FIRST_NAME, LAST_NAME, PHONE_NUMBER, EMAIL)
VALUES
(customer_seq.nextval, 'Bogdan', 'Popel', '0722345678', 'second@unidrep.ro');
INSERT INTO CUSTOMERS
(CUSTOMER_ID, FIRST_NAME, LAST_NAME, PHONE_NUMBER, EMAIL)
VALUES
(customer_seq.nextval, 'Andreea', 'Valentina', '0732345678', 'third@unidrep.ro');
INSERT INTO CUSTOMERS
(CUSTOMER_ID, FIRST_NAME, LAST_NAME, PHONE_NUMBER, EMAIL)
VALUES
(customer_seq.nextval, 'Matei', 'Lupascu', '0742345678', 'fourth@unidrep.ro');
INSERT INTO CUSTOMERS
(CUSTOMER_ID, FIRST_NAME, LAST_NAME, PHONE_NUMBER, EMAIL)
VALUES
(customer_seq.nextval, 'Andrei', 'Lupascu', '0752345678', 'fifth@unidrep.ro');
INSERT INTO CUSTOMERS
```

```
(CUSTOMER_ID, FIRST_NAME, LAST_NAME, PHONE_NUMBER, EMAIL)
VALUES
(customer_seq.nextval, 'Besoiu', 'Sebastian', '0762345678', 'sixth@unidrep.ro');
INSERT INTO CUSTOMERS
(CUSTOMER_ID, FIRST_NAME, LAST_NAME, PHONE_NUMBER, EMAIL)
VALUES
(customer_seq.nextval, 'Alexandra', 'Raileanu', '0772345678', 'seventh@unidrep.ro');
INSERT INTO CUSTOMERS
(CUSTOMER_ID, FIRST_NAME, LAST_NAME, PHONE_NUMBER, EMAIL)
VALUES
(customer_seq.nextval, 'Valentina', 'Antonescu', '0782345678', 'eighth@unidrep.ro');
INSERT INTO CUSTOMERS
(CUSTOMER_ID, FIRST_NAME, LAST_NAME, PHONE_NUMBER, EMAIL)
VALUES
(customer_seq.nextval, 'Mihai', 'Eminescu', '0792345678', 'noNine@unidrep.ro');
INSERT INTO CUSTOMERS
(CUSTOMER_ID, FIRST_NAME, LAST_NAME, PHONE_NUMBER, EMAIL)
VALUES
(customer_seq.nextval, 'Raul', 'Voinea', '0702345678', 'finalissima@unidrep.ro');

CREATE SEQUENCE job_seq
START WITH     1
INCREMENT BY   1
NOCACHE
NOCYCLE;
INSERT INTO JOBS
(JOB_ID, JOB_NAME, SALARY)
VALUES
(job_seq.nextval, 'Software Engineer', 8000);
INSERT INTO JOBS
(JOB_ID, JOB_NAME, SALARY)
VALUES
(job_seq.nextval, 'Driver', 5000);
INSERT INTO JOBS
(JOB_ID, JOB_NAME, SALARY)
VALUES
(job_seq.nextval, 'Hardware Specialist', 7000);
INSERT INTO JOBS
(JOB_ID, JOB_NAME, SALARY)
VALUES
(job_seq.nextval, 'CEO', 2500);
INSERT INTO JOBS
(JOB_ID, JOB_NAME, SALARY)
VALUES
(job_seq.nextval, 'Intern', 2000);

CREATE SEQUENCE employee_seq
START WITH     1
INCREMENT BY   1
NOCACHE
NOCYCLE;
 INSERT INTO EMPLOYEES
(EMPLOYEE_ID, FIRST_NAME, LAST_NAME, JOB_ID)
VALUES
```

```
(employee_seq.nextval, 'Josh', 'Joshua', 4);
 INSERT INTO EMPLOYEES
(EMPLOYEE_ID, FIRST_NAME, LAST_NAME, JOB_ID)
VALUES
(employee_seq.nextval, 'George', 'Vanuta', 1);
 INSERT INTO EMPLOYEES
(EMPLOYEE_ID, FIRST_NAME, LAST_NAME, JOB_ID)
VALUES
(employee_seq.nextval, 'Malina', 'Ciocirlan', 1);
 INSERT INTO EMPLOYEES
(EMPLOYEE_ID, FIRST_NAME, LAST_NAME, JOB_ID)
VALUES
(employee_seq.nextval, 'Arghir', 'Valeriu', 2);
 INSERT INTO EMPLOYEES
(EMPLOYEE_ID, FIRST_NAME, LAST_NAME, JOB_ID)
VALUES
(employee_seq.nextval, 'Mircea', 'Valentin', 3);
 INSERT INTO EMPLOYEES
(EMPLOYEE_ID, FIRST_NAME, LAST_NAME, JOB_ID)
VALUES
(employee_seq.nextval, 'Ionut Calin', 'Iamandoiu', 5);

CREATE SEQUENCE weight_seq
START WITH     1
INCREMENT BY   1
NOCACHE
NOCYCLE;
INSERT INTO WEIGHT_CATEGORY
(CATEGORY_ID, DESCRIPTION)
VALUES
(weight_seq.nextval, 'Under 1kg');
INSERT INTO WEIGHT_CATEGORY
(CATEGORY_ID, DESCRIPTION)
VALUES
(weight_seq.nextval, '1-5kg');
INSERT INTO WEIGHT_CATEGORY
(CATEGORY_ID, DESCRIPTION)
VALUES
(weight_seq.nextval, '5-10kg');
INSERT INTO WEIGHT_CATEGORYp
(CATEGORY_ID, DESCRIPTION)
VALUES
(weight_seq.nextval, '10-15kg');
INSERT INTO WEIGHT_CATEGORY
(CATEGORY_ID, DESCRIPTION)
VALUES
(weight_seq.nextval, 'Over 15kg');

CREATE SEQUENCE supplier_seq
START WITH     1
INCREMENT BY   1
NOCACHE
NOCYCLE;
INSERT INTO SUPPLIERS
```

```
(SUPPLIER_ID, SUPPLIER_NAME, ADDRESS, PHONE)
VALUES
(supplier_seq.nextval, 'Lenovo', 'Calea Floreasca 169A, Bucure?ti 077190', '+40742 726
843');
INSERT INTO SUPPLIERS
(SUPPLIER_ID, SUPPLIER_NAME, ADDRESS, PHONE)
VALUES
(supplier_seq.nextval, 'Apple', 'Grant Shopping Center, ?oseaua Virtu?ii 148', '021 200
5200');
INSERT INTO SUPPLIERS
(SUPPLIER_ID, SUPPLIER_NAME, ADDRESS, PHONE)
VALUES
(supplier_seq.nextval, 'Microsoft', 'Bulevardul Iuliu Maniu. 6P Cl?direa Campus, Bucure?ti
061103', '(800) 426-9400');
INSERT INTO SUPPLIERS
(SUPPLIER_ID, SUPPLIER_NAME, ADDRESS, PHONE)
VALUES
(supplier_seq.nextval, 'Samsung', 'Bulevardul Iuliu Maniu 7', '021 326 6063');
INSERT INTO SUPPLIERS
(SUPPLIER_ID, SUPPLIER_NAME, ADDRESS, PHONE)
VALUES
(supplier_seq.nextval, 'Nvidia', '2788 San Tomas Expressway Santa Clara, CA 95051', '+1
(408) 486-2000');

CREATE SEQUENCE product_seq
START WITH     1
INCREMENT BY   1
NOCACHE
NOCYCLE;
INSERT INTO PRODUCTS
(PRODUCT_ID, SUPPLIER_ID, CATEGORY_ID, PRODUCT_NAME, PRICE)
VALUES
(product_seq.nextval, 1, 1, 'Mouse Wireless Gaming', 500);
INSERT INTO PRODUCTS
(PRODUCT_ID, SUPPLIER_ID, CATEGORY_ID, PRODUCT_NAME, PRICE)
VALUES
(product_seq.nextval, 1, 2, 'Cooler Laptop Metalic', 300);
INSERT INTO PRODUCTS
(PRODUCT_ID, SUPPLIER_ID, CATEGORY_ID, PRODUCT_NAME, PRICE)
VALUES
(product_seq.nextval, 2, 1, 'Mouse Lifestyle', 1000);
INSERT INTO PRODUCTS
(PRODUCT_ID, SUPPLIER_ID, CATEGORY_ID, PRODUCT_NAME, PRICE)
VALUES
(product_seq.nextval, 2, 1, 'Overpriced smartphone', 15000);
INSERT INTO PRODUCTS
(PRODUCT_ID, SUPPLIER_ID, CATEGORY_ID, PRODUCT_NAME, PRICE)
VALUES
(product_seq.nextval, 2, 1, 'Overpriced laptop', 35000);
INSERT INTO PRODUCTS
(PRODUCT_ID, SUPPLIER_ID, CATEGORY_ID, PRODUCT_NAME, PRICE)
VALUES
(product_seq.nextval, 3, 1, 'WebCam UHD', 1000);
INSERT INTO PRODUCTS
```

```
(PRODUCT_ID, SUPPLIER_ID, CATEGORY_ID, PRODUCT_NAME, PRICE)
VALUES
(product_seq.nextval, 3, 3, 'Surface Laptop 5 15 inch', 5000);
INSERT INTO PRODUCTS
(PRODUCT_ID, SUPPLIER_ID, CATEGORY_ID, PRODUCT_NAME, PRICE)
VALUES
(product_seq.nextval, 3, 3, 'HoloLens 2', 17500);
INSERT INTO PRODUCTS
(PRODUCT_ID, SUPPLIER_ID, CATEGORY_ID, PRODUCT_NAME, PRICE)
VALUES
(product_seq.nextval, 4, 1, 'S20 FE', 2500);
INSERT INTO PRODUCTS
(PRODUCT_ID, SUPPLIER_ID, CATEGORY_ID, PRODUCT_NAME, PRICE)
VALUES
(product_seq.nextval, 4, 1, 'S20', 4500);
INSERT INTO PRODUCTS
(PRODUCT_ID, SUPPLIER_ID, CATEGORY_ID, PRODUCT_NAME, PRICE)
VALUES
(product_seq.nextval, 4, 4, '55 inch Odyssey Ark 4K', 10000);
INSERT INTO PRODUCTS
(PRODUCT_ID, SUPPLIER_ID, CATEGORY_ID, PRODUCT_NAME, PRICE)
VALUES
(product_seq.nextval, 5, 1, 'GTX 1660Ti', 2000);
INSERT INTO PRODUCTS
(PRODUCT_ID, SUPPLIER_ID, CATEGORY_ID, PRODUCT_NAME, PRICE)
VALUES
(product_seq.nextval, 5, 1, 'GeForce RTX 4080', 8000);

CREATE SEQUENCE order_seq
START WITH     1
INCREMENT BY   1
NOCACHE
NOCYCLE;
 INSERT INTO ORDERS
(ORDER_ID, CUSTOMER_ID, ORDER_DATE)
VALUES
(order_seq.nextval, 1, DATE '2022-11-14');
 INSERT INTO ORDERS
(ORDER_ID, CUSTOMER_ID, ORDER_DATE)
VALUES
(order_seq.nextval, 1, DATE '2022-11-15');
 INSERT INTO ORDERS
(ORDER_ID, CUSTOMER_ID, ORDER_DATE)
VALUES
(order_seq.nextval, 2, DATE '2022-11-14');
 INSERT INTO ORDERS
(ORDER_ID, CUSTOMER_ID, ORDER_DATE)
VALUES
(order_seq.nextval, 3, DATE '2022-11-16');
 INSERT INTO ORDERS
(ORDER_ID, CUSTOMER_ID, ORDER_DATE)
VALUES
(order_seq.nextval, 4, DATE '2022-11-18');
 INSERT INTO ORDERS
```

```
(ORDER_ID, CUSTOMER_ID, ORDER_DATE)
VALUES
(order_seq.nextval, 5, DATE '2022-11-11');
 INSERT INTO ORDERS
(ORDER_ID, CUSTOMER_ID, ORDER_DATE)
VALUES
(order_seq.nextval, 6, DATE '2022-11-11');
 INSERT INTO ORDERS
(ORDER_ID, CUSTOMER_ID, ORDER_DATE)
VALUES
(order_seq.nextval, 7, DATE '2022-11-11');
 INSERT INTO ORDERS
(ORDER_ID, CUSTOMER_ID, ORDER_DATE)
VALUES
(order_seq.nextval, 8, DATE '2022-11-19');
 INSERT INTO ORDERS
(ORDER_ID, CUSTOMER_ID, ORDER_DATE)
VALUES
(order_seq.nextval, 9, DATE '2022-11-30');
 INSERT INTO ORDERS
(ORDER_ID, CUSTOMER_ID, ORDER_DATE)
VALUES
(order_seq.nextval, 10, DATE '2022-11-15');
 INSERT INTO ORDERS
(ORDER_ID, CUSTOMER_ID, ORDER_DATE)
VALUES
(order_seq.nextval, 10, DATE '2022-11-16');
 INSERT INTO ORDERS
(ORDER_ID, CUSTOMER_ID, ORDER_DATE)
VALUES
(order_seq.nextval, 9, DATE '2022-11-20');

INSERT INTO ORDER_LINES
(ORDER_ID, PRODUCT_ID, QUANTITY)
VALUES
(1, 1, 1);
 INSERT INTO ORDER_LINES
(ORDER_ID, PRODUCT_ID, QUANTITY)
VALUES
(1, 2, 1);
 INSERT INTO ORDER_LINES
(ORDER_ID, PRODUCT_ID, QUANTITY)
VALUES
(2, 5, 1);
 INSERT INTO ORDER_LINES
(ORDER_ID, PRODUCT_ID, QUANTITY)
VALUES
(3, 4, 2);
 INSERT INTO ORDER_LINES
(ORDER_ID, PRODUCT_ID, QUANTITY)
VALUES
(4, 6, 28);
 INSERT INTO ORDER_LINES
(ORDER_ID, PRODUCT_ID, QUANTITY)
```

```
VALUES
(5, 11, 1);
 INSERT INTO ORDER_LINES
(ORDER_ID, PRODUCT_ID, QUANTITY)
VALUES
(6, 12, 1);
 INSERT INTO ORDER_LINES
(ORDER_ID, PRODUCT_ID, QUANTITY)
VALUES
(6, 13, 5);
 INSERT INTO ORDER_LINES
(ORDER_ID, PRODUCT_ID, QUANTITY)
VALUES
(7, 10, 1);
 INSERT INTO ORDER_LINES
(ORDER_ID, PRODUCT_ID, QUANTITY)
VALUES
(8, 9, 1);
 INSERT INTO ORDER_LINES
(ORDER_ID, PRODUCT_ID, QUANTITY)
VALUES
(9, 1, 3);
 INSERT INTO ORDER_LINES
(ORDER_ID, PRODUCT_ID, QUANTITY)
VALUES
(10, 1, 1);
 INSERT INTO ORDER_LINES
(ORDER_ID, PRODUCT_ID, QUANTITY)
VALUES
(11, 2, 1);
 INSERT INTO ORDER_LINES
(ORDER_ID, PRODUCT_ID, QUANTITY)
VALUES
(11, 3, 1);
INSERT INTO ORDER_LINES
(ORDER_ID, PRODUCT_ID, QUANTITY)
VALUES
(11, 6, 1);
INSERT INTO ORDER_LINES
(ORDER_ID, PRODUCT_ID, QUANTITY)
VALUES
(11, 12, 1);
INSERT INTO ORDER_LINES
(ORDER_ID, PRODUCT_ID, QUANTITY)
VALUES
(12, 4, 2);
INSERT INTO ORDER_LINES
(ORDER_ID, PRODUCT_ID, QUANTITY)
VALUES
(12, 1, 3);
INSERT INTO ORDER_LINES
(ORDER_ID, PRODUCT_ID, QUANTITY)
VALUES
(13, 2, 2);
```

```
CREATE SEQUENCE extra_task_seq
START WITH     1
INCREMENT BY   1
NOCACHE
NOCYCLE;
 INSERT INTO EXTRA_TASKS
(TASK_ID, EMPLOYEE_ID, TASK_NAME, BONUS)
VALUES
(extra_task_seq.nextval, 5, 'Deliver order with id = 10', 0.1);
 INSERT INTO EXTRA_TASKS
(TASK_ID, EMPLOYEE_ID, TASK_NAME, BONUS)
VALUES
(extra_task_seq.nextval, 1, 'Create Teams/Discord for IT', 0);
INSERT INTO EXTRA_TASKS
(TASK_ID, EMPLOYEE_ID, TASK_NAME, BONUS)
VALUES
(extra_task_seq.nextval, 1, 'Find Investors', 1);
 INSERT INTO EXTRA_TASKS
(TASK_ID, EMPLOYEE_ID, TASK_NAME, BONUS)
VALUES
(extra_task_seq.nextval, 1, 'Get Malina a work car', 0.1);
INSERT INTO EXTRA_TASKS
(TASK_ID, EMPLOYEE_ID, TASK_NAME, BONUS)
VALUES
(extra_task_seq.nextval, 5, 'Become Prolific in Oracle SQL', 0.5);

CREATE SEQUENCE delivery_seq
START WITH     1
INCREMENT BY   1
NOCACHE
NOCYCLE;
INSERT INTO DELIVERIES
(DELIVERY_ID, EMPLOYEE_ID, ADRESS)
VALUES
(delivery_seq.nextval, 4, 'Buzau');
 INSERT INTO DELIVERIES
(DELIVERY_ID, EMPLOYEE_ID, ADRESS)
VALUES
(delivery_seq.nextval,4, 'Braila');
 INSERT INTO DELIVERIES
(DELIVERY_ID, EMPLOYEE_ID, ADRESS)
VALUES
(delivery_seq.nextval,4, 'Bucuresti');
 INSERT INTO DELIVERIES
(DELIVERY_ID, EMPLOYEE_ID, ADRESS)
VALUES
(delivery_seq.nextval,4, 'Brasov');
 INSERT INTO DELIVERIES
(DELIVERY_ID, EMPLOYEE_ID, ADRESS)
VALUES
(delivery_seq.nextval,4, 'Galati');
INSERT INTO DELIVERIES
```

```
(DELIVERY_ID, EMPLOYEE_ID, ADRESS)
VALUES
(delivery_seq.nextval, 5, 'Satu-Mare');

CREATE SEQUENCE del_items_seq
START WITH     1
INCREMENT BY   1
NOCACHE
NOCYCLE;
INSERT INTO DELIVERY_ITEMS
(DELIVERY_ITEM_ID, DELIVERY_ID, ORDER_ID)
VALUES
(del_items_seq.nextval, 1, 6);
INSERT INTO DELIVERY_ITEMS
(DELIVERY_ITEM_ID, DELIVERY_ID, ORDER_ID)
VALUES
(del_items_seq.nextval,1,7);
INSERT INTO DELIVERY_ITEMS
(DELIVERY_ITEM_ID, DELIVERY_ID, ORDER_ID)
VALUES
(del_items_seq.nextval,1,8);
INSERT INTO DELIVERY_ITEMS
(DELIVERY_ITEM_ID, DELIVERY_ID, ORDER_ID)
VALUES
(del_items_seq.nextval,2,3);
INSERT INTO DELIVERY_ITEMS
(DELIVERY_ITEM_ID, DELIVERY_ID, ORDER_ID)
VALUES
(del_items_seq.nextval,2,11);
INSERT INTO DELIVERY_ITEMS
(DELIVERY_ITEM_ID, DELIVERY_ID, ORDER_ID)
VALUES
(del_items_seq.nextval,3,1);
INSERT INTO DELIVERY_ITEMS
(DELIVERY_ITEM_ID, DELIVERY_ID, ORDER_ID)
VALUES
(del_items_seq.nextval,3,2);
INSERT INTO DELIVERY_ITEMS
(DELIVERY_ITEM_ID, DELIVERY_ID, ORDER_ID)
VALUES
(del_items_seq.nextval,4,4);
INSERT INTO DELIVERY_ITEMS
(DELIVERY_ITEM_ID, DELIVERY_ID, ORDER_ID)
VALUES
(del_items_seq.nextval,4,5);
INSERT INTO DELIVERY_ITEMS
(DELIVERY_ITEM_ID, DELIVERY_ID, ORDER_ID)
VALUES
(del_items_seq.nextval,5,9);
INSERT INTO DELIVERY_ITEMS
(DELIVERY_ITEM_ID, DELIVERY_ID, ORDER_ID)
VALUES
(del_items_seq.nextval,6,10);
INSERT INTO DELIVERY_ITEMS
```

```
(DELIVERY_ITEM_ID, DELIVERY_ID, ORDER_ID)
VALUES
(del_items_seq.nextval,5,12);
INSERT INTO DELIVERY_ITEMS
(DELIVERY_ITEM_ID, DELIVERY_ID, ORDER_ID)
VALUES
(del_items_seq.nextval,2,13);
```

6. Două tipuri diferite de colecții studiate

Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze două tipuri diferite de colecții studiate. Apelați subprogramul.

PROBLEMA:

Realizati un subprogram stocat independent care, pentru fiecare client, calculează și afișează totalul cheltuielilor sale pe comenzi.

```
CREATE OR REPLACE PROCEDURE TOTAL_SPENT AS
   TYPE OrderTotalArray IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
-- max size
   TYPE CustomerIDs IS VARRAY(2147483647) OF INT;
   customer_IDs CustomerIDs := CustomerIDs();
   order_totals OrderTotalArray;
BEGIN
   SELECT customer_ID BULK COLLECT INTO customer_IDs FROM Customers;

   FOR i IN 1..customer_IDs.COUNT LOOP
      SELECT SUM(p.price * ol.quantity)
      INTO order_totals(i)
      FROM ORDERS o
      JOIN ORDER_LINES ol ON o.order_ID = ol.order_ID
      JOIN PRODUCTS p ON ol.product_ID = p.product_ID
      WHERE o.customer_ID = customer_IDs(i);
   END LOOP;

   FOR i IN 1..customer_IDs.COUNT LOOP
      DBMS_OUTPUT.PUT_LINE('The customer with ID = ' || customer_IDs(i) || ' has spent
on orders in total: ' || order_totals(i));
   END LOOP;
END;
/

BEGIN
TOTAL_SPENT;
END;
```

```
/
```

```
Procedure TOTAL_SPENT compiled

The customer with ID = 1 has spent on orders in total: 35800
The customer with ID = 2 has spent on orders in total: 30000
The customer with ID = 3 has spent on orders in total: 28000
The customer with ID = 4 has spent on orders in total: 10000
The customer with ID = 5 has spent on orders in total: 42000
The customer with ID = 6 has spent on orders in total: 4500
The customer with ID = 7 has spent on orders in total: 2500
The customer with ID = 8 has spent on orders in total: 1500
The customer with ID = 9 has spent on orders in total: 1100
The customer with ID = 10 has spent on orders in total: 35800
The customer with ID = 11 has spent on orders in total:


PL/SQL procedure successfully completed.
```

7. Doua tipuri diferite de cursoare studiate

Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat. Apelați subprogramul.

PROBLEMA:

Realizati un subprogram stocat independent care afișează, pentru un client cu id-ul cunoscut, detaliile livrărilor: către acesta: ce produse, câte produse, cine a livrat și ce job_id are, când a fost plasată comanda. Am primit plangeri de la anumiți clienți cu privire la starea anumitor produse. Încerc sa aflu ce s-a intamplat.

```
CREATE OR REPLACE PROCEDURE
GET_CUSTOMER_ORDER_DETAILS(customer_id_input IN
CUSTOMERS.customer_ID%TYPE) AS
   -- cursor parametrizat
   CURSOR order_cursor(customer_id_input IN CUSTOMERS.customer_ID%TYPE) IS
      SELECT o.order_ID, o.order_date, p.product_name, ol.quantity, d.employee_ID
      FROM ORDERS o
      JOIN ORDER_LINES ol ON o.order_ID = ol.order_ID
      JOIN PRODUCTS p ON ol.product_ID = p.product_ID
      JOIN DELIVERY_ITEMS di ON ol.order_ID = di.order_ID
      JOIN DELIVERIES d ON di.delivery_ID = d.delivery_ID
CREATE OR REPLACE PROCEDURE
GET_CUSTOMER_ORDER_DETAILS(customer_id_input IN
CUSTOMERS.customer_ID%TYPE) AS
   -- cursor parametrizat
```

```
    CURSOR order_cursor(customer_id_input IN CUSTOMERS.customer_ID%TYPE) IS
        SELECT o.order_ID, o.order_date, p.product_name, ol.quantity, d.employee_ID
        FROM ORDERS o
        JOIN ORDER_LINES ol ON o.order_ID = ol.order_ID
        JOIN PRODUCTS p ON ol.product_ID = p.product_ID
        JOIN DELIVERY_ITEMS di ON ol.order_ID = di.order_ID
        JOIN DELIVERIES d ON di.delivery_ID = d.delivery_ID
        WHERE o.customer_ID = customer_id_input;
    -- cursor dinamic
    dynamic_cursor SYS_REFCURSOR;
    -- pentru cursor dinamic
    emp_first_name EMPLOYEES.first_name%TYPE;
    emp_last_name EMPLOYEES.last_name%TYPE;
    emp_job_id EMPLOYEES.job_id%TYPE;
    -- pentru cursor parametrizat
    ord_order_id ORDERS.order_ID%TYPE;
    ord_order_date ORDERS.order_date%TYPE;
    pro_product_name PRODUCTS.product_name%TYPE;
    orl_quantity ORDER_LINES.quantity%TYPE;
    del_employee_id DELIVERIES.employee_ID%TYPE;

BEGIN
    OPEN order_cursor(customer_id_input);
    LOOP
        FETCH order_cursor INTO ord_order_id, ord_order_date, pro_product_name,
orl_quantity, del_employee_id;
        EXIT WHEN order_cursor%NOTFOUND;

        OPEN dynamic_cursor FOR 'SELECT first_name, last_name, job_id
                                 FROM EMPLOYEES
                                 WHERE employee_ID = :employee_id'
                            USING del_employee_id;
        FETCH dynamic_cursor INTO emp_first_name, emp_last_name, emp_job_id;
        CLOSE dynamic_cursor;
        DBMS_OUTPUT.PUT_LINE('The product: ' || pro_product_name || ', in a quantity of '
|| orl_quantity||
                             ', part of order with ID: ' || ord_order_id || ', from: ' || ord_order_date ||':');
        DBMS_OUTPUT.PUT_LINE('  ->Was delivered by ' || emp_first_name || ' ' ||
emp_last_name ||
                             ', with job ID: ' || emp_job_id||';');
    END LOOP;
    CLOSE order_cursor;
END;
/

DECLARE
    customer_id_cautat CUSTOMERS.customer_ID%TYPE := 10;
    customer_id_cautat2 CUSTOMERS.customer_ID%TYPE := 9;
BEGIN
    DBMS_OUTPUT.PUT_LINE('--------> Customer with id = ' || customer_id_cautat ||
'<--------');
    GET_CUSTOMER_ORDER_DETAILS(customer_id_cautat);
    DBMS_OUTPUT.PUT_LINE('');
```

```
DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------');
   DBMS_OUTPUT.PUT_LINE('');
   DBMS_OUTPUT.PUT_LINE('--------> Customer with id = ' || customer_id_cautat2 ||
'<--------');
   GET_CUSTOMER_ORDER_DETAILS(customer_id_cautat2);
END;
/
```

```
Procedure GET_CUSTOMER_ORDER_DETAILS compiled

--------> Customer with id = 10<--------
The product: Mouse Wireless Gaming, in a quantity of 3, part of order with ID: 12, from: 16-NOV-22:
  ->Was delivered by Arghir Valeriu, with job ID: 2;
The product: Cooler Laptop Metalic, in a quantity of 1, part of order with ID: 11, from: 15-NOV-22:
  ->Was delivered by Arghir Valeriu, with job ID: 2;
The product: Mouse Lifestyle, in a quantity of 1, part of order with ID: 11, from: 15-NOV-22:
  ->Was delivered by Arghir Valeriu, with job ID: 2;
The product: Overpriced smartphone, in a quantity of 2, part of order with ID: 12, from: 16-NOV-22:
  ->Was delivered by Arghir Valeriu, with job ID: 2;
The product: WebCam UHD, in a quantity of 1, part of order with ID: 11, from: 15-NOV-22:
  ->Was delivered by Arghir Valeriu, with job ID: 2;
The product: GTX 1660Ti, in a quantity of 1, part of order with ID: 11, from: 15-NOV-22:
  ->Was delivered by Arghir Valeriu, with job ID: 2;

-------------------------------------------------------------------------------

--------> Customer with id = 9<--------
The product: Mouse Wireless Gaming, in a quantity of 1, part of order with ID: 10, from: 30-NOV-22:
  ->Was delivered by Mircea Valentin, with job ID: 3;
The product: Cooler Laptop Metalic, in a quantity of 2, part of order with ID: 13, from: 20-NOV-22:
  ->Was delivered by Arghir Valeriu, with job ID: 2;
```

8. Subprogram stocat independent de tip funcție

Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definiți minim 2 excepții. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

PROBLEMA:

Pentru un angajat cu id-ul dat, creați un subprogram stocat independent de tip funcție care afișează în limbaj natural salariul de baza, limita superioară a veniturilor generate de bonusuri și limita superioară a salariului la final de luna. Încercați sa oferiti sugestii dacă valorile obținute nu par normale, prin output.

```
CREATE OR REPLACE PACKAGE user_defined_exceptions IS
   invalid_employee_id EXCEPTION;
   PRAGMA EXCEPTION_INIT(invalid_employee_id, -20001);
   too_high_potential EXCEPTION;
   PRAGMA EXCEPTION_INIT( too_high_potential, -20002);
   only_one_customer EXCEPTION;
   PRAGMA EXCEPTION_INIT(only_one_customer, -20003);
```

```
    customer_with_no_purchases EXCEPTION;
    PRAGMA EXCEPTION_INIT(customer_with_no_purchases, -20004);
    not_on_schedule EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_on_schedule, -20005);
    jobs_are_frozen EXCEPTION;
    PRAGMA EXCEPTION_INIT(jobs_are_frozen, -20006);

END user_defined_exceptions;
/

CREATE OR REPLACE FUNCTION
CALCULATE_MONTHLY_SALARY(employee_ID_input IN INT)
RETURN VARCHAR2
IS
    maximum_income NUMBER;
    possible_gain NUMBER:=0;
    base_salary NUMBER;
    employees_with_required_id NUMBER;
BEGIN

    SELECT COUNT(*)
    INTO employees_with_required_id
    FROM employees
    WHERE employee_id = employee_ID_input;

    IF employees_with_required_id  < 1 THEN
        RAISE user_defined_exceptions.invalid_employee_id;
    END IF;
    IF employees_with_required_id  < 1 THEN
        RAISE TOO_MANY_ROWS;
    END IF;

    SELECT (j.salary + NVL(j.salary * SUM(et.bonus), 0)), NVL(j.salary * SUM(et.bonus),
0), j.salary
    INTO maximum_income, possible_gain, base_salary
    FROM EMPLOYEES e
    RIGHT JOIN JOBS j ON e.job_id = j.job_id
    LEFT JOIN EXTRA_TASKS et ON e.employee_ID = et.employee_ID
    WHERE e.employee_ID = employee_ID_input
    GROUP BY j.salary, e.employee_ID;

    IF base_salary <  possible_gain THEN
        RAISE user_defined_exceptions.too_high_potential;
    END IF;

    RETURN 'Employee with ID: '||employee_ID_input||
    ' has a base salary of '||base_salary||', can earn '||possible_gain||' from bonuses,
resulting in a '||maximum_income||' maximum income.';

EXCEPTION
    WHEN user_defined_exceptions.invalid_employee_id THEN
        RETURN 'Employee with ID: '||employee_ID_input||' -> INVALID EMPLOYEE_ID';
    WHEN user_defined_exceptions.too_high_potential THEN
```

```
        SELECT count(*)
        INTO possible_gain
        FROM EMPLOYEES e
        LEFT JOIN EXTRA_TASKS et ON e.employee_ID = et.employee_ID
        WHERE e.employee_ID = employee_ID_input;

        SELECT sum(et.bonus)*100
        INTO maximum_income
        FROM EMPLOYEES e
        LEFT JOIN EXTRA_TASKS et ON e.employee_ID = et.employee_ID
        WHERE e.employee_ID = employee_ID_input;

        RETURN 'Employee with ID: '||employee_ID_input||'. Too many tasks ->
'||possible_gain||' ? Too rewarding tasks -> +'||maximum_income ||'% ? Should i increase
his salary('||base_salary|| ')? Max income =
'||TO_CHAR(base_salary+base_salary*maximum_income/100);
END;
/

DECLARE
    employee_ID1 INT := 5; -- nu are exceptii
    employee_ID2 INT := 12; -- invalid_employee_id
    employee_ID3 INT := 1; -- are bonusuri prea mari/prea multe
    sumary VARCHAR2(400);
BEGIN
    sumary := CALCULATE_MONTHLY_SALARY(employee_ID1);
    DBMS_OUTPUT.PUT_LINE(sumary);
    sumary := CALCULATE_MONTHLY_SALARY(employee_ID2);
    DBMS_OUTPUT.PUT_LINE(sumary);
    sumary := CALCULATE_MONTHLY_SALARY(employee_ID3);
    DBMS_OUTPUT.PUT_LINE(sumary);
END;
/
```

```
Package USER_DEFINED_EXCEPTIONS compiled


Function CALCULATE_MONTHLY_SALARY compiled

Employee with ID: 5. Too many tasks -> 2 ? Too rewarding tasks -> +100% ? Should i increase his salary(2000)? Max income = 4000
Employee with ID: 12 -> INVALID EMPLOYEE_ID
Employee with ID: 1. Too many tasks -> 3 ? Too rewarding tasks -> +110% ? Should i increase his salary(2500)? Max income = 5250
```

9. 5 dintre tabelele definite

Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile NO_DATA_FOUND și TOO_MANY_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

PROBLEMA:

Realizati un subprogram stocat independent de tip procedură care sa genereze un raport pentru un utilizator cu un id dat în care sa se afiseze informații care permit o analiză comparativă asupra produselor comandate din perspectiva veniturilor generate, categoriilor de greutate, numărului de bucăți achizitionate. Un astfel de raport deschide perspectiva unei

dezvoltări eficiente pe viitor. Putem afla ce vehicule sunt potrivite pentru companie, ce produse sunt profitabile d.p.d.v al raportului greutate/venit generat etc..

```
CREATE OR REPLACE PROCEDURE
calculate_customer_order_total(customer_id_input IN Customers.customer_ID%TYPE)
IS
   in_aux_revenue NUMBER;in_aux_quantity NUMBER;in_aux_desc
weight_category.description%TYPE;
   in_aux_total_revenue NUMBER := 0;in_aux_total_quantity NUMBER := 0;

   rest_aux_revenue NUMBER;rest_aux_quantity NUMBER;rest_aux_desc
weight_category.description%TYPE;
   rest_aux_total_revenue NUMBER := 0;rest_aux_total_quantity NUMBER := 0;

   CURSOR purchases_per_category is
      SELECT w.category_ID, w.description,
         NVL(SUM(ol.quantity), 0) AS products_no,
         NVL(SUM(ol.quantity * p.price), 0) AS revenue
      FROM Weight_Category w
      LEFT JOIN Products p ON w.category_ID = p.category_ID
      LEFT JOIN Order_Lines ol ON p.product_ID = ol.product_ID
      LEFT JOIN Orders o ON ol.order_ID = o.order_ID
      LEFT JOIN Customers c ON o.customer_ID = c.customer_ID
      WHERE c.customer_ID = customer_id_input
      GROUP BY w.category_ID, w.description;

   CURSOR in_order_sumary IS
      SELECT w.description,
         NVL(Sumary_per_category.products_no, 0),
         NVL(Sumary_per_category.revenue, 0)
      FROM Weight_Category w
      LEFT JOIN (
         SELECT p.category_ID,
             SUM(ol.quantity) as products_no,
             SUM(ol.quantity * p.price) as revenue
         FROM Products p
         LEFT JOIN Order_Lines ol ON p.product_ID = ol.product_ID
         LEFT JOIN Orders o ON ol.order_ID = o.order_ID
         LEFT JOIN Customers c ON o.customer_ID = c.customer_ID
         WHERE c.customer_ID = customer_id_input
         GROUP BY p.category_ID
      ) Sumary_per_category ON w.category_ID = Sumary_per_category.category_ID
      ORDER BY w.category_ID;

   CURSOR rest_order_sumary IS
      SELECT w.description,
        NVL(SUM(ol.quantity), 0),
        NVL(SUM(ol.quantity * p.price), 0)
      FROM Weight_Category w
      JOIN Products p ON w.category_id = p.category_ID
      LEFT JOIN Order_Lines ol ON p.product_ID = ol.product_ID
      GROUP BY w.category_ID, w.description
      ORDER BY w.category_ID;
```

```
BEGIN

   SELECT count(*)
   INTO in_aux_revenue
   FROM Customers
   Where customer_id = customer_id_input;

   IF in_aux_revenue = 0 THEN
      RAISE NO_DATA_FOUND;
   ELSE IF in_aux_revenue > 1 THEN
         RAISE TOO_MANY_ROWS;
      END IF;
   END IF;

   SELECT count(*)
   INTO in_aux_revenue
   FROM Customers
   Where customer_id != customer_id_input;

   IF in_aux_revenue = 0 THEN
      RAISE user_defined_exceptions.only_one_customer;
   END IF;

   OPEN purchases_per_category;
   FETCH purchases_per_category INTO in_aux_revenue, in_aux_desc, in_aux_quantity,
rest_aux_revenue;
   IF purchases_per_category%NOTFOUND THEN
      RAISE user_defined_exceptions.customer_with_no_purchases;
   END IF;

   OPEN in_order_sumary;
   OPEN rest_order_sumary;

   LOOP
      FETCH in_order_sumary INTO in_aux_desc, in_aux_quantity, in_aux_revenue;
      FETCH rest_order_sumary INTO rest_aux_desc, rest_aux_quantity,
rest_aux_revenue;

      rest_aux_total_revenue := rest_aux_total_revenue + rest_aux_revenue;
      rest_aux_total_quantity := rest_aux_total_quantity + rest_aux_quantity;

      in_aux_total_revenue := in_aux_total_revenue + in_aux_revenue;
      in_aux_total_quantity := in_aux_total_quantity + in_aux_quantity;

      EXIT WHEN in_order_sumary%NOTFOUND;
      EXIT WHEN REST_order_sumary%NOTFOUND;

      IF rest_aux_quantity = 0 THEN
          DBMS_OUTPUT.PUT_LINE('For this category ('||rest_aux_desc||') no products
have been sold.');
      ELSE IF in_aux_quantity = 0 THEN
         DBMS_OUTPUT.PUT_LINE('For this category ('||rest_aux_desc||') total revenue =
'||rest_aux_revenue||' ,total products sold = '||rest_aux_quantity);
```

```
            DBMS_OUTPUT.PUT_LINE('');
            DBMS_OUTPUT.PUT_LINE('For this category ('||rest_aux_desc||'), the customer
with id '||customer_id_input||' did not buy any products');
            ELSE
                DBMS_OUTPUT.PUT_LINE('For this category ('||rest_aux_desc||') total revenue
= '||rest_aux_revenue||' ,total products sold = '||rest_aux_quantity);
                DBMS_OUTPUT.PUT_LINE('');
                DBMS_OUTPUT.PUT_LINE('Customer with id '||customer_id_input||' spent
'||in_aux_revenue||', by buying '||in_aux_quantity ||' products, from category
'||in_aux_desc);
                DBMS_OUTPUT.PUT_LINE('Contribution percentage:
'||TO_CHAR(ROUND(100*in_aux_revenue/rest_aux_revenue,2))||'% of revenue and '||
TO_CHAR(ROUND(100*in_aux_quantity/rest_aux_quantity,2)) ||' % of product units');
            END IF;
        END IF;

        DBMS_OUTPUT.PUT_LINE('');

DBMS_OUTPUT.PUT_LINE('----------------------------------------------------------------------------------------
------------------------');
        DBMS_OUTPUT.PUT_LINE('');
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('TOTAL Contribution percentage:
'||TO_CHAR(ROUND(100*in_aux_total_revenue/rest_aux_total_revenue))||'% of revenue
and '|| TO_CHAR(ROUND(100*in_aux_total_quantity/rest_aux_total_quantity)) ||' % of
product units');

CLOSE in_order_sumary;
CLOSE rest_order_sumary;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No customer_id = ' || customer_id_input);
    WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE('Duplicates for customer_ID = ' || customer_id_input);
    WHEN user_defined_exceptions.only_one_customer THEN
    DBMS_OUTPUT.PUT_LINE('The customer with id '||customer_id_input||' is your sole
customer.');
    WHEN user_defined_exceptions.customer_with_no_purchases THEN
    DBMS_OUTPUT.PUT_LINE('The customer with id '||customer_id_input||' has not made
any purchases. Why does he exist?');
    WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('ERROR');
END;
/
BEGIN
    calculate_customer_order_total(10);
    DBMS_OUTPUT.PUT_LINE('');

DBMS_OUTPUT.PUT_LINE('----------------------------------------------------------------------------------------
------------------------');
    DBMS_OUTPUT.PUT_LINE('');
    calculate_customer_order_total(100);
    DBMS_OUTPUT.PUT_LINE('');
```

```
DBMS_OUTPUT.PUT_LINE('----------------------------------------------------------------------------------------
-----------------------');
    DBMS_OUTPUT.PUT_LINE('');
    calculate_customer_order_total(11);
END;
/
```

```
--------------------------------------------------------------------------------------------------------------

For this category (Over 15kg) total revenue = 10000 ,total products sold = 1

For this category (Over 15kg), the customer with id 10 did not buy any products

--------------------------------------------------------------------------------------------------------------

TOTAL Contribution percentage: 18% of revenue and 16 % of product units

--------------------------------------------------------------------------------------------------------------

No customer_id = 100

--------------------------------------------------------------------------------------------------------------

The customer with id 11 has not made any purchases. Why does he exist?


PL/SQL procedure successfully completed.
Procedure SUMMARY_WEIGHT_REVENUE compiled

For this category (Under 1kg) total revenue = 145000 ,total products sold = 51

Customer with id 10 spent 35500, by buying 8 products, from category Under 1kg
Contribution percentage: 24.48% of revenue and 15.69 % of product units

--------------------------------------------------------------------------------------------------------------

For this category (1-5kg) total revenue = 1200 ,total products sold = 4

Customer with id 10 spent 300, by buying 1 products, from category 1-5kg
Contribution percentage: 25% of revenue and 25 % of product units

--------------------------------------------------------------------------------------------------------------

For this category (5-10kg) no products have been sold.

--------------------------------------------------------------------------------------------------------------

For this category (10-15kg) total revenue = 35000 ,total products sold = 1

For this category (10-15kg), the customer with id 10 did not buy any products

--------------------------------------------------------------------------------------------------------------
```

10. LMD la nivel de comandă

Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.

```
CREATE OR REPLACE TRIGGER LMD_comanda
BEFORE INSERT OR UPDATE OR DELETE ON Extra_Tasks
BEGIN
    IF TO_CHAR(SYSDATE,'HH24') NOT BETWEEN 8 AND 20 THEN
        RAISE_APPLICATION_ERROR(-20005,'not on schedule!');
    ELSIF TRUNC(SYSDATE) < TO_DATE('31-DEC-2023', 'DD-MON-YYYY') THEN
        RAISE_APPLICATION_ERROR(-20006,'Boss said that he tries to stop
micromanaging!');
    END IF;
```

```
    NULL;
EXCEPTION
    WHEN user_defined_exceptions.not_on_schedule THEN
        DBMS_OUTPUT.PUT_LINE('Work hours are between 8:00 and 20:00. Stop
Working');
        WHEN user_defined_exceptions.jobs_are_frozen THEN
        DBMS_OUTPUT.PUT_LINE('Just do the main tasks. Bonuses will be awarded for
good work.');
END;
/
 INSERT INTO EXTRA_TASKS
(TASK_ID, EMPLOYEE_ID, TASK_NAME, BONUS)
VALUES
(extra_task_seq.nextval, 1, 'Finish FMI', 0.1);
```

```
Trigger LMD_COMANDA compiled


Work hours are between 8:00 and 20:00. Stop Working
```

11. LMD la nivel de linie

Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.

```
CREATE OR REPLACE TRIGGER LOGGER_ORDERS
AFTER INSERT OR UPDATE OR DELETE ON Orders
FOR EACH ROW
DECLARE
action VARCHAR(20);
BEGIN
    IF INSERTING THEN
        action := 'INSERT';
    ELSIF UPDATING THEN
        action := 'UPDATE';
    ELSIF DELETING THEN
        action := 'DELETE';
    END IF;
DBMS_OUTPUT.PUT_LINE('Do not forget to insert the order lines.');
INSERT INTO LOGGER
(LOG_ID, TABLE_NAME, ACTION, TIME_MOMENT)
VALUES
(logger_seq.nextval, 'Orders', action, SYSDATE);
DBMS_OUTPUT.PUT_LINE(action||' on '|| ' ORDERS ' ||' on '|| SYSDATE || ' Order_id =
SELECT MAX(order_id) INTO last_order FROM Orders;');

END;
/
INSERT INTO ORDERS
(ORDER_ID, CUSTOMER_ID, ORDER_DATE)
VALUES
(order_seq.nextval, 7, DATE '2022-11-21');
```

```
Trigger LOGGER_ORDERS compiled

Do not forget to insert the order lines.
INSERT on  ORDERS  on 26-MAY-23 Order_id = SELECT MAX(order_id) INTO last_order FROM Orders;


1 row inserted.
```

12. LDD

Definiți un trigger de tip LDD. Declanșați trigger-ul.

```
CREATE OR REPLACE TRIGGER audit_schema_per_user
   AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN

   IF ORA_SYSEVENT = 'CREATE' THEN
      INSERT INTO Logger_user
      VALUES (SYS.LOGIN_USER,
      SYS.SYSEVENT, SYS.DICTIONARY_OBJ_TYPE,
      SYS.DICTIONARY_OBJ_NAME, SYSTIMESTAMP, 1);
   ELSIF ORA_SYSEVENT = 'DROP' THEN
      INSERT INTO Logger_user
      VALUES (SYS.LOGIN_USER,
      SYS.SYSEVENT, SYS.DICTIONARY_OBJ_TYPE,
      SYS.DICTIONARY_OBJ_NAME, SYSTIMESTAMP, 1);
   ELSIF ORA_SYSEVENT = 'ALTER' THEN
      -- presupunem ca leaderul are incredere ca stiu sa manipuleze logica tabelelor, dar
      --verifica create si drop pentru a se asigura ca inca are toate datele necesare
companiei.
      -- de asemena, verifica functii/triggeri etc. sa vada daca au sens
      INSERT INTO Logger_user
      VALUES (SYS.LOGIN_USER,
      SYS.SYSEVENT, SYS.DICTIONARY_OBJ_TYPE,
      SYS.DICTIONARY_OBJ_NAME, SYSTIMESTAMP, 0);
   END IF;
END;
/

CREATE TABLE test(
   test int primary key,
   data int
);
drop table test;
```

```
Trigger AUDIT_SCHEMA_PER_USER compiled
```

```
Table TEST created.


Table TEST dropped.
```

2	POPELEMILBOGDAN	CREATE	TABLE	TEST	26-MAY-23 09.50.53.914000000 PM	1
3	POPELEMILBOGDAN	DROP	TABLE	TEST	26-MAY-23 09.51.51.797000000 PM	1