

# Proiect SGBD

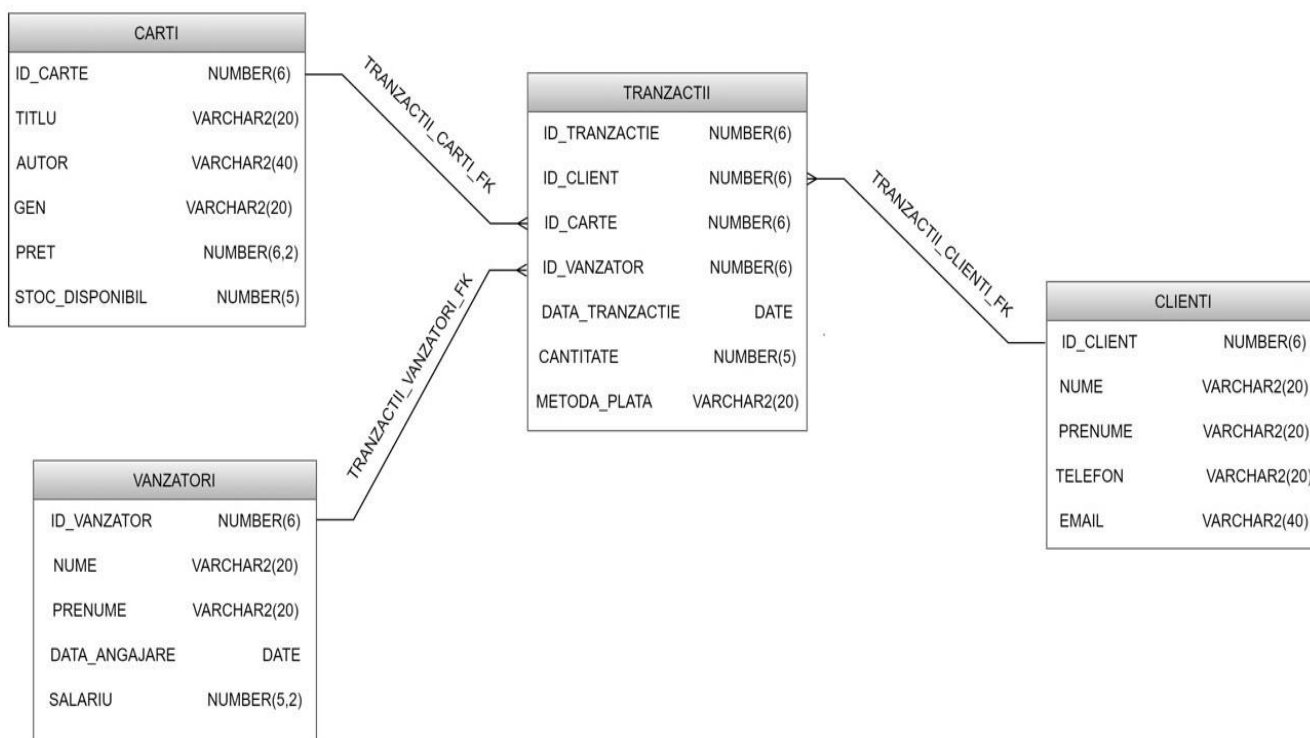
## Descrierea temei - Gestiunea unei librării

Proiectul reprezintă o gestiune a unei librării ce urmărește organizarea mai ușoară și sigură a informațiilor despre cărțile, clienții, vânzătorii și tranzacțiile respectivei librării, cât și legăturile dintre aceste cuvinte-cheie ce denumesc tabelele.

- Tabela „CARTI” reține informații cu privire la cărțile disponibile în librărie, și anume titlul, autorul, genul, prețul, stocul, cât și un identificator pentru fiecare carte.
- Tabela „VANZATORI” adăpostește informații cu privire la ID, nume, prenume, data angajării și salariu pentru angajații librăriei care se ocupă cu vânzările.
- Tabela „CLIENTI” stochează date legate de persoanele care au achiziționat cărți din librărie, precum ID-ul, numele, prenumele, numărul de telefon și email-ul acestora.
- Tabela „TRANZACTII” rezumă informațiile despre vânzările care au loc în librărie, prezentând atât un identificator propriu, cât și ID-uri corespunzătoare cărții cumpărate, clientului și vânzătorului, dar și metoda de plată și câte bucăți din respectiva carte au fost procurate.

În toate aceste tabele accentul este pus pe accesibilitatea datelor, proiectul oferind o viziune de bază asupra activităților librăriei, îmbunătățind eficiența operațională și experiența clienților, asigurându-se că detaliile esențiale sunt memorate.

## Schema conceptuală

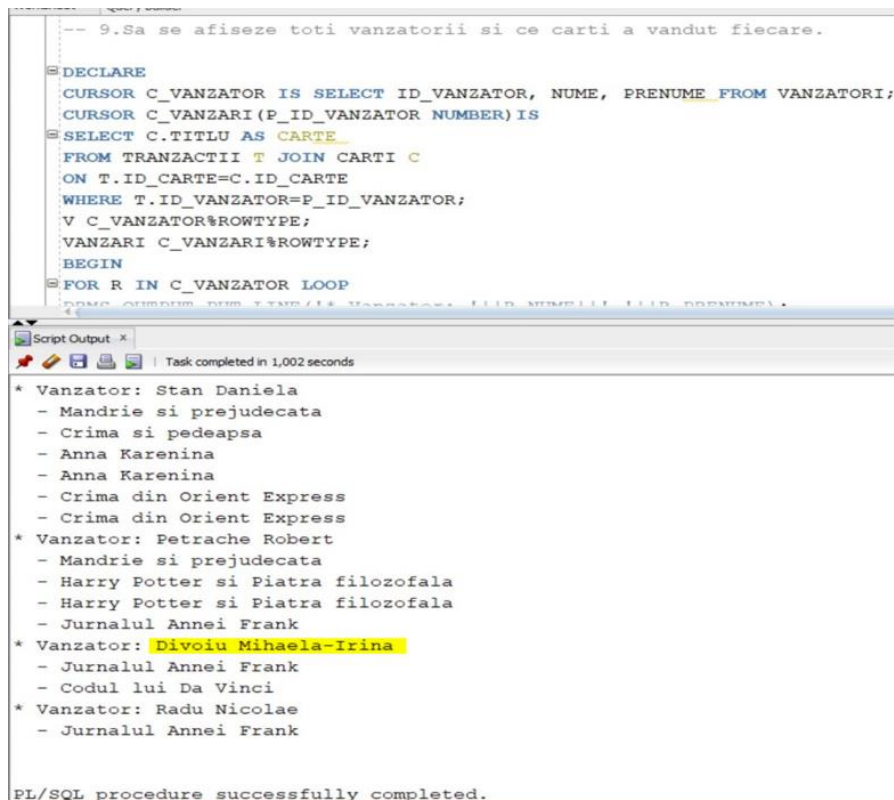


Schema din proiectul BD nu a fost modificată.

## Cerinte și rezolvări

-- 9.Sa se afiseze toti vanzatorii si ce carti a vandut fiecare. (Problema cu nume și prenume student)

```
DECLARE
CURSOR C_VANZATOR IS SELECT ID_VANZATOR, NUME, PRENUME FROM
VANZATORI;
CURSOR C_VANZARI(P_ID_VANZATOR NUMBER) IS
SELECT C.TITLU AS CARTE FROM TRANZACTII T JOIN CARTI C
ON T.ID_CARTE=C.ID_CARTE
WHERE T.ID_VANZATOR=P_ID_VANZATOR;
V C_VANZATOR%ROWTYPE;
VANZARI C_VANZARI%ROWTYPE;
BEGIN
FOR R IN C_VANZATOR LOOP
DBMS_OUTPUT.PUT_LINE('* Vanzator: '||R.NUME||' '||R.PRENUME);
OPEN C_VANZARI(R.ID_VANZATOR);
LOOP
FETCH C_VANZARI INTO VANZARI;
EXIT WHEN C_VANZARI%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(' - '||VANZARI.CARTE);
END LOOP;
CLOSE C_VANZARI;
END LOOP;
END;
/
```



```
-- 9.Sa se afiseze toti vanzatorii si ce carti a vandut fiecare.

DECLARE
CURSOR C_VANZATOR IS SELECT ID_VANZATOR, NUME, PRENUME FROM VANZATORI;
CURSOR C_VANZARI(P_ID_VANZATOR NUMBER) IS
SELECT C.TITLU AS CARTE
FROM TRANZACTII T JOIN CARTI C
ON T.ID_CARTE=C.ID_CARTE
WHERE T.ID_VANZATOR=P_ID_VANZATOR;
V C_VANZATOR%ROWTYPE;
VANZARI C_VANZARI%ROWTYPE;
BEGIN
FOR R IN C_VANZATOR LOOP
    DBMS_OUTPUT.PUT_LINE('* Vanzator: '||R.NUME||' '||R.PRENUME);
    OPEN C_VANZARI(R.ID_VANZATOR);
    LOOP
        FETCH C_VANZARI INTO VANZARI;
        EXIT WHEN C_VANZARI%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(' - '||VANZARI.CARTE);
    END LOOP;
    CLOSE C_VANZARI;
END LOOP;
END;
/
```

Script Output x

Task completed in 1,002 seconds

```
* Vanzator: Stan Daniela
- Mandrie si prejudecata
- Crima si pedeapsa
- Anna Karenina
- Anna Karenina
- Crima din Orient Express
- Crima din Orient Express
* Vanzator: Petrache Robert
- Mandrie si prejudecata
- Harry Potter si Piatra filozofala
- Harry Potter si Piatra filozofala
- Jurnalul Annei Frank
* Vanzator: Divoiu Mihaela-Irina
- Jurnalul Annei Frank
- Codul lui Da Vinci
* Vanzator: Radu Nicolae
- Jurnalul Annei Frank

PL/SQL procedure successfully completed.
```

```

WHERE T.ID_VANZATOR=P_ID_VANZATOR;
V C_VANZATOR%ROWTYPE;
VANZARI C_VANZARI%ROWTYPE;
BEGIN
FOR R IN C_VANZATOR LOOP
DBMS_OUTPUT.PUT_LINE('* Vanzator: '||R.NUME||' '||R.PRENUME);
OPEN C_VANZARI(R.ID_VANZATOR);
LOOP
FETCH C_VANZARI INTO VANZARI;
EXIT WHEN C_VANZARI%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(' - '||VANZARI.CARTE);
END LOOP;
CLOSE C_VANZARI;
END LOOP;
END;
/

```

Script Output x

Task completed in 1,002 seconds

```

* Vanzator: Stan Daniela
  - Mandrie si prejudecata
  - Crima si pedeapsa
  - Anna Karenina
  - Anna Karenina
  - Crima din Orient Express
  - Crima din Orient Express
* Vanzator: Petrache Robert
  - Mandrie si prejudecata
  - Harry Potter si Piatra filozofala
  - Harry Potter si Piatra filozofala
  - Jurnalul Annei Frank
* Vanzator: Divoiu Mihaela-Irina
  - Jurnalul Annei Frank
  - Codul lui Da Vinci
* Vanzator: Radu Nicolae
  - Jurnalul Annei Frank

```

-- 3.Sa se scrie un program in care sa se incarce numele si prenumele clientilor intr-un  
 -- tablou imbricat si sa se verifice daca exista vreun client cu numele "Marin".

```

DECLARE
TYPE TABELA_NUME_CLIENTI IS TABLE OF VARCHAR(50);
V_NUME TABELA_NUME_CLIENTI:=TABELA_NUME_CLIENTI();
V_EXISTA BOOLEAN:=FALSE;
BEGIN
SELECT NUME||' '||PRENUME
BULK COLLECT INTO V_NUME
FROM CLIENTI;
FOR i IN 1..V_NUME.COUNT LOOP
IF INSTR(V_NUME(i), 'Marin') > 0 THEN
V_EXISTA := TRUE;

```

```

EXIT;
END IF;
END LOOP;
IF V_EXISTA THEN
DBMS_OUTPUT.PUT_LINE('Exista cel putin un client cu numele "Marin" in lista
de nume.');
```

```

ELSE
DBMS_OUTPUT.PUT_LINE('Nu exista niciun client cu numele "Marin" in lista de
nume.');
```

```

END IF;
END;
/
```

```

-- 3.Sa se scrie un program in care sa se incarca numele si prenumele clientilor intr-un
-- tablou imbricat si sa se verifice daca exista vreun client cu numele "Marin".

DECLARE
TYPE TABELA_NUME_CLIENTI IS TABLE OF VARCHAR(50);
V_NUME TABELA_NUME_CLIENTI:=TABELA_NUME_CLIENTI();
V_EXISTA BOOLEAN:=FALSE;
BEGIN
SELECT NUME||' '||PRENUME
BULK COLLECT INTO V_NUME
FROM CLIENTI;
FOR i IN 1..V_NUME.COUNT LOOP
IF INSTR(V_NUME(i), 'Marin') > 0 THEN
V_EXISTA := TRUE;
EXIT;
END IF;
END LOOP;
IF V_EXISTA THEN
DBMS_OUTPUT.PUT_LINE('Exista cel putin un client cu numele "Marin" in lista de nume.');
```

```

ELSE
DBMS_OUTPUT.PUT_LINE('Nu exista niciun client cu numele "Marin" in lista de nume.');
```

```

END IF;
END;
/
```

Script Output x

Task completed in 0,034 seconds

```

PL/SQL procedure successfully completed.

Exista cel putin un client cu numele "Marin" in lista de nume.

PL/SQL procedure successfully completed.
```

-- 15.Sa se scrie o functie care are ca parametru de intrare un numarul de tranzactii si va returna o variabila de tip boolean pentru a verifica daca libraria depaseste acel numar (are succes).

```

CREATE OR REPLACE FUNCTION LIBRARIE(P_MIN_TRANZACTII IN
NUMBER)
RETURN BOOLEAN IS
```

```

V_NR_TRANZACTII NUMBER;
BEGIN
SELECT COUNT (*) INTO V_NR_TRANZACTII FROM TRANZACTII;
IF V_NR_TRANZACTII >= P_MIN_TRANZACTII THEN
RETURN TRUE;
ELSE RETURN FALSE;
END IF;
END;
/

DECLARE
V_REZULTAT BOOLEAN;
V_MIN_TRANZACTII NUMBER;
BEGIN
V_MIN_TRANZACTII:=&MINIM;
V_REZULTAT:=LIBRARIE(V_MIN_TRANZACTII);
IF V_REZULTAT THEN DBMS_OUTPUT.PUT_LINE('Libraria are succes.');
```

```

-- 15.Sa se scrie o functie care are ca parametru de intrare un numarul de tranzactii si va returna
-- o variabila de tip boolean pentru a verifica daca libraria depaseste acel numar (are succes).

CREATE OR REPLACE FUNCTION LIBRARIE(P_MIN_TRANZACTII IN NUMBER)
RETURN BOOLEAN IS
V_NR_TRANZACTII NUMBER;
BEGIN
SELECT COUNT (*) INTO V_NR_TRANZACTII FROM TRANZACTII;
IF V_NR_TRANZACTII >= P_MIN_TRANZACTII THEN
RETURN TRUE;
ELSE RETURN FALSE;
END IF;
END;
/

DECLARE
V_REZULTAT BOOLEAN;
V_MIN_TRANZACTII NUMBER;
BEGIN
V_MIN_TRANZACTII:=&MINIM;
V_REZULTAT:=LIBRARIE(V_MIN_TRANZACTII);
IF V_REZULTAT THEN DBMS_OUTPUT.PUT_LINE('Libraria are succes.');
```

```

ELSE DBMS_OUTPUT.PUT_LINE('Libraria nu are succes.');
```

```

END IF;
END;
/
```

Script Output x  
Task completed in 5,332 seconds

Libraria nu are succes.

PL/SQL procedure successfully completed.

Pentru exemplu s-a introdus de la tastatură 15 .

-- 18.Sa se creeze un pachet compus dintr-o procedura care calculeaza numarul de bucati  
-- vandute intr-o carte si o functie care obtine valoarea totala castigata din vanzarea cartii.

```
CREATE OR REPLACE PACKAGE PROFIT_CARTE AS
PROCEDURE BUCATI_VANDUTE(P_ID IN NUMBER, V_BUCATI OUT
NUMBER);
FUNCTION VALOARE_CASTIG(P_ID IN NUMBER) RETURN NUMBER;
END PROFIT_CARTE;
/
CREATE OR REPLACE PACKAGE BODY PROFIT_CARTE AS
PROCEDURE BUCATI_VANDUTE(P_ID IN NUMBER, V_BUCATI OUT
NUMBER) IS
BEGIN SELECT SUM(CANTITATE) INTO V_BUCATI FROM TRANZACTII
WHERE ID_CARTE = P_ID;
END BUCATI_VANDUTE;
FUNCTION VALOARE_CASTIG(P_ID IN NUMBER) RETURN NUMBER IS
V_NR_BUC NUMBER := 0;
V_PRET NUMBER;
V_VALOARE NUMBER := 0;
BEGIN
BUCATI_VANDUTE(P_ID, V_NR_BUC);
SELECT PRET INTO V_PRET FROM CARTI WHERE ID_CARTE = P_ID;
V_VALOARE := V_NR_BUC * V_PRET;
RETURN V_VALOARE;
END VALOARE_CASTIG;
END PROFIT_CARTE;
/

DECLARE
V_BUC NUMBER;
V_VAL NUMBER;
V_ID NUMBER;
BEGIN
V_ID := &ID;
PROFIT_CARTE.BUCATI_VANDUTE(V_ID,V_BUC);
DBMS_OUTPUT.PUT_LINE('Numarul total de bucati vandute: ' || V_BUC);
V_VAL := PROFIT_CARTE.VALOARE_CASTIG(V_ID);
DBMS_OUTPUT.PUT_LINE('Valoarea totala a vanzarilor cartii: ' || V_VAL);
END;
/

SELECT * FROM CARTI ORDER BY ID_CARTE;
```

```
-- 18.Sa se creeze un pachet compus dintr-o procedura care calculeaza numarul de bucati vandute
-- intr-o carte si o functie care obtine valoarea totala castigata din vanzarea cartii.

CREATE OR REPLACE PACKAGE PROFIT_CARTE AS
  PROCEDURE BUCATI_VANDUTE(P_ID IN NUMBER, V_BUCATI OUT NUMBER);
  FUNCTION VALOARE_CASTIG(P_ID IN NUMBER) RETURN NUMBER;
END PROFIT_CARTE;
/

CREATE OR REPLACE PACKAGE BODY PROFIT_CARTE AS
  PROCEDURE BUCATI_VANDUTE(P_ID IN NUMBER, V_BUCATI OUT NUMBER) IS
  BEGIN SELECT SUM(CANTITATE) INTO V_BUCATI FROM TRANZACTII WHERE ID_CARTE = P_ID;
  END BUCATI_VANDUTE;
  FUNCTION VALOARE_CASTIG(P_ID IN NUMBER) RETURN NUMBER IS
  V_NR_BUC NUMBER := 0;
  V_PRET NUMBER;
  V_VALOARE NUMBER := 0;
  BEGIN
    BUCATI_VANDUTE(P_ID, V_NR_BUC);
    SELECT PRET INTO V_PRET FROM CARTI WHERE ID_CARTE = P_ID;
    V_VALOARE := V_NR_BUC * V_PRET;
    RETURN V_VALOARE;
  END VALOARE_CASTIG;
END PROFIT_CARTE;
/
```

Script Output x

Task completed in 0,304 seconds

Package PROFIT\_CARTE compiled

Package Body PROFIT\_CARTE compiled

```
V_PRET NUMBER;
V_VALOARE NUMBER := 0;
BEGIN
  BUCATI_VANDUTE(P_ID, V_NR_BUC);
  SELECT PRET INTO V_PRET FROM CARTI WHERE ID_CARTE = P_ID;
  V_VALOARE := V_NR_BUC * V_PRET;
  RETURN V_VALOARE;
END VALOARE_CASTIG;
END PROFIT_CARTE;
/

DECLARE
  V_BUC NUMBER;
  V_VAL NUMBER;
  V_ID NUMBER;
  BEGIN
    V_ID := &ID;
    PROFIT_CARTE.BUCATI_VANDUTE(V_ID,V_BUC);
    DBMS_OUTPUT.PUT_LINE('Numarul total de bucati vandute: ' || V_BUC);
    V_VAL := PROFIT_CARTE.VALOARE_CASTIG(V_ID);
    DBMS_OUTPUT.PUT_LINE('Valoarea totala a vanzarilor cartii: ' || V_VAL);
  END;
/

SELECT * FROM CARTI ORDER BY ID_CARTE;
```

Script Output x

Task completed in 24,783 seconds

END;

Numarul total de bucati vandute: 1

Valoarea totala a vanzarilor cartii: 49.5

PL/SQL procedure successfully completed.

Pentru exemplu s-a introdus de la tastatură 24 .

## SCRIPT-UL COMPLET

```
SET SERVEROUTPUT ON
```

```
SELECT * FROM CARTI ORDER BY ID_CARTE;  
SELECT * FROM CLIENTI ORDER BY ID_CLIENT;  
SELECT * FROM VANZATORI ORDER BY ID_VANZATOR;  
SELECT * FROM TRANZACTII ORDER BY ID_TRANZACTIE;
```

```
-- 1.Pentru vanzatorii cu salariu mai mic de 2800 sa se mareasca acesta  
-- cu un anumit procent introdus de la tastatura.
```

```
DECLARE  
V_PROCENT NUMBER;  
BEGIN  
V_PROCENT:=&PROCENT;  
EXECUTE IMMEDIATE 'UPDATE VANZATORI  
SET SALARIU = SALARIU *' || V_PROCENT ||  
'WHERE SALARIU <2800';  
DBMS_OUTPUT.PUT_LINE('Salariile mai mici de 2800 au fost actualizate.');
```

```
END;  
/
```

```
ROLLBACK;
```

```
SELECT * FROM VANZATORI ORDER BY ID_VANZATOR;
```

```
-- 2.Sa se incarce toate datele din tabela CLIENTI intr-un tablou indexat.
```

```
DECLARE  
TYPE TABELA_INDEXATA IS TABLE OF CLIENTI%ROWTYPE INDEX BY  
PLS_INTEGER;  
INDEX_CLIENTI TABELA_INDEXATA;  
V_NR NUMBER:=0;  
BEGIN  
FOR R IN (SELECT * FROM CLIENTI) LOOP  
V_NR:=V_NR+1;  
INDEX_CLIENTI(V_NR):=R;  
END LOOP;  
DBMS_OUTPUT.PUT_LINE('Lista clienti: ');  
FOR I IN 1..V_NR LOOP  
DBMS_OUTPUT.PUT_LINE('ID client: ' || INDEX_CLIENTI(I).ID_CLIENT || ',  
nume: ' || INDEX_CLIENTI(I).NUME || ', prenume: ' ||  
INDEX_CLIENTI(I).PRENUME
```



```

|| ', telefon: ' || INDEX_CLIENTI(I).TELEFON || ', email: ' ||
INDEX_CLIENTI(I).EMAIL);
END LOOP;
END;
/

```

-- 3.Sa se scrie un program in care sa se incarca numele si prenumele clientilor intr-un  
-- tablou imbricat si sa se verifice daca exista vreun client cu numele "Marin".

```

DECLARE
TYPE TABELA_NUME_CLIENTI IS TABLE OF VARCHAR(50);
V_NUME TABELA_NUME_CLIENTI:=TABELA_NUME_CLIENTI();
V_EXISTA BOOLEAN:=FALSE;
BEGIN
SELECT NUME||' '||PRENUME
BULK COLLECT INTO V_NUME
FROM CLIENTI;
FOR i IN 1..V_NUME.COUNT LOOP
IF INSTR(V_NUME(i), 'Marin') > 0 THEN
V_EXISTA := TRUE;
EXIT;
END IF;
END LOOP;
IF V_EXISTA THEN
DBMS_OUTPUT.PUT_LINE('Exista cel putin un client cu numele "Marin" in lista
de nume. ');
ELSE
DBMS_OUTPUT.PUT_LINE('Nu exista niciun client cu numele "Marin" in lista de
nume. ');
END IF;
END;
/

```

-- 4.Sa se creeze un vector de lungime variabila VARRAYS in care sa se stocheze note  
(de la 1 la 10)  
-- date unei carti, apoi sa se verifice media acestora si cat de apreciata este cartea.

```

DECLARE
TYPE T_NOTE IS VARRAY(10) OF NUMBER;
NOTE T_NOTE:=T_NOTE(8,6,9,4,7);
NR_NOTE NUMBER := NOTE.COUNT;
TOTAL NUMBER := 0;
MEDIE INTEGER;
BEGIN

```

```

FOR I IN 1..NR_NOTE LOOP
TOTAL:=TOTAL+NOTE(I);
END LOOP;
MEDIE := TOTAL/NR_NOTE;
CASE
WHEN MEDIE>=8 THEN DBMS_OUTPUT.PUT_LINE('Carte recomandata');
WHEN MEDIE BETWEEN 5 AND 8 THEN DBMS_OUTPUT.PUT_LINE('Carte
considerata buna');
ELSE DBMS_OUTPUT.PUT_LINE('Carte cu nota medie scazuta');
END CASE;
EXCEPTION
WHEN ZERO_DIVIDE THEN
DBMS_OUTPUT.PUT_LINE('Eroare: impartire la zero în calculul mediei, nu exista
note atribuite cartii.');
```

```

-- 5.Sa se trateze exceptia in cazul in care se doreste stergerea primei carti
-- din tabela daca aceasta nu are autorul specificat.
```

```

DECLARE
EXCEPTIE EXCEPTION;
PRAGMA EXCEPTION_INIT(EXCEPTIE, -20001);
V_AUTOR CARTI.AUTOR%TYPE;
BEGIN
SELECT AUTOR
INTO V_AUTOR
FROM CARTI
WHERE ROWNUM=1;
IF V_AUTOR IS NULL THEN
DELETE FROM CARTI
WHERE ROWNUM = 1;
ELSE
RAISE EXCEPTIE;
END IF;
EXCEPTION
WHEN EXCEPTIE THEN
DBMS_OUTPUT.PUT_LINE('Cartea nu poate fi stearsa, deoarece autorul este
specificat.');
```

```

SELECT * FROM CARTI ORDER BY ID_CARTE;
```

-- 6.Sa se trateze exceptie daca se doreste stergerea datei de tranzactie la tranzactiile din 2024.

```
DECLARE
EXCEPTIE EXCEPTION;
PRAGMA EXCEPTION_INIT(EXCEPTIE, -20003);
BEGIN
IF EXTRACT(YEAR FROM SYSDATE) = 2024 THEN
RAISE EXCEPTIE;
END IF;
EXCEPTION
WHEN EXCEPTIE THEN
DBMS_OUTPUT.PUT_LINE('Nu se poate efectua stergerea pentru tranzactiile din
2024.');
```

-- 7.Sa se stearga clientii care nu au efectuat nicio comanda si sa se numere cate randuri au fost sterse.

```
DECLARE
NR_RANDOMURI_STERSE NUMBER:=0;
BEGIN
DELETE FROM CLIENTI C
WHERE NOT EXISTS(
SELECT 1 FROM TRANZACTII T
WHERE C.ID_CLIENT = T.ID_CLIENT);
NR_RANDOMURI_STERSE := SQL%ROWCOUNT;
IF NR_RANDOMURI_STERSE =0 THEN
DBMS_OUTPUT.PUT_LINE('Nu a fost sters niciun rand.');
```

ROLLBACK;

SELECT \* FROM TRANZACTII ORDER BY ID\_TRANZACTIE;

-- 8.Folosind un cursor sa se afiseze numele si prenumele celor mai recenti 5 clienti.

```
DECLARE
CURSOR C IS SELECT NUME, PRENUME, DATA_TRANZACTIE
```

```

FROM CLIENTI CL JOIN TRANZACTII TR
ON CL.ID_CLIENT=TR.ID_CLIENT
ORDER BY DATA_TRANZACTIE DESC;
BEGIN
FOR R IN C LOOP
EXIT WHEN C%ROWCOUNT>5;
DBMS_OUTPUT.PUT_LINE(R.NUME||' '||R.PRENUME||' a facut o tranzactie la
data de '||R.DATA_TRANZACTIE);
END LOOP;
END;
/

```

```

SELECT * FROM CLIENTI ORDER BY ID_CLIENT;
SELECT * FROM TRANZACTII ORDER BY ID_TRANZACTIE;

```

-- 9.Sa se afiseze toti vanzatorii si ce carti a vandut fiecare.

```

DECLARE
CURSOR C_VANZATOR IS SELECT ID_VANZATOR, NUME, PRENUME FROM
VANZATORI;
CURSOR C_VANZARI(P_ID_VANZATOR NUMBER)IS
SELECT C.TITLU AS CARTE
FROM TRANZACTII T JOIN CARTI C
ON T.ID_CARTE=C.ID_CARTE
WHERE T.ID_VANZATOR=P_ID_VANZATOR;
V C_VANZATOR%ROWTYPE;
VANZARI C_VANZARI%ROWTYPE;
BEGIN
FOR R IN C_VANZATOR LOOP
DBMS_OUTPUT.PUT_LINE('* Vanzator: '||R.NUME||' '||R.PRENUME);
OPEN C_VANZARI(R.ID_VANZATOR);
LOOP
FETCH C_VANZARI INTO VANZARI;
EXIT WHEN C_VANZARI%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(' - '||VANZARI.CARTE);
END LOOP;
CLOSE C_VANZARI;
END LOOP;
END;
/

```

-- 10.Sa se creeze o procedura care sa primeasca un id de carte ca parametru si sa afiseze  
-- titlul si autorul, apoi sa se trateze o exceptie in cazul in care cartea nu a fost gasita.

```
CREATE OR REPLACE PROCEDURE AFISARE_CARTE(P_ID_CARTE IN
NUMBER) IS
V_TITLU VARCHAR2(40);
V_AUTOR VARCHAR2(40);
BEGIN
SELECT TITLU, AUTOR
INTO V_TITLU, V_AUTOR
FROM CARTI
WHERE ID_CARTE=P_ID_CARTE;
DBMS_OUTPUT.PUT_LINE('Titlu: ' || V_TITLU);
DBMS_OUTPUT.PUT_LINE('Autor: ' || V_AUTOR);
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Nu exista carte cu acest ID.');
```

END;

/

EXECUTE AFISARE\_CARTE(24);

-- 11.Sa se returneze numarul de carti vandute al caror pret se aflate in intervalul dorit.

```
CREATE OR REPLACE FUNCTION
NUMAR_CARTI_INTERVAL(P_INC_INTERVAL NUMBER, P_SF_INTERVAL
NUMBER)
RETURN NUMBER IS
V_TOTAL NUMBER :=0;
V_NR NUMBER :=0;
V_ID CARTI.ID_CARTE%TYPE;
BEGIN
FOR V_ID IN 23..29 LOOP
SELECT COUNT (*)
INTO V_NR
FROM CARTI
WHERE ID_CARTE=V_ID AND PRET BETWEEN P_INC_INTERVAL AND
P_SF_INTERVAL;
V_TOTAL:=V_TOTAL+V_NR;
END LOOP;
RETURN V_TOTAL;
END;
```

/

```

DECLARE
V_CARTI_VANDUTE NUMBER;
BEGIN
V_CARTI_VANDUTE := NUMAR_CARTI_INTERVAL(20.00, 40.00);
IF V_CARTI_VANDUTE = 0 THEN
DBMS_OUTPUT.PUT_LINE('Nu au fost inregistrate vanzari de carti cu pretul
cuprins in acest interval. ');
ELSE
DBMS_OUTPUT.PUT_LINE('Au fost vandute ' || V_CARTI_VANDUTE || ' carti ce
au pretul cuprins in intervalul dorit. ');
END IF;
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLERRM);
END;
/

```

-- 12.Sa se creeze o procedura care calculeaza si afiseaza vechimea fiecarui vanzator in luni.

```

CREATE OR REPLACE PROCEDURE VECHIME IS
BEGIN
FOR R IN(
SELECT ID_VANZATOR, NUME || ' ' || PRENUME AS VANZATOR,
DATA_ANGAJARE
FROM VANZATORI) LOOP
DECLARE
V_VECHIME NUMBER;
BEGIN
V_VECHIME := FLOOR(MONTHS_BETWEEN(SYSDATE,
R.DATA_ANGAJARE));
DBMS_OUTPUT.PUT_LINE('Angajatul ' || R.VANZATOR || ' are o vechime de ' ||
V_VECHIME || ' luni. ');
END;
END LOOP;
END VECHIME;
/

```

```

BEGIN
    VECHIME;
END;
/

```

-- 13.Sa se scrie o procedura care calculeaza stocul total de carti.

```
CREATE OR REPLACE PROCEDURE STOC_TOTAL IS
V_STOC NUMBER;
BEGIN
SELECT SUM(STOC_DISPONIBIL)
INTO V_STOC
FROM CARTI;
DBMS_OUTPUT.PUT_LINE('Stocul total de carti este: ' || V_STOC);
END STOC_TOTAL;
/
```

```
BEGIN
STOC_TOTAL;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Nu s-au gasit date in tabela CARTI.');
```

END;

/

-- 14.Sa se creeze o functie care returneaza tranzactia cu cea mai mare cantitate de carti vanduta.

```
CREATE OR REPLACE FUNCTION TRANZACTIE_MAX
RETURN TRANZACTII%ROWTYPE IS
V_TRANZACTIE TRANZACTII%ROWTYPE;
BEGIN
SELECT *
INTO V_TRANZACTIE
FROM TRANZACTII
WHERE CANTITATE = (SELECT MAX(CANTITATE) FROM TRANZACTII);
RETURN V_TRANZACTIE;
END TRANZACTIE_MAX;
/
```

```
DECLARE
V_TRANZACTIE TRANZACTII%ROWTYPE;
BEGIN
V_TRANZACTIE := TRANZACTIE_MAX;
IF V_TRANZACTIE.ID_TRANZACTIE IS NOT NULL THEN
DBMS_OUTPUT.PUT_LINE('Tranzactia cu ID-ul
'||V_TRANZACTIE.ID_TRANZACTIE||' a avut cele mai multe carti vandute:
'||V_TRANZACTIE.CANTITATE);
END IF;
EXCEPTION
```

```

WHEN TOO_MANY_ROWS THEN
DBMS_OUTPUT.PUT_LINE('Eroare: Exista mai multe tranzactii cu cantitate
maxima de carti vandute.');
```

WHEN NO\_DATA\_FOUND THEN

```

DBMS_OUTPUT.PUT_LINE('Eroare: Nu s-a gasit nicio tranzactie.');
```

END;

/

-- 15.Sa se scrie o functie care are ca parametru de intrare un numarul de tranzactii si va returna

-- o variabila de tip boolean pentru a verifica daca libraria depaseste acel numar (are succes).

```

CREATE OR REPLACE FUNCTION LIBRARIE(P_MIN_TRANZACTII IN
NUMBER)
RETURN BOOLEAN IS
V_NR_TRANZACTII NUMBER;
BEGIN
SELECT COUNT (*) INTO V_NR_TRANZACTII FROM TRANZACTII;
IF V_NR_TRANZACTII >= P_MIN_TRANZACTII THEN
RETURN TRUE;
ELSE RETURN FALSE;
END IF;
END;
/
```

```

DECLARE
V_REZULTAT BOOLEAN;
V_MIN_TRANZACTII NUMBER;
BEGIN
V_MIN_TRANZACTII:=&MINIM;
V_REZULTAT:=LIBRARIE(V_MIN_TRANZACTII);
IF V_REZULTAT THEN DBMS_OUTPUT.PUT_LINE('Libraria are succes.');
```

ELSE DBMS\_OUTPUT.PUT\_LINE('Libraria nu are succes.');

```

END IF;
END;
/
```

-- 16.Sa se creeze un trigger pentru a nu depasi un pret maxim standard pentru orice carte.

```

CREATE OR REPLACE TRIGGER RESTRICIE_PRET
BEFORE INSERT OR UPDATE ON CARTI
FOR EACH ROW
```



```

DECLARE
V_PRET_MAX NUMBER;
BEGIN
V_PRET_MAX := &MAXIM;
IF :NEW.PRET > V_PRET_MAX THEN
RAISE_APPLICATION_ERROR (-20202, 'Nu se poate depasi pretul maxim
pentru functia data');
END IF;
END;
/

```

-- 17.Sa se creeze un trigger care sa verifice unicitatea codului fiecarui vanzator folosindu-se secventa.

```

CREATE SEQUENCE ANGAJATI
START WITH 1
INCREMENT BY 1
MAXVALUE 100
NOCYCLE;

```

```

CREATE OR REPLACE TRIGGER COD_UNIC
BEFORE INSERT ON VANZATORI
FOR EACH ROW
BEGIN
SELECT ANGAJATI.NEXTVAL INTO :NEW.ID_VANZATOR FROM DUAL;
END;
/

```

-- 18.Sa se creeze un pachet compus dintr-o procedura care calculeaza numarul de bucati vandute  
-- intr-o carte si o functie care obtine valoarea totala castigata din vanzarea cartii.

```

CREATE OR REPLACE PACKAGE PROFIT_CARTE AS
PROCEDURE BUCATI_VANDUTE(P_ID IN NUMBER, V_BUCATI OUT
NUMBER);
FUNCTION VALOARE_CASTIG(P_ID IN NUMBER) RETURN NUMBER;
END PROFIT_CARTE;
/

```

```

CREATE OR REPLACE PACKAGE BODY PROFIT_CARTE AS
PROCEDURE BUCATI_VANDUTE(P_ID IN NUMBER, V_BUCATI OUT
NUMBER) IS
BEGIN SELECT SUM(CANTITATE) INTO V_BUCATI FROM TRANZACTII
WHERE ID_CARTE = P_ID;

```

```

END BUCATI_VANDUTE;
FUNCTION VALOARE_CASTIG(P_ID IN NUMBER) RETURN NUMBER IS
V_NR_BUC NUMBER := 0;
V_PRET NUMBER;
V_VALOARE NUMBER := 0;
BEGIN
BUCATI_VANDUTE(P_ID, V_NR_BUC);
SELECT PRET INTO V_PRET FROM CARTI WHERE ID_CARTE = P_ID;
V_VALOARE := V_NR_BUC * V_PRET;
RETURN V_VALOARE;
END VALOARE_CASTIG;
END PROFIT_CARTE;
/

```

```

DECLARE
V_BUC NUMBER;
V_VAL NUMBER;
V_ID NUMBER;
BEGIN
V_ID := &ID;
PROFIT_CARTE.BUCATI_VANDUTE(V_ID,V_BUC);
DBMS_OUTPUT.PUT_LINE('Numarul total de bucati vandute: ' || V_BUC);
V_VAL := PROFIT_CARTE.VALOARE_CASTIG(V_ID);
DBMS_OUTPUT.PUT_LINE('Valoarea totala a vanzarilor cartii: ' || V_VAL);
END;
/

```

```

SELECT * FROM CARTI ORDER BY ID_CARTE;

```