**PROIECT**

**SISTEM DE GESTIUNE**

**A BAZELOR DE DATE**

**Tema proiectului:**

**Magazin online**

**Realizat de: Neculai Marius-Adrian**

**Cuprins**

[**Introducere** 3](#_Toc166673049)

[**Structuri alternative și repetitive** 4](#_Toc166673050)

[**Tratarea excepțiilor** 8](#_Toc166673051)

[**Gestionarea cursorilor: impliciți si expliciți** 11](#_Toc166673052)

[**Funcţii, proceduri, includerea acestora în pachete** 15](#_Toc166673053)

[**Declanşatori** 21](#_Toc166673054)

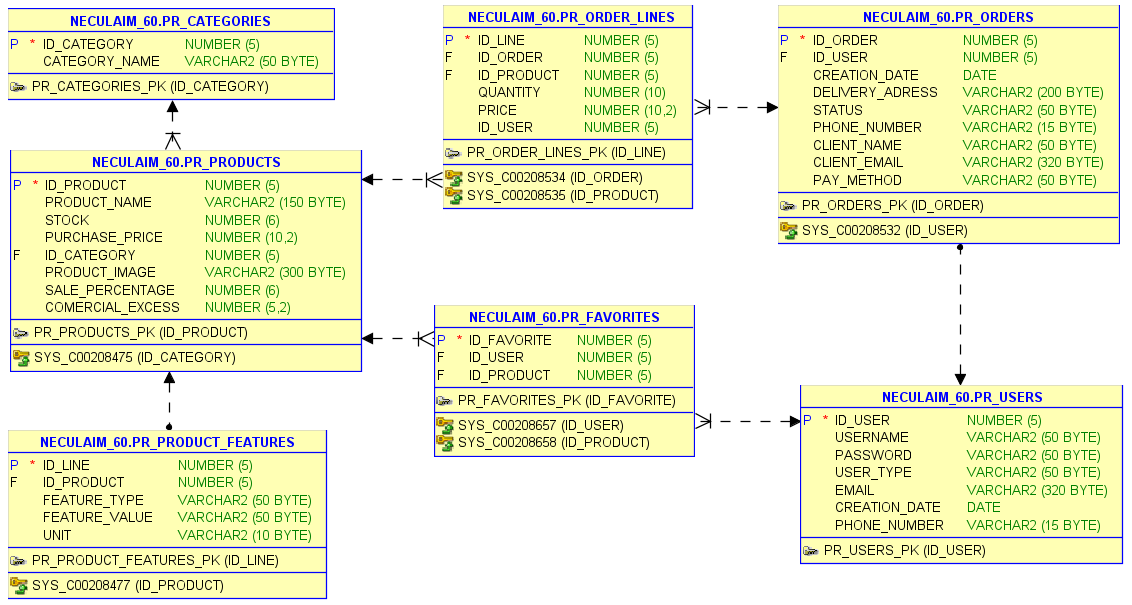
# **Introducere**

Tema economică aleasă pentru acest proiect este “Magazin online”. Am creeat o baza de date utilă pentru o platformă de tip magain online. Scopul principal al acestui proiect este de a construi o platformă eficientă și ușor de utilizat pentru administrarea unui magazinului, care să răspundă nevoilor atât ale administratorilor, cât și ale clienților.

Structura bazei de date este următoarea:

* **PRODUCTS**: tabela pentru stocarea datelor despre produse.
* **USERS**: tabela în care se stocheaza datele utilizatorilor care își creeaza cont pe platformă
* **CATEGORIES**: tabelă pentru gestionarea categoriilor de produse existente
* **PRODUCTS\_FEATURES**: tabelă pentru a gestiona specificațiile fiecărui produs
* **FAVORITES**: tabelă pentru memorarea listelor de produse favorite pentru fiecare utilizator, de asemenea joaca rolul unei tabele intermediare dintre USERS și PRODUCTS
* **ORDERS**: tabela pentru a reține informațiile fiecărei comenzi
* **ORDER\_LINES**: tabela intermediară între ORDERS și PRODUCTS

**SCHEMA CONCEPTUALĂ A BAZEI DE DATE**



# **Structuri alternative și repetitive**

1. Să se afișeze username-ul si id-ul utilizatorii care au id-ul cuprins intre 200 și 220.

declare

v\_id USERS.ID\_USER%TYPE;

v\_username USERS.USERNAME%TYPE;

begin

FOR v\_id IN 200..220 LOOP

BEGIN

SELECT username INTO v\_username

FROM USERS

WHERE id\_user = v\_id;

DBMS\_OUTPUT.PUT\_LINE('ID: ' || v\_id || ', Username: ' || v\_username);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

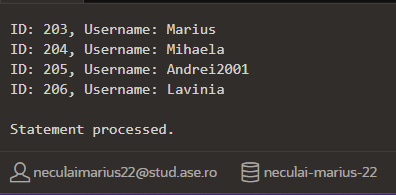
-- Daca nu se gaseste un user cu id = v\_id atunci trecem mai departe

NULL;

END;

END LOOP;

end;



1. Să se afișeze nume produselor care au id-ul cuprins intre 10 și 20. Daca nu exista un produs cu unul dintre aceste id-uri sa se afișeze un mesaj .

DECLARE

v\_id PRODUCTS.ID\_PRODUCT%TYPE := 10; -- Inițializează v\_id la 10

v\_product\_name PRODUCTS.PRODUCT\_NAME%TYPE;

BEGIN

WHILE v\_id <= 20 LOOP

BEGIN

SELECT product\_name INTO v\_product\_name FROM products

WHERE id\_product = v\_id;

DBMS\_OUTPUT.PUT\_LINE('ID: ' || v\_id || ', Product Name: ' || v\_product\_name);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista produs cu id-ul: ' || v\_id);

END;

v\_id := v\_id + 1;

end loop;

END;

/



1. Să se selecteze produsele care au id-ul cuprins între 10 și 15. Să se închidă structura repetititvă cand se întalneste un produs cu prețul de achiziție mai mic decat cel mediu.

DECLARE

v\_id PRODUCTS.ID\_PRODUCT%TYPE := 10; -- Inițializează v\_id la 10

v\_purchase\_price PRODUCTS.PRODUCT\_NAME%TYPE;

v\_average\_price PRODUCTS.PURCHASE\_PRICE%TYPE;

BEGIN

select avg(PURCHASE\_PRICE) into v\_average\_price from products;

DBMS\_OUTPUT.PUT\_LINE('Pretul mediu de achizitie este ' || v\_average\_price || '€ ');

loop

select purchase\_price into v\_purchase\_price from products

where id\_product=v\_id;

DBMS\_OUTPUT.PUT\_LINE('ID: ' || v\_id ||' Pret de achizitie: '||v\_purchase\_price|| '€ ');

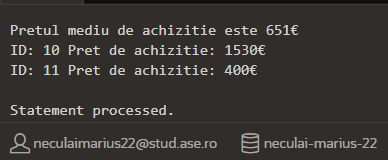
v\_id:=v\_id+1;

exit when v\_purchase\_price<v\_average\_price or v\_id>20;

end loop;

END;

/



1. Să se afișeze username-ul utilizatorului cu id 204. Daca username-ul este ‘Mihaela’ atunci să se modifice tipul de utilizator (user\_type) în ‘admin’.

DECLARE

v\_username USERS.USERNAME%TYPE;

v\_user\_type USERS.USER\_TYPE%TYPE;

BEGIN

select username, user\_type into v\_username,v\_user\_type from users where id\_user=204;

dbms\_output.put\_line('Username: '|| v\_username || ' | User type:'||v\_user\_type);

if v\_username='Mihaela' then

Update USERS

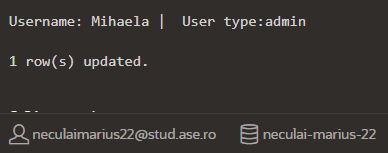
Set user\_type='admin'

where id\_user=204;

end if;

END;

/



1. Să se afișeze produsele cu id cuprins între 7 și 15. Să se calculeze costul de transport astfel:
   1. Dacă produsul face parte din categoria ‘Laptops’ costul este 20
   2. Dacă produsul face parte din categoria ‘TV’ costul este 100
   3. Dacă produsul face parte din alta categorie atunci costul este 35

DECLARE

v\_id PRODUCTS.ID\_PRODUCT%TYPE := 7;

v\_product\_name PRODUCTS.PRODUCT\_NAME%TYPE;

v\_category\_name CATEGORIES.CATEGORY\_NAME%TYPE;

v\_shipping\_cost NUMBER;

BEGIN

WHILE v\_id <= 15 LOOP

BEGIN

SELECT product\_name p, category\_name c INTO v\_product\_name, v\_category\_name FROM products p,categories c

WHERE c.id\_category=p.id\_category AND p.id\_product = v\_id;

CASE v\_category\_name

WHEN 'Laptops' THEN v\_shipping\_cost := 20;

WHEN 'TV' THEN v\_shipping\_cost := 100;

ELSE v\_shipping\_cost := 35;

END CASE;

DBMS\_OUTPUT.PUT\_LINE('ID: ' || v\_id || ', Product Name: ' || v\_product\_name || ', Category: ' || v\_category\_name || ', Shipping Cost: ' || v\_shipping\_cost);

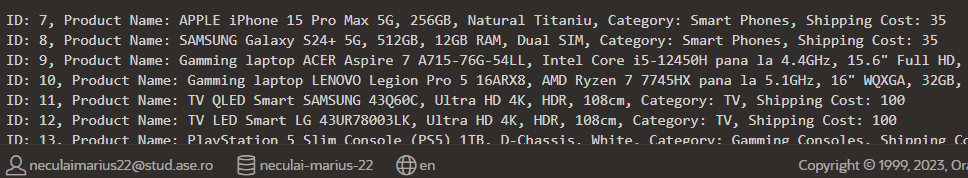
END;

v\_id := v\_id + 1;

END LOOP;

END;

/



# **Tratarea excepțiilor**

1. Să se afișeze userul cu id-ul 28. Să se trateze eroarea apărută în cazul în care nu există nici un user cu acest id.

DECLARE

v\_username USERS.username%TYPE;

BEGIN

select username into v\_username from users where id\_user=103;

dbms\_output.put\_line('Utilizator : '||v\_username);

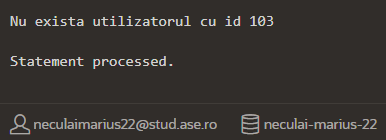
EXCEPTION

when no\_data\_found then

dbms\_output.put\_line('Nu exista utilizatorul cu id 103 ');

END;

/



1. Să se afișeze id-ul utilizatorului cu username-ul ‘Dani’. Să se trateze eroarea în cazul în care sunt mai mulți utilizatori cu acelasi usename.

DECLARE

v\_id\_user USERS.id\_user%TYPE;

BEGIN

select id\_user into v\_id\_user from users where username='Dani';

dbms\_output.put\_line('Id utilizator : '||v\_id\_user);

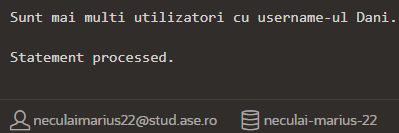
EXCEPTION

when too\_many\_rows then

dbms\_output.put\_line('Sunt mai multi utilizatori cu username-ul Dani.');

END;

/



1. Să se mărească stocul cu 10 unități pentru produsul cu id=999. Să se declanșeze o excepție prin care sa fie avertizat utilizatorul că nu s- a produs nici o modificare.

DECLARE

invalid\_exception exception;

BEGIN

update products

set stock=stock+10

where id\_product=999;

if sql%notfound then

raise invalid\_exception;

end if;

EXCEPTION

when invalid\_exception then

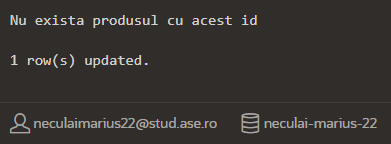
dbms\_output.put\_line('Nu exista produsul cu acest id');

when others then

dbms\_output.put\_line('A aparut o eroare, nu se poate actualiza stocul produsului');

END;

/



1. Să se modifice numarul de telefon al utilizatorului cu id=10. Daca acesta nu există, să se trateze excepția. Să se atribuie excepției un cod și un mesaj de eroare care sa fie afisat utilizatorului.

declare

invalid\_user exception;

pragma exception\_init(invalid\_user,-20999);

begin

update users

set phone\_number='0799443221'

where id\_user=10;

if sql%notfound then

raise\_application\_error(-20999,'Id user invalid!');

end if;

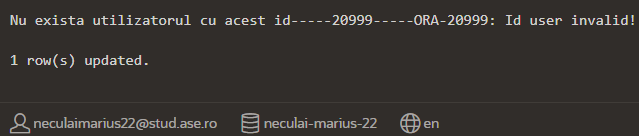
exception

when invalid\_user then

dbms\_output.put\_line('Nu exista utilizatorul cu acest id----'||SQLCODE||'-----'||SQLERRM);

end;

/



1. Să se afișeze suma totală a comenzii cu id-ul=999. Daca aceasta nu există atunci, să se invoce în mod explicit excepția predefinita ‘no\_data\_found’.

DECLARE

v\_suma ORDER\_LINES.PRICE%TYPE;

BEGIN

select sum(price\*quantity) into v\_suma from order\_lines

group by id\_order

having id\_order=999;

if sql%notfound then

raise no\_data\_found;

end if;

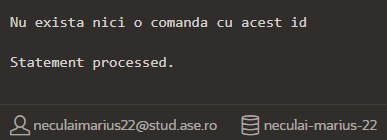
dbms\_output.put\_line('Suma totala a comenzii 999 este = '||v\_suma);

EXCEPTION

when no\_data\_found then

dbms\_output.put\_line('Nu exista nici o comanda cu acest id');

END;



# **Gestionarea cursorilor: impliciți si expliciți**

1. Să se mărească stocul produselor care fac parte din categria cu id=1. Să se afișeze numarul de randuri modificate.

BEGIN

UPDATE products

SET stock=stock+10

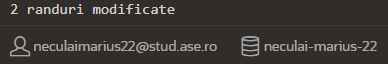
WHERE id\_category=1;

dbms\_output.put\_line(SQL%ROWCOUNT||' randuri modificate');

ROLLBACK;

END;

/



1. Să se modifice denumirea produsului cu id=89, daca nu există atunci să se afișeze un mesaj corespunzător.

BEGIN

UPDATE products

SET product\_name='Casti bluethoot'

WHERE id\_category=89;

IF sql%notfound then

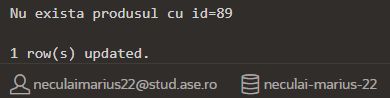
dbms\_output.put\_line('Nu exista produsul cu id=89');

END IF;

ROLLBACK;

END;

/



1. Să se afișeze id-ul, stocul și pretul de achizitie al produselor din categoria cu id=2.

declare

cursor cursor\_prod is select id\_product,stock,purchase\_price from products where id\_category=2;

prod\_rec cursor\_prod%rowtype;

begin

dbms\_output.put\_line('Lista cu produsele din categoria cu id=2');

open cursor\_prod;

loop

fetch cursor\_prod into prod\_rec;

exit when cursor\_prod%NOTFOUND;

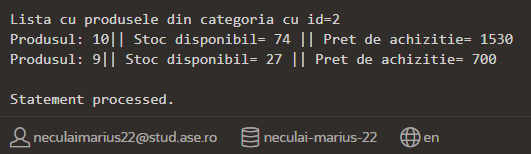
dbms\_output.put\_line('Produsul: '||prod\_rec.id\_product||'|| Stoc disponibil= '||prod\_rec.stock||' || Pret de achizitie= '||prod\_rec.purchase\_price);

end loop;

close cursor\_prod;

end;

/



1. Să se afișeze id-ul comenzii, numele clentului si valoarea totala pentru toate comenzile existente.

DECLARE

cursor cursor\_ord is

select o.id\_order,o.client\_name,sum(ol.price\*ol.quantity) as total from orders o, order\_lines ol

where o.id\_order=ol.id\_order

group by o.id\_order,client\_name;

BEGIN

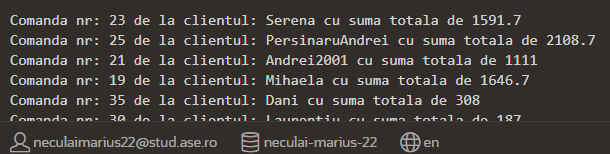
for order\_rec in cursor\_ord loop

dbms\_output.put\_line('Comanda nr: '||order\_rec.id\_order||' de la clientul: '||order\_rec.client\_name||' cu suma totala de '|| order\_rec.total);

end loop;

END;

/



1. Se afișează numele categoriilor de produse și numărul de produse din fiecare categorie.

DECLARE

cursor cursor\_cat is

select count(\*) nr,category\_name from categories c, products p where p.id\_category=c.id\_category

group by category\_name;

BEGIN

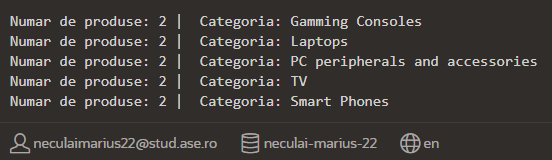
for rec in cursor\_cat loop

dbms\_output.put\_line('Numar de produse: '||rec.nr||' | Categoria: '||rec.category\_name);

end loop;

END;

/



1. Se afișează produsele care au stocul mai mare de 200 de unități.

DECLARE

cursor cursor\_prod(v\_stock number) is

select id\_product,stock from products where stock>v\_stock

order by stock desc;

v\_stoc\_introdus number;

BEGIN

v\_stoc\_introdus:=200;

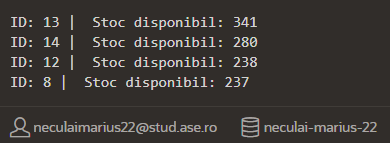
for rec in cursor\_prod(v\_stoc\_introdus) loop

dbms\_output.put\_line('ID: '||rec.id\_product||' | Stoc disponibil: '||rec.stock);

end loop;

END;

/



1. Să se afișeze utilizatorii a căror numere de telefon încep cu 0766.

DECLARE

cursor cursor\_user(v\_number varchar2) is

select id\_user,phone\_number,username from users where phone\_number like(v\_number||'%');

v\_numar\_introdus VARCHAR2(50);

BEGIN

v\_numar\_introdus:='0766';

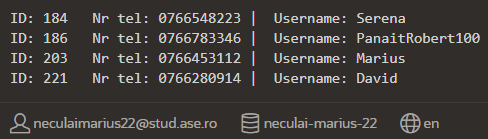
for rec in cursor\_user(v\_numar\_introdus) loop

dbms\_output.put\_line('ID: '||rec.id\_user||' Nr tel: '||rec.phone\_number||' | Username: '||rec.username);

end loop;

END;

/



# **Funcţii, proceduri, includerea acestora în pachete**

1. Să se creeze o funcție care verifică daca prețul de achiziție pentru un anumit produs este peste medie sau nu.

CREATE OR REPLACE FUNCTION este\_pret\_achizitie\_peste\_medie(p\_id\_product IN PRODUCTS.ID\_PRODUCT%TYPE)

RETURN BOOLEAN

IS

v\_purchase\_price PRODUCTS.PURCHASE\_PRICE%TYPE;

v\_avg\_purchase\_price NUMBER;

BEGIN

SELECT purchase\_price INTO v\_purchase\_price FROM products

WHERE id\_product = p\_id\_product;

SELECT AVG(purchase\_price)

INTO v\_avg\_purchase\_price

FROM products;

IF v\_purchase\_price > v\_avg\_purchase\_price THEN

RETURN TRUE;

ELSE

RETURN FALSE;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN RETURN FALSE;

WHEN OTHERS THEN RETURN FALSE;

END;

/

--Apelul funcției

DECLARE

v\_id\_product number;

BEGIN

v\_id\_product:=8;

IF (este\_pret\_achizitie\_peste\_medie(v\_id\_product)=TRUE) THEN

DBMS\_OUTPUT.PUT\_LINE('Prețul de achiziție al produsului este mai mare decât prețul de achiziție mediu.');

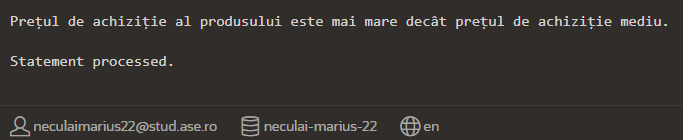
ELSE

DBMS\_OUTPUT.PUT\_LINE('Prețul de achiziție al produsului nu este mai mare decât prețul de achiziție mediu.');

END IF;

END;

/



1. Să se creeze o funcție care calculeaza suma totală cheltuită de un utilizator pentru comenzile plasate.

CREATE OR REPLACE FUNCTION suma\_totala\_comenzi(p\_id\_user IN NUMBER)

RETURN NUMBER

IS

val\_totala NUMBER := 0;

BEGIN

SELECT SUM(ol.price \* ol.quantity) INTO val\_totala FROM order\_lines ol

JOIN orders o ON ol.id\_order = o.id\_order

WHERE o.id\_user = p\_id\_user;

RETURN val\_totala;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN RETURN 0;

WHEN OTHERS THEN RETURN 0;

END;

/

--Apelul funcției

DECLARE

v\_id\_user NUMBER := 205;

v\_total\_amount NUMBER;

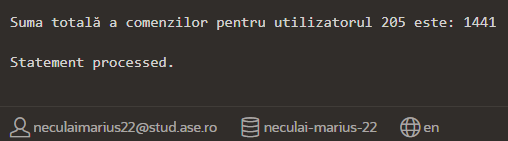
BEGIN

v\_total\_amount := suma\_totala\_comenzi(v\_id\_user);

DBMS\_OUTPUT.PUT\_LINE('Suma totală a comenzilor pentru utilizatorul ' || v\_id\_user || ' este: ' || v\_total\_amount);

END;

/



1. Să se creeze o funcție care returnează numarul de comenzi a utilizatorului trimis ca parametru.

CREATE OR REPLACE FUNCTION nr\_comenzi(p\_id\_user IN NUMBER)

RETURN NUMBER

IS

v\_order\_count NUMBER := 0;

BEGIN

SELECT COUNT(DISTINCT o.id\_order) INTO v\_order\_count FROM orders o

WHERE o.id\_user = p\_id\_user;

RETURN v\_order\_count;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 0;

WHEN OTHERS THEN

RETURN 0;

END;

/

DECLARE

v\_id\_user NUMBER := 222;

v\_order\_count NUMBER;

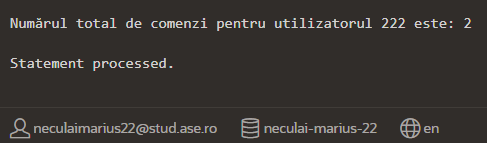
BEGIN

v\_order\_count := nr\_comenzi(v\_id\_user);

DBMS\_OUTPUT.PUT\_LINE('Numărul total de comenzi pentru utilizatorul ' || v\_id\_user || ' este: ' || v\_order\_count);

END;

/



1. Să se creeze o procedură pentru modificarea stocului unui produs.

CREATE OR REPLACE PROCEDURE modificare\_stoc (

p\_id\_product IN PRODUCTS.ID\_PRODUCT%TYPE,

p\_new\_stock IN PRODUCTS.STOCK%TYPE

) AS

BEGIN

UPDATE products

SET stock = p\_new\_stock

WHERE id\_product = p\_id\_product;

-- Verificare dacă actualizarea a avut loc

IF SQL%ROWCOUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Produsul cu ID-ul ' || p\_id\_product || ' nu a fost găsit.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Stocul pentru produsul cu ID-ul ' || p\_id\_product || ' a fost actualizat la ' || p\_new\_stock || '.');

END IF;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A apărut o eroare: ' || SQLERRM);

END;

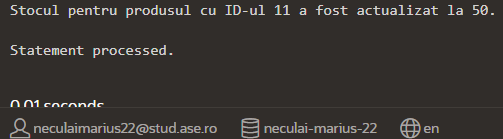
/

BEGIN

modificare\_stoc(999, 50);

END;

/



1. Să se creeze o procedură pentru adaugarea unui nou utilizator cu statutul de admin.

CREATE OR REPLACE PROCEDURE adauga\_utilizator (

p\_username IN USERS.USERNAME%TYPE,

p\_password IN USERS.PASSWORD%TYPE,

p\_email IN USERS.EMAIL%TYPE,

p\_phone\_number IN USERS.PHONE\_NUMBER%TYPE

) AS

BEGIN

INSERT INTO users (username, password, user\_type, email, creation\_date, phone\_numbe) VALUES (p\_username, p\_password, 'admin', p\_email, SYSDATE, p\_phone\_number);

DBMS\_OUTPUT.PUT\_LINE('Utilizatorul ' || p\_username || ' a fost adăugat cu succes.');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A apărut o eroare: ' || SQLERRM);

END;

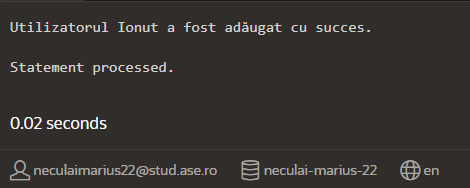
/

BEGIN

adauga\_utilizator('Ionut', 'parola123', 'ionut.mihai@yahoo.com', '0784553221');

END;

/



1. Să se creeze o proceduă pentru afisare listei de produse favorite pentru un anumit utilizator.

CREATE OR REPLACE PROCEDURE afiseaza\_favorite (

p\_id\_user IN FAVORITES.ID\_USER%TYPE

) AS

v\_product\_name PRODUCTS.PRODUCT\_NAME%TYPE;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Lista cu produse favorite pentru utilizatorul: ' || p\_id\_user);

FOR rec IN (

SELECT p.product\_name FROM favorites f

JOIN products p ON f.id\_product = p.id\_product

WHERE f.id\_user = p\_id\_user

) LOOP

DBMS\_OUTPUT.PUT\_LINE('--> ' || rec.product\_name);

END LOOP;

IF SQL%ROWCOUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Utilizatorul cu ID-ul ' || p\_id\_user || ' nu are produse favorite.');

END IF;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A apărut o eroare: ' || SQLERRM);

END;

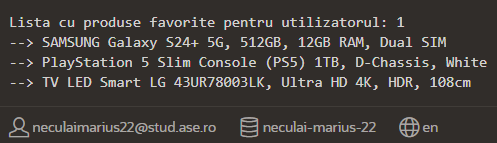
/

BEGIN

afiseaza\_favorite(1);

END;

/



1. Să se creeze un pachet in care să fie adaugate functia suma\_totala\_comenzi și procedura nr\_comenzi.

CREATE OR REPLACE PACKAGE magazin\_pck AS

FUNCTION suma\_totala\_comenzi(p\_id\_user IN NUMBER) RETURN NUMBER;

FUNCTION nr\_comenzi(p\_id\_user IN NUMBER) RETURN NUMBER;

END magazin\_pck;

/

CREATE OR REPLACE PACKAGE BODY magazin\_pck AS

FUNCTION suma\_totala\_comenzi(p\_id\_user IN NUMBER) RETURN NUMBER IS

val\_totala NUMBER := 0;

BEGIN

SELECT SUM(ol.price \* ol.quantity) INTO val\_totala FROM order\_lines ol

JOIN orders o ON ol.id\_order = o.id\_order

WHERE o.id\_user = p\_id\_user;

RETURN val\_totala;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN RETURN 0;

WHEN OT HERS THEN RETURN 0;

END suma\_totala\_comenzi;

FUNCTION nr\_comenzi(p\_id\_user IN NUMBER) RETURN NUMBER IS

v\_order\_count NUMBER := 0;

BEGIN

SELECT COUNT(DISTINCT o.id\_order) INTO v\_order\_count FROM orders o

WHERE o.id\_user = p\_id\_user;

RETURN v\_order\_count;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN RETURN 0;

WHEN OTHERS THEN RETURN 0;

END nr\_comenzi;

END magazin\_pck;

/

# **Declanşatori**

1. Să se creeze 2 declanșatori. Unul înainte de efectuarea unui update asupra stocului unui produs iar celălalt după. Aceștia vor afișa stocul produsului înainte, respectiv după modificare.

CREATE OR REPLACE TRIGGER before\_update\_stock

BEFORE UPDATE OF stock ON products

FOR EACH ROW

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Stocul produsului cu ID-ul ' || :OLD.id\_product || ' înainte de modificare: ' || :OLD.stock);

END before\_update\_stock;

/

CREATE OR REPLACE TRIGGER after\_update\_stock

AFTER UPDATE OF stock ON products

FOR EACH ROW

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Stocul produsului cu ID-ul ' || :NEW.id\_product || ' după modificare: ' || :NEW.stock);

END after\_update\_stock;

/

DECLARE

v\_stoc number;

BEGIN

v\_stoc:=142;

UPDATE products

SET stock=v\_stoc

Where id\_product=11;

END;

