

Proiect – SGBD – PL/SQL

Gestiunea unui Service GSM

Ailioaei Sorina-Elena

Seria C, Grupa 1051

1.Descrierea temei

În baza de date sunt organizate, după modelul relațional, datele cu privire la activitatea unei firme de reparații de telefoane mobile și anume, a unui Service GSM.

Această bază de date conține informații despre telefoanele mobile defecte, ce urmează a fi reparate, despre componenta telefonului ce trebuie schimbată, detalii despre client, vânzările pentru fiecare client în parte, și, de asemenea, alte accesorii ce pot fi achiziționate de aceștia, indiferent dacă doresc repararea telefonului sau nu.

Prima tabela este TELEFON, unde se găsesc următoarele atribute: COD_TELEFON (codul telefonului adus la reparat, care este și cheie primară în această tabelă), BRAND, MODEL, DEFECTIUNE.

A doua tabelă este denumită CLIENT_TEL și conține detalii referitoare la client cu următoarele coloane: COD_CLIENT(cheie primară), COD_TELEFON(cheie externă ce face legătură cu tabela TELEFON), NUME, PRENUME, NR_CONTACT. Telefonul este unic și va aparține unui singur client, însă, clientul poate să dețină mai multe telefoane, aceasta fiind o relație 1-n.

A treia tabelă o reprezintă COMPONENTA_SCHIMBATA, având detalii despre componenta defectă a telefonului ce trebuie schimbată, cu următoarele coloane: COD_COMP (cheie primară), DENUMIRE, PRET.

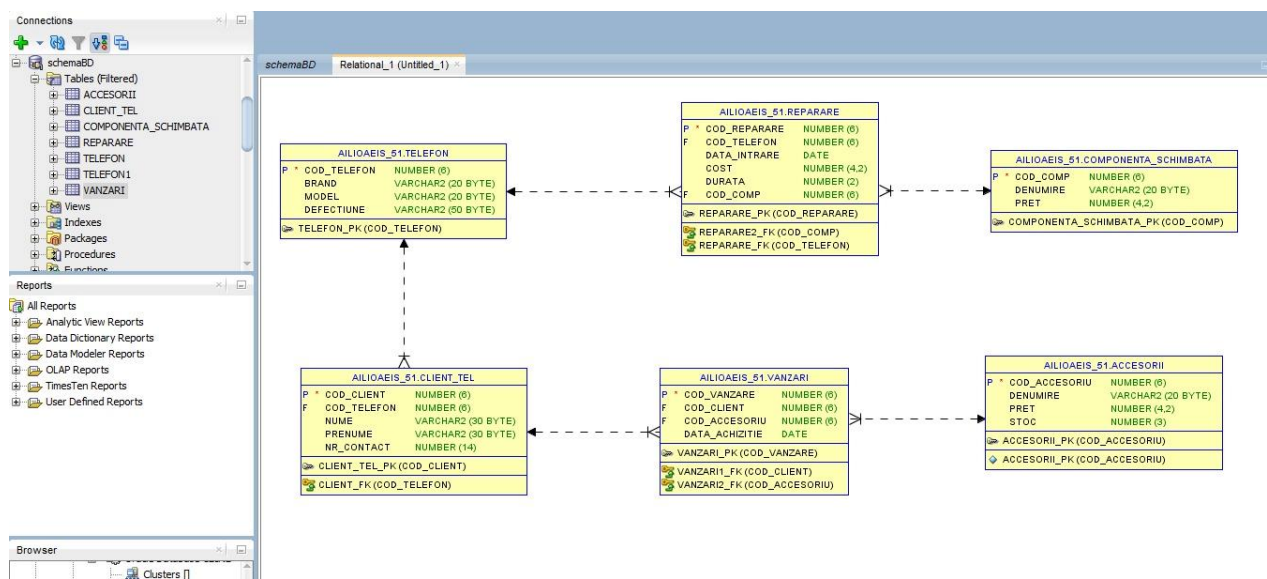
A patra tabelă denumită REPARARE, aduce informații despre procesul de reparare, cum ar fi: COD_REPARARE (cheie primară), COD_TELEFON(cheie externă ce face legătură cu tabela TELEFON), COD_COMP(cheie externă ce face legătură cu tabela COMPONENTA_SCHIMBATA), DATA_INTRARE(regăsim data în care telefonul a fost adus la reparat), COST(costul suplimentar, aferent procesului de reparație, un cost în afară de prețul componentei schimbate), DURATA(numărul de zile necesare angajatului pentru a repara

telefonul). Între această tabelă și TELEFON avem o relație 1:1 și anume, repararea se face pe un singur telefon, si invers, un telefon este supus unei singure reparări.

Următoarea tabelă, ACCESORII, cuprinde anumite accesorii pentru telefoane mobile cum ar fi: huse, încărcătoare, etc., de care dispune firma respectiva și care sunt puse la vânzare pentru clienți indiferent dacă au un telefon de reparat sau nu. Tabela are următoarele atribute: COD_ACCESORIU(cheie primară), DENUMIRE, PRET.

Ultima tabelă, VANZARI, reține detalii despre vânzarea de accesorii făcută cu fiecare client în parte, aceasta deține atributele: COD_VANZARE(cheie primara), COD_CLIENT(cheie externă ce face legătură cu tabela CLIENT_TEL), COD_ACCESORIU(cheie externă ce face legătură cu tabela ACCESORII), DATA.

2.Schema conceptuală



3. Construirea tabelelor:

1. Tabela TELEFON:

```

CREATE TABLE TELEFON (
    COD_TELEFON NUMBER(6) PRIMARY KEY,
    BRAND VARCHAR2(20),
    MODEL VARCHAR2(20),
    DEFECTIUNE VARCHAR2(50) );
    
```

2. Tabela CLIENT_TEL

```

CREATE TABLE CLIENT_TEL(
    COD_CLIENT NUMBER(6) PRIMARY KEY,
    
```

```
COD_TELEFON NUMBER(6),  
NUME VARCHAR2(30),  
PRENUME VARCHAR2(30),  
NR_CONTACT NUMBER(14),  
CONSTRAINT client_fk FOREIGN KEY(COD_TELEFON) REFERENCES  
TELEFON(COD_TELEFON) );
```

3. Tabela COMPONENTA_SCHIMBATA

```
CREATE TABLE COMPONENTA_SCHIMBATA(  
COD_COMP NUMBER(6) PRIMARY KEY,  
DENUMIRE VARCHAR2(20),  
PRET NUMBER(4, 2) );
```

4. Tabela REPARARE

```
CREATE TABLE REPARARE(  
COD_REPARARE NUMBER(6) PRIMARY KEY,  
COD_TELEFON NUMBER(6),  
DATA_INTRARE DATE,  
COST NUMBER(4, 2),  
DURATA NUMBER(2),  
CONSTRAINT reparare_fk FOREIGN KEY(COD_TELEFON) REFERENCES  
TELEFON(COD_TELEFON) );
```

5. Tabela ACCESORII

```
CREATE TABLE ACCESORII(  
COD_ACCESORIU NUMBER(6),  
DENUMIRE VARCHAR2(20),  
PRET NUMBER(4, 2) );
```

6. Tabela VANZARI

```
CREATE TABLE VANZARI(  
COD_VANZARE NUMBER(6) PRIMARY KEY,  
COD_CLIENT NUMBER(6),  
COD_ACCESORIU NUMBER(6),
```

DATA_ACHIZITIE DATE);

Exemple cu operațiile LDD:

Adaugare cheie primara pentru tabela ACCESORII

- ALTER TABLE ACCESORII ADD CONSTRAINT accesorii_pk PRIMARY KEY(COD_ACCESORIU);

Adăugați coloana STOC în tabela ACCESORII având tipul Number(3)

- ALTER TABLE ACCESORII ADD(STOC NUMBER(3));

Cheie externa a tabelii VANZARI cu tabela CLIENT_TEL

- ALTER TABLE VANZARI ADD CONSTRAINT vanzari1_fk FOREIGN KEY(COD_CLIENT) REFERENCES CLIENT_TEL(COD_CLIENT);

Cheie externa a tabelii VANZARI cu tabela ACCESORII.

- ALTER TABLE VANZARI ADD CONSTRAINT VANZARI2_fk FOREIGN KEY(COD_ACCESORIU) REFERENCES ACCESORII(COD_ACCESORIU);

Stergeti tabela COMPONENTA_SCHIMBATA

- DROP TABLE COMPONENTA_SCHIMBATA; (dupa care o creem din nou)

Adăugați restricția de integritate reparare_ck care să nu permită introducerea în câmpul COST a unor valori mai mici de 0.

- ALTER TABLE REPARARE ADD CONSTRAINT reparare_ck CHECK (COST >0);

Adăugați restricția de integritate reparare2_ck care să nu permită introducerea în câmpul DURATA a unor valori mai mici de 0 și mai mari decât 20.

- ALTER TABLE REPARARE ADD CONSTRAINT reparare2_ck CHECK(DURATA BETWEEN 0 AND 20);

Adăugați coloana COD_COMP în tabela REPARARE având tipul Number(6).

- ALTER TABLE REPARARE ADD(COD_COMP NUMBER(6));

Cheie externa a tabelii REPARARE cu tabela COMPONENTA_SCHIMBATA.

- ALTER TABLE REPARARE ADD CONSTRAINT REPARARE2_fk FOREIGN KEY(COD_COMP) REFERENCES COMPONENTA_SCHIMBATA(COD_COMP);

Modificati precizia în 5 coloanei PRET în tabela COMPONENTA_SCHIMBATA.

- ALTER TABLE COMPONENTA_SCHIMBATA MODIFY (PRET NUMBER(5,2));

Modificati valoarea maxima de caractere pe care o poate avea coloana DENUMIRE în tabela COMPONENTA_SCHIMBATA

```
• ALTER TABLE COMPONENTA_SCHIMBATA MODIFY(DENUMIRE  
VARCHAR2(40));
```

4. Exerictii:

1. Sa se afiseze telefoanele cu codul cuprins intre 100 si 106, mai putin pe 102 si 103.

```
set serveroutput on;
```

```
declare
```

```
type t_tel is record(brand varchar2(50), model varchar2(50));
```

```
v_tel t_tel;
```

```
begin
```

```
for i in 100..105 loop
```

```
if i not between 102 and 103 then
```

```
select brand, model into v_tel from telefon where cod_telefon=i;
```

```
dbms_output.put_line('Telefonul '||i||' este '||v_tel.brand||' '||v_tel.model);
```

```
end if;
```

```
end loop;
```

```
end; /
```

The screenshot displays the Oracle SQL Developer interface. On the left, the 'Tables (Filtered)' pane shows the database structure, including tables like ACCESORII, CLIENTI, and COMPONENTA_SCHIMBATA. The main editor window contains a PL/SQL script designed to loop through phone codes from 100 to 105, skipping 102 and 103, and displaying the brand and model of each phone. The script is as follows:

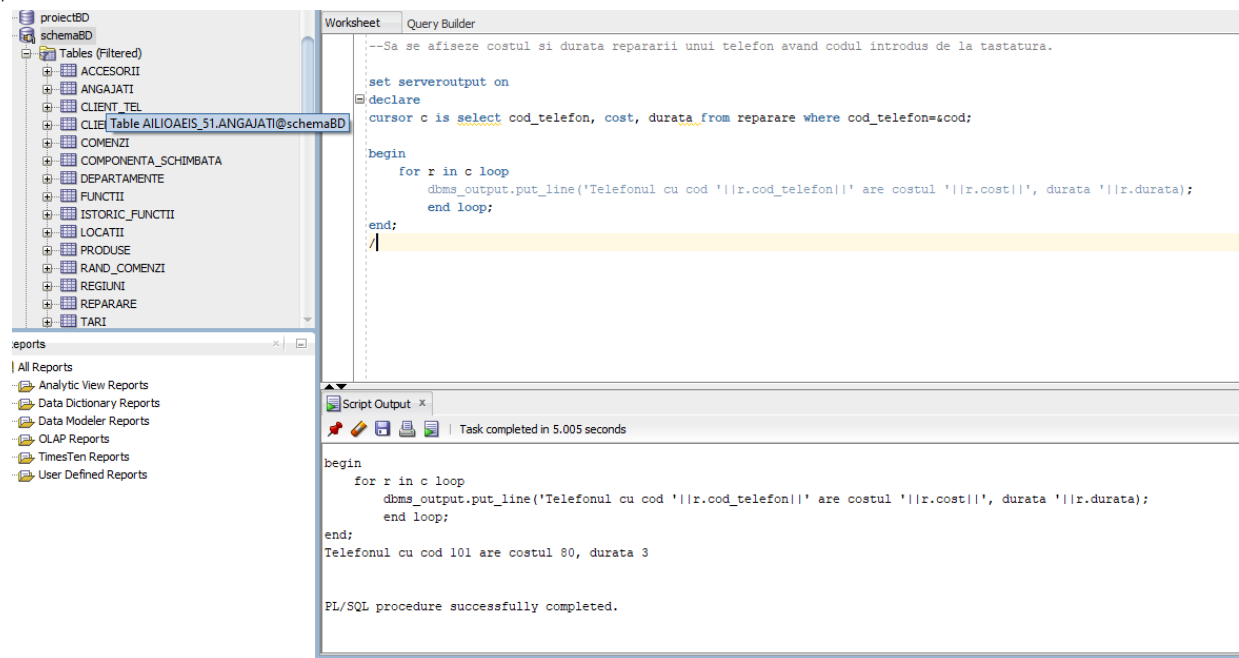
```
-- Sa se afiseze telefoanele cu codul cuprins intre 100 si 106, mai putin pe 102 si 103.  
set serveroutput on;  
declare  
type t_tel is record(brand varchar2(50), model varchar2(50));  
v_tel t_tel;  
begin  
for i in 100..105 loop  
if i not between 102 and 103 then  
select brand, model into v_tel from telefon where cod_telefon=i;  
dbms_output.put_line('Telefonul '||i||' este '||v_tel.brand||' '||v_tel.model);  
end if;  
end loop;  
end;  
/
```

Below the script, the 'Script Output' window shows the results of the execution:

```
Task completed in 0.418 seconds  
Telefonul 100 este iPhone 14  
Telefonul 101 este Samsung S23  
Telefonul 104 este iPhone 13  
Telefonul 105 este Samsung Hold  
  
PL/SQL procedure successfully completed.
```

2. Sa se afiseze costul si durata reparării unui telefon având codul introdus de la tastatura.

```
set serveroutput on
declare
cursor c is select cod_telefon, cost, durata from reparare where cod_telefon=&cod;
begin
    for r in c loop
        dbms_output.put_line('Telefonul cu cod '||r.cod_telefon||' are costul '||r.cost||', durata '||r.durata);
    end loop;
end;
/
```



3. Sa se afiseze telefonul fiecărui client in parte.

```
set serveroutput on

declare

cursor c1 is select * from client_tel;--retine clientii

cursor c2(p_cod_telefon number) is select brand, model from telefon where
cod_telefon=p_cod_telefon; --retine telefoanele

begin

    for i in c1 loop

        dbms_output.put_line('Clientul '||i.num||' '||i.prenume||' are telefonul: ');

        for j in c2(i.cod_telefon) loop
```

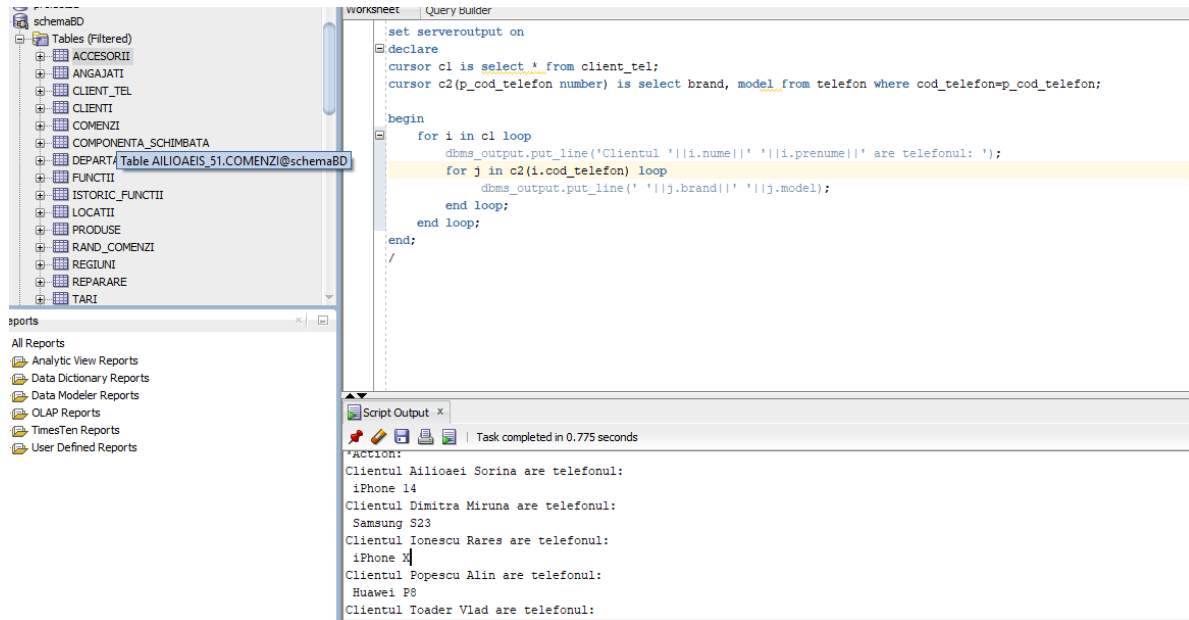
```

        dbms_output.put_line('||j.brand||' ||j.model);
    end loop;

end loop;

end;

```



4. Creați un bloc PL/SQL prin care se dublează prețul accesoriului al cărui cod este citit de la tastatură.

```

select * from accesorii;

set serveroutput on

declare

v_cod accesorii.cod_accesoriu%type:=&c;

begin

update accesorii set pret=2*pret where cod_accesoriu=v_cod;

if sql%rowcount = 0 then

    dbms_output.put_line('Nu s-a actualizat nicio linie');

else dbms_output.put_line('S-au actualizat '||sql%rowcount||' linii');

end if;

end;

/

```

The screenshot shows the Oracle SQL Developer interface. On the left, the 'Table AILIOAEIS_51.COMPONENTA_SCHIMBATA@schemaBD' is selected in the 'Tables (Filtered)' pane. The 'Query Builder' pane on the right contains the following PL/SQL code:

```

select * from accesorii;
set serveroutput on

declare
v_cod accesorii.cod_accesoriu%type:=sc;
begin
  update accesorii set pret=2*pret where cod_accesoriu=v_cod;
  if sql%rowcount = 0 then
    dbms_output.put_line('Nu s-a actualizat nicio linie');
  dbms_output.put_line('S-au actualizat '||sql%rowcount||' linii');
  end if;
end;
/

```

The 'Script Output' window at the bottom shows the execution results:

```

Task completed in 0.444 seconds

S-au actualizat 1 linii

PL/SQL procedure successfully completed.

```

COD_ACCESORIU	DENUMIRE	PRET	STOC
1	HUSA IPHONE	50	80

5. Sa se afişeze lista cu numele, prenumele si numărul de contact al clienţilor ar cărui brand de telefon este iPhone.

```
set serveroutput on;
```

```
declare
```

```
cursor c is select ct.num, ct.prenume, ct.nr_contact from client_tel ct, telefon t where
ct.cod_telefon=t.cod_telefon and t.brand='iPhone';
```

```
client_rec c%rowtype;
```

```
begin
```

```
dbms_output.put_line('Lista cu clientii care au telefon iPhone: ');
```

```
open c;
```

```
loop
```

```
fetch c into client_rec;
```

```
exit when c%notfound;
```

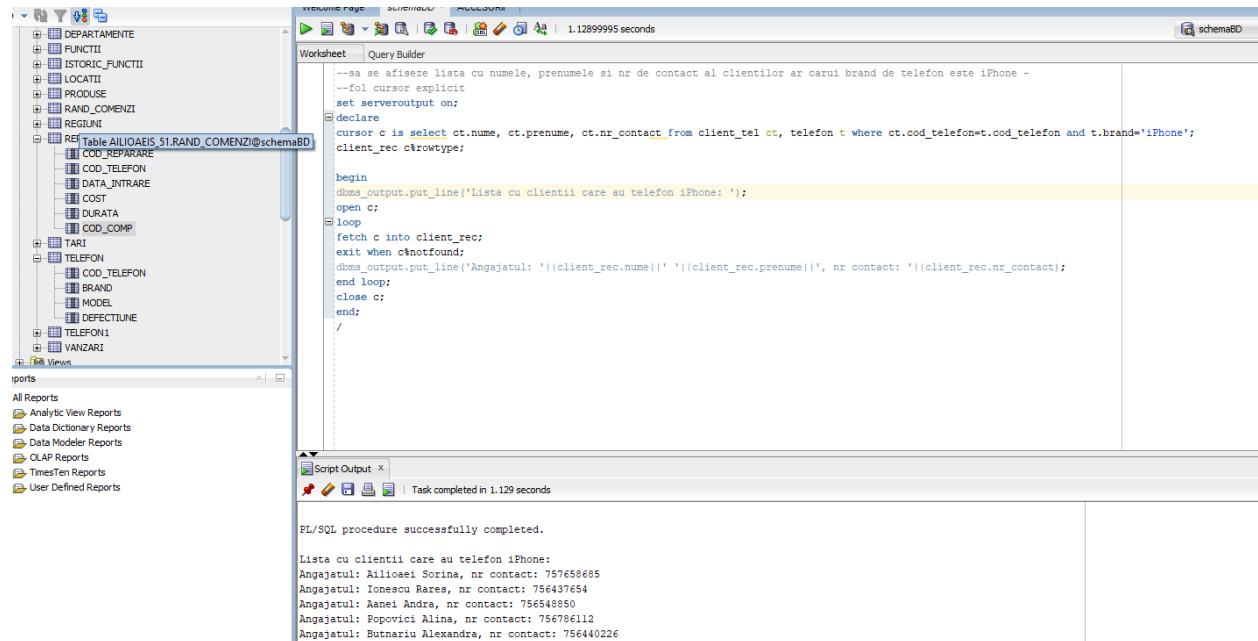
```
dbms_output.put_line('Angajatul: '||client_rec.num||' '||client_rec.prenume||', nr contact:
'||client_rec.nr_contact);
```

```
end loop;
```

```
close c;
```


end;

/



6. Sa se afişeze costul total al reparării (cost + preţ componenta schimbata) al fiecărui telefon in parte.

```
set serveroutput on;
```

```
begin
```

```
dbms_output.put_line('Total reparare pentru fiecare telefon din service de la momentul actual: ');
```

```
for tel_rec in (select r.cod_telefon cod, (r.cost+c.pret) cost_total from reparare r,
componenta_schimbata c
```

```
where r.cod_comp=c.cod_comp)
```

```
loop
```

```
dbms_output.put_line('Telefonul '||tel_rec.cod||' are un cost total de reparare de:
'||tel_rec.cost_total||' lei.');
```

```
end loop;
```

```
end;
```

/

```
select * from componenta_schimbata;
```

```
select* from reparare;
```

The screenshot shows a database query editor with a schema tree on the left, a query builder in the center, and a script output window at the bottom. The schema tree includes tables like NR_CONTACT, CLIENTI, COMENZI, and REPARARE. The query builder contains a PL/SQL script that calculates the total repair cost for each phone. The script output window shows the results of the script, including the total repair cost for each phone and the cost of the replaced component.

```

-- sa se afiseze costul total al repararii (cost + pret componenta schimbata) al fiecarui telefon
set serveroutput on;
begin
  dbms_output.put_line('Total reparare pentru fiecare telefon din service de la momentul actual: ');
  for tel_rec in (select r.cod_telefon cod, (r.cost+c.pret) cost_total from reparare r, componenta_schimbata c
    where r.cod_comp=c.cod_comp)
  loop
    dbms_output.put_line('Telefonul '||tel_rec.cod||' are un cost total de reparare de: '||tel_rec.cost_total||' lei. ');
  end loop;
end;
/
select * from componenta_schimbata;
select * from reparare;

```

8 rows selected.

```

Total reparare pentru fiecare telefon din service de la momentul actual:
Telefonul 101 are un cost total de reparare de: 480 lei.
Telefonul 110 are un cost total de reparare de: 480 lei.
Telefonul 103 are un cost total de reparare de: 460 lei.
Telefonul 108 are un cost total de reparare de: 460 lei.
Telefonul 107 are un cost total de reparare de: 200 lei.
Telefonul 102 are un cost total de reparare de: 300 lei.
Telefonul 105 are un cost total de reparare de: 300 lei.

```

7. Sa se afişeze codul, brandul si modelul telefoanelor ale căror coduri sunt mai mici decât codul introdus de la tastatură.

```
set serveroutput on;
```

```
declare
```

```
cursor c_tel(p_max_cod telefon.cod_telefon%type) is select cod_telefon, brand, model from
telefon where cod_telefon<p_max_cod;
```

```
v_cod_telefon telefon.cod_telefon%TYPE;
```

```
v_brand telefon.brand%TYPE;
```

```
v_model telefon.model%TYPE;
```

```
v_max_cod telefon.cod_telefon%TYPE;
```

```
begin
```

```
  dbms_output.put_line('Lista telefoanelor care au codul mai mic decat cel introdus: ');
```

```
  v_max_cod := &v_max_cod;
```

```
  open c_tel(v_max_cod);
```

```
  loop
```

```
    fetch c_tel into v_cod_telefon, v_brand, v_model;
```

```
    exit when c_tel%NOTFOUND;
```

```
    dbms_output.put_line('Telefonul ' || v_cod_telefon || ': ' || v_brand || ' ' || v_model);
```

```

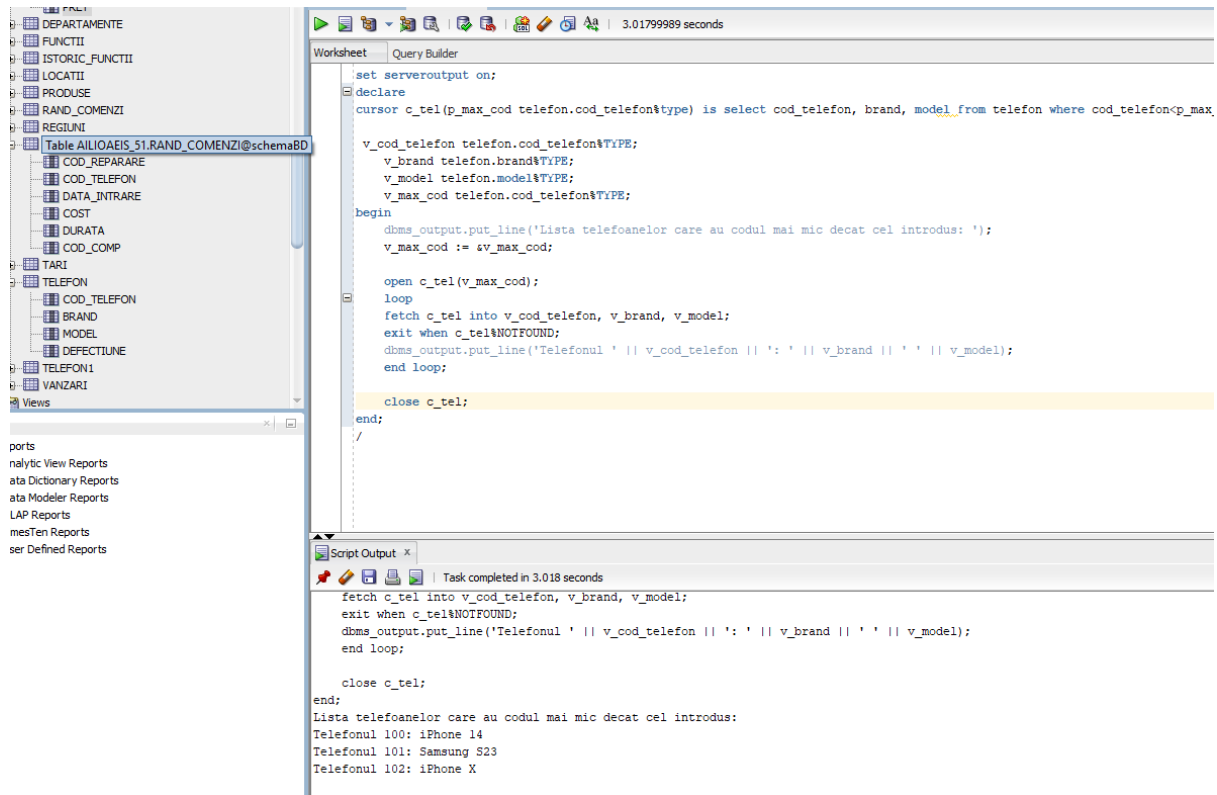
end loop;

close c_tel;

end;

```

/



UTILIZAREA EXCEPTIILOR:

8. Creați un bloc PL/SQL pentru a selecta codul de vânzare, codul accesoriului si data vânzărilor încheiate într-un an introdus de la tastatura. Dacă interogarea returnează mai mult de o valoare sau niciuna, tratați excepția cu o rutină de tratare corespunzătoare și afișați mesajul.

```
set serveroutput on
```

```
declare
```

```
v_cod vanzari.cod_vanzare%type;
```

```
v_cod_acc vanzari.cod_accesoriu%type;
```

```
v_data vanzari.data_achizitie%type;
```

```
v_an number:=&an;
```

```
begin
```

```
select cod_vanzare,cod_accesoriu, data_achizitie into v_cod,v_cod_acc, v_data from vanzari
```

```

where extract (year from data_achizitie) = v_an;
dbms_output.put_line(v_cod|| '||v_cod_acc||' ||v_data);
exception
when NO_DATA_FOUND then dbms_output.put_line('In anul '||v_an||' nu au fost vandute
accesorii!');
when TOO_MANY_ROWS then dbms_output.put_line('In anul '||v_an||' au fost vandute
multiple accesorii');

declare
cursor c is select cod_vanzare,cod_accesoriu, data_achizitie from vanzari where
extract(year from data_achizitie)= v_an;
begin
for rec_c in c loop
dbms_output.put_line(rec_c.cod_vanzare|| '||rec_c.cod_accesoriu||'
'||rec_c.data_achizitie);
end loop;
end;
when others then dbms_output.put_line('A aparut o alta eroare'||SQLERRM); --sql error msg
end;
/

```

The screenshot shows the SQL Developer interface. On the left, the 'Table Browser' pane displays the database schema, including tables like DEPARTAMENTE, FUNCTII, and VANZARI. The 'Vanzari' table is selected, showing columns for COD_VANZARE, COD_CLIENT, COD_ACESORIU, and DATA_ACHIZITIE. The main window displays a PL/SQL script that defines a cursor 'c' to select data from the 'vanzari' table for a specific year (v_an). The script includes an exception block to handle 'NO_DATA_FOUND' and 'TOO_MANY_ROWS' errors. The 'Script Output' pane at the bottom shows the execution results, indicating that the task was completed successfully in 2.265 seconds. The output message is: 'In anul 2018 nu au fost vandute accesorii!'.

```

declare
v_cod_vanzari.cod_vanzare%type;
v_cod_acc_vanzari.cod_accesoriu%type;
v_data_vanzari.data_achizitie%type;
v_an number:=&an;
begin
select cod_vanzare,cod_accesoriu, data_achizitie into v_cod,v_cod_acc, v_data from vanzari
where extract (year from data_achizitie) = v_an;
dbms_output.put_line(v_cod|| '||v_cod_acc||' ||v_data);
exception
when NO_DATA_FOUND then dbms_output.put_line('In anul '||v_an||' nu au fost vandute accesorii!');
when TOO_MANY_ROWS then dbms_output.put_line('In anul '||v_an||' au fost vandute multiple accesorii');

declare
cursor c is select cod_vanzare,cod_accesoriu, data_achizitie from vanzari where extract(year from data_achizitie)= v_an;
begin
for rec_c in c loop
dbms_output.put_line(rec_c.cod_vanzare|| '||rec_c.cod_accesoriu||' ||rec_c.data_achizitie);
end loop;
end;
when others then dbms_output.put_line('A aparut o alta eroare'||SQLERRM); --sql error msg
end;
/

```

Task completed in 2.265 seconds

```

when TOO_MANY_ROWS then dbms_output.put_line('In anul '||v_an||' au fost vandute multiple accesorii');

declare
cursor c is select cod_vanzare,cod_accesoriu, data_achizitie from vanzari where extract(year from data_achizitie)= v_an;
begin
for rec_c in c loop
dbms_output.put_line(rec_c.cod_vanzare|| '||rec_c.cod_accesoriu||' ||rec_c.data_achizitie);
end loop;
end;
when others then dbms_output.put_line('A aparut o alta eroare'||SQLERRM); --sql error msg
end;

In anul 2018 nu au fost vandute accesorii!

PL/SQL procedure successfully completed.

```

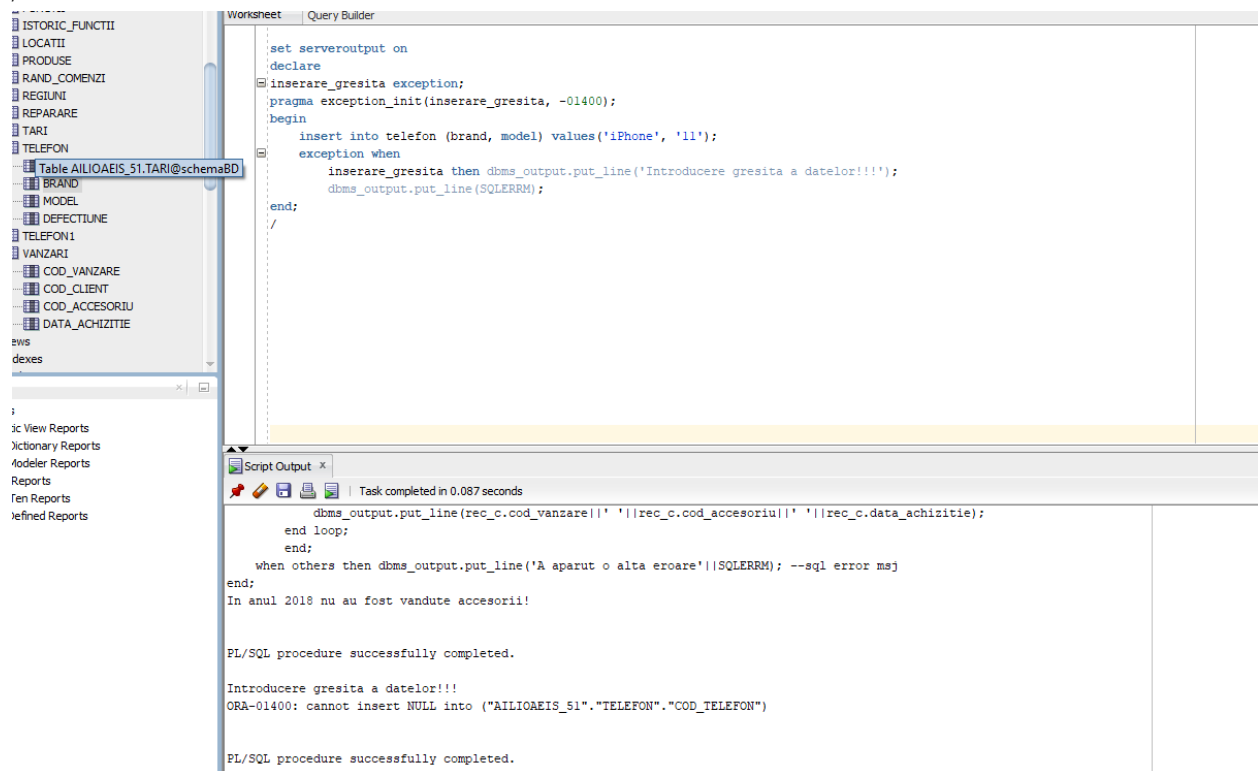
9. Sa se insereze in tabela telefon un nou telefon cu brand iPhone, model 11, dar fără introducerea codului.

set serveroutput on

```

declare
inserare_gresita exception;
pragma exception_init(inserare_gresita, -01400);
begin
    insert into telefon (brand, model) values('iPhone', '11');
    exception when
        inserare_gresita then dbms_output.put_line('Introducere gresita a datelor!!!');
        dbms_output.put_line(SQLERRM);
end;
/

```



10. Creați un bloc PL/SQL prin care se dublează prețul accesoriului al cărui cod este citit de la tastatură. În cazul în care acesta nu există (comanda UPDATE nu realizează nicio modificare) se va invoca o excepție. Tratați excepția prin afișarea unui mesaj.

```

set serveroutput on
declare
v_cod accesorii.cod_accesoriu%type:=&c;
update_ex exception;

begin
    update accesorii set pret=2*pret where cod_accesoriu=v_cod;
    if sql%rowcount = 0 then
        raise update_ex;
    else dbms_output.put_line('S-au actualizat '||sql%rowcount||' linii');

```

```
end if;
```

```
exception when
```

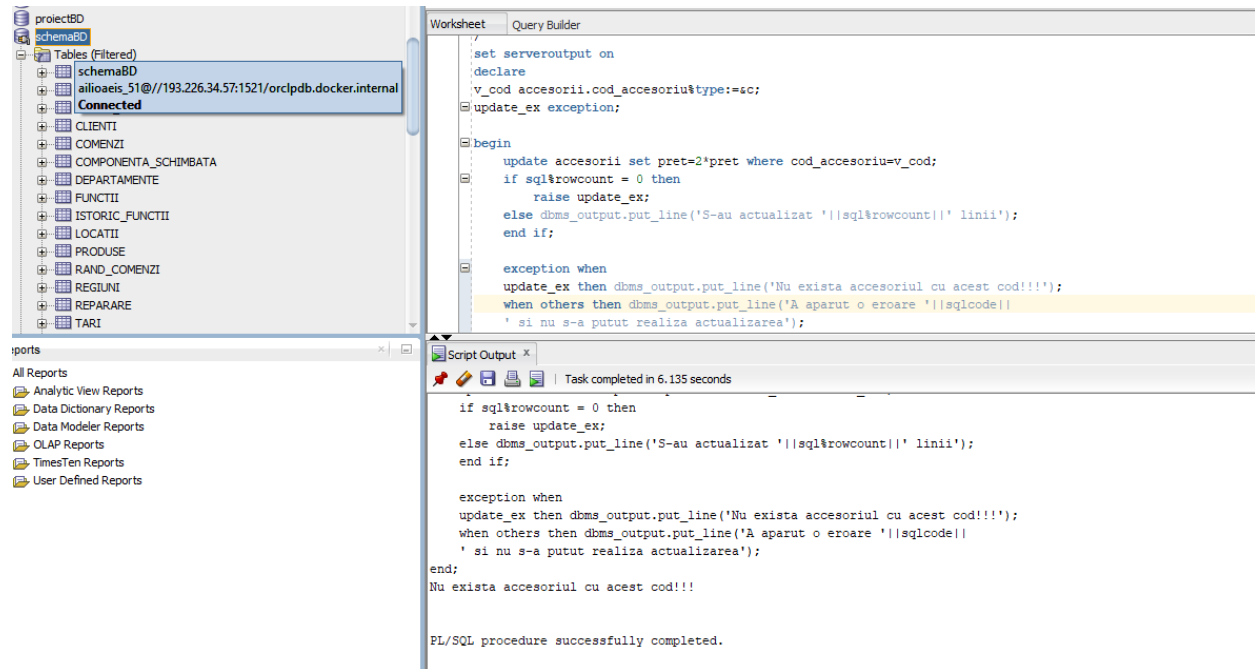
```
update_ex then dbms_output.put_line('Nu exista accesoriul cu acest cod!!!');
```

```
when others then dbms_output.put_line('A aparut o eroare '||sqlcode||
```

```
' si nu s-a putut realiza actualizarea');
```

```
end;
```

```
/
```



11. Sa se ștergă accesoriile care nu au fost niciodată vândute. Tratați excepțiile ce pot apărea in program.

```
set serveroutput on
```

```
declare
```

```
delete_ex exception;
```

```
begin
```

```
delete from accesorii where cod_accesoriu not in (select cod_accesoriu from vanzari);
```

```
if SQL%ROWCOUNT=0 then raise delete_ex;
```

```
else dbms_output.put_line('S-au sters '||SQL%ROWCOUNT||' accesorii');
```

```
end if;
```

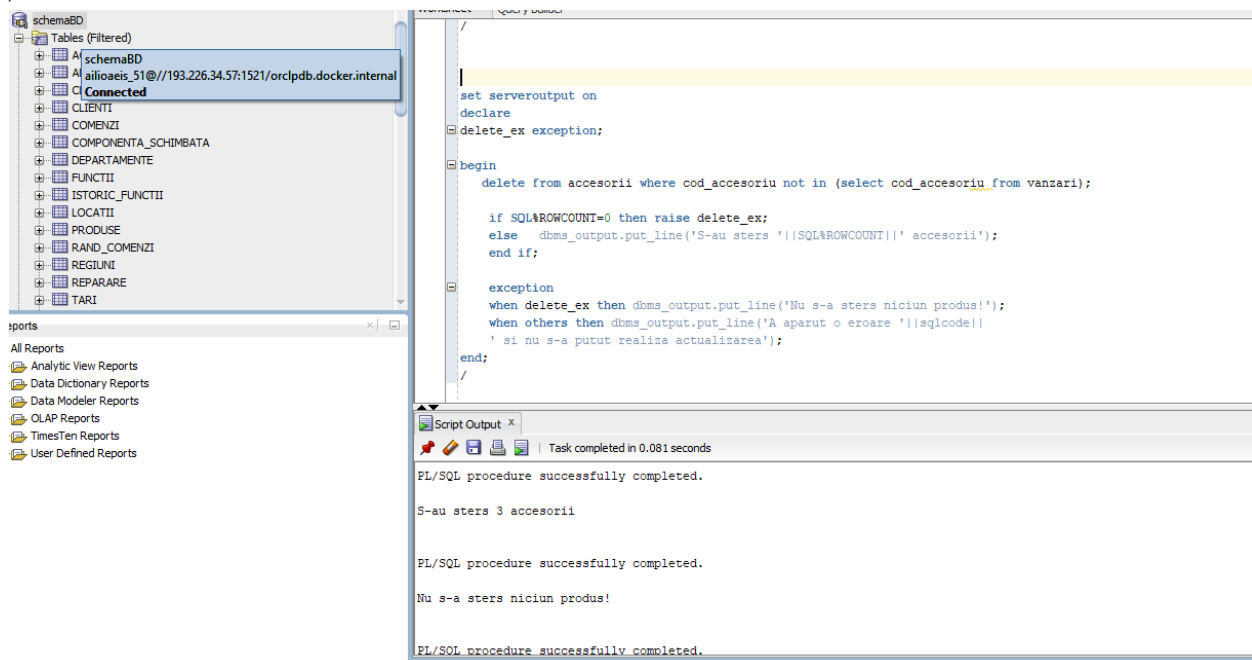
```
exception
```

```
when delete_ex then dbms_output.put_line('Nu s-a sters niciun produs!');
```

```
when others then dbms_output.put_line('A aparut o eroare '||sqlcode||
```

```
' si nu s-a putut realiza actualizarea');
```

```
end;
```



12. Să se trateze eroarea apărută în cazul în care se inserează o înregistrare cu un cod_telefon(primary key) deja utilizat.

declare

v_cod_telefon telefon.cod_telefon%type := 100;--OBS: in tabela exista deja pk cu val 100

v_brand telefon.brand%type := 'Samsung';

v_model telefon.model%type := 'Galaxy';

v_defectiune telefon.defectiune%type := 'Ecran spart';

begin

begin

insert into telefon (cod_telefon, brand, model, defectiune) values (v_cod_telefon, v_brand, v_model, v_defectiune);

dbms_output.put_line('Inregistrare inserata cu succes!');

exception

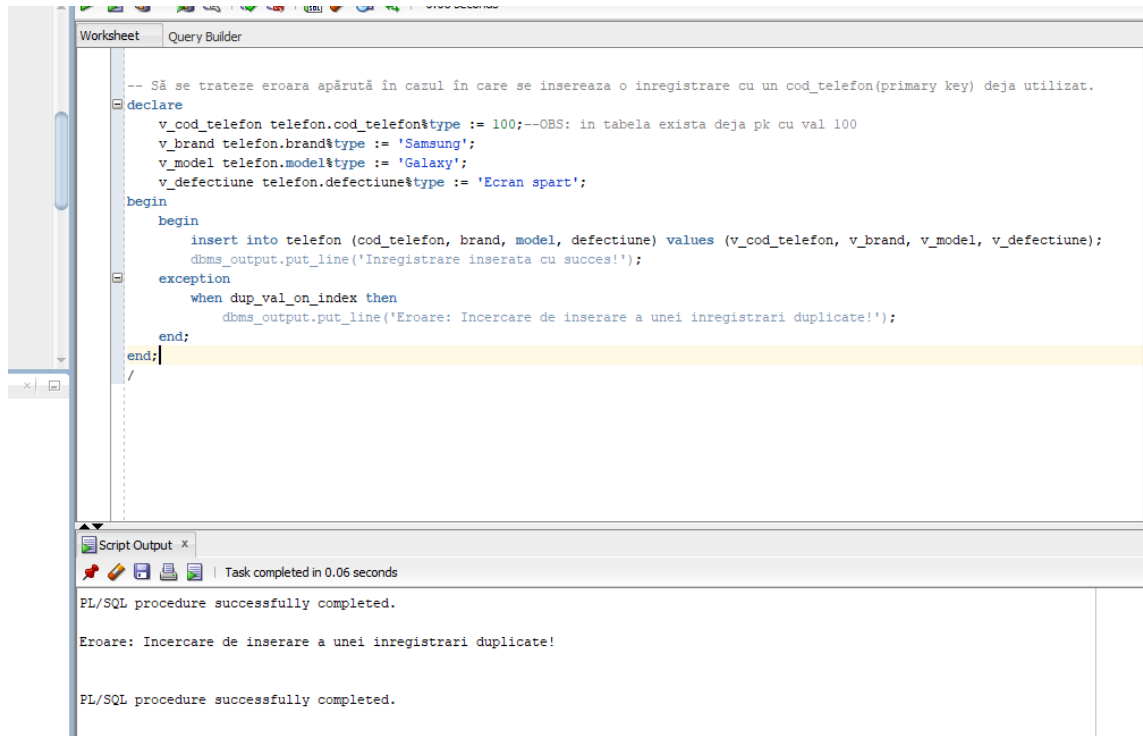
when dup_val_on_index then

dbms_output.put_line('Eroare: Incercare de inserare a unei inregistrari duplicate!');

end;

end;

/



```
-- Să se trateze eroarea apărută în cazul în care se inserează o înregistrare cu un cod_telefon(primary key) deja utilizat.
declare
v_cod_telefon telefon.cod_telefon%type := 100;--OBS: in tabela exista deja pk cu val 100
v_brand telefon.brand%type := 'Samsung';
v_model telefon.model%type := 'Galaxy';
v_defectiune telefon.defectiune%type := 'Ecran spart';
begin
begin
insert into telefon (cod_telefon, brand, model, defectiune) values (v_cod_telefon, v_brand, v_model, v_defectiune);
dbms_output.put_line('Inregistrare inserata cu succes!');
exception
when dup_val_on_index then
dbms_output.put_line('Eroare: Incercare de inserare a unei inregistrari duplicate!');
end;
end;
/
```

Script Output x

Task completed in 0.06 seconds

PL/SQL procedure successfully completed.

Eroare: Incercare de inserare a unei inregistrari duplicate!

PL/SQL procedure successfully completed.

UTILIZAREA FUNCȚIILOR, PROCEDURILOR ȘI PACHETELOR

1. Creați procedura top 5 reparații prin care să afișați informații despre primele 5 reparații (se va realiza filtrarea după durata reparației). Apelați procedura.

create or replace procedure top_5_reparatii as

cursor c is select cod_reparare, cost, durata from reparare order by durata asc;

begin

for r in c loop

exit when c%rowcount>5;

dbms_output.put_line('Reparatia ||r.cod_reparare|| costa ||r.cost||

' si are durata || r.durata|| zile.');

end loop;

end;

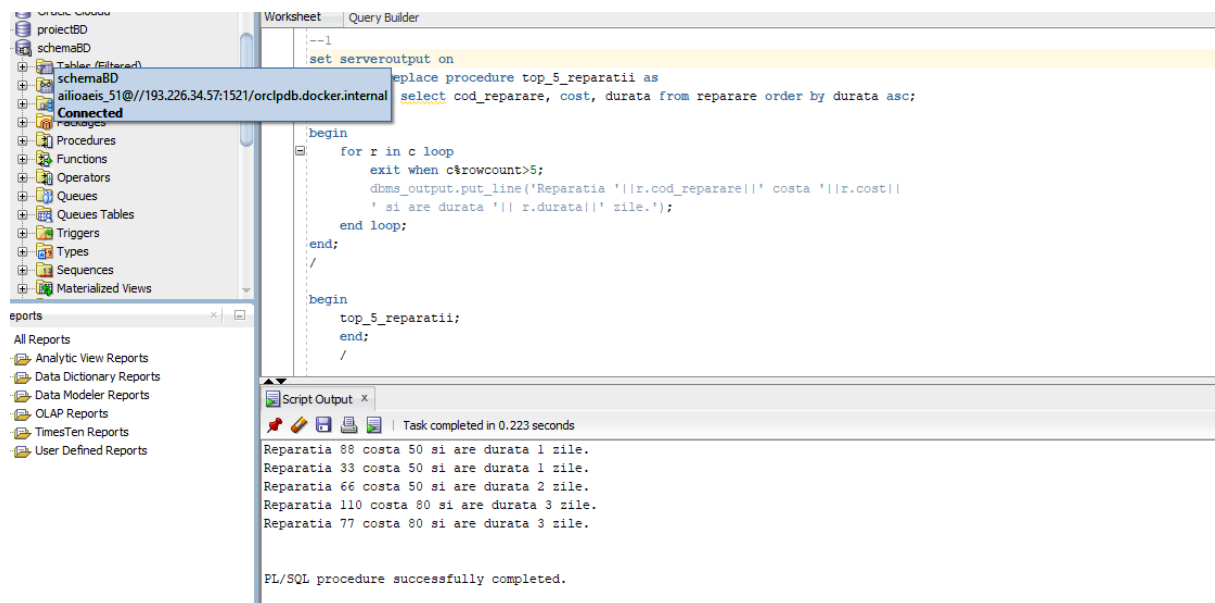
/

begin

top_5_reparatii;

end;

/



2. Construiți o procedura valoare_accesoriu care sa afișeze prețul si stocul fiecărui accesoriu vândut la anul indicat ca parametru de intrare. Apelați procedura.

```

CREATE OR REPLACE PROCEDURE valoare_accesoriu(p_an NUMBER) AS
CURSOR c IS
  SELECT a.pret, a.stoc
  FROM accesorii a, vanzari v
  WHERE a.cod_accesoriu = v.cod_accesoriu AND EXTRACT(YEAR FROM
v.data_achizitie) = p_an;
  v_count NUMBER := 0;
  fara_val exception;
BEGIN
  SELECT COUNT(*) INTO v_count FROM accesorii a, vanzari v
  WHERE a.cod_accesoriu = v.cod_accesoriu AND EXTRACT(YEAR FROM
v.data_achizitie) = p_an;

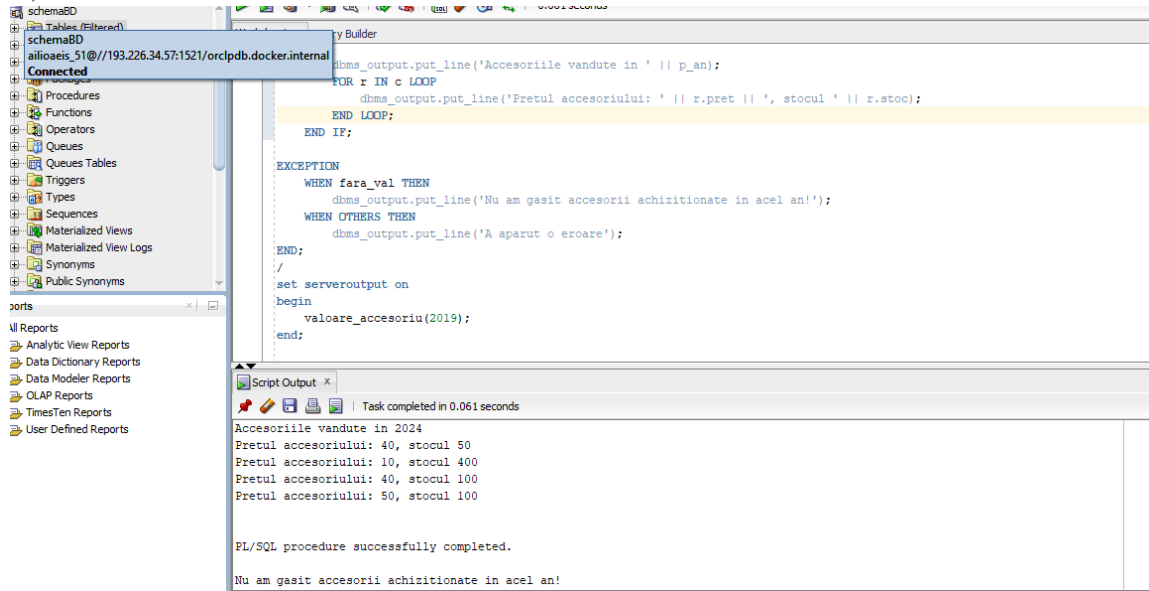
  IF v_count = 0 THEN
    RAISE fara_val;
  ELSE
    dbms_output.put_line('Accesoriile vandute in ' || p_an);
    FOR r IN c LOOP
      dbms_output.put_line('Pretul accesoriului: ' || r.pret || ', stocul ' || r.stoc);
    END LOOP;
  END IF;

EXCEPTION
  WHEN fara_val THEN
    dbms_output.put_line('Nu am gasit accesorii achizitionate in acel an!');
```

```

WHEN OTHERS THEN
    dbms_output.put_line('A aparut o eroare');
END;
/
set serveroutput on
begin
    valoare_accesoriu(2024);
end;

```



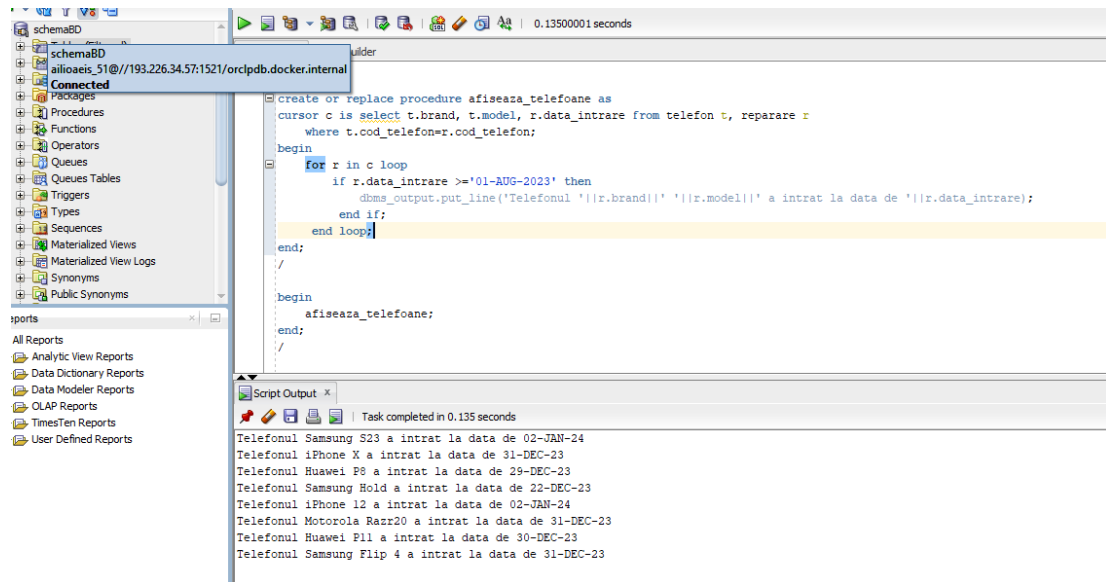
- Realizați o procedură afiseaza_telefoane pentru a selecta brandul, modelul, dar si data intrării in service. Parcurgeți fiecare rând al cursorului si in cazul in care data depășește 01.08.2023, afișați informațiile preluate. Apelați procedura.

```

create or replace procedure afiseaza_telefoane as
cursor c is select t.brand, t.model, r.data_intrare from telefon t, reparare r
    where t.cod_telefon=r.cod_telefon;
begin
    for r in c loop
        if r.data_intrare >='01-AUG-2023' then
            dbms_output.put_line('Telefonul '||r.brand||' '||r.model||' a intrat la data de
'||r.data_intrare);
        end if;
    end loop;
end;
/
begin
    afiseaza_telefoane;
end;

```

/



- Realizați o procedură prin intermediul căreia sa scădeți cu 10% costul reparării telefoanelor care au intrat in service într-un an primit de la tastatura. Returnați numărul de telefoane pentru care se realizează aceasta actualizare sau tratați in mod corespunzător o excepție daca nu exista niciun telefon pentru care se modifica costul.

set serveroutput on;

create or replace procedure marire_cost(p_an number, p_nr out number)

is

exceptie exception;

begin

update reparare

set cost=cost*0.9

where extract(year from data_intrare)=p_an;

if sql%found then

p_nr:=sql%rowcount;

else

raise exceptie;

end if;

exception

when exceptie then p_nr:=0;

end;

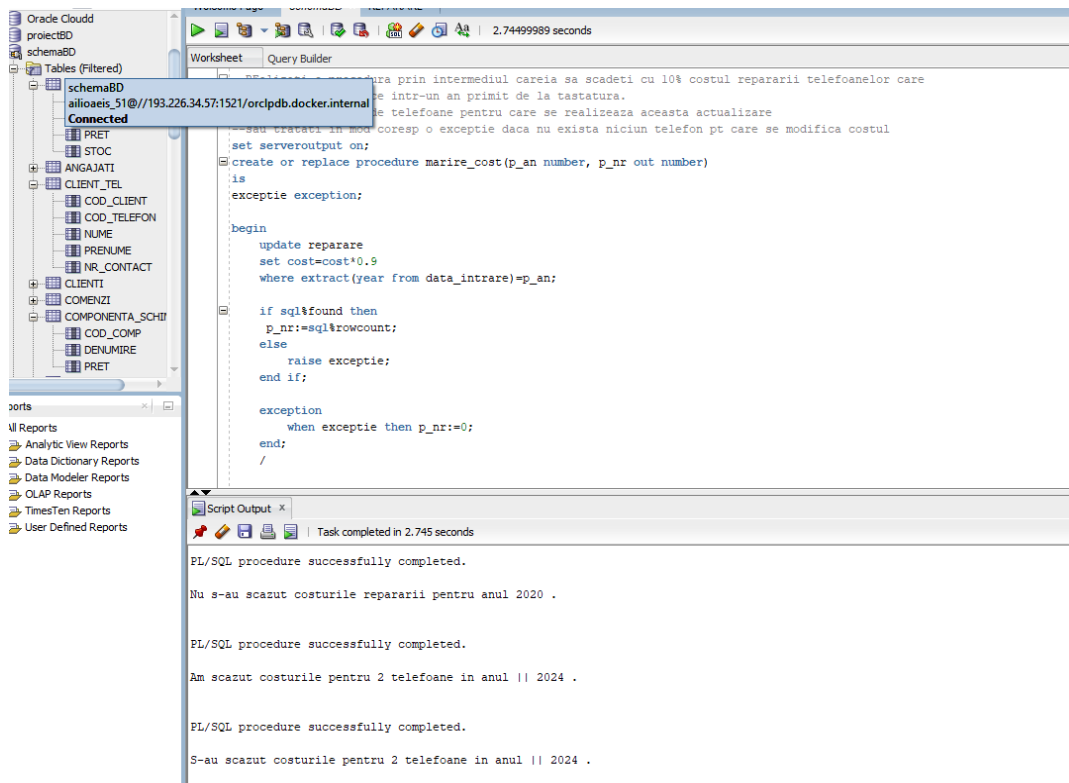
/

set verify off;

```

declare
nr_telefoane number;
v_an number:=&an;
begin
marire_cost(v_an, nr_telefoane);
if nr_telefoane>0 then
dbms_output.put_line('S-au scazut costurile pentru '||nr_telefoane ||' telefoane in anul || '|| v_an||'
.);
else
dbms_output.put_line('Nu s-au scazut costurile repararii pentru anul '||v_an||' .');
end if;
end;

```



5. Construiți o funcție val_reparatie care sa determine valoarea reparației (cost+pret) pentru un telefon cu codul introdus ca parametru de intrare.

```

create or replace function val_reparatie(p_cod number) return number as
val_reparatie number;
v_pret number;
v_cost number;

begin
select c.pret, r.cost into v_pret, v_cost from reparare r, componenta_schimbata c

```

```

        where r.cod_comp=c.cod_comp and r.cod_telefon=p_cod;
    val_reparatie := v_pret+v_cost;
    return val_reparatie;

```

```

exception when no_data_found then return -1;

```

```

end;

```

```

/

```

```

set serveroutput on

```

```

declare

```

```

    valoare number;

```

```

begin

```

```

    valoare:=val_reparatie(104);

```

```

    dbms_output.put_line('Costul reparatiei pentru acest telefon este de: '||valoare);

```

```

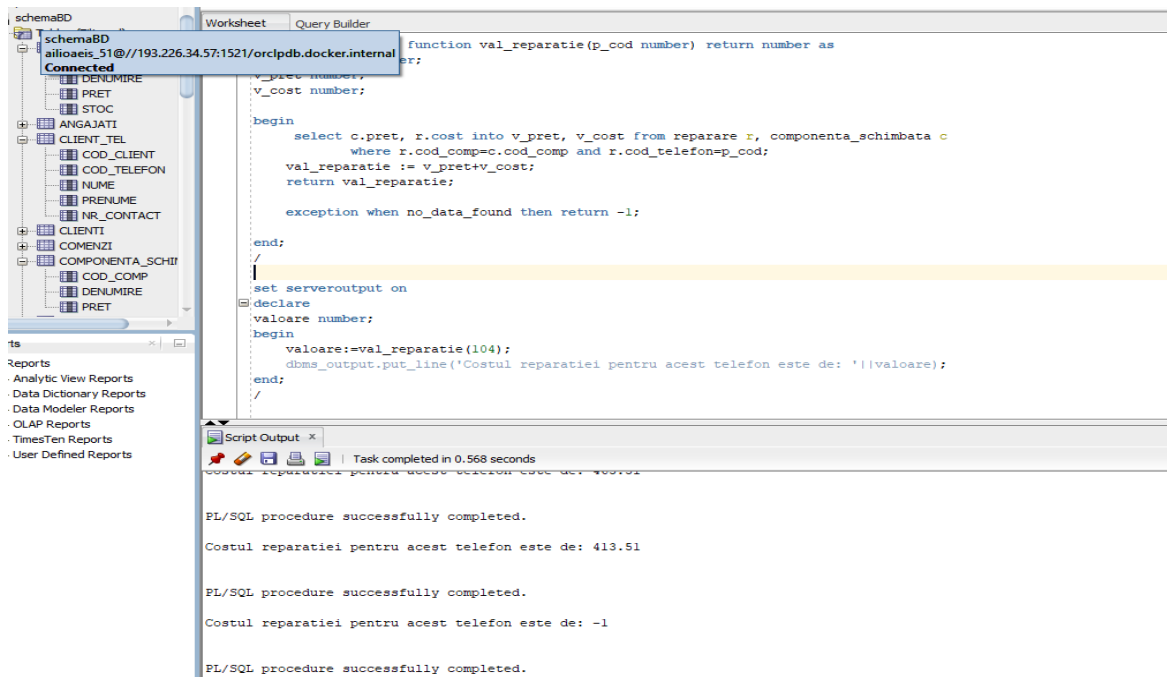
end;

```

```

/

```



- Realizați o funcție care calculează valoarea totală a accesoriului care are codul primit de la tastatură. Tratați în mod corespunzător o excepție dacă nu există niciun accesoriu cu acel cod.

```

create or replace function valoare_totala_acc(p_cod number) return number as
    val_totala number;

```

```

begin

```

```

    select stoc*pret into val_totala from accesorii where cod_accesoriu = p_cod;

```

```

return val_totala;
exception
    when no_data_found then
        return -1;
end;
/

set serveroutput on
declare
val_totala number;
v_id_acc number:=&id_acc;
begin
    val_totala:=valoare_totala_acc(v_id_acc);
    if(val_totala>0) then
        dbms_output.put_line('Valoarea totala a accesoriului este '||val_totala);
    else
        dbms_output.put_line (' Nu exista accesoriu cu acest cod!!');
    end if;
end;
/

```

The screenshot displays the Oracle SQL Developer environment. On the left, the 'schemaBD' tree shows a list of tables including ACCESORII, COD_ACCESORII, PRET, STOC, ANGAJATI, CLIENT_TEL, COD_CLIENT, COD_TELEFON, NUME, PRENUME, NR_CONTACT, CLIENTI, COMENZI, COMPONENTA_SCHII, COD_COMP, DENUMIRE, and PRET. The main window shows a PL/SQL script in the 'Query Builder' tab. The script defines a function 'valoare_totala_acc' and a procedure that calls it. The procedure uses a bind variable '&id_acc' to pass the value 2000. The script is executed, and the 'Script Output' window at the bottom shows the results: 'Task completed in 1.693 seconds', 'Nu exista accesoriu cu acest id!!', 'PL/SQL procedure successfully completed.', and 'Valoarea totala a accesoriului cu id2 este 2000'.

```

create or replace function valoare_totala_acc(p_cod number) return number as
val_totala number;
begin
    select stoc pret into val_totala from accesorii where cod_accesoriu = p_cod;
    return val_totala;
exception
    when no_data_found then
        return -1;
end;
/

set serveroutput on
declare
val_totala number;
v_id_acc number:=&id_acc;
begin
    val_totala:=valoare_totala_acc(v_id_acc);
    if(val_totala>0) then
        dbms_output.put_line('Valoarea totala a accesoriului cu id '|| v_id_acc ||' este '||val_totala);
    else
        dbms_output.put_line (' Nu exista accesoriu cu acest id!!');
    end if;
end;
/
select * from accesorii;

```

Script Output x

Task completed in 1.693 seconds

Nu exista accesoriu cu acest id!!

PL/SQL procedure successfully completed.

Valoarea totala a accesoriului cu id2 este 2000

PL/SQL procedure successfully completed.

7. Realizați o funcție care calculează vechimea reparației in ani pentru telefonul cu codul introdus ca parametru.

```
create or replace function vechime_reparatie(p_cod number) return number as
v_vechime number;
fara_rep exception;
begin
    select extract(year from sysdate)-extract(year from data_intrare) into v_vechime
        from reparare where cod_telefon=p_cod;
    return v_vechime;
    if(sql%notfound) then raise fara_rep;
    end if;
    exception
        when fara_rep then dbms_output.put_line('Nu exista reparatie cu acest cod!!!');
end;
/

set serveroutput on
declare
vechime number;
begin
    vechime:=vechime_reparatie(102);
    dbms_output.put_line('Reparatia a avut loc acum '||vechime||' ani');
end;
/
```

The screenshot displays the Oracle SQL Developer environment. The left pane shows the 'projectBD' schema with various database objects. The main window is the 'Query Builder' showing a PL/SQL script. The script defines a function 'vechime_reparatie' and then calls it with the value 102, outputting the result. The bottom pane, 'Script Output', shows the execution results. It indicates that no data was found for the first call (resulting in 0 years) and that the procedure completed successfully for the second call (resulting in 1 year).

```
end if;
exception
    when fara_rep then dbms_output.put_line('Nu exista reparatie cu acest cod!!!');
end;
/

set serveroutput on
declare
vechime number;
begin
    vechime:=vechime_reparatie(102);
    dbms_output.put_line('Reparatia a avut loc acum '||vechime||' ani');
end;
/
```

Script Output

Task completed in 0.092 seconds

01403. 00000 - "no data found"
Cause: No data was found from the objects.
Action: There was no data from the objects which may be due to end of fetch.
Reparatia a avut loc acum 0 ani

PL/SQL procedure successfully completed.

Reparatia a avut loc acum 1 ani

PL/SQL procedure successfully completed.

8. Realizați o funcție care sa returneze daca costul de reparare a unui telefon este mai mare decât media costurilor tuturor telefoanelor. Tratați în mod corespunzător o excepție dacă nu există niciun telefon cu codul transmis ca parametru.

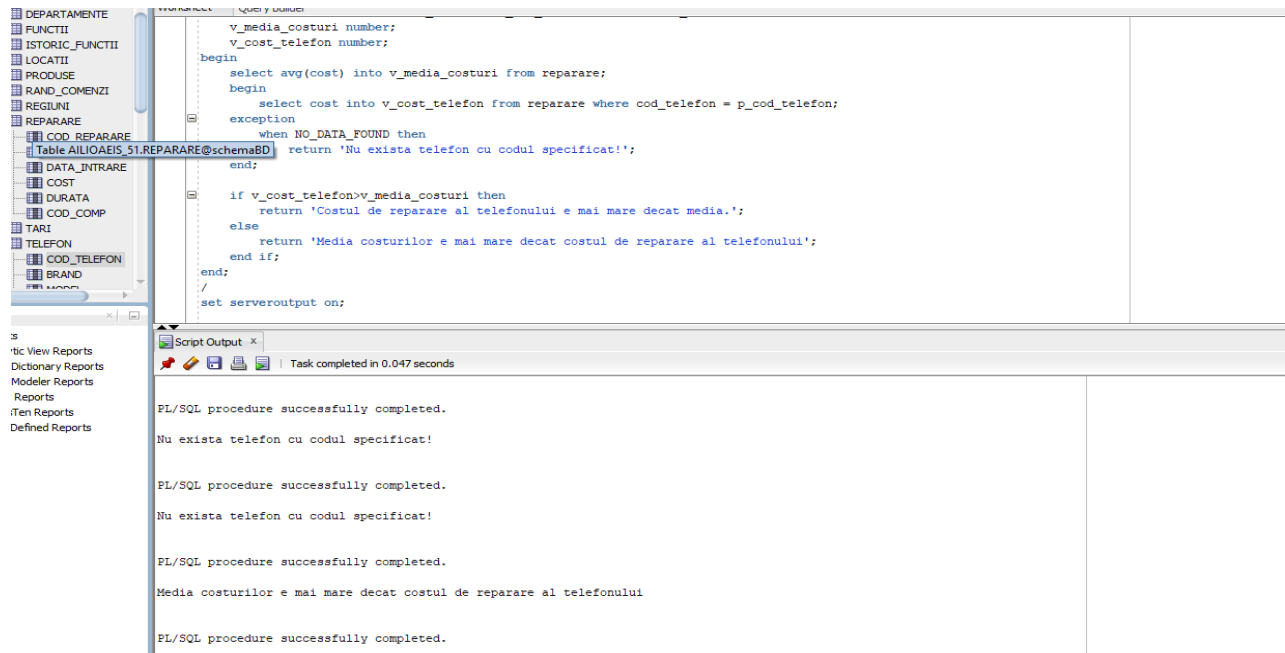
```
create or replace function cost_reparare(p_cod_telefon telefon.cod_telefon%type) return
varchar2 is
    v_media_costuri number;
    v_cost_telefon number;
begin
    select avg(cost) into v_media_costuri from reparare;
    begin
        select cost into v_cost_telefon from reparare where cod_telefon = p_cod_telefon;
    exception
        when NO_DATA_FOUND then
            return 'Nu există telefon cu codul specificat!';
    end;

    if v_cost_telefon > v_media_costuri then
        return 'Costul de reparare al telefonului e mai mare decât media.';
    else
        return 'Media costurilor e mai mare decât costul de reparare al telefonului';
    end if;
end;

/

set serveroutput on;
declare
    v_cod_telefon number := 102;
    v_rezultat varchar2(100);
begin
    v_rezultat := cost_reparare(v_cod_telefon);
    dbms_output.put_line(v_rezultat);
end;

/
```

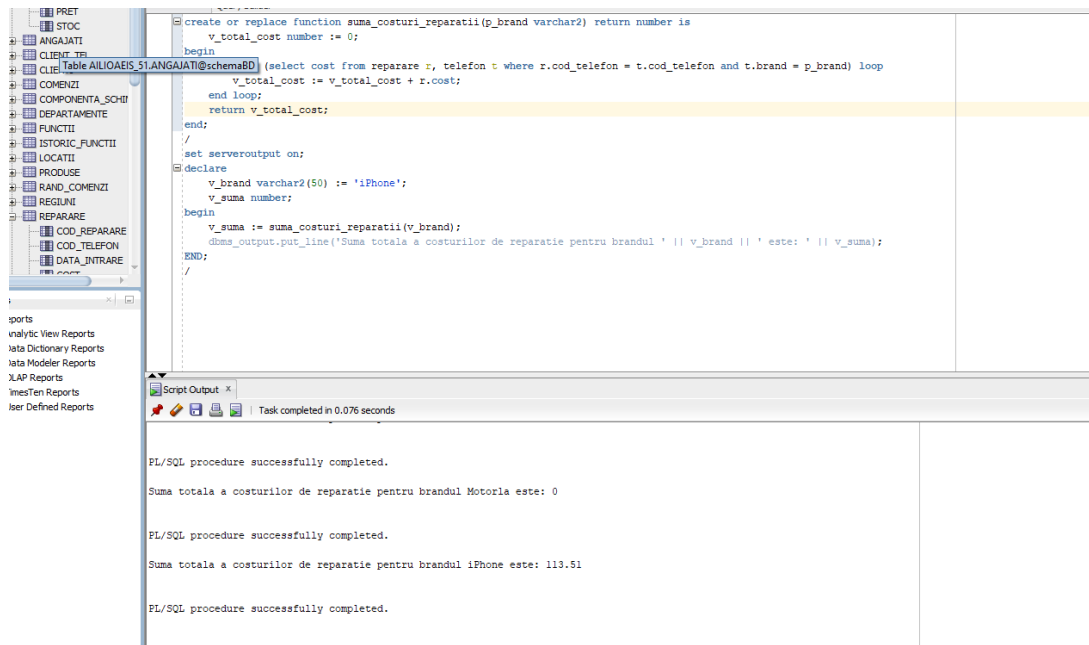



9. Realizați o funcție care sa returneze suma totală a costurilor de reparație pentru un anumit brand de telefoane.

```

create or replace function suma_costuri_reparatii(p_brand varchar2) return number is
  v_total_cost number := 0;
begin
  for r in (select cost from reparare r, telefon t where r.cod_telefon = t.cod_telefon and t.brand =
p_brand) loop
    v_total_cost := v_total_cost + r.cost;
  end loop;
  return v_total_cost;
end;
/
set serveroutput on;
declare
  v_brand varchar2(50) := 'iPhone';
  v_suma number;
begin
  v_suma := suma_costuri_reparatii(v_brand);
  dbms_output.put_line('Suma totala a costurilor de reparatie pentru brandul ' || v_brand || ' este:
' || v_suma);
end;
/

```



10. Creați un pachet detalii_reparatii care sa conțină funcții cu privire la reparațiile efectuate de service.

- Top 5 reparații ordonate crescător in funcție de durata
- Afișare telefoane care au data de intrare in service după 1 august 2023

--SPECIFICATII

create or replace package detalii_reparatii is

procedure top_5_reparatii;

procedure afiseaza_telefoane;

end detalii_reparatii;

--BODY:

create or replace package BODY detalii_reparatii is

procedure top_5_reparatii as

cursor c is select cod_reparare, cost, durata from reparare order by durata asc;

begin

for r in c loop

exit when c%rowcount>5;

dbms_output.put_line('Reparatia '||r.cod_reparare||' costa '||r.cost||
' si are durata '|| r.durata||' zile.');

end loop;

end;

procedure afiseaza_telefoane as

cursor c is select t.brand, t.model, r.data_intrare from telefon t, reparare r

where t.cod_telefon=r.cod_telefon;

```

begin
  for r in c loop
    if r.data_intrare >='01-AUG-2023' then
      dbms_output.put_line('Telefonul '||r.brand||' '||r.model||' a intrat la data de '||r.data_intrare);
    end if;
  end loop;
end;
end detalii_reparatii;

```

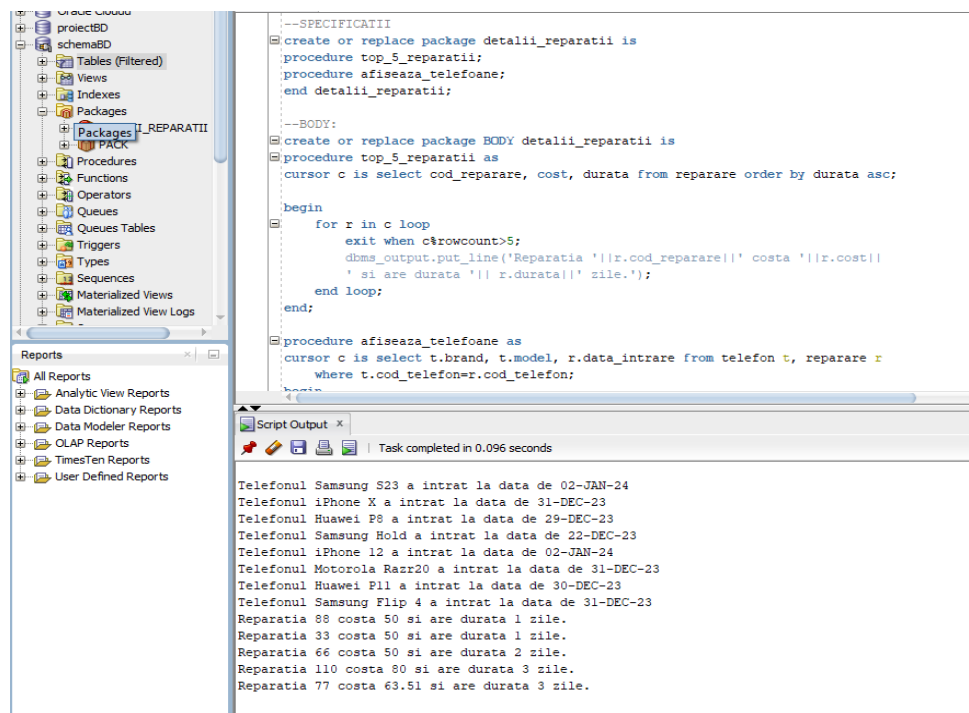
SET SERVEROUTPUT ON;

BEGIN

detalii_reparatii.afiseaza_telefoane;

detalii_reparatii.top_5_reparatii;

END;



11. Creați un pachet detalii_vanzari care sa conțină funcții si proceduri cu privire la vânzările efectuate de service:

- O procedura care sa conțină prețul si stocurile accesoriilor vândute intr-un an introdus ca parametru.
- O funcție care sa calculeze valoarea totala a unui accesoriu .

--SPECIFICATII

create or replace package detalii_vanzari is

procedure valoare_accesoriu(p_an number);

function valoare_totala_acc(p_cod number) return number;

```
end detalii_vanzari;
```

```
--BODY
```

```
create or replace package BODY detalii_vanzari is
```

```
procedure valoare_accesoriu(p_an number) as
```

```
cursor c is select a.pret, a.stoc from accesorii a, vanzari v
```

```
where a.cod_accesoriu=v.cod_accesoriu and extract(year from v.data_achizitie)=p_an;
```

```
v_count number := 0;
```

```
fara_val exception;
```

```
BEGIN
```

```
select count(*) into v_count from accesorii a, vanzari v
```

```
where a.cod_accesoriu = v.cod_accesoriu and extract(year from v.data_achizitie) = p_an;
```

```
if v_count = 0 then
```

```
raise fara_val;
```

```
else
```

```
dbms_output.put_line('Accesoriile vandute in ' || p_an);
```

```
FOR r IN c LOOP
```

```
dbms_output.put_line('Pretul accesoriului: ' || r.pret || ', stocul ' || r.stoc);
```

```
END LOOP;
```

```
end if;
```

```
EXCEPTION
```

```
WHEN fara_val THEN
```

```
dbms_output.put_line('Nu am gasit accesorii achizitionate in acel an!');
```

```
WHEN OTHERS THEN
```

```
dbms_output.put_line('A aparut o eroare');
```

```
END;
```

```
function valoare_totala_acc(p_cod number) return number as
```

```
val_totala number;
```

```
BEGIN
```

```
select stoc*pret into val_totala from accesorii where cod_accesoriu = p_cod;
```

```
return val_totala;
```

```
exception
```

```
when no_data_found then
```

```
return -1;
```

```
END;
```

```
END detalii_vanzari;
```

```
--apel 1
```

```
set serveroutput on
```

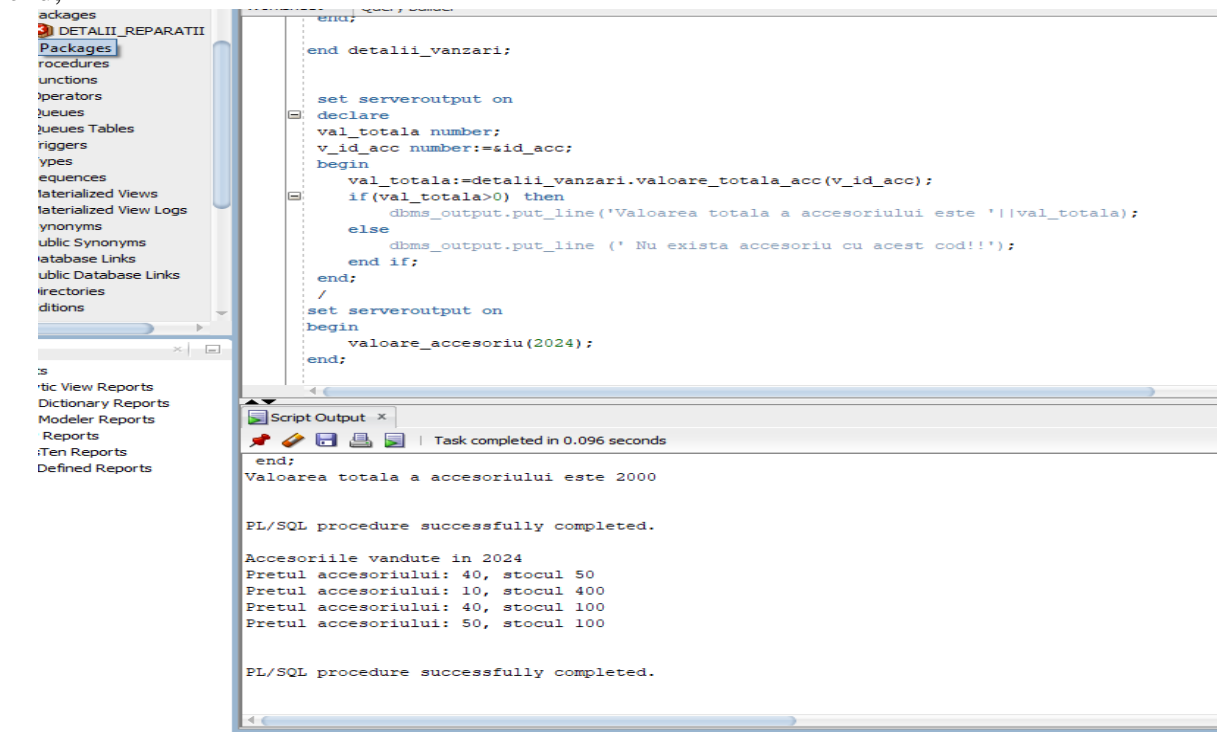
```
declare
```

```
val_totala number;
```

```

v_id_acc number:=&id_acc;
begin
    val_totala:=detalii_vanzari.valoare_totala_acc(v_id_acc);
    if(val_totala>0) then
        dbms_output.put_line('Valoarea totala a accesoriului este '||val_totala);
    else
        dbms_output.put_line (' Nu exista accesoriu cu acest cod!!');
    end if;
end;
/
--apel 2
set serveroutput on
begin
    valoare_accesoriu(2024);
end;

```



UTILIZAREA DECLANȘATORILOR

1. Sa se creeze un declanșator care sa împiedice inserarea unui accesoriu cu preț mai mare decât preț maxim

```

create or replace trigger restrictie_pret
before insert or update on accesorii

```

```

for each row
declare
v_pret_max accesorii.pret%type;
begin
    select max(pret) into v_pret_max from accesorii;
    if :new.pret>v_pret_max then
        raise_application_error(-20000, 'Nu se poate insera un accesoriu cu pretul mai mare decat
cel maxim!!!');
    end if;
end;
/

set serveroutput on
insert into accesorii values(11, 'cablu USB', 95 , 50);

```

The screenshot displays the Oracle SQL Developer environment. On the left, the 'Connections' pane shows the 'Table AIIIOAEIS_51.ACCESORII@schemaBD' selected. The main workspace shows the SQL script for creating a trigger and inserting a new record. The trigger 'restrictie_pret' is created to prevent inserting accessories with a price greater than the maximum price in the 'accesorii' table. The insert statement attempts to insert a 'cablu USB' with a price of 95 and a stock of 50. The script output shows the trigger was created successfully, but the insert statement failed with an error: 'ORA-06512: at 'AIIIOAEIS_51.RESTRICTIE_PRET', line 6'.

```

-- TRIGGERI
--1 Sa se creeze un declansator care sa impiedice inserarea unui accesoriu cu pretul mai mare decat pretul maxim

create or replace trigger restrictie_pret
before insert or update on accesorii
for each row
declare
v_pret_max accesorii.pret%type;
begin
    select max(pret) into v_pret_max from accesorii;
    if :new.pret>v_pret_max then
        raise_application_error(-20000, 'Nu se poate insera un accesoriu cu pretul mai mare decat cel maxim!!!');
    end if;
end;
/

set serveroutput on
insert into accesorii values(11, 'cablu USB', 95 , 50);

select * from accesorii;

```

Script Output

Task completed in 0.444 seconds

5	FOLIE STICLA IPH	50	100
6	FOLIE STICLA SMG	40	100
7	HUSA FLIP	50	100
8	CASTII WIRELESS	90	20
9	CASTII CU FIR	50	50

7 rows selected.

Error starting at line : 18 in command -

```

insert into accesorii values(11, 'cablu USB', 95 , 50)
Error report -
ORA-20000: Nu se poate insera un accesoriu cu pretul mai mare decat cel maxim!!!
ORA-06512: at "AIIIOAEIS_51.RESTRICTIE_PRET", line 6
ORA-04088: error during execution of trigger 'AIIIOAEIS_51.RESTRICTIE_PRET'

```

2. Sa se creeze un trigger care sa împiedice mărirea prețului cu 50% pentru accesoriile cu stocul mai mic de 100.

```

create or replace trigger verificare_pret
before update of pret on accesorii
for each row
when (new.pret > old.pret)
begin
    if :new.stoc<100 then
        raise_application_error(-20000, 'Nu se poate mari pretul!!!');
    end if;
end;

```

```

end if;
end;
/

```

update accesorii set pret = pret*1.5 where cod_accesoriu =9;

The screenshot shows the SQL Developer interface. On the left, the 'Tables (Filtered)' pane displays a schema with tables like ACCESORII, COD_ACCESORIU, DENUMIRE, PRET, STOC, etc. The 'Reports' pane is also visible. The main 'SQL Worksheet' contains the following SQL code:

```

select * from accesorii;

--2. Sa se creeze un trigger care sa impiedice marirea pretului cu 50% pentru accesoriile cu stocul mai mic de 100.

create or replace trigger verificare_pret
before update of pret on accesorii
for each row
when (new.pret > old.pret)
begin
    if :new.stoc < 100 then
        raise_application_error(-20000, 'Nu se poate mari pretul!!!');
    end if;
end;

update accesorii set pret = pret*1.5 where cod_accesoriu =9;

```

The 'Query Result' pane shows the output of the first query:

5	FOLIE STICLA IPH	50	100
6	FOLIE STICLA SMG	40	100
7	HUSA FLIP	50	100
8	CASTI WIRELESS	90	20
9	CASTI CU FIR	50	50

Below the table, it states '7 rows selected.' The 'Script Output' pane shows an error message:

```

Error starting at line : 37 in command -
update accesorii set pret = pret*1.5 where cod_accesoriu =9
Error report -
ORA-20000: Nu se poate mari pretul!!!
ORA-06512: at "AIIIOAEIS_51.VERIFICARE_PRET", line 3
ORA-04088: error during execution of trigger 'AIIIOAEIS_51.VERIFICARE_PRET'

```

3. Sa se construiască un trigger de tip AFTER care se declanșează in momentul in care se face INSERT, DELETE, sau UPDATE pe coloana data_intrare si durata din tabela REPARARE. Sa se insereze mesaje de la declararea triggerului in tabela nou creata, MODIFICARI.

```

CREATE TABLE MODIFICARI
(MODIF1 VARCHAR2(50),
MODIF2 VARCHAR2(50)
);

```

```

create or replace trigger modificari_facute
after insert or delete or update of cost, durata on reparare
for each row
begin
    if inserting then

```

```

insert into modificari(modif1, modif2) values ('S-a realizat o inserare in tabela REPARARE
', sysdate);
elseif deleting then
insert into modificari (modif1, modif2) values ('S-a realizat o stergere in tabela
REPARARE', sysdate);
elseif updating('cost') then
insert into modificari (modif1, modif2) values('Costul vechi = '||:OLD.cost||' Cost nou =
'||:new.cost, sysdate);
elseif updating('durata') then
insert into modificari (modif1, modif2) values('Durata veche = '||:OLD.durata||' Durata noua
= '||:new.durata, sysdate);
end if;
end;
/

```

```

insert into reparare values (120, 101, to_date('21-02-2020', 'dd-mm-yyyy'), 50, 2, 1);
update reparare set cost=70 where cod_reparare=66;
select * from modificari;

```

The screenshot shows the SQL Developer interface with the following components:

- Connections:** A tree view on the left showing the 'schemaBD' connection.
- Schema Browser:** A tree view on the left showing the 'schemaBD' schema with various tables and views.
- SQL Worksheet:** The main area displaying the SQL script being executed. The script includes a comment, a table creation statement, a trigger definition, and data manipulation statements.
- Script Output:** A panel at the bottom showing the execution results, including a message '1 row updated.' and a table of results.

The SQL script in the worksheet is as follows:

```

--Sa se insereze mesaje de la declararea triggerului in tabela nou creata, MODIFICARI.
CREATE TABLE MODIFICARI
(MODIF1 VARCHAR2(50),
MODIF2 VARCHAR2(50)
);
create or replace trigger modificari_facute
after insert or delete or update of cost, durata on reparare
for each row
begin
if inserting then
insert into modificari(modif1, modif2) values ('S-a realizat o inserare in tabela REPARARE ', sysdate);
elseif deleting then
insert into modificari (modif1, modif2) values ('S-a realizat o stergere in tabela REPARARE', sysdate);
elseif updating('cost') then
insert into modificari (modif1, modif2) values('Costul vechi = '||:OLD.cost||' Cost nou = '||:new.cost, sysdate);
elseif updating('durata') then
insert into modificari (modif1, modif2) values('Durata veche = '||:OLD.durata||' Durata noua = '||:new.durata, sysdate);
end if;
end;
/

insert into reparare values (120, 101, to_date('21-02-2020', 'dd-mm-yyyy'), 50, 2, 1);
update reparare set cost=70 where cod_reparare=66;
select * from modificari;

```

The Script Output panel shows the following results:

```

1 row updated.

MODIF1                                MODIF2
-----
Costul vechi = 50 Cost nou = 50        18-MAY-24
Costul vechi = 50 Cost nou = 70        18-MAY-24
S-a realizat o inserare in tabela REPARARE 18-MAY-24
Costul vechi = 70 Cost nou = 70        18-MAY-24

```


APLICATIA APEX

- Create page MASTER-DETAIL : TELEFON – CLIENT_TEL

SGBD-Proiect

Home

Telefoane

Detalii telefoane

Detalii reparatii

Telefoane

Search: All Text Columns Go Actions Edit Add Row Reset

		Cod telefon	Brand	Model	Defectiune
<input checked="" type="checkbox"/>		100	iPhone	14	mufa de incarcare defecta
<input type="checkbox"/>		102	iPhone	X	difuzor infundat
<input type="checkbox"/>		105	Samsung	Hold	buton deschidere defect
<input type="checkbox"/>		106	iPhone	12	camera frontala defecta
<input type="checkbox"/>		107	Motorola	Razr40	mufa de incarcare defecta
<input type="checkbox"/>		108	Huawei	P11	ecran spart
<input type="checkbox"/>		110	Samsung	Flip 4	ecran spart
<input type="checkbox"/>		104	iPhone	7	mufa de incarcare defecta

1 rows selected 1 - 8 of 8

Search: All Text Columns Go Actions Edit Add Row Reset

		Cod client	Cod telefon	Nume	Prenume	Nr Contact
<input checked="" type="checkbox"/>		1	100	Alicaei	Sorina	757658685

1 rows selected Total 1

App 179/46 Page 2 Session Debug Quick Edit Customize

- Creare raport si formular : Detalii telefoane – Editare telefon

SGBD-Proiect

Home

Telefoane

Detalii telefoane

Telefoane

Search: All Text Columns Go Actions Edit Add Row

		Cod telefon	Brand
<input checked="" type="checkbox"/>		100	iPhone
<input type="checkbox"/>		102	iPhone
<input type="checkbox"/>		105	Samsung
<input type="checkbox"/>		106	iPhone
<input type="checkbox"/>		107	Motorola
<input type="checkbox"/>		108	Huawei
<input type="checkbox"/>		110	Samsung
<input type="checkbox"/>		104	iPhone

1 rows selected

Search: All Text Columns Go Actions Edit Add Row

		Cod client	Cod telefon
<input checked="" type="checkbox"/>		1	100

1 rows selected

App 179/46 Page 3 Session Debug Quick Edit Customize

Detalii telefoane

Home \ Detalii telefoane

Search: All Text Columns Go

Actions

Create

	Brand	Model	Defectiune
<input checked="" type="checkbox"/>	iPhone	13	ecran spart
<input checked="" type="checkbox"/>	iPhone	14	mufa de incarcare defecta
<input checked="" type="checkbox"/>	Samsung	S9	ecran spart
<input checked="" type="checkbox"/>	iPhone	X	difuzor infundat
<input checked="" type="checkbox"/>	Huawei	P8	ecran spart
<input checked="" type="checkbox"/>	iPhone	7	mufa de incarcare defecta
<input checked="" type="checkbox"/>	Samsung	Hold	buton deschidere defect
<input checked="" type="checkbox"/>	iPhone	12	camera frontala defecta
<input checked="" type="checkbox"/>	Motorola	Razr40	mufa de incarcare defecta
<input checked="" type="checkbox"/>	Huawei	P11	ecran spart
<input checked="" type="checkbox"/>	Samsung	Flip 4	ecran spart

1 - 11

- Creare raport si formular : Detalii reparații – Editare

The screenshot shows the 'Detalii reparații' form in the SGBD-Proiect application. The left sidebar contains navigation links: Home, Telefoane, Detalii telefoane, and Detalii reparații. The main content area is titled 'Detalii reparații' and contains a table with columns 'Cod Reparație' and 'Cod Telefon'. The table lists 10 rows of data. Below the table, there is a 'Release 1.0' label and a status bar with various icons and text: 'App 179746', 'Page 6', 'Session', 'Debug', 'Quick Edit', 'Customize', and 'Apply Changes'.

Cod Reparație	Cod Telefon
11	110
22	112
33	102
44	103
55	104
66	105
77	106
88	107
99	108
110	110

The right section of the form is titled 'Detalii reparații' and contains the following fields:

- Cod Reparație: 11
- Cod Telefon: 110
- Data Intrare: 1/2/2024
- Cost: 70
- Durata: 2
- Cod Comp: 5

At the bottom right, there is a blue 'Apply Changes' button.

- Creare validări – raport Detalii Telefoane:

- pentru fiecare câmp necompletat
- codul telefonului trebuie sa fie unic

The screenshot shows the 'Telefoane' table and the 'Editare telefon' form in the SGBD-Proiect application. The left sidebar contains navigation links: Home, Telefoane, and Detalii telefoane. The main content area is titled 'Telefoane' and contains a table with columns 'Cod telefon' and 'Brand'. The table lists 7 rows of data. Below the table, there is a '1 rows selected' label and a status bar with various icons and text: 'App 179746', 'Page 4', 'Session', 'Debug', 'Quick Edit', 'Customize', and 'Create'.

Cod telefon	Brand
100	iPhone
102	iPhone
105	Samsung
106	iPhone
107	Motorola
108	Huawei
110	Samsung
104	iPhone

The right section of the form is titled 'Editare telefon' and contains the following fields:

- Cod telefon: 100 (Error: Codul este deja folosit!)
- Brand: iPhone
- Model: (Error: Completati campul!)
- Defectiune: ecran spart

A yellow error message box is displayed over the form, stating: '2 errors have occurred'.

- * Codul este deja folosit!
- * Completati campul!

At the bottom right, there is a blue 'Create' button.

- Creare validări – raport Detalii reparații :

- costul este cuprins intre 1 si 300
- codul reparații este unic.

SGBD-Proiect

Home

Telefoane

Detalii telefoane

Detalii reparatii

Home \

Detalii reparatii

Q

Go

Actions

	Cod Reparatie	Cod Telefon
	11	110
	22	112
	33	102
	44	103
	55	104
	66	105
	77	106
	88	107
	99	108
	110	110

Release 1.0

Detalii reparatii

Cod Reparatie

11

Cod Telefon

110

Data Intraire

1/2/2024

Cost

7033

Codul de reparare este cuprins intre 1 si 300!

Durata

2

Cod Comp

5

1 error has occurred

Costul este cuprins intre 1 si 300!

Apply Changes

SGBD-Proiect

Home

Telefoane

Detalii telefoane

Detalii reparatii

Home \

Detalii reparatii

Q

Go

Actions

	Cod Reparatie	Cod Telefon
	11	110
	22	112
	33	102
	44	103
	55	104
	66	105
	77	106
	88	107
	99	108
	110	110

Release 1.0

Detalii reparatii

Cod Reparatie

11

Cod Telefon

Data Intraire

5/01/2024

Cost

100

Durata

1

Cod Comp

2

1 error has occurred

Codul de reparare este deja folosit!

Create

--acțiuni dinamice: la o durata mai mare de 5 zile se aplica o reducere de 20%(dublu click pe reducere)

SGBD-Proiect

Home

Telefoane

Detalii telefoane

Detalii reparatii

Home \

Detalii reparatii

Q

Go

Actions

	Cod Reparatie	Cod Telefon
	11	110
	22	112
	33	102
	44	103
	55	104
	66	105
	77	106
	88	107
	99	108
	110	110

Release 1.0

Editare reparatii

Cod Reparatie

11

Cod Telefon

110

Data Intrare

1/2/2024

Cost

70

Durata

8

Cod Comp

5

Reducere

14

Apply Changes