Universitatea “Politehnica” din Timişoara

Facultatea de Automatică şi Calculatoare

Secţia Ingineria Sistemelor

Proiect la BAZE DE DATE

Proiectul nr. 5

Brînzan Ionuț-Alexandru

An II gr 1.2

**Tema proiectului**

Se consideră o aplicaţie pentru evidenţa parcului auto.

Baza de date Oracle va conţine următoarele informaţii:

* Număr înmatriculare
* Marca
* An fabricaţie
* Carburant
* Data alimentării
* Kilometraj – când s-a facut alimentarea
* Şoferul care a făcut alimentarea

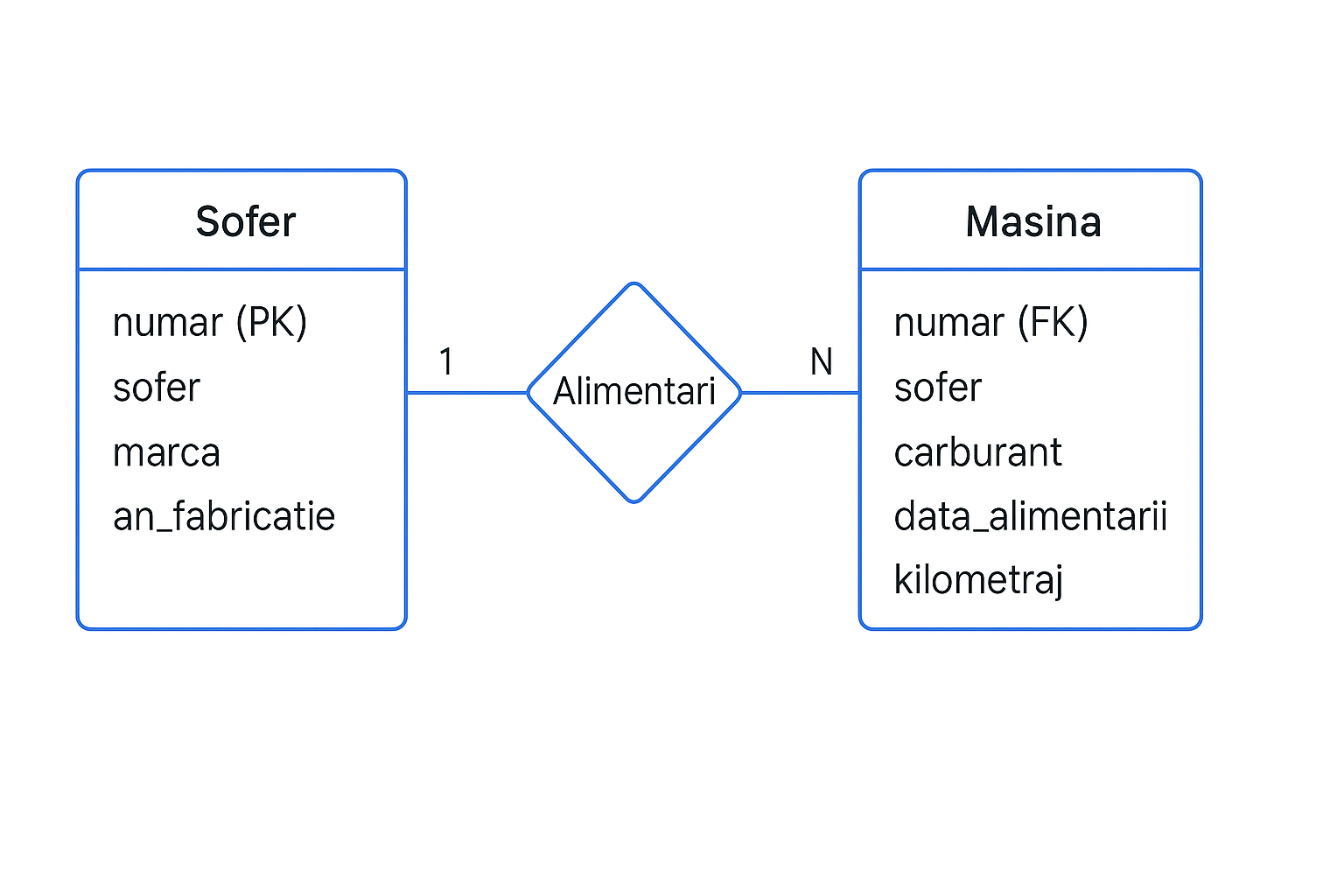
Ştiind că ‘Numărul’ are exact 7 caractere, ‘Marca’ nu depaşeşte 10 caractere, anul de fabricaţie este cuprins între 2005 şi 2021, carburant este numărul de litri de combustibil de la alimentare (nu poate fi mai mult de 60), kilometraj este un număr întreg, care conţine numărul de km efectuaţi de la ultima realimentare, şi şoferul este de maxim 10 caractere, şi că o maşină poate fi realimentată de oricâte ori, dar numai umplând la maxim rezervorul, se cere:

1. Să se realizeze proiectarea bazei de date aferente (structura de tabele, structura de coloane a fiecărei tabele, constrângeri);
2. Să se scrie comenzile SQL pentru tabelele proiectate la punctul anterior;
3. Să se scrie comenzile SQL pentru popularea bazei de date cu următoarele produse:

(aceste informatii vor fi introduse corespunzator in tabelele create):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Numar | Marca | An fabricatie | Carburant | Data alimentare | Kilometraj | Sofer |
| TMXXYY1 | Logan | 2007 | 40 | 10/01/2007 | 100 | Duma A. |
| TMXXYY2 | Renault | 2006 | 34 | 01/02/2003 | 50 | Ion M. |
| TMXXYY3 | Aro | 2008 | 35 | 07/12/2005 | 150 | Pop A. |
| TMXXYY4 | Cielo | 2020 | 55 | 11/12/2005 | 230 | Popa M. |
| TMXXYY5 | Matiz | 2005 | 34 | 14/05/2006 | 89 | Marin O. |
| TMXXYY1 | Logan | 2007 | 40 | 10/02/2007 | 180 | Duma A |

1. Să se scrie o procedură care să permită alimentarea unei maşini - procedura va fi apelată cu parametrii: număr, carburant, data, kilometraj, şofer.
2. Să se genereze un raport care să cuprindă numărul, anul fabricaţiei, şi consumul mediu al tuturor maşinilor din parc.
3. Să se genereze un raport detaliat care să cuprindă numărul, anul fabricaţiei, data alimentării, şoferul şi consumul, ordonat dupa şofer, consum, număr, în sens crescător şi data alimentării în sens descrescator.
4. Să se scrie un trigger care să nu permită adăugarea informaţiilor despre alimentări dacă consumul mediu rezultat este de 3 ori mai mic decât consumul mediu total pe maşina respectivă şi să semnaleze eroare.
5. Să se scrie o funcţie care să primească ca şi parametri şoferul, şi să returneze consumul mediu.
6. Să se afişeze primele trei maşini cu cele mai multe alimentări precizând: număr, an fabricaţie, marca, carburant, data alimentării.
7. Să se afişeze şoferul care are cel mai mare consum general, precizând şofer, număr, consum şi ponderea kilometrajului cu respectiva maşina în numărul total de km parcurşi de respectivul.

****

**Rezolvare cerinţe**

1. **Să se realizeze proiectarea bazei de date aferente (structura de tabele, structura de coloane a fiecarei tabele, constrangeri)**
2. **Să se scrie comenzile SQL pentru tabelele proiectate la punctul anterior**

Rezolvare:

*CREATE TABLE sofer (sofer VARCHAR(10) ,*

*numar VARCHAR(7) CHECK(length(numar)=7) PRIMARY KEY,*

*marca VARCHAR(10) ,*

*an\_fabricatie INTEGER CHECK(an\_fabricatie >=2005 AND an\_fabricatie <= 2021));*

*CREATE TABLE masina (sofer VARCHAR(10) ,*

*numar VARCHAR(7) CHECK(length(numar)=7) ,*

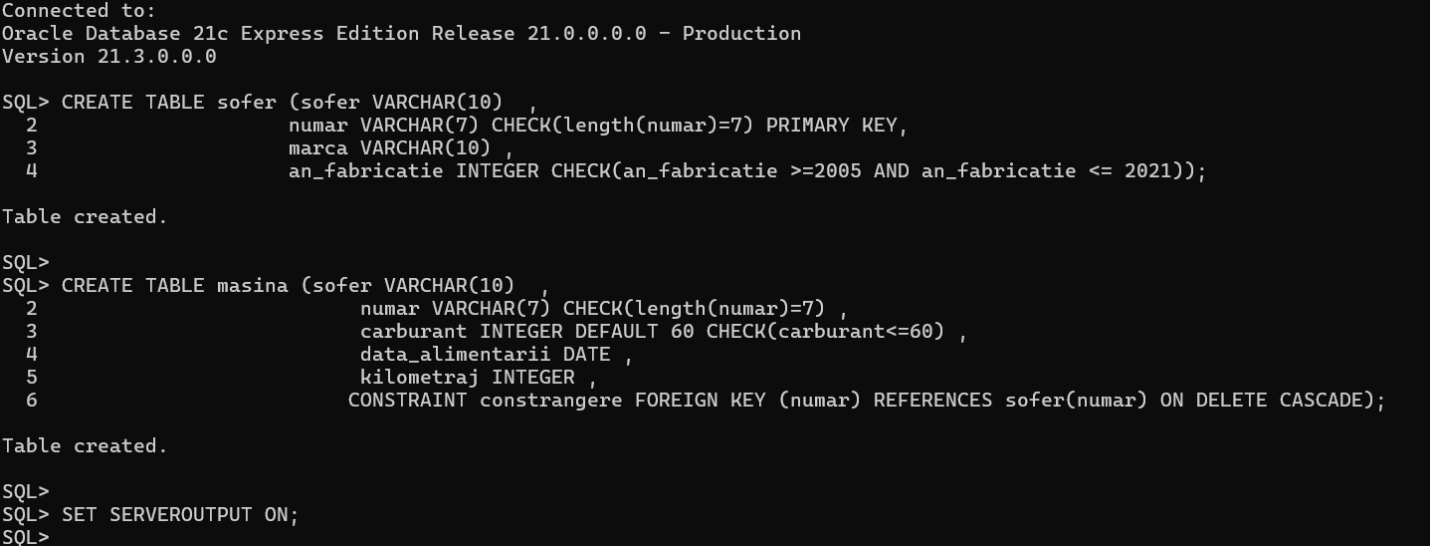
*carburant INTEGER DEFAULT 60 CHECK(carburant<=60) ,*

*data\_alimentarii DATE ,*

*kilometraj INTEGER ,*

*CONSTRAINT constrangere FOREIGN KEY (numar) REFERENCES sofer(numar) ON DELETE CASCADE);*

*SET SERVEROUTPUT ON;*

**

1. **Să se scrie comenzile SQL pentru popularea bazei de date cu următoarele produse:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Numar | Marca | An fabricatie | Carburant | Data alimentare | Kilometraj | Sofer |
| TMXXYY1 | Logan | 2007 | 40 | 10/01/2007 | 100 | Duma A. |
| TMXXYY2 | Renault | 2006 | 34 | 01/02/2003 | 50 | Ion M. |
| TMXXYY3 | Aro | 2008 | 35 | 07/12/2005 | 150 | Pop A. |
| TMXXYY4 | Cielo | 2020 | 55 | 11/12/2005 | 230 | Popa M. |
| TMXXYY5 | Matiz | 2005 | 34 | 14/05/2006 | 89 | Marin O. |
| TMXXYY1 | Logan | 2007 | 40 | 10/02/2007 | 180 | Duma A |

Rezolvare:

*INSERT INTO sofer VALUES('Duma A.' , 'TMXXYY1' , 'Logan' , 2007);*

*INSERT INTO sofer VALUES('Ion M.' , 'TMXXYY2' , 'Renault' , 2006);*

*INSERT INTO sofer VALUES('Pop A.' , 'TMXXYY3' , 'Aro' , 2008);*

*INSERT INTO sofer VALUES('Popa M.' , 'TMXXYY4' , 'Cielo' , 2020);*

*INSERT INTO sofer VALUES('Marin O.' , 'TMXXYY5' , 'Matiz' , 2005);*

*INSERT INTO masina VALUES('Duma A.' , 'TMXXYY1' , 40 , '10-Jan-2007' , 100);*

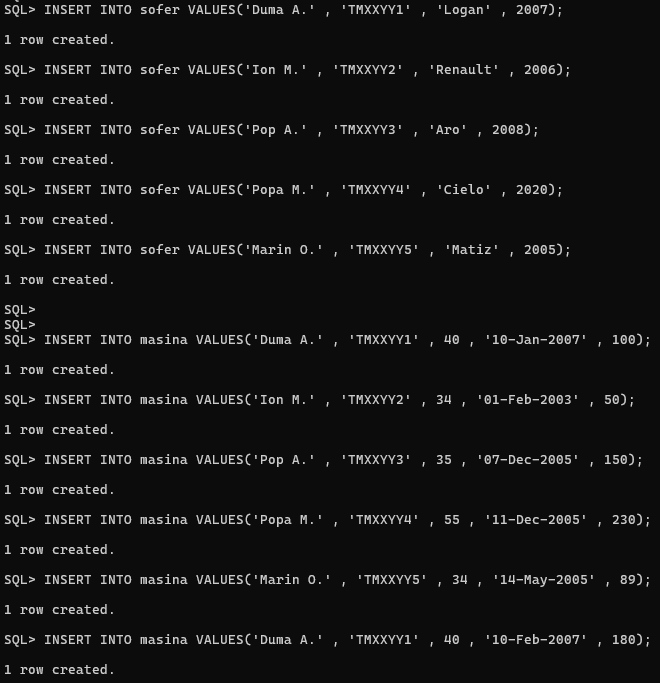
*INSERT INTO masina VALUES('Ion M.' , 'TMXXYY2' , 34 , '01-Feb-2003' , 50);*

*INSERT INTO masina VALUES('Pop A.' , 'TMXXYY3' , 35 , '07-Dec-2005' , 150);*

*INSERT INTO masina VALUES('Popa M.' , 'TMXXYY4' , 55 , '11-Dec-2005' , 230);*

*INSERT INTO masina VALUES('Marin O.' , 'TMXXYY5' , 34 , '14-May-2005' , 89);*

*INSERT INTO masina VALUES('Duma A.' , 'TMXXYY1' , 40 , '10-Feb-2007' , 180);*

**

1. **Să se scrie o procedură care să permită alimentarea unei maşini- procedura va fi apelată cu parametrii: număr, carburant, data, kilometraj, şofer.**

Rezolvare:

*CREATE OR REPLACE PROCEDURE procedura1(param1 VARCHAR , param2 INTEGER , param3 DATE , param4 INTEGER , param5 VARCHAR)*

*AS*

*x INTEGER;*

*y INTEGER;*

*z INTEGER;*

*BEGIN*

*SELECT COUNT(sofer) INTO x FROM sofer WHERE sofer.sofer=param5;*

*SELECT COUNT(numar) INTO y FROM sofer WHERE sofer.numar=param1;*

*IF length(param1) <> 7 THEN*

*z:=0;*

*ELSE*

*z:=1;*

*END IF;*

*IF x=0 OR y=0 OR z=0 THEN*

*IF x=0 THEN*

*DBMS\_OUTPUT.PUT\_LINE('nu se poate alimenta , soferul nu este in baza de date !');*

*END IF;*

*IF y=0 THEN*

*DBMS\_OUTPUT.PUT\_LINE('nu se poate alimenta , masina nu este in baza de date !');*

*END IF;*

*IF z=0 THEN*

*DBMS\_OUTPUT.PUT\_LINE('nu se poate alimenta , nr de inmatriculare nu este valid!');*

*END IF;*

*ELSE*

*IF param2 <= 60 THEN*

*INSERT INTO masina VALUES(param5 , param1 , param2 , param3 , param4);*

*ELSE DBMS\_OUTPUT.PUT\_LINE('depasire capacitate rezervor!');*

*END IF;*

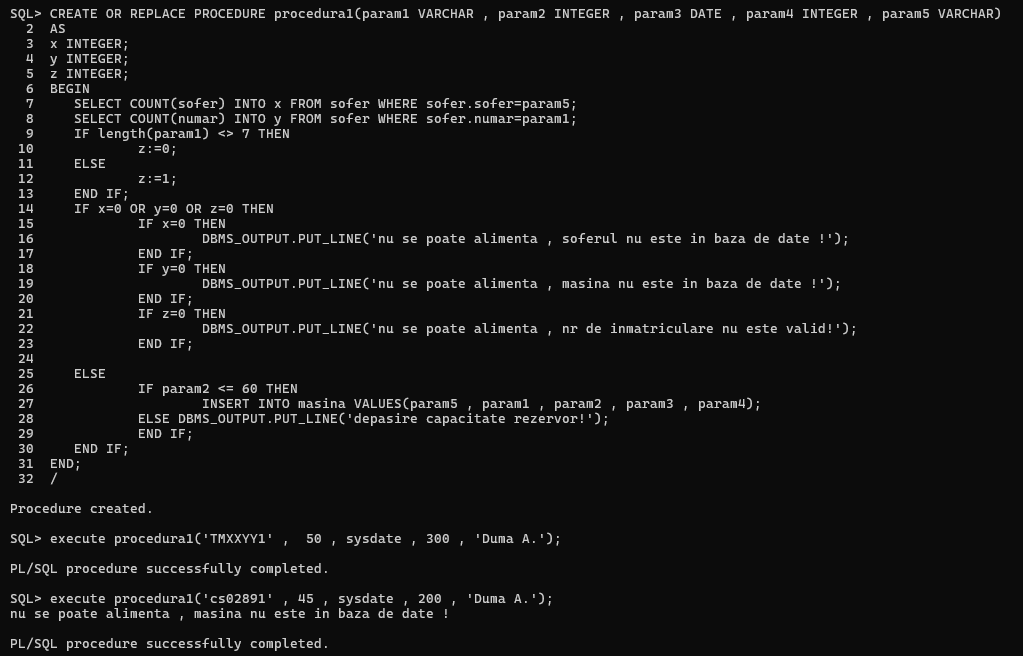
*END IF;*

*END;*

*/*

*execute procedura1('TMXXYY1' , 50 , sysdate , 300 , 'Duma A.');*

*execute procedura1('cs02891' , 45 , sysdate , 200 , 'Duma A.');*



1. **Să se genereze un raport care să cuprindă numărul, anul fabricaţiei, şi consumul mediu al tuturor maşinilor din parc.**

Rezolvare:

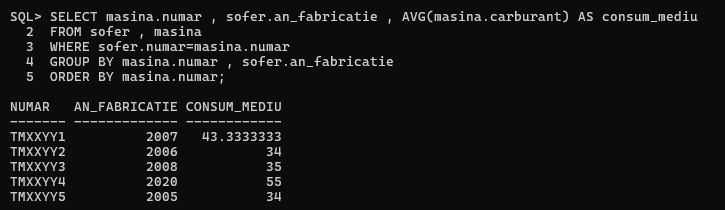
*SELECT masina.numar , sofer.an\_fabricatie , AVG(masina.carburant) AS consum\_mediu*

*FROM sofer , masina*

*WHERE sofer.numar=masina.numar*

*GROUP BY masina.numar , sofer.an\_fabricatie*

*ORDER BY masina.numar;*

**

1. **Să se genereze un raport detaliat care să cuprindă numărul, anul fabricaţiei, data alimentării, şoferul şi consumul, ordonat dupa şofer, consum, număr, în sens crescător şi data alimentării în sens descrescator.**

Rezolvare:

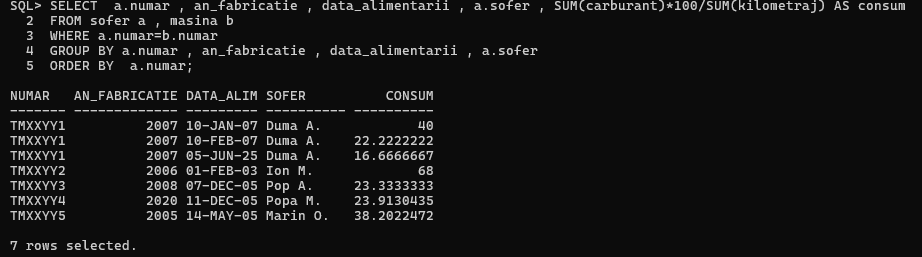
*SELECT a.numar , an\_fabricatie , data\_alimentarii , a.sofer , SUM(carburant)\*100/SUM(kilometraj) AS consum*

*FROM sofer a , masina b*

*WHERE a.numar=b.numar*

*GROUP BY a.numar , an\_fabricatie , data\_alimentarii , a.sofer*

*ORDER BY a.numar;*

**

1. **Să se scrie un trigger care să nu permită adăugarea informaţiilor despre alimentări dacă consumul mediu rezultat este de 3 ori mai mic decăt consumul mediu total pe maşina respectivă şi să semnaleze eroare.**

*Rezolvare:*

*CREATE OR REPLACE TRIGGER triger*

*BEFORE INSERT ON masina*

*FOR EACH ROW*

*DECLARE*

*x INTEGER;*

*y REAL;*

*z REAL;*

*aux1 INTEGER;*

*aux2 INTEGER;*

*consum\_mediu REAL;*

*consum\_mediu\_nou REAL;*

*BEGIN*

*SELECT COUNT(numar) INTO x FROM masina WHERE numar=:NEW.numar;*

*SELECT SUM(carburant) INTO y FROM masina WHERE numar=:NEW.numar;*

*SELECT SUM(kilometraj) INTO z FROM masina WHERE numar=:NEW.numar;*

*aux1:=:NEW.carburant;*

*aux2:=:NEW.kilometraj;*

*IF x<>0 THEN*

*consum\_mediu:= y\*100/(z\*x);*

*consum\_mediu\_nou := 3\*(aux1+y)\*100/((z+aux2)\*(x+1));*

*IF consum\_mediu\_nou<consum\_mediu THEN*

*RAISE\_APPLICATION\_ERROR(-20000 , 'NU S-A ALIMENTAT , CONSUM MEDIU PREA MIC');*

*END IF;*

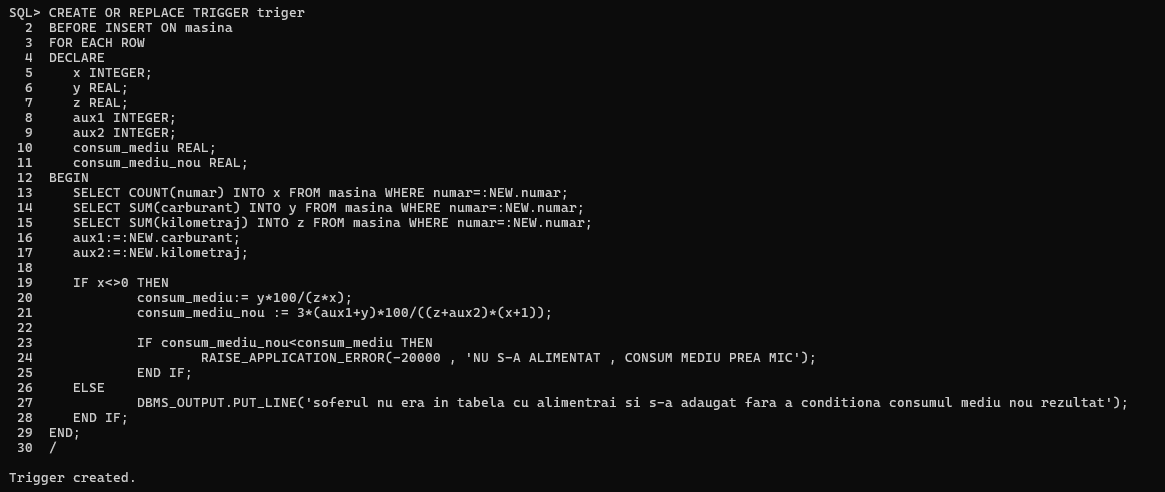
*ELSE*

*DBMS\_OUTPUT.PUT\_LINE('soferul nu era in tabela cu alimentrai si s-a adaugat fara a conditiona consumul mediu nou rezultat');*

*END IF;*

*END;*

*/*

**

1. **Să se scrie o funcţie care să primească ca şi parametri şofer, şi să returneze consumul mediu.**

Rezolvare:

*CREATE OR REPLACE FUNCTION functie (param VARCHAR)*

*RETURN NUMBER*

*AS*

*x INTEGER;*

*y INTEGER;*

*z INTEGER;*

*consum NUMBER(4,2);*

*BEGIN*

*SELECT COUNT(sofer) INTO x FROM masina WHERE sofer=param;*

*SELECT SUM(carburant) INTO y FROM masina WHERE sofer=param;*

*SELECT SUM(kilometraj) INTO z FROM masina WHERE sofer=param;*

*IF x<>0 THEN*

*consum:=(y\*100)/(z\*x);*

*ELSE*

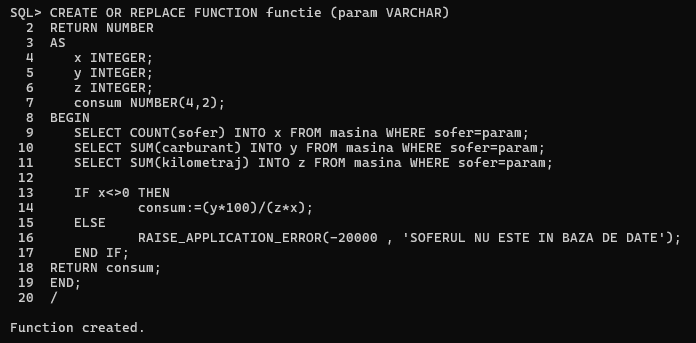
*RAISE\_APPLICATION\_ERROR(-20000 , 'SOFERUL NU ESTE IN BAZA DE DATE');*

*END IF;*

*RETURN consum;*

*END;*

*/*

**

1. **Să se afişeze primele trei maşini cu cele mai multe alimentări, în 2 ani consecutivi precizând: număr, an fabricaţie, marcă, carburant, data alimentării.**

Rezolvare:

*CREATE VIEW veder9 AS*

*(SELECT numar , data\_alimentarii FROM masina GROUP BY numar , data\_alimentarii);*

*SELECT a.numar , an\_fabricatie , marca , carburant , b.data\_alimentarii*

*FROM sofer a , masina b , veder9 c*

*WHERE a.numar=b.numar AND b.numar=c.numar AND a.numar=c.numar*

*GROUP BY a.numar , an\_fabricatie , marca , carburant , b.data\_alimentarii*

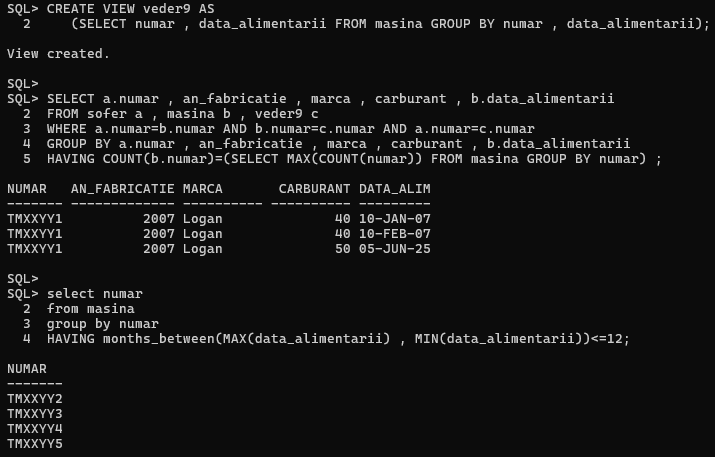
*HAVING COUNT(b.numar)=(SELECT MAX(COUNT(numar)) FROM masina GROUP BY numar) ;*

*select numar*

*from masina*

*group by numar*

*HAVING months\_between(MAX(data\_alimentarii) , MIN(data\_alimentarii))<=12;*

****

1. **Să se afişeze şoferul care are cel mai mare consum general, precizând şofer, număr, consum şi ponderea kilometrajului cu respectiva maşina în numărul total de km parcurşi.**

Rezolvare:

*CREATE VIEW veder1 AS*

*(SELECT sofer , numar , SUM(carburant)\*100/SUM(kilometraj) AS consum\_masina FROM masina GROUP BY sofer , numar);*

*CREATE VIEW veder2 AS*

*(SELECT sofer , numar , SUM(kilometraj) AS kilometraj*

*FROM masina*

*GROUP BY sofer , numar);*

*SELECT a.sofer*

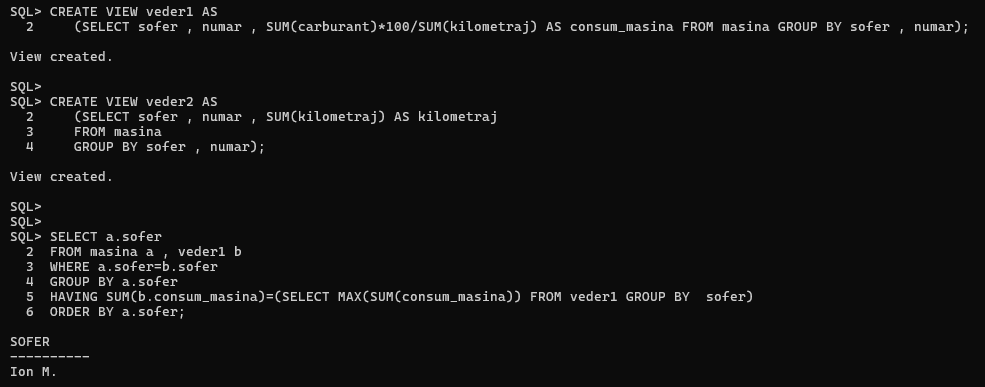
*FROM masina a , veder1 b*

*WHERE a.sofer=b.sofer*

*GROUP BY a.sofer*

*HAVING SUM(b.consum\_masina)=(SELECT MAX(SUM(consum\_masina)) FROM veder1 GROUP BY sofer)*

*ORDER BY a.sofer;*

**