

Proiect SGBD
Gestiune Magazin Online
Măciucă Emma-Iulia
Seria 24
Grupa 241
An Universitar 2024-2025

Cuprins

1. Prezentați pe scurt baza de date (utilitatea ei).....	4
2. Realizați diagrama entitate-relație (ERD): entitățile, relațiile și attributele trebuie definite în limba română (vezi curs SGBD, model de diagrama entitate-relație; nu se va accepta alt format).....	6
3. Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate attributele necesare: entitățile, relațiile și attributele trebuie definite în limba română	7
4. Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, adăugând toate constrângerile de integritate necesare (chei primare, cheile externe etc).....	8
5. Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru fiecare tabelă asociativă).....	27
6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul.....	46
7. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor . Apelați subprogramul.....	50
8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele create. Tratați toate excepțiile care pot apărea, inclusiv excepțiile predefinite NO_DATA_FOUND și TOO_MANY_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.....	54
9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să aibă minim 2 parametri și să utilizeze într-o singură comandă SQL 5 dintre tabelele create. Definiți minim 2 excepții proprii, altele decât cele predefinite la nivel de sistem. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate.....	60
10. Definiți un <i>trigger</i> de tip LMD la nivel de comandă. Declanșați <i>trigger-ul</i>	73
11. Definiți un <i>trigger</i> de tip LMD la nivel de linie. Declanșați <i>trigger-ul</i>	77
12. Definiți un <i>trigger</i> de tip LDD. Declanșați <i>trigger-ul</i>	83
13. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).....	86

1. Prezentați pe scurt baza de date (utilitatea ei).

Tema

Tema proiectului este o bază de date realizată pentru o aplicație de gestiune a unui magazin online, cu scopul de a facilita interacțiunea utilizatorilor și a angajaților cu funcționalitățile magazinului online.

Descriere

Baza de date este folositoare pentru a monitoriza și stoca detalii atât despre produsele disponibile, cât și despre utilizatori și istoricul de comenzi plasate în cadrul magazinului, precum și informații privind livrarea și plata.

Entitatea Utilizator stochează informații importante despre utilizatorii aplicatiei, precum date de contact și de autentificare. Entitatea Comanda urmărește comenzi plasate de utilizatori, iar entitatile Livrare și Plata retin informații despre metodele de livrare și plata ale unei comenzi. De asemenea, entitățile Curier și Masina_Livrare rețin informații suplimentare despre livrari. Entitatea Card stochează datele cardului cu care se efectuează plata, iar Adresa adresa unde se livrează comanda și se facturează chitanta de plată. Entitatile Cos_Cumpărături și Lista_Dorinte reprezintă funcționalități universale unui magazin online, unde utilizatorii pot retine produsele preferate. În final, entitatile Produs și Categorie stochează informații despre produsele disponibile în magazin.

Utilitate

Utilitatea acestei baze de date este de a îmbunătăți gestionarea stocului de produse din magazin și a comenzi plasate de către administratori, dar și pentru a ajuta utilizatorii înregistrați să își urmărească istoricul de comenzi plasate, datele personale vizibile din contul de utilizator, produsele favorite din lista de dorințe, dar și produsele adăugate în coșul de cumpărături.

Informațiile stocate în baza de date sunt folositoare atât pentru a asigura acces rapid și eficient la informații și a monitoriza performanța vânzărilor, cât și pentru personalizarea experienței utilizatorilor și menținerea securității și confidențialității datelor.

Functionalități

Funcționalitățile bazei de date includ gestiunea produselor, adăugarea, actualizarea și ștergerea informațiilor despre produse și categorii și a utilizatorilor, administrarea datelor personale ale utilizatorilor, inclusiv înregistrarea și autentificarea utilizatorilor.

De asemenea, aplicația permite monitorizarea comenzi plasate, inclusiv stările comenzi și detaliile aferente și administrarea informațiilor despre metodele de plată și livrare, inclusiv detalii despre curieri și vehicule de livrare.

In plus, coșul de cumpărături și lista de dorințe permite utilizatorilor să adauge și să gestioneze produsele din coșul de cumpărături și lista de dorințe.

Concluzie

În concluzie, baza de date proiectată pentru gestionarea unui magazin online facilitează optimizarea operațiunilor magazinului și îmbunătățirea experienței utilizatorilor. Prin stocarea și gestionarea eficientă a datelor despre produse, utilizatori, comenzi, plăți și livrări, baza de date asigură o funcționare optimă a magazinului online și permite luarea de decizii informate bazate pe date precise și actualizate.

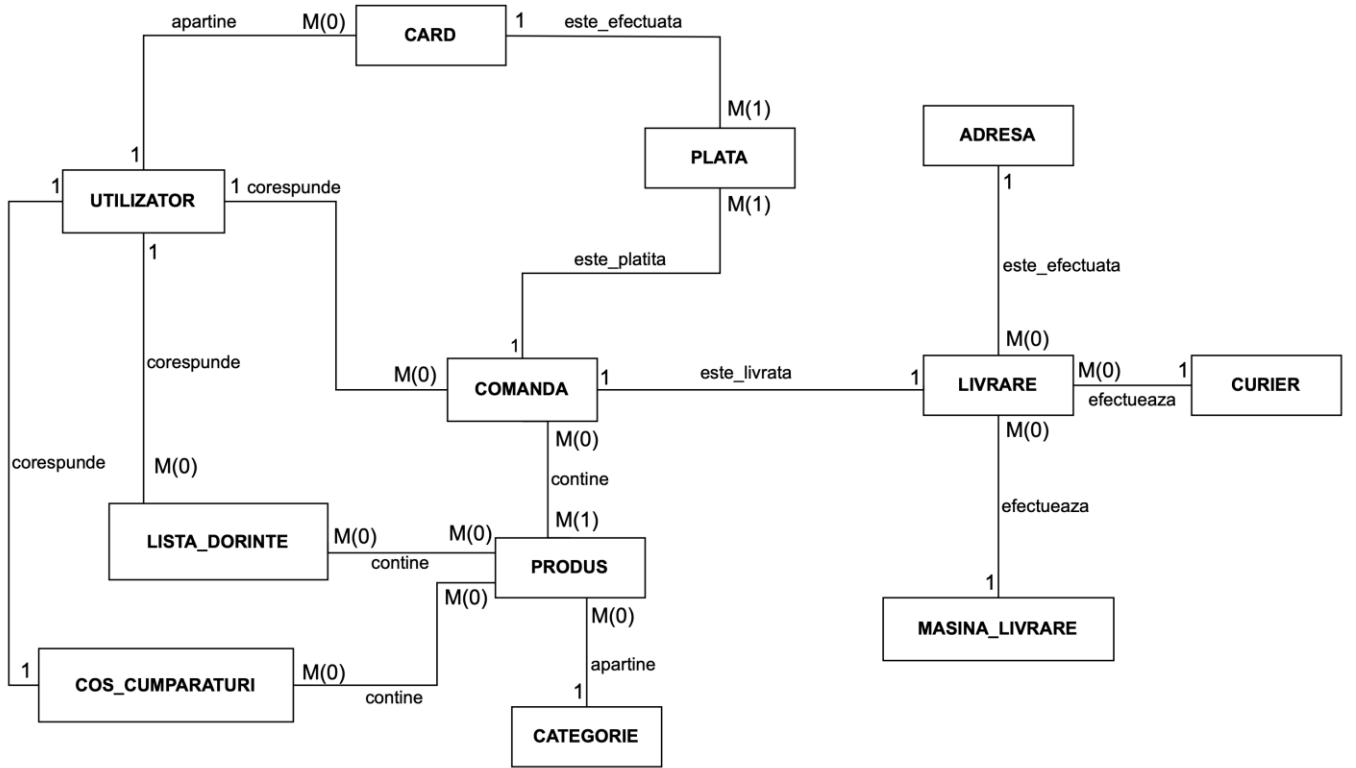
Infrastructura

Versiunea SGBD-ului este Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production Version 21.3.0.0.0.

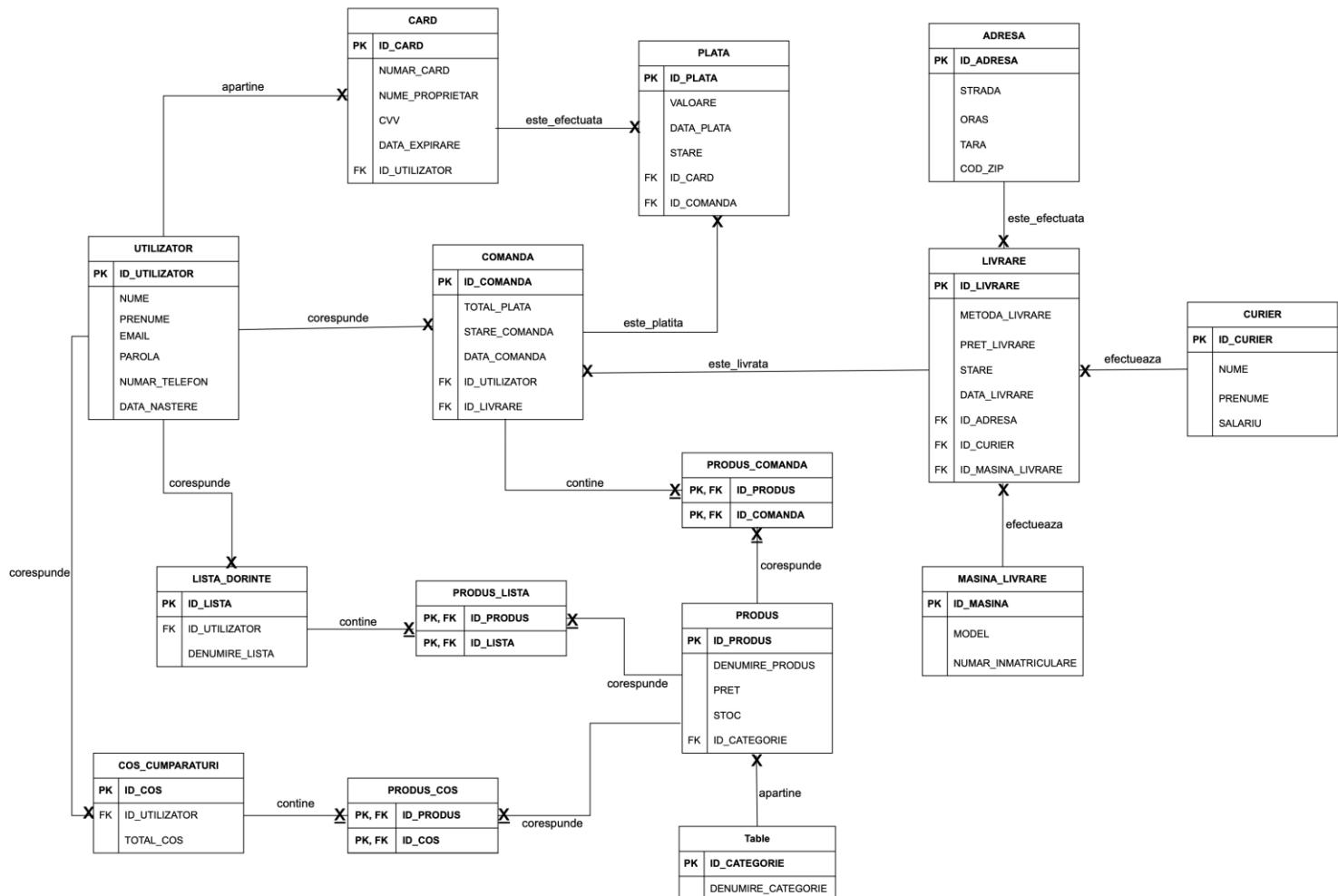
Pentru rezolvarea proiectului am folosit o mașină virtuală de Windows 11 instalată în platforma UTM pe un laptop ce folosește sistemul de operare macOS, unde am utilizat SQL Developer.

Mașina virtuală are alocată 4GB de memorie RAM.

2. Realizați diagrama entitate-relație (ERD): entitățile, relațiile și atributele trebuie definite în limba română (vezi curs SGBD, model de diagrama entitate-relație; nu se va accepta alt format).



3. Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate attributele necesare: entitățile, relațiile și attributele trebuie definite în limba română.



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

CREATE SEQUENCE UTILIZATOR_SEQ START WITH 1;

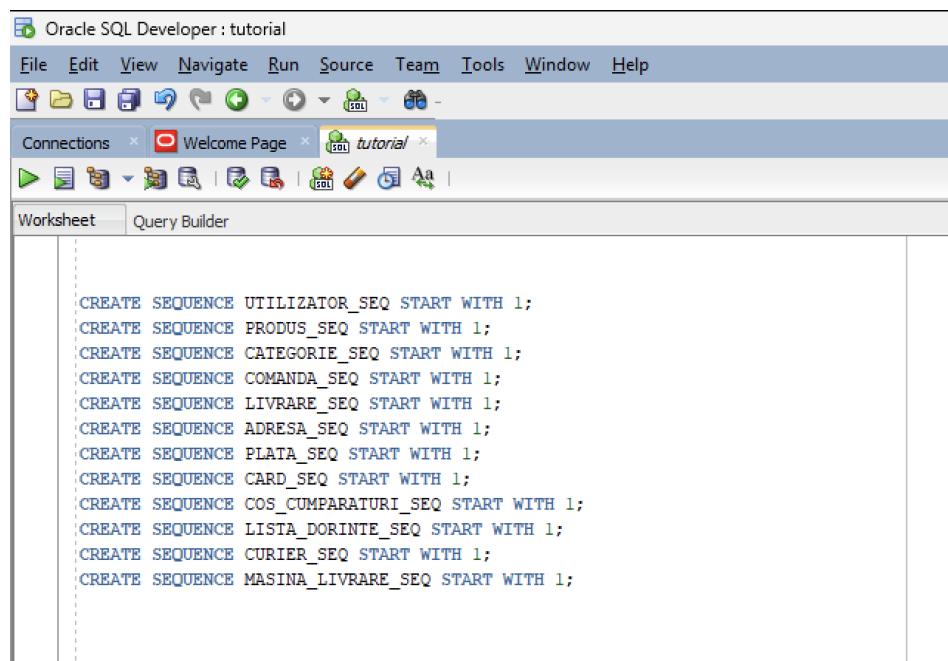
CREATE SEQUENCE PRODUS_SEQ START WITH 1;

CREATE SEQUENCE CATEGORIE_SEQ START WITH 1;

CREATE SEQUENCE COMANDA_SEQ START WITH 1;

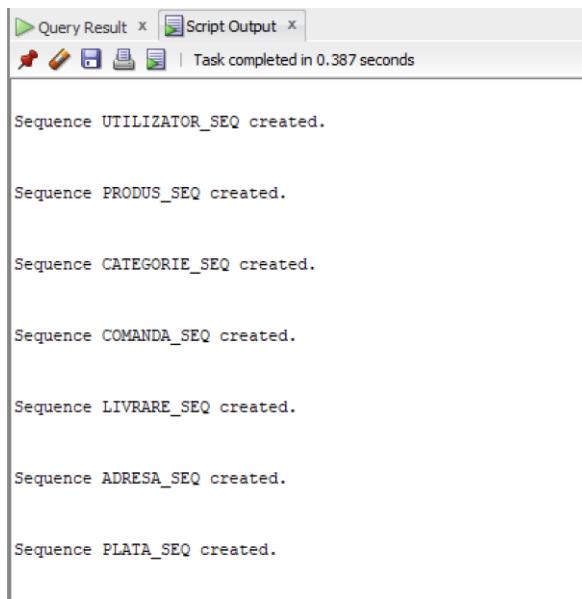
CREATE SEQUENCE LIVRARE_SEQ START WITH 1;

```
CREATE SEQUENCE ADRESA_SEQ START WITH 1;  
CREATE SEQUENCE PLATA_SEQ START WITH 1;  
CREATE SEQUENCE CARD_SEQ START WITH 1;  
CREATE SEQUENCE COS_CUMPARATURI_SEQ START WITH 1;  
CREATE SEQUENCE LISTA_DORINTE_SEQ START WITH 1;  
CREATE SEQUENCE CURIER_SEQ START WITH 1;  
CREATE SEQUENCE MASINA_LIVRARE_SEQ START WITH 1;
```



The screenshot shows the Oracle SQL Developer application window. The title bar reads "Oracle SQL Developer : tutorial". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar below the menu bar contains various icons for database management tasks. The main workspace has tabs for "Connections", "Welcome Page", and "tutorial". The "tutorial" tab is active and displays the sequence creation code. Below the tabs is another toolbar with icons for running queries, saving, and other functions. The bottom of the window shows two tabs: "Worksheet" (which is selected) and "Query Builder". The "Worksheet" tab contains the following SQL code:

```
CREATE SEQUENCE UTILIZATOR_SEQ START WITH 1;  
CREATE SEQUENCE PRODUS_SEQ START WITH 1;  
CREATE SEQUENCE CATEGORIE_SEQ START WITH 1;  
CREATE SEQUENCE COMANDA_SEQ START WITH 1;  
CREATE SEQUENCE LIVRARE_SEQ START WITH 1;  
CREATE SEQUENCE ADRESA_SEQ START WITH 1;  
CREATE SEQUENCE PLATA_SEQ START WITH 1;  
CREATE SEQUENCE CARD_SEQ START WITH 1;  
CREATE SEQUENCE COS_CUMPARATURI_SEQ START WITH 1;  
CREATE SEQUENCE LISTA_DORINTE_SEQ START WITH 1;  
CREATE SEQUENCE CURIER_SEQ START WITH 1;  
CREATE SEQUENCE MASINA_LIVRARE_SEQ START WITH 1;
```



The screenshot shows the Oracle SQL Developer interface with two tabs: "Query Result" and "Script Output". The "Query Result" tab is active and displays the following text:

```
Sequence UTILIZATOR_SEQ created.  
Sequence PRODUS_SEQ created.  
Sequence CATEGORIE_SEQ created.  
Sequence COMANDA_SEQ created.  
Sequence LIVRARE_SEQ created.  
Sequence ADRESA_SEQ created.  
Sequence PLATA_SEQ created.
```

A status bar at the bottom indicates "Task completed in 0.387 seconds".

Utilizator

```
CREATE TABLE UTILIZATOR (  
    ID_UTILIZATOR INT DEFAULT UTILIZATOR_SEQ.NEXTVAL,  
    NUME VARCHAR(255),  
    PRENUME VARCHAR(255),  
    EMAIL VARCHAR(255) NOT NULL,  
    PAROLA VARCHAR(255) NOT NULL,  
    NUMAR_TELEFON VARCHAR(10) CONSTRAINT VERIFICARE_TELEFON CHECK  
        ( LENGTH(NUMAR_TELEFON) = 10 ),  
    DATA_NASTERE DATE,  
    CONSTRAINT CHEIE_PRIMARA_UTILIZATOR PRIMARY KEY (ID_UTILIZATOR),  
    CONSTRAINT EMAIL_UNIC UNIQUE (EMAIL)  
);
```

Oracle SQL Developer : Table UTILIZATORUTILIZATOR@tutorial

File Edit View Navigate Run Team Tools Window Help

tutorial Connections UTILIZATOR

Columns Data Model Constraints Grants Statistics Triggers Flashback Dependencies Details Partitions Indexes SQL Actions...

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID_UTILIZATOR	NUMBER (38,0)	No	"UTILIZATOR"."UTILIZATOR_SEQ"."NEXTVAL"	1 (null)	
2 NUME	VARCHAR2 (255 BYTE)	Yes	(null)	2 (null)	
3 PRENUME	VARCHAR2 (255 BYTE)	Yes	(null)	3 (null)	
4 EMAIL	VARCHAR2 (255 BYTE)	No	(null)	4 (null)	
5 PAROLA	VARCHAR2 (255 BYTE)	No	(null)	5 (null)	
6 NUMAR_TELEFON	VARCHAR2(10 BYTE)	Yes	(null)	6 (null)	
7 DATA_NASTERE	DATE	Yes	(null)	7 (null)	

Oracle SQL Developer : tutorial

File Edit View Navigate Run Source Team Window Help

Connections Welcome Page tutorial

Worksheet Query Builder

```

CREATE TABLE UTILIZATOR (
    ID_UTILIZATOR INT DEFAULT UTILIZATOR_SEQ.NEXTVAL,
    NUME VARCHAR2(255),
    PRENUME VARCHAR2(255),
    EMAIL VARCHAR(255) NOT NULL,
    PAROLA VARCHAR(255) NOT NULL,
    NUMAR_TELEFON VARCHAR(10) CONSTRAINT VERIFICARE_TELEFON CHECK ( LENGTH(NUMAR_TELEFON) = 10),
    DATA_NASTERE DATE,
    CONSTRAINT CKEK_PRIMARA_UTILIZATOR PRIMARY KEY (ID_UTILIZATOR),
    CONSTRAINT EMAIL_UNIC UNIQUE (EMAIL)
);

INSERT INTO UTILIZATOR (NUME, PRENUME, EMAIL, PAROLA, NUMAR_TELEFON, DATA_NASTERE) VALUES
('Măruță', 'Emre', 'emreseaciuus@gmail.com', 'parola024', '0712345678', '01-PEL-2004');

INSERT INTO UTILIZATOR (NUME, PRENUME, EMAIL, PAROLA, NUMAR_TELEFON, DATA_NASTERE) VALUES
('Popescu', 'Andreea-Ioana', 'popescuandreea@gmail.com', 'abcparola1', '0757823352', '14-APR-2000');

INSERT INTO UTILIZATOR (NUME, PRENUME, EMAIL, PAROLA, NUMAR_TELEFON, DATA_NASTERE) VALUES
('Ion', 'Andrei', 'ionandrei@gmail.com', 'contparolapapa', '0717053379', '20-MAR-1999');

INSERT INTO UTILIZATOR (NUME, PRENUME, EMAIL, PAROLA, NUMAR_TELEFON, DATA_NASTERE) VALUES
('Andreecu', 'Maria', 'mariaandr@gmail.com', '123456789', '0720983487', '09-MAR-2001');

```

Script Output X | Query Result X

Task completed in 0.568 seconds

Sequence UTILIZATOR_SEQ created.

Table UTILIZATOR created.

1 row inserted.

Line 188 Column 1 | Insert | Modified | Windows

Oracle SQL Developer : Table UTILIZATORUTILIZATOR@tutorial

File Edit View Navigate Run Team Tools Window Help

tutorial > Connections > UTILIZATOR >

Columns Data Model Constraints Grants Statistics Triggers Flashback Dependencies Details Partitions Indexes SQL

Actions...

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID_UTILIZATOR	NUMBER(38,0)	No	"UTILIZATOR"."UTILIZATOR_SEQ"."NEXTVAL"	1	(null)
2 NUME	VARCHAR2(255 BYTE)	Yes	(null)	2	(null)
3 PRENUME	VARCHAR2(255 BYTE)	Yes	(null)	3	(null)
4 EMAIL	VARCHAR2(255 BYTE)	No	(null)	4	(null)
5 PAROLA	VARCHAR2(255 BYTE)	No	(null)	5	(null)
6 NUMAR_TELEFON	VARCHAR2(10 BYTE)	Yes	(null)	6	(null)
7 DATA_NASTERE	DATE	Yes	(null)	7	(null)

Adresa

```

CREATE TABLE ADRESA (
    ID_ADRESA INT DEFAULT ADRESA_SEQ.NEXTVAL,
    STRADA VARCHAR2(255) NOT NULL,
    ORAS VARCHAR2(255) NOT NULL,
    TARA VARCHAR2(255) NOT NULL,
    COD_ZIP VARCHAR2(10) NOT NULL,
    CONSTRAINT CHEIE_PRIMARA_ADRESA PRIMARY KEY (ID_ADRESA)
);

```

The screenshot shows the Oracle SQL Developer interface. In the central workspace, there is a code editor window containing the following SQL script:

```

--TABELA ADRESA
CREATE TABLE ADRESA (
    ID_ADRESA INT DEFAULT ADRESA_SEQ.NEXTVAL,
    STRADA VARCHAR2(255) NOT NULL,
    ORAS VARCHAR2(255) NOT NULL,
    TARA VARCHAR2(255) NOT NULL,
    COD_ZIP VARCHAR2(10) NOT NULL,
    CONSTRAINT CHEIE_PRIMARA_ADRESA PRIMARY KEY (ID_ADRESA)
);

INSERT INTO ADRESA (STRADA, ORAS, TARA, COD_ZIP) VALUES ('BULEVARDUL TOMIS 287', 'CONSTANTA', 'ROMANIA', '900407');

INSERT INTO ADRESA (STRADA, ORAS, TARA, COD_ZIP) VALUES ('STRADA VASILE PARVAN 2', 'BOUCURESTI', 'ROMANIA', '901294');

INSERT INTO ADRESA (STRADA, ORAS, TARA, COD_ZIP) VALUES ('BULEVARDUL ALEXANDRU LAPUSNEANU 13', 'CONSTANTA', 'ROMANIA', '900352');

INSERT INTO ADRESA (STRADA, ORAS, TARA, COD_ZIP) VALUES ('STRADA DEGRORIRII 100', 'CONSTANTA', 'ROMANIA', '900294');

INSERT INTO ADRESA (STRADA, ORAS, TARA, COD_ZIP) VALUES ('BULEVARDUL CAROL I NR.12', 'BOUCURESTI', 'ROMANIA', '010292');

INSERT INTO ADRESA (STRADA, ORAS, TARA, COD_ZIP) VALUES ('STRADA MATEI BASARAB NR.47A', 'GALATI', 'ROMANIA', '800921');

INSERT INTO ADRESA (STRADA, ORAS, TARA, COD_ZIP) VALUES ('SPLAIUL UNIRII NR.312 BLOC T12 SC.C', 'BOUCURESTI', 'ROMANIA', '010734');

```

Below the code editor, the 'Script Output' tab is active, showing the results of the executed statements:

```

Table ADRESA created.

1 row inserted.

```

The status bar at the bottom right indicates the task completed in 0.406 seconds.

Curier

```

CREATE TABLE CURIER(
    ID_CURIER INT DEFAULT CURIER_SEQ.NEXTVAL,
    NUME VARCHAR2(255) NOT NULL,
    PRENUME VARCHAR2(255)NOT NULL,
    SALARIU NUMBER(10,2),
    CONSTRAINT PK_CURIER PRIMARY KEY (ID_CURIER)
)

```

);

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, a SQL script is being run:

```
CREATE TABLE CURIER
(
    ID_CURIER INT DEFAULT CURIER_SEQ.NEXTVAL,
    NUME VARCHAR2(255) NOT NULL,
    PRENUME VARCHAR2(255) NOT NULL,
    SALARIU NUMBER(10,2),
    CONSTRAINT PK_CURIER PRIMARY KEY (ID_CURIER)
);

INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('ION', 'ALEXANDRU', 2800);
INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('POFESCU', 'MARIA', 3200);
INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('IOANESCU', 'GEORGE', 2900);
INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('STANESCU', 'IANA', 3100);
INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('MIRCEA', 'RADU', 3200);
INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('MARTINESCU', 'VANINA', 3300);
INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('ALEXANDRU', 'RADU', 3400);
INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('STEFANESCU', 'CRISTINA', 2800);
INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('IOANESCU', 'ADINA', 3500);
INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('RADU', 'FLORIN', 3500);
INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('STEFANESCU', 'VLAD', 3600);
INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('CONSTANTINESCU', 'MARIUS', 3700);
SELECT * FROM curier;
```

The bottom-left pane displays the results of the query:

```
Table CURIER created.

1 row inserted.

1 row inserted.
```

The bottom-right pane shows the system status and timestamp:

```
Line 174 Column 1 | Insert | Modified | Windows: O
73°F Sunny | Search | Home | Help | 10:07 AM 8/7/2024
```

The screenshot shows the Oracle SQL Developer interface with the CURIER table selected. The top navigation bar includes File, Edit, View, Navigate, Run, Team, Tools, Window, and Help. The title bar indicates the connection is to the UTILIZATOR.CURIER table at the tutorial schema.

The main area shows the table structure with the following columns:

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID_CURIER	NUMBER (38,0)	No	"UTILIZATOR"."CURIER_SEQ"."NEXTVAL"	1 (null)	
2 NUME	VARCHAR2 (255 BYTE)	No	(null)	2 (null)	
3 PRENUME	VARCHAR2 (255 BYTE)	No	(null)	3 (null)	
4 SALARIU	NUMBER(10,2)	Yes	(null)	4 (null)	

Masina_Livrare

```
CREATE TABLE MASINA_LIVRARE(
    ID_MASINA INT DEFAULT MASINA_LIVRARE_SEQ.NEXTVAL,
    MODEL_MASINA VARCHAR2(255),
    NUMAR_INMATRICULARE VARCHAR2(255)NOT NULL,
    CONSTRAINT PK_MASINA PRIMARY KEY (ID_MASINA)
);
```

The screenshot shows the Oracle SQL Developer interface with two windows open. The top window is titled 'Oracle SQL Developer : tutorial' and contains a 'Query Builder' pane with the following SQL script:

```
--Tabela MASINA_LIVRARE
CREATE TABLE MASINA_LIVRARE (
    ID_MASINA INT DEFAULT MASINA_LIVRARE_SEQ.NEXTVAL,
    MODEL_MASINA VARCHAR2(255),
    NUMAR_INMATRICULARE VARCHAR2(255) NOT NULL,
    CONSTRAINT PK_MASINA PRIMARY KEY (ID_MASINA)
);

INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE) VALUES ('Mercedes-Benz Sprinter', 'B-123-ABC');
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE) VALUES ('Ford Transit', 'BV-456-DEF');
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE) VALUES ('Fiat Ducato', 'B-789-GHI');
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE) VALUES ('Volkswagen Crafter', 'CT-101-JKL');
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE) VALUES ('Renault Master', 'B-202-MNO');
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE) VALUES ('Peugeot Boxer', 'GL-303-PQR');
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE) VALUES ('Iveco Daily', 'B-404-SUV');
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE) VALUES ('Vauxhall Movano', 'CT-505-WXY');
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE) VALUES ('Opel Movano', 'B-606-YZN');
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE) VALUES ('Vauxhall NV400', 'B-707-RCD');
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE) VALUES ('Hyundai B350', 'BV-808-EFG');
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE) VALUES ('KIA TOE', 'CT-909-HIJ');
```

The bottom window is titled 'Script Output' and shows the results of the execution:

```
Table MASINA_LIVRARE created.

1 row inserted.

1 row inserted.
```

Livrare

CREATE TABLE LIVRARE (

```
ID_LIVRARE INT DEFAULT LIVRARE_SEQ.NEXTVAL,  
METODA_LIVRARE VARCHAR2(255) NOT NULL,  
PRET_LIVRARE NUMBER(10,2),  
STARE VARCHAR2(255) NOT NULL,  
DATA_LIVRARE DATE,  
ID_ADRESA INT,  
ID_CURIER INT,
```

```

ID_MASINA INT,
CONSTRAINT PK_UTILIZATOR PRIMARY KEY (ID_LIVRARE),
CONSTRAINT FK_LIVRARE_ADRESA FOREIGN KEY(ID_ADRESA) REFERENCES
ADRESA(ID_ADRESA),
CONSTRAINT FK_LIVRARE_CURIER FOREIGN KEY(ID_CURIER) REFERENCES
CURIER(ID_CURIER),
CONSTRAINT FK_LIVRARE_MASINA FOREIGN KEY(ID_MASINA) REFERENCES
MASINA_LIVRARE(ID_MASINA),
CONSTRAINT STARE_CHECK CHECK (STARE IN ('IN CURS DE PROCESARE',
'EFFECTUATA', 'IN CURS DE LIVRARE')),
CONSTRAINT PRET_LIVRARE_CHECK CHECK (PRET_LIVRARE >= 0)
);

```

The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** Welcome Page / tutorial
- Worksheet:** Query Builder


```

CREATE TABLE LIVRARE (
    ID_LIVRARE INT DEFAULT LIVRARE_SEQ.NEXTVAL,
    METODA_LIVRARE VARCHAR2(255) NOT NULL,
    PRET_LIVRARE NUMBER(10,2),
    STARE VARCHAR2(255) NOT NULL,
    DATA_LIVRARE DATE,
    ID_ADRESA INT,
    ID_CURIER INT,
    ID_MASINA INT,
    CONSTRAINT PK_UTILIZATOR PRIMARY KEY (ID_LIVRARE),
    CONSTRAINT FK_LIVRARE_ADRESA FOREIGN KEY(ID_ADRESA) REFERENCES ADRESA(ID_ADRESA),
    CONSTRAINT FK_LIVRARE_CURIER FOREIGN KEY(ID_CURIER) REFERENCES CURIER(ID_CURIER),
    CONSTRAINT FK_LIVRARE_MASINA FOREIGN KEY(ID_MASINA) REFERENCES MASINA_LIVRARE(ID_MASINA),
    CONSTRAINT STARE_CHECK CHECK (STARE IN ('IN CURS DE PROCESARE', 'EFFECTUATA', 'IN CURS DE LIVRARE')),
    CONSTRAINT PRET_LIVRARE_CHECK CHECK (PRET_LIVRARE >= 0)
);

INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('CURIER', 12.90, 'EFFECTUATA', '12-MAY-2024', 1,2,7);
INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('EASTBOX', 6.90, 'EFFECTUATA', '15-MAY-2024', 4,3,10);
INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('EASTBOX', 6.90, 'EFFECTUATA', '15-MAY-2024', 4,3,10);
INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('CURIER', 16.99, 'IN CURS DE LIVRARE', '20-MAY-2024', 9,1,5);
INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('CURIER', 10, 'EFFECTUATA', '20-MAY-2024', 9,1,5);
INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('CURIER', 14.99, 'EFFECTUATA', '02-DEC-2024', 10,7,11);
      
```
- Script Output:** Task completed in 0.591 seconds
- Table:** LIVRARE created.
- System Status:** 73°F Sunny
- System Bar:** Line 226 Column 30 | Insert | Modified | Windows | 10:21 AM | 6/7/2024

The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : Table UTILIZATOR.LIVRARE@tutorial". The menu bar includes File, Edit, View, Navigate, Run, Team, Tools, Window, and Help. Below the menu is a toolbar with various icons. The main window shows a connection named "tutorial" and a table named "LIVRARE". The "Columns" tab is selected, displaying the following table structure:

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID_LIVRARE	NUMBER(38,0)	No	"UTILIZATOR"."LIVRARE_SEQ"."NEXTVAL"	1	(null)
2 METODA_LIVRARE	VARCHAR2(255 BYTE)	No	(null)	2	(null)
3 PRET_LIVRARE	NUMBER(10,2)	Yes	(null)	3	(null)
4 STARE	VARCHAR2(255 BYTE)	No	(null)	4	(null)
5 DATA_LIVRARE	DATE	Yes	(null)	5	(null)
6 ID_ADRESA	NUMBER(38,0)	Yes	(null)	6	(null)
7 ID_CURIER	NUMBER(38,0)	Yes	(null)	7	(null)
8 ID_MASINA	NUMBER(38,0)	Yes	(null)	8	(null)

Comanda

```

CREATE TABLE COMANDA (
    ID_COMANDA INT DEFAULT COMANDA_SEQ.NEXTVAL,
    TOTAL_PLATA NUMBER(10,2) NOT NULL,
    STARE_COMANDA VARCHAR2(255),
    DATA_COMANDA DATE,
    ID_UTILIZATOR INT,
    ID_LIVRARE INT UNIQUE,
    CONSTRAINT CHEIE_PRIMARA_COMANDA PRIMARY KEY (ID_COMANDA),
    CONSTRAINT TOTAL_PLATA_CHECK CHECK (TOTAL_PLATA >= 0),
    CONSTRAINT FK_COMANDA_UTILIZATOR FOREIGN KEY(ID_UTILIZATOR)
        REFERENCES UTILIZATOR(ID_UTILIZATOR),
    CONSTRAINT FK_COMANDA_LIVRARE FOREIGN KEY(ID_LIVRARE)
        REFERENCES LIVRARE(ID_LIVRARE),
    CONSTRAINT STARE_COMANDA_CHECK CHECK (STARE_COMANDA IN ('IN
CURS DE PROCESARE', 'EFECTUATA', 'IN CURS DE PREGATIRE'))
);

```

The screenshot shows the Oracle SQL Developer interface with two windows open. The top window is a Worksheet containing DDL and DML code for creating a table and inserting data. The bottom window is a Database Browser showing the structure of the COMANDA table and its data.

Worksheet Content:

```
--Tabela COMANDA
CREATE TABLE COMANDA (
    ID_COMANDA INT DEFAULT COMANDA_SEQ.NEXTVAL,
    TOTAL_PLATA NUMBER(10,2) NOT NULL,
    STARE_COMANDA VARCHAR2(255),
    DATA_COMANDA DATE,
    ID_UTILIZATOR INT,
    ID_LIVRARE NUMBER(38,0)
);
ALTER TABLE COMANDA
CONSTRAINT COMANDA_PK PRIMARY KEY (ID_COMANDA),
CONSTRAINT TOTAL_PLATA_CHECK CHECK (TOTAL_PLATA >= 0),
CONSTRAINT FK_COMANDA_UTILIZATOR FOREIGN KEY (ID_UTILIZATOR) REFERENCES UTILIZATOR(ID_UTILIZATOR),
CONSTRAINT FK_COMANDA_LIVRARE FOREIGN KEY (ID_LIVRARE) REFERENCES LIVRARE(ID_LIVRARE),
CONSTRAINT STARE_COMANDA_CHECK CHECK (STARE_COMANDA IN ('IN CURS DE PROCESARE', 'EFFECTUATA', 'IN CURS DE PREGATIRE'))
;

INSERT INTO COMANDA (TOTAL_PLATA, STARE_COMANDA, DATA_COMANDA, ID_UTILIZATOR, ID_LIVRARE) VALUES(100, 'EFFECTUATA', '23-FEB-2023', 1,1);
INSERT INTO COMANDA (TOTAL_PLATA, STARE_COMANDA, DATA_COMANDA, ID_UTILIZATOR, ID_LIVRARE) VALUES(106,9, 'IN CURS DE PREGATIRE', '10-MAY-2024', 3,2);
INSERT INTO COMANDA (TOTAL_PLATA, STARE_COMANDA, DATA_COMANDA, ID_UTILIZATOR, ID_LIVRARE) VALUES(137,6, 'IN CURS DE PREGATIRE', '9-MAY-2024', 5,3);
INSERT INTO COMANDA (TOTAL_PLATA, STARE_COMANDA, DATA_COMANDA, ID_UTILIZATOR, ID_LIVRARE) VALUES(129,5, 'IN CURS DE PREGATIRE', '15-MAY-2024', 6,10);
INSERT INTO COMANDA (TOTAL_PLATA, STARE_COMANDA, DATA_COMANDA, ID_UTILIZATOR, ID_LIVRARE) VALUES(65,4, 'EFFECTUATA', '23-FEB-2023', 1, 13);

Script Output: 4 | Query Result: 0 | Task completed in 0.431 seconds
```

Database Browser Content:

Column Name	Data Type	Nullable	Data Default	Column ID	Comments
1 ID_COMANDA	NUMBER(38, 0)	No	"UTILIZATOR"."COMANDA_SEQ"."NEXTVAL"	1 (null)	
2 TOTAL_PLATA	NUMBER(10, 2)	No	(null)	2 (null)	
3 STARE_COMANDA	VARCHAR2(255 BYTE)	Yes	(null)	3 (null)	
4 DATA_COMANDA	DATE	Yes	(null)	4 (null)	
5 ID_UTILIZATOR	NUMBER(38, 0)	Yes	(null)	5 (null)	
6 ID_LIVRARE	NUMBER(38, 0)	Yes	(null)	6 (null)	

Card

CREATE TABLE CARD (

```
ID_CARD INT DEFAULT CARD_SEQ.NEXTVAL,  
NUMAR_CARD VARCHAR2(16) NOT NULL,  
NUME_PROPIETAR VARCHAR2(255),  
CVV VARCHAR2(4),  
DATA_EXPIRARE DATE,  
ID_UTILIZATOR INT,
```

CONSTRAINT VERIFICARE_NUMAR_CARD CHECK (LENGTH(NUMAR_CARD) = 16),

CONSTRAINT VERIFICARE_CVV CHECK (LENGTH(CVV)=3 or LENGTH(CVV)=4),

CONSTRAINT CHEIE_PRIMARA_CARD PRIMARY KEY (ID_CARD),

CONSTRAINT NUMAR_CARD_UNIC UNIQUE (NUMAR_CARD)

);

```
-- TABELA CARD
CREATE TABLE CARD (
    ID_CARD INT DEFAULT CARD_SEQ.NEXTVAL,
    NUMAR_CARD VARCHAR2(16) NOT NULL,
    NUME_PROPIETAR VARCHAR2(255),
    CVV VARCHAR2(4),
    DATA_EXPIRARE DATE,
    ID_UTILIZATOR INT,
    CONSTRAINT VERIFICARE_NUMAR_CARD CHECK ( LENGTH(NUMAR_CARD) = 16),
    CONSTRAINT VERIFICARE_CVV CHECK ( LENGTH(CVV)=3 or LENGTH(CVV)=4),
    CONSTRAINT CHEIE_PRIMARA_CARD PRIMARY KEY (ID_CARD),
    CONSTRAINT NUMAR_CARD_UNIC UNIQUE (NUMAR_CARD)
);

INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator) VALUES ('0000111122223333', 'EMMA MACIOSCA', '123', '01-FEB-2026', 1);
INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator) VALUES ('1800789243671234', 'IULIANA MARIN', '812', '01-MAR-2028', 6);
INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator) VALUES ('8902489019003532', 'POPESTESCU ANDREEA IOANA', '602', '01-APR-2029', 2);
INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator) VALUES ('0000000000000000', 'ANDREI ANDREI', '241', '01-JUN-2025', 3);
INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator) VALUES ('1234567890123456', 'ANDRA BORDEA', '907', '01-NOV-2024', 4);
INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator) VALUES ('3456789012345678', 'MIRCEA SABINA', '109', '01-DEC-2029', 5);
INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator) VALUES ('4723699318001431', 'POP DARIO', '192', '01-JAN-2030', 7);
INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator) VALUES ('23674471601239', 'MARES DARUJUS', '276', '01-APR-2025', 9);
INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator) VALUES ('6892163417001465', 'POPA MIRTA', '125', '01-MAY-2027', 8);
INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator) VALUES ('1354235715003667', 'BALAN IOANA', '813', '01-JUL-2024', 10);
INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator) VALUES ('1565611319003575', 'BALAN IOANA', '837', '01-DEC-2029', 10);

Table CARD created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID_CARD	NUMBER(38,0)	No	"UTILIZATOR"."CARD_SEQ"."NEXTVAL"	1 (null)	
2 NUMAR_CARD	VARCHAR2(16 BYTE)	No	(null)	2 (null)	
3 NUME_PROPIETAR	VARCHAR2(255 BYTE)	Yes	(null)	3 (null)	
4 CVV	VARCHAR2(4 BYTE)	Yes	(null)	4 (null)	
5 DATA_EXPIRARE	DATE	Yes	(null)	5 (null)	
6 ID_UTILIZATOR	NUMBER(38,0)	Yes	(null)	6 (null)	

Plata

CREATE TABLE PLATA (

```

ID_PLATA INT DEFAULT PLATA_SEQ.NEXTVAL,
VALOARE NUMBER(10,2) NOT NULL,
DATA_PLATA DATE,
STARE VARCHAR2(255),
ID_CARD INT,
ID_COMANDA INT,
CONSTRAINT CHEIE_PRIMARA_PLATA PRIMARY KEY (ID_PLATA),
CONSTRAINT VALOARE_CHECK CHECK (VALOARE >= 0),
CONSTRAINT FK_PLATA_CARD FOREIGN KEY(ID_CARD) REFERENCES CARD(ID_CARD),
CONSTRAINT FK_PLATA_COMANDA FOREIGN KEY(ID_COMANDA) REFERENCES COMANDA(ID_COMANDA),
CONSTRAINT STARE_PLATA_CHECK CHECK (STARE IN ('IN CURS DE PROCESARE', 'EFFECTUATA', 'RESPINSA'))
);

```

The screenshot shows the Oracle SQL Developer interface with the following details:

- File Bar:** File, Edit, View, Navigate, Run, Source, Team, Tools, Window, Help.
- Connections:** Welcome Page, Autostart.
- Worksheet:** Query Builder.
- SQL Script:**

```

--Table: PLATA
CREATE TABLE PLATA (
    ID_PLATA INT DEFAULT PLATA_SEQ.NEXTVAL,
    VALOARE NUMBER(10,2) NOT NULL,
    DATA_PLATA DATE,
    STARE VARCHAR2(255),
    ID_CARD INT,
    ID_COMANDA INT,
    CONSTRAINT VALOARE_CHECK CHECK (VALOARE >= 0),
    CONSTRAINT FK_PLATA_CARD FOREIGN KEY(ID_CARD) REFERENCES CARD(ID_CARD),
    CONSTRAINT FK_PLATA_COMANDA FOREIGN KEY(ID_COMANDA) REFERENCES COMANDA(ID_COMANDA),
    CONSTRAINT STARE_PLATA_CHECK CHECK (STARE IN ('IN CURS DE PROCESARE', 'EFFECTUATA', 'RESPINSA'))
);

INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA) VALUES(100, '23-FEB-2023', 'RESPINSA', 1, 1);
INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA) VALUES(129, 5, '15-MAY-2024', 'EFFECTUATA', 2, 1);
INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA) VALUES(104, 9, '10-MAY-2024', 'EFFECTUATA', 4, 1);
INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA) VALUES(124, 9, '10-MAY-2024', 'RESPINSA', 3, 4);
INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA) VALUES(123, 6, '09-MAY-2024', 'EFFECTUATA', 5, 1);
INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA) VALUES(127, 6, '09-MAY-2024', 'EFFECTUATA', 6, 1);
INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA) VALUES(69, 4, '23-FEB-2023', 'EFFECTUATA', 1, 5);

```
- Script Output:** Task completed in 0.303 seconds.
- Table Data:**

```

Table PLATA created.

1 row inserted.

```
- System Status:** Sunny, 10:29 AM, 6/7/2024.

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID_PLATA	NUMBER(38,0)	No	"UTILIZATOR"."PLATA_SEQ"."NEXTVAL"	1 (null)	
2 VALOARE	NUMBER(10,2)	No	(null)	2 (null)	
3 DATA_PLATA	DATE	Yes	(null)	3 (null)	
4 STARE	VARCHAR2(255 BYTE)	Yes	(null)	4 (null)	
5 ID_CARD	NUMBER(38,0)	Yes	(null)	5 (null)	
6 ID_COMANDA	NUMBER(38,0)	Yes	(null)	6 (null)	

Cos_Cumparaturi

```

CREATE TABLE COS_CUMPARATURI (
    ID_COS INT DEFAULT COS_CUMPARATURI_SEQ.NEXTVAL,
    ID_UTILIZATOR INT UNIQUE NOT NULL,
    TOTAL_COS NUMBER(10,2),
    CONSTRAINT CHEIE_PRIMARA_COS_CUMPARATURI PRIMARY KEY (ID_COS),
    CONSTRAINT FK_COS_UTILIZATOR FOREIGN KEY(ID_UTILIZATOR)
    REFERENCES UTILIZATOR(ID_UTILIZATOR),
    CONSTRAINT TOTAL_COS_CHECK CHECK (TOTAL_COS >= 0)
);

```

```

CREATE TABLE COS_CUMPARATURI (
    ID_COS INT DEFAULT COS_CUMPARATURI_SEQ.NEXTVAL,
    ID_UTILIZATOR NUMBER(38,0),
    TOTAL_COS NUMBER(10,2),
    CONSTRAINT FK_COS_UTILIZATOR FOREIGN KEY(ID_UTILIZATOR)
    REFERENCES UTILIZATOR(ID_UTILIZATOR),
    CONSTRAINT TOTAL_COS_CHECK CHECK (TOTAL_COS >= 0)
);

INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (1,10.5);
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (1,150);
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (1,200);
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (1,49);
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (1,200.43);
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (1,120);
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (1,120);
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (1,99);
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (1,200.43);
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (1,200.43);

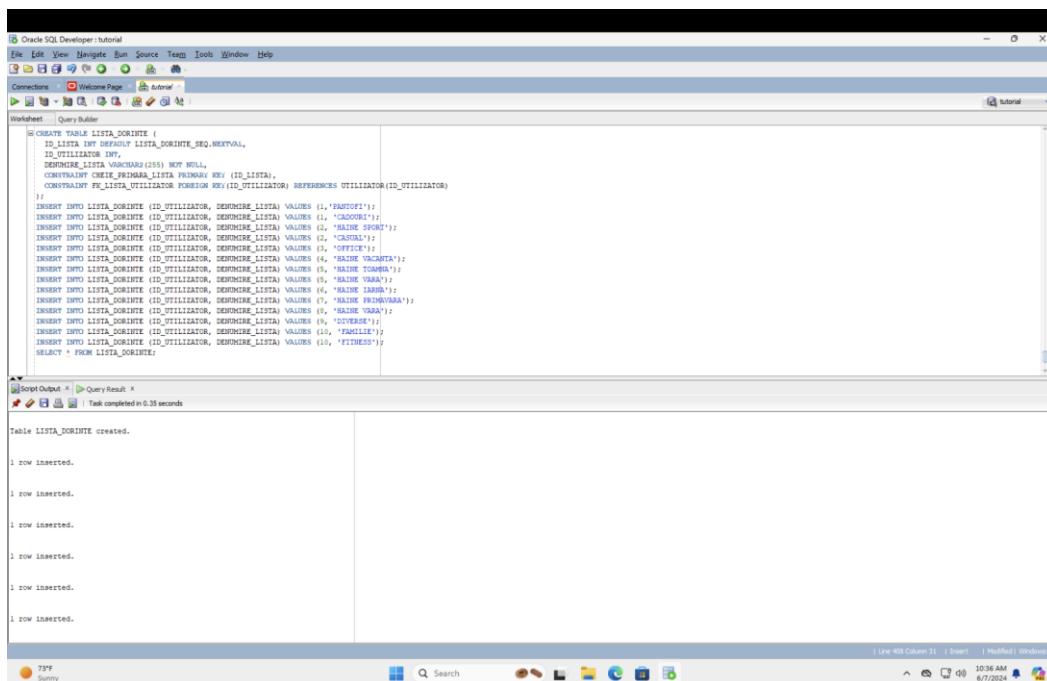
SELECT * FROM COS_CUMPARATURI;

```

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID_COS	NUMBER(38,0)	No	"UTILIZATOR"."COS_CUMPARATURI_SEQ"."NEXTVAL"	1	(null)
2 ID_UTILIZATOR	NUMBER(38,0)	No	(null)	2	(null)
3 TOTAL_COS	NUMBER(10,2)	Yes	(null)	3	(null)

Lista_Dorinte

```
CREATE TABLE LISTA_DORINTE (
    ID_LISTA INT DEFAULT LISTA_DORINTE_SEQ.NEXTVAL,
    ID_UTILIZATOR INT,
    DENUMIRE_LISTA VARCHAR2(255) NOT NULL,
    CONSTRAINT CHEIE_PRIMARA_LISTA PRIMARY KEY (ID_LISTA),
    CONSTRAINT FK_LISTA_UTILIZATOR FOREIGN KEY(ID_UTILIZATOR)
    REFERENCES UTILIZATOR(ID_UTILIZATOR)
);
```



The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : Table UTILIZATOR.LISTA_DORINTE@tutorial". The menu bar includes File, Edit, View, Navigate, Run, Team, Tools, Window, and Help. Below the menu is a toolbar with various icons. The main window displays the "LISTA_DORINTE" table structure. The table has three columns: ID_LISTA (NUMBER(38,0)), ID_UTILIZATOR (NUMBER(38,0)), and DENUMIRE_LISTA (VARCHAR2(255 BYTE)). The primary key constraint is defined as CHEIE_PRIMARA_CATEGORIE.

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID_LISTA	NUMBER (38, 0)	No	"UTILIZATOR"."LISTA_DORINTE_SEQ"."NEXTVAL"	1	(null)
2 ID_UTILIZATOR	NUMBER (38, 0)	Yes	(null)	2	(null)
3 DENUMIRE_LISTA	VARCHAR2 (255 BYTE)	No	(null)	3	(null)

Categorie

```
CREATE TABLE CATEGORIE (
    ID_CATEGORIE INT DEFAULT CATEGORIE_SEQ.NEXTVAL,
    DENUMIRE_CATEGORIE VARCHAR2(255) NOT NULL,
    CONSTRAINT CHEIE_PRIMARA_CATEGORIE PRIMARY KEY (ID_CATEGORIE)
);
```

The screenshot shows the Oracle SQL Developer interface. In the top window, a script is run to create the CATEGORIE table and insert 10 rows of data. The table is created with an ID_CATEGORIE primary key and a DENUMIRE_CATEGORIE column. The inserted data includes various categories like TRICOU, PANTALONI, etc.

```

-- TABELA CATEGORIE
CREATE TABLE CATEGORIE (
    ID_CATEGORIE INT DEFAULT CATEGORIE_SEQ.NEXTVAL,
    DENUMIRE_CATEGORIE VARCHAR2(255) NOT NULL,
    CONSTRAINT CHEIE_PRIMARA_CATEGORIE PRIMARY KEY (ID_CATEGORIE)
);

INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('TRICOU');
INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('PANTALONI');
INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('BLUZĂ');
INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('GEACĂ');
INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('VESTA');
INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('CAMASA');
INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('ŞEANTA');
INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('OCHELARI DE SOARE');
INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('POLIMER');
INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('POSTA');
INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('MĂRFA');

SELECT * FROM CATEGORIE;

```

The bottom window shows the CATEGORIE table details in the Object Navigator. It lists columns: ID_CATEGORIE (NUMBER), DENUMIRE_CATEGORIE (VARCHAR2(255)), and their respective properties and comments.

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID_CATEGORIE	NUMBER (38, 0)	No	"UTILIZATOR"."CATEGORIE_SEQ"."NEXTVAL"	1 (null)	
2 DENUMIRE_CATEGORIE	VARCHAR2 (255 BYTE)	No	(null)	2 (null)	

Produs

```

CREATE TABLE PRODUS (
    ID_PRODUS INT DEFAULT PRODUS_SEQ.NEXTVAL,
    DENUMIRE_PRODUS VARCHAR2(255),
    PRET NUMBER(10,2) NOT NULL,
    STOC NUMBER(10),
    ID_CATEGORIE INT NOT NULL,
    CONSTRAINT CHEIE_PRIMARA_PRODUS PRIMARY KEY (ID_PRODUS),
    CONSTRAINT FK_PRODUS_CATEGORIE FOREIGN KEY (ID_CATEGORIE)
        REFERENCES CATEGORIE(ID_CATEGORIE)
);

```

```

CREATE TABLE PRODUS (
    ID_PRODUS INT DEFAULT PRODUS_SEQ.NEXTVAL,
    DENUMIRE_PRODUS VARCHAR2(255 BYTE),
    PRET NUMBER(10,2) NOT NULL,
    STOC NUMBER(10),
    ID_CATEGORIE INT,
    CONSTRAINT PK_PRODUS PRIMARY KEY (ID_PRODUS),
    CONSTRAINT FK_PRODUS_CATEGORIE FOREIGN KEY (ID_CATEGORIE)
        REFERENCES CATEGORIE(ID_CATEGORIE)
);

INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE) VALUES ('TRICOU ALBASTRU',55.50,10,1);
INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE) VALUES ('FANTOMA NEGRU',120.00,6,0);
INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE) VALUES ('MAGNETIC ROZ',100.00,15,1);
INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE) VALUES ('TRICOU ORA',49.99,25,11);
INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE) VALUES ('BROCOLI DE PEAUN',99.99,10,11);
INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE) VALUES ('SALATĂ DE LIMBA',100.00,15,12);
INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE) VALUES ('CAROTA VERDE',59.99,15,4);
INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE) VALUES ('LIMBALĂ DE LIMBA',129.50,20,7);

```

Table PRODUS created.
1 row inserted.
1 row inserted.

Actions...						
COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS	
1 ID_PRODUS	NUMBER(38,0)	No	"UTILIZATOR"."PRODUS_SEQ"."NEXTVAL"	1 (null)		
2 DENUMIRE_PRODUS	VARCHAR2(255 BYTE)	Yes	(null)	2 (null)		
3 PRET	NUMBER(10,2)	No	(null)	3 (null)		
4 STOC	NUMBER(10,0)	Yes	(null)	4 (null)		
5 ID_CATEGORIE	NUMBER(38,0)	No	(null)	5 (null)		

Produs_Comanda

```

CREATE TABLE PRODUS_COMANDA(
    ID_PRODUS INT,
    ID_COMANDA INT,
    CONSTRAINT PK_PRODUS_COMANDA PRIMARY KEY (ID_PRODUS, ID_COMANDA),
    CONSTRAINT FK_PRODUS_COMANDA_PRODUS FOREIGN KEY (ID_PRODUS)
        REFERENCES PRODUS(ID_PRODUS),
    CONSTRAINT FK_PRODUS_COMANDA_COMANDA FOREIGN KEY (ID_COMANDA)
        REFERENCES COMANDA(ID_COMANDA)
);

```

The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : Table UTILIZATOR.PRODUS_COMANDA@tutorial". The menu bar includes File, Edit, View, Navigate, Run, Team, Tools, Window, and Help. Below the menu is a toolbar with various icons for database operations. The main window displays the "Connections" tab selected, showing the "PRODUS_COMANDA" table. The table structure is listed under the "Columns" tab, which includes columns for COLUMN_NAME, DATA_TYPE, NULLABLE, DATA_DEFAULT, COLUMN_ID, and COMMENTS. The table has two rows: ID_PRODUS and ID_COMANDA, both defined as NUMBER(38,0) and set to NOT NULL with a default value of null.

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID_PRODUS	NUMBER(38,0)	No	(null)	1	(null)
2	ID_COMANDA	NUMBER(38,0)	No	(null)	2	(null)

Produs Cos

```
CREATE TABLE PRODUS_COS(
    ID_PRODUS INT,
    ID_COS INT,
    CONSTRAINT PK_PRODUS_COS PRIMARY KEY (ID_PRODUS, ID_COS),
    CONSTRAINT FK_PRODUS_COS_PRODUS FOREIGN KEY (ID_PRODUS)
        REFERENCES PRODUS(ID_PRODUS),
    CONSTRAINT FK_PRODUS_COS_COS FOREIGN KEY (ID_COS) REFERENCES
        COS_CUMPARATURI(ID_COS)
);
```

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, the code for creating the `PRODUS_COS` table is displayed:

```
-- TABLE PRODUS_COS
CREATE TABLE PRODUS_COS(
    ID_PRODUS INT,
    ID_COS INT,
    CONSTRAINT PK_PRODUS_COS PRIMARY KEY (ID_PRODUS, ID_COS),
    CONSTRAINT FK_PRODUS_COS_PRODUS FOREIGN KEY (ID_PRODUS) REFERENCES PRODUS(ID_PRODUS),
    CONSTRAINT FK_PRODUS_COS_COS FOREIGN KEY (ID_COS) REFERENCES COS_COMPARATUI(ID_COS)
);

INSERT INTO PRODUS_COS VALUES(1,1);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(1,2);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(2,3);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(1,3);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(2,4);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(4,5);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(5,6);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(6,7);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(7,8);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(8,9);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(9,10);
SELECT * FROM PRODUS_COS;
```

In the bottom-left pane, the output of the query shows 10 rows inserted successfully. The status bar at the bottom right indicates the task completed in 0.34 seconds.

The screenshot shows the Oracle SQL Developer interface with the `PRODUS_COS` table selected. The table structure is displayed in the main pane:

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID_PRODUS	NUMBER (38,0)	No	(null)	1	(null)
2 ID_COS	NUMBER (38,0)	No	(null)	2	(null)

Produs_Lista

```
CREATE TABLE PRODUS_LISTA(
    ID_PRODUS INT,
    ID_LISTA INT,
    CONSTRAINT PK_PRODUS_LISTA PRIMARY KEY (ID_PRODUS, ID_LISTA),
    CONSTRAINT FK_PRODUS_LISTA_PRODUS FOREIGN KEY (ID_PRODUS)
        REFERENCES PRODUS(ID_PRODUS),
    CONSTRAINT FK_PRODUS_LISTA_LISTA FOREIGN KEY (ID_LISTA)
        REFERENCES LISTA_DORINTE(ID_LISTA)
);
```

The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : Table UTILIZATOR.PRODUS_LISTA@tutorial". The menu bar includes File, Edit, View, Navigate, Run, Team, Tools, Window, and Help. Below the menu is a toolbar with various icons. The main window displays the "Connections" tab selected, showing a list of connections including "tutorial" and "PRODUS_LISTA". A sub-tab bar below shows "Columns", "Data", "Model", "Constraints", "Grants", "Statistics", "Triggers", "Flashback", "Dependencies", "Details", "Partitions", "Indexes", and "SQL". The "Actions..." button is also visible. The main content area shows the table structure for "PRODUS_LISTA" with columns: COLUMN_NAME, DATA_TYPE, NULLABLE, DATA_DEFAULT, COLUMN_ID, and COMMENTS. The data rows are:

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID_PRODUS	NUMBER(38,0)	No	(null)	1	(null)
2	ID_LISTA	NUMBER(38,0)	No	(null)	2	(null)

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

Utilizator

```
INSERT INTO UTILIZATOR (NUME, PRENUME, EMAIL, PAROLA,  
NUMAR_TELEFON, DATA_NASTERE) VALUES  
('Maciuca', 'Emma', 'emmamaciuca@gmail.com', 'parola2024', '0712345678', '12-FEB-2004');  
  
INSERT INTO UTILIZATOR (NUME, PRENUME, EMAIL, PAROLA,  
NUMAR_TELEFON, DATA_NASTERE) VALUES  
('Popescu', 'Andreea-Ioana', 'popescuandreea@gmail.com', 'abcparola', '0757828352', '14-  
APR-2000');  
  
INSERT INTO UTILIZATOR (NUME, PRENUME, EMAIL, PAROLA,  
NUMAR_TELEFON, DATA_NASTERE) VALUES  
('Ion', 'Andrei', 'ionandrei@gmail.com', 'contmagazin', '0717025379', '20-MAR-1990');  
  
INSERT INTO UTILIZATOR (NUME, PRENUME, EMAIL, PAROLA,  
NUMAR_TELEFON, DATA_NASTERE) VALUES  
('Andreescu', 'Maria', 'mariaandr@gmail.com', '123456789', '0720953487', '09-MAY-2001');  
  
INSERT INTO UTILIZATOR (NUME, PRENUME, EMAIL, PAROLA,  
NUMAR_TELEFON, DATA_NASTERE) VALUES  
('Maciuca', 'Sara', 'maciucasara@gmail.com', 'sara1998', '0709348726', '26-MAY-1998');  
  
INSERT INTO UTILIZATOR (NUME, PRENUME, EMAIL, PAROLA,  
NUMAR_TELEFON, DATA_NASTERE) VALUES
```

('Marin', 'Iuliana', 'mariniuliana@gmail.com', 'iulianamarin12', '0746892537', '12-JUL-1999');

INSERT INTO UTILIZATOR (NUME, PRENUME, EMAIL, PAROLA, NUMAR_TELEFON, DATA_NASTERE) VALUES

('Pop', 'Daria', 'dariap56@gmail.com', 'fructabcd', '0756293641', '05-JUN-2002');

INSERT INTO UTILIZATOR (NUME, PRENUME, EMAIL, PAROLA, NUMAR_TELEFON, DATA_NASTERE) VALUES

('Popa', 'Mara', 'marapopa@gmail.com', 'maraa2103', '0747935728', '21-MAR-2000');

INSERT INTO UTILIZATOR (NUME, PRENUME, EMAIL, PAROLA, NUMAR_TELEFON, DATA_NASTERE) VALUES

('Mares', 'Darius', 'maresdarius@gmail.com', 'guh4ngj5nkf!', '0767937563', '08-JUN-2003');

INSERT INTO UTILIZATOR (NUME, PRENUME, EMAIL, PAROLA, NUMAR_TELEFON, DATA_NASTERE) VALUES

('Balan', 'Ioana', 'balan_ioana@gmail.com', 'ppp273ffs', '0763927455', '15-APR-1989');

INSERT INTO UTILIZATOR (NUME, PRENUME, EMAIL, PAROLA, NUMAR_TELEFON, DATA_NASTERE) VALUES

('Popescu', 'Marian', 'mariannopescu@gmail.com', 'parola_magazin', '0765223465', '11-JUN-1980');

SELECT * FROM UTILIZATOR;

ID_UTILIZATOR	NUME	PRENUME	EMAIL	PAROLA	NUMAR_TELEFON	DATA_NASTERE
1	1 Maciuca	Emma	emmamaciuca@gmail.com	parola2024	0712345678	12-FEB-04
2	2 Popescu	Andreea-Ioana	popescuandreea@gmail.com	abcparola	0757828352	14-APR-00
3	3 Ion	Andrei	ionandrei@gmail.com	contmagazin	0717025379	20-MAR-90
4	4 Andreeescu	Maria	mariaandr@gmail.com	123456789	0720953487	09-MAY-01
5	5 Maciuca	Sara	maciucasara@gmail.com	saral1998	0709348726	26-MAY-98
6	6 Marin	Iuliana	mariniuliana@gmail.com	iulianamarin12	0746892537	12-JUL-99
7	7 Pop	Daria	dariap56@gmail.com	fructabcd	0756293641	05-JUN-02
8	8 Popa	Mara	marapopa@gmail.com	maraa2103	0747935728	21-MAR-00
9	9 Mares	Darius	maresdarius@gmail.com	guh4ngj5nkf!	0767937563	08-JUN-03
10	10 Balan	Ioana	balan_ioana@gmail.com	ppp273ffs	0763927455	15-APR-89
11	11 Popescu	Marian	mariannopescu@gmail.com	parola_magazin	0765223465	11-JUN-80

Adresa

INSERT INTO ADRESA (STRADA, ORAS, TARA, COD_ZIP) VALUES ('BULEVARDUL TOMIS 287', 'CONSTANTA', 'ROMANIA', '900407');

INSERT INTO ADRESA (STRADA, ORAS, TARA, COD_ZIP) VALUES ('STRADA VASILE PARVAN 2', 'BUCURESTI', 'ROMANIA', '901294');

```

INSERT INTO ADRESA (STRADA, ORAS, TARA, COD_ZIP) VALUES ('BULEVARDUL
ALEXANDRU LAPUSNEANU 13', 'CONSTANTA', 'ROMANIA', '900352');

INSERT INTO ADRESA (STRADA, ORAS, TARA, COD_ZIP) VALUES ('STRADA
DEZROBIRII 100', 'CONSTANTA', 'ROMANIA', '900294');

INSERT INTO ADRESA (STRADA, ORAS, TARA, COD_ZIP) VALUES ('BULEVARDUL
CAROL I NR.12', 'BUCURESTI', 'ROMANIA', '010292');

INSERT INTO ADRESA (STRADA, ORAS, TARA, COD_ZIP) VALUES ('STRADA
MATEI BASARAB NR.67A', 'GALATI', 'ROMANIA', '800921');

INSERT INTO ADRESA (STRADA, ORAS, TARA, COD_ZIP) VALUES ('SPLAIUL
UNIRII NR.312 BLOC T12 SC.C', 'BUCURESTI', 'ROMANIA', '010734');

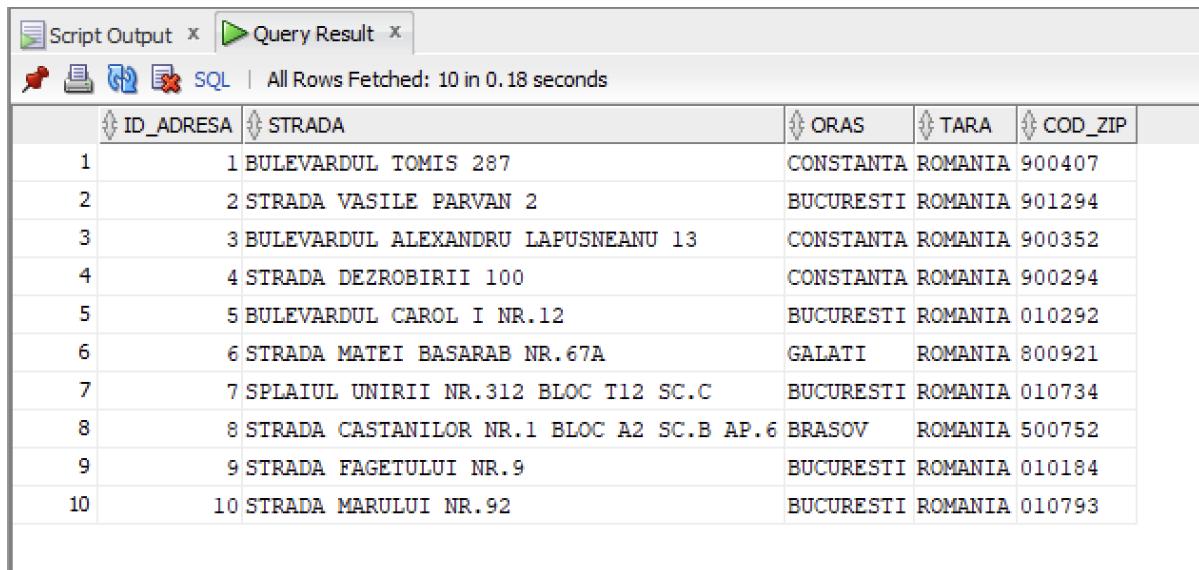
INSERT INTO ADRESA (STRADA, ORAS, TARA, COD_ZIP) VALUES ('STRADA
CASTANILOR NR.1 BLOC A2 SC.B AP.6', 'BRASOV', 'ROMANIA', '500752');

INSERT INTO ADRESA (STRADA, ORAS, TARA, COD_ZIP) VALUES ('STRADA
FAGETULUI NR.9', 'BUCURESTI', 'ROMANIA', '010184');

INSERT INTO ADRESA (STRADA, ORAS, TARA, COD_ZIP) VALUES ('STRADA
MARULUI NR.92', 'BUCURESTI', 'ROMANIA', '010793');

INSERT INTO ADRESA (STRADA, ORAS, TARA, COD_ZIP) VALUES ('STRADA
TACHE IONESCU 10', 'BRAILA', 'ROMANIA', '903644');

```



The screenshot shows the MySQL Workbench interface with the 'Query Result' tab selected. The results of the previous SQL statements are displayed in a grid. The columns are labeled: ID_ADRESA, STRADA, ORAS, TARA, and COD_ZIP. The data consists of 10 rows, each representing an address entry from the ADRESA table.

ID_ADRESA	STRADA	ORAS	TARA	COD_ZIP
1	1 BULEVARDUL TOMIS 287	CONSTANTA	ROMANIA	900407
2	2 STRADA VASILE PARVAN 2	BUCURESTI	ROMANIA	901294
3	3 BULEVARDUL ALEXANDRU LAPUSNEANU 13	CONSTANTA	ROMANIA	900352
4	4 STRADA DEZROBIRII 100	CONSTANTA	ROMANIA	900294
5	5 BULEVARDUL CAROL I NR.12	BUCURESTI	ROMANIA	010292
6	6 STRADA MATEI BASARAB NR.67A	GALATI	ROMANIA	800921
7	7 SPLAIUL UNIRII NR.312 BLOC T12 SC.C	BUCURESTI	ROMANIA	010734
8	8 STRADA CASTANILOR NR.1 BLOC A2 SC.B AP.6	BRASOV	ROMANIA	500752
9	9 STRADA FAGETULUI NR.9	BUCURESTI	ROMANIA	010184
10	10 STRADA MARULUI NR.92	BUCURESTI	ROMANIA	010793

Curier

```

INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('ION', 'ALEXANDRU',
2800);

```

```

INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('POPESCU', 'MARIA',
3200);

INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('IONESCU',
'GEORGE', 2900);

INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('STANESCU', 'ANA',
3100);

INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('DOBRESCU',
'MIHAI', 2700);

INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('MARINESCU',
'ANDREI', 3300);

INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('ALEXANDRU',
'RADU', 3400);

INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('STEFANESCU',
'CRISTINA', 2800);

INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('IONESCU', 'ADINA',
3500);

INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('RADU', 'FLORIN',
3500);

INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('STEFANESCU',
'VLAD', 3600);

INSERT INTO CURIER (NUME, PRENUME, SALARIU) VALUES ('CONSTANTINESCU', 'MARIUS', 3700);

select * from curier;

```

Script Output x | Query Result x

SQL | All Rows Fetched: 12 in 0.08 seconds

	ID_CURIER	NUME	PRENUME	SALARIU
1	1	ION	ALEXANDRU	2800
2	2	POPESCU	MARIA	3200
3	3	IONESCU	GEORGE	2900
4	4	STANESCU	ANA	3100
5	5	DOBRESCU	MIHAI	2700
6	6	MARINESCU	ANDREI	3300
7	7	ALEXANDRU	RADU	3400
8	8	STEFANESCU	CRISTINA	2800
9	9	IONESCU	ADINA	3500
10	10	RADU	FLORIN	3500
11	11	STEFANESCU	VLAD	3600
12	12	CONSTANTINESCU	MARIUS	3700

Masina_Livrare

```
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE)
VALUES ('Mercedes-Benz Sprinter', 'B-123-ABC');
```

```
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE)
VALUES ('Ford Transit', 'BV-456-DEF');
```

```
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE)
VALUES ('Fiat Ducato', 'B-789-GHI');
```

```
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE)
VALUES ('Volkswagen Crafter', 'CT-101-JKL');
```

```
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE)
VALUES ('Renault Master', 'B-202-MNO');
```

```
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE)
VALUES ('Peugeot Boxer', 'GL-303-PQR');
```

```
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE)
VALUES ('Iveco Daily', 'B-404-STU');
```

```
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE)
VALUES ('Citroën Jumper', 'CT-505-VWX');
```

```
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE)
VALUES ('Opel Movano', 'B-606-YZA');
```

```
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE)
VALUES ('Nissan NV400', 'B-707-BCD');
```

```
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE)
VALUES ('Hyundai H350', 'BV-808-EFG');
```

```
INSERT INTO MASINA_LIVRARE (MODEL_MASINA, NUMAR_INMATRICULARE)
VALUES ('MAN TGE', 'CT-909-HIJ');
```

```
select * from masina_livrare;
```

ID_MASINA	MODEL_MASINA	NUMAR_INMATRICULARE
1	1 Mercedes-Benz Sprinter	B-123-ABC
2	2 Ford Transit	BV-456-DEF
3	3 Fiat Ducato	B-789-GHI
4	4 Volkswagen Crafter	CT-101-JKL
5	5 Renault Master	B-202-MNO
6	6 Peugeot Boxer	GL-303-PQR
7	7 Iveco Daily	B-404-STU
8	8 Citroën Jumper	CT-505-VWX
9	9 Opel Movano	B-606-YZA
10	10 Nissan NV400	B-707-BCD
11	11 Hyundai H350	BV-808-EFG
12	12 MAN TGE	CT-909-HIJ

Livrare

INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('CURIER', 12.90 , 'EFECTUATA', '12-MAY-2024', 1,2,7);

INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('EASYBOX',6.90 , 'EFECTUATA', '15-MAY-2024', 4,3,10);

INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('EASYBOX',6.90 , 'EFECTUATA', '15-MAY-2024', 4,3,10);

INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('CURIER', 16.99 , 'IN CURS DE LIVRARE', '20-MAY-2024', 9,1,5);

INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('FANBOX', 10 , 'EFECTUATA', '20-FEB-2024', 8,5,1);

INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('CURIER', 16.99 , 'EFECTUATA', '02-DEC-2023', 10,7,11);

INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('SAMEDAY', 0 , 'IN CURS DE PROCESARE', '17-MAY-2024', 7,10,2);

INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('CURIER', 19.99 , 'EFECTUATA', '16-APR-2024', 6,1,5);

INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('CURIER', 16.99 , 'IN CURS DE LIVRARE', '26-JUN-2023', 2,6,3);

INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('EASYBOX', 0 , 'IN CURS DE PROCESARE', '16-MAY-2024', 9,11,8);

INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('CURIER', 14.99, 'EFECTUATA', '30-MAR-2024', 5, 4, 12);

INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('FANBOX', 10.50, 'EFECTUATA', '22-MAR-2024', 3, 6, 4);

INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('CURIER', 19.99, 'IN CURS DE LIVRARE', '01-JUN-2024', 2, 12, 9);

INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('EASYBOX', 10 , 'EFECTUATA', '15-MAY-2024', 8,3,7);

SELECT * FROM LIVRARE;

ID_LIVRARE	METODA_LIVRARE	PRET_LIVRARE	STARE	DATA_LIVRARE	ID_ADRESA	ID_CURIER	ID_MASINA
1	1 CURIER	12.9	EFFECTUATA	12-MAY-24	1	2	7
2	2 EASYBOX	6.9	EFFECTUATA	15-MAY-24	4	3	10
3	3 EASYBOX	6.9	EFFECTUATA	15-MAY-24	4	3	10
4	4 CURIER	16.99	IN CURS DE LIVRARE	20-MAY-24	9	1	5
5	5 FANBOX	10	EFFECTUATA	20-FEB-24	8	5	1
6	6 CURIER	16.99	EFFECTUATA	02-DEC-23	10	7	11
7	7 SAMEDAY	0	IN CURS DE PROCESARE	17-MAY-24	7	10	2
8	8 CURIER	19.99	EFFECTUATA	16-APR-24	6	1	5
9	9 CURIER	16.99	IN CURS DE LIVRARE	26-JUN-23	2	6	3
10	10 EASYBOX	0	IN CURS DE PROCESARE	16-MAY-24	9	11	8
11	11 CURIER	14.99	EFFECTUATA	30-MAR-24	5	4	12
12	12 FANBOX	10.5	EFFECTUATA	22-MAR-24	3	6	4
13	13 CURIER	19.99	IN CURS DE LIVRARE	01-JUN-24	2	12	9

Comanda

INSERT INTO COMANDA (TOTAL_PLATA, STARE_COMANDA, DATA_COMANDA, ID_UTILIZATOR, ID_LIVRARE) VALUES(100,'EFFECTUATA','23-FEB-2023', 1,1);

INSERT INTO COMANDA (TOTAL_PLATA, STARE_COMANDA, DATA_COMANDA, ID_UTILIZATOR, ID_LIVRARE) VALUES(106.9,'IN CURS DE PREGATIRE','10-MAY-2024', 3,2);

INSERT INTO COMANDA (TOTAL_PLATA, STARE_COMANDA, DATA_COMANDA, ID_UTILIZATOR, ID_LIVRARE) VALUES(237.6,'IN CURS DE PREGATIRE','9-MAY-2024', 5,3);

INSERT INTO COMANDA (TOTAL_PLATA, STARE_COMANDA, DATA_COMANDA, ID_UTILIZATOR, ID_LIVRARE) VALUES(129.5,'IN CURS DE PREGATIRE','15-MAY-2024',6 ,10);

INSERT INTO COMANDA (TOTAL_PLATA, STARE_COMANDA, DATA_COMANDA, ID_UTILIZATOR, ID_LIVRARE) VALUES (68.4, 'EFFECTUATA', '23-FEB-2023', 1, 13);

INSERT INTO COMANDA (TOTAL_PLATA, STARE_COMANDA, DATA_COMANDA, ID_UTILIZATOR, ID_LIVRARE) VALUES (126.9, 'IN CURS DE PREGATIRE', '10-MAY-2024', 2, 12);

INSERT INTO COMANDA (TOTAL_PLATA, STARE_COMANDA, DATA_COMANDA, ID_UTILIZATOR, ID_LIVRARE) VALUES (117.6, 'IN CURS DE PREGATIRE', '9-MAY-2024', 3, 11);

INSERT INTO COMANDA (TOTAL_PLATA, STARE_COMANDA, DATA_COMANDA, ID_UTILIZATOR, ID_LIVRARE) VALUES (66.98, 'IN CURS DE PREGATIRE', '15-MAY-2024', 4, 4);

```

INSERT INTO COMANDA (TOTAL_PLATA, STARE_COMANDA, DATA_COMANDA,
ID_UTILIZATOR, ID_LIVRARE) VALUES (62.5, 'EFECTUATA', '20-FEB-2024', 6, 6);

INSERT INTO COMANDA (TOTAL_PLATA, STARE_COMANDA, DATA_COMANDA,
ID_UTILIZATOR, ID_LIVRARE) VALUES (216.98, 'EFECTUATA', '02-DEC-2023', 7, 7);

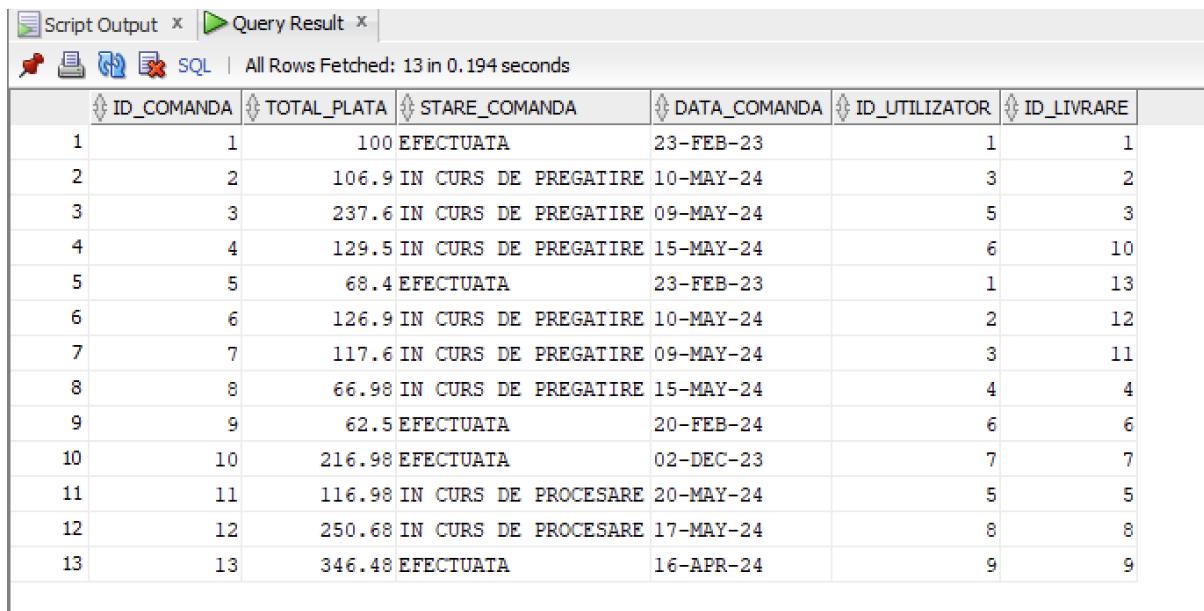
INSERT INTO COMANDA (TOTAL_PLATA, STARE_COMANDA, DATA_COMANDA,
ID_UTILIZATOR, ID_LIVRARE) VALUES (116.98, 'IN CURS DE PROCESARE', '20-
MAY-2024', 5, 5);

INSERT INTO COMANDA (TOTAL_PLATA, STARE_COMANDA, DATA_COMANDA,
ID_UTILIZATOR, ID_LIVRARE) VALUES (250.68, 'IN CURS DE PROCESARE', '17-
MAY-2024', 8, 8);

INSERT INTO COMANDA (TOTAL_PLATA, STARE_COMANDA, DATA_COMANDA,
ID_UTILIZATOR, ID_LIVRARE) VALUES (346.48, 'EFECTUATA', '16-APR-2024', 9, 9);

SELECT * FROM COMANDA;

```



The screenshot shows a database query results window with two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, displaying the following table:

ID_COMANDA	TOTAL_PLATA	STARE_COMANDA	DATA_COMANDA	ID_UTILIZATOR	ID_LIVRARE
1	1	100 EFECTUATA	23-FEB-23	1	1
2	2	106.9 IN CURS DE PREGATIRE	10-MAY-24	3	2
3	3	237.6 IN CURS DE PREGATIRE	09-MAY-24	5	3
4	4	129.5 IN CURS DE PREGATIRE	15-MAY-24	6	10
5	5	68.4 EFECTUATA	23-FEB-23	1	13
6	6	126.9 IN CURS DE PREGATIRE	10-MAY-24	2	12
7	7	117.6 IN CURS DE PREGATIRE	09-MAY-24	3	11
8	8	66.98 IN CURS DE PREGATIRE	15-MAY-24	4	4
9	9	62.5 EFECTUATA	20-FEB-24	6	6
10	10	216.98 EFECTUATA	02-DEC-23	7	7
11	11	116.98 IN CURS DE PROCESARE	20-MAY-24	5	5
12	12	250.68 IN CURS DE PROCESARE	17-MAY-24	8	8
13	13	346.48 EFECTUATA	16-APR-24	9	9

Card

```

INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator)
VALUES ('0000111122223333','EMMA MACIUCA', '123', '01-FEB-2026', 1);

INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator)
VALUES ('1800789243671234','IULIANA MARIN', '812', '01-MAR-2028', 6);

INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator)
VALUES ('8902489019003532','POPESCU ANDREEA IOANA', '602', '01-APR-2029', 2);

```

```

INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator)
VALUES ('2310649119007413','ION ANDREI', '241', '01-OCT-2025', 3);

INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator)
VALUES ('1475148418001975','ANDREESCU MARIA', '207', '01-NOV-2026', 4);

INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator)
VALUES ('3485120719001298','MACIUCA SARA', '109', '01-DEC-2029', 5);

INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator)
VALUES ('4723699318001431','POP DARIA', '192', '01-JAN-2030', 7);

INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator)
VALUES ('2367646716001239','MARES DARIUS', '276', '01-APR-2025', 9);

INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator)
VALUES ('6892163417001465','POPA MARA', '225', '01-MAY-2027', 8);

INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator)
VALUES ('1354235715003687','BALAN IOANA', '913', '01-JUL-2024', 10);

INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator)
VALUES ('1565611319003575','BALAN IOANA', '637', '01-DEC-2029', 10);

INSERT INTO CARD (numar_card, nume_propietar, CVV, data_expirare, id_utilizator)
VALUES ('8243186518006295','POPESCU ANDREEA IOANA', '265', '01-APR-2028', 2);

SELECT * FROM CARD;

```

Script Output x Query Result x

SQL | All Rows Fetched: 12 in 0.045 seconds

ID_CARD	NUMAR_CARD	NUME_PROPIETAR	CVV	DATA_EXPIRARE	ID_UTILIZATOR
1	1 0000111122223333	EMMA MACIUCA	123	01-FEB-26	1
2	2 1800789243671234	IULIANA MARIN	812	01-MAR-28	6
3	3 8902489019003532	POPESCU ANDREEA IOANA	602	01-APR-29	2
4	4 2310649119007413	ION ANDREI	241	01-OCT-25	3
5	5 1475148418001975	ANDREESCU MARIA	207	01-NOV-26	4
6	6 3485120719001298	MACIUCA SARA	109	01-DEC-29	5
7	7 4723699318001431	POP DARIA	192	01-JAN-30	7
8	8 2367646716001239	MARES DARIUS	276	01-APR-25	9
9	9 6892163417001465	POPA MARA	225	01-MAY-27	8
10	10 1354235715003687	BALAN IOANA	913	01-JUL-24	10
11	11 1565611319003575	BALAN IOANA	637	01-DEC-29	10
12	12 8243186518006295	POPESCU ANDREEA IOANA	265	01-APR-28	2

Plata

```
INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA)
VALUES(100,'23-FEB-2023','RESPINSA',1,1);

INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA)
VALUES(129.5,'15-MAY-2024','EFECTUATA',2,4);

INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA)
VALUES(106.9,'10-MAY-2024','EFECTUATA',4,2);

INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA)
VALUES(126.9,'10-MAY-2024','RESPINSA',3,6);

INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA)
VALUES(126.9,'10-MAY-2024','EFECTUATA',12,6);

INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA)
VALUES(237.6,'09-MAY-2024','EFECTUATA',6,3);

INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA)
VALUES(68.4,'23-FEB-2023','EFECTUATA',1,5);

INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA)
VALUES(117.6,'09-MAY-2024','IN CURS DE PROCESARE',4,7);

INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA)
VALUES(66.98,'15-MAY-2024','IN CURS DE PROCESARE',5,8);

INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA)
VALUES(62.5,'20-FEB-2024','EFECTUATA',2,9);

INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA)
VALUES(216.98,'02-DEC-2023','EFECTUATA',7,10);

INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA)
VALUES(116.98,'20-MAY-2024','RESPINSA',6,11);

INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA)
VALUES(250.68,'17-MAY-2024','IN CURS DE PROCESARE',9,12);

INSERT INTO PLATA (VALOARE, DATA_PLATA, STARE, ID_CARD, ID_COMANDA)
VALUES(346.48,'16-APR-2024','EFECTUATA',8,13);

select * from plata;
```

Script Output | Query Result | SQL | All Rows Fetched: 14 in 0.068 seconds

ID_PLATA	VALOARE	DATA_PLATA	STARE	ID_CARD	ID_COMANDA
1	1	100 23-FEB-23	RESPINSA	1	1
2	2	129.5 15-MAY-24	EFECTUATA	2	4
3	3	106.9 10-MAY-24	EFECTUATA	4	2
4	4	126.9 10-MAY-24	RESPINSA	3	6
5	5	126.9 10-MAY-24	EFECTUATA	12	6
6	6	237.6 09-MAY-24	EFECTUATA	6	3
7	7	68.4 23-FEB-23	EFECTUATA	1	5
8	8	117.6 09-MAY-24	IN CURS DE PROCESARE	4	7
9	9	66.98 15-MAY-24	IN CURS DE PROCESARE	5	8
10	10	62.5 20-FEB-24	EFECTUATA	2	9
11	11	216.98 02-DEC-23	EFECTUATA	7	10
12	12	116.98 20-MAY-24	RESPINSA	6	11
13	13	250.68 17-MAY-24	IN CURS DE PROCESARE	9	12
14	14	346.48 16-APR-24	EFECTUATA	8	13

Cos_Cumparaturi

```
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (1,55.50);
```

```
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (2,150);
```

```
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (3,220);
```

```
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (4,199.99);
```

```
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (5,202.48);
```

```
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (6,0);
```

```
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (7,129.5);
```

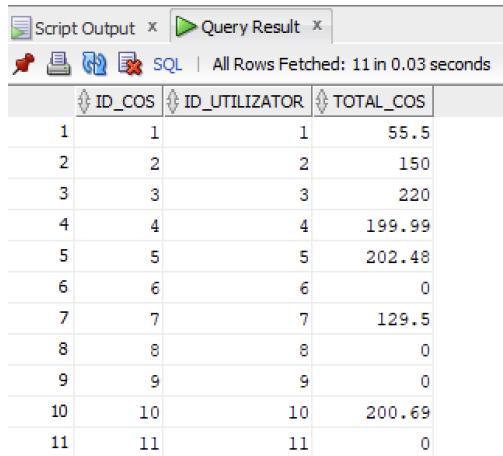
```
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (8,0);
```

```
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (9,0);
```

```
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (10,200.69);
```

```
INSERT INTO COS_CUMPARATURI (ID_UTILIZATOR, TOTAL_COS) VALUES (11,0);
```

```
SELECT * FROM COS_CUMPARATURI;
```



The screenshot shows a database query results window with the following details:

- Script Output tab is active.
- Query Result tab is also present.
- SQL icon is selected.
- Message: All Rows Fetched: 11 in 0.03 seconds
- Table structure:
 - Columns: ID_COS, ID_UTILIZATOR, TOTAL_COS
 - Row 1: 1, 1, 55.5
 - Row 2: 2, 2, 150
 - Row 3: 3, 3, 220
 - Row 4: 4, 4, 199.99
 - Row 5: 5, 5, 202.48
 - Row 6: 6, 6, 0
 - Row 7: 7, 7, 129.5
 - Row 8: 8, 8, 0
 - Row 9: 9, 9, 0
 - Row 10: 10, 10, 200.69
 - Row 11: 11, 11, 0

Lista_Dorinte

```
INSERT INTO LISTA_DORINTE (ID_UTILIZATOR, DENUMIRE_LISTA) VALUES (1,'PANTOFI');
```

```
INSERT INTO LISTA_DORINTE (ID_UTILIZATOR, DENUMIRE_LISTA) VALUES (1, 'CADOURI');
```

```
INSERT INTO LISTA_DORINTE (ID_UTILIZATOR, DENUMIRE_LISTA) VALUES (2, 'HAINA SPORT');
```

```
INSERT INTO LISTA_DORINTE (ID_UTILIZATOR, DENUMIRE_LISTA) VALUES (2, 'CASUAL');
```

```
INSERT INTO LISTA_DORINTE (ID_UTILIZATOR, DENUMIRE_LISTA) VALUES (3, 'OFFICE');
```

```
INSERT INTO LISTA_DORINTE (ID_UTILIZATOR, DENUMIRE_LISTA) VALUES (4, 'HAINA VACANTA');
```

```
INSERT INTO LISTA_DORINTE (ID_UTILIZATOR, DENUMIRE_LISTA) VALUES (5, 'HAINA TOAMNA');
```

```
INSERT INTO LISTA_DORINTE (ID_UTILIZATOR, DENUMIRE_LISTA) VALUES (5, 'HAINA VARA');
```

```
INSERT INTO LISTA_DORINTE (ID_UTILIZATOR, DENUMIRE_LISTA) VALUES (6, 'HAINA IARNA');
```

```
INSERT INTO LISTA_DORINTE (ID_UTILIZATOR, DENUMIRE_LISTA) VALUES (7, 'HAINA PRIMAVARA');
```

```

INSERT INTO LISTA_DORINTE (ID_UTILIZATOR, DENUMIRE_LISTA) VALUES (8, 'HAINE VARA');

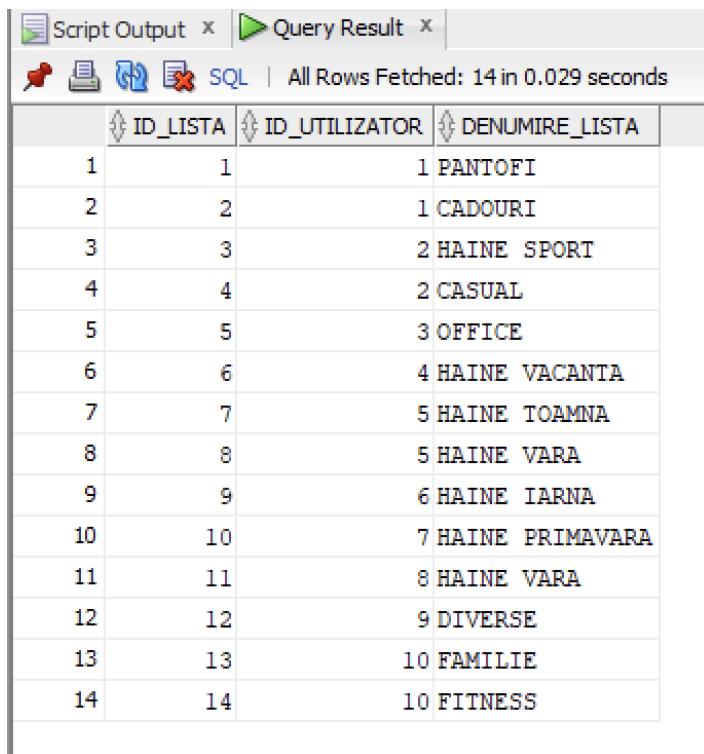
INSERT INTO LISTA_DORINTE (ID_UTILIZATOR, DENUMIRE_LISTA) VALUES (9, 'DIVERSE');

INSERT INTO LISTA_DORINTE (ID_UTILIZATOR, DENUMIRE_LISTA) VALUES (10, 'FAMILIE');

INSERT INTO LISTA_DORINTE (ID_UTILIZATOR, DENUMIRE_LISTA) VALUES (10, 'FITNESS');

SELECT * FROM LISTA_DORINTE;

```



ID_LISTA	ID_UTILIZATOR	DENUMIRE_LISTA
1	1	1 PANTOFI
2	2	1 CADOURI
3	3	2 HAINE SPORT
4	4	2 CASUAL
5	5	3 OFFICE
6	6	4 HAINE VACANTA
7	7	5 HAINE TOAMNA
8	8	5 HAINE VARA
9	9	6 HAINE IARNA
10	10	7 HAINE PRIMAVERA
11	11	8 HAINE VARA
12	12	9 DIVERSE
13	13	10 FAMILIE
14	14	10 FITNESS

Categorie

```

INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('TRICOU');

INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('PANTALONI');

INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('ROCHIE');

INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('GEACA');

INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('VESTA');

INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('CAMASA');

```

```

INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('GEANTA');

INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('OCHELARI DE SOARE');

INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('PULOVER');

INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('FUSTA');

INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('ESARFA');

INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('CACIULA');

INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('CACIULA');

INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('PAPUCI');

INSERT INTO CATEGORIE (DENUMIRE_CATEGORIE) VALUES ('RUCSAC');

SELECT * FROM CATEGORIE;

```

ID_CATEGORIE	DENUMIRE_CATEGORIE
1	1 TRICOU
2	2 PANTALONI
3	3 ROCHIE
4	4 GEACA
5	5 VESTA
6	6 CAMASA
7	7 GEANTA
8	8 OCHELARI DE SOARE
9	9 PULOVER
10	10 FUSTA
11	11 ESARFA

Produs

```

INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE)
VALUES ('TRICOU ALBASTRU',55.50,10,1);

INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE)
VALUES ('PANTALONI FORMALI',120.00,6,2);

INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE)
VALUES ('CAMASA DE IN',110.70,20,6);

INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE)
VALUES ('TRICOU GRI',49.99,25,1);

```

```
INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE)
VALUES ('ROCHIE DE PLAJA',99.99,10,3);

INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE)
VALUES ('OCHELARI DE SOARE COPII',52.50,5,8);

INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE)
VALUES ('GEACA DE IARNA',199.99,35,4);

INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE)
VALUES ('CAMASA VERDE',89.99,17,6);

INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE)
VALUES ('GEANTA DE UMAR',129.50,20,7);

INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE)
VALUES ('TRICOU IN DUNGI',49.99,15,1);

INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE)
VALUES ('VESTA ALBASTRA', 100 ,20,5);

INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE)
VALUES ('FUSTA MIDI', 75.00, 12, 10);

INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE)
VALUES ('PULOVER DE LANA', 150.00, 8, 9);

INSERT INTO PRODUS (DENUMIRE_PRODUS, PRET, STOC, ID_CATEGORIE)
VALUES ('ESARFA COLORATA', 30.00, 25, 11);

SELECT * FROM PRODUS;
```

The screenshot shows a SQL developer interface with two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, displaying a table with 14 rows of data. The columns are labeled: ID_PRODUS, DENUMIRE_PRODUS, PRET, STOC, and ID_CATEGORIE. The data represents various products with their descriptions, prices, stock levels, and category IDs.

ID_PRODUS	DENUMIRE_PRODUS	PRET	STOC	ID_CATEGORIE
1	1 TRICOU ALBASTRU	55.5	10	1
2	2 PANTALONI FORMALI	120	6	2
3	3 CAMASA DE IN	110.7	20	6
4	4 TRICOU GRI	49.99	25	1
5	5 ROCHIE DE PLAJA	99.99	10	3
6	6 OCHELARI DE SOARE COPIII	52.5	5	8
7	7 GEACA DE IARNA	199.99	35	4
8	8 CAMASA VERDE	89.99	17	6
9	9 GEANTA DE UMAR	129.5	20	7
10	10 TRICOU IN DUNGI	49.99	15	1
11	11 VESTA ALBASTRA	100	20	5
12	12 FUSTA MIDI	75	12	10
13	13 PULOVER DE LANA	150	8	9
14	14 ESARFA COLORATA	30	25	11

Produs_Comanda

```

INSERT INTO PRODUS_COMANDA (ID_PRODUS, ID_COMANDA) VALUES (2, 1);
INSERT INTO PRODUS_COMANDA (ID_PRODUS, ID_COMANDA) VALUES (11, 2);
INSERT INTO PRODUS_COMANDA (ID_PRODUS, ID_COMANDA) VALUES (2, 3);
INSERT INTO PRODUS_COMANDA (ID_PRODUS, ID_COMANDA) VALUES (3, 3);
INSERT INTO PRODUS_COMANDA (ID_PRODUS, ID_COMANDA) VALUES (9, 4);
INSERT INTO PRODUS_COMANDA (ID_PRODUS, ID_COMANDA) VALUES (1, 5);
INSERT INTO PRODUS_COMANDA (ID_PRODUS, ID_COMANDA) VALUES (2, 6);
INSERT INTO PRODUS_COMANDA (ID_PRODUS, ID_COMANDA) VALUES (3, 7);
INSERT INTO PRODUS_COMANDA (ID_PRODUS, ID_COMANDA) VALUES (4, 8);
INSERT INTO PRODUS_COMANDA (ID_PRODUS, ID_COMANDA) VALUES (6, 9);
INSERT INTO PRODUS_COMANDA (ID_PRODUS, ID_COMANDA) VALUES (7, 10);
INSERT INTO PRODUS_COMANDA (ID_PRODUS, ID_COMANDA) VALUES (5, 11);
INSERT INTO PRODUS_COMANDA (ID_PRODUS, ID_COMANDA) VALUES (8, 12);
INSERT INTO PRODUS_COMANDA (ID_PRODUS, ID_COMANDA) VALUES (3, 12);
INSERT INTO PRODUS_COMANDA (ID_PRODUS, ID_COMANDA) VALUES (10, 12);
INSERT INTO PRODUS_COMANDA (ID_PRODUS, ID_COMANDA) VALUES (9, 13);

```

```

INSERT INTO PRODUS_COMANDA (ID_PRODUS, ID_COMANDA) VALUES (7, 13);
SELECT * FROM PRODUS_COMANDA;

```

Script Output | Query Result | SQL | All Rows Fetched: 17 in 0.045 seconds

	ID_PRODUS	ID_COMANDA
1	1	5
2	2	1
3	2	3
4	2	6
5	3	3
6	3	7
7	3	12
8	4	8
9	5	11
10	6	9
11	7	10
12	7	13
13	8	12
14	9	4
15	9	13
16	10	12
17	11	2

Produs_Cos

```

INSERT INTO PRODUS_COS VALUES(1,1);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(13,2);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(2,3);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(11,3);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(7,4);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(4,5);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(5,5);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(6,5);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(9,7);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(3,10);
INSERT INTO PRODUS_COS (ID_PRODUS, ID_COS) VALUES(8,10);
SELECT * FROM PRODUS_COS;

```

The screenshot shows a MySQL Workbench interface with two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, displaying a table with the following data:

	ID_PRODUS	ID_COS
1	1	1
2	2	3
3	3	10
4	4	5
5	5	5
6	6	5
7	7	4
8	8	10
9	9	7
10	11	3
11	13	2

Produs_Lista

```

INSERT INTO PRODUS_LISTA (ID_PRODUS, ID_LISTA) VALUES (1, 1);
INSERT INTO PRODUS_LISTA (ID_PRODUS, ID_LISTA) VALUES (2, 1);
INSERT INTO PRODUS_LISTA (ID_PRODUS, ID_LISTA) VALUES (3, 2);
INSERT INTO PRODUS_LISTA (ID_PRODUS, ID_LISTA) VALUES (4, 3);
INSERT INTO PRODUS_LISTA (ID_PRODUS, ID_LISTA) VALUES (5, 4);
INSERT INTO PRODUS_LISTA (ID_PRODUS, ID_LISTA) VALUES (6, 4);
INSERT INTO PRODUS_LISTA (ID_PRODUS, ID_LISTA) VALUES (7, 5);
INSERT INTO PRODUS_LISTA (ID_PRODUS, ID_LISTA) VALUES (8, 6);
INSERT INTO PRODUS_LISTA (ID_PRODUS, ID_LISTA) VALUES (9, 6);
INSERT INTO PRODUS_LISTA (ID_PRODUS, ID_LISTA) VALUES (10, 7);
INSERT INTO PRODUS_LISTA (ID_PRODUS, ID_LISTA) VALUES (11, 7);
INSERT INTO PRODUS_LISTA (ID_PRODUS, ID_LISTA) VALUES (12, 9);
INSERT INTO PRODUS_LISTA (ID_PRODUS, ID_LISTA) VALUES (13, 10);
INSERT INTO PRODUS_LISTA (ID_PRODUS, ID_LISTA) VALUES (1, 11);

```

```

INSERT INTO PRODUS_LISTA (ID_PRODUS, ID_LISTA) VALUES (2, 12);
INSERT INTO PRODUS_LISTA (ID_PRODUS, ID_LISTA) VALUES (3, 13);
INSERT INTO PRODUS_LISTA (ID_PRODUS, ID_LISTA) VALUES (4, 14);
INSERT INTO PRODUS_LISTA (ID_PRODUS, ID_LISTA) VALUES (5, 14);
SELECT * FROM PRODUS_LISTA;

```

ID_PRODUS	ID_LISTA
1	1
2	11
3	1
4	12
5	2
6	13
7	3
8	14
9	4
10	14
11	4
12	5
13	6
14	6
15	7
16	7
17	9
18	10

6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul.

Cerinta:

Scrieți o procedura care reține într-un vector primii 3 curieri cei mai bine plătiți și afișeaza pe ecran numele și salariul acestora. În plus, procedura va stoca într-un tablou indexat id-ul curierului, vechimea calculată în funcție de data primei livrări și o lista a orașelor în care a efectuat livrări. În cazul în care nu există 3 curieri în tabel, se va afisa un mesaj de eroare.

Rezolvare:

```

create or replace procedure p_ex6
is
    --vector
    type curieri_max is varray(3) of curier%rowtype;
    t_curieri curieri_max:=curieri_max();
    verificare_curieri number;
    prea_putini_curieri exception;
    --tabel imbricat
    type tabel_orase is table of adresa.oras%type;

```

```

type info_curier is record (id_curier curier.id_curier%type,
    vechime number, orase tabel_orase);
--tabel indexat
type t_info_curier is table of info_curier index by PLS_INTEGER;
tabel_info_curieri t_info_curier;
vechime_luni number;
l_orase tabel_orase := tabel_orase();
begin
--verificam daca exista cel putin 3 curieri
select count(*)
into verificare_curieri
from curier;
if verificare_curieri < 3 then
    raise prea_putini_curieri;
end if;

-- verificare daca curieri are mai putin de 3 inregistrari
select *
bulk collect into t_curieri
from (select *
      from curier
      order by salariu desc)
where rownum <=3;

--tablou indexat cu record
dbms_output.put_line('Lista de curieri');
for i in 1..t_curieri.count loop
    dbms_output.put_line(t_curieri(i).nume||' '||t_curieri(i).prenume||' '||t_curieri(i).salariu);
    --id curier
    tabel_info_curieri(i).id_curier := t_curieri(i).id_curier;

    --vechime
    select trunc(months_between(sysdate,min(l.data_livrare)))
    into vechime_luni
    from curier c
    join livrare l on l.id_curier = c.id_curier
    where c.id_curier = t_curieri(i).id_curier;

    tabel_info_curieri(i).vechime := vechime_luni;

    --lista de orase

```

```

tabel_info_curieri(i).orase := tabel_orase();
for o in (select a.oras
            from adresa a
            join livrare l on l.id_adresa = a.id_adresa
            join curier c on l.id_curier = c.id_curier
            where c.id_curier = t_curieri(i).id_curier) loop

    tabel_info_curieri(i).orase.extend;
    tabel_info_curieri(i).orase(tabel_info_curieri(i).orase.count) := o.oras;
end loop;
end loop;

dbms_output.put_line(' ');
dbms_output.put_line('Informatii suplimentare despre curieri');
for i in 1..tabel_info_curieri.count loop
    dbms_output.put_line('Id_curier: ' || tabel_info_curieri(i).id_curier);
    dbms_output.put_line('Vechime: ' || tabel_info_curieri(i).vechime || ' luni');

    dbms_output.put_line('Orașele livrate:');
    for j in 1..tabel_info_curieri(i).orase.count loop
        dbms_output.put_line(' - ' || tabel_info_curieri(i).orase(j));
    end loop;
    dbms_output.put_line(' ');
end loop;

exception when prea_putini_curieri then dbms_output.put_line('Numarul de curieri este mai
mic decat 3'); end p_ex6;

begin p_ex6; end;

```

Worksheet | Query Builder

```
-->6
--Formulati in limbaj natural o problemă pe care să o rezolvati folosind un subprogram stocat
--independent care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul.

-- Scriptă o procedură care retine într-un vector primii 3 curieri cei mai bine platiti și afiseaza pe ecran numele și salariul acestora.
-- În plus, procedura va stoca într-un tabel indexat id-ul curierului, vechimea calculată în funcție de data primei livrări
-- și o lista a orașelor în care a efectuat livrări. În cazul în care nu există 3 curieri în tabel, se va afisa un mesaj de eroare.

@create or replace procedure p_ex6
is
--vector
type curieri_max is varray(3) of curier%rowtype;
t_curieri curieri_max:=curieri_max();
verificare_curieri number;
prea_putini_curieri exception;
--tabel indexat
type tabel_orase is table of adresa.orase%type;
type info_curier is record (id_curier curier.id_curier%type,
                           vechime number,
                           orase tabel_orase);
--tabel indexat
type t_info_curier is table of info_curier index by PLS_INTEGER;
tabel_info_curieri t_info_curier;
vechime_luni number;
l_orase tabel_orase := tabel_orase();
begin
    --verificam dacă există cel puțin 3 curieri
    select count(*)
    into verificare_curieri
    from curier;

    if verificare_curieri < 3 then
        raise prea_putini_curieri;
    end if;

    -- verificare dacă curieri sunt mai puțin de 3 înregistrari
    begin
        into vechime_luni
        from curier c
        join livrare l on l.id_curier = c.id_curier
        where c.id_curier = t_curieri(i).id_curier;

        tabel_info_curieri(i).vechime := vechime_luni;

        --listă de orașe
        tabel_info_curieri(i).orase := tabel_orase();
        for o in (select a.oras
                   from adresa a
                   join livrare l on l.id_adresa = a.id_adresa
                   join curier c on l.id_curier = c.id_curier
                   where c.id_curier = t_curieri(i).id_curier) loop

            tabel_info_curieri(i).orase.extend;
            tabel_info_curieri(i).orase(tabel_info_curieri(i).orase.count) := o.oras;
        end loop;
    end loop;

    dbms_output.put_line('');
    dbms_output.put_line('Informatii suplimentare despre curieri');
    for i in 1..tabel_info_curieri.count loop
        dbms_output.put_line('Id_curiere: ' || tabel_info_curieri(i).id_curier);
        dbms_output.put_line('Vechime: ' || tabel_info_curieri(i).vechime || ' luni');

        dbms_output.put_line('Orasele livrate:');
        for j in 1..tabel_info_curieri(i).orase.count loop
            dbms_output.put_line(' - ' || tabel_info_curieri(i).orase(j));
        end loop;
        dbms_output.put_line('');
    end loop;
    exception
        when prea_putini_curieri then
            dbms_output.put_line('Numarul de curieri este mai mic decât 3');
    end p_ex6;

begin
    p_ex6;
end;
```

Script Output | Task completed in 0.156 seconds

Procedure P_EX6 compiled

PL/SQL procedure successfully completed.

tutorial |

```
Lista de curieri
CONSTANTINESCU MARIUS 3700
STEFANESCU VLAD 3600
IONESCU ADINA 3500

Informatii suplimentare despre curieri
Id_curiere: 12
Vechime: 7 luni
Orasele livrate:
- BUCURESTI

Id_curiere: 11
Vechime: 9 luni
Orasele livrate:
- GALATI
- CONSTANTA
```

Worksheet | Query Builder

```
into vechime_luni
from curier c
join livrare l on l.id_curier = c.id_curier
where c.id_curier = t_curieri(i).id_curier;

tabel_info_curieri(i).vechime := vechime_luni;

--listă de orașe
tabel_info_curieri(i).orase := tabel_orase();
for o in (select a.oras
           from adresa a
           join livrare l on l.id_adresa = a.id_adresa
           join curier c on l.id_curier = c.id_curier
           where c.id_curier = t_curieri(i).id_curier) loop

    tabel_info_curieri(i).orase.extend;
    tabel_info_curieri(i).orase(tabel_info_curieri(i).orase.count) := o.oras;
end loop;
end loop;

dbms_output.put_line('');
dbms_output.put_line('Informatii suplimentare despre curieri');
for i in 1..tabel_info_curieri.count loop
    dbms_output.put_line('Id_curiere: ' || tabel_info_curieri(i).id_curier);
    dbms_output.put_line('Vechime: ' || tabel_info_curieri(i).vechime || ' luni');

    dbms_output.put_line('Orasele livrate:');
    for j in 1..tabel_info_curieri(i).orase.count loop
        dbms_output.put_line(' - ' || tabel_info_curieri(i).orase(j));
    end loop;
    dbms_output.put_line('');
end loop;
exception
when prea_putini_curieri then
    dbms_output.put_line('Numarul de curieri este mai mic decât 3');
end p_ex6;

begin
    p_ex6;
end;
```

Script Output | Task completed in 0.156 seconds

Procedure P_EX6 compiled

PL/SQL procedure successfully completed.

tutorial |

```
Lista de curieri
CONSTANTINESCU MARIUS 3700
STEFANESCU VLAD 3600
IONESCU ADINA 3500

Informatii suplimentare despre curieri
Id_curiere: 12
Vechime: 7 luni
Orasele livrate:
- BUCURESTI

Id_curiere: 11
Vechime: 9 luni
Orasele livrate:
- GALATI
- CONSTANTA
```

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

Cerinta:

Pentru toti utilizatorii care au plasat comenzi cu valoarea totala mai mare sau egala cu numarul dat ca parametru sa se afiseze lista tuturor produselor cumpărate. Daca exista mai putin de 3 clienti care respecta conditia, atunci se vor afisa produselor clientilor care au plasat cel putin o comanda in 2024. De asemenea, functia intoarce lista de utilizatori rezultati.

Rezolvare:

```
-- cursor_produse - cursor explicit parametrizat
```

```
-- cursor_utilizatori - cursor dinamic
```

```
--tabel imbricat pentru lista de utilizatori
```

```
create or replace type rez_utilizatori is table of varchar2(100);
```

```
create or replace function f_ex7 (valoare number)
```

```
return rez_utilizatori
```

```
is
```

```
type t_cursor is ref cursor return utilizator%rowtype;
```

```
cursor_utilizatori t_cursor;
```

```
nr_utilizatori number;
```

```
i utilizator%rowtype;
```

```
cursor cursor_produse(id_ut utilizator.id_utilizator%type) is
```

```

(select *

from produs p

join produs_comanda pc on pc.id_produs = p.id_produs

join comanda c on c.id_comanda = pc.id_comanda

where c.id_utilizator = id_ut);

tabel_utilizatori rez_utilizatori:=rez_utilizatori();

begin

select count(distinct u.id_utilizator)

into nr_utilizatori

from (select u1.id_utilizator id_utilizator,u1.nume nume,u1.prenume prenume,
sum(c1.total_plata) total

from utilizator u1

join comanda c1 on u1.id_utilizator = c1.id_utilizator

group by u1.id_utilizator,u1.nume,u1.prenume) u

join comanda c on c.id_utilizator = u.id_utilizator where total>=valoare;

if nr_utilizatori > 2 then

dbms_output.put_line('Sunt'||nr_utilizatori||' utilizatori cu totalul comezilor mai mare decat
'|| valoare);

open cursor_utilizatori for

select distinct u.id_utilizator, u.nume, u.prenume, u.email, u.parola, u.numar_telefon,
u.data_nastere

from (select u1.id_utilizator ,u1.nume ,u1.prenume , u1.email, u1.parola, u1.numar_telefon,
u1.numar_telefon, u1.data_nastere,sum(c1.total_plata) total

from utilizator u1

join comanda c1 on u1.id_utilizator = c1.id_utilizator

group by u1.id_utilizator, u1.nume, u1.prenume, u1.email, u1.parola,
u1.numar_telefon, u1.data_nastere) u

join comanda c on c.id_utilizator = u.id_utilizator

where total>=valoare;

else

dbms_output.put_line('Sunt mai putin de 3 utilizatori cu totalul comezilor mai mare decat
'||valoare);

```

```

dbms_output.put_line('Utilizatorii care au plasat cel putin o comanda in 2024: ');
open cursor_utilizatori for
select distinct u.*
from utilizator u
join comanda c on u.id_utilizator=c.id_utilizator
where to_number(to_char(data_comanda,'YYYY'))=2024;
end if;

loop
fetch cursor_utilizatori into i;
exit when cursor_utilizatori%notfound;
DBMS_OUTPUT.PUT_LINE(i.nume||' '||i.prenume);

tabel_utilizatori.extend;
tabel_utilizatori(tabel_utilizatori.count) := i.nume||' '||i.prenume;

for j in cursor_produse(i.id_utilizator) loop
    dbms_output.put_line(' * '||j.denumire_producator);
end loop;
end loop;
close cursor_utilizatori;
return tabel_utilizatori;

end f_ex7;

-- sunt mai mult de 3 utilizatori cu totalul comenzilor mai mare decat 200

declare
t_utilizatori rez_utilizatori:= rez_utilizatori();

begin
t_utilizatori := f_ex7(200);

dbms_output.put_line(");

dbms_output.put_line('Lista de utilizatori din blocul de apelare: ');

if t_utilizatori.count >0 then

    for i in 1..(t_utilizatori.count-1) loop

        dbms_output.put(t_utilizatori(i)||', ');

```

```

end loop;

dbms_output.put(t_utilizatori(t_utilizatori.count)||' ');

dbms_output.new_line;

else dbms_output.put_line('Nu exista utilizatori');

end if;

end;

-- sunt mai putin de 3 utilizatori cu totalul comenzilor mai mare decat 200

declare

t_utilizatori rez_utilizatori:= rez_utilizatori();

begin t_utilizatori := f_ex7(400);

dbms_output.put_line(");

dbms_output.put_line('Lista de utilizatori din blocul de apelare: ');

if t_utilizatori.count >0 then

for i in 1..(t_utilizatori.count-1) loop

dbms_output.put(t_utilizatori(i)||', ');

end loop;

dbms_output.put(t_utilizatori(t_utilizatori.count)||' ');

dbms_output.new_line;

else dbms_output.put_line('Nu exista utilizatori');

end if;

end;

```

```
-->-- pentru toti utilizatorii care au plasat comenzi in valoare de cel putin numarul dat ca parametru
-->-- sa se afiseze lista tuturor produselor cumparate
-->-- daca exista mai putin de 3 clienti care respecta conditia,
-->-- anunt si se vor afisa produsele clientilor ce au plasat cel putin o comanda in 2024
-->-- de asemenea, functia intocmeste lista de utilizatori rezultati

-->-- cursor_producere - cursor explicit parametrizat
-->-- cursor_utilizatori - cursor dinamic

-->-- tabel imbricat pentru lista de utilizatori
create or replace type res_utilizatori is table of varchar2(100);

-->-- create or replace function f_ex7 (valoare number)
-->-- return res_utilizatori;
-->-- type t_cursor is ref cursor return utilizator%rowtype;
-->-- cursor_utilizatori t_cursor;
-->-- nr_utilizatori number;
-->-- i utilizator%rowtype;
cursor cursor_producere(id_ut utilizator.id_utilizator%type) is
  (select *
   from produse p
   join produse_comanda pc on pc.id_produs = p.id_produs
   join comanda c on c.id_comanda = pc.id_comanda
   where c.id_utilizator = id_ut);
tabel_utilizatori res_utilizatori:=res_utilizatori();
begin
  select count(distinct u.id_utilizator)
  into nr_utilizatori
  from (select ul.id_utilizator,ul.nume,ul.prenume, sum(cl.total_piese) total
        from utilizatori ul
        join comanda cl on ul.id_utilizator = cl.id_utilizator
        group by ul.id_utilizator,ul.nume,ul.prenume) u
  join comanda c on ul.id_utilizator = u.id_utilizator
  where total=>valoare;
  where total=>valoare;
end;
-->-- select count(distinct u.id_utilizator)
-->-- into nr_utilizatori
-->-- from (select ul.id_utilizator,ul.nume,ul.prenume, sum(cl.total_piese) total
-->--          from utilizatori ul
-->--          join comanda cl on ul.id_utilizator = cl.id_utilizator
-->--          group by ul.id_utilizator,ul.nume,ul.prenume) u
-->-- join comanda c on ul.id_utilizator = u.id_utilizator
-->-- where total=>valoare;

Script Output X | Task completed in 0.296 seconds
type REZ_UTILIZATORI compiled

unction F_EX7 compiled

L/SQL procedure successfully completed.

-->-- sunt mai mult de 3 utilizatori cu totalul comenzi mai mare decat 200
-->-- declare
-->--   t_utilizatori res_utilizatori:= res_utilizatori();
-->-- begin
-->--   t_utilizatori := f_ex7(200);
-->--   dms_output.put_line('');
-->--   dms_output.put_line('Listele de utilizatori din blocul de apelare: ');
-->--   if t_utilizatori.count > 0 then
-->--     for i in 1..(t_utilizatori.count-1) loop
-->--       dms_output.put(t_utilizatori(i)||',');
-->--     end loop;
-->--     dms_output.put(t_utilizatori(t_utilizatori.count)||' ');
-->--     dms_output.new_line;
-->--   else
-->--     dms_output.put_line('Nu exista utilizatori');
-->--   end if;
-->-- end;
-->-- -- sunt mai putin de 3 utilizatori cu totalul comenzi mai mare decat 200
-->-- declare
-->--   t_utilizatori res_utilizatori:= res_utilizatori();
-->-- begin
-->--   t_utilizatori := f_ex7(400);
-->--   dms_output.put_line('');
-->--   dms_output.put_line('Listele de utilizatori din blocul de apelare: ');
-->--   if t_utilizatori.count > 0 then
-->--     for i in 1..(t_utilizatori.count-1) loop
-->--       dms_output.put(t_utilizatori(i)||',');
-->--     end loop;
-->--     dms_output.put(t_utilizatori(t_utilizatori.count)||' ');
-->--     dms_output.new_line;
-->--   else
-->--     dms_output.put_line('Nu exista utilizatori');
-->--   end if;
-->-- end;
-->-- Script Output X | Task completed in 0.184 seconds
-->-- unction F_EX7 compiled

L/SQL procedure successfully completed.

L/SQL procedure successfully completed.
```

8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele create. Tratați toate excepțiile care pot apărea, inclusiv excepțiile predefinite NO_DATA_FOUND și TOO_MANY_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

Cerinta:

Afisati detaliile masinii cu care s-au efectuat cele mai multe livrari pentru adresa al carui oras este dat ca parametru. In plus, pentru adresa modificati pretul tutror livrarilor efectuate aplicand o reducere de 10% si afisati pretul inainte de modificare. Tratati exceptiile in cazul in care:

- nu exista adresa pentru orasul dat
- sunt mai multe adrese pentru orasul dat
- nu exista livrari (si implicit o masina) pentru adresa gasita
- exista mai multe masini cu numarul maxim de livrari

Rezolvare:

```
create or replace type tabel_livrari_pret is table of number(10,2);
```

```
create or replace function f_ex8(v_oras adresa.oras%type)
```

```
return tabel_livrari_pret
```

```
is
```

```
info_masina masina_livrare%rowtype;
```

```
info_adresa adresa%rowtype;
```

```
nr_adrese number;
```

```
fara_adrese exception;
```

```
prea_multe_adrese exception;
```

```
pragma exception_init(fara_adrese,-20000);
```

```
pragma exception_init(pre_a_multe_adrese, -20001);
```

```
max_masina number;
```

```
verif_masini masina_livrare.id_masina%type;
```

```
t_pret tabel_livrari_pret := tabel_livrari_pret();
```

```
begin
```

```
--verificari adresa
```

```
select count(*)
```

```
into nr_adrese
```

```

from adresa
where oras = v_oras;

if nr_adrese = 0 then
    raise_application_error(-20000,'Nu existe adrese de livrare pentru acest oras');
elsif nr_adrese>1 then
    raise_application_error(-20001,'Exista prea multe adrese de livrare pentru acest oras');
else
    --adresa pentru care cautam masina
    select *
    into info_adresa
    from adresa
    where oras = v_oras;

    -- la adresa avem cel putin o livrare
    -- se poate declansa no_data_found
    select m.id_masina
    into verif_masini
    from masina_livrare m
    join livrare l on l.id_masina = m.id_masina
    join adresa a on l.id_adresa = a.id_adresa
    where a.strada = info_adresa.strada;

    --maximul de livrari cu o masina la adresa respectiva
    select max(nr)
    into max_masina
    from (select count(*) nr
          from masina_livrare m
          join livrare l on l.id_masina = m.id_masina
          join adresa a on l.id_adresa = a.id_adresa
          where a.strada = info_adresa.strada
          group by m.id_masina);

    select m.id_masina,m.model_masina,m.numar_inmatriculare
    into info_masina
    from masina_livrare m
    join livrare l on m.id_masina = l.id_masina
    join adresa a on l.id_adresa = a.id_adresa
    where a.strada = info_adresa.strada
    and (select count*)
         from livrare ll
         where ll.id_masina = m.id_masina and ll.id_adresa = a.id_adresa)= max_masina;

```

```

dbms_output.put_line('Masina cu modelul'||info_masina.model_masina|| si numarul de
inmatricularare '
|| info_masina.numar_inmatricularare || a efectuat'|| max_masina || la adresa '
    || info_adresa.strada|| ' din orasul'||info_adresa.oras);
select pret_livrare
bulk collect into t_pret
from livrare_copy
where id_adresa = info_adresa.id_adresa;

update livrare_copy
set pret_livrare = pret_livrare * 0.9
where id_adresa = info_adresa.id_adresa;

dbms_output.put_line('Reducerea s-a efectuat pentru'||t_pret.count|| livrari ');
dbms_output.put_line(' Pretul dupa reducere: ');

for c in (select *
           from livrare_copy
           where id_adresa = info_adresa.id_adresa) loop
    dbms_output.put_line('Livrarea cu id'||c.id_livrare|| are acum pretul'||c.pret_livrare);
end loop;
end if;
return t_pret;
exception
when fara_adrese then
    dbms_output.put_line(sqlerrm);
    return t_pret;
when prea_multe_adrese then
    dbms_output.put_line(sqlerrm);
    return t_pret;
when no_data_found then
    dbms_output.put_line('Nu exista livrari pentru aceasta adresa');
    return t_pret;
when too_many_rows then
    dbms_output.put_line('Exista mai multe masini cu numar maxim de livrari pentru aceasta
adresa');
    return t_pret;
end f_ex8;

--exemplu bun
declare
rez_pret tabel_livrari_pret := tabel_livrari_pret();
begin
rez_pret := f_ex8('GALATI');

```

```

if rez_pret.count >0 then
    dbms_output.put_line('Preturi inainte de reducere');
    for i in 1..rez_pret.count loop
        dbms_output.put_line(rez_pret(i));
    end loop;
else dbms_output.put_line(' ');
end if;
end;

--prea multe adrese
declare rez_pret tabel_livrari_pret := tabel_livrari_pret();
begin
rez_pret := f_ex8('CONSTANTA');
if rez_pret.count >0 then
    dbms_output.put_line('Preturi inainte de reducere');
    for i in 1..rez_pret.count loop
        dbms_output.put_line(rez_pret(i));
    end loop;
else dbms_output.put_line(' ');
end if;
end;

--nu exista adrese
declare
rez_pret tabel_livrari_pret := tabel_livrari_pret();
begin
rez_pret := f_ex8('CLUJ');
if rez_pret.count >0 then
    dbms_output.put_line('Preturi inainte de reducere');
    for i in 1..rez_pret.count loop
        dbms_output.put_line(rez_pret(i));
    end loop;
else dbms_output.put_line(' ');
end if;
end;

--nu exista livrari pentru adresa
declare
rez_pret tabel_livrari_pret := tabel_livrari_pret();
begin
rez_pret := f_ex8('BRAILA');
if rez_pret.count >0 then
    dbms_output.put_line('Preturi inainte de reducere');
    for i in 1..rez_pret.count loop
        dbms_output.put_line(rez_pret(i));
    end loop;
end if;
end;

```

```

end loop;
else dbms_output.put_line(' ');
end if;
end;

--mai multe masini cu numar maxim
declare
rez_pret tabel_livrari_pret := tabel_livrari_pret();
begin
rez_pret := f_ex8('BRASOV');
if rez_pret.count >0 then
    dbms_output.put_line('Preturi inainte de reducere');
    for i in 1..rez_pret.count loop
        dbms_output.put_line(rez_pret(i));
    end loop;
else dbms_output.put_line(' ');
end if;
end;

```

The screenshot shows the Oracle SQL Developer interface with two panes. The left pane displays the PL/SQL code for creating a function `f_ex8`. The right pane shows the results of executing the function for the city 'BRASOV'.

```

create or replace function f_ex8(v_oras adresă.oras%type)
return tabel_livrari_pret is
info_masina masina_livrare%rowtype;
id_masina masina_livrare.id_masina%type;
nr_adresă number;
fara_adresă exception;
prea_multe_adresă exception;
praga_exception_init(fara_adresă,-20000);
praga_exception_init(prea_multe_adresă,-20001);
max_masina number;
verificMasini masina_livrare.id_masina%type;
t_pret tabel_livrari_pret := tabel_livrari_pret();
begin
--verificari adresă
select count(*)
into nr_adresă
from adresă
where oras = v_oras;

```

Execution Results:

```

Masina cu modelul Renault Master si numarul de inmatriculare B-202-MHO a efectuat 1 la adresă
Rezultatul eșant pentru 1 livrari
Prețul după reducere
Livrarea cu nr. 8 are acum prețul 17.99
Preturi inainte de reducere
19.99

```

```

SQL Worksheet History | tutorial | Buffer Size:20000
Worksheet | Query Builder
--exemplu bun
declare
    res_pret tabel_livrari_pret := tabel_livrari_pret();
begin
    res_pret := f_ex8('GALATI');
    if res_pret.count > 0 then
        dbms_output.put_line('Preturi inainte de reducere');
        for i in 1..res_pret.count loop
            dbms_output.put_line(res_pret(i));
        end loop;
    else
        dbms_output.put_line(' ');
    end if;
end;

--prea multe adrese
declare
    res_pret tabel_livrari_pret := tabel_livrari_pret();
begin
    res_pret := f_ex8('CONSTANTA');
    if res_pret.count > 0 then
        dbms_output.put_line('Preturi inainte de reducere');
        for i in 1..res_pret.count loop
            dbms_output.put_line(res_pret(i));
        end loop;
    else
        dbms_output.put_line(' ');
    end if;
end;
|
--nu există adrese
declare
    res_pret tabel_livrari_pret := tabel_livrari_pret();
begin
    res_pret := f_ex8('CLUJ');
    if res_pret.count > 0 then
        dbms_output.put_line(' ');
    end if;
end;

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

Welcome Page | #AllSQLLog | tutorial | Connections | Compiler - Log | DBMS Output | Buffer Size:20000
Worksheet | Query Builder
--nu există adrese
declare
    res_pret tabel_livrari_pret := tabel_livrari_pret();
begin
    res_pret := f_ex8('CLUJ');
    if res_pret.count > 0 then
        dbms_output.put_line('Preturi inainte de reducere');
        for i in 1..res_pret.count loop
            dbms_output.put_line(res_pret(i));
        end loop;
    else
        dbms_output.put_line(' ');
    end if;
end;

--nu există livrari pentru adresa
declare
    res_pret tabel_livrari_pret := tabel_livrari_pret();
begin
    res_pret := f_ex8('BRAILA');
    if res_pret.count > 0 then
        dbms_output.put_line('Preturi inainte de reducere');
        for i in 1..res_pret.count loop
            dbms_output.put_line(res_pret(i));
        end loop;
    else
        dbms_output.put_line(' ');
    end if;
end;

--mai multe masini cu numar maxim
declare
    res_pret tabel_livrari_pret := tabel_livrari_pret();
begin
    res_pret := f_ex8('BRASOV');
    if res_pret.count > 0 then
        dbms_output.put_line('Preturi inainte de reducere');
        for i in 1..res_pret.count loop
            dbms_output.put_line(res_pret(i));
        end loop;
    else
        dbms_output.put_line(' ');
    end if;
end;

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

```

9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să aibă minim 2 parametri și să utilizeze într-o singură comandă SQL 5 dintre tabelele create. Definiți minim 2 excepții proprii, altele decât cele predefinite la nivel de sistem. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate.

Cerinta:

Scrieti o procedura care primeste 3 parametrii - un nume de categorie, un nume de produs si un numar. Procedura va returna prin ultimele 2 argumente cele mai populare produse din categoria data si numarul de aparitii , calculat in functie de numarul total de comenzi + numarul total din lista de dorinte + numarul total din cosul de cumparaturi. Daca exista mai multe produse cu numar maxim de aparitii se vor intoarce toate produsele. Ulterior, se va aplica o reducere de 10 lei pentru produsele populare care au pretul mai mare de 25 de lei.

Procedura afiseaza si o statistica a produselor din categoria respectiva: ziua in care au fost comandate cele mai multe produse din acea categorie (daca sunt mai multe zile se afiseaza toate), valoarea celei mai mari comenzi din care a facut parte un produs din acea categorie si media varstei persoanelor care au comandat produse din categoria

Tratati si urmatoarele exceptii:

- Daca categoria nu are nici un produs
- Daca categoria are un numarul maxim de aparitii ale produsului 0
- Daca categoria are 2 produse cu acelasi nume
- Daca stocul produsului este mai mare decat cerinta (nr CC + LD)
- Daca exista multe categorii cu acelasi nume
- Daca categoria nu exista

Rezolvare:

```
create or replace type t_produse is table of varchar2(100);
```

```
create or replace procedure p_ex9
    (nume_categoria IN categorie.denumire_categoria%type,
     nume_produse OUT t_produse,
     numar_produse OUT number)
```

```
is
```

```
produse t_produse := t_produse();
n_categ categorie.denumire_categoria%type;
nr_prod number;
max_aparitii number;
verif_duplicate number;
prod_pret produs.pret%type;
--exceptii
fara_produse exception;
pragma exception_init(fara_produse,-20002);
eroare_nume_produs exception;
pragma exception_init(eroare_nume_produs,-20003);
fara_aparitii exception;
pragma exception_init(fara_aparitii,-20004);
eroare_statistica exception;
pragma exception_init(eroare_statistica,-20005);
begin
```

```

-- aici se declaraza no_data_found in caz ca nu exista categorie cu acest nume
-- sau too_many_rows in caz ca exista mai multe categorii cu acest nume
select denumire_categorie
into n_categ
from categorie
where denumire_categorie = nume_categorie;
--verificam daca categoria are produse
select count(*)
into nr_prod
from produs p
join categorie c on c.id_categorie = p.id_categorie
where c.denumire_categorie = n_categ;
if nr_prod = 0 then
    raise_application_error(-20002,'Aceasta categorie nu are produse asociate');
end if;

```

--verificam daca exista mai multe produse cu acelasi nume in categoria data

```

select count(*)
into verif_duplicate
from (select p.denumire_produs, count(*) as nr_produse
      from produs p
      join categorie c on c.id_categorie = p.id_categorie
      where c.denumire_categorie = n_categ
      group by p.denumire_produs
      having count(*) >1);

```

if verif_duplicate > 0 then

```

    raise_application_error(-20003,'Exista mai multe produse cu acelasi nume');
end if;
```

--numarul maxim de aparitii al unui produs din categoria data

--se calculeaza in functie de de cate ori a fost comandat + in cate cosuri de cumparaturi apare
+ in cate liste de dorinte apare

```

select max(total)
into max_aparitii
from (select (count(distinct pc.id_comanda) + count(distinct pl.id_lista) + count(distinct
pcc.id_cos)) total,p.id_produs prod
      from produs p
      join categorie c on c.id_categorie = p.id_categorie
      left join produs_comanda pc on pc.id_produs = p.id_produs
      left join produs_lista pl on pl.id_produs = p.id_produs
      left join produs_cos pcc on pcc.id_produs = p.id_produs
      where c.denumire_categorie = n_categ
      group by p.id_produs);
```

```

group by p.id_produs);

if max_aparitii = 0 then
    raise_application_error(-20004,'Produsele din aceasta categorie nu au aparitii inca');
end if;

nume_produse := t_produse();
--produsele cele mai populare
select p.denumire_produs
bulk collect into nume_produse
from (select (count(distinct pc.id_comanda) + count(distinct pl.id_lista) + count(distinct
pcc.id_cos)) total,p.denumire_produs
      from produs p
      join categorie c on c.id_categorie = p.id_categorie
      left join produs_comanda pc on pc.id_produs = p.id_produs
      left join produs_lista pl on pl.id_produs = p.id_produs
      left join produs_cos pcc on pcc.id_produs = p.id_produs
      where c.denumire_categorie = n_categ
      group by p.denumire_produs) p
where p.total = max_aparitii;

-- se aplica reducere de 10 lei produselor populare
-- stim ca nu sunt 2 produse cu acelasi nume - am facut verificarea mai devreme
-- facem update la produs

numar_produse := max_aparitii;

for i in 1..nume_produse.count loop
    select pret
    into prod_pret
    from produs
    where denumire_produs = nume_produse(i);

    if prod_pret > 25 then
        update produs
        set pret = pret - 10
        where denumire_produs = nume_produse(i);
    end if;

    -- bloc de verificare stoc produs
    declare
        cerere number;
        stoc_limitat exception;

```

```

stoc_actual produs.stoc%type;
begin
    select (count(distinct pl.id_lista) + count(distinct pcc.id_cos))
    into cerere
    from produs p
    left join produs_lista pl on pl.id_produs = p.id_produs
    left join produs_cos pcc on pcc.id_produs = p.id_produs
    where p.denumire_produs = nume_produse(i)
    group by p.id_produs;

    select stoc
    into stoc_actual
    from produs
    where denumire_produs = nume_produse(i);

    if stoc_actual < cerere then
        raise stoc_limitat;
    end if;
exception
    when stoc_limitat then
        dbms_output.put_line('Atentie! Stocul produsului'|| nume_produse(i)||' este limitat');
    end;
end loop;

--statistica
--din statistica se propaga eroarea eroare_statistica care se trateaza in aceasta procedura
genereaza_statistica(n_categ);
exception
when no_data_found then
    dbms_output.put_line('Eroare'||sqlcode||': Nu exista categorii cu acest nume');
when too_many_rows then
    dbms_output.put_line('Eroare'||sqlcode||': Există prea multe categorii cu acest nume');
when fara_produse then
    dbms_output.put_line(sqlerrm);
when eroare_nume_produs then
    dbms_output.put_line(sqlerrm);
when fara_aparitii then
    dbms_output.put_line(sqlerrm);
when eroare_statistica then
    dbms_output.put_line(sqlerrm);
end;
create or replace procedure genereaza_statistica
    (nume_categ categorie.denumire_categorie%type)
is
zi_max t_produse := t_produse();

```

```

val_max number; medie_varsta number;
max_prod number;
id_categ categorie.id_categorie%type;
verif_prod number;
fara_cumparaturi exception;
cursor cursor_zile is
(select zi from
  (select to_char(c.data_comanda,'DAY') zi,count() nr
   from produs_comanda pc
   join produs p on pc.id_produs = p.id_produs
   join comanda c on c.id_comanda = pc.id_comanda
   where p.id_categorie = id_categ group by to_char(c.data_comanda,'DAY') having
count() = max_prod));
begin
--id categorie
select id_categorie
into id_categ
from categorie
where denumire_categorie = nume_categ;

--verificam daca au fost produse cumparate din aceasta categorie
--daca nu - se va trata exceptia in procedura principala
select count(*)
into verif_prod
from produs_comanda pc
join produs p on p.id_produs = pc.id_produs
where p.id_categorie = id_categ;

if verif_prod = 0 then
  raise fara_cumparaturi;
end if;

--calculam care e maximul de produse cumparate intr-o zi
select max(nr)
into max_prod
from (select count(*) nr
      from produs_comanda pc
      join produs p on pc.id_produs = p.id_produs
      join comanda c on c.id_comanda = pc.id_comanda
      where p.id_categorie = id_categ
      group by to_char(c.data_comanda,'DAY'));

dbms_output.put_line(' ');
dbms_output.put_line('-----Statistica cumparaturi: '|nume_categ||'-----');

```

```

--zilele cu cele mai multe comenzi
dbms_output.put_line('Au fost comandate'||max_prod|| produse in ziua de: ');
for i in cursor_zile loop
    dbms_output.put_line('*'||i.zi);
end loop;

--valoarea celei mai mari comenzi ce contine produse din categorie
select max(c.total_plata)
into val_max
from produs_comanda pc
join produs p on pc.id_produs = p.id_produs
join comanda c on c.id_comanda = pc.id_comanda
where p.id_categorie = id_categ;
dbms_output.put_line('Comanda cea mai mare din care a facut parte un produs este in valoare
de'||val_max|| lei');

--media varstei
select trunc(avg(to_number(to_char(sysdate, 'YYYY')) - to_number(to_char(u.data_nastere,
'YYYY'))))
into medie_varsta
from produs_comanda pc
join produs p on pc.id_produs = p.id_produs
join comanda c on c.id_comanda = pc.id_comanda
join utilizator u on u.id_utilizator = c.id_utilizator
where p.id_categorie = id_categ;
dbms_output.put_line('Media varstei utilizatorilor care au cumparat produse din categoria
este de'||medie_varsta);

exception
when fara_cumparaturi then
    raise_application_error(-20005,'Nu se poate realiza statistica - nu au fost cumparate
produse din aceasta categoria');
end;

--no_data_found
declare
v_prod t_produse:=t_produse();
nr number;
begin
p_ex9('FULAR',v_prod,nr);
if v_prod is not null then
    dbms_output.put_line(' ');

```

```

dbms_output.put_line('Numarul maxim de aparitii al unui produs este'|| nr || si produsele
cele mai populare sunt: ');
for i in 1..v_prod.count loop
    dbms_output.put_line(' *'|| v_prod(i));
end loop;
end if;
end;

--too_many_rows
declare
v_prod t_produse:=t_produse();
nr number;
begin
p_ex9('CACIULA',v_prod,nr);
if v_prod is not null then
    dbms_output.put_line(' ');
    dbms_output.put_line('Numarul maxim de aparitii al unui produs este'|| nr || si produsele
cele mai populare sunt: ');
    for i in 1..v_prod.count loop
        dbms_output.put_line(' *'|| v_prod(i));
    end loop;
end if;
end;

--fara_produse
declare
v_prod t_produse:=t_produse();
nr number;
begin
p_ex9('PALARIE',v_prod,nr);
if v_prod is not null then
    dbms_output.put_line(' ');
    dbms_output.put_line('Numarul maxim de aparitii al unui produs este'|| nr || si produsele
cele mai populare sunt: ');
    for i in 1..v_prod.count loop
        dbms_output.put_line(' *'|| v_prod(i));
    end loop;
end if;
end;

--produse cu acelasi nume
declare
v_prod t_produse:=t_produse();
nr number;
begin

```

```

p_ex9('VESTA',v_prod,nr);
if v_prod is not null then
    dbms_output.put_line(' ');
    dbms_output.put_line('Numarul maxim de aparitii al unui produs este'|| nr ||' si produsele
cele mai populare sunt: ');
    for i in 1..v_prod.count loop
        dbms_output.put_line(' *'|| v_prod(i));
    end loop;
end if;
end;

--produsele nu au aparitii
declare
v_prod t_produse:=t_produse();
nr number;
begin
p_ex9('PAPUCI',v_prod,nr);
if v_prod is not null then
    dbms_output.put_line(' ');
    dbms_output.put_line('Numarul maxim de aparitii al unui produs este'|| nr ||' si produsele
cele mai populare sunt: ');
    for i in 1..v_prod.count loop
        dbms_output.put_line(' *'|| v_prod(i));
    end loop;
end if;
end;

-- OK prod cu id 1 si 4 au acelasi nr de aparitii din categoria TRICOU
-- tricou albastru are stoc redus
declare
v_prod t_produse:=t_produse();
nr number;
begin
p_ex9('TRICOU',v_prod,nr);
if v_prod is not null then
    dbms_output.put_line(' ');
    dbms_output.put_line('Numarul maxim de aparitii al unui produs este'|| nr ||' si produsele
cele mai populare sunt: ');
    for i in 1..v_prod.count loop
        dbms_output.put_line(' *'|| v_prod(i));
    end loop;
end if;
end;

-- eroare genereaza_statistica

```

```

declare
v_prod t_produse:=t_produse();
nr number;
begin
p_ex9('RUCSAC',v_prod,nr);
if v_prod is not null then
    dbms_output.put_line(' ');
    dbms_output.put_line('Numarul maxim de aparitii al unui produs este'|| nr ||' si produsele
cele mai populare sunt: ');
    for i in 1..v_prod.count loop
        dbms_output.put_line('*'|| v_prod(i));
    end loop;
end if;
end;

```

```

-- Script o procedura care primeste 3 parametrii - un nume de categorie, un nume de produs si un numar.
-- procedura va returna prin ultimele 2 argumente cele mai populare produse din categoria data si numarul de aparitii
-- (nr total de comenzi + nr total din LD + nr total din CC)
-- Daca exista mai multe produse cu numar maxim de aparitii se vor intoarce toate produsele
-- Ulterior, se va aplica o reducere de 10 lei pentru produsele populare care au pretul mai mare de 25 de lei

-- Procedura afiseaza si o statistica a produselor din categoria respectiva:
-- ziua in care au fost comandate cele mai multe produse din acea categorie - daca sunt mai multe zile se afiseaza toate
-- valoarea celei mai mari comenzi din care a facut parte un produs din acea categorie
-- media varstei persoanelor care au comandat produse din categoria
-- pt afisare statistica - procedura noua care se apeleaza din interiorul procedurii mari
-- sa fie stocate intr-o colectie sau ceva poate?

-- Tratati exceptiile:
-- Daca categoria nu are nici un produs
-- Daca categoria are un numarul maxim de aparitii ale produsului 0
-- Daca categoria are 2 produse cu acelasi nume
-- Daca stocul produsului este mai mare decat cerinta (nr CC + LD) se va declansa eroarea stoc_limitat
-- Daca exista multe categorii cu acelasi nume
-- Daca categoria nu exista

create or replace type t_produse is table of varchar2(100);

create or replace procedure p_ex9
    (nume_categoria IN categorie.denumire_categoria%type,
     nume_produse OUT t_produse,
     numar_produse OUT number)
is
    produse t_produse := t_produse();
    n_categ categorie.denumire_categoria%type;
    nr_prod number;
    max_aparitii number;
    varif_duplicata NUMBER;

```

Script Output X | Task completed in 0.206 seconds

Type T_PRODUSE compiled

Procedure GENEREAZA_STATISTICA compiled

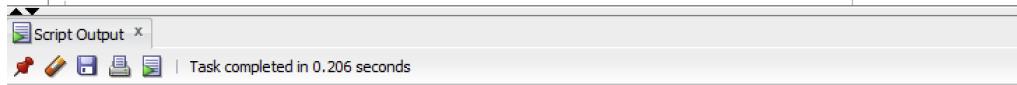
Procedure P_EX9 compiled

```

create or replace procedure genereaza_statistica
    (nume_categ categorie.denumire_categoria&type)
is
    zi_max_t_produse := t_produse();
    val_max number;
    medie_varsta number;
    max_prod number;
    id_categ categorie.id_categoria&type;
    verif_prod number;
    fara_cumparaturi exception;
    cursor cursor_zile is
        (select zi
         from (select to_char(c.data_comanda,'DAY') zi, count(*) nr
               from produs_comanda pc
               join produs p on pc.id_produs = p.id_produs
               join comanda c on c.id_comanda = pc.id_comanda
               where p.id_categoria = id_categ
               group by to_char(c.data_comanda,'DAY')
               having count(*) = max_prod));
begin
    --id categorie
    select id_categoria
    into id_categ
    from categorie
    where denumire_categoria = nume_categ;

    --verificam daca au fost produse cumparate din aceasta categorie
    --daca nu - se va trata exceptia in procedura principală
    select count(*)
    into verif_prod
    from produs_comanda pc

```



Type T_PRODUSE compiled

Procedure GENEREAZA_STATISTICA compiled

Procedure P_EX9 compiled

-- apelul procedurii

```

end;

--no_data_found
declare
    v_prod_t_producere:=t_produse();
    nr number;
begin
    p_ex9('FULAR',v_prod,nr);
    if v_prod is not null then
        dmbs_output.put_line(' ');
        dmbs_output.put_line('Numarul maxim de aparitii al unui produs este'|| nr ||' si produsele cele mai populare sunt: ');
        for i in 1..v_prod.count loop
            dmbs_output.put_line(' '|| v_prod(i));
        end loop;
    end if;
end;

--too_many_rows
declare
    v_prod_t_producere:=t_produse();
    nr number;
begin
    p_ex9('CACIULA',v_prod,nr);
    if v_prod is not null then
        dmbs_output.put_line(' ');
        dmbs_output.put_line('Numarul maxim de aparitii al unui produs este'|| nr ||' si produsele cele mai populare sunt: ');
        for i in 1..v_prod.count loop
            dmbs_output.put_line(' '|| v_prod(i));
        end loop;
    end if;
end;

--fara_produse
declare
    v_prod_t_producere:=t_produse();

```

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

Pretul produselor inainte de reducere:

Script Output | Query Result

All Rows Fetched: 17 in 0.04 seconds

	ID_PRODUS	DENUMIRE_PRODUS	PRET	STOC	ID_CATEGORIE
1	1	TRICOU ALBASTRU	55.5	2	1
2	2	PANTALONI FORMALI	120	6	2
3	3	CAMASA DE IN	110.7	20	6
4	4	TRICOU GRI	49.99	25	1
5	5	ROCHIE DE PLAJA	99.99	10	3
6	6	OCHELARI DE SOARE COPII	52.5	5	8

Pretul produselor după reducere:

	ID_PRODUS	DENUMIRE_PRODUS	PRET	STOC	ID_CATEGORIE
1	1	TRICOU ALBASTRU	45.5	2	1
2	2	PANTALONI FORMALI	120	6	2
3	3	CAMASA DE IN	110.7	20	6
4	4	TRICOU GRI	39.99	25	1
5	5	ROCHIE DE PLAJA	99.99	10	3

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

Cerinta:

Definiți un trigger care limitează curierii la a efectua maxim 5 livrări pe an. În cazul în care se fac mai multe de 5 livrări se va genera o eroare (la insert sau update). De asemenea, afișați un mesaj când un curier este aproape de limita de livrări.

```

-- pachet care tine cont de curieri si de numarul de livrari anuale

create or replace package contor_livrari
as
contor number := 0;
type curier_livrari is record
    (id_livrare.id_curier%type,
     nr_livrari number);
type tabel_curier_livrari is table of curier_livrari index by pls_integer;
t_curier tabel_curier_livrari;
end;
alter trigger t_ex10 enable;

-- trigger-ul la nivel de comanda
-- acesta se executa primul conform regulii de ordine a executiei triggerilor
create or replace trigger t_ex10
before insert or update of id_curier on livrare
begin
contor_livrari.contor := 0;
select id_curier, count(*)
bulk collect into contor_livrari.t_curier
from livrare
where extract(year from data_livrare) = 2024
group by id_curier;
-- procedura cu alerta pentru curierii care au efectuat deja numarul maxim de livrari
atentionare_curieri;
end;

-- se executa dupa triggerul t_ex10
create or replace trigger max_livrari
before insert or update of id_curier on livrare
for each row
begin
for i in 1..contor_livrari.t_curier.count loop
    if verif_max_livrari(i,:new.id_curier) then
        raise_application_error(-20000,'Curierul a efectuat deja 5 livrari in acest an!');
    end if;
end loop;
contor_livrari.contor := contor_livrari.contor+1;
end;

--functie care se apaleaza din max_livrari
create or replace function verif_max_livrari
    (nr number, id_livrare.id_curier%type)
return boolean

```

```

is
begin
if contor_livrari.t_curier(nr).id = id and contor_livrari.contor +
contor_livrari.t_curier(nr).nr_livrari = 5 then
    return true;
else
    return false;
end if;
end;

--procedura care notifica utilizatorul inainte sa introduca o livrare in privinta curierilor care
au nr max de livrari
create or replace procedure atentionare_curieri
is
type tabel_max is table of livrare.id_curier%type index by pls_integer;
t_max tabel_max;
begin
select id_curier
bulk collect into t_max
from livrare
where extract(year from data_livrare) = 2024
group by id_curier having count(*) = 5;

if t_max.count >0 then
    dbms_output.put_line('Atentie! Urmatorii curieri au efectuat maximul de livrari');
    for i in 1..t_max.count loop
        dbms_output.put_line(' * '||t_max(i));
    end loop;
end if;
end;

select * from livrare;

--merge

INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa,
id_curier, id_masina) VALUES ('CURIER',15.90 , 'EFECTUATA', '23-MAY-2024', 4,3,10);

--merge

INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa,
id_curier, id_masina) VALUES ('CURIER',15.90 , 'EFECTUATA', '23-MAY-2024', 4,3,10);

-- merge + atentionare

```

```
INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('CURIER',15.90 , 'EFFECTUATA', '23-MAY-2024', 4,5,10);
```

```
--nu merge
```

```
INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina) VALUES ('CURIER',15.90 , 'EFFECTUATA', '23-MAY-2024', 4,3,10);
```

```
alter trigger t_ex10 disable;
```

```
rollback;
```

The screenshot shows the Oracle SQL Developer interface with several windows open:

- Script Output X**: Shows the SQL code for creating package `CONTOR_LIVRARI`, procedure `ATENTIUNE_CURIERI`, function `VERIF_MAX_LIVRARI`, and trigger `T_EX10`.
- Package CONTOR_LIVRARI compiled**: Confirmation message.
- Procedure ATENTIUNE_CURIERI compiled**: Confirmation message.
- Function VERIF_MAX_LIVRARI compiled**: Confirmation message.
- Trigger T_EX10 compiled**: Confirmation message.
- SQL Worksheet History**: Shows the executed SQL code. It includes logic to check if a cursor has delivered more than 5 books, output an alert message, and then insert two new records into the `LIVRARE` table. The first record is inserted successfully, and the second record fails due to the trigger being disabled.
- tutorial x**: A note window titled "Atentie! Urmatorii curieri au efectuat maximul de livrari" with the value "3".
- Script Output X**: Shows the result of the query: "1 row inserted." followed by two "1 row inserted." messages, indicating the successful insertion of the first record and the failure of the second record.

```

--nu merge
INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina)
VALUES ('CURIER',15.90 , 'EFFECTUATA', '23-MAY-2024', 4,3,10);

alter trigger t_ex10 disable;

rollback;

```

Script Output x Query Result x
Task completed in 0.126 seconds

1 row inserted.

Error starting at line : 877 in command -
 INSERT INTO LIVRARE (metoda_livrare, pret_livrare, stare, data_livrare, id_adresa, id_curier, id_masina)
 VALUES ('CURIER',15.90 , 'EFFECTUATA', '23-MAY-2024', 4,3,10)
 Error report -
 ORA-20000: Curierul a efectuat deja 5 livrari in acest an!
 ORA-06512: at "UTILIZATOR.MAX_LIVRARI", line 4
 ORA-04088: error during execution of trigger 'UTILIZATOR.MAX_LIVRARI'

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

Cerinta:

Definiți un trigger la nivel de linie care updatează coloana stoc a tabelului produs și coloana total_cos a tabelului cos_cumparaturi atunci când un utilizator adaugă, sterge sau modifică o înregistrare din cosul de cumpăraturi.

Rezolvare:

```
alter trigger t_ex11 enable;
```

```
--trigger care se declanșează la adăugarea, stergerea sau modificarea unei linii din produs_cos
create or replace trigger t_ex11
after insert or delete or update on produs_cos
for each row
declare
  pret_produs number;
begin
  if inserting then
    select pret
```

```

into pret_produs
from produs
where id_produs = :new.id_produs;
modifica_total_cos(:new.id_cos,pret_produs);
modifica_stoc(:new.id_produs,-1);
elsif deleting then
    select pret
    into pret_produs
    from produs
    where id_produs = :old.id_produs;

    modifica_total_cos(:old.id_cos,-1*pret_produs);
    modifica_stoc(:old.id_produs,1);
else
    --daca se modifica cosul atunci se scade valoarea din cosul vechi si se muta in cosul nou
    --altfel daca se modifica produsul se scade din stocul produsului nou si se adauga in stocul
    produsului vechi
    if :new.id_cos <> :old.id_cos then
        select pret
        into pret_produs
        from produs
        where id_produs = :new.id_produs;

        modifica_total_cos(:old.id_cos,-1*pret_produs);
        modifica_total_cos(:new.id_cos,pret_produs);
    else
        select pret
        into pret_produs
        from produs
        where id_produs = :old.id_produs;

        modifica_stoc(:old.id_produs,1);
        modifica_total_cos(:old.id_cos,-1*pret_produs); -- se scade produsul vechi

        select pret
        into pret_produs
        from produs
        where id_produs = :new.id_produs;

        modifica_total_cos(:new.id_cos,pret_produs); --se aduga produsul nou
        modifica_stoc(:new.id_produs,-1);

    end if;
end if;

```

```

end;

alter trigger stoc_redus enable;

-- trigger care afiseaza o notificare atunci cand stocul e mai mic decat cerinta
create or replace trigger stoc_redus
after update of stoc on produs
for each row
begin
if :new.stoc < 3 then
    dbms_output.put_line('Atentie! Stocul acestui produs este redus');
end if;
end;
alter trigger stoc_redus disable;

--procedura care modifica totalul cosului de cumparaturi
create or replace procedure modifica_total_cos
(cos cos_cumparaturi.id_cos%type,
 valoare number)
is
begin
update cos_cumparaturi
set total_cos = total_cos + valoare
where id_cos = cos;
end;

create or replace procedure modifica_stoc
(produs produs.id_produs%type,
 cantitate number)
is
begin
update produs
set stoc = stoc + cantitate
where id_produs = produs;
end;
select * from cos_cumparaturi;
select * from produs_cos;
select * from produs order by id_produs;
--valoarea s-a modificat in urma exercitiilor anterioare
update cos_cumparaturi
set total_cos = 55.5
where id_cos = 1;

delete from produs_cos
where id_cos = 1;
insert into produs_cos values (1,1);

```

```

update produs_cos
set id_produs = 2
where id_produs = 1 and id_cos = 1;
alter trigger t_ex11 disable;
rollback;

```

```

--11
--Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.
-- de pus mesaje sa demonstrezi ca merge??

-- trigger la nivel de linie
-- Definiți un trigger la nivel de linie care updateaza coloana stoc a tabelului produs
-- si coloana total_cos a tabelului cos_cumparaturi atunci cand un utilizator adauga, sterge sau modifica o inregistrare din cosul de cumparaturi

-- cand se updateaza - aplică reducere la cos - reducerea nu poate varia mai mult de 20% din totalul cosului

alter trigger t_ex11 enable;
--trigger care se declansează la adăugarea, stergerea sau modificarea unei linii din produs_cos
create or replace trigger t_ex11
after insert or delete or update on produs_cos
for each row
declare
    pret_produs number;
begin
    if inserting then
        select pret
        into pret_produs
        from produs
        where id_produs = :new.id_produs;

        modifica_total_cos(:new.id_cos,pret_produs);
        modifica_stoc(:new.id_produs,-1);
    elsif deleting then
        select pret
        into pret_produs
        from produs

```

Script Output | Query Result | Task completed in 0.11 seconds

Procedure MODIFICA_STOC compiled

Procedure MODIFICA_TOTAL_COS compiled

Trigger STOC_REDUS compiled

Trigger T_EX11 compiled

Inainte de modificari:

ID_COS	ID_UTILIZATOR	TOTAL_COS		
1	1	55.5		
ID_PRODUS	DENUMIRE_PRODUS	PRET	STOC	ID_CATEGORIE
1	1 TRICOU ALBASTRU	55.5	2	1
2	2 PANTALONI FORMAT	120	6	2

Stergere

```

| delete from produs_cos
| where id_cos = 1;

```

Script Output x | Query Result x

SQL | All Rows Fetched: 11 in 0.014 seconds

ID_COS	ID_UTILIZATOR	TOTAL_COS
1	1	0
2	2	150

ID_PRODUS	DENUMIRE_PRODUS	PRET	STOC	ID_CATEGORIE
1	1 TRICOU ALBASTRU	55.5	3	1

Inserare

SQL Worksheet History

Query Find Aa Buffer Size: 20000

Worksheet Query Builder

```

| delete from produs_cos
| where id_produs = produs;
| end;

| select * from cos_comparaturi;
| select * from produs_cos;
| select * from produs order by id_produs;

--valoarea s-a modificarat in urma exercitiilor anterioare
| update cos_comparaturi
| set total_cos = 55.5
| where id_cos = 1;

| delete from produs_cos
| where id_cos = 1;

| insert into produs_cos values (1,1);

| update produs_cos_copy
| set id_produs = 2
| where id_produs = 1 and id_cos = 1;

| alter trigger t_ex11 disable;
| rollback;

```

Script Output x | Query Result x

Atentie! Stocul acestui produs este redus

1 row deleted.

1 row inserted.

Rollback complete.

1 row deleted.

1 row inserted.

Acti
Go to

```

| insert into produs_cos values (1,1);

Script Output x Query Result x
SQL | All Rows Fetched: 18 in 0.024 seconds


| ID_PRODUS | DENUMIRE_PRODUS     | PRET | STOC | ID_CATEGORIE |
|-----------|---------------------|------|------|--------------|
| 1         | 1 TRICOU ALBASTRU   | 55.5 | 2    | 1            |
| 2         | 2 PANTALONI FORMALI | 120  | 6    | 2            |


| insert into produs_cos values (1,1);

```

```

Script Output x Query Result x
SQL | All Rows Fetched: 11 in 0.023 seconds


| ID_COS | ID_UTILIZATOR | TOTAL_COS |
|--------|---------------|-----------|
| 1      | 1             | 55.5      |


```

Modificare

```

update produs_cos
set id_produs = 2
where id_produs = 1 and id_cos = 1;

Script Output x Query Result x
SQL | All Rows Fetched: 11 in 0.011 seconds


| ID_COS | ID_UTILIZATOR | TOTAL_COS |
|--------|---------------|-----------|
| 1      | 1             | 120       |


```

```

update produs_cos
set id_produs = 2
where id_produs = 1 and id_cos = 1;

Script Output x Query Result x
SQL | All Rows Fetched: 18 in 0.039 seconds


| ID_PRODUS | DENUMIRE_PRODUS     | PRET | STOC | ID_CATEGORIE |
|-----------|---------------------|------|------|--------------|
| 1         | 1 TRICOU ALBASTRU   | 55.5 | 3    | 1            |
| 2         | 2 PANTALONI FORMALI | 120  | 5    | 2            |


```

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

Cerinta:

Scripti un trigger care permite utilizatorului sa creeze doar functii, proceduri, triggeri sau pachete. In caz ca acesta incearca sa stearga ,sa modifice sau sa creeze tabele acesta va primi un mesaj de eroare. In plus, scrieti si o functie care permite dezactivarea si activarea trigger-ului creat.

Rezolvare:

```
begin
    activare_t_ex12('activare');
end;
create or replace trigger t_ex12
before create or alter or drop on schema
declare
    nume varchar2(100);
    tip varchar2(100);
begin
    nume := sys.dictionary_obj_name;
    tip := sys.dictionary_obj_type;
    verificare_object(nume,tip);

end;
create or replace procedure verificare_object
    (nume_object varchar2,
     tip_object varchar2)
is
```

```

begin
if tip_object = 'TABLE' then
    raise_application_error (-20010,'Nu ai voie sa adaugi, sa modifici sau sa stergi tabele!');
else
    creeaza_object(nume_object, tip_object);
end if;
end;
create or replace procedure creeaza_object
    (nume_object varchar2,
     tip_object varchar2)
is
begin
if tip_object in ('PROCEDURE','FUNCTION','PACKAGE','PACKAGE BODY','TRIGGER')
then
    dbms_output.put_line('Actiunea a fost realizata cu succes');
else raise_application_error (-20011,'Ai drepturi doar asupra obiectelor de tip procedura,
functie, pachet sau trigger!');
end if;
end;
create or replace procedure activare_t_ex12
    (functie varchar2)
is
begin
if functie = 'activare' then
    execute immediate 'alter trigger t_ex12 enable';
elsif functie = 'dezactivare' then
    execute immediate 'alter trigger t_ex12 disable';
end if;
end;

begin
    activare_t_ex12('dezactivare');
end;
create table test as select * from produs;
drop table test;
create or replace procedure p_test
is
begin
    dbms_output.put_line('Procedura creata');
end;
drop procedure p_test;

```

```

--12
-- Definiti un trigger de tip LDD. Declansati trigger-ul.

-- Scripti un trigger care permite utilizatorului sa creeze doar functii, proceduri, triggeri sau pachete
-- In caz ca incerca sa stearga , sa modifice sau sa creeze tabele acesta va primi un mesaj de eroare
-- In plus, scrieti si o functie care permite dezactivarea si activarea trigger ului

begin
    activare_t_ex12('dezactivare');
end;

create or replace trigger t_ex12
before create or alter or drop on schema
declare
    nume varchar2(100);
    tip varchar2(100);
begin
    nume := sys.dictionary_obj_name;
    tip := sys.dictionary_obj_type;

    verificare_object(nume,tip);
end;

create or replace procedure verificare_object
    (nume_object varchar2,
     tip_object varchar2)
is
begin
    if

```

Script Output X | Query Result X
Task completed in 0.104 seconds

```

Procedure ACTIVARE_T_EX12 compiled
Procedure CREEAZA_OBJECT compiled
Procedure VERIFICARE_OBJECT compiled
Trigger T_EX12 compiled

```

```

create table test as select * from produs;

```

Script Output X | Query Result X
Task completed in 0.473 seconds

```

Error starting at line : 1,071 in command -
create table test as select * from produs
Error report -
ORA-04088: error during execution of trigger 'UTILIZATOR.T_EX12'
ORA-00604: error occurred at recursive SQL level 1
ORA-20010: Nu ai voie sa adaugi, sa modifici sau sa stergi tabele!
ORA-06512: at "UTILIZATOR.VERIFICARE_OBJECT", line 7
ORA-06512: at line 8
04088. 00000 -  "error during execution of trigger '%s.%s'"
*Cause:   A runtime error occurred during execution of a trigger.
*Action:  Check the triggers which were involved in the operation.

```

```

SQL Worksheet | History
[SQL] Find  Buffer Size:20000 | tutorial
Worksheet Query Builder
is
begin
  if functie = 'activare' then
    execute immediate 'alter trigger t_ex12 enable';
  elsif functie = 'desactivare' then
    execute immediate 'alter trigger t_ex12 disable';
  end if;
end;

begin
  activare_t_ex12('desactivare');
end;

create table test as select * from produs;
create or replace procedure p_test
is
begin
  dbms_output.put_line('Procedura creata');
end;

alter trigger t_ex12 disable;

```

Script Output | Task completed in 0.126 seconds

```

create table test as select * from produs
Error report -
ORA-04088: error during execution of trigger 'UTILIZATOR.T_EX12'
ORA-00604: error occurred at recursive SQL level 1
ORA-20010: Nu ai voie sa adaugi, sa modifici sau sa stergi table!
ORA-06512: at "UTILIZATOR.VERIFICARE_OBJECT", line 7
ORA-06512: at line 8
04088. 00000 -  "error during execution of trigger '%s.%s'"
Cause:  A runtime error occurred during execution of a trigger.
Action: Check the triggers which were involved in the operation.

Procedure P_TEST compiled

```

Activate W
Go to Settings tc

13. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).

Cerinta:

Adaugati o functionalitate magazinului prin asocierea unui nivel fiecarui utilizator. Vor exista 3 nivele : GOLD, daca utilizatorul a cheltuit peste 200 de lei, SILVER, a cheltuit peste 100 de lei. BRONZE altfel. Aplicati reduceri cosului de cumparaturi in functie de nivel:

- reducere de 15 lei primilor 3 utilizator GOLD - celorlalți 10 lei
- reducere de 7,5 lei primilor 3 utilizator SILVER - celorlalți 5 lei
- utilizatorii BRONZE nu beneficiază de reducere

Folositi proceduri si functii pentru popularea colectiilor ce retin informatii despre utilizatori si nivelul acestora astfel incat sa fie posibila intreaga functionalitate doar prin apelul unei singure proceduri.

Logica exercitiului

Se apeleaza procedura creeaza_nivele - apeleaza procedura genereaza_nivel_client si functia aplica_reduceri la final

Din procedura genereaza_nivel_client se apeleaza functia genereaza_client_nivel

Rezolvare:

```
create or replace package p_ex12
is
type record_utilizator_comenzi is record
    (utilizator_comanda utilizator%rowtype,
     nr_comenzi number,
     valoare_cheltuita number);
type tabel_utilizator_comenzi is table of record_utilizator_comenzi;
t_utilizator_comenzi tabel_utilizator_comenzi := tabel_utilizator_comenzi();
type vector_utilizator is varray(100) of utilizator%rowtype;
type record_client_nivel is record
    (v_utilizator vector_utilizator,
     nivel varchar2(10));
type tabel_utilizator_nivel is table of record_client_nivel;
t_utilizator_nivel tabel_utilizator_nivel := tabel_utilizator_nivel();
nr_gold number := 0;
nr_silver number := 0;
nr_bronze number := 0;
nr_total number := 0;
procedure creeaza_nivele;
procedure genereaza_comenzi_client;
function genereaza_nivel_client
    (t_client_comenzi tabel_utilizator_comenzi)
return tabel_utilizator_nivel;
function aplica_reduceri
    (t_client_nivel tabel_utilizator_nivel)
return number;
end p_ex12;
```

create or replace package body p_ex12

is

```
--procedura creeaza_nivele
--aceasta procedura trebuie apelata pentru a se genera nivelele
--ea apeleaza restul functiilor si procedurilor
procedure creeaza_nivele
is
verif_utilizatori number;
fara_utilizatori exception;
begin
--se apeleaza procedura de populare cu utilizator doar daca acestia exista in baza de date
select count(*)
```

```

into verif_utilizatori
from utilizator;
if verif_utilizatori = 0 then
    raise fara_utilizatori;
end if;

-- daca exista utilizatori se populeaza tabloul de t_utilizatori_comenzi din procedura
genereaza_comenzi_client
genereaza_comenzi_client;

--ulterior, din procedura genereaza_comenzi_client se apeleaza functia
genereaza_nivel_client care populeaza t_utilizator_nivel

--afisam toti utilizatorii pe nivele
dbms_output.put_line(' ');
for i in 1..t_utilizator_nivel.count loop
    --parcugem utilizatorii
    if t_utilizator_nivel(i).nivel = 'GOLD' then
        dbms_output.put_line('Clientii GOLD:');
        for j in 1..t_utilizator_nivel(i).v_utilizator.count loop
            dbms_output.put_line(' * '|t_utilizator_nivel(i).v_utilizator(j).nume||'
'|t_utilizator_nivel(i).v_utilizator(j).prenume);
        end loop;
    elsif t_utilizator_nivel(i).nivel = 'SILVER' then
        dbms_output.put_line('Clientii SILVER:');
        for j in 1..t_utilizator_nivel(i).v_utilizator.count loop
            dbms_output.put_line(' * '|t_utilizator_nivel(i).v_utilizator(j).nume||'
'|t_utilizator_nivel(i).v_utilizator(j).prenume);
        end loop;
    else
        dbms_output.put_line('Clientii BRONZE:');
        for j in 1..t_utilizator_nivel(i).v_utilizator.count loop
            dbms_output.put_line(' * '|t_utilizator_nivel(i).v_utilizator(j).nume||'
'|t_utilizator_nivel(i).v_utilizator(j).prenume);
        end loop;
    end if;
end loop;

--la final se aplica reducerile
nr_total := aplica_reduceri(t_utilizator_nivel);
dbms_output.put_line(' ');
dbms_output.put_line('Au fost aplicate reduceri pentru'||nr_total||' dintre clientii GOLD si
SILVER');
exception

```

```

when fara_utilizatori then
    dbms_output.put_line('Nu exista utilizatori inregistrati! Nu se pot genera nivele!');
end creeaza_nivele;

--procedura genereaza_comenzi_client
--aceasta populeaza t_utilizatori_comenzi si t_utilizator_nivel prin apelarea functiei
genereaza_nivel_client
procedure genereaza_comenzi_client
is
    cursor c_utilizatori is
        select id_utilizator, nume, prenume, email, parola, numar_telefon, data_nastere
        from (select u.*, nvl(count(c.id_comanda),0) nr
              from utilizator u
              left join comanda c on c.id_utilizator = u.id_utilizator
              group by
u.id_utilizator, u.nume, u.prenume, u.email, u.parola, u.numar_telefon, u.data_nastere
              order by nr desc);
    -- utilizatorii sunt pusi in ordine in cursor pentru ca tablourile de comenzi si nivele sa fie
    corecte
    type record_utilizator_comenzi;
    ut utilizator%rowtype;
    nr number;
    val number;
begin
    --luam fiecare utilizator si il punem in t_utilizator_comenzi cu datele respective
    --daca utilizatorul nu a comandat nimic atunci va avea 0 la nr_comenzi si la
    valoare_cheltuita
    for u in c_utilizatori loop
        select nvl(count(*),0), nvl(sum(c.total_plata),0)
        into nr, val
        from comanda c
        where c.id_utilizator = u.id_utilizator;

        rec_utilizator.utilizator_comanda := u;
        rec_utilizator.nr_comenzi := nr;
        rec_utilizator.valoare_cheltuita := val;

        t_utilizator_comenzi.extend;
        t_utilizator_comenzi(t_utilizator_comenzi.last) := rec_utilizator;
    end loop;
    --afisam utilizatorii

```

```

dbms_output.put_line('Utilizatorii impreuna cu datele despre comenzi:');
for i in 1..t_utilizator_comenzi.count loop
    dbms_output.put_line(' *'||t_utilizator_comenzi(i).utilizator_comanda.nume||
    ||t_utilizator_comenzi(i).utilizator_comanda.prenume
        || ' nr comenzi'||t_utilizator_comenzi(i).nr_comenzi||' valoare comenzi
    ||t_utilizator_comenzi(i).valoare_cheltuita);

end loop;

--acum putem apela functia genereaza_nivel_client pentru a popula t_utilizator_nivel
t_utilizator_nivel := genereaza_nivel_client(t_utilizator_comenzi);
end genereaza_comenzi_client;

function genereaza_nivel_client
    (t_client_comenzi tabel_utilizator_comenzi)
return tabel_utilizator_nivel
is
    rec_nivel record_client_nivel;
    utilizator_gold vector_utilizator := vector_utilizator();
    utilizator_silver vector_utilizator := vector_utilizator();
    utilizator_bronze vector_utilizator := vector_utilizator();
    utilizator_nivel tabel_utilizator_nivel := tabel_utilizator_nivel();
begin
    for i in 1..t_client_comenzi.count loop
        if t_client_comenzi(i).valoare_cheltuita > 200 then
            utilizator_gold.extend;
            utilizator_gold(utilizator_gold.last) := t_client_comenzi(i).utilizator_comanda;
        elsif t_client_comenzi(i).valoare_cheltuita > 100 then
            utilizator_silver.extend;
            utilizator_silver(utilizator_silver.last) := t_client_comenzi(i).utilizator_comanda;
        else
            utilizator_bronze.extend;
            utilizator_bronze(utilizator_bronze.last) := t_client_comenzi(i).utilizator_comanda;
        end if;
    end loop;

    --utilizatorii gold
    rec_nivel.nivel := 'GOLD';
    rec_nivel.v_utilizator := utilizator_gold;
    utilizator_nivel.extend;
    utilizator_nivel(utilizator_nivel.last) := rec_nivel;

```

```

--utilizatorii silver
rec_nivel.nivel := 'SILVER';
rec_nivel.v_utilizator := utilizator_silver;
utilizator_nivel.extend;
utilizator_nivel(utilizator_nivel.last) := rec_nivel;

--utilizator bronze
rec_nivel.nivel := 'BRONZE';
rec_nivel.v_utilizator := utilizator_bronze;
utilizator_nivel.extend;
utilizator_nivel(utilizator_nivel.last) := rec_nivel;

return utilizator_nivel;

end genereaza_nivel_client;

--functia aplica_reduceri
function aplica_reduceri
(t_client_nivel tabel_utilizator_nivel)
return number
is
nr_modificari number := 0;
val_cos number;
reducere number;
begin
--parcuregem cele 3 nivele
for i in 1..t_client_nivel.count loop
--parcuregem utilizatorii
if t_client_nivel(i).nivel = 'GOLD' then
for j in 1..t_client_nivel(i).v_utilizator.count loop
if j <= 3 then
reducere := 15;
else
reducere := 10;
end if;

select total_cos
into val_cos
from cos_cumparaturi
where id_utilizator = t_client_nivel(i).v_utilizator(j).id_utilizator;

if val_cos > reducere then
update cos_cumparaturi

```

```

        set total_cos = total_cos - reducere
        where id_utilizator = t_client_nivel(i).v_utilizator(j).id_utilizator;
    end if;

        nr_gold := nr_gold + 1;
        nr_modificari := nr_modificari + 1;
    end loop;

    elsif t_client_nivel(i).nivel = 'SILVER' then
        for j in 1..t_client_nivel(i).v_utilizator.count loop
            if j <= 3 then
                reducere := 7.5;
            else
                reducere := 5;
            end if;

            select total_cos
            into val_cos
            from cos_cumparaturi
            where id_utilizator = t_client_nivel(i).v_utilizator(j).id_utilizator;

            if val_cos > reducere then
                update cos_cumparaturi_copy
                set total_cos = total_cos - reducere
                where id_utilizator = t_client_nivel(i).v_utilizator(j).id_utilizator;
            end if;

            nr_silver := nr_silver + 1;
            nr_modificari := nr_modificari + 1;
        end loop;
    else
        for j in 1..t_client_nivel(i).v_utilizator.count loop
            nr_bronze := nr_bronze + 1;
        end loop;
    end if;
end loop;

return nr_modificari;
end aplica_reduceri;

end p_ex12;

begin
    p_ex12.creeaza_nivele;

```

end;

rollback;

The screenshot shows the Oracle SQL Developer interface with two panes. The left pane displays the PL/SQL code for package p_ex12, which includes declarations for record and table types, a procedure to generate levels, and a function to calculate discounts. The right pane shows the results of a query titled 'Utilizatorii impreuna cu datele despre comenzi' and a summary of applied discounts for client levels.

```
SQL Worksheet: History
[SQL] Find | Aa | Buffer Size:20000 | tutorial x
Worksheet | Query Builder
create or replace package p_ex12
is
    type record_utilizator_comenzi is record
        (utilizator_comanda.utilizator%rowtype,
         nr_comenzi number,
         valoare_comenzi number);
    type tabel_utilizator_comenzi is table of record_utilizator_comenzi;
    t_utilizator_comenzi tabel_utilizator_comenzi := tabel_utilizator_comenzi();
    type vector_utilizator is varray(100) of utilizator%rowtype;
    type record_client_nivel is record
        (v_utilizator.vector_utilizator,
         nivel varchar2(10));
    type tabel_utilizator_nivel is table of record_client_nivel;
    t_utilizator_nivel tabel_utilizator_nivel := tabel_utilizator_nivel();
    nr_gold number := 0;
    nr_silver number := 0;
    nr_bronze number := 0;
    nr_total number := 0;

    procedure creeaza_nivele;
    procedure genereaza_comenzi_client;
    function genereaza_nivel_client
        (t_client_comenzi tabel_utilizator_comenzi)
    return tabel_utilizator_nivel;
    function aplică_reduceri
        (t_client_nivel tabel_utilizator_nivel)
    return number;
end p_ex12;

create or replace package body p_ex12
is

```

Script Output X | Task completed in 0.319 seconds

```
Package P_EX12 compiled

Package Body P_EX12 compiled

PL/SQL procedure successfully completed.
```

Activate Windows
Go to Settings to activate Windor

The screenshot shows the Oracle SQL Developer interface with two panes. The left pane displays the implementation of package body p_ex12, which includes loops for calculating discounts and updating tables. The right pane shows the same results as the previous screenshot, including the client list and discount summary.

```
SQL Worksheet: History
[SQL] Find | Aa | Buffer Size:20000 | tutorial x
Worksheet | Query Builder
begin
    p_ex12.creeaza_nivele;
end;
rollback;

end p_ex12;

begin
    p_ex12.aplica_reduceri;
end;

PL/SQL procedure successfully completed.

Package Body P_EX12 compiled

PL/SQL procedure successfully completed.
```

Activate Windows
Go to Settings to activate Window