



Academia de Studii Economice din București
Facultatea de Cibernetică, Statistică și Informatică Economică
Specializarea Informatică Economică

PROIECT

SISTEM DE GESTIUNE A

BAZELOR DE DATE

***Tema proiectului: Sistem informatic pentru
gestiunea unui magazin online***

Cadrul didactic coordonator:

Alexandra Maria Ioana CORBEA

Student:

Bejan Ionut

Introducere

Motivația alegerii acestei teme constă în importanța deosebită pe care o prezintă bazele de date în societatea contemporană, lucru datorat în principal eficienței și ușurinței de care acestea dispun în momentul în care o organizație dorește să gestioneze și să stocheze informații complexe. Un sistem informatic bine structurat asigură accesul rapid și corect la datele esențiale, oferind suport pentru luarea deciziilor în timp real.

Tema proiectului meu are în vedere realizarea unei gestiuni eficiente a activităților dintr-un magazin online, care reprezintă una dintre cele mai importante ramuri ale comerțului modern. Acest sistem are scopul de a organiza și centraliza informațiile despre produse, clienți, comenzi și stocuri, facilitând astfel administrarea operațiunilor zilnice. Ca și etapă inițială, am implementat o structură relațională a bazei de date, care stochează detalii esențiale despre produsele disponibile, comenzile plasate de clienți, precum și legăturile dintre acestea (chei primare și chei externe). Ulterior, această schemă a fost populată cu date relevante pentru simularea proceselor unui magazin online.

În etapa a doua a proiectului, am implementat diferite operații SQL, incluzând operații de actualizare, ștergere și interogare a datelor. Am folosit diverse funcții și expresii pentru a gestiona eficient informațiile despre produse (ex: calcularea prețurilor cu discount, stocuri minime), comenzi (ex: generarea automată a rapoartelor de vânzări) și clienți (ex: evidența comenzilor unui anumit client). De asemenea, am realizat interogări complexe, utilizând funcții SQL avansate, precum subinterogări, DECODE, CASE și operatorii UNION, INTERSECT și MINUS.

Pentru implementarea aplicației am utilizat mediul de dezvoltare oferit de ORACLE și anume SQL Developer, în care am rulat toate secvențele de cod și am proiectat schema bazei de date, inclusiv relațiile și constrângerile dintre tabele. Acest sistem informatic oferă un suport puternic pentru gestionarea eficientă și transparentă a activităților specifice unui magazin online.

Descrierea bazei de date

Tema aleasă are ca obiectiv organizarea, evidența și gestiunea produselor, comenzilor, clienților și altor elemente esențiale care aparțin unui magazin online. Această bază de date conține informații despre produse (ID, nume, descriere, preț, stoc), clienți (ID, nume, prenume, email, telefon, adresă), comenzi (ID comandă, client asociat, dată, total) și detalii despre comenzi (ID produs, cantitate, preț unitar). Sistemul urmărește să faciliteze procesul de vânzare și să ofere o evidență clară a relațiilor dintre produsele vândute, comenzile plasate și clienții magazinului.

Entități	Tip Relație	Relații
UTILIZATORI - COMENZI	One to Many	Fiecare utilizator poate plasa mai multe comenzi. Relația este implementată prin coloana ID_UTILIZATOR din tabelul COMENZI .
COMENZI - DETALII_COMENZI	One to Many	O comandă conține mai multe produse, detaliate în tabelul DETALII_COMENZI . Relația este implementată prin coloana ID_COMANDA.
PRODUSE - DETALII_COMENZI	One to Many	Un produs poate fi inclus în mai multe comenzi. Relația este definită prin coloana ID_PRODUS din tabelul DETALII_COMENZI .
CATEGORII - PRODUSE	One to Many	O categorie poate avea mai multe produse. Relația este implementată prin coloana ID_CATEGORIE din tabelul PRODUSE .
FURNIZORI - PRODUSE_FURNIZORI	One to Many	Fiecare furnizor poate livra mai multe produse. Relația este definită prin coloana ID_FURNIZOR din tabelul PRODUSE_FURNIZORI .
PRODUSE - RECENZII	One to Many	Fiecare produs poate avea mai multe recenzii. Relația este implementată prin coloana ID_PRODUS din tabelul RECENZII .
UTILIZATORI - RECENZII	One to Many	Fiecare utilizator poate lăsa mai multe recenzii. Relația este definită prin coloana ID_UTILIZATOR din tabelul RECENZII .
PRODUSE - REDUCERI	One to Many	Fiecare produs poate avea reduceri asociate, definite în tabelul REDUCERI . Relația este implementată prin coloana ID_PRODUS.

1. Definirea schemei bazei de date

-- UTILIZATORI – Salvează informațiile despre utilizatori.

```
CREATE TABLE UTILIZATORI (  
    ID_UTILIZATOR NUMBER(10) PRIMARY KEY,  
    NUME_UTILIZATOR VARCHAR2(50) UNIQUE NOT NULL,  
    EMAIL VARCHAR2(100) UNIQUE NOT NULL,  
    PAROLA VARCHAR2(100) NOT NULL,  
    NUME_COMPLET VARCHAR2(100),  
    DATA_NASTERII DATE  
);
```

--PRODUSE – Stochează informații despre produse.

```
CREATE TABLE PRODUSE (  
    ID_PRODUS NUMBER(10) PRIMARY KEY,  
    NUME_PRODUS VARCHAR2(100) NOT NULL,  
    PRET NUMBER(10, 2) NOT NULL,  
    STOC NUMBER DEFAULT 0,  
    ID_CATEGORIE NUMBER,  
    FOREIGN KEY (ID_CATEGORIE) REFERENCES  
CATEGORII(ID_CATEGORIE) ON DELETE SET NULL  
);
```

--CATEGORII – Grupați produsele pe categorii.

```
CREATE TABLE CATEGORII (  
    ID_CATEGORIE NUMBER(10) PRIMARY KEY,  
    NUME_CATEGORIE VARCHAR2(100) UNIQUE NOT NULL,  
    DESCRIERE VARCHAR2(255)  
);
```

--COMENZI – Detalii despre comenzile utilizatorilor.

```
CREATE TABLE COMENZI (  
    ID_COMANDA NUMBER(10) PRIMARY KEY,  
    ID_UTILIZATOR NUMBER NOT NULL,  
    DATA_COMANDA DATE DEFAULT SYSDATE,  
    TOTAL_PLATA NUMBER(10, 2) DEFAULT 0,  
    FOREIGN KEY (ID_UTILIZATOR) REFERENCES  
UTILIZATORI(ID_UTILIZATOR) ON DELETE CASCADE  
);
```

--DETALII_COMENZI – Lista produselor dintr-o comandă.

```
CREATE TABLE DETALII_COMENZI (  
    ID_DETALII_COMANDA NUMBER(10) PRIMARY KEY,  
    ID_COMANDA NUMBER NOT NULL,  
    ID_PRODUS NUMBER NOT NULL,  
    CANTITATE NUMBER DEFAULT 1 CHECK (CANTITATE > 0),  
    PRET_PE_UNITATE NUMBER(10, 2) NOT NULL,  
    FOREIGN KEY (ID_COMANDA) REFERENCES COMENZI(ID_COMANDA) ON  
DELETE CASCADE,  
    FOREIGN KEY (ID_PRODUS) REFERENCES PRODUSE(ID_PRODUS) ON  
DELETE CASCADE  
);
```

--FURNIZORI – Informații despre furnizori.

```
CREATE TABLE FURNIZORI (  
    ID_FURNIZOR NUMBER(10) PRIMARY KEY,  
    NUME_FURNIZOR VARCHAR2(100) NOT NULL,  
    EMAIL_CONTACT VARCHAR2(100) UNIQUE,  
    TELEFON VARCHAR2(15),  
    ADRESA VARCHAR2(255)  
);
```

--PRODUSE_FURNIZORI – Relația dintre furnizori și produse.

```
CREATE TABLE PRODUSE_FURNIZORI (  
    ID_PRODUS_FURNIZOR NUMBER(10) PRIMARY KEY,  
    ID_FURNIZOR NUMBER NOT NULL,  
    ID_PRODUS NUMBER NOT NULL,  
    PRET_FURNIZARE NUMBER(10, 2) NOT NULL,  
    FOREIGN KEY (ID_FURNIZOR) REFERENCES FURNIZORI(ID_FURNIZOR)  
ON DELETE CASCADE,  
    FOREIGN KEY (ID_PRODUS) REFERENCES PRODUSE(ID_PRODUS) ON  
DELETE CASCADE  
);
```

--RECENZII – Recenzii date de utilizatori pentru produse.

```
CREATE TABLE RECENZII (  
    ID_RECENZIE NUMBER(10) PRIMARY KEY,  
    ID_UTILIZATOR NUMBER NOT NULL,  
    ID_PRODUS NUMBER NOT NULL,  
    RATING NUMBER CHECK (RATING BETWEEN 1 AND 5),  
    TEXT_RECENZIE VARCHAR2(255),  
    DATA_RECENZIE DATE DEFAULT SYSDATE,  
    FOREIGN KEY (ID_UTILIZATOR) REFERENCES  
UTILIZATORI(ID_UTILIZATOR) ON DELETE CASCADE,  
    FOREIGN KEY (ID_PRODUS) REFERENCES PRODUSE(ID_PRODUS) ON  
DELETE CASCADE  
);
```

--REDUCERI – Reduceri aplicate produselor.

```
CREATE TABLE REDUCERI (
    ID_REDUCERE NUMBER(10) PRIMARY KEY,
    ID_PRODUS NUMBER NOT NULL,
    PROCENT_REDUCERE NUMBER(5, 2) CHECK (PROCENT_REDUCERE
BETWEEN 0 AND 100),
    DATA_INCEPUT DATE NOT NULL,
    DATA_SFARSIT DATE NOT NULL,
    FOREIGN KEY (ID_PRODUS) REFERENCES PRODUSE(ID_PRODUS) ON
DELETE CASCADE
);
```

Se utilizează comenzile CREATE, ALTER, DROP

- **Comanda ALTER** – Modificarea unor tabele existente:

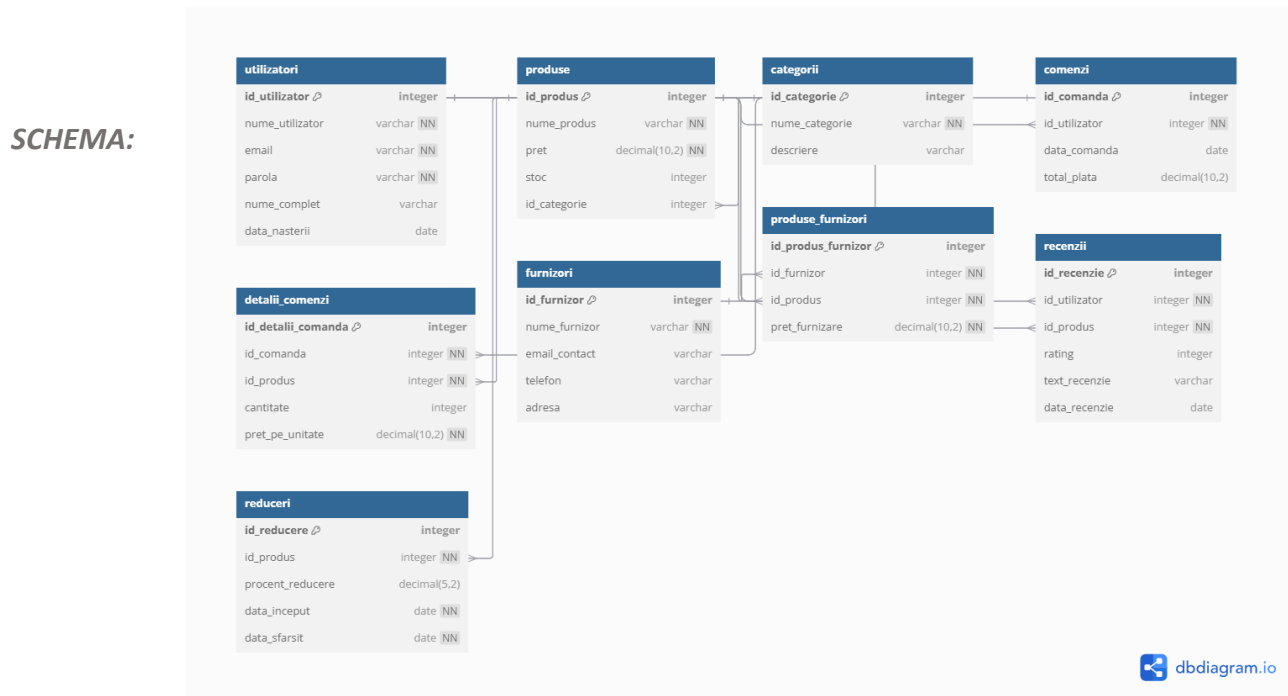
```
ALTER TABLE PRODUCE ADD
(STOC NUMBER(10) DEFAULT 0
);
```

```
ALTER TABLE UTILIZATORI ADD
DATA_INREGISTRARII DATE DEFAULT SYSDATE;
```

```
ALTER TABLE PRODUCE MODIFY
(STOC NUMBER(10,2) PRIMARY KEY
);
```

- **Comanda DROP** – Ștergerea si recuperarea unui tabel:

```
DROP TABLE PRODUCE ;
FLASHBACK TABLE PRODUCE TO BEFORE DROP;
```



Oracle SQL Developer: D:\USER\Desktop\sqldeveloper\sqldeveloper\ionut_1.sql

File Edit View Navigate Run Source Test Tools Window Help

Connections

Oracle Connections

ionut

Tables (Filtered)

- CATEGORIES
- COMENZI
- DETAILI_COMENZI
- FURNIZORI
- PRODUSE
- PRODUSE_FURNIZORI
- RECEZII
- REDOUCERI
- UTILIZATORI

Views

Indexes

Packages

Procedures

Functions

Operators

Queues

Queues Tables

Triggers

Types

Sequences

Materialized Views

Reports

- All Reports
- About Your Database
- All Objects
- Analytic View Reports
- Application Express
- ADP and ADP
- Database Administration
- Data Dictionary
- Data Dictionary Reports
- Data Modeler Reports
- OLAP Reports
- PL/SQL
- Security
- Streams
- Table
- Text/Text Reports
- User Defined Reports
- XML

SQL Worksheet

Query Builder

```

-- UTILIZATORI - Salvarea informatiilor despre utilizatori.
CREATE TABLE UTILIZATORI (
  ID_UTILIZATOR NUMBER(10) PRIMARY KEY,
  NUME_UTILIZATOR VARCHAR2(50) UNIQUE NOT NULL,
  EMAIL VARCHAR2(100) UNIQUE NOT NULL,
  PAROLA VARCHAR2(100) NOT NULL,
  ID_CATEGORIE NUMBER,
  DATA_INSCRIERII DATE
);

-- PRODUSE - Stocarea informatiilor despre produse.
CREATE TABLE PRODUSE (
  ID_PRODUS NUMBER(10) PRIMARY KEY,
  NUME_PRODUS VARCHAR2(100) NOT NULL,
  PRET NUMBER(10, 2) NOT NULL,
  STOC NUMBER DEFAULT 0,
  ID_CATEGORIE NUMBER,
  FURNIZOR KEY (ID_CATEGORIE) REFERENCES CATEGORIES(ID_CATEGORIE) ON DELETE SET NULL
);

-- CATEGORIES - Gruparea produselor pe categorii.
CREATE TABLE CATEGORIES (
  ID_CATEGORIE NUMBER(10) PRIMARY KEY,
  NOME_CATEGORIE VARCHAR2(100) UNIQUE NOT NULL,
  DESCRIERE VARCHAR2(255)
);

-- COMENZI - Detalii despre comenzile utilizatorilor.
CREATE TABLE COMENZI (
  ID_COMANDA NUMBER(10) PRIMARY KEY,
  ID_UTILIZATOR NUMBER NOT NULL,
  DATA_COMANDA DATE DEFAULT SYSDATE,
  TOTAL_PLATA NUMBER(10, 2) DEFAULT 0,
  FURNIZOR KEY (ID_UTILIZATOR) REFERENCES UTILIZATORI(ID_UTILIZATOR) ON DELETE CASCADE
);

```

Script Output

Task completed in 0.913 seconds

Table RECEZII created.

Table REDUCERI created.

Messages - Log

Messages Statements Logging Page

Click on an identifier with the Control key down to perform "Go to Declaration"

Line 57 Column 1 | Insert | Modified | Windows: O

Oracle SQL Developer: Table BSEMAN_SAPRODUSE@ionut

File Edit View Navigate Run Test Tools Window Help

Connections

Oracle Connections

ionut

Tables (Filtered)

- CATEGORIES
- COMENZI
- DETAILI_COMENZI
- FURNIZORI
- PRODUSE
- PRODUSE_FURNIZORI
- RECEZII
- REDOUCERI
- UTILIZATORI

Views

Indexes

Packages

Procedures

Functions

Operators

Queues

Queues Tables

Triggers

Types

Sequences

Materialized Views

Reports

- All Reports
- About Your Database
- All Objects
- Analytic View Reports
- Application Express
- ADP and ADP
- Database Administration
- Data Dictionary
- Data Dictionary Reports
- Data Modeler Reports
- OLAP Reports
- PL/SQL
- Security
- Streams
- Table
- Text/Text Reports
- User Defined Reports
- XML

Table: PRODUSE

ID_PRODUS	NOME_PRODUS	PRET	STOC	ID_CATEGORIE
1	103 Tastatura Razer	3999	15	2
2	101 Televizor Samsung	1500,99	10	2
3	102 Mouse Logitech G502	249,99	50	2
4	104 Scaun ergonomic	249,99	20	3
5	105 Săculeț Buzie	99,99	30	4
6	106 Geacă de Iarnă	499,99	15	5
7	201 Tastatura Gaming	349,99	15	1
8	202 Mouse Profesional	199,99	20	1
9	203 Monitor 4K	1299,99	10	2
10	204 Cesti Audio	499,99	30	2

Compiler - Log

Click on an identifier with the Control key down to perform "Go to Declaration"

Oracle SQL Developer: Table BEAHM_S4.FURNIZORI@lonut

File Edit View Navigate Run Tools Window Help

Connections

Oracle Connections

BEAHM_S4.FURNIZORI

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

Columns: ID_FURNIZOR, Nume_Furnizor, EMAIL_CONTACT, TELEFON, ADRESA

ID_FURNIZOR	Nume_Furnizor	EMAIL_CONTACT	TELEFON	ADRESA
1	Furnizor Tech	tech@example.com	0712345678	Str. Tehnologiei, Nr. 10
2	Distribuitor Gaming	gaming@example.com	0723456789	Jocurilor, Nr. 15
3	Produsator Office	office@example.com	0734567890	Str. Birourilor, Nr. 20
4	Echipamente Profesionale	pro@example.com	0745678901	Str. Profesioniștilor, Nr. 25
5	Accesorii Universale	access@example.com	0756789012	Str. Universalului, Nr. 30
6	Distribuitor IT	it@example.com	0767890123	Str. Calculatoarelor, Nr. 40
7	Furnizor Gadget-uri	gadget@example.com	0778901234	Str. Inovatiei, Nr. 50
8	Produsator Software	software@example.com	0789012345	Str. Programatorilor, Nr. 60
9	Megastin Online	online@example.com	0790123456	Str. E-commerce, Nr. 70
10	Furnizor Electroanice	electro@example.com	0711234567	Str. Electroaniceilor, Nr. 80

Reports

Compiler - Log

Oracle SQL Developer: Table BEAHM_S4.UTILIZATORI@lonut

File Edit View Navigate Run Tools Window Help

Connections

Oracle Connections

BEAHM_S4.UTILIZATORI

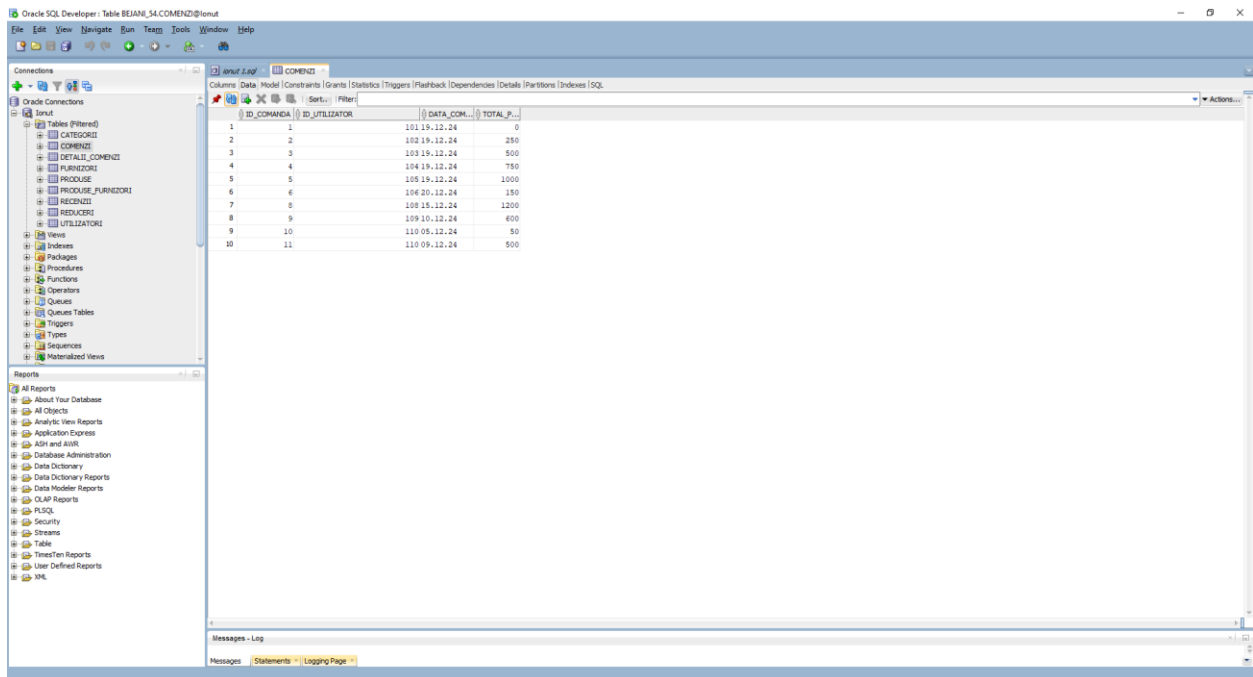
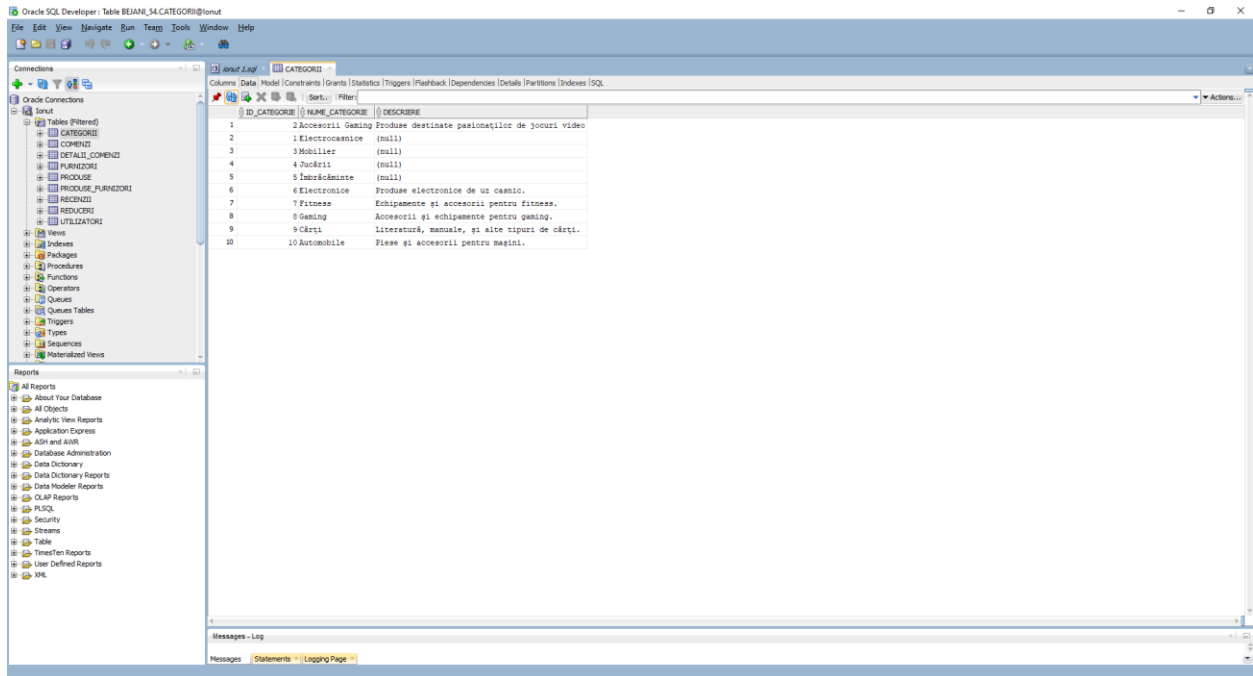
Columns Data Model Constraints Grants Statistics Triggers Flashback Dependencies Details Partitions Indexes SQL

Columns: ID_UTILIZATOR, Nume_Utilizator, EMAIL, PAROLA, Nume_Complet, DATA_NASTERII

ID_UTILIZATOR	Nume_Utilizator	EMAIL	PAROLA	Nume_Complet	DATA_NASTERII
1	Ionel Popescu	ion.popescu@example.com	parola1456	Ionel Popescu	21.03.99
2	101 user1	user1@example.com	parola1	User One	01.01.90
3	102 user2	user2@example.com	parola2	User Two	02.02.92
4	103 user3	user3@example.com	parola3	User Three	03.03.94
5	104 user4	user4@example.com	parola4	User Four	04.04.96
6	105 user5	user5@example.com	parola5	User Five	05.05.98
7	1 user1_new	new_user@example.com	parola123	User One	01.01.00
8	106 user6	user6@example.com	parola6	User Six	06.06.00
9	107 user7	user7@example.com	parola7	User Seven	07.07.01
10	108 user8	user8@example.com	parola8	User Eight	08.08.02
11	109 user9	user9@example.com	parola9	User Nine	09.09.03
12	110 user10	user10@example.com	parola10	User Ten	10.10.04

Reports

Compiler - Log



Oracle SQL Developer: Table BEIANI_S4DETAILI_COMENZI@lonut

File Edit View Navigate Run Tools Window Help

Connections

- Oracle Connections
 - lonut
 - Tables (Filtered)
 - CATEGORIE
 - COMENZI
 - DETAILI_COMENZI
 - ID_DETAILI_COMANDA
 - ID_COMANDA
 - ID_PRODUS
 - CANTITATE
 - PRET_PE_LINIE
 - PRODUSE
 - PRODUSE_FURNIZORI
 - RECONZII
 - REDACTARE
 - UTILIZATORI
 - Views
 - Indexes
 - Packages
 - Procedures
 - Functions
 - Operators
 - Queues

Reports

- All Reports
- About Your Database
- All Objects
- Analytic View Reports
- Application Express
- ASH and AWR
- Database Administration
- Data Dictionary
- Data Dictionary Reports
- Data Modeler Reports
- OLAP Reports
- PLSQL
- Security
- Streams
- Table
- TimesTen Reports
- User Defined Reports
- XML

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

Table: DETAILI_COMENZI

	ID_DETAILI_COMANDA	ID_COMANDA	ID_PRODUS	CANTITATE	PRET_PE_LINIE
1	1	1	201	2	150,5
2	2	1	202	1	200
3	3	2	203	3	350,75
4	4	2	204	5	100
5	11	6	201	8	150
6	12	6	202	4	225,5
7	13	3	201	2	150
8	14	4	202	3	200
9	15	5	203	1	350
10	16	6	204	4	100

Messages - Log

Messages | Statements | Logging Page

Oracle SQL Developer: Table BEIANI_S4PRODUSE_FURNIZORI@lonut

File Edit View Navigate Run Tools Window Help

Connections

- Oracle Connections
 - lonut
 - Tables (Filtered)
 - CATEGORIE
 - COMENZI
 - DETAILI_COMENZI
 - FURNIZORI
 - PRODUSE
 - ID_PRODUSE_FURNIZOR
 - ID_FURNIZOR
 - ID_PRODUS
 - PRET_FURNIZARE
 - PRODUSE_FURNIZORI
 - RECONZII
 - REDACTARE
 - UTILIZATORI
 - Views
 - Indexes
 - Packages
 - Procedures
 - Functions
 - Operators
 - Queues
 - Queues Tables
 - Triggers
 - Types
 - Sequences
 - Materialized Views

Reports

- All Reports
- About Your Database
- All Objects
- Analytic View Reports
- Application Express
- ASH and AWR
- Database Administration
- Data Dictionary
- Data Dictionary Reports
- Data Modeler Reports
- OLAP Reports
- PLSQL
- Security
- Streams
- Table
- TimesTen Reports
- User Defined Reports
- XML

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

Table: PRODUSE_FURNIZORI

	ID_PRODUSE_FURNIZOR	ID_FURNIZOR	ID_PRODUS	PRET_FURNIZARE
1	1	1	201	120
2	2	1	202	150
3	3	2	203	200
4	4	2	204	200
5	11	1	201	120
6	12	2	202	150
7	13	3	203	300
8	14	4	204	200
9	15	2	205	250
10	16	3	206	300

Messages - Log

Messages | Statements | Logging Page

Oracle SQL Developer: Table BEIANI_SARECENZII@ionut

File Edit View Navigate Run Tools Window Help

Connections

Oracle Connections

Ionut

Tables (Filtered)

CATEGORIES

COMENZI

DETAIL_COMENZI

FURNIZORI

PRODUSE

PRODUSE_FURNIZORI

RECENZII

REDUCERI

UTILIZATORI

Views

Indexes

Packages

Procedures

Functions

Operators

Queues

Queue Tables

Triggers

Types

Sequences

Materialized Views

Reports

All Reports

About Your Database

All Objects

Analytic View Reports

Application Express

ASH and ASHR

Database Administration

Data Dictionary

Data Dictionary Reports

Data Modeler Reports

OLAP Reports

PLSQL

Security

Streams

Table

TimeTen Reports

User Defined Reports

XPL

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

Columns: ID_RECENZIE ID_UTILIZATOR ID_PRODUS RATING TEXT_RECENZIE DATA_RECENZIE

1	1	101	201	5	Excelent produs!	19.12.24
2	2	102	202	4	Foarte bun, dar cu mici probleme.	19.12.24
3	3	103	203	5	Perfect pentru nevoile mele!	19.12.24
4	4	104	204	3	Mediocr, putea fi mai bun.	19.12.24
5	5	105	201	5	Recomand cu incredere!	19.12.24
6	6	101	202	4	Produs bun, dar livrarea a intarziat.	20.12.24
7	7	102	203	5	Calitate foarte buna, recomand!	20.12.24
8	8	103	204	3	Amaliaz dimensiune, dar produsul functioneaza.	20.12.24
9	9	104	205	2	Nu sunt multumit de performanta.	20.12.24
10	10	105	206	5	Un produs excelent, livrare rapida!	20.12.24

Messages - Log

Messages Statements Logging Page

Oracle SQL Developer: Table BEIANI_SARECENZII@ionut

File Edit View Navigate Run Tools Window Help

Connections

Oracle Connections

Ionut

Tables (Filtered)

CATEGORIES

COMENZI

DETAIL_COMENZI

FURNIZORI

PRODUSE

PRODUSE_FURNIZORI

RECENZII

REDUCERI

UTILIZATORI

Views

Indexes

Packages

Procedures

Functions

Operators

Queues

Queue Tables

Triggers

Types

Sequences

Materialized Views

Reports

All Reports

About Your Database

All Objects

Analytic View Reports

Application Express

ASH and ASHR

Database Administration

Data Dictionary

Data Dictionary Reports

Data Modeler Reports

OLAP Reports

PLSQL

Security

Streams

Table

TimeTen Reports

User Defined Reports

XPL

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

Columns: ID_RECENZIE ID_PRODUS PROCENT_REDUcere DATA_INCEPUT DATA_SFARSIT

1	1	201	10.01.12.24	31.12.24
2	2	202	15.05.12.24	20.12.24
3	3	203	20.10.12.24	25.12.24
4	4	204	5.15.12.24	31.12.24
5	5	205	25.01.12.24	10.12.24
6	6	206	30.05.12.24	20.12.24
7	7	207	12.5.10.12.24	25.12.24
8	8	208	8.15.12.24	31.12.24
9	9	209	10.01.12.24	10.12.24
10	10	210	50.01.12.24	31.12.24

Messages - Log

Messages Statements Logging Page

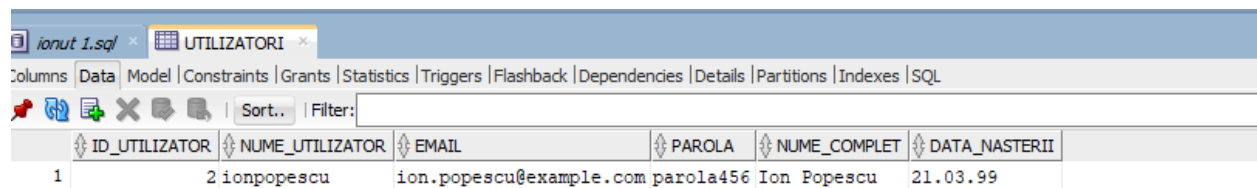
2.Operații DML(Insert, Update, Delete, Merge)

1. INSERT - Adăugare de date în tabele

- **EX 1: Adăugare utilizatori în tabelul UTILIZATORI**

```
INSERT INTO UTILIZATORI (ID_UTILIZATOR, NUME_UTILIZATOR, EMAIL,  
PAROLA, NUME_COMPLET, DATA_NASTERII)
```

```
VALUES (2, 'ionpopescu', 'ion.popescu@example.com', 'parola456',  
'Ion Popescu', TO_DATE('1999-03-21', 'YYYY-MM-DD'));
```



The screenshot shows a database client window titled 'ionut 1.sql' and 'UTILIZATORI'. The 'Data' tab is selected, displaying a table with 6 columns: ID_UTILIZATOR, NUME_UTILIZATOR, EMAIL, PAROLA, NUME_COMPLET, and DATA_NASTERII. There is one row of data with the following values: 2, ionpopescu, ion.popescu@example.com, parola456, Ion Popescu, and 21.03.99.

ID_UTILIZATOR	NUME_UTILIZATOR	EMAIL	PAROLA	NUME_COMPLET	DATA_NASTERII
2	ionpopescu	ion.popescu@example.com	parola456	Ion Popescu	21.03.99

- **EX 2: Adăugare produse în tabelul PRODUSE**

```
INSERT INTO PRODUSE (ID_PRODUS, NUME_PRODUS, PRET, STOC,  
ID_CATEGORIE)VALUES (102, 'Mouse Logitech G502', 249.99, 50, 2);
```

- **EX 3: Adăugare recenzii în tabelul RECENZII**

```
INSERT INTO RECENZII (ID_RECENZIE, ID_UTILIZATOR, ID_PRODUS,  
RATING, TEXT_RECENZIE, DATA_RECENZIE)
```

```
VALUES (1, 2, 102, 5, 'Mouse excelent pentru gaming!', SYSDATE);
```

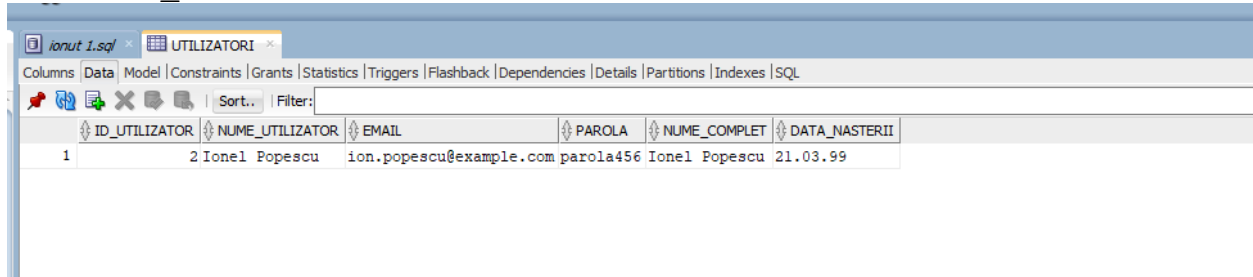
2. UPDATE - Modificare date existente

- **EX 1: Actualizează prețul unui produs**

```
UPDATE PRODUSE  
SET PRET = 219.99  
WHERE ID_PRODUS = 102;
```

- **EX 2: Actualizează numele complet al unui utilizator**

```
UPDATE UTILIZATORI
SET NUME_UTILIZATOR = 'Ionel Popescu'
WHERE ID_UTILIZATOR = 2;
```



ID_UTILIZATOR	NUME_UTILIZATOR	EMAIL	PAROLA	NUME_COMPLET	DATA_NASTERII
1	2 Ionel Popescu	ion.popescu@example.com	parola456	Ionel Popescu	21.03.99

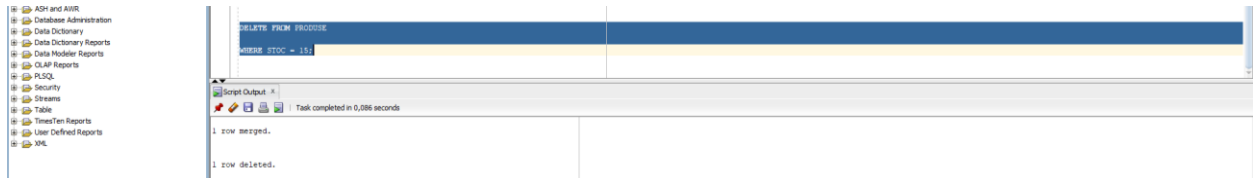
- **EX 3: Actualizează cantitatea stocului pentru un produs**

```
UPDATE PRODUSE
SET STOC = STOC + 20
WHERE ID_PRODUS = 102;
```

3. DELETE - Ștergerea datelor

- **EX 1: Șterge un produs care nu mai este în stoc**

```
DELETE FROM PRODUSE
WHERE STOC = 15;
```



Task completed in 0.086 seconds

1 row merged.

1 row deleted.

- **EX 2: Șterge toate recenziile unui utilizator specific**

```
DELETE FROM RECENZII
WHERE ID_UTILIZATOR = 2;
```

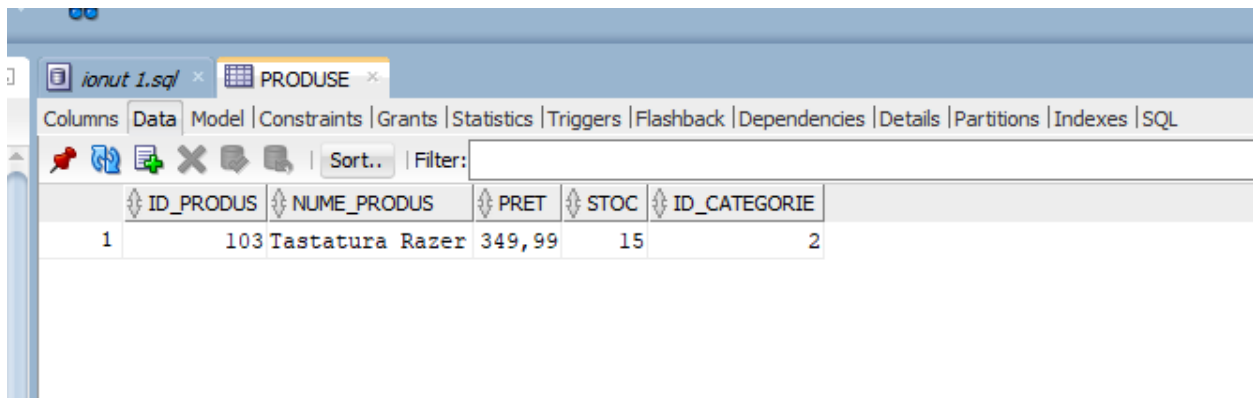
- **EX 3: Șterge un furnizor din baza de date**

```
DELETE FROM FURNIZORI
WHERE ID_FURNIZOR = 3;
```

4. MERGE - Inserare sau actualizare condiționată

- **EX 1: Actualizează prețul unui produs sau inserează-l dacă nu există**

```
MERGE INTO PRODUSE P
USING (SELECT 103 AS ID_PRODUS,
        'Tastatura Razer' AS NUME_PRODUS,
        349.99 AS PRET,
        15 AS STOC,
        2 AS ID_CATEGORIE
FROM DUAL) SRC
ON (P.ID_PRODUS = SRC.ID_PRODUS)
WHEN MATCHED THEN
    UPDATE SET PRET = SRC.PRET, STOC = SRC.STOC
WHEN NOT MATCHED THEN
    INSERT (ID_PRODUS, NUME_PRODUS, PRET, STOC, ID_CATEGORIE)
    VALUES (SRC.ID_PRODUS, SRC.NUME_PRODUS, SRC.PRET, SRC.STOC,
SRC.ID_CATEGORIE);
```



The screenshot shows the Oracle SQL Developer interface. The 'PRODUSE' table is selected, and the 'Data' tab is active. The table contains one row with the following values:

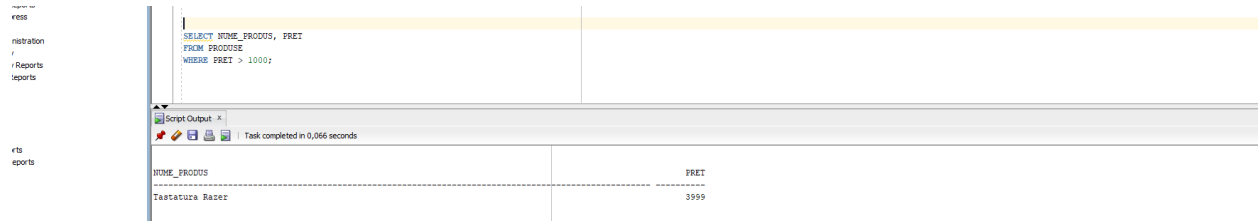
ID_PRODUS	NUME_PRODUS	PRET	STOC	ID_CATEGORIE
103	Tastatura Razer	349,99	15	2

3.Exemple de interogări variate (SELECT)

1. Utilizarea operatorilor de comparație

- **EX 1: Afișează produsele cu preț mai mare de 1000.**

```
SELECT NUME_PRODUS, PRET
FROM PRODUSE
WHERE PRET > 1000;
```



NUME_PRODUS	PRET
Tastatura Razer	3999

- **EX 2: Afișează comenzile cu totalul mai mic de 500.**

```
SELECT ID_COMANDA, TOTAL_PLATA
FROM COMENZI
WHERE TOTAL_PLATA < 500;
```

- **EX 3: Afișează produsele cu stoc între 10 și 50.**

```
SELECT NUME_PRODUS, STOC
FROM PRODUSE
WHERE STOC BETWEEN 10 AND 50;
```

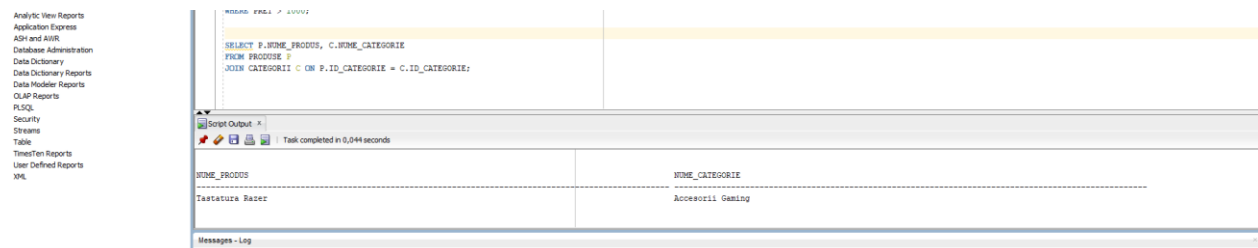
2. Join-uri

- **EX 1: Afișează comenzile împreună cu numele utilizatorilor care le-au plasat.**

```
SELECT C.ID_COMANDA, U.NUME_UTILIZATOR, C.DATA_COMANDA,
C.TOTAL_PLATA
FROM COMENZI C
JOIN UTILIZATORI U ON C.ID_UTILIZATOR = U.ID_UTILIZATOR;
```

- **EX 2: Afișează produsele și categoriile lor.**

```
SELECT P.NUME_PRODUS, C.NUME_CATEGORIE
FROM PRODUSE P
JOIN CATEGORII C ON P.ID_CATEGORIE = C.ID_CATEGORIE;
```



NUME_PRODUS	NUME_CATEGORIE
Tastatura Razer	Accesorii Gaming

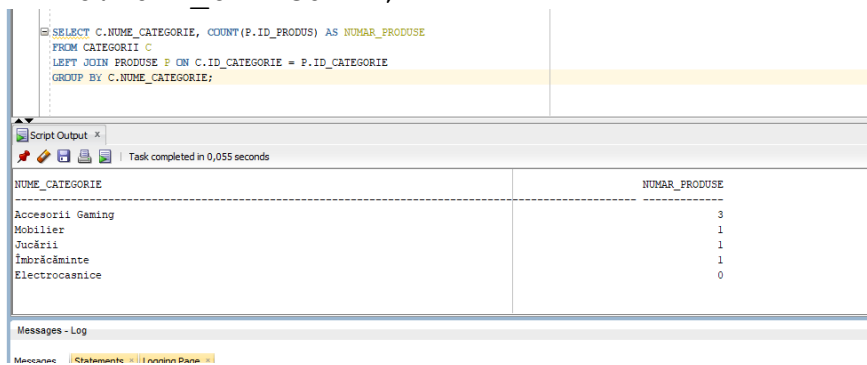
- **EX 3: Afișează recenziile produselor împreună cu utilizatorii care le-au scris.**

```
SELECT R.ID_RECENZIE, U.NUME_UTILIZATOR, R.RATING,
R.TEXT_RECENZIE
FROM RECENZII R
JOIN UTILIZATORI U ON R.ID_UTILIZATOR = U.ID_UTILIZATOR;
```

3. Utilizarea funcțiilor de grup și condiții asupra acestora

- **EX 1: Afișează numărul de produse în fiecare categorie.**

```
SELECT C.NUME_CATEGORIE, COUNT(P.ID_PRODUS) AS NUMAR_PRODUSE
FROM CATEGORII C
LEFT JOIN PRODUSE P ON C.ID_CATEGORIE = P.ID_CATEGORIE
GROUP BY C.NUME_CATEGORIE;
```



NUME_CATEGORIE	NUMAR_PRODUSE
Accesorii Gaming	3
Mobilier	1
Jucării	1
Îmbrăcăminte	1
Electrocasnice	0

- **EX 2: Afișează suma totală a comenzilor pentru fiecare utilizator.**

```
SELECT U.NUME_UTILIZATOR, SUM(C.TOTAL_PLATA) AS TOTAL_COMENZI
FROM COMENZI C
JOIN UTILIZATORI U ON C.ID_UTILIZATOR = U.ID_UTILIZATOR
GROUP BY U.NUME_UTILIZATOR;
```

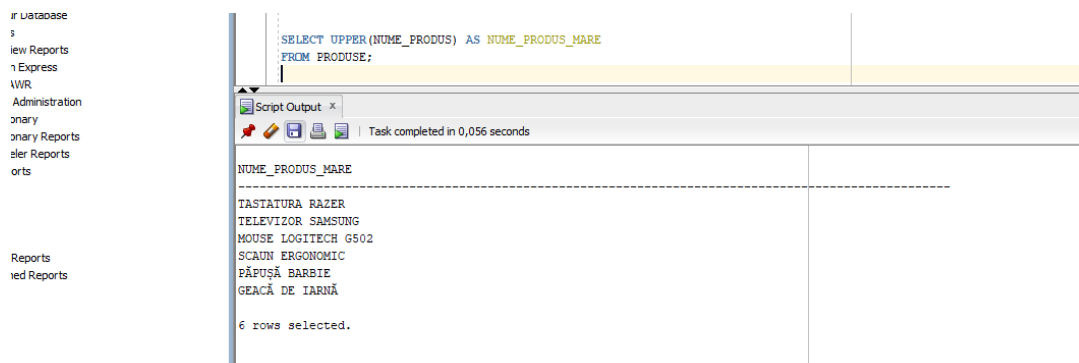
- **EX3: Afișează stocul mediu al produselor dacă acesta este mai mare de 20.**

```
SELECT AVG(STOC) AS STOC_MEDIU
FROM PRODUSE
HAVING AVG(STOC) > 20;
```

4. Utilizarea funcțiilor numerice, de tip caracter, pentru data și timp

- **EX 1: Afișează numele produselor în litere mari.**

```
SELECT UPPER(NUME_PRODUS) AS NUME_PRODUS_MARE
FROM PRODUSE;
```



NUME_PRODUS_MARE
TASTATURA RAZER
TELEVIZOR SAMSUNG
MOUSE LOGITECH G502
SCAUN ERGONOMIC
PĂPUȘĂ BARBIE
GEACĂ DE IARNĂ

6 rows selected.

- **EX 2: Afișează produsele reduse în anul curent.**

```
SELECT NUME_PRODUS, DATA_INCEPUT
FROM REDUCERI
WHERE EXTRACT(YEAR FROM DATA_INCEPUT) = EXTRACT(YEAR FROM
SYSDATE);
```

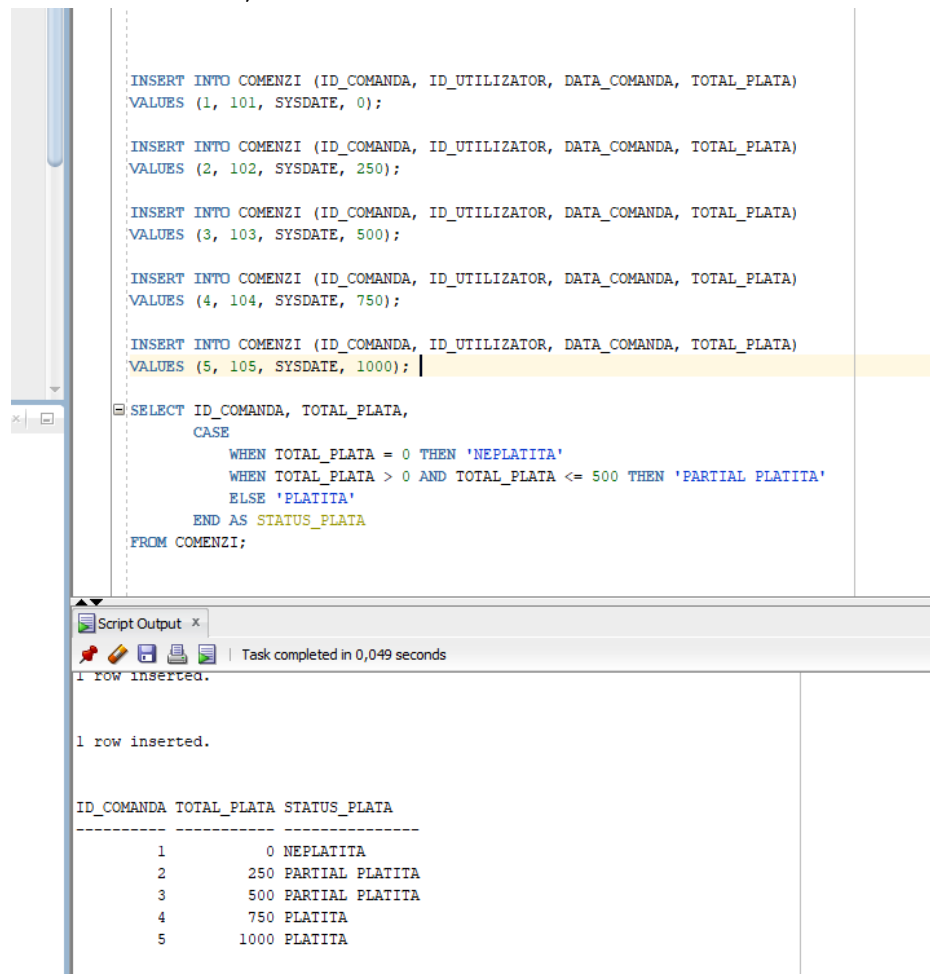
- **EX 3: Afișează numărul de zile dintre data de început și sfârșit a reducerilor.**

```
SELECT ID_REDCERE, DATA_INCEPUT, DATA_SFARSIT, (DATA_SFARSIT -
DATA_INCEPUT) AS ZILE_REDCERE
FROM REDUCERI;
```

5. Construirea de expresii cu DECODE și CASE

- **EX 1: Afișează comenzile cu statusul plății (NEPLĂTITĂ, PARȚIAL PLĂTITĂ, PLĂTITĂ).**

```
SELECT ID_COMANDA, TOTAL_PLATA,
CASE
    WHEN TOTAL_PLATA = 0 THEN 'NEPLATITA'
    WHEN TOTAL_PLATA > 0 AND TOTAL_PLATA <= 500 THEN
'PARTIAL PLATITA'
    ELSE 'PLATITA'
END AS STATUS_PLATA
FROM COMENZI;
```



```
INSERT INTO COMENZI (ID_COMANDA, ID_UTILIZATOR, DATA_COMANDA, TOTAL_PLATA)
VALUES (1, 101, SYSDATE, 0);

INSERT INTO COMENZI (ID_COMANDA, ID_UTILIZATOR, DATA_COMANDA, TOTAL_PLATA)
VALUES (2, 102, SYSDATE, 250);

INSERT INTO COMENZI (ID_COMANDA, ID_UTILIZATOR, DATA_COMANDA, TOTAL_PLATA)
VALUES (3, 103, SYSDATE, 500);

INSERT INTO COMENZI (ID_COMANDA, ID_UTILIZATOR, DATA_COMANDA, TOTAL_PLATA)
VALUES (4, 104, SYSDATE, 750);

INSERT INTO COMENZI (ID_COMANDA, ID_UTILIZATOR, DATA_COMANDA, TOTAL_PLATA)
VALUES (5, 105, SYSDATE, 1000);

SELECT ID_COMANDA, TOTAL_PLATA,
CASE
    WHEN TOTAL_PLATA = 0 THEN 'NEPLATITA'
    WHEN TOTAL_PLATA > 0 AND TOTAL_PLATA <= 500 THEN 'PARTIAL PLATITA'
    ELSE 'PLATITA'
END AS STATUS_PLATA
FROM COMENZI;
```

Script Output x

Task completed in 0,049 seconds

1 row inserted.

1 row inserted.

ID_COMANDA	TOTAL_PLATA	STATUS_PLATA
1	0	NEPLATITA
2	250	PARTIAL PLATITA
3	500	PARTIAL PLATITA
4	750	PLATITA
5	1000	PLATITA

- ```
SELECT NUME_PRODUS, STOC,
 DECODE(
 SIGN(STOC - 10),
 -1, 'STOC SCAZUT',
 0, 'STOC MINIM',
 1, 'STOC SUFICIENT'
) AS STATUS_STOC
FROM PRODUSE;
```

- **EX 1:** Am combinați numele produselor și furnizorilor într-o listă unică, care poate fi utilizată pentru a genera un catalog general al resurselor unui magazin.

ports

- All Reports
- About Your Database
- All Objects
- Analytic View Reports
- Application Express
- ASH and AWR
- Database Administration
- Data Dictionary
- Data Dictionary Reports
- Data Modeler Reports
- OLAP Reports
- PLSQL
- Security
- Streams
- Table
- TimesTen Reports
- User Defined Reports
- XML

```

SELECT NUME_PRODUS AS DENUMIRE, 'PRODUS' AS TIP
FROM PRODUSE
UNION
SELECT NUME_FURNIZOR AS DENUMIRE, 'FURNIZOR' AS TIP
FROM FURNIZORI;

```

Script Output x

Task completed in 0,114 seconds

More Details :  
<https://docs.oracle.com/error-help/db/ora-00942/>

| DENUMIRE            | TIP    |
|---------------------|--------|
| Tastatura Razer     | PRODUS |
| Televizor Samsung   | PRODUS |
| Mouse Logitech G502 | PRODUS |
| Scaun ergonomic     | PRODUS |
| Păpușă Barbie       | PRODUS |
| Geacă de iarnă      | PRODUS |
| Tastatura Gaming    | PRODUS |
| Mouse Profesional   | PRODUS |
| Monitor 4K          | PRODUS |
| Casti Audio         | PRODUS |
| Televizor Smart     | PRODUS |
|                     |        |
| DENUMIRE            | TIP    |
| Mouse Gaming        | PRODUS |
| Monitor FullHD      | PRODUS |

Messages - Log

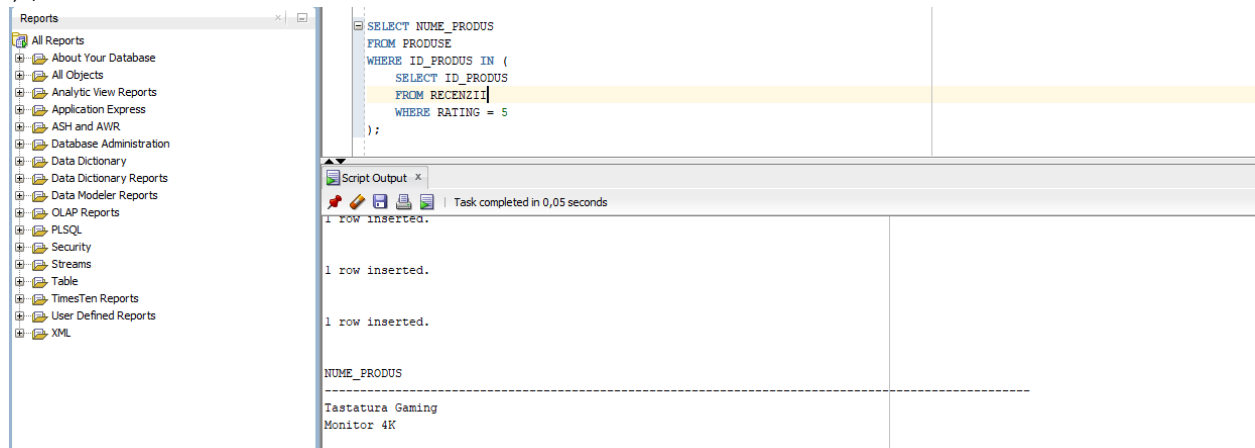
- ```
SELECT ID_PRODUS
FROM PRODUSE
MINUS
SELECT ID_PRODUS
FROM RECENZII;
```

- ```
SELECT ID_PRODUS
FROM RECENZII
INTERSECT
SELECT ID_PRODUS
FROM REDUCERI;
```

## 7. Subcereri (cereri imbricate)

- **EX 1: Afișează produsele care au recenzii cu rating maxim.**

```
SELECT NUME_PRODUS
FROM PRODUSE
WHERE ID_PRODUS IN (
 SELECT ID_PRODUS
 FROM RECENZII
 WHERE RATING = 5
);
```



- **EX 2: Afișează utilizatorii care au plasat comenzi.**

```
SELECT NUME_UTILIZATOR
FROM UTILIZATORI
WHERE ID_UTILIZATOR IN (
 SELECT ID_UTILIZATOR
 FROM COMENZI
);
```

- **EX 3: Afișează comenzile cu suma totală peste media tuturor comenzilor.**

```
SELECT ID_COMANDA, TOTAL_PLATA
FROM COMENZI
WHERE TOTAL_PLATA > (
 SELECT AVG(TOTAL_PLATA)
 FROM COMENZI
);
```

## 4. Gestiunea altor obiecte ale bazei de date.

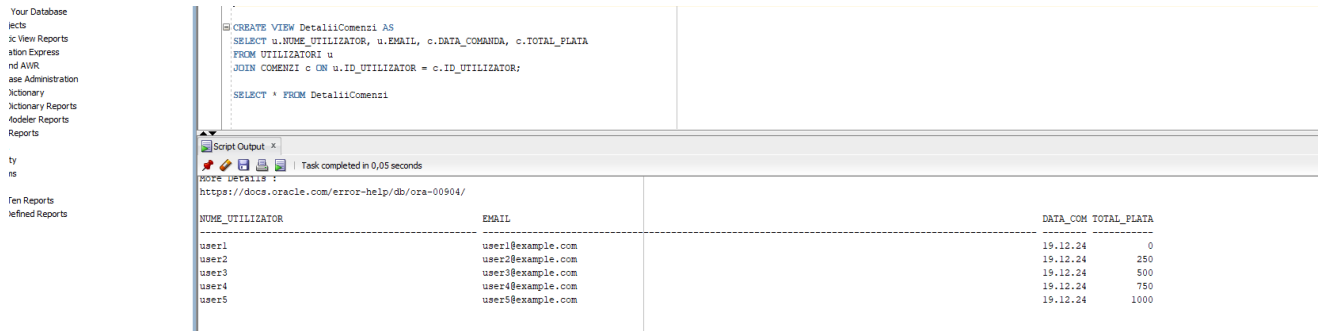
### 1. Vederi (Views)

*Vederile sunt structuri logice ce permit accesarea datelor într-un mod simplificat. Ele pot fi folosite pentru a ascunde complexitatea interogărilor sau pentru securitate (afișând doar anumite coloane).*

```
CREATE VIEW DetaliiComenzi AS
SELECT u.NUME_UTILIZATOR, u.EMAIL, c.DATA_COMANDA, c.TOTAL_PLATA
FROM UTILIZATORI u
JOIN COMENZI c ON u.ID_UTILIZATOR = c.ID_UTILIZATOR;
```

Utilizarea vederii:

```
SELECT * FROM DetaliiComenzi
```



Script Output: Task completed in 0,05 seconds

<https://docs.oracle.com/error-help/db/ora-00904/>

| NUME_UTILIZATOR | EMAIL             | DATA_COM | TOTAL_PLATA |
|-----------------|-------------------|----------|-------------|
| user1           | user1@example.com | 19.12.24 | 0           |
| user2           | user2@example.com | 19.12.24 | 250         |
| user3           | user3@example.com | 19.12.24 | 500         |
| user4           | user4@example.com | 19.12.24 | 750         |
| user5           | user5@example.com | 19.12.24 | 1000        |

### 2. Indecși (Indexes)

*Indecșii accelerează interogările, mai ales când sunt utilizate frecvent coloane în filtre (WHERE) sau în alăturări (JOIN).*

-Coloana **DATA\_COMANDA** este utilizată frecvent pentru interogări care filtrează comenzile după dată.

-Crearea unui index pe această coloană optimizează interogările care implică sortări sau filtre pe bază de dată.

```
CREATE INDEX idx_data_comanda ON COMENZI (DATA_COMANDA);
SELECT * FROM COMENZI
WHERE DATA_COMANDA >= TO_DATE('2024-01-01', 'YYYY-MM-DD');
```

```

CREATE INDEX idx_data_comanda ON COMENZI (DATA_COMANDA);
SELECT * FROM COMENZI
WHERE DATA_COMANDA >= TO_DATE('2024-01-01', 'YYYY-MM-DD');

```

Script Output x

Task completed in 0,038 seconds

additional columns in the 'CREATE INDEX' statement.

Index IDX\_DATA\_COMANDA created.

| ID_COMANDA | ID_UTILIZATOR | DATA_COM | TOTAL_PLATA |
|------------|---------------|----------|-------------|
| 1          | 101           | 19.12.24 | 0           |
| 2          | 102           | 19.12.24 | 250         |
| 3          | 103           | 19.12.24 | 500         |
| 4          | 104           | 19.12.24 | 750         |
| 5          | 105           | 19.12.24 | 1000        |
| 6          | 106           | 20.12.24 | 150         |
| 8          | 108           | 15.12.24 | 1200        |
| 9          | 109           | 10.12.24 | 600         |
| 10         | 110           | 05.12.24 | 50          |
| 11         | 110           | 09.12.24 | 500         |

10 rows selected.

### 3. Sinonime (Synonyms)

*Sinonimele sunt aliasuri pentru obiectele bazei de date. Sunt utile pentru a simplifica accesul la tabele sau alte obiecte din baze de date diferite.*

Exemplu:

```
-- Crearea unui sinonim pentru tabelul Comenzi
CREATE SYNONYM ComenziSimplu FOR Comenzi;
```

Utilizarea sinonimului:

```
SELECT * FROM ComenziSimplu;
```

```

Exemplu:
-- Crearea unui sinonim pentru tabelul Comenzi
CREATE SYNONYM ComenziSimplu FOR Comenzi;

Utilizarea sinonimului:
SELECT * FROM ComenziSimplu;

```

Script Output x

Task completed in 0,127 seconds

| ID_COMANDA | ID_UTILIZATOR | DATA_COM | TOTAL_PLATA |
|------------|---------------|----------|-------------|
| 1          | 101           | 19.12.24 | 0           |
| 2          | 102           | 19.12.24 | 250         |
| 3          | 103           | 19.12.24 | 500         |
| 4          | 104           | 19.12.24 | 750         |
| 5          | 105           | 19.12.24 | 1000        |

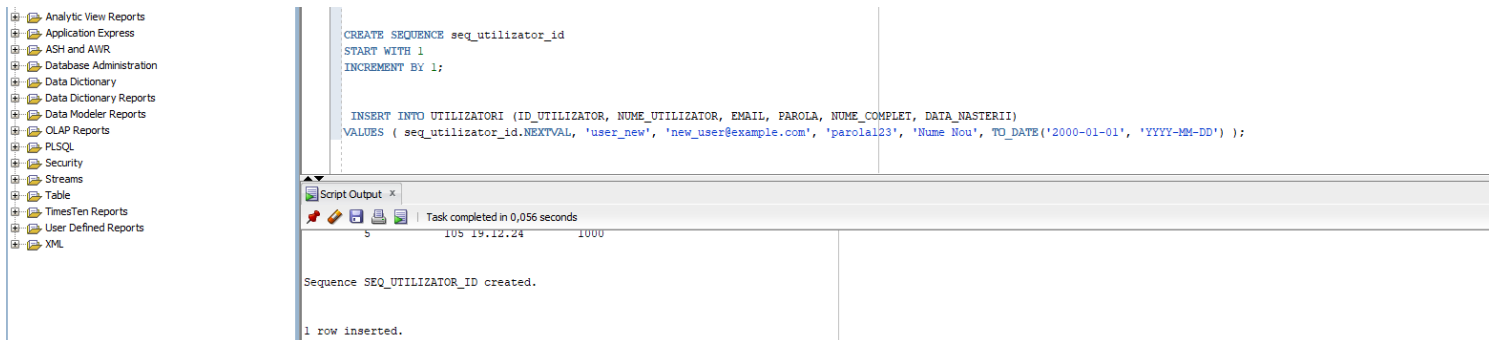
## 4. Secvențe (Sequences)

*Secvențele generează valori unice, utile pentru cheile primare.*

Exemplu:

```
-- Crearea unei secvențe pentru generarea automată a ID-urilor
utilizatorilor
CREATE SEQUENCE seq_utilizator_id
START WITH 1
INCREMENT BY 1;

-- Inserarea unui utilizator nou folosind secvența pentru ID
INSERT INTO UTILIZATORI (ID_UTILIZATOR, NUME_UTILIZATOR, EMAIL,
PAROLA, NUME_COMPLET, DATA_NASTERII)
VALUES (seq_utilizator_id.NEXTVAL, 'user_new',
'new_user@example.com', 'parola123', 'Nume Nou', TO_DATE('2000-
01-01', 'YYYY-MM-DD'));
```



## 5. Alte Obiecte: Triggere

*Triggerele sunt utilizate pentru a automatiza acțiuni la modificarea datelor (inserare, actualizare, ștergere).*

Exemplu:

```
-- Crearea unui trigger care verifică dacă suma comenzii este
mai mare decât 0
CREATE OR REPLACE TRIGGER trg_verificare_suma
BEFORE INSERT OR UPDATE ON Comenzi
FOR EACH ROW
BEGIN
 IF :NEW.Suma <= 0 THEN
 RAISE_APPLICATION_ERROR(-20001, 'Suma comenzii trebuie
să fie mai mare decât 0!');
 END IF;
END;
/
```

Materialized Views

Reports

All Reports

About Your Database

All Objects

Analytic View Reports

Application Express

ASH and AWR

Database Administration

Data Dictionary

Data Dictionary Reports

Data Modeler Reports

OLAP Reports

PL/SQL

Security

Streams

Table

TimesTen Reports

User Defined Reports

XML

CREATE OR REPLACE TRIGGER trg\_verificare\_suma  
BEFORE INSERT OR UPDATE ON Comenzi  
FOR EACH ROW  
BEGIN  
IF :NEW.TOTAL\_PLATA <= 0 THEN  
RAISE\_APPLICATION\_ERROR(-20001, 'Suma comenzii trebuie să fie mai mare decât 0!');  
END IF;  
END;  
INSERT INTO Comenzi (ID\_COMANDA, ID\_UTILIZATOR, DATA\_COMANDA, TOTAL\_PLATA)  
VALUES (1, 101, SYSDATE, -50);

Script Output x

Task completed in 0,08 seconds  
VALUES (1, 101, SYSDATE, -50)  
Error at Command Line : 309 Column : 13  
Error report -  
SQL Error: ORA-20001: Suma comenzii trebuie să fie mai mare decât 0!

## 6. Comenzi SQL în PL/SQL (LDD și LMD)

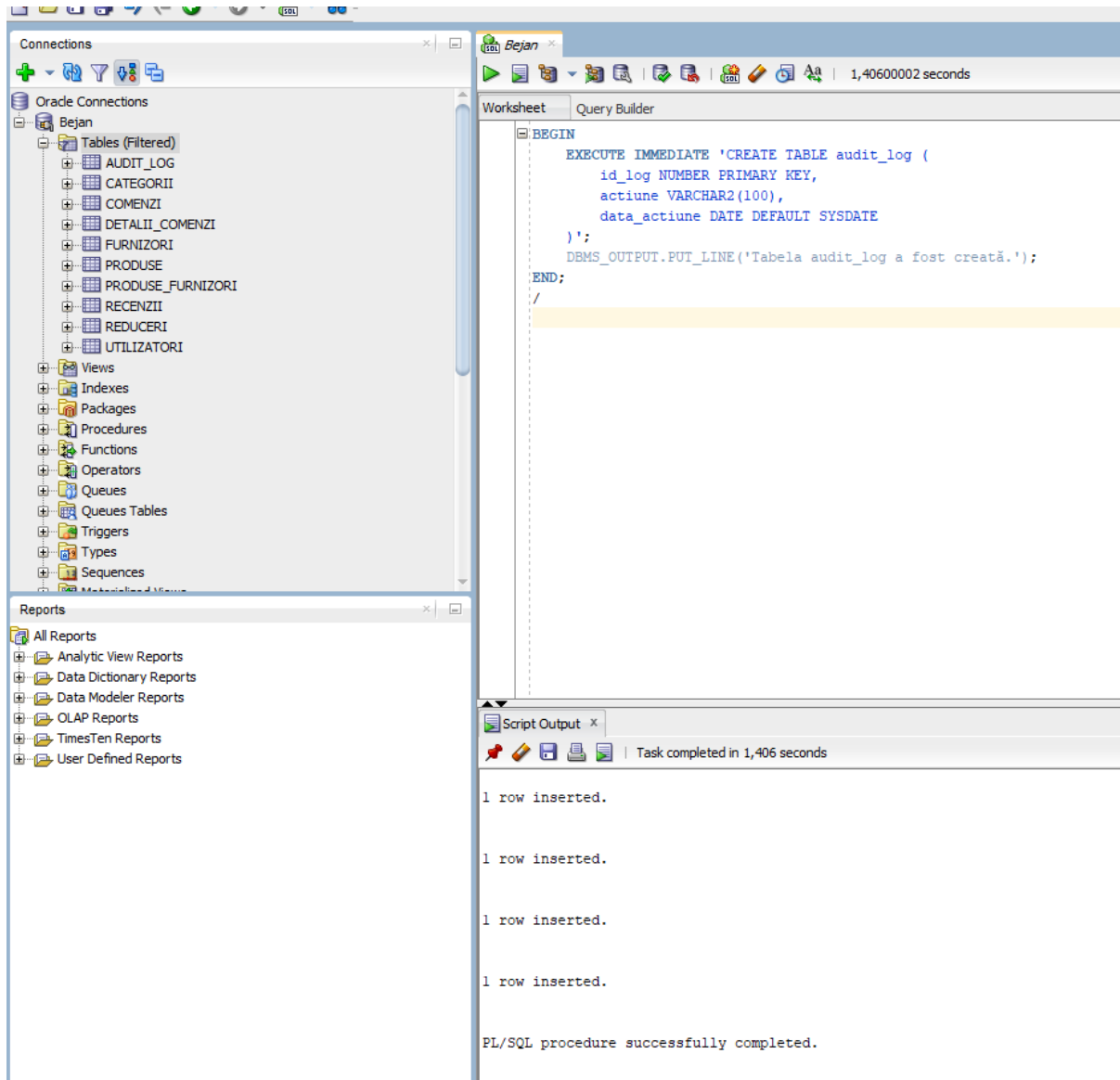
### 1. CREATE TABLE – Creare tabel nou

#### EX 1: Creează o tabelă de audit

```
BEGIN
 EXECUTE IMMEDIATE 'CREATE TABLE audit_log (
 id_log NUMBER PRIMARY KEY,
 actiune VARCHAR2(100),
 data_actiune DATE DEFAULT SYSDATE
)';
 DBMS_OUTPUT.PUT_LINE('Tabela audit_log a fost creata.');
```

END;

/





## 2. ALTER TABLE – Modificare structură tabelă

### EX 1: Adaugă coloana „utilizator” în audit\_log

```
BEGIN
 EXECUTE IMMEDIATE 'ALTER TABLE audit_log ADD (utilizator
VARCHAR2(50))';
 DBMS_OUTPUT.PUT_LINE('Coloana utilizator a fost adaugata.');
```

END;  
/

The screenshot displays the Oracle SQL Developer interface. On the left, the 'Connections' pane shows a tree view of the database schema, including tables like AUDIT\_LOG, ID\_LOG, ACTIUNE, DATA\_ACTIUNE, and UTILIZATOR. The 'Reports' pane at the bottom left lists various report types. The main 'Worksheet' pane contains the SQL script being executed. Below the worksheet, the 'Script Output' pane shows the results of the execution.

**Worksheet:**

```
BEGIN
 EXECUTE IMMEDIATE 'ALTER TABLE audit_log ADD (utilizator VARCHAR2(50))';
 DBMS_OUTPUT.PUT_LINE('Coloana utilizator a fost adăugată.');
```

**Script Output:**

```
1 row inserted.

1 row inserted.

1 row inserted.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.
```

### 3. DROP TABLE – Ștergerea unei tabele

#### EX 1: Șterge tabela audit\_log

```
BEGIN
 EXECUTE IMMEDIATE 'DROP TABLE audit_log';
 DBMS_OUTPUT.PUT_LINE('Tabela audit_log a fost stearsa.');
```

END;

/

The screenshot displays the Oracle SQL Developer environment. On the left, the 'Connections' pane shows the 'Bejan' connection selected, with a tree view of database objects including Tables (Filtered), Views, Indexes, Packages, Procedures, Functions, Operators, Queues, Queues Tables, Triggers, Types, Sequences, and Materialized Views. The 'Tables (Filtered)' section is expanded, showing a list of tables including CATEGORII, CO, DETALII\_COMENZI, FURNIZORI, PRODUSE, PRODUSE\_FURNIZORI, RECENZII, REDUCERI, and UTILIZATORI. The 'Scripts' pane at the bottom left shows a list of reports including All Reports, Analytic View Reports, Data Dictionary Reports, Data Modeler Reports, OLAP Reports, TimesTen Reports, and User Defined Reports.

The main workspace is divided into two panes: 'Worksheet' and 'Query Builder'. The 'Worksheet' pane contains the following PL/SQL script:

```
BEGIN
 EXECUTE IMMEDIATE 'ALTER TABLE audit_log ADD (utilizator VARCHAR2(50))';
 DBMS_OUTPUT.PUT_LINE('Coloana utilizator a fost adăugată.');
```

END;

/

```
BEGIN
 EXECUTE IMMEDIATE 'DROP TABLE audit_log';
 DBMS_OUTPUT.PUT_LINE('Tabela audit_log a fost ștearsă.');
```

END;

/

The 'Script Output' pane at the bottom right shows the execution results:

```
1 row inserted.

1 row inserted.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.
```

The task was completed in 0,08 seconds.

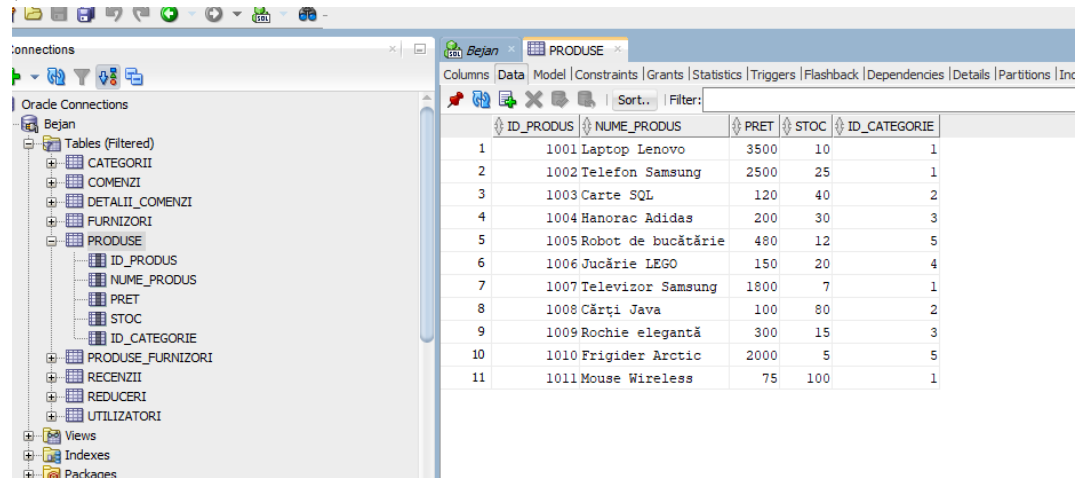
## 4. INSERT – Adăugare date noi

### EX 1: Adaugă un produs nou

```
BEGIN
 INSERT INTO produse (id_produs, nume_produs, pret, stoc,
id_categorie)
 VALUES (1011, 'Mouse Wireless', 75, 100, 1);
 DBMS_OUTPUT.PUT_LINE('Produs adaugat cu succes.');
```

END;

/



The screenshot shows the Oracle SQL Developer interface. On the left, the 'Connections' pane shows the 'Bejan' connection. The 'Tables (Filtered)' list includes CATEGORII, COMENZI, DETALII\_COMENZI, FURNIZORI, PRODUSE, PRODUSE\_FURNIZORI, RECENTZII, REDUCERI, and UTILIZATORI. The 'PRODUSE' table is selected. The main pane displays the 'Data' tab for the 'PRODUSE' table, showing 11 rows of data. The columns are ID\_PRODUS, NUME\_PRODUS, PRET, STOC, and ID\_CATEGORIE.

| ID_PRODUS | NUME_PRODUS             | PRET | STOC | ID_CATEGORIE |
|-----------|-------------------------|------|------|--------------|
| 1         | 1001 Laptop Lenovo      | 3500 | 10   | 1            |
| 2         | 1002 Telefon Samsung    | 2500 | 25   | 1            |
| 3         | 1003 Carte SQL          | 120  | 40   | 2            |
| 4         | 1004 Hanorac Adidas     | 200  | 30   | 3            |
| 5         | 1005 Robot de bucătărie | 480  | 12   | 5            |
| 6         | 1006 Jucărie LEGO       | 150  | 20   | 4            |
| 7         | 1007 Televizor Samsung  | 1800 | 7    | 1            |
| 8         | 1008 Cărți Java         | 100  | 80   | 2            |
| 9         | 1009 Rochie elegantă    | 300  | 15   | 3            |
| 10        | 1010 Frigider Arctic    | 2000 | 5    | 5            |
| 11        | 1011 Mouse Wireless     | 75   | 100  | 1            |

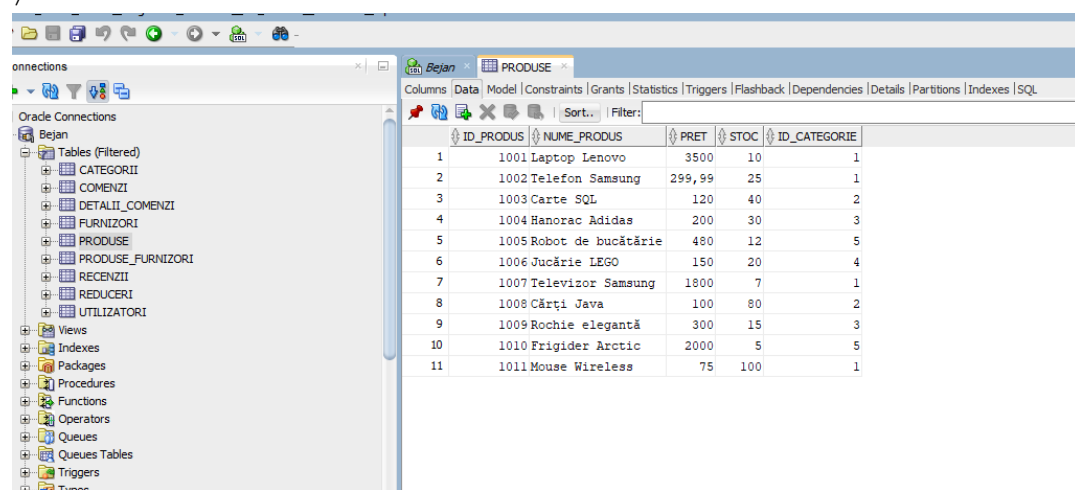
## 5. UPDATE – Modificare date existente

### EX 1: Actualizează prețul unui produs

```
BEGIN
 UPDATE produse
 SET pret = 299.99
 WHERE id_produs = 1002;
 DBMS_OUTPUT.PUT_LINE('Prețul a fost actualizat pentru
produsul 1002.');
```

END;

/



The screenshot shows the Oracle SQL Developer interface. On the left, the 'Connections' pane shows the 'Bejan' connection. The 'Tables (Filtered)' list includes CATEGORII, COMENZI, DETALII\_COMENZI, FURNIZORI, PRODUSE, PRODUSE\_FURNIZORI, RECENTZII, REDUCERI, and UTILIZATORI. The 'PRODUSE' table is selected. The main pane displays the 'Data' tab for the 'PRODUSE' table, showing 11 rows of data. The columns are ID\_PRODUS, NUME\_PRODUS, PRET, STOC, and ID\_CATEGORIE. The price for product 1002 (Telefon Samsung) has been updated to 299.99.

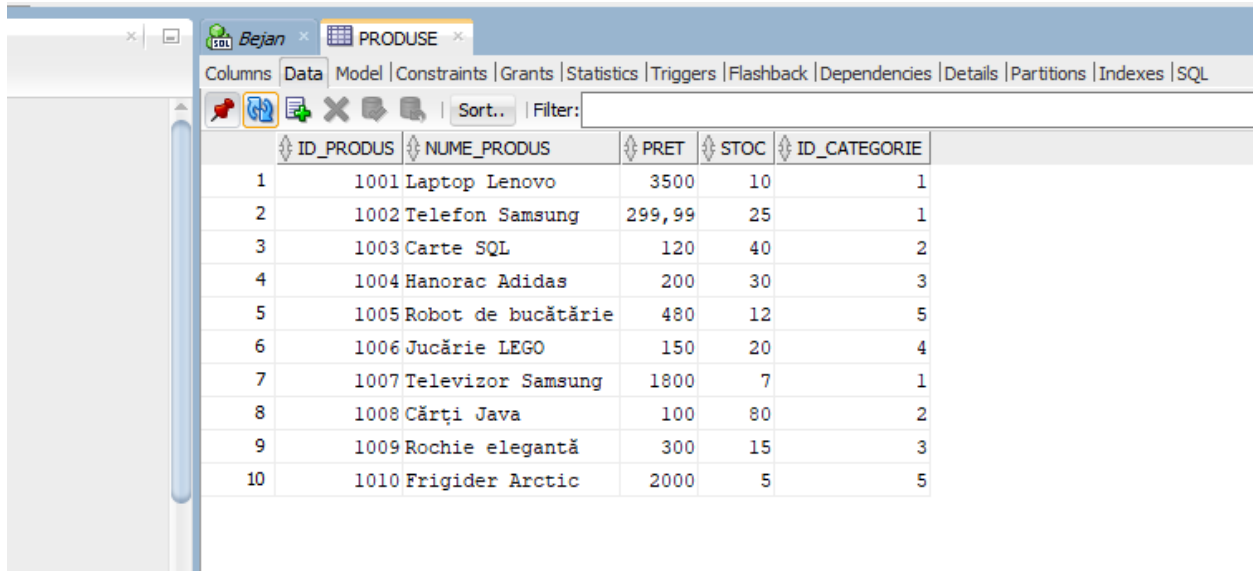
| ID_PRODUS | NUME_PRODUS             | PRET   | STOC | ID_CATEGORIE |
|-----------|-------------------------|--------|------|--------------|
| 1         | 1001 Laptop Lenovo      | 3500   | 10   | 1            |
| 2         | 1002 Telefon Samsung    | 299,99 | 25   | 1            |
| 3         | 1003 Carte SQL          | 120    | 40   | 2            |
| 4         | 1004 Hanorac Adidas     | 200    | 30   | 3            |
| 5         | 1005 Robot de bucătărie | 480    | 12   | 5            |
| 6         | 1006 Jucărie LEGO       | 150    | 20   | 4            |
| 7         | 1007 Televizor Samsung  | 1800   | 7    | 1            |
| 8         | 1008 Cărți Java         | 100    | 80   | 2            |
| 9         | 1009 Rochie elegantă    | 300    | 15   | 3            |
| 10        | 1010 Frigider Arctic    | 2000   | 5    | 5            |
| 11        | 1011 Mouse Wireless     | 75     | 100  | 1            |

## 6. DELETE – Ștergere rând

### EX 1: Șterge produsul cu ID-ul 1011

```
BEGIN
 DELETE FROM produse
 WHERE id_produs = 1011;
 DBMS_OUTPUT.PUT_LINE('Produsul cu ID 1011 a fost sters.');
```

END;  
/



The screenshot shows the Oracle SQL Developer interface with the 'PRODUSE' table selected. The 'Data' tab is active, displaying a list of 10 products. The columns are ID\_PRODUS, NUME\_PRODUS, PRET, STOC, and ID\_CATEGORIE. The data is as follows:

|    | ID_PRODUS | NUME_PRODUS        | PRET   | STOC | ID_CATEGORIE |
|----|-----------|--------------------|--------|------|--------------|
| 1  | 1001      | Laptop Lenovo      | 3500   | 10   | 1            |
| 2  | 1002      | Telefon Samsung    | 299,99 | 25   | 1            |
| 3  | 1003      | Carte SQL          | 120    | 40   | 2            |
| 4  | 1004      | Hanorac Adidas     | 200    | 30   | 3            |
| 5  | 1005      | Robot de bucătărie | 480    | 12   | 5            |
| 6  | 1006      | Jucărie LEGO       | 150    | 20   | 4            |
| 7  | 1007      | Televizor Samsung  | 1800   | 7    | 1            |
| 8  | 1008      | Cărți Java         | 100    | 80   | 2            |
| 9  | 1009      | Rochie elegantă    | 300    | 15   | 3            |
| 10 | 1010      | Frigider Arctic    | 2000   | 5    | 5            |

## 7. MERGE – Inserare/actualizare condiționată

### EX 1: Dacă există produsul 1003, actualizează stocul; altfel inserează-l

```
BEGIN
 MERGE INTO produse p
 USING (SELECT 1003 AS id_produs FROM dual) src
 ON (p.id_produs = src.id_produs)
 WHEN MATCHED THEN
 UPDATE SET stoc = stoc + 20
 WHEN NOT MATCHED THEN
 INSERT (id_produs, nume_produs, pret, stoc,
id_categorie)
 VALUES (1003, 'Carte SQL', 120, 40, 2);
 DBMS_OUTPUT.PUT_LINE('Operatie MERGE executata.');
```

END;  
/

The screenshot shows the Oracle SQL Developer interface. On the left, the 'Connections' pane lists the 'Bejan' connection. Under 'Tables (Filtered)', the 'PRODUSE' table is selected. The main window displays the 'PRODUSE' table data in a grid view. The table has columns: ID\_PRODUS, NUME\_PRODUS, PRET, STOC, and ID\_CATEGORIE. The data is as follows:

| ID_PRODUS | NUME_PRODUS             | PRET   | STOC | ID_CATEGORIE |
|-----------|-------------------------|--------|------|--------------|
| 1         | 1001 Laptop Lenovo      | 3500   | 10   | 1            |
| 2         | 1002 Telefon Samsung    | 299,99 | 25   | 1            |
| 3         | 1003 Carte SQL          | 120    | 60   | 2            |
| 4         | 1004 Hanorac Adidas     | 200    | 30   | 3            |
| 5         | 1005 Robot de bucătărie | 480    | 12   | 5            |
| 6         | 1006 Jucărie LEGO       | 150    | 20   | 4            |
| 7         | 1007 Televizor Samsung  | 1800   | 7    | 1            |
| 8         | 1008 Cărți Java         | 100    | 80   | 2            |
| 9         | 1009 Rochie elegantă    | 300    | 15   | 3            |
| 10        | 1010 Frigider Arctic    | 2000   | 5    | 5            |

## 7. Structuri de Control

### 1. IF...THEN...ELSIF... cu stoc și preț

#### EX 1: Verifică dacă un produs este disponibil și clasifică prețul

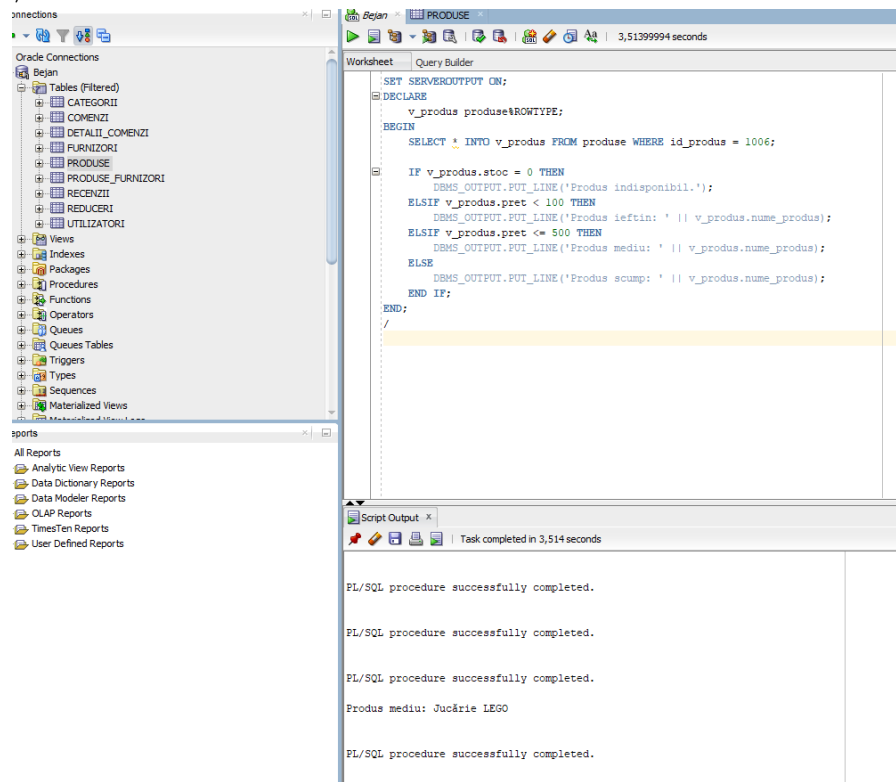
```
DECLARE
 v_produs produse%ROWTYPE;
BEGIN
 SELECT * INTO v_produs FROM produse WHERE id_produs = 1006;

 IF v_produs.stoc = 0 THEN
 DBMS_OUTPUT.PUT_LINE('Produs indisponibil.');
```

ELSIF v\_produs.pret < 100 THEN

```
 DBMS_OUTPUT.PUT_LINE('Produs ieftin: ' ||
v_produs.ume_produs);
 ELSIF v_produs.pret <= 500 THEN
 DBMS_OUTPUT.PUT_LINE('Produs mediu: ' ||
v_produs.ume_produs);
 ELSE
 DBMS_OUTPUT.PUT_LINE('Produs scump: ' ||
v_produs.ume_produs);
 END IF;
END;
```

/

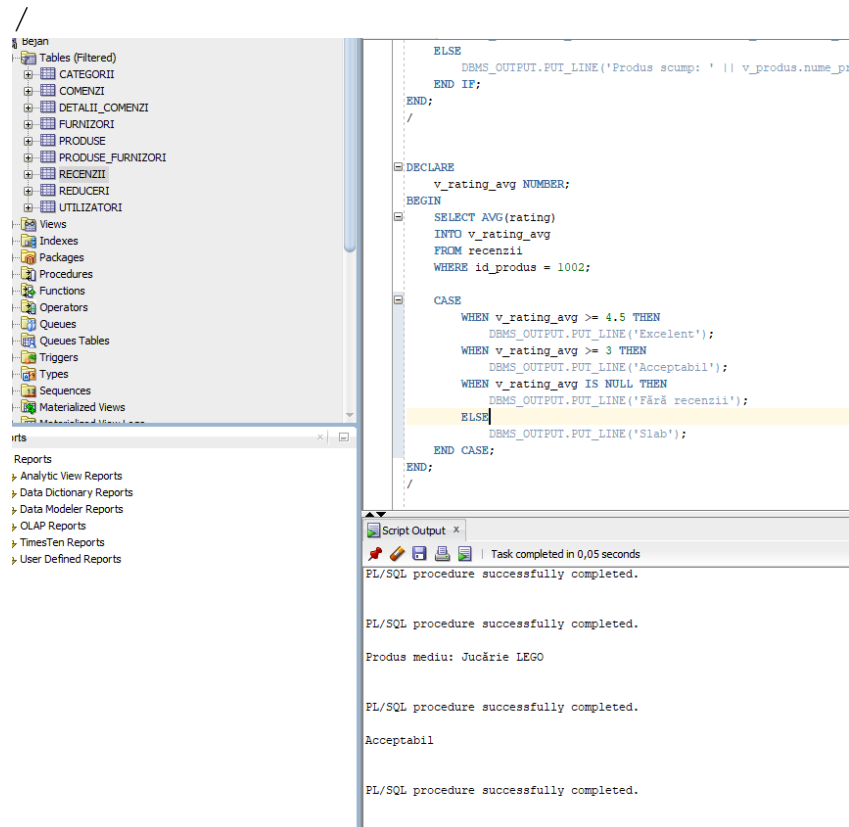


## 2. CASE...WHEN pe baza ratingului mediu al unui produs

### EX 1: Evaluează produsul în funcție de recenzii

```
DECLARE
 v_rating_avg NUMBER;
BEGIN
 SELECT AVG(rating)
 INTO v_rating_avg
 FROM recenzii
 WHERE id_produs = 1002;

 CASE
 WHEN v_rating_avg >= 4.5 THEN
 DBMS_OUTPUT.PUT_LINE('Excelent');
 WHEN v_rating_avg >= 3 THEN
 DBMS_OUTPUT.PUT_LINE('Acceptabil');
 WHEN v_rating_avg IS NULL THEN
 DBMS_OUTPUT.PUT_LINE('Fara recenzii');
 ELSE
 DBMS_OUTPUT.PUT_LINE('Slab');
 END CASE;
END;
```



The screenshot displays the Oracle SQL Developer environment. On the left, a tree view shows the database schema with tables like CATEGORII, COMENZI, DETALII\_COMENZI, FURNIZORI, PRODUSE, PRODUSE\_FURNIZORI, RECENZII, REDUCERI, and UTILIZATORI. The main editor window contains the PL/SQL script from the previous block. Below the editor, the 'Script Output' window shows the execution results:

```
Task completed in 0,05 seconds
PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

Produs mediu: Jucărie LEGO

PL/SQL procedure successfully completed.

Acceptabil

PL/SQL procedure successfully completed.
```

### 3. LOOP..END LOOP cu căutare produs pe stoc > X și preț < Y

#### EX 1: Caută primul produs disponibil sub 500 lei și cu stoc ≥ 10

```
DECLARE
 v_id NUMBER := 1001;
 v_pret NUMBER;
 v_stoc NUMBER;
BEGIN
 LOOP
 EXIT WHEN v_id > 1010;

 BEGIN
 SELECT pret, stoc INTO v_pret, v_stoc
 FROM produse WHERE id_produs = v_id;

 IF v_pret < 500 AND v_stoc >= 10 THEN
 DBMS_OUTPUT.PUT_LINE('Găsit: produs ' || v_id ||
', pret: ' || v_pret || ', stoc: ' || v_stoc);
 EXIT;
 END IF;
 EXCEPTION
 WHEN NO_DATA_FOUND THEN NULL;
 END;

 v_id := v_id + 1;
 END LOOP;
END;
```

The screenshot displays the Oracle SQL Developer environment. On the left, the 'Connections' pane shows the 'Bejan' connection. The 'Tables (Filtered)' pane lists various tables including CATEGORII, COMENZI, DETALII\_COMENZI, FURNIZORI, and PRODUSE. The main window shows the SQL script being executed, which is the same PL/SQL code provided in the previous block. The bottom pane shows the output of the script, indicating that the procedure completed successfully and found the first product meeting the criteria: 'Găsit: produs 1002, pret: 299,99, stoc: 25'.



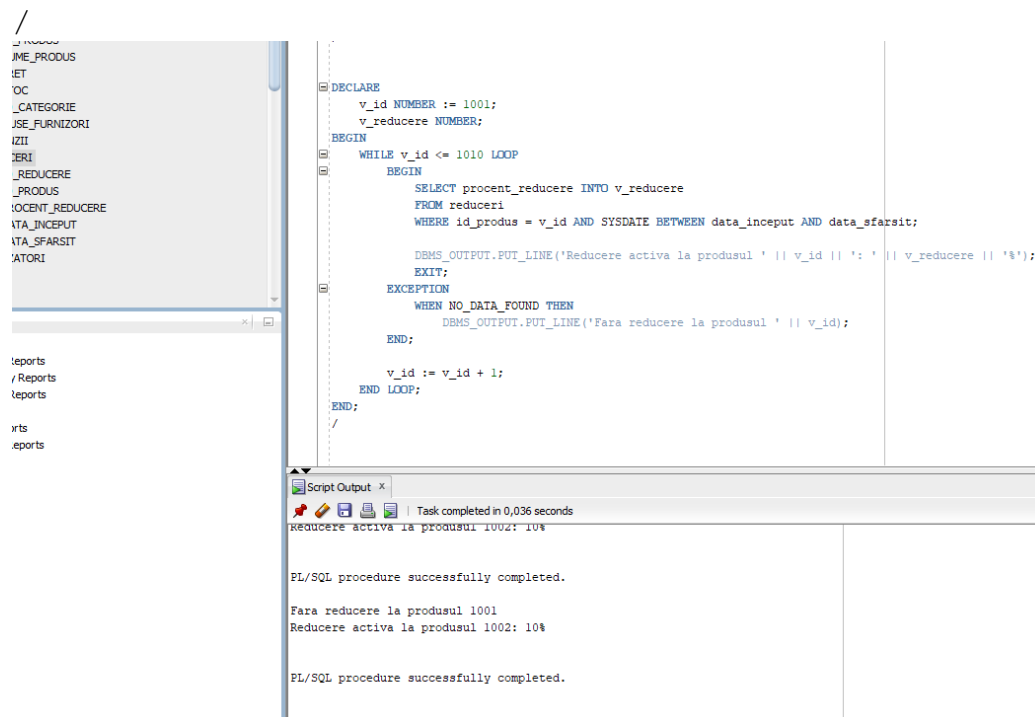
## 5. WHILE..LOOP...END LOOP cu bloc imbricat și etichetă

### EX 1: Caută produse și oprește la prima reducere activă

```
DECLARE
 v_id NUMBER := 1001;
 v_reducere NUMBER;
BEGIN
 WHILE v_id <= 1010 LOOP
 BEGIN
 SELECT procent_reducere INTO v_reducere
 FROM reduceri
 WHERE id_produs = v_id AND SYSDATE BETWEEN
 data_inceput AND data_sfarsit;

 DBMS_OUTPUT.PUT_LINE('Reducere activa la produsul '
 || v_id || ': ' || v_reducere || '%');
 EXIT;
 EXCEPTION
 WHEN NO_DATA_FOUND THEN
 DBMS_OUTPUT.PUT_LINE('Fara reducere la produsul
 ' || v_id);
 END;

 v_id := v_id + 1;
 END LOOP;
END;
```



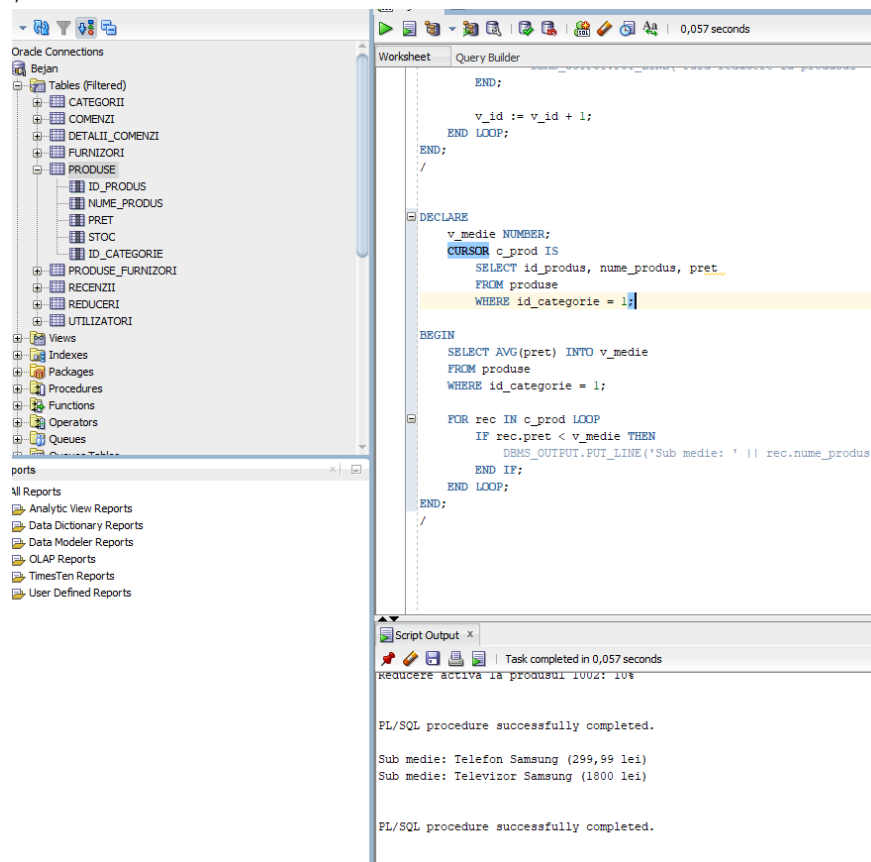
## 6. FOR LOOP cu cursor și SELECT

### EX 1: Afișează produsele din categoria 1 cu preț sub media categoriei

```
DECLARE
 v_medie NUMBER;
 CURSOR c_prod IS
 SELECT id_produș, nume_produș, pret
 FROM produse
 WHERE id_categorie = 1;

BEGIN
 SELECT AVG(pret) INTO v_medie
 FROM produse
 WHERE id_categorie = 1;

 FOR rec IN c_prod LOOP
 IF rec.pret < v_medie THEN
 DBMS_OUTPUT.PUT_LINE('Sub medie: ' ||
rec.nume_produș || ' (' || rec.pret || ' lei)');
 END IF;
 END LOOP;
END;
/
```



The screenshot displays the Oracle SQL Developer interface. On the left, the 'Oracle Connections' tree shows the 'Bejan' database with various tables and views. The main window shows a PL/SQL script being executed. The script declares a variable `v_medie` and a cursor `c_prod` to select products from category 1. It then calculates the average price and iterates through the products, displaying those with a price below the average. The 'Script Output' pane at the bottom shows the execution results, including the completion message and the output lines: 'Sub medie: Telefon Samsung (299,99 lei)' and 'Sub medie: Televizor Samsung (1800 lei)'.

```
END;

v_id := v_id + 1;
END LOOP;
END;
/

DECLARE
 v_medie NUMBER;
 CURSOR c_prod IS
 SELECT id_produș, nume_produș, pret
 FROM produse
 WHERE id_categorie = 1;

BEGIN
 SELECT AVG(pret) INTO v_medie
 FROM produse
 WHERE id_categorie = 1;

 FOR rec IN c_prod LOOP
 IF rec.pret < v_medie THEN
 DBMS_OUTPUT.PUT_LINE('Sub medie: ' || rec.nume_produș |
 END IF;
 END LOOP;
END;
/
```

Script Output x

Task completed in 0,057 seconds

Reducere activa la produsul 1002: 10%

PL/SQL procedure successfully completed.

Sub medie: Telefon Samsung (299,99 lei)  
Sub medie: Televizor Samsung (1800 lei)

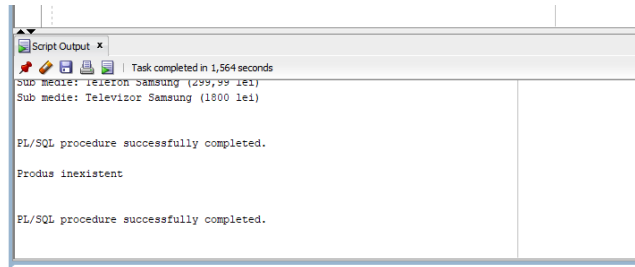
PL/SQL procedure successfully completed.

## 7. Tratarea Exceptiilor

### 1. EXCEPȚII IMPLICITE

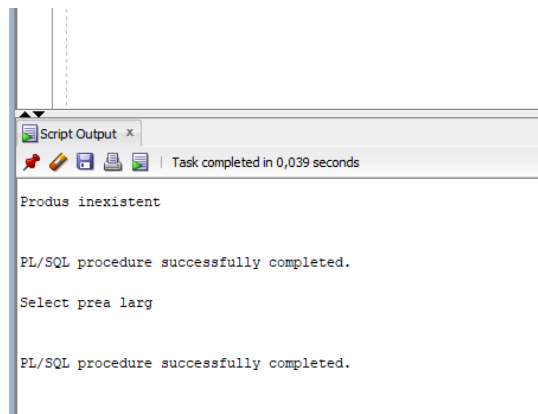
#### EX 1: NO\_DATA\_FOUND

```
DECLARE
 v_nume produse.nume_produc%TYPE;
BEGIN
 SELECT nume_produc INTO v_nume FROM produse WHERE id_produc
= 9999;
 DBMS_OUTPUT.PUT_LINE(v_nume);
EXCEPTION
 WHEN NO_DATA_FOUND THEN
 DBMS_OUTPUT.PUT_LINE('Produs inexistent');
END;
/
```



#### EX 2: TOO\_MANY\_ROWS

```
DECLARE
 v_nume produse.nume_produc%TYPE;
BEGIN
 SELECT nume_produc INTO v_nume FROM produse WHERE
id_categorie = 1;
 DBMS_OUTPUT.PUT_LINE(v_nume);
EXCEPTION
 WHEN TOO_MANY_ROWS THEN
 DBMS_OUTPUT.PUT_LINE('Select prea larg');
END;
/
```



### EX 3: ZERO\_DIVIDE

```
DECLARE
 v_x NUMBER := 10;
 v_y NUMBER := 0;
 v_r NUMBER;
BEGIN
 v_r := v_x / v_y;
 DBMS_OUTPUT.PUT_LINE(v_r);
EXCEPTION
 WHEN ZERO_DIVIDE THEN
 DBMS_OUTPUT.PUT_LINE('Impartire la zero');
END;
/
```

UTILIZATORI

vs

axes

pages

cedures

ctions

rators

ues

use Tables

View Reports

tionary Reports

delers Reports

ports

n Reports

ined Reports

```
 WHEN TOO_MANY_ROWS THEN
 DBMS_OUTPUT.PUT_LINE('Select prea larg');
END;
/
DECLARE
 v_x NUMBER := 10;
 v_y NUMBER := 0;
 v_r NUMBER;
BEGIN
 v_r := v_x / v_y;
 DBMS_OUTPUT.PUT_LINE(v_r);
EXCEPTION
 WHEN ZERO_DIVIDE THEN
 DBMS_OUTPUT.PUT_LINE('Impartire la zero');
END;
/
```

Script Output x

Task completed in 0,186 seconds

Împăr?ire la zero

PL/SQL procedure successfully completed.

Impartire la zero

PL/SQL procedure successfully completed.

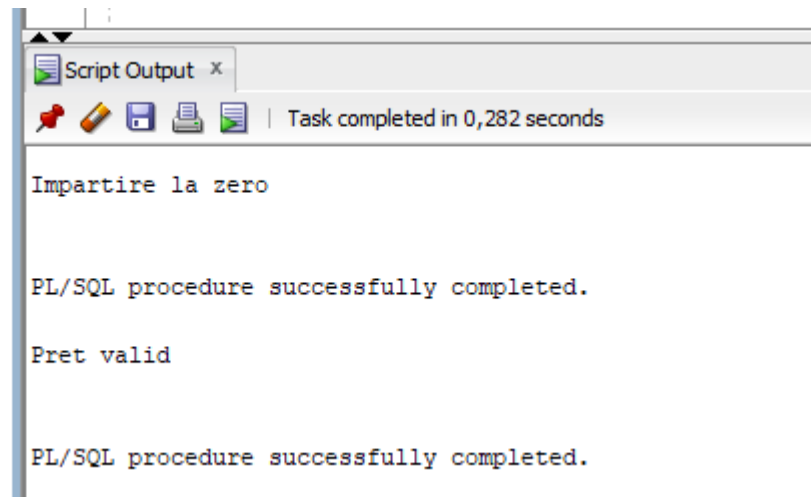
## 2. EXCEPȚII IMPLICITE

### EX 1: RAISE cu excepție definită

```
DECLARE
 e_invalid EXCEPTION;
 v_pret produse.pret%TYPE;
BEGIN
 SELECT pret INTO v_pret FROM produse WHERE id_produș
 = 1001;

 IF v_pret < 0 THEN
 RAISE e_invalid;
 END IF;

 DBMS_OUTPUT.PUT_LINE('Pret valid');
EXCEPTION
 WHEN e_invalid THEN
 DBMS_OUTPUT.PUT_LINE('Pret invalid');
END;
/
```



### EX 2: RAISE\_APPLICATION\_ERROR cu cod și mesaj

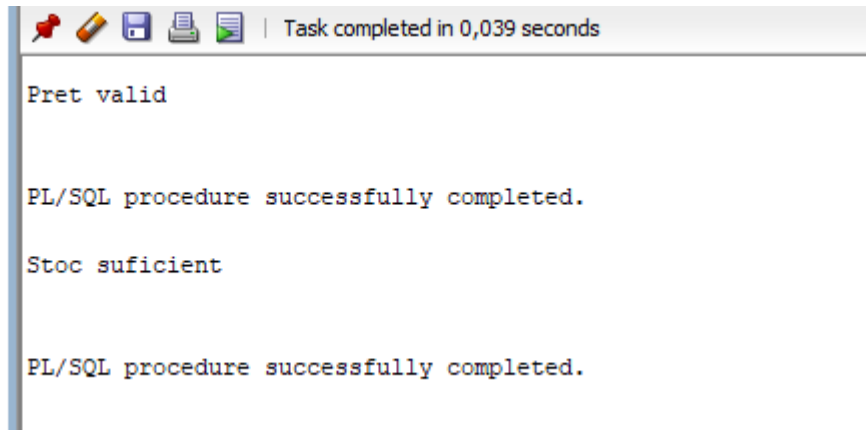
```
DECLARE
 v_stoc produse.stoc%TYPE;
BEGIN
 SELECT stoc INTO v_stoc FROM produse WHERE id_produș =
 1003;

 IF v_stoc < 5 THEN
 RAISE_APPLICATION_ERROR(-20002, 'Stoc critic');
 END IF;
```

```

 DBMS_OUTPUT.PUT_LINE('Stoc suficient');
EXCEPTION
 WHEN OTHERS THEN
 DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/

```

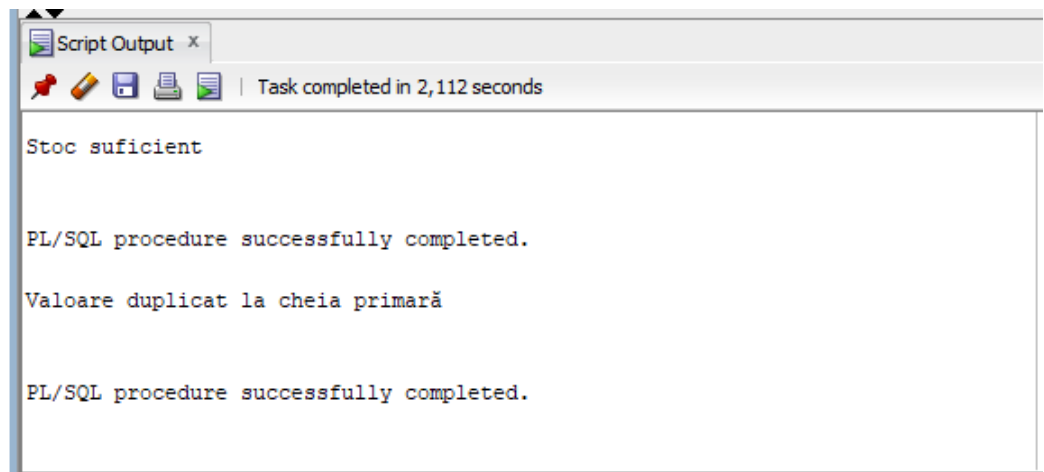


### EX 3: EXCEPTION\_INIT cu cod Oracle

```

DECLARE
 e_constraint EXCEPTION;
 PRAGMA EXCEPTION_INIT(e_constraint, -00001);
BEGIN
 INSERT INTO utilizatori VALUES (1, 'user01', 'mail', 'pw',
 'nume', SYSDATE);
EXCEPTION
 WHEN e_constraint THEN
 DBMS_OUTPUT.PUT_LINE('Valoare duplicat la cheia
 primară');
END;
/

```

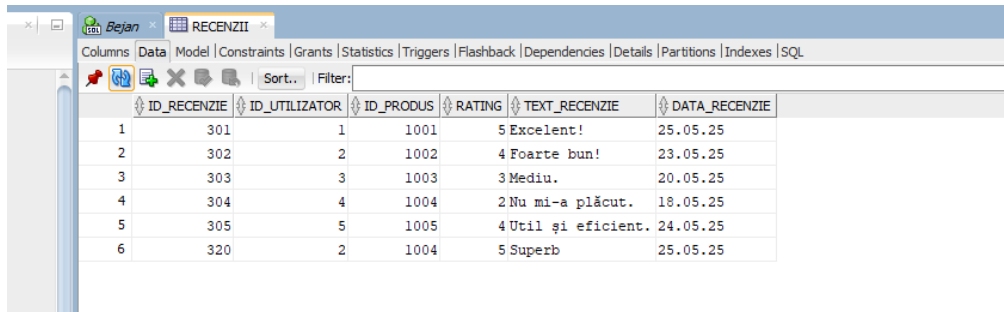


## 8. Gestionarea Cursorilor

### 1. CURSORI IMPLICITI

#### EX 1: INSERT cu validare și rollback dacă nu s-a inserat

```
BEGIN
 INSERT INTO recenzii VALUES (320, 2, 1004, 5, 'Superb',
 SYSDATE);
 IF SQL%ROWCOUNT = 0 THEN
 ROLLBACK;
 ELSE
 DBMS_OUTPUT.PUT_LINE('Inserare reusita');
 END IF;
END;
/
```

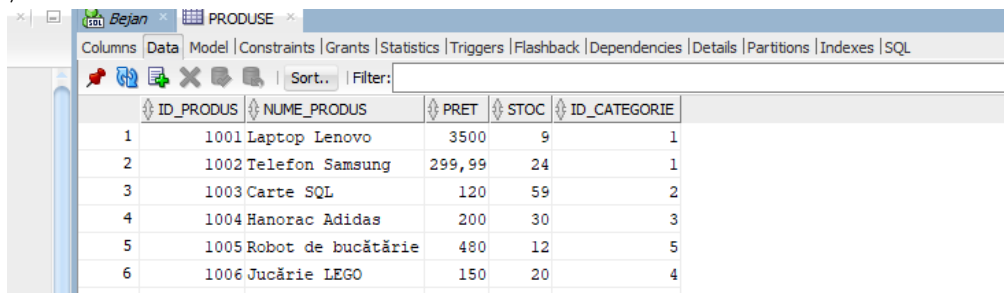


| ID_RECENZIE | ID_UTILIZATOR | ID_PRODUS | RATING | TEXT_RECENZIE       | DATA_RECENZIE |
|-------------|---------------|-----------|--------|---------------------|---------------|
| 1           | 301           | 1         | 1001   | 5 Excelent!         | 25.05.25      |
| 2           | 302           | 2         | 1002   | 4 Foarte bun!       | 23.05.25      |
| 3           | 303           | 3         | 1003   | 3 Mediu.            | 20.05.25      |
| 4           | 304           | 4         | 1004   | 2 Nu mi-a plăcut.   | 18.05.25      |
| 5           | 305           | 5         | 1005   | 4 Util și eficient. | 24.05.25      |
| 6           | 320           | 2         | 1004   | 5 Superb            | 25.05.25      |

#### EX 2: UPDATE + verificare multiplă (SQL%ROWCOUNT și SQL%FOUND)

```
BEGIN
 UPDATE produse
 SET stoc = stoc - 1
 WHERE id_produc IN (1001, 1002, 1003);

 IF SQL%FOUND AND SQL%ROWCOUNT > 1 THEN
 DBMS_OUTPUT.PUT_LINE('Stoc actualizat la ' ||
 SQL%ROWCOUNT || ' produse');
 ELSE
 DBMS_OUTPUT.PUT_LINE('Nicio actualizare semnificativa');
 END IF;
END;
/
```

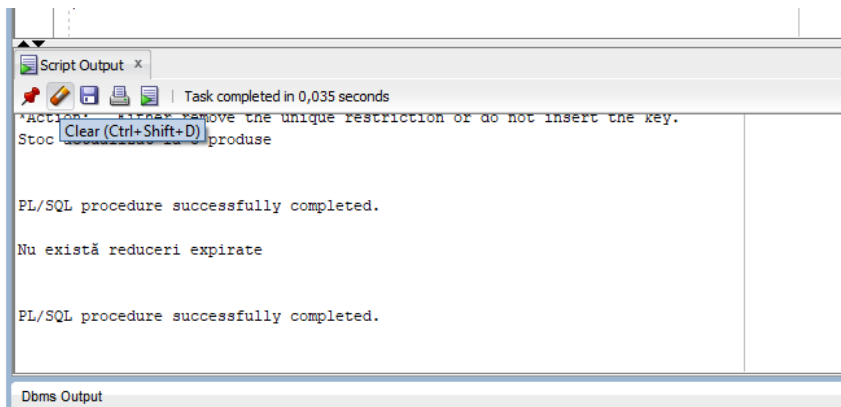


| ID_PRODUS | NUME_PRODUS             | PRET   | STOC | ID_CATEGORIE |
|-----------|-------------------------|--------|------|--------------|
| 1         | 1001 Laptop Lenovo      | 3500   | 9    | 1            |
| 2         | 1002 Telefon Samsung    | 299,99 | 24   | 1            |
| 3         | 1003 Carte SQL          | 120    | 59   | 2            |
| 4         | 1004 Hanorac Adidas     | 200    | 30   | 3            |
| 5         | 1005 Robot de bucătărie | 480    | 12   | 5            |
| 6         | 1006 Jucărie LEGO       | 150    | 20   | 4            |

### EX 3: DELETE cu mesaje condiționate

```
BEGIN
 DELETE FROM reduceri WHERE SYSDATE > data_sfarsit;

 IF SQL%ROWCOUNT = 0 THEN
 DBMS_OUTPUT.PUT_LINE('Nu există reduceri expirate');
 ELSIF SQL%ROWCOUNT > 5 THEN
 DBMS_OUTPUT.PUT_LINE('Au fost eliminate ' ||
SQL%ROWCOUNT || ' reduceri expirate');
 ELSE
 DBMS_OUTPUT.PUT_LINE('Reducerile au fost sterse');
 END IF;
END;
```

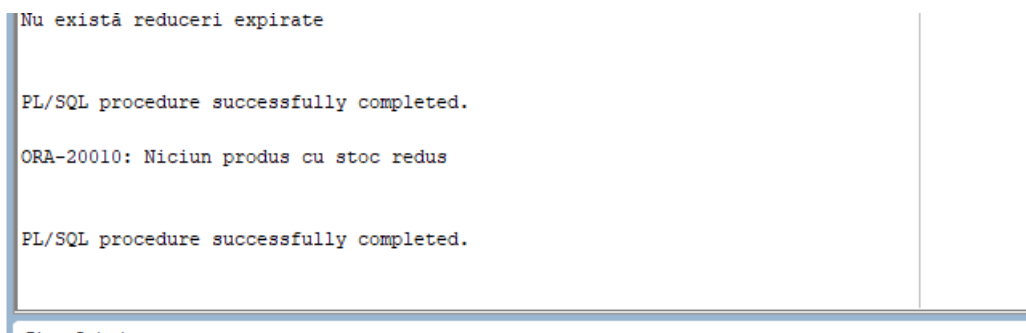


/

### EX 4: UPDATE + EXCEPTION integrat

```
BEGIN
 UPDATE produse SET pret = pret * 0.9 WHERE stoc < 5;
 IF SQL%ROWCOUNT = 0 THEN
 RAISE_APPLICATION_ERROR(-20010, 'Niciun produs cu stoc
redus');
 END IF;
EXCEPTION
 WHEN OTHERS THEN
 DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
```

/

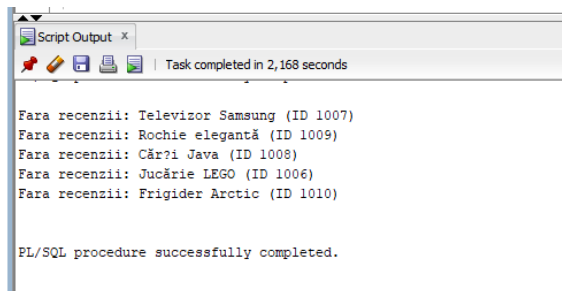




## 1. CURSORI EXPLICITI

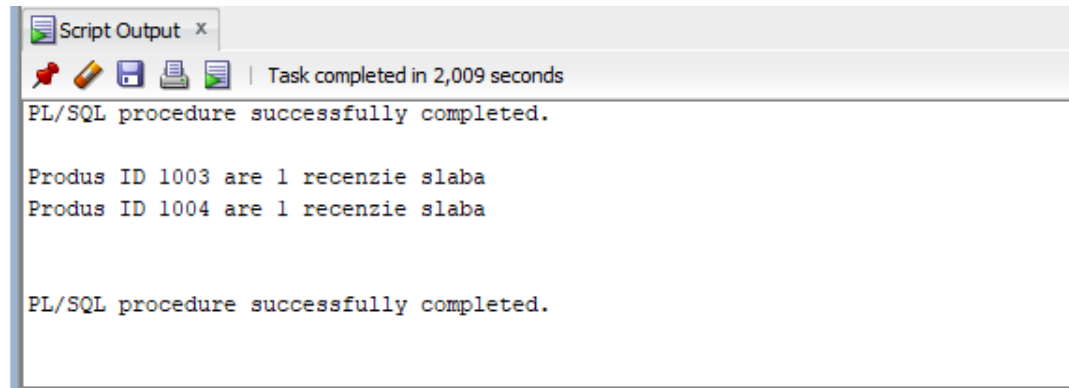
### EX 1: OPEN-FETCH-CLOSE, produse fără recenzii

```
DECLARE
 CURSOR c1 IS
 SELECT p.id_produc, p.nume_produc
 FROM produse p
 WHERE NOT EXISTS (
 SELECT 1 FROM recenzii r WHERE r.id_produc =
p.id_produc
);
 v_id produse.id_produc%TYPE;
 v_nume produse.nume_produc%TYPE;
BEGIN
 OPEN c1;
 LOOP
 FETCH c1 INTO v_id, v_nume;
 EXIT WHEN c1%NOTFOUND;
 DBMS_OUTPUT.PUT_LINE('Fara recenzii: ' || v_nume || '
(ID ' || v_id || ')');
 END LOOP;
 CLOSE c1;
END;
/
```



### EX 2: FOR...IN, produse cu mai multe recenzii negative

```
DECLARE
 CURSOR c2 IS
 SELECT id_produc, COUNT(*) nr
 FROM recenzii
 WHERE rating <= 2
 GROUP BY id_produc
 HAVING COUNT(*) >= 1;
BEGIN
 FOR rec IN c2 LOOP
 DBMS_OUTPUT.PUT_LINE('Produs ID ' || rec.id_produc || '
are ' || rec.nr || ' recenzie slaba');
 END LOOP;
END;
/
```



```
Script Output x
Task completed in 2,009 seconds

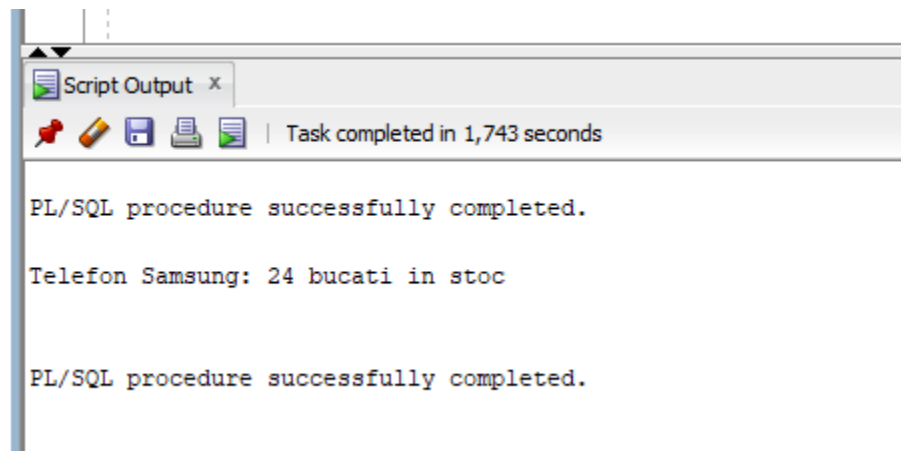
PL/SQL procedure successfully completed.

Produs ID 1003 are 1 recenzie slaba
Produs ID 1004 are 1 recenzie slaba

PL/SQL procedure successfully completed.
```

### EX 3: Cu parametru, și filtrare după stoc și categorie

```
DECLARE
 CURSOR c3(p_categ NUMBER, p_min_stoc NUMBER) IS
 SELECT id_produc, nume_produc, stoc
 FROM produse
 WHERE id_categorie = p_categ AND stoc >= p_min_stoc;
 v_id produse.id_produc%TYPE;
 v_nume produse.nume_produc%TYPE;
 v_stoc produse.stoc%TYPE;
BEGIN
 OPEN c3(1, 15);
 LOOP
 FETCH c3 INTO v_id, v_nume, v_stoc;
 EXIT WHEN c3%NOTFOUND;
 DBMS_OUTPUT.PUT_LINE(v_nume || ': ' || v_stoc || '
bucati in stoc');
 END LOOP;
 CLOSE c3;
END;
/
```



```
Script Output x
Task completed in 1,743 seconds

PL/SQL procedure successfully completed.

Telefon Samsung: 24 bucati in stoc

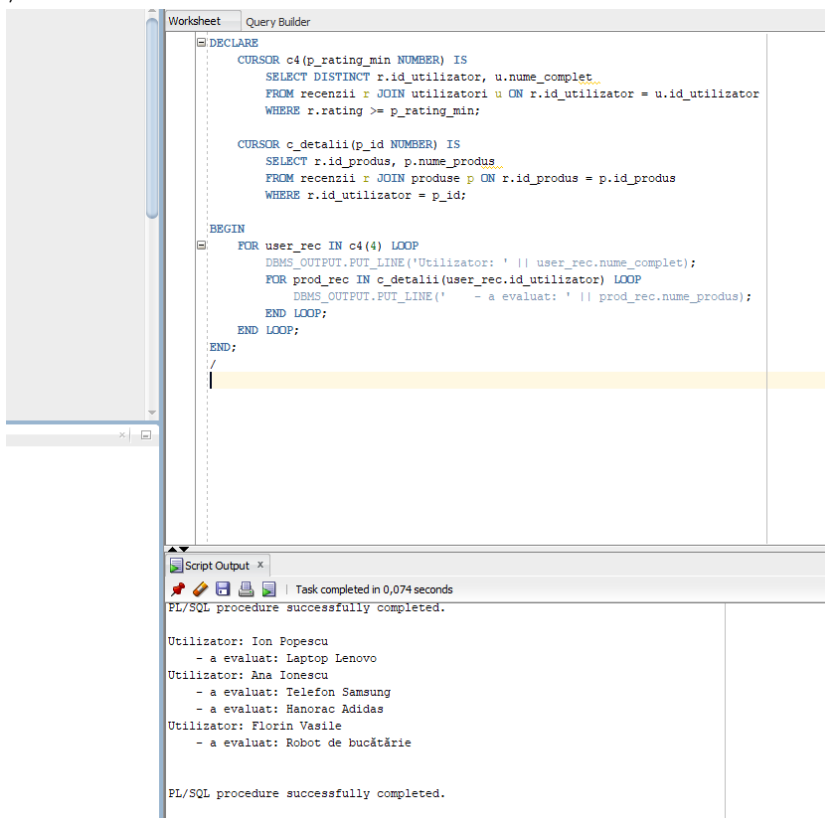
PL/SQL procedure successfully completed.
```

#### EX 4: Cu parametru și cursor imbricat

```
DECLARE
 CURSOR c4(p_rating_min NUMBER) IS
 SELECT DISTINCT r.id_utilizator, u.ume_complet
 FROM recenzii r JOIN utilizatori u ON r.id_utilizator
 = u.id_utilizator
 WHERE r.rating >= p_rating_min;

 CURSOR c_detalii(p_id NUMBER) IS
 SELECT r.id_produs, p.ume_produs
 FROM recenzii r JOIN produse p ON r.id_produs =
 p.id_produs
 WHERE r.id_utilizator = p_id;

BEGIN
 FOR user_rec IN c4(4) LOOP
 DBMS_OUTPUT.PUT_LINE('Utilizator: ' ||
user_rec.ume_complet);
 FOR prod_rec IN c_detalii(user_rec.id_utilizator) LOOP
 DBMS_OUTPUT.PUT_LINE(' - a evaluat: ' ||
prod_rec.ume_produs);
 END LOOP;
 END LOOP;
END;
/
```



## 9. Funcții , Proceduri si includerea acestora in pachete

### 1. FUNCȚII

#### EX 1: Returnează prețul cu reducere

```
CREATE OR REPLACE FUNCTION pret_final(p_id NUMBER) RETURN NUMBER
IS
 v_pret NUMBER;
 v_red NUMBER;
BEGIN
 SELECT pret INTO v_pret FROM produse WHERE id_produc = p_id;
 SELECT NVL(procent_reducere, 0) INTO v_red
 FROM reduceri WHERE id_produc = p_id AND SYSDATE BETWEEN
data_inceput AND data_sfarsit;
 RETURN v_pret * (1 - v_red / 100);
EXCEPTION
 WHEN NO_DATA_FOUND THEN RETURN NULL;
END;
/
BEGIN
 DBMS_OUTPUT.PUT_LINE('Pret final: ' || pret_final(1002));
END;
/
```

```
Function PRET_FINAL compiled
Pret final: 269,991
PL/SQL procedure successfully completed.
```

#### EX 2: Returnează numărul de recenzii pentru un produs

```
CREATE OR REPLACE FUNCTION nr_recenzii(p_id NUMBER) RETURN
NUMBER IS
 v_nr NUMBER;
BEGIN
 SELECT COUNT(*) INTO v_nr FROM recenzii WHERE id_produc =
p_id;
 RETURN v_nr;
END;
/
```

```
Function NR_RECENZII compiled

Nr recenzii: 1

PL/SQL procedure successfully completed.
```

### ***EX 3: Returnează utilizatorul cu cele mai multe comenzi***

```
CREATE OR REPLACE FUNCTION utilizator_fidel RETURN VARCHAR2 IS
 v_nume utilizatori.ume_complet%TYPE;
BEGIN
 SELECT u.ume_complet INTO v_nume
 FROM utilizatori u JOIN comenzi c ON u.id_utilizator =
c.id_utilizator
 GROUP BY u.ume_complet
 ORDER BY COUNT(*) DESC FETCH FIRST 1 ROWS ONLY;
 RETURN v_nume;
END;
/
```

```
BEGIN
 DBMS_OUTPUT.PUT_LINE('Utilizator fidel: ' ||
utilizator_fidel);
END;
/
```

```
Function UTILIZATOR_FIDEL compiled

Utilizator fidel: Ion Popescu

PL/SQL procedure successfully completed.
```

### ***EX 4: Returnează media prețurilor pe categorie***

```
CREATE OR REPLACE FUNCTION medie_pret(p_categ NUMBER) RETURN
NUMBER IS
 v_avg NUMBER;
BEGIN
 SELECT AVG(pret) INTO v_avg FROM produse WHERE
id_categorie = p_categ;
 RETURN v_avg;
END;
/
```

```
BEGIN
 DBMS_OUTPUT.PUT_LINE('Media preturilor: ' ||
medie_pret(1));
END;
/
```



|    | ID_PRODUS | NUME_PRODUS        | PRET   | STOC | ID_CATEGORIE |
|----|-----------|--------------------|--------|------|--------------|
| 1  | 1001      | Laptop Lenovo      | 3500   | 99   | 1            |
| 2  | 1002      | Telefon Samsung    | 299,99 | 24   | 1            |
| 3  | 1003      | Carte SQL          | 120    | 59   | 2            |
| 4  | 1004      | Hanorac Adidas     | 200    | 30   | 3            |
| 5  | 1005      | Robot de bucătărie | 480    | 12   | 5            |
| 6  | 1006      | Jucărie LEGO       | 150    | 20   | 4            |
| 7  | 1007      | Televizor Samsung  | 1800   | 7    | 1            |
| 8  | 1008      | Cărți Java         | 100    | 80   | 2            |
| 9  | 1009      | Rochie elegantă    | 300    | 15   | 3            |
| 10 | 1010      | Frigider Arctic    | 2000   | 5    | 5            |

### EX 3: Afișează toate recenziile unui produs

```

CREATE OR REPLACE PROCEDURE recenzii_produs(p_id NUMBER) IS
BEGIN
 FOR rec IN (SELECT rating, text_recenzie FROM recenzii WHERE
id_produs = p_id) LOOP
 DBMS_OUTPUT.PUT_LINE(rec.rating || ' - ' ||
rec.text_recenzie);
 END LOOP;
END;
/
BEGIN
 recenzii_produs(1003);
END;
/

```

```

Task completed in 0,165 seconds
PL/SQL procedure successfully completed.

Procedure RECENZII_PRODUS compiled

3 - Mediu.

PL/SQL procedure successfully completed.

```

### EX 4: Șterge comenzile fără detalii

```

CREATE OR REPLACE PROCEDURE sterge_comenzi_goale IS
BEGIN
 DELETE FROM comenzi
 WHERE id_comanda NOT IN (SELECT DISTINCT id_comanda FROM
detalii_comenzi);
 COMMIT;
END;
/

BEGIN
 sterge_comenzi_goale;
END;
/

```

| Columns   Data   Model   Constraints   Grants   Statistics   Triggers   Flashback   Dependencies   Details   Part |            |               |              |             |
|-------------------------------------------------------------------------------------------------------------------|------------|---------------|--------------|-------------|
| Sort..   Filter:                                                                                                  |            |               |              |             |
|                                                                                                                   | ID_COMANDA | ID_UTILIZATOR | DATA_COMANDA | TOTAL_PLATA |
| 1                                                                                                                 | 5001       |               | 1 15.05.25   | 3700        |
| 2                                                                                                                 | 5002       |               | 2 20.05.25   | 300         |
| 3                                                                                                                 | 5003       |               | 3 22.05.25   | 2700        |
| 4                                                                                                                 | 5004       |               | 4 24.05.25   | 620         |
| 5                                                                                                                 | 5005       |               | 5 25.05.25   | 2100        |

### 3. PACHETE

**EX 1: Acest pachet oferă un set de operații esențiale pentru gestionarea produselor din baza de date.**

```
CREATE OR REPLACE PACKAGE gestiune_produce IS
 PROCEDURE adauga(p_id NUMBER, p_nume VARCHAR2, p_pret
NUMBER, p_stoc NUMBER, p_cat NUMBER);
 PROCEDURE actualizare_stoc(p_id NUMBER, p_nou_stoc NUMBER);
 FUNCTION exista(p_id NUMBER) RETURN BOOLEAN;
END;
/
```

```
CREATE OR REPLACE PACKAGE BODY gestiune_produce IS
 PROCEDURE adauga(p_id NUMBER, p_nume VARCHAR2, p_pret
NUMBER, p_stoc NUMBER, p_cat NUMBER) IS
 BEGIN
 INSERT INTO produse VALUES (p_id, p_nume, p_pret,
p_stoc, p_cat);
 END;

 PROCEDURE actualizare_stoc(p_id NUMBER, p_nou_stoc NUMBER)
IS
 BEGIN
 UPDATE produse SET stoc = p_nou_stoc WHERE id_produc =
p_id;
 END;

 FUNCTION exista(p_id NUMBER) RETURN BOOLEAN IS
 v_dummy NUMBER;
 BEGIN
 SELECT 1 INTO v_dummy FROM produse WHERE id_produc =
p_id;
 RETURN TRUE;
 EXCEPTION
 WHEN NO_DATA_FOUND THEN RETURN FALSE;
 END;
END;
/
```



```

BEGIN
 gestiune_produce.adauga(1015, 'Tastatura Gaming', 200, 50,
1);
END;
/

```

```

BEGIN
 gestiune_produce.actualizare_stoc(1015, 75);
END;
/

```

```

BEGIN
 IF gestiune_produce.exista(1015) THEN
 DBMS_OUTPUT.PUT_LINE('Produsul exista.');
```

```

ELSE
 DBMS_OUTPUT.PUT_LINE('Produsul NU exista.');
```

```

END IF;
END;
/

```

|    | ID_PRODUS | NUME_PRODUS        | PRET   | STOC | ID_CATEGORIE |
|----|-----------|--------------------|--------|------|--------------|
| 1  | 1001      | Laptop Lenovo      | 3500   | 99   | 1            |
| 2  | 1002      | Telefon Samsung    | 299,99 | 24   | 1            |
| 3  | 1003      | Carte SQL          | 120    | 59   | 2            |
| 4  | 1004      | Hanorac Adidas     | 200    | 30   | 3            |
| 5  | 1005      | Robot de bucătărie | 480    | 12   | 5            |
| 6  | 1006      | Jucărie LEGO       | 150    | 20   | 4            |
| 7  | 1007      | Televizor Samsung  | 1800   | 7    | 1            |
| 8  | 1008      | Cărți Java         | 100    | 80   | 2            |
| 9  | 1009      | Rochie elegantă    | 300    | 15   | 3            |
| 10 | 1010      | Frigider Arctic    | 2000   | 5    | 5            |
| 11 | 1015      | Tastatura Gaming   | 200    | 75   | 1            |

**EX 2: Acest pachet oferă funcții utile pentru analiza utilizatorilor în contextul comenzilor realizate.**

```

CREATE OR REPLACE PACKAGE info_utilizatori IS
 FUNCTION comenzi_total(p_id NUMBER) RETURN NUMBER;
 FUNCTION exista_utilizator(p_id NUMBER) RETURN BOOLEAN;
END;
/

```

```

CREATE OR REPLACE PACKAGE BODY info_utilizatori IS

```

```

 FUNCTION comenzi_total(p_id NUMBER) RETURN NUMBER IS
 v_total NUMBER;
 BEGIN
 SELECT COUNT(*) INTO v_total FROM comenzi WHERE
id_utilizator = p_id;
 RETURN v_total;
 END;

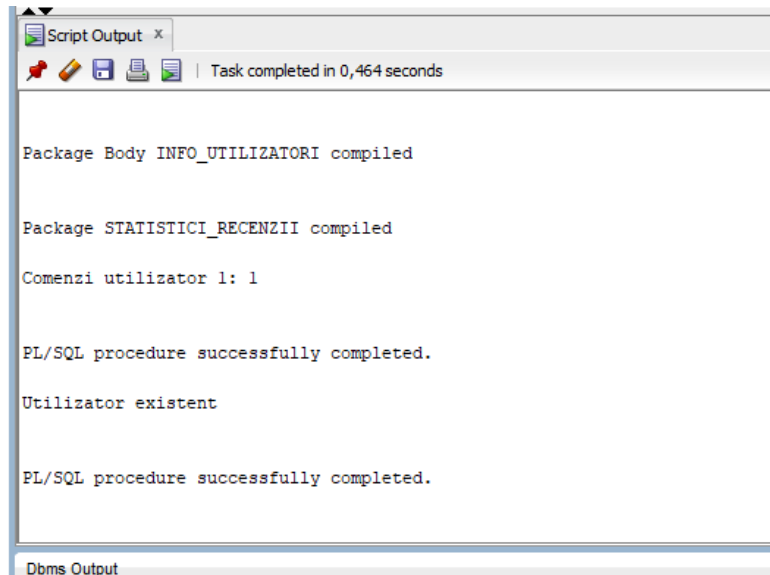
 FUNCTION exista_utilizator(p_id NUMBER) RETURN BOOLEAN IS
 v_dummy NUMBER;
 BEGIN
 SELECT 1 INTO v_dummy FROM utilizatori WHERE
id_utilizator = p_id;
 RETURN TRUE;
 EXCEPTION
 WHEN NO_DATA_FOUND THEN RETURN FALSE;
 END;
 END;
/

CREATE OR REPLACE PACKAGE statistici_recenzii IS
 FUNCTION rating_mediu(p_id NUMBER) RETURN NUMBER;
 PROCEDURE detalii_rating(p_id NUMBER);
END;
/

BEGIN
 DBMS_OUTPUT.PUT_LINE('Comenzi utilizator 1: ' ||
info_utilizatori.comenzi_total(1));
END;
/

BEGIN
 IF info_utilizatori.exista_utilizator(2) THEN
 DBMS_OUTPUT.PUT_LINE('Utilizator existent');
 ELSE
 DBMS_OUTPUT.PUT_LINE('Utilizator inexistent');
 END IF;
END;
/

```



**EX 3: Scop: Gruparea funcțiilor legate de recenziile produselor, pentru a permite atât analiza cantitativă (media) cât și calitativă (comentarii text).**

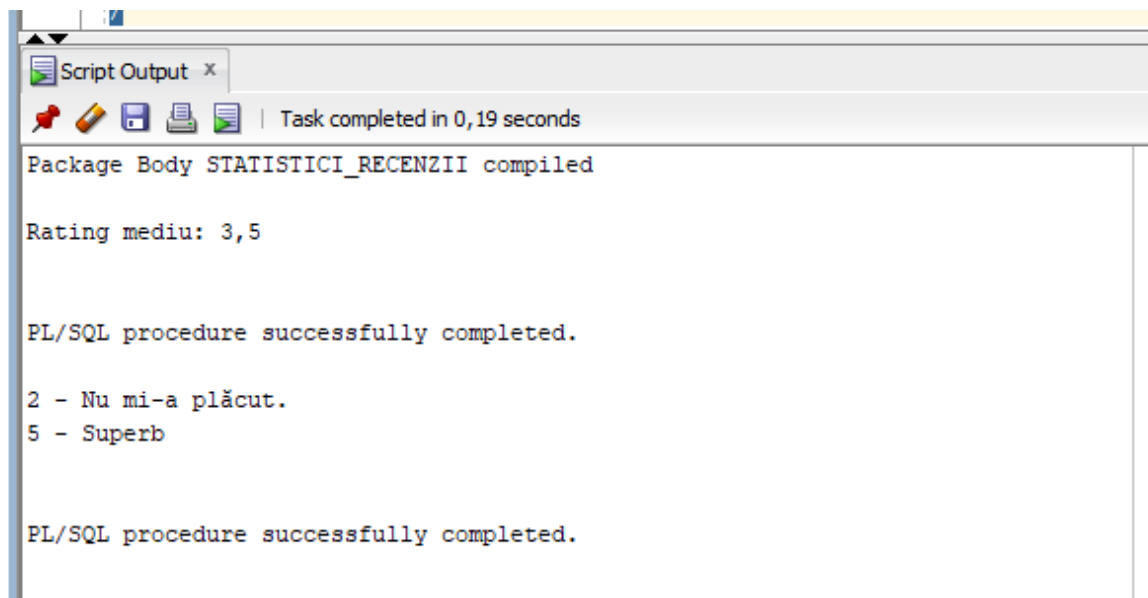
```
CREATE OR REPLACE PACKAGE statistici_recenzii IS
 FUNCTION rating_mediu(p_id NUMBER) RETURN NUMBER;
 PROCEDURE detalii_rating(p_id NUMBER);
END;
/
```

```
CREATE OR REPLACE PACKAGE BODY statistici_recenzii IS
 FUNCTION rating_mediu(p_id NUMBER) RETURN NUMBER IS
 v_avg NUMBER;
 BEGIN
 SELECT AVG(rating) INTO v_avg FROM recenzii WHERE
id_produș = p_id;
 RETURN v_avg;
 END;

 PROCEDURE detalii_rating(p_id NUMBER) IS
 BEGIN
 FOR r IN (SELECT rating, text_recenzie FROM
recenzii WHERE id_produș = p_id) LOOP
 DBMS_OUTPUT.PUT_LINE(r.rating || ' - ' ||
r.text_recenzie);
 END LOOP;
 END;
END;
/
```

```
BEGIN
 DBMS_OUTPUT.PUT_LINE('Rating mediu: ' ||
 statistici_recenzii.rating_mediu(1004));
END;
/
```

```
BEGIN
 statistici_recenzii.detalii_rating(1004);
END;
/
```



## 10. Declansatori (TRIGGERI)

### 1. BEFORE INSERT

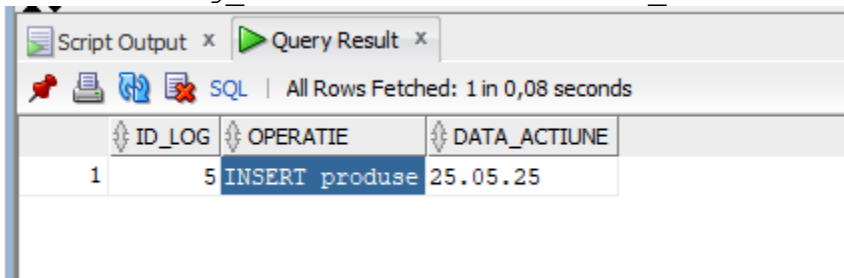
#### EX 1: Înregistrează acțiunea în LOG

```
CREATE TABLE log_actiuni (
 id_log NUMBER GENERATED BY DEFAULT AS IDENTITY,
 operatie VARCHAR2(20),
 data_actiune DATE DEFAULT SYSDATE
);
/

CREATE OR REPLACE TRIGGER trg_log_produce
BEFORE INSERT ON produce
BEGIN
 INSERT INTO log_actiuni (operatie) VALUES ('INSERT
produce');
END;
/

INSERT INTO produce (id_produc, nume_produc, pret, stoc,
id_categorie)
VALUES (1016, 'Laptop ASUS', 3100, 10, 1);

SELECT * FROM log_actiuni ORDER BY data_actiune DESC;
```



| ID_LOG | OPERATIE         | DATA_ACTIUNE |
|--------|------------------|--------------|
| 1      | 5 INSERT produce | 25.05.25     |

### 2. AFTER UPDATE

#### EX 1: Marcheaz modificări în stoc

```
CREATE TABLE stoc_modificat (
 id_produc NUMBER,
 data_modificare DATE DEFAULT SYSDATE
);
/

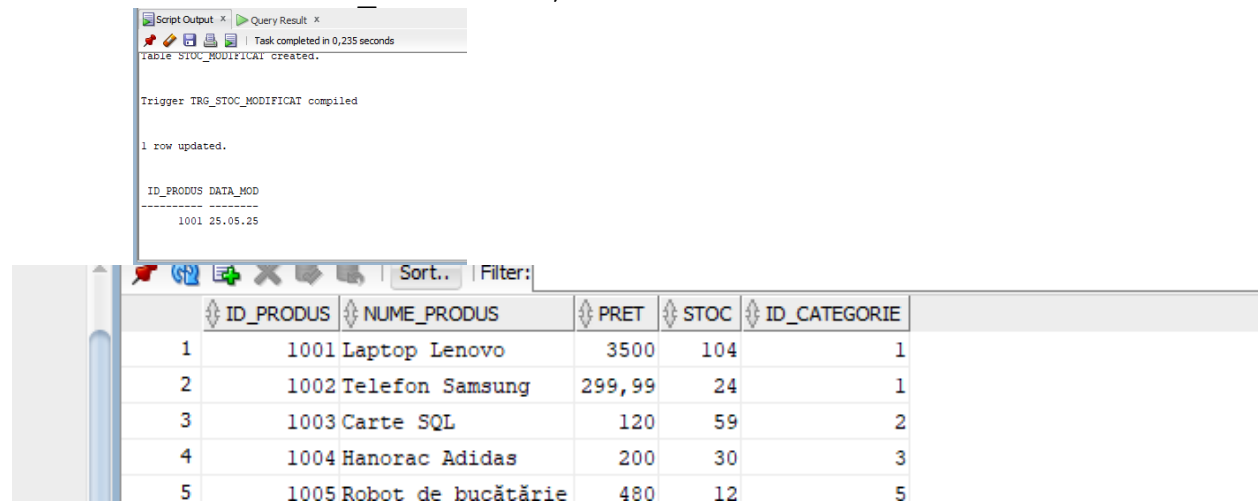
CREATE OR REPLACE TRIGGER trg_stoc_modificat
AFTER UPDATE ON produce
```

```

FOR EACH ROW
WHEN (OLD.stoc != NEW.stoc)
BEGIN
 INSERT INTO stoc_modificat (id_produs) VALUES
 (:NEW.id_produs);
END;
/

UPDATE produse
SET stoc = stoc + 5
WHERE id_produs = 1001;
/
SELECT * FROM stoc_modificat;

```



The screenshot shows a database management tool interface. At the top, a 'Script Output' window displays the following messages: 'Table STOC\_MODIFICAT created.', 'Trigger TRG\_STOC\_MODIFICAT compiled', and '1 row updated.'. Below this, a table view for 'ID\_PRODUS DATA\_MOD' shows a single row with the value '1001 25.05.25'. The main window displays a table with the following data:

|   | ID_PRODUS | NUME_PRODUS        | PRET   | STOC | ID_CATEGORIE |
|---|-----------|--------------------|--------|------|--------------|
| 1 | 1001      | Laptop Lenovo      | 3500   | 104  | 1            |
| 2 | 1002      | Telefon Samsung    | 299,99 | 24   | 1            |
| 3 | 1003      | Carte SQL          | 120    | 59   | 2            |
| 4 | 1004      | Hanorac Adidas     | 200    | 30   | 3            |
| 5 | 1005      | Robot de bucătărie | 480    | 12   | 5            |

### 3. BEFORE DELETE

#### EX 1: Validare la ștergerea produsului

```

CREATE OR REPLACE TRIGGER trg_blocare_stergere
BEFORE DELETE ON produse
FOR EACH ROW
BEGIN
 IF :OLD.stoc > 0 THEN
 RAISE_APPLICATION_ERROR(-20011, 'Nu se pot sterge
produse cu stoc pozitiv');
 END IF;
END;
/

DELETE FROM produse WHERE id_produs = 1001;
/

```

```
Task completed in 2,021 seconds

Trigger TRG_BLOCARE_STERGERE compiled

Error starting at line : 11 in command -
DELETE FROM produse WHERE id_produs = 1001
Error report -
ORA-20011: Nu se pot sterge produse cu stoc pozitiv
ORA-06512: la "BEJANI_54.TRG_BLOCARE_STERGERE", linia 3
ORA-04088: eroare în timpul execuției triggerului 'BEJANI_54.TRG_BLOCARE_STERGERE'
```

#### 4. INSTEAD OF INSERT

##### EX 1: Inserează într-un view

```
CREATE OR REPLACE VIEW v_recenzii_utilizatori AS
SELECT u.ume_complet, r.id_produs, r.rating,
r.text_recenzie
FROM utilizatori u
JOIN recenzii r ON u.id_utilizator = r.id_utilizator;
/

CREATE OR REPLACE TRIGGER trg_instead_insert
INSTEAD OF INSERT ON v_recenzii_utilizatori
FOR EACH ROW
BEGIN
 INSERT INTO recenzii (id_recenzie, id_utilizator,
id_produs, rating, text_recenzie)
VALUES (seq_rec.NEXTVAL, (SELECT id_utilizator FROM
utilizatori WHERE ume_complet = :NEW.ume_complet),
:NEW.id_produs, :NEW.rating, :NEW.text_recenzie);
END;
/

CREATE SEQUENCE seq_rec START WITH 1000 INCREMENT BY 1;
/

INSERT INTO v_recenzii_utilizatori (ume_complet,
id_produs, rating, text_recenzie)
VALUES ('Ion Popescu', 1002, 5, 'Foarte bun');
/

SELECT * FROM recenzii WHERE id_utilizator = (SELECT
id_utilizator FROM utilizatori WHERE ume_complet = 'Ion
Popescu');
```

Database management interface showing a schema tree on the left and a SQL script editor on the right. The schema tree includes tables like FURNIZORI, PRODUSE, RECENZII, and UTILIZATORI, along with Views, Indexes, Packages, Procedures, Functions, Operators, Queues, Queues Tables, Triggers, and Types.

The SQL script in the editor defines a view, a trigger, a sequence, and inserts data into the view.

```
CREATE OR REPLACE VIEW v_recenzii_utilizatori AS
SELECT u.ume_complet, r.id_produs, r.rating, r.text_recenzie
FROM utilizatori u
JOIN recenzii r ON u.id_utilizator = r.id_utilizator;
/

CREATE OR REPLACE TRIGGER trg_instead_insert
INSTEAD OF INSERT ON v_recenzii_utilizatori
FOR EACH ROW
BEGIN
INSERT INTO recenzii (id_recenzie, id_utilizator, id_produs, rating, text_recenzie)
VALUES (seq_rec.NEXTVAL, (SELECT id_utilizator FROM utilizatori WHERE nume_complet = :1), :2, :3, :4);
END;
/

CREATE SEQUENCE seq_rec START WITH 1000 INCREMENT BY 1;
/

INSERT INTO v_recenzii_utilizatori (ume_complet, id_produs, rating, text_recenzie)
VALUES ('Ion Popescu', 1002, 5, 'Foarte bun');
/

SELECT * FROM recenzii WHERE id_utilizator = (SELECT id_utilizator FROM utilizatori WHERE nume_complet = 'Ion Popescu');
```

The script output shows the sequence creation and the insertion of data into the view.

Sequence SEQ\_REC created.

1 row inserted.

| ID_RECENZIE | ID_UTILIZATOR | ID_PRODUS | RATING | TEXT_RECENZIE |
|-------------|---------------|-----------|--------|---------------|
| 301         | 1             | 1001      | 5      | Excelent!     |
| 1000        | 1             | 1002      | 5      | Foarte bun    |