Proiect baze de date

Aplicație pentru magazin online

Hăbeanu Flavius-Ștefan

Grupa 144

CUPRINS

1. Descrierea modelului real, a utilității acestuia și a regulilor de funcționare

Modelul de date gestionează informații despre o aplicație pentru un magazin online care le permite clienților să vizualizeze și să cumpere produse disponibile dintr-o gamă variată. Fiecare produs conține detalii importante, cum ar fi prețul, stocul, producătorul sau reducerea atribuită acestuia. Pentru a începe o comandă, utilizatorul trebuie să completeze câteva date de contact. Plata pentru cumpărături sau pentru transport se poate face fie cash, fie card, acestea putând beneficia de reduceri. Clientul poate lăsa o recenzie pentru orice produs cumpărat.

Modelul real are utilitatea de a oferi o experiență de cumpărare rapidă, sigură și convenabilă pentru clienți. Aceasta permite utilizatorilor să caute și să vizualizeze produsele disponibile, să plaseze comenzi, să aleagă metode de plată convenabile, să lase recenzii produselor și să beneficieze de reduceri și promoții, fie prin puncte de fidelitate, fie prin reduceri directe asupra cumpărăturilor.

Datele obținute în urma fiecărei comenzi pot fi utilizate pentru diverse statistici și sondaje folosite atât în cadrul companiei, pentru a vedea produsul cel mai vândut sau profitul anual, cât și pentru utilizatori, care vor putea accesa în conturile lor cât au economisit prin diverse reduceri sau cheltuielile anuale.

2. Prezentarea constrângerilor (restricții, reguli) impuse asupra modelului

Fiecare comandă este identificată printr-un cod unic și aparține unui singur client, are cel puțin un produs, o modalitate de plată și o metodă de transport.

Fiecare client poate plasa mai multe comenzi și are modalități de contactare, bonusuri și poate scrie o recenzie pentru un produs.

Fiecare produs are un cod unic si poate apartine mai multor comenzi.

Produsele conțin un preț, în funcție de producător și de stoc. Fiecare produs are o singură reducere activă la un moment dat.

Costul de transport trebuie să fie mereu mai mare sau egal cu 0.

Modalitatea de plată trebuie să fie doar ‘cash’ sau ‘card’.

Valoarea bonusului trebuie să fie strict pozitivă.

Ratingul unei recenzii trebuie să fie între 1 și 5.

3. Descrierea entităților, incluzând precizarea cheii primare

ENTITATE	CHEIE PRIMARĂ	OBSERVAȚII
COMANDA	ID_COMANDA	Cuprinde detalii cu privire la produsele comandate, client, transport, modalitate plată
CLIENT	ID_CLIENT	Persoana care plasează comanda
PRODUS	ID_PRODUS	Produse care pot fi comandate
TRANSPORT	ID_TRANSPORT	Detalii despre transportul comenzii
MOD_PLATA	ID_MODPLATA	Detalii despre modalitate plată
CONTACTE_CLIENT	ID_CONTACTECLIENT	Informații cu privire la modalitățile de contactare ale unui client
BONUSURI	ID_BONUSURI	Tipurile de bonusuri și informații despre acestea ale unui client
STOC	ID_PRODUS, ID_PRODUCATOR	Detalii despre disponibilitatea unui produs și preț în funcție de producător
COMANDA_PRODUS	ID_COMANDA, ID_PRODUS	Detalii despre ce produse are o comandă, cantitatea și prețul fiecăruia (în funcție de stoc)
PRODUCATOR	ID_PRODUCATOR	Detalii despre producătorul unui produs
REDUCERE	ID_REDUCERE	Reducerea aplicată unui produs
RECENZIE	ID_RECENZIE	Detalii despre recenziile lăsate de clienți pentru anumite produse

4. Descrierea relațiilor, incluzând precizarea cardinalității acestora

RELAȚIE	CARDINALITATE	OBSERVAȚII
APARȚINE	COMANDA-CLIENT: many-to-one	O comandă aparține unui singur client
FOLOSEȘTE	COMANDA-MOD_PLATA: many-to-one	Fiecare comandă folosește un singur mod de plată
FOLOSEȘTE	COMANDA-TRANSPORT: many-to-one	Fiecare comandă folosește un singur tip de transport
ARE	CLIENT-CONTACTE_CLIENT: one-to-many	Un client poate avea mai multe contacte (telefon,email,etc.)
PRIMEȘTE	CLIENT-BONUSURI: one-to-many	Un client poate primi mai multe bonusuri
SCRIE	CLIENT-RECENZIE: one-to-many	Un client poate lăsa 0 sau mai multe recenzii
PRIMEȘTE	PRODUS-RECENZIE: one-to-many	Un produs primește 0 sau mai multe recenzii
OFERI	PRODUCATOR-RECENZIE: one-to-many	O recenzie se poate oferi fiecărui produs, în funcție de producător
ARE	PRODUS-STOC: one-to-many	Un produs poate avea mai multe stocuri cu preț diferit
ARE	PRODUCATOR-STOC: one-to-many	Stocul produselor poate avea producători diferiți, iar prețul variază
ARE	STOC-REDUCERE: one-to-many	Fiecare stoc poate avea >=0 reduceri în funcție de producător
APARȚINE	COMANDA_PRODUS-COMANDA: many-to-one COMANDA_PRODUS-PRODUS: many-to-one	Tabel asociativ între comenzi și produse, aparțin entității COMANDA_PRODUS

5. Descrierea atributelor, incluzând tipul de date și eventualele constrângeri, valori implicite, valori posibile ale atributelor

COMANDA

Atribut	Tip de date	Constrângeri	Valori posibile și implicite
id_comanda	NUMBER(13)	PK	
id_modplata	NUMBER(13)	FK către MOD_PLATA	id modalitate plată existentă
id_transport	NUMBER(13)	FK către TRANSPORT	id transport existent
id_client	NUMBER(13)	FK către CLIENT	id al unui client
data_comanda	DATE	REFERENCES client(id_client)	24-MAR-2025, implicit DEFAULT SYSDATE
status_comanda	VARCHAR2(20)	CHECK (comandă in proces, livrată sau anulată)	‘in proces’, ‘livrata’, ‘anulata’, prima implicită

CLIENT

Atribut	Tip de date	Constrângeri	Valori posibile și implicite
id_client	NUMBER(13)	PK	
nume	VARCHAR2(50)	NOT NULL	Ion Maria
data_inregistrare	DATE	NOT NULL	24-MAR-2025, implicit DEFAULT SYSDATE
tip_client	VARCHAR2(50)	CHECK (este pers fizică sau juridică)	‘persoana fizica’, ‘persoana juridica’, prima implicită

CONTACTE_CLIENT

Atribut	Tip de date	Constrângeri	Valori posibile și implicite
id_contacteclient	NUMBER(13)	PK	
id_client	NUMBER(13)	FK către CLIENT	id client existent
tip_contact	VARCHAR2(20)	NOT NULL	‘telefon’, ‘email’, ‘adresa’
valoare_contact	VARCHAR2(100)	NOT NULL	‘0799166864’, ‘exemplu@dm.ro’
status_contact	VARCHAR2(15)	CHECK (este activ sau inactiv), DEFAULT ‚activ’	‘activ’, ‘inactiv’

BONUSURI

Atribut	Tip de date	Constrângeri	Valori posibile și implicite
id_bonusuri	NUMBER(13)	PK	
id_client	NUMBER(13)	FK către CLIENT	id client existent
valoare_bonus	NUMBER(10,2)	NOT NULL	10.50, 50
data_expirare	DATE	NULL (doar dacă expiră nu e null)	06-MAR-2025, NULL

MOD_PLATA

Atribut	Tip de date	Constrângeri	Valori posibile și implicite
id_modplata	NUMBER(13)	PK	
tip_plata	VARCHAR2(20)	NOT NULL, CHECK (este cash sau card)	‘cash’, ‘card’

CASH

Atribut	Tip de date	Constrângeri	Valori posibile și implicite
id_modplata	NUMBER(13)	PK	
moneda	VARCHAR2(20)	NOT NULL	‘RON’, ‘EUR’

CARD

Atribut	Tip de date	Constrângeri	Valori posibile și implicite
id_modplata	NUMBER(13)	PK	
numar_card	VARCHAR2(20)	NOT NULL	‘4444555566667777’
data_expirare	DATE	NOT NULL	06-MAR-2025
tip_card	VARCHAR2(20)	NOT NULL	‘Visa’, ‘Mastercard’

TRANSPORT

Atribut	Tip de date	Constrângeri	Valori posibile și implicite
id_transport	NUMBER(13)	PK	
firma	VARCHAR2(100)	NOT NULL	‘DHL’, ‘FedEx’
timp	NUMBER(3)	NOT NULL	2, 3 (zile)
cost	NUMBER(10,2)	NOT NULL	30.50 (lei)

COMANDA_PRODUS (tabel asociativ)

Atribut	Tip de date	Constrângeri	Valori posibile și implicite
id_comanda	NUMBER(13)	PK (FK către COMANDA)	
id_produs	NUMBER(13)	PK (FK către PRODUS)	
cantitate	NUMBER(10)	NOT NULL	500 (produse comandate pentru o comandă)
pret	NUMBER(10,2)	NOT NULL	30.15

PRODUS

Atribut	Tip de date	Constrângeri	Valori posibile și implicite
id_produs	NUMBER(13)	PK	
nume_produs	VARCHAR2(100)	NOT NULL	‘Lapte’
categorie	VARCHAR2(50)		‘birou’, ‘arta'

RECENZIE

Atribut	Tip de date	Constrângeri	Valori posibile și implicite
id_recenzie	NUMBER(13)	PK	
id_client	NUMBER(13)	FK către CLIENT	id al unui client existent
id_produs	NUMBER(13)	FK către PRODUS	id-ul unui produs

id_producator	NUMBER(13)	FK către PRODUCATOR	id-ul unui producator
rating	NUMBER(2)	CHECK (între 1 și 5)	2, 5
data_recenzie	DATE	NOT NULL	06-MAR-2025, implicit DEFAULT SYSDATE

PRODUCATOR

Atribut	Tip de date	Constrângeri	Valori posibile și implicite
id_producator	NUMBER(13)	PK	
nume	VARCHAR2(100)	NOT NULL	‘Hochland’, ‘Kinder’
tara_origine	VARCHAR2(100)	NOT NULL	‘Romania’, ‘Austria’
telefon	VARCHAR2(20)	NOT NULL	‘0799166864’

STOC (Tabel asociativ)

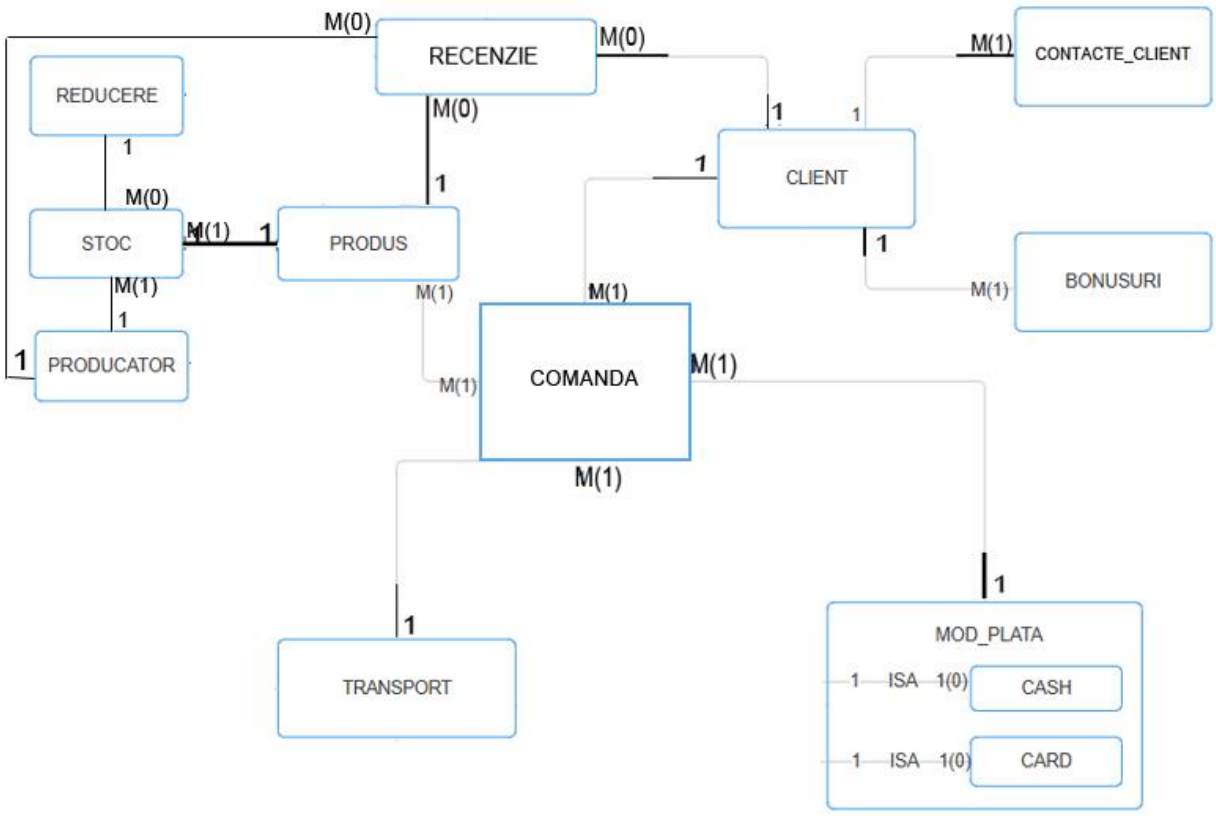
Atribut	Tip de date	Constrângeri	Valori posibile și implicite
id_producator	NUMBER(13)	PK (FK către PRODUCATOR)	
id_produs	NUMBER(13)	PK (FK către PRODUS)	
cantitate_disp	NUMBER(10)	NOT NULL, CHECK (>=0)	10
pret	NUMBER(10,2)	NOT NULL CHECK (pret >= 0)	55.20
minim_critic	NUMBER(10)	NOT NULL, CHECK (>=0)	implicit 3, exemple valori: 5, 10
ultima_actualizare	DATE	NOT NULL	06-MAR-2025, implicit DEFAULT SYSDATE

greutate	NUMBER(5,2)	CHECK (greutate >0)	0.52, 20 (kg)
expira	DATE	NOT NULL	06-MAR-2025

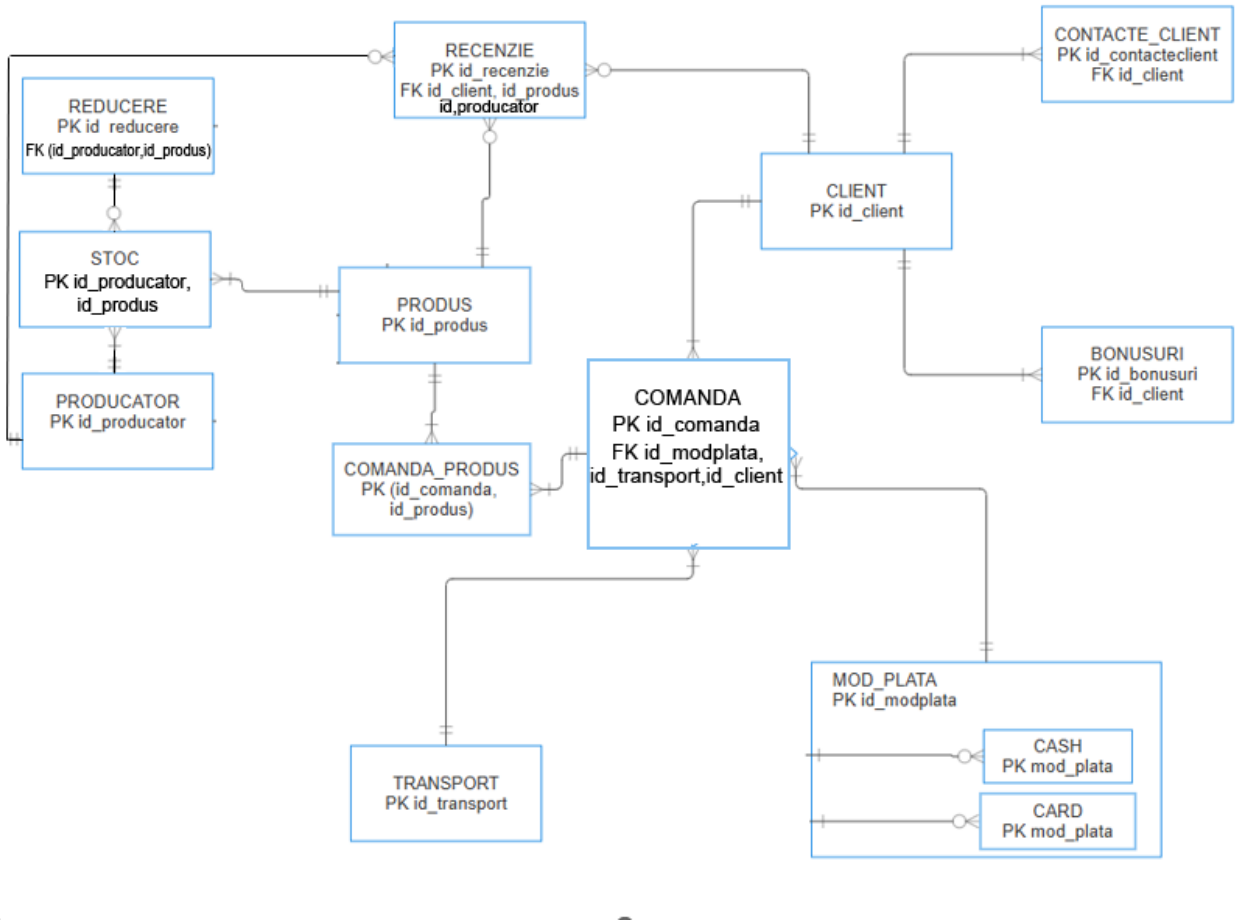
REDUCERE

Atribut	Tip de date	Constrângeri	Valori posibile și implicite
id_reducere	NUMBER(13)	PK	
id_produs	NUMBER(13)	FK către PRODUS	id al unui produs existent
id_producator	NUMBER(13)	FK către PRODUCATOR	id al unui producător existent
procentaj	NUMBER(10)	CHECK (>=0)	implicit NULL, altfel 5, 10
metoda_aplicare	VARCHAR2(100)	CHECK (este aplicată automat sau cu cod prom.)	‘automat’, ‘cod promotional’
transferabila	VARCHAR2(100)	CHECK (se poate transfera la alt client sau nu)	‘da’, ‘nu’

6. Realizarea diagramei entitate-relație corespunzătoare descrierii de la punctele 3-5



7. **Realizarea diagramei conceptuale corespunzătoare diagramei entitate-relație proiectate la punctul 6. Diagrama conceptuală obținută trebuie să conțină minimum 7 tabele (fără considerarea subentităților), dintre care cel puțin un tabel asociativ**



8. Enumerarea schemelor relaționale corespunzătoare diagramei conceptuale proiectate la punctul 7

COMANDA (id_comanda(PK), id_client(FK), id_modplata(FK), id_transport(FK), data_comanda, status_comanda)

CLIENT (id_client(PK), nume, data_inregistrare, tip_client)

CONTACTE_CLIENT (id_contacteclient(PK), id_client(FK), tip_contact, valoare_contact, status_contact)

BONUSURI (id_bonusuri(PK), id_client(FK), valoare_bonus, data_expirare)

MOD_PLATA (id_modplata(PK), tip_plata)

CASH (id_modplata(PK), moneda)

CARD (id_modplata(PK), numar_card, data_expirare, tip_card)

TRANSPORT (id_transport(PK), firma, timp, cost)

COMANDA_PRODUS (id_comanda (PK,FK), id_produs(PK,FK), cantitate, pret)

PRODUS (id_produs(PK), nume_produs, categorie)

RECENZIE (id_recenzie(PK), id_client(FK), id_produs(FK), id_producator(FK), rating, data_recenzie)

STOC (id_producator(PK,FK), id_produs (PK,FK), cantitate_disp, pret, minim_critic, ultima_actualizare, greutate, expira)

PRODUCATOR (id_producator(PK), nume, tara_origine, telefon)

REDUCERE (id_reducere(PK), id_produs(FK), id_producator(FK), procentaj, metoda_aplicare, transferabila)

9. Realizarea normalizării până la forma normală 3 (FN1-FN3)

- Non FN1:

CLIENT			
ID_CLIENT	NUME	TELEFOANE	EMAILURI
1	Matei Andrei	0799251452, 0721456325	matei@gm.com
2	Cristina Martha	0796243615	cristina@yahoo.com, mar@gmail.com

- FN1:

CLIENT	
ID_CLIENT	NUME
1	Matei Andrei
2	Cristina Martha

CONTACTE_CLIENT			
ID_CONTACTECLIENT	ID_CLIENT	TIP_CONTACT	VALOARE_CONTACT
1	1	telefon	0799251452
2	1	telefon	0721456325
3	1	email	matei@gm.com
4	2	telefon	0796243615
5	2	email	cristina@yahoo.com
6	2	email	mar@gmail.com

FN1 presupune faptul că toate atributele sunt atomice (nu liste, nu valori multiple în același câmp). În exemplul de mai sus, descompunerea este realizată prin introducerea datelor de tip contact într-o entitate nouă (contacte_client), diferențiându-le după tipul fiecărui contact. Inițial, un client ar fi putut avea în același câmp mai multe date de contact, ceea ce nu ar fi respectat definiția FN1.

- Non FN2:

COMANDA_PRODUS			
ID_COMANDA	ID_PRODUS	NUME_PRODUS	CANTITATE
1	5	Lapte	5
1	7	Cafea	1

- FN2:

COMANDA_PRODUS		
ID_COMANDA	ID_PRODUS	CANTITATE
1	5	5
1	7	1

PRODUS		
ID_PRODUS	NUME_PRODUS	alte atribute...
5	Lapte	...
7	Cafea	...

Un tabel se află în FN2 dacă se află în FN1 și fiecare atribut care nu participă la cheia primară este dependent de întreaga cheie primară. În schema inițială se observă cum nume_produs depinde doar de id_produs, iar cheile primare sunt id_comanda și id_produs, ceea ce rezultă o dependență parțială, deci non FN2. Pentru a normaliza, am împărțit tabelul în comanda_produs, de unde am șters nume_produs, și în produs, care va conține numele produsului și alte atribute specifice acestuia. Astfel, nume_produs depinde doar de cheia primară simplă id_produs, ceea ce respectă definiția FN2.

- Non FN3:

CLIENT				
ID_CLIENT	ID_CONTACTECLIENT	TIP_CONTACT	VALOARE_CONTACT	TIP_CLIENT
1	1	email	matei@gm.com	pers fizica
2	4	telefon	0799251452	pers juridica

- FN3:

CLIENT	
ID_CLIENT	TIP_CLIENT
1	pers fizica
2	pers juridica

CONTACTE_CLIENT			
ID_CONTACTECLIENT	ID_CLIENT	TIP_CONTACT	VALOARE_CONTACT
1	1	email	matei@gm.com
4	2	telefon	0799251452

Un tabel se află în FN3 dacă se află în FN2 și nu există dependențe tranzitive. În Non FN3, în client se află și informațiile de contact, ceea ce duce la dependențe parțiale și redundanță. În FN3, se află separate client, care păstrează doar datele strict despre client, și contacte_client, care are propriul tabel, legat prin FK id_client.

10. Crearea unei secvențe ce va fi utilizată în inserarea înregistrărilor în tabele (punctul 11)

```
CREATE SEQUENCE seq_client START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;
CREATE SEQUENCE seq_comanda START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;
CREATE SEQUENCE seq_produs START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;
CREATE SEQUENCE seq_transport START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;
CREATE SEQUENCE seq_modplata START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;
CREATE SEQUENCE seq_contacte_client START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;
CREATE SEQUENCE seq_bonusuri START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;
CREATE SEQUENCE seq_recenzie START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;
CREATE SEQUENCE seq_stoc START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;
CREATE SEQUENCE seq_producator START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;
CREATE SEQUENCE seq_reducere START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;
```

Script Output

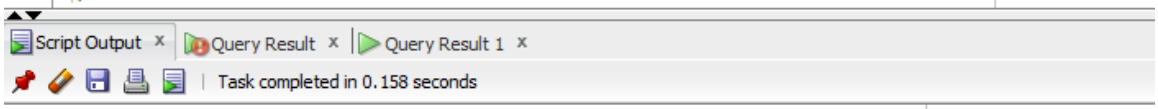
Task completed in 0.099 seconds

```
Sequence SEQ_CLIENT created.

Sequence SEQ_COMANDA created.

Sequence SEQ_PRODUS created.

Sequence SEQ_TRANSPORT created.

Sequence SEQ_MODPLATA created.

Sequence SEQ_CONTACTE_CLIENT created.

Sequence SEQ_BONUSURI created.

Sequence SEQ_RECENZIE created.

Sequence SEQ_STOC created.

Sequence SEQ_PRODUCATOR created.

Sequence SEQ_REDUCERE created.
```

```
CREATE SEQUENCE seq_client START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;

CREATE SEQUENCE seq_comanda START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;

CREATE SEQUENCE seq_produs START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;

CREATE SEQUENCE seq_transport START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;

CREATE SEQUENCE seq_modplata START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;

CREATE SEQUENCE seq_contacte_client START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;

CREATE SEQUENCE seq_bonusuri START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;

CREATE SEQUENCE seq_recenzie START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;

CREATE SEQUENCE seq_stoc START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;

CREATE SEQUENCE seq_producator START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;

CREATE SEQUENCE seq_reducere START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 999999 NOCYCLE NOCACHE;
```

11. Crearea tabelelor în SQL și inserarea de date coerente în fiecare dintre acestea (minimum 5 înregistrări în fiecare tabel neasociativ; minimum 10 înregistrări în tabelele asociative; maxim 30 de înregistrări în fiecare tabel)

```
CREATE TABLE TRANSPORT (
    id_transport NUMBER(13),
    firma VARCHAR2(100) NOT NULL,
    timp NUMBER(3) NOT NULL,
    cost NUMBER(10,2) NOT NULL,
    CONSTRAINT pk_transport PRIMARY KEY (id_transport)
);
CREATE TABLE MOD_PLATA (
    id_modplata NUMBER(13),
    tip_plata VARCHAR2(20) NOT NULL CHECK (tip_plata IN ('cash', 'card')),
    CONSTRAINT pk_mod_plata PRIMARY KEY (id_modplata)
);
CREATE TABLE CASH (
    id_modplata NUMBER(13),
    moneda VARCHAR2(20) NOT NULL,
    CONSTRAINT pk_cash PRIMARY KEY (id_modplata),
    CONSTRAINT fk_cash_mod_plata FOREIGN KEY (id_modplata) REFERENCES MOD_PLATA(id_modplata)
);
CREATE TABLE CARD (
    id_modplata NUMBER(13),
    numar_card VARCHAR2(20) NOT NULL,
    data_expirare DATE NOT NULL,
    tip_card VARCHAR2(20) NOT NULL,
    CONSTRAINT pk_card PRIMARY KEY (id_modplata),
    CONSTRAINT fk_card_mod_plata FOREIGN KEY (id_modplata) REFERENCES MOD_PLATA(id_modplata)
);
CREATE TABLE CLIENT (
    id_client NUMBER(13),
    nume VARCHAR2(50) NOT NULL,
    data_inregistrare DATE DEFAULT SYSDATE NOT NULL,
    tip_client VARCHAR2(50) DEFAULT 'persoana fizica',
    CONSTRAINT chk_tip_client CHECK (tip_client IN ('persoana fizica', 'persoana juridica')),
    CONSTRAINT pk_client PRIMARY KEY (id_client)
);
```




```
Table TRANSPORT created.


Table MOD_PLATA created.


Table CASH created.
```

```
CREATE TABLE CONTACTE_CLIENT (
    id_contacteclient NUMBER(13),
    id_client NUMBER(13),
    tip_contact VARCHAR2(20) NOT NULL CHECK (tip_contact IN ('telefon', 'email', 'adresa')),
    valoare_contact VARCHAR2(100) NOT NULL,
    status_contact VARCHAR2(15) DEFAULT 'activ' CHECK (status_contact IN ('activ', 'inactiv')),
    CONSTRAINT pk_contacte PRIMARY KEY (id_contacteclient),
    CONSTRAINT fk_contact_client FOREIGN KEY (id_client) REFERENCES CLIENT(id_client)
);
CREATE TABLE BONUSURI (
    id_bonusuri NUMBER(13),
    id_client NUMBER(13),
    valoare_bonus NUMBER(10,2) NOT NULL,
    data_expirare DATE NULL,
    CONSTRAINT pk_bonusuri PRIMARY KEY (id_bonusuri),
    CONSTRAINT fk_bonus_client FOREIGN KEY (id_client) REFERENCES CLIENT(id_client)
);
CREATE TABLE COMANDA (
    id_comanda NUMBER(13),
    id_modplata NUMBER(13),
    id_transport NUMBER(13),
    id_client NUMBER(13),
    data_comanda DATE DEFAULT SYSDATE,
    status_comanda VARCHAR2(20) DEFAULT 'in proces' CHECK (status_comanda IN ('in proces', 'livrata', 'anulata')),
    CONSTRAINT pk_comanda PRIMARY KEY (id_comanda),
    CONSTRAINT fk_comanda_modplata FOREIGN KEY (id_modplata) REFERENCES MOD_PLATA(id_modplata),
    CONSTRAINT fk_comanda_transport FOREIGN KEY (id_transport) REFERENCES TRANSPORT(id_transport),
    CONSTRAINT fk_comanda_client FOREIGN KEY (id_client) REFERENCES CLIENT(id_client)
);
CREATE TABLE PRODUS (
    id_produs NUMBER(13),
    nume_produs VARCHAR2(100) NOT NULL,
    categorie VARCHAR2(50),
    CONSTRAINT pk_produs PRIMARY KEY (id_produs)
);
```

Script Output | Query Result | Query Result 1

Task completed in 0.158 seconds

```
Table CONTACTE_CLIENT created.


Table BONUSURI created.


Table COMANDA created.
```

```
CREATE TABLE PRODUCATOR (
    id_producator NUMBER(13),
    nume VARCHAR2(100) NOT NULL,
    tara_origine VARCHAR2(100) NOT NULL,
    telefon VARCHAR2(20) NOT NULL,
    CONSTRAINT pk_producator PRIMARY KEY (id_producator)
);
CREATE TABLE STOC (
    id_produs NUMBER(13),
    id_producator NUMBER(13),
    cantitate_disp NUMBER(10) NOT NULL CHECK (cantitate_disp >= 0),
    pret NUMBER(10,2) NOT NULL CHECK (pret >= 0),
    minim_critic NUMBER(10) DEFAULT 3 NOT NULL CHECK (minim_critic >= 0),
    ultima_actualizare DATE DEFAULT SYSDATE NOT NULL,
    greutate NUMBER(5,2) CHECK (greutate > 0),
    expira DATE NOT NULL,
    CONSTRAINT pk_stoc PRIMARY KEY (id_produs, id_producator),
    CONSTRAINT fk_stoc_produs FOREIGN KEY (id_produs) REFERENCES PRODUS(id_produs),
    CONSTRAINT fk_stoc_producator FOREIGN KEY (id_producator) REFERENCES PRODUCATOR(id_producator)
);
CREATE TABLE REDUCERE (
    id_reducere NUMBER(13),
    id_produs NUMBER(13),
    id_producator NUMBER(13),
    procentaj NUMBER(10),
    metoda_aplicare VARCHAR2(100) CHECK (metoda_aplicare IN ('automat', 'cod promotional')),
    transferabila VARCHAR2(100) CHECK (transferabila IN ('da', 'nu')),
    CONSTRAINT pk_reducere PRIMARY KEY (id_reducere),
    CONSTRAINT fk_reducere_stoc FOREIGN KEY (id_produs, id_producator)
        REFERENCES STOC(id_produs, id_producator),
    CONSTRAINT unq_reducere_stoc UNIQUE (id_produs, id_producator)
);
```

Script Output | Query Result | Query Result 1

Task completed in 0.158 seconds

```
Table PRODUCATOR created.


Table STOC created.


Table REDUCERE created.


Table RECENZIE created.
```

```sql
CREATE TABLE RECENZIE (
    id_recenzie NUMBER(13),
    id_client NUMBER(13),
    id_produs NUMBER(13),
    id_producator NUMBER(13),
    rating NUMBER(2) CHECK (rating >= 1 AND rating <= 5),
    data_recenzie DATE DEFAULT SYSDATE NOT NULL,
    CONSTRAINT pk_recenzie PRIMARY KEY (id_recenzie),
    CONSTRAINT fk_recenzie_client FOREIGN KEY (id_client) REFERENCES CLIENT(id_client),
    CONSTRAINT fk_recenzie_produs FOREIGN KEY (id_produs) REFERENCES PRODUS(id_produs),
    CONSTRAINT fk_recenzie_producator FOREIGN KEY (id_producator) REFERENCES PRODUCATOR(id_producator)
);
CREATE TABLE COMANDA_PRODUS (
    id_comanda NUMBER(13),
    id_produs NUMBER(13),
    cantitate NUMBER(10) NOT NULL,
    pret NUMBER(10,2) NOT NULL,
    CONSTRAINT pk_comanda_produs PRIMARY KEY (id_comanda, id_produs),
    CONSTRAINT fk_comanda_produs_comanda FOREIGN KEY (id_comanda) REFERENCES COMANDA(id_comanda),
    CONSTRAINT fk_comanda_produs_produs FOREIGN KEY (id_produs) REFERENCES PRODUS(id_produs)
);
```

Script Output × | Query Result × | Query Result 1 ×

Task completed in 0.158 seconds

```
Table REDUCERE created.

Table RECENZIE created.

Table COMANDA_PRODUS created.
```

```sql
INSERT INTO TRANSPORT VALUES (seq_transport.nextval, 'Fan Courier', 2, 19.99);
INSERT INTO TRANSPORT VALUES (seq_transport.nextval, 'DPD', 3, 15.50);
INSERT INTO TRANSPORT VALUES (seq_transport.nextval, 'Sameday', 1, 12.00);
INSERT INTO TRANSPORT VALUES (seq_transport.nextval, 'Cargus', 4, 18.30);
INSERT INTO TRANSPORT VALUES (seq_transport.nextval, 'Bookurier', 2, 10.00);
select * from transport;
```

Script Output × | Query Result × | Query Result 1 ×

SQL | All Rows Fetched: 5 in 0.004 seconds

	ID_TRANSPORT	FIRMA	TIMP	COST
1	1	Fan Courier	2	19.99
2	2	DPD	3	15.5
3	3	Sameday	1	12
4	4	Cargus	4	18.3
5	5	Bookurier	2	10

```sql
INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'cash');
INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'card');
INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'cash');
INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'card');
INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'cash');
INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'card');
INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'cash');
INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'card');
INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'cash');
INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'card');
select * from mod_plata;
```

Script Output × | Query Result × | Query Result 1 ×

SQL | All Rows Fetched: 10 in 0.001 seconds

	ID_MODPLATA	TIP_PLATA
1	1	cash
2	2	card
3	3	cash
4	4	card
5	5	cash
6	6	card
7	7	cash
8	8	card
9	9	cash
10	10	card

```sql
INSERT INTO CASH VALUES (1, 'RON');
INSERT INTO CASH VALUES (3, 'EUR');
INSERT INTO CASH VALUES (5, 'USD');
INSERT INTO CASH VALUES (7, 'RON');
INSERT INTO CASH VALUES (9, 'EUR');
select * from cash;
```

Script Output × | Query Result × | Query Resul

SQL | All Rows Fetched: 5 in 0.004 seco

	ID_MODPLATA	MONEDA
1	1	RON
2	3	EUR
3	5	USD
4	7	RON
5	9	EUR

```
INSERT INTO CARD VALUES (2, '1234123412341234', TO_DATE('2026-10-10','YYYY-MM-DD'), 'Visa');
INSERT INTO CARD VALUES (4, '5678567856785678', TO_DATE('2025-12-31','YYYY-MM-DD'), 'MasterCard');
INSERT INTO CARD VALUES (6, '9999888877776666', TO_DATE('2027-01-01','YYYY-MM-DD'), 'Visa');
INSERT INTO CARD VALUES (8, '4444333322221111', TO_DATE('2025-09-30','YYYY-MM-DD'), 'Maestro');
INSERT INTO CARD VALUES (10, '1111222233334444', TO_DATE('2026-03-15','YYYY-MM-DD'), 'Revolut');
select * from card;
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 5 in 0.003 seconds

	ID_MODPLATA	NUMAR_CARD	DATA_EXPIRARE	TIP_CARD
1	2	1234123412341234	10-OCT-26	Visa
2	4	5678567856785678	31-DEC-25	MasterCard
3	6	9999888877776666	01-JAN-27	Visa
4	8	4444333322221111	30-SEP-25	Maestro
5	10	1111222233334444	15-MAR-26	Revolut

```
INSERT INTO CLIENT VALUES (seq_client.nextval, 'Ion Popescu', TO_DATE('2024-01-01','YYYY-MM-DD'), 'persoana fizica');
INSERT INTO CLIENT VALUES (seq_client.nextval, 'SC Tehnoserv SRL', TO_DATE('2024-02-01','YYYY-MM-DD'), 'persoana juridica');
INSERT INTO CLIENT VALUES (seq_client.nextval, 'Maria Ionescu', TO_DATE('2024-03-01','YYYY-MM-DD'), 'persoana fizica');
INSERT INTO CLIENT VALUES (seq_client.nextval, 'Andrei Dumitrescu', TO_DATE('2024-04-01','YYYY-MM-DD'), 'persoana fizica');
INSERT INTO CLIENT VALUES (seq_client.nextval, 'DigitalNet Solutions', TO_DATE('2024-05-01','YYYY-MM-DD'), 'persoana juridica');
select * from client;
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 5 in 0.006 seconds

	ID_CLIENT	NUME	DATA_INREGISTRARE	TIP_CLIENT
1	1	Ion Popescu	01-JAN-24	persoana fizica
2	2	SC Tehnoserv SRL	01-FEB-24	persoana juridica
3	3	Maria Ionescu	01-MAR-24	persoana fizica
4	4	Andrei Dumitrescu	01-APR-24	persoana fizica
5	5	DigitalNet Solutions	01-MAY-24	persoana juridica

```
INSERT INTO PRODUCATOR VALUES (seq_producator.nextval, 'Zentari Corp', 'Romania', '1234567890');
INSERT INTO PRODUCATOR VALUES (seq_producator.nextval, 'Novera Ltd', 'Polonia', '0987654321');
INSERT INTO PRODUCATOR VALUES (seq_producator.nextval, 'Veltria GmbH', 'Germania', '5678901234');
INSERT INTO PRODUCATOR VALUES (seq_producator.nextval, 'Auralis S.p.A.', 'Italia', '1122334455');
INSERT INTO PRODUCATOR VALUES (seq_producator.nextval, 'Solvexa SARL', 'Franta', '5566778899');
select * from producator;
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 5 in 0.004 seconds

	ID_PRODUCATOR	NUME	TARA_ORIGINE	TELEFON
1	1	Zentari Corp	Romania	1234567890
2	2	Novera Ltd	Polonia	0987654321
3	3	Veltria GmbH	Germania	5678901234
4	4	Auralis S.p.A.	Italia	1122334455
5	5	Solvexa SARL	Franta	5566778899

```
INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL, 'pix albastru', 'papetarie');
INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL, 'caiet A5', 'papetarie');
INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL, 'agenda', 'papetarie');
INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL, 'mapa plastic', 'organizare birou');
INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL, 'creion mecanic', 'papetarie');
INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL, 'marker permanent', 'papetarie');
INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL, 'dosar carton', 'organizare birou');
INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL, 'set acuarele', 'arta');
INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL, 'lipici solid', 'papetarie');
INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL, 'taietor hartie', 'organizare birou');
select * from produs;
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 10 in 0.005 seconds

	ID_PRODUS	NUME_PRODUS	CATEGORIE
1	1	pix albastru	papetarie
2	2	caiet A5	papetarie
3	3	agenda	papetarie
4	4	mapa plastic	organizare birou
5	5	creion mecanic	papetarie
6	6	marker permanent	papetarie
7	7	dosar carton	organizare birou
8	8	set acuarele	arta
9	9	lipici solid	papetarie
10	10	taietor hartie	organizare birou

```
INSERT INTO STOC VALUES (1, 1, 50, 4000.00, 3, TO_DATE('2025-05-01', 'YYYY-MM-DD'), 1.5, TO_DATE('2025-08-01', 'YYYY-MM-DD'));
INSERT INTO STOC VALUES (1, 2, 30, 3950.00, 3, TO_DATE('2025-05-03', 'YYYY-MM-DD'), 1.4, TO_DATE('2025-08-05', 'YYYY-MM-DD'));
INSERT INTO STOC VALUES (2, 1, 60, 5000.00, 3, TO_DATE('2025-05-02', 'YYYY-MM-DD'), 2.0, TO_DATE('2025-07-20', 'YYYY-MM-DD'));
INSERT INTO STOC VALUES (2, 3, 45, 5100.00, 3, TO_DATE('2025-04-28', 'YYYY-MM-DD'), 1.8, TO_DATE('2025-07-15', 'YYYY-MM-DD'));
INSERT INTO STOC VALUES (3, 2, 100, 150.00, 3, TO_DATE('2025-05-05', 'YYYY-MM-DD'), 0.5, TO_DATE('2025-06-30', 'YYYY-MM-DD'));
INSERT INTO STOC VALUES (3, 4, 80, 160.00, 3, TO_DATE('2025-04-30', 'YYYY-MM-DD'), 0.6, TO_DATE('2025-07-01', 'YYYY-MM-DD'));
INSERT INTO STOC VALUES (4, 5, 70, 300.00, 3, TO_DATE('2025-05-06', 'YYYY-MM-DD'), 1.0, TO_DATE('2025-09-15', 'YYYY-MM-DD'));
INSERT INTO STOC VALUES (4, 1, 65, 290.00, 3, TO_DATE('2025-05-01', 'YYYY-MM-DD'), 1.2, TO_DATE('2025-09-10', 'YYYY-MM-DD'));
INSERT INTO STOC VALUES (5, 3, 40, 800.00, 3, TO_DATE('2025-04-15', 'YYYY-MM-DD'), 3.0, TO_DATE('2025-10-01', 'YYYY-MM-DD'));
INSERT INTO STOC VALUES (5, 2, 35, 820.00, 3, TO_DATE('2025-05-07', 'YYYY-MM-DD'), 2.8, TO_DATE('2025-09-28', 'YYYY-MM-DD'));
select * from stoc;
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 10 in 0.005 seconds

	ID_PRODUS	ID_PRODUCATOR	CANTITATE_DISP	PRET	MINIM_CRITIC	ULTIMA_ACTUALIZARE	GREUTATE	EXPIRA
1	1	1	50	4000	3	01-MAY-25	1.5	01-AUG-25
2	1	2	30	3950	3	03-MAY-25	1.4	05-AUG-25
3	2	1	60	5000	3	02-MAY-25	2	20-JUL-25
4	2	3	45	5100	3	28-APR-25	1.8	15-JUL-25
5	3	2	100	150	3	05-MAY-25	0.5	30-JUN-25
6	3	4	80	160	3	30-APR-25	0.6	01-JUL-25
7	4	5	70	300	3	06-MAY-25	1	15-SEP-25
8	4	1	65	290	3	01-MAY-25	1.2	10-SEP-25
9	5	3	40	800	3	15-APR-25	3	01-OCT-25
10	5	2	35	820	3	07-MAY-25	2.8	28-SEP-25

```
INSERT INTO CONTACTE_CLIENT VALUES (seq_contacte_client.nextval, 1, 'telefon', '0712345678', 'activ');
INSERT INTO CONTACTE_CLIENT VALUES (seq_contacte_client.nextval, 2, 'email', 'client@company.com', 'activ');
INSERT INTO CONTACTE_CLIENT VALUES (seq_contacte_client.nextval, 3, 'adresa', 'Str. Exemplu 10, Bucuresti', 'activ');
INSERT INTO CONTACTE_CLIENT VALUES (seq_contacte_client.nextval, 4, 'telefon', '0723456789', 'inactiv');
INSERT INTO CONTACTE_CLIENT VALUES (seq_contacte_client.nextval, 5, 'email', 'contact@firma.ro', 'activ');
select * from contacte_client;
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 5 in 0.004 seconds

	ID_CONTACTECLIENT	ID_CLIENT	TIP_CONTACT	VALOARE_CONTACT	STATUS_CONTACT
1	1	1	telefon	0712345678	activ
2	2	2	email	client@company.com	activ
3	3	3	adresa	Str. Exemplu 10, Bucuresti	activ
4	4	4	telefon	0723456789	inactiv
5	5	5	email	contact@firma.ro	activ

```
INSERT INTO BONUSURI VALUES (seq_bonusuri.nextval, 1, 50.00, TO_DATE('2025-12-31', 'YYYY-MM-DD'));
INSERT INTO BONUSURI VALUES (seq_bonusuri.nextval, 2, 25.00, TO_DATE('2025-06-30', 'YYYY-MM-DD'));
INSERT INTO BONUSURI VALUES (seq_bonusuri.nextval, 3, 75.00, NULL);
INSERT INTO BONUSURI VALUES (seq_bonusuri.nextval, 4, 100.00, TO_DATE('2025-05-15', 'YYYY-MM-DD'));
INSERT INTO BONUSURI VALUES (seq_bonusuri.nextval, 5, 30.00, TO_DATE('2025-11-30', 'YYYY-MM-DD'));
select * from bonusuri;
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 5 in 0.005 seconds

	ID_BONUSURI	ID_CLIENT	VALOARE_BONUS	DATA_EXPIRARE
1	1	1	50	31-DEC-25
2	2	2	25	30-JUN-25
3	3	3	75	(null)
4	4	4	100	15-MAY-25
5	5	5	30	30-NOV-25

```
INSERT INTO COMANDA VALUES (seq_comanda.nextval, 1, 1, 1, TO_DATE('2025-05-01', 'YYYY-MM-DD'), 'in proces');
INSERT INTO COMANDA VALUES (seq_comanda.nextval, 2, 2, 2, TO_DATE('2025-05-02', 'YYYY-MM-DD'), 'livrata');
INSERT INTO COMANDA VALUES (seq_comanda.nextval, 3, 3, 3, TO_DATE('2025-05-03', 'YYYY-MM-DD'), 'anulata');
INSERT INTO COMANDA VALUES (seq_comanda.nextval, 4, 4, 4, TO_DATE('2025-05-04', 'YYYY-MM-DD'), 'in proces');
INSERT INTO COMANDA VALUES (seq_comanda.nextval, 5, 5, 5, TO_DATE('2025-05-05', 'YYYY-MM-DD'), 'livrata');
select * from comanda;
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 5 in 0.005 seconds

	ID_COMANDA	ID_MODPLATA	ID_TRANSPORT	ID_CLIENT	DATA_COMANDA	STATUS_COMANDA
1	1	1	1	1	01-MAY-25	in proces
2	2	2	2	2	02-MAY-25	livrata
3	3	3	3	3	03-MAY-25	anulata
4	4	4	4	4	04-MAY-25	in proces
5	5	5	5	5	05-MAY-25	livrata

```
INSERT INTO REDUCERE VALUES (seq_reducere.nextval, 1, 1, 10, 'automat', 'da');
INSERT INTO REDUCERE VALUES (seq_reducere.nextval, 2, 3, 15, 'cod promotional', 'nu');
INSERT INTO REDUCERE VALUES (seq_reducere.nextval, 3, 4, 20, 'automat', 'da');
INSERT INTO REDUCERE VALUES (seq_reducere.nextval, 4, 1, 5, 'cod promotional', 'da');
INSERT INTO REDUCERE VALUES (seq_reducere.nextval, 5, 2, 30, 'automat', 'nu');
INSERT INTO REDUCERE VALUES (seq_reducere.nextval, 2, 1, 15, 'cod promotional', 'nu');
INSERT INTO REDUCERE VALUES (seq_reducere.nextval, 3, 2, 17, 'automat', 'da');
select * from reducere;
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 7 in 0.004 seconds

	ID_REDUCERE	ID_PRODUS	ID_PRODUCATOR	PROCENTAJ	METODA_APLICARE	TRANSFERABILA
1	1	1	1	10	automat	da
2	2	2	3	15	cod promotional	nu
3	3	3	4	20	automat	da
4	4	4	1	5	cod promotional	da
5	5	5	2	30	automat	nu
6	6	2	1	15	cod promotional	nu
7	7	3	2	17	automat	da

```
INSERT INTO RECENZIE VALUES (seq_recenzie.nextval, 1, 1, 1, 5, TO_DATE('2025-05-06', 'YYYY-MM-DD'));
INSERT INTO RECENZIE VALUES (seq_recenzie.nextval, 2, 2, 3, 4, TO_DATE('2025-05-07', 'YYYY-MM-DD'));
INSERT INTO RECENZIE VALUES (seq_recenzie.nextval, 3, 3, 4, 3, TO_DATE('2025-05-08', 'YYYY-MM-DD'));
INSERT INTO RECENZIE VALUES (seq_recenzie.nextval, 4, 5, 2, 2, TO_DATE('2025-05-09', 'YYYY-MM-DD'));
INSERT INTO RECENZIE VALUES (seq_recenzie.nextval, 5, 1, 1, 1, TO_DATE('2025-05-10', 'YYYY-MM-DD'));
INSERT INTO RECENZIE VALUES (seq_recenzie.nextval, 3, 2, 3, 4, TO_DATE('2025-05-11', 'YYYY-MM-DD'));
INSERT INTO RECENZIE VALUES (seq_recenzie.nextval, 4, 4, 5, 5, TO_DATE('2025-05-12', 'YYYY-MM-DD'));
select * from recenzie;
```

Script Output | Query Result

SQL | All Rows Fetched: 7 in 0.004 seconds

	ID_RECENZIE	ID_CLIENT	ID_PRODUS	ID_PRODUCATOR	RATING	DATA_RECENZIE
1	1	1	1	1	5	06-MAY-25
2	2	2	2	3	4	07-MAY-25
3	3	3	3	4	3	08-MAY-25
4	4	4	5	2	2	09-MAY-25
5	5	5	1	1	1	10-MAY-25
6	6	3	2	3	4	11-MAY-25
7	7	4	4	5	5	12-MAY-25

```
INSERT INTO COMANDA_PRODUS VALUES (1, 1, 2, 19.99);
INSERT INTO COMANDA_PRODUS VALUES (2, 2, 1, 15.50);
INSERT INTO COMANDA_PRODUS VALUES (3, 3, 1, 12.00);
INSERT INTO COMANDA_PRODUS VALUES (4, 4, 3, 18.30);
INSERT INTO COMANDA_PRODUS VALUES (5, 5, 2, 10.00);
INSERT INTO COMANDA_PRODUS VALUES (1, 4, 1, 19.99);
INSERT INTO COMANDA_PRODUS VALUES (2, 1, 2, 15.50);
INSERT INTO COMANDA_PRODUS VALUES (3, 2, 3, 12.00);
INSERT INTO COMANDA_PRODUS VALUES (4, 5, 1, 18.30);
INSERT INTO COMANDA_PRODUS VALUES (5, 3, 2, 10.00);
select * from comanda_produs;
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 10 in 0.004 seconds

	ID_COMANDA	ID_PRODUS	CANTITATE	PRET
1	1	1	2	19.99
2	2	2	1	15.5
3	3	3	1	12
4	4	4	3	18.3
5	5	5	2	10
6	1	4	1	19.99
7	2	1	2	15.5
8	3	2	3	12
9	4	5	1	18.3
10	5	3	2	10

```
CREATE TABLE TRANSPORT (
    id_transport NUMBER(13),
    firma VARCHAR2(100) NOT NULL,
    timp NUMBER(3) NOT NULL,
    cost NUMBER(10,2) NOT NULL,
    CONSTRAINT pk_transport PRIMARY KEY (id_transport)
);

CREATE TABLE MOD_PLATA (
    id_modplata NUMBER(13),
    tip_plata VARCHAR2(20) NOT NULL CHECK (tip_plata IN ('cash', 'card')),
    CONSTRAINT pk_mod_plata PRIMARY KEY (id_modplata)
);

CREATE TABLE CASH (
    id_modplata NUMBER(13),
    moneda VARCHAR2(20) NOT NULL,
    CONSTRAINT pk_cash PRIMARY KEY (id_modplata),
    CONSTRAINT fk_cash_mod_plata FOREIGN KEY (id_modplata) REFERENCES MOD_PLATA(id_modplata)
);

CREATE TABLE CARD (
    id_modplata NUMBER(13),
    numar_card VARCHAR2(20) NOT NULL,
    data_expirare DATE NOT NULL,
    tip_card VARCHAR2(20) NOT NULL,
    CONSTRAINT pk_card PRIMARY KEY (id_modplata),
    CONSTRAINT fk_card_mod_plata FOREIGN KEY (id_modplata) REFERENCES MOD_PLATA(id_modplata)
);

CREATE TABLE CLIENT (
    id_client NUMBER(13),
    nume VARCHAR2(50) NOT NULL,
    data_inregistrare DATE DEFAULT SYSDATE NOT NULL,
    tip_client VARCHAR2(50) DEFAULT 'persoana fizica',
    CONSTRAINT chk_tip_client CHECK (tip_client IN ('persoana fizica', 'persoana juridica')),
    CONSTRAINT pk_client PRIMARY KEY (id_client)
);

CREATE TABLE CONTACTE_CLIENT (
    id_contacteclient NUMBER(13),
    id_client NUMBER(13),
    tip_contact VARCHAR2(20) NOT NULL CHECK (tip_contact IN ('telefon', 'email', 'adresa')),
    valoare_contact VARCHAR2(100) NOT NULL,
    status_contact VARCHAR2(15) DEFAULT 'activ' CHECK (status_contact IN ('activ', 'inactiv')),
    CONSTRAINT pk_contacte PRIMARY KEY (id_contacteclient),
    CONSTRAINT fk_contact_client FOREIGN KEY (id_client) REFERENCES CLIENT(id_client)
);
```

```
CREATE TABLE BONUSURI (
   id_bonusuri NUMBER(13),
   id_client NUMBER(13),
   valoare_bonus NUMBER(10,2) NOT NULL,
   data_expirare DATE NULL,
   CONSTRAINT pk_bonusuri PRIMARY KEY
(id_bonusuri),
   CONSTRAINT fk_bonus_client FOREIGN KEY
(id_client) REFERENCES CLIENT(id_client)
);



CREATE TABLE COMANDA (
   id_comanda NUMBER(13),
   id_modplata NUMBER(13),
   id_transport NUMBER(13),
   id_client NUMBER(13),
   data_comanda DATE DEFAULT SYSDATE,
   status_comanda VARCHAR2(20) DEFAULT 'in
proces' CHECK (status_comanda IN ('in proces',
'livrata', 'anulata')),
   CONSTRAINT pk_comanda PRIMARY KEY
(id_comanda),
   CONSTRAINT fk_comanda_modplata FOREIGN KEY
(id_modplata) REFERENCES
MOD_PLATA(id_modplata),
   CONSTRAINT fk_comanda_transport FOREIGN KEY
(id_transport) REFERENCES
TRANSPORT(id_transport),
   CONSTRAINT fk_comanda_client FOREIGN KEY
(id_client) REFERENCES CLIENT(id_client)
);
```

```
CREATE TABLE PRODUS (
   id_produs NUMBER(13),
   nume_produs VARCHAR2(100) NOT NULL,
   categorie VARCHAR2(50),
   CONSTRAINT pk_produs PRIMARY KEY (id_produs)
);


CREATE TABLE PRODUCATOR (
   id_producator NUMBER(13),
   nume VARCHAR2(100) NOT NULL,
   tara_origine VARCHAR2(100) NOT NULL,
   telefon VARCHAR2(20) NOT NULL,
   CONSTRAINT pk_producator PRIMARY KEY
(id_producator)
);


CREATE TABLE STOC (
   id_produs NUMBER(13),
   id_producator NUMBER(13),
   cantitate_disp NUMBER(10) NOT NULL CHECK
(cantitate_disp >= 0),
   pret NUMBER(10,2) NOT NULL CHECK (pret >= 0),
   minim_critic NUMBER(10) DEFAULT 3 NOT NULL
CHECK (minim_critic >= 0),
   ultima_actualizare DATE DEFAULT SYSDATE NOT
NULL,
   greutate NUMBER(5,2) CHECK (greutate > 0),
   expira DATE NOT NULL,
   CONSTRAINT pk_stoc PRIMARY KEY (id_produs,
id_producator),
```

```
    CONSTRAINT fk_stoc_produs FOREIGN KEY
(id_produs) REFERENCES PRODUS(id_produs),

    CONSTRAINT fk_stoc_producator FOREIGN KEY
(id_producator) REFERENCES
PRODUCATOR(id_producator)

);


CREATE TABLE REDUCERE (

    id_reducere NUMBER(13),

    id_produs NUMBER(13),

    id_producator NUMBER(13),

    procentaj NUMBER(10),

    metoda_aplicare VARCHAR2(100) CHECK
(metoda_aplicare IN ('automat', 'cod promotional')),

    transferabila VARCHAR2(100) CHECK (transferabila
IN ('da', 'nu')),

    CONSTRAINT pk_reducere PRIMARY KEY
(id_reducere),

    CONSTRAINT fk_reducere_stoc FOREIGN KEY
(id_produs, id_producator)

        REFERENCES STOC(id_produs, id_producator),

    CONSTRAINT unq_reducere_stoc UNIQUE
(id_produs, id_producator)

);


CREATE TABLE RECENZIE (

    id_recenzie NUMBER(13),

    id_client NUMBER(13),

    id_produs NUMBER(13),

    id_producator NUMBER(13),

    rating NUMBER(2) CHECK (rating >= 1 AND rating
<= 5),

    data_recenzie DATE DEFAULT SYSDATE NOT NULL,

    CONSTRAINT pk_recenzie PRIMARY KEY
(id_recenzie),

    CONSTRAINT fk_recenzie_client FOREIGN KEY
(id_client) REFERENCES CLIENT(id_client),

    CONSTRAINT fk_recenzie_produs FOREIGN KEY
(id_produs) REFERENCES PRODUS(id_produs),

    CONSTRAINT fk_recenzie_producator FOREIGN
KEY (id_producator) REFERENCES
PRODUCATOR(id_producator)

);



CREATE TABLE COMANDA_PRODUS (

    id_comanda NUMBER(13),

    id_produs NUMBER(13),

    cantitate NUMBER(10) NOT NULL,

    pret NUMBER(10,2) NOT NULL,

    CONSTRAINT pk_comanda_produs PRIMARY KEY
(id_comanda, id_produs),

    CONSTRAINT fk_comanda_produs_comanda
FOREIGN KEY (id_comanda) REFERENCES
COMANDA(id_comanda),

    CONSTRAINT fk_comanda_produs_produs
FOREIGN KEY (id_produs) REFERENCES
PRODUS(id_produs)

);
```

```sql
INSERT INTO TRANSPORT VALUES (seq_transport.nextval, 'Fan Courier', 2, 19.99);
INSERT INTO TRANSPORT VALUES (seq_transport.nextval, 'DPD', 3, 15.50);
INSERT INTO TRANSPORT VALUES (seq_transport.nextval, 'Sameday', 1, 12.00);
INSERT INTO TRANSPORT VALUES (seq_transport.nextval, 'Cargus', 4, 18.30);
INSERT INTO TRANSPORT VALUES (seq_transport.nextval, 'Bookurier', 2, 10.00);

INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'cash');
INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'card');
INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'cash');
INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'card');
INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'cash');
INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'card');
INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'cash');
INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'card');
INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'cash');
INSERT INTO MOD_PLATA VALUES (seq_modplata.nextval, 'card');

INSERT INTO CASH VALUES (1, 'RON');
INSERT INTO CASH VALUES (3, 'EUR');
INSERT INTO CASH VALUES (5, 'USD');
INSERT INTO CASH VALUES (7, 'RON');
INSERT INTO CASH VALUES (9, 'EUR');

INSERT INTO CARD VALUES (2, '1234123412341234', TO_DATE('2026-10-10','YYYY-MM-DD'), 'Visa');
INSERT INTO CARD VALUES (4, '5678567856785678', TO_DATE('2025-12-31','YYYY-MM-DD'), 'MasterCard');
INSERT INTO CARD VALUES (6, '9999888877776666', TO_DATE('2027-01-01','YYYY-MM-DD'), 'Visa');
INSERT INTO CARD VALUES (8, '4444333322221111', TO_DATE('2025-09-30','YYYY-MM-DD'), 'Maestro');
INSERT INTO CARD VALUES (10, '1111222233334444', TO_DATE('2026-03-15','YYYY-MM-DD'), 'Revolut');

INSERT INTO CLIENT VALUES (seq_client.nextval, 'Ion Popescu', TO_DATE('2024-01-01','YYYY-MM-DD'), 'persoana fizica');
INSERT INTO CLIENT VALUES (seq_client.nextval, 'SC Tehnoserv SRL', TO_DATE('2024-02-01','YYYY-MM-DD'), 'persoana juridica');
INSERT INTO CLIENT VALUES (seq_client.nextval, 'Maria Ionescu', TO_DATE('2024-03-01','YYYY-MM-DD'), 'persoana fizica');
INSERT INTO CLIENT VALUES (seq_client.nextval, 'Andrei Dumitrescu', TO_DATE('2024-04-01','YYYY-MM-DD'), 'persoana fizica');
INSERT INTO CLIENT VALUES (seq_client.nextval, 'DigitalNet Solutions', TO_DATE('2024-05-01','YYYY-MM-DD'), 'persoana juridica');
```

```
INSERT INTO PRODUCATOR VALUES
(seq_producator.nextval, 'Zentari Corp', 'Romania',
'1234567890');

INSERT INTO PRODUCATOR VALUES
(seq_producator.nextval, 'Novera Ltd', 'Polonia',
'0987654321');

INSERT INTO PRODUCATOR VALUES
(seq_producator.nextval, 'Veltria GmbH', 'Germania',
'5678901234');

INSERT INTO PRODUCATOR VALUES
(seq_producator.nextval, 'Auralis S.p.A.', 'Italia',
'1122334455');

INSERT INTO PRODUCATOR VALUES
(seq_producator.nextval, 'Solvexa SARL', 'Franta',
'5566778899');


INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL,
'pix albastru', 'papetarie');

INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL,
'caiet A5', 'papetarie');

INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL,
'agenda', 'papetarie');

INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL,
'mapa plastic', 'organizare birou');

INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL,
'creion mecanic', 'papetarie');

INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL,
'marker permanent', 'papetarie');

INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL,
'dosar carton', 'organizare birou');

INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL,
'set acuarele', 'arta');

INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL,
'lipici solid', 'papetarie');

INSERT INTO PRODUS VALUES (seq_produs.NEXTVAL,
'taietor hartie', 'organizare birou');

INSERT INTO STOC VALUES (1, 1, 50, 4000.00, 3,
TO_DATE('2025-05-01', 'YYYY-MM-DD'), 1.5,
TO_DATE('2025-08-01', 'YYYY-MM-DD'));

INSERT INTO STOC VALUES (1, 2, 30, 3950.00, 3,
TO_DATE('2025-05-03', 'YYYY-MM-DD'), 1.4,
TO_DATE('2025-08-05', 'YYYY-MM-DD'));

INSERT INTO STOC VALUES (2, 1, 60, 5000.00, 3,
TO_DATE('2025-05-02', 'YYYY-MM-DD'), 2.0,
TO_DATE('2025-07-20', 'YYYY-MM-DD'));

INSERT INTO STOC VALUES (2, 3, 45, 5100.00, 3,
TO_DATE('2025-04-28', 'YYYY-MM-DD'), 1.8,
TO_DATE('2025-07-15', 'YYYY-MM-DD'));

INSERT INTO STOC VALUES (3, 2, 100, 150.00, 3,
TO_DATE('2025-05-05', 'YYYY-MM-DD'), 0.5,
TO_DATE('2025-06-30', 'YYYY-MM-DD'));

INSERT INTO STOC VALUES (3, 4, 80, 160.00, 3,
TO_DATE('2025-04-30', 'YYYY-MM-DD'), 0.6,
TO_DATE('2025-07-01', 'YYYY-MM-DD'));

INSERT INTO STOC VALUES (4, 5, 70, 300.00, 3,
TO_DATE('2025-05-06', 'YYYY-MM-DD'), 1.0,
TO_DATE('2025-09-15', 'YYYY-MM-DD'));

INSERT INTO STOC VALUES (4, 1, 65, 290.00, 3,
TO_DATE('2025-05-01', 'YYYY-MM-DD'), 1.2,
TO_DATE('2025-09-10', 'YYYY-MM-DD'));

INSERT INTO STOC VALUES (5, 3, 40, 800.00, 3,
TO_DATE('2025-04-15', 'YYYY-MM-DD'), 3.0,
TO_DATE('2025-10-01', 'YYYY-MM-DD'));

INSERT INTO STOC VALUES (5, 2, 35, 820.00, 3,
TO_DATE('2025-05-07', 'YYYY-MM-DD'), 2.8,
TO_DATE('2025-09-28', 'YYYY-MM-DD'));


INSERT INTO CONTACTE_CLIENT VALUES
(seq_contacte_client.nextval, 1, 'telefon',
'0712345678', 'activ');

INSERT INTO CONTACTE_CLIENT VALUES
(seq_contacte_client.nextval, 2, 'email',
'client@company.com', 'activ');
```

```
INSERT INTO CONTACTE_CLIENT VALUES
(seq_contacte_client.nextval, 3, 'adresa', 'Str.
Exemplu 10, Bucuresti', 'activ');

INSERT INTO CONTACTE_CLIENT VALUES
(seq_contacte_client.nextval, 4, 'telefon',
'0723456789', 'inactiv');

INSERT INTO CONTACTE_CLIENT VALUES
(seq_contacte_client.nextval, 5, 'email',
'contact@firma.ro', 'activ');


INSERT INTO BONUSURI VALUES
(seq_bonusuri.nextval, 1, 50.00, TO_DATE('2025-12-
31', 'YYYY-MM-DD'));

INSERT INTO BONUSURI VALUES
(seq_bonusuri.nextval, 2, 25.00, TO_DATE('2025-06-
30', 'YYYY-MM-DD'));

INSERT INTO BONUSURI VALUES
(seq_bonusuri.nextval, 3, 75.00, NULL);

INSERT INTO BONUSURI VALUES
(seq_bonusuri.nextval, 4, 100.00, TO_DATE('2025-05-
15', 'YYYY-MM-DD'));

INSERT INTO BONUSURI VALUES
(seq_bonusuri.nextval, 5, 30.00, TO_DATE('2025-11-
30', 'YYYY-MM-DD'));


INSERT INTO COMANDA VALUES
(seq_comanda.nextval, 1, 1, 1, TO_DATE('2025-05-
01', 'YYYY-MM-DD'), 'in proces');

INSERT INTO COMANDA VALUES
(seq_comanda.nextval, 2, 2, 2, TO_DATE('2025-05-
02', 'YYYY-MM-DD'), 'livrata');

INSERT INTO COMANDA VALUES
(seq_comanda.nextval, 3, 3, 3, TO_DATE('2025-05-
03', 'YYYY-MM-DD'), 'anulata');

INSERT INTO COMANDA VALUES
(seq_comanda.nextval, 4, 4, 4, TO_DATE('2025-05-
04', 'YYYY-MM-DD'), 'in proces');

INSERT INTO COMANDA VALUES
(seq_comanda.nextval, 5, 5, 5, TO_DATE('2025-05-
05', 'YYYY-MM-DD'), 'livrata');


INSERT INTO REDUCERE VALUES
(seq_reducere.nextval, 1, 1, 10, 'automat', 'da');

INSERT INTO REDUCERE VALUES
(seq_reducere.nextval, 2, 3, 15, 'cod promotional',
'nu');

INSERT INTO REDUCERE VALUES
(seq_reducere.nextval, 3, 4, 20, 'automat', 'da');

INSERT INTO REDUCERE VALUES
(seq_reducere.nextval, 4, 1, 5, 'cod promotional',
'da');

INSERT INTO REDUCERE VALUES
(seq_reducere.nextval, 5, 2, 30, 'automat', 'nu');

INSERT INTO REDUCERE VALUES
(seq_reducere.nextval, 2, 1, 15, 'cod promotional',
'nu');

INSERT INTO REDUCERE VALUES
(seq_reducere.nextval, 3, 2, 17, 'automat', 'da');


INSERT INTO RECENZIE VALUES
(seq_recenzie.nextval, 1, 1, 1, 5, TO_DATE('2025-05-
06', 'YYYY-MM-DD'));

INSERT INTO RECENZIE VALUES
(seq_recenzie.nextval, 2, 2, 3, 4, TO_DATE('2025-05-
07', 'YYYY-MM-DD'));

INSERT INTO RECENZIE VALUES
(seq_recenzie.nextval, 3, 3, 4, 3, TO_DATE('2025-05-
08', 'YYYY-MM-DD'));

INSERT INTO RECENZIE VALUES
(seq_recenzie.nextval, 4, 5, 2, 2, TO_DATE('2025-05-
09', 'YYYY-MM-DD'));

INSERT INTO RECENZIE VALUES
(seq_recenzie.nextval, 5, 1, 1, 1, TO_DATE('2025-05-
10', 'YYYY-MM-DD'));
```

```
INSERT INTO RECENZIE VALUES
(seq_recenzie.nextval, 3, 2, 3, 4, TO_DATE('2025-05-
11', 'YYYY-MM-DD'));

INSERT INTO RECENZIE VALUES
(seq_recenzie.nextval, 4, 4, 5, 5, TO_DATE('2025-05-
12', 'YYYY-MM-DD'));


INSERT INTO COMANDA_PRODUS VALUES (1, 1, 2,
19.99);

INSERT INTO COMANDA_PRODUS VALUES (2, 2, 1,
15.50);

INSERT INTO COMANDA_PRODUS VALUES (3, 3, 1,
12.00);

INSERT INTO COMANDA_PRODUS VALUES (4, 4, 3,
18.30);

INSERT INTO COMANDA_PRODUS VALUES (5, 5, 2,
10.00);

INSERT INTO COMANDA_PRODUS VALUES (1, 4, 1,
19.99);

INSERT INTO COMANDA_PRODUS VALUES (2, 1, 2,
15.50);

INSERT INTO COMANDA_PRODUS VALUES (3, 2, 3,
12.00);

INSERT INTO COMANDA_PRODUS VALUES (4, 5, 1,
18.30);

INSERT INTO COMANDA_PRODUS VALUES (5, 3, 2,
10.00);
```

12. Formulați în limbaj natural și implementați 5 cereri SQL complexe ce vor utiliza, în ansamblul lor, următoarele elemente: a) subcereri sincronizate în care intervin cel puțin 3 tabele b) subcereri nesincronizate în clauza FROM c) grupări de date, funcții grup, filtrare la nivel de grupuri cu subcereri nesincronizate (în clauza de HAVING) d) ordonări și utilizarea funcțiilor NVL și DECODE (în cadrul aceleiași cereri) e) utilizarea a cel puțin 2 funcții pe șiruri de caractere, 2 funcții pe date calendaristice, a cel puțin unei expresii CASE f) utilizarea a cel puțin 1 bloc de cerere (clauza WITH)

- Afișează, pentru fiecare comandă livrată, ID-ul comenzii, numele clientului cu litere mici, tipul plății scris cu litere mari și numele firmei de transport, alături de lungimea numelui firmei. (a) Subcerere sincronizată care implică >=3 tabele (COMANDA, CLIENT, MOD_PLATA, TRANSPORT) + e) funcții pe șiruri: LOWER, UPPER, LENGTH)

```
--Afiseaza, pentru fiecare comanda livrata, ID-ul comenzii, numele clientului cu litere mici,
--tipul platii scris cu litere mari si numele firmei de transport, alaturi de lungimea numelui firmei.
-- a) subcerere sincronizata care implica >=3 tabele (COMANDA, CLIENT, MOD_PLATA, TRANSPORT)
-- e) functii pe siruri: LOWER, UPPER, LENGTH
SELECT
    c_filtrate.id_comanda,
    LOWER(c_filtrate.nume_client) AS nume_client,
    UPPER(c_filtrate.tip_plata) AS tip_plata,
    c_filtrate.firma_transport,
    LENGTH(c_filtrate.firma_transport) AS lungime_nume_transport
FROM (
    SELECT co.id_comanda, cl.nume AS nume_client, mp.tip_plata, t.firma AS firma_transport
    FROM COMANDA co
    JOIN CLIENT cl ON co.id_client = cl.id_client
    JOIN MOD_PLATA mp ON co.id_modplata = mp.id_modplata
    JOIN TRANSPORT t ON co.id_transport = t.id_transport
    WHERE co.status_comanda = 'livrata'
) c_filtrate;
```

Query Result

SQL | All Rows Fetched: 2 in 0.023 seconds

	ID_COMANDA	NUME_CLIENT	TIP_PLATA	FIRMA_TRANSPORT	LUNGIME_NUME_TRANSPORT
1	2	sc tehnoserv srl	CARD	DPD	3
2	5	digitalnet solutions	CASH	Bookurier	9

- Afișează numele clienților care s-au înregistrat cu mai mult de 12 luni în urmă, data de înregistrare în formatul DD-MON-YYYY și numărul de luni care au trecut de la acel moment. (b) subcereri nesincronizate în clauza FROM + e) funcții pe date calendaristice: MONTHS_BETWEEN, TO_CHAR)

```
--Afiseaza numele clientilor care s-au inregistrat cu mai mult de 12 luni in urma,
--data de inregistrare in formatul DD-MON-YYYY si numarul de luni care au trecut
--de la acel moment.
--b) subcereri nesincronizate în clauza FROM
--e) functii pe date calendaristice: MONTHS_BETWEEN, TO_CHAR
SELECT c.nume,
       TO_CHAR(c.data_inregistrare, 'DD-MON-YYYY') AS data_inregistrare,
       TRUNC(MONTHS_BETWEEN(SYSDATE, c.data_inregistrare)) AS luni_de_la_inregistrare
FROM (SELECT id_client, nume, data_inregistrare FROM CLIENT) c
JOIN COMANDA co ON c.id_client = co.id_client
WHERE MONTHS_BETWEEN(SYSDATE, c.data_inregistrare) > 12;
```

Query Result

SQL | All Rows Fetched: 5 in 0.005 seconds

	NUME	DATA_INREGISTRARE	LUNI_DE_LA_INREGISTRARE
1	Ion Popescu	01-JAN-2024	16
2	SC Tehnoserv SRL	01-FEB-2024	15
3	Maria Ionescu	01-MAR-2024	14
4	Andrei Dumitrescu	01-APR-2024	13
5	DigitalNet Solutions	01-MAY-2024	12

- Afișează ID-ul producătorilor, numele producătorilor și valoarea totală a stocului fiecăror producători care au cel puțin un produs în stoc, dar doar dacă valoarea totală a produselor lor aflate în stoc este mai mare decât media valorii totale a stocurilor tuturor producătorilor. (c) grupări date (id_producator, nume), funcții grup (SUM, AVG), subcerere nesincronizată HAVING)

```
--Afiseaza producatorii care au cel putin un produs in stoc, dar doar daca valoarea
--totala a produselor lor aflate in stoc este mai mare decat media valorii totale a
--stocurilor tuturor producatorilor. (afiseaza ID producator, nume producator, valoare totala stoc)
-- c) grupari date (id_producator, nume), functii grup (SUM,AVG), subcerere nesincronizata (HAVING)
SELECT
    p.id_producator,
    p.nume,
    SUM(s.cantitate_disp * s.pret) AS valoare_totala_stoc
FROM PRODUCATOR p
JOIN STOC s ON p.id_producator = s.id_producator
GROUP BY p.id_producator, p.nume
HAVING SUM(s.cantitate_disp * s.pret) > (
    SELECT AVG(valoare_stoc)
    FROM (
        SELECT SUM(cantitate_disp * pret) AS valoare_stoc
        FROM STOC
        GROUP BY id_producator
    )
);
```

Query Result

SQL | All Rows Fetched: 2 in 0.009 seconds

	ID_PRODUCATOR	NUME	VALOARE_TOTALA_STOC
1	1	Zentari Corp	518850
2	3	Veltria GmbH	261500

- Afișează lista de produse, reducerile lor și clasificarea acestora în funcție de mărimea reducerii (‘Fara reducere’ pentru reducere 0, ‘Reducere mica’ pentru reducere între 1 și 10 inclusiv, ‘Reducere mare’ pentru reducere mai mare decât 10) (f) WITH + e) CASE)

```
--Afiseaza lista de produse cu reducerile lor si clasificarea acestora
--in functie de marimea reducerii (=0 'Fara reducere', intre 1 si 10
--'Reducere mica', >10 'Reducere mare')
-- f) WITH
-- e) CASE
WITH reduceri_produs AS (
    SELECT
        p.id_produs,
        p.nume_produs,
        r.procentaj
    FROM
        PRODUS p
    LEFT JOIN
        REDUCERE r
    ON
        p.id_produs = r.id_produs
)
SELECT
    rp.nume_produs,
    NVL(rp.procentaj, 0) AS reducere_produs,
    CASE
        WHEN NVL(rp.procentaj, 0) = 0 THEN 'Fara reducere'
        WHEN NVL(rp.procentaj, 0) BETWEEN 1 AND 10 THEN 'Reducere mica'
        WHEN NVL(rp.procentaj, 0) > 10 THEN 'Reducere mare'
    END AS categorie_reducere
FROM
    reduceri_produs rp
ORDER BY
    rp.nume_produs;
```

Query Result

SQL | All Rows Fetched: 12 in 0.003 seconds

	NUME_PRODUS	REDUCERE_PRODUS	CATEGORIE_REDUCERE
1	agenda	20	Reducere mare
2	agenda	17	Reducere mare
3	caiet A5	15	Reducere mare
4	caiet A5	15	Reducere mare
5	creion mecanic	30	Reducere mare
6	dosar carton	0	Fara reducere
7	lipici solid	0	Fara reducere
8	mapa plastic	5	Reducere mica
9	marker permanent	0	Fara reducere
10	pix albastru	10	Reducere mica
11	set acuarele	0	Fara reducere
12	taietor hartie	0	Fara reducere

- Afișează numele tuturor produselor, numele producătorului asociat fiecărui produs, reducerea produsului (dacă nu exista sau este NULL, afișează 0) și starea stocului fiecărui produs (‚Epuizat' dacă stocul este 0, ‚Mic' dacă este 1, altfel ‚Disponibil'). Rezultatele vor fi ordonate alfabetic după numele producătorului, apoi al produsului. (d) DECODE, NVL (în aceeași cerere), ORDER BY)

```
--Afiseaza numele tuturor produselor, numele producatorului asociat fiecarui produs,
--reducerea produsului (daca nu exista sau este NULL, afiseaza 0) si starea stocului
--fiecarui produs ('Epuizat' daca este stoc 0, 'Mic' daca este 1, altfel 'Disponibil').
--Rezultatul ordoneaza-l alfabetic, dupa numele producatorului, apoi al produsului.
-- d) DECODE, NVL (in aceeasi cerere), ORDER BY
SELECT
    p.nume_produs,
    pr.nume AS nume_producator,
    NVL(r.procentaj, 0) AS reducere_produs,
    DECODE(s.cantitate_disp,
           0, 'Stoc epuizat',
           1, 'Stoc mic',
           'Disponibil') AS stare_stoc
FROM PRODUS p
JOIN STOC s ON p.id_produs = s.id_produs
JOIN PRODUCATOR pr ON s.id_producator = pr.id_producator
LEFT JOIN REDUCERE r ON s.id_produs = r.id_produs AND s.id_producator = r.id_producator
ORDER BY pr.nume, p.nume_produs;
```

Query Result

SQL | All Rows Fetched: 10 in 0.009 seconds

	NUME_PRODUS	NUME_PRODUCATOR	REDUCERE_PRODUS	STARE_STOC
1	agenda	Auralis S.p.A.	20	Disponibil
2	agenda	Novera Ltd	17	Disponibil
3	creion mecanic	Novera Ltd	30	Disponibil
4	pix albastru	Novera Ltd	0	Disponibil
5	mapa plastic	Solvexa SARL	0	Disponibil
6	caiet A5	Veltria GmbH	15	Disponibil
7	creion mecanic	Veltria GmbH	0	Disponibil
8	caiet A5	Zentari Corp	15	Disponibil
9	mapa plastic	Zentari Corp	5	Disponibil
10	pix albastru	Zentari Corp	10	Disponibil

--Afiseaza, pentru fiecare comanda livrata, ID-ul comenzii, numele clientului cu litere mici,

--tipul platii scris cu litere mari si numele firmei de transport, alaturi de lungimea numelui firmei.

-- a) subcerere sincronizata care implica >=3 tabele (COMANDA, CLIENT, MOD_PLATA, TRANSPORT)

-- e) functii pe siruri: LOWER, UPPER, LENGTH

SELECT

c_filtrate.id_comanda,

LOWER(c_filtrate.nume_client) AS nume_client,

UPPER(c_filtrate.tip_plata) AS tip_plata,

c_filtrate.firma_transport,

LENGTH(c_filtrate.firma_transport) AS lungime_nume_transport

FROM (

SELECT co.id_comanda, cl.nume AS nume_client, mp.tip_plata, t.firma AS firma_transport

FROM COMANDA co

JOIN CLIENT cl ON co.id_client = cl.id_client

JOIN MOD_PLATA mp ON co.id_modplata = mp.id_modplata

JOIN TRANSPORT t ON co.id_transport = t.id_transport

WHERE co.status_comanda = 'livrata'

) c_filtrate;

```
--Afiseaza numele clientilor care s-au inregistrat cu mai mult de 12 luni in urma,
--data de inregistrare in formatul DD-MON-YYYY si numarul de luni care au trecut
--de la acel moment.
--b) subcereri nesincronizate în clauza FROM
--e) functii pe date calendaristice: MONTHS_BETWEEN, TO_CHAR
SELECT c.nume,
    TO_CHAR(c.data_inregistrare, 'DD-MON-YYYY') AS data_inregistrare,
    TRUNC(MONTHS_BETWEEN(SYSDATE, c.data_inregistrare)) AS luni_de_la_inregistrare
FROM (SELECT id_client, nume, data_inregistrare FROM CLIENT) c
JOIN COMANDA co ON c.id_client = co.id_client
WHERE MONTHS_BETWEEN(SYSDATE, c.data_inregistrare) > 12;


--Afiseaza producatorii care au cel putin un produs in stoc, dar doar daca valoarea
--totala a produselor lor aflate in stoc este mai mare decat media valorii totale a
--stocurilor tuturor producatorilor. (afiseaza ID producator, nume producator, valoare totala stoc)
-- c) grupari date (id_producator, nume), functii grup (SUM,AVG), subcerere nesincronizata (HAVING)
SELECT
  p.id_producator,
  p.nume,
  SUM(s.cantitate_disp * s.pret) AS valoare_totala_stoc
FROM PRODUCATOR p
JOIN STOC s ON p.id_producator = s.id_producator
GROUP BY p.id_producator, p.nume
HAVING SUM(s.cantitate_disp * s.pret) > (
  SELECT AVG(valoare_stoc)
  FROM (
    SELECT SUM(cantitate_disp * pret) AS valoare_stoc
    FROM STOC
    GROUP BY id_producator
  )
);


--Afiseaza numele tuturor produselor, numele producatorului asociat fiecarui produs,
--reducerea produsului (daca nu exista sau este NULL, afiseaza 0) si starea stocului
--fiecarui produs ('Epuizat' daca este stoc 0, 'Mic' daca este 1, altfel 'Disponibil').
--Rezultatul ordoneaza-l alfabetic, dupa numele producatorului, apoi al produsului.
-- d) DECODE, NVL (in aceeasi cerere), ORDER BY
SELECT
  p.nume_produs,
  pr.nume AS nume_producator,
  NVL(r.procentaj, 0) AS reducere_produs,
  DECODE(s.cantitate_disp,
      0, 'Stoc epuizat',
      1, 'Stoc mic',
      'Disponibil') AS stare_stoc
FROM PRODUS p
JOIN STOC s ON p.id_produs = s.id_produs
```

```
JOIN PRODUCATOR pr ON s.id_producator =
pr.id_producator

LEFT JOIN REDUCERE r ON s.id_produs = r.id_produs
AND s.id_producator = r.id_producator

ORDER BY pr.nume, p.nume_produs;


--Afiseaza lista de produse cu reducerile lor si
clasificarea acestora

--in functie de marimea reducerii (=0 'Fara reducere',
intre 1 si 10

--'Reducere mica', >10 'Reducere mare')

-- f) WITH

-- e) CASE

WITH reduceri_produs AS (
    SELECT
        p.id_produs,
        p.nume_produs,
        r.procentaj
    FROM
        PRODUS p
    LEFT JOIN
        REDUCERE r
    ON
        p.id_produs = r.id_produs
)
SELECT
    rp.nume_produs,
    NVL(rp.procentaj, 0) AS reducere_produs,
    CASE
        WHEN NVL(rp.procentaj, 0) = 0 THEN 'Fara
reducere'
        WHEN NVL(rp.procentaj, 0) BETWEEN 1 AND 10
THEN 'Reducere mica'
        WHEN NVL(rp.procentaj, 0) > 10 THEN
'Reducere mare'
    END AS categorie_reducere
FROM
    reduceri_produs rp
ORDER BY
    rp.nume_produs;
```

13. Implementarea a 3 operații de actualizare și de suprimare a datelor utilizând subcereri

- Să se actualizeze prețurile produselor care au recenzii cu rating mai mic sau egal cu 2 astfel încât prețul lor să scadă cu 10%.

```
--Sa se actualizeze prețurile produselor care au recenzii cu rating <=2
--astfel incat pretul lor sa scada cu 10%
UPDATE STOC s
SET pret = pret * 0.9
WHERE id_produs IN (
    SELECT id_produs
    FROM RECENZIE
    WHERE rating <= 2
);
```

Script Output | Query Result

Task completed in 0.019 seconds

```
4 rows updated.
```

- Să se șteargă contactele clienților care au comandat produsul ‘pix albastru’ și prețul total al comenzii este mai mic decât 50.

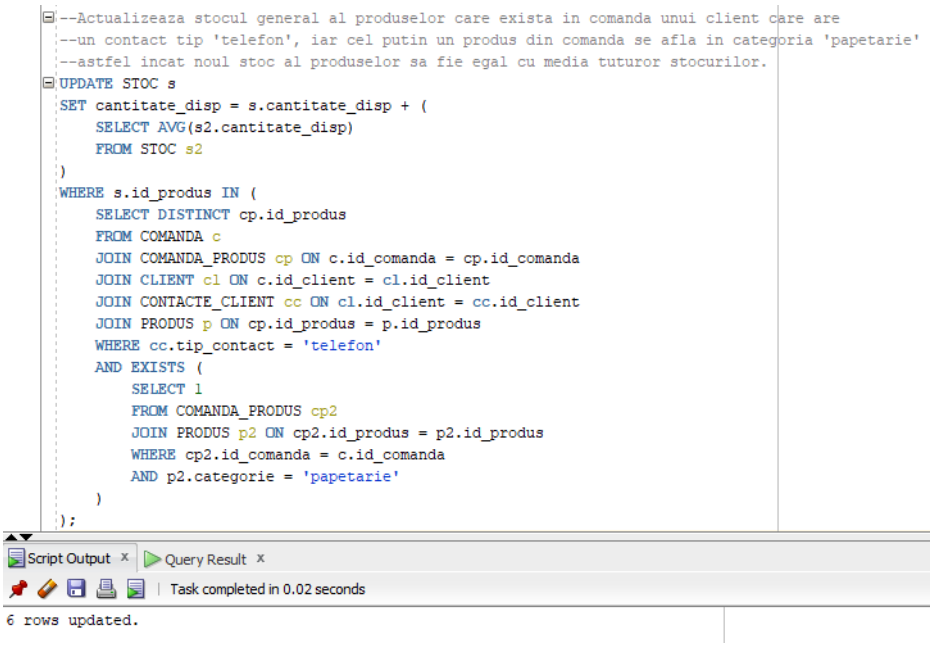
```
--Sa se stearga contactele clientilor care au comandat produsul
--'pix albastru' iar pretul total al comenzii este mai mic de 50
DELETE FROM CONTACTE_CLIENT
WHERE id_client IN (
    SELECT c.id_client
    FROM COMANDA c
    JOIN COMANDA_PRODUS cp ON c.id_comanda = cp.id_comanda
    JOIN PRODUS p ON cp.id_produs = p.id_produs
    WHERE p.nume_produs = 'pix albastru'
    AND c.id_comanda IN (
        SELECT cp2.id_comanda
        FROM COMANDA_PRODUS cp2
        GROUP BY cp2.id_comanda
        HAVING SUM(cp2.pret * cp2.cantitate) < 50
    )
);
```

Script Output

Task completed in 0.042 seconds

```
1 row deleted.
```

- Actualizează stocul general al produselor (din tabela STOC) pentru produsele care există în comanda unui client care are un contact de tip ‘telefon’, iar cel puțin un produs din comanda clientului respectiv se află în categoria ‘papetarie’, astfel încât noul stoc al produselor să fie egal cu stocul initial la care se adaugă media tuturor stocurilor.



```
--Actualizeaza stocul general al produselor care exista in comanda unui client care are
--un contact tip 'telefon', iar cel putin un produs din comanda se afla in categoria 'papetarie'
--astfel incat noul stoc al produselor sa fie egal cu media tuturor stocurilor.
UPDATE STOC s
SET cantitate_disp = s.cantitate_disp + (
    SELECT AVG(s2.cantitate_disp)
    FROM STOC s2
)
WHERE s.id_produs IN (
    SELECT DISTINCT cp.id_produs
    FROM COMANDA c
    JOIN COMANDA_PRODUS cp ON c.id_comanda = cp.id_comanda
    JOIN CLIENT cl ON c.id_client = cl.id_client
    JOIN CONTACTE_CLIENT cc ON cl.id_client = cc.id_client
    JOIN PRODUS p ON cp.id_produs = p.id_produs
    WHERE cc.tip_contact = 'telefon'
    AND EXISTS (
        SELECT 1
        FROM COMANDA_PRODUS cp2
        JOIN PRODUS p2 ON cp2.id_produs = p2.id_produs
        WHERE cp2.id_comanda = c.id_comanda
        AND p2.categorie = 'papetarie'
    )
);
```

Script Output | Query Result | Task completed in 0.02 seconds

```
6 rows updated.
```

```
--Sa se actualizeze prețurile produselor care au recenzii cu rating <=2
--astfel incat pretul lor sa scada cu 10%
UPDATE STOC s
SET pret = pret * 0.9
WHERE id_produs IN (
   SELECT id_produs
   FROM RECENZIE
   WHERE rating <= 2
);

--Sa se stearga contactele clientilor care au comandat produsul
--'pix albastru' iar pretul total al comenzii este mai mic de 50
DELETE FROM CONTACTE_CLIENT
WHERE id_client IN (
   SELECT c.id_client
   FROM COMANDA c
   JOIN COMANDA_PRODUS cp ON c.id_comanda = cp.id_comanda
   JOIN PRODUS p ON cp.id_produs = p.id_produs
   WHERE p.nume_produs = 'pix albastru'
   AND c.id_comanda IN (
      -- Subcerere pentru a verifica comenzile cu valoare totală < 50
      SELECT cp2.id_comanda
```

```
      FROM COMANDA_PRODUS cp2
      GROUP BY cp2.id_comanda
      HAVING SUM(cp2.pret * cp2.cantitate) < 50
   )
);

--Actualizeaza stocul general al produselor care exista in comanda unui client care are
--un contact tip 'telefon', iar cel putin un produs din comanda se afla in categoria 'papetarie'
--astfel incat noul stoc al produselor sa fie egal cu media tuturor stocurilor.
UPDATE STOC s
SET cantitate_disp = s.cantitate_disp + (
   SELECT AVG(s2.cantitate_disp)
   FROM STOC s2
)
WHERE s.id_produs IN (
   SELECT DISTINCT cp.id_produs
   FROM COMANDA c
   JOIN COMANDA_PRODUS cp ON c.id_comanda = cp.id_comanda
   JOIN CLIENT cl ON c.id_client = cl.id_client
   JOIN CONTACTE_CLIENT cc ON cl.id_client = cc.id_client
   JOIN PRODUS p ON cp.id_produs = p.id_produs
   WHERE cc.tip_contact = 'telefon'
   AND EXISTS (
      SELECT 1
      FROM COMANDA_PRODUS cp2
      JOIN PRODUS p2 ON cp2.id_produs = p2.id_produs
      WHERE cp2.id_comanda = c.id_comanda
      AND p2.categorie = 'papetarie'
   )
);
commit;
```

14. Crearea unei vizualizări complexe. Dați un exemplu de operație LMD permisă pe vizualizarea respectivă și un exemplu de operație LMD nepermisă

Crearea vizualizării complexe: Afișează informații detaliate despre comenzile plasate de clienți.

```
CREATE VIEW Vizualizare_Comenzi_Complete AS
SELECT
    C.id_comanda,
    Cl.nume AS nume_client,
    P.nume_produs,
    CP.cantitate,
    CP.pret,
    Pr.nume AS producator,
    T.firma AS firma_transport,
    T.timp,
    T.cost AS cost_transport,
    MP.tip_plata
FROM COMANDA C
JOIN CLIENT Cl ON C.id_client = Cl.id_client
JOIN COMANDA_PRODUS CP ON C.id_comanda = CP.id_comanda
JOIN PRODUS P ON CP.id_produs = P.id_produs
JOIN STOC S ON P.id_produs = S.id_produs
JOIN PRODUCATOR Pr ON S.id_producator = Pr.id_producator
JOIN TRANSPORT T ON C.id_transport = T.id_transport
JOIN MOD_PLATA MP ON C.id_modplata = MP.id_modplata;
```

Query Result | Script Output | Task completed in 0.059 seconds

```
View VIZUALIZARE_COMENZI_COMPLETE created.
```

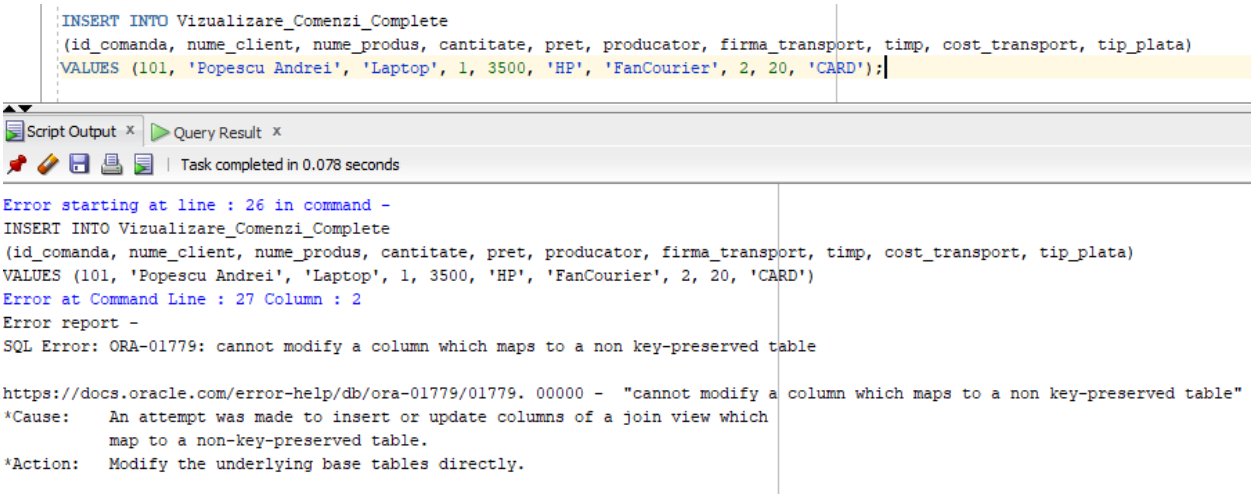
Exemplu de operație LMD permisă: Obține informații despre toate comenzile plătite cu card. Este o operație permisă deoarece doar interoghează datele dintr-o vizualizare definită printr-un SELECT.

```
SELECT *
FROM Vizualizare_Comenzi_Complete
WHERE tip_plata = 'card';
```

Script Output | Query Result | SQL | All Rows Fetched: 8 in 0.017 seconds

	ID_COMANDA	NUME_CLIENT	NUME_PRODUS	CANTITATE	PRET	PRODUCATOR	FIRMA_TRANSPORT	TIMP	COST_TRANSPORT	TIP_PLATA
1	2	SC Tehnoserv SRL	pix albastru	2	15.5	Zentari Corp	DPD	3	15.5	card
2	2	SC Tehnoserv SRL	caiet A5	1	15.5	Zentari Corp	DPD	3	15.5	card
3	4	Andrei Dumitrescu	mapa plastic	3	18.3	Zentari Corp	Cargus	4	18.3	card
4	2	SC Tehnoserv SRL	pix albastru	2	15.5	Novera Ltd	DPD	3	15.5	card
5	4	Andrei Dumitrescu	creion mecanic	1	18.3	Novera Ltd	Cargus	4	18.3	card
6	2	SC Tehnoserv SRL	caiet A5	1	15.5	Veltria GmbH	DPD	3	15.5	card
7	4	Andrei Dumitrescu	creion mecanic	1	18.3	Veltria GmbH	Cargus	4	18.3	card
8	4	Andrei Dumitrescu	mapa plastic	3	18.3	Solvexa SARL	Cargus	4	18.3	card

Exemplu de operație LMD nepermisă: Încercarea de a insera sau actualiza datele într-o vizualizare ce conține joinuri multiple, coloane din tabele diferite sau chei compuse. Operația de mai jos este nepermisă pentru că vizualizarea include date din multe tabele legate între ele și nu este clar cum să insereze date în fiecare tabel relevant. Vizualizarea nu este actualizabilă implicit.




```
CREATE VIEW Vizualizare_Comenzi_Complete AS
SELECT
   C.id_comanda,
   Cl.nume AS nume_client,
   P.nume_produs,
   CP.cantitate,
   CP.pret,
   Pr.nume AS producator,
   T.firma AS firma_transport,
   T.timp,
   T.cost AS cost_transport,
   MP.tip_plata
FROM COMANDA C
JOIN CLIENT Cl ON C.id_client = Cl.id_client
JOIN COMANDA_PRODUS CP ON C.id_comanda = CP.id_comanda
JOIN PRODUS P ON CP.id_produs = P.id_produs
```

```
JOIN STOC S ON P.id_produs = S.id_produs

JOIN PRODUCATOR Pr ON S.id_producator = Pr.id_producator

JOIN TRANSPORT T ON C.id_transport = T.id_transport

JOIN MOD_PLATA MP ON C.id_modplata = MP.id_modplata;


SELECT *

FROM Vizualizare_Comenzi_Complete

WHERE tip_plata = 'card';


INSERT INTO Vizualizare_Comenzi_Complete

(id_comanda, nume_client, nume_produs, cantitate, pret, producator, firma_transport, timp, cost_transport, tip_plata)

VALUES (101, 'Popescu Andrei', 'Laptop', 1, 3500, 'HP', 'FanCourier', 2, 20, 'CARD');
```

15. Formulați în limbaj natural și implementați în SQL: o cerere ce utilizează operația outer-join pe minimum 4 tabele, o cerere ce utilizează operația division și o cerere care implementează analiza top-n

a) Outer-join pe minimum 4 tabele

Afișează toate comenzile, inclusiv cele care nu au produse asociate, transport sau metodă de plată, împreună cu informații despre client, produs, transport și metodă de plată, dacă există.

```
SELECT
    C.id_comanda,
    Cl.nume AS nume_client,
    P.nume_produs,
    T.firma AS firma_transport,
    MP.tip_plata
FROM COMANDA C
LEFT JOIN CLIENT Cl ON C.id_client = Cl.id_client
LEFT JOIN COMANDA_PRODUS CP ON C.id_comanda = CP.id_comanda
LEFT JOIN PRODUS P ON CP.id_produs = P.id_produs
LEFT JOIN TRANSPORT T ON C.id_transport = T.id_transport
LEFT JOIN MOD_PLATA MP ON C.id_modplata = MP.id_modplata;
```

Script Output | Query Result

SQL | All Rows Fetched: 10 in 0.007 seconds

	ID_COMANDA	NUME_CLIENT	NUME_PRODUS	FIRMA_TRANSPORT	TIP_PLATA
1	1	Ion Popescu	pix albastru	Fan Courier	cash
2	2	SC Tehnoserv SRL	pix albastru	DPD	card
3	2	SC Tehnoserv SRL	caiet A5	DPD	card
4	3	Maria Ionescu	caiet A5	Sameday	cash
5	3	Maria Ionescu	agenda	Sameday	cash
6	5	DigitalNet Solutions	agenda	Bookurier	cash
7	1	Ion Popescu	mapa plastic	Fan Courier	cash
8	4	Andrei Dumitrescu	mapa plastic	Cargus	card
9	4	Andrei Dumitrescu	creion mecanic	Cargus	card
10	5	DigitalNet Solutions	creion mecanic	Bookurier	cash

b) Operația division

Afișează clienții care au dat comenzi pentru toate produsele din categoria ‘arta’.

```
SELECT C.id_client
FROM COMANDA C
JOIN COMANDA_PRODUS CP ON C.id_comanda = CP.id_comanda
JOIN PRODUS P ON CP.id_produs = P.id_produs
WHERE P.categorie = 'arta'
GROUP BY C.id_client
HAVING COUNT(DISTINCT P.id_produs) = (
    SELECT COUNT(DISTINCT id_produs)
    FROM PRODUS
    WHERE categorie = 'arta'
);
```

Script Output | Query Result

SQL | All Rows Fetched: 0 in 0.001 seconds

ID_CLIENT

c) Analiza top-n

Afișează top 3 clienți care au cheltuit cel mai mult în total pe comenzi.

```
SELECT
    C.id_client,
    Cl.nume,
    SUM(CP.pret * CP.cantitate) AS total_cheltuit
FROM COMANDA C
JOIN CLIENT Cl ON C.id_client = Cl.id_client
JOIN COMANDA_PRODUS CP ON C.id_comanda = CP.id_comanda
GROUP BY C.id_client, Cl.nume
ORDER BY total_cheltuit DESC
FETCH FIRST 3 ROWS ONLY;
```

Script Output | Query Result

SQL | All Rows Fetched: 3 in 0.011 seconds

	ID_CLIENT	NUME	TOTAL_CHELTUIT
1	4	Andrei Dumitrescu	73.2
2	1	Ion Popescu	59.97
3	3	Maria Ionescu	48

```
SELECT

   C.id_comanda,

   Cl.nume AS nume_client,

   P.nume_produs,

   T.firma AS firma_transport,

   MP.tip_plata

FROM COMANDA C

LEFT JOIN CLIENT Cl ON C.id_client = Cl.id_client

LEFT JOIN COMANDA_PRODUS CP ON C.id_comanda = CP.id_comanda

LEFT JOIN PRODUS P ON CP.id_produs = P.id_produs

LEFT JOIN TRANSPORT T ON C.id_transport = T.id_transport

LEFT JOIN MOD_PLATA MP ON C.id_modplata = MP.id_modplata;


SELECT C.id_client

FROM COMANDA C
```

```
JOIN COMANDA_PRODUS CP ON C.id_comanda = CP.id_comanda
JOIN PRODUS P ON CP.id_produs = P.id_produs
WHERE P.categorie = 'arta'
GROUP BY C.id_client
HAVING COUNT(DISTINCT P.id_produs) = (
   SELECT COUNT(DISTINCT id_produs)
   FROM PRODUS
   WHERE categorie = 'arta'
);

SELECT
   C.id_client,
   Cl.nume,
   SUM(CP.pret * CP.cantitate) AS total_cheltuit
FROM COMANDA C
JOIN CLIENT Cl ON C.id_client = Cl.id_client
JOIN COMANDA_PRODUS CP ON C.id_comanda = CP.id_comanda
GROUP BY C.id_client, Cl.nume
ORDER BY total_cheltuit DESC
FETCH FIRST 3 ROWS ONLY;
```

16. Optimizarea unei cereri, aplicând regulile de optimizare ce derivă din proprietățile operatorilor algebrei relaționale. Cererea va fi exprimată prin expresie algebrică, arbore algebric și limbaj (SQL), atât anterior cât și ulterior optimizării

Cerere: Afișează comenzile cu status “livrata”, plasate de clienți care au primit bonusuri de peste 20 lei, împreună cu metoda de plată și data comenzii.

a) Varianta inițială (neoptimizată)

Expresie algebrică:

$$\pi\_id\_comanda, data\_comanda, tip\_plata\ (\ \sigma_{status_{comanda}} =' LIVRATA' \wedge valoare_{bonus} > 20\ (\ (((COMANDA \bowtie CLIENT) \bowtie BONUSURI) \bowtie MOD\_PLATA)\ ))$$

σ este operatorul de selecție, π este de proiecție, iar ⋈ este de join.

Arbore algebric:

π (C.id_comanda, C.data_comanda, MP.tip_plata)

|

σ (C.status_comanda = 'LIVRATA') ∧ (B.valoare_bonus > 20) ∧ (C.id_client = B.id_client) ∧ (C.id_modplata = MP.id_modplata)

|

(COMANDA × BONUSURI × MOD_PLATA)

Limbaj (SQL):

```
SELECT C.id_comanda, C.data_comanda, MP.tip_plata
FROM COMANDA C
JOIN CLIENT CL ON C.id_client = CL.id_client
JOIN BONUSURI B ON CL.id_client = B.id_client
JOIN MOD_PLATA MP ON C.id_modplata = MP.id_modplata
WHERE C.status_comanda = 'livrata' AND B.valoare_bonus > 20;
```

Script Output | Query Result

SQL | All Rows Fetched: 2 in 0.007 seconds

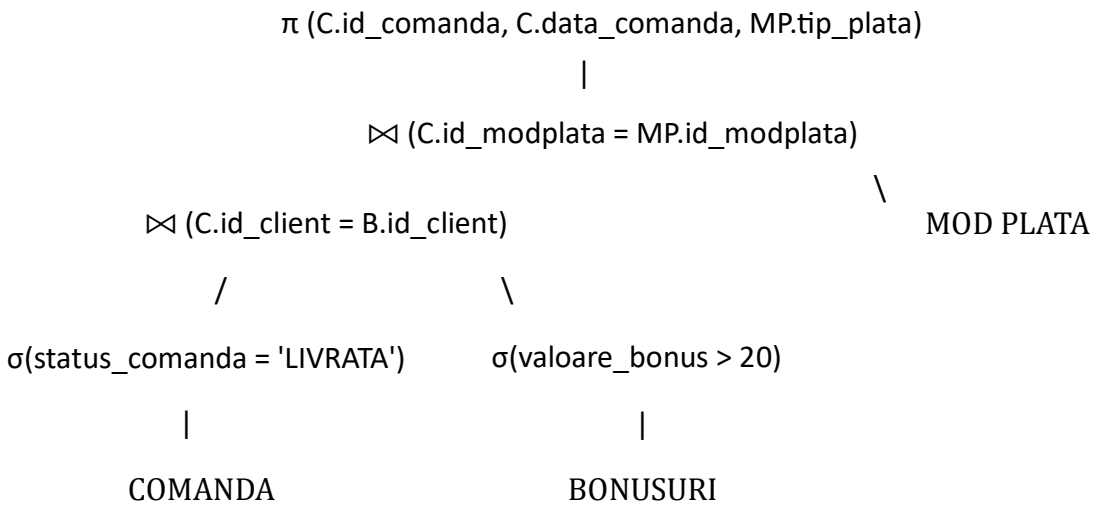
	ID_COMANDA	DATA_COMANDA	TIP_PLATA
1	2	02-MAY-25	card
2	5	05-MAY-25	cash

b) Varianta finală (optimizată)

Expresie algebrică:

$$\pi_{C.id\_comanda, C.data\_comanda, MP.tip\_plata}\Big(\big((\sigma_{status\_comanda = 'LIVRATA'}(COMANDA) \bowtie \sigma_{valoare\_bonus > 20}(BONUSURI)) \bowtie MOD\_PLATA\big)\Big)$$

Arbore algebric:



Limbaj SQL:

```
SELECT C.id_comanda, C.data_comanda, MP.tip_plata
FROM (
    SELECT * FROM COMANDA WHERE status_comanda = 'livrata'
) C
JOIN (
    SELECT * FROM BONUSURI WHERE valoare_bonus > 20
) B ON C.id_client = B.id_client
JOIN MOD_PLATA MP ON C.id_modplata = MP.id_modplata;
```

Script Output | Query Result

SQL | All Rows Fetched: 2 in 0.004 seconds

	ID_COMANDA	DATA_COMANDA	TIP_PLATA
1	2	02-MAY-25	card
2	5	05-MAY-25	cash

```
SELECT C.id_comanda, C.data_comanda, MP.tip_plata
FROM COMANDA C
JOIN CLIENT CL ON C.id_client = CL.id_client
JOIN BONUSURI B ON CL.id_client = B.id_client
JOIN MOD_PLATA MP ON C.id_modplata = MP.id_modplata
WHERE C.status_comanda = 'livrata' AND B.valoare_bonus > 20
```

```
SELECT C.id_comanda, C.data_comanda, MP.tip_plata
FROM (
   SELECT * FROM COMANDA WHERE status_comanda = 'livrata'
) C
JOIN (
   SELECT * FROM BONUSURI WHERE valoare_bonus > 20
) B ON C.id_client = B.id_client
JOIN MOD_PLATA MP ON C.id_modplata = MP.id_modplata
```

17. Realizarea normalizării BCNF, FN4, FN5 și aplicarea denormalizării, justificând eficiența

- Non BCNF:

MOD_PLATA	
ID_MODPLATA	TIP_PLATA
1	CASH
2	CARD

CASH	
ID_MODPLATA	MONEDA
1	RON

CARD			
ID_MODPLATA	NUMAR_CARD	DATA_EXPIRARE	TIP_CARD
2	2541365214526582	18-MAR-2029	VISA

- BCNF:

MOD_PLATA	
ID_MODPLATA	TIP_PLATA
1	CASH
2	CARD

CASH		
ID_CASH	ID_MODPLATA	MONEDA
1	1	RON

CARD				
ID_CARD	ID_MODPLATA	NUMAR_CARD	DATA_EXPIRARE	TIP_CARD
1	2	2541365214526582	18-MAR-2029	VISA

O schemă a bazei de date se află în Forma Normală Boyce-Codd (BCNF) dacă nu există anomalii cauzate de dependențele funcționale non-triviale în care determinantul nu este o cheie candidat. Este o formă normală mai strictă decât FN3. De aceea, trebuie să verificăm că fiecare tabel este în FN3, iar apoi să verificăm ca toate dependențele funcționale să fie determinate de o cheie candidat.

În NON-BCNF, id_modplata determină tip_plata, dar tip_plata determină indirect ce tabel secundar este utilizat (CASH sau CARD). Deci, tip_plata determină alte atribute și nu este cheie

candidat, așadar schema nu este în BCNF. Pentru a satisface BCNF, CASH și CARD sunt separate complet, eliminând relația directă cu mod_plata. Astfel, vom avea MOD_PLATA (id_modplata (PK), tip_plata), CASH (id_cash(PK), id_modplata(FK), moneda) și CARD(id_card(PK), id_modplata(FK), numar_card, data_expirare, tip_card).

- Non FN4:

CLIENT			
ID_CLIENT	NUME	TELEFON	EMAIL
1	Andrei Matei	0799123456	andrei@gmail.com
1	Andrei Matei	0722123456	matei@yahoo.ro

- FN4:

CONTACTE_CLIENT				
ID_CONTACTECLIENT	ID_CLIENT	TIP_CONTACT	VALOARE_CONTACT	STATUS_CONTACT
1	1	telefon	0799123456	activ
2	1	telefon	0722123456	inactiv
3	1	email	andrei@gmail.com	activ
4	1	email	matei@yahoo.com	activ

În FN4, nu este permis ca o entitate să aibă două sau mai multe seturi de atribute multivaluate independente între ele. Tabelul trebuie să se afle deja in FN4 și să se suprime toate multidependențele care nu sunt și dependențe funcționale. În exemplul din Non FN4, există două atribute multivaluate independente: telefon și email. Acestea nu se determină între ele, deci sunt dependențe multivaluate independente, încălcând FN4. În FN4, se creează separat tabela CONTACTE_CLIENT, fiecare contact având un ID unic, separând contactele în rânduri atomice cu tipuri.

- Non FN5:

COMANDA_PRODUS				
ID_COMANDA	ID_PRODUS	ID_PRODUCATOR	CANTITATE	PRET
101	301	501	2	50
101	302	502	1	80
102	301	501	1	50
102	303	503	3	90

- FN5:

COMANDA_PRODUS			
ID_COMANDA	ID_PRODUS	CANTITATE	PRET
101	301	2	50
101	301	1	80
102	301	1	50
102	303	1	90

STOC			
ID_PRODUCATOR	ID_PRODUS	CANTITATE_DISP	PRET
501	301	2	50
502	301	1	80
503	301	1	50

Un tabel se află în FN5 dacă deja se află în FN4 și elimină redundanțele datorate dependenței ciclice. Se suprimă toate join-dependențele care nu sunt implicate de o cheie. În Non FN5, COMANDA_PRODUS include o relație indirectă între id_produs, id_producator și id_comanda, dar nu descompune corect legăturile many-to-many între aceste entități.Dacă un

produs poate fi livrat de mai mulți producători, aceasta creează o anomalie de îmbinare care poate duce la redundanță și inconsistență. În FN5, COMANDA_PRODUS nu mai conține id_producator, astfel încât relațiile many-to-many dintre produs și producator sunt gestionate corect prin stoc.

Denormalizarea poate fi utilă în următoarele situații:

- îmbunătățirea performanței interogărilor: join-urile multiple pot încetini interogările complexe;

- reducerea încărcării pe server: fără denormalizare, interogările pot necesita unirea mai multor tabele, consumând resurse suplimentare;

- simplificarea accesului la date: aplicațiile care trebuie să acceseze frecvent date agregate pot beneficia de o structură mai simplă.

În schema prezentă, un exemplu de denormalizare poate presupune adăugarea numelui producătorului direct în produs. Pentru a afla producătorul unui produs, trebuie să facem join între produs și producător, ceea ce afectează performanța (se poate vedea în justificarea FN5). Pentru a aplica denormalizarea, adăugăm nume_producator direct în produs. Denormalizăm tabela produs pentru a reduce join-urile frecvente, adăugând numele producătorului. Astfel, se evită join-uri repetate, permite afișarea rapidă a produselor cu numele producătorului și reduce încărcarea bazei de date în query-urile frecvente.