

***PROIECT BAZE DE DATE - ANUL I,
SEMESTRUL II***

***Sistemul de gestiune a bazelor de date ale unor hoteluri
deținute de un antreprenor în România***

~ Bejenaru Ioana, FMI, grupa 133 ~

CUPRINS:

1. Descrierea modelului real, a utilității acestuia și a regulilor de funcționare.
2. Prezentarea constrângerilor (restricții, reguli) impuse asupra modelului.
3. Descrierea entităților, incluzând precizarea cheii primare.
4. Descrierea relațiilor, incluzând precizarea cardinalității acestora.
5. Descrierea atributelor, incluzând tipul de date și eventualele constrângeri, valori implicite, valori posibile ale atributelor.
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.
9. Realizarea normalizării până la forma normală 3 (FN1-FN3)
10. Crearea unei secvențe ce va fi utilizată în inserarea înregistrărilor în tabele (punctul 11).
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).
12. Formulați în limbaj natural și implementați 5 cereri SQL complexe ce vor utiliza, în ansamblul lor, următoarele elemente:
 - subcereri sincronizate în care intervin cel puțin 3 tabele
 - subcereri nesincronizate în clauza FROM
 - grupări de date, funcții grup, filtrare la nivel de grupuri cu subcereri nesincronizate (în clauza de HAVING) în care intervin cel puțin 3 tabele (în cadrul aceleiași cereri)
 - ordonări și utilizarea funcțiilor NVL și DECODE (în cadrul aceleiași cereri)
 - utilizarea a cel puțin 2 funcții pe șiruri de caractere, 2 funcții pe date calendaristice, a cel puțin unei expresii CASE
 - utilizarea a cel puțin 1 bloc de cerere (clauza WITH)
13. Implementarea a 3 operații de actualizare și de suprimare a datelor utilizând subcereri.
14. Crearea unei vizualizări complexe. Dați un exemplu de operație LMD permisă pe vizualizarea respectivă și un exemplu de operație LMD nepermisă.
15. Formulați în limbaj natural și implementați în SQL: o cerere ce utilizează operația outerjoin pe minimum 4 tabele, o cerere ce utilizează operația division și o cerere care implementează analiza top-n. (cele 3 cereri sunt diferite de cererile de la exercițiul 12)
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.
17. Realizarea normalizării BCNF, FN4, FN5 și aplicarea denormalizării, justificând necesitatea acesteia.

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

Sistemul de gestiune a bazelor de date ale unor hoteluri deținute de un antreprenor are ca scop evidența informațiilor cu privire la rezervările, clienții, angajații, tranzacțiile și abonamentele făcute în cadrul hotelurilor. Utilitatea acestui SGBD se regăsește în faptul că antreprenorul poate gestiona toate clădirile deținute cu ajutorul unui singur SGBD și poate crea statistici atât la nivel de hotel (sau lanț de hoteluri), cât și la nivelul întregii afaceri. De asemenea, antreprenorul poate avea mai multe hoteluri de același tip, formându-se astfel un lanț hotelier. În cadrul acestor hoteluri pot lucra angajați, dar doar într-unul singur și nu în mai multe, această regulă aplicându-se și în cazul departamentelor. Hotelurile dispun de mai multe facilități precum spa, piscină, sală de fitness, etc., iar acestea se pot regăsi la oricare din hotelurile antreprenorului. Pe baza acestor facilități există și abonamente care le includ ce pot fi deținute de clienți. Camerele pot fi de mai multe tipuri (dublă, triplă, garsonieră, etc.), acestea având un preț standard. Pentru fiecare hotel, prețul real al unei camere este aflat cu tariful atribuit de antreprenor.

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

- Un hotel poate avea mai multe facilități diferite, dar nu mai multe de același tip (ex: nu pot exista 2 cantine, 2 cinematografe, 2 saune, etc.).
- O rezervare poate fi făcută cu o singură tranzacție (nu se poate plăti în tranșe).
- Nu pot fi făcute rezervări pe aceeași cameră în aceeași zi.
- Pot exista camere cu același număr la hoteluri diferite.
- La tranzacțiile cu “tip_tranzacție = ‘CASH’”, atributul “banca” va fi NULL.
- Un angajat lucrează la un singur hotel și într-un singur departament.
- Abonamentele sunt “generale”, adică pot fi folosite la orice hotel deținut de antreprenor. Ele au specificații general valabile (exemplu: în abonamentul “GOLD” este inclus “SALA_FITNESS” dar nu toate hotelurile dispun de astfel de săli; așadar abonamentele pot conține opțiuni valabile la orice hotel al antreprenorului, dar și contrariul)
- Un client are un singur tip de abonament pe parcursul șederii acestuia la hotel.
- Pentru a modela realitatea, prețurile camerelor trebuie să fie diferite de la hotel la hotel și în funcție de stelele acestora. Astfel, hotelurile vor avea un atribut “tarif” care va fi înmulțit cu prețul camerelor.
- Din tabela “TIPURI_CAMERE” nu pot fi șterse linii deoarece ele reprezintă o orientare (la modul general) pentru hoteluri. Prețul camerelor pentru fiecare hotel este aflat cu tariful fiecăruia. Liniile pot fi totuși modificate.
- Precum tabela “TIPURI_CAMERE”, tabela “FACILITĂȚI” este o tabelă care conține lucruri general valabile (exemplu: PISCINĂ, SPA, SALĂ_DE_FITNESS, CANTINĂ). Astfel, nu se pot face operații de *DELETE* asupra datelor din tabelă, dar se pot modifica. (exemplu: piscina a existat, există și va exista ca facilități valabilă pentru un hotel dar prețul construcției poate varia sau pot exista mai multe tipuri)

3. Descrierea entităților, incluzând precizarea cheii primare.

DEPARTAMENT = subdiviziune a unei întreprinderi (hotel) în care lucrează angajați; cheia primară este **cod_departament**.

ANGAJAT = persoană care lucrează în cadrul unui departament și într-un hotel; cheia primară este **cod_angajat**.

HOTEL = clădire mare cu multe camere, destinată șederii clienților; cheia primară este **cod_hotel**.

ORAS = formă complexă de așezare umană care reprezintă locația hotelurilor; cheia primară este **cod_oras**.

CAMERA = încăperi ale hotelurilor ce pot fi rezervate de clienți; cheia primară este una compusă formată din tuplul (**cod_camera**, **cod_hotel**).

TIP_CAMERA = entitate ce denumește structura și tipul unei camere (ex: single, dublă, triplă, etc.); cheia primară este **cod_tip_camera**.

FACILITATE = loc / clădire care servește un anumit scop precum cel de relaxare, recreativ, ș.a.; cheia primară este **cod_facilitate**.

CLIENT = persoană care beneficiază de serviciile hotelurilor în urma unei tranzacții; cheia primară este **cod_client**.

ABONAMENT = conveție prin care, în schimbul unei sume, clienții au dreptul de a folosi anumite servicii ale hotelurilor pe o anumită perioadă și de un anumit număr de ori; cheia primară este **cod_abonament**;

TRANZACTIE = schimb comercial prin care se realizează o rezervare în cadrul unui hotel; cheia primară este **cod_tranzactie**;

4. Descrierea relațiilor, incluzând precizarea cardinalității acestora.

ANGAJAT lucrează_în **DEPARTAMENT** = între cele două entități se stabilește o relație de tip “one-to-many” care descrie faptul că un angajat poate lucra într-un singur departament; cardinalitatea acestei relații este următoarea: într-un departament pot lucra mai mulți angajați sau niciunul, un angajat trebuie să lucreze într-un departament. Astfel, minim avem 0:1, maxim n:1;

ANGAJAT lucrează_la **HOTEL** = între cele două entități se stabilește o relație de tip “one-to-many” care descrie faptul că un angajat poate lucra la un singur hotel; cardinalitatea

acestei relații este următoarea: într-un hotel pot lucra mai mulți angajați sau niciunul, un angajat trebuie să lucreze la un hotel. Astfel, minim avem 0:1, maxim n:1;

HOTEL aparține **ORAȘ** = între cele două entități se stabilește o relație de tip “one-to-many” care descrie faptul că un hotel (anume) poate aparține unui singur oraș; cardinalitatea acestei relații este următoarea: într-un oraș pot exista mai multe hoteluri sau niciunul, un hotel trebuie să aparțină unui oraș. Astfel, minim avem 0:1, maxim n:1;

HOTEL are **CAMERĂ** = între cele două entități se stabilește o relație de tip “one-to-many” care descrie faptul că o cameră poate aparține unui singur hotel; cardinalitatea acestei relații este următoarea: într-un hotel pot exista mai multe camere sau măcar una (pentru a putea exista acel hotel și pentru a-și îndeplini rolul de a primi clienți), o cameră aparține unui singur hotel. Astfel, minim avem 1:1, maxim n:1;

CAMERĂ reprezintă **TIP_CAMERĂ** = între cele două entități se stabilește o relație de tip “one-to-many” care descrie faptul că o cameră reprezintă un singur tip de cameră; cardinalitatea acestei relații este următoarea: un tip de cameră se poate regăsi în mai multe camere sau în niciuna, o cameră trebuie să reprezinte un tip de cameră. Astfel, minim avem 0:1, maxim n:1;

HOTEL dispune de **FACILITATE** = între cele două entități se stabilește o relație de tip “many-to-many” care descrie următoarea cardinalitate: un hotel poate dispune de mai multe facilități sau de niciuna, o facilitate poate fi regăsită în mai multe hoteluri sau în niciunul. Astfel, minim avem 0:0, maxim n:n.

ABONAMENT include **FACILITATE** = între cele două entități se stabilește o relație de tip “many-to-many” care descrie următoarea cardinalitate: un abonament poate dispune de mai multe facilități sau de niciuna, o facilitate poate fi inclusă într-un abonament sau în niciunul. Astfel, minim avem 0:0, maxim n:n.

CLIENT deține **ABONAMENT** = între cele două entități se stabilește o relație de tip “one-to-many” care descrie faptul că un client poate deține un singur abonament; cardinalitatea acestei relații este următoarea: un client poate și trebuie să aibă un singur tip de abonament, un abonament poate fi deținut de mai mulți clienți sau de niciunul. Astfel, minim avem 0:1, maxim n:1;

REZERVARE = relație de tip trei între entitățile **CLIENT**, **CAMERĂ**, **TRANZACȚIE** care ajută la gestionarea hotelurilor și prin care se poate controla foarte ușor aflarea informațiilor de tipul “Ce hoteluri a vizitat clientul X?”, “Care este suma totală cheltuită de un client X la hotelurile deținute de antreprenor”, “În ce camere a stat un anumit client la hotelul X?” ș.a.

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

ENTITĂȚI:

DEPARTAMENTE:

- `cod_departament#` = cheie primară (nu poate fi NULL), variabilă de tip `int` de maxim 3 cifre (exemplu: codul pentru departamentul “RECEPTIE” = 3)
- `denumire` = variabilă de tip `char` de maxim 15 caractere care reține numele departamentului (exemplu: BUCATARIE, RECEPTIE, SECRETARIAT, PAZA, CURĂȚENIE etc.)
- `ore_pe_luna` = variabilă de tip `int` de maxim 4 cifre care semnifică numărul de ore pe care un angajat dintr-un departament le are de îndeplinit pe lună.

ANGAJAȚI:

- `cod_angajat#` = cheie primară (nu poate fi NULL), variabilă de tip `int` de maxim 5 cifre (exemplu: codul pentru angajatul “Ungureanu Darius” = 8383)
- `cod_departament` = variabilă de tip `int` de maxim 3 cifre (care se regăsește și în entitatea DEPARTAMENT); reprezintă departamentul din care face parte un angajat
- `cod_hotel` = variabilă de tip `int` de maxim 4 cifre (care se regăsește și în entitatea HOTEL); reprezintă hotelul la care lucrează un angajat
- `nume` = variabilă de tip `char` de maxim 15 caractere care reprezintă numele de familie al angajatului
- `prenume` = variabilă de tip `char` de maxim 20 caractere care reprezintă prenumele angajatului
- `salariu` = variabilă de tip `int` de maxim 7 cifre care reprezintă venitul lunar al unui angajat
- `data_angajare` = variabilă de tip dată calendaristică care reprezintă ziua în care un angajat a început job-ul (exemplu: “15-SEP-2003”)

HOTELURI:

- `cod_hotel#` = cheie primară (nu poate fi NULL), variabilă de tip `int` de maxim 4 cifre care reprezintă codului unui hotel prin care acesta este identificat unic
- `cod_oras` = variabilă de tip `int` de maxim 4 cifre (care se regăsește și la entitatea ORAȘ); reprezintă codul orașului în care se află un hotel
- `denumire` = variabilă de tip `char` de maxim 15 caractere care reține numele hotelului (exemplu: ROYAL_ORCHID, SERENITY_SUITES, METROPOLITAN, etc.)
- `număr_stele` = variabilă de tip `int` de maxim 2 cifre
- `numar_etaje` = variabilă de tip `int` de maxim 2 cifre
- `tarif` = variabilă de tip `double` / `float` de maxim 2 cifre și 3 cifre decimale care arată “modul” în care vor fi calculate prețurile camerelor în funcție de fiecare hotel în parte

ORAȘE:

- `cod_oras#` = cheie primară (nu poate fi NULL), variabilă de tip `int` de maxim 4 cifre și care identifică unic un oraș din România
- `denumire` = variabilă de tip `char` de maxim 15 caractere care reține numele orașului (exemplu: VASLUI, IAȘI, BUCUREȘTI, etc.)

- populatie = variabilă de tip int de maxim 12 cifre care reprezintă numărul de locuitori ai unui oraș

FACILITĂȚI:

- cod_facilitate# = cheie primară (nu poate fi NULL), variabilă de tip int de maxim 4 cifre și care identifică unic un tip de spațiu din cadrul unui hotel
- pret_construcie = variabilă de tip int de maxim 10 cifre care reprezintă suma alocată construirii unui astfel de spațiu
- denumire = variabilă de tip char de maxim 15 caractere care reține numele facilității (exemplu: PISCINĂ, SALA_DE_FITNES, SPA, etc.)

ABONAMENTE:

- cod_abonament# = cheie primară (nu poate fi NULL), variabilă de tip int de maxim 3 cifre și care identifică unic un tip de abonament care este deținut de un client
- pret_pe_zi = variabilă de tip int de maxim 4 cifre (poate fi 0 în cazul abonamentului "STANDARD") care reprezintă suma plătită pentru a beneficia de serviciile abonamentului într-o zi
- denumire = variabilă de tip char de maxim 15 caractere care reține numele unui abonament (exemplu: STANDARD, BRONZE, SILVER, GOLD etc.)

CLIEȚI:

- cod_client# = cheie primară (nu poate fi NULL), variabilă de tip int de maxim 15 cifre și care identifică unic un client care a venit sau este cazat la hotel
- cod_abonament = variabilă de tip int de maxim 4 cifre (care se regăsește și la entitatea ABONAMENT); reprezintă codul abonamentului deținut de un client
- nume = variabilă de tip char de maxim 15 caractere care reprezintă numele de familie al clientului
- prenume = variabilă de tip char de maxim 20 caractere care reprezintă prenumele clientului
- data_nastere = variabilă de tip dată calendaristică care reprezintă ziua de naștere a unui client (exemplu: "15-SEP-2003")
- numar_telefon = variabilă de tip char de maxim 12 caractere

TRANZACȚII:

- cod_tranzactie# = cheie primară (nu poate fi NULL), variabilă de tip int de maxim 15 cifre și care identifică unic o tranzacție pentru o rezervare
- tip_tranzactie = variabilă de tip char de maxim 5 caractere care poate lua doar două valori ("CASH", "CARD")
- banca = variabilă de tip char de maxim 15 caractere care reprezintă numele unei bănci în cazul în care "tip_tranzacție = 'CARD' " și NULL în caz contrar
- suma = variabilă de tip int de maxim 10 cifre care reprezintă valoarea plătită pentru o rezervare

CAMERE:

- cod_camera# = cheie primară (nu poate fi NULL), variabilă de tip int de maxim 4 cifre și care alături de "cod_hotel#" identifică unic o cameră care aparține unui anumit hotel (ne folosim de faptul că o cameră nu poate exista fără un hotel și de faptul că vrem să modelăm realitatea

- `cod_hotel#` = împreună cu “`cod_cameră`” formează cheia primară (nu poate fi NULL), variabilă de tip int de maxim 4 cifre (se regăsește și la entitatea HOTEL)
- `cod_tip_camera` = variabilă de tip int de maxim 2 cifre
- `etaj` = variabilă de tip int de maxim 2 cifre care reprezintă nivelul la care se află camera

TIPURI_CAMERE:

- `cod_tip_camera#` = cheie primară (nu poate fi NULL), variabilă de tip int de maxim 2 cifre și care identifică unic un tip de cameră
- `denumire` = variabilă de tip char de maxim 15 caractere care reține numele unui camere (exemplu: SINGLE, DUBLĂ, TRIPLĂ, etc.)
- `pret_noapte` = variabilă de tip int de maxim 4 cifre
- `capacitate` = variabilă de tip int de maxim 2 cifre care face referire la numărul de oameni care pot fi cazați într-o cameră de un anumit tip

RELAȚII:

FACILITĂȚI_HOTELURI:

- `cod_hotel#` = parte din cheia primară (nu poate fi NULL), variabilă de tip int de maxim 4 cifre
- `cod_facilitate#` = parte din cheia primară (nu poate fi NULL), variabilă de tip int de maxim 4 cifre
- `numar_locuri` = variabilă de tip int de maxim 6 cifre care specifică câte persoane încap într-un loc amenajat (piscină, spa, ...) al hotelului
- `ora_deschidere` = variabilă de tip char de maxim 6 caractere care indică ora începerii programului pentru o anumită parte a hotelului
- `ora_închidere` = variabilă de tip char de maxim 6 caractere care indică ora terminării programului pentru o anumită parte a hotelului

FACILITĂȚI_ABONAMENTE:

- `cod_facilitate#` = parte din cheia primară (nu poate fi NULL), variabilă de tip int de maxim 4 cifre
- `cod_abonament#` = parte din cheia primară (nu poate fi NULL), variabilă de tip int de maxim 3 cifre
- `numar_intrari_zi` = variabilă de tip int de maxim 3 cifre care indică de câte ori poate fi accesată o zonă a hotelului în cadrul unui anumit abonament

REZERVĂRI:

- `cod_camera#` = parte din cheia primară (nu poate fi NULL), variabilă de tip int de maxim 4 cifre
- `cod_hotel#` = parte din cheia primară (nu poate fi NULL), variabilă de tip int de maxim 4 cifre
- `cod_client#` = parte din cheia primară (nu poate fi NULL), variabilă de tip int de maxim 15 cifre
- `cod_tranzactie#` = parte din cheia primară (nu poate fi NULL), variabilă de tip int de maxim 15 cifre

- `data_venire#` = parte din cheia primară (nu poate fi NULL), variabilă de tip dată calendaristică care indică ziua începerii unei rezervări;
 De ce avem, nevoie de “`data_venire`” în tuplul pentru cheia primară?
 Să presupunem următorul scenariu: cheia primară este formată doar din următorul tuplu (`cod_cameră`, `cod_hotel`, `cod_client`, `cod_tranzacție`).

cod_cameră	cod_hotel	cod_client	cod_tranzacție
1	1	1	1
1	1	1	2
1	1	2	3

În această situație, dacă cheia primară ar fi aleasă în acest mod, realitatea nu ar mai fi conturată în SGBD hotelurilor unui antreprenor și ar putea apărea situații nedorite precum un client care realizează mai multe rezervări pe aceeași cameră în aceeași zi sau aceeași cameră rezervată de mai mulți clienți în aceeași zi.

Soluția ar fi să fie adăugat un atribut nou în componența cheii primare, intitulat “`data_venire`”, care să rezolve problemele menționate anterior. Totuși, este de ajuns?

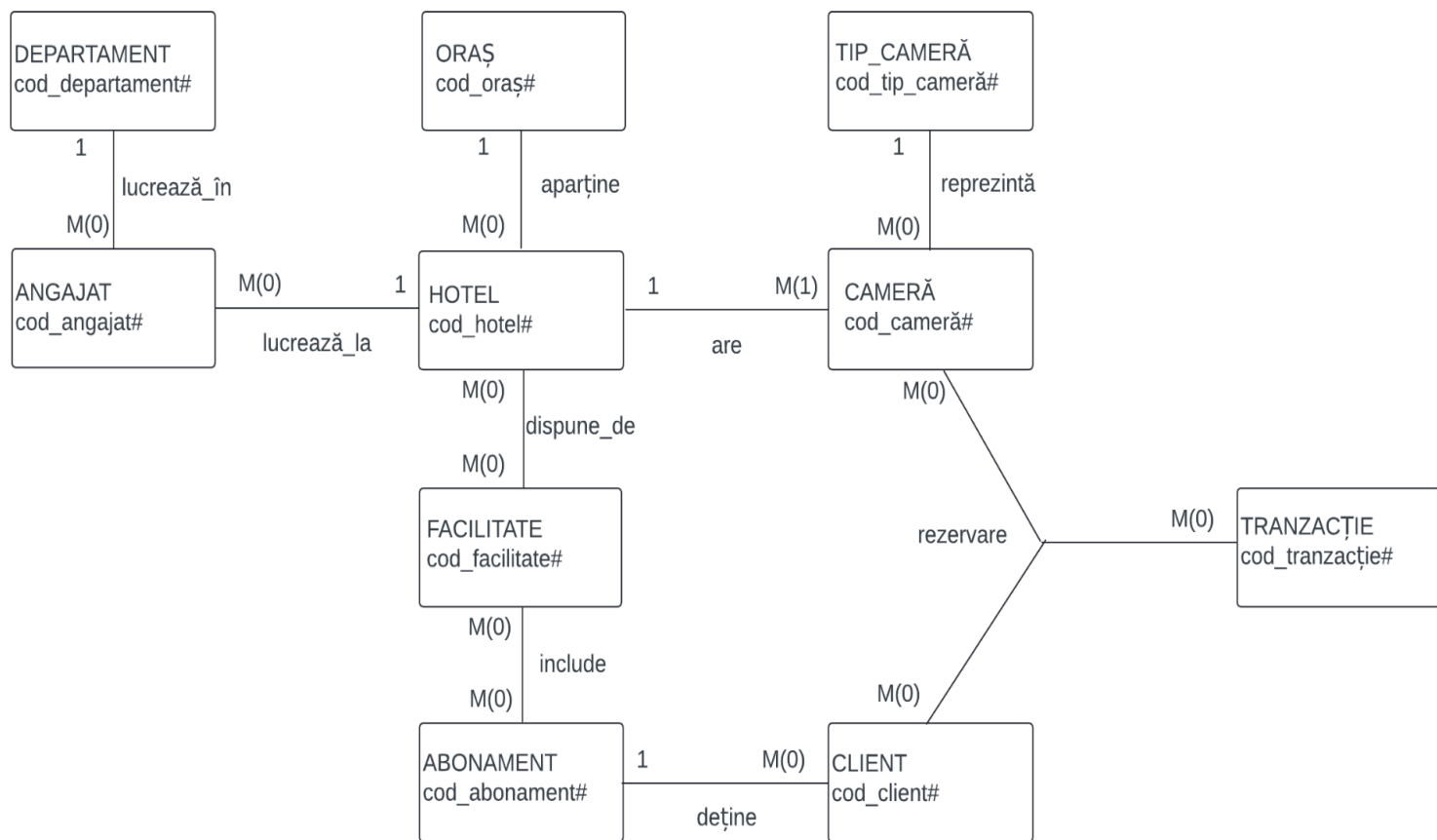
Prin tabelul următor se pot observa următoarele lucruri:

cod_cameră	cod_hotel	cod_client	cod_tranzacție	data_venire
1	1	1	1	1
1	1	1	2	1
1	1	2	3	1

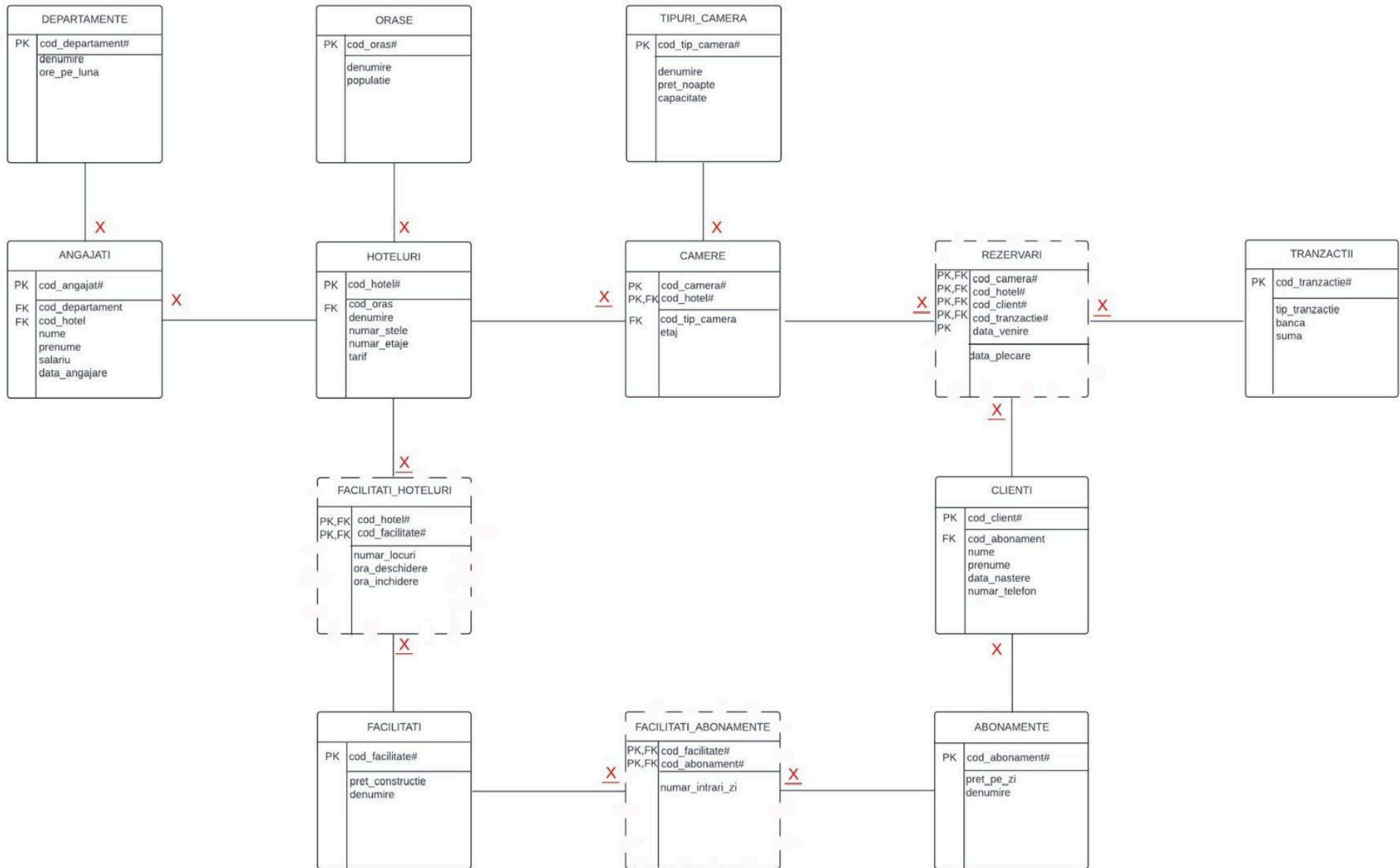
În continuare, apariția atributului “`data_venire`” în cadrul cheii primare nu a rezolvat problemele de la primul tabel. Pentru a rezolva definitiv aceste inconveniențe este de ajuns să setăm tuplul (`cod_cameră`, `cod_hotel`, `data_venire`) din cheia primară a tabelului asociativ ca fiind *unic*.

- `data_plecure` = variabilă de tip dată calendaristică care indică ziua încheierii unei rezervări

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



- 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.

DEPARTAMENTE (cod_departament# - PK, denumire, ore_pe_lună)

ANGAJAȚI (cod_angajat# - PK, cod_departament - FK, cod_hotel - FK, nume, prenume, salariu, dată_angajare)

HOTELURI (cod_hotel# - PK, cod_orăș - FK, denumire, număr_stele, număr_etaje, tarif)

CAMERE (cod_cameră# - PK, cod_hotel# - PK, FK, cod_tip_cameră - FK, etaj)

ORAȘE (cod_orăș# - PK, denumire, populație)

TIPURI_CAMERĂ (cod_tip_cameră# - PK, denumire, preț_noapte, capacitate)

FACILITĂȚI_HOTELURI (cod_hotel# - PK, FK, cod_facilitate - PK, FK, număr_locuri, ora_deschidere, ora_închidere)

FACILITĂȚI (cod_facilitate# - PK, preț_construcție, denumire)

FACILITĂȚI_ABONAMENTE (cod_facilitate# - PK, FK, cod_abonament# - PK, FK, număr_intrări_zi)

ABONAMENTE (cod_abonament# - PK, preț_zi, denumire)

CLIEȚI (cod_client# - PK, cod_abonament - FK, nume, prenume, dată_naștere, număr_telefon)

REZERVĂRI (cod_cameră# - PK, FK, cod_hotel# - PK, FK, cod_tranzacție# - PK, FK, cod_client# - PK, FK, dată_venire# - PK, dată_plecure)

TRANZACȚII (cod_tranzacție# - PK, tip_tranzacție, bancă, sumă)

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

➔ **FN1** este o formă de normalizare care face referire la faptul că fiecărui atribut îi corespunde o valoare atomică în cadrul unei relații. Ce înseamnă asta de fapt? Luând exemplele următoare putem observa faptul că fiecărui oraș îi corespunde un hotel și fiecărui hotel îi corespunde un oraș. Atfel vom avea în tabele înregistrări de tipul celor de la exemplul cu FN1. Alte exemple de FN1 sunt următoarele: fiecărui angajat îi corespunde un departament și datele nu sunt memorate “cod_departament - cod_angajat1, cod_angajat2, ... cod_angajat_n”; datele pentru evidența facilităților

care sunt incluse în abonamente nu sunt memorate sub forma “cod_abonament - cod_facilitate1, cod_facilitate2, ... cod_facilitaten”.

Non - FN1:

cod_orăș#	hoteluri
1	1,3,5
2	2, 4

FN1:

cod_orăș#	cod_hotel#
1	1
1	3
1	5
2	2
2	4

→ **FN2** reprezintă a doua formă de normalizare și face referire la faptul că relația trebuie să fie neapărat în FN1 și atributele din cadrul relației trebuie să fie dependente de întreaga cheie primară, nu parțial sau deloc.

Non-FN2: (numar_stele și numar_etaj fac referire la hotel)

cod_camera#	cod_hotel#	cod_tip_camera	etaj (cam.)	numar_stele	numar_etaje
1	1	2	1	5	7
2	13	2	4	3	6

FN2:

cod_camera#	cod_hotel#	cod_tip_camera	etaj
1	1	2	1
2	13	2	4

cod_hotel#	numar_stele	numar_etaje	alte_informatii
1	5	7	—
13	3	6	—

Din cauza faptului că relația de la non-FN2 avea attribute care depindeau doar de cod_hotel# și nu de întreaga cheie primară, relația este împărțită în alte două relații.

→ **FN3** reprezintă a treia formă de normalizare și face referire la faptul că o relație trebuie să fie neapărat în FN2 și fiecare atribut din cadrul relației nu este dependent tranzitiv de nicio cheie din relație.

Non-FN3:

cod_camera#	cod_hotel#	cod_tip_camera	etaj	pret
1	1	2	1	140
2	13	2	3	140

În cazul acestei relații (K1,K2,X,Y,Z) reprezintă relația cu K1 = cod_camera#, K2 = cod_hotel#, X = cod_tip_camera, Y = etaj, Z = pret.

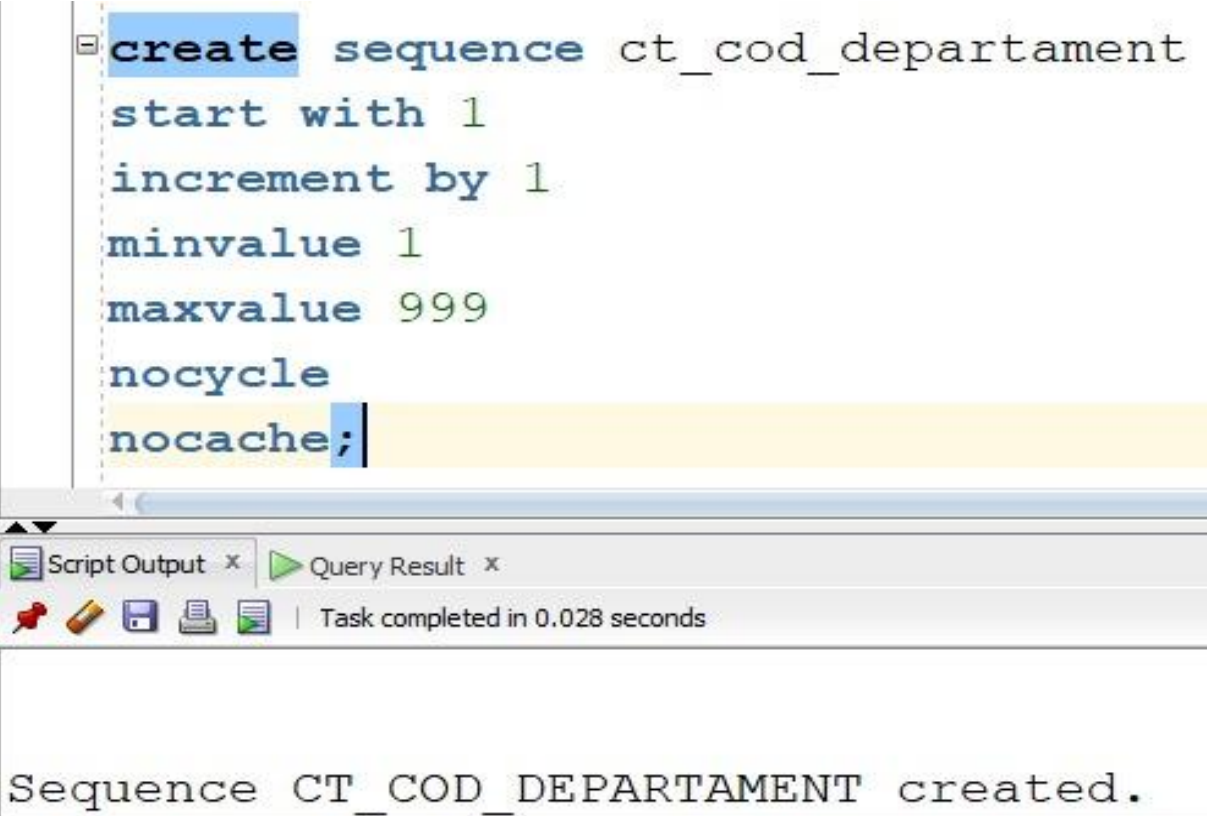
În cadrul bazei de date reprezentate în acest proiect, prețul depinde de cod_tip_camera deoarece el este unul orientativ setat de antreprenor, urmând ca prețul real al camerei (cod_camera, cod_hotel) să fie aflat în urma aplicării tarifului hotelului corespunzător. Astfel apare următoarea dependență tranzitivă $K1 \rightarrow X \rightarrow Z$, ceea ce face ca această relație să fie non-FN3.

FN3:

cod_camera#	cod_hotel#	cod_tip_camera	etaj
1	1	2	1
2	13	2	3

cod_tip_camera#	pret	alte_informatii
1	100	—
2	140	—

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



The screenshot shows a SQL IDE window with a script editor and a results pane. The script editor contains the following SQL code:

```
create sequence ct_cod_departament  
start with 1  
increment by 1  
minvalue 1  
maxvalue 999  
nocycle  
nocache;
```

The results pane shows the message: "Sequence CT_COD_DEPARTAMENT created." The status bar at the bottom indicates "Task completed in 0.028 seconds".

```
create sequence ct_cod_departament  
start with 1  
increment by 1  
minvalue 1  
maxvalue 999  
nocycle  
nocache;
```

```
create sequence ct_cod_oras  
start with 1  
increment by 1  
minvalue 1  
maxvalue 9999  
nocycle  
nocache;
```

```
create sequence ct_cod_hotel
```

```
start with 1
increment by 1
minvalue 1
maxvalue 9999
nocycle
nocache;
```

```
create sequence ct_cod_angajat
start with 1
increment by 1
minvalue 1
maxvalue 99999
nocycle
nocache;
```

```
create sequence ct_cod_facilitate
start with 1
increment by 1
minvalue 1
maxvalue 9999
nocycle
nocache;
```

```
create sequence ct_cod_tip_camera
start with 1
increment by 1
minvalue 1
maxvalue 99
nocycle
nocache;
```

```
create sequence ct_cod_client
start with 1
increment by 1
minvalue 1
maxvalue 9999999999999999
nocycle
nocache;
```

```
create sequence ct_cod_tranzactie
start with 1
increment by 1
minvalue 1
maxvalue 9999999999999999
```



```
nocycle  
nocache;
```

Următoarele secvențe sunt create pentru a ști să numerotăm camerele fiecărui hotel. Se utilizează această variantă pentru a avea în baza de date numere realiste (numerotarea camerelor la orice hotel începe cu 1).

```
create sequence ct_cod_camera_hotel_1  
start with 1  
increment by 1  
minvalue 1  
maxvalue 9999  
nocycle  
nocache;
```

```
create sequence ct_cod_camera_hotel_2  
start with 1  
increment by 1  
minvalue 1  
maxvalue 9999  
nocycle  
nocache;
```

```
create sequence ct_cod_camera_hotel_3  
start with 1  
increment by 1  
minvalue 1  
maxvalue 9999  
nocycle  
nocache;
```

```
create sequence ct_cod_camera_hotel_4  
start with 1  
increment by 1  
minvalue 1  
maxvalue 9999  
nocycle  
nocache;
```

```
create sequence ct_cod_camera_hotel_5  
start with 1  
increment by 1  
minvalue 1  
maxvalue 9999
```

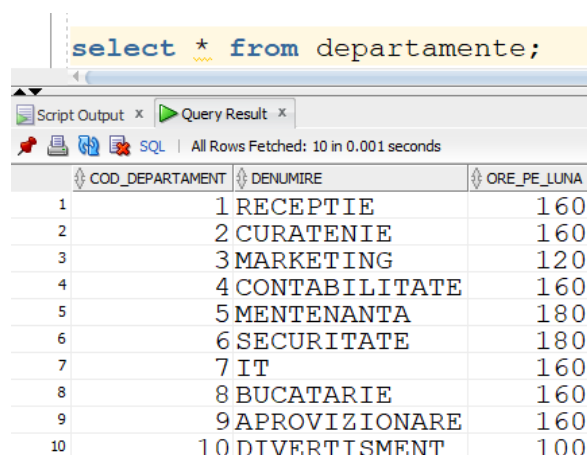
```
nocycle  
nocache;
```

...

```
create sequence ct_cod_camera_hotel_n  
start with 1  
increment by 1  
minvalue 1  
maxvalue 9999  
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).

DEPARTAMENTE:



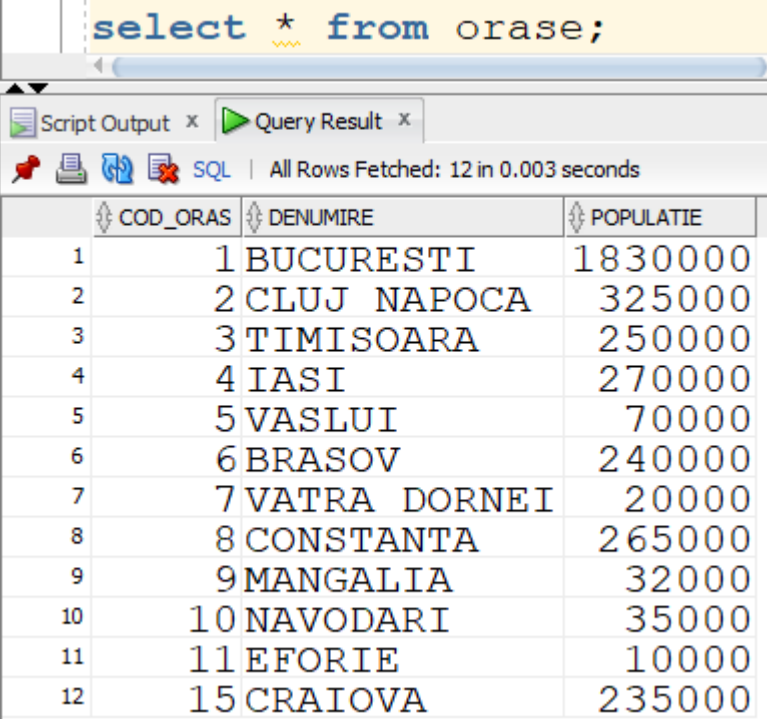
	COD_DEPARTAMENT	DENUMIRE	ORE_PE_LUNA
1	1	RECEPTIE	160
2	2	CURATENIE	160
3	3	MARKETING	120
4	4	CONTABILITATE	160
5	5	MENTENANTA	180
6	6	SECURITATE	180
7	7	IT	160
8	8	BUCATARIE	160
9	9	APROVIZIONARE	160
10	10	DIVERTISMENT	100

```
create table departamente(  
    cod_departament number(3) constraint pk_cod_departament primary key,  
    denumire varchar(15) constraint n_denumire not null unique,  
    ore_pe_luna number(3) constraint n_ore_pe_luna not null);
```

```
insert into departamente values(ct_cod_departament.nextval,'RECEPTIE',160);  
insert into departamente values(ct_cod_departament.nextval,'CURATENIE',160);  
insert into departamente values(ct_cod_departament.nextval,'MARKETING',120);  
insert into departamente values(ct_cod_departament.nextval,'CONTABILITATE',160);  
insert into departamente values(ct_cod_departament.nextval,'MENTENANTA',180);  
insert into departamente values(ct_cod_departament.nextval,'SECURITATE',180);  
insert into departamente values(ct_cod_departament.nextval,'IT',160);  
insert into departamente values(ct_cod_departament.nextval,'BUCATARIE',160);  
insert into departamente values(ct_cod_departament.nextval,'APROVIZIONARE',160);
```

```
insert into departamente values(ct_cod_departament.nextval,'DIVERTISMENT',100);
```

ORASE:



The screenshot shows an SQL query result in a web interface. The query is `select * from orase;`. The result is displayed in a table with three columns: `COD_ORAS`, `DENUMIRE`, and `POPULATIE`. The table contains 12 rows of data, numbered 1 to 12. The data is as follows:

	COD_ORAS	DENUMIRE	POPULATIE
1	1	BUCURESTI	1830000
2	2	CLUJ NAPOCA	325000
3	3	TIMISOARA	250000
4	4	IASI	270000
5	5	VASLUI	70000
6	6	BRASOV	240000
7	7	VATRA DORNEI	20000
8	8	CONSTANTA	265000
9	9	MANGALIA	32000
10	10	NAVODARI	35000
11	11	EFORIE	10000
12	15	CRAIOVA	235000

```
create table orase(  
    cod_oras number(4) constraint pk_cod_oras primary key,  
    denumire varchar(15) constraint n_denumire_orase not null unique,  
    populatie number(12) constraint n_populatie not null);
```

```
insert into orase values(ct_cod_oras.nextval,'BUCURESTI',1830000);  
insert into orase values(ct_cod_oras.nextval,'CLUJ_NAPOCA',325000);  
insert into orase values(ct_cod_oras.nextval,'TIMISOARA',250000);  
insert into orase values(ct_cod_oras.nextval,'IASI',270000);  
insert into orase values(ct_cod_oras.nextval,'VASLUI',70000);  
insert into orase values(ct_cod_oras.nextval,'BRASOV',240000);  
insert into orase values(ct_cod_oras.nextval,'VATRA_DORNEI',20000);  
insert into orase values(ct_cod_oras.nextval,'CONSTANTA',265000);  
insert into orase values(ct_cod_oras.nextval,'MANGALIA',32000);  
insert into orase values(ct_cod_oras.nextval,'NAVODARI',35000);  
insert into orase values(ct_cod_oras.nextval,'EFORIE',10000);  
insert into orase values(ct_cod_oras.nextval,'CRAIOVA',235000);
```

HOTELURI:

`select * from hoteluri;`

	COD_HOTEL	COD_ORAS	DENUMIRE	NUMAR_STELE	NUMAR_ETAJE	TARIF
1	1	1	ROYAL ORCHID	5	7	3.2
2	2	2	ROYAL ORCHID	5	7	3.2
3	3	3	ROYAL ORCHID	5	7	3.2
4	4	4	ROYAL ORCHID	5	7	3.2
5	5	8	EMERALD SHORES	4	5	2.8
6	6	9	EMERALD SHORES	4	5	2.8
7	7	11	EMERALD SHORES	4	5	2.8
8	8	8	OCEAN BREEZE	3	6	2.4
9	9	9	OCEAN BREEZE	3	6	2.4
10	10	10	OCEAN BREEZE	3	6	2.4
11	11	1	BUSINESS OASIS	5	6	3.2
12	12	2	BUSINESS OASIS	5	6	3.2
13	13	6	EVEREST	3	6	2.4
14	14	7	EVEREST	3	6	2.4
15	15	5	ARRAKIS	3	4	2.4

```
create table hoteluri(
    cod_hotel number(4) constraint pk_cod_hotel primary key,
    cod_oras number(4) constraint n_cod_oras not null,
    denumire varchar(15) constraint n_denumire_hoteluri not null,
    numar_stele number(2) constraint n_numar_stele not null,
    numar_etaje number(2) constraint n_numar_etaje not null,
    tarif number(2,3) constraint n_tarif not null,
    constraint fk_oras
        foreign key(cod_oras)
        references orase(cod_oras) on delete cascade);
```

```
insert into hoteluri values(ct_cod_hotel.nextval,1,'ROYAL_ORCHID',5,7,3.2);
insert into hoteluri values(ct_cod_hotel.nextval,2,'ROYAL_ORCHID',5,7,3.2);
insert into hoteluri values(ct_cod_hotel.nextval,3,'ROYAL_ORCHID',5,7,3.2);
insert into hoteluri values(ct_cod_hotel.nextval,4,'ROYAL_ORCHID',5,7,3.2);
insert into hoteluri values(ct_cod_hotel.nextval,8,'EMERALD_SHORES',4,5,2.8);
insert into hoteluri values(ct_cod_hotel.nextval,9,'EMERALD_SHORES',4,5,2.8);
insert into hoteluri values(ct_cod_hotel.nextval,11,'EMERALD_SHORES',4,5,2.8);
insert into hoteluri values(ct_cod_hotel.nextval,8,'OCEAN_BREEZE',3,6,2.4);
insert into hoteluri values(ct_cod_hotel.nextval,9,'OCEAN_BREEZE',3,6,2.4);
insert into hoteluri values(ct_cod_hotel.nextval,10,'OCEAN_BREEZE',3,6,2.4);
insert into hoteluri values(ct_cod_hotel.nextval,1,'BUSINESS_OASIS',5,6,3.2);
insert into hoteluri values(ct_cod_hotel.nextval,2,'BUSINESS_OASIS',5,6,3.2);
insert into hoteluri values(ct_cod_hotel.nextval,6,'EVEREST',3,6,2.4);
insert into hoteluri values(ct_cod_hotel.nextval,7,'EVEREST',3,6,2.4);
insert into hoteluri values(ct_cod_hotel.nextval,5,'ARRAKIS',3,4,2.4);
```

TIPURI_CAMERA:

select * from tipuri_camera;

	COD_TIP_CAMERA	DENUMIRE	PRET_NOAPTE	CAPACITATE
1	1	SINGLE	100	1
2	2	DOUBLE	140	2
3	3	TRIPLE	180	3
4	4	GARSONIERA	250	2
5	5	APARTAMENT	320	4
6	6	PENTHOUSE	700	6

```
create table tipuri_camera(
  cod_tip_camera number(2) constraint pk_cod_tip_camera primary key,
  denumire varchar(15) constraint n_denumire_tipuri_camera not null,
  pret_noapte number(4) constraint n_pret_noapte not null,
  capacitate number(2) constraint n_capacitate not null);
```

```
insert into tipuri_camera values(ct_cod_tip_camera.nextval,'SINGLE',100,1);
insert into tipuri_camera values(ct_cod_tip_camera.nextval,'DOUBLE',140,2);
insert into tipuri_camera values(ct_cod_tip_camera.nextval,'TRIPLE',180,3);
insert into tipuri_camera values(ct_cod_tip_camera.nextval,'GARSONIERA',250,2);
insert into tipuri_camera values(ct_cod_tip_camera.nextval,'APARTAMENT',320,4);
insert into tipuri_camera values(ct_cod_tip_camera.nextval,'PENTHOUSE',700,6);
```

FACILITĂȚI:

select * from facilitati;

	COD_FACILITATE	PRET_CONSTRUCTIE	DENUMIRE
1	1	500000	PISCINA INT
2	2	500000	PISCINA EXT
3	3	200000	SPA
4	4	300000	SALA FITNESS
5	5	700000	CINEMATOGRAF
6	6	600000	RESTAURANT
7	7	200000	SAUNA
8	8	1000000	PARCARE
9	9	400000	SALA EVENIMENTE

```
create table facilitati(
  cod_facilitate number(4) constraint pk_cod_facilitate primary key,
  pret_constructie number(10) constraint n_pret_constructie not null,
  denumire varchar(15) constraint n_denumire_facilitati not null);
```

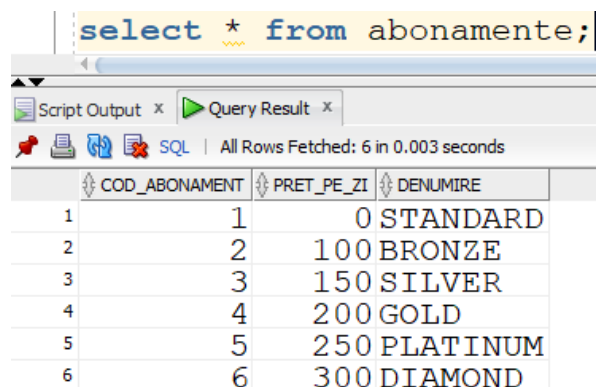
```
insert into facilitati values(ct_cod_facilitate.nextval,500000,'PISCINA_INT');
```

```

insert into facilitati values(ct_cod_facilitate.nextval,500000,'PISCINA_EXT');
insert into facilitati values(ct_cod_facilitate.nextval,200000,'SPA');
insert into facilitati values(ct_cod_facilitate.nextval,300000,'SALA_FITNESS');
insert into facilitati values(ct_cod_facilitate.nextval,700000,'CINEMATOGRAF');
insert into facilitati values(ct_cod_facilitate.nextval,600000,'RESTAURANT');
insert into facilitati values(ct_cod_facilitate.nextval,200000,'SAUNA');
insert into facilitati values(ct_cod_facilitate.nextval,1000000,'PARCARE');
insert into facilitati values(ct_cod_facilitate.nextval,400000,'SALA_EVENTIMENTE');

```

ABONAMENTE:



	COD_ABONAMENT	PRET_PE_ZI	DENUMIRE
1	1	0	STANDARD
2	2	100	BRONZE
3	3	150	SILVER
4	4	200	GOLD
5	5	250	PLATINUM
6	6	300	DIAMOND

```

create table abonamente(
    cod_abonament number(3) constraint pk_cod_abonament primary key,
    pret_pe_zi number(4) constraint n_pret_pe_zi not null,
    denumire varchar(15) constraint n_denumire_abonamente not null unique);

```

```

insert into abonamente values(ct_cod_abonament.nextval,0,'STANDARD');
insert into abonamente values(ct_cod_abonament.nextval,100,'BRONZE');
insert into abonamente values(ct_cod_abonament.nextval,150,'SILVER');
insert into abonamente values(ct_cod_abonament.nextval,200,'GOLD');
insert into abonamente values(ct_cod_abonament.nextval,250,'PLATINUM');
insert into abonamente values(ct_cod_abonament.nextval,300,'DIAMOND');

```

FACILITĂȚI_HOTELURI:

`select * from facilitati_hoteluri;`

Script Output x Query Result x

SQL All Rows Fetched: 30 in 0.004 seconds

	COD_HOTEL	COD_FACILITATE	NUMAR_LOCURI	ORA_DESCHIDERE	ORA_INCHIDERE
1	1	8	2000	00:00	23:59
2	2	8	2000	00:00	23:59
3	3	8	2000	00:00	23:59
4	4	8	2000	00:00	23:59
5	11	8	2000	00:00	23:59
6	12	8	2000	00:00	23:59
7	1	6	300	09:00	22:00
8	2	6	300	09:00	22:00
9	3	6	300	09:00	22:00
10	4	6	300	09:00	22:00
11	13	6	300	09:00	22:00
12	14	6	300	09:00	22:00
13	11	6	300	09:00	22:00
14	11	1	150	10:00	22:00
15	12	1	150	10:00	22:00
16	8	2	300	08:00	20:00
17	9	2	300	08:00	20:00
18	10	2	300	08:00	20:00
19	5	2	300	08:00	20:00
20	6	2	300	08:00	20:00
21	7	2	300	08:00	20:00
22	1	3	50	06:00	18:00
23	2	3	50	06:00	18:00
24	3	3	50	06:00	18:00
25	4	3	50	06:00	18:00
26	1	4	100	06:00	20:00
27	2	4	100	06:00	20:00
28	3	4	100	06:00	20:00
29	4	4	100	06:00	20:00
30	1	5	70	13:00	00:00

```

create table facilitati_hoteluri(
    cod_hotel number(4) constraint fk_cod_hotel
        references hoteluri(cod_hotel) on delete cascade,
    cod_facilitate number(4),
    numar_locuri number(6) constraint n_numar_locuri not null,
    ora_deschidere varchar(6) constraint n_ora_deschidere not null,
    ora_inchidere varchar(6) constraint n_ora_inchidere not null,
    constraint fk_cod_facilitate
        foreign key (cod_facilitate)
        references facilitati(cod_facilitate),
    constraint pk_facilitati_hoteluri primary key(cod_hotel, cod_facilitate));

```

--parcari

```

insert into facilitati_hoteluri values(1,8,2000,'00:00','23:59');
insert into facilitati_hoteluri values(2,8,2000,'00:00','23:59');
insert into facilitati_hoteluri values(3,8,2000,'00:00','23:59');
insert into facilitati_hoteluri values(4,8,2000,'00:00','23:59');
insert into facilitati_hoteluri values(11,8,2000,'00:00','23:59');
insert into facilitati_hoteluri values(12,8,2000,'00:00','23:59');

```

--retaurante

```

insert into facilitati_hoteluri values(1,6,300,'09:00','22:00');
insert into facilitati_hoteluri values(2,6,300,'09:00','22:00');
insert into facilitati_hoteluri values(3,6,300,'09:00','22:00');
insert into facilitati_hoteluri values(4,6,300,'09:00','22:00');

```

```

insert into facilitati_hoteluri values(13,6,300,'09:00','22:00');
insert into facilitati_hoteluri values(14,6,300,'09:00','22:00');
insert into facilitati_hoteluri values(11,6,300,'09:00','22:00');
--piscina_int
insert into facilitati_hoteluri values(11,1,150,'10:00','22:00');
insert into facilitati_hoteluri values(12,1,150,'10:00','22:00');
--piscina_ext
insert into facilitati_hoteluri values(8,2,300,'08:00','20:00');
insert into facilitati_hoteluri values(9,2,300,'08:00','20:00');
insert into facilitati_hoteluri values(10,2,300,'08:00','20:00');
insert into facilitati_hoteluri values(5,2,300,'08:00','20:00');
insert into facilitati_hoteluri values(6,2,300,'08:00','20:00');
insert into facilitati_hoteluri values(7,2,300,'08:00','20:00');
--spa
insert into facilitati_hoteluri values(1,3,50,'06:00','18:00');
insert into facilitati_hoteluri values(2,3,50,'06:00','18:00');
insert into facilitati_hoteluri values(3,3,50,'06:00','18:00');
insert into facilitati_hoteluri values(4,3,50,'06:00','18:00');
--sala_fitness
insert into facilitati_hoteluri values(1,4,100,'06:00','20:00');
insert into facilitati_hoteluri values(2,4,100,'06:00','20:00');
insert into facilitati_hoteluri values(3,4,100,'06:00','20:00');
insert into facilitati_hoteluri values(4,4,100,'06:00','20:00');
--cinematograf
insert into facilitati_hoteluri values(1,5,70,'13:00','00:00');

```

FACILITĂȚI_ABONAMENTE:

select * from facilitati_abonamente;

Script Output x Query Result x

All Rows Fetched: 25 in 0.003 seconds

	COD_FACILITATE	COD_ABONAMENT	NUMAR_INTRARI_ZI
1	8	1	5
2	8	2	5
3	6	2	3
4	8	3	5
5	6	3	3
6	1	3	1
7	2	3	1
8	8	4	5
9	6	4	4
10	1	4	2
11	2	4	2
12	8	5	10
13	6	5	6
14	1	5	4
15	2	5	4
16	3	5	1
17	4	5	1
18	5	5	1
19	8	6	10
20	6	6	8
21	1	6	6
22	2	6	6
23	3	6	2
24	4	6	2
25	5	6	2

```
create table facilitati_abonamente(
```



```

cod_facilitate number(4) constraint fk_cod_facilitate_fa
references facilitati(cod_facilitate),
--daca un abonament este scos de antreprenor din cadrul hotelurilor acestuia
--atunci si liniile aferente din tabela facilitati_abonamente trebuie sterse
cod_abonament number(3) constraint fk_cod_abonament_fa
references abonamente(cod_abonament) on delete cascade,
numar_intrari_zi number(3) constraint n_numar_intrari_zi not null,
constraint pk_facilitati_abonamente
primary key(cod_facilitate, cod_abonament));

--standard
insert into facilitati_abonamente values(8,1,5);
--bronze
insert into facilitati_abonamente values(8,2,5);
insert into facilitati_abonamente values(6,2,3);
--silver
insert into facilitati_abonamente values(8,3,5);
insert into facilitati_abonamente values(6,3,3);
insert into facilitati_abonamente values(1,3,1);
insert into facilitati_abonamente values(2,3,1);
--gold
insert into facilitati_abonamente values(8,4,5);
insert into facilitati_abonamente values(6,4,4);
insert into facilitati_abonamente values(1,4,2);
insert into facilitati_abonamente values(2,4,2);
--platinum
insert into facilitati_abonamente values(8,5,10);
insert into facilitati_abonamente values(6,5,6);
insert into facilitati_abonamente values(1,5,4);
insert into facilitati_abonamente values(2,5,4);
insert into facilitati_abonamente values(3,5,1);
insert into facilitati_abonamente values(4,5,1);
insert into facilitati_abonamente values(5,5,1);
--diamond
insert into facilitati_abonamente values(8,6,10);
insert into facilitati_abonamente values(6,6,8);
insert into facilitati_abonamente values(1,6,6);
insert into facilitati_abonamente values(2,6,6);
insert into facilitati_abonamente values(3,6,2);
insert into facilitati_abonamente values(4,6,2);
insert into facilitati_abonamente values(5,6,2);

```

ANGAJAȚI:

`select * from angajati;`

Script Output x Query Result x Query Result 1 x Query Result 2 x

All Rows Fetched: 28 in 0.006 seconds

	COD_ANGAJAT	COD_DEPARTAMENT	COD_HOTEL	NUME	PRENUME	SALARIU	DATA_ANGAJARE
1	1	1	1	Popescu	Monica	3840	12-AUG-14
2	2	2	2	Popa	Ilinca	2400	09-MAR-19
3	3	4	3	Diaconu	Laurentiu	4200	02-JAN-17
4	4	5	4	Chirila	Matei	3960	23-MAY-13
5	5	6	5	Caliniuc	Gelu	3000	11-JUN-18
6	6	7	11	Unquireanu	Darius	7000	14-DEC-17
7	7	8	6	Bontea	Sorin	2400	03-NOV-20
8	8	9	7	Popa	Cristian	3000	17-FEB-19
9	9	1	8	Lunqu	Florin	3200	04-OCT-18
10	10	2	9	Griqorescu	Claudia	3600	01-APR-12
11	11	4	10	Sorescu	Bianca	4600	28-JUL-20
12	12	5	12	Liteanu	Sergiu	3720	08-SEP-16
13	13	6	13	Obreja	Catalin	3500	16-AUG-17
14	14	7	1	Bejenaru	Ioana	6500	06-NOV-18
15	15	8	14	Dumitru	Carmen	3800	20-JAN-17
16	16	9	2	Constantinescu	Boqdan	2700	27-APR-17
17	17	1	3	Dobrescu	Roxana	3000	05-OCT-16
18	18	2	4	Miculescu	Georgiana	2400	22-JUN-20
19	19	4	5	Constandache	Alexia	4000	05-JUL-19
20	20	5	6	Acatrinei	Horia	3200	14-MAR-21
21	21	6	7	Mircea	Petrisor	3700	07-FEB-21
22	22	7	8	Mircea	Draqos	5500	05-MAY-17
23	23	8	9	Ouatu	Lidia	4200	22-JUN-22
24	24	9	10	Lefter	Razvan	2900	10-DEC-19
25	25	1	11	Olaru	Denisa	3100	09-JUL-18
26	26	2	12	Ciurescu	Crina	2600	01-MAR-17
27	27	4	13	Unquireanu	Olimpia	3700	16-AUG-19
28	28	6	14	Chiratcu	Andrei	3000	17-APR-22

create table angajati(

cod_angajat number(5) constraint pk_cod_angajat primary key,

cod_departament number(3) constraint n_cod_departament not null,

cod_hotel number(4) constraint n_cod_hotel not null,

nume varchar(15) constraint n_nume not null,

prenume varchar(20) constraint n_prenume not null,

salariu number(7) constraint n_salariu not null,

data_angajare date constraint n_data_angajare not null,

constraint fk_departament

foreign key (cod_departament)

references departamente(cod_departament) on delete cascade,

constraint fk_hotel

foreign key (cod_hotel)

references hoteluri(cod_hotel) on delete cascade);

insert into angajati

values(ct_cod_angajat.nextval,1,1,'Popescu','Monica',3200,to_date('12-august-2014'));

insert into angajati

values(ct_cod_angajat.nextval,2,2,'Popa','Ilinca',2400,to_date('9-march-2019'));

```

insert into angajati
values(ct_cod_angajat.nextval,4,3,'Diaconu','Laurentiu',4200,to_date('2-january-2017'));
insert into angajati
values(ct_cod_angajat.nextval,5,4,'Chirila','Matei',3300,to_date('23-may-2013'));
insert into angajati
values(ct_cod_angajat.nextval,6,5,'Caliniuc','Gelu',3000,to_date('11-june-2018'));
insert into angajati
values(ct_cod_angajat.nextval,7,11,'Ungureanu','Darius',7000,to_date('14-december-2017'));
insert into angajati
values(ct_cod_angajat.nextval,8,6,'Bontea','Sorin',2400,to_date('3-november-2020'));
insert into angajati
values(ct_cod_angajat.nextval,9,7,'Popa','Cristian',3000,to_date('17-february-2019'));
insert into angajati
values(ct_cod_angajat.nextval,1,8,'Lungu','Florin',3200,to_date('4-october-2018'));
insert into angajati
values(ct_cod_angajat.nextval,2,9,'Grigorescu','Claudia',3000,to_date('1-april-2012'));
insert into angajati
values(ct_cod_angajat.nextval,4,10,'Sorescu','Bianca',4600,to_date('28-july-2020'));
insert into angajati
values(ct_cod_angajat.nextval,5,12,'Liteanu','Sergiu',3100,to_date('8-september-2016'));
insert into angajati
values(ct_cod_angajat.nextval,6,13,'Obreja','Catalin',3500,to_date('16-august-2017'));
insert into angajati
values(ct_cod_angajat.nextval,7,1,'Bejenaru','Ioana',6500,to_date('6-november-2018'));
insert into angajati
values(ct_cod_angajat.nextval,8,14,'Dumitru','Carmen',3800,to_date('20-january-2017'));
insert into angajati
values(ct_cod_angajat.nextval,9,2,'Constantinescu','Bogdan',2700,to_date('27-april-2017'));
insert into angajati
values(ct_cod_angajat.nextval,1,3,'Dobrescu','Roxana',2500,to_date('5-october-2016'));
insert into angajati
values(ct_cod_angajat.nextval,2,4,'Miculescu','Georgiana',2400,to_date('22-june-2020'));
insert into angajati
values(ct_cod_angajat.nextval,4,5,'Constandache','Alexia',4000,to_date('5-july-2019'));
insert into angajati
values(ct_cod_angajat.nextval,5,6,'Acatrinei','Horia',3200,to_date('14-march-2021'));
insert into angajati
values(ct_cod_angajat.nextval,6,7,'Mircea','Petrisor',3700,to_date('7-february-2021'));
insert into angajati
values(ct_cod_angajat.nextval,7,8,'Mircea','Dragos',5500,to_date('5-may-2017'));
insert into angajati
values(ct_cod_angajat.nextval,8,9,'Ouatu','Lidia',4200,to_date('22-june-2022'));
insert into angajati
values(ct_cod_angajat.nextval,9,10,'Lefter','Razvan',2900,to_date('10-december-2019'));

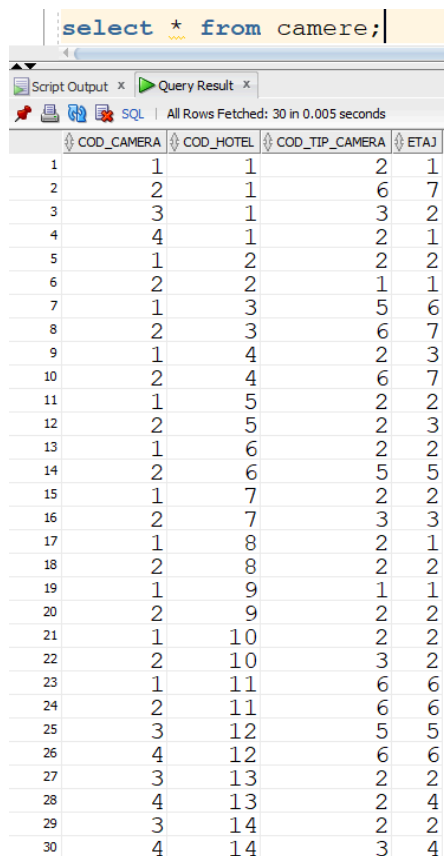
```

```

insert into angajati
values(ct_cod_angajat.nextval,1,11,'Olaru','Denisa',3100,to_date('9-july-2018'));
insert into angajati
values(ct_cod_angajat.nextval,2,12,'Ciurescu','Crina',2600,to_date('1-march-2017'));
insert into angajati
values(ct_cod_angajat.nextval,4,13,'Ungureanu','Olimpia',3700,to_date('16-august-2019'));
insert into angajati
values(ct_cod_angajat.nextval,6,14,'Chiratcu','Andrei',3000,to_date('17-april-2022'));

```

CAMERE:



	COD_CAMERA	COD_HOTEL	COD_TIP_CAMERA	ETAJ
1	1	1	2	1
2	2	1	6	7
3	3	1	3	2
4	4	1	2	1
5	1	2	2	2
6	2	2	1	1
7	1	3	5	6
8	2	3	6	7
9	1	4	2	3
10	2	4	6	7
11	1	5	2	2
12	2	5	2	3
13	1	6	2	2
14	2	6	5	5
15	1	7	2	2
16	2	7	3	3
17	1	8	2	1
18	2	8	2	2
19	1	9	1	1
20	2	9	2	2
21	1	10	2	2
22	2	10	3	2
23	1	11	6	6
24	2	11	6	6
25	3	12	5	5
26	4	12	6	6
27	3	13	2	2
28	4	13	2	4
29	3	14	2	2
30	4	14	3	4

```

create table camere(
  cod_camera number(4),
  cod_hotel number(4),
  cod_tip_camera number(2) constraint n_cod_tip_camera not null,
  etaj number(2) constraint n_etaj not null,
  constraint fk_cod_tip_camera
    foreign key (cod_tip_camera)
    references tipuri_camera(cod_tip_camera),
--daca se intampla ceva cu un hotel, se intampla automat si cu camerele acestuia
  constraint fk_cod_hotel_camere
    foreign key (cod_hotel)
    references hoteluri(cod_hotel) on delete cascade,

```

```
constraint pk_camere  
primary key(cod_camera, cod_hotel));
```

```
insert into camere values(ct_cod_camera_hotel_1.nextval,1,2,1);  
insert into camere values(ct_cod_camera_hotel_1.nextval,1,6,7);  
insert into camere values(ct_cod_camera_hotel_1.nextval,1,3,2);  
insert into camere values(ct_cod_camera_hotel_1.nextval,1,2,1);  
insert into camere values(ct_cod_camera_hotel_2.nextval,2,2,2);  
insert into camere values(ct_cod_camera_hotel_2.nextval,2,1,1);  
insert into camere values(ct_cod_camera_hotel_3.nextval,3,5,6);  
insert into camere values(ct_cod_camera_hotel_3.nextval,3,6,7);  
insert into camere values(ct_cod_camera_hotel_4.nextval,4,2,3);  
insert into camere values(ct_cod_camera_hotel_4.nextval,4,6,7);  
insert into camere values(ct_cod_camera_hotel_5.nextval,5,2,2);  
insert into camere values(ct_cod_camera_hotel_5.nextval,5,2,3);  
insert into camere values(ct_cod_camera_hotel_6.nextval,6,2,2);  
insert into camere values(ct_cod_camera_hotel_6.nextval,6,5,5);  
insert into camere values(ct_cod_camera_hotel_7.nextval,7,2,2);  
insert into camere values(ct_cod_camera_hotel_7.nextval,7,3,3);  
insert into camere values(ct_cod_camera_hotel_8.nextval,8,2,1);  
insert into camere values(ct_cod_camera_hotel_8.nextval,8,2,2);  
insert into camere values(ct_cod_camera_hotel_9.nextval,9,1,1);  
insert into camere values(ct_cod_camera_hotel_9.nextval,9,2,2);  
insert into camere values(ct_cod_camera_hotel_10.nextval,10,2,2);  
insert into camere values(ct_cod_camera_hotel_10.nextval,10,3,2);  
insert into camere values(ct_cod_camera_hotel_11.nextval,11,6,6);  
insert into camere values(ct_cod_camera_hotel_11.nextval,11,6,6);  
insert into camere values(ct_cod_camera_hotel_12.nextval,12,5,5);  
insert into camere values(ct_cod_camera_hotel_12.nextval,12,6,6);  
insert into camere values(ct_cod_camera_hotel_13.nextval,13,2,2);  
insert into camere values(ct_cod_camera_hotel_13.nextval,13,2,4);  
insert into camere values(ct_cod_camera_hotel_14.nextval,14,2,2);  
insert into camere values(ct_cod_camera_hotel_14.nextval,14,3,4);
```

CLIENTI:

`select * from clienti;`

Script Output x Query Result x

SQL | All Rows Fetched: 18 in 0.001 seconds

	COD_CLIENT	COD_ABONAMENT	NUME	PRENUME	DATA_NASTERE	NUMAR_TELEFON
1	1	1	Ionescu	Maria	11-MAY-98	0741123456
2	2	2	Vasilescu	Alexandru	08-JUN-90	0722654321
3	3	3	Popa	Elena	28-NOV-93	0765987654
4	4	4	Teodorescu	Raluca	06-AUG-87	0733456789
5	5	5	Dumitrescu	Ana	13-APR-03	0788321654
6	6	6	Marin	Andrei	10-SEP-00	0799112233
7	7	1	Georgescu	Cristina	14-JUN-97	0744556677
8	8	2	Iliescu	Gabriel	22-FEB-91	0712345678
9	9	3	Preda	Ioana	09-AUG-01	0755123789
10	10	4	Mihailescu	Radu	03-OCT-98	0766987123
11	11	5	Paunescu	Laura	11-JUL-96	0723654987
12	12	6	Rusu	Vasile	04-DEC-93	0745123789
13	13	1	Florescu	Diana	07-JAN-92	0734456123
14	14	2	Enache	George	02-MAR-78	0789321456
15	15	3	Anghel	Carmen	27-JUN-77	0764777888
16	16	2	Dumitru	Daniela	15-SEP-92	0743111222
17	17	3	Teodor	Carmen	27-JUN-77	0798654321
18	18	3	Sandu	Florin	14-JUL-81	0721333444

```

create table clienti(
  cod_client number(15) constraint pk_cod_client primary key,
  cod_abonament number(3) constraint n_cod_abonament not null,
  nume varchar(15) constraint n_nume_clienti not null,
  prenume varchar(20) constraint n_prenume_clienti not null,
  data_nastere date constraint n_data_nastere not null,
  numar_telefon varchar(12) constraint n_numar_telefon not null unique,
  constraint fk_cod_abonament
    foreign key (cod_abonament)
    references abonamente(cod_abonament) on delete set null);

insert into clienti
values(ct_cod_client.nextval,1,'Ionescu','Maria',to_date('11-may-1998'),'0741123456');
insert into clienti
values(ct_cod_client.nextval,2,'Vasilescu','Alexandru',to_date('8-june-1990'),'0722654321');
insert into clienti
values(ct_cod_client.nextval,3,'Popa','Elena',to_date('28-november-1993'),'0765987654');
insert into clienti
values(ct_cod_client.nextval,4,'Teodorescu','Raluca',to_date('6-maugust-1987'),'0733456789')
;
insert into clienti
values(ct_cod_client.nextval,5,'Dumitrescu','Ana',to_date('13-april-2003'),'0788321654');
insert into clienti
values(ct_cod_client.nextval,6,'Marin','Andrei',to_date('10-september-2000'),'0799112233');
insert into clienti
values(ct_cod_client.nextval,1,'Georgescu','Cristina',to_date('14-june-1997'),'0744556677');
insert into clienti
values(ct_cod_client.nextval,2,'Iliescu','Gabriel',to_date('22-february-1991'),'0712345678');
insert into clienti
values(ct_cod_client.nextval,3,'Preda','Ioana',to_date('9-august-2001'),'0755123789');

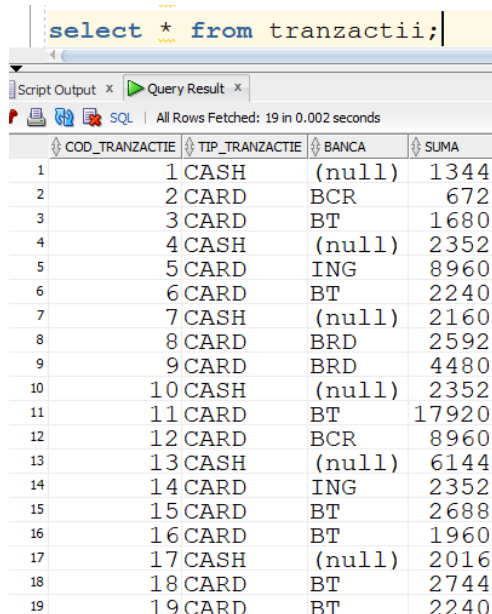
```

```

insert into clienti
values(ct_cod_client.nextval,4,'Mihailescu','Radu',to_date('3-october-1998'),'0766987123');
insert into clienti
values(ct_cod_client.nextval,5,'Paunescu','Laura',to_date('11-july-1996'),'0723654987');
insert into clienti
values(ct_cod_client.nextval,6,'Rusu','Vasile',to_date('4-december-1993'),'0745123789');
insert into clienti
values(ct_cod_client.nextval,1,'Florescu','Diana',to_date('7-january-1992'),'0734456123');
insert into clienti
values(ct_cod_client.nextval,2,'Enache','George',to_date('2-march-1978'),'0789321456');
insert into clienti
values(ct_cod_client.nextval,3,'Anghel','Carmen',to_date('27-june-1977'),'0764777888');
insert into clienti
values(ct_cod_client.nextval,2,'Dumitru','Daniela',to_date('15-september-1992'),'0743111222)
;
insert into clienti
values(ct_cod_client.nextval,3,'Teodor','Carmen',to_date('27-june-1977'),'0798654321');
insert into clienti
values(ct_cod_client.nextval,3,'Sandu','Florin',to_date('14-july-1981'),'0721333444');

```

TRANZACȚII:



	COD_TRANZACTIE	TIP_TRANZACTIE	BANCA	SUMA
1	1	CASH	(null)	1344
2	2	CARD	BCR	672
3	3	CARD	BT	1680
4	4	CASH	(null)	2352
5	5	CARD	ING	8960
6	6	CARD	BT	2240
7	7	CASH	(null)	2160
8	8	CARD	BRD	2592
9	9	CARD	BRD	4480
10	10	CASH	(null)	2352
11	11	CARD	BT	17920
12	12	CARD	BCR	8960
13	13	CASH	(null)	6144
14	14	CARD	ING	2352
15	15	CARD	BT	2688
16	16	CARD	BT	1960
17	17	CASH	(null)	2016
18	18	CARD	BT	2744
19	19	CARD	BT	2240

```

create table tranzactii(
    cod_tranzactie number(15) constraint pk_cod_tranzactie primary key,
    tip_tranzactie varchar(5) constraint n_tip_tranzactie not null,
    banca varchar(15),
    suma number(10) constraint n_suma not null);

```

```

insert into tranzactii values(ct_cod_tranzactie.nextval,'CASH',NULL,1344);

```

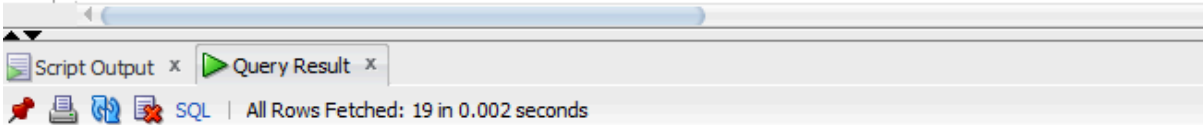
```

insert into tranzactii values(ct_cod_tranzactie.nextval,'CARD','BCR',672);
insert into tranzactii values(ct_cod_tranzactie.nextval,'CARD','BT',1680);
insert into tranzactii values(ct_cod_tranzactie.nextval,'CASH',NULL,2352);
insert into tranzactii values(ct_cod_tranzactie.nextval,'CARD','ING',8960);
insert into tranzactii values(ct_cod_tranzactie.nextval,'CARD','BT',2240);
insert into tranzactii values(ct_cod_tranzactie.nextval,'CASH',NULL,2160);
insert into tranzactii values(ct_cod_tranzactie.nextval,'CARD','BRD',2592);
insert into tranzactii values(ct_cod_tranzactie.nextval,'CARD','BRD',4480);
insert into tranzactii values(ct_cod_tranzactie.nextval,'CASH',NULL,2352);
insert into tranzactii values(ct_cod_tranzactie.nextval,'CARD','BT',17920);
insert into tranzactii values(ct_cod_tranzactie.nextval,'CARD','BCR',8960);
insert into tranzactii values(ct_cod_tranzactie.nextval,'CASH',NULL,6144);
insert into tranzactii values(ct_cod_tranzactie.nextval,'CARD','ING',2352);
insert into tranzactii values(ct_cod_tranzactie.nextval,'CARD','BT',2688);
insert into tranzactii values(ct_cod_tranzactie.nextval,'CARD','BT',1960);
insert into tranzactii values(ct_cod_tranzactie.nextval,'CASH',NULL,2016);
insert into tranzactii values(ct_cod_tranzactie.nextval,'CARD','BT',2744);
insert into tranzactii values(ct_cod_tranzactie.nextval,'CARD','BT',2240);

```

REZERVĂRI:

`select * from rezervari;`



	COD_CAMERA	COD_HOTEL	COD_CLIENT	COD_TRANZACTIE	DATA_VENIRE	DATA_PLECAR
1	1	8	1	1	18-JUN-24	22-JUN-24
2	1	13	2	2	01-JUL-24	03-JUL-24
3	2	9	3	3	24-AUG-24	29-AUG-24
4	1	5	4	4	15-JUN-23	21-JUN-23
5	2	1	5	5	22-JUN-24	26-JUN-24
6	1	2	6	6	08-AUG-24	13-AUG-24
7	2	14	7	7	16-FEB-25	21-FEB-25
8	2	10	8	8	21-JUN-24	27-JUN-24
9	2	6	9	9	05-SEP-24	10-SEP-24
10	1	7	10	10	02-SEP-24	08-SEP-24
11	2	12	11	11	10-NOV-24	18-NOV-24
12	1	11	12	12	13-MAR-24	17-MAR-24
13	1	3	13	13	18-JUL-24	24-JUL-24
14	1	8	14	14	16-JUL-24	23-JUL-24
15	1	4	15	15	23-DEC-23	29-DEC-23
16	1	10	17	17	22-JUN-24	28-JUN-24
17	1	6	18	18	03-JUL-24	10-JUL-24
18	1	2	6	19	15-SEP-23	20-SEP-23
19	1	2	16	16	01-JAN-23	06-JAN-23


```

create table rezervari(
    cod_camera number(4),
    cod_hotel number(4),
    cod_client number(15),
    cod_tranzactie number(15),
    data_venire date,
    data_plecare date,
    constraint fk_cod_client
        foreign key(cod_client)
        references clienti(cod_client) on delete cascade,
    constraint fk_camere
        foreign key (cod_camera, cod_hotel)
        references camere(cod_camera, cod_hotel) on delete cascade,
    constraint fk_cod_tranzactie
        foreign key (cod_tranzactie)
        references tranzactii(cod_tranzactie) on delete cascade,
    constraint pk_rezervari
        primary key(cod_camera, cod_hotel, cod_client, cod_tranzactie, data_venire),
    unique (cod_camera, cod_hotel, data_venire));

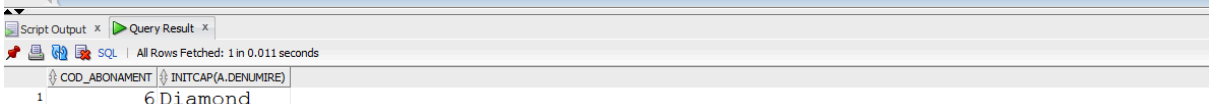
insert into rezervari values(1,8,1,1,to_date('18-june-2024'),to_date('22-june-2024'));
insert into rezervari values(1,13,2,2,to_date('1-july-2024'),to_date('3-july-2024'));
insert into rezervari values(2,9,3,3,to_date('24-august-2024'),to_date('29-august-2024'));
insert into rezervari values(1,5,4,4,to_date('15-june-2023'),to_date('21-june-2023'));
insert into rezervari values(2,1,5,5,to_date('22-june-2024'),to_date('26-june-2024'));
insert into rezervari values(1,2,6,6,to_date('8-august-2024'),to_date('13-august-2024'));
insert into rezervari values(2,14,7,7,to_date('16-february-2025'),to_date('21-february-2025'));
insert into rezervari values(2,10,8,8,to_date('21-june-2024'),to_date('27-june-2024'));
insert into rezervari
values(2,6,9,9,to_date('5-september-2024'),to_date('10-september-2024'));
insert into rezervari
values(1,7,10,10,to_date('2-september-2024'),to_date('8-september-2024'));
insert into rezervari
values(2,12,11,11,to_date('10-november-2024'),to_date('18-november-2024'));
insert into rezervari values(1,11,12,12,to_date('13-march-2024'),to_date('17-march-2024'));
insert into rezervari values(1,3,13,13,to_date('18-july-2024'),to_date('24-july-2024'));
insert into rezervari values(1,8,14,14,to_date('16-july-2024'),to_date('23-july-2024'));
insert into rezervari
values(1,4,15,15,to_date('23-december-2023'),to_date('29-december-2023'));
insert into rezervari values(1,10,17,17,to_date('22-june-2024'),to_date('28-june-2024'));
insert into rezervari values(1,6,18,18,to_date('3-july-2024'),to_date('10-july-2024'));
insert into rezervari
values(1,2,6,19,to_date('15-september-2023'),to_date('20-september-2023'));
insert into rezervari values(1,2,16,16,to_date('1-january-2023'),to_date('6-january-2023'));

```

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

➔ Afișați codul și denumirea (cu prima literă mare și restul mici) abonamentelor care au fost cel mai folosite de clienți în rezervările încheiate (care nu sunt în desfășurare sau care nu urmează).

```
--pasul 1:construiesc o tabela in care am doar rezervariile incheiate
with aux as(select c.cod_abonament
              from rezervari r, clienti c
              where r.cod_client = c.cod_client and
                    r.data_plecure < sysdate),
--pasul 2:construiesc o alta tabela in care am codul abonamentelor si numarul lor total
aux_2 as(select count(*) as numar_abonamente, cod_abonament
          from aux
          group by cod_abonament)
--pasul 3: trebuie sa afisez toate abonamentele care sunt maxime
select aa.cod_abonament, initcap(a.denumire)
from aux_2 aa, abonamente a
where aa.cod_abonament = a.cod_abonament and
      aa.numar_abonamente = (select max(numar_abonamente)
                             from aux_2);
```

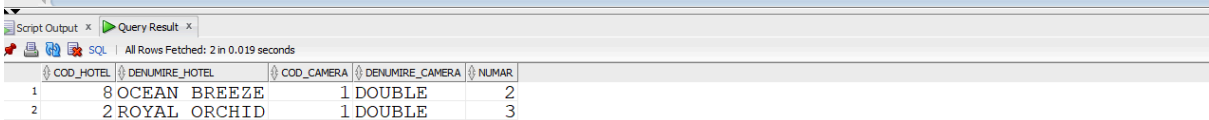


```
with aux as(select c.cod_abonament
              from rezervari r, clienti c
              where r.cod_client = c.cod_client and
                    r.data_plecure < sysdate),
aux_2 as(select count(*) as numar_abonamente, cod_abonament
          from aux
          group by cod_abonament)
select aa.cod_abonament, initcap(a.denumire)
from aux_2 aa, abonamente a
where aa.cod_abonament = a.cod_abonament and
      aa.numar_abonamente = (select max(numar_abonamente)
                             from aux_2);
```

UTILIZATE: o funcție pe șiruri de caractere, clauza with

- ➔ Pentru acele camere care au fost rezervate de mai multe ori decât minimul rezervărilor unei camere, să se afișeze codul și denumirea hotelului, codul și denumirea camerei și numărul de rezervari făcute pe respectiva cameră. Camerele nu trebuie să aparțină aceluiași hotel pentru a li se compara rezervările. Rezultatele vor fi ordonate după numărul de rezervări.

```
--Pas 1: fac o tabela care contine informatii pe care vreau sa le afisez
with aux as (
select h.cod_hotel, h.denumire as denumire_hotel, c.cod_camera, tc.denumire as denumire_camera
from hoteluri h, camere c, tipuri_camera tc
where h.cod_hotel = c.cod_hotel and
c.cod_tip_camera = tc.cod_tip_camera)
--Pas 2: aflu numarul de rezervari pentru fiecare camera
select a.cod_hotel, a.denumire_hotel, a.cod_camera, a.denumire_camera, numar
from aux a, (select count(*) numar, cod_camera as cm, cod_hotel as ch
from rezervari
group by cod_camera, cod_hotel
having count(*) > (select min(count(*))
from rezervari
group by cod_camera, cod_hotel)) aux_2
where a.cod_hotel = ch and a.cod_camera = cm
order by numar;
```



	COD_HOTEL	DENUMIRE_HOTEL	COD_CAMERA	DENUMIRE_CAMERA	NUMAR
1	8	OCEAN BREEZE	1	DOUBLE	2
2	2	ROYAL ORCHID	1	DOUBLE	3

```
with aux as(
select h.cod_hotel, h.denumire as denumire_hotel, c.cod_camera, tc.denumire as
denumire_camera
from hoteluri h, camere c, tipuri_camera tc
where h.cod_hotel = c.cod_hotel and
c.cod_tip_camera = tc.cod_tip_camera)

select a.cod_hotel, a.denumire_hotel, a.cod_camera, a.denumire_camera, numar
from aux a, (select count(*) numar, cod_camera as cm, cod_hotel as ch
from rezervari
group by cod_camera, cod_hotel
having count(*) > (select min(count(*))
from rezervari
group by cod_camera, cod_hotel)) aux_2
where a.cod_hotel = ch and a.cod_camera = cm
order by numar;
```

UTILIZATE: subcerere nesincronizată în clauza FROM, grupări de date, funcții grup, filtrare la nivel de grupuri cu subcereri nesincronizate (în clauza de HAVING) în care intervin cel puțin 3 tabele (în cadrul aceleiași cereri)

- Să se afișeze următoarele informații referitoare la rezervări: codul camerei, codul hotelului, codul clientului, data venirii clientului și cea a plecării, suma plătită pentru rezervare, banca prin care a fost efectuată tranzacția pentru rezervare (în cazul în care tranzacția a fost efectuată cash se va afișa mesajul “Nicio bancă - CASH”) și tipul rezervării (“ÎNCHEIATĂ”, “ÎN DESFĂȘURARE”, “VIITOARE”). Rezultatele vor fi ordonate după tipul rezervării.

```

select cod_camera, cod_hotel, cod_client, data_venire, data_plecare, suma,
decode(nvl(banca,'null'),'null','Nicio banca - cash','Tranzactie prin ' || banca) as banca,
case
    when data_venire <= sysdate and sysdate < data_plecare then 'IN DESFASURARE'
    when data_plecare < sysdate then 'INCHEIATA'
    else 'VIITOARE'
end as tip_rezervare
from rezervari r, tranzactii t
where r.cod_tranzactie = t.cod_tranzactie
order by tip_rezervare;

```

	COD_CAMERA	COD_HOTEL	COD_CLIENT	DATA_VENIRE	DATA_PLECARE	SUMA	BANCA	TIP_REZERVARE
1	1	2	615	SEP-23	20-SEP-23	2240	Tranzactie prin BT	INCHEIATA
2	1	5	415	JUN-23	21-JUN-23	2352	Nicio banca - cash	INCHEIATA
3	1	11	1213	MAR-24	17-MAR-24	8960	Tranzactie prin BCR	INCHEIATA
4	1	4	1523	DEC-23	29-DEC-23	2688	Tranzactie prin BT	INCHEIATA
5	1	2	1601	JAN-23	06-JAN-23	1960	Tranzactie prin BT	INCHEIATA
6	2	10	821	JUN-24	27-JUN-24	2592	Tranzactie prin BRD	VIITOARE
7	2	6	905	SEP-24	10-SEP-24	4480	Tranzactie prin BRD	VIITOARE
8	1	7	1002	SEP-24	08-SEP-24	2352	Nicio banca - cash	VIITOARE
9	2	12	1110	NOV-24	18-NOV-24	17920	Tranzactie prin BT	VIITOARE
10	1	3	1318	JUL-24	24-JUL-24	6144	Nicio banca - cash	VIITOARE
11	1	8	1416	JUL-24	23-JUL-24	2352	Tranzactie prin ING	VIITOARE
12	1	10	1722	JUN-24	28-JUN-24	2016	Nicio banca - cash	VIITOARE
13	1	6	1803	JUL-24	10-JUL-24	2744	Tranzactie prin BT	VIITOARE
14	2	14	716	FEB-25	21-FEB-25	2160	Nicio banca - cash	VIITOARE
15	1	2	608	AUG-24	13-AUG-24	2240	Tranzactie prin BT	VIITOARE
16	2	1	522	JUN-24	26-JUN-24	8960	Tranzactie prin ING	VIITOARE
17	2	9	324	AUG-24	29-AUG-24	1680	Tranzactie prin BT	VIITOARE
18	1	8	118	JUN-24	22-JUN-24	1344	Nicio banca - cash	VIITOARE
19	1	13	201	JUL-24	03-JUL-24	672	Tranzactie prin BCR	VIITOARE

```

select cod_camera, cod_hotel, cod_client, data_venire, data_plecare, suma,
decode(nvl(banca,'null'),'null','Nicio banca - cash','Tranzactie prin ' || banca) as banca,
case
    when data_venire <= sysdate and sysdate < data_plecare then 'IN DESFASURARE'
    when data_plecare < sysdate then 'INCHEIATA'
    else 'VIITOARE'
end as tip_rezervare
from rezervari r, tranzactii t
where r.cod_tranzactie = t.cod_tranzactie
order by tip_rezervare;

```

UTILIZATE: o funcție pentru date calendaristice, NVL, DECODE, ordonare

- Pentru fiecare client să se afișeze suma totală pe care a cheltuit-o pe rezervări în cadrul tuturor hotelurilor antreprenorului.

```

select nume, prenume, (select sum(suma)
                        from tranzactii t, rezervari r

```

where c.cod_client = r.cod_client and r.cod_tranzactie = t.cod_tranzactie) as
suma_totala
from clienti c
order by 1,2;

```

select nume, prenume, (select sum(suma)
                        from tranzactii t, rezervari r
                        where c.cod_client = r.cod_client and r.cod_tranzactie = t.cod_tranzactie) as suma_totala
from clienti c
order by 1,2;

```

	NUME	PRENUME	SUMA_TOTALA
1	Anghel	Carmen	2688
2	Dumitrescu	Ana	8960
3	Dumitru	Daniela	1960
4	Enache	George	2352
5	Florescu	Diana	6144
6	Georgescu	Cristina	2160
7	Iliescu	Gabriel	2592
8	Ionescu	Maria	1344
9	Marin	Andrei	4480
10	Mihailescu	Radu	2352
11	Paunescu	Laura	17920
12	Popa	Elena	1680
13	Preda	Ioana	4480
14	Rusu	Vasile	8960
15	Sandu	Florin	2744
16	Teodor	Carmen	2016
17	Teodorescu	Raluca	2352
18	Vasilescu	Alexandru	672

UTILIZATE: subcerere sincronizata în care intervin cel puțin 3 tabele

- ➔ Să se afișeze numele și prenumele (cu majuscule și într-o singură coloană denumită 'nume_complet') clienților care au fost sau vor fi cazați măcar o dată la un hotel de 4 sau 5 stele până la terminarea anului 2024. Clienții trebuie să aibă și un abonament de tip 5 sau 6.

```

with hoteluri_stele as (select cod_hotel, denumire, numar_stele
                        from hoteluri
                        where numar_stele = 4 or numar_stele = 5)
select distinct concat(concat(upper(nume), ' '), upper(prenume)) as nume_complet
from clienti c, rezervari r
where c.cod_client = r.cod_client and (c.cod_abonament = 5 or c.cod_abonament = 6) and
data_venire < to_date('31-december-2024')
and exists (select 1
            from hoteluri_stele
            where cod_hotel = r.cod_hotel);

```

	NUME_COMPLET
1	MARIN ANDREI
2	RUSU VASILE
3	DUMITRESCU ANA
4	PAUNESCU LAURA

```

with hoteluri_stele as (select cod_hotel, denumire, numar_stele
                        from hoteluri
                        where numar_stele = 4 or numar_stele = 5)
select distinct concat(concat(upper(nume), ' '), upper(prenume)) as nume_complet
from clienti c, rezervari r

```

```

where c.cod_client = r.cod_client and (c.cod_abonament = 5 or c.cod_abonament = 6) and
data_venire < to_date('31-december-2024')
and exists (select 1
            from hoteluri_stele
            where cod_hotel = r.cod_hotel);

```

UTILIZATE: funcție pentru date calendaristice, 2 funcții pentru șiruri de caractere, clauza WITH

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

→ Să se mărească salariile tuturor angajaților care au peste 8 ani vechime cu 1.2.

```

update angajati
set salariu = salariu * 1.2
where cod_angajat in (select a.cod_angajat
                      from angajati a
                      where to_number(to_char(sysdate,'yyyy'))
                        - to_number(to_char(data_angajare,'yyyy')) >= 8);

```

5 rows updated.

```

update angajati
set salariu = salariu * 1.2
where cod_angajat in (select a.cod_angajat
                      from angajati a
                      where to_number(to_char(sysdate,'yyyy'))
                        - to_number(to_char(data_angajare,'yyyy')) >= 8);

```

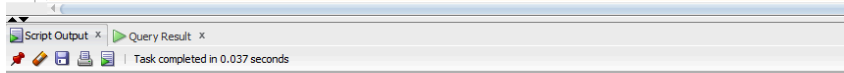
→ Să se șteargă din tabela ORASE toate orașele în care antreprenorul nu are niciun hotel.

```

insert into orase values(ct_cod_oras.nextval,'CRAIOVA',242000);
insert into orase values(ct_cod_oras.nextval,'GALATI',218000);

delete
from orase
where cod_oras not in(select distinct h.cod_oras
                       from hoteluri h);

```



Commit complete.

1 row inserted.

1 row inserted.

2 rows deleted.

(inserez întâi două linii noi în tabela ORASE pentru ca exercițiul să aiba un rezultat - “2 rows deleted”)

```

insert into orase values(ct_cod_oras.nextval,'CRAIOVA',242000);
insert into orase values(ct_cod_oras.nextval,'GALATI',218000);

```

```

delete
from orase
where cod_oras not in(select distinct h.cod_oras from hoteluri h);

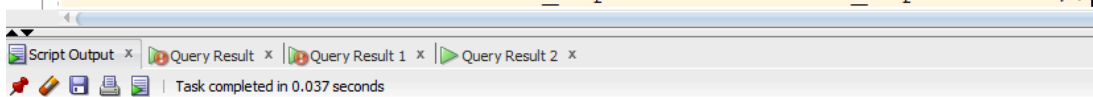
```

- ➔ Să se mărească cu 10% salariile tuturor angajaților care au salariul mai mic decât media tuturor salariilor din cadrul departamentului din care fac parte (se ține cont doar de departament nu și de hotelul din care fac parte)

```

update angajati
set salariu = salariu + 0.1 * salariu
where salariu < (select avg(a.salariu)
                 from angajati a
                 where a.cod_departament = cod_departament);

```



15 rows updated.

Rollback complete.

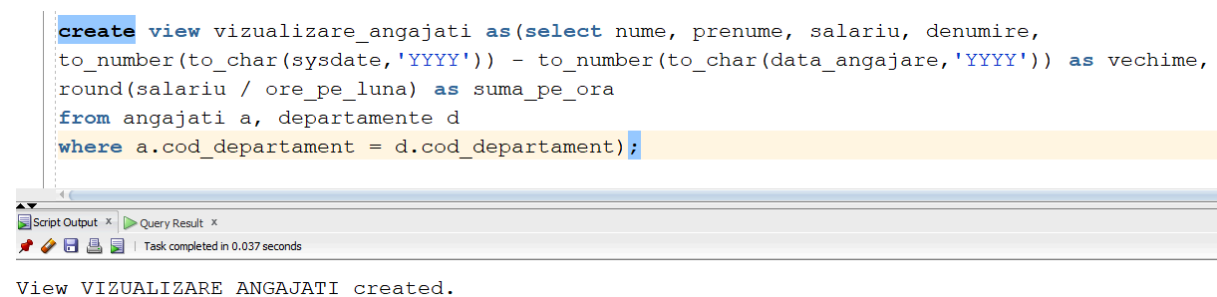
15 rows updated.

update angajati

```
set salariu = salariu + 0.1 * salariu
where salariu < (select avg(a.salariu)
                from angajati a
                where a.cod_departament = cod_departament);
```

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

Să se creeze o vizualizare care conține date despre toți angajații referitoare la numele și prenumele acestora, departamentul din care fac parte, vechimea lor și cât fac aceștia pe oră.



```
create view vizualizare_angajati as(select nume, prenume, salariu, denumire,
to_number(to_char(sysdate,'YYYY')) - to_number(to_char(data_angajare,'YYYY')) as
vechime,
round(salariu / ore_pe_luna) as suma_pe_ora
from angajati a, departamente d
where a.cod_departament = d.cod_departament);
```



```
select *
from vizualizare_angajati;
```

	NUME	PRENUME	SALARIU	DENUMIRE	VECHIME	SUMA_PE_ORA
1	Lunqu	Florin	3200	RECEPTIE	6	20
2	Popescu	Monica	3200	RECEPTIE	10	20
3	Olaru	Denisa	3100	RECEPTIE	6	19
4	Dobrescu	Roxana	125	RECEPTIE	8	1
5	Popa	Ilinca	120	CURATENIE	5	1
6	Ciurescu	Crina	130	CURATENIE	7	1
7	Miculescu	Georgiana	120	CURATENIE	4	1
8	Griqorescu	Claudia	3000	CURATENIE	12	19
9	Diaconu	Laurentiu	4200	CONTABILITATE	7	26
10	Unquireanu	Olimpia	3700	CONTABILITATE	5	23
11	Constandache	Alexia	4000	CONTABILITATE	5	25
12	Sorescu	Bianca	4600	CONTABILITATE	4	29
13	Acatrinei	Horia	160	MENTENANTA	3	1
14	Chirila	Matei	165	MENTENANTA	11	1
15	Liteanu	Sergiu	155	MENTENANTA	8	1
16	Mircea	Petrisor	3700	SECURITATE	3	21
17	Obreja	Catalin	3500	SECURITATE	7	19
18	Caliniuc	Gelu	150	SECURITATE	6	1
19	Chiratcu	Andrei	150	SECURITATE	2	1
20	Unquireanu	Darius	7000	IT	7	44
21	Mircea	Dragos	5500	IT	7	34
22	Bejenaru	Ioana	6500	IT	6	41
23	Ouatu	Lidia	4200	BUCATARIE	2	26
24	Dumitru	Carmen	3800	BUCATARIE	7	24
25	Bontea	Sorin	120	BUCATARIE	4	1
26	Lefter	Razvan	145	APROVIZIONARE	5	1
27	Constantinescu	Boqdan	135	APROVIZIONARE	7	1
28	Popa	Cristian	3000	APROVIZIONARE	5	19

Operație LMD permisă pe această vizualizare:

Să se mărească cu 5% salariul tuturor angajaților care primesc între 15 și 18 lei pe oră (inclusiv).

```
update vizualizare_angajati
set salariu = salariu * 0.05
where 15 <= suma_pe_ora and suma_pe_ora <= 18;
```

	Script Output	Query Result
	Task completed in 0.034 seconds	

12 rows updated.

```
update vizualizare_angajati
set salariu = salariu * 0.05
```

where 15 <= suma_pe_oras and suma_pe_oras <= 18;

Operație LDM nepermisă pe această vizualizare:

```
insert into vizualizare_angajati values('Grigorescu','Emilia',4500,'MARKETING',0,38);
```

Script Output x Query Result x

Task completed in 0.064 seconds

01776. 00000 - "cannot modify more than one base table through a join view"

*Cause: Columns belonging to more than one underlying table were either inserted into or updated.

*Action: Phrase the statement as two or more separate statements.

insert into vizualizare_angajati values('Grigorescu','Emilia',4500,'MARKETING',0,38);

15. Formulați în limbaj natural și implementați în SQL: o cerere ce utilizează operația outerjoin pe minimum 4 tabele, o cerere ce utilizează operația division și o cerere care implementează analiza top-n. (cele 3 cereri sunt diferite de cererile de la exercițiul 12)

→ OUTERJOIN

Să se afișeze denumirile orașelor, denumirile hotelurilor, numele și prenumele angajaților și denumirile departamentelor inclusiv pentru orașele în care nu sunt hoteluri, pentru hotelurile în care nu lucrează niciun angajat și pentru departamentele în care nu lucrează niciun angajat.

```
select o.denumire as denumire_oras, h.denumire as denumire_hotel, a.cod_angajat, a.num, a.prenume, d.denumire as denumire_departament
from orase o
left outer join hoteluri h on h.cod_oras = o.cod_oras
left outer join angajati a on h.cod_hotel = a.cod_hotel
full outer join departamente d on a.cod_departament = d.cod_departament;
```

Script Output x Query Result x

All Rows Fetched: 32 in 0.006 seconds

ID	DENUMIRE_ORAS	DENUMIRE_HOTEL	COD_ANGAJAT	NUME	PRENUME	DENUMIRE_DEPARTAMENT
7	MANGALIA	EMERALD SHORES	35	Bontea	Sorin	BUCATARIE
8	EFORIE	EMERALD SHORES	36	Popa	Cristian	APROVIZIONARE
9	CONSTANTA	OCEAN BREEZE	37	Lungu	Florin	RECEPTIE
10	MANGALIA	OCEAN BREEZE	38	Grigorescu	Claudia	CURATENIE
11	NAVODARI	OCEAN BREEZE	39	Sorescu	Bianca	CONTABILITATE
12	CLUJ NAPOCA	BUSINESS OASIS	40	Liteanu	Sergiu	MENTENANTA
13	BRASOV	EVEREST	41	Obreja	Catalin	SECURITATE
14	BUCURESTI	ROYAL ORCHID	42	Bejenaru	Ioana	IT
15	VATRA DORNEI	EVEREST	43	Dumitru	Carmen	BUCATARIE
16	CLUJ NAPOCA	ROYAL ORCHID	44	Constantinescu	Bogdan	APROVIZIONARE
17	TIMISOARA	ROYAL ORCHID	45	Dobrescu	Roxana	RECEPTIE
18	IASI	ROYAL ORCHID	46	Miculescu	Georgiana	CURATENIE
19	CONSTANTA	EMERALD SHORES	47	Constandache	Alexia	CONTABILITATE
20	MANGALIA	EMERALD SHORES	48	Acatrinei	Horia	MENTENANTA
21	EFORIE	EMERALD SHORES	49	Mircea	Petrisor	SECURITATE
22	CONSTANTA	OCEAN BREEZE	50	Mircea	Draos	IT
23	MANGALIA	OCEAN BREEZE	51	Ouatu	Lidia	BUCATARIE
24	NAVODARI	OCEAN BREEZE	52	Lefter	Razvan	APROVIZIONARE
25	BUCURESTI	BUSINESS OASIS	53	Olaru	Denisa	RECEPTIE
26	CLUJ NAPOCA	BUSINESS OASIS	54	Ciurescu	Crina	CURATENIE
27	BRASOV	EVEREST	55	Unquareanu	Olimpia	CONTABILITATE
28	VATRA DORNEI	EVEREST	56	Chiriacu	Andrei	SECURITATE
29	VASLUI	ARRAKIS	(null)	(null)	(null)	(null)
30	CRAIOVA	(null)	(null)	(null)	(null)	(null)
31	(null)	(null)	(null)	(null)	(null)	DIVERTISMENT
32	(null)	(null)	(null)	(null)	(null)	MARKETING

```
select o.denumire as denumire_oras, h.denumire as denumire_hotel, a.cod_angajat, a.num, a.prenume, d.denumire as denumire_departament
from orase o
left outer join hoteluri h on h.cod_oras = o.cod_oras
```

left outer join angajati a on h.cod_hotel = a.cod_hotel

full outer join departamente d on a.cod_departament = d.cod_departament;

- ➔ **DIVISION**: este un concept în sensul în care nu există ca operație propriu-zisă cum sunt GROUP BY, JOIN, etc. Acesta are la bază ideea de a găsi toate attributele dintr-un tabel X care se potrivesc cu fiecare rând din tabelul Y și de a divide anumite elemente de altele pe baza unor proprietăți.

Să se afișeze facilitățile care sunt incluse *doar* în aceleași abonamente ca și facilitatea cu codul egal cu 2. (se folosește metoda de division cu $A \text{ include } B \Leftrightarrow B - A = \text{mulțimea vidă}$)

Cerința spune că trebuie afișate doar acele facilități care sunt incluse exclusiv în abonamentele în care apare facilitatea cu codul 2. Astfel, trebuie verificat ca o anumită facilitate să nu fie cumva inclusă în mai puține abonamente, dar nici în mai multe. Aici intervine dubla incluziune: $A - B = B - A = \text{mulțimea vidă} \rightarrow \text{mulțimile sunt egale}$.

```
with aux as(select cod_abonament
              from facilitati_abonamente
              where cod_facilitate = 2)
select cod_facilitate, denumire
from facilitati f
where not exists(select *
                  from aux
                  minus
                  select cod_abonament
                  from facilitati_abonamente
                  where f.cod_facilitate = cod_facilitate)
and not exists(select cod_abonament
                from facilitati_abonamente
                where f.cod_facilitate = cod_facilitate
                minus
                select *
                from aux);
```

	COD_FACILITATE	DENUMIRE
1	1	PISCINA INT
2	2	PISCINA EXT

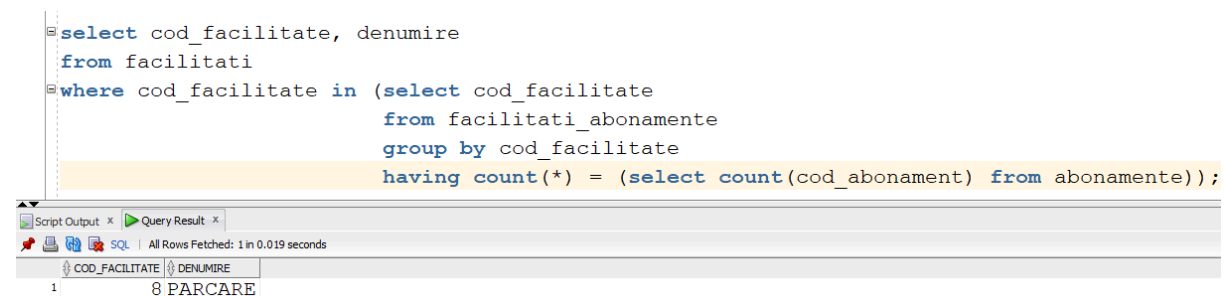
```
with aux as(select cod_abonament
              from facilitati_abonamente
              where cod_facilitate = 2)
select cod_facilitate, denumire
```

```

from facilitati f
where not exists(select *
                  from aux
                  minus
                  select cod_abonament
                  from facilitati_abonamente
                  where f.cod_facilitate = cod_facilitate)
and not exists(select cod_abonament
               from facilitati_abonamente
               where f.cod_facilitate = cod_facilitate
               minus
               select *
               from aux);

```

Să se afișeze toate facilitățile care sunt incluse în toate abonamentele existente în cadrul hotelurilor. (exemplu mai simplu care folosește metoda de division cu count)



```

select cod_facilitate, denumire
from facilitati
where cod_facilitate in (select cod_facilitate
                        from facilitati_abonamente
                        group by cod_facilitate
                        having count(*) = (select count(cod_abonament) from abonamente));

```

➔ ANALIZA TOP-N: este un concept prin care se analizează datele existente și se afișează doar un top al acestora pe baza unor condiții, cerințe.

Să se afișeze hotelurile care au adus primele 3 cele mai mari sume antreprenorului.

```

with aux as(
select h.cod_hotel, h.denumire, suma_totala
from hoteluri h, (select sum(suma) as suma_totala, r.cod_hotel as cd
                  from rezervari r, tranzactii t
                  where r.cod_tranzactie = t.cod_tranzactie
                  group by r.cod_hotel)
where cd = h.cod_hotel)
select cod_hotel, denumire, suma_totala
from aux
where suma_totala in (select suma_totala
                      from (select distinct a.suma_totala
                            from aux a
                            order by 1 desc)
                      where rownum < 4);

```

	COD_HOTEL	DENUMIRE	SUMA_TOTALA
1	12	BUSINESS OASIS	17920
2	11	BUSINESS OASIS	8960
3	1	ROYAL ORCHID	8960
4	6	EMERALD SHORES	7224

```

with aux as(
select h.cod_hotel, h.denumire, suma_totala
from hoteluri h, (select sum(suma) as suma_totala, r.cod_hotel as cd
                  from rezervari r, tranzactii t
                  where r.cod_tranzactie = t.cod_tranzactie
                  group by r.cod_hotel)
where cd = h.cod_hotel)
select cod_hotel, denumire, suma_totala
from aux
where suma_totala in (select suma_totala
                      from (select distinct a.suma_totala
                            from aux a
                            order by 1 desc)
                      where rownum < 4);

```

- 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.

Să se afișeze numele, prenumele, numărul de telefon și banca prin care s-a efectuat tranzacția pentru clienții care au venit la hotelurile antreprenorului în anul 2024 și care au ales ca metodă de plată 'CARD'.

Înainte de optimizare:

Cerere:

```
select nume, prenume, numar_telefon, banca
from clienti c
join rezervari r on c.cod_client = r.cod_client
join tranzactii t on t.cod_tranzactie = r.cod_tranzactie
where to_char(data_venire,'yyyy') = 2024 and tip_tranzactie = 'CARD';
```

Expresie algebrică:

R1 = join(clienti, rezervari, cod_client)

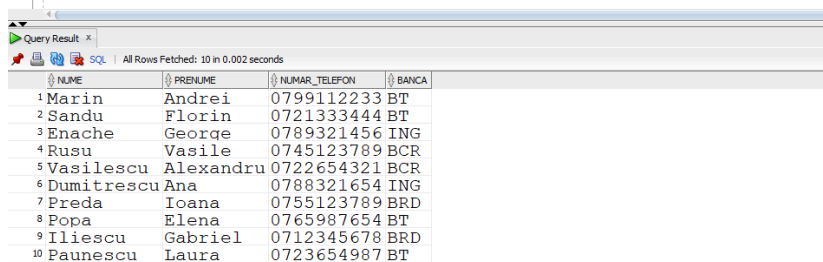
R2 = join(**R1**, tranzactii, cod_tranzactie)

R3 = select(**R2**, to_char(data_venire, 'yyyy') = 2024)

R4 = select(**R3**, tip_tranzactie = 'CARD')

Rezultat : **R5** = project(**R4**, nume, prenume, numar_telefon, banca)

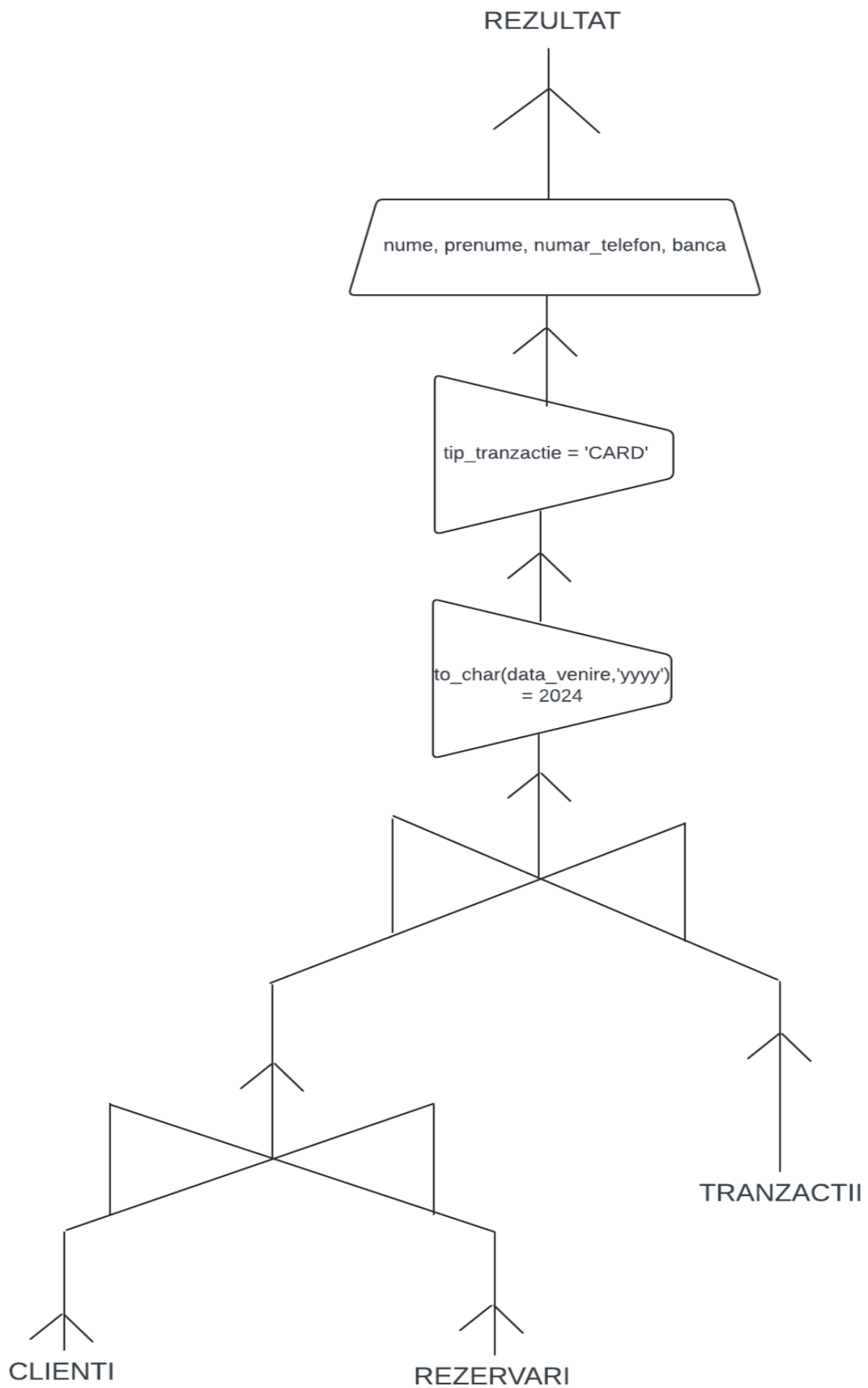
```
select nume, prenume, numar_telefon, banca
from clienti c
join rezervari r on c.cod_client = r.cod_client
join tranzactii t on t.cod_tranzactie = r.cod_tranzactie
where to_char(data_venire,'yyyy') = 2024 and tip_tranzactie = 'CARD';
```



Query Result: X

All Rows Fetched: 10 in 0.002 seconds

	NUME	PRENUME	NUMAR_TELEFON	BANCA
1	Marin	Andrei	0799112233	BT
2	Sandu	Florin	0721333444	BT
3	Enache	George	0789321456	ING
4	Rusu	Vasile	0745123789	BCR
5	Vasilescu	Alexandru	0722654321	BCR
6	Dumitrescu	Ana	0788321654	ING
7	Preda	Ioana	0755123789	BRD
8	Popa	Elena	0765987654	BT
9	Iliescu	Gabriel	0712345678	BRD
10	Paunescu	Laura	0723654987	BT



După optimizare:

Cerere:

```
select nume, prenume, numar_telefon, banca
from (select cod_client, nume, prenume, numar_telefon
      from clienti) clienti
join
  (select cod_client, cod_tranzactie
   from rezervari
   where to_char(data_venire,'yyyy') = 2024) rezervari on clienti.cod_client =
rezervari.cod_client
join
  (select cod_tranzactie, banca
   from tranzactii
   where tip_tranzactie = 'CARD') tranzactii on tranzactii.cod_tranzactie =
rezervari.cod_tranzactie;
```

Regula de optimizare 4 = proiecțiile se execută la început pentru a îndepărta attributele nefolositoare.

Având în vedere că cerința dorește să fie afișate doar câteva attribute din tabela **CLIENTI**, prima oară vom avea grijă să avem o proiecție pentru asta.

R1 = project(CLIENTI, cod_client, nume, prenume, numar_telefon)
(includem și cod_client pentru a putea face join-urile ulterioare)

Regula de optimizare 1 = selecțiile se execută cât mai devreme posibil

R2 = select(REZERVARI, to_char(data_venire, yyyy) = 2024)

R3 = project(**R2**, cod_client, cod_tranzactie)

(includem cod_client, cod_tranzactie pentru a putea face join-urile viitoare)

R4 = join(**R1**, **R3**, cod_client)

R5 = select(TRANZACTII, tip_tranzactie = 'CARD')

(includem cod_tranzactie pentru a putea face join și banca deoarece enunțul cere ca pentru fiecare client să se afișeze banca prin care au făcut tranzacție)

R6 = project(**R4**, cod_tranzactie, banca)

Rezultat : **R7** = join(**R5**, **R6**, cod_tranzactie)

Regula de optimizare 2 = produsele carteziane se înlocuiesc cu join-uri

În cadrul acestui enunț nu sunt necesare produse carteziane.

Regula de optimizare 3 = dacă sunt mai multe join-uri atunci cel mai restrictiv se execută primul

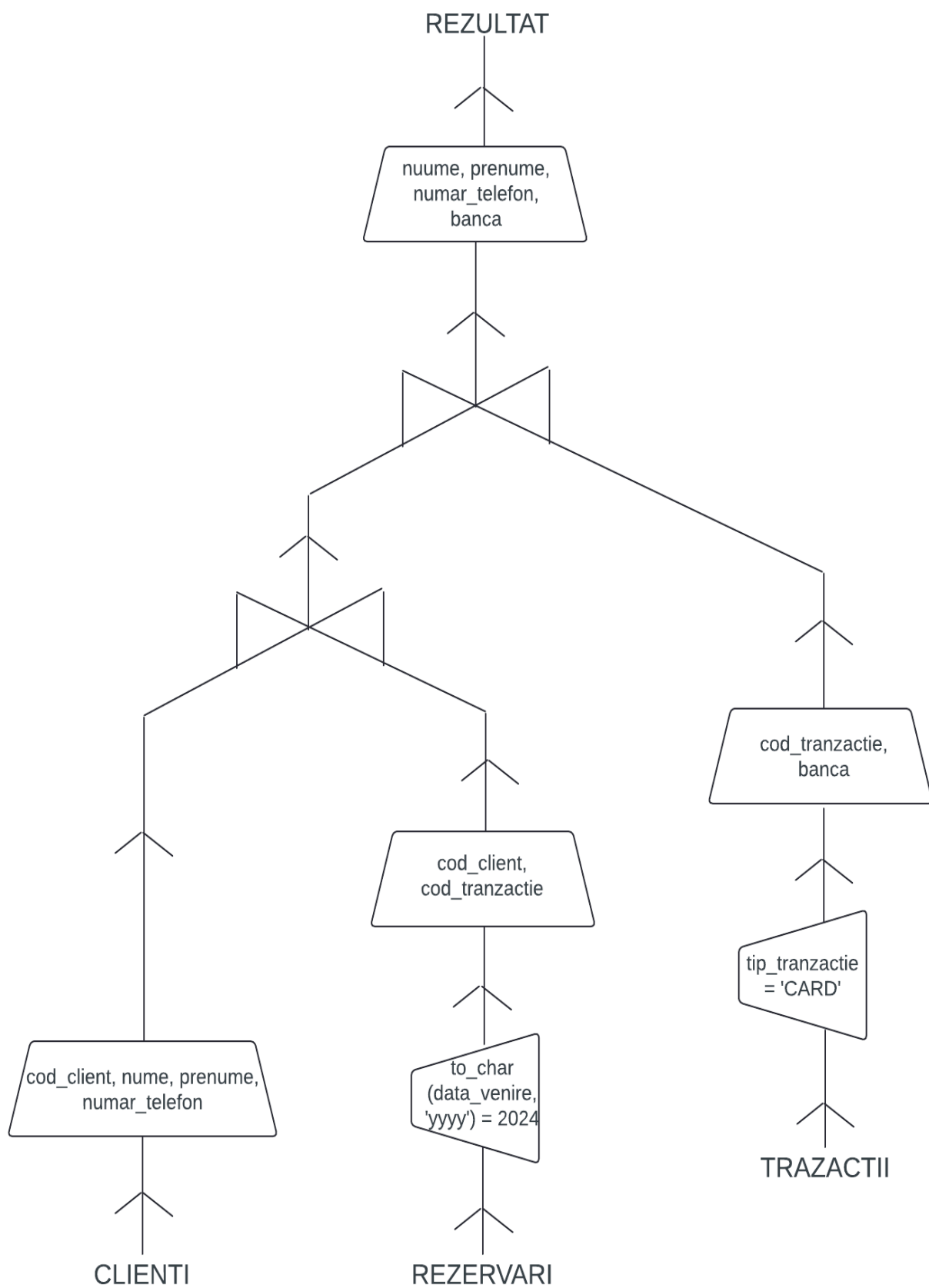
Există 2 join-uri la fel de restrictive deci ordinea lor poate fi oricare.


```

select nume, prenume, numar_telefon, banca
from (select cod_client, nume, prenume, numar_telefon
      from clienti) clienti
join
  (select cod_client, cod_tranzactie
   from rezervari
   where to_char(data_venire,'yyyy') = 2024) rezervari on clienti.cod_client = rezervari.cod_client
join
  (select cod_tranzactie, banca
   from tranzactii
   where tip_tranzactie = 'CARD') tranzactii on tranzactii.cod_tranzactie = rezervari.cod_tranzactie;

```

Query Result x				
SQL All Rows Fetched: 10 in 0.001 seconds				
	NUME	PRENUME	NUMAR_TELEFON	BANCA
1	Marin	Andrei	0799112233	BT
2	Sandu	Florin	0721333444	BT
3	Enache	George	0789321456	ING
4	Rusu	Vasile	0745123789	BCR
5	Vasilescu	Alexandru	0722654321	BCR
6	Dumitrescu	Ana	0788321654	ING
7	Preda	Ioana	0755123789	BRD
8	Popa	Elena	0765987654	BT
9	Iliescu	Gabriel	0712345678	BRD
10	Paunescu	Laura	0723654987	BT



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

Non-BCNF:

Presupunem că ar mai exista o tabelă CURSURI, oferite de antreprenor, la care angajații pot merge pentru a-și îmbunătăți abilitățile (de marketing, culinare, public speaking)

cod_angajat	cod_curs	cod_profesor
1	1	1
1	2	2
2	1	3

Cheia primară este compusă din (cod_angajat, cod_curs), iar tabelul satisface FN1, FN2 și FN3 (nu sunt dependențe tranzitive). Totuși, poate apărea o astfel de dependență cod_profesor -> cod_curs, deci BCNF este încălcat.

BCNF:

Putem separa tabela CURSURI în două tabele separate: ANGAJATI_SI_PROFESORI, PROFESORI_SI_CURSURI

cod_angajat	cod_profesor
1	1
1	2
2	3

cod_profesor	nume	prenume	denumire_curs
1	Popovici	Iulian	ARTE_CULINARE
2	Marin	Ionela	PUBLIC_SPEAKING
3	Lazar	Matei	EXCELL&IT