**Proiect Baze de Date**

**Calomfirescu Victor – 144**

**Content**

[1. Descrierea modelului real, a utilității acestuia și a regulilor de funcționare: 5](#_Toc198640140)

[2. Prezentarea constrângerilor (restricții, reguli) impuse asupra modelului: 5](#_Toc198640141)

[3. Descrierea entităților și cheile primare: 6](#_Toc198640142)

[4. Descrierea relațiilor și cardinalitățile: 6](#_Toc198640143)

[5. Descrierea atributelor, incluzând tipul de date și eventualele constrângeri, valori implicite, valori posibile ale atributelor. 6](#_Toc198640144)

[6.Realizarea diagramei entitate-relație corespunzătoare descrierii de la punctele 3-5. 8](#_Toc198640145)

[7. Diagrama conceptuala 8](#_Toc198640146)

[8. Schemele relaționale corespunzătoare diagramei conceptuale proiectate la pct 7 9](#_Toc198640147)

[9. Realizarea normalizării până la forma normală 3 (FN1-FN3). 11](#_Toc198640148)

[10. Crearea unei secvențe ce va fi utilizată în inserarea înregistrărilor în tabele (punctul 11). 14](#_Toc198640149)

[11. Crearea tabelelor în SQL și inserarea de date coerente în fiecare dintre acestea 14](#_Toc198640150)

[12. Cereri SQL Complexe 16](#_Toc198640157)

[Cerere 1. Subcerere corelată pe 3 tabele. RESPECTA PUNCTUL a) 17](#_Toc198640158)

[Cerere 2. Subcereri nesincronizate în FROM . RESPECTA PUNCTUL b) 18](#_Toc198640159)

[Cerere 3. Grupări, funcții grup și HAVING cu subcerere nesincronizată RESPECTA PUNCTUL c) 19](#_Toc198640160)

[Cerere 4. ORDER BY, COALESCE (NVL) și DECODE (expresie CASE) RESPECTA PUNCTUL d) 20](#_Toc198640161)

[Cerere 5. Funcții pe șiruri și date, expresie CASE, bloc WITH. RESPECTA PUNCTELE e) si f) 21](#_Toc198640162)

[13. Implementarea a 3 operații de actualizare și de suprimare a datelor utilizând subcereri 23](#_Toc198640163)

[14. Crearea unei vizualizări complexe 28](#_Toc198640164)

[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*. 29](#_Toc198640165)

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

Compania (clientul) închiriază online unul sau mai multe birouri (sau spații de lucru) dintr-un centru de co-working, pentru o perioadă de timp, cu scopul de a fi folosite de utilizatori (angajați sau colaboratori ai companiei). Compania semnează contractul și efectuează plățile. Utilizatorul este persoana care folosește efectiv spațiul de lucru. Birourile (spațiile de lucru) se pot încadra în diverse tipuri, au un anumit număr de locuri (1–4), pot dispune de diverse facilități se pot afta in diferite cladiri. Cladirile sunt de un anumit tip, au o adresa, si un proprietar. Plata se face prin diverse metode (cash, card, transfer bancar). Doar compania are dreptul să modifice perioada de închiriere, nu și utilizatorii.

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

* Valorile cheilor primare (Companie\_ID, Inchiriere\_ID, Birou\_ID, Facilitate\_ID, Tip\_Birouri\_ID, Cladiri\_ID, Numar\_Locuri\_ID, Metoda\_plata\_ID, Utilizator\_ID, Plati\_ID) nu pot fi NULL
* Există un număr limitat de cladiri, MAX 10, fiecare cladire are maxim 10 etaje.
* Birourile pot fi in diferite cladiri, iar birourile sunt numerotate de la 1 la 100 (de exemplu, biroul 21 se află pe etajul 2).
* Un birou are maximum 4 locuri.
* Birourile sunt de mai multe tipuri, de exemplu:
  + 'Birou deschis',
  + 'Birou dedicat',
  + 'Birou privat',
  + 'Birou pentru echipă',
  + 'Sală de conferință'
* Facilitățile posibile includ:
  + 'Espressor de cafea',
  + 'Imprimanta',
  + 'TV',
  + 'Proiector',
  + 'Microfoane profesioniste',
  + 'Tablă smart',
  + 'Aparatură de conferință video',
  + 'Parcare',
  + 'Serviciu de recepție';
* Plățile se pot face prin: cash, card, transfer bancar.
* Doar compania poate modifica data de început și data de final a închirierii.

## 3. Descrierea entităților și cheile primare:

* Companii care are cheie primara: Companii \_ID
* Inchirieri care are cheie primara: Inchirieri \_ID
* Plati care are cheie primara: Plati\_ID
* Metoda\_de\_Plata care are cheie primara: Metoda\_plata\_ID
* Cladiri are cheia primara: Cladiri\_ID
* Birouri care are cheie primara: Birouri \_ID
* Tip\_Birouri care are cheie primara: Tip\_ Birouri \_ID
* Numar\_Locuri care are cheie primara: Numar\_ Locuri \_ID
* Lista\_Facilitati care are cheie primara: Facilitate\_ID
* Utilizatori care are cheie primara: Utilizatori \_ID
* Inchiriere\_birouri: nu are cheie primara si este un tabel asociativ ce leaga entitatile: Utilizator, Inchiriere si Birou.
* Facilitati\_Birouri: nu are cheie primara si are un tabel asociativ ce leaga entitatile: Birouri si Lista\_Facilitati.

## 4. Descrierea relațiilor și cardinalitățile:

* O Companie poate face mai multe Inchirieri (1 : M).
* O Companie poate face mai multe Plati (1 : M).
* O Inchiriere poate include mai multe rezervări de birouri și utilizatori (1 : M).
* O Inchiriere poate fi achitată prin mai multe Plati (1 : M).
* O Metoda\_de\_Plata este asociată mai multor Plati (1 : M).
* Un Utilizator poate participa la mai multe Inchirieri (1 : M).
* Un Birou poate apărea în mai multe Inchirieri (1 : M).
* Un Birou are mai multe facilități (1 : M).
* O Inchiriere are mai multe Plati (1:M)
* O Cladire are mai multe birouri. (1:M)
* O Facilitate se poate regăsi în mai multe birouri (1 : M).
* Un Tip\_Birouri poate fi asociat mai multor Birouri (1 : M).
* Un Numar\_Locuri poate fi asociat mai multor Birouri (1 : M).

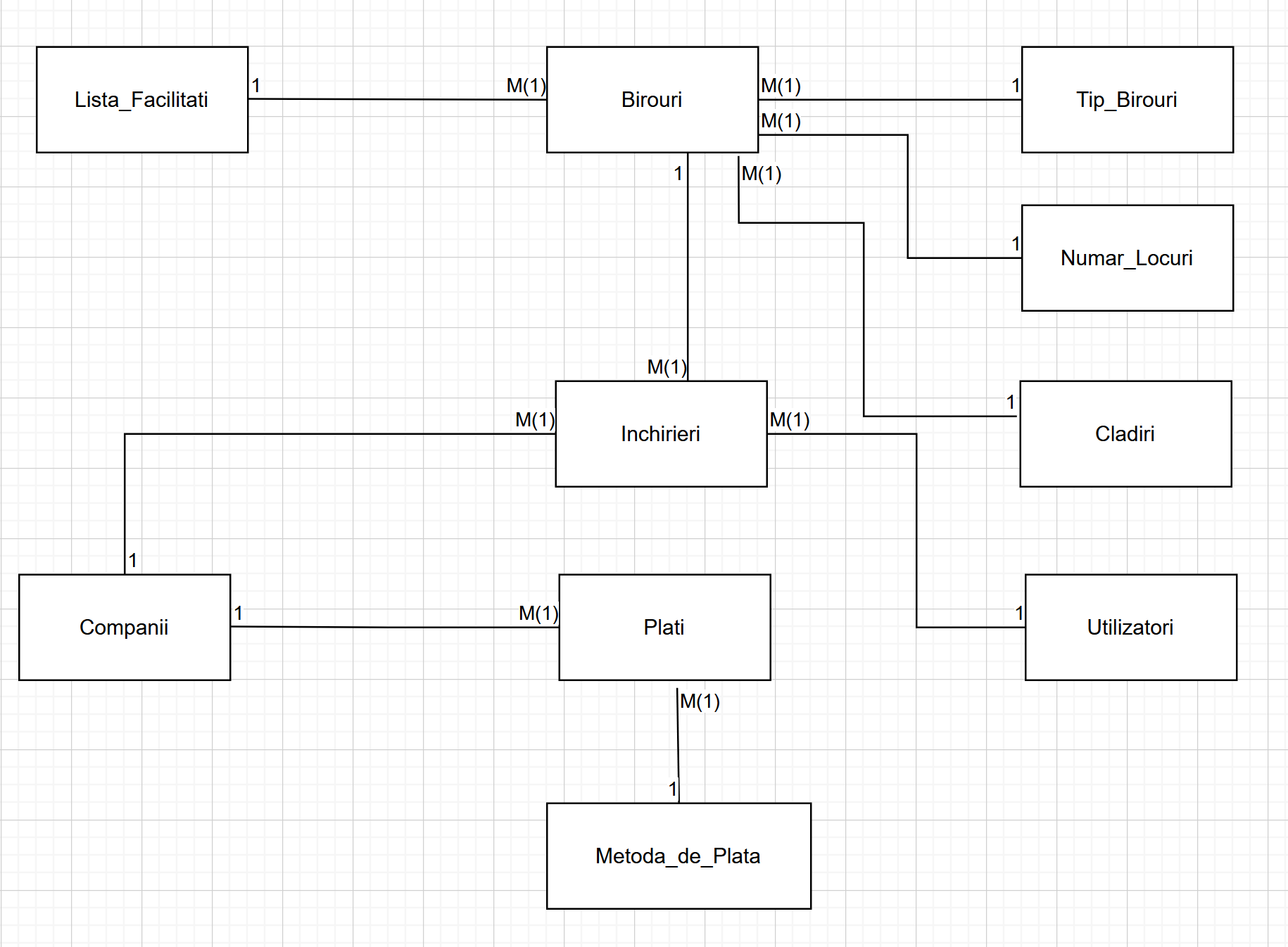
## 5. Descrierea atributelor, incluzând tipul de date și eventualele constrângeri, valori implicite, valori posibile ale atributelor.

Descrierea atributelor:

* **Companii**:
  + Nume: VARCHAR2(255), diferit de NULL
  + Email: VARCHAR2(255), diferit de NULL
  + Telefon: VARCHAR2(50), diferit de NULL
* **Inchirieri**:
  + Data\_inchirierii: DATE, diferit de NULL (data contractului)
  + Data\_start: DATE, diferit de NULL
  + Data\_final: DATE, diferit de NULL
* **Metoda\_de\_Plata**:
  + Metoda\_Plata: VARCHAR2(50), diferit de NULL (valori: cash, card, transfer bancar)
* **Plati**:
  + Suma\_de\_Plata: NUMBER (12,2), diferit de NULL, > 0
  + Data\_plata: DATE, diferint de NULL
* **Utilizatori**:
  + Nume: VARCHAR2(100), diferit de NULL
  + Prenume: VARCHAR2(100), diferit de NULL
  + Email: VARCHAR2(255), diferit de NULL
  + Telefon: VARCHAR2(50), diferit de NULL
* **Cladiri:**
  + Denumire: VARCHAR2(200), diferit de NULL
  + Adresa: VARCHAR2(400), diferit de NULL
  + Proprietar: VARCHAR2(255), diferit de NULL
* **Birouri**:
  + Etaj: NUMBER, diferit de NULL, valori intre 1 si 10
  + Numar: NUMBER, diferit de NULL, valori intre 1 si 100
* **Lista\_Facilitati**:
  + Descriere\_Facilitate: VARCHAR2(200), diferit de NULL (ex.: aparat de cafea, proiector, TV, microfoane profesioniste, imprimantă etc.)
* **Tip\_Birouri**:
  + Tip: VARCHAR2(50), diferit de NULL
    - Posibile valori: Open Desk, Dedicated Desk, Private Office, Team Office, Conference Room
  + Pret\_Loc: NUMBER(10,2), diferit de NULL, > 0
* **Numar\_Locuri**:
  + Numar\_Locuri: NUMBER, diferit de NULL, valori posibile intre 1 si 4.

Inchiriere\_Birouri si Facilitati\_Birouri nu au atribute, ele fiind tabele asociative.

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

****

## 7. Diagrama conceptuala

Diagrama conceptuala:

Companii = {Companie\_ID, Nume, Email, Telefon}

Inchirieri = {Inchiriere\_ID, Companie\_ID, Data\_inchirierii, Data\_start, Data\_final}

Metoda\_de\_Plata = {Metoda\_plata\_ID, Metoda\_Plata}

Plati = {Plata\_ID, Inchiriere\_ID, Companie\_ID, Metoda\_plata\_ID, Suma\_de\_Plata, Data\_plata}

Utilizatori = {Utilizator\_ID, Nume, Prenume, Email, Telefon}

Birouri = {Birou\_ID, Tip\_Birouri\_ID, Numar\_Locuri\_ID, Cladire\_ID, Etaj, Numar}

Lista\_Facilitati = {Facilitate\_ID, Descriere\_Facilitate}

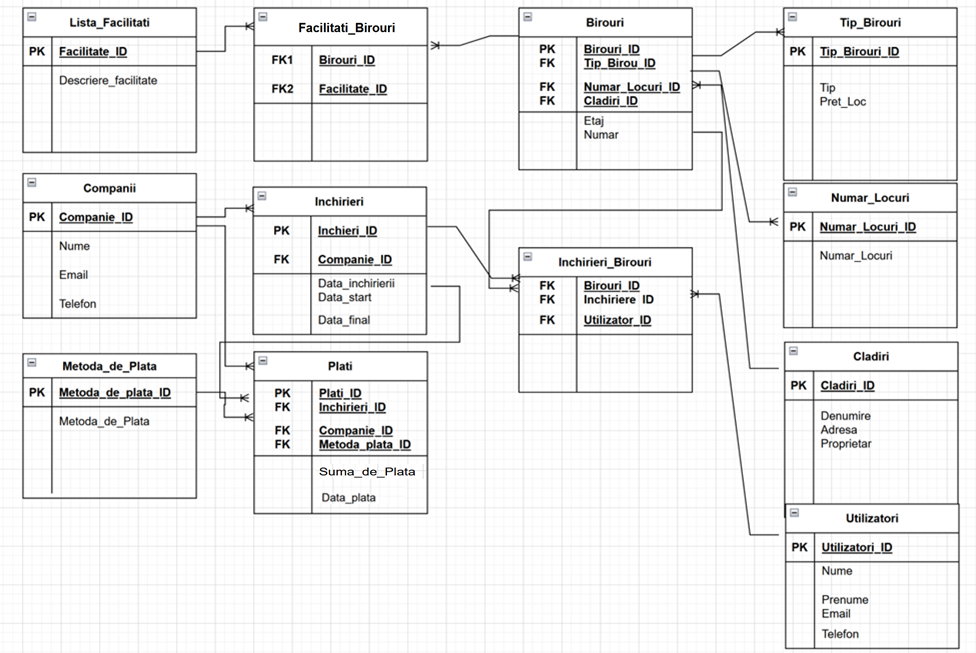
Cladiri = {Cladire\_ID, Denumire, Adresa, Proprietar}

Tip\_Birouri = {Tip\_Birouri\_ID, Tip, Pret\_Loc}

Numar\_Locuri = {Numar\_Locuri\_ID, Numar\_Locuri}

Inchirieri\_Birouri = {Birouri\_ID, Inchirieri\_ID, Utilizator\_ID}

Facilitati\_Birouri = {Birouri\_ID, Facilitate\_ID}

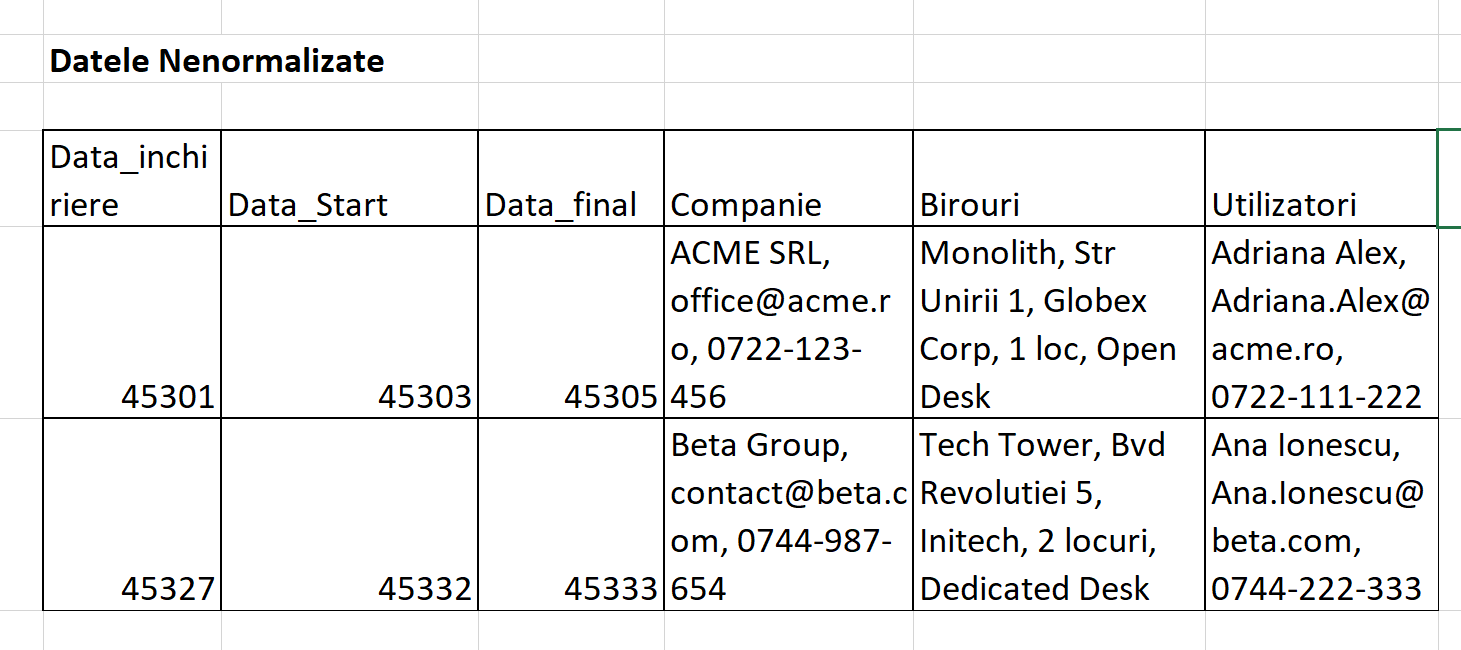


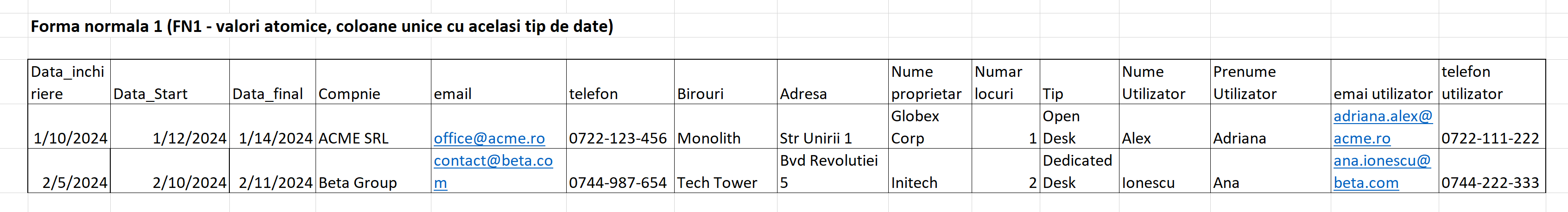
## 8. Schemele relaționale corespunzătoare diagramei conceptuale proiectate la pct 7

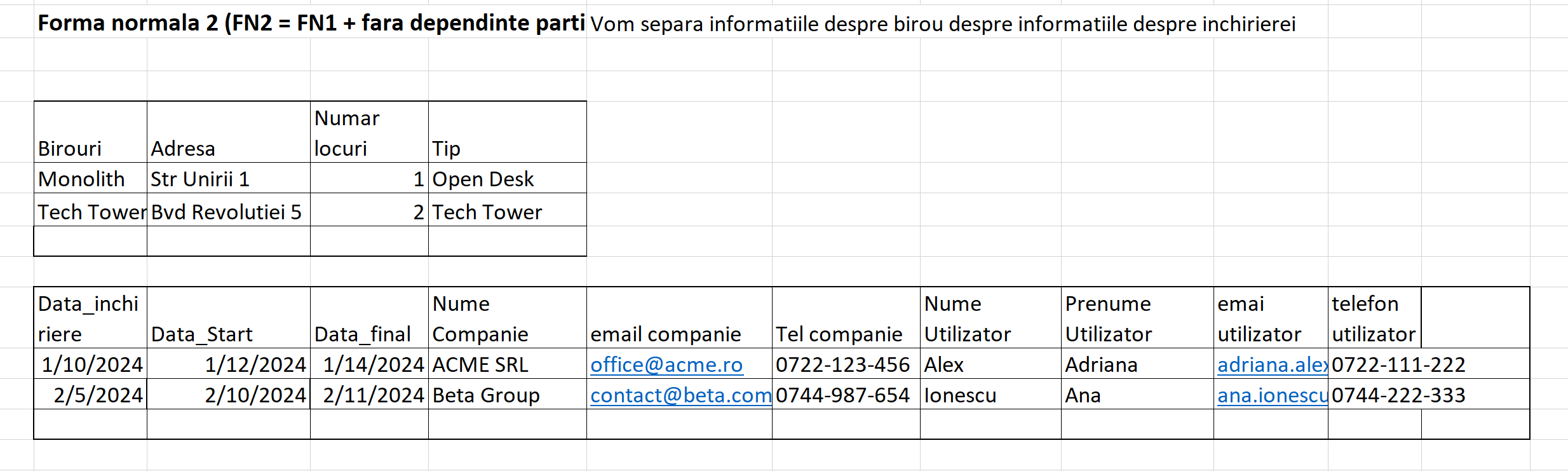
Schema relationala:

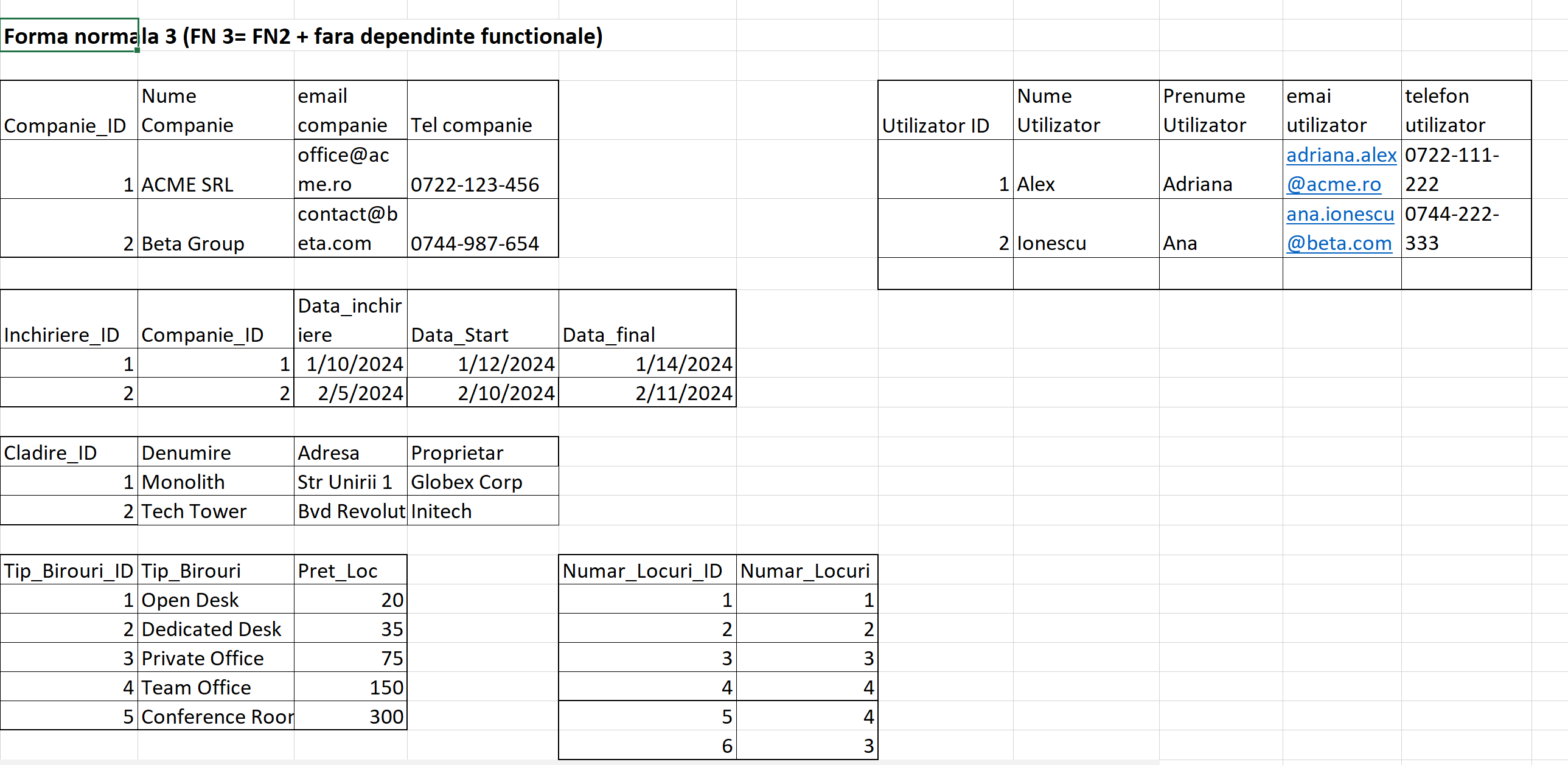
* 1. Companii:
     1. Companie\_ID: NUMBER
     2. Nume: VARCHAR2(255)
     3. Email: VARCHAR2(255)
     4. Telefon: VARCHAR2(50)
  2. Inchirieri:
     1. Inchirieri\_ID: NUMBER
     2. Companie\_ID: NUMBER
     3. Data\_inchirierii: DATE
     4. Data\_start: DATE
     5. Data\_final: DATE
  3. Metoda\_de\_Plata:
     1. Metoda\_plata\_ID: NUMBER
     2. Metoda\_de\_Plata: VARCHAR2(50)
  4. Plati:
     1. Plata\_ID: NUMBER
     2. Inchirieri\_ID: NUMBER
     3. Companie\_ID: NUMBER
     4. Metoda\_plata\_ID: NUMBER
     5. Suma\_de\_Plata: NUMBER (12,2)
     6. Data\_plata: DATE
  5. Utilizatori:
     1. Utilizatori\_ID: NUMBER
     2. Nume: VARCHAR2(100)
     3. Prenume: VARCHAR2(100)
     4. Email: VARCHAR2(255)
     5. Telefon: VARCHAR2(50)
  6. Birouri:
     1. Birou\_ID: NUMBER
     2. Tip\_Birouri\_ID: NUMBER
     3. Numar\_Locuri\_ID: NUMBER
     4. Cladiri\_ID: NUMBER
     5. Etaj: NUMBER
     6. Numar: NUMBER
  7. Lista\_Facilititati:
     1. Facilitate\_ID: NUMBER
     2. Descriere\_Facilitate: VARCHAR2(200)
  8. Tip\_Birouri:
     1. Tip\_Birouri\_ID: NUMBER
     2. Tip: VARCHAR2(50)
     3. Pret\_Loc: NUMBER(10,2)
  9. Numar\_Locuri:
     1. Numar\_Locuri\_ID: NUMBER
     2. Numar\_Locuri: NUMBER
  10. Cladiri:
      1. Cladiri \_ID: NUMBER
      2. Denumire: VARCHAR2(200)
      3. Adresa: VARCHAR2(400)
      4. Proprietar: VARCHAR2(255)
  11. Inchirieri\_Birouri:
      1. Birou\_ID: NUMBER
      2. Inchirieri\_ID: NUMBER
      3. Utilizator\_ID: NUMBER
  12. Facilitati\_Birouri
      1. Birou\_ID: NUMBER
      2. Facilitate\_ID: NUMBER

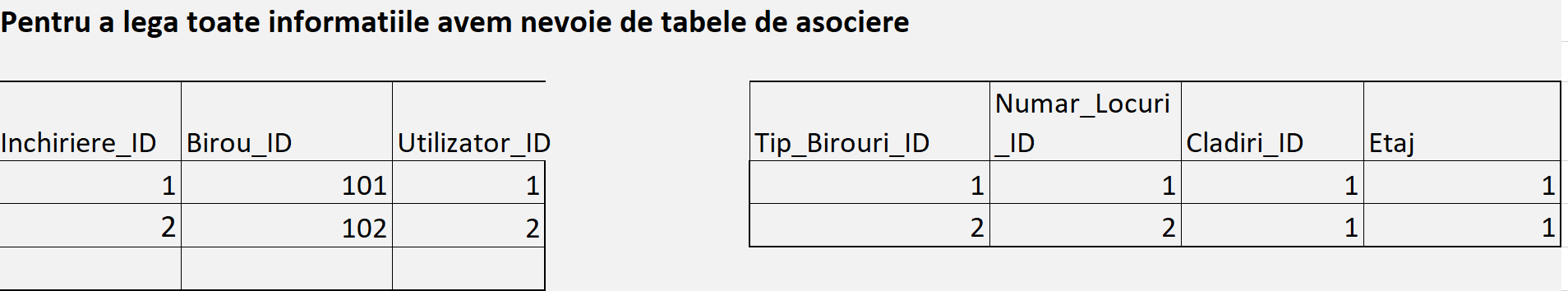
## 9. Realizarea normalizării până la forma normală 3 (FN1-FN3).

****

****

****

****

****

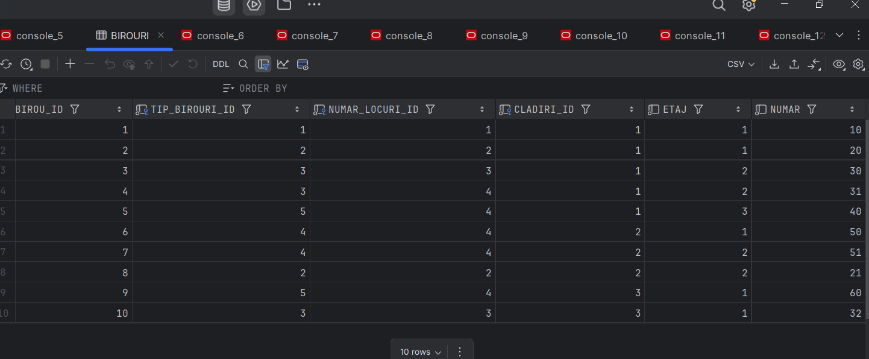
## 10. Crearea unei secvențe ce va fi utilizată în inserarea înregistrărilor în tabele (punctul 11).

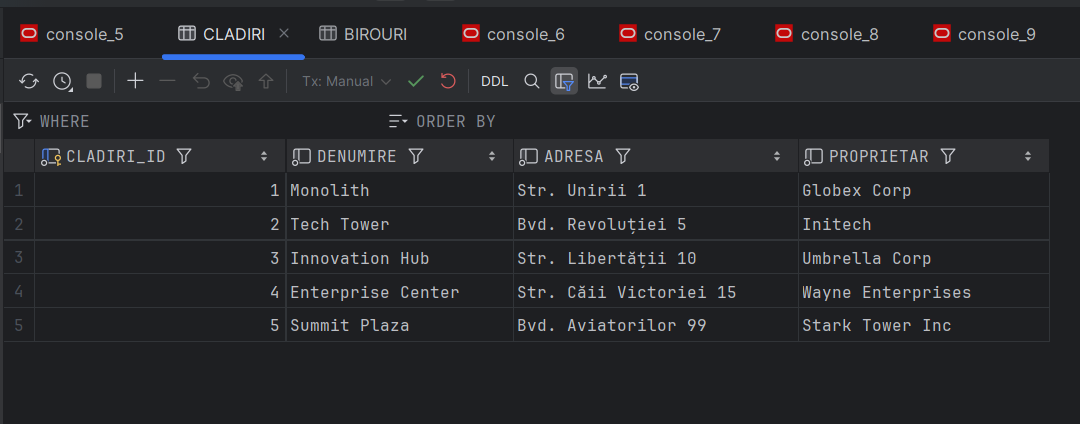
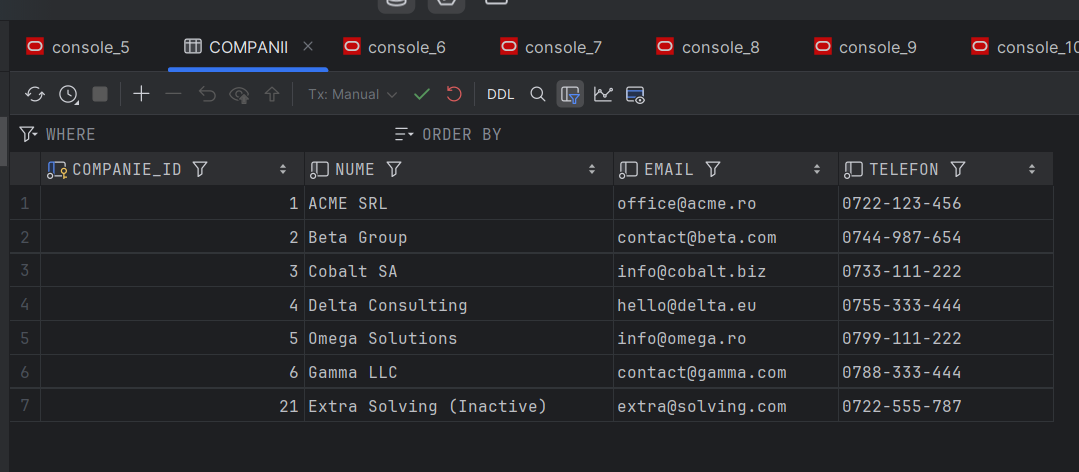
* Instalez Oracle de pe link-ul din fisier.
* Se deschide DataGrip si se creeaza un nou proiect
* Se creaza un nou data source oracle, conectat cu baza de date deja instalata
* Se alege Host: localhost, Port 1521, SID: XE si ne autentificam cu userul si parola specifica serverului oracle deja instalat, Verificam conectiunea pe "Test Connection" si daca este "Succesful" apasam pe OK.
* Apasam pe "+" si cream "query console" si de aici incepem introducerea codului de mai jos din fisierul **144\_Calomfirescu\_Victor – Creare\_Inserare.txt.**

## 11. Crearea tabelelor în SQL și inserarea de date coerente în fiecare dintre acestea

Crearea efectiva si inserarea datelor conform fisierului 144\_Calomfirescu\_Victor -Creare\_Inserare.txt

(minimum 5 înregistrări în fiecare tabel neasociativ; minimum 10 înregistrări în tabelele asociative; maxim 30 de înregistrări în fiecare tabel).





## 

## 

## 

## 

## 

## 

## 12. Cereri SQL Complexe

/\*Formulați în limbaj natural și implementați 5 cereri SQL complexe ce vor utiliza, în ansamblul lor, următoarele elemente:

* 1. subcereri sincronizate în care intervin cel puțin 3 tabele
  2. subcereri nesincronizate în clauza FROM
  3. grupări de date, funcții grup, filtrare la nivel de grupuri cu subcereri nesincronizate (în clauza de HAVING)
  4. ordonări și utilizarea funcțiilor NVL și DECODE (în cadrul aceleiași cereri)
  5. utilizarea a cel puțin 2 funcții pe șiruri de caractere, 2 funcții pe date calendaristice, a cel puțin unei expresii CASE
  6. utilizarea a cel puțin 1 bloc de cerere (clauza WITH)\*/

**Conform fisierului 144\_Calomfirescu\_Victor – exemple 12-15.txt**

### Cerere 1. Subcerere corelată pe 3 tabele. RESPECTA PUNCTUL a)

/\*

Descriere: Afișează numele companiilor care

au făcut cel puțin o plată pentru un birou situat la etajul 3

Acest cod rezolva cerinta a)

\*

SELECT

c.nume AS companie

FROM companii c

WHERE EXISTS (

SELECT 1

FROM plati p

JOIN inchirieri i

ON p.inchiriere\_id = i.inchiriere\_id

JOIN inchirieri\_birouri ib

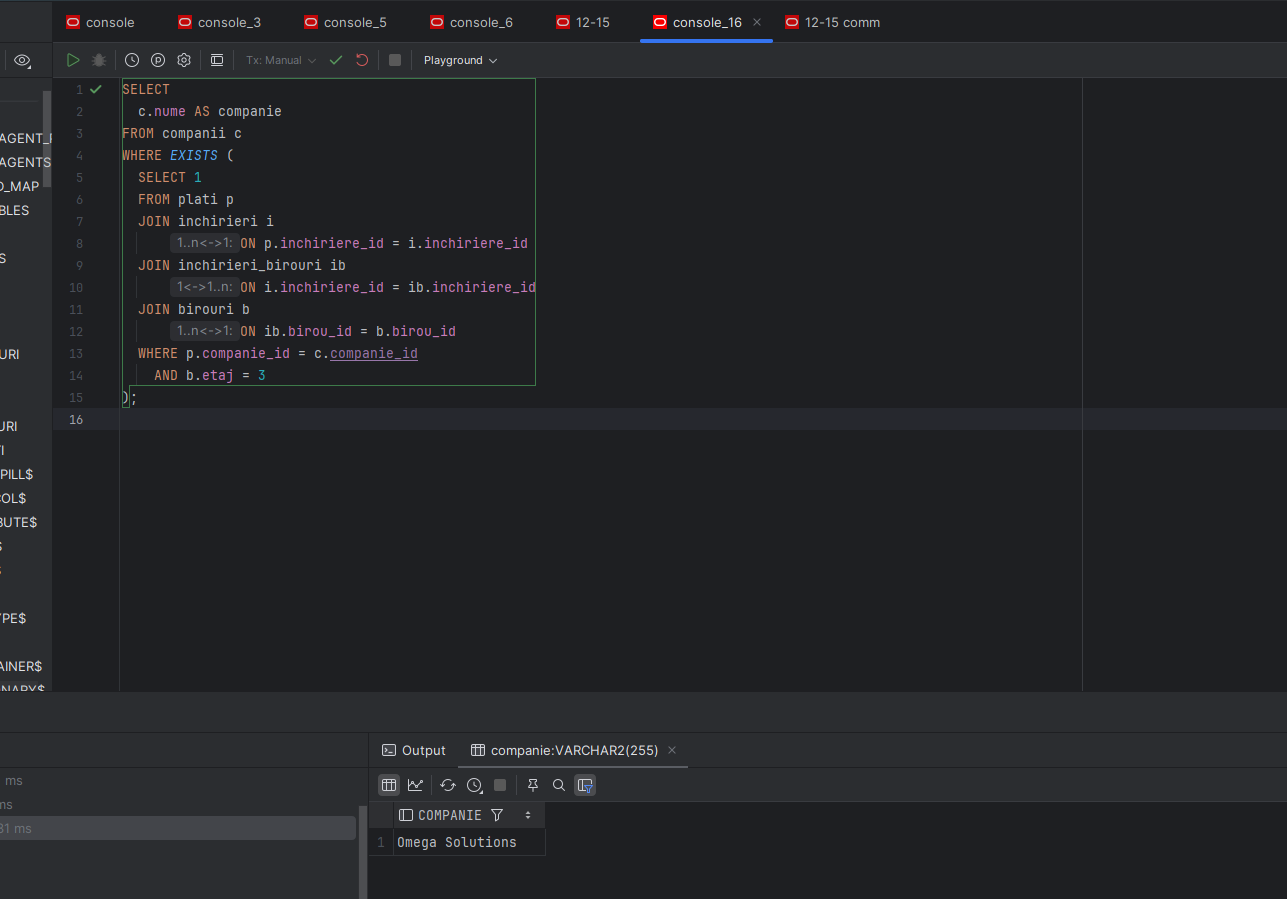
ON i.inchiriere\_id = ib.inchiriere\_id

JOIN birouri b

ON ib.birou\_id = b.birou\_id

WHERE p.companie\_id = c.companie\_id

AND b.etaj = 3

); 

### Cerere 2. Subcereri nesincronizate în FROM . RESPECTA PUNCTUL b)

/\* ============================================================

Descriere: Calculează pentru fiecare clădire prețul mediu

per loc și afișează lista de clădiri cu acest

preț, sortată descrescător.

Acest cod rezolva cerinta b)

\*/

SELECT

c.Cladiri\_ID,

c.Denumire,

stats.pret\_med\_per\_loc

FROM Cladiri c

JOIN (

SELECT

b.Cladiri\_ID,

AVG(t.Pret\_Loc / nl.Numar\_Locuri) AS pret\_med\_per\_loc

FROM Birouri b

JOIN Tip\_Birouri t

ON b.Tip\_Birouri\_ID = t.Tip\_Birouri\_ID

JOIN Numar\_Locuri nl

ON b.Numar\_Locuri\_ID = nl.Numar\_Locuri\_ID

GROUP BY

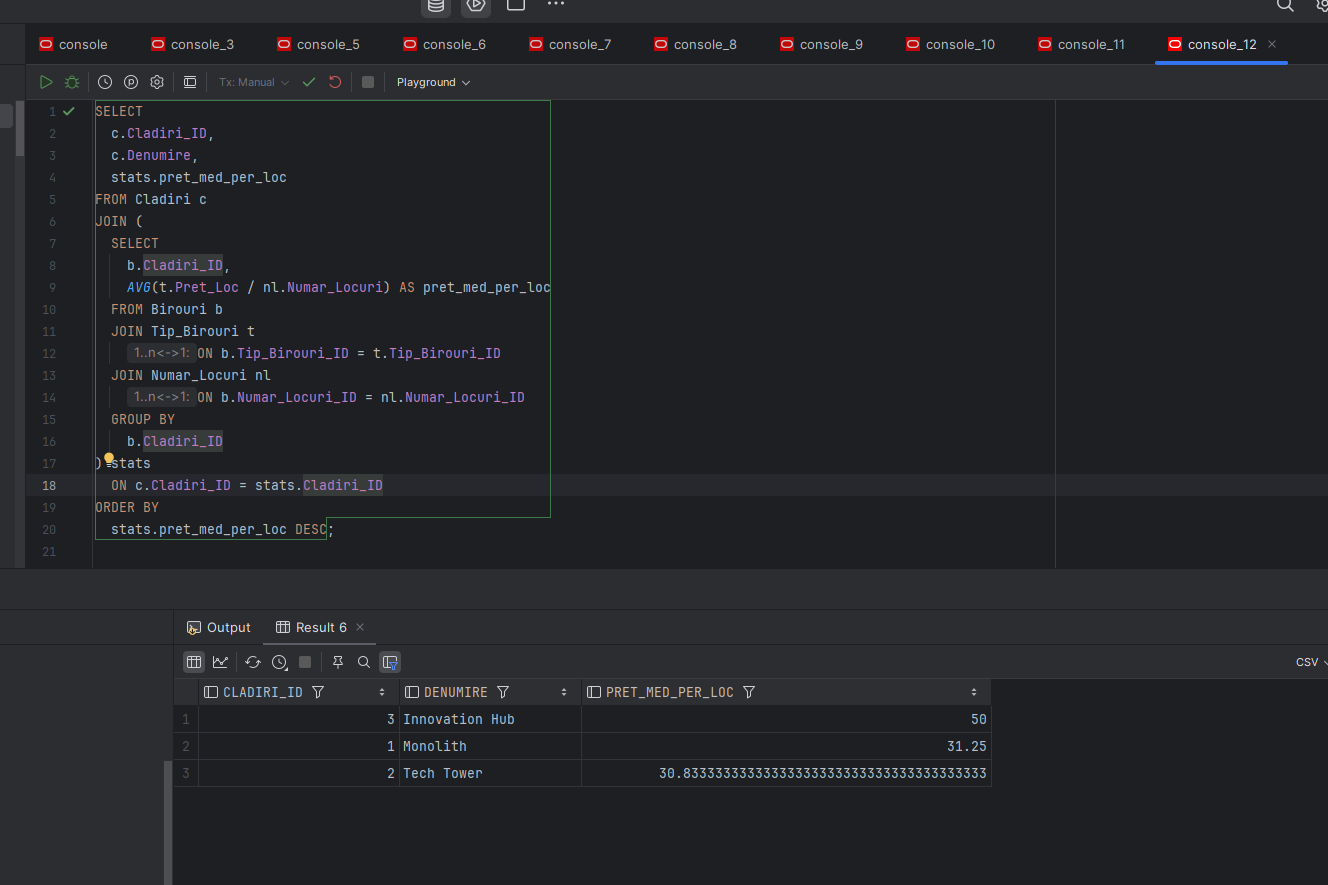
b.Cladiri\_ID

) stats

ON c.Cladiri\_ID = stats.Cladiri\_ID

ORDER BY

stats.pret\_med\_per\_loc DESC;



### Cerere 3. Grupări, funcții grup și HAVING cu subcerere nesincronizată RESPECTA PUNCTUL c)

/\* ============================================================

Descriere: Afișează companiile și totalul cheltuielilor lor,

dar numai pe acelea al căror total depășește

media totală a cheltuielilor tuturor companiilor.

Acest cod rezolva cerinta c)

\*/

SELECT

c.companie\_id,

c.nume,

SUM(p.suma\_de\_plata) AS total\_incasari

FROM companii c

JOIN plati p

ON c.companie\_id = p.companie\_id

GROUP BY c.companie\_id, c.nume

HAVING SUM(p.suma\_de\_plata) > (

SELECT AVG(sub.total)

FROM (

SELECT SUM(p2.suma\_de\_plata) AS total

FROM plati p2

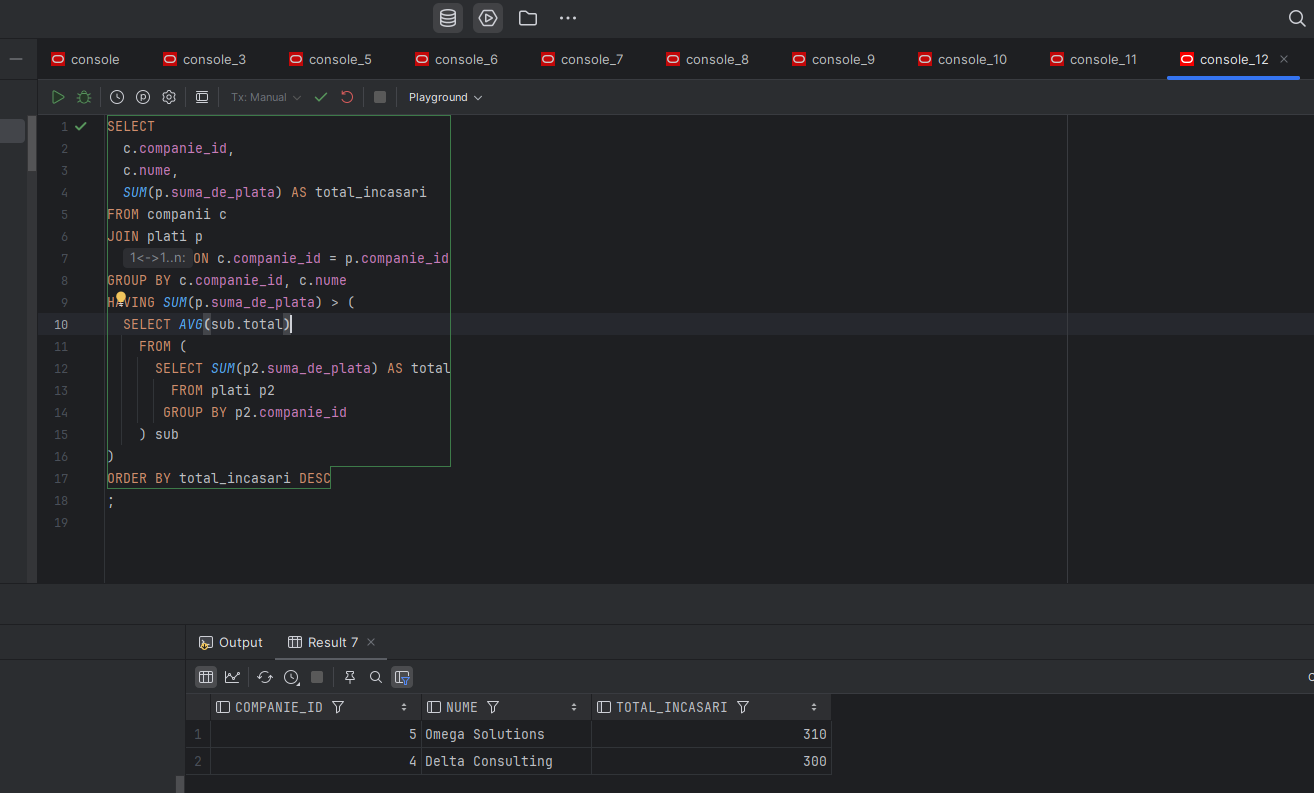
GROUP BY p2.companie\_id

) sub

)

ORDER BY total\_incasari DESC

;



### Cerere 4. ORDER BY, COALESCE (NVL) și DECODE (expresie CASE) RESPECTA PUNCTUL d)

/\* ============================================================

Descriere: Afișează fiecare plată cu ID-ul, numele

companiei, suma (0 dacă e NULL), descrierea

metodei de plată prin DECODE și data plății,

ordonate descrescător după dată.

Acest cod rezolva cerinta d)

\*/

SELECT

p.plati\_id,

c.nume AS companie,

NVL(p.suma\_de\_plata, 0) AS suma,

DECODE(

mp.metoda\_plata,

'cash', 'Plata numerar',

'card', 'Plata cu card',

'transfer bancar', 'Plata prin transfer',

'Altă metodă'

) AS metoda\_prietenoasa,

p.data\_plata

FROM plati p

JOIN companii c

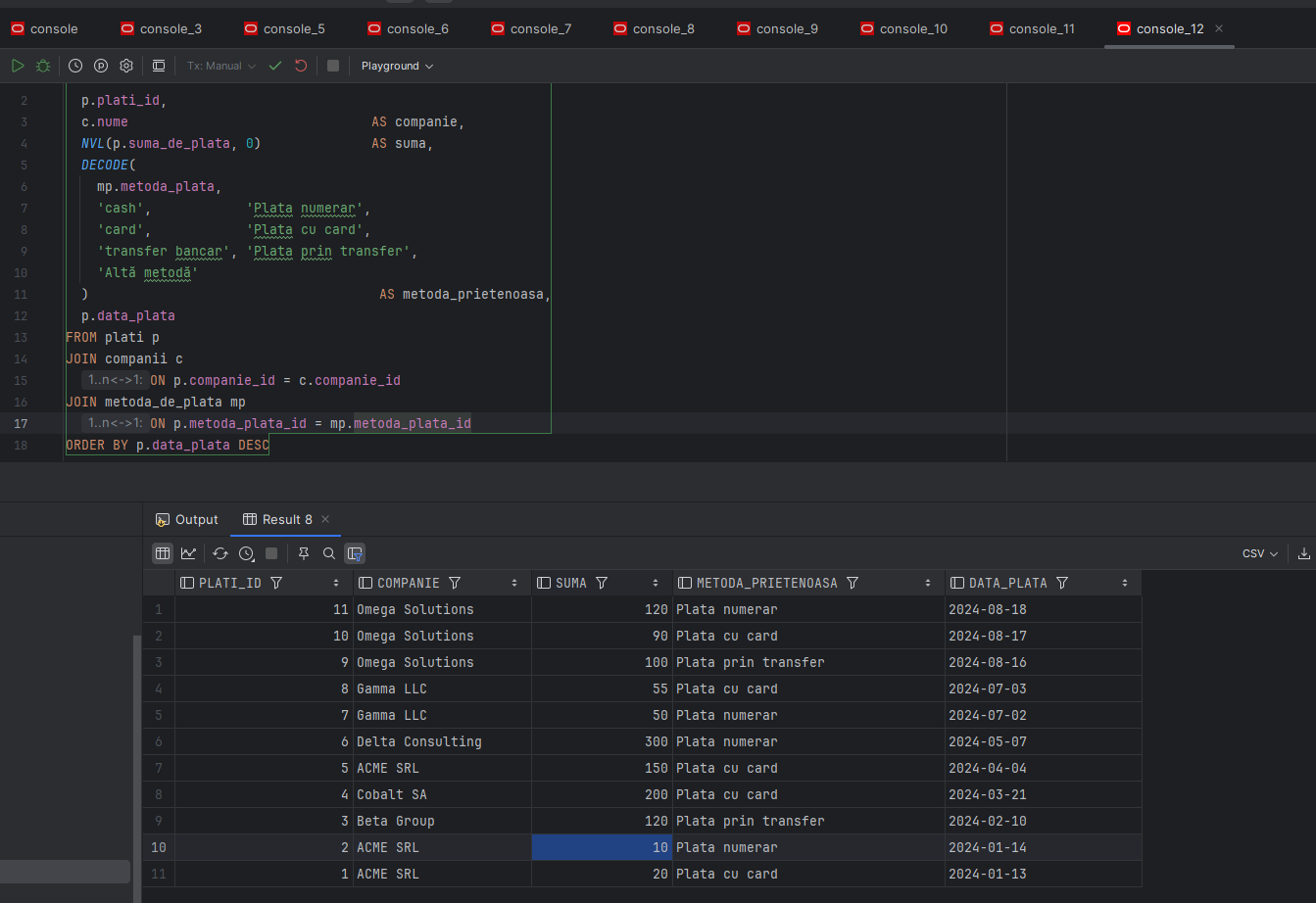
ON p.companie\_id = c.companie\_id

JOIN metoda\_de\_plata mp

ON p.metoda\_plata\_id = mp.metoda\_plata\_id

ORDER BY p.data\_plata DESC

;



### Cerere 5. Funcții pe șiruri și date, expresie CASE, bloc WITH. RESPECTA PUNCTELE e) si f)

/\* ============================================================

Descriere: Pentru fiecare birou, află data ultimei

închirieri, numărul de zile scurse de atunci

și clasifică-l (‘Recent’, ‘Moderate’, ‘Vechi’),

afișând și un cod și un tip prescurtat.

Acest cod rezolva cerinta e) si f)

\*/

WITH ultimele\_inchirieri AS (

SELECT

b.birou\_id,

MAX(i.data\_final) AS data\_ultima

FROM birouri b

JOIN inchirieri\_birouri ib

ON b.birou\_id = ib.birou\_id

JOIN inchirieri i

ON ib.inchiriere\_id = i.inchiriere\_id

GROUP BY b.birou\_id

)

SELECT

'Birou ' || TO\_CHAR(b.birou\_id) AS cod\_birou, -- funcție pe șir

UPPER(SUBSTR(t.tip,1,3)) AS tip\_scurt, -- funcții pe șir

ui.data\_ultima AS data\_final,

TRUNC(SYSDATE - ui.data\_ultima) AS zile\_trecute, -- funcții pe date

CASE -- expresie CASE

WHEN SYSDATE - ui.data\_ultima < 300 THEN 'Recent'

WHEN SYSDATE - ui.data\_ultima <= 400 THEN 'Moderate'

ELSE 'Vechi'

END AS categorie

FROM birouri b

JOIN ultimele\_inchirieri ui

ON b.birou\_id = ui.birou\_id

JOIN tip\_birouri t

ON b.tip\_birouri\_id = t.tip\_birouri\_id

ORDER BY zile\_trecute DESC

;

SELECT

b.Birou\_ID,

'Etaj ' || b.Etaj || ' #' || b.Numar AS DescriereBirou,

( -- subcerere corelată

SELECT COUNT(\*)

FROM Plati p

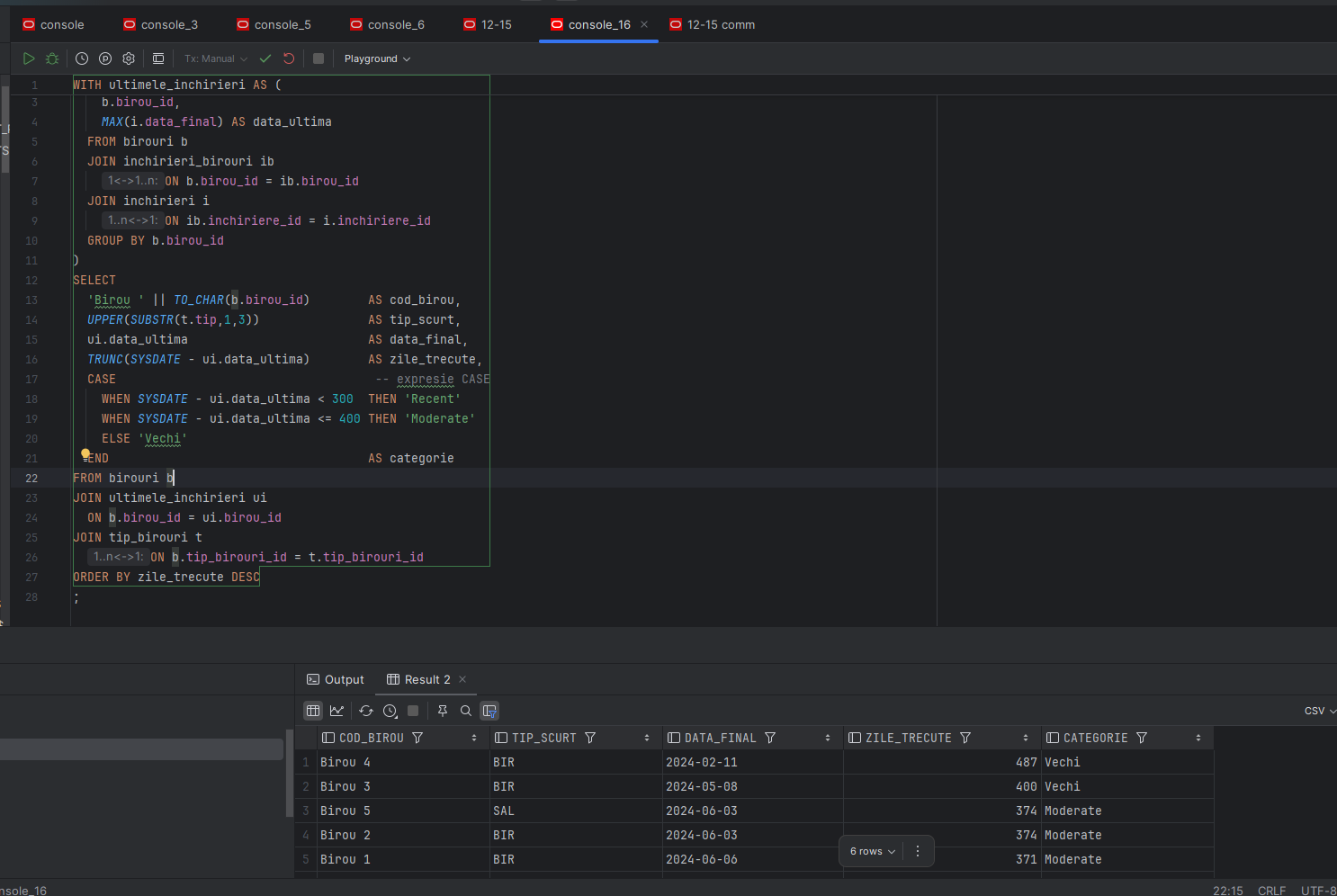
JOIN Inchirieri\_birouri ib ON p.Inchiriere\_ID = ib.Inchiriere\_ID

WHERE ib.Birou\_ID = b.Birou\_ID

AND strftime('%Y', p.Data\_plata) = '2024'

) AS NrPlati2024

FROM Birouri b;



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

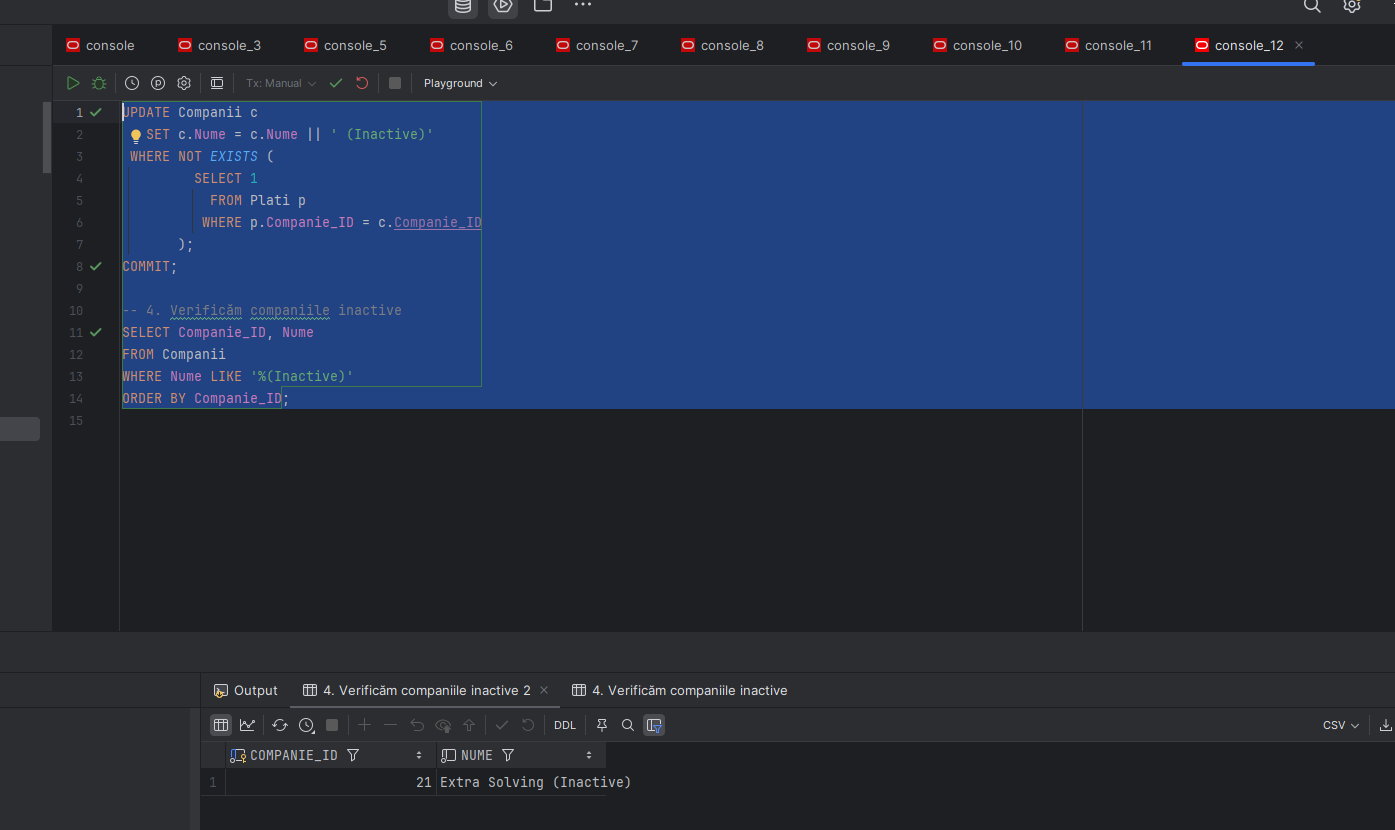
/\* ----------------------------------------------------------------

**Operațiunea 1: Marcarea companiilor inactive - UPDATE cu subcerere corelată (EXISTS)**

Aflăm companiile care nu au niciună plată înregistrată și adăugăm sufixul " (Inactive)" la câmpul Nume.

------------------------------------------------------------------ \*/

UPDATE Companii c  
 SET c.Nume = c.Nume || ' (Inactive)'  
 WHERE NOT *EXISTS* (  
 SELECT 1  
 FROM Plati p  
 WHERE p.Companie\_ID = c.Companie\_ID  
 );  
COMMIT;  
  
-- 4. Verificăm companiile inactive  
SELECT Companie\_ID, Nume  
FROM Companii  
WHERE Nume LIKE '%(Inactive)'  
ORDER BY Companie\_ID;



/\* ----------------------------------------------------------------

**Operațiunea 2: Majorare preț loc pentru tipuri populare. UPDATE cu subcerere nesincronizată în FROM**

Pentru fiecare tip de birou care a fost închiriat de mai mult de 5 ori (adică acel tip apare în tabelul INCHIRIERI\_BIROURI de mai mult de 5 ori), creștem prețul per loc cu 10%.

------------------------------------------------------------------ \*/

UPDATE tip\_birouri t

SET t.pret\_loc = t.pret\_loc \* 1.10

WHERE t.tip\_birouri\_id IN (

SELECT tip\_birouri\_id

FROM (

SELECT b.tip\_birouri\_id,

COUNT(\*) AS cnt

FROM birouri b

JOIN inchirieri\_birouri ib

ON b.birou\_id = ib.birou\_id

GROUP BY b.tip\_birouri\_id

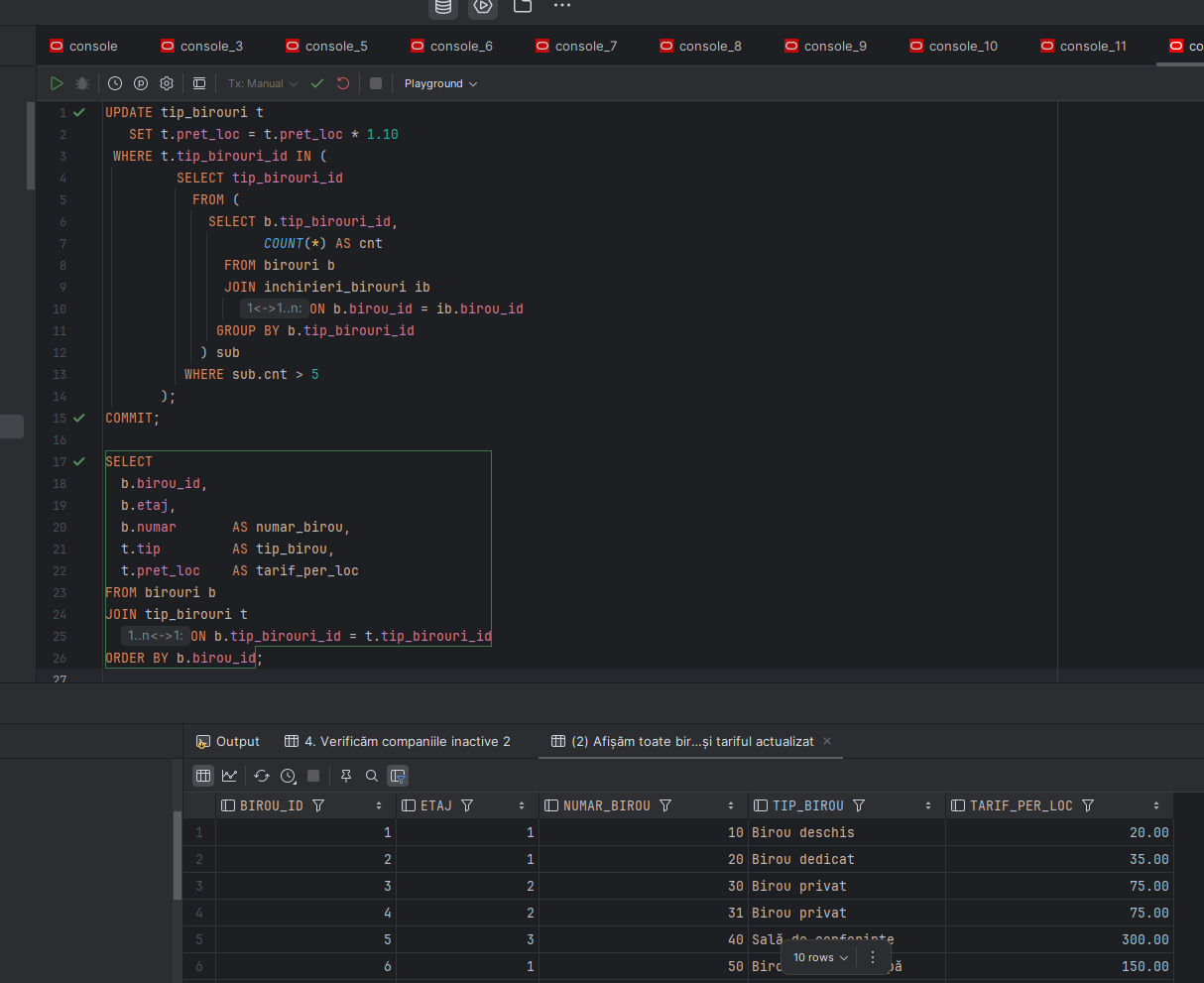
) sub

WHERE sub.cnt > 5

);

COMMIT;

SELECT  
 b.birou\_id,  
 b.etaj,  
 b.numar AS numar\_birou,  
 t.tip AS tip\_birou,  
 t.pret\_loc AS tarif\_per\_loc  
FROM birouri b  
JOIN tip\_birouri t  
 ON b.tip\_birouri\_id = t.tip\_birouri\_id  
ORDER BY b.birou\_id;



/\* ----------------------------------------------------------------

**Operațiunea 3: Ștergerea închirierilor vechi. DELETE cu subcerere (IN)**

Ștergem toate rândurile din INCHIRIERI\_BIROURI care au asociate închirieri finalizate înainte de 1 ianuarie 2024, apoi ștergem și înregistrările din INCHIRIERI.

------------------------------------------------------------------ \*/

-- 1) Ștergem asocierile birou–închiriere pentru închirieri finalizate înainte de 2024

DELETE FROM Inchirieri\_birouri ib

WHERE ib.Inchiriere\_ID IN (

SELECT i.Inchiriere\_ID

FROM Inchirieri i

WHERE i.Data\_final < DATE '2024-01-01'

);

-- 2) Arătăm starea după ștergerea asocierilor

SELECT \*

FROM Inchirieri\_birouri

ORDER BY Inchiriere\_ID, Birou\_ID, Utilizator\_ID;

--3) Ștergem închirierile finalizate înainte de 2024

DELETE FROM Inchirieri i

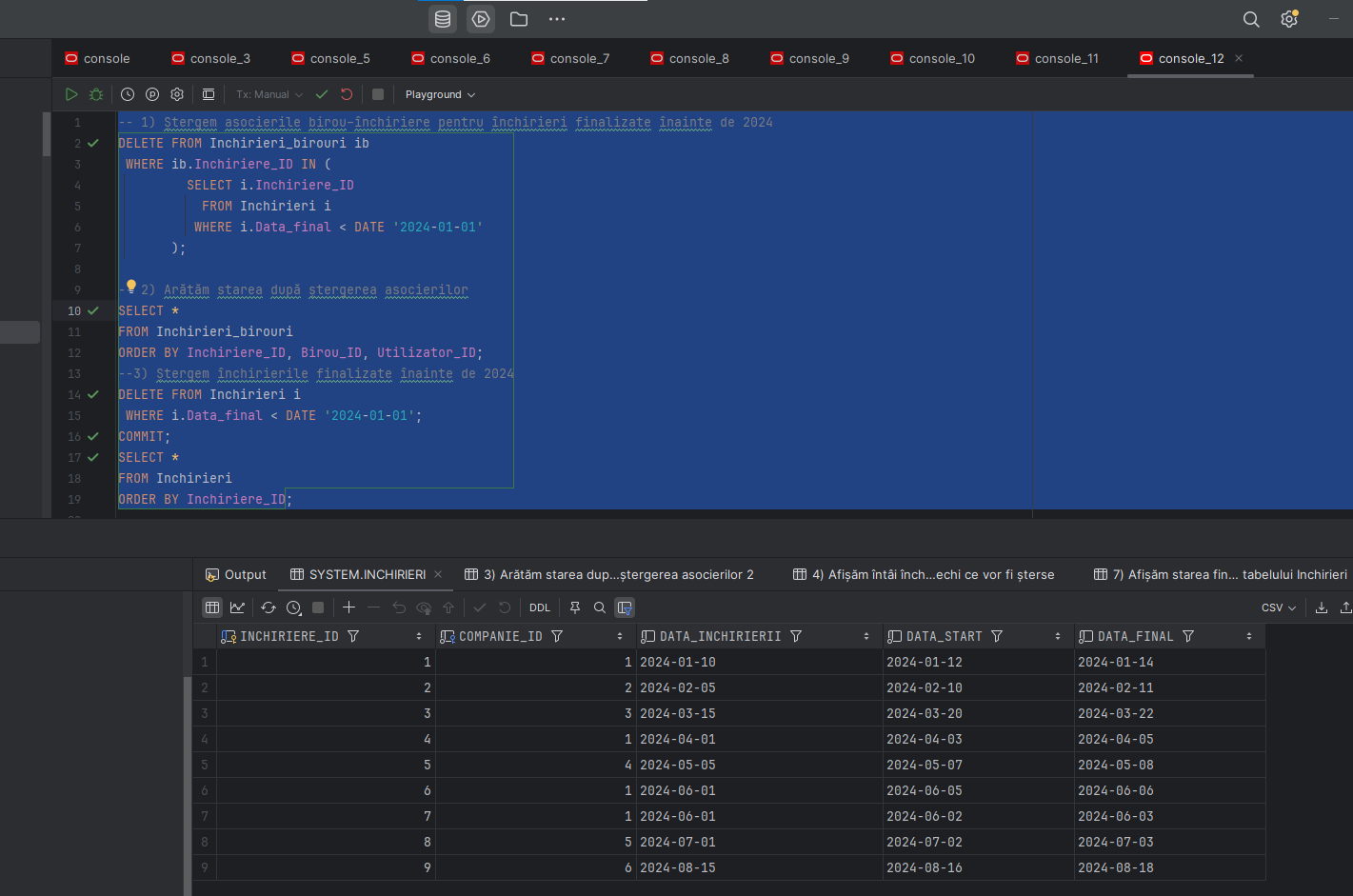
WHERE i.Data\_final < DATE '2024-01-01';

COMMIT;

SELECT \*

FROM Inchirieri

ORDER BY Inchiriere\_ID;



## 

## 14. Crearea unei vizualizări complexe

/\*Dați un exemplu de operație LMD permisă pe vizualizarea respectivă și un exemplu de operație LMD nepermisă.\*/

-- 1) Crearea unei view complexe

CREATE VIEW Vw\_BirouriDetaliate AS

SELECT

b.Birou\_ID,

c.Denumire AS NumeCladire,

t.Tip AS TipBirou,

nl.Numar\_Locuri,

-- coloană calculată

(t.Pret\_Loc \* nl.Numar\_Locuri) AS CostTotalLocuri

FROM Birouri b

JOIN Cladiri c ON b.Cladiri\_ID = c.Cladiri\_ID

JOIN Tip\_Birouri t ON b.Tip\_Birouri\_ID = t.Tip\_Birouri\_ID

JOIN Numar\_Locuri nl ON b.Numar\_Locuri\_ID = nl.Numar\_Locuri\_ID;

-- OPERATIE LMD permisa pe acest view

-- putem interoga (SELECT) liniştit

SELECT \*

FROM Vw\_BirouriDetaliate

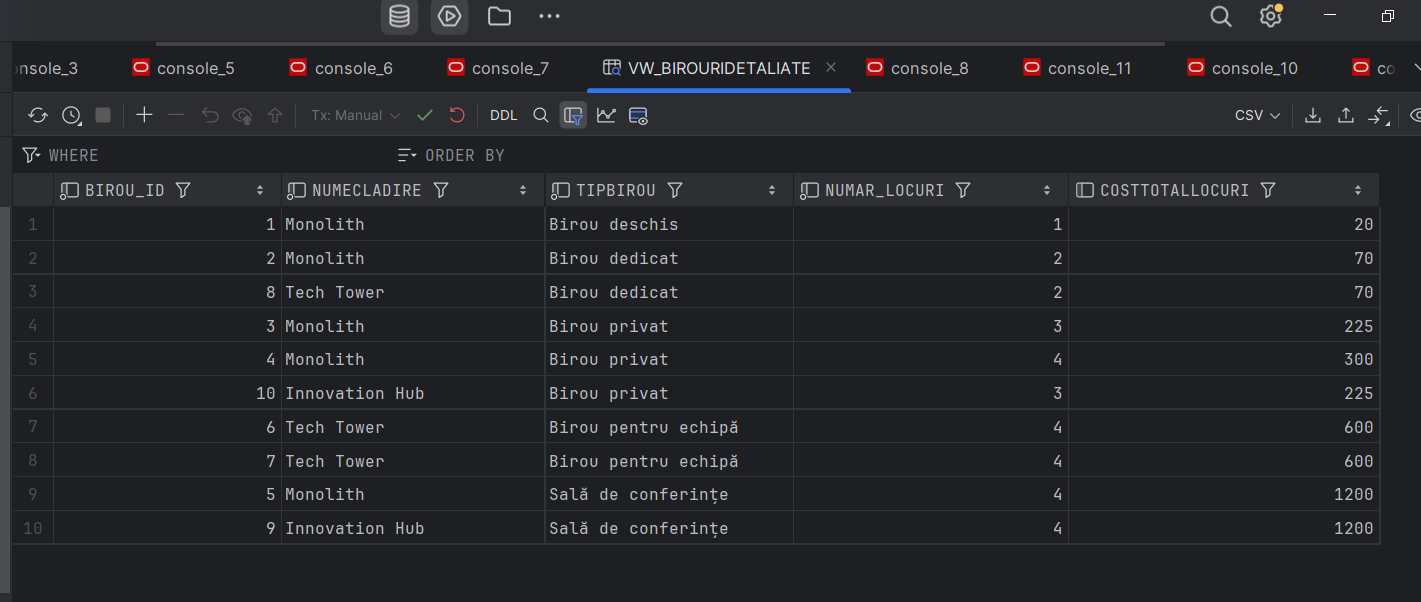
WHERE CostTotalLocuri < 200;

-- OPERATIE LMD nepermisa

-- încercare de INSERT, va eşua

INSERT INTO Vw\_BirouriDetaliate (Birou\_ID, NumeCladire, TipBirou, Numar\_Locuri, CostTotalLocuri)

VALUES (11, 'Skyline Plaza', 'Open Desk', 1, 20);



## 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*.

-- 1) OUTER JOIN pe minim 4 tabele

-- Scop: afișează toate birourile, cu facilități (dacă există),

-- tipul biroului (dacă e definit), numărul de locuri (dacă e definit)

-- și numele clădirii (necesar, deci INNER JOIN pe cladiri).

-- Notă: coloanele TIP\_BIROURI\_ID și NUMAR\_LOCURI\_ID din BIROURI trebuie să accepte NULL

SELECT

bir.birou\_id,

cld.denumire AS nume\_cladire,

bir.etaj,

bir.numar AS numar\_birou,

lfct.descriere\_facilitate,

tbir.tip,

nrl.numar\_locuri

FROM birouri bir

LEFT OUTER JOIN facilitati\_birouri fbir

ON fbir.birou\_id = bir.birou\_id

LEFT OUTER JOIN lista\_facilitati lfct

ON lfct.facilitate\_id = fbir.facilitate\_id

LEFT OUTER JOIN tip\_birouri tbir

ON tbir.tip\_birouri\_id = bir.tip\_birouri\_id

LEFT OUTER JOIN numar\_locuri nrl

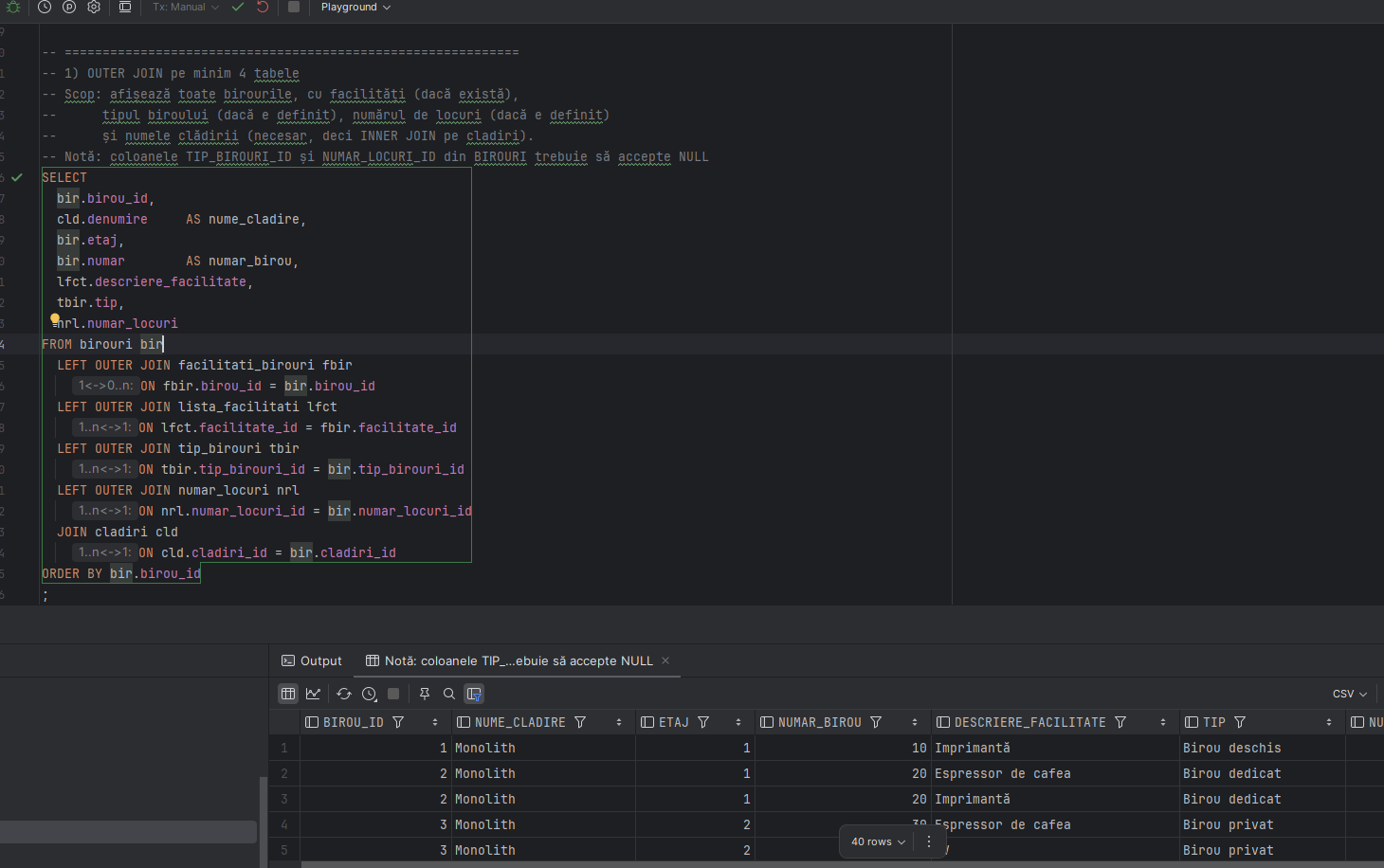
ON nrl.numar\_locuri\_id = bir.numar\_locuri\_id

JOIN cladiri cld

ON cld.cladiri\_id = bir.cladiri\_id -

ORDER BY bir.birou\_id

;



-- 2) DIVISION: tipuri de birouri care NU sunt folosite în nicio închiriere

-- Scop: găsește toate tipurile pentru care nu există niciun birou în inchirieri\_birouri

--nu merge sa scot check-ul si nu pot introduce tipuri noi de birouri

--dar am rulat in alt mediu fara checkuri si functioneaza

INSERT INTO Tip\_Birouri (Tip, Pret\_Loc)

VALUES ('Open Desk\_1', 200000);

COMMIT;

SELECT \*

FROM Tip\_Birouri tb

WHERE NOT EXISTS (

SELECT 1

FROM Inchirieri\_birouri ib

JOIN Birouri b

ON ib.Birou\_ID = b.Birou\_ID

WHERE b.Tip\_Birouri\_ID = tb.Tip\_Birouri\_ID

)

ORDER BY tb.Tip\_Birouri\_ID

;

-- 3) TOP-N (primele 4 cele mai mari sume de plată)

-- Scop: afișează top 4 plăți ordonate descrescător după sumă

-- (se poate adapta pentru paginare cu OFFSET…FETCH NEXT)

SELECT

p.plati\_id,

c.nume AS companie,

NVL(p.suma\_de\_plata, 0) AS suma,

DECODE(

mp.metoda\_plata,

'cash', 'Plata numerar',

'card', 'Plata cu card',

'transfer bancar', 'Plata prin transfer',

'Altă metodă'

) AS metoda\_prietenoasa,

p.data\_plata

FROM plati p

JOIN companii c

ON p.companie\_id = c.companie\_id

JOIN metoda\_de\_plata mp

ON p.metoda\_plata\_id = mp.metoda\_plata\_id

ORDER BY 3 DESC

FETCH FIRST 4 ROWS ONLY

; 