

# Baza de date

## “FitPro”

Proiect realizat de Fuciuc Filip-Luca,  
Grupa 134, Anul I

Cuprins

Pagina 3: 1. Descrierea modelului real, a utilitatii acestuia si a regulilor de functionare.

Pagina 5: 2. Prezentarea constrangerilor (restrictii, reguli) impuse asupra modelului.

Pagina 8: 3. Descrierea entitatilor, incluzand precizarea cheii primare.

Pagina 12: 4. Descrierea relatiilor, incluzand precizarea cardinalitatii acestora.

Pagina 15: 5. Descrierea atributelor, incluzand tipul de date si eventualele constrangeri, valori implice, valori posibile ale atributelor.

Pagina 19: 6. Realizarea diagramei entitate-relatie corespunzatoare descrierii de la punctele 3-5.

Pagina 20: 7. Realizarea diagramei conceptuale corespunzatoare diagramei entitate-relatie proiectate la punctul 6.

Pagina 21: 8. Enumerarea schemelor relationale corespunzatoare diagramei conceptuale proiectate la punctul 7.

Pagina 22: 9. Realizarea normalizarii pana la forma normala 3 (FN1-FN3).

Pagina 25: 10. Crearea unei sechete ce va fi utilizata in inserarea inregistrarilor in tabele (punctul 11).

Pagina 28: 11. Crearea tabelelor in SQL si inserarea de date coerente in fiecare dintre acestea.

Pagina 61: 12. Formulati in limbaj natural si implementati 5 cereri SQL complexe.

Pagina 75: 13. Implementarea a 3 operatii de actualizare si de suprimare a datelor utilizand subcereri.

Pagina 78: 14. Crearea unei vizualizari complexe. Dati un exemplu de operatie LMD permisa pe vizualizarea respectiva si un exemplu de operatie LMD nepermisa.

Pagina 82: 15. Formulati in limbaj natural si implementati in SQL: o cerere ce utilizeaza operatia outer-join pe minimum 4 tabele, o cerere ce utilizeaza operatia division si o cerere care implementeaza analiza top-n.

Pagina 85: 16. Optimizarea unei cereri, aplicand regulile de optimizare ce deriva din proprietatile operatorilor algebrei relationale. Cererea va fi exprimata prin expresie algebraica, arbore algebraic si limbaj (SQL), atat anterior cat si ulterior optimizarii

Pagina 90: 17. a. Realizarea normalizarii BCNF, FN4, FN5.

Pagina 93: 17. b. Aplicarea denormalizarii, justificand necesitatea acestuia

## 1. Descrierea modelului real, a utilitatii acestuia si a regulilor de functionare

### Tema

Tema acestui proiect este o baza de date realizata pentru o aplicatie mobila denumita "FitPro", destinata gestionarii activitatilor desfasurate in cadrul unei retele de sali de fitness. Aplicatia are ca scop monitorizarea participarii clientilor la cursuri, programarea antrenorilor, gestionarea echipamentelor din sali, administrarea abonamentelor si centralizarea feedback-ului primit de la clienti.

### Descrierea modelului real

Aceasta baza de date este proiectata pentru a reflecta activitatile desfasurate intr-o retea de sali de fitness si pentru a asigura o evidenta completa a interactiunilor dintre clienti, antrenori, sali si echipamente. In cadrul acestui sistem, fiecare client este identificat printr-un cod unic si poate beneficia de diferite servicii: achizitionarea de abonamente, participarea la cursuri de grup sau programarea unor sesiuni individuale cu antrenori personali.

Salile de fitness sunt gestionate in functie de locatia geografica, fiecare sala fiind asociata unui oras. Fiecare sala are o capacitate proprie si este dotata cu diverse echipamente, a caror stare si denumire sunt inregistrate in sistem.

Cursurile sunt activitati de grup desfasurate in sali, sub indrumarea unor antrenori. Antrenorii sunt clasificati in doua categorii: antrenori de curs, care predau lectii de grup si au un numar de ani de experienta asociat, si antrenori personali, care ofera sesiuni individuale de antrenament si au o specializare pe dezvoltarea masei musculare.

Programarile cursurilor stablesc detalii esentiale, precum data si ora programarii, cursul la care se face programarea si antrenorul desemnat pentru a sustine acel curs. Participarea clientilor la aceste cursuri este gestionata prin inregistrarea prezentei lor, astfel incat se poate urmari cine a participat la fiecare curs si cat de aglomerata a fost clasa.

Abonamentele achizitionate de clienti sunt stocate in baza de date, impreuna cu informatiile despre sala frecventata si perioada de valabilitate. Acest lucru permite urmarirea istoricului de participare al fiecarui client. De asemenea, clientii au posibilitatea de a oferi feedback antrenorilor prin mesaje care includ data si continutul evaluarii.

Sesiunile individuale sunt programate intre un client si un antrenor personal la o data si ora stabilita, oferind o experienta personalizata de antrenament. Toate aceste date sunt interconectate pentru a oferi o imagine clara si completa asupra activitatilor desfasurate in cadrul retelei de sali de fitness.

## Utilitatea

Aceasta baza de date este esentiala pentru a facilita o administrare eficienta a activitatilor dintr-o retea de sali de fitness. Ea permite planificarea atenta a cursurilor de grup si a sesiunilor individuale, monitorizarea participarii clientilor, gestionarea resurselor (sali, antrenori, echipamente) si urmarirea abonamentelor. De asemenea, prin colectarea feedback-ului, managerii pot evalua calitatea serviciilor oferite de antrenori si pot lua decizii informate pentru imbunatatirea experientei clientilor.

Baza de date ajuta la organizarea programului antrenorilor, la evidențierea gradului de ocupare a salilor, la verificarea starii echipamentelor si la generarea de rapoarte utile pentru management. Clientii beneficiaza de o planificare clara a activitatilor disponibile, iar antrenorii au o viziune completa asupra sesiunilor si feedback-ului primit.

## Functionalitati

Baza de date permite stocarea si gestionarea completa a informatiilor despre clienti, sali, echipamente, antrenori, cursuri, programari, sesiuni individuale, abonamente si feedback. Aceasta ofera suport pentru inregistrarea si urmarirea participarii clientilor la activitati, programarea antrenorilor in functie de specializare, gestionarea starii echipamentelor si a resurselor disponibile in sali, precum si evidențierea istoricului fiecarui client.

Prin aceasta organizare, aplicatia FitPro devine un instrument valoros pentru o retea de sali de fitness, asigurand un control complet asupra activitatilor si o experienta optima pentru clienti si personal.

## 2. Prezentarea constrangerilor (restrictii, reguli) impuse asupra modelului

### 1. Tabela CLIENT:

- id\_client este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din tabel.
- nume trebuie sa aiba o lungime maxima de 50 de caractere.
- varsta trebuie sa fie o valoare intreaga pozitiva.

## 2. Tabela ORAS:

- id\_oras este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din tabel.
- nume\_oras trebuie sa aiba o lungime maxima de 50 de caractere.
- judet trebuie sa aiba o lungime maxima de 50 de caractere.
- populatie trebuie sa fie o valoare intreaga pozitiva.

## 3. Tabela SALA:

- id\_sala este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din tabel.
- id\_oras este o cheie externa catre tabela Oras si trebuie sa se refere la un id\_oras existent.
- nume\_sala trebuie sa aiba o lungime maxima de 50 de caractere.
- adresa trebuie sa aiba o lungime maxima de 100 de caractere.
- capacitate trebuie sa fie o valoare intreaga pozitiva.

## 4. Tabela ECHIPAMENT:

- id\_echipament este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din tabel.
- id\_sala este o cheie externa catre tabela Sala si trebuie sa se refere la un id\_sala existent.
- stare trebuie sa fie unul dintre valorile permise, de exemplu: "buna", "avariat", avand o lungime maxima de 30 de caractere.
- denumire trebuie sa aiba o lungime maxima de 50 de caractere.

## 5. Tabela CURS:

- id\_curs este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din tabel.

- id\_sala este o cheie externa catre tabela Sala si trebuie sa se refere la un id\_sala existent.
- nivel\_dificultate trebuie sa fie unul dintre valorile permise, de exemplu: "incepator", "intermediar", "avansat".
- denumire trebuie sa aiba o lungime maxima de 50 de caractere.

## 6. Tabela ANTRENOR:

- id\_antrenor este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din tabel.
- nume trebuie sa aiba o lungime maxima de 50 de caractere.
- email trebuie sa aiba o lungime maxima de 50 de caractere.
- tip trebuie sa fie unul dintre valorile "curs" sau "personal".

## 7. Tabela ANTRENOR\_CURS:

- id\_antrenor este cheia primara si trebuie sa fie o cheie externa catre tabela Antrenor.
- ani\_experienta trebuie sa fie o valoare intreaga pozitiva.

## 8. Tabela ANTRENOR\_PERSONAL:

- id\_antrenor este cheia primara si trebuie sa fie o cheie externa catre tabela Antrenor.
- specializare\_musculatura trebuie sa aiba o lungime maxima de 50 de caractere.

## 9. Tabela PROGRAMARE\_CURS:

- id\_programare este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din tabel.
- id\_antrenor este o cheie externa catre tabela Antrenor si trebuie sa se refere la un id\_antrenor existent.
- id\_curs este o cheie externa catre tabela Curs si trebuie sa se refere la un id\_curs existent.
- nr\_locuri trebuie sa fie o valoare intreaga pozitiva.
- data trebuie sa fie de tip DATE.

- ora\_programare trebuie sa fie de tip varchar2(5) si retine ora la care incepe cursul.

## 10. Tabela COMPONENTA\_CLASA:

- id\_componenta este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din tabel.
- id\_client este o cheie externa catre tabela Client si trebuie sa se refere la un id\_client existent.
- id\_programare este o cheie externa catre tabela Programare\_curs si trebuie sa se refere la un id\_programare existent.
- prezenta\_efectiva trebuie sa fie o valoare intreaga pozitiva (asemenea unui camp boolean) (0 sau 1) care indica daca persoana a fost prezenta efectiv la curs.
- data\_confirmare trebuie sa fie de tip DATE.

## 11. Tabela TIP\_ABONAMENT:

- id\_tip\_abonament este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din tabel.
- pret trebuie sa aiba o valoare intreaga pozitiva.
- facilitati trebuie sa aiba o lungime maxima de 100 de caractere.

## 12. Tabela ISTORIC:

- id\_istoric este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din tabel.
- id\_client este o cheie externa catre tabela Client.
- id\_tip\_abonament este o cheie externa catre tabela Tip\_abonament.
- id\_sala este o cheie externa catre tabela Sala.
- data\_inceput si data\_expirare trebuie sa fie de tip DATE, cu data\_expirare mai mare decat data\_inceput.

## 13. Tabela FEEDBACK:

- id\_feedback este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din tabel.

- id\_antrenor este o cheie externa catre tabela Antrenor.
- id\_client este o cheie externa catre tabela Client.
- data trebuie sa fie de tip DATE.
- mesaj trebuie sa aiba o lungime maxima de 100 de caractere.

## 14. Tabela SESIUNE\_INDIVIDUALA:

- id\_sesiune este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din tabel.
- id\_antrenor este o cheie externa catre tabela Antrenor.
- id\_client este o cheie externa catre tabela Client.
- data trebuie sa fie de tip DATE.
- ora trebuie sa fie de tip varchar(10).

## 3. Descrierea entitatilor, incluzand precizarea cheii primare

### 1. Entitatea Client

Aceasta tabela contine informatii despre clientii care beneficiaza de serviciile salii de fitness. Fiecare inregistrare este identificata prin cheia primara id\_client si contine informatii despre numele clientului (nume) si varsta acestuia (varsta). Aceste informatii sunt utile pentru gestionarea relatiei cu clientii, personalizarea programelor de antrenament si monitorizarea participarii.

### 2. Entitatea Oras

Aceasta tabela stocheaza informatii despre orasele in care exista sali de fitness. Fiecare inregistrare este identificata prin cheia primara id\_oras si contine informatii despre numele orasului (nume\_oras), judetul in care este situat (judet) si populatia estimata (populatie). Aceste date sunt utile pentru a organiza si analiza distributia salilor in functie de locatie si potentialul de clienti.

### 3. Entitatea Sala

Aceasta tabela contine informatii despre salile de fitness disponibile in diverse orase. Fiecare inregistrare este identificata prin cheia primara id\_sala si este asociata cu un oras prin cheia externa id\_oras. De asemenea, tabela include detalii despre numele salii (nume\_sala), adresa (adresa) si capacitatea maxima (capacitate). Aceste informatii sunt esentiale pentru administrarea locatiilor si programarea activitatilor.

### 4. Entitatea Echipament

Aceasta tabela contine informatii despre echipamentele disponibile in cadrul salilor de fitness. Fiecare inregistrare este identificata prin cheia primara id\_echipament si este asociata cu o sala prin cheia externa id\_sala. Tabela include informatii despre denumirea echipamentului (denumire) si starea acestuia (stare - de exemplu, "buna", "avariat"). Aceste date sunt utile pentru gestionarea inventarului si intretinerea echipamentelor.

### 5. Entitatea Curs

Aceasta tabela stocheaza informatii despre cursurile organizate in salile de fitness. Fiecare inregistrare este identificata prin cheia primara id\_curs si este asociata cu o sala prin cheia externa id\_sala. Informatiile despre nivelul de dificultate al cursului (nivel\_dificultate - de exemplu, "incepator", "intermediar", "avansat") si denumirea cursului (denumire) sunt utile pentru a permite clientilor sa aleaga cursurile potrivite nivelului lor de pregatire.

### 6. Entitatea Antrenor

Aceasta tabela contine informatii despre antrenorii disponibili in cadrul salilor de fitness. Fiecare inregistrare este identificata prin cheia primara id\_antrenor si contine detalii despre numele antrenorului (nume), adresa de email (email) si tipul acestuia (tip - de exemplu, "curs" sau "personal"). Aceste informatii sunt utile pentru a gestiona echipa de antrenori si pentru a programa sesiunile de antrenament corespunzator.

### 7. Entitatea Antrenor\_curs

Aceasta tabela contine informatii suplimentare despre antrenorii care predau cursuri. Fiecare inregistrare este identificata prin cheia primara id\_antrenor, care este, de asemenea, o cheie externa catre tabela Antrenor. Coloana ani\_experienta indica vechimea antrenorului in domeniul cursurilor si este utila pentru a selecta antrenori in functie de experienta lor.

## 8. Entitatea Antrenor\_personal

Aceasta tabela contine informatii despre antrenorii care ofera sesiuni individuale. Fiecare inregistrare este identificata prin cheia primara id\_antrenor, care este si o cheie externa catre tabela Antrenor. Coloana specializare\_musculatura contine informatii despre specializarea antrenorului in tipurile de grupe musculare (de exemplu, "upper", "lower", "full-body"). Aceste date sunt utile pentru a recomanda clientilor antrenori potriviti nevoilor lor specifice.

## 9. Entitatea Programare\_curs

Aceasta tabela contine informatii despre programarile cursurilor disponibile in salile de fitness. Fiecare inregistrare este identificata prin cheia primara id\_programare. De asemenea, sunt definite chei externe catre tabelele Antrenor (id\_antrenor) si Curs (id\_curs). Coloanele suplimentare includ data, care indica ziua la care are loc cursul programat acum si ora\_programare, care indica ora de incepere a cursului, care in general are data si ore flexibile. Aceste informatii sunt esentiale pentru organizarea programarilor de cursuri si asigurarea desfasurarii eficiente a acestora.

## 10. Entitatea Componenta\_clasa

Aceasta tabela pastreaza evidenta participarii clientilor la cursuri. Fiecare inregistrare este identificata prin cheia primara id\_componenta. Tabela include chei externe catre Client (id\_client) si Programare\_curs (id\_programare). Coloanele suplimentare sunt prezenta\_efectiva (valoare intreaga 0/1 pentru a marca daca clientul a fost prezent sau nu) si data\_confirmare (data la care clientul a confirmat participarea). Aceste informatii permit monitorizarea prezentei la cursuri.

## 11. Entitatea Tip\_abonament

Aceasta tabela contine informatii despre tipurile de abonamente disponibile in cadrul salii de fitness. Fiecare inregistrare este identificata prin cheia primara id\_tip\_abonament. Tabela contine informatii despre pretul abonamentului (pret) si despre facilitatile incluse (facilitati). Aceste informatii sunt utile pentru a gestiona ofertele de abonamente ale salii si pentru a oferi clientilor detalii clare despre optiunile disponibile.

## 12. Entitatea Istoric

Aceasta tabela contine informatii despre istoricul abonamentelor fiecarui client. Fiecare inregistrare este identificata prin cheia primara id\_istoric. Tabela include chei externe catre Client (id\_client), Tip\_abonament (id\_tip\_abonament) si Sala (id\_sala). Informatiile despre perioada de valabilitate a abonamentului sunt retinute prin coloanele data\_inceput si data\_expirare. Aceasta tabela permite urmarirea utilizarii abonamentelor de catre clienti in diferite sali.

## 13. Entitatea Feedback

Aceasta tabela stocheaza feedback-ul oferit de clienti pentru antrenori. Fiecare inregistrare este identificata prin cheia primara id\_feedback. Tabela include chei externe catre Antrenor (id\_antrenor) si Client (id\_client). Informatiile despre feedback sunt completate prin coloanele data (data transmiterii feedback-ului) si mesaj (textul efectiv al feedback-ului). Aceste informatii sunt utile pentru evaluarea calitatii serviciilor oferite de antrenori.

## 14. Entitatea Sesiune\_individuala

Aceasta tabela pastreaza informatii despre sesiunile de antrenament individual. Fiecare inregistrare este identificata prin cheia primara id\_sesiune. Tabela include chei externe catre Antrenor (id\_antrenor) si Client (id\_client). Coloanele suplimentare sunt data (data sesiunii) si ora (ora de incepere a sesiunii). Aceste informatii sunt utile pentru programarea si monitorizarea sesiunilor individuale.

## 4. Descrierea relatiilor, incluzand precizarea cardinalitatii acestora

Tabela CLIENT are o relatie de tip "one-to-many" cu tabela ISTORIC, deoarece un client poate avea mai multe inregistrari in istoricul abonamentelor (sau niciuna), iar o inregistrare din ISTORIC apartine unui singur client (obligatoriu). Cardinalitatea acestei relatii este 1:M(0).

Tabela ORAS are o relatie de tip "one-to-many" cu tabela SALA, intrucat o sala este amplasata intr-un singur oras (obligatoriu), dar un oras poate avea mai multe sali (sau niciuna). Cardinalitatea acestei relatii este 1:M(0).

Tabela SALA are o relatie de tip "one-to-many" cu tabela ECHIPAMENT, deoarece un echipament este asociat unei singure sali (obligatoriu), iar o sala poate avea mai multe echipamente (sau niciunul). Cardinalitatea acestei relatii este 1:M(0).

Tabela SALA are o relatie de tip "one-to-many" cu tabela CURS, pentru ca un curs se desfasoara intr-o singura sala (obligatoriu), dar o sala poate gazdui mai multe cursuri (sau niciunul). Cardinalitatea acestei relatii este 1:M(0).

Tabela SALA are o relatie de tip "one-to-many" cu tabela ISTORIC, intrucat o inregistrare din ISTORIC este asociata unei singure sali (obligatoriu), iar o sala poate avea mai multe inregistrari (sau niciuna). Cardinalitatea acestei relatii este 1:M(0).

Tabela CURS are o relatie de tip "one-to-many" cu tabela PROGRAMARE\_CURS, deoarece o programare de curs apartine unui singur curs (obligatoriu), dar un curs poate avea mai multe programari (sau niciuna). Cardinalitatea acestei relatii este 1:M(0).

Tabela ANTRENOR are o relatie de tip "one-to-many" cu tabela PROGRAMARE\_CURS, intrucat o programare de curs este sustinuta de un singur antrenor (obligatoriu), dar un antrenor poate conduce mai multe programari (sau niciuna). Cardinalitatea acestei relatii este 1:M(0).

Tabela PROGRAMARE\_CURS are o relatie de tip "one-to-many" cu tabela COMPONENTA\_CLASA, deoarece fiecare componenta a clasei este asociata unei singure programari de curs (obligatoriu), iar o programare de curs poate avea mai multi participanti (sau niciunul). Cardinalitatea acestei relatii este 1:M(0).

Tabela CLIENT are o relatie de tip "one-to-many" cu tabela COMPONENTA\_CLASA, intrucat un client poate participa la mai multe cursuri programate (sau niciunul), dar o inregistrare din COMPONENTA\_CLASA apartine unui singur client (obligatoriu). Cardinalitatea acestei relatii este 1:M(0).

Tabela CLIENT are o relatie de tip "one-to-many" cu tabela FEEDBACK, pentru ca un client poate lasa mai multe feedbackuri (sau niciunul), iar fiecare feedback este asociat unui singur client (obligatoriu). Cardinalitatea acestei relatii este 1:M(0).

Tabela ANTRENOR are o relatie de tip "one-to-many" cu tabela FEEDBACK, deoarece un antrenor poate primi mai multe feedbackuri (sau niciunul), iar fiecare feedback este asociat unui singur antrenor (obligatoriu). Cardinalitatea acestei relatii este 1:M(0).

Tabela CLIENT are o relatie de tip "one-to-many" cu tabela SESIUNE\_INDIVIDUALA, deoarece un client poate avea mai multe sesiuni individuale (sau niciuna), iar o sesiune individuala apartine unui singur client (obligatoriu). Cardinalitatea acestei relatii este 1:M(0).

Tabela ANTRENOR are o relatie de tip "one-to-many" cu tabela SESIUNE\_INDIVIDUALA, intrucat un antrenor poate sustine mai multe sesiuni individuale (sau niciuna), dar o sesiune individuala apartine unui singur antrenor (obligatoriu). Cardinalitatea acestei relatii este 1:M(0).

Tabela TIP\_ABONAMENT are o relatie de tip "one-to-many" cu tabela ISTORIC, deoarece un tip de abonament poate fi asociat cu mai multe inregistrari din ISTORIC (sau niciuna), iar o inregistrare din ISTORIC apartine unui singur tip de abonament (obligatoriu). Cardinalitatea acestei relatii este 1:M(0).

Tabela ANTRENOR are o relatie de tip "one-to-one" cu tabela ANTRENOR\_CURS, intrucat fiecare antrenor poate avea cel mult o inregistrare specifica in ANTRENOR\_CURS (sau niciuna), iar o inregistrare din ANTRENOR\_CURS apartine unui singur antrenor (obligatoriu). Cardinalitatea acestei relatii este 1:1(0).

Tabela ANTRENOR are o relatie de tip "one-to-one" cu tabela ANTRENOR\_PERSONAL, deoarece fiecare antrenor poate avea cel mult o inregistrare specifica in ANTRENOR\_PERSONAL (sau niciuna), iar o inregistrare din ANTRENOR\_PERSONAL apartine unui singur antrenor (obligatoriu). Cardinalitatea acestei relatii este 1:1(0).

## 5. Descrierea atributelor, incluzand tipul de date si eventualele constrangeri, valori implicite, valori posibile ale atributelor

### 1. Tabela CLIENT:

- id\_client (int, PK): cheia primara a tabelului, identificator unic al fiecarui client.
- nume (varchar(50)): numele clientului, stocat ca sir de caractere cu lungimea maxima de 50.
- varsta (int): varsta clientului, stocata ca numar intreg, trebuie sa fie o valoare pozitiva.

### 2. Tabela ORAS:

- id\_oras (int, PK): cheia primara a tabelului, identificator unic al fiecarui oraș.
- nume\_oras (varchar(50)): numele orasului, stocat ca sir de caractere cu lungimea maxima de 255.
- judet (varchar(50)): numele judetului in care se afla orasul, sir de caractere cu lungimea maxima de 50.
- populatie (int): populatia orasului, stocata ca numar intreg pozitiv.

### 3. Tabela SALA:

- id\_sala (int, PK): cheia primara a tabelului, identificator unic al fiecarei sali.
- id\_oras (int, FK): cheie externa catre tabela ORAS, trebuie sa corespunda unui id\_oras existent.
- nume\_sala (varchar(50)): numele salii, sir de caractere cu lungimea maxima de 50.
- adresa (varchar(100)): adresa salii, sir de caractere cu lungimea maxima de 100.
- capacitate (int): capacitatea maxima a salii, stocata ca numar intreg pozitiv.

### 4. Tabela ECHIPAMENT:

- id\_echipament (int, PK): cheia primara a tabelului, identificator unic al fiecarui echipament.

- **id\_sala** (int, FK): cheie externa catre tabela SALA, trebuie sa corespunda unui id\_sala existent.
- **stare** (varchar(30)): starea echipamentului, cu valori posibile 'buna' sau 'avariat'.
- **denumire** (varchar(50)): denumirea echipamentului, sir de caractere cu lungimea maxima de 50.

## 5. Tabela CURS:

- **id\_curs** (int, PK): cheia primara a tabelului, identificator unic al fiecarui curs.
- **id\_sala** (int, FK): cheie externa catre tabela SALA, trebuie sa corespunda unui id\_sala existent.
- **nivel\_dificultate** (varchar(20)): nivelul de dificultate al cursului, cu valori posibile 'incepator', 'intermediar' sau 'avansat'.
- **denumire** (varchar(50)): denumirea cursului, sir de caractere cu lungimea maxima de 50.

## 6. Tabela ANTRENOR:

- **id\_antrenor** (int, PK): cheia primara a tabelului, identificator unic al fiecarui antrenor.
- **nume** (varchar(50)): numele antrenorului, sir de caractere cu lungimea maxima de 50.
- **email** (varchar(50)): adresa de email a antrenorului, sir de caractere cu lungimea maxima de 50.
- **tip** (varchar(20)): tipul antrenorului, poate fi 'curs' sau 'personal'.

## 7. Tabela ANTRENOR\_CURS:

- **id\_antrenor** (int, PK, FK): cheia primara a tabelului si cheie externa catre tabela ANTRENOR, trebuie sa corespunda unui id\_antrenor existent.
- **ani\_experienta** (int): numarul de ani de experienta al antrenorului, stocat ca numar intreg pozitiv.

## 8. Tabela ANTRENOR\_PERSONAL:

- **id\_antrenor** (int, PK, FK): cheia primara a tabelului si cheie externa catre tabela ANTRENOR, trebuie sa corespunda unui id\_antrenor existent.

- `specializare_musculatura` (varchar(50)): specializarea antrenorului personal, sir de caractere cu lungimea maxima de 50.

## 9. Tabela PROGRAMARE\_CURS:

- `id_programare` (int, PK): cheia primara a tabelului, identificator unic al fiecarei programari.
- `id_antrenor` (int, FK): cheie externa catre tabela ANTRENOR, trebuie sa corespunda unui `id_antrenor` existent.
- `id_curs` (int, FK): cheie externa catre tabela CURS, trebuie sa corespunda unui `id_curs` existent.
- `data` (date): data programarii, stocata ca tip de data.
- `ora_programare` (varchar2(5)): ora de incepere a programarii, stocata ca varchar2(5).

## 10. Tabela COMPONENTA\_CLASA:

- `id_componenta` (int, PK): cheia primara a tabelului, identificator unic al fiecarei inregistrari.
- `id_client` (int, FK): cheie externa catre tabela CLIENT, trebuie sa corespunda unui `id_client` existent.
- `id_programare` (int, FK): cheie externa catre tabela PROGRAMARE\_CURS, trebuie sa corespunda unui `id_programare` existent.
- `prezenta_efectiva` (number(1)): indica daca un client a fost prezent efectiv la curs (1 = present, 0 = absent).
- `data_confirmare` (date): data confirmarii prezentei, stocata ca tip de data.

## 11. Tabela TIP\_ABONAMENT:

- `id_tip_abonament` (int, PK): cheia primara a tabelului, identificator unic al fiecarui tip de abonament.
- `pret` (int): pretul tipului de abonament, exprimat in lei, stocat ca numar intreg pozitiv.
- `facilitati` (varchar(100)): lista facilitatilor incluse in abonament, stocata ca sir de caractere cu lungimea maxima de 100.

## 12. Tabela ISTORIC:

- id\_istoric (int, PK): cheia primara a tabelului, identificator unic al fiecarei inregistrari.
- id\_client (int, FK): cheie externa catre tabela CLIENT, trebuie sa corespunda unui id\_client existent.
- id\_tip\_abonament (int, FK): cheie externa catre tabela TIP\_ABONAMENT, trebuie sa corespunda unui id\_tip\_abonament existent.
- id\_sala (int, FK): cheie externa catre tabela SALA, trebuie sa corespunda unui id\_sala existent.
- data\_inceput (date): data de inceput a perioadei de valabilitate a abonamentului, stocata ca tip de data.
- data\_expirare (date): data de expirare a abonamentului, stocata ca tip de data.

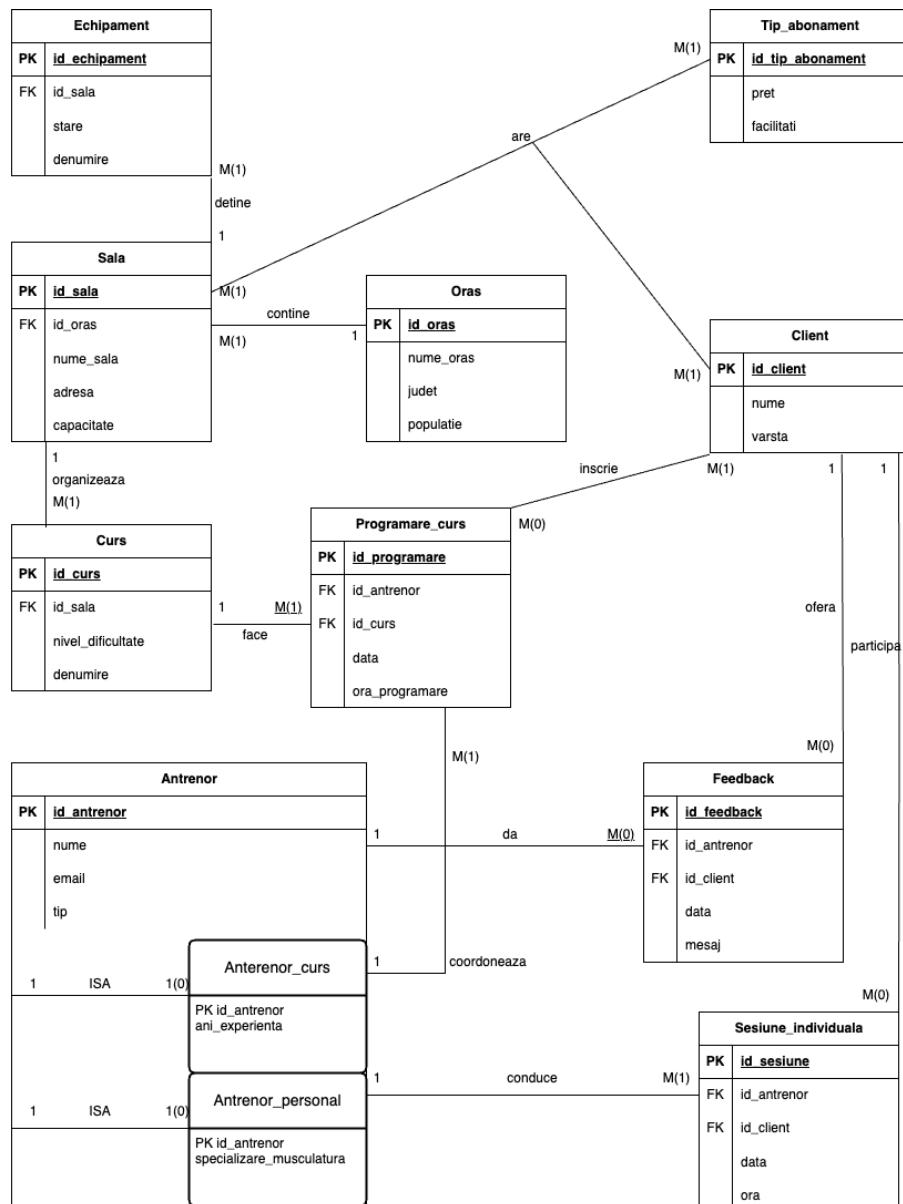
### 13. Tabela FEEDBACK:

- id\_feedback (int, PK): cheia primara a tabelului, identificator unic al fiecarui feedback.
- id\_antrenor (int, FK): cheie externa catre tabela ANTRENOR, trebuie sa corespunda unui id\_antrenor existent.
- id\_client (int, FK): cheie externa catre tabela CLIENT, trebuie sa corespunda unui id\_client existent.
- data (date): data la care a fost oferit feedback-ul, stocata ca tip de data.
- mesaj (varchar(100)): mesajul/descrierea feedback-ului, stocat ca sir de caractere cu lungimea maxima de 100.

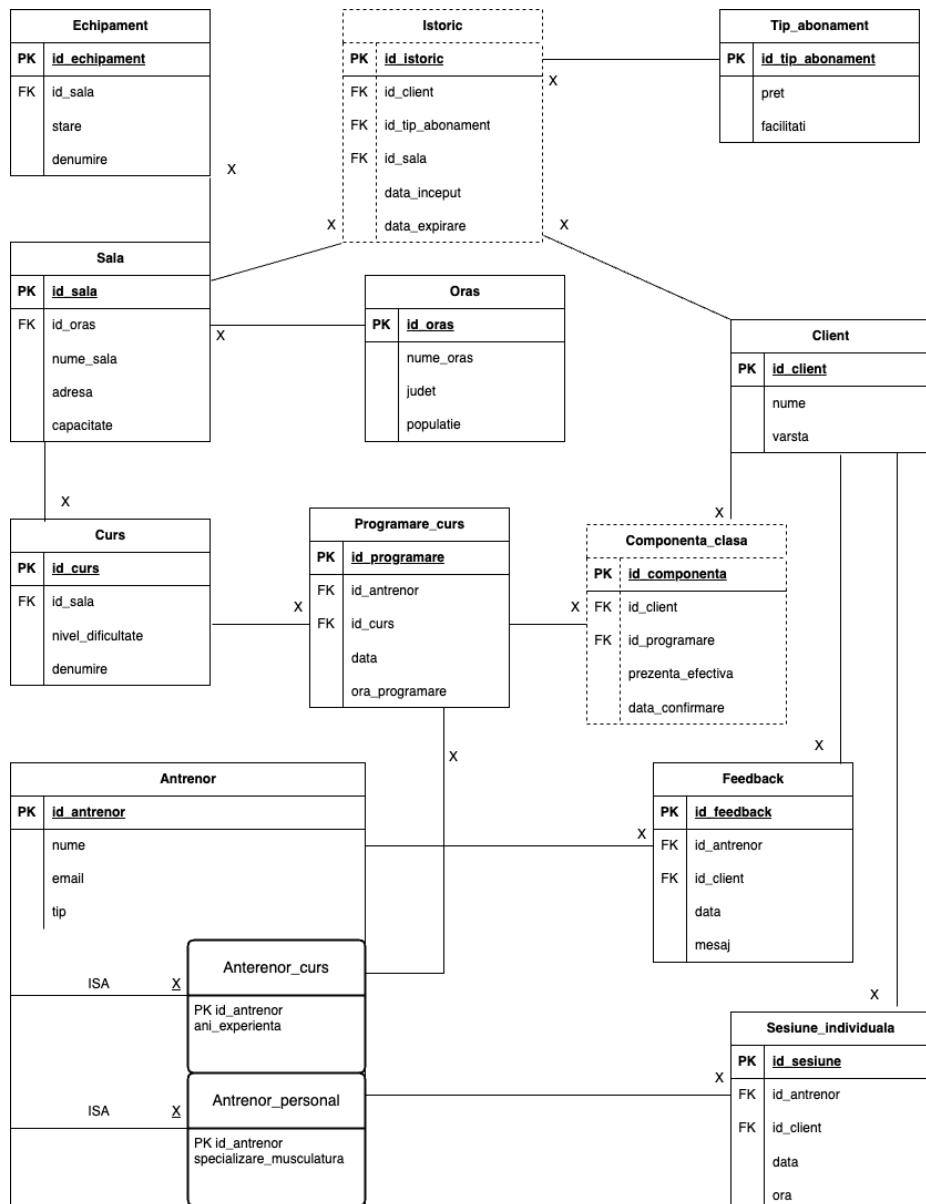
### 14. Tabela SESIUNE\_INDIVIDUALA:

- id\_seсиune (int, PK): cheia primara a tabelului, identificator unic al fiecarei sesiuni individuale.
- id\_antrenor (int, FK): cheie externa catre tabela ANTRENOR, trebuie sa corespunda unui id\_antrenor existent.
- id\_client (int, FK): cheie externa catre tabela CLIENT, trebuie sa corespunda unui id\_client existent.
- data (date): data programarii sesiunii, stocata ca tip de data.
- ora (varchar(10)): ora programarii sesiunii, stocata ca tip varchar.

## 6. Realizarea diagramei entitate-relatie corespunzatoare descrierii de la punctele 3-5



## 7. Realizarea diagramei conceptuale corespunzatoare diagramei entitate-relatie proiectate la punctul 6



## 8. Enumerarea schemelor relationale corespunzatoare diagramei conceptuale proiectate la punctul 7

CLIENT (id\_client PK, nume, varsta)

ORAS (id\_oras PK, nume\_oras, judet, populatie)

SALA (id\_sala PK, id\_oras FK, nume\_sala, adresa, capacitate)

ECHIPAMENT (id\_echipament PK, id\_sala FK, stare, denumire)  
CURS (id\_curs PK, id\_sala FK, nivel\_dificultate, denumire)  
ANTRENOR (id\_antrenor PK, nume, email, tip)  
ANTRENOR\_CURS (id\_antrenor PK, ani\_experienta)  
ANTRENOR\_PERSONAL (id\_antrenor PK, specializare\_musculatura)  
PROGRAMARE\_CURS (id\_programare PK, id\_antrenor FK, id\_curs FK, data, ora\_programare)  
COMPONENTA\_CLASA (id\_componenta PK, id\_client FK, id\_programare FK, prezenta\_efectiva, data\_confirmare)  
TIP\_ABONAMENT (id\_tip\_abonament PK, pret, facilitati)  
ISTORIC(id\_istoric PK, id\_client FK, id\_tip\_abonament FK, id\_sala FK, data\_inceput, data\_expirare)  
FEEDBACK (id\_feedback PK, id\_antrenor FK, id\_client FK, data, mesaj)  
SESIUNE\_INDIVIDUALA (id\_sesiune PK, id\_antrenor FK, id\_client FK, data, ora)

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

FN1:

O relație se află în Forma Normală 1 (FN1) dacă toate atributele acesteia contin doar valori atomice (n DIVIZIBILE), iar fiecare camp conține o singură valoare.

În cadrul bazei mele de date, voi utiliza un exemplu determinat de tabelele SALA și ORAS pentru a prezenta aflarea acestora în forma normală 1.

În cazul unui oraș, este probabil să existe mai multe săli diferite:

Id_oras	Nume_oras	Id_sala
---------	-----------	---------

101	Bucuresti	1,2,3
102	Iasi	4,5

Pentru a corecta aceasta greseala, int entitatea Oras, care determina corespondenta dintre sala si orasul careia ii apartine, separam pe cate un rand fiecare id\_sala. Astfel, ne vom incadra in FN1:

Id_oras	Nume_oras	Id_sala
101	Bucuresti	1
101	Bucuresti	2
101	Bucuresti	3
102	Iasi	4
102	Iasi	5

## FN2:

O relatie se afla in FN2 daca si numai daca aceasta relatie este deja in FN1, iar fiecare atribut care nu este cheie primara este dependent de intreaga cheie primara. Astfel, stim ca FN2 presupune sa nu existe dependente functionale partiale in cadrul relatiei.

In cazul acestui model, se respecta principiile FN2, dar voi prezinta o situatie in care nu ar fi respectat si modul in care ar fi trebuit sa facem modificari pentru a corecta eroarea.

Iau de exemplu o entitate SALA\_ORAS, care ar avea drept atribute Id\_sala, Nume\_sala, Id\_oras si Nume\_oras.

Id_sala	Nume_sala	Id_oras	Nume_oras
1	UltraGym	71	Timisoara
2	BodyFit	71	Timisoara
3	BodyLine	72	Arad
4	Energym	72	Arad

In cazul acesta, Nume\_sala nu este cheie primara si trebuie sa depinda de intreaga cheie primara (compusa din Id\_sala si Id\_oras). Insa acest lucru nu se intampla, deoarece acest atribut nu depinde direct de intreaga cheie primara, observandu-se dependenta directa doar dintre Id\_sala si Nume\_sala. Analog, se remarca si dependenta directa dintre Id\_oras si Nume\_oras. Asadar, in aceasta forma, entitatea nu se afla in FN2.

Pentru a corecta gresela, aplicam regula Casey Delobel si observam ca atributul Nume\_oras trebuie sa apartina doar tableei SALA, iar Nume\_oras doar tableei ORAS. Vom separa in doua entitati separate, iar tabela SALA va avea drept cheie secundara Id\_oras:

Tabela SALA:

Id_sala	Nume_sala	Id_oras
1	UltraGym	71
2	BodyFit	71
3	BodyLine	72
4	Energym	72

Tabela ORAS:

Id_oras	Nume_oras
71	Timisoara
71	Timisoara
72	Arad
72	Arad

### FN3:

O relatie se afla in FN3 daca si numai daca aceasta relatie este deja in FN2 (si implicit in FN1), iar fiecare atribut care nu este cheie primara este dependent de cheia primara.

Pentru a exemplifica acest lucru in cadrul bazei mele de date, voi lua ca exemplu situatia tableei SALA. Initial aceasta avea ca atribute Id\_sala, Nume\_sala, Capacitate Nume\_oras si Populatie.

Id_sala	Nume_sala	Capacitate	Nume_oras	Populatie
1	UltraGym	170	Constanta	263 000
2	BodyFit	50	Constanta	263 000
3	BodyLine	85	Tulcea	65 000
4	Energym	40	Tulcea	65 000

Atributele Nume\_sala si Capacitate depind in mod direct de cheia primara Id\_sala. Cu toate acestea, atributul Populatie nu este cheie primara si totodata nu depinde in mod direct de cheia primara, acesta avand o dependenta directa fata de un alt atribut din această tabela, Nume\_oras

Pentru a aduce aceasta relatie in FN3, voi separa atributele legate de oras de tabela SALA si le voi introduce in alta tabela, numita ORAS. In cadrul tablei SALA, vom inlocui informatiile legate de oras cu atributul id\_oras pentru a face legatura intre cele doua entitati si va lua nastere o relatie de tip one-to-many (un oras poate avea mai multe sali, dar o sala apartine unui singur oras).

Tabela SALA:

Id_sala	Nume_sala	Capacitate	Id_oras
1	UltraGym	170	81
2	BodyFit	50	81
3	BodyLine	85	82
4	Energym	40	82

Id_oras	Nume_oras	Populatie
81	Constanta	263 000
81	Constanta	263 000
82	Tulcea	65 000
82	Tulcea	65 000

## 10. Crearea unei secvente ce va fi utilizata in inserarea inregistrarilor in table (punctul 11)

CREATE SEQUENCE CLIENT\_SEQ START WITH 1;

CREATE SEQUENCE ORAS\_SEQ START WITH 1;

CREATE SEQUENCE SALA\_SEQ START WITH 1;

CREATE SEQUENCE ECHIPAMENT\_SEQ START WITH 1;

CREATE SEQUENCE CURS\_SEQ START WITH 1;

CREATE SEQUENCE ANTRENOR\_SEQ START WITH 1;

CREATE SEQUENCE PROGRAMARE\_SEQ START WITH 1;

```
CREATE SEQUENCE COMPONENTA_SEQ START WITH 1;
```

```
CREATE SEQUENCE TIP_ABONAMENT_SEQ START WITH 1;
```

```
CREATE SEQUENCE ISTORIC_SEQ START WITH 1;
```

```
CREATE SEQUENCE FEEDBACK_SEQ START WITH 1;
```

```
CREATE SEQUENCE SESIUNE_SEQ START WITH 1;
```

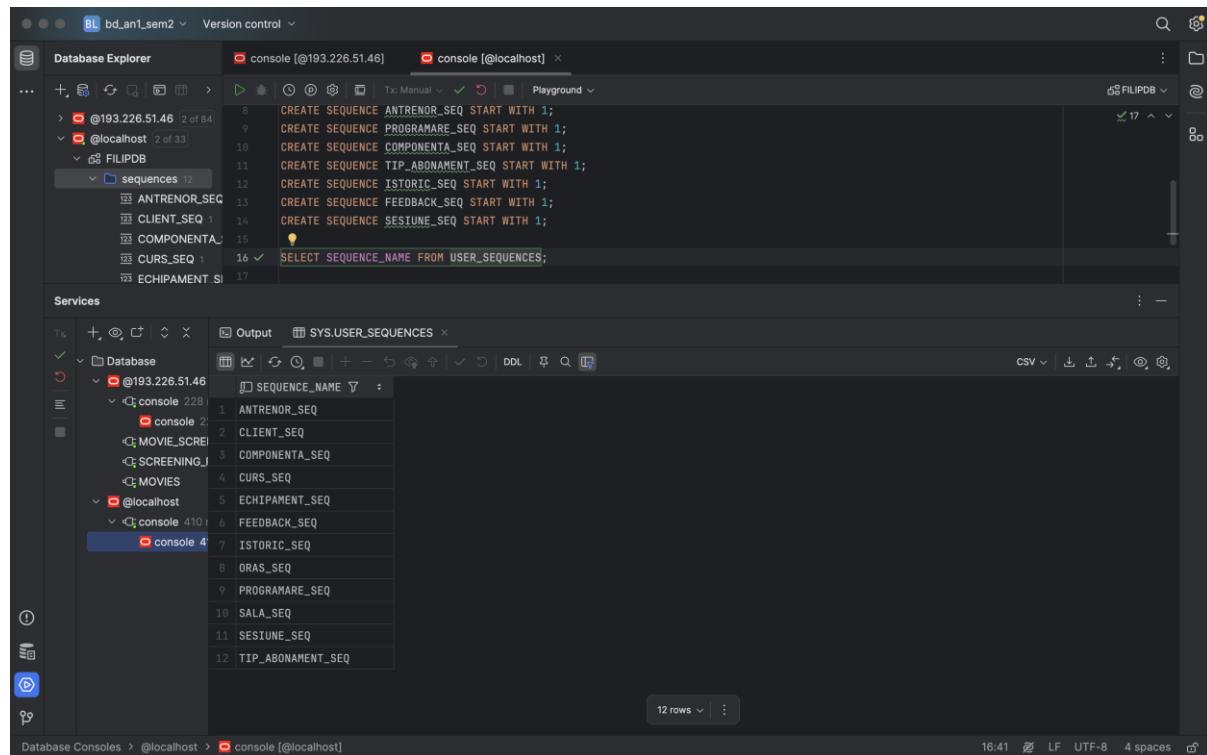
PrintScreen cu codul sursa:

The screenshot shows a database management interface with the following details:

- Database:** bd\_an1\_sem2
- Console:** console [localhost] (Tx: Manual)
- Object Tree:** FILIPDB > sequences (12 objects listed)
- Code Area:** SQL code for sequence creation:

```
--Crearea unei secvențe ce va fi utilizată în inserarea înregistrărilor în tabele (punctul 1)
1 CREATE SEQUENCE CLIENT_SEQ START WITH 1;
2 CREATE SEQUENCE DRAS_SEQ START WITH 1;
3 CREATE SEQUENCE SALA_SEQ START WITH 1;
4 CREATE SEQUENCE ECHIPAMENT_SEQ START WITH 1;
5 CREATE SEQUENCE CURS_SEQ START WITH 1;
6 CREATE SEQUENCE ANTRENOR_SEQ START WITH 1;
7 CREATE SEQUENCE PROGRAMARE_SEQ START WITH 1;
8 CREATE SEQUENCE COMPONENTA_SEQ START WITH 1;
9 CREATE SEQUENCE TIP_ABONAMENT_SEQ START WITH 1;
10 CREATE SEQUENCE ISTORIC_SEQ START WITH 1;
11 CREATE SEQUENCE FEEDBACK_SEQ START WITH 1;
12 CREATE SEQUENCE SESIUNE_SEQ START WITH 1;
13 CREATE SEQUENCE ORAS_SEQ START WITH 1;
14 CREATE SEQUENCE ECHIPAMENT_SEQ START WITH 1;
```
- Services:** Shows recent connections and their execution times.

Apoi, folosind comanda SELECT SEQUENCE\_NAME FROM USER\_SEQUENCES vom putea vizualiza secvențele create:



## 11. Crearea tablelor in SQL si inserarea de date coerente in fiecare dintre acestea

### 1. Tabela CLIENT

```
CREATE TABLE CLIENT (
    id_client INT DEFAULT CLIENT_SEQ.NEXTVAL PRIMARY KEY,
    nume VARCHAR(50),
    varsta INT
);
```

```
INSERT INTO CLIENT (nume, varsta) VALUES ('Popescu Andrei', 28);
```

```
INSERT INTO CLIENT (nume, varsta) VALUES ('Ionescu Maria', 35);
```

```
INSERT INTO CLIENT (nume, varsta) VALUES ('Vasilescu Elena', 24);
```

```
INSERT INTO CLIENT (nume, varsta) VALUES ('Georgescu Mihai', 30);
```

```
INSERT INTO CLIENT (nume, varsta) VALUES ('Dumitru Ana', 40);
```

```
INSERT INTO CLIENT (nume, varsta) VALUES ('Radu Cristian', 22);

INSERT INTO CLIENT (nume, varsta) VALUES ('Ilie Gabriela', 31);

INSERT INTO CLIENT (nume, varsta) VALUES ('Mihailescu Tudor', 27);

INSERT INTO CLIENT (nume, varsta) VALUES ('Toma Irina', 33);

INSERT INTO CLIENT (nume, varsta) VALUES ('Preda Sorin', 29);

SELECT *
FROM CLIENT
ORDER BY id_client
```

PrintScreen cu rezultatul:

The screenshot shows a database management interface with the following details:

- Left Panel (Object Explorer):** Shows the database structure:
  - Tables: CLIENT (selected), PUBLIC, Server Objects.
  - Sequences: sequences (12).
- Middle Panel (Console):** Displays the SQL code used to create the table and insert data.

```
--Crearea tabelelor si inserarea datelor
CREATE TABLE CLIENT (
    id_client INT DEFAULT CLIENT_SEQ.NEXTVAL PRIMARY KEY,
    nume VARCHAR(50),
    varsta INT
);

INSERT INTO CLIENT (nume, varsta) VALUES ('Popescu Andrei', 28);
INSERT INTO CLIENT (nume, varsta) VALUES ('Ionescu Maria', 35);
INSERT INTO CLIENT (nume, varsta) VALUES ('Vasilescu Elena', 24);
INSERT INTO CLIENT (nume, varsta) VALUES ('Georgescu Mihai', 39);
INSERT INTO CLIENT (nume, varsta) VALUES ('Dumitru Ana', 40);
INSERT INTO CLIENT (nume, varsta) VALUES ('Radu Cristian', 22);
INSERT INTO CLIENT (nume, varsta) VALUES ('Ilie Gabriela', 31);
INSERT INTO CLIENT (nume, varsta) VALUES ('Mihailescu Tudor', 27);
INSERT INTO CLIENT (nume, varsta) VALUES ('Toma Irina', 33);
INSERT INTO CLIENT (nume, varsta) VALUES ('Preda Sorin', 29);

SELECT *
FROM CLIENT
ORDER BY id_client
```
- Bottom Panel (Output):** Shows the resulting data in a grid format.

ID_CLIENT	NUME	VARSTA
1	Popescu Andrei	28
2	Ionescu Maria	35
3	Vasilescu Elena	24

## 2. Tabela ORAS

```
CREATE TABLE ORAS(
    id_oras INT DEFAULT ORAS_SEQ.NEXTVAL PRIMARY KEY,
```

```
nume_oras VARCHAR(50),
judet VARCHAR(50),
populatie INT
);

INSERT INTO ORAS (nume_oras, judet, populatie) VALUES ('Bucuresti', 'Ilfov', 1800000);

INSERT INTO ORAS (nume_oras, judet, populatie) VALUES ('Cluj-Napoca', 'Cluj', 300000);

INSERT INTO ORAS (nume_oras, judet, populatie) VALUES ('Iasi', 'Iasi', 290000);

INSERT INTO ORAS (nume_oras, judet, populatie) VALUES ('Timisoara', 'Timis', 320000);

INSERT INTO ORAS (nume_oras, judet, populatie) VALUES ('Constanta', 'Constanta', 280000);

SELECT *
FROM ORAS
ORDER BY id_oras;
```

PrintScreen cu rezultatul:

The screenshot shows the DataGrip IDE interface. On the left, the Database Explorer pane displays the database structure, including tables like CLIENT and ORAS. The main pane shows a code editor with the following SQL script:

```
50 ORDER BY id_client;

51
52
53
54
55
56 CREATE TABLE ORAS(
57     id_oras INT DEFAULT ORAS_SEQ.NEXTVAL PRIMARY KEY,
58     nume_oras VARCHAR(50),
59     judet VARCHAR(50),
60     populatie INT
61 );
62
63
64 INSERT INTO ORAS (nume_oras, judet, populatie) VALUES ('Bucuresti', 'Ilfov', 1800000);
65
66 INSERT INTO ORAS (nume_oras, judet, populatie) VALUES ('Cluj-Napoca', 'Cluj', 300000);
67
68 INSERT INTO ORAS (nume_oras, judet, populatie) VALUES ('Iasi', 'Iasi', 290000);
69
70 INSERT INTO ORAS (nume_oras, judet, populatie) VALUES ('Timisoara', 'Timis', 320000);
71
72 INSERT INTO ORAS (nume_oras, judet, populatie) VALUES ('Constanta', 'Constanta', 280000);
73
74
75 ✓ SELECT *
76 FROM ORAS
77 ORDER BY id_oras;
```

The screenshot shows the DBeaver application interface. At the top, there's a navigation bar with tabs for 'Database Explorer', 'console [@193.226.51.46]', and 'console [@localhost]'. Below the navigation bar is a 'Playground' section containing several SQL statements. The 'Database Explorer' panel on the left lists databases, tables, and sequences. The 'Services' panel at the bottom shows a table named 'FILIPDB.CLIENT' with 10 rows of data.

ID_CLIENT	NUME	VARSTA
1	Popescu Andrei	28
2	Ionescu Maria	35
3	Vasilescu Elena	24
4	Georgescu Mihai	30
5	Dumitru Ana	40
6	Radu Cristian	22
7	Ilie Gabriela	31
8	Mihăilescu Tudor	27
9	Toma Irina	33
10	Preda Sorin	29

### 3. Tabela SALA

```
CREATE TABLE SALA(
    id_sala INT DEFAULT SALA_SEQ.NEXTVAL PRIMARY KEY,
    id_oras INT,
    nume_sala VARCHAR(50),
    adresa VARCHAR(100),
    capacitate INT,
    FOREIGN KEY (id_oras) REFERENCES ORAS(id_oras)
);

INSERT INTO SALA (id_oras, nume_sala, adresa, capacitate) VALUES (1, 'Fitness Arena',
    'Strada Muncii 23', 120);

INSERT INTO SALA (id_oras, nume_sala, adresa, capacitate) VALUES (2, 'Body Shape',
    'Bd. Eroilor 10', 100);

INSERT INTO SALA (id_oras, nume_sala, adresa, capacitate) VALUES (3, 'FitLife',
    'Str. Libertatii 45', 90);

INSERT INTO SALA (id_oras, nume_sala, adresa, capacitate) VALUES (4, 'GymPro',
    'Str. Independentei 15', 110);
```

```
INSERT INTO SALA (id_oras, nume_sala, adresa, capacitate) VALUES (5, 'ActiveZone',  
                                'Bd. Tomis 200', 85);
```

```
SELECT *  
FROM SALA  
ORDER BY id_sala;
```

PrintScreen cu rezultatul:

The screenshot shows a database management interface with the following details:

- Database Explorer:** Shows the schema structure:
  - Tables: SALA, ORAS, CLIENT.
  - Sequences: sequences (12).
- Console [localhost]:** Displays the SQL code used to create the SALA table and insert data into it. The table has columns: id\_sala (INT, primary key), id\_oras (INT), nume\_sala (VARCHAR(50)), adresa (VARCHAR(100)), and capacitate (INT). It also includes a FOREIGN KEY constraint linking id\_oras to ORAS(id\_oras). Data is inserted into SALA with values corresponding to the ORAS table entries.
- Playground:** Shows the query: `SELECT * FROM SALA ORDER BY id_sala;`

```

198 INSERT INTO SALA (id_oras, nume_sala, adresa, capacitate) VALUES (1, 'Fitness Arena', 'Strada Muncii 23', 120);
199 INSERT INTO SALA (id_oras, nume_sala, adresa, capacitate) VALUES (2, 'Body Shape', 'Bd. Eroilor 10', 100);
200 INSERT INTO SALA (id_oras, nume_sala, adresa, capacitate) VALUES (3, 'FitLife', 'Str. Libertatii 45', 90);
201 INSERT INTO SALA (id_oras, nume_sala, adresa, capacitate) VALUES (4, 'GymPro', 'Str. Independentei 15', 110);
202 INSERT INTO SALA (id_oras, nume_sala, adresa, capacitate) VALUES (5, 'ActiveZone', 'Bd. Tomis 200', 85);

SELECT *
FROM SALA
ORDER BY id_sala;
  
```

ID_SALA	ID_ORAS	NUME_SALA	ADRESA	CAPACITATE
1	1	Fitness Arena	Strada Muncii 23	120
2	2	Body Shape	Bd. Eroilor 10	100
3	3	FitLife	Str. Libertatii 45	90
4	4	GymPro	Str. Independentei 15	110
5	5	ActiveZone	Bd. Tomis 200	85

## 4. Tabela ECHIPAMENT

```

CREATE TABLE ECHIPAMENT(
    id_echipament INT DEFAULT ECHIPAMENT_SEQ.NEXTVAL PRIMARY KEY,
    id_sala INT,
    stare VARCHAR(30),
    denumire VARCHAR(50),
    FOREIGN KEY (id_sala) REFERENCES SALA(id_sala)
);
  
```

```

INSERT INTO ECHIPAMENT (id_sala, stare, denumire) VALUES (1, 'Buna', 'Banda alergare');
  
```

```

INSERT INTO ECHIPAMENT (id_sala, stare, denumire) VALUES (1, 'Avariat', 'Bicicleta stationara');
  
```

```

INSERT INTO ECHIPAMENT (id_sala, stare, denumire) VALUES (2, 'Buna', 'Aparat spate');
  
```

```

INSERT INTO ECHIPAMENT (id_sala, stare, denumire) VALUES (3, 'Buna', 'Bara haltere');
  
```

```
INSERT INTO ECHIPAMENT (id_sala, stare, denumire) VALUES (4, 'Buna', 'Banca  
abdomene');
```

```
INSERT INTO ECHIPAMENT (id_sala, stare, denumire) VALUES (5, 'Avariat', 'Gantere');
```

```
SELECT *  
FROM ECHIPAMENT  
ORDER BY id_echipament;
```

PrintScreen cu rezultatul:

The screenshot shows a database management interface with the following details:

- Database Explorer:** Shows the schema structure:
  - Tables: ECHIPAMENT, CLIENT, ORAS, SALA.
  - Sequences: 12.
  - Server Objects.
- Console:** Displays the SQL code used to create the table and insert data.

```
111 ORDER BY id_sala;  
112  
113  
114  
115  
116  
117 CREATE TABLE ECHIPAMENT(  
118     id_echipament INT DEFAULT ECHIPAMENT_SEQ.NEXTVAL PRIMARY KEY,  
119     id_sala INT,  
120     stare VARCHAR(30),  
121     denumire VARCHAR(50),  
122     FOREIGN KEY (id_sala) REFERENCES SALA(id_sala)  
123 );  
124  
125 INSERT INTO ECHIPAMENT (id_sala, stare, denumire) VALUES ( id_sala_1, stare 'Buna', denumire 'Banca alergare');  
126  
127 INSERT INTO ECHIPAMENT (id_sala, stare, denumire) VALUES ( id_sala_1, stare 'Avariat', denumire 'Biciclete stationara');  
128  
129 INSERT INTO ECHIPAMENT (id_sala, stare, denumire) VALUES ( id_sala_2, stare 'Buna', denumire 'Aparat spate');  
130  
131 INSERT INTO ECHIPAMENT (id_sala, stare, denumire) VALUES ( id_sala_3, stare 'Buna', denumire 'Bara haltere');  
132  
133 INSERT INTO ECHIPAMENT (id_sala, stare, denumire) VALUES ( id_sala_4, stare 'Buna', denumire 'Banca abdomene');  
134  
135 INSERT INTO ECHIPAMENT (id_sala, stare, denumire) VALUES ( id_sala_5, stare 'Avariat', denumire 'Gantere');  
136  
137 ✓  
138 SELECT *  
139 FROM ECHIPAMENT  
140 ORDER BY id_echipament;
```
- Status Bar:** Shows the time (13:16), file format (LF), encoding (UTF-8), and code style (4 spaces).

```

Database Explorer
@193.226.51.46 2 of 84
@localhost 2 of 33
FILIPDB
tables 4
CLIENT
ECHIPAMENT
ORAS
SALA
sequences 12
PUBLIC
Server Objects

126 INSERT INTO ECHIPAMENT (id_sala, stare, denumire) VALUES (1, 'Avariat', 'Bicicleta stationara');
127
128 INSERT INTO ECHIPAMENT (id_sala, stare, denumire) VALUES (2, 'Buna', 'Aparat spate');
129
130 INSERT INTO ECHIPAMENT (id_sala, stare, denumire) VALUES (3, 'Buna', 'Bara haltere');
131
132 INSERT INTO ECHIPAMENT (id_sala, stare, denumire) VALUES (4, 'Buna', 'Banca abdomene');
133
134 INSERT INTO ECHIPAMENT (id_sala, stare, denumire) VALUES (5, 'Avariat', 'Gantere');
135
136
137 SELECT *
138 FROM ECHIPAMENT
139 ORDER BY id_echipament;

Services
Database
@193.226.51.46
console 228
ID_ECHIPAMENT
ID_SALA
STARE
DENUMIRE
1 Avariat Banda alergare
2 Buna Bicicleta stationara
3 Buna Aparat spate
4 Buna Bara haltere
5 Buna Banca abdomene
6 Avariat Gantere
6 rows

Database Consoles > @localhost > console [@localhost]

```

## 5. Tabela CURS

```

CREATE TABLE CURS(
    id_curs INT DEFAULT CURS_SEQ.NEXTVAL PRIMARY KEY,
    id_sala INT,
    nivel_dificultate VARCHAR(20),
    denumire VARCHAR(50),
    FOREIGN KEY (id_sala) REFERENCES SALA(id_sala)
);

INSERT INTO CURS (id_sala, nivel_dificultate, denumire) VALUES (1, 'Incepator', 'Zumba');

INSERT INTO CURS (id_sala, nivel_dificultate, denumire) VALUES (2, 'Intermediar', 'Pilates');

INSERT INTO CURS (id_sala, nivel_dificultate, denumire) VALUES (3, 'Avansat', 'Pilates');

INSERT INTO CURS (id_sala, nivel_dificultate, denumire) VALUES (4, 'Avansat', 'Zumba');

```

```
INSERT INTO CURS (id_sala, nivel_dificultate, denumire) VALUES (5, 'Incepator', 'Functional Training');
```

```
SELECT *
FROM CURS
ORDER BY id_curs;
```

PrintScreen cu rezultatul:

The screenshot shows a database management interface with a sidebar for 'Database Explorer' and a main area for 'console [localhost]'. The 'Database Explorer' sidebar shows the database structure: FILIPDB > tables 5 > CURS. The main console area displays the following SQL code:

```
141 ORDER BY id_echipament;
142
143
144
145
146
147 CREATE TABLE CURS(
148     id_curs INT DEFAULT CURS_SEQ.NEXTVAL PRIMARY KEY,
149     id_sala INT,
150     nivel_dificultate VARCHAR(20),
151     denumire VARCHAR(50),
152     FOREIGN KEY (id_sala) REFERENCES SALA(id_sala)
153 );
154
155
156 INSERT INTO CURS (id_sala, nivel_dificultate, denumire) VALUES ( id_sala_1, nivel_dificultate 'Incepator', denumire 'Zumba');
157
158 INSERT INTO CURS (id_sala, nivel_dificultate, denumire) VALUES ( id_sala_2, nivel_dificultate 'Intermediate', denumire 'Pilates');
159
160 INSERT INTO CURS (id_sala, nivel_dificultate, denumire) VALUES ( id_sala_3, nivel_dificultate 'Avansat', denumire 'Pilates');
161
162 INSERT INTO CURS (id_sala, nivel_dificultate, denumire) VALUES ( id_sala_4, nivel_dificultate 'Avansat', denumire 'Zumba');
163
164 INSERT INTO CURS (id_sala, nivel_dificultate, denumire) VALUES ( id_sala_5, nivel_dificultate 'Incepator', denumire 'Functional Training');
165
166
167 ✓ SELECT *
168   FROM CURS
169   ORDER BY id_curs;
```

```

157 INSERT INTO CURS (id_sala, nivel_dificultate, denumire) VALUES (1, 'Intermediar', 'Pilates');
158 INSERT INTO CURS (id_sala, nivel_dificultate, denumire) VALUES (2, 'Avansat', 'Pilates');
159 INSERT INTO CURS (id_sala, nivel_dificultate, denumire) VALUES (3, 'Avansat', 'Zumba');
160 INSERT INTO CURS (id_sala, nivel_dificultate, denumire) VALUES (4, 'Avansat', 'Zumba');
161 INSERT INTO CURS (id_sala, nivel_dificultate, denumire) VALUES (5, 'Incepator', 'Functional Training');

167 SELECT *
168 FROM CURS
169 ORDER BY id_curs;
  
```

ID_CURS	ID_SALA	NIVEL_DIFICULTATE	DENUMIRE
1	1	Incepator	Zumba
2	2	Intermediar	Pilates
3	3	Avansat	Pilates
4	4	Avansat	Zumba
5	5	Incepator	Functional Training

## 6. Tabela ANTRENOR

```
CREATE TABLE ANTRENOR(
```

```

id_antrenor INT DEFAULT ANTRENOR_SEQ.NEXTVAL PRIMARY KEY,
nume VARCHAR(50),
email VARCHAR(50),
tip VARCHAR(20)
);
```

```
INSERT INTO ANTRENOR (nume, email, tip) VALUES ('Popescu Andrei',
'andrei.p@example.com', 'Curs');
```

```
INSERT INTO ANTRENOR (nume, email, tip) VALUES ('Ionescu Maria',
'maria.i@example.com', 'Personal');
```

```
INSERT INTO ANTRENOR (nume, email, tip) VALUES ('Vasilescu George',
'george.v@example.com', 'Curs');
```

```
INSERT INTO ANTRENOR (nume, email, tip) VALUES ('Stanescu Ioana',
'ioana.s@example.com', 'Personal');
```

```
INSERT INTO ANTRENOR (nume, email, tip) VALUES ('Dumitrescu Mihai',
```

```
'mihai.d@example.com', 'Curs');
```

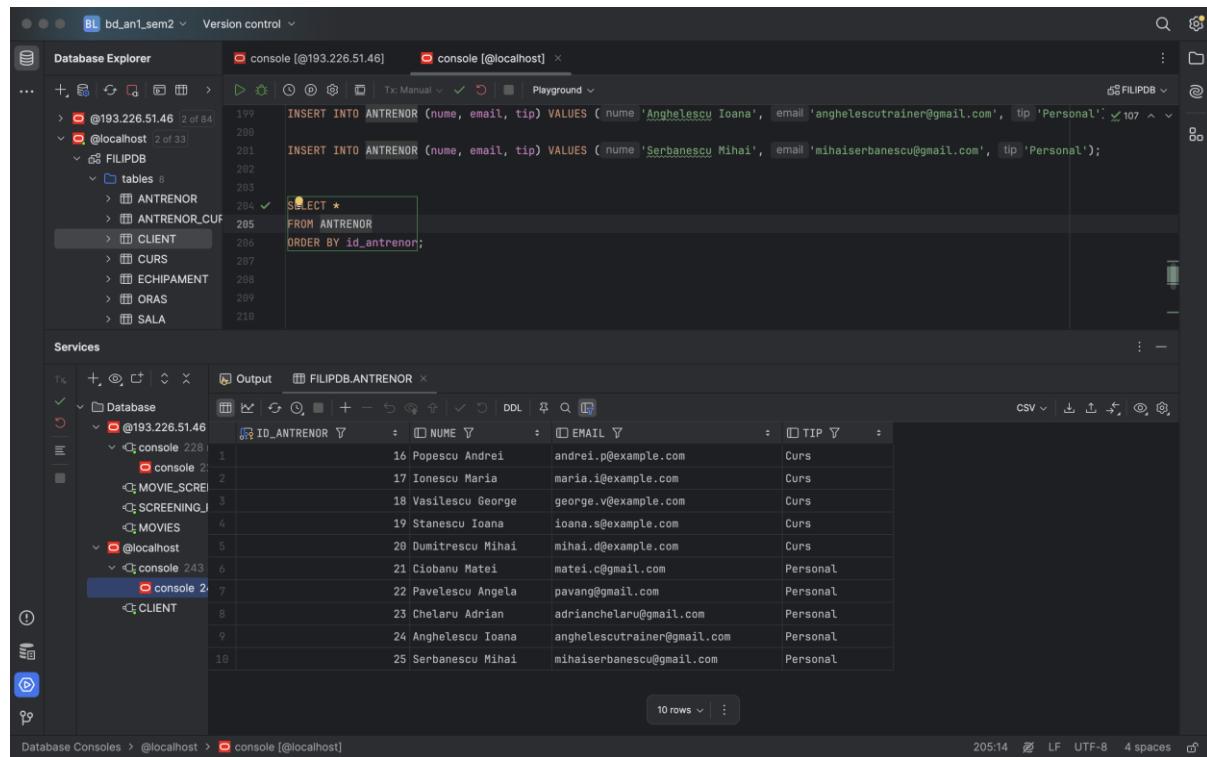
```
SELECT *
FROM ANTRENOR
ORDER BY id_antrenor;
```

PrintScreen cu rezultatul:

```
CREATE TABLE ANTRENOR(
    id_antrenor INT DEFAULT ANTRENOR_SEQ.NEXTVAL PRIMARY KEY,
    nume VARCHAR(50),
    email VARCHAR(50),
    tip VARCHAR(20)
);

INSERT INTO ANTRENOR (nume, email, tip) VALUES ('Popescu Andrei', 'andrei.p@example.com', 'Curs');
INSERT INTO ANTRENOR (nume, email, tip) VALUES ('Ionescu Maria', 'maria.i@example.com', 'Curs');
INSERT INTO ANTRENOR (nume, email, tip) VALUES ('Vasilescu George', 'george.v@example.com', 'Curs');
INSERT INTO ANTRENOR (nume, email, tip) VALUES ('Stanescu Ioana', 'ioana.s@example.com', 'Curs');
INSERT INTO ANTRENOR (nume, email, tip) VALUES ('Dumitrescu Mihai', 'mihai.d@example.com', 'Curs');
INSERT INTO ANTRENOR (nume, email, tip) VALUES ('Ciobanu Matei', 'matei.c@gmail.com', 'Personal');
INSERT INTO ANTRENOR (nume, email, tip) VALUES ('Pavelescu Angela', 'pavang@gmail.com', 'Personal');
INSERT INTO ANTRENOR (nume, email, tip) VALUES ('Chelaru Adrian', 'adrianchelaru@gmail.com', 'Personal');
INSERT INTO ANTRENOR (nume, email, tip) VALUES ('Anghelescu Ioana', 'anghelescutrainer@gmail.com', 'Personal');
INSERT INTO ANTRENOR (nume, email, tip) VALUES ('Serbanescu Mihai', 'mihaiserbanescu@gmail.com', 'Personal');

SELECT *
FROM ANTRENOR
ORDER BY id_antrenor;
```



## 7. Tabela TIP\_ABONAMENT

```
CREATE TABLE TIP_ABONAMENT(
    id_tip_abonament INT DEFAULT TIP_ABONAMENT_SEQ.NEXTVAL PRIMARY
KEY,
    pret INT,
    facilitati VARCHAR(100)
);
```

```
INSERT INTO TIP_ABONAMENT (pret, facilitati) VALUES (200, 'Acces general + Pilates');
```

```
INSERT INTO TIP_ABONAMENT (pret, facilitati) VALUES (250, 'Acces general + Zumba + Functional');
```

```
INSERT INTO TIP_ABONAMENT (pret, facilitati) VALUES (300, 'Acces general + Personal Trainer');
```

```
INSERT INTO TIP_ABONAMENT (pret, facilitati) VALUES (100, 'Acces limitat');
```

```
INSERT INTO TIP_ABONAMENT (pret, facilitati) VALUES (150, 'Acces limitat +
```

Pilates');

```
SELECT *
FROM TIP_ABONAMENT
ORDER BY id_tip_abonament;
```

PrintScreen cu rezultatul:

The screenshot shows a database management interface with the following details:

- Database Explorer:** Shows the schema structure of the database, including tables like ANTRENOR, CLIENT, CURS, EQUIPAMENT, ORAS, SALA, and TIP\_ABONAMENT.
- Console:** Displays the execution of SQL statements. The statements include creating a table TIP\_ABONAMENT with columns id\_tip\_abonament (primary key), pret (INT), and facilitati (VARCHAR(100)). Data is then inserted into this table with various values for pret and facilitati, including 'Acces general + Pilates', 'Acces general + Zumba + Functional', 'Acces general + Personal Trainer', 'Acces limitat', and 'Acces limitat + Pilates'.
- Result:** The final result of the SELECT query is shown, displaying the data from the TIP\_ABONAMENT table ordered by id\_tip\_abonament.

```

211 INSERT INTO TIP_ABONAMENT (pret, facilitati) VALUES ( pret 250, facilitati 'Acces general + Zumba + Functional');
212
213 INSERT INTO TIP_ABONAMENT (pret, facilitati) VALUES ( pret 300, facilitati 'Acces general + Personal Trainer');
214
215 INSERT INTO TIP_ABONAMENT (pret, facilitati) VALUES ( pret 100, facilitati 'Acces limitat');
216
217 INSERT INTO TIP_ABONAMENT (pret, facilitati) VALUES ( pret 150, facilitati 'Acces limitat + Pilates');
218
219 | SELECT *
220 | FROM TIP_ABONAMENT
221 | ORDER BY id_tip_abonament;
222
223

```

ID_TIP_ABONAMENT	PRET	FACILITATI
1	200	Acces general + Pilates
2	250	Acces general + Zumba + Functional
3	300	Acces general + Personal Trainer
4	100	Acces limitat
5	150	Acces limitat + Pilates

## 8. Tabela ANTRENOR\_CURS

```

CREATE TABLE ANTRENOR_CURS (
    id_antrenor INT PRIMARY KEY,
    ani_experienta INT,
    FOREIGN KEY (id_antrenor) REFERENCES ANTRENOR(id_antrenor)
);

INSERT INTO ANTRENOR_CURS (id_antrenor, ani_experienta) VALUES (16, 5);

INSERT INTO ANTRENOR_CURS (id_antrenor, ani_experienta) VALUES (17, 8);

INSERT INTO ANTRENOR_CURS (id_antrenor, ani_experienta) VALUES (18, 3);

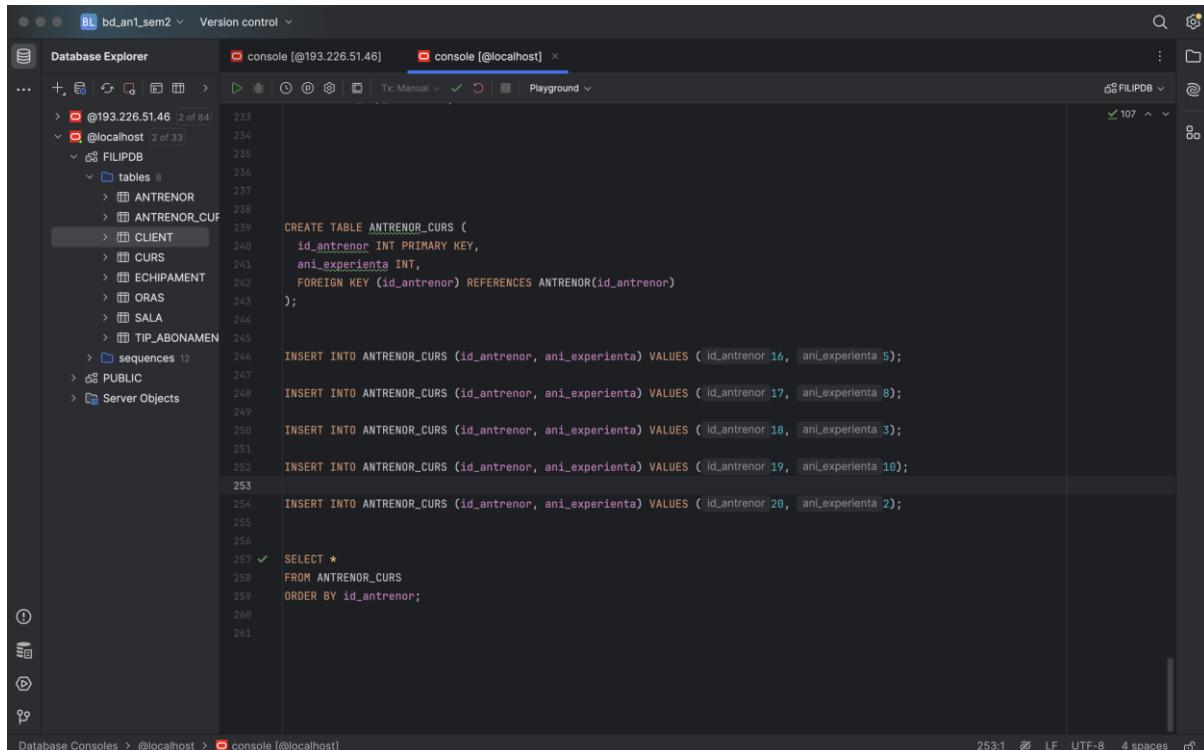
INSERT INTO ANTRENOR_CURS (id_antrenor, ani_experienta) VALUES (19, 10);

INSERT INTO ANTRENOR_CURS (id_antrenor, ani_experienta) VALUES (20, 2);

SELECT *
FROM ANTRENOR_CURS
ORDER BY id_antrenor;

```

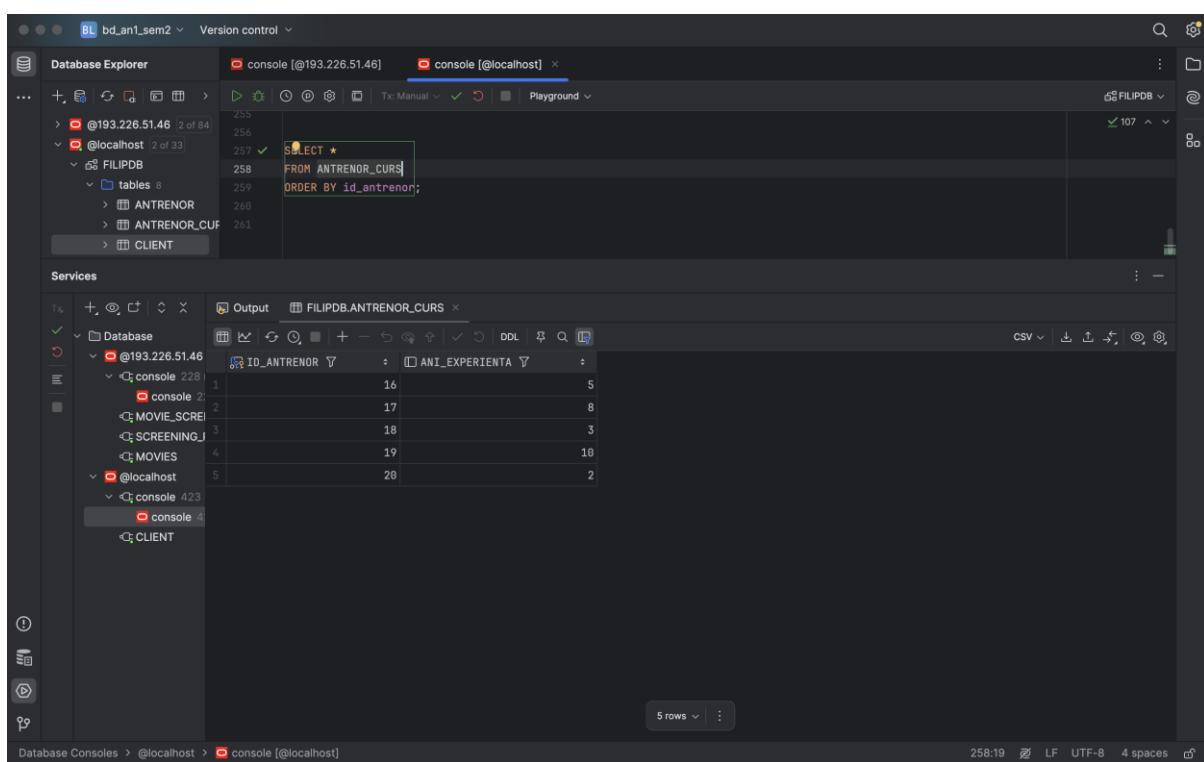
PrintScreen cu rezultatul:



```
CREATE TABLE ANTRENOR_CURS (
    id_antrenor INT PRIMARY KEY,
    ani_experienta INT,
    FOREIGN KEY (id_antrenor) REFERENCES ANTRENOR(id_antrenor)
);

INSERT INTO ANTRENOR_CURS (id_antrenor, ani_experienta) VALUES (16, 5);
INSERT INTO ANTRENOR_CURS (id_antrenor, ani_experienta) VALUES (17, 8);
INSERT INTO ANTRENOR_CURS (id_antrenor, ani_experienta) VALUES (18, 3);
INSERT INTO ANTRENOR_CURS (id_antrenor, ani_experienta) VALUES (19, 10);
INSERT INTO ANTRENOR_CURS (id_antrenor, ani_experienta) VALUES (20, 2);

SELECT *
FROM ANTRENOR_CURS
ORDER BY id_antrenor;
```



```
SELECT *
FROM ANTRENOR_CURS
ORDER BY id_antrenor;
```

ID ANTRENOR	ANI EXPERIENTA
16	5
17	8
18	3
19	10
20	2

## 9. Tabela ANTRENOR\_PERSONAL

```
CREATE TABLE ANTRENOR_PERSONAL (
    id_antrenor INT PRIMARY KEY,
    specializare_musculatura VARCHAR(50),
    FOREIGN KEY (id_antrenor) REFERENCES ANTRENOR(id_antrenor)
);
```

```
INSERT INTO ANTRENOR_PERSONAL (id_antrenor, specializare_musculatura)
VALUES (21, 'Upper-body');
```

```
INSERT INTO ANTRENOR_PERSONAL (id_antrenor, specializare_musculatura)
VALUES (22, 'Lower-body');
```

```
INSERT INTO ANTRENOR_PERSONAL (id_antrenor, specializare_musculatura)
VALUES (23, 'Lower-body');
```

```
INSERT INTO ANTRENOR_PERSONAL (id_antrenor, specializare_musculatura)
VALUES (24, 'Full-body');
```

```
INSERT INTO ANTRENOR_PERSONAL (id_antrenor, specializare_musculatura)
VALUES (25, 'Full-body');
```

```
SELECT *
FROM ANTRENOR_PERSONAL
ORDER BY id_antrenor;
```

PrintScreen cu rezultatul:

The screenshot shows the Database Explorer interface with the following details:

- Database Explorer:** Shows the structure of the FILIPDB database, including tables like ANTRENOR, ANTRENOR\_CUF, ANTRENOR\_PER, CLIENT, CURS, ECHIPAMENT, ORAS, SALA, TIP\_ABONAMENT, and sequences.
- Console [@localhost]:** Displays the creation of the ANTRENOR\_PERSONAL table and a subsequent SELECT query.
- Code:**

```
258 FROM ANTRENOR_CURS
259 ORDER BY id_antrenor;
260
261
262
263
264
265
266 CREATE TABLE ANTRENOR_PERSONAL (
267     id_antrenor INT PRIMARY KEY,
268     specializare_musculatura VARCHAR(50),
269     FOREIGN KEY (id_antrenor) REFERENCES ANTRENOR(id_antrenor)
270 );
271
272
273 INSERT INTO ANTRENOR_PERSONAL (id_antrenor, specializare_musculatura) VALUES (id_antrenor 21, specializare_musculatura 'Upper-body');
274
275 INSERT INTO ANTRENOR_PERSONAL (id_antrenor, specializare_musculatura) VALUES (id_antrenor 22, specializare_musculatura 'Lower-body');
276
277 INSERT INTO ANTRENOR_PERSONAL (id_antrenor, specializare_musculatura) VALUES (id_antrenor 23, specializare_musculatura 'Lower-body');
278
279 INSERT INTO ANTRENOR_PERSONAL (id_antrenor, specializare_musculatura) VALUES (id_antrenor 24, specializare_musculatura 'Full-body');
280
281 INSERT INTO ANTRENOR_PERSONAL (id_antrenor, specializare_musculatura) VALUES (id_antrenor 25, specializare_musculatura 'Full-body');
282
283
284 ✓ SELECT *
285 FROM ANTRENOR_PERSONAL
286 ORDER BY id_antrenor;
```
- Bottom Status:** Shows the time as 285:23, file format as LF, encoding as UTF-8, and a 4 spaces indentation setting.

The screenshot shows the Database Explorer interface with the following details:

- Database Explorer:** Shows the structure of the FILIPDB database, including tables like ANTRENOR, ANTRENOR\_CUF, and ANTRENOR\_DCD.
- Services:** Shows the execution of a SELECT query on the ANTRENOR\_PERSONAL table.
- Output Tab:** Displays the results of the query, showing five rows of data:

ID_ANTRENOR	SPECIALIZARE_MUSCULATURA
21	Upper-body
22	Lower-body
23	Lower-body
24	Full-body
25	Full-body

- Bottom Status:** Shows the time as 284:9, file format as LF, encoding as UTF-8, and a 4 spaces indentation setting.

## 10. Tabela PROGRAMARE\_CURS

```
CREATE TABLE PROGRAMARE_CURS (
    id_programare INT PRIMARY KEY,
    id_antrenor INT,
```

```
id_curs INT,  
data DATE,  
ora_programare VARCHAR2(5),  
FOREIGN KEY (id_antrenor) REFERENCES ANTRENOR(id_antrenor),  
FOREIGN KEY (id_curs) REFERENCES CURS(id_curs)  
);  
  
INSERT INTO PROGRAMARE_CURS (id_programare, id_antrenor, id_curs, data,  
ora_programare)  
VALUES (1, 16, 1, DATE '2025-06-01', '10:00');  
  
INSERT INTO PROGRAMARE_CURS (id_programare, id_antrenor, id_curs, data,  
ora_programare)  
VALUES (2, 16, 2, DATE '2025-06-02', '12:00');  
  
INSERT INTO PROGRAMARE_CURS (id_programare, id_antrenor, id_curs, data,  
ora_programare)  
VALUES (3, 17, 3, DATE '2025-06-03', '14:00');  
  
INSERT INTO PROGRAMARE_CURS (id_programare, id_antrenor, id_curs, data,  
ora_programare)  
VALUES (4, 17, 4, DATE '2025-06-04', '16:00');  
  
INSERT INTO PROGRAMARE_CURS (id_programare, id_antrenor, id_curs, data,  
ora_programare)  
VALUES (5, 18, 5, DATE '2025-06-05', '18:00');  
  
INSERT INTO PROGRAMARE_CURS (id_programare, id_antrenor, id_curs, data,  
ora_programare)  
VALUES (6, 18, 1, DATE '2025-06-06', '10:00');  
  
INSERT INTO PROGRAMARE_CURS (id_programare, id_antrenor, id_curs, data,  
ora_programare)  
VALUES (7, 19, 2, DATE '2025-06-07', '12:00');  
  
INSERT INTO PROGRAMARE_CURS (id_programare, id_antrenor, id_curs, data,  
ora_programare)  
VALUES (8, 19, 3, DATE '2025-06-08', '14:00');  
  
INSERT INTO PROGRAMARE_CURS (id_programare, id_antrenor, id_curs, data,  
ora_programare)  
VALUES (9, 20, 4, DATE '2025-06-09', '16:00');
```

```
INSERT INTO PROGRAMARE_CURS (id_programare, id_antrenor, id_curs, data,  
ora_programare)  
VALUES (10, 20, 5, DATE '2025-06-10', '18:00');
```

```
SELECT *  
FROM PROGRAMARE_CURS  
ORDER BY id_programare;
```

PrintScreen cu rezultatul:

The screenshot shows a database management interface with the following details:

- Database Explorer:** Shows the schema structure of the database, including tables like ANTRENOR, CURS, and PROGRAMARE\_CURS.
- Console:** Displays the SQL code being run:

```
CREATE TABLE PROGRAMARE_CURS (  
    id_programare INT PRIMARY KEY,  
    id_antrenor INT,  
    id_curs INT,  
    data DATE,  
    ora_programare VARCHAR2(5),  
    FOREIGN KEY (id_antrenor) REFERENCES ANTRENOR(id_antrenor),  
    FOREIGN KEY (id_curs) REFERENCES CURS(id_curs)  
);  
  
INSERT INTO PROGRAMARE_CURS (id_programare, id_antrenor, id_curs, data, ora_programare)  
VALUES (1, 16, 1, DATE '2025-06-01', '10:00');  
  
INSERT INTO PROGRAMARE_CURS (id_programare, id_antrenor, id_curs, data, ora_programare)  
VALUES (2, 16, 2, DATE '2025-06-02', '12:00');  
  
INSERT INTO PROGRAMARE_CURS (id_programare, id_antrenor, id_curs, data, ora_programare)  
VALUES (3, 17, 3, DATE '2025-06-03', '14:00');  
  
INSERT INTO PROGRAMARE_CURS (id_programare, id_antrenor, id_curs, data, ora_programare)  
VALUES (4, 17, 4, DATE '2025-06-04', '16:00');  
  
INSERT INTO PROGRAMARE_CURS (id_programare, id_antrenor, id_curs, data, ora_programare)  
VALUES (5, 18, 5, DATE '2025-06-05', '18:00');  
  
INSERT INTO PROGRAMARE_CURS (id_programare, id_antrenor, id_curs, data, ora_programare)  
VALUES (6, 18, 1, DATE '2025-06-06', '10:00');  
  
INSERT INTO PROGRAMARE_CURS (id_programare, id_antrenor, id_curs, data, ora_programare)  
VALUES (7, 19, 2, DATE '2025-06-07', '12:00');  
  
INSERT INTO PROGRAMARE_CURS (id_programare, id_antrenor, id_curs, data, ora_programare)  
VALUES (8, 19, 3, DATE '2025-06-08', '14:00');
```
- Bottom Status Bar:** Shows the current file size as 329.1, encoding as LF, and character set as UTF-8.

The screenshot shows the DataGrip IDE interface. The top navigation bar includes tabs for 'Database Explorer' and 'Services'. The 'Database Explorer' pane on the left lists database connections and tables under the 'FILIPDB' schema. The 'Services' pane on the right displays the results of a query against the 'PROGRAMARE\_CURS' table.

**Database Explorer**

- Connections: bd\_an1\_sem2, console [@193.226.51.46], console [@localhost]
- Tables under FILIPDB:
  - ANTRONOR (selected)
  - ANTRONOR.CUF
  - ANTRONOR.PER
  - CLIENT
  - CURS
  - ECHIPAMENT
  - OCAC

**Services**

Output: FILIPDB.PROGRAMARE\_CURS

ID_PROGRAMARE	ID_ANTRENOR	ID_CURS	DATA	ORA_PROGRAMARE
1	1	16	2025-06-01	10:00
2	2	16	2025-06-02	12:00
3	3	17	2025-06-03	14:00
4	4	17	2025-06-04	16:00
5	5	18	2025-06-05	18:00
6	6	18	2025-06-06	10:00
7	7	19	2025-06-07	12:00
8	8	19	2025-06-08	14:00
9	9	20	2025-06-09	16:00
10	10	20	2025-06-10	18:00

## 11. Tabela COMPONENTA CLASA

```
CREATE TABLE COMPONENTA_CLASA(
    id_componenta INT DEFAULT COMPONENTA_SEQ.NEXTVAL PRIMARY KEY,
    id_client INT,
    id_programare INT,
    prezenta_efectiva NUMBER(1),
    data_confirmare DATE,
    FOREIGN KEY (id_client) REFERENCES CLIENT(id_client),
    FOREIGN KEY (id_programare) REFERENCES PROGRAMARE_CURS(id_programare)
);

INSERT INTO COMPONENTA_CLASA (id_client, id_programare, prezenta_efectiva,
data_confirmare)
VALUES (1, 2, '1', TO_DATE('2025-06-01', 'YYYY-MM-DD'));

INSERT INTO COMPONENTA_CLASA (id_client, id_programare, prezenta_efectiva,
data_confirmare)
VALUES (2, 3, '0', TO_DATE('2025-06-01', 'YYYY-MM-DD'));

INSERT INTO COMPONENTA_CLASA (id_client, id_programare, prezenta_efectiva,
data_confirmare)
VALUES (3, 4, '1', TO_DATE('2025-06-03', 'YYYY-MM-DD'));

INSERT INTO COMPONENTA_CLASA (id_client, id_programare, prezenta_efectiva,
data_confirmare)
VALUES (4, 1, '1', TO_DATE('2025-06-01', 'YYYY-MM-DD'));

INSERT INTO COMPONENTA_CLASA (id_client, id_programare, prezenta_efectiva,
data_confirmare)
VALUES (5, 5, '0', TO_DATE('2025-06-03', 'YYYY-MM-DD'));

INSERT INTO COMPONENTA_CLASA (id_client, id_programare, prezenta_efectiva,
data_confirmare)
VALUES (1, 6, '1', TO_DATE('2025-06-01', 'YYYY-MM-DD'));

INSERT INTO COMPONENTA_CLASA (id_client, id_programare, prezenta_efectiva,
data_confirmare)
VALUES (2, 7, '0', TO_DATE('2025-06-03', 'YYYY-MM-DD'));

INSERT INTO COMPONENTA_CLASA (id_client, id_programare, prezenta_efectiva,
data_confirmare)
```

```
VALUES (3, 8, '1', TO_DATE('2025-06-05', 'YYYY-MM-DD'));

INSERT INTO COMPONENTA_CLASA (id_client, id_programare, prezenta_efectiva,
data_confirmare)
VALUES (4, 9, '1', TO_DATE('2025-06-06', 'YYYY-MM-DD'));

INSERT INTO COMPONENTA_CLASA (id_client, id_programare, prezenta_efectiva,
data_confirmare)
VALUES (5, 10, '0', TO_DATE('2025-06-08', 'YYYY-MM-DD'));

SELECT *
FROM COMPONENTA_CLASA
ORDER BY id_componenta;
```

PrintScreen cu rezultatul:

The screenshot shows the Oracle Database SQL Developer application. In the Database Explorer on the left, the schema 'FILIPDB' is selected, showing tables like ANTRENOR, ANTRENOR\_CUF, ANTRENOR\_PER, CLIENT, COMPONENTA\_CLASA, CURS, ECHIPAMENT, ORAS, PROGRAMARE\_C, SALA, TIP\_ABONAMENT, and sequences. The 'COMPONENTA\_CLASA' table is currently selected. The main pane displays the following SQL code:

```
CREATE TABLE COMPONENTA_CLASA(
    id_componenta INT DEFAULT COMPONENTA_SEQ.NEXTVAL PRIMARY KEY,
    id_client INT,
    id_programare INT,
    prezenta_efectiva NUMBER(1),
    data_confirmare DATE,
    FOREIGN KEY (id_client) REFERENCES CLIENT(id_client),
    FOREIGN KEY (id_programare) REFERENCES PROGRAMARE_C(id_programare)
);

INSERT INTO COMPONENTA_CLASA (id_client, id_programare, prezenta_efectiva, data_confirmare)
VALUES (1, 1, '1', TO_DATE('2025-06-01', 'YYYY-MM-DD'));

INSERT INTO COMPONENTA_CLASA (id_client, id_programare, prezenta_efectiva, data_confirmare)
VALUES (2, 2, '1', TO_DATE('2025-06-01', 'YYYY-MM-DD'));

INSERT INTO COMPONENTA_CLASA (id_client, id_programare, prezenta_efectiva, data_confirmare)
VALUES (3, 3, '1', TO_DATE('2025-06-03', 'YYYY-MM-DD'));

INSERT INTO COMPONENTA_CLASA (id_client, id_programare, prezenta_efectiva, data_confirmare)
VALUES (4, 4, '1', TO_DATE('2025-06-01', 'YYYY-MM-DD'));

INSERT INTO COMPONENTA_CLASA (id_client, id_programare, prezenta_efectiva, data_confirmare)
VALUES (5, 5, '0', TO_DATE('2025-06-03', 'YYYY-MM-DD'));

INSERT INTO COMPONENTA_CLASA (id_client, id_programare, prezenta_efectiva, data_confirmare)
VALUES (1, 6, '1', TO_DATE('2025-06-01', 'YYYY-MM-DD'));

INSERT INTO COMPONENTA_CLASA (id_client, id_programare, prezenta_efectiva, data_confirmare)
VALUES (2, 7, '0', TO_DATE('2025-06-03', 'YYYY-MM-DD'));

INSERT INTO COMPONENTA_CLASA (id_client, id_programare, prezenta_efectiva, data_confirmare)
```

The screenshot shows the DataGrip IDE interface with the following details:

- Database Explorer** pane:
  - Connected to `console [@193.226.51.46]`
  - Current query step: 2 of 84
  - Query results:
    - Step 385: `SELECT * FROM COMPONENTA_CLASA`
    - Step 386: `ORDER BY id_componenta;`
- Services** pane:
  - Connected to `@193.226.51.46`
  - Current database: `FILIPDB`
  - Tables:
    - `ANTRENOR`
    - `ANTRENOR_CUF`
    - `ANTRENOR_DOD`
- Output** pane:
  - Table: `FILIPDB.COMPONENTA_CLASA`
  - Columns: `ID_COMPONENTA`, `ID_CLIENT`, `ID_PROGRAMARE`, `PREZENTA_EFECTIVA`, `DATA_CONFIRMARE`
  - Data:

	ID_COMPONENTA	ID_CLIENT	ID_PROGRAMARE	PREZENTA_EFECTIVA	DATA_CONFIRMARE
1	1	1	2	1	2025-06-01
2	2	2	3	0	2025-06-01
3	3	3	4	1	2025-06-03
4	4	4	1	1	2025-06-01
5	5	5	5	0	2025-06-03
6	6	1	6	1	2025-06-01
7	7	2	7	0	2025-06-03
8	8	3	8	1	2025-06-05
9	9	4	9	1	2025-06-06
10	10	5	10	0	2025-06-08

## 12. Tabela ISTORIC

```
CREATE TABLE ISTORIC(
    id_istoric INT DEFAULT ISTORIC_SEQ.NEXTVAL PRIMARY KEY,
    id_client INT,
    id_tip_abonament INT,
    id_sala INT,
    data_inceput DATE,
    data_expirare DATE,
    FOREIGN KEY (id_client) REFERENCES CLIENT(id_client),
    FOREIGN KEY (id_tip_abonament) REFERENCES
TIP_ABONAMENT(id_tip_abonament),
    FOREIGN KEY (id_sala) REFERENCES SALA(id_sala)
);
```

```
INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES (1, 1, 1, TO_DATE('2025-06-01', 'YYYY-MM-DD'), TO_DATE('2025-08-01',
'YYYY-MM-DD'));
```

```
INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES (2, 1, 2, TO_DATE('2025-06-01', 'YYYY-MM-DD'), TO_DATE('2025-08-01',
'YYYY-MM-DD'));
```

```
INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES (3, 2, 3, TO_DATE('2025-06-01', 'YYYY-MM-DD'), TO_DATE('2025-08-01',
'YYYY-MM-DD'));
```

```
INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES (4, 2, 4, TO_DATE('2025-06-01', 'YYYY-MM-DD'), TO_DATE('2025-08-01',
'YYYY-MM-DD'));
```

```
INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES (5, 3, 5, TO_DATE('2025-06-01', 'YYYY-MM-DD'), TO_DATE('2025-09-01',
'YYYY-MM-DD'));
```

```
INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES (6, 3, 1, TO_DATE('2025-06-02', 'YYYY-MM-DD'), TO_DATE('2025-09-02',
'YYYY-MM-DD'));
```

```
INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES (7, 4, 2, TO_DATE('2025-06-02', 'YYYY-MM-DD'), TO_DATE('2025-07-02',
'YYYY-MM-DD'));
```

```
INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
```

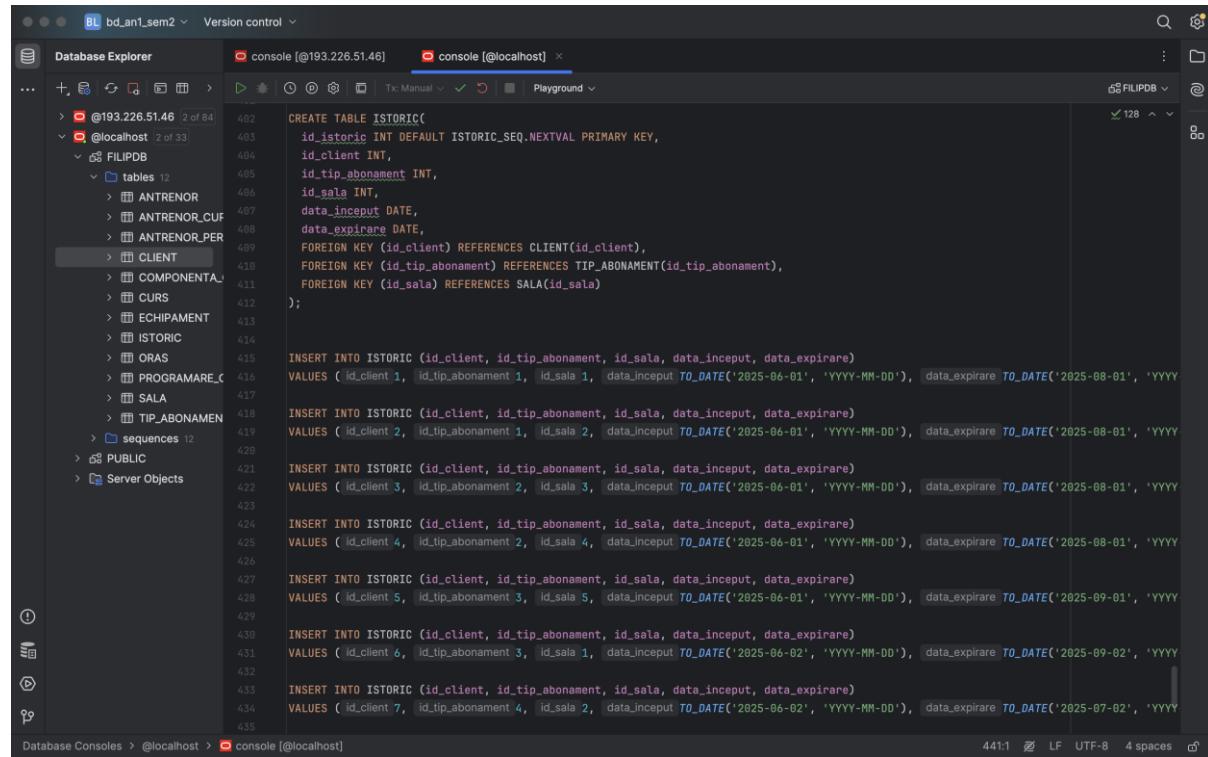
```
VALUES (8, 4, 3, TO_DATE('2025-06-02', 'YYYY-MM-DD'), TO_DATE('2025-07-02', 'YYYY-MM-DD'));
```

```
INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES (9, 5, 4, TO_DATE('2025-06-02', 'YYYY-MM-DD'), TO_DATE('2025-07-02', 'YYYY-MM-DD'));
```

```
INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES (10, 5, 5, TO_DATE('2025-06-02', 'YYYY-MM-DD'), TO_DATE('2025-07-02', 'YYYY-MM-DD'));
```

```
SELECT *
FROM ISTORIC
ORDER BY id_istoric;
```

PrintScreen cu rezultatul:



The screenshot shows the MySQL Workbench interface with the following details:

- Database Explorer:** Shows the connection to `@193.226.51.46` and the schema `FILIPDB`.
- Console:** Displays the SQL code used to create the `ISTORIC` table and insert test data.
- Code:**

```
CREATE TABLE ISTORIC(
    id_istoric INT DEFAULT ISTORIC_SEQ.NEXTVAL PRIMARY KEY,
    id_client INT,
    id_tip_abonament INT,
    id_sala INT,
    data_inceput DATE,
    data_expirare DATE,
    FOREIGN KEY (id_client) REFERENCES CLIENT(id_client),
    FOREIGN KEY (id_tip_abonament) REFERENCES TIP_ABONAMENT(id_tip_abonament),
    FOREIGN KEY (id_sala) REFERENCES SALA(id_sala)
);

INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES (1, 1, 1, TO_DATE('2025-06-01', 'YYYY-MM-DD'), TO_DATE('2025-08-01', 'YYYY-MM-DD'));

INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES (2, 1, 2, TO_DATE('2025-06-01', 'YYYY-MM-DD'), TO_DATE('2025-08-01', 'YYYY-MM-DD'));

INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES (3, 2, 3, TO_DATE('2025-06-01', 'YYYY-MM-DD'), TO_DATE('2025-08-01', 'YYYY-MM-DD'));

INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES (4, 2, 4, TO_DATE('2025-06-01', 'YYYY-MM-DD'), TO_DATE('2025-08-01', 'YYYY-MM-DD'));

INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES (5, 3, 5, TO_DATE('2025-06-01', 'YYYY-MM-DD'), TO_DATE('2025-09-01', 'YYYY-MM-DD'));

INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES (6, 3, 1, TO_DATE('2025-06-02', 'YYYY-MM-DD'), TO_DATE('2025-09-02', 'YYYY-MM-DD'));

INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES (7, 4, 2, TO_DATE('2025-06-02', 'YYYY-MM-DD'), TO_DATE('2025-07-02', 'YYYY-MM-DD'));
```

The screenshot shows the Database Explorer in Visual Studio Code. A connection to `localhost` is selected. The script pane contains the following SQL code:

```
INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES ( id_client_3, id_tip_abonament_2, id_sala_3, data_inceput TO_DATE('2025-06-01', 'YYYY-MM-DD'), data_expirare TO_DATE('2025-08-01', 'YYYY-MM-DD') )

INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES ( id_client_4, id_tip_abonament_2, id_sala_4, data_inceput TO_DATE('2025-06-01', 'YYYY-MM-DD'), data_expirare TO_DATE('2025-08-01', 'YYYY-MM-DD') )

INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES ( id_client_5, id_tip_abonament_3, id_sala_5, data_inceput TO_DATE('2025-06-01', 'YYYY-MM-DD'), data_expirare TO_DATE('2025-09-01', 'YYYY-MM-DD') )

INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES ( id_client_6, id_tip_abonament_3, id_sala_1, data_inceput TO_DATE('2025-06-02', 'YYYY-MM-DD'), data_expirare TO_DATE('2025-09-02', 'YYYY-MM-DD') )

INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES ( id_client_7, id_tip_abonament_4, id_sala_2, data_inceput TO_DATE('2025-06-02', 'YYYY-MM-DD'), data_expirare TO_DATE('2025-07-02', 'YYYY-MM-DD') )

INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES ( id_client_8, id_tip_abonament_4, id_sala_3, data_inceput TO_DATE('2025-06-02', 'YYYY-MM-DD'), data_expirare TO_DATE('2025-07-02', 'YYYY-MM-DD') )

INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES ( id_client_9, id_tip_abonament_5, id_sala_4, data_inceput TO_DATE('2025-06-02', 'YYYY-MM-DD'), data_expirare TO_DATE('2025-07-02', 'YYYY-MM-DD') )

INSERT INTO ISTORIC (id_client, id_tip_abonament, id_sala, data_inceput, data_expirare)
VALUES ( id_client_10, id_tip_abonament_5, id_sala_5, data_inceput TO_DATE('2025-06-02', 'YYYY-MM-DD'), data_expirare TO_DATE('2025-07-02', 'YYYY-MM-DD') )

SELECT *
FROM ISTORIC
ORDER BY id_istoric;
```

The screenshot shows the Database Explorer and Output tabs in VS Code. The Database Explorer tab displays a tree view of databases, tables, and columns. The Output tab shows the results of an SQL query executed against the FILIPDB database.

**Database Explorer**

- @193.226.51.46 (2 of 84)
- @localhost (2 of 33)
  - FILIPDB
    - tables (14)
      - ANTRENOR
      - ANTRENOR\_CUF
      - ANTRENOR\_PER
      - CLIENT
      - COMPONENTA\_
      - CURS
      - ECHIPAMENT

**Output**

SELECT \*  
FROM ISTORIC  
ORDER BY id\_istoric;

ID_ISTORIC	ID_CLIENT	ID_TIP_ABONAMENT	ID_SALA	DATA_INCEPUT	DATA_EXPIRARE
1	31	1	1	1 2025-06-01	2025-08-01
2	32	2	1	2 2025-06-01	2025-08-01
3	33	3	2	3 2025-06-01	2025-08-01
4	34	4	2	4 2025-06-01	2025-08-01
5	35	5	3	5 2025-06-01	2025-09-01
6	36	6	3	1 2025-06-02	2025-09-02
7	37	7	4	2 2025-06-02	2025-07-02
8	38	8	4	3 2025-06-02	2025-07-02
9	39	9	5	4 2025-06-02	2025-07-02
10	40	10	5	5 2025-06-02	2025-07-02

### 13. Tabela FEEDBACK

```
CREATE TABLE FEEDBACK(
    id_feedback INT DEFAULT FEEDBACK_SEQ.NEXTVAL PRIMARY KEY,
    id_antrenor INT,
    id_client INT,
    data DATE,
    mesaj VARCHAR(100),
    FOREIGN KEY (id_antrenor) REFERENCES ANTRENOR(id_antrenor),
    FOREIGN KEY (id_client) REFERENCES CLIENT(id_client)
);

INSERT INTO FEEDBACK (id_antrenor, id_client, data, mesaj)
VALUES (16, 1, TO_DATE('2025-06-21', 'YYYY-MM-DD'), 'Foarte curat!');

INSERT INTO FEEDBACK (id_antrenor, id_client, data, mesaj)
VALUES (16, 2, TO_DATE('2025-06-22', 'YYYY-MM-DD'), 'Excelent!');

INSERT INTO FEEDBACK (id_antrenor, id_client, data, mesaj)
VALUES (19, 3, TO_DATE('2025-07-03', 'YYYY-MM-DD'), 'A fost greu, dar util!');

INSERT INTO FEEDBACK (id_antrenor, id_client, data, mesaj)
VALUES (17, 4, TO_DATE('2025-06-04', 'YYYY-MM-DD'), 'Foarte implicat!');

INSERT INTO FEEDBACK (id_antrenor, id_client, data, mesaj)
VALUES (24, 5, TO_DATE('2025-06-15', 'YYYY-MM-DD'), 'Super experienta!');

INSERT INTO FEEDBACK (id_antrenor, id_client, data, mesaj)
VALUES (24, 2, TO_DATE('2025-07-06', 'YYYY-MM-DD'), 'Nota 10!');

INSERT INTO FEEDBACK (id_antrenor, id_client, data, mesaj)
VALUES (20, 3, TO_DATE('2025-07-17', 'YYYY-MM-DD'), 'M-a motivat mult!');

INSERT INTO FEEDBACK (id_antrenor, id_client, data, mesaj)
VALUES (23, 1, TO_DATE('2025-06-28', 'YYYY-MM-DD'), 'Recomand!');

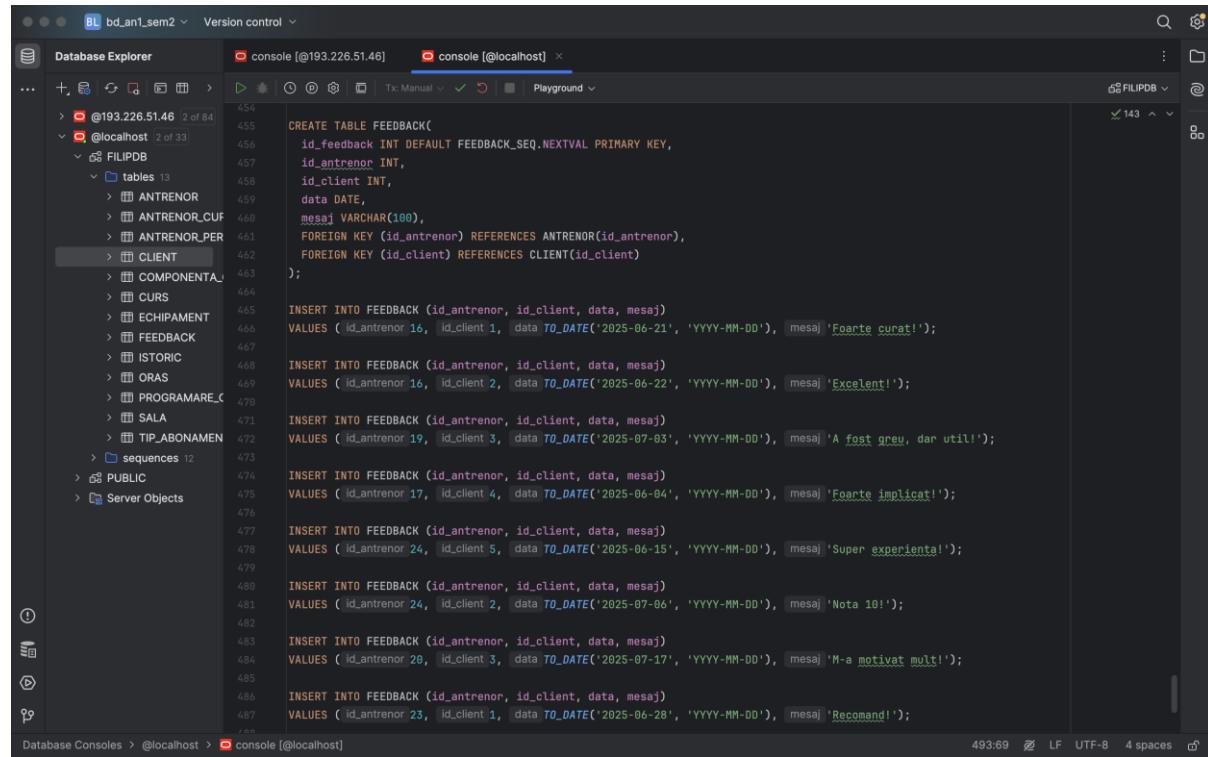
INSERT INTO FEEDBACK (id_antrenor, id_client, data, mesaj)
VALUES (18, 7, TO_DATE('2025-06-19', 'YYYY-MM-DD'), 'Super curs!');

INSERT INTO FEEDBACK (id_antrenor, id_client, data, mesaj)
VALUES (25, 8, TO_DATE('2025-07-10', 'YYYY-MM-DD'), 'Foarte clar!');

SELECT *
```

```
FROM FEEDBACK
ORDER BY id_feedback;
```

PrintScreen cu rezultatul:



```

CREATE TABLE FEEDBACK(
    id_feedback INT DEFAULT FEEDBACK_SEQ.NEXTVAL PRIMARY KEY,
    id_antrenor INT,
    id_client INT,
    data DATE,
    mesaj VARCHAR(100),
    FOREIGN KEY (id_antrenor) REFERENCES ANTRENOR(id_antrenor),
    FOREIGN KEY (id_client) REFERENCES CLIENT(id_client)
);

INSERT INTO FEEDBACK (id_antrenor, id_client, data, mesaj)
VALUES (16, 1, data TO_DATE('2025-06-21', 'YYYY-MM-DD'), 'Foarte curat!');

INSERT INTO FEEDBACK (id_antrenor, id_client, data, mesaj)
VALUES (16, 2, data TO_DATE('2025-06-22', 'YYYY-MM-DD'), 'Excellent!');

INSERT INTO FEEDBACK (id_antrenor, id_client, data, mesaj)
VALUES (19, 3, data TO_DATE('2025-07-03', 'YYYY-MM-DD'), 'A fost greu, dar util!');

INSERT INTO FEEDBACK (id_antrenor, id_client, data, mesaj)
VALUES (17, 4, data TO_DATE('2025-06-04', 'YYYY-MM-DD'), 'Foarte implicat!');

INSERT INTO FEEDBACK (id_antrenor, id_client, data, mesaj)
VALUES (24, 5, data TO_DATE('2025-06-15', 'YYYY-MM-DD'), 'Super experienta!');

INSERT INTO FEEDBACK (id_antrenor, id_client, data, mesaj)
VALUES (24, 2, data TO_DATE('2025-07-06', 'YYYY-MM-DD'), 'Nota 10!');

INSERT INTO FEEDBACK (id_antrenor, id_client, data, mesaj)
VALUES (20, 3, data TO_DATE('2025-07-17', 'YYYY-MM-DD'), 'M-a motivat mult!');

INSERT INTO FEEDBACK (id_antrenor, id_client, data, mesaj)
VALUES (23, 1, data TO_DATE('2025-06-28', 'YYYY-MM-DD'), 'Recomand!');
```

The screenshot shows a developer environment with multiple tabs open. The Database Explorer tab is active, showing a tree view of database objects. The SQL console tab is also active, displaying a series of INSERT statements and a SELECT query.

```
Database Explorer
+ @193.226.51.46 2 of 84
  + @localhost 2 of 33
    + FILIPDB
      + tables 13
        + ANTRENOR
        + ANTRENOR_CUF
        + ANTRENOR_PER
        + CLIENT
        + COMPONENTA_
        + CURS
        + ECHIPAMENT
        + FEEDBACK
        + ISTORIC
        + ORAS
        + PROGRAMARE_C
        + SALA
        + TIP_ABONAMENT
      + sequences 12
    + PUBLIC
  + Server Objects

console [@193.226.51.46]
  + console [@localhost] (selected)
    + Ti: Manual ✓
    + Playground

... 143

@193.226.51.46 478
@localhost 2 of 84 479
FILIPDB 480
tables 13 481
ANTRENOR 482
ANTRENOR_CUF 483
ANTRENOR_PER 484
CLIENT 485
COMPONENTA_ 486
CURS 487
ECHIPAMENT 488
FEEDBACK 489
ISTORIC 490
ORAS 491
PROGRAMARE_C 492
SALA 493
TIP_ABONAMENT 494
sequences 12 495
PUBLIC 496
Server Objects 497

SELECT *
FROM FEEDBACK
ORDER BY id_feedback;
```

The screenshot shows a code editor interface with two main panes: Database Explorer and Services.

**Database Explorer:**

- Connected to `@193.226.51.46` (localhost).
- Selected database: `FILIPDB`.
- Tables listed under `FILIPDB`: `ANTRENOR`, `ANTRENOR.CUF`, `ANTRENOR.PER`, `CLIENT`, `COMPONENTA_`, `CURS`, `ECHIPAMENT`.
- Execution history:
  - Line 492: `INSERT INTO FEEDBACK (id_antrenor, id_client, data, mesaj)`
  - Line 493: `VALUES ( id_antrenor 25 , id_client 8 , data TO_DATE('2025-07-10', 'YYYY-MM-DD') , mesaj 'Foarte clar!')`
  - Line 496: `SELECT * FROM FEEDBACK ORDER BY id_feedback;`
- Total rows: 143.

**Services:**

- Connected to `@193.226.51.46` (localhost).
- Selected schema: `FILIPDB.FEEDBACK`.
- Data grid showing feedback records:

ID_FEEDBACK	ID_ANTRENOR	ID_CLIENT	DATA	MESAJ
1	16	1	2025-06-21	Foarte curat!
2	16	2	2025-06-22	Excellent!
3	19	3	2025-07-03	A fost greu, dar util!
4	17	4	2025-06-04	Foarte implicat!
5	24	5	2025-06-15	Super experienta!
6	24	2	2025-07-06	Nota 10!
7	28	3	2025-07-17	M-a motivat mult!
8	23	1	2025-06-28	Recomand!
9	18	7	2025-06-19	Super curs!
10	25	8	2025-07-10	Foarte clar!
- Output tab shows the same data grid.
- Total rows: 10.

## 14. Tabela SESIUNE INDIVIDUALA

```
CREATE TABLE SESIUNE_INDIVIDUALA(
    id_sesiune INT DEFAULT SESIUNE_SEQ.NEXTVAL PRIMARY KEY,
    id_antrenor INT,
    id_client INT,
    data DATE,
    ora VARCHAR(10),
    FOREIGN KEY (id_antrenor) REFERENCES ANTRENOR(id_antrenor),
    FOREIGN KEY (id_client) REFERENCES CLIENT(id_client)
);
```

```
INSERT INTO SESIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)
VALUES (21, 5, TO_DATE('2025-06-01', 'YYYY-MM-DD'), '10:30');
```

```
INSERT INTO SESIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)
VALUES (22, 5, TO_DATE('2025-07-05', 'YYYY-MM-DD'), '11:00');
```

```
INSERT INTO SESIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)
VALUES (23, 6, TO_DATE('2025-06-03', 'YYYY-MM-DD'), '12:00');
```

```
INSERT INTO SESIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)
VALUES (24, 6, TO_DATE('2025-06-04', 'YYYY-MM-DD'), '17:30');
```

```
INSERT INTO SESIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)
VALUES (25, 6, TO_DATE('2025-08-07', 'YYYY-MM-DD'), '14:45');
```

```
INSERT INTO SESIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)
VALUES (21, 5, TO_DATE('2025-07-16', 'YYYY-MM-DD'), '15:00');
```

```
INSERT INTO SESIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)
VALUES (22, 6, TO_DATE('2025-08-27', 'YYYY-MM-DD'), '16:45');
```

```
INSERT INTO SESIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)
VALUES (23, 5, TO_DATE('2025-06-18', 'YYYY-MM-DD'), '09:00');
```

```
INSERT INTO SESIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)
VALUES (24, 5, TO_DATE('2025-08-09', 'YYYY-MM-DD'), '18:15');
```

```
INSERT INTO SESIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)
VALUES (25, 6, TO_DATE('2025-06-30', 'YYYY-MM-DD'), '19:00');
```

```
SELECT *
```

```
FROM SEIUNE_INDIVIDUALA  
ORDER BY id_seiune;
```

PrintScreen cu rezultatul:

The screenshot shows a Database Explorer interface with a list of tables and sequences under the FILIPDB schema. The SEIUNE\_INDIVIDUALA table is selected, displaying its schema and data. The schema includes columns: id\_seiune (INT, primary key), id\_antrenor (INT), id\_client (INT), data (DATE), and ora (VARCHAR(10)). Foreign keys reference ANTRENOR and CLIENT tables. The data section shows multiple INSERT statements for the table, each specifying values for the columns.

```
CREATE TABLE SEIUNE_INDIVIDUALA(  
    id_seiune INT DEFAULT SEIUNE_SEQ.NEXTVAL PRIMARY KEY,  
    id_antrenor INT,  
    id_client INT,  
    data DATE,  
    ora VARCHAR(10),  
    FOREIGN KEY (id_antrenor) REFERENCES ANTRENOR(id_antrenor),  
    FOREIGN KEY (id_client) REFERENCES CLIENT(id_client)  
);  
  
INSERT INTO SEIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)  
VALUES (1, 1, TO_DATE('2025-06-01', 'YYYY-MM-DD'), '10:30');  
  
INSERT INTO SEIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)  
VALUES (2, 1, TO_DATE('2025-07-05', 'YYYY-MM-DD'), '11:00');  
  
INSERT INTO SEIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)  
VALUES (3, 1, TO_DATE('2025-06-03', 'YYYY-MM-DD'), '12:00');  
  
INSERT INTO SEIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)  
VALUES (4, 1, TO_DATE('2025-06-04', 'YYYY-MM-DD'), '17:30');  
  
INSERT INTO SEIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)  
VALUES (5, 1, TO_DATE('2025-08-07', 'YYYY-MM-DD'), '14:45');  
  
INSERT INTO SEIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)  
VALUES (6, 1, TO_DATE('2025-07-16', 'YYYY-MM-DD'), '15:00');  
  
INSERT INTO SEIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)  
VALUES (7, 1, TO_DATE('2025-08-27', 'YYYY-MM-DD'), '16:45');  
  
INSERT INTO SEIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)  
VALUES (8, 1, TO_DATE('2025-06-18', 'YYYY-MM-DD'), '09:00');
```

Fuciuc Filip-Luca  
Grupa 134

BL bd\_an1\_sem2 Version control

Database Explorer

console [@193.226.51.46] console [@localhost] 2 of 33

FILIPDB ✓ 147 ↻

```
528 INSERT INTO SESIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)
529 VALUES (id_antrenor_23, id_client_6, data TO_DATE('2025-06-03', 'YYYY-MM-DD'), ora '12:00');
530
531 INSERT INTO SESIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)
532 VALUES (id_antrenor_24, id_client_6, data TO_DATE('2025-06-04', 'YYYY-MM-DD'), ora '17:30');
533
534 INSERT INTO SESIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)
535 VALUES (id_antrenor_25, id_client_6, data TO_DATE('2025-08-07', 'YYYY-MM-DD'), ora '14:45');
536
537 INSERT INTO SESIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)
538 VALUES (id_antrenor_21, id_client_5, data TO_DATE('2025-07-16', 'YYYY-MM-DD'), ora '15:00');
539
540 INSERT INTO SESIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)
541 VALUES (id_antrenor_22, id_client_6, data TO_DATE('2025-08-27', 'YYYY-MM-DD'), ora '16:45');
542
543 INSERT INTO SESIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)
544 VALUES (id_antrenor_23, id_client_5, data TO_DATE('2025-06-18', 'YYYY-MM-DD'), ora '09:00');
545
546 INSERT INTO SESIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)
547 VALUES (id_antrenor_24, id_client_5, data TO_DATE('2025-08-09', 'YYYY-MM-DD'), ora '18:15');
548
549 INSERT INTO SESIUNE_INDIVIDUALA (id_antrenor, id_client, data, ora)
550 VALUES (id_antrenor_25, id_client_6, data TO_DATE('2025-06-30', 'YYYY-MM-DD'), ora '19:00');
551
552 SELECT *
553 FROM SESIUNE_INDIVIDUALA
554 ORDER BY id_sesiune;
```

The screenshot shows the DataGrip IDE interface with the following details:

- Database Explorer:** Shows connections to `@193.226.51.46` and `@localhost`. The `FILIPDB` database is selected.
- Console:** Displays the following SQL query:

```
SELECT *
FROM SESIUNE_INDIVIDUALA
ORDER BY id_sesiune;
```
- Services:** Shows a table named `FILIPDB.SESIUNE_INDIVIDUALA` with the following data:

ID_SESIUNE	ID_ANTRENOR	ID_CLIENT	DATA	ORA
1	1	21	5 2025-06-01	10:30
2	2	22	5 2025-07-05	11:00
3	3	23	6 2025-06-03	12:00
4	4	24	6 2025-06-04	17:30
5	5	25	6 2025-08-07	14:45
6	6	21	5 2025-07-16	15:00
7	7	22	6 2025-08-27	16:45
8	8	23	5 2025-06-18	09:00
9	9	24	5 2025-08-09	18:15
10	10	25	6 2025-06-30	19:00

Bottom status bar: 546:25 LF UTF-8 4 spaces

12. Formulati in limbaj natural si implementati 5 cereri SQL complexe ce vor utiliza, in ansamblul lor, urmatoarele elemente:

- a) subcereri sincronizate in care intervin cel putin 3 tabele
- b) subcereri nesincronizate in clauza FROM
- c) grupari de date, functii grup, filtrare la nivel de grupuri cu subcereri nesincronizate (in clauza de HAVING) in care intervin cel putin 3 tabele (in cadrul aceleiasi cereri)
- d) ordonari si utilizarea functiilor NVL si DECODE (in cadrul aceleiasi cereri)
- e) utilizarea a cel putin 2 functii pe siruri de caractere, 2 functii pe date calendaristice, a cel putin unei expresii CASE
- f) utilizarea a cel putin 1 bloc de cerere (clauza WITH)

### Cererea 1:

Formulare in limbaj natural:

Afisati toate salile de antrenament (ID-ul si capacitatea acestora), impreuna cu numele antrenorilor care au primit feedback pozitiv (mesaje ce contin cuvintele "super" sau "excellent"). Salile selectate trebuie sa fie cele in care s-au desfasurat cursuri sustinute de acesti antrenori.

Rezolvare:

```
SELECT DISTINCT S.id_sala, S.capacitate, A.nume AS Antrenor
FROM SALA S
JOIN (
    SELECT F.id_antrenor
    FROM FEEDBACK F
    WHERE LOWER(F.mesaj) LIKE '%super%' OR LOWER(F.mesaj) LIKE '%excellent%'
) FB_A ON S.id_sala IN (
    SELECT DISTINCT P.id_curs
```

```
FROM PROGRAMARE_CURS P
WHERE P.id_antrenor = FB_A.id_antrenor
)
JOIN ANTRENOR A ON FB_A.id_antrenor = A.id_antrenor;
```

PrintScreen cu rezultatul:

```
--12--
-- 1
SELECT DISTINCT S.id_sala, S.capacitate, A.nume AS Antrenor
FROM SALA S
JOIN (
    SELECT F.id_antrenor
    FROM FEEDBACK F
    WHERE LOWER(F.mesaj) LIKE '%super%' OR LOWER(F.mesaj) LIKE '%excellent%'
) FB_A ON S.id_sala IN (
    SELECT DISTINCT P.id_curs
    FROM PROGRAMARE_CURS P
    WHERE P.id_antrenor = FB_A.id_antrenor
)
JOIN ANTRENOR A ON FB_A.id_antrenor = A.id_antrenor;
```

Se respecta urmatoarele cerinte:

- subcereri nesincronizate in clauza FROM

## Cererea 2:

Sa se obtina o statistica pentru fiecare antrenor care a sustinut cursuri in luna iunie 2025. Pentru fiecare antrenor, se va afisa numele sau (cu prima litera mare), varsta medie a clientilor care au participat la cursurile sale in acea luna, numarul de cursuri diferite sustinute, data ultimei programari sustinute in acea luna, numarul de clienti care au un abonament activ (la data de 15 iunie 2025) intr-o sa in care antrenorul a sustinut cursuri, precum si

nivelul de experienta al antrenorului (stabilit in functie de anii de experienta: daca are peste 5 ani este experimentat, daca are sub 2 ani este considerat nou, iar in celelalte cazuri se considera ca are experienta medie). In raport vor fi inclusi doar antrenorii care au cel putin 3 clienti cu abonamente active in salile in care predau. Rezultatele vor fi ordonate descrescator in functie de numarul de clienti cu abonamente active.

Rezolvare:

```
WITH abonamente_active AS (
    SELECT I.id_client, I.id_sala
    FROM ISTORIC I
    WHERE TO_DATE('2025-06-15', 'YYYY-MM-DD') BETWEEN I.data_inceput AND
I.data_expirare
)
SELECT
    INITCAP(A.num) AS nume_antrenor,
    ROUND(AVG(C.varsta), 2) AS varsta_medie_clienti,
    COUNT(DISTINCT PC.id_curs) AS nr_cursuri_sustinute,
    TO_CHAR(MAX(PC.data), 'DD-MON-YYYY') AS data_ultima_programare,
    NVL((
        SELECT COUNT(DISTINCT AA.id_client)
        FROM abonamente_active AA
        WHERE AA.id_sala IN (
            SELECT DISTINCT S.id_sala
            FROM CURS CU
            JOIN SALA S ON CU.id_sala = S.id_sala
            WHERE CU.id_curs IN (
                SELECT PC2.id_curs FROM PROGRAMARE_CURS PC2 WHERE
PC2.id_antrenor = A.id_antrenor
            )
        )
    ), 0) AS nr_clienti_abonati,
    CASE
        WHEN AC.an_experienta > 5 THEN 'Experimentat'
        WHEN AC.an_experienta < 2 THEN 'Nou'
        ELSE 'Experienta medie'
    END AS nivel_experienta
FROM ANTRENOR A
JOIN ANTRENOR_CURS AC ON A.id_antrenor = AC.id_antrenor
JOIN PROGRAMARE_CURS PC ON A.id_antrenor = PC.id_antrenor
JOIN COMPONENTA_CLASA CC ON PC.id_programare = CC.id_programare
JOIN CLIENT C ON CC.id_client = C.id_client
```

```

WHERE TO_CHAR(PC.data, 'MM-YYYY') = '06-2025'
GROUP BY A.id_antrenor, A.nume, AC.ani_experienta
HAVING (
    SELECT COUNT(DISTINCT AA.id_client)
    FROM abonamente_active AA
    WHERE AA.id_sala IN (
        SELECT DISTINCT S.id_sala
        FROM CURS CU
        JOIN SALA S ON CU.id_sala = S.id_sala
        WHERE CU.id_curs IN (
            SELECT PC2.id_curs FROM PROGRAMARE_CURS PC2 WHERE
PC2.id_antrenor = a.id_antrenor
        )
    )
) >= 3
ORDER BY nr_clienti_abonati DESC;

```

PrintScreen cu rezultatul:

```

--2
--Sa se obtina o statistica pentru fiecare antrenor care a sustinut cursuri in luna iunie 2025. Pentru fiecare antrenor, se va afisa
WITH abonamente_active AS (
    SELECT I.id_client, I.id_sala
    FROM ISTORIC I
    WHERE TO_DATE('2025-06-15', 'YYYY-MM-DD') BETWEEN I.data_inceput AND I.data_expirare
)
SELECT
    INITCAP(A.nume) AS nume_antrenor,
    ROUND(AVG(C.varsta), 2) AS varsta_medie_clienti,
    COUNT(DISTINCT PC.id_curs) AS nr_cursuri_sustinute,
    TO_CHAR(MAX(PC.data), 'DD-MON-YYYY') AS data_ultima_programare,
    NVL(
        SELECT COUNT(DISTINCT AA.id_client)
        FROM abonamente_active AA
        WHERE AA.id_sala IN (
            SELECT DISTINCT S.id_sala
            FROM CURS CU
            JOIN SALA S ON CU.id_sala = S.id_sala
            WHERE CU.id_curs IN (
                SELECT PC2.id_curs FROM PROGRAMARE_CURS PC2 WHERE PC2.id_antrenor = A.id_antrenor
            )
        ),
        0
    ) AS nr_clienti_abonati,
    CASE
        WHEN AC.ani_experienta > 5 THEN 'Experimentat'
        WHEN AC.ani_experienta < 2 THEN 'Nou'
        ELSE 'Experienta medie'
    END AS nivel_experienta
FROM ANTRENOR A
JOIN ANTRENOR_CURS AC ON A.id_antrenor = AC.id_antrenor
JOIN PROGRAMARE_CURS PC ON A.id_antrenor = PC.id_antrenor
JOIN COMPOUNTA_CLASA CC ON PC.id_programare = CC.id_programare

```

Fuciuc Filip-Luca  
Grupa 134

```

SELECT PC2.id_curs FROM PROGRAMARE_CURS PC2 WHERE PC2.id_antrenor = A.id_antrenor
)
),
B) AS nr_clienti_abonati,
CASE
    WHEN AC.anii_experienta > 5 THEN 'Experientat'
    WHEN AC.anii_experienta < 2 THEN 'Nou'
    ELSE 'Experienta medie'
END AS nivel_experiencia
FROM ANTRENOR A
JOIN ANTRENOR_CURS AC [1..<->1] ON A.id_antrenor = AC.id_antrenor
JOIN PROGRAMARE_CURS PC [1..<->1..n] ON A.id_antrenor = PC.id_antrenor
JOIN COMPONENTA_CLASA CC [1..n->1..n] ON PC.id_programare = CC.id_programare
JOIN CLIENT C [1..n->1..n] ON CC.id_client = C.id_client
WHERE TO_CHAR(PC.data, 'MM-YYYY') = '06-2025'
GROUP BY A.id_antrenor, A.numar, AC.anii_experienta
HAVING (
    SELECT COUNT(DISTINCT AA.id_client)
    FROM abonamente_active AA
    WHERE AA.id_sala IN (
        SELECT DISTINCT S.id_sala
        FROM CURS CU
        JOIN SALA S [1..n->1..1] ON CU.id_sala = S.id_sala
        WHERE CU.id_curs IN (
            SELECT PC2.id_curs FROM PROGRAMARE_CURS PC2 WHERE PC2.id_antrenor = a.id_antrenor
        )
    )
) >= 3
ORDER BY nr_clienti_abonati DESC;

```

```

-- 1
-Afisati toate salile de antrenament (ID-ul si capacitatea acestora), impreună cu numele antrenorilor care au primit feedback pozitiv

```

	NUME_ANTRENOR	VARSTA_MEDIE_CLIENTI	NR_CURSURI_SUSTINUTE	DATA_ULTIMA_PROGRAMARE	NR_CLIENTI_ABONATI
1	Popescu Andrei	29	2	02-JUN-2025	
2	Vasilescu George	34	2	06-JUN-2025	
3	Dumitrescu Mihai	35	2	10-JUN-2025	
4	Stanescu Ioana	29.5	2	08-JUN-2025	
5	Ionescu Maria	29.5	2	04-JUN-2025	

```
-- 1
-- Afisati toate salile de antrenament (ID-ul si capacitatea acestora), impreună cu numele antrenorilor care au primit feedback pozitiv
SELECT DISTINCT S.id_sala, S.capacitate, A.nume AS Antrenor
FROM SALA S
JOIN (
    SELECT F.id_antrenor
    FROM EFERIREAKT F
) AS F
ON S.id_sala = F.id_sala
WHERE F.id_antrenor IN (
    SELECT id_antrenor
    FROM FEEDBACK
    WHERE feedback = 'pozitiv'
)
```

Services

	Output	Sa se obtina o stat...cu abonamente active.			
Database	TA_MEDIE_CLIENTII	NR_CURSURI_SUSTINUTE	DATA_ULTIMA_PROGRAMARE	NR_CLIENTI_ABONATI	NIVEL_EXPERIENTA
@localhost	1	29	2 02-JUN-2025	4	Experienta medie
console	2	34	2 06-JUN-2025	4	Experienta medie
FEEDBACK	3	35	2 10-JUN-2025	4	Experienta medie
ISTORIC	4	29.5	2 08-JUN-2025	4	Experimentat
CLIENT	5	29.5	2 04-JUN-2025	4	Experimentat

Database Consoles > @localhost > console [localhost]

Se respecta urmatoarele cerinte:

- utilizarea a cel putin 1 bloc de cerere (clauza WITH)
- grupari de date cu subcereri nesincronizate in care intervin cel putin 3 tabele, functii grup (AVG, COUNT, MAX), filtrare la nivel de grupuri (HAVING)
- utilizarea unei functii pe siruri de caractere (INITCAP), 2 functii pe date calendaristice (TO\_CHAR, TO\_DATE), a cel putin unei expresii CASE
- ordonarea datelor si utilizarea functiei NVL

### Cererea 3:

Se cere sa se afiseze pentru fiecare curs: denumirea cursului, nivelul de dificultate, numarul de participanti, rata medie de prezenta calculata procentual, numele si email-ul antrenorului (scris cu litere mici), precum si denumirea si capacitatea salii in care se tine cursul.

Rezolvare:

```
SELECT
    statistici_curs.denumire_curs,
    statistici_curs.nivel_dificultate,
```

```
statistici_curs.numar_participanti,
ROUND(statistici_curs.rata_presenza) AS procent_presenza,
informatii_antrenor.nume_antrenor,
LOWER(informatii_antrenor.email_antrenor) AS mail_antrenor,
informatii_sala.nume_sala,
informatii_sala.capacitate
FROM (
    SELECT
        C.id_curs,
        C.denumire AS denumire_curs,
        C.nivel_dificultate,
        C.id_sala,
        COUNT(CC.id_client) AS numar_participanti,
        AVG(CC.prezenta_efectiva) * 100 AS rata_presenza
    FROM CURS C
    LEFT JOIN PROGRAMARE_CURS PC ON C.id_curs = PC.id_curs
    LEFT JOIN COMPONENTA_CLASA CC ON PC.id_programare = CC.id_programare
    GROUP BY C.id_curs, C.denumire, C.nivel_dificultate, C.id_sala
) statistici_curs
LEFT JOIN (
    SELECT DISTINCT
        PC.id_curs,
        A.nume AS nume_antrenor,
        A.email AS email_antrenor
    FROM PROGRAMARE_CURS PC
    JOIN ANTRENOR A ON PC.id_antrenor = A.id_antrenor
    WHERE A.tip = 'Curs'
) informatii_antrenor ON statistici_curs.id_curs = informatii_antrenor.id_curs
LEFT JOIN (
    SELECT
        S.id_sala,
        S.nume_sala,
        S.capacitate
    FROM SALA S
) informatii_sala ON statistici_curs.id_sala = informatii_sala.id_sala;
```

PrintScreen cu rezultatul:

Fuciuc Filip-Luca  
Grupa 134

The screenshot shows the Microsoft Visual Studio Code interface with the following details:

- Title Bar:** BL bd\_an1\_sem2 > Version control
- Database Explorer:** Shows database connections at 192.26.51.46 and localhost, and a list of tables under the FILIPDB database.
- SQL Console:** The active tab, showing a complex SQL query for calculating participation rates and averages across multiple tables (ANTRENOR, INFORMATII\_ANTRENOR, PROGRAMARE\_CURS, and COMPONENTA\_CLASA).
- Status Bar:** Includes file navigation (Database Consoles > localhost), status indicators (665:25, LF, UTF-8, 4 spaces), and a zoom icon.

```
SELECT
    statistici_curs.denumire_curs,
    statistici_curs.nivel_dificultate,
    statistici_curs.numar_participanti,
    ROUND(statistici_curs.rata_prezenta) AS procent_presenza,
    informatii_antrenor.nume_antrenor,
    LOWER(informatii_antrenor.email_antrenor) AS mail_antrenor,
    informatii_sala.nume_sala,
    informatii_sala.capacitate

FROM (
    SELECT
        C.id_curs,
        C.denumire AS denumire_curs,
        C.nivel_dificultate,
        C.id_sala,
        COUNT(CC.id_client) AS numar_participanti,
        AVG(CC.prezenta_efectiva) * 100 AS rata_presenza
    FROM CURS C
    LEFT JOIN PROGRAMARE_CURS PC 1->0..n ON C.id_curs = PC.id_curs
    LEFT JOIN COMPONENTA_CLASA CC 1->0..n ON PC.id_programare = CC.id_programare
    GROUP BY C.id_curs, C.denumire, C.nivel_dificultate, C.id_sala
) statistici_curs
LEFT JOIN (
    SELECT DISTINCT
        PC.id_curs,
        A.nume AS nume_antrenor,
        A.email AS email_antrenor
    FROM PROGRAMARE_CURS PC
    JOIN ANTRENOR A 1..n->1..1 ON PC.id_antrenor = A.id_antrenor
    WHERE A.tip = 'Curs'
) informatii_antrenor ON statistici_curs.id_curs = informatii_antrenor.id_curs
LEFT JOIN (
```

The screenshot shows the DataGrip IDE interface with the following components:

- Database Explorer** (Left): Shows connections to `@193.226.51.46` and `@localhost`. The `FILIPDB` database is selected, displaying tables: `ANTRENOR`, `ANTRENOR_CUF`, `ANTRENOR_PER`, `CLIENT`, `COMPONENTA`, and `CURS`.
- SQL Editor** (Top Center): Displays a complex SQL query for selecting training programs (CURS) based on antrenor tip ('Curs'). It joins multiple tables: `ANTRENOR`, `INFORMATII_ANTRONOR`, `INFORMATII_CURS`, `SALA`, and `INFORMATII_SALA`.
- Output Tab** (Bottom Center): Shows the results of the executed query. The columns are: `DENUMIRE_CURS`, `NIVEL_DIFFICULTATE`, `NUMAR_PARTICIPANTI`, `PROCENT_PREZENTA`, `NUME_ANTRONOR`, and `MAIL_ANT`. The data is as follows:

DENUMIRE_CURS	NIVEL_DIFFICULTATE	NUMAR_PARTICIPANTI	PROCENT_PREZENTA	NUME_ANTRONOR	MAIL_ANT
Zumba	Incepator	2	100	Popescu Andrei	andrei.p@ex
Pilates	Intermediar	2	50	Popescu Andrei	andrei.p@ex
Pilates	Avansat	2	50	Ionescu Maria	maria.i@exa
Zumba	Avansat	2	100	Ionescu Maria	maria.i@exa
Functional Training	Incepator	2	0	Vasilescu George	george.v@ex
Zumba	Incepator	2	100	Vasilescu George	george.v@ex
Pilates	Intermediar	2	50	Stanescu Ioana	ioana.s@exa
Pilates	Avansat	2	50	Stanescu Ioana	ioana.s@exa
Zumba	Avansat	2	100	Dumitrescu Mihai	mihai.d@exa
Functional Training	Incepator	2	0	Dumitrescu Mihai	mihai.d@exa

Bottom status bar: Database Consoles > @localhost > console [localhost] 10 rows LF UTF-8 4 spaces

The screenshot shows the DBeaver application interface. In the top navigation bar, it says "bd\_an1\_sem2" and "Version control". Below this, the "Database Explorer" tab is selected, showing a tree view of databases and tables. A specific query is being run in the "console [193.226.51.46]" tab:

```

FROM PROGRAMARE_CURS PC
JOIN ANTRENOR A [1..n->1] ON PC.id_antrenor = A.id_antrenor
WHERE A.tip = 'Curs'
) informatii_antrenor ON statistici_curs.id_curs = informatii_antrenor.id_curs
LEFT JOIN (
    SELECT
        S.id_sala,
        S.numar_sala,
        S.capacitate
    FROM SALA S
) informatii_sala ON statistici_curs.id_sala = informatii_sala.id_sala;
  
```

In the bottom right corner of the query window, there are status indicators: 3 rows, 331 bytes, and a warning icon.

The "Services" tab is also visible, showing a table named "Database" with the following data:

	NUMAR_PARTICIPANTI	PROCENT_PREZENTA	NUME_ANTRENOR	MAIL_ANTRENOR	NUME_SALA	CAPACITATE
1	2	100	Popescu Andrei	andrei.p@example.com	Fitness Arena	120
2	2	50	Popescu Andrei	andrei.p@example.com	Body Shape	100
3	2	50	Ionescu Maria	maria.i@example.com	Fitlife	90
4	2	100	Ionescu Maria	maria.i@example.com	GymPro	110
5	2	0	Vasilescu George	george.v@example.com	ActiveZone	85
6	2	100	Vasilescu George	george.v@example.com	Fitness Arena	120
7	2	50	Stanescu Ioana	ioana.s@example.com	Body Shape	100
8	2	50	Stanescu Ioana	ioana.s@example.com	Fitlife	90
9	2	100	Dumitrescu Mihai	mihai.d@example.com	GymPro	110
10	2	0	Dumitrescu Mihai	mihai.d@example.com	ActiveZone	85

The status bar at the bottom right shows "685:25" and "LF UTF-8 4 spaces".

Se respecta urmatoarele cerinte:

- subcereri nesincronizate in clauza FROM (datorita celor 3 subcereri statistici\_curs, informatii\_antrenor, informatii\_sala)
- grupari de date, functii grup (COUNT, AVG), filtrare la nivel de grupuri (GROUP BY) cu subcereri nesincronizate in care intervin cel putin 3 tabele
- utilizarea unei functii pe siruri de caractere (LOWER)

## Cererea 4:

Sa se obtina o lista cu toate programarile de cursuri realizate in luna iunie 2025. Pentru fiecare programare, se vor afisa urmatoarele informatii: denumirea cursului, numele antrenorului, denumirea salii in care se tine cursul, precum si numarul de echipamente in stare buna disponibile in sala respectiva. In cazul in care nu exista echipamente in stare buna, se va afisa valoarea 0. Rezultatele vor fi ordonate descrescator in functie de numarul de echipamente disponibile.

Rezolvare:

```
SELECT DISTINCT
    PC.id_programare,
    C.denumire AS nume_curs,
    A.nume AS antrenor,
    S.nume_sala,
    NVL(COUNT(E.id_echipament), 0) AS nr_echipamente
FROM
    PROGRAMARE_CURS PC
JOIN
    CURS C ON PC.id_curs = C.id_curs
JOIN
    ANTRENOR A ON PC.id_antrenor = A.id_antrenor
JOIN
    SALA S ON C.id_sala = S.id_sala
LEFT JOIN
    ECHIPAMENT E ON S.id_sala = E.id_sala AND LOWER(E.stare) LIKE '%buna%'
WHERE
    TO_CHAR(PC.data, 'MM-YYYY') = '06-2025'
GROUP BY
    PC.id_programare, C.denumire, A.nume, S.nume_sala
ORDER BY
    nr_echipamente DESC;
```

PrintScreen cu rezultatul:

```

SELECT DISTINCT
    PC.id_programare,
    C.denumire AS nume_curs,
    A.num AS antrenor,
    S.num_sala,
    NVL(COUNT(E.id_echipament), 0) AS nr_echipamente
FROM
    PROGRAMARE_CURS PC
JOIN
    CURS C 1..n<=>1: ON PC.id_curs = C.id_curs
JOIN
    ANTRENOR A 1..n<=>1: ON PC.id_antrenor = A.id_antrenor
JOIN
    SALA S 1..n<=>1: ON C.id_sala = S.id_sala
LEFT JOIN
    ECHIPAMENT E ON S.id_sala = E.id_sala AND LOWER(E.stare) LIKE '%buna%'
WHERE
    TO_CHAR(PC.data, 'MM-YYYY') = '06-2025'
GROUP BY
    PC.id_programare, C.denumire, A.num, S.num_sala
ORDER BY
    nr_echipamente DESC;
  
```

```

SELECT DISTINCT
    PC.id_programare,
    C.denumire AS nume_curs,
    A.num AS antrenor,
    S.num_sala,
    NVL(COUNT(E.id_echipament), 0) AS nr_echipamente
FROM
    PROGRAMARE_CURS PC
JOIN
    CURS C 1..n<=>1: ON PC.id_curs = C.id_curs
JOIN
    ANTRENOR A 1..n<=>1: ON PC.id_antrenor = A.id_antrenor
JOIN
    SALA S 1..n<=>1: ON C.id_sala = S.id_sala
LEFT JOIN
    ECHIPAMENT E ON S.id_sala = E.id_sala AND LOWER(E.stare) LIKE '%buna%'
WHERE
    TO_CHAR(PC.data, 'MM-YYYY') = '06-2025'
GROUP BY
    PC.id_programare, C.denumire, A.num, S.num_sala
ORDER BY
    nr_echipamente DESC;
  
```

ID PROGRAMARE	NUME_CURS	ANTRENOR	NUM_SALA	NR_ECHIPAMENTE
1	Zumba	Popescu Andrei	Fitness Arena	1
2	Zumba	Vasilescu George	Fitness Arena	1
3	Pilates	Popescu Andrei	Body Shape	1
4	Pilates	Stanescu Ioana	Body Shape	1
5	Pilates	Ionescu Maria	FitLife	1
6	Zumba	Dumitrescu Mihai	GymPro	1
7	Pilates	Stanescu Ioana	FitLife	1
8	Zumba	Ionescu Maria	GymPro	1
9	Functional Training	Vasilescu George	ActiveZone	0
10	Functional Training	Dumitrescu Mihael	ActiveZone	0

Se respecta urmatoarele cerinte:

- grupari de date, functii grup (COUNT), filtrare la nivel de grupuri (GROUP BY)
- ordonarea inregistrarilor si utilizarea functiei NVL (in cadrul aceleiasi cereri)

## Cererea 5:

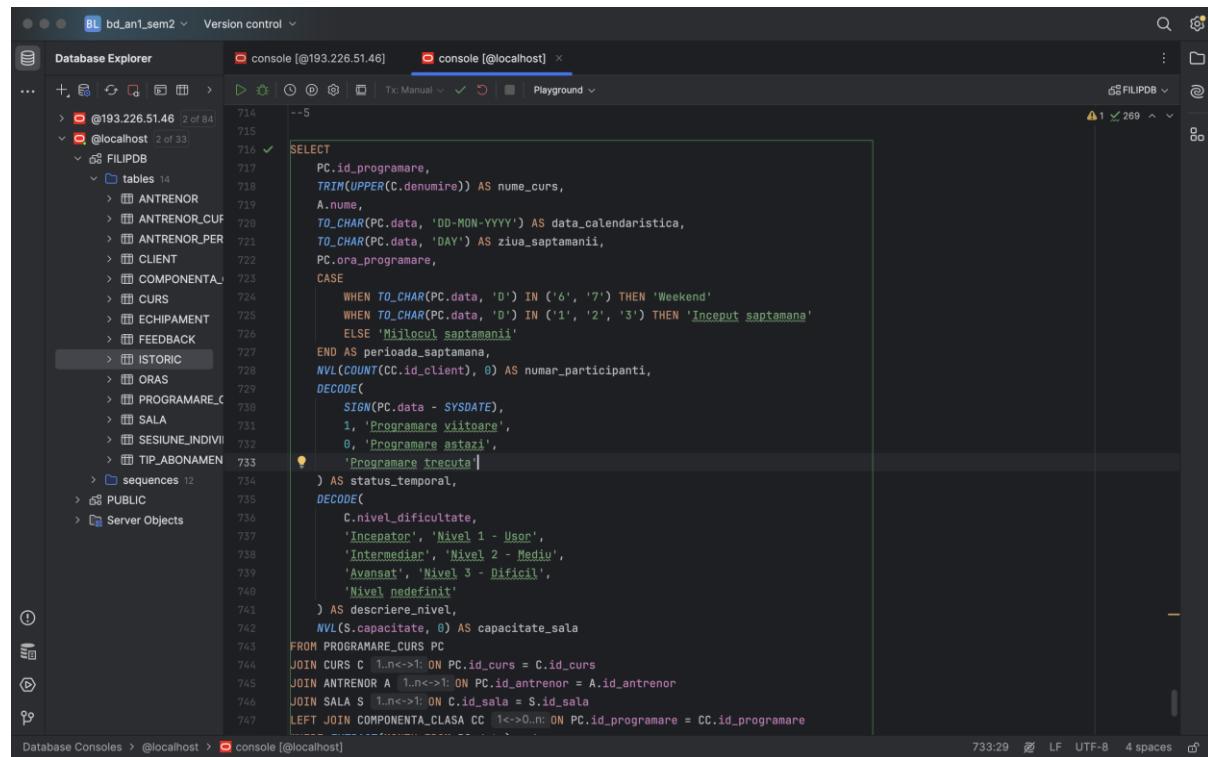
Sa se afiseze detalii despre programarile cursurilor din luna iunie 2025. Pentru fiecare programare, sa se afiseze ID-ul programarii, numele cursului scris cu majuscule si fara spatii suplimentare, numele antrenorului, data programarii in format calendaristic si ziua saptamanii, ora programarii, perioada saptamanii clasificata (Weekend, Inceput de saptamana, Mijlocul saptamanii, numarul participantilor, statusul temporal al programarii (viitoare, de azi sau trecuta), o descriere detaliata a nivelului de dificultate al cursului, precum si capacitatea salii unde se tine programarea. Rezultatele vor fi ordonate crescator dupa data programarii, ora programarii si nivelul de dificultate.

Rezolvare:

```
SELECT
    PC.id_programare,
    TRIM(UPPER(C.denumire)) AS nume_curs,
    A.nume,
    TO_CHAR(PC.data, 'DD-MON-YYYY') AS data_calendaristica,
    TO_CHAR(PC.data, 'DAY') AS ziua_saptamanii,
    PC.ora_programare,
    CASE
        WHEN TO_CHAR(PC.data, 'D') IN ('6', '7') THEN 'Weekend'
        WHEN TO_CHAR(PC.data, 'D') IN ('1', '2', '3') THEN 'Inceput saptamana'
        ELSE 'Mijlocul saptamanii'
    END AS perioada_saptamana,
    NVL(COUNT(CC.id_client), 0) AS numar_participanti,
    DECODE(
        SIGN(PC.data - SYSDATE),
        1, 'Programare viitoare',
        0, 'Programare astazi',
        'Programare trecuta'
    ) AS status_temporal,
    DECODE(
        C.nivel_dificultate,
        'Incepator', 'Nivel 1 - Usor',
        'Intermediar', 'Nivel 2 - Mediu',
        'Avansat', 'Nivel 3 - Dificil',
        'Nivel nedefinit'
    ) AS descriere_nivel,
```

```
NVL(S.capacitate, 0) AS capacitate_sala
FROM PROGRAMARE_CURS PC
JOIN CURS C ON PC.id_curs = C.id_curs
JOIN ANTRENOR A ON PC.id_antrenor = A.id_antrenor
JOIN SALA S ON C.id_sala = S.id_sala
LEFT JOIN COMPONENTA_CLASA CC ON PC.id_programare = CC.id_programare
WHERE EXTRACT(MONTH FROM PC.data) = 6
AND EXTRACT(YEAR FROM PC.data) = 2025
GROUP BY PC.id_programare, C.denumire, A.nume, PC.data, PC.ora_programare,
C.nivel_dificultate, S.capacitate
ORDER BY PC.data,
NVL(PC.ora_programare, '00:00'),
DECODE(C.nivel_dificultate, 'Incepator', 1, 'Intermediar', 2, 'Avansat', 3, 0);
```

PrintScreen cu rezultatul:



```
--5
714
715
716 ✓ SELECT
717     PC.id_programare,
718     TRIM(UPPER(C.denumire)) AS nume_curs,
719     A.nume,
720     TO_CHAR(PC.data, 'DD-MON-YYYY') AS data_calendaristica,
721     TO_CHAR(PC.data, 'DAY') AS ziua_saptamanii,
722     PC.ora_programare,
723     CASE
724         WHEN TO_CHAR(PC.data, 'D') IN ('6', '7') THEN 'Weekend'
725         WHEN TO_CHAR(PC.data, 'D') IN ('1', '2', '3') THEN 'Inceput saptamana'
726         ELSE 'Mijlocul saptamani'
727     END AS perioada_saptamana,
728     NVL(COUNT(CC.id_client), 0) AS numar_participanti,
729     DECODE(
730         SIGN(PC.data - SYSDATE),
731         1, 'Programare viitoare',
732         0, 'Programare astazi',
733         'Programare trecuta'
734     ) AS status_temporal,
735     DECODE(
736         C.nivel_dificultate,
737         'Incepator', 'Nivel 1 - Usor',
738         'Intermediar', 'Nivel 2 - Mediu',
739         'Avansat', 'Nivel 3 - Dificil',
740         'Nivel nedefinit'
741     ) AS descriere_nivel,
742     NVL(S.capacitate, 0) AS capacitate_sala
743
744     FROM PROGRAMARE_CURS PC
745     JOIN CURS C 1..n->1: ON PC.id_curs = C.id_curs
746     JOIN ANTRENOR A 1..n->1: ON PC.id_antrenor = A.id_antrenor
747     JOIN SALA S 1..n->1: ON C.id_sala = S.id_sala
748     LEFT JOIN COMPONENTA_CLASA CC 1<->0..n: ON PC.id_programare = CC.id_programare
```

The screenshot shows the Visual Studio Code interface with the following components:

- Database Explorer** pane on the left, showing database connections and tables.
- SQL Editor** pane in the center, displaying an SQL query:

```
    759      'Avansat', 'Nivel 5 - Utilicil',
    748      'Nivel nedefinit'
    741  ) AS descriere_nivel,
    742  NVL(S.capacitate, 0) AS capacitate_sala
FROM PROGRAMARE_CURS PC
JOIN CURS C 1..n<->1..n ON PC.id_curs = C.id_curs
JOIN ANTRENOR A 1..n<->1..n ON PC.id_antrenor = A.id_antrenor
JOIN SALA S 1..n<->1..n ON C.id_sala = S.id_sala
LEFT JOIN COMPONENTA_CLASA CC 1->0..n ON PC.id_programare = CC.id_programare
WHERE EXTRACT(MONTH FROM PC.data) = 6
AND EXTRACT(YEAR FROM PC.data) = 2025
GROUP BY PC.id_programare, C.denumire, A.nume, PC.data, PC.ora_programare,
        C.nivel_dificultate, S.capacitate
ORDER BY PC.data,
        NVL(PC.ora_programare, '00:00'),
        DECODE(C.nivel_dificultate, 'Incepator', 1, 'Intermediar', 2, 'Avansat', 3, 0);
```

- Services** pane at the bottom, showing a table of program activities.

ID_PROGRAMARE	NUME_CURS	NUME	DATA_CALENDARISTICA	ZIUA_SAPTMANII	ORA_PROGRAMARE
1	ZUMBA	Popescu Andrei	01-JUN-2025	SUNDAY	10:00
2	PILATES	Popescu Andrei	02-JUN-2025	MONDAY	12:00
3	PILATES	Ionescu Maria	03-JUN-2025	TUESDAY	14:00
4	ZUMBA	Ionescu Maria	04-JUN-2025	WEDNESDAY	16:00
5	FUNCTIONAL TRAINING	Vasilescu George	05-JUN-2025	THURSDAY	18:00
6	ZUMBA	Vasilescu George	06-JUN-2025	FRIDAY	10:00
7	PILATES	Stanescu Ioana	07-JUN-2025	SATURDAY	12:00
8	PILATES	Stanescu Ioana	08-JUN-2025	SUNDAY	14:00
9	ZUMBA	Dumitrescu Mihai	09-JUN-2025	MONDAY	16:00
10	FUNCTIONAL TRAINING	Dumitrescu Mihai	10-JUN-2025	TUESDAY	18:00

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

- Database Explorer**: Shows connections to `@193.226.51.46` and `@localhost`. The `FILIPDB` schema is expanded, showing tables: `ANTRONOR`, `ANTRONOR_CUF`, `ANTRONOR_PER`, `CLIENT`, `COMPONENTA`, `CURS`, `ECHIPAMENT`, `FEEDBACK`, `ISTORIC`, and `ORAS`.
- SQL Editor**: A complex SQL query is being typed, involving multiple joins and conditions related to `ANTRONOR`, `SALA`, `COMPONENTA`, and `PC` tables.
- Services**: Shows a table with columns: `PERIODA_SAPTMANA`, `NUMAR_PARTICIPANTI`, `STATUS_TEMPORAL`, `DESCRIERE_NIVEL`, and `CAPACITATE_SALA`. The data includes rows for Weekend, Inceput saptamanii, and Mijlocul saptamanii, with various levels of participation and capacity.

**Se respecta urmatoarele cerinte:**

- grupari de date cu functii de grup (COUNT) si filtrare la nivel de grupuri (GROUP BY) in cadrul aceleiasi cereri;
  - utilizarea functiilor NVL si DECODE in cadrul aceleiasi cereri;

- ordonarea inregistrarilor (ORDER BY);
- utilizarea a doua functii pe siruri de caractere (TRIM si UPPER).
- utilizarea unei functii pe date calendaristice (TO\_CHAR, EXTRACT).
- utilizarea unei expresii CASE (pentru perioada\_saptamana).

### 13. Implementarea a 3 operatii de actualizare si de suprimare a datelor utilizand subcereri

Prima operatie:

Sa se scada cu 10 capacitatea salilor (coloana “capacitate” din tabela SALA) in care exista macar un aparat avariat.

```
UPDATE SALA
SET capacitate = capacitate - 10
WHERE id_sala IN (
    SELECT DISTINCT E.id_sala
    FROM ECHIPAMENT E
    WHERE LOWER(E.stare) = 'avariat'
);
```

The screenshot shows a database management interface with the following details:

- Database Explorer:** Shows the structure of the FILIPDB database, including tables like ANTRENOR, ANTRENOR\_CUF, ANTRENOR\_PER, CLIENT, COMPONENTA, CURS, ECHIPAMENT, FEEDBACK, ISTORIC, ORAS, PROGRAMARE, SALA, SESIUNE\_INDIVII, and TIP\_ABONAMENT.
- Console:** Displays the execution of a SQL UPDATE statement:

```
UPDATE SALA
SET capacitate = capacitate - 10
WHERE id_sala IN (
    SELECT DISTINCT e.id_sala
    FROM ECHIPAMENT e
    WHERE LOWER(e.stare) = 'avariat'
);
```
- Services:** Shows the status of the Database service, indicating it is running on port 193.226.51.46.
- Logs:** Shows the log output for the update operation, indicating 2 rows affected in 99 ms.

## A doua operatie:

Sa se stearga toate tipurile de abonament pe care nu le-a cumparat nimeni.

```
DELETE FROM TIP_ABONAMENT
WHERE id_tip_abonament NOT IN (
    SELECT DISTINCT id_tip_abonament
    FROM ISTORIC
);
```

The screenshot shows a database management interface with the following details:

- Database Explorer:** Shows the schema of the `FILIPDB` database, including tables like `ANTRENOR`, `CLIENT`, `COMPONENTA`, `CURS`, `ECHIPAMENT`, `FEEDBACK`, `ISTORIC`, `ORAS`, `PROGRAMARE`, `SALA`, `SESIUNE_INDIVI`, `TIP_ABONAMENT`, and sequences.
- Script Editor:** Displays a SQL script with two main sections:
  - Section 1 (lines 766-778): An `UPDATE SALA` statement setting `SET capacitate = capacitate - 10` for rows where `id_sala` is in a subquery selecting distinct `id_sala` from `ECHIPAMENT` where `LOWER(E.stare) = 'avarist'`.
  - Section 2 (lines 779-785): A `DELETE FROM TIP_ABONAMENT` statement where `id_tip_abonament` is not in a subquery selecting distinct `id_tip_abonament` from `ISTORIC`.
- Services:** Shows a database connection to `FILIPDB` at `193.226.51.46` with a status of `OK`. The command history shows the execution of the delete query from the script editor, resulting in 5 rows affected in 61 ms.
- Console:** Shows the command `FILIPDB> DELETE FROM TIP_ABONAMENT WHERE id_tip_abonament NOT IN ( SELECT DISTINCT id_tip_abonament FROM ISTORIC )` and its execution results.

## A treia operatie:

Sa se actualizeze nivelul de dificultate ca fiind pentru incepatori pentru cursurile la care media de prezenta efectiva este de sub o persoana, pentru a le face mai atractive pentru cei incepatori.

```
UPDATE CURS
SET nivel_dificultate = 'Incepator'
WHERE id_curs IN (
    SELECT C.id_curs
    FROM CURS C
    JOIN PROGRAMARE_CURS PC ON C.id_curs = PC.id_curs
    JOIN COMPONENTA_CLASA CC ON PC.id_programare = CC.id_programare
    GROUP BY C.id_curs
    HAVING AVG(CC.prezenta_efectiva) < 1
);
```

The screenshot shows a database management interface with the following details:

- Database Explorer:** Shows a tree view of the FILIPDB schema with tables like ANTRENOR, CURS, and CLIENT.
- Console:** Displays two queries:
  - Query 1 (selected): Deletes rows from TIP\_ABONAMENT where id\_tip\_abonament is not in a list of distinct values from ISTORIC.
  - Query 2 (commented out): Updates CURS table based on Curs.C.id\_curs, joining with PROGRAMARE\_CURS, COMPONENTA\_CLASA, and GROUPING BY C.id\_curs having an average value less than 1.
- Services:** Shows a list of databases and their connections, with FILIPDB selected.
- Console Output:** Shows the execution results of the second query, indicating 3 rows affected in 38 ms.

14 .Crearea unei vizualizari complexe. Dati un exemplu de operatie LMD permisa pe vizualizarea respectiva si un exemplu de operatie LMD nepermisa

Sa se creeze o vizualizare care sa afiseze, pentru fiecare programare de curs, denumirea cursului, numele antrenorului, sala, data programarii si numarul de participanti inscrisi (inclusiv 0 pentru programarile fara participanti).

```
CREATE OR REPLACE VIEW v_statistici_cursuri AS
SELECT
    PC.id_programare,
    C.denumire AS nume_curs,
    A.nume AS nume_antrenor,
    S.nume_sala,
    TO_CHAR(PC.data, 'DD-MON-YYYY') AS data_programare,
```

```

NVL(COUNT(CC.id_client), 0) AS numar_participanti
FROM PROGRAMARE_CURS PC
JOIN CURS C ON PC.id_curs = C.id_curs
JOIN ANTRENOR A ON PC.id_antrenor = A.id_antrenor
JOIN SALA S ON C.id_sala = S.id_sala
LEFT JOIN COMPONENTA_CLASA CC ON PC.id_programare = CC.id_programare
GROUP BY PC.id_programare, C.denumire, A.nume, S.nume_sala, PC.data;

```

PrintScreen cu rezultatul:

```

CREATE OR REPLACE VIEW v_statistici_cursuri AS
SELECT
    PC.id_programare,
    C.denumire AS nume_curs,
    A.nume AS nume_antrenor,
    S.nume_sala,
    TO_CHAR(PC.data, 'DD-MON-YYYY') AS data_programare,
    NVL(COUNT(CC.id_client), 0) AS numar_participanti
FROM PROGRAMARE_CURS PC
JOIN CURS C ON PC.id_curs = C.id_curs
JOIN ANTRENOR A ON PC.id_antrenor = A.id_antrenor
JOIN SALA S ON C.id_sala = S.id_sala
LEFT JOIN COMPONENTA_CLASA CC ON PC.id_programare = CC.id_programare
GROUP BY PC.id_programare, C.denumire, A.nume, S.nume_sala, PC.data;

```

The screenshot shows a database management interface with the following details:

- Database Explorer:** Shows the schema structure of the FILIPDB database, including tables (14), views (1), sequences (12), and other objects.
- Query Results:** A grid displaying data from the V\_STATISTICI\_CURSURI view. The columns are: ID\_PROGRAMARE, NUME\_CURS, NUME\_ANTRENOR, NUME\_SALA, DATA\_PROGRAMARE, and NUMAR\_PARTICIPANTI. The data includes rows for Zumba, Pilates, Functional Training, and other exercises across various studios and dates.
- Services:** A panel showing active sessions and their queries. One session at localhost is executing a SELECT statement from the V\_STATISTICI\_CURSURI view, filtering by ROWNUM <= 501.
- Bottom Navigation:** Shows the full path: Database > @localhost > FILIPDB > views > V\_STATISTICI\_CURSURI.

Exemplu de operatie LMD permisa:

```
SELECT * FROM v_statistici_cursuri WHERE nume_curs = 'Zumba';
```

Aceasta operatie este permisa, deoarece este doar o cerere de citit care combina date din mai multe tabele. Astfel, doar citim date prelucrate deja de catre vizualizare si nu modificam nimic.

The screenshot shows the DBeaver application interface. In the Database Explorer pane, there is a tree view of database objects under the 'FILIPDB' schema, including tables, views, sequences, and public objects. The 'V\_STATISTICI\_CURSURI' view is selected. In the SQL Editor pane, two queries are shown:

```

CREATE OR REPLACE VIEW v_statistici_cursuri AS
SELECT *
  FROM cursuri
  LEFT JOIN COMPOENTA_CLASA CC 1<->0..n ON PC.id_programare = CC.id_programare
 GROUP BY PC.id_programare, C.denumire, A.nume, S.nume_sala, PC.data;
SELECT * FROM v_statistici_cursuri WHERE nume_curs = 'Zumba';

```

In the Services pane, a table named 'FILIPDB.V\_STATISTICI\_CURSURI' is displayed with the following data:

ID_PROGRAMARE	NUME_CURS	NUME_ANTRENOR	NUME_SALA	DATA_PROGRAMARE	NUMAR_PARTICIPANTI
1	Zumba	Ionescu Maria	GymPro	04-JUN-2025	1
2	Zumba	Popescu Andrei	Fitness Arena	01-JUN-2025	1
3	Zumba	Vasilescu George	Fitness Arena	06-JUN-2025	1
4	Zumba	Dumitrescu Mihai	GymPro	09-JUN-2025	1

Exemplu de operatie LMD nepermisa:

```

INSERT INTO v_statistici_cursuri (id_programare, nume_curs, nume_antrenor, nume_sala,
data_programare, numar_participanti)
VALUES (101, 'Pilates', 'Andrei Ionescu', 'GymLife', '15-JUN-2025', 10);

```

Aceasta operatie nu este permisa, intrucat vizualizarea creata anterior foloseste JOIN-uri, functia agregata COUNT si GROUP\_BY. Astfel, nu s-ar sti unde sa se insereze noile date, nici n-am avea cum sa inseram valoarea 10 intr-o coloana unde sunt valori calculate, nu are sens.

```

SELECT * FROM v_statistici_cursuri WHERE nume_curs = 'Zumba';

INSERT INTO v_statistici_cursuri (id_programare, nume_curs, nume_antrenor, nume_sala, data_programare, numar_participanti)
VALUES (101, 'Pilates', 'Andrei Ionescu', 'GymLife', '15-JUN-2025', 10)

```

[42000][1779] ORA-01779: cannot modify a column which maps to a non key-preserved table  
Position: 34

15. Formulati in limbaj natural si implementati in SQL: o cerere ce utilizeaza operatia outer-join pe minimum 4 tabele, o cerere ce utilizeaza operatia division si o cerere care implementeaza analiza top-n

O cerere ce utilizeaza operatia outer-join pe minimum 4 tabele:

Sa se afiseze toate salile, si, daca exista, cursurile desfasurate in ele, alaturi de antrenorii care le predau si feedbackurile primite de acestia.

Rezolvare:

```

SELECT
    S.nume_sala,
    C.denumire AS denumire_curs,
    A.nume AS nume_antrenor,
    F.mesaj AS feedback
FROM SALA S
LEFT JOIN CURS C ON S.id_sala = C.id_sala
LEFT JOIN PROGRAMARE_CURS PC ON C.id_curs = PC.id_curs

```

```
LEFT JOIN ANTRENOR A ON PC.id_antrenor = A.id_antrenor
LEFT JOIN FEEDBACK F ON A.id_antrenor = F.id_antrenor;
```

NUME_SALA	DENUMIRE_CURS	NUME_ANTRENOR	FEEDBACK
Fitness Arena	Zumba	Popescu Andrei	Foarte curat!
Body Shape	Pilates	Popescu Andrei	Foarte curat!
Fitness Arena	Zumba	Popescu Andrei	Excelent!
Body Shape	Pilates	Popescu Andrei	Excelent!
Body Shape	Pilates	Stanescu Ioana	A fost greu, dar util!
Fitlife	Pilates	Stanescu Ioana	A fost greu, dar util!
Fitlife	Pilates	Ionescu Maria	Foarte implicat!
GymPro	Zumba	Ionescu Maria	Foarte implicat!
GymPro	Zumba	Dumitrescu Mihai	M-a motivat mult!
ActiveZone	Functional Training	Dumitrescu Mihai	M-a motivat mult!
ActiveZone	Functional Training	Vasilescu George	Super curs!
Fitness Arena	Zumba	Vasilescu George	Vasilescu George

## O cerere ce utilizeaza operatia division:

Sa se selecteze salile in care toate echipamentele se afla in stare buna.

Rezolvare:

```
SELECT S.id_sala, S.numa_sala
FROM SALA S
WHERE NOT EXISTS (
    SELECT 1
    FROM ECHIPAMENT E
    WHERE E.id_sala = S.id_sala
        AND LOWER(E.stare) != 'buna'
);
```

The screenshot shows a database management interface with the following details:

- Database Explorer:** Shows a tree view of databases, including `@localhost` and `FILIPDB`. `FILIPDB` contains tables like `tables`, `views`, and `sequences`.
- Console Tab:** Contains the following SQL query:

```
858
859
860
861 --2
862
863 ✓ SELECT S.id_sala, S.num_sala
864 FROM SALA S
865 WHERE NOT EXISTS (
866     SELECT 1
867     FROM ECHIPAMENT E
868     WHERE E.id_sala = S.id_sala
869     AND LOWER(E.stare) != 'buna'
870 );
871
872
```
- Output Tab:** Shows the results of the query:

ID_SALA	NUME_SALA
1	2 Body Shape
2	3 FitLife
3	4 GymPro

3 rows
- Services Tab:** Shows database connections and logs.

O cerere care implementeaza analiza top-n:

Sa se afiseze primii 3 antrenori, in functie de numarul de feedbackuri primite.

Rezolvare:

```
SELECT A.id_antrenor, A.nume, COUNT(F.id_feedback) AS numar_feedbackuri
FROM ANTRENOR A
JOIN FEEDBACK F ON A.id_antrenor = F.id_antrenor
GROUP BY A.id_antrenor, A.nume
ORDER BY numar_feedbackuri DESC
FETCH FIRST 3 ROWS ONLY;
```

The screenshot shows a database management system interface with the following details:

- Database Explorer:** Shows the database structure:
  - Host: @localhost (2 of 33)
  - Schema: FILIPDB (14 tables, 1 view, 12 sequences)
  - View: V\_STATISTICLCA (selected)
- SQL Editor:** Displays the following SQL query:

```
870 );  
871  
872  
873  
874  
875 --3  
876  
877 SELECT A.id_antrenor, A.numere, COUNT(F.id_feedback) AS numar_feedbackuri  
878 FROM ANTRENOR A  
879 JOIN FEEDBACK F 1<->1..m ON A.id_antrenor = F.id_antrenor  
880 GROUP BY A.id_antrenor, A.numere  
881 ORDER BY numar_feedbackuri DESC  
882  
883 FETCH FIRST 3 ROWS ONLY;  
884
```
- Services Tab:** Shows the results of the query in a table:

ID_ANTRENOR	NUME	NUMAR_FEEDBACKURI
16	Popescu Andrei	2
24	Anghelescu Ioana	2
19	Stanescu Ioana	1

16. Optimizarea unei cereri, aplicand regulile de optimizare ce deriva din proprietatile operatorilor algebrei relationale. Cererea va fi exprimata prin expresie algebraica, arbore algebraic si limbaj (SQL), atat anterior cat si ulterior optimizarii

### Formulare in limbaj natural:

Sa se afiseze numele clientului, ora programarii, denumirea cursului, numele salii si numele orasului pentru toti clienti care au participat la cursuri programare incepand cu ora 16.

SQL initial:

```

SELECT CL.nume, PC.ora_programare, C.denumire, S.nume_sala, O.nume_oras
FROM PROGRAMARE_CURS PC, COMPONENTA_CLASA CC, CLIENT CL,
CURS C, SALA S, ORAS O
WHERE PC.id_programare = CC.id_programare
AND CC.id_client = CL.id_client
AND C.id_curs = PC.id_curs
AND C.id_sala = S.id_sala
AND S.id_oras = O.id_oras
AND PC.ora_programare >= TO_CHAR(16);

```

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, the Database Explorer lists sessions and their statements. Statement 1013 contains the query shown above. In the bottom-right pane, the results of the query are displayed in a table:

	NUME	ORA_PROGRAMARE	DENUMIRE	NUME_SALA	NUME_ORAS
1	Vasilescu Elena	16:00	Zumba	GymPro	Timisoara
2	Georgescu Mihai	16:00	Zumba	GymPro	Timisoara
3	Dumitru Ana	18:00	Functional Training	ActiveZone	Constanta
4	Dumitru Ana	18:00	Functional Training	ActiveZone	Constanta

Expresie algebraica initiala:

$\Pi$  nume, ora\_programare, denumire, nume\_sala, nume\_oras (

$\sigma$

(

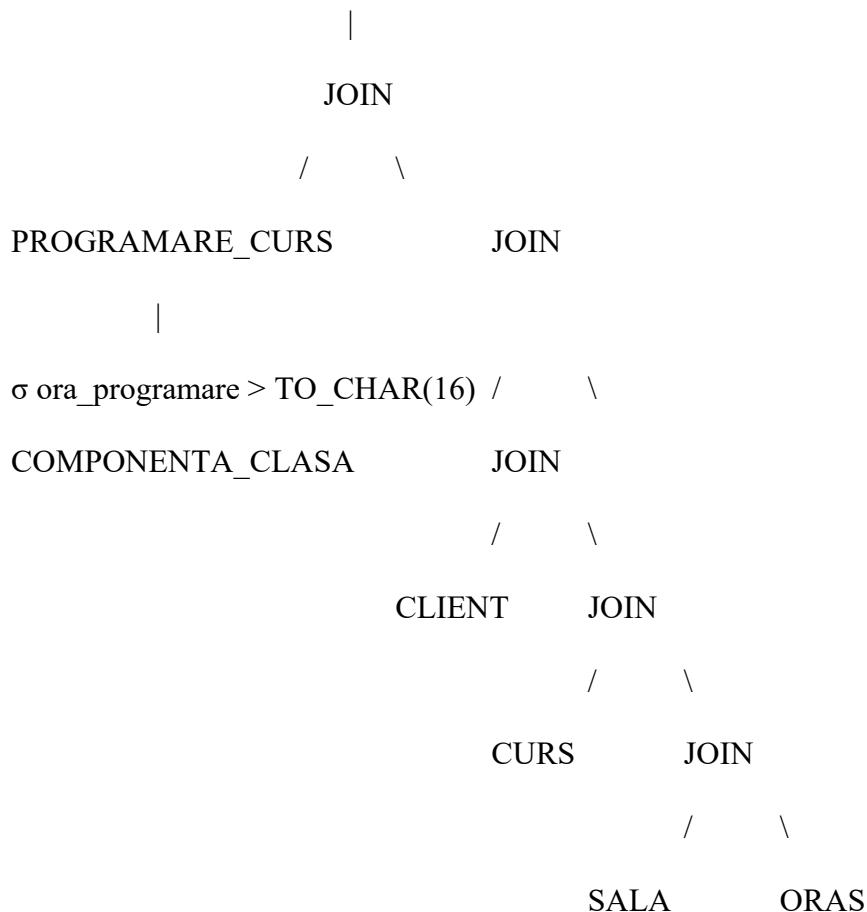
PROGRAMARE\_CURS X COMPONENTA\_CLASA X CLIENT X CURS X SALA X ORAS

)

)

## Arbore algebric initial:

$\Pi$  nume, ora\_programare, denumire, nume\_sala, nume\_oras



## Aplicarea regulilor de optimizare:

Regula 1: Selectiile se executa cat mai devreme posibil

- Separam conditia  $\sigma$  ora\_programare > TO\_CHAR(16)

Regula 2: Produsele carteziene se inlocuiesc cu JOIN-uri

Regula 3: JOIN-ul cel mai restrictiv se executa primul

- Reordonam JOIN-urile pentru eficienta maxima

Regula 4: Proiectiile se executa la inceput

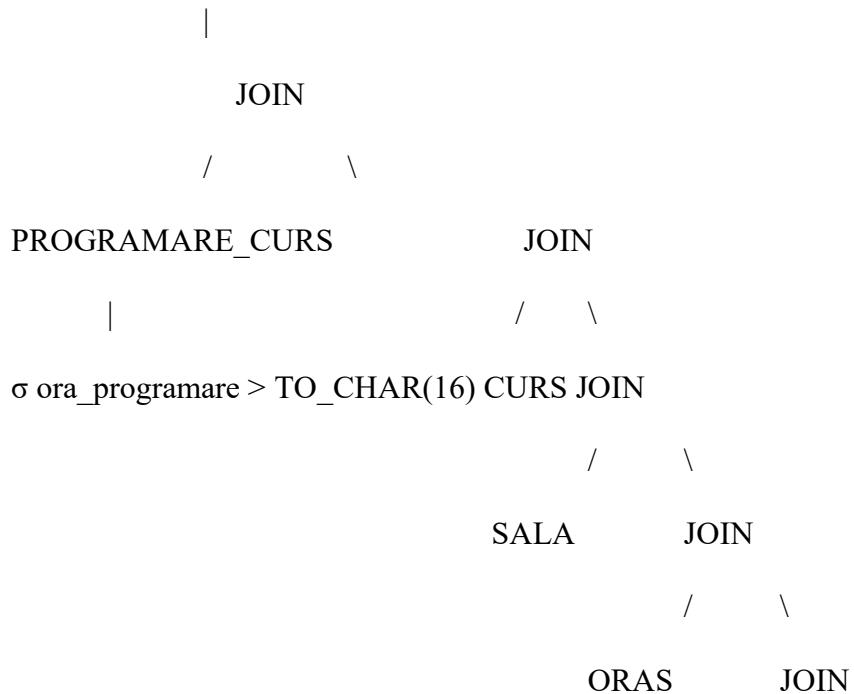
- Eliminam atributele neutilizate din calculele anterioare

## Expresie algebraica optimizata:

```
ΠI nume, ora_programare, denumire, nume_sala, nume_oras (  
JOIN( σ ora_programare > TO_CHAR(16) (PROGRAMARE_CURS)  
      JOIN( CURS,  
            JOIN( SALA,  
                  JOIN( ORAS,  
                        JOIN( COMPONENTA_CLASA, CLIENT))))))  
)
```

## Arbore algebric optimizat:

$\Pi$  nume, ora\_programare, denumire, nume\_sala, nume\_oras



/ \

## COMPONENTA\_CLASA CLIENT

SQL optimizat:

```
SELECT CL.nume, PC.ora_programare, C.denumire, S.num_sala, O.num_oras
FROM PROGRAMARE_CURS PC
JOIN CURS C ON C.id_curs = PC.id_curs AND PC.ora_programare >= TO_CHAR(16)
JOIN SALA S on S.id_sala = C.id_sala
JOIN ORAS O on O.id_oras = S.id_oras
JOIN COMPONENTA_CLASA CC on PC.id_programare = CC.id_programare
JOIN CLIENT CL ON CL.id_client = CC.id_client;
```

	NUME	ORA_PROGRAMARE	DENUMIRE	NUM_SALA	NUM_ORAS
1	Vasilescu Elena	16:00	Zumba	GymPro	Timisoara
2	Georgescu Mihael	16:00	Zumba	GymPro	Timisoara
3	Dumitru Ana	18:00	Functional Training	ActiveZone	Constanta
4	Dumitru Ana	18:00	Functional Training	ActiveZone	Constanta

17. a) Realizarea normalizarii BCNF, FN4, FN5

## BCNF:

O relatie este in forma normala Boyce-Codd (BCNF) daca si numai daca pentru fiecare dependenta functionala ne-triviala de forma  $X \rightarrow Y$  care exista in relatie, X este supercheie. Aceasta inseamna ca atributul sau combinatia de atribute X determina in mod unic toate atributele din relatie, iar orice dependenta functionala ne-triviala trebuie sa aiba partea stanga (X) ca supercheie.

Daca ne imaginam urmatoarea tabela SALA:

```
SALA (
    id_sala PK,
    Id_echipament PK,
    stare_echipament,
    capacitate
)
```

Entitatea respecta FN1 (toate atributele contin doar valori atomice), FN2 (toate atributele depind de cheia primara compusa) si FN3 (nu exista dependente tranzitive). Chiar daca stare\_echipament determina capacitate, nu este tranzitiva, intrucat capacitatea este deja determinata de cheia primara compusa.

Cu toate acestea, se incalca BCNF, deoarece toti determinantii trebuie sa fie super-cheie, iar stare\_echipament este un atribut oarecare. Pentru a rezolva aceasta problema, impartim in doua tabele separate, SALA si ECHIPAMENT.

```
SALA (
    id_sala PK,
    capacitate
)
ECHIPAMENT (
    id_echipament PK,
    stare_echipament
)
```

Astfel, s-a eliminat redundanta si am ajuns sa respectam BCNF.

## FN4:

O relatie este in FN4 daca este in BCNF si nu contine dependente multivaluate non-triviale care sa impuna redundanta. Mai precis, pentru orice dependenta multivaluata non-triviala  $X ->-> Y$  din relatie, X trebuie sa fie o supercheie.

Sa presupunem ca in tabela CLIENT am mai fi avut ca atribute Obiectiv (pentru ce merge la sala) si Accesoriu(cu ce accesorii vine, spre exemplu chingi, benzi cu rezistenta, prize pentru glezna etc). Tabela CLIENT ar fi aratat astfel:

```
CLIENT (
    id_client PK,
    nume,
    varsta,
    obiectiv,
    accesoriu
)
```

Se observa ca, daca un client ar fi avut X accesorii si Y obiective, acestea ar ocupa  $X^*Y$  randuri, ceea ce, in cazul unor baze de date mai mari, este extrem de ineficient si costisitor. Solutia este sa mai adaugam tabelele OBIECTIVE\_CLIENT si ACCESORII\_CLIENT, iar acum vom avea  $X+Y$  inregistrari, in loc de  $X^*Y$ .

```
OBIECTIVE_CLIENT (
```

```
    id_obiectiv PK,
    id_client FK,
    obiectiv
)
```

```
ACCESORII_CLIENT (
```

```
    id_accesoriu PK,
    id_client FK,
    accesoriu
)
```

## FN5:

O relatie este in FN5 daca este in FN4 si nu exista o descompunere posibila a relatiei in mai multe tabele mai mici (bazate pe proiectii) care sa permita refacerea exacta a relatiei originale prin JOIN.

Un exemplu ipotetic se refera la situatia in care as fi vrut sa includ in modelul meu si furnizori de echipamente de fitness. Astfel, instinctul ar fi fost sa construiesc entitatea SALA astfel:

```
SALA (  
    id_sala PK,  
    id_echipament PK,  
    id_furnizor PK  
)
```

Toate cele trei coduri ar fi fost chei primare, intrucat o sala poate avea mai multe echipamente si mai multi furnizori, iar un furnizor poate avea mai multe echipamente si poate colabora cu mai multe sali. Se observa insa ca, separand aceasta tabela in 3 tabele diferite, SALA\_ECHIPAMENT, SALA\_FURNIZOR, si FURNIZOR\_ECHIPAMENT, nu am pierde informatii daca vom da JOIN intre ele. Astfel, se poate elimina redundanta si se poate respecta FN5:

```
SALA_ECHIPAMENT (  
    id_sala PK,  
    id_echipament PK  
)  
  
SALA_FURNIZOR (  
    id_sala PK,  
    id_furnizor PK  
)  
  
FURNIZOR_ECHIPAMENT (  
    id_furnizor PK,  
    id_echipament PK  
)
```

## 17. b) Aplicarea denormalizarii, justificand necesitatea acesteia

Interrogare frecventa: Selecteaza toate programarile la cursuri cu detalii despre programare, sala si cursul respectiv.

In schema normalizata, aceasta interogare ar necesita JOIN-uri intre entitatile PROGRAMARE\_CURS, CURS si SALA. Pentru a optimiza performanta, putem denormaliza aceste tabele.

Schema denormalizata:

PROGRAMARE\_DENORMALIZATA( id\_programare, id\_antrenor, id\_oras, data, ora\_programare, nivel\_dificultate, denumire\_curs, nume\_sala, adresa, capacitate)

SQL:

```
SELECT PC.data, PC.ora_programare,
       C.nivel_dificultate, C.denumire,
       S.nume_sala, S.adresa, S.capacitate
    FROM PROGRAMARE_CURS PC
   JOIN CURS C ON C.id_curs = PC.id_curs
   JOIN SALA S ON S.id_sala = C.id_sala;
```

The screenshot shows the Oracle SQL Developer interface. In the top-left corner, there's a tree view of the database schema under 'Database Explore'. A query is being run against the 'FILIPDB' database, specifically against the 'localhost' connection. The query itself is:

```

SELECT PC.data, PC.ora_programare,
       C.nivel_dificultate, C.denumire,
       S.nume_sala, S.adresa, S.capacitate
  FROM PROGRAMARE_CURS PC
 JOIN CURS C  ON C.id_curs = PC.id_curs
 JOIN SALA S  ON S.id_sala = C.id_sala;
    
```

At the bottom of the query window, there's a 'ROLLBACK ;' command. The results of the query are displayed in a table titled 'Result 5'. The columns are: DATA, ORA\_PROGRAMARE, NIVEL\_DIFICULTATE, DENUMIRE, NUME\_SALA, ADRESA, and CAPACITA. The data consists of 10 rows, each representing a program session with details like date, time, class name, instructor, room name, address, and capacity.

DATA	ORA_PROGRAMARE	NIVEL_DIFICULTATE	DENUMIRE	NUME_SALA	ADRESA	CAPACITA
2025-06-01	10:00	Incepator	Zumba	Fitness Arena	Strada Muncii 23	
2025-06-02	12:00	Incepator	Pilates	Body Shape	Bd. Eroilor 10	
2025-06-03	14:00	Incepator	Pilates	Fitlife	Str. Libertatii 45	
2025-06-04	16:00	Avansat	Zumba	GymPro	Str. Independentei 15	
2025-06-05	18:00	Incepator	Functional Training	ActiveZone	Bd. Tomis 200	
2025-06-06	10:00	Incepator	Zumba	Fitness Arena	Strada Muncii 23	
2025-06-07	12:00	Incepator	Pilates	Body Shape	Bd. Eroilor 10	
2025-06-08	14:00	Incepator	Pilates	Fitlife	Str. Libertatii 45	
2025-06-09	16:00	Avansat		GymPro	Str. Independentei 15	
2025-06-10	18:00	Incepator	Functional Training	ActiveZone	Bd. Tomis 200	

## Justificarea denormalizarii:

- Eliminam necesitatea JOIN-urilor multiple, ceea ce poate imbunatati performanta interogarii
- Crestem viteza de raspuns pentru interogarile frecvente care necesita aceste informatii agregate
- Simplificam schema si interogarile pentru analistii si dezvoltatorii de baze de date