

Academia de Studii Economice

Facultatea de Cibernetică, Statistică și Informatică Economică

**PROIECT BAZE DE DATE**

**Gestiunea datelor dintr-o sala de fitness**

**Matei Tudor-Mihai**

Grupa 1058

Seria D

1. **Obiectivul proiectului**

Aceasta baza de date **centralizeaza si supraveheaza** datele legate de o sala de fitness. Astfel, prin aceasta platforma se poate imbunatatii:

* **Experienta clientilor** prin urmărirea programărilor și a abonamentelor, poți asigura că clienții primesc serviciile pentru care au plătit și că sunt programați în mod corespunzător cu antrenorii sau se poate optimiza
* **Optimizarea programului antrenorilor** prin urmărirea programărilor, poți asigura că antrenorii își maximizează timpul și că sunt disponibili pentru clienți atunci când este nevoie de ei.
* **Urmărirea performanței sălii de fitness** prin urmărirea numărului de clienți, a abonamentelor și a programărilor, poți obține o imagine clară a performanței sălii de fitness.
* **Facilitarea luării deciziilor** cu o imagine clară a datelor, poți lua decizii mai informate despre cum să îți îmbunătățești sala de fitness.

## Descrierea bazei de date

## Relațiile dintre tabele

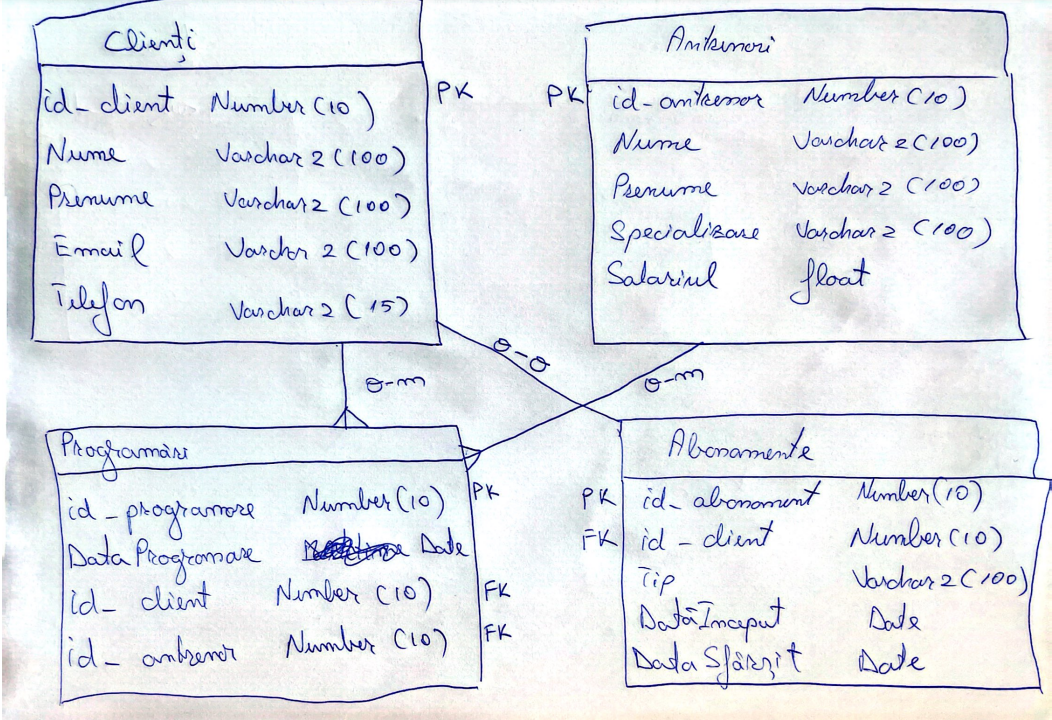
**Tabela Clienti:** Aceasta este o tabelă care conține informații despre clienții sălii de fitness. Fiecare client are un ID unic (ID\_Client), nume, prenume, email și număr de telefon.

**Tabela Antrenori:** Aceasta este o tabelă care conține informații despre antrenorii sălii de fitness. Fiecare antrenor are un ID unic (ID\_Antrenor), nume, prenume și specializare.

**Tabela Programari:** Aceasta este o tabelă care conține informații despre programările făcute de clienți la antrenori. Fiecare programare are un ID unic (ID\_Programare), ID-ul clientului (ID\_Client), ID-ul antrenorului (ID\_Antrenor) și data și ora programării (DataOra). Există o relație **“many-to-many”** între tabelele Clienti și Antrenori prin intermediul acestei tabele.

**Tabela Abonamente:** Aceasta este o tabelă care conține informații despre abonamentele clienților. Fiecare abonament are un ID unic (ID\_Abonament), ID-ul clientului (ID\_Client), tipul abonamentului (Tip), data de început (DataInceput) și data de sfârșit (DataSfarsit). Există o relație **“one-to-one”** între tabelele Clienti și Abonamente, deoarece un client poate doar un abonament.

## Atribute și restricții



**CLIENTI:**

ID\_Client NUMBER(10) PRIMARY KEY,

Nume VARCHAR2(100),

Prenume VARCHAR2(100),

Email VARCHAR2(100),

Telefon VARCHAR2(15)

**ANTRENORI:**

ID\_Antrenor NUMBER(10) PRIMARY KEY,

Nume VARCHAR2(100),

Prenume VARCHAR2(100),

Specializare VARCHAR2(100)

**PROGRAMRI:**

ID\_Programare NUMBER(10) PRIMARY KEY,

ID\_Client NUMBER(10),

ID\_Antrenor NUMBER(10),

DataProgramare DATE,

FOREIGN KEY (ID\_Client) REFERENCES Clienti(ID\_Client),

FOREIGN KEY (ID\_Antrenor) REFERENCES Antrenori(ID\_Antrenor)

**ABONAMETE:**

ID\_Abonament NUMBER(10) PRIMARY KEY,

ID\_Client NUMBER(10),

Tip VARCHAR2(50),

DataInceput DATE,

DataSfarsit DATE,

FOREIGN KEY (ID\_Client) REFERENCES Clienti(ID\_Client)

1. **Descrierea tabelelor și comanda CREATE**

**Baza de date este alcătuită din 4 tabele:**

**1.**

DROP TABLE Clienti CASCADE CONSTRAINTS;

DROP TABLE Antrenori CASCADE CONSTRAINTS;

DROP TABLE Programari CASCADE CONSTRAINTS;

DROP TABLE Abonamente CASCADE CONSTRAINTS;

CREATE TABLE Clienti (

ID\_Client NUMBER(10) PRIMARY KEY,

Nume VARCHAR2(100),

Prenume VARCHAR2(100),

Email VARCHAR2(100),

Telefon VARCHAR2(15)

);

CREATE TABLE Antrenori (

ID\_Antrenor NUMBER(10) PRIMARY KEY,

Nume VARCHAR2(100),

Prenume VARCHAR2(100),

Specializare VARCHAR2(100),

Salariul FLOAT

);

CREATE TABLE Programari (

ID\_Programare NUMBER(10) PRIMARY KEY,

ID\_Client NUMBER(10),

ID\_Antrenor NUMBER(10),

DataProgramare DATE,

FOREIGN KEY (ID\_Client) REFERENCES Clienti(ID\_Client),

FOREIGN KEY (ID\_Antrenor) REFERENCES Antrenori(ID\_Antrenor)

);

CREATE TABLE Abonamente (

ID\_Abonament INT PRIMARY KEY,

ID\_Client INT,

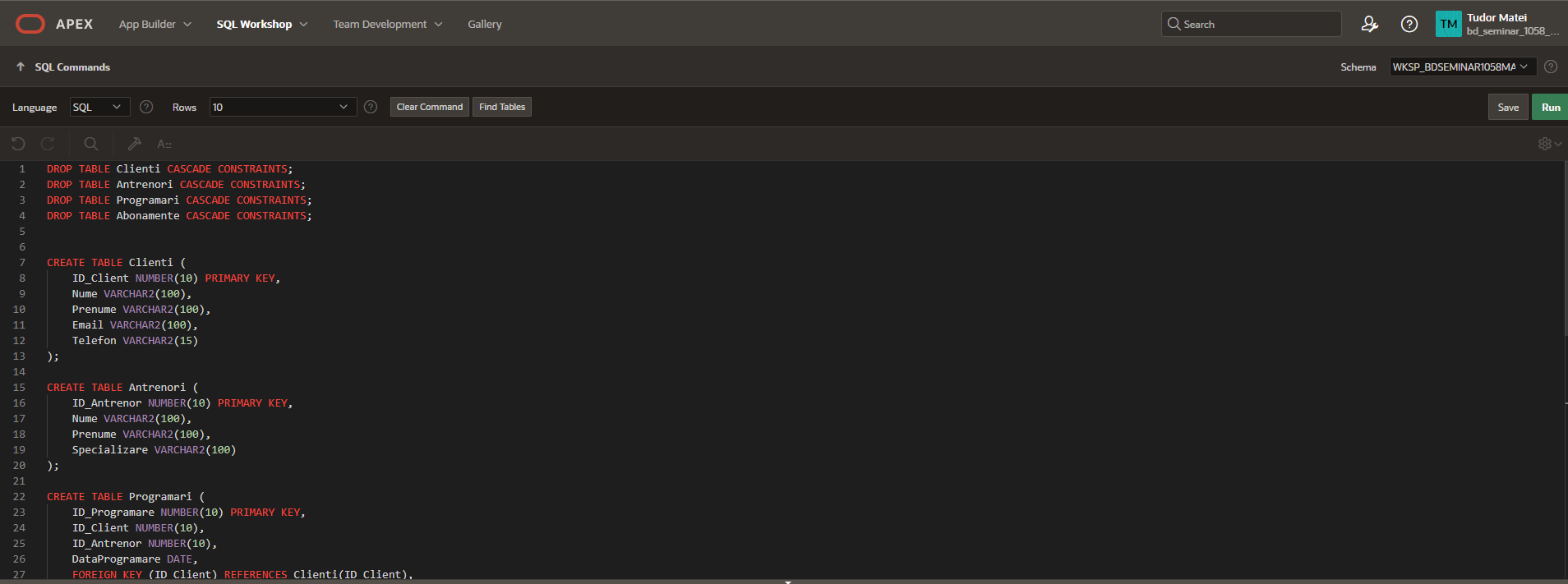
Tip VARCHAR(50),

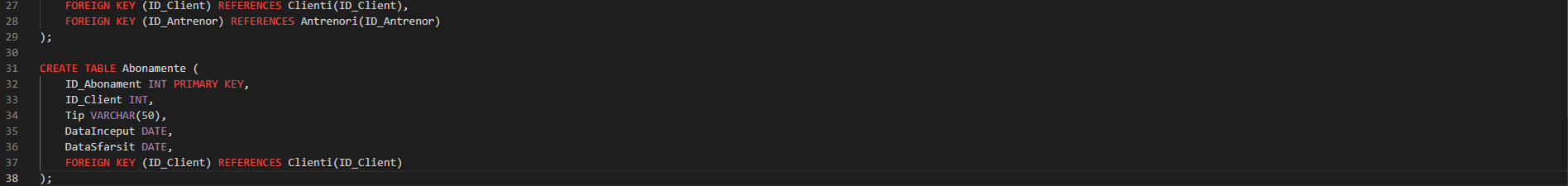
DataInceput DATE,

DataSfarsit DATE,

FOREIGN KEY (ID\_Client) REFERENCES Clienti(ID\_Client)

);

****

****

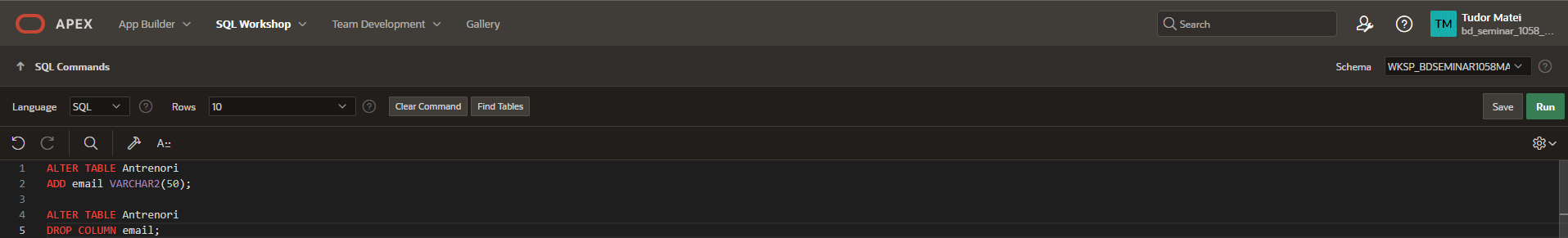
**2.Modificarea tabelului Antrenori**

ALTER TABLE Antrenori

ADD email VARCHAR2(50);

ALTER TABLE Antrenori

DROP COLUMN email;



3.Popularea tabelelor

INSERT INTO Clienti (ID\_Client, Nume, Prenume, Email, Telefon)

VALUES (1, 'Popescu', 'Ion', 'ion.popescu@email.com', '0722000000');

INSERT INTO Clienti (ID\_Client, Nume, Prenume, Email, Telefon)

VALUES (2, 'Ionescu', 'Ana', 'ana.ionescu@email.com', '0722000001');

INSERT INTO Clienti (ID\_Client, Nume, Prenume, Email, Telefon)

VALUES (3, 'Marinescu', 'Mihai', 'mihai.marinescu@email.com', '0722000002');

INSERT INTO Clienti (ID\_Client, Nume, Prenume, Email, Telefon)

VALUES (4, 'Dumitrescu', 'Elena', 'elena.dumitrescu@email.com', '0722000003');

INSERT INTO Clienti (ID\_Client, Nume, Prenume, Email, Telefon)

VALUES (5, 'Stanescu', 'Radu', 'radu.stanescu@email.com', '0722000004');

INSERT INTO Clienti (ID\_Client, Nume, Prenume, Email, Telefon)

VALUES (6, 'Popa', 'Maria', 'maria.popa@email.com', '0722000005');

INSERT INTO Clienti (ID\_Client, Nume, Prenume, Email, Telefon)

VALUES (7, 'Vasilescu', 'Andrei', 'andrei.vasilescu@email.com', '0722000006');

INSERT INTO Clienti (ID\_Client, Nume, Prenume, Email, Telefon)

VALUES (8, 'Mihai', 'Ioana', 'ioana.mihai@email.com', '0722000007');

INSERT INTO Clienti (ID\_Client, Nume, Prenume, Email, Telefon)

VALUES (9, 'Georgescu', 'Cristian', 'cristian.georgescu@email.com', '0722000008');

INSERT INTO Clienti (ID\_Client, Nume, Prenume, Email, Telefon)

VALUES (10, 'Stefanescu', 'Laura', 'laura.stefanescu@email.com', '0722000009');

INSERT INTO Antrenori (ID\_Antrenor, Nume, Prenume, Specializare, Salariul)

VALUES (1, 'Ionescu', 'Ana', 'Fitness', 5000.00);

INSERT INTO Antrenori (ID\_Antrenor, Nume, Prenume, Specializare, Salariul)

VALUES (2, 'Popescu', 'Ion', 'Aerobic', 4500.00);

INSERT INTO Antrenori (ID\_Antrenor, Nume, Prenume, Specializare, Salariul)

VALUES (3, 'Marinescu', 'Mihai', 'Yoga', 5500.00);

INSERT INTO Antrenori (ID\_Antrenor, Nume, Prenume, Specializare, Salariul)

VALUES (4, 'Dumitrescu', 'Elena', 'Pilates', 6000.00);

INSERT INTO Antrenori (ID\_Antrenor, Nume, Prenume, Specializare, Salariul)

VALUES (5, 'Stanescu', 'Radu', 'Fitness', 4800.00);

INSERT INTO Antrenori (ID\_Antrenor, Nume, Prenume, Specializare, Salariul)

VALUES (6, 'Popa', 'Maria', 'Aerobic', 4700.00);

INSERT INTO Antrenori (ID\_Antrenor, Nume, Prenume, Specializare, Salariul)

VALUES (7, 'Vasilescu', 'Andrei', 'Yoga', 5300.00);

INSERT INTO Antrenori (ID\_Antrenor, Nume, Prenume, Specializare, Salariul)

VALUES (8, 'Mihai', 'Ioana', 'Pilates', 5200.00);

INSERT INTO Antrenori (ID\_Antrenor, Nume, Prenume, Specializare, Salariul)

VALUES (9, 'Georgescu', 'Cristian', 'Fitness', 5100.00);

INSERT INTO Antrenori (ID\_Antrenor, Nume, Prenume, Specializare, Salariul)

VALUES (10, 'Stefanescu', 'Laura', 'Aerobic', 4900.00);

INSERT INTO Programari (ID\_Programare, ID\_Client, ID\_Antrenor, DataProgramare)

VALUES (1, 1, 1, TO\_DATE('2024-01-10', 'YYYY-MM-DD'));

INSERT INTO Programari (ID\_Programare, ID\_Client, ID\_Antrenor, DataProgramare)

VALUES (2, 2, 2, TO\_DATE('2024-01-11', 'YYYY-MM-DD'));

INSERT INTO Programari (ID\_Programare, ID\_Client, ID\_Antrenor, DataProgramare)

VALUES (3, 3, 3, TO\_DATE('2024-01-12', 'YYYY-MM-DD'));

INSERT INTO Programari (ID\_Programare, ID\_Client, ID\_Antrenor, DataProgramare)

VALUES (4, 4, 4, TO\_DATE('2024-01-13', 'YYYY-MM-DD'));

INSERT INTO Programari (ID\_Programare, ID\_Client, ID\_Antrenor, DataProgramare)

VALUES (5, 5, 5, TO\_DATE('2024-01-14', 'YYYY-MM-DD'));

INSERT INTO Programari (ID\_Programare, ID\_Client, ID\_Antrenor, DataProgramare)

VALUES (6, 6, 6, TO\_DATE('2024-01-15', 'YYYY-MM-DD'));

INSERT INTO Programari (ID\_Programare, ID\_Client, ID\_Antrenor, DataProgramare)

VALUES (7, 7, 7, TO\_DATE('2024-01-16', 'YYYY-MM-DD'));

INSERT INTO Programari (ID\_Programare, ID\_Client, ID\_Antrenor, DataProgramare)

VALUES (8, 8, 8, TO\_DATE('2024-01-17', 'YYYY-MM-DD'));

INSERT INTO Programari (ID\_Programare, ID\_Client, ID\_Antrenor, DataProgramare)

VALUES (9, 9, 9, TO\_DATE('2024-01-18', 'YYYY-MM-DD'));

INSERT INTO Programari (ID\_Programare, ID\_Client, ID\_Antrenor, DataProgramare)

VALUES (10, 10, 10, TO\_DATE('2024-01-19', 'YYYY-MM-DD'));

INSERT INTO Abonamente (ID\_Abonament, ID\_Client, Tip, DataInceput, DataSfarsit)

VALUES (1, 1, 'Lunar', TO\_DATE('2024-01-01', 'YYYY-MM-DD'), TO\_DATE('2024-02-01', 'YYYY-MM-DD'));

INSERT INTO Abonamente (ID\_Abonament, ID\_Client, Tip, DataInceput, DataSfarsit)

VALUES (2, 2, 'Trimestrial', TO\_DATE('2024-01-01', 'YYYY-MM-DD'), TO\_DATE('2024-04-01', 'YYYY-MM-DD'));

INSERT INTO Abonamente (ID\_Abonament, ID\_Client, Tip, DataInceput, DataSfarsit)

VALUES (3, 3, 'Semestrial', TO\_DATE('2024-01-01', 'YYYY-MM-DD'), TO\_DATE('2024-07-01', 'YYYY-MM-DD'));

INSERT INTO Abonamente (ID\_Abonament, ID\_Client, Tip, DataInceput, DataSfarsit)

VALUES (4, 4, 'Anual', TO\_DATE('2024-01-01', 'YYYY-MM-DD'), TO\_DATE('2025-01-01', 'YYYY-MM-DD'));

INSERT INTO Abonamente (ID\_Abonament, ID\_Client, Tip, DataInceput, DataSfarsit)

VALUES (5, 5, 'Lunar', TO\_DATE('2024-02-01', 'YYYY-MM-DD'), TO\_DATE('2024-03-01', 'YYYY-MM-DD'));

INSERT INTO Abonamente (ID\_Abonament, ID\_Client, Tip, DataInceput, DataSfarsit)

VALUES (6, 6, 'Trimestrial', TO\_DATE('2024-02-01', 'YYYY-MM-DD'), TO\_DATE('2024-05-01', 'YYYY-MM-DD'));

INSERT INTO Abonamente (ID\_Abonament, ID\_Client, Tip, DataInceput, DataSfarsit)

VALUES (7, 7, 'Semestrial', TO\_DATE('2024-02-01', 'YYYY-MM-DD'), TO\_DATE('2024-08-01', 'YYYY-MM-DD'));

INSERT INTO Abonamente (ID\_Abonament, ID\_Client, Tip, DataInceput, DataSfarsit)

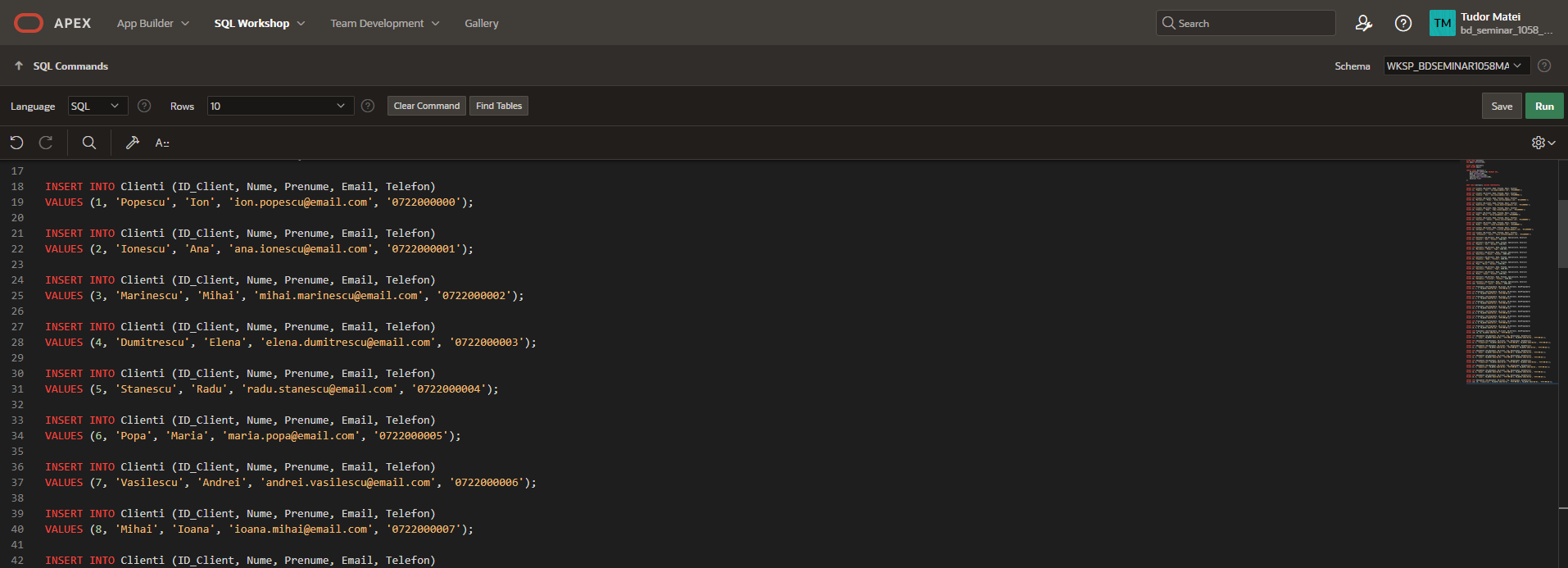
VALUES (8, 8, 'Anual', TO\_DATE('2024-02-01', 'YYYY-MM-DD'), TO\_DATE('2025-02-01', 'YYYY-MM-DD'));

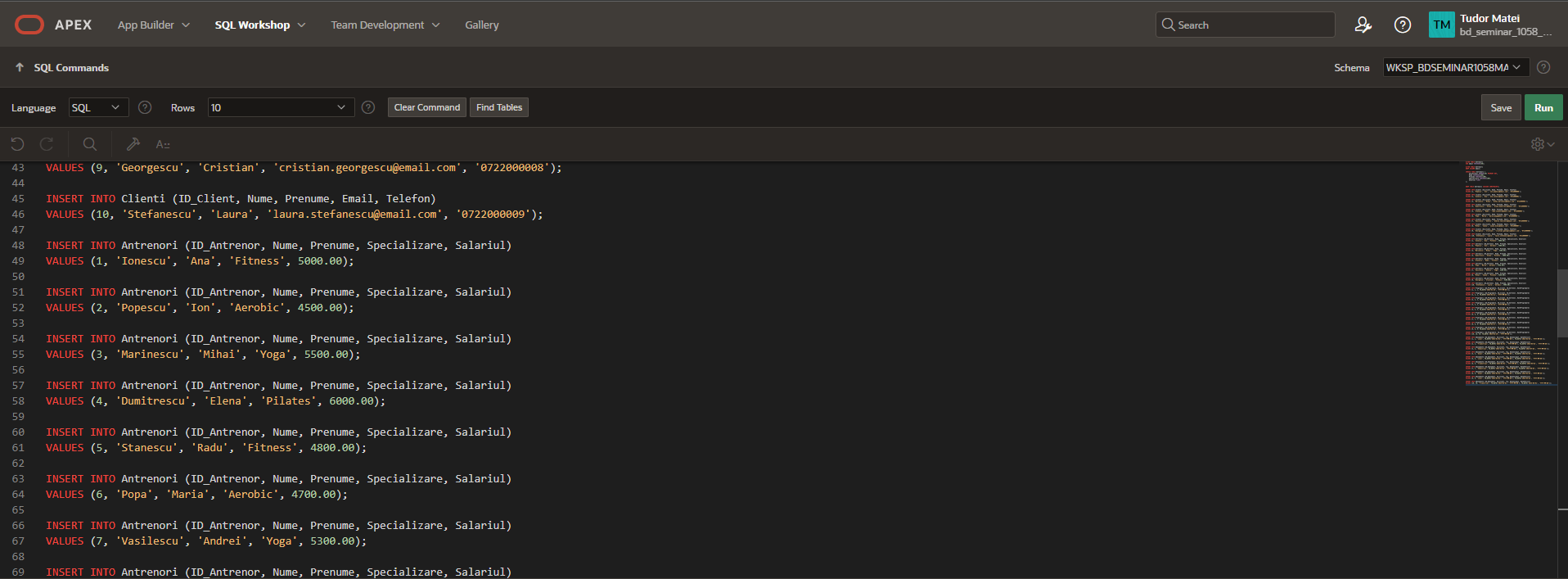
INSERT INTO Abonamente (ID\_Abonament, ID\_Client, Tip, DataInceput, DataSfarsit)

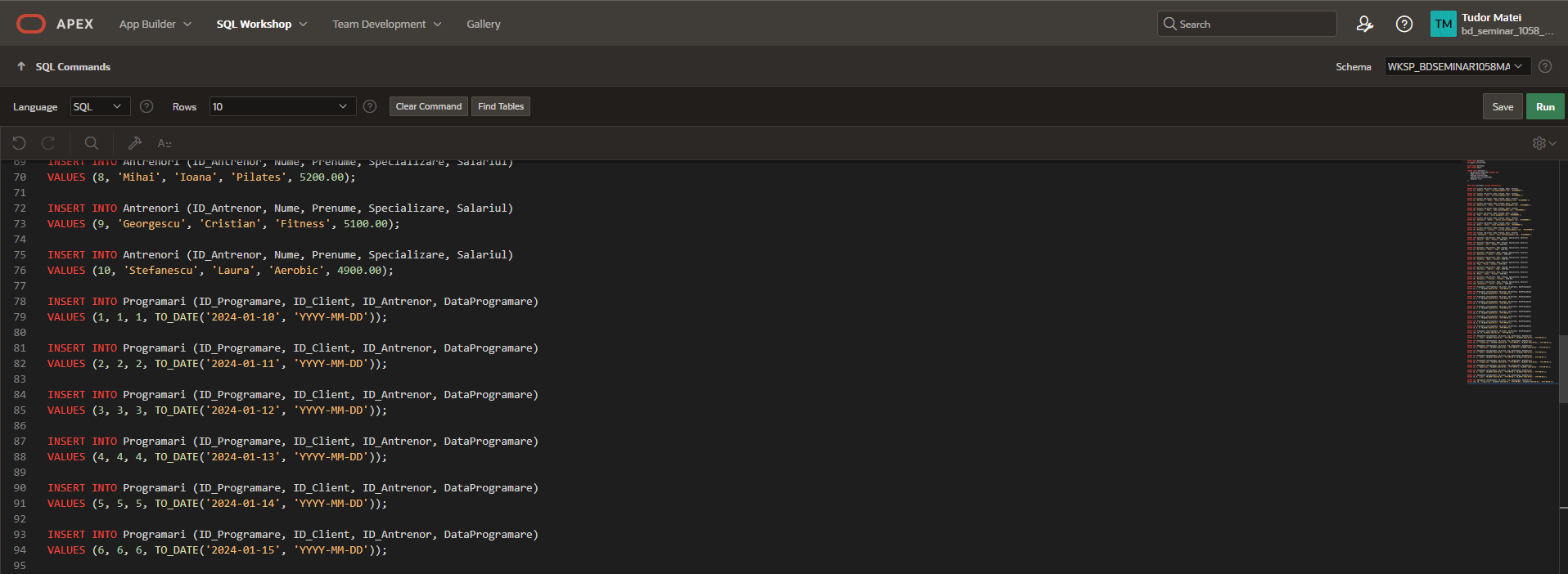
VALUES (9, 9, 'Lunar', TO\_DATE('2024-03-01', 'YYYY-MM-DD'), TO\_DATE('2024-04-01', 'YYYY-MM-DD'));

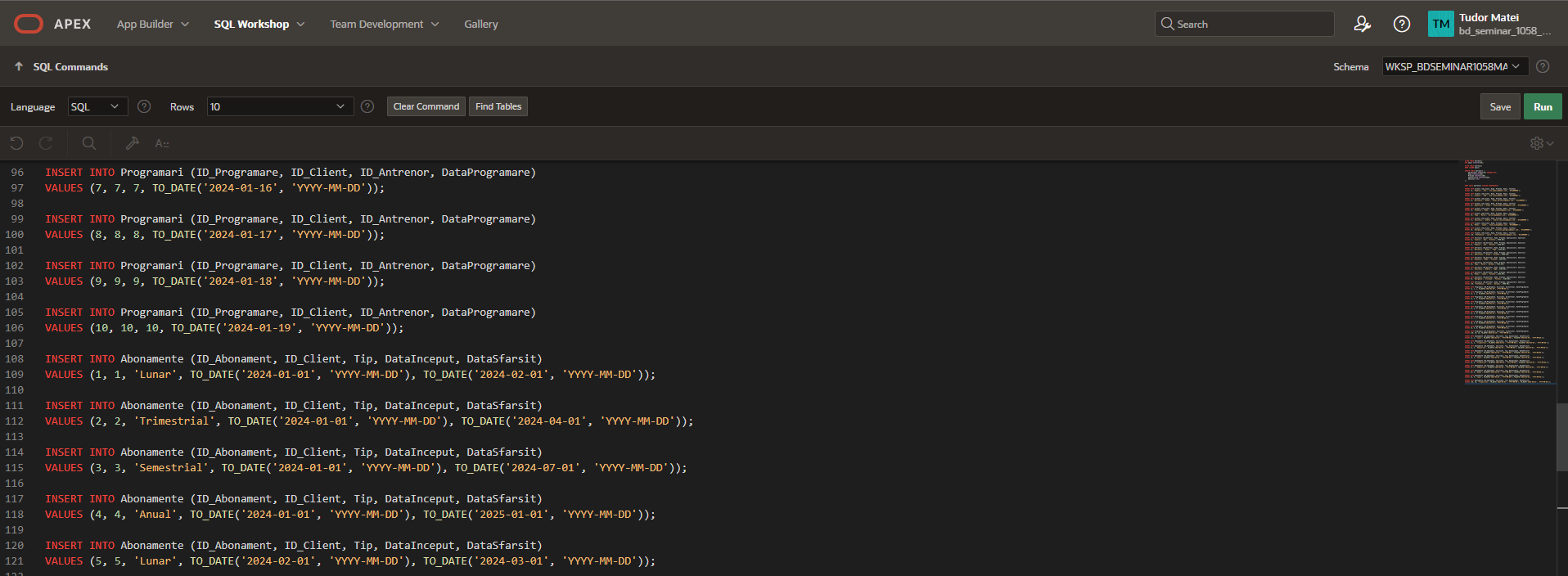
INSERT INTO Abonamente (ID\_Abonament, ID\_Client, Tip, DataInceput, DataSfarsit)

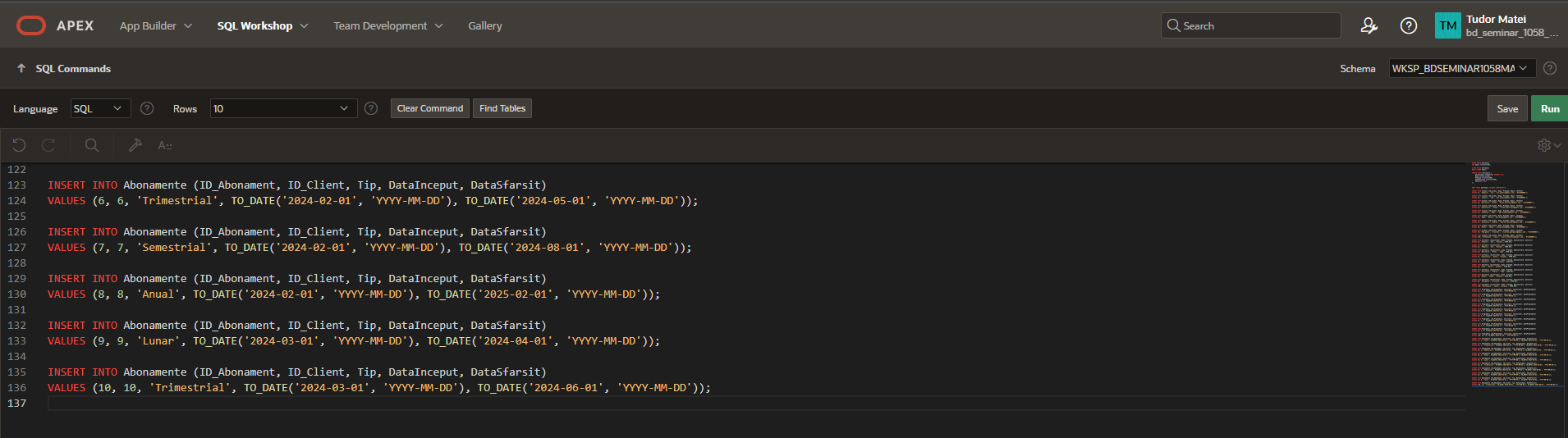
VALUES (10, 10, 'Trimestrial', TO\_DATE('2024-03-01', 'YYYY-MM-DD'), TO\_DATE('2024-06-01', 'YYYY-MM-DD'));









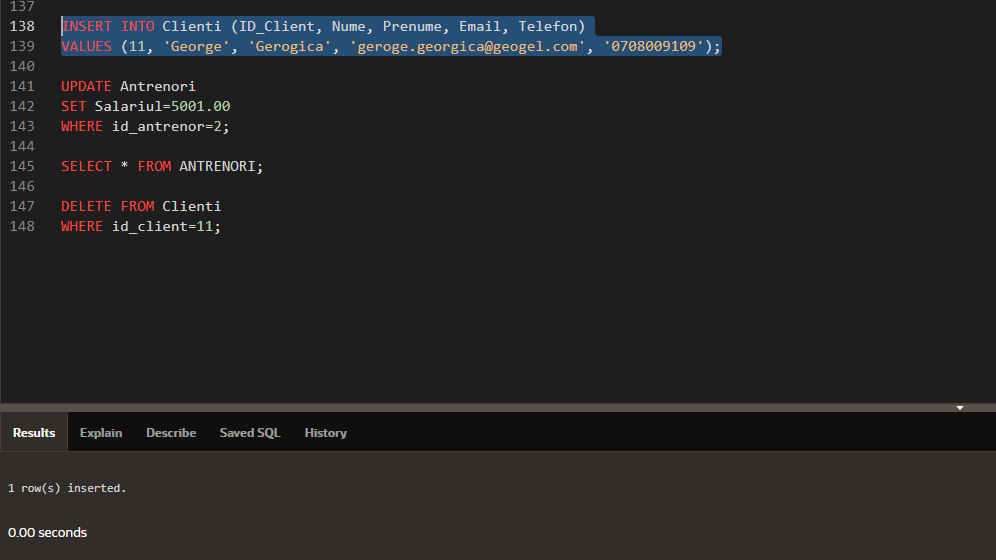


**4.**

**a) Inserarea de date noi in tabela ‘Clienti’:**

INSERT INTO Clienti (ID\_Client, Nume, Prenume, Email, Telefon)

VALUES (11, 'George', 'Gerogica', 'geroge.georgica@geogel.com', '0708009109');

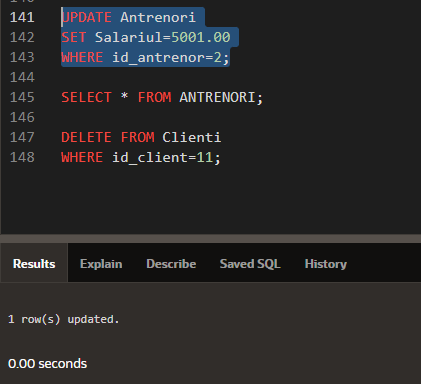


**b) Actualizare a salariului unui angajat în tabela ‘Antrenori’:**

UPDATE Antrenori

SET Salariul=5001.00

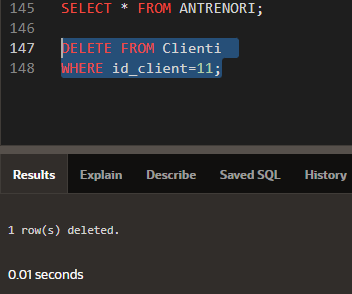
WHERE id\_antrenor=2;



**c) Ștergere a unui angajat din tabela ‘Clineti’:**

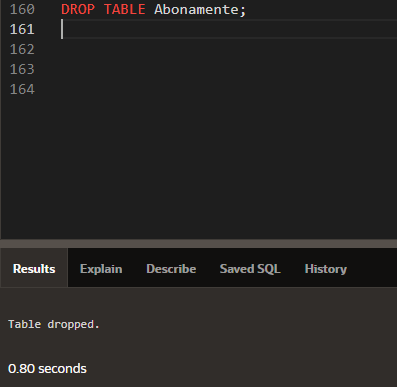
DELETE FROM Clienti

WHERE id\_client=11;



**5.Stergerea unei tabele**

DROP TABLE Abonamente;

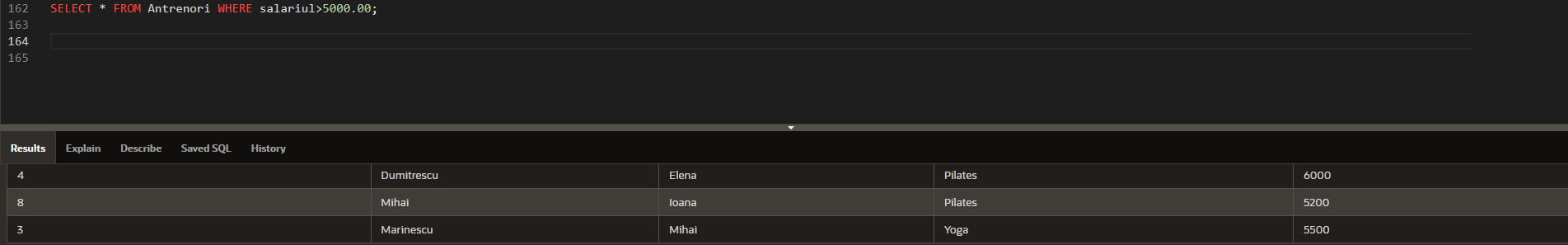


**6.**

1. **Utilizarea operatorilor de comparatie:**

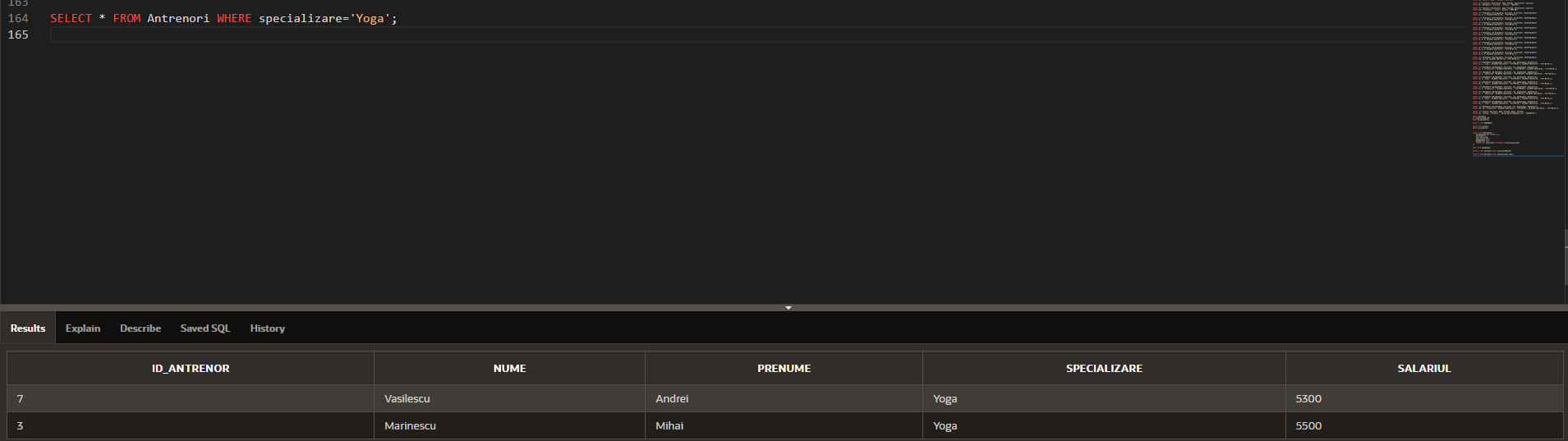
**-Sa se afiseze toți antrenorii cu un salariu mai mare de 5000:**

SELECT \* FROM Antrenori WHERE salariul>5000.00;



**-Sa se afiseze toti antrenorii cu specializarea ‚Yoga’:**

SELECT \* FROM Antrenori WHERE specializare='Yoga';



**-Sa se afiseze antrenorul cu cel mai lung nume**

SELECT Nume

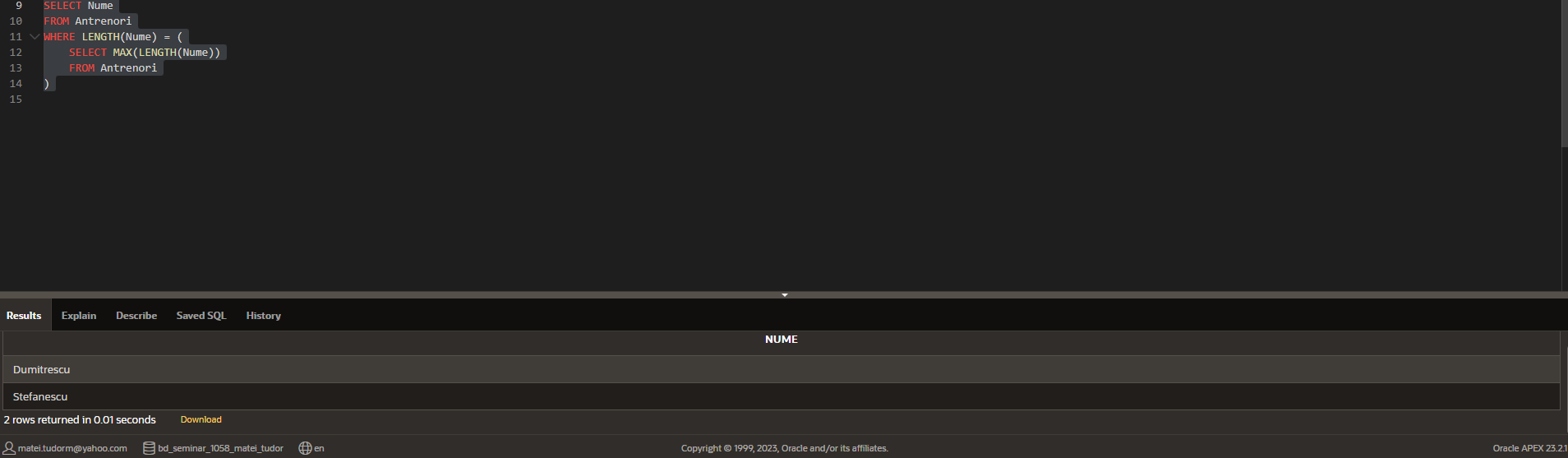
FROM Antrenori

WHERE LENGTH(Nume) = (

SELECT MAX(LENGTH(Nume))

FROM Antrenori

)



**b)Join-uri:**

**-** **Să se afiseze clientii care au avut mai mult sau egal cu o programare cu acelasi antrenor**

SELECT C.Nume, C.Prenume, A.Nume AS Nume\_Antrenor, COUNT(P.ID\_Programare) AS Numar\_Programari

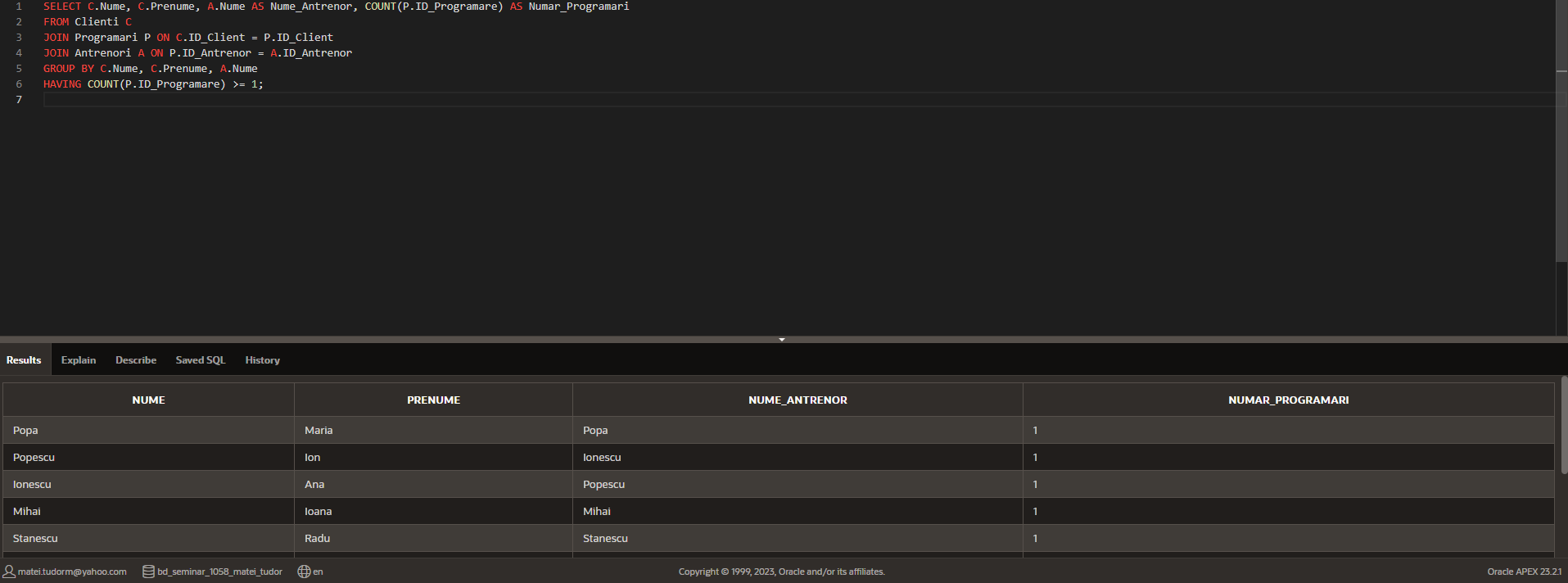
FROM Clienti C

JOIN Programari P ON C.ID\_Client = P.ID\_Client

JOIN Antrenori A ON P.ID\_Antrenor = A.ID\_Antrenor

GROUP BY C.Nume, C.Prenume, A.Nume

HAVING COUNT(P.ID\_Programare) >= 1;

****

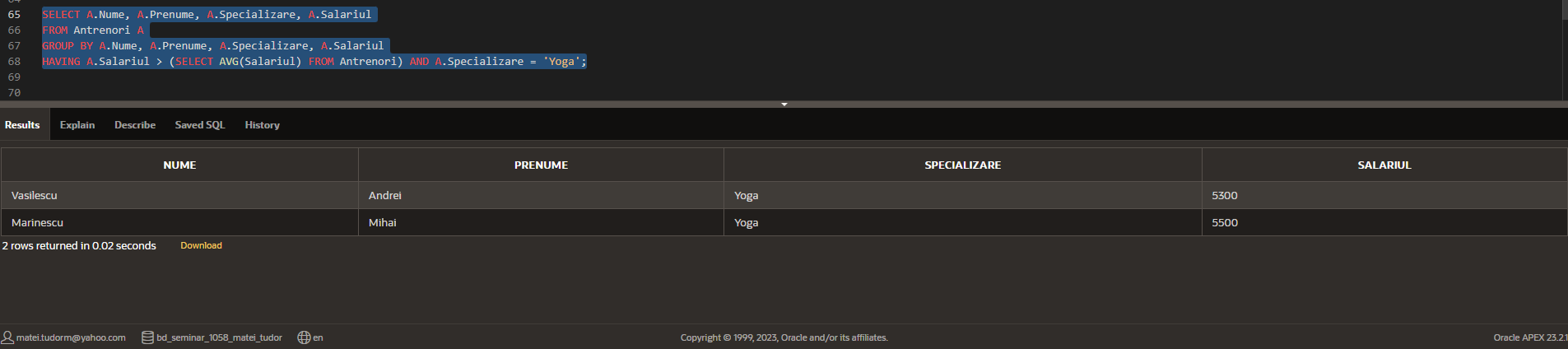
- **Sa se afiseze antrenorii care au un salariu mai mare decat media salariilor si care sunt specializati in ‘Yoga’:**

SELECT A.Nume, A.Prenume, A.Specializare, A.Salariul

FROM Antrenori A

GROUP BY A.Nume, A.Prenume, A.Specializare, A.Salariul

HAVING A.Salariul > (SELECT AVG(Salariul) FROM Antrenori) AND A.Specializare = 'Yoga';

****

**-** **Sa se afiseze toți clientii care au avut cel putin o programare cu un antrenor specializat in “Fitness”:**

SELECT C.Nume, C.Prenume

FROM Clienti C

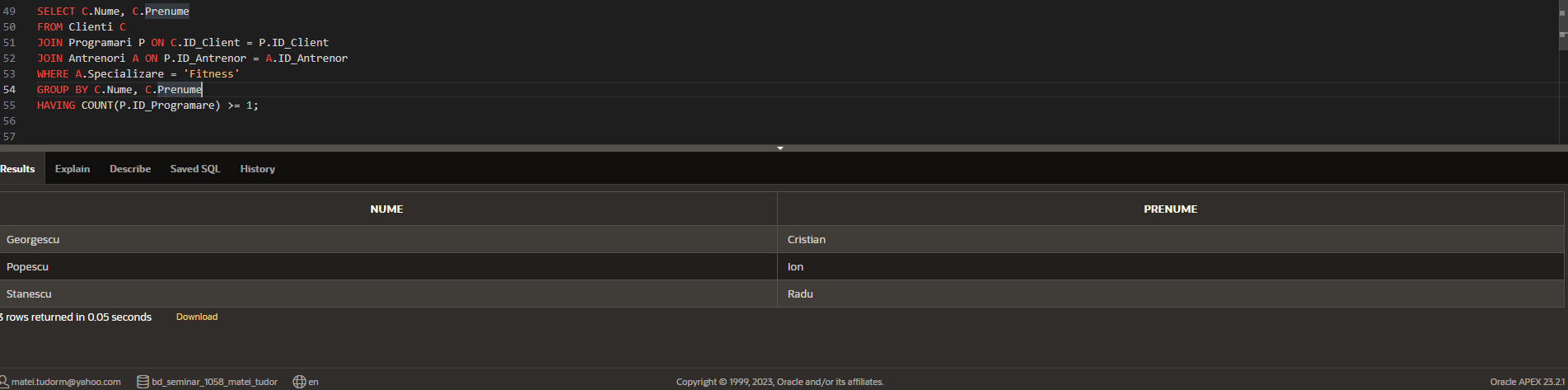
JOIN Programari P ON C.ID\_Client = P.ID\_Client

JOIN Antrenori A ON P.ID\_Antrenor = A.ID\_Antrenor

WHERE A.Specializare = 'Fitness'

GROUP BY C.Nume, C.Prenume

HAVING COUNT(P.ID\_Programare) >= 1;



**-Sa se afiseze numele si prenumele clintului care este programat sa se antreneze cu antrenorul cu salariul cel mai mare**

SELECT C.Nume, C.Prenume

FROM Clienti C

JOIN Programari P ON C.ID\_Client = P.ID\_Client

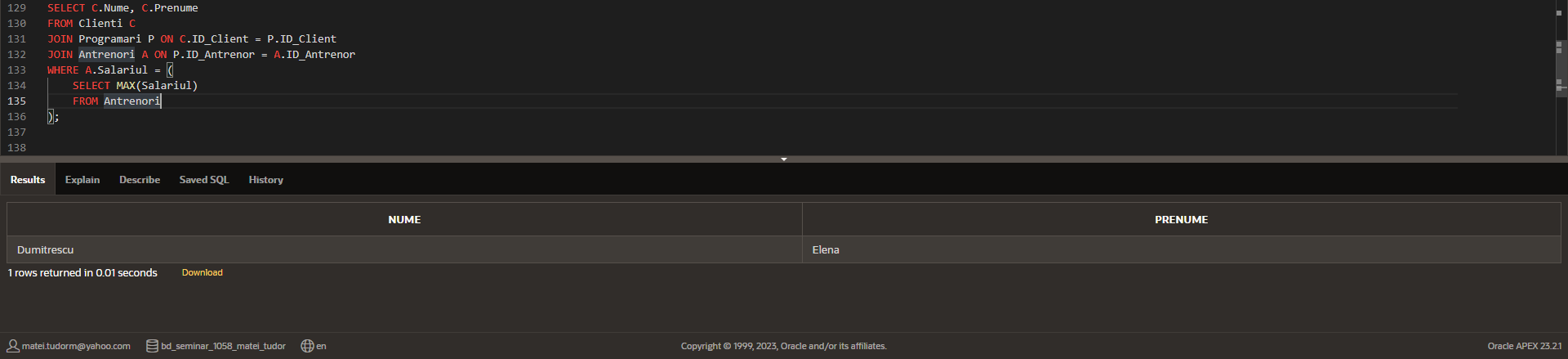
JOIN Antrenori A ON P.ID\_Antrenor = A.ID\_Antrenor

WHERE A.Salariul = (

SELECT MAX(Salariul)

FROM Antrenori

);



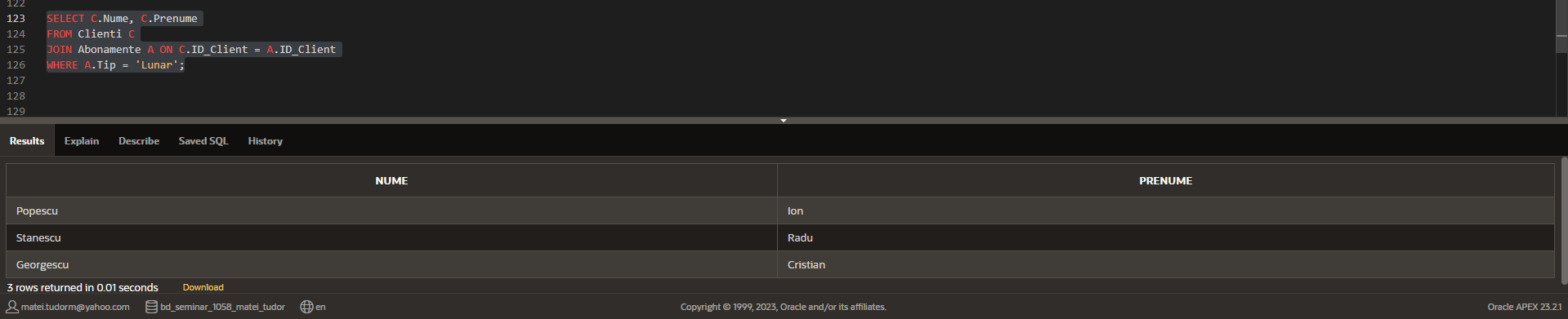
**- Sa se afiseze numele și prenumele clientilor care au un abonament lunar**

SELECT C.Nume, C.Prenume

FROM Clienti C

JOIN Abonamente A ON C.ID\_Client = A.ID\_Client

WHERE A.Tip = 'Lunar';



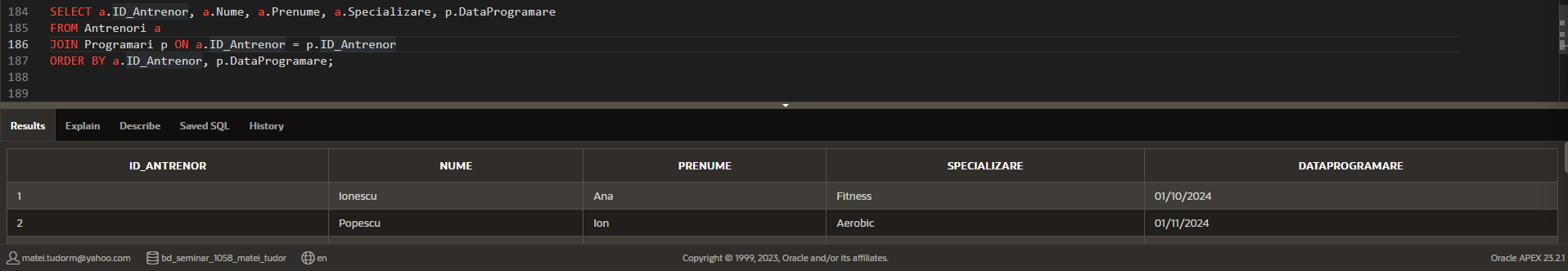
**-Sa se afiseze detalii despre antrenori si programarile lor**

SELECT a.ID\_Antrenor, a.Nume, a.Prenume, a.Specializare, p.DataProgramare

FROM Antrenori a

JOIN Programari p ON a.ID\_Antrenor = p.ID\_Antrenor

ORDER BY a.ID\_Antrenor, p.DataProgramare;



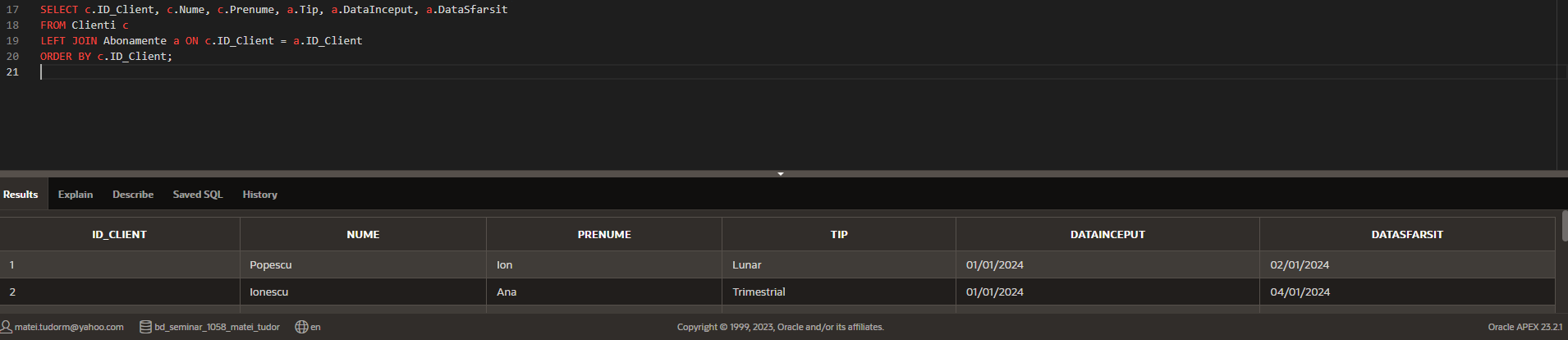
**-Sa se afiseze clientul și abonamentul pe care il are:**

SELECT c.ID\_Client, c.Nume, c.Prenume, a.Tip, a.DataInceput, a.DataSfarsit

FROM Clienti c

LEFT JOIN Abonamente a ON c.ID\_Client = a.ID\_Client

ORDER BY c.ID\_Client;



**-Sa se afiseze antrenorul care antrezeaza clientul cu cel mai lung prenume**

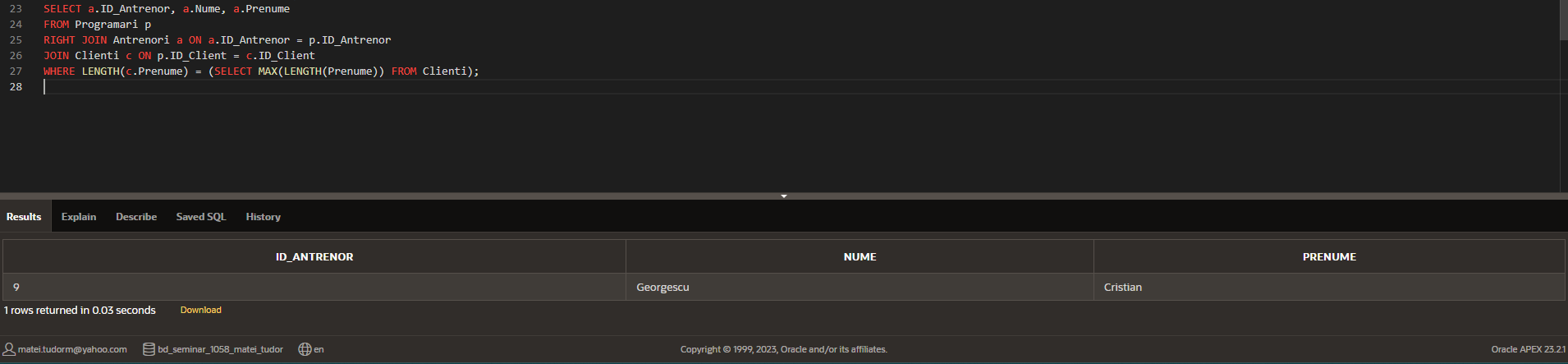
SELECT a.ID\_Antrenor, a.Nume, a.Prenume

FROM Programari p

RIGHT JOIN Antrenori a ON a.ID\_Antrenor = p.ID\_Antrenor

JOIN Clienti c ON p.ID\_Client = c.ID\_Client

WHERE LENGTH(c.Prenume) = (SELECT MAX(LENGTH(Prenume)) FROM Clienti);



**c)Utilizarea functiilor de grup si conditii asupra acestora:**

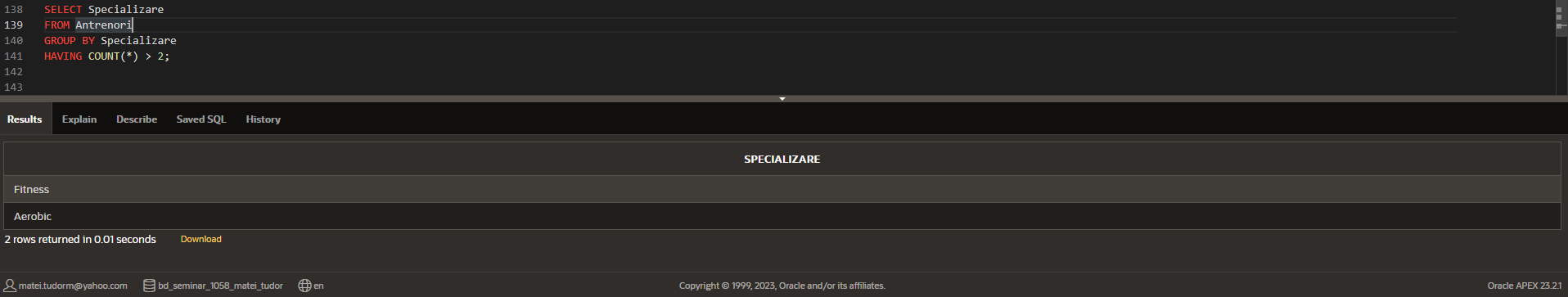
-Sa se afiseze specializarile care au mai mult de 2 antrenori:

SELECT Specializare

FROM Antrenori

GROUP BY Specializare

HAVING COUNT(\*) > 2;

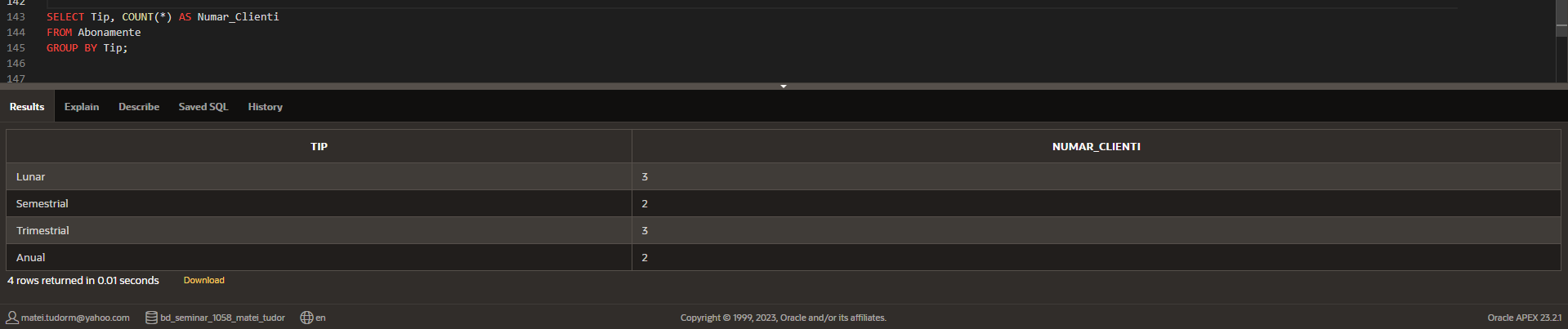


- Sa se afiseze numarul de clienti care au un abonament de fiecare tip:

SELECT Tip, COUNT(\*) AS Numar\_Clienti

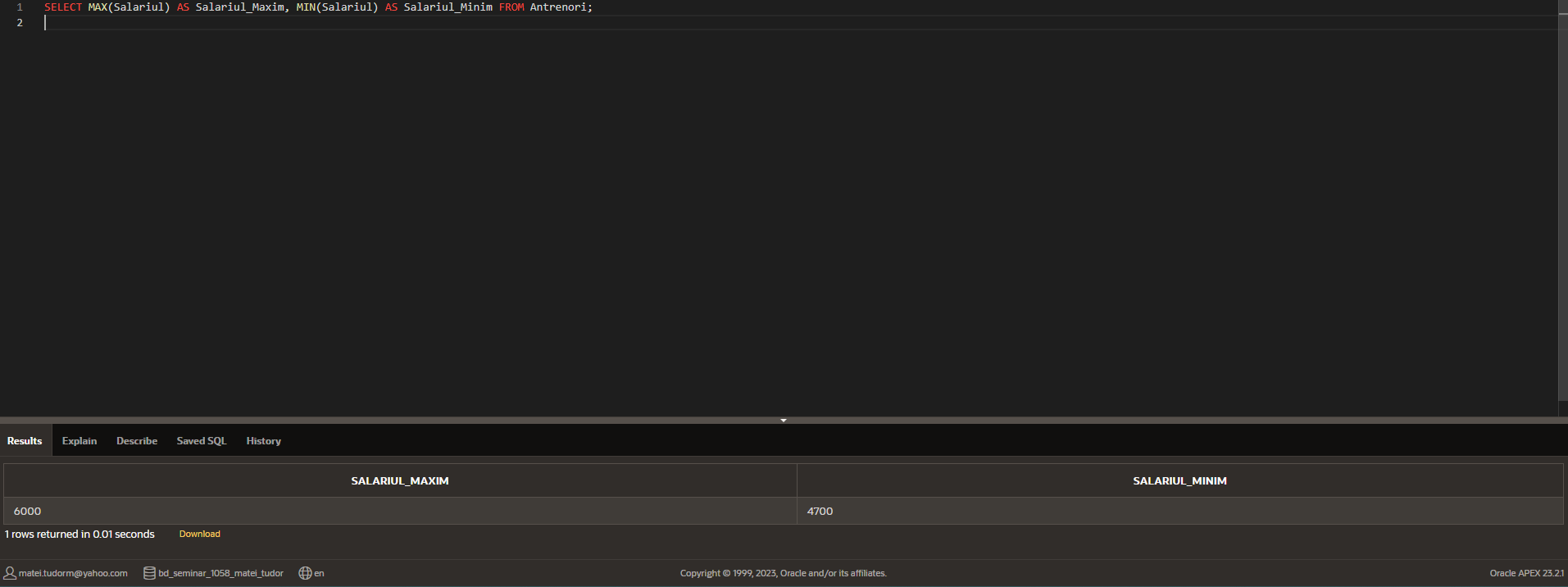
FROM Abonamente

GROUP BY Tip;



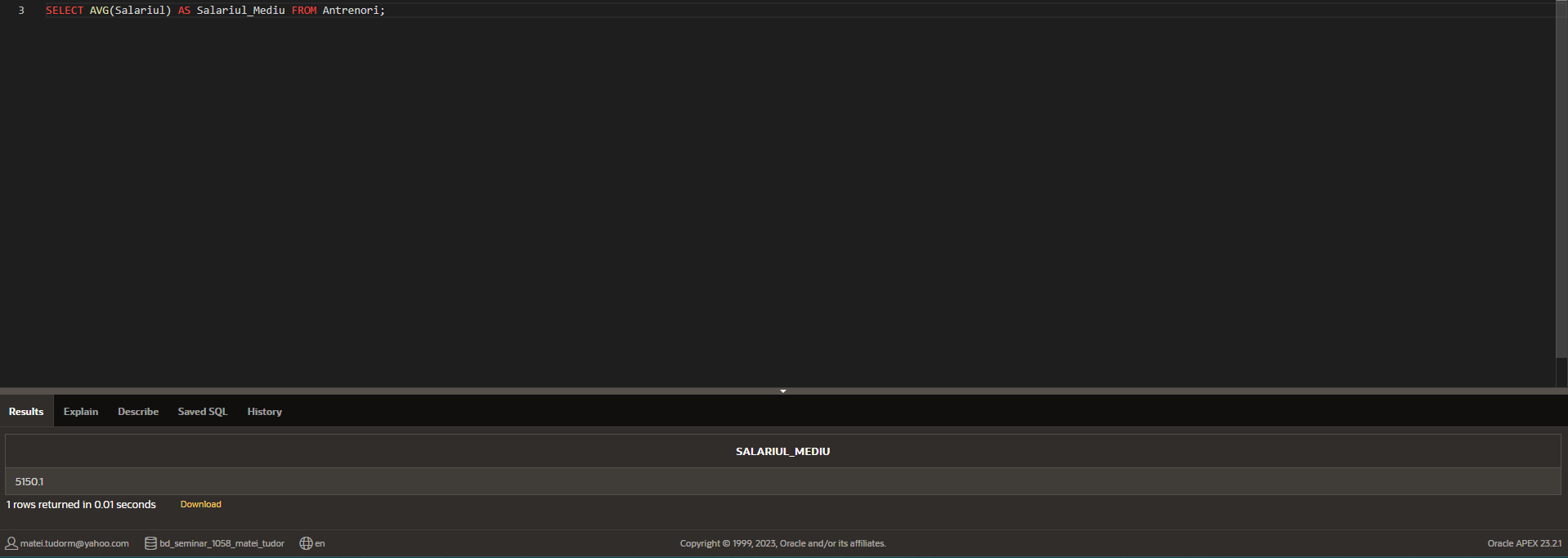
**-Sa se afiseze salariul maxim și minim din tabela Antrenori:**

SELECT MAX(Salariul) AS Salariul\_Maxim, MIN(Salariul) AS Salariul\_Minim FROM Antrenori;



**-Sa se afiseze salariul mediu al antrenorilor**

SELECT AVG(Salariul) AS Salariul\_Mediu FROM Antrenori;

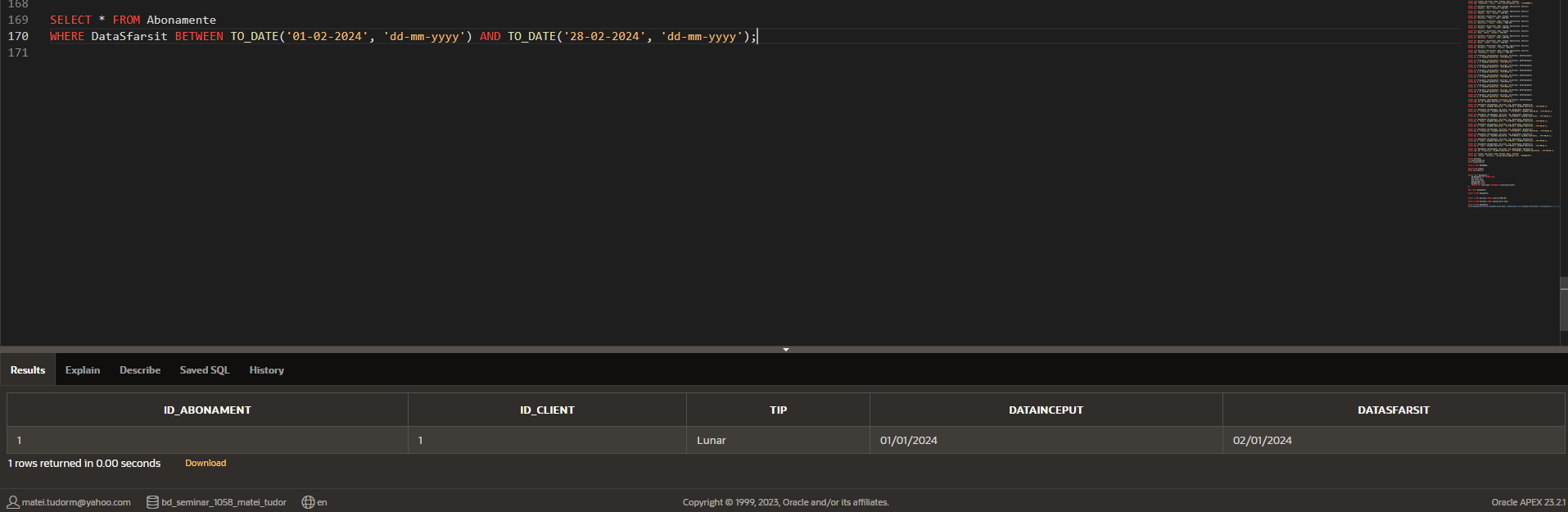


**d)Utilizarea functiilor numerice, de tip caracter, pentru data si timp:**

**-Sa se afiseze toate abonamentele care se termina in februarie 2024**

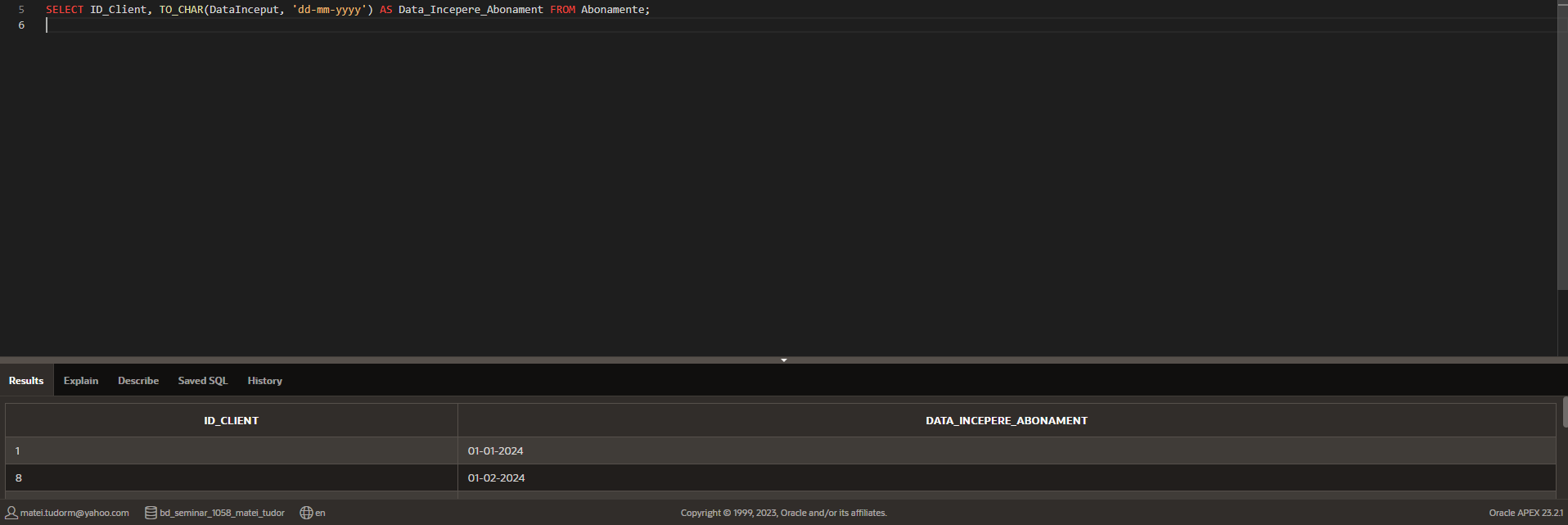
SELECT \* FROM Abonamente

WHERE DataSfarsit BETWEEN TO\_DATE('01-02-2024', 'dd-mm-yyyy') AND TO\_DATE('28-02-2024', 'dd-mm-yyyy');



**-Sa se afiseze data la care incepe abonamentul fiecarui client**

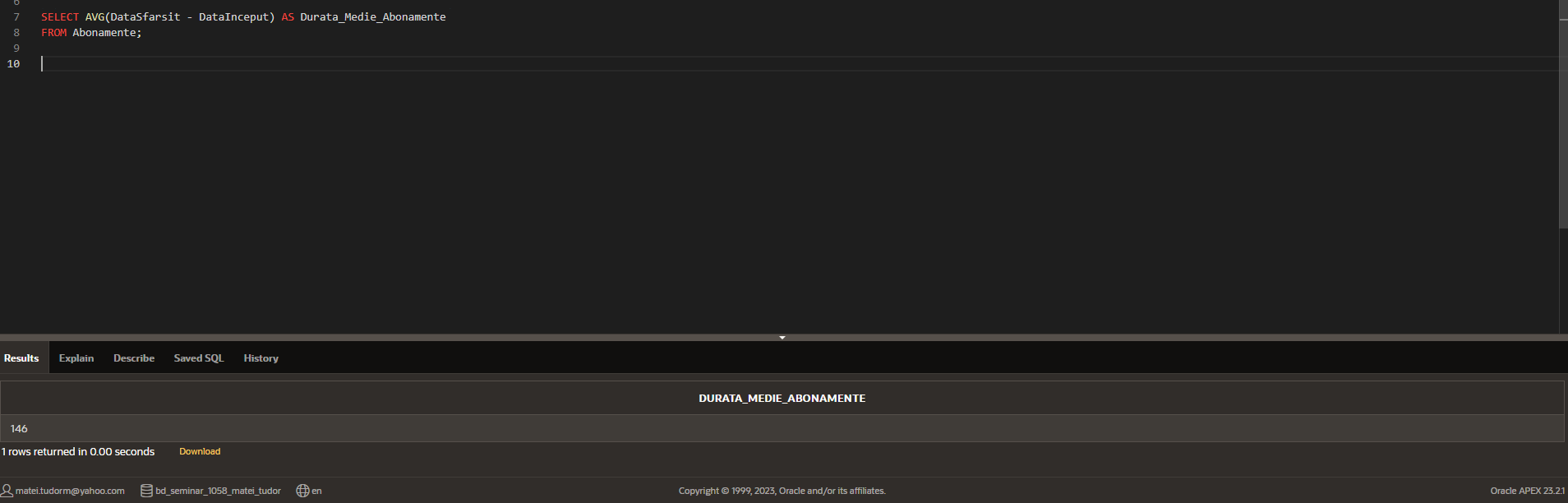
SELECT ID\_Client, TO\_CHAR(DataInceput, 'dd-mm-yyyy') AS Data\_Incepere\_Abonament FROM Abonamente;



**-Sa se afiseze durata medie a abonamentelor**

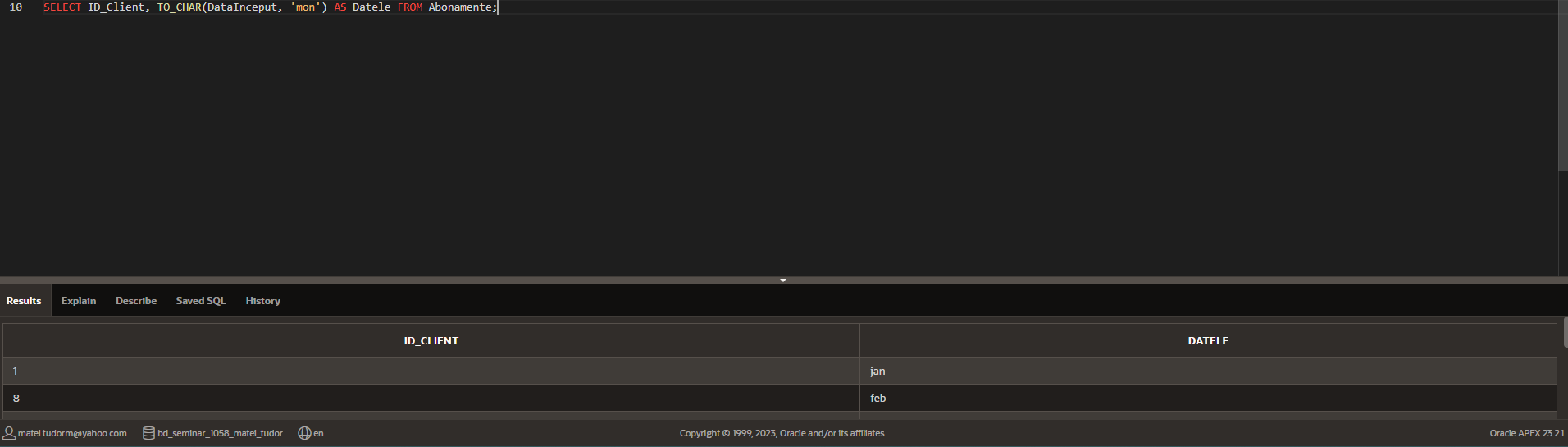
SELECT AVG(DataSfarsit - DataInceput) AS Durata\_Medie\_Abonamente

FROM Abonamente;



**-Sa se afiseze abonamentele in functie doar de luna**

SELECT ID\_Client, TO\_CHAR(DataInceput, 'mon') AS Datele FROM Abonamente;



**e)Construirea de expresii cu DECODE și CASE:**

**-Sa se clasifice antrenorii in functie de salariul lor(Incepator, Experimentat, Avansat)**

SELECT Nume, Prenume,

DECODE (

SIGN(Salariul - 5000),

-1, 'Incepator',

DECODE (

SIGN(Salariul - 5500),

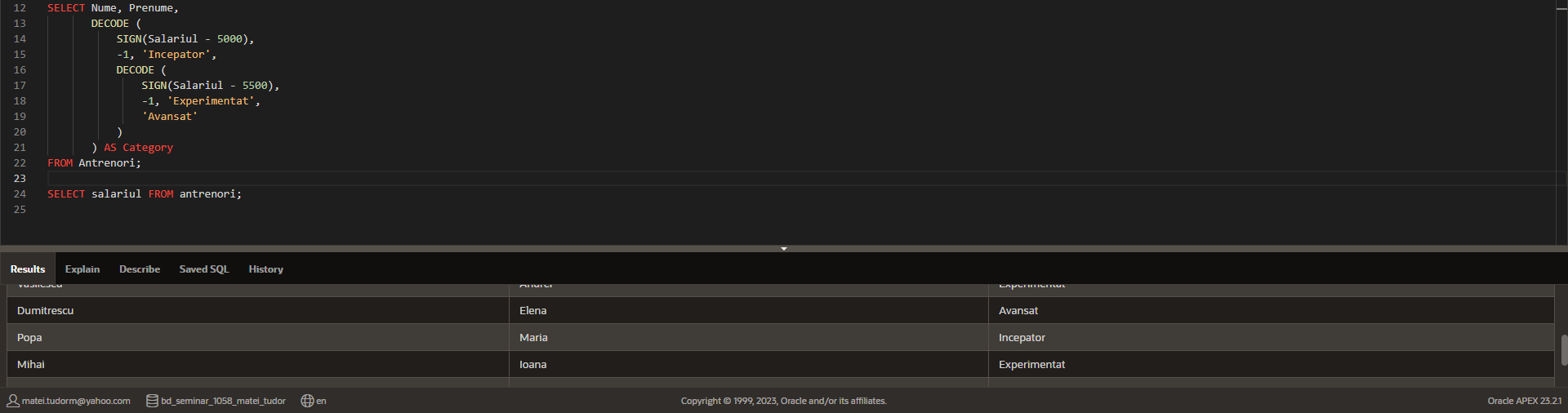
-1, 'Experimentat',

'Avansat'

)

) AS Category

FROM Antrenori;

****

**-Aceeasi problema ca mai sus, dar folosind case**

SELECT Nume, Prenume,

CASE

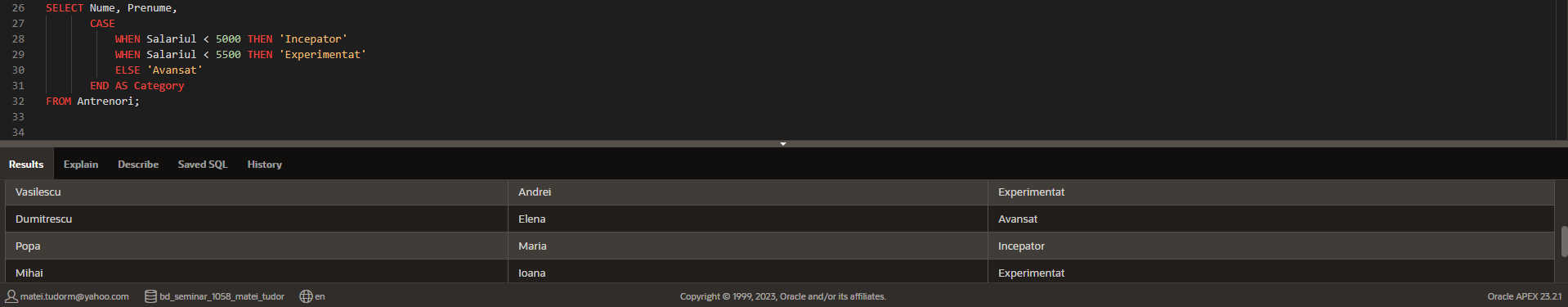
WHEN Salariul < 5000 THEN 'Incepator'

WHEN Salariul < 5500 THEN 'Experimentat'

ELSE 'Avansat'

END AS Category

FROM Antrenori;



f) Utilizarea operatorilor UNION, MINUS, INTERSECT:

**- Sa se afiseze numele si prenumele atat al clientilor cat si al antrenorilor folosind functia UNION**

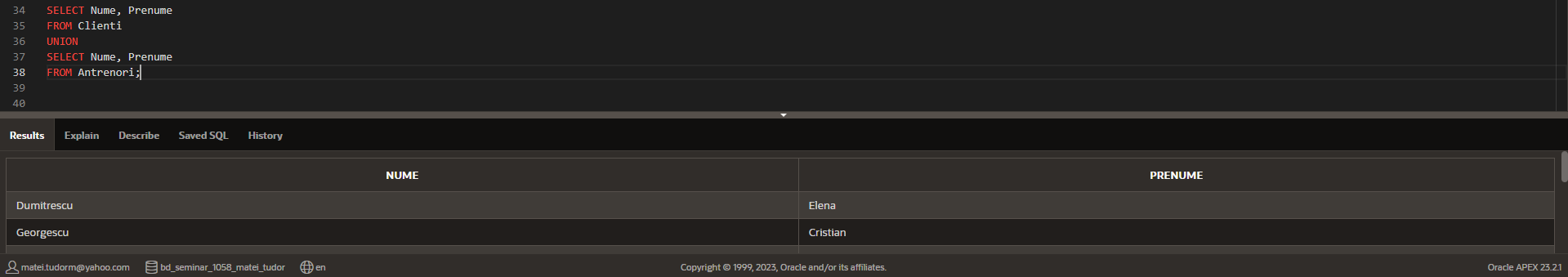
SELECT Nume, Prenume

FROM Clienti

UNION

SELECT Nume, Prenume

FROM Antrenori;



**-Afiseaza clientii care au abonament lunar folosind functia MINUS**

SELECT ID\_Client

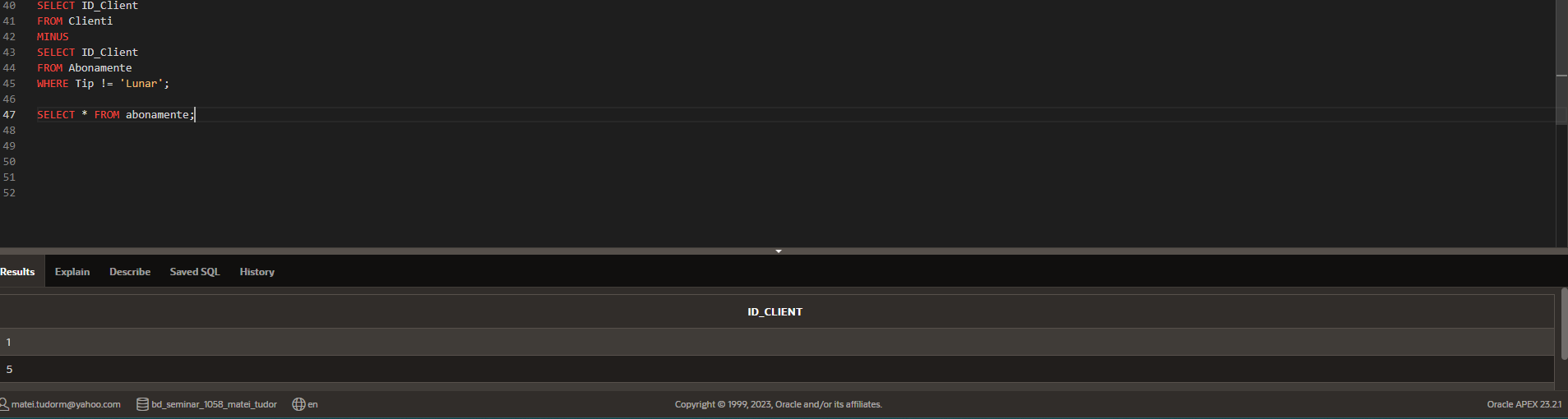
FROM Clienti

MINUS

SELECT ID\_Client

FROM Abonamente

WHERE Tip != 'Lunar';

****

**-Afiseaza clientii care au si abonamente, dar si programari**

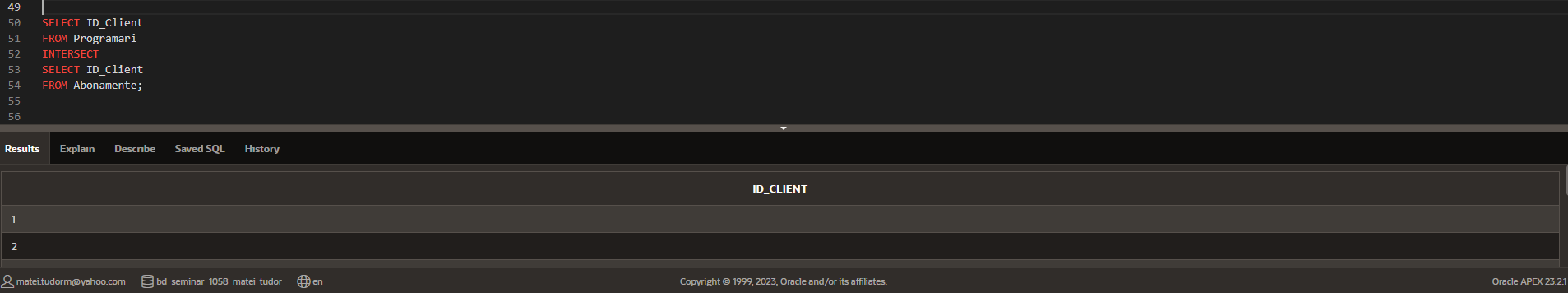
SELECT ID\_Client

FROM Programari

INTERSECT

SELECT ID\_Client

FROM Abonamente**;**

****

**g)Subcereri (cereri imbricate):**

**-** **Afiseaza clientul care este programat sa se antreneze cu antrenorul cu cel mai lung nume**

SELECT C.Nume, C.Prenume

FROM Clienti C

JOIN Programari P ON C.ID\_Client = P.ID\_Client

WHERE P.ID\_Antrenor = (

SELECT ID\_Antrenor

FROM Antrenori

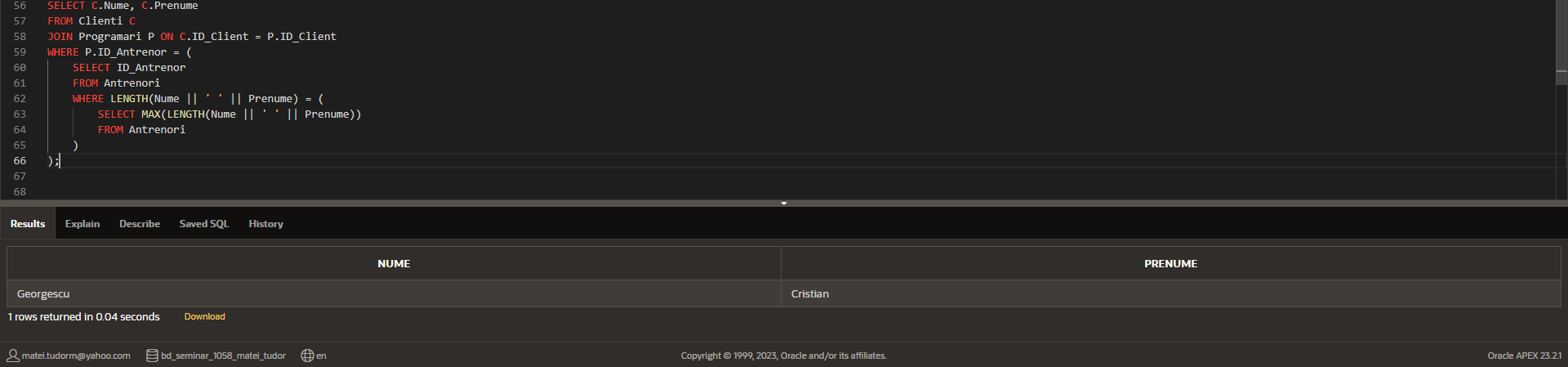
WHERE LENGTH(Nume || ' ' || Prenume) = (

SELECT MAX(LENGTH(Nume || ' ' || Prenume))

FROM Antrenori

)

);



-**Afiseaza clientii cu cel mai lung abonament**

SELECT Nume, Prenume

FROM Clienti

WHERE ID\_Client IN (

SELECT ID\_Client

FROM Abonamente

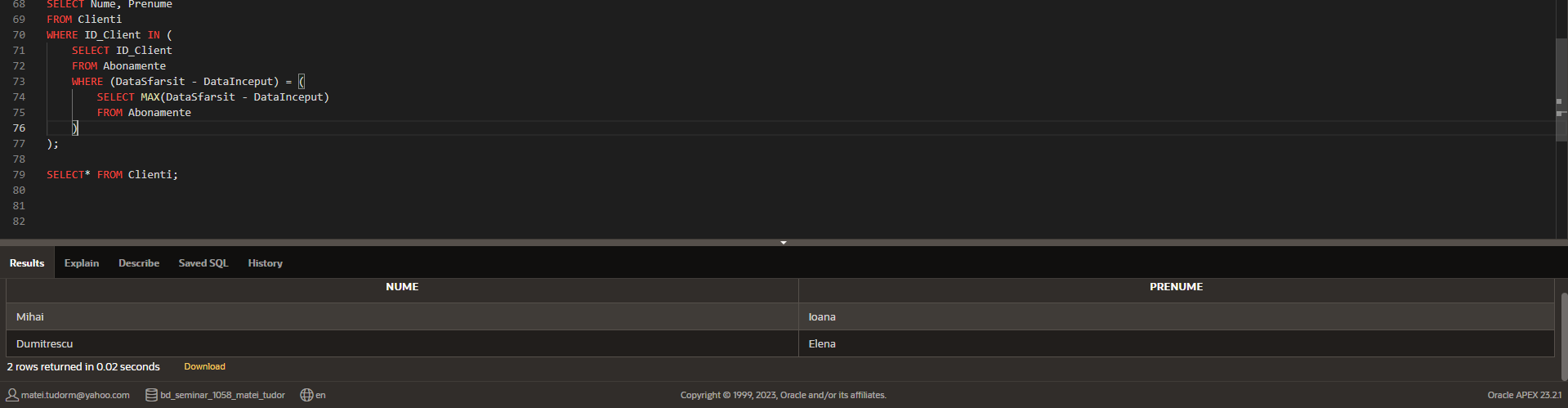
WHERE (DataSfarsit - DataInceput) = (

SELECT MAX(DataSfarsit - DataInceput)

FROM Abonamente

)

);



**-Afiseaza antrenorii cei mai bine platiti pentru fiecare specializare**

SELECT A1.Nume, A1.Prenume, A1.Specializare, A1.Salariul

FROM Antrenori A1

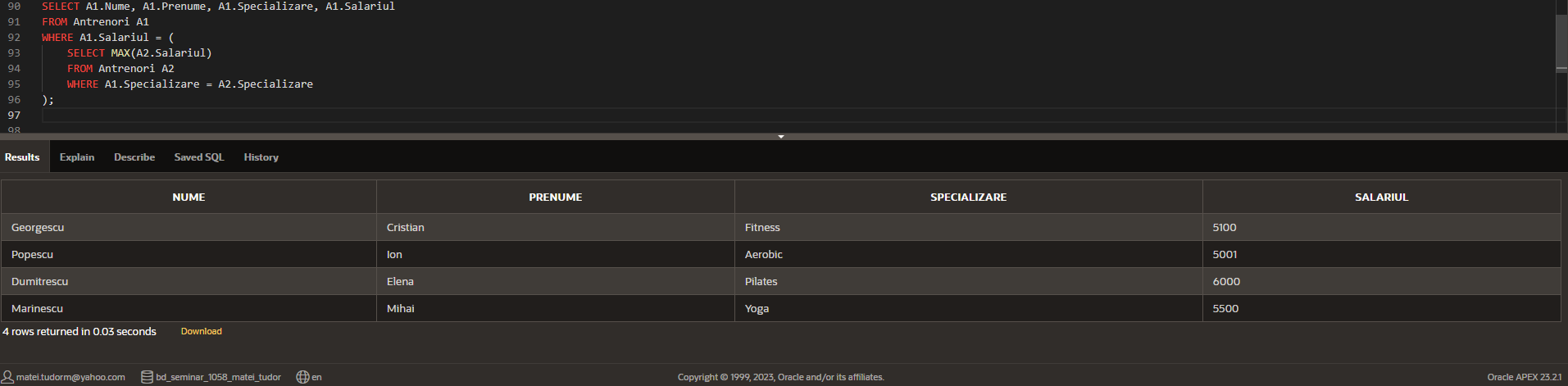
WHERE A1.Salariul = (

SELECT MAX(A2.Salariul)

FROM Antrenori A2

WHERE A1.Specializare = A2.Specializare

);



**7.** **Gestiunea altor obiecte ale bazei de date: vederi, indecsi, sinonime, secvente etc**

**-Creati o vedere care sa afiseze numele si telefonul al clientilor împreună cu numele antrenoriilor:**

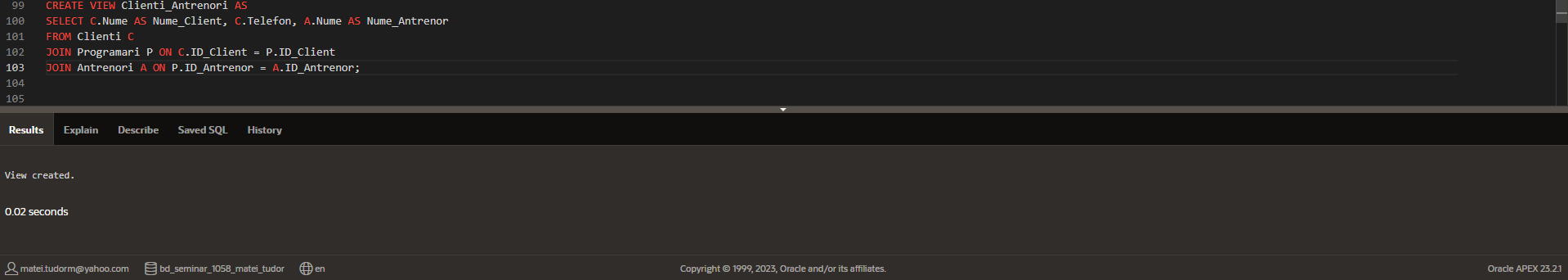
CREATE VIEW Clienti\_Antrenori AS

SELECT C.Nume AS Nume\_Client, C.Telefon, A.Nume AS Nume\_Antrenor

FROM Clienti C

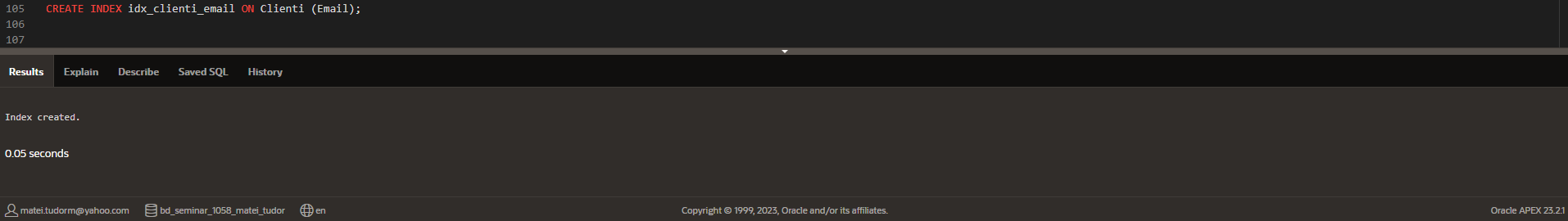
JOIN Programari P ON C.ID\_Client = P.ID\_Client

JOIN Antrenori A ON P.ID\_Antrenor = A.ID\_Antrenor;



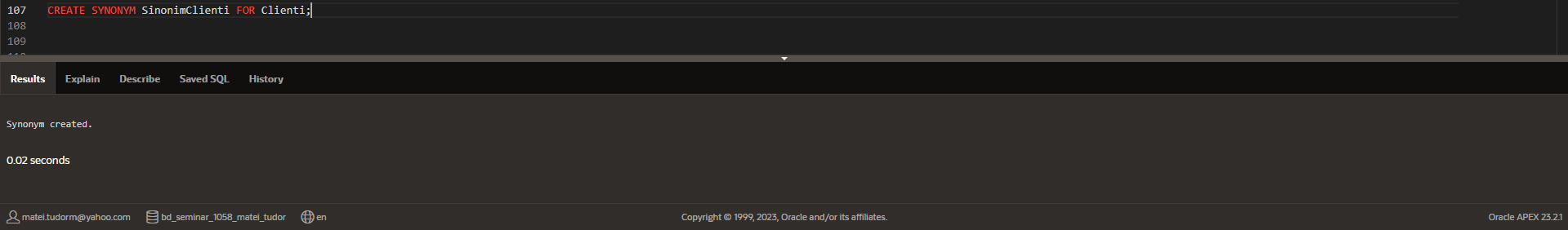
**-Creati un index pentru coloana email din tabela Clienti pentru a eficientiza operatiile de cautare:**

CREATE INDEX idx\_clienti\_email ON Clienti (Email);



**-Creati un sinonim pentru tabela Clienti:**

CREATE SYNONYM SinonimClienti FOR Clienti;



**-Creati o secventa:**

CREATE SEQUENCE seq\_clienti

START WITH 1

INCREMENT BY 1;

