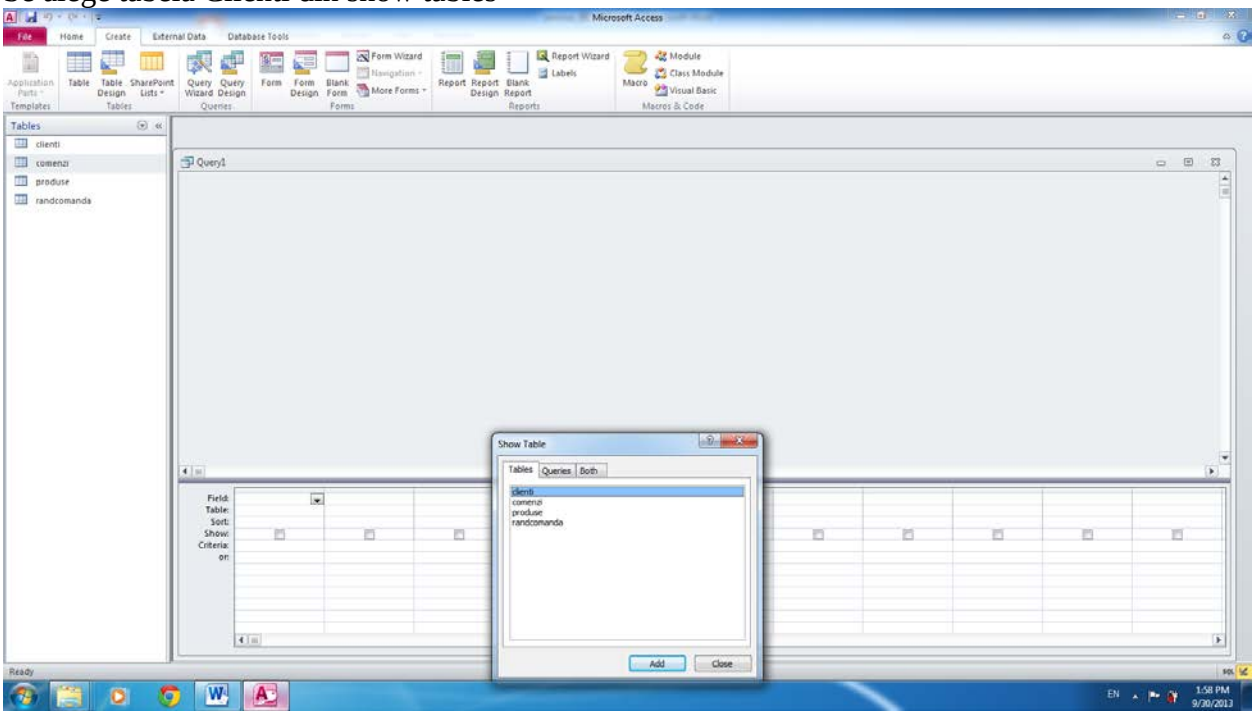
**Realizarea cererilor**

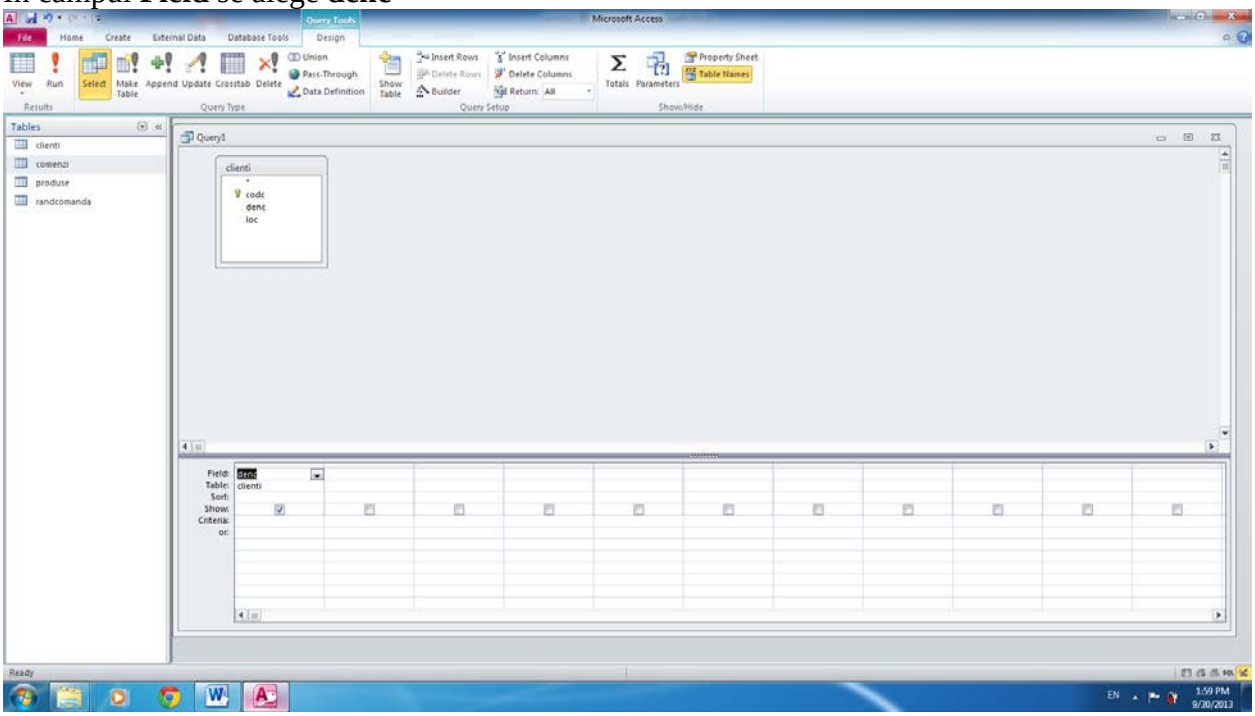
1. Sa se afiseze toti clientii (denumirea lor)

Create/query design

Se alege tabela Clienti din show tables

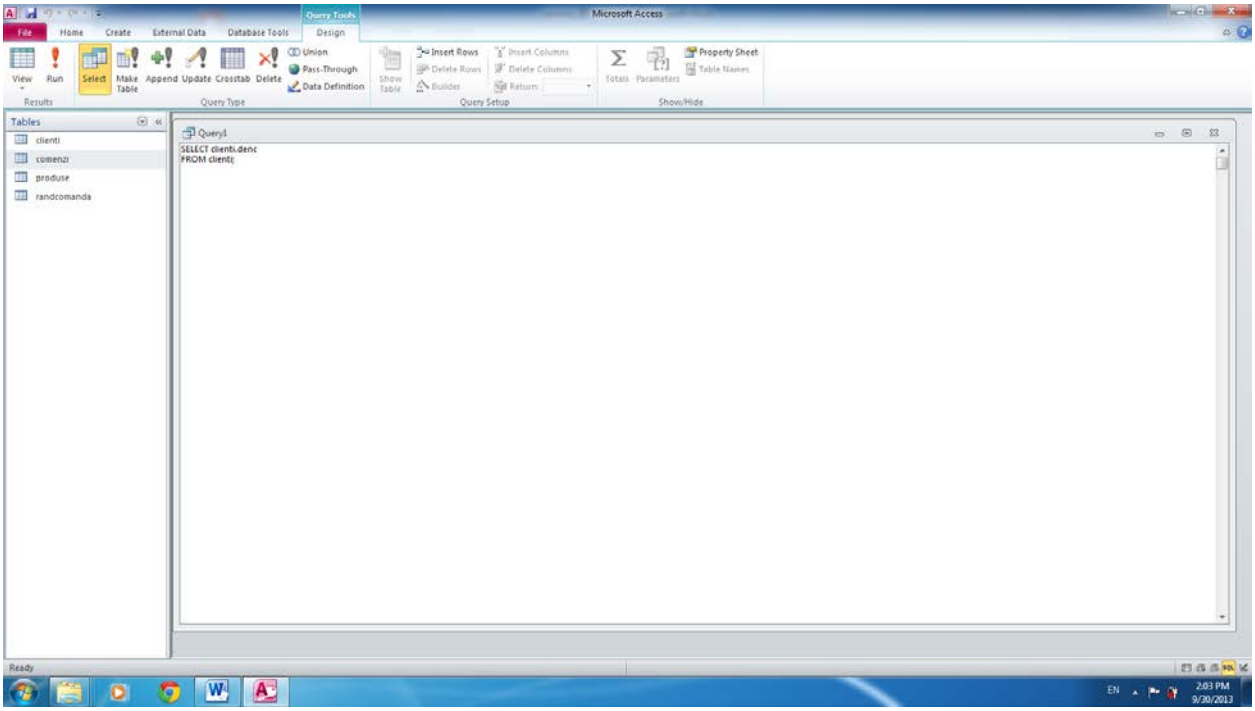



In campul **Field** se alege **denc**



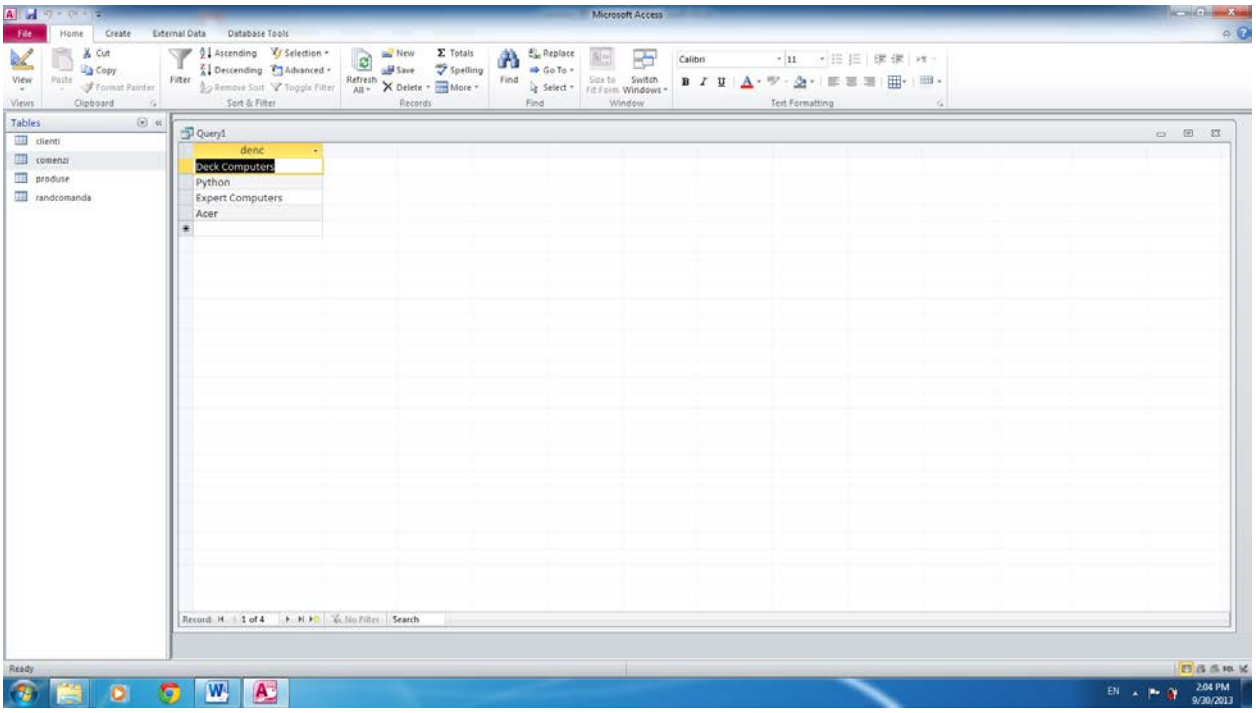
Se genereaza o comanda SELECT-SQL (optiunea View/SQL view) :
SELECT atribut
FROM tabela

SELECT denc ---------optiunea Field=denc
From clienti----------optiunea table

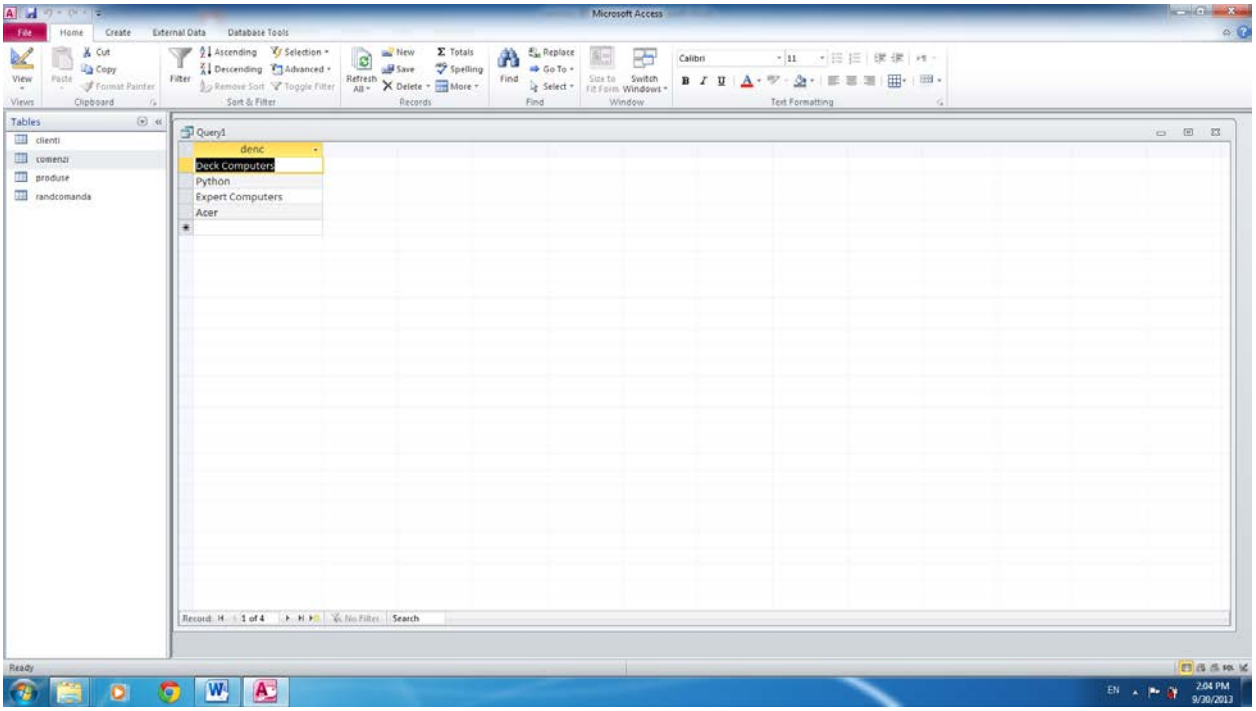



Se revine selectand **View/design view design**

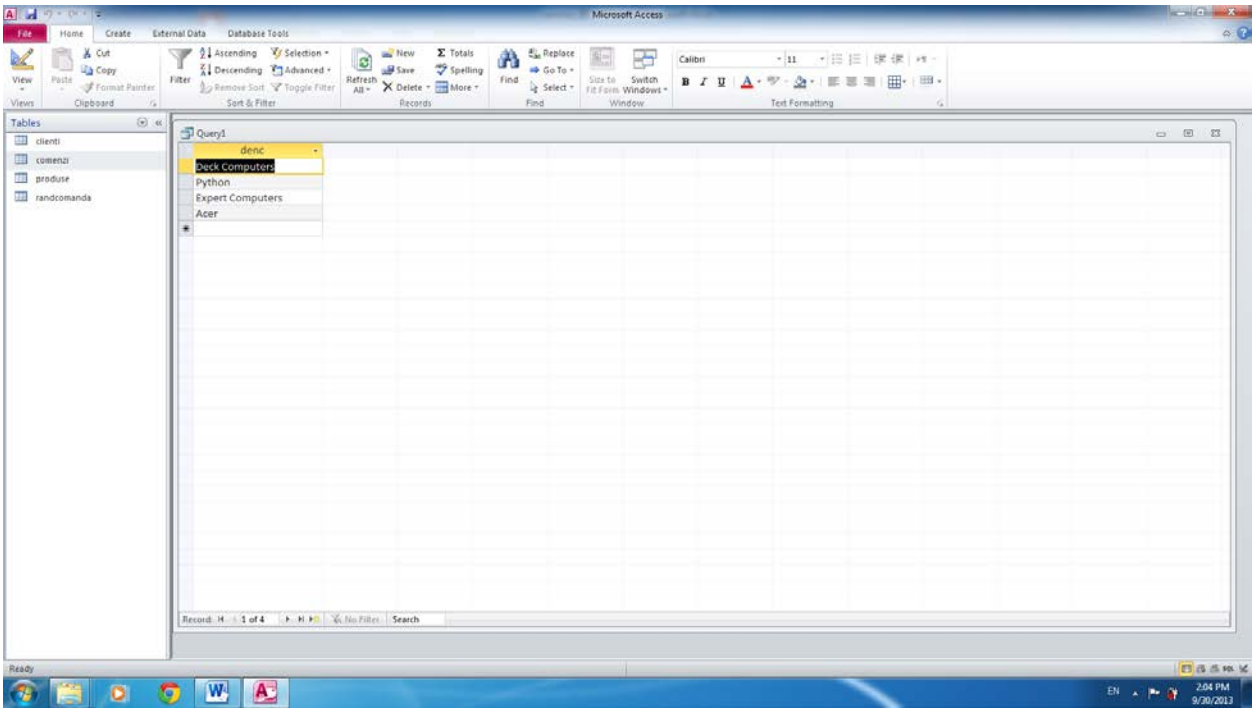
Se executa cererea: optiunea **Run**

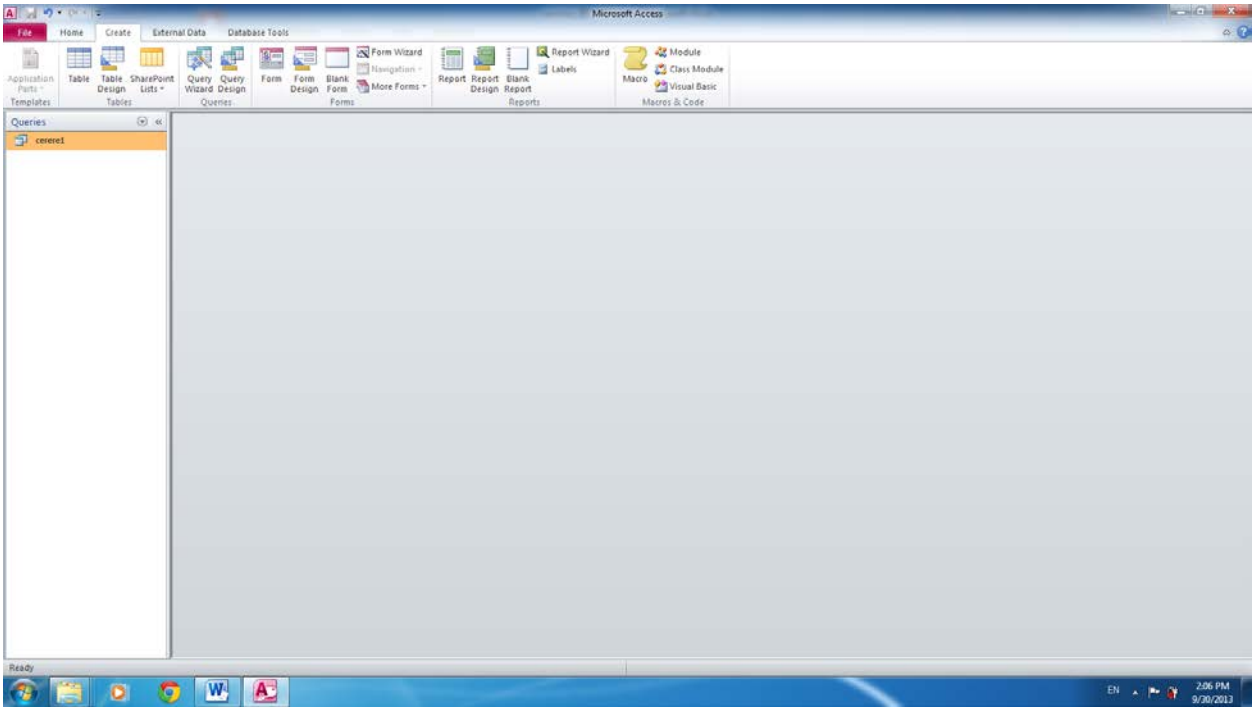



Se salveaza :




Se va salva cu denumirea **Cerere1** - se gaseste pe ramura **Query din navigator**

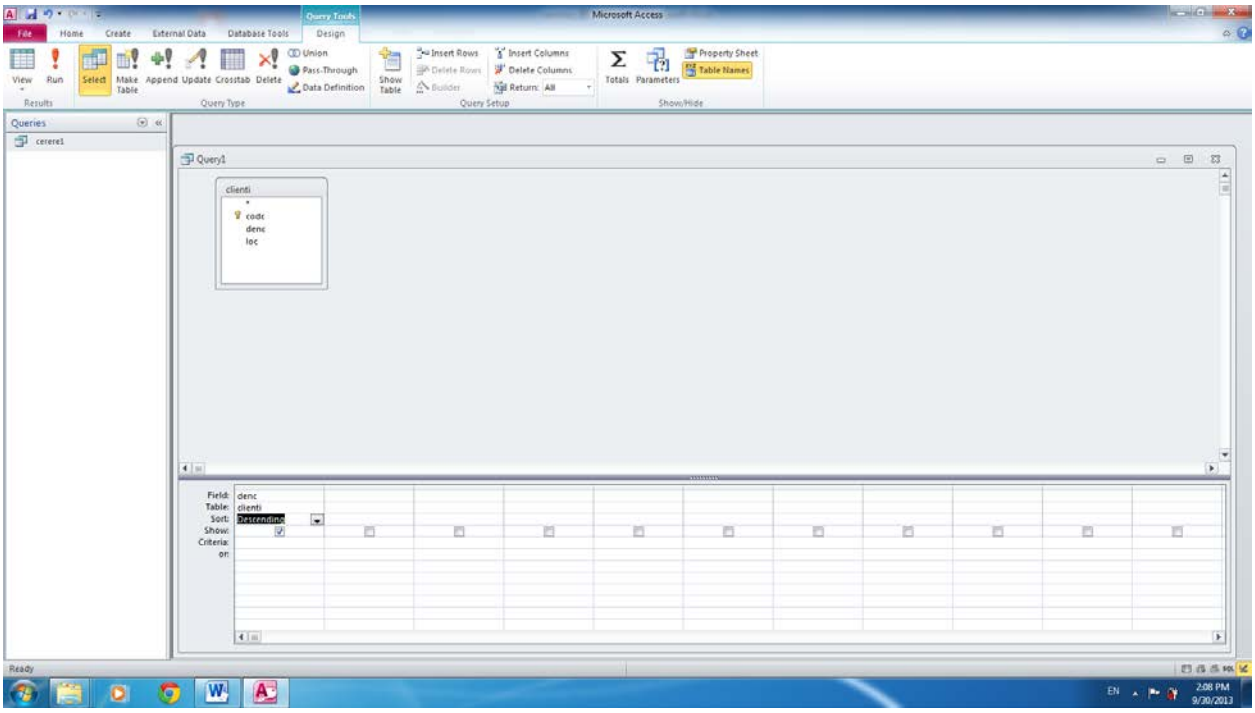



Se poate executa (dublu click pe cerere)

2. **Sa se afiseze toti clientii ordonati alfabetic descrescator**

Create/Query design/Show table se alege tabela **Clienti**




SELECT clienti.denc----------optiunea Field
FROM clienti----------optiunea table
ORDER BY clienti.denc DESC;  ------optiunea Sort

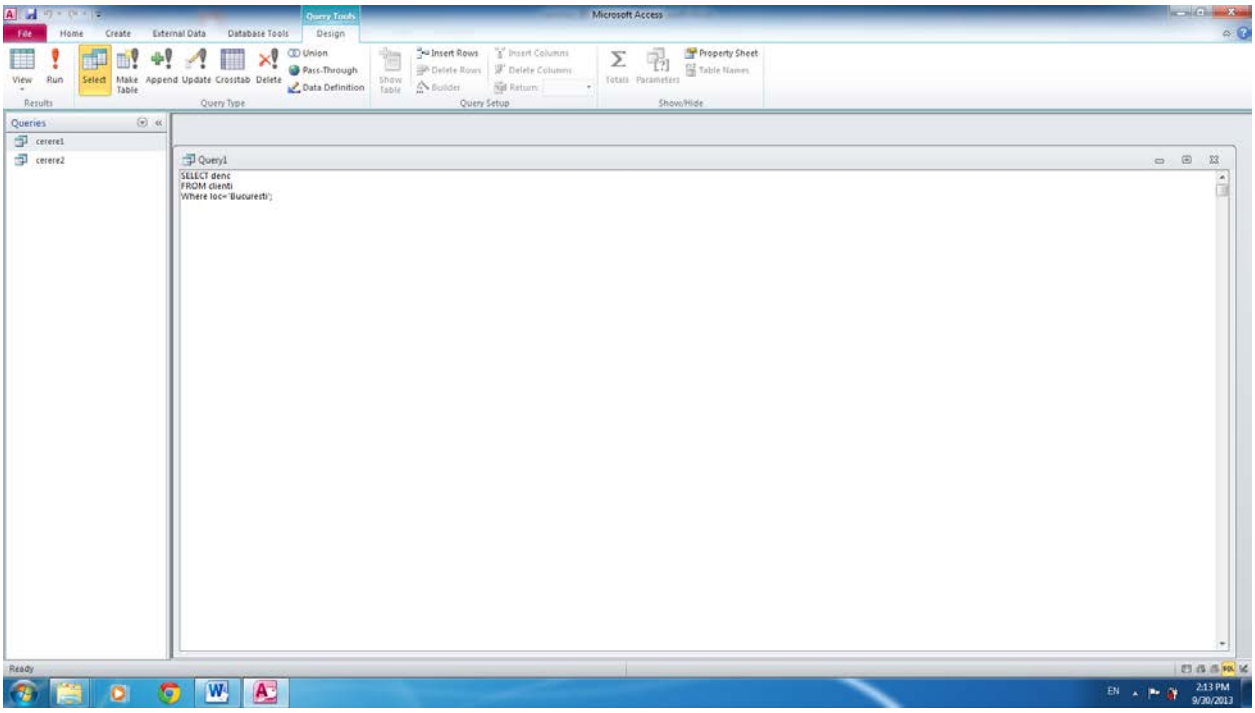
Se poate modifica codul SQL (optiunea View/SQL View).
Se salveaza: **cerere2,** apoi se executa

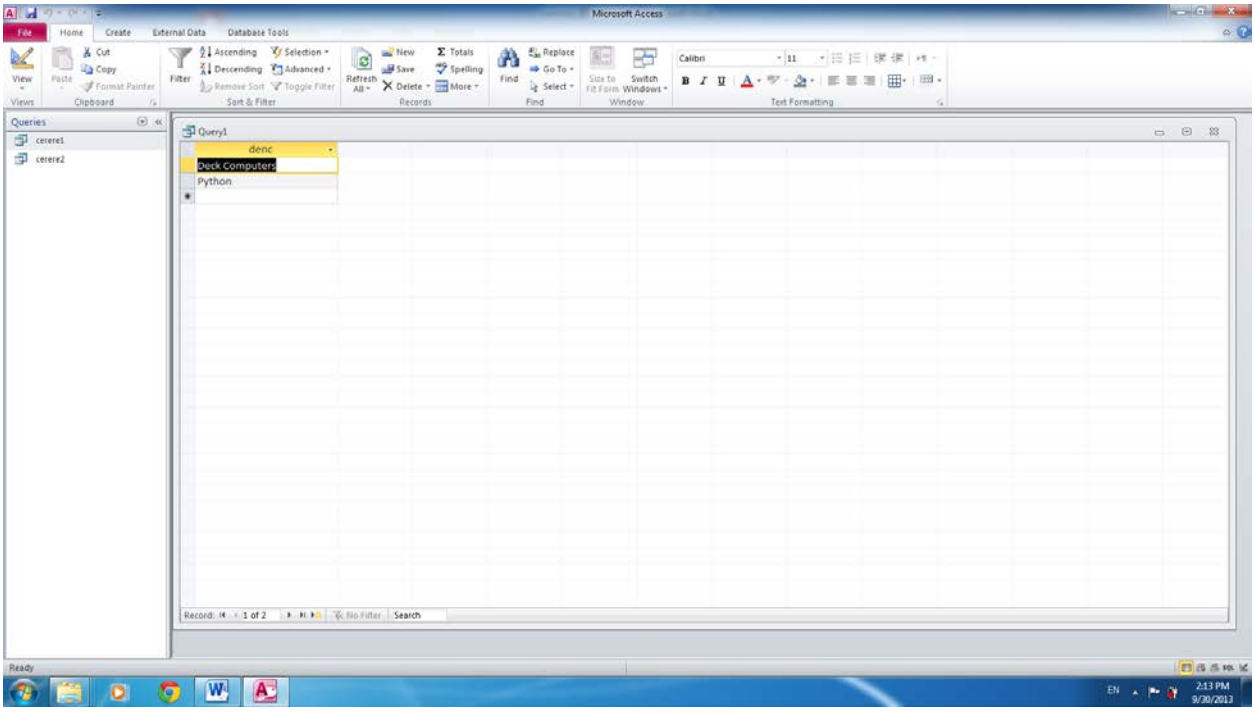
3. **Sa se afiseze clientii din Bucuresti**

Vom incerca sa scriem cod SQL:
Create/query design/ se alege tabela Clienti din Show Table
Apoi View/SQL view si modificam comanda SQL

SELECT denc
FROM clienti
Where loc='Bucuresti';

!!!!!!Atentie: se pune o sg data (;) la sf comenzii



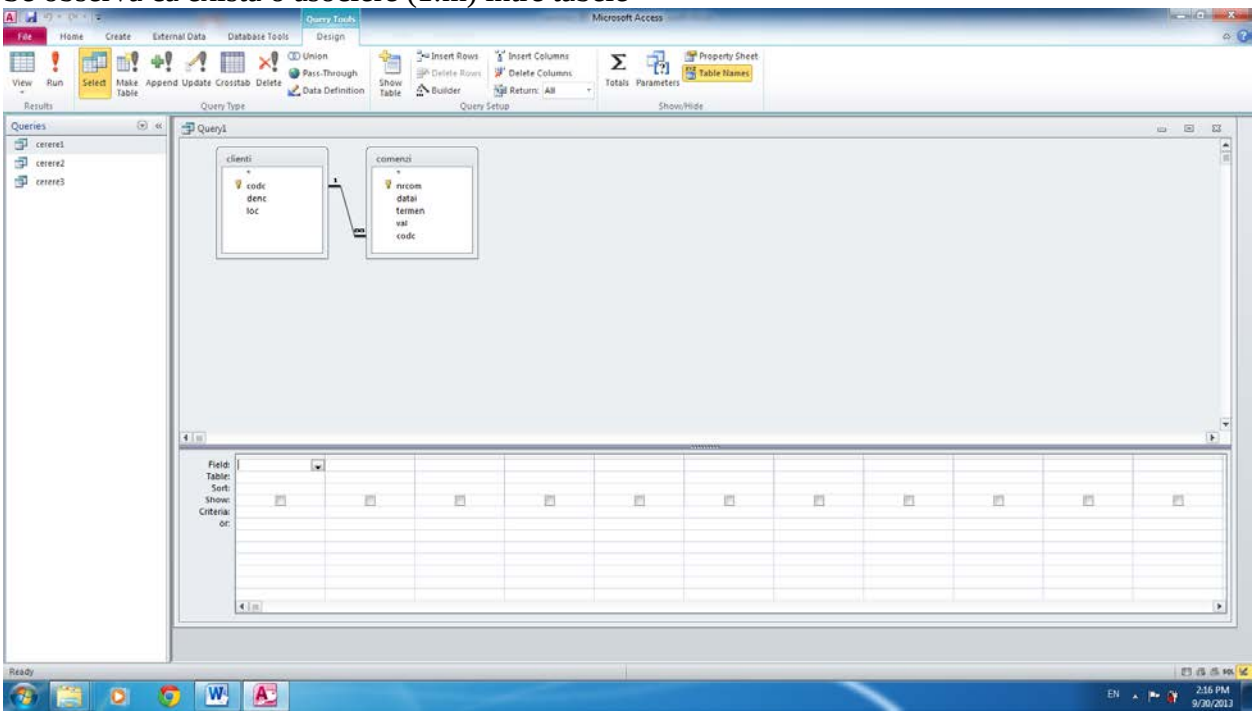

Se salveaza : **cerere3**
Se executa

**Lucru individual:**

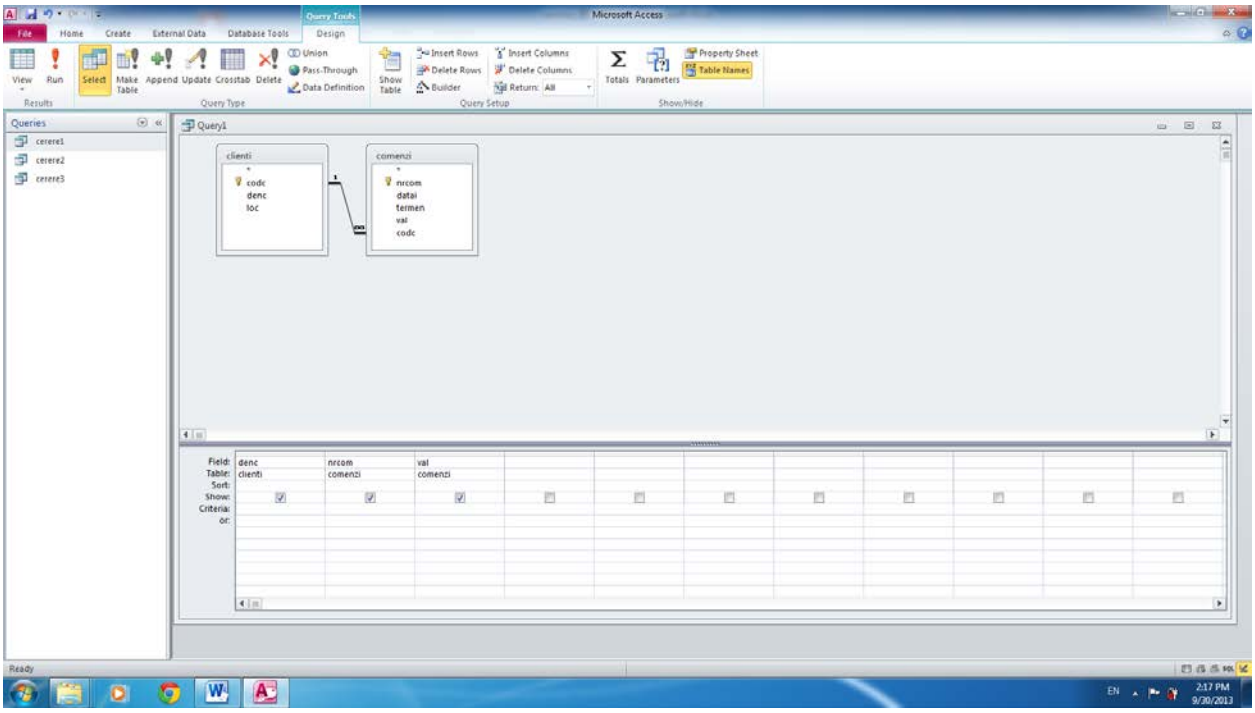
4. Sa se afiseze localitatile ordonate alfabetic
5. Sa se afiseze clientii care au in denumire “Computers” (se utilizeaza operatorul LIKE)
6. Sa se afiseze produsele cu stoc 0 (stoc=0)
7. Sa se afiseze produsele care au un pret unitar >50 RON (pu>50)

8. **Sa se afiseze pentru fiecare client, comenzile incheiate (nrcom, val)**

Realizarea jonctiunilor dintre tabele: **Clienti** si **Comenzi**
Create/Query design/se aleg tab Clienti si Comenzi
Se observa ca exista o asociere (1:m) intre tabele

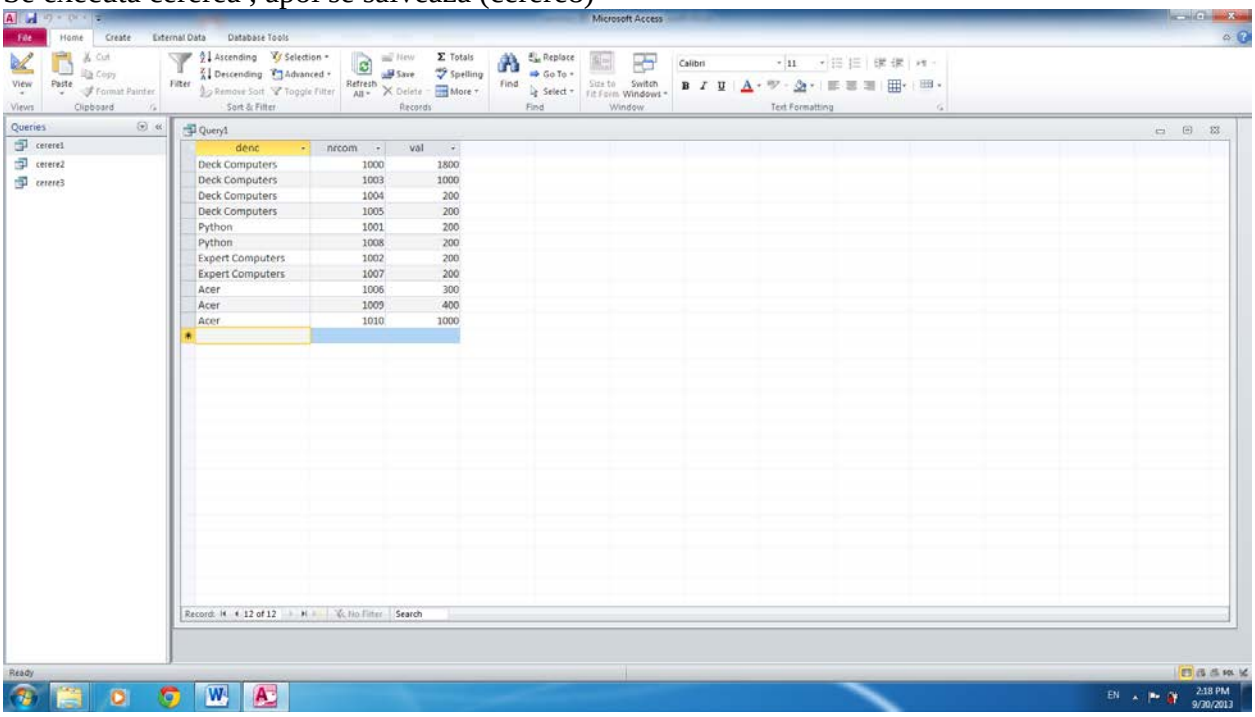



Se selecteaza atributul denc si se “trage” in Fields
Se selecteaza atributul nrcom si se “trage” in Fields
Idem atributul val




View/SQL view: se observa cum se realizeaza automat o jonctiune de egalitate intre cele doua tabele:
SELECT clienti.denc, comenzi.nrcom, comenzi.val
FROM clienti INNER JOIN comenzi ON clienti.codc = comenzi.codc;

Se executa cererea , apoi se salveaza (cerere8)

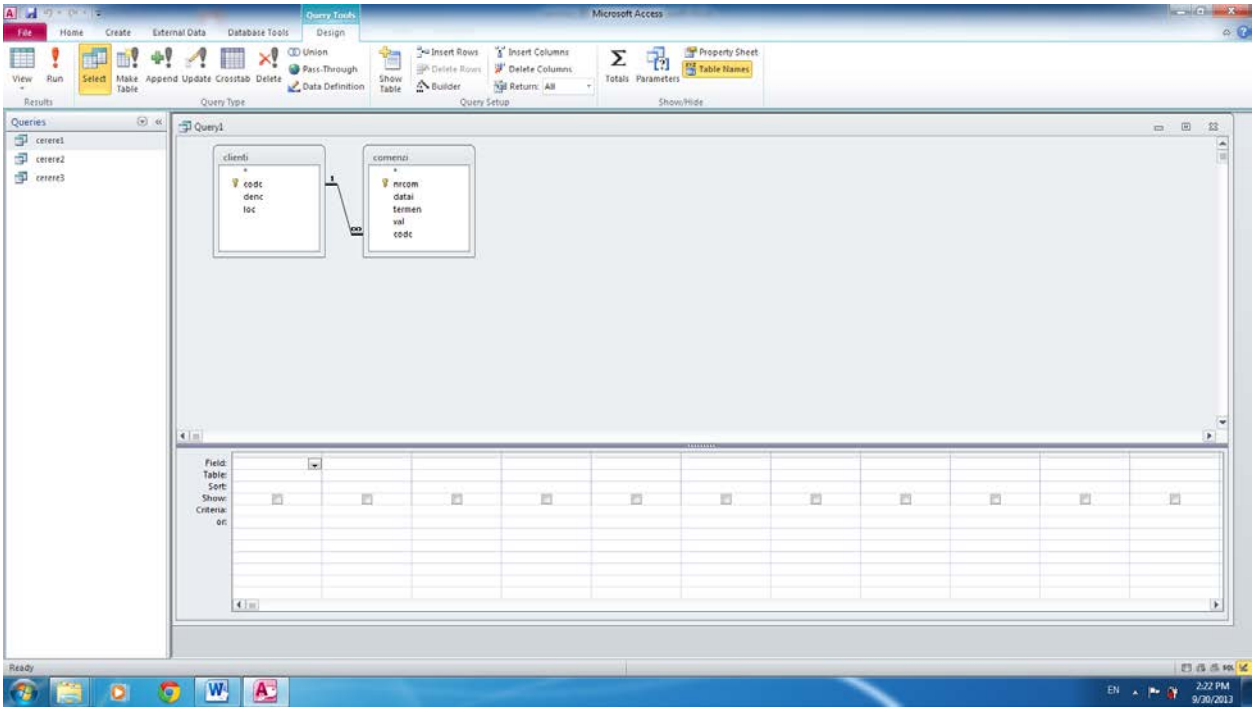


9. Sa se afiseze pentru fiecare client cate comenzi are

Utilizarea clauzei Group BY atribut de grupare

SELECT atribut, atribut, ....
FROM tabela1, tabela2, [conditie de jonctiune]
WHERE cond de filtrare pe tupluri
GROUP BY conditie de grupare
HAVING conditie de filtrare pe grupuri
ORDER BY atribut1, atribut 2

**CREATE/query design/**se selecteaza tabela Clienti si tabela Comenzi. Atributul de grupare este atributul **codc** si trebuie sa contorizam pt fiecare client cate comenzi are (se utilizeaza functia count())



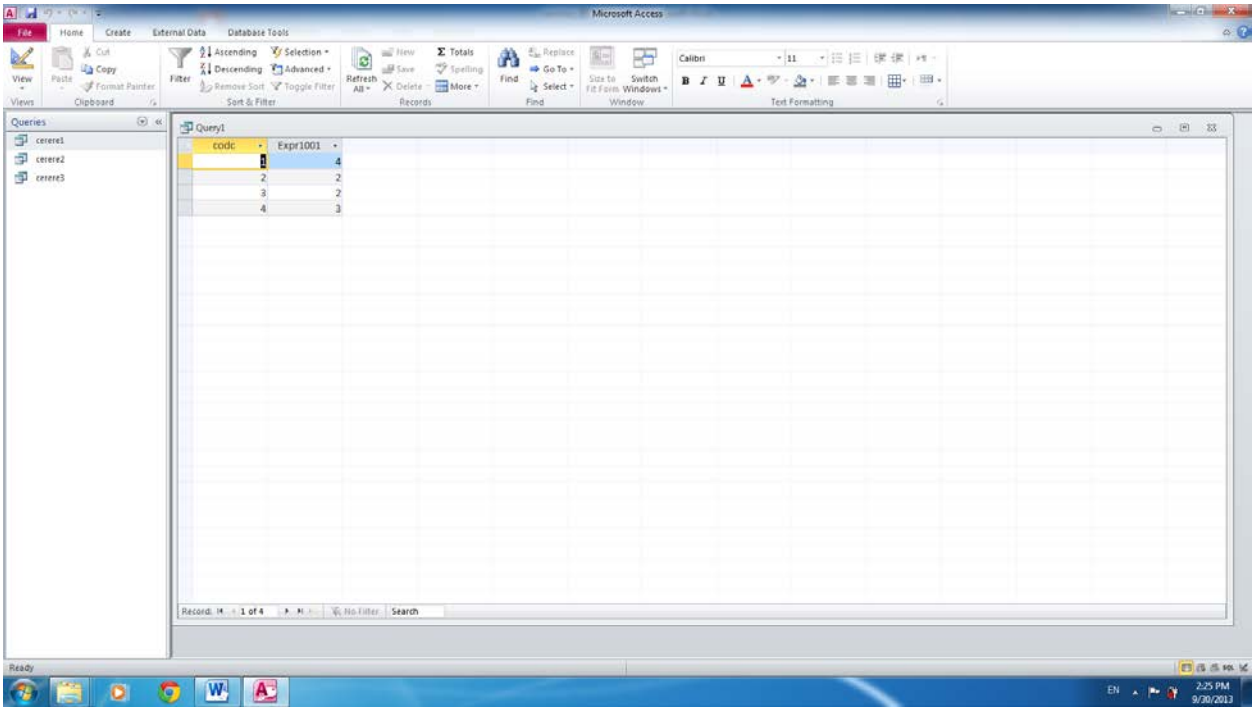

Se va scrie cod SQL: View/SQL view

SELECT clienti.codc, count(comenzi.nrcom)
FROM clienti INNER JOIN comenzi ON clienti.codc = comenzi.codc
Group by clienti.codc;

Atributul **codc** apare in ambele tabele, deci trebuie sa specificam denumirea tabelei de unde este preluat atributul: **clienti.codc**
Mod de referire: denumiretabela.denumire atribut

Revenim in Designer View si observam ce modificari apar in interfata, dupa ce am scris codul SQL

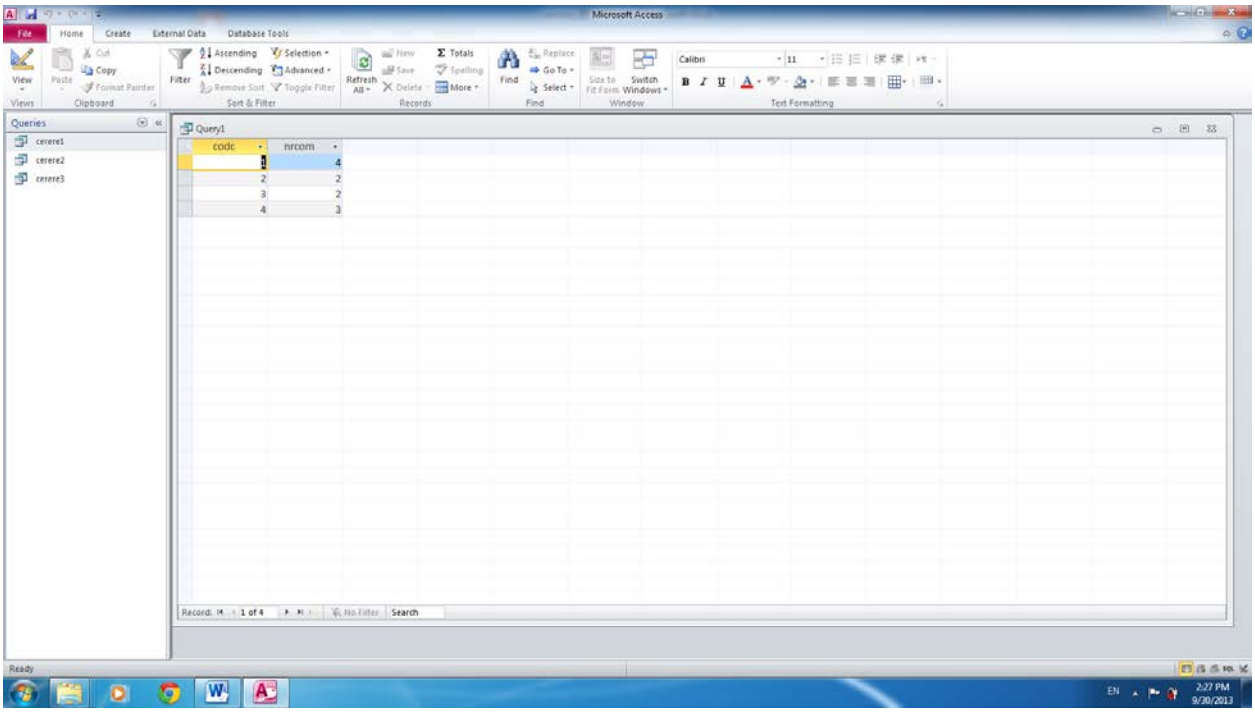
Se executa cererea




Observam ca afiseaza in **caption**: **expr1001** –adica este o expresie, putem sa modificam utilizand clauza AS :

```
SELECT clienti.codc, count(comenzi.nrcom) as nrcom
FROM clienti INNER JOIN comenzi ON clienti.codc = comenzi.codc
Group by clienti.codc;
```

Se executa cererea:

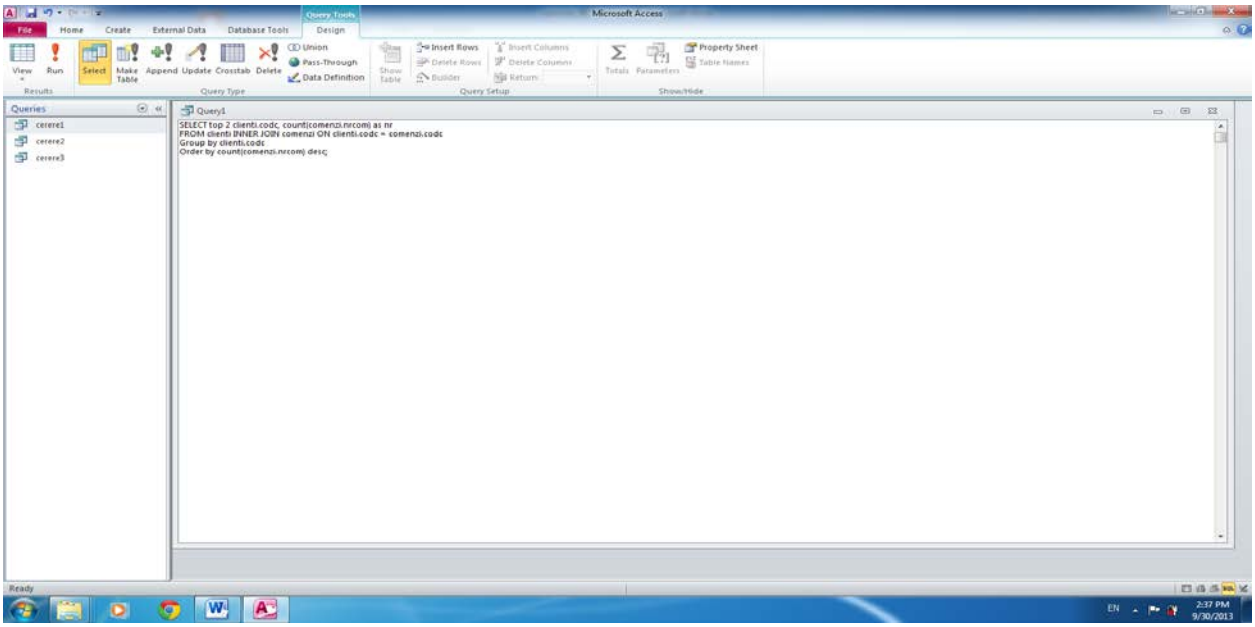



Reveniti in Designer View pt a observa ce modificari apar in interfata vizuala

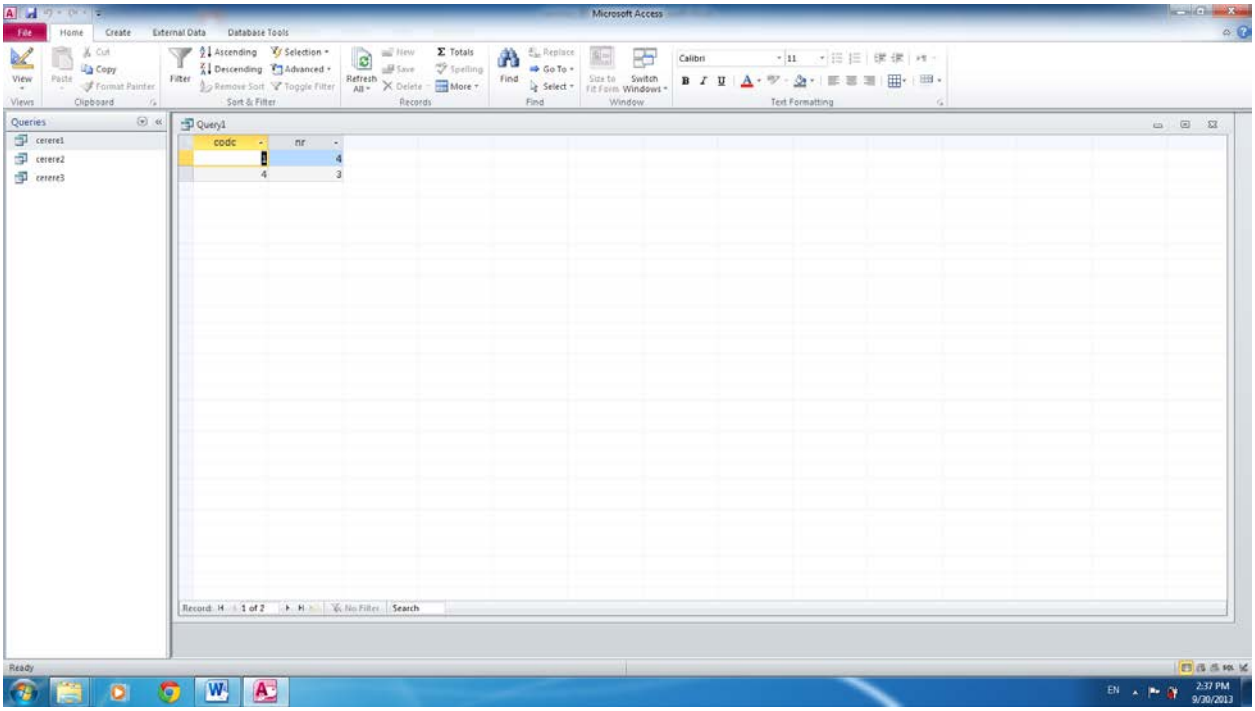
10. Sa se afiseze primii doi clienti dupa numarul de comenzi incheiate (analiza). Se va utiliza cererea de la p 9.

Se va modifica codul SQL (View/SQL View):

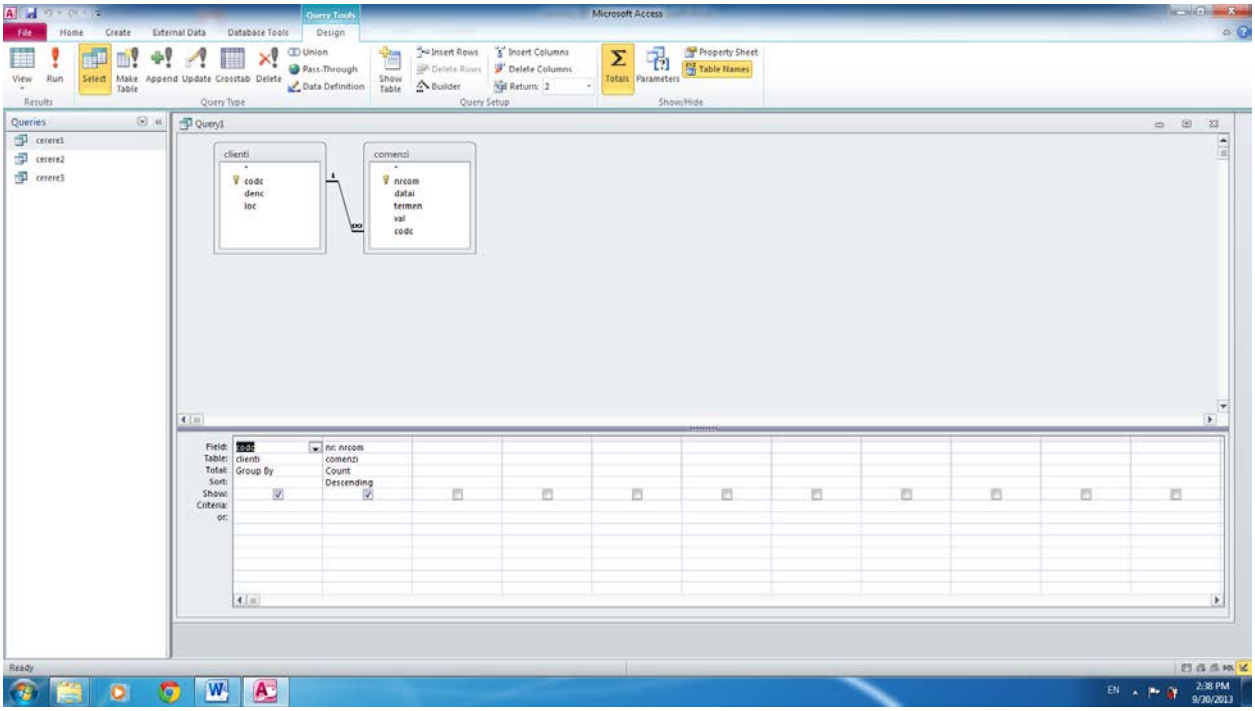
```
SELECT top 2 clienti.codc, count(comenzi.nrcom) as nr
FROM clienti INNER JOIN comenzi ON clienti.codc = comenzi.codc
Group by clienti.codc
Order by count(comenzi.nrcom) desc; ---------afisam descrescator dupa nr de comenzi
```



Se executa




Reveniti in View|Designer View pt a observa ce modificari apar in interfata vizuala

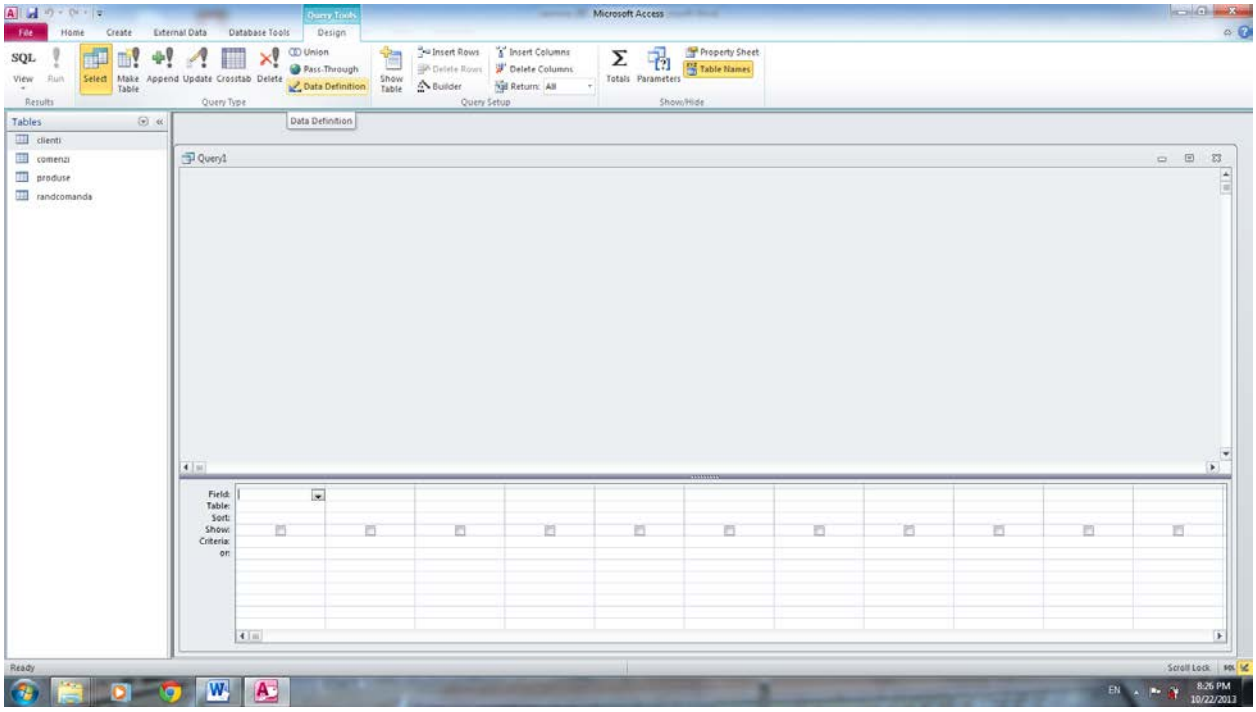



11.

**Studiu individual:**

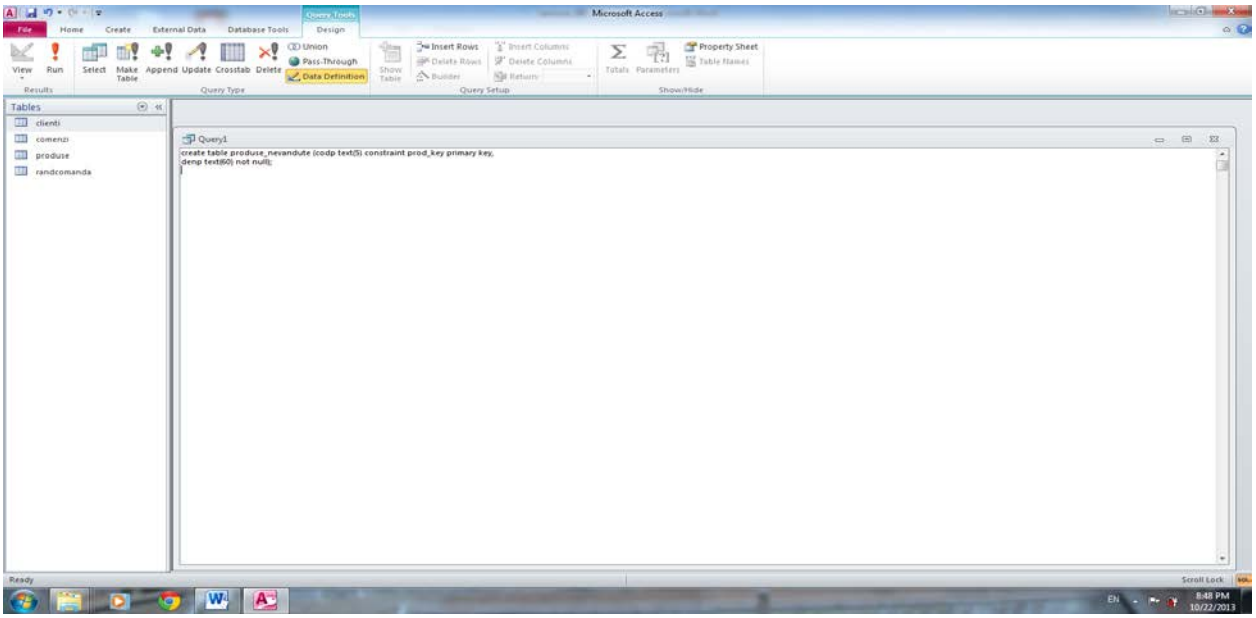
1. Sa se afiseze numarul de clienti din fiecare localitate
2. Sa se afiseze vanzarile pentru fiecare produs (se va afisa atributul **denp**).
   O vanzare=pu*cant. Grupare se face pe produs (atributul **codp**), total vanzare pe fiecare produs inseamna **sum(vanzare)**
3. Sa se afiseze primele doua produse care au fost cel mai bine vandute

12. Sa se creeze tabela **Produse_nevandute**, utilizand **Query Designer/data definition**

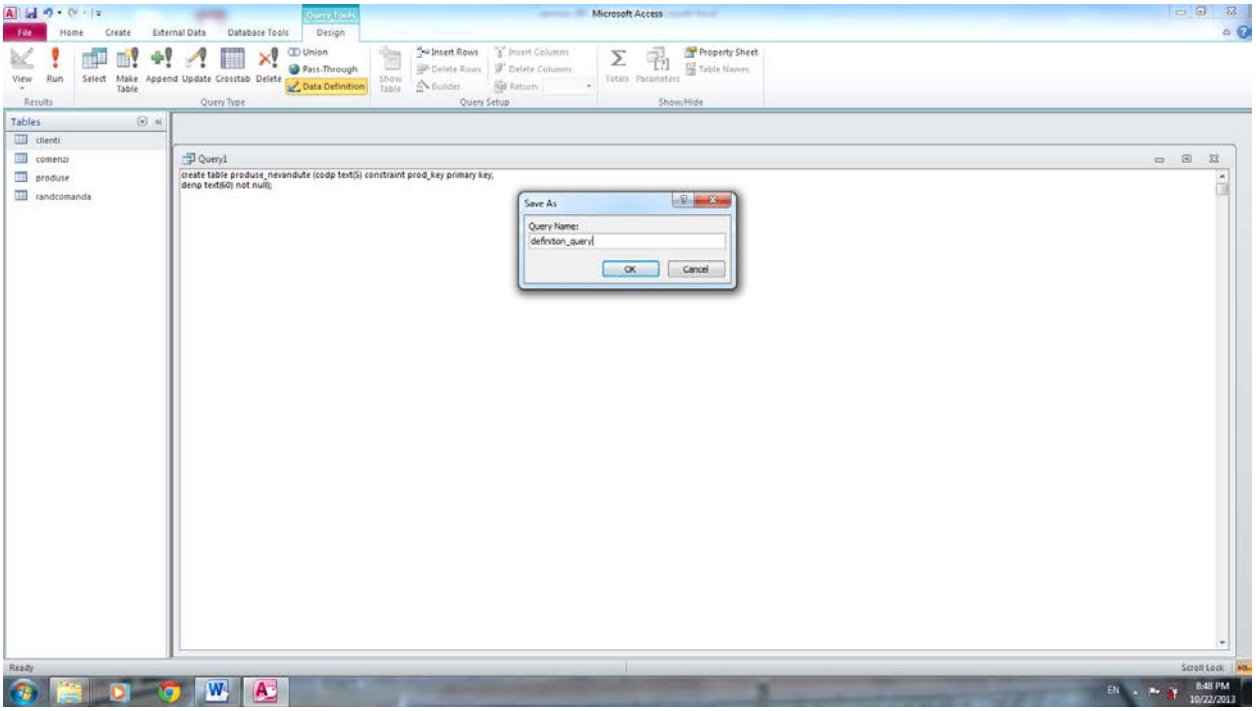



Se introduce urm cod SQL

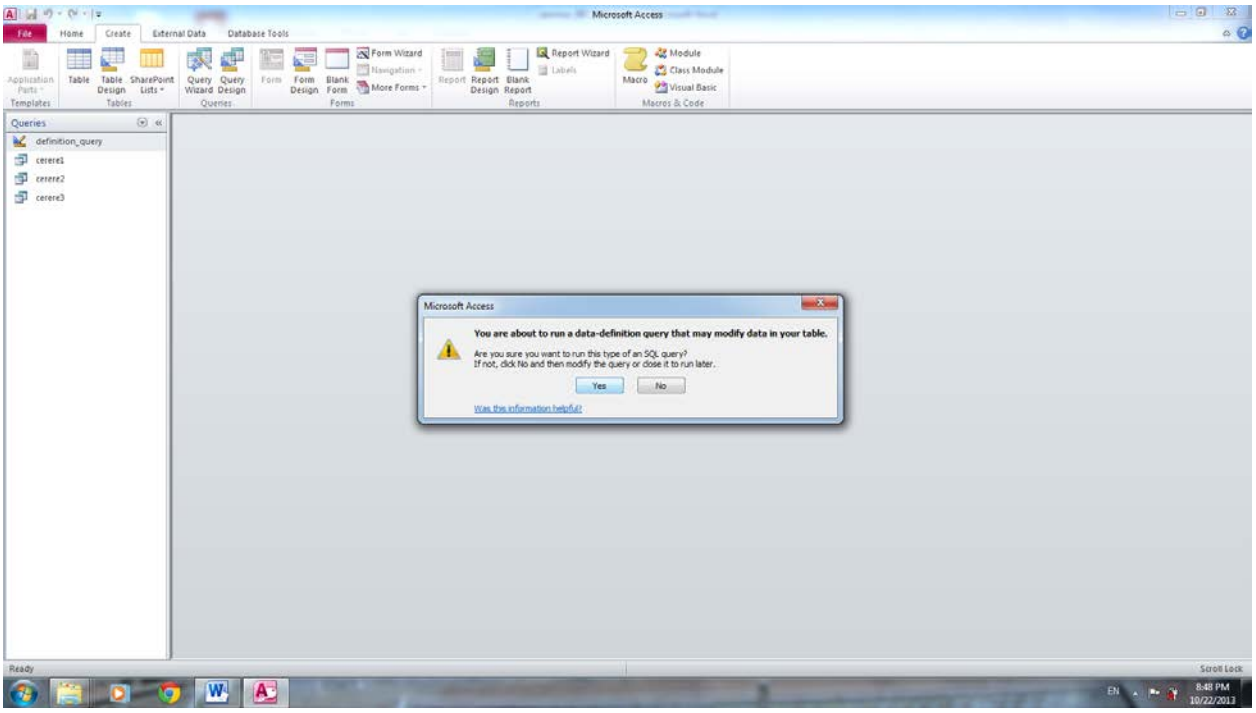
create table produse_nevandute (codp text(5) constraint prod_key primary key, denp text(60) not null);



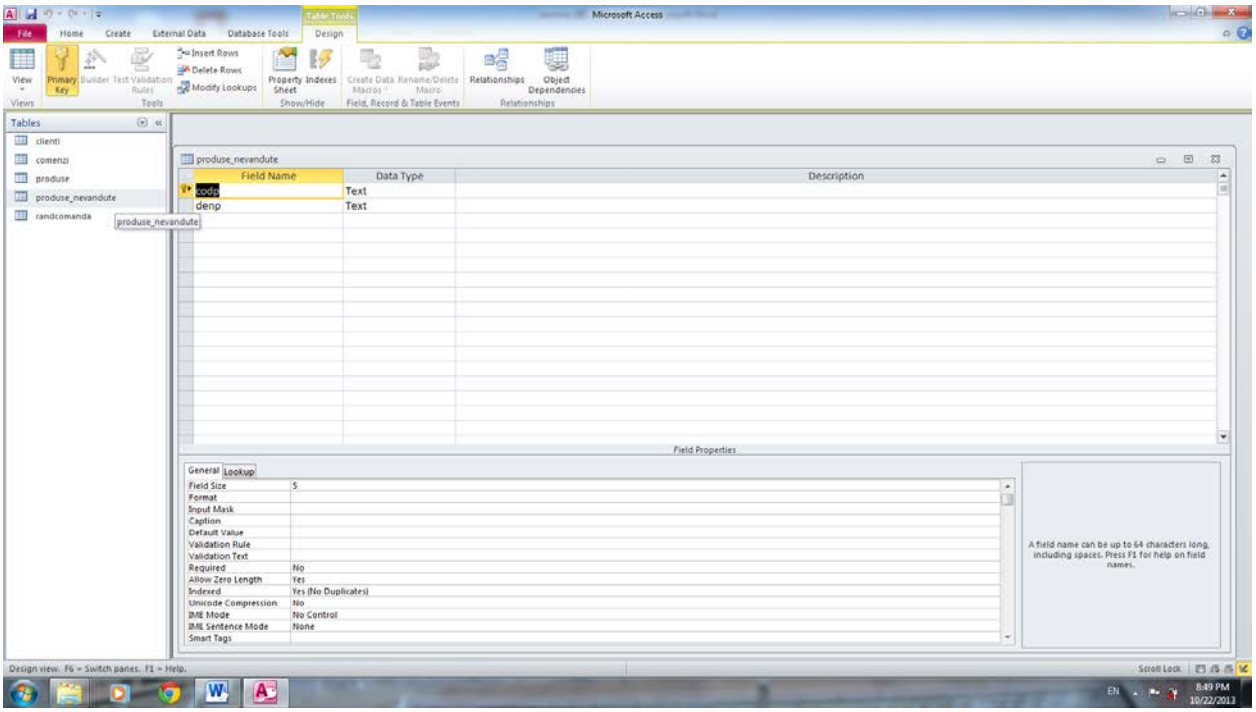

Se salveaza:




Se executa cererea:

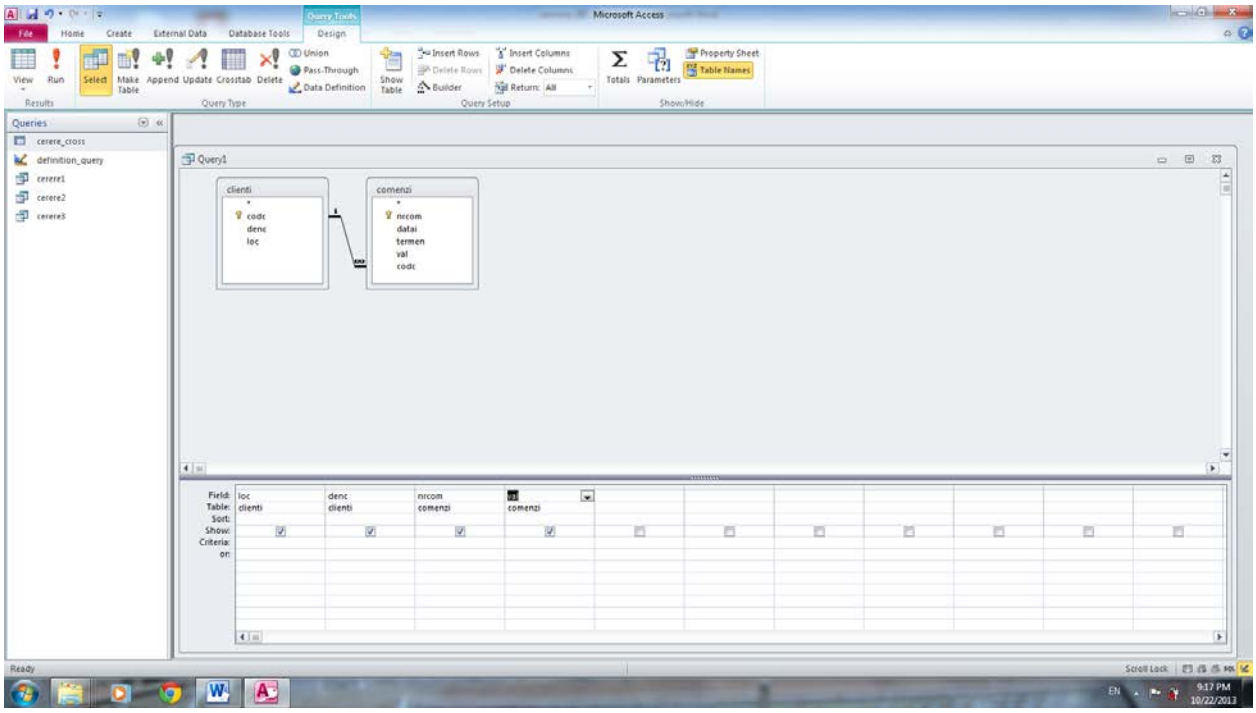



Verificam daca s-a creat tabela

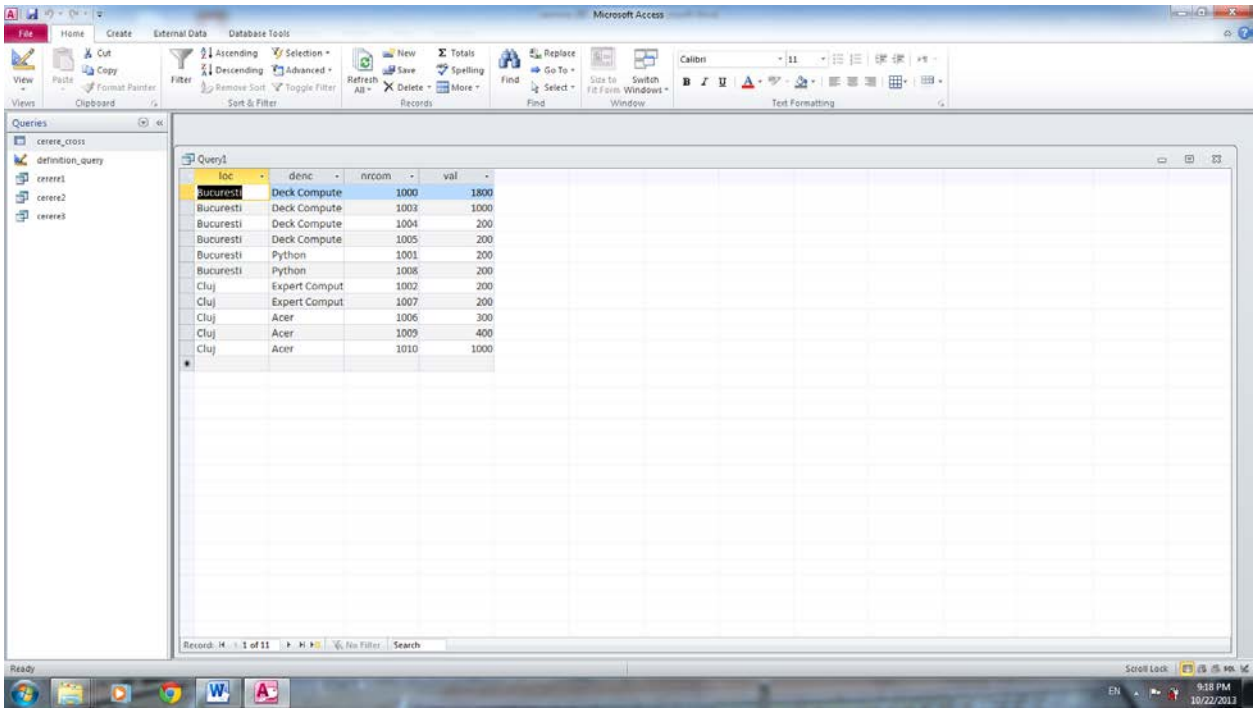



13. Sa se creeze o cerere de tip (Pivot table):

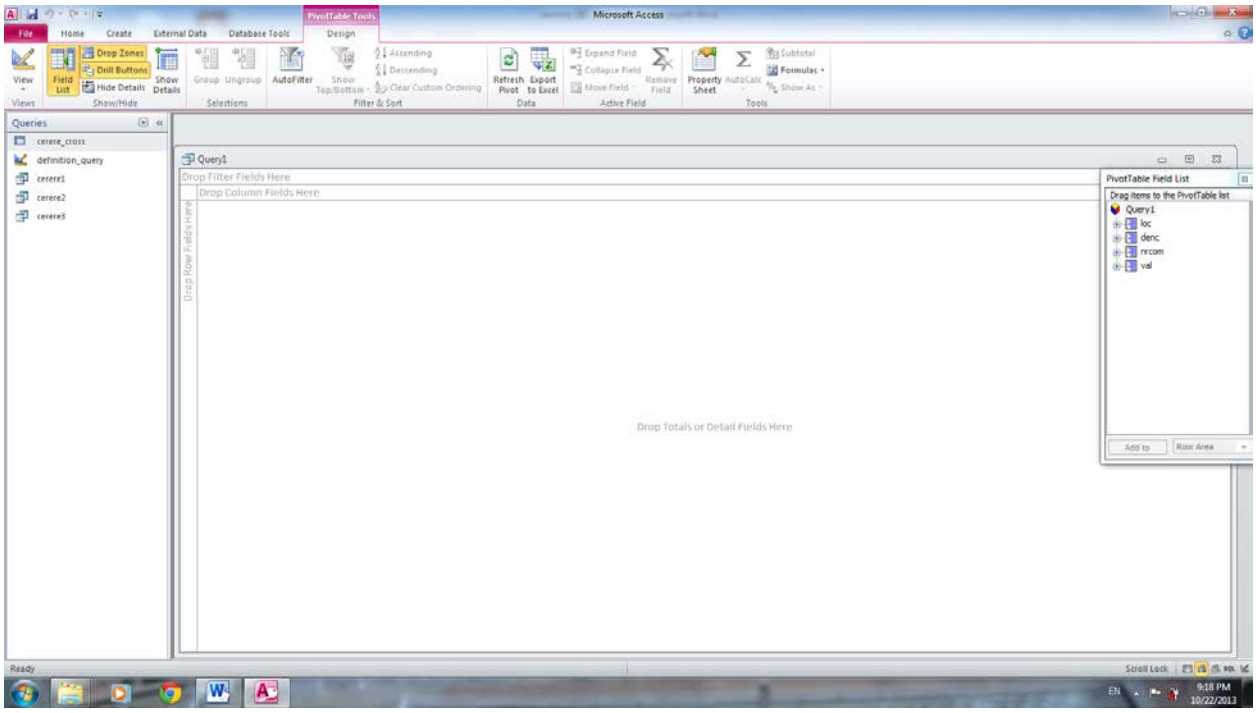
    Create/Query Design/Show table: tab Clienti, Comenzi



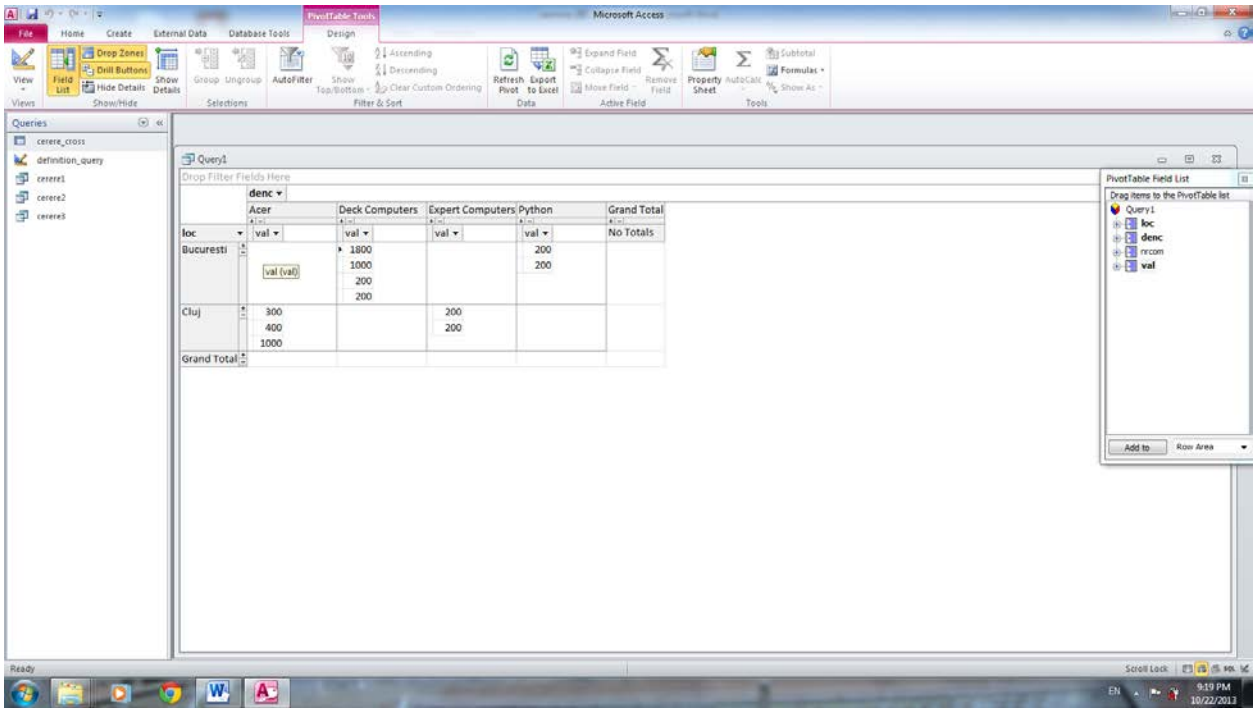

Daca se executa cererea:

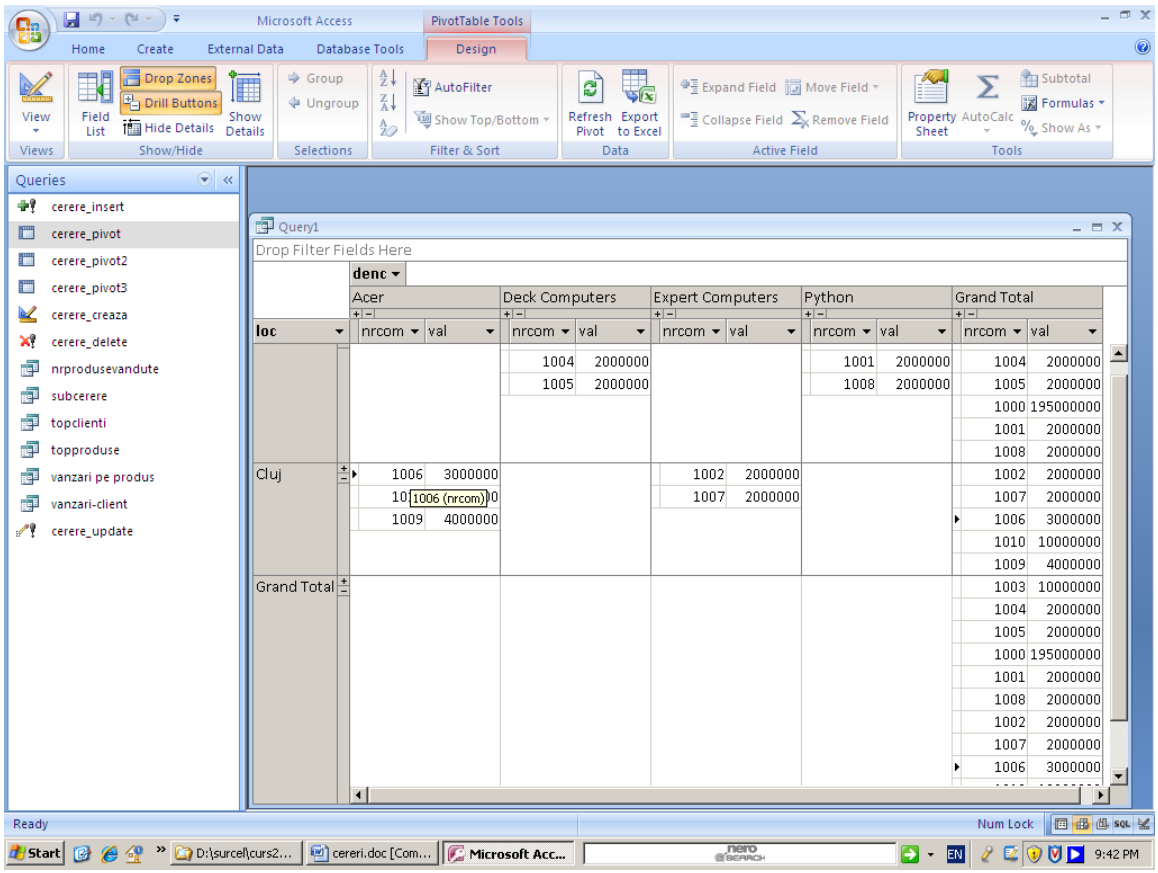



Apoi se selecteaza View/pivottableview

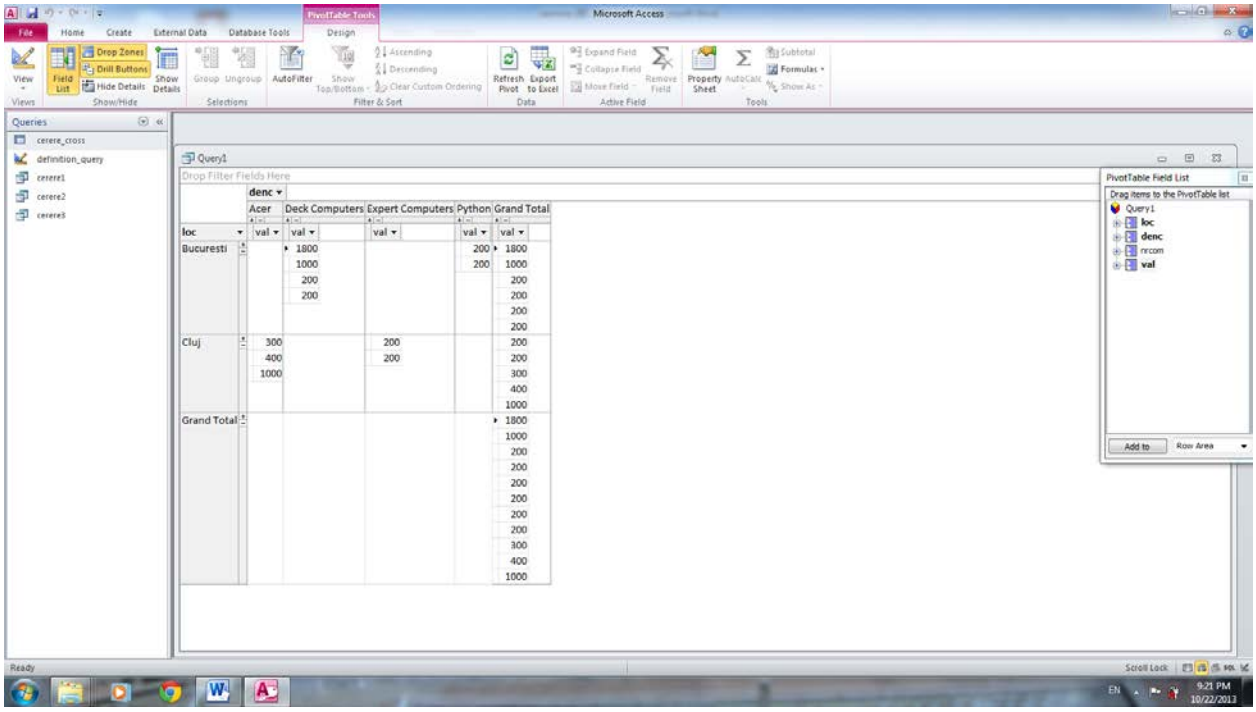



Se trage atributul **loc** pe coloane, atrib **denc** pe randuri si in celule atrib **val**

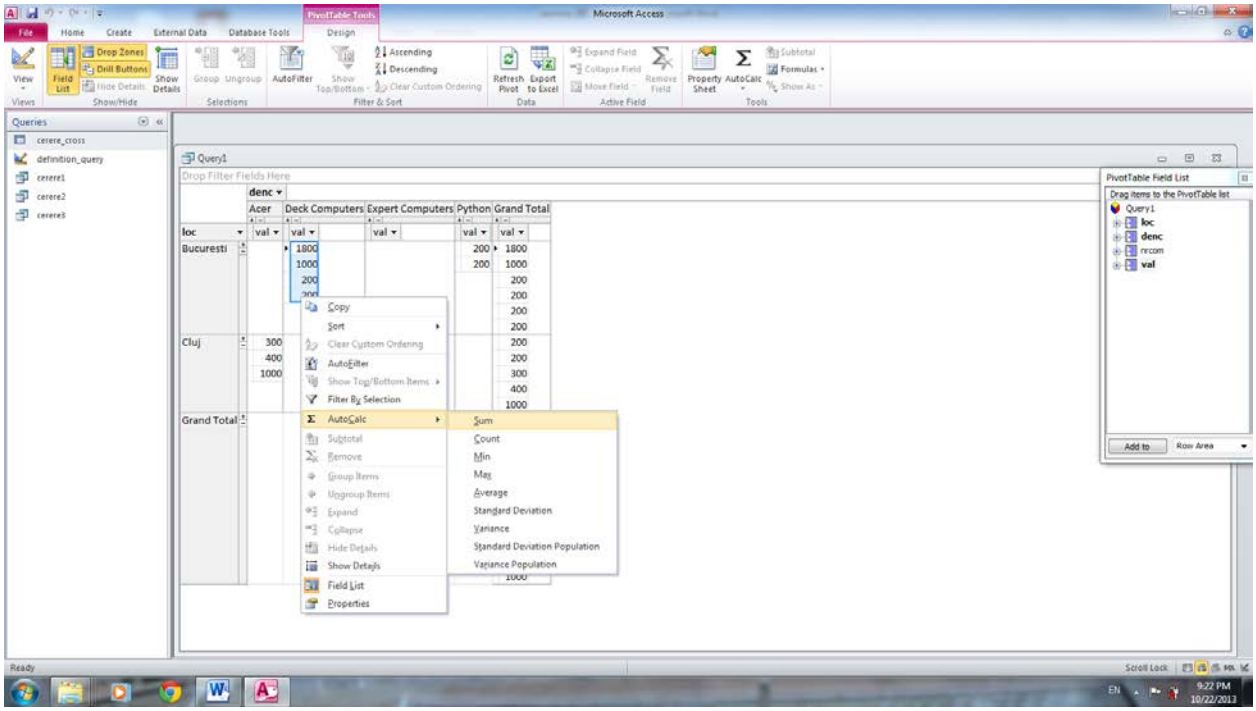


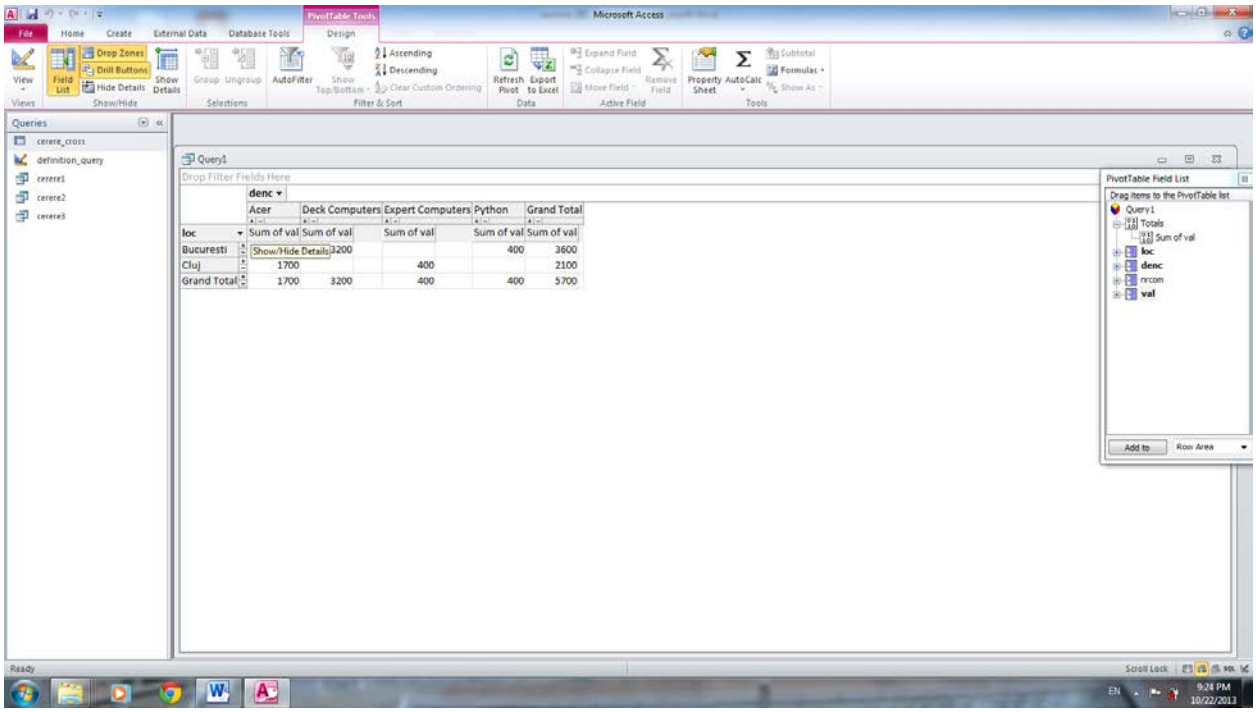



Am facut click pe grand total (pe +) si se afiseaza fara functii de agregare

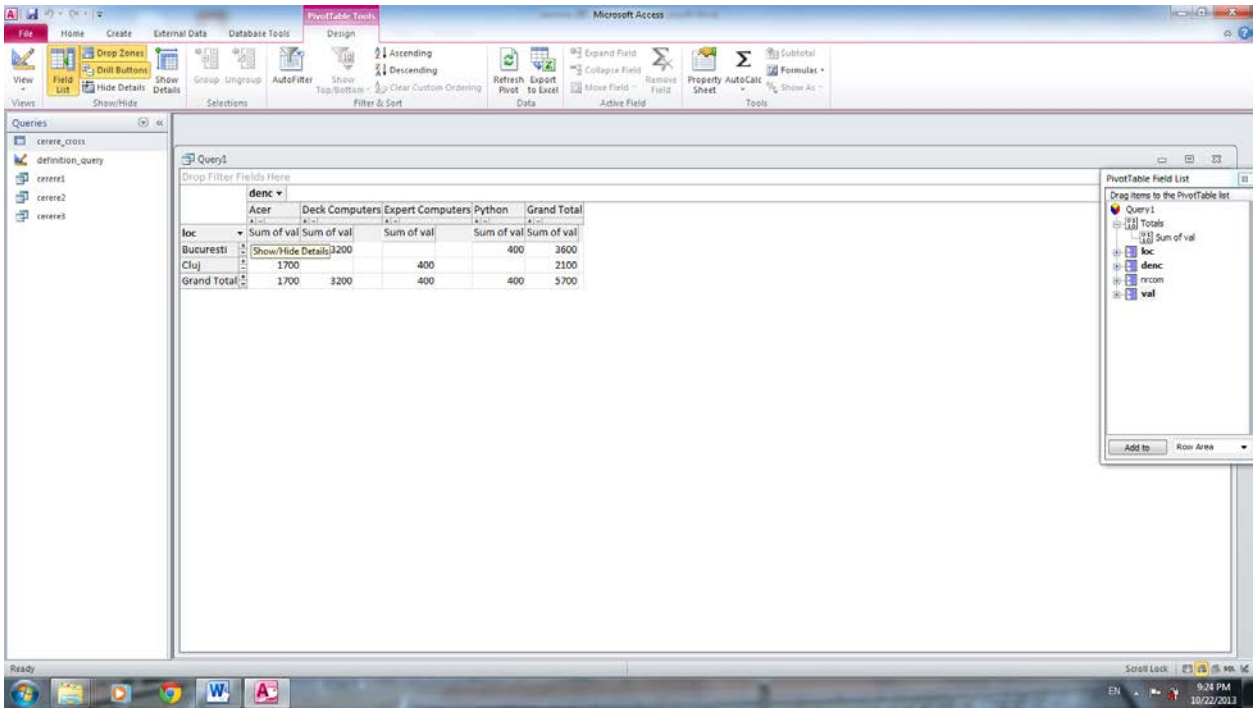


Pot face sum(val) prin selectarea valorile si click pe dreapta

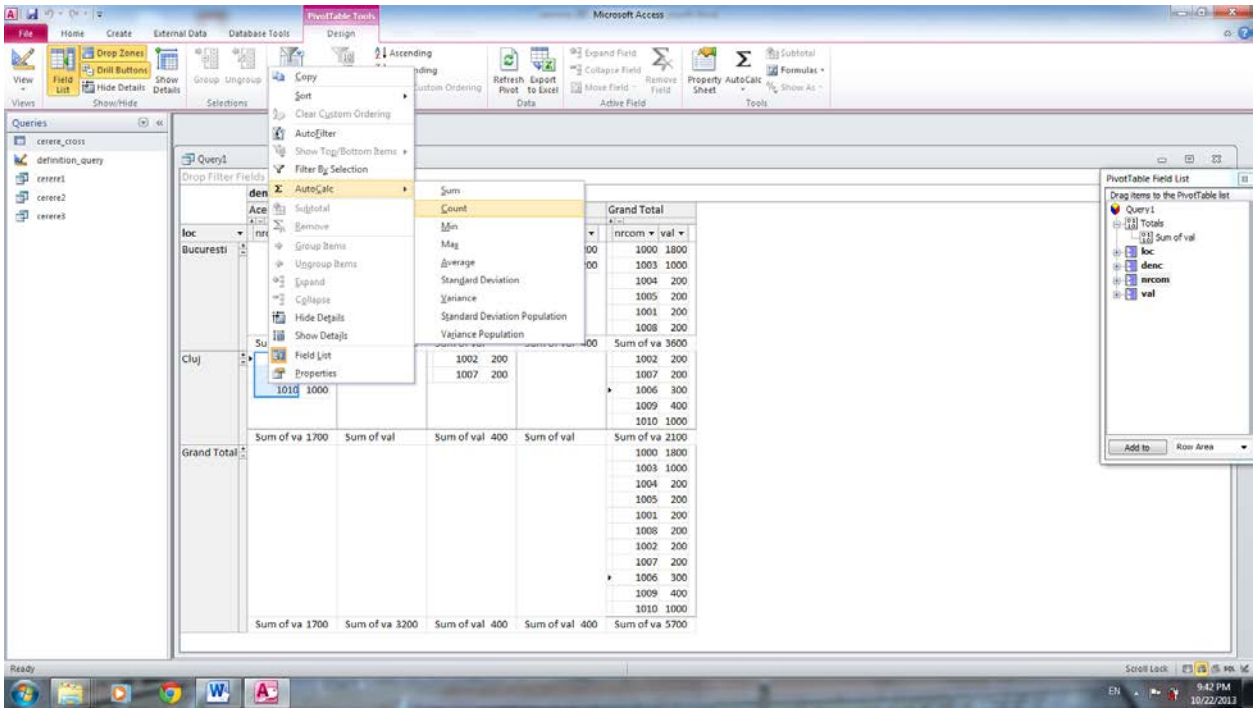


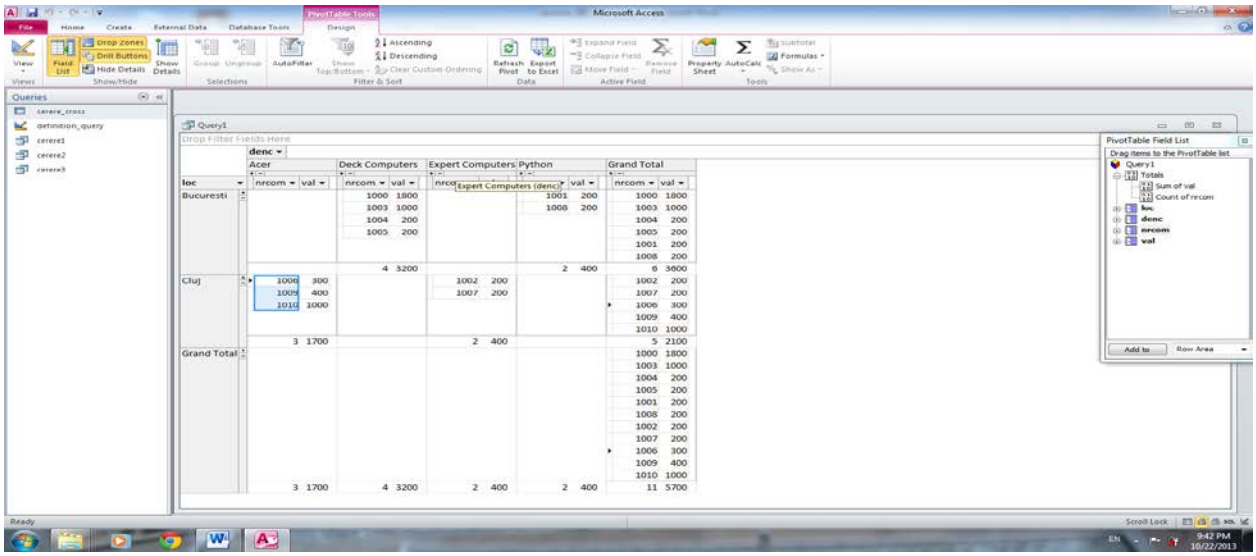



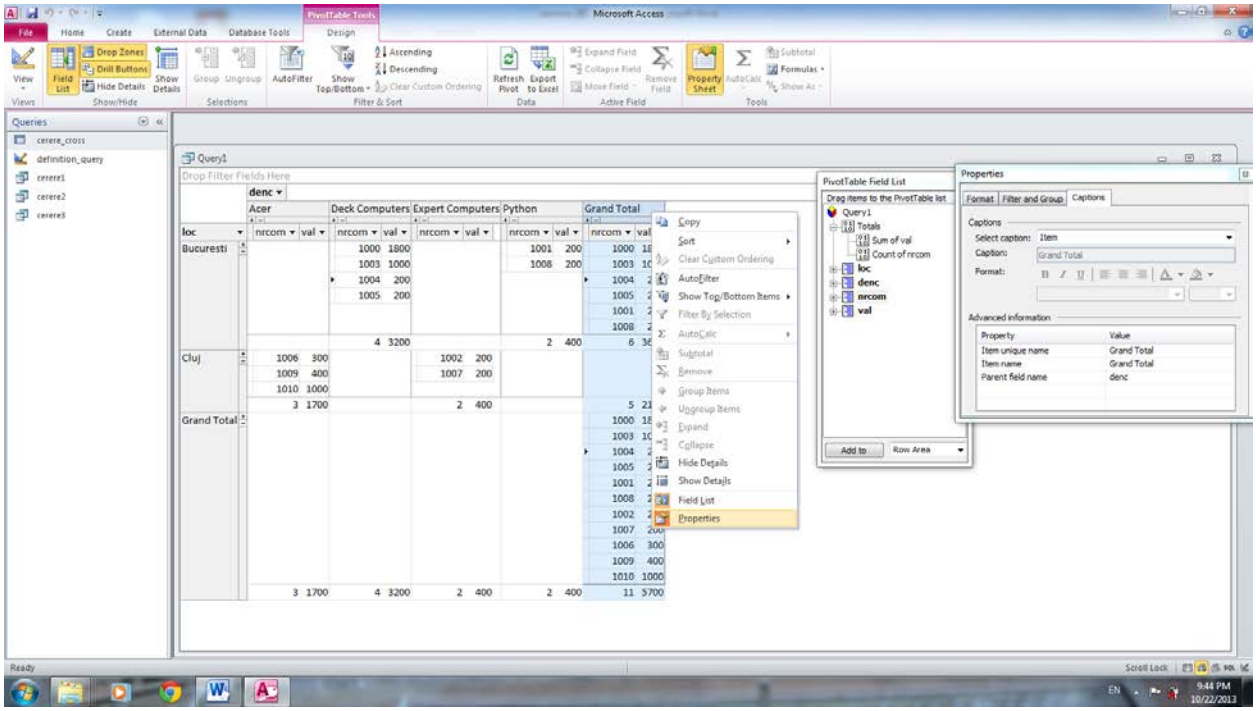

Adaugam si atributul nrcom in celule



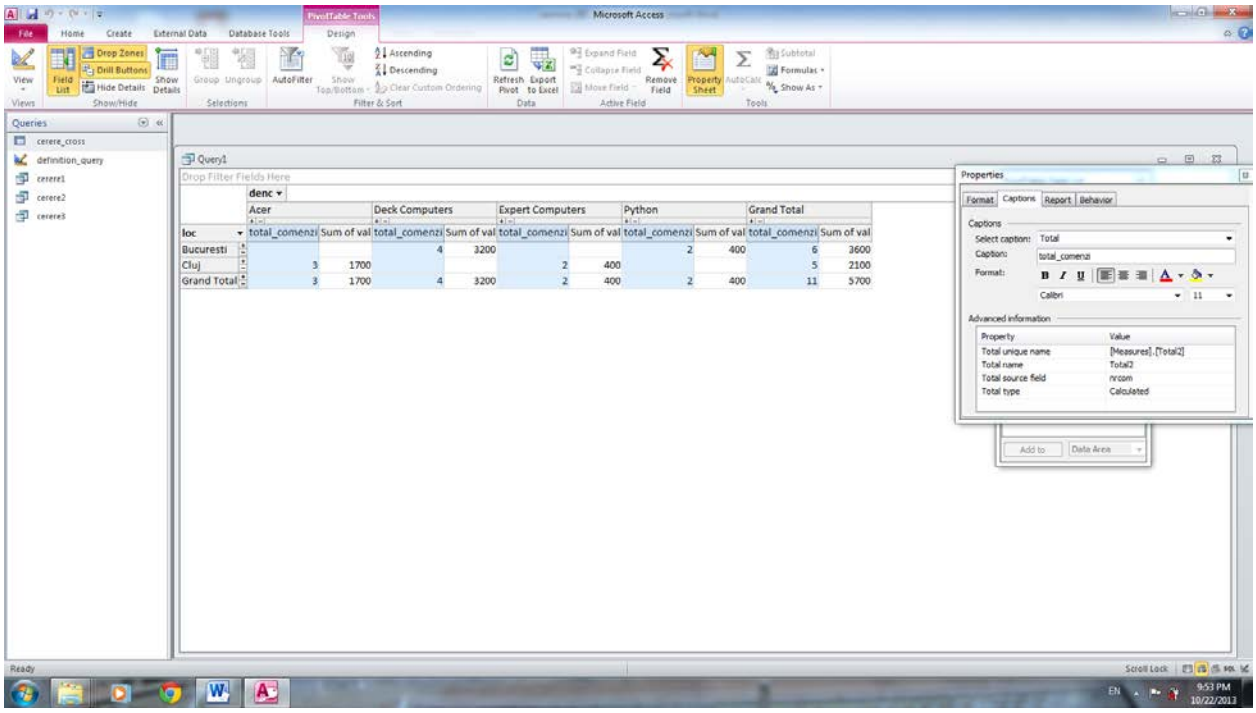

Apoi calculam: total numar de comenzi







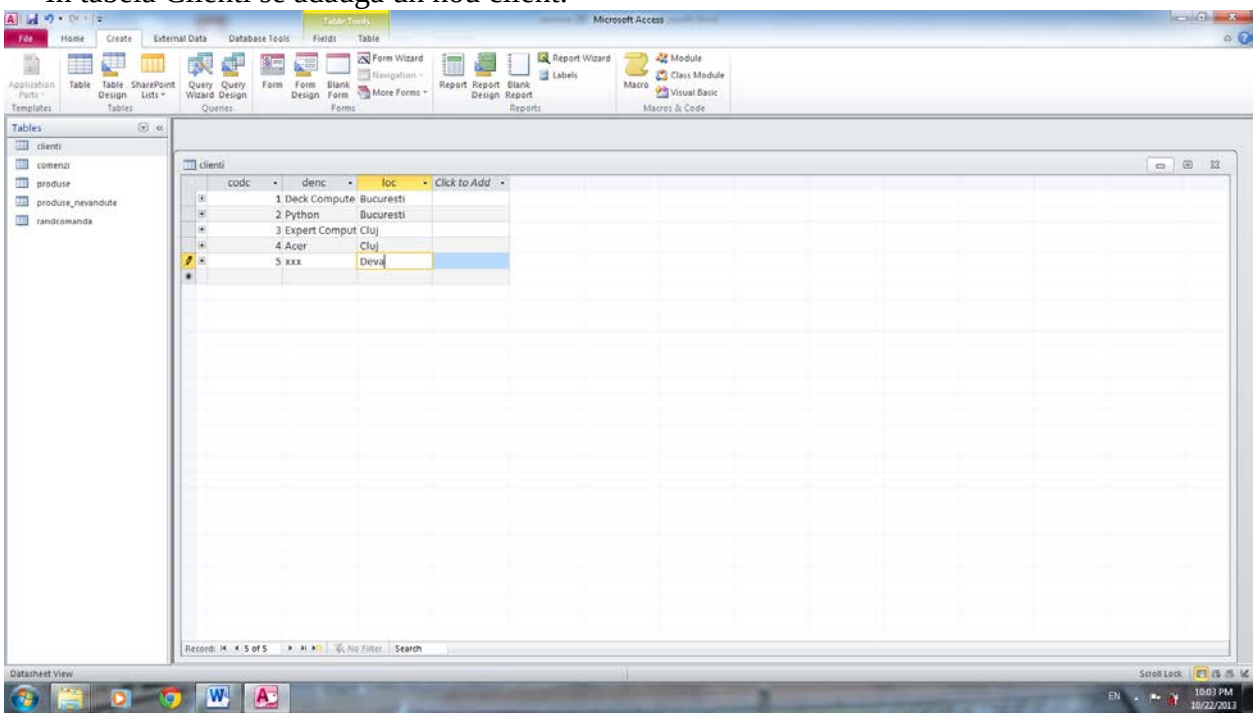
Se poate modifica **caption**



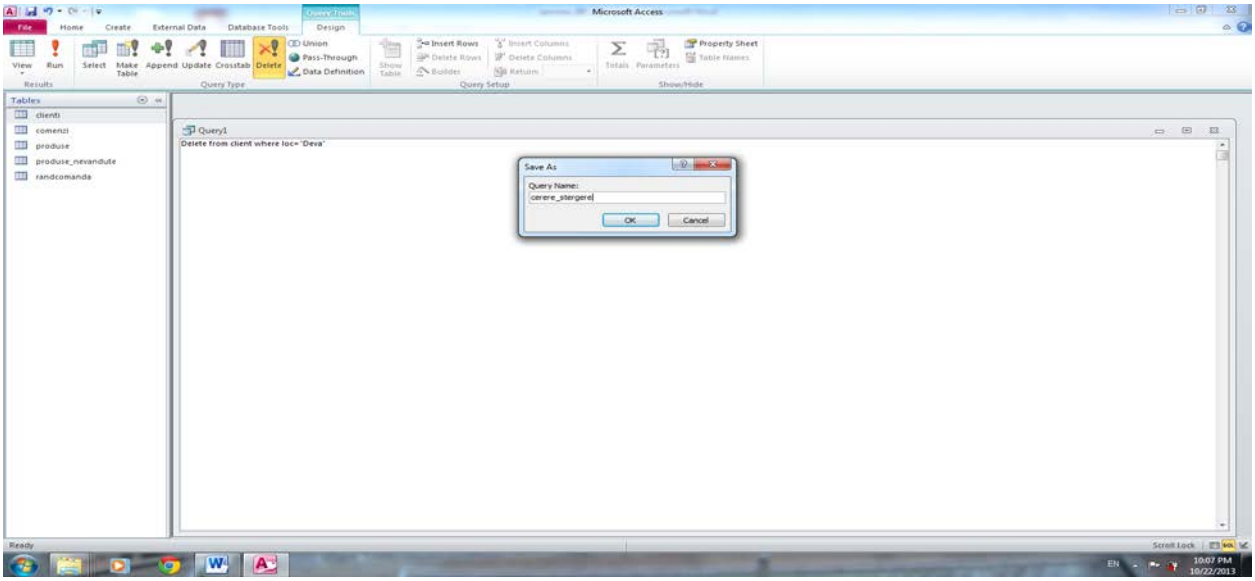

**14. Action queries=** permit actualizarea tabelelor (insert/update/delete)

Delete from clienti where loc=*"Deva"*
UPDATE produse SET *stoc = stoc-10*
WHERE codp="MP1";

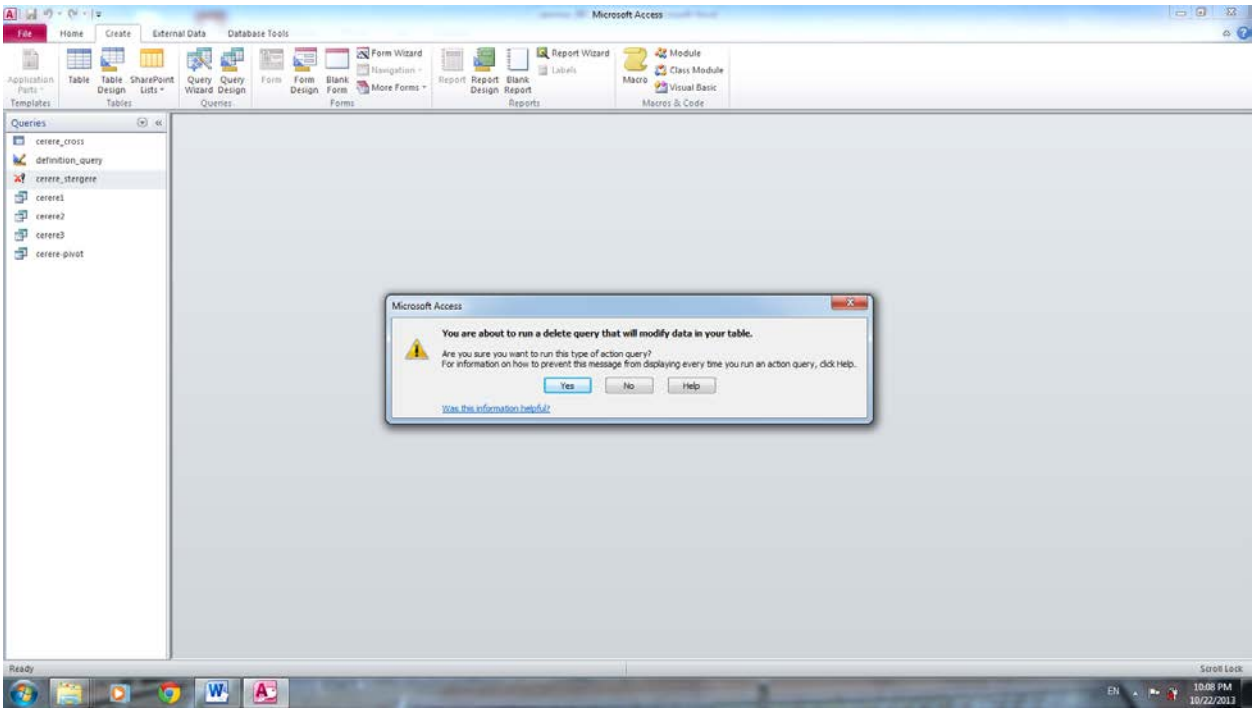
In tabela Clienti se adauga un nou client:



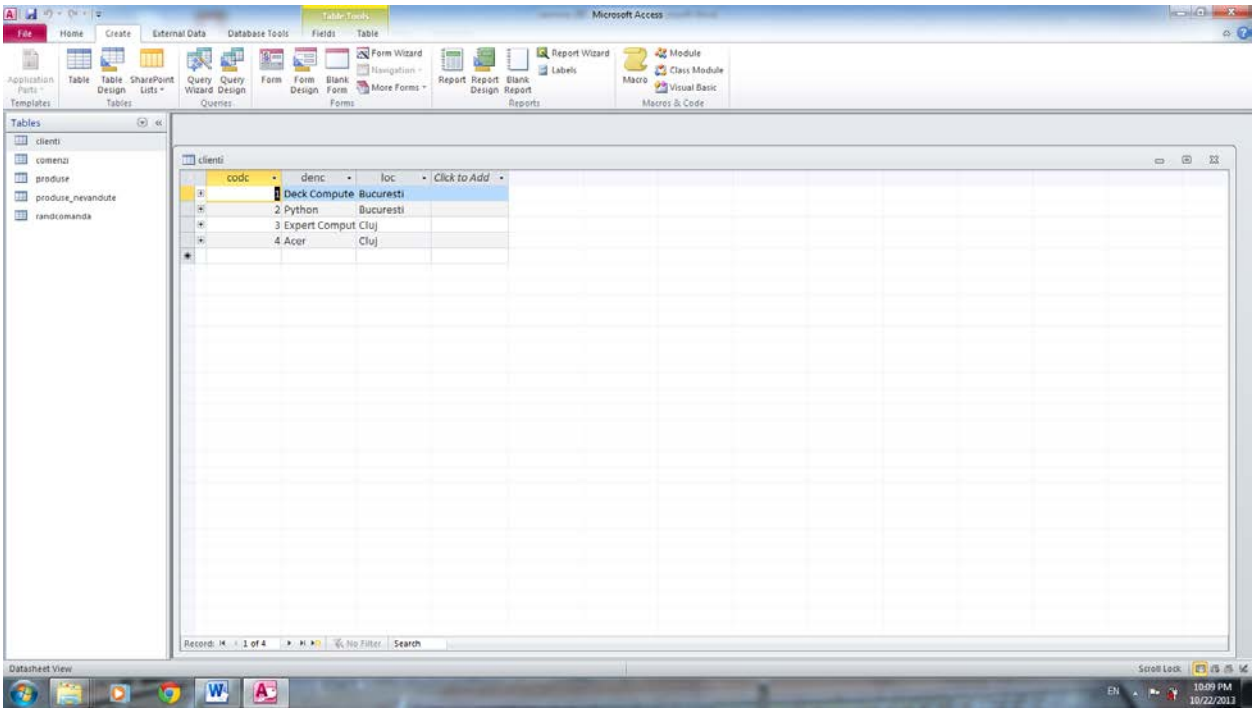
Create /Query design/se inchide fereastra Show tables, se activeaza icoana Delete. Apoi View/SQL View si scriem comanda SQL




Se executa cererea




Apoi se verifica daca s-a sters tuplul

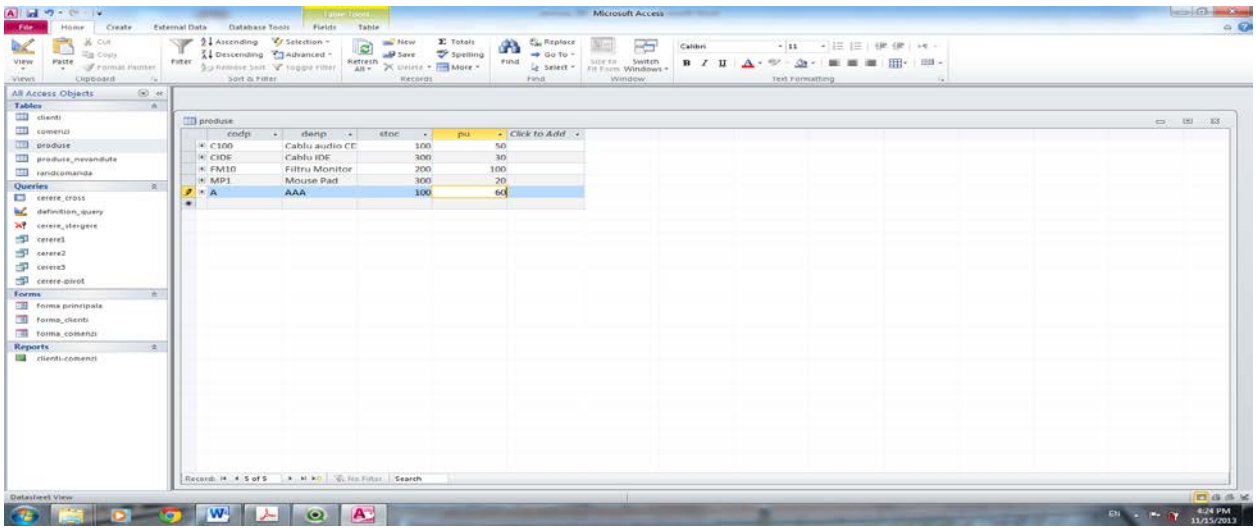



Studiu individual: celelalte exemple

15. Sa se afiseze produsele care nu se vand (realizarea unei subcerere/subquery).

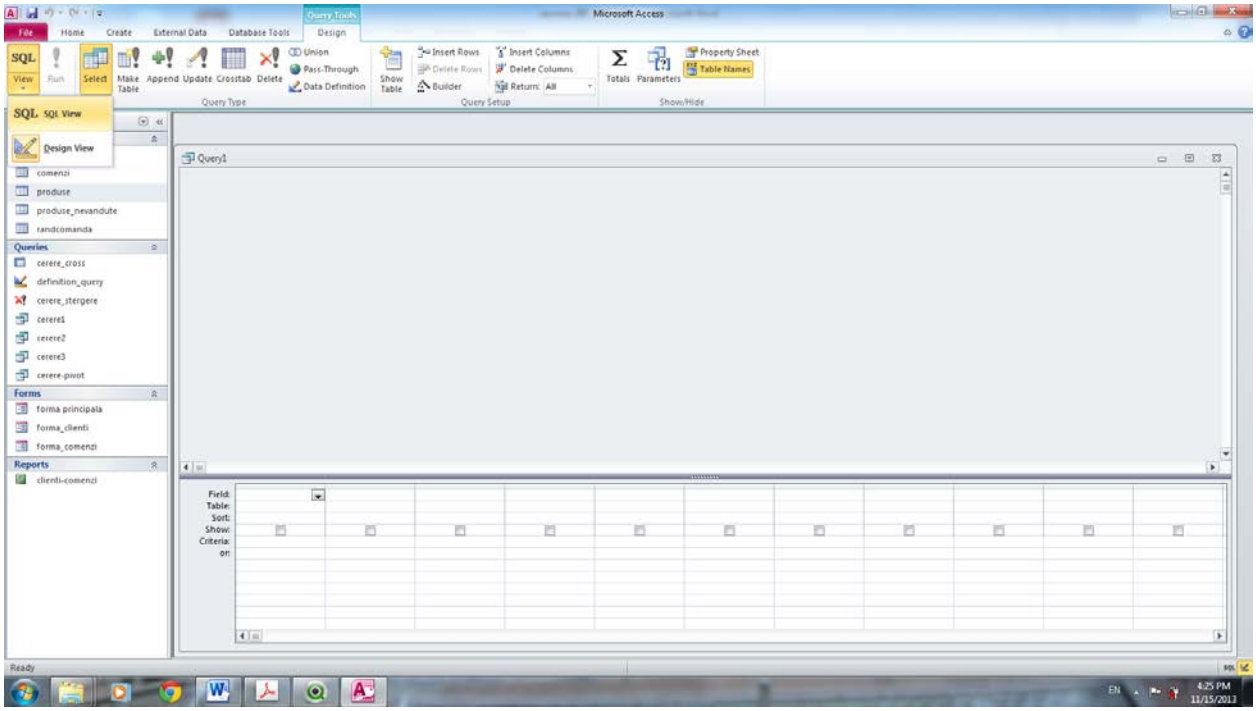
```
SELECT denp
FROM produse
where codp not in (select distinct codp from randcomanda);
```

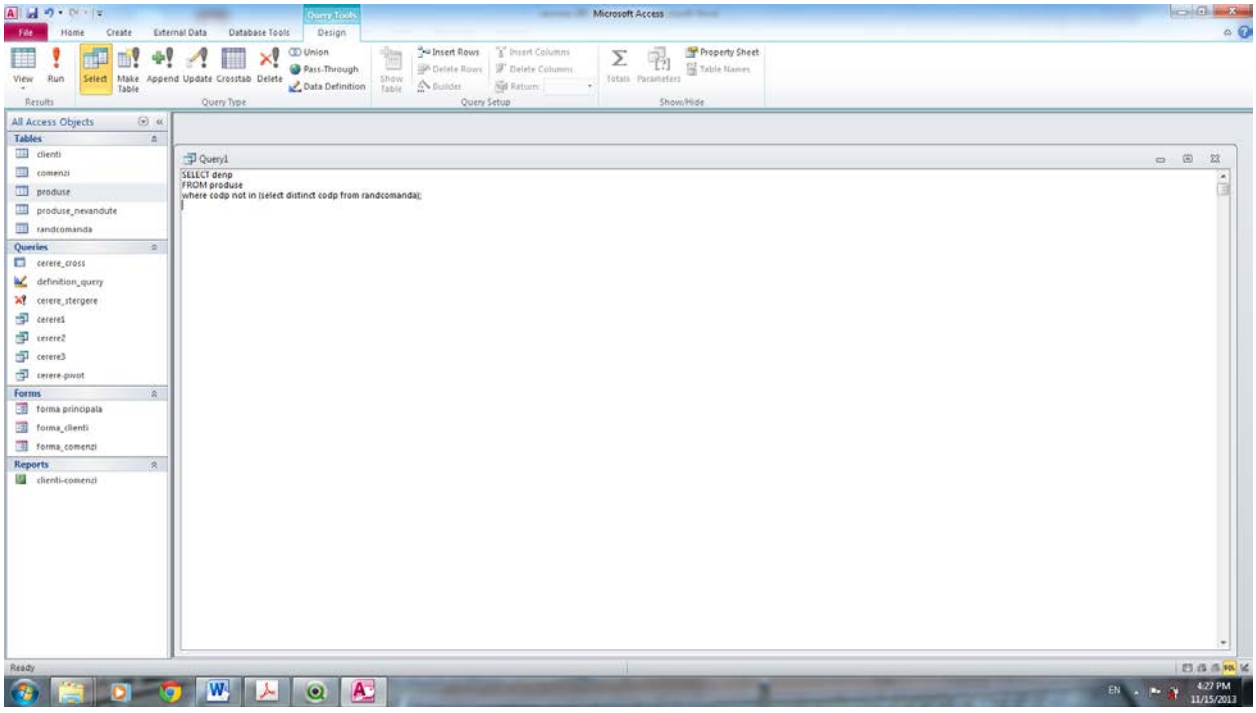
Adaugam un nou produs in tabela Produse



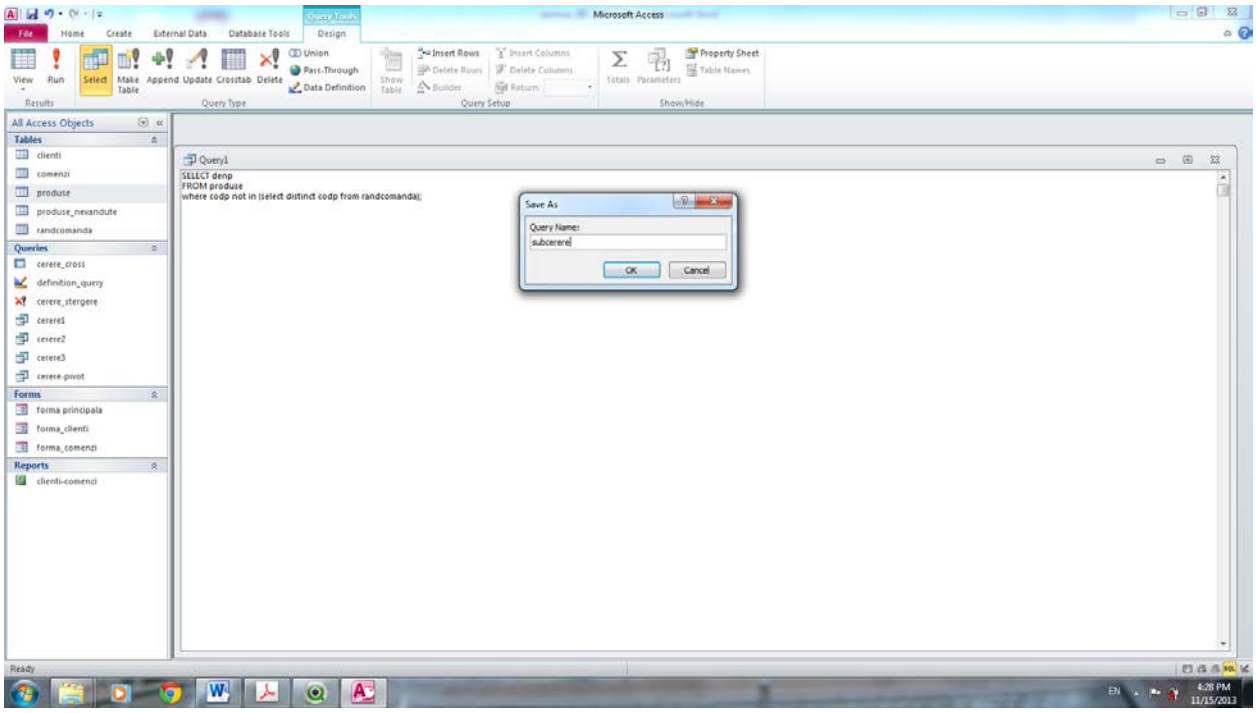

Apoi Create/Query Design, nu se selecteaza nici o tabela

View/SQL View si scriem codul SQL

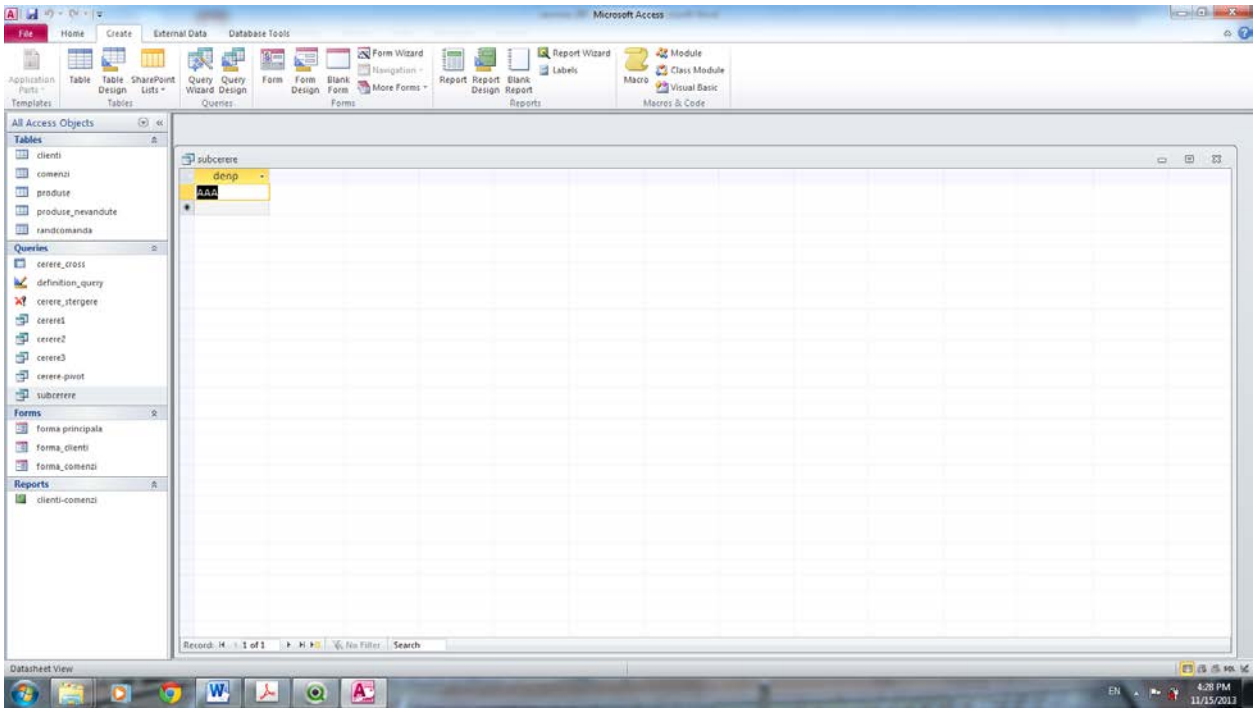



Se salveaza



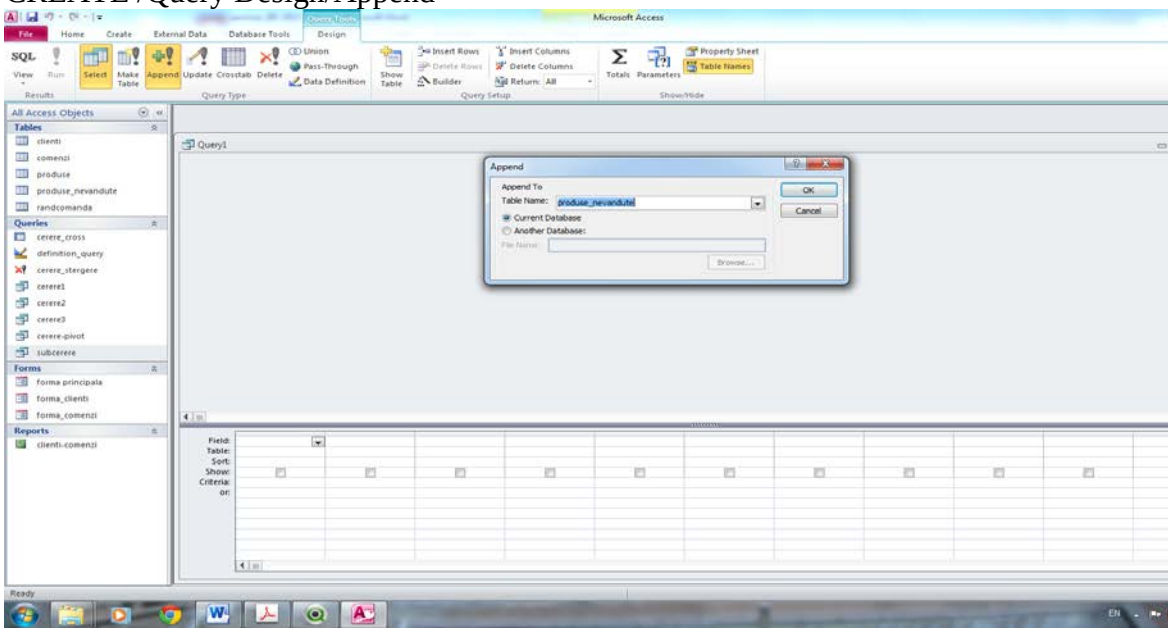

Apoi se executa




Se va afisa produsul care nu s-a vandut

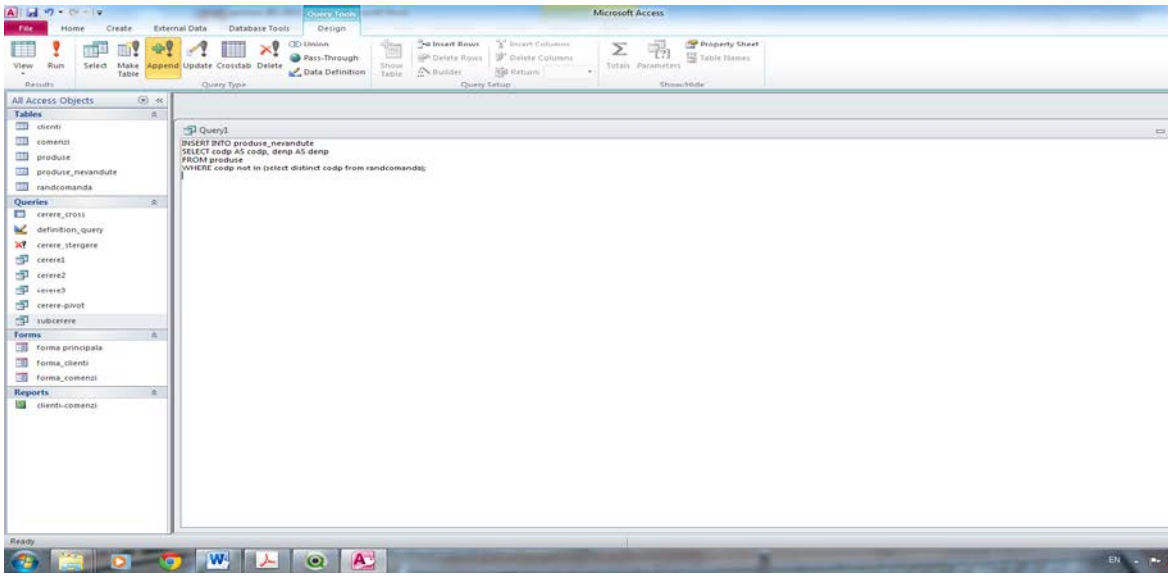
16. Sa se adauge in tabela **Produse_nevandute**, produsele care nu s-au vandut
Se va utiliza INSERT impreuna cu SELECT:

CREATE /Query Design/Append

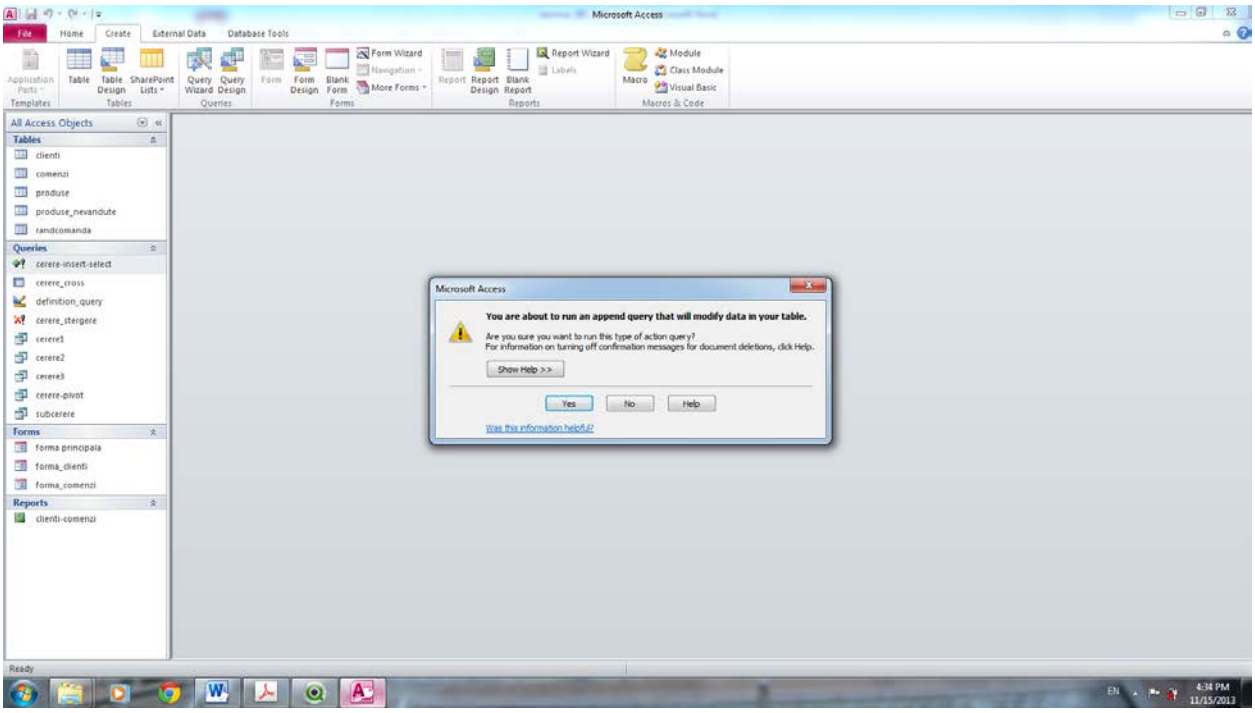


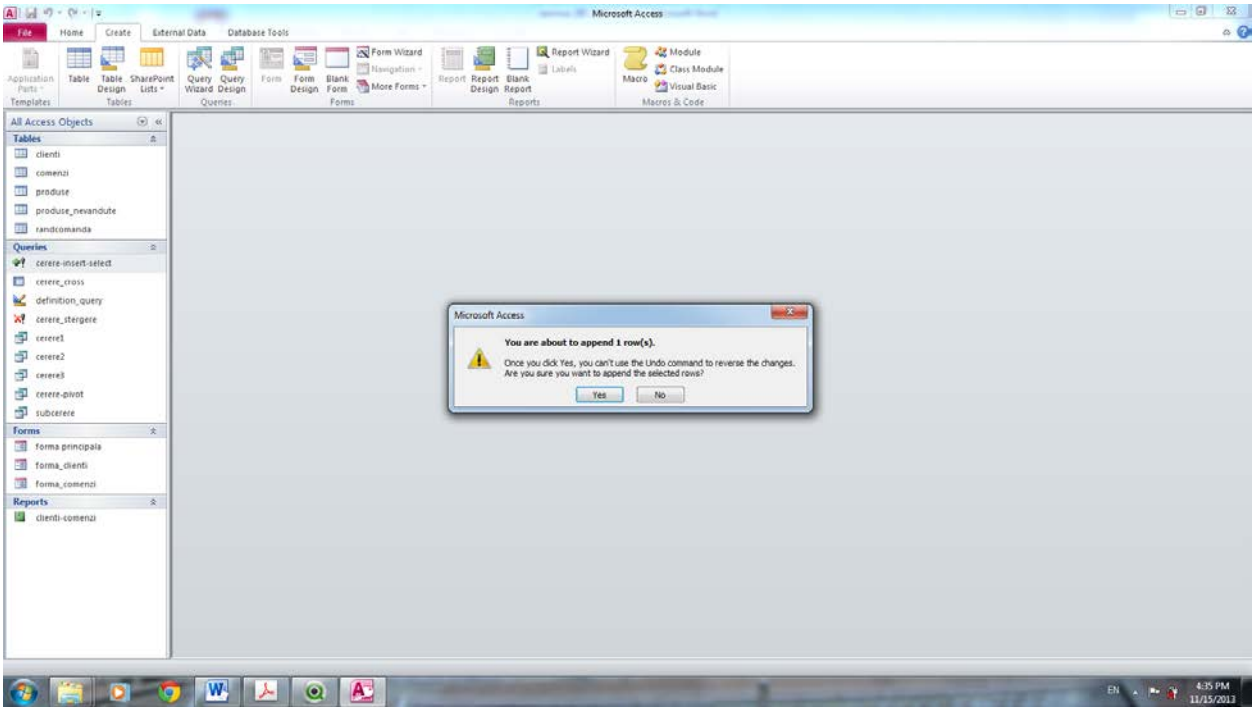

View/SQL View si se adauga codul SQL

```
INSERT INTO produse_nevandute
SELECT codp AS codp, denp AS denp
FROM produse
WHERE codp not in (select distinct codp from randcomanda);
```

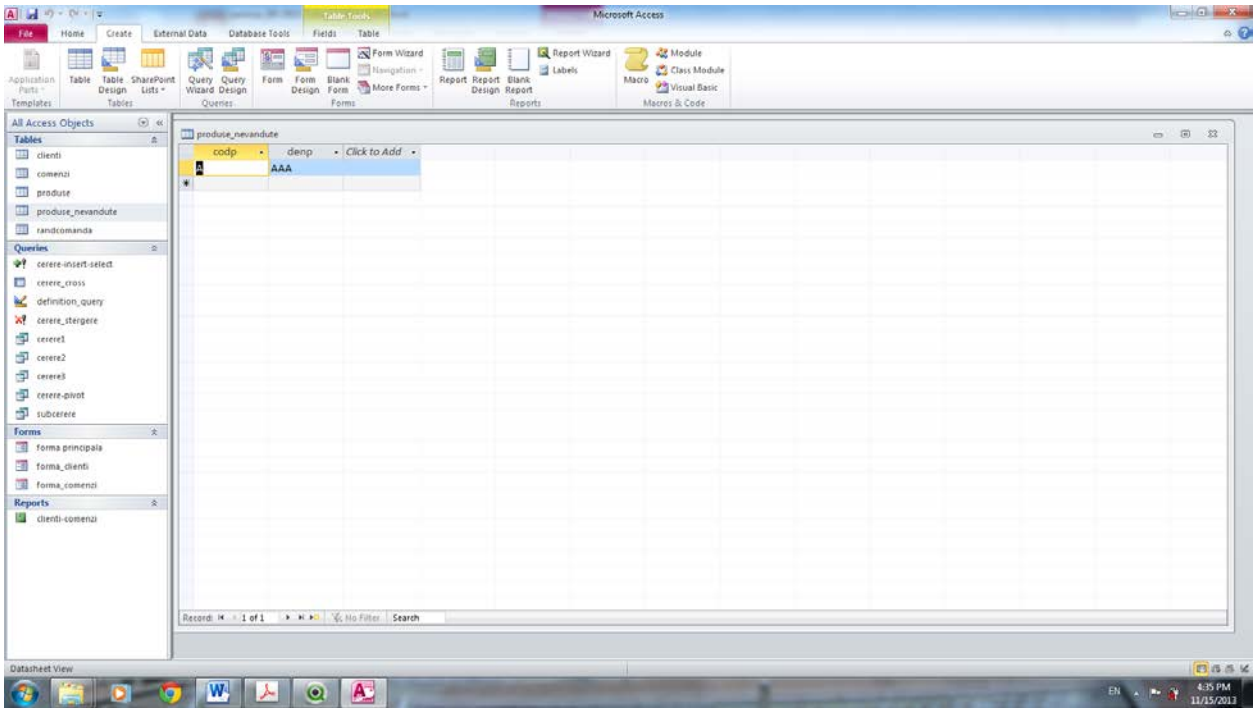



Se salveaza si apoi se executa cererea

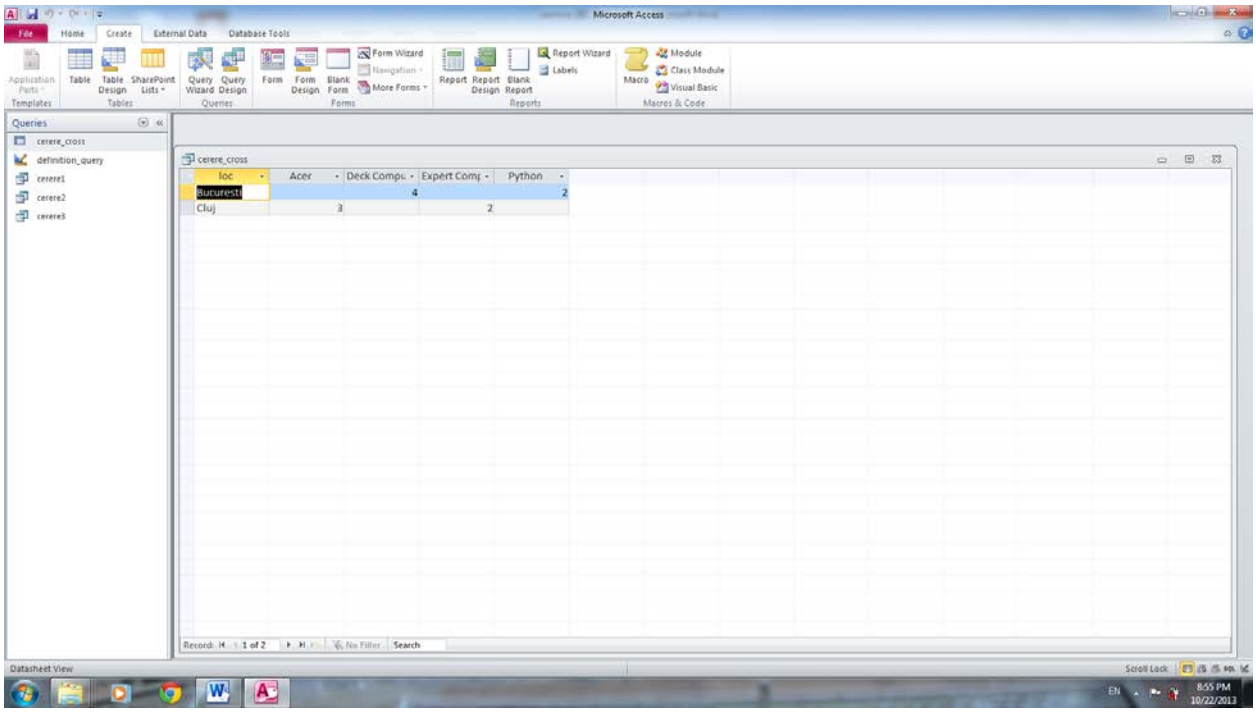



Se verifica in tabela **Produse_nevandute** daca s-a facut inserarea

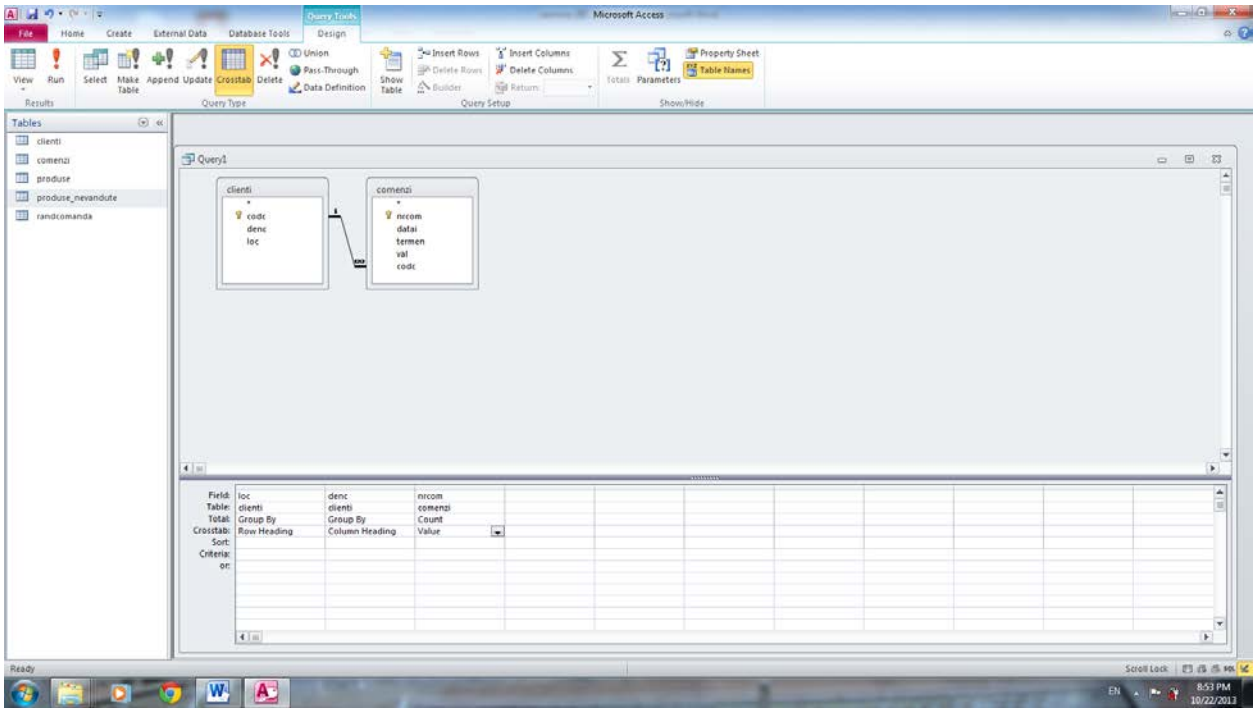



17. Sa se creeze o cerere de tip crosstab (matrice)




Se selecteaza icoana **Crosstab**

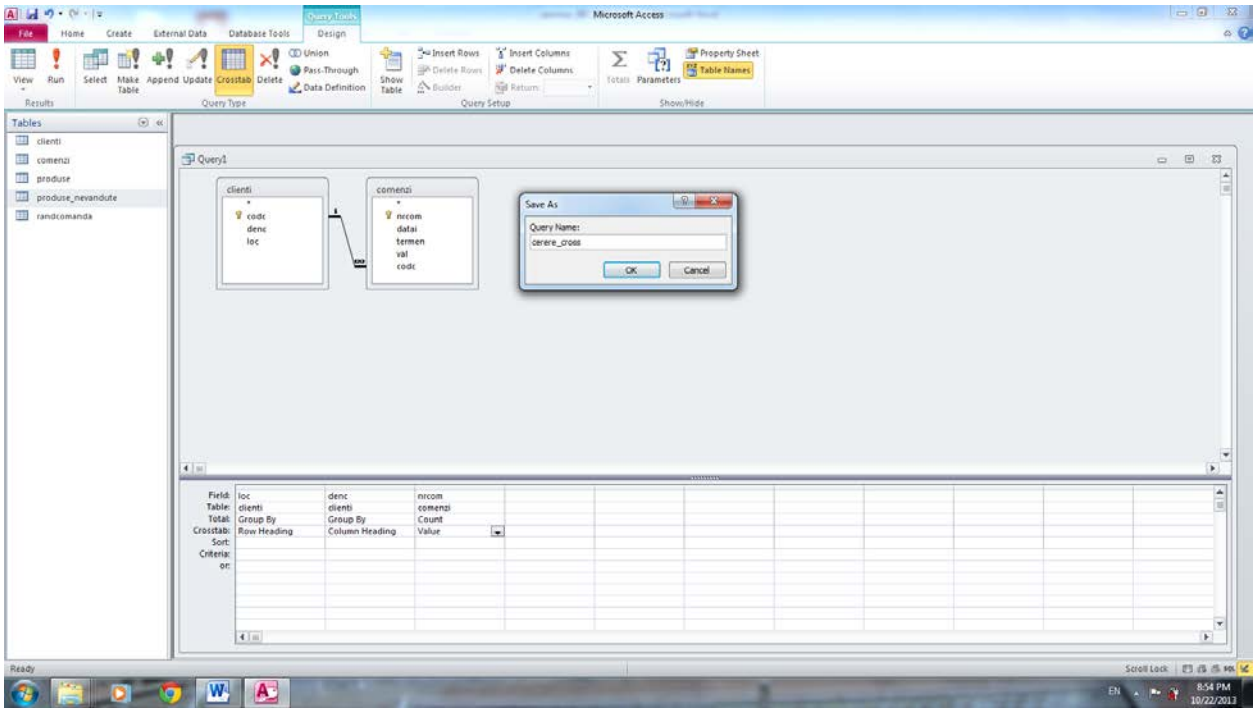
Se fac urm setari, conform fig de mai jos




In View/SQL View apare urm cod SQL creat

```
TRANSFORM Count(comenzi.nrcom) AS CountOfnrcom
SELECT clienti.loc
FROM clienti INNER JOIN comenzi ON clienti.codc = comenzi.codc
GROUP BY clienti.loc
PIVOT clienti.denc;
```

Se salveaza cererea:




Se executa cererea