Logo

Description automatically generated

ACADEMIA DE STUDII ECONOMICE BUCUREȘTI

FACULTATEA DE CIBERNETICĂ, STATISTICĂ ȘI INFORMATICĂ ECONOMICĂ

SPECIALIZAREA INFORMATICĂ ECONOMICĂ

*Pachete Software*

**Analiza performanțelor în vânzări a unui start-up**

**Coordonator științific**

Prof. univ. dr. ANDREESCU Anca Ioana

**Studenți**

Coman Claudia Ana-Maria, grupa 1082

Ciobanu Cristina-Maria, grupa 1082

MAI 2023 BUCUREȘTI

Cuprins

[**Capitolul 1 Programare Python** 3](#_Toc135550047)

[**Exerciții propuse seminarii Python** 19](#_Toc135550048)

[**Capitolul 2 Programare SAS** 32](#_Toc135550049)

[**Exerciții propuse seminarii SAS** 47](#_Toc135550050)

# **Capitolul 1 Programare Python**

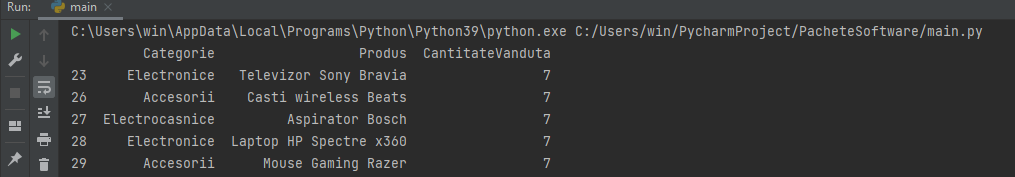
**EXERCIȚIUL 1.** Sa se citeasca din fisierul de tip CSV toate produsele cu cantitatea cea mai mare vanduta, sa se afiseze inclusiv categoria din care face parte, dar si denumirea produsului intr-un nou DataFrame

**Rezolvare:**

Am importat fisierul CSV cu ajutorului pachetului Pandas si al functiei ***read\_csv(),*** fisierul este intr-un DataFrame. Ulterior am selectat din DataFrame coloana „CantitateVanduta” si am apelat functia de max() pentru a gasi elementul maxim din coloana. Am facut filtrarea datelor pentru a obține produsele cu cantitatea maximă vândută si am selectat doar coloanele "Categorie" și "Produs" în noul DataFrame.

#EXERCITII PYTHON  
import pandas as pd  
  
  
#1. Sa se citeasca din csv toate produsele cu cantitatea cea mai mare vanduta, sa se afiseze inclusiv categoria din care face parte produsul, dar si produsul  
#intr-un nou DataFrame  
  
#citim datele din fisier  
  
# Citirea datelor din fișierul CSV într-un DataFrame  
df = pd.read\_csv('ProduseVandute.csv')  
  
# Găsirea maximului pentru coloana "CantitateVanduta"  
max\_cantitate = df['CantitateVanduta'].max()  
  
# Filtrarea datelor pentru a obține produsele cu cantitatea maximă vândută  
df\_max\_cantitate = df[df['CantitateVanduta'] == max\_cantitate]  
  
# Selectarea doar a coloanelor "Categorie" și "Produs" în noul DataFrame  
df\_result = df\_max\_cantitate[['Categorie', 'Produs',"CantitateVanduta"]]  
  
# Afișarea rezultatului  
print(df\_result)

**Output:**

****

**Interpretarea economica a rezultatelor obtinute:**

Din rândurile filtrate, observăm că există mai multe produse din diferite categorii (Electronice, Accesorii, Electrocasnice) care au fost vândute în cantitatea maximă de 7 unități. Interpretarea economică a acestui rezultat ar putea fi că aceste produse au fost foarte populare și că există o cerere mare pentru ele în rândul clienților. De asemenea, ar putea fi util să analizăm și alți factori, cum ar fi prețul acestor produse, pentru a înțelege mai bine motivele din spatele succesului lor în vânzări.

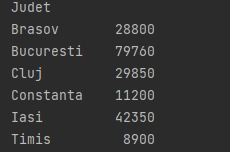
**EXERCIȚIUL 2.** Sa se determine valoarea totala a vanzarilor pentru fiecare judet.

**Rezolvare:**

Am creat o functie ce calculeaza valoarea totala a vanzarilor si primeste ca parametru DataFrame-ul cu datele din fisier. Valoarea totala este calculata ca produsul dintre cantitatea vanduta si pretul unitar al unui produs si este salvat intr-o noua coloana „ValoareTotala”, ulterior grupam aceasta coloana dupa Judet, facem suma pe coloana obtinuta si returnam rezultatul.

#2. Sa se determine valoarea totala a vanzarilor pentru fiecare judet  
  
  
def valTotala(df):  
 df['ValoareTotala'] = df['CantitateVanduta'] \* df['PretUnitarPerReducere']  
 valoare\_totala\_judet = df.groupby('Judet')['ValoareTotala'].sum()  
 return valoare\_totala\_judet  
  
# Apelarea funcției valTotala pentru a calcula valoarea totală a vânzărilor pentru fiecare județ  
rezultat = valTotala(df)  
  
# Afișarea valorii totale a vânzărilor pentru fiecare județ  
print(rezultat)

**Output:**

****

**Interpretarea economica a rezultatelor obtinute:**

1. Bucureștiul are cea mai mare valoare totală a vânzărilor, indicând că este cel mai important județ din punct de vedere al volumului de vânzări.
2. Iași și Cluj au și ele valori ridicate ale vânzărilor, sugerând o activitate comercială semnificativă în aceste județe.
3. Brasov și Constanta au valori totale mai mici, ceea ce poate indica un nivel de vânzări mai scăzut comparativ cu celelalte județe.
4. Timiș înregistrează cea mai mică valoare totală a vânzărilor, sugerând un volum mai mic de tranzacții comerciale în această zonă.

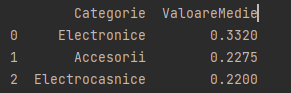
**EXERCIȚIUL 3**. Calcularea valorii medii a taxei pentru fiecare categorie de produse.

**Rezolvare:**

Am adaugat o noua coloana in DataFrame-ul existent denumita „ValoareMedie”, aceasta va contine valoarea medie a taxelor pentru fiecare categorie de produse. Ulterior am utilizat metoda groupby pentru a grupa rândurile DataFrame-ului după coloana 'Categorie'. Apoi, se aplică funcția transform('mean') asupra coloanei 'ValoareTaxe' pentru a calcula valoarea medie a taxelor pentru fiecare categorie. Se selectează doar coloanele 'Categorie' și 'ValoareMedie' din DataFrame-ul original și apoi se elimină rândurile duplicate utilizând metoda drop\_duplicates(). Apelând reset\_index(drop=True), se resetează indexul DataFrame-ului rezultat.

#3. Calcularea valorii medii a taxei pentru fiecare categorie de produse.  
  
# Calcularea valorii medii a taxei pentru fiecare categorie de produse  
df['ValoareMedie'] = df.groupby('Categorie')['ValoareTaxe'].transform('mean')  
  
# Selecția unice a categoriilor de produse și a valorilor medii ale taxei  
valoare\_medie\_taxa = df[['Categorie', 'ValoareMedie']].drop\_duplicates().reset\_index(drop=True)  
  
# Afișarea valorii medii a taxei pentru fiecare categorie de produse  
print(valoare\_medie\_taxa)

**Output:**



**Interpretarea economica a rezultatelor obtinute:**

* Categoriile de produse din domeniul electronicii înregistrează o valoare medie a taxelor mai ridicată (0.3320). Aceasta poate indica faptul că aceste produse pot fi supuse unor taxe mai mari sau pot avea o valoare mai mare adăugată, ceea ce duce la o taxă mai mare. Aceasta poate include produse precum televizoare, laptopuri sau telefoane mobile.
* Accesoriile, care includ produse precum căști, mouse-uri sau tastaturi, înregistrează o valoare medie a taxelor mai mică (0.2275) în comparație cu categoria electronice. Acest lucru poate indica faptul că accesoriile pot avea o valoare adăugată mai mică sau pot fi supuse unor taxe mai mici.
* Categoria electrocasnicelor, care poate include produse precum aspiratoare, cuptoare sau mașini de spălat, înregistrează o valoare medie a taxelor de 0.2200. Aceasta poate indica faptul că aceste produse sunt supuse unor taxe relativ constante sau au o valoare adăugată similară.

**EXERCIȚIUL 4**. Găsirea orașului cu cele mai multe vânzări de produse electrocasnice.

**Rezolvare:**

Am definit o functie care primeste ca parametru DataFrame-ul. Pentru inceput,filtrez datele pentru a include doar produsele din categoria „Electrocasnice”, tot ce fac este sa realizez o selecție a rândurilor din DataFrame-ul care au valoarea 'Electrocasnice' în coloana „Categorie”.

Utilizez DataFrame-ul „electrocasnice” pentru a grupa rândurile după coloana „Oras”. Apoi, se calculează suma cantităților vândute pentru fiecare oraș utilizând coloana „CantitateVanduta”. Cu ajutorul metodei **.idxmax()**, se obține orașul cu suma maximă a cantităților vândute.

#4. Găsirea orașului cu cele mai multe vânzări de produse electrocasnice.  
  
def gaseste\_orasul\_max\_vanzari\_electrocasnice(df):  
 # Filtrarea datelor pentru a include doar produsele de categorie "Electrocasnice"  
 electrocasnice = df[df['Categorie'] == 'Electrocasnice']  
  
 # Găsirea orașului cu cele mai multe vânzări de produse electrocasnice  
 oras\_max\_vanzari = electrocasnice.groupby('Oras')['CantitateVanduta'].sum().idxmax()  
  
 return oras\_max\_vanzari  
  
# Exemplu de utilizare  
oras\_max\_vanzari\_electrocasnice = gaseste\_orasul\_max\_vanzari\_electrocasnice(df)  
print("Orașul cu cele mai multe vânzări de produse electrocasnice:", oras\_max\_vanzari\_electrocasnice)

**Output:**

****

**Interpretarea economica a rezultatelor obtinute:**

Din rezultatele obținute, orașul cu cele mai multe vânzări de produse electrocasnice este Constanța. Putem spune că în acest oraș s-au vândut cantități mai mari de produse electrocasnice în comparație cu celelalte orașe în cadrul setului de date.

Interpretarea economică a acestui rezultat poate fi următoarea:

1. Constanța poate avea o piață puternică pentru produsele electrocasnice, cu cerere mare din partea consumatorilor din acest oraș. Acest lucru poate fi influențat de factori cum ar fi nivelul de trai, demografia, preferințele locale sau disponibilitatea de magazine și centre comerciale specializate în electrocasnice.
2. Orașul Constanța beneficiază și de fluxuri mari de turiști, ceea ce poate contribui la o cerere crescută pentru produsele electrocasnice, cum ar fi electrocasnicele de uz casnic, din partea turiștilor care călătoresc în zonă.

**EXERCIȚIUL 5**. Sa se afișeze un mesaj corespunzător având în vedere că se dorește afișarea majorării cu 25% a prețului de vânzare cu taxe pentru categoria de „Electronice” și diminuarea cu 5% a prețului de vânzare cu taxe pentru categoria de “Accesorii” împreună cu județul și data când a fost făcută tranzacția.

**Rezolvare:**

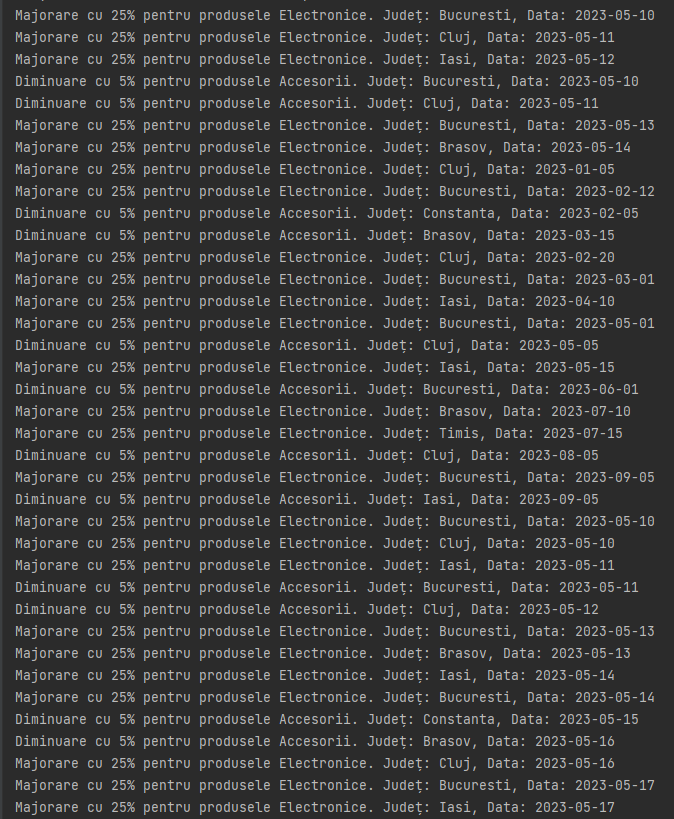
Funcția afisare\_mesaj primește un DataFrame df ca argument și aplică anumite modificări asupra prețurilor de vânzare cu taxe pentru produsele din categoriile "Electronice" și "Accesorii".

1. Majorarea cu 25% a prețului de vânzare cu taxe pentru produsele din categoria "Electronice". Am utilizat metoda **loc**, si se selectează rândurile din DataFrame unde coloana 'Categorie' are valoarea 'Electronice', iar coloana 'PretUnitarVanzareFaraTaxe' este înmulțită cu 1.25, ceea ce corespunde unei majorări cu 25% a prețului.
2. Diminuarea cu 5% a prețului de vânzare cu taxe pentru produsele din categoria "Accesorii". Am utilizat metoda **loc**, se selectează rândurile din DataFrame unde coloana 'Categorie' are valoarea 'Accesorii', iar coloana 'PretUnitarVanzareFaraTaxe' este înmulțită cu 0.95, ceea ce corespunde unei diminuări cu 5% a prețului.

Ulterior se parcurge fiecare rând din DataFrame folosind metoda **iterrows()**. Pentru fiecare rând, se verifică categoria produsului și se construiește un mesaj corespunzător care conține informații despre județ și dată. Mesajul este apoi afișat pe ecran.

#5. Sa se afișeze un mesaj corespunzător având în vedere că se dorește afișarea majorării cu 25% a prețului de vânzare cu taxe pentru categoria de „Electronice”  
# și diminuarea cu 5% a prețului de vânzare cu taxe pentru categoria de “Accesorii” împreună cu județul și data când a fost făcută tranzacția.  
  
def afisare\_mesaj(df):  
 # Majorarea cu 25% a prețului de vânzare cu taxe pentru Electronice  
 df.loc[df['Categorie'] == 'Electronice', 'PretUnitarVanzareFaraTaxe'] \*= 1.25  
  
 # Diminuarea cu 5% a prețului de vânzare cu taxe pentru Accesorii  
 df.loc[df['Categorie'] == 'Accesorii', 'PretUnitarVanzareFaraTaxe'] \*= 0.95  
  
 # Afișarea mesajului corespunzător pentru fiecare tranzacție modificată  
 for index, row in df.iterrows():  
 if row['Categorie'] == 'Electronice':  
 mesaj = f"Majorare cu 25% pentru produsele Electronice. Județ: {row['Judet']}, Data: {row['DataComandarii']}"  
 print(mesaj)  
 elif row['Categorie'] == 'Accesorii':  
 mesaj = f"Diminuare cu 5% pentru produsele Accesorii. Județ: {row['Judet']}, Data: {row['DataComandarii']}"  
 print(mesaj)  
  
print(afisare\_mesaj(df))

**Output:**

****

**Interpretarea economica a rezultatelor obtinute:**

Rezultatul afișat este lista mesajelor care corespund modificărilor făcute asupra prețurilor de vânzare pentru produsele din categoriile "Electronice" și "Accesorii". Fiecare mesaj indică tipul modificării (majorare sau diminuare), categoria produsului, județul în care a avut loc tranzacția și data acesteia.

**EXERCIȚIUL 6**. Să se afișeze toate vanzarile din județul Bucuresti unde produsele vandute au fost „Telefon Samsung Galaxy S20”.

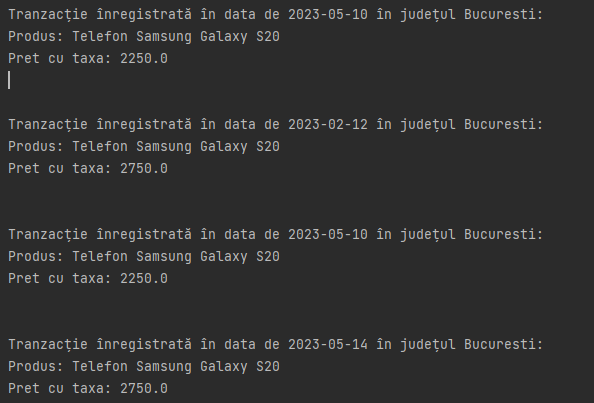
**Rezolvare:**

1. Se definește un filtru pentru a selecta înregistrările cu județul "Bucuresti" și produsul "Telefon Samsung Galaxy S20".
2. Se aplică filtrul asupra datelor pentru a obține o nouă structură de date care conține doar coloanele "DataComandarii", "Judet", "Produs" și "PretUnitarVanzareFaraTaxe" pentru înregistrările care satisfac condițiile de filtrare.
3. Se parcurge fiecare înregistrare din structura de date filtrată și se extrag informațiile relevante: data comenzii, județul, produsul și prețul unitar de vânzare fără taxe.
4. Pentru fiecare înregistrare, se afișează un mesaj care include data comenzii, județul, produsul și prețul unitar de vânzare fără taxe.

Astfel, codul afișează informații specifice despre tranzacțiile cu produsul "Telefon Samsung Galaxy S20" în județul "Bucuresti", inclusiv data comenzii și prețul unitar de vânzare fără taxe pentru fiecare tranzacție.

#6. Să se afișeze toate vanzarile din județul Bucuresti unde produsele vandute au fost „Telefon Samsung Galaxy S20”.  
  
# Filtrarea datelor pentru județul București și produsul "Telefon Samsung Galaxy S20"  
filt = (df['Judet'] == 'Bucuresti') & (df['Produs'] == 'Telefon Samsung Galaxy S20')  
filtered\_data = df.loc[filt, ['DataComandarii', 'Judet', 'Produs', 'PretUnitarVanzareFaraTaxe']]  
  
# Afișarea mesajului corespunzător pentru fiecare înregistrare  
for \_, row in filtered\_data.iterrows():  
 data\_comanda = row['DataComandarii']  
 judet = row['Judet']  
 produs = row['Produs']  
 pret = row['PretUnitarVanzareFaraTaxe']  
 print(f"Tranzacție înregistrată în data de {data\_comanda} în județul {judet}:")  
 print(f"Produs: {produs}")  
 print(f"Pret cu taxa: {pret}")  
 print("\n")

**Output:**

****

**Interpretarea economica a rezultatelor obtinute:**

Conform output-ului prezentat putem observa ca in Bucuresti au fost facute 4 tranzactii pentru achizitionarea telefonului „Samsung Galaxy S20”. De asemenea este important de observat faptul ca pretul telefonului difera, acesta poate fi influentat de mai multi factori. In primul rand, nu cunoastem locul de unde a fost achizitionat acest produs, spre exemplu, daca a fost cumparat dintr-un magazin specializat sau de pe Internet, pretul acestuia difera in functie de anumite taxe, durata de utilizare, etc.

In al doilea rand, pretul acestui produs poate sa difere si in functie de perioada in care a fost achizitionat, spre exemplu, pretul poate sa fie mai ridicat in momentul lansarii comparativ cu o perioada mai mare de la aparitia acestuia.

**EXERCIȚIUL 7**. Să se afișeze pentru fiecare județ:  
- cantitatea totală vândută pe categorii de produse;  
- prețul total mediu de vânzare.

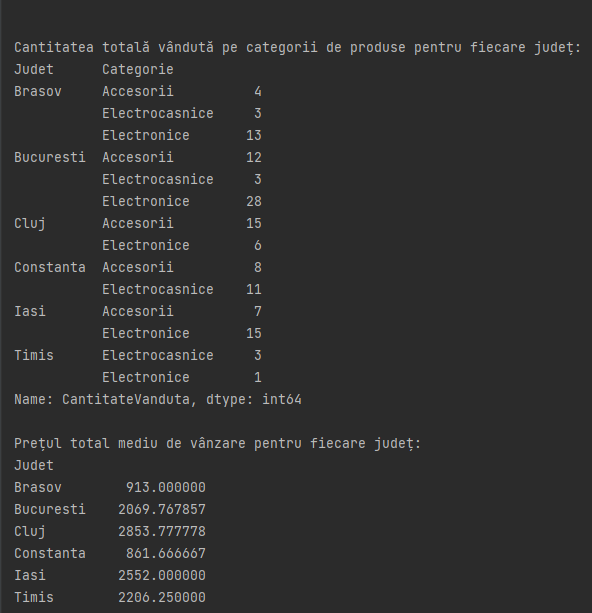
**Rezolvare:**

Pentru a calcula cantitatea totala vanduta pe fiecare categorie de produse se grupează datele după coloanele "Judet" și "Categorie", iar apoi se calculează suma cantității vândute pentru fiecare grup. Astfel, rezultatul reprezintă cantitatea totală vândută pe categorii de produse pentru fiecare județ.

Pentru a calcula pretul mediu se grupează datele doar după coloana "Judet" și se calculează media valorilor din coloana "PretUnitarVanzareFaraTaxe" pentru fiecare județ. Astfel, rezultatul reprezintă prețul total mediu de vânzare pentru fiecare județ.

#7. Să se afișeze pentru fiecare județ:  
#- cantitatea totală vândută pe categorii de produse;  
#- prețul total mediu de vânzare;  
  
cantitate\_totala = df.groupby(['Judet', 'Categorie'])['CantitateVanduta'].sum()  
  
# Calcularea prețului total mediu de vânzare pentru fiecare județ  
pret\_mediu = df.groupby('Judet')['PretUnitarVanzareFaraTaxe'].mean()  
  
# Afișarea rezultatelor  
print("Cantitatea totală vândută pe categorii de produse pentru fiecare județ:")  
print(cantitate\_totala)  
print()  
print("Prețul total mediu de vânzare pentru fiecare județ:")  
print(pret\_mediu)

**Output:**



**Interpretarea economica a rezultatelor obtinute:**

1. Cantitatea totală vândută pe categorii de produse pentru fiecare județ:
   * În județul Brasov s-au vândut în total: 4 produse din categoria Accesorii, 3 produse din categoria Electrocasnice și 13 produse din categoria Electronice.
   * În județul Bucuresti s-au vândut în total: 12 produse din categoria Accesorii, 3 produse din categoria Electrocasnice și 28 produse din categoria Electronice.
   * În județul Cluj s-au vândut în total: 15 produse din categoria Accesorii și 6 produse din categoria Electronice.
   * În județul Constanta s-au vândut în total: 8 produse din categoria Accesorii și 11 produse din categoria Electrocasnice.
   * În județul Iasi s-au vândut în total: 7 produse din categoria Accesorii și 15 produse din categoria Electronice.
   * În județul Timis s-au vândut în total: 3 produse din categoria Electrocasnice și 1 produs din categoria Electronice.
2. Prețul total mediu de vânzare pentru fiecare județ:
   * În județul Brasov, prețul total mediu de vânzare este de aproximativ 913 unități monetare.
   * În județul Bucuresti, prețul total mediu de vânzare este de aproximativ 2069.77 unități monetare.
   * În județul Cluj, prețul total mediu de vânzare este de aproximativ 2853.78 unități monetare.
   * În județul Constanta, prețul total mediu de vânzare este de aproximativ 861.67 unități monetare.
   * În județul Iasi, prețul total mediu de vânzare este de aproximativ 2552 unități monetare.
   * În județul Timis, prețul total mediu de vânzare este de aproximativ 2206.25 unități monetare.

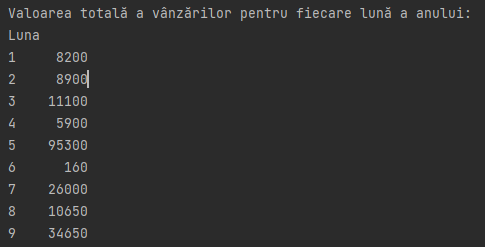
**EXERCIȚIUL 8**. Calcularea valorii totale a vânzărilor pentru fiecare lună a anului.

**Rezolvare:**

1. Convertim coloana "DataComandarii" la tipul dată utilizând funcția pd.to\_datetime(). Acest lucru ne permite să lucrăm cu datele în formatul corespunzător.
2. Adăugăm o nouă coloană numită "Luna" care conține numărul lunii pentru fiecare înregistrare din coloana "DataComandarii". Utilizăm metoda dt.month pentru a extrage numărul lunii.
3. Calculăm valoarea totală a vânzărilor pentru fiecare lună, grupând dataframe-ul după coloana "Luna" și apoi aplicând funcția sum() pe coloana "ValoareTotala".
4. Afișăm rezultatele, afișând valoarea totală a vânzărilor pentru fiecare lună a anului.

#8. Calcularea valorii totale a vânzărilor pentru fiecare lună a anului.  
  
  
# Convertim coloana "DataComandarii" la tipul dată  
df['DataComandarii'] = pd.to\_datetime(df['DataComandarii'])  
  
# Adăugăm o coloană pentru luna comenzii  
df['Luna'] = df['DataComandarii'].dt.month  
  
# Calculăm valoarea totală a vânzărilor pentru fiecare lună  
sales\_total\_per\_month = df.groupby('Luna')['ValoareTotala'].sum()  
  
# Afișăm rezultatele  
print("Valoarea totală a vânzărilor pentru fiecare lună a anului:")  
print(sales\_total\_per\_month)

**Output:**



**Interpretarea economica a rezultatelor obtinute:**

* În luna ianuarie (Luna 1) s-au înregistrat vânzări în valoare totală de 8200.
* În luna februarie (Luna 2) s-au înregistrat vânzări în valoare totală de 8900.
* În luna martie (Luna 3) s-au înregistrat vânzări în valoare totală de 11100.
* În luna aprilie (Luna 4) s-au înregistrat vânzări în valoare totală de 5900.
* În luna mai (Luna 5) s-au înregistrat vânzări în valoare totală de 95300.
* În luna iunie (Luna 6) s-au înregistrat vânzări în valoare totală de 160.
* În luna iulie (Luna 7) s-au înregistrat vânzări în valoare totală de 26000.
* În luna august (Luna 8) s-au înregistrat vânzări în valoare totală de 10650.
* În luna septembrie (Luna 9) s-au înregistrat vânzări în valoare totală de 34650.

**EXERCIȚIUL 9**. Găsirea produsului cu cea mai mare medie a prețului de vânzare pe unitate.

**Rezolvare:**

Prin utilizarea **df.groupby('Produs')**, datele sunt grupate pe baza coloanei "Produs", iar apoi se calculează media valorilor din coloana "PretUnitarVanzareFaraTaxe" pentru fiecare grup, folosind metoda **.mean()**. Acest lucru rezultă într-un obiect de tip Series care conține mediile prețurilor de vânzare pentru fiecare produs.

Apelând **.idxmax()** pe obiectul Series, se identifică indicele (produsul) asociat cu valoarea maximă. Aceasta reprezintă produsul cu cea mai mare medie a prețului de vânzare pe unitate.

#9. Găsirea produsului cu cea mai mare medie a prețului de vânzare pe unitate:  
  
max\_avg\_price\_product = df.groupby('Produs')['PretUnitarVanzareFaraTaxe'].mean().idxmax()  
print("Produsul cu cea mai mare medie a prețului de vânzare pe unitate:", max\_avg\_price\_product)

**Output:**



**Interpretarea economica a rezultatelor obtinute:**

Acest lucru înseamnă că, în medie, prețul de vânzare al acestui produs este cel mai ridicat comparativ cu celelalte produse din setul de date analizat.

**EXERCIȚIUL 10.** Reprezentați grafic cantitățile vândute din fiecare categorie.

**Rezolvare:**

Pasul 1: Calculul sumei cantităților vândute pe categorii Folosind funcția **groupby** pe coloana "Categorie" și apoi aplicând metoda **sum()** pe coloana "CantitateVanduta", se calculează suma cantităților vândute pentru fiecare categorie. Rezultatul este stocat în variabila **category\_sales**.

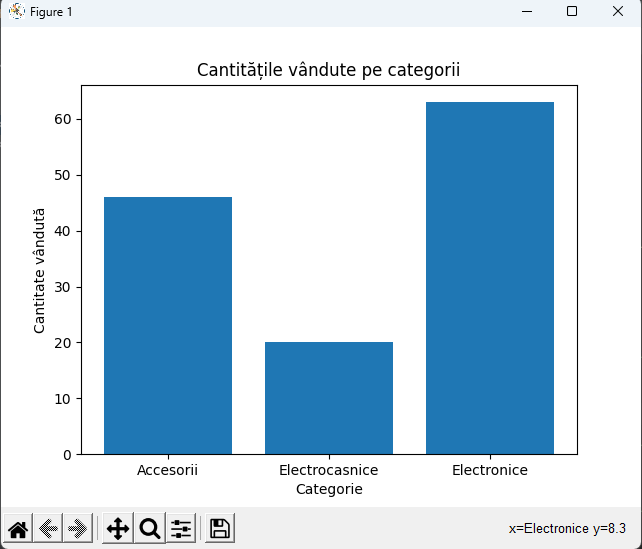
Pasul 2: Crearea graficului de bare Folosind funcția **plt.bar()**, se construiește graficul de bare. Se furnizează ca argumente lista cu numele categoriilor (**category\_sales.index**) și lista cu valorile sumelor cantităților vândute (**category\_sales.values**).

Pasul 3: Setarea titlului și etichetelor axelor Folosind funcțiile **plt.title()**, **plt.xlabel()** și **plt.ylabel()**, se setează titlul graficului și etichetele axelor x și y, respectiv.

Pasul 4: Afișarea graficului Folosind funcția **plt.show()**, se afișează graficul de bare.

#9. Găsirea produsului cu cea mai mare medie a prețului de vânzare pe unitate:  
  
max\_avg\_price\_product = df.groupby('Produs')['PretUnitarVanzareFaraTaxe'].mean().idxmax()  
print("Produsul cu cea mai mare medie a prețului de vânzare pe unitate:", max\_avg\_price\_product)  
  
#10. Reprezentați grafic cantitățile vândute din fiecare categorie  
  
  
# Calcularea sumei cantităților vândute pe categorii  
category\_sales = df.groupby('Categorie')['CantitateVanduta'].sum()  
  
# Crearea graficului de bare  
plt.bar(category\_sales.index, category\_sales.values)  
  
# Setarea titlului și etichetelor axelor  
plt.title('Cantitățile vândute pe categorii')  
plt.xlabel('Categorie')  
plt.ylabel('Cantitate vândută')  
  
# Afișarea graficului  
plt.show()

**Output:**



**Interpretarea economica a rezultatelor obtinute:**

Conform graficului, cantitatea vanduta de produse din categoria de „Accesorii” este de 48. Cantitatea vanduta de produse din categoria de „Electrocasnice” este de 20. Cantitatea vanduta de produse din categoria de „Electronice” este de 60.

# **Exerciții propuse seminarii Python**

**Exercițiul 0.a:** Sortați lista folosind list.sort() și sorted(); Cum se poate schimba direcția de sortare (descrescător)?

lista = ["laptop", "creion", "flipchart"] lista.sort()

print(lista) # Output: ['creion', 'flipchart', 'laptop']

lista = ["laptop", "creion", "flipchart"] sorted\_lista = sorted(lista)

print(sorted\_lista) # Output: ['creion', 'flipchart', 'laptop']

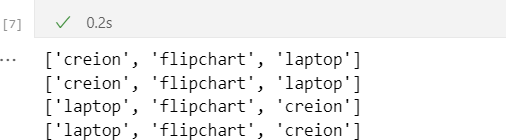
lista = ["laptop", "creion", "flipchart"] lista.sort(reverse=True)

print(lista) # Output: ['laptop', 'flipchart', 'creion']

lista = ["laptop", "creion", "flipchart"] sorted\_lista = sorted(lista, reverse=True)

print(sorted\_lista) # Output: ['laptop', 'flipchart', 'creion']

**OUTPUT:**



**Exercițiul 0.b:** creați obiecte set și utilizați câteva metode, la alegere

# Crearea unui set

fructe = {"măr", "portocală", "banană", "prună"}

# Adăugarea unui element în set folosind metoda add() fructe.add("piersică")

print(fructe) # Output: {'portocală', 'banană', 'măr', 'prună', 'piersică'}

# Eliminarea unui element din set folosind metoda remove() fructe.remove("măr")

print(fructe) # Output: {'portocală', 'banană', 'prună', 'piersică'}

# Verificarea apartenenței unui element în set folosind operatorul 'in' if "portocală" in fructe:

print("Portocala este în set.") else:

print("Portocala nu este în set.")

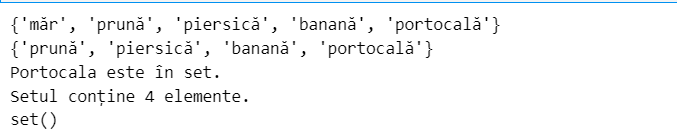
# Determinarea lungimii setului folosind funcția len() lungime = len(fructe)

print("Setul conține", lungime, "elemente.")

# Ștergerea tuturor elementelor din set folosind metoda clear() fructe.clear()

print(fructe) # Output: set()

**OUTPUT:**



În acest exemplu, am creat un set numit fructe care conține câteva elemente. Apoi am implementat câteva metode comune folosite în seturi, cum ar fi add(), remove(), in, len(), și clear().

**Exercițiul 0.c:** creați un dicționar și utilizați câteva metode, la alegere.

# Crearea unui dicționar student = {

"nume": "John Doe", "varsta": 20,

"materii": ["Matematică", "Engleză", "Istorie"],

"note": {"Matematică": 9.5, "Engleză": 8.75, "Istorie": 9.0}

}

# Accesarea valorilor din dicționar print(student["nume"]) # Output: John Doe print(student["varsta"]) # Output: 20

# Actualizarea valorii unei chei în dicționar student["varsta"] = 21 print(student["varsta"]) # Output: 21

# Verificarea existenței unei chei în dicționar folosind operatorul 'in' if "materii" in student:

print("Cheia 'materii' există în dicționar.")

# Accesarea unei liste de chei în dicționar folosind metoda keys() chei = student.keys()

print(chei)

# Accesarea unei liste de valori în dicționar folosind metoda values() valori = student.values()

print(valori)

# Accesarea unei liste de perechi (cheie, valoare) în dicționar folosind metoda items()

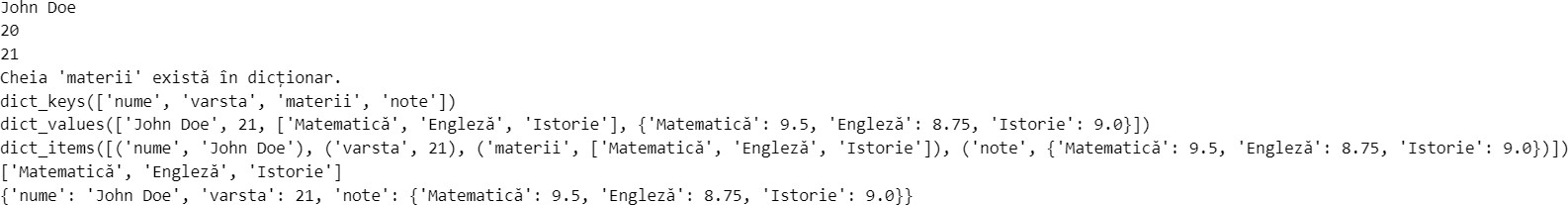
perechi = student.items() print(perechi)

# Ștergerea unei chei și valorii corespunzătoare din dicționar folosind metoda pop()

valoare\_materii = student.pop("materii") print(valoare\_materii)

print(student)

**OUTPUT:**



În acest exemplu, am creat un dicționar numit student care conține informații despre un student. Am demonstrat utilizarea unor metode comune precum accesarea valorilor prin chei, actualizarea valorilor, verificarea existenței cheilor, accesarea listelor de chei, valori și perechi cheie-valoare, ștergerea cheilor și valorilor folosind metoda pop().

**Exercitiul 3.** Să se citească un număr n < 10, să se creeze o listă de numere întregi, cu dimensiunea n şi să se adauge elementele citite de la tastatură. Să se ordoneze crescător elementele listei şi să se afișeze lista.

#Scrie codul aici:

n = int(input("Introduceți un număr n (mai mic decât 10): ")) lista = []

for i in range(n):

numar = int(input("Introduceți un număr întreg: ")) lista.append(numar)

lista.sort()

print("Lista ordonată crescător:", lista)

**OUTPUT:**



* Începem prin citirea numărului `n` de la tastatură folosind funcția `input()`. Îl convertim într-un întreg utilizând funcția `int()`.
* Inițializăm o listă goală `lista` pentru a stoca numerele citite.
* Utilizând un ciclu `for` și funcția `range(n)`, parcurgem valorile de la 0 la `n-1`.
* În fiecare iterație, citim un număr întreg de la tastatură și îl convertim într-un întreg utilizând funcția

`int()`. Apoi, adăugăm numărul la lista `lista` folosind metoda `append()`.

* După ce am adăugat toate numerele în listă, folosim metoda `sort()` pentru a ordona elementele în ordine crescătoare.
* La final, afișăm lista ordonată utilizând funcția `print()`.

**Exercitiul 4.** Să se creeze o listă de 5 elemente cu denumirile unor oraşe. Să se afişeze lista ordonată descrescător după lungimea șirului, utilizând opţiunile metodei 'sort()' - o funcție ce returnează lungimea șirului utilizată ca cheie de sortare.

help(list.sort)

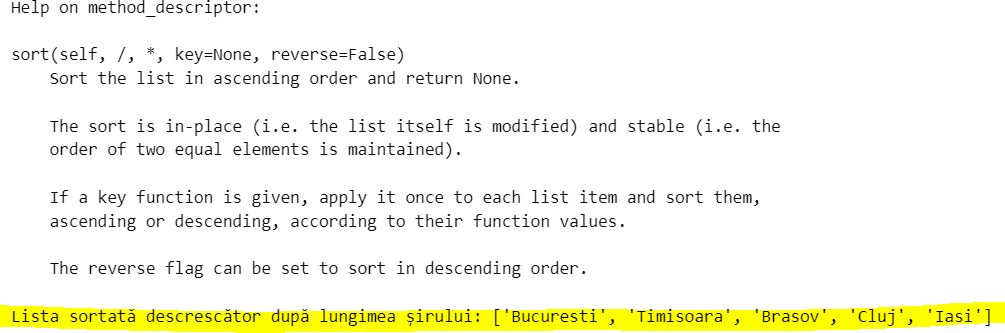
#Scrie codul aici:

oras\_list = ["Bucuresti", "Timisoara", "Cluj", "Iasi", "Brasov"]

oras\_list.sort(reverse=True, key=len)

print("Lista sortată descrescător după lungimea șirului:", oras\_list)

**OUTPUT:**



**Exercitiul 5.** Să se creeze o listă de liste cu denumiri de echipamente IT, preţul şi cantitatea acestora. Calculaţi valoarea fiecărui echipament, adăugaţi-o în listă şi sortaţi în funcţie de valoare. (sugestie: folosiți metoda `list.sort()` și o funcție lambda ca cheie de sortare).

#Scrie codul aici: echipamente = [

["Laptop", 2000, 5],

["Monitor", 500, 10],

["Tastatură", 100, 20],

["Mouse", 50, 30],

["HDD", 200, 15]

]

# Calcularea valorii fiecărui echipament și adăugarea în listă for echipament in echipamente:

valoare = echipament[1] \* echipament[2] echipament.append(valoare)

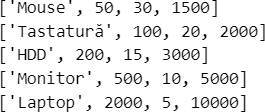
# Sortarea listei în funcție de valoare utilizând metoda sort() și o funcție lambda ca cheie de sortare

echipamente.sort(key=lambda x: x[3])

# Afișarea listei sortate

for echipament in echipamente: print(echipament)

**OUTPUT:**



**Exercitiul 6.** Să se creeze două liste: lista\_angajati cu numele şi prenumele angajaţilor şi lista\_clienti cu numele şi prenumele clienţilor.

1. Să se afişeze numele angajaților care sunt şi clienți.
2. Rezolvați tema folosind seturi și operații specifice acestora.

#Scrie codul aici:

lista\_angajati = ["John Smith", "Alice Johnson", "Robert Davis", "Emma Wilson"]

lista\_clienti = ["Alice Johnson", "David Anderson", "Emma Wilson", "Sophia Miller"]

# a. Afisarea angajatilor care sunt si clienti angajati\_si\_clienti = []

for angajat in lista\_angajati: if angajat in lista\_clienti:

angajati\_si\_clienti.append(angajat)

print("Angajatii care sunt si clienti:", angajati\_si\_clienti) # b. Rezolvarea temei folosind seturi

set\_angajati = set(lista\_angajati) set\_clienti = set(lista\_clienti)

angajati\_si\_clienti\_set = set\_angajati.intersection(set\_clienti)

print("Angajatii care sunt si clienti (folosind seturi):", angajati\_si\_clienti\_set)

**OUTPUT:**



**Exercitiul 7.** Să se creeze o listă de dicţionare cu următoarele chei: id, nume şi salariu pentru următorii angajaţi: Popescu, Ionescu, Vasilescu. Dacă angajaţii au salariul mai mic decât 5000, să se majoreze salariul cu 10%.

#Scrie codul aici:

angajati = [

{"id": 1, "nume": "Popescu", "salariu": 4500},

{"id": 2, "nume": "Ionescu", "salariu": 5500},

{"id": 3, "nume": "Vasilescu", "salariu": 4800}

]

for angajat in angajati:

if angajat["salariu"] < 5000: angajat["salariu"] \*= 1.1

print(angajati)

**OUTPUT:**



**Exercitiul 8.** Să se creeze un nou set de date din clienti\_leasing500.csv, având coloanele name\_client, deposit\_amount și prescoring, dacă val\_credits\_ron = 0 și deposit\_amount > 1000 și să se modifice coloana prescoring = 6 dacă deposit\_amount > 5000. Salvați setul modificat într-un nou fișier csv.

import pandas as pd

# Citirea datelor din fișierul CSV

data = pd.read\_csv('clienti\_leasing500.csv')

# Selecția datelor conforme cu condițiile

selected\_data = data[(data['VAL\_CREDITS\_RON'] == 0) & (data['DEPOSIT\_AMOUNT']

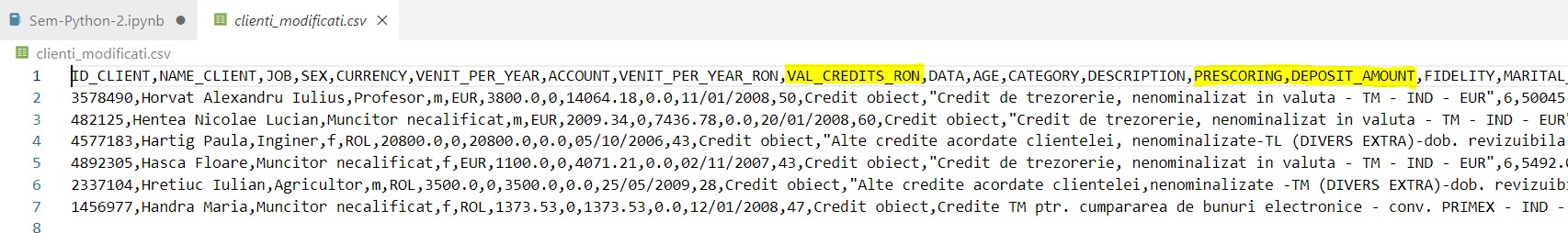
> 1000)]

# Modificarea coloanei 'prescoring' în cazul în care 'deposit\_amount' este mai mare decât 5000

selected\_data.loc[selected\_data['DEPOSIT\_AMOUNT'] > 5000, 'PRESCORING'] = 6

# Salvarea setului de date modificat într-un nou fișier CSV selected\_data.to\_csv('clienti\_modificati.csv', index=False)

**OUTPUT:**



**Exercitiul 9.** Să se reprezinte grafic (de tip pie) durata însumată a sms pe lună, din \*\*phone\_data.csv\*\*.

import pandas as pd

import matplotlib.pyplot as plt

# Citirea datelor din fișierul CSV data = pd.read\_csv('phone\_data.csv')

# Conversia coloanei 'date' la tipul de dată 'datetime' data['date'] = pd.to\_datetime(data['date'])

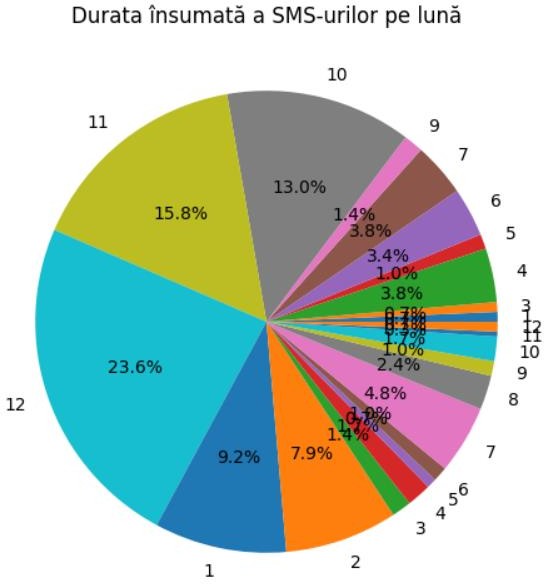
# Extrage anul și luna din coloana 'date' data['year'] = data['date'].dt.year data['month'] = data['date'].dt.month

# Gruparea datelor pe lună și suma duratei SMS-urilor sms\_duration\_by\_month = data[data['item'] == 'sms'].groupby(['year', 'month'])['duration'].sum()

# Transformarea rezultatului într-un DataFrame sms\_duration\_by\_month = sms\_duration\_by\_month.reset\_index()

# Graficul pie cu durata însumată a SMS-urilor pe lună plt.figure(figsize=(8, 6)) plt.pie(sms\_duration\_by\_month['duration'], labels=sms\_duration\_by\_month['month'], autopct='%1.1f%%') plt.title('Durata însumată a SMS-urilor pe lună') plt.show()

**OUTPUT:**



**Exercitiul 10.** Să se creeze setul format din \*\*user\_usage\*\* și \*\*supported\_devices\*\* și să se reprezinte grafic (bare verticale) traficul însumat (coloana \*\*monthly\_mb\*\*) pentru fiecare brand (coloana \*\*Retail Branding\*\*).

import pandas as pd

import matplotlib.pyplot as plt

# Citirea datelor din fișiere

user\_usage = pd.read\_csv("user\_usage.csv") supported\_devices = pd.read\_csv("supported\_devices.csv") user\_device = pd.read\_csv("user\_device.csv")

# Unirea seturilor de date pe baza coloanei 'use\_id' merged\_data = pd.merge(user\_usage, user\_device, on='use\_id')

merged\_data = pd.merge(merged\_data, supported\_devices, on='device')

# Calcularea traficului însumat pentru fiecare brand

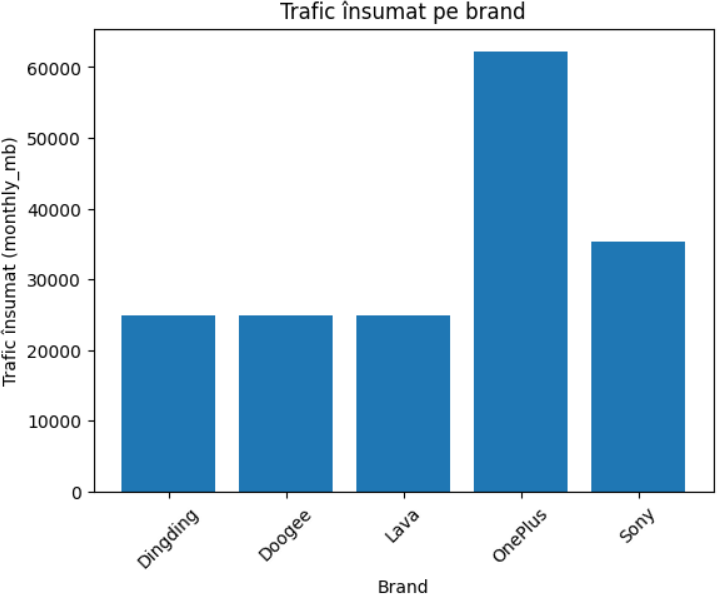
traffic\_by\_brand = merged\_data.groupby('Retail Branding')['monthly\_mb'].sum()

# Reprezentarea grafică (bare verticale) plt.bar(traffic\_by\_brand.index, traffic\_by\_brand.values) plt.xlabel('Brand')

plt.ylabel('Trafic însumat (monthly\_mb)') plt.title('Trafic însumat pe brand') plt.xticks(rotation=45)

plt.show()

**OUTPUT:**



**Exercitiul 11.** Să se afișeze, utilizând fișierul \*\*phone\_data.csv\*\*, durata însumată pentru fiecare lună și durata însumată pentru un anumit tip de rețea (\*\*mobile\*\*) pentru fiecare lună.

import pandas as pd

# Citirea datelor din fișier

phone\_data = pd.read\_csv("phone\_data.csv")

# Convertirea coloanei 'date' în format dată phone\_data['date'] = pd.to\_datetime(phone\_data['date'])

# Extragem luna din coloana 'date' phone\_data['month'] = phone\_data['date'].dt.month

# Calcularea duratei însumate pentru fiecare lună total\_duration\_per\_month = phone\_data.groupby('month')['duration'].sum()

# Calcularea duratei însumate pentru rețeaua "mobile" pentru fiecare lună mobile\_duration\_per\_month = phone\_data[phone\_data['network\_type'] == 'mobile'].groupby('month')['duration'].sum()

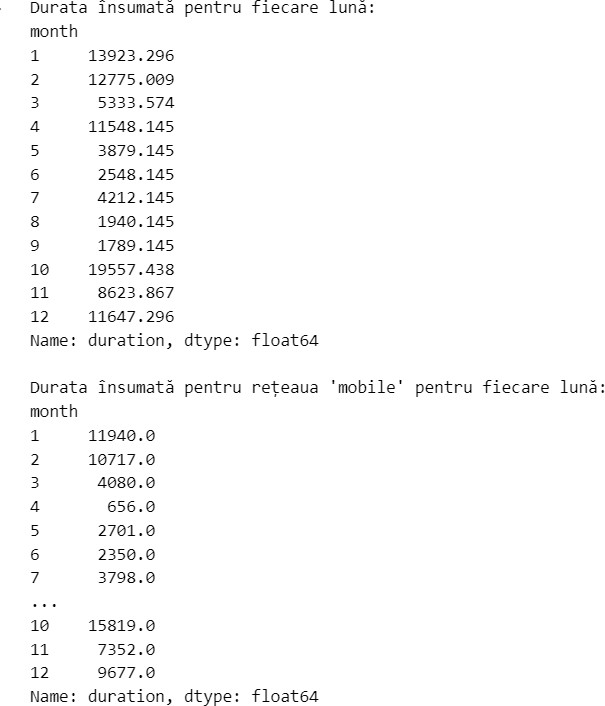
# Afișarea rezultatelor

print("Durata însumată pentru fiecare lună:") print(total\_duration\_per\_month)

print()

print("Durata însumată pentru rețeaua 'mobile' pentru fiecare lună:") print(mobile\_duration\_per\_month)

**Output:**



# **Capitolul 2 Programare SAS**

**EXERCIȚIUL 1.** Să se importe datele referitoare la produsele vandute din Excel în SAS Studio și să se creeze un set de date persistent pe baza acesteia.

**Rezolvare:**

1. **libname proiect '/home/u63378497/ProiectPacheteSoftware';**: Această instrucțiune stabilește o legătură între o bibliotecă SAS (denumită "proiect") și un director local din sistemul de fișiere. Calea specificată (**/home/u63378497/ProiectPacheteSoftware**) reprezintă directorul în care se află fișierul Excel și va fi utilizată ulterior pentru a specifica calea către fișier.
2. **FILENAME REFFILE '/home/u63378497/ProiectPacheteSoftware/ProduseVandute.xlsx';**: Această instrucțiune definește o variabilă de sistem care conține calea completă către fișierul Excel pe care dorim să-l importăm.
3. **PROC IMPORT DATAFILE=REFFILE DBMS=XLSX OUT=proiect.baza\_date replace;**: Această instrucțiune pornește procesul de import al fișierului Excel în SAS. **DATAFILE=REFFILE** specifică fișierul sursă, **DBMS=XLSX** indică tipul de fișier (în acest caz, un fișier Excel), **OUT=proiect.baza\_date** specifică numele și locația setului de date SAS rezultat (în biblioteca "proiect"), iar **replace** indică că dacă există deja un set de date cu același nume, acesta va fi înlocuit.
4. **SHEET='Sheet1';**: Această opțiune specifică numele foii de calcul din fișierul Excel pe care dorim să o importăm. În acest exemplu, se presupune că foaia de calcul se numește "Sheet1".
5. **GETNAMES=YES;**: Această opțiune indică faptul că prima linie din fișierul Excel conține numele variabilelor, iar acestea trebuie utilizate ca denumiri de coloane în setul de date SAS rezultat.
6. **RUN;**: Această instrucțiune încheie blocul PROC IMPORT.

/\*1. Să se importe datele referitoare la produsele vandute din Excel în SAS Studio

și să se creeze un set de date persistent pe baza acesteia.\*/

libname proiect '/home/u63378497/ProiectPacheteSoftware';

FILENAME REFFILE '/home/u63378497/ProiectPacheteSoftware/ProduseVandute.xlsx';

PROC IMPORT DATAFILE=REFFILE

DBMS=XLSX

OUT=proiect.baza\_date replace;

SHEET='Sheet1';

GETNAMES=YES;

RUN;

TITLE "Baza de date";

PROC CONTENTS DATA=proiect.baza\_date

RUN;

**Output:**



**EXERCIȚIUL 2.** Să se definească un format care să transforme numele județelor în codurile lor corespunzătoare (Constanța - CT, București - B, Cluj - CJ, Iasi - IS) și apoi să se verifice câte comenzi sunt înregistrate pentru fiecare județ în parte.

**Rezolvare:**

Definirea un format pentru județe:

**proc format;**

**value $judet 'Constanta' = 'CT'**

**'Bucuresti' = 'B'**

**'Cluj' = 'CJ' 'Iasi' = 'IS';**

**run;**

Aplicarea formatului pe tabela noastră pentru a verifica:

**proc print data=proiect.baza\_date;**

**format Judet $judet. ;**

**run;**

Frecvența pentru fiecare județ, folosind procedura ***FREQ()***:

title "Frecventa de aparitie pentru fiecare judet";

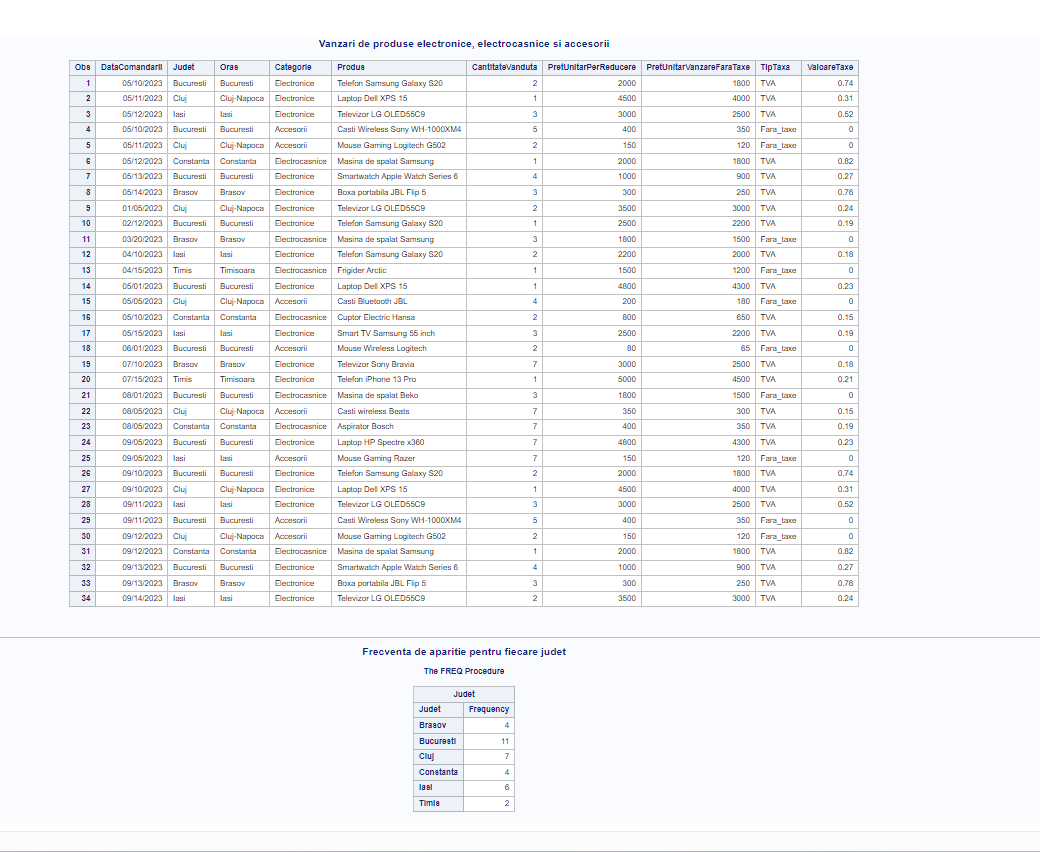
proc FREQ data=proiect.baza\_date;

TABLES Judet /nocum nopercent;

FORMAT Judet $judet.;

run;

**Output:**



**Interpretarea economica a rezultatelor obtinute:**

Conform rezultatelor obținute, putem observa faptul că județul Bucuresti a înregistrat cele mai multe vânzări.

**EXERCIȚIUL 3.** Utilizând setul de date importat, să se afișeze toate produsele din categoria „Electronice” care au cantitatea vândută cuprinsă între 2 și 3 bucăți.

**Rezolvare:**

Pentru a afișa produsele din categoria „Electronice” care au cantitatea vândută între 2 și 3 bucăți se utilizează procedura ***PRINT***, iar pe baza bazei de date importate se afișează coloanele *DataComandarii, Categorie, Produs* și *CantitateVanduta*.

Se folosește instrucțiunea ***WHERE*** pentru a crea o procesare condițională a datelor. Astfel, categoria se dorește a fi "Electronice", iar *CantitateVanduta* să fie în intervalul închis [2,3]. Pentru aceasta din urmă se folosește operatorul ***BETWEEN AND***.

title "Achizitii de produse electronice care au cantitatea vândută cuprinsă între 2 și 3 bucăți.";

proc print data=proiect.baza\_date;

where Categorie eq 'Electronice' and (CantitateVanduta between 2 and 3);

var DataComandarii Categorie Produs CantitateVanduta;

run;

**Output:**



**EXERCIȚIUL 4.** Directorul firmei de aprovizionări dorește să analizeze datele înregistrate pentru fiecare oraș în parte din anul 2023.

**Rezolvare:**

DATA proiect.bucuresti; SET proiect.baza\_date;

WHERE Oras eq 'Bucuresti' AND YEAR(DataComandarii) eq 2023;

RUN;

DATA proiect.iasi; SET proiect.baza\_date;

WHERE Oras eq 'Iasi' AND YEAR(DataComandarii) eq 2023;

RUN;

DATA proiect.constanta; SET proiect.baza\_date;

WHERE Oras eq 'Constanta' AND YEAR(DataComandarii) eq 2023;

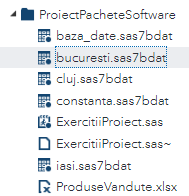
RUN;

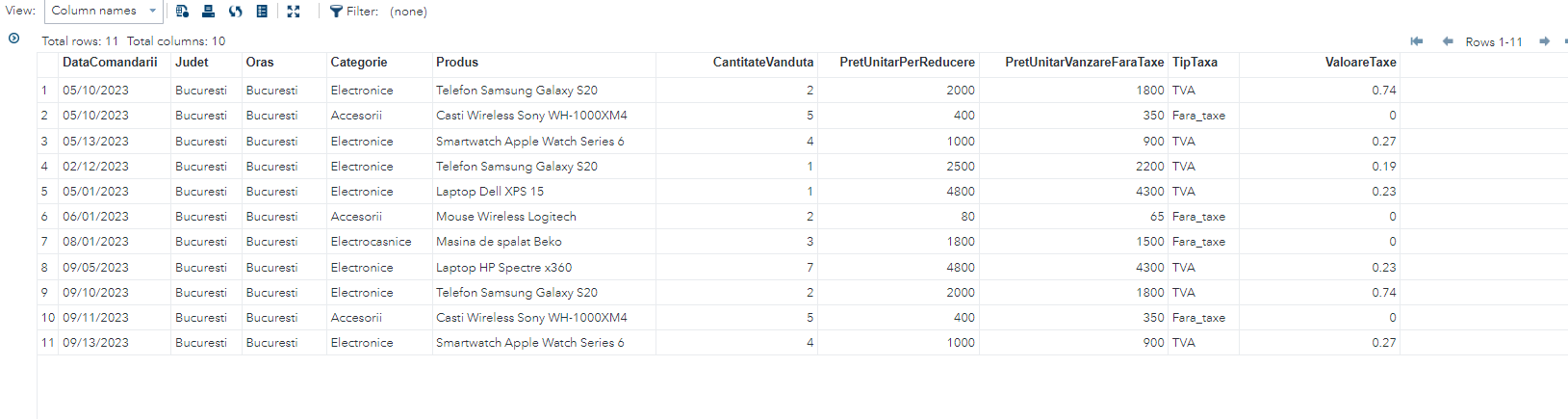
DATA proiect.cluj; SET proiect.baza\_date;

WHERE Oras eq 'Cluj' AND YEAR(DataComandarii) eq 2023;

RUN;

**Output:**





**Interpretarea economica a rezultatelor obtinute:**

Avem acum 4 subtabele, unul pentru fiecare oraș, care conțin toate tranzacțiile înregistrate în anul anterior (2023). Acestea pot fi folosite mai departe pentru a genera statistici și rapoarte mai bine organizate și ușor de interpretat.

**EXERCIȚIUL 5.** Să se realizeze un raport privind totalul de produse vândute în fiecare județ, totodată, afișând și cât însumează acesta.

**Rezolvare:**

**PROC** **SORT** DATA = proiect.baza\_date; BY judet;

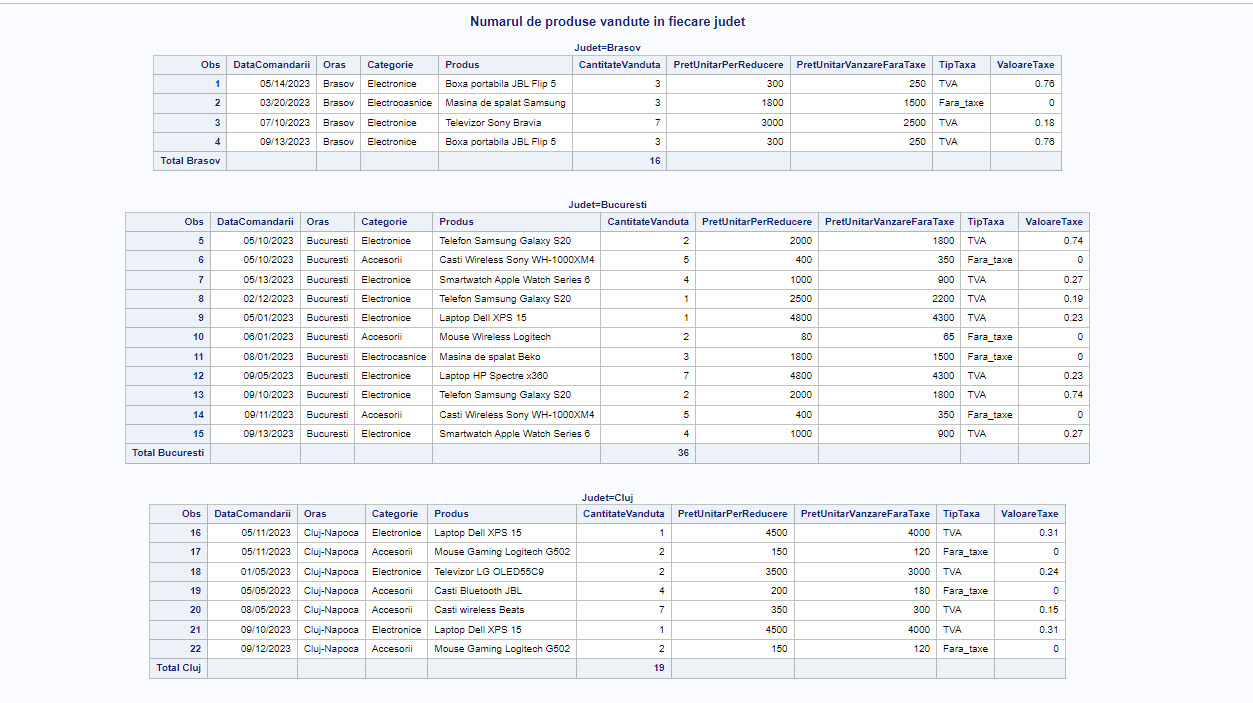
**PROC** **PRINT** DATA = proiect.baza\_date sumlabel = 'Total #byval(judet)' grandtotal\_label = 'Total';

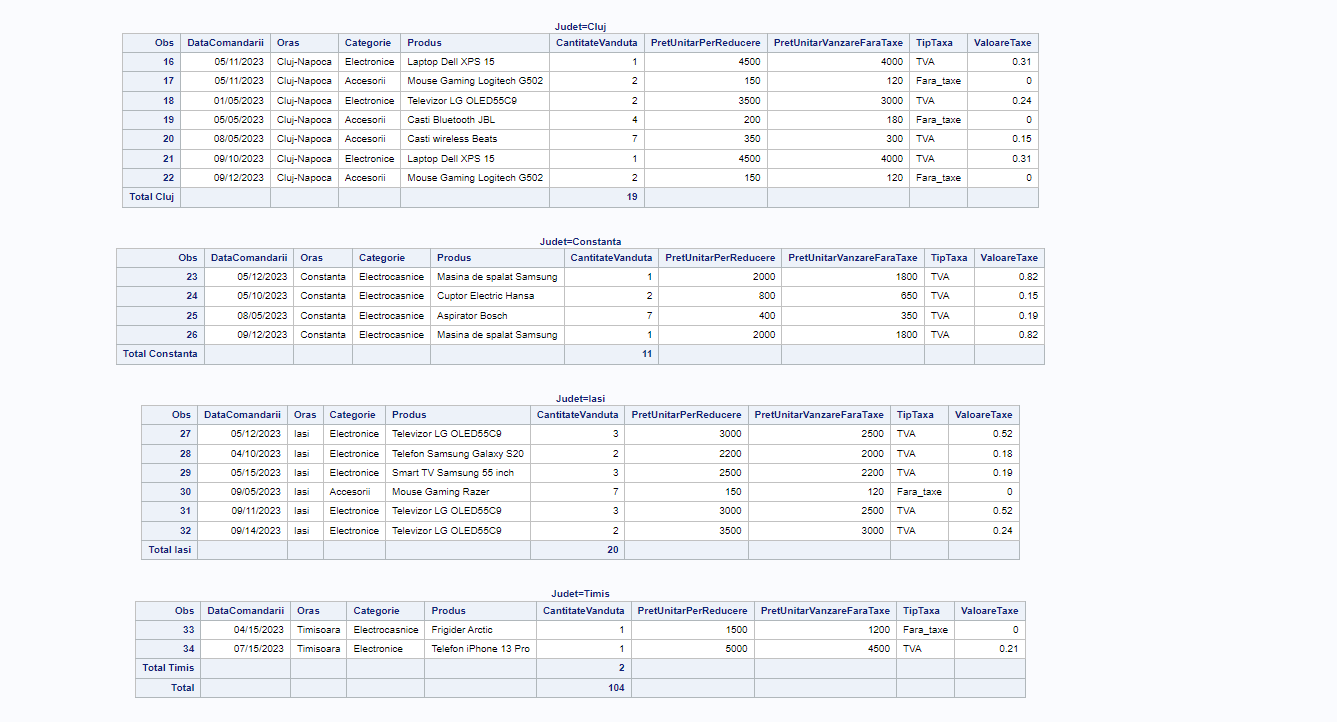
BY judet;

SUM CantitateVanduta;

TITLE 'Numarul de produse vandute in fiecare judet'; **RUN**;

**Output:**





**EXERCIȚIUL 6.**  Să se realizeze un raport privind cantitatea vândută din fiecare produs care face parte din categoria „Accesorii”, precum și totalul de bucăți de produse vândute din această categorie

**Rezolvare:**

proc sort data=proiect.baza\_date; where Categorie eq 'Accesorii'; by Produs;

proc print data=proiect.baza\_date

sumlabel="Total #byval(Produs)" grandtotal\_label="Total";

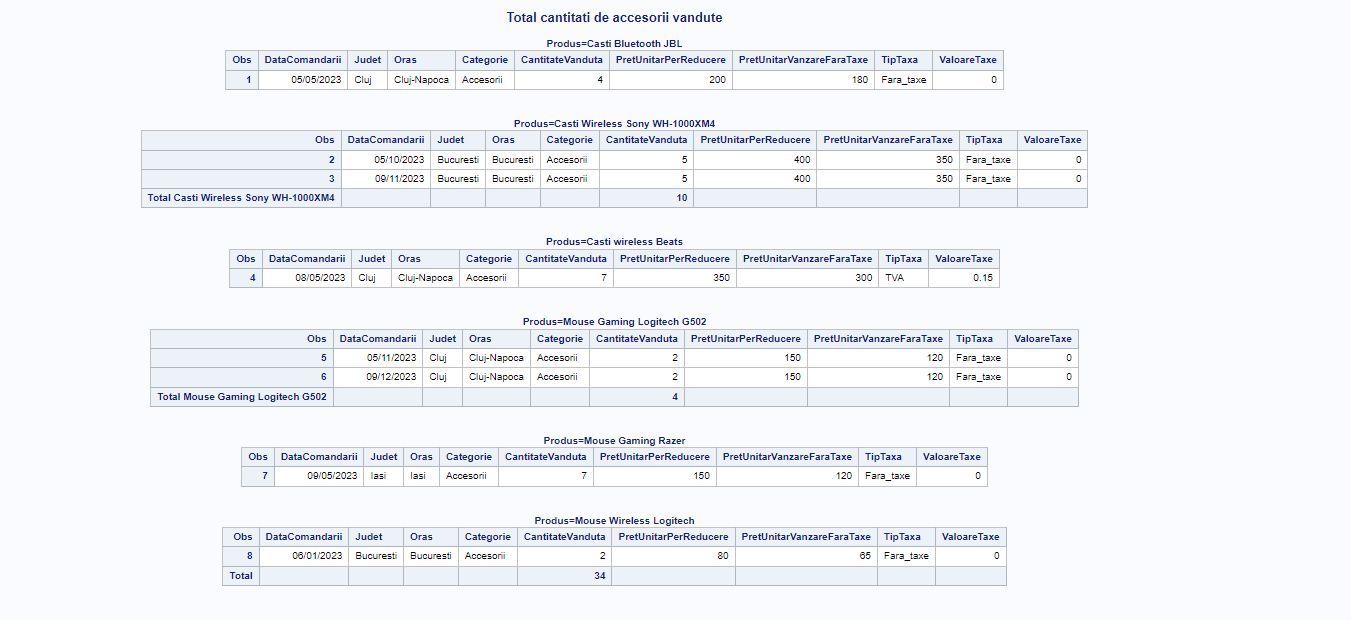
by Produs;

sum CantitateVanduta;

title "Total cantitati de accesorii vandute";

run;

**Output:**



**EXERCIȚIUL 7.** Să se identifice cele mai mici și cele mai mari 5 valori extreme distincte pentru cantitatea de produse vândute.

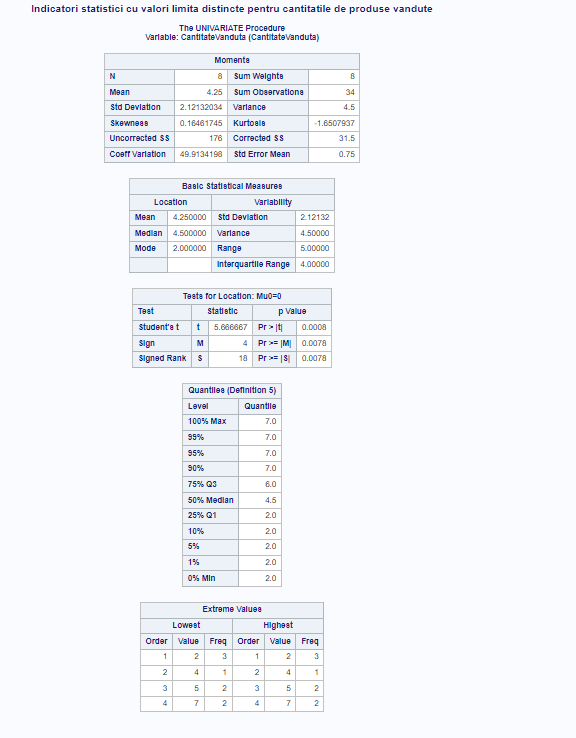
**Rezolvare:**

PROC UNIVARIATE DATA=proiect.baza\_date NEXTRVAL=5 NEXTROBS=0; VAR CantitateVanduta;

Title "Indicatori statistici cu valori limita distincte pentru cantitatile de produse vandute";

RUN;

**Output:**

****

**EXERCIȚIUL 8.** Folosind subseturile de date generate anterior pentru fiecare oraș să se deseneze grafice care să ne arate cantitatea vândută din fiecare produs pentru a putea analiza cererea din fiecare oraș.

**Rezolvare:**

TITLE "Distributia produselor vandute in Bucuresti";

PATTERN value = solid;

PROC GCHART data=proiect.bucuresti;

VBAR Produs / sumvar=CantitateVanduta

type=sum;

RUN;

QUIT;

TITLE "Distributia produselor vandute in Iasi";

PATTERN value = solid;

PROC GCHART data=proiect.iasi;

VBAR Produs / sumvar=CantitateVanduta

type=sum;

RUN;

QUIT;

TITLE "Distributia produselor vandute in Constanta";

PATTERN value = solid;

PROC GCHART data=proiect.constanta;

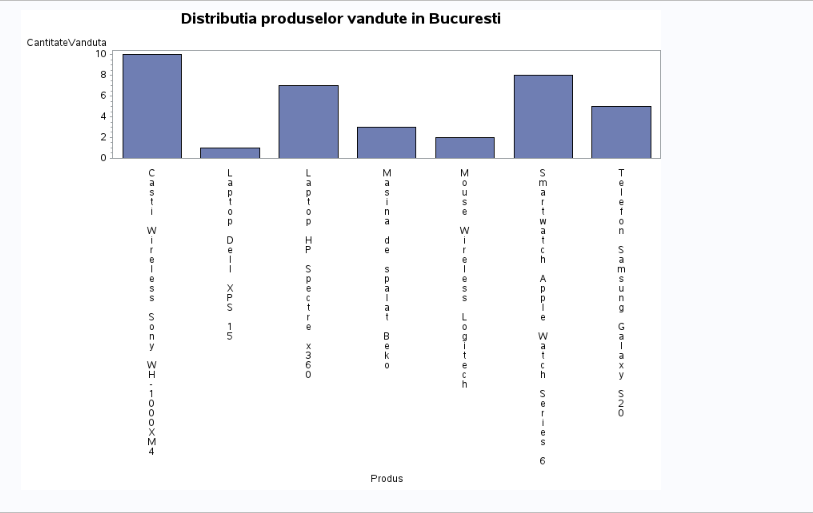
VBAR Produs / sumvar=CantitateVanduta

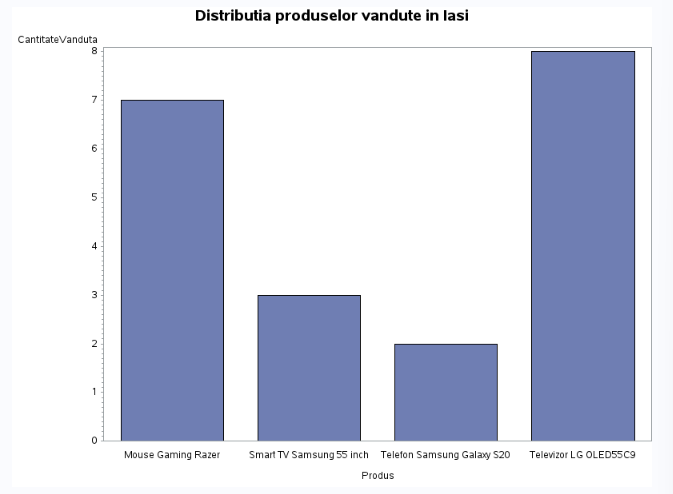
type=sum;

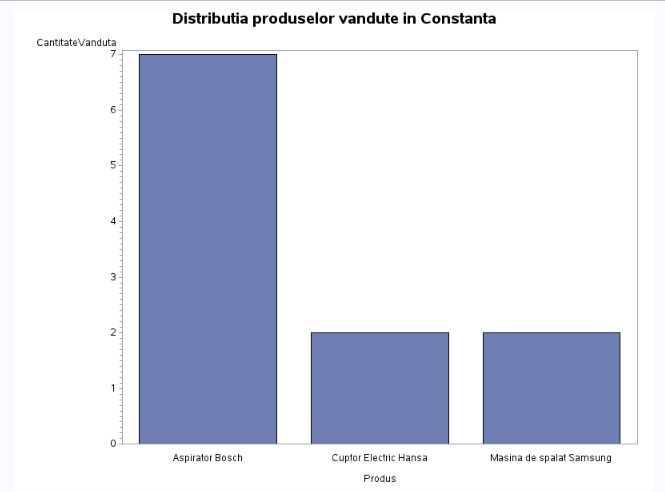
RUN;

QUIT;

**Output:**







**EXERCIȚIUL 9.** Să se afișeze toate produsele din categoria "Accesorii" cu un preț unitar mai mic de 100 de unități monetare

**Rezolvare:**

PROC PRINT DATA=proiect.baza\_date;

WHERE Categorie = 'Accesorii' AND PretUnitarPerReducere < 100;

VAR Produs Categorie PretUnitarPerReducere;

TITLE "Accesorii cu PretUnitar < 100";

RUN;

**Output:**



**EXERCIȚIUL 10.** Să se afișeze toate produsele din categoria"Accesorii" cu un preț unitar cuprins între 100 și 500 unitati monetare.

**Rezolvare:**

title "Achizitii de accesorii care au pretul unitar cuprins între 100 și 500 unitati monetare ";

proc print data=proiect.baza\_date;

where Categorie eq 'Accesorii' and (PretUnitarPerReducere between 100 and 500);

var DataComandarii Categorie Produs CantitateVanduta PretUnitarPerReducere;

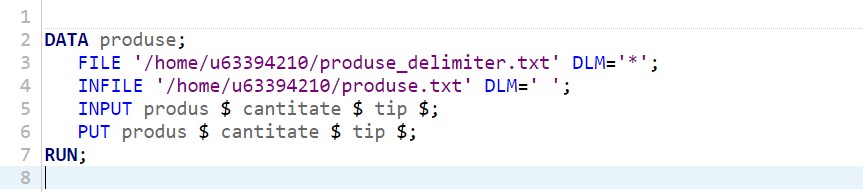
run;

**Output:**

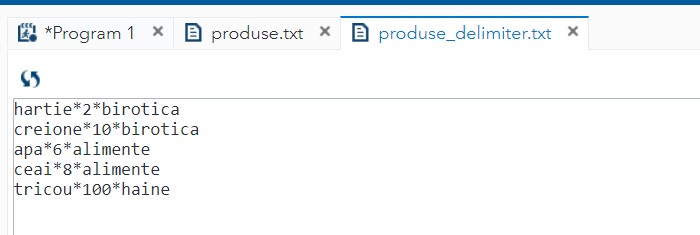
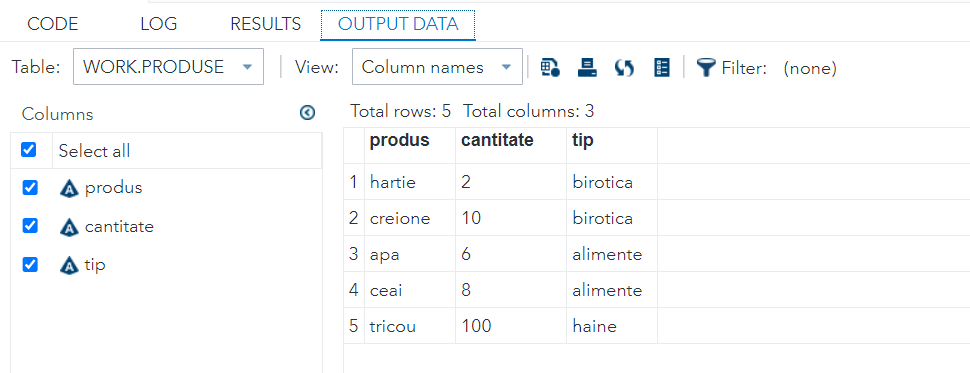
****

# **Exerciții propuse seminarii SAS**

1. **Să se definească, folosind datele din fișierul produse.txt, un fișier text cu delimitatorul\*.**

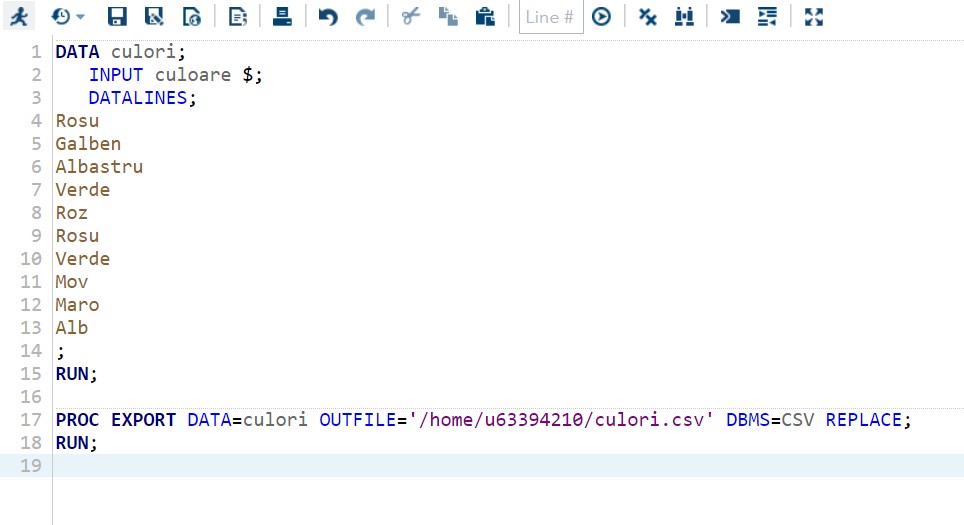


**Output:**

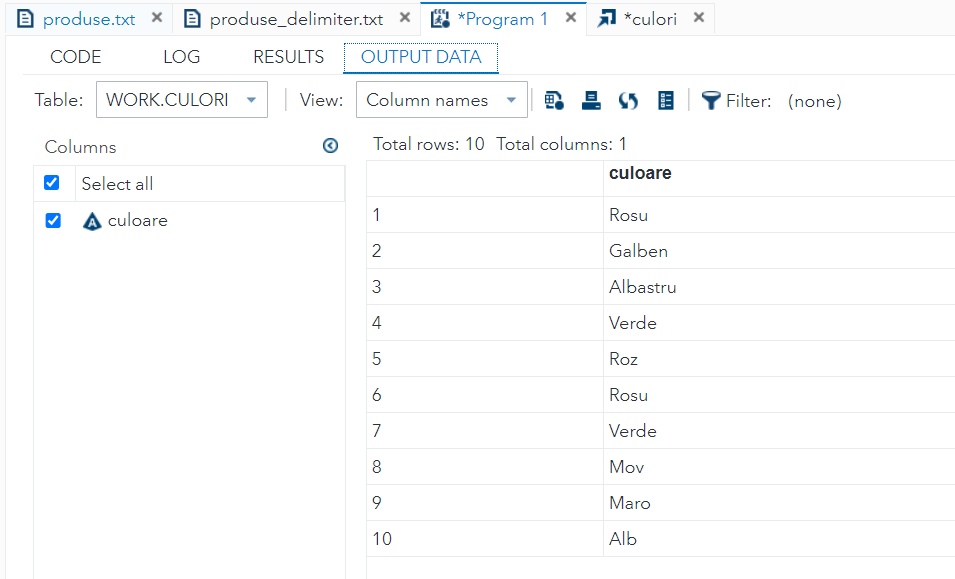


1. **Fie următoarele date:**

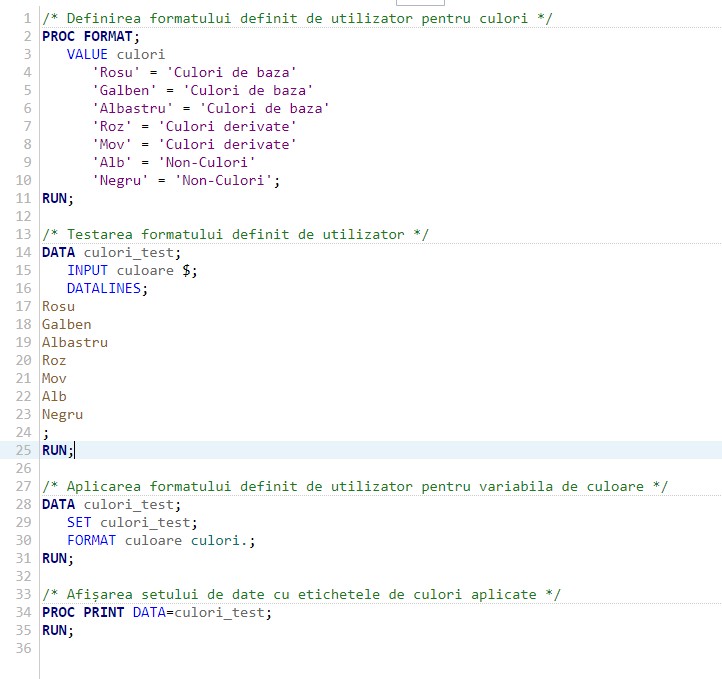
**Rosu, Galben, Albastru, Verde, Roz, Rosu, Verde, Mov, Maro, Alb Să se creeze un fişier de date SAS pe baza lor.**



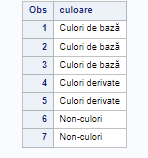
**Output:**



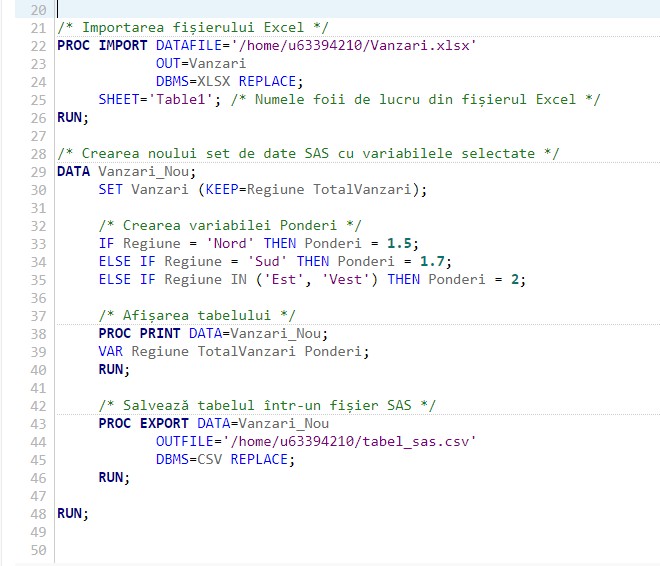
1. **Să se creeze un format definit utilizator pentru: culorile de bază (Rosu, Galben, Albastru), culorile derivate și non-culori (Alb, Negru).**



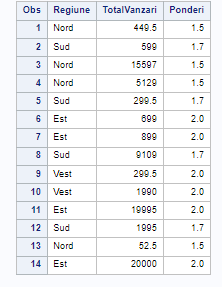
**Output:**



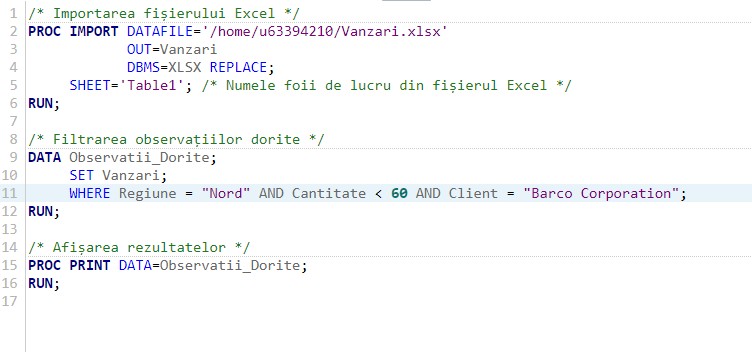
1. **Utilizând fişierul Excel ‘Vanzari’, creaţi cu SELECT un nou set de date SAS ce va conţine Regiune şi VanzariTotale (utilizând optiunea keep), împreună cu o nouă variabilă numită Ponderi, cu următoarele valori:**
   * 1,5 pentru regiunea Nord;
   * 1,7 pentru regiunea Sud;
   * 2 pentru regiunile Est şi Vest.



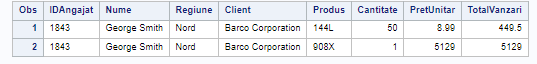
**Output:**



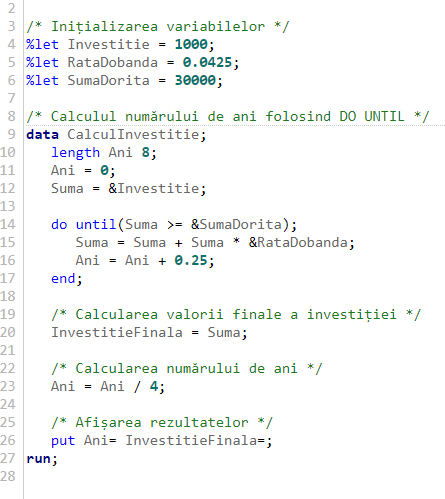
1. **Folosind datele din fişierul Excel ‘Vânzări’, afişaţi toate observaţiile pentru Regiunea Nord şi Cantitatea mai mică de 60. Includeţi în listă orice observaţie unde numele clientului este *Barco Corporation.***



**Output:**



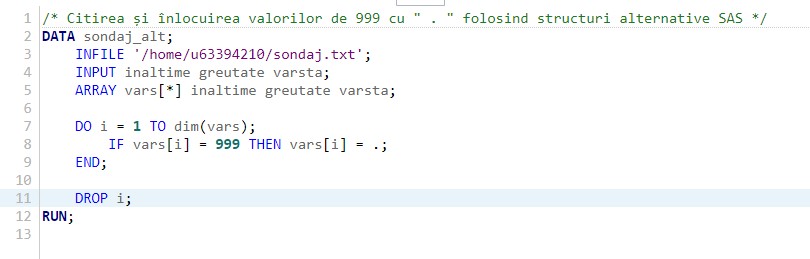
1. **Aveţi o investiţie iniţială de 1000$, cu o rată a dobânzii trimestriale de 4,25%. În câţi ani veţi obţine 30000$? Utilizați DO WHILE sau DO UNTIL pentru rezolvare.**



**Output:**



1. **În fişierul text sondaj.txt, valorile lipsă pentru variabilele numerice au fost înlocuite cu valoarea 999 (o practică des întâlnită în pachetele statistice cum ar fi SPSS). Se cere să se înlocuiască valoarea 999 cu “ .“, în două variante, folosind stucturi SAS alternative şi respectiv masive. Fişierul sondaj.txt conţine observaţii referitoare la înălţimea, greutatea şi vârsta, toate numerice. Datele sunt separate prin spaţiu.**



**Output:**



1. **Se dorește crearea a două seturi de date cu următorul conținut (fie prin import, fie prin specificarea datelor în cadrul programului SAS):**

3 C

1 A

7 E

și 2 B

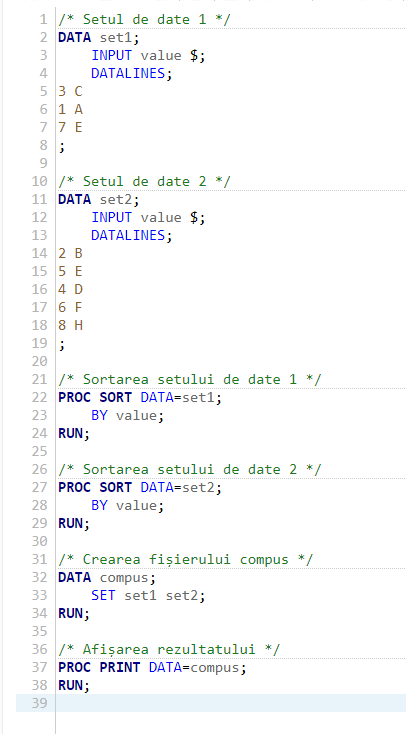
5 E

4 D

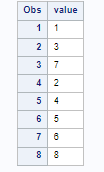
6 F

8 H.

**Se cere să se creeze un fișier compus care conține datele în ordine alfabetică.**

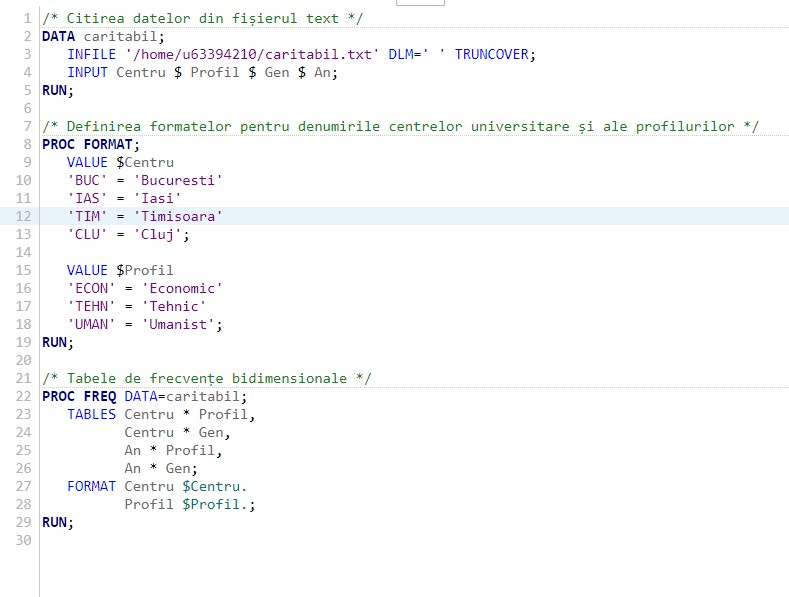


**Output:**

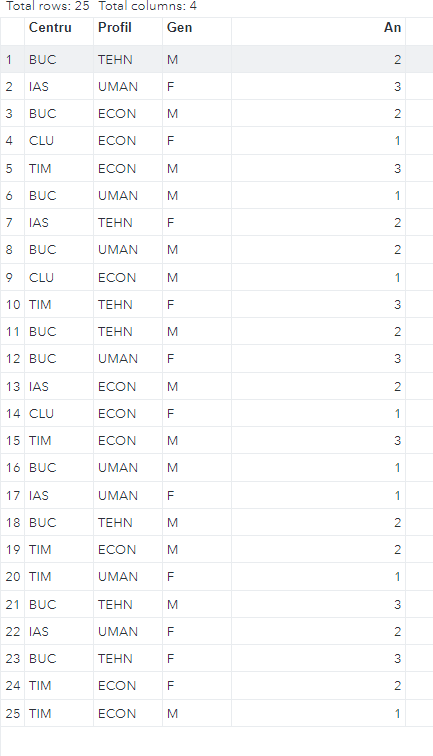


1. Un număr de 25 de studenți din diferite centre universitare au participat la strângerea de fonduri pentru un eveniment caritabil. În fișierul text “caritabil.txt” au fost salvate, în ordine, date referitoare la centrul universitar din care provine studentul (BUC – Bucuresti, IAS – Iasi, TIM – Timisoara, CLU - Cluj), profilul facultății urmate (ECON – economic, TEHN – tehnic. UMAN - Umanist), genul studentului (M sau F), precum și anul de studiu (1-3).

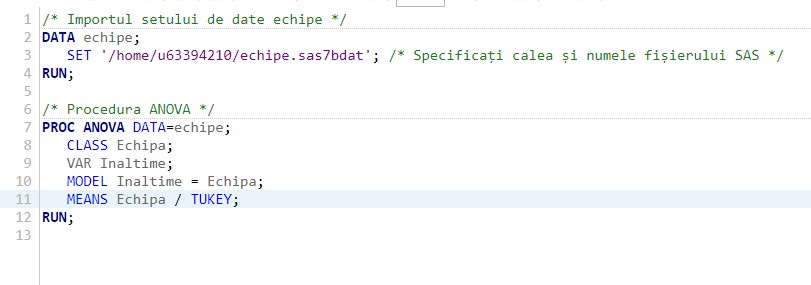
Se cere să se creeze tabele de frecvențe bidimensionale pentru perechile: Centru-Profil, Centru-Gen, An-Profil și An-Gen. Se va folosi procedura FREQ cu o declarație de tipul TABLES (A B) \* (C D). Se asemenea, se cere să se afișeze forma detaliată a valorilor pentru denumirile centrelor universitare și ale profilurilor.



**Output:**



1. Într-un campionat de bachet între școli evoluează cinci echipe de fete. Se cere să se determine dacă există diferențe semnificative în ceea ce privește înalțimea jucătoarelor celor cinci echipe, dorindu-se, ca la acest nivel, să existe echitate între echipele înscrise în campionat. Se vor folosi datele din setul de date SAS “echipe”.



**Output:**

