**Seminar 3 Programare SAS**

**A. Combinarea seturilor de date**

**1. Concatenarea seturilor de date folosind declaraţia SET**

Declaraţia SET cu un singur set de date în cadrul unei secţiuni de date permite citirea şi modificarea unui set de date existent. Cu două sau mai multe seturi de date, pe lângă citire şi modificare, declaraţia SET ajută la concatenarea datelor, adică la alipirea unui set de date la altul. Aceasta este folositoare atunci când se doreşte combinarea unor seturi de date care au toate sau majoritatea variabilelor identice, observaţiile fiind diferite. Se va crea un set de date nou, cu un număr de observaţii egal cu suma observaţiilor din seturile de date de intrare. Ordinea observaţiilor este determinată de ordinea în care seturile de date apar în declaraţia SET. Dacă unul din seturile de date conţine o variabilă care nu se găseşte în alt set de date, atunci observaţiile acestuia din urmă pentru variabila nouă vor conţine valori lipsă.

***Sintaxa:***

**DATA** *set\_date\_nou;*

**SET** *set\_date\_1 set\_date\_2 ...;*

***Exerciţiul 1:*** Un parc de distracţii are două intrări (sud şi nord) prin intermediul cărora colectează date despre clienţi. Aceste date se referă la numărul tichetului de trecere, dimensiunea grupului din care face parte clientul, vârsta, iar pentru intrarea de nord se colectează şi date referitoare la lotul în care clientul a parcat maşina. Se cere să se combine cele două seturi de date şi să se creeze o variabilă nouă care calculează preţul tichetului de intrare pentru fiecare client în funcţie de vârstă. Rezultatele rulării codului de mai jos sunt prezentate în figura 1.

**\*Crearea setului de date pentru intrarea din sud;  
DATA intrare\_sud;  
 INFILE '/home/nume.prenume/Sud.dat';  
 INPUT Intrare $ NrTichet DimGrup Varsta;  
PROC PRINT DATA = intrare\_sud;  
 TITLE 'Date Intrare Sud';  
RUN;  
\*Crearea setului de date pentru intrarea din nord;  
DATA intrare\_nord;  
 INFILE '/home/nume.prenume/Nord.dat';  
 INPUT Intrare $ NrTichet DimGrup Varsta Lot;  
PROC PRINT DATA = intrare\_nord;  
 TITLE 'Date Intrare Nord';  
RUN;   
\* Crearea setului de date concatenat si a unei variabile noi;  
DATA concatenare;  
 SET intrare\_sud intrare\_nord;  
 IF Varsta = . THEN Plata = .;  
 ELSE IF Varsta < 3 THEN Plata = 0;  
 ELSE IF Varsta < 65 THEN Plata = 60;  
 ELSE Plata = 25;  
PROC PRINT DATA = concatenare;  
 TITLE 'Ambele intrari';  
RUN;**

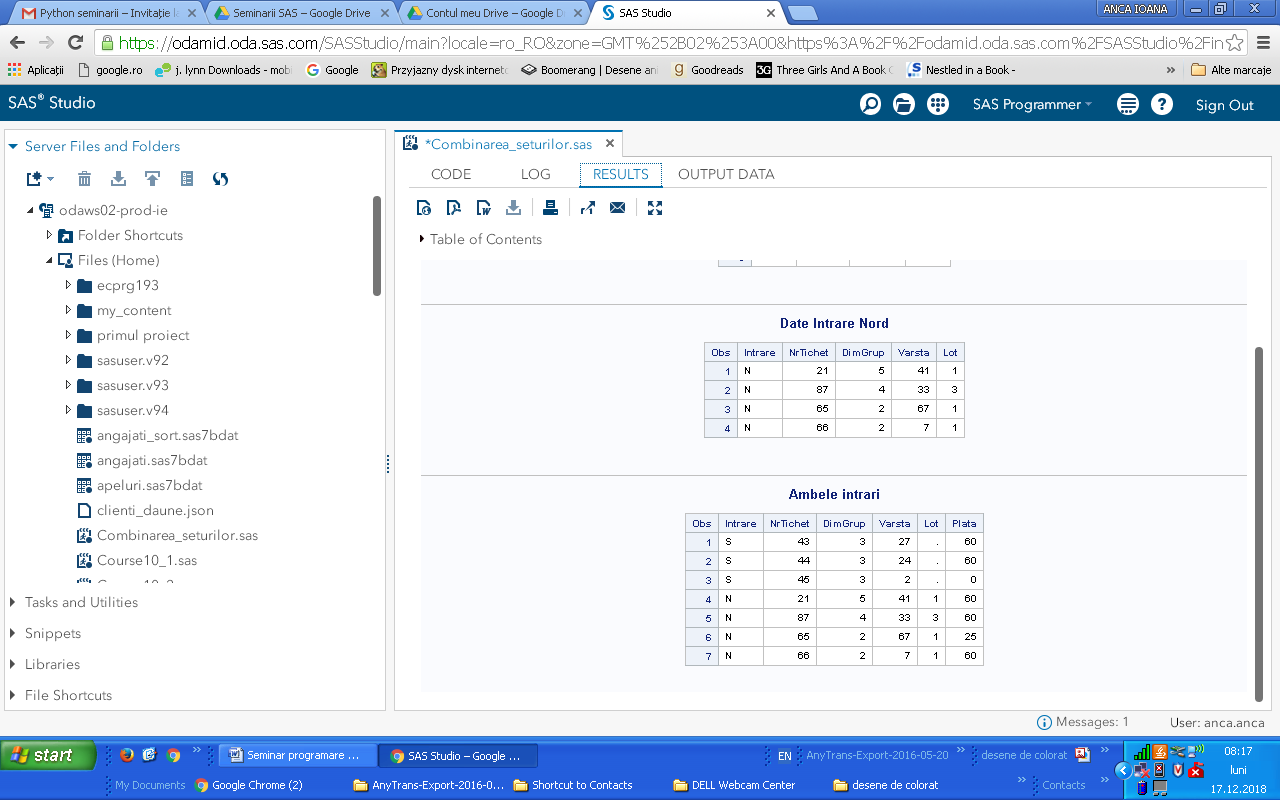
****

Figura 1. Listarea datelor concatenate

**Redenumirea variabilelor**

Există situaţii în care coloane din două seturi de date conţin aceeaşi informaţie, numai că numele coloanelor diferă, în aceest caz neputându-se realiza combinarea seturilor de date în mod corect. Pentru a schimba numele uneia sau mai multor variabile, se poate folosi opţiunea RENAME = în secţiunea de date.

***Sintaxa:***

**S***et\_date (***RENAME** *=(nume\_vechi\_1=nume\_nou\_1*

*nume\_vechi\_2=nume\_nou\_2*

*....*

*nume\_vechi\_n=nume\_nou\_n))*

Opţiunea RENAME = trebuie specificată imediat după setul de date asociat acesteia. De remarcat setul de paranteze externe care încadrează întreaga opţiune şi setul de paranteze interne în cadrul căruia se specifică una sau mai multe variabile care vor fi redenumite.

***Exerciţiul 2.*** O companie are două filiale în Braşov şi Cluj. Se cere să se realizeze un set de date combinat cu date referitoare la angajaţii din cele două filiale. În codul de mai jos sunt exemplificate concatenarea fără redenumirea variabilelor, cu redenumirea unei variabile sau a mai multor variabile.

**\*Crearea setului de date pentru Brasov;**

**data angajatiBV;**

**input Angajat $ Gen $ Judet$;**

**datalines;**

**Damian M Brasov**

**Popa F Brasov**

**Ionita M Brasov**

**;**

**run;**

**\*Crearea setului de date pentru Cluj;**

**data angajatiCJ;**

**input Angajat $ Gen $ Zona$;**

**datalines;**

**Irimia M Cluj**

**Velicu F Cluj**

**;**

**run;**

**data angajati;**

**set angajatiBV angajatiCJ;**

**run;**

**PROC PRINT DATA = angajati;**

**TITLE 'Setul de date combinat fara redenumirea variabilelor';**

**RUN;**

**data angajati;**

**set angajatiBV angajatiCJ (RENAME=(Zona=Judet));**

**run;**

**PROC PRINT DATA = angajati;**

**TITLE 'Setul de date combinat cu redenumirea unei variabile';**

**RUN;**

**data angajati;**

**set angajatiBV (RENAME =(Angajat=Nume Gen=Sex))**

**angajatiCJ (RENAME=(Angajat=Nume Gen = Sex Zona=Judet));**

**run;**

**PROC PRINT DATA = angajati;**

**TITLE 'Setul de date combinat cu redenumirea mai multor variabile';**

**RUN;**

**2. Interclasarea seturilor de date folosind declaraţia SET**

Dacă seturile de date sunt deja sortate (prin intermediul procedurii SORT) după o variabilă relevantă, atunci doar simpla concatenare a acestora poate produce un set de date nesortat. Totuşi, combinarea seturilor de date sortate, numită interclasare, se poate realiza similar concatenării prin adăugarea clauzei BY.

***Sintaxa:***

**DATA** *set\_date\_nou;*

**SET** *set\_date\_1 set\_date\_2 ...;*

**BY**  *listă\_variabile;*

Se aplică aceleași principii ale combinării datelor ca şi la concatenare, cu specificarea faptului că înainte de interclasare observaţiile trebuie sortate anterior după variabila(le) prin clauza BY.

***Exerciţiul 3:*** Se consideră aceleaşi date referitoare la un parc de distracţii ca şi la Exerciţiul 1. Se cere să se combine cele două seturi de date, crescător, în funcţie de numărul tichetului de intrare. Rezultatele rulării codului de mai jos sunt prezentate în figura 2.

**DATA intrare\_sud;  
 INFILE '/home/nume.prenume/Sud.dat';  
 INPUT Intrare $ NrTichet DimGrup Varsta;  
PROC SORT DATA = intrare\_sud;  
 BY NrTichet;  
RUN;  
DATA intrare\_nord;  
 INFILE '/home/nume.prenume/Nord.dat';  
 INPUT Intrare $ NrTichet DimGrup Varsta Lot;  
PROC SORT DATA = intrare\_nord;  
 BY NrTichet;  
RUN;   
\* Interclasarea seturilor de date;  
DATA interclasare;  
 SET intrare\_sud intrare\_nord;  
 BY NrTichet;  
RUN;  
PROC PRINT DATA = interclasare;  
 TITLE 'Ambele intrari in functie de numarul tichetului';  
RUN;**

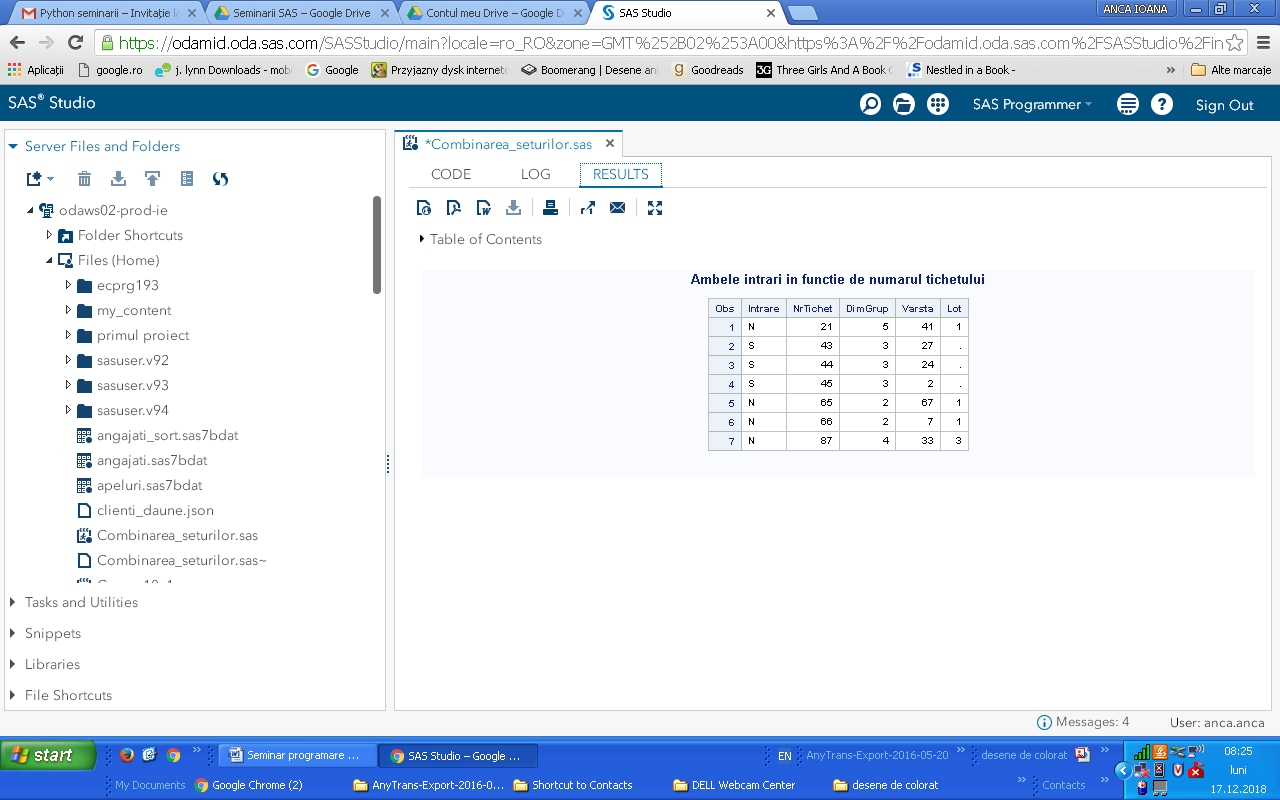
******

Figura 2. Listarea datelor interclasate

**3. Fuziune pe baza unei corespondenţe unu-la-unu**

În situaţiile în care se doreşte realizarea unei corespondenţe între observaţiile unor seturi de date, în vederea combinării acestora, se poate folosi declaraţia MERGE în secţiunea de date. Dacă se cunoaşte faptul că seturile de date sunt în exact aceeaşi ordine din punctul de vedere al entităţilor pe care le descriu, atunci nu este necesară existenţa unei variabile comune. Însă, de obicei, pentru a avea siguranţa existenţei unei corespondenţe, se urmăreşte prezenţa uneia sau mai multor variabile care să definească în mod unic fiecare observaţie. De asemenea, seturile de date de intrare trebuie să fie sortate în prealabil după variabila(le) comune.

***Sintaxa:***

**DATA** *set\_date\_nou;*

**MERGE** *set\_date\_1 set\_date\_2;*

**BY**  *listă\_variabile;*

**Atenţie!** Dacă vor fuziona două seturi de date care au variabile cu acelaşi nume (altele decât cele din clauza BY), atunci variabilele din cel de-al doilea set de date vor suprascrie orice variabilă care are acelaşi nume în primul set de date.

***Exerciţiul 4:*** Un producător de ciocolată păstrează o evidenţă a vânzărilor sale zilnice pentru fiecare tip de ciocolată vândut. Fişierul referitor la vânzări stochează codul produsului şi numărul de unităţi vândute într-o zi, iar un alt fişier conţine informații despre fiecare produs, în termeni de cod, denumire şi descriere. În scopul realizării unui raport privind vânzările zilnice, se cere să se realizeze o fuziune între cele doua fişiere (fişierul cu descrierea produselor este sortat după cod).

**\*Fuziunea bazata pe corespondenta unu-la-unu;  
DATA descriere;   
 INFILE '/home/nume.prenume/Ciocolata.dat' TRUNCOVER;   
 INPUT Cod $ 1-4 Nume $ 6-14 Descriere $ 15-60;   
RUN;**

**PROC SORT DATA = descriere;  
 BY Cod;  
RUN;  
DATA vanzari;   
 INFILE '/home/nume.prenume/Ciocolata\_Vanzari.dat';;   
 INPUT Cod $ 1-4 UnitatiVandute 6-7;   
PROC SORT DATA = vanzari;  
 BY Cod;  
RUN;  
\*Fuziune dupa cod;  
DATA ciocolate;   
 MERGE vanzari descriere;   
 BY Cod;  
PROC PRINT DATA = ciocolate;   
 TITLE "Vanzarile de ciocolata de astazi &sysdate";   
RUN;**

Rezultatele rulării acestui cod sunt prezentate în figura 3.

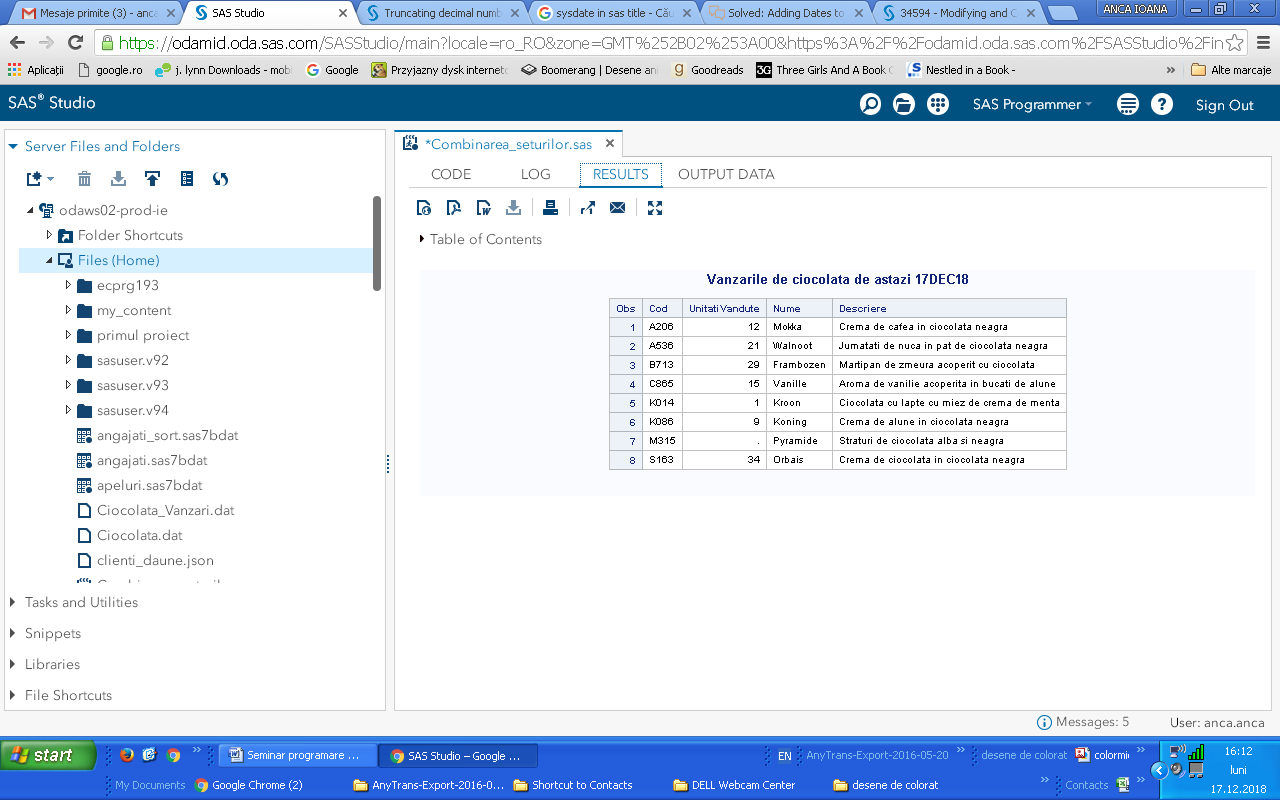
******

Figura 3. Listarea raportului cu vânzările de ciocolată

**Atenţie!** Observaţi că în setul de date final există o valoare lipsă pentru cea de-a şaptea observaţie. De aici putem deduce că în setul de date final sunt incluse toate înregistrările din datele de intrare, indiferent dacă acestea au sau nu corespondent.

**4. Fuziune pe baza unei corespondenţe unu-la-mulţi**

Uneori este nevoie să se combine două seturi de date prin realizarea unei corespondenţe între o înregistrare a unui set de date şi mai multe înregistrări ale altuia. Sintaxa pentru acest tip de fuziune este identică cu cea întâlnită la corespondenţa unu-la-unu. Ordinea în care apar seturile de date în declaraţia MERGE nu este relevantă. Ca şi în cazul anterior, seturile de date trebuie sortate după una sau mai multe variabile comune.

**Atenţie!** Dacă nu se defineşte nicio variabilă în clauza BY, SAS realizează o simplă joncţiune a observaţiilor din seturile de date în funcţie de poziţia acestora în fiecare set, cu alte cuvinte are loc o **fuziune unu-la-unu fără corespondenţă**.

***Exerciţiul 5:*** Un distribuitor de încălţăminte sportivă oferă reduceri pentru toate produsele sale faţă de preţul iniţial. Compania are la dispoziţie două fişiere: unul care conţine detalii despre încălţăminte (Denumire, Sport, Pret) în setul de date SAS “Incaltaminte” şi altul cu coeficienţii de reducere pentru luna martie (Sport, Reducere) în setul de de date SAS „Inc\_reducere”. Se cere să se creeze un set de date combinat din cele două care să conţină şi preţul final de vânzare rotunjit la două zecimale.

**\*Fuziunea bazata pe corespondenta unu-la-multi;  
libname ad\_data "/home/nume.prenume";  
PROC SORT DATA = ad\_data.Incaltaminte;  
 BY Sport;  
RUN;  
PROC SORT DATA = ad\_data.Inc\_reducere;  
 BY Sport;  
RUN;  
DATA Inc\_preturi;  
 MERGE ad\_data.Incaltaminte ad\_data.Inc\_reducere;  
 BY Sport;  
 Pret\_Nou = ROUND(Pret - (Pret \* Reducere), .01);  
PROC PRINT DATA = Inc\_preturi;  
 TITLE 'Lista preturilor pentru luna martie';  
RUN;**

**5. Fuziunea între un raport statistic agregat şi datele iniţiale**

Există situaţii în care este necesară combinarea rezultatelor unor analize statistice agregate cu datele iniţiale, spre exemplu atunci când se doreşte compararea fiecărei observaţii cu media grupului sau când vrem să calculăm un procent folosind un total de grup. În acest sens, se poate utiliza procedura MEANS pentru a agrega datele, urmată de o fuziune a rezultatelor acesteia cu datele iniţiale folosind o corespondenţă unu-la-mulţi.

***Exerciţiul 6:*** Distribuitorul de încălţăminte sportivă de la exerciţiul 5 dispune de o situaţie a vânzărilor din ultima săptămână pentru fiecare tip de încălţăminte (setul de date SAS „inc\_Vanzari” având coloanele Denumire, Sport,Vanzari). Departamentul de marketing solicită realizarea unei analize a vânzărilor din această perioadă, evidențiind, procentual, cu cât a contribuit fiecare model vândut în totalul vânzărilor din categoria sa sportivă.

**\*Fuziunea intre un raport statistic agregat si datele initiale;  
libname ad\_data "/home/nume.prenume";  
proc SORT data=ad\_data.inc\_vanzari;   
BY Sport;  
RUN;  
PROC MEANS NOPRINT DATA = ad\_data.Inc\_vanzari;  
 VAR Vanzari;  
 BY Sport;  
 OUTPUT OUT = date\_agregate SUM(Vanzari) = Total;  
PROC PRINT DATA = date\_agregate;  
 TITLE 'Setul de date agregat';  
RUN;  
\* Fuziunea totalului cu datele initiale;  
DATA Inc\_agregare;  
 MERGE ad\_data.inc\_Vanzari date\_agregate;  
 BY Sport;  
 Procent = Vanzari / Total \* 100;  
PROC PRINT DATA = Inc\_agregare;  
 BY Sport;  
 ID Sport;  
 VAR Sport Vanzari Total Procent;  
 TITLE "Contributia in totalul vanzarilor pe categorii";  
RUN;**

**6. Folosirea opţiunii IN= pentru urmărirea şi selectarea observaţiilor**

Opţiunea IN= asociată unui set de date poate fi folosită pentru a urmări care din seturile de date de intrare a contribuit cu valori la fiecare observaţie din setul de date nou creat. Se utilizează cu precădere pentru a selecta observaţiile care au sau nu corespondent în cadrul unui fuziuni.

Opţiunea IN= se specifică după numele fiecărui set de date şi este urmată de un nume de variabilă temporară ales de utilizator (aceasta nu va fi inclusă în setul de date rezultat). Variabilele temporare definite prin IN= iau valoarea 1 (adevărat) dacă setul de date la care se referă contribuie la observaţia curentă din setul de date nou şi 0 (fals) în caz contrar. Deoarece acestea nu fac parte dintr-un set de date, se va folosi declaraţia PUT pentru a le lista.

***Exerciţiul 7:*** O firmă are un număr de colaboratori care lucrează pe diferite categorii de posturi şi care sunt remuneraţi în funcţie de orele lucrate. În săptămâna 21-27 ianuarie, numai o parte dintre aceştia au lucrat. Codul de mai jos evidenţiază folosirea opţiunii IN= în cazul fuziunii seturilor de date cu informaţii despre colaboratori (Cod, Nume) şi a orelor lucrate (Cod, Categorie, Ore).

**\*\*\*\*\* Utilizarea optiunii IN= ;   
data colaboratori;  
 INPUT Cod Nume$;  
datalines;  
1 Irimia  
2 Popa  
4 Radu  
5 Ionita  
7 Stoicea  
;  
data ore\_lucrate;  
 INPUT Cod Categorie$ Ore;  
datalines;  
1 A 37  
4 B 44  
5 A 32  
9 B 54  
;  
data nou;  
 merge colaboratori (IN=inColaboratori)  
 ore\_lucrate (IN=inOre);  
 by Cod;  
 file print;   
 /\* directionează iesirea care este produsă de PUT   
 în acelasi fisier ca iesirea care este produsă de DATA \*/**

**put Cod= inColaboratori= inOre= Nume= Categorie= Ore=;  
RUN;**

***Exerciţiul 8:*** Pornind de la seturile de date de la exerciţiul anterior, să se afişeze lista colaboratorilor care au lucrat în săptămâna 21-27 ianuarie, indicând şi onorariul lor, ştiind că cei care lucrează pe poziţii din categoria A sunt platiţi cu 70 RON/oră, iar cei din categoria B cu 50 RON/oră.

**data onorarii;  
 merge colaboratori (IN=inColaboratori)  
 ore\_lucrate (IN=inOre);  
 by Cod;   
 if inColaboratori and inOre;  
 if Categorie eq "A" then Onorariu = 70\*Ore;  
 else if Categorie eq "B" then Onorariu = 50\*Ore;  
RUN;   
title 'Onorarii datorate colaboratorilor in perioada 21-27 Ianuarie';  
Proc PRINT data=onorarii;  
RUN;**

**De lucru!!!!**

Pornind de la exemplul de precedent, prin intermediul unei proceduri de fuziune, identificaţi care colaboratori au lucrat în perioada 21-27 ianuarie, dar ale căror detalii de identificare nu se regăsesc în setul de date corespunzător.

Pentru aceştia, recalculaţi şi afişaţi din nou onorariile datorate colaboratorilor, ştiind că atunci cînd se depăşeşte nivelul de 40 de ore pe săptămână, orele peste acest nivel se plătesc dublu.

**B. Proceduri specifice SQL**

Se pot utiliza instrucţiuni SQL în programele SAS pentru a crea, citi şi modifica seturi de date. Există două modalităţi de bază prin care se poate utiliza SQL împreună cu SAS:

* încorporarea de instrucţiuni SQL complete în procedura SQL;
* încorporarea declaraţiei WHERE pentru a selecta rânduri în secţiunile de date şi proceduri SAS.

Ambele variante sunt disponibile cu Base SAS şi nu necesită software adiţional.

***Sintaxa:***

**PROC SQL**;

*comandă \_sql;*

**QUIT***;*

De reţinut că *comandă\_sql* poate fi orice comandă de tipul ALTER, CREATE, DELETE,

DESCRIBE, DROP, INSERT, SELECT sau UPDATE având la sfârşit seminul “;”. Pot fi definite oricâte comenzi SQL într-o procedură SQL. Este o bună practică să se plaseze QUIT la finalul declaraţiei, nefiind însă obligatoriu.

***Exerciţiul 9:*** O companie care comercializează aparate de aer condiţionat are filiale în diferite judeţe ale ţării. Sunt disponibile date referitoare la numărul de aparate vândute în anul anterior (cantitate) şi la agenţii comerciali pe care îi are fiecare filială. Codul SAS de mai jos exemplifică folosirea diferitelor tipuri de joncţiuni folosind procedura SQL.

**\* Jonctiuni cu proceduri specifice SQL;  
Data Filiale;  
Input ID\_fil Filiala$ Cantitate;  
cards;  
10 Cluj 150  
20 Iasi 170  
30 Timis 350  
40 Dolj 140  
50 Sibiu 180  
;  
run;   
Data Personal;  
Input ID\_ang Nume$ ID\_fil;  
cards;  
1 Popescu 30  
2 Matei 10  
3 Avram 10  
4 Costache 10**

**5 Ionescu 40**

**6 Enache 40**

**7 Anton 40**

**8 Codreanu**

**9 Iftene 50  
;  
run;   
\*INNER JOIN;  
PROC SQL;  
Create table inner\_join as  
Select \* from Filiale as f, Personal as p  
where f.ID\_fil = p.ID\_fil;  
QUIT;  
\*Cod alternativ;  
PROC SQL;  
Create table inner\_join1 as  
Select \* from Filiale as f inner join Personal as p  
On f.ID\_fil = p.ID\_fil;  
QUIT;  
\*LEFT JOIN;  
PROC SQL;  
Create table left\_join as  
Select \* from Filiale as f left join Personal as p  
On f.ID\_fil = p.ID\_fil;  
QUIT;   
\*RIGHT JOIN;  
PROC SQL;  
Create table right\_join as  
Select \* from Filiale as f right join Personal as p  
On f.ID\_fil = p.ID\_fil;**

**QUIT;**

**\*FULL JOIN;**

**PROC SQL;**

**Create table full\_join as**

**Select \* from Filiale as f full join Personal as p  
On f.ID\_fil = p.ID\_fil;**

**QUIT;  
\*Utilizarea functiei SQL COALESCE care returneaza primul argument nenul;  
proc SQL;  
select id\_ang, f.id\_fil, coalesce (p.ID\_ang, f.ID\_fil,5) as ID from Filiale as f full join personal as p on f.id\_fil = p.id\_fil;  
QUIT;**

***Exerciţiul 10:*** Un producător de echipamente sportive doreşte să trimită un reprezentant de la vânzări pentru a contacta toţi clienţii care nu au plasat comenzi în primul trimestru al anului. Se cere: a) să se creeze, cu ajutorul procedurilor SQL, două tabele SAS conţinând datele clienţilor şi respectiv valoarea comenzilor plasate de aceştia în primul trimestru; b) un raport care listează clienţii care trebuie contactaţi.

**\*Proceduri specifice SQL;   
LIBNAME ad\_data '/home/nume.prenume';  
PROC SQL;  
 CREATE TABLE ad\_data.client  
 (CodClient num,  
 Nume char(17),  
 Adresa char(20));  
 INSERT INTO ad\_data.client  
 VALUES (101, 'Murphy''s Sports ', '115 Main St. ')  
 VALUES (102, 'Sun N Ski ', '2106 Newberry Ave. ')  
 VALUES (103, 'Sports Outfitters', '19 Cary Way ')  
 VALUES (104, 'Cramer & Johnson ', '4106 Arlington Blvd.')  
 VALUES (105, 'Sports Savers ', '2708 Broadway ');  
 TITLE 'Datele despre clientii sport';  
 SELECT \*  
 FROM ad\_data.client;  
 CREATE TABLE ad\_data.comenzi  
 (CodClient num,  
 Valoare num );  
 INSERT INTO ad\_data.comenzi  
 VALUES ( 102, 562)  
 VALUES ( 104, 254)  
 VALUES ( 102, 1242)  
 VALUES ( 101, 3497)  
 VALUES ( 102, 385);  
 TITLE 'Datele despre comenzile clientilor sport';  
 SELECT \*  
 FROM ad\_data.comenzi;   
PROC SQL;  
CREATE TABLE clienti\_fara\_comenzi as  
SELECT \* from ad\_data.Client as x left join ad\_data.comenzi as y  
On x.CodClient = y.CodClient  
WHERE y.CodClient is null;  
TITLE 'Clienti sport fara comenzi in primul trimestru';  
 SELECT \*  
 FROM clienti\_fara\_comenzi;   
QUIT;**

**De lucru!!!!** Folosind MERGE, să se creeze un raport similar celui de la punctul b) al exerciţiului 10. Se vor folosi seturile de date permanente create în exerciţiul 10.

**C. Lucrul cu masive în SAS**

Masivele sunt utile în SAS atunci când se doreşte repetarea aceleiaşi secvenţe de paşi pentru mai multe variabile. Un masiv reprezintă un grup ordonat de elemente similare. În SAS, acesta reprezintă un grup de variabile, toate de tip numeric sau de tip caracter. Pot fi variabile existente în setul de date cu care se lucrează sau variabile noi.

***Sintaxa simplificată pentru un masiv unidimensional:***

**ARRAY *nume*(n) $ *lista\_variabile***;

unde:

* *nume* = numele masivului;
* n = numărul de elemente;
* $ = se foloseşte în cazul în care sunt referite variabile de tip caracter, de obicei când acestea nu au fost definite;
* parantezele rotunde pot fi înlocuite cu paranteze pătrate sau acolade.

Această sintaxă poartă denumirea de masiv explicit, în care numărul de variabile este specificat la definire. Pentru a referi o variabilă din masiv se va folosi sintaxa *nume(index*), indexul unui masiv începând de la 1. În practică, există o multitudine de situaţii unde masivele își pot demonstra utilitatea, două dintre acestea fiind exemplificate în continuare.

1. **Executarea de calcule repetitive**

În această situaţie, masivele ajută la creşterea eficienţei în programare, mai ales atunci când se lucrează cu un număr mare de variabile.

***Exerciţiul 11:*** Se doreşte realizarea unui sondaj prin intermediul căruia să se măsoare gradul de satisfacţie al cursurilor din domeniul economic oferite de o platformă de *e-learning*. Respondenţii pot acorda calificative de la 1 la 5, iar în situaţia în care nu au urmat un curs sau nu doresc să acorde un calificativ, se va introduce valoarea 0. Au fost culese şi date referitoare la oraşul de reşedinţă al repondenţilor şi vârsta acestora. Se cere să se înlocuiască valoarea 0 cu valoarea lipsă pentru toate variabilele care conţin calificative.

**\*Calcule repetitive cu masive;  
DATA cursuri;  
 INFILE '/home/nume.prenume/Cursuri.txt';  
 Length Oras $9;  
 INPUT Oras$ Varsta ECON MRKT FINA CONT STAT MATE INFO;  
ARRAY curs (7) ECON MRKT FINA CONT STAT MATE INFO;  
 DO i = 1 TO 7;  
 IF curs(i) = 0 THEN curs(i) = .;  
 END;  
RUN;  
PROC PRINT DATA = cursuri;  
 TITLE 'Sondaj cursuri economice';  
RUN;**

**Atenţie!!!** Observaţi că membrii masivului de la curs(1) la curs(7) nu vor face parte din setul de date, dar variabila “i” este inclusă în acesta. Folosiţi o opţiune potrivită pentru a o îndepărta din setul de date.

**Folosirea abrevierilor pentru numele de liste de variabile**

O listă abreviată de nume de variabile poate fi plasată în orice expresie unde apare şi o listă de variabile specificate în detaliu. Ca şi parametri ai funcţiilor, listele abreviate vor fi precedate de cuvântul cheie OF, spre exemplu, SUM (OF val1-val12). Există trei tipuri de liste abreviate, și anume:

* **Liste cu valori numerice**, în care variabilele încep cu aceleaşi caractere şi se termină cu numere consecutive ( **INPUT Tip1- Tip8;** ).
* **Listele cu domeniu denumit** depind de ordinea internă, sau de poziţia, variabilelor în setul de date. Spre exemplu, lista abreviată **s -- p** referă variabilele care au fost definite prin declaraţia **INPUT s t o p**; .
* **Listele cu nume speciale**, \_ALL\_ , \_CHARACTER\_ , \_NUMERIC\_ , pot fi plasate orinde dorim să specificăm fie toate variabilele, fie numai pe cele numerice sau de tip caracter (SUM(OF \_NUMERIC\_)).

***Exerciţiul 12:*** Se cere ca pornind de la datele de la Exerciţiul 11, să se înlocuiască toate valorile 0 cu valoarea lipsă, fără a afecta datele originale, prin crearea unor noi variabile denumite Curs1-Curs7. Totodată, se cere să se determine media punctajelor acordate de fiecare repondent.

**\*\*\*\*\*Aplicatii cu liste abreviate si masive;  
DATA cursuri;  
 INFILE '/home/nume.prenume/Cursuri.txt';  
 Length Oras $9;  
 INPUT Oras$ Varsta ECON MRKT FINA CONT STAT MATE INFO;  
 ARRAY nou (7) Curs1 - Curs7;  
 ARRAY vechi (7) ECON -- INFO;  
 DO i = 1 TO 7;  
 IF vechi(i) = 0 THEN nou(i) = .;  
 ELSE nou(i) =vechi(i);  
 END;  
 MedieCalif = MEAN (OF Curs1 - Curs7);  
 RUN;  
PROC PRINT DATA = cursuri;  
 TITLE 'Sondaj cursuri economice cu media calificativelor';  
RUN;**

***Exerciţiul 13:*** Au fost memorate răspunsurile date de studenţi la un test. Se cere să se înlocuiască toate răspunsurile care nu corespund caracterelor a, b, c, d (litere mari sau mici) cu valoarea lipsă şi să se transforme toate valorile variabilelor de tip caracter în litere mari.

**\*Aplicatii cu nume speciale de liste SAS;  
libname ad\_data "/home/nume.prenume";  
DATA raspunsuri;  
 SET ad\_data.RASPUNSURI;  
 ARRAY var\_caracter {\*} \_character\_;  
 DO i = 1 TO dim (var\_caracter);  
 IF var\_caracter(i) NOT IN ('a' 'b' 'c' 'd' 'A' 'B' 'C' 'D') then  
 call missing(var\_caracter(i));  
 ELSE var\_caracter(i)= upcase (var\_caracter(i));  
 END;  
DROP i;  
RUN;  
PROC PRINT DATA = raspunsuri;  
 TITLE 'Raspunsurile studentilor la test';  
RUN;**

**Atenţie!!!** A fost definit un masiv fără a specifica numărul de elemente **(prin semnul \*)** şi care conţine **doar variabilele de tip caracter**. La parcurgerea masivului se foloseşte funcţia **dim ()** pentru a determina numărul de elemente ale acestuia. **Call missing** reprezintă o rutină prin care se asignează o valoare lipsă oricărei variabile din lista de argumente.

1. **Transformarea structurii seturilor de date**

Transformarea structurii seturilor de date presupune schimbarea perspectivei asupra datelor, spre exemplu atunci când dorim să creăm observații multiple din observații singulare sau viceversa. Aceste restructurări sunt utile la numărarea frecvențelor, la procesarea folosind variabile de grupare sau ca necesitate a anumitor reprezentări grafice sau analize statistice.

Transformarea cu ajutorul masivelor este o alternativă mai flexibilă față de folosirea procedurii SAS TRANSPOSE. Exemplele următoare ilustrează crearea unui set de date cu mai multe observații per subiect (set de date lung) dintr-un set de date cu o singură observație per subiect (set de date lat).

***Exerciţiul 13:*** O companie dorește să întocmească un raport privind nivelurile de competență relevante ale fiecărui angajat în cunoașterea limbilor engleză (E), franceză (F), spaniolă (S) și germană (G), nivelurile de competență variind de la A la C. Se cere să se determine câți angajați cunosc cele patru limbri străine la fiecare nivel.

**\*Transformarea seturilor de date cu masive;  
DATA competente;  
INPUT ID CO1 $ CO2 $ CO3$;  
datalines;  
01 EA GA SB  
02 FA EB EB   
03 SA GB EC  
04 EA FA GB  
05 SB EB GB  
;  
RUN;  
\*varianta fara folosirea masivelor;  
DATA competente\_nou;  
 SET competente;  
 CO=CO1;  
 IF not missing(CO) THEN OUTPUT;  
 CO=CO2;  
 IF not missing(CO) THEN OUTPUT;  
 CO=CO3;  
 IF not missing(CO) THEN OUTPUT;  
 KEEP ID CO;  
RUN;  
\*varianta cu folosirea masivelor;  
DATA competente\_nou;  
 SET competente;  
 ARRAY COARRAY[3] CO1-CO3;  
 DO i=1 to 3;  
 CO=COARRAY[i];  
 IF not missing (CO) THEN OUTPUT;  
 END;  
 KEEP ID CO;  
RUN;  
\*Determinarea numerului de cunoscatori de limbi straine per nivel;  
Title "Numarul de angajati pentru fiecare nivel de competenta in limbile straine";  
PROC FREQ DATA=competente\_nou;  
 TABLES CO /NOCUM;  
RUN;**

***Exerciţiul 14:*** De-a lungul anului 2018 trei echipe de voluntari au participat la plantarea de puieți într-o pepinieră, în patru etape. Se cunoaște numărul de puieți platați de fiecare echipă în fiecare etapă. Se cere să se determine numărul mediu de puieți plantați în 2018 de către fiecare echipă și să realizeze un raport al plantărilor de puieți în anul 2018, per echipă și per total .

**DATA echipe;  
INFILE '/home/nume.prenume/Echipe.txt';  
 INPUT CODE $ A1 A2 A3 A4;  
 Medie = MEAN (of A1 - A4);  
RUN;**

**DATA echipe\_timp;  
 SET echipe;  
 ARRAY A [4];  
 DO timp = 1 TO 4;  
 Puieti = A[timp];  
 OUTPUT;  
 END;  
 KEEP CODE timp puieti;  
RUN;**

**Title "Plantarile de puieti in anul 2018 per echipa si per total";  
PROC PRINT DATA = echipe\_timp noobs  
 sumlabel='Total #byval(CODE) ' grandtotal\_label='Total General';  
 BY CODE;  
 SUM puieti;  
 VAR timp puieti;  
 RUN;**

**Probleme propuse**

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.
2. 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.

1. Un grup de studenți și-a propus să întocmească un plan pentru cărțile citite în decursul unui an, stabilind ținte pentru cărțile din domeniile beletristică și științe. În seturile de date „tinta” și „citit”, create cu prin codul sursă de mai jos, au fost memorate datele privind planurile studenților, respectiv ceea ce au reușit să citească în acel an.

|  |  |
| --- | --- |
| **data tinta; INPUT ID Nume $ tinta\_Bel tinta\_Sti; datalines; 1 Andrei 10 3  2 Mircea 25 2 3 Iulia 20 4 4 Daniela 30 2  5 Mihai 15 5 ; data citit; INPUT ID citit\_Bel citit\_Sti; datalines; 1 9 4 2 18 2 4 22 0 5 12 4  6 16 3 ;** | Figura 4. Raportul cărților citite de studenți |

Se cere să se realizeze un raport similar celui din figura 4.