**Seminar 1 Programare SAS**

**Crearea seturilor de date SAS și folosirea formatelor**

Pe parcursul următoarelor seminarii se va lucra cu SAS Studio, care este complet accesibil prin intermediul navigatoarelor Web si nu necesită nicio instalare. Se vor folosi conturi de studenți create conform instrucțiunilor de aici: <http://support.sas.com/software/products/ondemand-academics/#s1=2>

Pentru exemplificare se va considera un cont de forma nume.prenume.

Pentru conectarea la SAS Studio, click pe : <https://odamid.oda.sas.com/SASLogon/login?service=https%3A%2F%2Fodamid.oda.sas.com%2FSASStudio%2Fj_spring_cas_security_check>

1. **Citirea datelor din fişiere flat**

Datele din fişierele flat pot fi regăsite sub formă de fişiere externe sau plasate ca linii de date direct în programele SAS. Atunci când încearcă să citească date din fişiere flat externe, SAS caută fişiere care au valorile datelor separate prin delimitatori sau coloane fixe. Datele trebuie salvate în fişiere de tip *txt* sau *csv* dacă delimitatorul este virgulă. Alte tipuri de delimitatori sunt spaţiile libere, tab-urile, semnul punct şi virgulă etc.

Accesarea surselor de date într-un program SAS semnifică trecerea unui fişier de date flat printr-un proces de conversie în interiorul unei secţiuni de date, având ca rezultat un set de date SAS. Rezultatele pot fi salvate ca fişiere externe sau exportate ca alte tipuri comune de fişiere precum *html, xls, txt, mdb* etc. Procesul de citire a fişierelor de date flat este prezentat în figura 1.

Figura 1. Accesare surselor de date externe

Cea mai uşoară modalitate de a citi fişiere externe este folosirea metodei **input** pentru valorile care sunt separate prin unul sau mai multe spaţii. Aceasta reprezintă metoda cea mai frecvent folosită pentru citirea datelor delimitate prin separatori în SAS. Considerând că avem un fişier numit produse.txt care conţine numele, preţul şi categoria a cinci produse, fiecare având valoarea separată printr-un spaţiu, acesta ar avea următoarea formă:

hartie 2 birotica

creione 10 birotica

apa 6 alimente

ceai 8 alimente

tricou 100 haine

Presupunând că fişierul este stocat pe server în rădăcina utilizatorului, programul SAS care citeşte date din acest fişier este:

**data produse;**

**infile '/home/nume.prenume/produse.txt';**

**input Nume $ Pret Categorie $;**

**run;**

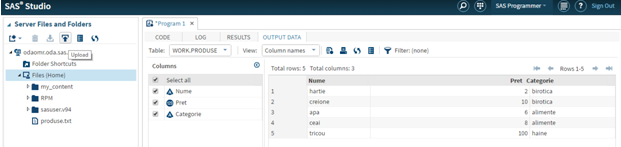
****

Figura 2. Interfaţa SAS Studio – importul fisierului txt

Setul de date SAS care va fi creat poartă numele de *produse*, iar fişierul de date de intrare este specificat folosind comanda *infile*. Declaraţia *input* asociază datele cu nume de variabilă. Dacă o variabiă este de tip caracter, numele acesteia se va încheia cu semnul $.

Atunci când lucrăm cu valori care sunt separate prin virgulă, se pot folosi fişiere *txt* sau *csv*. În oricare dintre situaţii, comanda *infile* trebuie să aibă opţiunea *dsd* care schimbă delimitatorul din spaţiu, în virgulă. De asemenea, dacă există două virgule una după alta, va considera că este o valoare lipsă, fără a fi nevoie ca între virgule să existe un spaţiu. O altă proprietate a acestei opţiuni este aceea că, pentru variabilele caracter, valorile pot fi specificate între ghilimele în fişierul sursă.

Pentru fişierul de date:

hartie,2,birotica

creione,10,birotica

apa,6,alimente

ceai,8,alimente

tricou,100,haine

Se vor folosi următoarele declaraţii pentru a obţine setul de date din fişierul *produse2.txt*:

**data produse;**

**infile '/home/nume.prenume/produse2.txt' dsd;**

**input Nume $ Pret Categorie $;**

**run;**

Pentru orice alt tip de delimitatori, declaraţia *infile* trebuie folosită împreună cu opţiunea *dlm* sau *delimiter*. Au aceeaşi semnificaţie, prima fiind abrevierea celeilalte.

Exemplu: **infile '/home/nume.prenume/produse3.txt' delimiter='/';** sau

**infile '/home/nume.prenume/produse4.txt' dlm='/'**;.

**data produse;**

**infile '/home/nume.prenume/produse3.txt' dlm='/';**

**input Nume $ Pret Categorie $;**

**run;**

Dacă valorile datelor nu sunt stocate într-un fişier extern, ele pot fi plasate în cadrul programului, folosind declaraţia *datalines*.

**data produse;**

**input Nume $ Pret Categorie $;**

**datalines;**

**hartie 2 birotica**

**creione 10 birotica**

**apa 6 alimente**

**ceai 8 alimente**

**tricou 100 haine;**

Cel de-al doilea tip de fişiere externe pe care SAS poate să le citească sunt cele care conţin date în coloane cu lăţime fixă. Pentru acestea există două modalităţi de citire: coloane de intrare (*column input*) şi intrări formatate (*formatted input*). Avantajul folosirii datelor în coloane cu lăţime fixă este acela că datele pot fi citite în orice ordine este necesară, iar valorile lipsă pot fi specificate prin tot atâtea spaţii libere cât are şi lăţimea coloanei.

Considerăm fişierul *produse4.txt*, care conţine următoarele valori:

hartie 2 birotica

creioane10 birotica

apa 6 alimente

ceai 8 alimente

tricou 100haine

Aşa cum se poate observa, numele are 8 caractere, preţul 3, iar categoria 8.

Metoda 1: Coloane de intrare foloseşte declaraţia *input* specificând pentru fiecare variabilă numele său, semnul $ pentru variabilele caracter, precum şi poziţia de început şi de sfârşit a fiecărei variabile în linia de date.

**data produse;**

**infile '/home/nume.prenume/produse4.txt';**

**input Nume $ 1-8**

**Pret 9-11**

**Categorie $ 12-19;**

**run;**

Metoda 2: Intrări formatate poate citi diferite tipuri de formate, cum ar fi monedă (numere având semnul dolar sau euro), numerele cu zecimale, date în diferite formate. Declaraţia *input* are umătoarele opţiuni: semnul @ urmat de poziţia de început a variabilei în linia de date, numele variabilei, formatul variabilei, dacă acesta este necesar, precum şi lăţimea coloanei asociată variabilei.

Considerăm datele anterioare la care adăugam semnul dolar în fața prețurilor. Datele pot fi citite direct din program sau din fișierul produse5.

**data produse;**

**/\*infile '/home/nume.prenume/produse5.txt'; \*/**

**input @1 Nume $8.**

**@9 Pret dollar4.**

**@13 Categorie $8.;**

**format Pret dollar6.0;**

**datalines;**

**hartie $2 birotica**

**creioane$10 birotica**

**apa $6 alimente**

**ceai $8 alimente**

**tricou $100haine**

**run;**

**2. Crearea seturilor de date permanente**

Este important să se poată crea seturi de date permanente, deoarece astfel ele pot fi accesate oricând de pe calculatorul personal sau de pe server fără a rula cod SAS. Sunt economisite astfel resurse de calcul și timp, în special atunci când se lucrează cu seturi mari de date care sunt folosite de mai multe ori în proiecte SAS.

Atunci când se folosește declarația *data* un set de date temporar este creat. El va fi șters când sesiunea curentă este închisă. Implicit este stocat temporar într-o bibliotecă referită (*libref*) numită *Work*. Pentru ca setul de date să fie stocat permanent trebuie ca biblioteca în care se salvează să fie specificată explicit.

Biblioteca în care se vor stoca datele este specificată în declarația *libname*. În exemplul următor este creată biblioteca *produse*, iar setul de date rezultat este *date\_test*.

**libname produse '/home/nume.prenume';**

**data produse.date\_test;**

**input Nume $ 1-8**

**Pret 9-11**

**Categorie $ 12-20;**

**datalines;**

**hartie 2 birotica**

**creioane10 birotica**

**apa 6 alimente**

**ceai 8 alimente**

**tricou 100haine**

**run;**

După rularea acestui script, fișierul persistent *date\_test.sas7bdat* va fi creat pe server (așa cum se vede în figura 4), iar biblioteca Produse va fi creată împreună cu tabela de date SAS *date\_test* (figura 3). Acestea din urmă sunt temporare.

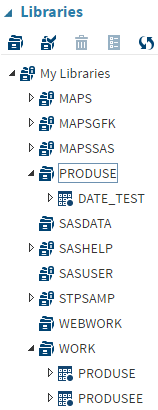


Figura 3. Exemplu de bibliotecă SAS

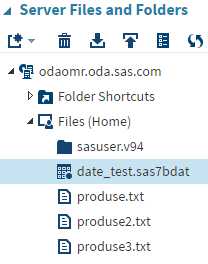


Figura 4. Exemplu de fișier persistent creat în SAS Studio

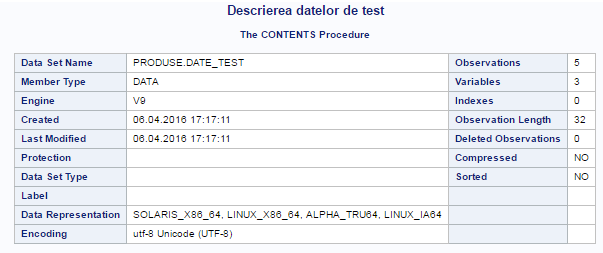
Pentru vizualizarea informațiilor referitoare la un set de date SAS, se pot folosi procedurile CONTENTS și PRINT. *Proc contents* afișează descrierea datelor, în timp ce *proc print* afișează datele. Următorul exemplu folosește procedura *contents*.

**title "Descrierea datelor de test";**

**proc contents data=produse.date\_test;**

**run;**

Rezultatul rulării acestui script este un raport ce conține numărul de variabile și de observații, numele, tipul și dimensiunile variabilelor și alte informații referitoare la setul de date din fișier. Raportul este prezentat în figura 5.



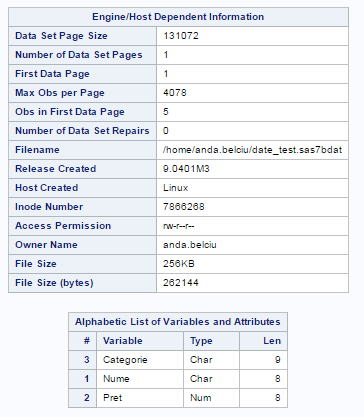


Figura 5. Exemplu procedură *proc contents*

Procedura *print* poate fi folosită într-un program astfel:

**title "Datele din fisierul date\_test";**

**proc print data=produse.date\_test;**

**run;**

Rezultatul obținut în urma rulării acestui script este un raport SAS ce conține numele variabilelor și observațiile. Acest raport este obținut în figura 6.



Figura 6. Exemplu de aplicare a procedurii *proc print*

**3. Formate și etichete**

Formatele (sau formatările) și etichetele sunt folosite pentru a evidenția informația într-un mod mai atractiv și pentru a explicita niște denumiri abreviate. Etichetele pot fi create în secțiunile DATA sau PROC.

De exemplu, se vor adăuga etichete pe un set de date referitoare la notele obținute de studenți. Se folosește declarația LABEL urmată de numele variabilelor, semnul = și eticheta scrisă între ghilimele simple sau duble. O etichetă poate avea până la 256 de caractere.

**DATA note\_an3;**

**input cod $ Nota1-Nota3;**

**label cod = 'Cod Student'**

**Nota1 = 'Nota psw'**

**Nota2 = 'Nota SGBD'**

**Nota3 = 'Nota finante';**

**datalines;**

**1 10 10 8**

**2 8 7 8**

**3 6 4 4**

**;**

Dacă etichetele sunt folosite în secțiunea DATA ele vor rămâne asociate variabilelor respective. Dacă însă sunt folosite în secțiunea PROC, atunci ele vor fi valabile doar în cadrul secțiunii respective.

Formatele sunt folosite pentru a îmbunătăți modul în care datele sunt afișate. Există formate predefinite SAS care pot fi folosite de exemplu pentru afișarea valutei sau pentru a pune virgula de ordinul miilor la numere. De asemenea, pot fi construite formate de către utilizator, care apoi să fie folosite pentru diferite valori sau grupuri de valori. De exemplu, M, F (valori pentru variabila *gen*) să fie afișate Masculin, Feminin, iar notele sub 4,5 sa fie afișate ca Examen Picat.

Formatele definite de utilizator se creează cu ajutorul procedurii FORMAT. Exemplul de mai sus va fi transpus astfel în cod SAS:

**proc format;**

**value $gen 'M' = 'Masculin'**

**'F' = 'Feminin'**

**other = 'format necoresp.';**

**value nota low-4.4 = 'Examen Picat';**

**4.5-high = 'Examen trecut';**

**run;**

Pentru fiecare format definit de utilizator se precizează cuvântul cheie *value* urmat de semnul dolar pentru variabilele de tip caracter (pentru care se va aplica formatul). Apoi este specificat numele formatului și pentru fiecare valoare posibilă (sau interval de valori) se precizează formatarea. Dacă rămân valori nespecificate, și nu se folosește nici clauza *other*, atunci ele vor rămâne neformatate.

În general formatările sunt valabile doar pentru afișări și nu pentru procesarea datelor. Există însă și câteva excepții. De exemplu, procedura FREQ calculează frecvențele de apariție ale valorilor formatate, dacă este aplicat un format.

În continuare aplicăm formatările *nota* și *gen* pe setul de date *note*. Se observă că, la fel ca la formatările predefinite SAS, atunci când acestea se aplică, numele lor trebuie urmat de punct.

**data note;**

**input nume$ 1-9 nota gen$;**

**datalines;**

**Ionescu 5.6 M**

**Popescu 4.3 F**

**Vasilescu 3.3 b**

**Georgescu 9.2 F**

**Iliescu 10 M**

**Samoila 5.9 M**

**Toma 7 F**

**;**

**title "Date despre exemenul de PSW";**

**proc print data=note;**

**var nume nota gen;**

**format nota nota.**

**gen $gen.;**

**run;**

Rezultatul obținut după aplicarea formatărilor se poate observa în figura 7. Pentru variabila nota, formatul se aplică numai pentru valorile < 4.5, restul observațiilor rămânând nemodificate.

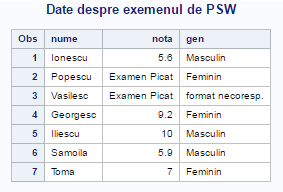


Figura 7. Aplicarea formatelor definite-utilizator

Se cere să se determine numărul de studenţi din fiecare gen şi numărul de studenţi care au formatul necorespunzător. Se va folosi procedura FREQ care determină frecvența de apariţie a valorilor variabilelor. Pentru variabila gen se va aplica formatul definit de utilizator, asftel încât FREQ va număra valorile rezultate după aplicarea formatului. Rezultatul rulării se poate observa în figura 8.

**title "Frecventa de aparitie pentru gen";**

**proc FREQ data=note;  
 TABLES gen /nocum nopercent;  
 FORMAT gen $gen.;  
run;**

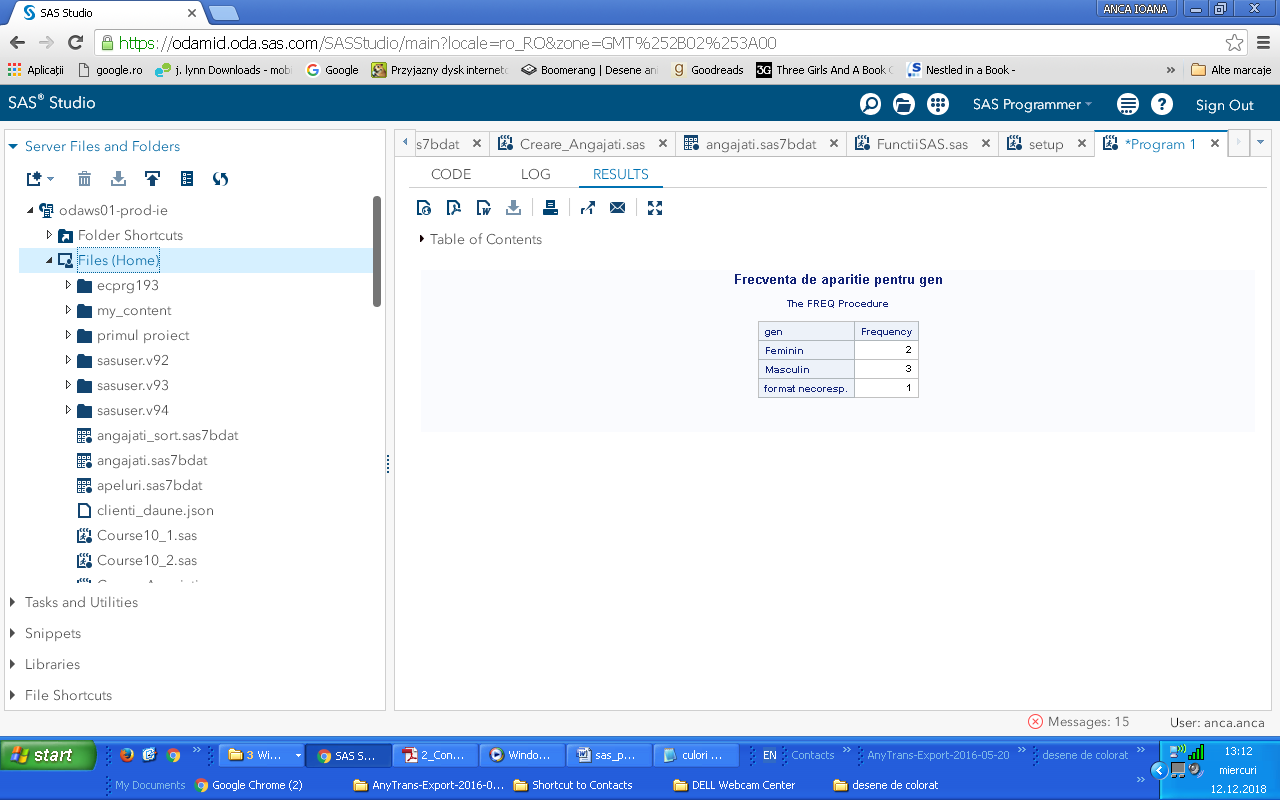
****

Figura 8. Procedura FREQ cu aplicare de formate pentru gen

Să se determine câţi studenţi au picat şi câţi au trecut examenul. Rezultatul rulării se poate observa în figura 9.

**proc format;**

**value nota low-4.4 = 'Examen Picat'**

**4.5-high = 'Examen trecut';**

**run;**

**title "Frecventa de aparitie pentru note";**

**proc FREQ data=note;  
 TABLES nota /nocum nopercent;  
 FORMAT nota nota.;  
run;**

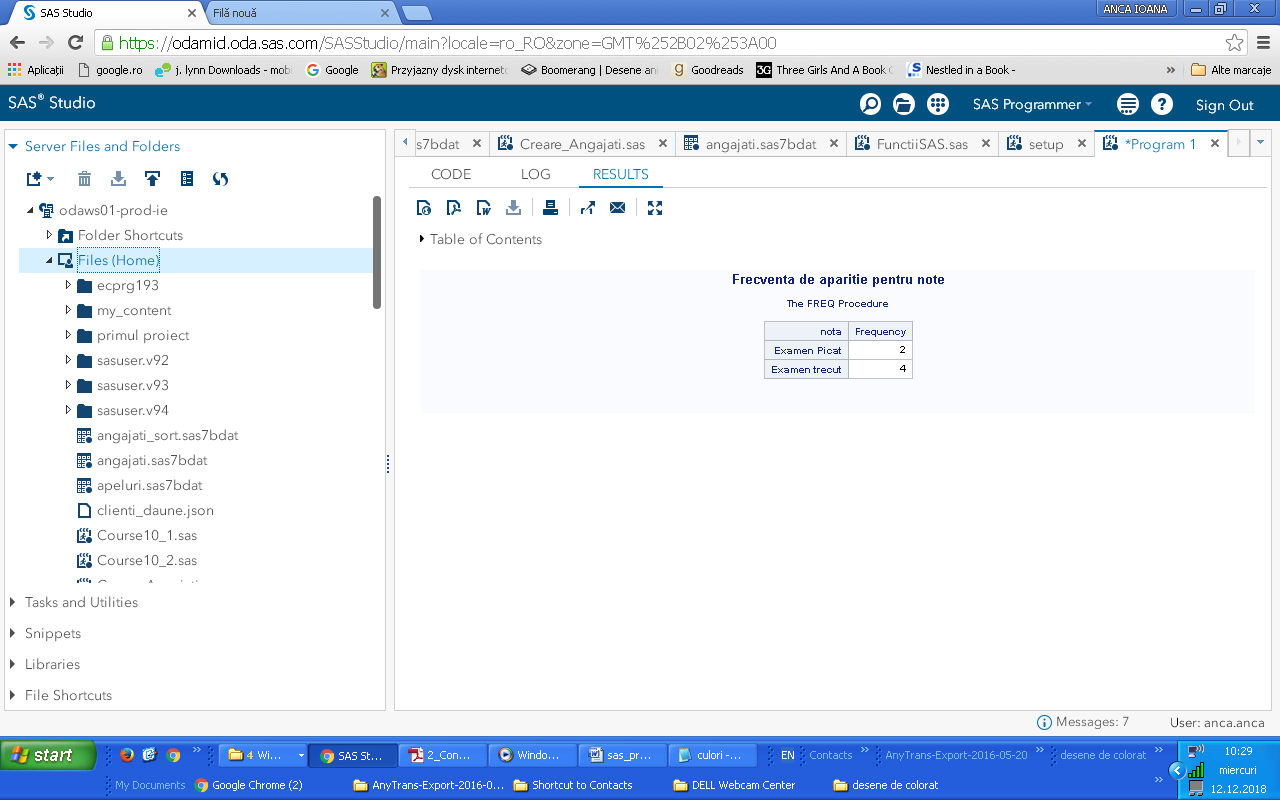


Figura 9. Procedura FREQ cu aplicare de formate pentru nota

Pentru ca formatele să capete persistență, ele trebuie stocate în biblioteci definite prin *libname nume\_biblioteca 'calea director…'* și apoi accesate astfel:

**libname formate '/home/nume.prenume';**

**options fmtsearch=(formate);**

**proc format library = formate;**

**value pl low-10 = 'acceptabil'**

**11-high = 'peste limita';**

**run;**

**proc print data = formate.date\_test1;**

**format pret pl.;**

**run;**

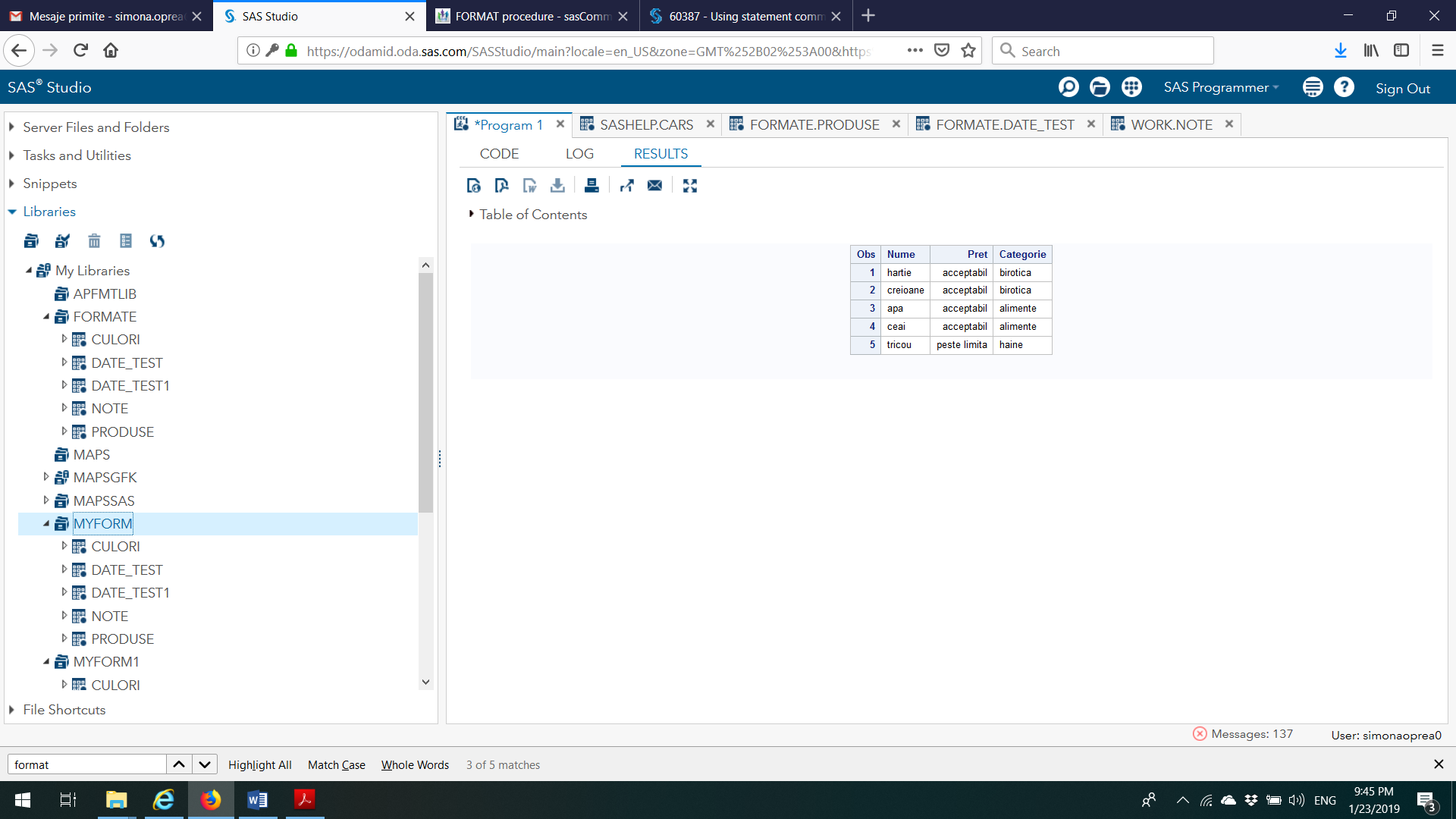


Figura 10. Aplicarea unui format permanent

În cadrul procedurilor SAS pot fi aplicate o multitudine de formate predefinite. În exemplul de mai jos se cere ca preţul să fie afişat având în faţă semnul „$”, iar data să fie afişată în formatul ZZ-LL-AAAA. **Observaţi şi analizaţi modul în care au fost citite datele de intrare şi ce formate de citire au fost folosite!**

**DATA comenzi;  
input iD\_Comanda ID\_Produs :$4. Cantitate Pret Data :mmddyy.;  
datalines;  
102 1053 3 125 05/12/2018  
103 243 25 30.7 05/15/2018  
104 3052 1 106.83 06/21/2018  
105 129 6 3.42 06/24/2018  
;**

**\*fara aplicarea formatelor de afisare;  
proc print data=comenzi;  
run;**

**\*cu aplicarea formatelor de afisare;  
proc print data=comenzi;  
format Pret DOLLAR8.2 Data ddmmyyd10.;  
run;**

**4. Fișiere JSON**

Fișierele JSON (JavaScript Object Notation) reprezintă un format din ce în ce mai popular, folosit pentru interschimburi de date. Este independent de limbajele de programare și are un format ușor de citit de către utilizatori sau de către limbajele de programare. Aceste tipuri de fișiere conțin de obicei date de mari dimensiuni, nestructurate sau parțial structurate. Este asociat adesea cu termenul de Big Data, reprezentând o sursă foarte bună de furnizare a unor astfel de date.

Conținutul unui astfel de fișier este practic alcătuit dintr-o multitudine de fraze în care de obicei informația este căutată în funcție de anumite cuvinte cheie.

Pentru exemplificare vom folosi fișierul *clienti\_daune.json* în care datele referitoare la daunele posesorilor de mașini, apar sub forma următoare:

{

"VIN":"X4ALT2S284AIPQ29AKJ106",

"Tara Producator":"United States",

"Regiune Producator":"North America",

"Marca":"FORD",

"Producator":"FORD MOTOR COMPANY",

"Model":"EXPLORER",

"An fabricatie":2010,

"Data Cerere":"2012-01-06",

"Geocode reprezentanta":"36.153889 -95.992500",

"Reprezentanta":"FORD OF TULSA",

"Zona Reprezentanta":"OK",

"Oras Reprezentanta":"Tulsa",

"Opera'ii realizate":"Replace Alternator",

"Componenta":"ELECTRICAL SYSTEM:ALTERNATOR/GENERATOR/REGULATOR",

"Dauna":"OPERATION:WILL NOT START",

"Cod":"Ah63ANFG224MN2-0974",

"Data vanzarii":"2010-10-07",

"Tara furnizor":"England",

"Furnizor":"Delphi Automotive Systems U K Ltd.",

"Zona Funizor":"Luton",

"Pret manopera":"$444.87",

"Pret Total":"$316.87",

"Valoare Dauna":"$761.74",

"Id\_Client":9194439

},

SAS nu oferă încă posibilitatea importării unor astfel de fișiere, ci doar o procedură destinată exportului în fișiere JSON. Vom extrage din fișierul inițial valoarea daunei și clientul într-un fișier de date SAS, care va fi ulterior exportat tot într-un fișier JSON.

Extragerea datelor se face folosind următorul cod SAS:

**filename daune '/home/nume.prenume/clienti\_daune.json';**

**data daune\_nou;**

**infile daune truncover scanover;**

**input @'Valoare Dauna":"' valoare $15.**

**@'Id\_Client":' idclient 8.;**

**put valoare=;**

**put idclient=;**

**run;**

Căutarea în fișier se face folosind declarațiile *truncover* și *scanover* în funcție de expresiile *Valoare Dauna":"* și *Id\_Client":.* TRUNCOVER îi spune motorului SAS să se oprească din citit atunci când ajunge la sfârșitul unei linii, iar SCANOVER poziționează pointerul în funcție de o condiție de căutare, apoi citește normal din fișier. Rezultatele sunt expuse în figura 11.

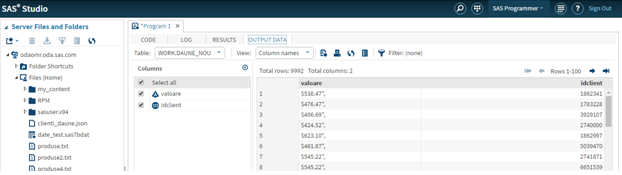


Figura 11. Extragerea datelor din fişierul .json

Pentru a rafina datele preluate din fișierul .JSON este nevoie să edităm interogarea SQL pe care se bazează tabela de date SAS *interogare*. Valoarea daunei nu are o lungime fixă, de aceea ea trebuie *curățată*. Verificăm dacă valoarea extrasă se termină în cifre sau în “,. În cel de-al doilea caz, eliminăm caracterele incorecte. De asemenea, prin condiția **valoare<>'",'**, eliminăm valorile vide. Procedura PROC SQL permite scrierea de cod SQL, în cazul de față, crearea unei tabele pe baza unei interogări. Pentru condiționarea alternativă am folosit clauza CASE.

**PROC SQL;**

**CREATE TABLE WORK.interogare AS**

**SELECT case when substr(valoare,length(valoare)-1,2)='",' then substr(valoare,1,length(valoare)-2) else valoare end as val, idclient FROM WORK.DAUNE\_NOU where valoare<>'",' ;**

**RUN;**

**QUIT;**

Pentru exportul datelor într-un fișier .JSON se folosește procedura JSON pentru care se precizează fișierul în care se face exportul și setul de date SAS de exportat.

**proc json out='/home/nume.prenume/daune\_noi.json';**

**export WORK.interogare;**

**run;**

Se poate observa că în interogarea de mai sus, variabila val este de tip caracter, ceea ce înseamnă că asupra ei nu pot fi efectuate în SAS operaţii numerice. Se doreşte crearea unei noi tabele, îm care variabila valoare să stocheze valori numerice. În acest sens, se va folosi funcţia INPUT pentru a converti o valoare de tip caracter la o valoare numerică şi a stoca valoarea acesteia într-o nouă variabilă. De asemenea, se va folosi formatul de intrare DOLLAR10.2 pentru a indica faptul că datele de intrare au în faţă semnul $ urmat de un număr cu două zecimale.

**DATA interogare1 (drop= val);   
 Set interogare;  
 Valoare = input(val, DOLLAR10.2);  
Run;**

**5. Probleme propuse**

1. Să se definească, folosind datele din fișierul produse.txt, un set de date cu delimitatorul \*.
2. Folosind datele de la a) să se creeze un fișier permanent de date SAS.
3. Fie următoarele date:

Rosu, Galben, Albastru, Verde, Roz, Rosu, Verde, Mov, Maro, Alb

Să se creeze un fişier de date SAS pe baza lor.

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

**e)** Să se aplice formatul definit utilizator pentru fișierul de la punctul c).

**f)** Pornind de fişierul de date “interogare1” de la punctul 4 al seminarului, să se determine câte dosare au dauna sub $500, câte între $500 şi câte peste $800, folosind procedura de calcul a frecvenţelor de apariţie şi un format definit de utilizator.

**Probleme propuse rezolvate**

1. Setul de date se găseşte în fişierul „produse11” din subdirectorul Probleme rezolvate

**data produse;**

**infile '/home/nume.prenume/produse11.txt' dlm='\*';**

**input Nume $ Pret Categorie $;**

**run;**

1. Setul de date se găseşte în fişierul „culori” din subdirectorul Probleme rezolvate

**data culori;  
 infile '/home/nume.prenume/culori.txt';  
 input Culoare $;  
run;**

**proc format;  
value $culoare 'Rosu', 'Galben', 'Albastru'='De baza'  
 'Alb','Negru'= 'Non-culoare'  
 other = 'Derivata';  
run;**

**title "Lista culorilor ca si tip";  
proc print data=culori;  
 format culoare $culoare.;  
run;**

**proc format;  
 value dauna   
 low-499.99 = 'sub $500'  
 500-800 = 'intre $500 si $800'  
 800.01-high ='peste $800';  
run;  
  
title "Numar de dosare de dauna in functie de valoare";  
proc FREQ data=interogare1;  
 TABLES valoare /nocum nopercent;  
 FORMAT valoare dauna.;  
run;**