**SAS PROGRAMMING 2 EĞİTİM NOTLARI**

**Chapter 2 :**

Tabloda, bölüm adı, o bölümdeki çalışan sayısı ve yıllık artış oranı var. Bizden istenen, 5 bölümde, gelecek 3 yıl sonunda ortalama çalışan sayısı ne olacaktır?

**Ds den bir kayıt okunup 2.ds ye yazılması isteniyorsa**, output kullanılıyor.

data forecast;

drop NumEmps;

set prog2.growth;

Year=1;

NewTotal=NumEmps\*(1+Increase);

output;

Year=2;

NewTotal=NewTotal\*(1+Increase);

output;

Year=3;

NewTotal=NewTotal\*(1+Increase);

output;

run;

**Bir ds den birden fazla ds yaratmak için :**

Army, Navy, AirForce,Marines datalarını içeren prog2.military ds miz var. bizden istenen,

Type = Army ise o kaydı Army ds sine yazsın…

data army navy airforce marines;

drop Type;

set prog2.military;

if Type eq 'Army' then   
 output army;  
 else if Type eq 'Naval' then   
 output navy;

else if Type eq 'Air Force' then  
 output airforce;

else if Type eq 'Marine' then

output marines;

run;

output ta :

hangi kayıtların okunacağını, hangi kayıtların yazılacağını, kaç kayıt okunacağı gibi soruların cevaplarını data ve proc step ler altında verebiliyoruz.

Default olarak output a tüm değişkenler yazılır. Bunu drop-keep ile engelleyebiliriz.

**Farklı ds lerde farklı drop-keep ler uygulamak için :**

data army(drop=City State Country Type)

navy(drop=Type)

airforce(drop=Code Type)

marines;

set prog2.military;

if Type eq 'Army' then

output army;

else if Type eq 'Naval' then

output navy;

else if Type eq 'Air Force' then

output airforce;

else if Type eq 'Marine' then

output marines;

run;

**okunacak son kaydın numarasını belirlemek için** : OBS kullanılır.

data army;

set prog2.military(obs=25); // 25. kayda kadar oku!!

if Type eq 'Army' then output;

run;

set prog2.military(obs=MAX); tüm kayıtların okunmasını sağlar

**okunacak ilk kaydın numarasını vermek için :** FIRSTOBS kullanılır.

data army;

set prog2.military(firstobs=11 obs=25); //11.kayıttan başla 25.kayda kadar oku

if Type eq 'Army' then output;

run;

proc print içinde de firstobs ve obs kullanılabiliyor.

proc print data=army(firstobs=2 obs=4);

var Code Airport;

run;

**dışarıdaki bir dosyaya yazmak için** :

**1.yol :** CSVALL kullanılır.THE CSVALL option is new in SAS®9.

ods csvall file='*raw-data-file*';

footnote1 'data: prog2.maysales';

proc print noobs data=prog2.maysales;

format listdate

selldate date9.;

run;

ods csvall close;

kayıtlar “” içinde external file içine kaydedilir.

**2.yol :**

Proc export kullanılır. ??????????

DATA \_NULL\_;

İle geçici bir ds yaratıyoruz.

File ‘......’ oluşturacağımız file ın location ınını belirliyoruz.

Put ile yeni dosyada hangi değişkenlerin var olacağını belirliyoruz

PUT *variable-1 variable-2 … variable-n*;

Boş bir satır için :

Put ;

**Değişkenlere çeşitli formatlar uygulanacaksa;**

PUT *variable-1* : *format-1.   
 variable-2* : *format-2.*

…

*variable-n* : *format-n.*;

data \_null\_;

set prog2.maysales;

file '*raw-data-file*';

put Description

ListDate : date9.

SellDate : date9.  
 SellPrice;

run;

**txt dosyasının içeriğini görmek için :**

proc fslist fileref='*raw-data-file*';  
run;

txt de en üst satıra kolon başlıkları eklemek için :

if \_N\_=1 then // program ilk kez çalıştırılıyorsa ;

put 'Description ' 'ListDate '

'SellDate ' 'SellPrice';

\* kayıt sayısı okumak için : \_N\_ ile yazdırabiliyoruz.

**Dosya sonuna gelip gelmediğimizi görmek için :**

SET *SAS-data-set* END=*variable* <*options*>;

Dosya sonuna geldiysek islast = 1 gelmediysek islast=0 oluyor.

data \_null\_;

set prog2.maysales end=IsLast;

file '*raw-data-file*';

if \_N\_=1 then

put 'Description ' 'ListDate '

'SellDate ' 'SellPrice';

put Description

ListDate : date9.   
 SellDate : date9.   
 SellPrice;

if IsLast=1 then // footnote yazdırıyor

put 'Data: PROG2.MAYSALES';

run;

**default delimiter virgül dür. Ama bunu değiştirebiliriz.**

data \_null\_;

set prog2.maysales end=IsLast;

file '*raw-data-file*' dlm=',';

if \_N\_=1 then

put 'Description,ListDate,'

'SellDate,SellPrice';

put Description

ListDate : date9.   
 SellDate : date9.   
 SellPrice;

if IsLast=1 then

put 'Data: PROG2.MAYSALES';

run;

**Chapter 3 :**

Bu chapter da ;

-pdv içindeki değişkenin sistem tarafından nasıl sıfırlandığı

-yeni değer atanması

-kümülatif değişkenler oluşturulması

Konuları işlenecek

Prog2.daysales içinde günlük satış bilgileri var. Nisan ayının her bir günü SaleDate, her gün ki satış bilgileri SaleAmt değişkenleri altında tutuluyor.

Mağaza müdürü 1 ayın sonundaki kümülatif toplamı görmek istiyor. Mth2Dte

Mth2Dte = Mth2Dte + SaleAmt ile kümülatif toplam hesaplanır. Ancak Mth2Dte in değerinin sistemde bir yerde tutulması gerekiyor.

**Bunun için retain statement** ını kullanıyoruz.

Retain ile :

-data step içindeki işlemler sırasında değişkenin pdv içindeki değerinin korunması sağlanır.

-data step ilk çalıştırıldığında değişkene ilk değer atanır.

-sadece compile aşamasında çalışır.

Retain Mth2Dte 0; // ilk değer ataması yapılmazsa tüm değerler missing gelir.

data mnthtot;

set prog2.daysales;

retain Mth2Dte 0;

Mth2Dte=Mth2Dte+SaleAmt;

run;

Kümülatif toplamları alırken, arada missing değer var ise ne olur ?

**Tüm değerler missing olarak gelir.**

**Bunu engellemek için sum fonksiyonunu** kullanıyoruz.

Mth2Dte = sum (Mth2Dte + SaleAmt); // retain yapmıyor

Sum Statement ı daha kullanışlıdır.

* + işaretinin solundaki değişken yaratılır ya da değer atanır
* Otomatik olarak retain işlemini yapar

data mnthtot2;

set prog2.daysales2;

Mth2Dte+SaleAmt;

run;

**veri grupları için kümülatif toplamlar** :

prog2.empsals altında çalışanların id leri, maaş bilgileri ve departmanları var. Bizden istenen, yeni bir ds yaratılması. Bu ds içiinde her bir bölüme göre ödenen toplam maaş bilgisi var.

Önce datayı gruplamamız gerekiyor.

proc sort data=prog2.empsals out=salsort;

by Div;

run;

**yeni ds de gruplandırma yapmak için** ;

set ve by birlikte kullanılır.

data divsals(keep=Div DivSal);

set salsort;

by Div;

*additional SAS statements*

run;

**Gruplara göre toplam aldırmak için first ve last processing** kullanılıyor.

First.Div = 1 ise o grubun ilk elemanı, Last.Div=1 ise son elemanıdır.

İşleyiş :

Kümülatif toplam değişkenin başlangıç değeri 0 yapılıyor.

Toplam işlemleri yapılıyor.

O grubun son elemanına gelindiğinde output yapılıyor.

data divsals(keep=Div DivSal);

set salsort;

by Div;

if First.Div then DivSal=0;

DivSal+Salary;

if Last.Div;

run;

**önemli :**

iki değişkene göre gruplama yapılması istendiğinde :

primary sort un hangi değişkene göre yapılacağı önemlidir. Compilation aşamasında primary sort a göre first ve last değerleri değişir.

**Chapter 4 :**

Daha önce belli bir şekilde ayrılmamış, değerlerin bitişik olduğu txt dosyasından değerleri okuduk.

Şimdi belli bir delimiter ı olan txt den değerleri okuyacağız. Satır sonunda missing değer varsa ne yapılır?

Data fixed kolonlarda değilse,

Delimiter boşluk ise,

Standart olmayan data varsa, //belli formatı olmayan dataya free format deniliyor.

List input yöntemi kullanılır.

Ayıraçlarımız : boşluk, virgül, tab olabilir.

**Standart olmayan dataları içeriye almak için informat** uygulanır.

INPUT *variable* : *informat*;

Sas datayı ayıraçtan ayıraca okur. Belli kolonlar yoksa, sas informatın uzunluğuna göre okur.

data airplanes;

infile '*raw-data-file*';

input ID $

InService : date9.

PassCap CargoCap;

run;

**Uzunlukları ayarlamak için length kullanılır.** Length ile id değişkeni yaratılmış olur.

data airplanes;

length ID $ 5;

infile '*raw-data-file*';

input ID $

InService : date9.

PassCap CargoCap;

run;

**delimiter olarak virgül kullanılmış ise ;**

data airplanes2;

length ID $ 5;

infile '*raw-data-file*' dlm=',';

input ID $

InService : date9.

PassCap CargoCap;

run;

**satır sonlarında missing değer varsa; //2.seçenek truncover**

missover kullanılır. Missing değer yerine nokta yada boşluk konulur.

data airplanes3;

length ID $ 5;

infile '*raw-data-file*' dlm=',' missover;

input ID $

InService : date9.

PassCap CargoCap;

run;

**Aralarda missing değerler varsa**;DSD option ını kullan.Buna göre ;Virgül default delimiterdır.

data airplanes4;

length ID $ 5;

infile '*raw-data-file*' dsd;

input ID $

InService : date9.

PassCap CargoCap;

run;

infile statement ı içinde : dlm, missover,dsd kullanılır.

PAD????

Datayı içeriye alırken kontrol edebiliriz.

Mesela ;

Tek satırda birden fazla kayıt olabilir,

Karışık olabilir,

Tek kayıt birden fazla satırda olabilir..

Bunları çözmek için ayrı input statementları tanımlamamız gerekir.

**Tek kayıt birden fazla satırda olabilir..**

data address;

length LName FName $ 20

City $ 25 State $ 2

Phone $ 8;

infile '*raw-data-file*' dlm=',';

input LName $ FName $; //1.satır

input City $ State $; //2.satır

input Phone $; //3.satır

run;

**yada ayrı input lar yerine line pointer control kullanabiliriz**.

data address;

length LName FName $ 20 City $ 25 State $ 2 Phone $ 8;

infile '*raw-data-file*' dlm=',';

input LName $ FName $ /City $ State $ /Phone $;

run;

2.yol :

data address;

length LName FName $ 20 City $ 25 State $ 2 Phone $ 8;

infile '*raw-data-file*' dlm=',';

input #1 LName $ FName $

#2 City $ State $

#3 Phone $;

run;

**gelen data karışık olabilir. Bunun için informat uygulamamız gerekir.**

input SalesID $ Location $;

if Location='USA' then

input SaleDate : mmddyy10.

Amount;

else if location='EUR' then

input SaleDate : date9.

Amount : commax8.; // virgül yerine decimal koyar.

**Başarılı olmaz . çünkü kaydı bekletmemiz gerekiyor. Bunun için ;**

**@ single trailing kullanılır. Single trailing ile kayıt bir sonraki input statement’ına kadar tutulur.**

INPUT *var1 var2 var3 … @*;

input SalesID $ Location $ @;

if location='USA' then

input SaleDate : mmddyy10.

Amount;

else if Location='EUR' then

input SaleDate : date9.

Amount : commax8.;

Retain - @ farkı :

Retain : farklı koşullara bağlı olarak, 2. 3. run’dan sonra bile değişkenin değeri tutulur

@ : bir sonraki input’a kadar değişkenin değerini tutar

**Tek satırda birden fazla kayıt olabilir,**Double trailing @@ kullanılır.

data work.retire;

length EmpID $ 6;

infile '*raw-data-file*';

input EmpID $ Contrib @@;

run;

**hiyerarşik datayı okumak için :**

bizden hiyerarşik file ı okuyup sas ds haline çevirmemiz,

detay bilgilerinin özetini almamız istenebilir.

Örneğimize göre :

E= employee –header bilgisi

D= dependence- detail bilgisi

C= child S = eş

Single trailingde, hala aynı kayıttasın, diğer değişkenlere bakılıyor.

Retain de bir sonraki kayıttasın, değişkenin değeri tutuluyor.

Örnekte : aylık sigorta ödemeleri için ;

Kişi kendi için sigorta ücreti ödemiyor,

Eşi için 50$, çocuğu için 25$ ödüyor.

İstenen her bir işçinin toplam ne kadar sigorta ücreti ödediği bilgisi.

Burada tüm işlemler id ye göre yapılacağından id retain edilir.

data work.insurance(drop=Type DepName Relation);

length Type $ 1 ID $ 6 DepName $ 20

Relation $ 1;

retain ID;

infile '*raw-data-file*' dlm=':';

input Type $ @;

if Type='E' then do;

if \_N\_ > 1 then output; // ilk kayıt ise

input ID $;

Deduct=0;

end;

else do;

input DepName $ Relation $;

if Relation='C' then Deduct+25;

else Deduct+50;

end;

run;

**son kayıtsa, devam etmesin anlamında**

data work.insurance(drop=Type DepName Relation);

length Type $ 1 ID $ 6 DepName $ 20 Relation $ 1;

retain ID;

infile '*raw-data-file*'

dlm=':' end=LastRec; /**/önemli**

input Type $ @;

if Type='E' then do;

if \_N\_>1 then output;

input ID $;

Deduct=0;

end;

else do;

input DepName $ Relation $;

if Relation='C' then Deduct+25;

else Deduct+50;

end;

if LastRec then output; //**önemli**

run;

**Chapter 5** : Data çevirimleri - Fonksiyonlar

Sas fonksiyonları : bir değer döndürürler. Her arguman birbirinden , işareti ile ayrılır. Arguman olarak, sabit, değişken, fonksiyon ve hesaplamaları alabilirler.

Total=sum(Qtr1,Qtr2,Qtr3,Qtr4); yerine aşağıdaki statement yazılabilir.

Total=sum(of Qtr1-Qtr4);

Değişken listesi olarak farklı şeyler kullanılabilir.

X1-xn numbered range lists

x-a name range lists

sum(of ...) name prefix lists

\_all\_ : tüm değişkenler \_numeric\_ : tüm numeric değişkenler \_character\_:tüm karakter tipi değişkenler

Prog2.freqflyers ds altında uçuş bilgileri var. Yeni bir ds yaratılması isteniyor. Bunun için farklı fonksiyonlar kullanılıyor.

**Karakter tipi değişkenlerde uygulanacak fonksiyonlar :**

**1. Substr : starting position verilmezse, en baştan okumaya başlar.**

Location = columbus, oh 43227

X = Substr(location,11(starting pos),2(charac.extracted)) // baştan itibaren 11 oku, 2 al

X = oh

**2.Right(argument) :**

X= Right(y) // y yi sağa dayalı yazdırır.

**3.Left(argument) :**

X= Left(y) // y yi sola dayalı yazdırır.

**4.Scan(string,n, delimiter) :**

\*ilk kelimeden önceki delimiter ın bir önemi yoktur.

\*iki veya daha fazla delimiter tek delimiter olarak alınır.

\*n negatif ise okumaya en sondan başlar, başa doğru ilerler.

Phrase=software and services

X= Scan(phrase,2, ‘ ’); // delimiter ı ayıraç kabul edip string i parçalar, n . kısmı alır.

X= and

data scan;  
 Text='(Thursday July 4, 1776)';  
 Var1=scan(Text,1);  
 Var2=scan(Text,4);  
 Var3=scan(Text,5);  
 Var4=scan(Text,2,',');  
 Var5=scan(Text,2,',)');   
run;

VAR1:Thursday VAR2:1776 VAR3: VAR4: 1776) VAR5: 1776

**5.concatenation :** !! işareti.

Fullname = firstname !! lastname // arada boşluk kalabilir.

**6.Trim :** trimn ile benzerdir. Eğer içerisi boş ise, trimn 0 değeri döndürür.

Compbl birden fazla boşluğu tek boşluğa dönüştürür.

Fname = trim(firstname)

data labels(keep=FullName Address1 Address2);  
 length FMName LName $ 10;  
 set prog2.freqflyers;  
 if substr(right(ID),6)='1' then  
 Title='Ms.';   
 else if substr(right(ID),6)='2' then  
 Title='Mr.';   
 FMName=scan(Name,2,',');   
 LName=scan(Name,1,',');  
 FullName=Title !! ' ' !! trim(FMName) !!  
 ' ' !! LName;  
 Address2=scan(Address2,1,',') !! ', ' !!   
 scan(Address2,3,',') !! ' ' !!   
 scan(Address2,4,',');  
run;

**7.catx :** baştaki ve sondaki boşlukları siler, istenilen ayıraçı ekler.

FullName = catx(' ',Title,FMName,LName); //aralara boşluk eklemiş

Cat : baştaki ve sondaki boşlukları silmeden birleştirir.

Catt : sadece sondaki boşlukları silip birleştirir.

**Searching :**

Prog2.ffhistory altında; üyelik seviyeleri, ve yıllar var. Silver üye olanları ve bunların silver üye oldukları yılları içeren bir ds yaratılması isteniyor.

**8.Find :**

Text="This target contains a BULL'S-EYE.";

Pos=find(Text,"BULL'S-EYE"); // 24 döndürür. text içinde, string I bulduğu position I döndürür. sas 9 da yenidir. Bulamazsa 0 döndürür.

2 parametresi var:

1 . I : incase sensitive : büyük küçük harf farketmez

2. T : Trailing .. sondaki boşlukları ignore eder.

data index;  
 Text='DELIMIT IT WITH BLANKS.';  
 Pos1=find(Text,'IT');  
 Pos2=find(Text,' IT ');  
 Pos3=find(Text,'it');  
 Pos4=find(Text,'it','I');

run;

pos1 :6 pos2 :8 pos3 :0 pos4 :6

data index2;  
 length String $ 5;  
 String='IT';  
 Text='DELIMIT IT WITH BLANKS.';   
 Pos5=find(Text,String);  
 Pos6=find(Text,String,'T');  
 Pos7=find(Text,' ' !! trim(String) !! ' ');  
run;

pos5:0 pos6:6 pos7:8

string : it\_\_\_ boşluktan dolayı

data silver;  
 set prog2.ffhistory;  
 if find(Status,'silver','I') > 0;  
run;

**9.Index :**

Find gibi. Tek farkı parametresi yok.

data index;  
 Text='DELIMIT IT WITH BLANKS.';  
 Pos1=index(Text,'IT');  
 Pos2=index(Text,' IT ');  
 Pos3=index(Text,'it');  
run;

pos1:6 pos2:8 pos3:0

**10. upcase :** büyük harfe çevirmek için

**11. propercase** : baş harf büyük gerisi küçük. Sas 9 ile geldi.

**12. tranwrd :** text içinden istenilen string i yeni bir string ile değiştirir.

Dessert = Pumpkin pie

Dessert=tranwrd(Dessert,'Pumpkin','Apple');

Dessert : Apple pie

**13. lowcase :** küçük harfe çevirmek için

**14.substr solda ise**; değer ataması yapılır.

Location =Columbus, GA 43227

substr(Location,11,2)='OH';

Location = Columbus, OH 43227

**Numeric tipteki değişkenlerde uygulanacak fonksiyonlar :**

**1.Round :** en yakın sayıya yuvarlar.

data truncate;  
 NewVar1=round(12.12); //12  
 NewVar2=round(42.65,.1); //42.7  
 NewVar3=round(6.478,.01); //6.48  
 NewVar4=round(96.47,10); //100  
run;

**2.Ceil** : aşağı yuvarlar.

**3.Floor** : yukarı yuvarlar.

**4.Int :** ondalık sayının tam kısmını alır.

data truncate;  
 Var1=6.478;  
 NewVar1=ceil(Var1); //7  
 NewVar2=floor(Var1); //6  
 NewVar3=int(Var1); //6  
run;

**5. sum**

**6. mean:** missing değeri alan değişkeni görmezden geliyor,

data summary;  
 Var1=12;  
 Var2=.;  
 Var3=6;  
 NewVar=mean(Var1,Var2,Var3); // 9

run;

NewVar=mean(of Var1 – Var3) yazılabilir.

**7. max**

**8. min**

**Date tipindeki değişkenlerde uygulanacak fonksiyonlar :**

*NewDate*=MDY(*month*,*day*,*year*);

*NewDate*=TODAY();

*NewMonth*=MONTH(*SAS-date-value*);

*NewDay*=DAY(*SAS-date-value*);

*NewYear*=YEAR(*SAS-date-value*);

*NewVal=*YRDIF(*sdate*,*edate*,*basis*); // iki yıl arasındaki farkı bulur

Basis :

Act/act

30/360

Act/360

Act/365

DATDIF : YRDIF ile aynı görevdedir. Sadece act/act ve 30/360 parametreleri ile çalışır.

**Data çevirimleri yapmak için** :

**Input : karakterden numeric e çevirmek için kullanılır**

Sas içinde karakter tipteki bir değişken otomatik olarak numeric değişken e çevrilebilir(değişken standart bir değişken ise) Aksi halde, bir takım informat lar uygulamak gerekiyorsa , bu iş için input kullanılır.

data conversion;  
 CVar1='32000';  
 CVar2='32,000';  
 CVar3='03may2008';  
 CVar4='050308';  
 NVar1=input(CVar1,5.);  
 NVar2=input(CVar2,comma6.);  
 NVar3=input(CVar3,date9.);  
 NVar4=input(CVar4,mmddyy6.);  
run;

proc contents data=conversion;   
run;

# Variable Type Len Pos

1 CVar1 Char 5 32

2 CVar2 Char 6 37

3 CVar3 Char 9 43

4 CVar4 Char 6 52

5 NVar1 Num 8 0

6 NVar2 Num 8 8

7 NVar3 Num 8 16

8 NVar4 Num 8 24

data bonuses;  
 set prog2.salary2;  
 Bonus=.10\*input(GrossPay,comma6.);  
run;

**bir değişkenin tipini değiştirdiğimizde yenisini aynı isimle yaratamayız. Bunu yapabilmek için değişkenin ismini değiştirip daha sonra çevrimi yapmamız gerekir.**

data bonuses(drop=CharGross);  
 set prog2.salary2(rename=(GrossPay= CharGross));  
 GrossPay=input(CharGross,comma6.);  
 Bonus=.10\*GrossPay;  
run;

**Put : numeric tipteki bir değişkeni karaktere çevirmek için kullanılır**

Best12. formatını kullanır. Yani 12 characterlik alan yaratır. Bunu engellemek için, formatlı çevirim için, put kullanılır.

data conversion;  
 NVar1=614;  
 NVar2=55000;  
 NVar3=366;  
 CVar1=put(NVar1,3.);  
 CVar2=put(NVar2,dollar7.);  
 CVar3=put(NVar3,date9.);   
run;  
  
proc contents data=conversion varnum;  
run;

--

data phonenumbers;  
 set prog2.phones;  
 Phone='(' !! put(Code,3.) !! ') ' !!  
 Telephone;  
run;

**Chapter 6** : hatalar

1.syntax error : sas kurallarına uygun yazılmamış

2.logical error : sas kurallarına uygun fakat mantık yanlış .

**Put :**

**Kodun neresi execute edildi, neresi edilmedi,değişkenlerin değerleri.. bilgilerini verir.**

PUT *variable-name*=; // log a değişkeni yazdırır

put Var=; // var =5 ise ; loga var = 5 yazdırır

put var $s. ; // formatlı yazdırır

put \_all\_ //tüm değişkenleri yazdırır

putlog : sas 9 ile gelmiştir. Put tan tek farkı; sadece log a yazar

debug için :

data work.xx / debug ;

|  |  |  |
| --- | --- | --- |
| Command | Abbreviation | Action |
| STEP | ENTER key | Steps through a program one statement at a time. |
| EXAMINE | E *variable(s)* | Displays the value of the variable. |
| WATCH | W *variable(s)* | Suspends execution when the value of the variable changes. |
| LIST WATCH | L W | Lists variables that are watched. |
| QUIT | Q | Halts execution of the  DATA step. |

**Chapter 7 :** Döngüler ve Diziler

**Do Loop :**

data invest;

do Year=2001 to 2003;

Capital+5000;

Capital+(Capital\*.075);

end;

run;

her yılın sonundaki capital i vermesini istersek;

data invest;

do Year=2001 to 2003;

Capital+5000;

Capital+(Capital\*.075);

output;

end;

run;

**Belli bir koşul vermek istiyorsak; Do until :**

data invest; **kaydı yazar**

do until(Capital>1000000);

Year+1;

Capital+5000;

Capital+(Capital\*.075);

end;

run;

önce hesaplamaları yapar, sonra koşulu kontrol eder.

**Do while : kaydı yazmaz**

data invest;

do while(Capital>1000000);

Year+1;

Capital+5000;

Capital+(Capital\*.075);

end;

run;

önce koşulu kontrol eder, sonra hesaplamaları yapar.

Tekrarlayan işlemleri nerelerde kullanıyoruz? :

* aynı özelliklere sahip birden fazla değişken yaratmak istediğimizde,
* datayı okurken,
* rotate ile ds’yi döndürürken
* değişkenleri karşılaştırırken

**Arrays:**

Array içindeki her bir objeye eleman deniliyor.

**Array tanımlaması :**

array Contrib{4} Qtr1 Qtr2 Qtr3 Qtr4;

array Contrib{4} $ Qtr1 Qtr2 Qtr3 Qtr4; // karakter tipi değişkenlerden oluşan array

array isimleri, label, format, drop, keep, lenght statementları içinde kullanılamaz.

**Array lerin kullanımı :**

**Do – to – end yapısı içinde kullanıyoruz**

data charity(drop=Qtr);

set prog2.donate;

array Contrib{4} Qtr1 Qtr2 Qtr3 Qtr4; // Qtr1-Qtr4 yazılabilir.

do Qtr=1 to 4;

Contrib{Qtr}=Contrib{Qtr}\*1.25;

end;

run;

**Arrayleri kullanarak yeni arrayler yaratabiliriz.**

data percent(drop=Qtr);

set prog2.donate;

Total=sum(of Qtr1-Qtr4);

array Contrib{4} Qtr1-Qtr4;

array Percent{4};

do Qtr=1 to 4;

Percent{Qtr}=Contrib{Qtr}/Total;

end;

run;

**iki array elemanı arasındaki farkı bulmak :**

data change(drop=i);

set prog2.donate;

array Contrib{4} Qtr1-Qtr4;

array Diff{3};

do i=1 to 3;

Diff{i}=Contrib{i+1}-Contrib{i};

end;

run;

**Array elemanlarına initial değer vermek ve farkı bulmak :**

data compare(drop=Qtr Goal1-Goal4);

set prog2.donate;

array Contrib{4} Qtr1-Qtr4;

array Diff{4};

array Goal{4} Goal1-Goal4 (10,15,5,10); // initial values

do Qtr=1 to 4;

Diff{Qtr}=Contrib{Qtr}-Goal{Qtr};

end;  
run;

**geçici bir array yaratmak için :**

array Goal{4} \_temporary\_ (10,15,5,10); // drop a gerek kalmaz.

**Rotate için :**

data rotate(drop=Qtr1-Qtr4);

set prog2.donate;

array Contrib{4} Qtr1-Qtr4;

do Qtr=1 to 4;

Amount=Contrib{Qtr};

output;

end;

run;

**Chapter 8** : merging

Prog2.transact içinde, haftanın uçuş bilgileri var, account number ve kaç kişi uçacak,

Prog2.branches içinde, uçuşla ilgili account number ve pilot bilgileri var.

Bizden istenen,

1. haftanın tüm uçuşları - newtrans
2. o hafta uçuşu olmayan accountlar -noactiv
3. uçuşu olup 2. ds de olmayanlar - noacct

data newtrans

noactiv (drop=Trans Amnt)

noacct (drop=Branch);

merge prog2.transact(in=InTrans) // geçici değişken

prog2.branches(in=InBanks); // geçici değişken

by ActNum;

*additional SAS statements*

run;

iki tabloda da olması için ;

InTrans =1 InBanks =1

1.tabloda olup 2.tabloda olmaması için;

InTrans =1 InBanks =0

2.tabloda olup 1.tabloda olmaması için;

InTrans =0 InBanks =1

data newtrans

noactiv (drop=Trans Amnt)

noacct (drop=Branch);

merge prog2.transact(in=InTrans)

prog2.branches(in=InBanks);

by ActNum;

if InTrans and InBanks

then output newtrans;

else if InBanks and not InTrans

then output noactiv;

else if intrans and not InBanks

then output noacct;

run;

merge işleminden önce, ds ler sort edilmelidir.

**SQL :**

-sas ds yaratmadan raporlama ve birleştirme işlemleri yapmayı sağlar

-dataları sort etmeden join işlemi yapar

-karışık match ler var

Proc sql default olarak bir rapor döndürür, ds değil!

Dezavantajı : CPU ve memory den, data step merge e göre, daha fazla yer

proc sql;

select \*

from prog2.transact, **//kartezyen çarpımı yapılır**.

prog2.branches

;

quit;

proc sql;

select Transact.ActNum, Trans, Amnt, Branch

from prog2.transact, prog2.branches

where Transact.ActNum=Branches.ActNum **// iki tabloda da olan ActNum a gore birleştirme yapılır**

;

quit;

**Alias : pratiklik sağlar**

proc sql;

select T.ActNum, Trans, Amnt, Branch

from prog2.transact as T, prog2.branches as B

where T.ActNum=B.ActNum

;

quit;