**SAS PROGRAMMING 1 EĞİTİM NOTLARI**

Sas ile veriler bilgiye dönüştürülüyor.

\* Sas altında v9.. değişiklikleri yapabileceğimiz dosya

**Sas genel bilgiler:**

**CHAPTER 1:**

**Syntax:**

PROC CHART Data = Sas-data-set;

HBar / VBar char-variables;

RUN;

**SAS Bileşenleri: base sas üzerine gelebilecek diğer çözümler.**

* Reporting and graphics- raporlama ve grafik
* Analytical –analiz – data mining- turkcell,tüik kullanıyor..zaman serisi analizi yapılabiliyor.
* Visualization and discovery Data Access and management- veri erişimi ve yönetimi
* Base Sas (en alttaki çözüm- bunu kullanarak diğer componentlerin yaptıklarını yapabiliyoruz)
* Business Solutions- iş çözümleri
* User Interfaces- kullanıcı arayüzleri
* Application Development- uygulama geliştirme
* Web Enablement- web erişimi- portal üzerinden kullanıcılar raporlarını takip edebiliyorlar.

**SAS TASKS-aşamaları : sas ile neler yapılabilir? Data bilgiye dönüşürken çeşitli aşamalardan geçiyor.**

* Data Access:adresler vs . platform bağımsız, elektronik ortamdaki her türlü bilgi alınabilir.
* Data Management:verinin istenildiği gibi şekillendirilmesi yani veri üzerinde yapılacak değişiklikler.
* Data Analysis: Zaman serisi analizleri
* Data Presentation: raporlama fonksiyonları, enterprise guide ile raporlama yapılabilir.

**Verinin Bilgiye Dönüşümü: verinin elektronik ortamda olması yeterli.**

**Bu sürecin %80 inde veri ile ilgili işlemler yapılır. %20 sinde analiz işlemleri yapılır.**

Data—Data Step(kodlama yaparak)—Sas Datasets—Proc Step—information

Sas ın kullanıldığı yerler: turkcell solaris üzerinde kurulu, tüik: Linux üzerinde kurulu ama çoğunlukla Windows üzerinde kurulu.

**Design of the Sas System:**

%90 ı bağımsız sistemlerden oluşuyor: pc, Workstation, süper bilgisayarlar, server, mainframe…

**Veriyi alabildiğimiz yerler**: oracle, db2, dbase… Database’e erişim için: Base Sas yanında lisanslı SAS/ACCESS (DBMS (DataBase Management System) için) e sahip olunması gerekiyor.

**Course scenario = Uçuş, Yolcu, Kargo, Çalışan ve Ücret bilgileri**

**CHAPTER 2:**

**Sas Programları:** veri row data veya daha önce oluşturulmuş bir dataset olabilir.

Row Data(ham veri)/ Sas Dataset –Data Steps—Sas DataSet—Proc Steps (Sas dataset oluşturmak için, rapor ve grafik oluştur, edit data, sort data)—reports (proc steplerden geçerek rpr oluşturulur)

**Sas programları 2 stepten oluşuyor.**

**1. Data Step:** Dataset oluşturuyor, bunu yaparken input statement içindeki değerleri okuyor. Data ile başlayanlar data step.

**2. Proc Step:** Datasetler üzerinde işlem yapıp rapor oluşturuluyor. Proc ile başlayanlar proc step

**Proc Print:** Rapor oluşturuluyor.

Proc print data = work.staff;

Run;

**Proc Means**: özet istatiksel bilgilerin oluşturduğu rapor oluşturuyor.

Stepler run ve quit ile yada yeni bir step başlangıcı ile bitiyor. Her zaman run a ihtiyaç duymuyor çünkü default olarak run varmış gibi program execute ediliyor. loglarda bu durum warning olarak gözüküyor.

Ör:

Data = work.staff;

Infile ‘row data file’ ; okumak istediğimiz yer

Input LastName $ 1-20 FirstName $ 20-30; neleri alıyoruz

Run;

Proc print data = work.staff; run yok!!!

Proc means data = work.staff; mean prosedürü içinde farklı fonksiyonlar var. Mean, standart

Class jobTitle; deviation, max, min vs.. (average yok!!)

Var Salary;

Run;

Sas’,

İnteraktif modda, batch modunda, enterprise guide gibi uygulamalarda ve interaktif olmayan modda, yani farklı format ve modüllerde çalışabilir.

* Interaktif olmayan uygulamalarda:

Sas FileName, Sas Input(FileName) ile uygulama içinden Sas’ı çağırabiliriz.

* Sas programını çağırdığımızda karşımızda iki bölüm olacak:

1. Sas Log- işleyiş ile ilgili mesajların olduğu bilgileri içeren bölüm. Hata mesajları kırmızı, uyarı mesajları yeşil renk.
2. Sas Output- raporlar

* Program editörü açmak için: view- program editör
* Programı açmak için ikinci yol olarak : command bar a ‘include ‘a.sas’ ’ yaz enter a bas.
* Submit için: run—submit
* Loglara erişim : Windows—log . burada yazılan kodlar, değişkenler, kaç tane kayıt olduğu, warning ve error mesajları gözükür..
* Windows ve Unix de programı açmak için 2 yol vardır.

1. file- open seç
2. include ‘a.sas’ yaz enter a bas

* Z/OS de programı açmak için tek yol vardır.

1-include ‘.prog1.sascode(a.sas)’ yaz enter a bas.

* Sas programı: sas kodlarını içerir

Sas Dataset: veri değerlerinin olduğu structured dosyadır. Veriyi alabilmemiz için bir takım steplerden sonra sas dataset haline dönüştürülmüş olması gerekir.

Exercises-2-30 – c02ex1.sas--- hazır var.

* **Sas Datasets**: iki bölümden oluşur.

**1-** **Decriptor Portion:** Tanımlama, değişkenlere ait genel bilgiler, location bilgileri ve diğer dataset bilgileri yer alır. Ds adı, ds label, ne zaman yaratıldı, storage inf. , kayıt sayısı , değişkenlerin ismi, tipi, uzunluğu, position, format, informat, label bilgileri

Proc contents data = work.staff;

Run;

**2- Data Portion:** veri değerleri bulunur.

Proc print data = work.staff;

Run;

* **Variable Values:**

**1- Character:** 1 byte = 1 char. Boşlukta bir karakterdir.32767 uzunluğunda olabilir. (version 6 ve öncesinde en fazla 200 karakter)

**2- Numeric:** 8byte = 1 number-buffer içinde.. 8 byte= 16 digit

Bilgiler önce buffer a sonra dataset e atılır.

Sas variable name: 32 karakter uzunluğunda olabilir. Harfle veya altçizgi ile başlar, harf, altçizgi, sayı ile devam eder. (version 6 ve öncesinde max uzunluk 8)

Date formatında : 01/Jan / 1960 , 0 noktası kabul ediliyor.

Belli bir bölümü çalıştırmak için : o bölümü seçip, submit selected..

Satır sonlarında ; kullanılır.

Missing value eğer karakter ise, blank olarak

Missing value eğer numeric ise period . olarak gözükür.

(isnull: boşluk varsa karakterde boş, numeric te . getirir)

* Özel karakterler kullanmak için:

Ör: Options validvarname = any;

class ‘Flight#’ n;

Sas, free formattır. Yani istenildiği kadar boşluk bırakılabilir.

Sas ın default library si, work tur.

* Açıklama satırı: /\*…………………\*/

\* infile ………..

**Program hata verdiğinde çözmek için:**

Windows: break icon –cancel submitted sta. – y to cancel sub sta.

(in the tasking window manager)

Unix: open-sas session management- cancel submitted sta. – y to cancel sub sta.

Z/OS: press attention key- attention command- cancel submitted sta. – y to cancel ..

Eksik, noktalama, yanlış proc kullanımı gibi durumlarda doğrusunu alıyor fakat loglar içinde warning ile uyarıyor.

Datayı Excel içinden görebiliriz. Sağ tuş- Excel- export(datanın atanacağı ortamı seçebiliyoruz)

**Location of our files:**

C: \ workshop\ winsas\prog1 \ myprog.sas

**Enhanced editör:** sadece Windows da kullanılıyor, daha kolay bir editör- renkli. 9.0 ile geldi

Tools-options-enhanced editör

**Sas libraries:** Explorer tab içinde tüm sas libraries bulunuyor. 2. yol : view- Explorer

Library altındaki elemanlardan birine sağ- prog- dataset ile ilgili bilgiler bulunuyor.

Dataset ‘ i açtığımızda viewtable karşımıza çıkar.

Enable startup ı tıklarsak, kalıcı bir kütüphane olur.

Z/OS için: pmenu yaz- seçmek istemediğinin üzerinde enter a bas..

Tools-options-turn all menus off ile tüm menüleri kapa.

Open için ? yaz enter a bas, open seç. Dataset i açtığımızda karşımıza FSVIEW çıkar.

KEYS yazarak tüm key leri görebiliriz. Tools- options- keys

**CHAPTER 3 :**

**Library:** Sas datasetlerini sakladığımız yerlerdir. Sistemle gelen bazı kütüphaneler vardır.

Work-temporary (default olarak buraya atılır) temporary library leri yazmasak ta olur.

Sasuser, ia- permanent

**Libref (Library Referance**): kütüphane yaratmak için kullanıyoruz. Tüm kütüphanelere libref atanır. Temp library ler session bitince silinir, permanent lar silinmez.

**Libref atama:** libref en fazla 8 karakter olabilir.

Libname libref ‘sas-data-library’ ;

Ör: Windows için: libname ia ‘c:\workshop\winsas\prog1’; Libref atamak!!!

Unix için: libname ia ‘/users/userid’;

Z/OS için: libname ia ‘userid.prog1.sasdata’ disp =shr (old-default olarak read and write, shr-sadece read);

Kodlama dışında kütüphane atama:

Kütüphaneler bölümünde sağ tuş- new-enable at startup tıkla(permanent olması için)

Submit ettiğimizde: doküman ile libref arasında bir bağlantı kurulur, session bittiğinde bağlantı kesilir.

Kütüphane adı en fazla 8 karakter olabilir.

**Data files:** data file isimleri iki bölümden oluşur.

Libref.filename Ör: ia.sales

Temp datafiles da, eğer libref yoksa sadece dosya ismi yazılabilir.

Library içindeki tüm file ları görebilmek için:

Proc contents data = libref(ia). \_all\_ nods; (all ve nods birlikte kullanılmalı)

Run;

Bir library içindeki file içeriğini görmek için:

Proc contents data = ia.crew;

Run;

Libname statement işleyişte run a ihtiyaç duymaz.

**CHAPTER 4:**

**Alt + Tab= slaytları değiştirmek için**

Raporlama proc print ile yapılır. Proc print içinde işlemler sadece görsel olarak yapılır, orjinal data değişmez. Kolonları değiştirmek, label, format vs...

Ör: libname ia ‘sas-data-library’;

Proc print data = ia.empdata;

Run;

Bazı değişkenlerin listelenmesini ya da raporlama sırasını belirlemek istiyorsak:

Var variable(s);

Ör: proc print data =ia.empdata;

Var jobCode empID Salary;

Run;

Rowların başındaki obs numaralarının çıkmasını istemiyorsak:

Ör: Proc print data = ia.empdata noobs;

Var jobCode empID Salary;

Run;

­­­­­­**Where:**

Belli kayıtları seçmek için kullanılır. İçinde case sensitive- büyük küçük harf ayrımı var

**Operatorler :**

**- Comparison**

**- Logical**

**- Special**

**- Functions**

**Comparison(karşılaştırma) operatörleri:**

Not equal: ^= ~=

15 den küçük BClass ----- BClass lt 15 (Lt gt ge le ne )

Where Salary= . ; Salary i bilgisi boş gelenler.

­­­­­­­­Where JobCode not in (‘pilot’,’fltat’); ikisi aynı

JobCode not in (‘pilot’ ’fltat’);

Between… and…

Contains ? where salary ? ‘10000’;

lastname ? ‘lam’ ;

like ‘e\_U%’ \_ tek karakter %birden fazla karakter

Sounds like =\* türkçe de yok.

Is missing / is null where Flight is missing

% numeric değerler için kullanılır.

Sum(variables) :kolonların toplamını alır.

Sum Fclass Eclass ; toplanacak değişkenler ardarda yazılır.

N option : number of observation- noobs dan sonra n yaz . her sayfadaki kayıt sayısının sayfanın altında gösterilmesini sağlar.

**Sorting:** gruplama yapmak için verinin önce sort edilmiş olması gerekir.

out ile sort edilmiş yeni bir dataset oluşturulur.

Proc sort data = ia.empdata; asıl datayı değiştiriyor.

By salary;

Run;

Proc sort data = ia.empdata out = work.empdata (çıkıştaki ismi) ; asıl datayı değiştirmiyor.

By salary;

Run;

Sort aynı zamanda gruplandırma için kullanılır.

Proc sort data = ia.empdata out= work.empdata;

By descending JobCode; neye göre sort edileceğini gösterir.

Run;

Proc print data = work.empdata;

By JobCode; (by ve sum ı birlikte kullandığımızda alt grupları toplamakla

Sum Salary; birlikte, tüm toplamı alabiliyoruz)

Run;

(JobCode’ a göre grupla, maaşların toplamını ve genel toplamı al)

**Page by:** sub grupları birbirinden ayırmak, farklı sayfalarda basılması için kullanılır

Önce proc sort ile empdata nın sıralı hale getirilmesi gerekiyor.

Proc print data = work.empdata;

By jobCode ;

Pageby jobCode; farklı sayfalar

Sum Salary;

Run;

Belli bir kolonu obs yapmak için: id kullanılır.

Proc print data = ia.empdata; (noobs yazılmıyor!)

Id jobCode;

Var EmpID Salary;

Run;

Id ve by birlikte özel bir format oluşturmak için kullanılırlar. Obs kolon olmadığında by line saklandığında: JobCode aynı olanların JobCode unu tekrar tekrar göstermez.

Proc sort data = ia.empdata out= work.empdata;

By JobCode;

Run;

Proc print data = work.empdata; ( sort etmeden bu işlemleri yapamayız)

By JobCode; (by ve sum ı birlikte kullandığımızda alt grupları toplamakla

Id JobCode; birlikte ana toplamı da alabiliyoruz.)

Sum Salary;

Run;

(by ve id tekrarsız obs için kullanılır)

**CHAPTER 5**- rapor görünüşü oluşturma:

**Title eklemek:** Title n ‘text’ ; n en fazla 10 olabilir.

Default title : ‘The Sas System’

**Dipnot eklemek:** Footnote n ‘text’ ;

Title tek başına kullanıldığında diğer tüm title ları engeller.

Columns names – kolon başlıklarına label atama:label en fazla 256 karakter olabilir.

Proc print data = ia.empdata label(split = ‘’ – kelimeleri bölmek için)

Label FirstName = ‘FirstName’;

LastName = ‘LastName’;

Table daki isim rapordaki isim

Title ‘ yeni’ ;

­Run;

**Options :** proc tan önce yazılır , rapor görünümünü değiştirmek için kullanılır.

Date: print için tarih belirler.

Notdate:

Linesize : width (ls = width) satır genişliği

Pagesize : width (ps = width) bir sayfadaki satır sayısı

Number : sayfa numarasının görünmesini sağlar

Nonumber : sayfa numarasının görünmemesini sağlar

Pageno : n sayfa numarası verir. Başlangıçtaki sayfa numarası

Ör: option nodate nonumber ls = 72 ;

**\* ard arda çalıştırdığımızda running modda asılı kalır. Bunu düzeltmek için:**

**! işareti -> yes to cancel..**

**Data format:**

1.Standart formatlar:

w.d: standart format (num) 8.2 (uzunluğu=8 virgülden sonra= 2) 12345.21

$w = Standard format(char) $5 – kathy

Commaw.d (sayı içinde virgül) comma7.2 12,3.14

Dollarw.d (dolar işareti ve virgül) dollar9.2 $12,23.15

Date formats: Mmddyyw6-8-10 uzadıkça : önce / lar gelir sonra yıl uzar.

Standart date formatları:

Date9. :02Jan1960

Mmddyy8. :01/01/60

WordDate. : December 31,1960

Ddmmyy12. : 31/12/2006

Weekdate : Sunday, January 1, 1961

**Formatı kullanmak:**

Proc print ia.employee;

Format Salary dollar11.2;

Run;

**Format yaratmak:**

Proc format;

Value formatname range1 = ‘label1’;

Range2= ‘label2’;

Run;

**2.user defined format:** Önce formatı yarat, sonra kullan

Formatname: 32 karakterden küçük olmalı, karakterler $ ile, numerikler harf veya \_ ile başlar, sayı ile bitemez.

Ör:

Proc format;

Value gender1= ‘female’;

2= ‘male’;

Other = ‘miscoded’;

Value low-49 = ‘az’;

Value $grade ‘A’ = ‘süper’ ;

…..

Proc format…

Proc print data = ia.empdata;

Format salary $grade. ;

Run;

Ör:

proc format;

value $codefmt 'FLTAT'(table içindeki hali)='Flight Attendant'

'PILOT'='Pilot';

value money low-<25000 ='Less than 25,000'

25000-50000='25,000 to 50,000'

50000<-high='More than 50,000';

run;

proc print data=ia.empdata split=' ' noobs;

label LastName='Last Name'

FirstName='First Name'

Salary='Annual Salary';

format Jobcode $codefmt. Salary money.;

title1 'Salary Report in Categories';

run;

**HTML raporları yaratmak:**

Bunun için ODS- output delivery sistem kullanılır.

ODS:window, html,printer,file,other…formlardaki bilgileri alıp Sas raporları halinde, html formatında gösterebilir.

Html sağ tuş-excel de göster

Ods html file = ‘….’ ;

Proc print ..

Proc means…

Proc freq..

Ods html close;

**CHAPTER 6 – Creating a Data set**

Geçici dataset yaratmak için:

Data work.dfwlex;

Kalıcı dataset yaratmak için:

Libname ia ‘sas-data-library’ ;

Data ia.dfwlex;

**data *SAS-data-set-name*;** datasetin adını gösterir. **infile '*raw-data-filename*';** dosyanın yerini gösterir.  
 **input *input-specifications*;** dosyanın nasıl okunacağını gösterir.  
**run;**

**Dosyanın yerini göstermek için:** infile ın kullanımı farklı sistemlerde farklı şekillerdedir.

Infile ‘userid.prog1.dfwlax’ ; Z/OS

Infile ‘/users/userid/dfwlax.dat’ unix

Infile ‘c:\workshop\winsas\prog1\dfwlax.dat’ Windows

**Datafield okumak için:**

Input input specifications; ( sas variable names, variable ların num or char olduğunu gösterir, location bilgileri)

Compilation aşamasında :

Buffer oluşturulur

Pdv oluşturulur

Ds’nin descriptor portion’ ı yaratılır.

**Bilgileri 2 türlü içeri alabiliriz.**

**1. Column input**

**2. Formatted input**

**1.Datayı kolon isimlerini kullanarak okumak: (column input)**

Input variable $ start col- end col;

Ör:

Data work.dflax; geçici bir dataset oluşturduk.

Infile ‘…’ PDV ile (program data vector)

Input flight(kolon ismi) $ 1-3 (kolon numarası) ;

**Dataset yaratma örneği :**

libname ia '*SAS-data-library*'; kalıcı

data ia.dfwlax;

infile '*raw-data-file*';

input Flight $ 1-3 Date $ 4-11 Dest $ 12-14 FirstClass 15(tek karakter) Economy 18-20;

run;

**Compile:** programın çalışıp çalışmadığını kontrol eder.

Bu aşamada: input.. komutları ile o özelliklere sahip boş alanlar buffer içinde pdv yardımıyla oluşturulur.

**Execute :** sonuçları döndürür, Compile aşamasında oluşturulan boş dataset doldurulur

**İşleyiş:** dataseti oluşturduk, içine bilgiler gelirken, her çalışmasında 1 row un kayıtları gelir, run ile row sayısı kadar execute yapılır ve yeni kaydı eskisine ekler, eski kayıt kaybolmaz.

**Oluşturduğumuz dataset e erişim:**

Proc print data = work.dfwlax; temporary

Run;

Libname ia ‘sas data library’ ;

Proc print data = ia.dfwlax ; permanent

Run;

Ruler: kolon numaralarını gösteriyor.

**2.Kendi oluşturduğumuz(formatted input) formattaki datayı okumak için:**

Gelen veriler standart formatlarda olmayabilir. Ör: $ veya virgüllü...bu tür verileri okumak için formatted input oluşturulur. Standart veriler içinde kullanılabilir ancak ilk yöntem daha kolaydır.

Input pointer-control variable informat…

Pointer control: input pointer ı başlangıç noktasına taşımak için kullanılır.

Variable: informat belirlemek için kullanılır.

Informat : içinde input alanının genişliği, dataların nasıl okunacağı ile ilgili bilgiler yer alır.

@n – pointer ı n kolonuna taşır

+ n – pointer ı n pozisyonuna taşır.

**Ör:**

input @1 Flight $3. @4 Date mmddyy8. @12 Dest $3. @15 FirstClass 3. @18 Economy 3.;

Sas informat:

Dataların nasıl okunacağını gösterir.

$(char) informat namew(width). D(decimal)

Eğer number da .varsa informattaki . yı görmezden gelir, yoksa uygun yere yerleştirir.

$8 : sola dayalı

$char8 : sağa dayalı

Comma7.0 : 7 karakteri okur. $ ve , varsa siler

Date: 01/jan 1960 ile verilen tarih arasındaki gün sayısını gösterir.

Errors: dataset e yazılmıyor, fakat erişilebilir.

Karakter okunurken baştaki boşluğu siler. Baştaki boşluğu tutmak için $Char8 kullanılır...

Ör:

libname ia '*SAS-data-library*';

data ia.dfwlax;

infile '*raw-data-file*';

input @1 Flight $3. @4 Date mmddyy8. @12 Dest $3. @15 FirstClass 3. @18 Economy 3.;

(datayı nasıl çekeceğini gösterir)

run;

(1.e git flight ı 3 karakter olarak oku, 15 e git Fclass ı numeric olarak oku)

Değişkenlere özellik atama:

Geçici ve kalıcı olarak dataset içine atabiliyoruz.

Proc step: sadece output u etkiler.

Data step: ds yi etkiler.

Date value ları hem okurken hem yazarken bir format uygulamamız gerekir.

Geçici:

proc print data=ia.dfwlax label;

format Date mmddyy10.; ekrana çıkarırken ki uygulanması gereken format

label Dest='Destination'

FirstClass='First Class Passengers'

Economy='Economy Passengers';

run;

Kalıcı:

libname ia '*SAS-data-library*';

data ia.dfwlax;

infile '*raw-data-file*';

input @1 Flight $3. @4 Date mmddyy8.

@12 Dest $3. @15 FirstClass 3.

@18 Economy 3.;

format Date mmddyy10.;

label Dest='Destination'

FirstClass='First Class Passengers'

Economy='Economy Passengers';

run;

\*\*date normalde ekranda, 01/01/1960 dan sonraki gün sayısı şeklinde durur. Date9 formatı ile

Bunu düzeltebiliriz.

**format Date date9.;**

**Sadece ekranda gözükmesi için:**

Proc contents data = ia.dfwlax ile default attribute leri görüyoruz.

**Kalıcı olarak saklamak için:**

Proc print data = ia.dfwlax label; label ı eklemezsek label sız gösterir.

Run;

Proc print altındaki format data altındakini ezer, datanın orjinali etkilenmez.

Değişkenlere özellik atamak:

Dataset yaratıldığında uzunluğu, tipi ve ismi otomatik olarak atanır. Format.. gibi şeyleri biz sonradan atayabiliriz. Bunu server explorerdan yapabiliriz..

**\*\*Değişkenlerin name, labeli format vs gibi özelliklerini değiştirmek için:**

**Menüden :**

**Columns-sağ tuş- modify- kalıcı**

**PROC DATASETS** LIBRARY=*libref* ;

**MODIFY** *SAS-data-set* ;

**RENAME** *old-name-1=new-name-1*

<. . . *old-name-n=new-name-n*>;

**LABEL** *variable-1*=*'label-1'*  
<. . . *variable-n*=*'label-n'*>;

**FORMAT** *variable-list-1* *format-1*  
<. . . *variable-list-n* *format-n*>;

**INFORMAT** *variable-list-1* *informat-1*   
<. . . *variable-list-n* *informat-n*>;

**RUN;**

Ör: daha önce oluşturulmuş bir ds üzerinde değişiklik yapmak için

Proc datasets library =ia;

Modify dfwlax;

Rename Dest (eski adı) = Destination (yeni adı);

Run;

Proc content;..

**\*\*Excelden data çekmek için:**

2 şekilde yapılır.

1- Import wizard kullanıyoruz..lisansa bağlı olarak değişebilir.. vizard sonundaki kod u sonradan kullanabiliriz. File – import data

2-Kod ile: eğer lisans izin vermiyorsa bu işlem editor altından kod ile yapılabilir.

PROC IMPORT OUT= WORK.DFWLAX (sas dataset ismi)

DATAFILE= "DallasLA.xls" (kaynak ismi)

DBMS=EXCEL2000 REPLACE;

GETNAMES=YES;

RUN;

Tab delimited file: text ten dosya almak için:

PROC IMPORT OUT= WORK.DFWLAX

DATAFILE= "DallasLA.txt"

DBMS=TAB REPLACE;

GETNAMES=YES;

RUN;

Replace : eskisinin yerine yazar

DMBS : dosya tipi

Mixed : yes : numericleri karaktere çevir.

Getnames : yes : ilk kaydı kolon adı olarak al

Sheet = “sheet1” : exceldeki belli bir sheet’ I oku

**CHAPTER 7**

**Dataset okuyarak dataset oluşturmak için:**

1-Data: yeni dataset adını oluşturmak için.

2-Set: datayı alacağımız yer. İçindeki işlemi PDV yapar, boş bir kalıp gelir.

3-Assignment içinde operand ve operatörler var.

Operandlar: değişken isimleri, sabitler

Operatörler: aritmetik semboller, Sas function ları.

Ör:

Data onboard;

Set ia.dfwlex;

Total = fc + E; değişkenlerden birinde değer yoksa, total boş gelir. Bunu engellemek için sum fonksiyonunu kullanıyoruz.

Total= sum(FC,E);

Kümülatif toplam için:

Sum Total;

Compile aşamasında: boş kalıp geliyor, execution aşamasında kalıp dolduruluyor.run ile diğer satırlara geçiyor.

**Sas Date fonksiyonları:**

Today() = sistem saatinden değer alıyor.

MDY() = month day year olarak döndürüyor.

Year() = sas date’ i alıyor, yıl bilgisini döndürüyor

QTR() = kaçıncı çeyrekte

Month() = kaçıncı ay olduğu bilgisini döndürüyor

Weekday() = haftanın kaçıncı günü olduğu bilgisini döndürüyor

Data step içinde: DayOfWeek=weekday(Date);

proc print data=onboard;

var Flight Dest Total DayOfWeek Date;

format Date weekdate.;

run;

Yaratılacak dataset içinde bazı kolonları eksiltmek istiyorsak: drop variable

Tutmak istiyorsak : keep variable

**Conditional Statements:**

If-then-else: if bloğu içinde tek statement vardır.

Ör:

Data flightrew;

Set ia.dfwlax;

Total = sum(FC,E);

If Dest = ‘Lax’ then

Revenue = sum(2000\*FC,1200\*E);

Else if Dest = ‘Dfw’ then

Revenue = sum(1500\*FC,900\*E);

Run;

**Büyük harfe çevirmek için:** upcase(variable);

If Upcase(Dest) = ‘Lax’ then

Revenue = sum(2000\*FC,1200\*E);

If.. then do / End: Birden fazla işlem yaptırmak istediğimizde.

\* Yeni field yaratıldığında ilk değişkenin uzunluğunu bütün field alır.

Yeni bir field yarattığımızda lenght kullanımı format a göre daha mantıklıdır.

Lenght city $11;

**\* Silmek için:**

If.... then delete;

if Total le 175 then delete;

total değeri 175 den küçük tüm kayıtları siler

\* If in tek başına kullanımı mümkündür.

If newSal>10;

Set ile dataset ten alıyoruz, raw data dan almak için infile kullanıyoruz...

If date lt ‘14dec2000’ d ;

String değil date bilgisi olduğunu algılaması için d kullanılıyor.

75- hangi statement içinde nelerin kullanılacağını göstermek için..

**\* yeni oluşturulan dataset e belli değişkenlerin yazılması için:**

Keep ve drop kullanıyoruz.

*SAS-data-set*(**DROP=***variables*)

or

*SAS-data-set*(**KEEP=***variables*)

data onboard(drop=FirstClass Economy);

((data onboard(keep=Flight Date Dest Total); ))

set ia.dfwlax;

Total=FirstClass+Economy;

run;

\* Date formatında zamanı saati ile almak için:

Datepart(datetime);

Datepart(01dec2000)

Datetime19. : uygulanacak format

**CHAPTER 8- Datasetleri bağlamak**

**Datasetleri birleştirmek için:** bütün kolonlar aynı olduğunda kullanılır...

data work.qtr1; yenidataset

set work.jan work.feb work.mar; Set oluşturulacakdataset

run;

Datasetleri birleştirirken; değişkenler aynı değerler farklı ise ardarda ekliyor.

Değişkenler aynı fakat değişken isimleri farklı ise ör JobCode jCode;

Farklı kolon gibi çıkıyor. Bunu engellemek için;

Data new;

Set fa1 fa2 (rename = (eskiisim fa2 nin = yeniisim fa1 in));

Run;

Sıralı olmasaydı önce proc sort ile sıralayacaktık.gruplandırma için sıralı olması gerekir.

Önce işlemi yapıp sonra sıraladığında süre ve yer sorunu olabilir.

Data work.al;

Set ia.bl;

Ia.cl;

Ia.pl;

By id; (birleştirme yaparken bu değişkene göre yapar. Bunun için orj ds’nin de bu değişkenegöre sıralı olması gerekir.)

Run;

proc sort data=fa1;

by name;

run;

proc sort data=fa2;

by name;

run;

data work.allemp;

set ia.miamiemp

ia.parisemp

ia.romeemp;

by ID;

run;

\* İki dataset kolonlarını birleştirip yeni kolon oluşturmak için: merge . merge den önce sort edilmesi gerekiyor.

Ör : Aylık satışları aylık hedefler ile karşılaştırıyor ve aradaki farkları alıyor.

data ia.compare;  
 merge ia.performance ia.goals;  
 by Month;  
 Difference=Sales-Goal;  
run;

Run ile pdv deki output dataset e atanıyor.

**\* Bazı kolonların çıkmasını istemiyorsak:**

Merge ia.gercrew(in=InCrew) InCrew adında geçici bir değişken yaratıyor.

Work.gersched(in =InSched)

(eğer InCrew ia.gerCrew altında varsa ve geliyorsa 1 döndür, değilse 0 döndür, yani ds den bilgi gelmişse 1 , gelmemişse 0 değerini alıyor.

If InSched = 1

..

Proc print ile kaç kişi uçacağını gösterir.

**CHAPTER 9**

Proc print dışındaki rapor tipleri: Rapor yaratmak için kullandığımız datasetlerimiz bazen çok büyük olabilir. Bu durumda datayı özet hale getirip raporu oluşturmamız gerekebilir.bunun için farklı rapor tipleri kullanabiliriz.

**1-Proc Freq** : frekans tabloları. Çalışanların iş gruplarına göre maaşları? İş kodlarına göre çalışanların toplam içindeki % si..

**2- Proc Means** : içinde basit fonksiyonlar var. İş kodlarına göre, ortalama standart sapma vs alınabilir.

**3-Proc Report** : iş koduna göre esnek detay veya özet raporlar alabiliyoruz.

**4-Proc Tabulate** : tabulate şeklinde geliyor. Ör: FLTAT1 dekilerin oturdukları yere göre maaşlarının ortalaması ( yerine başka bir fonksiyon yazabilriz) en sondaki toplamı verir.

Raporlarda neler yapabiliriz:

Tek yönlü veya çift yönlü frekans tabloları oluşturabiliriz. Gruplama yapılabilir. Means prosedürü kullanılabilir. Tüm değişkenleri kullanmak zorunda değiliz.

Bu raporlar enterprise guide ile daha güzel bir şekilde yapılabilir.

İlk hedef: jobcode lara göre toplam kaç kişi çalışıyor?

İkinci hedef: jobcode lara göre her bir iş grubundaki kişi sayısı ve maaşları..

**1-Proc freq:**

Dataset altındaki tüm değişkenlerin frekanslarını yani kaç kere tekrarlandıklarını hesaplıyor.

Proc freq data= ia.crew;

Run;

Bir frekans tablosu yaratırken:

Tüm değişkenler analiz edilir. Her bir alandaki toplan kayıt sayısı hesaplanır. Kaç tane missing değer olduğunu gösterir.

Proc freq data= ia.crew;

Run;

Bu kodu çalıştırdığımızda, her değişkene göre hesapladığından anlamsız ve karışık bir sonuç ortaya çıkar. Bunu engellemek için tables kullanıyoruz.

Proc freq data = ia.crew;

Tables JobCode; (JobCode a göre frekans analizi yapıyor.)

Run;

Frekans tablosunda kaç level olduğunu göstermek istiyorsak:

Proc freq data = ia.crew nlevels;

Tables Location; (Location a göre frekans analizi yapıyor.)

Run;

Çift yönlü frekans analizi:Birden fazla alana göre frekans tablosu yaptığımızda: iki değişkenin birbirleri ile tüm kombinasyonları.

Tables f1 \* f2;

Çift yönlü frekans analizinde istersek liste şeklinde gösterebiliriz.

Proc freq data = ia.crew nlevels / crosslist;

Tables JobCode\* Location;

Run;

Frekans tablosunda cumulative percentage ve cumulative frequency çıkmasını engellemek için:

Proc freq data = ia.crew nlevels;

Tables Location; / nocum;

Run;

**2- Proc Means:**

Business task:

Her bir iş koduna göre, maksimum minimum ve ortalama maaşların hesaplanmasını istiyor.

Bunun için proc mean prosedürünü kullanıyoruz.

Proc means data = ia.crew;

Run;

Proc mean prosedürürün tek özelliği: numeric değişkenler üzerinde işlem yapılabiliyor. Default olarak min max mean stand dev n = kayıt sayısıhesaplıyor.

Ek olarak: range median sum nmiss=missinglerin kayıt sayısı hesaplatabiliyoruz.

Proc means de: bir table içindeki değişkenlerden sadece bazıları üzerinde işlem yapmak istiyorsak;

Proc means data = ia.crew;

Var Salary;

Run;

Mean içinde class statement ı kullanarak gruplama yapabiliyoruz. İş kodlarına göre maaşların ortalamasını al:

Proc means data = ia.crew;

Var Salary;

Class JobCode;

Run;

Hesaplamaları yaparken sayılardaki digit sayısını belirleyebiliyoruz

Proc means data = ia.crew maxdec = 2; digit sayısı:2

Var Salary;

Class JobCode; // gruplama elemanı

Run;

**3- Proc Report:**

Bu raporu iki ayrı ortamda açabiliriz.

1. interaktif ortam- windows ortam

proc report data = ia.crew wd;

run;

1. raporu output ekranında açması için:

proc report data = ia.crew nowd;

run;

report procedure ile: tüm değişkenler listelenebilir. Kolon başlıkları , değişken isimleri, kolonlar için default genişlikler listelenir. Karakter leri sola, numeric leri sağa yaslıyabiliyoruz. Kayıtları sıralı olarak getirebiliriz.

Proc means deki var yerine proc report ta görüntülenmesini istediğimiz değişkenler için column kullanılır.

proc report data = ia.crew wd;

column JobCode Location;

run;

**Define:** formatta, kolon başlıkları.. değiştirmek istiyorsak define kullanıyoruz.

Label, format, width,

proc report data = ia.crew wd;

column JobCode Location;

define JobCode / width = 34 ‘Job Code’;

define Salary / format= dollar10.;

run;

**Order:** tekrarlamalar için kullanılır.

proc report data = ia.crew wd;

column JobCode Location;

define JobCode / order width = 34 ‘Job Code’;

define Salary / format= dollar10.;

run;

Business Task: her location için iş kodlarına göre maaşların özetlerini istiyor.

Normal raporda iş kodları gruplanmaz. Bunun için group kullanırız. Birden fazla değişkeni grup değişkeni olarak tanımlayabiliriz. Eğer grup kullanıyorsak order a ihtiyaç yok çünkü tekrarlama yok.

proc report data = ia.crew wd;

column JobCode Location;

define JobCode / group width = 34 ‘Job Code’;

define Salary / format= dollar10.;

run;

Defalt olarak sum fonksiyonu ile toplamı hesaplıyoruz. Bu değiştirilebilir:

Sum, N(nonmissing datalar için), mean, max, min hesaplanabilir.

define Salary / mean format= dollar10.;

**toplam alınmasını istiyorsak**:

rbreak before/after summarize(toplam)///ol(toplam üstüne tek çizgi), dol(toplam üstüne çift çizgi), ul(toplam altına tek çizgi),dul (toplam altına çift çizgi).///

proc report data = ia.crew wd;

column JobCode Location;

define JobCode / group width = 34 ‘Job Code’;

define Salary / format= dollar10.;

rbreak before /summarize dol;

run;

**raporun header kısmına çizgi eklemek için**: headline headskip

proc report data = ia.crew wd headline headskip;

column JobCode Location;

define JobCode / group width = 34 ‘Job Code’;

define Salary / format= dollar10.;

rbreak before /summarize dol;

run;

Proc print – Proc report farkı :

Proc print’ te olmayanlar : summary report, crosstabular report, auto label, sort data

**4- Proc Tabulate:**

İçiçe geçen , tablo şeklinde raporlar oluşturuluyor. Bu prosedür ile , değişkenler, format, label atamaları, bazı istatiksel hesaplamalar yapılabiliyor. Dezavantajı: ayrıntılı rapor hazırlanamıyor ve yeni bir kolon oluşturulamıyor.

**Tek taraflı rapor:**

Proc tabulate data =ia.fltat; dataset ismi

Class Location; gruplama yapılacak değişken ismi

Var Salary; analizi yapılacak değişken ismi

Table Location All; satır sütun bilgileri, All ile o satır toplamı alınır.

Run;

**Çift taraflı rapor:** maaşların analizi yapılıyor

Proc tabulate data =ia.fltat;

Class Location JobCode;

Var Salary; // normalde maaşlar değil sayılar listeleniyor

Table JobCode (satır), Location(sütun);

Run;

**Var yoksa Kaç kişi olduğunun analizi yapılıyor.**

Proc tabulate data =ia.fltat;

Class Location JobCode;

Table Location(row) All JobCode(column) All; JobCode ve Location daki değerlerin toplamını alır.

Run;

**Location a göre maaş bilgisi:**

Proc tabulate data =ia.fltat;

Class Location JobCode;

Var Salary;

Table JobCode, Location\* Salary;

Run;

Default olarak Sum yapılıyor. Bunu değiştirebiliriz.

Proc tabulate data =ia.fltat;

Class Location JobCode;

Var Salary;

Table JobCode, Location\* Salary\* **mean**; // normalde sum yapıyor

Run;

**Diğer seçenekler:**

Nmiss: missing kayıt sayısı

Std: standart sapma

Min: minimum değer

Max: maksimum değer

Range: aralık

Median: ortalama değer

**İşkodlarına göre toplam maaş ortalaması için:**

Proc tabulate data =ia.fltat;

Class Location JobCode;

Var Salary;

Table JobCode all, Location\* Salary\* mean **all\* salary\* mean**;

Run;

**CHAPTER 10: Chart,Plot**

**1-Charts: Default olarak frequency hesaplanır**

Proc Gchart data= ia.crew;

Hbar JobCode; JobCode ‘un horizontal bar ı

Vbar JobCode; JobCode ‘un vertical bar ı

Pie JobCode; JobCode ‘un pie chart

Run;

Karakter değişkenler için: dikey frekanslar tek tek gelir.

Numeric değişkenler için: dikey frekanslar aralık olarak verilir. Daha küçük aralıklar vermek için Discrete opsiyonu kullanılır. Neye göre ölçeklendirme yapılıyor????

Proc Gchart data= ia.crew;

vbar JobCode / discrete;

Run;

Her zaman işkodunun frekansı istenmez. İşkodlarına göre maaşların ortalamasını istediğimizde:

Sumvar ve type statementlarını kullanırız.

Sumvar ile değişken ismi

Type ile sum, mean..

Proc Gchart data= ia.crew;

Hbar JobCode / sumvar=salary type= mean;

Run;

**Pie chart’larda: içinin farklı bir şekilde doldurulması istenebilir.** Ör:Amaç daha az kartuş kullanmak.. default olarak fill = s kullanılır.

Proc Gchart data= ia.crew;

Pie JobCode / sumvar=salary type= mean fill=x;

Run;

**Pastadan bir dilimin çıkarılması istenebilir.**

Proc Gchart data= ia.crew;

Pie JobCode / sumvar=salary type= mean fill=x explode= ‘PILOT3’;

Run;

**Output ile ilgili uygulamalar:** Yazıcı belirleyebiliyoruz..title, footnote verip bunların fontunu rengini yüksekliğini ayarlayabiliyoruz..

Goptions dev=HPD..(printer cinsi)

Title color= green ‘titleee’ height=3 in (cm , pct) ;

Footnote color= green ‘footnote’ height=3 in (cm , pct) ;

**2-Plots:**

Proc gplot data = ia.crew;

Plot Boarded(vertical) \* Date(horizontal);

Run;

**Birleştirme noktalarını ve çizgi şeklini belirlemek için:** Symbol

Birleştirme noktaları:

Value(v)=plus,star,square,diamond,triangle,none.

Çizgi şekli:

Interpolation(I)=join-düz çizgi, spline-yumuşak çizgi, needle-dikey çizgiler..

Proc gplot data = ia.crew;

Plot Boarded(vertical) \* Date(horizontal);

Symbol value=square i=join;

Run;

**Çizginin rengini kalınlığını ayarlayabiliyoruz**

Proc gplot data = ia.crew;

Plot Boarded(vertical) \* Date(horizontal);

Symbol value=square i=join c=red w=2;

Run;

**Symbol ile ilgili:**

Symbol1; yazarsak symbol1 ile ilgili daha önceki tüm tanımlamaları sileriz

Goptions reset=Symbol; daha önceki tüm symbol tanımlamalarını sileriz.

**Eksenlere yeni özellikler ekleyebiliyoruz..**

Proc gplot data = ia.crew;

Plot Boarded\* Date/ vaxis=100 to 200 by 10 ctext=blue ;

Symbol value=square i=join c=red w=2;

Run;

Haxis = 0 to 25 by 5

Vaxis = 0 to 25 by 5

Caxis = colour of axis

Ctext = colour of text

**Eksenlere label ekleyebiliyoruz..**

Proc gplot data = ia.crew;

Plot Boarded\* Date/ vaxis=100 to 200 by 10 ctext=blue ;

Label Boarded = ‘passengers boarded’

Date = ‘Departure date’;

Run;