/\* example 2.2 - creates permanent data set w/ column input from instream data\*/

LIBNAME stat480 '\\client\C$\Users\Owner\Documents\Whit Masters\STAT 480\data\'; \*Specifies the SAS data

library (directory);

**DATA** stat480.temp2;

input subj **1**-**4** gender **6** height **8**-**9** weight **11**-**13**;

DATALINES;

1024 1 65 125

1167 1 68 140

1168 2 68 190

1201 2 72 190

1302 1 63 115

;

**RUN**;

**PROC** **PRINT** data=stat480.temp2;

title 'Output dataset: STAT480.TEMP2';

**RUN**;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Reading Raw Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\* example 2.3 - creates temp data set w/ column input from raw data (.txt or .dat)\*/

**DATA** temp3;

infile '\\client\C$\Users\Owner\Documents\Whit Masters\STAT 480\data\temp3.txt';

input subj **1**-**4** gender **6** height **8**-**9** weight **11**-**13**;

**RUN**;

**PROC** **PRINT** data=temp3;

title 'Output dataset: TEMP3';

**RUN**;

/\* example 2.4 - creates temp data set w/ column input from raw data (.txt or .dat) using FILENAME statement\*/

FILENAME patients '\\client\C$\Users\Owner\Documents\Whit Masters\STAT 480\data\temp3.txt'; /\*creates a file reference to external file. 1-8 characters\*/

**DATA** temp4;

infile patients;

input subj **1**-**4** gender **6** height **8**-**9** weight **11**-**13**;

**RUN**;

**PROC** **PRINT** data=temp4;

title 'Output dataset: TEMP4';

**RUN**;

/\* example 2.5 column input \*/

options nodate; /\* eliminates the date and time from the output\*/

**DATA** temp;

input subj **1**-**4** name $ **6**-**23** gender **25** height **27**-**28** weight **30**-**32**;

CARDS;

1024 Alice Smith 1 65 125

1167 Maryann White 1 68 140

1168 Thomas Jones 2 68 190

1201 Benedictine Arnold 2 68 190

1302 Felicia Ho 1 63 115

;

**RUN**;

**PROC** **PRINT** data=temp;

title 'Output dataset: TEMP';

**RUN**;

/\* example 2.6 column input advanced \*/

options nodate; /\* eliminates the date and time from the output\*/

**DATA** temp;

input init $ **6** f\_name $ **6**-**16** l\_name $ **18**-**23**

weight **30**-**32** height **27**-**28**;

CARDS;

1024 Alice Smith 1 65 125

1167 Maryann White 1 68 140

1168 Thomas Jones 2 190

1201 Benedictine Arnold 2 68 190

1302 Felicia Ho 1 63 115

;

**RUN**;

**PROC** **PRINT** data=temp;

title 'Output dataset: TEMP';

**RUN**;

/\* example 2.7 reading data in from other data SAS sets \*/

options nodate; /\* eliminates the date and time from the output\*/

LIBNAME stat480 '\\client\C$\Users\Owner\Documents\Whit Masters\STAT 480\data\';

**DATA** temp;

set stat480.temp2; \*note: the sas file, temp2.sas7bdat, must be already saved in the name file location;

**RUN**;

**PROC** **PRINT** data=temp;

title 'Output dataset: TEMP';

**RUN**;

**Lesson 3**

* **Viewing content of SAS library**

**DATA** sasuser.temp3;

infile 'C:\Users\Owner\Documents\Whit Masters\STAT 480\data\temp3.txt';

input subj **1**-**4** gender **6** height **8**-**9** weight **11**-**13**;

RUN;

/\*DATA sasuser.temp3;

data statment generates a new permanent sas fil in the sasuser library using the raw data file on hard drive\*/

/\* 3.1 – viewing contensts of SAS library: sasuser\*/

**PROC** **CONTENTS** data = sasuser.\_ALL\_ nods;

**RUN**;

/\* 3.2 - listing of descriptor information for temp3 data set in sasuser library\*/

**PROC** **CONTENTS** data = sasuser.temp3;

**RUN**;

/\* 3.3 - listing of variable present in temp3 data set, in correct order, in sasuser library\*/

**PROC** **CONTENTS** data = sasuser.temp3 varnum;

**RUN**;

* **LIST Input**

/\* 3.4 - simple example of list input\*/

**DATA** temp;

input subj name $ gender height weight;

\* The $ that follows name tells SAS that it is

a character variable;

\* By default, name only allows up to 8 characters

to be read in;

CARDS;

1024 Alice 1 65 125

1167 Maryann 1 68 140

1168 Thomas 2 68 190

1201 Benny 2 72 190

1302 Felicia 1 63 115

;

RUN;

**PROC** **PRINT** data=temp;

title 'Output dataset: TEMP';

**RUN**;

/\* 3.5 - example when missing value is not accounted for with a "." \*/

**DATA** temp;

input subj name $ gender height weight;

CARDS;

1024 Alice 1 65 125

1167 Maryann 1 68 140

1168 Thomas 2 68 190

1201 Benny 2 . 190

1302 Felicia 1 63 115

;

RUN;

**PROC** **PRINT** data=temp;

title 'Output dataset: TEMP';

**RUN**;

/\* 3.6 - example how character variable is truncated at 8 characters\*/

**DATA** temp;

input subj name $ gender height weight;

CARDS;

1024 Alice 1 65 125

1167 Maryann 1 68 140

1168 Thomas 2 68 190

1201 Benedictine 2 68 190

1302 Felicia 1 63 115

;

RUN;

**PROC** **PRINT** data=temp;

title 'Output dataset: TEMP';

**RUN**;

/\* 3.7 - example of when delimiter is not a space, use INFILE statement\*/

**DATA** temp;

infile cards delimiter=',';

input subj name $ gender height weight;

CARDS;

1024,Alice,1,65,125

1167,Maryann,1,68,140

1168,Thomas,2,68,190

1201,Benny,2,.,190

1302,Felicia,1,63,115

;

RUN;

**PROC** **PRINT** data=temp;

title 'Output dataset: TEMP';

**RUN**;

* **Formatted Input**

/\* 3.8 - example of uses the @n column pointer control and standard

numeric informats to read three numeric variables — subj, height, and

weight — into a temporary SAS data set called temp\*/

**DATA** temp;

input @**1** subj **4.** /\* 4. is 'w.d' format: w=# of columns wide, d=# of decimal places; only 'w.' is required for informat\*/

@**27** height **2.**

@**30** weight **3.**;

DATALINES;

1024 Alice Smith 1 65 125 12/1/95 2,036

1167 Maryann White 1 68 140 12/01/95 1,800

1168 Thomas Jones 2 190 12/2/95 2,302

1201 Benedictine Arnold 2 68 190 11/30/95 2,432

1302 Felicia Ho 1 63 115 1/1/96 1,972

;

RUN;

**PROC** **PRINT** data = temp;

title 'Output dataset: TEMP';

**RUN**;

/\* 3.9 - example of using the @n column pointer control and standard

numeric informats (w.d) to read numeric and character variables

into a temporary SAS data set called temp\*/

**DATA** temp;

input @**18** l\_name $6.

@**6** f\_name $11.

@**30** weight **3.**

@**27** height **2.**;

DATALINES;

1024 Alice Smith 1 65 125 12/1/95 2,036

1167 Maryann White 1 68 140 12/01/95 1,800

1168 Thomas Jones 2 190 12/2/95 2,302

1201 Benedictine Arnold 2 68 190 11/30/95 2,432

1302 Felicia Ho 1 63 115 1/1/96 1,972

;

RUN;

**PROC** **PRINT** data = temp;

title 'Output dataset: TEMP';

**RUN**;

/\* 3.10 - example of using the absolute column pointer control and standard

numeric informats (w.d), SAS program uses +n relative pointer controls with nonstandard informats

into a temporary SAS data set called temp\*/

**DATA** temp;

input @**1** subj **4.**

@**6** f\_name $11.

@**18** l\_name $6.

+**3** height **2.** /\*moves pointer ahead 3 columns from 24\*/

+**5** wt\_date mmddyy8. /\*moves pointer ahead 5 columns\*/

+**1** calorie comma5.; /\*moves pointer ahead 1 columns\*/

FORMAT wt\_date mmddyy8. calorie comma5.; \*required to format wt-date and calorie;

DATALINES;

1024 Alice Smith 1 65 125 12/1/95 2,036

1167 Maryann White 1 68 140 12/01/95 1,800

1168 Thomas Jones 2 190 12/2/95 2,302

1201 Benedictine Arnold 2 68 190 11/30/95 2,432

1302 Felicia Ho 1 63 115 1/1/96 1,972

;

RUN;

**PROC** **PRINT** data = temp;

title 'Output dataset: TEMP';

**RUN**;

**Lesson 4**

* Modifying data

/\* 4.1\*/

**DATA** grades;

input name $ **1**-**15** e1 e2 e3 e4 p1 f1;

DATALINES;

Alexander Smith 78 82 86 69 97 80

John Simon 88 72 86 . 100 85

Patricia Jones 98 92 92 99 99 93

Jack Benedict 54 63 71 49 82 69

Rene Porter 100 62 88 74 98 92

;

RUN;

**PROC** **PRINT** data = grades;

var name e1 e2 e3 e4 p1 f1;

**RUN**;

/\* 4.2 simple assignment statement\*/

**DATA** grades;

input name $ **1**-**15** e1 e2 e3 e4 p1 f1;

\* add up each students four exam scores

and store it in examtotal;

examtotal = e1 + e2 + e3 + e4;

DATALINES;

Alexander Smith 78 82 86 69 97 80

John Simon 88 72 86 . 100 85

Patricia Jones 98 92 92 99 99 93

Jack Benedict 54 63 71 49 82 69

Rene Porter 100 62 88 74 98 92

;

RUN;

**PROC** **PRINT** data = grades;

var name e1 e2 e3 e4 examtotal;

**RUN**;

/\* 4.3 modifying existing variable \*/

**DATA** grades;

input name $ **1**-**15** e1 e2 e3 e4 p1 f1;

e2 = e2 + **8**; \* add 8 to each student's

second exam score (e2);

DATALINES;

Alexander Smith 78 82 86 69 97 80

John Simon 88 72 86 . 100 85

Patricia Jones 98 92 92 99 99 93

Jack Benedict 54 63 71 49 82 69

Rene Porter 100 62 88 74 98 92

;

RUN;

**PROC** **PRINT** data = grades;

var name e1 e2 e3 e4 p1 f1;

**RUN**;

/\*4.5\* calculating the mean\*/

**DATA** grades;

input name $ **1**-**15** e1 e2 e3 e4 p1 f1;

final = **0.6**\*(e1+e2+e3+e4)/**4** + **0.2**\*p1 + **0.2**\*f1;

DATALINES;

Alexander Smith 78 82 86 69 97 80

John Simon 88 72 86 . 100 85

Patricia Jones 98 92 92 99 99 93

Jack Benedict 54 63 71 49 82 69

Rene Porter 100 62 88 74 98 92

;

RUN;

**PROC** **PRINT** data = grades;

var name e1 e2 e3 e4 p1 f1 final;

**RUN**;

/\* 4.6 using the mean function is SAS\*/

**DATA** grades;

input name $ **1**-**15** e1 e2 e3 e4 p1 f1;

\* calculate the average by definition;

avg1 = (e1+e2+e3+e4)/**4**;

\* calculate the average using the mean function;

avg2 = mean(e1,e2,e3,e4);

DATALINES;

Alexander Smith 78 82 86 69 97 80

John Simon 88 72 86 . 100 85

Patricia Jones 98 92 92 99 99 93

Jack Benedict 54 63 71 49 82 69

Rene Porter 100 62 88 74 98 92

;

RUN;

**PROC** **PRINT** data = grades;

var name e1 e2 e3 e4 avg1 avg2;

**RUN**;

/\* 4.7 using the INT function to extract integer area codes\*/

**DATA** grades;

input name $ **1**-**15** phone e1 e2 e3 e4 p1 f1;

areacode = int(phone/**10000000**);

DATALINES;

Alexander Smith 8145551212 78 82 86 69 97 80

John Simon 8145562314 88 72 86 . 100 85

Patricia Jones 7175559999 98 92 92 99 99 93

Jack Benedict 5705551111 54 63 71 49 82 69

Rene Porter 8145542323 100 62 88 74 98 92

;

RUN;

**PROC** **PRINT** data = grades;

var name phone areacode;

**RUN**;

/\* 4.8 - examples of nested functions, SAS works from inside out\*/

**DATA** grades;

input name $ **1**-**15** e1 e2 e3 e4 p1 f1;

\*calculate the average using the mean function

and then round it to the nearest digit;

avg1 = mean(e1,e2,e3,e4);

avg = round(avg1,**1**); \* rounded to nearest ones digit;

DATALINES;

Alexander Smith 78 82 86 69 97 80

John Simon 88 72 86 . 100 85

Patricia Jones 98 92 92 99 99 93

Jack Benedict 54 63 71 49 82 69

Rene Porter 100 62 88 74 98 92

;

RUN;

**PROC** **PRINT** data = grades;

var name e1 e2 e3 e4 avg avg1;

**RUN**;

**/\* 4.9 creating a character variable using if-then-else statement\*/**

**DATA** grades;

input name $ **1**-**15** e1 e2 e3 e4 p1 f1;

\* calculate the average using the mean function;

avg = mean(e1,e2,e3,e4);

\* if the average is less than 65 indicate failed,

otherwise indicate passed;

if (avg < **65**) then status = "Failed";

else status = "Passed";

DATALINES;

Alexander Smith 78 82 86 69 97 80

John Simon 88 72 86 . 100 85

Patricia Jones 98 92 92 99 99 93

Jack Benedict 54 63 71 49 82 69

Rene Porter 100 62 88 74 98 92

;

RUN;

**PROC** **PRINT** data = grades;

var name e1 e2 e3 e4 avg status;

**RUN**;

/\* 4.11 SAS tries to convert characters to numeric automatically\*/

**DATA** grades;

input name $ **1**-**15** e1 $ e2 $ e3 $ e4 $ standtest $;

avg = round(mean(e1,e2,e3,e4),**1**);

std = standtest/**4**;

DATALINES;

Alexander Smith 78 82 86 69 1,210

John Simon 88 72 86 . 990

Patricia Jones 98 92 92 99 1,010

Jack Benedict 54 63 71 49 875

Rene Porter 100 62 88 74 1,180

;

RUN;

**PROC** **PRINT** data = grades;

**RUN**;

/\* use input function to convert standtest to numeric value using informat comma5.\*/

**DATA** grades;

input name $ **1**-**15** e1 $ e2 $ e3 $ e4 $ standtest $;

std = input(standtest,comma5.)/**4**;

DATALINES;

Alexander Smith 78 82 86 69 1,210

John Simon 88 72 86 . 990

Patricia Jones 98 92 92 99 1,010

Jack Benedict 54 63 71 49 875

Rene Porter 100 62 88 74 1,180

;

RUN;

**PROC** **PRINT** data = grades;

var name standtest std;

**RUN**;

LESSON 5

/\*5.1 creates a character variable status,

whose value depends on whether or not the student's first

exam grade is less than 65\*/

**DATA** grades;

input name $ **1**-**15** e1 e2 e3 e4 p1 f1;

\* if the first exam is less than 65 indicate failed;

if (e1 LT **65**) then status = 'Failed';

DATALINES;

Alexander Smith 78 82 86 69 97 80

John Simon 88 72 86 . 100 85

Patricia Jones 98 92 92 99 99 93

Jack Benedict 54 63 71 49 82 69

Rene Porter 100 62 88 74 98 92

;

RUN;

**PROC** **PRINT** data = grades;

var name e1 status;

**RUN**;

/\*5.2 use the IN operator (alternate SAS syntax)\*/

**DATA** grades;

input name $ **1**-**15** e1 e2 e3 e4 p1 f1;

if p1 in (**98**, **99**, **100**) then project = 'Excellent';

DATALINES;

Alexander Smith 78 82 86 69 97 80

John Simon 88 72 86 . 100 85

Patricia Jones 98 92 92 99 99 93

Jack Benedict 54 63 71 49 82 69

Rene Porter 100 62 88 74 98 92

;

RUN;

**PROC** **PRINT** data = grades;

var name p1 project;

**RUN**;

/\*5.3 if-then-else statement\*/

**DATA** grades;

input name $ **1**-**15** e1 e2 e3 e4 p1 f1;

\* if the first exam is less than 65 indicate failed;

if (e1 < **65**) then status = 'Failed';

\* otherwise indicate passed;

else status = 'Passed';

DATALINES;

Alexander Smith 78 82 86 69 97 80

John Simon 88 72 86 . 100 85

Patricia Jones 98 92 92 99 99 93

Jack Benedict 54 63 71 49 82 69

Rene Porter 100 62 88 74 98 92

;

RUN;

**PROC** **PRINT** data = grades;

var name e1 status;

**RUN**;

/\*5.4 if-then-else statement with a missing value\*/

**DATA** grades;

input name $ **1**-**15** e1 e2 e3 e4 p1 f1;

\* if the fourth exam is less than 65 indicate failed;

if (e4 < **65**) then status = 'Failed';

\* otherwise indicate passed;

else status = 'Passed';

;

Alexander Smith 78 82 86 69 97 80

John Simon 88 72 86 . 100 85

Patricia Jones 98 92 92 99 99 93

Jack Benedict 54 63 71 49 82 69

Rene Porter 100 62 88 74 98 92

;

RUN;

**PROC** **PRINT** data = grades;

var name e4 status;

**RUN**;

/\* 5.5 if-then-else statment programming for missing value\*/

**DATA** grades;

length status $ **6**;

input name $ **1**-**15** e1 e2 e3 e4 p1 f1;

\* if the fourth exam is missing indicate missing;

\* else if the fourth exam is less than 65 indicate failed;

\* otherwise indicate passed;

## if (e4 = .) then status = ' ';

else if (e4 < **65**) then status = 'Failed';

else status = 'Passed';

DATALINES;

Alexander Smith 78 82 86 69 97 80

John Simon 88 72 86 . 100 85

Patricia Jones 98 92 92 99 99 93

Jack Benedict 54 63 71 49 82 69

Rene Porter 100 62 88 74 98 92

;

RUN;

**PROC** **PRINT** data = grades;

var name e4 status;

**RUN**;

/\*5.6 Use of multiple 'else if' statments arranged from most common to least common\*/

**DATA** grades;

length overall $ **10**;

input name $ **1**-**15** e1 e2 e3 e4 p1 f1;

avg = round((e1+e2+e3+e4)/**4**,**0.1**);

if (avg = **.**) then overall = 'Incomplete';

else if (avg >= **90**) then overall = 'A';

else if (avg >= **80**) and (avg < **90**) then overall = 'B';

else if (avg >= **70**) and (avg < **80**) then overall = 'C';

else if (avg >= **65**) and (avg < **70**) then overall = 'D';

else if (avg < **65**) then overall = 'F';

DATALINES;

Alexander Smith 78 82 86 69 97 80

John Simon 88 72 86 . 100 85

Patricia Jones 98 92 92 99 99 93

Jack Benedict 54 63 71 49 82 69

Rene Porter 100 62 88 74 98 92

;

RUN;

**PROC** **PRINT** data = grades;

var name avg overall;

**RUN**;

/\*5.7 illustrates use of alternate sas syntax\*/

**DATA** grades;

length overall $ **10**;

input name $ **1**-**15** e1 e2 e3 e4 p1 f1;

avg = round((e1+e2+e3+e4)/**4**,**0.1**);

if (avg EQ **.**) then overall = 'Incomplete';

else if (**90** LE avg LE **100**) then overall = 'A';

else if (**80** LE avg LT **90**) then overall = 'B';

else if (**70** LE avg LT **80**) then overall = 'C';

else if (**65** LE avg LT **70**) then overall = 'D';

else if (**0** LE avg LT **65**) then overall = 'F';

DATALINES;

Alexander Smith 78 82 86 69 97 80

John Simon 88 72 86 . 100 85

Patricia Jones 98 92 92 99 99 93

Jack Benedict 54 63 71 49 82 69

Rene Porter 100 62 88 74 98 92

;

RUN;

**PROC** **PRINT** data = grades;

var name avg overall;

**RUN**;

/\* 5.8 illustrates use of OR, AND, and OR and AND operators\*/

**DATA** grades;

input name $ **1**-**15** e1 e2 e3 e4 p1 f1;

avg = round((e1+e2+e3+e4)/**4**,**0.1**);

if ((e1 < e3) and (e1 < e4))

or ((e2 < e3) and (e2 < e4)) then adjavg = avg + **2**;

else adjavg = avg;

DATALINES;

Alexander Smith 78 82 86 69 97 80

John Simon 88 72 86 . 100 85

Patricia Jones 98 92 92 99 99 93

Jack Benedict 54 63 71 49 82 69

Rene Porter 100 62 88 74 98 92

;

RUN;

**PROC** **PRINT** data = grades;

var name e1 e2 e3 e4 avg adjavg;

**RUN**;

/\*5.9 comparing character values by first converting status to upper case\*/

**DATA** grades;

length action $ **7**

action2 $ **7**;

input name $ **1**-**15** e1 e2 e3 e4 p1 f1 status $;

if (status = 'passed') then action = 'none';

else if (status = 'failed') then action = 'contact';

else if (status = 'incomp') then action = 'contact';

if (upcase(status) = 'PASSED') then action2 = 'none';

else if (upcase(status) = 'FAILED') then action2 = 'contact';

else if (upcase(stnodaatus) = 'INCOMP') then action2 = 'contact';

DATALINES;

Alexander Smith 78 82 86 69 97 80 passed

John Simon 88 72 86 . 100 85 incomp

Patricia Jones 98 92 92 99 99 93 PAssed

Jack Benedict 54 63 71 49 82 69 FAILED

Rene Porter 100 62 88 74 98 92 PASSED

;

RUN;

**PROC** **PRINT** data = grades;

var name status action action2;

**RUN**;

/\*5.10 DO-END clause, to assign a grade of zero to any student

who missed the fourth exam, as well as notify the student that she has done so\*/

**DATA** grades;

input name $ **1**-**15** e1 e2 e3 e4 p1 f1;

if e4 = **.** then do;

e4 = **0**;

notify = 'YES';

end;

DATALINES;

Alexander Smith 78 82 86 69 97 80

John Simon 88 72 86 . 100 85

Patricia Jones 98 92 92 99 99 93

Jack Benedict 54 63 71 49 82 69

Rene Porter 100 62 88 74 98 92

;

RUN;

**PROC** **PRINT** data = grades;

var name e1 e2 e3 e4 p1 f1 notify;

**RUN**;

**LESSON 6**

/\*6.1 example of basic print procedure\*/

OPTIONS LS = **75** PS = **58** NODATE;

**DATA** basic;

input subj **1**-**4** name $ **6**-**23** clinic $ **25**-**28**

gender **30** no\_vis **32**-**33** type\_vis **35**-**37**

expense **39**-**45**;

DATALINES;

1024 Alice Smith LEWN 1 7 101 1001.98

1167 Maryann White LEWN 1 2 101 2999.34

1168 Thomas Jones ALTO 2 10 190 3904.89

1201 Benedictine Arnold ALTO 2 1 190 1450.23

1302 Felicia Ho MNMC 1 7 190 1209.94

1471 John Smith MNMC 2 6 187 1763.09

1980 Jane Smiley MNMC 1 5 190 3567.00

;

RUN;

**PROC** **PRINT** data = basic;

**RUN**;

/\* 6.2 using var statement to only pring variables name, no\_vis, and expense\*/

**PROC** **PRINT** data = basic;

var name no\_vis expense;

**RUN**;

/\*6.3 using noobs option to suppress printing of observation number\*/

**PROC** **PRINT** data = basic noobs;

var name no\_vis expense;

**RUN**;

/\* 6.4 using ID statment to choose which variable is in first column which suppresses printing of observation number\*/

**PROC** **PRINT** data = basic;

id name;

var gender expense;

**RUN**;

/\*6.5 using ID statement when observation are long\*/

OPTIONS LS = **64** PS = **58** NODATE; \*changing the LS and PS ensures the data will not fit on one line;

**PROC** **PRINT** data = basic;

id name;

var subj name clinic gender

subj no\_vis type\_vis expense;

**RUN**;

/\* 6.6 using firstobs= and OBS= to pring 2nd through 5th observations\*/

OPTIONS LS = **75** PS = **58** NODATE;

**PROC** **PRINT** data = basic (FIRSTOBS = **2** OBS = **5**);

var subj name no\_vis expense;

**RUN**;

/\* 6.7 using Where statement to print obervations meeting the criteria defined\*/

**PROC** **PRINT** data = basic;

var name no\_vis type\_vis expense;

where no\_vis > **5**;

**RUN**;

/\* 6.8 using the CONTAINS operator (?) to select observations wher the name variable contains 'Smi'\*/

**PROC** **PRINT** data = basic;

var name gender no\_vis type\_vis expense;

where name ? 'Smi';

**RUN**;

/\* 6.9 PROC SORT procedure first sorts the data by clinic and no\_vis then uses PROC PRINT to print the sorted data\*/

**PROC** **SORT** data = basic out = srtd\_basic;

by clinic no\_vis;

**RUN**;

**PROC** **PRINT** data = srtd\_basic NOOBS;

var clinic no\_vis subj name gender type\_vis expense;

**RUN**;

/\* 6.10 using PROC SORT, BY DECENDING procedure to sort and print data sorted in decending order\*/

**PROC** **SORT** data = basic out = srtd\_basic;

by descending clinic decending no\_vis;

**RUN**;

**PROC** **PRINT** data = srtd\_basic NOOBS;

var clinic no\_vis subj name gender type\_vis expense;

**RUN**;

/\* 6.11 using the SUM statement to generate report of number of total visits for patients undergoing physical therapy (type\_vis=190)\*/

**PROC** **PRINT** data = basic;

id name;

var clinic no\_vis;

where type\_vis = **190**;

sum no\_vis;

**RUN**;

\*or;

**PROC** **PRINT** data = basic;

id name;

var clinic;

where type\_vis = **190**;

sum no\_vis;

**RUN**;

/\*6.12 using the SUM, BY statement to print the subtotal for each clinic

First the data must be sorted by the variable desiered to be summed then the data can be printed by that variable\*/

**PROC** **SORT** data = basic out = srtd\_basic;

by clinic;

**RUN**;

**PROC** **PRINT** data = srtd\_basic;

by clinic;

var subj name no\_vis type\_vis expense;

sum expense;

**RUN**;

/\*6.13 using the uniform option to make sure columns in data line up from one group to another\*/

**PROC** **SORT** data = basic out = srtd\_basic;

by clinic;

**RUN**;

**PROC** **PRINT** data = srtd\_basic UNIFORM;

by clinic;

var subj name no\_vis type\_vis expense;

sum expense;

**RUN**;

/\* 6.14 Using the BY and ID statements to suppress the by variable in output by defining the same variable in each\*/

**PROC** **SORT** data = basic out = srtd\_basic;

by clinic;

**RUN**;

**PROC** **PRINT** data = srtd\_basic UNIFORM;

by clinic;

var subj name no\_vis type\_vis expense;

sum expense;

id clinic; \*same variable defined in by statement;

**RUN**;

/\* 6.15 using the PAGEBY statment to print each group on a separate page\*/

**PROC** **SORT** data = basic out = srtd\_basic;

by clinic;

**RUN**;

**PROC** **PRINT** data = srtd\_basic UNIFORM;

by clinic;

var subj name no\_vis type\_vis expense;

sum expense;

id clinic;

pageby clinic;

**RUN**;

/\* 6.16 adding TITLE and FOOTNOTE statements to clarify the output\*/

OPTIONS LS = **72** PS = **20** NODATE NONUMBER;

**PROC** **PRINT** data = basic;

title 'Our BASIC Data Set';

footnote1 'Clinic: ALTO = altoona, LEWN = Lewistown, MNMC = Mount Nittany';

footnote3 'Type\_vis: 101 = Gynecology, 190 = Physical Therapy, 187 = Cardiology';

footnote5 'Gender: 1 = female, 2 = male';

**RUN**;

FOOTNOTE; \*TURNS OFF THE FOOTNOTES;

**PROC** **PRINT**;

**RUN**;

/\*6.17 uses double spacing in output\*/

OPTIONS PS = **58** LS = **72**;

**PROC** **PRINT** data = basic NOOBS DOUBLE;

title 'Our BASIC Data Set';

var subj name clinic no\_vis type\_vis expense;

**RUN**;

/\* 6.18 label variables for printing\*/

**PROC** **PRINT** data = basic LABEL;

label name = 'Name'

no\_vis = 'Number of Visits'

type\_vis = 'Type of Visit'

expense = 'Expense';

id name;

var no\_vis type\_vis expense;

**RUN**;

title; \*removes the title previously defined;

/\* 6.19 using the SPLIT option to define where the labels should be split\*/

**PROC** **PRINT** data = basic SPLIT='/';

label name = 'Name';

label no\_vis = 'Number of/Visits';

label type\_vis = 'Type of Visit';

label expense = 'Expense';

id name;

var no\_vis type\_vis expense;

**RUN**;

/\* 6.20 use the FORMAT statement to display the expense variable using the dollar9.2 format\*/

**PROC** **PRINT** data = basic LABEL;

label name = 'Name'

clinic = 'Clinic'

expense = 'Expense';

format expense dollar9.2;

id name;

var clinic expense;

**RUN**;

/\* 7.1 simple data set\*/

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

**DATA** trees;

input type $ **1**-**16** circ\_in hght\_ft crown\_ft;

DATALINES;

oak, black 222 105 112

hemlock, eastern 149 138 52

ash, white 258 80 70

cherry, black 187 91 75

maple, red 210 99 74

elm, american 229 127 104

;

**PROC** **SORT** data = trees;

by type;

**RUN**;

**PROC** **PRINT** data = trees;

title 'Tree data';

**RUN**;

/\* 7.2 sorts data tree by height, but height is not a defined variable\*/

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

**DATA** trees;

input type $ **1**-**16** circ\_in hght\_ft crown\_ft;

DATALINES;

oak, black 222 105 112

hemlock, eastern 149 138 52

ash, white 258 80 70

cherry, black 187 91 75

maple, red 210 99 74

elm, american 229 127 104

;

**PROC** **SORT** data = trees;

by height;

**RUN**;

**PROC** **PRINT** data = trees;

title 'Tree data again';

id type;

**RUN**;

/\*extra example\*/

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

**DATA** trees;

input type $ **1**-**16** circ\_in hght\_ft crown\_ft;

volume = (**0.319**\*hght\_ft)\*(**0.0000163**\*circ\_in\*\***2**);

DATALINES;

oak, black 222 105 112

hemlock, eastern 149 138 52

ash, white 258 80 70

cherry, black 187 91 75

maple, red 210 99 74

elm, american 229 127 104

;

RUN;

**PROC** **PRINT** data = trees;

**RUN**;

/\* 7.5 adding debug option- launches DEBUGGER Source window and debugger log window\*/

**DATA** trees / DEBUG;

input type $ **1**-**16** circ\_in hght\_ft crown\_ft;

volume = (**0.319**\*hght\_ft)\*(**0.0000163**\*circ\_in\*\***2**);

DATALINES;

oak, black 222 105 112

hemlock, eastern 149 138 52

ash, white 258 80 70

cherry, black 187 91 75

maple, red 210 99 74

elm, american 229 127 104

;

RUN;

\*8.1 - missing semicolon: SAS thinks the program doesn't contain a DATA statement;

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

\*Read in the trees data set

DATA trees;

input type $ **1**-**16** circ\_in hght\_ft crown\_ft;

DATALINES;

oak, black **222** **105** **112**

hemlock, eastern **149** **138** **52**

ash, white **258** **80** **70**

cherry, black **187** **91** **75**

maple, red **210** **99** **74**

elm, american **229** **127** **104**

;

**RUN**;

\*8.2 missing semicolon after data statement- input statement becomes part of DATA statement, syntax error since $ not allowed;

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

\*Read in the trees data set;

**DATA** trees

input type $ **1**-**16** circ\_in hght\_ft crown\_ft;

DATALINES;

oak, black 222 105 112

hemlock, eastern 149 138 52

ash, white 258 80 70

cherry, black 187 91 75

maple, red 210 99 74

elm, american 229 127 104

;

RUN;

\*8.3 missing semiocolon after data statement and use of DATASTMTCHK;

OPTIONS DATASTMTCHK = ALLKEYWORDS;

\*Read in the trees data set;

**DATA** trees

input type $ **1**-**16** circ\_in hght\_ft crown\_ft;

DATALINES;

oak, black 222 105 112

hemlock, eastern 149 138 52

ash, white 258 80 70

cherry, black 187 91 75

maple, red 210 99 74

elm, american 229 127 104

;

RUN;

\*8.4 Error: option error;

**DATA** trees (ROP = crown\_ft);

input type $ **1**-**16** circ\_in hght\_ft crown\_ft;

DATALINES;

oak, black 222 105 112

hemlock, eastern 149 138 52

ash, white 258 80 70

cherry, black 187 91 75

maple, red 210 99 74

elm, american 229 127 104

;

RUN;

\*8.5 ERROR: syntax error;

**DATA** trees;

input \*type $ **1**-**16** circ\_in hght\_ft crown\_ft;

DATALINES;

oak, black 222 105 112

hemlock, eastern 149 138 52

ash, white 258 80 70

cherry, black 187 91 75

maple, red 210 99 74

elm, american 229 127 104

;

RUN;

\*8.6 ERROR: statement not valid or out of order;

**DATA** trees;

input type $ **1**-**16** circ\_in hght\_ft crown\_ft;

DATALINES;

oak, black 222 105 112

hemlock, eastern 149 138 52

ash, white 258 80 70

cherry, black 187 91 75

maple, red 210 99 74

elm, american 229 127 104

;

RUN;

**PROC** **PRINT**;

set type circ\_in hght\_ft;

**RUN**;

\*8.7 missing end quote;

**DATA** trees;

input type $ **1**-**16** circ\_in hght\_ft crown\_ft;

DATALINES;

oak, black 222 105 112

hemlock, eastern 149 138 52

ash, white 258 80 70

cherry, black 187 91 75

maple, red 210 99 74

elm, american 229 127 104

;

RUN;

**PROC** **PRINT**;

var type circ\_in hght\_ft;

title **'Some trees in Kentucky**

**title2 'D**ivision of Forestry';

RUN;

\*';**run**; \*corrects the errant program in example 8.7;

\*8.8 when raw data inconsistent with INPUT statement;

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

**DATA** trees;

input type $ **1**-**16** circ\_in hght\_ft crown\_ft;

DATALINES;

oak, black 222 1O5 112

hemlock, eastern 149 138 52

ash, white 258 8O 70

cherry, black 187 91 75

maple, red 210 99 74

elm, american 229 127 104

;

RUN;

**PROC** **PRINT** data = trees;

title 'Trees in Kentucky';

**RUN**;

\*8.9 NOTE:variable not initilized or Error: vaariable not found;

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

**DATA** trees;

input type $ **1**-**16** circ\_in hght\_ft crown\_ft;

volume = (**0.319**\*hght)\*(**0.0000163**\*circ\_in\*\***2**);

DATALINES;

oak, black 222 105 112

hemlock, eastern 149 138 52

ash, white 258 80 70

cherry, black 187 91 75

maple, red 210 99 74

elm, american 229 127 104

;

RUN;

**PROC** **PRINT** data = trees;

var type height circ\_in volume;

**RUN**;

**proc** **contents** data=trees;

**RUN**;

\*8.10 missing placeholders with list input- only 4 records are generated;

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

**DATA** trees;

input treeID circ\_in hght\_ft crown\_ft;

DATALINES;

101 222 105 112

102 149 138

103 258 80 70

104 187 91

105 210 99 74

106 229 127 104

;

RUN;

**PROC** **PRINT** data = trees;

title 'Tree data';

**RUN**;

\*8.11 using the INFILE missover option when there are missing data;

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

**DATA** trees;

INFILE DATALINES MISSOVER;

input treeID circ\_in hght\_ft crown\_ft;

DATALINES;

101 222 105 112

102 149 138

103 258 80 70

104 187 91

105 210 99 74

106 229 127 104

;

RUN;

**PROC** **PRINT** data = trees;

title 'Tree data';

**RUN**;

\*8.12 propagation of missing values;

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

**DATA** trees;

input type $ **1**-**16** circ\_in hght\_ft crown\_ft;

volume = (**0.319**\*hght\_ft)\*(**0.0000163**\*circ\_in\*\***2**);

DATALINES;

oak, black 222 . 112

hemlock, eastern . 138 52

ash, white 258 80 70

cherry, black 187 91 75

maple, red 210 99 74

elm, american 229 127 104

;

RUN;

**PROC** **PRINT** data = trees;

title 'Tree data';

**RUN**;

\*8.13 using IF statement to find missing values;

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

**DATA** trees;

input type $ **1**-**16** circ\_in hght\_ft crown\_ft;

volume = (**0.319**\*hght\_ft)\*(**0.0000163**\*circ\_in\*\***2**);

if volume = **.**;

DATALINES;

oak, black 222 . 112

hemlock, eastern . 138 52

ash, white 258 80 70

cherry, black 187 91 75

maple, red 210 99 74

elm, american 229 127 104

;

RUN;

**PROC** **PRINT** data = trees;

title 'Trees with Missing Volumes';

**RUN**;

OPTIONS PS = **58** LS = **80** NODATE NONUMBER;

LIBNAME icdb 'C:\Users\Owner\Documents\Whit Masters\STAT 480\data';

**DATA** back;

set icdb.back;

age = (v\_date - b\_date)/**365.25**;

if subj in (**110051**, **110088**, **210012**, **220004**, **230006**,

**310083**, **410012**, **420037**, **510027**, **520017**);

keep subj v\_date b\_date age sex state country race relig;

format age **4.1**;

RUN;

**PROC** **PRINT**;

title 'Output Dataset: BACK';

**RUN**;

\*creates raw ASCII data file (.dat) from permanent SAS data file;

**DATA** \_NULL\_;

set back;

file 'C:\Users\Owner\Documents\Whit Masters\STAT 480\data\back.dat';

put subj **1**-**6** @**8** b\_date mmddyy8. sex **17** race **19**

relig **21** state **23**-**24** country **26**-**27**

@**29** age **4.1** @**34** v\_date mmddyy8.;

RUN;

\* 9.3 Using the format procedure to create new character informats;

**PROC** **FORMAT**;

invalue $insex '1' = 'M'

'2' = 'F';

invalue $inrace '1' = 'Indian'

'2' = 'Asian'

'3' = 'Black'

'4' = 'White';

**RUN**;

\* 9.4 creates temp file using the defined informats;

**DATA** temp1;

infile 'C:\Users\Owner\Documents\Whit Masters\STAT 480\data\back.dat';

length sex $ **1** race $ **6**;

input subj **1**-**6** @**17** sex $insex1. @**19** race $inrace1.;

RUN;

**PROC** **CONTENTS** data=temp1;

title 'Output Dataset: TEMP1';

**RUN**;

**PROC** **PRINT** data=temp1;

var subj sex race;

**RUN**;

\*9.5 creating formats using value statement to display data;

**PROC** **FORMAT**;

value sexfmt **1** = 'Male'

**2** = 'Female';

value racefmt **1** = 'Indian'

**2** = 'Asian'

**3** = 'Black'

**4** = 'White';

**RUN**;

\*9.6 use the defined formats;

**DATA** temp2;

set back;

f\_race=race;

f\_sex=sex;

format f\_race racefmt. f\_sex sexfmt.;

RUN;

**PROC** **PRINT** data=temp2;

title 'Output Dataset: TEMP2';

var subj sex f\_sex race f\_race;

**RUN**;

**PROC** **CONTENTS** data=temp2;

**RUN**;

\*9.7using format statement for categorical variables;

**PROC** **FORMAT**;

value age2fmt **1** = 'LT 20'

**2** = '20-44'

**3** = '45-54'

**4** = 'GE 54'

OTHER = 'Missing';

**RUN**;

**DATA** temp3;

set back;

if age = **.** then age2 = **.**;

else if age lt **20** then age2 = **1**;

else if age ge **20** and age lt **45** then age2 = **2**;

else if age ge **45** and age lt **54** then age2 = **3**;

else if age ge **54** then age2 = **4**;

format age2 age2fmt.;

RUN;

**PROC** **FREQ** data=temp3;

title 'Age Frequency in TEMP3';

table age2;

**RUN**;

\*9.8 simplify the code from 9.7;

**PROC** **FORMAT**;

value agefmt LOW-<**20** = 'LT 20'

**20**-<**45** = '20-44'

**45**-<**54** = '45-54'

**54**-HIGH = 'GE 54'

OTHER = 'Missing';

**RUN**;

**PROC** **FREQ** data=back;

title 'Age Frequency in BACK';

format age agefmt.;

table age;

**RUN**;

\*9.9 creates permanent formats catalog referenced by library;

LIBNAME library 'C:\Users\Owner\Documents\Whit Masters\STAT 480\data';

**PROC** **FORMAT** library=library;

value sex2fmt **1** = 'Male'

**2** = 'Female';

value race2fmt **3** = 'Black'

**4** = 'White'

OTHER = 'Other';

**RUN**;

**DATA** temp4;

infile 'C:\Users\Owner\Documents\Whit Masters\STAT 480\data\back.dat';

input subj **1**-**6** sex **17** race **19**;

format sex sex2fmt. race race2fmt.;

RUN;

**PROC** **CONTENTS** data=temp4;

title 'Output Dataset: TEMP4';

**RUN**;

**PROC** **PRINT** data=temp4;

**RUN**;

\*9.10 create a SAS program file which contains a FORMAT procedure with desired value and invalue statements using %include;

%INCLUDE 'C:\Users\Owner\Documents\Whit Masters\STAT 480\data\backfmt.sas';

**PROC** **FREQ** data=back;

title 'Frequency Count of STATE (statefmt)';

format state statefmt.;

table state/missing;

**RUN**;

\*9.11 create SAS data set called states from state\_cd, a codebook for the variable state;

**PROC** **PRINT** data=icdb.state\_cd;

title 'Codebook for States';

**RUN**;

**DATA** states;

set icdb.state\_cd (rename = (code = start name=label));

fmtname = 'stat2fmt';

RUN;

**PROC** **FORMAT** cntlin=states;

**RUN**;

**PROC** **FREQ** data=back;

title 'Freqency Count of STATE (stat2fmt)';

format state stat2fmt.;

table state;

**RUN**;

\*9.12 use picture statement in format procedure;

**PROC** **FORMAT**;

picture subjpix LOW-HIGH = '00-0000';

**RUN**;

**PROC** **PRINT** data=back;

title 'BACK dataset with SUBJ pictured as 00-0000';

format subj subjpix.;

var subj v\_date sex;

**RUN**;

\*9.13 additional picture statements;

**DATA** temp5;

input subj ssn expens;

cards;

110051 001111111 1099.99

110088 022234567 10876.34

210012 123345567 9567.21

220004 120451207 5640.12

230006 125398710 344.46

310083 237982019 3235.09

410012 323432429 1343.03

420037 340234839 11348.29

510027 928373402 7362.79

520017 433492349 3295.09

;

RUN;

**PROC** **FORMAT**;

picture ssnpix LOW-HIGH = '999-99-9999';

picture dolpix LOW-HIGH = '000,000.00' (prefix='$' fill='\*');

**RUN**;

**PROC** **PRINT** data=temp5;

title 'Output Dataset: TEMP5. Examples of Picture Formats.';

format ssn ssnpix. expens dolpix.;

var subj ssn expens;

**RUN**;

\*9.14 FMTLIB option tells SAS to print the information about the formats listed;

**PROC** **FORMAT** FMTLIB;

title 'Selected Formats from WORK.FORMAT Catalog';

select racefmt ssnpix dolpix;

**RUN**;

**Lesson 10**

OPTIONS PS = **58** LS = **80** NODATE NONUMBER;

LIBNAME stat480 'C:\Users\Owner\Documents\Whit Masters\STAT 480\data';

**PROC** **PRINT** data = stat480.penngolf;

title 'Some Pennsylvania Golf Courses';

**RUN**;

\*10.2 basic list report;

LIBNAME stat480 'C:\Users\Owner\Documents\Whit Masters\STAT 480\data';

**PROC** **REPORT** data = stat480.penngolf NOWINDOWS;

title 'Some Pennsylvania Golf Courses';

**RUN**;

\*10.3 using headline option to underline column row, and column statement to choose columns to print;

**PROC** **REPORT** data = stat480.penngolf NOWINDOWS HEADLINE;

title 'Some Pennsylvania Golf Courses';

column Name Year Type Par Yards;

**RUN**;

\*10.4 using where statement to choose courses where type=private or resort;

**PROC** **REPORT** data = stat480.penngolf NOWINDOWS HEADSKIP HEADLINE;

title 'Some Pennsylvania Golf Courses';

column Name Year Type Par Yards;

where Type in ('Private', 'Resort');

**RUN**;

\*10.5 with define statement - using format attribute to display Yards variable;

**PROC** **REPORT** data = stat480.penngolf NOWINDOWS HEADLINE;

title 'Some Pennsylvania Golf Courses';

column Name Year Type Par Yards;

define Yards / format = comma5.0;

**RUN**;

\* 10.6 SAS column width defaults: character=max#characters, numeric=9;

**DATA** penngolf;

set stat480.penngolf;

length CourseType $ **8**;

CourseType = Type;

drop Type;

format Slope **3.**;

RUN;

**PROC** **CONTENTS** data = penngolf;

**RUN**;

**PROC** **REPORT** data = penngolf NOWINDOWS HEADLINE;

title 'Some Pennsylvania Golf Courses';

column Name Year CourseType Slope Par Yards;

define Yards / format = comma5.0;

**RUN**;

\*10.7 solves column widith problem with define, width statements;

**PROC** **REPORT** data = penngolf NOWINDOWS HEADLINE;

title 'Some Pennsylvania Golf Courses';

column Name Year CourseType Slope Par Yards;

define Yards / format = comma5.0;

define CourseType / width = **10**;

define Slope / width = **5**;

**RUN**;

\*10.8 use the define spacing statement to adjust white space b/w columns;

**PROC** **REPORT** data = penngolf NOWINDOWS HEADLINE;

title 'Some Pennsylvania Golf Courses';

column Name Year CourseType Slope Par Yards;

define Yards / format = comma5.0 spacing = **5**;

define CourseType / width = **10** spacing = **6**;

define Slope / width = **5**;

**RUN**;

\*10.9 use define statement to set headings;

**PROC** **REPORT** data = stat480.penngolf NOWINDOWS HEADLINE;

title 'Some Pennsylvania Golf Courses';

column Name Year Type Par Yards;

define Yards / format = comma5.0 'Total Yardage'

width = **7** spacing = **4**;

define Type / 'Type of Course' spacing = **6**;

**RUN**;

\*10.10 uses default forward slash split character;

**PROC** **REPORT** data = stat480.penngolf NOWINDOWS HEADLINE;

title 'Some Pennsylvania Golf Courses';

column Name Year Type Par Yards;

define Yards / format = comma5.0 'Total/Yardage'

width = **7** spacing = **4**;

define Type / 'Type of/Course' spacing = **6** width = **8**;

**RUN**;

\*10.11 center columns;

**PROC** **REPORT** data = stat480.penngolf NOWINDOWS HEADLINE;

title 'Some Pennsylvania Golf Courses';

column Name Year Type Par Yards;

define Yards / format = comma5.0 'Total/Yardage'

width = **7** spacing = **4** center;

define Type / 'Type of/Course' spacing = **6**

width = **8** center;

**RUN**;

\*10.12 change order of varibles with order in define statement;

**PROC** **REPORT** data = stat480.penngolf NOWINDOWS HEADLINE;

title 'Some Pennsylvania Golf Courses';

column Par Name Year Type Yards;

define Yards / order format = comma5.0 'Total/Yardage'

width = **7** spacing = **4** center;

define Type / 'Type of/Course' spacing = **6**

width = **8** center;

define Par / order;

**RUN**;

\*10.13 order variables in descending order;

**PROC** **REPORT** data = stat480.penngolf NOWINDOWS HEADLINE;

title 'Some Pennsylvania Golf Courses';

column Par Name Year Type Yards;

define Yards / order format = comma5.0 'Total/Yardage'

width = **7** spacing = **4** center;

define Type / 'Type of/Course' spacing = **6**

width = **8** center;

define Par / order descending;

**RUN**;

\*10.14 group by type;

**PROC** **REPORT** data = stat480.penngolf NOWINDOWS HEADLINE;

title 'Some Pennsylvania Golf Courses';

column Type Par Yards;

define Type / group 'Type of/Course' spacing = **6**

width = **8** center;

define Par / analysis 'Total/Par';

define Yards / analysis format = comma6.0 'Total/Yardage'

width = **7** spacing = **4** center;

**RUN**;

\*10.15 create report for average yards and par;

**PROC** **REPORT** data = stat480.penngolf NOWINDOWS HEADLINE;

title 'Average Size of Some PA Golf Courses by Type';

column Type Par Yards;

define Type / group 'Type of/Course' spacing = **6**

width = **8**;

define Par / mean format= **4.1**

'Average/Par' width = **7** center;

define Yards / mean format = comma6.0 'Average/Yardage'

width = **7** spacing = **4** center;

**RUN**;

\*example of report with only numeric variables;

**PROC** **REPORT** data = stat480.penngolf NOWINDOWS HEADLINE;

title 'Size of Some PA Golf Courses';

column Par Yards;

define Par / mean format= **4.1**

'Average/Par' width = **7** center;

define Yards / format = comma7.0 'Total/Yardage'

width = **7** spacing = **4** center;

**RUN**;

\*10.17 use accross variable to create horizontal groups;

**PROC** **REPORT** data = stat480.penngolf NOWINDOWS HEADLINE;

title 'Some Pennsylvania Golf Courses';

column Type Par Yards;

define Type / across 'Type of/Course' spacing = **6**

width = **8** center;

define Par / analysis 'Total/Par';

define Yards / analysis format = comma6.0 'Total/Yardage'

width = **7** spacing = **4** center;

**RUN**;

\*10.18 create a computed variable;

**PROC** **REPORT** data = stat480.penngolf NOWINDOWS HEADLINE;

title 'Some Pennsylvania Golf Courses';

column Name Slope USGA Bogey;

define Bogey / computed 'Bogey/Rating' format = **7.3**;

define USGA / format = **4.1** spacing = **5**;

compute Bogey;

Bogey = **0.186**\*Slope.sum + USGA.sum;

endcomp;

**RUN**;

**Lesson 11**

\*11.1 display contents for frist 15 observations;

OPTIONS PS = **58** LS = **80** NODATE NONUMBER;

LIBNAME icdb 'C:\Users\Owner\Documents\Whit Masters\STAT 480\data';

**PROC** **CONTENTS** data = icdb.hem2 position;

**RUN**;

**PROC** **PRINT** data = icdb.hem2 (OBS = **15**);

**RUN**;

\*11.2 basic mean procedure summarising all variables;

**PROC** **MEANS** data = icdb.hem2;

**run**;

\*11.3 mean procedure with VAR statement to summarize specific variables;

**PROC** **MEANS** data = icdb.hem2;

var wbc rbc hemog hcrit mcv mch mchc;

**RUN**;

\*11.4 sets maximum decimal places to 2, sets max field width to 10;

**PROC** **MEANS** data = icdb.hem2 MAXDEC = **2** FW = **10**;

var wbc rbc hemog hcrit mcv mch mchc;

**RUN**;

\*11.5 replace means with summary, use print option;

**PROC** **SUMMARY** data = icdb.hem2 MAXDEC = **2** FW = **10** PRINT;

var wbc rbc hemog hcrit mcv mch mchc;

**RUN**;

\*11.6 calculate sum, range, median using options;

**PROC** **MEANS** data = icdb.hem2 MAXDEC = **2** FW = **10** sum range median;

var wbc rbc hemog hcrit mcv mch mchc;

**RUN**;

\*11.7 use class statement to summarize stats for each of nine hospitals;

**PROC** **MEANS** data=icdb.hem2 fw=**10** maxdec=**2**;

var rbc wbc hcrit;

class hosp;

**RUN**;

\*11.8 class statement with 2 variables: groups by type then region;

**DATA** parks;

input ParkName $ **1**-**21** Type $ Region $ Museums Camping;

DATALINES;

Dinosaur NM West 2 6

Ellis Island NM East 1 0

Everglades NP East 5 2

Grand Canyon NP West 5 3

Great Smoky Mountains NP East 3 10

Hawaii Volcanoes NP West 2 2

Lava Beds NM West 1 1

Statue of Liberty NM East 1 0

Theodore Roosevelt NP West 2 2

Yellowstone NP West 9 11

Yosemite NP West 2 13

;

RUN;

**PROC** **MEANS** data = parks fw = **10** maxdec = **0** sum;

var museums camping;

class type region;

**RUN**;

\*11.9 using the by statment to categorize observations;

**PROC** **SORT** data = parks out = srtdparks;

by type region;

**RUN**;

**PROC** **MEANS** data = srtdparks fw = **10** maxdec = **0** sum min max;

var museums camping;

by type region;

**RUN**;

\*11.10 using output statement to create temporary data set;

**PROC** **MEANS** data=icdb.hem2 NOPRINT;

var rbc wbc hcrit;

class hosp;

output out = hospsummary

mean = MeanRBC MeanWBC MeanHCRIT

median = MedianRBC MedianWBC MedianHCRIT;

**RUN**;

**PROC** **PRINT**;

title 'Hospital Statistics';

**RUN**;

\*11.11 use proc summary procedure to create same output as in 11.10;

**PROC** **SUMMARY** data=icdb.hem2;

var rbc wbc hcrit;

class hosp;

output out = hospsummary

mean = MeanRBC MeanWBC MeanHCRIT

median = MedianRBC MedianWBC MedianHCRIT;

**RUN**;

**PROC** **PRINT**;

title 'Hospital Statistics';

**RUN**;

\*11.12 use the by statment instead of the class statement to generate summary;

**PROC** **SORT** data = icdb.hem2 out = srtdhem2;

by hosp;

**RUN**;

**PROC** **MEANS** data=srtdhem2 NOPRINT;

var rbc wbc hcrit;

by hosp;

output out = hospsummary

mean = MeanRBC MeanWBC MeanHCRIT

median = MedianRBC MedianWBC MedianHCRIT;

**RUN**;

**PROC** **PRINT**;

title 'Hospital Statistics';

**RUN**;

\*11.13 simple iteraction plot: two class variables: sex and race with analysis variable of education level;

**PROC** **SORT** data=icdb.back out=back;

by sex race;

**RUN**;

**PROC** **MEANS** data=back noprint;

by sex race;

var ed\_level;

output out=meaned mean=mn\_edlev;

**RUN**;

**PROC** **PRINT**;

title 'Mean Education Level for Sex and Race combinations';

**RUN**;

**PROC** **PLOT** data=meaned;

title 'Interaction Plot of SEX, RACE, and Mean Education Level';

plot mn\_edlev\*race=sex;

**RUN**;

\*11.14 simple example of univariate;

**PROC** **UNIVARIATE** data = icdb.hem2;

title 'Univariate Analysis of RBC';

var rbc;

**RUN**;

\*11.15 use NORMAL PLOT option;

**PROC** **UNIVARIATE** data = icdb.hem2 NORMAL PLOT;

title 'Univariate Analysis of RBC with NORMAL and PLOT Options';

var rbc;

**RUN**;

\*11.16 use ID statement to specify five largest and five smallest observations;

**PROC** **UNIVARIATE** data = icdb.hem2;

title 'Univariate Analysis of RBC with ID Option';

var rbc;

id subj;

**RUN**;

***LESSON 12***

\*12.1 one way frequency table;

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

LIBNAME icdb 'C:\Users\Owner\Documents\Whit Masters\STAT 480\data';

**PROC** **FREQ** data=icdb.back;

title 'Frequency Count of SEX';

tables sex;

**RUN**;

\*12.2 freq table using NOCUM option to suppress printing of cum frequencies and cum percentages;

**PROC** **FREQ** data=icdb.back;

title 'Frequency Count of SEX: No Cumulative Stats';

tables sex/nocum;

**RUN**;

\*12.3 2 x one-way frequency tables using PAGE option to print each table on separate page;

**PROC** **FREQ** data=icdb.back page;

title 'Frequency Count of SEX and RACE';

tables sex race;

**RUN**;

\*12.4 create a one-way frequency table for variable ed\_level for each level of sex, must sort data by sex first

results in two one-way frequency tables, one for each sex;

**PROC** **SORT** data=icdb.back out=s\_back;

by sex;

**RUN**;

**PROC** **FREQ** data=s\_back;

title 'Frequency Count of Education Level within Each Level of Sex';

tables ed\_level;

by sex;

**RUN**;

\*12.5 using MISSING option to tell SAS to include missing values as nonmissing values and include in all calculations,

using MISSPRINT option to tell SAS to treat missing values as nonmissing for frequencies only, not in other calculations;

**PROC** **FREQ** data=icdb.back;

title 'One-way Table of State: with MISSING Option';

tables state/missing;

**RUN**;

**PROC** **FREQ** data=icdb.back;

title 'One-way Table of State: with MISSPRINT Option';

tables state/missprint;

**RUN**;

\*12.6 creating a two-way table for sex and ed\_level;

**PROC** **FREQ** data=icdb.back;

title 'Crosstabulation of Education Level and Sex';

tables ed\_level\*sex;

**RUN**;

\*12.7 create three-way table of sex, job\_chng and ed\_level

result: SAS creates two 2-way tables of job\_chng and ed\_level, one for each sex (strata);

**PROC** **FREQ** data=icdb.back;

title '3-way Table of Sex, Job Change, and Ed. Level';

tables sex\*job\_chng\*ed\_level;

**RUN**;

\*12.8 NOROW, NOCOL, NOPERCENT options suppress printing of row percentages, column percentages, and

joint cell percentages, respectively;

**PROC** **FREQ** data=icdb.back;

title 'Crosstabulation of SEX and RACE: No percents';

tables race\*sex/norow nocol nopercent;

**RUN**;

\*12.9 requests EXPECTED and CELLCHI2 stats be printed;

**PROC** **FREQ** data=icdb.back;

title 'Crosstabulation of SEX and RACE: With Expecteds';

tables race\*sex/expected cellchi2 norow nocol nopercent;

**RUN**;

\*12.10 use LIST table option to generate list output for crosstabulations;

**PROC** **FREQ** data=icdb.back;

title 'Crosstabulation of SEX and RACE: In List Format';

tables sex\*race/list;

**RUN**;

\*12.11 FREQ procedure with CROSSLIST option;

**PROC** **FREQ** data=icdb.back;

title 'Crosstabulation of SEX and RACE: In Crosslist Format';

tables sex\*race/crosslist;

**RUN**;

\*12.12 tells SAS to create an output that contains counts and percentages for each

combination of sex and race;

**PROC** **FREQ** data=icdb.back;

tables sex\*race/out=sexfreq noprint;

**RUN**;

**PROC** **PRINT**;

title 'Output Dataset: SEXFREQ';

**RUN**;

\*12.13 use SPARSE table option to generate a data set containing a record for each possible

combination of sex and race, output: 16 rows;

**PROC** **FREQ** data=icdb.back;

tables sex\*race/out=sexfreq noprint sparse;

**RUN**;

**PROC** **PRINT**;

title 'Output Dataset: SEXFREQ with SPARSE option';

**RUN**;

\*12.14 displays variable names for first 15 observations;

OPTIONS NOFMTERR;

**PROC** **CONTENTS** data = icdb.analysis1 position;

title 'The Analysis data set';

**RUN**;

**PROC** **PRINT** data = icdb.analysis1 (OBS = **15**);

title 'The Analysis data set';

**RUN**;

\*12.15 Chi-square to determine whether there is an association b/w cyst\_hb and ctr;

**PROC** **FORMAT**;

value cystfmt **0** = 'Local'

**1** = 'Both'

**2** = 'Hydro'

OTHER = 'Nothing';

**RUN**;

**PROC** **FREQ** data=icdb.analysis1;

title 'Chi-square Test of Hospital and Cystoscopy Procedure: CHISQ';

format cyst\_hb cystfmt.;

tables ctr\*cyst\_hb/nopercent nocol missing chisq;

**RUN**;

\*12.16 uses MEASURES tables option to obtain basic set of measures association and SE;

**PROC** **FORMAT**;

value cystfmt **0** = 'Local'

**1** = 'Both'

**2** = 'Hydro'

OTHER = 'Nothing';

**RUN**;

**PROC** **FREQ** data=icdb.analysis1;

title 'Chi-square Test of Hospital and Cystoscopy Procedure: MEASURES';

format cyst\_hb cystfmt.;

tables ctr\*cyst\_hb/nopercent nocol missing measures;

**RUN**;

\*12.17 use CMH test for association b/w cyst\_hb and sym\_1 while adjusting for the ctr;

OPTIONS NOFMTERR;

**PROC** **FORMAT**;

value cystfmt **0** = 'Local'

**1** = 'Both'

**2** = 'Hydro'

OTHER = 'Nothing';

**RUN**;

**PROC** **FREQ** data=icdb.analysis1;

title 'Chi-square Test of Hospital and Cystoscopy Procedure: CMH';

title2 'Adjusting for Ctr';

format cyst\_hb cystfmt.;

tables ctr\*cyst\_hb\*sym\_1/nopercent nocol cmh;

**RUN**;

**Lesson 13**

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

LIBNAME stat481 'C:\Users\Owner\Documents\Whit Masters\STAT 481\data';

**DATA** penngolf;

set stat481.penngolf;

RUN;

**PROC** **PRINT** data = penngolf NOOBS;

title 'The penngolf data set';

**RUN**;

\*13.2 if-then-delete statement to exclude observations;

**DATA** penngolf;

set stat481.penngolf;

if par = **70** then DELETE;

RUN;

**PROC** **PRINT** data = penngolf NOOBS;

title 'The penngolf data set';

**RUN**;

\*13.3 if statement to include observations;

**DATA** penngolf;

set stat481.penngolf;

if par GT **70**;

RUN;

**PROC** **PRINT** data = penngolf NOOBS;

title 'The penngolf data set';

**RUN**;

\*13.4 use DROP statement to reduce number of variables in data set;

**DATA** penngolf;

set stat481.penngolf;

drop Architect;

RUN;

**PROC** **PRINT** data = penngolf NOOBS;

title 'The penngolf data set';

**RUN**;

\*13.5 use an assignment statement to create a new variable;

**DATA** penngolf;

set stat481.penngolf;

Bogey = **0.186**\*Slope + USGA;

RUN;

**PROC** **PRINT** data = penngolf NOOBS;

title 'The penngolf data set';

var Name Yards Par Slope USGA Bogey;

**RUN**;

\*13.6 changing variable attributes: use the label statement to create descriptive labels and format statement to format display;

**DATA** penngolf;

set stat481.penngolf;

Bogey = **0.186**\*Slope + USGA;

label Bogey = 'Bogey Rating'

USGA = 'USGA Rating'

Slope = 'Course Slope'

Par = 'Course Par'

Yards = 'Total Yardage';

format Bogey **4.1** Yards comma5.;

RUN;

**PROC** **PRINT** data = penngolf NOOBS LABEL;

title 'The penngolf data set';

var Name Yards Par Slope USGA Bogey;

**RUN**;

\*13.7 create an accumulator variable using a sum statement;

**DATA** penngolf;

set stat481.penngolf;

TotalYards + Yards;

RUN;

**PROC** **PRINT** data = penngolf NOOBS;

title 'The penngolf data set';

var Name Yards TotalYards;

**RUN**;

\*13.8 use SELECT group to create a numeric varaible, similar to IF-then-else statements (select-when-otherwise);

**DATA** penngolf;

set stat481.penngolf;

select (Type);

when ("Resort") AssnFee = **5000**;

when ("Private") AssnFee = **4000**;

when ("SemiPri") AssnFee = **2000**;

when ("Public") AssnFee = **1000**;

otherwise AssnFee = **.**;

end;

format Yards AssnFee comma5.;

RUN;

**PROC** **PRINT** data = penngolf NOOBS;

title 'The penngolf data set';

var Name Yards Type AssnFee;

**RUN**;

\*13.8- b alternative programing for select group;

**DATA** penngolf;

set stat481.penngolf;

select;

when (Type = "Resort") AssnFee = **5000**;

when (Type = "Private") AssnFee = **4000**;

when (Type = "SemiPri") AssnFee = **2000**;

when (Type = "Public") AssnFee = **1000**;

otherwise AssnFee = **.**;

end;

format Yards AssnFee comma5.;

RUN;

**PROC** **PRINT** data = penngolf NOOBS;

title 'The penngolf data set';

var Name Yards Type AssnFee;

**RUN**;

\*13.9 use select group to crete a new variable;

**DATA** penngolf;

set stat481.penngolf;

LENGTH Course $ **7**;

select;

when (Yards GE **6700**) Course = 'Long';

when (**6400** LE Yards LT **6700**) Course = 'Medium';

when (Yards LT **6400**) Course = 'Short';

otherwise Course = 'Unknown';

end;

format Yards comma5.;

RUN;

**PROC** **PRINT** data = penngolf NOOBS;

title 'The penngolf data set';

var Name Yards Course;

**RUN**;

\* 13.10 emphasising the need for a length statement, program identical to 13.9 but misssing length stmt;

**DATA** penngolf;

set stat481.penngolf;

select;

when (Yards = **.**) Course = 'Unknown';

when (Yards LT **6400** ) Course = 'Short';

when (Yards LT **6700**) Course = 'Medium';

otherwise Course = 'Long';

end;

format Yards comma5.;

RUN;

**PROC** **PRINT** data = penngolf NOOBS;

title 'The penngolf data set';

var Name Yards Course;

**RUN**;

\*13.11 sort data by store,find first and last observations in each group;

LIBNAME stat481 'C:\Users\Owner\Documents\Whit Masters\STAT 481\data';

**PROC** **SORT** data = stat481.sales out = srtdsales;

by Store;

**RUN**;

**DATA** storesales;

set srtdsales;

by Store;

firstStore = first.Store;

lastStore = last.Store;

RUN;

**PROC** **PRINT** data= storesales;

title 'Behind the scene view of the storesales data set';

id Store;

**RUN**;

\*13.12 determine first and last observations for each store and determine total sales ;

**PROC** **SORT** data = stat481.sales out = srtdsales;

by Store;

**RUN**;

**DATA** storesales;

set srtdsales;

by Store;

if first.Store then StoreTotal = **0**;

StoreTotal + Sales;

if last.Store; \*subsetting if statement;

drop Dept Quarter Sales;

format StoreTotal Dollar13.2;

RUN;

**PROC** **PRINT** data= storesales;

title 'Sales by Store';

id Store;

sum StoreTotal;

**RUN**;

\*13.13 sort sas data by store and dept, look at first and last observations in subgroups;

**PROC** **SORT** data = stat481.sales out = srtdsales;

by Store Dept;

**RUN**;

**DATA** storesales;

set srtdsales;

by Store Dept;

firstStore = first.Store;

lastStore = last.Store;

firstDept = first.Dept;

lastDept = last.Dept;

RUN;

**PROC** **PRINT** data = storesales;

title 'Behind the scene view of the storesales data set';

id Store;

**RUN**;

\*13.14 use set and by statements to identify first and last observations for store and dept, then

determine total sales for each dept within each store, displays depttotal and overall total;

**PROC** **SORT** data = stat481.sales out = srtdsales;

by Store Dept;

**RUN**;

**DATA** storesales;

set srtdsales;

by Store Dept;

if first.Dept then DeptTotal = **0**;

DeptTotal + Sales;

if last.Dept;

drop Quarter Sales;

format DeptTotal Dollar13.2;

RUN;

**PROC** **PRINT** data = storesales NOOBS;

title 'Sales by Store and Department';

sum DeptTotal;

**RUN**;

\*13.15 use accumulator variable to determine overall sales;

LIBNAME stat481 'C:\Users\Owner\Documents\Whit Masters\STAT 481\data';

**DATA** storesales;

set stat481.sales;

TotalSales + Sales;

format Sales TotalSales Dollar13.2;

RUN;

**PROC** **PRINT** data = storesales NOOBS;

title;

**RUN**;

\*13.16 use the SET statements END= option and subsetting if statement to

write only the last observation in the input data set, output: 1 observation, overall total sales of all stores;

**DATA** storesales;

set stat481.sales end=last;

TotalSales + Sales;

format Sales TotalSales Dollar13.2;

drop Store Dept Sales Quarter;

if last;

RUN;

**PROC** **PRINT** data = storesales NOOBS;

title;

**RUN**;

\*13.17 use SET statement to crete temporary data set, create new variable salestax;

**DATA** tax;

set stat481.sales;

SalesTax = Sales \* **0.06**;

RUN;

**PROC** **PRINT** data = tax;

title 'The tax data set';

**RUN**;

**Lesson 14**

/\*14.1 uses the OBS= option to create temp data set with first 25 obervations\*/

OPTIONS PS=**58** LS=**80** NODATE NONUMBER;

LIBNAME icdb 'C:\Users\Owner\Documents\Whit Masters\STAT 481\data';

**DATA** back;

set icdb.back (obs=**25**);

RUN;

**PROC** **PRINT** data=back;

title 'A Subset of the Background Data Set';

**RUN**;

\*14.2 FIRSTOBS= and OBS= options to print observations 7 through 20;

LIBNAME icdb 'C:\Users\Owner\Documents\Whit Masters\STAT 481\data';

**DATA** back1;

set icdb.back (FIRSTOBS=**7** OBS=**20**);

RUN;

**PROC** **PRINT** data=back1;

title 'Output Dataset: BACK1';

**RUN**;

\*14.3 use the keep option in SET statement to determine which variables are read from the original data set;

**DATA** back2;

set back1 (keep = subj v\_date b\_date);

age = (v\_date - b\_date)/**365**; \* Calculate AGE in years;

format age **4.1**;

RUN;

**PROC** **PRINT** data=back2;

title 'Output Dataset: BACK2';

**RUN**;

\*14.4 use the drop option in set statement to determine which variables are dropped from the origincal

dataset;

**DATA** back3;

set back1 (drop = r\_id race ethnic relig mar\_st

ed\_level emp\_st job\_chng income);

RUN;

**PROC** **PRINT** data=back3;

title 'Output Dataset: BACK3';

**RUN**;

\*14.5 keep option in data step to keep variables when writing to new data set;

**DATA** back2a (keep = subj v\_date b\_date age);

set back1;

age = (v\_date - b\_date)/**365**; \* Calculate AGE in years;

format age **4.1**;

RUN;

**PROC** **PRINT** data=back2a;

title 'Output Dataset: BACK2A';

**RUN**;

\*14.6 using the drop option in the data step to drop variables before creating the new data set;

**DATA** back3a (drop = r\_id race ethnic relig mar\_st

ed\_level emp\_st job\_chng income);

set back1;

RUN;

**PROC** **PRINT** data=back3a;

title 'Output Dataset: BACK3A';

**RUN**;

\*14.7 using the keep options in the DATA and SET statements;

**DATA** back6 (keep = subj age);

set back1 (keep= subj v\_date b\_date);

age = (v\_date - b\_date)/**365**;

format age **4.1**;

RUN;

**PROC** **PRINT** data=back6;

title 'Output Dataset: BACK6';

**RUN**;

\*14.8 using the keep statement in DATA step, similar to Keep option in data step. affects which variables are

written from the program data vector to the SAS data set;

**DATA** back4;

set back1;

age = (v\_date - b\_date)/**365**;

format age **4.1**;

keep subj v\_date b\_date;

RUN;

**PROC** **PRINT** data=back4;

title 'Output Dataset: BACK4';

**RUN**;

\*14.9 similar to 14.8, use the drop statement to determine which variables are not written from the porgram

data vector to the SAS data set;

**DATA** back5;

set back1;

age = (v\_date - b\_date)/**365**;

format age **4.1**;

drop r\_id race ethnic relig mar\_st ed\_level emp\_st job\_chng

income sex state country;

RUN;

**PROC** **PRINT** data=back5;

title 'Output Dataset: BACK5';

**RUN**;

\*14.10 using the WHERE= option to select observations from SAS data set and divide them into smaller data

sets, SAS selects which observations to write from the program data vector to the output data set;

**DATA** temple (where = (int(subj/**10000**)=**23**))

okla (where = (int(subj/**10000**)=**31**));

set icdb.back;

drop r\_id race ethnic relig mar\_st ed\_level

emp\_st job\_chng income;

RUN;

**PROC** **PRINT** data=temple;

title 'Output Dataset: TEMPLE';

**RUN**;

**PROC** **PRINT** data=okla;

title 'Output Dataset: OKLA';

**RUN**;

\*14.11 using the where option in a set statement, selection process occurs when SAS reads observations from

data set to temp dataset;

**DATA** temple2;

set icdb.back (where = (int(subj/**10000**)=**23**));

drop r\_id race ethnic relig mar\_st ed\_level emp\_st job\_chng income;

RUN;

**PROC** **PRINT** data = temple2;

title 'Output Dataset: TEMPLE2';

**RUN**;

\*14.12 use the rename option in the set statement, changes sex to gender and bdate to birth when program

data vector is created, therefore the new variable names must be used in the data step;

**DATA** back7 (keep = subj gender v\_date birth age);

set back3 (rename=(sex=gender b\_date=birth));

age = (v\_date - birth)/**365**; \*MUST use NEW name for date of birth;

RUN;

**PROC** **PRINT** data=back7;

title 'Output Dataset: BACK7';

**RUN**;

\*14.13 use the RENAME option to rename variables in data statement, changes variable name when SAS writes

data to output data set;

**DATA** back8 (rename=(sex=gender b\_date=birth)

keep = subj sex v\_date b\_date age);

set back3;

age = (v\_date - b\_date)/**365**; \*MUST use OLD name for date of birth;

RUN;

**PROC** **PRINT** data=back8;

title 'Output Dataset: BACK8';

**RUN**;

\*14.14 use IN option when appedning okla data set to temple data set: temporary variable okie created

and set to 1 if observation came from okla data set, okie used to create permanent variable hospital;

**DATA** back9;

set temple okla (in=okie);

if okie = **1** then hospital = **31**;

else if okie = **0** then hospital = **23**;

RUN;

**PROC** **PRINT** data=back9;

title 'Output Dataset: BACK9';

**RUN**;

\*14.15 programming trick when using in option, uses SAS assumption (if varname=1 is equivalent to if varname);

**DATA** back10;

set temple okla (in=inokie);

if inokie then hospital = **31**;

else hospital = **23**;

RUN;

**PROC** **PRINT** data=back10;

title 'Output Dataset: BACK10';

**RUN**;

/\*15.1 one to one reading to combine two data sets\*/

**DATA** patients;

input ID Sex $ Age;

DATALINES;

1157 F 33

2395 F 48

1098 M 39

4829 F 24

3456 M 30

5920 M 41

1493 F 42

;

RUN;

**DATA** scale;

input ID Height Weight;

DATALINES;

1157 65 122

2395 64 130

1098 70 178

4829 67 142

3456 72 190

5920 71 188

;

RUN;

**DATA** one2oneread;

set patients;

set scale;

RUN;

**PROC** **PRINT** NOOBS;

title 'The one2oneread data set';

**RUN**;

\*15.2 one to one reading in reverse order of 15.1;

**DATA** one2oneread2;

set scale;

set patients;

RUN;

**PROC** **PRINT** NOOBS;

title 'The one2oneread2 data set';

**RUN**;

\*15.3 combining two data sets using one to one reading;

**DATA** one;

input ID VarA $ VarB $;

DATALINES;

10 A1 B1

20 A2 B2

30 A3 B3

;

RUN;

**DATA** two;

input ID VarB $ VarC $;

DATALINES;

40 B4 C1

50 B5 C2

;

RUN;

**DATA** onetwo;

set one;

set two;

RUN;

**PROC** **PRINT** data = onetwo NOOBS;

title 'The onetwo data set';

**RUN**;

\*15.4 use one-to-one merge to generate data set with # of obs=largest data set;

**DATA** one2onemerge;

merge patients scale;

RUN;

**PROC** **PRINT** NOOBS;

title 'The one2onemerge data set';

**RUN**;

\*15.5 one-to-one merge of two data sets showing overwritng of similar variables;

**DATA** one;

input ID VarA $ VarB $;

DATALINES;

10 A1 B1

20 A2 B2

30 A3 B3

;

RUN;

**DATA** two;

input ID VarB $ VarC $;

DATALINES;

40 B4 C1

50 B5 C2

;

RUN;

**DATA** onetwomerged;

merge one two;

RUN;

**PROC** **PRINT** data = onetwomerged NOOBS;

title 'The onetwomerged data set';

**RUN**;

\*15.6 contatenate two data sets to create tall dataset;

**DATA** store1;

input Store Day $ Sales;

DATALINES;

1 M 1200

1 T 1435

1 W 1712

1 R 1529

1 F 1920

1 S 2325

;

RUN;

**DATA** store2;

input Store Day $ Sales;

DATALINES;

2 M 2215

2 T 2458

2 W 1798

2 R 1692

2 F 2105

2 S 2847

;

RUN;

**DATA** bothstores;

set store1 store2;

RUN;

**PROC** **PRINT** data = bothstores NOOBS;

title 'The bothstores data set';

**RUN**;

\*15.7 concatenate two datasets;

**DATA** one;

input ID VarA $ VarB $;

DATALINES;

10 A1 B1

20 A2 B2

30 A3 B3

;

RUN;

**DATA** two;

input ID VarB $ VarC $;

DATALINES;

40 B4 C1

50 B5 C2

;

RUN;

**DATA** onetopstwo;

set one two;

RUN;

**PROC** **PRINT** data = onetopstwo NOOBS;

title 'The onetopstwo data set';

**RUN**;

\*15.8 concatenating variables that are not the same: yields error;

**DATA** store3;

input Store $ **1** Day $ **3** Sales **5**-**8**;

DATALINES;

1 M 1200

1 T 1435

1 W 1712

1 R 1529

1 F 1920

1 S 2325

;

RUN;

**DATA** store4;

input Store **1** Day $ **3** Sales **5**-**8**;

DATALINES;

2 M 2215

2 T 2458

2 W 1798

2 R 1692

2 F 2105

2 S 2847

;

RUN;

**DATA** bothstores2;

set store3 store4;

RUN;

**PROC** **PRINT** data = bothstores2 NOOBS;

title 'The bothstores2 data set';

**RUN**;

\*converts the format of the store variable from character to numeric using PUT statement to allow for concatenating;

**DATA** store4 (rename = (Store2 = Store));

set store4;

Store2 = put(Store,**1.**);

drop Store;

RUN;

**PROC** **CONTENTS** data = store3;

title 'Contents of store3';

**RUN**;

**PROC** **CONTENTS** data = store4;

title 'Contents of store4';

**RUN**;

\*concatenating store 3 and store 4 data sets;

**DATA** bothstores2;

set store3 store4;

RUN;

**PROC** **PRINT** data = bothstores2 NOOBS;

title 'The bothstores2 data set';

**RUN**;

\*15.9 first create two data sets with different labels and formats for the sales variable;

**DATA** store5;

input Store **1** Day $ **3** Sales **5**-**8**;

format Sales comma5.;

label Sales = 'Total Sales';

DATALINES;

1 M 1200

1 T 1435

1 W 1712

1 R 1529

1 F 1920

1 S 2325

;

RUN;

**DATA** store6;

input Store **1** Day $ **3** Sales **5**-**8**;

format Sales dollar6.;

label Sales = 'Sales for Day';

DATALINES;

2 M 2215

2 T 2458

2 W 1798

2 R 1692

2 F 2105

2 S 2847

;

RUN;

**PROC** **CONTENTS** data = store5;

title 'Contents of the store5 data set';

**RUN**;

**PROC** **CONTENTS** data = store6;

title 'Contents of the store6 data set';

**RUN**;

\*concatenate store5 and store6 data sets, note the label and format of sales variable which should be that

of the store5 dataset;

**DATA** bothstores3;

set store5 store6;

RUN;

**PROC** **PRINT** data = bothstores3 NOOBS LABEL;

title 'The bothstores3 data set';

**RUN**;

\*concatenate store6 and store5 data sets, note the label and format of sales variable which should be that

of the store6 dataset;

**DATA** bothstores4;

set store6 store5;

RUN;

**PROC** **PRINT** data = bothstores4 NOOBS LABEL;

title 'The bothstores4 data set';

**RUN**;

\*15.10 create two data sets with different lengths for the day and store variable;

**DATA** store7;

length Store **4**;

input Store **1** Day $ **3**-**5** Sales **7**-**10**;

DATALINES;

1 Mon 1200

1 Tue 1435

1 Wed 1712

1 Thu 1529

1 Fri 1920

1 Sat 2325

;

RUN;

**DATA** store8;

input Store **1** Day $ **3** Sales **5**-**8**;

DATALINES;

2 M 2215

2 T 2458

2 W 1798

2 R 1692

2 F 2105

2 S 2847

;

RUN;

**PROC** **CONTENTS** data = store7;

title 'Contents of the store7 data set';

**RUN**;

**PROC** **CONTENTS** data = store8;

title 'Contents of the store8 data set';

**RUN**;

\*concatenate store 7 and store 8, note the length of the store and day variables will be taht

of store7 data set;

**DATA** bothstores5;

set store7 store8;

RUN;

**PROC** **PRINT** data = bothstores5 NOOBS LABEL;

title 'The bothstores5 data set';

**RUN**;

**PROC** **CONTENTS** data = bothstores5;

title 'Contents of the bothstores5 data set';

**RUN**;

\*reverse the order of concatenation above note the length of the store and day variables will be taht

of store8 data set;

**DATA** bothstores6;

set store8 store7;

RUN;

**PROC** **PRINT** data = bothstores6 NOOBS LABEL;

title 'The bothstores5 data set';

**RUN**;

**PROC** **CONTENTS** data = bothstores6;

title 'Contents of the bothstores5 data set';

**RUN**;

\*15.11 interleave data sets one and two;

**DATA** one;

input year x;

DATALINES;

2000 1

2001 2

2002 3

2003 4

;

RUN;

**DATA** two;

input year x;

DATALINES;

2001 5

2002 6

2003 7

2004 8

;

RUN;

**DATA** three;

set one two;

by year;

RUN;

**PROC** **PRINT** data = three NOOBS;

title 'The interleaved three data set';

**RUN**;

\*15.12 interleaving is same as two-step process of concatenating data sets then sorting them;

**DATA** unsortedfour;

set one two;

RUN;

**PROC** **PRINT** data = unsortedfour NOOBS;

title 'The unsortedfour data set';

**RUN**;

**PROC** **SORT** data = unsortedfour out = four;

by year;

**RUN**;

**PROC** **PRINT** data = four NOOBS;

title 'The four data set';

**RUN**;

\*15.13 attempts a one-to one merge which does not work;

**DATA** firstnames;

input subj **5**-**9** name $ **10**-**16** gender **19**

height **21**-**22** weight **24**-**26**;

CARDS;

1024 Alice 1 65 125

1167 Maryann 1 68 140

1168 Thomas 2 68 190

1201 Benny 2 68 190

1302 Felicia 1 63 115

;

RUN;

**DATA** lastnames;

input name $ **4**-**9** sysbp **11**-**13** diasbp **14**-**15**;

CARDS;

Smith 120 80

White 130 90

Jones 125 72

Arnold 135 95

;

RUN;

**DATA** alldata;

merge firstnames lastnames;

RUN;

**PROC** **PRINT** data=alldata NOOBS;

title 'The alldata data set';

**RUN**;

\*first change the variable name in each data set, then merge them;

**DATA** alldata2;

merge firstnames (rename = (name=f\_name))

lastnames (rename = (name=l\_name));

RUN;

**PROC** **PRINT** data=alldata2 NOOBS;

title 'The alldata2 data set';

**RUN**;

Lesson 16

\*16.1 simple match-merge, each obs in the first data set matches exactly one

obs in the second dataset, data sorted in ascending order;

**DATA** demog;

input subj gender $ age;

cards;

1000 M 42

1001 M 20

1002 F 53

1003 F 40

1004 M 29

;

RUN;

**DATA** status;

input subj disease $ test $ ;

cards;

1000 Y Y

1001 N Y

1002 N N

1003 Y Y

1004 N N

;

RUN;

**DATA** patients;

merge demog status;

by subj;

RUN;

**PROC** **PRINT** data=patients NOOBS;

title 'The patients data set';

**RUN**;

\*16.2 simple match-merge, each obs in the first data set matches exactly one

obs in the second dataset, data sorted in descending order;

**PROC** **SORT** data = demog out = descdemog;

by descending subj;

**RUN**;

**PROC** **SORT** data = status out = descstatus;

by descending subj;

**RUN**;

**DATA** descpatients;

merge descdemog descstatus;

by descending subj;

RUN;

**PROC** **PRINT** data = descpatients NOOBS;

title 'The descpatients data set';

**RUN**;

\*16.3 datasets to be merged contain diff number of obs, each obs in the

first data set matches exactly oneobs in the second dataset;

**DATA** newdemog;

input subj gender $ age;

cards;

1000 M 42

1001 M 20

1002 F 53

1003 F 40

1004 M 29

1005 F 29

;

RUN;

**DATA** status;

input subj disease $ test $ ;

cards;

1000 Y Y

1001 N Y

1002 N N

1003 Y Y

1004 N N

;

RUN;

**DATA** newpatients;

merge newdemog status;

by subj;

RUN;

**PROC** **PRINT** data=newpatients NOOBS;

title 'The newpatients data set';

**RUN**;

\*merging data sets when there are common variables other than the linking variable,

data sets share subj and v\_date, output: v-date overwritten by morestatus data, 6 variables;

**DATA** moredemog;

input subj gender $ age v\_date mmddyy8.;

format v\_date mmddyy8.;

DATALINES;

1000 M 42 03/10/96

1001 M 20 02/19/96

1002 F 53 02/01/96

1003 F 40 12/31/95

1004 M 29 01/10/97

;

RUN;

**DATA** morestatus;

input subj disease $ test $ v\_date mmddyy8.;

format v\_date mmddyy8.;

DATALINES;

1000 Y Y 03/17/96

1001 N Y 03/01/96

1002 N N 02/18/96

1003 Y Y 01/15/96

1004 N N 02/01/97

;

RUN;

**DATA** morepatients;

merge moredemog morestatus;

by subj;

RUN;

**PROC** **PRINT** data=morepatients NOOBS;

title 'The morepatients data set';

**RUN**;

\*reversing the order of match-merging;

**DATA** morepatients2;

merge morestatus moredemog;

by subj;

RUN;

**PROC** **PRINT** data=morepatients2 NOOBS;

title 'The morepatients2 data set';

**RUN**;

\*16.5 match-merge were obs in first data set match one or more obs in the second

data set, ie. two 2004 obs in one set, 3 2004 obs in second dataset;

**DATA** salesone;

input year prd sales;

DATALINES;

2004 1 100

2004 2 200

2005 3 300

2006 4 400

2007 5 500

2008 6 600

;

RUN;

**DATA** salestwo;

input year loc sales;

DATALINES;

2004 7 700

2004 8 800

2004 9 900

2006 10 950

2007 11 960

2008 12 970

;

RUN;

**DATA** allsales;

merge salesone salestwo;

by year;

RUN;

**PROC** **PRINT** data=allsales NOOBS;

title 'The allsales data set';

**RUN**;

\*merge data sets in reverse order;

**DATA** allsales2;

merge salestwo salesone;

by year;

RUN;

**PROC** **PRINT** data=allsales2 NOOBS;

title 'The allsales2 data set';

**RUN**;

\*16.8 using the RENAME option for v\_date;

**DATA** demogtwo;

input subj gender $ age v\_date mmddyy8.;

format v\_date mmddyy8.;

DATALINES;

1000 M 42 03/10/96

1001 M 20 02/19/96

1002 F 53 02/01/96

1003 F 40 12/31/95

1004 M 29 01/10/97

;

RUN;

**DATA** statustwo;

input subj disease $ test $ v\_date mmddyy8.;

format v\_date mmddyy8.;

DATALINES;

1000 Y Y 03/17/96

1001 N Y 03/01/96

1002 N N 02/18/96

1003 Y Y 01/15/96

1004 N N 02/01/97

;

RUN;

**DATA** patientstwo;

merge demogtwo (rename = (v\_date = demogdate))

statustwo (rename = (v\_date = statusdate));

by subj;

RUN;

**PROC** **PRINT** data=patientstwo NOOBS;

title 'The patientstwo data set';

**RUN**;

\*creates the data sets;

**DATA** patients;

input id v\_date : mmddyy8.;

format v\_date mmddyy8.;

DATALINES;

110011 01/01/06

110012 01/02/06

110013 01/04/06

;

RUN;

**DATA** allvoids;

input id v\_date : mmddyy8. void\_no volume;

format v\_date mmddyy8.;

DATALINES;

110011 01/01/06 1 250

110011 01/01/06 2 300

110011 01/01/06 3 302

110011 01/01/06 4 231

110012 01/02/06 1 305

110012 01/02/06 2 225

110012 01/02/06 3 400

110013 01/04/06 1 300

110013 01/04/06 2 333

110013 01/04/06 3 401

110013 01/04/06 4 404

110014 01/06/06 1 398

110014 01/06/06 2 413

;

RUN;

\*match merge the data sets;

**DATA** analysis;

merge patients allvoids;

by id v\_date;

RUN;

**PROC** **PRINT** data=analysis NOOBS;

title 'The analysis data set';

**RUN**;

\*match merge the data sets and ensure we do not include patients whose ID numbers

are not in both data sets by using the IN= option and IF statement;

**DATA** analysis;

merge patients (in = inpatients)

allvoids (in = inallvoids);

by id v\_date;

if inpatients and inallvoids;

RUN;

**PROC** **PRINT** data=analysis NOOBS;

title 'The analysis data set';

**RUN**;

\*print a subset of observations;

LIBNAME icdb 'C:\Users\Owner\Documents\Whit Masters\STAT 481\data';

**PROC** **PRINT** data = icdb.back (OBS=**5**) NOOBS;

title 'The back data set';

**RUN**;

**PROC** **PRINT** data = icdb.purg (OBS=**5**) NOOBS;

title 'The purg data set';

where v\_type = **0**;

**RUN**;

**PROC** **PRINT** data = icdb.fhx (OBS=**5**) NOOBS;

title 'The fhx data set';

**RUN**;

LIBNAME icdb 'C:\Simon\Stat481WC\sp09\04combineII\sasndata';

\*merge three previous data sets;

**DATA** icdbdata (drop = v\_type);

merge icdb.back (keep = subj b\_date sex)

icdb.purg (where = (v\_type = **0**)

keep = subj v\_type purg\_1 purg\_2)

icdb.fhx (keep = subj fhx\_1 fhx\_2);

by subj;

RUN;

**PROC** **PRINT** data = icdbdata (OBS = **10**) NOOBS;

title 'The icdbdata data set';

**RUN**;

Leson 17

\* use output statement to tell SAS to write observations to data sets based on certain conditions;

OPTIONS PS=**58** LS=**80** NODATE NONUMBER;

LIBNAME stat481 'C:\Users\Owner\Documents\Whit Masters\STAT 481\data';

**DATA** s210006 s310032 s410010;

set stat481.icdblog;

if (subj = **210006**) then output s210006;

else if (subj = **310032**) then output s310032;

else if (subj = **410010**) then output s410010;

RUN;

**PROC** **PRINT** data = s210006 NOOBS;

title 'The s210006 data set';

**RUN**;

**PROC** **PRINT** data = s310032 NOOBS;

title 'The s310032 data set';

**RUN**;

**PROC** **PRINT** NOOBS;

title 'The s410010 data set';

**RUN**;

\*17.2 example of what happens if you fail to direct all obs to output, note

output statment required to program all output for a step;

**DATA** subj210006 subj310032;

set stat481.icdblog;

if (subj = **210006**) then output subj210006;

RUN;

**PROC** **PRINT** data = subj210006 NOOBS;

title 'The subj210006 data set';

**RUN**;

**PROC** **PRINT** data = subj310032 NOOBS;

title 'The subj310032 data set';

**RUN**;

\*17.3 to write new variables with assignment statements, ensure the stmts

are placed before the output stmts, example below does not do this

result: missing values for current and days\_vis;

**DATA** subj210006 subj310032 subj410010;

set stat481.icdblog;

if (subj = **210006**) then output subj210006;

else if (subj = **310032**) then output subj310032;

else if (subj = **410010**) then output subj410010;

current = today();

days\_vis = current - v\_date;

format current mmddyy8.;

RUN;

**PROC** **PRINT** data = subj310032 NOOBS;

title 'The subj310032 data set';

**RUN**;

\*corrected code for 17.3 to include new variables in output;

**DATA** subj210006 subj310032 subj410010;

set stat481.icdblog;

current = today();

days\_vis = current - v\_date;

format current mmddyy8.;

if (subj = **210006**) then output subj210006;

else if (subj = **310032**) then output subj310032;

else if (subj = **410010**) then output subj410010;

RUN;

**PROC** **PRINT** data = subj310032 NOOBS;

title 'The subj310032 data set';

**RUN**;

\*17.4 illustrates how you can create different data sets with same obs.

datasets do NOT have to be mutually exclusive;

**DATA** symptoms visitsix;

set stat481.icdblog;

if form = 'sympts' then output symptoms;

if v\_type = **6** then output visitsix;

RUN;

**PROC** **PRINT** data = symptoms NOOBS;

title 'The symptoms data set';

**RUN**;

**PROC** **PRINT** data = visitsix NOOBS;

title 'The visitsix data set';

**RUN**;

\*17.5 create grades data set for remaider of lesson;

**DATA** grades;

input idno **1**-**2** l\_name $ **5**-**9** gtype $ **12**-**13** grade **15**-**17**;

cards;

10 Smith E1 78

10 Smith E2 82

10 Smith E3 86

10 Smith E4 69

10 Smith P1 97

10 Smith F1 160

11 Simon E1 88

11 Simon E2 72

11 Simon E3 86

11 Simon E4 99

11 Simon P1 100

11 Simon F1 170

12 Jones E1 98

12 Jones E2 92

12 Jones E3 92

12 Jones E4 99

12 Jones P1 99

12 Jones F1 185

;

RUN;

**PROC** **PRINT** data = grades NOOBS;

title 'The grades data set';

**RUN**;

\*17.6 use the retain statement to compare values across observations to

determine each student's lowest grade;

**DATA** exams;

set grades (where = (gtype in ('E1', 'E2', 'E3', 'E4')));

RUN;

**DATA** lowest (rename = (lowtype = gtype));

set exams;

by idno;

retain lowgrade lowtype;

if first.idno then lowgrade = grade;

lowgrade = min(lowgrade, grade);

if grade = lowgrade then lowtype = gtype;

if last.idno then output;

drop gtype;

RUN;

**PROC** **PRINT** data=lowest;

title 'Output Dataset: LOWEST';

**RUN**;

\*17.7 illustrates the automatic retention of two variables: sum variable and

variable created by IN= option;

**PROC** **SORT** data=grades;

by idno gtype;

**RUN**;

**DATA** final;

merge grades lowest (in=lowest);

by idno gtype;

if lowest then delete;

if first.idno then total = **0**;

total + grade;

if last.idno then fnl = (total/**600**)\***100**;

format fnl **5.1**;

drop lowgrade gtype;

RUN;

**PROC** **PRINT** data=final;

title 'Output Dataset: FINAL GRADES';

**RUN**;

\*17.8 failed attempt at transposing data set to one obs per student

missing a retain statement;

**DATA** grades2;

set grades;

by idno;

if gtype = 'E1' then E1 = grade;

else if gtype = 'E2' then E2 = grade;

else if gtype = 'E3' then E3 = grade;

else if gtype = 'E4' then E4 = grade;

else if gtype = 'P1' then P1 = grade;

else if gtype = 'F1' then F1 = grade;

if last.idno then output;

drop gtype grade;

RUN;

**PROC** **PRINT** data=grades2;

title 'Output Dataset: FAULTY TRANSPOSED GRADES';

**RUN**;

\*17.8A correcting program with retain statement;

**DATA** grades3;

set grades;

by idno;

if gtype = 'E1' then E1 = grade;

else if gtype = 'E2' then E2 = grade;

else if gtype = 'E3' then E3 = grade;

else if gtype = 'E4' then E4 = grade;

else if gtype = 'P1' then P1 = grade;

else if gtype = 'F1' then F1 = grade;

if last.idno then output;

retain E1 E2 E3 E4 P1 F1;

drop gtype grade;

RUN;

**PROC** **PRINT** data=grades3;

title 'Output Dataset: TRANSPOSED GRADES';

**RUN**;

\*calculates final grades using the fat data set created in 17.8A

note generates same result as seen in 17.7;

**DATA** grades4;

set grades3;

if E1 = min(E1, E2, E3, E4) then E1 = **.**;

else if E2 = min(E1, E2, E3, E4) then E2 = **.**;

else if E3 = min(E1, E2, E3, E4) then E3 = **.**;

else if E4 = min(E1, E2, E3, E4) then E4 = **.**;

fnl = sum(E1, E2, E3, E4, P1, F1)/**6**;

format fnl **5.1**;

RUN;

**PROC** **PRINT** data=grades4;

title 'Output Dataset: FINAL GRADES calculated from TRANSPOSED GRADES';

**RUN**;

\*18.1 determine 4 x 3 through repeated addition;

OPTIONS PS = **58** LS = **78** NODATE NONUMBER;

**DATA** multiply;

answer = **0**;

do i = **1** to **4**;

answer + **3**;

end;

RUN;

**PROC** **PRINT** NOOBS;

title 'Four Times Three Equals...';

**RUN**;

\*determine multiples of 5 up to 100 using iterative DO loop;

**DATA** multiply (drop = i);

multiple = **0**;

do i = **1** to **20**;

multiple + **5**;

output;

end;

RUN;

**PROC** **PRINT** NOOBS;

title 'Multiples of 5 up to 100';

**RUN**;

\*18.3 iterative DO loop to count backwards by 1;

**DATA** backwardsbyone;

do i = **20** to **1** by -**1**;

output;

end;

RUN;

**PROC** **PRINT** data = backwardsbyone NOOBS;

title 'Counting Backwards by 1';

**RUN**;

\*18.4 generate a 4 x 5 factorial design by nesting DO loops;

**DATA** design;

DO i = **1** to **4**;

DO j = **1** to **5**;

output;

END;

END;

RUN;

**PROC** **PRINT** data = design;

TITLE '4 by 5 Factorial Design';

**RUN**;

\*18.5 use two do loops with BY options to generate a more meaningful factorial design;

**DATA** design;

DO i = **10** to **40** by **10**;

DO j = **3** to **15** BY **3**;

output;

END;

END;

RUN;

**PROC** **PRINT** data = design;

TITLE '4 by 5 Factorial Design';

**RUN**;

\*18.6 iteratively procssing data: example involves reading a data set and using a DO loop to compute a new variable

ex. determining howmuch each CD will earn at maturity;

**DATA** cdinvest (drop = i);

input Type $ **1**-**7** AnnualRate Months;

Investment = **5000**;

do i = **1** to Months;

Investment + (AnnualRate/**12**)\*Investment;

end;

format Investment dollar8.2;

DATALINES;

03Month 0.01980 3

06Month 0.02230 6

09Month 0.02230 9

12Month 0.02470 12

18Month 0.02470 18

24Month 0.02570 24

36Month 0.02720 36

48Month 0.02960 48

60Month 0.03445 60

;

RUN;

**PROC** **PRINT** data = cdinvest NOOBS;

title 'Comparison of Different CD Rates';

**RUN**;

\*18.7 conditional DO loops: SAS executes the DO loop until the expression is true

ex: determine how many years it will take to accumulate 50K if you deposit 1.2K

each year into acct with 5% interest;

**DATA** investment;

DO UNTIL (value >= **50000**);

value + **1200**;

value + value \* **0.05**;

year + **1**;

OUTPUT;

END;

RUN;

**PROC** **PRINT** data = investment NOOBS;

title 'Years until at least $50,000';

**RUN**;

\*18.8 conditional DO loops: DO WHILE tells SAS to execute the DO loop while the expression

specified is true

ex: attempt to answer the same question in 18.7, result: no output, 0 observations;

**DATA** investtwo;

DO WHILE (value >= **50000**);

value + **1200**;

value + value \* **0.05**;

year + **1**;

OUTPUT;

END;

RUN;

**PROC** **PRINT** data = investtwo NOOBS;

title 'Years until at least $50,000';

**RUN**;

\*18.9 change the inequality in 18.8 to allow the program to generate an answer;

**DATA** investthree;

value = **0**;

DO WHILE (value < **50000**);

value + **1200**;

value + value \* **0.05**;

year + **1**;

OUTPUT;

END;

RUN;

**PROC** **PRINT** data = investthree NOOBS;

title 'Years until at least $50,000';

**RUN**;

\*use conditional iterative DO loop to accumulate your investment until we reach 15 years

or until the value of the investment exceeds 50K

result: 15 years comes first;

**DATA** investfour (drop = i);

DO i = **1** to **15** UNTIL (value >= **50000**);

value + **1200**;

value + value \* **0.05**;

year + **1**;

OUTPUT;

END;

RUN;

**PROC** **PRINT** data = investfour NOOBS;

title 'Value of Investment';

**RUN**;

\*18.11 conditional iterative DO loop

ex: determine how many years it will take to accumulate 50K if you deposit 3.6K

each year into acct with 5% interest

note: accumulate your investment until we reach 15 years

or until the value of the investment exceeds 50K

result: investment exceeds 50K;

**DATA** investfive (drop = i);

DO i = **1** to **15** UNTIL (value >= **50000**);

value + **3600**;

value + value \* **0.05**;

year + **1**;

OUTPUT;

END;

RUN;

**PROC** **PRINT** data = investfive NOOBS;

title 'Value of Investment';

**RUN**;

\*18.12 using a iterative DO loop to select samples from larger data set;

OPTIONS LS = **72** PS = **34** NODATE NONUMBER;

LIBNAME stat481 'C:\Users\Owner\Documents\Whit Masters\STAT 481\data';

**DATA** sample;

DO i = **100** to **8600** by **100**;

set stat481.log11 point = i;

output;

END;

stop;

RUN;

**PROC** **PRINT** data = sample NOOBS;

title 'Subset of Logged Observations for Hospital 11';

**RUN**;

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

\*19.1 create a dataset;

**DATA** avgcelsius;

input City $ **1**-**18** jan feb mar apr may jun

jul aug sep oct nov dec;

DATALINES;

State College, PA -2 -2 2 8 14 19 21 20 16 10 4 -1

Miami, FL 20 20 22 23 26 27 28 28 27 26 23 20

St. Louis, MO -1 1 6 13 18 23 26 25 21 15 7 1

New Orleans, LA 11 13 16 20 23 27 27 27 26 21 16 12

Madison, WI -8 -5 0 7 14 19 22 20 16 10 2 -5

Houston, TX 10 12 16 20 23 27 28 28 26 21 16 12

Phoenix, AZ 12 14 16 21 26 31 33 32 30 23 16 12

Seattle, WA 5 6 7 10 13 16 18 18 16 12 8 6

San Francisco, CA 10 12 12 13 14 15 15 16 17 16 14 11

San Diego, CA 13 14 15 16 17 19 21 22 21 19 16 14

;

RUN;

**PROC** **PRINT** data = avgcelsius;

title 'Average Monthly Temperatures in Celsius';

id City;

var jan feb mar apr may jun

jul aug sep oct nov dec;

**RUN**;

\*19.2 change celcius to fahrenheit;

**DATA** avgfahrenheit;

set avgcelsius;

janf = **1.8**\*jan + **32**;

febf = **1.8**\*feb + **32**;

marf = **1.8**\*mar + **32**;

aprf = **1.8**\*apr + **32**;

mayf = **1.8**\*may + **32**;

junf = **1.8**\*jun + **32**;

julf = **1.8**\*jul + **32**;

augf = **1.8**\*aug + **32**;

sepf = **1.8**\*sep + **32**;

octf = **1.8**\*oct + **32**;

novf = **1.8**\*nov + **32**;

decf = **1.8**\*dec + **32**;

drop jan feb mar apr may jun

jul aug sep oct nov dec;

RUN;

**PROC** **PRINT** data = avgfahrenheit;

title 'Average Monthly Temperatures in Fahrenheit';

id City;

var janf febf marf aprf mayf junf

julf augf sepf octf novf decf;

**RUN**;

\*19.2 uses a one-dimensional array to convert temperatures;

**DATA** avgfahrenheit;

set avgcelsius;

array fahr(**12**) jan feb mar apr may jun

jul aug sep oct nov dec;

do i = **1** to **12**;

fahr(i) = **1.8**\*fahr(i) + **32**;

end;

RUN;

**PROC** **PRINT** data = avgfahrenheit;

title 'Average Monthly Temperatures in Fahrenheit';

id City;

var jan feb mar apr may jun

jul aug sep oct nov dec;

**RUN**;

\*19.3 using (\*) in defining an array;

**DATA** avgfahrenheittwo;

set avgcelsius;

array fahr(\*) jan feb mar apr may jun

jul aug sep oct nov dec;

do i = **1** to **12**;

fahr(i) = **1.8**\*fahr(i) + **32**;

end;

RUN;

**PROC** **PRINT** data = avgfahrenheittwo;

title 'Average Monthly Temperatures in Fahrenheit';

id City;

var jan feb mar apr may jun

jul aug sep oct nov dec;

**RUN**;

\*19.4 use m1-m12 as shortcut in defining an array;

**DATA** avgtempsF;

input City $ **1**-**18** m1 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12;

array fahr(\*) m1-m12;

do i = **1** to **12**;

fahr(i) = **1.8**\*fahr(i) + **32**;

end;

DATALINES;

State College, PA -2 -2 2 8 14 19 21 20 16 10 4 -1

Miami, FL 20 20 22 23 26 27 28 28 27 26 23 20

St. Louis, MO -1 1 6 13 18 23 26 25 21 15 7 1

New Orleans, LA 11 13 16 20 23 27 27 27 26 21 16 12

Madison, WI -8 -5 0 7 14 19 22 20 16 10 2 -5

Houston, TX 10 12 16 20 23 27 28 28 26 21 16 12

Phoenix, AZ 12 14 16 21 26 31 33 32 30 23 16 12

Seattle, WA 5 6 7 10 13 16 18 18 16 12 8 6

San Francisco, CA 10 12 12 13 14 15 15 16 17 16 14 11

San Diego, CA 13 14 15 16 17 19 21 22 21 19 16 14

;

RUN;

**PROC** **PRINT** data = avgtempsF;

title 'Average Monthly Temperatures in Fahrenheit';

id City;

var m1-m12;

**RUN**;

\*19.5 use of a numeric all special list;

**DATA** avgtempsFtwo;

input City $ **1**-**18** jan feb mar apr may jun

jul aug sep oct nov dec;

array fahr(\*) \_NUMERIC\_;

do i = **1** to **12**;

fahr(i) = **1.8**\*fahr(i) + **32**;

end;

DATALINES;

State College, PA -2 -2 2 8 14 19 21 20 16 10 4 -1

Miami, FL 20 20 22 23 26 27 28 28 27 26 23 20

St. Louis, MO -1 1 6 13 18 23 26 25 21 15 7 1

New Orleans, LA 11 13 16 20 23 27 27 27 26 21 16 12

Madison, WI -8 -5 0 7 14 19 22 20 16 10 2 -5

Houston, TX 10 12 16 20 23 27 28 28 26 21 16 12

Phoenix, AZ 12 14 16 21 26 31 33 32 30 23 16 12

Seattle, WA 5 6 7 10 13 16 18 18 16 12 8 6

San Francisco, CA 10 12 12 13 14 15 15 16 17 16 14 11

San Diego, CA 13 14 15 16 17 19 21 22 21 19 16 14

;

RUN;

**PROC** **PRINT** data = avgtempsFtwo;

title 'Average Monthly Temperatures in Fahrenheit';

id City;

var jan--dec;

**RUN**;

\*19.6 identical to 19.2;

**DATA** avgfahrenheit;

set avgcelsius;

array fahr(**12**) jan feb mar apr may jun

jul aug sep oct nov dec;

do i = **1** to **12**;

fahr(i) = **1.8**\*fahr(i) + **32**;

end;

RUN;

**PROC** **PRINT** data = avgfahrenheit;

title 'Average Monthly Temperatures in Fahrenheit';

id City;

var jan feb mar apr may jun

jul aug sep oct nov dec;

**RUN**;

\*19.7 creating new variables in array statement;

**DATA** avgtemps;

set avgcelsius;

array celsius(**12**) jan feb mar apr may jun

jul aug sep oct nov dec;

array fahr(**12**) janf febf marf aprf mayf junf

julf augf sepf octf novf decf;

do i = **1** to **12**;

fahr(i) = **1.8**\*celsius(i) + **32**;

end;

RUN;

**PROC** **PRINT** data = avgtemps;

title 'Average Monthly Temperatures';

id City;

var jan janf feb febf mar marf;

var apr aprf may mayf jun junf;

var jul julf aug augf sep sepf;

var oct octf nov novf dec decf;

**RUN**;

\*19.8 let SAS do the naming of hte variables in fahr array;

**DATA** avgtempsinF;

set avgcelsius;

array celsius(**12**) jan feb mar apr may jun

jul aug sep oct nov dec;

array fahr(**12**);

do i = **1** to **12**;

fahr(i) = **1.8**\*celsius(i) + **32**;

end;

RUN;

**PROC** **PRINT** data = avgtempsinF;

title 'Average Monthly Temperatures in Fahrenheit';

id City;

var fahr1-fahr12;

**RUN**;

\*19.9 checking for errors: checks variable 1,2, or 3 were recoreded as expected;

**DATA** qul errors;

input subj qul3a qul3b qul3c qul3d qul3e

qul3f qul3g qul3h qul3i qul3j;

flag = **0**;

if qul3a not in (**1**, **2**, **3**) then flag = **1**;

if qul3b not in (**1**, **2**, **3**) then flag = **1**;

if qul3c not in (**1**, **2**, **3**) then flag = **1**;

if qul3d not in (**1**, **2**, **3**) then flag = **1**;

if qul3e not in (**1**, **2**, **3**) then flag = **1**;

if qul3f not in (**1**, **2**, **3**) then flag = **1**;

if qul3g not in (**1**, **2**, **3**) then flag = **1**;

if qul3h not in (**1**, **2**, **3**) then flag = **1**;

if qul3i not in (**1**, **2**, **3**) then flag = **1**;

if qul3j not in (**1**, **2**, **3**) then flag = **1**;

if flag = **1** then output errors;

else output qul;

drop flag;

DATALINES;

110011 1 2 3 3 3 3 2 1 1 3

210012 2 3 4 1 2 2 3 3 1 1

211011 1 2 3 2 1 2 3 2 1 3

310017 1 2 3 3 3 3 3 2 2 1

411020 4 3 3 3 3 2 2 2 2 2

510001 1 1 1 1 1 1 2 1 2 2

;

RUN;

**PROC** **PRINT** data = qul;

TITLE 'Observations in Qul data set with no errors';

**RUN**;

**PROC** **PRINT** data = errors;

TITLE 'Observations in Qul data set with errors';

**RUN**;

\*19.10 error checking with two arrays introducing three new variables, similar to 19.9;

**DATA** qul errors;

input subj qul3a qul3b qul3c qul3d qul3e

qul3f qul3g qul3h qul3i qul3j;

array bounds (**3**) error1 - error3 (**1** **2** **3**);

array quldata (**10**) qul3a -- qul3j;

flag = **0**;

do i = **1** to **10**;

if quldata(i) ne bounds(**1**) and

quldata(i) ne bounds(**2**) and

quldata(i) ne bounds(**3**)

then flag = **1**;

end;

if flag = **1** then output errors;

else output qul;

drop i flag;

DATALINES;

110011 1 2 3 3 3 3 2 1 1 3

210012 2 3 4 1 2 2 3 3 1 1

211011 1 2 3 2 1 2 3 2 1 3

310017 1 2 3 3 3 3 3 2 2 1

411020 4 3 3 3 3 2 2 2 2 2

510001 1 1 1 1 1 1 2 1 2 2

;

RUN;

**PROC** **PRINT** data = qul;

TITLE 'Observations in Qul data set with no errors';

**RUN**;

**PROC** **PRINT** data = errors;

TITLE 'Observations in Qul data set with errors';

**RUN**;

\*19.11 uses temporary array elements to solve 19.10;

**DATA** qul errors;

input subj qul3a qul3b qul3c qul3d qul3e

qul3f qul3g qul3h qul3i qul3j;

array bounds (**3**) \_TEMPORARY\_ (**1** **2** **3**);

array quldata (**10**) qul3a -- qul3j;

flag = **0**;

do i = **1** to **10**;

if quldata(i) ne bounds(**1**) and

quldata(i) ne bounds(**2**) and

quldata(i) ne bounds(**3**)

then flag = **1**;

end;

if flag = **1** then output errors;

else output qul;

drop i flag;

DATALINES;

110011 1 2 3 3 3 3 2 1 1 3

210012 2 3 4 1 2 2 3 3 1 1

211011 1 2 3 2 1 2 3 2 1 3

310017 1 2 3 3 3 3 3 2 2 1

411020 4 3 3 3 3 2 2 2 2 2

510001 1 1 1 1 1 1 2 1 2 2

;

RUN;

**PROC** **PRINT** data = qul;

TITLE 'Observations in Qul data set with no errors';

**RUN**;

**PROC** **PRINT** data = errors;

TITLE 'Observations in Qul data set with errors';

**RUN**;

\*19.12 use DIM function to recode 2 to 1 and 1 to zero;

**DATA** survey (DROP = i);

INPUT subj q1 q2 q3 q4 q5 q6;

ARRAY qxs(**4**) q3-q6;

DO i = **1** to dim(qxs);

qxs(i) = qxs(i) - **1**;

END;

DATALINES;

1001 1 2 1 2 1 1

1002 2 1 2 2 2 1

1003 2 2 2 1 . 2

1004 1 . 1 1 1 2

1005 2 1 2 2 2 1

;

RUN;

**PROC** **PRINT** data = survey;

TITLE 'The survey data using dim function';

**RUN**;

\*19.13 redefine the lower and upper bounds in 19.12;

**DATA** survey2 (DROP = i);

INPUT subj q1 q2 q3 q4 q5 q6;

ARRAY qxs(**3**:**6**) q3-q6;

DO i = **3** to **6**;

qxs(i) = qxs(i) - **1**;

END;

DATALINES;

1001 1 2 1 2 1 1

1002 2 1 2 2 2 1

1003 2 2 2 1 . 2

1004 1 . 1 1 1 2

1005 2 1 2 2 2 1

;

RUN;

**PROC** **PRINT** data = survey2;

TITLE 'The survey data using bounded arrays';

**RUN**;

\*19.14 using a one-dimensional array to recode four survey variables

- \* used to tell SAS to determine dimension of qxs array

- lbound and hbound functions used to tell sas to determine bounds dynamically;

**DATA** survey2 (DROP = i);

INPUT subj q1 q2 q3 q4 q5 q6;

ARRAY qxs(**3**:**6**) q3-q6;

DO i = **3** to **6**;

qxs(i) = qxs(i) - **1**;

END;

DATALINES;

1001 1 2 1 2 1 1

1002 2 1 2 2 2 1

1003 2 2 2 1 . 2

1004 1 . 1 1 1 2

1005 2 1 2 2 2 1

;

RUN;

**PROC** **PRINT** data = survey2;

TITLE 'The survey data using bounded arrays';

**RUN**;

\*19.15 generate tallgrades dataset;

**DATA** tallgrades;

input idno **1**-**2** l\_name $ **5**-**9** gtype $ **12**-**13** grade **15**-**17**;

cards;

10 Smith E1 78

10 Smith E2 82

10 Smith E3 86

10 Smith E4 69

10 Smith P1 97

10 Smith F1 160

11 Simon E1 88

11 Simon E2 72

11 Simon E3 86

11 Simon E4 99

11 Simon P1 100

11 Simon F1 170

12 Jones E1 98

12 Jones E2 92

12 Jones E3 92

12 Jones E4 99

12 Jones P1 99

12 Jones F1 185

;

RUN;

**PROC** **PRINT** data = tallgrades NOOBS;

TITLE 'The tall grades data set';

**RUN**;

\*19.16 using retain and output statements with by-processing to transpose data;

**DATA** fatgrades;

set tallgrades;

by idno;

if gtype = 'E1' then E1 = grade;

else if gtype = 'E2' then E2 = grade;

else if gtype = 'E3' then E3 = grade;

else if gtype = 'E4' then E4 = grade;

else if gtype = 'P1' then P1 = grade;

else if gtype = 'F1' then F1 = grade;

if last.idno then output;

retain E1 E2 E3 E4 P1 F1;

drop gtype grade;

RUN;

**PROC** **PRINT** data=fatgrades;

title 'The fat grades data set';

**RUN**;

\*19.17 use an array to transpose data;

**DATA** fatgrades;

set tallgrades;

by idno;

array allgrades (**6**) G1 - G6;

if first.idno then i = **1**;

allgrades(i) = grade;

if last.idno then output;

i + **1**;

retain G1 - G6;

drop i gtype grade;

RUN;

**PROC** **PRINT** data=fatgrades;

title 'The fat grades data set';

**RUN**;

\*19.18 check a subset of data for missing data using two one-

dimensional arrays;

**DATA** fhx;

input subj v\_date mmddyy8. fhx1-fhx14;

array edit(**14**) fhx1-fhx14;

array status(**14**) stat1-stat14;

do i = **1** to **14**;

status(i) = **0**;

if edit(i) = **.** then status(i) = **1**;

end;

DATALINES;

220004 07/27/93 0 0 0 . 8 0 0 1 1 1 . 1 0 1

410020 11/11/93 0 0 0 . 0 0 0 0 0 0 . 0 0 0

520013 10/29/93 0 0 0 . 0 0 0 0 0 0 . 0 0 1

520068 08/10/95 0 0 0 0 0 1 1 0 0 1 1 0 1 0

520076 08/25/95 0 0 0 0 1 8 0 0 0 1 1 0 0 1

;

RUN;

**PROC** **PRINT** data = fhx;

var fhx1-fhx14;

TITLE 'The FHX data itself';

**RUN**;

**PROC** **PRINT** data = fhx;

var stat1-stat14;

TITLE 'The presence of missing values in FHX data';

**RUN**;

\*19.19 use one two-dimensional array for 19.18;

**DATA** fhx2;

input subj v\_date mmddyy8. fhx1-fhx14;

array edit(**2**,**14**) fhx1-fhx14 stat1-stat14;

do i = **1** to **14**;

edit(**2**,i) = **0**;

if edit(**1**,i) = **.** then edit(**2**,i) = **1**;

end;

DATALINES;

220004 07/27/93 0 0 0 . 8 0 0 1 1 1 . 1 0 1

410020 11/11/93 0 0 0 . 0 0 0 0 0 0 . 0 0 0

520013 10/29/93 0 0 0 . 0 0 0 0 0 0 . 0 0 1

520068 08/10/95 0 0 0 0 0 1 1 0 0 1 1 0 1 0

520076 08/25/95 0 0 0 0 1 8 0 0 0 1 1 0 0 1

;

RUN;

**PROC** **PRINT** data = fhx2;

var fhx1-fhx14;

TITLE 'The FHX2 data itself';

**RUN**;

**PROC** **PRINT** data = fhx2;

var stat1-stat14;

TITLE 'The presence of missing values in FHX2 data';

**RUN**;

\*20.1;

**DATA** temp;

input subj **1**-**4** name $ **6**-**23** gender **25** height **27**-**28** weight **30**-**32**;

CARDS;

1024 Alice Smith 1 65 125

1167 Maryann White 1 68 140

1168 Thomas Jones 2 68 190

1201 Benedictine Arnold 2 68 190

1302 Felicia Ho 1 63 115

;

RUN;

**PROC** **PRINT** data=temp;

title 'Output dataset: TEMP';

**RUN**;

\*20.2 formatted input;

**DATA** temp;

input @**1** subj **4.**

@**6** f\_name $11.

@**18** l\_name $6.

+**3** height **2.**

+**5** wt\_date mmddyy8.

+**1** calorie comma5.;

format wt\_date mmddyy8. calorie comma5.;

DATALINES;

1024 Alice Smith 1 65 125 12/1/95 2,036

1167 Maryann White 1 68 140 12/01/95 1,800

1168 Thomas Jones 2 190 12/2/95 2,302

1201 Benedictine Arnold 2 68 190 11/30/95 2,432

1302 Felicia Ho 1 63 115 1/1/96 1,972

;

RUN;

**PROC** **PRINT** data = temp;

title 'Output dataset: TEMP';

id subj;

**RUN**;

\*20.3 list input;

**DATA** temp;

input subj name $ gender height weight;

CARDS;

1024 Alice 1 65 125

1167 Maryann 1 68 140

1168 Thomas 2 68 190

1201 Benedictine . 68 190

1302 Felicia 1 63 115

;

RUN;

**PROC** **PRINT** data=temp NOOBS;

title 'Output dataset: TEMP';

**RUN**;

\*20.4 same as 20.2, print contents to see length of variables;

**DATA** temp;

input @**1** subj **4.**

@**6** f\_name $11.

@**18** l\_name $6.

+**3** height **2.**

+**5** wt\_date mmddyy8.

+**1** calorie comma5.;

format wt\_date mmddyy8. calorie comma5.;

DATALINES;

1024 Alice Smith 1 65 125 12/1/95 2,036

1167 Maryann White 1 68 140 12/01/95 1,800

1168 Thomas Jones 2 190 12/2/95 2,302

1201 Benedictine Arnold 2 68 190 11/30/95 2,432

1302 Felicia Ho 1 63 115 1/1/96 1,972

;

RUN;

**PROC** **CONTENTS** data = temp;

title 'Contents of TEMP Data Set';

**RUN**;

\*20.5 change 20.4 to all absolute pointer controls and overlap formats;

**DATA** temp;

input @**1** subj **4.**

@**6** f\_name $11.

@**18** l\_name $6.

@**30** height **8.**

@**34** wt\_date mmddyy8.

@**43** calorie comma5.;

format wt\_date mmddyy8. calorie comma5.;

DATALINES;

1024 Alice Smith 1 65 125 12/1/95 2,036

1167 Maryann White 1 68 140 12/01/95 1,800

1168 Thomas Jones 2 190 12/2/95 2,302

1201 Benedictine Arnold 2 68 190 11/30/95 2,432

1302 Felicia Ho 1 63 115 1/1/96 1,972

;

RUN;

**PROC** **PRINT** data = temp;

title 'The TEMP Data Set';

**RUN**;

\*20.6 try to read in a variable length data file using formatted input;

**DATA** temp;

infile 'C:\Users\Owner\Documents\Whit Masters\STAT 481\data\addresses.dat';

input @**1** subj **4.** name $ **6**-**23** street $ **27**-**45**;

RUN;

**PROC** **PRINT**;

title 'Temp data set';

**RUN**;

\*20.7 use the missover option to tell SAS to advance to next line and set missing variables

to missing;

**DATA** temp;

infile 'C:\Users\Owner\Documents\Whit Masters\STAT 481\data\addresses.dat' MISSOVER;

input @**1** subj **4.** name $ **6**-**23** street $ **27**-**45**;

RUN;

**PROC** **PRINT**;

title 'Temp data set';

**RUN**;

\*20.8 reading variable length datasets: use the PAD option in the infile statement to tell SAS to pad each record in the data file with blanks

so that allof the datalines have the same length;

**DATA** temp;

infile 'C:\Users\Owner\Documents\Whit Masters\STAT 481\data\addresses.dat' PAD;

input @**1** subj **4.** name $ **6**-**23** street $ **27**-**45**;

RUN;

**PROC** **PRINT**;

title 'Temp data set';

**RUN**;

\* 20.9 use the missover option to read in data when some values are missing at end

of record;

**DATA** reading;

infile DATALINES MISSOVER;

input Name $ Week1-Week5;

DATALINES;

Robin 3 2 5 1 6

Jack 4 4 4 3 4

Tim 3 0 0

Martin 1 0 1 1

Caroline 2 3 4 5 6

RUN;

**PROC** **PRINT** data = reading;

title 'Summer Reading Program';

var Name Week1-Week5;

**RUN**;

\*20.10 use the DSD option in conjunciton with DLM= option in the infile statement when missing values are in middle of

an observation;

**DATA** survey;

infile DATALINES DLM=',' DSD;

input Name $ (Q1-Q5) ($);

DATALINES;

Robert,,A,C,A,D

William,B,C,A,D,A

Linda,C,B,,A,C

Lisa,D,D,D,C,A

Katherine,A,B,C,D,A

RUN;

**PROC** **PRINT** data = survey;

title 'Survey Results';

var Name Q1-Q5;

**RUN**;

\*20.11 using the DSD and DLM options when value missing at beginning of observation;

**DATA** survey;

infile DATALINES DLM=' ' DSD;

input Name $ (Q1-Q5) ($);

DATALINES;

Robert A C A D

William B C A D A

Linda C B A C

D D D C A

Katherine A B C D A

RUN;

**PROC** **PRINT** data = survey;

title 'Survey Results';

var Name Q1-Q5;

**RUN**;

\* 20.12 use the length statment to define length of name variable;

**DATA** survey;

infile DATALINES DLM=' ' DSD;

length Name $ **9**;

input Name $ (Q1-Q5) ($);

DATALINES;

Robert A C A D

William B C A D A

Linda C B A C

D D D C A

Katherine A B C D A

RUN;

**PROC** **PRINT** data = survey;

title 'Survey Results';

var Name Q1-Q5;

**RUN**;

\*20.13 failing to use list input to read in data correctly, need to use modified list input;

**DATA** citypops;

infile DATALINES FIRSTOBS = **2**;

input city pop2000;

DATALINES;

City Yr2000Popn

New York 8,008,278

Los Angeles 3,694,820

Chicago 2,896,016

Houston 1,953,631

Philadelphia 1,517,550

Phoenix 1,321,045

San Antonio 1,144,646

San Diego 1,223,400

Dallas 1,188,580

San Jose 894,943

;

RUN;

**PROC** **PRINT** data = citypops;

title 'The citypops data set';

**RUN**;

\*20.14 use the ampersand modifier to read in embedded spaces;

**DATA** citypops;

infile DATALINES FIRSTOBS = **2**;

length city $ **12**;

input city & pop2000;

DATALINES;

City Yr2000Popn

New York 8008278

Los Angeles 3694820

Chicago 2896016

Houston 1953631

Philadelphia 1517550

Phoenix 1321045

San Antonio 1144646

San Diego 1223400

Dallas 1188580

San Jose 894943

;

RUN;

**PROC** **PRINT** data = citypops;

title 'The citypops data set';

format pop2000 comma10.;

**RUN**;

\*20.15 removing the length statment and using an informat in the input statement;

**DATA** citypops;

infile DATALINES FIRSTOBS = **2**;

input city & $12. pop2000;

DATALINES;

City Yr2000Popn

New York 8008278

Los Angeles 3694820

Chicago 2896016

Houston 1953631

Philadelphia 1517550

Phoenix 1321045

San Antonio 1144646

San Diego 1223400

Dallas 1188580

San Jose 894943

;

RUN;

**PROC** **PRINT** data = citypops;

title 'The citypops data set';

format pop2000 comma10.;

**RUN**;

\*20.16 use the colon delimiter to use list input to read nonstandard values and character values

longer than 8 characters w/o embedded blanks

- colon indicates the values are read until a space delimiter is encountered, then informat applied;

**DATA** citypops;

infile DATALINES FIRSTOBS = **2**;

input city & $12. pop2000 : comma.;

DATALINES;

City Yr2000Popn

New York 8,008,278

Los Angeles 3,694,820

Chicago 2,896,016

Houston 1,953,631

Philadelphia 1,517,550

Phoenix 1,321,045

San Antonio 1,144,646

San Diego 1,223,400

Dallas 1,188,580

San Jose 894,943

;

RUN;

**PROC** **PRINT** data = citypops;

title 'The citypops data set';

format pop2000 comma10.;

**RUN**;

\*20.17 example using list input, column input, and formatted input;

**DATA** nationalparks;

input ParkName $ **1**-**22** State $ Year @**40** Acreage comma9.;

DATALINES;

Yellowstone ID/MT/WY 1872 4,065,493

Everglades FL 1934 1,398,800

Yosemite CA 1864 760,917

Great Smoky Mountains NC/TN 1926 520,269

Wolf Trap Farm VA 1966 130

;

RUN;

**PROC** **PRINT** data = nationalparks;

format acreage comma9.;

**RUN**;

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

\*21.1;

**DATA** january;

input City & $13. State $;

input AvgHigh AvgLow Mean;

input Precip;

DATALINES;

State College PA

32 16 25

2.4

Miami FL

75 58 67

2.0

Honolulu HI

80 65 74

3.6

;

RUN;

**PROC** **PRINT** data = january;

title 'January Weather for Three U.S. Cities';

**RUN**;

\*21.2 use the forward slash (/) instead of additional input statements;

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

**DATA** january;

input City & $13. State $

/ AvgHigh AvgLow Mean

/ Precip;

DATALINES;

State College PA

32 16 25

2.4

Miami FL

75 58 67

2.0

Honolulu HI

80 65 74

3.6

;

RUN;

**PROC** **PRINT** data = january;

title 'January Weather for Three U.S. Cities';

**RUN**;

\*21.3 incorrect use of the forward slash pointer control

always make sure the data contains the same number of records for each observation;

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

**DATA** january;

input City & $13. State $

/ AvgHigh AvgLow Mean

/ Precip;

DATALINES;

State College PA

32 16 25

2.4

Miami FL

75 58 67

Honolulu HI

80 65 74

3.6

;

RUN;

**PROC** **PRINT** data = january;

title 'January Weather for Three U.S. Cities';

**RUN**;

\*21.4 using the # N pointer control to read data non-sequentially;

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

**DATA** january;

input #**2** AvgHigh AvgLow Mean

#**3** Precip

#**1** City & $13. State $;

DATALINES;

State College PA

32 16 25

2.4

Miami FL

75 58 67

2.0

Honolulu HI

80 65 74

3.6

;

RUN;

**PROC** **PRINT** data = january;

title 'January Weather for Three U.S. Cities';

**RUN**;

\*21.5 use / and #n to read data sequentially and non-sequentailly, respectively;

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

**DATA** january;

input #**3** Precip

#**1** City & $13. State $

/ AvgHigh AvgLow Mean;

DATALINES;

State College PA

32 16 25

2.4

Miami FL

75 58 67

2.0

Honolulu HI

80 65 74

3.6

;

RUN;

**PROC** **PRINT** data = january;

title 'January Weather for Three U.S. Cities';

**RUN**;

\*21.6 using the @@ trailing sign to read in the average temperatures;

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

**DATA** StateCollege;

input Month $ AvgHigh AvgLow @@;

DATALINES;

Jan 32 16 Feb 35 18 Mar 46 26

Apr 58 37 May 68 47 Jun 78 56

Jul 82 60 Aug 80 58 Sep 72 51

Oct 61 40 Nov 48 32 Dec 37 22

;

RUN;

**PROC** **PRINT** data = StateCollege;

title 'Average Temperatures for State College';

id Month;

**RUN**;

\*21.7 using a single trailing @ sign to read repeating fields;

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

**DATA** grades;

input ID @;

input score @;

output;

input score @;

output;

input score @;

output;

DATALINES;

111000234 79 82 100

922232573 87 89 95

252359873 65 72 73

205804679 92 95 99

;

RUN;

**PROC** **PRINT** data = grades NOOBS;

title 'Grades data set';

**RUN**;

\*21.8 use an iterative DO loop to program 21.7;

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

**DATA** grades;

input ID @;

DO exam = **1** to **3**;

input score @;

output;

END;

DATALINES;

111000234 79 82 100

922232573 87 89 95

252359873 65 72 73

205804679 92 95 99

;

RUN;

**PROC** **PRINT** data = grades NOOBS;

title 'Grades data set';

**RUN**;

\*21.9 reading a varying number of repeating fields in each record

using trailing @, infile statements missover option, and DO WHILE loop;

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

**DATA** dietdata;

infile DATALINES MISSOVER;

input id weight @;

weighin = **0**;

DO WHILE (weight ne **.**);

weighin+**1**;

output;

input weight @;

END;

DATALINES;

1001 179 172 169

1002 250 249

1003 190 196 195 164 158

1004 232 224 219 212 208

1005 211 208 204 202

;

RUN;

**PROC** **PRINT** data = dietdata NOOBS;

title 'The dietdata data set';

var id weighin weight;

**RUN**;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

HW # 21

File name: C:\Users\Owner\Documents\Whit Masters\STAT 481\Homework\21.sas

Written by: Whitney York

Written on: March 22, 2013

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The following OPTIONS statement sets the line size (LS) to 72 characters, the

page size to 58, removes the date and time and number from the output.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

OPTIONS PS = **58** LS = **72** NODATE NONUMBER;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Question #1

The purpose of the following code is to generate a report from the raw

data file, generating missing values where necessary.

The following DATA step genreates a temporary SAS dataset, pumpkins.

The INFILE statement tells SAS the location of the raw data.

The MISSOVER option tells SAS to assign a missing value for missing values

at the end of an observation.

The input style is modified list input since the data are not arranged in

columns and are separated by at least one space.

- The name variable uses the ampersand modifier to tell SAS the variable

contained embedded spaces. The informat defines the character variable

to 16 bytes.

- The type variable is defined as character variable.

- The EnterDate variable uses the colon modifier to tell SAS to read the

values until a blank is encountered.

The FORMAT statement defines the format of the EnterDate variable in the output.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**DATA** results;

infile 'C:\Users\Owner\Documents\Whit Masters\STAT 481\data\results.dat';

input team member $ score @@;

RUN;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The PROC PRINT procedure is used to generate output.

The TITLE statement tells SAS the title of the report.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**PROC** **PRINT** data = results;

title 'The results data set';

id team;

**RUN**;

\*modified list input;

**DATA** addresses;

infile 'C:\Users\Owner\Documents\Whit Masters\STAT 481\data\addresses.dat';

input #**4** ID /\*non-sequential pointer control \*/

#**1** Name & $18. /\*non-sequential pointer control \*/

/ Street & $19. /\*sequential pointer control\*/

/ Town & $11. State $ Zip; /\*sequential pointer control\*/

RUN;

**PROC** **PRINT** data = addresses NOOBS;

title 'The addresses data set';

**RUN**;

**DATA** results2;

infile 'C:\Users\Owner\Documents\Whit Masters\STAT 481\data\results2.dat';

input @**10** source @;

if source = **1** then input @**1** id name $ source score;

else input @**1** id score source name $;

run;

**PROC** **PRINT** data = results2 NOOBS;

title 'The results2 dataset';

**RUN**;

\*22.3 date calculations;

**DATA** diet;

input subj **1**-**4** l\_name $ **18**-**23** weight **30**-**32**

+**1** wt\_date mmddyy8. @**43** b\_date mmddyy8.;

nxt\_date = wt\_date + **14**;

age\_wt = (wt\_date - b\_date)/**365.25**;

avg\_date = MEAN(wt\_date, b\_date);

format wt\_date b\_date nxt\_date avg\_date date7.

age\_wt **4.1**;

DATALINES;

1024 Alice Smith 1 65 125 12/1/05 01/01/60

1167 Maryann White 1 68 140 12/01/05 01/01/59

1168 Thomas Jones 2 190 12/2/05 06/15/60

1201 Benedictine Arnold 2 68 190 11/30/05 12/31/60

1302 Felicia Ho 1 63 115 1/1/06 06/15/58

;

RUN;

**PROC** **PRINT** data=diet;

title 'The diet data set with three new variables';

**RUN**;

\*22.4 sort by date, asecending order;

**PROC** **SORT** data = diet out = sorteddiet;

by nxt\_date;

**RUN**;

**PROC** **PRINT** data = sorteddiet;

TITLE 'The diet data set sorted by nxt\_date';

**RUN**;

\*22.5 compare dates: use where= option to select indiv whose b\_date

is before Jan 1, 1960;

**DATA** diet (where = (b\_date < **'01jan1960'd**));

input subj **1**-**4** l\_name $ **18**-**23** weight **30**-**32**

+**1** wt\_date mmddyy8. @**43** b\_date mmddyy8.;

format wt\_date b\_date date9.;

DATALINES;

1024 Alice Smith 1 65 125 12/1/05 01/01/60

1167 Maryann White 1 68 140 12/01/05 01/01/59

1168 Thomas Jones 2 190 12/2/05 06/15/60

1201 Benedictine Arnold 2 68 190 11/30/05 12/31/60

1302 Felicia Ho 1 63 115 1/1/06 06/15/58

;

RUN;

**PROC** **PRINT** data=diet;

title 'Birthdays in the diet data set before 01/01/1960';

**RUN**;

\*22.6 using date functions to create date values in SAS;

**DATA** createdates;

current1= date();

current2 = today();

current3 = datejul(**2008095**);

mon = **4**; day = **4**; year = **2008**;

current4 = mdy(mon, day, year);

current5 = current4;

current6 = yyq(**2008**, **2**);

format current1 current2 current3 current5 current6 date9.;

RUN;

**PROC** **PRINT** data=createdates;

title 'The createdates data set';

var current1 current2 current3 current4 current5 current6;

**RUN**;

\*22.7 take apart date values to create new variables;

**DATA** takeapart;

input subj **1**-**4** l\_name $ **18**-**23** weight **30**-**32**

+**1** wt\_date mmddyy8. @**43** b\_date mmddyy8.;

wt\_mo = month(wt\_date);

wt\_day = day(wt\_date);

wt\_yr = year(wt\_date);

format wt\_date b\_date date9.;

DATALINES;

1024 Alice Smith 1 65 125 12/1/05 01/01/60

1167 Maryann White 1 68 140 12/01/05 01/01/59

1168 Thomas Jones 2 190 12/2/05 06/15/60

1201 Benedictine Arnold 2 68 190 11/30/05 12/31/60

1302 Felicia Ho 1 63 115 1/1/06 06/15/58

;

RUN;

**PROC** **PRINT** data=takeapart;

title 'The dissected weight dates';

var wt\_date wt\_mo wt\_day wt\_yr;

**RUN**;

\*22.8 manipulating dates;

**DATA** massaged;

input subj **1**-**4** l\_name $ **18**-**23** weight **30**-**32**

+**1** wt\_date mmddyy8. @**43** b\_date mmddyy8.;

wt\_jul1 = juldate(wt\_date);

wt\_jul2 = juldate7(wt\_date);

wt\_qtr = qtr(wt\_date);

wt\_day = weekday(wt\_date);

format wt\_date b\_date date9.;

DATALINES;

1024 Alice Smith 1 65 125 12/1/05 01/01/60

1167 Maryann White 1 68 140 12/01/05 01/01/59

1168 Thomas Jones 2 190 12/2/05 06/15/60

1201 Benedictine Arnold 2 68 190 11/30/05 12/31/60

1302 Felicia Ho 1 63 115 1/1/06 06/15/58

;

RUN;

**PROC** **PRINT** data = massaged;

title 'The massaged data set';

var wt\_date wt\_jul1 wt\_jul2 wt\_qtr wt\_day;

**RUN**;

\*22.9 use yrdiff and datdif fuctions to calcualte diff in year

and days, respectively;

**DATA** diet;

input subj **1**-**4** l\_name $ **18**-**23** weight **30**-**32**

+**1** wt\_date1 mmddyy8. @**43** wt\_date2 mmddyy8. @**52**

b\_date mmddyy8.;

age = yrdif(b\_date, wt\_date1, 'act/act');

days = datdif(wt\_date1, wt\_date2, 'act/act');

format wt\_date1 wt\_date2 b\_date date9. age **4.1**;

DATALINES;

1024 Alice Smith 1 65 125 12/1/05 03/04/06 01/01/60

1167 Maryann White 1 68 140 12/01/05 03/07/06 01/01/59

1168 Thomas Jones 2 190 12/2/05 3/30/06 06/15/60

1201 Benedictine Arnold 2 68 190 11/30/05 2/27/06 12/31/60

1302 Felicia Ho 1 63 115 1/1/06 4/1/06 06/15/58

;

RUN;

**PROC** **PRINT** data=diet;

TITLE "The calculation of subject's age";

var subj b\_date wt\_date1 age;

**RUN**;

**PROC** **PRINT** data=diet;

TITLE 'The calculation of days between weighings';

var subj wt\_date1 wt\_date2 days;

**RUN**;

\*22.10 use different functions to calculate 22.9;

**DATA** diet;

input subj **1**-**4** l\_name $ **18**-**23** weight **30**-**32**

+**1** wt\_date1 mmddyy8. @**43** wt\_date2 mmddyy8. @**52**

b\_date mmddyy8.;

age\_yrdif = yrdif(b\_date, wt\_date1, 'act/act');

age\_intck = intck('year', b\_date, wt\_date1);

days\_datdif = datdif(wt\_date1, wt\_date2, 'act/act');

days\_intck = intck('day', wt\_date1, wt\_date2);

format wt\_date1 wt\_date2 b\_date date9. age **4.1**;

DATALINES;

1024 Alice Smith 1 65 125 12/1/05 03/04/06 01/01/60

1167 Maryann White 1 68 140 12/01/05 03/07/06 01/01/59

1168 Thomas Jones 2 190 12/2/05 3/30/06 06/15/60

1201 Benedictine Arnold 2 68 190 11/30/05 2/27/06 12/31/60

1302 Felicia Ho 1 63 115 1/1/06 4/1/06 06/15/58

;

RUN;

**PROC** **PRINT** data=diet;

TITLE "The calculation of subject's age";

var subj b\_date wt\_date1 age\_yrdif age\_intck;

**RUN**;

**PROC** **PRINT** data=diet;

TITLE 'The calculation of days between weighings';

var subj wt\_date1 wt\_date2 days\_datdif days\_intck;

**RUN**;

\*22.11 intck() functions;

**DATA** timeintervals1;

days = intck('day', **'31dec2006'd**,**'01jan2007'd**);

weeks = intck('week', **'31dec2006'd**,**'01jan2007'd**);

months = intck('month', **'31dec2006'd**,**'01jan2007'd**);

years = intck('year', **'31dec2006'd**,**'01jan2007'd**);

years2 = intck('year', **'01jan2007'd**, **'31dec2007'd**);

years3 = intck('year', **'01jan2007'd**, **'01jan2008'd**);

RUN;

**PROC** **PRINT** data = timeintervals1;

TITLE 'Time intervals as calculated by intck function';

**RUN**;

\*22.12 further explore intck() function;

**DATA** timeintervals2;

days = intck('day', **'15mar2007'd**,**'15mar2008'd**);

weeks = intck('week', **'15mar2007'd**,**'15mar2008'd**);

weekdays = intck('weekday', **'15mar2007'd**,**'15mar2008'd**);

months = intck('month', **'15mar2007'd**,**'15mar2008'd**);

qtrs = intck('qtr', **'15mar2007'd**,**'15mar2008'd**);

years = intck('year', **'15mar2007'd**,**'15mar2008'd**);

RUN;

**PROC** **PRINT** data = timeintervals2;

TITLE 'Time intervals as calculated by intck function';

**RUN**;

\*22.13 use the intnx function to calculate date of nextvisit;

**DATA** diet;

input subj **1**-**4** l\_name $ **18**-**23** weight **30**-**32**

+**1** wt\_date mmddyy8. @**43** b\_date mmddyy8.;

nxdate\_b1 = intnx('month', wt\_date, **3**);

nxdate\_b2 = intnx('month', wt\_date, **3**, 'beginning');

nxdate\_m = intnx('month', wt\_date, **3**, 'middle');

nxdate\_e = intnx('month', wt\_date, **3**, 'end');

nxdate\_s = intnx('month', wt\_date, **3**, 'sameday');

format wt\_date b\_date nxdate\_b1 nxdate\_b2

nxdate\_m nxdate\_e nxdate\_s date9.;

DATALINES;

1024 Alice Smith 1 65 125 12/1/05 01/01/60

1167 Maryann White 1 68 140 12/01/05 01/01/59

1168 Thomas Jones 2 190 12/2/05 06/15/60

1201 Benedictine Arnold 2 68 190 11/30/05 12/31/60

1302 Felicia Ho 1 63 115 1/1/06 06/15/58

;

RUN;

**PROC** **PRINT** data=diet;

TITLE 'The data set containing next weight dates';

VAR subj wt\_date nxdate\_b1 nxdate\_b2

nxdate\_m nxdate\_e nxdate\_s;

**RUN**;

\*22.14 SAS informats and formats;

**DATA** inputdates1;

INPUT @**6** date1 mmddyy6. @**13** date2 mmddyy8. @**22** date3 mmddyy10.;

FORMAT date1 ddmmyy10. date2 ddmmyyb10. date3 ddmmyyc10.;

DATALINES;

041008 04-10-08 04 10 2008

;

RUN;

**PROC** **PRINT** data = inputdates1;

TITLE 'The mmddyy informat and the ddmmyy format';

**RUN**;

\*22.15 converting ddmmyy format to mmddyy;

**DATA** inputdates2;

INPUT @**6** date1 ddmmyy6. @**13** date2 ddmmyy8. @**22** date3 ddmmyy10.;

FORMAT date1 mmddyyd10. date2 mmddyyn8. date3 mmddyyp10.;

DATALINES;

100408 10-04-08 10 04 2008

;

RUN;

**PROC** **PRINT** data = inputdates2;

TITLE 'The ddmmyy informat and the mmddyy format';

**RUN**;

\*22.16 date7, date9, and date11 informats w/ weekdate25,

worddate19, worddatx19 formats;

**DATA** inputdates3;

INPUT @**6** date1 date7. @**14** date2 date9. @**24** date3 date11.;

FORMAT date1 weekdate25.

date2 worddate19.

date3 worddatx19.;

DATALINES;

10Apr08 10Apr2008 10-Apr-2008

;

RUN;

**PROC** **PRINT** data = inputdates3;

TITLE 'The date7 informat and the weekdate and worddate formats';

**RUN**;

\*22.17 yearcutoff system option;

OPTIONS YEARCUTOFF=**1920**;

**DATA** twodigits1920;

INPUT date1 mmddyy8.;

FORMAT date1 worddatx20.;

DATALINES;

01/03/20

01/03/21

01/03/49

01/03/50

01/03/51

01/03/99

01/03/00

01/03/01

01/03/19

;

RUN;

**PROC** **PRINT** data=twodigits1920;

title 'Years with two-digits when YEARCUTOFF = 1920';

**RUN**;

\*22.18 change yearcutoff to 1950 to illustrate problem with years printed;

OPTIONS YEARCUTOFF=**1950**;

**DATA** twodigits1950;

INPUT date1 mmddyy8.;

FORMAT date1 worddatx20.;

DATALINES;

01/03/20

01/03/21

01/03/49

01/03/50

01/03/51

01/03/99

01/03/00

01/03/01

01/03/19

;

RUN;

**PROC** **PRINT** data=twodigits1950;

title 'Years with two-digits when YEARCUTOFF = 1950';

**RUN**;

\*22.19 time informats and formats;

**DATA** diet;

input subj **1**-**4** l\_name $ **18**-**23** weight **30**-**32**

+**1** wt\_date mmddyy8. @**43** b\_date mmddyy8.

@**52** wt\_time time8.;

wtm\_fmt1 = wt\_time;

wtm\_fmt2 = wt\_time;

wtm\_fmt3 = wt\_time;

format wtm\_fmt1 hhmm.

wtm\_fmt2 hour5.2

wtm\_fmt3 time8.;

DATALINES;

1024 Alice Smith 1 65 125 12/1/05 01/01/60 00:01:00

1167 Maryann White 1 68 140 12/01/05 01/01/59 00:15:00

1168 Thomas Jones 2 190 12/2/05 06/15/60 12:00:00

1201 Benedictine Arnold 2 68 190 11/30/05 12/31/60 00:00:00

1302 Felicia Ho 1 63 115 1/1/06 06/15/58 23:59:59

;

RUN;

**PROC** **PRINT** data=diet;

title 'The diet data set with formatted weight times';

var subj wt\_time wtm\_fmt1 wtm\_fmt2 wtm\_fmt3;

**RUN**;

\*22.20 using SAS time functions;

**DATA** diet;

input subj **1**-**4** l\_name $ **18**-**23** weight **30**-**32**

+**1** wt\_date mmddyy8. @**43** b\_date mmddyy8.

@**52** wt\_time time8.;

curtime = time();

wt\_hr = hour(wt\_time);

wt\_min = minute(wt\_time);

wt\_sec = second(wt\_time);

wt\_time2 = hms(wt\_hr, wt\_min, wt\_sec);

format curtime wt\_time wt\_time2 time8.;

DATALINES;

1024 Alice Smith 1 65 125 12/1/05 01/01/60 00:01:00

1167 Maryann White 1 68 140 12/01/05 01/01/59 00:15:00

1168 Thomas Jones 2 190 12/2/05 06/15/60 12:00:00

1201 Benedictine Arnold 2 68 190 11/30/05 12/31/60 00:00:00

1302 Felicia Ho 1 63 115 1/1/06 06/15/58 23:59:59

;

RUN;

**PROC** **PRINT** data=diet;

title 'The diet data set with five new variables';

var subj curtime wt\_time wt\_hr wt\_min wt\_sec wt\_time2;

**RUN**;

\*22.21 comparing times;

**DATA** diet (where = ((wt\_time ge **'00:00:00't**)

and (wt\_time le **'12:00:00't**)));;

input subj **1**-**4** l\_name $ **18**-**23** weight **30**-**32**

+**1** wt\_date mmddyy8. @**43** b\_date mmddyy8.

@**52** wt\_time time8.;

time\_int = abs((wt\_time - **'05:00:00't**)/**3600**);

format wt\_time time8. time\_int **4.1**;

DATALINES;

1024 Alice Smith 1 65 125 12/1/05 01/01/60 00:01:00

1167 Maryann White 1 68 140 12/01/05 01/01/59 00:15:00

1168 Thomas Jones 2 190 12/2/05 06/15/60 12:00:00

1201 Benedictine Arnold 2 68 190 11/30/05 12/31/60 00:00:00

1302 Felicia Ho 1 63 115 1/1/06 06/15/58 23:59:59

;

RUN;

**PROC** **PRINT** data=diet;

title 'The subsetted diet data set';

var subj l\_name wt\_time time\_int;

**RUN**;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Code typed in by: Laura J. Simon

Date: 02 March 2008

Code originally created by: Ron Cody

Date: For the SAS Global Forum 2007 Conference in Orlando, FL

The examples contained in this SAS program are from Ron Cody's

"An Introduction to SAS Character Functions" (Paper 217-2007).

The page numbers in the program refer to the page numbers from

this version of the paper:

http://www2.sas.com/proceedings/forum2007/217-2007.pdf

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

OPTIONS NODATE NONUMBER PS=**58** LS=**80**;

/\* Page 1 \*/

/\* How Lengths of Character Variables are Set in a SAS Data Step \*/

/\* #1 \*/

**DATA** chars1;

FILE print;

string = 'abc';

length string $ **7**; /\* Does this do anything \*/

storage\_length = lengthc(string);

display = ":" || string || ":";

put storage\_length= ;

put display= ;

RUN;

/\* Page 2 \*/

/\* #2 \*/

**DATA** chars2;

FILE print;

length string $ **7**; /\* Does this do anything \*/

string = 'abc';

storage\_length = lengthc(string);

display = ":" || string || ":";

put storage\_length= ;

put display= ;

RUN;

/\* Converting Multiple Blanks to a Single Blank \*/

/\* #3 \*/

**DATA** multiple;

INPUT #**1** @**1** name $20.

#**2** @**1** address $30.

#**3** @**1** city $15.

@**20** state $2.

@**25** zip $5.;

name = compbl(name);

address = compbl(address);

city = compbl(city);

DATALINES;

Ron Cody

89 Lazy Brook Road

Flemington NJ 08822

Bill Brown

28 Cathy Street

North City NY 11518

;

RUN;

**PROC** **PRINT** data = multiple NOOBS;

TITLE 'The multiple data set';

ID name;

VAR address city state zip;

**RUN**;

/\* Page 3 \*/

/\* How to Remove Characters from a String \*/

/\* #4 \*/

**DATA** phone;

INPUT phone $ **1**-**15**;

phone1 = compress(phone);

phone2 = compress(phone,'(-) ');

DATALINES;

(908)235-4490

(201) 555-77 99

;

RUN;

**PROC** **PRINT** data = phone NOOBS;

TITLE 'The phone data set';

**RUN**;

/\* Page 4 \*/

/\* Character Data Verification \*/

/\* #5 \*/

**DATA** verify;

INPUT @**1** id $3.

@**5** answer $5.;

position = verify(answer,'abcde');

DATALINES;

001 acbed

002 abxde

003 12cce

004 abc e

;

**PROC** **PRINT** data = verify NOOBS;

TITLE 'The verify data set';

**RUN**;

/\* #6 \*/

**DATA** trailing;

length string $ **10**;

string = 'abc';

pos1 = verify(string,'abcde');

pos2 = verify(trim(string),'abcde');

RUN;

**PROC** **PRINT** data = trailing NOOBS;

TITLE 'The trailing data set';

**RUN**;

/\* Page 5 \*/

/\* Substring Example \*/

/\* #7 \*/

**DATA** pieces\_parts;

INPUT id $ **1**-**9**;

length state $ **2**;

state = substr(id,**1**,**2**);

num = input(substr(id,**7**,**3**),**3.**);

DATALINES;

NYXXXX123

NJ1234567

;

RUN;

**PROC** **PRINT** data = pieces\_parts NOOBS;

TITLE 'The pieces\_parts data set';

**RUN**;

/\* Page 6 \*/

/\* Using the SUBSTR Function on the Left-Hand Side of the Equal Sign \*/

/\* #8 \*/

**DATA** pressure;

INPUT sbp dbp @@;

length sbp\_chk dbp\_chk $ **4**;

sbp\_chk = put(sbp,**3.**);

dbp\_chk = put(dbp,**3.**);

if sbp gt **160** then substr(sbp\_chk,**4**,**1**) = '\*';

if dbp gt **90** then substr(dbp\_chk,**4**,**1**) = '\*';

DATALINES;

120 80 180 92 200 110

;

RUN;

**PROC** **PRINT** data = pressure NOOBS;

TITLE 'The pressure data set';

**RUN**;

/\* Page 7 \*/

/\* Unpacking a String \*/

/\* #9 \*/

**DATA** pack;

INPUT string $ **1**-**5**;

DATALINES;

12345

8 642

;

RUN;

**DATA** unpack;

SET pack;

array x[**5**];

DO j = **1** to **5**;

x[j] = input(substr(string,j,**1**),**1.**);

END;

DROP j;

RUN;

**PROC** **PRINT** data = unpack NOOBS;

TITLE 'The unpack data set';

**RUN**;

/\* Parsing a String \*/

/\* #10 \*/

**DATA** parse;

INPUT long\_str $ **1**-**80**;

array pieces[**5**] $ **10** piece1-piece5;

do i = **1** to **5**;

pieces[i] = scan(long\_str,i,',.! ');

end;

drop long\_str i;

DATALINES;

this line,contains!five.words

abcdefghijkl xxx yyy

;

RUN;

**PROC** **PRINT** data = parse NOOBS;

TITLE 'The parse data set';

**RUN**;

/\* Page 8 \*/

/\* Using the SCAN function to Extract a Last Name \*/

/\* #11 \*/

**DATA** first\_last;

INPUT @**1** name $20.

@**21** phone $13.;

\*\*\* The next statement extracts the last name from name;

last\_name = scan(name,-**1**,' '); /\* scans from the right \*/

DATALINES;

Jeff W. Snoker (908)782-4382

Raymond Albert (732)235-4444

Alred Edward Newman (800)123-4321

Steven J. Foster (201)567-9876

Jose Romerez (516)593-2377

;

RUN;

**PROC** **REPORT** data = first\_last NOWINDOWS;

TITLE 'Names and Phone Numbers in Alphabetical Order (by Last Name)';

COLUMNS name phone last\_name;

DEFINE last\_name / order noprint width = **20**;

DEFINE name / display 'Name' left width = **20**;

DEFINE phone / display 'Phone Number' width = **13** format=$13.;

**RUN**;

/\* Page 9 \*/

/\* Locating the Position of One String within Another String \*/

/\* #12 \*/

**DATA** locate;

INPUT string $ **1**-**10**;

first = index(string,'xyz');

first\_c = indexc(string,'x','y','z');

DATALINES;

abcxyz1234

1234567890

abcx1y2z39

abczzzxyz3

;

RUN;

**PROC** **PRINT** data = locate NOOBS;

TITLE 'The locate data set';

**RUN**;

/\* Changing Lower Case to Upper Case and Vice Versa \*/

/\* Page 10 \*/

/\* #13 \*/

**DATA** up\_down;

length a b c d e $ **1**;

INPUT a b c d e x y;

DATALINES;

M f P p D 1 2

m f m F M 3 4

;

RUN;

**DATA** upper;

SET up\_down;

array all\_c[\*] \_character\_;

DO i = **1** to dim(all\_c);

all\_c[i] = upcase(all\_c[i]);

END;

DROP i;

RUN;

**PROC** **PRINT** data = upper NOOBS;

TITLE 'The upper data set';

**RUN**;

/\* Converting String to Proper Case \*/

/\* #14 \*/

**DATA** proper;

INPUT Name $40.;

propname = propcase(Name);

DATALINES;

rOn coDY

the tall and the short

the "%$#@!" escape

;

RUN;

**PROC** **PRINT** data = proper NOOBS;

TITLE 'The proper data set';

**RUN**;

/\* Page 11 \*/

/\* Substituting One Word for Another in a String \*/

/\* #15 \*/

**DATA** convert;

INPUT @**1** address $20.;

\*\*\* Convert Street, Avenue, and Road

to their abbreviations;

address = tranwrd(address,'Street','St.');

address = tranwrd(address,'Avenue','Ave.');

address = tranwrd(address,'Road','Rd.');

DATALINES;

89 Lazy Brook Road

123 River Rd.

12 Main Street

;

RUN;

**PROC** **PRINT** data = convert;

TITLE 'The convert data set';

**RUN**;

/\* Fuzzy Merging: The SPEDIS Function \*/

/\* Page 12 \*/

/\* #16 \*/

**DATA** compare;

length string1 string2 $ **15**;

INPUT string1 string2;

points = spedis(string1, string2);

DATALINES;

same same

same sam

firstletter xirstletter

lastletter lastlettex

receipt reciept

;

RUN;

**PROC** **PRINT** data = compare NOOBS;

TITLE 'The compare data set';

**RUN**;

/\* Demonstrating the "ANY" Functions \*/

/\* #17 \*/

**DATA** find\_alpha\_digit;

input string $20.;

first\_alpha = anyalpha(string);

first\_digit = anydigit(string);

DATALINES;

no digits here

the 3 and 4

123 456 789

;

RUN;

**PROC** **PRINT** data = find\_alpha\_digit NOOBS;

TITLE 'The find\_alpha\_digit data set';

**RUN**;

/\* Page 13 \*/

/\* Demonstrating the "NOT" Functions \*/

/\* #18 \*/

**DATA** data\_cleaning;

input string $20.;

only\_alpha = notalpha(trim(string));

only\_digit = notdigit(trim(string));

DATALINES;

abcdefg

1234567

abc123

1234abcd

;

RUN;

**PROC** **PRINT** data = data\_cleaning NOOBS;

TITLE 'The data\_cleaning data set';

**RUN**;

/\* Page 14 \*/

/\* The New Concatenation Functions \*/

/\* #19 \*/

**DATA** join\_up;

length cats $ **6** catx $ **17**;

string1 = 'ABC ';

string2 = ' XYZ ';

string3 = '12345';

cats = cats(string1, string2);

catx = catx('\*\*\*', string1, string2, string3);

RUN;

**PROC** **PRINT** data = join\_up NOOBS;

TITLE 'The join\_up data set';

VAR string1 string2 string3 cats catx;

**RUN**;

/\* The Length, Lengthn, and LengthC Functions \*/

/\* Page 15 \*/

/\* #20 \*/

**DATA** how\_long;

one = 'ABC ';

two = ' '; /\* character missing value \*/

three = 'ABC XYZ';

length\_one = length(one);

lengthn\_one = lengthn(one);

lengthc\_one = lengthc(one);

length\_two = length(two);

lengthn\_two = lengthn(two);

lengthc\_two = lengthc(two);

length\_three = length(three);

lengthn\_three = lengthn(three);

lengthc\_three = lengthc(three);

RUN;

**PROC** **PRINT** data = how\_long NOOBS;

TITLE 'The how\_long data set';

**RUN**;

/\* Page 16 \*/

/\* Counting Occurrences of Characters or Substrings Using the

COUNT and COUNTC Functions \*/

/\* #21 \*/

**DATA** Dracula;

INPUT string $20.;

count\_a\_or\_b = count(string,'ab');

countc\_a\_or\_b = countc(string,'ab');

count\_abc = count(string,'abc');

countc\_abc = countc(string,'abc');

case\_a = countc(string,'a','i');

DATALINES;

xxabcxabcxxbbbb

cbacba

aaAA

;

RUN;

**PROC** **PRINT** data = Dracula NOOBS;

TITLE 'The Dracula data set';

**RUN**;