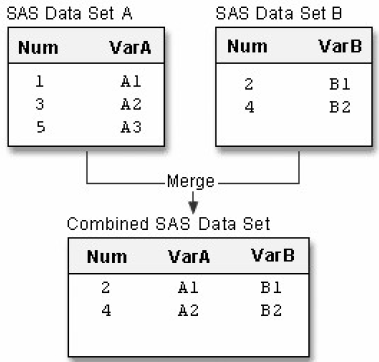
**Chapter XII – Combining SAS Data Set**

1. **One-to-One Merging**

In one-to-one merging you can read different data sets, or you can read the same data set more than once, as if you were reading from separate data sets.

* Basic code:

**DATA** *output-SAS-data-set***;**

**SET** *SAS-data-set-1***;**

**SET** *SAS-data-set-2***;**

**RUN;**

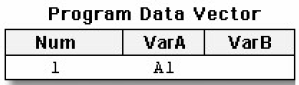
*output-SAS-data-set* names the data set to be created

*SAS-data-set-1* and *SAS-data-set-2* specify the data sets to be read.

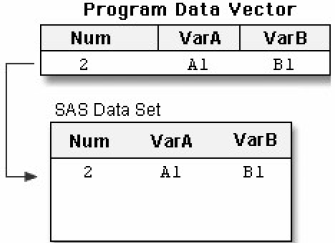
* When processing One-to-One Merging
* 新的data set会包含所有输入的data set。如果两组data set拥有相同的名字，那么最先被读取的data set中的信息 将被后读入的data set中的信息替代
* 在新的data set中number of observation会自动取旧的data set中的最小值。Observation在新的data set中的位置会与它在旧的data set中的位置相同。The **DATA** step stops after it has read the last observation from the smallest data set.
* Eg:

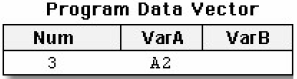
**DATA** one2one;

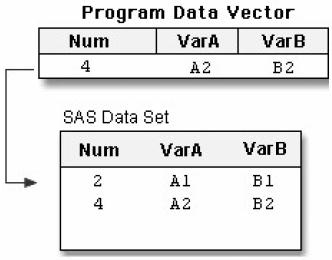
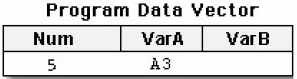
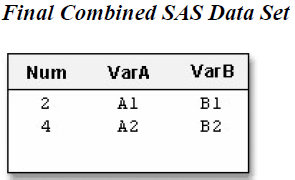
**SET** A;

**SET** B;

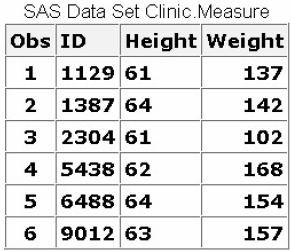
**RUN**;

* The process of O2O Merging
* The first SET statement reads the first observation from data set A into the program data vector.
* The second SET statement reads the first observation from data set B into the program data vector and SAS writes the contents of the program data vector to the new data set. The value for Num from data set B overwrites the value for Num from data set A.



* The first SET statement reads the second observation from data set A into the program data vector.
* The second SET statement reads the second observation from data set B, and SAS writes the contents of the program data vector to the new data set. The value for Num from data set B overwrites the value for Num from data set A.
* The first SET statement reads the third observation from data set A into the program data vector
* The second SET statement reads the end of file in data set B, which stops the DATA step processing with no further output written to the data set. The last observation in data set A is read into the program data vector, but it is not written to the output data set.
* Eg:

合并Clinic.Patients和Clinic.Measure

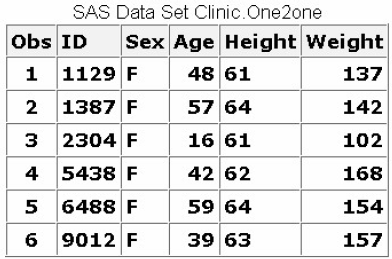
**DATA** clinic.one2one;

**SET** clinic.patients;

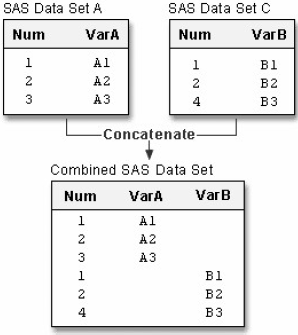
**IF** age<60;

**SET** clinic.measure;

**RUN**;

**Result**

1. **Concatenating**

Appends the observations from one data set to another data set, which allow you to specify a list of data set names in the SET statement

* Basic code:

*output-SAS-data-set* names the data set to be created

*SAS-data-set-1* and *SAS-data-set-2* specify the data sets to be read.

**DATA** *output-SAS-data-set***;**

**SET** *SAS-data-set-1*

*SAS-data-set-2***;**

**RUN;**

* Eg:

**DATA** concat;

**SET** a c;

**RUN**;

* When processing concatenating:
* 两个样本的Num或者common variable必须拥有相同的属性，否则SAS自动停止运行DATA step并且弹出error massage以说明variables are incompatible
* 如果两个样本的length, label, format, and informat attribute不同，SAS自动选择第一个data set的相应属性

1. **Interleaving**

Concatenating with **BY** statement (in **SET** statement), which allow the user intersperses observations from two or more data sets, based on one or more common variables

* Basic code:

*Output-SAS-data-set* names the data set to be created

*SAS-data-set-1 SAS-data-set-2* specify the data set to be read

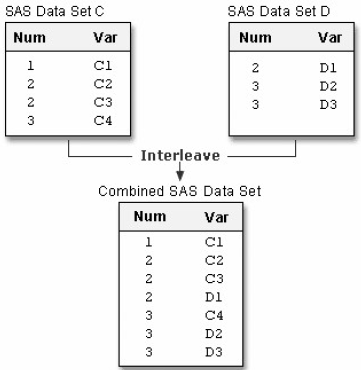
*variable(s)* specifies one or more variables that are used to interleave observations

**DATA** *output-SAS-data-set***;**

**SET** *SAS-data-set-1 SAS-data-set-2***;**

**BY** *variable(s)***;**

**RUN;**



* Eg:

**DATA** interlv;

**SET** c d;

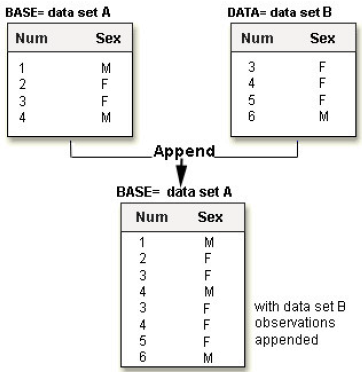
**BY** num;

**RUN**;

1. **Appending**

Appending and concatenating具有相似性，但是concatenating在**DATA** step中建立了新的data set，而**PROC APPEND**只是在BASE的observations基础上添加另一个data set中的observations。在Appending中SAS不会建立新的data set

* Basic code

**PROC APPEND *BASE****=SAS-data-set*

***DATA****=SAS-data-set***;**

**RUN;**

*BASE=* names the data set to which observations are added

*DATA=* names the data set containing observations that are added to the base data set

* Eg:

proc append base=A

data=B;

run;

* 注意当使用Appending时：
* 一次只能读取两个data set
* SAS不会读取base data set中的observation
* The variable information in the descriptor portion of the base data set cannot change.
* **FORCE** option

当使用PROC APPEND时，两组data set的variable definitions不相匹配时，使用FORCE option来完成combine

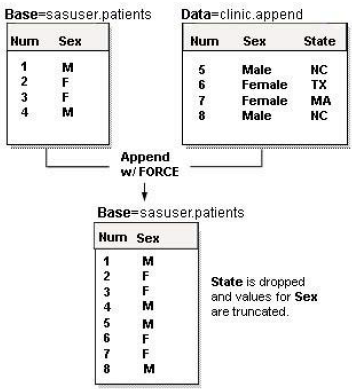
* Basic code:

**PROC APPEND** *BASE=SAS-data-set*

DATA=SAS-data-set FORCE;

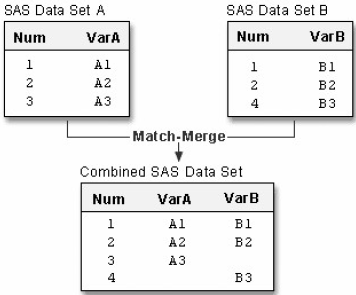
**RUN;**

The FORCE option is needed when the DATA= data set contains variables that meet any one of the following criteria:

* They are not in the BASE= data set.
* They are variables of a different type They are longer than the variables in the BASE= data set.
* When processing Appending
* 如果*DATA*= data set中的variable比*BASE*= data set中的长，SAS会自动把*DATA*的variable中截取到与*BASE*中variable等长的部分
* 如果两者type of variable不相同，SAS会用missing value替代所有*DATA*= data set中的variable，同时保留*BASE*= data set中的specified的variable type
* 如果*BASE*= data set中的部分variable不被*DATA*= data set所包含，相应observation会在*DATA*= data set中被默认为missing value。如果*DATA*= data set中的部分variable不被*BASE*= data set所包含，相应部分会被drop from output

1. **Match-Merging**

Combine observations from two or more data sets into a single observation in a new data set according to the values of a common variable, all observations from all input data sets will be included within the new data set

* Basic code:

**DATA** *output-SAS-data-set***;**

**MERGE** *SAS-data-set-1 SAS-data-set-2***;**

**BY** <DESCENDING> *variable(s)***;**

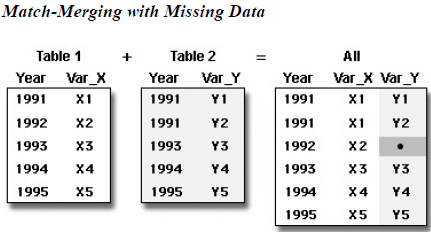
**RUN;**

*output-SAS-data-set* names the data set to be created.

*SAS-data-set-1 SAS-data-set-2* specify the data sets to be read.

*variable(s)* in the **BY** statement specifies one or more variables whose values are used to match observations.

注意：You cannot use the **DESCENDING** option with indexed data sets because indexes are always stored in ascending order.

* Eg:

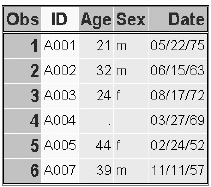
**DATA** merged;

**MERGE** a b;

**BY** num;

**RUN**;

* 注意：
* You can add statements and options to select only matching observations
* If an input data set doesn't have any observations for a particular value of the by-variable, then the observation in ***Sorting clinic.demog***

the output data set contains missing values for the variables

that are unique to that input data set.

* Eg:

**PROC** **SORT** data=clinic.demog;

**BY** **DESCENDING** id;

**RUN**;

**PROC** **SORT** data=clinic.visit;

**BY** **DESCENDING** id;

**RUN**; ***Sorting clinic.visit***

**DATA** clinic.merged;

**MERGE** clinic.demog clinic.visit;

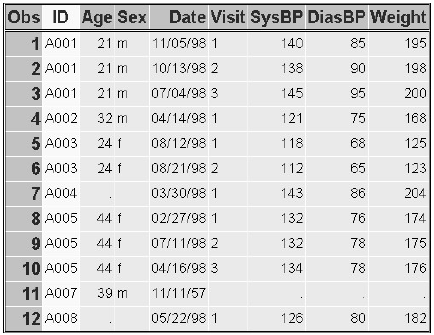
**BY** **DESCENDING** id;

**RUN**;

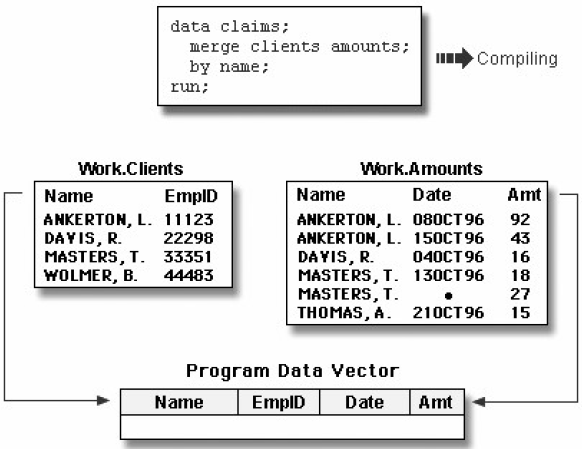
**PROC** **PRINT** data=clinic.merged;

**RUN**;

***Result***



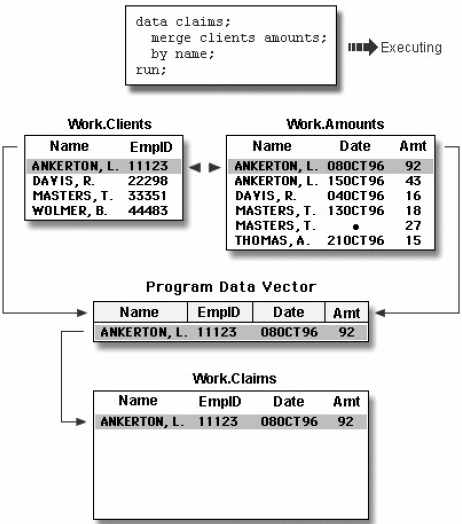
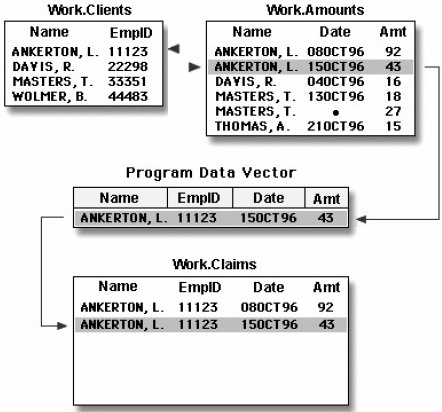
注意：the **DESCENDING** option in the **BY** statements in both the **PROC** **SORT** steps and the **DATA** step. If you omit the **DESCENDING** option in the **DATA** step, you generate error messages about improperly sorted BY variables.（Quiz 8）

1. **Match-Merge Processing**

* Compilation Phase

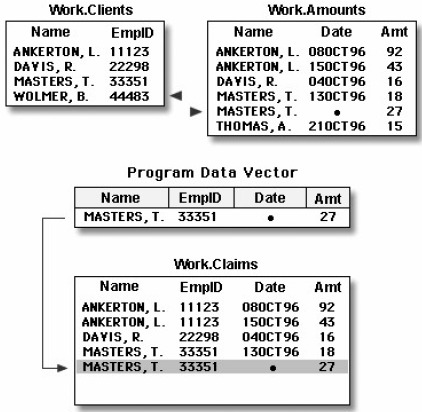
If variables that have the same name appear in more than one data set, the variable from the first data set that contains the variable (in the order listed in the MERGE statement) determines the length of the

* variable.Check syntax
* Compile program (transfer program to machine code)
* Set up descriptor information for output data set
* Create PDV
* Execution Phase
* Moving the pointers down each observation of each data set and checking to see whether the BY values match
* If matches, observations are read into the PDV in order. Same-named variable are overwritten by values of the same-named variable in subsequent data sets. SAS writes the combined observation to the new data set and retains the values in the PDV until the BY value changes in all the data sets.
* If does not matches, SAS determines which BY value comes first and reads the observation that contains this value into the PDV

***Matches PDV***

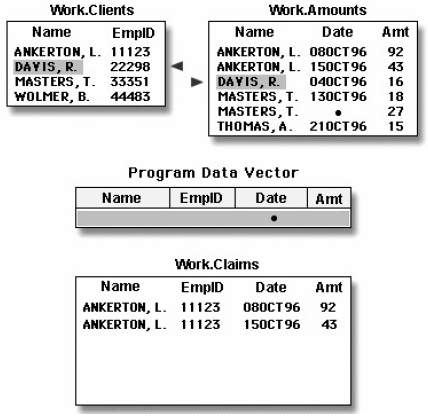
* Observations that have missing values for the BY variable appear at the top of the output data set because missing values sort first in ascending order.

***Initializing the PDV to Missing***



* If an input data set doesn't have a matching BY value, then the observation in the output data set contains missing values for the variables that are unique to that input data set.

***Handling Unmatched Observations and Missing Values***



1. **Rename Variables**

DATA step match-merging overwrites values of the like-named variable in the first data set in which it appears with values of the like-named variable in subsequent data sets.（Quiz 5）

* Basic code:

**(RENAME=(***old-variable-name=new-variable-name***))**

**RENAME**= option, in parentheses, follows the name of each data set that contains one or more variables to be renamed

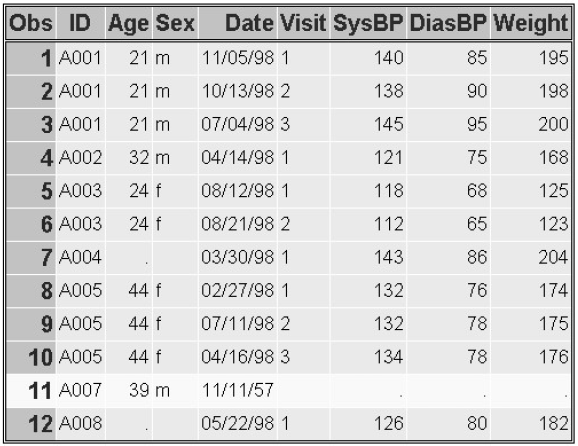
*old-variable-name* specifies the variable to be renamed

*new-variable-name* specifies the new name for the variable.

* Eg:

Clinic.Demog contains the variable Date (date of birth), and Clinic.Visit also contains Date (date of the clinic visit in 1998). The DATA step below overwrites the date of birth with the date of the clinic visit.

|  |  |
| --- | --- |
| Original code: | After rename |
| **DATA** clinic.merged; | **DATA** clinic.merged; |
| **MERGE** clinic.demog clinic.visit; | **MERGE** clinic.demog  (rename=(date=BirthDate)) |
|  | clinic.visit  (rename=(date=VisitDate)); |
| **BY** id; | **BY** id; |
| **RUN**; | **RUN**; |
| **PROC** **PRINT** data=clinic.merged; | **PROC** **PRINT** data=clinic.merged; |
| **RUN**; | **RUN**; |



1. **Excluding Unmatched Observations**

Select observations match for two or more specified input data sets

* exclude unmatched observations from your output data set by using **IN**= data set and **IF** statement
* the **IN**= data set option to create and name a variable that indicates whether the data set contributed data to the current observation
* the subsetting **IF** statement to check the **IN** = values and write to the merged data set only matching observations.
* Basic code:
* **IN**= data set option]

**(IN=** *variable***)**

**IN**= option, in parentheses, follows the data set name…………

* The value of the variable is **1** if the data set contributed data to the current observation. Otherwise, its value is **0**. The first **IN**= creates the temporary variable indemog, which is set to **1** when an observation from Clinic.Demog contributes to the current observation. Otherwise, it is set to **0**. Likewise, the value of invisit depends on whether Clinic.Visit contributes to an bservation or not.
* **IF** statement
* SAS evaluates the expression within an **IF** statement to produce a result that is either nonzero, zero, or missing. A nonzero and nonmissing result causes the expression to be true; a zero or missing result causes the expression to be false.
* you can specify the subsetting **IF** statement from the previous example in either of the following ways. The first **IF** statement checks specifically for a value of **1**. The second IF statement checks for a value that is neither missing nor **0** (which for **IN**= variables is always **1**).
* Eg:

**IF** indemog=1 and invisit=1;

OR **IF** indemog and invisit;

* Eg followed after the eg in7

Match-merge the data sets Clinic.Demog and Clinic.Visit and select only observations that appear in both data sets.

**DATA** clinic.merged;

**MERGE** clinic.demog(**IN**=indemog

**RENAME**=(date=BirthDate))

clinic.visit(**IN**=invisit

**RENAME**=(date=VisitDate));

**BY** id;

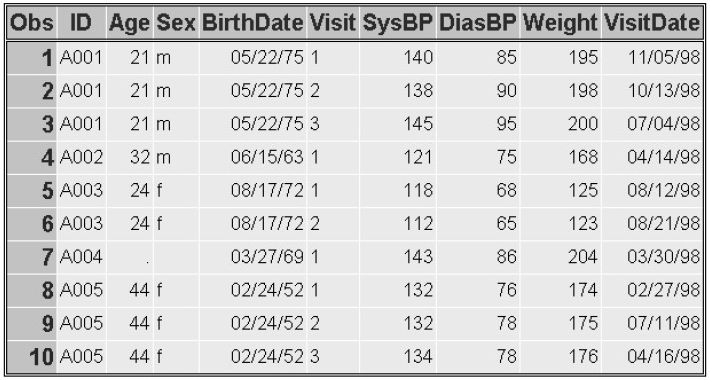
**IF** indemog=1 and invisit=1;

**RUN**;

**PROC** **PRINT** data=clinic.merged;

**RUN**;

***Result* (**Compare with result in 7, unmatched observations are deleted**)**



1. **Selecting Variables**

* If you don't reference certain variables and you don't want them to appear in the new data set, specify them in the **DROP**= option in the **MERGE** statement.

Eg:

**MERGE** clinic.demog(**IN**=indemog **RENAME**=(date=BirthDate))

clinic.visit(**DROP**=weight **IN**=invisit **RENAME** =(date=VisitDate));

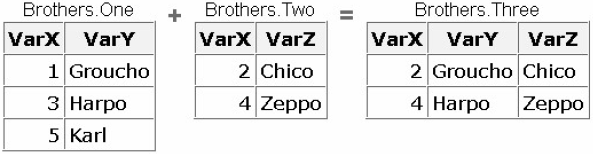
* If you *do* need to reference a variable in the original data set (in a subsetting **IF** statement, for example), then you must specify the variable in the **DROP**= option in the **DATA** statement. Otherwise, you may get unexpected results and your variable will be uninitialized.

Eg:

**DATA** clinic.merged (**DROP**=id);

练习

1. Which program will combine Brothers.One and Brothers.Two to produce Brothers.Three?\



1. **DATA** brothers.three;

**SET** brothers.one;

**SET** brothers.two;

**RUN**;

1. **DATA** brothers.three;

**SET** brothers.one brothers.two;

**RUN**;

1. **DATA** brothers.three;

**SET** brothers.one brothers.two;

**BY** varx;

**RUN**;

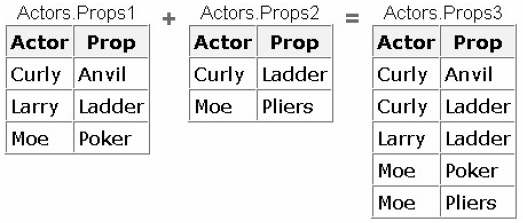
1. **DATA** brothers.three;

**MERGE** brothers.one brothers.two;

**BY** varx;

**RUN**;

1. Which program will combine Actors.Props1 and Actors.Props2 to produce Actors.Props3?



* + 1. **DATA** actors.props3;

**SET** actors.props1;

**SET** actors.props2;

**RUN**;

* + 1. **DATA** actors.props3;

**SET** actors.props1 actors.props2;

**RUN**;

* + 1. **DATA** actors.props3;

**SET** actors.props1 actors.props2;

**BY** actor;

**RUN**;

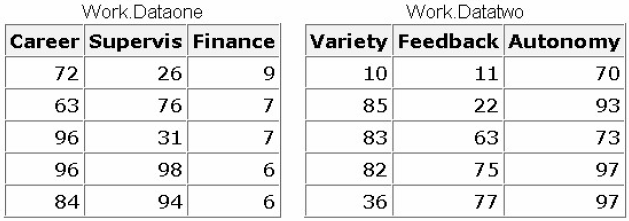
* + 1. **DATA** actors.props3;

**MERGE** actors.props1 actors.props2;

**BY** actor;

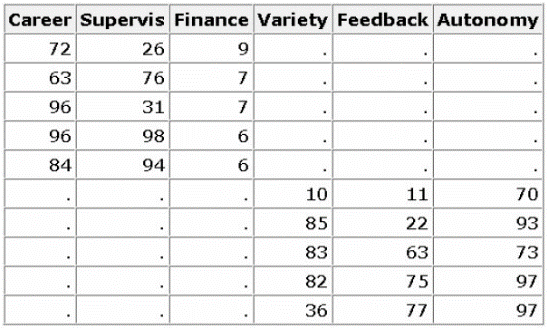
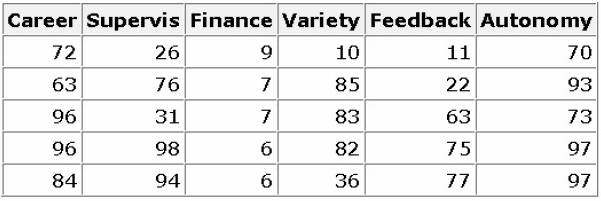
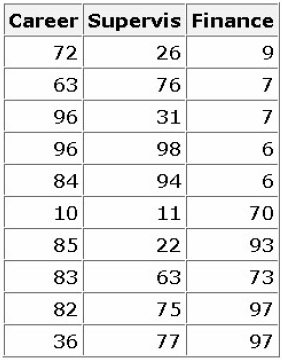
**RUN**;

1. If you submit the following program, which new data set is created?

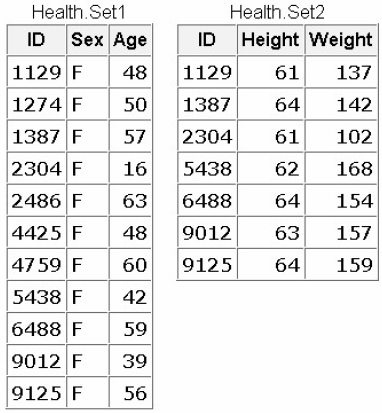
**DATA** work.jobsatis;

**SET** work.dataone work.datatwo;

**RUN**;

1. 
2. 
3. 
4. none of the above
5. If you concatenate the data sets below in the order shown, what is the value of Sale in observation 2 of the new data set?
6. missing
7. $30,000
8. $40,000
9. you cannot concatenate these data sets
10. What happens if you merge the following data sets by the variable SSN?



1. The values of Age in the 1st data set overwrite the values of Age in the 2nd data set.
2. The values of Age in the 2nd data set overwrite the values of Age in the 1st data set.
3. The DATA step fails because the two data sets contain same-named variables that have different values.
4. The values of Age in the 2nd data set are set to missing.
5. Suppose you merge data sets Health.Set1 and Health.Set2 below:

Which output does the following program create?

**DATA** work.merged;

**MERGE** health.set1(**IN**=in1) health.set2(**IN**=in2);

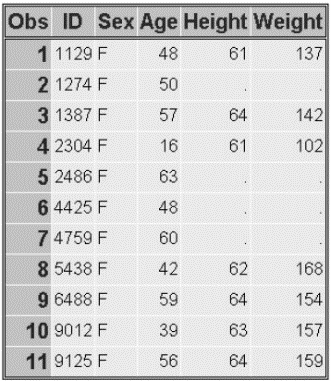
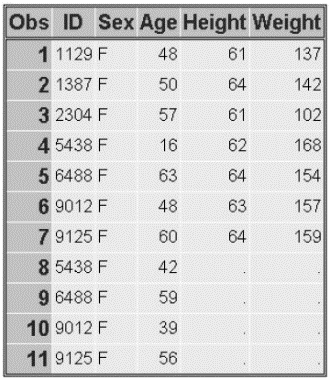
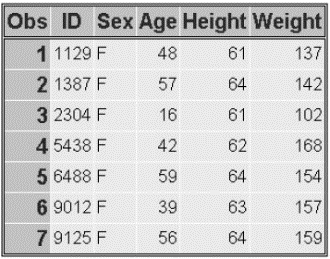
**BY** id;

**IF** in1 and in2;

**RUN**;

**PROC PRINT** data=work.merged;

**RUN**;

1. 
2. 
3. 
4. none of the above
5. The data sets Ensemble.Spring and Ensemble.Sum both contain a variable named Blue. How do you prevent the values of the variable Blue from being overwritten when you merge the two data sets?
   * 1. **DATA** ensemble.merged;

**MERGE** ensemble.spring(**IN**=blue) ensemble.summer;

**BY** fabric;

**RUN**;

* + 1. **DATA** ensemble.merged;

**MERGE** ensemble.spring(**OUT**=blue) ensemble.summer;

**BY** fabric;

**RUN**;

* + 1. **DATA** ensemble.merged;

**MERGE** ensemble.spring(leather blue=navy) ensemble.summer;

**BY** fabric;

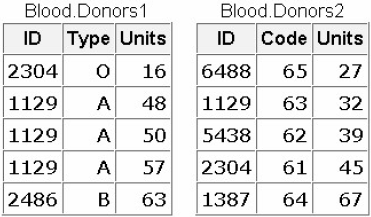
**RUN**;

* + 1. **DATA** ensemble.merged;

**MERGE** ensemble.spring(**RENAME**=(blue=navy)) ensemble.summer;

**BY** fabric;

**RUN**;

1. What happens if you submit the following program to merge Blood.Donors1 and Blood.Donors2, shown below?

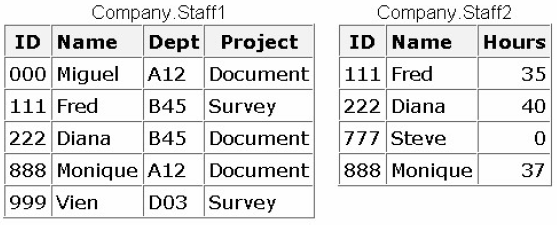
**DATA** work.merged;

**MERGE** blood.donors1 blood.donors2;

**BY** id;

**RUN**;

1. The Merged data set contains some missing values because not all observations have matching observations in the other data set.
2. The Merged data set contains eight observations.
3. The DATA step produces errors.
4. Values for Units in Blood.Donors2 overwrite values of Units in Blood.Donors1.
5. If you merge Company.Staff1 and Company.Staff2 below by ID, how many observations does the new data set contain?



1. 4
2. 5
3. 6
4. 9
5. If you merge data sets Sales.Reps, Sales.Close, and Sales.Bonus by ID, what is the value of Bonus in the third observation in the new data set?



1. $4,000
2. $3,000
3. missing
4. can't tell from the information given