

Part D. More on EpiData software

Part D: More on EpiData software

Exercise 1: Relational database and aggregating vs from “Long-to-wide”

Exercise 2: A statistical process control chart

Exercise 3: A simplified survival analysis

Exercise 4: Creating a menu for standard reports

Introductory note

Part D will address operationally relevant concepts in data collection and data analysis:

- How do we deal with a situation, where each individual has a varying number of observations?
- How do we determine statistically relevant deviations from a proportion over an observation period when the denominator varies with each time subset over the observation period?
- What is survival analysis and how to deal with it in EpiData Analysis?
- How to make a menu-driven, HTML-based EpiData Analysis interface to run standard reports?

Exercise 1: A relational database and “Aggregating” vs from “long-to-wide”

At the end of this exercise you should be able to:

- a. Create a relational database for a varying number of observations
- b. Merge a child file to the parent file
- c. Recognizing when to use “Aggregate” and when to transpose data
- d. Transpose multiple observations (columns) into a single record (row)

Not all laboratories keep their registers as The Union and WHO recommend for the Tuberculosis Laboratory Register, where 1 line corresponds to 1 examinee rather than to 1 examination. In fact, many laboratories enter the results for each examination on one line. If such an approach is chosen, we may find a register as follows:

Patient	Date of exam	Sex	Marital status	Blood sugar	Sputum	Result
A	24-Mar-2007	Male	Married	6.3	Muroid	1+
B	24-Mar-2007	Male	Divorced	4.9	Muco-purulent	Neg
C	24-Mar-2007	Female	Single	5.2	Purulent	Neg
D	24-Mar-2007	Female	Widowed	7.3	Blood-tinged	2 per 100
A	25-Mar-2007	Male		7.3	Salivary	Neg
D	25-Mar-2007	Female		7.4	Muroid	2+
A	26-Mar-2007			7.2	Purulent	1+
C	26-Mar-2007	Female		4.8	Muco-purulent	Neg
E	27-Mar-2007	Male	Married	8.2		1+
F	27-Mar-2007	Female	Annulled	7.4	Purulent	Neg
G	27-Mar-2007	Male	Cohabiting	6.9	Salivary	Neg
G	28-Mar-2007	Male		7.2	Muroid	2+
E	28-Mar-2007	Male		7.9	Purulent	2+
F	31-Mar-2007	Female		7.2	Muco-purulent	3+
H	31-Mar-2007		Married	6.6	Muroid	Neg
I	31-Mar-2007	Male	Separated	8.3	Salivary	Neg
H	1-Apr-2007	Female		6.9	Muco-purulent	1+
F	1-Apr-2007	Female	Engaged	7.7	Purulent	2+
I	1-Apr-2007	Male	Single	8.0	Muroid	8 per 100
G	1-Apr-2007	Male		7.6	Muco-purulent	1+
K	2-Apr-2007	Female	Married	4.5	Purulent	Neg
I	2-Apr-2007	Male		8.2	Muco-purulent	
H	2-Apr-2007	Female		6.6	Muroid	1+
I	3-Apr-2007	Male		8.1	Muroid	1+

This type of a register requires a different approach to both data entry and data analysis than we used before. Two important things need to be considered:

- 1) The same patient may appear again and again on sequential dates

2) Not every patient has the same number of visits

Some patient characteristics do not change over time such as, in this example, the identity of the patient, sex, and marital status (well, perhaps not during these short intervals). Others do change, such as the date of examination, blood sugar, the aspect of the sputum and the sputum smear examination result.

To capture such information in a single data entry form as was done in Part A would be very inconvenient: 1) one would have to anticipate the maximum of allowable visits, and 2) if one patient has a single visit, one would still have to complete all fields with the codes for missing values up to the maximum allotted if we insist that all fields must be MUSTENTER fields.

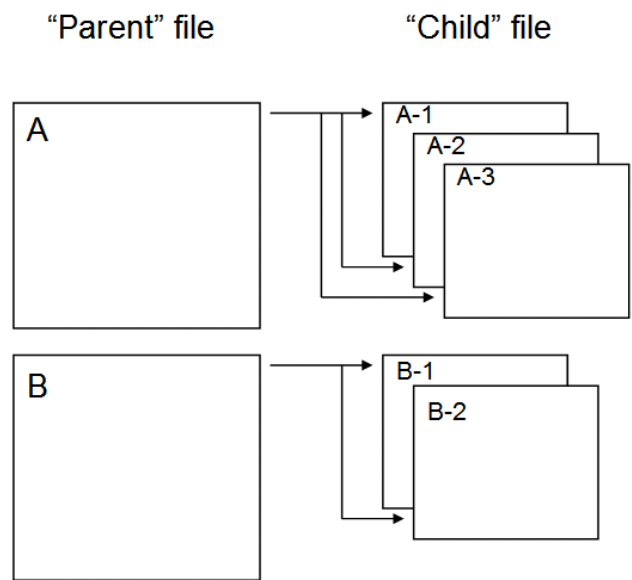
Rule: *If an individual has a single observation for each variable or a fixed number of observations for each variable, then a single EpiData Entry triplet is the best solution. If an individual has a variable number of observations for each variable, the choice is a relational data base.*

Building a relational database requires determining which information is static for an individual and which changes over repeated observations.

EpiData Entry

Introduction

What is the structure of such a relational database? The figure below outlines a relational database with two levels.



At the Level 1 (the "Parent" file), we may have patients, denoted here with A and B. At Level 2 (the "Child" file), denoted here as A-1, A-2, and A-3, may reflect different visits to a clinic where each time the blood sugar, blood pressure, and weight are measured. Each

individual may have at least one, but up to an unlimited number of such visits, and the number of visits varies for each individual.

Note: Because of the simplicity, it is always preferable to have a single data entry form.

However, if the data available concern an individual with fixed information (e.g. age, sex, etc) on one hand, and variable information (e.g. serial examinations) and there are fixed sets of serial examinations (e.g. repeated measures of blood sugar and blood pressure) and each examinee has a varying number of examinations, then it may become more efficient to use a relational database.

In the following, we will call the records at level 1 (the parent file) records on **Examinees** (a person) and the records at level 2 records on **Examinations**.

For building the relational database, we want to collect different information on each level and link the two levels, so that at some point during entering information for an examinee a trigger will open the file to enter records on examinations for the specific examinee. The information for one or more records about examinations is entered until one wishes to return back to the field following the trigger at the upper level.

Important components for the structuring of the relation between the two EpiData Entry triplets are:

- The two files have a common identifier field. In the Check file this field must be KEY UNIQUE at Level 1 (Examinees are unique) and KEY at Level 2 for examinations (where there might be multiple records for the same examinee). This identifier might be called IDPERSON.
- The Examination file must have its own unique identifier, which might be called IDEXAM.
- At an appropriate point in the Check file at Level 1 (Examinee) a RELATE command must be entered, making the relation to Level 2 (examinations).
- After returning from Level 2 back up to Level 1, one must arrive “somewhere” and it will be the next field after the field from which the diversion occurred. Thus, the field with the RELATE command at Level 1 should not be the last field of the Level 1 (examinee) record.

Level 1: Check file of the examinee :	Level 2: Check file of the examination :
1_examinee.rec	2_examination.rec
idperson MUSTENTER KEY UNIQUE 1 END	
var01 RELATE idperson 2_examination.rec END	Idperson NOENTER KEY 1 END
	Idexam MUSTENTER

	KEY UNIQUE END
varlast	

If there is a single record to be entered in the child record, then the RELATE command in the parent record must indicate so with 1 after the name of the child file:

```
RELATE idperson 2_examination.rec 1
```

If there is frequently only non-variable information on the examinee, but no variable examination information, one can introduce a command so that there is not automatically a deviation to the file on examinations:

```
var01
  MUSTENTER
  AFTER ENTRY
  HELP "Is there information on examinations?\n y: yes\n n: no" KEYS="YN"
  TYPE=CONFIRMATION
  IF RESULTLETTER="Y" THEN
    RELATE idparent 2_examination.rec
  ENDIF
  END
END

Varlast
  MUSTENTER
  END
```

Note the use of KEYS and the command RESULTLETTER (see more about it in the Help file searching for RESULTLETTER).

To return from the lower to the upper level, use the key F10 (there are other possibilities) and best write this into the data entry form.

As always, start with a *data documentation sheet* for the fields that will be used for the above dataset before you start working in EpiData Entry. You must define which fields belong to the “parent” file (defined as the file that has one-time information for the individual) and which fields will belong to the “child” file (defined as the related file containing changing information for the individual).

The working example

You will create two final EpiData Entry triplets with the following names:

```
d_ex01_patient.qes
d_ex01_patient.rec
d_ex01_patient.chk

d_ex01_examination.qes
d_ex01_examination.rec
d_ex01_examination.chk
```

Note the following:

- 1) The identifier that relates the two files must be KEY UNIQUE in the parent file and KEY in the child file.

- 2) The RELATE command should follow after entering the value for the second to last field in the parent file.

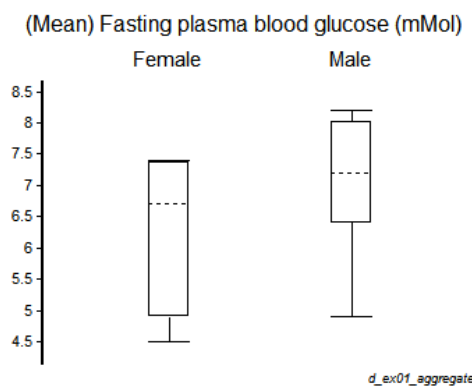
EpiData Entry will automatically ask if you enter the same identifier for a second time (KEY UNIQUE) whether you want to see the record where this identifier was used before. Upon approval, you will get to it and you can then add the information for the additional examination starting from that record in the parent file.

You will note during data entry that this requires a lot of concentration and errors are easily made. It will thus be critical to enter the dataset twice to allow subsequent validation. As you will have to validate each of the two related files independently, you must also have a unique identifier not only in the parent but also in the child file to allow validation on that key.

EpiData Analysis

The final result of the analysis will be to obtain the following EpiData Analysis output:

Part 1 with AGGREGATE



Part 2 with transposing values

Overall microscopy result			
of 4 serial smears	Negative	Positive	Total
N—	2	0	2
NN—	1	0	1
NN9P	0	1	1
NP—	0	1	1
NPP—	0	3	3
PNP—	0	1	1
PP—	0	1	1
Total	3	7	10

Patient's sex				
Incremental yield of first 3 smears	Female	% Male	% Total	%
NNP	0 (0.0)	1 (25.0)	1 (14.3)	
NPx	3 (100.0)	1 (25.0)	4 (57.1)	
Px	0 (0.0)	2 (50.0)	2 (28.6)	
Total	3 (100.0)	4 (100.0)	7	

Percents: (Col)

It doesn't really look like much. Nevertheless, quite a few steps are needed to get from the source files D_EX01_PATIENT.REC and D_EX01_EXAMINATION.REC to this point. We will elaborate on some considerations you have to make and offer hints for the sequential components in EpiData Analysis.

Merging the files

In EpiData Entry you made a relation through a unique identifier from the parent to the child file. In EpiData Analysis you must now merge these to files to get the following dataset (first 13 records only shown):

	patid	examid	dateexam	bs	sputum	micro	sex	marital	MergeVar	exam
1	A	A-3-24	24/03/2007	6.3	1	1.0	2	5	3	1
2	A	A-3-25	25/03/2007	7.3	5	0.0	2	5	3	2
3	A	A-3-26	26/03/2007	7.2	2	1.0	2	5	3	3
4	B	B-3-24	24/03/2007	4.9	3	0.0	2	3	3	1
5	C	C-3-24	24/03/2007	5.2	2	0.0	1	7	3	1
6	C	C-3-26	26/03/2007	4.8	3	0.0	1	7	3	2
7	D	D-3-24	24/03/2007	7.3	4	0.2	1	8	3	1
8	D	D-3-25	25/03/2007	7.4	1	2.0	1	8	3	2
9	E	E-3-27	27/03/2007	8.2	9	1.0	2	5	3	1
10	E	E-3-28	28/03/2007	7.9	2	2.0	2	5	3	2
11	F	F-3-27	27/03/2007	7.4	2	0.0	1	4	3	1
12	F	F-3-31	31/03/2007	7.2	3	3.0	1	4	3	2
13	F	F-4-1	01/04/2007	7.7	2	2.0	1	4	3	3

where each record is an observation from the child file, to which the information from the parent file is repetitively added for each observation for the same individual. In other words, you start from the other way around than in EpiData Entry, starting in EpiData Analysis with the child file and using the parent as a lookup table. The grammar of the commands to accomplish this is:

```
read "childfile.rec"
merge parentidentifier /file="parentfile.rec" /table
```

It is important that following merging, the data are sorted first by the individual identifier, and then within that identifier by the date of the visit.

Note: We commonly suggested to code unknown dates as "01/01/1800". With sorting, the unknown date will thus come first (sorting is ascending by default) and this might not be desirable. In this dataset we don't have unknown dates. But only because the small dataset allows us to see that, you would have to anticipate that possibility with a larger dataset and thus first make a new date variable where the unknowns take a value for a date in the future, so they appear with sorting at the end of the information on an individual.

Following the sorting, it might be advantageous to number the visits in order to be able to make a frequency on them to see what the maximum number of visits is (here we can see that it is a maximum of 4 visits, but in a large database, it would be more difficult).

To create a new variable EXAM and set its default initially to 1 (each record takes first the value 1), we have so far always used the following grammar:

```
define exam #
exam=1
```

EpiData Analysis offers a more elegant one-line alternative approach which accomplishes exactly the same thing:

```
gen i exam=1
```

The command GEN replaces DEFINE and "i" stands for an integer field.

Note: the command GEN will produce integer of length of 9. If you need a shorter and fixed field length integer, you must utilize DEFINE.

There are other similar commands:

```
gen f doorheight=1.85
gen d birthdate=date("31/12/1899")
gen s firstname="john"
```

for date fields (d) float (real number) fields (f), and string (text) fields (s). Look it up in the help file (type `gen+F1` in the command line).

Now that you have a new variable EXAM, how can you tell EpiData that it should look at the person (identified by an ID) and number each visit, starting with 1 until the next individual comes, when it must start again with 1. The command is:

```
if id=id[_n-1] then exam=exam[_n-1]+1
```

This looks admittedly complex. Let's thus take it apart. `[_n]` identifies the current record and accordingly `[_n-1]` the immediately preceding record. Let's say, EpiData Analysis has proceeded to record 547 and looks at the ID of this record. It looks whether record 546 had the same ID as record number 547: `if id=id[_n-1]`. If that is the case, then it should take the EXAM number of record 546 and add 1 to it for record 547: `exam=exam[_n-1]+1`. If it is not the case, then the default stays (which we defined as 1) and it moves on to the next record.

As a result, you should get:

	patid	dateexam	exam
1	A	24/03/2007	1
2	A	25/03/2007	2
3	A	26/03/2007	3
4	B	24/03/2007	1
5	C	24/03/2007	1
6	C	26/03/2007	2
7	D	24/03/2007	1
8	D	25/03/2007	2
9	E	27/03/2007	1
10	E	28/03/2007	2

the first variable column showing the identifier, the second the date of the examination, and the third the number of the examination for that individual.

Aggregating data

If we look at the sex (given the field name SEX), date of examination (given the field name DATEEXAM), and blood sugar (given the field name BS):

	patid	sex	dateexam	bs
1	A	2	24/03/2007	6.3
2	A	2	25/03/2007	7.3
3	A	2	26/03/2007	7.2
4	B	2	24/03/2007	4.9
5	C	1	24/03/2007	5.2
6	C	1	26/03/2007	4.8

SEX remains obviously the same (and is thus from the parent file), and is a categorical variable unique to the examined person, while BS varies by examination date and is a continuous variable. If we want to examine blood sugar by sex, there is no need to transpose the blood sugar values from the vertical to the horizontal as EpiData Analysis has inbuilt a tool to aggregate the data. For the individual and its sex we would write:

```
aggregate patid sex
```

and get (for all ten individuals):

	patid	sex
1	A	2
2	B	2
3	C	1
4	D	1
5	E	2
6	F	1
7	G	2
8	H	1
9	I	2
10	K	1

We can expand this command using an option to calculate the MEAN of blood sugar for each individual:

```
aggregate patid sex /mean="bs"
```

and get:

	patid	sex	N	Nbs	MEAbs
1	A	2	3	3	6.9
2	B	2	1	1	4.9
3	C	1	2	2	5.0
4	D	1	2	2	7.4
5	E	2	2	2	8.0
6	F	1	3	3	7.4
7	G	2	3	3	7.2
8	H	1	3	3	6.7
9	I	2	4	4	8.1
10	K	1	1	1	4.5

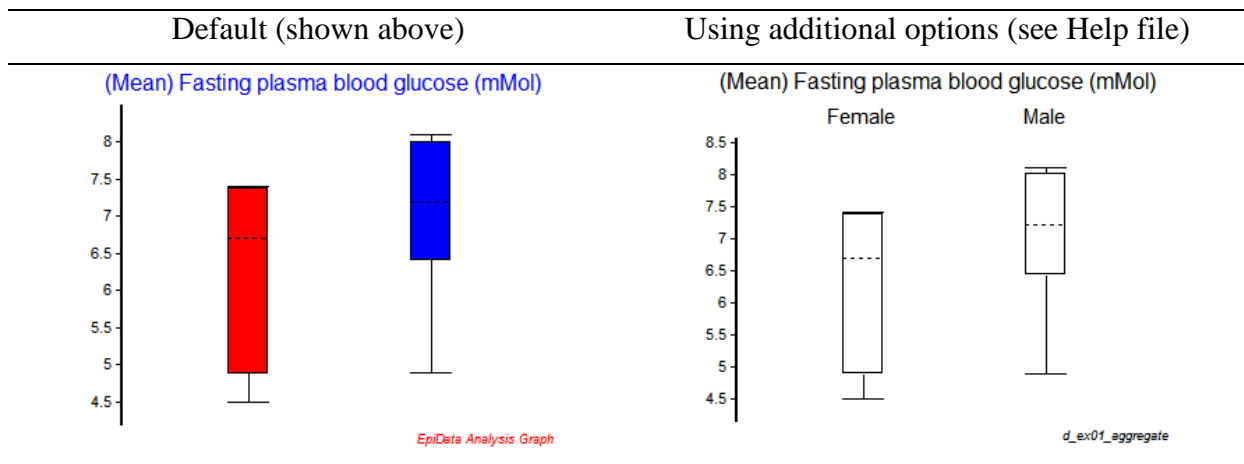
where MEAbs is the calculated mean value of blood sugar for an individual from all the individual's measurements. To save the aggregate data in a file, we add another option:

```
aggregate patid sex /mean="bs" /save="d_ex01_aggregate.rec" /replace
```

Having accomplished this, it is now straight forward to write:

```
cls
close
read "d_ex01_aggregate.rec"
boxplot meabs /by=sex
```

and get:



From long-to-wide

For showing means, there is thus no need to transpose data from the vertical to the horizontal. But it is different for the macroscopic aspect of sputum the examination results. We do not want (nor would we get a sensible result) aggregate sputum smear results. We wish to know each result from each individual.

The first thing before starting copying results from the vertical to the horizontal, we need to know the maximum number of examinations an individual had in the data set. We can get this with a frequency on the EXAM:

```
freq exam
```

Number of examination	
	N
1	10
2	8
3	5
4	1
Total	24

This shows that the maximum number of examinations an individual had in this dataset was 4. We need thus to prepare 4 new variables for each field that are in the sequence of the examination in the vertical but should also become part of each record.

Let's say we have a variable VAR1 with different values for each visit:

ID	VISIT	VAR1
B1	1	1
B1	2	3
B1	3	2
B1	4	2
C1	1	3
D1	1	2

What we need is:

ID	VISIT	VAR1	VAR11	VAR12	VAR13	VAR14
B1	1	1				
B1	2	3				
B1	3	6				
B1	4	2				
C1	1	3				
D1	1	2				

First, we make these 4 variables and give them all the default value of -1 (the minus one is a good way to see missing data and helps later in the selection):

```
gen i var11=-1
gen i var12=-1
gen i var13=-1
gen i var14=-1
```

After these four command lines, our above dataset becomes:

ID	VISIT	VAR1	VAR11	VAR12	VAR13	VAR14
B1	1	1	-1	-1	-1	-1
B1	2	3	-1	-1	-1	-1
B1	3	6	-1	-1	-1	-1
B1	4	2	-1	-1	-1	-1
C1	1	3	-1	-1	-1	-1
D1	1	2	-1	-1	-1	-1

and we are ready to copy the values from the vertical to the horizontal by respecting that the value of VAR1 from VISIT 1 goes to VAR11, the value from VISIT 2 to VAR12, etc.

For VISIT 1, the value for VAR11 is equal to the value of VAR1 and we make it thus the default:

```
VAR11=VAR1
```

Then we use the same approach as above to identify the record:

```
if (id[_n])=(id[_n+1]) then var12=var1[_n+1]
```

This means that if the current record [_n] has the same ID as the next record [_n+1], then VAR12 in the current record should take the value of VAR1 from the next record [_n+1].

Now we do this for all possible 4 records (the maximum of VISITS):

```
if (id[_n])=(id[_n+2]) then var13=var1[_n+2]
if (id[_n])=(id[_n+3]) then var14=var1[_n+3]
```

All the lines to be written for this original field VAR1 are thus:

```
gen i var11=-1
gen i var12=-1
```

```

gen i var13=-1
gen i var14=-1
VAR11=VAR1
if (id[_n])=(id[_n+1]) then var12=var1[_n+1]
if (id[_n])=(id[_n+2]) then var13=var1[_n+2]
if (id[_n])=(id[_n+3]) then var14=var1[_n+3]

```

and we get (assuming that the last patient D1 had only 1 VISIT):

ID	VISIT	VAR1	VAR11	VAR12	VAR13	VAR14
B1	1	1	1	3	6	2
B1	2	3	1	3	6	-1
B1	3	6	1	3	-1	-1
B1	4	2	1	-1	-1	-1
C1	1	3	3	-1	-1	-1
D1	1	2	2	-1	-1	-1

You may note that only for VISIT 1 for patient B1 with four visits all new 4 variables have all respective 4 values from the 4 VISITs.

Now we can safely get rid of the records of VISITs 2, 3, and 4 and we end up just with individuals who have all information from each VISIT:

```
select visit=1
```

and we get:

ID	VISIT	VAR1	VAR11	VAR12	VAR13	VAR14
B1	1	1	1	3	6	2
C1	1	3	3	-1	-1	-1
D1	1	2	2	-1	-1	-1

The conversion from “Long-to-wide” is thus successfully completed, well, for this variable. Of course, before you make this selection, you have to repeat the same approach for each field, so that in the end you get something like:

	patid	sex	marital	dateexam1	dateexam2	dateexam3	dateexam4	sputum1	sputum2	sputum3	sputum4	res1	res2	res3	res4
1	A	2	5	24/03/2007	25/03/2007	26/03/2007	01/01/1900	1	5	2	-1	10	0	10	-1
2	B	2	3	24/03/2007	01/01/1900	01/01/1900	01/01/1900	3	-1	-1	-1	0	-1	-1	-1
3	C	1	7	24/03/2007	26/03/2007	01/01/1900	01/01/1900	2	3	-1	-1	0	0	-1	-1
4	D	1	8	24/03/2007	25/03/2007	01/01/1900	01/01/1900	4	1	-1	-1	0	20	-1	-1
5	E	2	5	27/03/2007	28/03/2007	01/01/1900	01/01/1900	9	2	-1	-1	10	20	-1	-1
6	F	1	4	27/03/2007	31/03/2007	01/04/2007	01/01/1900	2	3	2	-1	0	30	20	-1
7	G	2	2	27/03/2007	28/03/2007	01/04/2007	01/01/1900	5	1	3	-1	0	20	10	-1
8	H	1	5	31/03/2007	01/04/2007	02/04/2007	01/01/1900	1	3	1	-1	0	10	10	-1
9	I	2	7	31/03/2007	01/04/2007	02/04/2007	03/04/2007	5	1	3	1	0	0	90	10
10	K	1	5	02/04/2007	01/01/1900	01/01/1900	01/01/1900	2	-1	-1	-1	0	-1	-1	-1

You have now ten patients and you are at the point where you can continue to work in the same way as you used to work before. While it is much more complex to get to here from 1 line per examination than from 1 line per examinee, it is also obvious that in the end this is much more informative:

For each examination we have the date and for each specimen we have the quality of the sputum. In the Union / WHO approach you have only one date (the date of collection of the first specimen) for a series of three, and the quality of sputum in the Tuberculosis Laboratory is not that informative as it is very possible that every day the quality of the specimen is different, but there is no space allocated to write 3 different ones.

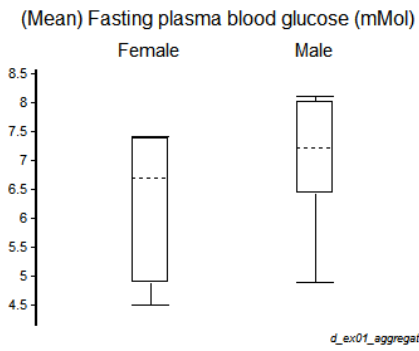
Tasks:

- *Prepare a data documentation sheet*
- *Prepare two EpiData Entry triplets to create a relational database*
- *Enter the data from the following sample data set:*

Patient	Date of exam	Sex	Marital status	Blood sugar	Sputum	Result
A	24-Mar-2007	Male	Married	6.3	Mucoid	1+
B	24-Mar-2007	Male	Divorced	4.9	Muco-purulent	Neg
C	24-Mar-2007	Female	Single	5.2	Purulent	Neg
D	24-Mar-2007	Female	Widowed	7.3	Blood-tinged	2 per 100
A	25-Mar-2007	Male		7.3	Salivary	Neg
D	25-Mar-2007	Female		7.4	Mucoid	2+
A	26-Mar-2007			7.2	Purulent	1+
C	26-Mar-2007	Female		4.8	Muco-purulent	Neg
E	27-Mar-2007	Male	Married	8.2		1+
F	27-Mar-2007	Female	Annulled	7.4	Purulent	Neg
G	27-Mar-2007	Male	Cohabiting	6.9	Salivary	Neg
G	28-Mar-2007	Male		7.2	Mucoid	2+
E	28-Mar-2007	Male		7.9	Purulent	2+
F	31-Mar-2007	Female		7.2	Muco-purulent	3+
H	31-Mar-2007		Married	6.6	Mucoid	Neg
I	31-Mar-2007	Male	Separated	8.3	Salivary	Neg
H	1-Apr-2007	Female		6.9	Muco-purulent	1+
F	1-Apr-2007	Female	Engaged	7.7	Purulent	2+
I	1-Apr-2007	Male	Single	8.0	Mucoid	8 per 100
G	1-Apr-2007	Male		7.6	Muco-purulent	1+
K	2-Apr-2007	Female	Married	4.5	Purulent	Neg
I	2-Apr-2007	Male		8.2	Muco-purulent	
H	2-Apr-2007	Female		6.6	Mucoid	1+
I	3-Apr-2007	Male		8.1	Mucoid	1+

- *Write a program D_EX01.PGM that merges the two files, then prepare sets for the aggregated data and for the “long-to-wide” transformation to produce the following output (obtaining the same numbers is relevant) respectively (see next page):*

From aggregating the data:



From transformation “long-to-wide”:

Overall microscopy result			
of 4 serial smears	Negative	Positive	Total
N—	2	0	2
NN—	1	0	1
NN9P	0	1	1
NP—	0	1	1
NPP—	0	3	3
PNP—	0	1	1
PP—	0	1	1
Total	3	7	10

Patient's sex				
Incremental yield of first 3 smears	Female	% Male	% Total	%
NNP	0 (0.0)	1 (25.0)	1 (14.3)	
NPx	3 (100.0)	1 (25.0)	4 (57.1)	
Px	0 (0.0)	2 (50.0)	2 (28.6)	
Total	3 (100.0)	4 (100.0)	7	

Percent: (Col)