ABSTRACT
Patient registries seek to capture data from broader-based patient populations having a single diagnosis and may lack clinical detail or precision whereas clinical data are drawn from intensive discovery and are limited to select patients related to the study. Clinical data may be used to validate information collected in the patient registry, and patient registry data may complement ongoing clinical studies, such as to provide essential baseline or historical data. A simple and versatile record-linkage is performed between mock registry and clinical datasets that represent data collected in hospitals nationwide. The linkage presumes the lack of common unique patient identifiers and a multi-key approach is considered for defining matches. Composite keys are developed based on shared attributes within the data schemata. Various composite keys are developed consisting of hospital, discharge and arrival dates, age, and gender. Blocking is used to reduce number of non-matches resulting from cartesian product match merges. Finally, exact- and fuzzy-matching techniques are used to link records. Simple methods to deduplicate matches are also proposed. Risks to privacy and confidentiality of subjects are of notable concern. SAS® BASE and SAS DATA STEP on Windows NT SAS version 6.12 are used for the match-merge process.

INTRODUCTION
There are often tradeoffs when designing data collection tools - a process that is driven primarily by scarce resources. Where the Registry dataset seeks to obtain many cases it does so by relaxing the protocols that govern the process of data management and quality and by limiting the quantity of information collected. Where the Clinical dataset seeks to collect a great deal of quality information under managed processes it does so by limiting the total number of records it can populate. Each repository of information is designed and developed independently, each may share a common data schema, and each has its own strengths and weaknesses. A mutually beneficial cross sharing of information between the two repositories can add value to the knowledge of each as well as provide economic and scientific synergies. This paper will outline a simple record-linkage between two mock datasets that do not contain a common primary-key identifier.

PREPARING DATASETS
Before the match-merge process can begin, both datasets must adhere to conventions. Integrating the representation of data is essential for the match-merge process and naming conventions facilitate code development.

INTEGRATION
The datasets are presumed to represent similar patient populations characterized by a single diagnosis, and hospitals from the Clinical and Registry datasets must have a common source identification number. If the Clinical dataset is longitudinal in design, the event record for which the diagnosis is reported should be selected and subsequent visit information should be drawn up into the event record in the Clinical dataset. In the Registry data, baseline data also include a medication. In this example, two heterogeneous Clinical and Registry datasets representing data collected nationwide are presented in Appendix 1 and 2.

All field names should be unique across the Clinical and Registry datasets. The variable types and lengths of the fields on which the match is performed should compare identically. Formats and informats should be removed. Field nomenclature should be the similar. The convention is to prefix the field name with one character that identifies the data source of all fields. Identically suffix the field when fields are shared between the two data sources. Fields can be labeled. In fields that are either binary or logical, e.g., in gender or cpage (Age Check), values should be coded as 1 or 0. Field naming conventions are shown in Figure 1.

FORCING TEMPORAL CONCURRENCE
Reduce the number of records that are not related in regards to time period. Using the Clinical Dataset, the minimum admission date and maximum discharge date are determined for each hospital and then applied to the larger the Registry dataset. From the Registry dataset, records are removed in which non-overlapping discharge or admission dates occur with respect to the Clinical Dataset (see Appendix 3 and 5).

BLOCKING AND REDUCTION
Limiting the match-merge process to geographically identical locations can greatly reduce processing time. Blocking is used to reduce number of non-matches resulting from cartesian product match merges. The prepared Registry and Clinical datasets are split into separate datasets based on the state in which the patient was admitted. This keeps the number of cartesian product match merges to a minimum (see Appendix 6). Any hospitals not shared between the datasets are removed along with their patients (see Appendix 5).

MATCH-MERGE
To match records between the Registry and Clinical datasets, an iterative program module is used to identify up to 5-Key exact or fuzzy matches. In order to make exact or fuzzy comparisons with respect to common fields, one record from the first SAS dataset is held open while it reads all the records of the second database for possible matches to the one record in the first. This process is accomplished by using the POINT option under the SET statement in a datastep (see Appendix 7). When all attributes within the key were the same between Registry and Clinical datasets, the match was considered exact. Variance allowances for select attributes are permitted. Variables, such as Age, can be
permitted an allowance of up to ±1 or ±2 years. Matches on
dates can be fuzzed as well. Records with missing values in any
field represented in the 5 key-ID are removed (see Appendix 1
and 2).

MATCHED FIELDS
Hospital ID, Patient Age, Gender, Admission Date, and
Discharge Date comprise the basic composite key. The
composite key can consist of supplemental fields such as patient
initials or date of birth (DOB), if known. DOB should not be used
in the match but rather can be used to calculate an age in years
of the patient if a self-reported age is not already provided.
The same standard for age calculation should be used on each side.
The FLOOR function used in calculating age of patient best
describes a patient's self-reported age (see Appendix 1 and 2).
The composite keys can be classified into a 5-Key ID (Hospital
ID, Age, Gender, Admit Date, & Discharge) and different
variations within the 5-key such as a 4-key, 3-key, or 2-key ID
(see Appendix 7). From each composite-key match, a separate
dataset is appended to the next with each set coded for the type
of match (see Appendix 8).

DEDUPLICATION
The purpose of deduplication is to identify and remove multiple
records containing the same or similar information as defined by
the chosen composite key. It is possible that duplicate records
can exist in either the Registry or Clinical dataset before the
match-merge process begins. Although the deduplication process
must occur after the match-merge process, it is as done in this
paper (see Appendix 9), deduplication can optionally occur at the
level of each dataset and prior to the match-merge process.
Using the combined method of deduplication is particularly helpful
with large datasets or datasets that lack high standards for data
collection.

SPREADING
During the match-merge process, spreading (replication of same
record) occurs whenever more than one hit occurs in either
dataset (see Appendix 7). Hits are defined as a record in which
the composite-key ID in the first dataset is found in one or more
records base on the same composite-key ID in the second. To
identify replication, records are arbitrarily assigned a unique
record called CSYSID and RSYSID identifier based on _N_
processing at the level of the initial datastep found in Appendix 1
and 2. The unique system identifier is needed for tiebreaking. The
use of unique patient identification numbers may be problematic,
especially when the dataset contains patient IDs that are not
unique. Therefore, the patient identification number is not used
even though they are present in each dataset.

METHODS FOR TIEBREAKING
A simple method based on weighting or scoring algorithms is
used to assign preferential treatment to the match whenever a
replicate group is identified based on repeating values of CSYSID
and RSYSID. Whenever a replicate CSYSID is produced, it
means that a duplicate record exists in the Registry dataset, and
a replicate RSYSID means that a duplicate record exists in the
Clinical dataset. Logical fields that test equalities are produced
processing at the level of the initial datastep found in Appendix 1
and 2. The unique system identifier is needed for tiebreaking. The
use of unique patient identification numbers may be problematic,
especially when the dataset contains patient IDs that are not
unique. Therefore, the patient identification number is not used
even though they are present in each dataset.

PROC SORT is used in combination with FIRST. processing to
position and select records with higher preference in the case
where replicates (spreaders) exist (see Appendix 9).

TESTING AND ANALYSIS
The Clinical dataset can serve to validate the Registry dataset
using a field that was not part of the match but is common to both
datasets. The binary fields of HXDIAB or DEATH can be used in
PROC FREQ using the AGREE option to produce kappa
statistics (See Appendix 9). In addition, baseline covariates
collected in the Registry dataset, e.g. medications and possible
procedures, can be analyzed in SAS/STAT® PROC PHREG (not
shown here) for possible effects on 60-day and 1-year survival
data, which are provided in the matched record from the Clinical
dataset.

DISCUSSION
In Appendix 9, the processing speed is greatly enhanced by using
only the fields that are directly related to the match. The RSYSID
and the CSYSID fields can be used to merge the matched
records back into the Registry or Clinical datasets for
corraboration of the match and accessing other data that are not
directly related to the merge. An error-free "match" is not
 guaranteed. Although the match-merge process is designed to
control matching error, the conclusions to be drawn from any
match should ultimately rely on the quality of each data
repository. Mutually distrusting parties will want to preserve the
integrity, privacy, and confidentiality of each data repository. A
blinded (encrypted) patient ID should be used in lieu of the actual
patient ID provided that a crosswalk file to the originally patient ID
is securely held by each party. Policies regarding the release of
the content of cross-information that occurs between parties
should be governed by policy guidelines that are put in place prior
to data sharing. The data repositories should be isolated and
protected with operating system authentication and other security
protocols and should not be shared with external parties.

CONCLUSION
Two different datasets sharing common fields and patients are
matched. A simple match-merge between two similar datasets is
demonstrated. The better the data source reliability, the more
likely the success and quality of each hit in the match-merge
process. As concern over patient confidentiality grows, date fields
such as admission date, discharge date, birth date and hospital
ID or hospital location are less likely to be available to
researchers in developing unique composite keys for data
integration.

REFERENCES
Patridge, C., "The Fuzzy Feeling SAS Provides: Electronic
Matching of Records without Common Keys", Proceedings of the
Twenty-Second Annual SAS® Users Group International
Conference, 1997; San Diego, California.

Contact Information
Your comments and questions are valued and encouraged.
Contact the author at:
Paul D. Frederick
Ovation Research Group
5910 Evanston Avenue N.
Seattle, WA 98103
Work Phone: (206) 789-4143
Fax: (206) 789-4173
Email: pfrederick@ovation.org

SAS and all other SAS Institute Inc. product or service names are
registered trademarks or marks of SAS Institute Inc. in the
USA and other countries. © indicates USA registration.
APPENDIX 1 (CONTINUED):

Data CLINICAL (Drop=NB1-NB4);
Input
@1 CPID 4. @5 CSTATE $2. @7 CHOSPID 3.
@10 CGENDER $1.
@11 CADMIT_D MMDDY110.
@21 CDISCH_D MMDDY110.
@31 CFIRST $1. @32 CMIDDLE $1. @33 CLAST $1.
@34 CXHDIAB 1.
@35 CDEATH 1. @36 C60DEATH 1. @37 CYRDEATH 1.
@38 CDOB MMDYY110.
@48 COTHER 1.
;
LABEL
CPIID='Patient ID [C]'
CSTATE='State [C]'
CHOSPID='Hospital [C]'
CGENDER='Gender [C]'
CADMIT_D='Admission [C]'
CDISCH_D='Discharge [C]'
CFIRST='First [C]'
CMIDDLE='Middle [C]'
CLAST='Last [C]'
CHXDIAB='History of Diabetes [C]'
CDEATH='Death [In-Hospital] [C]'
C60DEATH='Death [60-Day] [C]'
CYRDEATH='Death [1-Year] [C]'
CDOB='DOB [C]'
COTHER='Therapy [C]'
CAGE='Patient [C]'
CRANDOMC='Random Score [C]'
CNOTBLNK='Completeness Score [C]'
;
FORMAT CADMIT_D CDISCH_D CDOB MMDDYY10.;
CSTATE= FLOOR((CADMIT_D - CDOB)/365.25);
CRANDOMC=ranuni(5234527);

APPENDIX 2 (REGISTRY DATA):

500101011F05/15/1996/29/19966A10010/21/1996100
500201101F05/15/1996/29/19966B20010/21/1996100
500301101F05/16/1996/29/19966C30010/21/1996100
500401101F05/16/1996/29/19966D40010/21/1996100
500501101F05/16/1996/29/19966E50010/21/1996100
500601101F05/16/1996/29/19966F60010/21/1996100
500701101F05/16/1996/29/19966G70010/21/1996100
500801101F05/16/1996/29/19966H80010/21/1996100
500901101F05/16/1996/29/19966I90010/21/1996100
500100101F05/16/1996/29/19966J00010/21/1996100
500110101F05/16/1996/29/19966K10010/21/1996100
500120101F05/16/1996/29/19966L20010/21/1996100
500130101F05/16/1996/29/19966M30010/21/1996100
500140101F05/16/1996/29/19966N40010/21/1996100
500150101F05/16/1996/29/19966O50010/21/1996100
500160101F05/16/1996/29/19966P60010/21/1996100
500170101F05/16/1996/29/19966Q70010/21/1996100
500180101F05/16/1996/29/19966R80010/21/1996100
500190101F05/16/1996/29/19966S90010/21/1996100
500200101F05/16/1996/29/19966T00010/21/1996100
500210101F05/16/1996/29/19966U10010/21/1996100
500220101F05/16/1996/29/19966V20010/21/1996100
500230101F05/16/1996/29/19966W30010/21/1996100
500240101F05/16/1996/29/19966X40010/21/1996100
500250101F05/16/1996/29/19966Y50010/21/1996100
500260101F05/16/1996/29/19966Z60010/21/1996100
500270101F05/16/1996/29/19966A70010/21/1996100
500280101F05/16/1996/29/19966B80010/21/1996100
500300101F05/16/1996/29/19966C90010/21/1996100
500320101F05/16/1996/29/19966D10010/21/1996100
500330101F05/16/1996/29/19966E20010/21/1996100
500340101F05/16/1996/29/19966F30010/21/1996100
500350101F05/16/1996/29/19966G40010/21/1996100
500360101F05/16/1996/29/19966H50010/21/1996100
500370101F05/16/1996/29/19966I60010/21/1996100

APPENDIX 2 (CONTINUED):

Add CLINICAL Data Here
;
Run;
Proc Contents;
Run;

3
APPENDIX 3:

%Macro MM(Dfile, Pfx, ByVar, Var1, VN1, Var2, VN2);
/* Create Dataset */
/* - For Each Hospital */
/* - Produce Max Min Admission Date Ranges */
/* - Produce Max Min Discharge Date Ranges */
/* - Create Dataset for later merge */
*/
Proc Sort Data=&Dfile.; By &ByVar.;
Proc Univariate Data=&Dfile. NoPrint;
var &Var1. &Var2.;
Output
Out=&Pfx.MAXMIN
Min=&Pfx.MIN&VN1. &Pfx.MIN&VN2.;
By &ByVar.;
Run;
%Mend MM;
%MM(CLINICAL, C, CHOSPID, CADMIT_D, AD, CDISCH_D, DD)
%MM(REGISTRY, R, RHOSPID, RADMIT_D, AD, RDISCH_D, DD)

APPENDIX 4:

%Macro INITIALS(DFile, Pfx, DName);
/* Create Dataset */
/* - Write to Log Any 5-Key replicates FYI */
/* - Determine Length of Patient Initials */
/* - Create System Unique Record ID */
*/
Proc Sort Data=&DFile.; By
&Pfx.STATE &Pfx.HOSPID &Pfx.GENDER
&Pfx.ADMIT_D &Pfx.DISCH_D &Pfx.AGE;
Run;
Data &DFile.1;
Length &Pfx.NINITS2 $3.;
Length &Pfx.NINITS3 $3.;
Length &Pfx.LINITS 4;
Set &DFile.;
By
&Pfx.STATE &Pfx.HOSPID &Pfx.GENDER
&Pfx.ADMIT_D &Pfx.DISCH_D &Pfx.AGE;
IF Not (Last.&Pfx.AGE and First.&Pfx.AGE)
then Put &Pfx.PID=
"Possible Replicate Based on
&Pfx.STATE &Pfx.HOSPID &Pfx.GENDER
&Pfx.ADMIT_D &Pfx.DISCH_D &Pfx.AGE"
;
&Pfx.SYSID=_N_;
label &Pfx.SYSID = "System &DName. ID";
If
&Pfx.FIRST Ne ' ' AND
&Pfx.MIDDLE Ne ' ' AND
&Pfx.LAST Ne ' ' Then Do;
&Pfx.NINITS3 = Trim(&Pfx.FIRST) ||
Trim(&Pfx.MIDDLE) ||
Trim(&Pfx.LAST);
&Pfx.NINITS2 = Trim(&Pfx.FIRST) ||
Trim(&Pfx.LAST);
&Pfx.LINITS=Length(&Pfx.NINITS3);
End;
Else
If
&Pfx.FIRST Ne ' ' AND
&Pfx.LAST Ne ' ' Then Do;
&Pfx.NINITS3 = Trim(&Pfx.FIRST) ||
Trim(&Pfx.LAST);
&Pfx.LINITS=Length(&Pfx.NINITS2);
End;
Label &Pfx.LINITS="Length of Initials";
APPENDIX 5:
/* ------------------------------------------ */
/* Datastep                                  */
/* - Apply Date Ranges                       */
/* - From CLINICAL Dataset                   */
/* - To REGISTRY Dataset                     */
/* - Eliminate Hospitals in CLINICAL That     */
/* - are not in REGISTRY                     */
/* - Remove REGISTRY Patients that Fall      */
/* - Outside Min/Max Arrival/Discharge Dates  */
/* - 2 day allowance for Min/Min             */
/* ------------------------------------------ */
Proc Sort Data=WORK.REGISTRY1; By RHOSPID; Run;
Proc Sort Data=WORK.CMAXMIN; By CHOSPID; Run;
Data WORK.REGISTRY2
  (Label='SELECTED PATIENTS & HOSPITALS');
  Merge
  WORK.REGISTRY1(In=A)
  WORK.CMAXMIN(In=B Rename=(CHOSPID=RHOSPID));
  By RHOSPID;
  If A and B;   /* <--REDUCTION */
    If    /* <--FORCING TEMPORAL CONCURRENCE */
      CMAXDD Ne . And CMINAD Ne . And
      RDISCH_D Ne . And RADMIT_D Ne . Then DO;
      IF (RDISCH_D > (CMAXDD+2)) Then DELETE;
      IF (RADMIT_D < (CMINAD-2)) Then DELETE;
    End;
  Run;

APPENDIX 6:
%Macro State(Pfx); /* <--BLOCKING */
/* Continue states [01-50] As Needed */
  If &Pfx.state = '01' Then OUTPUT &Pfx.01;
  ELSE IF &Pfx.state = '02' Then OUTPUT &Pfx.02;
  ELSE IF &Pfx.state = '03' Then OUTPUT &Pfx.03;
  ELSE IF &Pfx.state = '04' Then OUTPUT &Pfx.04;
  ELSE IF &Pfx.state = '05' Then OUTPUT &Pfx.05;
  ELSE IF &Pfx.state = '06' Then OUTPUT &Pfx.06;
  ELSE IF &Pfx.state = '07' Then OUTPUT &Pfx.07;
  ELSE IF &Pfx.state = '08' Then OUTPUT &Pfx.08;
  %Mend State;
/* Continue states [C01-C50] As Needed */
Data C01 C02 C03 C04 C05 C06 C07 C08;
Set WORK.CLINICAL1; %State(C);
Run;
/* Continue states [R01-R50] As Needed */
Data R01 R02 R03 R04 R05 R06 R07 R08;
Set WORK.REGISTRY2; %State(R);
Run;

APPENDIX 7:
%macro outstate(File);
  Data
    aMatchFile bMatchFile cMatchFile
    dMatchFile eMatchFile fMatchFile
    UnMatchFile;
    Set Work.R&file (Keep=RPID RSYSID RSTATE RHOSPID RAGE RGENDER
                      RADMIT_D RDISCH_D RDOB RNINITS2 RNINITS3
                      RLINITS RNOTBLNK RDEATH RANDRSCO RHXDIAB
                      )
    Point=I Nobs=N;
    If i gt n Then Goto Startovr;
    If _Error_ = 1 Then Abort;
    /* ------------------------------------------ */
    /* - Create logic/flag fields                 */
    /* - for cross-field comparison checks [Ck]   */
    /* ------------------------------------------ */
    %Let Lg=%Str(Length=4 label=);
    attrib CPHOSPID &Lg.'Hospital Ck';
    attrib CPSTATE &Lg.'State Ck';
    attrib CPAGE &Lg.'Age Ck';
    attrib FZCPAGE1 &Lg.'Age Ck[FZ 1 Yrs]';
    attrib FZCPAGE2 &Lg.'Age Ck[FZ 2 Yrs]';
    attrib CPGENDER &Lg.'Gender Ck';
    attrib CPADM &Lg.'Arr Date Ck';
    attrib FZCPADM1 &Lg.'Arr Date Ck[FZ 1 Yrs]';
    attrib FZCPADM2 &Lg.'Arr Date Ck[FZ 2 Yrs]';
    attrib CPDIS &Lg.'Disch Date Ck';
    attrib FZCPDIS1 &Lg.'Disch Date Ck[FZ 1 Yrs]';
    attrib FZCPDIS2 &Lg.'Disch Date Ck[FZ 2 Yrs]';
    attrib CPDOB &Lg.'Date of Birth Ck';
    attrib CPINITIALS &Lg.'Patient Initials Ck';
    /* ------------------------------------------ */
    /* - Produce logic/flag fields                */
    /* ------------------------------------------ */
    CPHOSPID=(RHOSPID=CHOSPID); /* <-EXACT */
    CPSTATE=(RSTATE=CSTATE);
    CPAGE=(RAGE=CAGE);
    FZCPAGE1=(Abs(RAGE-CAGE) in (0 1)); /* <-FUZZ */
    FZCPAGE2=(Abs(RAGE-CAGE) in (0 1 2));
    CPGENDER=(CGENDER=RGENDER);
    CPADM=(CAMID=D=RADMIT_D);
    FZCPADM1=(Abs(RADMIT_D-CAMID) in (0 1));
    FZCPADM2=(Abs(RADMIT_D-CAMID) in (0 1 2));
    CPDIS=(CDISCH_D=RDISCH_D);
    FZCPDIS1=(Abs(RDISCH_D-CDISCH_D) in (0 1));
    FZCPDIS2=(Abs(RDISCH_D-CDISCH_D) in (0 1 2));
    CPDOB=((CDOB ne .) and (RDOB ne .))
    and (CDOB=RDOB);
    CPINITIALS=0;
    If RLINITS=3 and CLINITS=3 and trim(RNINITS3)=trim(CNINITS3)
      then CPINITIALS=1; Else
    If RLINITS=2 and CLINITS=2 and trim(RNINITS2)=trim(CNINITS2)
      then CPINITIALS=1;
/* 5 Match Files in decrease order preference */
/* ------------------------------------------ */
if CPHOSPID=1 and CPAGE=1 and CPGENDER=1 and
CPADM=1 and CPDIS=1 then do;
output amatch&file.; end;
else if CPHOSPID=1 and CPAGE=1 and CPGENDER=1
and CPADM=1 and FZCPDIS1=1 then do;
output bmatch&file.; end;
else if CPHOSPID=1 and CPAGE=1 and CPGENDER=1
and FZCPADM1=1 and CPDIS=1 then do;
output cmatch&file.; end;
else if CPHOSPID=1 and CPAGE=1 and CPGENDER=1
and CPADM=1 then do;
output dmatch&file.; end;
else if CPHOSPID=1 and (FZCPADM1=1 or
FZCPDIS1=1) and CPINITIALS=1 then do;
output ematch&file.; end;
else if CPHOSPID=1 and CPDOB=1 then do;
output fmatch&file.; end;
end; else output unmat&File.;

Startovr: If i Gt n Then Goto Getnext; end;
Getnext: If i Gt n Then i=1;
If _Error_ = 1 Then _Error_ = 0;
Run;
%Mend outstate;

APPENDIX 8:
/* Append State-Based in decreased preference */
/* ------------------------------------------ */
%Macro Set(Mtype, MPrefNo);
Data &Mtype. (label="Match Priority: &MPrefNo.");
Length MType 4;
set /* Continue states [01-50] As Needed */
&MType.01 &MType.02 &MType.03
&MType.04 &MType.05 &MType.06 ;
MType=&MPrefNo.;
Run;
%Mend Set;
%Set(aMatch, 1) %Set(bMatch, 2) %Set(cMatch, 3)
%Set(dMatch, 4) %Set(eMatch, 5) %Set(fMatch, 6)
/* Can Set if Wanted: %Set(UnMat, 99) */
/* set Prioritized DataSets*/
Data AllMtch1 (label='Matched Data w/ Replicates');
set Work.aMatch Work.bMatch Work.cMatch
Work.dMatch Work.eMatch Work.fMatch
/* Can Set if Wanted: Work.UnMat */;
Label MTYPE="Priority or Preference of Match";
run;

APPENDIX 9:
/* rectify replicates on REGISTRY side */
/* replicate CLINICAL IDS [CSYSID] */
/* means spreaders on REGISTRY side */
/* ------------------------------------------ */
Proc sort data=WORK.AllMtch1;
by CSYSID;
descending CPHOSPID descending CPAGE
descending CPGENDER descending CPADM
descending CPDIS descending CPDOB
descending CPINITIALS descending RDEATH
descending RNOTBLNK descending RrandSCO;
run;

Data WORK.AllMtch2
(label='No Replicates - REGISTRY side');
Set WORK.AllMtch1;
By CSYSID;
if first.CSYSID;
Run;

Proc print data=WORK.AllMtch2;
Var CSYSID RSYSID CPID RPID CPHOSPID CPAGR
CPGENDER CPADM CPDIS CPDOB CPINITIALS RDEATH
CDEATH RNOTBLNK CNOTBLNK RrandSCO CRandSCO
MType; Run;

Proc sort data=WORK.AllMtch2;
by RSYSID
descending CPHOSPID descending CPAGE
descending CPGENDER descending CPADM
descending CPDIS descending CPDOB
descending CPINITIALS descending CDEATH
descending CNOTBLNK descending CRandSCO ;
run;

Title 'Replicates in the CLINICAL Dataset';
proc print data=WORK.AllMtch1;
Var CSYSID RSYSID CPID RPID CPHOSPID CPAGR
CPGENDER CPADM CPDIS CPDOB CPINITIALS RDEATH
CDEATH RNOTBLNK CNOTBLNK RrandSCO CRandSCO
MType; Run;

Proc sort data=AllMtch2;
by RSYSID
descending CPHOSPID descending CPAGE
descending CPGENDER descending CPADM
descending CPDIS descending CPDOB
descending CPINITIALS descending CDEATH
descending CNOTBLNK descending CRandSCO;
run;

Title 'Matched CLINICAL/REGISTRY File';
proc print data=WORK.AllMtch3;
Var RSYSID CSYSID CPID RPID CPHOSPID CPAGR
CPGENDER CPADM CPDIS CPDOB CPINITIALS CDEATH
CNOTBLNK RrandSCO CRandSCO MType;
run;

Title 'Agreement Between Two Sources of Data';
Proc Freq Data=AllMtch3; tables CHXDIAB*RHXDIAB
CDEATH*RDEATH / Agree;
Run;
Proc Contents Data=WORK.AllMtch3; Run;