CALIFORNIA DEPARTMENT OF JUSTICE

Research Requests

Researcher User Guide

Criminal Records Application Support

Version 3.0

August 31, 2018

Provides examples on extracting and analyzing research request generated CORI data.

Change Log

VERSION	DATE	AUTHOR	COMMENT
1.0	01-25-2017	Bernard Jayawardene	Initial draft.
1.1	03-08-2017	Bernard Jayawardene	Expanded Section 2.
1.2	03-27-2017	Bernard Jayawardene	Added Scenarios to Section 3.
1.3	04-21-2017	Kevin Walker	Expanded Section 2 to include code for SPSS.
1.4	04-25-2017	Bernard Jayawardene	Expanded Section 3 to include code for SAS.
2.0	04-25-2017	Bernard Jayawardene	Sync. document version with software release version.
2.1	06-13-2017	Bernard Jayawardene	Added cycle age.
2.2	02-01-2018	Bernard Jayawardene	Mapping of RACE codes.
3.0	08-30-2018	Bernard Jayawardene	Add new attributes for Oracle, SAS and SPSS code. Update Oracle and SAS results for v3 of sample data.

Table of Contents

1 INTRODUCTION

This section describes the purpose, scope, audience, and overview of this document. It also provides a list of definitions, acronyms and abbreviations used throughout the document as well as any references.

1.1 Purpose

The purpose of this document is to provide examples of analyzing information from a research request data extract. Sample Oracle, SAS, and SPSS code will be provided to support the examples being described.

NOTE: These are simple examples of how one may analyze the given sample data. It is not the only way nor should it be interpreted as the only way to retrieve the expected results. The sample data is not representative of actual arrest data.

1.2 Scope

The examples described in this document are run against a set of sample CORI data. The examples will not use nor expect any other sources of data. All needed data is contained in the sample test file provided.

1.3 Audience

The intended audiences for this document are individuals who are responsible for analyzing generated CORI data.

1.4 Definitions, Acronyms, and Abbreviations

1.4.1 Acronyms

The following Acronyms may be used throughout the document:

ACRONYM	DEFINITION
ACHS	Automated Criminal History System
CJIS	Criminal Justice Information System
CORI	Criminal Offender Record Information
CRAS	Criminal Records Application Support
DOJ	California Department of Justice
PDR	Personal Descriptor Record
PII	Personally Identifiable Information
RR	Research Request
SAS	Statistical Analysis System
SPSS	Statistical Package for the Social Sciences

1.4.1 Definitions

- **Event**: An event is anything that is captured as part of a subject's history. Example: arrests, Court findings, confinements in prison, applications for employment, etc.

Within ACHS, CORI data is further defined by the following:

- **Cycle**: A Cycle holds a collection of related information or Events. Cycles may further identify or define the specific events with multiple Steps.
- **Step**: A Step identifies or defines a specific Event. Example: Arrest/Booking, Court Disposition, Custody, etc. Each Step may contain multiple Counts.
- **Count**: A Count defines detailed information about an Event. Example: three Counts of burglary, one Count of fraud, five disposition Counts, etc.

1.5 References

- Research Request-File Format Specification-v3_0.pdf
  - Describes the Research Request Record structure in detail

- Generated sample CORI data file: RR_Sample_Data-v3-comma_delimited.dat
  - Contains test data generated by the DOJ RR process and used in the examples

- Generated SAS Code to import sample data file: Import RR_Sample_Data-v3_0.sas
  - Contains the complete SAS generated code to import a CORI data file

1.6 Overview of Document

Section 2, Process CORI Data, provides examples in Oracle, SAS and SPSS to load and manipulate the sample CORI data.

Section 3, Scenarios, provides examples of queries that are answered against the sample data.

2 PROCESS CORI DATA

This section will describe the steps to load and manipulate the sample CORI data. Each subsection will give an example of how the step can be carried out using Oracle, SAS and SPSS.

For Oracle, standard oracle tools and SQL will be used to manipulate the data. All versions of Oracle should be able to handle the SQL given; however, the examples were tested on Oracle Database 11g Enterprise Edition Release 11.1.0.7.0.

For SAS, Enterprise Guide (EG) functionality and DATA Steps will be used to manipulate the data. SAS EG version 7.11 HF3 was used to test the examples. The SQL provided for Oracle could also be used within SAS EG as PROC SQL commands with minimal changes if desired.

NOTE: The examples below specify a one-level name for object names in SAS. This will normally default to the WORK library, which is temporary and cleared after every session. If a permanent copy is desired, please use a "libname" command, "options user=" command, a two-level name, or some other mechanism to point to a permanent library.

For SPSS, syntax will be used to import and manipulate the data. IBM SPSS Statistics 23.0 was used to test the examples.

2.1 Create Staging Table

2.1.1 Oracle

```
CREATE TABLE rr_stage
(
  record_id            VARCHAR2(12) NOT NULL,
  subject_status       VARCHAR2(2),
  subject_id           VARCHAR2(12) NOT NULL,
  req_seg_sep          VARCHAR2(1),
  req_cii_number       VARCHAR2(10),  req_name             VARCHAR2(30), req_gender          VARCHAR2(1),
  req_dob              VARCHAR2(8),   req_cdl              VARCHAR2(8),  req_ssn             VARCHAR2(9),
  pii_seg_sep          VARCHAR2(1),
  cii_number           VARCHAR2(10),  pri_name             VARCHAR2(30), gender              VARCHAR2(1),
  pri_dob              VARCHAR2(8),   pri_ssn              VARCHAR2(9),  pri_cdl             VARCHAR2(8),
  pri_idn              VARCHAR2(8),   pri_inn              VARCHAR2(8),  fbi_number          VARCHAR2(9),
  pdr_seg_sep          VARCHAR2(1),
  race_code            VARCHAR2(1),   race_descr           VARCHAR2(15),
  eye_color_code       VARCHAR2(3),   eye_color_descr      VARCHAR2(30),
  hair_color_code      VARCHAR2(3),   hair_color_descr     VARCHAR2(30),
  height               VARCHAR2(3),   weight               VARCHAR2(3),
  single_source        VARCHAR2(1),   multi_source         VARCHAR2(1),
  pob_code             VARCHAR2(2),   pob_name             VARCHAR2(30), pob_type            VARCHAR2(8),
  citizenship_list     VARCHAR2(30),
  cyc_seg_sep          VARCHAR2(1),
  cyc_order            VARCHAR2(12),  cyc_date             VARCHAR2(8),
  stp_seg_sep          VARCHAR2(1),
  stp_order            VARCHAR2(12),  stp_event_date       VARCHAR2(8),
  stp_type_code        VARCHAR2(2),   stp_type_descr       VARCHAR2(30),
  stp_ori_type         VARCHAR2(1),   stp_ori_type_descr   VARCHAR2(25),
  stp_ori_code         VARCHAR2(9),   stp_ori_descr        VARCHAR2(25),
  stp_ori_cnty_code    VARCHAR2(2),   stp_ori_cnty_name    VARCHAR2(30),
  cnt_seg_sep          VARCHAR2(1),
  cnt_order            VARCHAR2(12),  disp_date            VARCHAR2(8),  ofn                 VARCHAR2(20),
  offense_code         VARCHAR2(5),   offense_descr        VARCHAR2(50), offense_toc         VARCHAR2(1),
  offense_qual_lst     VARCHAR2(30),
  disp_offense_code    VARCHAR2(5),   disp_offense_descr   VARCHAR2(50),
  disp_offense_toc     VARCHAR2(1),   disp_offense_qual_lst VARCHAR2(30),
  conv_offense_order   VARCHAR2(12),  conv_offense_code    VARCHAR2(5),  conv_offense_descr VARCHAR2(50),
  conv_offense_toc     VARCHAR2(1),   conv_offense_qual_lst VARCHAR2(30),
  fe_num_order         VARCHAR2(12),  fe_num_arr_agy       VARCHAR2(20), fe_num_bnch_warr   VARCHAR2(20),
```

```
  fe_num_cite          VARCHAR2(20),  fe_num_docket        VARCHAR2(20), fe_num_incident    VARCHAR2(20),
  fe_num_booking       VARCHAR2(20),  fe_num_number        VARCHAR2(20), fe_num_remand      VARCHAR2(20),
  fe_num_oos_inn       VARCHAR2(20),  fe_num_crt_case      VARCHAR2(20), fe_num_warrant     VARCHAR2(20),
  disp_order           VARCHAR2(12),  disp_code            VARCHAR2(4),  disp_descr         VARCHAR2(40),
  conv_stat_code       VARCHAR2(1),   conv_stat_descr      VARCHAR2(20),
  sent_seg_sep         VARCHAR2(1),
  sent_order           VARCHAR2(12),  sent_loc_code        VARCHAR2(2),  sent_loc_descr     VARCHAR2(15),
  sent_length          VARCHAR2(3),   sent_time_code       VARCHAR2(1),  sent_time_descr    VARCHAR2(15),
  cyc_age              VARCHAR2(3),   cii_type             VARCHAR2(1),  cii_type_alpha     VARCHAR2(1),
  comment_text         VARCHAR2(150), end_of_rec           VARCHAR2(1)
);
```

2.1.2 SAS

This step is not necessary for SAS. The equivalent table will be created when the sample data is loaded in the next section.

2.1.3 SPSS

This step is not necessary for SPSS. The equivalent file will be created when the sample data is imported in the next section.

2.2 Load Staging Table

2.2.1 Oracle

SQL*Loader is available in all versions of Oracle. It will be used to load the sample data into the staging table.

Please note the following items regarding the script below:

- The script, as written, is to load a comma delimited input file
- Appends data to staging table (to allow for loading of multiple files using same script)
- The table to be loaded (RR_STAGE) is hard coded in the script
- The INFILE, BADFILE and DISCARDFILE are hard coded to the example input data file
    - The above options may be overridden if the script is being used to load requesting agency data files
- First row of the first generated delimited file is a header row. DISCARDS are set to 1 to allow script to ignore the header row

```
OPTIONS (SILENT=(HEADER, FEEDBACK))
LOAD DATA
INFILE 'RR_Sample_Data-v3-comma_delimited.dat'
BADFILE 'RR_Sample_Data-v3-comma_delimited.bad'
DISCARDFILE 'RR_Sample_Data-v3-comma_delimited.dsc'
DISCARDS 1
INTO TABLE "RR_STAGE"
APPEND
WHEN (RECORD_ID != "RECORD_ID")
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"' TRAILING NULLCOLS
(
  RECORD_ID, SUBJECT_STATUS, SUBJECT_ID,
  REQ_SEG_SEP,
  REQ_CII_NUMBER, REQ_NAME, REQ_GENDER, REQ_DOB, REQ_CDL, REQ_SSN,
  PII_SEG_SEP,
  CII_NUMBER, PRI_NAME, GENDER, PRI_DOB, PRI_SSN, PRI_CDL, PRI_IDN, PRI_INN, FBI_NUMBER,
  PDR_SEG_SEP,
  RACE_CODE, RACE_DESCR, EYE_COLOR_CODE, EYE_COLOR_DESCR, HAIR_COLOR_CODE, HAIR_COLOR_DESCR,
  HEIGHT, WEIGHT, SINGLE_SOURCE, MULTI_SOURCE, POB_CODE, POB_NAME, POB_TYPE, CITIZENSHIP_LIST,
  CYC_SEG_SEP,
```

```
  CYC_ORDER, CYC_DATE,
  STP_SEG_SEP,
  STP_ORDER, STP_EVENT_DATE, STP_TYPE_CODE, STP_TYPE_DESCR,
  STP_ORI_TYPE, STP_ORI_TYPE_DESCR, STP_ORI_CODE, STP_ORI_DESCR,
  STP_ORI_CNTY_CODE, STP_ORI_CNTY_NAME,
  CNT_SEG_SEP,
  CNT_ORDER, DISP_DATE, OFN, OFFENSE_CODE, OFFENSE_DESCR, OFFENSE_TOC, OFFENSE_QUAL_LST,
  DISP_OFFENSE_CODE, DISP_OFFENSE_DESCR, DISP_OFFENSE_TOC, DISP_OFFENSE_QUAL_LST,
  CONV_OFFENSE_ORDER, CONV_OFFENSE_CODE, CONV_OFFENSE_DESCR,
  CONV_OFFENSE_TOC, CONV_OFFENSE_QUAL_LST,
  FE_NUM_ORDER, FE_NUM_ARR_AGY, FE_NUM_BNCH_WARR, FE_NUM_CITE, FE_NUM_DOCKET,
  FE_NUM_INCIDENT, FE_NUM_BOOKING, FE_NUM_NUMBER, FE_NUM_REMAND, FE_NUM_OOS_INN,
  FE_NUM_CRT_CASE, FE_NUM_WARRANT,
  DISP_ORDER, DISP_CODE, DISP_DESCR, CONV_STAT_CODE, CONV_STAT_DESCR,
  SENT_SEG_SEP,
  SENT_ORDER, SENT_LOC_CODE, SENT_LOC_DESCR, SENT_LENGTH, SENT_TIME_CODE, SENT_TIME_DESCR,
  CYC_AGE, CII_TYPE, CII_TYPE_ALPHA, COMMENT_TEXT,
  END_OF_REC
)

Table "RR_STAGE":
  31374 Rows successfully loaded.
  0 Rows not loaded due to data errors.
  1 Row not loaded because all WHEN clauses were failed.
  0 Rows not loaded because all fields were null.
```

2.2.2 SAS

```
/* ---------------------------------------
Code exported from SAS Enterprise Guide
DATE: Friday, August 31, 2018     TIME: 12:29:32 PM
PROJECT: Import_sample_data-v3
--------------------------------------- */

%LET _CLIENTTASKLABEL='Import Data (RR_Sample_Data-v3-comma_delimited.dat)';
%LET _CLIENTPROJECTPATH='';
%LET _CLIENTPROJECTNAME='Import_sample_data-v3.egp';

GOPTIONS ACCESSIBLE;
/* ---------------------------------------------------------------------
   Code generated by a SAS task

   Generated on Friday, August 31, 2018 at 12:28:16 PM
   By task:     Import Data Wizard

   Source file: RR_Sample_Data-v3-comma_delimited.dat
   Server:      Local File System

   Output data: LIBRR.RR_Sample_Data_v3
   Server:      ******

   Note: In preparation for running the following code, the Import Data wizard has used
   internal routines to transfer the source data file from the local file system to ******.
   There is no SAS code available to represent this action.
   --------------------------------------------------------------------- */

NOTE: Code below edited to show first 3 and last 3 attributes in a given section.
      For a complete listing please see sample code file.
DATA LIBRR.RR_Sample_Data_v3;
    LENGTH
        RECORD_ID         $ 12        SUBJECT_STATUS    $ 2         SUBJECT_ID        $ 12
. . .
        CII_TYPE_ALPHA    $ 1         COMMENT_TEXT      $ 1         END_OF_REC        $ 1 ;
    DROP
        F96 ;
    FORMAT
        RECORD_ID         $CHAR12.        SUBJECT_STATUS    $CHAR2.         SUBJECT_ID        $CHAR12.
. . .
```

```
        CII_TYPE_ALPHA    $CHAR1.          COMMENT_TEXT      $CHAR1.          END_OF_REC        $CHAR1. ;
    INFORMAT
        RECORD_ID         $CHAR12.         SUBJECT_STATUS    $CHAR2.          SUBJECT_ID        $CHAR12.
. . .
        CII_TYPE_ALPHA    $CHAR1.          COMMENT_TEXT      $CHAR1.          END_OF_REC        $CHAR1. ;
    INFILE '/opt/saswork/---/#LN00017'
        LRECL=32767
        FIRSTOBS=2
        ENCODING="LATIN1"
        DLM='2c'x
        MISSOVER
        DSD ;
    INPUT
        RECORD_ID         : $CHAR12.       SUBJECT_STATUS    : $CHAR2.          SUBJECT_ID        : $CHAR12.
. . .
        CII_TYPE_ALPHA    : $CHAR1.        COMMENT_TEXT      : $CHAR1.          END_OF_REC        : $CHAR1.
        F96               : $1. ;
RUN;

NOTE: The infile '/opt/saswork/---/#LN00017' is:
      Filename=/opt/saswork/---/#LN00017,
      Last Modified=30Aug2018:17:34:14,
      File Size (bytes)=14654303

NOTE: 31374 records were read from the infile '/opt/saswork/---/#LN00017'.
NOTE: The data set LIBRR.RR_SAMPLE_DATA_V3 has 31374 observations and 95 variables.
```

```
** Create rr_stage table with a corresponding index;
Data rr_stage(index=(rr_stage_ix=(record_id subject_id cyc_order stp_order cnt_order sent_order)));
  Set RR_Sample_Data_comma_delimited;
Run;

NOTE: There were 31374 observations read from the data set LIBRR.RR_SAMPLE_DATA_V3.
NOTE: The data set LIBRR.RR_STAGE has 31374 observations and 95 variables.
```

2.2.3 SPSS

```
GET DATA  /TYPE=TXT
  /FILE= */ 'contents removed - file name/path generated by SPSS and is unique to researcher' /*
  /ENCODING='UTF8'
  /DELCASE=LINE
  /DELIMITERS=","
  /QUALIFIER='"'
  /ARRANGEMENT=DELIMITED
  /FIRSTCASE=2
  /IMPORTCASE=ALL
  /VARIABLES=
  RECORD_ID A12  SUBJECT_STATUS A2  SUBJECT_ID A12  REQ_SEG_SEP A1  REQ_CII_NUMBER A10  REQ_NAME A30
  REQ_GENDER A1  REQ_DOB F8.0  REQ_CDL A8  REQ_SSN A9  PII_SEG_SEP A1  CII_NUMBER A10  PRI_NAME A30
  GENDER A1  PRI_DOB F8.0  PRI_SSN A9  PRI_CDL A8  PRI_IDN A8  PRI_INN A8  FBI_NUMBER A9  PDR_SEG_SEP A1
  RACE_CODE A1  RACE_DESCR A15  EYE_COLOR_CODE A3  EYE_COLOR_DESCR A30  HAIR_COLOR_CODE A3
  HAIR_COLOR_DESCR A30  HEIGHT F3.0  WEIGHT F3.0  SINGLE_SOURCE F1.0  MULTI_SOURCE F1.0  POB_CODE A2
  POB_NAME A30  POB_TYPE A8  CITIZENSHIP_LIST A25  CYC_SEG_SEP A1  CYC_ORDER F12.0  CYC_DATE F8.0
  STP_SEG_SEP A1  STP_ORDER F12.0  STP_EVENT_DATE F8.0  STP_TYPE_CODE A2  STP_TYPE_DESCR A30
  STP_ORI_TYPE A1  STP_ORI_TYPE_DESCR A25  STP_ORI_CODE A9  STP_ORI_DESCR A25  STP_ORI_CNTY_CODE A2
  STP_ORI_CNTY_NAME A30  CNT_SEG_SEP A1  CNT_ORDER F12.0  DISP_DATE F8.0  OFN A20  OFFENSE_CODE F5.0
  OFFENSE_DESCR A50  OFFENSE_TOC A1  OFFENSE_QUAL_LST A30  DISP_OFFENSE_CODE F1.0  DISP_OFFENSE_DESCR A40
  DISP_OFFENSE_TOC A1  DISP_OFFENSE_QUAL_LST F1.0  CONV_OFFENSE_ORDER F12.0  CONV_OFFENSE_CODE F5.0
  CONV_OFFENSE_DESCR A14  CONV_OFFENSE_TOC A1  CONV_OFFENSE_QUAL_LST A2  FE_NUM_ORDER F12.0
  FE_NUM_ARR_AGY F1.0  FE_NUM_BNCH_WARR F1.0  FE_NUM_CITE F1.0  FE_NUM_DOCKET F1.0  FE_NUM_INCIDENT F1.0
  FE_NUM_BOOKING F1.0  FE_NUM_NUMBER F5.0  FE_NUM_REMAND F1.0  FE_NUM_OOS_INN F1.0  FE_NUM_CRT_CASE F1.0
  FE_NUM_WARRANT F1.0  DISP_ORDER F12.0  DISP_CODE F4.0  DISP_DESCR A39  CONV_STAT_CODE F1.0
  CONV_STAT_DESCR A20  SENT_SEG_SEP A1  SENT_ORDER F12.0  SENT_LOC_CODE A2  SENT_LOC_DESCR A15
  SENT_LENGTH F3.0  SENT_TIME_CODE A1  SENT_TIME_DESCR A15  CYC_AGE A3
  CII_TYPE A1  CII_TYPE_ALPHA A1  COMMENT_TEXT A150  END_OF_REC A1.
CACHE.
EXECUTE.
DATASET NAME rr_stage WINDOW=FRONT.

** Sort imported rr_stage data set.
** This is needed to minimize potential data order issues in future steps.
SORT CASES BY RECORD_ID(A) CYC_ORDER(A) STP_ORDER(A) CNT_ORDER(A).
```

2.3 Extract Subject Data

This section will extract all the attributes associated with a subject from the data in the staging table. The extracted attributes are members of the following segments:

- **Subject Record Segment**: Unique non-PII based identifier for subject and other record related information
- **Subject Request Segment**: Original requested subject information used to match a subject in ACHS
- **Subject PII Segment**: PII data for subject
- **Subject PDR Segment**: Non-PII data for subject

Please refer to section "2.2 Research Request Record Segments" in the "Research Requests – File Format Specification" document for a detailed description of the above segments.

2.3.1 Oracle

Two methods are shown under Oracle. Depending on size of data (number of subjects or number of rows) and available resources it is recommended that tables be created to hold the separate segments not just the views.

Creating tables and the associated indexes will improve performance when querying large sets of data.

```
CREATE OR REPLACE VIEW rr_subject_v AS
  SELECT DISTINCT record_id, subject_status, subject_id, req_cii_number, req_name, req_gender, req_dob,
                  req_cdl, req_ssn, cii_number, cii_type, cii_type_alpha, pri_name, gender, pri_dob,
                  pri_ssn, pri_cdl, pri_idn, pri_inn, fbi_number, race_code, race_descr,
                  eye_color_code, eye_color_descr, hair_color_code, hair_color_descr, height, weight,
                  single_source, multi_source, pob_code, pob_name, pob_type, citizenship_list
    FROM rr_stage;

SELECT Count(*) tot_rows FROM rr_stage;

SELECT Count(*) tot_unique_subjects FROM rr_subject_v;
```

```
SELECT Count(*) tot_valid_unique_subjects FROM rr_subject_v where subject_status = 'F';

-- for performance improvements it is recommended that the following items be created
-- this will be significant when joining across multiple tables
CREATE TABLE rr_subject AS SELECT * FROM rr_subject_v;

CREATE UNIQUE INDEX rr_subject_pk ON rr_subject(record_id, subject_id);

ALTER TABLE rr_subject
  ADD (CONSTRAINT rr_subject_pk PRIMARY KEY (record_id, subject_id) USING INDEX rr_subject_pk);

-- Results of above Select count(*) statements

  TOT_ROWS
----------
     31374

TOT_UNIQUE_SUBJECTS
-------------------
               3658

TOT_VALID_UNIQUE_SUBJECTS
-------------------------
                    3559
```

2.3.2 SAS

```
** Create unique set of subjects from staging data set;
Data rr_subject;
      Set rr_stage (keep= record_id subject_status subject_id req_cii_number req_name req_gender req_dob
            req_cdl req_ssn pii_seg_sep cii_number cii_type cii_type_alpha pri_name gender pri_dob
            pri_ssn pri_cdl pri_idn pri_inn fbi_number race_code race_descr eye_color_code eye_color_descr
            hair_color_code hair_color_descr height weight single_source multi_source pob_code
            pob_name pob_type citizenship_list);
      by record_id subject_id;


      If first.record_id and first.subject_id; /* use first.* to get distinct rows */
Run;

NOTE: There were 31374 observations read from the data set LIBRR.RR_STAGE.
NOTE: The data set LIBRR.RR_SUBJECT has 3658 observations and 35 variables.
```

2.3.3 SPSS

```
*Create unique set of subjects from staging data set.
DATASET ACTIVATE rr_stage.
SAVE OUTFILE= 'rr_subject.sav'
    /KEEP
  RECORD_ID  SUBJECT_STATUS  SUBJECT_ID
  REQ_CII_NUMBER  REQ_NAME  REQ_GENDER  REQ_DOB  REQ_CDL  REQ_SSN
  CII_NUMBER  CII_TYPE  CII_TYPE_ALPHA  PRI_NAME  GENDER  PRI_DOB  PRI_SSN  PRI_CDL  PRI_IDN  PRI_INN
  FBI_NUMBER  RACE_CODE  RACE_DESCR  EYE_COLOR_CODE  EYE_COLOR_DESCR  HAIR_COLOR_CODE  HAIR_COLOR_DESCR
  HEIGHT WEIGHT  SINGLE_SOURCE  MULTI_SOURCE  POB_CODE  POB_NAME  POB_TYPE  CITIZENSHIP_LIST.

GET FILE = 'rr_subject.sav'.

*Identify Duplicate Cases.
SORT CASES BY RECORD_ID(A) SUBJECT_ID(A).
MATCH FILES
  /FILE=*
  /BY RECORD_ID SUBJECT_ID
  /FIRST=PrimaryFirst.

VARIABLE LABELS  PrimaryFirst 'Indicator of each first matching case as Primary'.
VALUE LABELS  PrimaryFirst 0 'Duplicate Case' 1 'Primary Case'.
VARIABLE LEVEL  PrimaryFirst (ORDINAL).
FREQUENCIES VARIABLES=PrimaryFirst.
SELECT IF PrimaryFirst = 1.
FREQUENCIES VARIABLES=PrimaryFirst.
DELETE VARIABLES PrimaryFirst.
```

2.4 Extract Cycle Data

This section will extract all the attributes associated with a Cycle and the attributes to allow joining to the associated subject. The extracted attributes are members of the following segments:

- **Subject Record Segment**: Attributes to uniquely identify a subject
- **Cycle Segment**: Cycle information for subject

2.4.1 Oracle

```
CREATE OR REPLACE VIEW rr_cycle_v AS
  SELECT DISTINCT record_id, subject_id, cyc_order, cyc_date, cyc_age
    FROM rr_stage
   WHERE cyc_order IS NOT NULL;

SELECT Count(*) tot_unique_cycles FROM rr_cycle_v;

-- for performance improvements it is recommended that the following items be created
-- this will be significant when joining across multiple tables
CREATE TABLE rr_cycle AS SELECT * FROM rr_cycle_v;

CREATE UNIQUE INDEX rr_cycle_pk ON rr_cycle(record_id, cyc_order);

ALTER TABLE rr_cycle
  ADD (CONSTRAINT rr_cycle_pk PRIMARY KEY (record_id, cyc_order) USING INDEX rr_cycle_pk);

-- Results of above Select count(*) statement

TOT_UNIQUE_CYCLES
-----------------
             9637
```

2.4.2 SAS

```
** Create unique set of cycles from staging data set;
Data rr_cycle;
      Set rr_stage (keep= record_id subject_id cyc_order cyc_date cyc_age);
      by record_id subject_id cyc_order;

      ** eliminate cycles which do not exist for subjects that are not found;
      If missing(cyc_order) then delete;

      /* use first.* to get distinct rows but only if cycle exist */
      If first.cyc_order;
Run;

NOTE: There were 31374 observations read from the data set LIBRR.RR_STAGE.
NOTE: The data set LIBRR.RR_CYCLE has 9637 observations and 5 variables.
```

2.4.3 SPSS

```
*Create unique set of cycles from staging data set.
DATASET ACTIVATE rr_stage.
SAVE OUTFILE= 'rr_cycle.sav'
    /KEEP
  RECORD_ID  SUBJECT_ID  CYC_ORDER  CYC_DATE  CYC_AGE.

GET FILE = 'rr_cycle.sav'.

SORT CASES BY RECORD_ID(A) SUBJECT_ID(A) CYC_ORDER(A).

*Eliminate cycles which do not exist for subjects that are not found.
SELECT IF NOT (SYSMIS(CYC_ORDER)).

*Get distinct rows but only if cycle exists.
SORT CASES BY RECORD_ID(A) SUBJECT_ID(A) CYC_ORDER(A).
MATCH FILES
  /FILE=*
  /BY RECORD_ID SUBJECT_ID CYC_ORDER
  /FIRST=PrimaryFirst.

VARIABLE LABELS  PrimaryFirst 'Indicator of each first matching case as Primary'.
VALUE LABELS  PrimaryFirst 0 'Duplicate Case' 1 'Primary Case'.
VARIABLE LEVEL  PrimaryFirst (ORDINAL).
FREQUENCIES VARIABLES=PrimaryFirst.
SELECT IF PrimaryFirst = 1.
FREQUENCIES VARIABLES=PrimaryFirst
  /ORDER=ANALYSIS.
DELETE VARIABLES PrimaryFirst.
```

2.5 Extract Step Data

This section will extract all the attributes associated with a Step and the attributes to allow joining to the associated cycle and subject. The extracted attributes are members of the following segments:

- **Subject Record Segment**: Attributes to uniquely identify a subject
- **Cycle Segment**: Cycle order
- **Step Segment**: Step information for cycles

2.5.1 Oracle

```
CREATE OR REPLACE VIEW rr_step_v AS
  SELECT DISTINCT record_id, subject_id, cyc_order, stp_order, stp_event_date, stp_type_code,
                  stp_type_descr, stp_ori_type, stp_ori_type_descr, stp_ori_code, stp_ori_descr,
                  stp_ori_cnty_code, stp_ori_cnty_name
    FROM rr_stage
   WHERE stp_order IS NOT NULL;

SELECT Count(*) tot_unique_steps FROM rr_step_v;

-- for performance improvements it is recommended that the following items be created
-- this will be significant when joining across multiple tables
CREATE TABLE rr_step AS SELECT * FROM rr_step_v;

CREATE UNIQUE INDEX rr_step_pk ON rr_step(record_id, stp_order);

ALTER TABLE rr_step
  ADD (CONSTRAINT rr_step_pk PRIMARY KEY (record_id, stp_order) USING INDEX rr_step_pk);

-- Results of above Select count(*) statement

TOT_UNIQUE_STEPS
----------------
           16680
```

2.5.2 SAS

```
** Create unique set of steps from staging data set;
Data rr_step;
      Set rr_stage (keep= record_id subject_id cyc_order stp_order stp_event_date stp_type_code
        stp_type_descr stp_ori_type stp_ori_type_descr stp_ori_code stp_ori_descr
        stp_ori_cnty_code stp_ori_cnty_name);
      by record_id subject_id cyc_order stp_order;

      ** eliminate steps which do not exist for subjects that are not found;
      If missing(stp_order) then delete;

      /* use first.* to get distinct rows but only if step exist */
      If first.stp_order;
Run;

NOTE: There were 31374 observations read from the data set LIBRR.RR_STAGE.
NOTE: The data set LIBRR.RR_STEP has 16680 observations and 13 variables.
```

2.5.3 SPSS

```
*Create unique set of steps from staging data set.
DATASET ACTIVATE rr_stage.
SAVE OUTFILE= 'rr_step.sav'
    /KEEP
  RECORD_ID  SUBJECT_ID  CYC_ORDER
  STP_ORDER  STP_EVENT_DATE  STP_TYPE_CODE  STP_TYPE_DESCR  STP_ORI_TYPE  STP_ORI_TYPE_DESCR
  STP_ORI_CODE  STP_ORI_DESCR  STP_ORI_CNTY_CODE  STP_ORI_CNTY_NAME.

GET FILE = 'rr_step.sav'.

SORT CASES BY RECORD_ID(A) SUBJECT_ID(A) CYC_ORDER(A) STP_ORDER(A).

*Eliminate cycles which do not exist for subjects that are not found.
SELECT IF NOT (SYSMIS(STP_ORDER)).

*Get distinct rows, but only if step exists.
SORT CASES BY RECORD_ID(A) SUBJECT_ID(A) CYC_ORDER(A) STP_ORDER(A).
MATCH FILES
  /FILE=*
  /BY RECORD_ID SUBJECT_ID CYC_ORDER STP_ORDER
  /FIRST=PrimaryFirst.

VARIABLE LABELS  PrimaryFirst 'Indicator of each first matching case as Primary'.
VALUE LABELS  PrimaryFirst 0 'Duplicate Case' 1 'Primary Case'.
VARIABLE LEVEL  PrimaryFirst (ORDINAL).
FREQUENCIES VARIABLES=PrimaryFirst.
SELECT IF PrimaryFirst = 1.
FREQUENCIES VARIABLES=PrimaryFirst
  /ORDER=ANALYSIS.
DELETE VARIABLES PrimaryFirst.
```

2.6 Extract Count Data

This section will extract all the attributes associated with Counts and the attributes to allow joining to the associated cycle, step and subject. The extracted attributes are members of the following segments:

- **Subject Record Segment**: Attributes to uniquely identify a subject
- **Cycle Segment**: Cycle order
- **Step Segment**: Step order
- **Count Segment**: Count information for steps
    - Convicted Offenses, Field Events and Dispositions will be captured into separate tables/libraries

2.6.1 Oracle

```
CREATE OR REPLACE VIEW rr_count_v AS
  SELECT DISTINCT record_id, subject_id, cyc_order, stp_order, cnt_order, disp_date, ofn, offense_code,
                  offense_descr, offense_toc, offense_qual_lst, disp_offense_code, disp_offense_descr,
                  disp_offense_toc, disp_offense_qual_lst, comment_text
    FROM rr_stage
   WHERE cnt_order IS NOT NULL;

SELECT Count(*) tot_unique_counts FROM rr_count_v;

-- ============================================

CREATE OR REPLACE VIEW rr_count_conv_offense_v AS
  SELECT DISTINCT record_id, subject_id, cyc_order, stp_order, cnt_order, conv_offense_order,
                  conv_offense_code, conv_offense_descr, conv_offense_toc, conv_offense_qual_lst
    FROM rr_stage
   WHERE conv_offense_order IS NOT NULL;

SELECT Count(*) tot_unique_conv_offenses FROM rr_count_conv_offense_v;

-- ============================================

CREATE OR REPLACE VIEW rr_count_fe_v AS
  SELECT DISTINCT record_id, subject_id, cyc_order, stp_order, cnt_order, fe_num_order, fe_num_arr_agy,
```

```
                    fe_num_bnch_warr, fe_num_cite, fe_num_docket, fe_num_incident, fe_num_booking,
                    fe_num_number, fe_num_remand, fe_num_oos_inn, fe_num_crt_case, fe_num_warrant
    FROM rr_stage
   WHERE fe_num_order IS NOT NULL;

SELECT Count(*) tot_unique_fe FROM rr_count_fe_v;

-- ================================================

CREATE OR REPLACE VIEW rr_count_disp_v AS
  SELECT DISTINCT record_id, subject_id, cyc_order, stp_order, cnt_order,
                  disp_order, disp_code, disp_descr, conv_stat_code, conv_stat_descr
    FROM rr_stage
   WHERE disp_order IS NOT NULL;

SELECT Count(*) tot_unique_disp FROM rr_count_disp_v;

-- ================================================

-- for performance improvements it is recommended that the following items be created
-- this will be significant when joining across multiple tables
CREATE TABLE rr_count AS SELECT * FROM rr_count_v;

CREATE UNIQUE INDEX rr_count_pk ON rr_count(record_id, cnt_order);

ALTER TABLE rr_count
  ADD (CONSTRAINT rr_count_pk PRIMARY KEY (record_id, cnt_order) USING INDEX rr_count_pk);

-- ================================================

CREATE TABLE rr_count_conv_offense AS SELECT * FROM rr_count_conv_offense_v;

CREATE UNIQUE INDEX rr_count_conv_offense_pk ON
                    rr_count_conv_offense(record_id, cnt_order, conv_offense_order);

ALTER TABLE rr_count_conv_offense
  ADD (CONSTRAINT rr_count_conv_offense_pk PRIMARY KEY (record_id, cnt_order, conv_offense_order)
       USING INDEX rr_count_conv_offense_pk);

-- ================================================

CREATE TABLE rr_count_fe AS SELECT * FROM rr_count_fe_v;

CREATE UNIQUE INDEX rr_count_fe_pk ON rr_count_fe(record_id, cnt_order, fe_num_order);
```

```
ALTER TABLE rr_count_fe
  ADD (CONSTRAINT rr_count_fe_pk PRIMARY KEY (record_id, cnt_order, fe_num_order)
       USING INDEX rr_count_fe_pk);

-- ============================================

CREATE TABLE rr_count_disp AS SELECT * FROM rr_count_disp_v;

CREATE UNIQUE INDEX rr_count_disp_pk ON rr_count_disp(record_id, cnt_order, disp_order);

ALTER TABLE rr_count_disp
  ADD (CONSTRAINT rr_count_disp_pk PRIMARY KEY (record_id, cnt_order, disp_order)
       USING INDEX rr_count_disp_pk);

-- Results of above Select count(*) statements

TOT_UNIQUE_COUNTS
-----------------
            27403

TOT_UNIQUE_CONV_OFFENSES
------------------------
                     389

TOT_UNIQUE_FE
-------------
         1031

TOT_UNIQUE_DISP
---------------
           5909
```

2.6.2 SAS

```
** Create unique set of counts from staging data set;
Data rr_count;
      Set rr_stage (keep= record_id subject_id cyc_order stp_order cnt_order disp_date ofn offense_code
            offense_descr offense_toc offense_qual_lst disp_offense_code disp_offense_descr
            disp_offense_toc disp_offense_qual_lst comment_text);
      by record_id subject_id cyc_order stp_order cnt_order;

      ** eliminate counts which do not exist for subjects;
      If missing(cnt_order) then delete;

      /* use first.* to get distinct rows but only if step exist */
      If first.cnt_order;
Run;

NOTE: There were 31374 observations read from the data set LIBRR.RR_STAGE.
NOTE: The data set LIBRR.RR_COUNT has 27403 observations and 16 variables.
```

```
** Create unique set of convicted offenses for a given count from staging data set;
Data rr_count_conv_offense;
      Set rr_stage (keep= record_id subject_id cyc_order stp_order cnt_order conv_offense_order
            conv_offense_code conv_offense_descr conv_offense_toc conv_offense_qual_lst);
      by record_id subject_id cyc_order stp_order cnt_order conv_offense_order;

      If missing(conv_offense_order) then delete;

      If first.conv_offense_order;
Run;

NOTE: The data set LIBRR.RR_COUNT_CONV_OFFENSE has 389 observations and 10 variables.
```

```
** Create unique set of field events for a given count from staging data set;
Data rr_count_fe;
      Set rr_stage (keep= record_id subject_id cyc_order stp_order cnt_order fe_num_order
            fe_num_arr_agy fe_num_bnch_warr fe_num_cite fe_num_docket fe_num_incident
            fe_num_booking fe_num_number fe_num_remand fe_num_oos_inn fe_num_crt_case fe_num_warrant);
      by record_id subject_id cyc_order stp_order cnt_order fe_num_order;
```

```
      If missing(fe_num_order) then delete;

      If first.fe_num_order;
Run;

NOTE: The data set LIBRR.RR_COUNT_FE has 1031 observations and 17 variables.
```

```
Data rr_count_disp;
      Set rr_stage (keep= record_id subject_id cyc_order stp_order cnt_order disp_order
            disp_code disp_descr conv_stat_code conv_stat_descr);
      by record_id subject_id cyc_order stp_order cnt_order disp_order;

      If missing(disp_order) then delete;

      If first.disp_order;
Run;

NOTE: The data set LIBRR.RR_COUNT_DISP has 5909 observations and 10 variables.
```

2.6.3 SPSS

```
*Create unique set of counts from staging data set.
DATASET ACTIVATE rr_stage.
SAVE OUTFILE= 'rr_count.sav'
    /KEEP
  RECORD_ID SUBJECT_ID CYC_ORDER STP_ORDER CNT_ORDER
  DISP_DATE OFN OFFENSE_CODE OFFENSE_DESCR OFFENSE_TOC OFFENSE_QUAL_LST DISP_OFFENSE_CODE
  DISP_OFFENSE_DESCR DISP_OFFENSE_TOC DISP_OFFENSE_QUAL_LST COMMENT_TEXT.

GET FILE = 'rr_count.sav'.
SORT CASES BY RECORD_ID(A) SUBJECT_ID(A) CYC_ORDER(A) STP_ORDER(A) CNT_ORDER(A).

*Eliminate counts which do not exist for subjects.
SELECT IF NOT (SYSMIS(CNT_ORDER)).

*Get distinct rows, but only if count exists.
SORT CASES BY RECORD_ID(A) SUBJECT_ID(A) CYC_ORDER(A) STP_ORDER(A) CNT_ORDER(A).
MATCH FILES
   /FILE=*
   /BY RECORD_ID SUBJECT_ID CYC_ORDER STP_ORDER CNT_ORDER
   /FIRST=PrimaryFirst.

VARIABLE LABELS  PrimaryFirst 'Indicator of each first matching case as Primary'.
VALUE LABELS  PrimaryFirst 0 'Duplicate Case' 1 'Primary Case'.
VARIABLE LEVEL  PrimaryFirst (ORDINAL).
FREQUENCIES VARIABLES=PrimaryFirst.
SELECT IF PrimaryFirst = 1.
FREQUENCIES VARIABLES=PrimaryFirst
  /ORDER=ANALYSIS.
DELETE VARIABLES PrimaryFirst.
```

```
*Create unique set of convicted offenses for a given count from stating data set.
DATASET ACTIVATE rr_stage.
SAVE OUTFILE= 'rr_count_conv_offense.sav'
   /KEEP
  RECORD_ID SUBJECT_ID CYC_ORDER STP_ORDER CNT_ORDER
  CONV_OFFENSE_ORDER CONV_OFFENSE_CODE CONV_OFFENSE_DESCR CONV_OFFENSE_TOC CONV_OFFENSE_QUAL_LST.

GET FILE = 'rr_count_conv_offense.sav'.
SORT CASES BY RECORD_ID(A) SUBJECT_ID(A) CYC_ORDER(A) STP_ORDER(A) CNT_ORDER(A) CONV_OFFENSE_ORDER(A).

*Eliminate convictions which do not exist for subjects.
SELECT IF NOT(SYSMIS(CONV_OFFENSE_ORDER)).

*Get distinct rows, but only if conviction exists.
SORT CASES BY RECORD_ID(A) SUBJECT_ID(A) CYC_ORDER(A) STP_ORDER(A) CNT_ORDER(A) CONV_OFFENSE_ORDER(A).
MATCH FILES
   /FILE=*
   /BY RECORD_ID SUBJECT_ID CYC_ORDER STP_ORDER CNT_ORDER CONV_OFFENSE_ORDER
   /FIRST=PrimaryFirst.

VARIABLE LABELS  PrimaryFirst 'Indicator of each first matching case as Primary'.
VALUE LABELS  PrimaryFirst 0 'Duplicate Case' 1 'Primary Case'.
VARIABLE LEVEL  PrimaryFirst (ORDINAL).
FREQUENCIES VARIABLES=PrimaryFirst.
SELECT IF PrimaryFirst = 1.
FREQUENCIES VARIABLES=PrimaryFirst
  /ORDER=ANALYSIS.
DELETE VARIABLES PrimaryFirst.
```

```
*Create unique set of field events for a given count from staging data set.
DATASET ACTIVATE rr_stage.
SAVE OUTFILE= 'rr_count_fe.sav'
   /KEEP
  RECORD_ID  SUBJECT_ID  CYC_ORDER  STP_ORDER  CNT_ORDER  FE_NUM_ORDER  FE_NUM_ARR_AGY
  FE_NUM_BNCH_WARR  FE_NUM_CITE  FE_NUM_DOCKET  FE_NUM_INCIDENT  FE_NUM_BOOKING
  FE_NUM_NUMBER  FE_NUM_REMAND  FE_NUM_OOS_INN  FE_NUM_CRT_CASE  FE_NUM_WARRANT.

GET FILE = 'rr_count_fe.sav'.
SORT CASES BY RECORD_ID(A) SUBJECT_ID(A) CYC_ORDER(A) STP_ORDER(A) CNT_ORDER(A) FE_NUM_ORDER(A).

*Eliminate field events which do not exist for subjects.
SELECT IF NOT(SYSMIS(FE_NUM_ORDER)).

*Get distinct rows, but only if field event exists.
SORT CASES BY RECORD_ID(A) SUBJECT_ID(A) CYC_ORDER(A) STP_ORDER(A) CNT_ORDER(A) FE_NUM_ORDER(A).
MATCH FILES
   /FILE=*
   /BY RECORD_ID SUBJECT_ID CYC_ORDER STP_ORDER CNT_ORDER FE_NUM_ORDER
   /FIRST=PrimaryFirst.

VARIABLE LABELS  PrimaryFirst 'Indicator of each first matching case as Primary'.
VALUE LABELS  PrimaryFirst 0 'Duplicate Case' 1 'Primary Case'.
VARIABLE LEVEL  PrimaryFirst (ORDINAL).
FREQUENCIES VARIABLES=PrimaryFirst.
SELECT IF PrimaryFirst = 1.
FREQUENCIES VARIABLES=PrimaryFirst
  /ORDER=ANALYSIS.
DELETE VARIABLES PrimaryFirst.
```

```
*Create unique set of dispositions for a given count from staging data set.
DATASET ACTIVATE rr_stage.
SAVE OUTFILE= 'rr_count_disp.sav'
   /KEEP
  RECORD_ID SUBJECT_ID CYC_ORDER STP_ORDER CNT_ORDER
  DISP_ORDER DISP_CODE DISP_DESCR CONV_STAT_CODE CONV_STAT_DESCR.

GET FILE = 'rr_count_disp.sav'.
SORT CASES BY RECORD_ID(A) SUBJECT_ID(A) CYC_ORDER(A) STP_ORDER(A) CNT_ORDER(A) DISP_ORDER(A).

*Eliminate field events which do not exist for subjects.
SELECT IF NOT(SYSMIS(DISP_ORDER)).

*Get distinct rows, but only if disposition exists.
SORT CASES BY RECORD_ID(A) SUBJECT_ID(A) CYC_ORDER(A) STP_ORDER(A) CNT_ORDER(A) DISP_ORDER(A).
MATCH FILES
   /FILE=*
   /BY RECORD_ID SUBJECT_ID CYC_ORDER STP_ORDER CNT_ORDER DISP_ORDER
   /FIRST=PrimaryFirst.

VARIABLE LABELS  PrimaryFirst 'Indicator of each first matching case as Primary'.
VALUE LABELS  PrimaryFirst 0 'Duplicate Case' 1 'Primary Case'.
VARIABLE LEVEL  PrimaryFirst (ORDINAL).
FREQUENCIES VARIABLES=PrimaryFirst.
SELECT IF PrimaryFirst = 1.
FREQUENCIES VARIABLES=PrimaryFirst
  /ORDER=ANALYSIS.
DELETE VARIABLES PrimaryFirst.
```

2.7 Extract Sentence Data

This section will extract all the attributes associated with Sentences and the attributes to allow joining to the associated cycle, step, count and subject. The extracted attributes are members of the following segments:

- **Subject Record Segment**: Attributes to uniquely identify a subject
- **Cycle Segment**: Cycle order
- **Step Segment**: Step order
- **Count Segment**: Count order
- **Sentence Segment**: Sentence information for counts

2.7.1 Oracle

```
CREATE OR REPLACE VIEW rr_sentence_v AS
  SELECT DISTINCT record_id, subject_id, cyc_order, stp_order, cnt_order, sent_order, sent_loc_code,
                  sent_loc_descr, sent_length, sent_time_code, sent_time_descr, comment_text
    FROM rr_stage
   WHERE sent_order IS NOT NULL;

SELECT Count(*) tot_unique_sentences FROM rr_sentence_v;

-- for performance improvements it is recommended that the following items be created
-- this will be significant when joining across multiple tables
CREATE TABLE rr_sentence AS SELECT * FROM rr_sentence_v;

CREATE UNIQUE INDEX rr_sentence_pk ON rr_sentence(record_id, cnt_order, sent_order);

ALTER TABLE rr_sentence
  ADD (CONSTRAINT rr_sentence_pk PRIMARY KEY (record_id, cnt_order, sent_order)
       USING INDEX rr_sentence_pk);

-- Results of above Select count(*) statements
TOT_UNIQUE_SENTENCES
--------------------
                4115
```

2.7.2 SAS

```
** Create unique set of sentences from staging data set;
Data rr_sentence;
      Set rr_stage (keep= record_id subject_id cyc_order stp_order cnt_order sent_order sent_loc_code
                          sent_loc_descr sent_length sent_time_code sent_time_descr comment_text);
      by record_id subject_id cyc_order stp_order cnt_order sent_order;

      If missing(sent_order) then delete;

      If first.sent_order;
Run;

NOTE: There were 31374 observations read from the data set LIBRR.RR_STAGE.
NOTE: The data set LIBRR.RR_SENTENCE has 4115 observations and 12 variables.
```

2.7.3 SPSS

```
*Create unique set of sentence information from staging data set.
DATASET ACTIVATE rr_stage.
SAVE OUTFILE= 'rr_sentence.sav'
   /KEEP
  RECORD_ID SUBJECT_ID CYC_ORDER STP_ORDER CNT_ORDER SENT_ORDER
  SENT_LOC_CODE SENT_LOC_DESCR SENT_LENGTH SENT_TIME_CODE SENT_TIME_DESCR COMMENT_TEXT.

GET FILE = 'rr_sentence.sav'.
SORT CASES BY RECORD_ID(A) SUBJECT_ID(A) CYC_ORDER(A) STP_ORDER(A) CNT_ORDER(A) SENT_ORDER(A).

*Eliminate field events which do not exist for subjects.
SELECT IF NOT(SYSMIS(SENT_ORDER)).

*Get distinct rows, but only if disposition exists.
SORT CASES BY RECORD_ID(A) SUBJECT_ID(A) CYC_ORDER(A) STP_ORDER(A) CNT_ORDER(A) SENT_ORDER(A).
MATCH FILES
   /FILE=*
   /BY RECORD_ID SUBJECT_ID CYC_ORDER STP_ORDER CNT_ORDER SENT_ORDER
   /FIRST=PrimaryFirst.

VARIABLE LABELS  PrimaryFirst 'Indicator of each first matching case as Primary'.
VALUE LABELS  PrimaryFirst 0 'Duplicate Case' 1 'Primary Case'.
VARIABLE LEVEL  PrimaryFirst (ORDINAL).
FREQUENCIES VARIABLES=PrimaryFirst.
SELECT IF PrimaryFirst = 1.
FREQUENCIES VARIABLES=PrimaryFirst
  /ORDER=ANALYSIS.
DELETE VARIABLES PrimaryFirst.
```

3 SCENARIOS

This section will provide examples of different scenarios on extracting information from the tables created in section 2. Each scenario will define the scenario name, objective(s), any definitions and instructions on meeting said objective(s).

3.1 Scenario 1 – Convicted Cycles

3.1.1 Objectives

1. Identify all counts by subject that have a Court level conviction
2. Indicate the most egregious convicted type of charge (i.e. Felony, Misdemeanor, etc.) for the Step
    a. For example, if there are 4 convicted counts, what is the type of charge for the most egregious convicted count

3.1.2 Definitions

- **Convicted**: DISP_CODE between 2500 and 2799
- **Most egregious conviction**: CONV_STAT_CODE (1 – highest to 6 – lowest) if given or OFFENSE_TOC otherwise

3.1.3 Instructions

Processing steps:

1. Identify all convicted counts
    a. If CONV_STAT_CODE is not given, map OFFENSE_TOC to the equivalent CONV_STAT_CODE
2. Determine most egregious conviction for the step (stp_order) for results from processing step 1

The subsections below will offer sample code to meet the objectives in Oracle and SAS.

3.1.3.1 Oracle

```
WITH conv_det AS
        (SELECT c.record_id, c.subject_id, c.cyc_order, c.stp_order, c.cnt_order,
                d.disp_order, c.disp_date, c.ofn, c.offense_code, c.offense_descr,
                c.offense_toc, d.disp_code, d.disp_descr, d.conv_stat_code, d.conv_stat_descr,
                CASE
                  WHEN d.conv_stat_code IS NOT NULL THEN conv_stat_code
                  WHEN c.offense_toc = 'F' THEN '1'
                  WHEN c.offense_toc = 'M' THEN '2'
                  WHEN c.offense_toc = 'I' THEN '6'
                  ELSE '9'
                END
                  most_egr_conv
           FROM rr_count c, rr_count_disp d
          WHERE d.record_id = c.record_id
            AND d.cnt_order = c.cnt_order
            AND d.disp_code BETWEEN '2500' AND '2799'),
      egr AS
        (SELECT c.record_id, c.subject_id, c.cyc_order, c.stp_order, c.cnt_order,
                c.disp_order, c.disp_date, c.ofn, c.offense_code, c.offense_descr, c.offense_toc,
                c.disp_code, c.disp_descr, c.conv_stat_code, c.conv_stat_descr,
                Min(most_egr_conv) OVER (PARTITION BY c.record_id, c.stp_order) most_egr_conv
           FROM conv_det c)
SELECT record_id, subject_id, cyc_order, stp_order, cnt_order,
       disp_order, offense_code, offense_descr, offense_toc,
       disp_code, disp_descr, conv_stat_code, conv_stat_descr, most_egr_conv,
       (CASE
          WHEN most_egr_conv = '1' THEN 'Felony'
          WHEN most_egr_conv = '6' THEN 'Infraction'
          WHEN most_egr_conv = '9' THEN 'Unknown'
          ELSE 'Misdemeanor'
        END)
         most_egr_conv_descr
  FROM egr
ORDER BY record_id, stp_order, cnt_order, disp_order;

-- Above select returned 1942 rows
```

Sample results from above SQL. A few columns were excluded for readability.

RECORD_ID	STP_ORDER	OFFENSE_DESCR	OFFENSE TOC	DISP CODE	DISP_DESCR	CONV STAT CODE	CONV STAT DESCR	MOST EGR CONV DESCR
100000000076	102001000000	459 PC-BURGLARY:FIRST DEGREE	F	2500	CONVICTED			Felony
100000000076	104002000000	459 PC-BURGLARY	F	2585	CONVICTED COMMITTED TO PRISON	1	FELONY	Felony
100000000076	104002000000	460(B) PC-BURGLARY:SECOND DEGREE	F	2598	CONVICTED-JAIL	2	MISDEMEANOR	Felony
100000000076	104002000000	211 PC-ROBBERY	F	2600	CONVICTED-PROB/JAIL	3	MISDEMEANOR	Felony
100000000076	104002000000	369G PC-DRIVE ALONG RAILROAD TRACK	M	2598	CONVICTED-JAIL	4	MISDEMEANOR	Felony
100000000076	104002000000	382.5 PC-SELL/PRESCRIBE DINITROPHENOL	F	2597	CONVICTED-PROBATION	5	MISDEMEANOR	Felony
100000000076	104002000000	374B PC-DUMP OFFENSIVE MATTER	I	2598	CONVICTED-JAIL	6	INFRACTION	Felony
100000000077	101002000000	459 PC-BURGLARY	F	2597	CONVICTED-PROBATION	3	MISDEMEANOR	Misdemeanor
100000001078	101002000000	23152(B) VC-DUI ALCOHOL/0.08% W/PRIORS	F	2600	CONVICTED-PROB/JAIL	2	MISDEMEANOR	Felony
100000001078	101002000000	23152(B) VC-DUI ALCOHOL/0.08% W/PRIORS	F	2585	CONVICTED COMMITTED TO PRISON	1	FELONY	Felony
100000002494	101003000000	464 PC-BURGLARY WITH EXPLOSIVES/ETC	F	2585	CONVICTED COMMITTED TO PRISON	1	FELONY	Felony
100000002494	101003000000	459 PC-BURGLARY	F	2598	CONVICTED-JAIL	2	MISDEMEANOR	Felony
100000002494	101003000000	464 PC-BURGLARY WITH EXPLOSIVES/ETC	F	2500	CONVICTED	2	MISDEMEANOR	Felony

3.1.3.2 SAS

```
options user=LIBRR; /* setup default library to use (optional) */

Data w1 (keep=record_id subject_id cyc_order stp_order cnt_order disp_order
  disp_date ofn offense_code offense_descr offense_toc disp_code disp_descr
  conv_stat_code conv_stat_descr egr_conv egr_conv_descr);
  merge rr_count (IN=l) rr_count_disp (IN=r);
  by record_id subject_id cyc_order stp_order cnt_order;
  format egr_conv $char1.;
  format egr_conv_descr $char12.;

  If l and r; ** inner join;

  ** exclude observations where disposition is not a conviction;
  If '2500' <= disp_code <= '2799';

  ** setup temp cols for use later to determine most egregious;
  if not missing(conv_stat_code) then
    do;
      egr_conv = conv_stat_code;
      egr_conv_descr = propcase(conv_stat_descr);
    end;
  else
    do;
      select (offense_toc);
        when ('F') do; egr_conv = '1'; egr_conv_descr = 'Felony'; end;
        when ('M') do; egr_conv = '2'; egr_conv_descr = 'Misdemeanor'; end;
        when ('I') do; egr_conv = '6'; egr_conv_descr = 'Infraction'; end;
        otherwise  do; egr_conv = '9'; egr_conv_descr = 'Unknown'; end;
      end;
    end;
Run;

Proc Sort data=w1 EQUALS;
  by record_id subject_id cyc_order stp_order disp_order egr_conv;
Run;
```

```
** determine most egregious conviction at ACHS Step level;
Data Scenario01_results (keep=record_id subject_id cyc_order stp_order cnt_order disp_order
  disp_date ofn offense_code offense_descr offense_toc disp_code disp_descr
  conv_stat_code conv_stat_descr egr_conv egr_conv_descr most_egr_conv most_egr_conv_descr);
  set w1;
  by record_id subject_id cyc_order stp_order disp_order egr_conv;
  format most_egr_conv $char1.;
  format most_egr_conv_descr $char12.;

  If first.stp_order then
    do;
      mec = egr_conv;
      mecd = egr_conv_descr;
      retain mec;
      retain mecd;
    end;

  most_egr_conv = mec;
  most_egr_conv_descr = mecd;
Run;

Proc Print data=Scenario01_results noobs;
  var record_id stp_order offense_descr offense_toc disp_code disp_descr conv_stat_code conv_stat_descr
most_egr_conv_descr;

title 'Scenario 1 - Convicted Cycles';

Run;

** do a little cleanup (optional);
Proc Datasets library=LIBRR nolist nodetails;
  delete w1;
Run;

NOTE: There were 27403 observations read from the data set LIBRR.RR_COUNT.
NOTE: There were 5909 observations read from the data set LIBRR.RR_COUNT_DISP.
NOTE: The data set LIBRR.W1 has 1942 observations and 17 variables.
NOTE: The data set LIBRR.SCENARIO01_RESULTS has 1942 observations and 19 variables.
```

3.2 Scenario 2 – Count of Arrests by Hair Color and Gender

3.2.1 Objectives

1. Report number of subjects that were arrested broken down by Hair Color and Gender
   a. Only examine subjects for which CORI data was pulled (i.e. valid subjects)
   b. Count arrest regardless of disposition
   c. A subject should only be counted once for this total
2. Report % of subjects with an arrest by Hair Color and Gender
3. Report number of total arrests broken down by Hair Color and Gender
   a. Count arrest regardless of disposition
   b. A subject may be counted multiple times for this total because of multiple arrest cycles
4. Report % of overall arrests by Hair Color and Gender

3.2.2 Definitions

- **Valid Subject**: SUBJECT_STATUS = 'F'
- **Arrest**: STP_TYPE_CODE is one of '1A', '1B', or '1C'
- **Multiple Arrests**: Multiple Cycles with an Arrest step for same subject

3.2.3 Instructions

Processing steps:

1. Identify all valid subjects with and without an arrest
   a. Determine the unique number of subjects by hair color and gender
   b. Determine the unique number of subjects arrested by hair color and gender
   c. Determine the unique number of subjects arrested overall
2. Generate report grouped by hair color and gender

The subsections below will offer sample code to meet the objectives in Oracle and SAS.

3.2.3.1 Oracle

```
WITH w2 AS
        (SELECT s.record_id, s.hair_color_descr, s.gender, a.cyc_order,
                (CASE
                   WHEN a.record_id IS NOT NULL AND a.cyc_order IS NOT NULL THEN 1
                   ELSE 0
                 END) arrest,
                Row_number() OVER(PARTITION BY s.record_id ORDER BY s.record_id) subj_rn,
                Row_number() OVER(PARTITION BY a.record_id ORDER BY a.record_id) arr_rn,
                Count(DISTINCT a.record_id) OVER () arrest_subj_tot,
                Count(DISTINCT s.record_id) OVER () unique_subj_tot,
                Count(*) OVER () obs_tot
           FROM rr_subject s LEFT OUTER JOIN rr_step a
                  ON (a.record_id = s.record_id AND a.stp_type_code IN ('1A', '1B', '1C'))
          WHERE s.subject_status = 'F'),
      w2m AS
        (SELECT hair_color_descr, gender, Count(DISTINCT record_id) unique_subj_cnt,
                Count(*) all_arr_cnt, Sum(arrest) arrest_cnt,
                Sum(CASE
                      WHEN arrest = 1 AND arr_rn = 1 THEN 1
                      ELSE 0
                    END) subj_arrested_cnt,
                Max(arrest_subj_tot) arrest_subj_tot, Max(unique_subj_tot) unique_subj_tot,
                Max(obs_tot) obs_tot
           FROM w2
         GROUP BY hair_color_descr, gender)
SELECT hair_color_descr "Hair Color", gender "Gender",
       subj_arrested_cnt "Subjects Arrested", unique_subj_cnt "Total Subjects",
       Round((subj_arrested_cnt / unique_subj_cnt) * 100, 2) "% Arrested",
       Round((subj_arrested_cnt / arrest_subj_tot) * 100, 2) "% Overall Arrests"
  FROM w2m
ORDER BY hair_color_descr, gender;

-- Above select returned 25 rows
```

Showing results from above SQL. **Please note that the numbers shown are not indicative of actual arrest information.** The sample data is not representative of actual arrest data.

Hair Color	Gender	Subjects Arrested	Total Subjects	% Arrested	% Overall Arrests
Bald	F	4	5	80	0.16
Bald	M	5	7	71.43	0.2
Bald	X	2	2	100	0.08
Black	F	177	213	83.1	7.06
Black	M	223	316	70.57	8.89
Black	X	97	128	75.78	3.87
Blond or Strawberry	F	38	48	79.17	1.52
Blond or Strawberry	M	76	91	83.52	3.03
Blond or Strawberry	X	3	3	100	0.12
Blue	F	1	1	100	0.04
Blue	M	8	13	61.54	0.32
Brown	F	179	254	70.47	7.14
Brown	M	586	828	70.77	23.37
Brown	X	28	41	68.29	1.12
Gray or Partially Gray	M	10	10	100	0.4
Gray or Partially Gray	X	0	1	0	0
Green	F	1	1	100	0.04
Red or Auburn	F	23	30	76.67	0.92
Red or Auburn	M	21	35	60	0.84
Red or Auburn	X	2	3	66.67	0.08
Sandy	F	2	2	100	0.08
Unknown or Completely Bald	F	67	77	87.01	2.67
Unknown or Completely Bald	M	59	109	54.13	2.35
Unknown or Completely Bald	X	894	1338	66.82	35.65
White	M	2	3	66.67	0.08

3.2.3.2 SAS

```
options user=LIBRR; /* setup default library to use (optional) */

Data wStep (keep=record_id subject_id cyc_order arrested_subj arrest);
  set rr_step;
  by record_id subject_id cyc_order;
  where stp_type_code in ('1A', '1B', '1C');

  if first.cyc_order;

  arrested_subj = 0;
  arrest = 1;

  if first.record_id then arrested_subj = 1;
Run;

** Generate the data set containing the subject and arrest information to count;
** This will generate one row per subject per arrest cycle or 1 row if no arrest cycles;
Data w2 (keep=record_id subject_id hair_color_descr gender cyc_order first_subj arrested_subj arrest);
  merge rr_subject (IN=l) wStep (IN=r);
  by record_id subject_id;

  If subject_status='F';

  first_subj = 0;

  if first.record_id then first_subj = 1;

  if (l and not r) then
    do;
      arrested_subj = 0;
      arrest=0;
    end;
Run;

Proc Means data=w2 noprint;
  var first_subj arrested_subj arrest;
  class hair_color_descr gender;
  output out=w2m (drop=_freq_) n(first_subj) = obs_cnt
    sum(first_subj arrested_subj arrest) = unique_subj_cnt unique_arrested_subj arrest_cnt;
Run;
```

```
Proc Sort data=w2m out=w2ms;
  by hair_color_descr _type_ gender;
Run;

Data Scenario02_results;
  set w2ms;
  by hair_color_descr _type_ gender;

  If _type_ = 0 then
    do;
      tot_obs = obs_cnt;
      tot_u_subj = unique_subj_cnt;
      tot_ua_subj = unique_arrested_subj;
      tot_arr = arrest_cnt;
      retain tot_obs;
      retain tot_u_subj;
      retain tot_ua_subj;
      retain tot_arr;
    end;

  If _type_ = 3;

  pct_arr = (unique_arrested_subj/unique_subj_cnt) * 100;
  pct_ovr_arr = (unique_arrested_subj/tot_ua_subj) * 100;
Run;

Proc Print data=Scenario02_results noobs;
  var hair_color_descr gender unique_arrested_subj unique_subj_cnt pct_arr pct_ovr_arr;
  label
    hair_color_descr ='Hair Color'
    gender ='Gender'
    unique_arrested_subj ='Subjects Arrested'
    unique_subj_cnt ='Total Subjects'
    pct_arr ='% Arrested'
    pct_ovr_arr='% Overall Arrests';
  format pct_arr pct_ovr_arr 6.2;

  title "Count of Arrests by Hair Color and Gender";
Run;

** do a little cleanup (optional);
Proc Datasets library=LIBRR nolist nodetails;
  delete w2 wStep w2m w2ms;
Run;
```

```
NOTE: There were 7905 observations read from the data set LIBRR.RR_STEP.
      WHERE stp_type_code in ('1A', '1B', '1C');
NOTE: The data set LIBRR.WSTEP has 7282 observations and 5 variables.

NOTE: There were 3658 observations read from the data set LIBRR.RR_SUBJECT.
NOTE: There were 7282 observations read from the data set LIBRR.WSTEP.
NOTE: The data set LIBRR.W2 has 8333 observations and 8 variables.

NOTE: The data set LIBRR.W2M has 40 observations and 7 variables.
NOTE: The data set LIBRR.W2MS has 40 observations and 7 variables.
NOTE: The data set LIBRR.SCENARIO02_RESULTS has 25 observations and 13 variables.
```

3.3 Scenario 3 – Arrested and Convicted of Burglary by County

3.3.1 Objectives

1. All of the reported values below are to be grouped by arrest County
2. Report number of subjects that were arrested for burglary
   a. Report arrest regardless of disposition
   b. Multiple arrest steps for burglary under same cycle should be counted as one occurrence
   c. A subject may be counted multiple times for this total because of multiple arrest cycles
3. Report number of subjects that were arrested and convicted of burglary (in same cycle)
   a. A subject may be counted multiple times for this total because of multiple arrest/conviction cycles
   b. Multiple conviction counts for burglary under same step should be counted as one count
   c. Level of conviction is not used in this scenario, only that subjects were convicted of burglary
4. Report % of overall arrests and convictions of burglary by County

3.3.2 Definitions

- **Arrest**: STP_TYPE_CODE is one of '1A', '1B', or '1C'
- **Multiple Arrests**: Multiple Cycles with an Arrest step
- **Arrest County**: STP_ORI_CNTY_NAME from Arrest Step
- **Convicted**: DISP_CODE between 2500 and 2799
- **Burglary**: OFFENSE_CODE = '22004'

3.3.3 Instructions

Processing steps

1. Identify all cycles with an arrest step for burglary
   a. Group by county name, subject and cycle
2. Identify all cycles with a conviction for burglary for cycles identified in processing step 1
3. Merge result sets from steps 1 and 2 above and generate report columns

The subsections below will offer sample code to meet the objectives in Oracle and SAS.

3.3.3.1 Oracle

```
WITH arr_det AS
        (SELECT stp.stp_ori_cnty_name, stp.record_id, stp.cyc_order, 1 arr_cnt
           FROM rr_step stp, rr_count cnt
          WHERE stp.stp_type_code IN ('1A', '1B', '1C')
            AND cnt.offense_code = '22004'
            AND cnt.record_id = stp.record_id AND cnt.stp_order = stp.stp_order
         GROUP BY stp.stp_ori_cnty_name, stp.record_id, stp.cyc_order),
      conv_det AS
        (SELECT a.stp_ori_cnty_name, a.record_id, a.cyc_order, 1 convicted
           FROM arr_det a, rr_step stp, rr_count cnt, rr_count_disp cd
          WHERE stp.record_id = a.record_id AND stp.cyc_order = a.cyc_order
            AND stp.stp_type_code = '40'
            AND cnt.record_id = stp.record_id AND cnt.stp_order = stp.stp_order
            AND cnt.offense_code = '22004'
            AND cd.record_id = cnt.record_id AND cd.cnt_order = cnt.cnt_order
            AND cd.disp_code BETWEEN '2500' AND '2799'
         GROUP BY a.stp_ori_cnty_name, a.record_id, a.cyc_order),
      tot AS
        (SELECT Nvl(stp_ori_cnty_name, 'UNKNOWN') stp_ori_cnty_name, cnty_arr_tot, cnty_conv_tot,
                Sum(cnty_arr_tot) OVER () arr_tot, Sum(cnty_conv_tot) OVER () conv_tot
           FROM (SELECT a.stp_ori_cnty_name, Sum(a.arr_cnt) cnty_arr_tot, Sum(c.convicted) cnty_conv_tot
                   FROM arr_det a
                        LEFT OUTER JOIN conv_det c
                          ON (c.stp_ori_cnty_name = a.stp_ori_cnty_name
                          AND c.record_id = a.record_id
                          AND c.cyc_order = a.cyc_order)
                 GROUP BY a.stp_ori_cnty_name))
SELECT t.stp_ori_cnty_name "County Name", t.cnty_arr_tot "County Arrest Total",
       Round((t.cnty_arr_tot / t.arr_tot) * 100, 2) "% Overall Arrests",
       t.cnty_conv_tot "County Conviction Total",
       Round((t.cnty_conv_tot / t.conv_tot) * 100, 2) "% Overall Convictions"
  FROM tot t
ORDER BY t.stp_ori_cnty_name;
```

Showing results from above SQL. **Please note that the numbers shown are not indicative of actual arrest information.** The sample data is not representative of actual arrest data. The fact that 73% of burglaries occurred in Sacramento is purely because of the sample CORI data.

County Name	County Arrest Total	% Overall Arrests	County Conviction Total	% Overall Convictions
ALAMEDA	8	0.42	2	1.03
AMADOR	1	0.05		
BUTTE	3	0.16		
CALAVERAS	5	0.26		
COLUSA	2	0.11		
CONTRA COSTA	3	0.16		
DEL NORTE	1	0.05		
EL DORADO	1	0.05		
FRESNO	6	0.32		
HUMBOLDT	1	0.05		
IMPERIAL	1	0.05		
INYO	9	0.48	2	1.03
KERN	12	0.63	6	3.08
LASSEN	8	0.42		
LOS ANGELES	226	11.95	21	10.77
MADERA	2	0.11		
MARIPOSA	21	1.11		
MERCED	1	0.05		
MONTEREY	2	0.11		
NAPA	8	0.42		
NEVADA	4	0.21		
ORANGE	8	0.42	2	1.03
PLACER	8	0.42	4	2.05
PLUMAS	2	0.11		
RIVERSIDE	4	0.21		
SACRAMENTO	1349	71.30	143	73.33
SAN BERNARDINO	7	0.37		
SAN DIEGO	6	0.32	1	0.51
SAN FRANCISCO	57	3.01	8	4.10
SAN JOAQUIN	72	3.81	6	3.08
SAN LUIS OBISPO	1	0.05		
SAN MATEO	4	0.21		

SANTA BARBARA	3	0.16
SANTA CLARA	1	0.05
SANTA CRUZ	2	0.11
SHASTA	3	0.16
SISKIYOU	1	0.05
SOLANO	2	0.11
STANISLAUS	3	0.16
TEHAMA	10	0.53
TULARE	3	0.16
UNKNOWN	14	0.74
VENTURA	2	0.11
YOLO	3	0.16

3.3.3.2 SAS

```
options user=LIBRR; /* setup default library to use (optional) */

Data wArr (keep=record_id subject_id cyc_order stp_order arr_stp_type_code arr_cnty_name arrested);
  merge rr_step(IN=l rename=(stp_ori_cnty_name=arr_cnty_name stp_type_code=arr_stp_type_code))
        rr_count (IN=r);
  by record_id subject_id cyc_order stp_order;

  if arr_stp_type_code in ('1A', '1B', '1C') and offense_code = '22004';

  arrested = 1;
  if missing(arr_cnty_name) then arr_cnty_name='UNKNOWN';
Run;

Data wArr (keep=record_id subject_id cyc_order stp_order arr_stp_type_code arr_cnty_name arrested);
  set wArr;
  by record_id subject_id cyc_order;

  if first.cyc_order;
Run;

Data wCnv (keep=record_id subject_id cyc_order stp_order stp_type_code convicted);
  merge rr_step(IN=s) rr_count(IN=c) rr_count_disp(IN=d);
  by record_id subject_id cyc_order stp_order;

  if stp_type_code = ('40') and offense_code = '22004';
```

```
  if '2500' <= disp_code <= '2799';
  convicted = 1;
Run;

Data wCnv (keep=record_id subject_id cyc_order stp_order stp_type_code arr_cnty_name arrested convicted);
  merge wArr (IN=l) wCnv(IN=r);
  by record_id subject_id cyc_order;

  if first.cyc_order;
Run;

Proc Means data=wCnv noprint;
  var arrested convicted;
  class arr_cnty_name;
  output out=w3m (drop=_freq_) n(arrested) = obs_cnt
    sum(arrested convicted) = arrested_cnt convicted_cnt;
Run;

Data Scenario03_results;
  set w3m;
  by _type_ arr_cnty_name;

  If _type_ = 0 then
    do;
      tot_obs = obs_cnt;
      tot_arrested = arrested_cnt;
      tot_convicted = convicted_cnt;
      retain tot_obs;
      retain tot_arrested;
      retain tot_convicted;
    end;

  If _type_ = 1;
  pct_ovr_arr = (arrested_cnt/tot_arrested) * 100;
  pct_ovr_cnv = (convicted_cnt/tot_convicted) * 100;
Run;

Proc Print data=Scenario03_results label noobs;
  var arr_cnty_name arrested_cnt pct_ovr_arr convicted_cnt pct_ovr_cnv;
  label
    arr_cnty_name ='County Name'
    arrested_cnt ='County Arrest Total'
    pct_ovr_arr ='% Overall Arrests'
    convicted_cnt ='County Conviction Total'
    pct_ovr_cnv ='% Overall Convictions';
```

```
  format pct_ovr_arr pct_ovr_cnv 6.2;
  title "Scenario 3 - Arrested and Convicted of Burglary by County";
Run;

** do a little cleanup (optional);
Proc Datasets library=LIBRR nolist nodetails;
  delete wArr wCnv w3m;
Run;

NOTE: There were 16680 observations read from the data set LIBRR.RR_STEP.
NOTE: There were 27403 observations read from the data set LIBRR.RR_COUNT.
NOTE: The data set LIBRR.WARR has 3651 observations and 7 variables.

NOTE: There were 3651 observations read from the data set LIBRR.WARR.
NOTE: The data set LIBRR.WARR has 1890 observations and 7 variables.

NOTE: There were 16680 observations read from the data set LIBRR.RR_STEP.
NOTE: There were 27403 observations read from the data set LIBRR.RR_COUNT.
NOTE: There were 5909 observations read from the data set LIBRR.RR_COUNT_DISP.
NOTE: The data set LIBRR.WCNV has 416 observations and 6 variables.

NOTE: There were 1890 observations read from the data set LIBRR.WARR.
NOTE: There were 416 observations read from the data set LIBRR.WCNV.
NOTE: The data set LIBRR.WCNV has 1919 observations and 8 variables.

NOTE: There were 1919 observations read from the data set LIBRR.WCNV.
NOTE: The data set LIBRR.W3M has 46 observations and 5 variables.
```