|  |
| --- |
| Photo displaying partial image of two pie charts on a canvas-textured page |
| Documentation of CCAP Data Retrieval and AnalysisIn Support of the UW Law Review Article – “The Use of Wisconsin’s Bail Jumping Statute: A Legal and Quantitative Analysis” |
|

|  |  |  |
| --- | --- | --- |
| Mark Johnson | 10/15/17 | The Use of Wisconsin’s Bail Jumping Statute: A Legal and Quantitative Analysis |

 |

Table of Contents

[Introduction 2](#_Toc495734143)

[CCAP Data 3](#_Toc495734144)

[Data Retrieval Logic 4](#_Toc495734145)

[Google Sheet Approach 4](#_Toc495734146)

[Java Approach 4](#_Toc495734147)

[Get Case Count (CCAP\_switch == 1) 5](#_Toc495734148)

[Get Case Count and Write to Table CCAP\_CASE\_ALL (CCAP\_switch == 2) 6](#_Toc495734149)

[Get Case Count and Write to JSON Table CCAP\_JSON (CCAP\_switch == 3) 9](#_Toc495734150)

[Populate CCAP\_CASE\_SUMMARY\_MASTER (CCAP\_switch == 4) 10](#_Toc495734151)

[Populate CCAP\_REPORT\_MASTER (CCAP\_switch == 5) 12](#_Toc495734152)

[Reports, Graphs and SQL Statements 14](#_Toc495734153)

[Case and Charges Counts 14](#_Toc495734154)

[Bail Jumping Percentage (2000 – 2016) 15](#_Toc495734155)

[Bail Jumping Charges Dismissed 16](#_Toc495734156)

[Top Ten Charges - Statewide 18](#_Toc495734157)

[County Level Charges – All Counties 19](#_Toc495734158)

[Bail Jumping Charges Over Time 21](#_Toc495734159)

[Top Ten Charges – Each County 22](#_Toc495734160)

[Bail Jumping Leverage – All Counties 23](#_Toc495734161)

[Bail Jumping Leverage – County 2000 to 2016 comparison 24](#_Toc495734162)

[Bail Jumping Leverage – County Graph 25](#_Toc495734163)

[Bail Jumping Dismissed, Other Charge(s) not plead – 7 counties 26](#_Toc495734164)

[Bail Jumping Dismissed, Other Charge(s) not plead – All counties over time 27](#_Toc495734165)

[High Number of Bail Jumping Charges in single case 28](#_Toc495734166)

[Excessive Bail Jumping charges in single case 30](#_Toc495734167)

[Excel Spreadsheet - CCAP results (September 2017) MASTER 31](#_Toc495734168)

[Database Structures 32](#_Toc495734169)

[CCAP\_CASE\_ALL 32](#_Toc495734170)

[CCAP\_JSON 33](#_Toc495734171)

[CCAP\_ CASE\_SUMMARY\_MASTER 33](#_Toc495734172)

[CCAP\_ REPORT\_MASTER 34](#_Toc495734173)

# Introduction

This document describes the process used to retrieve and tabulate the Wisconsin Consolidated Court Automation Programs (CCAP) data related to the analysis of the State of Wisconsin Bail Jumping Charges from the years 2000 to 2016, as described in the University of Wisconsin Law Review article “The Use of Wisconsin’s Bail Jumping Statute: A Legal and Quantitative Analysis”.

Bail jumping charges were evaluated over time and between counties. The focus was to determine if the occurrence of bail jumping charges are increasing, and if bail jumping charges are being used in order to gain a plea to other charges associated with a case (also referred to as leverage).

The document consists of the following Sections.

* **CCAP Data.** Provides an overview of the CCAP data
* **Data Retrieval Logic.** Describes the approach to retrieve, parse and store the CCAP data in a database.
* **Reports, Graphs and SQL Statements.** Lists all the reports and graphs found in the Law Review paper, along with the SQL statements that were used to generate the needed data.
* **Excel Spreadsheet.** Contains a link to the spreadsheet that contains all the reports and graphs of the Law Review paper.
* **Database Structures.** Provides the structures of the database tables.

# CCAP Data

The data is retrieved from the Wisconsin Court System using the JSON access provided by the Consolidated Court Automation Programs (CCAP) Case Management system.

CCAP data is available to public view under Wisconsin's open records law, sections 19.31-19.39, Wisconsin Statutes. Citizens have access to this data through the public URL <https://www.wicourts.gov/casesearch.htm>. This URL provides the ability for a citizen to search for cases based on criteria such as name, county, and birth date.

While this search procedure satisfies a citizen or business need to search for specific cases, it is not useful in retrieving all the cases and charges (for felonies and misdemeanors) in the State of Wisconsin from the years 2000-2016. This consists of over 1.62 million cases and 3.28 million charges. In order to retrieve this large amount of data the CCAP JSON data retrieval mechanism is used.

The CCAP JSON data retrieval consists of a set of APIs that can be called, as documented in the following location.

<https://wccarest.wicourts.gov/api/v1/documentation#!/api-docs>

Note that the CCAP JSON data retrieval is not accessible by the general public, and in order to access the data via JSON a userid and password is required. The JSON API that is used to access the case data was the “GET /cases/{countyNo}/{caseNo}”. This API returns detailed case information, and requires that the county number and case number be supplied. An example of this JSON call is shown below:

<https://xxxxx/api/v1/cases/13/2016CF000001> (note, xxxx has replaced the internal call location).

“13” is the county number for Dane county. The case number (2016CF000001) consists of YYYYCCNNNNNN, where YYYY represents the year, CC represents the case type (CM for Misdemeanor and CF for Felony), and NNNNNN represents the case number.

Limitations of the CCAP JSON facility are:

1. There is no API that provides the number of cases for a given county/case type.
- This makes it difficult to retrieve all the cases for a given county/case type.
2. There are “gaps” in the case data. For example, 2016CM000913 for Milwaukee county (40) is not found. 2016CM000914 does exist.
- This also makes it difficult to retrieve all the cases for a given county/case type, since one cannot simply stop retrieving data at the first encounter of a “not found” condition.
3. Some cases have duplicate charge numbers. For example, case 2016CF000895 for Dane county (13) has two charge number 2 items ("chargeNo": 2,).
- Initially a unique key was established on the case data entity, consisting of case number and charge number. However, since the CCAP data has these duplicate charge data integrity issues, the unique key was removed.

Given this understanding of the data, data retrieval logic was developed to retrieve all the Wisconsin Felony and Misdemeanor cases for the years 2000 to 2016. This logic is described in the following sections.

# Data Retrieval Logic

## Google Sheet Approach

The initial logic to retrieve the case data using the CCAP JSON routine was written in a Google Sheet. The Sheet called a javascript routine named getCCAPBase, passing in the county, year and case type. A range of the cases to retrieve was also passed. Google Sheet javascript routines have a 30 second constraint, if the script runs for more than 30 seconds it fails. Therefore, only a small set of cases (normally 100) could be retrieved using one call. Another Google Sheet restriction is the amount of times the JSON routine could be called in one day. The logic calls UrlFetchApp.fetch, Google limits calls to UrlFetch to 50,000 calls per day.

While the Google Sheet approach provided a productive mechanism to understand the CCAP data, it was not feasible to use to retrieve 1.5 million cases. Because of these limitations, the Google Sheet approach was abandoned.

## Java Approach

A Java class was written named CCAP to retrieve the CCAP data. There are no limitations or constraints with using Java to call the CCAP JSON routine.

The Java code was developed within Eclipse, and consists of a simple main() method that contains all the logic.
The code is executed within Eclipse via the Run command.

There are 5 different execution processes within the main() method, driven by the CCAP\_switch attribute.

// if 1 will just get counts
// if 2 get counts and write to table (non json) - CCAP\_CASE\_ALL
// if 3 get counts and write to json table - CCAP\_JSON
// if 4 populates CCAP\_CASE\_SUMMARY\_MASTER
// if 5 populates CCAP\_REPORT\_MASTER

The five different execution modes are described in the following sections.

### Get Case Count (CCAP\_switch == 1)

// if 1 will just get counts
*getNumCCAPCases (getCasesYear, cmCaseType, getCaseCounty)*

As mentioned above, there isn’t a CCAP JSON routine to get the number of cases for a given county/case type.

The getNumCCAPCases() method determines the number of cases there are for a given year, case type and county. The routine executes a type of binary search, searching repeatedly for a case number. The search routine is bound by a high value, and does not search beyond this point. The high value is set to 10,000, which is higher than the max number of cases for a given county/case type for all counties. If the search returns a case not found, then the value is halved and the search is executed again. If the search returns a case found, then the value is incremented by an amount equal to the value just used plus the high value not found, divided by two. Markers are used and updated to keep track of the “low” and “high” values where case numbers are found. The search continues until the “low” and “high” values are the same.

Note that this method is not fool-proof. As mentioned previously, there are “gaps” in the case data. It is possible that the binary search could land on a gap, causing it to not locate cases above this point.

For this reason, the case count is incremented by a value of 300 when calling the CCAP JSON routine to retrieve the case data.

### Get Case Count and Write to Table CCAP\_CASE\_ALL (CCAP\_switch == 2)

// if 2 get counts and write to table (non json) - CCAP\_CASE\_ALL
*processCCAP (caseNumber, year, "CF/CM", county, DBconnection)*

For this execution type, the getNumCCAPCases() is called, after which the processCCAP() method is called, which retrieves the CCAP JSON data and inserts the data into the CCAP\_CASE\_ALL table. processCCAP() is called multiple times, based on the number of cases returned by the getNumCCAPCases() method. As mentioned earlier, 300 is added to this number because of the case gap issue with the CCAP data.

processCCAP() is passed the case number, year, case type (either CF or CM), the county, and the database connection. processCCAP() creates an HttpURLConnection object, setting the credentials, URL, and request method. connection.getInputStream() is called which retrieves the CCAP information for a single case into an InputStream parameter. The InputStream is read, put into a JSONObject and parsed.

The following data elements are parsed from the CCAP JSON results set:

* caseNo (Case Number)
* statusCode (Status of the case, either OP (open) or CL (closed))
* dob (Date of birth of the defendant)
* sex (Sex of the defendant)
* raceCode (Race code of the defendant)

All the Charges have the following information for each charge:

* chargeNo (Charge Number – note that there can be duplicates)
* descr (description of the charge)
* statuteCite (the statute that is cited for the charge).
* pleaCode (the plea code for the charge)
* dispoCode (the disposition code for the charge. Note that some of the charges have multiple disposition codes)

Each Case/Charge combination is then inserted into the CCAP\_CASE\_ALL table, which is shown below:

CCAP\_CASE\_ALL

 [CaseNum] [varchar](50) NOT NULL,

 [ChargeNum] [varchar](50) NOT NULL,

 [County] [varchar](50) NOT NULL,

 [Year] [varchar](50) NOT NULL,

 [Status] [varchar](50) NOT NULL,

 [CaseType] [varchar](50) NOT NULL,

 [Statute] [varchar](50) NOT NULL,

 [Description] [varchar](150) NULL,

 [PartyNum] [varchar](50) NULL,

 [PartyType] [varchar](50) NULL,

 [DOB] [varchar](50) NULL,

 [Sex] [varchar](50) NULL,

 [Race] [varchar](50) NULL,

 [PleaCode] [varchar](50) NULL,

 [JdgmtSeqNum] [varchar](50) NULL,

 [DispCode1] [varchar](50) NULL,

 [DispCode2] [varchar](50) NULL

To improve performance, CCAP\_CASE\_ALL has indexes on Year, Status, DispCode1, DispCode2, and Statute, as detailed below:

CREATE NONCLUSTERED INDEX [IX\_CCAP\_CASE\_ALL] ON [dbo].[CCAP\_CASE\_ALL]

( [Year] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

CREATE NONCLUSTERED INDEX [IX\_CCAP\_CASE\_ALL\_1] ON [dbo].[CCAP\_CASE\_ALL]

( [Status] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

CREATE NONCLUSTERED INDEX [IX\_CCAP\_CASE\_ALL\_2] ON [dbo].[CCAP\_CASE\_ALL]

( [DispCode1] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

CREATE NONCLUSTERED INDEX [IX\_CCAP\_CASE\_ALL\_3] ON [dbo].[CCAP\_CASE\_ALL]

( [DispCode2] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

CREATE NONCLUSTERED INDEX [IX\_CCAP\_CASE\_ALL\_4] ON [dbo].[CCAP\_CASE\_ALL]

( [Statute] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

During the runs it may be beneficial to update the statistics of the CCAP\_CASE\_ALL table.

UPDATE STATISTICS [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_ALL];

GO

Note that this process is run for a specified year, and takes about 6 hours to retrieve and parse the data and insert one years’ worth of data into the CCAP\_CASE\_ALL table. Also note that some cases will be sealed and are not available, and as mentioned, there are gaps and some cases are not found. Examples shown below (Year 2016).

for county: 67
case sealed and is not available :2016CF000297

for county: 64

case not found :2016CF000104

The number of rows inserted into the CCAP\_CASE\_ALL table are shown below. Note that the February 2017 run did not extend 300 searches beyond the maximum limit. This is the reason that the number of rows is greater in the September 2017 run.

|  |  |  |
| --- | --- | --- |
| Year | Feb 2017 Run | Sept 2017 Run |
| 2000 | 170326 | 171321 |
| 2001 | 179426 | 182065 |
| 2002 | 191536 | 191717 |
| 2003 | 195617 | 197730 |
| 2004 | 199704 | 201552 |
| 2005 | 209386 | 211390 |
| 2006 | 212357 | 212366 |
| 2007 | 204787 | 205464 |
| 2008 | 197676 | 198826 |
| 2009 | 184697 | 186212 |
| 2010 | 179058 | 180780 |
| 2011 | 178277 | 179995 |
| 2012 | 189263 | 189892 |
| 2013 | 192712 | 193355 |
| 2014 | 189056 | 190357 |
| 2015 | 194703 | 195506 |
| 2016 | 198011 | 199049 |

### Get Case Count and Write to JSON Table CCAP\_JSON (CCAP\_switch == 3)

// if 3 get counts and write to json table - CCAP\_JSON
processCCAPJSON (caseNumber, year, *"CF/CM"*, county, DBconnection)

processCCAPJSON() is very similar to the processCCAP() method. processCCAP() is passed the case number, year, case type (either CF or CM), the county, and the database connection. The CCAP information for a single case is retrieved and put into a JSONObject. The JSONObject is not parsed, but inserted directly into the CCAP\_JSON table.

Each Case is inserted into the CCAP\_JSON table, which is shown below:

CCAP\_JSON

 [CaseNum] [varchar](50) NOT NULL,

 [County] [varchar](50) NOT NULL,

 [Year] [varchar](50) NOT NULL,

 [CaseJSON] [nvarchar](max) NULL,
 [CountyCaseNum] [nvarchar](50) NULL

Note that this process takes a long time, completing the JSON population for one years’ worth of data in about 7 hours.

Also note that with the free version of Microsoft SQL server there is a data limitation to the size of a database, and that only 5 years’ worth of JSON data can be written to a database.

### Populate CCAP\_CASE\_SUMMARY\_MASTER (CCAP\_switch == 4)

// if 4 populates CCAP\_CASE\_SUMMARY\_MASTER
insertCaseSumMaster (year, DBconnectionParm)

insertCaseSumMaster () is very simple, it summarizes all the CCAP case data, accumulating the charge information associated with each case. Note that only closed cases are selected, and the case data is summarized for the year that is passed in to the method. The method is composed of a single SQL statement.

The purpose of the SQL statement is to tally the number of bail jumping and non-bail jumping charges that are associated with each case so this information can be used to generate reports. The key data elements of the CCAP\_CASE\_SUMMARY\_MASTER table are described below.

* NumBailJumpDismissed – Number of Bail Jumping charges that were dismissed in the associated case.
* NumBailJumpNotDismissed – Number of Bail Jumping charges that were NOT dismissed in the associated case.
* NumNotBailJumpPleadGuiltyorNC – Number of NON Bail Jumping charges that were plead guilty or no contest to for the associated case.
* NumNotBailJumpPleadNotGuiltyorNC – Number of NON Bail Jumping charges that were NOT plead guilty or no contest to for the associated case.
* AllChargesDismissed – Yes/No value indicating if all the charges have been dismissed. If all charges of a given case are dismissed, then no leverage can be asserted.
* AllChargesBailJumping – Yes/No value indicating if all charges of the case are bail jumping. If all charges of a given case are bail jumping charges, then no leverage can be asserted.

While insertCaseSumMaster () is simple, it takes approximately two hours to execute for each year. The method basically consists of one SQL statement. Bail jumping is determined by the value of the Statute, having a value of either '946.49(1)(a)' or '946.49(1)(b)'. Dismissed charges are determined by the value of the DispCode, having any of the following values: '21', 'DCOM', '36', '82', 'DRI', 'DSMIS', '28', '30'.

Note that this process is run for all years, and takes about 36 hours complete for the years 2000-2016.

The structure of the CCAP\_CASE\_SUMMARY\_MASTER is shown below:

[CCAP\_CASE\_SUMMARY\_MASTER](

 [CaseNum] [varchar](50) NOT NULL,

 [Status] [varchar](50) NULL,

 [CaseType] [varchar](50) NULL,

 [Year] [varchar](50) NULL,

 [County] [varchar](50) NOT NULL,

 [Race] [varchar](50) NULL,

 [CountyCaseNum] [nvarchar](50) NULL,

 [NumBailJumpDismissed] [int] NULL,

 [NumBailJumpNotDismissed] [int] NULL,

 [NumNotBailJumpPleadGuiltyorNC] [int] NULL,

 [NumNotBailJumpPleadNotGuiltyorNC] [int] NULL,

 [AllChargesDismissed] [varchar](50) NULL,

 [AllChargesBailJumping] [varchar](50) NULL

The insert statement that populates the table is shown below:

("INSERT INTO [CCAP].[dbo].[ccap\_case\_summary\_master] " +
"SELECT DISTINCT (a.CaseNum), a.Status, a.CaseType, a.Year, a.County, a.Race, a.County+a.CaseNum, " +

" (Select Count(\*) FROM [CCAP].[dbo].[CCAP\_CASE\_ALL] WHERE CaseNum = a.CaseNum and County = a.County and Status = 'CL' " +

" and Statute in ('946.49(1)(a)', '946.49(1)(b)') and (DispCode1 in ('21', 'DCOM', '36', '82', 'DRI', 'DSMIS', '28', '30') OR " +

" DispCode2 in ('21', 'DCOM', '36', '82', 'DRI', 'DSMIS', '28', '30'))), " +

" (SELECT Count(\*) FROM [CCAP].[dbo].[CCAP\_CASE\_ALL] WHERE CaseNum = a.CaseNum and County = a.County and Status = 'CL' " +

" and Statute in ('946.49(1)(a)', '946.49(1)(b)') and (DispCode1 not in ('21', 'DCOM', '36', '82', 'DRI', 'DSMIS', '28', '30') and " +

" DispCode2 not in ('21', 'DCOM', '36', '82', 'DRI', 'DSMIS', '28', '30'))), " +

" (Select Count(\*) FROM [CCAP].[dbo].[CCAP\_CASE\_ALL] WHERE CaseNum = a.CaseNum and County = a.County and Status = 'CL' " +

" and Statute not in ('946.49(1)(a)', '946.49(1)(b)') and (DispCode1 in ('GAP', 'GGP', 'GNC', '23') OR " +

" DispCode2 in ('GAP', 'GGP', 'GNC', '23'))), " +

" (SELECT Count(\*) FROM [CCAP].[dbo].[CCAP\_CASE\_ALL] WHERE CaseNum = a.CaseNum and County = a.County and Status = 'CL' " +

" and Statute not in ('946.49(1)(a)', '946.49(1)(b)') and (DispCode1 not in ('GAP', 'GGP', 'GNC', '23') AND " +

 " DispCode2 not in ('GAP', 'GGP', 'GNC', '23'))), " +

 " (select DISTINCT ('Y') FROM [CCAP].[dbo].[CCAP\_CASE\_ALL] " +

 " where (SELECT Count(\*) " +

 " FROM [CCAP].[dbo].[CCAP\_CASE\_ALL] " +

 " WHERE CaseNum = a.CaseNum and County = a.County " +

 " and Status = 'CL') = " +

 " (SELECT Count(\*) FROM [CCAP].[dbo].[CCAP\_CASE\_ALL] " +

 " WHERE CaseNum = a.CaseNum and County = a.County " +

 " and Status = 'CL' and (DispCode1 in ('21', 'DCOM', '36', '82', 'DRI', 'DSMIS', '28', '30') OR " +

 " DispCode2 in ('21', 'DCOM', '36', '82', 'DRI', 'DSMIS', '28', '30')))), " +

 " (select DISTINCT ('Y') FROM [CCAP].[dbo].[CCAP\_CASE\_ALL] " +

 " where (SELECT Count(\*) " +

 " FROM [CCAP].[dbo].[CCAP\_CASE\_ALL] " +

 " WHERE CaseNum = a.CaseNum and County = a.County " +

 " and Status = 'CL') = " +

 " (SELECT Count(\*) FROM [CCAP].[dbo].[CCAP\_CASE\_ALL] " +

 " WHERE CaseNum = a.CaseNum and County = a.County and Status = 'CL' " +

 " and Statute in ('946.49(1)(a)', '946.49(1)(b)'))) " +

 " FROM [CCAP].[dbo].[CCAP\_CASE\_ALL] a " +

 " where a.status='CL' and a.year = ? ");

The number of rows inserted into the CCAP\_CASE\_SUMMARY\_MASTER table are shown below. Note that the February 2017 run did not extend 300 searches beyond the maximum limit. This is the reason that the number of rows is greater in the September 2017 run. For 2016 the September number is greater because there were many more closed cases for 2016 in September 2017 than in February 2017.

|  |  |  |
| --- | --- | --- |
| year | Feb 2017 Run | Sept 2017 Run |
| 2000 | 96654 | 97163 |
| 2001 | 99609 | 101000 |
| 2002 | 102694 | 102873 |
| 2003 | 104559 | 105510 |
| 2004 | 105540 | 106430 |
| 2005 | 108249 | 109573 |
| 2006 | 108401 | 108475 |
| 2007 | 103975 | 104444 |
| 2008 | 99624 | 100179 |
| 2009 | 92712 | 93424 |
| 2010 | 88329 | 89426 |
| 2011 | 85540 | 86764 |
| 2012 | 88013 | 88394 |
| 2013 | 87662 | 88291 |
| 2014 | 82800 | 84162 |
| 2015 | 78588 | 82792 |
| 2016 | 48540 | 71907 |

### Populate CCAP\_REPORT\_MASTER (CCAP\_switch == 5)

// if 5 populates CCAP\_REPORT\_MASTER
insertReportMaster (year, county, DBconnectionParm)

insertReportMaster() populates the CCAP\_REPORT\_MASTER table. While the CCAP\_CASE\_SUMMARY table accumulates data at the case level, CCAP\_REPORT\_MASTER accumulates data at the year and county level. This table contains the data used on the CCAP bail jumping reports and graphs that are rolled up to the year/county level. The method is called passing the year, county and database connection. The method executes 29 SQL statements, in 3 different iterations. The first iteration is for all cases regardless of race, the second is for white defendants, and the third is for non-white defendants. After the 29 SQL statements are executed, the data is inserted into the CCAP\_REPORT\_MASTER table.

An example of one of the SQL statements is shown below. It finds the count of all cases where Bail Jumping charges were dismissed while other charges were plead for closed cases in a given year and for a given county.

query = "select count(distinct(casenum)) as BJDismissedAndPlead FROM [CCAP].[dbo].[CCAP\_CASE\_SUMMARY\_MASTER] where "

 + " NumBailJumpDismissed > 0 "

 + " and NumBailJumpNotDismissed = 0 "

 + " and NumNotBailJumpPleadGuiltyorNC > 0 "

 + " and NumNotBailJumpPleadNotGuiltyorNC = 0 "

 + " and (AllChargesDismissed <> 'Y' OR AllChargesDismissed IS NULL) "

 + " and (AllChargesBailJumping <> 'Y' OR AllChargesBailJumping IS NULL) "

 + " and Year = ? and County=? and status = 'CL'";

Note that this process is run for all years, and takes about 7 hours complete for the years 2000-2016.

The structure of the CCAP\_REPORT\_MASTER is shown below:

[CCAP\_REPORT\_MASTER](

 [Year] [varchar](50) NULL,

 [County] [varchar](50) NOT NULL,

 [NumCases] [int] NULL,

 [ALLNOTBJCases] [int] NULL,

 [AllNOTBJCasesAllDis] [int] NULL,

 [NoBJAndNOTPlead] [int] NULL,

 [NoBJAndPlead] [int] NULL,

 [NoBJAndPleadAndNOTPlead] [int] NULL,

 [ERROR\_ROWS] [int] NULL,

 [AllBJCases] [int] NULL,

 [BJAndAllDis] [int] NULL,

 [AllBJ] [int] NULL,

 [AllBJAndAllDis] [int] NULL,

 [BJCasesInPlay] [int] NULL,

 [BJNOTDismissedAndNOTPlead] [int] NULL,

 [BJNOTDismissedAndPlead] [int] NULL,

 [BJNOTDismissedAndPleadAndNOTPlead] [int] NULL,

 [BJDismissedAndNOTPlead] [int] NULL,

 [BJDismissedAndPlead] [int] NULL,

 [BJDismissedAndPleadAndNOTPlead] [int] NULL,

 [BJDismissedAndNOTDismissedAndNOTPlead] [int] NULL,

 [BJDismissedAndNOTDismissedAndPlead] [int] NULL,

 [BJDismissedAndNOTDismissedAndPleadAndNOTPlead] [int] NULL,

 [BJCharges] [int] NULL,

 [BJChargesDismissed] [int] NULL,

 [NOTBailJumpCharges] [int] NULL,

 [NOTBailJumpChargesDimissed] [int] NULL,

 [ALLBJCases2] [int] NULL,

 [ALLBJCasesSomeDis] [int] NULL,

 [More1BailJump] [int] NULL,

 [More4BailJump] [int] NULL,

 [Race] [varchar](10) NULL

# Reports, Graphs and SQL Statements

The following sections contain the spreadsheets and graphs that are contained in the bail jumping paper, and the sql statements that are used to generate the data for the spreadsheets. The name of the sql file is provided in bold (e.g. **REPORT\_CCAP CASE COUNT).**

The sequence below matches the sequence of the bail jumping paper. The page number and a quote from the paper is provided to identify the location.

## Case and Charges Counts

Page 20 - “This consisted of over 1.62 million cases”

**REPORT\_CCAP CASE COUNT**

SELECT count(\*) as Cases

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_SUMMARY\_MASTER] ;

“and 3.28 million charges”

**REPORT\_CCAP CHARGE COUNT**

SELECT count(\*) as Cases

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_ALL] ;

## Bail Jumping Percentage (2000 – 2016)

Page 21 - ”chart below shows how the bail jumping charge percentage has increased”.



**REPORT\_BJ and Not BJ Charges (MASTER TABLE-ALL COUNTIES)**

SELECT year, count(\*) as BJcharges

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_ALL]

 WHERE Status = 'CL'

 and Statute in ('946.49(1)(a)', '946.49(1)(b)')

 group by year

 order by year;

SELECT year, Count(\*) as NOTBailJumpCharges

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_ALL]

 WHERE Status = 'CL'

 and Statute Not in ('946.49(1)(a)', '946.49(1)(b)')

 group by year

 order by year ;

## Bail Jumping Charges Dismissed

Page 22 - “63.85% of all bail jumping charges that were fully adjudicated in 2000 were dismissed”

**REPORT - BJ Charges DISMISSED Over Time**

SELECT year, sum (BJChargesDismissed) as bjDis, Sum (BJCharges) as BJCharges,

(CAST(sum(BJChargesDismissed) AS DECIMAL)/ nullif(sum(BJCharges), 0)) as percentage

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_REPORT\_MASTER]

 where RACE = 'ALL'

 group by (year)

 order by year

Page 23 – “chart below shows the dismissal rate for bail jumping and non-bail jumping charges from 2000 to 2016”



**REPORT - BJ Charges DISMISSED Over Time**

SELECT year, sum (BJChargesDismissed) as bjDis, Sum (BJCharges) as BJCharges,

(CAST(sum(BJChargesDismissed) AS DECIMAL)/ nullif(sum(BJCharges), 0)) as percentage

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_REPORT\_MASTER]

 where RACE = 'ALL'

 group by (year)

 order by year

**REPORT - Non BJ Charges DISMISSED Over Time**

SELECT year, sum (NOTBailJumpChargesDimissed) as Dis, Sum (NOTBailJumpCharges) as Charges, (CAST(sum(NOTBailJumpChargesDimissed) AS DECIMAL)/ nullif(sum(NOTBailJumpCharges), 0)) as percentage

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_REPORT\_MASTER]

 where race = 'ALL'

 group by (year)

 order by year

## Top Ten Charges - Statewide

Page 24 – “Table II-1 below shows the top ten charged offenses in 2000 and 2016”.



**REPORT - Top Charges by County**

SELECT Statute, count (\*) as StatuteCount

from [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_ALL]

where [Status] = 'CL' and year = '2016'

 group by (Statute)

 order by StatuteCount desc

 /\* and county = '25' \*/

## County Level Charges – All Counties

Page 26 – “Table II.2 below provides 2016 charge information for each of the seven counties”



**REPORT\_SELECTED COUNTY CHARGE INFO (7 COUNTIES)**

SELECT county, count(\*) as TOTCharges

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_ALL]

 WHERE County in ('05', '09', '13', '18', '25', '40', '51')

 and year = '2016'

 group by county

 order by county;

SELECT county, count(\*) as BJcharges

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_ALL]

 WHERE County in ('05', '09', '13', '18', '25', '40', '51')

 and year = '2016'

 and Statute in ('946.49(1)(a)', '946.49(1)(b)')

 group by county

 order by county;

SELECT county, count(\*) as BJchargesClosed

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_ALL]

 WHERE Status = 'CL'

 and County in ('05', '09', '13', '18', '25', '40', '51')

 and year = '2016'

 and Statute in ('946.49(1)(a)', '946.49(1)(b)')

 group by county

 order by county;

 SELECT county, count(\*) as BJchargesDismissed

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_ALL]

 WHERE Status = 'CL'

 and County in ('05', '09', '13', '18', '25', '40', '51')

 and year = '2016'

 and Statute in ('946.49(1)(a)', '946.49(1)(b)') and

 (DispCode1 in ('21', 'DCOM', '36', '82', 'DRI', 'DSMIS') OR

 DispCode2 in ('21', 'DCOM', '36', '82', 'DRI', 'DSMIS'))

 group by county

 order by county;

## Bail Jumping Charges Over Time

Page 26 – “graph below shows the bail jump charges dismissed percentage from 2000 to 2016 for the seven selected counties”



**REPORT - BJ Charges DISMISSED Over Time**

SELECT year, sum (BJChargesDismissed) as bjDis, Sum (BJCharges) as BJCharges,

(CAST(sum(BJChargesDismissed) AS DECIMAL)/ nullif(sum(BJCharges), 0)) as percentage

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_REPORT\_MASTER]

 where RACE = 'ALL'

 and county = '51'

 group by (year)

 order by year

**REPORT - Non BJ Charges DISMISSED Over Time**

SELECT year, sum (NOTBailJumpChargesDimissed) as Dis, Sum (NOTBailJumpCharges) as Charges, (CAST(sum(NOTBailJumpChargesDimissed) AS DECIMAL)/ nullif(sum(NOTBailJumpCharges), 0)) as percentage

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_REPORT\_MASTER]

 where race = 'ALL'

 and county = '51'

 group by (year)

 order by year

## Top Ten Charges – Each County

Page 27 – “analyzed the county data to identify the top ten charges in each of the seven counties as illustrated in the tables below”.

(Note, there are seven tables like the one below, from pages 28-34. Only Brown county is shown below)



 **REPORT - Top Charges by County**

SELECT Statute, count (\*) as StatuteCount

from [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_ALL]

where [Status] = 'CL' and year = '2000'

 and county = '05'

 group by (Statute)

 order by StatuteCount desc

## Bail Jumping Leverage – All Counties

Page 36 – “The following chart shows the bail jump leverage percentage for all counties from 2000-2016”



**REPORT - Leverage Over Time**

SELECT year, sum (BJDismissedAndPlead + BJDismissedAndPleadAndNOTPlead +

BJDismissedAndNOTDismissedAndPlead + BJDismissedAndNOTDismissedAndPleadAndNOTPlead) as lev,

 Sum (BJCasesInPlay) as InPlay

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_REPORT\_MASTER]

 WHERE RACE = 'ALL'

 group by (year)

 order by year

## Bail Jumping Leverage – County 2000 to 2016 comparison

Page 36-37 – “The table below provides that data for each of the seven counties selected for both 2000 and 2016”



**REPORT - Leverage Over Time**

SELECT year, sum (BJDismissedAndPlead + BJDismissedAndPleadAndNOTPlead +

BJDismissedAndNOTDismissedAndPlead + BJDismissedAndNOTDismissedAndPleadAndNOTPlead) as lev,

 Sum (BJCasesInPlay) as InPlay

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_REPORT\_MASTER]

 WHERE RACE = 'ALL'

 AND COUNTY = '51'

 group by (year)

 order by year

## Bail Jumping Leverage – County Graph

Page 37-38 – “The following graph shows the bail jumping leverage percentage from 2000 to 2016 for the seven selected counties, as well as the Statewide percentage “



**REPORT - Leverage Over Time**

SELECT year, sum (BJDismissedAndPlead + BJDismissedAndPleadAndNOTPlead +

BJDismissedAndNOTDismissedAndPlead + BJDismissedAndNOTDismissedAndPleadAndNOTPlead) as lev,

 Sum (BJCasesInPlay) as InPlay

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_REPORT\_MASTER]

 WHERE RACE = 'ALL'

 AND COUNTY = '51'

 group by (year)

 order by year

##  Bail Jumping Dismissed, Other Charge(s) not plead – 7 counties

Page 39 – “Table II.3 below shows the number of these situations in 2000 and 2016”



**REPORT - BJ Charges DISMISSED and Other Charges NOT plead Over Time(2000 & 2016 by County)**

/\* BJ Dismissed and NOT Plead \*/

 SELECT county, count(distinct(countycasenum)) as BJDismissedandNOTPlead2000

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_SUMMARY\_MASTER]

 where NumBailJumpDismissed > 0 and

 NumBailJumpNotDismissed = 0 and

 NumNotBailJumpPleadGuiltyorNC = 0 and

 NumNotBailJumpPleadNotGuiltyorNC > 0 and

(AllChargesDismissed <> 'Y' OR AllChargesDismissed IS NULL)

 and

 (AllChargesBailJumping <> 'Y' OR AllChargesBailJumping IS NULL)

 and year = '2000'

 group by county;

/\* BJ Dismissed and NOT Plead \*/

 SELECT county, count(distinct(countycasenum)) as BJDismissedandNOTPlead2016

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_SUMMARY\_MASTER]

 where NumBailJumpDismissed > 0 and

 NumBailJumpNotDismissed = 0 and

 NumNotBailJumpPleadGuiltyorNC = 0 and

 NumNotBailJumpPleadNotGuiltyorNC > 0 and

(AllChargesDismissed <> 'Y' OR AllChargesDismissed IS NULL)

 and

 (AllChargesBailJumping <> 'Y' OR AllChargesBailJumping IS NULL)

 and year = '2016'

 group by county;

## Bail Jumping Dismissed, Other Charge(s) not plead – All counties over time

Page 40 – “The following chart shows this condition for all counties from 2000 to 2016”



REPORT - BJ Charges DISMISSED and Other Charges NOT plead Over Time

/\* BJ Dismissed and NOT Plead \*/

 SELECT year, count(distinct(countycasenum)) as BJDismissedandNOTPlead

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_SUMMARY\_MASTER]

 where NumBailJumpDismissed > 0 and

 NumBailJumpNotDismissed = 0 and

 NumNotBailJumpPleadGuiltyorNC = 0 and

 NumNotBailJumpPleadNotGuiltyorNC > 0 and

(AllChargesDismissed <> 'Y' OR AllChargesDismissed IS NULL)

 and

 (AllChargesBailJumping <> 'Y' OR AllChargesBailJumping IS NULL)

 group by year;

## High Number of Bail Jumping Charges in single case

Page 39 – “ In 2000, 22.78% of closed bail jumping cases had more than one bail jumping charge”



**REPORT\_High Num BJ Charges**

/\* all the CASES \*/

select count(\*) as NumCases

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_SUMMARY\_MASTER]

 WHERE YEAR = '2016' and status = 'CL'

 /\* Tot NON BJ cases\*/

 SELECT count(distinct(countycasenum)) as ALLNOTBJcases

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_SUMMARY\_MASTER]

 where (NumBailJumpDismissed = 0 and

 NumBailJumpNotDismissed = 0)

 AND YEAR = '2016'

 and status = 'CL'

/\* Should never happen \*/

 SELECT count(distinct(countycasenum)) as ERROR\_ROWS

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_SUMMARY\_MASTER]

 where NumBailJumpDismissed = 0 and

 NumBailJumpNotDismissed = 0 and

 NumNotBailJumpPleadGuiltyorNC = 0 and

 NumNotBailJumpPleadNotGuiltyorNC = 0 and

 (AllChargesDismissed <> 'Y' OR AllChargesDismissed IS NULL)

 and

 (AllChargesBailJumping <> 'Y' OR AllChargesBailJumping IS NULL)

 AND YEAR = '2016'

 /\* Tot BJ cases\*/

 SELECT count(distinct(countycasenum)) as ALLBJcases

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_SUMMARY\_MASTER]

 where (NumBailJumpDismissed > 0 or

 NumBailJumpNotDismissed > 0)

 AND YEAR = '2016'

 and status = 'CL'

 /\* BJ cases and all dismissed\*/

SELECT count(distinct(countycasenum)) as bjandalldis

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_SUMMARY\_MASTER]

 where (NumBailJumpDismissed > 0 or

 NumBailJumpNotDismissed > 0)

 and AllChargesDismissed = 'Y'

 and (AllChargesBailJumping <> 'Y' OR AllChargesBailJumping IS NULL)

 AND YEAR = '2016'

 and status = 'CL'

 /\* All BJ cases \*/

 SELECT count(distinct(countycasenum)) as ALLJustBJ

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_SUMMARY\_MASTER]

 WHERE AllChargesBailJumping = 'Y' AND

 (AllChargesDismissed <> 'Y' OR AllChargesDismissed IS NULL) AND

 YEAR = '2016'

 and status = 'CL'

 /\* cases with >1 bail jump charges \*/

 SELECT count(distinct(countycasenum)) as More1BailJump

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_SUMMARY\_MASTER]

 where (NumBailJumpDismissed + NumBailJumpNotDismissed )> 1

 AND YEAR = '2016'

 /\* cases with five or more bail jump charges \*/

 SELECT count(distinct(countycasenum)) as More4BailJump

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_SUMMARY\_MASTER]

 where (NumBailJumpDismissed + NumBailJumpNotDismissed )> 4

 AND YEAR = '2016'

## Excessive Bail Jumping charges in single case

Page 41,42 – “ Table II.5 below provides a sample of the individual closed cases with excessive bail jumping charges in 2016”



CCAP CASE SUMMARY - cases with high num BJ charges

SELECT casenum, county, NumBailJumpDismissed, NumBailJumpNotDismissed

 FROM [CCAP(Sept 2017)].[dbo].[CCAP\_CASE\_SUMMARY\_MASTER]

 where (NumBailJumpDismissed)> 15

 and year = '2016'

 order by NumBailJumpDismissed DESC

# Excel Spreadsheet - CCAP results (September 2017) MASTER

The Excel Spreadsheet “*CCAP results (September 2017) MASTER*” contains all the tables and graphs presented above, and is embedded below.



# Database Structures

## CCAP\_CASE\_ALL

CCAP\_CASE\_ALL

 [CaseNum] [varchar](50) NOT NULL,

 [ChargeNum] [varchar](50) NOT NULL,

 [County] [varchar](50) NOT NULL,

 [Year] [varchar](50) NOT NULL,

 [Status] [varchar](50) NOT NULL,

 [CaseType] [varchar](50) NOT NULL,

 [Statute] [varchar](50) NOT NULL,

 [Description] [varchar](150) NULL,

 [PartyNum] [varchar](50) NULL,

 [PartyType] [varchar](50) NULL,

 [DOB] [varchar](50) NULL,

 [Sex] [varchar](50) NULL,

 [Race] [varchar](50) NULL,

 [PleaCode] [varchar](50) NULL,

 [JdgmtSeqNum] [varchar](50) NULL,

 [DispCode1] [varchar](50) NULL,

 [DispCode2] [varchar](50) NULL

To improve performance, CCAP\_CASE\_ALL has indexes on Year, Status, DispCode1, DispCode2, and Statute, as detailed below:

CREATE NONCLUSTERED INDEX [IX\_CCAP\_CASE\_ALL] ON [dbo].[CCAP\_CASE\_ALL]

( [Year] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

CREATE NONCLUSTERED INDEX [IX\_CCAP\_CASE\_ALL\_1] ON [dbo].[CCAP\_CASE\_ALL]

( [Status] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

CREATE NONCLUSTERED INDEX [IX\_CCAP\_CASE\_ALL\_2] ON [dbo].[CCAP\_CASE\_ALL]

( [DispCode1] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

CREATE NONCLUSTERED INDEX [IX\_CCAP\_CASE\_ALL\_3] ON [dbo].[CCAP\_CASE\_ALL]

( [DispCode2] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

CREATE NONCLUSTERED INDEX [IX\_CCAP\_CASE\_ALL\_4] ON [dbo].[CCAP\_CASE\_ALL]

( [Statute] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

[CCAP].[dbo].[CCAP\_REPORT\_MASTER]

## CCAP\_JSON

CCAP\_JSON

 [CaseNum] [varchar](50) NOT NULL,

 [County] [varchar](50) NOT NULL,

 [Year] [varchar](50) NOT NULL,

 [CaseJSON] [nvarchar](max) NULL,
 [CountyCaseNum] [nvarchar](50) NULL

## CCAP\_ CASE\_SUMMARY\_MASTER

[CCAP\_CASE\_SUMMARY\_MASTER](

 [CaseNum] [varchar](50) NOT NULL,

 [Status] [varchar](50) NULL,

 [CaseType] [varchar](50) NULL,

 [Year] [varchar](50) NULL,

 [County] [varchar](50) NOT NULL,

 [Race] [varchar](50) NULL,

 [CountyCaseNum] [nvarchar](50) NULL,

 [NumBailJumpDismissed] [int] NULL,

 [NumBailJumpNotDismissed] [int] NULL,

 [NumNotBailJumpPleadGuiltyorNC] [int] NULL,

 [NumNotBailJumpPleadNotGuiltyorNC] [int] NULL,

 [AllChargesDismissed] [varchar](50) NULL,

 [AllChargesBailJumping] [varchar](50) NULL

## CCAP\_ REPORT\_MASTER

[CCAP\_REPORT\_MASTER](

 [Year] [varchar](50) NULL,

 [County] [varchar](50) NOT NULL,

 [NumCases] [int] NULL,

 [ALLNOTBJCases] [int] NULL,

 [AllNOTBJCasesAllDis] [int] NULL,

 [NoBJAndNOTPlead] [int] NULL,

 [NoBJAndPlead] [int] NULL,

 [NoBJAndPleadAndNOTPlead] [int] NULL,

 [ERROR\_ROWS] [int] NULL,

 [AllBJCases] [int] NULL,

 [BJAndAllDis] [int] NULL,

 [AllBJ] [int] NULL,

 [AllBJAndAllDis] [int] NULL,

 [BJCasesInPlay] [int] NULL,

 [BJNOTDismissedAndNOTPlead] [int] NULL,

 [BJNOTDismissedAndPlead] [int] NULL,

 [BJNOTDismissedAndPleadAndNOTPlead] [int] NULL,

 [BJDismissedAndNOTPlead] [int] NULL,

 [BJDismissedAndPlead] [int] NULL,

 [BJDismissedAndPleadAndNOTPlead] [int] NULL,

 [BJDismissedAndNOTDismissedAndNOTPlead] [int] NULL,

 [BJDismissedAndNOTDismissedAndPlead] [int] NULL,

 [BJDismissedAndNOTDismissedAndPleadAndNOTPlead] [int] NULL,

 [BJCharges] [int] NULL,

 [BJChargesDismissed] [int] NULL,

 [NOTBailJumpCharges] [int] NULL,

 [NOTBailJumpChargesDimissed] [int] NULL,

 [ALLBJCases2] [int] NULL,

 [ALLBJCasesSomeDis] [int] NULL,

 [More1BailJump] [int] NULL,

 [More4BailJump] [int] NULL,

 [Race] [varchar](10) NULL