



SAMS Database Procedures and Processes Administrator's Guide

Version 1.0

June 2014



Table of Contents

- 1 Introduction3**
 - 1.1 Document Organization 3
 - 1.2 Document Purpose..... 3
 - 1.3 Version Information 4
- 2 Data Cleansing Process.....5**
 - 2.1 Overview..... 5
 - 2.2 SAMS_CET Data..... 5
 - 2.3 SAMS_STAGE_LOAD. COPY_CET_DATA Procedure..... 7
 - 2.4 SAMS_STAGE_LOAD. PRESERVE_CET_DATA Procedure 7
- 3 Analysis Queue8**
 - 3.1 Overview..... 8
 - 3.2 SAMS_ANALYSIS_CUE Table..... 8
 - 3.3 ScheduleAnalysis Procedure..... 8
 - 3.4 AnalysisCueMonitor Procedure 8
 - 3.4.1 ANALYSIS_CUE_MONITOR DBMS_SCHEDULER Job 9
 - 3.4.2 SAMS_ANALYSIS_UTIL.ScheduleAnalysis Procedure 9
 - 3.5 Completing a Queue Run..... 9
- 4 Sliding Scale Process10**
 - 4.1 Overview.....10
 - 4.2 Packages.....10
 - 4.3 Database Tables10
 - 4.4 Database Diagram10
 - 4.5 Workflow10
- 5 Homogeneous Intersection Analysis Process13**
 - 5.1 Overview.....13
 - 5.2 Packages.....13
 - 5.3 Database Tables13
 - 5.4 Database Diagram13



5.5 Workflow13

6 Homogeneous Section Analysis Process16

6.1 Overview.....16

6.2 Packages.....16

6.3 Database Tables16

6.4 Database Diagram16

6.5 Workflow16

Appendix A – Analysis Class Definition Attributes19

Appendix B – Crash Type Percentages20

Appendix C – Intersection Attributes.....21

Appendix D – Intersection Description Attributes22

Appendix E – Crash Indicator Flags23

Appendix F – Section Attributes.....21

Appendix G – Section Description Attributes22

Appendix H – Sliding Scale Crash Type Percentages.....23

Appendix I – Sliding Scale Database Diagram24

Appendix J – Homogeneous Intersection Database Diagram37

Appendix K – Homogeneous Section Database Diagram45





1 Introduction

1.1 Document Organization

This document is separated into 17 sections that are defined below.

- 1.



- 2. Introduction – provides an overview of the contents of this document.
- 3.

1 . 0	June 3, 2014	Matt Hiland	Accept all changes and modify version number.
-------------	--------------	-------------	---



4. Data Cleansing Process – explains the procedures for maintain the cleansed data.
5. Analysis Queue – outlines the procedures that support the queue that manages analyses preventing the contention for system reports.
6. Sliding Scale Process – details the workflow for creating a Sliding Scale Analysis.
7. Homogeneous Intersection Analysis Process – details the workflow for creating a Homogenous Intersection Analysis.
8. Homogeneous Section Analysis Process – details the workflow for creating a Homogeneous Section Analysis.
9. Appendix A – Analysis Class Definition Attributes
10. Appendix B – Crash Type Percentages
11. Appendix C – Intersection Attributes
- 12.



- 13. Appendix D – Intersection Description [Attributes](#)
- 14. Appendix E – Crash Indicator Flags
- 15. Appendix F – Section Attributes
- 16. Appendix G – Section Description Attributes
- 17. Appendix H – Sliding Scale Crash Type Percentages
- 18. Appendix I – Sliding Scale Database Diagram
- 19. Appendix J – Homogeneous Intersection Database Diagram
- 20. Appendix K – Homogeneous Section Database Diagram.

1.2 Document Purpose

The SAMS Database Administrator’s Guide is intended to provide developers and system administrators with an overview of key database procedures and processes. While not an exhaustive and detailed document, it serves to provide those who maintain SAMS with an understanding of the key database pieces and how they fit together.

1.3 Version Information

Version Num.	Edit Date	Edited By	Comments
1.0	June 3, 2014	Matt Hiland	Accept all changes and modify version number.



2 Data Cleansing Process

2.1 Overview

The Crash Editing Tool (CET) is a custom web application, developed by Intergraph, for cleansing crash data. This cleansed data needs to be persisted throughout the staging process. The CET Copy and Preserve procedures were created to address this requirement. The CET Copy procedure moves the CET changes from the SAMS_CET schema to the staging tables. These values are then injected into the staging process at various intervals by the CET Preserve procedure. The CET Preserve procedure is then used to maintain these values in future staging runs. While a complete understanding of the SAMS Data Loading processes is not necessary to support them, it is important to have a basic knowledge of the system components in order to be able to support these from an administration standpoint.

2.2 SAMS_CET Data

The Staging Copy/Preserve routines utilize four tables from the SAMS_CET schema. These tables are listed below as well as displayed in Figure 1 - SAMS CET Tables:

- **CLEANSED_CRASH_RECORD** – Cleansed Crash Record, only columns that have been cleansed and are not null
- **CLEANSED_CRASH_RECORD_COPIED** – Record indicating the copy status of the crash (Copied_Flag → 1 = success, -1 failure, NULL not attempted – will be attempted on the next run)
- **CLEANSED_CRASH_VECHICLE** – Cleansed Vehicle Crash Record, only columns that have been cleansed and are not null
- **CLEANSED_CRASH_VECHICLE_COPIED** – Record indicating the copy status of the vehicle data associated with a crash.

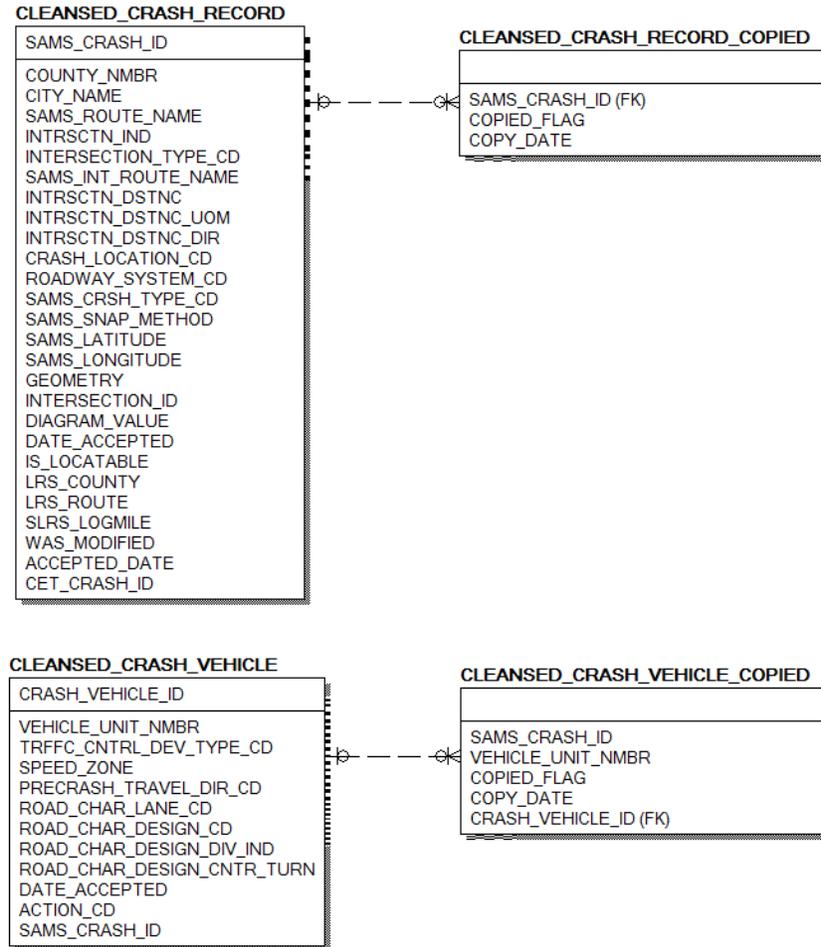


Figure 1 - SAMS CET Tables



2.3 SAMS_STAGE_LOAD.COPY_CET_DATA Procedure

The SAMS_STAGE_LOAD process loads data from the staging tables into the SAMS crash tables on a nightly basis. To maintain the attribute changes of the CET application, the changes made in the SAMS_CET schema need to be copied to the SAMS_STAGE.CRASH and SAMS_STAGE.CRASH_VEHICLE tables for data staging. This is accomplished by the COPY_CET_DATA procedure called from within the SAMS_STAGE_LOAD.STAGE_CRASH procedure in the SAMS_STAGE.SAMS_STAGE_LOAD package. The copy procedure transfers the data in these tables along with the other data to be staged from incoming files. This copy process needs to be completed once for each record. The SAMS_CET.CLEANSSED_CRASH_RECORD_COPIED and SAMS_CET.CLEANSSED_CRASH_VEHICLE_COPIED tables track the status of whether a record has been copied. The copy procedure only copies records from the SAMS_CET.CLEANSSED_CRASH_RECORD and SAMS_CET.CLEANSSED_CRASH_VEHICLE tables where the corresponding crash in the “copied” tables has a NULL value in the “COPIED_FLAG” column. The copy process can be run multiple times on a single record without issue as the staging process handles records already existing in the SAMS.CRASH table (though it’s only required that it be copied once unless there are additional attribute changes). To accomplish this, the record’s COPIED_FLAG column needs to be set to NULL. This must be done manually in the database. Following a run the COPIED_FLAG will either be set to 1 (success) or -1 (failure).

2.4 SAMS_STAGE_LOAD.PRESERVE_CET_DATA Procedure

Once the cleansed data is transferred to the staging tables, it is staged as any other staging data (see the Staging Admin Guide for additional details). This means that some of the columns may be overwritten during the staging process undoing the work done by the cleansing process. The SAMS_STAGE_LOAD.PRESERVE_CET_DATA Procedure is called following the copy procedure and before the snapping process. It is called again following the snapping process so that the data going into the snapping process contains the cleansed data, and to ensure that the data coming out of it preserves the cleansed data. The SAMS_STAGE_LOAD.PRESERVE_CET_DATA Procedure updates the data in the SAMS.CRASH and SAMS.GWM_CRASH_DEFAULT tables.



3 Analysis Queue

3.1 Overview

MDOT’s SAMS application allows users to create custom data analysis result sets. These result sets can take anywhere from a few minutes to several hours to process (currently there is no limit on how long an analysis is allowed to run). In the past, users needed to be careful not to schedule multiple analyses concurrently for fear that they would all fail due to contention for resources, table locking, etc. The Analysis Queue was created to address this problem. While a complete understanding of the SAMS Data Loading processes is not necessary to support them, it is important to have a basic knowledge of the system components in order to be able to support these from an administration standpoint.

3.2 SAMS_ANALYSIS_CUE Table

The SAMS Analysis Queue system is based around the SAMS.SAMS_ANALYSIS_CUE table. This table holds information about each analysis request. The columns of this table are explained below:

- **ANALYSIS_ID** – The ID of the analysis from the corresponding analysis table (Intersection, Section, and Sliding Scale)
- **ANALYSIS_TYPE** – Abbreviated delineation of the analysis type (INT, SEC, and SLI)
- **SUBMISSION_DATE** – Date/Time Stamp of the queue request
- **CUE_START_DATE** – Date/Time Stamp the analysis was actually started from the queue
- **CUE_END_DATE** - Date/Time Stamp the analysis was actually finished from the queue
- **CUE_STATUS** – Current status of the queue request – (99=Running, 1=Success, -1=Failure/Error)

This table can be queried at any time to find if an analysis is currently running or has completed (either successfully or in failure). There should only be one record marked with a status of 99 at any given time.

3.3 ScheduleAnalysis Procedure

The SAMS_ANALYSIS_UTIL.ScheduleAnalysis procedure enters the queue data into the SAMS_ANALYSIS_CUE table. Each of the different Analysis types has a procedure that passes the request to this procedure for scheduling of the analysis. The queue details are inserted into the SAMS_ANALYSIS_CUE table and then the SAMS_HOMOG_LOC_ANALYSIS.AnalysisCueMonitor procedure is called.

3.4 AnalysisCueMonitor Procedure

The SAMS_HOMOG_LOC_ANALYSIS.AnalysisCueMonitor Procedure is called in two ways as defined below.



3.4.1 ANALYSIS_CUE_MONITOR DBMS_SCHEDULER Job

This job is scheduled to run every two minutes, Monday through Friday, from 3 am to 7 pm.

3.4.2 SAMS_ANALYSIS_UTIL.ScheduleAnalysis Procedure

This procedure is called each time a request is made. The SAMS_HOMOG_LOC_ANALYSIS.AnalysisCueMonitor procedure is called to start the process if there are no other analyses currently running (this prevents the user from waiting up to two minutes before the analysis even begins).

The SAMS_HOMOG_LOC_ANALYSIS.AnalysisCueMonitor procedure requires SELECT privileges on the v\$instance Oracle system view. This grant allows the Queue Monitor to determine the last time the database restarted. The Queue Monitor will restart an analysis in the event that the database is restarted while an analysis is running. This prevents the queue from having to be restarted and manipulated after a database restart.

The SAMS Application Staging Processes should not be run while an analysis is running. The SAMS_DD.SAMS_WAREHOUSE_STATUS table has a single record with two columns. The ANALYSIS_RUNNING column is set to 1 while an analysis is running. This will prevent the Staging task from executing that evening.

When the SAMS_HOMOG_LOC_ANALYSIS.AnalysisCueMonitor procedure determines an analysis is ready to be started, it calls the SCHEDULE_JOB.NEW_ANALYSIS_JOB procedure. This procedure creates a DBMS_SCHEDULER job that kicks off the analysis. When the analysis completes, it sets the SAMS_ANALYSIS_CUE.CUE_STATUS column to 1 or -1 depending on the result and the DBMS_SCHEDULER job is dropped.

Each run of the SAMS_HOMOG_LOC_ANALYSIS.AnalysisCueMonitor procedure deletes any records that are over 30 days old in an effort to prevent a large number of queue records from accumulating as they are of little value.

3.5 Completing a Queue Run

Once an analysis is started by the queue, its related analysis procedure will record the success or failure of the run. The corresponding procedures are listed below:

- SAMS_SLIDING_SCALE_ANALYSIS.RunAnalysis
- SAMS_HOMOG_INT_ANALYSIS.RunAnalysis
- SAMS_HOMOG_SEC_ANALYSIS.RunAnalysis

Each of these procedures inserts the completion date/time and the queue status directly into the SAMS_ANALYSIS_CUE table.



4 Sliding Scale Process

4.1 Overview

MDOT’s SAMS application allows users to execute a Sliding Scale Analysis for gap analysis (section analysis), optimization of clusters, and adjustable road lengths. This type of analysis is used to identify clusters or *hotspots* of high crash rates occurring along the highway network. While a complete understanding of the Sliding Scale Process is not necessary to support it, it is important to have a basic knowledge of the system components in order to be able to support this process from an administration standpoint.

4.2 Packages

- SAMS_ANALYSIS_UTIL – Utility routines that support all analysis types
- SAMS_HOMOG_LOC_ANALYSIS – routines that service both intersection and section analyses
- SAMS_SLIDING_SCALE_ANALYSIS – routines that are specific to the Sliding Scale analysis
- SAMS_GEOM_UTIL – routines to maintain spatial data
- SAMS_UTIL – generic routines used by the SAMS application

4.3 Database Tables

Schema	Table Name	Description
SAMS_DD	SAMS_WAREHOUSE_STATUS	Flag indicating if an analysis is running
SAMS	SAMS_ANALYSIS_REQ	XML definition for the analysis and start/end dates
SAMS	SS_ANALYSIS	Basic sliding scale analysis information
SAMS	SAMS_ANALYSIS_SHARE	Defines which groups can access the analysis
SAMS	SS_CLUSTER_CRASHES	Crash location information for the sliding scale analysis
SAMS	SS_CLUSTERS	Sliding scale cluster information for the analysis
SAMS	V_SS_CRASH_ROAD	Sliding scale routes
SAMS	CRASH	Crash information

4.4 Database Diagram

Please refer to Appendix I – Sliding Scale Database Diagram.

4.5 Workflow

- When an analysis is created, it is assigned an Analysis ID by the scheduler. This Analysis ID is then passed to the RunAnalysis stored procedure in the SAMS_SLIDING_SCALE_ANALYSIS package.
- Set the analysis active flag notifying other system processes that an analysis is currently running.



- Get the cluster criteria defined by the user for the analysis including:
 - Cluster Length and Unit of Measure
 - Cluster Tolerance Distance and Unit of Measure
 - Minimum Number of Crashes
- In case the analysis was run previously, purge records from the following tables:
 - SS_ANALYSIS
 - SS_CLUSTER_CRASHES
 - SS_CLUSTERS
- Insert the user groups that can access the analysis into SAMS_ANALYSIS_SHARE. These groups were specified by the analysis when it was saved and information posted to the SAMS_ANALYSIS_REQ table in the XML definition.
- Retrieve the analysis definition including:
 - Geographic area of interest
 - Title
 - Description
 - Default analysis indicator
 - Create date
 - Created by username
 - Analysis run start and end time stamps
- Insert the analysis information into SS_ANALYSIS.
- Select the set of routes for the area of interest defined for the analysis.
- Generate clusters for each route inserting cluster records into SS_CLUSTERS and crash records into SS_CLUSTER_CRASHES. For details regarding the clustering algorithm, please refer to Section 11.14 Sliding Scale Analysis in the SAMS Phase II Detailed Design for Production Release document.
- Build the geometry for the Sliding Scale clusters using the ROUTE_MEASURE procedure in the LRSX package. This package is found in the TDSYS schema and was created by TransDecisions.
- Compute the percentages for each cluster for each of the crash indicators listed in Appendix H – Sliding Scale Crash Type Percentages.
- In addition to the crash percentages, the following information is posted for each cluster:
 - Fatal Crash Count
 - Number of Fatalities
 - Number of Crashes by Injury Severity (A, B, C, and PDO)



- Functional Class
- Route Name
- DUI Count
- AADT
- Severity Index
- Crash Rate
- Rural Indicator
- Populate the HCL Score and Rank fields for each cluster record:
 - Crash Rate Score
 - Crash Count Score
 - Severity Index Score
 - Total Score
 - Overall Rank
- Update the analysis record marking it as completed.
- Update the status in the analysis queue setting it to complete.
- Set the analysis active flag to zero notifying other system processes that an analysis is not currently running.
- An email is then sent to the user notifying them that the analysis has completed either successfully or unsuccessfully.



5 Homogeneous Intersection Analysis Process

5.1 Overview

MDOT’s SAMS application allows users to execute a Homogeneous Intersection Analysis. SAMS provides users with the ability to set the parameters for homogeneous intersections and calculate statewide aver crash rates based on those definitions. While a complete understanding of the Homogeneous Intersection Analysis Process is not necessary to support it, it is important to have a basic knowledge of the system components in order to be able to support this process from an administration standpoint.

5.2 Packages

- SAMS_ANALYSIS_UTIL – Utility routines that support all analysis types
- SAMS_HOMOG_LOC_ANALYSIS – routines that service both homogeneous intersection and section analyses
- SAMS_HOMOG_INT_ANALYSIS – routines that are specific to the homogeneous intersection analysis

5.3 Database Tables

Schema	Table Name	Description
SAMS_DD	SAMS_WAREHOUSE_STATUS	Flag indicating if an analysis is running
SAMS	SAMS_ANALYSIS_REQ	XML definition for the analysis and start/end dates
SAMS	SAMS_ANALYSIS_SHARE	Defines which groups can access the analysis
SAMS	SAMS_ANALYSIS_MSG	Error messages logged while running the analysis
SAMS	SAMS_H_INT_ANALYSIS	Basic intersection analysis information
SAMS	SAMS_H_INT_CLASS	Homogeneous Intersection Class information for the analysis
SAMS	SAMS_H_LOC_CRASH	Crash location information for the Homogeneous Intersection analysis
SAMS	SAMS_H_INT	Homogeneous Intersection information for the analysis
SAMS	SAMS_INTERSECTION	Intersection information
SAMS	CRASH	Crash information

5.4 Database Diagram

Please refer to Appendix J – Homogeneous Intersection Database Diagram.

5.5 Workflow

- When an analysis is created, it is assigned an Analysis ID by the scheduler. This Analysis ID is then passed to the RunAnalysis stored procedure in the SAMS_HOMOG_INT_ANALYSIS package.



- Set the analysis active flag notifying other system processes that an analysis is currently running.
- Retrieve the analysis definition including:
 - Number of classes defined
 - Geographic area of interest
 - K-value
 - Title
 - Description
 - Default analysis indicator
 - Create date
 - Created by username
 - Analysis run start and end time stamps
- Classes are defined based on the attributes listed in Appendix A – Analysis Class Definition Attributes.
- If the default indicator is set, clear the indicator for the previous default.
- In case the analysis was run previously, purge records from the following tables:
 - SAMS_H_INT_ANALYSIS
 - SAMS_ANALYSIS_SHARE
 - SAMS_ANALYSIS_MSG
 - SAMS_H_INT_CLASS
 - SAMS_H_LOC_CRASH
 - SAMS_H_INT
- Insert the analysis information into SAMS_H_INT_ANALYSIS.
- Insert the user groups that can access the analysis into SAMS_ANALYSIS_SHARE. These groups were specified by the analysis when it was saved and information posted to the SAMS_ANALYSIS_REQ table in the XML definition.
- Analyze each defined class as follows:
 - Get an ID for the homogeneous class record from the sequence.
 - Express the class definition in terms of a WHERE clause.
 - Compute the average crash rate for the class based on the class definition, geographical area of interest, and start/end dates.
 - Compute the percentages for the class for each of the crash indicators listed in Appendix B – Crash Type Percentages.
 - Insert the class-level information for the analysis into SAMS_H_INT_CLASS.



- Analyze each intersection as follows:
 - Retrieve the intersection information for each attribute listed in Appendix C – Intersection Attributes.
 - Compute the Severity Index, Critical Crash Rate, and Critical Crash Rate Ratio. Please refer to the Detailed Design document for an explanation of these computations.
 - Compute the percentages for the intersection for each of the crash indicators listed in Appendix B – Crash Type Percentages.
 - Get an ID for the homogeneous intersection record from the sequence.
 - Insert the intersection-level information for the analysis into SAMS_H_INT.
 - Insert a record for each crash that occurred at the intersection into SAMS_H_LOC_CRASH linking crashes to the intersection for the analysis.
- Populate the description fields based on codes values for each attribute listed in



- Appendix D – Intersection Description [Attributes](#).
- Populate the number of fatalities and injuries for each intersection based on data from the SAMS_H_LOC_CRASH and CRASH tables.
- Populate the AADT Source field based on data from SAMS_INTERSECTION.
- Set the crash indicator flags in SAMS_H_LOC_CRASH for each attribute listed in Appendix E – Crash Indicator Flags.
- Populate the HCL Score and Rank fields for each homogeneous intersection record:
 - Crash Rate Score
 - Crash Count Score
 - Severity Index Score
 - Total Score
 - Overall Rank
 - Rank Within Class
- Update the analysis record marking it as completed.
- Update the status in the analysis queue setting it to complete.
- Set the analysis active flag to zero notifying other system processes that an analysis is not currently running.
- An email is then sent to the user notifying them that the analysis has completed either successfully or unsuccessfully.



6 Homogeneous Section Analysis Process

6.1 Overview

MDOT’s SAMS application allows users to execute a Homogeneous Section Analysis. SAMS provides users with the ability to set the parameters for homogeneous sections and calculate statewide aver crash rates based on those definitions. While a complete understanding of the Homogeneous Section Analysis Process is not necessary to support it, it is important to have a basic knowledge of the system components in order to be able to support this process from an administration standpoint.

6.2 Packages

- SAMS_ANALYSIS_UTIL – Utility routines that support all analysis types
- SAMS_HOMOG_LOC_ANALYSIS – routines that service both homogeneous intersection and section analyses
- SAMS_HOMOG_SEC_ANALYSIS – routines that are specific to the homogeneous section analysis

6.3 Database Tables

Schema	Table Name	Description
SAMS_DD	SAMS_WAREHOUSE_STATUS	Flag indicating if an analysis is running
SAMS	SAMS_ANALYSIS_REQ	XML definition for the analysis and start/end dates
SAMS	SAMS_H_SEC_ANALYSIS	Basic section analysis information
SAMS	SAMS_ANALYSIS_SHARE	Defines which groups can access the analysis
SAMS	SAMS_ANALYSIS_MSG	Error messages logged while running the analysis
SAMS	SAMS_H_SEC_CLASS	Homogeneous Section Class information for the analysis
SAMS	SAMS_H_LOC_CRASH	Crash location information for the Homogeneous Section analysis
SAMS	SAMS_H_SEC	Homogeneous Section information for the analysis
SAMS	ROAD_SEGMENT_H	Road segment information
SAMS	V_ROUTE_SEGMENT	Route Segment Begin/End mileage and length
SAMS	CRASH	Crash information

6.4 Database Diagram

Please refer to Appendix K – Homogeneous Section Database Diagram.

6.5 Workflow

- When an analysis is created, it is assigned an Analysis ID by the scheduler. This Analysis ID is then passed to the RunAnalysis stored procedure in the SAMS_HOMOG_SEC_ANALYSIS package.



- Set the analysis active flag notifying other system processes that an analysis is currently running.
- Retrieve the analysis definition including:
 - Number of classes defined
 - Geographic area of interest
 - K-value
 - Title
 - Description
 - Default analysis indicator
 - Create date
 - Created by username
 - Analysis run start and end time stamps
- Classes are defined based on the attributes listed in Appendix A – Analysis Class Definition Attributes.
- If the default indicator is set, clear the indicator for the previous default.
- In case the analysis was run previously, purge records from the following tables:
 - SAMS_H_SEC_ANALYSIS
 - SAMS_ANALYSIS_SHARE
 - SAMS_ANALYSIS_MSG
 - SAMS_H_SEC_CLASS
 - SAMS_H_LOC_CRASH
 - SAMS_H_SEC
- Insert the analysis information into SAMS_H_SEC_ANALYSIS.
- Insert the user groups that can access the analysis into SAMS_ANALYSIS_SHARE. These groups were specified by the analysis when it was saved and information posted to the SAMS_ANALYSIS_REQ table in the XML definition.
- Analyze each defined class as follows:
 - Get an ID for the homogeneous class record from the sequence.
 - Express the class definition in terms of a WHERE clause.
 - Identify the sections for the class by selecting the segments from ROAD_SEGMENT_H based on the class and geography definitions.
 - Get the segment definitions from V_ROUTE_SEGMENT based on the segment id concatenating neighboring segments for each route.



- Compute the average crash rate for the class based on the class definition, geographical area of interest, and start/end dates.
- Compute the percentages for the class for each of the crash indicators listed in Appendix B – Crash Type Percentages
- Insert the class-level information for the analysis into SAMS_H_SEC_CLASS.
- Analyze each section as follows:
 - Retrieve the section information for each attribute listed in Appendix F – Section Attributes
 - Compute the Severity Index, Critical Crash Rate, and Critical Crash Rate Ratio. Please refer to the Detailed Design document for an explanation of these computations.
 - Compute the percentages for the section for each of the crash indicators listed in Appendix B – Crash Type Percentages.
 - Get an ID for the homogeneous section record from the sequence.
 - Insert the section-level information for the analysis into SAMS_H_SEC.
 - Insert a record for each crash that occurred on the section into SAMS_H_LOC_CRASH linking crashes to the section for the analysis.
- Populate the description fields based on codes values for each attribute listed in Appendix G – Section Description Attributes.
- Set the crash indicator flags in SAMS_H_LOC_CRASH for each attribute listed in Appendix E – Crash Indicator Flags.
- Populate the HCL Score and Rank fields for each homogeneous intersection record:
 - Crash Rate Score
 - Crash Count Score
 - Severity Index Score
 - Total Score
 - Overall Rank
 - Rank Within Class
- Update the analysis record marking it as completed.
- Update the status in the analysis queue setting it to complete.
- Set the analysis active flag to zero notifying other system processes that an analysis is not currently running.
- An email is then sent to the user notifying them that the analysis has completed either successfully or unsuccessfully.



Appendix A – Analysis Class Definition Attributes

These attributes may be used in defining classes for analysis:

- AADT
- Number of Lanes
- Signalized
- Through Lane Width
- Route Class
- Functional Class
- Speed Limit
- Divided/Undivided
- Urban/Rural.



Appendix B – Crash Type Percentages

Percentages are computed for the following types of crashes:

- Run off Road (Right, Left, Straight)
- Overturn
- Fell from vehicle
- Other in road
- Pedestrian
- Parked vehicle
- Train
- Bicycle
- Animal
- Fixed Object
- Other Object
- Rear end slow or stop
- Rear end turn
- Left turn same roadway
- Turn cross traffic (Left, Right)
- Head on
- Sideswipe
- Angle
- Other
- Hit and Run
- Jackknife
- Deer.



Appendix C – Intersection Attributes

These attributes are used to describe each intersection:

- SAMS Route Name
- SAMS Intersecting Route Name
- County ID
- Logmile
- AADT Leg 1 - 5
- Route Class
- Functional Class
- Speed Limit
- Maintenance District
- City FIPS
- Divided
- Left Turn Lane
- Urban/Rural
- Lane Width
- Number of Lanes
- Signalized
- SLRS Logmile
- SLRS Route ID
- Total Intersection Volume
- Crash Count
- Crash Rate
- Fatal Crash Count/Percentage
- Injury Crash Count/Percentage
- Wet Pavement Rating
- Darkness Rating
- Number of Daylight Crashes
- Number of DUI Crashes
- Number of Crashes by Injury Severity (A, B, C, Total, and PDO)
- Geometry.



Appendix D – Intersection Description Attributes

These attributes are descriptions assigned to coded intersection attributes:

- Number of Lanes Description
- Route Class Description
- Divided Description
- Urban/Rural Description
- Signalized Description
- Functional Class Description
- Left Turn Lane Indicator Description
- County Name
- Speed Limit
- Through Lane Width.



Appendix E – Crash Indicator Flags

These indicators are tacked for each crash:

- Night
- Day
- Multiple Vehicles
- Older Driver
- Younger Driver
- Right Turn
- Speed-Related
- Speed-Related (Daylight, Night, Dry, Wet)
- Wet
- Night Wet
- Pedestrian
- All Turns
- Bicycle
- Emergency Vehicle
- Head-On
- Run Off Road
- Overturned Vehicle
- Left Turn
- Right Angle
- Read-End
- Sideswipe
- All Crash Types
- Fixed Object
- Run Off Road (Right, Left, Straight)
- Fell From Vehicle
- Other In Road
- Parked Vehicle
- Train
- Animal
- Other Object
- Rear End Slow Stop
- Rear End Turn
- Left Turn Same Roadway
- Left Turn Cross Traffic
- Right Turn Cross Traffic
- Angle
- MUCR Other
- Hit and Run
- Jackknife
- Deer
- Other
- Animal
- Parking
- Head-On Sideswipe
- Snow
- Truck-Related
- Wet Pavement
- Crash Type Code
- Crash Severity
- Class.



Appendix F – Section Attributes

These attributes are used to describe each section:

- Section ID (based on sequence)
- Route Name
- Begin Logmile
- End Logmile
- Section Length
- Total AADT
- Minimum/Maximum AADT
- Route Class
- Functional Class
- Speed Limit
- Divided
- Urban/Rural
- Lane Width
- Shoulder Width
- Median Width
- Number of Lanes
- Signalized
- Crash Count
- Crash Rate
- Fatal Crash Count/Percentage
- Number of Fatalities
- Injury Crash Count/Percentage
- Number of Injuries
- Wet Pavement Rating
- Darkness Rating
- Number of Daylight Crashes
- Number of DUI Crashes
- Number of Crashes by Injury Severity (A, B, C, Total, and PDO)
- Geometry.



Appendix G – Section Description Attributes

These attributes are descriptions assigned to coded section attributes:

- Number of Lanes Description
- Route Class Description
- Divided Description
- Urban/Rural Description
- Functional Class Description
- Speed Limit
- Median Width
- Shoulder Width
- Lane Width.



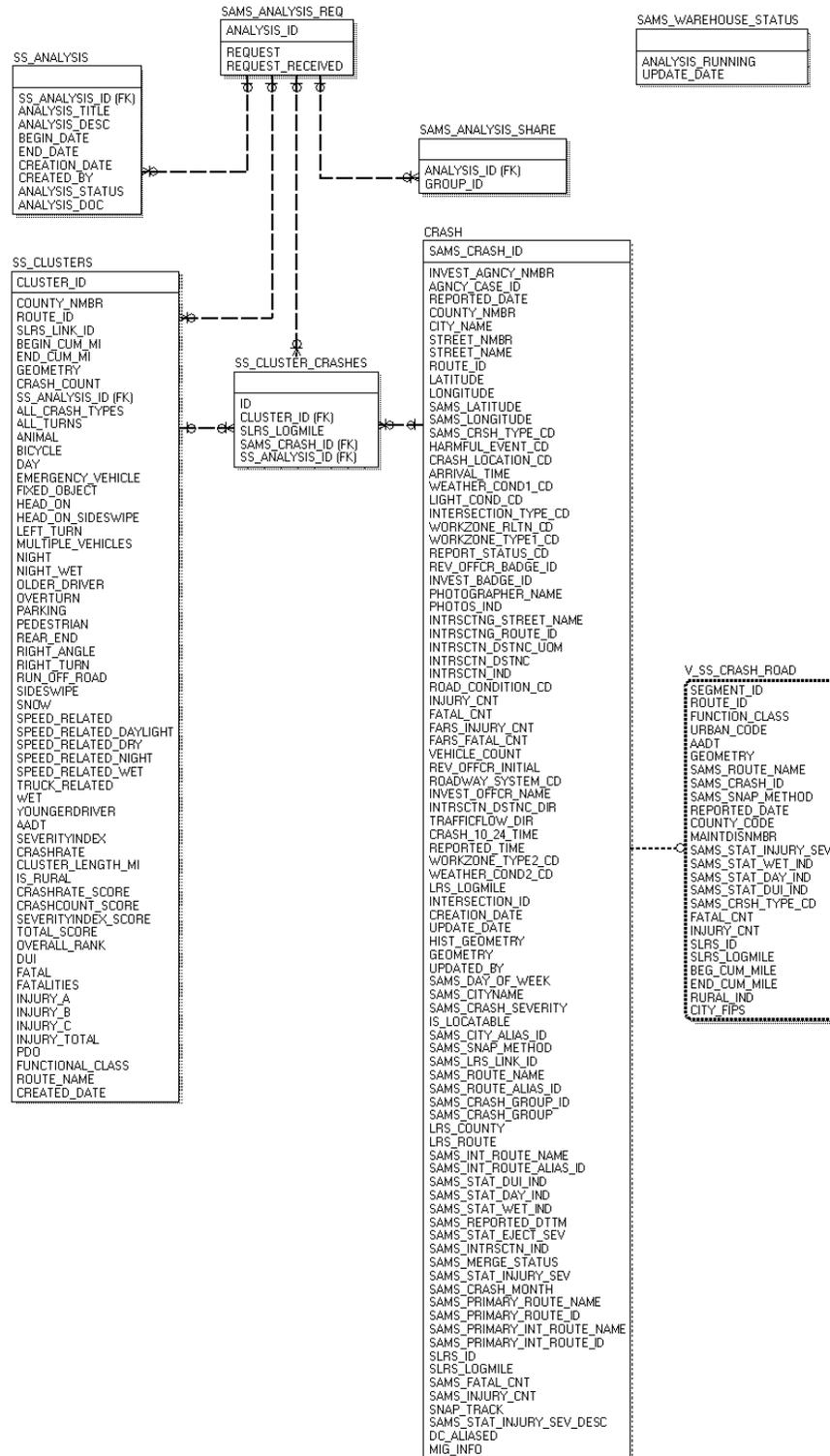
Appendix H – Sliding Scale Crash Type Percentages

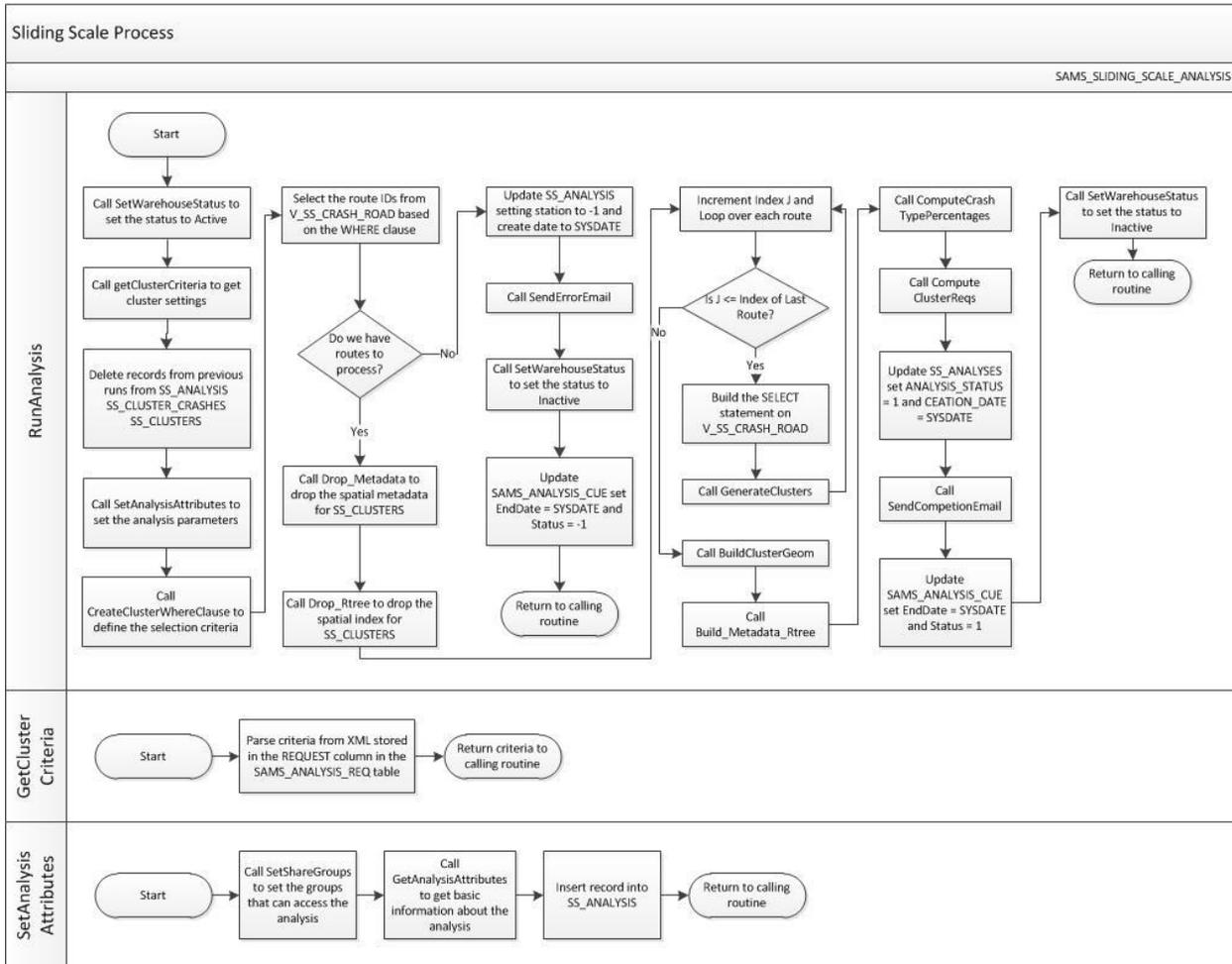
Percentages are computed for each cluster for the following types of crashes:

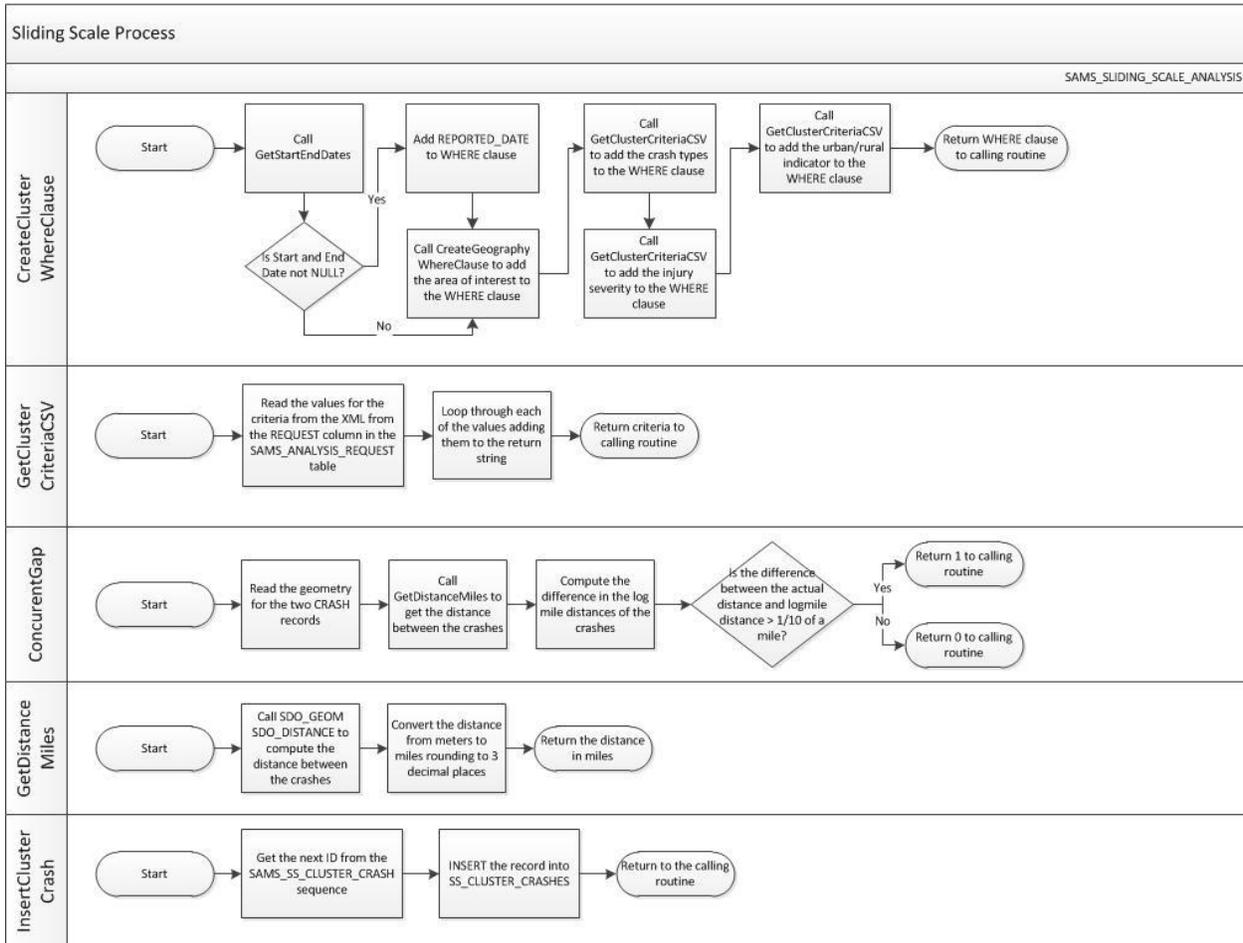
- All Crash Types
- All Turns
- Animal
- Bicycle
- Day
- Emergency Vehicle
- Fixed Object
- Head On
- Head On Sideswipe
- Left Turn
- Multiple Vehicles
- Night
- Night Wet
- Older Driver
- Overturned Vehicle
- Parking
- Pedestrian
- Rear End
- Right Angle
- Right Turn
- Run Off Road
- Sideswipe
- Snow
- Speed-Related
- Speed-Related (Daylight, Dry, Night, Wet)
- Truck-Related
- Wet
- Younger Driver.

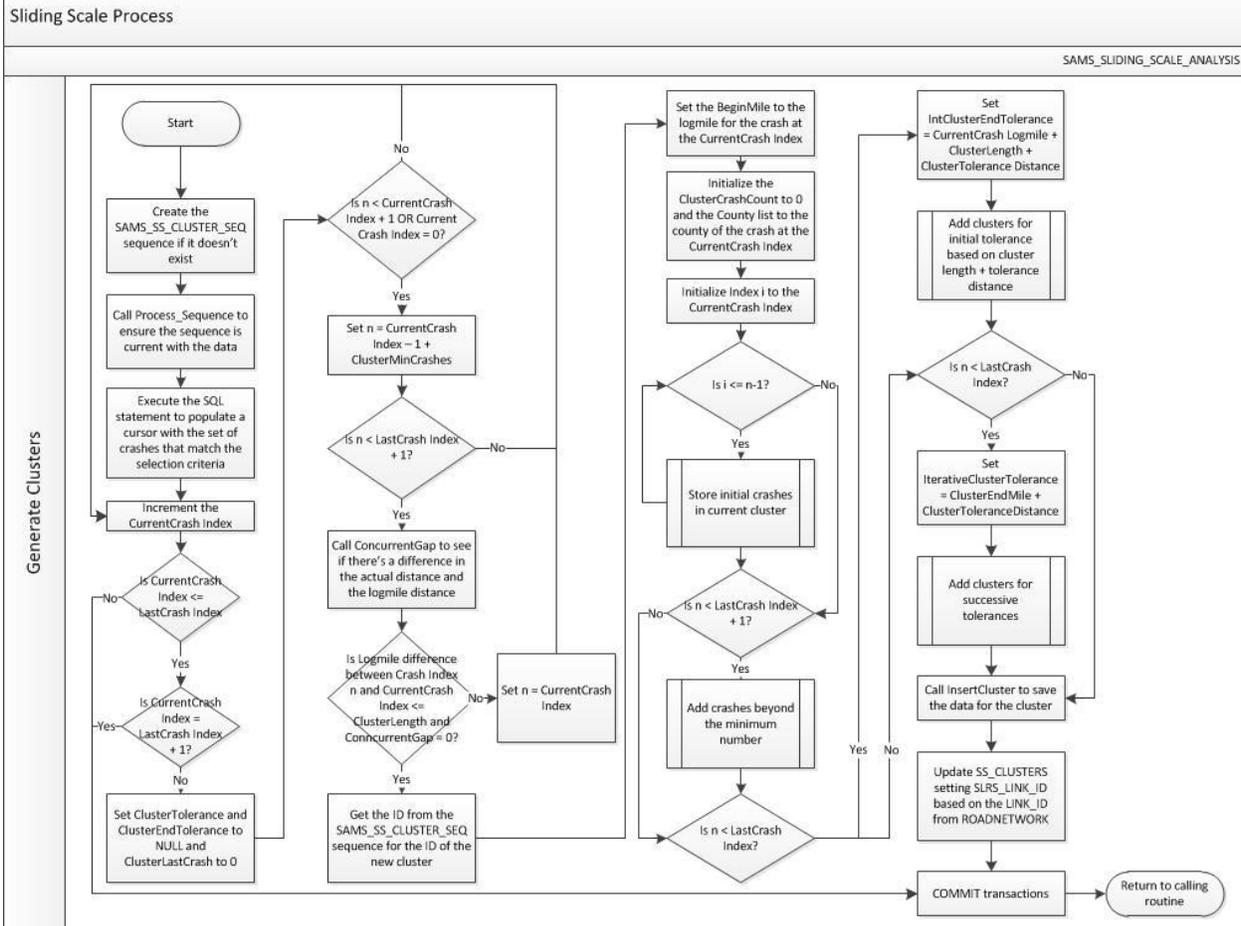


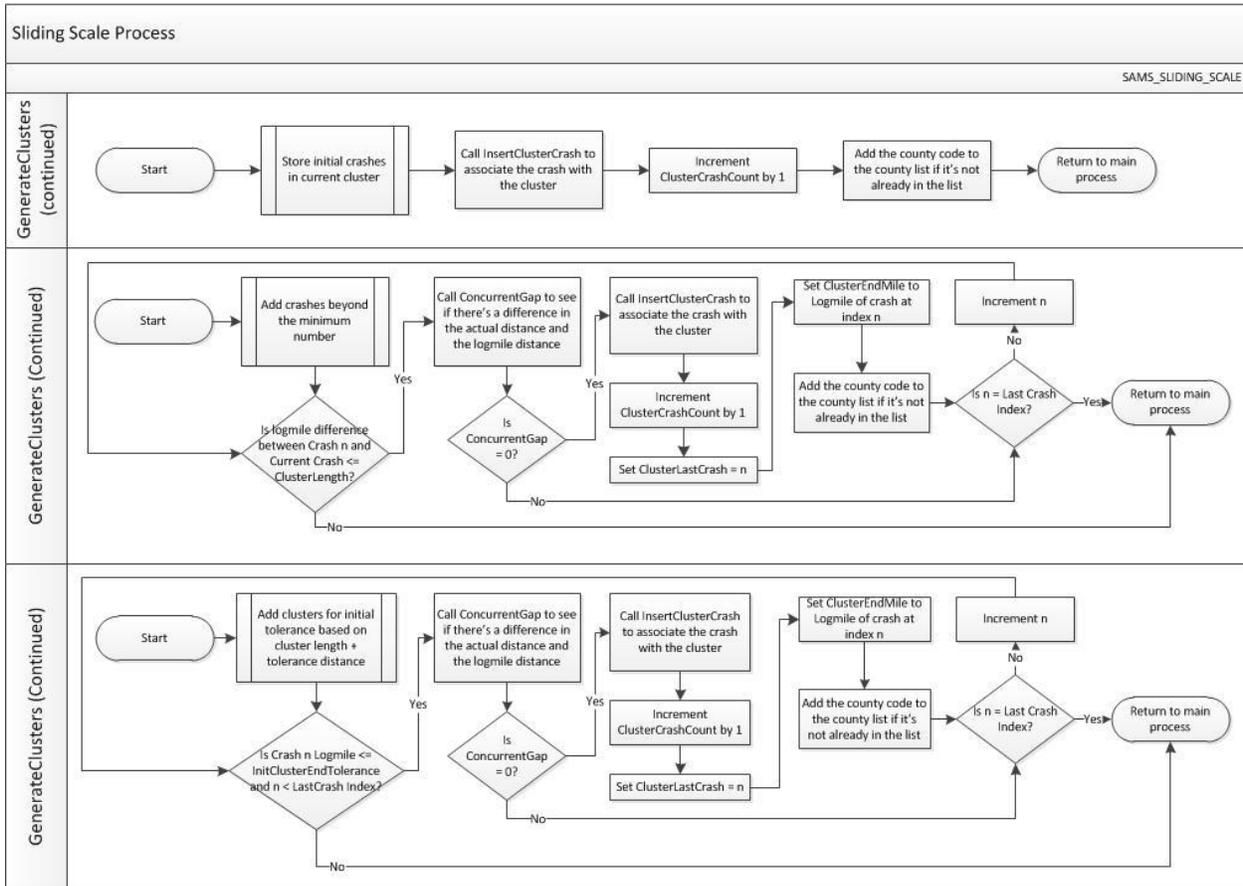
Appendix I – Sliding Scale Database Diagram

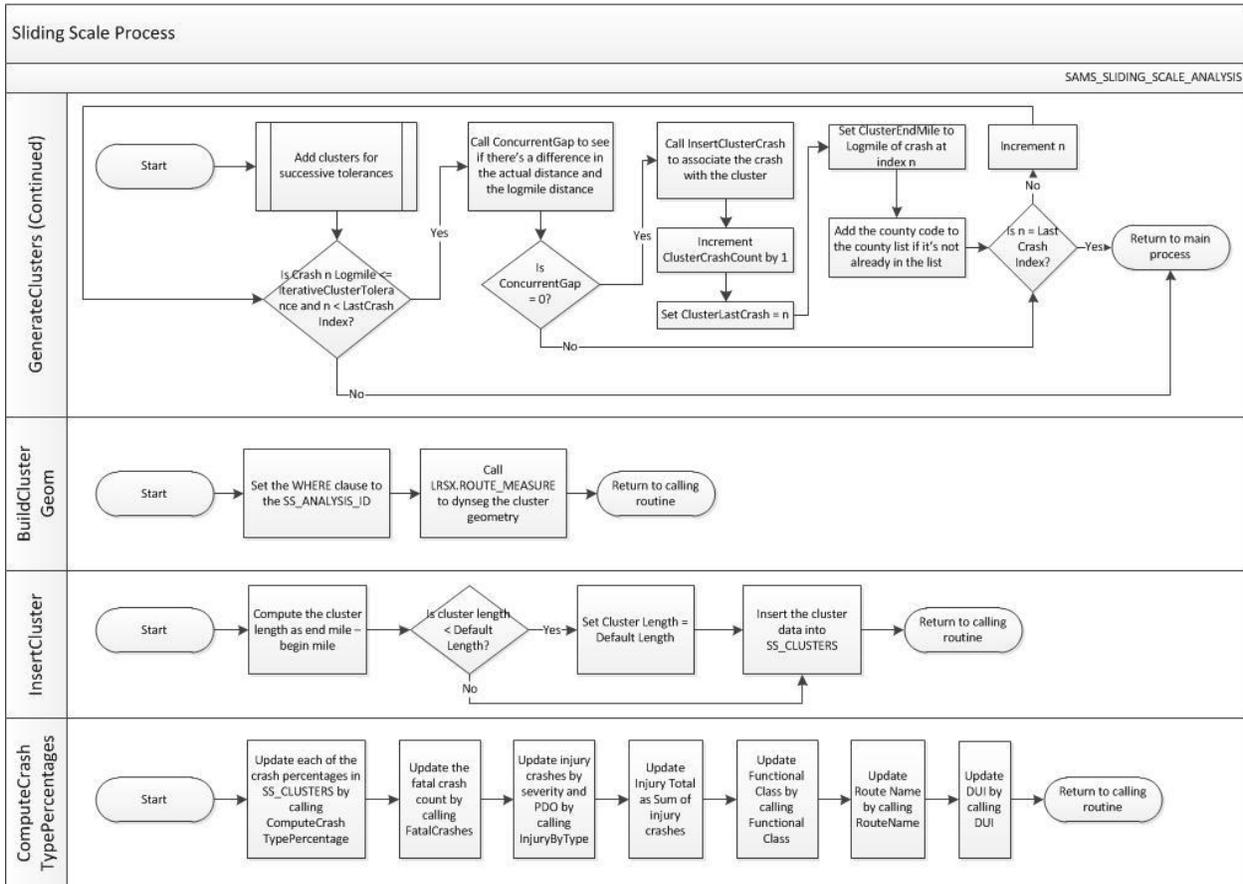






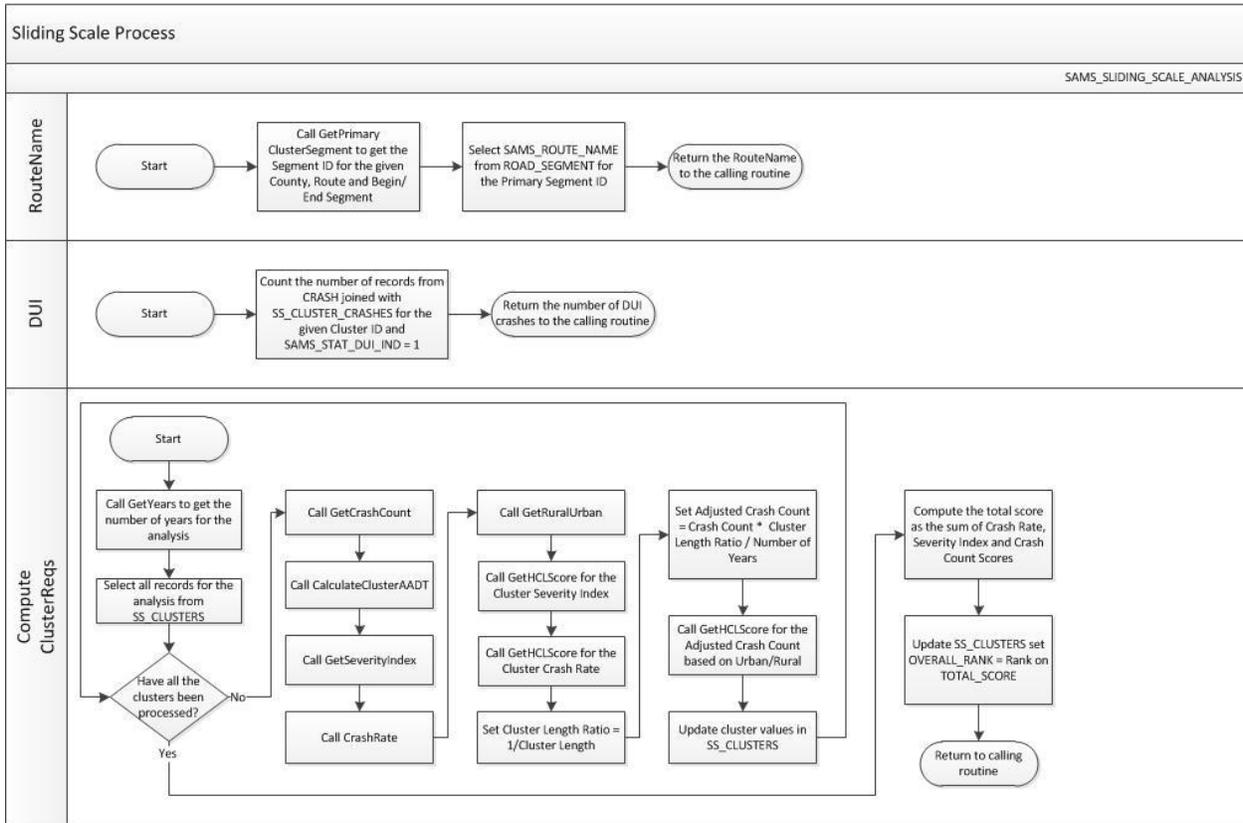


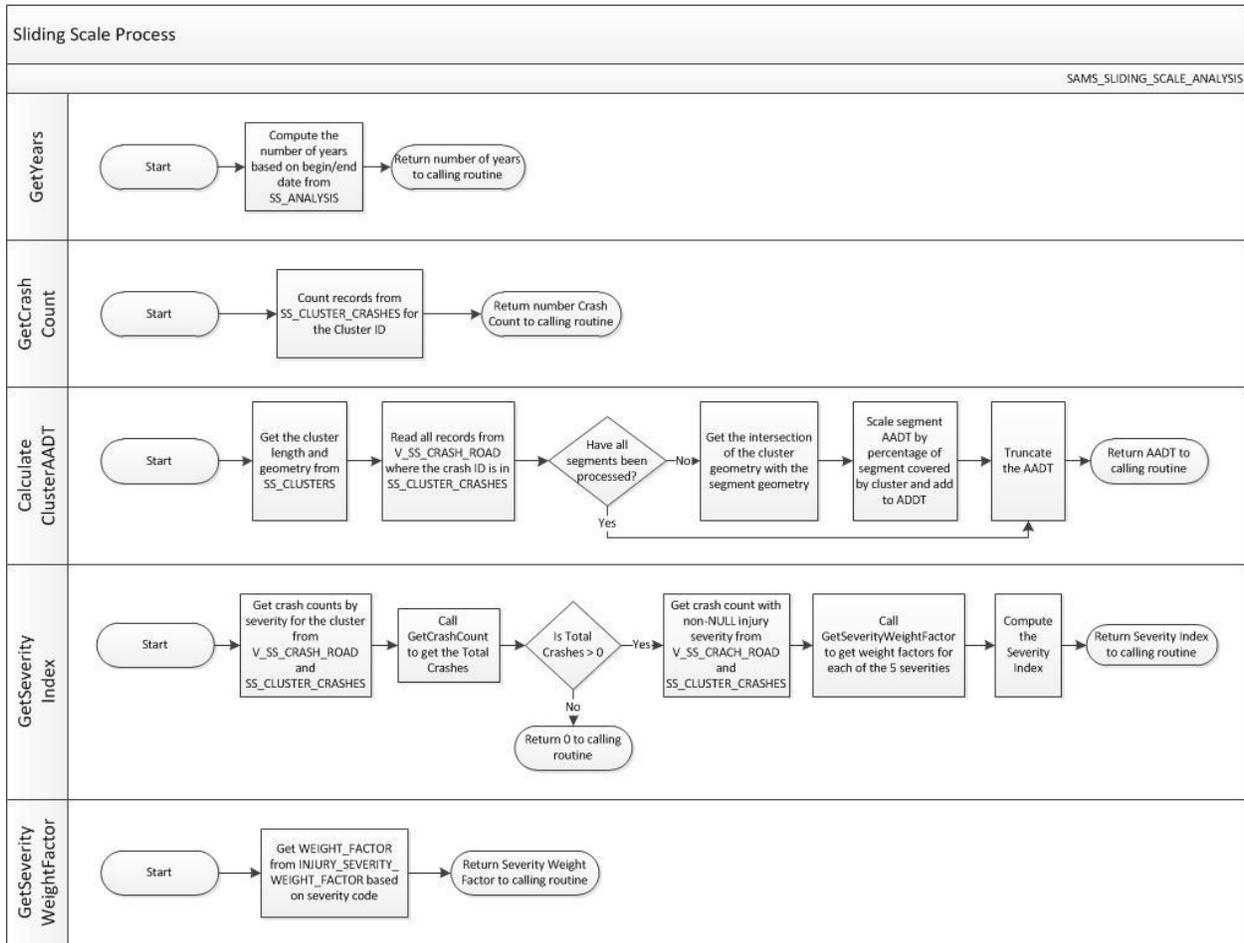


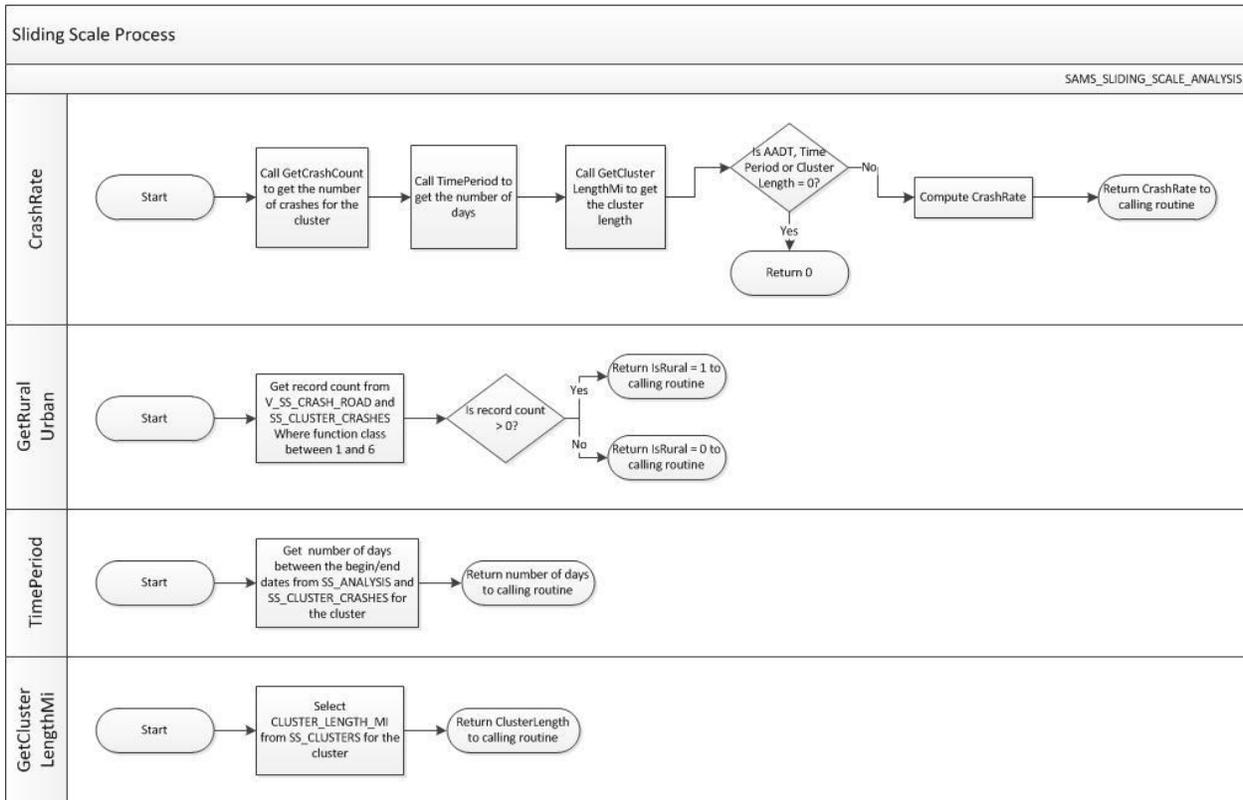


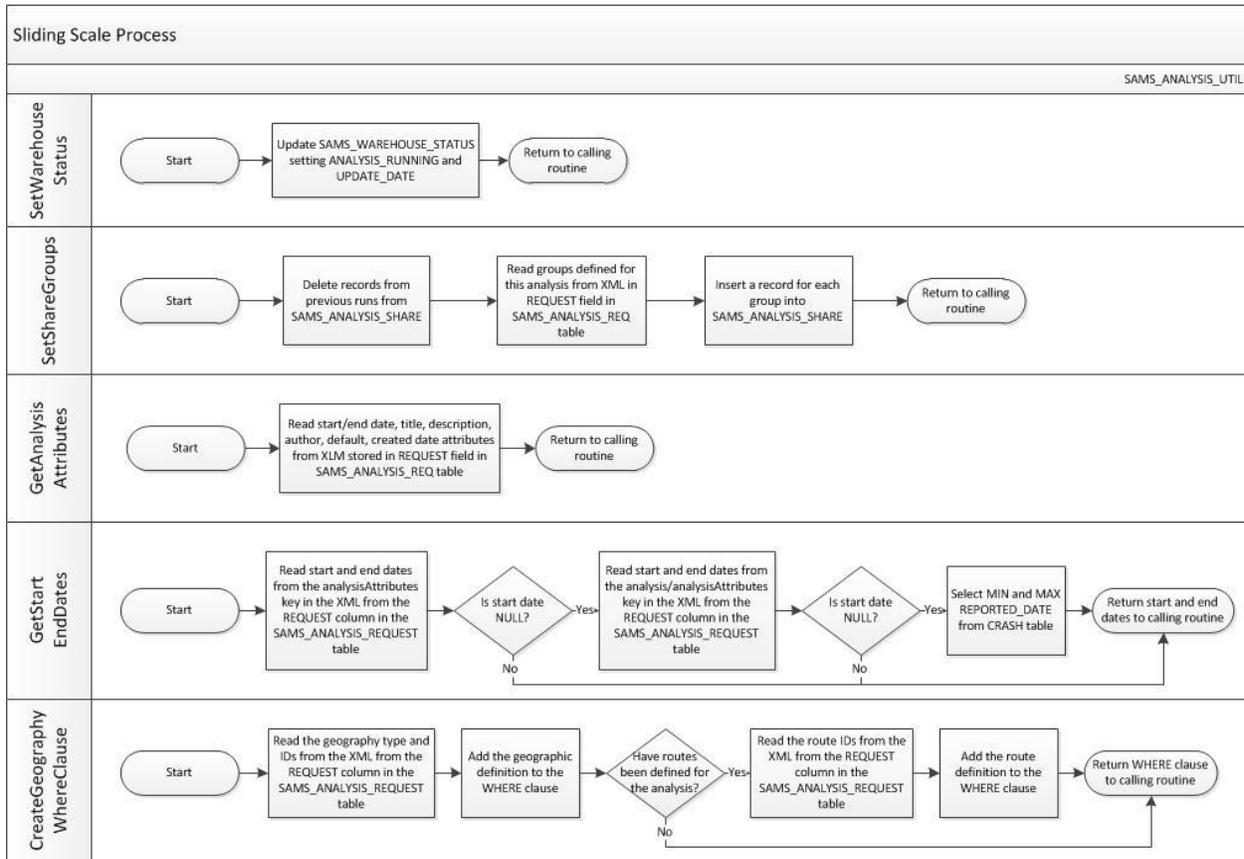


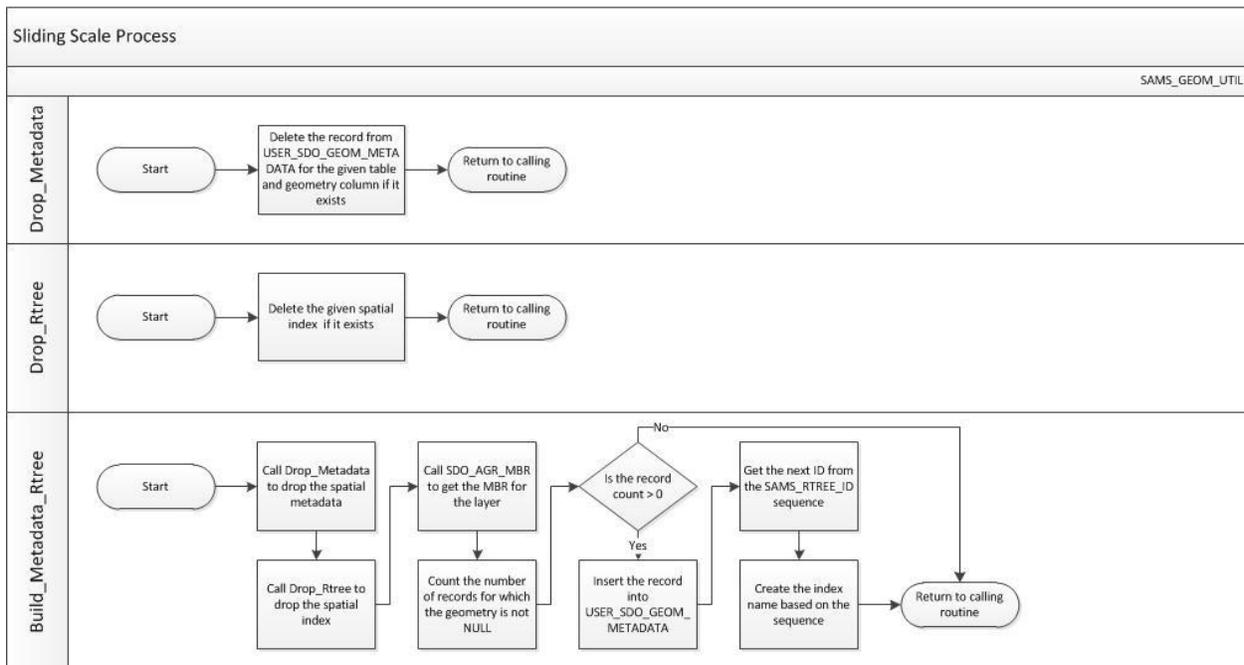
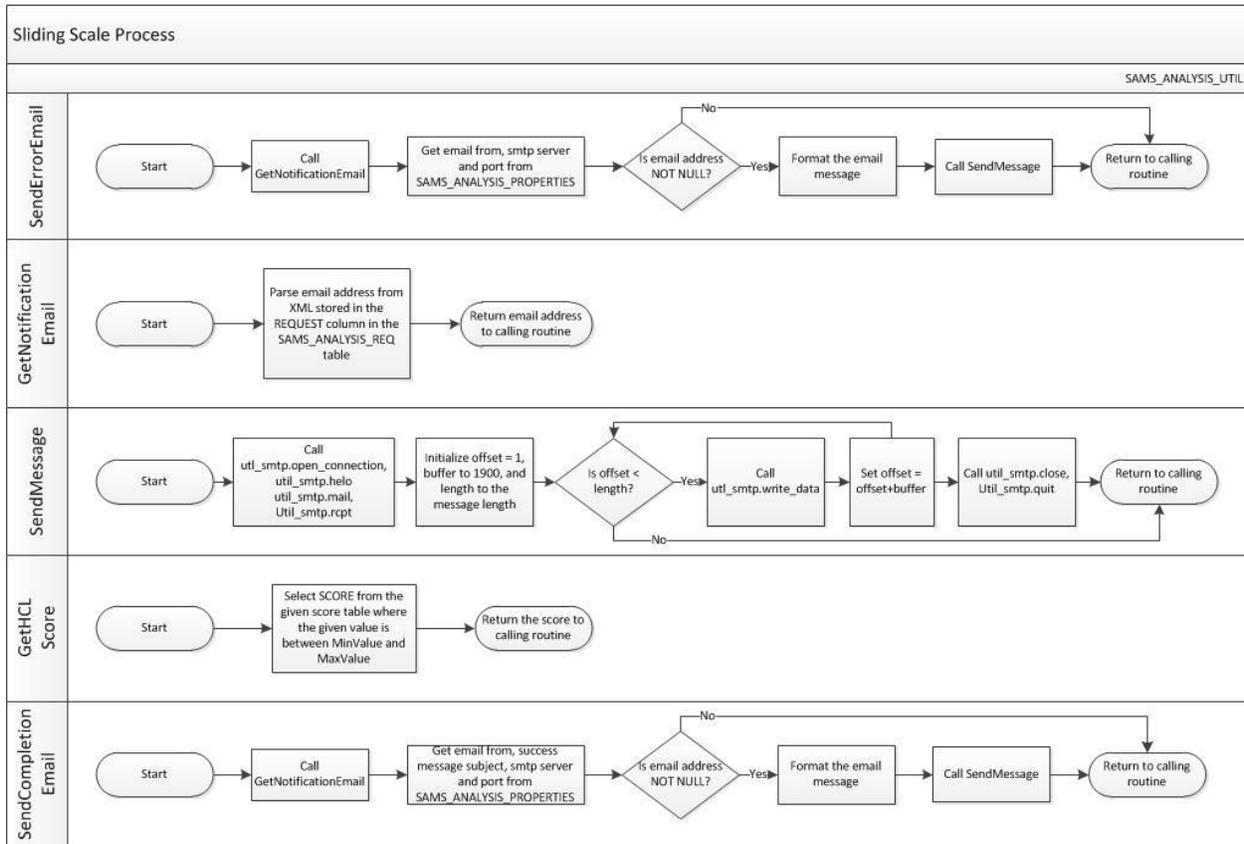
Sliding Scale Process	
SAMS_SLIDING_SCALE_ANALYSIS	
ComputeCrash TypePercentage	<pre> graph LR Start([Start]) --> Process[Count the number of records from SAMS_ANALYSIS_CRASH_TYPE joined with SS_CLUSTER_CRASHES for the given type, analysis ID, and cluster ID] Process --> Convert[Convert to a percentage by dividing by the total number of crashes] Convert --> End([Return the percentage to the calling routine]) </pre>
FatalCrashes	<pre> graph LR Start([Start]) --> Process[Count the number of records from CRASH joined with SS_CLUSTER_CRASHES for the given Cluster ID where SAMS_STAT_INJURY_SEV = 1] Process --> End([Return the number of fatal crashes to the calling routine]) </pre>
InjuryByType	<pre> graph LR Start([Start]) --> Process[Count the number of records from CRASH joined with SS_CLUSTER_CRASHES for the given Cluster ID and severity code] Process --> End([Return the number of crashes for the injury type to the calling routine]) </pre>
FunctionalClass	<pre> graph LR Start([Start]) --> Process1[Call GetPrimaryClusterSegment to get the Segment ID for the given County, Route and Begin/End Segment] Process1 --> Process2[Select FUNCTION_CLASS from ROAD_SEGMENT for the Segment ID] Process2 --> End([Return the Function Class to the calling routine]) </pre>
GetPrimary ClusterSegment	<pre> graph LR Start([Start]) --> Process[Select the PrimaryClusterSeg and SegLen from ROAD_SEGMENT for the given County, Route and Begin/End Segment] Process --> End([Return the PrimaryClusterSegment to the calling routine]) </pre>

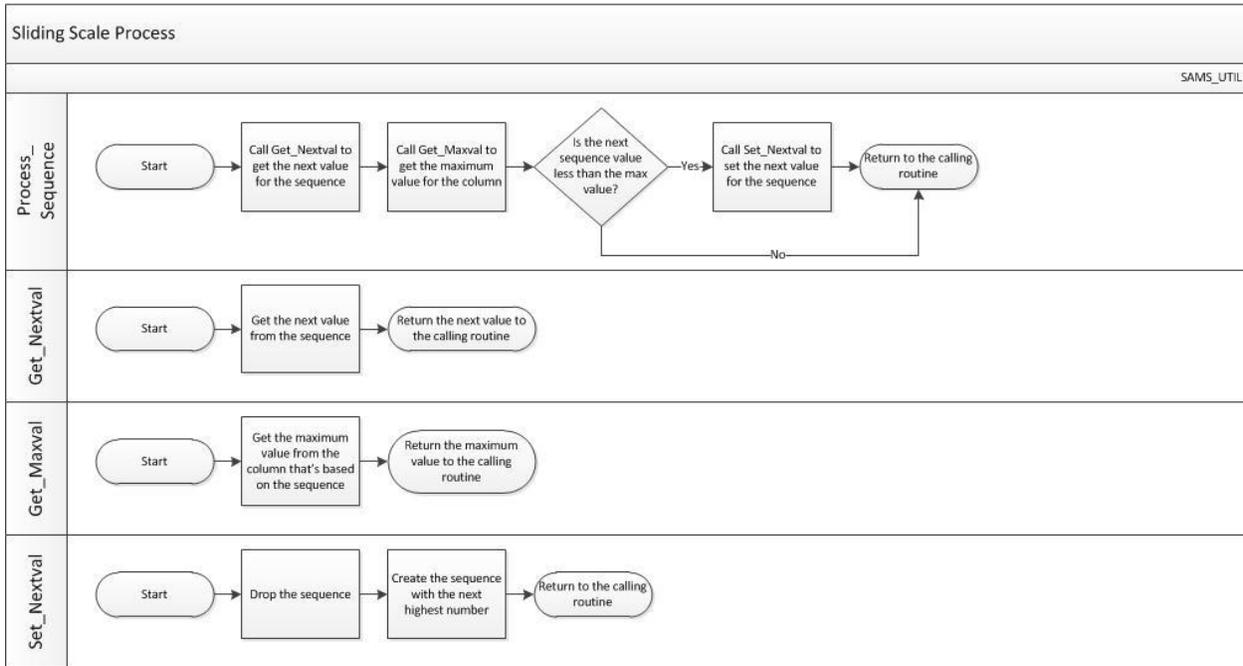








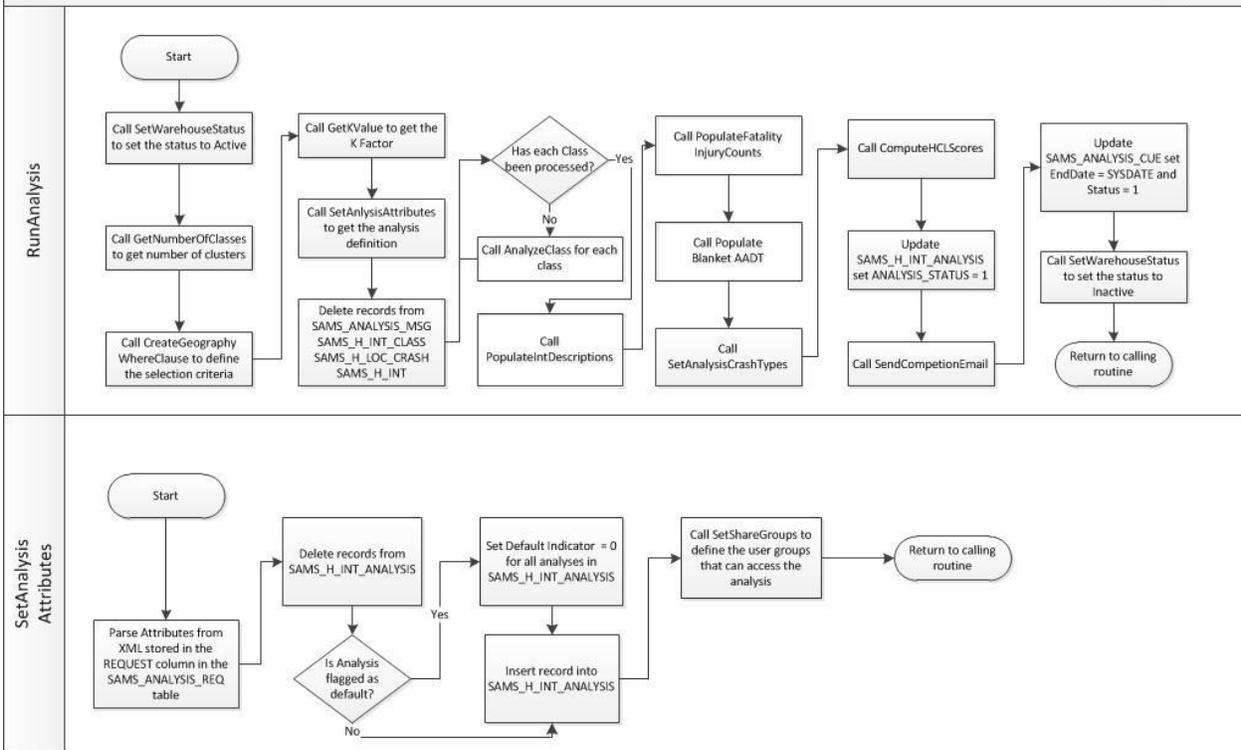


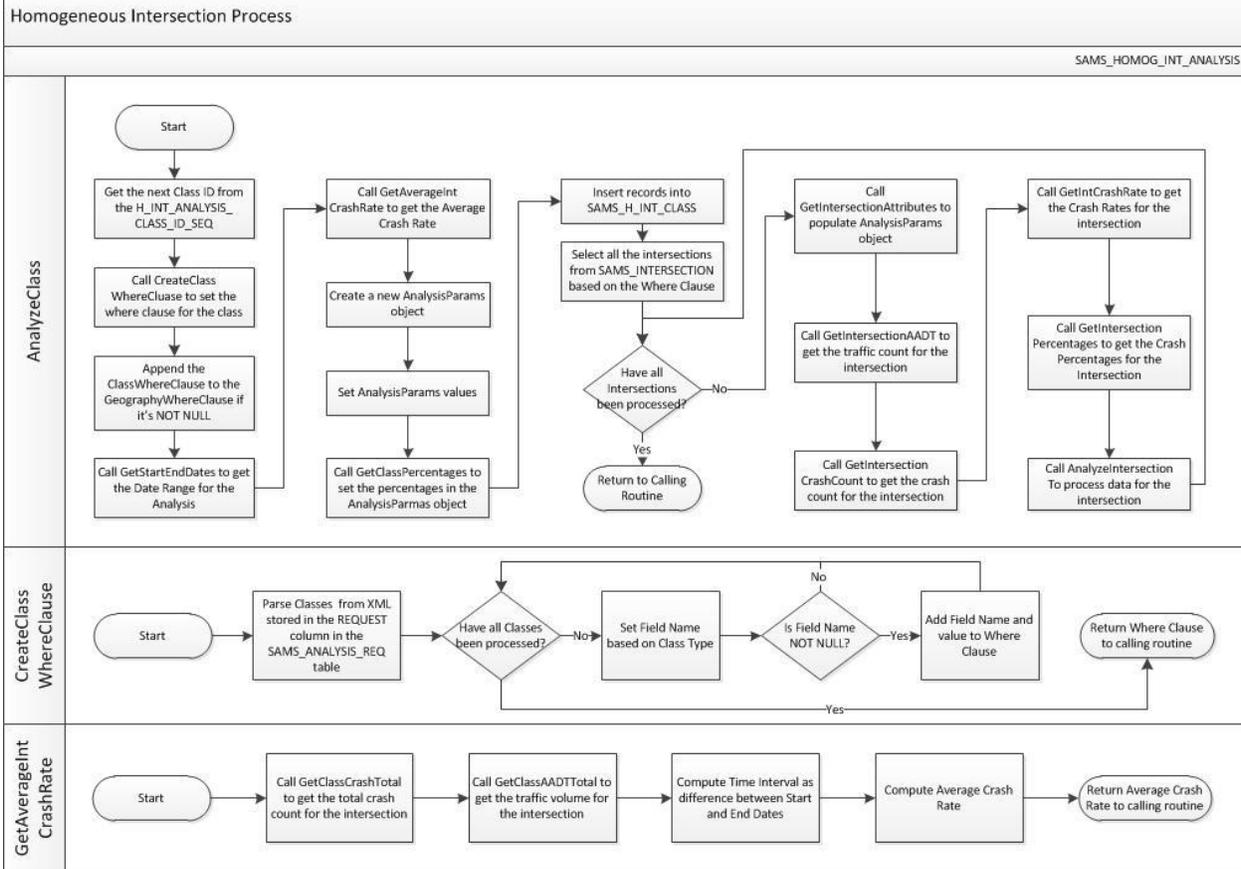


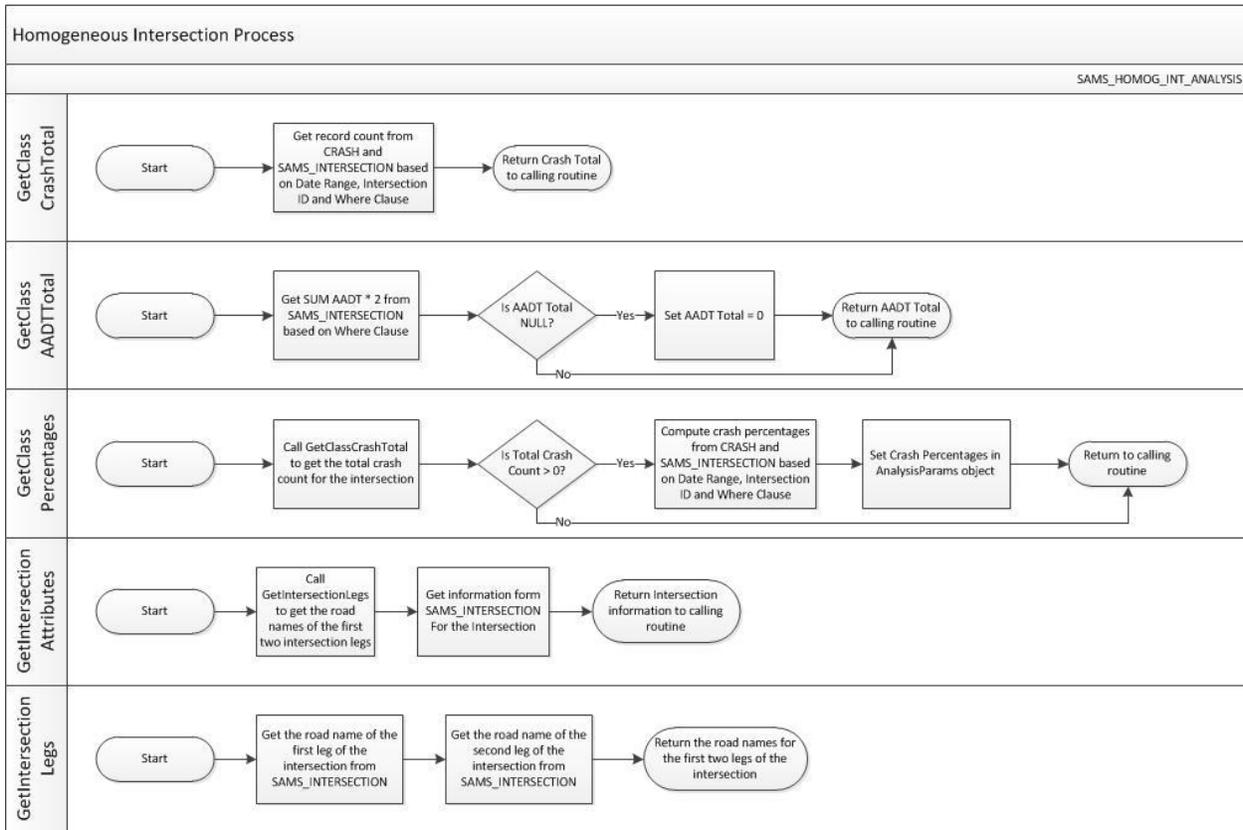


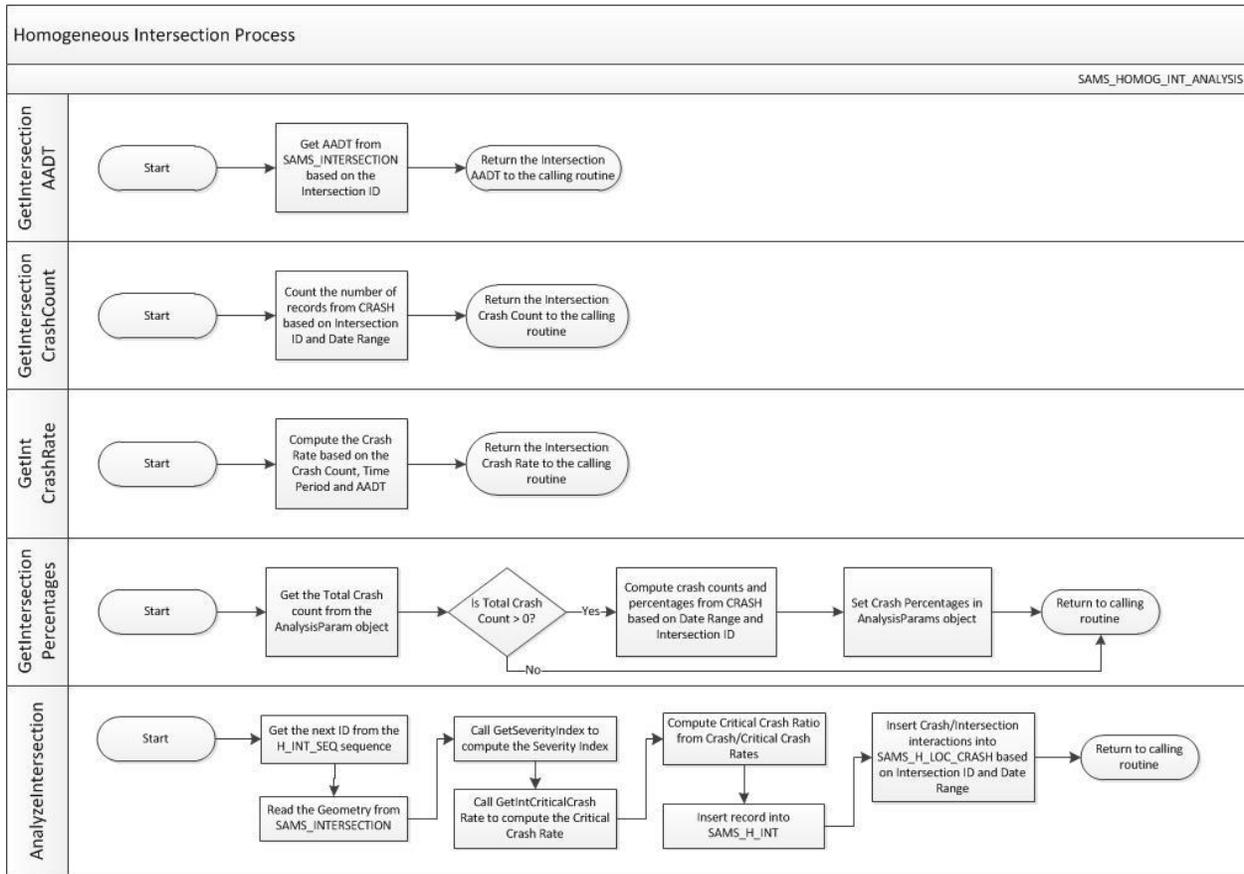
Homogeneous Intersection Process

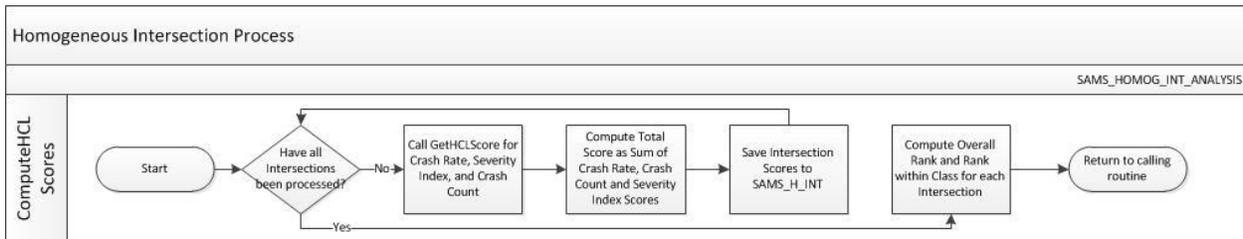
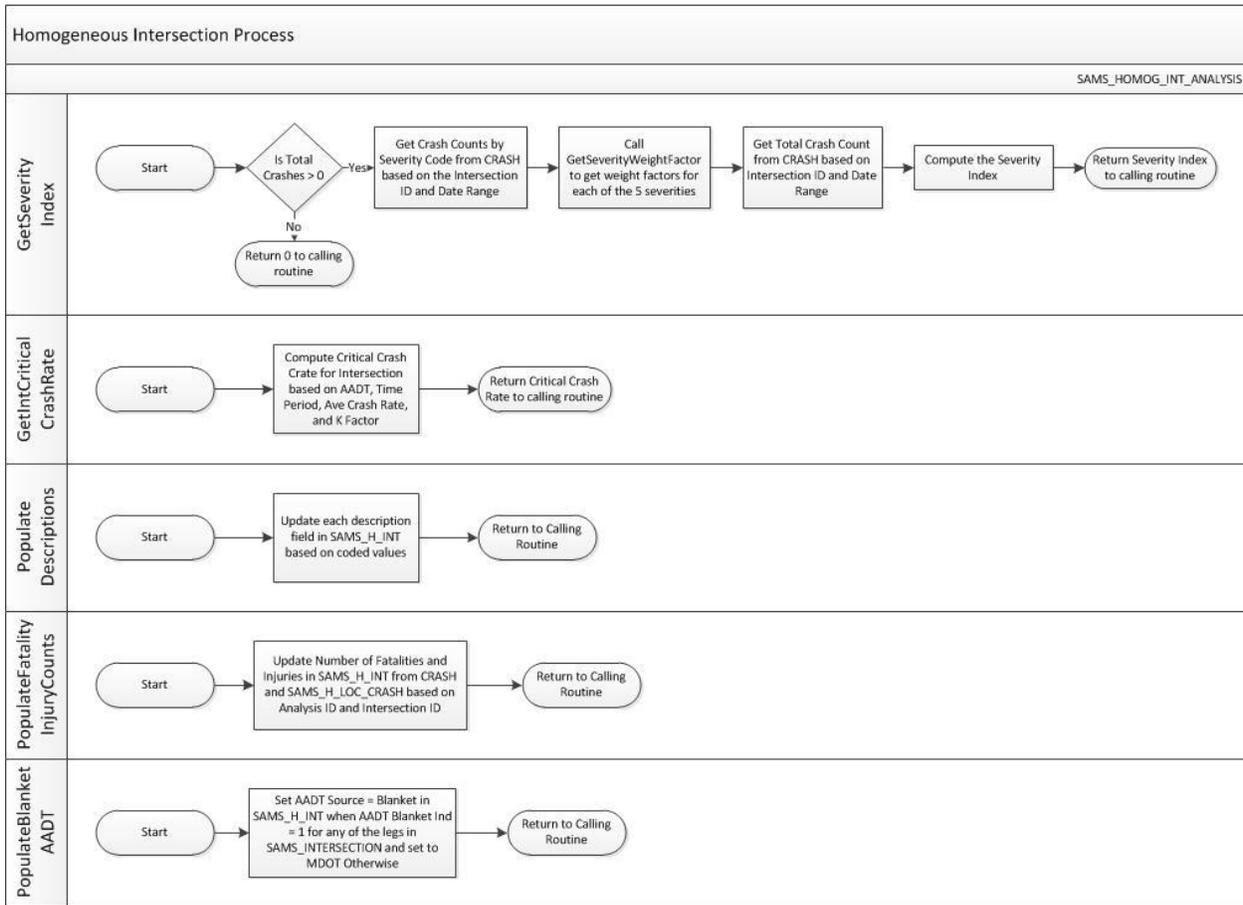
SAMS_HOMOG_INT_ANALYSIS

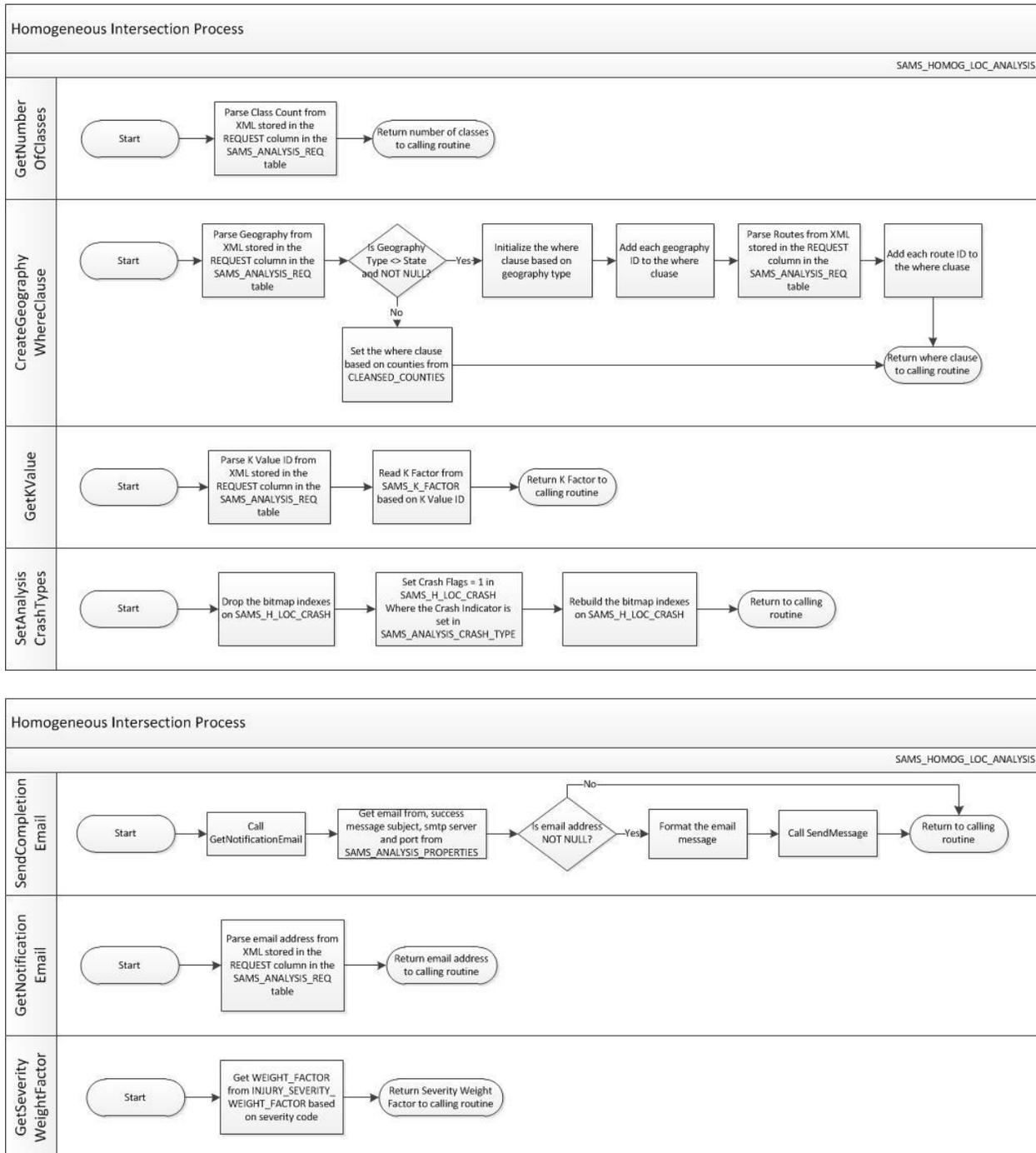


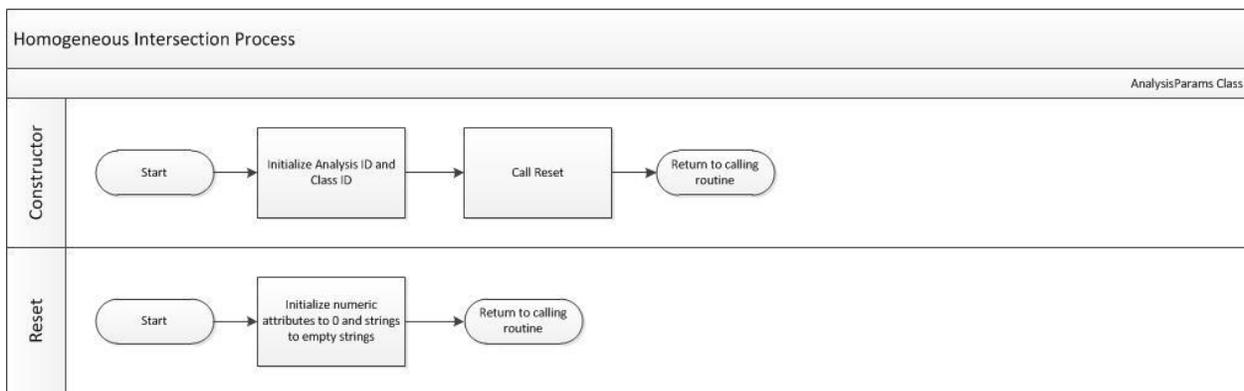
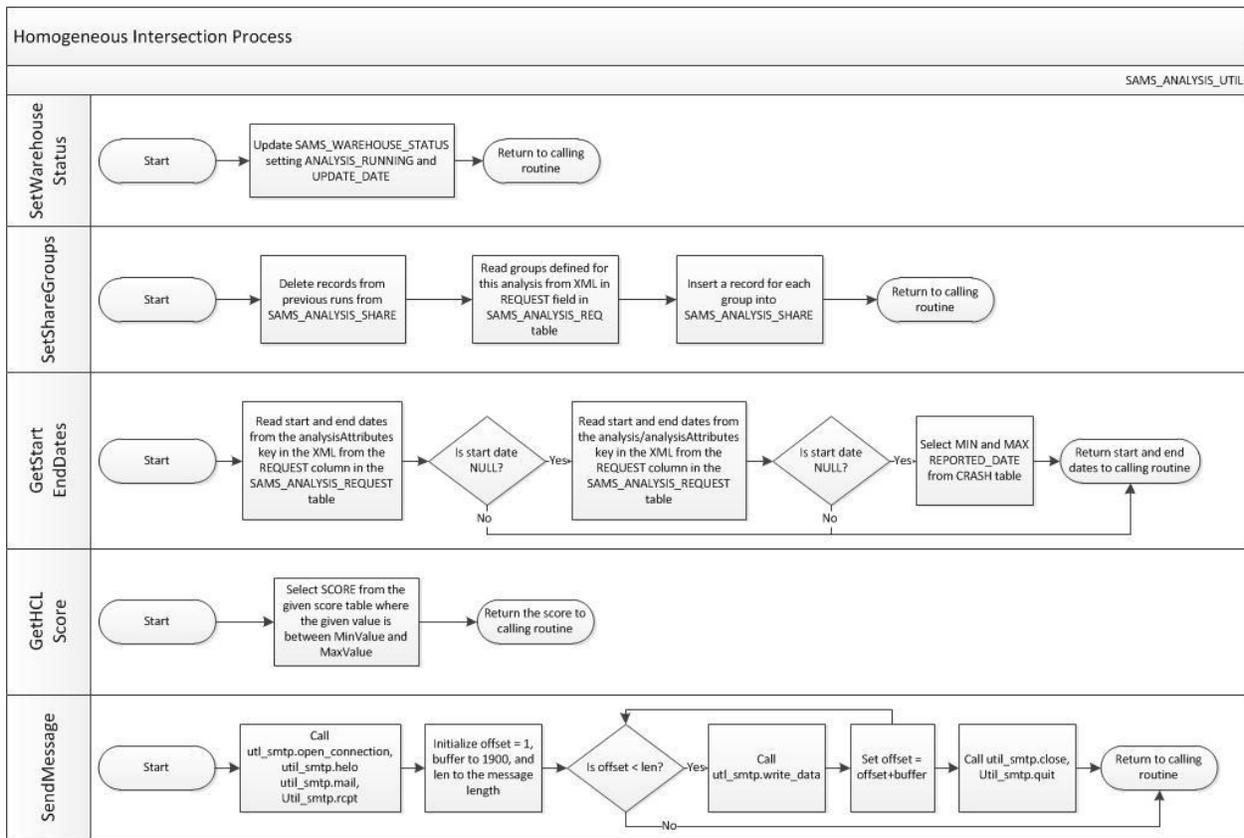














Appendix K – Homogeneous Section Database Diagram

