# CATHERINE DELA CRUZ

SQL Data Engineer
and Catastrophe Modeling Analyst

Brooklyn Park, Minnesota, USA
www.linkedin.com/in/catherine-a-delacruz/
github.com/cateydel


## CODING SAMPLE #1

**Microsoft SQL Server Run-Time Optimization**

## PROBLEM

A predecessor in my role wrote MS SQL scripts to extract property insurance policy and location data from an industry-standard risk model database for ingestion into the employer's online visualization software model. Three issues arose with the extraction of 272 million records of insurance portfolio data residing in sixteen tables:

a) High RAM consumption.

b) Endless creation of transaction log rollback records, such that all the memory on the hard drive was consumed before the script could finish.

c) Failures in ingestion resulting from carriage return and line feed characters in the extraction.

## SOLUTION

a) Decompose the single-query location extraction script into smaller queries, creating intermediate, temporary tables that contribute to the final extraction.

b) Extract only the columns that are necessary for the intermediate tables and final extraction.

c) Avoid UPDATE, INSERT, and DELETE queries wherever possible, as these are the key culprits in the creation of high volumes of rollback records.

d) Run a "string sweep" function to eliminate carriage returns, line feeds, and other problematic characters from the extraction.

## RESULTS

a) When run against the 272 million-record portfolio, the script, which had failed to finish running after two days prior to the optimization, finished in 5½ hours after applying the above solution.

b) When run against smaller portfolios, the script led to run-time improvements by factors of 2.0 to 5.0.

## FILES

https://tinyurl.com/y7baslmq · https://tinyurl.com/yc53zgrc
https://tinyurl.com/yd8tz4lw · https://tinyurl.com/yc7hj4ek

**Prior scripts (to be run sequentially):**

*Sample 1 - Prior Code - Query 1 - Policy Script.sql*
*Sample 1 - Prior Code - Query 2 - Location Script.sql*

**Scripts providing solution:**

*Sample 1 - Solution - Query 1 - Policy Script.sql*
*Sample 1 - Solution - Query 2 - Location Script.sql*

# CATHERINE DELA CRUZ

SQL Data Engineer
and Catastrophe Modeling Analyst

Brooklyn Park, Minnesota, USA
www.linkedin.com/in/catherine-a-delacruz/
github.com/cateydel


## CODING SAMPLE #2

**Circle Chain Geographic Concentration Algorithm**

## PROBLEM

A property insurance company wants to define underwriting zones for the sake of pricing and revenue forecasting. The company typically draws zone boundaries along postal, municipal, or county boundaries, or in grid blocks by latitude and longitude, but it is also aware that it insures meaningful concentrations of exposure that cross postal, municipal, county, or grid block boundary lines. While the company may have sophisticated visualization tools on hand that allow its underwriters to draw underwriting zone boundaries intuitively based on heat maps or point concentrations, its risk analysts may want to start with an algorithmic approach to drawing those boundaries.

## SOLUTION

A **circle chain geographic concentration algorithm** is founded on the idea of picturing a circle of diameter *x* centered around each property in an insurance portfolio. Properties are grouped together when they form chains of overlapping circles — in other words, circle chains.

The exposure aggregations defined by a circle chain algorithm have three desirable attributes:

a) The chains aren't constrained to any one shape.

b) Metro area chains tend to be larger than exurban and rural chains.

c) Chains end, and new chains begin, where properties begin to disperse.

## DISCUSSION

See the Adobe PDF document *Sample 2 - Circle Chain Property Aggregations.pdf* (viewable as noted below) for a summary of the methodology, applications, and caveats of the algorithm, and a summary of key conclusions to be drawn from it.

## FILES

**https://tinyurl.com/ybm4a6kk** · **https://tinyurl.com/y7pfba6j**

*Sample 2 - Circle Chain Geographic Concentration Algorithm.sql* (script)

*Sample 2 - Circle Chain Property Aggregations.pdf* (summary document)

# CATHERINE DELA CRUZ

SQL Data Engineer
and Catastrophe Modeling Analyst

Brooklyn Park, Minnesota, USA
www.linkedin.com/in/catherine-a-delacruz/
github.com/cateydel


## CODING SAMPLE #3

**User-Defined Incomplete Beta Function**

## PROBLEM

A property insurance actuary, ceded-reinsurance manager, or risk modeler may wish to calculate a policy or reinsurance contract's expected claim costs based on assumptions about the average claim cost to that contract per dollar of exposure (the *mean damage ratio*), the standard deviation of that claim cost as a ratio to the average (the damage ratio's *coefficient of variation*), and limits per claim occurrence, coupled with the assumption that all possible claim cost scenarios follow an *incomplete beta function*. This function is readily available in applications such as R, MATLAB, SAS, and Microsoft Excel, but not in Microsoft SQL Server – the analyst, however, wants to implement the function in Microsoft SQL Server for the sake of keeping the entire analytic process of data extraction, cleansing, validation, computation, and reporting contained within a single application.

## SOLUTION

a) Find two numerical recipes:

   1) One for a Lanczos approximation of a gamma function, such as the following in MATLAB: https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/13714/versions/1/previews/gammafun.m/index.html

   2) Another for an incomplete beta function, such as the following in C: https://codeplea.com/incomplete-beta-function-c

b) Manually convert each recipe to a user-defined function (UDF) in SQL Server Management Studio, calling on the gamma function UDF within the script for the incomplete beta function UDF.

## RESULTS

The calculation is admittedly not as fast in Microsoft SQL Server as it would be in the mathematically-oriented environments – for example, it takes six minutes to run 2,700 limited expected value calculations. This approach is best when the number of calculations is small, and when maintaining the entire analytic process within one application is essential.

## FILE

**https://tinyurl.com/yb8g8gs6**

*Sample 3 - Incomplete Beta Function.sql*