**Project: Global Commercial Banking Model Implementation**

**Project Purpose -> Implementation process flow requirement-> Implementation plan -> Implementation details -> Output monitoring**

# Purpose of the project

GCB (Global Commercial Banking) comprises one of biggest portfolios of Santander Holdings. As per the Regulators guidance, ‘Statistics/Econometrics’ based Loss Estimation Models are expected for large portfolios.

The GCB model is a very complex model, it is based on stochastics theory, and uses different models to determine the movement of each metric to get the final expected loss. To implement the model, I need to understand the model document provided by the model developer, and write the production code in SAS.

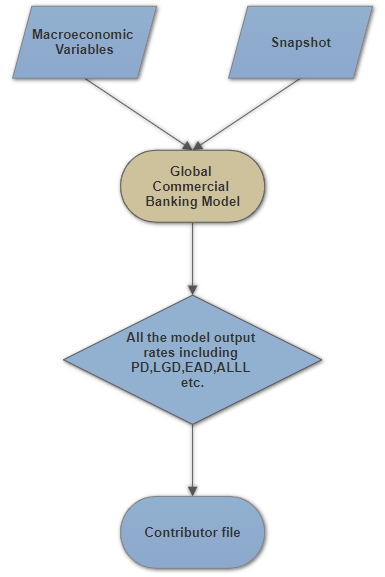
The skill needed in this project:

1. Knowledge of statistical concepts;
2. Deep understanding of modeling techniques that include Correlation Analysis, Linear Regression and Time Series Modeling, and the theories of stochastic processes;
3. Knowledge of Advanced linear algebra, such as matrix calculation, and probability tools like probability distributions;
4. Strong programming techniques include programming in SAS and Python which are required in constructing a model, getting the model output and implementing portfolio data analysis techniques.

# Process Flow Overview

According to the prevalent industrial risk model implementation process and internal or external control requirements, the standardized implementation process flow can be divided into three layers, which is illustrated in the flow chart below:

1. Querying the relevant data from database (Macroeconomic variables, Snapshot data);
2. Putting the data into the model and generating the rates computed by the model;
3. Generating the contributor file (standardized model output file can be read by the aggregated engine) as the final output.

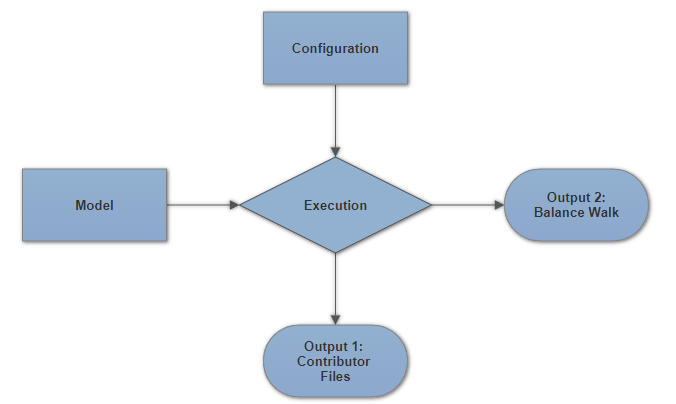


# Implementation plan

The implementation code can be divided into 3 modules to meet the requirements and usages of all related parties.

1. Configuration: environmental variable settings.
2. Model: all the models and computing details of the Global Commercial Banking model.
3. Execution: aggregated script for model execution to generate contributor file and balance walk result.

The relationship among these modules can be shown below:



To summarize, the *Configuration* module will supply the *Execution* module with the necessary information, including the environmental parameters and constant rates. The *Model* module will supply the *Execution* module with detailed computation steps. And the *Execution* module makes it easier for people to execute the model without knowing the details of the model, and to generate the model output, including the contributor files and balance walk.

To meet the predefined process shown above, the functionalities of each module should be:

The *Configuration* module has the following functionalities:

1. Import historical transition matrix of risk rating for different model segments;
2. Set up the environmental parameters for Oracle database query;
3. Store the constant rate for model computing;
4. Create inventories for storing the model results.

The *Model* module has the following functionalities:

1. M-factor computation;
2. Transition Matrix computation ;
3. Probability of Default calculation based on matrix migration;
4. Loss Given Default calculation based on matrix migration;
5. ALLL and Contingency rate calculation based on the matrix migration;
6. Balance walk generation.

The *Execution* module has the following functionalities:

1. Import the information from *Configuration* module;
2. Import the functionalities from *Model* module;
3. Specify the model execution steps.

# Implementation code outline

For the *Configuration* module and *Execution* module, one is served as the setting and storage script and the other is served as the interface for users, both of which has a little connection with model computing.

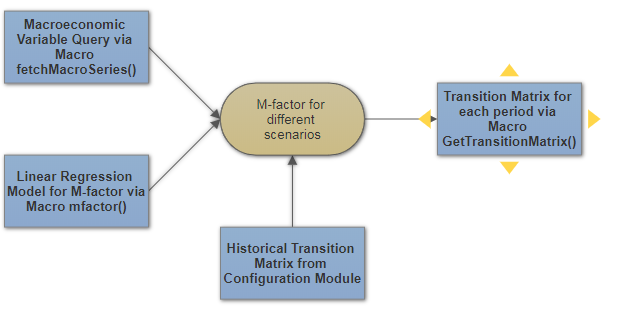
The core component of the whole model implementation code is *Model* module, which conveys all the model computing details. The *Model* module has about 1,000 lines’ code and more than 20 functions (so-called Macro in SAS).

**Core Component: *Model* module**

To outline the functionality of this module, it would be better to split it into several parts.

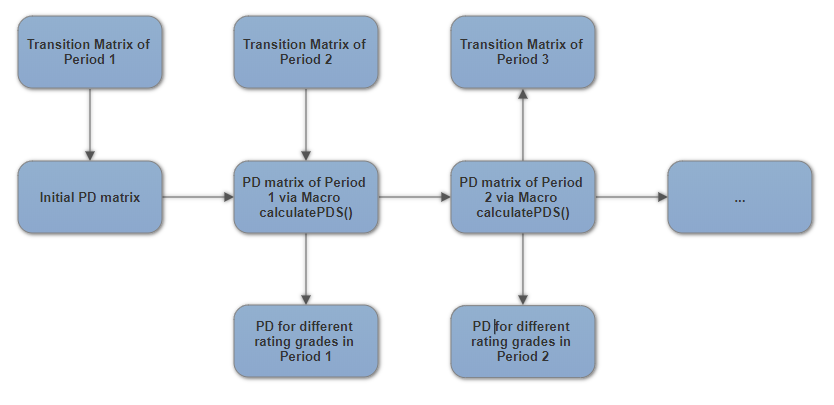
1. M-factor and Transition Matrix computation.

The functions (or called Macros) that are involved in this parts are fetchMacroSeries(), mfactor(), GetTransitionMatrix(). The basic idea is to put macroeconomic variables into linear regression model and then to get the M-factor, which is then used to compute the migration risk rating matrix for each time period.



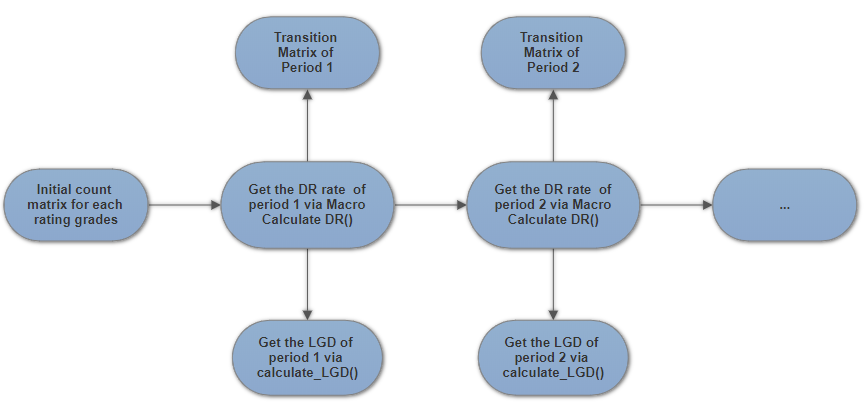
1. Probability of Default calculation.

Probability of Default computing is based on the matrix transition, so the macro named calculatePDS is involved in this part. Moreover, the transition direction is determined by the transition matrix of different period.



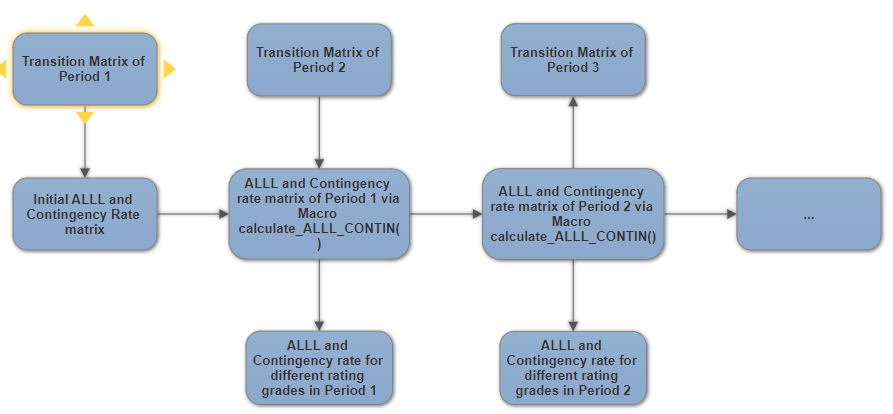
1. Loss Given Default calculation.

For the LGD computing, the process is more complicated. The LGD computing involves probability distribution approximation and calculation. The Macros that are used for this part are calculate\_LGD(), DR\_Calculation().



1. ALLL and Contingency rate calculation

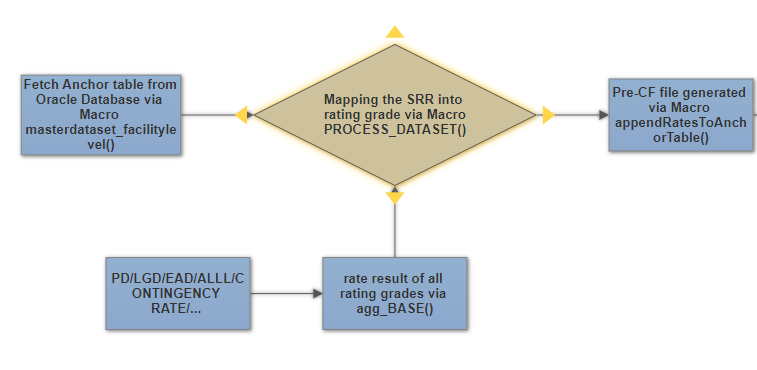
ALLL rate and Contingency rate calculation are similar as PD computing but with minor difference. The macros involved are calculate\_ALLL\_CONTIN(). The process is shown below.



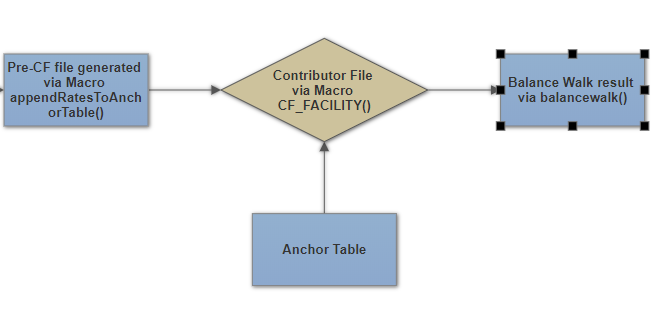
1. Contributor file and balance walk generation.

For the contributor file generation. It is an aggregated process which contains all computing parts listed above. And lots of data manipulation functions are involved in this part. The steps are following:

1. Calclulate PD/LGD/ALLL/Contingency rate and fetch EAD and other constant rates from *Configuration* module for each rating grade and append all the rate into one table;
2. Merge the rate result table with Anchor table based on rating grades to get the pre-cf file;
3. Convert the quarterly rate into monthly rate;
4. Transform the rate table into the required format.

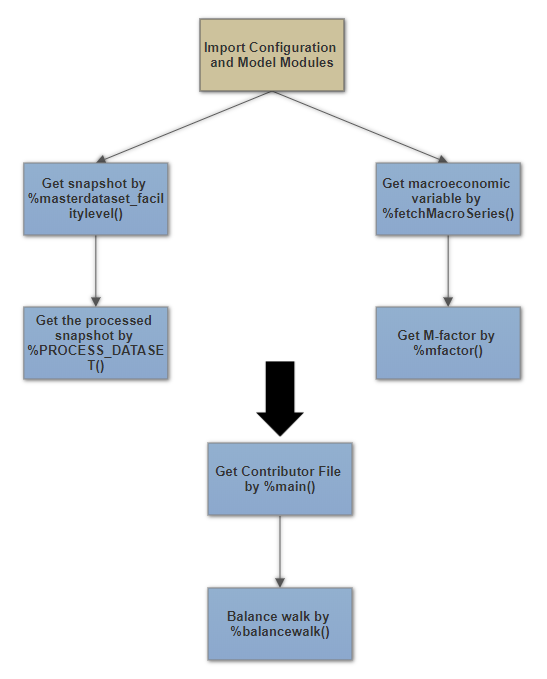


This is the first half part of the whole process in this part. The result generated is called Pre-CF file, which later will be transformed into the required format. And the second half, which is used to generate formatted contributor file and balance walk result are shown below.



**Interface Component: *Execution* module**

The *Execution* module is served as the interface for the implementation code, which is clearly organized and easy to use. It refers to *Model* module for the functions(Macro) and *Configuration* module for the necessary information. The structure of this module can be shown like below:



# Implementation result monitoring

To make sure the result generated by executing the model implementation code is correct, it is very necessary to check the output result (mainly the contributor file) by using the checking tools. The Macro check\_rate\_range() is a useful tool to check the correctness of the contributor file. It will check the length of the contributor file, the range of all the output rate, including PD, LGD, ALLL, Contingency Rate and etc.