Datamart Analysis Tool

1. Introduction
   1. Objectives

Nowadays most of us are involved in various new projects implementation and clients support. It is crucial for us to have a tool to automatically validate the Datamart design of clients.

The output of the tool can be used to

1. check the build quality of the Datamart
2. analysis the performance of the Datamart

Inspired from Anand’s presentation on Datamart analysis and also based on my work on HSBC existing Datamart analysis, the tool is designed to be able to analyze a Datamart in following aspects:

1. Dynamic table
2. Datamart table
3. Feeder (including batch of feeders)
4. Performance

Details of above analysis will be discussed from part III to VI.

* 1. Coding language

As Python is a flexible program language and has a rich set of libraries, Python was chosen as the coding language.

* 1. Installation
     1. Unzip the file into a folder

Please unzip the setup.zip file into an empty folder.

* + 1. Prepare the database connection file

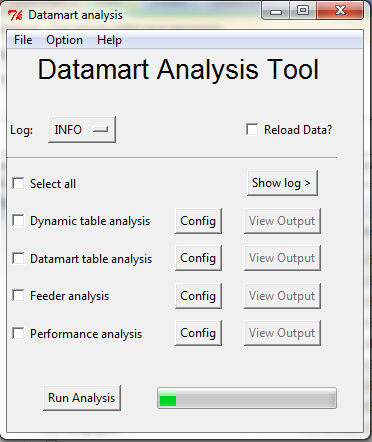
Please copy the dbsource.mxres from the targeting analysis mx environment to folder Properties if you can access the targeting mx environment. This will enable the DM analysis tool to retrieve the information in real time.

* 1. Execution

Please double click run.exe in the DM\_Analysis folder.

1. GUI
   1. Main window

After the tool is started up, main window will be rendered.



* 1. Tool Bar

In the top of the main window, there are three options on the tool bar.



* + 1. File -> Exit

You can exit the tool by clicking Exit.

* + 1. Option -> General Setting

In general setting, you can set the default behavior of following settings

|  |  |
| --- | --- |
| Name of setting | Values |
| Reload\_data | 0:No, 1:Yes |
| Sql\_directory | Where the sql files are stored |
| Input\_directory | Where the input files are stored. Input files can be generated automatically when reload\_data is set to 1 or provided by clients |
| Output\_directory | Where the output files are stored. |

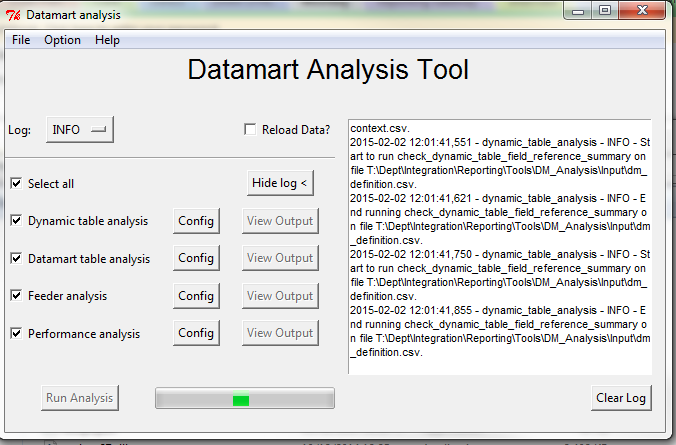
* + 1. Help -> About

Show About box.

* 1. Log settings



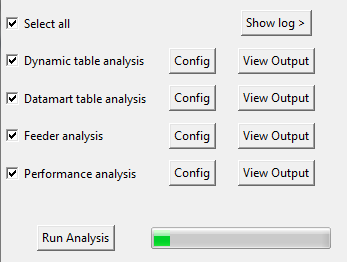
This is the Log settings for the tool. There are 5 log levels, i.e. Debug, Info, Warning, Error, Critical. Log files are stored in Logs folder. You can also click “show log>” button to expand the main window and see logs directly. The “show log>” button will become “Hide log<” when the log panel is ON.



* 1. Reload Data?

This is a switch to override the reload\_data setting in general settings. When “Reload data?” is ticked, the tool will try to connect to database and retrieve dm configuration.

* 1. Pick up the analysis type and run



In the bottom part of the main screen, we can select which analysis to execute by tick the box in front. We can also tick “select all” to select all analysis.

After any analysis is selected, we can execute the analysis by clicking “Run Analysis”.

* 1. Config

Config button for each analysis is used to set the parameters for each analysis. The details will be revealed in later chapters.

* 1. View Output

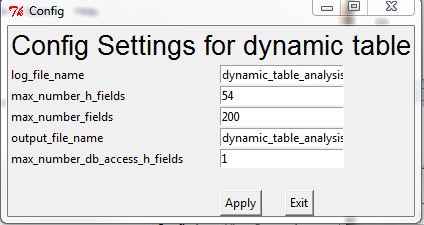
After the analysis is executed successfully, “View output” button will be enabled. By clicking it, Excel will be launched and the result will be shown.

1. Dynamic table analysis
   1. How to run “Dynamic table analysis”

Dynamic table analysis can be launched by select “dynamic table analysis” on the main window and click “Run Analysis”.

* 1. General Parameters

Click the “config” button right to “Dynamic table analysis”



Log\_filename defines the log file for this anlaysis and output\_file\_name defines the output file name of this analysis. The other settings will be introduced in next points.

* 1. Checking total number of dynamic table fields selected (sheet: Field\_Check)

The purpose of this is to make sure that any single dynamic table will not select more than maximum fields. The maximum fields can be configured by “max\_number\_fields” in configuration. By default, this number is set 200.



These fields include both native dynamic table fields and horizontal fields. Here is a sample output. Only dynamic table that has more than maximum fields will be shown in the output sheet.

|  |  |  |  |
| --- | --- | --- | --- |
| **Number of Dynamic table fields that exceeds 200** | | | |
| **Dynamic table name** | **Category** | **Dynamic table type** | **Field count** |
| **HL\_FXOExpiry** | **User** | **DYN\_TRNRP\_PL** | **639** |
| **HL\_MM\_SEC\_Risk2** | **User** | **DYN\_TRNRP\_PL** | **374** |

* 1. Checking total number of horizontal fields (sheet :H\_Field\_Check)

The purpose of this is to make sure that any single dynamic table will not create more than maximum horizontal fields. The maximum horizontal fields can be configured by “max\_number\_h\_fields” in configuration. SQL Qurey\_dm\_config will be executed for the raw data and following fields from the output will be used. Only dynamic table that has more than maximum fields will be shown in the output sheet.

|  |  |  |  |
| --- | --- | --- | --- |
| **Number of Dynamic table horizontal fields that exceeds 54** | | | |
| **Dynamic table name** | **Category** | **Dynamic table type** | **Horizontal Field count** |
| **HLMY\_BOPESS** | **User** | **Accounting** | **68** |
| **HLMY\_FIN\_GLINT** | **User** | **Accounting** | **72** |
| **HLSG\_BOND** | **User** | **DYN\_TRNRP\_PL** | **57** |
| **HLSG\_BOND\_NEW** | **User** | **DYN\_TRNRP\_PL** | **56** |
| **HLSG\_IRS** | **User** | **DYN\_TRNRP\_PL** | **79** |
| **HLSG\_SWAP** | **User** | **DYN\_TRNRP\_PL** | **64** |

* 1. Check horizontal fields with \*TBLFIELD and \*TABLE (sheet: H\_DB\_Field\_Check)

We shall, in any cases, avoid use parser function \*TBLFIELD and \*TABLE in a horizontal field. This analysis will return all dynamic table that has horizontal fields that access database directly more than a maximum time. The maximum fields can be configured by “max\_number\_db\_access\_h\_fields” in configuration. Here is a sample output.

|  |  |  |  |
| --- | --- | --- | --- |
| **Number of Dynamic table horizontal fields that access database which exceeds 1** | | | |
| **Dynamic table name** | **Category** | **Dynamic table type** | **Direct DB access Parser function used times** |
| **ERM\_COLLAT\_MV** | **User Additional** | **DYN\_TRNRP\_MV** | **3** |
| **DCI\_PACKAGE** | **User** | **DYN\_TRNRP\_PL** | **2** |
| **MLC\_TRNRP\_MV** | **User Additional** | **DYN\_TRNRP\_MV** | **18** |
| **HLMY\_BOND\_FI** | **User** | **DYN\_TRNRP\_PL** | **34** |
| **HLMY\_BOND\_FI\_2** | **User** | **DYN\_TRNRP\_PL** | **39** |

* 1. Check Disabled computer flag and sensitivity flag (Sheet: Sensi\_Flag\_Check)

This analysis will query all the DYN\_TRNRP like dynamic table that has enable sensitivity flag but no S\_ fields are selected. Here is a sample output.

|  |  |
| --- | --- |
| **Dynamic tables which sensitivity compute flag can be disabled** | |
| **Dynamic table name** | **Category** |
| **G\_PLCG3.4e /TRNRP\_PL** | **Murex** |
| **G\_PLAA.2 /TRNRP\_CS** | **Murex** |
| **G\_PL02.3 /RPPL** | **Murex** |
| **G\_PLCG2.4 /RPPL** | **Murex** |
| **G\_FX.4 /TRNRP\_DT** | **Murex** |

We shall open the dynamic tables found and disable “sensitivity flag”

* 1. Check dynamic table underlying simulation viewer context (sheet: Build\_Mode\_Check)

If the simulation viewer is based on consolidated, then dynamic table shall be consolidated otherwise the dynamic table shall be set to detailed mode. Here is a sample output.

|  |  |  |  |
| --- | --- | --- | --- |
| **Dynamic table with wrong build on mode** | | | |
| **Dynamic table name** | **Category** | **Simulation name** | **Build on mode** |
|  |  |  |  |

* 1. Field reference summary (sheet: Field\_Reference\_Summary)

The page will list how many times a dynamic fields referenced in all Datamart tables. The sheet will display field name, dynamic table type and total referenced number. It is sorted by number of reference.

|  |  |  |
| --- | --- | --- |
| **Summary of dynamic table field reference** | | |
| **Dynamic table field** | **Dynamic table type** | **# of reference** |
| **IDENTITY** | **Transaction** | **207** |
| **REF\_DATA** | **Transaction** | **207** |
| **TIMESTAMP** | **Transaction** | **207** |
| **MX\_REF\_JOB** | **Transaction** | **207** |
| **NB** | **Transaction** | **190** |
| **TP\_CNTRP** | **Transaction** | **160** |
| **TP\_DTEEXP** | **Transaction** | **156** |
| **TP\_DTETRN** | **Transaction** | **155** |
| **TRN\_TYPE** | **Transaction** | **149** |
| **TRN\_GRP** | **Transaction** | **148** |
| **TRN\_FMLY** | **Transaction** | **143** |
| **TP\_PFOLIO** | **Transaction** | **139** |
| **TP\_ENTITY** | **Transaction** | **138** |

* 1. Field reference detail (sheet: Field\_Reference\_Detail)

The sheet display details of reference status of each field. It is sorted by dynamic table field.

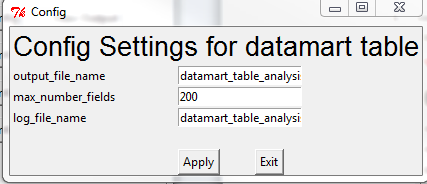
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Summary of dynamic table field reference** | | | | |
| **Dynamic table field** | **Dynamic table category** | **Dynamic table** | **Dynamic table type** | **Datamart table** |
| **AAREFNO** | **User** | **HL\_TRN\_CPUDF** | **Copy Creation** | **HL\_TRN\_CPUDF.REP** |
| **ACCCUR** | **User** | **HL\_PFLD** | **Copy Creation** | **HL\_PFLD.REP** |
| **ACCOUNT** | **Murex** | **DYN\_ACJT** | **Accounting (Reporting)** | **ACJT.REP** |
| **ACCRTE\_YTD** | **User** | **HL\_MM\_SEC\_NEW** | **Transaction** | **HL\_MM\_SEC\_NEW.REP** |
| **ACCRTE\_YTD** | **User** | **HL\_MM\_SEC\_SG** | **Transaction** | **HL\_MM\_SEC\_SG.REP** |
| **ACCRTE\_YTD** | **User** | **HL\_MM\_SEC\_TEST** | **None** | **HL\_MM\_SEC\_TEST.REP** |
| **ACCRTE\_YTD** | **User** | **HL\_MM\_SEC\_1** | **Transaction** | **HL\_MM\_SEC1.REP** |
| **ACCRUEDAMT** | **User** | **HL\_MM\_SEC\_SG** | **Transaction** | **HL\_MM\_SEC\_SG.REP** |
| **ACCR\_MODE** | **Murex** | **DYN\_TPCR** | **Transaction** | **TPCR.REP** |
| **ACCSECTION** | **User** | **HL\_PFLD** | **Copy Creation** | **HL\_PFLD.REP** |

1. Datamart table analysis
   1. How to run “Datamart table analysis”

Dynamic table analysis can be launched by select “Datamart table analysis” on the main window and click “Run Analysis”.

* 1. General Parameters

Click the “config” button right to “Dynamic table analysis”



Log\_filename defines the log file for this anlaysis and output\_file\_name defines the output file name of this analysis. The other settings will be introduced in next points.

* 1. Checking total number of datamart table fields selected (sheet: Field\_Check)

The purpose of this is to make sure that any single datamart table will not select too many fields. The maximum fields can be configured by “max\_number\_fields” in configuration. By default, this number is set 200.



These fields include both native dynamic table fields and horizontal fields. Here is a sample output. Only Datamart table that has more than maximum fields will be shown in the output sheet.

|  |  |
| --- | --- |
| **Datamart table fields exceeds 203** | |
| **Datamart table name** | **Field count** |
| **ERM\_MV.REP** | **216** |
| **PL\_012.REP** | **248** |
| **HLSG\_BOND.REP** | **224** |
| **HL\_PL\_012.REP** | **254** |
| **HLSG\_BOND\_NEW.REP** | **311** |

* 1. Check number of datamart table fields that is different from underlying dynamic table (sheet: Field\_Inconsistent

If datamart table contains fewer fields than underlying dynamic table, it means that we waste computing power by populating the fields but not storing it anyway! We want to find out such datamart tables. In the output sheet, all datamart tables which has less fields than dynamic table will be listed. Here is a sample output.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Datamart table has less fields than dynamic table fields** | | | | |
| **Datamart table name** | **Field count** | **Dynamic table name** | **Category** | **Dynamic table field** |
| **HLMY\_FEC\_CPP.REP** | **59** | **HLMY\_FEC\_CPP** | **User** | **71** |
| **PL\_DETAILED.REP** | **99** | **DYN\_PL\_DETAILED** | **Murex** | **130** |
| **MKBK\_2.REP** | **11** | **DYN\_MKBK\_2** | **Murex** | **31** |
| **BT\_ACT\_MYR\_MKTOP.REP** | **12** | **ERM\_BT\_ACT\_MYR\_MKTOP** | **User** | **14** |
| **FIXING.REP** | **25** | **DYN\_FIXING** | **Murex** | **27** |
| **HLMY\_BOPESS.REP** | **67** | **HLMY\_BOPESS** | **User** | **84** |
| **MLC\_CS\_DEAD2.DBF** | **19** | **MLC\_CS\_DEAD2** | **User Additional** | **38** |
| **HL\_FIN\_KLGINT.DBF** | **44** | **HL\_FIN\_KLGLINT\_TBL** | **User** | **66** |

* 1. Check index definition for a datamart table (sheet: No\_Index\_Table)

Any datamart table shall have more than one index. SQL Qurey\_dm\_config will be executed for the raw data and following fields from the output will be used. Here is a sample output.

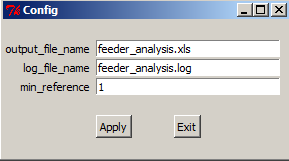
|  |  |
| --- | --- |
| **Following Datamart table dose not have index** | |
| **Datamart table name** | **Index count** |
| **C\_ST\_CAL\_HOL.REP** | **0** |
| **C\_SITRN.REP** | **0** |
| **IRD\_BASIS\_RISK.REP** | **0** |
| **BT\_ACT\_MYR\_MKTOP.REP** | **0** |
| **C\_CNT\_EVT\_UDF.REP** | **0** |
| **ACC\_JOURNAL.REP** | **0** |
| **C\_SITRN\_PRE.REP** | **0** |

1. Feeder analysis
   1. How to run “Datamart table analysis”

Dynamic table analysis can be launched by select “Datamart table analysis” on the main window and click “Run Analysis”.

* 1. General Parameters

Click the “config” button right to “Dynamic table analysis”



Log\_filename defines the log file for this anlaysis and output\_file\_name defines the output file name of this analysis. Min\_reference defines that the minimum number of same datamart table/feeder/batch feeder that are considered as duplication. A common number will be 2.

* 1. Summary of multi-referenced datamart objects. (sheet: summary)

In this sheet, datamart table/feeder/batch of feeders that are referenced more than the times defined in the field min\_reference of config page will be shown.

|  |  |
| --- | --- |
| **A summary of multi-referenced dm objects.** | |
| **Name of Datamart table** | **# of Referenced feeders** |
| **HEDGE\_MSRMNT.REP** | **2** |
| **DESC\_THEO\_HDG.REP** | **2** |
| **HDG\_TRD\_DETAILS.REP** | **2** |
| **CLT\_TPPL\_KYC.REP** | **6** |
| **REF\_DETPL.REP** | **4** |

* 1. Check if batch of feeders with same label has consistent configuration (sheet: Dataset\_consistency)

Batch of feeder with same label of data must have same historization and data computed by several batch and Data(published/private). In the output sheet, all the batch of feeders with their configuration will be listed. If the batch of feeder has same setting with the rest, its setting will be marked “OK”, otherwise it will be marked as “Inconsistent”.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Batch of feeder has possible wrong dataset settings were marked as inconsistent in last column** | | | | | |
| **Batch of feeder** | **Label of Data** | **Historisation** | **Private** | **Data Computed by Several Batch** | **Setting is correct** |
| **SG\_SWAP\_RECON** | **SWAP\_RECON** | **ONE DATA SET** | **PRIVATE** | **Y** | **OK** |
| **BF\_BROKERAGE** | **LOD\_BROKERAGE** | **ONE DATA SET** | **PRIVATE** | **Y** | **OK** |
| **MX\_MSRMNT\_BF** | **HEDGE** | **ONE DATA SET PER DAY** | **DATA PUBLISHED** | **Y** | **OK** |
| **MX\_IAS\_BF** | **HEDGE** | **ONE DATA SET PER DAY** | **DATA PUBLISHED** | **Y** | **OK** |
| **MX\_HDG\_TRD\_BF** | **HEDGE** | **ONE DATA SET PER DAY** | **DATA PUBLISHED** | **Y** | **OK** |
| **WL\_FD\_ACAC** | **260209** | **ONE DATA SET** | **PRIVATE** | **Y** | **OK** |
| **WL\_FD\_ACJS** | **260209** | **ONE DATA SET** | **PRIVATE** | **Y** | **OK** |
| **BDF\_ALM\_LFUT** | **ALM\_LFUT** | **ONE DATA SET** | **PRIVATE** | **Y** | **OK** |
| **BDF\_NIDI\_CRM** | **NIDI\_CRM1** | **ONE DATA SET** | **PRIVATE** | **Y** | **OK** |
| **C\_MD\_HISTRT\_BF** | **EOD\_DATA\_D** | **ONE DATA SET PER DAY** | **DATA PUBLISHED** | **Y** | **OK** |
| **C\_BE\_DCI\_MID\_RT** | **EOD\_DATA\_D** | **ONE DATA SET** | **DATA PUBLISHED** | **N** | **Inconsistent** |
| **C\_MKT\_SENSI\_BF** | **EOD\_DATA\_D** | **ONE DATA SET PER DAY** | **DATA PUBLISHED** | **Y** | **OK** |
| **C\_BE\_HWRK\_FTPMR** | **EOD\_DATA\_D** | **ONE DATA SET** | **PRIVATE** | **N** | **Inconsistent** |

* 1. Checking datamart object referenced status (sheet: Object\_referred\_Summary)

This will give a summary of datamart tables/feeders/batch of feeders that are referenced more than one time. Here is a sample output.

|  |  |
| --- | --- |
| **A summary of multi-referenced dm objects.** | |
| **Name of Datamart table** | **# of Referenced feeders** |
| **HL\_TRNRP\_NOTYPO.REP** | **2** |
| **MKBK\_2.REP** | **2** |
| **HL\_FIN\_KLGINT.REP** | **2** |
| **Name of feeders** | **# of Referenced Batch feeders** |
| **FD\_XNIDI\_CRM** | **2** |
| **FD\_ACAC** | **5** |
| **FD\_HL\_MLCRISK3** | **2** |
| **Name of batch feeders** | **# of Referenced Batch feeders** |
| **BDF\_HL\_OPT\_ORG** | **2** |
| **BDF\_ALM\_BOND** | **3** |
| **BDF\_STPSIA\_CRM** | **4** |
| **BDF\_LOANDEPOSIT** | **2** |

* 1. Check if same datamart table is defined more than 1 single feeder (sheet: DM\_Table\_Duplication)

If same datamart table is defined more than 1 feeders, it may means redundant. Here is a sample output. The last execution date gives the latest execution time for the feeder. If a feeder was executed recently, it may indicate that the feeder is still in use, otherwise the feeder can be possible purged from the system.

|  |  |  |
| --- | --- | --- |
| **Datamart tables that definded in multiple single feeders** | | |
| **DM Table Name** | **Feeder name** | **Last Execution Date** |
| **HL\_TRN\_HDR\_DER.REP** | **HL\_TRN\_HDR\_DER** |  |
| **HL\_TRN\_HDR\_DER.REP** | **FD\_TRN\_HDR\_DER** |  |
| **PL\_012.REP** | **A\_PL\_012** |  |
| **PL\_012.REP** | **FD\_MXG\_SR1** |  |

* 1. Check if same feeder is defined more than 1 batch of feeders (sheet: Feeder\_Duplication)

If same feeder is defined more than 1 batch of feeders, it may means redundant.

Here is a sample output. The last execution date gives the latest execution time for the batch feeder. If a batch feeder was executed recently, it may indicate that the batch feeder is still in use, otherwise the batch feeder can be possible purged from the system.

|  |  |  |
| --- | --- | --- |
| **Feeder that definded in multiple batches** | | |
| **Feeder** | **Batch of feeder** | **Last Execution Date** |
| **FD\_TRNRP\_PL\_FEC** | **REC\_BDF\_MX\_22** |  |
| **FD\_TRNRP\_PL\_FEC** | **FD\_TRNRP\_PL\_FEC** |  |
| **FD\_TRNRP\_PL\_FEC** | **BDF\_MX\_22** |  |
| **FD\_NIDI\_CRM** | **BDF\_NIDI\_CRM** |  |
| **FD\_NIDI\_CRM** | **BDF\_NIDIBB** | **20140315** |

* 1. Check if same batch feeder is defined more than 1 processing script. (Batch\_Feeder\_Duplication)

If same batch feeder is defined more than processing script, it may means redundant. Here is a sample output.

|  |  |
| --- | --- |
| **Batch of Feeder that definded in multiple processing scripts** | |
| **Batch of Feeder** | **Processing Script** |
| **BDF\_STPSIA\_CRM** | **AVERAGE\_BALANCE** |
| **BDF\_STPSIA\_CRM** | **CRM\_STPSIA** |
| **BDF\_STPSIA\_CRM** | **CKM\_STPSIA** |

* 1. Check the usage of the scanner engine (sheet: Scanner\_Engine)

Scanner engine is used to speed up the execution time for batch of feeders. By using scanner engine is sometimes also fixing 4GB memory issues. It is always recommended to enable scanner engine for eligible batch of feeders. The output of this analysis will list all eligible batch of feeder names and number of scanner engine selected. If the number if 0, we can enable the scanner engine setting for the batch of feeder.

|  |  |  |
| --- | --- | --- |
| **Batch of feeders scanner engine usage** | | |
| **Batch of feeder** | **Dynamic table type** | **Scanner engine number** |
| **C\_TP\_OPT\_BF** | **DYN\_TRNRP\_PL** | **8** |
| **BDF\_SG\_BOND\_NEW** | **DYN\_TRNRP\_PL** | **0** |
| **MX\_PL\_SECFIN\_BF** | **DYN\_TRNRP\_PL** | **0** |
| **BDF\_SR2\_FIN** | **DYN\_TRNRP\_PL** | **0** |

* 1. Checking number of feeders in a batch of feeder (sheet: Feeder\_number)

Normally we don’t encourage inserting too many feeders into a batch feeder. In the output sheet, it lists all batch of feeder and the number of its underlying feeders. If there are too many feeders in one batch of feeder, we can consider to reduce the number.

|  |  |
| --- | --- |
| **Number of feeders in a batch of feeders** | |
| **Batch of feeder** | **Number of Feeder** |
| **MX\_PL\_SECFIN\_BF** | **1** |
| **BF\_MRA\_MYR\_VAR** | **1** |
| **ERM\_CS\_BF** | **1** |
| **MLC\_CNS** | **2** |
| **MX\_FXC\_DETPL\_BF** | **1** |

* 1. Checking if there is any filter conflicting between dynamic table default setting and global filter

The global filter set in a batch of feeder will override or logical “AND” with underlying dynamic table’s default setting. This will sometimes cause unwanted filtering. This analysis will show all possible conflict between dynamic table default setting and global filter. Here is link to the sample output.

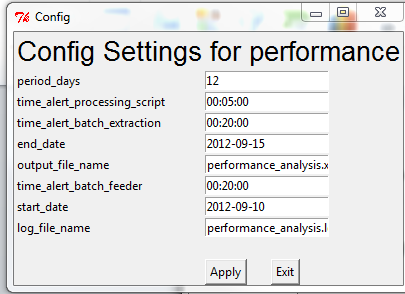


1. Performance analysis
   1. How to run “Datamart table analysis”

Dynamic table analysis can be launched by select “Datamart table analysis” on the main window and click “Run Analysis”.

* 1. General Parameters

Click the “config” button right to “Dynamic table analysis”



Log\_filename defines the log file for this anlaysis and output\_file\_name defines the output file name of this analysis. The other settings will be introduced in next points.

* 1. Processing script performance(sheet: Processing script performance)

This analysis will process the processing script execution in a range and list those processing scripts take too long. The input parameters are

|  |  |
| --- | --- |
| Parameter Name | Description |
| Period\_days | Number of days that the analysis will be applied |
| Time\_alert\_processing\_script | The threshold for processing script execution time. If the execution above this, the output will be highlighted |
| Start\_date | The date that the underlying SQL will read from |
| End\_date | The date that the underlying SQL will read till |

Following is a sample output.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DM processing scripts listed according to execution time** | | | | |
| **MX Date** | **System Date** | **Script name** | **Execution time** | **Highlight** |
| **2012-09-14 00:00:00** | **2014-12-29 00:00:00** | **FEED\_DYN\_CS\_TPPL\_A** | **00:00:27** | **False** |
| **2012-09-14 00:00:00** | **2013-07-23 00:00:00** | **TS\_ERM\_ADON\_RiskComp** | **00:00:16** | **False** |
| **2012-09-14 00:00:00** | **2013-07-23 00:00:00** | **TS\_ERM\_MKT\_RiskComp** | **00:00:14** | **False** |
| **2012-09-13 00:00:00** | **2013-06-27 00:00:00** | **TS\_ERM\_ADON\_RiskComp** | **00:00:14** | **False** |
| **2012-09-13 00:00:00** | **2013-06-27 00:00:00** | **TS\_ERM\_MKT\_RiskComp** | **00:00:10** | **False** |
| **2012-09-13 00:00:00** | **2013-06-27 00:00:00** | **FXCASH\_PL\_FEED** | **00:00:10** | **False** |
| **2012-09-13 00:00:00** | **2013-06-27 00:00:00** | **FXCASH\_DET\_PL\_FEED** | **00:00:09** | **False** |
| **2012-09-13 00:00:00** | **2013-06-27 00:00:00** | **PS\_MX\_PL\_BKD\_D** | **00:00:06** | **False** |
| **2012-09-13 00:00:00** | **2013-06-27 00:00:00** | **PS\_MX\_PL\_BKD\_C** | **00:00:05** | **False** |
| **2012-09-13 00:00:00** | **2013-06-27 00:00:00** | **PS\_MX\_ACC\_BALANCE** | **00:00:02** | **False** |

* 1. Processing script detailed (Sheet: Processing script detailed)

The analysis will produce all the script within the time frame. It lists all the details. The following attachment shows a sample output.



Appendices

I. Query\_dm\_config SQL



II. Query\_sensitivity\_flag



III. Query\_processing\_script\_time



IV. Query\_simulation\_context



V. Query\_dm\_definition

