

Tactical SQL Stored Procedure

Version: 0.1

Created: October 20th, 2016

**Table of Contents**

[1 Overview 3](#_Toc465065115)

[1.1 Executive Summary 3](#_Toc465065116)

[1.2 Technical Specifications 3](#_Toc465065117)

[2 Technical Workflow 4](#_Toc465065118)

[2.1 Scenario Logic 5](#_Toc465065119)

[2.2 Overview 5](#_Toc465065120)

[2.2.1 Carousel Transactions 7](#_Toc465065121)

[2.2.2 Change in Behavior: Trading in New Geographies 14](#_Toc465065122)

[2.2.3 Change in Behavior: Trading in New Goods 19](#_Toc465065123)

[2.3 Python Interface 55](#_Toc465065124)

[Python Packages: 55](#_Toc465065125)

[ODBC connection: 55](#_Toc465065126)

[Setting up Application using Flask 55](#_Toc465065127)

[Python functions for different scenarios 56](#_Toc465065128)

[2.4 Front-end Development 58](#_Toc465065129)

# Scenario Logic

## Overview

Tactical Monitoring Scenarios

The technical design of all the 7 Tactical Monitoring scenarios are based on the general details provided below, which is further customized to serve each individual scenario.

US\_DATA\_MART

Stored Tactical Scenario Procedures

Multi\_Drawings\_Hist

Multi\_LC\_Hist

Transaction Data

execute

Prime

Temporary tables

Alerts

store

Tactical Scenario Web Application

Fetch Alerts

Remote Server

All the latest unique transactions from the LC and Drawings tables from the US\_DATA\_MART are retrieved into the Prime database using a linked server object which establishes a connection between the two databases. The retrieved transactions are stored into temporary[[1]](#footnote-2) LC and Drawings tables. The procedures extract the requisite data from these temporary tables for detecting potential suspicious activities as per the conditions defined in the “Tactical Monitoring Rulebook”.

The subsequent sections below explain the customized version of the above process for each of the seven scenarios.

Note: the terms, Focal Entity and Customer are used interchangeably.

Debit Card Scenarios

The technical design of the 2 Debit Card scenarios, Pattern of Excessive Withdrawals and Foreign Debit Card Transactions, are based on the general data flow details provided below.

store

Prime

Stored Tactical Scenario Procedures

AccountOwner

Customer

Transaction Data

execute

Prime

Temporary tables

Alerts

Tactical Scenario Web Application

Fetch Alerts

Remote Server

Debit\_Transactions

Vantiv Data Report

Data

Data from the Vantiv data report is flown into the Debit\_Transactions table in Prime. The scenario uses data from Debit\_Transactions, AccountOwner, and Customer in Prime to generate alerts that are then flown into the Tactical Scenario Web Application.

## Carousel Transactions

This scenario monitors for carousel transactions i.e. repeated import and export of identical high value goods by a Customer.

The logical flow of the procedure implemented to monitor this scenario is explained below:

1. Input parameters
   1. @LOOKBACKSTART: start of the monitoring period in the format YYYYMMDD
   2. @CURRPERIODSTART: current month start day in the format YYYYMMDD
   3. @CURRPERIODEND: current month end day in the format YYYYMMDD
   4. @CaslHRGoodsTradeThsld: threshold for Total number of carousel trades involving High Risk High Value Goods Type in the monitoring period
   5. @CaslHRDocumentAmtThsld: threshold for sum of Document Amount of carousel trades involving High Risk High Value Goods Type in the monitoring period.
   6. @CaslDUGoodsTradeThsld: threshold for total number of carousel trades involving Dual Use High Value Goods Type in the monitoring period
   7. @CaslDUDocumentAmtThsld: threshold for sum of Document Amount of carousel trades involving Dual Use High Value Goods Type in the monitoring period
   8. @CaslGoodsTradeThsld: threshold for total number of carousel trades with High Value Goods Type in the monitoring period
   9. @CaslDocumentAmtThsld: threshold for sum of Document Amount of carousel trades High Value Goods Type in the monitoring period

**Example:**

@LOOKBACKSTART INT =20151201,

@CURRPERIODSTART INT =20160601,

@CURRPERIODEND INT =20160631,

@CASLHRGOODSTRADETHSLD INT =2,

@CASLDUGOODSTRADETHSLD INT =2,

@CASLGOODSTRADETHSLD INT =2,

@CASLHRDOCUMENTAMTTHSLD DECIMAL(19,2) =100000, --

@CASLDUDOCUMENTAMTTHSLD DECIMAL(19,2) =100000, --

@CASLDOCUMENTAMTTHSLD DECIMAL(19,2) =100000

The customers who made transactions/drawings in the period beginning from @ CURRPERIODSTART and ending at CURRPERIODEND are monitored for Carousel transactions. The transactions made in the current month are compared with the transactions in the period beginning from @ LOOKBACKSTART and ending at @ CURRPERIODEND

1. Data Retrieval: All the latest unique transactions from the LC and Drawings tables from the US\_DATA\_MART are retrieved into the Prime database using a linked server object which establishes a connection between the two databases. The retrieved transactions are stored into temporary tables’ i.e. #MULTI\_LC\_HIST and #MULTI\_DRAWINGS\_HIST.
2. Another temporary table, #CAROUSELTRANSACTIONS, is created to store all the transactions for the monitoring period. The table #MULTI\_DRAWINGS\_HIST and #MULTI\_LC\_HIST are joined together based on the field ID from the LC\_Table and the first 12 digits from the field ID from the Drawings table The data retrieval process for the collection LC Types (LC\_TYPE field value beginning with ‘C’) and the rest of the LC Type differs and both processes are explained below.

Note: in both the cases as mentioned above, the transactions belonging to LC\_TYPE ‘RUS’ and the transfer LC\_TYPES i.e. those which begin with ‘ET’. This condition was included based on the SME judgement as ‘RUS’ do not have required fields as these are not mandatory. The Transfer LCs are of the child LC Types and therefore would have all the field values identical to the parent LC record. Including this field would have resulted in duplication of records as the parent LC Types are monitored by default based on the conditions listed. Additionally, transactions with ‘ZZ’ values for the field ‘ACT\_GOODS\_DESC’ are ignored

* 1. For LC Types other than collections (i.e. recognized by the LC\_TYPE field value which do not begin with ‘C’) the transactions are extracted based on the DOC\_RECE\_DT field value falling between @LOOKBACKSTART and @CURRPERIODEND

Data Mapping: All the fields retrieved from #MULTI\_LC\_HIST and #MULTI\_DRAWINGS\_HIST into the table #CAROUSELTRANSACTIONS for LC type other than collection is as shown below:

|  |  |  |
| --- | --- | --- |
| #CAROUSELTRANSACTIONS | #MULTI\_DRAWINGS\_HIST | #MULTI\_LC\_HIST |
| ID | ID |  |
| FOCALENTITY |  | For import LC types (LC\_TYPE beginning with ‘I’ or ‘SO’), the FOCALENTITY is populated with **APPLICANT\_CUSTNO,** if absent then the field is populated with **APPLICANT**. If both these fields are absent then the field is populated with the **ID**  For export LC types (LC\_TYPES which begin with all other alphabets other than ‘I’ or ‘SO’ ) the FOCALENTITY is populated with **BENEFICIARY\_CUSTNO,** if absent then the field is populated with the **BENEFICIARY**.If both these fields are absent then the field is populated with the **ID** |
| OLD\_LC\_NUMBER |  | OLD\_LC\_NUMBER |
| LC\_TYPE |  | LC\_TYPE |
| HRGOODS | Refer step 4 below  Refer step 4 below | |
| DUGOODS |
| DRAWING\_TYPE | DRAWING\_TYPE |  |
| COUNTRY\_ORIG | COUNTRY\_ORIG |  |
| SHIP\_FM\_COUNTRY | SHIP\_FM\_COUNTRY |  |
| SHIP\_TO\_COUNTRY | SHIP\_TO\_COUNTRY |  |
| DOCUMENT\_AMOUNT | DOCUMENT\_AMOUNT |  |
| LC\_AMOUNT |  | LC\_AMOUNT |
| LC\_NUMBER |  | LC\_NUMBER |
| TRANS\_DATE | DOC\_RECE\_DT |  |
| VALUE\_DATE | VALUE\_DATE |  |
| DATE\_TIME | DATE\_TIME |  |
| APPLICANT |  | APPLICANT |
| APPLICANT\_CUSTNO |  | APPLICANT\_CUSTNO |
| APPLICANT\_ACC |  | APPLICANT\_ACC |
| BENEFICIARY |  | BENEFICIARY |
| BENEFICIARY\_CUSTNO |  | BENEFICIARY\_CUSTNO |
| BENEFICIARY\_ACC |  | BENEFICIARY\_ACC |
| ACT\_GOODS\_DESC | ACT\_GOODS\_DESC |  |
| GOODSTYPE | GOODSTYPE |  |

* 1. For Collection LC Types the transactions are extracted based on the ISSUE\_DATE falling between the @LookBackStart and @CurrPeriodEND

Data Mapping: All the fields retrieved from #MULTI\_LC\_HIST and #MULTI\_DRAWINGS\_HIST into the table #CAROUSELTRANSACTIONS for the collection LC type is as shown below:

|  |  |  |
| --- | --- | --- |
| #CAROUSELTRANSACTIONS | #MULTI\_DRAWINGS\_HIST | #MULTI\_LC\_HIST |
| ID | ID |  |
| FOCALENTITY |  | The FOCALENTITY is populated with **APPLICANT\_CUSTNO,** if absent then the field is populated with **APPLICANT**. If both these fields are absent then the field is populated with the **ID** |
| OLD\_LC\_NUMBER |  | OLD\_LC\_NUMBER |
| HRGOODS | Refer step 4 below | |
| DUGOODS | Refer step 4 below | |
| LC\_TYPE |  | LC\_TYPE |
| DRAWING\_TYPE | DRAWING\_TYPE |  |
| COUNTRY\_ORIG |  | COUNTRY\_ORIG |
| SHIP\_FM\_COUNTRY |  | SHIP\_FM\_COUNTRY |
| SHIP\_TO\_COUNTRY |  | SHIP\_TO\_COUNTRY |
| DOCUMENT\_AMOUNT | DOCUMENT\_AMOUNT |  |
| LC\_AMOUNT |  | LC\_AMOUNT |
| LC\_NUMBER |  | LC\_NUMBER |
| TRANS\_DATE |  | ISSUE\_DATE |
| VALUE\_DATE | VALUE\_DATE |  |
| DATE\_TIME | DATE\_TIME |  |
| APPLICANT |  | APPLICANT |
| APPLICANT\_CUSTNO |  | APPLICANT\_CUSTNO |
| APPLICANT\_ACC |  | APPLICANT\_ACC |
| BENEFICIARY |  | BENEFICIARY |
| BENEFICIARY\_CUSTNO |  | BENEFICIARY\_CUSTNO |
| BENEFICIARY\_ACC |  | BENEFICIARY\_ACC |
| ACT\_GOODS\_DESC |  | ACT\_GOODS\_DESC |
| GOODSTYPE |  | GOODSTYPE |

1. Since the historical transactions, i.e. transaction predating Oct 2016, do not have the field GoodsType populated, the code uses a reference table ‘GOODSREFID’ to populate the goodstype ID based on the field ‘Acts\_Goods\_Desc’. This process will be executed for all the records from the LC and Drawings table until 20170502, after which this code block will not be executed.
2. As the scenario needs to monitor only high value goods, delete all the transactions from the table #CAROUSELTRANSACTIONS which do not have GOODSTYPE in (0010, 0050, 0070, 0090, 0120, 0121, 0130, 0140, 0350). These IDs indicate high value goods type.
3. For the fields HRGoods, DUGoods in the #CAROUSELTRANSACTIONS, join with the table LCGOODSTYPE table on GOODSTYPE and flag (1 if true and 0 if false) the high risk goods and Dual use goods respectively.
4. Replace the GOODTYPE in the #CAROUSELTRANSACTIONS field with GOODS\_DISPLAY field from the LCGOODSTYPE table
5. Create another temporary table #CAROUSELTRANSACTIONS with the same fields as #CAROUSELTRANSACTIONS
6. Insert all the import type LCs (LC Types beginning with ‘I’, ‘SO’ and ‘CI’) into a new table #IMPORT\_PWC\_CAROUSELTRANSACTIONS from #CAROUSELTRANSACTIONS with TRANS\_DATE between @LOOKBACKSTART and @CURRPERIODSTART. This table holds all the import type transactions except those in the current month. Including the current month transaction would lead to duplicate alerts.
7. Insert all the EXPORT type LCs (LC Types beginning with ‘E’,’CE’,’LE,’SI’) into a new table #EXPORT\_PWC\_CAROUSELTRANSACTIONS from #CAROUSELTRANSACTIONS with TRANS\_DATE between @LOOKBACKSTART and @CURRPERIODEND. This table holds all the export type transactions.
8. Insert all the import type LCs (LC Types beginning with ‘I’, ‘SO’ and ‘CI’) belonging to the current month into a new table #IMPORT\_PWC\_CURRCAROUSELTRANSACTIONS from #CAROUSELTRANSACTIONS with TRANS\_DATE between @ CURRPERIODSTART and @CURRPERIODEND. This table holds all the import type transactions for the current month.
9. Insert all the EXPORT type LCs (LC Types beginning with ‘E’,’CE’,’LE,’SI’) belonging to the current month into a new table #EXPORT\_CURRPWC\_CAROUSELTRANSACTIONS from #CAROUSELTRANSACTIONS with TRANS\_DATE between @ CURRPERIODSTART and @CURRPERIODEND. This table holds all the export type transactions for the current month.
10. Create another table #CURRCAROUSELTRANSACTIONS\_ALERTS to store all the alerted transactions
11. Compare all the import type transactions to the export type transactions in the current month using the tables #IMPORT\_PWC\_CAROUSELTRANSACTIONS and #EXPORT\_CURRPWC\_CAROUSELTRANSACTIONS. The tables are joined on FOCALENTITY and GOODSTYPE. If there are any matches, add the transactions to the #CURRCAROUSELTRANSACTIONS\_ALERTS.
12. Compare all the export type transactions to the import type transactions in the current month using the tables #EXPORT\_PWC\_CAROUSELTRANSACTIONS and #IMPORT\_CURRPWC\_CAROUSELTRANSACTIONS. The tables are joined on FOCALENTITY and GOODSTYPE. If there are any matches, add the transactions to the #CURRCAROUSELTRANSACTIONS\_ALERTS.
13. Create table #TT1 with the following table structure.

FOCALENTITY VARCHAR(100),

TRANSCOUNT INT,

HRGOODS INT,

DUGOODS INT,

HRDOCUMENT\_AMOUNT DECIMAL(19,2),

DUDOCUMENT\_AMOUNT DECIMAL(19,2),

TOTDOCUMENT\_AMOUNT DECIMAL(19,2),

ALERTINGCONDITION VARCHAR(50)

1. For each FOCALENTITY insert count the number of transactions (COUNT(FocalEntity)), number of high risk goods (SUM(HRGoods)), number of dual use goods (SUM(DUGoods)) and sum of document amount (SUM(Document\_Amount)) into the table #TT1 respectively using the records in table #CURRCAROUSELTRANSACTIONS\_ALERTS. This is done using a group by function.
2. Update the table and add the sum of document amount for high risk goods (HRDOCUMENT\_AMOUNT) and dual use goods (DUDOCUMENT\_AMOUNT) using the records in table #CURRCAROUSELTRANSACTIONS\_ALERTS.
3. Create another table #CURRCAROUSELTRANSACTIONS\_ALERTS\_2. It has the same table structure as #CAROUSELTRANSACTIONS except for another additional field ‘ALERTINGCONDTITION’. This field will contain a value to indicate the exact conditions for which the FOCALENTITY was alerted.
4. All the transaction which breach the thresholds set for no risk goods, high risk goods and dual use goods are inserted into the table #CURRCAROUSELTRANSACTIONS\_ALERTS\_2.
5. Insert distinct FOCALENTITY and ALERTINGCONDTITION into a new table #TT2 from #CURRCAROUSELTRANSACTIONS\_ALERTS\_2.
6. For each FOCALENTITY, get all the ALERTINGCONDTITIONs and insert these records into #TT3
7. For each focal entity in #TT3, get all the fields from #TT1 by joining on FOCALENTITY, and insert the records into a new table PBSA.DBO.CAROUSELALERTCUSTOMERS. Also, cast all the fields as varchar while inserting them into the table.
8. Insert all the fields from #CURRCAROUSELTRANSACTIONS\_ALERTS\_2 into a new table PBSA.DBO.CAROUSELALERTTRANSACTIONS. Also, cast all the fields as varchar while inserting them into the table. Also, replace the spaces in the field GOODSTYPE by underscore.
9. Create a table PBSA.DBO.CAROUSELGEOMAPDATA and insert all the transactions belonging to all the alerted customers for the entire monitoring period (i.e. all transactions between @LOOKBACKSTART and @CURRPERIODEND) into it.
10. Insert the ‘CAROUSEL\_TRADE’ as ALERT\_NAME, MONTH, ALERTINGCONDITIONS, FOCALENTITY, CUSTOMERNAME into the CONSOLIDATEDALERTS tables
11. Insert the Focalentity/Customer ID, Scenario, Alerting Condition, Customer Name, Alert Month, Run Month into the table ‘CONSOLIDATEDALERTS’ which maintains an archive of all the historical alerts
12. Insert all the transaction ID and other relevant fields into the table ' HISTORICALTRANSACTIONS’ which maintains an archive of all the historical transactions of alerted customers

Note: every time the scenario is run, the tables PBSA.DBO.CAROUSELALERTCUSTOMERS, PBSA.DBO.CAROUSELALERTTRANSACTIONS and PBSA.DBO.CAROUSELGEOMAPDATA are deleted, if present, at the beginning by the procedure. These tables are later created again so as to contain only those alerts w.r.t the run period.

## Change in Behavior: Trading in New Geographies

This scenario identifies activity which does not match a customer’s profile or historical activity, w.r.t to the trading countries.

The logical flow of the procedure implemented to monitor this scenario is explained below:

1. Input parameters
   1. @LOOKBACKSTART: start of the lookback period in the format YYYYMMDD
   2. @LOOKBACKEND: end of the lookback period in the format YYYYMMDD
   3. @CURRPERIODSTART: current month start day in the format YYYYMMDD
   4. @CURRPERIODEND: current month end day in the format YYYYMMDD
   5. @NewShipFMThsld: threshold for total number of trades involving a new Ship From Country for a given customer in the scenario run period
   6. @NewShipTOThsld: threshold for total number of trades involving a new Ship to Country for a given customer in the scenario run period
   7. @NewOrigThsld: threshold for total number of trades involving a new Country of Origin for a given customer in the scenario run period
   8. @HRCustNWGeo: threshold for total number of trades involving new Country (Ship From/Ship To/Country of Origin) for a given High Risk customer in the scenario run period

**Example:**

@LOOKBACKSTART INT= 20151201,

@LOOKBACKEND INT= 20160531,

@CURRPERIODSTART INT= 20160601,

@CURRPERIODEND INT= 20160631,

@NEWSHIPFMTHSLD INT= 3,

@NEWORIGTHSLD INT= 3,

@NEWSHIPTOTHSLD INT= 3,

@HRCUSTNWGEO INT= 1

If the customer trades with a country in the current period (i.e. period between CURRPERIODSTART and @ CURRPERIODEND) and with which it hadn’t traded during the look back period (i.e. period between @LOOKBACKSTART and @LOOKBACKEND) and breaches the rest of tunable thresholds, it’ll be alerted

1. Step 2 to 4 are same as that of Carousel Transactions except for the following two differences:
   1. Rather than adding the data into #CAROUSELTRANSACTIONS, the records are inserted into table #TRADEINNEWGEO.
   2. Transactions are not excluded if the ACT\_GOODS\_DESC field contains ‘ZZ’ values.

Data Mapping: All the fields retrieved from #MULTI\_LC\_HIST and #MULTI\_DRAWINGS\_HIST into the table #TRADEINNEWGEO for LC type other than collection is as shown below:

|  |  |  |
| --- | --- | --- |
| # TRADEINNEWGEO | #MULTI\_DRAWINGS\_HIST | #MULTI\_LC\_HIST |
| ID | ID |  |
| FOCALENTITY |  | For import LC types (LC\_TYPE beginning with ‘I’ or ‘SO’), the FOCALENTITY is populated with **APPLICANT\_CUSTNO,** if absent then the field is populated with **APPLICANT**. If both these fields are absent then the field is populated with the **ID**  For export LC types (LC\_TYPES which begin with all other alphabets other than ‘I’ or ‘SO’ ) the FOCALENTITY is populated with **BENEFICIARY\_CUSTNO,** if absent then the field is populated with the **BENEFICIARY**.If both these fields are absent then the field is populated with the **ID** |
| OLD\_LC\_NUMBER |  | OLD\_LC\_NUMBER |
| LC\_TYPE |  | LC\_TYPE |
| HIGHRISKCUSTOMER | Refer step 12 |  |
| NEWSHIPFMCOUNTRY | Refer step 7 |  |
| NEWORIGCOUNTRY | Refer step 9 |  |
| NEWSHIPTOCOUNTRY | Refer step 11 |  |
| DRAWING\_TYPE | DRAWING\_TYPE |  |
| COUNTRY\_ORIG | COUNTRY\_ORIG |  |
| SHIP\_FM\_COUNTRY | SHIP\_FM\_COUNTRY |  |
| SHIP\_TO\_COUNTRY | SHIP\_TO\_COUNTRY |  |
| VESSEL\_NAME | VESSEL\_NAME |  |
| VOYAGE\_NUMBER | VOYAGE\_NUMBER |  |
| DOCUMENT\_AMOUNT | DOCUMENT\_AMOUNT |  |
| LC\_AMOUNT |  | LC\_AMOUNT |
| LC\_NUMBER |  | LC\_NUMBER |
| TRANS\_DATE | DOC\_RECE\_DT |  |
| VALUE\_DATE | VALUE\_DATE |  |
| DATE\_TIME | DATE\_TIME |  |
| APPLICANT |  | APPLICANT |
| APPLICANT\_CUSTNO |  | APPLICANT\_CUSTNO |
| APPLICANT\_ACC |  | APPLICANT\_ACC |
| BENEFICIARY |  | BENEFICIARY |
| BENEFICIARY\_CUSTNO |  | BENEFICIARY\_CUSTNO |
| BENEFICIARY\_ACC |  | BENEFICIARY\_ACC |
| ACT\_GOODS\_DESC | ACT\_GOODS\_DESC |  |
| GOODSTYPE | GOODSTYPE |  |

Data Mapping: All the fields retrieved from #MULTI\_LC\_HIST and #MULTI\_DRAWINGS\_HIST into the table #TRADEINNEWGEO for the collection LC type is as shown below:

|  |  |  |
| --- | --- | --- |
| # TRADEINNEWGEO | #MULTI\_DRAWINGS\_HIST | #MULTI\_LC\_HIST |
| ID | ID |  |
| FOCALENTITY |  | The FOCALENTITY is populated with **APPLICANT\_CUSTNO,** if absent then the field is populated with **APPLICANT**. If both these fields are absent then the field is populated with the **ID** |
| OLD\_LC\_NUMBER |  | OLD\_LC\_NUMBER |
| LC\_TYPE |  | LC\_TYPE |
| HIGHRISKCUSTOMER | Refer step 12 |  |
| NEWSHIPFMCOUNTRY | Refer step 7 |  |
| NEWORIGCOUNTRY | Refer step 9 |  |
| NEWSHIPTOCOUNTRY | Refer step 11 |  |
| DRAWING\_TYPE | DRAWING\_TYPE |  |
| COUNTRY\_ORIG |  | COUNTRY\_ORIG |
| SHIP\_FM\_COUNTRY |  | SHIP\_FM\_COUNTRY |
| SHIP\_TO\_COUNTRY |  | SHIP\_TO\_COUNTRY |
| VESSEL\_NAME |  | VESSEL\_NAME |
| VOYAGE\_NUMBER |  | VOYAGE\_NUMBER |
| DOCUMENT\_AMOUNT | DOCUMENT\_AMOUNT |  |
| LC\_AMOUNT |  | LC\_AMOUNT |
| LC\_NUMBER |  | LC\_NUMBER |
| TRANS\_DATE |  | ISSUE\_DATE |
| VALUE\_DATE | VALUE\_DATE |  |
| DATE\_TIME | DATE\_TIME |  |
| APPLICANT |  | APPLICANT |
| APPLICANT\_CUSTNO |  | APPLICANT\_CUSTNO |
| APPLICANT\_ACC |  | APPLICANT\_ACC |
| BENEFICIARY |  | BENEFICIARY |
| BENEFICIARY\_CUSTNO |  | BENEFICIARY\_CUSTNO |
| BENEFICIARY\_ACC |  | BENEFICIARY\_ACC |
| ACT\_GOODS\_DESC |  | ACT\_GOODS\_DESC |
| GOODSTYPE |  | GOODSTYPE |

1. Replace the GOODTYPE in the #CAROUSELTRANSACTIONS field with GOODS\_DISPLAY field from the LCGOODSTYPE table
2. Create new table #CURRTRADEINNEWGEO and insert into it the records belonging to the current period (i.e. TRANS\_DATE between @CURRPERIODSTART and @CURRPERIODEND).
3. Create a new table #CUSTSHIPFMCTRYPROFILE and insert the FOCALENTITY and the all the SHIP\_FM\_COUNTRY values from the #TRADEINNEWGEO from the lookback period (i.e. TRANS\_DATE between @LOOKBACKPERIODSTART and @LOOKBACKPERIODEND). This table will hold the historical ship from country profile for all customers.
4. Compare the SHIP\_FM\_COUNTRY in the #CURRTRADEINNEWGEO and the #CUSTSHIPFMCTRYPROFILE for each FOCALENTITY and if there is no match and set the flag NEWSHIPFMCOUNTRY to 1. In case there is a match, set it to 0.
5. Create a new table #CUSTSHIPTOCTRYPROFILE and insert the FOCALENTITY and the all the SHIP\_TO\_COUNTRY values from the #TRADEINNEWGEO from the lookback period (i.e. TRANS\_DATE between @LOOKBACKPERIODSTART and @LOOKBACKPERIODEND). This table will hold the historical ship to country profile for customers
6. Compare the SHIP\_TO\_COUNTRY in the #CURRTRADEINNEWGEO and the #CUSTSHIPFMCTRYPROFILE for each FOCALENTITY and if there is no match and set the flag NEWSHIPTOCOUNTRY to 1. In case there is a match, set it to 0.
7. Create a new table #CURRTRADEINNEWGEO and insert the FOCALENTITY and the all the COUNTRY\_ORIG values from the #TRADEINNEWGEO from the lookback period (i.e. TRANS\_DATE between @LOOKBACKPERIODSTART and @LOOKBACKPERIODEND). This table will hold the historical origin country profile for customers
8. Compare the COUNTRY\_ORIG in the #CURRTRADEINNEWGEO and the #CUSTSHIPFMCTRYPROFILE for each FOCALENTITY and if there is no match and set the flag NEWORIGCOUNTRY to 1. In case there is a match, set it to 0.

Note: in steps 7, 9 and 11, no flags are set if the respective county field has the value ‘ZZ’ or is NULL.

1. Using the FOCALENTITY from the #CURRTRADEINNEWGEO check if the customer is rated as high risk in the CUSTOMER table. Set the flag HIGHRISKCUSTOMER to 1 if the customer is High risk, else set it to 0.
2. Create a table #TT1 with the following structure:

FOCALENTITY VARCHAR(50),

HIGHRISKCUSTOMER INT DEFAULT 0,

NEWSHIPFMCOUNTRY INT DEFAULT 0,

NEWORIGCOUNTRY INT DEFAULT 0,

NEWSHIPTOCOUNTRY INT DEFAULT 0)

1. For each FOCALENTITY, get the HIGHRISKCUSTOMER flag (MAX(CAST(HIGHRISKCUSTOMER AS INT))), total number of SHIP\_FM\_COUNTRY (SUM(NEWSHIPFMCOUNTRY)), COUNTRY\_ORIG (SUM(NEWORIGCOUNTRY)) and SHIP\_TO\_COUNTRY (SUM(NEWSHIPTOCOUNTRY)) from the #CURRTRADEINNEWGEO table and insert the respective values into the table #TT1. This is accomplished using a group by function.
2. Create a new table #TRADEINNEWGEO\_ALERTS which has the same structure as #CURRTRADEINNEWGEO except for an addition field ALERTINGCONDTITION. This field will contain a value to indicate the exact conditions for which the FOCALENTITY was alerted.
3. All the transaction which breach the thresholds for new ship from country, new ship to country, origin country and new country (new ship from, ship to and origin) for a high risk customer are inserted into the table #TRADEINNEWGEO\_ALERTS.
4. Insert distinct FOCALENTITY and ALERTINGCONDTITION into a new table #TT2 from #TRADEINNEWGEO\_ALERTS.
5. For each FOCALENTITY, get all the ALERTINGCONDTITIONs and insert these records into #TT3
6. For each focal entity in #TT3, get all the fields from #TT1 by joining on FOCALENTITY, and insert the records into a new table PBSA.DBO. TRADEINNEWGEOALERTCUSTOMERS. Also, cast all the fields as varchar while inserting them into the table.
7. Insert all the fields from #TRADEINNEWGEO\_ALERTS into a new table PBSA.DBO. TRADEINNEWGEOALERTTRANSACTIONS.
8. Create a table PBSA.DBO. TRADEINNEWGEOMAPDATA and insert all the transactions belonging to all the alerted customers for the entire monitoring period (i.e. all transactions between @LOOKBACKSTART and @CURRPERIODEND) into it. Replace all the spaces in the fields which have them, with underscore ‘\_’
9. Insert ‘TRADE\_IN\_NEW\_GEO' as ALERT\_NAME, MONTH, ALERTINGCONDITIONS, FOCALENTITY, CUSTOMERNAME into the CONSOLIDATEDALERTS tables. Thius table maintains the historical alerts.
10. Insert the Focalentity/Customer ID, Scenario, Alerting Condition, Customer Name, Alert Month, Run Month into the table ‘CONSOLIDATEDALERTS’ which maintains an archive of all the historical alerts
11. Insert all the transaction ID and other relevant fields into the table ' HISTORICALTRANSACTIONS’ which maintains an archive of all the historical transactions of alerted customers

Note: every time the scenario is run, the tables PBSA.DBO. TRADEINNEWGEOALERTCUSTOMERS, PBSA.DBO. TRADEINNEWGEOALERTTRANSACTIONS and PBSA.DBO. TRADEINNEWGEOMAPDATA are deleted, if present, at the beginning by the procedure. These tables are later created again so as to contain only those alerts w.r.t the run period.

## Change in Behavior: Trading in New Goods

Periodic report generated to identify activity which does not match a customer’s historical behavior with respect to traded goods

The logical flow of the procedure implemented to monitor this scenario is explained below:

1. Input parameters
   1. @LOOKBACKSTART: start of the lookback period in the format YYYYMMDD
   2. @LOOKBACKEND: end of the lookback period in the format YYYYMMDD
   3. @CURRPERIODSTART: current month start day in the format YYYYMMDD
   4. @CURRPERIODEND: current month end day in the format YYYYMMDD
   5. @HRGoodsThsld: threshold for total number of trades in the current period involving a new High Risk Goods Type for a given customer
   6. @DUGoodsThsld: threshold for total number of trades in the current period involving a new Dual Use Goods Type for a given customer
   7. @NewGoodsHRCustThsld: threshold for total number of trades in the current period involving a new Goods Type/s for a given High Risk customer
   8. @NewGoodsThsld: threshold for total number of trades in the current period involving a new Goods Type/s for a given customer

**Example:**

@LOOKBACKSTART INT = 20151101,

@LOOKBACKEND INT = 20160431,

@CURRPERIODSTART INT = 20160501, --

@CURRPERIODEND INT = 20160531,

@HRGOODSTHSLD INT = 1,

@DUGOODSTHSLD INT = 1,

@NEWGOODSTHSLD INT = 1,

@NEWGOODSHRCUSTTHSLD INT = 1

If the customer trades in a goods type in the current period (i.e. period between CURRPERIODSTART and @ CURRPERIODEND) and in which it hadn’t traded during the look back period (i.e. period between @LOOKBACKSTART and @LOOKBACKEND) and breaches the rest of tunable thresholds, it’ll be alerted

1. Step 2 to 4 are same as that of Carousel Transactions except for the following difference:
   1. Rather than adding the data into #CAROUSELTRANSACTIONS, the records are inserted into table #TRADEINNWHRDUGOODS

Data Mapping: All the fields retrieved from #MULTI\_LC\_HIST and #MULTI\_DRAWINGS\_HIST into the table #TRADEINNEWGEO for LC type other than collection is as shown below:

|  |  |  |
| --- | --- | --- |
| # TRADEINNWHRDUGOODS | #MULTI\_DRAWINGS\_HIST | #MULTI\_LC\_HIST |
| ID | ID |  |
| FOCALENTITY |  | For import LC types (LC\_TYPE beginning with ‘I’ or ‘SO’), the FOCALENTITY is populated with **APPLICANT\_CUSTNO,** if absent then the field is populated with **APPLICANT**. If both these fields are absent then the field is populated with the **ID**  For export LC types (LC\_TYPES which begin with all other alphabets other than ‘I’ or ‘SO’ ) the FOCALENTITY is populated with **BENEFICIARY\_CUSTNO,** if absent then the field is populated with the **BENEFICIARY**.If both these fields are absent then the field is populated with the **ID** |
| OLD\_LC\_NUMBER |  | OLD\_LC\_NUMBER |
| LC\_TYPE |  | LC\_TYPE |
| DRAWING\_TYPE | DRAWING\_TYPE |  |
| COUNTRY\_ORIG | COUNTRY\_ORIG |  |
| SHIP\_FM\_COUNTRY | SHIP\_FM\_COUNTRY |  |
| SHIP\_TO\_COUNTRY | SHIP\_TO\_COUNTRY |  |
| VESSEL\_NAME | VESSEL\_NAME |  |
| VOYAGE\_NUMBER | VOYAGE\_NUMBER |  |
| DOCUMENT\_AMOUNT | DOCUMENT\_AMOUNT |  |
| LC\_AMOUNT |  | LC\_AMOUNT |
| LC\_NUMBER |  | LC\_NUMBER |
| TRANS\_DATE | DOC\_RECE\_DT |  |
| VALUE\_DATE | VALUE\_DATE |  |
| DATE\_TIME | DATE\_TIME |  |
| APPLICANT |  | APPLICANT |
| APPLICANT\_CUSTNO |  | APPLICANT\_CUSTNO |
| APPLICANT\_ACC |  | APPLICANT\_ACC |
| BENEFICIARY |  | BENEFICIARY |
| BENEFICIARY\_CUSTNO |  | BENEFICIARY\_CUSTNO |
| BENEFICIARY\_ACC |  | BENEFICIARY\_ACC |
| ACT\_GOODS\_DESC | ACT\_GOODS\_DESC |  |
| GOODSTYPE | GOODSTYPE |  |

Data Mapping: All the fields retrieved from #MULTI\_LC\_HIST and #MULTI\_DRAWINGS\_HIST into the table #TRADEINNEWGEO for the collection LC type is as shown below:

|  |  |  |
| --- | --- | --- |
| # TRADEINNWHRDUGOODS | #MULTI\_DRAWINGS\_HIST | #MULTI\_LC\_HIST |
| ID | ID |  |
| FOCALENTITY |  | The FOCALENTITY is populated with **APPLICANT\_CUSTNO,** if absent then the field is populated with **APPLICANT**. If both these fields are absent then the field is populated with the **ID** |
| OLD\_LC\_NUMBER |  | OLD\_LC\_NUMBER |
| LC\_TYPE |  | LC\_TYPE |
| DRAWING\_TYPE | DRAWING\_TYPE |  |
| COUNTRY\_ORIG |  | COUNTRY\_ORIG |
| SHIP\_FM\_COUNTRY |  | SHIP\_FM\_COUNTRY |
| SHIP\_TO\_COUNTRY |  | SHIP\_TO\_COUNTRY |
| VESSEL\_NAME |  | VESSEL\_NAME |
| VOYAGE\_NUMBER |  | VOYAGE\_NUMBER |
| DOCUMENT\_AMOUNT | DOCUMENT\_AMOUNT |  |
| LC\_AMOUNT |  | LC\_AMOUNT |
| LC\_NUMBER |  | LC\_NUMBER |
| TRANS\_DATE |  | ISSUE\_DATE |
| VALUE\_DATE | VALUE\_DATE |  |
| DATE\_TIME | DATE\_TIME |  |
| APPLICANT |  | APPLICANT |
| APPLICANT\_CUSTNO |  | APPLICANT\_CUSTNO |
| APPLICANT\_ACC |  | APPLICANT\_ACC |
| BENEFICIARY |  | BENEFICIARY |
| BENEFICIARY\_CUSTNO |  | BENEFICIARY\_CUSTNO |
| BENEFICIARY\_ACC |  | BENEFICIARY\_ACC |
| ACT\_GOODS\_DESC |  | ACT\_GOODS\_DESC |
| GOODSTYPE |  | GOODSTYPE |

1. Create new table #CURRTRADEINNWHRDUGOODS and insert into it the records belonging to the current period (i.e. TRANS\_DATE between @CURRPERIODSTART and @CURRPERIODEND) from table #TRADEINNWHRDUGOODS.
2. Create a new table #CUSTGOODSPROFILE and insert the FOCALENTITY and the all the GOODSTYPE values from the #CURRTRADEINNWHRDUGOODS from the lookback period (i.e. TRANS\_DATE between @LOOKBACKPERIODSTART and @LOOKBACKPERIODEND). This table will hold the historical goods type profile for all customers.
3. Alter the table #CURRTRADEINNWHRDUGOODS by adding four new columns NEWGOODS, HRGOODS, DUGOODS and HIGHRISKCUSTOMER. These fields act as flag, indicating new goods type, high risk goods type and high risk customer
4. Compare the GOODTYPE in the # CURRTRADEINNWHRDUGOODS and the # CUSTGOODSPROFILE for each FOCALENTITY and if there is no match and set the flag NEWGOODS to 1. In case there is a match, set it to 0.
5. For all the transaction flagged as NEWGOODS, set HRGOODS and DUGOODS to 1 if the GOODSTYPE is a high risk or a dual use goods type. This is done by joining the tables #CURRTRADEINNWHRDUGOODS and LCGOODSTYPE on GOODSTYPE field.
6. Using the FOCALENTITY from the # CURRTRADEINNWHRDUGOODS check if the customer is rated as high risk in the CUSTOMER table. Set the flag HIGHRISKCUSTOMER to 1 if the customer is High risk, else set it to 0.
7. Replace the GOODTYPE in the # CURRTRADEINNWHRDUGOODS and # TRADEINNWHRDUGOODS field with GOODS\_DISPLAY field from the LCGOODSTYPE table
8. Create a table #TT1 with the following structure:

FOCALENTITY VARCHAR(50),

NEWGOODS INT DEFAULT 0,

HRGOODS INT DEFAULT 0,

DUGOODS INT DEFAULT 0,

HIGHRISKCUSTOMER INT DEFAULT 0,

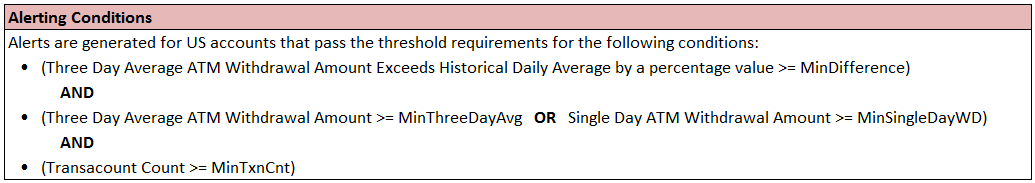
ALERTINGCONDTITION VARCHAR (25))

1. Using table # CURRTRADEINNWHRDUGOODS, for each FOCALENTITY, get the number of transactions with new goods (SUM(NEWGOODS)), with high risk goods (SUM(HRGOODS)), with dual use goods (SUM(DUGOODS)), flag value of the field HIGHRISKCUSTOMER (MAX(HIGHRISKCUSTOMER)) using a group by function and insert the data into #TT1 table.
2. Create a new table #TRADEINNWHRDUGOODS\_ALERTS which has the same structure as # CURRTRADEINNWHRDUGOODS except for an addition field ALERTINGCONDTITION. This field will contain a value to indicate the exact conditions for which the FOCALENTITY was alerted.
3. All the transactions which breach the thresholds for new goods type , new high risk goods type, new dual use goods type and total new goods type for a high risk customer are inserted into the table #TRADEINNWHRDUGOODS\_ALERTS. This is done by joining the # TT1 table with # CURRTRADEINNWHRDUGOODS on FOCALENTITY.
4. Insert distinct FOCALENTITY and ALERTINGCONDTITION into a new table #TT2 from # TRADEINNWHRDUGOODS\_ALERTS.
5. For each FOCALENTITY, get all the ALERTINGCONDTITIONs and insert these records into #TT3
6. For each focal entity in #TT3, get all the fields from #TT1 by joining on FOCALENTITY, and insert the records into a new table PBSA.DBO. TRADEINNWHRDUALERTCUSTOMERS.
7. Insert all the fields from #TRADEINNWHRDUGOODS\_ALERTS into a new table PBSA.DBO. TRADEINNWHRDUALERTTRANSACTIONS. Also, cast all the fields as varchar while inserting them into the table.
8. Create a table PBSA.DBO. TRADEINNEWGOODSMAPDATA and insert all the transactions belonging to all the alerted customers for the entire monitoring period (i.e. all transactions between @LOOKBACKSTART and @CURRPERIODEND) into it.
9. Insert the Focalentity/Customer ID, Scenario, Alerting Condition, Customer Name, Alert Month, Run Month into the table ‘CONSOLIDATEDALERTS’ which maintains an archive of all the historical alerts
10. Insert all the transaction ID and other relevant fields into the table ' HISTORICALTRANSACTIONS’ which maintains an archive of all the historical transactions of alerted customers

Note: every time the scenario is run, the tables PBSA.DBO. TRADEINNWHRDUALERTCUSTOMERS and PBSA.DBO. TRADEINNWHRDUALERTTRANSACTIONS are deleted if present at the beginning by the procedure. These tables are later created again so as to contain only those alerts w.r.t the run period.

## Pattern of Excessive Withdrawals

Scenario Description: This scenario generates alerts for US accounts with a sudden increase in debit card activity. The following alerting conditions must be satisfied for an alert to be generated during a given monitoring month.



The logical flow of the procedure implemented to monitor this scenario is explained below. In the stored procedure, the following parameters and tables are created to retrieve data and perform calculations in order to identify accounts that should be alerted.

1. **Input Parameters**
   1. @YearStart: Start date of the lookback period, which is the start of the month that is 1 year before the end of the current monitoring month
   2. @YearEnd: End date of the lookback period, which is the day before the start of the current monitoring month
   3. @TxnDateStart: Start date of the current monitoring month
   4. @TxnDateEnd: End date of the current monitoring month
   5. @MinSingleDayWD: Minimum Single Day Withdrawal threshold, which is $500
   6. @MinThreeDayAvg: Minimum Three Day Average threshold, which is $300
   7. @MinDifference: Minimum Difference threshold, which is 50%
   8. @MinTxnCnt: Minimum Transaction Count threshold, which is 2

Example of Parameter Values for Monitoring Month of July 2016:

@YearStart DATETIME = '2015-08-01',

@YearEnd DATETIME = '2016-06-30',

@TxnDateStart DATETIME = '2016-07-01',

@TxnDateEnd DATETIME = '2016-07-31',

@MinSingleDayWD DECIMAL(19,2) = 500.00,

@MinThreeDayAvg DECIMAL(19,2) = 300.00,

@MinDifference DECIMAL(6,4) = 0.500,

@MinTxnCnt INT = 2

The values for @YearStart, @YearEnd, @TxnDateStart, and @TxnDateEnd need to be updated during stored procedure execution or changed within the procedure itself to reflect the updated dates when the scenario is run each month.

1. **#ExcessiveWithdrawals\_Tran Table**: This temporary table is used to store all the relevant transactions for this scenario by pulling the following data fields. These data fields are retrieved from the Vantiv data imported into the Debit\_Transactions table, as well as the AccountOwner and Customer tables in Prime. These are the relevant data fields for the scenario that calculations in other tables are based on.

|  |  |
| --- | --- |
| #ExcessiveWithdrawals\_Tran | Data Field Source |
| ID | Created as a unique ID identifier |
| Transaction Date | Debit\_Transactions |
| Terminal Country | Debit\_Transactions |
| Transaction Type | Debit\_Transactions |
| Transaction Amount | Debit\_Transactions |
| Transaction Status | Debit\_Transactions |
| From Account Number | Debit\_Transactions; ‘0’ is added before each account number for standardization with Prime |
| Customer ID | AccountOwner |
| Customer Name | Customer |
| Relationship | AccountOwner |
| Customer Residence | Customer |

This table is used to store all authorized and completed ATM cash withdrawal transactions for US accounts for the current alert month. In order to select this data, the following filters are applied to retrieve the relevant data for this scenario:

1. Transaction Date is in current monitoring month (between @TxnDateStart and @TxnDateEnd)
2. Transaction Type = Cash Withdrawal
3. Transaction Status = Authorized and Completed
4. Customer’s Country of Residence = US
5. **#ExcessiveWithdrawals\_HistTran Table**: This temporary table is used to store all the relevant historical transactions by retrieving all authorized and completed ATM cash withdrawal transactions for US accounts for the historical lookback period of 1 year.  
     
   This table has the same data fields and filters as the #ExcessiveWithdrawals\_Tran table, except the Transaction Date filter is used for the historical lookback period (between @YearStart and @YearEnd)
6. **#ExcessiveWithdrawals\_HistAvg Table**: This temporary table is used to calculate the Historical Daily Average ATM Withdrawal Amount for each account. It uses the transaction data in #ExcessiveWithdrawals\_HistTran to calculate the Historical Average for each account using the definition below.
7. **#ExcessiveWithdrawals\_TxnCnt Table**: This temporary table is used to calculate the Transaction Count during the alert month for each account. It uses the transaction data in #ExcessiveWithdrawals\_Tran to calculate the number of transactions each account has in the alert month.
8. **#ExcessiveWithdrawals\_SingleDayWD Table**: This temporary table is used to calculate the Single Day ATM Withdrawal Amount during the alert month for each account. It uses the transaction data in #ExcessiveWithdrawals\_Tran to calculate the aggregate transaction amount that each account withdraws on each transaction date.
9. **#ExcessiveWithdrawals\_ThreeDayAvg Table**: This temporary table is used to calculate the Three Day Average ATM Withdrawal Amount during the alert month for each account. It uses the transaction data in #ExcessiveWithdrawals\_SingleDayWD because this table has transactions aggregated for each account and transaction date. Three Day Average is calculated for each account and transaction date according to the definition below.
10. **#ExcessiveWithdrawals\_Max Table**: This temporary table is used to calculate the highest Single Day Withdrawal and Three Day Average during the alert month for each account. The rationale behind using the highest value for each of these parameters is that the alerting condition for Minimum Single Day Withdrawal and Three Day Average is satisfied when an account exceeds these thresholds for any day during a given alert month. This table uses the calculations from the #ExcessiveWithdrawals\_ThreeDayAvg and #ExcessiveWithdrawals\_SingleDayWD tables to calculate the highest Single Day Withdrawal and Three Day Average for each account.
11. **#ExcessiveWithdrawals\_Acct Table**: This temporary table is used to calculate Difference and Alert Flag, as well as display account data and calculations from previous tables for each account. Difference is calculated for each account as the percentage difference between Maximum Three Day Average and Historical Average. Alert Flag has a value of 1 when the account satisfies each of the three alerting conditions, and has a value of 0 when at least one of the three alerting conditions is not satisfied. This table uses data and calculations from #ExcessiveWithdrawals\_Max, #ExcessiveWithdrawals\_HistAvg, and #ExcessiveWithdrawals\_TxnCnt, and #ExcessiveWithdrawals\_Tran.
12. **ExcessiveWithdrawals\_Alert Table**: This table is used to display data and calculations for alerted accounts. It has the same data fields as #ExcessiveWithdrawals\_Acct, but only includes information for alerted accounts.
13. **#ExcessiveWithdrawals\_AlertedTxn1 Table**: This table is used to store all transactions that triggered alerts for alerted accounts during the current alert month. It uses data and calculations from the #ExcessiveWithdrawals\_Tran, #ExcessiveWithdrawals\_HistAvg, #ExcessiveWithdrawals\_ThreeDayAvg, #ExcessiveWithdrawals\_SingleDayWD, and #ExcessiveWithdrawals\_TxnCnt tables to determine if a transaction satisfies any of the alerting conditions for an alerted account.
14. **ExcessiveWithdrawals\_AlertedTxn Table**: This table is used to store all transactions for alerted accounts during the current alert month. It uses data from the #ExcessiveWithdrawals\_Tran table.
15. **ExcessiveWithdrawals\_Alert1YearTxn Table**: This table is used to store all of the transactions for alerted accounts during the current alert month, as well as the historical lookback period. This table uses the #ExcessiveWithdrawals\_Tran and #ExcessiveWithdrawals\_HistTran tables to retrieve all transactions during 1 year for alerted accounts.
16. **DebitCard\_AlertArchive Table**: This table is used as an archive to store data for alerted accounts for both Debit Card scenarios (Pattern of Excessive Withdrawals and Foreign Debit Card) each time the scenarios are run. This table uses the ExcessiveWithdrawals\_Alert table to retrieve alert data.
17. **DebitCard\_TranArchive Table**: This table is used as an archive to store transactions for alerted accounts for both Debit Card Scenarios (Pattern of Excessive Withdrawals and Foreign Debit Card) each time the scenarios are run. This table uses the ExcessiveWithdrawals\_Alert1YearTxn table to retrieve relevant transaction data for each alerted account.

## Discrepancy in Vessel or Shipment Information

Scenario Description: Identifies potential discrepancies in Shipment information provided to Trade Services

The logical flow of the procedure implemented to monitor this scenario is explained below:

1. Input parameters
   1. @CURRPERIODSTART: current month start day in the format YYYYMMDD
   2. @CURRPERIODEND: current month end day in the format YYYYMMDD

**Example:**

@ CURRPERIODSTART = 20160601

@ CURRPERIODEND = 20160631

The scenario has 4 alerting conditions. Three of the alerting conditions depend on the following 5 fields: Ship To Country, Ship From Country, Origin Country, Vessel Name and Voyage Number. If transactions belonging to multiple Customers have identical values for 4 of the attributes and the value for a fifth parameter differs, then respective customers and transactions are alerted. W.r.t the fourth alerting condition, if multiple transactions made by a customer has identical values for transaction date (i.e. Issue date for collection LC types and Document receive date for the rest of the LC types) and vessel name but the voyager number differs, then the respective customer and the transactions are alerted.

1. Step 2 to 4 are same as that of Carousel Transactions except for the following difference:
   1. Rather than adding the data into #CAROUSELTRANSACTIONS, the records are inserted into table #SHIPMENT\_DISCREPANCY

Data Mapping: All the fields retrieved from #MULTI\_LC\_HIST and #MULTI\_DRAWINGS\_HIST into the table #TRADEINNEWGEO for LC type other than collection is as shown below:

|  |  |  |
| --- | --- | --- |
| #MULTIPLEINVOICE | #MULTI\_DRAWINGS\_HIST | #MULTI\_LC\_HIST |
| ID | ID |  |
| FOCALENTITY |  | For import LC types (LC\_TYPE beginning with ‘I’ or ‘SO’), the FOCALENTITY is populated with **APPLICANT\_CUSTNO,** if absent then the field is populated with **APPLICANT**. If both these fields are absent then the field is populated with the **ID**  For export LC types (LC\_TYPES which begin with all other alphabets other than ‘I’ or ‘SO’ ) the FOCALENTITY is populated with **BENEFICIARY\_CUSTNO,** if absent then the field is populated with the **BENEFICIARY**.If both these fields are absent then the field is populated with the **ID** |
| OLD\_LC\_NUMBER |  | OLD\_LC\_NUMBER |
| LC\_TYPE |  | LC\_TYPE |
| DRAWING\_TYPE | DRAWING\_TYPE |  |
| COUNTRY\_ORIG | COUNTRY\_ORIG |  |
| SHIP\_FM\_COUNTRY | SHIP\_FM\_COUNTRY |  |
| SHIP\_TO\_COUNTRY | SHIP\_TO\_COUNTRY |  |
| VESSEL\_NAME | VESSEL\_NAME |  |
| VOYAGE\_NUMBER | VOYAGE\_NUMBER |  |
| DOCUMENT\_AMOUNT | DOCUMENT\_AMOUNT |  |
| LC\_AMOUNT |  | LC\_AMOUNT |
| LC\_NUMBER |  | LC\_NUMBER |
| TRANS\_DATE | DOC\_RECE\_DT |  |
| VALUE\_DATE | VALUE\_DATE |  |
| DATE\_TIME | DATE\_TIME |  |
| APPLICANT |  | APPLICANT |
| APPLICANT\_CUSTNO |  | APPLICANT\_CUSTNO |
| APPLICANT\_ACC |  | APPLICANT\_ACC |
| BENEFICIARY |  | BENEFICIARY |
| BENEFICIARY\_CUSTNO |  | BENEFICIARY\_CUSTNO |
| BENEFICIARY\_ACC |  | BENEFICIARY\_ACC |
| ACT\_GOODS\_DESC | ACT\_GOODS\_DESC |  |
| GOODSTYPE | GOODSTYPE |  |

Data Mapping: All the fields retrieved from #MULTI\_LC\_HIST and #MULTI\_DRAWINGS\_HIST into the table #TRADEINNEWGEO for the collection LC type is as shown below:

|  |  |  |
| --- | --- | --- |
| #CAROUSELTRANSACTIONS | #MULTI\_DRAWINGS\_HIST | #MULTI\_LC\_HIST |
| ID | ID |  |
| FOCALENTITY |  | The FOCALENTITY is populated with **APPLICANT\_CUSTNO,** if absent then the field is populated with **APPLICANT**. If both these fields are absent then the field is populated with the **ID** |
| OLD\_LC\_NUMBER |  | OLD\_LC\_NUMBER |
| LC\_TYPE |  | LC\_TYPE |
| DRAWING\_TYPE | DRAWING\_TYPE |  |
| COUNTRY\_ORIG |  | COUNTRY\_ORIG |
| SHIP\_FM\_COUNTRY |  | SHIP\_FM\_COUNTRY |
| SHIP\_TO\_COUNTRY |  | SHIP\_TO\_COUNTRY |
| VESSEL\_NAME |  | VESSEL\_NAME |
| VOYAGE\_NUMBER |  | VOYAGE\_NUMBER |
| DOCUMENT\_AMOUNT | DOCUMENT\_AMOUNT |  |
| LC\_AMOUNT |  | LC\_AMOUNT |
| LC\_NUMBER |  | LC\_NUMBER |
| TRANS\_DATE |  | ISSUE\_DATE |
| VALUE\_DATE | VALUE\_DATE |  |
| DATE\_TIME | DATE\_TIME |  |
| APPLICANT |  | APPLICANT |
| APPLICANT\_CUSTNO |  | APPLICANT\_CUSTNO |
| APPLICANT\_ACC |  | APPLICANT\_ACC |
| BENEFICIARY |  | BENEFICIARY |
| BENEFICIARY\_CUSTNO |  | BENEFICIARY\_CUSTNO |
| BENEFICIARY\_ACC |  | BENEFICIARY\_ACC |
| ACT\_GOODS\_DESC |  | ACT\_GOODS\_DESC |
| GOODSTYPE |  | GOODSTYPE |

1. Create new table #SHIPMENT\_DISCREPANCY\_ALERTS. Insert all the transactions which violate one or more of the alerting conditions: i.e. transactions belonging to multiple customers into this table which have identical values only for four of the five parameters listed below:

Ship To Country, Ship From Country, Origin Country, Vessel Name and Voyage Number.

Also, insert the transactions which have identical values for Trans\_date and Vessel\_Name but different Voyager number belonging to the same customer.

All of the inserted transactions constitute the alerted population.

1. Create four new table for each of the alerting conditions and insert respective alerted transactions into each of the table.
2. To number the pairs of alerts, (e.g. if there are five alerted transactions,

Transaction 1: Transaction date=a, Vessel Name= b, Voyage Number=c Customer=C1

Transaction 2: Transaction date=a, Vessel Name= b, Voyage Number=v Customer=C1

Transaction 3: Transaction date=a, Vessel Name= b, Voyage Number=c Customer=C2

Transaction 4: Transaction date=a, Vessel Name= b, Voyage Number=v Customer=C2

Transaction 5: Transaction date=a, Vessel Name= b, Voyage Number=c Customer=C2

Transactions 1, 2 form 1 pair of alerts as these together breach the alerting condition (identical values for Vessel Name and Transaction Date and different value for Voyage Number). For the same alerting condition, transaction 3,4 and 5 form another pair of alert. The same logic holds true of the rest of the alerting condition) the dense rank function is utilized.

1. The numbered transactions are then inserted into a new table DISCPNCYSHPMNTTRANSACTIONS
2. Update table DISCPNCYSHPMNTTRANSACTIONS by replacing the Goods ID value in the GOODSTYPE field by the Goods Category (i.e. field GOODS\_DISPLAY from #LCGOODSTYPE).
3. In a new table VSSLSHMTDSPYALERTCUSTOMERS, insert all the alerting conditions, and the respective group numbers of the alerted transactions for each FOCALENTITY.
4. Create temporary table #TT3 and insert all the alerted transaction IDs and all the breached alerting conditions by those transactions, since one transaction can breach multiple alerting conditions.

E.g. row in #TT5: Row 1: ID=TF0084393242345, alerting condition=Ship\_FM\_Country, Ship\_To\_Country

1. Create temporary table #TT4 and insert all the customers and all the breached alerting conditions by that customer. #TT5 has multiple rows which may correspond to same focalentity Use this table to insert all the Focalentities and the corresponding focalentites in a single row into table #TT5.

E.g.

**Table #TT4:**

Row 1: FOCALENTITY=1, SHIPMENT\_DISCREPANCY= Ship\_FM\_Country

Row 2: FOCALENTITY=1, SHIPMENT\_DISCREPANCY= Ship\_To\_Country

**Table #TT5:**

Row 1: FOCALENTITY=1, SHIPMENT\_DISCREPANCY= Ship\_FM\_Country, Ship\_To\_Country

1. Create another table SHIPPINGDISCREPANCYGEOMAPDATA and insert all the records corresponding to the alerted customer for the current month from #SHIPMENT\_DISCREPANCY. Use table #TT3 in a join condition to obtain transactions belonging to alerted customers and to get the alerting condition for each alerted transaction
2. Update table SHIPPINGDISCREPANCYGEOMAPDATA by replacing the Goods ID value in the GOODSTYPE field by the Goods Category (i.e. field GOODS\_DISPLAY from #LCGOODSTYPE).
3. Insert the Focalentity/Customer ID, Scenario, Alerting Condition, Customer Name, Alert Month, Run Month into the table ‘CONSOLIDATEDALERTS’ which maintains an archive of all the historical alerts
4. Insert all the transaction ID and other relevant fields into the table ' HISTORICALTRANSACTIONS’ which maintains an archive of all the historical transactions of alerted customers

Note: every time the scenario is run, the tables PBSA.DBO. DISCPNCYSHPMNTTRANSACTIONS, PBSA.DBO.VSSLSHMTDSPYALERTCUSTOMERS and PBSA.DBO, SHIPPINGDISCREPANCYGEOMAPDATA are deleted if present at the beginning by the procedure. These tables are later created again so as to contain only those alerts w.r.t the run period.

## Multiple Invoicing

Scenario Description: The scenario monitors transactions to alert instances where the invoice number is identical but the value of Document Amount/Customer/Goods Type varies

The logical flow of the procedure implemented to monitor this scenario is explained below:

1. Input parameters
   1. @CURRPERIODSTART: current month start day in the format YYYYMMDD
   2. @CURRPERIODEND: current month end day in the format YYYYMMDD

**Example:**

@ CURRPERIODSTART = 20160601

@ CURRPERIODEND = 20160631

If two transactions in the same month belonging to the same customer have identical values for the fields Country of Origin, Ship From Country, Ship To Country, Applicant, Beneficiary, Goods Type, Vessel Name, Voyage Number, Document Amount, the respective customer and transactions are alerted

1. Step 2 to 4 are same as that of Carousel Transactions except for the following difference:
   1. Rather than adding the data into #CAROUSELTRANSACTIONS, the records are inserted into table #MULTIPLEINVOICE

Data Mapping: All the fields retrieved from #MULTI\_LC\_HIST and #MULTI\_DRAWINGS\_HIST into the table #TRADEINNEWGEO for LC type other than collection is as shown below:

|  |  |  |
| --- | --- | --- |
| #MULTIPLEINVOICE | #MULTI\_DRAWINGS\_HIST | #MULTI\_LC\_HIST |
| ID | ID |  |
| FOCALENTITY |  | For import LC types (LC\_TYPE beginning with ‘I’ or ‘SO’), the FOCALENTITY is populated with **APPLICANT\_CUSTNO,** if absent then the field is populated with **APPLICANT**. If both these fields are absent then the field is populated with the **ID**  For export LC types (LC\_TYPES which begin with all other alphabets other than ‘I’ or ‘SO’ ) the FOCALENTITY is populated with **BENEFICIARY\_CUSTNO,** if absent then the field is populated with the **BENEFICIARY**.If both these fields are absent then the field is populated with the **ID** |
| OLD\_LC\_NUMBER |  | OLD\_LC\_NUMBER |
| LC\_TYPE |  | LC\_TYPE |
| DRAWING\_TYPE | DRAWING\_TYPE |  |
| COUNTRY\_ORIG | COUNTRY\_ORIG |  |
| SHIP\_FM\_COUNTRY | SHIP\_FM\_COUNTRY |  |
| SHIP\_TO\_COUNTRY | SHIP\_TO\_COUNTRY |  |
| VESSEL\_NAME | VESSEL\_NAME |  |
| VOYAGE\_NUMBER | VOYAGE\_NUMBER |  |
| DOCUMENT\_AMOUNT | DOCUMENT\_AMOUNT |  |
| LC\_AMOUNT |  | LC\_AMOUNT |
| LC\_NUMBER |  | LC\_NUMBER |
| TRANS\_DATE | DOC\_RECE\_DT |  |
| VALUE\_DATE | VALUE\_DATE |  |
| DATE\_TIME | DATE\_TIME |  |
| APPLICANT |  | APPLICANT |
| APPLICANT\_CUSTNO |  | APPLICANT\_CUSTNO |
| APPLICANT\_ACC |  | APPLICANT\_ACC |
| BENEFICIARY |  | BENEFICIARY |
| BENEFICIARY\_CUSTNO |  | BENEFICIARY\_CUSTNO |
| BENEFICIARY\_ACC |  | BENEFICIARY\_ACC |
| ACT\_GOODS\_DESC | ACT\_GOODS\_DESC |  |
| GOODSTYPE | GOODSTYPE |  |

Data Mapping: All the fields retrieved from #MULTI\_LC\_HIST and #MULTI\_DRAWINGS\_HIST into the table #TRADEINNEWGEO for the collection LC type is as shown below:

|  |  |  |
| --- | --- | --- |
| #CAROUSELTRANSACTIONS | #MULTI\_DRAWINGS\_HIST | #MULTI\_LC\_HIST |
| ID | ID |  |
| FOCALENTITY |  | The FOCALENTITY is populated with **APPLICANT\_CUSTNO,** if absent then the field is populated with **APPLICANT**. If both these fields are absent then the field is populated with the **ID** |
| OLD\_LC\_NUMBER |  | OLD\_LC\_NUMBER |
| LC\_TYPE |  | LC\_TYPE |
| DRAWING\_TYPE | DRAWING\_TYPE |  |
| COUNTRY\_ORIG |  | COUNTRY\_ORIG |
| SHIP\_FM\_COUNTRY |  | SHIP\_FM\_COUNTRY |
| SHIP\_TO\_COUNTRY |  | SHIP\_TO\_COUNTRY |
| VESSEL\_NAME |  | VESSEL\_NAME |
| VOYAGE\_NUMBER |  | VOYAGE\_NUMBER |
| DOCUMENT\_AMOUNT | DOCUMENT\_AMOUNT |  |
| LC\_AMOUNT |  | LC\_AMOUNT |
| LC\_NUMBER |  | LC\_NUMBER |
| TRANS\_DATE |  | ISSUE\_DATE |
| VALUE\_DATE | VALUE\_DATE |  |
| DATE\_TIME | DATE\_TIME |  |
| APPLICANT |  | APPLICANT |
| APPLICANT\_CUSTNO |  | APPLICANT\_CUSTNO |
| APPLICANT\_ACC |  | APPLICANT\_ACC |
| BENEFICIARY |  | BENEFICIARY |
| BENEFICIARY\_CUSTNO |  | BENEFICIARY\_CUSTNO |
| BENEFICIARY\_ACC |  | BENEFICIARY\_ACC |
| ACT\_GOODS\_DESC |  | ACT\_GOODS\_DESC |
| GOODSTYPE |  | GOODSTYPE |

1. Insert all the transactions into table MULTINVALERTCUSTTRANS which have identical values for the fields Country of Origin, Ship From Country, Ship To Country, Applicant, Beneficiary, Goods Type, Vessel Name, Voyage Number, Document Amount. This is accomplished by using an inner join.
2. Update table MULTINVALERTCUSTTRANS by replacing the Goods ID value in the GOODSTYPE field by the Goods Category (i.e. field GOODS\_DISPLAY from #LCGOODSTYPE).
3. Create a table MULTINVALERTCUSTOMERS and insert the FOCALENTITY and total number of transactions which were identified as ‘Multiple Invoice’ in the previous step.
4. Create a table PBSA.DBO. MULTINVOICEMAPDATA and insert all the transactions belonging to all the alerted customers for the entire month (i.e. all transactions between @CURRPERIODSTART and @CURRPERIODEND) into it.
5. Insert ‘MULTIPLE\_INVOICING’ as ALERT\_NAME and ALERTINGCONDITIONS, MONTH, FOCALENTITY, CUSTOMERNAME into the CONSOLIDATEDALERTS tables
6. Insert the Focalentity/Customer ID, Scenario, Alerting Condition, Customer Name, Alert Month, Run Month into the table ‘CONSOLIDATEDALERTS’ which maintains an archive of all the historical alerts
7. Insert all the transaction ID and other relevant fields into the table ' HISTORICALTRANSACTIONS’ which maintains an archive of all the historical transactions of alerted customers

Note: every time the scenario is run, the tables PBSA.DBO. MULTINVALERTCUSTTRANS, PBSA.DBO. MULTINVALERTCUSTOMERS and PBSA.DBO. MULTINVOICEMAPDATA are deleted if present at the beginning by the procedure. These tables are later created again so as to contain only those alerts w.r.t the run period.

## Trades Involving High Risk Jurisdictions

This scenario identifies activity if the country with which a customer traded is a High Risk country

The logical flow of the procedure implemented to monitor this scenario is explained below:

1. Input parameters
   1. @CURRPERIODSTART: start date of current monitoring month (format:yyyymmdd)
   2. @CURRPERIODEND: end date of current monitoring month (format:yyyymmdd)
   3. @HRShipFMCntryThsld: threshold for total number of trades in the current month involving a High Risk Ship From Country for a given customer
   4. @HRShipToCntryThsld: threshold for total number of trades in the current month involving a High Risk Ship To Country for a given customer
   5. @HROrigCntryThsld: threshold for total number of trades in the current month involving a High Risk Country of Origin for a given customer
   6. @HRCustHRGeoThsld: threshold for total number of trades in the current month involving a High Risk Country (Ship From Country/ Ship To Country or Country of Origin) for a given High Risk Customer

**Example:**

@CURRPERIODSTART INT =20160601,

@CURRPERIODEND INT =20160631,

@HRSHIPFMCNTRYTHSLD INT =1,

@HRORIGCNTRYTHSLD INT =1, --

@HRSHIPTOCNTRYTHSLD INT =1,

@HRCUSTHRGEOTHSLD INT =1

The customer trades with a country in the current period (i.e. period between CURRPERIODSTART and @ CURRPERIODEND) and breaches the rest of tunable thresholds, it’ll be alerted

1. Step 2 to 4 are same that of Carousel Transactions except for the following two differences:
   1. Rather than adding the data into #CAROUSELTRANSACTIONS, the records are inserted into table #TRADEINHRGEO.
   2. Transactions are not excluded if the ACT\_GOODS\_DESC field contains ‘ZZ’ values.

Data Mapping: All the fields retrieved from #MULTI\_LC\_HIST and #MULTI\_DRAWINGS\_HIST into the table #TRADEINNEWGEO for LC type other than collection is as shown below:

|  |  |  |
| --- | --- | --- |
| #TRADEINHRGEO | #MULTI\_DRAWINGS\_HIST | #MULTI\_LC\_HIST |
| ID | ID |  |
| FOCALENTITY |  | For import LC types (LC\_TYPE beginning with ‘I’ or ‘SO’), the FOCALENTITY is populated with **APPLICANT\_CUSTNO,** if absent then the field is populated with **APPLICANT**. If both these fields are absent then the field is populated with the **ID**  For export LC types (LC\_TYPES which begin with all other alphabets other than ‘I’ or ‘SO’ ) the FOCALENTITY is populated with **BENEFICIARY\_CUSTNO,** if absent then the field is populated with the **BENEFICIARY**.If both these fields are absent then the field is populated with the **ID** |
| OLD\_LC\_NUMBER |  | OLD\_LC\_NUMBER |
| LC\_TYPE |  | LC\_TYPE |
| HIGHRISKCUSTOMER | Refer step 7 | |
| HIGHRISKSHIPFMCOUNTRY | Refer step 4 | |
| HIGHRISKORIGCOUNTRY | Refer step 6 | |
| HIGHRISKSHIPTOCOUNTRY | Refer step 5 | |
| DRAWING\_TYPE | DRAWING\_TYPE |  |
| COUNTRY\_ORIG | COUNTRY\_ORIG |  |
| SHIP\_FM\_COUNTRY | SHIP\_FM\_COUNTRY |  |
| SHIP\_TO\_COUNTRY | SHIP\_TO\_COUNTRY |  |
| VESSEL\_NAME | VESSEL\_NAME |  |
| VOYAGE\_NUMBER | VOYAGE\_NUMBER |  |
| DOCUMENT\_AMOUNT | DOCUMENT\_AMOUNT |  |
| LC\_AMOUNT |  | LC\_AMOUNT |
| LC\_NUMBER |  | LC\_NUMBER |
| TRANS\_DATE | DOC\_RECE\_DT |  |
| VALUE\_DATE | VALUE\_DATE |  |
| DATE\_TIME | DATE\_TIME |  |
| APPLICANT |  | APPLICANT |
| APPLICANT\_CUSTNO |  | APPLICANT\_CUSTNO |
| APPLICANT\_ACC |  | APPLICANT\_ACC |
| BENEFICIARY |  | BENEFICIARY |
| BENEFICIARY\_CUSTNO |  | BENEFICIARY\_CUSTNO |
| BENEFICIARY\_ACC |  | BENEFICIARY\_ACC |
| ACT\_GOODS\_DESC | ACT\_GOODS\_DESC |  |
| GOODSTYPE | GOODSTYPE |  |

Data Mapping: All the fields retrieved from #MULTI\_LC\_HIST and #MULTI\_DRAWINGS\_HIST into the table #TRADEINHRGEO for the collection LC type is as shown below:

|  |  |  |
| --- | --- | --- |
| #TRADEINHRGEO | #MULTI\_DRAWINGS\_HIST | #MULTI\_LC\_HIST |
| ID | ID |  |
| FOCALENTITY |  | The FOCALENTITY is populated with **APPLICANT\_CUSTNO,** if absent then the field is populated with **APPLICANT**. If both these fields are absent then the field is populated with the **ID** |
| OLD\_LC\_NUMBER |  | OLD\_LC\_NUMBER |
| LC\_TYPE |  | LC\_TYPE |
| HIGHRISKCUSTOMER | Refer step 7 | |
| HIGHRISKSHIPFMCOUNTRY | Refer step 4 | |
| HIGHRISKORIGCOUNTRY | Refer step 6 | |
| HIGHRISKSHIPTOCOUNTRY | Refer step 5 | |
| DRAWING\_TYPE | DRAWING\_TYPE |  |
| COUNTRY\_ORIG |  | COUNTRY\_ORIG |
| SHIP\_FM\_COUNTRY |  | SHIP\_FM\_COUNTRY |
| SHIP\_TO\_COUNTRY |  | SHIP\_TO\_COUNTRY |
| VESSEL\_NAME |  | VESSEL\_NAME |
| VOYAGE\_NUMBER |  | VOYAGE\_NUMBER |
| DOCUMENT\_AMOUNT | DOCUMENT\_AMOUNT |  |
| LC\_AMOUNT |  | LC\_AMOUNT |
| LC\_NUMBER |  | LC\_NUMBER |
| TRANS\_DATE |  | ISSUE\_DATE |
| VALUE\_DATE | VALUE\_DATE |  |
| DATE\_TIME | DATE\_TIME |  |
| APPLICANT |  | APPLICANT |
| APPLICANT\_CUSTNO |  | APPLICANT\_CUSTNO |
| APPLICANT\_ACC |  | APPLICANT\_ACC |
| BENEFICIARY |  | BENEFICIARY |
| BENEFICIARY\_CUSTNO |  | BENEFICIARY\_CUSTNO |
| BENEFICIARY\_ACC |  | BENEFICIARY\_ACC |
| ACT\_GOODS\_DESC |  | ACT\_GOODS\_DESC |
| GOODSTYPE |  | GOODSTYPE |

1. Create new table #CURRTRADEINHRGEO and insert into it the records belonging to the current period (i.e. TRANS\_DATE between @CURRPERIODSTART and @CURRPERIODEND).
2. Compare the SHIP\_FM\_COUNTRY in the #CURRTRADEINHRGEO and the UpdatedCountries table for each FOCALENTITY and if there is no match for High Risk Country and set the flag HIGHRISKSHIPFMCOUNTRY to 1. In case there is a match, set it to 0.
3. Compare the SHIP\_TO\_COUNTRY in the #CURRTRADEINHRGEO and the UpdatedCountries table for each FOCALENTITY and if there is no match for High Risk Country and set the flag HIGHRISKSHIPTOCOUNTRY to 1. In case there is a match, set it to 0.
4. Compare the ORIG\_COUNTRY in the #CURRTRADEINHRGEO and the UpdatedCountries table for each FOCALENTITY and if there is no match for High Risk Country and set the flag HIGHRISKSHIPORIGCOUNTRY to 1. In case there is a match, set it to 0.
5. Compare the Customer in the #CURRTRADEINHRGEO and the RiskClass of the customer from the customer table for each FOCALENTITY and if there is no match for High Risk Customer then set the flag HIGHRISKCUSTOMER to 1. In case there is a match, set it to 0.
6. Create a table #TT1 with the following structure:

FOCALENTITY VARCHAR(50),

HIGHRISKCUSTOMER INT DEFAULT 0,

HIGHRISKSHIPFMCOUNTRY INT DEFAULT 0,

HIGHRISKORIGCOUNTRY INT DEFAULT 0,

HIGHRISKSHIPTOCOUNTRY INT DEFAULT 0)

1. For each FOCALENTITY, get the HIGHRISKCUSTOMER flag (MAX(CAST(HIGHRISKCUSTOMER AS INT))), total number of SHIP\_FM\_COUNTRY (SUM(HIGHRISKSHIPFMCOUNTRY)), COUNTRY\_ORIG (SUM(HIGHRISKORIGCOUNTRY)) and SHIP\_TO\_COUNTRY (SUM(HIGHRISKSHIPTOCOUNTRY)) from the #CURRTRADEINHRGEO table and insert the respective values into the table #TT1. This is accomplished using a group by function.
2. Create a new table #TRADEINHRGEO\_ALERTS which has the same structure as #CURRTRADEINHRGEO except for an addition field ALERTINGCONDTITION. This field will contain a value to indicate the exact conditions for which the FOCALENTITY was alerted.
3. All the transaction which breach the thresholds for high risk ship from country, high risk ship to country, origin country for a regular customer and high risk country (high risk ship from, ship to and origin) for a high risk customer are inserted into the table #TRADEINHRGEO\_ALERTS.
4. Insert distinct FOCALENTITY and ALERTINGCONDITION into a new table #TT2 from #TRADEINHRGEO\_ALERTS.
5. For each FOCALENTITY, get all the ALERTINGCONDITIONs and insert these records into #TT3
6. For each focal entity in #TT3, get all the fields from #TT1 by joining on FOCALENTITY, and insert the records into a new table PBSA.DBO. TRADEINHRGEOALERTCUSTOMERS. Also, cast all the fields as varchar while inserting them into the table.
7. Insert all the fields from #TRADEINHRGEO\_ALERTS into a new table PBSA.DBO. TRADEINHRGEOALERTTRANSACTIONS.
8. Create a table PBSA.DBO. TRADEINHRGEOMAPDATA and insert all the transactions belonging to all the alerted customers for the entire monitoring period (i.e. all transactions between @LOOKBACKSTART and @CURRPERIODEND) into it. Replace all the spaces in the fields which have them, with underscore ‘\_’
9. Insert the Focalentity/Customer ID, Scenario, Alerting Condition, Customer Name, Alert Month, Run Month into the table ‘CONSOLIDATEDALERTS’ which maintains an archive of all the historical alerts
10. Insert all the transaction ID and other relevant fields into the table ' HISTORICALTRANSACTIONS’ which maintains an archive of all the historical transactions of alerted customers

Note: every time the scenario is run, the tables PBSA.DBO. TRADEINHRGEOALERTCUSTOMERS, PBSA.DBO. TRADEINHRGEOALERTTRANSACTIONS and PBSA.DBO.TRADEINHRGEOMAPDATA are deleted, if present, at the beginning by the procedure. These tables are later created again so as to contain only those alerts w.r.t the run period.

## Change in Behavior: Pattern of Documentary Discrepancy

Periodic report generated to identify activity with pattern of discrepancies in the documents for a lookback window

The logical flow of the procedure implemented to monitor this scenario is explained below:

1. Input parameters
   1. @LOOKBACKSTART: start of the lookback period in the format YYYYMMDD
   2. @CURRPERIODSTART: current month start day in the format YYYYMMDD
   3. @CURRPERIODEND: current month end day in the format YYYYMMDD
   4. @TotNoOfDispncyHRGDsThsld: threshold for number of transactions in the monitoring period with discrepancies involving High Risk goods
   5. @TotNoOfDispncyDUGDsThsld: threshold number of transactions in the monitoring period with discrepancies involving Dual Use Goods Type
   6. @TotDispncyThsld: threshold for number of transactions in the monitoring period with discrepancies involving all Goods Types
   7. @PerHRGDsDispncyThsld: threshold for percentage of transactions in the monitoring period with discrepancies involving High Risk Goods Types
   8. @PerDUGDsDispncyThsld: threshold for Percentage of transactions in the monitoring period with discrepancies involving Dual Use goods
   9. @PerDispncyThsld: threshold for percentage of transactions in the monitoring period with discrepancies for all Goods Types

**Example:**

@LOOKBACKSTART INT = 20151201,

@CURRPERIODSTART INT = 20160601,

@CURRPERIODEND INT = 20160631

@PERDISPNCYTHSLD INT = 6,

@PERHRGDSDISPNCYTHSLD INT = 0.03,

@PERDUGDSDISPNCYTHSLD INT = 0.6,

@TOTDISPNCYTHSLD INT = 2,

@TOTNOOFDISPNCYHRGDSTHSLD INT = 2,

@TOTNOOFDISPNCYDUGDSTHSLD INT = 2

If the customer has a transaction in the current period (i.e. in the period between @CURRPERIODSTART, @CURRPERIODEND) and total number of discrepancies for the customer in the period @LOOKBACKSTART to @CURRPERIODEND is greater than the set threshold, then the customer and the respective transactions with discrepancies are alerted.

1. Step 2 to 4 are same that of Carousel Transactions except for the following difference:
   1. Rather than adding the data into #CAROUSELTRANSACTIONS, the records are inserted into table #DOCDISCREPANCY

Data Mapping: All the fields retrieved from #MULTI\_LC\_HIST and #MULTI\_DRAWINGS\_HIST into the table #DOCDISCREPANCY for LC type other than collection is as shown below:

|  |  |  |
| --- | --- | --- |
| #DOCDISCREPANCY | #MULTI\_DRAWINGS\_HIST | #MULTI\_LC\_HIST |
| ID | ID |  |
| FOCALENTITY |  | For import LC types (LC\_TYPE beginning with ‘I’ or ‘SO’), the FOCALENTITY is populated with **APPLICANT\_CUSTNO,** if absent then the field is populated with **APPLICANT**. If both these fields are absent then the field is populated with the **ID**  For export LC types (LC\_TYPES which begin with all other alphabets other than ‘I’ or ‘SO’ ) the FOCALENTITY is populated with **BENEFICIARY\_CUSTNO,** if absent then the field is populated with the **BENEFICIARY**.If both these fields are absent then the field is populated with the **ID** |
| OLD\_LC\_NUMBER |  | OLD\_LC\_NUMBER |
| LC\_TYPE |  | LC\_TYPE |
| DRAWING\_TYPE | DRAWING\_TYPE |  |
| Discrepancy | Discrepancy |  |
| DiscrepancyFlag | Refer Step 5 | |
| hrgoods | Refer Step 6 | |
| dugoods | Refer Step 7 | |
| COUNTRY\_ORIG | COUNTRY\_ORIG |  |
| SHIP\_FM\_COUNTRY | SHIP\_FM\_COUNTRY |  |
| SHIP\_TO\_COUNTRY | SHIP\_TO\_COUNTRY |  |
| VESSEL\_NAME | VESSEL\_NAME |  |
| VOYAGE\_NUMBER | VOYAGE\_NUMBER |  |
| DOCUMENT\_AMOUNT | DOCUMENT\_AMOUNT |  |
| LC\_AMOUNT |  | LC\_AMOUNT |
| LC\_NUMBER |  | LC\_NUMBER |
| TRANS\_DATE | DOC\_RECE\_DT |  |
| VALUE\_DATE | VALUE\_DATE |  |
| DATE\_TIME | DATE\_TIME |  |
| APPLICANT |  | APPLICANT |
| APPLICANT\_CUSTNO |  | APPLICANT\_CUSTNO |
| APPLICANT\_ACC |  | APPLICANT\_ACC |
| BENEFICIARY |  | BENEFICIARY |
| BENEFICIARY\_CUSTNO |  | BENEFICIARY\_CUSTNO |
| BENEFICIARY\_ACC |  | BENEFICIARY\_ACC |
| ACT\_GOODS\_DESC | ACT\_GOODS\_DESC |  |
| GOODSTYPE | GOODSTYPE |  |

Data Mapping: All the fields retrieved from #MULTI\_LC\_HIST and #MULTI\_DRAWINGS\_HIST into the table #TRADEINNEWGEO for the collection LC type is as shown below:

|  |  |  |
| --- | --- | --- |
| #CAROUSELTRANSACTIONS | #MULTI\_DRAWINGS\_HIST | #MULTI\_LC\_HIST |
| ID | ID |  |
| FOCALENTITY |  | The FOCALENTITY is populated with **APPLICANT\_CUSTNO,** if absent then the field is populated with **APPLICANT**. If both these fields are absent then the field is populated with the **ID** |
| OLD\_LC\_NUMBER |  | OLD\_LC\_NUMBER |
| LC\_TYPE |  | LC\_TYPE |
| DRAWING\_TYPE | DRAWING\_TYPE |  |
| Discrepancy | Discrepancy |  |
| DiscrepancyFlag | Refer Step 5 |  |
| HRGoods | Refer Step 6 |  |
| dugoods | Refer Step 7 |  |
| COUNTRY\_ORIG |  | COUNTRY\_ORIG |
| SHIP\_FM\_COUNTRY |  | SHIP\_FM\_COUNTRY |
| SHIP\_TO\_COUNTRY |  | SHIP\_TO\_COUNTRY |
| VESSEL\_NAME |  | VESSEL\_NAME |
| VOYAGE\_NUMBER |  | VOYAGE\_NUMBER |
| DOCUMENT\_AMOUNT | DOCUMENT\_AMOUNT |  |
| LC\_AMOUNT |  | LC\_AMOUNT |
| LC\_NUMBER |  | LC\_NUMBER |
| TRANS\_DATE |  | ISSUE\_DATE |
| VALUE\_DATE | VALUE\_DATE |  |
| DATE\_TIME | DATE\_TIME |  |
| APPLICANT |  | APPLICANT |
| APPLICANT\_CUSTNO |  | APPLICANT\_CUSTNO |
| APPLICANT\_ACC |  | APPLICANT\_ACC |
| BENEFICIARY |  | BENEFICIARY |
| BENEFICIARY\_CUSTNO |  | BENEFICIARY\_CUSTNO |
| BENEFICIARY\_ACC |  | BENEFICIARY\_ACC |
| ACT\_GOODS\_DESC |  | ACT\_GOODS\_DESC |
| GOODSTYPE |  | GOODSTYPE |

1. Delete entities with no transactions in current month
2. Flag transactions which have a discrepancy. DISCREPANCYFLAG is set to 1 if the discrepancy is not any of 'NO','NONE','N','ZZ','IN ORDER' else set discrepancy flag to 0
3. Set HRGOODS to 1 if H\_Risk column from #LCGOODSTYPE is 1 for that particular goods type and discrepancy flag is 1
4. Similarly set DUGOODS to 1 if DUAL\_USE column from #LCGOODSTYPE is 1 for that particular goods type and discrepancy flag is 1
5. Create a table #TT1 with the following structure:

FOCALENTITY VARCHAR(50),

TOTDISCREPANCY NUMERIC(5),

TOTTRANSACTIONS NUMERIC(5),

PER\_DISPNCY NUMERIC(5),

TOT\_HRGOODS\_DISPNCY NUMERIC (5),

PER\_HRGOODS\_DISPNCY NUMERIC (5),

TOT\_DUGOODS\_DISPANCY NUMERIC (5),

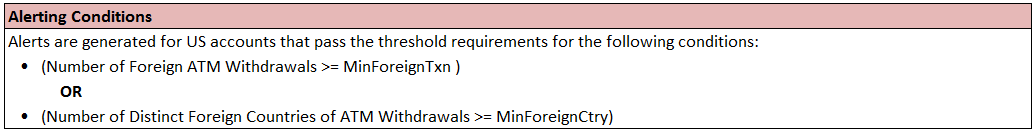
PER\_DUGOODS\_DISPANCY NUMERIC (5))

1. Using table #DOCDISCREPANCY, for each FOCALENTITY, get the number of transactions with discrepancies (SUM(DISCREPANCYFLAG)), total number of focal entities(COUNT(FOCALENTITY)), total number of high risk goods (SUM(HRGOODS) )and dual use goods (SUM(DUGOODS) ) using a group by function and insert the data into #TT1 table.
2. Calculate percentage of discrepancies for total number of discrepancies, High Risk Goods and Dual Use Goods.
3. Create a new table #DOCDISCREPANCYALERTS.
4. All the transactions which breach the thresholds for new goods type , new high risk goods type, new dual use goods type and total new goods type for a high risk customer are inserted into the table #DOCDISCREPANCYALERTS. This is done by joining the # TT1 table with #DOCDISCREPANCY on FOCALENTITY.
5. Insert distinct FOCALENTITY and ALERTINGCONDTITION into a new table #TT2 from #DOCDISCREPANCYALERTS..
6. For each FOCALENTITY, get all the ALERTINGCONDTITIONs and insert these records into #TT3
7. For each focal entity in #TT3, get all the fields from #TT1 by joining on FOCALENTITY, and insert the records into a new table PBSA.DBO.DOCDSPYALERTCUSTOMERS.
8. Insert all the fields from #DOCDISCREPANCYALERTS into a new table PBSA.DBO.DOCDSPYALERTTRANSACTION. Also, cast all the fields as varchar while inserting them into the table.
9. Create a table PBSA.DBO.DOCDISCREPANCYMAPDATA and insert all the transactions belonging to all the alerted customers for the entire monitoring period (i.e. all transactions between @LOOKBACKSTART and @CURRPERIODEND) into it.
10. Insert the Focalentity/Customer ID, Scenario, Alerting Condition, Customer Name, Alert Month, Run Month into the table ‘CONSOLIDATEDALERTS’ which maintains an archive of all the historical alerts
11. Insert all the transaction ID and other relevant fields into the table ' HISTORICALTRANSACTIONS’ which maintains an archive of all the historical transactions of alerted customers

Note: every time the scenario is run, the tables PBSA.DBO.DOCDSPYALERTCUSTOMERS and PBSA.DBO.DOCDSPYALERTTRANSACTION are deleted if present at the beginning by the procedure. These tables are later created again so as to contain only those alerts w.r.t the run period.

## Foreign Debit Card Transactions

Scenario Description: This scenario generates alerts for US accounts with a change in foreign debit card activity. The following alerting conditions must be satisfied for an alert to be generated during a given monitoring month. Transactions from China are excluded from this scenario.



The logical flow of the procedure implemented to monitor this scenario is explained below. In the stored procedure, the following parameters and tables are created to retrieve data and perform calculations in order to identify accounts that should be alerted.

1. **Input Parameters**
   1. @TxnDateStart: Start date of the current monitoring month
   2. @TxnDateEnd: End date of the current monitoring month
   3. @MinForeignTxn: Minimum Foreign Transaction Count threshold, which is 1
   4. @MinForeignCtry: Minimum Foreign Country Count threshold, which is 1

Example of Parameter Values for Monitoring Month of July 2016:

@TxnDateStart DATETIME = '2016-07-01',

@TxnDateEnd DATETIME = '2016-07-31',

@MinForeignTxn INT = 1.0,

@MinForeignCtry INT = 1.0

The values for @TxnDateStart and @TxnDateEnd need to be updated during stored procedure execution or changed within the procedure itself to reflect the updated dates when the scenario is run each month.

1. **#ForeignDebit\_Tran Table**: This temporary table is used to store all the relevant transactions for this scenario by pulling the following data fields. These data fields are retrieved from the Vantiv data imported into the Debit\_Transactions table, as well as the AccountOwner and Customer tables in Prime. These are the relevant data fields for the scenario that calculations in other tables are based on.

|  |  |
| --- | --- |
| #ExcessiveWithdrawals\_Tran | Data Field Source |
| ID | Created as a unique ID identifier |
| Transaction Date | Debit\_Transactions |
| Terminal Country | Debit\_Transactions |
| Transaction Type | Debit\_Transactions |
| Transaction Amount | Debit\_Transactions |
| Transaction Status | Debit\_Transactions |
| From Account Number | Debit\_Transactions; ‘0’ is added before each account number for standardization with Prime |
| Customer ID | AccountOwner |
| Customer Name | Customer |
| Relationship | AccountOwner |
| Customer Residence | Customer |

This table is used to store all authorized and completed ATM cash withdrawal transactions for US accounts for the current alert month. In order to select this data, the following filters are applied to retrieve the relevant data for this scenario:

* 1. Transaction Date is in current monitoring month (between @TxnDateStart and @TxnDateEnd)
  2. Transactio Type = Cash Withdrawal
  3. Transaction Status = Authorized and Completed
  4. Customer’s Country of Residence = US

1. **#ForeignDebit\_ForeignTran Table**: This temporary table is used to store all same transactions from #ForeignDebit\_Tran, but also includes a Foreign Country flag that takes on the value of 1 if the transaction occurs in any country other than the United States and China, and the value of 0 otherwise.
2. **#ForeignDebit\_Acct Table**: This temporary table is used to calculate the Foreign Transaction Count and Foreign Country Count for each account during the alert month. It uses the transaction data in #ForeignDebit\_ForeignTran to calculate the number of foreign transactions and distinct foreign countries of withdrawals for each account.
3. **#ForeignDebit\_Acct2 Table**: This table is used to calculate Alert Flag, as well as display account data and calculations from previous tables for each account. Alert Flag has a value of 1 when the account satisfies either of the two alerting conditions, and has a value of 0 when both alerting conditions are not satisfied. This table uses data and calculations from #ForeignDebit\_Acct and #ForeignDebit\_Tran.
4. **ForeignDebit\_Alert Table**: This table is used to display data and calculations for alerted accounts. It has the same data fields as #ForeignDebit\_Acct2, but only includes information for alerted accounts.
5. **ForeignDebit\_AlertedTxn Table**: This table is used to store all transactions that triggered alerts for alerted accounts during the current alert month. It uses data and calculations from the #ForeignDebit\_ForeignTran and ForeignDebit\_Alert tables to determine if a transaction satisfies any of the alerting conditions for an alerted account.
6. **ForeignDebit\_Alert1YearTxn Table**: This table is used to store all of the transactions for alerted accounts during the current alert month, as well as the past year. This table uses a similar query to the one for the #ForeignDebit\_Tran table to retrieve all transactions during 1 year for alerted accounts.
7. **DebitCard\_AlertArchive Table**: This table is used as an archive to store data for alerted accounts for both Debit Card Scenarios (Pattern of Excessive Withdrawals and Foreign Debit Card) each time the scenarios are run. This table uses the ForeignDebit\_Alert table to retrieve alert data.
8. **DebitCard\_TranArchive Table**: This table is used as an archive to store transactions for alerted accounts for both Debit Card Scenarios (Pattern of Excessive Withdrawals and Foreign Debit Card) each time the scenarios are run. This table uses the ForeignDebit\_Alert1YearTxn table to retrieve relevant transaction data for each alerted account.

1. All the temporary tables are automatically deleted once the procedures execute [↑](#footnote-ref-2)