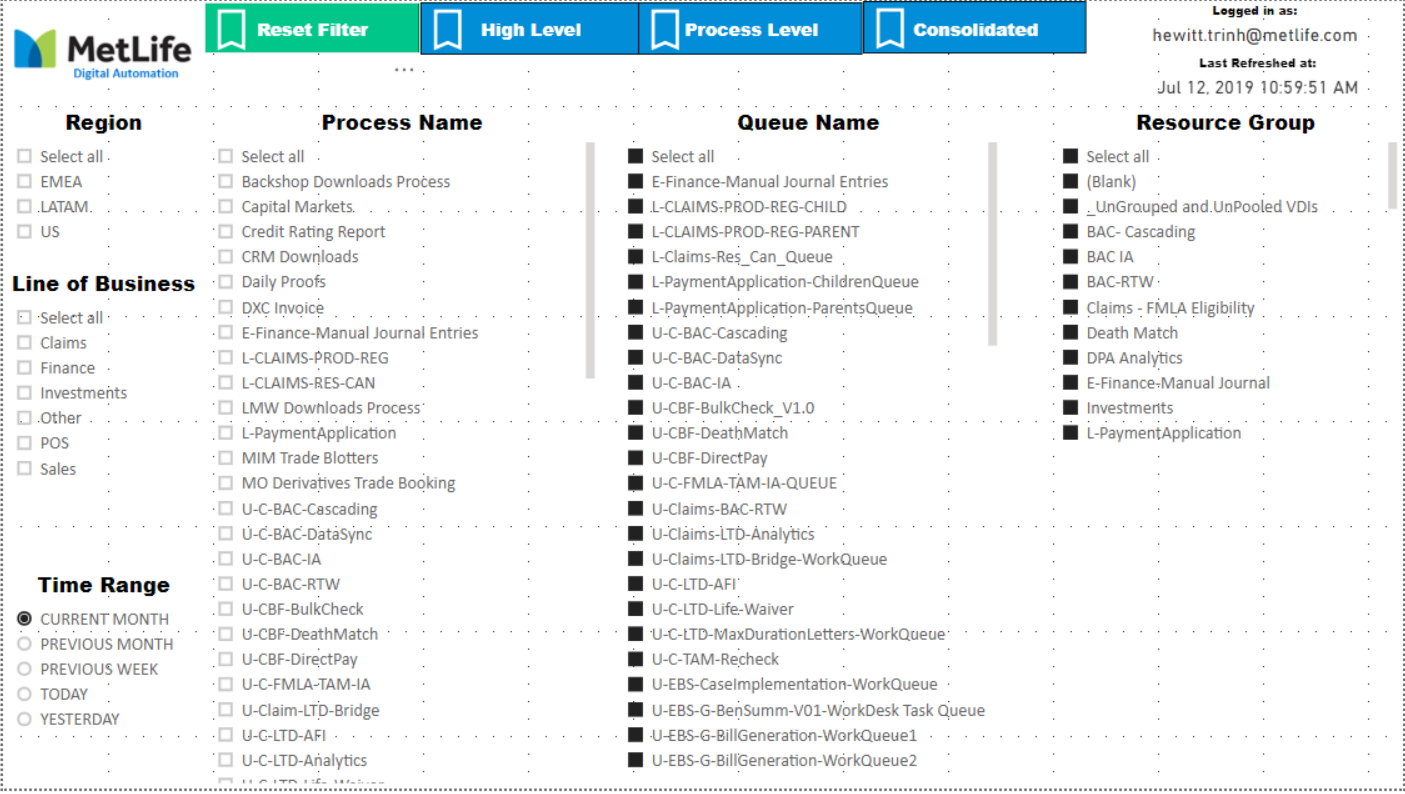
Power BI Phase 2: Digital Automation Platform Reporting Functional & Technical Document

# Filter Selection Page



## Overview

The Filter Selection Page exists for 3 reasons. It provides users ability to select different filters to apply on their reports. It allows Power BI authors to control filters based on users view permission. Lastly, it gives users a snapshot when was the last time the data was refreshed. Overall, users use this page to adjust the default filters they view from Consolidated Page

## Data

### Custom Mapping

Line of Business, Process Name, Queue Name, Resource Group data are provided in the Custom Mapping excel sheet. The Custom Mapping provides relationships to link these slicers to the database date.

Power BI is expected to load this sheet from designated location (shared drive accessed by people in the following AD group APP\_10888\_PBI\_BLUEPRISMLOB\_EXTERNALDATA\_CH). A version of the mapping is provided in the appendix.

### Time Range

Time Range is static value coded into Power BI configuration. It gives the user to view the metrics in a specific time range. The specific dates are calculated dynamically depending on the current time the user view the reports.

Time Range values are defined as follows:

* Yesterday: The preceding day of today (Sunday if today is Monday, Thursday if today is Friday)
* Today: Today
* Current Month: From the beginning of the current month to today (if today is July 1, only data from July 1 is shown, if today is June 30, all data from June 1 to June 30 is shown)
* Previous Month: From the beginning of the last month to end of month, last month is defined as of today (if today is July 1, data from June 1 to June 30 is returned. if today is June 30, data from May 1 to May 30 is returned)
* Previous Week: From beginning of last week to end of last week, last week is defined as of today (if today is Friday, data from Monday to Friday of the previous week is returned)

## Visualization

Each slicers in the snapshot are dictated by these important relationships:

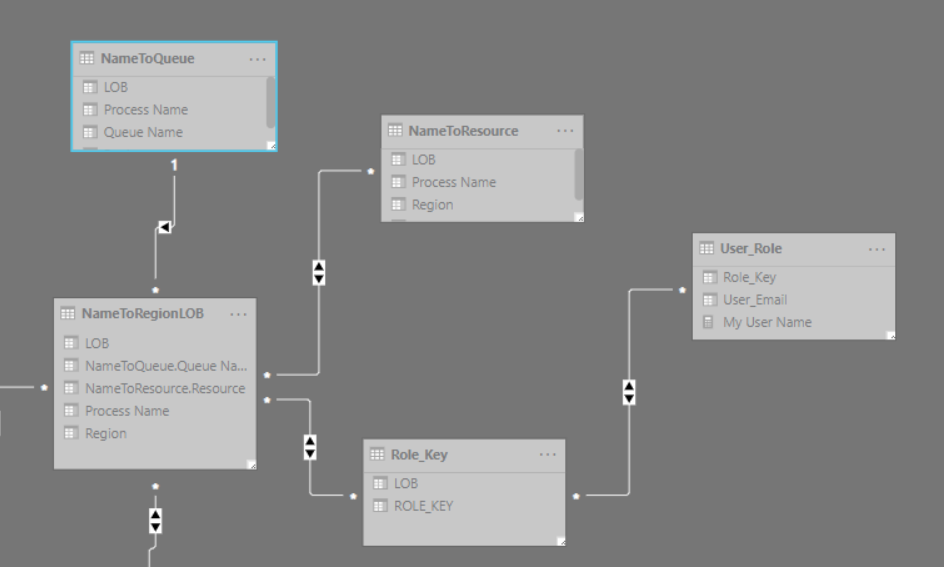
**Line of Business**: Line of Business values are determined by row-level security, ultimately determined by users group

**Line of Business – Process Name**: There is one to many relationship provided in the ProcessNameToLOB tab in the custom mapping sheet.. Multiple selections are allowed. With each selection in Line of Business, linked process name should appear as slicer option.

**Process Name – Queue Name**: There is one to one/many relationship provided in the ProcessNameToQueue tab in the custom mapping sheet. With each selection in process name, linked queue name should be selected automatically.

**Process Name – Resource Group**: There is one to one/many relationship provided in the ProcessNameToResource tab in the custom mapping sheet. With each selection in process name, linked resource group should be selected automatically.

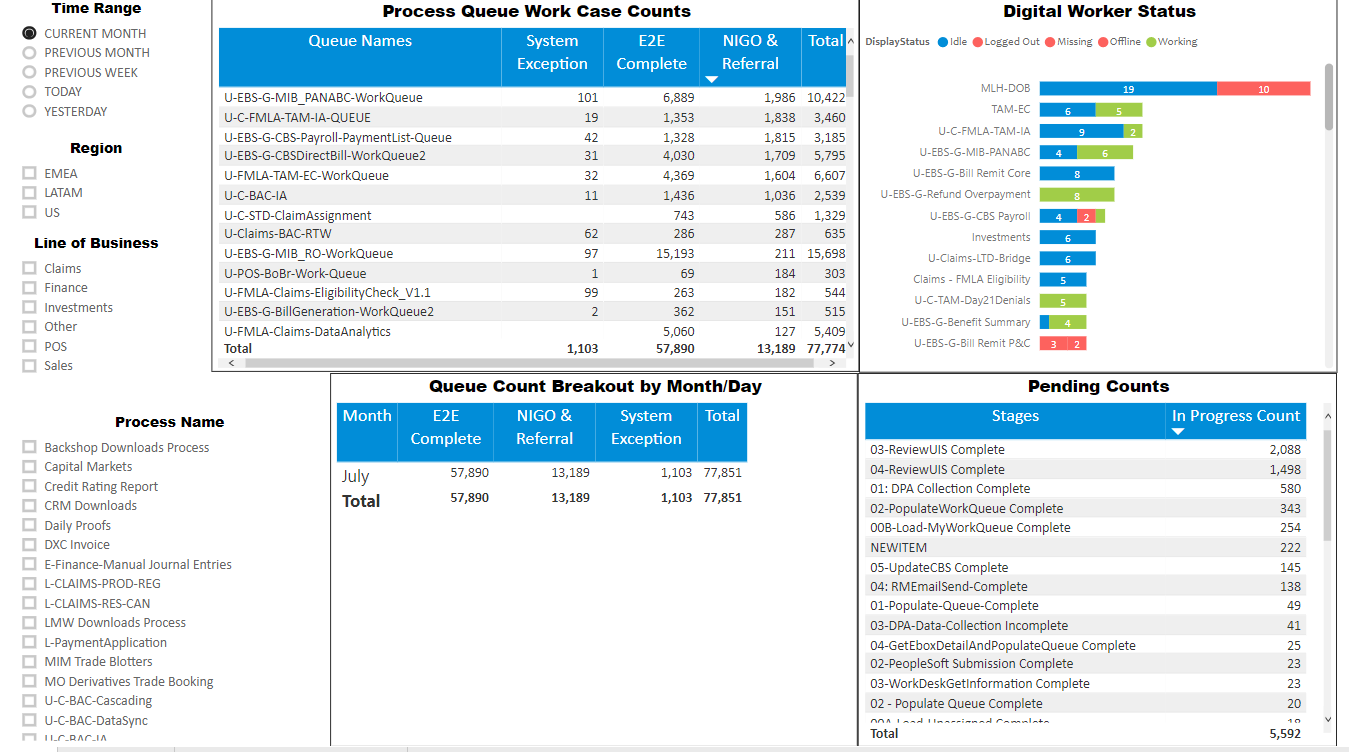
Queue Name and Resource Group are all linked of Process Name. By default, by selecting a process name, users choose to select ALL associated queues and resource group to that process. User can deselect them if they opt not to.



## User Mapping

One thing to note is that Row Level Security is also controlled by the relationship between User Mapping and NameToRegionLOB table.The User List & User Mapping is stored at designated shared drive (accessed by people in the following AD group APP\_10888\_PBI\_BLUEPRISMLOB\_EXTERNALDATA\_CH ). Further details on setting up and maintain user list is in Appendix.

# Consolidated Page



## Function

Consolidated page is one-stop shop for high-level performance of processes in production. On this page, four embedded reports are Work Case Count, Digital Worker Status, Pending Counts and Queue Count Breakout by Month/Day. Work Case Count shows the count of statuses (System Exception, E2E Complete, NIGO&Referral) for work cases worked on the selected time range. Digital Worker Status brings back the status (Idle, Logged Out, Missing, Working) of digital workers belong to the selected process. Pending Counts gives the count of all stages of pending items for selected processes. Queue Count Breakout by Month/Day gives another level of dicing to see the count group by day or month.

## Data

All the four reports shared the same underlying dataset, which is BPVWorkQueueItem table. Following are the report and the corresponding data points as well as calculations

**Process Queue Work Case Counts:**

Data Points: Work Queue Name, state, exceptionreason, tags, lastupdated

Calculations:

System Exception: State = 5 AND No strings like (be:, nigo, referral) in exceptionreason or tags

E2E Complete: State = 4 AND No strings like (be:, nigo, referral) in exceptionreason or tags

NIGO & Referral: Total – Deferred – Locked – Pending – System Exception & E2E Complete

Pending Counts: State = 1 regardless of time range

Totals: E2E Complete + NIGO & Referral + System Exception + Pending Counts

**Digital Worker Status:**

Data Points: Group Resouce Name, State (VDI table)

**Pending Counts:**

Data Points: Status, State (Work Queue Table)

**Queue Count Breakout by Month/Day**

Data Points: lastupdated

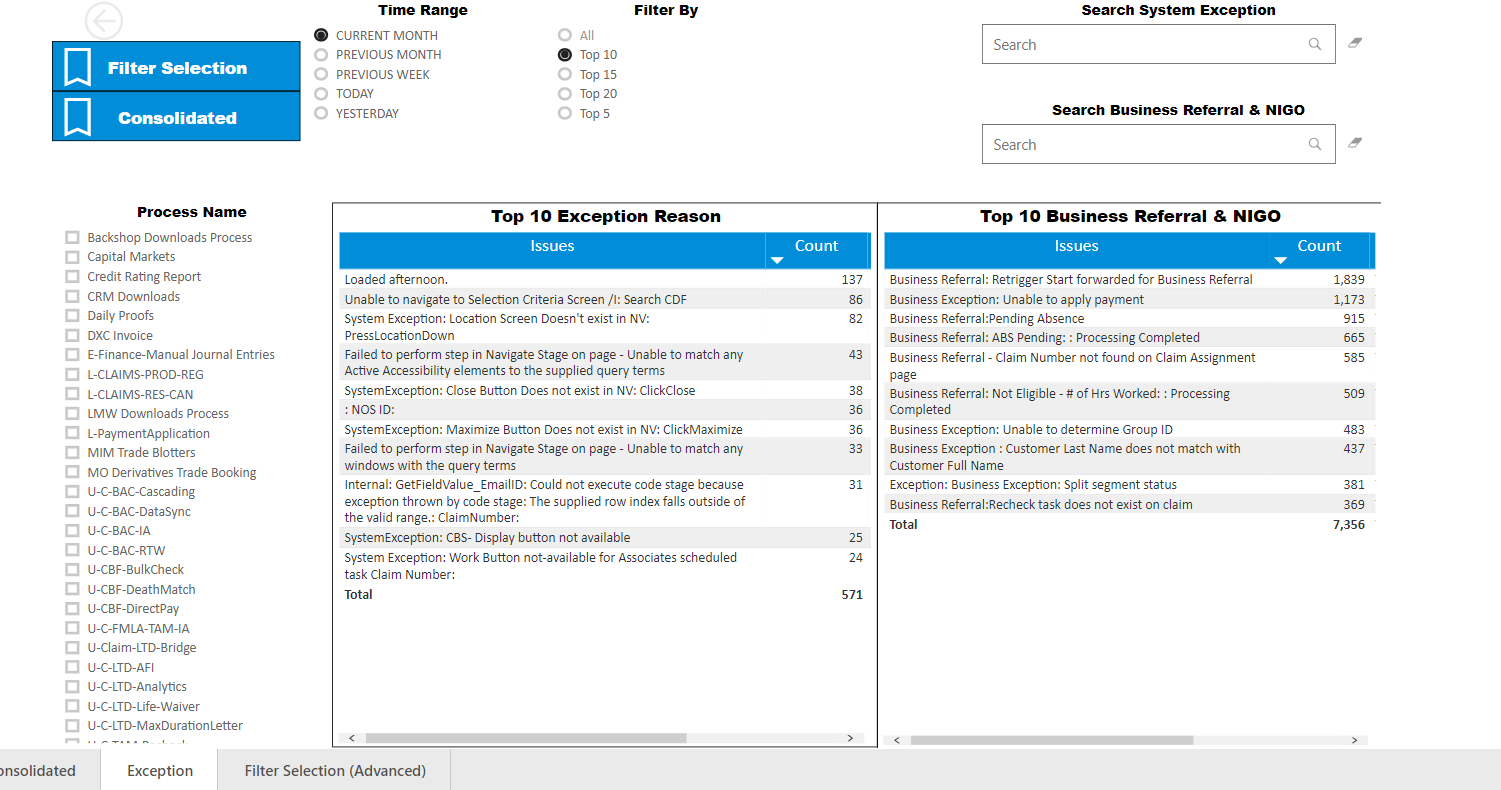
## Visualization

All four reports are filtered by the slicers – Time Range. Region, Line Of Business, Process Name in some degree.

Process Queue Work Case Count and Queue Count Breakout by Month/Day are filtered by the selected time range, therefore, dependent on the current time of the users and the last time the data was refreshed.

Digital Worker Status and Pending Counts are time independent, therefore, only filtered by the selected process name.

# Exception Page



## Function

The Process Exception Page provides summarized count of each system exception and business exception reason over a period of time. They also rank the issues by the count and gives user options to filter by the ranking of the counts.

## Data

Exception Reasons:

Data Points: exception reason

Calculation: all exception reason goes through sanitization process executed by the following algorithm:

* Remove all numbers, words that have “@”
* Remove all details between (), ‘’

NIGO & Referral: tags

Calculation: all tags go through sanitization process executed by the following algorithm:

* Remove all numbers, words that have “@”
* Remove all details between (), ‘’

## Visualization

In addition to the count of each issues in the report, each issue will have a ranking measure which indicates which top N it belongs to. It also a driving measure for Filter By selection on top of the dashboard.

Currently in this version, users can opt to view only top 5, 10, 15, 20 or All the issues based on the count.

Also, users can search for issues in the search box on top corner if the issues count is too small to show in the top N ranking.

# Updating & Maintenance

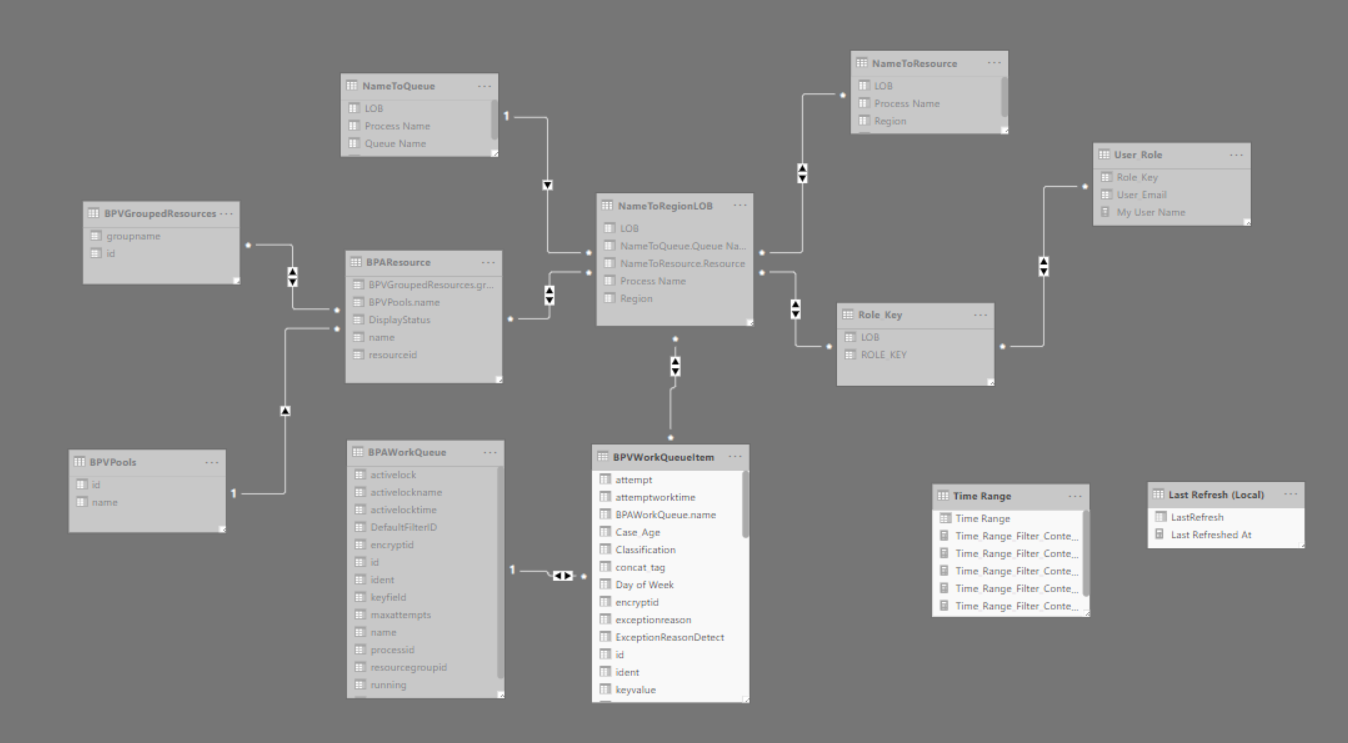
## Onboarding New Users

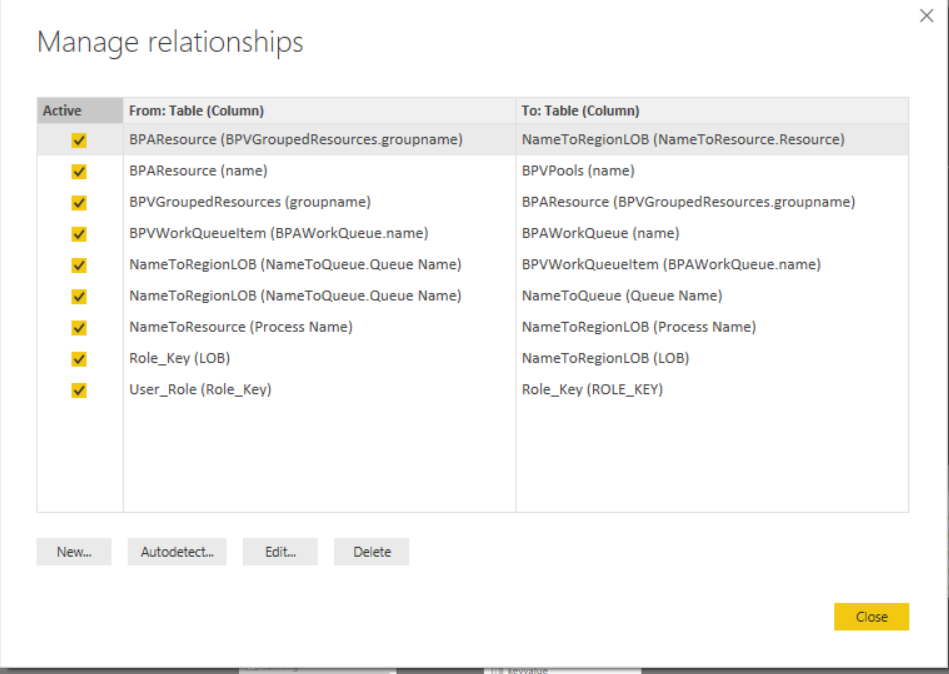
## User Mapping

## Custom Mapping

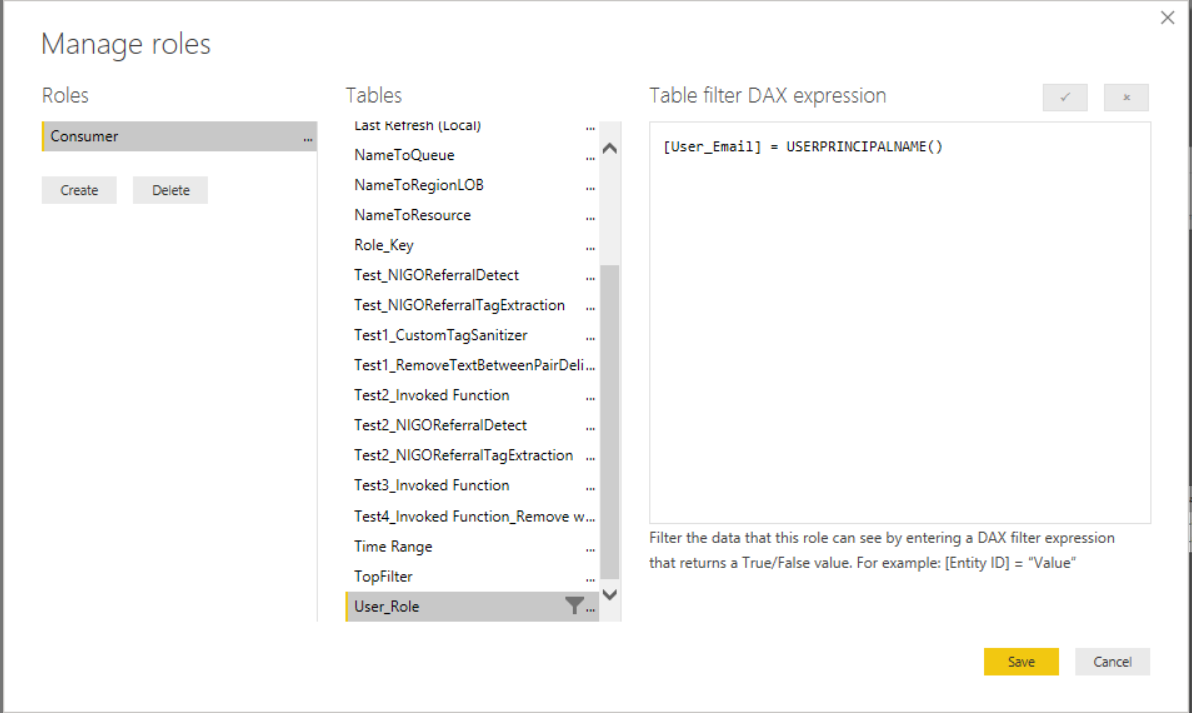
# Technical Appendix

## Data Model

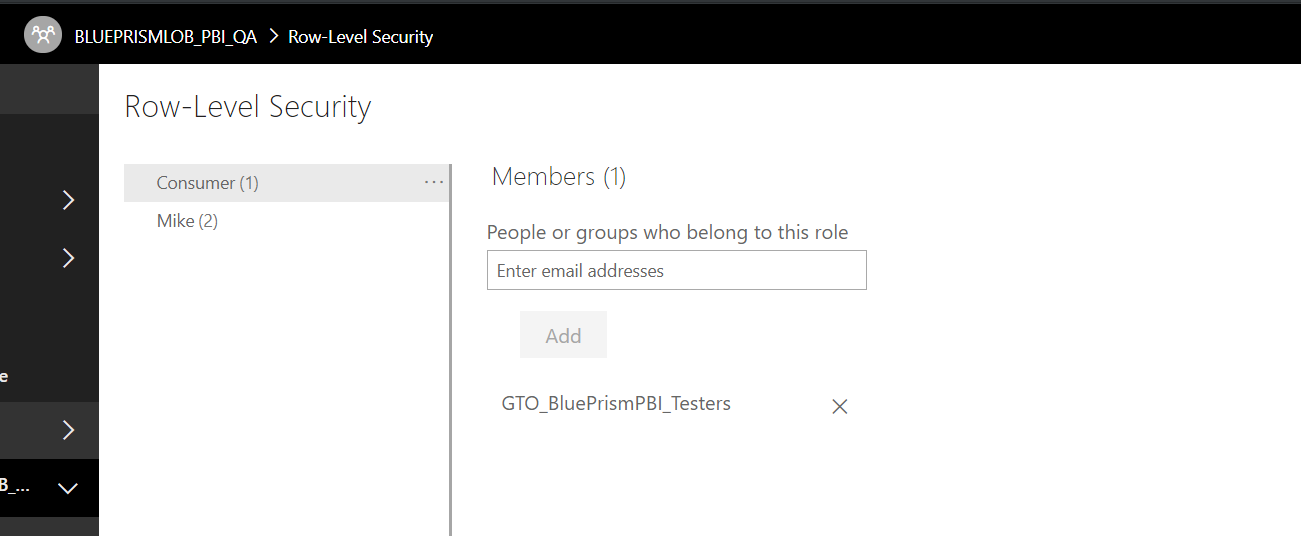




## Role Level Security



Activate Roles in WorkSpace



## Custom Mapping



## Underlying DAX Queries

DAX Query on BPV WorkQueueItem

let

Source = Sql.Database("aglisp10888.metnet.net,1569", "BluePrism\_Prod", [Query="SELECT \* FROM (

SELECT [ident]

,[id]

,[queueid]

,[keyvalue]

,[status]

,[attempt]

,[exceptionreason]

,[worktime]

,[prevworktime]

,[attemptworktime]

,[state]

,[loaded]

,[lastupdated]

,[encryptid]

FROM [BluePrism\_Prod].[dbo].[BPVWorkQueueItem] WITH (NOLOCK)

WHERE [lastupdated] in (Select Max(lastupdated) from [BPVWorkQueueItem] WITH (NOLOCK) Group by [keyvalue]) and

[lastupdated] > = DATEADD(day, -61, GETDATE())) quitem

LEFT JOIN

(Select [queueitemident]

,concat\_tag = STUFF((

Select N';' + [tag] from [BPViewWorkQueueItemTag] WITH (NOLOCK)

where [queueitemident] = itag.queueitemident order by [queueitemident] desc

for XML PATH(''), Type).value(N'.[1]', N'nvarchar(max)'), 1, 1, N'')

FROM [BluePrism\_Prod].[dbo].[BPViewWorkQueueItemTag] as itag WITH (NOLOCK)

Group by [queueitemident]) qutg

on

quitem.ident = qutg.queueitemident

Order by [lastupdated] desc", CommandTimeout=#duration(0, 0, 1, 0)]),

#"Merge Queue Name" = Table.NestedJoin(Source, {"queueid"}, BPAWorkQueue, {"id"}, "BPAWorkQueue", JoinKind.LeftOuter),

#"Expanded BPAWorkQueue1" = Table.ExpandTableColumn(#"Merge Queue Name", "BPAWorkQueue", {"name"}, {"BPAWorkQueue.name"}),

#"Inserted Day of Week" = Table.AddColumn(#"Expanded BPAWorkQueue1", "Day of Week", each Date.DayOfWeek([lastupdated]), Int64.Type),

#"Inserted Week of Year" = Table.AddColumn(#"Inserted Day of Week", "Week of Year", each Date.WeekOfYear([lastupdated]), Int64.Type),

#"Inserted Month" = Table.AddColumn(#"Inserted Week of Year", "Month Of Year", each Date.Month([lastupdated]), Int64.Type),

#"Invoked Custom Function" = Table.AddColumn(#"Inserted Month", "ExceptionReasonDetect", each NIGOReferralDetect(Text.Lower([exceptionreason]), "be:,nigo,business,referral", ",")),

#"Replaced Errors" = Table.ReplaceErrorValues(#"Invoked Custom Function", {{"ExceptionReasonDetect", false}}),

#"Invoked Custom Function1" = Table.AddColumn(#"Replaced Errors", "TagDetect", each NIGOReferralDetect(Text.Lower([concat\_tag]), "be:,nigo,business,referral", ",")),

#"Replaced Errors1" = Table.ReplaceErrorValues(#"Invoked Custom Function1", {{"TagDetect", false}}),

#"Invoked Custom Function2" = Table.AddColumn(#"Replaced Errors1", "sanitized\_tags", each NIGOReferralTagDetect([concat\_tag], ";", "be:,nigo,business,referral", ",")),

#"Invoked Custom Function3" = Table.AddColumn(#"Invoked Custom Function2", "sanitized\_tags\_2", each CustomTagSanitizer([sanitized\_tags])),

#"Invoked Custom Function4" = Table.AddColumn(#"Invoked Custom Function3", "Sanitized\_Exception\_Reason", each CustomTagSanitizer([exceptionreason])),

#"Replaced Errors2" = Table.ReplaceErrorValues(#"Invoked Custom Function4", {{"Sanitized\_Exception\_Reason", null}})

in

#"Replaced Errors2"

DAX Query on NameToRegionLOB

let

Source = Excel.Workbook(Web.Contents("https://mydrive.metlife.com/personal/bfclarke\_metlife\_com/Documents/Digital%20Automation/09%20MI%20Reporting/Dashboards/Power%20BI%20MI%20Reporting/Custom%20Schema%20Mapping.xlsx"), null, true),

NameToRegionLOB\_Sheet = Source{[Item="NameToRegionLOB",Kind="Sheet"]}[Data],

#"Changed Type" = Table.TransformColumnTypes(NameToRegionLOB\_Sheet,{{"Column1", type text}, {"Column2", type text}, {"Column3", type text}}),

#"Promoted Headers" = Table.PromoteHeaders(#"Changed Type", [PromoteAllScalars=true]),

#"Changed Type1" = Table.TransformColumnTypes(#"Promoted Headers",{{"Process Name", type text}, {"Region", type text}, {"LOB", type text}}),

#"Merged Queries" = Table.FuzzyNestedJoin(#"Changed Type1", {"Process Name"}, NameToQueue, {"Process Name"}, "NameToQueue", JoinKind.LeftOuter, [IgnoreCase=true, IgnoreSpace=true]),

#"Expanded NameToQueue" = Table.ExpandTableColumn(#"Merged Queries", "NameToQueue", {"Queue Name"}, {"NameToQueue.Queue Name"}),

#"Merged Queries1" = Table.NestedJoin(#"Expanded NameToQueue", {"Process Name"}, NameToResource, {"Process Name"}, "NameToResource", JoinKind.LeftOuter),

#"Expanded NameToResource" = Table.ExpandTableColumn(#"Merged Queries1", "NameToResource", {"Resource"}, {"NameToResource.Resource"}),

#"Reordered Columns" = Table.ReorderColumns(#"Expanded NameToResource",{"Region", "LOB", "Process Name", "NameToQueue.Queue Name", "NameToResource.Resource"})

in

#"Reordered Columns"

## Custom Function in Power BI

NIGOReferralDetect

let func =

(BigString as text, SearchText as text, Delimiter as text) as logical =>

let

TextToList = List.Buffer(Text.Split(SearchText, Delimiter)),

FilterList = List.Select(TextToList, each \_ <> ""),

Result = List.AnyTrue(List.Transform(FilterList, (substring) => Text.Contains(BigString, substring, Comparer.OrdinalIgnoreCase)))

in

Result

, documentation = [

Documentation.Name = " NIGOReferralDetect

", Documentation.Description = " Returns True or False if any of strings or substring delimited by delimitor appears in the input text

" , Documentation.LongDescription = "

", Documentation.Category = "Text Detection

", Documentation.Source = "

", Documentation.Author = " Hewitt Trinh

", Documentation.Examples = {[Description = " Detect occurance of strings in a string

" , Code = " NIGOReferralDetect(""Business Exception: Claims number is invalid"",""business;invalid"","";"")

", Result = " ""True""

"]}]

in

Value.ReplaceType(func, Value.ReplaceMetadata(Value.Type(func), documentation))

NIGOReferralTagDetect

let func =

(BigString as text, BigStringDelimiter as text, SearchText as text, Delimiter as text) as text =>

let

BigStringToList = List.Buffer(Text.Split(BigString, BigStringDelimiter)),

BigStringFilterList = List.Select(BigStringToList, each \_ <> ""),

Result = List.Accumulate(BigStringFilterList, "", (state, current) =>

if NIGOReferralDetect(current, SearchText, Delimiter)

then state&current else state)

in Result

, documentation = [

Documentation.Name = " NIGOReferralTagDetect

", Documentation.Description = " Returns the tag if any of strings or substring delimited by delimitor appears in the tag after split by a delimiter

" , Documentation.LongDescription = "

", Documentation.Category = "Text Detection

", Documentation.Source = "

", Documentation.Author = " Hewitt Trinh

", Documentation.Examples = {[Description = " Detect occurance of strings in a bigger string, return the bigger string that contains the keywords

" , Code = " NIGOReferralTagDetect(""Business Exception: Claims number is invalid;System Exception"","";"", ""business;invalid"","";"")

", Result = " ""Business Exception: Claims number is invalid""

"]}]

in

Value.ReplaceType(func, Value.ReplaceMetadata(Value.Type(func), documentation))

RemoveTextBetweenPairDelimiters

let func =

(BigString as text, BeginningDelimiter as text, EndingDelimiter as text) as text =>

let

BeginningString = Text.BeforeDelimiter(BigString, BeginningDelimiter, if List.Count(Text.Split(BigString, BeginningDelimiter)) > 2 then 0 else 3 ),

EndingString = Text.AfterDelimiter(BigString, EndingDelimiter, if BeginningDelimiter = EndingDelimiter then 1 else if Text.Length(BeginningString) > 0 then 0 else 10),

Result = Text.Trim(BeginningString) & " " & Text.Trim(EndingString)

in Result

, documentation = [

Documentation.Name = " RemoveTextBetweenPairDelimiters

", Documentation.Description = " Assume any input delimiter has an identical closing delimiter, the function seeks to remove the content between those 2 delimiters while keep the delimiters. Ending delimiter is defined as the one next the first occurence of beginning delimiter

" , Documentation.LongDescription = "

", Documentation.Category = "Text Detection

", Documentation.Source = "

", Documentation.Author = " Hewitt Trinh

", Documentation.Examples = {[Description = "

" , Code = " RemoveTextBetweenPairDelimiters(""Claim Number (ABC123) is invalid"",""("","")

", Result = " ""Claims number () is invalid""

"]}]

in

Value.ReplaceType(func, Value.ReplaceMetadata(Value.Type(func), documentation))

CustomTagSanitizer

let func =

(BigString as text) as text =>

let

Result0\_0 = RemoveTextBetweenPairDelimiters(BigString, "(", ")"),

//Temporary hack to remove another pair

Result0\_1 = RemoveTextBetweenPairDelimiters(Result0\_0, "(", ")"),

Result1\_0 = RemoveTextBetweenPairDelimiters(Result0\_1, "'", "'"),

//Temporary hack to remove another pair

Result1\_1 = RemoveTextBetweenPairDelimiters(Result1\_0, "'", "'"),

WordList = Text.Split(Result1\_1, " "),

WordListSelect = List.Select(WordList, each \_ <> ""),

Result = List.Accumulate(WordListSelect, "",

(state,current) => if Text.Length(Text.Remove(current, {"0".."9", "@"})) < Text.Length(current)

then state & ""

else state & " " & current )

in Result

, documentation = [

Documentation.Name = " NIGO Referral Sanitize Tags

", Documentation.Description = " Customized function to remove text between ""(),'',:"", remove special text like numbers, slashes

" , Documentation.LongDescription = "

", Documentation.Category = "Text Detection

", Documentation.Source = "

", Documentation.Author = " Hewitt Trinh

", Documentation.Examples = {[Description = "

" , Code = " RemoveTextBetweenPairDelimiters(""Claim Number (ABC123) is invalid"",""("","")

", Result = " ""Claims number () is invalid""

"]}]

in

Value.ReplaceType(func, Value.ReplaceMetadata(Value.Type(func), documentation))

## Calculated Columns

Case\_Age = DATEDIFF(BPVWorkQueueItem[loaded],BPVWorkQueueItem[lastupdated], HOUR)

Classification = SWITCH(TRUE(),

AND(BPVWorkQueueItem[state] = 4, AND(NOT(BPVWorkQueueItem[ExceptionReasonDetect]),NOT(BPVWorkQueueItem[TagDetect]))), "E2E Complete",

AND(BPVWorkQueueItem[state] = 5, AND(NOT(BPVWorkQueueItem[ExceptionReasonDetect]),NOT(BPVWorkQueueItem[TagDetect]))), "System Exception",

"")

## Calculated Measures

Average Case Age = ROUND(AVERAGE(BPVWorkQueueItem[Case\_Age]), 2) & " Hour(s)"

Average Handling Time = ROUND(AVERAGE(BPVWorkQueueItem[worktime])/60,2) & " Minutes"

E2E Completion Rate = [Filtered\_E2EComplete]/[True\_Total]

Filtered\_E2EComplete = CALCULATE(

    COUNTA('BPVWorkQueueItem'[state]),

    FILTER(BPVWorkQueueItem,

BPVWorkQueueItem[Classification] = "E2E Complete" &&

'BPVWorkQueueItem'[Month Of Year] <= 'Time Range'[Time\_Range\_Filter\_Context\_Month\_UpperBound] &&

'BPVWorkQueueItem'[Month Of Year] > 'Time Range'[Time\_Range\_Filter\_Context\_Month\_LowerBound] &&

'BPVWorkQueueItem'[Week of Year] <= 'Time Range'[Time\_Range\_Filter\_Context\_Week\_UpperBound] &&

'BPVWorkQueueItem'[Week of Year] > 'Time Range'[Time\_Range\_Filter\_Context\_Week\_LowerBound] &&

'BPVWorkQueueItem'[Day of Week] <= 'Time Range'[Time\_Range\_Filter\_Context\_Day\_UpperBound] &&

'BPVWorkQueueItem'[Day of Week] > 'Time Range'[Time\_Range\_Filter\_Context\_Day\_LowerBound])

)

Filtered\_InProcess\_Deferred\_Locked\_Count = CALCULATE(

    COUNTA('BPVWorkQueueItem'[state]),

    FILTER(BPVWorkQueueItem, BPVWorkQueueItem[state] IN {1,2,3} &&

'BPVWorkQueueItem'[Month Of Year] <= 'Time Range'[Time\_Range\_Filter\_Context\_Month\_UpperBound] &&

'BPVWorkQueueItem'[Month Of Year] > 'Time Range'[Time\_Range\_Filter\_Context\_Month\_LowerBound] &&

'BPVWorkQueueItem'[Week of Year] <= 'Time Range'[Time\_Range\_Filter\_Context\_Week\_UpperBound] &&

'BPVWorkQueueItem'[Week of Year] > 'Time Range'[Time\_Range\_Filter\_Context\_Week\_LowerBound] &&

'BPVWorkQueueItem'[Day of Week] <= 'Time Range'[Time\_Range\_Filter\_Context\_Day\_UpperBound] &&

'BPVWorkQueueItem'[Day of Week] > 'Time Range'[Time\_Range\_Filter\_Context\_Day\_LowerBound])

)

Filtered\_System\_Exception = CALCULATE(

    COUNTA('BPVWorkQueueItem'[state]),

    FILTER(BPVWorkQueueItem, BPVWorkQueueItem[Classification] = "System Exception" &&

'BPVWorkQueueItem'[Month Of Year] <= 'Time Range'[Time\_Range\_Filter\_Context\_Month\_UpperBound] &&

'BPVWorkQueueItem'[Month Of Year] > 'Time Range'[Time\_Range\_Filter\_Context\_Month\_LowerBound] &&

'BPVWorkQueueItem'[Week of Year] <= 'Time Range'[Time\_Range\_Filter\_Context\_Week\_UpperBound] &&

'BPVWorkQueueItem'[Week of Year] > 'Time Range'[Time\_Range\_Filter\_Context\_Week\_LowerBound] &&

'BPVWorkQueueItem'[Day of Week] <= 'Time Range'[Time\_Range\_Filter\_Context\_Day\_UpperBound] &&

'BPVWorkQueueItem'[Day of Week] > 'Time Range'[Time\_Range\_Filter\_Context\_Day\_LowerBound])

)

Filtered\_Total\_Count =

CALCULATE(

    COUNTA('BPVWorkQueueItem'[state]),

    FILTER(BPVWorkQueueItem,'BPVWorkQueueItem'[Month Of Year] <= 'Time Range'[Time\_Range\_Filter\_Context\_Month\_UpperBound] &&

'BPVWorkQueueItem'[Month Of Year] > 'Time Range'[Time\_Range\_Filter\_Context\_Month\_LowerBound] &&

'BPVWorkQueueItem'[Week of Year] <= 'Time Range'[Time\_Range\_Filter\_Context\_Week\_UpperBound] &&

'BPVWorkQueueItem'[Week of Year] > 'Time Range'[Time\_Range\_Filter\_Context\_Week\_LowerBound] &&

'BPVWorkQueueItem'[Day of Week] <= 'Time Range'[Time\_Range\_Filter\_Context\_Day\_UpperBound] &&

'BPVWorkQueueItem'[Day of Week] > 'Time Range'[Time\_Range\_Filter\_Context\_Day\_LowerBound])

)

Filtered\_NIGO\_Referral = [Filtered\_Total\_Count] - [Filtered\_InProcess\_Deferred\_Locked\_Count] - [Filtered\_System\_Exception] - [Filtered\_E2EComplete]

InProcess\_Total\_Count = CALCULATE(

    COUNTA('BPVWorkQueueItem'[state]),

    FILTER(BPVWorkQueueItem,'BPVWorkQueueItem'[state] = 1)

)

True\_Total = [Filtered\_E2EComplete] + [Filtered\_NIGO\_Referral] + [Filtered\_System\_Exception] + [InProcess\_Total\_Count]

Rank\_BusinessException\_Measure = IF(RANKX(ALL(BPVWorkQueueItem[sanitized\_tags\_2]),

[Filtered\_NIGO\_Referral],,, Dense

) <= [SelectedTopNCriteria], "Top " & [SelectedTopNCriteria], "")

Rank\_SystemException\_Measure = IF(RANKX(ALL(BPVWorkQueueItem[Sanitized\_Exception\_Reason]),

[Filtered\_System\_Exception],,, Dense

) <= [SelectedTopNCriteria], "Top " & [SelectedTopNCriteria], "")

Selected\_LOB = SELECTEDVALUE(NameToRegionLOB[LOB], "No Selection")

Selected\_Process\_Name = SELECTEDVALUE(NameToRegionLOB[Process Name], "No Selection")

Selected\_Process\_Queue = SELECTEDVALUE(NameToRegionLOB[NameToQueue.Queue Name], "No Selection")

Selected\_Process\_Resource = SELECTEDVALUE(NameToRegionLOB[NameToResource.Resource], "No Selection")

Selected\_Time\_Range = SELECTEDVALUE('Time Range'[Time Range], "No Selection")

Last Refreshed At = FORMAT(LASTDATE('Last Refresh (Local)'[LastRefresh]),"mmm dd, yyyy hh:mm:ss AM/PM")

Time\_Range\_Filter\_Context\_Day\_LowerBound = VAR FilterContext = [Selected\_Time\_Range] RETURN

SWITCH(TRUE(),

FilterContext = "YESTERDAY", WEEKDAY(TODAY()) - 3,

FilterContext = "PREVIOUS WEEK", 0,

FilterContext = "PREVIOUS MONTH", 0,

FilterContext = "TODAY", WEEKDAY(TODAY()) - 2,

FilterContext = "CURRENT MONTH", 0)

Time\_Range\_Filter\_Context\_Day\_UpperBound = VAR FilterContext = [Selected\_Time\_Range] RETURN

SWITCH(TRUE(),

FilterContext = "YESTERDAY", WEEKDAY(TODAY()) - 2,

FilterContext = "PREVIOUS WEEK", 7,

FilterContext = "PREVIOUS MONTH", 7,

FilterContext = "TODAY", WEEKDAY(TODAY()),

FilterContext = "CURRENT MONTH", 7

)

Time\_Range\_Filter\_Context\_Month\_LowerBound = VAR FilterContext = [Selected\_Time\_Range] RETURN

SWITCH(TRUE(),

FilterContext = "YESTERDAY", MONTH(TODAY()) - 1,

FilterContext = "PREVIOUS WEEK", MONTH(TODAY()) - 2,

FilterContext = "PREVIOUS MONTH", MONTH(TODAY()) - 2,

FilterContext = "TODAY", MONTH(TODAY()) - 1,

FilterContext = "CURRENT MONTH", MONTH(TODAY()) - 1

)

Time\_Range\_Filter\_Context\_Month\_UpperBound = VAR FilterContext = [Selected\_Time\_Range] RETURN

SWITCH(TRUE(),

FilterContext = "YESTERDAY", MONTH(TODAY()),

FilterContext = "PREVIOUS WEEK", MONTH(TODAY()),

FilterContext = "PREVIOUS MONTH", MONTH(TODAY()) - 1,

FilterContext = "TODAY", MONTH(TODAY()),

FilterContext = "CURRENT MONTH", MONTH(TODAY())

)

Time\_Range\_Filter\_Context\_Week\_LowerBound = VAR FilterContext = [Selected\_Time\_Range] RETURN

SWITCH(TRUE(),

FilterContext = "YESTERDAY", WEEKNUM(TODAY()) - 1,

FilterContext = "PREVIOUS WEEK", WEEKNUM(TODAY()) - 2,

FilterContext = "PREVIOUS MONTH", 0,

FilterContext = "TODAY", WEEKNUM(TODAY()) - 1,

FilterContext = "CURRENT MONTH", 0

)

Time\_Range\_Filter\_Context\_Week\_UpperBound = VAR FilterContext = [Selected\_Time\_Range] RETURN

SWITCH(TRUE(),

FilterContext = "YESTERDAY", WEEKNUM(TODAY()),

FilterContext = "PREVIOUS WEEK", WEEKNUM(TODAY()) - 1,

FilterContext = "PREVIOUS MONTH", WEEKNUM(TODAY()),

FilterContext = "TODAY", WEEKNUM(TODAY()),

FilterContext = "CURRENT MONTH", WEEKNUM(TODAY())

)

SelectedFilterBy = SELECTEDVALUE(TopFilter[FilterBy], "ALL")

SelectedTopNCriteria = SELECTEDVALUE(TopFilter[RankingCriteria], 100)

TopN\_TableHeader\_NR = [SelectedFilterBy] & " Business Referral & NIGO"

TopN\_TableHeader\_SE = [SelectedFilterBy] & " Exception Reason"