Open AR Dynamic Aging

Friday, February 7, 2025 9:28 AM

[OpenAR Dynamic Aging Documentation.docx]()

Check with Markus and Rosemary Feb7 930am
-

Transaction Date and Last Settlement Date
So Trans Date before Selected Date
AND
Last Settlement Date AFTER Selected Date WITH Blanks

```
InvoicesAgingAmount =
VAR AsOfDate = [Selected Date]
RETURN
    SUMX(
        'Customer Transactions',
        VAR DaysDue = DATEDIFF('Customer Transactions'[Transaction Date], AsOfDate, DAY)
        VAR TransactionCheck = 'Customer Transactions'[Transaction Date] <= AsOfDate
        VAR SettlementCheck = ISBLANK('Customer Transactions'[Closed]) || 'Customer Transactions'[Closed] >
AsOfDate
        VAR AgingBucketCheck = DaysDue >= MIN('Aging Groups'[Min]) && DaysDue <= MAX('Aging Groups'[Max])
        RETURN
            IF(
                TransactionCheck && SettlementCheck && AgingBucketCheck,
                'Customer Transactions'[TotalAmountDue],
                0
            )
    )
```

```
InvoicesAgingAmount =
VAR AsOfDate = [Selected Date]
RETURN
    CALCULATE(
        SUM('Customer Transactions'[TotalAmountDue]),
        FILTER(
            'Customer Transactions',
            'Customer Transactions'[Transaction Date] <= AsOfDate &&
            (ISBLANK('Customer Transactions'[Closed]) || 'Customer Transactions'[Closed] > AsOfDate)
        ),
        FILTER(
            'Aging Groups',
            DATEDIFF('Customer Transactions'[Transaction Date], AsOfDate, DAY) >= 'Aging Groups'[Min] &&
DATEDIFF('Customer Transactions'[Transaction Date], AsOfDate, DAY) <= 'Aging Groups'[Max]
)
)
```

```
===================================

From scratch
Connected to the AMG Cust Trans table and now need
```

```
InvoicesAgingAmount =
VAR AsOfDate = [Selected Date]
RETURN
    SUMX(
        'Customer Transactions',
        VAR DaysDue = DATEDIFF('Customer Transactions'[Transaction Date], AsOfDate, DAY)
        RETURN
            IF(
                DaysDue >= MIN('Aging Groups'[Min]) && DaysDue <= MAX('Aging Groups'[Max]),
                'Customer Transactions'[TotalAmountDue],
                0
            )
    )
```

```
  ,

SelectedDateFilter =
VAR SelectedDate = SELECTEDVALUE('User Selected Date'[Date])
RETURN
IF(
   NOT ISBLANK(SelectedDate) &&
   MAX('Customer Transactions'[Transaction Date]) < SelectedDate &&
   (MAX('Customer Transactions'[Last Settle Date]) > SelectedDate || MAX('Customer Transactions'[Last Settle
Date]) = DATE(1900,1,1)),
   1,
   0
)
```

```
DateTable =
ADDCOLUMNS (
  CALENDAR (DATE(1900, 1, 1), TODAY() + 365),  -- Extending 1 year into the future
  "Year", YEAR([Date]),
  "Month Name", FORMAT([Date], "MMMM"),
  "Month Num", MONTH([Date]),
  "Quarter", "Q" & FORMAT([Date], "Q"),
  "Weekday", FORMAT([Date], "dddd"),
  "Weekday Num", WEEKDAY([Date], 2)  -- (Monday = 1, Sunday = 7)
)
```

```
Aging Bucket =
SWITCH(
  TRUE(),
  'Customer Transactions'[Due Date] > TODAY(), "Not Due",
  TODAY() - 'Customer Transactions'[Due Date] <= 30, "0-30 Days",
  TODAY() - 'Customer Transactions'[Due Date] <= 60, "31-60 Days",
  TODAY() - 'Customer Transactions'[Due Date] <= 90, "61-90 Days",
  TODAY() - 'Customer Transactions'[Due Date] <= 120, "91-120 Days",
  TODAY() - 'Customer Transactions'[Due Date] > 120, "120+ Days",
  "Unknown"
)
```

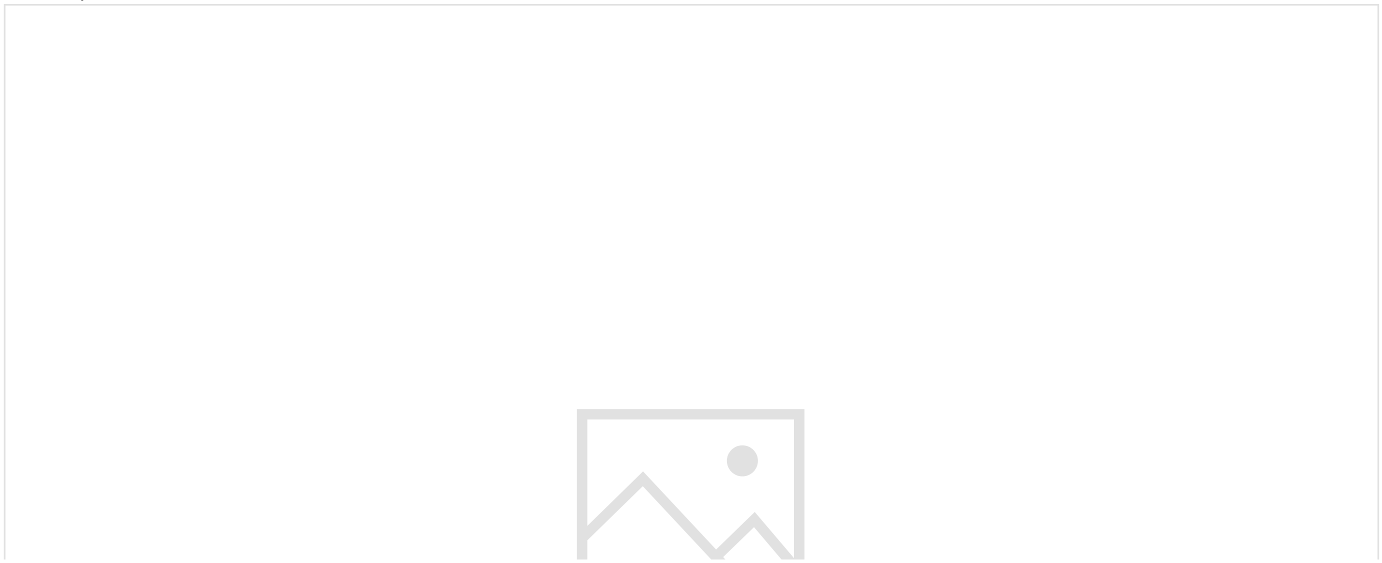
```
AgingBuckets =
DATATABLE (
  "Aging Bucket", STRING,
  "SortOrder", INTEGER,
  {
     {"Not Due", 0},
     {"0-30 Days", 1},
     {"31-60 Days", 2},
     {"61-90 Days", 3},
     {"91-120 Days", 4},
     {"120+ Days", 5}
  }
)
```

```
Open AR =
VAR SelectedDate = MAX('DateTable'[Date])  -- Get selected date from slicer
RETURN
CALCULATE(
  SUM('Customer Transactions'[Invoice Amount]) -
  SUM('Customer Transactions'[Payment Amount]),
  'Customer Transactions'[Due Date] <= SelectedDate,  -- Only consider invoices due before the selected date
  'Customer Transactions'[Payment Date] > SelectedDate || ISBLANK('Customer Transactions'[Payment Date])  -- Ignore fully
paid invoices
)
```

```
Open AR =
VAR SelectedDate = MAX('DateTable'[Date])  -- Get selected date from slicer
RETURN
CALCULATE(
  SUM('Customer Transactions'[Total Amount Due]),
  'Customer Transactions'[Transaction Date] <= SelectedDate  -- Include transactions before or on selected date
  'Customer Transactions'[Transaction Date] <= SelectedDate,  -- Include transactions before or on selected date
  'Customer Transactions'[Settlement Date] > SelectedDate ||
  ISBLANK('Customer Transactions'[Settlement Date])  -- Consider only unsettled transactions
)



AgingBuckets =
DATATABLE (
  "Aging Group", STRING,
  "SortOrder", INTEGER,
  "MinDaysPastDue", INTEGER,
  "MaxDaysPastDue", INTEGER,
  "JoinDaysLeft", INTEGER,
  {
     {"Not Due", 0, -1000, 0, -1000},
     {"0-30 Days", 1, 1, 30, BLANK()},
     {"31-60 Days", 2, 31, 60, BLANK()},
     {"61-90 Days", 3, 61, 90, BLANK()},
     {"91-120 Days", 4, 91, 120, BLANK()},
     {"120+ Days", 5, 121, 1000, BLANK()}
  }
) DaysPastDue =
VAR SelectedDate = MAX('DateTable'[Date])  -- Selected Date from slicer
RETURN
SELECTEDDATE - 'Customer Transactions'[Transaction Date]

Aging Bucket =
LOOKUPVALUE(
  AgingBuckets[Aging Group],
  AgingBuckets[MinDaysPastDue],
  MAXX(FILTER(AgingBuckets, 'Customer Transactions'[DaysPastDue] >= AgingBuckets[MinDaysPastDue]),
AgingBuckets[MinDaysPastDue]),
  AgingBuckets[MaxDaysPastDue],
  MINX(FILTER(AgingBuckets, 'Customer Transactions'[DaysPastDue] <= AgingBuckets[MaxDaysPastDue]),
AgingBuckets[MaxDaysPastDue])
)

Now = Table.AddColumn(#"Changed Type1", "Now", each DateTime.FixedLocalNow())

DateJoin = Table.AddColumn(Now, "DateJoin", each if [Due Date] = #datetime(1900, 1, 1, 0, 0, 0) then [Transaction Date]
else [Due Date])

Open Transaction = Table.AddColumn(DateJoin, "Open Transaction", each if [Closed] = #datetime(1900, 1, 1, 0, 0, 0)
then true else false)

Balance = Table.AddColumn(#"Open Transaction", "Balance", each [Reporting Currency Amount]-[Settle Amount
Currency])

Past Due Amount = Table.AddColumn(Balance, "Past Due Amount", each if [Invoice Due Date] <= [Now] and [Open
Transaction] then [Balance]
   else null)

TotalAmountDue = Table.AddColumn(#"Changed Type" , "TotalAmountDue", each if [Open Transaction] then [Balance]
else 0, Currency.Type)

TotalPaymentsReceived = Table.AddColumn(AddTotalAmountDue, "TotalPaymentsReceived", each if [Transaction Type]
= "Payment" then [Reporting Currency Amount] else 0, Currency.Type)

TotalInvoices = Table.AddColumn(AddTotalPaymentsReceived, "TotalInvoices", each if [Transaction Type] = "Sales
Order" then [Reporting Currency Amount] else 0, Currency.Type)

DaysDue = Table.AddColumn(AddTotalInvoices, "DaysDue", each if [Open Transaction] then Duration.Days([Now] -
[Invoice Due Date]) else 0, Int64.Type)

AgingBucket = Table.AddColumn(AddDaysDue, "AgingBucket", each if [DaysDue] <= 0 then "Current" else if [DaysDue]
<= 30 then "1-30 days" else if [DaysDue] <= 60 then "31-60 days" else if [DaysDue] <= 90 then "61-90 days" else if
[DaysDue] <= 120 then "91-120 days" else "Over 120 days")

OverdueInvoicesCount = Table.AddColumn(AddAgingBucket, "OverdueInvoicesCount", each if [Invoice Due Date]
< [Now] and [Open Transaction] then 1 else 0, Int64.Type)

OutstandingInvoices = Table.AddColumn(AddOverdueInvoices, "OutstandingInvoices", each if [Open Transaction] then 1
else 0, Int64.Type)

AgingBucketSeq = Table.AddColumn(OutstandingInvoices, "AgingBucketSeq", each if [DaysDue] <= 0 then 0 else if
[DaysDue] <= 30 then 1 else if [DaysDue] <= 60 then 2 else if [DaysDue] <= 90 then 3 else if [DaysDue] <= 120 then 4 else
5)
```

```
DaysPastDueDynamic =
VAR SelectedDate = SELECTEDVALUE(DateSelectionTable[Date], TODAY())
VAR ClosedDate = IF( ISBLANK( Transactions[Closed Date] ), DATE(1900,1,1), Transactions[Closed Date] )
RETURN
  IF(
     ClosedDate > SelectedDate || ClosedDate = DATE(1900,1,1),
     DATEDIFF(Transactions[Transaction Date], SelectedDate, DAY),
     BLANK()  // Exclude transactions that were closed before the selected date
  ) AgingBucketDynamic =
VAR DaysLate =
  VAR SelectedDate = SELECTEDVALUE(DateSelectionTable[Date], TODAY())
  VAR ClosedDate = IF( ISBLANK( Transactions[Closed Date] ), DATE(1900,1,1), Transactions[Closed Date] )
  RETURN
     IF(
        ClosedDate > SelectedDate || ClosedDate = DATE(1900,1,1),
        DATEDIFF(Transactions[Transaction Date], SelectedDate, DAY),
        BLANK()
     )
RETURN
  SWITCH(
     TRUE(),
     ISBLANK(DaysLate), BLANK(),  // Exclude closed transactions
     DaysLate <= 30, "0-30 Days",
     DaysLate <= 60, "31-60 Days",
     DaysLate <= 90, "61-90 Days",
     DaysLate > 90, "90+ Days"
  )
```

==
Feb 12, 2025
Closest yet'



CODE USED

```
AmountChosenDate = 
IF(NOT ISBLANK('Customer Transactions'[DaysPastDueDynamicWClosed]),'Customer Transactions'[AMOUNTCUR],0)


AgingBucketDynamicWClosed = 
   SWITCH(
       TRUE(),
       ISBLANK('Customer Transactions'[DaysPastDueDynamicWClosed]), BLANK(),  // Exclude closed transactions
       'Customer Transactions'[DaysPastDueDynamicWClosed] <= 30, "0-30 Days",
       'Customer Transactions'[DaysPastDueDynamicWClosed] <= 60, "31-60 Days",
       'Customer Transactions'[DaysPastDueDynamicWClosed] <= 90, "61-90 Days",
       'Customer Transactions'[DaysPastDueDynamicWClosed] <= 120, "91-120 Days",
       'Customer Transactions'[DaysPastDueDynamicWClosed] > 120, "Over 120 Days"
   )


DaysPastDueDynamicWClosed = 
VAR SelectedDate = SELECTEDVALUE(Date_Table[Date], TODAY())
VAR SettledDate = IF( ISBLANK( 'Customer Transactions'[LASTSETTLEDATE]), DATE(1900,1,1), 'Customer 
Transactions'[LASTSETTLEDATE] )
RETURN
   IF(
       OR(SettledDate >= SelectedDate , SettledDate = DATE(1900,1,1)),
       DATEDIFF('Customer Transactions'[TRANSDATE], SelectedDate, DAY),
       BLANK()  // Exclude transactions that were closed before the selected date
   )
```

AX 10031.72
43
App 3747.85
Cha 3.85