```
= let fnDateTable = ( StartDate as date, EndDate as date, optional FYStartMonthNum as number,
optional Holidays as list, optional WDStartNum as number, optional AddRelativeNetWorkdays as logical
) as table =>

 let

    FYStartMonth = List.Select( {1..12}, each _ = FYStartMonthNum ){0}? ?? 1,

    WDStart = List.Select( {0..1}, each _ = WDStartNum ){0}? ?? 0,

    CurrentDate = #date(2015, 5, 6),

    DayCount = Duration.Days( Duration.From( EndDate - StartDate)) +1,

    Source = List.Dates( StartDate, DayCount, #duration(1,0,0,0)),

    AddToday = if EndDate < CurrentDate then List.Combine( {Source, {CurrentDate}}) else Source,

    ToTable = Table.FromList(AddToday, Splitter.SplitByNothing(), type table [Date = Date.Type] ),

    InsertYear = Table.AddColumn(ToTable, "Year", each Date.Year([Date]), type number),

    InsertYearOffset = Table.AddColumn(InsertYear, "CurrYearOffset", each Date.Year([Date]) -
Date.Year( Date.From(CurrentDate)), type number),

    InsertCompletedYear = Table.AddColumn(InsertYearOffset, "YearCompleted", each
Date.EndOfYear([Date]) < Date.From( Date.EndOfYear(CurrentDate)), type logical),


    InsertQuarterNum = Table.AddColumn(InsertCompletedYear, "Quarter Number", each
Date.QuarterOfYear([Date]), type number),

    InsertQuarter = Table.AddColumn(InsertQuarterNum, "Quarter", each "Q" &
Number.ToText([Quarter Number]), type text),

    InsertStartOfQuarter = Table.AddColumn(InsertQuarter, "Start of Quarter", each
Date.StartOfQuarter([Date]), type date),

    InsertEndOfQuarter = Table.AddColumn(InsertStartOfQuarter, "End of Quarter", each
Date.EndOfQuarter([Date]), type date),

    InsertCalendarQtr = Table.AddColumn(InsertEndOfQuarter, "Quarter & Year", each "Q" &
Number.ToText( Date.QuarterOfYear([Date])) & Date.ToText([Date], [Format = " yyyy"]), type text),

    InsertQuarternYear = Table.AddColumn(InsertCalendarQtr, "QuarternYear", each [Year] * 10 +
[Quarter Number], type number),

    InsertQuarterOffset = Table.AddColumn(InsertQuarternYear, "CurrQuarterOffset", each ((4 *
Date.Year([Date])) +  Date.QuarterOfYear([Date])) - ((4 * Date.Year(Date.From(CurrentDate))) +
Date.QuarterOfYear(Date.From(CurrentDate))), type number),

    InsertCompletedQuarter = Table.AddColumn(InsertQuarterOffset, "QuarterCompleted", each
Date.EndOfQuarter([Date]) < Date.From(Date.EndOfQuarter(CurrentDate)), type logical),


    InsertMonth = Table.AddColumn(InsertCompletedQuarter, "Month", each Date.Month([Date]), type
number),
    InsertStartOfMonth = Table.AddColumn(InsertMonth, "Start of Month", each
Date.StartOfMonth([Date]), type date),

    InsertEndOfMonth = Table.AddColumn(InsertStartOfMonth, "End of Month", each
Date.EndOfMonth([Date]), type date),

    InsertCalendarMonth = Table.AddColumn(InsertEndOfMonth, "Month & Year", each Text.Proper(
Date.ToText([Date], [Format = "MMM yyyy"])), type text),

    InsertMonthnYear = Table.AddColumn(InsertCalendarMonth , "MonthnYear", each [Year] * 100 +
[Month], type number),

    InsertMonthOffset = Table.AddColumn(InsertMonthnYear, "CurrMonthOffset", each ((12 *
Date.Year([Date])) +  Date.Month([Date])) - ((12 * Date.Year(Date.From(CurrentDate))) +
Date.Month(Date.From(CurrentDate))), type number),

    InsertCompletedMonth = Table.AddColumn(InsertMonthOffset, "MonthCompleted", each
Date.EndOfMonth([Date]) < Date.From(Date.EndOfMonth(CurrentDate)), type logical),

    InsertMonthName = Table.AddColumn(InsertCompletedMonth, "Month Name", each Text.Proper(
Date.ToText([Date], "MMMM")), type text),

    InsertMonthShort = Table.AddColumn( InsertMonthName, "Month Short", each Text.Proper(
Date.ToText([Date], "MMM")), type text),

    InsertMonthInitial = Table.AddColumn(InsertMonthShort, "Month Initial", each Text.Start([Month
Name], 1) & Text.Repeat( Character.FromNumber(8203), Date.Month([Date]) ), type text),

    InsertDayOfMonth = Table.AddColumn(InsertMonthInitial, "Day of Month", each Date.Day([Date]),
type number),


    InsertWeekNumber = Table.AddColumn(InsertDayOfMonth, "Week Number", each

     if Number.RoundDown((Date.DayOfYear([Date])-(Date.DayOfWeek([Date],
Day.Monday)+1)+10)/7)=0

     then Number.RoundDown((Date.DayOfYear(#date(Date.Year([Date])-1,12,31))-
(Date.DayOfWeek(#date(Date.Year([Date])-1,12,31), Day.Monday)+1)+10)/7)

     else if (Number.RoundDown((Date.DayOfYear([Date])-(Date.DayOfWeek([Date],
Day.Monday)+1)+10)/7)=53 and (Date.DayOfWeek(#date(Date.Year([Date]),12,31), Day.Monday)+1<4))

     then 1 else Number.RoundDown((Date.DayOfYear([Date])-(Date.DayOfWeek([Date],
Day.Monday)+1)+10)/7), type number),

    InsertStartOfWeek = Table.AddColumn(InsertWeekNumber, "Start of Week", each
Date.StartOfWeek([Date], Day.Monday), type date),

    InsertWeekEnding = Table.AddColumn(InsertStartOfWeek, "End of Week", each Date.EndOfWeek(
[Date], Day.Monday), type date),

    InsertCalendarWk = Table.AddColumn(InsertWeekEnding, "Week & Year", each "W" & Text.PadStart(
Text.From( [Week Number] ), 2, "0") & " " & Text.From(Date.Year( Date.AddDays(
Date.StartOfWeek([Date], Day.Monday), 3 ))), type text ),
    InsertWeeknYear = Table.AddColumn(InsertCalendarWk, "WeeknYear", each Date.Year(
Date.AddDays( Date.StartOfWeek([Date], Day.Monday), 3 )) * 100 + [Week Number],  Int64.Type),

    InsertWeekOffset = Table.AddColumn(InsertWeeknYear, "CurrWeekOffset", each
(Number.From(Date.StartOfWeek([Date], Day.Monday))-Number.From(Date.StartOfWeek(CurrentDate,
Day.Monday)))/7, type number),

    InsertCompletedWeek = Table.AddColumn(InsertWeekOffset, "WeekCompleted", each
Date.EndOfWeek( [Date], Day.Monday) < Date.From(Date.EndOfWeek(CurrentDate, Day.Monday)), type
logical),


    InsertDayWeek = Table.AddColumn(InsertCompletedWeek, "Day of Week Number", each
Date.DayOfWeek([Date], Day.Monday) + WDStart, Int64.Type),

    InsertDayName = Table.AddColumn(InsertDayWeek, "Day of Week Name", each Text.Proper(
Date.ToText([Date], "dddd" )), type text),

    InsertDayInitial = Table.AddColumn(InsertDayName, "Day of Week Initial", each
Text.Proper(Text.Start([Day of Week Name], 1)) & Text.Repeat( Character.FromNumber(8203),
Date.DayOfWeek([Date], Day.Monday) + WDStart ), type text),

    InsertDayOfYear = Table.AddColumn(InsertDayInitial, "Day of Year", each Date.DayOfYear([Date]),
Int64.Type),

    InsertDayInt = Table.AddColumn(InsertDayOfYear, "DateInt", each [Year] * 10000 + [Month] * 100 +
[Day of Month], type number),

    InsertDayOffset = Table.AddColumn(InsertDayInt, "CurrDayOffset", each Number.From([Date]) -
Number.From(CurrentDate), type number),

    InsertIsAfterToday = Table.AddColumn(InsertDayOffset, "IsAfterToday", each not ([Date] <=
Date.From(CurrentDate)), type logical),

    InsertIsWorkingDay = Table.AddColumn(InsertIsAfterToday, "IsWeekDay", each if
Date.DayOfWeek([Date], Day.Monday) > 4 then false else true, type logical),

    InsertIsHoliday = Table.AddColumn(InsertIsWorkingDay, "IsHoliday", each if Holidays = null then
"Unknown" else List.Contains( Holidays, [Date] ), if Holidays = null then type text else type logical),

    InsertIsBusinessDay = Table.AddColumn(InsertIsHoliday, "IsBusinessDay", each if [IsWeekDay] = true
and [IsHoliday] <> true then true else false, type logical),

    InsertDayType = Table.AddColumn(InsertIsBusinessDay, "Day Type", each if [IsHoliday] = true then
"Holiday" else if [IsWeekDay] = false then "Weekend" else if [IsWeekDay] = true then "Weekday" else
null, type text),


    InsertISOYear = Table.AddColumn( InsertDayType, "ISO Year", each Date.Year( Date.AddDays(
Date.StartOfWeek([Date], Day.Monday), 3 )), type number),

    InsertISOqNum = Table.AddColumn(InsertISOYear, "ISO Quarter Number", each if [Week Number]
>39 then 4 else if [Week Number] >26 then 3 else if [Week Number] >13 then 2 else 1, Int64.Type),

    InsertISOqtr = Table.AddColumn(InsertISOqNum, "ISO Quarter", each "Q" & Number.ToText([ISO
Quarter Number]), type text),
    InsertISOQuarter = Table.AddColumn(InsertISOqtr, "ISO Quarter & Year", each "Q" &
Number.ToText([ISO Quarter Number]) & " " & Number.ToText([ISO Year]), type text),

    InsertISOqNy = Table.AddColumn(InsertISOQuarter, "ISO QuarternYear", each [ISO Year] * 10 + [ISO
Quarter Number], type number),


    // BufferTable = Table.Buffer(Table.Distinct( InsertISOqNy[[ISO Year], [DateInt]])),

    // InsertISOday = Table.AddColumn(InsertISOqNy, "ISO Day of Year", (OT) => Table.RowCount(
Table.SelectRows( BufferTable, (IT) => IT[DateInt] <= OT[DateInt] and IT[ISO Year] = OT[ISO Year])),
Int64.Type),

    AddFY = Table.AddColumn(InsertISOqNy, "Fiscal Year", each "FY" & (if [Month] >= FYStartMonth and
FYStartMonth >1 then Text.From([Year] +1) else Text.From([Year])), type text),

    //AddFYs = Table.AddColumn(AddFY, "Fiscal Year short", each "FY" & (if [Month] >= FYStartMonth
and FYStartMonth >1 then Text.PadEnd( Text.End( Text.From([Year] +1), 2), 2, "0") else Text.End(
Text.From([Year]), 2)), type text),

    AddFQ = Table.AddColumn(AddFY, "Fiscal Quarter", each "FQ" & Text.From( Number.RoundUp(
Date.Month( Date.AddMonths( [Date], - (FYStartMonth -1) )) / 3 )) & " " & (if [Month] >= FYStartMonth
and FYStartMonth >1 then Text.From([Year] +1) else Text.From([Year])), type text),

    AddFQnYr = Table.AddColumn(AddFQ, "FQuarternYear", each (if [Month] >= FYStartMonth and
FYStartMonth >1 then [Year] +1 else [Year]) * 10 + Number.RoundUp( Date.Month( Date.AddMonths(
[Date], - (FYStartMonth -1) )) / 3 ), type number),

    AddFM = Table.AddColumn(AddFQnYr, "Fiscal Period Number", each if [Month] >= FYStartMonth and
FYStartMonth >1 then [Month] - (FYStartMonth-1) else if [Month] >= FYStartMonth and FYStartMonth
=1 then [Month] else [Month] + (12-FYStartMonth+1), type number),

    AddFP = Table.AddColumn(AddFM, "Fiscal Period", each "FP" & Text.PadStart( Text.From([Fiscal
Period Number]), 2, "0") & " " & (if [Month] >= FYStartMonth and FYStartMonth >1 then
Text.From([Year] +1) else Text.From([Year])), type text),

    AddFMnYr = Table.AddColumn(AddFP , "FPeriodnYear", each (if [Month] >= FYStartMonth and
FYStartMonth >1 then [Year] +1 else [Year]) * 100 + [Fiscal Period Number], type number),

    FYCalendarStart = #date( Date.Year(StartDate)-1, FYStartMonth, 1 ),

    InsertFFD = Table.AddColumn( AddFMnYr, "FiscalFirstDay", each if [Month] >= FYStartMonth and
FYStartMonth >1 then #date( Date.Year([Date])+1, FYStartMonth, 1) else #date( Date.Year([Date]),
FYStartMonth, 1), type date ),


    InitTable = Table.FromList( List.Transform( {Number.From(FYCalendarStart) ..
Number.From(EndDate)}, Date.From), Splitter.SplitByNothing(), type table [DateFW = Date.Type]),

    AddFFD = Table.AddColumn( InitTable, "FiscalFirstDay", each if Date.Month([DateFW]) <
FYStartMonth then #date(Date.Year([DateFW]), FYStartMonth, 1) else #date(Date.Year([DateFW]) + 1,
FYStartMonth, 1)),

    AddFWSD = Table.AddColumn( AddFFD, "FWStartDate", each
Date.AddYears(Date.StartOfWeek([DateFW], Day.Monday), 1)),
    Group1 = Table.Group( AddFWSD, {"FiscalFirstDay", "FWStartDate"}, {{"AllRows", each _, type table
[DateFW = nullable date, FiscalFirstDay = date, FWStartDate = date]}}),

    Group2 = Table.Group( Group1, {"FiscalFirstDay"}, {{"AllRows2", each _, type table [FiscalFirstDay =
date, FWStartDate = date, AllRows = table]}}),

    AddIndex = Table.AddColumn( Group2, "Custom", each Table.AddIndexColumn([AllRows2], "Fiscal
Week Number", 1, 1) )[[Custom]],

    ExpandG2 = Table.ExpandTableColumn( AddIndex, "Custom", {"FiscalFirstDay", "FWStartDate",
"AllRows", "Fiscal Week Number"}, {"FiscalFirstDay", "FWStartDate", "AllRows", "Fiscal Week
Number"}),

    ExpandG1 = Table.ExpandTableColumn( ExpandG2, "AllRows", {"DateFW"}, {"DateFW"} )[[DateFW],
[Fiscal Week Number]],

    MergeFYW = Table.Join( InsertFFD, {"Date"}, ExpandG1, {"DateFW"}, JoinKind.LeftOuter,
JoinAlgorithm.SortMerge ),

    FWlogic = List.Contains( {null}, FYStartMonthNum),

    UpdateFYWeek = if FWlogic then Table.ReplaceValue(MergeFYW, each [Fiscal Week Number], each if
FYStartMonth =1 then [Week Number] else [Fiscal Week Number], Replacer.ReplaceValue, {"Fiscal Week
Number"}) else MergeFYW,

    AddFYW = Table.AddColumn( UpdateFYWeek, "Fiscal Week", each if FWlogic then "F" & [#"Week &
Year"] else if FYStartMonth =1 then "FW" & Text.PadStart( Text.From([Fiscal Week Number]), 2, "0") &
Date.ToText([Date], " yyyy") else if Date.Month([Date]) < FYStartMonth then "FW" & Text.PadStart(
Text.From([Fiscal Week Number]), 2, "0") & Date.ToText([Date], " yyyy") else "FW" &
Text.PadStart(Text.From([Fiscal Week Number]), 2, "0") & " " & Text.From( Date.Year([Date])+1), type
text),

    InsertFWeeknYear = Table.AddColumn(AddFYW, "FWeeknYear", each if FWlogic then [WeeknYear]
else (if FYStartMonth =1 then Date.Year([Date]) else if Date.Month([Date]) < FYStartMonth then
Date.Year([Date]) else Date.Year([Date])+1) * 100 + [Fiscal Week Number],  Int64.Type),


    CurrentDateRecord = Table.SelectRows(InsertFWeeknYear, each ([Date] = CurrentDate)),

    CurrentISOyear = CurrentDateRecord{0}[ISO Year],

    CurrentISOqtr = CurrentDateRecord{0}[ISO Quarter Number],

    CurrentYear = CurrentDateRecord{0}[Year],

    CurrentMonth = CurrentDateRecord{0}[Month],

    CurrentFiscalFirstDay = CurrentDateRecord{0}[FiscalFirstDay],

    PrevFiscalFirstDay = Date.AddYears(CurrentFiscalFirstDay, -1),

    CurrentFY = CurrentDateRecord{0}[Fiscal Year],

    CurrentFQ = CurrentDateRecord{0}[FQuarternYear],

    CurrentFP = CurrentDateRecord{0}[FPeriodnYear],

    CurrentFW = CurrentDateRecord{0}[FWeeknYear],
    InsertISOYrOffset = Table.AddColumn(InsertFWeeknYear, "ISO CurrYearOffset", each [ISO Year] -
CurrentISOyear, type number),

    InsertISOQtrOffset = Table.AddColumn(InsertISOYrOffset, "ISO CurrQuarterOffset", each ((4 * [ISO
Year]) +  [ISO Quarter Number]) - ((4 * CurrentISOyear) + CurrentISOqtr), type number),

    InsertFYoffset = Table.AddColumn(InsertISOQtrOffset, "Fiscal CurrYearOffset", each try (if [Month] >=
FYStartMonth then [Year]+1 else [Year]) - (if CurrentMonth >= FYStartMonth then CurrentYear+1 else
CurrentYear) otherwise null, type number),

    InsertCurrentFY = Table.AddColumn(InsertFYoffset, "IsCurrentFY", each if [Fiscal Year] = CurrentFY
then true else false, type logical),

    InsertCurrentFQ = Table.AddColumn(InsertCurrentFY, "IsCurrentFQ", each if [FQuarternYear] =
CurrentFQ then true else false, type logical),

    InsertCurrentFP = Table.AddColumn(InsertCurrentFQ, "IsCurrentFP", each if [FPeriodnYear] =
CurrentFP then true else false, type logical),

    InsertCurrentFW = Table.AddColumn(InsertCurrentFP, "IsCurrentFW", each if [FWeeknYear] =
InsertISOYrOffset then true else false, type logical),

    InsertPYTD = Table.AddColumn(InsertCurrentFW, "IsPYTD", each if CurrentYear-1 = [Year] and [Day
of Year] <= CurrentDateRecord{0}[Day of Year] then true else false, type logical),

      ListPrevFYDates = List.Buffer( Table.SelectRows( Table.ExpandTableColumn( Table.NestedJoin(

          Table.AddIndexColumn( Table.RenameColumns( Table.TransformColumnTypes( Table.FromList(
List.Dates( PrevFiscalFirstDay, Number.From(CurrentFiscalFirstDay-
PrevFiscalFirstDay),#duration(1,0,0,0)), Splitter.SplitByNothing()),{{"Column1", type date}}),
{{"Column1", "DateFY"}}), "Index", 1, 1), {"Index"},

          Table.AddIndexColumn( Table.RenameColumns( Table.TransformColumnTypes( Table.FromList(
List.Dates( Date.AddYears( PrevFiscalFirstDay, -1), Number.From( PrevFiscalFirstDay - Date.AddYears(
PrevFiscalFirstDay, -1)),#duration(1,0,0,0)), Splitter.SplitByNothing()),{{"Column1", type date}}),
{{"Column1", "DateFY"}}), "Index", 1, 1)

          , {"Index"}, "Table", JoinKind.LeftOuter), "Table", {"DateFY"}, {"PrevDateFY"}), each [DateFY] <=
CurrentDate)[PrevDateFY] ),

    InsertPFYTD = Table.AddColumn(InsertPYTD, "IsPFYTD", each if [Fiscal CurrYearOffset] = -1 and
List.Contains(ListPrevFYDates, [Date] ) then true else false, type logical),

    InsertNetWorkdays = if AddRelativeNetWorkdays = true then Table.AddColumn(InsertPFYTD,
"Relative Networkdays", each fxNETWORKDAYS( StartDate, [Date], Holidays ), type number ) else
InsertPFYTD,

    fxNETWORKDAYS = (StartDate, EndDate, optional Holidays as list) =>

      let

        ListOfDates = List.Dates( StartDate, Number.From(EndDate-StartDate)+1, Duration.From(1) ),

        DeleteHolidays = if Holidays = null then ListOfDates else List.Difference( ListOfDates,
List.Transform(Holidays, Date.From )),
      DeleteWeekends = List.Select( DeleteHolidays, each Date.DayOfWeek( _, Day.Monday) < 5 ),

      CountDays = List.Count( DeleteWeekends)

     in

      CountDays,

    RemoveToday = Table.RemoveColumns( if EndDate < CurrentDate then
Table.SelectRows(InsertNetWorkdays, each ([Date] <> CurrentDate)) else InsertNetWorkdays, {"Day of
Year", "FiscalFirstDay"}),

    ChType = Table.TransformColumnTypes(RemoveToday,{{"Year", Int64.Type}, {"Quarter Number",
Int64.Type}, {"Month", Int64.Type}, {"Day of Month", Int64.Type}, {"DateInt", Int64.Type}, {"Day of
Week Number", Int64.Type}, {"ISO CurrYearOffset", Int64.Type}, {"ISO QuarternYear", Int64.Type}, {"ISO
CurrQuarterOffset", Int64.Type}, {"Week Number", Int64.Type}, {"WeeknYear", Int64.Type},
{"MonthnYear", Int64.Type}, {"QuarternYear", Int64.Type}, {"FQuarternYear", Int64.Type}, {"Fiscal
Period Number", Int64.Type}, {"FPeriodnYear", Int64.Type}, {"CurrWeekOffset", Int64.Type},
{"CurrMonthOffset", Int64.Type}, {"CurrQuarterOffset", Int64.Type}, {"CurrYearOffset", Int64.Type},
{"Fiscal CurrYearOffset", Int64.Type}, {"Fiscal Week Number", Int64.Type}}),

    ReorderCols = Table.ReorderColumns(ChType,{"Date", "Year", "CurrYearOffset", "YearCompleted",
"Quarter Number", "Quarter", "Start of Quarter", "End of Quarter", "Quarter & Year", "QuarternYear",
"CurrQuarterOffset", "QuarterCompleted", "Month", "Start of Month", "End of Month", "Month &
Year", "MonthnYear", "CurrMonthOffset", "MonthCompleted", "Month Name", "Month Short", "Month
Initial", "Day of Month", "Week Number", "Start of Week", "End of Week", "Week & Year",
"WeeknYear", "CurrWeekOffset", "WeekCompleted", "Day of Week Number", "Day of Week Name",
"Day of Week Initial", "DateInt", "CurrDayOffset", "IsAfterToday", "IsWeekDay", "IsHoliday",
"IsBusinessDay", "Day Type", "ISO Year", "ISO CurrYearOffset", "ISO Quarter Number", "ISO Quarter",
"ISO Quarter & Year", "ISO QuarternYear", "ISO CurrQuarterOffset", "Fiscal Year", "Fiscal
CurrYearOffset", "Fiscal Quarter", "FQuarternYear", "Fiscal Period Number", "Fiscal Period",
"FPeriodnYear", "DateFW", "Fiscal Week Number", "Fiscal Week", "FWeeknYear", "IsCurrentFY",
"IsCurrentFQ", "IsCurrentFP", "IsCurrentFW", "IsPYTD", "IsPFYTD"}),

    ListCols = if FWlogic then Table.RemoveColumns(ReorderCols,{"ISO Quarter Number", "Fiscal Year",
"Fiscal Quarter", "FQuarternYear", "Fiscal Period Number", "Fiscal Period", "FPeriodnYear", "DateFW",
"Fiscal Week Number", "Fiscal Week", "FWeeknYear", "Fiscal CurrYearOffset", "IsCurrentFQ",
"IsCurrentFP", "IsCurrentFW"}) else Table.RemoveColumns(ReorderCols,{"Fiscal Period Number",
"DateFW", "Fiscal Week Number", "ISO Quarter Number"})

  in

    ListCols,

    Documentation = [

    Documentation.Name =  " fxCalendar",

    Documentation.Description = " Date table function to create an ISO-8601 calendar",

    Documentation.LongDescription = " Date table function to create an ISO-8601 calendar",

    Documentation.Category = " Table",

    Documentation.Version = " 2.01: full code review",

    Documentation.Source = " local",
    Documentation.Author = " Melissa de Korte",
    Documentation.Examples = { [Description =  " See: https://forum.enterprisedna.co/t/extended-date-
table-power-query-m-function/6390",
     Code = " Optional paramters: #(lf)
     (FYStartMonthNum) Month number the fiscal year starts, Januari if omitted #(lf)
     (Holidays) Select a query (and column) that contains a list of holiday dates #(lf)
     (WDStartNum) Switch default weekday numbering from 0-6 to 1-7 by entering a 1 #(lf)
     (AddRelativeNetWorkdays) if true adds a Relative Networkdays column to the date table #(lf)
     #(lf)
     Important to note: #(lf)
     [Fiscal Week] starts on a Monday and can contain less than 7 days in a First- and/or Last Week of a
FY #(lf)
     [IsWeekDay] does not take holiday dates into account  #(lf)
     [IsBusinessDay] does take optional holiday dates into account  #(lf)
     [IsPYTD] and [IsPFYTD] compare Previous [Day of Year] with the Current [Day of Year] number, so
dates don't align in leap years #(lf)
     IMPORTANT! No Fiscal columns will be added if the (FYStartMonthNum) is omitted",
     Result = " " ] }
    ]
  in
  Value.ReplaceType( fnDateTable, Value.ReplaceMetadata( Value.Type( fnDateTable ), Documentation
))
```