let

/\*

Data Inicial. Especifique o início da sua tabela. Extremamente recomendado o uso de parâmetros.

Apesar de não aconselhável pode-se usar List.Min(Tabela[Coluna]) para obter a menor data de uma coluna ou ainda

List.Min(List.Combine({Tabela1[Coluna], Tabela2[Coluna2], TabelaN[ColunaN]}) quando precisar obter de várias colunas,

contudo é reforçado que pode causar perdas de performance durante a carga.

\*/

dataInicial = #date(2020, 1, 1),

/\*

Data Final. O padrão é o final do ano da data atual.

Pode se usar os mesmos recursos que a dataInicial

\*/

dataFinal = Date.EndOfYear(Date.From(DateTime.LocalNow())),

//Dia em que inicia a semana

inicioSemana = Day.Monday,

//Mês do início de cada ano fiscal

mesInicioAnoFiscal = 4,

/\*

Use o script do site da Anbima pré-configurado ou especifique a sua tabela de feriados.

Esta tabela deve conter conter apenas duas colunas: [Data] e [Feriado].

\*/

tabelaFeriados = Table.SelectRows(

Table.TransformColumnTypes(

Excel.Workbook(

Web.Contents("https://www.anbima.com.br/feriados/arqs/feriados\_nacionais.xls"),

true,

true

)

{[Name="Feriados"]}[Data]

[[Data],[Feriado]],

{{"Data", type date}}

),

each [Data] >= dataInicial and [Data] <= dataFinal

),

//Geração da tabela

tabela =

let

idioma = "pt-BR",

dataAtual = Date.From(DateTime.LocalNow()),

//Quantidade de dias

duracao = Duration.Days(dataFinal-dataInicial) + 1,

//Lista com todas as datas

listaDatas = List.Dates(dataInicial, duracao, #duration(1, 0, 0, 0))

in

//Invoca a função do construtor de tabela

#table(

//Define os nomes e tipos das colunas

type table[

Indice = Int64.Type,

Data = date,

DataOffset = Int64.Type,

DataAtual = text,

Ano = Int64.Type,

AnoInicio = date,

AnoFim = date,

AnoIndice = Int64.Type,

AnoDecrescenteNo = Int64.Type,

AnoDescrescente = Int64.Type,

AnoFiscal = Int64.Type,

AnoOffset = Int64.Type,

AnoAtual = text,

DiaDoMes = Int64.Type,

DiaDoAno = Int64.Type,

DiaDaSemanaNo = Int64.Type,

DiaDaSemana = text,

DiaDaSemanaAbrev = text,

DiaDaSemanaIniciais = text,

MesNo = Int64.Type,

Mes = text,

MesAbrev = text,

MesIniciais = text,

MesAnoNo = Int64.Type,

MesAno = text,

MesDiaNo = Int64.Type,

MesDia = text,

MesInicio = date,

MesFim = date,

MesIndice = Int64.Type,

MesOffset = Int64.Type,

MesAtual = text,

MesAbrevAtual = text,

MesAnoAtual = text,

TrimestreNo = Int64.Type,

TrimestreInicio = date,

TrimestreFim = date,

TrimestreAnoNo = Int64.Type,

TrimestreAno = text,

TrimestreIndice = Int64.Type,

TrimestreOffset = Int64.Type,

TrimestreAtual = text,

SemanaIsoNo = Int64.Type,

AnoIso = Int64.Type,

SemanaIsoAnoNo = Int64.Type,

SemanaIsoAno = text,

SemanaIsoInicio = date,

SemanaIsoFim = date,

SemanaIsoIndice = Int64.Type,

SemanaOffset = Int64.Type,

SemanaAtual = text,

SemanaDoMesNo = Int64.Type,

AnoSemanal = Int64.Type,

MesSemanalNo = Int64.Type,

MesSemanal = text,

MesSemanalAbrev = text,

MesAnoSemanalNo = Int64.Type,

MesAnoSemanal = text,

QuinzenaDoMesNo = Int64.Type,

QuinzenaMesNo = Int64.Type,

QuinzenaMes = text,

QuinzenaMesAnoNo = Int64.Type,

QuinzenaMesAno = text,

QuinzenaIndice = Int64.Type,

QuinzenaOffset = Int64.Type,

QuinzenaAtual = text,

SemestreDoAnoNo = Int64.Type,

SemestreAnoNo = Int64.Type,

SemestreAno = text,

SemestreIndice = Int64.Type,

SemestreOffset = Int64.Type,

SemestreAtual = text,

BimestreDoAnoNo = Int64.Type,

BimestreAnoNo = Int64.Type,

BimestreAno = text,

BimestreIndice = Int64.Type,

BimestreOffset = Int64.Type,

BimestreAtual = text,

Feriados = text,

DiaUtilNo = Int64.Type,

Diautil = text,

EstacaoAnoNo = Int64.Type,

EstacaoAno = text

],

//Transforma a lista de datas

List.Transform(

//Lista que será transformada

listaDatas,

//Lista das trasnformações

each {

//Indice

List.PositionOf(listaDatas, \_) + 1,

//Data

\_,

//DataOffset

Number.From(\_ - dataAtual),

//DataAtual =

let

offset = Number.From(\_ - dataAtual)

in

if offset = 0 then "Data Atual"

else if offset = -1 then "Data Anterior"

else if offset = 1 then "Próxima Data"

else Date.ToText(\_, [Format="dd/MM/yyyy"]),

//Ano

Date.Year(\_),

//AnoInicio

Date.StartOfYear(\_),

//AnoFinal

Date.EndOfYear(\_),

//AnoIndice

Date.Year(\_) - Date.Year(dataInicial) + 1,

//AnoDecrescenteNo

Date.Year(\_) \* -1,

//AnoDecrescente

Date.Year(\_),

//AnoFiscal

if Date.Month(\_) >= mesInicioAnoFiscal

then Date.Year(\_)

else Date.Year(\_) - 1,

//AnoOffset

Date.Year(\_) - Date.Year(dataAtual),

//AnoAtual

let

offset = Date.Year(\_) - Date.Year(dataAtual)

in

if offset = 0 then "Ano Atual"

else if offset = -1 then "Ano Anterior"

else if offset = 1 then "Próximo Ano"

else Date.ToText(\_,[Format="yyyy"]),

//DiaDoMes

Date.Day(\_),

//DiaDoAno

Date.DayOfYear(\_),

//DiaDaSemanaNo

Date.DayOfWeek(\_, inicioSemana) + 1,

//DiaDaSemana

Text.Proper(Date.DayOfWeekName(\_, idioma)),

//DiaDaSemanaAbrev

Text.Start(Text.Proper(Date.DayOfWeekName(\_, idioma)), 3),

//DiaDaSemanaIniciais

Text.Repeat(

Character.FromNumber(8203),

7 - Date.DayOfWeek(\_, inicioSemana) + 1

)

& Text.Start(Text.Proper(Date.DayOfWeekName(\_, idioma)), 1),

//MesNo

Date.Month(\_),

//Mes

Text.Proper(Date.MonthName(\_, idioma)),

//MesAbrev

Text.Start(Text.Proper(Date.MonthName(\_, idioma)), 3),

//MesIniciais

Text.Repeat(

Character.FromNumber(8203),

12 - Date.Month(\_)

)

& Text.Start(Text.Proper(Date.MonthName(\_, idioma)), 1),

//MesAnoNo

Date.Year(\_) \* 100 + Date.Month(\_),

//MesAno

Text.Proper(Date.ToText(\_, [Format="MMM/yy", Culture=idioma])),

//MesDiaNo

Date.Month(\_) \* 100 + Date.Day(\_),

//MesDia

Text.Proper(Date.ToText(\_, [Format="MMM/dd", Culture=idioma])),

//MesInicio

Date.StartOfMonth(\_),

//MesFim

Date.EndOfMonth(\_),

//IndiceMes

12 \* (Date.Year(\_) - Date.Year(dataInicial)) + Date.Month(\_),

//MesOffset

let

contexto = Date.Year(\_) \* 12 - 1 + Date.Month(\_),

atual = Date.Year(dataAtual) \* 12 - 1 + Date.Month(dataAtual)

in

contexto - atual,

//MesAtual =

let

contexto = Date.Year(\_) \* 12 - 1 + Date.Month(\_),

atual = Date.Year(dataAtual) \* 12 - 1 + Date.Month(dataAtual),

offset = contexto - atual

in

if offset = 0 then "Mês Atual"

else if offset = -1 then "Mês Anterior"

else if offset = 1 then "Próximo Mês"

else Text.Proper(Date.MonthName(\_, idioma)),

//MesAbrevAtual =

let

contexto = Date.Year(\_) \* 12 - 1 + Date.Month(\_),

atual = Date.Year(dataAtual) \* 12 - 1 + Date.Month(dataAtual),

offset = contexto - atual

in

if offset = 0 then "Mês Atual"

else if offset = -1 then "Mês Anterior"

else if offset = 1 then "Próximo Mês"

else Text.Proper(Text.Start(Date.MonthName(\_, idioma), 3)),

//MesAnoAtual =

let

contexto = Date.Year(\_) \* 12 - 1 + Date.Month(\_),

atual = Date.Year(dataAtual) \* 12 - 1 + Date.Month(dataAtual),

offset = contexto - atual

in

if offset = 0 then "Mês Atual"

else if offset = -1 then "Mês Anterior"

else if offset = 1 then "Próximo Mês"

else Text.Proper(Date.ToText(\_, [Format="MMM/yy", Culture=idioma])),

//TrimestreNo

Date.QuarterOfYear(\_),

//TrimestreInicio

Date.StartOfQuarter(\_),

//TrimestreFim

Date.EndOfQuarter(\_),

//TrimestreAnoNo

Date.Year(\_) \* 100 + Date.QuarterOfYear(\_),

//TrimestreAno

"T" & Text.From(Date.QuarterOfYear(\_)) & "/" & Text.From(Date.Year(\_)),

//TrimestreIndice

4 \* (Date.Year(\_) - Date.Year(dataInicial)) + Date.QuarterOfYear(\_),

//TrimestreOffset

let

contexto = Date.Year(\_) \* 4 - 1 + Date.QuarterOfYear(\_),

atual = Date.Year(dataAtual) \* 4 - 1 + Date.QuarterOfYear(dataAtual)

in

contexto - atual,

//TrimestreAtual

let

contexto = Date.Year(\_) \* 4 - 1 + Date.QuarterOfYear(\_),

atual = Date.Year(dataAtual) \* 4 - 1 + Date.QuarterOfYear(dataAtual),

offset = contexto - atual

in

if offset = 0 then "Trimestre Atual"

else if offset = -1 then "Trimestre Anterior"

else if offset = 1 then "Próximo Trimestre"

else "T" & Text.From(Date.QuarterOfYear(\_)) & "/" & Text.From(Date.Year(\_)),

//SemanaIsoDoAnoNo

let

quintaNaSemana = Date.AddDays(\_, 3 - Date.DayOfWeek(\_, Day.Monday)),

inicioAnoQuintaNaSemana = #date(Date.Year(quintaNaSemana), 1, 1),

difDias = Duration.Days(quintaNaSemana - inicioAnoQuintaNaSemana)

in

Number.IntegerDivide(difDias, 7, 0) + 1,

//AnoIso

let

quintaNaSemana = Date.AddDays(\_, 3 - Date.DayOfWeek(\_, Day.Monday)),

inicioAnoQuintaNaSemana = #date(Date.Year(quintaNaSemana), 1, 1),

difDias = Duration.Days(quintaNaSemana - inicioAnoQuintaNaSemana),

semanaDoAno = Number.IntegerDivide(difDias, 7, 0) + 1

in

Date.Year(Date.AddDays(\_, 26 - semanaDoAno)),

//SemanaIsoAnoNo

let

quintaNaSemana = Date.AddDays(\_, 3 - Date.DayOfWeek(\_, Day.Monday)),

inicioAnoQuintaNaSemana = #date(Date.Year(quintaNaSemana), 1, 1),

difDias = Duration.Days(quintaNaSemana - inicioAnoQuintaNaSemana),

semanaDoAno = Number.IntegerDivide(difDias, 7, 0) + 1,

ano = Date.Year(Date.AddDays(\_, 26 - semanaDoAno))

in

ano \* 100 + semanaDoAno,

//SemanaIsoAno

let

quintaNaSemana = Date.AddDays(\_, 3 - Date.DayOfWeek(\_, Day.Monday)),

inicioAnoQuintaNaSemana = #date(Date.Year(quintaNaSemana), 1, 1),

difDias = Duration.Days(quintaNaSemana - inicioAnoQuintaNaSemana),

semanaDoAno = Number.IntegerDivide(difDias, 7, 0) + 1,

ano = Date.Year(Date.AddDays(\_, 26 - semanaDoAno))

in

"S" & Text.PadStart(Text.From(semanaDoAno), 2, "0")

& "/" & Text.From(ano),

//SemanaIsoInicio

Date.StartOfWeek(\_, inicioSemana),

//SemanaIsoFim

Date.EndOfWeek(\_, inicioSemana),

//SemanaIsoIndice

Number.From(Date.StartOfWeek(\_, inicioSemana) - Date.StartOfWeek(dataInicial, inicioSemana))/7 + 1,

//SemanaOffset

Number.From(

Date.StartOfWeek(\_, inicioSemana)

- Date.StartOfWeek(dataAtual, inicioSemana)

) / 7,

//SemanaAtual =

let

offset = Number.From(

Date.StartOfWeek(\_, inicioSemana)

- Date.StartOfWeek(dataAtual, inicioSemana)

) / 7

in

if offset = 0 then "Semana Atual"

else if offset = -1 then "Semana Anterior"

else if offset = 1 then "Próxima Semana"

else

let

quintaNaSemana = Date.AddDays(\_, 3 - Date.DayOfWeek(\_, Day.Monday)),

inicioAnoQuintaNaSemana = #date(Date.Year(quintaNaSemana), 1, 1),

difDias = Duration.Days(quintaNaSemana - inicioAnoQuintaNaSemana),

semanaDoAno = Number.IntegerDivide(difDias, 7, 0) + 1,

ano = Date.Year(Date.AddDays(\_, 26 - semanaDoAno))

in

"S" & Text.PadStart(Text.From(semanaDoAno), 2, "0")

& "/" & Text.From(ano),

//SemanaDoMesNo

let

inicioMes = Date.StartOfMonth(Date.StartOfWeek(\_, inicioSemana)),

primeirosSeteDias = List.Dates(inicioMes, 7, #duration(1, 0, 0, 0 )),

primeiraSegundaFeira = List.Select(

primeirosSeteDias,

each Date.DayOfWeek(\_, inicioSemana) = 0

){0}

in

Number.RoundUp(Duration.Days(\_ - primeiraSegundaFeira) / 7 + 0.05),

//AnoSemanal

Date.Year(Date.StartOfWeek(\_, inicioSemana)),

//MesSemanalNo

Date.Month(Date.StartOfWeek(\_, inicioSemana)),

//MesSemanal

Text.Proper(Date.MonthName(Date.StartOfWeek(\_, inicioSemana), idioma)),

//MesSemanalAbrev

Text.Start(Text.Proper(Date.MonthName(Date.StartOfWeek(\_, inicioSemana), idioma)), 3),

//MesAnoSemanalNo

Date.Year(Date.StartOfWeek(\_, inicioSemana)) \* 100

+ Date.Month(Date.StartOfWeek(\_, inicioSemana)),

//MesAnoSemanal

Text.Proper(

Date.ToText(

Date.StartOfWeek(\_, inicioSemana),

[Format="MMM/yy", Culture=idioma]

)

),

//QuinzenaDoMesNo

if Date.Day(\_) <= 15 then 1 else 2,

//QuinzenaMesNo

let

quinzena = if Date.Day(\_) <= 15 then 1 else 2,

mes = Date.Month(\_)

in

mes \* 10 + quinzena,

//QuinzenaMes

let

quinzena = if Date.Day(\_) <= 15 then 1 else 2,

mes = Text.Proper(Text.Start(Date.MonthName(\_, idioma), 3))

in

"qui " & Text.From(quinzena) & "-" & mes,

//QuinzenaMesAnoNo

let

quinzena = if Date.Day(\_) <= 15 then 1 else 2,

ano = Date.Year(\_),

mes = Date.Month(\_)

in

ano \* 100 + mes \* 10 + quinzena,

//QuinzenaMesAno

let

quinzena = if Date.Day(\_) <= 15 then 1 else 2,

anoMes = Text.Proper(Date.ToText(\_, [Format="MMM/yy", Culture=idioma]))

in

"Q" & Text.From(quinzena) & "-" & anoMes,

//QuinzenaIndice

let

quinzena = if Date.Day(\_) <= 15 then 1 else 2

in

(24 \* (Date.Year(\_) - Date.Year(dataInicial)))

+ (2 \* (Date.Month(\_) - Date.Month(dataInicial)))

+ quinzena,

//QuinzenaOffset

let

contexto =

let

quinzena = if Date.Day(\_) <= 15 then 1 else 2

in

(24 \* (Date.Year(\_) - Date.Year(dataInicial)))

+ (2 \* (Date.Month(\_) - Date.Month(dataInicial)))

+ quinzena,

atual =

let

quinzena = if Date.Day(dataAtual) <= 15 then 1 else 2

in

(24 \* (Date.Year(dataAtual) - Date.Year(dataInicial)))

+ (2 \* (Date.Month(dataAtual) - Date.Month(dataInicial)))

+ quinzena

in

contexto - atual,

//QuinzenaAtual

let

offset =

let

contexto =

let

quinzena = if Date.Day(\_) <= 15 then 1 else 2

in

(24 \* (Date.Year(\_) - Date.Year(dataInicial)))

+ (2 \* (Date.Month(\_) - Date.Month(dataInicial)))

+ quinzena,

atual =

let

quinzena = if Date.Day(dataAtual) <= 15 then 1 else 2

in

(24 \* (Date.Year(dataAtual) - Date.Year(dataInicial)))

+ (2 \* (Date.Month(dataAtual) - Date.Month(dataInicial)))

+ quinzena

in

contexto - atual,

quinzenaMesAno =

let

quinzena = if Date.Day(\_) <= 15 then 1 else 2,

anoMes = Text.Proper(Date.ToText(\_, [Format="MMM/yy", Culture=idioma]))

in

"Q" & Text.From(quinzena) & "-" & anoMes

in

if offset = 0 then "Quinzena Atual"

else if offset = -1 then "Quinzena Anterior"

else if offset = 1 then "Próxima Quinzena"

else quinzenaMesAno,

//SemestreDoAnoNo

if Date.Month(\_) <= 6 then 1 else 2,

//SemestreAnoNo

let

semestre = if Date.Month(\_) <= 6 then 1 else 2,

ano = Date.Year(\_)

in

ano \* 100 + semestre,

//SemestreAno

let

semestre = if Date.Month(\_) <= 6 then 1 else 2,

ano = Date.Year(\_)

in

"S" & Text.From(semestre) & "-" & Text.From(ano),

//SemestreIndice

let

semestre = if Date.Month(\_) <= 6 then 1 else 2

in

(2 \* (Date.Year(\_) - Date.Year(dataInicial)))

+ semestre,

//SemestreOffset

let

contexto =

let

semestre = if Date.Month(\_) <= 6 then 1 else 2

in

(2 \* (Date.Year(\_) - Date.Year(dataInicial)))

+ semestre,

atual =

let

semestre = if Date.Month(dataAtual) <= 6 then 1 else 2

in

(2 \* (Date.Year(dataAtual) - Date.Year(dataInicial)))

+ semestre

in

contexto - atual,

//SemestreAtual

let

offset =

let

contexto =

let

semestre = if Date.Month(\_) <= 6 then 1 else 2

in

(2 \* (Date.Year(\_) - Date.Year(dataInicial)))

+ semestre,

atual =

let

semestre = if Date.Month(dataAtual) <= 6 then 1 else 2

in

(2 \* (Date.Year(dataAtual) - Date.Year(dataInicial)))

+ semestre

in

contexto - atual,

semestreAno =

let

semestre = if Date.Month(\_) <= 6 then 1 else 2,

ano = Date.Year(\_)

in

"S" & Text.From(semestre) & "-" & Text.From(ano)

in

if offset = 0 then "Semestre Atual"

else if offset = -1 then "Semestre Anterior"

else if offset = 1 then "Próximo Semestre"

else semestreAno,

//BimestreDoAnoNo

Number.RoundUp(Date.Month(\_)/2, 0),

//BimestreAnoNo

let

bimestre = Number.RoundUp(Date.Month(\_)/2, 0),

ano = Date.Year(\_)

in

ano \* 100 + bimestre,

//BimestreAno

let

bimestre = Number.RoundUp(Date.Month(\_)/2, 0),

ano = Date.Year(\_)

in

"B" & Text.From(bimestre) & "-" & Text.From(ano),

//BimestreIndice

let

bimestre = Number.RoundUp(Date.Month(\_)/2, 0)

in

(6 \* (Date.Year(\_) - Date.Year(dataInicial)))

+ bimestre,

//BimestreOffset

let

contexto =

let

bimestre = Number.RoundUp(Date.Month(\_)/2, 0)

in

(6 \* (Date.Year(\_) - Date.Year(dataInicial)))

+ bimestre,

atual =

let

bimestre = Number.RoundUp(Date.Month(dataAtual)/2, 0)

in

(6 \* (Date.Year(dataAtual) - Date.Year(dataInicial)))

+ bimestre

in

contexto - atual,

//BimestreAtual

let

offset =

let

contexto =

let

bimestre = Number.RoundUp(Date.Month(\_)/2, 0)

in

(6 \* (Date.Year(\_) - Date.Year(dataInicial)))

+ bimestre,

atual =

let

bimestre = Number.RoundUp(Date.Month(dataAtual)/2, 0)

in

(6 \* (Date.Year(dataAtual) - Date.Year(dataInicial)))

+ bimestre

in

contexto - atual,

bimestreAno =

let

bimestre = Number.RoundUp(Date.Month(\_)/2, 0),

ano = Date.Year(\_)

in

"B" & Text.From(bimestre) & "-" & Text.From(ano)

in

if offset = 0 then "Bimestre Atual"

else if offset = -1 then "Bimestre Anterior"

else if offset = 1 then "Próximo Bimestre"

else bimestreAno,

//Feriados

try tabelaFeriados{[Data=\_]}[Feriado] otherwise null,

//DiaUtilNo

let

feriado = try tabelaFeriados{[Data=\_]}[Feriado] otherwise null,

diaDaSemana = Date.DayOfWeek(\_, inicioSemana) + 1

in

if feriado <> null or List.Contains({6..7}, diaDaSemana) then 0 else 1,

//DiaUtil

let

feriado = try tabelaFeriados{[Data=\_]}[Feriado] otherwise null,

diaDaSemana = Date.DayOfWeek(\_, inicioSemana) + 1

in

if feriado <> null or List.Contains({6..7}, diaDaSemana) then "Dia Não Útil" else "Dia Útil",

//EstacoesAnoNo

let

ref = Date.Month(\_) \* 100 + Date.Day(\_)

in

if ref >= 321 and ref <= 620 then 1 else

if ref >= 621 and ref <= 921 then 2 else

if ref >= 922 and ref <= 1221 then 3 else

4,

//EstaoesAno

let

ref = Date.Month(\_) \* 100 + Date.Day(\_)

in

if ref >= 321 and ref <= 620 then "Outono" else

if ref >= 621 and ref <= 921 then "Inverno" else

if ref >= 922 and ref <= 1221 then "Primavera" else

"Verão"

}

)

)

in

tabela