Documentação técnica– Power BI

Data da documentação: 04/11/2024

Nome do relatório: Desafio\_RodoTransportes

Páginas

|  |
| --- |
| **Páginas** |
| Capa |
| Produtividade |
| Log |

Tabelas

|  |  |  |  |
| --- | --- | --- | --- |
| **Tabelas** | **Colunas** | **Tipo de dados** | **Coluna calculada?** |
| f\_manutencoes | Placa | string | Não |
| f\_manutencoes | Inicio | dateTime | Não |
| f\_manutencoes | Fim | dateTime | Não |
| f\_manutencoes | saldo\_manutencao | int64 | Não |
| f\_manutencoes | DataFiltro | dateTime | Não |
| f\_ocorrencias | Causa | string | Não |
| f\_ocorrencias | Data Hora Abertura | dateTime | Não |
| f\_ocorrencias | Data Hora Fechamento | dateTime | Não |
| f\_ocorrencias | Problema | string | Não |
| f\_ocorrencias | ORDEM | int64 | Não |
| f\_ocorrencias | Placa | string | Não |
| f\_ocorrencias | Origem | string | Não |
| f\_ocorrencias | DataFiltro | dateTime | Não |
| f\_viagens | Placa | string | Não |
| f\_viagens | ORDEM | int64 | Não |
| f\_viagens | ID Viagem | string | Não |
| f\_viagens | Cadastro da viagem | dateTime | Não |
| f\_viagens | Status monitoramento | string | Não |
| f\_viagens | Status viagem | string | Não |
| f\_viagens | Operação da viagem | string | Não |
| f\_viagens | Origem | string | Não |
| f\_viagens | Entrada na origem | dateTime | Não |
| f\_viagens | Saída da origem | dateTime | Não |
| f\_viagens | CPF\_CNPJ | int64 | Não |
| f\_viagens | Entrada na entrega | dateTime | Não |
| f\_viagens | Saída da entrega | dateTime | Não |
| f\_viagens | Entrada no destino | dateTime | Não |
| f\_viagens | Saída do destino | dateTime | Não |
| f\_viagens | Índice | int64 | Não |
| f\_viagens | Cliente | string | Não |
| f\_viagens | Destino | string | Não |
| f\_viagens | saldo\_origem | double | Não |
| f\_viagens | saldo\_entrega | double | Não |
| f\_viagens | saldo\_destino | double | Não |
| d\_calendario | DataIndice | int64 | Não |
| d\_calendario | Data | dateTime | Não |
| d\_calendario | DataOffset | int64 | Não |
| d\_calendario | DataNomeAtual | string | Não |
| d\_calendario | AnoNum | int64 | Não |
| d\_calendario | AnoInicio | dateTime | Não |
| d\_calendario | AnoFim | dateTime | Não |
| d\_calendario | AnoIndice | int64 | Não |
| d\_calendario | AnoDecrescenteNome | int64 | Não |
| d\_calendario | AnoDecrescenteNum | int64 | Não |
| d\_calendario | AnoFiscal | int64 | Não |
| d\_calendario | AnoOffset | int64 | Não |
| d\_calendario | AnoNomeAtual | string | Não |
| d\_calendario | DiaDoMesNum | int64 | Não |
| d\_calendario | DiaDoAnoNum | int64 | Não |
| d\_calendario | DiaDaSemanaNum | int64 | Não |
| d\_calendario | DiaDaSemanaNome | string | Não |
| d\_calendario | DiaDaSemanaNomeAbrev | string | Não |
| d\_calendario | DiaDaSemanaNomeIniciais | string | Não |
| d\_calendario | MesNum | int64 | Não |
| d\_calendario | MesNome | string | Não |
| d\_calendario | MesNomeAbrev | string | Não |
| d\_calendario | MesNomeIniciais | string | Não |
| d\_calendario | MesAnoNum | int64 | Não |
| d\_calendario | MesAnoNome | string | Não |
| d\_calendario | MesDiaNum | int64 | Não |
| d\_calendario | MesDiaNome | string | Não |
| d\_calendario | MesInicio | dateTime | Não |
| d\_calendario | MesFim | dateTime | Não |
| d\_calendario | MesIndice | int64 | Não |
| d\_calendario | MesOffset | int64 | Não |
| d\_calendario | MesNomeAtual | string | Não |
| d\_calendario | MesNomeAbrevAtual | string | Não |
| d\_calendario | MesAnoNomeAtual | string | Não |
| d\_calendario | TrimestreNum | int64 | Não |
| d\_calendario | TrimestreInicio | dateTime | Não |
| d\_calendario | TrimestreFim | dateTime | Não |
| d\_calendario | TrimestreAnoNum | int64 | Não |
| d\_calendario | TrimestreAnoNome | string | Não |
| d\_calendario | TrimestreIndice | int64 | Não |
| d\_calendario | TrimestreOffset | int64 | Não |
| d\_calendario | TrimestreAnoNomeAtual | string | Não |
| d\_calendario | SemanaIsoNum | int64 | Não |
| d\_calendario | AnoIsoNum | int64 | Não |
| d\_calendario | SemanaIsoAnoNum | int64 | Não |
| d\_calendario | SemanaIsoAnoNome | string | Não |
| d\_calendario | SemanaIsoInicio | dateTime | Não |
| d\_calendario | SemanaIsoFim | dateTime | Não |
| d\_calendario | SemanaIsoIndice | int64 | Não |
| d\_calendario | SemanaIsoOffset | int64 | Não |
| d\_calendario | SemanaIsoAnoNomeAtual | string | Não |
| d\_calendario | SemanaNum | int64 | Não |
| d\_calendario | SemanaAnoNum | int64 | Não |
| d\_calendario | SemanaAnoNome | string | Não |
| d\_calendario | SemanaInicio | dateTime | Não |
| d\_calendario | SemanaFim | dateTime | Não |
| d\_calendario | SemanaPeriodo | string | Não |
| d\_calendario | SemanaIndice | int64 | Não |
| d\_calendario | SemanaOffset | int64 | Não |
| d\_calendario | SemanaAnoNomeAtual | string | Não |
| d\_calendario | SemanaDoMesNum | int64 | Não |
| d\_calendario | AnoSemanalNum | int64 | Não |
| d\_calendario | MesSemanalNum | int64 | Não |
| d\_calendario | MesSemanalNome | string | Não |
| d\_calendario | MesSemanalNomeAbrev | string | Não |
| d\_calendario | MesAnoSemanalNum | int64 | Não |
| d\_calendario | MesAnoSemanalNome | string | Não |
| d\_calendario | SemanaDoMesPadraoNum | int64 | Não |
| d\_calendario | SemanaDoMesAnoPadraoNome | string | Não |
| d\_calendario | SemanaDoMesAnoPadraoNum | int64 | Não |
| d\_calendario | QuinzenaDoMesNum | int64 | Não |
| d\_calendario | QuinzenaMesNum | int64 | Não |
| d\_calendario | QuinzenaMesNome | string | Não |
| d\_calendario | QuinzenaMesAnoNum | int64 | Não |
| d\_calendario | QuinzenaMesAnoNome | string | Não |
| d\_calendario | QuinzenaPeriodo | string | Não |
| d\_calendario | QuinzenaIndice | int64 | Não |
| d\_calendario | QuinzenaOffset | int64 | Não |
| d\_calendario | QuinzenaMesAnoNomeAtual | string | Não |
| d\_calendario | SemestreDoAnoNum | int64 | Não |
| d\_calendario | SemestreAnoNum | int64 | Não |
| d\_calendario | SemestreAnoNome | string | Não |
| d\_calendario | SemestreIndice | int64 | Não |
| d\_calendario | SemestreOffset | int64 | Não |
| d\_calendario | SemestreAnoNomeAtual | string | Não |
| d\_calendario | BimestreDoAnoNum | int64 | Não |
| d\_calendario | BimestreAnoNum | int64 | Não |
| d\_calendario | BimestreAnoNome | string | Não |
| d\_calendario | BimestreIndice | int64 | Não |
| d\_calendario | BimestreOffset | int64 | Não |
| d\_calendario | BimestreAnoNomeAtual | string | Não |
| d\_calendario | FeriadoNome | string | Não |
| d\_calendario | DiaUtilNum | int64 | Não |
| d\_calendario | DiaUtilNome | string | Não |
| d\_calendario | EstacaoAnoNum | int64 | Não |
| d\_calendario | EstacaoAnoNome | string | Não |
| d\_calendario | MesFechamentoNum | int64 | Não |
| d\_calendario | MesFechamentoNome | string | Não |
| d\_calendario | MesFechamentoNomeAbrev | string | Não |
| d\_calendario | AnoFechamentoNum | int64 | Não |
| d\_calendario | MesAnoFechamentoNum | int64 | Não |
| d\_calendario | MesAnoFechamentoNome | string | Não |
| d\_calendario | DataFutura | boolean | Não |
| d\_calendario | DiaUtilDoMes | int64 | Não |
| o\_atualizacao | ultima\_atualizacao | dateTime | Não |
| o\_log\_desenvolvimento | Observacao | string | Não |
| o\_log\_desenvolvimento | Total de Minutos | double | Não |
| o\_log\_desenvolvimento | dt\_inicio | dateTime | Não |
| o\_log\_desenvolvimento | dt\_fim | dateTime | Não |
| o\_log\_desenvolvimento | saldo\_segundos | double | Não |
| d\_placas | Placa | string | Não |
| d\_status\_viagem | status\_viagem | string | Não |
| \_medidas | Column | int64 | Sim |
| Produtividade | Produtividade | string | Sim |
| Produtividade | Produtividade Campos | string | Sim |
| Produtividade | Produtividade Pedido | int64 | Sim |
| Numero Ocorrencias | Numero Ocorrencias | int64 | Sim |

Medidas

|  |  |
| --- | --- |
| **Medida** | **Expressão** |
| 1 - Soma Horas Origem | SUM(f\_viagens[saldo\_origem]) |
| 2 - Soma Horas Entrega | sum(f\_viagens[saldo\_entrega]) |
| 3 - Soma Horas Destino | sum(f\_viagens[saldo\_destino]) |
| Soma Horas | [1 - Soma Horas Origem] + [2 - Soma Horas Entrega] + [3 - Soma Horas Destino] |
| Soma Horas Formatada | VAR vTotalHoras = [Soma Horas] VAR vHoras = INT(vTotalHoras / 3600) VAR vMinutos = INT(((vTotalHoras/3600) - vHoras) \* 60) var vSegundos = ROUND((((((vTotalHoras/3600) - vHoras) \* 60) - vMinutos) \* 60), 0) RETURN vHoras \* 10000 + vMinutos \* 100 + vSegundos |
| % Produtividade | DIVIDE( [Saldo Produtividade], CALCULATE([Saldo Produtividade], ALLSELECTED()), -1 ) |
| Soma Horas Manutencao | SUMX( VALUES(f\_viagens[Placa]), [4 - Soma Horas Manutencao] ) |
| Soma Horas Manutencao Formatado | VAR vTotalHorasOriginal = [Soma Horas Manutencao] VAR vTotalHoras = IF( vTotalHorasOriginal < 0, vTotalHorasOriginal \* -1, vTotalHorasOriginal ) VAR vDias = INT(vTotalHoras/(3600\*24)) VAR vHoras = INT(((vTotalHoras/(3600\*24)) - vDias) \* 24) VAR vMinutos = INT(((((vTotalHoras/(3600\*24)) - vDias) \* 24) - vHoras) \* 60) var vSegundos = INT(((((((vTotalHoras/(3600\*24)) - vDias) \* 24) - vHoras) \* 60) - vMinutos) \* 60) RETURN IF( vTotalHorasOriginal < 0, (vDias\*1000000 + vHoras\*10000 + vMinutos\*100 + vSegundos) \* -1, vDias\*1000000 + vHoras\*10000 + vMinutos\*100 + vSegundos ) |
| Saldo Produtividade | [Soma Horas] - [Soma Horas Manutencao] |
| Saldo Produtividade Formatado | VAR vTotalHorasOriginal = [Saldo Produtividade] VAR vTotalHoras = IF( vTotalHorasOriginal < 0, vTotalHorasOriginal \* -1, vTotalHorasOriginal ) VAR vDias = INT(vTotalHoras/(3600\*24)) VAR vHoras = INT(((vTotalHoras/(3600\*24)) - vDias) \* 24) VAR vMinutos = INT(((((vTotalHoras/(3600\*24)) - vDias) \* 24) - vHoras) \* 60) var vSegundos = INT(((((((vTotalHoras/(3600\*24)) - vDias) \* 24) - vHoras) \* 60) - vMinutos) \* 60) RETURN IF( vTotalHorasOriginal < 0, (vDias\*1000000 + vHoras\*10000 + vMinutos\*100 + vSegundos) \* -1, vDias\*1000000 + vHoras\*10000 + vMinutos\*100 + vSegundos ) |
| 4 - Soma Horas Manutencao | SUM(f\_manutencoes[saldo\_manutencao]) |
| Produtividade Cor | IF( [Saldo Produtividade] > 0, "#00CDAC", "#B11623" ) |
| Ranking Problemas | VAR vTopN = 'Numero Ocorrencias'[Valor Numero Ocorrencias] VAR vMaioresOcorrencias = WINDOW( 0, ABS, vTopN, ABS, ALLSELECTED( f\_ocorrencias[Problema], f\_ocorrencias[Causa], f\_ocorrencias[Placa] ), ORDERBY( [Qtde Problema], DESC ) ) var vResult = CALCULATE( [Qtde Problema], KEEPFILTERS(vMaioresOcorrencias) ) RETURN vResult |
| Qtde Problema | COUNTROWS(f\_ocorrencias) |
| Top 10 Produtividade | var vMaisProdutivos = WINDOW( 0, ABS, 10, ABS, // Se colocar mais campos na tabela, deve-se escolher a tabela FATO por conter mais campos relacionados // Aqui optou-se por colocar a d\_placas por ser apenas um campo (sem ser medida) para mostrar na tela (visual de tabela) ALLSELECTED( d\_placas[Placa] ), ORDERBY( [Saldo Produtividade], DESC ) ) var vResult = CALCULATE( [Saldo Produtividade Formatado], KEEPFILTERS(vMaisProdutivos) ) RETURN vResult |
| Produtividade Planejado | VAR vNumeroDiasFiltrados = COUNTROWS( FILTER( d\_calendario, d\_calendario[DiaUtilNum] = 1 ) ) VAR vNumeroCaminhoesFiltrados = COUNTROWS( VALUES(f\_viagens[Placa]) ) VAR vHorasUteisPorDia = (TIME(18, 0, 0) - TIME(08, 00, 00)) \* 24 RETURN vNumeroCaminhoesFiltrados \* vHorasUteisPorDia \* vNumeroDiasFiltrados |
| Produtividade Realizada | INT([Soma Horas] / 3600) |
| % Aderencia | DIVIDE( [Produtividade Realizada], [Produtividade Planejado], -1 ) |
| Soma Horas Manutencao Formatada (h) | INT([Soma Horas Manutencao] / 3600) |
| Soma Horas Formatada (h) | INT([Soma Horas] / 3600) |
| Saldo Produtividade Formatado (h) | INT([Saldo Produtividade]/3600) |
| Qtde Viagens | COUNTROWS(f\_viagens) |
| Numero Viagens | VAR vViagens = SUMMARIZE( f\_viagens, f\_viagens[ID Viagem], "@Viagens", COUNT(f\_viagens[Índice]) ) VAR vResult = CALCULATE( [Qtde Viagens], vViagens ) RETURN vResult |
| Media Horas Origem | AVERAGE(f\_viagens[saldo\_origem])/3600 |
| Media Horas Entrega | AVERAGE(f\_viagens[saldo\_entrega])/3600 |
| Media Horas Destino | AVERAGE(f\_viagens[saldo\_destino]) / 3600 |
| Valor Numero Ocorrencias | SELECTEDVALUE('Numero Ocorrencias'[Numero Ocorrencias]) |

Visuais

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Página** | **X** | **Y** | **Altura** | **Largura** | **Tipo de visual** | **Medidas utilizadas** |
| Capa | 748 | 51 | 41 | 460 | textbox | Não há medidas utilizadas no visual |
| Capa | 675 | 404 | 115 | 510 | shape | Não há medidas utilizadas no visual |
| Capa | 994 | 258 | 60 | 255 | slicer | ['AnoNum', 'MesNome', 'DiaDoMesNum'] |
| Capa | 592 | 568 | 137 | 675 | textbox | Não há medidas utilizadas no visual |
| Produtividade | 232 | 115 | 97 | 318 | card | ['Produtividade Planejado'] |
| Produtividade | 232 | 247 | 227 | 642 | donutChart | ['Produtividade Planejado', 'Produtividade Realizada'] |
| Produtividade | 16 | 534 | 65 | 180 | slicer | ['Destino'] |
| Produtividade | 570 | 118 | 94 | 310 | card | ['Produtividade Realizada'] |
| Produtividade | 1595 | 205 | 1057 | 420 | barChart | ['Placa', '% Produtividade', '% Produtividade'] |
| Produtividade | 16 | 128 | 65 | 180 | slicer | ['AnoNum', 'MesNome', 'DiaDoMesNum'] |
| Produtividade | 232 | 948 | 305 | 642 | tableEx | ['Placa', 'Soma Horas Formatada (h)', 'Soma Horas Manutencao Formatada (h)', 'Saldo Produtividade Formatado (h)', 'Saldo Produtividade Formatado'] |
| Produtividade | 232 | 574 | 332 | 1328 | tableEx | ['Placa', 'Problema', 'Causa', 'Ranking Problemas'] |
| Produtividade | 16 | 435 | 65 | 180 | slicer | ['Origem'] |
| Produtividade | 918 | 247 | 220 | 642 | columnChart | ['Placa', 'Top 10 Produtividade'] |
| Produtividade | 1605 | 125 | 75 | 404 | advancedSlicerVisual | ['Produtividade'] |
| Produtividade | 16 | 732 | 65 | 180 | slicer | ['Cliente'] |
| Produtividade | 16 | 335 | 65 | 180 | slicer | ['Problema'] |
| Produtividade | 918 | 115 | 97 | 308 | card | ['% Aderencia'] |
| Produtividade | 16 | 236 | 65 | 180 | slicer | ['Placa'] |
| Produtividade | 918 | 950 | 304 | 642 | tableEx | ['ID Viagem', 'Placa', 'Numero Viagens'] |
| Produtividade | 232 | 501 | 72 | 1328 | advancedSlicerVisual | ['Numero Ocorrencias'] |
| Produtividade | 1254 | 115 | 97 | 318 | card | ['Soma Horas Manutencao Formatada (h)'] |
| Produtividade | 16 | 633 | 65 | 180 | slicer | ['status\_viagem'] |
| Log | 0 | 0 | 135 | 1280 | textbox | Não há medidas utilizadas no visual |
| Log | 0 | 135 | 475 | 1280 | tableEx | ['dt\_inicio', 'dt\_fim', 'Observacao', 'Total de Minutos)'] |

Fontes

|  |  |  |  |
| --- | --- | --- | --- |
| **Tabela** | **Modo de importação** | **Tipo de importação** | **Fonte** |
| f\_manutencoes | import | m | let  planilha = "Tabela Manutenção.xlsx",  arquivo\_xlsx = Text.Combine({dir\_xlsx, planilha}),  Fonte = Excel.Workbook(File.Contents(arquivo\_xlsx), null, true),  Planilha1\_Sheet = Fonte{[Item="Planilha1",Kind="Sheet"]}[Data],  #"Cabeçalhos Promovidos" = Table.PromoteHeaders(Planilha1\_Sheet, [PromoteAllScalars=true]),  #"Tipo Alterado" = Table.TransformColumnTypes(#"Cabeçalhos Promovidos",{{"Placa", type text}, {"Data Hora Início", type datetime}, {"Data Hora Fim 2", type datetime}}),  #"Colunas Renomeadas" = Table.RenameColumns(#"Tipo Alterado",{{"Data Hora Início", "Inicio"}, {"Data Hora Fim 2", "Fim"}}),  // lista de feriados bufferizados para otimizar o tratamento  feriados\_buffered = List.Buffer(d\_feriados\_anbima[Data]),  #"Função Personalizada Invocada" = Table.AddColumn(  #"Colunas Renomeadas",  "saldo\_manutencao", each fx\_horas\_uteis(  tim\_inicio\_expediente,  tim\_fim\_expediente,  [Inicio],  [Fim],  feriados\_buffered,  null,  null  )  ),  #"Total de Segundos Calculados" = Table.TransformColumns(#"Função Personalizada Invocada",{{"saldo\_manutencao", Duration.TotalSeconds, type number}}),  // Como apareceram registros com data e hora diferentes, houve uma distorcao nos dados, necessitando arredondar para nao atrapalhar os calculos  #"Arredondar Para Cima Inserido" = Table.AddColumn(#"Total de Segundos Calculados", "Arredondar para Cima", each Number.RoundUp([saldo\_manutencao]), Int64.Type),  #"Colunas Removidas" = Table.RemoveColumns(#"Arredondar Para Cima Inserido",{"saldo\_manutencao"}),  #"Colunas Renomeadas1" = Table.RenameColumns(#"Colunas Removidas",{{"Arredondar para Cima", "saldo\_manutencao"}}),  #"Duplicatas Removidas" = Table.Distinct(#"Colunas Renomeadas1", {"Placa", "Inicio", "Fim"}),  #"Data Inserida" = Table.AddColumn(#"Duplicatas Removidas", "Data", each DateTime.Date([Inicio]), type date),  #"Colunas Renomeadas2" = Table.RenameColumns(#"Data Inserida",{{"Data", "DataFiltro"}}) in  #"Colunas Renomeadas2" |
| f\_ocorrencias | import | m | let  planilha = "Tabela Ocorrencias.xlsx",  arquivo\_xlsx = Text.Combine( {dir\_xlsx, planilha} ),  Fonte = Excel.Workbook(File.Contents(arquivo\_xlsx), null, true),  Planilha1\_Sheet = Fonte{[Item="Planilha1",Kind="Sheet"]}[Data],  #"Cabeçalhos Promovidos" = Table.PromoteHeaders(Planilha1\_Sheet, [PromoteAllScalars=true]),  #"Tipo Alterado" = Table.TransformColumnTypes(#"Cabeçalhos Promovidos",{{"Causa", type text}, {"Data Hora Abertura", type datetime}, {"Data Hora Fechamento", type datetime}, {"Problema", type text}, {"ORDEM", Int64.Type}, {"Placa", type text}, {"Origem", type text}}),  #"Data Inserida" = Table.AddColumn(#"Tipo Alterado", "Data", each DateTime.Date([Data Hora Abertura]), type date),  #"Colunas Renomeadas" = Table.RenameColumns(#"Data Inserida",{{"Data", "DataFiltro"}}) in  #"Colunas Renomeadas" |
| f\_viagens | import | m | let  planilha = "Tabela Viagens.xlsx",  arquivo\_xlsx = Text.Combine( {dir\_xlsx, planilha} ),  Fonte = Excel.Workbook(File.Contents(arquivo\_xlsx), null, true),  Planilha1\_Sheet = Fonte{[Item="Planilha1",Kind="Sheet"]}[Data],  #"Cabeçalhos Promovidos" = Table.PromoteHeaders(Planilha1\_Sheet, [PromoteAllScalars=true]),  #"Tipo Alterado" = Table.TransformColumnTypes(#"Cabeçalhos Promovidos",{{"Placa", type text}, {"ORDEM", Int64.Type}, {"ID Viagem", type text}, {"Cadastro da viagem", type date}, {"Status monitoramento", type text}, {"Status viagem", type text}, {"Operação da viagem", type text}, {"Origem", type text}, {"Entrada na origem", type datetime}, {"Saída da origem", type datetime}, {"CPF\_CNPJ", Int64.Type}, {"Cliente", type text}, {"Entrada na entrega", type datetime}, {"Saída da entrega", type datetime}, {"Destino", type text}, {"Entrada no destino", type datetime}, {"Saída do destino", type datetime}, {"Índice", Int64.Type}}),  // corrigir dados invalidos oriundos do Excel...  #"@AddColClienteSemErro" = Table.AddColumn(#"Tipo Alterado", "ClienteSemErro", each try [Cliente] otherwise null),  #"@AddColDestinoSemErro" = Table.AddColumn(#"@AddColClienteSemErro", "DestinoSemErro", each try [Destino] otherwise null),  #"@DelColOriginais" = Table.RemoveColumns(#"@AddColDestinoSemErro", {"Cliente", "Destino"}),  #"@RenColunasCorrigidas" = Table.RenameColumns(#"@DelColOriginais", {{"ClienteSemErro", "Cliente"}, {"DestinoSemErro", "Destino"}}),  #"@TypeColunasCorrigidas" = Table.TransformColumnTypes(#"@RenColunasCorrigidas",{{"Cliente", type text}, {"Destino", type text}, {"Cadastro da viagem", type datetime}}),  // quando colocado o nome da coluna, dentro do colchete, o retorno é uma LISTA!  feriados\_buffered = List.Buffer( d\_feriados\_anbima[Data] ),  #"@AddSaldoOrigem" = Table.AddColumn(  #"@TypeColunasCorrigidas",  "saldo\_origem",  each fx\_horas\_uteis(  tim\_inicio\_expediente,  tim\_fim\_expediente,  [Entrada na origem],  [Saída da origem],  feriados\_buffered,  null,  null  )  ),  #"@AddSaldoEntrega" = Table.AddColumn(  #"@AddSaldoOrigem",  "saldo\_entrega",  each fx\_horas\_uteis(  tim\_inicio\_expediente,  tim\_fim\_expediente,  [Entrada na entrega],  [Saída da entrega],  feriados\_buffered,  null,  null  )  ),  #"@AddSaldoDestino" = Table.AddColumn(  #"@AddSaldoEntrega",  "saldo\_destino",  each fx\_horas\_uteis(  tim\_inicio\_expediente,  tim\_fim\_expediente,  [Entrada no destino],  [Saída do destino],  feriados\_buffered,  null,  null  )  ),  #"Total de Segundos Calculados" = Table.TransformColumns(  #"@AddSaldoDestino",  {  {"saldo\_origem", Duration.TotalSeconds, type number},   {"saldo\_entrega", Duration.TotalSeconds, type number},   {"saldo\_destino", Duration.TotalSeconds, type number}  }  ),  #"Tipo Alterado1" = Table.TransformColumnTypes(#"Total de Segundos Calculados",{{"Cadastro da viagem", type date}}) in  #"Tipo Alterado1" |
| d\_calendario | import | m | let   //Especifique a data inicial e final do projeto  menor\_ano\_entre\_tabelas = Date.Year(  Date.From(  List.Min( // colocar aqui a lista de tabelas a analisar (somente tabelas fato)  f\_viagens[Cadastro da viagem]  )  )  ),  maior\_data\_entre\_tabelas = Date.From(  List.Max( // colocar aqui a lista de tabelas a analisar (somente tabelas fato)  f\_viagens[Cadastro da viagem]   )  ),  dataInicial = #date ( menor\_ano\_entre\_tabelas, 1, 1 ),   dataFinal = Date.EndOfYear ( Date.From ( maior\_data\_entre\_tabelas ) ),   //Especifique a data atual. Padrão: Data atual do servidor   dataAtual = Date.From ( DateTime.LocalNow ( ) ),   //Especifique o início da semana. Padrão: Day.Monday (Segunda-feira)   inicioSemana = Day.Monday,   //Especifique o mês do início do ano fiscal. Padrão: 4 (Abril)  mesInicioAnoFiscal = 4,   /\*Dia de início do mês de fechamento.   Padrão: 16 (siginifica que o mês de fechamento vai de 16 do mês corrente   até o dia 15 do próximo mês \*/  diaInicioMesFechamento = 16,   //Especifique o idioma. Padrão: pt-BR (Português do Brasil)  idioma = "pt-BR",   //Geração da tabela  dCalendario = let  //Lista com todas as datas   datasRange = List.Buffer (   List.Transform (   { Number.From ( dataInicial ) .. Number.From ( dataFinal ) },   Date.From   )   ),   //Lista com todos os anos   anosRange = List.Buffer (   { Date.Year ( dataInicial ) .. Date.Year ( dataFinal )}   ),   //Feriados fixos que ocorrem em todos os anos   feriadosFixosCadastro = #table (   type table [   Dia = Int64.Type,   Mes = Int64.Type,   Feriado = text   ],  {  // Dia, Mês, Feriado  { 01, 01, "Confraternização Universal" }  // , { 25, 01, "Aniversário da Cidade" }   , { 21, 04, "Tiradentes" }   , { 01, 05, "Dia do Trabalhador" }   // , { 09, 07, "Revolução Constitucionalista" }   , { 07, 09, "Independência do Brasil" }   , { 12, 10, "N. Srª Aparecida" }   , { 02, 11, "Finados" }   , { 15, 11, "Proclamação da República" }   // , { 20, 11, "Consciência Negra" }   // , { 24, 12, "Véspera de Natal" }   , { 25, 12, "Natal" }   // , { 31, 12, "Véspera de Ano Novo" }   }   ),   //Função que gera os feriados fixos para todos os anos   fxGeraFeriadosFixos = ( ano ) =>   Table.AddColumn (   feriadosFixosCadastro,   "Data",   each #date ( ano, [Mes], [Dia] ),   type date   )   [ [Data], [Feriado] ],   //Gera a tabela de feriados fixos   feriadosFixos = Table.Combine (   List.Transform (   anosRange,   fxGeraFeriadosFixos   )   ),   //Função que gera os feriados móveis   fxGeraFeriadosMoveis = ( ano ) =>   let   modExcel = ( x, y ) => let m = Number.Mod ( x, y ) in if m < 0 then m + y else m,   pascoa = Date.From (   Number.Round (   Number.From ( #date ( ano, 4, 1 ) ) / 7   + modExcel ( 19 \* modExcel ( ano, 19 ) - 7, 30 ) \* 0.14,   0,   RoundingMode.Up   )   \* 7 - 6   ),   feriadosMoveis = #table (   type table [   Data = date,   Feriado = text   ],   {   { Date.AddDays(pascoa, -2 ), "Sexta-Feira Santa" }   , { pascoa , "Páscoa" }  , { Date.AddDays(pascoa, 60 ), "Corpus Christi" }  // , { Date.AddDays(pascoa, -48 ), "Segunda-feira de Carnaval" }  // , { Date.AddDays(pascoa, -47 ), "Terça-feira de Carnaval" }  // , { Date.AddDays(pascoa, -46 ), "Quarta-feira de Cinzas" }  }   )   in   feriadosMoveis,   //Gera a tabela de feriados móveis   feriadosMoveis = Table.Combine (   List.Transform (   anosRange,   fxGeraFeriadosMoveis   )   ),   //Tabela contendo todos os feriados   feriados = Table.Combine (   { feriadosFixos, feriadosMoveis }   ),   //Define a função de transformação de cada data   fxLinhaCalendario = ( data ) =>   let   //Transformações de simples passada para reutilização  zws = Character.FromNumber ( 8203 ), //Caracter invisível (zero-width spacing )  dataOffset = Number.From ( data - dataAtual ),   ano = Date.Year ( data ),   mes = Date.Month ( data ),   trimestre = Date.QuarterOfYear ( data ),   anoInicial = Date.Year ( dataInicial ),   anoAtual = Date.Year ( dataAtual ),   anoFinal = Date.Year ( dataFinal ),   anoOffset = ano - anoAtual,   mesInicial = Date.Month ( dataInicial ),   mesAtual = Date.Month ( dataAtual ),   trimestreInicial = Date.QuarterOfYear ( dataInicial ),   trimestreAtual = Date.QuarterOfYear ( dataAtual ),   diaSemana = Date.DayOfWeek ( data, inicioSemana ) + 1,   diaSemanaIndiceZws = Text.Repeat ( zws, 7 - diaSemana ),   semanaDoAnoIsoNo =   let   quintaNaSemana = Date.AddDays ( data, 3 - Date.DayOfWeek ( data, Day.Monday ) ),   inicioAnoQuintaNaSemana = #date ( Date.Year ( quintaNaSemana ), 1, 1 ),   difDias = Duration.Days ( quintaNaSemana - inicioAnoQuintaNaSemana )   in   Number.IntegerDivide ( difDias, 7, 0 ) + 1,   semanaDoAnoNum = Date.WeekOfYear ( data, inicioSemana ),   diaSemanaNome = Text.Proper ( Date.DayOfWeekName ( data, idioma ) ),   mesNome = Text.Proper ( Date.MonthName ( data, idioma ) ),   mesIndiceZws = Text.Repeat ( zws, 12 - mes ),   mesAnoIndice = 12 \* ( ano - anoInicial ) + mes,   inicioMes = Date.StartOfMonth ( data ),  fimMes = Date.EndOfMonth ( data ),  diaMes = Date.Day ( data ),   mesOffset = ( ( ano \* 12 ) - 1 + mes ) - ( ( anoAtual \* 12 ) - 1 + mesAtual ),   trimestreOffset = ( ( ano \* 4 ) - 1 + trimestre ) - ( ( anoAtual \* 4 ) - 1 + trimestreAtual ),   trimestreAno = "T" & Text.From ( trimestre ) & "/" & Text.From ( ano ),   anoIso = Date.Year ( Date.AddDays ( data, 26 - semanaDoAnoIsoNo ) ),   inicioSemanaIsoLinha = Date.StartOfWeek ( data, Day.Monday ),   fimSemanaIsoLinha = Date.EndOfWeek ( data, Day.Monday ),   inicioSemanaLinha = Date.StartOfWeek ( data, inicioSemana ),   fimSemanaLinha = Date.EndOfWeek ( data, inicioSemana ),   inicioSemanaIsoInicial = Date.StartOfWeek ( data, Day.Monday ),   inicioSemanaInicial = Date.StartOfWeek ( dataInicial, inicioSemana ),   inicioSemanaIsoAtual = Date.StartOfWeek ( dataAtual, Day.Monday ),  inicioSemanaAtual = Date.StartOfWeek ( dataAtual, inicioSemana ),   semanaIsoOffset = Number.From ( inicioSemanaIsoLinha - inicioSemanaIsoAtual ) / 7,  semanaOffset = Number.From ( inicioSemanaLinha - inicioSemanaAtual ) / 7,   semanaAnoIso = "S" & Text.PadStart ( Text.From ( semanaDoAnoIsoNo ), 2, "0" ) & "/" & Text.From ( anoIso ),   semanaAno = "S" & Text.PadStart ( Text.From ( semanaDoAnoNum ), 2, "0" ) & "/" & Text.From ( ano ),   semanaDoMes =   let   inicioMes = Date.StartOfMonth ( Date.StartOfWeek ( data,inicioSemana ) ),   primeirosSeteDias = List.Dates ( inicioMes, 7, #duration ( 1, 0, 0, 0 ) ),   primeiroDia = List.Select ( primeirosSeteDias, each Date.DayOfWeek ( \_, inicioSemana ) = 0 ) { 0 }   in   Number.RoundUp ( Duration.Days ( data-primeiroDia ) / 7 + 0.05 ),   anoSemanal = Date.Year ( inicioSemanaLinha ),   mesSemanal = Date.Month ( inicioSemanaLinha ) ,   mesSemanalNome = Text.Proper ( Date.MonthName ( inicioSemanaLinha, idioma ) ),   mesSemanalNomeAbrev = Text.Start ( mesSemanalNome, 3 ),   semanaDoMesPadraoNum = Date.WeekOfMonth ( data, inicioSemana ),   quinzenaDoMesNo = if Date.Day ( data ) <= 15 then 1 else 2,   quinzenaDoMesNoAtual = if Date.Day ( dataAtual ) <= 15 then 1 else 2,   mesAno = Text.Proper ( Date.ToText ( data, [ Format="MMM/yy", Culture =idioma ] ) ),   quinzenaMesAno = "Qui " & Text.From ( quinzenaDoMesNo ) & " - " & mesAno,   quinzenaIndice = 24 \* ( ano - anoInicial ) + 2 \* ( mes - mesInicial ) + quinzenaDoMesNo,   quinzenaIndiceAtual = 24 \* ( anoAtual - anoInicial ) + 2 \* ( mesAtual - mesInicial ) + quinzenaDoMesNoAtual,   quinzenaOffset = quinzenaIndice - quinzenaIndiceAtual,   semestreNo = if mes <= 6 then 1 else 2,   semestreNoAtual = if mesAtual <= 6 then 1 else 2,   semestreIndice = ( 2 \* ( ano - anoInicial ) ) + semestreNo,   semestreIndiceAtual = ( 2 \* ( anoAtual - anoInicial ) ) + semestreNoAtual,   semestreOffset = semestreIndice - semestreIndiceAtual,   semestreAno = "S" & Text.From ( semestreNo ) & " - " & Text.From ( ano ),   bimestreNo = Number.RoundUp ( mes / 2, 0 ),   bimestreNoAtual = Number.RoundUp ( mesAtual / 2, 0 ),   bimestreAno = "B" & Text.From ( bimestreNo ) & " - " & Text.From ( ano ),   bimestreIndice = ( 6 \* ( ano - anoInicial ) ) + bimestreNo,   bimestreIndiceAtual = ( 6 \* ( anoAtual - anoInicial ) ) + bimestreNoAtual,   bimestreOffset = bimestreIndice - bimestreIndiceAtual,   feriado = try feriados{[ Data = data ]}[Feriado] otherwise null,   diaUtilNo = if feriado <> null   or List.Contains (   { 6 .. 7 },   Date.DayOfWeek ( data, Day.Monday ) + 1   )   then 0   else 1,   mesDiaNo = Date.Month ( data ) \* 100 + Date.Day ( data ),   estacaoAnoNo =   if mesDiaNo >= 321 and mesDiaNo <= 620 then 1 else   if mesDiaNo >= 621 and mesDiaNo <= 921 then 2 else   if mesDiaNo >= 922 and mesDiaNo <= 1221 then 3 else   4 ,  dataReferenciaFechamento = if diaMes <= diaInicioMesFechamento - 1 then data else Date.AddMonths ( data, 1 )   in   //Saída das transformações   {   //DataIndice   Number.From ( data - dataInicial ) + 1,   //Data   data,   //DataOffset   dataOffset,   //DataNomeAtual   if dataOffset = 0 then "Data Atual"   else if dataOffset = -1 then "Data Anterior"   else if dataOffset = 1 then "Próxima Data"   else Date.ToText ( data, "dd/MM/yyyy" ),   //AnoNum   ano,   //AnoInicio   Date.StartOfYear ( data ),   //AnoFim   Date.EndOfYear ( data ),   //AnoIndice   ano - anoInicial + 1,   //AnoDecrescenteNome   ano,   //AnoDescrescenteNum   ano \* -1,   //AnoFiscal   if mes >= mesInicioAnoFiscal then ano else ano - 1,   //AnoOffset   anoOffset,   //AnoNomeAtual   if anoOffset = 0 then "Ano Atual"   else if anoOffset = -1 then "Ano Anterior"   else if anoOffset = 1 then "Próximo Ano"   else Date.ToText ( data, "yyyy" ),  //DiaDoMesNum   diaMes,   //DiaDoAnoNum   Date.DayOfYear ( data ),   //DiaDaSemanaNum   diaSemana,   //DiaDaSemanaNome   diaSemanaNome,   //DiaDaSemanaNomeAbrev   Text.Start ( diaSemanaNome, 3 ),   //DiaDaSemanaNomeIniciais   diaSemanaIndiceZws & Text.Start ( diaSemanaNome, 1 ),   //MesNum   mes,   //MesNome   mesNome,   //MesNomeAbrev   Text.Start ( mesNome, 3 ),   //MesNomeIniciais   mesIndiceZws & Text.Start ( mesNome, 1 ),   //MesAnoNum   ano \* 100 + mes,   //MesAnoMome   mesAno,   //MesDiaNum   mes \* 100 + diaMes,   //MesDiaNome  Text.Proper ( Date.ToText ( data, [Format="MMM/dd", Culture =idioma] )),   //MesInicio   Date.StartOfMonth ( data ),   //MesFim   Date.EndOfMonth ( data ),   //MesIndice   mesAnoIndice,   //MesOffset   mesOffset,   //MesNomeAtual   if mesOffset = 0 then "Mês Atual"   else if mesOffset = -1 then "Mês Anterior"   else if mesOffset = 1 then "Próximo Mês"   else mesNome,   //MesNomeAbrevAtual   if mesOffset = 0 then "Mês Atual"   else if mesOffset = -1 then "Mês Anterior"   else if mesOffset = 1 then "Próximo Mês"   else Text.Start ( mesNome, 3 ),   //MesAnoNomeAtual   if mesOffset = 0 then "Mês Atual"   else if mesOffset = -1 then "Mês Anterior"   else if mesOffset = 1 then "Próximo Mês"   else mesAno,   //TrimestreNum   trimestre,   //TrimestreInicio   Date.StartOfQuarter ( data ),   //TrimestreFinal   Date.EndOfQuarter ( data ),   //TrimestreAnoNum  ano \* 100 + trimestre,   //TrimestreAnoNome   trimestreAno,   //TrimestreIndice   4 \* ( ano - anoInicial ) + trimestre,   //TrimestreOffset   trimestreOffset,   //TrimestreAnoNomeAtual   if trimestreOffset = 0 then "Trimestre Atual"   else if trimestreOffset = -1 then "Trimestre Anterior"   else if trimestreOffset = 1 then "Próximo Trimestre"   else trimestreAno,   //SemanaIsoNum   semanaDoAnoIsoNo,   //AnoIsoNum   anoIso,   //SemanaIsoAnoNum   anoIso \* 100 + semanaDoAnoIsoNo,   //SemanaIsoAnoNome   semanaAnoIso,   //SemanaIsoInicio   inicioSemanaIsoLinha,   //SemanaIsoFim   fimSemanaIsoLinha,   //SemanaIsoIndice   Number.From ( inicioSemanaIsoLinha - inicioSemanaIsoInicial ) / 7 + 1,   //SemanaIsoOffset   semanaOffset,   //SemanaIsoAnoNomeAtual   if semanaIsoOffset = 0 then "Semana Atual"   else if semanaIsoOffset = -1 then "Semana Anterior"   else if semanaIsoOffset = 1 then "Próxima Semana"   else semanaAnoIso,   //SemanaNo   semanaDoAnoNum,   //SemanaAnoNum   ano \* 100 + semanaDoAnoNum,   //SemanaAnoNome   semanaAno,   //SemanaInicio   inicioSemanaLinha,   //SemanaFim   fimSemanaLinha,   //SemanaPeriodo  Date.ToText ( inicioSemanaLinha, "dd/MM/yyyy" ) & " a " & Date.ToText ( fimSemanaLinha, "dd/MM/yyyy" ),  //SemanaIndice   Number.From ( inicioSemanaLinha - inicioSemanaInicial ) / 7 + 1,   //SemanaOffset  semanaOffset,   //SemanaAnoNomeAtual   if semanaOffset = 0 then "Semana Atual"   else if semanaOffset = -1 then "Semana Anterior"   else if semanaOffset = 1 then "Próxima Semana"   else semanaAno,   //SemanaDoMesNum   semanaDoMes,   //AnoSemanalNum   anoSemanal,   //MesSemanalNum   mesSemanal,   //MesSemanalNome   mesSemanalNome,   //MesSemanalNomeAbrev   mesSemanalNomeAbrev,   //MesAnoSemanalNum   anoSemanal \* 100 + mesSemanal,   //MesAnoSemanalNome   Text.Proper ( Date.ToText ( inicioSemanaLinha, [ Format="MMM/yy", Culture=idioma ] )),   //SemanaDoMesPadraoNum  semanaDoMesPadraoNum,  //SemanaDoMesAnoPadraoNome  mesAno & " " & Text.From ( semanaDoMesPadraoNum ),  //SemanaDoMesAnoPadraoNum  ano \* 10000 + mes \* 100 + semanaDoMesPadraoNum,  //QuinzenaDoMesNum   quinzenaDoMesNo,   //QuinzenaMesNum   mes \* 10 + quinzenaDoMesNo,   //QuinzenaMesNome   "Qui " & Text.From ( quinzenaDoMesNo ) & " - " & mesNome,   //QuinzenaMesAnoNum   ano \* 10000 + mes \* 100 + quinzenaDoMesNo,   //QuinzenaMesAnoNome   quinzenaMesAno,   //QuinzenaPeriodo  let   inicioQuinzena = if quinzenaDoMesNo = 1 then inicioMes else Date.AddDays ( inicioMes, 15 ),  fimQuinzena = if quinzenaDoMesNo = 2 then fimMes else Date.AddDays ( inicioMes, 14 )  in  Date.ToText ( inicioQuinzena, "dd/MM/yyyy" ) & " a " & Date.ToText ( fimQuinzena, "dd/MM/yyyy" ),  //QuinzenaIndice  quinzenaIndice,  //QuinzenaOffset   quinzenaOffset,   //QuinzenaMesAnoNomeAtual   if quinzenaOffset = 0 then "Quinzena Atual"   else if quinzenaOffset = -1 then "Quinzena Anterior"   else if quinzenaOffset = 1 then "Próxima Quinzena"   else quinzenaMesAno,   //SemestreDoAnoNum   semestreNo,   //SemestreAnoNum   ano \* 100 + semestreNo,   //SemestreAnoNome  "S" & Text.From ( semestreNo ) & " - " & Text.From ( ano ),   //SemestreIndice   semestreIndice,   //Semestre Offset   semestreOffset,   //SemestreAnoNomeAtual   if semestreOffset = 0 then "Semestre Atual"   else if semestreOffset = -1 then "Semestre Anterior"   else if semestreOffset = 1 then "Próximo Semestre"   else semestreAno,   //BimestreDoAnoNum   bimestreNo,   //BimestreAnoNum   ano \* 100 + bimestreNo,   //BimestreAnoNome   bimestreAno,   //BimestreIndice   bimestreIndice,   //BimestrOffset   bimestreOffset,   //BimestreAnoNomeAtual   if bimestreOffset = 0 then "Bimestre Atual"   else if bimestreOffset = -1 then "Bimestre Anterior"   else if bimestreOffset = 1 then "Próximo Bimestre"   else bimestreAno,   //FeriadoNome   feriado,   //DiaUtilNum   diaUtilNo,   //DiaUtilNome   if diaUtilNo = 0 then "Dia Não Útil" else "Dia Útil",   //EstacaoAnoNum   estacaoAnoNo,   //EstacaoAnoNome   if estacaoAnoNo = 1 then "Outono"   else if estacaoAnoNo = 2 then "Inverno"   else if estacaoAnoNo = 3 then "Primavera"   else "Verão",  //MesFechamentoNum  Date.Month ( dataReferenciaFechamento ),  //MesFechamentoNome  Text.Proper ( Date.ToText ( dataReferenciaFechamento, [ Format = "MMMM", Culture = idioma ] ) ),  //MesFechamentoNomeAbrev  Text.Proper ( Date.ToText ( dataReferenciaFechamento, [ Format = "MMM", Culture = idioma ] ) ),  //AnoFechamentoNum  Date.Year ( dataReferenciaFechamento ),  //MesAnoFechamentoNum  Date.Year ( dataReferenciaFechamento ) \* 100 + Date.Month ( dataReferenciaFechamento ),  //MesAnoFechamentoNome  Text.Proper ( Date.ToText ( dataReferenciaFechamento, [ Format = "MMM/yy", Culture = idioma ] ) ),  //DataFutura  dataOffset > 0  },   //Gera a tabela calendário   calendario = #table (   //Nomes e tipos das colunas   type table [   DataIndice = Int64.Type,   Data = date,   DataOffset = Int64.Type,   DataNomeAtual = text,   AnoNum = Int64.Type,   AnoInicio = date,   AnoFim = date,   AnoIndice = Int64.Type,   AnoDecrescenteNome = Int64.Type,   AnoDecrescenteNum = Int64.Type,   AnoFiscal = Int64.Type,   AnoOffset = Int64.Type,   AnoNomeAtual = text,   DiaDoMesNum = Int64.Type,   DiaDoAnoNum = Int64.Type,   DiaDaSemanaNum = Int64.Type,   DiaDaSemanaNome = text,   DiaDaSemanaNomeAbrev = text,   DiaDaSemanaNomeIniciais = text,  MesNum = Int64.Type,   MesNome = text,   MesNomeAbrev = text,   MesNomeIniciais = text,   MesAnoNum = Int64.Type,   MesAnoNome = text,   MesDiaNum = Int64.Type,   MesDiaNome = text,   MesInicio = date,   MesFim = date,   MesIndice = Int64.Type,   MesOffset = Int64.Type,   MesNomeAtual = text,   MesNomeAbrevAtual = text,   MesAnoNomeAtual = text,   TrimestreNum = Int64.Type,   TrimestreInicio = date,   TrimestreFim = date,   TrimestreAnoNum = Int64.Type,  TrimestreAnoNome = text,   TrimestreIndice = Int64.Type,   TrimestreOffset = Int64.Type,   TrimestreAnoNomeAtual = text,   SemanaIsoNum = Int64.Type,  AnoIsoNum = Int64.Type,   SemanaIsoAnoNum = Int64.Type,   SemanaIsoAnoNome = text,   SemanaIsoInicio = date,   SemanaIsoFim = date,   SemanaIsoIndice = Int64.Type,   SemanaIsoOffset = Int64.Type,   SemanaIsoAnoNomeAtual = text,   SemanaNum = Int64.Type,   SemanaAnoNum = Int64.Type,   SemanaAnoNome = text,   SemanaInicio = date,   SemanaFim = date,   SemanaPeriodo = text,  SemanaIndice = Int64.Type,   SemanaOffset = Int64.Type,   SemanaAnoNomeAtual = text,   SemanaDoMesNum = Int64.Type,   AnoSemanalNum = Int64.Type,   MesSemanalNum = Int64.Type,   MesSemanalNome = text,   MesSemanalNomeAbrev = text,   MesAnoSemanalNum = Int64.Type,   MesAnoSemanalNome = text,   SemanaDoMesPadraoNum = Int64.Type,  SemanaDoMesAnoPadraoNome = text,   SemanaDoMesAnoPadraoNum = Int64.Type,   QuinzenaDoMesNum = Int64.Type,  QuinzenaMesNum = Int64.Type,   QuinzenaMesNome = text,   QuinzenaMesAnoNum = Int64.Type,   QuinzenaMesAnoNome = text,   QuinzenaPeriodo = text,  QuinzenaIndice = Int64.Type,   QuinzenaOffset = Int64.Type,   QuinzenaMesAnoNomeAtual = text,   SemestreDoAnoNum = Int64.Type,   SemestreAnoNum = Int64.Type,   SemestreAnoNome = text,   SemestreIndice = Int64.Type,   SemestreOffset = Int64.Type,   SemestreAnoNomeAtual = text,  BimestreDoAnoNum = Int64.Type,   BimestreAnoNum = Int64.Type,   BimestreAnoNome = text,   BimestreIndice = Int64.Type,   BimestreOffset = Int64.Type,   BimestreAnoNomeAtual = text,   FeriadoNome = text,   DiaUtilNum = Int64.Type,   DiaUtilNome = text,   EstacaoAnoNum = Int64.Type,   EstacaoAnoNome = text,   MesFechamentoNum = Int64.Type,  MesFechamentoNome = text,   MesFechamentoNomeAbrev = text,   AnoFechamentoNum = Int64.Type,  MesAnoFechamentoNum = Int64.Type,  MesAnoFechamentoNome = text,  DataFutura = Logical.Type   ],   //Invoca função de transformação   List.Transform ( datasRange, fxLinhaCalendario )   ),  // Adiciona o número do dia útil do mês  adDiaUtilMes =   let  tabela = Table.Buffer (   Table.SelectColumns (   calendario,   { "DiaDoMesNum", "MesIndice", "DiaUtilNum" }   )   ),   addDiaUtilMes = Table.AddColumn (  tabela,   "DiaUtilDoMes",   each let   \_\_mesIndice = [MesIndice],   \_\_DiaDoMes = [DiaDoMesNum]   in   List.Sum (   Table.SelectRows (   tabela,   each [MesIndice] = \_\_mesIndice   and [DiaDoMesNum] <= \_\_DiaDoMes   )  [DiaUtilNum]  ),   Int64.Type  )[ [MesIndice], [DiaDoMesNum], [DiaUtilDoMes] ],   join = Table.Join (  calendario, { "MesIndice", "DiaDoMesNum" },  addDiaUtilMes, { "MesIndice", "DiaDoMesNum" }  )   in  join  in  adDiaUtilMes in  dCalendario // in dataInicial |
| o\_atualizacao | import | m | let  Fonte = Table.FromValue(DateTime.FixedLocalNow()),  #"Colunas Renomeadas" = Table.RenameColumns(Fonte,{{"Value", "ultima\_atualizacao"}}),  #"Tipo Alterado" = Table.TransformColumnTypes(#"Colunas Renomeadas",{{"ultima\_atualizacao", type datetime}}) in  #"Tipo Alterado" |
| o\_log\_desenvolvimento | import | m | let  Fonte = Table.FromRows(Json.Document(Binary.Decompress(Binary.FromText("lZFfS8MwFMW/SuiTwqRpN3Hb2xTBPcjqpvgwhty2d1swyZUkdein97bdXxmIUHIvac7J757M51HajxMZpzLtRZ0o6Q2l5K9ub4bdbftIJWpYoREleVFCvRZkREYbdE8Vuq9o0fntdJAn/X2bOfwABwUQWwkwuUIbUNR7YqWCuCCrlcUO7/gAXvCvE4WlgDnRe6sIDnLQa3ANDesvW47BEYfs7yeSg32bOWVQOaqFRoVG1pV/BnEYWFyJO6daLGQXXdkGt2Rf6wl9S7iEQmkVmBBEsTvPBw2WiuuZew+MaXKEeza2NRmk5VIV2wQLskHZim+jXTYk+NUK8CAulOVI9c7kVdmSNj7ObsfxpPGIR6ECrb6hYPw2SZnESXKCVnN1h9d1naIn/Ylsw49EuUYDwvLIld2m8jCZjmZvL8/341nrlh65Ne9Su6Zt/Z/b4gc=", BinaryEncoding.Base64), Compression.Deflate)), let \_t = ((type nullable text) meta [Serialized.Text = true]) in type table [Data = \_t, Inicio = \_t, Fim = \_t, Observacao = \_t]),  #"Tipo Alterado" = Table.TransformColumnTypes(Fonte,{{"Data", type date}, {"Inicio", type time}, {"Fim", type time}}),  #"Coluna Mesclada Inserida" = Table.AddColumn(  #"Tipo Alterado",  "dt\_inicio",  each   Text.Combine(  {  Text.From([Data], "pt-BR"),   Text.From([Inicio], "pt-BR")  },   " "  ),   type datetime  ),  #"Personalização Adicionada" = Table.AddColumn(#"Coluna Mesclada Inserida", "Personalizar", each ([Fim] - [Inicio]), type time),  #"Minutos Totais Inseridos" = Table.AddColumn(#"Personalização Adicionada", "Total de Minutos", each Duration.TotalMinutes([Personalizar]), type number),  #"Colunas Removidas" = Table.RemoveColumns(#"Minutos Totais Inseridos",{"Personalizar"}),  #"Coluna Mesclada Inserida1" = Table.AddColumn(#"Colunas Removidas", "dt\_fim", each Text.Combine({Text.From([Data], "pt-BR"), Text.From([Fim], "pt-BR")}, " "), type datetime),  #"Tipo Alterado1" = Table.TransformColumnTypes(#"Coluna Mesclada Inserida1",{{"dt\_inicio", type datetime}, {"dt\_fim", type datetime}}),  // obter as horas uteis trabalhadas  #"Função Personalizada Invocada" = Table.AddColumn(  #"Tipo Alterado1",  "saldo\_segundos", each fx\_horas\_uteis(  #time(8, 0, 0),  #time(18, 0, 0),  [dt\_inicio],  [dt\_fim],  d\_feriados\_anbima[Data]  )  ),  #"Colunas Removidas1" = Table.RemoveColumns(#"Função Personalizada Invocada",{"Data", "Inicio", "Fim"}),  #"Tipo Alterado2" = Table.TransformColumnTypes(#"Colunas Removidas1",{{"saldo\_segundos", type number}}) in  #"Tipo Alterado2" |
| d\_placas | import | m | let  Fonte = Table.FromList(  List.Distinct(  List.Combine(  {  f\_viagens[Placa],  f\_manutencoes[Placa],  f\_ocorrencias[Placa]  }  )  ),  Splitter.SplitByNothing(),  type table[Placa = Text.Type]  ) in  Fonte |
| d\_status\_viagem | import | m | let  Fonte = Table.FromList(  List.Distinct(  List.Combine(  {  f\_viagens[Status viagem]  }  )  ),  Splitter.SplitByNothing(),  type table[status\_viagem = Text.Type]  ) in  Fonte |
| \_medidas | import | calculated | Row("Column", BLANK()) |
| Produtividade | import | calculated | {  ("Produtividade Percentual", NAMEOF('\_medidas'[% Produtividade]), 0),  ("Produtividade Numérica", NAMEOF('\_medidas'[Saldo Produtividade Formatado]), 1) } |
| Numero Ocorrencias | import | calculated | GENERATESERIES(1, 5, 1) |

Relacionamentos

|  |  |  |  |
| --- | --- | --- | --- |
| **Da tabela** | **Para tabela** | **Da coluna** | **Para coluna** |
| f\_ocorrencias | d\_placas | Placa | Placa |
| f\_viagens | d\_placas | Placa | Placa |
| f\_viagens | d\_status\_viagem | Status viagem | status\_viagem |
| f\_viagens | d\_calendario | Cadastro da viagem | Data |
| f\_manutencoes | d\_calendario | DataFiltro | Data |
| f\_ocorrencias | d\_calendario | DataFiltro | Data |
| f\_manutencoes | f\_viagens | Placa | Placa |
| f\_ocorrencias | f\_viagens | Placa | Placa |