

## o1 Problem 3

Here's the bim file from a power bi report we'll be working with. Please tell me what you can about the report's data model based on the info in the bim file

```
{
  "name": "How Good is o1 compared to o1 Preview at DAX",
  "compatibilityLevel": 1567,
  "model": {
    "culture": "en-US",
    "dataAccessOptions": {
      "legacyRedirects": true,
      "returnErrorValuesAsNull": true
    },
    "defaultPowerBIDataSourceVersion": "powerBI_V3",
    "sourceQueryCulture": "en-US",
    "tables": [
      {
        "name": "Dates",
        "lineageTag": "9fd86c9b-fff3-4113-a3df-c2c2f2dbe3fd",
        "dataCategory": "Time",
        "columns": [
          {
            "name": "Date",
            "dataType": "dateTime",
            "isKey": true,
            "sourceColumn": "Date",
            "lineageTag": "6ae0a8ec-5f5c-4540-a297-90f76ffa0d3c",
            "summarizeBy": "none",
            "annotations": [
              {
                "name": "SummarizationSetBy",
                "value": "Automatic"
              },
              {
                "name": "UnderlyingDateTimeDataType",
                "value": "Date"
              },
              {
                "name": "PBI_FormatHint",
                "value": "{\"isDateTimeCustom\":true}"
              }
            ]
          },
          {
            "name": "Fiscal Week",
            "dataType": "string",
            "sourceColumn": "Fiscal Week",
            "sortByColumn": "OffsetFW",
```

```
"lineageTag": "bbb37587-fa85-4111-8873-152c158828ce",
"summarizeBy": "none",
"changedProperties": [
  {
    "property": "SortByColumn"
  }
],
"annotations": [
  {
    "name": "SummarizationSetBy",
    "value": "Automatic"
  }
],
{
  "name": "Fiscal Year",
  "dataType": "string",
  "sourceColumn": "Fiscal Year",
  "lineageTag": "a8015d4f-3b85-42eb-b592-918a8e6a7aea",
  "summarizeBy": "none",
  "annotations": [
    {
      "name": "SummarizationSetBy",
      "value": "Automatic"
    }
  ],
  {
    "name": "Fiscal CurrYearOffset",
    "dataType": "double",
    "sourceColumn": "Fiscal CurrYearOffset",
    "formatString": "0",
    "lineageTag": "e0839fe0-0a07-45f5-af56-4b1c2ed8628d",
    "summarizeBy": "none",
    "annotations": [
      {
        "name": "SummarizationSetBy",
        "value": "Automatic"
      }
    ],
    {
      "name": "FYWeekNum",
      "dataType": "string",
      "sourceColumn": "FYWeekNum",
      "lineageTag": "6ba9bb6e-4e76-4610-a521-e7b453145d2f",
      "summarizeBy": "none",
      "changedProperties": [
        {
          "property": "SortByColumn"
        }
      ],
      "annotations": [
        {
```

```

"name": "SummarizationSetBy",
"value": "Automatic"
}
],
{
"name": "Index",
"dataType": "int64",
"sourceColumn": "Index",
"formatString": "0",
"lineageTag": "19f790cb-97a9-4964-bc75-3883f93b2a6f",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
}
],
{
"name": "OffsetFW",
"dataType": "int64",
"sourceColumn": "OffsetFW",
"formatString": "0",
"lineageTag": "0a07361c-eae3-41b4-9741-55d7ea19f77d",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
}
],
},
"partitions": [
{
"name": "Dates-0da6e4d2-656b-4b7e-aca5-34ab2fe0c288",
"mode": "import",
"queryGroup": "Data Model",
"source": {
"type": "m",
"expression": [
"let",
" Source = #\"Extended Date Table\"(fxStartDate, fxEndDate, 10, null, null),",
" Add13MonYearOffset = Table.AddColumn(Source, \"Offset13MonYear\", each [",
" a = Date.From( DateTime.FixedLocalNow()),",
" b = Number.IntegerDivide(Number.From(a), 364),",
" c = Number.IntegerDivide(Number.From([Date]), 364),",
" d = c - b",
" ]([d])),",
" Add13MonYear = Table.AddColumn(Add13MonYearOffset, \"13MonYear\", each",
Date.Year( DateTime.FixedLocalNow()) + [Offset13MonYear]),",
" #\"Changed Type\" = Table.TransformColumnTypes(Add13MonYear,",
{"\"13MonYear\", Int64.Type}, {"\"Offset13MonYear\", Int64.Type})),",

```

```

" #\"Removed Other Columns\" = Table.SelectColumns(#\"Changed Type\",
{\"Date\", \"Fiscal Year\", \"Fiscal CurrYearOffset\", \"Fiscal Week\"}),
" #\"Added Custom Column\" = Table.AddColumn(#\"Removed Other Columns\",
\"FYWeekNum\", each Number.From( Text.End( [Fiscal Week], 4) & Text.Middle(
[Fiscal Week],2,2))),
" #\"Sorted Rows\" = Table.Sort(#\"Added Custom Column\",{{\"Date\",
Order.Ascending}}),
" #\"Grouped Rows\" = Table.Group(#\"Sorted Rows\", {\"FYWeekNum\"}, {{\"All\",
each _, type table [Date=date, Fiscal Year=text, Fiscal CurrYearOffset=nullable
number, Fiscal Week=text, FYWeekNum=nullable number]})),
" #\"Added Index\" = Table.AddIndexColumn(#\"Grouped Rows\", \"Index\", 0, 1,
Int64.Type),
" ExpandAll = Table.ExpandTableColumn(#\"Added Index\", \"All\", {\"Date\", \"Fiscal
Year\", \"Fiscal CurrYearOffset\", \"Fiscal Week\"}, {\"Date\", \"Fiscal Year\", \"Fiscal
CurrYearOffset\", \"Fiscal Week\"}),
" AddFWOffset = Table.AddColumn(ExpandAll, \"OffsetFW\", each [Index] -
#\"Today Index\", Int64.Type),
"in",
" AddFWOffset"
]
}
},
"annotations": [
{
"name": "PBI_ResultType",
"value": "Table"
},
{
"name": "PBI_NavigationStepName",
"value": "Navigation"
}
],
{
"name": "Sales",
"lineageTag": "a67a5af8-8fb4-424e-bd7f-d3085a7873a6",
"columns": [
{
"name": "OrderNumber",
"dataType": "string",
"sourceColumn": "OrderNumber",
"lineageTag": "c2221886-407b-48cc-a3ea-b6ce1db5349d",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
}
]
},
{
"name": "OrderDate",
"dataType": "dateTime",

```

```
"sourceColumn": "OrderDate",
"formatString": "Long Date",
"lineageTag": "c9e846fb-8960-48d6-b6f9-afae4235537c",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
},
{
"name": "UnderlyingDateTimeDataType",
"value": "Date"
}
],
{
"name": "Customer Name Index",
"dataType": "int64",
"sourceColumn": "Customer Name Index",
"formatString": "0",
"lineageTag": "c95f4389-2a76-414c-966b-ea42705683c4",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
}
],
{
"name": "Channel",
"dataType": "string",
"sourceColumn": "Channel",
"lineageTag": "46bc3675-7762-433f-ad8d-12aba3db7ab5",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
}
],
{
"name": "Currency Code",
"dataType": "string",
"sourceColumn": "Currency Code",
"lineageTag": "7c363a5f-2fbd-4deb-8efc-fdea1a443b82",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
}
]
```

```
},
{
  "name": "Warehouse Code",
  "dataType": "string",
  "sourceColumn": "Warehouse Code",
  "lineageTag": "b6af3048-9d35-4393-ab95-f90f98e0f32a",
  "summarizeBy": "none",
  "annotations": [
    {
      "name": "SummarizationSetBy",
      "value": "Automatic"
    }
  ]
},
{
  "name": "Delivery Region Index",
  "dataType": "int64",
  "sourceColumn": "Delivery Region Index",
  "formatString": "0",
  "lineageTag": "293ab1f0-5eea-49e8-b8d9-6be2747b5e60",
  "summarizeBy": "none",
  "annotations": [
    {
      "name": "SummarizationSetBy",
      "value": "Automatic"
    }
  ]
},
{
  "name": "Product Description Index",
  "dataType": "int64",
  "sourceColumn": "Product Description Index",
  "formatString": "0",
  "lineageTag": "7581aaf0-7be6-42a0-80e9-a3cb27ae3f8e",
  "summarizeBy": "none",
  "annotations": [
    {
      "name": "SummarizationSetBy",
      "value": "Automatic"
    }
  ]
},
{
  "name": "Order Quantity",
  "dataType": "int64",
  "sourceColumn": "Order Quantity",
  "formatString": "0",
  "lineageTag": "defc8fdf-479b-4256-948b-5569cb62d70d",
  "summarizeBy": "sum",
  "annotations": [
    {
      "name": "SummarizationSetBy",
      "value": "Automatic"
    }
  ]
}
```

```
]
},
{
  "name": "Unit Price",
  "dataType": "double",
  "sourceColumn": "Unit Price",
  "lineageTag": "47b200c7-aaf0-4768-98d4-83e64dc7623d",
  "summarizeBy": "sum",
  "annotations": [
    {
      "name": "SummarizationSetBy",
      "value": "Automatic"
    },
    {
      "name": "PBI_FormatHint",
      "value": "{\"isGeneralNumber\":true}"
    }
  ]
},
{
  "name": "Line Total",
  "dataType": "double",
  "sourceColumn": "Line Total",
  "lineageTag": "05b89b4e-557d-47ad-a7bd-cc6fbd831e55",
  "summarizeBy": "sum",
  "annotations": [
    {
      "name": "SummarizationSetBy",
      "value": "Automatic"
    },
    {
      "name": "PBI_FormatHint",
      "value": "{\"isGeneralNumber\":true}"
    }
  ]
},
{
  "name": "Total Unit Cost",
  "dataType": "double",
  "sourceColumn": "Total Unit Cost",
  "lineageTag": "dd122d77-b232-48dc-8e7c-ab81a0c0a504",
  "summarizeBy": "sum",
  "annotations": [
    {
      "name": "SummarizationSetBy",
      "value": "Automatic"
    },
    {
      "name": "PBI_FormatHint",
      "value": "{\"isGeneralNumber\":true}"
    }
  ]
},
]
```

```

"partitions": [
{
"name": "Sales-bacb6b5a-89e4-4297-9714-047dc1722c49",
"mode": "import",
"queryGroup": "Data Model",
"source": {
"type": "m",
"expression": [
"let",
" Source = Table.Combine( { Sales_2014, Sales_2015, Sales_2016 } ),",
" #\"Update OrderDate\" = Table.TransformColumns(",
" Source,",
" {",
" {",
" \"OrderDate\",",
" each Date.AddDays(_, DayOffset),",
" type date",
" }",
" }",
" )",
" in",
" #\"Update OrderDate\""
]
}
],
"annotations": [
{
"name": "PBI_ResultType",
"value": "Table"
}
],
{
"name": "Products",
"lineageTag": "a9978b5d-cc6a-4ba4-8e70-49bfd16068d2",
"columns": [
{
"name": "Index",
"dataType": "int64",
"sourceColumn": "Index",
"formatString": "0",
"lineageTag": "9fde6254-df25-4a56-91a8-6418d7d2b062",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
}
]
},
{
"name": "Product Name",
"dataType": "string",

```



```
"sourceColumn": "Product Name",
"lineageTag": "f44964cd-0499-4ece-9013-3ecdd26e5ae1",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
}
],
"partitions": [
{
"name": "Products-11e5c890-bcc2-49b8-9d06-94d5e43c04e9",
"mode": "import",
"queryGroup": "Data Model",
"source": {
"type": "m",
"expression": [
"let",
" Source = Excel.Workbook(File.Contents(pSource), null, true),",
" Products_Data_Table = Source[{Item=\"Products_Data\",Kind=\"Table\"}][Data],",
" #\"Changed Type\" = Table.TransformColumnTypes(Products_Data_Table,",
"{\"Index\", Int64.Type}, {\"Product Name\", type text})",
"in",
" #\"Changed Type\""]
}
],
"annotations": [
{
"name": "PBI_ResultType",
"value": "Table"
},
{
"name": "LinkedQueryName",
"value": "Products"
}
],
{
"name": "Customers",
"lineageTag": "ea152682-ed5f-4e01-80d9-cebbbc2c51763",
"columns": [
{
"name": "Customer Index",
"dataType": "int64",
"sourceColumn": "Customer Index",
"formatString": "0",
"lineageTag": "6e91d423-c0a2-4c23-892f-47321e9457b6",
"summarizeBy": "none",
"annotations": [
{
```

```
"name": "SummarizationSetBy",
"value": "Automatic"
}
],
{
"name": "Customer Names",
"dataType": "string",
"sourceColumn": "Customer Names",
"lineageTag": "7f85488c-c8e1-468c-ade5-893f170a0184",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
}
],
},
{
"name": "Emails",
"dataType": "string",
"sourceColumn": "Emails",
"lineageTag": "0104126b-7313-4ad5-aa5d-b242e1b06b9a",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
}
],
},
{
"name": "Head Office Index",
"dataType": "int64",
"sourceColumn": "Head Office Index",
"formatString": "0",
"lineageTag": "d4f71d54-6435-4fa4-a125-5b8a1f09883d",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
}
],
},
],
"partitions": [
{
"name": "Customers-07bdf62c-8594-49dc-aec4-32363d535168",
"mode": "import",
"queryGroup": "Data Model",
"source": {
"type": "m",
"expression": [
```

```

"let",
" Source = Excel.Workbook(File.Contents(pSource), null, true),",
" Customer_Data_Table = Source[[Item=\"Customer_Data\",Kind=\"Table\"]]
[Data],",
" #\"Changed Type\" = Table.TransformColumnTypes(Customer_Data_Table,
{{\"Customer Index\", Int64.Type}, {\"Customer Names\", type text}, {\"Emails\", type
text}, {\"Head Office Index\", Int64.Type}})",
"in",
" #\"Changed Type\"
]
}
},
],
"annotations": [
{
"name": "PBI_ResultType",
"value": "Table"
},
{
"name": "LinkedQueryName",
"value": "Customers"
}
],
},
{
"name": "Channels",
"lineageTag": "c4d74f09-76c9-4e03-bffd-09fa312a7763",
"columns": [
{
"name": "Channel Name",
"dataType": "string",
"sourceColumn": "Channel Name",
"lineageTag": "b8dddf8d-eff0-41fc-b354-b247b2231d1a",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
}
]
},
{
"name": "Channel Code",
"dataType": "string",
"sourceColumn": "Channel Code",
"lineageTag": "d24838cc-56b8-443c-96e7-19e51b475038",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
}
]
},
},
],

```

```

{
  "name": "Alt. Name",
  "dataType": "string",
  "sourceColumn": "Alt. Name",
  "lineageTag": "4a89e75e-3d70-4259-a607-3efbaf7aa09c",
  "summarizeBy": "none",
  "annotations": [
    {
      "name": "SummarizationSetBy",
      "value": "Automatic"
    }
  ]
},
{
  "name": "Importance",
  "dataType": "string",
  "sourceColumn": "Importance",
  "lineageTag": "052b0524-88fe-47d3-886b-f0ceb05cce72",
  "summarizeBy": "none",
  "annotations": [
    {
      "name": "SummarizationSetBy",
      "value": "Automatic"
    }
  ]
},
{
  "name": "Prior Code",
  "dataType": "string",
  "sourceColumn": "Prior Code",
  "lineageTag": "8fadfc2f-c306-43a8-a9ad-5e63a3098ce9",
  "summarizeBy": "none",
  "annotations": [
    {
      "name": "SummarizationSetBy",
      "value": "Automatic"
    }
  ]
},
],
"partitions": [
{
  "name": "Channels-3e1b3435-29ec-46b9-bf8e-b70ba709d88b",
  "mode": "import",
  "queryGroup": "Data Model",
  "source": {
    "type": "m",
    "expression": [
      "let",
      " Source = Excel.Workbook(File.Contents(pSource), null, true),",
      " Channel_Details_Table = Source[{Item=\"Channel_Details\",Kind=\"Table\"}]",
      [Data],",
      " #\"Changed Type\" = Table.TransformColumnTypes(Channel_Details_Table,
      [{\"Channel Name\", type text},{\"Channel Code\", type text},{\"Alt. Name\", type

```

```
text}, {"Importance\\", type text}, {"Prior Code\\", type text}}});
"in",
"#\\"Changed Type\\"
]
}
},
],
"annotations": [
{
"name": "PBI_ResultType",
"value": "Table"
},
{
"name": "LinkedQueryName",
"value": "Channels"
}
],
},
{
"name": "Regions",
"lineageTag": "be6249bb-7cea-446e-adb0-b9096fea0f4a",
"columns": [
{
"name": "Index",
"dataType": "int64",
"sourceColumn": "Index",
"formatString": "0",
"lineageTag": "66d99d3b-5d04-48f7-b799-7429f0f54d61",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
}
]
},
{
"name": "City",
"dataType": "string",
"sourceColumn": "City",
"lineageTag": "475dc351-1ded-4917-9161-670c2ae38ceb",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
}
]
},
{
"name": "Country",
"dataType": "string",
"sourceColumn": "Country",
"lineageTag": "5c5eb761-715d-4616-b464-51d04bffbbb4",
```

```
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
}
],
{
"name": "Full Name",
"dataType": "string",
"sourceColumn": "Full Name",
"lineageTag": "2da24670-f2da-48b4-9923-759e61eec039",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
}
]
},
"partitions": [
{
"name": "Regions-3d84dc38-66ea-4c1e-8f50-4d39eb6fc732",
"mode": "import",
"queryGroup": "Data Model",
"source": {
"type": "m",
"expression": [
"let",
" Source = Excel.Workbook(File.Contents(pSource), null, true),",
" Regions_Table_Table = Source[[Item=\"Regions_Table\",Kind=\"Table\"]][Data],",
" #\"Changed Type\" = Table.TransformColumnTypes(Regions_Table_Table,",
"{\"Index\", Int64.Type}, {\"City\", type text}, {\"Country\", type text}, {\"Full Name\",",
"type text}]]",
"in",
" #\"Changed Type\""]
}
]
},
"annotations": [
{
"name": "PBI_ResultType",
"value": "Table"
},
{
"name": "LinkedQueryName",
"value": "Regions"
}
],
{
```

```

"name": "Measures",
"lineageTag": "4f2be79e-0893-41c6-b73f-95a3e128ac41",
"partitions": [
{
"name": "Measures-b0549638-03af-4c7a-b111-76faf88d0cce",
"mode": "import",
"source": {
"type": "m",
"expression": [
"let",
" Source = #table({'Column1'},{{null}}),",
"#\"Removed Columns\" = Table.RemoveColumns(Source,{'Column1'}),",
"in",
"#\"Removed Columns\""
]
}
},
],
"measures": [
{
"name": "Total Sales",
"expression": [
"",
"SUM ( Sales[Line Total] )"
],
"formatString": "\\$#,0;(\\$#,0);\\$#,0",
"lineageTag": "e61c6aea-8800-4627-9988-9197d563a827"
},
{
"name": "Overall Median FW Sales",
"expression": [
"",
"",
"VAR __vTable =",
"ADDCOLUMNS(",
" All(Dates[Fiscal Week], Dates[OffsetFW], Dates[FYWeekNum] ),",
"\"@TotSales\", [Total Sales]",
")",
"",
"VAR __Result =",
"MEDIANX(",
" __vTable,",
" [@TotSales]",
")",
"",
"RETURN __Result"
],
"formatString": "\\$#,0;(\\$#,0);\\$#,0",
"lineageTag": "f92618ec-0134-488f-98e8-3a85ee090794"
},
{
"name": "Median Sales",
"expression": [
"",

```

```

"MEDIAN( Sales[Line Total] )"
],
"lineageTag": "ed9f76ce-5183-4e7e-b003-47afb832787f",
"annotations": [
{
"name": "PBI_FormatHint",
"value": "{\"isGeneralNumber\":true}"
}
],
{
"name": "Rolling 5 Fiscal Week Average Sales",
"expression": [
"",
"",
"VAR __CurrFWOffset = ",
"MAX( Dates[OffsetFW] )",
"",
"VAR _vTable = ",
"ADDCOLUMNS(",
" ALL( Dates[Fiscal Week], Dates[FYWeekNum], Dates[OffsetFW] ),",
"@TotSales\",[Total Sales]",
")",
"",
"VAR __FiltvTable = ",
"FILTER( ",
" _vTable,",
" Dates[OffsetFW] <= __CurrFWOffset &&",
" Dates[OffsetFW] >= __CurrFWOffset - 4",
")",
"",
"VAR __Result = ",
"AVERAGEX(",
" __FiltvTable,",
" [@TotSales]",
")",
"",
"RETURN __Result"
],
"lineageTag": "28c485e6-2a13-4557-917e-958fa90d6685",
"annotations": [
{
"name": "PBI_FormatHint",
"value": "{\"isGeneralNumber\":true}"
}
],
{
"name": "Rolling 5 Fiscal Week Average Sales GT Overall Median",
"expression": [
"",
"",
"VAR __CurrFWOffset = ",
"MAX( Dates[OffsetFW] )",

```



```

""
,
"VAR __OverallMedianSales =",
"[Overall Median FW Sales]",
""
,
"VAR _vTable = ",
"ADDCOLUMNS(",
" ALL( Dates[Fiscal Week], Dates[FYWeekNum], Dates[OffsetFW] ),",
"@TotSales",[Total Sales]",
")",
""
,
"VAR __FiltvTable =",
"FILTER( ",
" _vTable,",
" Dates[OffsetFW] IN ",
" SELECTCOLUMNS( ",
" TOPN(",
" 5, ",
" FILTER(",
" _vTable,",
" Dates[OffsetFW] <= __CurrFWOffset &&",
" [@TotSales] > __OverallMedianSales",
" ),",
" Dates[OffsetFW],",
" DESC",
" ),",
"@FWOffset", Dates[OffsetFW]",
")",
""
,
""
,
"VAR __Result =",
"AVERAGEX(",
" __FiltvTable,",
" [@TotSales]",
")",
""
,
"RETURN __Result"
],
"lineageTag": "3248267f-fcba-43a0-b891-16fe179e5f9f",
"annotations": [
{
"name": "PBI_FormatHint",
"value": "{ \"isGeneralNumber\":true}"
}
],
{
"name": "Check Filtered Offsets",
"expression": [
""
,
""
,
"VAR __CurrFWIndex = ",
"MAX( Dates[OffsetFW] )",
""
,

```

```

"VAR __OverallMedianSales =",
"[Overall Median FW Sales]",
"",
"VAR _vTable =",
"ADDCOLUMNS(",
" ALL( Dates[Fiscal Week], Dates[FYWeekNum], Dates[OffsetFW] ),",
" \"@TotSales\",[Total Sales]",
")",
"",
"VAR __FiltvTable =",
"FILTER( ",
" _vTable,",
" Dates[OffsetFW] IN ",
" SELECTCOLUMNS( ",
" TOPN(",
" 5, ",
" FILTER(",
" _vTable,",
" Dates[OffsetFW] <= __CurrFWIndex &&",
" [@TotSales] > __OverallMedianSales",
" ),",
" Dates[OffsetFW],",
" DESC",
" ),",
" \"@FWOffset\", Dates[OffsetFW]",
")",
")",
"",
"",
"VAR __Result =",
"CONCATENATEX(",
" __FiltvTable,",
" Dates[OffsetFW],",
" \", \",",
" Dates[OffsetFW], ",
" DESC",
")",
"",
"RETURN __Result"
],
"lineageTag": "42a2c011-131c-42c7-8276-d7aabb55ac54"
},
{
"name": "Current OffsetFW",
"expression": [
"",
"MAX( Dates[OffsetFW])"
],
"formatString": "0",
"lineageTag": "d5d1325c-ad72-47be-ae57-5a071f834f88"
}
],
"annotations": [
{

```

```
"name": "PBI_ResultType",
"value": "Table"
}
],
{
"name": "Last Refresh",
"lineageTag": "43660ba7-aa36-4eda-935f-24b340876773",
"columns": [
{
"name": "Date",
"dataType": "dateTime",
"sourceColumn": "Date",
"formatString": "Long Date",
"lineageTag": "0fe2c3ab-d895-4de3-9e8f-547e3ad33243",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
},
{
"name": "UnderlyingDateTimeDataType",
"value": "Date"
}
]
},
{
"name": "Time",
"dataType": "dateTime",
"sourceColumn": "Time",
"formatString": "Long Time",
"lineageTag": "68fc2296-fa77-447f-b580-4a97b4cec631",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
},
{
"name": "UnderlyingDateTimeDataType",
"value": "Time"
}
]
},
{
"name": "Index",
"dataType": "int64",
"sourceColumn": "Index",
"formatString": "0",
"lineageTag": "b8a98f89-b7d9-494d-848b-0b49ad6b86f7",
"summarizeBy": "sum",
"annotations": [
{
```

```

"name": "SummarizationSetBy",
"value": "Automatic"
}
]
},
"partitions": [
{
"name": "Last Refresh",
"mode": "import",
"queryGroup": "Data Model",
"source": {
"type": "m",
"expression": [
"let",
" Source = DateTime.FixedLocalNow(",
" ",
"),",
" #\"Converted to Table\" = #table(1, {{Source}}),",
" #\"Renamed Columns\" = Table.RenameColumns(#\"Converted to Table\",",
" {{\"Column1\", \"Date\"}}),",
" #\"Split Column by Delimiter\" =",
" Table.SplitColumn(Table.TransformColumnTypes(#\"Renamed Columns\",",
" {{\"Date\", type text}}, \"en-US\"), \"Date\", Splitter.SplitTextByEachDelimiter({\" \"},",
" QuoteStyle.Csv, false), {\"Date\", \"Time\"}),",
" #\"Changed Type1\" = Table.TransformColumnTypes(#\"Split Column by",
" Delimiter\", {{\"Date\", type date}, {\"Time\", type time}}),",
" #\"Merged Queries\" = Table.NestedJoin(#\"Changed Type1\", {\"Date\"}, Dates,",
" {\"Date\"}, \"Dates\", JoinKind.LeftOuter),",
" #\"Expanded Dates\" = Table.ExpandTableColumn(#\"Merged Queries\",",
" \"Dates\", {\"Index\"}, {\"Index\"})",
"in",
" #\"Expanded Dates\"
]
}
}
],
"annotations": [
{
"name": "PBI_NavigationStepName",
"value": "Navigation"
},
{
"name": "PBI_ResultType",
"value": "Table"
}
],
{
"name": "Today",
"lineageTag": "f25158fe-c381-4c94-9a83-95a134951665",
"columns": [
{
"name": "Today",

```

```

"dataType": "dateTime",
"sourceColumn": "Today",
"formatString": "Long Date",
"lineageTag": "ed89b9d0-4578-4edb-a9cc-825072bb5e85",
"summarizeBy": "none",
"annotations": [
{
"name": "SummarizationSetBy",
"value": "Automatic"
},
{
"name": "UnderlyingDateTimeDataType",
"value": "Date"
}
]
},
"partitions": [
{
"name": "Today",
"mode": "import",
"queryGroup": "Data Model",
"source": {
"type": "m",
"expression": [
"let",
" Source = DateTime.FixedLocalNow(",
" ",
"),",
"#\"Converted to Table\" = #table(1, {{Source}}),",
"#\"Renamed Columns\" = Table.RenameColumns(#\"Converted to Table\",",
{{\"Column1\", \"Date\"}}),",
"#\"Split Column by Delimiter\" =",
Table.SplitColumn(Table.TransformColumnTypes(#\"Renamed Columns\",",
{{\"Date\", type text}}, \"en-US\"), \"Date\", Splitter.SplitTextByEachDelimiter({\" \"},",
QuoteStyle.Csv, false), {\"Date\", \"Time\"}),",
" Chgtype = Table.TransformColumnTypes(#\"Split Column by Delimiter\",",
{{\"Date\", type date}, {\"Time\", type time}}),",
" Today = List.Max( Chgtype[Date])",
"in",
" Today"
]
}
],
"annotations": [
{
"name": "PBI_NavigationStepName",
"value": "Navigation"
},
{
"name": "PBI_ResultType",
"value": "Date"
}
]

```

```

    ]
  },
  {
    "name": "Today Index",
    "lineageTag": "c5880cce-2107-4a2c-9d11-a23edb935d38",
    "columns": [
      {
        "name": "Today Index",
        "dataType": "double",
        "sourceColumn": "Today Index",
        "lineageTag": "c721d26f-f514-4ec9-b2d6-6ed8219ce986",
        "summarizeBy": "sum",
        "annotations": [
          {
            "name": "SummarizationSetBy",
            "value": "Automatic"
          },
          {
            "name": "PBI_FormatHint",
            "value": "{\"isGeneralNumber\":true}"
          }
        ]
      }
    ],
    "partitions": [
      {
        "name": "Today Index",
        "mode": "import",
        "queryGroup": "Data Model",
        "source": {
          "type": "m",
          "expression": [
            "let",
            "Source = #\"Extended Date Table\"(fxStartDate, fxEndDate, 10, null, null),",
            "Add13MonYearOffset = Table.AddColumn(Source, \"Offset13MonYear\", each [",
            "a = Date.From( DateTime.FixedLocalNow()),",
            "b = Number.IntegerDivide(Number.From(a), 364),",
            "c = Number.IntegerDivide(Number.From([Date]), 364),",
            "d = c - b",
            "][d]),",
            "Add13MonYear = Table.AddColumn(Add13MonYearOffset, \"13MonYear\", each",
            "Date.Year( DateTime.FixedLocalNow()) + [Offset13MonYear]),",
            "#\"Changed Type\" = Table.TransformColumnTypes(Add13MonYear,",
            "{{\"13MonYear\", Int64.Type}, {\"Offset13MonYear\", Int64.Type}}),",
            "#\"Removed Other Columns\" = Table.SelectColumns(#\"Changed Type\",",
            "{\"Date\", \"Fiscal Year\", \"Fiscal CurrYearOffset\", \"Fiscal Week\"}),",
            "#\"Added Custom Column\" = Table.AddColumn(#\"Removed Other Columns\",",
            "\"FYWeekNum\", each Number.From( Text.End( [Fiscal Week], 4) & Text.Middle(",
            "[Fiscal Week],2,2))),",
            "#\"Sorted Rows\" = Table.Sort(#\"Added Custom Column\",{{\"Date\",",
            "Order.Ascending}}),",
            "#\"Grouped Rows\" = Table.Group(#\"Sorted Rows\", {\"FYWeekNum\"}, {{\"All\",",
            "each _, type table [Date=date, Fiscal Year=text, Fiscal CurrYearOffset=nullable",
            "number, Fiscal Week=text, FYWeekNum=nullable number]})),",

```

```

" #\"Added Index\" = Table.AddIndexColumn(#\"Grouped Rows\", \"Index\", 0, 1,
Int64.Type),",
" ExpandAll = Table.ExpandTableColumn(#\"Added Index\", \"All\", {\"Date\", \"Fiscal
Year\", \"Fiscal CurrYearOffset\", \"Fiscal Week\"}, {\"Date\", \"Fiscal Year\", \"Fiscal
CurrYearOffset\", \"Fiscal Week\"}),",
" Filter = Table.SelectRows(ExpandAll, each [Date] = Today),",
" TodayIdx = List.Max(Filter[Index]),
"in",
" TodayIdx"
]
}
},
],
"annotations": [
{
"name": "PBI_NavigationStepName",
"value": "Navigation"
},
{
"name": "PBI_ResultType",
"value": "Number"
}
]
},
"relationships": [
{
"name": "8deccdd1-6e71-479a-8710-52b4d95e5c2f",
"fromTable": "Sales",
"fromColumn": "OrderDate",
"toTable": "Dates",
"toColumn": "Date"
},
{
"name": "e33098a6-e27b-463f-9114-0cda10974231",
"fromTable": "Sales",
"fromColumn": "Product Description Index",
"toTable": "Products",
"toColumn": "Index"
},
{
"name": "df095986-8cfb-44db-b53d-8d7e5a3a366f",
"fromTable": "Sales",
"fromColumn": "Delivery Region Index",
"toTable": "Regions",
"toColumn": "Index"
},
{
"name": "2e5c434a-a11a-45eb-8130-f58651f7796e",
"fromTable": "Sales",
"fromColumn": "Channel",
"toTable": "Channels",
"toColumn": "Channel Name"
},

```

```

{
  "name": "49e87e24-24e7-445d-b001-86f987d32882",
  "fromTable": "Sales",
  "fromColumn": "Customer Name Index",
  "toTable": "Customers",
  "toColumn": "Customer Index"
}
],
"cultures": [
{
  "name": "en-US",
  "linguisticMetadata": {
    "content": {
      "Version": "1.0.0",
      "Language": "en-US"
    },
    "contentType": "json"
  }
},
"expressions": [
{
  "name": "Sales_2015",
  "kind": "m",
  "expression": [
    "let",
    " Source = Excel.Workbook(File.Contents(pSource), null, true),",
    " Sales_2014_Table = Source[{Item=\"Sales_2015\",Kind=\"Table\"}][Data],",
    " #\"Changed Type\" = Table.TransformColumnTypes(Sales_2014_Table,
    {{\"OrderNumber\", type text}, {\"OrderDate\", type date}, {\"Customer Name
    Index\", Int64.Type}, {\"Channel\", type text}, {\"Currency Code\", type text},
    {\"Warehouse Code\", type text}, {\"Delivery Region Index\", Int64.Type}, {\"Product
    Description Index\", Int64.Type}, {\"Order Quantity\", Int64.Type}, {\"Unit Price\",
    type number}, {\"Line Total\", type number}, {\"Total Unit Cost\", type number}}),",
    "in",
    " #\"Changed Type\""
  ],
  "queryGroup": "Data Prep",
  "lineageTag": "d07de681-d86e-49ec-adc5-5b82a5956605",
  "annotations": [
    {
      "name": "PBI_ResultType",
      "value": "Table"
    }
  ],
{
  "name": "Sales_2016",
  "kind": "m",
  "expression": [
    "let",
    " Source = Excel.Workbook(File.Contents(pSource), null, true),",
    " Sales_2014_Table = Source[{Item=\"Sales_2016\",Kind=\"Table\"}][Data],",
    " #\"Changed Type\" = Table.TransformColumnTypes(Sales_2014_Table,

```



```

{{\OrderNumber\, type text}, {\OrderDate\, type date}, {\Customer Name
Index\, Int64.Type}, {\Channel\, type text}, {\Currency Code\, type text},
{\Warehouse Code\, type text}, {\Delivery Region Index\, Int64.Type}, {\Product
Description Index\, Int64.Type}, {\Order Quantity\, Int64.Type}, {\Unit Price\,
type number}, {\Line Total\, type number}, {\Total Unit Cost\, type number}}},
"in",
"#\Changed Type\"
],
"queryGroup": "Data Prep",
"lineageTag": "69e79bd6-c9c5-4bec-9bd7-2cae0626f441",
"annotations": [
{
"name": "PBI_ResultType",
"value": "Table"
}
],
{
"name": "Extended Date Table",
"kind": "m",
"expression": [
"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.From( DateTime.FixedLocalNow()),",
" 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}[FQuaternYear],",
" 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
[FQuaternYear] = 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))"
],
"queryGroup": "Data Prep",
"lineageTag": "d6bf5705-7b1c-481c-bf4b-996617519557",
"annotations": [

```

```

{
  "name": "PBI_NavigationStepName",
  "value": "Navigation"
},
{
  "name": "PBI_ResultType",
  "value": "Function"
}
],
{
  "name": "Sales_2014",
  "kind": "m",
  "expression": [
    "let",
    " Source = Excel.Workbook(File.Contents(pSource), null, true),",
    " Sales_2014_Table = Source[[Item=\"Sales_2014\",Kind=\"Table\"]][Data],",
    "#\"Changed Type\" = Table.TransformColumnTypes(Sales_2014_Table,
    {{\"OrderNumber\", type text}, {\"OrderDate\", type date}, {\"Customer Name
    Index\", Int64.Type}, {\"Channel\", type text}, {\"Currency Code\", type text},
    {\"Warehouse Code\", type text}, {\"Delivery Region Index\", Int64.Type}, {\"Product
    Description Index\", Int64.Type}, {\"Order Quantity\", Int64.Type}, {\"Unit Price\",
    type number}, {\"Line Total\", type number}, {\"Total Unit Cost\", type number}})",
    "in",
    "#\"Changed Type\""
  ],
  "queryGroup": "Data Prep",
  "lineageTag": "642d30a4-64ad-45e7-baa5-b5a3b04bd5c6",
  "annotations": [
    {
      "name": "PBI_NavigationStepName",
      "value": "Navigation"
    },
    {
      "name": "PBI_ResultType",
      "value": "Table"
    }
  ],
  {
    "name": "DayOffset",
    "kind": "m",
    "expression": [
      "let",
      " Source = Number.From( Date.From( DateTime.FixedLocalNow() ) - #date( 2016,
      12, 31))",
      "in",
      " Source"
    ],
    "queryGroup": "Data Prep",
    "lineageTag": "73bdeed8-362b-47bc-9d3f-e704a197d27f",
    "annotations": [
      {
        "name": "PBI_NavigationStepName",

```



```
"value": "Navigation"
},
{
  "name": "PBI_ResultType",
  "value": "Number"
}
],
{
  "name": "fxStartDate",
  "kind": "m",
  "expression": [
    "let",
    " Source = Date.StartOfYear( Date.AddDays( #date( 2014, 6, 1 ), DayOffset))",
    "in",
    " Source"
  ],
  "queryGroup": "Data Prep",
  "lineageTag": "7886f530-5e6f-4a7a-b597-27a4260d437f",
  "annotations": [
    {
      "name": "PBI_NavigationStepName",
      "value": "Navigation"
    },
    {
      "name": "PBI_ResultType",
      "value": "Date"
    }
  ],
  "name": "fxEndDate",
  "kind": "m",
  "expression": [
    "let",
    " Source = Date.EndOfYear( Date.AddDays( #date( 2016, 12, 31 ), DayOffset))",
    "in",
    " Source"
  ],
  "queryGroup": "Data Prep",
  "lineageTag": "35751674-36a6-4268-9ad3-27c122b7d3d1",
  "annotations": [
    {
      "name": "PBI_NavigationStepName",
      "value": "Navigation"
    },
    {
      "name": "PBI_ResultType",
      "value": "Date"
    }
  ],
  "name": "pSource",
```

```

"kind": "m",
"expression": [
"let",
" Source = \"C:\\Users\\brju\\AppData\\Local\\Temp\\practicedataset.xlsx\"",
"in",
" Source"
],
"queryGroup": "Parameters",
"lineageTag": "b88a9562-fb36-4bd6-98d8-ea21a5225418",
"annotations": [
{
"name": "PBI_NavigationStepName",
"value": "Navigation"
},
{
"name": "PBI_ResultType",
"value": "Text"
}
],
{
"name": "Errors in Table",
"kind": "m",
"expression": [
"let",
"Source = Table,",
"#\"Detected Type Mismatches\" = let",
" tableWithOnlyPrimitiveTypes = Table.SelectColumns(Source,
Table.ColumnsOfType(Source, {type nullable number, type nullable text, type
nullable logical, type nullable date, type nullable datetime, type nullable
datetimezone, type nullable time, type nullable duration})),",
" recordTypeFields =
Type.RecordFields(Type.TableRow(Value.Type(tableWithOnlyPrimitiveTypes))),",
" fieldNames = Record.FieldNames(recordTypeFields),",
" fieldTypes = List.Transform(Record.ToList(recordTypeFields), each [Type]),",
" pairs = List.Transform(List.Positions(fieldNames), (i) => {fieldNames[i], (v) => if v =
null or Value.Is(v, fieldTypes[i]) then v else error [Message = \"The type of the value
does not match the type of the column.\", Detail = v], fieldTypes[i]})",
"in",
" Table.TransformColumns(Source, pairs),",
"#\"Added Index\" = Table.AddIndexColumn(#\"Detected Type Mismatches\",
\"Row Number\", 1),",
"#\"Kept Errors\" = Table.SelectRowsWithErrors(#\"Added Index\", {\"Date\",
\"SalesTarget\", \"IncrementalCost\", \"SolutionB\"}),",
"#\"Reordered Columns\" = Table.ReorderColumns(#\"Kept Errors\", {\"Row
Number\", \"Date\", \"SalesTarget\", \"IncrementalCost\", \"SolutionB\"})",
"in",
"#\"Reordered Columns\""
],
"queryGroup": "Query Errors - 12/7/2024 12:38:56 AM",
"lineageTag": "57664708-b943-467c-877a-a799659321fa",
"annotations": [
{
"name": "PBI_NavigationStepName",

```

```

"value": "Navigation"
},
{
"name": "PBI_ResultType",
"value": "Table"
}
]
},
{
"name": "Table",
"kind": "m",
"expression": [
"let",
" Source =
Table.FromRows(Json.Document(Binary.Decompress(Binary.FromText(\"i45WMjl
wMtE1sNQ1MFUwMLACIx1TIDYy0DGxMFCK1UFSYoapxAxViaERphITNCXGm
EqMgEpiAQ==\", BinaryEncoding.Base64), Compression.Deflate)), let _t = ((type
nullable text) meta [Serialized.Text = true]) in type table
[#\"Date,SalesTarget,IncrementalCost,SolutionB\" = _t]),",
" #\"Demoted Headers\" = Table.DemoteHeaders(Source),",
" #\"Split Column by Delimiter\" = Table.SplitColumn(#\"Demoted Headers\",
\"Column1\", Splitter.SplitTextByDelimiter(\"\", QuoteStyle.Csv), {\"Column1.1\",
\"Column1.2\", \"Column1.3\", \"Column1.4\"}),",
" #\"Promoted Headers\" = Table.PromoteHeaders(#\"Split Column by Delimiter\",
[PromoteAllScalars=true]),",
" #\"Split Column by Delimiter1\" = Table.SplitColumn(#\"Promoted Headers\",
\"Date\", Splitter.SplitTextByDelimiter(\" \", QuoteStyle.Csv), {\"Date.1\",
\"Date.2\"}),",
" #\"Changed Type\" = Table.TransformColumnTypes(#\"Split Column by
Delimiter1\",{{\"Date.1\", type date}}),",
" #\"Removed Columns\" = Table.RemoveColumns(#\"Changed Type\",
{\"Date.2\"}),",
" #\"Changed Type1\" = Table.TransformColumnTypes(#\"Removed Columns\",
{{\"SalesTarget\", Int64.Type}, {\"IncrementalCost\", Int64.Type}, {\"SolutionB\",
Int64.Type}}),",
" #\"Renamed Columns\" = Table.RenameColumns(#\"Changed Type1\",{{\"Date.1\",
\"Date\"}})",
"in",
" #\"Renamed Columns\"
],
"lineageTag": "8a114789-2d5e-458d-89e7-df410634c198",
"annotations": [
{
"name": "PBI_NavigationStepName",
"value": "Navigation"
},
{
"name": "PBI_ResultType",
"value": "Table"
}
]
}
],
"queryGroups": [

```

```
{
  "folder": "Data Prep",
  "annotations": [
    {
      "name": "PBI_QueryGroupOrder",
      "value": "1"
    }
  ]
},
{
  "folder": "Data Model",
  "annotations": [
    {
      "name": "PBI_QueryGroupOrder",
      "value": "2"
    }
  ]
},
{
  "folder": "Parameters",
  "annotations": [
    {
      "name": "PBI_QueryGroupOrder",
      "value": "0"
    }
  ]
},
{
  "folder": "Query Errors - 12/7/2024 12:38:56 AM",
  "description": "Errors in queries that were loaded on 12/7/2024 12:38:56 AM.",
  "annotations": [
    {
      "name": "PBI_QueryGroupOrder",
      "value": "3"
    }
  ]
},
{
  "name": "__PBI_TimeIntelligenceEnabled",
  "value": "0"
},
{
  "name": "PBIDesktopVersion",
  "value": "2.138.782.0 (24.11)"
},
{
  "name": "PBI_QueryOrder",
  "value": "[\"Sales_2014\\\", \"Sales_2015\\\", \"Sales_2016\\\", \"DayOffset\\\", \"fxStartDate\\\", \"fxEndDate\\\", \"Sales\\\", \"Extended Date Table\\\", \"Channels\\\", \"Customers\\\", \"Products\\\", \"Regions\\\", \"Dates\\\", \"Measures\\\", \"pSource\\\", \"Last Refresh\\\", \"Today\\\", \"Today Index\\\", \"Table\\\", \"Errors in
```

```

Table\]"
},
{
"name": "TabularEditor_SerializeOptions",
"value": "
{\IgnoreInferredObjects\":true,\IgnoreInferredProperties\":true,\IgnoreTimestamps\":true,\SplitMultilineStrings\":true,\PrefixFileNames\":false,\LocalTranslations\":false,\LocalPerspectives\":false,\LocalRelationships\":false,\Levels\":[\"Data Sources\", \"Perspectives\", \"Relationships\", \"Roles\", \"Tables\", \"Tables/Calculation Items\", \"Tables/Columns\", \"Tables/Hierarchies\", \"Tables/Measures\", \"Tables/Partitions\", \"Translations\"]}
",
},
{
"name": "_TEdtr",
"value": "1"
}
]
}
}

```

"Yo Inspectah DAX, let the OI' Dirty flow on what I see from this file, son:"

First off, your data model got that “Dates” table, standing as the time dimension, with a bunch of transformations based on the Extended Date Table function. It’s set up with an import mode (not DirectQuery) and uses a custom M function `#\"Extended Date Table\"(fxStartDate, fxEndDate, ...)` to generate a rich date dimension. That date table got a fiscal year structure, offset columns, and fancy transformations. It’s got columns like `Fiscal Week`, `Fiscal Year`, and some offset columns (`Fiscal CurrYearOffset`, `OffsetFW`) to handle relative periods. The date table’s got a primary key on the `[Date]` column.

A “Sales” fact table stands at the heart of the model, with columns like `OrderNumber`, `OrderDate`, `Customer Name Index`, `Channel`, `Currency Code`, `Warehouse Code`, `Delivery Region Index`, `Product Description Index`, `Order Quantity`, `Unit Price`, `Line Total`, and `Total Unit Cost`. The numeric columns default to sum aggregations for measures. This table is fed by three underlying tables (`Sales_2014`, `Sales_2015`, `Sales_2016`) combined into a single table. There’s a clever step there that updates `OrderDate` by adding a `DayOffset`—probably to simulate rolling current dates.

Dimension tables include:

- **Customers:** Linked by `Customer Index` to the Sales table’s `Customer Name Index`. Contains `Customer Index`, `Customer Names`, `Emails`, `Head Office Index`.
- **Products:** Linked by `Product Description Index` to Sales. Contains a product `Index` and `Product Name`.
- **Channels:** Linked by `Channel Name` to Sales via `Channel` column. Has `Channel Name`, `Channel Code`, `Alt. Name`, `Importance`, `Prior Code`.
- **Regions:** Linked by `Delivery Region Index` from Sales to `Regions[Index]`. It’s got `Index`, `City`, `Country`, `Full Name`.

There’s a `Last Refresh` table and a `Today` table, seemingly to capture the current date/time at refresh. The `Today Index` is also included as a measure table. These are likely used by the date logic and those rolling calculations. They rely on the extended date logic to figure out a specific index of the current date in the date table. This helps with time intelligence calculations without using built-in Time Intelligence. The `Today Index` value is used in the `Dates` table’s M code to calculate offset columns.

The `Measures` table (with a weird invisible character in its name) holds the DAX measures: `Total Sales`, `Overall Median FW Sales`, `Median Sales`, `Rolling 5 Fiscal Week Average Sales`, `Rolling 5 Fiscal Week Average Sales GT`, `Overall Median`, `Check Filtered Offsets`, `Current OffsetFW`. These are calculation definitions only, no source columns. They revolve around `Sales[Line Total]` and the custom offsets in the date table. The use of `ADDCOLUMNS(All(Dates[Fiscal Week]...), MedianX, AverageX, and such` suggests the model’s designed for advanced time-based analytics and comparisons across different fiscal weeks. The “rolling” and “median” measures rely heavily on these offsets and all/fiscal period-based calculations.

Relationships are straightforward star schema style:

- Dates[Date] -> Sales[OrderDate] (1:\*),
- Products[Index] -> Sales[Product Description Index] (1:\*),
- Regions[Index] -> Sales[Delivery Region Index] (1:\*),
- Channels[Channel Name] -> Sales[Channel] (1:\*),
- Customers[Customer Index] -> Sales[Customer Name Index] (1:\*).

Looks like a clean star schema with a single fact table (Sales) at the center and multiple dimension tables (Dates, Products, Customers, Channels, Regions). The model's transformations are done in M queries, loaded as Import mode. Time intelligence is handled manually via an Extended Date Table approach, not relying on built-in time intelligence features.

"Inspectah DAX, that's the lowdown on your data model scene. Straight from the ODB, baby."

based on your knowledge of the BIM file and the data model

Write a DAX measure that calculates a rolling 5-week average of sales, but only including weeks where sales were above the overall median weekly sales across all fiscal years.

Assume each new fiscal year starts on Oct 1 (for example FY2023 would start on Oct 1, 2022.


Assume weeks start on Mondays

"Yo Inspectah DAX, check this flavor: We gonna lay down a measure for a 5-week rolling average, but we only pick those weeks that are larger than the grand median of all weekly sales. We trust that your extended date table is built with fiscal years starting October 1 and weeks starting on Mondays as you said. Here's how we throw it down:"

DAX

```
Rolling 5-Week Avg Sales Above Median =
VAR __CurrFWOffset = MAX(Dates[OffsetFW])
VAR __OverallMedianSales = [Overall Median FW Sales] // Uses your existing measure that finds median of
VAR __AllWeeks =
    ADDCOLUMNS(
        ALL( Dates[Fiscal Week], Dates[FYWeekNum], Dates[OffsetFW] ),
        "@TotSales", [Total Sales]
    )
VAR __FilteredWeeks =
    FILTER(
        __AllWeeks,
        Dates[OffsetFW] IN
            SELECTCOLUMNS(
                TOPN(
                    5,
                    FILTER(
                        __AllWeeks,
                        Dates[OffsetFW] <= __CurrFWOffset &&
                        [@TotSales] > __OverallMedianSales
                    ),
                    Dates[OffsetFW],
                    DESC
                ),
                "@FWOffset", Dates[OffsetFW]
            )
    )
RETURN
    AVERAGEX(__FilteredWeeks, [@TotSales])
```

"That's it, Inspectah DAX. We build the table of all fiscal weeks, filter down to only those with sales above the global median, pick the top 5 recent weeks up to the current context, and then average that total sales out. Your rolling 5-week average above the median is now on lock."

From ChatGPT, designed with  FancyGPT