EUV – logs in Litho Supplier – dashboard

[1) Introduction 3](#_Toc132628434)

[2) Actual\_TR\_Logs 4](#_Toc132628435)

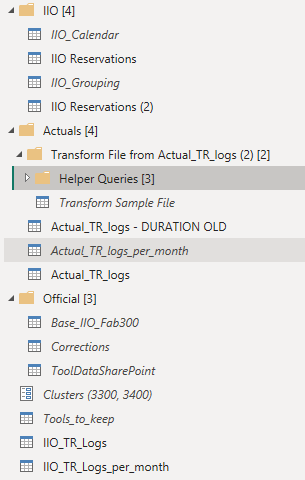
[3) IIO\_Reservations 9](#_Toc132628436)

[4) IIO\_Grouping 10](#_Toc132628437)

[5) IIO\_TR\_Logs 15](#_Toc132628438)

# Introduction

The work here happens for the EUV-tool (Extreme Ultra Violet), the most advanced family of tools at Imec nowadays.

Following are the tables present in the report...from which we’ll work on the data.

It comes down to the following. Sometimes tool-logs are used instead of on top of the database reservations (IIO Reservations, IIO stands for Imec In One, it is a reservation - software).

Actual\_TR\_Logs are logs that are coming from the tool. In this exercise it only happens for one specific tool. This is a specific and special tool that can be use in 2 ways. First of all there is the

* Track. In the track certain parameters are set have lighting on certain places or not
* Scanner can be added to that, in the same “round”. The scanner by itself doesn’t run separately. Together the track and Scanner form the LithoCluster or cluster.

Shape, rectangle

Description automatically generatedThe actual logic that needs to happen, already occurred in the previous exercise...

Here we’ve taken the Reservations also, we did combine it back then with Fab300. This is not something that has happened so far with work here.

Shape, rectangle

Description automatically generated

And here the Lithoclusters were defined and in a way taken together in the reservations. This still had nothing to do with the Actual Logs. This is the part that is happening now.

There are Actual Log files, they contain TR and SC information on Wafer-level. A Wafer is the smallest entity that goes inside the tool. A Foup has 25 Slots, available for Wafers. The Foup gets entered in the machine, and the tool gives log information back about that. We get FOUP ID, Time Start, Slots used, ...some parameters entered by the operator,...

Either how, based on this input, the tool will reproduce raw data per day. This is on wafer-level and gets all kind of information. Relevant for us is that we have a script that updates every day, a Python script running this raw data, taking information out of it. Connecting to the database to enrich it a bit...and in the end updating the files...Those files in the end, form once again input for the calculation of how the log files can be intertwined with the Reservations..From here on I will continue documenting with the existing M-Query code, and give extra comments.

# Actual\_TR\_Logs

Actual START indicated in green is below,

the part in blue is a function...Power BI writes its code this way (as does Python actually)

let

DefineBatches = (x as table) =>

let

#"Sorted Rows" = Table.Sort(x,{{"Start", Order.Ascending}}),

#"Added Index" = Table.AddIndexColumn(#"Sorted Rows", "Index", 0, 1),

* State\_DOWN\_Remove = Table.RemoveColumns(#"Added Index",{"Cluster", "Start", "KD", "Duration", "Exposure", "Index"}),
* State\_DOWN\_Rename = Table.RenameColumns(State\_DOWN\_Remove,{{"Finish", "Finish\_DOWN"}}),
* State\_DOWN\_Insert = Table.InsertRows(State\_DOWN\_Rename, 0, {[Finish\_DOWN=null]} ),

*Insert a row in the beginning of the table, based on the state of the previous step. In this “new table shape” (new dataframe), The value for Finish\_DOWN in this first row is set to “null”*

* State\_DOWN\_Add\_Index = Table.AddIndexColumn(State\_DOWN\_Insert, "Index", 0, 1),  
  *and after that an index is added to the newly created dataframe...most likely this index is used to merge with the index of the previous table...and the FINISH\_DOWN of the new table will be compared to the state “finish” of the other table that is actually from one row higher...the batches are defined in this function (of all the different wafers (each row in the source file is a wafer))*

With\_DOWN = Table.NestedJoin(#"Added Index", {"Index"}, State\_DOWN\_Add\_Index, {"Index"}, "Tmp", JoinKind.LeftOuter),

With\_DOWN\_Expanded = Table.ExpandTableColumn(With\_DOWN, "Tmp", {"Finish\_DOWN"}, {"Finish\_DOWN"}),

#"Added Custom1" = Table.AddColumn(With\_DOWN\_Expanded, "batch\_start", each if (Duration.TotalMinutes([Start] - [Finish\_DOWN]) < 10) then false else true) *--> this basically means...if “Start” – “Finish (of the previous row (but with the new table it got shifted down so it is on the same row to be able to compare it))” < 10 minutes, then this is not a new batch start (false), otherwise it is (true)*  
*--> This also means that “Start” would be minimum 10 minutes ahead, compared to the end of the wafers*

#"Replaced Errors" = Table.ReplaceErrorValues(#"Added Custom1", {{"batch\_start", true}}),

*The first line will be an error value as Finish\_DOWN is set to “null”...so there the value of the new column batch \_start can be “true”*

#"Changed Type2" = Table.TransformColumnTypes(#"Replaced Errors",{{"batch\_start", type logical}}),

* batch\_start\_filter = Table.SelectRows(#"Changed Type2", each ([batch\_start] = true)),
* useless\_clm\_remv = Table.RemoveColumns(batch\_start\_filter,{"Duration", "Exposure", "Finish\_DOWN", "batch\_start", "Cluster", "Start", "Finish", "KD"}),
* batch\_nb\_add = Table.AddIndexColumn(useless\_clm\_remv, "batch", 1, 1),

*Here there is an index called “batch” added to the temporary table (dataframe) that only contains the “start = true” rows...but there is already an index present in the table with the name “Index”*

batch\_together = Table.NestedJoin(#"Changed Type2", {"Index"}, batch\_nb\_add, {"Index"}, "batch\_starts", JoinKind.LeftOuter),

*So here the actual join with the index happens, as is it a LeftOuter join, the batch column has many openings, but every time at the start it will be filled, will be used in a filled down to come, and then this is to give an indication of the batches itself.*

#"Expanded batch\_starts" = Table.ExpandTableColumn(batch\_together, "batch\_starts", {"batch"}, {"batch"}),

remove\_tmp\_clmns = Table.RemoveColumns(#"Expanded batch\_starts",{"Index", "Finish\_DOWN", "batch\_start"}),

#"Filled Down" = Table.FillDown(remove\_tmp\_clmns,{"batch"}),

//#"Merged Queries" = Table.NestedJoin(#"Filled Down", {"KD"}, JDP\_WBS, {"KD"}, "WBS\_To\_Track", JoinKind.LeftOuter),

//#"Expanded WBS\_To\_Track" = Table.ExpandTableColumn(#"Merged Queries", "WBS\_To\_Track", {"Company"}, {"Company"}),

//#"Replaced Value2" = Table.ReplaceValue(#"Expanded WBS\_To\_Track",null,"Other",Replacer.ReplaceValue,{"Company"}),

//#"Grouped Rows1" = Table.Group(#"Replaced Value2", {"Cluster", "batch", "Company", "KD"}, {  
*Not to be done now...WBS is the cost allocation per Company, KD is a direct number to the company...there is filtering here becasuse the processing of all of this in Power BI is too slow...but with Python this is all easily manageable*

#"Grouped Rows1" = Table.Group(#"Filled Down", {"Cluster", "batch", "KD"}, {

{"Usage", each List.Sum([Duration]), type number},

{"Start", each List.Min([Start]), type datetime},

{"Finish", each List.Max([Finish]), type datetime},

{"Exposure", each List.Contains([Exposure], true), type logical}

})  
*Grouping per Cluster, Batch and KD. From each batch, the column “Usage” should be summed, “Start” the minimum start time, “Finish”, the maximum start time and “Exposure” true of there is at least one value in the group that contains true for this field.*

in

#"Grouped Rows1",

*Everything in gray below is not executed, since this is to obtain the source files. Now it will happen in a different way.*

ReadFile = (path) =>

let

Source = Csv.Document(path, [Delimiter=",", Columns=26, Encoding=1252, QuoteStyle=QuoteStyle.None]),

#"Promoted Headers" = Table.PromoteHeaders(Source, [PromoteAllScalars=true])

in

#"Promoted Headers",

min\_date = Record.Field(Date\_limits, "MinD"),

min\_year = Date.Year(min\_date),

max\_date = Record.Field(Date\_limits, "MaxD"),

max\_year = Date.Year(max\_date),

years = #table( type table

[Year = Int64.Type],

{{min\_year},

{max\_year}}),

Source = SharePoint.Files("https://imecinternational.sharepoint.com/sites/Team-ToolData", [ApiVersion = 15]),

#"Filtered Rows0" = Table.SelectRows(Source, each not Text.Contains([Name], "ICT0") and Text.Contains([Name], "2022")),

Custom1 = Table.SelectRows(#"Filtered Rows0", each Text.StartsWith([Name], "Identified\_History\_NewCount")),

//#"Filtered Rows" = Table.SelectRows(Source, each Text.Contains([Name], "Identified\_History") and (Text.Contains([Name], "2019") or Text.Contains([Name], "2020"))),

ReadFiles = Table.AddColumn(Custom1, "CSV\_Files", each ReadFile([Content])),

#"Renamed Columns1" = Table.RenameColumns(ReadFiles, {"Name", "Source.Name"}),

NeededColumns = Table.SelectColumns(#"Renamed Columns1", {"Source.Name", "CSV\_Files"}),

#"Expanded CSV\_Files" = Table.ExpandTableColumn(NeededColumns, "CSV\_Files", {"Tool Name", "Lot Name", "Job Name", "Wafer Result", "Wafer Start Time", "Wafer End Time", "Lot Start Time", "Lot End Time", "Recipe class", "Recipe", "Load Port", "FOUP", "Slot", "Exposure Start Time", "Exposure End Time", "Lot regexp", "KD regexp", "Wafer", "DT\_FOUP", "DT\_Lot", "Lot", "Route", "Lot\_Final", "KD\_Final", "Duration [h]"}, {"Tool Name", "Lot Name", "Job Name", "Wafer Result", "Wafer Start Time", "Wafer End Time", "Lot Start Time", "Lot End Time", "Recipe class", "Recipe", "Load Port", "FOUP", "Slot", "Exposure Start Time", "Exposure End Time", "Lot regexp", "KD regexp", "Wafer", "DT\_FOUP", "DT\_Lot", "Lot", "Route", "Lot\_Final", "KD\_Final", "Duration [h]"}),

#"Filtered Rows" = Table.SelectRows(#"Expanded CSV\_Files", each true),

START HERE WITH THE SOURCE FILE(S)

Source: Input are CSV-files, the

names are of this type: Identified\_History\_NewCount\_of\_TR3400\_2023.csv

#"Changed Type" = Table.TransformColumnTypes(#"Filtered Rows",{{"Tool Name", type text}, {"Lot Name", type text}, {"Job Name", type text}, {"Wafer Result", Int64.Type}, {"Wafer Start Time", type datetime}, {"Wafer End Time", type datetime}, {"Lot Start Time", type datetime}, {"Lot End Time", type datetime}, {"Recipe class", type text}, {"Recipe", type text}, {"Load Port", Int64.Type}, {"FOUP", type text}, {"Slot", Int64.Type}, {"Exposure Start Time", type datetime}, {"Exposure End Time", type datetime}, {"Lot regexp", type text}, {"KD regexp", type text}, {"Wafer", type text}, {"DT\_FOUP", type datetime}, {"DT\_Lot", type datetime}, {"Lot", type text}, {"Route", type text}, {"Lot\_Final", type text}, {"KD\_Final", type text}, {"Duration [h]", type number}}),

* *The RemoveColumns from below could be performed first*

#"Replaced Value" = Table.ReplaceValue(#"Changed Type","LITHIUS PRO Z 2G","3400",Replacer.ReplaceText,{"Tool Name"}),

#"Replaced Value1" = Table.ReplaceValue(#"Replaced Value","LITHIUS PRO Z","3300",Replacer.ReplaceText,{"Tool Name"}),

#"Removed Columns" = Table.RemoveColumns(#"Replaced Value1",{"Source.Name"}),

#"Filtered Rows1" = Table.SelectRows(#"Removed Columns", each [KD\_Final] <> null and [KD\_Final] <> ""),

#"Added Custom" = Table.AddColumn(#"Filtered Rows1", "Exposure", each if[Exposure Start Time] = null then false else true),

#"Changed Type1" = Table.TransformColumnTypes(#"Added Custom",{{"Exposure", type logical}}),

#"Removed Columns1" = Table.RemoveColumns(#"Changed Type1",{

* "Lot Name",
* "Job Name",
* "Wafer Result",
* "Lot Start Time",
* "Lot End Time",
* "Recipe class",
* "Recipe",
* "Load Port",
* "FOUP",
* "Slot",
* "Exposure Start Time",
* "Exposure End Time",
* "Lot regexp",
* "KD regexp",
* "DT\_FOUP",
* "DT\_Lot",
* "Lot",
* "Route",
* "Lot\_Final"}),

#"Renamed Columns" = Table.RenameColumns(#"Removed Columns1",{{

* "Tool Name", "Cluster"},
* {"Wafer Start Time", "Start"},
* {"Wafer End Time", "Finish"},
* {"KD\_Final", "KD"},
* {"Duration [h]", "Duration"}}),

#"Grouped Rows" = Table.Group(#"Renamed Columns", {"Cluster"}, {{"ActualRows", each DefineBatches(\_)}}),

#"Expanded ActualRows" = Table.ExpandTableColumn(#"Grouped Rows", "ActualRows", {"batch", "KD", "Usage", "Start", "Finish", "Exposure"}, {"batch", "KD", "Usage", "Start", "Finish", "Exposure"}),

#"Changed Type2" = Table.TransformColumnTypes(#"Expanded ActualRows",{{"Start", type datetime}, {"Finish", type datetime}, {"Exposure", type logical}, {"batch", Int64.Type}, {"Usage", type number}}),

#"Filtered Rows2" = Table.SelectRows(#"Changed Type2", each true)

in

#"Filtered Rows2"

# IIO\_Reservations

* This is already done in the previous exercise. It looks a bit different but basically it is the same thing...Except that for here the reservations that are kept are only the ones for the Cluster, while in the other data-pipeline all the reservations of every tool are kept.

let

**Source = Sql.Database("sqlclusp1\_fab.imec.be", "dwh\_prod"),**

**fab\_W\_STATUS\_RESERVATION\_RES = Source{[Schema="fab",Item="W\_STATUS\_RESERVATION\_RES"]}[Data],**

#"Filtered Rows" = Table.SelectRows(fab\_W\_STATUS\_RESERVATION\_RES, each [KD] <> null and [KD] <> ""),

#"Sorted Rows" = Table.Sort(#"Filtered Rows",{{"Begin\_Date", Order.Ascending}}),

Min\_Date = Record.Field(Date\_limits, "MinDT"),

Max\_Date = Record.Field(Date\_limits, "MaxDT"),

#"Filtered Rows1" = Table.SelectRows(#"Sorted Rows", each ([RES\_DATE] < Max\_Date and [Begin\_Date] > Min\_Date)),

#"Filtered Rows2" = Table.SelectRows(#"Filtered Rows1", each ([FACILITY] = "PLINE300") and ([STATUS] <> "denied")),

#"Removed Columns" = Table.RemoveColumns(#"Filtered Rows2",{"Duration", "RES\_TIME", "FROM\_HR", "UNTIL\_HR", "RES\_TYPE", "STATUS", "MODULE", "COMMENT", "RESDESCR", "CHANGED", "RES\_CATEG", "PARENT\_RES\_TK", "MULTIMOD", "RES\_TYPE\_DESCR", "POST\_ACTION\_TYPE", "POST\_ACTION\_DETAIL", "POST\_ACTION\_LOT", "RFPD", "PROCESS\_LOT", "RES\_ITEM", "REPORTING", "APPROVAL", "ITEM\_TYPE", "TOOL\_TK", "NEW\_MAT\_USED", "OUTSIDE\_BOUNDARIES", "DISCUSSED\_AT\_TAM", "OPERATOR\_JOB", "PROC\_DESCR", "Area", "Supplier", "KP"}),

#"Merged Queries" = Table.NestedJoin(#"Removed Columns", {"RES\_TOOL"}, Tools\_to\_keep, {"ToolName"}, "Tools\_to\_keep", JoinKind.Inner),

#"Expanded Tools\_to\_keep" = Table.ExpandTableColumn(#"Merged Queries", "Tools\_to\_keep", {"Clusters"}, {"Clusters"})

in

#"Expanded Tools\_to\_keep"

# IIO\_Grouping

* Needs to be further investigated how this corresponds with the function 8. merge\_adjacent and point 7. IIO\_without\_modules...here also grouping of the IIO reservations happen... #"Grouped rows" = Table.Group(#"Removed columns", {"FACILITY", "Tool", "Module", "WBS"}, {{"Rows", each merge\_adjacent(\_)

let

ClusterProcessing = (x as table) =>

let

MapToCalendar = (x as table) =>

let

#"Removed Columns1" = Table.RemoveColumns(x,{"RES\_DATE", "month", "RES\_TOOL", "GRP\_NAME", "USER", "UGRPNAME", "Clusters"}),

#"Unpivoted Columns" = Table.Unpivot(#"Removed Columns1", {"Begin\_Date", "End\_Date"}, "BE", "DateTime"),

#"Extracted Text Before Delimiter" = Table.TransformColumns(#"Unpivoted Columns", {{"BE", each Text.BeforeDelimiter(\_, "\_"), type text}}),

#"Merged Queries" = Table.NestedJoin(#"Extracted Text Before Delimiter", {"DateTime"}, IIO\_Calendar, {"DateTime"}, "IIO\_Calendar", JoinKind.RightOuter),

#"Removed Columns" = Table.RemoveColumns(#"Merged Queries",{"DateTime"}),

#"Expanded IIO\_Calendar" = Table.ExpandTableColumn(#"Removed Columns", "IIO\_Calendar", {"DateTime"}, {"DateTime"}),

#"Sorted Rows" = Table.Buffer(Table.Sort(#"Expanded IIO\_Calendar",{{"DateTime", Order.Ascending}, {"BE", Order.Descending}})),

#"Filled Down" = Table.FillDown(#"Sorted Rows",{"KD", "BE"}),

#"Added Index" = Table.AddIndexColumn(#"Filled Down", "Index", 0, 1),

State\_DOWN\_Remove = Table.RemoveColumns(#"Added Index",{"KD", "DateTime", "Index"}),

State\_DOWN\_Rename = Table.RenameColumns(State\_DOWN\_Remove,{{"BE", "BE\_DOWN"}}),

State\_DOWN\_Insert = Table.InsertRows(State\_DOWN\_Rename, 0, {[BE\_DOWN="Begin"]} ),

State\_DOWN\_Add\_Index = Table.AddIndexColumn(State\_DOWN\_Insert, "Index", 0, 1),

With\_DOWN = Table.NestedJoin(#"Added Index", {"Index"}, State\_DOWN\_Add\_Index, {"Index"}, "Tmp", JoinKind.LeftOuter),

With\_DOWN\_Expanded = Table.ExpandTableColumn(With\_DOWN, "Tmp", {"BE\_DOWN"}, {"BE\_DOWN"}),

#"RemoveHeader" = Table.SelectRows(With\_DOWN\_Expanded, each [KD] <> null and [KD] <> ""),

#"Removed Columns2" = Table.RemoveColumns(#"RemoveHeader", {"BE\_DOWN", "Index"})

in

#"Removed Columns2",

#"Grouped\_Rows1" = Table.Group(x, {"RES\_TOOL"}, {{"ToolRows", each MapToCalendar(\_)}}),

scanner\_row = Table.SelectRows(#"Grouped\_Rows1", each Text.StartsWith([RES\_TOOL], "SC")),

/\*scanner\_row = (if try (Table.SelectRows(#"Grouped Rows1", each Text.StartsWith([RES\_TOOL], "SC")))[HasError] = true

then Table.TransformColumnTypes(#"Grouped Rows1",{{"KD", type text}, {"RES\_TK", type text}, {"BE", type text}, {"Datetime", type text}})

else Table.SelectRows(#"Grouped Rows1", each Text.StartsWith([RES\_TOOL], "SC"))),\*/

scanner\_record = Table.SingleRow(scanner\_row),

scanner = if (Table.RowCount(scanner\_row) = 0)

then #table({"KD", "RES\_TK","BE", "DateTime"}, {})

else Record.Field(scanner\_record, "ToolRows"),

track\_row = Table.SelectRows(Grouped\_Rows1, each Text.StartsWith([RES\_TOOL], "TR")),

track\_record = Table.SingleRow(track\_row),

track = Record.Field(track\_record, "ToolRows"),

trsc = Table.NestedJoin(scanner, {"DateTime", "BE"}, track, {"DateTime", "BE"}, "TR", JoinKind.FullOuter),

#"Expanded TR" = Table.ExpandTableColumn(trsc, "TR", {"KD", "RES\_TK", "BE", "DateTime"}, {"TR.KD", "TR.RES\_TK", "TR.BE", "TR.DateTime"}),

#"Renamed Columns" = Table.RenameColumns(#"Expanded TR",{{"DateTime", "SC.DateTime"}}),

//#"Renamed Columns99" = Table.RenameColumns(#"Renamed Columns",{{"WBS", "SC.WBS"}}),

#"Added Custom" = Table.AddColumn(#"Renamed Columns", "DateTime", each if [SC.DateTime] = null then [TR.DateTime] else [SC.DateTime]),

//#"Added Custom99" = Table.AddColumn(#"Added Custom", "WBS", each if [SC.WBS] = null then [TR.WBS] else [SC.WBS]),

#"Removed Columns" = Table.RemoveColumns(#"Added Custom",{"SC.DateTime", "TR.DateTime"}),

#"Sorted Rows" = Table.Sort(#"Removed Columns",{{"DateTime", Order.Ascending}}),

#"Filtered Rows" = Table.SelectRows(#"Sorted Rows", each ([BE] = "Begin" or [TR.BE] = "Begin")),

#"Added Custom1" = Table.AddColumn(#"Filtered Rows", "KDs", each

if [KD] = [TR.KD] then

[KD]

else if [KD] = null then

[TR.KD]

else if [TR.KD] = null then

[KD]

else

[KD] & "," & [TR.KD]

),

#"Split Column by Delimiter" = Table.ExpandListColumn(Table.TransformColumns(#"Added Custom1", {{"KDs", Splitter.SplitTextByDelimiter(",", QuoteStyle.Csv), let itemType = (type nullable text) meta [Serialized.Text = true] in type {itemType}}}), "KDs"),

#"Added Custom2" = Table.AddColumn(#"Split Column by Delimiter", "Resource", each

if [BE] = null then

"track"

else if ([KD] <> [TR.KD]) and ([TR.KD] = [KDs]) then

"track"

else

"cluster"),

#"Removed Columns1" = Table.RemoveColumns(#"Added Custom2",{"KD", "BE", "TR.KD", "TR.BE"}),

#"Renamed Columns1" = Table.RenameColumns(#"Removed Columns1",{{"RES\_TK", "SC.RES\_TK"}}),

#"Added Custom3" = Table.AddColumn(#"Renamed Columns1", "RES\_TK", each if [Resource] = "cluster" then [SC.RES\_TK] else [TR.RES\_TK]),

#"Removed Columns2" = Table.RemoveColumns(#"Added Custom3",{"SC.RES\_TK", "TR.RES\_TK"}),

#"Filled Down" = Table.FillDown(#"Removed Columns2",{"RES\_TK"})

in

#"Filled Down",

Source = #"IIO Reservations",

#"Grouped Rows" = Table.Group(Source, {"Clusters"}, {{"IIORows", each ClusterProcessing(\_)}}),

#"Expanded IIORows" = Table.ExpandTableColumn(#"Grouped Rows", "IIORows", {"DateTime", "KDs", "RES\_TK", "Resource"}, {"DateTime", "KDs", "RES\_TK", "Resource"}),

#"Changed Type" = Table.TransformColumnTypes(#"Expanded IIORows",{{"DateTime", type datetime}, {"Resource", type text}}),

#"Merged Queries1" = Table.NestedJoin(#"Changed Type", {"KDs"}, JDP\_WBS, {"KD"}, "JDP\_WBS", JoinKind.LeftOuter),

#"Removed Columns" = Table.RemoveColumns(#"Merged Queries1",{"JDP\_WBS"}),

#"Merged Queries" = Table.NestedJoin(#"Removed Columns", {"KDs"}, WBS\_To\_Track, {"KD\_BK"}, "WBS\_To\_Track", JoinKind.LeftOuter),

#"Expanded WBS\_To\_Track" = Table.ExpandTableColumn(#"Merged Queries", "WBS\_To\_Track", {"Company"}, {"Company"}),

#"Replaced Value" = Table.ReplaceValue(#"Expanded WBS\_To\_Track",null,"Other",Replacer.ReplaceValue,{"Company"}),

#"Changed Type1" = Table.TransformColumnTypes(#"Replaced Value",{{"RES\_TK", Int64.Type}}),

#"Filtered Rows" = Table.SelectRows(#"Changed Type1", each true)

in

#"Filtered Rows"

# IIO\_TR\_Logs

let

MapActualsToIIO = (x as table) =>

let

AddedCustom = Table.AddColumn(x, "ComboDT", each if [Start] = null then [DateTime] else [Start]),

AddedCustom2 = Table.AddColumn(AddedCustom, "DateTimeBack", each [DateTime]),

AddedCustom3 = Table.AddColumn(AddedCustom2, "KD\_IIOBack", each [KD\_IIO]),

//AddedCustom99 = Table.AddColumn(AddedCustom3, "WBS\_IIOBack", each [WBS\_IIO]),

AddedCustom4 = Table.AddColumn(AddedCustom3, "Company\_IIOBack", each [Company\_IIO]),

AddedCustom5 = Table.AddColumn(AddedCustom4, "ResourceBack", each [Resource]),

AddedCustom6 = Table.AddColumn(AddedCustom5, "RES\_TKBack", each [RES\_TK]),

#"Renamed Columns" = Table.RenameColumns(AddedCustom6,{{"DateTime", "DateTimeFwd"}, {"KD\_IIO", "KD\_IIOFwd"}, {"Company\_IIO", "Company\_IIOFwd"}, {"Resource", "ResourceFwd"}, {"RES\_TK", "RES\_TKFwd"}}),

ChangedType = Table.TransformColumnTypes(#"Renamed Columns",{{"ComboDT", type datetime}}),

SortedRows = Table.Sort(ChangedType,{{"ComboDT", Order.Ascending}}),

FilledDown = Table.FillDown(SortedRows,{"DateTimeFwd", "KD\_IIOFwd", "Company\_IIOFwd", "ResourceFwd", "RES\_TKFwd"}),

FilledUp = Table.FillUp(FilledDown,{"DateTimeBack", "KD\_IIOBack", "Company\_IIOBack", "ResourceBack", "RES\_TKBack"}),

FilteredRows = Table.SelectRows(FilledUp, each ([KD] <> null)),

ExtraFwd = Table.AddColumn(FilteredRows, "ExtraFwd", each if ([Company] = [Company\_IIOFwd] or [KD\_IIOFwd] = "31852") and Duration.TotalMinutes([Start] - [DateTimeFwd]) <= 30 then true else false),

ExtraBack = Table.AddColumn(ExtraFwd, "ExtraBack", each if ([Company] = [Company\_IIOBack] or [KD\_IIOBack] = "31852") and Duration.TotalMinutes([DateTimeBack] - [Start]) <= 30 then true else false),

AddedCustom7 = Table.AddColumn(ExtraBack, "DateTime", each if [ExtraBack] then [DateTimeBack] else [DateTimeFwd]),

//AddedCustom100 = Table.AddColumn(AddedCustom7, "WBS\_IIO", each if [ExtraBack] then [WBS\_IIOBack] else [WBS\_IIOFwd]),

AddedCustom8 = Table.AddColumn(AddedCustom7, "KD\_IIO", each if [ExtraBack] then [KD\_IIOBack] else [KD\_IIOFwd]),

AddedCustom9 = Table.AddColumn(AddedCustom8, "Company\_IIO", each if [ExtraBack] then [Company\_IIOBack] else [Company\_IIOFwd]),

AddedCustom10 = Table.AddColumn(AddedCustom9, "Resource", each if [ExtraBack] then [ResourceBack] else [ResourceFwd]),

AddedCustom11 = Table.AddColumn(AddedCustom10, "RES\_TK", each if [ExtraBack] then [RES\_TKBack] else [RES\_TKFwd]),

Extra = Table.AddColumn(AddedCustom11, "Extra", each [ExtraFwd] or [ExtraBack])

in

Extra,

Source = IIO\_Grouping,

#"Renamed Columns" = Table.RenameColumns(Source,{{"KDs", "KD\_IIO"}, {"Clusters", "Cluster"}, {"Company", "Company\_IIO"}}),

with\_cluster = Table.SelectRows(#"Renamed Columns", each [Resource] = "cluster"),

with\_track = Table.SelectRows(#"Renamed Columns", each [Resource] = "track"),

wc\_actuals = Table.Combine({with\_cluster, #"Actual\_TR\_logs - DURATION OLD"}),

#"Grouped Rows" = Table.Group(wc\_actuals, {"Cluster"}, {{"ActualRows", each MapActualsToIIO(\_)}}),

#"Expanded ActualRows" = Table.ExpandTableColumn(#"Grouped Rows", "ActualRows", {"DateTime", "KD\_IIO", "RES\_TK", "Resource", "Company\_IIO", "batch", "Company", "KD", "Usage", "Start", "Finish", "Exposure", "ComboDT", "Extra"}, {"DateTime", "KD\_IIO", "RES\_TK", "Resource", "Company\_IIO", "batch", "Company", "KD", "Usage", "Start", "Finish", "Exposure", "ComboDT", "Extra"}),

wc\_missed = Table.SelectRows(#"Expanded ActualRows", each [Extra] = false),

wc\_missed\_no\_extra = Table.SelectColumns(wc\_missed, Table.ColumnNames(#"Actual\_TR\_logs - DURATION OLD")),

wt\_actuals = Table.Combine({with\_track, wc\_missed\_no\_extra}),

#"Grouped Rows1" = Table.Group(wt\_actuals, {"Cluster"}, {{"ActualRows", each MapActualsToIIO(\_)}}),

#"Expanded ActualRows1" = Table.ExpandTableColumn(#"Grouped Rows1", "ActualRows", {"DateTime", "KD\_IIO", "RES\_TK", "Resource", "Company\_IIO", "batch", "Company", "KD", "Usage", "Start", "Finish", "Exposure", "ComboDT", "Extra"}, {"DateTime", "KD\_IIO", "RES\_TK", "Resource", "Company\_IIO", "batch", "Company", "KD", "Usage", "Start", "Finish", "Exposure", "ComboDT", "Extra"}),

wtc = Table.Combine({#"Expanded ActualRows", #"Expanded ActualRows1"}),

#"Sorted Rows" = Table.Buffer(Table.Sort(wtc,{{"Cluster", Order.Ascending}, {"batch", Order.Ascending}, {"Start", Order.Ascending}, {"Extra", Order.Descending}})),

#"Removed Duplicates" = Table.Distinct(#"Sorted Rows", {"Cluster", "Start", "Finish"}),

#"Changed Type" = Table.TransformColumnTypes(#"Removed Duplicates",{{"RES\_TK", Int64.Type}, {"Exposure", type logical}, {"Extra", type logical}, {"ComboDT", type datetime}, {"Finish", type datetime}, {"Start", type datetime}, {"DateTime", type datetime}}),

Mapped = Table.SelectRows(#"Changed Type", each ([Extra] = true)),

Mapped\_MQ = Table.NestedJoin(Mapped, {"RES\_TK"}, #"IIO Reservations", {"RES\_TK"}, "300mm\_IIO", JoinKind.Inner),

#"Expanded 300mm\_IIO" = Table.ExpandTableColumn(Mapped\_MQ, "300mm\_IIO", {"Begin\_Date"}, {"Begin\_Date"}),

Mapped\_Grp = Table.Group(#"Expanded 300mm\_IIO", {"Cluster", "RES\_TK", "Company"}, {

{"Usage", each Duration.TotalHours(List.Max([Finish]) - List.Min({List.Min([Start]), List.Min([Begin\_Date])})), type number},

{"IIO\_Begin", each List.Min([Begin\_Date]), type datetime},

{"KD", each List.Min([KD\_IIO]), type datetime},

{"Exposure", each List.Contains([Exposure], true) or List.Contains([Resource], "cluster"), type logical}

}),

Mapped\_RC = Table.RemoveColumns(Mapped\_Grp,{"RES\_TK"}),

Mapped\_RnmC = Table.RenameColumns(Mapped\_RC,{{"IIO\_Begin", "Start"}}),

Mapped\_CT = Table.TransformColumnTypes(Mapped\_RnmC,{{"KD", type text}, {"Company", type text}}),

Lost = Table.SelectRows(#"Changed Type", each ([Extra] = false)),

Lost\_RC = Table.RemoveColumns(Lost,{"DateTime", "KD\_IIO", "RES\_TK", "Resource", "Company\_IIO", "batch", "Finish", "ComboDT", "Extra"}),

Lost\_CT = Table.TransformColumnTypes(Lost\_RC,{{"Usage", type number}, {"Company", type text}}),

Mapped\_n\_Lost = Table.Combine({Mapped\_CT, Lost\_CT}),

#"Sorted Rows1" = Table.Sort(Mapped\_n\_Lost,{{"Start", Order.Ascending}}),

#"Added Custom" = Table.AddColumn(#"Sorted Rows1", "Date", each Date.From([Start])),

#"Changed Type1" = Table.TransformColumnTypes(#"Added Custom",{{"Date", type date}})

in

#"Changed Type1"

If the KpKD is changed onlky for the child