Digital Egypt Bioneers Initiatives

Manufacturing Downtime Analysis Project Documentation

Power BI

Info Verse Team

Digital Egypt Bioneers Initiatives

Power BI Specialist

Manufacturing Downtime

Project Documentation

-InfoVerse Team-

AlHussein AlAttar *Mohammed AlMukadam*

Youssif Ali *Youssef Ezzat*

Mina Ayad

Table of Contents

Executive Summary

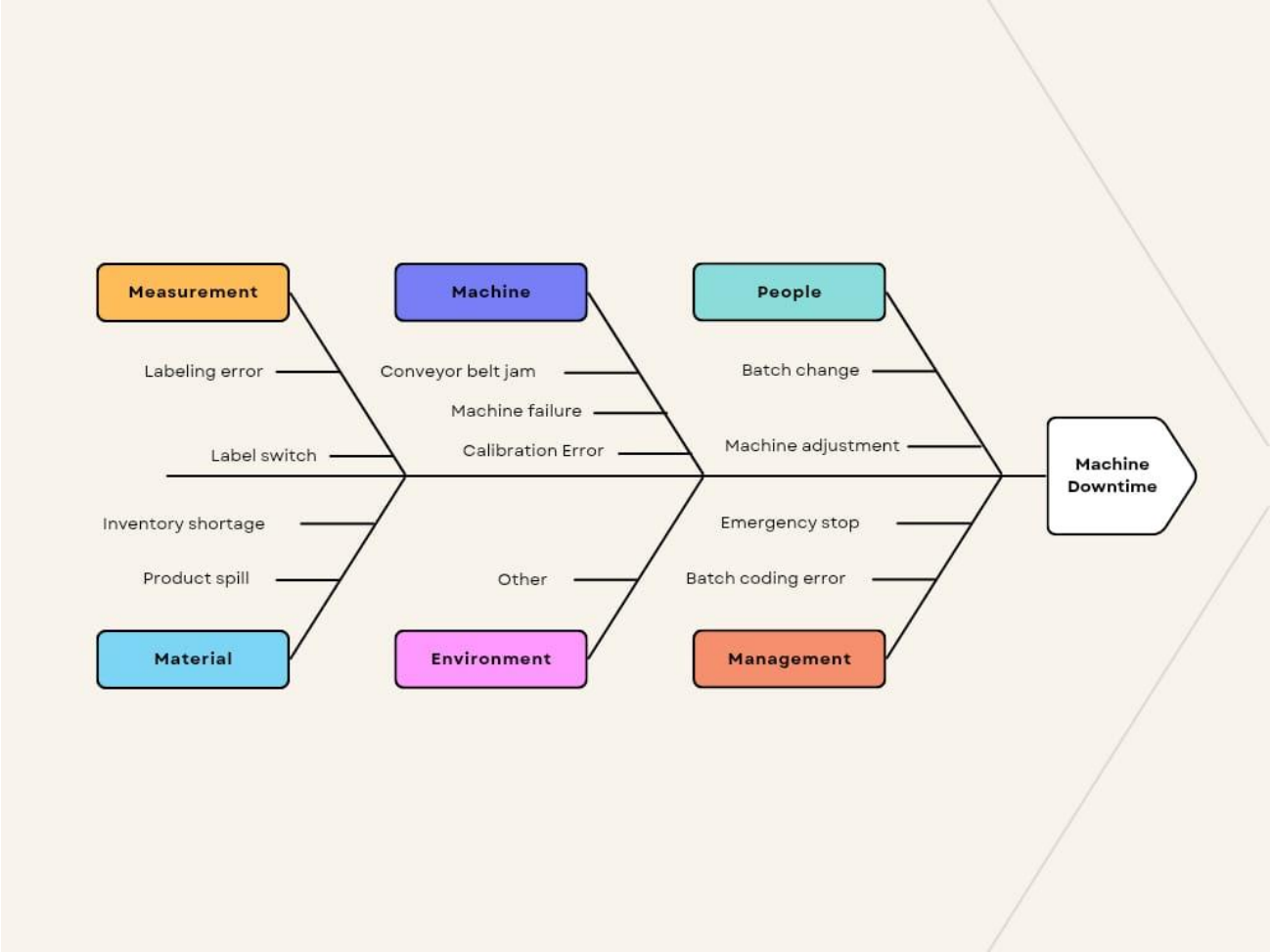
This Power BI project analysed production line efficiency, downtime patterns, and operator performance using detailed manufacturing data across batches, products, and work shifts. The primary goal was to measure overall productivity, identify the key factors contributing to downtime, and uncover opportunities to improve operational performance and workforce effectiveness. The analysis revealed that downtime was mainly driven by mechanical issues and operator-related errors, while certain product lines experienced longer batch cycles and inconsistent efficiency levels. Performance differences were also observed across work shifts, indicating potential for scheduling adjustments and training improvements. These findings highlight clear opportunities to reduce downtime and enhance equipment utilization. By implementing targeted process improvements, preventive maintenance, and operator development initiatives, the organization can improve production efficiency by an estimated **5–10%**, achieving more consistent and reliable manufacturing performance.

Objectives

1. **Increase production efficiency** by minimizing downtime and optimizing batch operations.
2. **Identify key downtime causes** to support preventive maintenance and continuous improvement.
3. **Evaluate operator performance** to guide targeted training and resource planning.
4. **Enhance visibility** of production performance across products, shifts, and time periods.
5. **Provide actionable dashboards** that empower management to make data-driven operational decisions.

Downtime Analysis

Fishbone



Problem Analysis: Machine Downtime

The goal of this analysis is to identify, categorize, and visualize the potential causes of **machine downtime** across production lines. By mapping downtime drivers, we can target areas for process improvement, preventive maintenance, and operator training.

Methodology

A **Cause-and-Effect (Fishbone/Ishikawa) Diagram** was developed to organize the major contributing factors leading to machine downtime. The analysis is divided into six key categories, each representing a distinct source of operational disruption.

Root Cause Categories

1. **Measurement**
    - Labeling error
    - Label switch These issues arise from incorrect calibration or improper labeling setups, often leading to product rework or batch delays.
2. **Machine**
    - Conveyor belt jam
    - Machine failure
    - Calibration error These mechanical and technical issues directly impact production flow, often requiring maintenance intervention.
3. **People**
    - Batch change
    - Machine adjustment
    - Emergency stop
    - Batch coding error Human factors such as operational errors, manual interventions, and incorrect machine handling cause unplanned stoppages.
4. **Material**
    - Inventory shortage
    - Product spill These reflect raw material handling and supply chain inconsistencies that interrupt production schedules.
5. **Environment**
    - Other (general or situational factors) Covers external or unpredictable environmental influences affecting machinery or workflow (e.g., humidity, temperature).
6. **Management**
    - Batch coding error
    - Emergency procedures Issues tied to planning, coordination, or policy decisions that affect operational continuity.

Key Insight

Machine downtime is a **multi-factorial issue** involving technical, human, and organizational dimensions.
Understanding these categories enables targeted interventions such as:

- **Preventive maintenance schedules** (Machine)
- **Operator retraining and SOP updates** (People)
- **Improved material logistics** (Material)
- **Enhanced process monitoring** (Measurement & Management)

Integration with Data Model

This analysis aligns with the **dimDowntimeFactors** table in the data model, which stores the downtime cause, description, and operator error flag. By linking downtime causes to measurable data (from factLineDowntime), analysts can:

- Quantify downtime per category
- Track trends over time
- Identify the most frequent root causes by operator, shift, or machine

Questions

Page 1: Overview Analysis

Objective: Understand overall production performance, downtime impact, and key efficiency drivers.

Questions:

1. **How efficiently is the production line operating overall?**
2. **What portion of total production time is lost to downtime?**
3. **Which products or factors contribute most to downtime?**
4. **How does downtime trend over time or by hour of the day?**
5. **Are there patterns indicating when or where production is most efficient?**

Page 2: Batch Analysis

Objective: Evaluate batch-level productivity and identify variations or inefficiencies affecting production consistency.

Questions:

1. **Are batches being completed within expected target times?**
2. **Which batches take the longest or shortest time to produce, and why?**
3. **What downtime causes appear most frequently across batches?**
4. **How strong is the relationship between total batch time and downtime?**
5. **Which products or operators are linked to slower batch performance?**

Page 3: Operator Analysis

Objective: Assess operator performance, identify training needs, and analyze downtime related to human or shift-based factors.

Questions:

1. **Which operators are the most and least efficient?**
2. **How much downtime is linked to operator errors or manual intervention?**
3. **What is the top downtime causes across operators?**
4. **How does performance vary across different shifts?**
5. **Are certain operators more efficient with specific products or time periods?**

Project Milestones

Milestone 1: Business Questions and Requirements (Week 1)

Objectives:

1. Identify the main business goals and KPIs.
2. Define the problem statement.
3. Collect all data sources.
4. Draft a list of business questions to answer using the dashboard.
5. Sketch initial ideas for visuals and insights.

Deliverables:

- Business requirement document.
- List of key metrics and dimensions.
- Draft dashboard wireframe.

Milestone 2: Data Cleaning & Modelling (Week 2)

Objectives:

1. Import and transform data in Power Query.
2. Handle missing, duplicate, or inconsistent data.
3. Create relationships between fact and dimension tables.
4. Design a star schema.
5. Ensure data integrity and refresh setup.

Deliverables:

- Cleaned dataset ready for analysis.
- Completed data model diagram in Power BI.
- Documented data dictionary.

Milestone 3: DAX Measures & Mockup (Week 3)

Objectives:

1. Create DAX measures for KPIs (e.g., total downtime, OEE, productivity rate).
2. Implement calculated columns/tables as needed.
3. Test accuracy of calculations.
4. Build mockup visuals to check insights alignment with business questions.

Deliverables:

- List of all DAX measures with descriptions.
- Mockup dashboard layout.
- Validation report of DAX calculations.

Milestone 4: Data Visualization & Reporting (Week 4)

Objectives:

1. Design the final dashboard following visual best practices.
2. Apply color themes, slicers, filters, and interactivity.
3. Write insights and recommendations based on findings.
4. Publish and present the report.

Deliverables:

- Fully interactive Power BI dashboard.
- Presentation/report with insights and recommendations.
- Documentation of dashboard usage and KPIs.

Power Query

Query Name: Line Productivity

Purpose:

This query imports and cleans productivity data for each manufacturing line. It prepares the dataset for further analysis by promoting headers, converting data types, and removing unnecessary columns.

Step-by-Step Transformation Details

Step	Transformation	Description
1	**Source**	The query imports data from the Excel file located at: F:\رواد مصر الرقمية\Final Project Details\Final Project\Manufacturing_Line_Productivity.xlsx. All sheets in the workbook are read into Power Query.
2	**Line productivity_Sheet**	Extracts the worksheet named **"Line productivity"** from the Excel file. This step isolates the relevant sheet that contains the productivity data.
3	**Promoted Headers**	Promotes the first row of the sheet to become the column headers, ensuring that column names (such as *Date*, *Product*, *Batch*, etc.) are properly recognized.
4	**Changed Type**	• **Date** → Date • **Product** → Text • **Batch** → Whole Number (Int64) • **Operator** → Text • **Start Time** → Date/Time • **End Time** → Date/Time • **Column7** → Any (temporary column)
5	**Removed Columns**	Removes the unnecessary column **Column7**, which may have been a blank or placeholder column in the source file.

Output:

A clean table with the following columns, ready for loading or further transformation:

- Date
- Product
- Batch
- Operator
- Start Time
- End Time

Code:

```
let
    Source = Excel.Workbook(File.Contents("F:\رواد مصر الرقمية\Final Project Details\Final Project\Manufacturing_Line_Productivity.xlsx"), null, true),
    #"Line productivity_Sheet" = Source{[Item="Line productivity",Kind="Sheet"]}[Data],
    #"Promoted Headers" = Table.PromoteHeaders(#"Line productivity_Sheet", [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Date", type date}, {"Product", type text}, {"Batch", Int64.Type}, {"Operator", type text}, {"Start Time", type datetime}, {"End Time", type datetime}, {"Column7", type any}}),
    #"Removed Columns" = Table.RemoveColumns(#"Changed Type",{"Column7"})
in
    #"Removed Columns"
```

Query Name: Line Downtime

Purpose:

This query imports and prepares machine downtime data from the manufacturing line Excel file. It ensures that headers and data types are correctly structured for use in Power BI analysis of production interruptions.

Step-by-Step Transformation Details

Step	Transformation	Description
1	**Source**	Imports data from the Excel workbook located at: F:\رواد مصر الرقمية\Final Project Details\Final Project\Manufacturing_Line_Productivity.xlsx.
2	**Line downtime_Sheet**	Extracts the worksheet named **"Line downtime"** from the Excel file, isolating the relevant downtime data.
3	**Promoted Headers**	Promotes the first row of the sheet to become column headers. This helps properly identify column names such as *Downtime factor*, *Batch*, etc.
4	**Changed Type**	Initially assigns data types to the columns. The **Downtime factor** column is set to *Whole Number*, while Columns 3–13 are also converted to *Whole Number* type. The first column (Column1) remains as type *any* since its data might not be numeric.
5	**Promoted Headers1**	A second header promotion is performed — this step usually occurs when the dataset contains **two header rows** at the top of the Excel sheet. It ensures that the correct header row (e.g., "Batch", "1", "2", "3", etc.) is used as the final column names.
6	**Changed Type1**	• **Batch** → Whole Number • **1–12** → Whole Numbers

Output:

The resulting table is structured and cleaned with the following columns:

- Batch
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12

Each numeric column (1–12) likely represents **specific downtime factors or categories** across different lines or time intervals.

Code:

```
let
    Source = Excel.Workbook(File.Contents("F:\رواد مصر الرقمية\Final Project Details\Final
Project\Manufacturing_Line_Productivity.xlsx"), null, true),
    #"Line downtime_Sheet" = Source{[Item="Line downtime",Kind="Sheet"]}[Data],
    #"Promoted Headers" = Table.PromoteHeaders(#"Line downtime_Sheet", [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Column1", type any},
{"Downtime factor", Int64.Type}, {"Column3", Int64.Type}, {"Column4", Int64.Type}, {"Column5",
Int64.Type}, {"Column6", Int64.Type}, {"Column7", Int64.Type}, {"Column8", Int64.Type}, {"Column9",
Int64.Type}, {"Column10", Int64.Type}, {"Column11", Int64.Type}, {"Column12", Int64.Type},
{"Column13", Int64.Type}}),
    #"Promoted Headers1" = Table.PromoteHeaders(#"Changed Type", [PromoteAllScalars=true]),
    #"Changed Type1" = Table.TransformColumnTypes(#"Promoted Headers1",{{"Batch", Int64.Type}, {"1",
type number}, {"2", Int64.Type}, {"3", Int64.Type}, {"4", Int64.Type}, {"5", Int64.Type}, {"6",
Int64.Type}, {"7", Int64.Type}, {"8", Int64.Type}, {"9", Int64.Type}, {"10", Int64.Type}, {"11",
Int64.Type}, {"12", Int64.Type}})
in
    #"Changed Type1"
```

Query Name: Products

Purpose:

This query imports and structures the product master data, defining each product's attributes such as flavor, size, and minimum batch time. The resulting table serves as a **dimension table** in the Power BI data model for product-related analysis and relationships.

Step-by-Step Transformation Details

Step	Transformation	Description
1	**Source**	Loads data from the Excel workbook located at: F:\رواد مصر الرقمية\Final Project Details\Final Project\Manufacturing_Line_Productivity.xlsx.
2	**Products_Sheet**	Extracts the worksheet named **"Products"** from the Excel file. This sheet contains the list of products and their attributes.
3	**Promoted Headers**	Promotes the first row to become the column headers, ensuring that field names such as *Product*, *Flavor*, *Size*, and *Min batch time* are recognized.
4	**Changed Type**	• **Product** → Text • **Flavor** → Text • **Size** → Text • **Min batch time** → Whole Number

Output:

A clean and structured **Products dimension table** with the following columns:

- Product
- Flavor
- Size
- Min batch time

This table is ready to be linked with fact tables such as **Line Productivity** or **Line Downtime** for detailed performance analysis by product type.

Code:

```
let
    Source = Excel.Workbook(File.Contents("F:\رواد مصر الرقمية\Final Project Details\Final Project\Manufacturing_Line_Productivity.xlsx"), null, true),
    Products_Sheet = Source{[Item="Products",Kind="Sheet"]}[Data],
    #"Promoted Headers" = Table.PromoteHeaders(Products_Sheet, [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Product", type text}, {"Flavor", type text}, {"Size", type text}, {"Min batch time", Int64.Type}})
in
    #"Changed Type"
```

Query Name: Downtime Factors

Purpose:

This query imports and cleans the list of machine downtime factors from the Excel file. It defines the causes of downtime, their corresponding descriptions, and whether they are related to operator error. The resulting table serves as a dimension table for categorizing downtime events in the Power BI model.

Step-by-Step Transformation Details

Step	Transformation	Description
1	**Source**	Loads data from the Excel workbook located at: F:\رواد مصر الرقمية\Final Project Details\Final Project\Manufacturing_Line_Productivity.xlsx.
2	**Downtime factors_Sheet**	Extracts the worksheet named "Downtime factors" from the Excel file. This sheet contains factor IDs, descriptions, and indicators of operator-related issues.
3	**Promoted Headers**	Promotes the first row of the sheet to become column headers, ensuring that column names like *Factor*, *Description*, and *Operator Error* are properly identified.
4	**Changed Type**	• **Factor** → Whole Number • **Description** → Text • **Operator Error** → Text

Output:

A structured Downtime Factors dimension table with the following columns:

- Factor
- Description
- Operator Error

This table provides context for downtime events and supports detailed analysis of downtime causes in Power BI dashboards.

Code:

```
let
    Source = Excel.Workbook(File.Contents("F:\رواد مصر الرقمية\Final Project Details\Final Project\Manufacturing_Line_Productivity.xlsx"), null, true),
    #"Downtime factors_Sheet" = Source{[Item="Downtime factors",Kind="Sheet"]}[Data],
    #"Promoted Headers" = Table.PromoteHeaders(#"Downtime factors_Sheet", [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Factor", Int64.Type}, {"Description", type text}, {"Operator Error", type text}})
in
    #"Changed Type"
```

Query Name: factLineProductivity

Purpose:

This query prepares the Line Productivity Fact Table, combining raw production data with dimension tables to create a clean, relational dataset. It links production batches with product and operator information, ensuring consistent data typing and proper table relationships within the Power BI model.

Step-by-Step Transformation Details

Step	Transformation	Description
1	**Source**	Imports data from the Excel workbook: F:\رواد مصر الرقمية\Final Project Details\Final Project\Manufacturing_Line_Productivity.xlsx.
2	**Line productivity_Sheet**	Extracts the "Line productivity" worksheet from the Excel file.
3	**Promoted Headers**	Promotes the first row to become column headers for proper field recognition.
4	**Changed Type**	• **Date** → Date • **Product** → Text • **Batch** → Whole Number • **Operator** → Text • **Start Time & End Time** → DateTime • **Column7** → Any
5	**Removed Columns**	Removes the unnecessary column Column7, which likely contains blank or irrelevant data.
6	**Changed Type1**	Converts Start Time and End Time from *datetime* to *time* for clearer time-based calculations.
7	**Merged Queries (dimProducts)**	Performs a Left Outer Join with the dimProducts table on the *Product* field to bring in product IDs. This step enriches the fact table with related product information.
8	**Expanded dimProducts**	Expands the joined column to include ProductID from the dimProducts table.

Step	Transformation	Description
9	**Removed Columns1**	Removes the original *Product* column to avoid redundancy, since ProductID now serves as the key.
10	**Reordered Columns**	Reorganizes columns in a logical order: *Date → ProductID → Batch → Operator → Start Time → End Time*.
11	**Renamed Columns**	Renames dimProducts.ProductID to simply ProductID for consistency.
12	**Merged Queries1 (dimOperatorName)**	Performs another Left Outer Join, this time linking to the dimOperatorName table on the *Operator* field to include operator identifiers.
13	**Expanded dimOperatorName**	Expands the joined column to include the OperatorID from the dimOperatorName dimension.
14	**Reordered Columns1**	Reorders columns to group related data fields: *Date, ProductID, Batch, OperatorID, Operator, Start Time, End Time*.
15	**Removed Columns2**	Removes the textual *Operator* column, keeping only OperatorID for relational consistency.
16	**Changed Type2**	• **OperatorID** → Text • **Batch** → Text
17	**Renamed Columns1**	Renames Start Time → StartTime and End Time → EndTime for standardized naming conventions.
18	**Changed Type3**	Ensures ProductID is a Whole Number (*Int64*) to maintain proper relationships with the Product dimension table.

Output:

The resulting factLineProductivity table contains the following key columns:

- Date
- ProductID
- Batch
- OperatorID
- StartTime
- EndTime

This table is used for time-based and performance analysis in Power BI dashboards, linked to dimProducts and dimOperatorName to support detailed insights on productivity per product and operator.

Code:

```
let
    Source = Excel.Workbook(File.Contents("F:\رواد مصر الرقمية\Final Project Details\Final Project\Manufacturing_Line_Productivity.xlsx"), null, true),
    #"Line productivity_Sheet" = Source{[Item="Line productivity",Kind="Sheet"]}[Data],
    #"Promoted Headers" = Table.PromoteHeaders(#"Line productivity_Sheet", [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Date", type date}, {"Product", type text}, {"Batch", Int64.Type}, {"Operator", type text}, {"Start Time", type datetime}, {"End Time", type datetime}, {"Column7", type any}}),
    #"Removed Columns" = Table.RemoveColumns(#"Changed Type",{"Column7"}),
    #"Changed Type1" = Table.TransformColumnTypes(#"Removed Columns",{{"Start Time", type time}, {"End Time", type time}}),
    #"Merged Queries" = Table.NestedJoin(#"Changed Type1", {"Product"}, dimProducts, {"Product"}, "dimProducts", JoinKind.LeftOuter),
    #"Expanded dimProducts" = Table.ExpandTableColumn(#"Merged Queries", "dimProducts", {"ProductID"}, {"dimProducts.ProductID"}),
    #"Removed Columns1" = Table.RemoveColumns(#"Expanded dimProducts",{"Product"}),
    #"Reordered Columns" = Table.ReorderColumns(#"Removed Columns1",{"Date", "dimProducts.ProductID", "Batch", "Operator", "Start Time", "End Time"}),
    #"Renamed Columns" = Table.RenameColumns(#"Reordered Columns",{{"dimProducts.ProductID", "ProductID"}}),
    #"Merged Queries1" = Table.NestedJoin(#"Renamed Columns", {"Operator"}, dimOperatorName, {"Operator"}, "dimOperatorName", JoinKind.LeftOuter),
    #"Expanded dimOperatorName" = Table.ExpandTableColumn(#"Merged Queries1", "dimOperatorName", {"OperatorID"}, {"OperatorID"}),
    #"Reordered Columns1" = Table.ReorderColumns(#"Expanded dimOperatorName",{"Date", "ProductID", "Batch", "OperatorID", "Operator", "Start Time", "End Time"}),
    #"Removed Columns2" = Table.RemoveColumns(#"Reordered Columns1",{"Operator"}),
    #"Changed Type2" = Table.TransformColumnTypes(#"Removed Columns2",{{"OperatorID", type text}, {"Batch", type text}}),
    #"Renamed Columns1" = Table.RenameColumns(#"Changed Type2",{{"Start Time", "StartTime"}, {"End Time", "EndTime"}}),
    #"Changed Type3" = Table.TransformColumnTypes(#"Renamed Columns1",{{"ProductID", Int64.Type}})
in
    #"Changed Type3"
```

Query Name: factLineDowntime

Purpose:

This query transforms and structures the **Line Downtime** data into a proper fact table format. It cleans, reshapes, and standardizes the raw data from the Excel source, converting wide-format downtime entries into a normalized table. The result is a ready-to-use dataset for analyzing machine downtime by batch and downtime factor in Power BI.

Step-by-Step Transformation Details

Step	Transformation	Description
1	**Source**	Imports data from the Excel workbook: F:\رواد مصر الرقمية\Final Project Details\Final Project\Manufacturing_Line_Productivity.xlsx.
2	**Line downtime_Sheet**	Extracts the worksheet named **"Line downtime"** that contains downtime records by batch and factor.
3	**Promoted Headers**	Promotes the first row to column headers for proper field naming.
4	**Changed Type**	Converts multiple columns (e.g., *Downtime factor*, *Column3*–*Column13*) to appropriate numeric types (mostly Int64). The first column remains as *any* type.
5	**Promoted Headers1**	Performs a **second header promotion**, often necessary when the source sheet includes an extra header row. The result correctly labels columns such as *Batch*, *1*, *2*, … *12*.
6	**Changed Type1**	• **Batch** → Whole Number • **1–12** → Whole Number
7	**Unpivoted Other Columns**	Converts the data from **wide format** (multiple downtime factor columns) into a **long format** — each downtime factor (1–12) becomes a row with corresponding batch and value. **Batch** is kept as a fixed column, while other columns are unpivoted into *Attribute* and *Value* pairs.
8	**Renamed Columns**	Renames the *Value* column to **DowntimeNo** to represent the numeric downtime values.

Step	Transformation	Description
9	**Changed Type2**	Converts **DowntimeNo** to Whole Number (*Int64*) for consistency.
10	**Renamed Columns1**	Renames **DowntimeNo** → **DowntimeMinutes**, clarifying that these numbers represent the downtime duration in minutes.
11	**Changed Type3**	Converts **Batch** from Whole Number to *Text* type to align with other related tables (e.g., *factLineProductivity*).
12	**Renamed Columns2**	Renames the *Attribute* column to **DowntimeFactorID**, indicating the factor or category responsible for the downtime.

Output:

The final **factLineDowntime** table contains the following columns:

- Batch
- DowntimeFactorID
- DowntimeMinutes

This structure enables efficient relationships with dimension tables (e.g., **dimDowntimeFactors**) and allows for clear analysis of downtime causes, durations, and frequency per batch in Power BI.

Code:

```
let
    Source = Excel.Workbook(File.Contents("F:\رواد مصر الرقمية\Final Project Details\Final
Project\Manufacturing_Line_Productivity.xlsx"), null, true),
    #"Line downtime_Sheet" = Source{[Item="Line downtime",Kind="Sheet"]}[Data],
    #"Promoted Headers" = Table.PromoteHeaders(#"Line downtime_Sheet", [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Column1", type any},
{"Downtime factor", Int64.Type}, {"Column3", Int64.Type}, {"Column4", Int64.Type}, {"Column5",
Int64.Type}, {"Column6", Int64.Type}, {"Column7", Int64.Type}, {"Column8", Int64.Type}, {"Column9",
Int64.Type}, {"Column10", Int64.Type}, {"Column11", Int64.Type}, {"Column12", Int64.Type},
{"Column13", Int64.Type}}),
    #"Promoted Headers1" = Table.PromoteHeaders(#"Changed Type", [PromoteAllScalars=true]),
    #"Changed Type1" = Table.TransformColumnTypes(#"Promoted Headers1",{{"Batch", Int64.Type}, {"1",
type number}, {"2", Int64.Type}, {"3", Int64.Type}, {"4", Int64.Type}, {"5", Int64.Type}, {"6",
Int64.Type}, {"7", Int64.Type}, {"8", Int64.Type}, {"9", Int64.Type}, {"10", Int64.Type}, {"11",
Int64.Type}, {"12", Int64.Type}}),
    #"Unpivoted Other Columns" = Table.UnpivotOtherColumns(#"Changed Type1", {"Batch"}, "Attribute",
"Value"),
    #"Renamed Columns" = Table.RenameColumns(#"Unpivoted Other Columns",{{"Value", "DowntimeNo"}}),
    #"Changed Type2" = Table.TransformColumnTypes(#"Renamed Columns",{{"DowntimeNo", Int64.Type}}),
    #"Renamed Columns1" = Table.RenameColumns(#"Changed Type2",{{"DowntimeNo", "DowntimeMinutes"}}),
    #"Changed Type3" = Table.TransformColumnTypes(#"Renamed Columns1",{{"Batch", type text}}),
    #"Renamed Columns2" = Table.RenameColumns(#"Changed Type3",{{"Attribute", "DowntimeFactorID"}})
in
    #"Renamed Columns2"
```

Query Name: dimDowntimeFactors

Purpose:

This query creates the Downtime Factors Dimension Table, containing the unique list of downtime causes, their descriptions, and whether they are related to operator errors. It standardizes column names and data types for accurate relationships with the factLineDowntime table.

Step-by-Step Transformation Details

Step	Transformation	Description
1	**Source**	Loads data from the Excel workbook: F:\رواد مصر الرقمية\Final Project Details\Final Project\Manufacturing_Line_Productivity.xlsx.
2	**Downtime factors_Sheet**	Extracts the worksheet named "Downtime factors", which lists all possible downtime categories and related information.
3	**Promoted Headers**	Promotes the first row to become column headers, identifying *Factor*, *Description*, and *Operator Error*.
4	**Changed Type**	• **Factor** → Whole Number • **Description** → Text • **Operator Error** → Text
5	**Renamed Columns**	• **Factor** → DowntimeFactorID • **Operator Error** → OperatorError
6	**Changed Type1**	Converts DowntimeFactorID from *Whole Number* to *Text* type to align with the key type used in factLineDowntime, ensuring proper table relationships in the data model.

Output:

A structured dimDowntimeFactors table with the following columns:

- DowntimeFactorID
- Description
- OperatorError

This table serves as the lookup or dimension table for categorizing and analyzing downtime events by cause and operator responsibility in Power BI.

Code:

```
let
    Source = Excel.Workbook(File.Contents("F:\رواد مصر الرقمية\Final Project Details\Final Project\Manufacturing_Line_Productivity.xlsx"), null, true),
    #"Downtime factors_Sheet" = Source{[Item="Downtime factors",Kind="Sheet"]}[Data],
    #"Promoted Headers" = Table.PromoteHeaders(#"Downtime factors_Sheet", [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Factor", Int64.Type}, {"Description", type text}, {"Operator Error", type text}}),
    #"Renamed Columns" = Table.RenameColumns(#"Changed Type",{{"Factor", "DowntimeFactorID"}, {"Operator Error", "OperatorError"}}),
    #"Changed Type1" = Table.TransformColumnTypes(#"Renamed Columns",{{"DowntimeFactorID", type text}})
in
    #"Changed Type1"
```

Query Name: dimProducts

Purpose:

This query creates the Products Dimension Table, containing each product's unique identifier, name, flavor, size, and minimum batch time.
It cleans and standardizes product details, ensuring consistency in the Power BI data model and enabling reliable joins with the factLineProductivity table.

Step-by-Step Transformation Details

Step	Transformation	Description
1	**Source**	Loads data from the Excel workbook: F:\رواد مصر الرقمية\Final Project Details\Final Project\Manufacturing_Line_Productivity.xlsx.
2	**Products_Sheet**	Extracts the worksheet named "Products", which contains product details such as Product, Flavor, Size, and Minimum Batch Time.
3	**Promoted Headers**	Promotes the first row of data to column headers, labeling columns correctly.
4	**Changed Type**	• **Product** → Text • **Flavor** → Text • **Size** → Text • **Min batch time** → Whole Number
5	**Replaced Value / Replaced Value1**	**Cleans and standardizes the "Size" column:** 1. Replaces "2 L" with "2000" 2. Removes " ml" suffix
6	**Changed Type1**	Converts "Size" column from Text to Whole Number after cleaning.
7	**Renamed Columns**	• **Size** → Size(ml) • **Min batch time** → BatchTime(Min)
8	**Added Index**	Adds an auto-generated sequential ProductID column starting at 1, serving as the unique key for each product.
9	**Reordered Columns**	Reorders columns for clarity: ProductID, Product, Flavor, Size(ml), BatchTime(Min).

Step	Transformation	Description
10	**Removed Errors**	Removes any rows containing data errors to ensure a clean dimension table.

Output:

A well-structured dimProducts table with the following columns:

- ProductID
- Product
- Flavor
- Size(ml)
- BatchTime(Min)

The dimProducts table serves as the lookup/dimension table for all product-related analysis. It connects to factLineProductivity (and potentially other fact tables) through the ProductID field, enabling consistent filtering and reporting by product characteristics.

Code:

```
let
    Source = Excel.Workbook(File.Contents("F:\رواد مصر الرقمية\Final Project Details\Final
Project\Manufacturing_Line_Productivity.xlsx"), null, true),
    Products_Sheet = Source{[Item="Products",Kind="Sheet"]}[Data],
    #"Promoted Headers" = Table.PromoteHeaders(Products_Sheet, [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Product", type text},
{"Flavor", type text}, {"Size", type text}, {"Min batch time", Int64.Type}}),
    #"Replaced Value" = Table.ReplaceValue(#"Changed Type","2
L","2000",Replacer.ReplaceText,{"Size"}),
    #"Replaced Value1" = Table.ReplaceValue(#"Replaced Value","
ml","",Replacer.ReplaceText,{"Size"}),
    #"Changed Type1" = Table.TransformColumnTypes(#"Replaced Value1",{{"Size", Int64.Type}}),
    #"Renamed Columns" = Table.RenameColumns(#"Changed Type1",{{"Size", "Size(ml)"}, {"Min batch
time", "BatchTime(Min)"}}),
    #"Added Index" = Table.AddIndexColumn(#"Renamed Columns", "Index", 1, 1, Int64.Type),
    #"Reordered Columns" = Table.ReorderColumns(#"Added Index",{"Index", "Product", "Flavor",
"Size(ml)", "BatchTime(Min)"}),
    #"Renamed Columns1" = Table.RenameColumns(#"Reordered Columns",{{"Index", "ProductKey"}}),
    #"Changed Type2" = Table.TransformColumnTypes(#"Renamed Columns1",{{"ProductKey", type text}}),
    #"Renamed Columns2" = Table.RenameColumns(#"Changed Type2",{{"ProductKey", "ProductID"}}),
    #"Removed Errors" = Table.RemoveRowsWithErrors(#"Renamed Columns2"),
    #"Changed Type3" = Table.TransformColumnTypes(#"Removed Errors",{{"ProductID", Int64.Type}})
in
    #"Changed Type3"
```

Query Name: dimOperatorName

Purpose:

This query creates the **Operator Dimension Table**, providing a clean reference for all machine operators.
It ensures each operator has a unique identifier (OperatorID) and a consistent name format, which supports accurate performance and productivity analysis in Power BI.

Step-by-Step Transformation Details

Step	Transformation	Description
1	**Source**	Loads data from the Excel workbook: F:\رواد مصر الرقمية\Final Project Details\Final Project\Manufacturing_Line_Productivity.xlsx.
2	**OperatorName_Sheet**	Extracts the worksheet named **"OperatorName"**, containing operator identifiers and their names.
3	**Promoted Headers**	Promotes the first row to become column headers (OperatorKey, Operator).
4	**Changed Type**	• **OperatorKey** → Whole Number • **Operator** → Text
5	**Renamed Columns**	Renames OperatorKey to **OperatorID** — standardizing naming conventions for model relationships.
6	**Changed Type1**	Converts OperatorID from Whole Number to Text to match key types used in related fact tables such as factLineProductivity and factLineDowntime.

Output:

A clean and standardized **dimOperatorName** table with the following columns:

- OperatorID
- Operator

The dimOperatorName table acts as a **lookup dimension** for all operator-related data. It connects to fact tables such as **factLineProductivity** via OperatorID, enabling analysis of productivity, efficiency, and error trends by operator within Power BI.

Code:

```
let
    Source = Excel.Workbook(File.Contents("F:\رواد مصر الرقمية\Final Project Details\Final
Project\Manufacturing_Line_Productivity.xlsx"), null, true),
    OperatorName_Sheet = Source{[Item="OperatorName",Kind="Sheet"]}[Data],
    #"Promoted Headers" = Table.PromoteHeaders(OperatorName_Sheet, [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"OperatorKey", Int64.Type},
{"Operator", type text}}),
    #"Renamed Columns" = Table.RenameColumns(#"Changed Type",{{"OperatorKey", "OperatorID"}}),
    #"Changed Type1" = Table.TransformColumnTypes(#"Renamed Columns",{{"OperatorID", type text}})
in
    #"Changed Type1"
```

Query Name: dimDate

Purpose:

This query creates a Date Dimension Table using Power Query M code. It automatically generates all dates between a user-defined start and end date, along with key time attributes like year, quarter, week number, month number, month name, and day name. This table supports time-based analysis and relationships in the data model.

Step-by-Step Transformation Details

Step	Transformation	Description
1	**Parameters (StartDate, EndDate)**	The function takes two parameters: a StartDate and an EndDate — defining the range of dates for the calendar.
2	**Normalize Dates**	Converts both parameters to valid date formats using: #date(Date.Year(StartDate), Date.Month(StartDate), Date.Day(StartDate))
3	**GetDateCount**	Calculates how many days exist between the start and end dates using: Duration.Days(EndDate - StartDate)
4	**GetDateList**	Creates a continuous list of dates using: List.Dates(StartDate, GetDateCount, #duration(1,0,0,0)) — this generates a list incremented by 1 day.
5	**DateListToTable**	Converts the list into a Power Query table with a single column named "Date".
6	**Add Year Column**	Adds a Year column using Date.Year([Date]).
7	**Add Quarter Column**	Adds a Quarter column in the format "Q1", "Q2", etc., using: "Q" & Number.ToText(Date.QuarterOfYear([Date])).
8	**Add Week Number Column**	Adds the Week Number of each date using Date.WeekOfYear([Date]).
9	**Add Month Number Column**	Adds a Month Number column using Date.Month([Date]).

Step	Transformation	Description
10	**Add Month Name Column**	Adds a full Month Name (e.g., January, February) using: Date.ToText([Date],"MMMM").
11	**Add Day of Week Column**	Adds a full Day Name (e.g., Monday, Tuesday) using: Date.ToText([Date],"dddd").

Output:

A structured dimDate table with the following columns:

- Date
- Year
- Quarter
- Week Number
- Month Number
- Month
- Day of Week

The dimDate table acts as a calendar dimension, linked to other fact tables (such as factLineProductivity or factLineDowntime) through the Date field. This enables powerful time intelligence analysis in Power BI — for example:

- Downtime trend by week or month
- Productivity per quarter
- Year-over-year comparison

Code:

```
//Create Date Dimension
(StartDate as date, EndDate as date)=>

let
    //Capture the date range from the parameters
    StartDate = #date(Date.Year(StartDate), Date.Month(StartDate),
    Date.Day(StartDate)),
    EndDate = #date(Date.Year(EndDate), Date.Month(EndDate),
    Date.Day(EndDate)),

    //Get the number of dates that will be required for the table
    GetDateCount = Duration.Days(EndDate - StartDate),

    //Take the count of dates and turn it into a list of dates
    GetDateList = List.Dates(StartDate, GetDateCount,
    #duration(1,0,0,0)),

    //Convert the list into a table
    DateListToTable = Table.FromList(GetDateList,
    Splitter.SplitByNothing(), {"Date"}, null, ExtraValues.Error),

    //Create various date attributes from the date column
    //Add Year Column
    YearNumber = Table.AddColumn(DateListToTable, "Year",
    each Date.Year([Date])),

    //Add Quarter Column
    QuarterNumber = Table.AddColumn(YearNumber , "Quarter",
    each "Q" & Number.ToText(Date.QuarterOfYear([Date]))),

    //Add Week Number Column
    WeekNumber= Table.AddColumn(QuarterNumber , "Week Number",
    each Date.WeekOfYear([Date])),

    //Add Month Number Column
    MonthNumber = Table.AddColumn(WeekNumber, "Month Number",
    each Date.Month([Date])),

    //Add Month Name Column
    MonthName = Table.AddColumn(MonthNumber , "Month",
    each Date.ToText([Date],"MMMM")),

    //Add Day of Week Column
    DayOfWeek = Table.AddColumn(MonthName , "Day of Week",
    each Date.ToText([Date],"dddd"))

in
    DayOfWeek
```

DAX

Table: factLineProductivity

Calculated columns:

1. StartDateTime

Formula:

```
StartDateTime = factLineProductivity[Date] + factLineProductivity[StartTime]
```

Description:
Combines the Date and StartTime columns into a single datetime value. This creates a precise timestamp for when an operation begins.

Purpose:
Used for accurate duration calculations and time-based analysis across days.

2. EndDateTime

Formula:

```
EndDateTime =
IF(
    factLineProductivity[EndTime] < factLineProductivity[StartTime],
    DATEADD(dimCalendar[Date], 1, DAY) + factLineProductivity[EndTime],
    factLineProductivity[Date] + factLineProductivity[EndTime]
)
```

Description:
Determines the actual end date and time of an operation. If the EndTime is earlier than the StartTime, it assumes the process continued past midnight and adds one day to the date.

Purpose:
Ensures correct handling of operations that cross over to the next day.

Logic Summary:

- Normal case: EndTime > StartTime → same-day operation
- Crossover case: EndTime < StartTime → operation continues into the next day

3. Duration (Min)

Formula:

```
Duration (Min) =
DATEDIFF(
    factLineProductivity[StartDateTime],
    factLineProductivity[EndDateTime],
    MINUTE
)
```

Description:
Calculates the total duration of each operation in minutes.

Purpose:
Used to measure machine or process activity time — essential for productivity, efficiency, and downtime analysis.

Dependencies:
Relies on both StartDateTime and EndDateTime being correctly calculated.

Table: TimeTable

Table Formula:

```
TimeTable =
ADDCOLUMNS (
    GENERATESERIES (0, 1439, 1),    -- 0 to 1439 minutes (24*60 - 1)
    "Time", TIME (INT ([Value] / 60), MOD ([Value], 60), 0),
    "Hour", INT ([Value] / 60),
    "Minute", MOD ([Value], 60),
    "Period", IF (INT ([Value] / 60) < 12, "AM", "PM"),
    "TimeText", FORMAT (TIME (INT ([Value] / 60), MOD ([Value], 60), 0), "hh:mm
AM/PM")
)
```

Description:

This DAX script creates a **Time Dimension Table** with one row for every minute of a 24-hour day (from 00:00 to 23:59). It serves as a helper table for time-based reporting and joins with the time columns (like **StartTime** and **EndTime**) in fact tables.

Column Name	Description
Value	Sequential numeric index representing minutes since midnight (0 to 1439).
Time	The actual time value (e.g., 8:30 AM). Created using the TIME() function.
Hour	The hour part of the time (0–23). Useful for grouping by hour.
Minute	The minute part of the time (0–59).
Period	AM/PM indicator, separating morning and evening times.
TimeText	Readable time format (e.g., “08:30 AM”). Ideal for slicers and visuals.

Purpose:

- Enables **time-based slicing and filtering** in Power BI dashboards.
- Facilitates **relationships** with time fields in fact tables (e.g., **StartTime**, **EndTime**).
- Supports **custom visuals** like hourly trend charts and machine productivity heatmaps.

Calculated Column Name: Shift Time

Formula:

```
Shift Time =
SWITCH(
    TimeTable[Period],
    "AM", "Day Shift",
    "PM", "Night Shift"
)
```

Description:
This calculated column classifies each time record into a **work shift** based on the Period value from the **TimeTable**. If the period is "AM", it assigns the label **"Day Shift"**; if "PM", it assigns **"Night Shift"**. This simplifies filtering and analysis of productivity, downtime, or operator performance by **shift type**.

Usage Example:

Can be used in visuals or filters to compare performance between **Day Shift** and **Night Shift**, such as average production duration, downtime, or efficiency metrics.

Table: _Measures

Measure Name: Total Productive Time (Min)

Formula:

```
Total Productive Time (Min) =
[Total Production Time (Min)] - [Total Downtime (Min)]
```

Description:

This measure calculates the **total productive minutes** of the manufacturing line by subtracting the downtime duration from the total production time. It provides a clear view of how much time the line was actually producing goods versus being idle.

Business Purpose:

Used to analyze **line efficiency and productivity**, helping identify how effectively production time is utilized after accounting for machine or operator downtime.

Measure Name: Total Productive Time

Formula:

```
Total Productive Time =
VAR TotalMinutes = [Total Productive Time (Min)]
VAR Hours = INT(TotalMinutes / 60)
VAR Minutes = MOD(TotalMinutes, 60)
RETURN
    Hours & " Hrs " & FORMAT(Minutes, "00") & " Mins"
```

Description:

This measure converts the total productive time (in minutes) into a **readable text format** displaying hours and minutes. It enhances data presentation by translating numeric time values into a more understandable and user-friendly format.

Business Purpose:

Used primarily for **dashboard visualization and reporting**, allowing managers and operators to instantly grasp total productive time without interpreting raw numbers.

Measure Name: Total Production Time (Min)

Formula:

```
Total Production Time (Min) =
SUM(factLineProductivity[Duration(Min)])
```

Description:

This measure calculates the **total duration of all production activities** in minutes by summing up the Duration(Min) column from the factLineProductivity table.

Business Purpose:

Used to quantify the **overall time spent in production operations**, serving as a base metric for efficiency, utilization, and downtime analyses in manufacturing performance dashboards.

Dependencies:

- **Table:** factLineProductivity
- **Column:** Duration(Min) (calculated column that measures time between StartDateTime and EndDateTime)

Measure Name: Total Production Time

```
Total Production Time =
VAR TotalMinutes = SUM(factLineProductivity[Duration(Min)])
VAR Hours = INT(TotalMinutes / 60)
VAR Minutes = MOD(TotalMinutes, 60)
RETURN
    Hours & " Hrs " & FORMAT(Minutes, "00") & " Mins"
```

Description:

This measure provides a **human-readable format** of the total production time by converting the total minutes of production into hours and minutes.

While Total Production Time (Min) gives a numeric value (e.g., *185 minutes*), this measure converts that into a more interpretable display such as **"3 Hrs 05 Mins."**

Business Purpose:

Used to display total production duration in a **clear, user-friendly format** for dashboards and reports. It enhances readability when presenting time metrics to operations teams or management.

Dependencies:

- **Table:** factLineProductivity
- **Column:** Duration(Min) (calculated column for elapsed time in minutes)

Logic Breakdown:

1. **TotalMinutes:** Calculates total duration in minutes.
2. **Hours:** Divides total minutes by 60 and takes the integer part.
3. **Minutes:** Calculates the remainder minutes using the MOD function.
4. **Return:** Combines both with formatted text ("Hrs" and "Mins").

Measure Name: Total Downtime (Min)

Formula:

```
Total Downtime (Min) =
SUM(factLineDowntime[DowntimeMinutes])
```

Description:

This measure computes the **total sum of downtime minutes** from the factLineDowntime table. It quantifies all recorded periods when the production line was not operating, expressed in minutes.

It serves as the foundational metric for downtime analysis and is used in other measures such as **Total Productive Time (Min)** and **Line Efficiency (%)**.

Business Purpose:

To measure **total downtime duration** for a selected time frame, production line, product, or operator. This metric helps identify bottlenecks, maintenance needs, or operator-related inefficiencies.

Dependencies:

- **Table:** factLineDowntime
- **Column:** DowntimeMinutes (numeric column representing downtime duration per event)

Logic Breakdown:

1. Retrieves all downtime durations from the fact table.
2. Sums them to return the total downtime in minutes for the applied filter context (date, product, operator, etc.).

Measure Name: Total Downtime

Formula:

```
Total Downtime =
    VAR TotalMinutes = SUM(factLineDowntime[DowntimeMinutes])
    VAR Hours = INT(TotalMinutes / 60)
    VAR Minutes = MOD(TotalMinutes, 60)
    RETURN
        Hours & " Hrs " & FORMAT(Minutes, "00") & " Mins"
```

Description:

This measure provides a **formatted textual version** of total downtime, converting minutes into a **readable hours–minutes format** (e.g., "3 Hrs 45 Mins"). It is used for dashboards and reports where you want to display duration values more intuitively than raw numbers.

Business Purpose:

Improves the **readability of downtime metrics** in reports and KPI cards, allowing managers and operators to quickly understand how long production lines were down without converting minutes mentally.

Dependencies:

- **Table:** factLineDowntime
- **Column:** DowntimeMinutes

Logic Breakdown:

1. **SUM:** Aggregates total downtime in minutes from all events.
2. **INT:** Extracts the whole number of hours.
3. **MOD:** Computes the remaining minutes after full hours.
4. **FORMAT:** Converts results into a string like "Hrs Mins".

Measure Name: Total Batches

Formula:

```
Total Batches =
    COUNTROWS(factLineProductivity)
```

Description:

Calculates the **total number of production batches** recorded in the factLineProductivity table. Each row in this fact table represents a distinct production batch, so counting rows provides the total number of completed or logged batches.

Business Purpose:

This measure is used to monitor **production output** over a given period (day, week, month, or year) and forms the foundation for productivity and efficiency analysis.

Dependencies:

- **Table:** factLineProductivity

Logic Breakdown:

1. **COUNTROWS:** Counts every row in the factLineProductivity table, which corresponds to one unique batch record.

Measure Name: Top Downtime Product

Formula:

```
Top Downtime Product =
    VAR Summary =
        ADDCOLUMNS(
            SUMMARIZE(
                'dimProducts',
                'dimProducts'[ProductID]
            ),
            "ProducedQty", CALCULATE(COUNTROWS('factLineProductivity'))
        )
    VAR TopProduct =
        TOPN(
            1,
            Summary,
            [ProducedQty],
            DESC
        )
    VAR Result =
        CALCULATE(
            SELECTEDVALUE('dimProducts'[Product]),
            KEEPFILTERS(TopProduct)
        )
    RETURN
        Result
```

Description:

This measure determines the **product with the highest production frequency** — the one that appears most often in the factLineProductivity table. It's used to highlight the top-performing or most frequently manufactured product, which can help correlate with downtime or performance metrics.

Business Purpose:

- Identifies the **most produced product**, useful for capacity planning, efficiency tracking, and downtime impact analysis.
- Enables **dynamic KPI cards** or **tooltips** that display which product drives the majority of production time or downtime.

Dependencies:

- **factLineProductivity** (for production counts)
- **dimProducts** (for product names and IDs)

Measure Name: Top Downtime Factor

Formula:

```
Top Downtime Factor =
    VAR Summary =
        ADDCOLUMNS(
            SUMMARIZE(
                'dimDowntimeFactors',
                'dimDowntimeFactors'[DowntimeFactorID]
            ),
            "Count", CALCULATE(COUNTROWS('factLineDowntime'))
        )
    VAR TopFactor =
        TOPN(
            1,
            Summary,
            [Count],
            DESC
        )
    VAR Result =
        CALCULATE(
            SELECTEDVALUE('dimDowntimeFactors'[Description]),
            KEEPFILTERS(TopFactor)
        )
    RETURN
        Result
```

Description:

This measure identifies the **downtime factor with the highest frequency of occurrence** in the production line. It dynamically evaluates which downtime reason (from the dimDowntimeFactors table) appears most often in the factLineDowntime table and returns its **description** (e.g., "Power Failure", "Operator Error").

Business Purpose:

Used by production managers and analysts to quickly determine the **primary cause of downtime** over a selected period.
It supports **root cause analysis** and enables the team to prioritize maintenance, training, or process improvements based on the most common issue.

Dependencies:

- **Dimension Table:** dimDowntimeFactors
- **Fact Table:** factLineDowntime

Logic Breakdown:

- **SUMMARIZE:** Groups the downtime data by each unique Downtime Factor ID.
- **ADDCOLUMNS:** Adds a calculated column called "Count" that counts how many downtime records exist for each factor.
- **TOPN(1, ..., DESC):** Selects the factor with the highest count — the top downtime cause.
- **SELECTEDVALUE:** Returns the descriptive name of that top downtime factor.

Measure Name: Shortest Batch Time (Min)

Formula:

```
Shortest Batch Time (Min) =
    MIN(factLineProductivity[Duration(Min)])
```

Description:

This measure calculates the **minimum production duration** (in minutes) among all recorded batches in the factLineProductivity table. It represents the **fastest or most efficient batch** completed within the selected period or filter context.

Business Purpose:

Used to **evaluate peak operational efficiency** by identifying the shortest time taken to produce a batch. It helps engineers and managers benchmark performance, detect outliers, and study conditions that led to optimal production speed.

Dependencies:

- **Fact Table:** factLineProductivity
- **Column Used:** [Duration(Min)] (calculated as the difference between StartDateTime and EndDateTime)

Logic Breakdown:

- **MIN:** Returns the smallest numerical value from the [Duration(Min)] column, representing the quickest batch duration recorded.
- Automatically respects any filters applied in reports (such as by date, product, or operator).

Measure Name: Overall Production Efficiency

Formula:

```
Overall Production Efficiency =
    DIVIDE(
        [Total Productive Time (Min)],
        [Total Production Time (Min)],
        0
    )
```

Description:

This measure calculates the **ratio of productive time to total production time**, providing a direct measure of how efficiently the production line is operating. It expresses the **percentage of total time** spent on actual productive manufacturing activities (excluding downtime).

Business Purpose:

The measure is a **key performance indicator (KPI)** for evaluating line efficiency and resource utilization. It helps identify productivity bottlenecks and monitor improvement over time. A higher value indicates more efficient production with minimal downtime.

Dependencies:

- **Measure:** [Total Productive Time (Min)]
- **Measure:** [Total Production Time (Min)]
- **Fact Table:** factLineProductivity
- **Fact Table:** factLineDowntime

Logic Breakdown:

- **DIVIDE:** Safely divides two measures while avoiding division-by-zero errors.
- **Numerator:** [Total Productive Time (Min)] — total time actually used for production.
- **Denominator:** [Total Production Time (Min)] — total scheduled time including downtime.
- **Return Value (0):** Default output when denominator equals zero.

Measure Name: Longest Batch Time (Min)

Formula:

```
Longest Batch Time (Min) =
    MAX(factLineProductivity[Duration(Min)])
```

Description:

This measure returns the **maximum production duration (in minutes)** among all recorded batches in the factLineProductivity table. It identifies the **longest-running batch**, which can help highlight inefficiencies or variations in production time.

Business Purpose:

Used to analyze **production performance consistency** and detect anomalies. By comparing the longest batch time with the average and shortest batch durations, management can identify process delays, machine slowdowns, or specific product types that require longer production times.

Dependencies:

- **Table:** factLineProductivity
- **Column:** [Duration(Min)]

Logic Breakdown:

- **MAX:** Finds the largest numeric value in the [Duration(Min)] column.
- Each batch record in factLineProductivity includes its duration, so this function identifies the single batch with the **maximum processing time**.

Measure Name: Downtime for Mac (Min)

Formula:

```
Downtime for Mac (Min) =
    VAR DeeBatches =
        SELECTCOLUMNS(
            FILTER(
                factLineProductivity,
                RELATED(dimOperatorName[Operator]) = "Mac"
            ),
            "Batch", factLineProductivity[Batch]
        )
    RETURN
        CALCULATE(
            SUM(factLineDowntime[DowntimeMinutes]),
            FILTER(
                factLineDowntime,
                factLineDowntime[Batch] IN DeeBatches
            )
        )
```

Description:

This measure calculates the **total downtime (in minutes)** specifically for the operator **"Mac."** It identifies all production batches handled by Mac and then sums the corresponding downtime recorded in the factLineDowntime table.

Business Purpose:

Used to **evaluate operator-specific performance** and downtime contribution. Helps supervisors identify if particular operators experience higher downtime, which could indicate the need for **training**, **process adjustments**, or **equipment maintenance.**

Dependencies:

- **Fact Tables:**
  - factLineProductivity
  - factLineDowntime
- **Dimension Table:**
  - dimOperatorName
- **Columns Used:**
  - factLineProductivity[Batch]

- factLineDowntime[DowntimeMinutes]
- dimOperatorName[Operator]

Logic Breakdown:

1. **FILTER + RELATED:** Selects all batches from factLineProductivity where the operator's name (from dimOperatorName) equals "Mac".
2. **SELECTCOLUMNS:** Extracts only the Batch column for those records.
3. **CALCULATE + SUM:** Sums the downtime minutes from factLineDowntime for all batches identified as belonging to Mac.
4. **FILTER (factLineDowntime):** Restricts the downtime calculation only to matching batch numbers.

Measure Name: Downtime for Mac

Formula:

```
Downtime for Mac =
    VAR TotalMinutes = [Downtime for Mac (Min)]
    VAR Hours = INT(DIVIDE(TotalMinutes, 60))
    VAR Minutes = MOD(TotalMinutes, 60)
    RETURN
        FORMAT(Hours, "0") & " Hrs " &
        FORMAT(Minutes, "00") & " Mins"
```

Description:

This measure converts the total downtime for operator **"Mac"**—previously calculated in minutes by the measure **[Downtime for Mac (Min)]**—into a **readable time format** of hours and minutes (e.g., *2 Hrs 30 Mins*). It's designed for display in Power BI visuals or dashboards to improve readability for end users.

Business Purpose:

Helps stakeholders and supervisors quickly understand downtime duration for **operator Mac** in a **human-friendly format**, making performance reports more interpretable without requiring manual conversion from minutes to hours.

Dependencies:

- **Measure:**
  - [Downtime for Mac (Min)]
- **No direct table dependency**, as it references the precomputed measure above.

Logic Breakdown:

1. **Retrieve Total Minutes:** Gets total downtime minutes from the existing [Downtime for Mac (Min)] measure.
2. **Convert to Hours and Minutes:**
    - INT(DIVIDE(...)) extracts the full hours.
    - MOD(..., 60) gets the remainder (minutes).
3. **Format Output:** Combines both components into a string with clear labeling — "Hrs" and "Mins" — for clean reporting visuals.

Measure Name: Downtime for Dennis (Min)

Formula:

```
Downtime for Dennis (Min) =
    VAR DeeBatches =
        SELECTCOLUMNS(
            FILTER(
                factLineProductivity,
                RELATED(dimOperatorName[Operator]) = "Dennis"
            ),
            "Batch", factLineProductivity[Batch]
        )
    RETURN
        CALCULATE(
            SUM(factLineDowntime[DowntimeMinutes]),
            FILTER(
                factLineDowntime,
                factLineDowntime[Batch] IN DeeBatches
            )
        )
```

Description:

This measure calculates the **total downtime duration (in minutes)** associated with the operator **"Dennis."** It identifies all production batches handled by Dennis from the factLineProductivity table, then sums the corresponding downtime minutes from the factLineDowntime table for those same batches.

Business Purpose:

This measure enables targeted operator performance analysis. By isolating downtime data for **Dennis**, production managers can evaluate individual efficiency, identify improvement opportunities, and monitor how specific operators impact overall productivity.

Dependencies:

- **Tables:**
  - factLineProductivity — to identify Dennis's batches.
  - factLineDowntime — to aggregate downtime minutes.
  - dimOperatorName — to map each batch to its corresponding operator.

Logic Breakdown:

1. **Create Batch List (DeeBatches):** Uses FILTER and RELATED to retrieve all batches from factLineProductivity where the operator equals "Dennis."
2. **Calculate Downtime:** CALCULATE sums DowntimeMinutes from factLineDowntime, restricted to only those batches in DeeBatches.
3. **Return Result:** Outputs the total downtime in **minutes** for all batches handled by Dennis.

Measure Name: Downtime for Dennis

Formula:

```
Downtime for Dennis =
    VAR TotalMinutes = [Downtime for Dennis (Min)]
    VAR Hours = INT(DIVIDE(
        TotalMinutes,
        60
    ))
    VAR Minutes = MOD(
        TotalMinutes,
        60
    )
    RETURN
        FORMAT(
            Hours,
            "0"
        ) & " Hrs " & FORMAT(
            Minutes,
            "00"
        ) & " Mins"
```

Description:

This measure converts the total downtime duration (in minutes) associated with the operator **"Dennis"** into a formatted time string that displays **hours and minutes**. It references the [Downtime for Dennis (Min)] measure, which calculates the total downtime in minutes, and then breaks that value into hours and remaining minutes for improved readability in reports.

Business Purpose:

This measure provides a **human-readable representation** of downtime for a specific operator. By showing hours and minutes instead of a single numeric value, it enhances **dashboard clarity** and allows managers to more intuitively compare operator performance or understand total downtime contributions during analysis.

Dependencies:

Measures:

- [Downtime for Dennis (Min)] — provides the total downtime in minutes for Dennis.

Tables:

- factLineProductivity — identifies production batches handled by Dennis.
- factLineDowntime — stores downtime duration data.
- dimOperatorName — maps batches to operators.

Logic Breakdown:

1. **Retrieve Total Downtime:** The measure begins by calling [Downtime for Dennis (Min)], which returns total downtime in minutes.
2. **Convert to Hours:** Uses INT(DIVIDE(TotalMinutes, 60)) to calculate the whole number of hours.
3. **Get Remaining Minutes:** Uses MOD(TotalMinutes, 60) to find leftover minutes after dividing by 60.
4. **Format Output:** Combines both values into a string formatted as: "X Hrs YY Mins" (e.g., 2 Hrs 15 Mins).

Measure Name: Downtime for Dee (Min)

Formula:

```
Downtime for Dee (Min) =
    VAR DeeBatches =
        SELECTCOLUMNS(
            FILTER(
                factLineProductivity,
                RELATED(dimOperatorName[Operator]) = "Dee"
            ),
            "Batch", factLineProductivity[Batch]
        )
    RETURN
        CALCULATE(
            SUM(factLineDowntime[DowntimeMinutes]),
            FILTER(
                factLineDowntime,
                factLineDowntime[Batch] IN DeeBatches
            )
        )
```

Description:

This measure calculates the **total downtime duration (in minutes)** for the operator **"Dee."** It first identifies all production batches managed by Dee from the factLineProductivity table and then sums the corresponding downtime minutes from the factLineDowntime table for those batches.

Business Purpose:

The measure isolates **operator-specific downtime** to assess individual performance. By focusing on Dee's downtime, managers can analyze productivity bottlenecks, identify recurring issues, and track how each operator impacts the overall line efficiency.

Dependencies:

Tables:

- factLineProductivity — used to determine batches handled by Dee.
- factLineDowntime — provides downtime minutes for each batch.
- dimOperatorName — links each production batch to its respective operator.

Logic Breakdown:

1. **Identify Dee's Batches (DeeBatches):** Uses FILTER and RELATED to extract all batch numbers from factLineProductivity where the operator equals "Dee."
2. **Calculate Total Downtime:** The CALCULATE function sums up the DowntimeMinutes from factLineDowntime but filters only the batches that appear in DeeBatches.
3. **Return Result:** Outputs the total downtime in minutes for all batches associated with Dee.

Measure Name: Downtime for Dee

Formula:

```
Downtime for Dee =
    VAR TotalMinutes = [Downtime for Dee (Min)]
    VAR Hours = INT(DIVIDE(
        TotalMinutes,
        60
    ))
    VAR Minutes = MOD(
        TotalMinutes,
        60
    )
    RETURN
        FORMAT(
            Hours,
            "0"
        ) & " Hrs " & FORMAT(
            Minutes,
            "00"
        ) & " Mins"
```

Description:

This measure converts the total downtime for the operator **"Dee"** (calculated in minutes by the [Downtime for Dee (Min)] measure) into a **readable time format (hours and minutes)**. It helps display downtime more clearly in dashboards and reports for easier interpretation.

Business Purpose:

Used to enhance the **presentation and readability** of downtime metrics. Instead of showing total downtime as a large number of minutes, this measure expresses it in the **"Hrs : Mins"** format, making operator performance reports more user-friendly and professional.

Dependencies:

Measures:

- [Downtime for Dee (Min)] — provides the total downtime in minutes.

Tables:

- Indirectly depends on factLineProductivity, factLineDowntime, and dimOperatorName through its dependency on [Downtime for Dee (Min)].

Logic Breakdown:

1. **Retrieve Total Minutes:** The variable TotalMinutes pulls the downtime value from [Downtime for Dee (Min)].
2. **Convert to Hours and Minutes:**
    - Hours = total minutes ÷ 60 (integer division).
    - Minutes = remainder of total minutes ÷ 60.
3. **Format Output:** Combines the results into a string formatted as "X Hrs YY Mins" for display.

Measure Name: Downtime for Charlie (Min)

Formula:

```
Downtime for Charlie (Min) =
    VAR DeeBatches =
        SELECTCOLUMNS(
            FILTER(
                factLineProductivity,
                RELATED(dimOperatorName[Operator]) = "Charlie"
            ),
            "Batch", factLineProductivity[Batch]
        )
    RETURN
        CALCULATE(
            SUM(factLineDowntime[DowntimeMinutes]),
            FILTER(
                factLineDowntime,
                factLineDowntime[Batch] IN DeeBatches
            )
        )
```

Description:

This measure calculates the **total downtime duration (in minutes)** associated with the operator **"Charlie."**
It identifies all production batches handled by Charlie in the factLineProductivity table and then sums up the corresponding downtime minutes from the factLineDowntime table for those same batches.

Business Purpose:

This measure supports **operator-specific performance analysis**.
By isolating downtime data for Charlie, it helps managers understand how much of the total downtime is linked to this operator. This insight is valuable for identifying training needs, process inefficiencies, or machine-related issues tied to specific operators.

Dependencies:

Tables:

- factLineProductivity — used to identify the batches managed by Charlie.
- factLineDowntime — provides downtime data linked to each batch.
- dimOperatorName — links operator names to their respective keys.

Logic Breakdown:

1. **Identify Charlie's Batches:**
   The variable DeeBatches uses FILTER and RELATED to collect all batches from factLineProductivity where the operator is "Charlie."
2. **Aggregate Downtime:**
   The CALCULATE function sums DowntimeMinutes from factLineDowntime but only for batches that appear in the DeeBatches list.
3. **Return Total Downtime:**
   The measure outputs the total downtime in **minutes** for all batches handled by Charlie.

Measure Name: Downtime for Charlie

Formula:

```
Downtime for Charlie =
    VAR TotalMinutes = [Downtime for Charlie (Min)]
    VAR Hours = INT(DIVIDE(TotalMinutes, 60))
    VAR Minutes = MOD(TotalMinutes, 60)
    RETURN
        FORMAT(Hours, "0") & " Hrs " &
        FORMAT(Minutes, "00") & " Mins"
```

Description:

This measure converts the **total downtime for Charlie** (in minutes) into a **formatted text value** showing hours and minutes. It enhances report readability by expressing downtime in a more intuitive time format — for example, "1 Hrs 30 Mins" instead of just "90."

Business Purpose:

The formatted output is useful for **dashboard presentation** and **management summaries**, where displaying downtime in hours and minutes helps communicate operational performance more clearly than raw numeric values.

Dependencies:

Measures:

- [Downtime for Charlie (Min)] — provides the total downtime (in minutes) for Charlie, which this measure formats.

Tables:

- factLineProductivity (indirectly through the dependent measure)
- factLineDowntime
- dimOperatorName

Logic Breakdown:

1. **Retrieve Total Downtime:** Uses the previously calculated measure [Downtime for Charlie (Min)] to get total downtime in minutes.
2. **Convert Minutes to Hours and Remaining Minutes:**
    - Hours = integer division of total minutes by 60.
    - Minutes = remainder after dividing total minutes by 60.

3. **Format the Output:** Combines both values into a text string formatted as **"X Hrs YY Mins."**

Measure Name: Average Production Time (Min)

Formula:

```
Average Production Time (Min) =
    AVERAGE(factLineProductivity[Duration(Min)])
```

Description:

This measure calculates the **average duration (in minutes)** of production batches by taking the mean of all batch times recorded in the factLineProductivity table. It provides a clear view of the **typical production cycle length** for a given dataset or time period.

Business Purpose:

This measure helps production managers and analysts to:

- Monitor **process consistency** and **efficiency trends** over time.
- Identify **deviations** from standard production duration targets.
- Benchmark **operator or machine performance** by comparing average batch times across different categories.

Dependencies:

Table:

- factLineProductivity

Column:

- Duration(Min) — represents the time taken for each production batch in minutes.

Logic Breakdown:

1. **Data Source:** Uses the Duration(Min) column, which stores each batch's total production time.
2. **Aggregation Function:** The AVERAGE() DAX function computes the arithmetic mean of all available durations.
3. **Output:** Returns a single numeric value representing the average production duration in minutes.

Measure Name: Average Downtime (Min)

Formula:

```
Average Downtime (Min) =
    AVERAGE(factLineDowntime[DowntimeMinutes])
```

Description:

This measure calculates the **average downtime duration (in minutes)** across all recorded downtime events in the factLineDowntime table. It shows the **typical length of downtime occurrences**, helping assess how long production is usually interrupted.

Business Purpose:

This measure is essential for evaluating **operational efficiency** and identifying **bottlenecks** in production.
It allows managers to:

- Track **average delay per event** over time.
- Measure **improvement or degradation** in maintenance response.
- Set realistic **downtime reduction goals** for the production line.

Dependencies:

Table:

- factLineDowntime

Column:

- DowntimeMinutes — represents the duration of each downtime event in minutes.

Logic Breakdown:

1. **Data Source:** Pulls all downtime durations from the DowntimeMinutes column in factLineDowntime.
2. **Aggregation:** Uses the AVERAGE() DAX function to compute the arithmetic mean of downtime durations.
3. **Output:** Returns a single numeric value representing the **average downtime per event** in minutes.

Measure Name: % Batches Meeting Target

Formula:

```
% Batches Meeting Target =
    VAR TotalBatches =
        COUNTROWS(factLineProductivity)
    VAR BatchesMetTarget =
        COALESCE(
            COUNTROWS(
                FILTER(
                    factLineProductivity,
                    VAR ActualTime = factLineProductivity[Duration(Min)]
                    VAR TargetTime = RELATED(dimProducts[BatchTime(Min)])
                    RETURN
                        NOT ISBLANK(TargetTime)
                        && ActualTime <= TargetTime
                )
            ),
            0
        )
    RETURN
        DIVIDE(
            BatchesMetTarget,
            TotalBatches,
            0
        )
```

Description:

This measure calculates the **percentage of production batches** that were completed **on or below the target batch time** defined in the product dimension (dimProducts). It compares each batch's actual production duration (Duration(Min)) against the predefined target duration (BatchTime(Min)) and expresses the proportion that met the target as a percentage.

Business Purpose:

This KPI is vital for measuring **production efficiency and performance consistency**. It helps production managers and supervisors:

- Track **how often production meets expectations**.
- Identify **process inefficiencies** or **delays**.
- Assess whether **productivity targets are realistic** or need adjustment.

Higher percentages indicate better adherence to planned production schedules.

Dependencies:

Tables:

- factLineProductivity — contains actual batch duration data.
- dimProducts — stores target batch times for each product.

Columns:

- factLineProductivity[Duration(Min)] — actual time per batch.
- dimProducts[BatchTime(Min)] — target time per batch.

Logic Breakdown:

1. **Calculate Total Batches:** COUNTROWS(factLineProductivity) counts all batches recorded in the productivity fact table.
2. **Identify Batches Meeting Target:**
    - FILTER() iterates through factLineProductivity.
    - For each row, compares **actual duration** with **target duration** (from dimProducts).
    - Includes only batches where ActualTime <= TargetTime.
3. **Handle Missing Data:** COALESCE() ensures the count defaults to 0 if no matching rows exist.
4. **Compute Percentage:** DIVIDE() calculates (BatchesMetTarget / TotalBatches) safely, returning 0 if the denominator is zero.

Measure Name: % Batches Not Meeting Target

Formula:

```
% Batches Not Meeting Target =
VAR TotalBatches =
    COUNTROWS(factLineProductivity)
VAR BatchesNotMetTarget =
    COALESCE(
        COUNTROWS(
            FILTER(
                factLineProductivity,
                VAR ActualTime = factLineProductivity[Duration(Min)]
                VAR TargetTime = RELATED(dimProducts[BatchTime(Min)])
                RETURN
                    NOT ISBLANK(TargetTime)
                    && ActualTime > TargetTime
            )
        ),
        0
    )
RETURN
    DIVIDE(
        BatchesNotMetTarget,
        TotalBatches,
        0
    )
```

Description:

This measure calculates the **percentage of production batches that exceeded their target batch time**. It compares each batch's actual production duration (Duration(Min)) against the predefined target time from the **dimProducts** table (BatchTime(Min)). Only batches with a valid target time are included in the calculation. The final result shows how many batches did **not meet their performance target**, helping identify process inefficiencies and improvement areas.

Usage Example:

Used in performance dashboards to complement % Batches Meeting Target, allowing supervisors to quickly visualize operational deviations or delays across production lines.

Table: Tables_Info

Formula:

```
Tables_Info = INFO.VIEW.TABLES()
```

Description:

This DAX expression retrieves **detailed metadata information about all tables** in the current Power BI data model. It displays structural, visibility, and storage-related properties of each table, helping developers and modelers analyze and document their data models directly through DAX.

Table: Columns_Info

Formula:

```
Columns_Info = INFO.VIEW.COLUMNS()
```

Description:

This DAX measure retrieves **detailed metadata information about every column** within all tables in the Power BI data model. It provides visibility into column names, data types, table associations, storage mode, visibility status, and other attributes. This function is part of Power BI's new **INFO.VIEW** family, designed to help developers inspect and document their models directly from DAX.

Table: Measures_Info

Formula:

```
Measures_Info = INFO.VIEW.MEASURES()
```

Description:

This DAX expression retrieves a **comprehensive metadata table of all measures** defined in the Power BI data model. It provides detailed information such as the measure name, table location, expression, formatting, display folder, and visibility status. This is especially useful for documentation, auditing, or maintaining large data models with many DAX measures.

Table: Relations_Info

Formula:

```
Relations_Info = INFO.VIEW.RELATIONSHIPS()
```

Description:

This DAX expression returns a **complete metadata view of all relationships** defined within the Power BI data model. It lists every relationship between tables, including the columns that form the relationship, the cardinality (one-to-many, many-to-one, etc.), and the active/inactive status. This is an essential tool for documenting or auditing the data model structure.

Data Modeling

Data Model

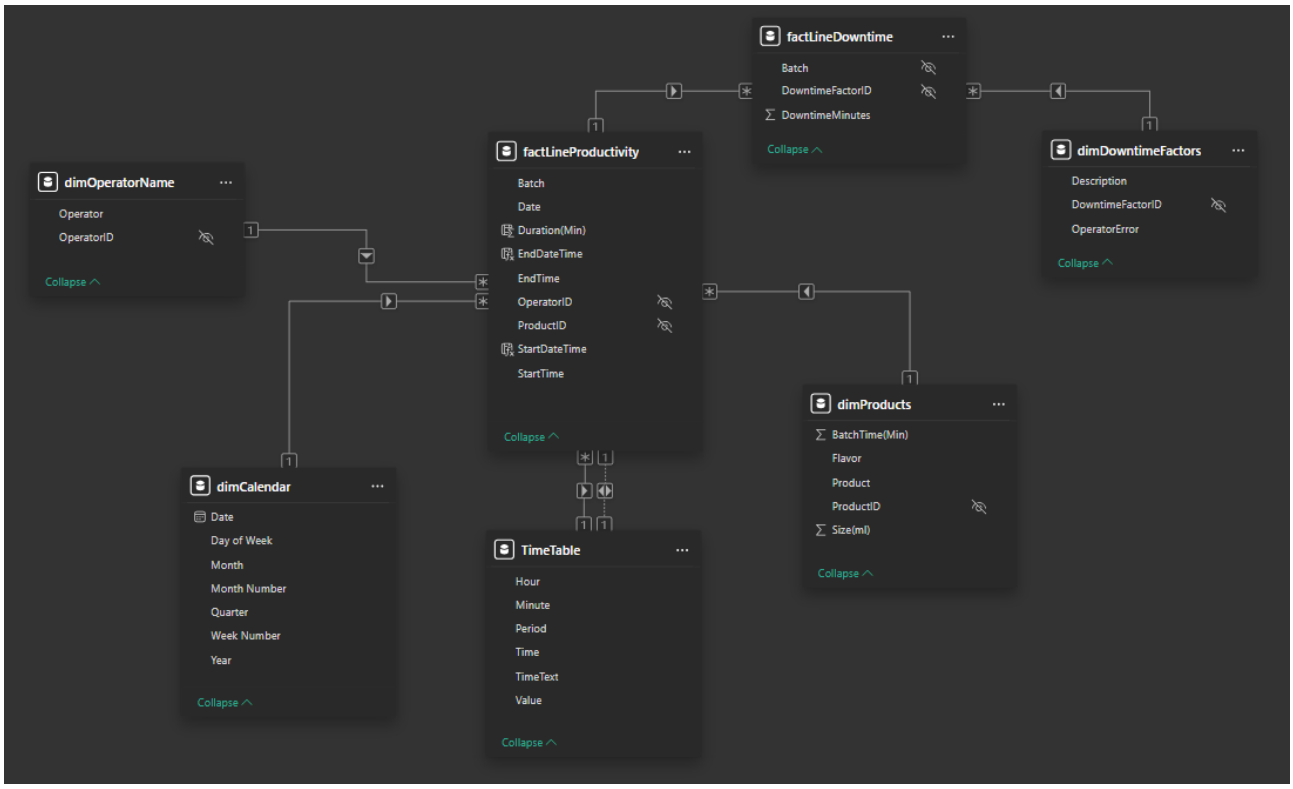


Figure 1: Data model

Data Model Description

Model Overview

The Power BI data model integrates production and downtime information to evaluate overall manufacturing performance. It follows a **star schema**, centered around two main fact tables — **factLineProductivity** and **factLineDowntime** — that connect to several descriptive dimension tables. This design allows for flexible analysis by product, operator, date, and time, supporting KPIs like *Average Production Time*, *Average Downtime*, and *Operator Efficiency*.

Meta data

1. Table: factLineProductivity

Description: Stores production event data including duration, batch, operator, and product. It acts as the primary fact table for measuring efficiency.

Column Name	Data Type	Description
Date	Date	The date of the production run[2].
ProductID	Whole Number (Int64)	Foreign key linking to dimProducts. Used to identify the product being manufactured[3].
Batch	Text	Unique identifier for the production batch. Converted to text to align with other tables[4].
OperatorID	Text	Foreign key linking to dimOperatorName. Identifies the operator responsible for the batch[5].
StartTime	Time	The specific time production started[6].
EndTime	Time	The specific time production ended[7].
StartDateTime	DateTime (Calculated)	Combines Date and StartTime into a single timestamp for accurate duration calculations[8].
EndDateTime	DateTime (Calculated)	Determines the actual end date/time, accounting for operations that cross over midnight (End Time < Start Time)[9].
Duration (Min)	Whole Number (Calculated)	Calculates the total duration of the operation in minutes (Difference between StartDateTime and EndDateTime)[10].

2. Table: factLineDowntime

Description: Captures machine downtime incidents, including the specific factor (reason) and the duration of the stoppage.

Column Name	Data Type	Description
Batch	Text	Foreign key linking to factLineProductivity. Associates the downtime with a specific production batch[12].
DowntimeFactorID	Text	Foreign key linking to dimDowntimeFactors. Represents the category or reason for the downtime[13].
DowntimeMinutes	Whole Number (Int64)	The duration of the downtime event in minutes[14].

3. Table: dimProducts

Description: Contains product master data such as flavor, size, and target batch times. Used for analyzing performance by product type.

Column Name	Data Type	Description
ProductID	Whole Number (Int64)	Unique identifier for the product. Generated via Index in Power Query[16].
Product	Text	The name or code of the product (e.g., CO-600)[17].
Flavor	Text	The flavor variant of the product[18].
Size(ml)	Whole Number (Int64)	The volume size of the product in milliliters (cleaned from text like "2 L" to 2000)[19].
BatchTime (Min)	Whole Number (Int64)	The **Target** minimum time required to produce a batch of this product. Used to calculate if a batch met its target[20].

4. Table: dimOperatorName

Description: A dimension table containing information about machine operators, ensuring standardized names and IDs.

Column Name	Data Type	Description
OperatorID	Text	Unique identifier for the operator[22].
Operator	Text	The full name of the machine operator[23].

5. Table: dimDowntimeFactors

Description: Describes the reasons for downtime (e.g., "Machine Failure") and categorizes whether the issue is related to operator error.

Column Name	Data Type	Description
DowntimeFactorID	Text	Unique identifier for the downtime cause[25].
Description	Text	Descriptive name of the downtime reason (e.g., "Machine adjustment", "Inventory shortage")[26].
OperatorError	Text	A flag ("Yes"/"No") indicating if the downtime factor is attributed to human/operator error[27].

6. Table: dimDate (dimCalendar)

Description: A generated calendar table for time-based analysis (Year, Month, Day). It supports trending metrics over time.

Column Name	Data Type	Description
Date	Date	The specific calendar date[29].
Year	Whole Number	The 4-digit year (e.g., 2025)[30].
Quarter	Text	The quarter of the year (e.g., "Q1")[31].
Week Number	Whole Number	The week number of the year (1-52)[32].
Month Number	Whole Number	The month number (1-12)[33].
Month	Text	The full name of the month (e.g., "January")[34].
Day of Week	Text	The full name of the day (e.g., "Monday")[35].

7. Table: TimeTable

Description: A DAX-generated table with one row per minute (0-1439). It is used to analyze performance by hour and shift.

Column Name	Data Type	Description
Value	Whole Number	Sequential index representing minutes since midnight (0 to 1439)[37].
Time	Time	The actual time value (e.g., 08:30:00)[38].
Hour	Whole Number	The hour component (0-23). Useful for hourly trend analysis[39].
Minute	Whole Number	The minute component (0-59)[40].
Period	Text	Indicator for "AM" or "PM"[41].
TimeText	Text	Readable string format (e.g., "08:30 AM") for slicers[42].
Shift Time	Text (Calculated)	Classifies the time into "Day Shift" (AM) or "Night Shift" (PM)[43].

Relationships Summary

From Table (Foreign Key)	To Table (Primary Key)	Relationship Type
factLineProductivity [OperatorID]	dimOperatorName [OperatorID]	Many-to-One (Single)
factLineProductivity [ProductID]	dimProducts [ProductID]	Many-to-One (Single)
factLineProductivity [Date]	dimDate [Date]	Many-to-One (Single)
factLineProductivity [StartTime]	TimeTable [Time]	Many-to-One (Single)
factLineDowntime [DowntimeFactorID]	dimDowntimeFactors [DowntimeFactorID]	Many-to-One (Single)
factLineDowntime [Batch]	factLineProductivity [Batch]	Many-to-One (Single)

Model Characteristics

- **Schema Type:** Star Schema
- **Primary Relationships:** Based on surrogate keys (IDs and Dates)
- **Cross-Filtering:** Primarily single-direction for performance; bidirectional between downtime and productivity where batch alignment is required
- **Purpose:** To enable combined performance insights, such as total production vs. downtime, per product, operator, and time frame.

Visualization and answering business questions

Wire framing

UI Wireframing and Dashboard Design

Purpose

The wireframe was designed in **Figma** to visualize the structure, layout, and interaction flow of the **Manufacturing Downtime Analysis Dashboard** before implementation in Power BI. This step ensured that all required insights — such as productivity trends, downtime factors, and operator performance — were clearly represented with a user-friendly interface.

Overview

The wireframe defines:

- **Dashboard layout:** Placement of KPI cards, charts, and filters
- **Navigation flow:** Logical grouping of downtime analysis, production metrics, and operator performance
- **Color scheme and hierarchy:** Consistent with professional manufacturing analytics dashboards
- **Data storytelling flow:** From high-level performance (KPIs) down to root cause analysis and downtime trends

Design Tool

- **Tool Used:** Figma
- **Purpose:** Create a visual mockup to align stakeholders on report structure and user experience prior to Power BI development.

Integration with Power BI

The wireframe guided the following design aspects in Power BI:

1. **Page Layout:** Balanced visual hierarchy between KPIs, charts, and filters.
2. **Color Usage:** Consistent use of blue, orange, and gray tones to emphasize status and categories.
3. **Navigation:** Logical flow from overall production efficiency to detailed downtime causes.
4. **User Experience:** Optimized for intuitive interpretation by managers and operators.

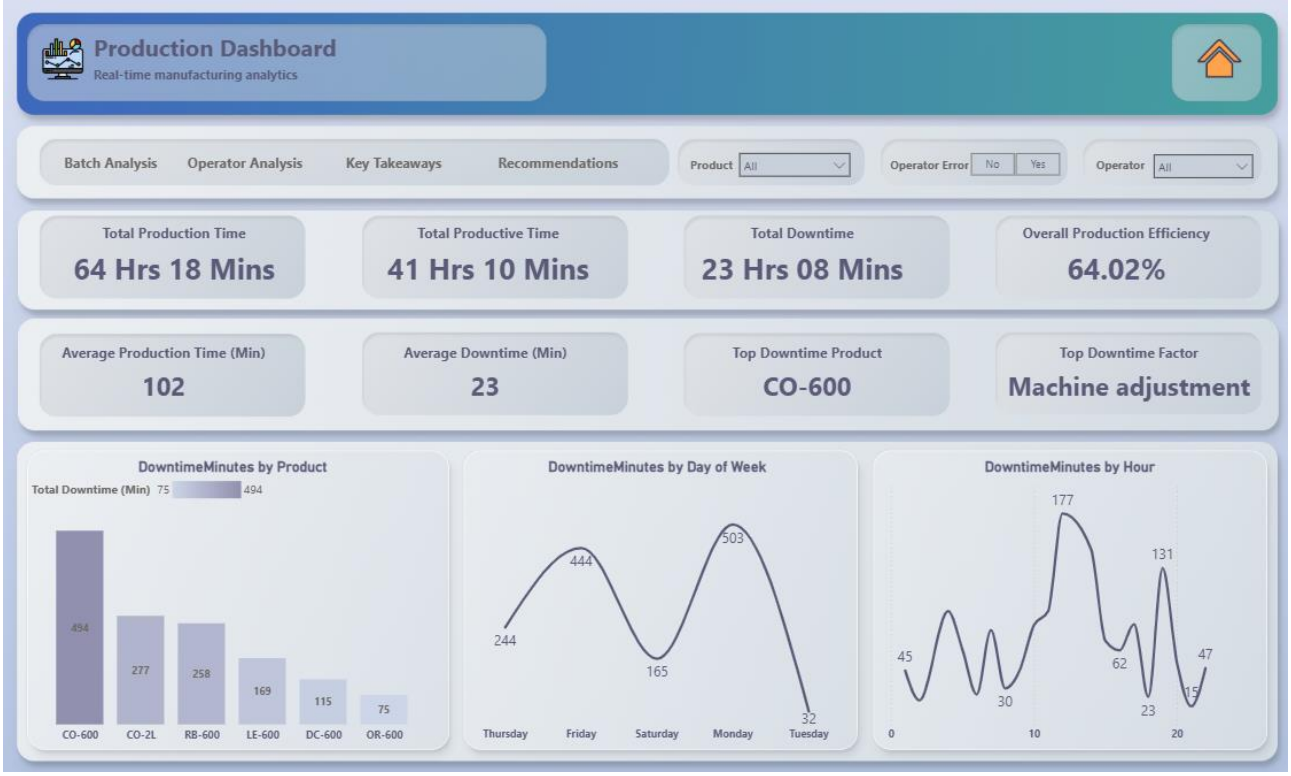
Outcome

The Figma wireframe acted as the **visual blueprint** for building the Power BI dashboard, ensuring:

- Faster report development
- Alignment between data engineers and stakeholders
- A consistent and professional visual identity

Visualizing and answering

Page 1: Overview Analysis



1. How efficiently is the production line operating overall? The production line is operating at **64.02% Overall Production Efficiency**. This indicates that while the majority of time is productive, there is significant room for improvement (approx. 36% inefficiency).

2. What portion of total production time is lost to downtime? Out of a **Total Production Time of 64 Hrs 18 Mins**, the line lost **23 Hrs 08 Mins** to downtime. This confirms that roughly **36%** of the total available time is being lost to stoppages.

3. Which products or factors contribute most to downtime?

- **Top Product:** The product **CO-600** is the biggest offender, causing **494 minutes** of downtime—nearly double the next closest product (CO-2L at 277 mins).
- **Top Factor:** The primary reason for these stoppages is **"Machine adjustment,"** suggesting that changeovers or calibration issues are the main bottleneck.

4. How does downtime trend over time or by hour of the day?

- **By Day:** Downtime is highly volatile. **Monday is the worst day** (503 mins) followed closely by Friday (444 mins). Tuesday is the best day (32 mins).

- **By Hour:** There is a massive spike in downtime around the **11th–12th hour** (177 mins) and another spike around the **20th hour (8 PM)** (131 mins).

5. Are there patterns indicating when or where production is most efficient?

- **When:** Production is most efficient on **Tuesdays** and generally during the very early hours (0–5) or late evening dips (around hour 19 and 21), where downtime drops to as low as 15–23 minutes.
- **Where (Product):** Production runs involving **OR-600** and **DC-600** are the most efficient, accumulating the least downtime (75 and 115 minutes respectively).

Page 2: Batch Analysis

Based on the **Page 2: Batch Analysis** dashboard, here are the answers to your questions. This page reveals a critical consistency issue in your production process.

Page 2: Batch Analysis

1. Are batches being completed within expected target times? No, significantly far from it. Only **7.89%** of batches are meeting their target times. A massive **92.11%** of batches are failing to meet production targets, indicating a systemic issue rather than isolated incidents.

2. Which batches take the longest or shortest time to produce, and why?

- **Longest Batch:** The longest batch took **205 minutes**.
- **Shortest Batch:** The fastest batch took **60 minutes**.
- **Why:** The scatter plot (*Production Time vs Total Downtime*) shows a direct link: the batches that take the longest are simply the ones suffering the most downtime. The longest batch (205 mins) had over **100 minutes of downtime**, effectively doubling its production time.

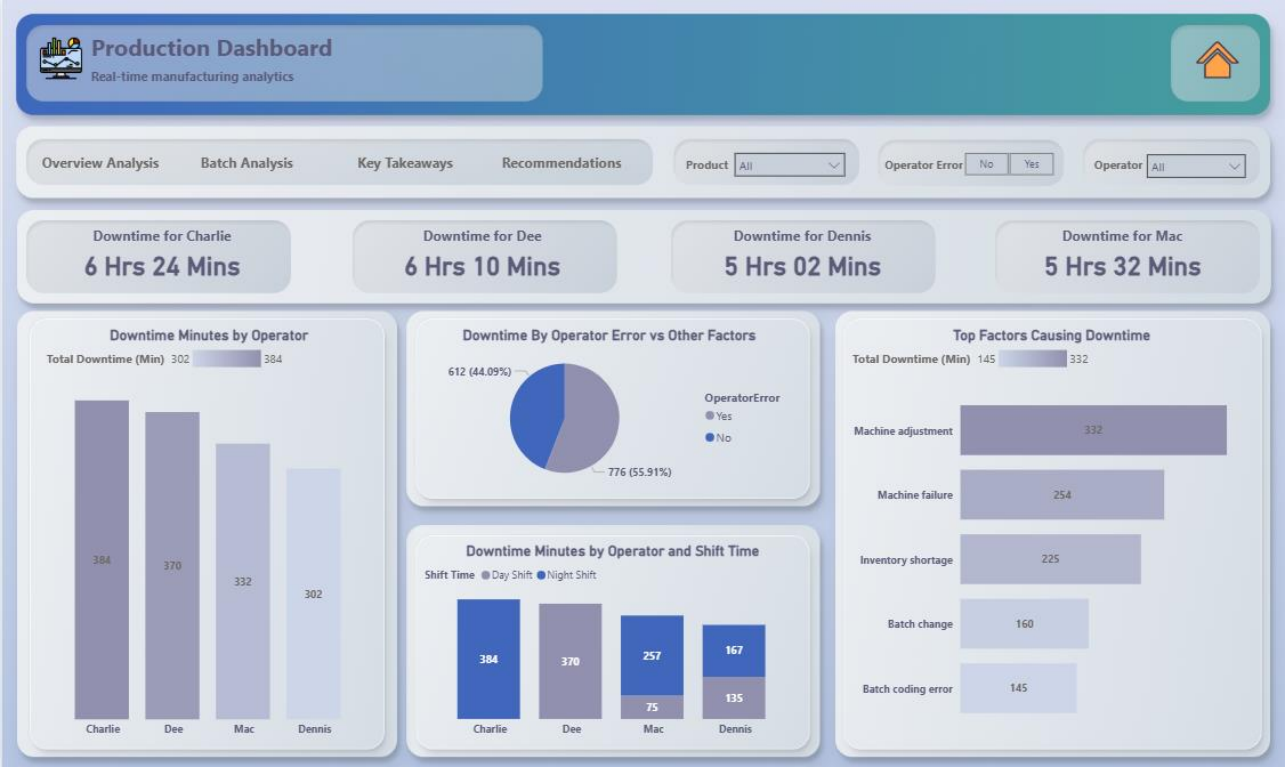
3. What downtime causes appear most frequently across batches? According to the breakdown matrix, the top three causes driving these delays are:

1. **Machine adjustment:** 332 minutes (The dominant issue).
2. **Machine failure:** 254 minutes.
3. **Inventory shortage:** 225 minutes. *Note: "Inventory shortage" is a new insight here; it suggests that logistics/supply chain issues are stopping the line almost as often as mechanical failures.*

4. How strong is the relationship between total batch time and downtime? Very strong. The scatter plot shows a clear linear progression. As downtime increases, total production time increases proportionally. This implies that your actual "run speed" is likely consistent, and almost all variance in batch duration is caused by these stoppages.

5. Which products or operators are linked to slower batch performance? While this specific visual focuses on **Batch IDs** (e.g., Batch 422111 had significant "Batch change" issues), it does not explicitly list operator names in the matrix. However, it does highlight that **Inventory Shortage** and **Machine Adjustment** are the links to slower performance. To name specific operators, we will likely need to look at the next page.

Page 3: Operator Analysis



1. Which operators are the most and least efficient?

- **Least Efficient: Charlie** is contributing the most to downtime with **384 minutes** (6 Hrs 24 Mins), followed very closely by **Dee** (370 minutes).
- **Most Efficient: Dennis** is currently the best performer with the lowest downtime of **302 minutes**, followed by Mac (332 minutes).

2. How much downtime is linked to operator errors or manual intervention?

- **Significantly High.** According to the pie chart ("Downtime By Operator Error"), **55.91% (776 minutes)** of all downtime is explicitly linked to Operator Error.
- This is a major finding. It suggests that more than half of your lost production time is a training or behavioral issue, rather than purely mechanical failure.

3. What is the top downtime causes across operators?

- **#1 Machine Adjustment (332 mins):** This aligns with the "Operator Error" statistic. Adjustments are often manual processes; if operators aren't trained to do them quickly, downtime spikes.

- **#2 Machine Failure (254 mins):** Mechanical breakdown.
- **#3 Inventory Shortage (225 mins):** Operators waiting for materials.

4. How does performance vary across different shifts? The "Downtime Minutes by Operator and Shift Time" chart reveals a stark pattern:

- **Charlie** works exclusively on the **Night Shift** (Blue bar) and has the highest downtime.
- **Dee** works exclusively on the **Day Shift** (Grey bar) and has the second-highest downtime.
- **Mac and Dennis** work a mix of both shifts and perform better overall.
- **Insight:** The issue isn't necessarily "Day vs. Night"; rather, the specific full-time operators on those shifts (Charlie and Dee) are struggling more than the operators who rotate (Mac and Dennis).

5. Are certain operators more efficient with specific products or time periods?

- **Time Periods:** As noted above, the rotational operators (Mac/Dennis) are more efficient than the fixed-shift operators.
- **Products:** While this specific view doesn't break down operators by *product* (the slicer is set to "All"), we can infer from Page 1 that since **CO-600** is the highest downtime product, Charlie and Dee are likely the ones struggling most with the **Machine Adjustments** required for that specific product line.

Recommendations and next steps

Now that we have analyzed all three pages, here is the complete story of your production downtime:

1. **The Core Problem:** Your line efficiency is low (64%) because **92% of your batches miss their target times**.
2. **The Root Cause:** The primary bottleneck is **Machine Adjustment** (Changeovers/Calibration) and **Operator Error** (56% of downtime).
3. **The "Who": Charlie (Night)** and **Dee (Day)** are struggling the most. They are likely taking too long to perform machine adjustments on product **CO-600**.
4. **The "Hidden" Issue: Inventory Shortages** are the 3rd biggest reason for stops—your operators are often just waiting for parts/product.

Next Steps:

- **Training:** Focus immediate retraining on Charlie and Dee specifically regarding *Machine Adjustments for CO-600*.
- **SOP Review:** Create a standardized checklist for "Machine Adjustment" to reduce the 332 minutes of lost time.
- **Logistics:** Investigate why "Inventory Shortage" is causing 225 minutes of downtime. Is the warehouse not restocking the line fast enough?

Deployment & Execution

Prerequisites:

- Microsoft Power BI Desktop (Latest Version).
- Source Excel File: Manufacturing_Line_Productivity.xlsx.

Execution Steps:

1. Open Manufacturing Analysis.pbix.
2. Navigate to "Transform Data" -> "Data Source Settings" to update the file path to the local Excel file location.
3. Click "Refresh" to load the latest data.
4. Interact with the dashboard via the "Home" navigation button.