



# Capstone Project

## MANUFACTURING **DOWNTIME** ANALYSIS

Guidebook

**CREATED BY:**

CLS-CA13-DAT2-G5-T1

**SUPERVISED BY:**

Mahmoud Seraj

**TECHNICAL PARTNER:**

Computik Learning Solutions - CLS

Digital Egypt Pioneers Initiative

# Capstone Project

# **MANUFACTURING DOWNTIME ANALYSIS**

Guidebook

CREATED BY:

CLS-CAI3-DAT2-G5-T1

## About the Authors: A Mosaic of Expertise

We are a team of six brought together by the Digital Egypt Pioneers Initiative (DEPI), each contributing unique professional and academic backgrounds. Our diversity - from MIS and computer science to agriculture, chemistry, sales analysis, and procurement - forms the foundation of our strength.

### Technical Growth & New Perspectives

Sohaib, Hamza, and Kareem represent the technical core of the team, transitioning from MIS, agriculture, and computer science into the world of data analytics, each bringing curiosity, adaptability, and hands-on analytical experience.

### Industry Expertise & Strategic Insight

Mohamed Ali, Mohamed Ezzat, and Hussein Habashi add decades of industry knowledge in sales reporting, procurement, and marketing, enriching the project with real-world business understanding and strategic decision-making experience.

### One Shared Mission

Despite our different paths, we are united by a single goal: mastering data analytics and applying it to solve real industrial challenges.



**Hamza Bahgat**

hamzabahgat53@gmail.com



**Hussien Habashy**

hussainahabashy@gmail.com



**Kareem AbdelHamied**

kareemabdelhamied97@gmail.com



**Mohammed Ali**

m.sarky@gmail.com



**Mohammed Ezzat**

mohamedazzat@gmail.com



**Sohaib Atef**

sohaib.aatef@gmail.com

## Acknowledgements

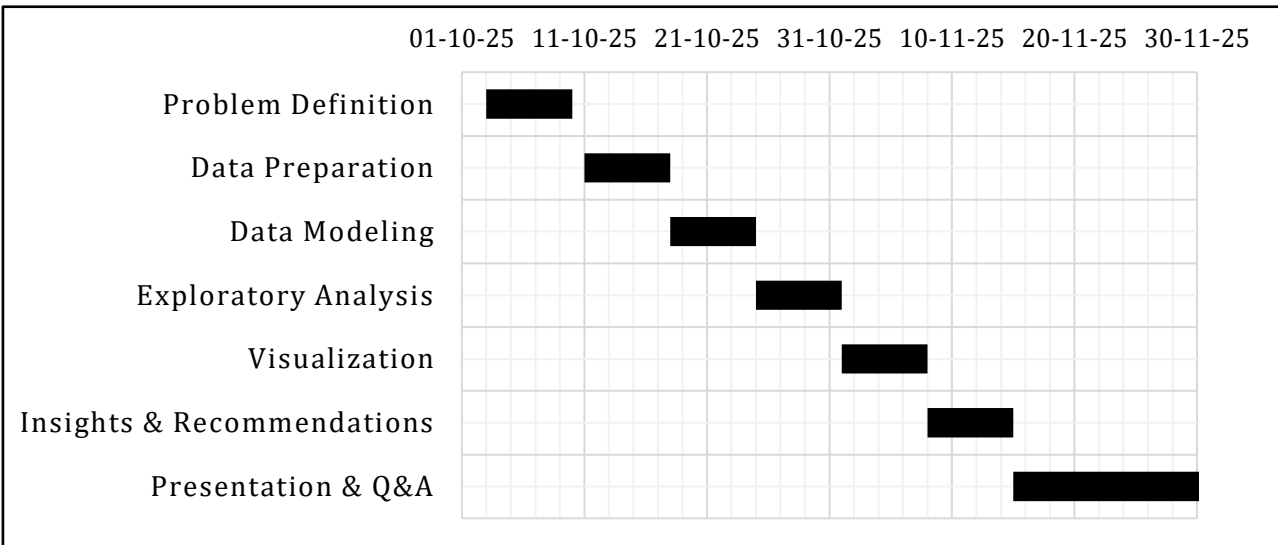
We would like to express our deepest gratitude to everyone who contributed to the success of this project.

First and foremost, we extend our sincere appreciation to **H.E. President Abdel Fattah El-Sisi** for launching the ambitious national initiatives aiming to develop the technology sector in Egypt and empower the youth. We also thank the **Ministry of Communications and Information Technology (MCIT)**, led by **Dr. Amr Talaat**, for their continuous efforts in fostering digital transformation through the **Digital Egypt Pioneers Initiative (DEPI)**.

We are also grateful to **CLS Learning Solutions** for providing us with a professional learning environment and the necessary resources to complete this analysis. A special vote of thanks goes to our mentor and instructor, **Eng. Mahmoud Seraj**, whose guidance, technical expertise, and constructive feedback were instrumental in mastering Power BI.

Finally, our heartfelt thanks go to our **families and friends**. Your unwavering support, patience, and encouragement provided the motivation we needed to push through the challenges and reach this milestone.

# Project Timeline Summary (October–November 2025)



The project follows a structured two-month timeline starting in early October and extending through the end of November. It begins with defining the problem and conducting research, followed by data collection, cleaning, and modeling. After establishing a validated data model, exploratory analysis is carried out to uncover initial patterns.

The visualization phase spans early to mid-November, where the dashboard is designed, implemented, and refined. Insights and recommendations are then developed and validated with the team.

The final week of November is dedicated to preparing the presentation, rehearsing, handling expected Q&A and delivering the final project.

This initiative is offered by the Ministry of Communications and Information Technology under the name *Digital Egypt Pioneers Initiative (DEPI)*, in collaboration with private training partners such as CLS Learning Solutions, Leading 365, Berlitz, and Elharefa. Launched in 2023, the program aims to equip students for the freelance and digital job market, providing specialized tracks in technical skills, soft skills, freelancing, and English.



# Contents

**Introduction..... vi**

**Chapter 1: Data Preparation..... 1**

**Overview .....2**

**Key Steps in Data Preparation Using Power Query .....2**

        1. Table: Downtime Factors ..... 2

        2. Table: Line Downtime (Fact Table) ..... 3

        3. Table: Products ..... 4

        4. Table: Operators ..... 5

        5. Table: Line Productivity (Fact Table) ..... 6

**Chapter 2: System Analysis & Design .....8**

**1. Database Design & Data Modeling .....9**

        Data Schema Diagram..... 9

**2. UI/UX Design & Prototyping ..... 11**

        2.0 Design Tools & Methodology..... 11

        2.1 Wireframes & Mockups..... 11

        2.3 UI/UX Guidelines..... 14

**3. Dashboard Navigation Guide..... 15**

        3.1 Page 1: Home Page ..... 15

        3.2 Page 2: Downtime Analysis ..... 16

        3.3 Page 3: Operator Performance ..... 17

**Chapter 3: Analysis & Insights..... 18**

**Insights ..... 19**

        1. Downtime Is Significantly Above Normal Operational Limits ..... 19

        2. Machine-related Issues Account for Most Non-operator Downtime, Led By ‘Machine Adjustment’ ..... 19

        3. Operator Errors Are the Main Driver ..... 19

        4. Some Operators Consistently Contribute More Downtime Than Others..... 20

        5. Certain Downtime Causes Are Concentrated on Specific Products ..... 20

**Executive Summary ..... 20**

Key Findings .....	21
Target .....	<b>21</b>
To achieve a 20% increase in line availability:.....	21
Suggested Strategy: .....	21
Recommendations .....	<b>22</b>
1. Operator Performance .....	22
2. Equipment & Maintenance .....	22
3. Product-Specific .....	22
7. Inventory & Material Flow .....	22
8. Continuous Improvement Roadmap.....	22
Conclusion .....	<b>23</b>

# Introduction

## Abstract

This project conducts a rigorous analysis of industrial production line data with the primary aim of monitoring and evaluating 'Downtime.' Utilizing advanced data analytics tools - specifically Power BI - the project focuses on assessing operational efficiency and identifying the root causes of delays. The study was applied to real-world datasets covering a 5-day production cycle, providing a practical model for transforming raw data into actionable insights that support decision-making.

## Objectives

The overarching goal of this project is to transform production records into strategies that minimize waste and boost efficiency. To achieve this, the project focused on several key pillars:

- **Comprehensive Performance Assessment:** Examining production logs to pinpoint gaps in the overall performance of the manufacturing line.
- **Analysis of Human & Technical Variables:** Studying operator productivity alongside machine failures, while correlating human error with production delays to identify specific training needs.
- **Product Efficiency Study:** Determining the time utilization for each Product type and Batch to understand the specific production bottlenecks associated with different items.
- **Data Modeling:** Developing a structured data architecture that allows for repeatable analysis and the visual extraction of Key Performance Indicators (KPIs)

## Scope Coverage

To ensure the accuracy and relevance of the results, the scope of the study was defined to capture all recorded incidents within the observation period. The analytical sample included:

- **Focus Area:** A deep-dive analysis of **1 single production line**.
- **Production Diversity:** Coverage of **6 different products** distributed across **38 distinct batches**.
- **Workforce:** Analysis of shifts covered by **4 different operators**.
- **Incident Volume:** A detailed review of **61 downtime incidents** recorded over **5 working days**.

## Expected Outcomes

Upon completion of this analysis, the project delivers an interactive dashboard that provides:

- Clear visibility into recurring downtime patterns.
- Precise identification of weaknesses within the production process.
- Data-driven recommendations to optimize scheduling and enhance overall productivity.

# Chapter 1: Data Preparation

- ❖ Overview
- ❖ Key Steps in Data Preparation Using Power Query

# Overview

Data preparation is a critical phase in building a reliable and accurate data analysis solution. Before any reporting, modeling, or visualization can occur, the raw data must be cleaned, structured, and transformed to ensure consistency and accuracy. In this project, data preparation was performed using Power Query (M Language), which provides a powerful environment for importing data, cleansing it, and shaping it to support analytical requirements.

In this project, the data preparation work focused on:

- Bringing together data from different production sheets into a single, consistent data model.
- Correcting and standardizing the information, such as dates, times, product names, and operator records.
- Removing errors and inconsistencies that could affect calculations or KPIs.
- Creating clean reference tables for products, downtime factors, and operators.
- Structuring production records so that performance, output, and downtime can be analyzed accurately.

By completing this preparation, we ensured that all metrics, such as *line productivity*, *downtime analysis*, and *operator performance*, are based on trustworthy and well-organized data. This enables faster reporting, clearer insights, and more confident decision-making across the manufacturing process.

# Key Steps in Data Preparation Using Power Query

## Transformation Approach in Power Query

### 1. Table: Downtime Factors

Step	Description
1. Import Source File	<b>Source</b> <code>=Excel.Workbook(File.Contents("...Manufacturing_Line_Productivity.xlsx"), null, true)</code> The query loads the source Excel file containing all manufacturing datasets.
2. Select “Downtime factors” Sheet	<code>#"Downtime factors_Sheet" = Source{[Item="Downtime factors", Kind="Sheet"]}[Data]</code> This step extracts the Downtime factors sheet from the workbook as a raw data table.
3. Promote First Row to Column Headers	<code>#"Promoted Headers" = Table.PromoteHeaders(#"Downtime factors_Sheet", [PromoteAllScalars=true])</code> The first row of the sheet is promoted to become the column headers.
4. Set Data Types	<code>#"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Factor", type text}, {"Description", type text}, {"Operator Error", type text}}),</code>

	Data types for three columns - Factor, Description, and Operator Error - are set to Text.
5. Rename Columns	<b>#"Renamed Columns" = Table.RenameColumns(#"Changed Type",{{"Factor", "FactorID"}}),</b> The column 'Factor' is renamed to 'FactorID' to follow consistent naming conventions and better reflect its role as an identifier in the data model.
6. Standardize Text Format	<b>#"Capitalized Each Word" = Table.TransformColumns(#"Renamed Columns",{{"Description", Text.Proper, type text}})</b> The Description column is transformed to Proper Case (capitalize each word).

**Final Output:** The final output is a well-organized Downtime Factors dimension table that provides clear, unique factor identifiers and descriptions. This dimension enhances the data model by enabling accurate categorization and interpretation of downtime events.

## 2. Table: Line Downtime (Fact Table)

Step	Description
1. Import Source File	<b>Source = Excel.Workbook(File.Contents("...Manufacturing_Line_Productivity.xlsx"), null, true)</b> Loads the manufacturing productivity Excel file into Power Query.
2. Load "Line downtime" Sheet	<b>#"Line downtime_Sheet" = Source[Item="Line downtime",Kind="Sheet"][Data]</b> Extracts the sheet named Line downtime as a raw data table.
3. Promote First Row to Headers	<b>#"Promoted Headers" = Table.PromoteHeaders(#"Line downtime_Sheet", [PromoteAllScalars=true])</b> Converts the first row into column headers to structure the dataset.
4. Apply Initial Data Types	<b>#"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Column1", type any}, {"Downtime factor", Int64.Type}, {"Column3", Int64.Type}, ...})</b> Assigns numeric (Whole Number) data types to all downtime columns to ensure proper calculations.
5. Promote Second Header Row	<b>#"Promoted Headers1" = Table.PromoteHeaders(#"Changed Type", [PromoteAllScalars=true])</b> A second header row is promoted because the sheet contains two embedded header rows. This creates the final proper column structure.
6. Unpivot Downtime Columns	<b>#"Unpivoted Columns" = Table.UnpivotOtherColumns(#"Promoted Headers1", {"Batch"}, "Attribute", "Value")</b> Converts downtime columns from <b>wide</b> format into <b>long</b> format. Each downtime factor becomes a row with: - <b>Batch</b> - <b>Attribute</b> (Factor name) - <b>Value</b> (Downtime minutes).
7. Rename Columns	<b>#"Renamed Columns" = Table.RenameColumns(#"Unpivoted Columns",{{"Attribute", "Factor_ID"}, {"Value", "DownTime(Min)"}})</b> Renames the unpivoted fields to meaningful names: Factor_ID and DownTime(Min).
8. Finalize Data Types	<b>#"Changed Type1" = Table.TransformColumnTypes(#"Renamed Columns",{{"Batch", type text}})</b>

## 4 Chapter 1: Data Preparation

	Ensures the Batch column is correctly stored as Text to avoid numeric misinterpretation.
--	--

**Final Output:** The final output is a fully transformed Line Downtime fact table, structured in a long-format layout to support accurate reporting of downtime durations by batch and factor. The unpivoting step ensures the data is optimized for analytical modeling in Power BI.

### 3. Table: Products

Step	Description
1. Import Source File	<b>Source =</b> <b>Excel.Workbook(File.Contents("...Manufacturing_Line_Productivity.xlsx"),</b> <b>null, true)</b> Loads the Excel file containing the Products sheet into Power Query.
2. Load “Products” Sheet	<b>Products_Sheet = Source{[Item="Products",Kind="Sheet"]}[Data]</b> Extracts the Products sheet as the raw data table.
3. Promote First Row to Headers	<b>#"Promoted Headers" = Table.PromoteHeaders(Products_Sheet,</b> <b>[PromoteAllScalars=true])</b> Converts the first row into the column headers.
4. Apply Initial Data Types	<b>#"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Min batch time", type number}, {"Size", type text}, {"Flavor", type text}, {"Product", type text}})</b> Assigns proper data types for numerical and text columns.
5. Capitalize Flavor Values	<b>#"Capitalized Each Word" = Table.TransformColumns(#"Changed Type",{{"Flavor", Text.Proper, type text}})</b> Normalizes the Flavor column by applying proper capitalization.
6. Rename Size Column	<b>#"Renamed Columns" = Table.RenameColumns(#"Capitalized Each Word",{{"Size", "Size (ml)"}})</b> Improves clarity by renaming <i>Size</i> to <i>Size (ml)</i> .
7. Replace “L” With Numeric Value	<b>#"Replaced Value" = Table.ReplaceValue(#"Renamed Columns", "L", "000", Replacer.ReplaceText, {"Size (ml)"})</b> Converts values like “1 L” into “1000 ml” by replacing “L” with “000”.
8. Remove “ml” Text	<b>#"Replaced Value1" = Table.ReplaceValue(#"Replaced Value", "ml", "", Replacer.ReplaceText, {"Size (ml)"})</b> Removes the " ml" suffix to keep only numerical size values.
9. Convert Size Column to Number	<b>#"Changed Type1" = Table.TransformColumnTypes(#"Replaced Value1",{{"Size (ml)", Int64.Type}})</b> Transforms the Size column to Whole Number for consistency.
10. Rename Columns (Standardization)	<b>#"Renamed Columns1" = Table.RenameColumns(#"Changed Type1",{{"Size (ml)", "Size(ml)"}, {"Product", "ProductID"}})</b> Standardizes field names: Size(ml) and ProductID.
11. Finalize Data Type for Size	<b>#"Changed Type2" = Table.TransformColumnTypes(#"Renamed Columns1",{{"Size(ml)", type number}})</b> Ensures Size(ml) is stored as a numeric type.
12. Create Product Name Column	<b>#"Inserted Merged Column" = Table.AddColumn(#"Changed Type2", "Product Name", each Text.Combine({[Flavor], " ", Text.From([#"Size(ml)"]},</b>

	<b>"en-US"), "ml"}), type text)</b> Creates a dynamic product name by combining Flavor + Size, e.g., <i>Orange 250ml</i> .
--	---

**Final Output:** The final output is a fully standardized Products dimension table, containing clean product attributes and a unified product naming format. This table supports consistent product identification across the data model.

#### 4. Table: Operators

Step	Description
1. Import Source File	<b>Source = Excel.Workbook(File.Contents("...Manufacturing_Line_Productivity.xlsx"), null, true)</b> Loads the Excel workbook containing the Line Productivity data.
2. Load "Line productivity" Sheet	<b>#"Line productivity_Sheet" = Source{[Item="Line productivity",Kind="Sheet"]}[Data]</b> Extracts the <i>Line productivity</i> sheet as the source table.
3. Promote First Row to Headers	<b>#"Promoted Headers" = Table.PromoteHeaders(#"Line productivity_Sheet", [PromoteAllScalars=true])</b> Converts the first row into column headers for proper field naming.
4. Apply Initial Data Types	<b>#"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Start Time", type time}, {"Date", type date}})</b> Assigns the correct data types for Date and Start Time.
5. Remove Unnecessary Column	<b>#"Removed Columns" = Table.RemoveColumns(#"Changed Type",{"Column7"})</b> Eliminates an irrelevant column from the dataset.
6. Apply Additional Data Types	<b>#"Changed Type1" = Table.TransformColumnTypes(#"Removed Columns",{{"Product", type text}, {"Batch", type text}, {"Operator", type text}, {"End Time", type time}})</b> Ensures Product, Batch, Operator, and End Time have the proper formats.
7. Rename Columns (Standardization)	<b>#"Renamed Columns" = Table.RenameColumns(#"Changed Type1",{{"Product", "ProductID"}, {"Batch", "BatchID"}})</b> Renames Product and Batch fields for consistency with data model naming conventions.
8. Keep Only Operator Column	<b>#"Removed Other Columns" = Table.SelectColumns(#"Renamed Columns",{"Operator"})</b> Extracts only the Operator column to build the Operators dimension table.
9. Remove Duplicate Operator Names	<b>#"Removed Duplicates" = Table.Distinct(#"Removed Other Columns")</b> Removes duplicate operator names to generate a unique operator list.
10. Add Index Column (OperatorID)	<b>#"Added Index" = Table.AddIndexColumn(#"Removed Duplicates", "Index", 1, 1, Int64.Type)</b> Create a sequential ID (starting at 1) to uniquely identify each operator.
<b>11. Reorder Columns</b>	<b>#"Reordered Columns" = Table.ReorderColumns(#"Added Index",{"Index", "Operator"})</b> Puts the new OperatorID column first for better readability.

12. Rename Index to OperatorID	<b>#"Renamed Columns1" = Table.RenameColumns(#"Reordered Columns",{{"Index", "OperatorID"}})</b> Renames the index column into OperatorID for use as a key in the data model.
13. Convert OperatorID to Text	<b>#"Changed Type2" = Table.TransformColumnTypes(#"Renamed Columns1",{{"OperatorID", type text}})</b> Ensures OperatorID is stored as Text to match key format in relationships.

**Final Output:** The final output produces a unique and well-structured Operators dimension table, assigning each operator a standardized OperatorID for reliable model relationships.

## 5. Table: Line Productivity (Fact Table)

Step	Description
1. Import Source File	<b>Source = Excel.Workbook(File.Contents("...Manufacturing_Line_Productivity.xlsx"), null, true)</b> Loads the manufacturing productivity workbook into Power Query for processing.
2. Load "Line productivity" Sheet	<b>#"Line productivity_Sheet" = Source{[Item="Line productivity", Kind="Sheet"]}[Data]</b> Extracts the Line productivity sheet into Power Query as a raw dataset.
3. Promote First Row to Headers	<b>#"Promoted Headers" = Table.PromoteHeaders(#"Line productivity_Sheet", [PromoteAllScalars=true])</b> Converts the first row into the column headers.
4. Apply Initial Data Types	<b>#"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers", {{"Start Time", type time}, {"Date", type date}})</b> Assigns proper data types for Start Time and Date to ensure accurate time/date operations.
5. Remove Unnecessary Columns	<b>#"Removed Columns" = Table.RemoveColumns(#"Changed Type",{"Column7"})</b> Deletes an irrelevant column to clean the dataset.
6. Apply Additional Data Types	<b>#"Changed Type1" = Table.TransformColumnTypes(#"Removed Columns", {{"Product", type text}, {"Batch", type text}, {"Operator", type text}, {"End Time", type time}})</b> Sets product, batch, and operator fields as Text and ensures End Time is correctly formatted.
7. Rename Columns (Standardization)	<b>#"Renamed Columns" = Table.RenameColumns(#"Changed Type1", {{"Product", "ProductID"}, {"Batch", "BatchID"}})</b> Renames Product and Batch to standardized names for data model consistency.
8. Merge With Operators Dimension	<b>#"Merged Queries" = Table.NestedJoin(#"Renamed Columns", {"Operator"}, dOperators, {"Operator"}, "fLineProductivity", JoinKind.LeftOuter)</b> Performs a Left Outer Join to bring related operator information from the dOperators table.
9. Expand Merged Operator Table	<b>#"Expanded fLineProductivity" = Table.ExpandTableColumn(#"Merged Queries", "fLineProductivity", {"OperatorID"}, {"OperatorID"})</b> Extracts the OperatorID field to enrich the fact table.

10. Reorder Columns	<b>#"Reordered Columns" = Table.ReorderColumns(#"Expanded fLineProductivity",{"Date", "ProductID", "BatchID", "Operator", "OperatorID", "Start Time", "End Time"})</b> Organizes columns in a logical order before finalizing the table.
11. Remove Redundant Column	<b>#"Removed Columns1" = Table.RemoveColumns(#"Reordered Columns",{"Operator"})</b> Removes the original Operator column after extracting OperatorID to prevent duplication.

**Final Output:** The final output returns a fully cleaned and structured Line Productivity table. This table serves as the core fact table in the data model and is enriched through a join with the Operator dimension to ensure accurate operator mapping.

# Chapter 2: System Analysis & Design

- ❖ Database Design & Data Modeling
- ❖ UI/UX Design & Prototyping
- ❖ Dashboard Navigation Guide

# System Analysis & Design

## 1. Database Design & Data Modeling

### Data Schema Diagram

The analytical data model in Power BI is designed using a Fact Constellation structure with selective Snowflaking, where multiple fact tables share common dimension tables. The diagram represents the Power BI logical data schema, as opposed to a traditional Entity–Relationship Diagram typically used in database engineering. This design approach supports model purity, consistent granularity, and efficient filtering performance across analytical workloads.

#### A. Model Structure and Fact Separation:

1. **Separation of Fact Tables:** To maintain semantic clarity and ensure accurate analytical calculations, the model separates operational metrics into two distinct fact tables based on their granularity and business meaning:
  - *fLineProductivity*: Contains positive time-related metrics, such as total productivity or operating time.
  - *fLineDowntime*: Contains negative time-related metrics, including total downtime and detailed downtime events.
  - Justification: This separation enforces Measure Purity, prevents metric duplication, and ensures reliable computation of composite KPIs such as Net Operating Time by enabling simple additive/subtractive logic (e.g., *Production Time - Downtime*). Separating the facts also preserves the integrity of granular event-level records.
2. **Shared Dimensions:** Shared dimension tables - including *dProducts*, *dDowntimeFactors*, and *dOperators* - provide a unified filtering context across both fact tables. These dimensions support consistent slicers, model navigation, and cross-table analysis within Power BI, while reducing redundancy through centralized reference data.

#### B. Strategic Dimension Creation: The *dOperators* Table:

- **Normalization:** The *dOperators* dimension was created by extracting operator - related descriptive attributes from the fact table. This follows standard normalization principles, resulting in a cleaner structure, reduced redundancy, and improved model maintainability.
- **Performance:** Storing textual and descriptive attributes in a separate dimension table linked via a numeric key (*OperatorID*) enhances query performance by improving columnar compression, reducing memory usage, and increasing filtering efficiency within Power BI.

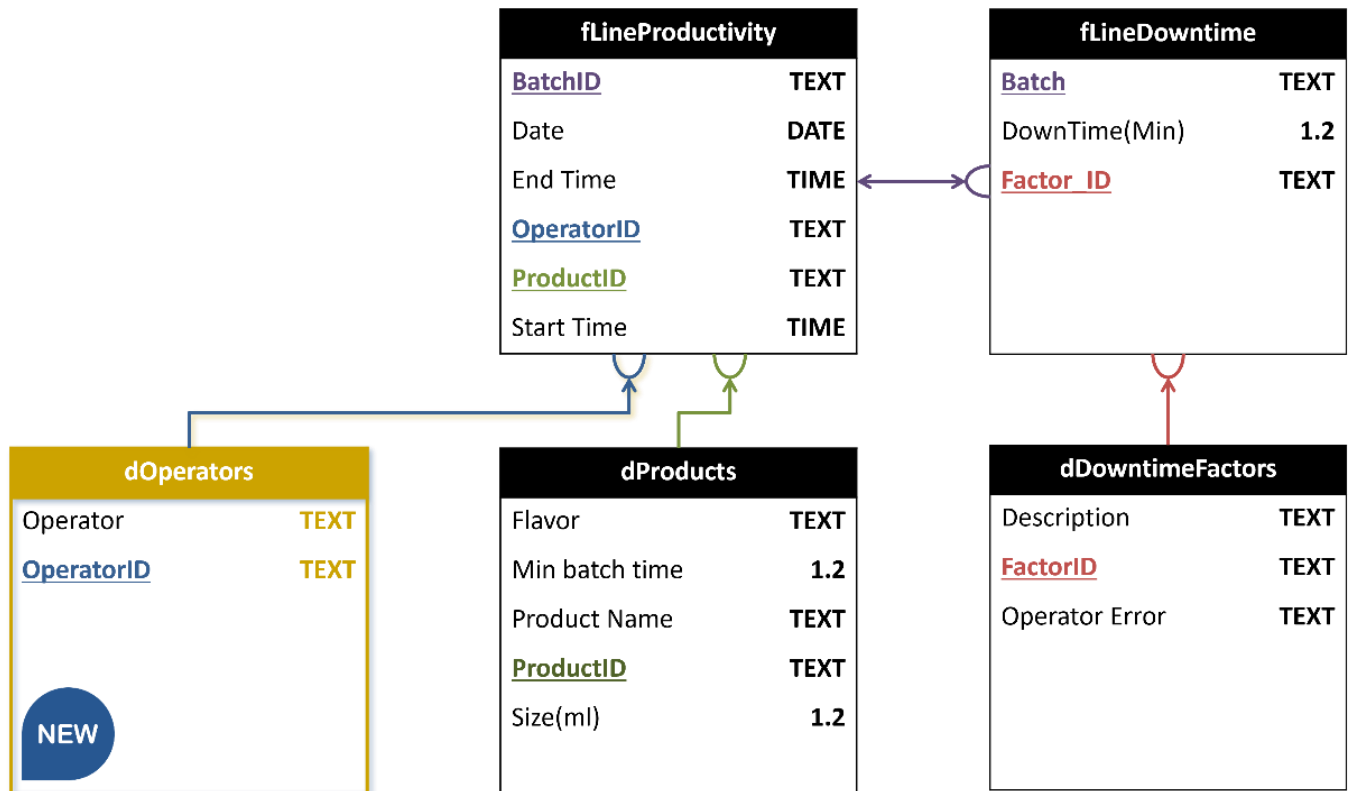


Figure 4-2-1 Snowflake Schema

## 2. UI/UX Design & Prototyping

This section outlines the UI/UX design system, layout strategy, interaction flow, and visual identity used in the Dashboard. The purpose is to ensure a consistent, scalable, and user-centered design across all dashboard pages.

### 2.0 Design Tools & Methodology

A standardized UI/UX workflow was followed throughout the dashboard development process.

Initial wireframes were created to validate layout logic and user flow. These structures were then translated into high-fidelity visual mockups using **Figma**, applying the approved color palette, typography, and interaction guidelines.

The finalized layouts were subsequently implemented in Microsoft Power BI, ensuring full alignment between design intent and functional execution.

#### Tools Used:

- **Figma:** Visual design & component styling
- **Draw.io:** Wireframes & structural mapping
- **Power BI:** Final dashboards & interactive implementation

### 2.1 Wireframes & Mockups

The following wireframe descriptions represent the structure of the dashboard pages. Actual visuals are implemented in Power BI.

1. **Home Page:** Title, introduction, shortcuts, and navigation.
2. **Production Overview:** Batch performance, productivity trends.
3. **Downtime Analysis:** Factors breakdown, top contributors, trend charts.
4. **Operators Performance:** Operator error frequency, benchmark comparison.
5. **Drill-through Pages:** Detailed view for a selected product or operator.

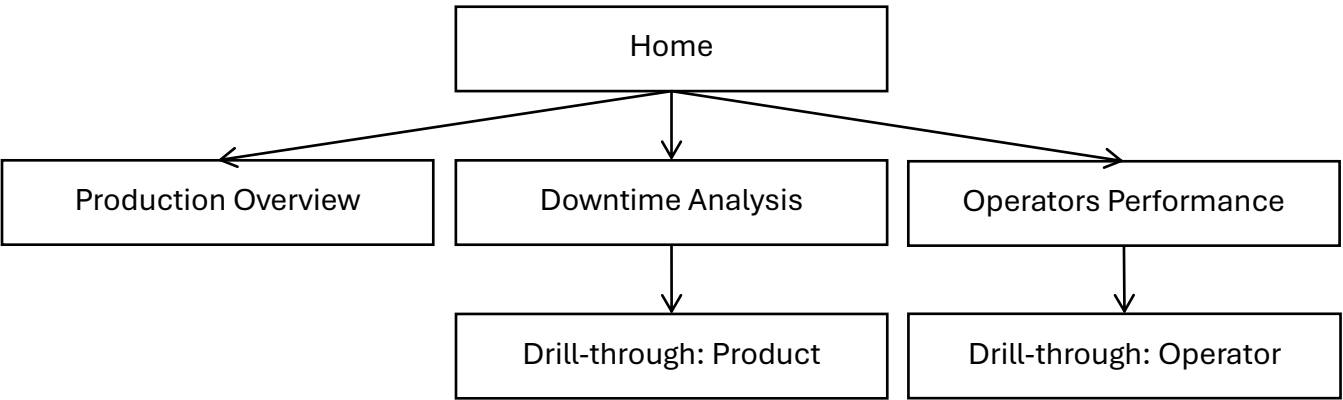
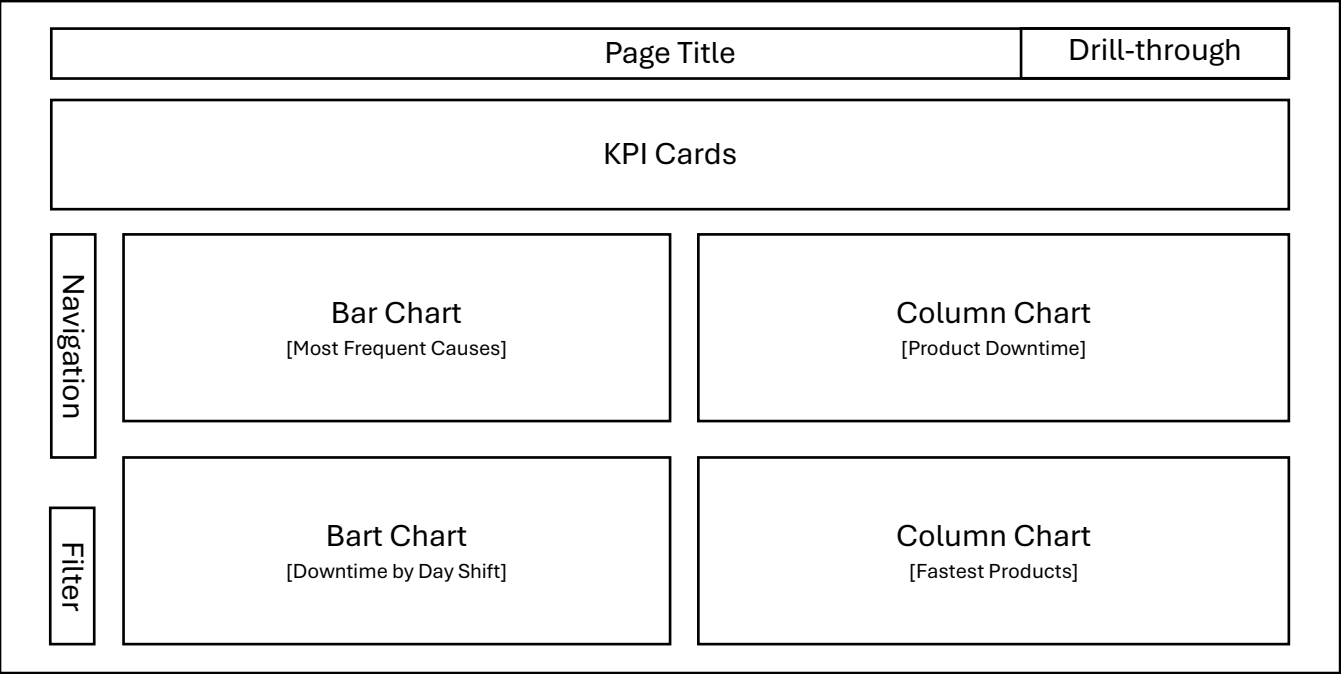
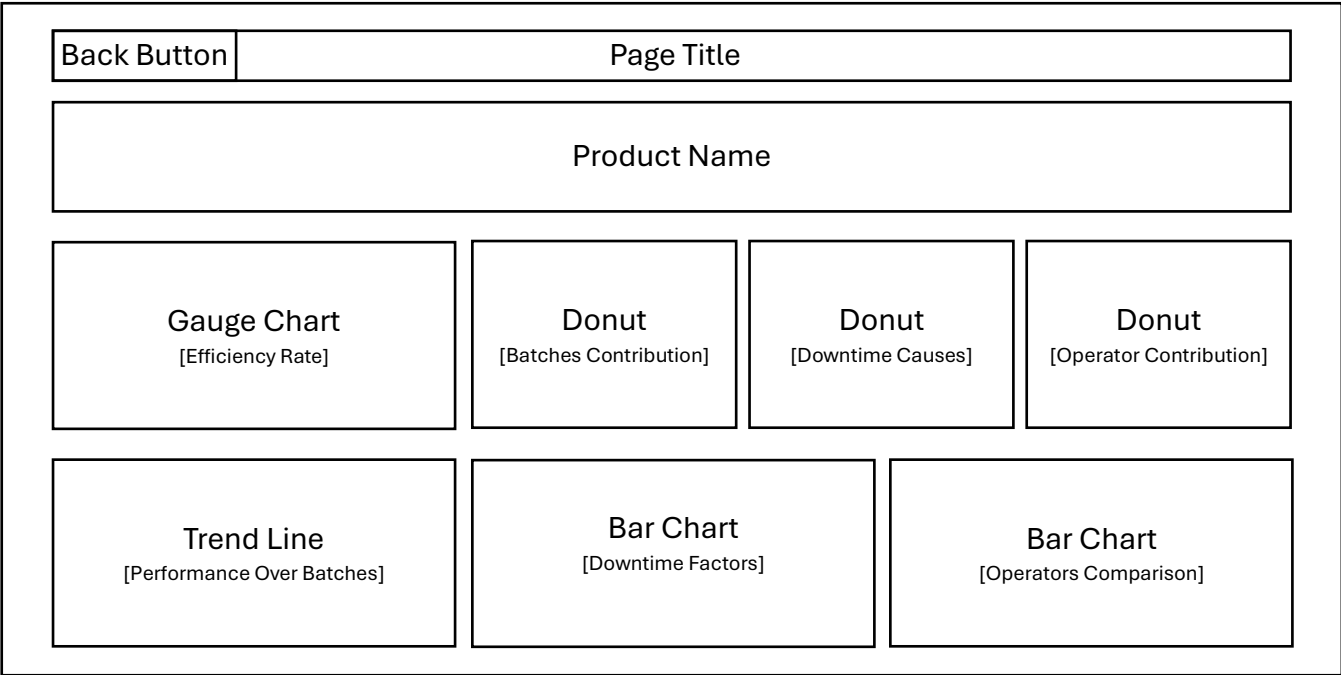


Figure 4-4-1-1 Dashboard Navigation Map



**Figure 4-4-1-2** “Downtime Analysis” Page Wireframe



**Figure 4-4-1-2** “Product Details” Page Wireframe

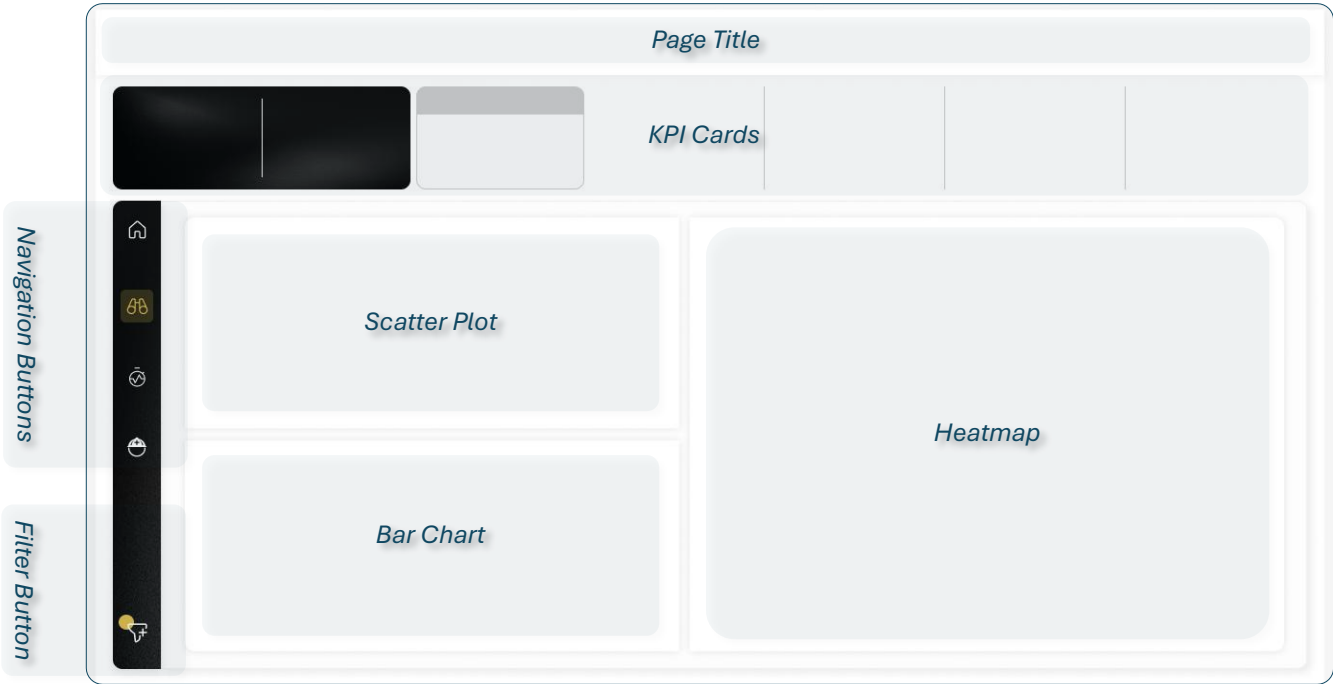


Figure 4-4-1-3 “Production Overview” Page *Figma* Mockup

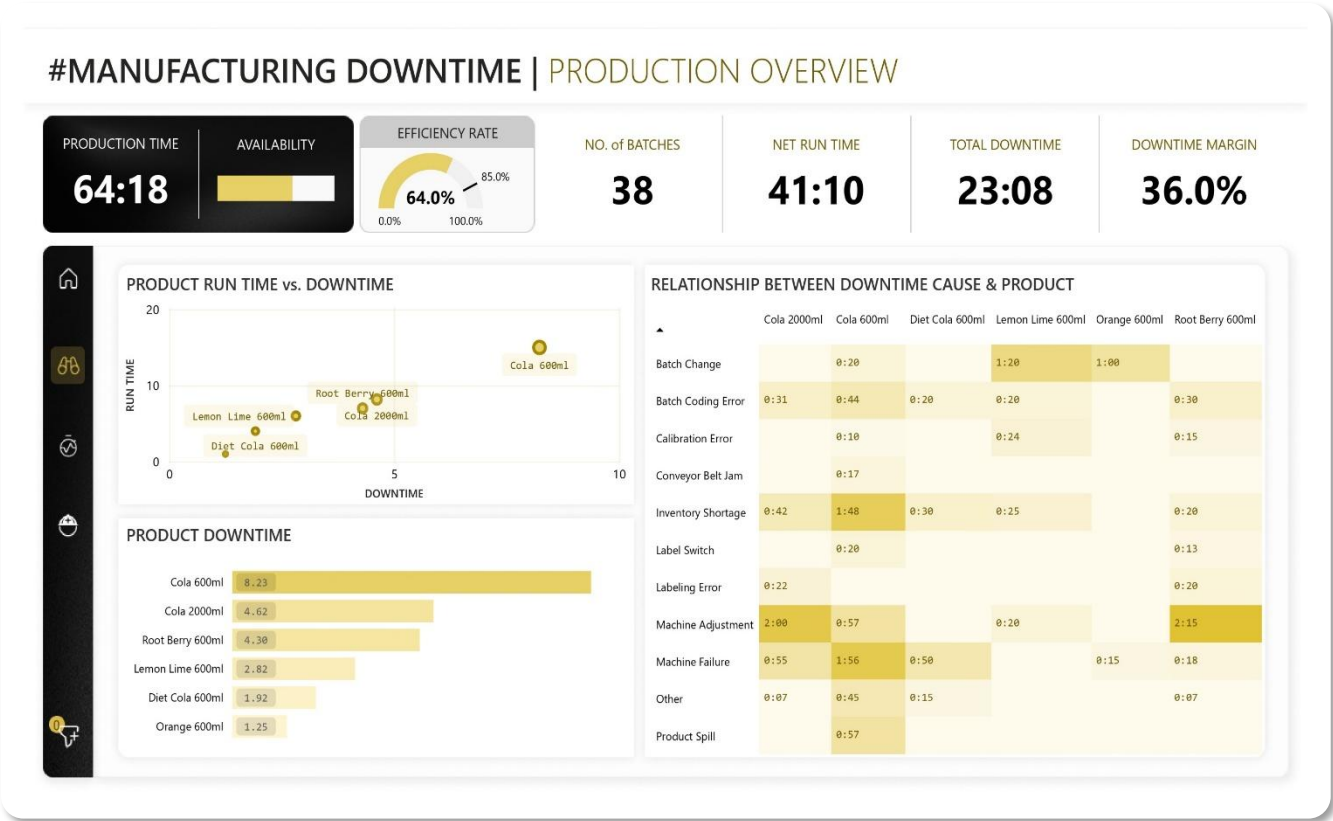


Figure 4-4-1-4 “Production Overview” Final Page

2.3 UI/UX Guidelines

Color Palette

The dashboard uses a dark industrial theme with gold accents to ensure high contrast and visual clarity.

Color	Hex Code	Usage
White	#FFFFFF	Background
Gray	#838383, #CCCCCC	Strokes & Borders
Soft Black	#0D0D0D	Navigation Panel
Gold Accent	#E8D166	KPIs & Highlights
Light Brown	#6D5A00	Text & Numerical Values
Black	#000000	

Typography

The chosen fonts prioritize clarity and hierarchy across KPIs, titles, and visuals.

Element	Font Style
Headers / Page Titles	Segoe UI Bold / Light – 18–24 pt
KPIs & Metrics	Segoe UI Semibold – 24–36 pt
Body Text	Segoe UI Regular – 12–14 pt
Captions / Notes	Segoe UI Light – 10–11 pt

Interaction Flow

1. Hovering highlights key datapoints in gold.
2. Selecting a downtime factor refreshes all visuals contextually.
3. Right-click drill-through navigates to detail pages, or just a left-click and the drill-through button will be automatically activated.
4. Navigation panel remains static for quick switching.
5. Filters update cards, charts, and totals instantly.

Design Decisions & Rationale

The design is built around clarity, contrast, and analytical readability. The dark theme reduces eye fatigue during long operational monitoring sessions, while gold accents ensure that critical KPIs stand out immediately. Horizontal charts enable fast left-to-right comparison of downtime factors.

### 3. Dashboard Navigation Guide

#### 3.1 Page 1: Home Page

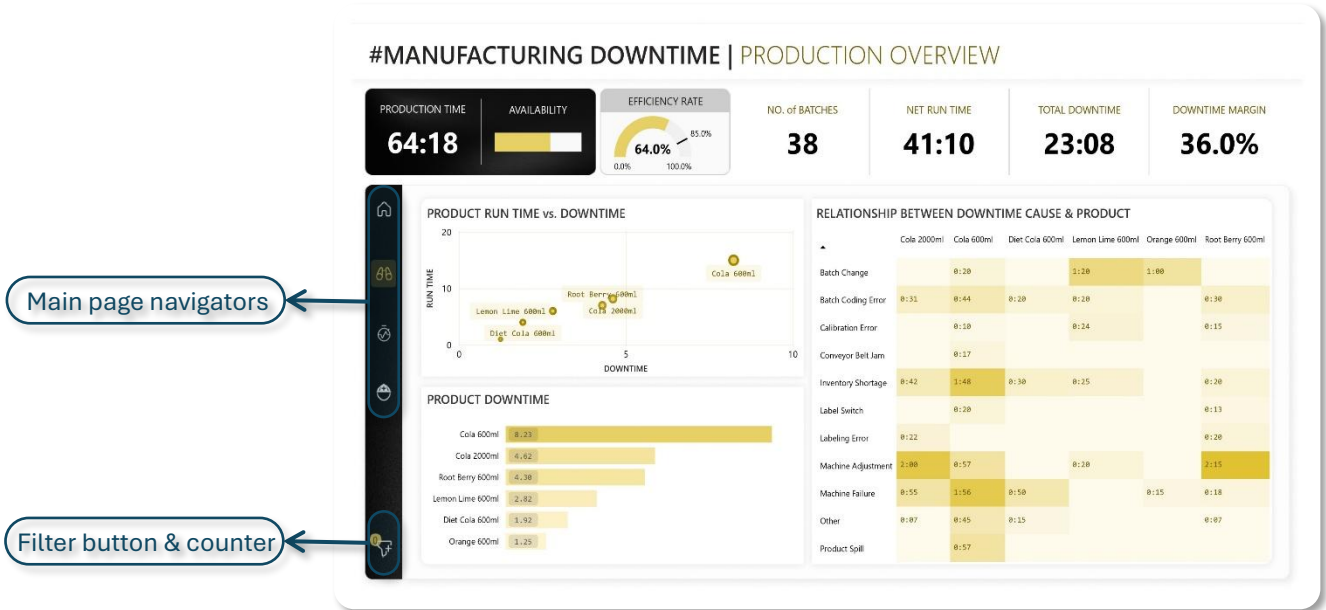
##### A. Cover Page

Shows the main project title with a short introduction. It includes a button to jump directly to the Production Overview page and a link to the GitHub repository for the project.



##### B. Production Overview

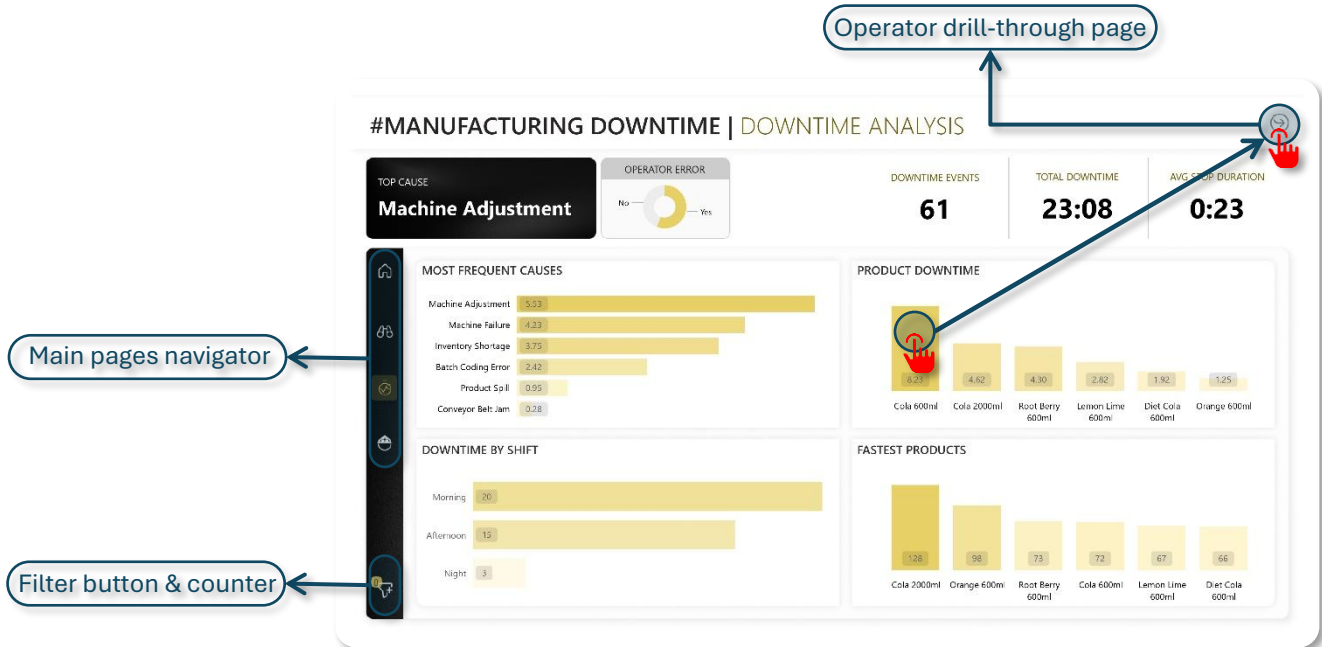
The Production Overview page can be accessed either from the cover page or by clicking the second button in the side menu available on other pages. It presents a general view of production performance along with downtime information, offering a broad comparison between different downtime causes and their relation to products. The page also includes a filter at the bottom to refine the displayed data and a fixed side menu on the left to ensure easy navigation to other sections of the dashboard.



### 3.2 Page 2: Downtime Analysis

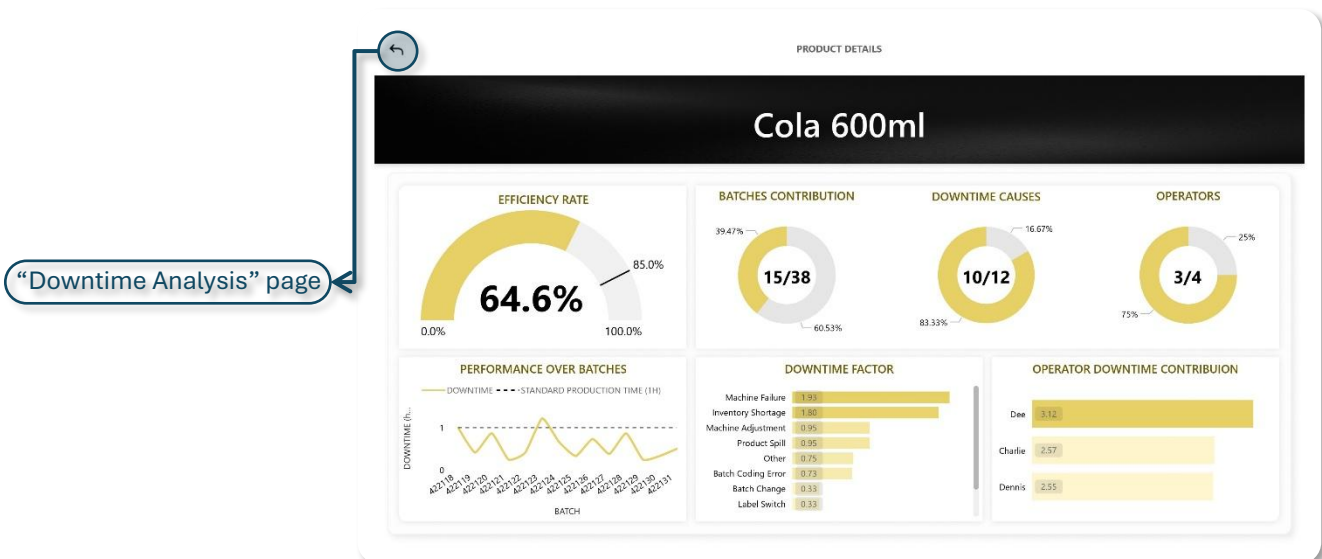
#### A. Overall Downtime Analysis

The Downtime Analysis page is reached through the 3<sup>rd</sup> button at the side menu on the left, available on all main pages except the cover. It shows comparisons between products and downtime causes, dividing them into operator-related and non-operator reasons, and displays the downtime causes timing across daily shifts. The page also has the same filter and side menu as before and includes a drill-through option to explore details for each product.



#### B. Product Details

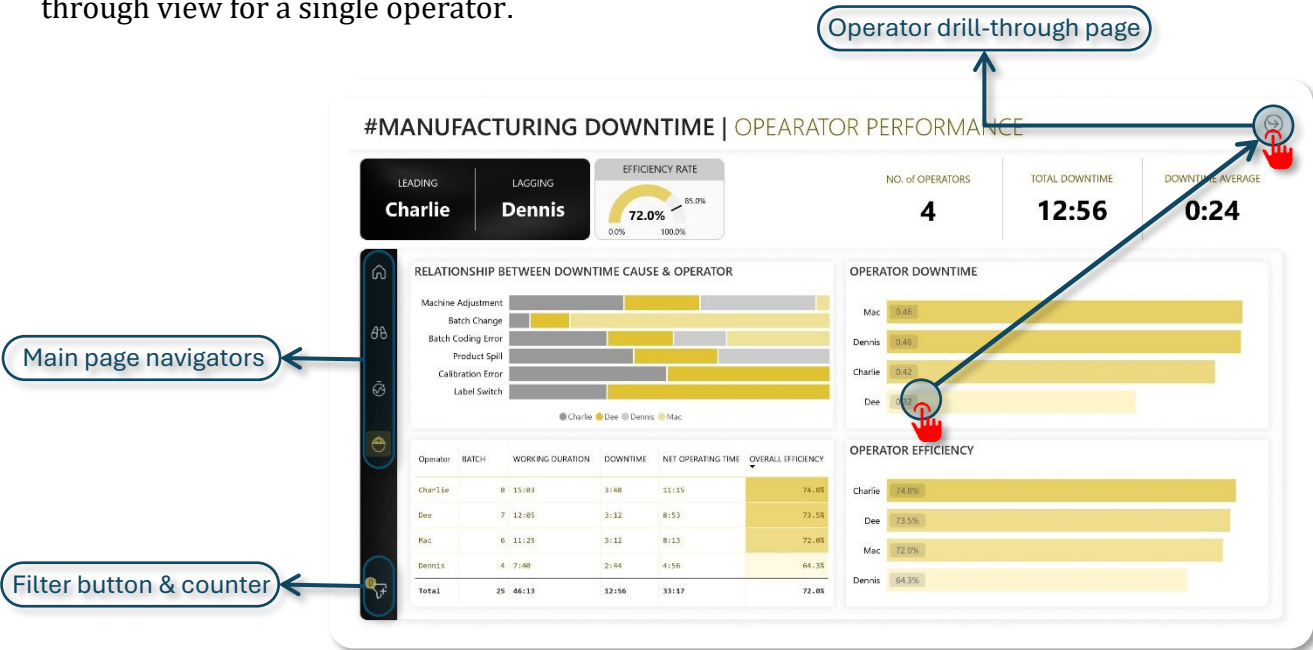
The Drill-through page shows the product name at the top with visuals below linking it to downtime causes, batches, and workers. It also displays the product's status across weekly batches, and a button on the top left lets users return to the Downtime Analysis page.



3.3 Page 3: Operator Performance

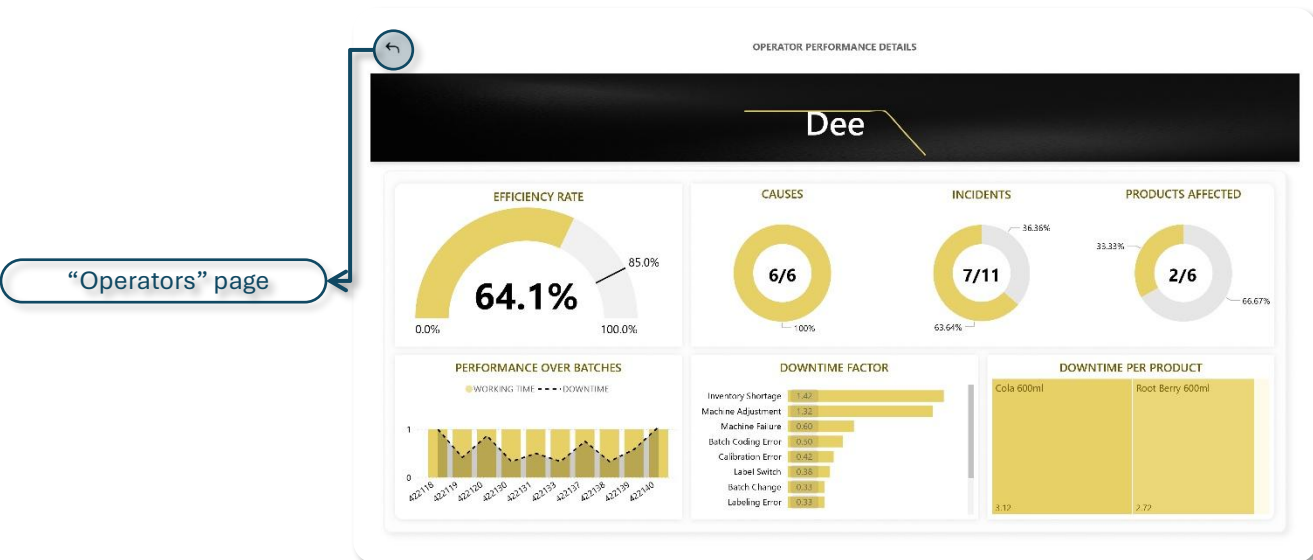
A. Overall Operators

The Operator Performance page follows the same style as the Downtime Analysis page and is accessed through the last button in the side menu. It compares operators and their associated downtime factors to highlight performance differences. The page keeps the same side menu and bottom filter for consistent navigation and includes a button that opens a detailed drill-through view for a single operator.



B. Operator Performance Details

The Operator Drill-through page displays the operator's name at the top, supported by a trend line and visuals showing performance across batches and links to specific downtime factors or products. A button in the top-left allows returning to the main Operator Performance page.

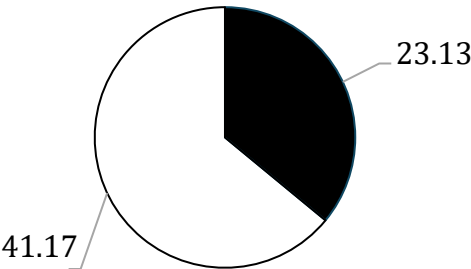


# Chapter 3: Analysis & Insights

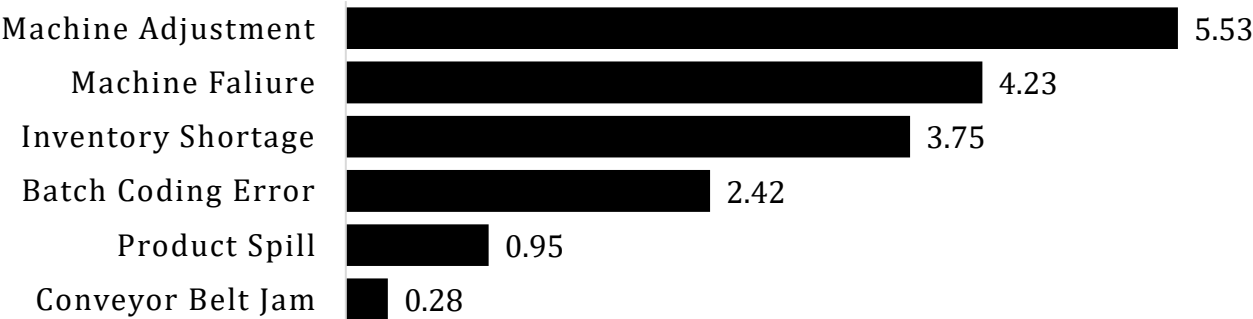
- ❖ Insights
- ❖ Target
- ❖ Recommendations

Insights

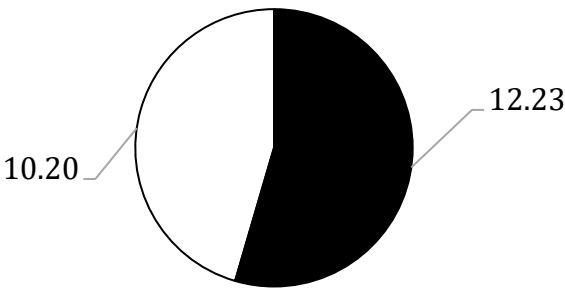
1. Downtime Is Significantly Above Normal Operational Limits



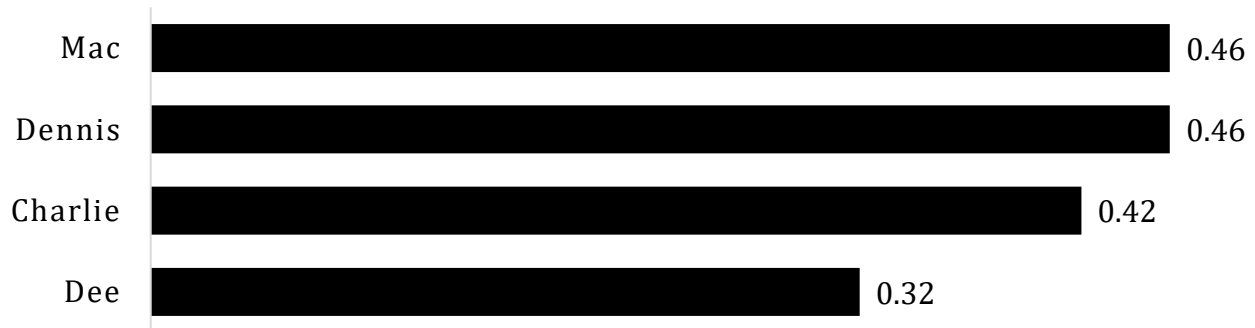
2. Machine-related Issues Account for Most Non-operator Downtime, Led By ‘Machine Adjustment’



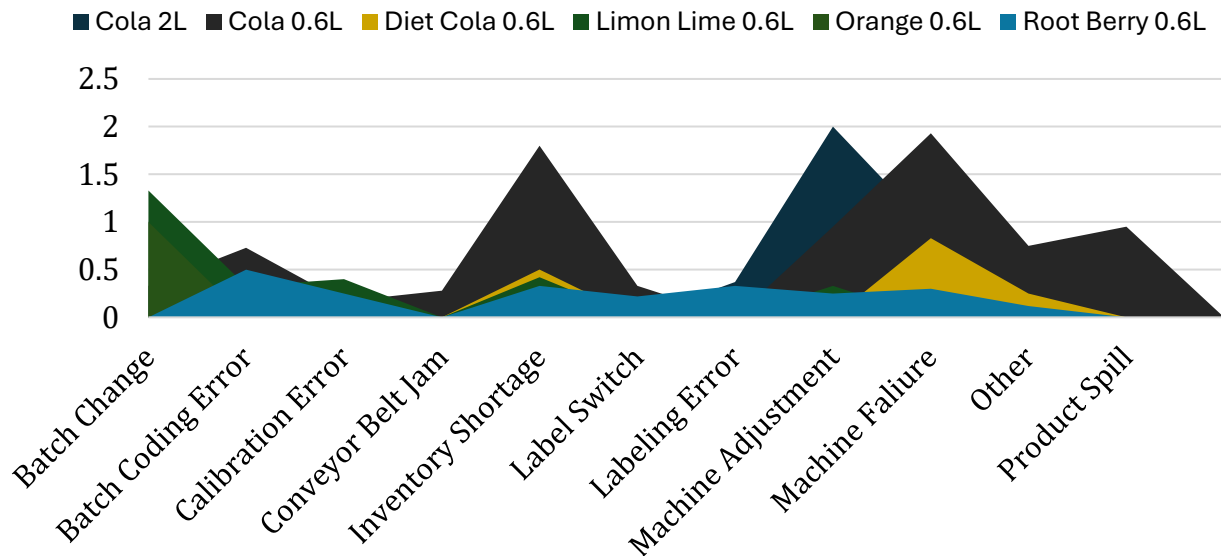
3. Operator Errors Are the Main Driver



#### 4. Some Operators Consistently Contribute More Downtime Than Others



#### 5. Certain Downtime Causes Are Concentrated on Specific Products



#### Executive Summary

- Total downtime reached **36%**, reducing line availability to **64%**.
- **Operator errors (54%)** were the primary driver of downtime.
- **Mac** and **Dennis** recorded the highest downtime contributions.
- **Machine Adjustment** was the main mechanical cause of delays.
- Products **Cola 2L** and **Cola 600ml** showed recurring downtime incidents.
- Downtime is concentrated around operator capability, machine stability, and product sensitivity.

## Key Findings

### 1. Machine Efficiency

- **Mechanical instability** (adjustment + failures) is repetitive and predictable.
- **Preventive maintenance routines** are not consistently executed.

### 2. Operator Performance

- Significant performance variation between operators.
- **Charlie** is the top-performing operator with the highest efficiency (74.8%) and lowest downtime per batch.
- **Dennis** lags with the lowest efficiency (69.4%) and higher downtime contributions.
- **Dee** has moderate efficiency (64.1%) but is involved in a high number of incidents (7 out of 11).

### 3. Product-Specific Downtime

- **Cola 600ml** and **Cola 2L** have the highest cumulative downtime.
- **Lemon Lime 600ml** shows consistent downtime across batches, especially due to Batch Change and Inventory Shortage.

## Target

### To achieve a 20% increase in line availability:

- **Current Availability:** 64%
- **Target:** 84%

### Suggested Strategy:

- Reduce downtime from **23:08** to under **18:00**.
- Focus on top 3 downtime causes (**Machine Adjustment, Operator Error, Inventory Shortage**).
- Improve operator efficiency by **5 – 10%** through training and accountability.
- Streamline batch transitions and reduce incident frequency, by making the switch from one production batch to another faster, smoother, and less error-prone - ultimately cutting downtime and boosting line availability.

## Recommendations

### 1. Operator Performance

- Conduct targeted retraining for **Mac** and **Dennis** focused on setup, calibration, and troubleshooting.
- Introduce **Standard Operating Procedures (SOPs)** for changeovers and adjustments.
- Apply **operator certification** before working independently.
- Track weekly operator performance using **individual scorecards**.

### 2. Equipment & Maintenance

- Perform **full diagnostic inspection** of machines associated with high downtime.
- Implement a structured **Preventive Maintenance (PM) plan** with scheduled calibration.
- Replace repetitive-failure components; evaluate **machine replacement** if instability continues.
- Maintain an internal stock of **critical spare parts** to minimize delays.
- Install **monitoring sensors** to detect misalignment, pressure drops, or temperature changes early.

### 3. Product-Specific

- Create **dedicated setup and calibration checklists** for **Cola 2L**, which is highly sensitive to machine adjustments.
- Conduct a full root-cause review of the **Cola 600ml** process path.
- Perform **test verification runs** after each product changeover to ensure calibration accuracy.

### 7. Inventory & Material Flow

- Activate **minimum stock alerts** to avoid material shortages.
- Pre-stage materials **one shift in advance**.
- Assign a dedicated **material readiness leader** for accountability.
- Increase synchronization between the **warehouse and production**.

### 8. Continuous Improvement Roadmap

- Perform **Root Cause Analysis (RCA)** after major downtime events.
- Hold **monthly cross-functional reviews** between Production, Quality, and Maintenance.
- Track downtime segmented by **operator, machine, product, and shift**.
- Update SOPs regularly based on **lessons learned**.

# Conclusion

