Data Insights Using Data Cubes

Estimated time to read: 7 minutes

This document will provide a quick guide for the usage of the new Data Platform Data Cubes within the MES ecosystem.

Overview

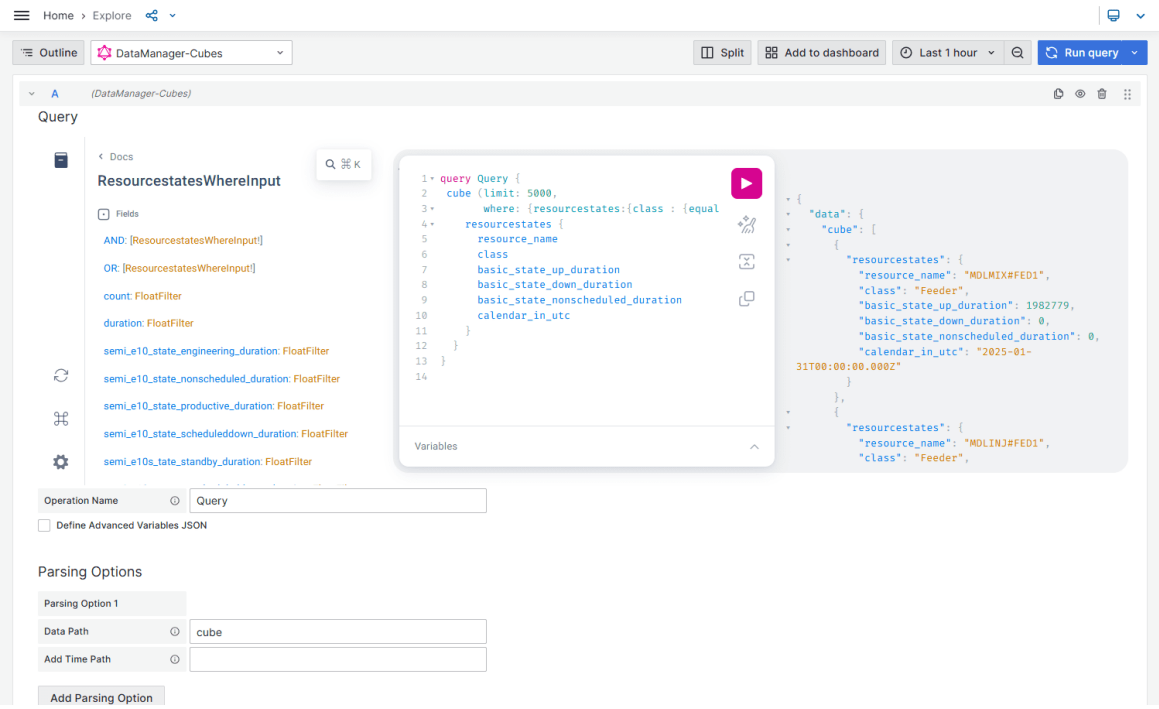
Data cubes allow you to efficiently analyze large datasets by pre-aggregating and organizing data in a structured format. With CubeJS, we can expose these cubes through a **GraphQL API**, making data retrieval seamless and optimized.

In this tutorial, we will explore how to access **Data Cubes** using CubeJS, integrate them into **Grafana** via a custom plugin, and demonstrate a **Work-in-Progress (WIP)** analysis use case.

Data Manager (GraphQL API)

Our **Data Manager** provides a GraphQL API for querying **Data Cubes** efficiently. The API enables:

- **Dynamic querying**: Retrieve specific data without excessive filtering on the frontend.
- **Pre-aggregated insights**: Avoid slow database queries by leveraging CubeJS optimizations.
- **Integration with Grafana**: Our Grafana plugin connects directly to Data Manager via GraphQL API, enabling powerful visualizations.



List of Data Cubes

Below are some commonly available data cubes within our Data Manager:

Cube Name	Description	Use Cases
`cube_material_wip`	Latest snapshot of WIP (over last 5 minutes).	**WIP Monitoring**
`material_movement`	Tracks the movement/processing of materials through various steps within it's manufacturing life-cycle, used for calculating Overall Equipment Effectiveness (OEE) and analyzing material losses.	**Manufacturing performance**
`cube_wip_history`	WIP history records to understand WIP flows and analysis of past constraints.	**WIP tracking**
`wip_state_distribution`	Key performance indicators regarding time WIP spent in each state.	**WIP Constraints / bottlenecks**
`resource_maintenance`	Detailed tracking and analysis of maintenance operations, allowing insights into maintenance stages, orders, and calendar events over time.	**Maintenance performance**
`resourcestates`	This cube provides detailed tracking and analysis of resource states, allowing insights into the operational performance, availability, and reliability of equipment over time.	**Machine performance**
`resourcestates_non_working_times`	Describes the resource states during working and non-working times.	**Machine performance**

Each cube is structured to facilitate efficient querying based on relevant measures and dimensions.

How to use Data Manager Cubes (Grafana plugin)

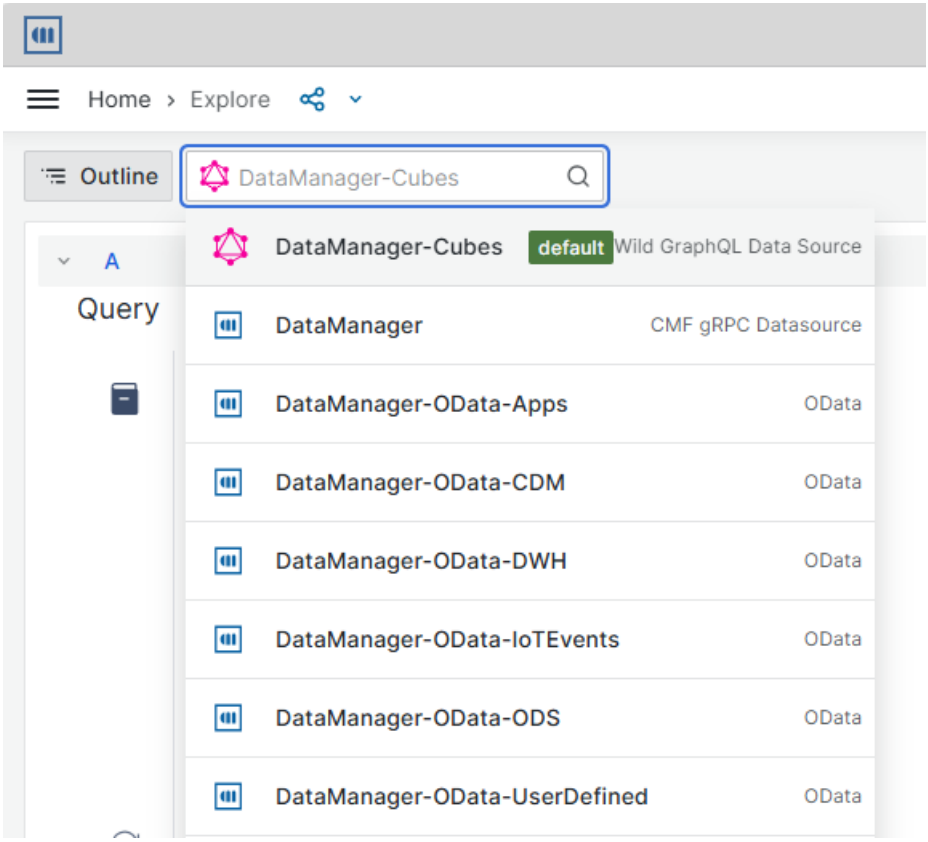
To visualize Data Cubes in Grafana, follow these steps:

Step 1: Open Grafana

- Navigate to Grafana on Explore section. It will open a query editor.
- Critical Manufacturing Grafana instance is accessible at `https://{host url}/Grafana`.

Step 2: Configure the Data Source

- Choose the Data Manager - Cubes plugin from the dropdown.
- Select `Data Manager-Cubes`. The plugin is already pre-configured with a GraphQL API Endpoint (example: `https://cmf-cubejs-instance.com/graphql`).



Step 3: Write a simple Query

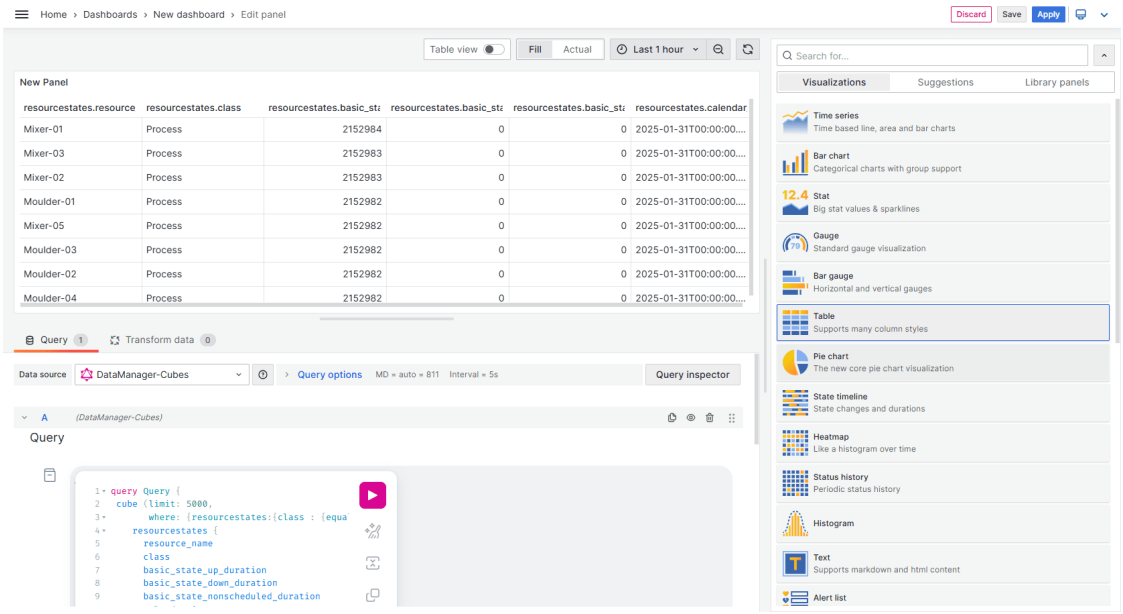
- In the query editor, input a GraphQL query to fetch cube data (example: WIP status, resource performance, availability, etc.).
- A typical GraphQL query to retrieve data from a cube might look like this:

```
query Query {
cube (limit: 5000,
      where: {resourcestates:{class : {equals: "Process"}}}){
    resourcestates {
      resource_name
      class
      basic_state_up_duration
      basic_state_down_duration
      basic_state_nonscheduled_duration
      calendar_in_utc
    }
  }
}
```

- Select Run Query to preview the results.

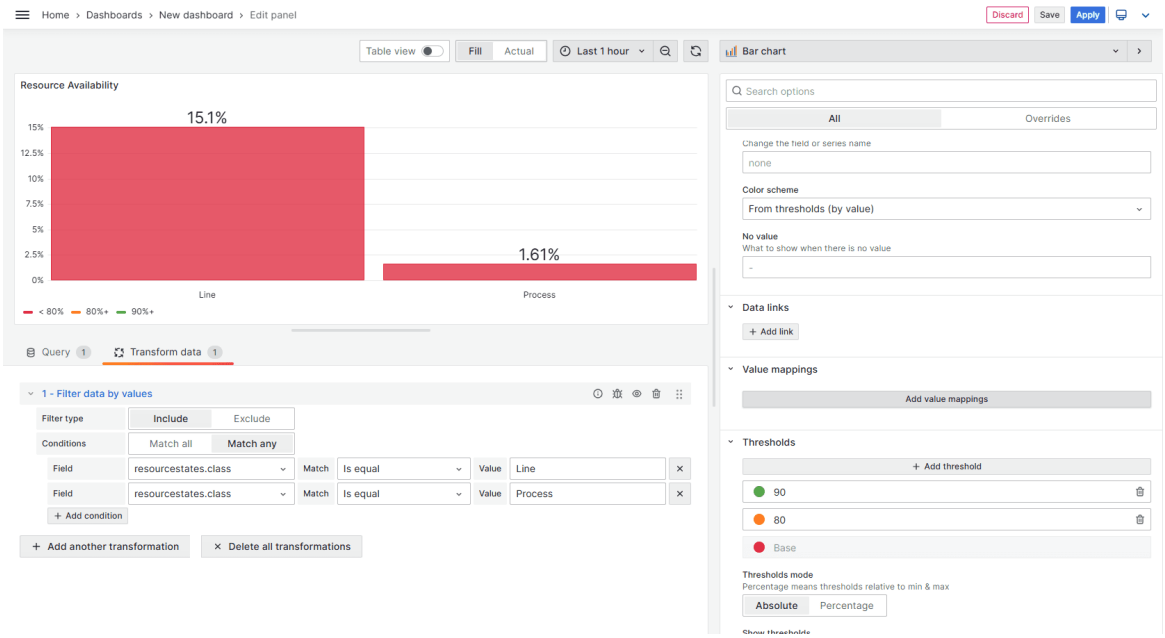
Step 4: Visualize the Data

- Select **Add to dashboard** to open a new dashboard with visualization panel.
- Select edit on the right side of the panel:



- On top right you may choose the visualization type (Table, Graph, or Bar Chart).
- Select apply.

Congrats! You've successfully visualized data from the Data Manager Cubes plugin in Grafana.



Use Case | WIP Analysis

A manufacturing plant needs to track the Work-in-Progress (WIP) status across different production areas. They want to:

Objective

- Monitor how many units are currently in progress.
- Identify bottlenecks by analyzing cycle times or WIP build ups.

Motivation

- Ensure smooth production flow with minimal delays.

To achieve this, we'll leverage the `cube_material_wip` data cube, which provides insights into the current WIP status. Moreover, we'll use the `cube_wip_history` cube to track the history of WIP status changes. If aiming to analyze cycle times, we may use the wip state distribution cube for instance.

To explore the API, you may also use tools like Postman, or ApiDog (see this link), sending queries to `https://{host url}/graphql` endpoint.

Here are the steps to achieve this:

Define the Dashboard Layout

- Having the objectives in mind, it seems we need to create 4 visualization panels:
  - **Panel 1**: Current WIP Status on selected area
  - **Panel 2**: Current WIP Status on selected area per step
  - **Panel 3**: WIP History to understand the flow of WIP
  - **Panel 4**: Bottleneck Analysis

Panel 1: Query WIP Status for a Specific Area

Retrieve real-time data from the data cubes. We then extract the Queued, InProcess and Processed WIP using the following GraphQL query:

```
query query($area:String){
  cube(limit:5000, where:{cube_material_wip: {area_name: {equals: $area}}}) {
    cube_material_wip {
      area_name
      queued_free_primaryqty
      inprocess_free_primaryqty
      processed_free_primaryqty
      queued_hold_primaryqty
      inprocess_hold_primaryqty
      processed_hold_primaryqty
      queued_rework_primaryqty
      inprocess_rework_primaryqty
      processed_rework_primaryqty
    }
  }
}
```

To achieve the visualization defined for Panel 1, we need to apply some transformations in Grafana to the data. We want to sum all the components of WIP that are in Rework and on Hold. For that we apply the `Add field from calculation` option. What this does is to calculate the sum of the following fields and add the respective columns:

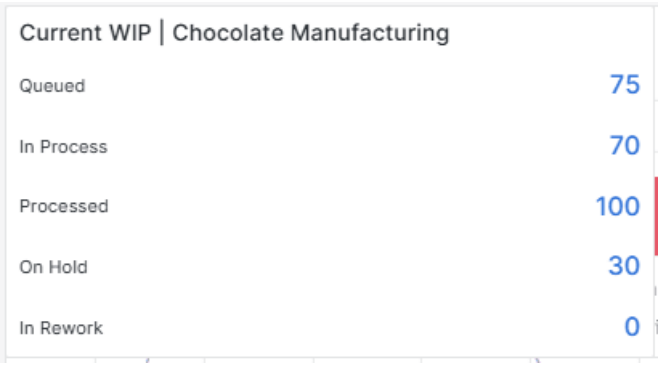
- `queued_rework_primaryqty`
- `inprocess_rework_primaryqty`
- `processed_rework_primaryqty`
- `queued_hold_primaryqty`
- `inprocess_hold_primaryqty`
- `processed_hold_primaryqty`

> **Info**
>
> For more information about the transformations, see the Grafana documentation here.

Additionally, we will include two more transformations:

- `Filter fields by name` (to filter the columns of interest)
- `Organize fields by name` (to rename the columns of interest)

Since we want to monitor WIP status, the end result is the following:



> **Info**
>
> For the plugin to work with Grafana variables you need to map the variables to the query parameters. For more information, see the GraphQL Plugin documentation here.

Panel 2: Decompose WIP status by step

Next to WIP status of the selected area, we want to monitor the WIP status per step. We will use the same query as in Panel 1, but we will add the step name to the query as an output parameter.

> **Info**
>
> You may define thresholds in the thresholds section, if needed.

Panel 3: WIP evolution over time

To monitor the evolution of WIP over time, we'll use the `cube_wip_history` cube. This cube provides a history of WIP status changes, allowing us to track the flow of WIP.

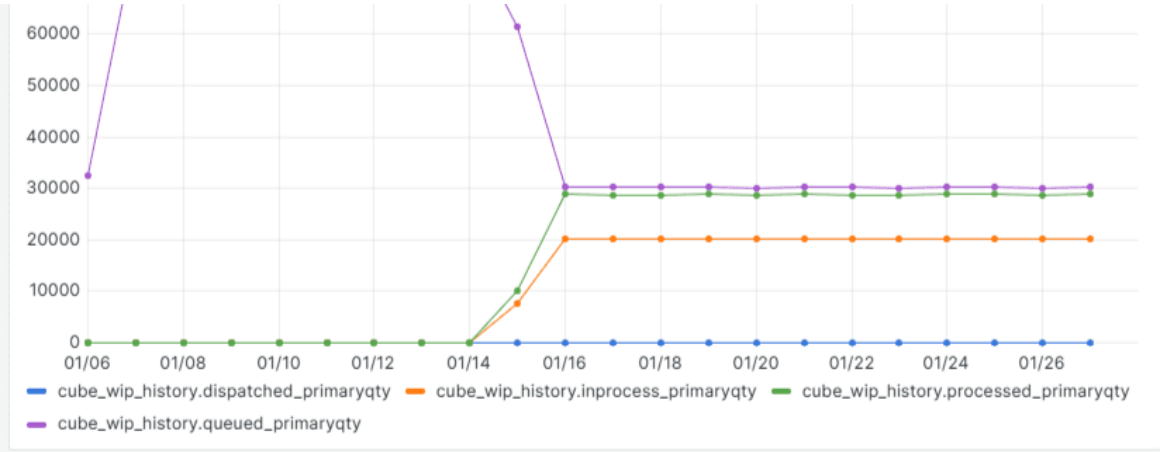
This was the applied query:

```
query ($area: String, $from: String!, $to: String!) {
  cube(limit: 5000, where: { cube_wip_history: {area_name: {equals: $area}, calendarday:
{inDateRange: [$from, $to]}}}) {
    cube_wip_history{
      inprocess_primaryqty
      processed_primaryqty
      queued_primaryqty
      calendarday {
        value
      }
      area_name
    }
```

```
    }
}
```

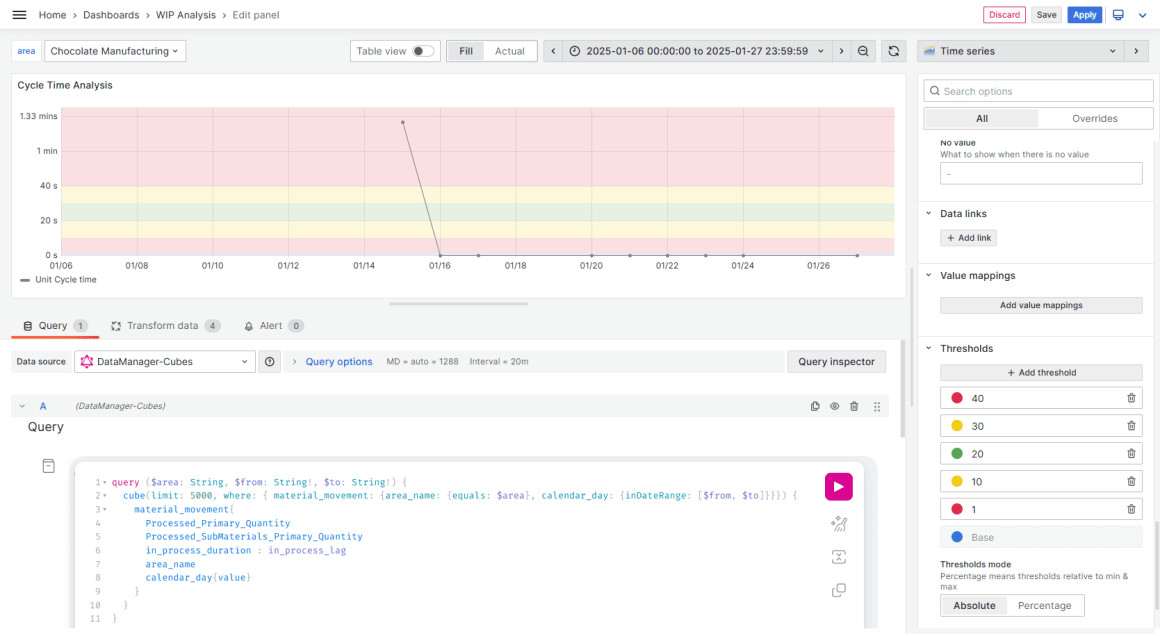
As we can see, we are using the `cube_wip_history` cube to retrieve the WIP history data for a selected area. We are also using the `calendarday` field to filter the data by date range. It is a date field that is used to filter the data by a date range. The output is a list of WIP status changes over time.

Here is the visualization end result:



Panel 4: Bottleneck Analysis

For this panel, and to compare cycle times with WIP history chart, we have decided to add a visualization that shows the unit cycle time within the selected area. For this, we will use the `material_movement` cube to calculate the unit cycle time for the selected `area_name` (see image below). Grafana will allow us to define visual thresholds for the unit cycle time.



Expected Outcomes

We can now monitor the WIP status of the selected area in real-time, and track the flow of WIP over time. Additionally, we can identify area constraints and take data-driven actions to optimize the production process.

In conclusion, here are the expected outcomes of this use case:

- Real-time visibility of WIP across areas and steps.
- Identification of potential bottlenecks and constraints.
- Improved decision-making for production planning.

This was our final dashboard, which is just an example. You can create your own dashboard with the data cubes that you need.