phuse·share
connect·share·advance

Single Day Event

LLM-Powered Data Governance and Analytics Solution for Pharma Commercial

Ding Cheng,
Abbvie - AA

Disclaimer

The information contained in this presentation is for **informational purposes only**. While every effort has been made to ensure the accuracy of the data and insights presented, **no warranties or guarantees** are made regarding the completeness or correctness of the content. The views and opinions expressed herein are **those of the presenter** and do not necessarily reflect the official policies or positions of any organization mentioned.

This presentation should **not** be considered as professional advice on business, financial, legal, technical, or any other matters. Any **reliance** on the information provided in this presentation is at the **sole discretion and risk** of the recipient. For specific advice, please consult with relevant professionals.

The presenter reserves the right to **update or modify** the content at any time without prior notice.
The views and opinions expressed in this presentation are **solely my own** and do not reflect the official policy or position of any company or organization I am affiliated with. This presentation is intended for informational purposes only and should not be considered as professional advice.

Project Background

Users without technical knowledge can interact with data instead of dashboards and reports

Example interaction questions:

What is the account performance for given product(s) for a given market?

What is the performance for given product indication(s) for a given market?

Who are the top hospitals for a given geography for given product(s) / market?

What are sales time trends for a product(s) under different groups?

Dashboard

Chatbot-Agent

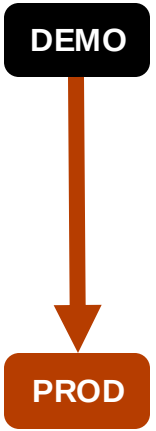
Performance and Quality

Assume the user uses the product only once per workday :

Accuracy Rate	Error Rate	Frequency of Issue
< 80%	>20%	Within a week
90%	10%	Every two weeks
95%	5%	Monthly
> 99%	< 1%	Quarterly

2x

> 5x

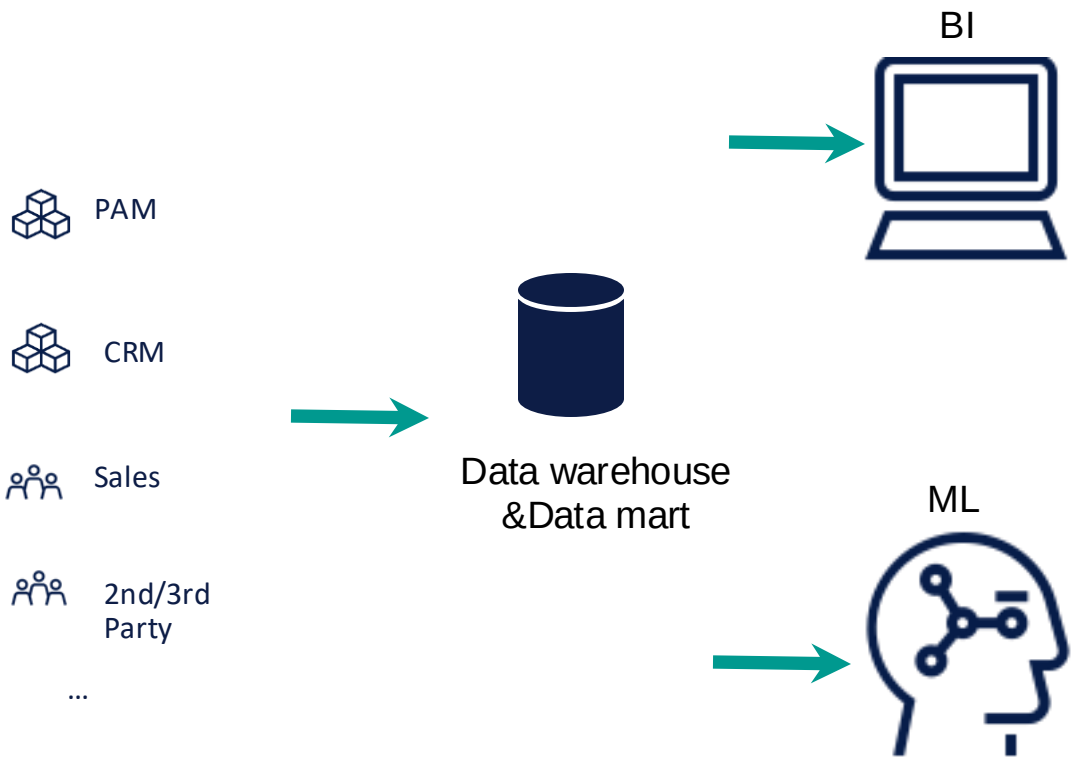


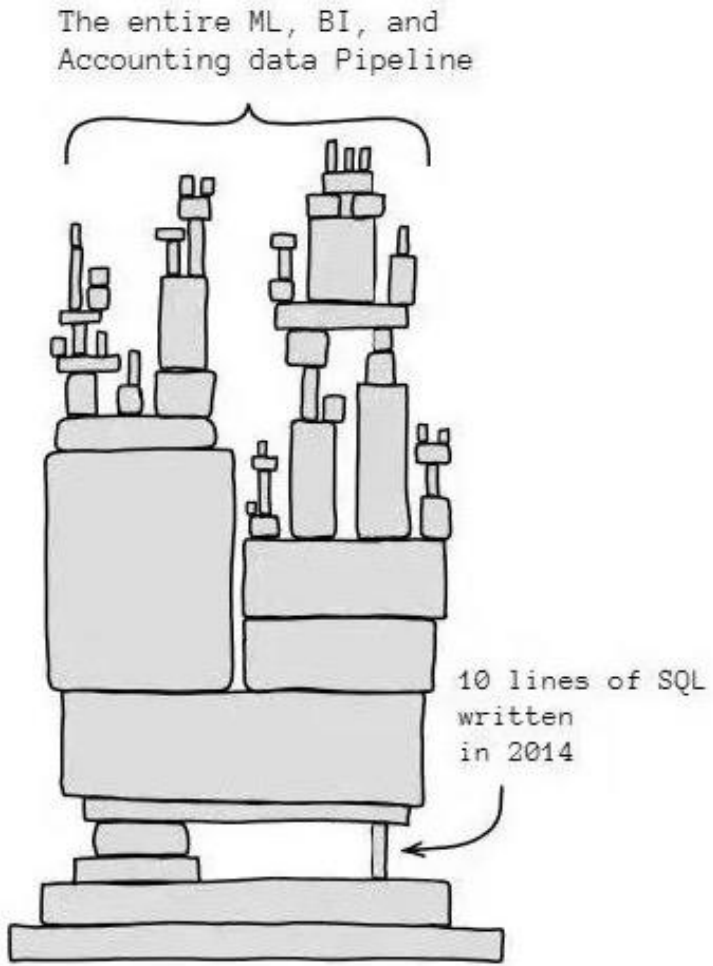
“Metadata”

1. Ugly and outdated Excel files

2. Nested codes with almost no comments

Spaghetti Data Pipeline





Data Governance - Data Lineage

Troubleshooting

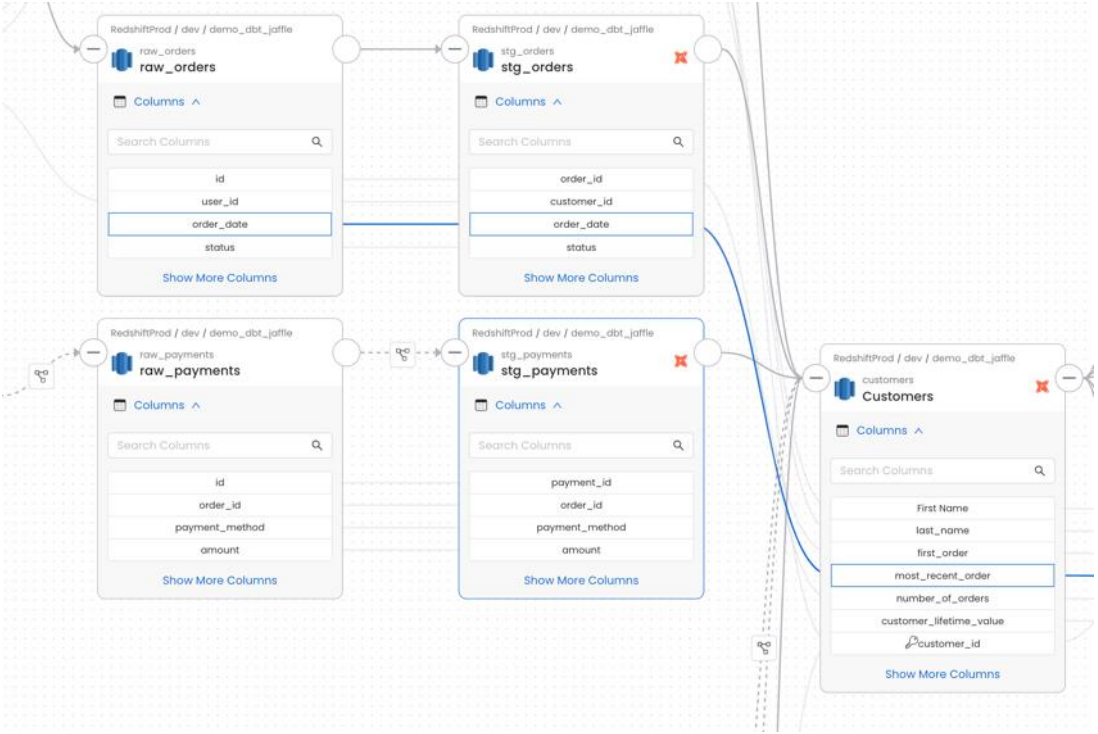
Identify the source of issues quickly when data problems arise, speeding up resolution

Impact Tracking

How upstream data affect downstream results; data asset priority

Observability

Data source transparency; compliance and risk management efforts



Data Lineage Breakdown

Table-level Lineage

Tracks the relationships and flow between entire tables.

- **Joins**
- **Unions**

Column-level Lineage

Focuses on how specific columns are derived or transformed.

- **Column derivations**
- **Column renaming**
- **Pivot(Melt)**
- **Column splitting/combining**

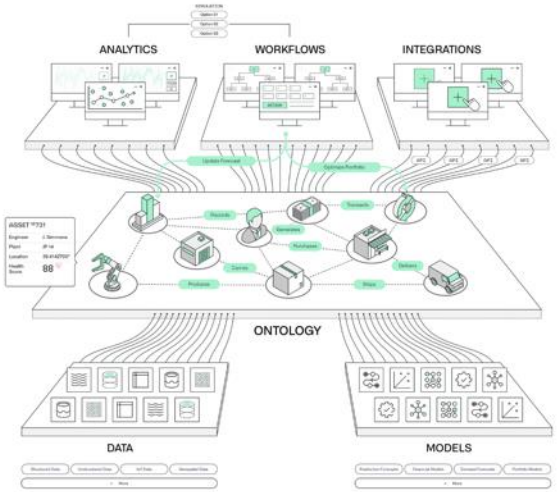
Row-level Lineage

Tracks how individual rows evolve through data transformations.

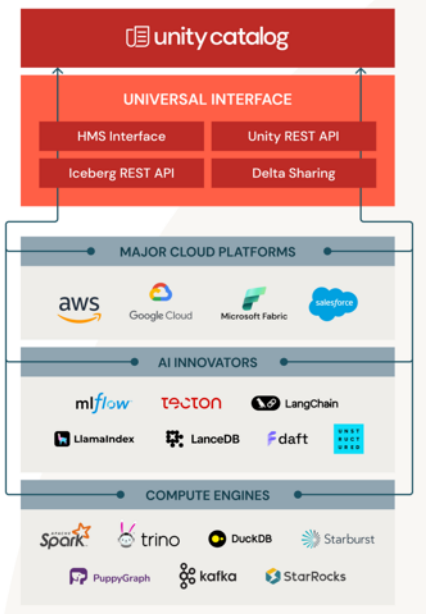
- **Aggregations** (e.g., sum, average, count)
- **Filters** (e.g., exclude nulls, apply conditions)
- **group-based calculations**

Solution on market

Palantir



databricks



Collibra

snowflake

Alibaba Cloud

LLM Capabilities & Limitations

LLM

Microsoft Azure OpenAI

Qwen

NLP Problems:

2-step reasoning

Translation
Translate from SAS to Python

Paraphrasing
Draft code from examples

Extraction

Logical or math problems:

Multi-step reasoning
Cleaning and transforming data through multiple steps

Planning
Complex Problem Decomposition

Critical thinking

Computation

LLM: Translation

```
Filter patients older than 40 with blood pressure above 130
Filter for diagnoses within the past year
Group by gender, disease type, and treatment method
Calculate the average blood pressure and patient count for each group
Output the results
```

```sql
WITH filtered_data AS (
    SELECT * FROM patients
    WHERE age > 40 AND blood_pressure > 130 AND diagnosis_date > DATEADD(year, -1, GETDATE())
)
SELECT
    gender,
    disease,
    treatment,
    AVG(blood_pressure) AS mean_blood_pressure,
    COUNT(*) AS patient_count
FROM
    filtered_data
GROUP BY
    gender,
    disease,
    treatment;
```


```python
filtered_data = data[(data['age'] > 40) & (data['blood_pressure'] > 130)]

one_year_ago = datetime.now() - timedelta(days=365)
filtered_data['diagnosis_date'] = pd.to_datetime(filtered_data['diagnosis_date'])
filtered_data = filtered_data[filtered_data['diagnosis_date'] > one_year_ago]

grouped_data = filtered_data.groupby(['gender', 'disease', 'treatment']).agg(
    mean_blood_pressure=('blood_pressure', 'mean'),
    patient_count=('blood_pressure', 'size')
).reset_index()
```

```r
result <- data %>%
    filter(age > 40, blood_pressure > 130) %>%
    filter(diagnosis_date > (Sys.Date() - years(1))) %>%
    group_by(gender, disease, treatment) %>%
    summarize(
        mean_blood_pressure = mean(blood_pressure, na.rm = TRUE),
        patient_count = n()
    )
```

LLM: Extraction

Data Lineage Standard：source, destination, spec

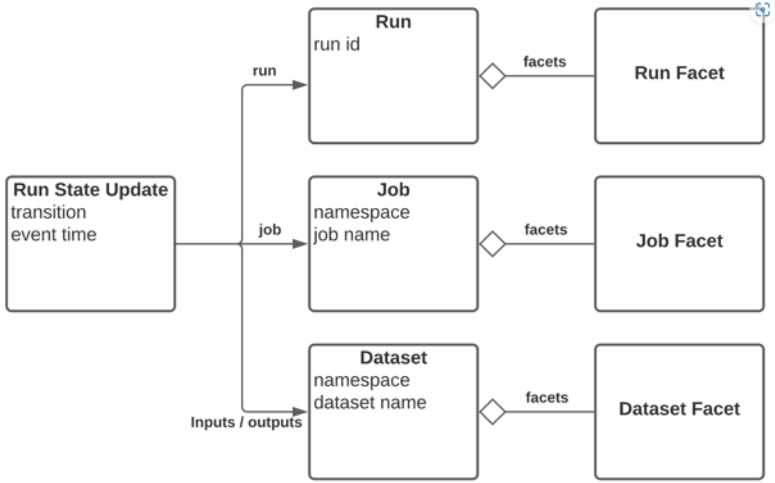
SQL

```sql
WITH filtered_data AS (
    SELECT * FROM patients
    WHERE age > 40 AND blood_pressure > 130 AND diagnosis_date > DATEADD(year, -1, GETDATE())
)
SELECT
    gender,
    disease,
    treatment,
    AVG(blood_pressure) AS mean_blood_pressure,
    COUNT(*) AS patient_count
FROM
    filtered_data
GROUP BY
    gender,
    disease,
    treatment;
```

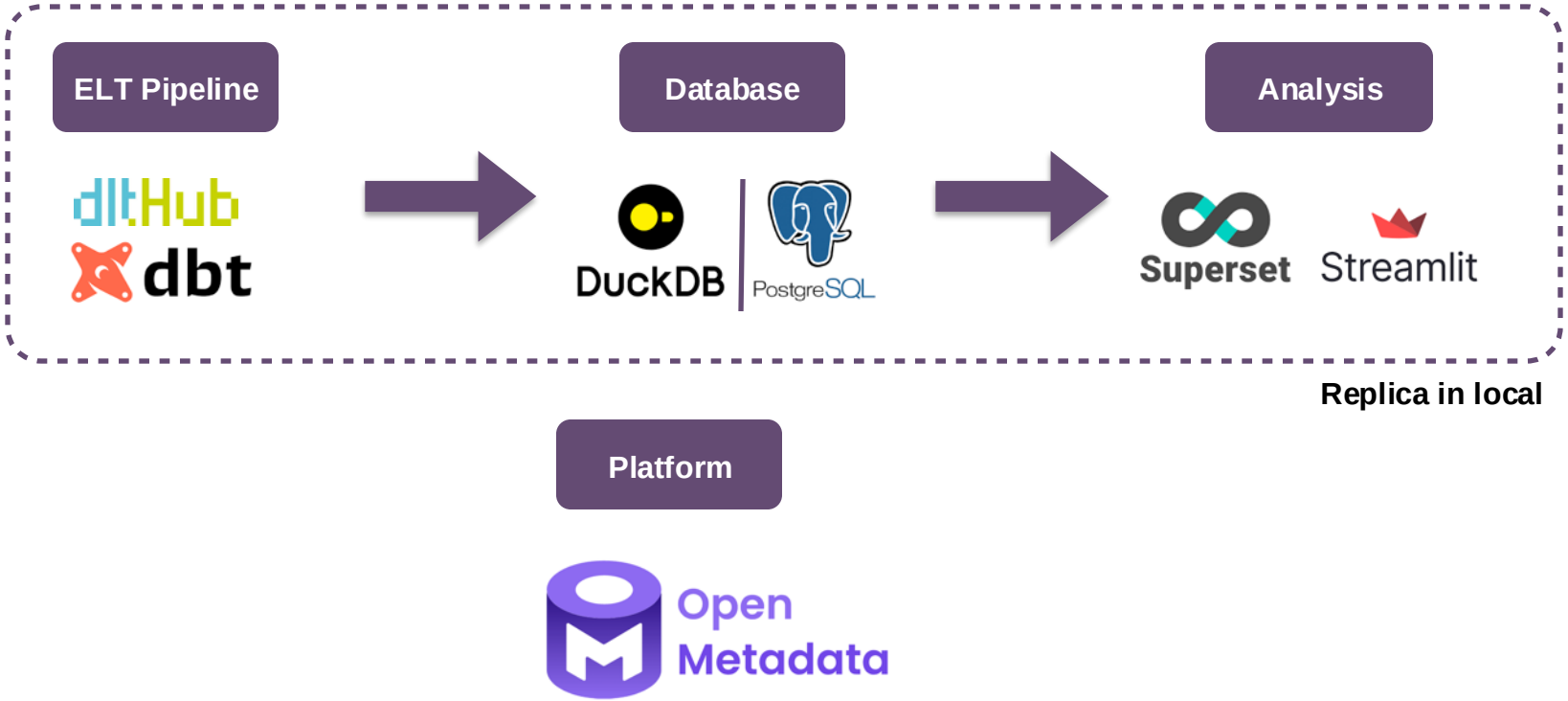
{.json}

```json
{
  "eventType": "COMPLETE",
  "eventTime": "2024-10-31T12:00:00Z",
  "job": {
    "namespace": "exampleNamespace",
    "name": "join_sales_customer_job"
  },
  "run": {
    "runId": "1234-5678-91011",
    "facets": {
      "runFacet": {
        "nominalStartTime": "2024-10-31T11:00:00Z",
        "nominalEndTime": "2024-10-31T12:00:00Z"
      },
      "transformationFacet": {
        "type": "sql",
        "description": "Join sales and customer tables on customer_id and filter sales over $100",
        "steps": [
          {
            "step": 1,
            "name": "Filter Sales",
            "description": "Filter out sales with amount less than $100",
            "transformationLogic": "SELECT * FROM Sales WHERE amount >= 100",
            "inputFields": ["amount"],
            "outputFields": ["amount"]
          },
          {
            "step": 2,
            "name": "Join Tables",
            "description": "Join Sales and Customer tables on customer_id",
            "transformationLogic": "SELECT s.sale_id, c.customer_name, s.amount, s.date, c.region FROM Sales s JOIN Cu
            "inputFields": ["Sales.customer_id", "Customer.customer_id"],
            "outputFields": ["sale_id", "customer_name", "amount", "date", "region"]
          }
        ]
      }
```

OpenLineage



Tech Framework Overview

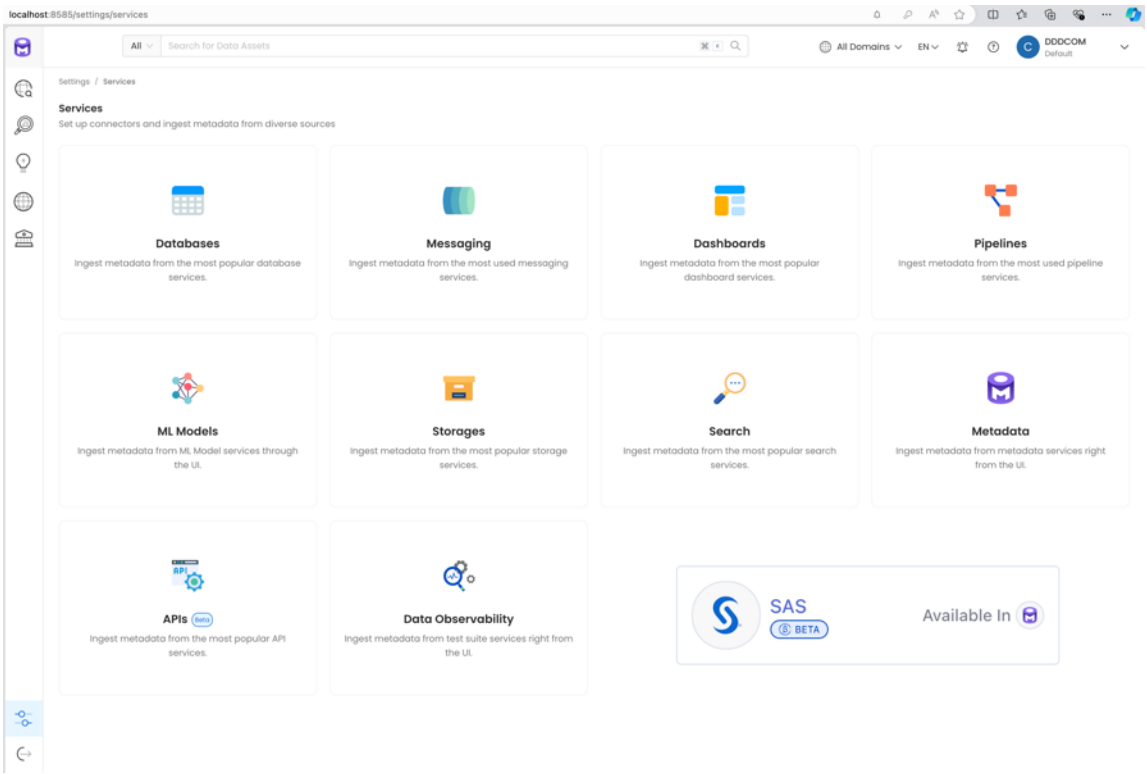


Human-in-the-loop

Open Metadata

Python SDK

Presentation of a high-level Python API as a type-safe and gentle wrapper for the OpenMetadata backend.



1. Manual and automated management
2. Standalone Lineage output

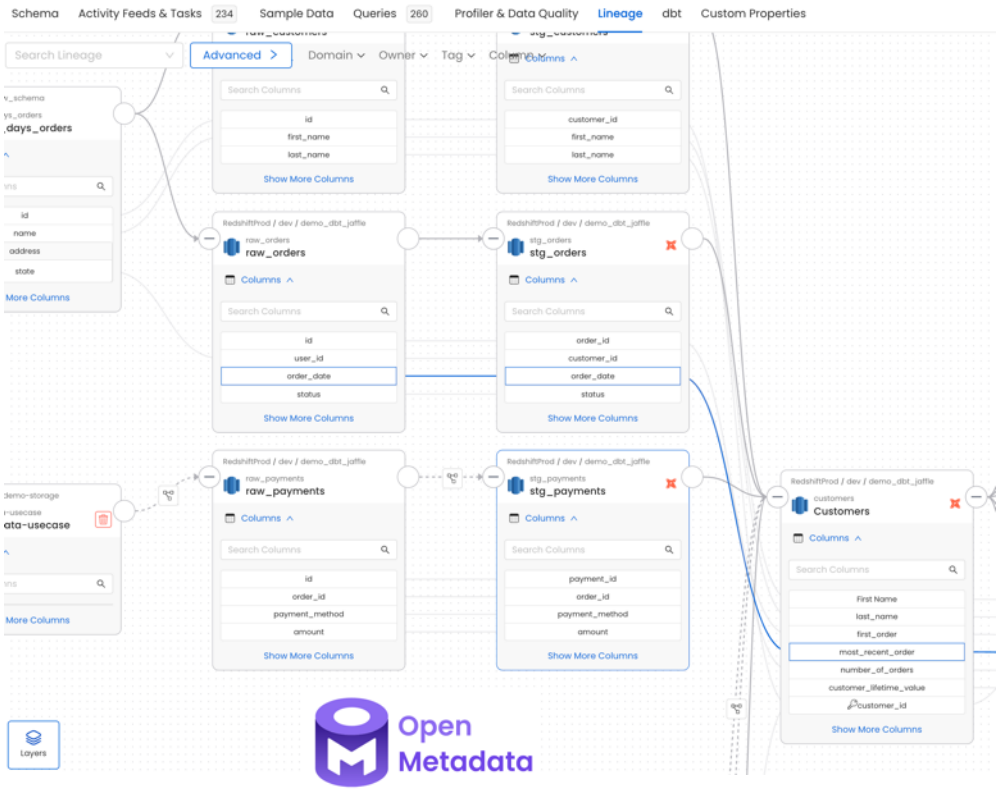
https://github.com/open-metadata/OpenMetadata?tab=readme-ov-file#install-and-run-openmetadata

Think Data Lineage

Troubleshooting

Impact Tracking

Observability



Thank You

The Global Healthcare Data Science Community

Contact Channels
UK +44 1843 609600
office@phuse.global
phuse.global

Social Media
@phusetwitta
/phusebook
/phusetube
/company/phuse