DATA+AI SUMMIT 2022

Discover Data Lakehouse With End-To-End Lineage

Tao Feng
Staff Software Engineer, Databricks

ORGANIZED BY databricks

Product Safe Harbor Statement

About me

Tao Feng

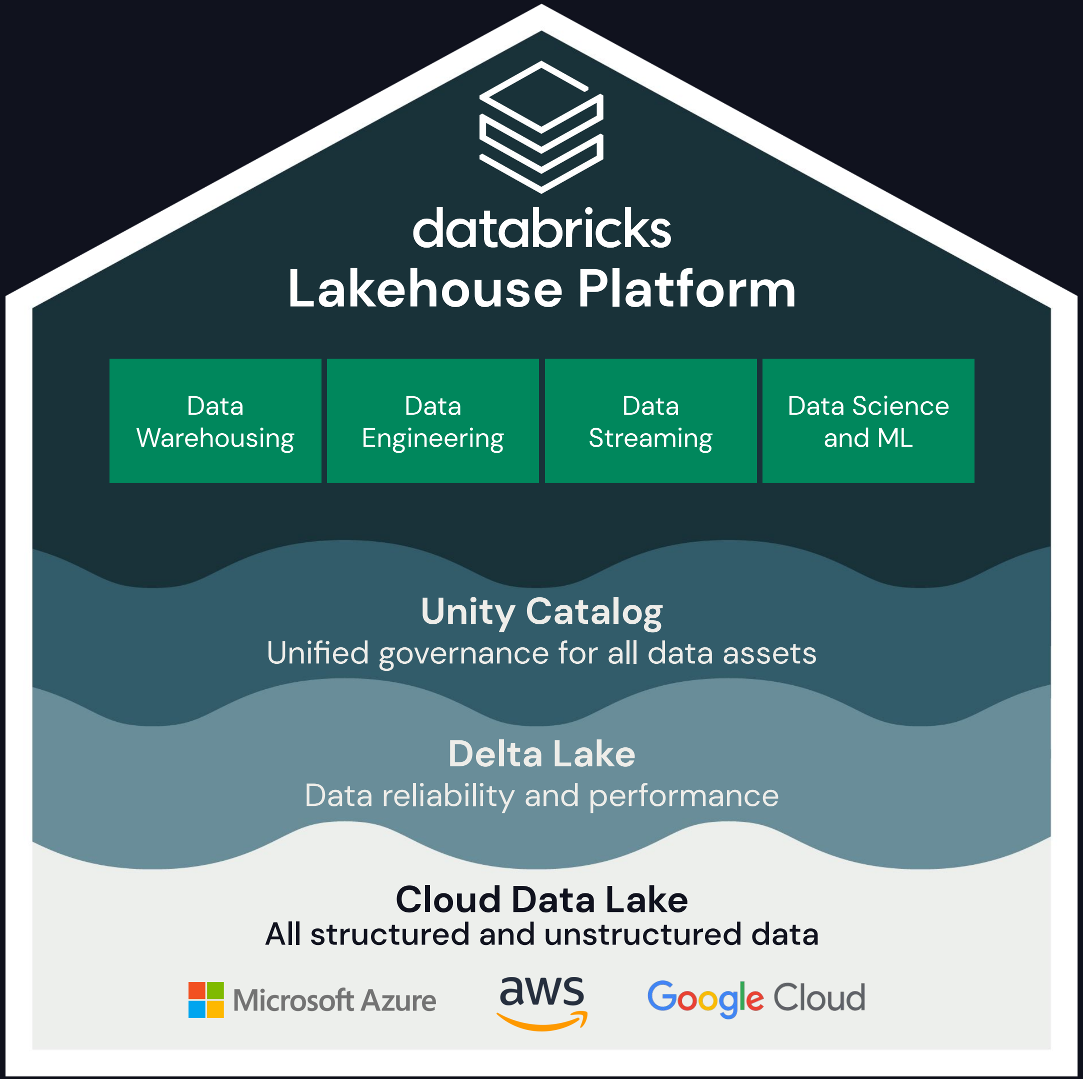
- Staff Engineer at Databricks
- Working on Data Discovery and Lineage
- Co-Creator of Amundsen and Apache Airflow PMC
- Previously worked at Lyft, and various other tech companies

Agenda

- Background
- Lineage Demo
- Lineage Deep Dive
- Lineage Roadmap

Background

What Is Data Lakehouse



Databricks Lakehouse Platform

Simple

Unify your data warehousing and AI use cases on a single platform

Multicloud

One consistent data platform across clouds

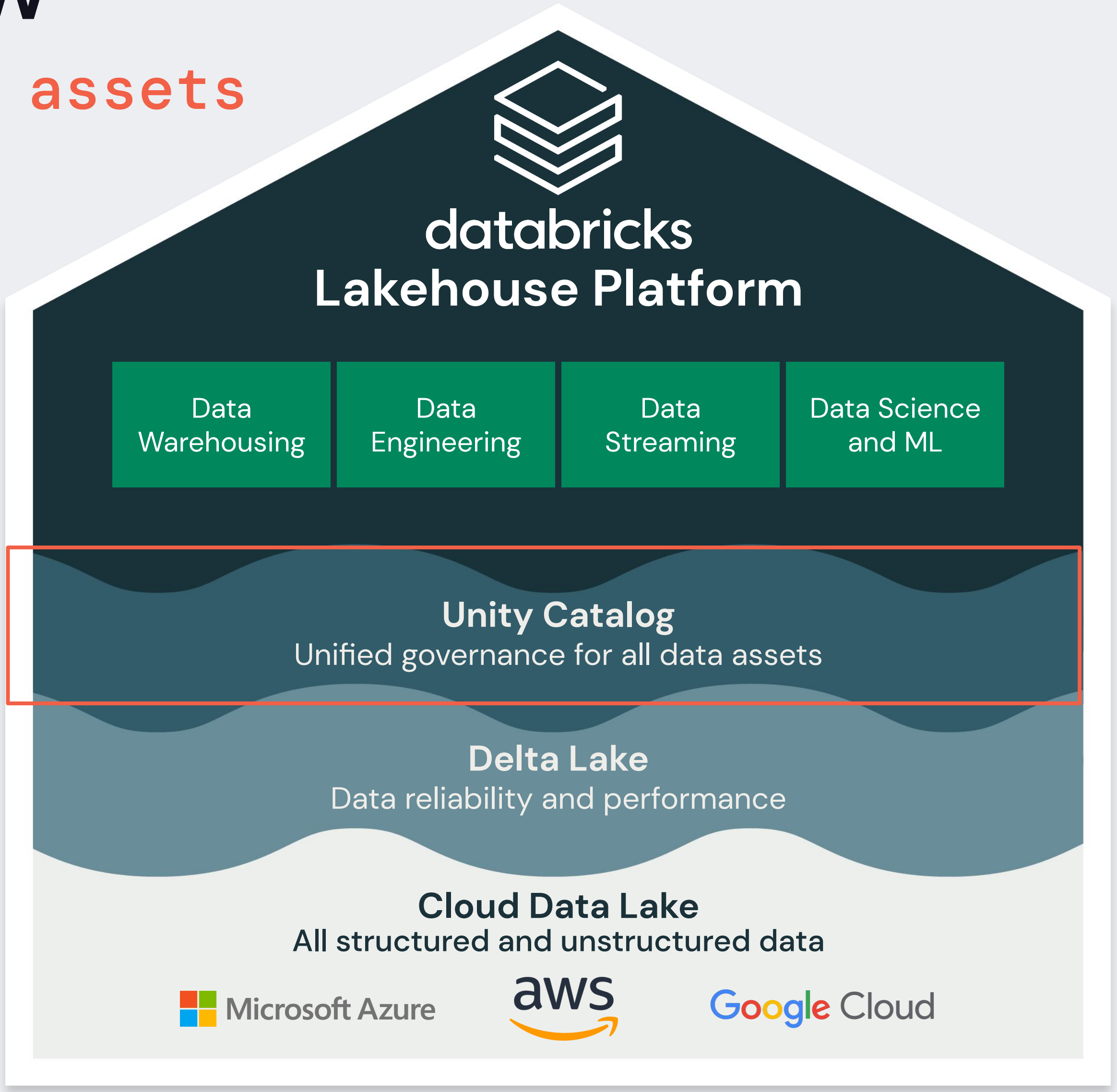
Open

Built on open source and open standards

Unity Catalog – Overview

Unified governance for all data assets

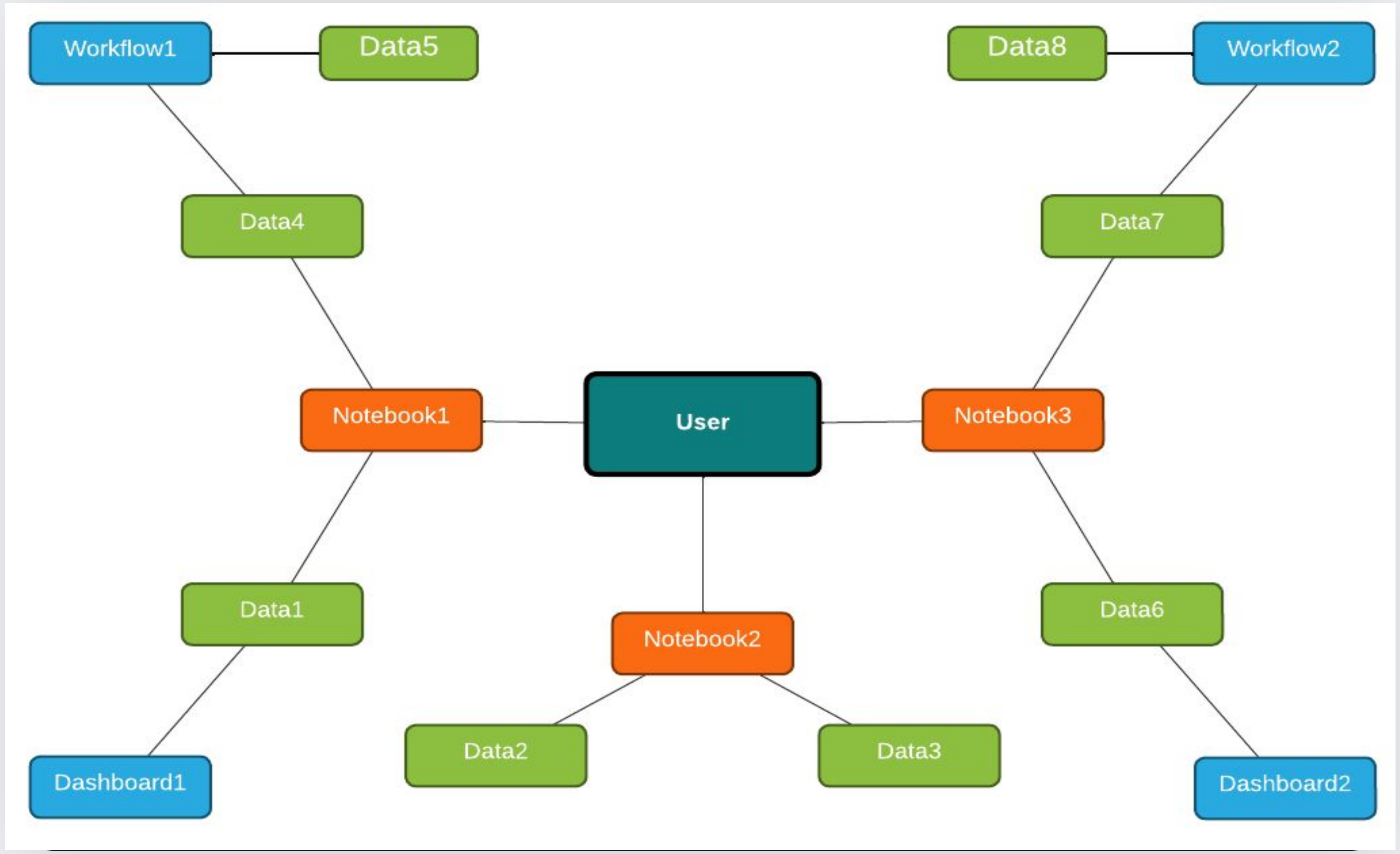
- Centralized governance for Data and AI
- Built-in search and discovery
- Performance and scalability
- Automated lineage for all workloads
- Integrated with your existing tools



What Is Data Lineage

What Is Data Lineage

- Lineage is a **graph** that connects different **data entities** in the Lakehouse and tracks their **dependencies**



Lakehouse Use Cases

Lakehouse Personas

Analysts

Data Scientists

Product Managers

Compliance Legals

Engineers

Machine Learning Practitioners

Use Case 1: Data Discovery and Analysis

Data Consumer

Analysts

Data Scientists

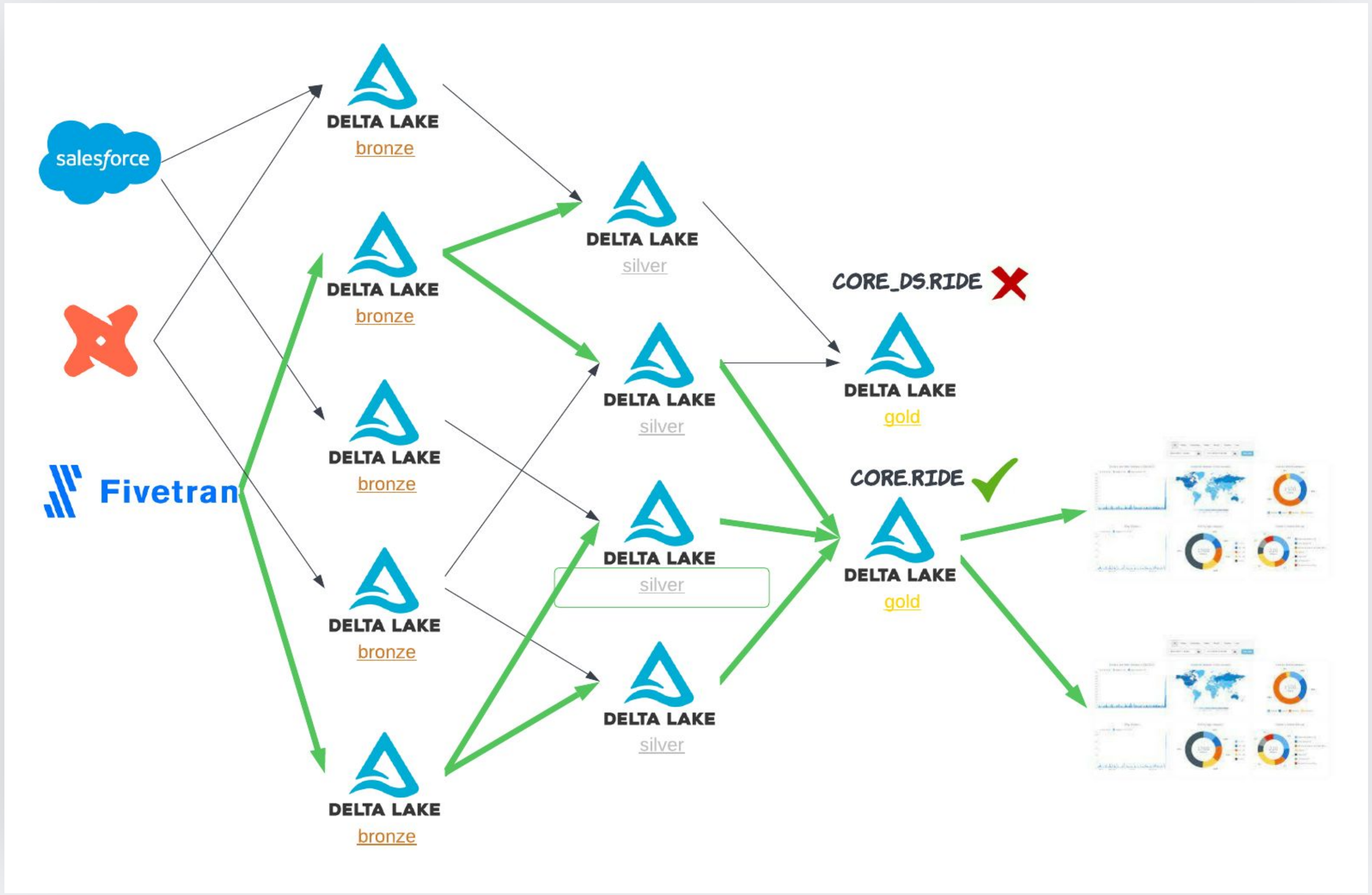
Product Managers

Machine Learning Practitioners

- Explore and understand data with its context and document and its origins
- Find out who are using the data
- Figure out what are the data producers and the consumers
- Make data driven decisions

Use Case 1: Data Discovery and Analysis

- How do I discovery the **trustworthy** data to use for my analysis?



Use Case 2: Change Management

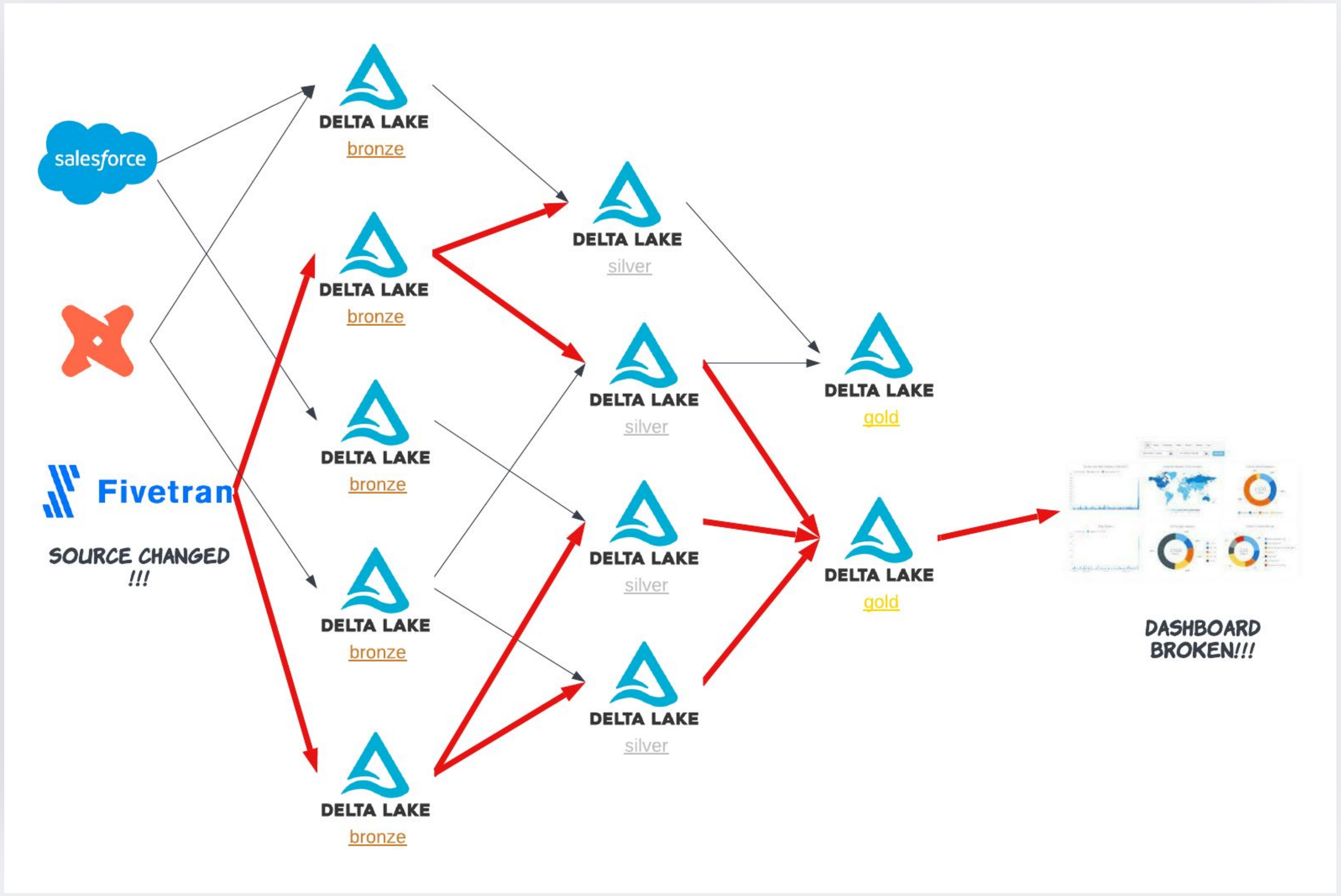
Data Producer

Engineers

- Identify data quality issue of a data artifact (e.g. dashboard, dataset)
- Deprecate column with downstream usage / Impact analysis
- Triage unused dashboards / artifacts and reduces costs

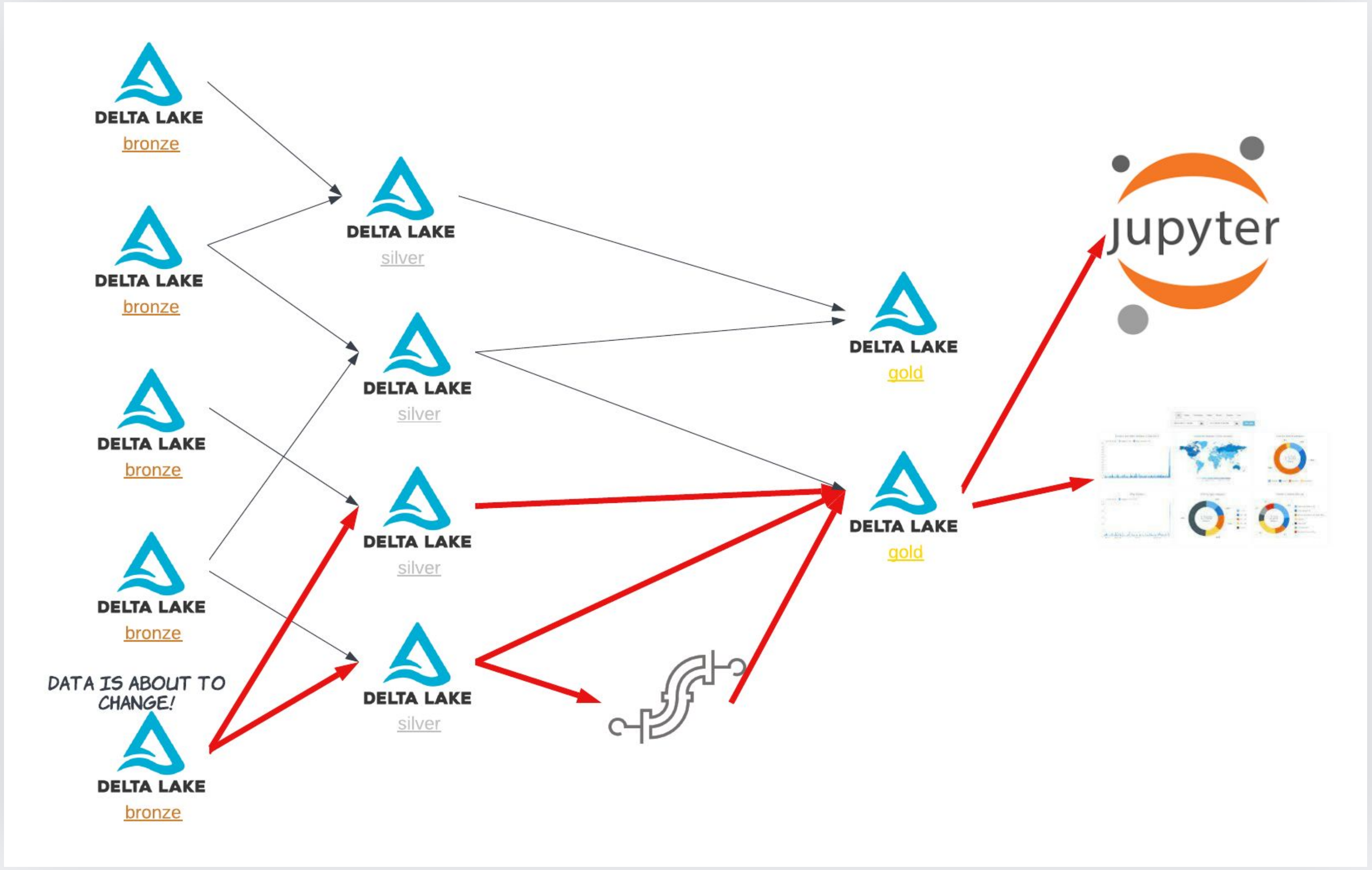
Use Case 2: Change Management

- Biz dashboard is broken, how to identify the **culprit**?



Use Case 2: Change Management

- Now we need to deprecate a column of a source table, what entities we need to change? And who should we notify?



Use Case 3: Data Governance

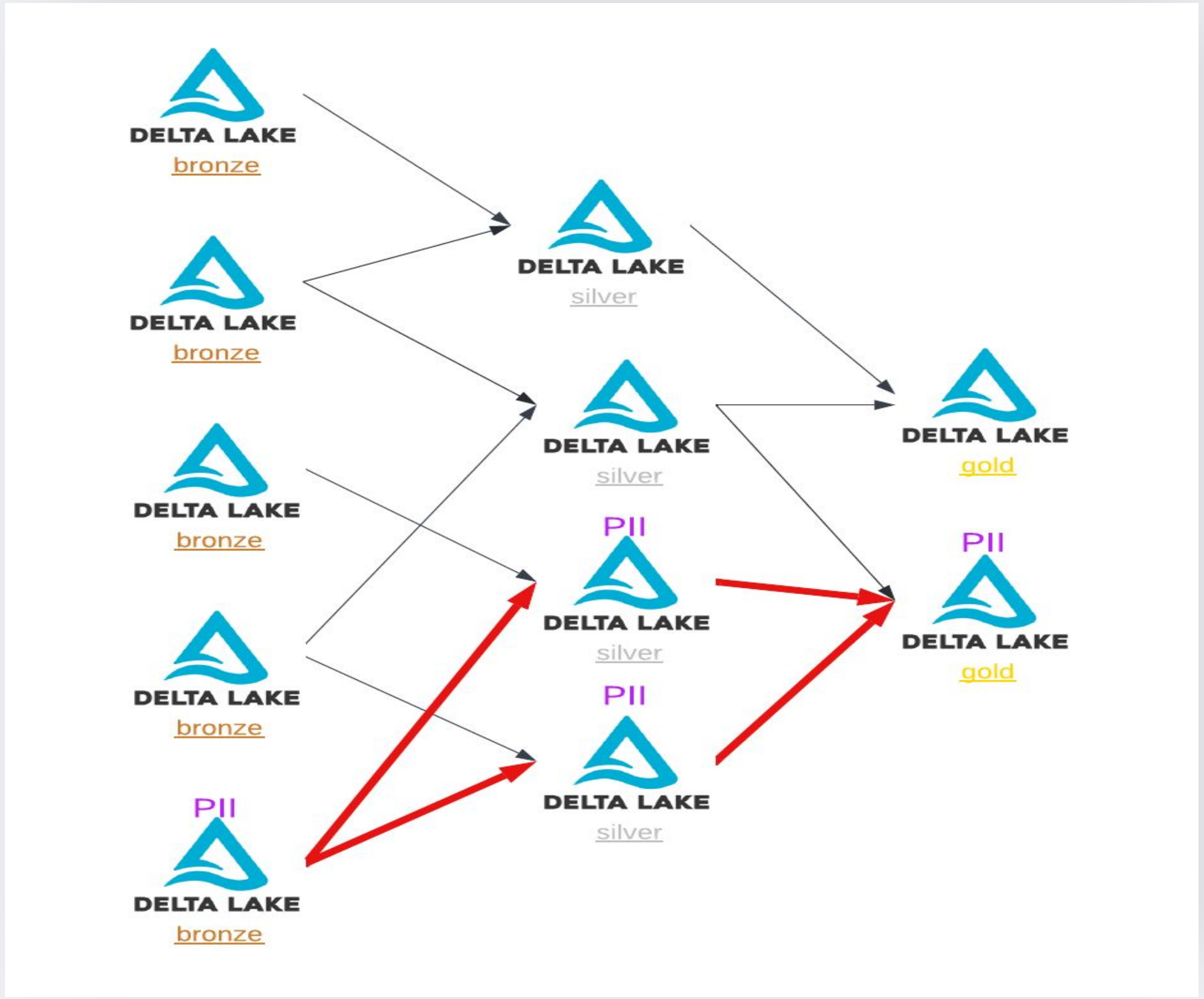
Compliance

Compliance Legals

- Identify PII or other sensitive information within the lakehouse

Use Case 3: Data Governance

- How to identify all the tables that have PII?



Why Is Lineage Important

Why Is Data Lineage Important

Compliance	***Discovery***	***Data Observability***
> Regulatory requirements to verify data lineage > Track the spread of sensitive data across datasets	> Understand context and trustworthiness of data before using in analytics > Prevent duplicative work and datasets	> Track down issues / discrepancies in reports by tracing back the data > Analyze impact of proposed changes to downstream reports eg column deprecation

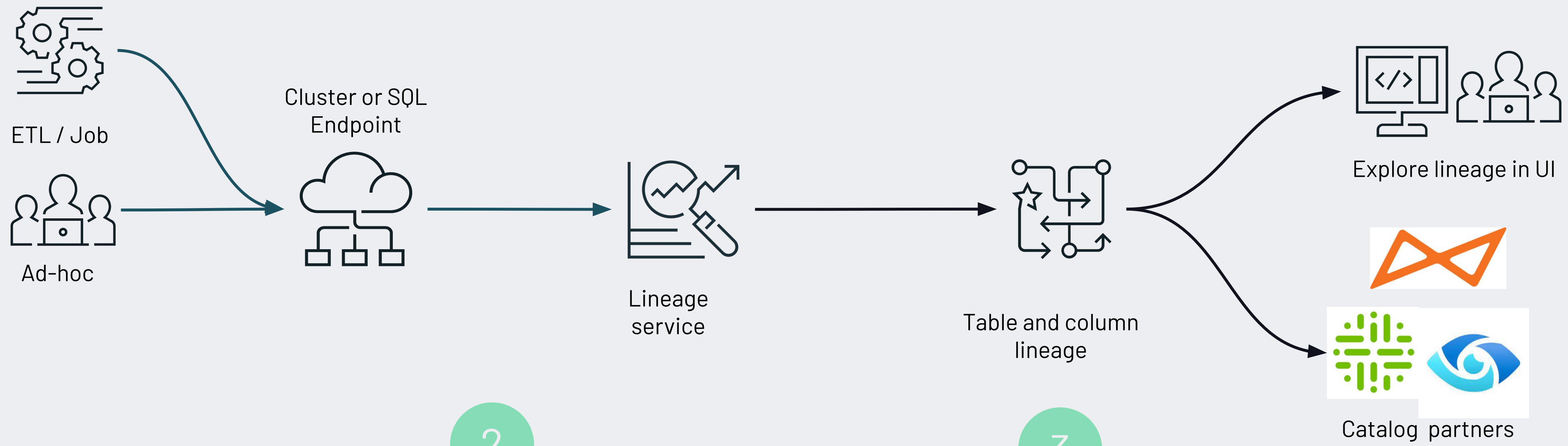
Demo

Demo

DATA+AI SUMMIT 2022

Deep Dive

Data Lineage With Unity Catalog



1

Code (any language) is submitted to a cluster or SQL Warehouse endpoint

2

Lineage service analyzes lineage events emitted from the cluster / SQL Endpoint

Assembles column and table level lineage

3

Presented to the end user graphically in Databricks

Lineage can be exported via API and imported into other catalog partners

Data Lineage With Unity Catalog

Auto-capture runtime data lineage across all languages

Track lineage down to the **table and column level**

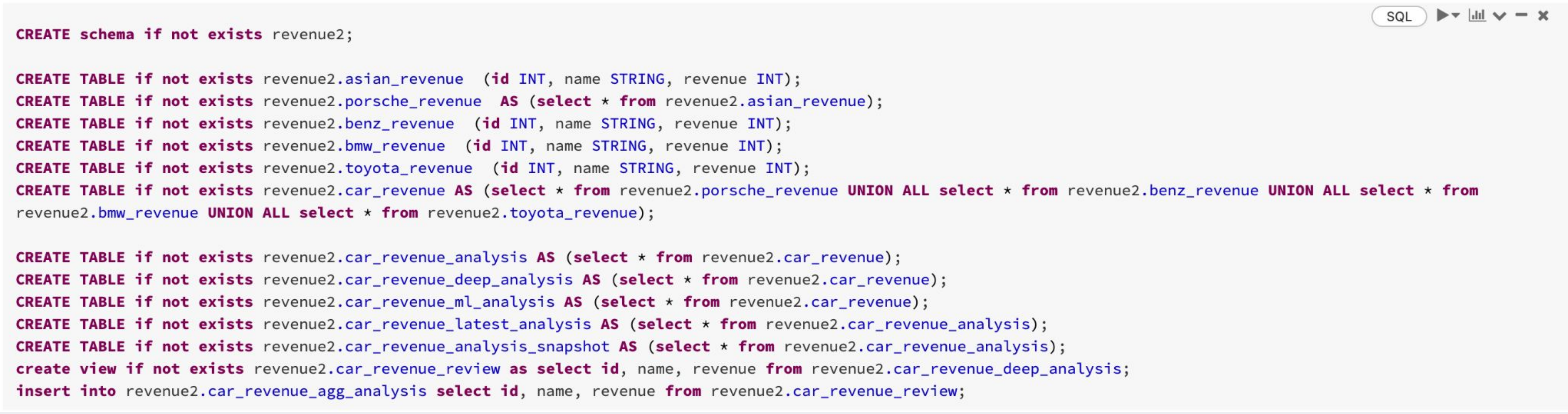
Govern access by **Unity Catalog** Permission Model

Pipelines to surface lineage in **near real-time**

Table/Column/Entity Lineage **Graph** Visualization

Lineage Example

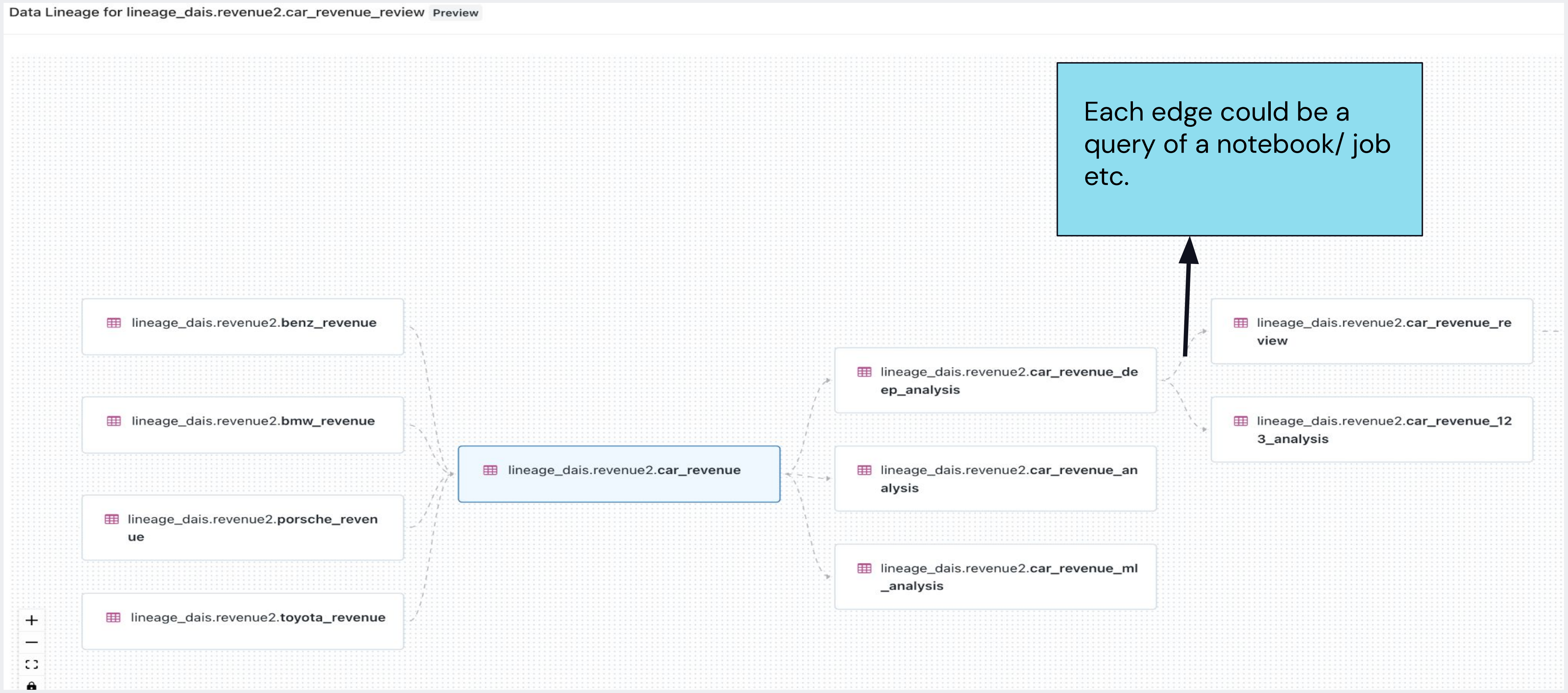
- Define car sales revenue table, aggregate table and an analysis view



```sql
CREATE schema if not exists revenue2;

CREATE TABLE if not exists revenue2.asian_revenue  (id INT, name STRING, revenue INT);
CREATE TABLE if not exists revenue2.porsche_revenue  AS (select * from revenue2.asian_revenue);
CREATE TABLE if not exists revenue2.benz_revenue  (id INT, name STRING, revenue INT);
CREATE TABLE if not exists revenue2.bmw_revenue  (id INT, name STRING, revenue INT);
CREATE TABLE if not exists revenue2.toyota_revenue  (id INT, name STRING, revenue INT);
CREATE TABLE if not exists revenue2.car_revenue AS (select * from revenue2.porsche_revenue UNION ALL select * from revenue2.benz_revenue UNION ALL select * from
revenue2.bmw_revenue UNION ALL select * from revenue2.toyota_revenue);

CREATE TABLE if not exists revenue2.car_revenue_analysis AS (select * from revenue2.car_revenue);
CREATE TABLE if not exists revenue2.car_revenue_deep_analysis AS (select * from revenue2.car_revenue);
CREATE TABLE if not exists revenue2.car_revenue_ml_analysis AS (select * from revenue2.car_revenue);
CREATE TABLE if not exists revenue2.car_revenue_latest_analysis AS (select * from revenue2.car_revenue_analysis);
CREATE TABLE if not exists revenue2.car_revenue_analysis_snapshot AS (select * from revenue2.car_revenue_analysis);
create view if not exists revenue2.car_revenue_review as select id, name, revenue from revenue2.car_revenue_deep_analysis;
insert into revenue2.car_revenue_agg_analysis select id, name, revenue from revenue2.car_revenue_review;
```

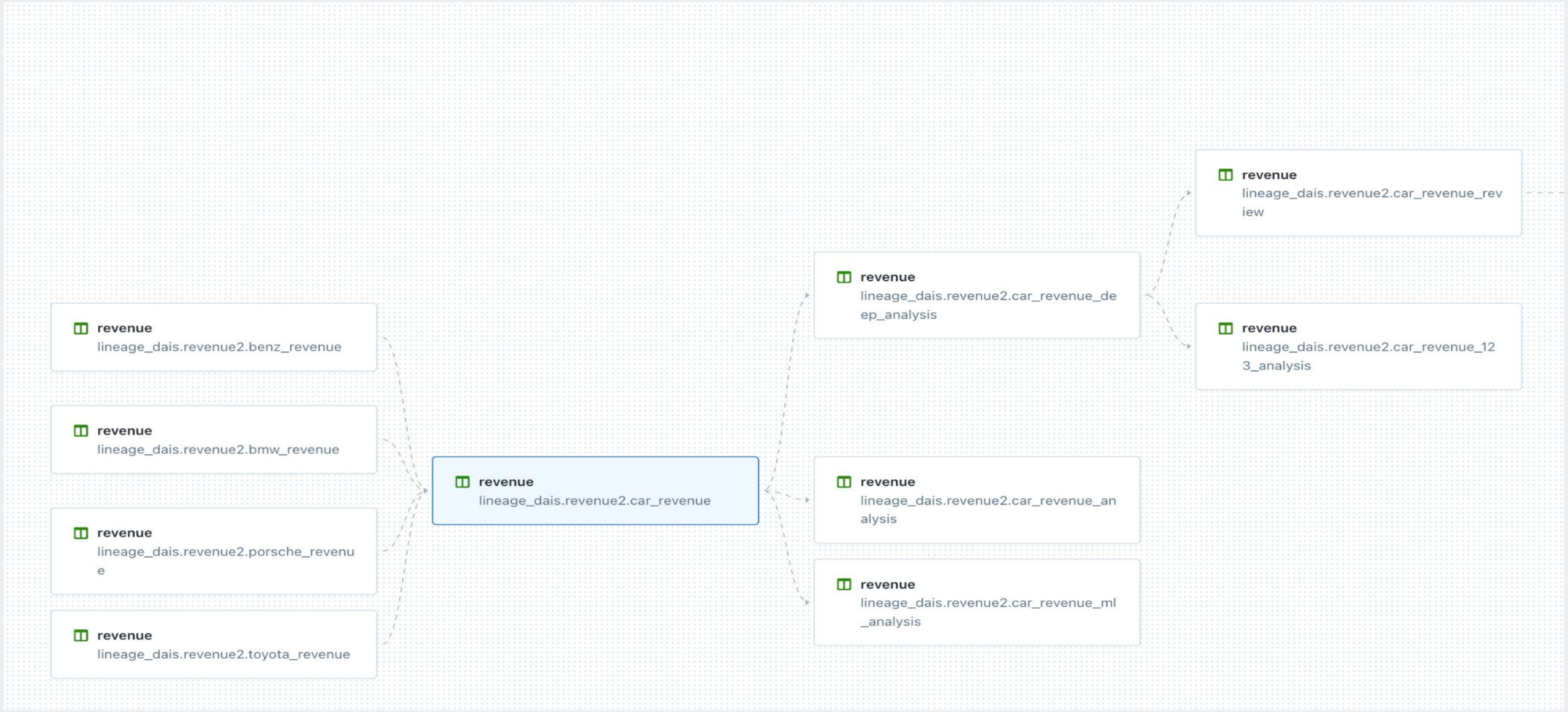
Lineage Example

Table level lineage


Data Lineage for lineage_dais.revenue2.car_revenue_review Preview
lineage_dais.revenue2.benz_revenue
lineage_dais.revenue2.bmw_revenue
lineage_dais.revenue2.porsche_revenue
lineage_dais.revenue2.toyota_revenue
lineage_dais.revenue2.car_revenue
lineage_dais.revenue2.car_revenue_deep_analysis
lineage_dais.revenue2.car_revenue_analysis
lineage_dais.revenue2.car_revenue_ml_analysis
lineage_dais.revenue2.car_revenue_review
lineage_dais.revenue2.car_revenue_123_analysis
Each edge could be a query of a notebook/ job etc.

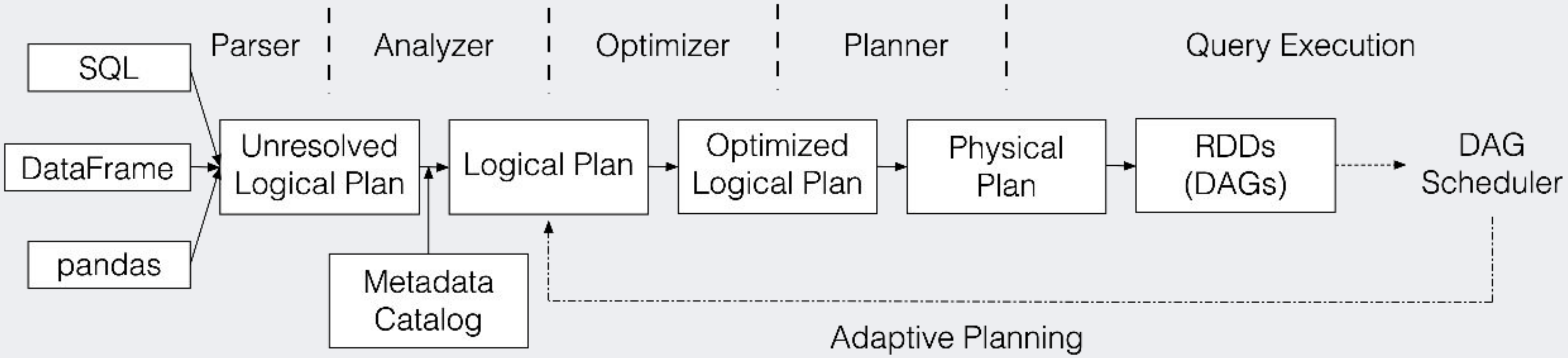
Lineage Example

Column level lineage


revenue
lineage_dais.revenue2.benz_revenue
revenue
lineage_dais.revenue2.bmw_revenue
revenue
lineage_dais.revenue2.porsche_revenue
revenue
lineage_dais.revenue2.toyota_revenue
revenue
lineage_dais.revenue2.car_revenue
revenue
lineage_dais.revenue2.car_revenue_deep_analysis
revenue
lineage_dais.revenue2.car_revenue_analysis
revenue
lineage_dais.revenue2.car_revenue_ml_analysis
revenue
lineage_dais.revenue2.car_revenue_review
revenue
lineage_dais.revenue2.car_revenue_123_analysis

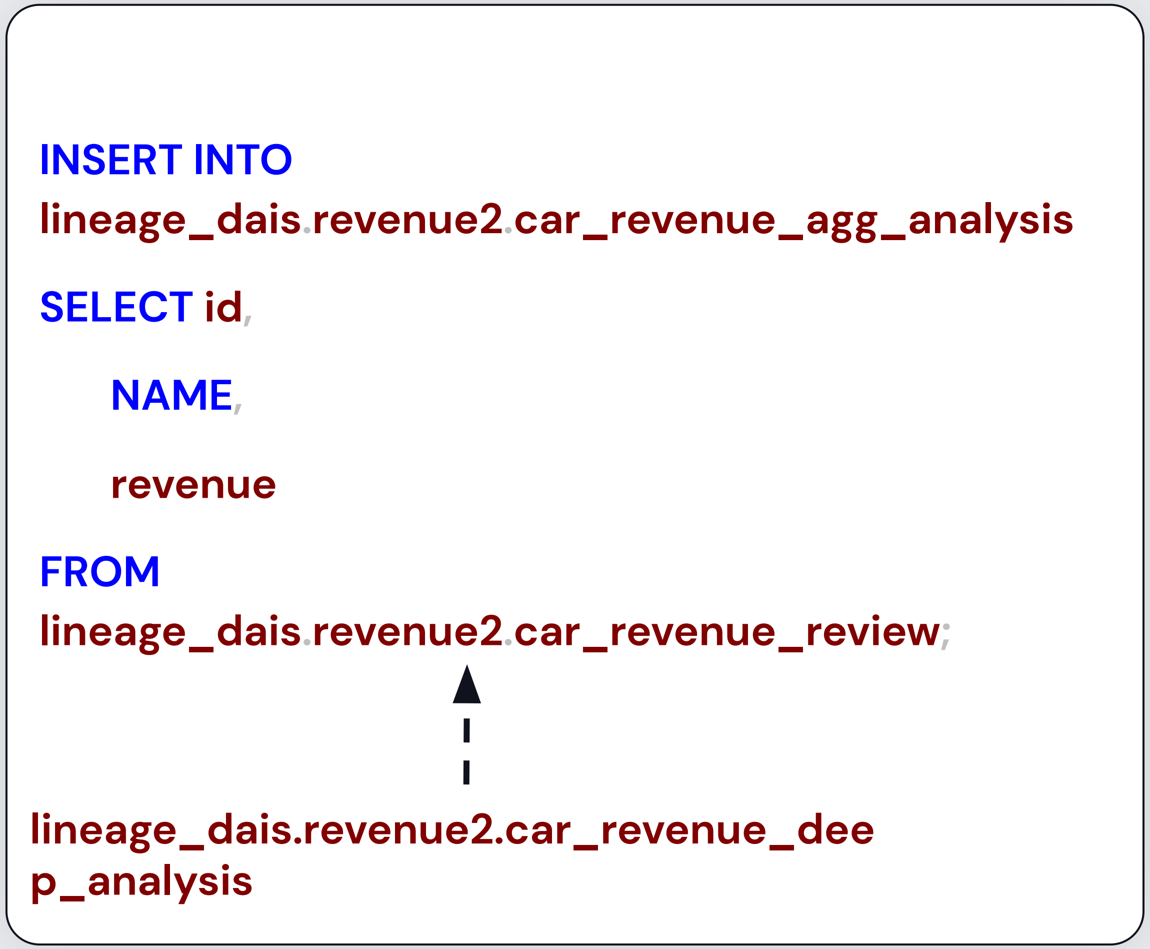
How to capture the lineage of a query?

Life of a Query in Spark



- SQL Language
- Dataset/DataFrame/Pandas APIs: richer, language-integrated and user-friendly interfaces

Tree: Abstractions of Users' Programs



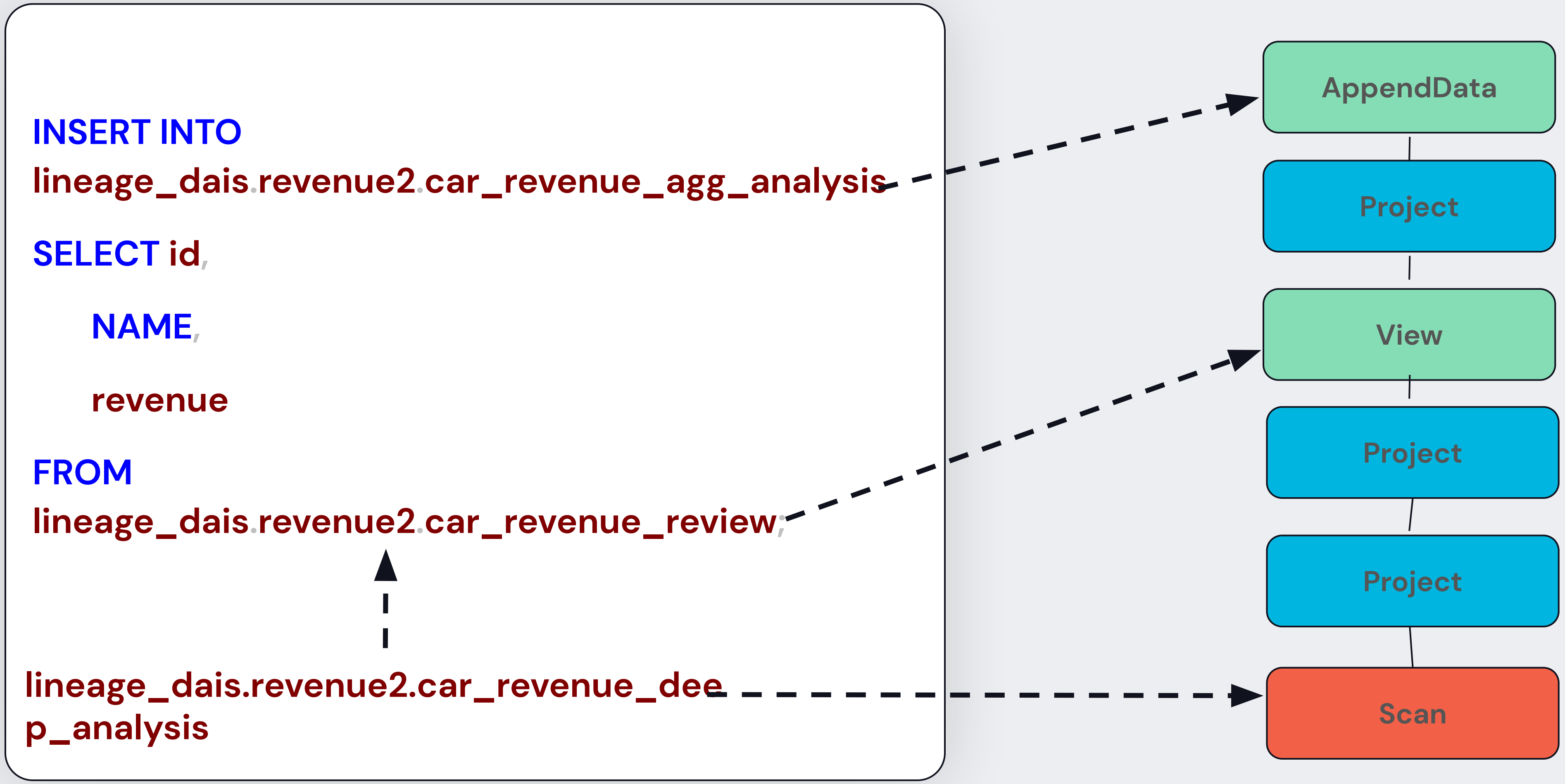
```
INSERT INTO
lineage_dais.revenue2.car_revenue_agg_analysis
SELECT id,
    NAME,
    revenue
FROM
lineage_dais.revenue2.car_revenue_review;
```

lineage_dais.revenue2.car_revenue_deep_analysis

- **Expression:** An expression represents a new value, computed based on input values
- **Attribute**: A column of a dataset (e.g id) or a column generated by a specific data operation

DATA+AI SUMMIT 2022

Tree: Abstractions of Users' Programs

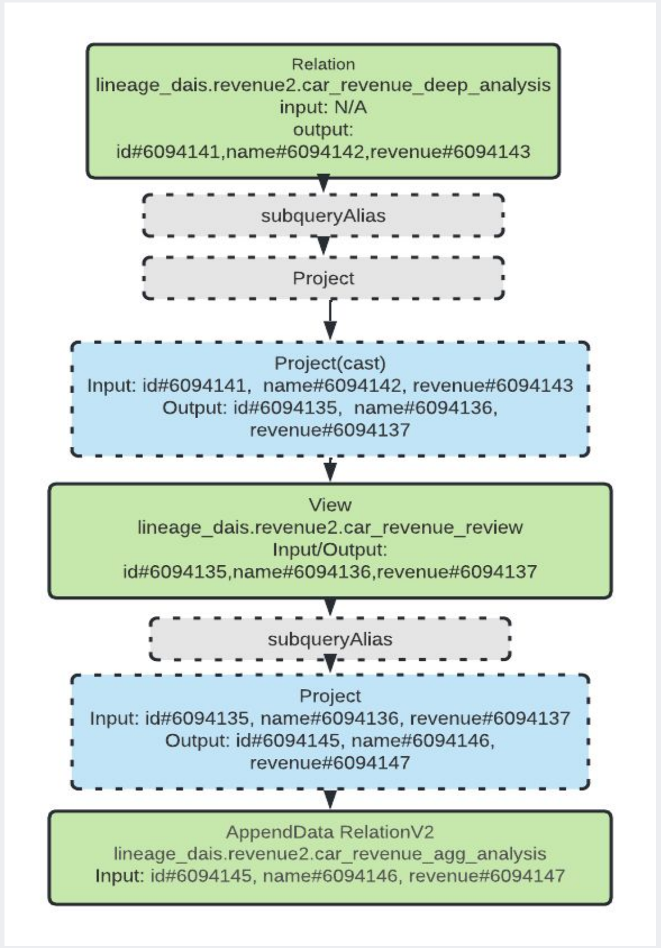


```
INSERT INTO
lineage_dais.revenue2.car_revenue_agg_analysis
SELECT id,
      NAME,
      revenue
FROM
lineage_dais.revenue2.car_revenue_review;


lineage_dais.revenue2.car_revenue_dee
p_analysis
```

Analyzed Logical Plan

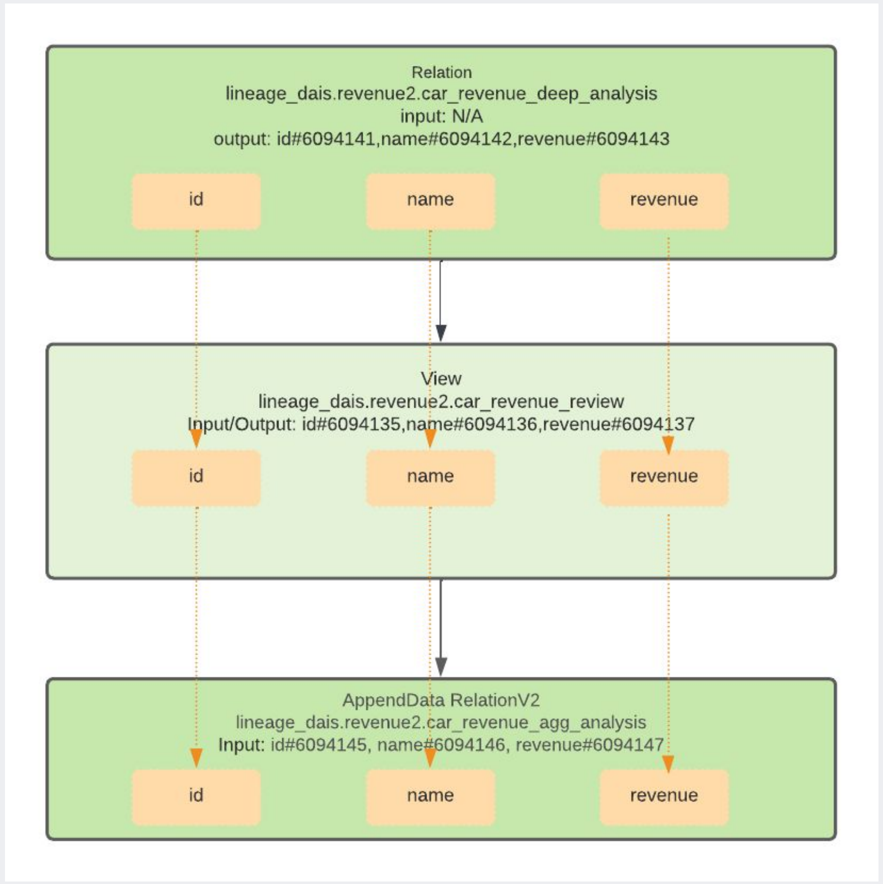
```
INSERT INTO
lineage_dais.revenue2.car_revenue_agg_analysis

SELECT id,

      NAME,

      revenue

FROM
lineage_dais.revenue2.car_revenue_review;
```

lineage_dais.revenue2.car_revenue_dee
p_analysis


Relation
lineage_dais.revenue2.car_revenue_deep_analysis
input: N/A
output:
id#6094141,name#6094142,revenue#6094143
subqueryAlias
Project
Project(cast)
Input: id#6094141, name#6094142, revenue#6094143
Output: id#6094135, name#6094136, revenue#6094137
View
lineage_dais.revenue2.car_revenue_review
Input/Output:
id#6094135,name#6094136,revenue#6094137
subqueryAlias
Project
Input: id#6094135, name#6094136, revenue#6094137
Output: id#6094145, name#6094146, revenue#6094147
AppendData RelationV2
lineage_dais.revenue2.car_revenue_agg_analysis
Input: id#6094145, name#6094146, revenue#6094147

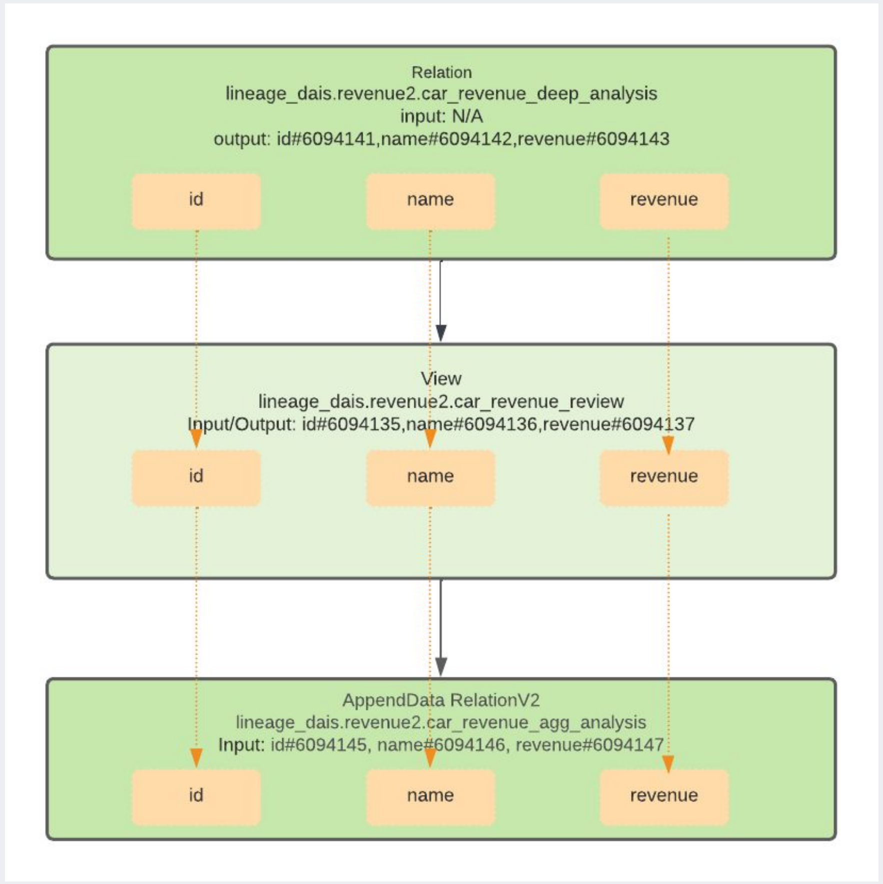
Capture Lineage

- **Criteria**
  - Capture table name with 3L namespace (catalog.schema.table)
  - Capture persistent view with source table dependence
  - Capture target column and its source columns for column level lineage
  - Don't capture lineage unless the command executed successfully
  - Shouldn't throw exception during command execution


Relation
lineage_dais.revenue2.car_revenue_deep_analysis
input: N/A
output: id#6094141,name#6094142,revenue#6094143
id
name
revenue
View
lineage_dais.revenue2.car_revenue_review
Input/Output: id#6094135,name#6094136,revenue#6094137
id
name
revenue
AppendData RelationV2
lineage_dais.revenue2.car_revenue_agg_analysis
Input: id#6094145, name#6094146, revenue#6094147
id
name
revenue

Capture Lineage

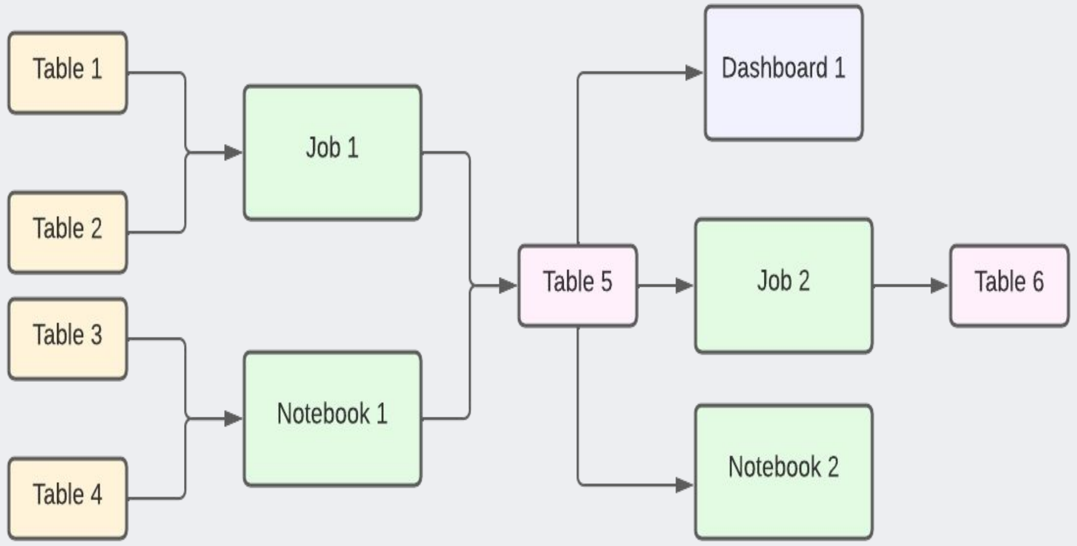
- **Challenges**
  - No existing data for table / column level heuristic validation
  - No generic solution to support all Spark use cases
  - Support across different Spark versions


Relation
lineage_dais.revenue2.car_revenue_deep_analysis
input: N/A
output: id#6094141,name#6094142,revenue#6094143
id
name
revenue
View
lineage_dais.revenue2.car_revenue_review
Input/Output: id#6094135,name#6094136,revenue#6094137
id
name
revenue
AppendData RelationV2
lineage_dais.revenue2.car_revenue_agg_analysis
Input: id#6094145, name#6094146, revenue#6094147
id
name
revenue

How to capture the entity lineage?

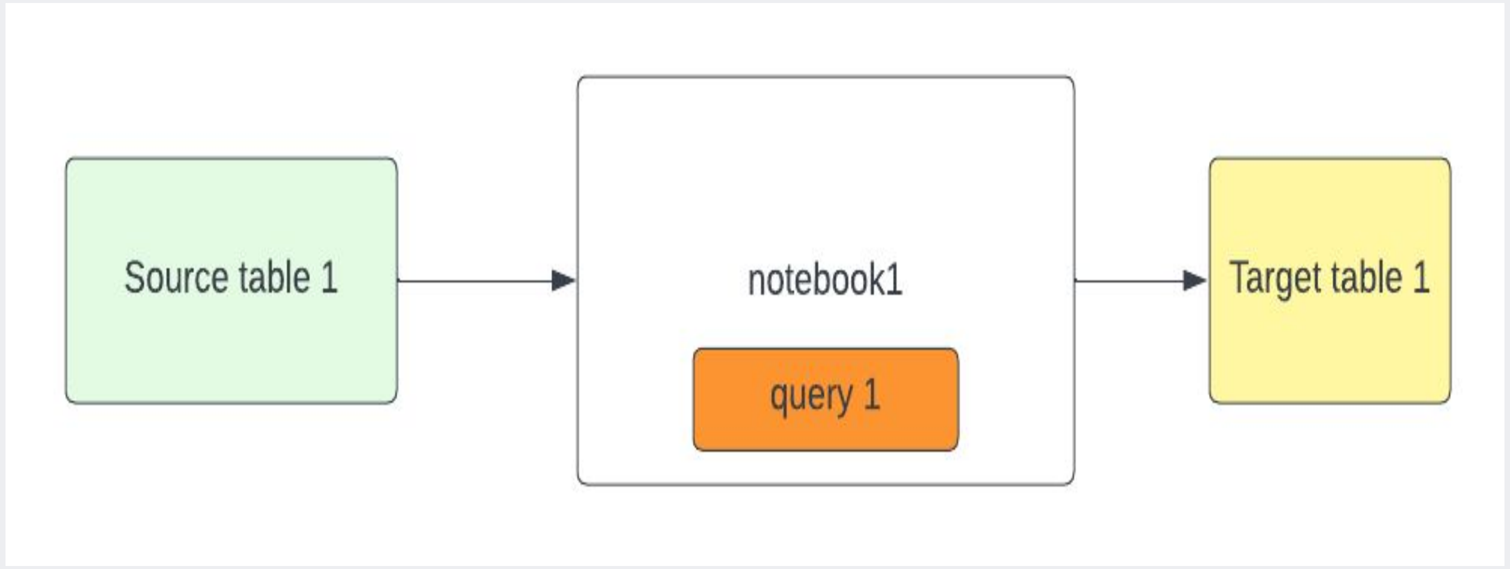
Entity Lineage

- Capture all entity artifacts lineage within the lakehouse
  - Initial entities: notebook, Job and DBSQL Dashboard
- Two main use cases:
  - **Transformer** : Notebook and Job consumes bunches of input tables and produces output tables
  - **Consumer**: Notebook and DBSQL dashboard only consumes tables and produces artifacts(e.g BI Dashboard)



How to capture the entity lineage

- Capture data lineage along with entity type + entity ID
- Capture the lineage even for the read query
- Respect entity ACL
- Link back to the original entity



Details

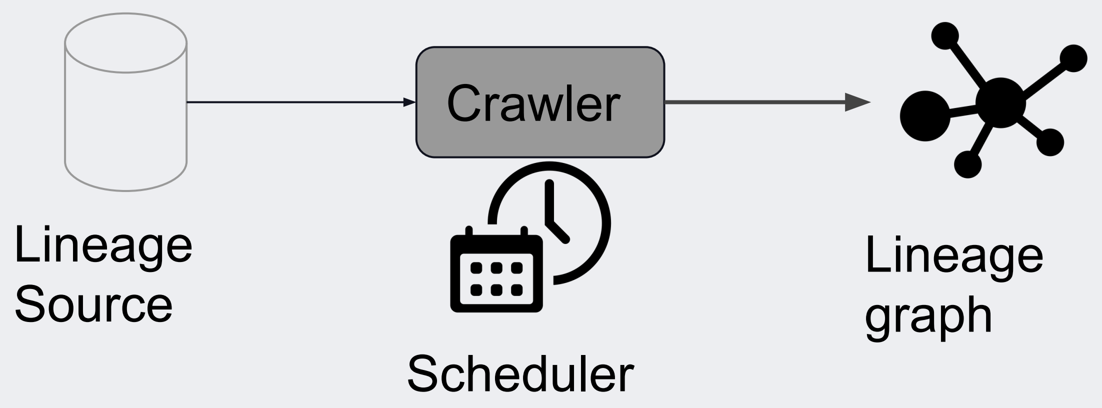
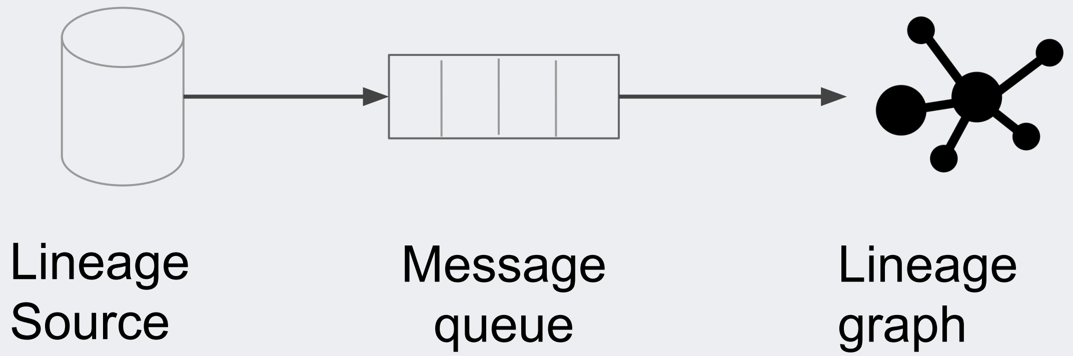
#1 How to represent entities in the lineage

- Lineage is like a graph
- Entity type + Unique ID / name + Scope to uniquely identify the entity in the lineage graph
  - Workspace level entity: Workspace ID + Entity Type + Entity ID
  - Account level entity (shared across workspace): Metastore ID + Table + Column

#2 What is the approach to collect lineage

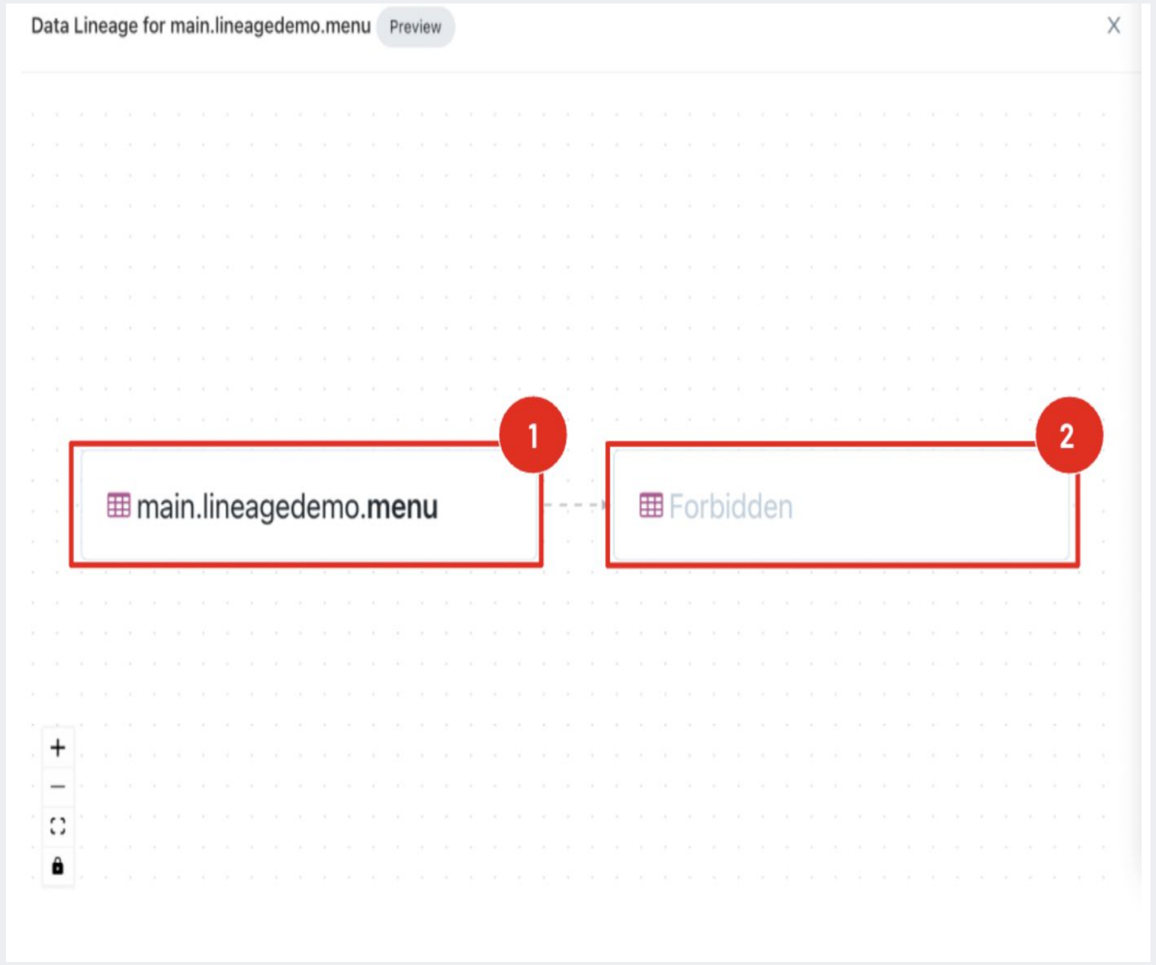
Pattern	**Description**	**Example**	**Key Benefit**	**Key Challenge**
Manual linked by User	Manual added and described how datasets are linked			Does not scale
Inferred from SQL / SQL parsing	Programmatic extracting lineage with SQL dialect	https://github.com/uber/queryparser	Accurate, supports all sql dialect	SQL is easier, but can't support Spark which has multi lang API
Auto capture during runtime	**Language agnostic , auto captured in query execution phase**	**Lineage in Unity catalog**	● **Correct** ● **Handle multi-lang**	

#3 What is the latency to update lineage

ETL Batch Update	Neal realtime Update
	
Pros • The graph could be fully rebuilt	**Pros** • The graph only keeps 1 level upstream / dowstream of entity • Provide lineage timely to end users

#4 What is the permission model of lineage

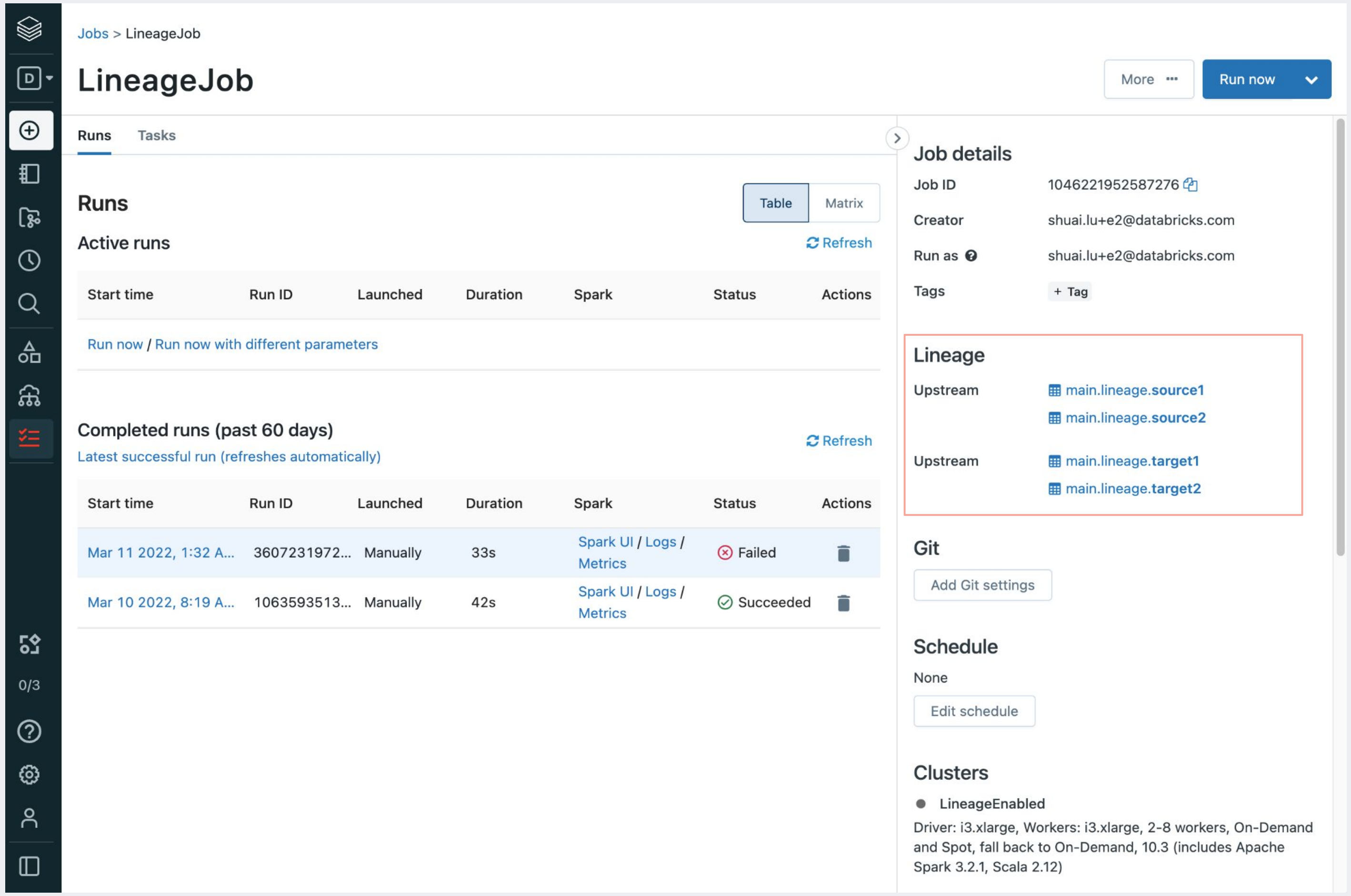
- Table Lineage leverages Unity Catalog permission model to view lineage
- Entity(notebook, workflow, dashboard) Lineage subjects to the entity permission ACL model



Roadmap

In Context Lineage

- Lineage should be context aware.
- Lineage in job page
- Debugging and impact analysis



Jobs > LineageJob

LineageJob

Runs

Active runs Refresh

Start time	Run ID	Launched	Duration	Spark	Status	Actions
Run now / Run now with different parameters						

Completed runs (past 60 days) Refresh

Latest successful run (refreshes automatically)

Start time	Run ID	Launched	Duration	Spark	Status	Actions
Mar 11 2022, 1:32 A...	3607231972...	Manually	33s	Spark UI / Logs / Metrics	Failed	
Mar 10 2022, 8:19 A...	1063593513...	Manually	42s	Spark UI / Logs / Metrics	Succeeded	

Job details

Job ID	1046221952587276
Creator	shuai.lu+e2@databricks.com
Run as	shuai.lu+e2@databricks.com
Tags	+ Tag

Lineage

Upstream	main.lineage.source1 main.lineage.source2
Upstream	main.lineage.target1 main.lineage.target2

Git

Add Git settings

Schedule

None

Edit schedule

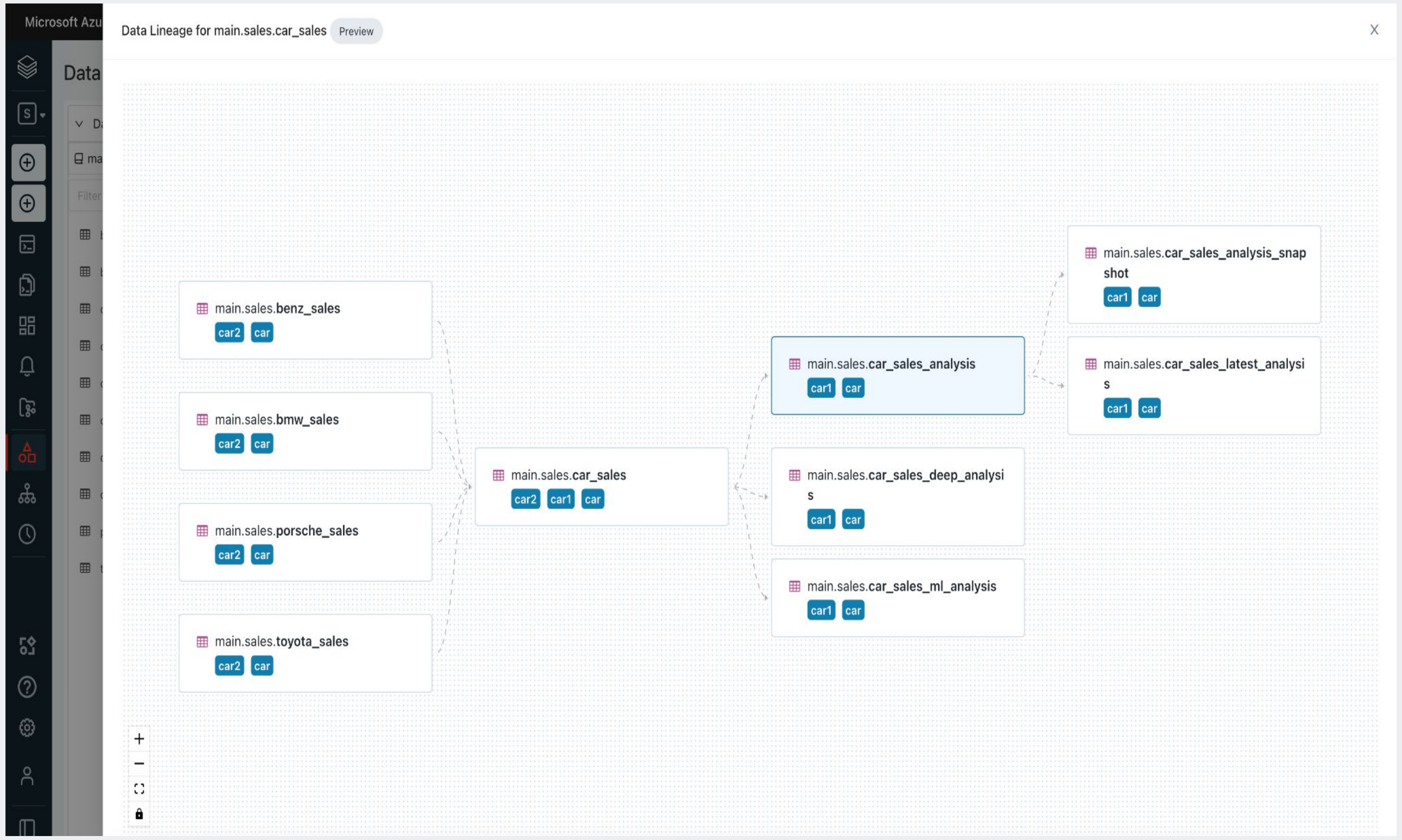
Clusters

LineageEnabled

Driver: i3.xlarge, Workers: i3.xlarge, 2-8 workers, On-Demand and Spot, fall back to On-Demand, 10.3 (includes Apache Spark 3.2.1, Scala 2.12)

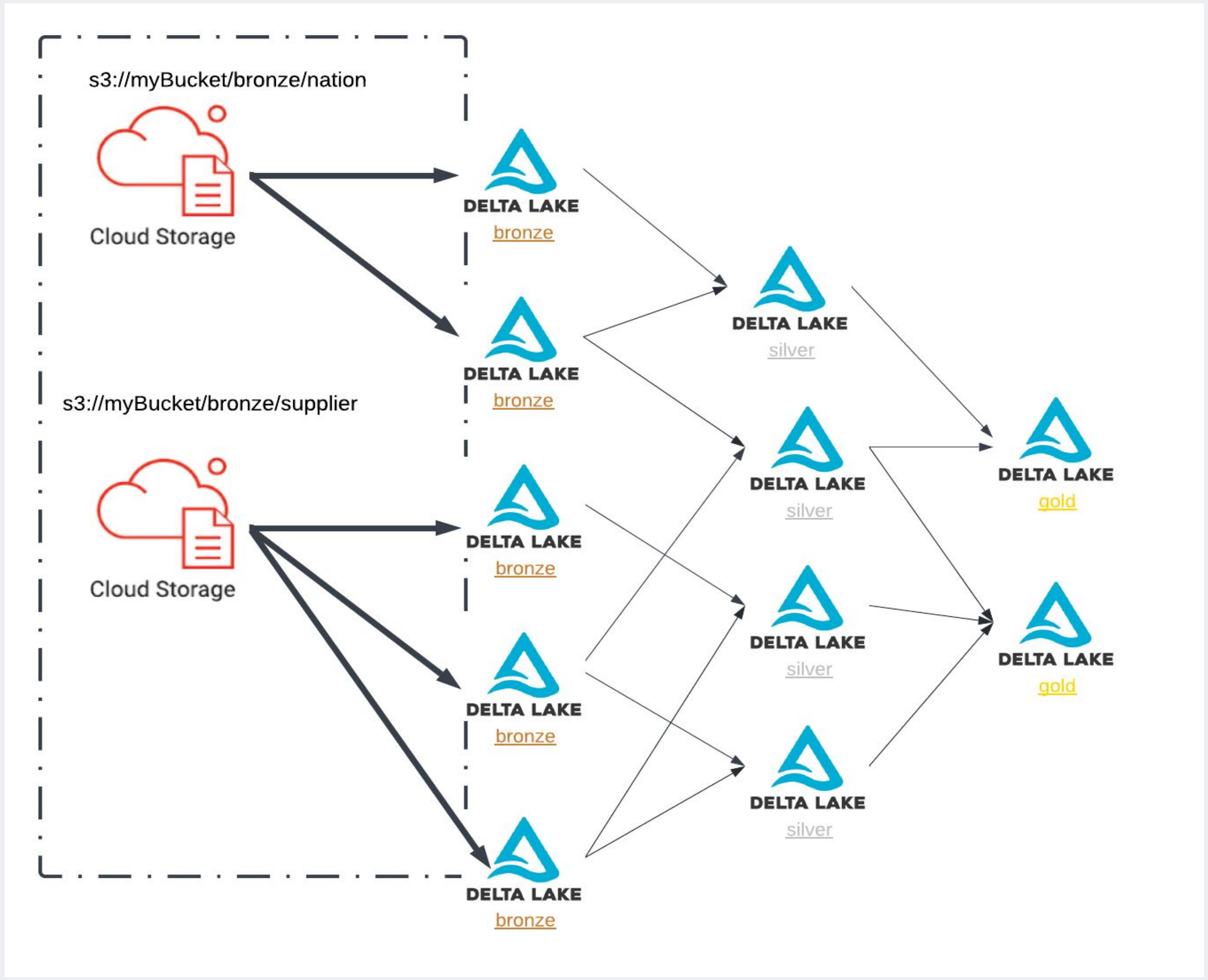
Tag Propagation

- Propagate tag through lineage automatically


Microsoft Azu
Data Lineage for main.sales.car_sales
Preview
Data
main.sales.benz_sales
car2
car
main.sales.bmw_sales
car2
car
main.sales.porsche_sales
car2
car
main.sales.toyota_sales
car2
car
main.sales.car_sales
car2
car1
car
main.sales.car_sales_analysis
car1
car
main.sales.car_sales_deep_analysis
car1
car
main.sales.car_sales_ml_analysis
car1
car
main.sales.car_sales_analysis_snapshot
car1
car
main.sales.car_sales_latest_analysis
car1
car

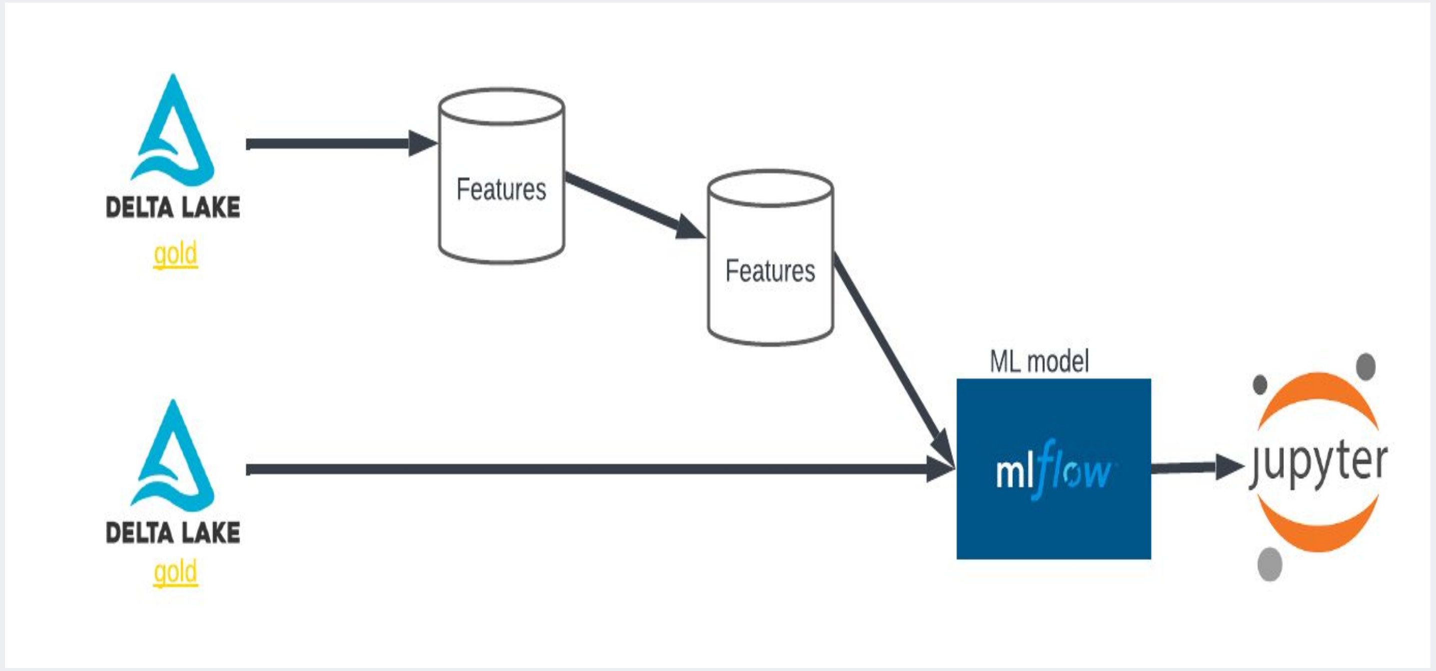
Files Lineage

- Capture file lineage of the `first-mile ETL` which the ETL writes to cloud storage first

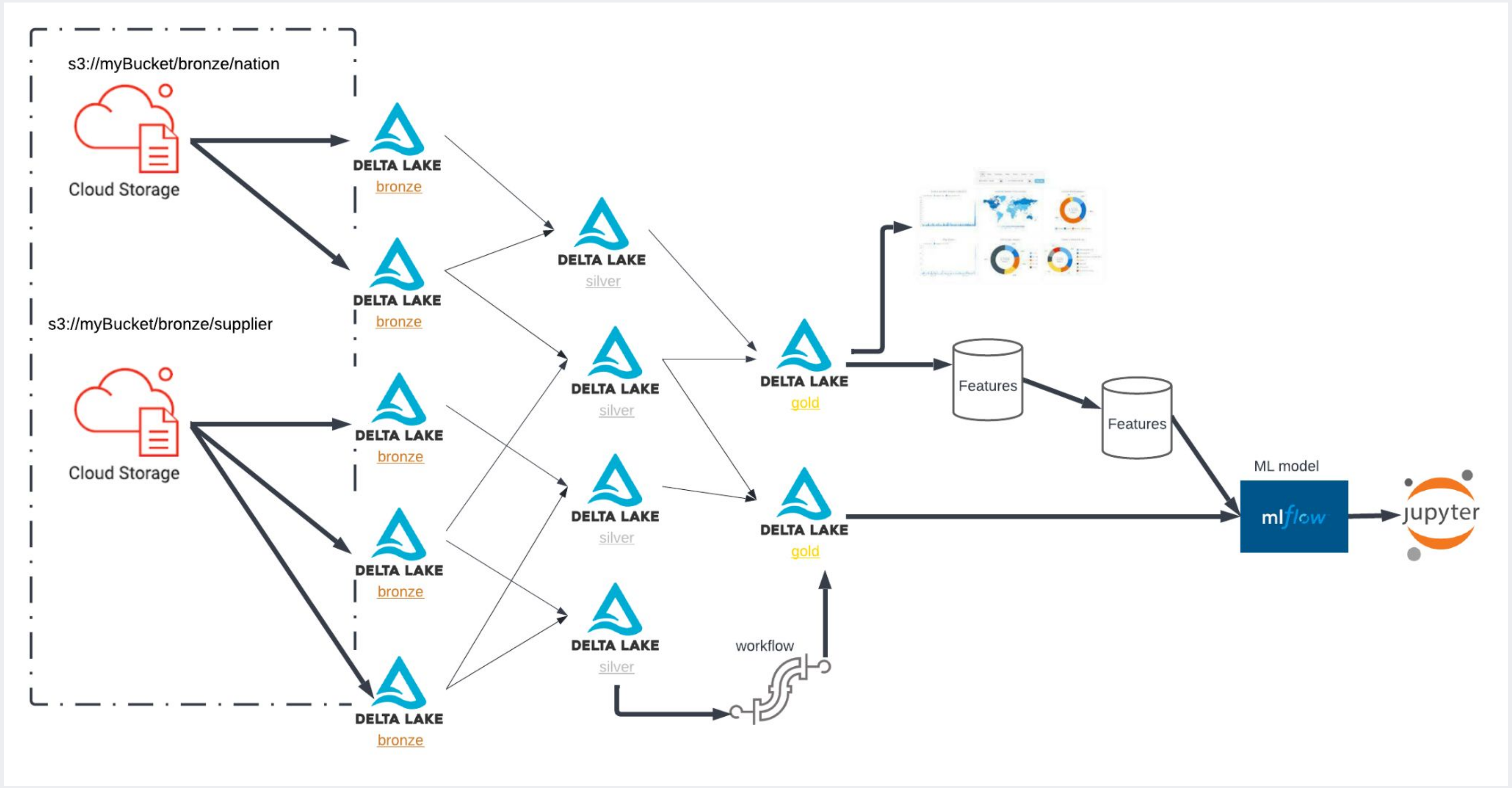


ML Lineage

- Provide end-to-end lineage from dataset to features to MLflow models



Lakehouse E2E Lineage



DATA+AI
SUMMIT 2022

Thank you

Tao Feng

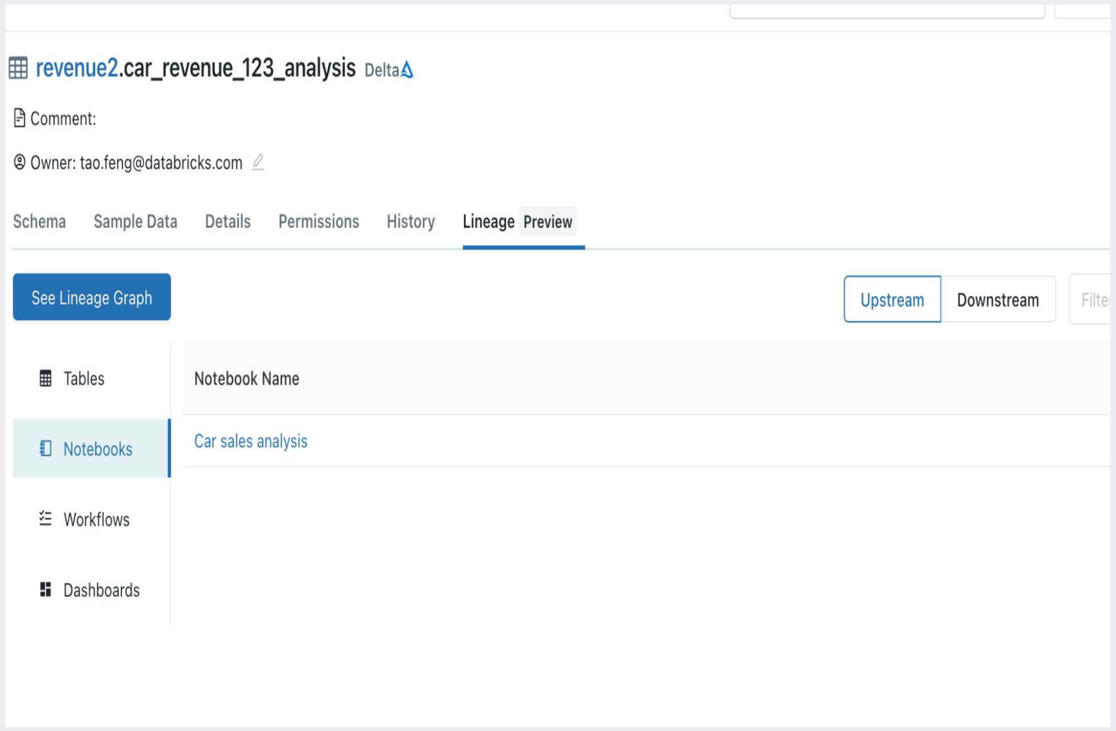
Staff Software Engineer, Databricks

ORGANIZED BY databricks

Appendix

Notebook Lineage

- Currently table centric
- Capture upstream / downstream notebooks of a given table
- Respect Notebook ACL
- Could direct back to the original notebook



revenue2.car_revenue_123_analysis Delta

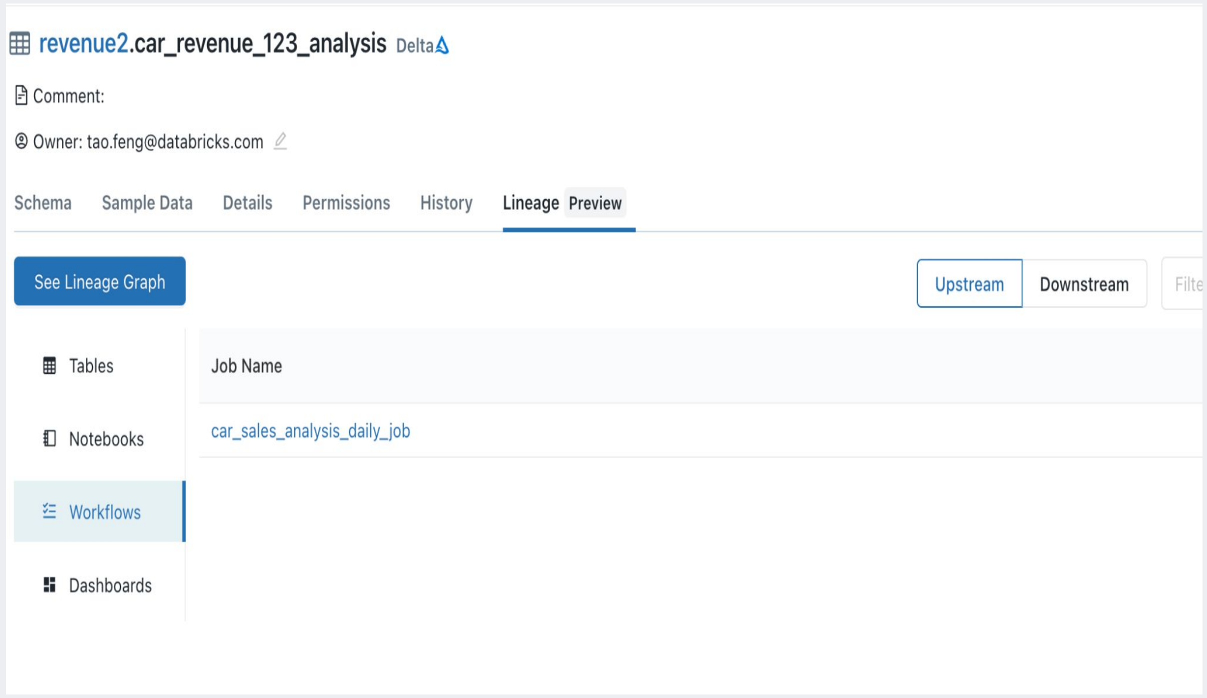
Comment:

Owner: tao.feng@databricks.com

Notebook Name
Car sales analysis

Workflow Lineage

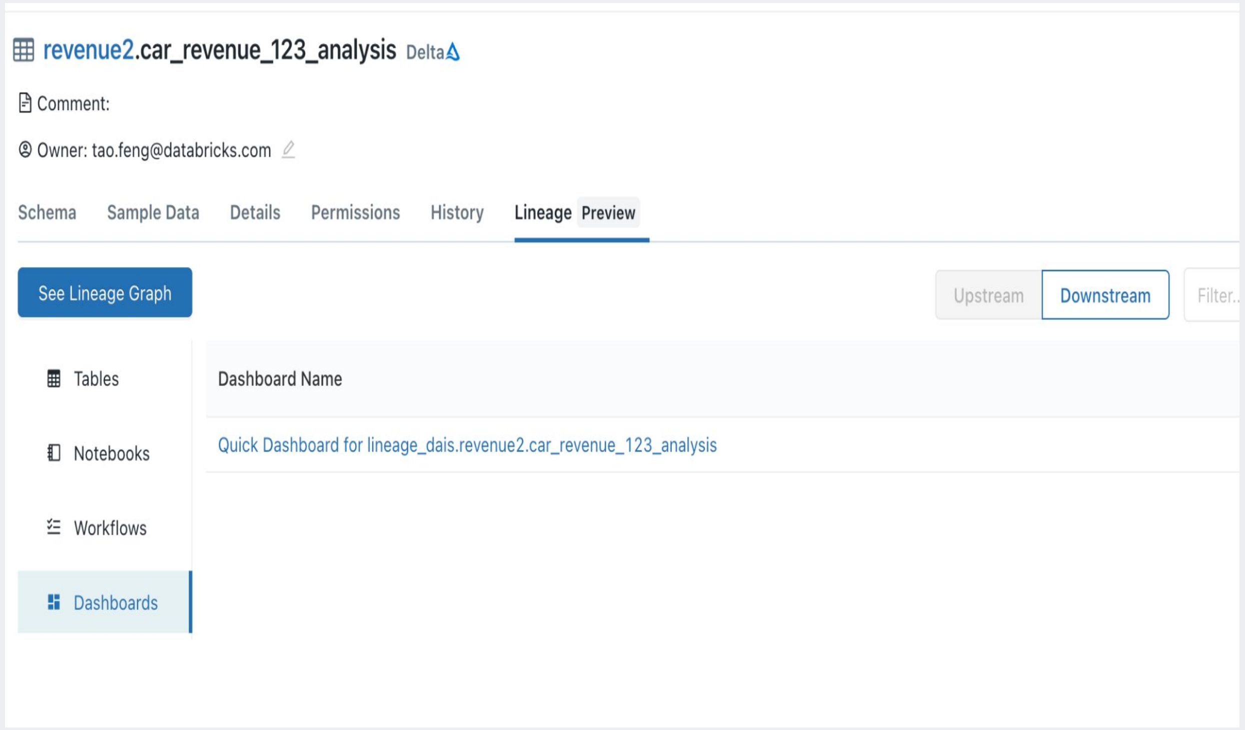
- Capture upstream / downstream workflow / job of a given table
- Could surface task level dependency if needed
- Respect workflow / job ACL
- Could direct back to workflow / job page



Job Name
car_sales_analysis_daily_job

Dashboard Lineage

- Capture downstream dashboard of a given table
- Respect dashboard ACL
- Easy to find out impacted dashboard if the table has issues


revenue2.car_revenue_123_analysis Delta

Comment:

Owner: tao.feng@databricks.com

Dashboard Name
Quick Dashboard for lineage_dais.revenue2.car_revenue_123_analysis