

Design and implementation of data science pipelines

A new paradigm based on analytics engineers



**Politecnico
di Torino**

Ferdinando Micco
Advisor : Clemente Cetera
Supervisor : Paolo Garza

Index



1

Problem



2

Proposed Solution



3

Proof of Concept



4

Achievements

Problem

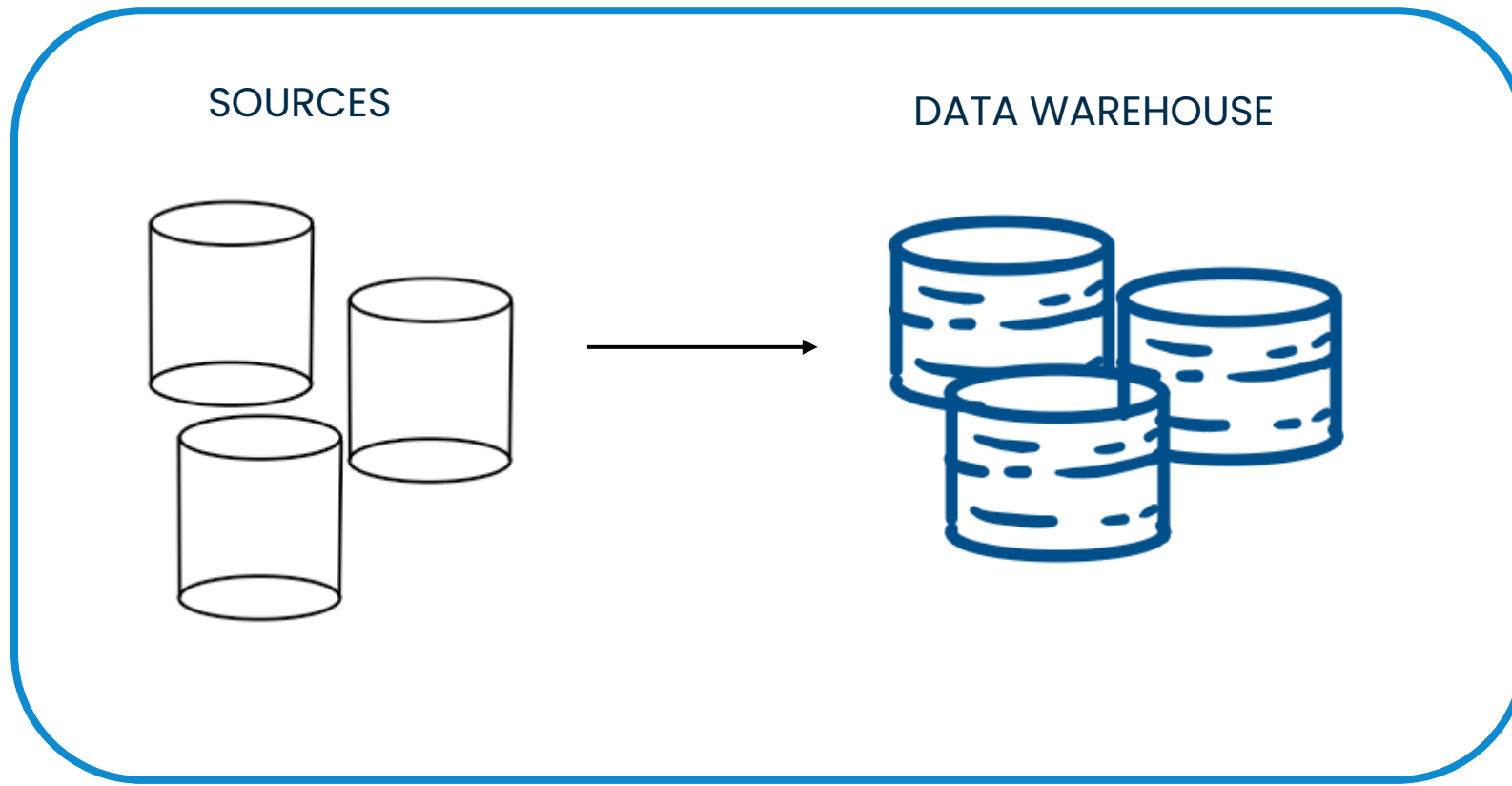
The gap between
business and IT

Data engineer

Data analyst

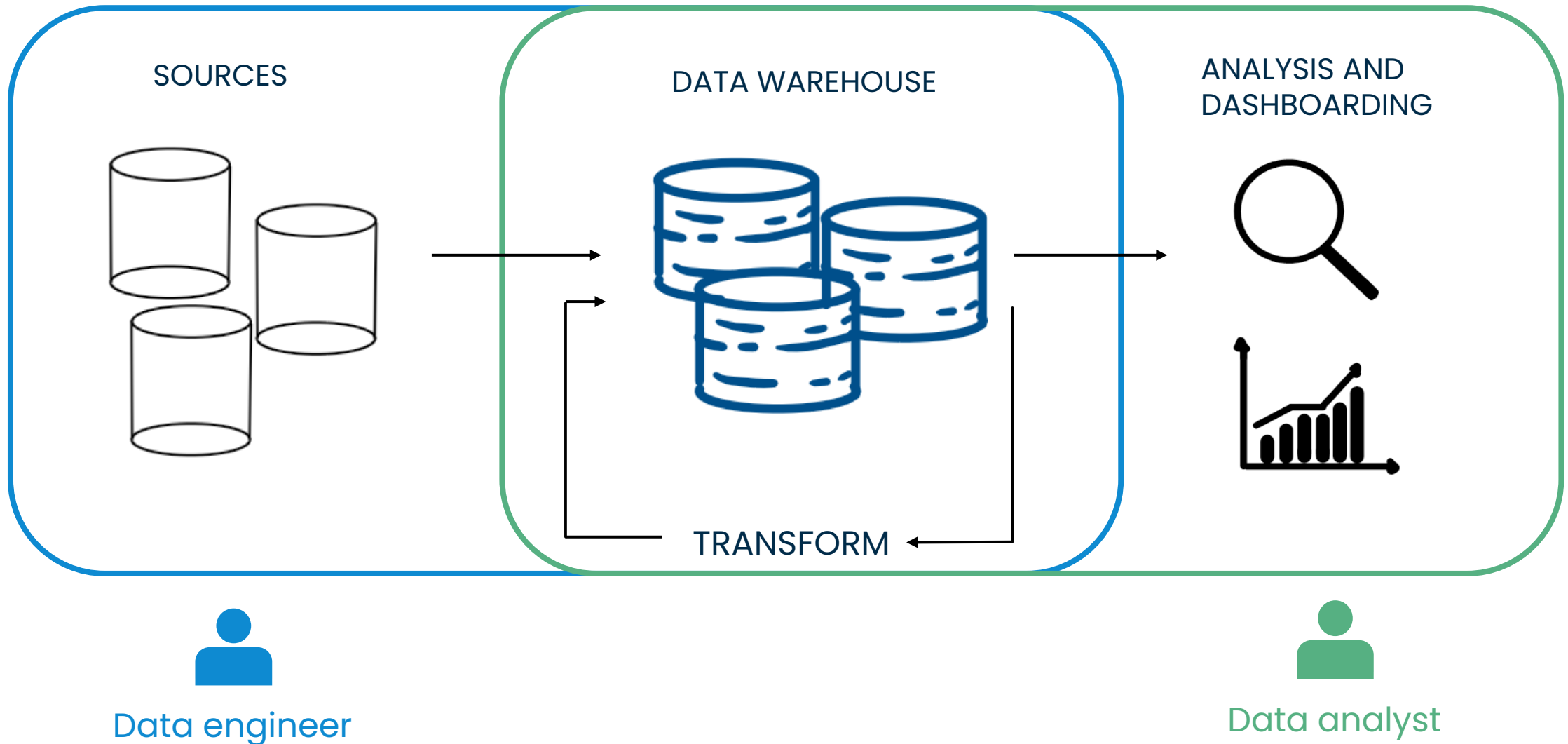


The gap between bussiness and IT



Data engineer

The gap between bussiness and IT



The gap between bussiness and IT

Technical expertise



Analysis skills



The gap between bussiness and IT

Technical expertise



Data engineer

- Build and maintain the data platform
- Build custom data ingestion integration
- Develop and deploy machine learning algorithm
- Data warehouse performance optimization

Analysis skills

The gap between bussiness and IT

Technical expertise



Data engineer

- Build and maintain the data platform
- Build custom data ingestion integration
- Develop and deploy machine learning algorithm
- Data warehouse performance optimization

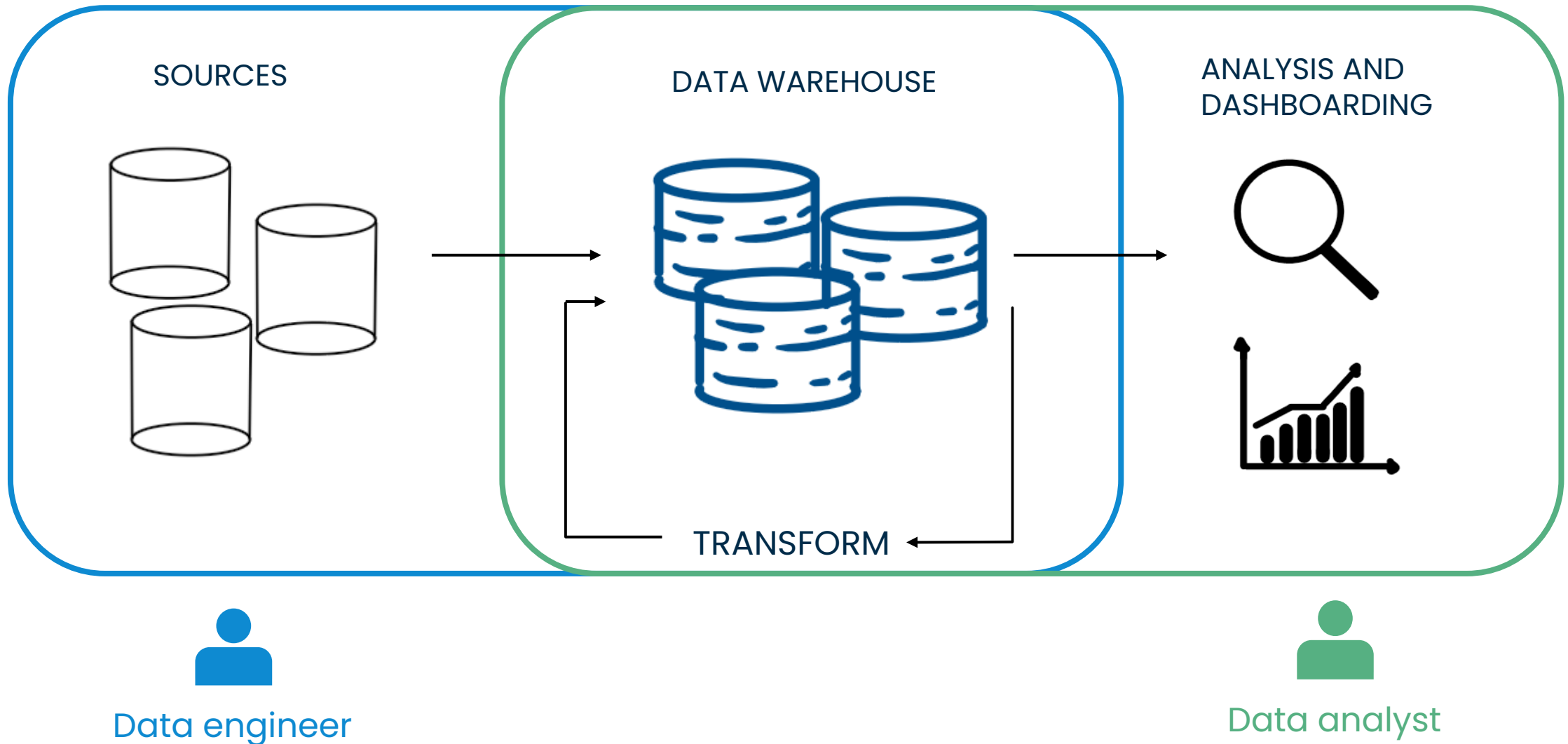
Analysis skills



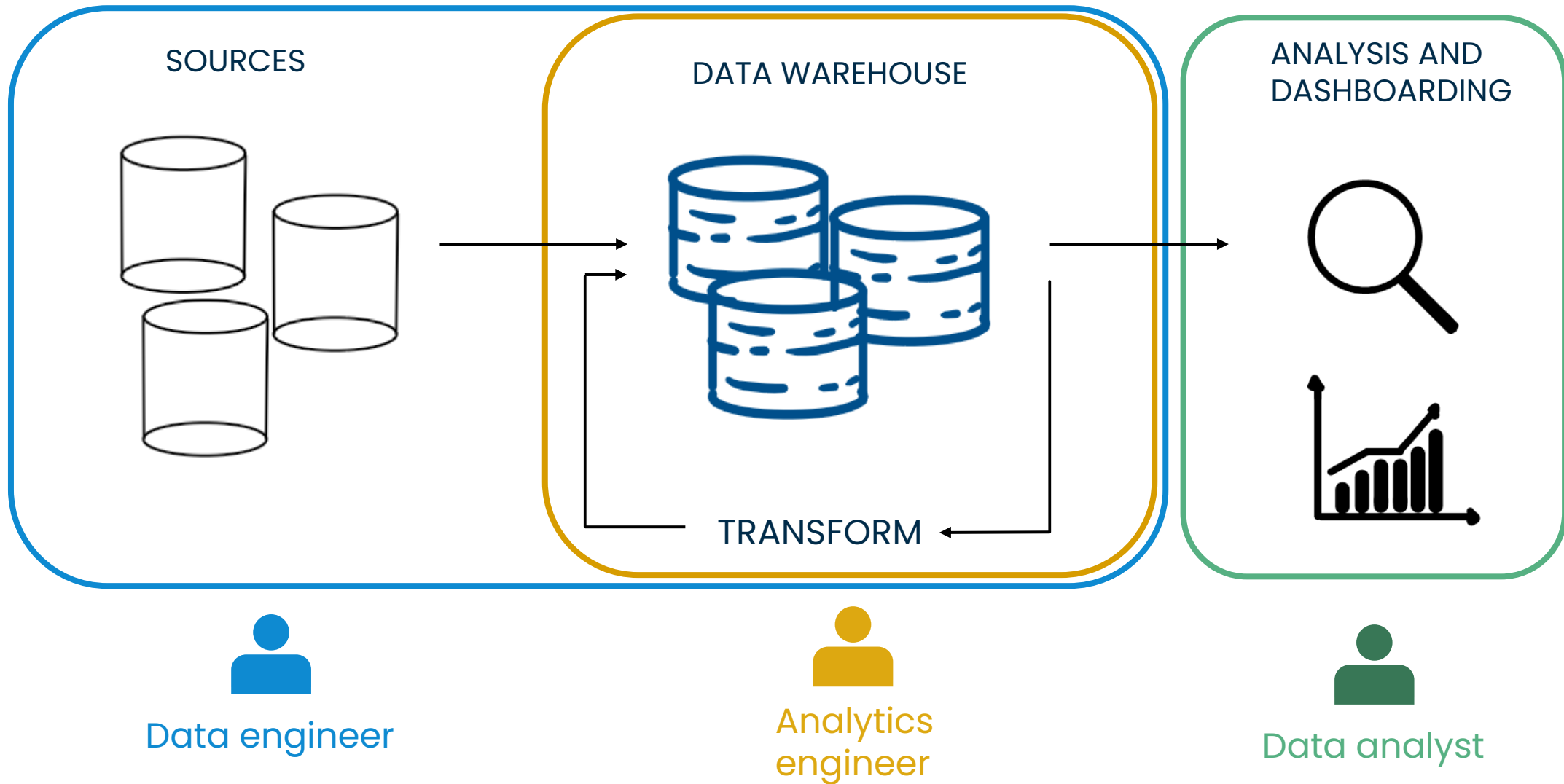
Data analyst

- Work with business users to understand data requirements
- Deep insights work
- Build critical dashboard
- Forecasting

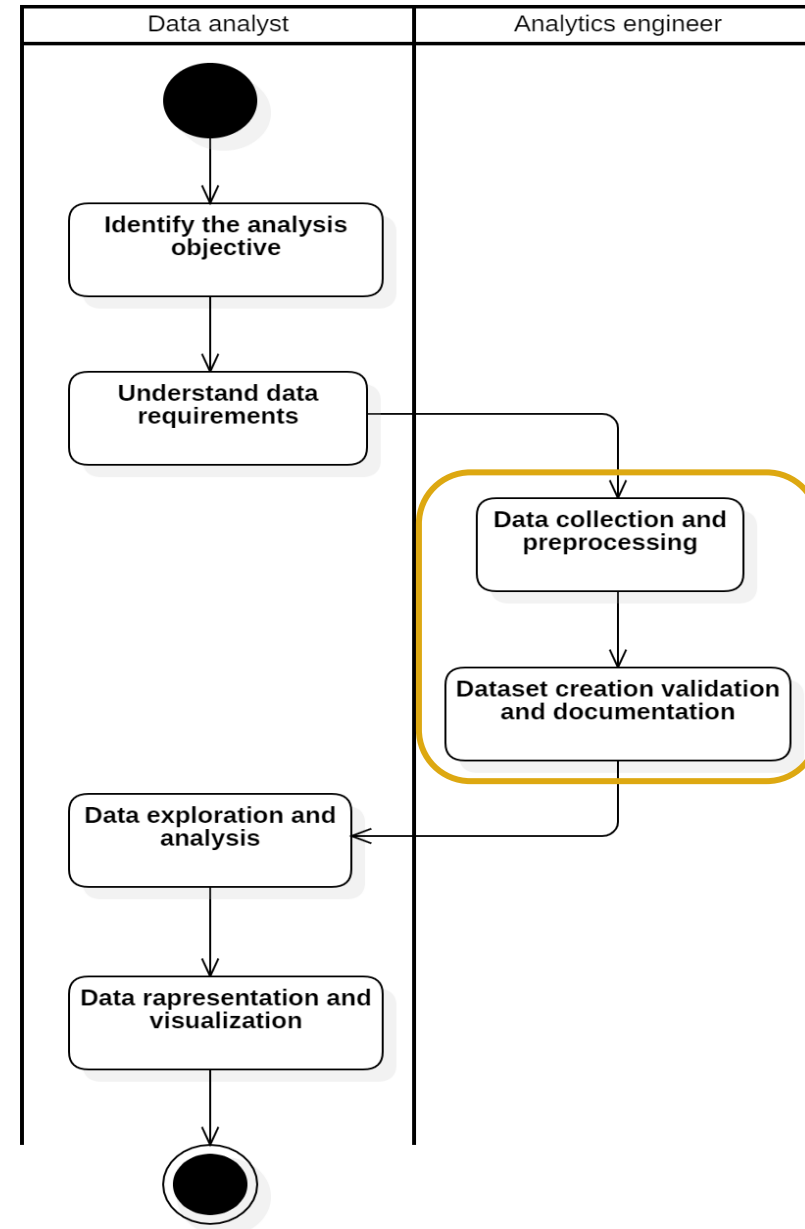
The gap between bussiness and IT



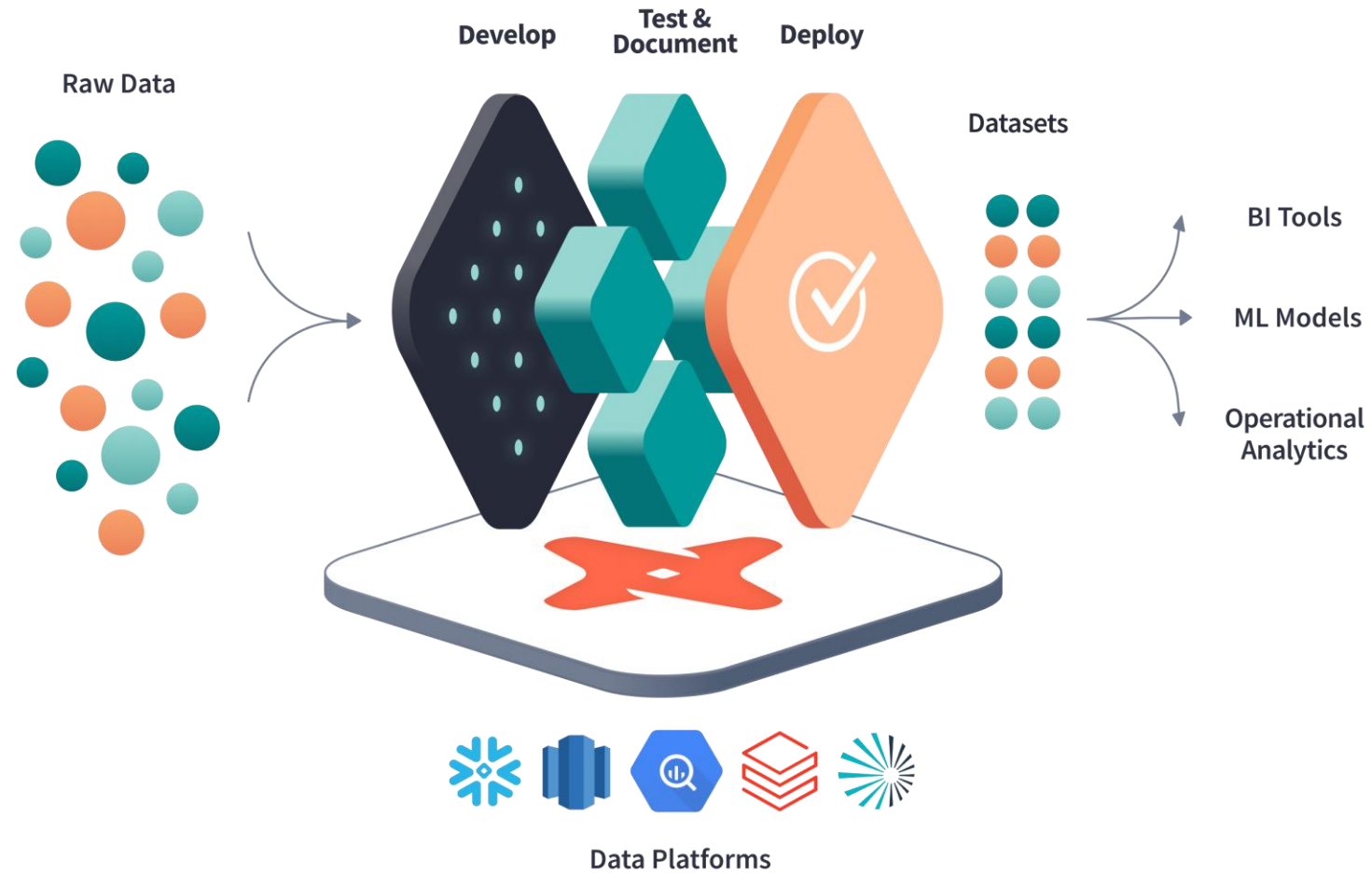
The gap between bussiness and IT



Data model creation process



What is dbt



Proof of Concept

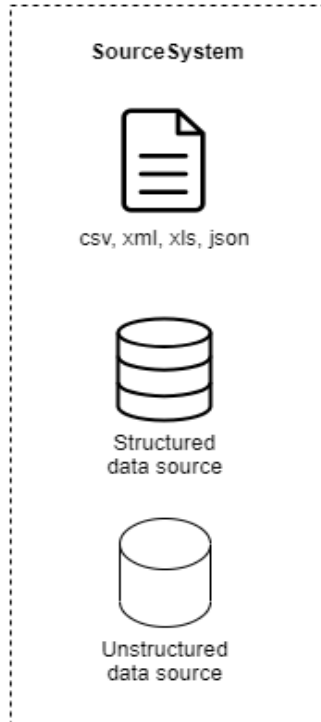
AWS Data pipeline with dbt

Dbt for data transformation

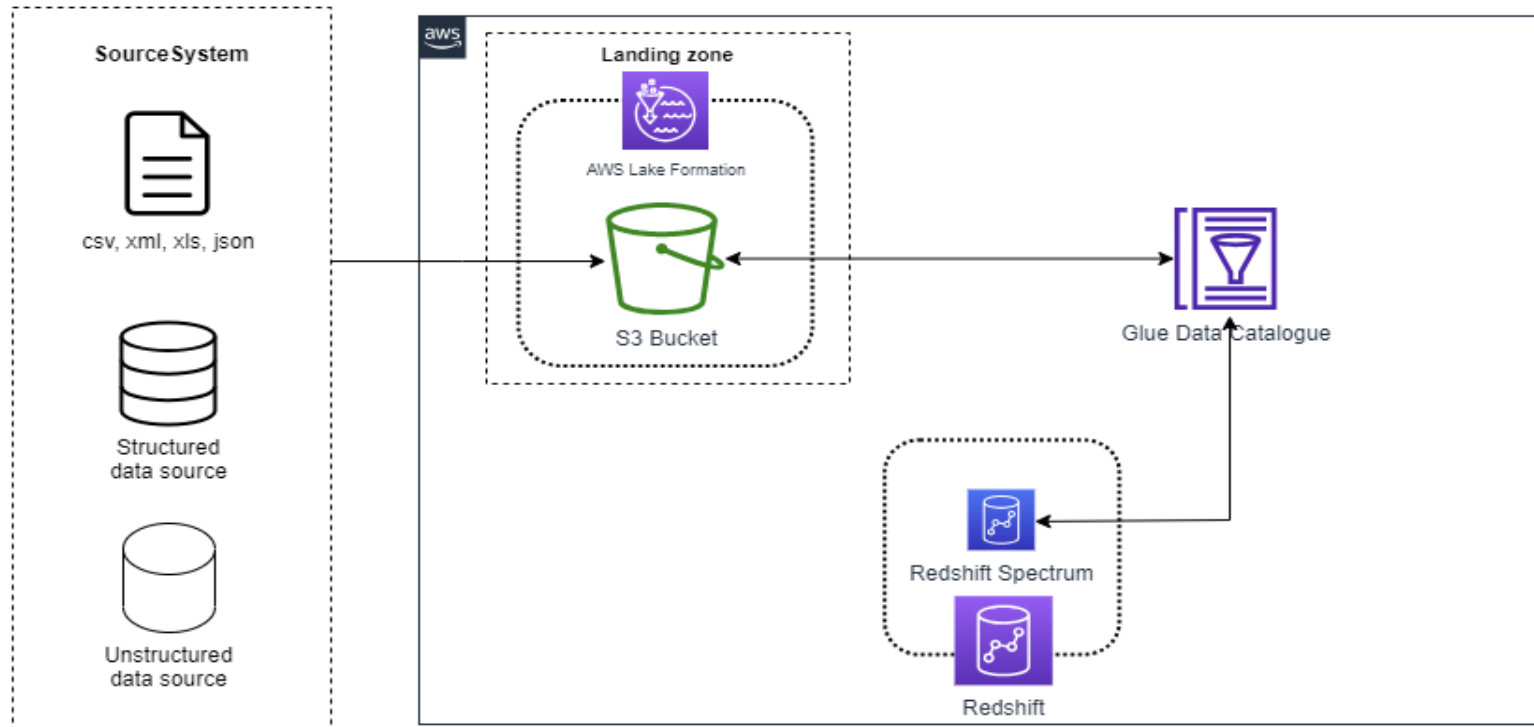
Full Data life cycle



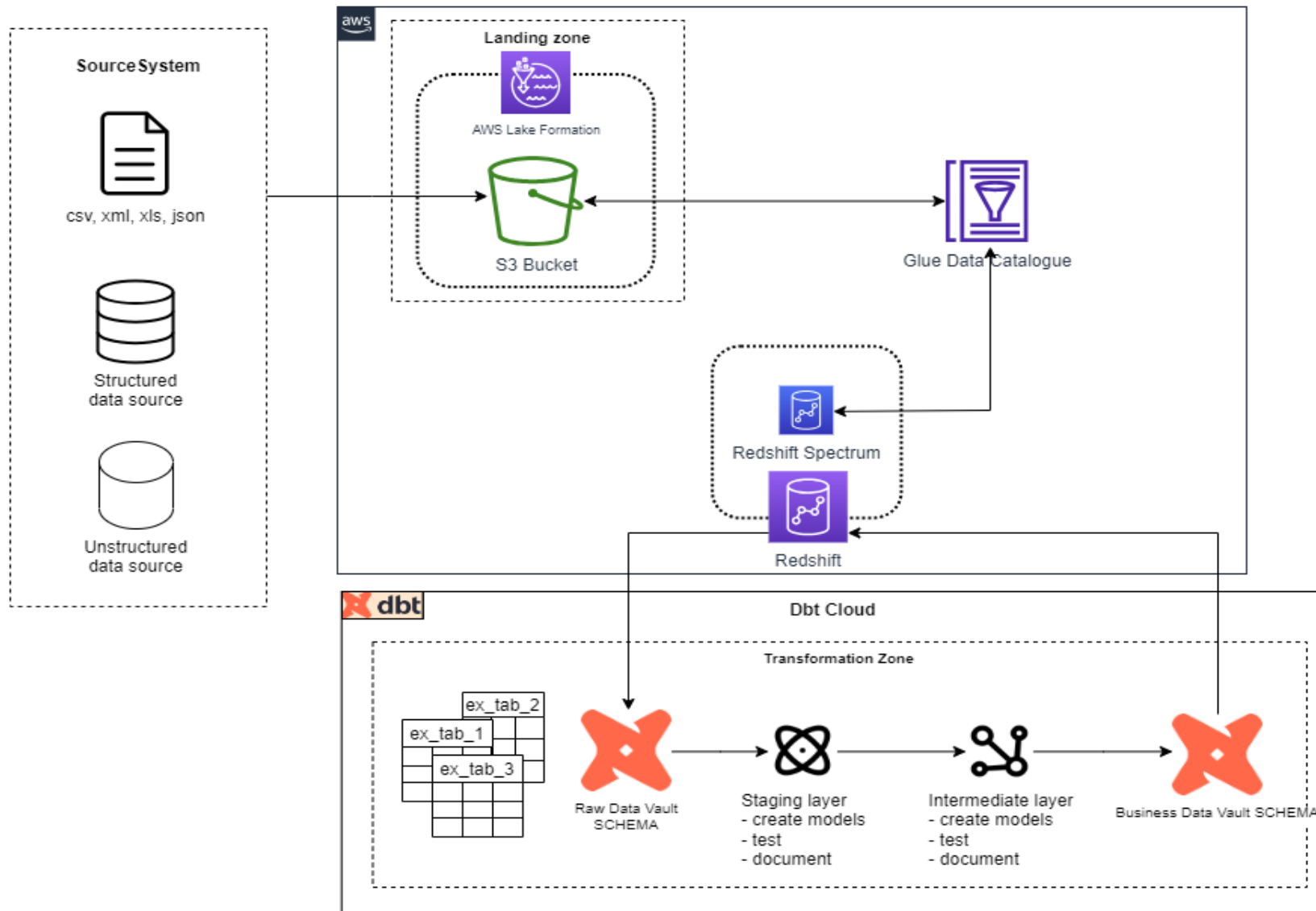
AWS Data pipeline with dbt



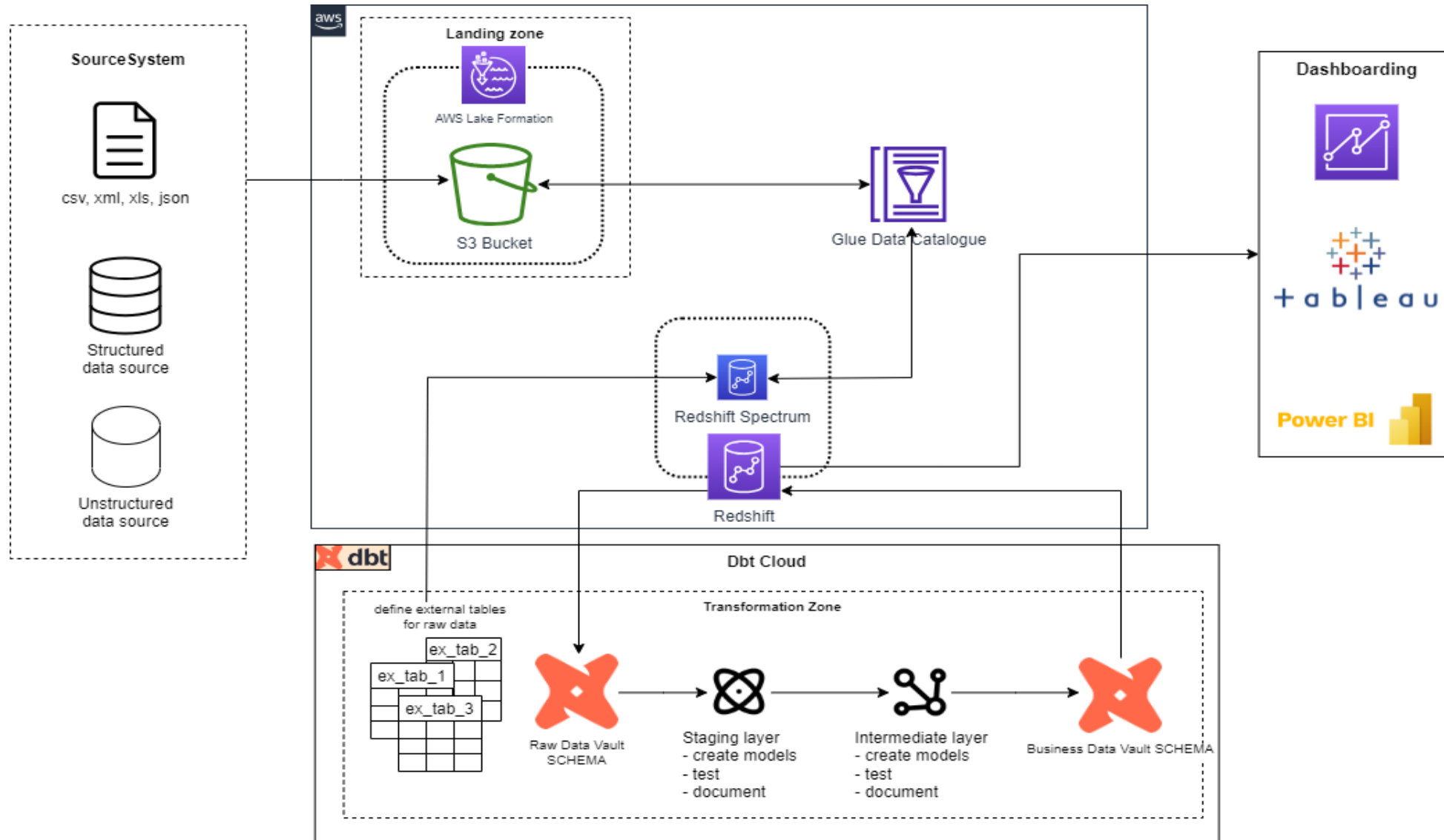
AWS Data pipeline with dbt



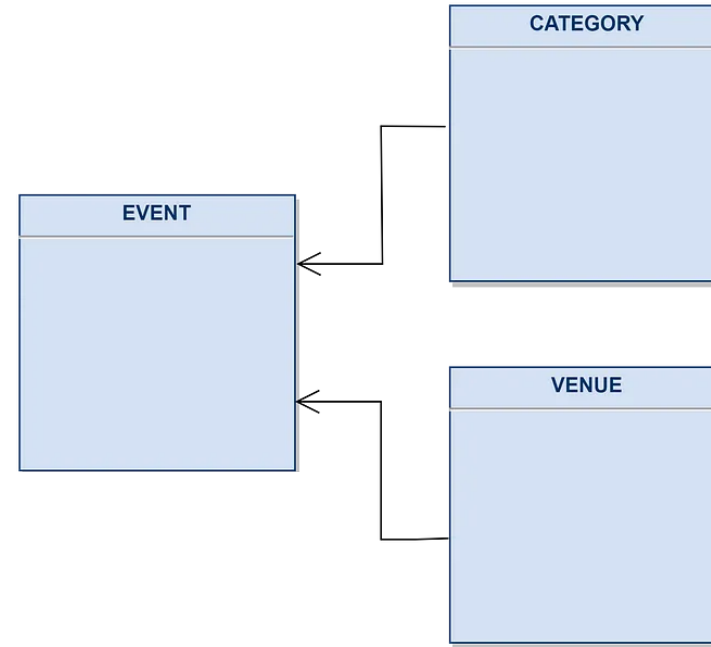
AWS Data pipeline with dbt



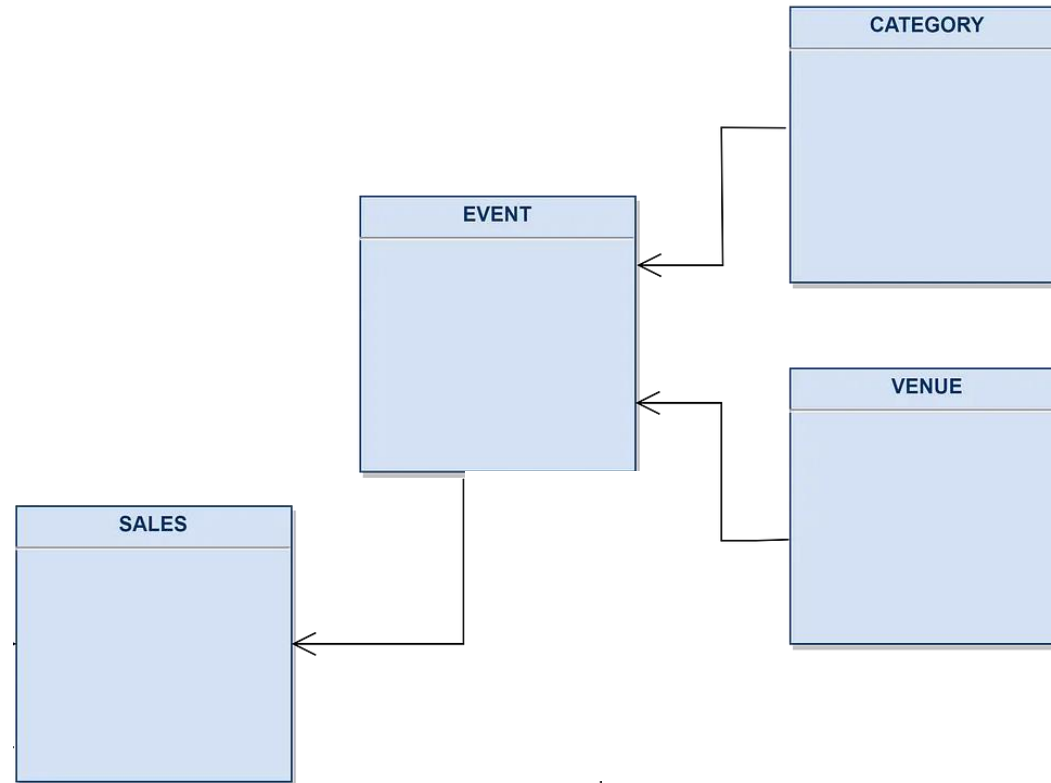
AWS Data pipeline with dbt



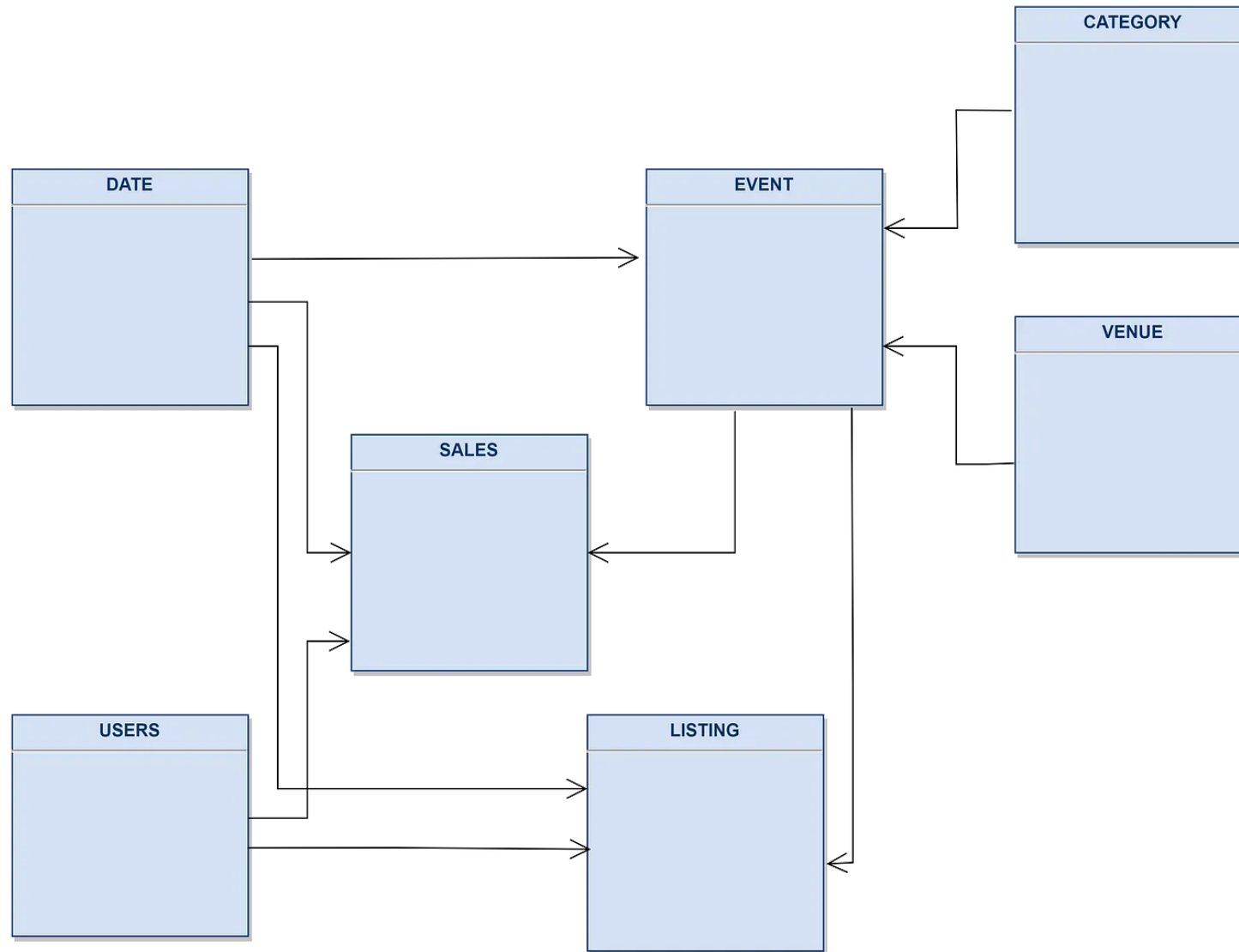
Full data life cycle - raw data



Full data life cycle - raw data



Full data life cycle - raw data



Full data life cycle – Development

Data models are SQL
select statements

No DDL and DML code

SQL empowered with Jinja

```
{% set categories = dbt_utils.get_column_values(ref('stg_tickit__categories')) %}

select
    e.venue_city,
    sum(s.qty_sold) as tickets_sold,
    sum(s.price_paid) as amount_paid,
    {% for category in categories %}
    sum(case when e.cat_name = '{{ category }}' then s.qty_sold end),
    {% if not loop.last %},{% endif %}
    {% endfor %}

from
    sales as s
    join events as e on s.event_id = e.event_id
group by
    e.venue_city
```

Full data life cycle – Development

Data models are SQL
select statements

No DDL and DML code

SQL empowered with Jinja

```
{% set categories = dbt_utils.get_column_values(ref('stg_tickit__categories')) %}

select
    e.venue_city,
    sum(s.qty_sold) as tickets_sold,
    sum(s.price_paid) as amount_paid,
    {% for category in categories %}
    sum(case when e.cat_name = '{{ category }}' then s.qty_sold end),
    {% if not loop.last %},{% endif %}
    {% endfor %}

from
    sales as s
    join events as e on s.event_id = e.event_id
group by
    e.venue_city
```

Full data life cycle – Development

Data models are SQL
select statements

No DDL and DML code

SQL empowered with Jinja

```
{% set categories = dbt_utils.get_column_values(ref('stg_tickit__categories')) %}

select
    e.venue_city,
    sum(s.qty_sold) as tickets_sold,
    sum(s.price_paid) as amount_paid,
    {% for category in categories %}
    sum(case when e.cat_name = '{{ category }}' then s.qty_sold end),
    {% if not loop.last %},{% endif %}
    {% endfor %}

from
    sales as s
    join events as e on s.event_id = e.event_id
group by
    e.venue_city
```

Full data life cycle – Development

Data models are SQL
select statements

No DDL and DML code

SQL empowered with Jinja

```
{% set categories = dbt_utils.get_column_values(ref('stg_tickit__categories')) %}

select
    e.venue_city,
    sum(s.qty_sold) as tickets_sold,
    sum(s.price_paid) as amount_paid,
    {% for category in categories %}
    sum(case when e.cat_name = '{{ category }}' then s.qty_sold end),
    {% if not loop.last %},{% endif %}
    {% endfor %}

from
    sales as s
    join events as e on s.event_id = e.event_id
group by
    e.venue_city
```


Full data life cycle – Development

Data models are SQL
select statements

No DDL and DML code

SQL empowered with Jinja

```
{% set categories = dbt_utils.get_column_values(ref('stg_tickit__categories')) %}

select
    e.venue_city,
    sum(s.qty_sold) as tickets_sold,
    sum(s.price_paid) as amount_paid,
    {% for category in categories %}
    sum(case when e.cat_name = '{{ category }}' then s.qty_sold end),
    {% if not loop.last %},{% endif %}
    {% endfor %}

from
    sales as s
    join events as e on s.event_id = e.event_id
group by
    e.venue_city
```

Full data life cycle – Testing and validation

Built-in test

Singular test

Generic test

```
- name: fct_sales
  description: All sales with details
  columns:
    - name: sale_id
      description: primary key
      tests:
        - unique
        - not_null

- name: kpi_sales_per_city_category
  description: Indicators of tickets' sales by city per each category
  columns:
    - name: tickets_sold
      tests:
        - test_generic_assert_positive_value
    - name: amount_paid
      tests:
        - test_generic_assert_positive_value
```

Full data life cycle – Testing and validation

Built-in test

Singular test

Generic test

```
- name: fct_sales
  description: All sales with details
  columns:
    - name: sale_id
      description: primary key
      tests:
        - unique
        - not_null

- name: kpi_sales_per_city_category
  description: Indicators of tickets' sales by city per each category
  columns:
    - name: tickets_sold
      tests:
        - test_generic_assert_positive_value
    - name: amount_paid
      tests:
        - test_generic_assert_positive_value
```

Full data life cycle – Testing and validation

Built-in test

Singular test

Generic test

```
- name: fct_sales
  description: All sales with details
  columns:
    - name: sale_id
      description: primary key
      tests:
        - unique
        - not_null

- name: kpi_sales_per_city_category
  description: Indicators of tickets' sales by city per each category
  columns:
    - name: tickets_sold
      tests:
        - test_generic_assert_positive_value
    - name: amount_paid
      tests:
        - test_generic_assert_positive_value
```

Full data life cycle - Documentation

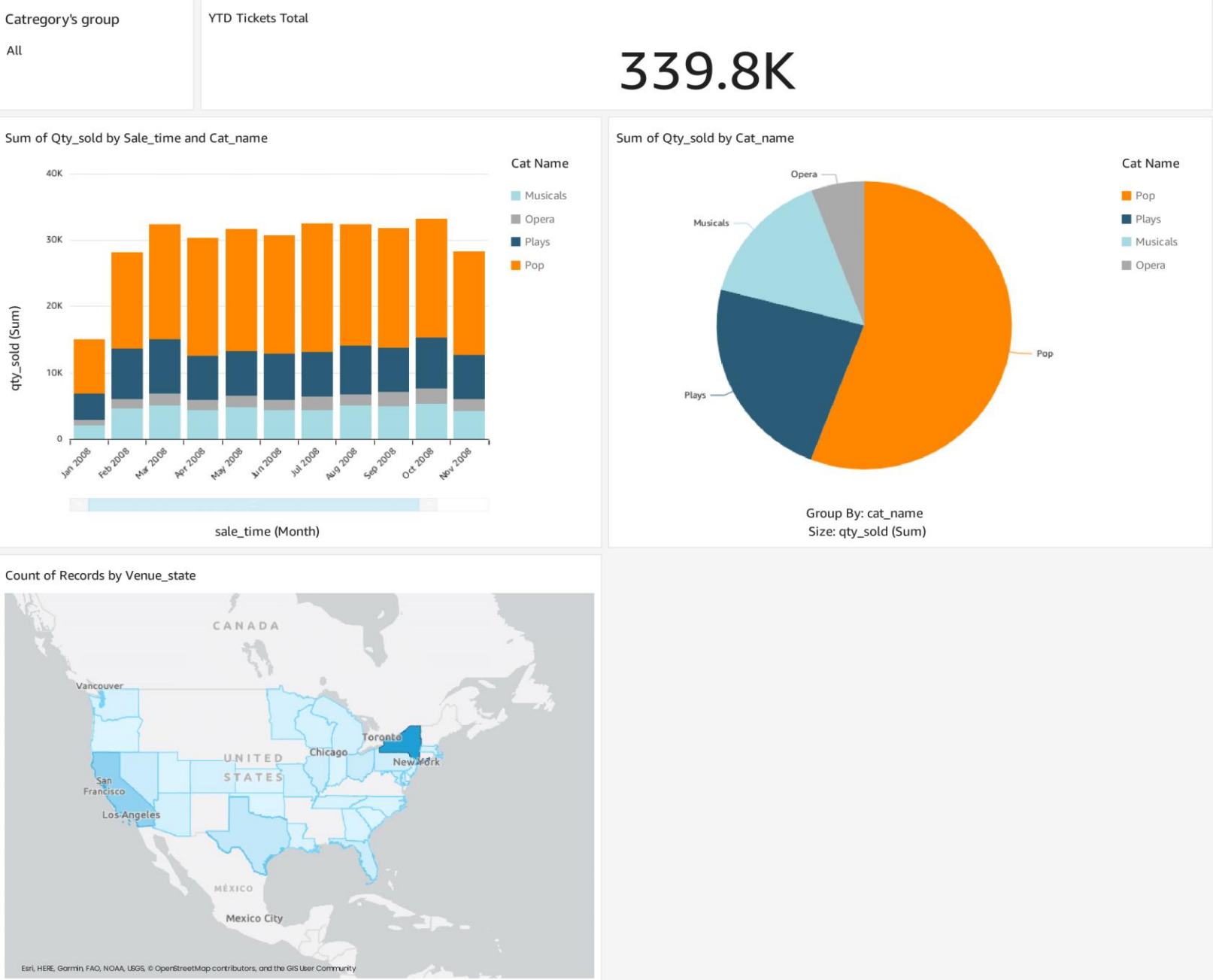
«dbt docs generate»

Dbt extract model's dependency and fields description from all .yml file and .sql models

automatically generate documentation and Lineage Graph



Data analysis and visualization



Achievements

Process diagram analysis between traditional and proposed paradigm

Design and implementation of a cloud data pipeline on AWS

Build a mature dbt project for data transformation

Simulate with a fictional database the data life cycle from source ingestion to analysis and dashboarding



Thank you for your
attention!



**Politecnico
di Torino**