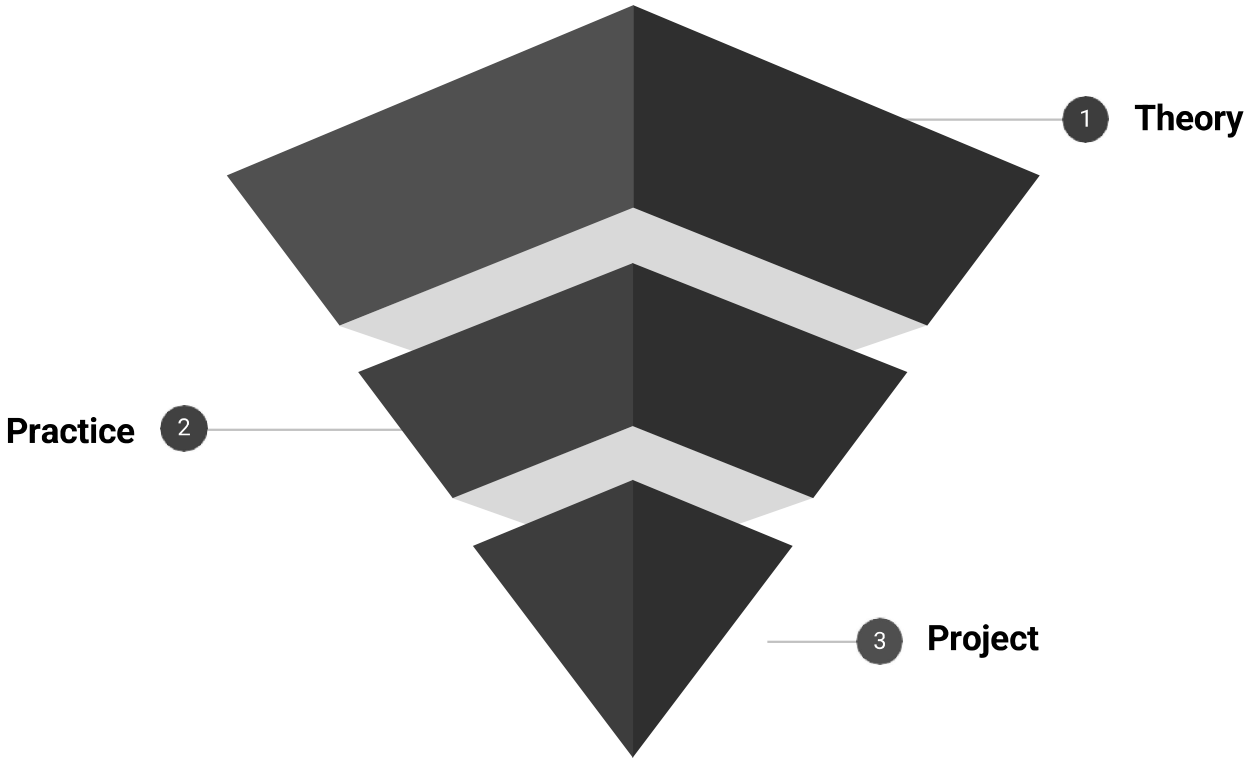
Course Introduction

OBJECTIVES

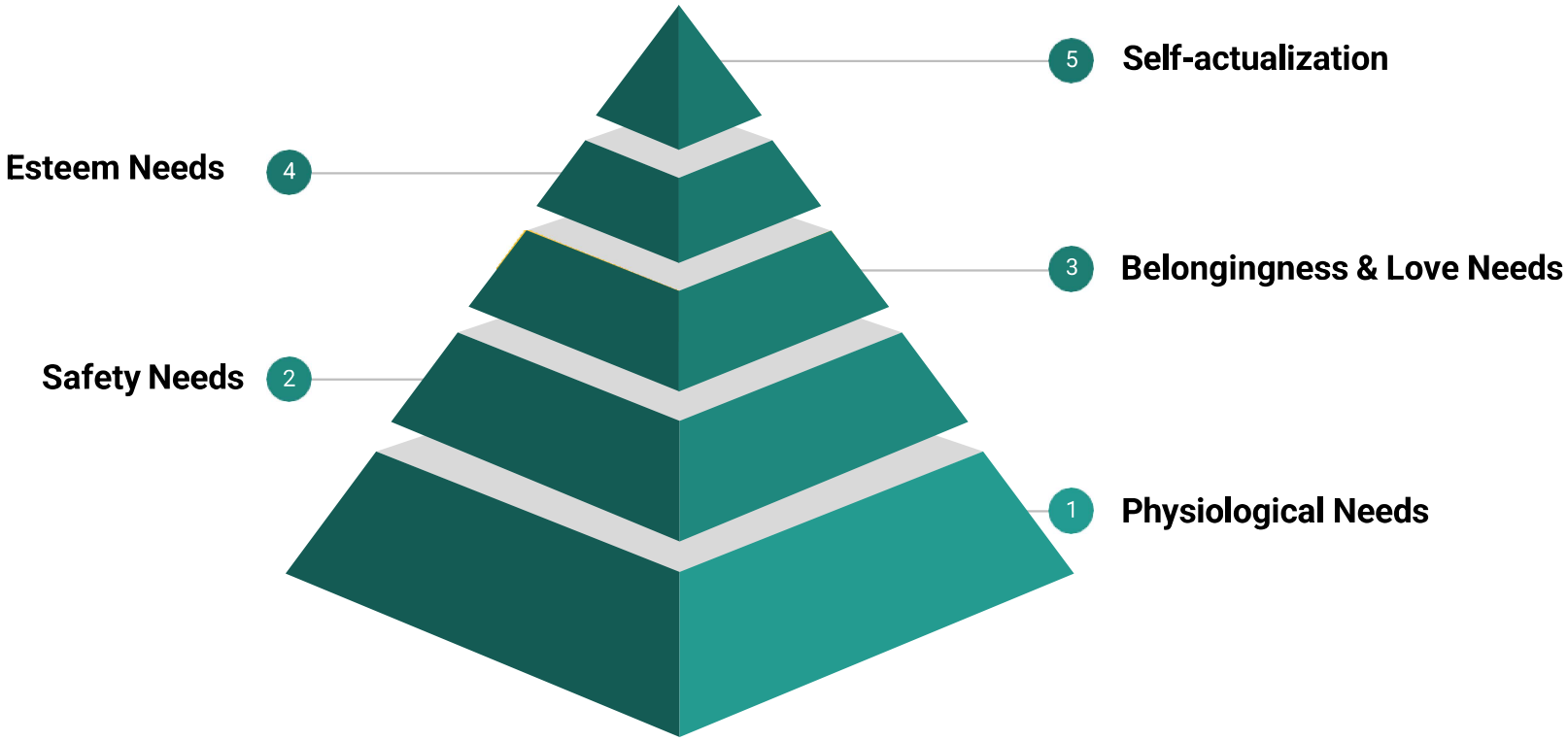
- Data-maturity model
- dbt and data architectures
- Data warehouses, data lakes, and lakehouses
- ETL and ELT procedures
- dbt fundamentals
- Analytics Engineering

TOP-DOWN

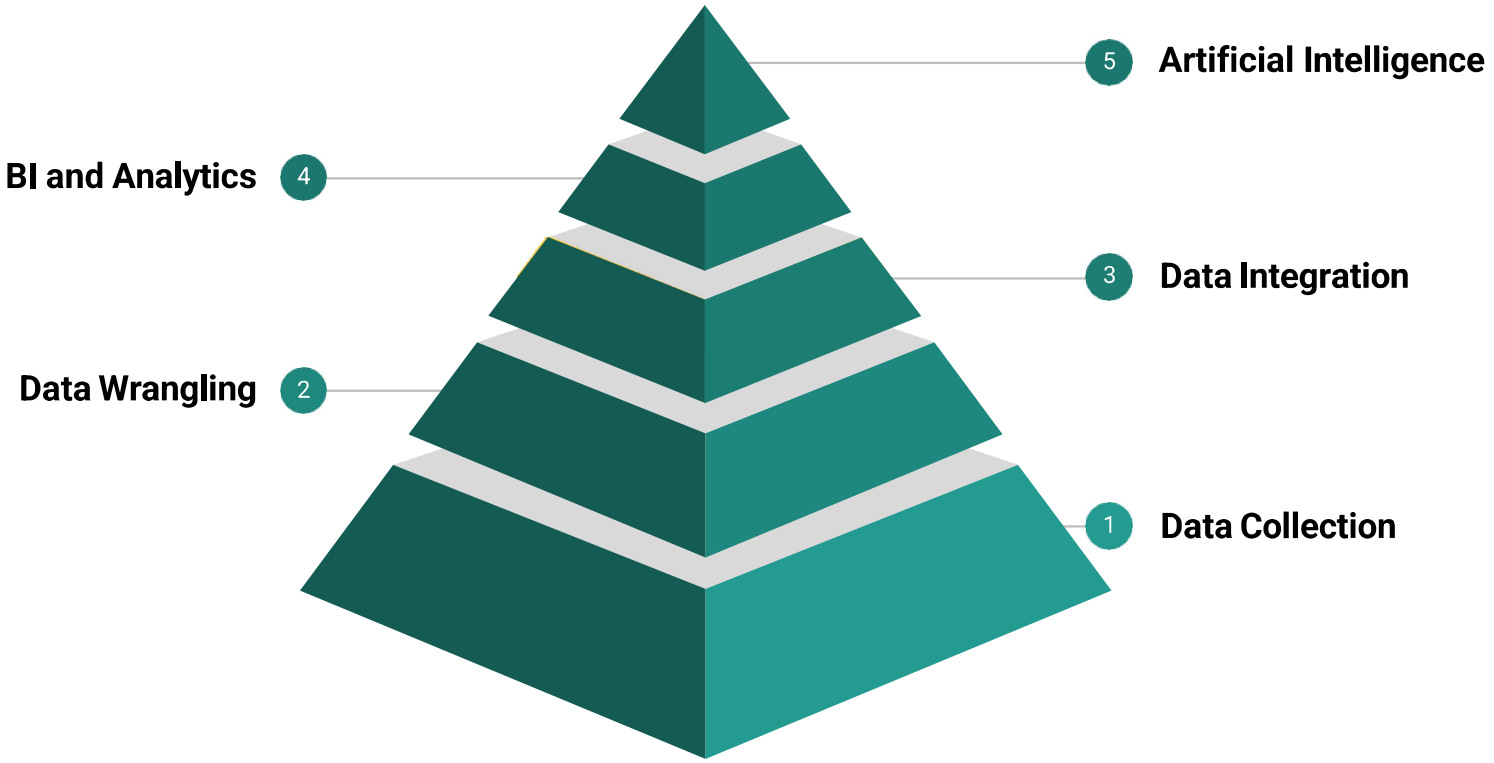


Data Maturity Model

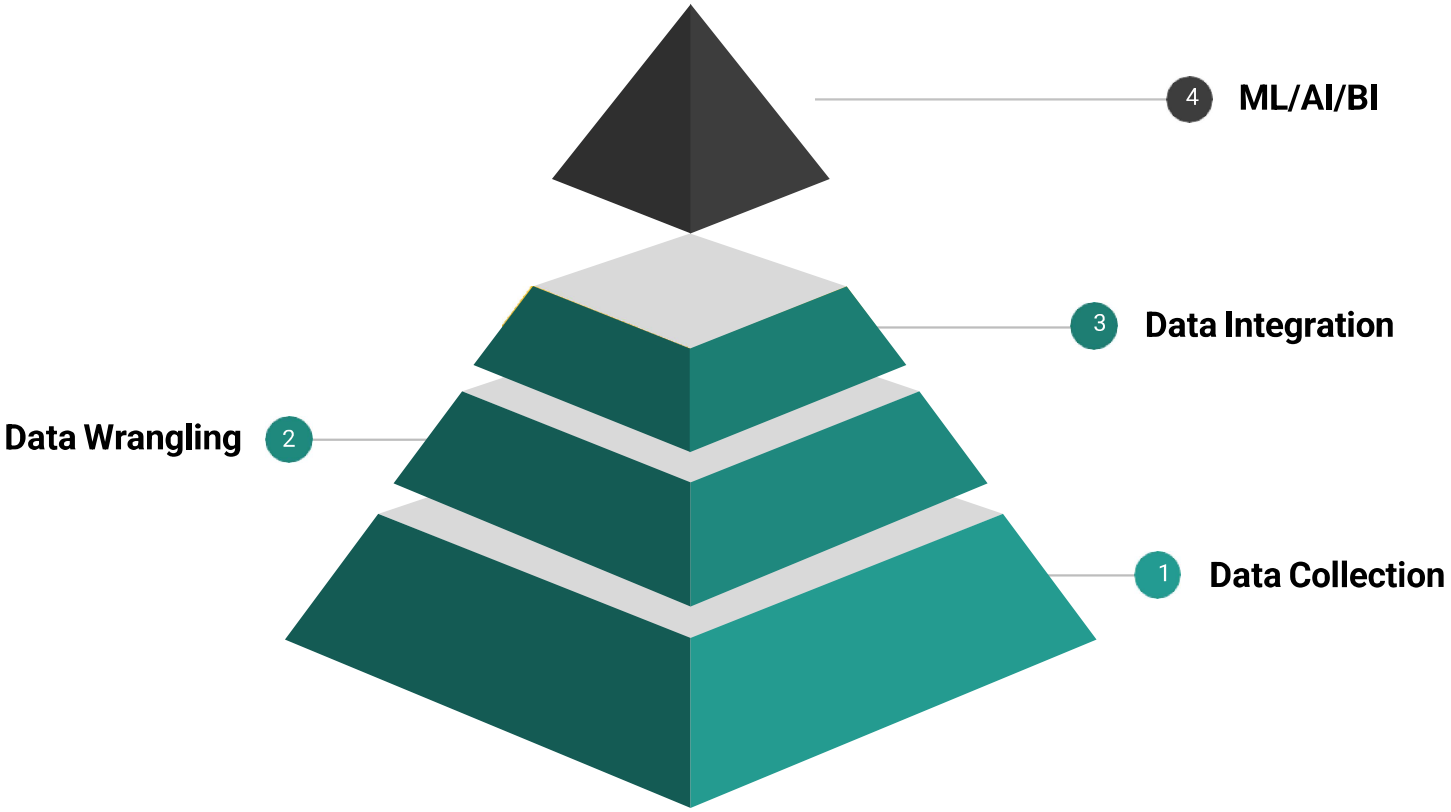
Maslow's Hierarchy of Needs



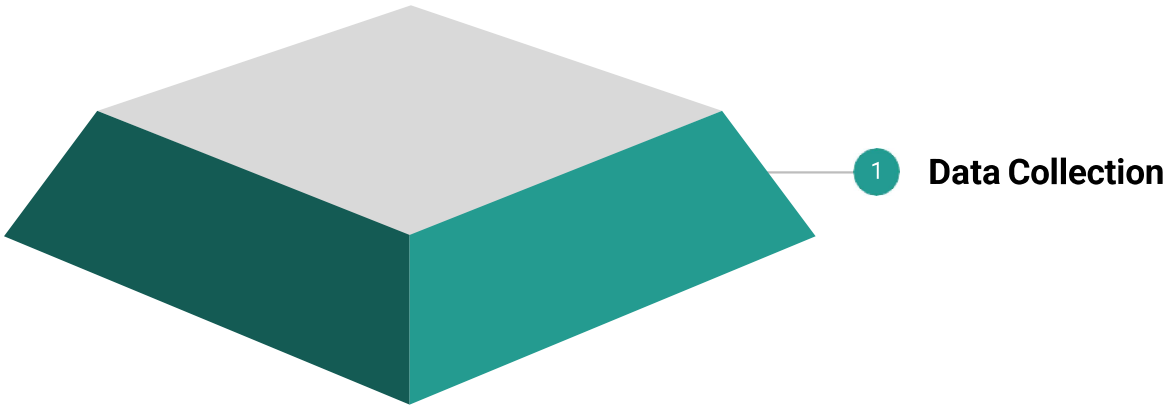
Data-Maturity Model



Typical Data Architecture



Data Collection

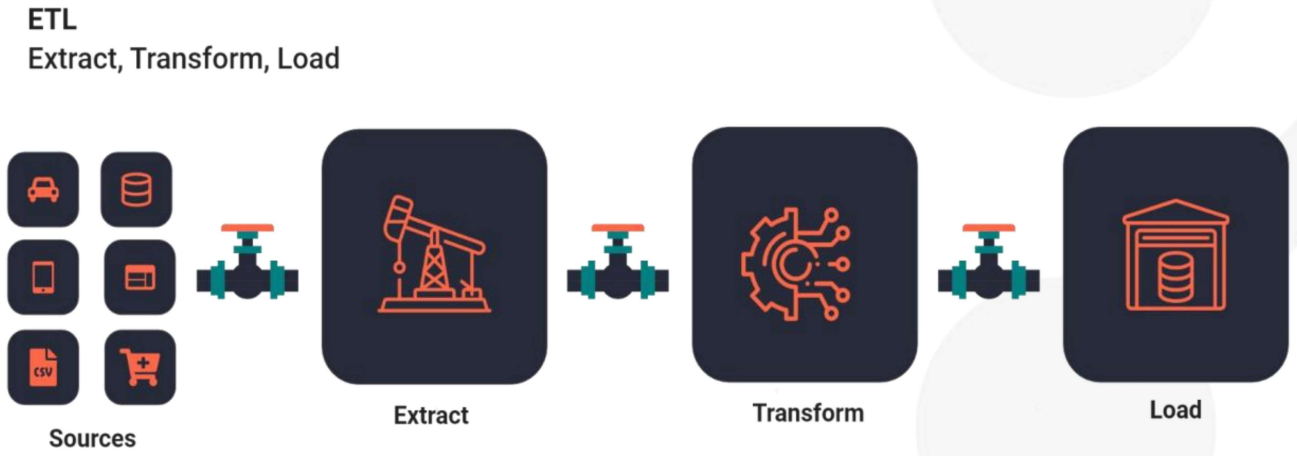


Data Wrangling

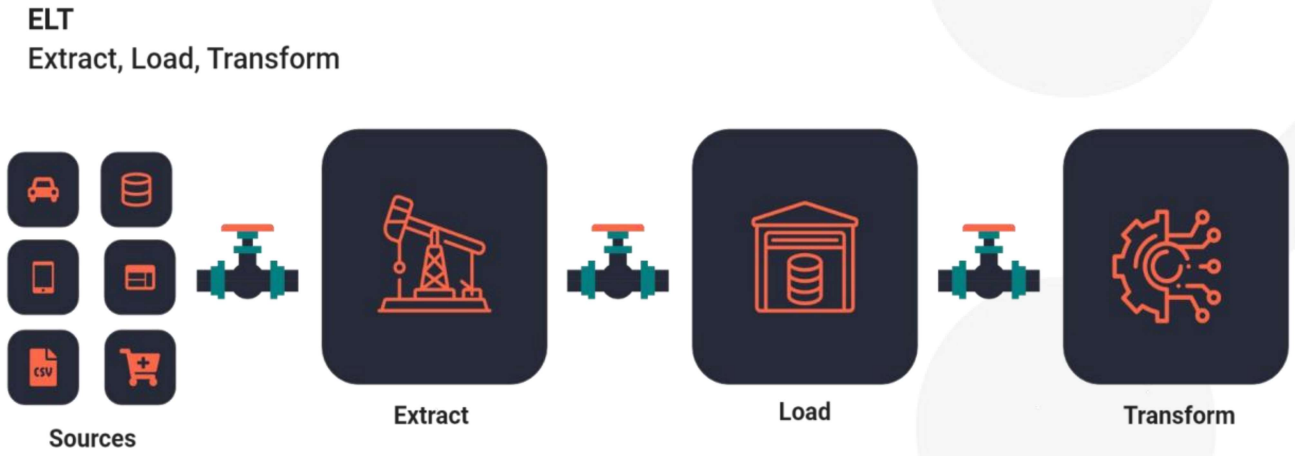
Data Integration

ETL - ELT#

ETL


ETL
Extract, Transform, Load
csv
Sources
Extract
Transform
Load

ELT


ELT
Extract, Load, Transform
csv
Sources
Extract
Load
Transform

Data Warehouse

Data Warehouse

On-Premise

ORACLE IBM teradata.

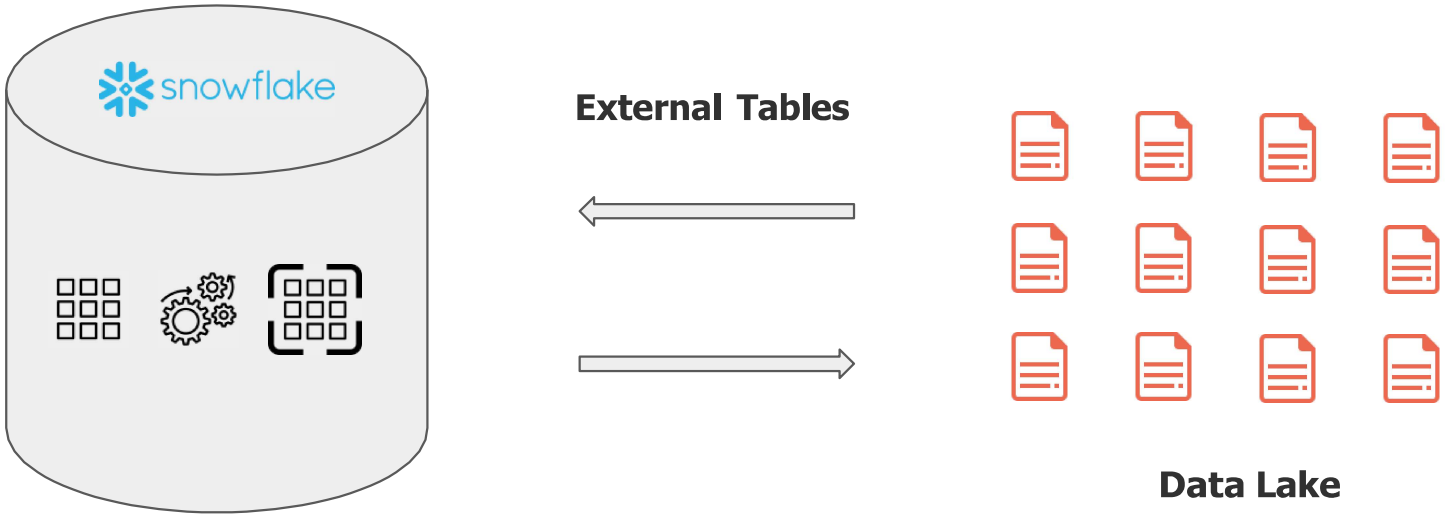
Cloud

amazon REDSHIFT Google BigQuery snowflake

PNG Credits: https://databricks.com/wp-content/uploads/2020/01/data-lakehouse-new-1024x538.png

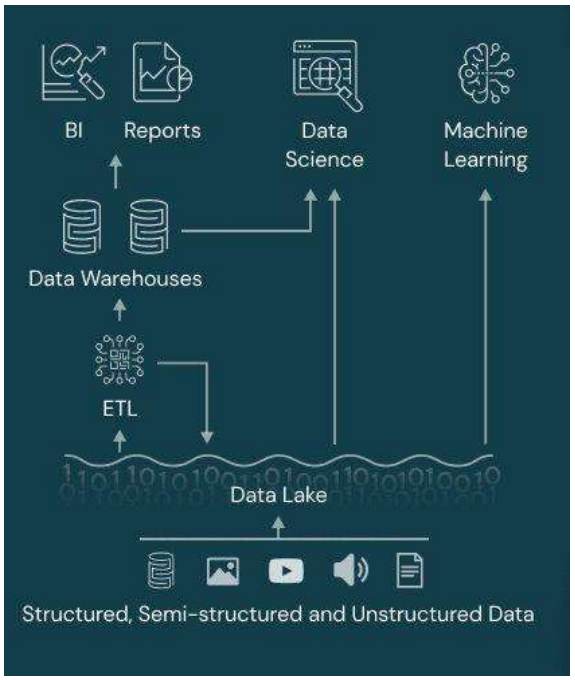
External Tables

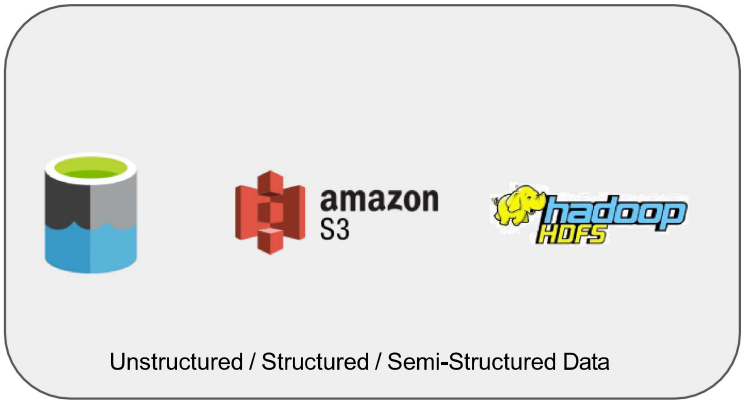
External Tables



Data Lake

Data Lake

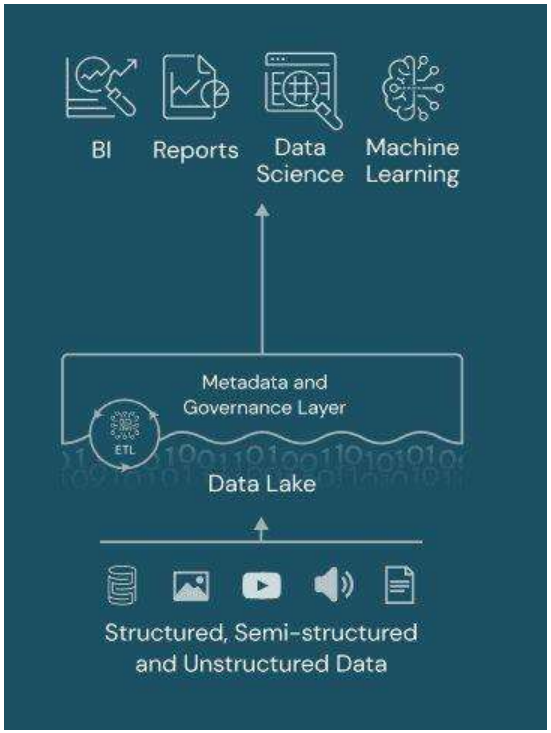

BI
Reports
Data Science
Machine Learning
Data Warehouses
ETL
Data Lake
Structured, Semi-structured and Unstructured Data


amazon S3
hadoop HDFS
Unstructured / Structured / Semi-Structured Data

Data Lakehouse

PNG Credits: https://databricks.com/wp-content/uploads/2020/01/data-lakehouse-new-1024x538.png

Data Lakehouse



Slowly Changing Dimensions

PNG Credits: https://databricks.com/wp-content/uploads/2020/01/data-lakehouse-new-1024x538.png

SCD Type 0

Not updating the DWH table when a Dimension changes

Source	DWH
Host ID | Name | Fax Number #1 | John Doe | 1-408-999 8888	Host ID | Name | Fax Number #1 | John Doe | 555-123-4567

SCD Type 1

Updating the DWH table when a Dimension changes, overwriting the original data

	Source	DWH
No Air-conditioning	Host ID: #1, Air-conditioning: No	Host ID: #1, Air-conditioning: No
Installed Air-conditioning	Host ID: #1, Air-conditioning: Yes	Host ID: #1, Air-conditioning: No
DWH updated	Host ID: #1, Air-conditioning: Yes	Host ID: #1, Air-conditioning: Yes

SCD Type 2

Keeping full history - Adding additional (historic data) rows for each dimension change

Current rental price ($300)

Source		DWH				
Apartment ID	Price	Host Key	Apartment ID	Price	Start_Date	End_Date
#1	$300	1000	#1	$300	2020-01-01T00:00:00	NULL

Change in the rental price ($450)

Source		DWH				
Apartment ID	Price	Host Key	Apartment ID	Price	Start_Date	End_Date
#1	$450	1000	#1	$300	2020-01-01T00:00:00	2021-01-01T00:00:00

DWH updated

Source		DWH				
Apartment ID	Price	Host Key	Apartment ID	Price	Start_Date	End_Date
#1	$450	1000	#1	$300	2020-01-01T00:00:00	2021-01-01T00:00:00
		1001	#1	$450	2021-01-01T00:00:00	NULL

SCD Type 2

Keeping full history - Adding additional (historic data) rows for each dimension change

SCD Type 3

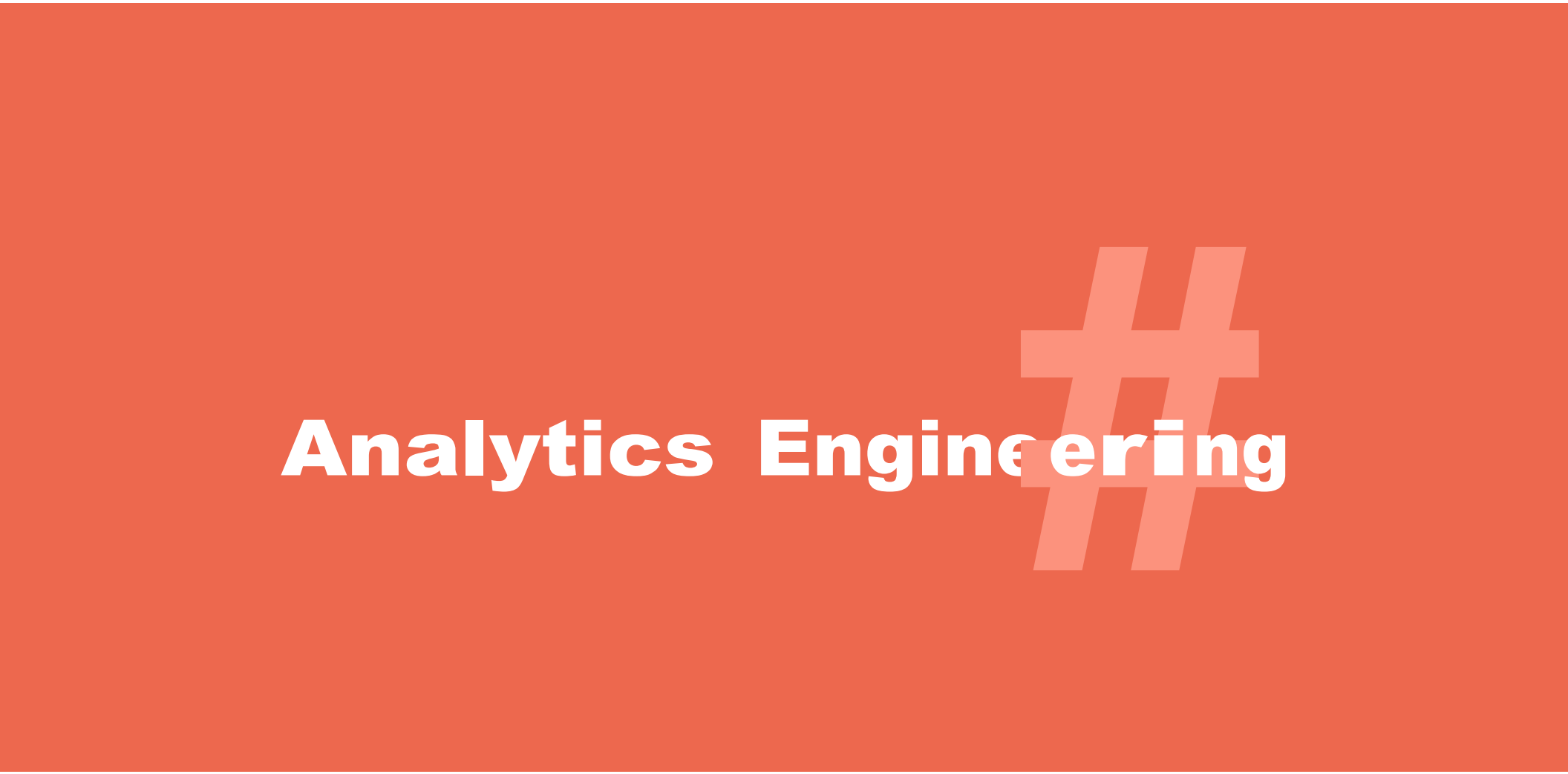
Keeping limited data history - adding separate columns for original and current value

	Source	DWH
Listed as Private	Apartment ID: #1, Type: Private	Apartment ID: #1, Previous Type: Private, Current Type: Private
Host changed Private to Entire	Apartment ID: #1, Type: Entire	Apartment ID: #1, Previous Type: Private, Current Type: Entire
Host changed Entire to Shared	Apartment ID: #1, Type: Shared	Apartment ID: #1, Previous Type: Entire, Current Type: Shared

dbt Overview

dbt Overview

PNG Credits: https://data.solita.fi/wp-content/uploads/2021/06/dbt-transform.png


Analytics Engineering

Common Table Expression (CTE)

CTE

Syntax

```SQL
WITH <name_of_the_result_set> ([column_names])
AS
(
   <cte_query>
)
<reference_the_CTE>
```

CTE

Example

```sql
-- STEP 1
WITH raw_listings AS (

-- STEP 2
        SELECT * FROM [source].[listings]
)

-- STEP 3
SELECT
   id AS listing_id,
   listing_url,
   name AS listing_name,
   room_type,
   minimum_nights,
   host_id,
   price AS price_str,
   created_at,
   updated_at

FROM raw_listings
```

PROJECT OVERVIEW

Analytics Engineering with Airbnb

ANALYTICS ENGINEERING WITH AIRBNB

- Simulating the life of an Analytics Engineer in Airbnb
- Loading, Cleansing, Exposing data
- Writing test, automations and documentation
- Data source: Inside Airbnb: Berlin

TECH STACK

REQUIREMENTS

- Modeling changes are easy to follow and revert
- Explicit dependencies between models
- Explore dependencies between models
- Data quality tests
- Error reporting
- Incremental load of fact tables
- Track history of dimension tables
- Easy-to-access documentation

STEPS - SETUP

- Snowflake registration
- Dataset import
- dbt installation
- dbt setup, snowflake connection
- git synchronization
- Visual Studio code installation

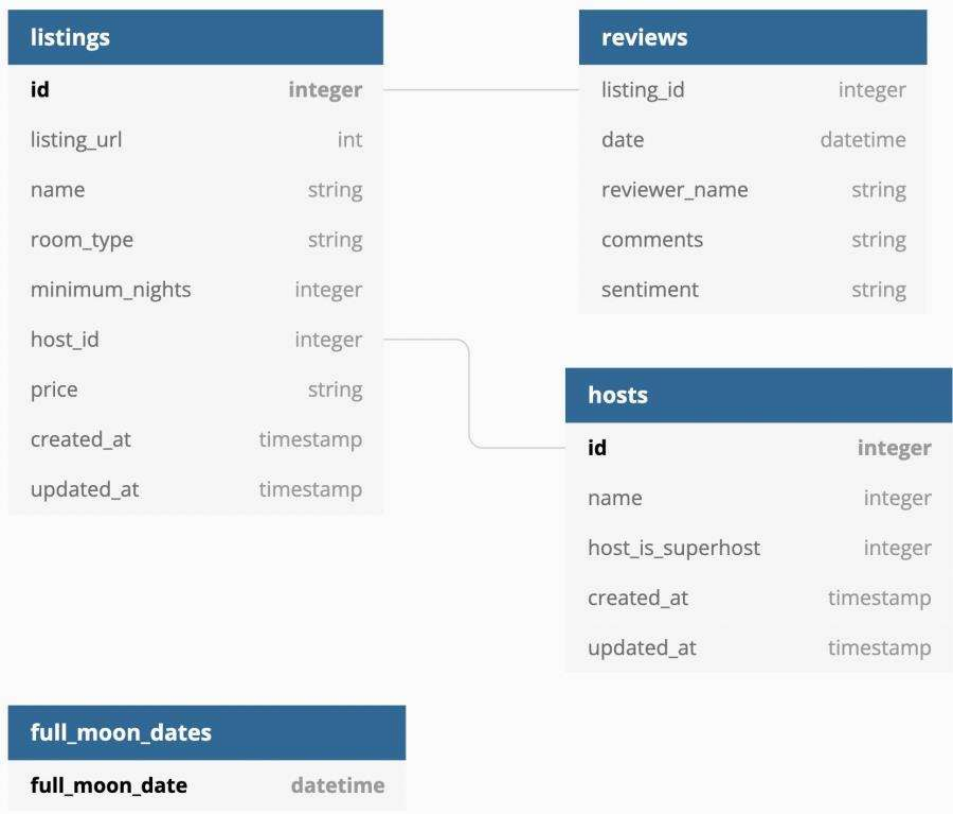
SNOWFLAKE

Registering a Trial account

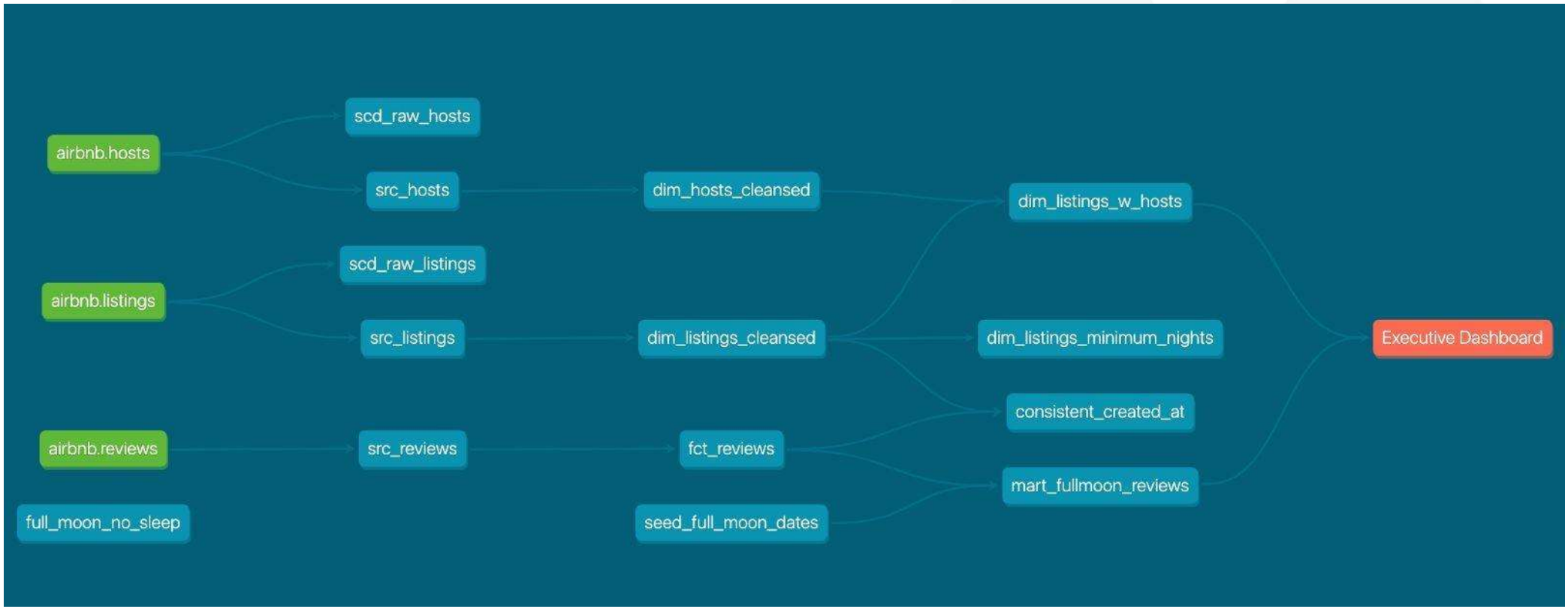
DATA FLOW

Overview

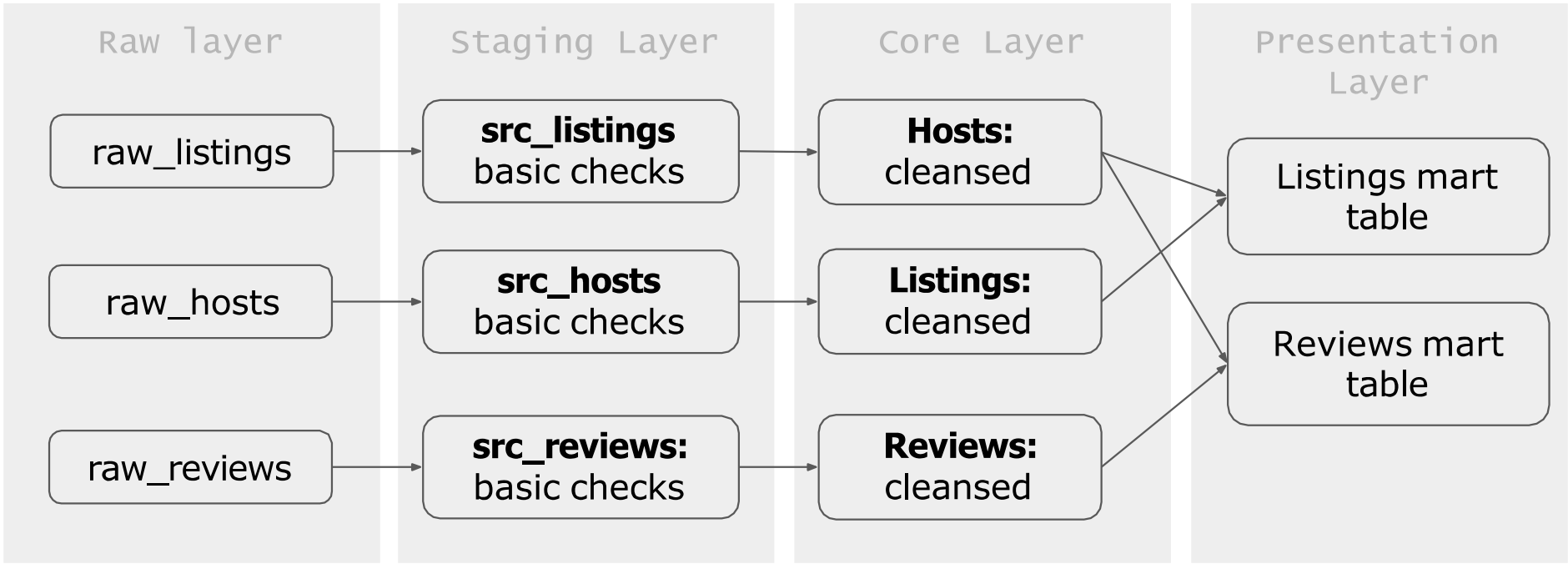
INPUT DATA MODEL



DATA FLOW OVERVIEW



DATA FLOW OVERVIEW



VIRTUALENV SETUP

- Install Python 3.11 and the Python virtualenv package
- Create a virtualenv
- Activate virtualenv

DBT SETUP

Windows

INSTALLING DBT

- Install Python3
- Create a virtualenv
- Activate virtualenv
- Install dbt and the dbt-snowflake connector

VSCODE SETUP

Installing the dbt power user extension

DBT SETUP

dbt init and connecting to Snowflake

MODELS

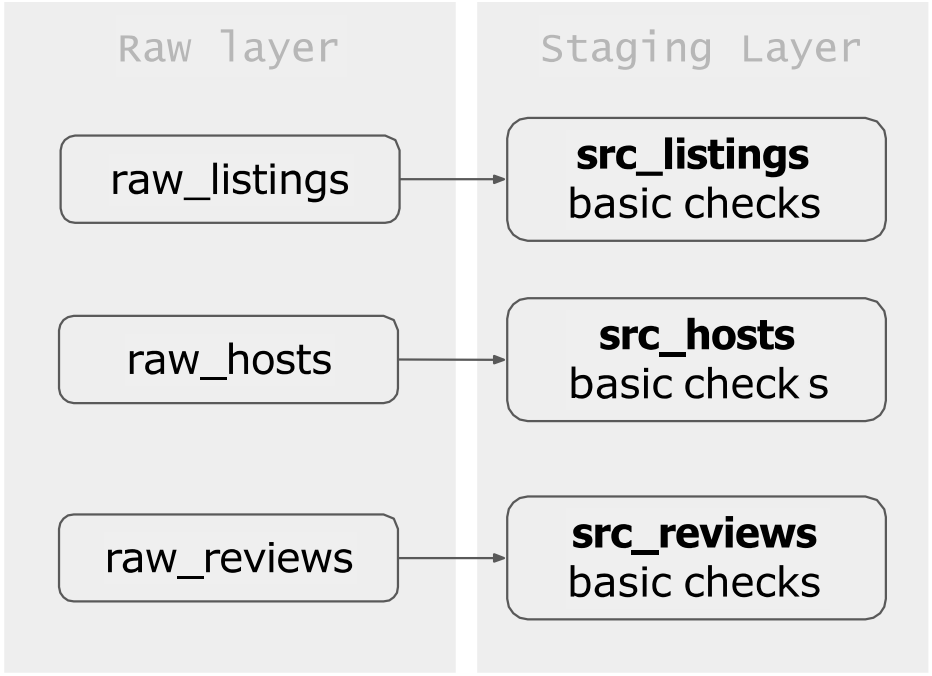
LEARNING OBJECTIVES

- Understand the data flow of our project
- Understand the concept of Models in dbt
- Create three basic models:
  - src_listings
  - src_reviews: guided exercises
  - src_hosts: individual lab

MODELS OVERVIEW

- Models are the basic building block of your business logic
- Materialized as tables, views, etc…
- They live in SQL files in the `models` folder
- Models can reference each other and use templates and macros

DATA FLOW PROGRESS



GUIDED EXERCISE

src_reviews.sql

Create a new model in the `models/src/` folder called `src_reviews.sql`.

- Use a CTE to reference the AIRBNB.RAW.RAW_REVIEWS table
- SELECT every column and every record, and rename the following columns:
  - *date* to *review_date*
  - *comments* to *review_text*
  - *sentiment* to *review_sentiment*
- Execute `dbt run` and verify that your model has been created

MATERIALIZATIONS

LEARNING OBJECTIVES

- Understand how models can be connected
- Understand the four built-in materializations
- Understand how materializations can be configured on the file and project level
- Use *dbt run* with extra parameters

MATERIALIZATIONS

MATERIALISATIONS OVERVIEW

View

Use it
- You want a lightweight representation
- You don't reuse data too often

Don't use it
- You read from the same model several times

Table

Use it
- You read from this model repeatedly

Don't use it
- Building single-use models
- Your model is populated incrementally

Incremental (table appends)

Use it
- Fact tables
- Appends to tables

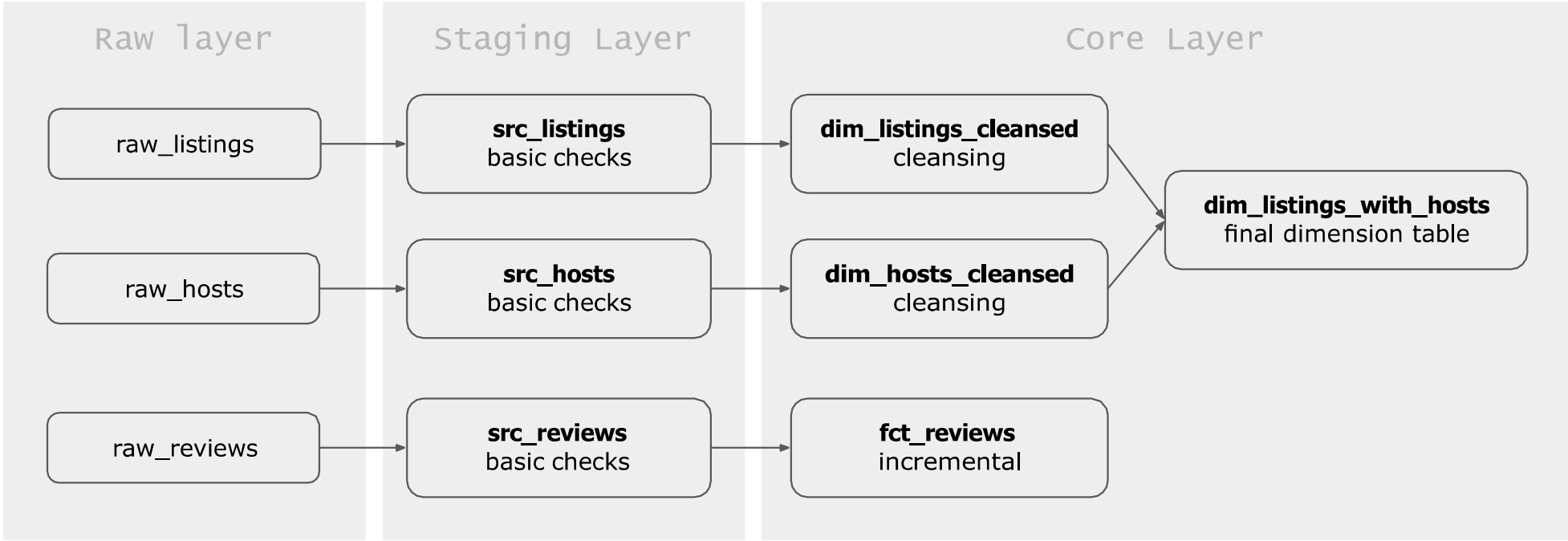
Don't use it
- You want to update historical records

Ephemeral (CTEs)

Use it
- You merely want an alias to your date

Don't use it
- You read from the same model several times

DATA FLOW PROGRESS



GUIDED EXERCISE

dim_hosts_cleansed.sql

Create a new model in the `models/dim/` folder called `dim_hosts_cleansed.sql`.

- Use a CTE to reference the `src_hosts` model
- SELECT every column and every record, and add a cleansing step to *host_name*:
  - If *host_name* is not null, keep the original value
  - If *host_name* is null, replace it with the value *'Anonymous'*
  - Use the *NVL(column_name, default_null_value)* function
- Execute `dbt run` and verify that your model has been created

SOURCES & SEEDS

LEARNING OBJECTIVES

- Understand the difference between seeds and sources
- Understand source-freshness
- Integrate sources into our project

SOURCES AND SEEDS OVERVIEW

- Seeds are local files that you upload to the data warehouse from dbt
- Sources is an abstraction layer on the top of your input tables
- Source freshness can be checked automatically

SNAPSHOTS

LEARNING OBJECTIVES

- Understand how dbt handles type-2 slowly changing dimensions
- Understand snapshot strategies
- Learn how to create snapshots on top of our *listings* and *hosts* models

SNAPSHOTS

Overview

TYPE-2 SLOWLY CHANGING DIMENSIONS

host_id	host_name	email
1	Alice	alice.airbnb@gmail.com
2	Bob	**bob.airbnb@gmail.com**

TYPE-2 SLOWLY CHANGING DIMENSIONS

host_id	host_name	email
1	Alice	alice.airbnb@gmail.com
2	Bob	**bobs.new.address@gmail.com**

TYPE-2 SLOWLY CHANGING DIMENSIONS

host_id	host_name	email	dbt_valid_from	dbt_valid_to
1	Alice	alice.airbnb@gmail.com	2022-01-01 00:00:00	null
2	Bob	bob.airbnb@gmail.com	2022-01-01 00:00:00	2022-03-01 12:53:20
3	Bob	**bobs.new.address@gmail.com**	2022-03-01 12:53:20	null

CONFIGURATION AND STRATEGIES

- Snapshots live in the *snapshots* folder
- Strategies:
  - *Timestamp:* A **unique key** and an **updated_at** field is defined on the source model. These columns are used for determining changes.
  - *Check:* Any change in a set of columns (or all columns) will be picked up as an update.

GUIDED EXERCISE

scd_raw_hosts.sql

Create a new snapshot in the `snapshots/` folder called `scd_raw_hosts.sql`.

- Set the target table name to *scd_raw_hosts*
- Set the output schema to *dev*
- Use the *timestamp* strategy, figure out the unique key and updated_at column to use
- Execute `dbt snapshot` and verify that your snapshot has been created

(You can find the solution among the resources)

TESTS

LEARNING OBJECTIVES

- Understand how tests can be defined
- Configure built-in generic tests
- Create your own singular tests

TESTS OVERVIEW

- There are two types of tests: *singular* and *generic*
- Singular tests are SQL queries stored in tests which are expected to return an empty resultset
- There are four built-in generic tests:
  - unique
  - not_null
  - accepted_values
  - Relationships
- You can define your own custom generic tests or import tests from dbt packages (will discuss later)

GUIDED EXERCISE

TEST dim_hosts_cleansed

Create a generic tests for the `dim_hosts_cleansed` model.

- *host_id:* Unique values, no nulls
- *host_name* shouldn't contain any null values
- *Is_superhost* should only contain the values *t* and *f*.
- Execute `dbt test` to verify that your tests are passing
- **Bonus: Figure out which tests to write for `fct_reviews` and implement them**

(You can find the solution among the resources)

MACROS, CUSTOM TESTS AND PACKAGES

LEARNING OBJECTIVES

- Understand how macros are created
- Use macros to implement your own generic tests
- Find and install third-party dbt packages

MACROS, CUSTOM TESTS AND PACKAGES

- Macros are jinja templates created in the *macros* folder
- There are many built-in macros in DBT
- You can use macros in model definitions and tests
- A special macro, called *test,* can be used for implementing your own generic tests
- dbt packages can be installed easily to get access to a plethora of macros and tests

DOCUMENTATION

LEARNING OBJECTIVES

- Understand how to document models
- Use the documentation generator and server
- Add assets and markdown to the documentation
- Discuss dev vs. production documentation serving

DOCUMENTATION OVERVIEW

- Documentations can be defined two ways:
  - In yaml files (like *schema.yml*)
  - In standalone markdown files
- Dbt ships with a lightweight documentation web server
- For customizing the landing page, a special file, *overview.md* is used
- You can add your own assets (like images) to a special project folder

ANALYSES, HOOKS AND EXPOSURES

LEARNING OBJECTIVES

- Understand how to store ad-hoc analytical queries in dbt
- Work with dbt hooks to manage table permissions
- Build a dashboard in Preset
- Create a dbt exposure to document the dashboard

HOOKS

- Hooks are SQLs that are executed at predefined times
- Hooks can be configured on the project, subfolder, or model level
- Hook types:
  - on_run_start: executed at the start of *dbt {run, seed, snapshot}*
  - on_run_end: executed at the end of *dbt {run, seed, snapshot}*
  - pre-hook: executed before a model/seed/snapshot is built
  - post-hook: executed after a model/seed/snapshot is built

HERO

ORCHESTRATION

THE ORCHESTRATION LANDSCAPE

Apache Airflow

Azure Data Factory

PREFECT

dagster

dbt Cloud

DAGSTER - SIMILAR DATA CONCEPTS

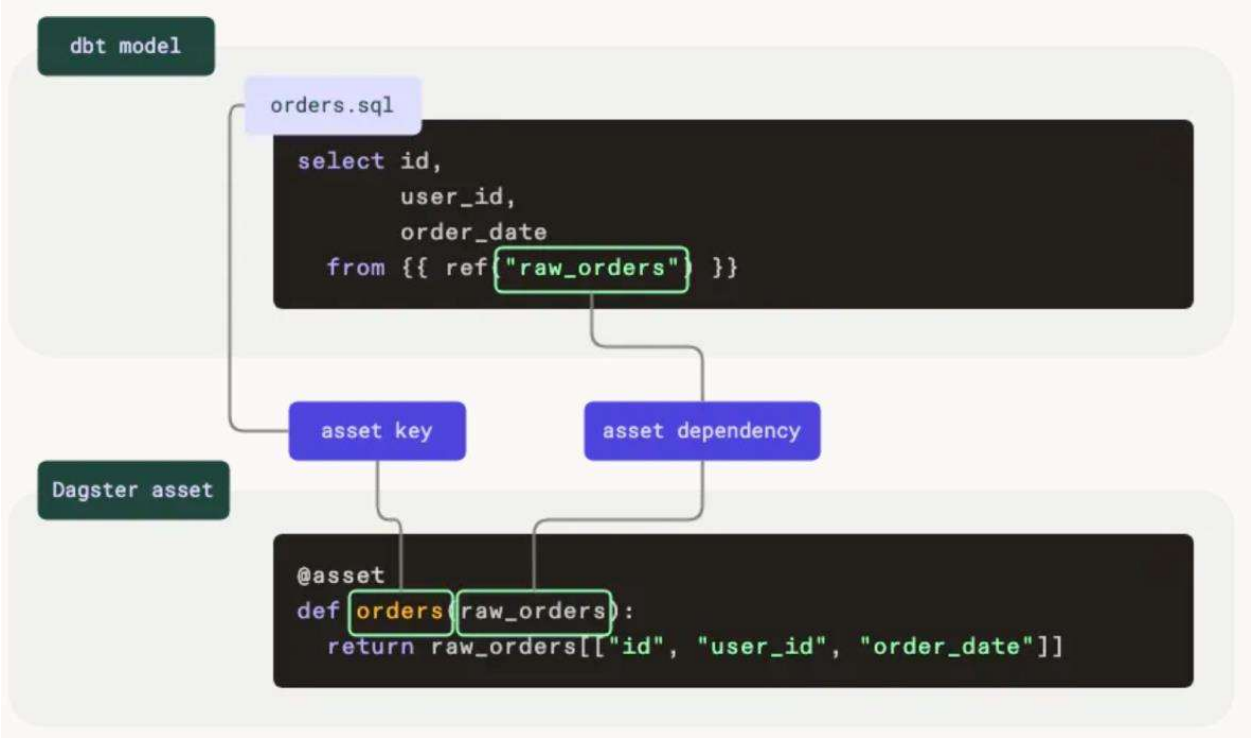


Image source: https://docs.dagster.io/integrations/dbt