

Course Introduction

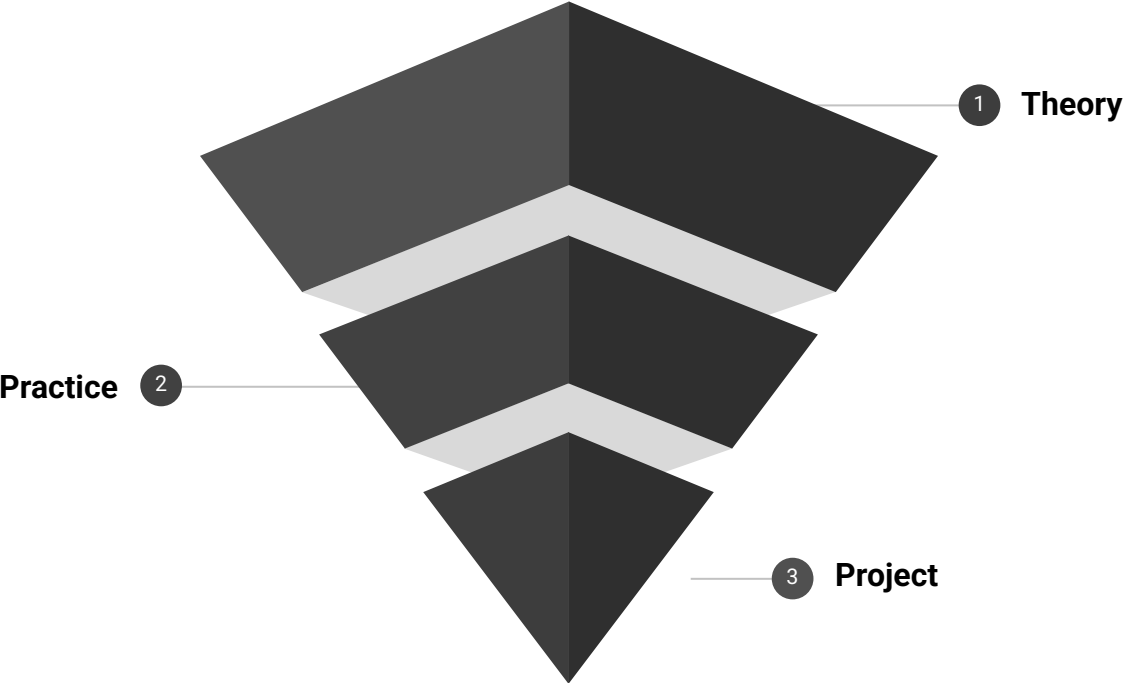


OBJECTIVES

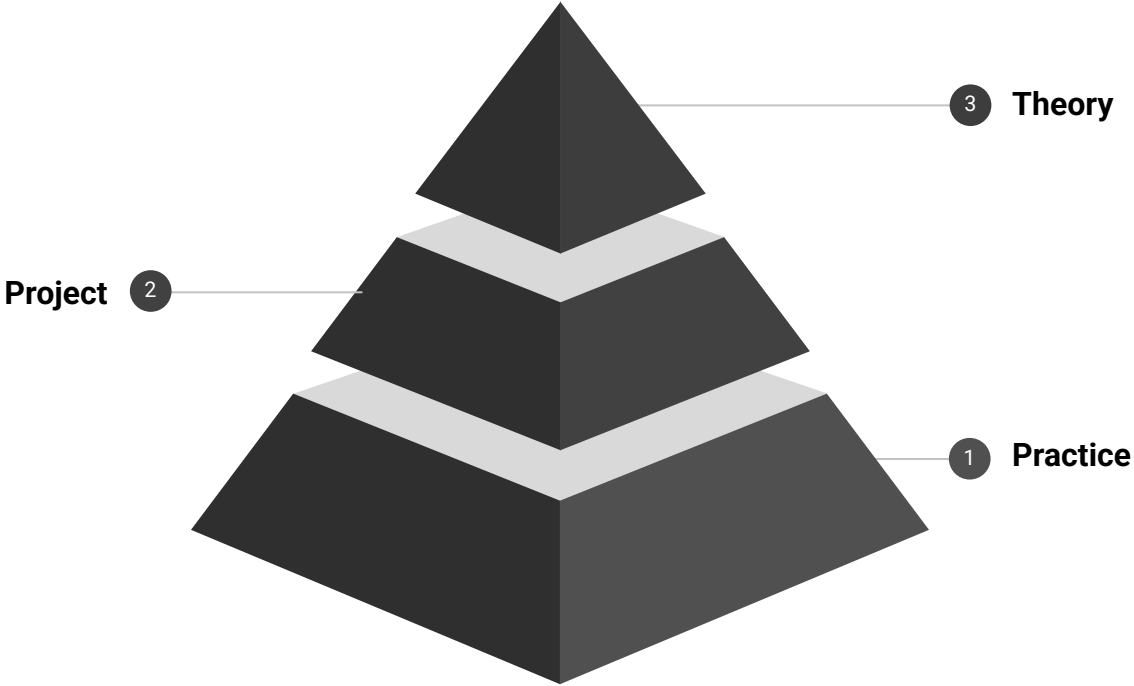
- Data-maturity model
- dbt and data architectures
- Data warehouses, data lakes, and lakehouses
- ETL and ELT procedures
- dbt fundamentals
- Analytics Engineering



TOP-DOWN



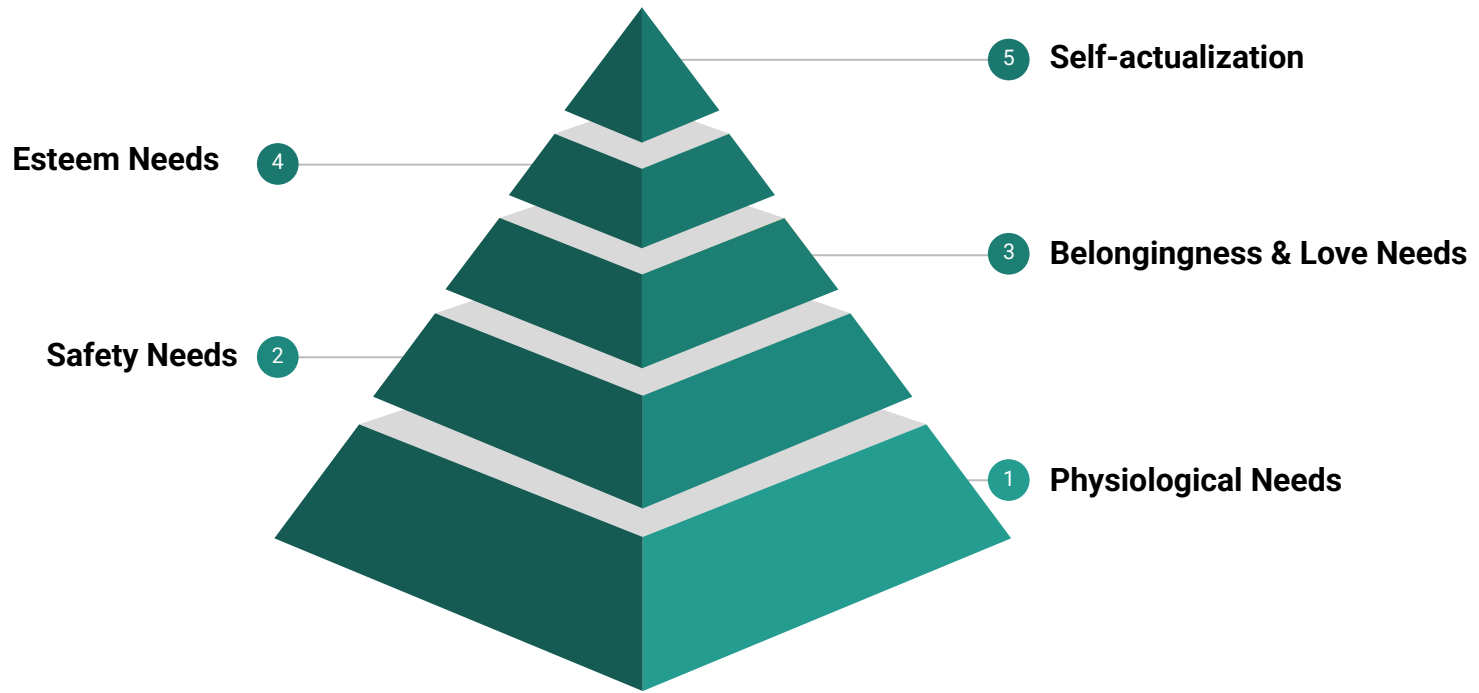
BOTTOM-UP



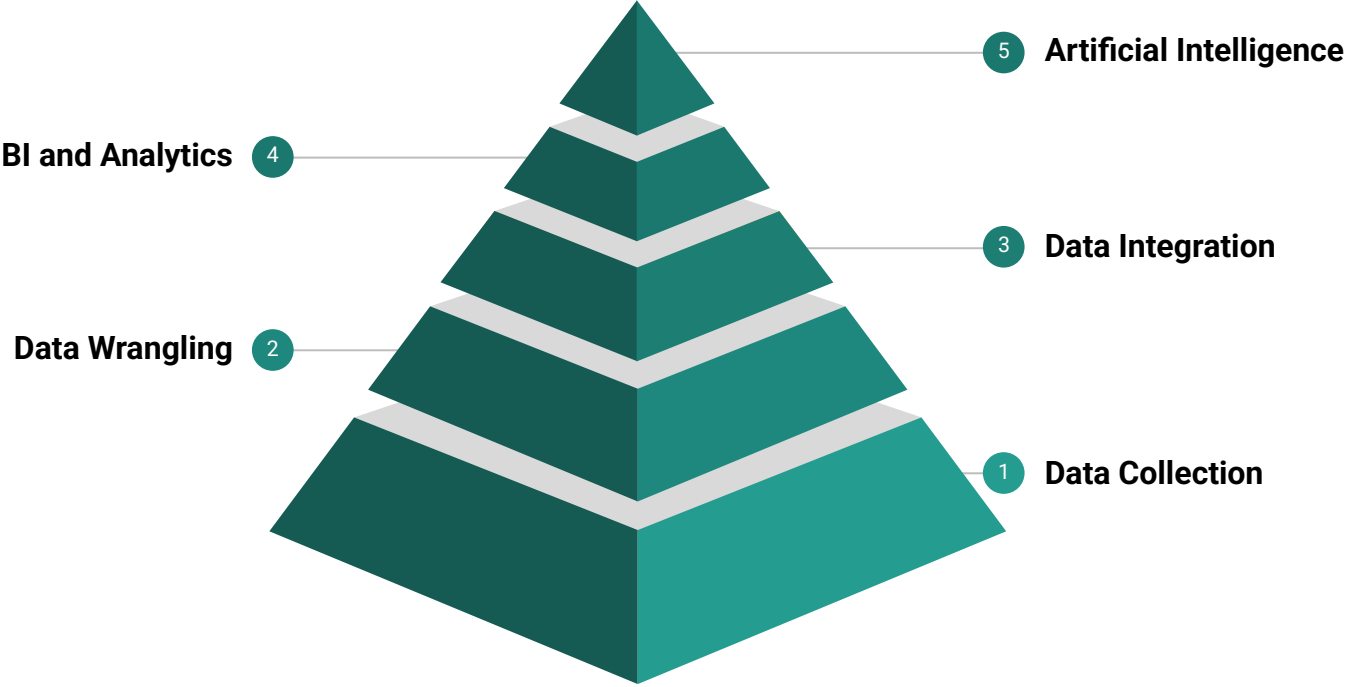
Data Maturity Model



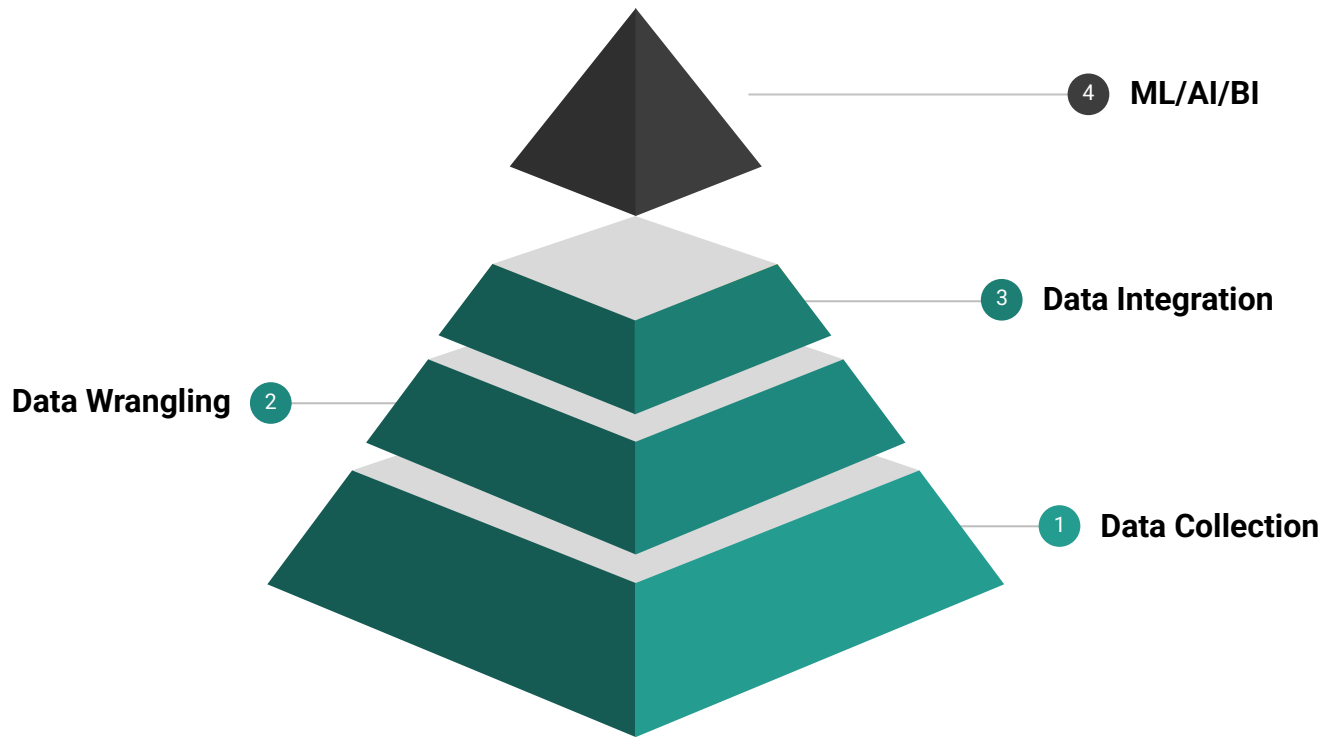
Maslow's Hierarchy of Needs



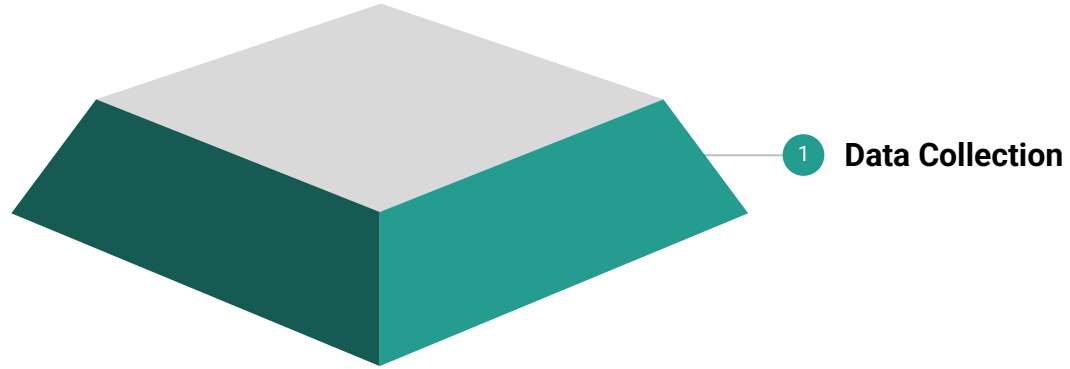
Data-Maturity Model



Typical Data Architecture



Data Collection



Data Wrangling



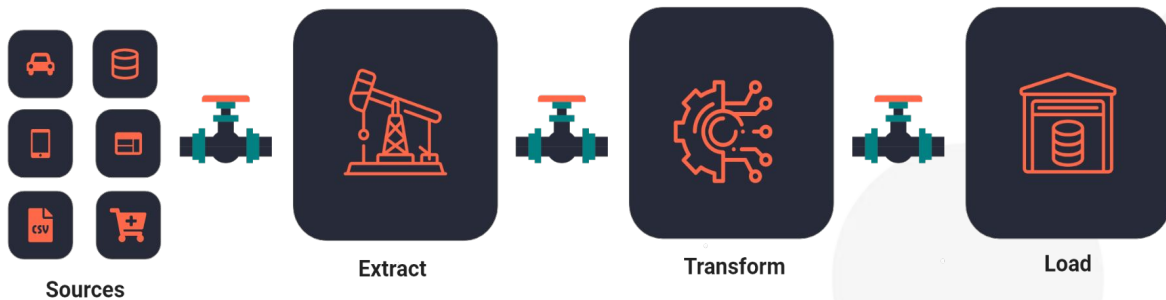
Data Integration



ETL - ELT #

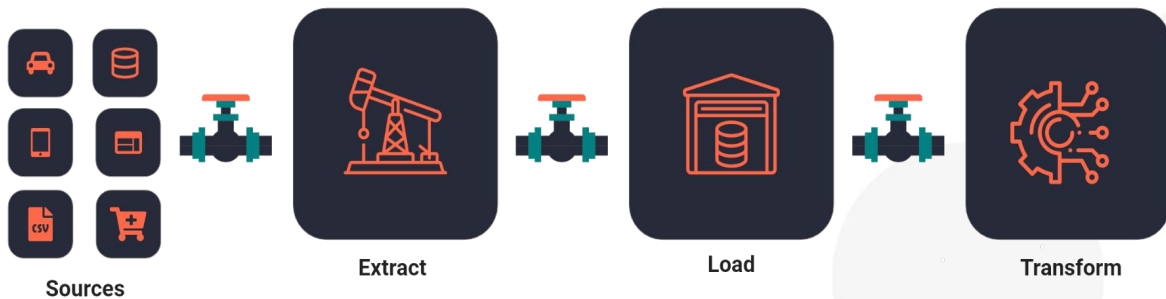
ETL

ETL
Extract, Transform, Load

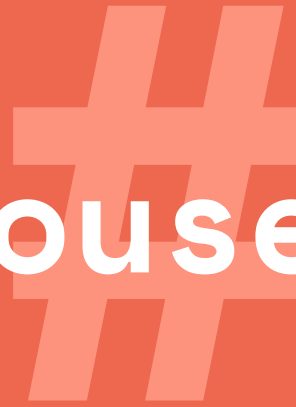


ELT

ELT
Extract, Load, Transform



Data Warehouse



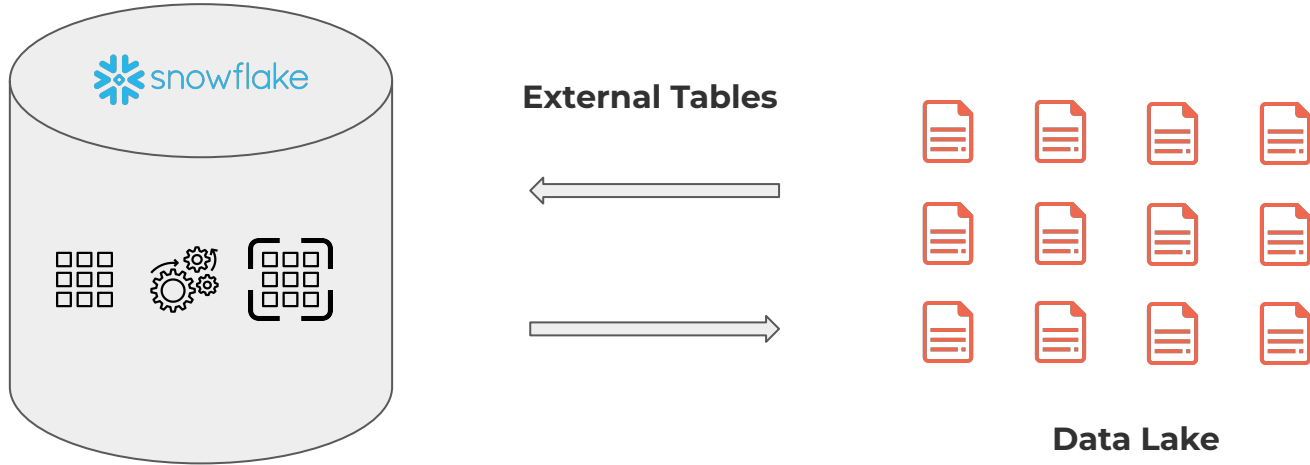
Data Warehouse



External Tables



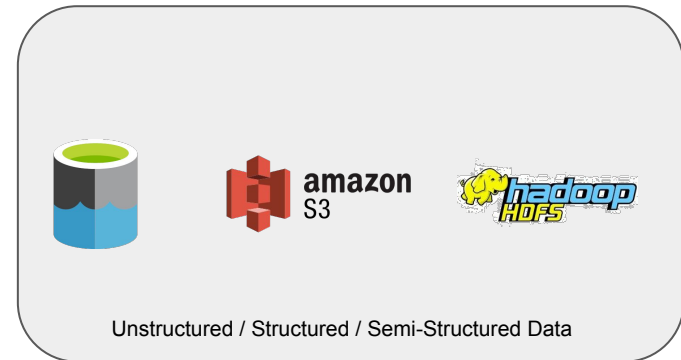
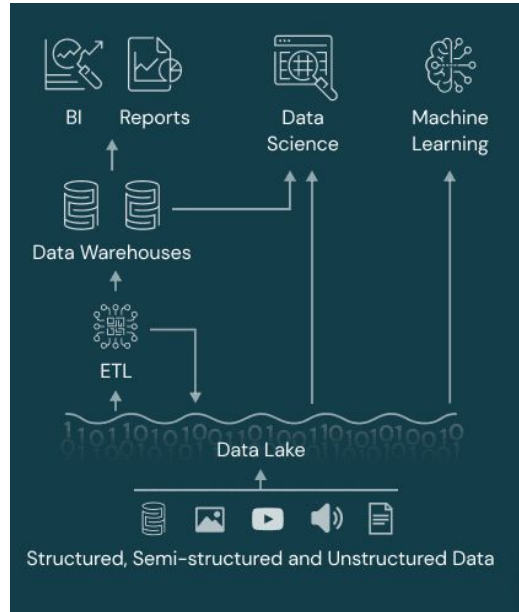
External Tables



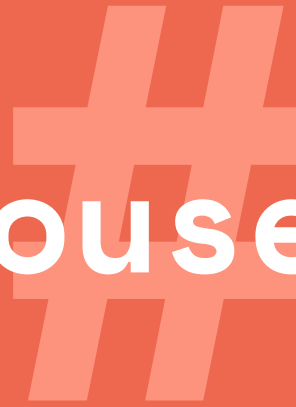
Data Lake



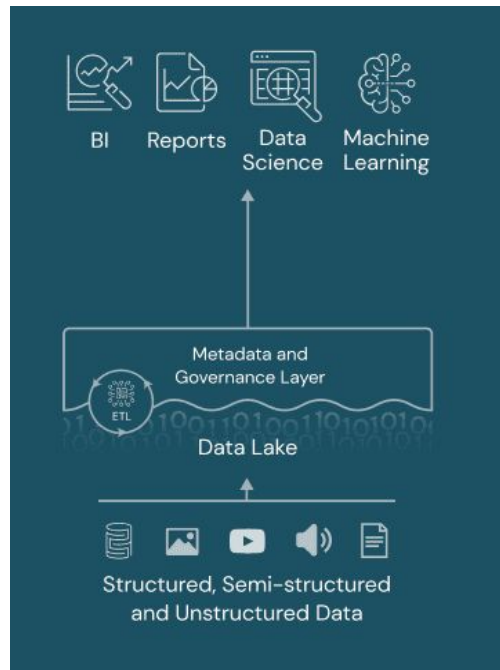
Data Lake



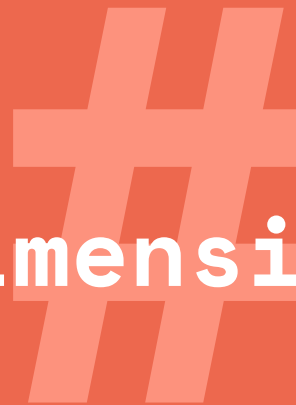
Data Lakehouse



Data Lakehouse



Slowly Changing Dimensions



SCD Type 0

Not updating the DWH table when a Dimension changes

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

SCD Type 1

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

No Air-conditioning

Host ID	Air-conditioning
#1	No

Installed Air-conditioning

Host ID	Air-conditioning
#1	Yes

DWH updated

Host ID	Air-conditioning
#1	Yes

DWH

Host ID	Air-conditioning
#1	No

Host ID	Air-conditioning
#1	No

Host ID	Air-conditioning
#1	Yes

SCD Type 2

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

Current rental price (\$300)

Apartment ID	Price
#1	\$300

Change in the rental price (\$450)

Apartment ID	Price
#1	\$450

DWH updated

Source		DWH																						
<table><tr><th>Apartment ID</th><th>Price</th></tr><tr><td>#1</td><td>\$300</td></tr></table>	Apartment ID	Price	#1	\$300		<table><tr><th>Host Key</th><th>Apartment ID</th><th>Price</th><th>Start_Date</th><th>End_Date</th></tr><tr><td>1000</td><td>#1</td><td>\$300</td><td>2020-01-01T00:00:00</td><td>NULL</td></tr></table>	Host Key	Apartment ID	Price	Start_Date	End_Date	1000	#1	\$300	2020-01-01T00:00:00	NULL								
	Apartment ID	Price																						
#1	\$300																							
Host Key	Apartment ID	Price	Start_Date	End_Date																				
1000	#1	\$300	2020-01-01T00:00:00	NULL																				
<table><tr><th>Apartment ID</th><th>Price</th></tr><tr><td>#1</td><td>\$450</td></tr></table>	Apartment ID	Price	#1	\$450		<table><tr><th>Host Key</th><th>Apartment ID</th><th>Price</th><th>Start_Date</th><th>End_Date</th></tr><tr><td>1000</td><td>#1</td><td>\$300</td><td>2020-01-01T00:00:00</td><td>2021-01-01T00:00:00</td></tr></table>	Host Key	Apartment ID	Price	Start_Date	End_Date	1000	#1	\$300	2020-01-01T00:00:00	2021-01-01T00:00:00								
	Apartment ID	Price																						
#1	\$450																							
Host Key	Apartment ID	Price	Start_Date	End_Date																				
1000	#1	\$300	2020-01-01T00:00:00	2021-01-01T00:00:00																				
<table><tr><th>Apartment ID</th><th>Price</th></tr><tr><td>#1</td><td>\$450</td></tr></table>	Apartment ID	Price	#1	\$450		<table><tr><th>Host Key</th><th>Apartment ID</th><th>Price</th><th>Start_Date</th><th>End_Date</th></tr><tr><td>1000</td><td>#1</td><td>\$300</td><td>2020-01-01T00:00:00</td><td>2021-01-01T00:00:00</td></tr><tr><td>1001</td><td>#1</td><td>\$450</td><td>2021-01-01T00:00:00</td><td>NULL</td></tr></table>	Host Key	Apartment ID	Price	Start_Date	End_Date	1000	#1	\$300	2020-01-01T00:00:00	2021-01-01T00:00:00	1001	#1	\$450	2021-01-01T00:00:00	NULL			
	Apartment ID	Price																						
	#1	\$450																						
Host Key	Apartment ID	Price	Start_Date	End_Date																				
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

Current rental price (\$300)

Apartment ID	Price
#1	\$300

Change in the rental price (\$450)

Apartment ID	Price
#1	\$450

DWH updated

Source		DWH						
	Apartment ID	Price	Host Key	Apartment ID	Price	Start_Date	End_Date	Is_Current
	#1	\$300	1000	#1	\$300	2020-01-01T00:00:00	9999-12-12T00:00:00	Y
	Apartment ID	Price	Host Key	Apartment ID	Price	Start_Date	End_Date	Is_Current
	#1	\$450	1000	#1	\$300	2020-01-01T00:00:00	9999-12-12T00:00:00	N
	Apartment ID	Price	Host Key	Apartment ID	Price	Start_Date	End_Date	Is_Current
	#1	\$450	1000	#1	\$300	2020-01-01T00:00:00	2021-01-01T00:00:00	N
			1001	#1	\$450	2021-01-01T00:00:00	9999-12-12T00:00:00	Y

SCD Type 3

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

Listed as Private

Source	DWH										
<table><tr><th>Apartment ID</th><th>Type</th></tr><tr><td>#1</td><td>Private</td></tr></table>	Apartment ID	Type	#1	Private	<table><tr><th>Apartment ID</th><th>Previous Type</th><th>Current Type</th></tr><tr><td>#1</td><td>Private</td><td>Private</td></tr></table>	Apartment ID	Previous Type	Current Type	#1	Private	Private
Apartment ID	Type										
#1	Private										
Apartment ID	Previous Type	Current Type									
#1	Private	Private									
<table><tr><th>Apartment ID</th><th>Type</th></tr><tr><td>#1</td><td>Entire</td></tr></table>	Apartment ID	Type	#1	Entire	<table><tr><th>Apartment ID</th><th>Previous Type</th><th>Current Type</th></tr><tr><td>#1</td><td>Private</td><td>Entire</td></tr></table>	Apartment ID	Previous Type	Current Type	#1	Private	Entire
Apartment ID	Type										
#1	Entire										
Apartment ID	Previous Type	Current Type									
#1	Private	Entire									
<table><tr><th>Apartment ID</th><th>Type</th></tr><tr><td>#1</td><td>Shared</td></tr></table>	Apartment ID	Type	#1	Shared	<table><tr><th>Apartment ID</th><th>Previous Type</th><th>Current Type</th></tr><tr><td>#1</td><td>Entire</td><td>Shared</td></tr></table>	Apartment ID	Previous Type	Current Type	#1	Entire	Shared
Apartment ID	Type										
#1	Shared										
Apartment ID	Previous Type	Current Type									
#1	Entire	Shared									

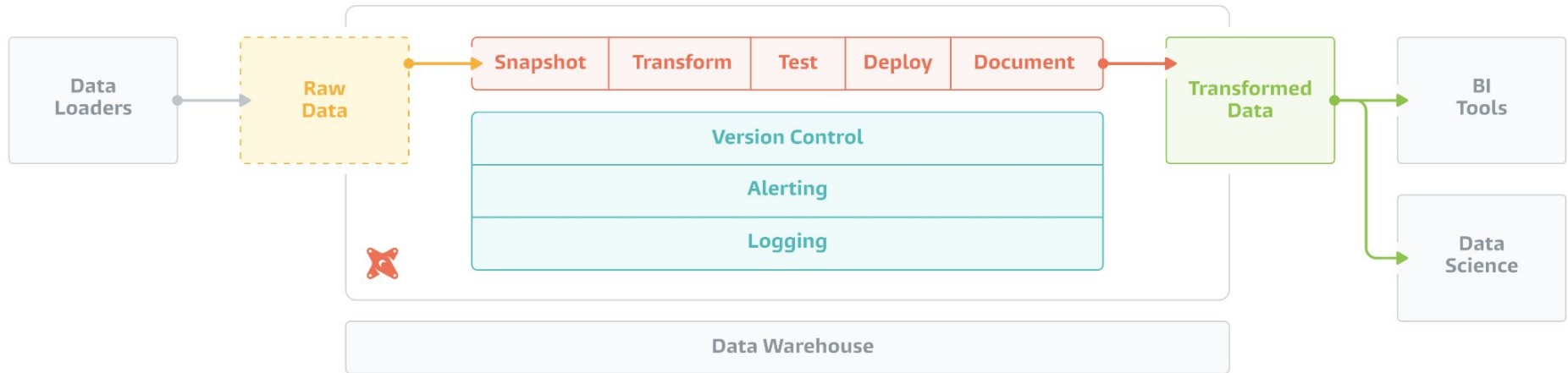
Host changed Private to Entire

Host changed Entire to Shared

dbt Overview



dbt Overview



Analytics Engineering



Common Table Expression (CTE)

A large, stylized orange hashtag symbol (#) is positioned on the right side of the slide, partially overlapping the text "Common Table Expression (CTE)".

CTE

Syntax

SQL ▾

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

CTE

Example

```
-- 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

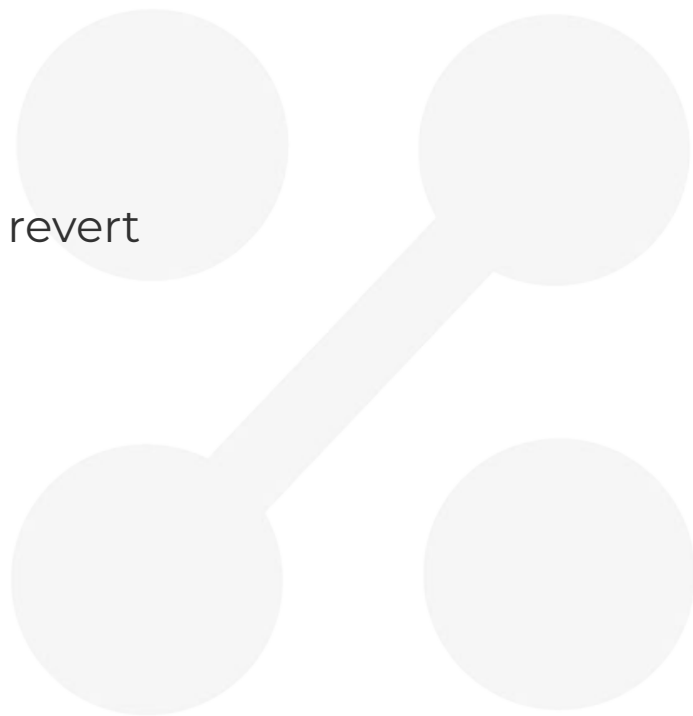


TECH STACK



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



NEXT STEPS - SETUP

- Snowflake registration
- Dataset import
- dbt installation
- dbt setup, snowflake connection

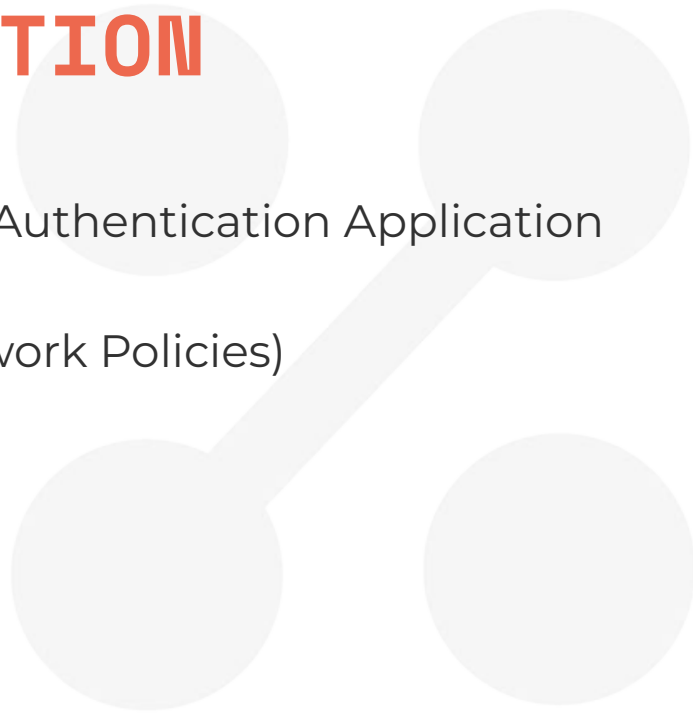
That's it :)



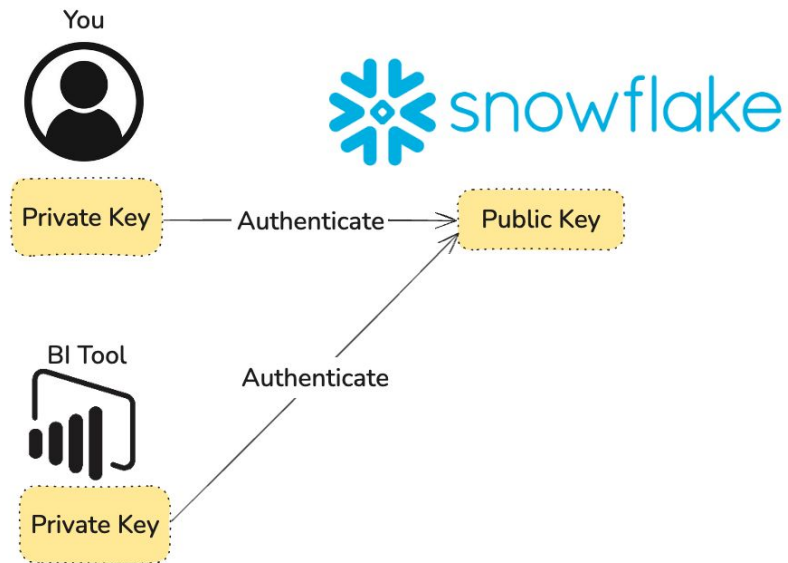
SNOWFLAKE

Registering a Trial account

SNOWFLAKE AUTHENTICATION

- ✗ Username & Password + Multi-Factor Authentication Application
 - ✗ Personal Access Token (Requires Network Policies)
 - ✗ Single Sign-On (SSO)
 - ✓ Keypair-Based Authentication
- 

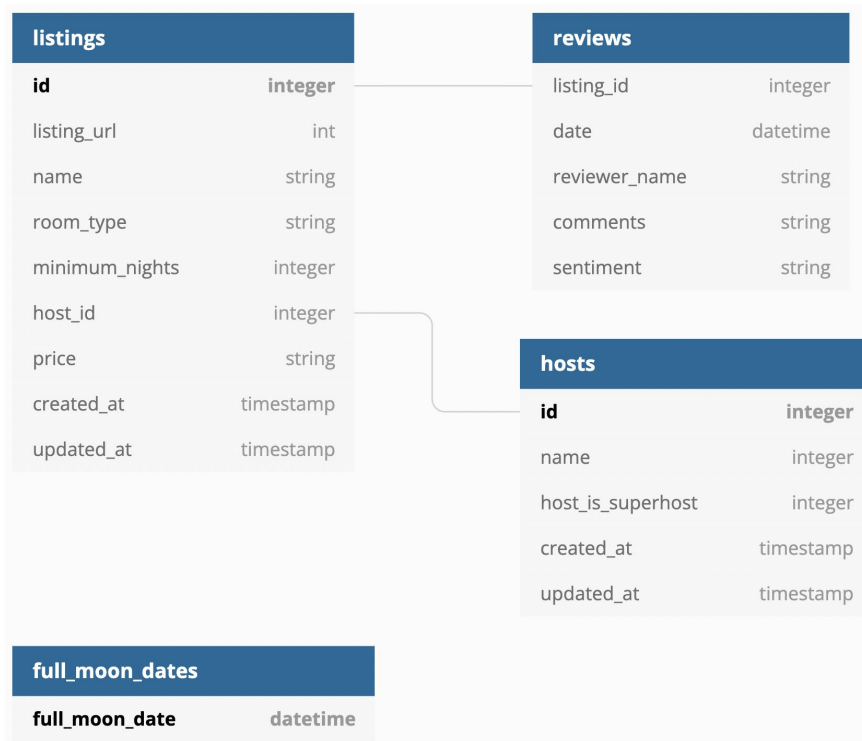
Public-Key Authentication (PKI)



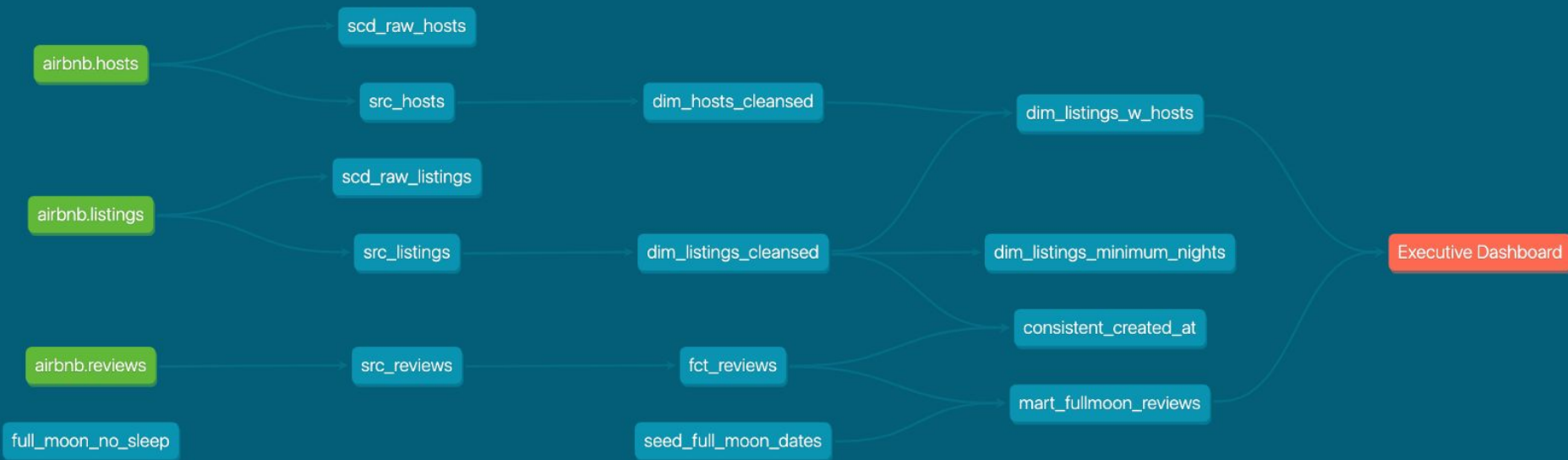
DATA FLOW

Overview

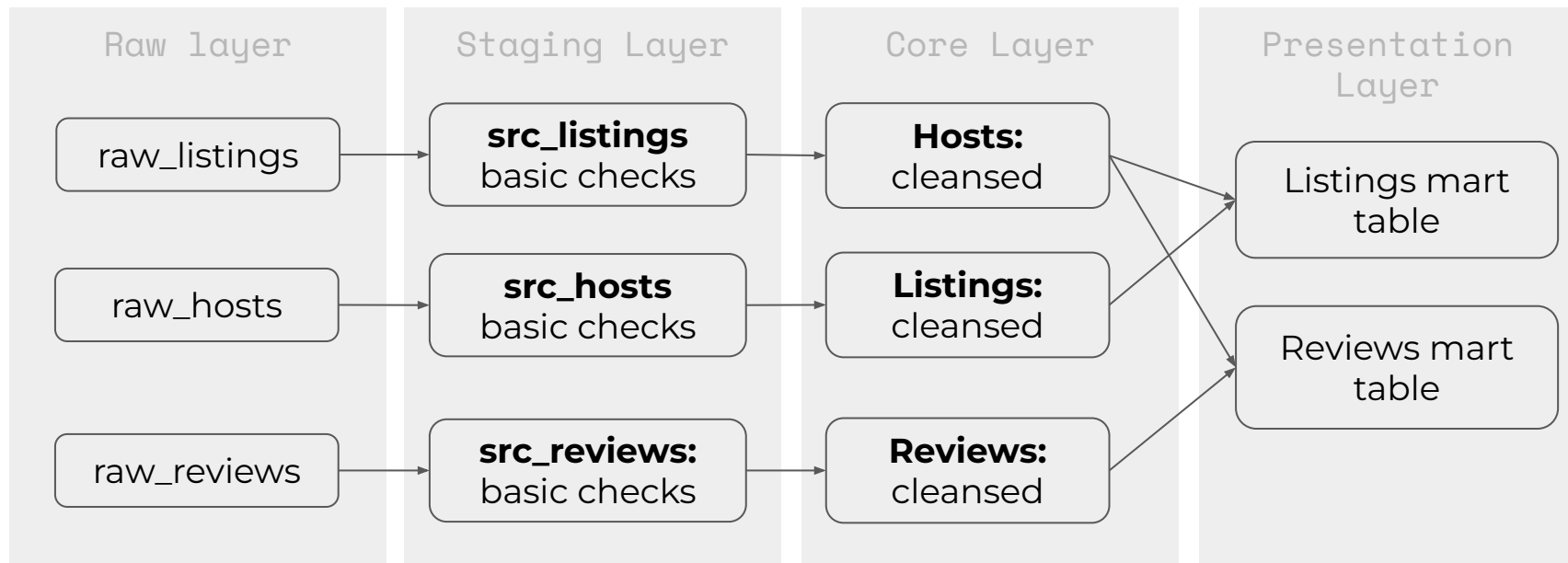
INPUT DATA MODEL



DATA FLOW OVERVIEW



DATA FLOW OVERVIEW



DBT SETUP

Mac

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

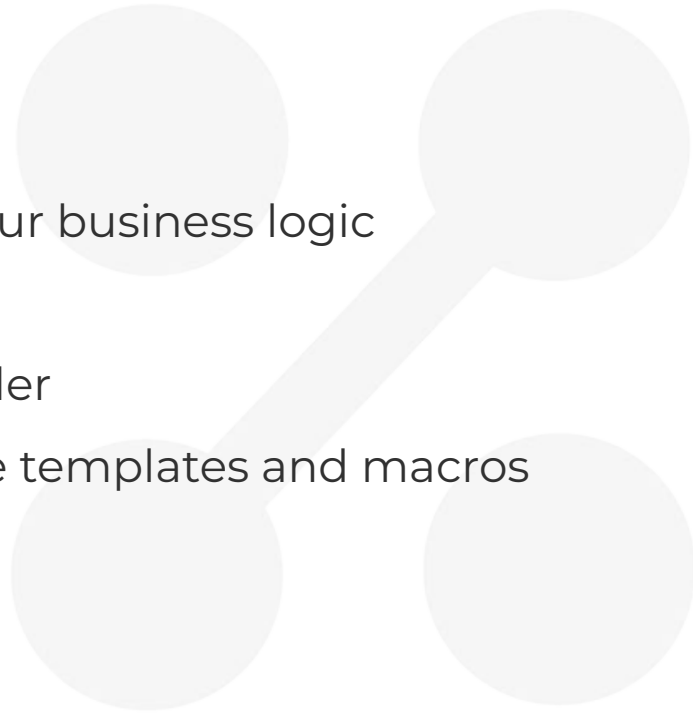
MODELS

LEARNING OBJECTIVES

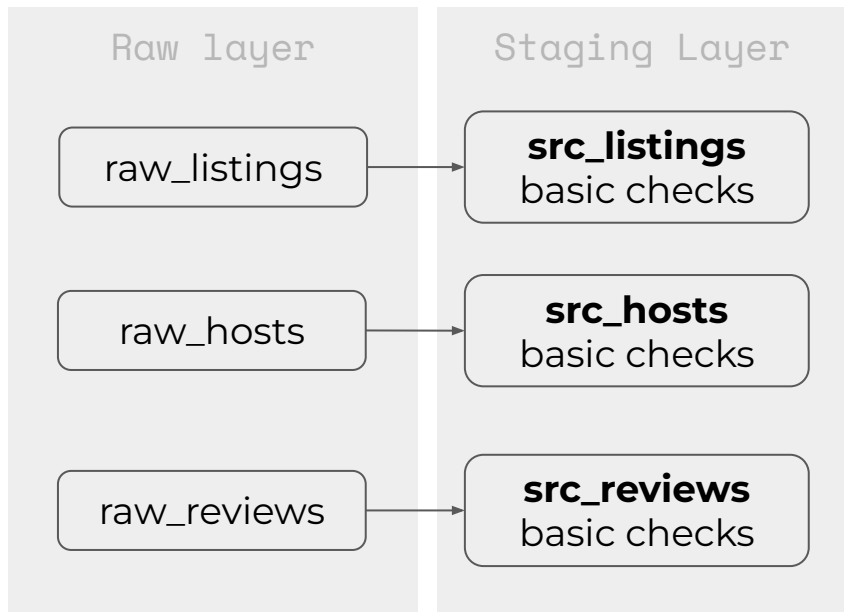
- Understand the data flow of our project
- Understand the concept of Models in dbt
- Create three basic models:
 - src_listings
 - src_reviews: guided exercises
 - src_hosts: individual lab



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``.

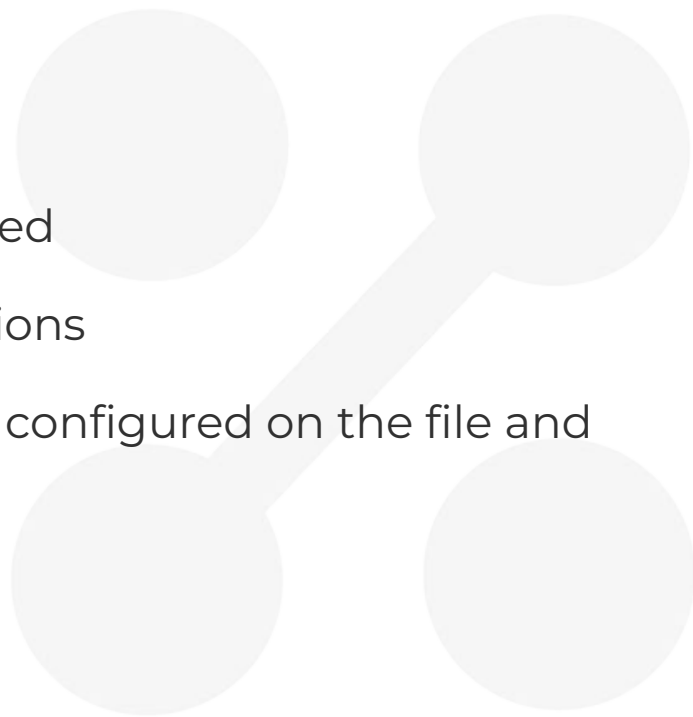
- Use a CTE to reference the AIRBNB.RAW.RAW_REVIEWS table
- SELECT every column and every record, and rename the following columns:
 - *date* to *review_date*
 - *comments* to *review_text*
 - *sentiment* to *review_sentiment*
- Execute ``dbt run`` and verify that your model has been created

(You can find the solution among the resources)

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)

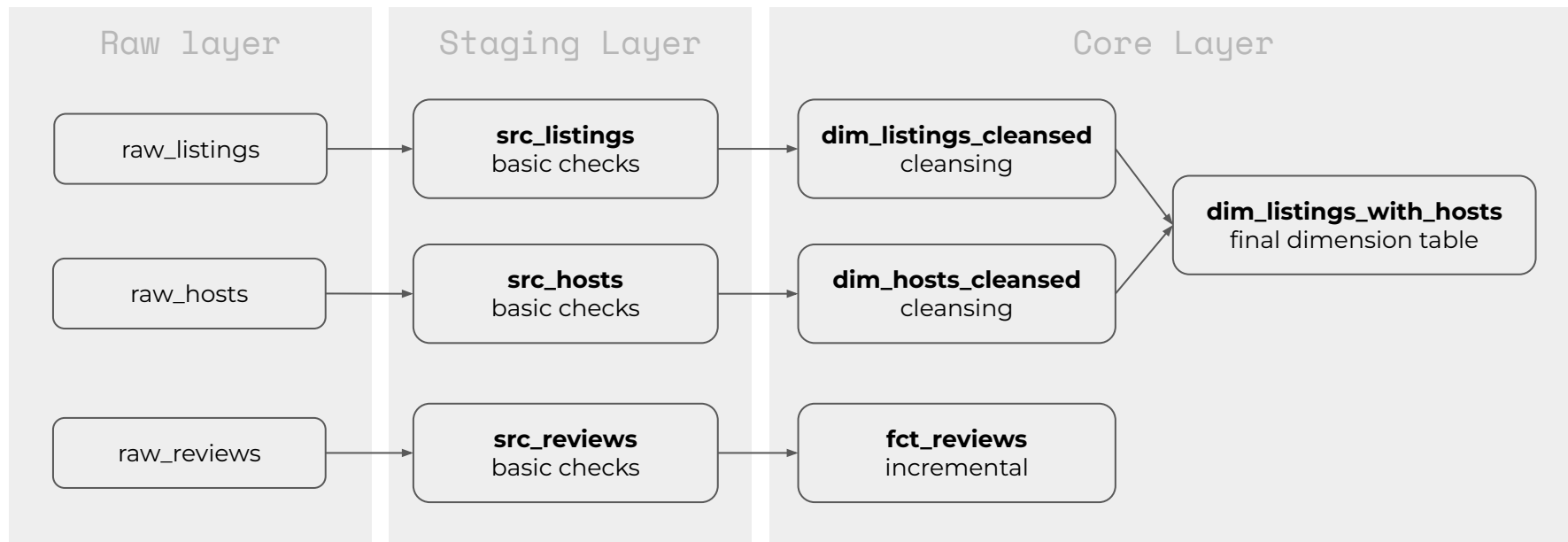
Use it

- You merely want an alias to your data

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

(You can find the solution among the resources)


SOURCES & SEEDS

LEARNING OBJECTIVES

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

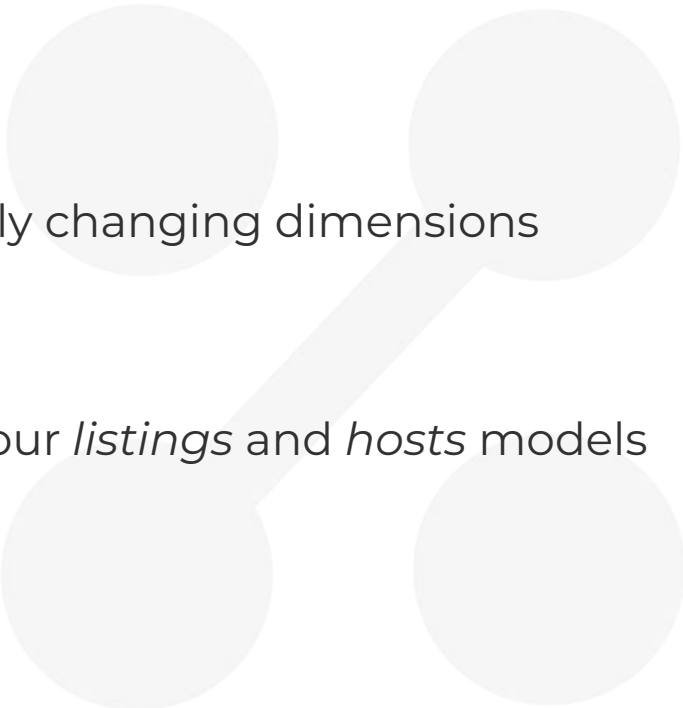


SOURCES AND SEEDS OVERVIEW

- 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.

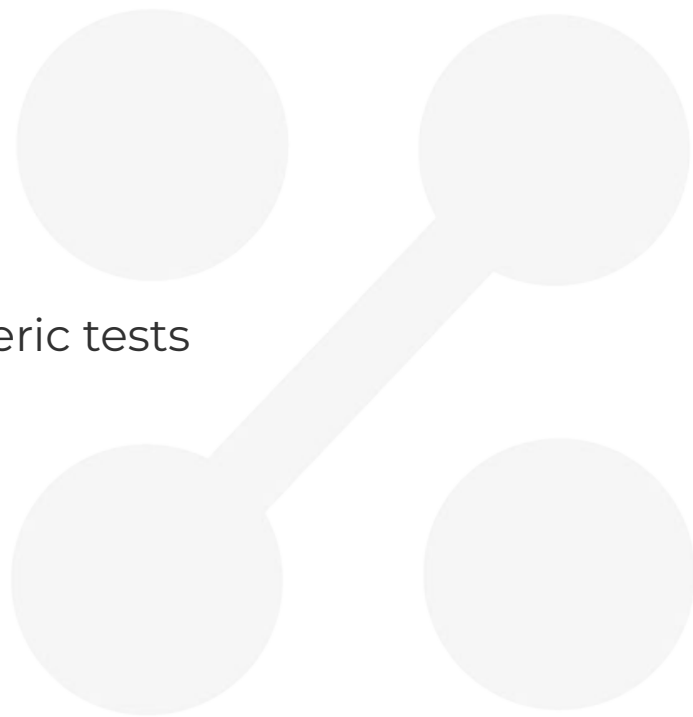
- *host_id*: Unique values, no nulls
- *host_name* shouldn't contain any null values
- *Is_superhost* should only contain the values *t* and *f*.
- Execute ``dbt test`` to verify that your tests are passing
- **Bonus: Figure out which tests to write for ``fct_reviews`` and implement them**

(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



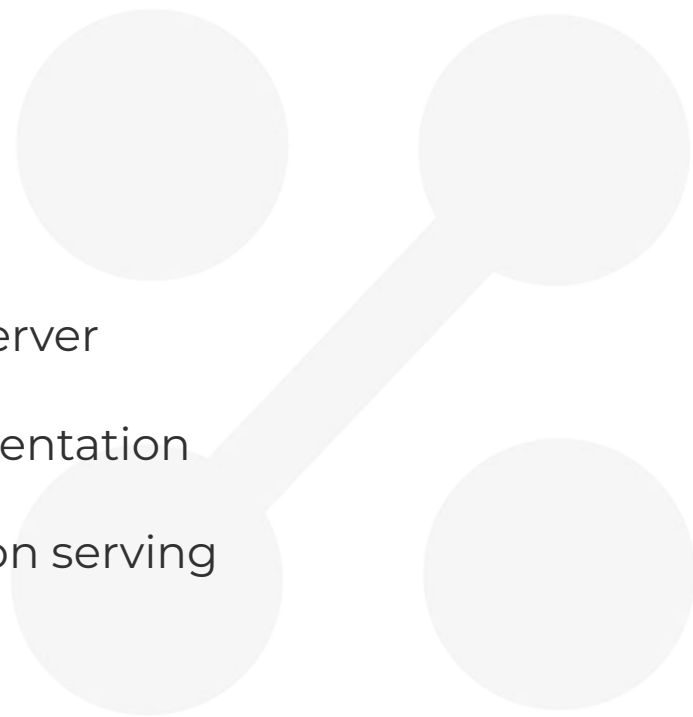
MACROS, CUSTOM TESTS AND PACKAGES

- 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

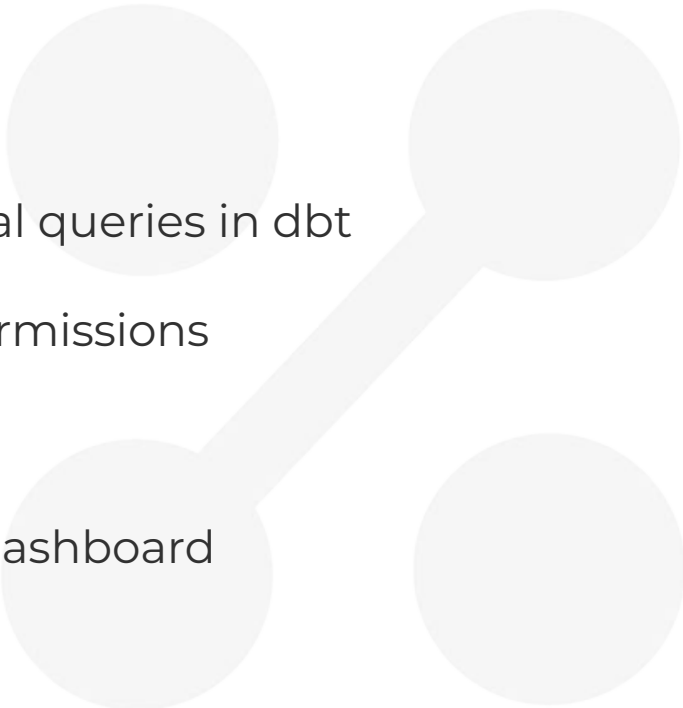


DOCUMENTATION OVERVIEW

- 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
-

dbt Fusion

&

The Official VSCode Extension
(THIS SECTION IS WORK IN PROGRESS
AND WILL BE RELEASED MID OCTOBER)

A new Engine for dbt

- Completely rewritten from the ground up in Rust
 - Magnitudes-Faster Parsing
 - “Code Comprehension Engine”
- Released as Source-available from dbt Labs
- Released along an Official Visual Studio Code Extension

Parsing a 10,000 model project

Cold parse, Rust	-	0.00s
Partial parse, Python	—	0.00s
Cold parse, Python	—————	0.00s

LEARNING OBJECTIVES

- Understand how the Parser and Compiler work
- Feature Matrix - dbt Fusion vs dbt Core
- License differences
- Get hands-on with dbt Fusion and the VSCode Extension



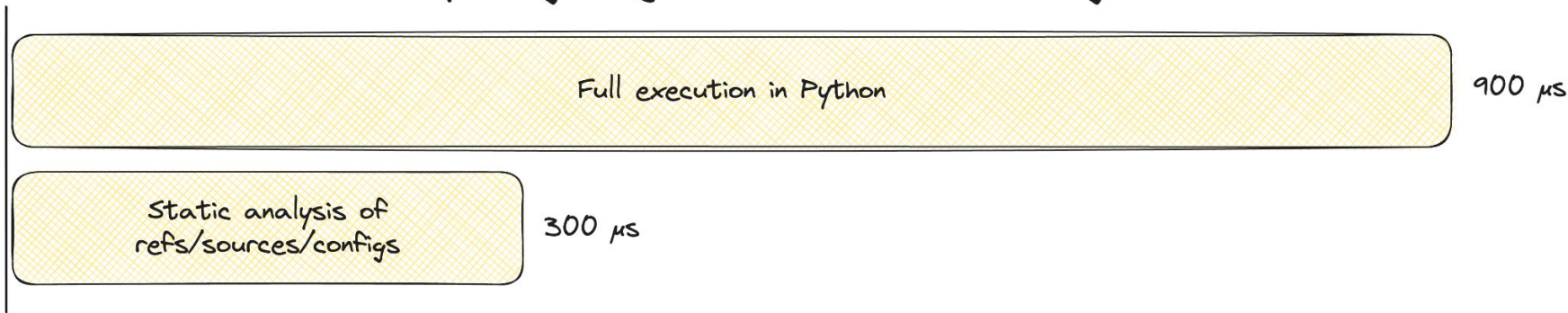
dbt Fusion

Technical Deep Dive

(THIS SECTION IS WORK IN PROGRESS
AND WILL BE RELEASED MID OCTOBER)

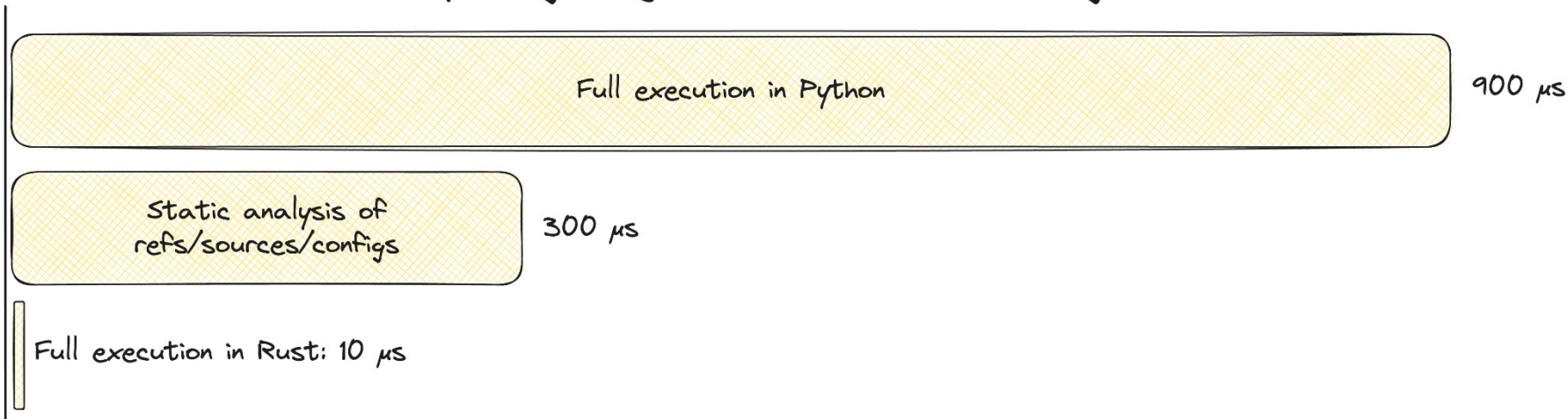
PARSING

Comparing Jinja evaluation strategies



PARSING WITH RUST

Comparing Jinja evaluation strategies



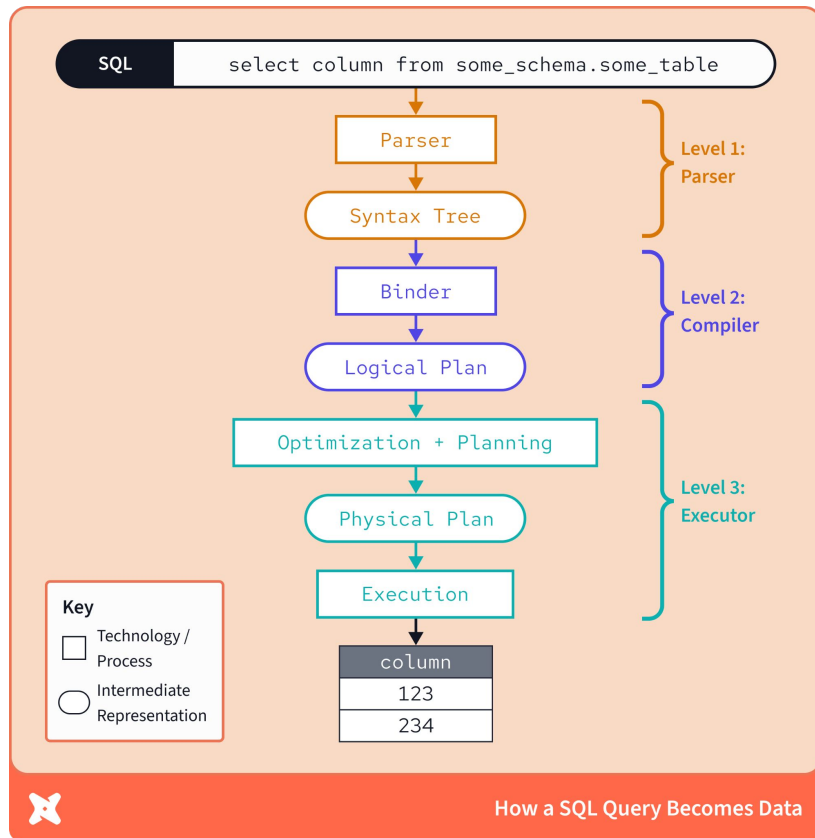
SQL COMPREHENSION

The Levels of SQL Comprehension: What can be validated	
Input	<code>select dateadd('day', 1, '2025-01-01') as new_day</code>
Level 1: Parser Syntax only	<code>select █████(█████, █, █████) as new_day</code>
Level 2: Compiler Level 1 + function names, argument count and data types	<code>select dateadd(datepart, int, date_or_time_expr) as new_day</code>
Level 3: Executor Level 2 + the data itself	<code>select dateadd('day', 1, '2025-01-01') as new_day</code> (validates everything!)

Source: <https://docs.getdbt.com/blog/faster-project-parsing-with-rust>

SQL COMPREHENSION

- **dbt Core:** let the Data Warehouse compile the SQL code
- **dbt Fusion:** let's compile and comprehend as much as possible without touching the DWH

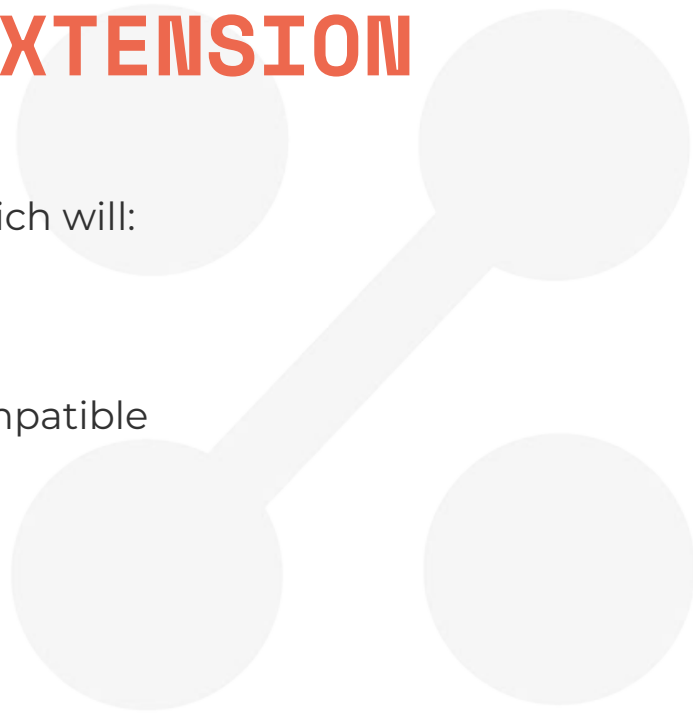


dbt Fusion License

- dbt Core: **Apache 2**
- dbt Fusion: Mostly Elastic License v2 (**ELv2**)
 - You can use it for free, but you cannot provide it as a service for free
- Might include proprietary licensed enterprise features later
- For you most probably the license change won't make a difference

VISUAL STUDIO CODE EXTENSION

Let's go ahead and install the **VSCode Extension**, which will:

- Install dbt Fusion
 - Upgrade your dbt project to be 100% Fusion compatible
 - Compile your project for SQL Comprehension
- 
-

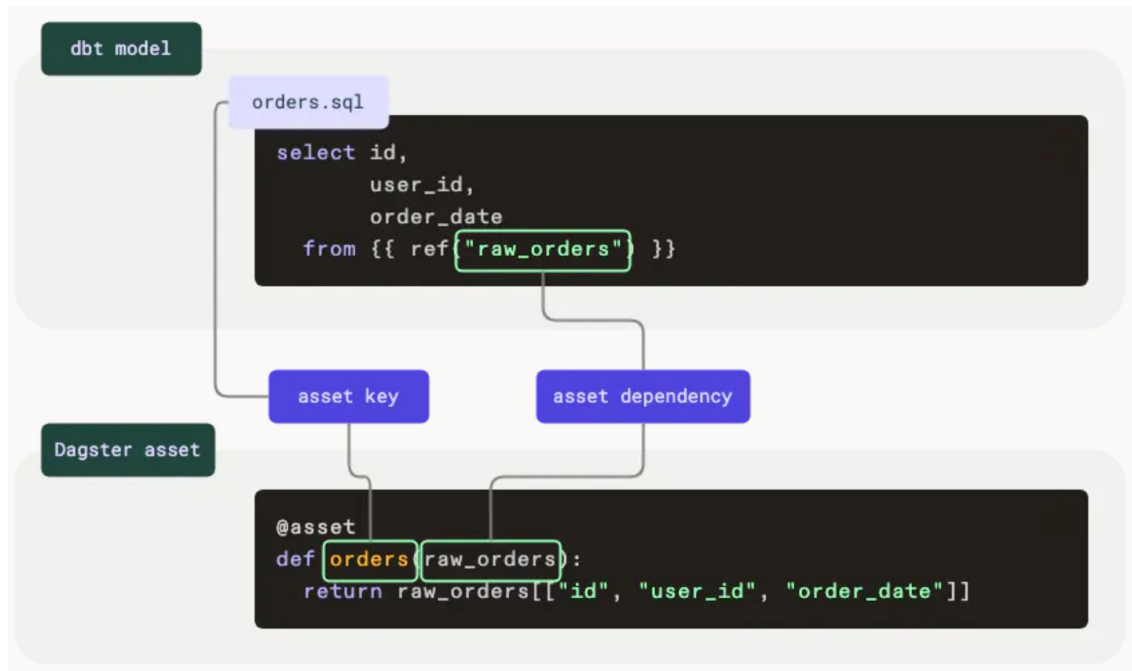
HERO

ORCHESTRATION

THE ORCHESTRATION LANDSCAPE



DAGSTER - SIMILAR DATA CONCEPTS



GREAT UI



☰

Overview

Runs

Assets

Deployment

materialize_dbt_models Job in dbt_dagster_project At 00:00 UTC Latest run: 15 Jan, 10:32 View 8 assets

Overview Runs

Filter

Type an asset subset... (ex: ++dim_listings_w_hosts)

Clear query

Materialize selected

listings

revs

reviews

scd_raw_listings

dim_listings_cleaned

scd_raw_hosts

dim_hosts_cleaned

revs_reviews

seed_full_moon_dates

dim_listings_w_hosts

seed_full_moon_reviews

unique_id

invocation_id

Execution Duration

Materialization system tags

Metadata plots

Execution Duration

Code Version

Config

Required resources

Metadata

View as Asset Graph

10:32:02 a.m.

10:32:14 a.m.

10:32:26 a.m.

10:32:38 a.m.

10:33:02 a.m.

10:33:14 a.m.

10:33:26 a.m.

10:33:38 a.m.

table_schema

dagster_dbt/manifest

dagster_dbt/dagster_dbt_translator

Show Table Schema

[DbtManifestWrapper]

(unserializable)

[DagsterDbtTranslator]

EASY TO DEBUG



Overview Runs Assets Deployment

Search... /

b5aef1ea View job View tags and config

Success Run of materialize_dbt_models @ b6dde2d1 9 assets 15 Jan, 10:32:29 0:01:13

Hide not started steps Re-execute all (*)

30s 60s 90s

Preparing (0) No steps are waiting to execute

Executing (0) No steps are executing

Errored (0) No steps have errored

Type a step subset (ex: dbtlea) Hide unselected steps

Events stdout stderr Filter... Levels (5)

TIMESTAMP	OP	EVENT TYPE	INFO
			step_keys ["dbtlearn_dbt_assets"] captured_logs View stdout / stderr
10:32:32.515	dbtlearn_dbt_assets	STEP_START	Started execution of step "dbtlearn_dbt_assets".
10:32:32.551	dbtlearn_dbt_assets	INFO	A dbt subsetted execution is not being performed. Using the default dbt selection arguments ["--select", "fq:.*"].
10:32:32.553	dbtlearn_dbt_assets	INFO	Copying "/Users/zoltantoth/src/dbt-zero-to-hero/orchestration/dbtlearn/target/partial_parse.msgpack" to "/Users/zoltantoth/src/dbt-zero-to-hero/orchestration/dbtlearn/target/dbtlearn_dbt_assets-b5aef1e-5c7bfdf/partial_parse.msgpack" to take advantage of partial parsing.
10:32:32.555	dbtlearn_dbt_assets	INFO	Running dbt command: "dbt build --select fq:.*".
10:32:43.377	dbtlearn_dbt_assets	STEP_OUTPUT	Yielded output "seed_full_moon_dates" of type "Nothing". (Type check passed). unique_id seed.dbtlearn.seed_full_moon_dates invocation_id 6d8b8365-92f4-4a93-8c40-c527d3d747df Execution Duration 3.139401

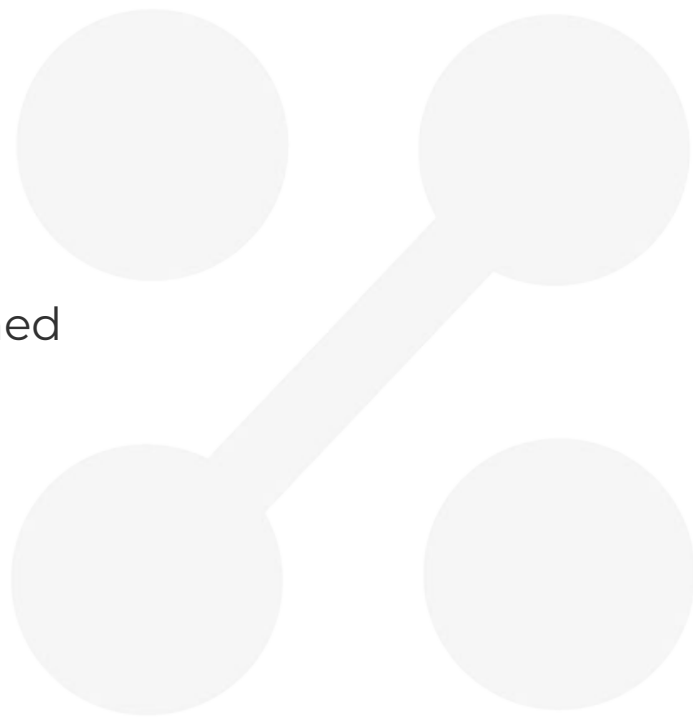
CONGRATS !

**FROM THIS POINT
ON THE SLIDES
ARE WORK IN
PROGRESS**

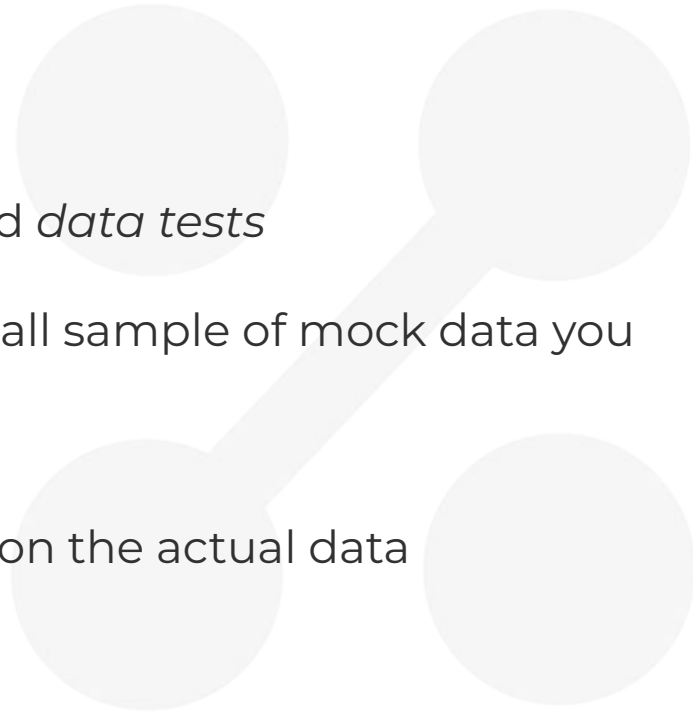
TESTS

LEARNING OBJECTIVES

- Understand Unit Tests and Data Tests
- Understand how these tests can be defined
- Configure built-in generic tests
- Create your own singular tests
- Create Unit Tests for your data



TESTS OVERVIEW

- There are two types of tests: *unit tests* and *data tests*
 - Unit Tests test transformations with a small sample of mock data you provide
 - Data Tests test data integrity and quality on the actual data
- 
-

DATA 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
- You can import tests from dbt packages (will discuss later)



GUIDED EXERCISE

TEST `dim_hosts_cleansed`

Create a generic data 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)