

Airbnb DBT Project

By: Adrian Chavez-Loya

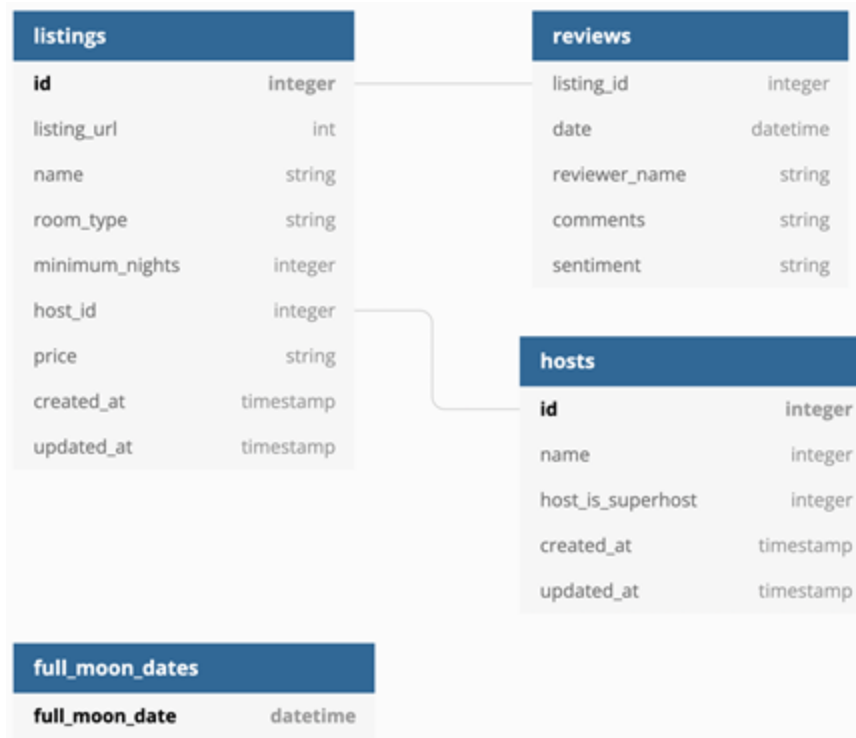
Source data:

Comes from official Airbnb open-source data:

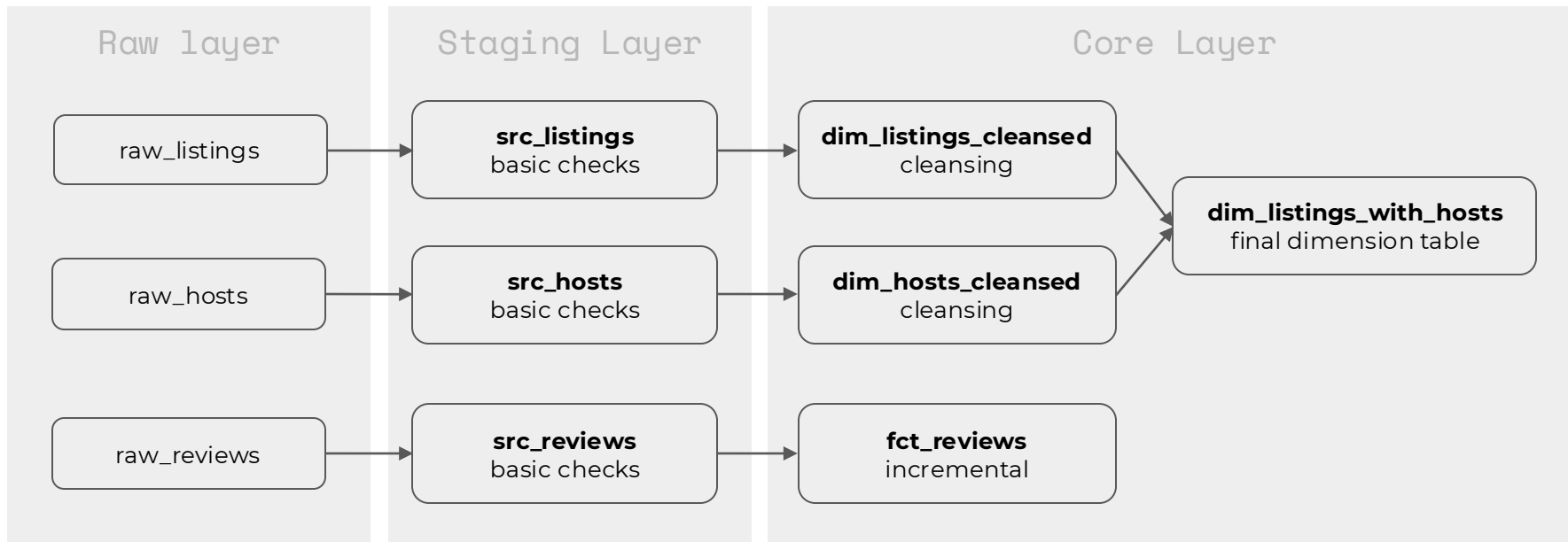
<https://insideairbnb.com/get-the-data/>

The historical data we used is for listing and host details from Berlin from the years 2015-2021. This data was stored in AWS.

Input source data we took from AWS S3



Our Data Stream in Snowflake we will implement



DB/Schema Creation

The screenshot displays a SQL IDE interface with three tabs: `airbnb_raw_setup_and_impo...`, `airbnb_user_role_creation.sql`, and `operations.sql`. The active tab is `airbnb_raw_setup_and_impo...`, showing a SQL script for database setup and data import. The script includes comments and SQL commands for creating warehouses, databases, schemas, and tables, as well as copying data from S3. The right pane shows a preview of the SQL code for creating and copying data into `raw_reviews` and `raw_hosts` tables.

```
1  -- Set up the defaults
2  CREATE WAREHOUSE IF NOT EXISTS COMPUTE_WH;
3  USE WAREHOUSE COMPUTE_WH;
4
5  DROP DATABASE IF EXISTS AIRBNB CASCADE;
6
7  CREATE DATABASE AIRBNB;
8
9  CREATE SCHEMA IF NOT EXISTS AIRBNB.RAW;
10 CREATE SCHEMA IF NOT EXISTS AIRBNB.DEV;
11
12 USE DATABASE AIRBNB;
13 USE SCHEMA RAW;
14
15 -- Create our three tables and import the data from S3
16 CREATE OR REPLACE TABLE raw_listings
17     (id integer,
18      listing_url string,
19      name string,
20      room_type string,
21      minimum_nights integer,
22      host_id integer,
23      price string,
24      created_at datetime,
25      updated_at datetime);
26
27 COPY INTO raw_listings (id,
28                        listing_url,
29                        name,
30                        room_type,
```

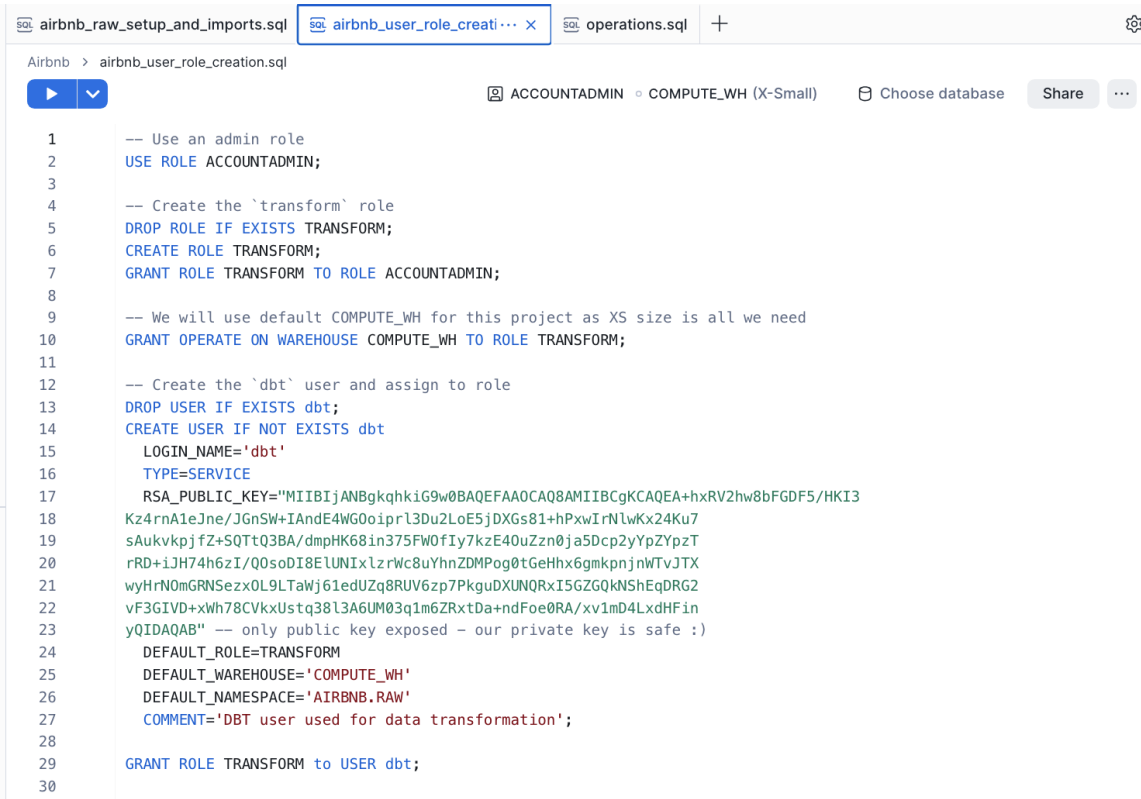
```
CREATE OR REPLACE TABLE raw_reviews
(
  listing_id integer,
  date datetime,
  reviewer_name string,
  comments string,
  sentiment string);

COPY INTO raw_reviews (listing_id, date, reviewer_name, comments, sentiment)
  from 's3://dbt-datasets/reviews.csv'
  FILE_FORMAT = (type = 'CSV' skip_header = 1
  FIELD_OPTIONALLY_ENCLOSED_BY = '');

CREATE OR REPLACE TABLE raw_hosts
(
  id integer,
  name string,
  is_superhost string,
  created_at datetime,
  updated_at datetime);

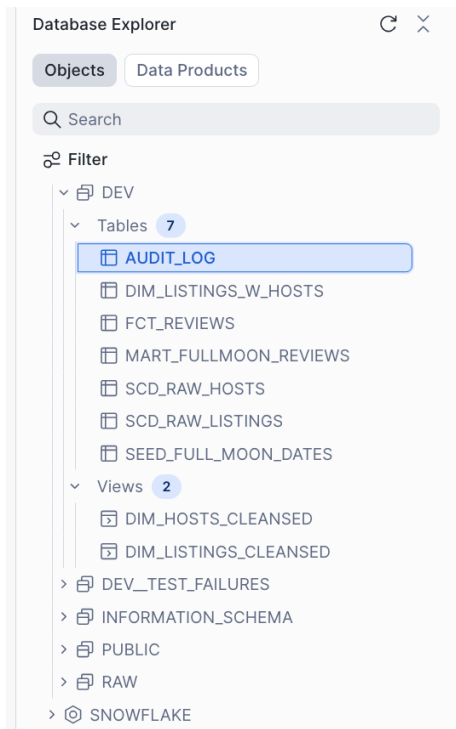
COPY INTO raw_hosts (id, name, is_superhost, created_at, updated_at)
  from 's3://dbt-datasets/hosts.csv'
  FILE_FORMAT = (type = 'CSV' skip_header = 1
  FIELD_OPTIONALLY_ENCLOSED_BY = '');
```

Created roles/warehouse, and dbt access user (private key is safe don't worry)



The screenshot shows a SQL editor interface with a tab titled "airbnb_user_role_creation.sql". The script contains SQL commands to create a role, a warehouse, and a user. The user is created with a public key and a comment indicating it is for dbt access. The script is as follows:

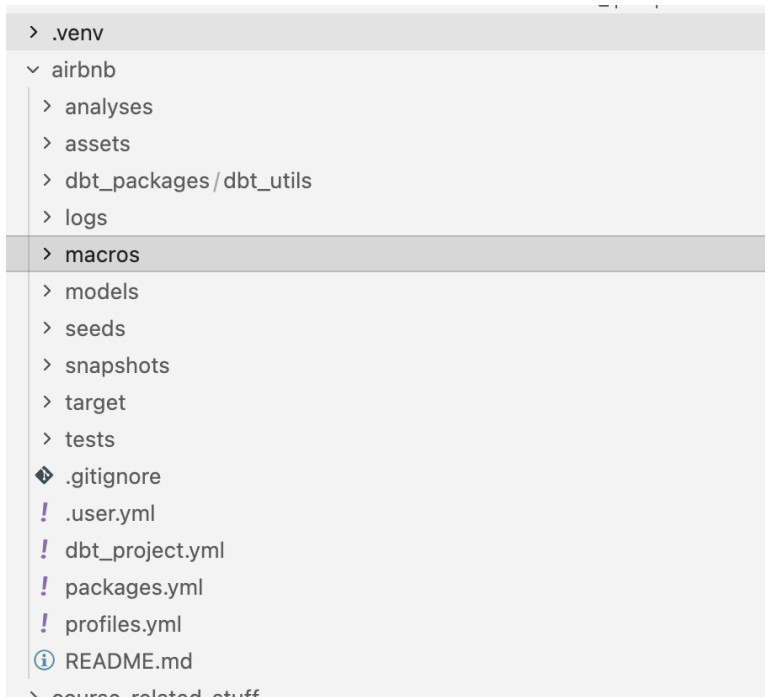
```
1  -- Use an admin role
2  USE ROLE ACCOUNTADMIN;
3
4  -- Create the `transform` role
5  DROP ROLE IF EXISTS TRANSFORM;
6  CREATE ROLE TRANSFORM;
7  GRANT ROLE TRANSFORM TO ROLE ACCOUNTADMIN;
8
9  -- We will use default COMPUTE_WH for this project as XS size is all we need
10 GRANT OPERATE ON WAREHOUSE COMPUTE_WH TO ROLE TRANSFORM;
11
12 -- Create the `dbt` user and assign to role
13 DROP USER IF EXISTS dbt;
14 CREATE USER IF NOT EXISTS dbt
15     LOGIN_NAME='dbt'
16     TYPE=SERVICE
17     RSA_PUBLIC_KEY="MIIBIjANBgkqhkiG9w0BAQEFAAACQ8AMIIBCgKCAQEAhxRV2hw8bFGDF5/HKI3
18 Kz4rnA1eJne/JGnSW+IAndE4WG0oipr13Du2LoE5jDXGs81+hPxwIrNlwKx24Ku7
19 sAukvkpfZ+SQTtQ3BA/dmpHK68in375FW0fIy7kzE40uZzn0ja5Dcp2yYpZYpZT
20 rRD+iJH74h6zI/Q0soDI8ELUNIXLzrWc8uYhnZMPog0tGeHx6gmkpjnnWTvJTX
21 wyHrN0mGRNSezzx0L9LTaWj61edUZq8RUV6zp7PkguDXUNQRxI5GZGQkNShEqDRG2
22 vF3GIVD+xWh78CVkxUstq38L3A6UM03q1m6ZRxtDa+ndFoe0RA/xv1mD4LxdHFin
23 yQIDAQAB" -- only public key exposed - our private key is safe :)
24     DEFAULT_ROLE=TRANSFORM
25     DEFAULT_WAREHOUSE='COMPUTE_WH'
26     DEFAULT_NAMESPACE='AIRBNB.RAW'
27     COMMENT='DBT user used for data transformation';
28
29 GRANT ROLE TRANSFORM to USER dbt;
30
```



Final view of our data warehouse in the end. Includes audit_log which provides logs of model runs (made in dbt)

MODEL_NAME		RUN_TIMESTAMP	
AIRBNB.DEV.dim_hosts_cleansed	20.0%		
AIRBNB.DEV.dim_listings_cleansed	20.0%		
+3 more		11/12/2025	11/12/2025
1	AIRBNB.DEV.dim_hosts_cleansed	2025-11-11 23:40:20.785	
2	AIRBNB.DEV.dim_listings_cleansed	2025-11-11 23:40:21.416	
3	AIRBNB.DEV.fct_reviews	2025-11-11 23:40:22.865	
4	AIRBNB.DEV.dim_listings_w_hosts	2025-11-11 23:40:24.234	
5	AIRBNB.DEV.mart_fullmoon_reviews	2025-11-11 23:40:26.508	
6	AIRBNB.DEV.dim_hosts_cleansed	2025-11-12 03:18:32.195	
7	AIRBNB.DEV.dim_listings_cleansed	2025-11-12 03:18:32.969	
8	AIRBNB.DEV.fct_reviews	2025-11-12 03:18:34.564	
9	AIRBNB.DEV.dim_listings_w_hosts	2025-11-12 03:18:44.620	
10	AIRBNB.DEV.mart_fullmoon_reviews	2025-11-12 03:18:48.717	

We initialized and isolated dependencies with a virtual environment. On the left is the final view of the dbt project.



```
(.venv) adrianchavezloya@Adrian-MacBook dbt bootcamp % dbt init --skip-profile-setup airbnb
14:36:01 Running with dbt=1.10.13
14:36:01
four new dbt project "airbnb" was created!

For more information on how to configure the profiles.yml file,
please consult the dbt documentation here:

https://docs.getdbt.com/docs/configure-your-profile

One more thing:


Need help? Don't hesitate to reach out to us via GitHub issues or on Slack:

https://community.getdbt.com/

Happy modeling!

(.venv) adrianchavezloya@Adrian-MacBook dbt bootcamp %
```


We also generated a public/private key for secure access to Snowflake (user/password method is going to be discontinued in Snowflake). You saw the public key earlier.

✓  security-keys

 rsa_key.p8

 rsa_key.pub

We have our profiles.yml connecting dbt to our data warehouse

! profiles.yml ×

airbnb > ! profiles.yml > {} airbnb

```
1  Airbnb:
2    outputs:
3      dev:
4        type: snowflake
5        account: phbsrbo-kw29860
6        user: dbt
7
8        role: TRANSFORM
9        private_key: "super encrypted key don't use me if you aren't authorize! "
10       private_key_passphrase: q
11
12       database: AIRBNB
13       schema: DEV
14       threads: 1
15       warehouse: COMPUTE_WH
16 target: dev
17
```

dbt_project.yml defines our targets and configurations for our folders. In this, we practiced usings grants for roles, hooks, and set our types of default materialization settings we wanted for our data layers.

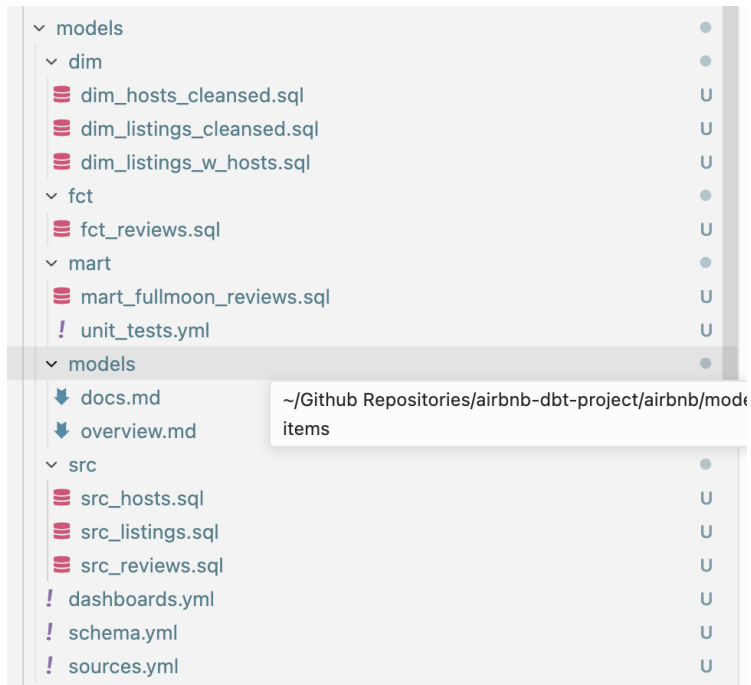
```
2 # Name your project! Project names should contain only lowercase characters
3 # and underscores. A good package name should reflect your organization's
4 # name or the intended use of these models
5 name: 'airbnb'
6 version: '1.0.0'
7
8 # This setting configures which "profile" dbt uses for this project.
9 profile: 'airbnb'
10
11 # These configurations specify where dbt should look for different types of files.
12 # The `model-path` dbt Project tes that models in this project can be
13 # found in the "models" directory. You can have as many paths as you like, but you'll probably won't need to change these!
14 model-paths: ["models"]
15 analysis-paths: ["analyses"]
16 test-paths: ["tests"]
17 seed-paths: ["seeds"]
18 macro-paths: ["macros"]
19 snapshot-paths: ["snapshots"]
20 asset-paths: ["assets"]
21
22 clean-targets: # directories to be removed by `dbt clean`
23   - "target"
24   - "dbt_packages"
```

```
on-run-start:
  - "CREATE TABLE IF NOT EXISTS {{ target.schema }}.audit_log (
      model_name STRING,
      run_timestamp TIMESTAMP
    )"

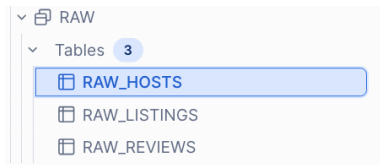
models:
  airbnb:
    +grants:
      select: ["transform", "reporter"] # grants to our roles
    +post-hook:
      - "INSERT INTO {{ target.schema }}.audit_log VALUES ('{{ this }}', CURRENT_TIMESTAMP)"
    +materialized: view
    dim:
      +materialized: table
    src:
      +materialized: ephemeral ## Source views no longer needed

data_tests:
  +store_failures: true
  +schema: _test_failures
```

Models folder has all our models. Look at them all in detail in the github repository



Our source layer
(staging) tables!
I just adjusted
some columns from
our raw layer.



Raw tables

```
! package-lock.yml U  src_hosts.sql U X
airbnb > models > src > src_hosts.sql
1  --all sources defaulting to views
2  WITH raw_hosts AS (
3      SELECT
4          *
5      FROM
6          {{ source('airbnb', 'hosts') }}
7  )
8  SELECT
9      id AS host_id,
10     NAME AS host_name,
11     is_superhost,
12     created_at,
13     updated_at
14 FROM
15     raw_hosts
16
```

```
! package-lock.yml U  src_listings.sql U X
airbnb > models > src > src_listings.sql
1  WITH raw_listings AS (
2      SELECT
3          *
4      FROM
5          {{ source('airbnb', 'listings') }}
6  )
7  SELECT
8      id AS listing_id,
9      name AS listing_name,
10     listing_url,
11     room_type,
12     minimum_nights,
13     host_id,
14     price AS price_str,
15     created_at,
16     updated_at
17 FROM
18     raw_listings
19
```

```
! package-lock.yml U  src_reviews.sql U X
airbnb > models > src > src_reviews.sql
1  WITH raw_reviews AS (
2      SELECT
3          *
4      FROM
5          {{ source('airbnb', 'reviews') }}
6  )
7  SELECT
8      listing_id,
9      date AS review_date,
10     reviewer_name,
11     comments AS review_text,
12     sentiment AS review_sentiment
13 FROM
14     raw_reviews
15
```

This is a mart table we created for fun to view reviews to see if people are more moody and perhaps give worse reviews.

! package-lock.yml U

mart_fullmoon_reviews.sql U X



airbnb > models > mart > mart_fullmoon_reviews.sql

```
1  {{ config(
2    |   materialized = 'table',
3    | ) }}
4
5  WITH fct_reviews AS (
6    |   SELECT * FROM {{ ref('fct_reviews') }}
7    | ),
8  full_moon_dates AS (
9    |   SELECT * FROM {{ ref('seed_full_moon_dates') }}
10   | )
11
12  SELECT
13    r.*,
14    CASE
15      | WHEN fm.full_moon_date IS NULL THEN 'not full moon'
16      | ELSE 'full moon'
17    | END AS is_full_moon
18  FROM
19    fct_reviews
20    r
21  LEFT JOIN
22    full_moon_dates
23    fm
24  ON (TO_DATE(r.review_date) = DATEADD(DAY, 1, fm.full_moon_date))
```

```

dim_hosts_cleansed.sql U X
airbnb > models > dim > dim_hosts_cleansed.sql
1 {{
2   config(
3     materialized = 'view'
4   )
5 }}
6
7
8 WITH src_hosts AS (
9   SELECT
10    *
11   FROM
12    {{ ref('src_hosts') }}
13 )
14
15 SELECT
16   host_id,
17   COALESCE(host_name, 'Anonymous')::TEXT as host_name,
18   is_superhost,
19   created_at,
20   updated_at
21 FROM
22   src_hosts

```

```

dim_listings_cleansed.sql U X
airbnb > models > dim > dim_listings_cleansed.sql
1 {{
2   config(
3     materialized = 'view'
4   )
5 }}
6
7
8 WITH src_listings as (
9   SELECT * FROM {{ ref('src_listings') }}
10 )
11
12
13
14 SELECT
15   listing_id,
16   listing_name,
17   room_type,
18   CASE
19     WHEN minimum_nights = 0 THEN 1 -- min zero nights means one night actually
20     ELSE minimum_nights
21   END AS minimum_nights,
22   host_id,
23   REPLACE(price_str, '$', '') :: NUMBER(10,2) AS price, -- remove '$' and cast
24   created_at,
25   updated_at
26 FROM
27   src_listings

```

Dimension tables for listings and hosts are made and combined for better analysis. We mainly use `dim_listings_w_hosts` for our dashboard later.

```

dim_listings_w_hosts.sql U X
airbnb > models > dim > dim_listings_w_hosts.sql
1 -- Will be initialized as a table
2 WITH
3 l AS (
4   SELECT
5     *
6   FROM
7     {{ ref('dim_listings_cleansed') }}
8 ),
9 h AS (
10   SELECT *
11   FROM {{ ref('dim_hosts_cleansed') }}
12 )
13
14 SELECT
15   l.listing_id,
16   l.listing_name,
17   l.room_type,
18   l.minimum_nights,
19   l.price,
20   l.host_id,
21   h.host_name,
22   h.is_superhost as host_is_superhost,
23   l.created_at,
24   GREATEST(l.updated_at, h.updated_at) as updated_at
25 FROM l
26 LEFT JOIN h ON (h.host_id = l.host_id)

```

We created a fact table for reviews but doesn't include the typical metrics a typical fact table would

```
fct_reviews.sql U X
airbnb > models > fct > fct_reviews.sql

1 {{
2   config(
3     materialized = 'incremental',
4     on_schema_change='fail'
5   )
6 }}
7 WITH src_reviews AS (
8   SELECT * FROM {{ ref('src_reviews') }}
9 )
10 SELECT
11   {{ dbt_utils.generate_surrogate_key(['listing_id', 'review_date', 'reviewer_name', 'review_text']) }}
12   AS review_id,
13   *
14 FROM src_reviews
15 WHERE review_text is not null
16 {% if is_incremental() %}
17   AND review_date > (select max(review_date) from {{ this }})
18 {% endif %}
```


We created easy markdown files to create our UI menu which includes our input data and some text.

localhost:8888/#/overview

AMAZON JOBS PYTHON INTERV... regex101: build, te... Books Google SQL PRACTICE Indeed Job Search Snowflake login

dbt Search for models...

Overview

Project Database Group

Sources

- airbnb

Exposures

- Dashboard

Projects

- airbnb
- dbt_utils

Airbnb pipeline

Hey, welcome to our Airbnb pipeline documentation!

Here is the schema of our input data:

listings	
id	integer
listing_url	int
name	string
room_type	string
minimum_nights	integer
host_id	integer
price	string
created_at	timestamp
updated_at	timestamp

reviews	
listing_id	integer
date	datetime
reviewer_name	string
comments	string
sentiment	string

hosts	
id	integer
name	integer
host_is_superhost	integer
created_at	timestamp
updated_at	timestamp

full_moon_dates	
full_moon_date	datetime

profiles.yml dashboards.yml

airbnb > models > ! dashboards.yml > [] exposures > { } 0 > { } owner > email

```
1
2 exposures:
3   - name: executive_dashboard
4     label: Executive Dashboard
5     type: dashboard
6     maturity: medium
7     url: https://9b99d5a1.us1a.app.preset.io/superset/dashboard/9/?native_filters_key=-7KwrWS06KV
8     description: Executive Dashboard about Airbnb listings and hosts
9
10    depends_on:
11      - ref('dim_listings_w_hosts')
12      - ref('mart_fullmoon_reviews')
13
14    owner:
15      name: Adrian Chavez-Loya
16      email: adrianchavezloya@gmail.com
17
```

We also have a direct link to our simple dashboard for this project



Search for models...

Overview

Project Database Group

Sources

airbnb

Exposures

Dashboard

Executive Dashboard

Projects

airbnb

dbt_utils

Executive Dashboard exposure

[View this exposure](#)

[Details](#) [Description](#) [Depends On](#)

Details

TAGS	PACKAGE	CONTRACT	MATURITY	OWNER	EXPOSURE NAME
untagged	airbnb	Not Enforced	medium	Adrian Chavez-Loya <adrianchavezloya@gmail.com>	executive_dashboard

Description

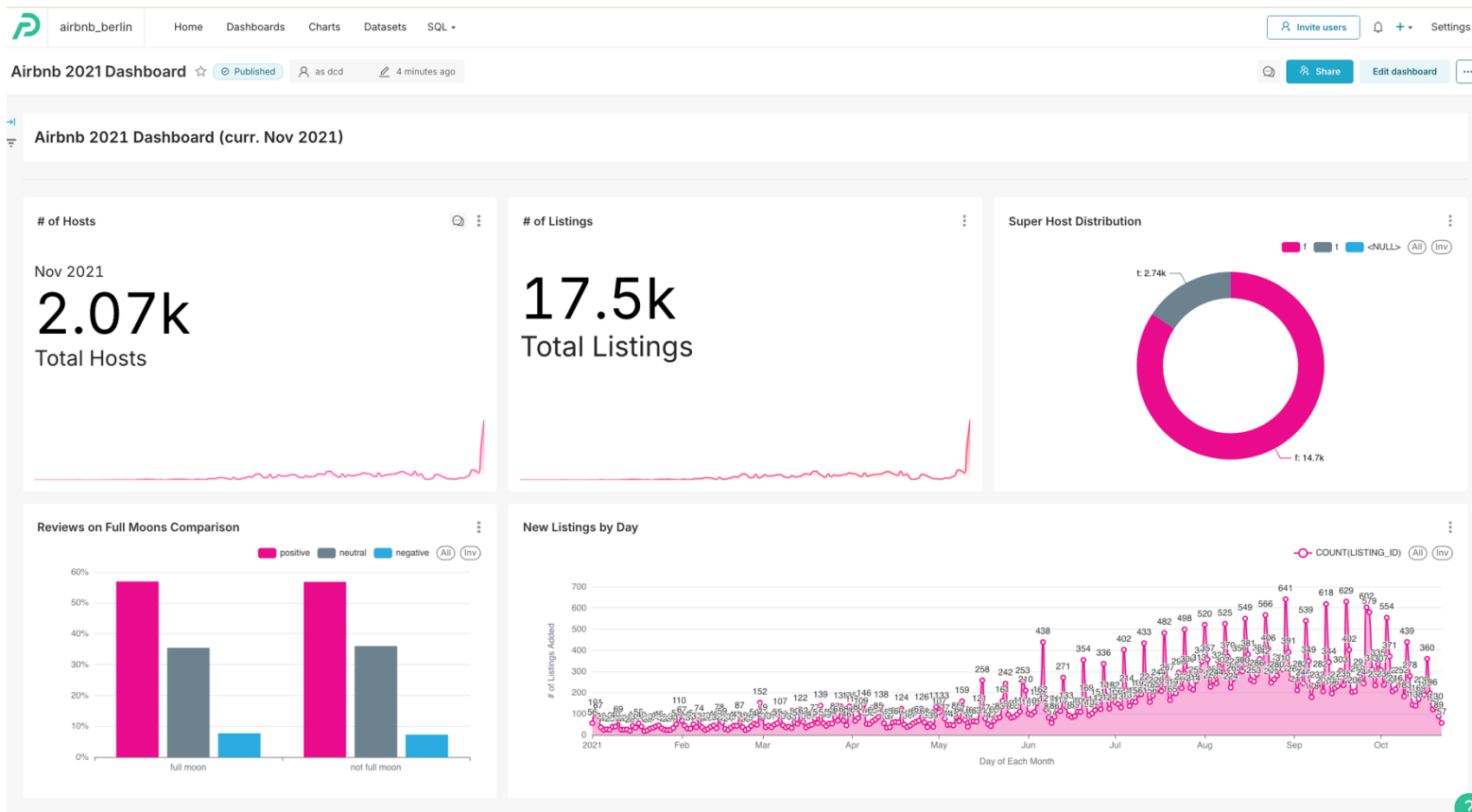
Executive Dashboard about Airbnb listings and hosts

Depends On

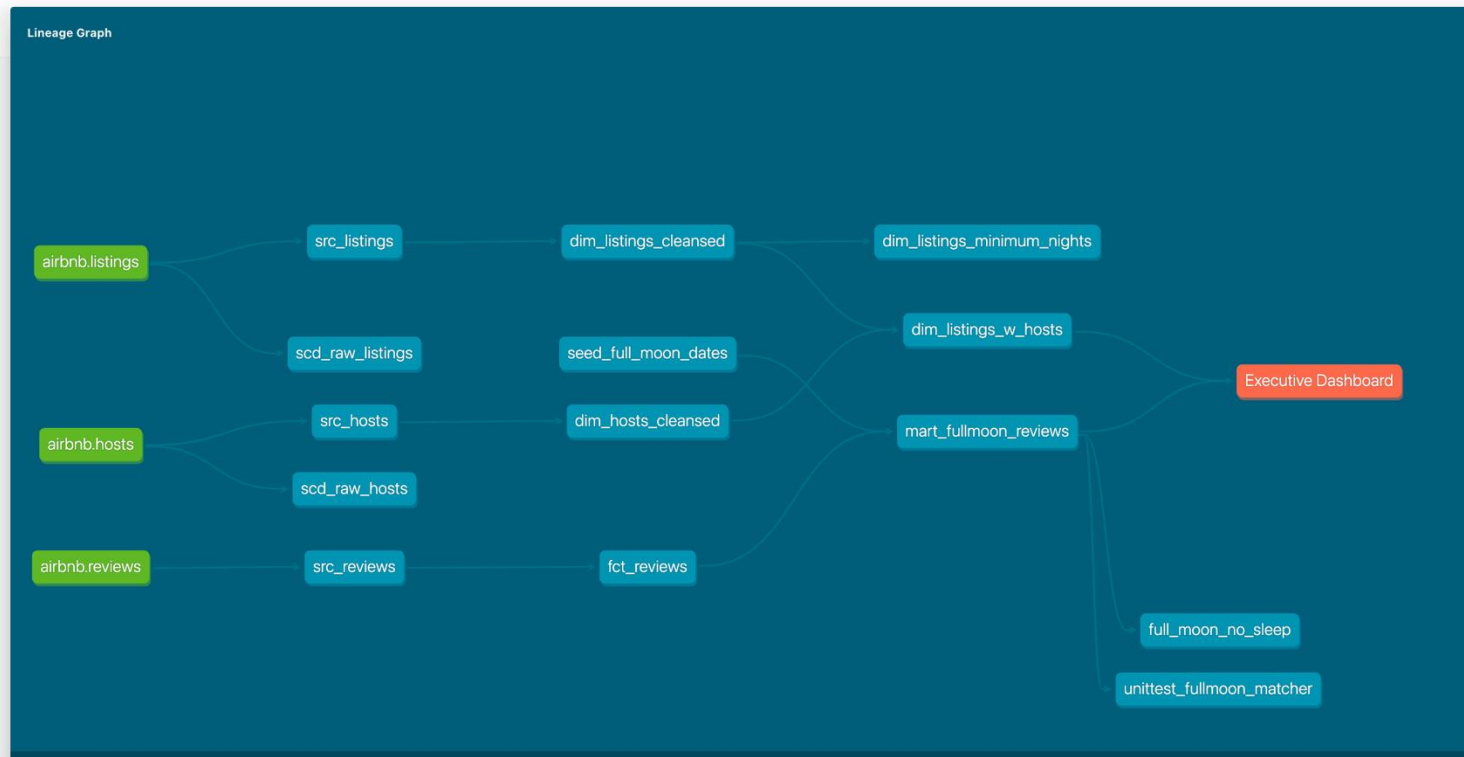
Models

[dim_listings_w_hosts](#)
[mart_fullmoon_reviews](#)

Our dashboard we connected to Snowflake and created with Preset (Side Note: turns out people are unaffected by full moons when reviewing)



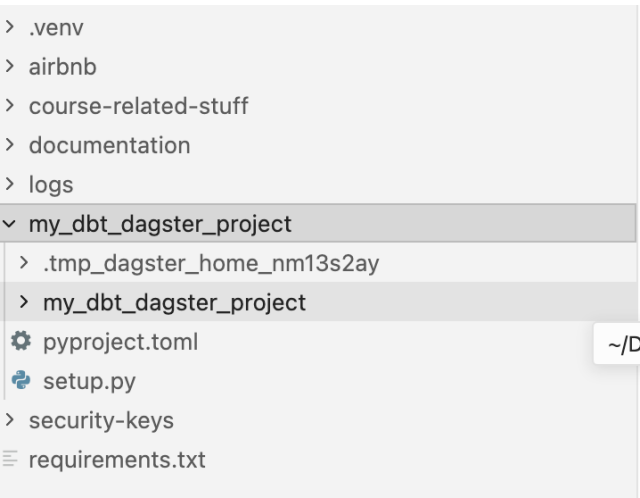
Here is our lineage graph made with dbt



We did a number of other exercises which can be found in the repository. This includes:

- Implementing Jinja2 macros
- Created snapshots
- Made sample validation tests
- Experimented with dagster orchestration

I will show you some screenshots of my initialized UI interface for dagster which I used to also play around with its scheduler and see how it can be used for orchestration much like Airflow. We were able to scaffold dagster into our dbt project.



Here is asset lineage within Dagster


The image displays two side-by-side screenshots of the Dagster web interface, specifically the 'Global Asset Lineage' section. The interface is dark-themed with a navigation bar at the top containing links for Overview, Runs, Assets, Jobs, Automation, and Deployment. A search bar and a 'Reload definitions' button are located in the top right corner of the main content area.




On the left screenshot, the 'Jump to...' dropdown is set to 'dim_hosts_cleansed'. The asset lineage graph shows a vertical stack of assets on the left, with 'dim_hosts_cleansed' at the top. Arrows indicate dependencies from 'dim_hosts_cleansed' to 'dim_listings_w_hosts', 'dim_listings_cleansed', 'fct_reviews', 'scd_raw_hosts', 'scd_raw_listings', and 'seed_full_moon_dates'. Each asset card displays its latest event, asset checks (e.g., '0 / 4 Passed'), and automation status.

On the right screenshot, the 'Jump to...' dropdown is set to 'dim_listings_w_hosts'. The asset lineage graph shows a vertical stack of assets on the right, with 'dim_listings_w_hosts' at the top. Arrows indicate dependencies from 'dim_listings_w_hosts' to 'dim_hosts_cleansed', 'dim_listings_cleansed', 'fct_reviews', 'scd_raw_hosts', 'scd_raw_listings', and 'seed_full_moon_dates'. Each asset card displays its latest event, asset checks (e.g., '4 / 4 Passed'), and automation status.

Both screenshots show a list of assets on the left side of the graph, including 'dim_hosts_cleansed', 'dim_listings_cleansed', 'dim_listings_w_hosts', 'fct_reviews', 'mart_fullmoon_reviews', 'scd_raw_hosts', 'scd_raw_listings', and 'seed_full_moon_dates'. The right screenshot also includes a 'Materialize all' button in the top right corner.




If we wanted to continually run our dbt models on schedule we could with Dagster!

 Overview Runs Assets Jobs **Automation** Deployment



Automation

Actions ▾

<input type="checkbox"/>	Name	Type	Target	Last tick	Last run
my_dbt_dagster_project ▾					
<input type="checkbox"/>	<input checked="" type="checkbox"/> materialize_dbt_models_schedule	 At 12:00 AM UTC Next tick: Nov 13, 12:00 AM UTC 	 ma	None	None

Ultimately through this project, we were able to learn some of the following:

- How to implement data models and perform ELT for Snowflake using dbt
- How to tune and tweak different settings to implement different table/view types at different stages
- Learned about Jira macros, seeds, snapshots, hooks exposures, and so much more!
- Practiced using a creating a data warehouse that was usable for analysis