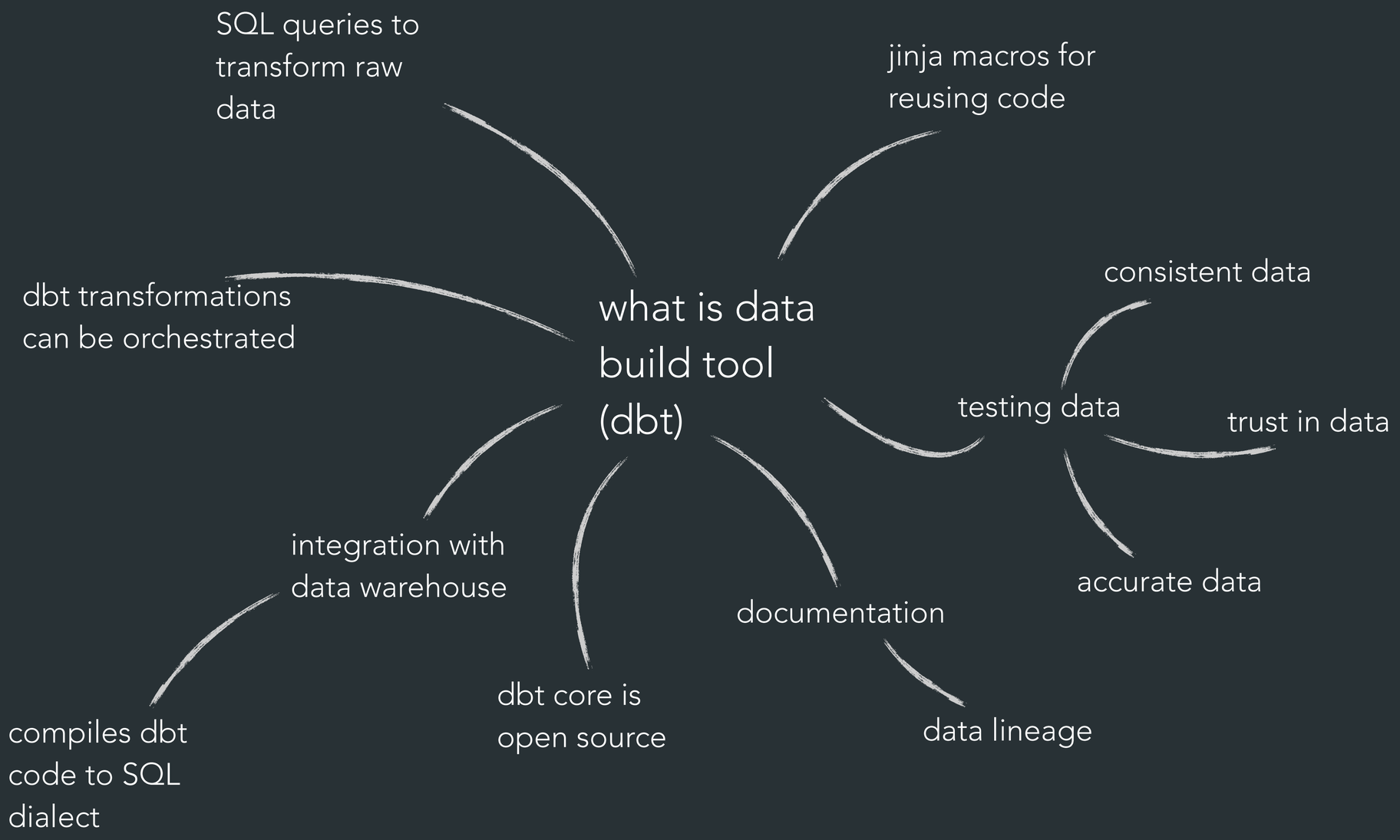
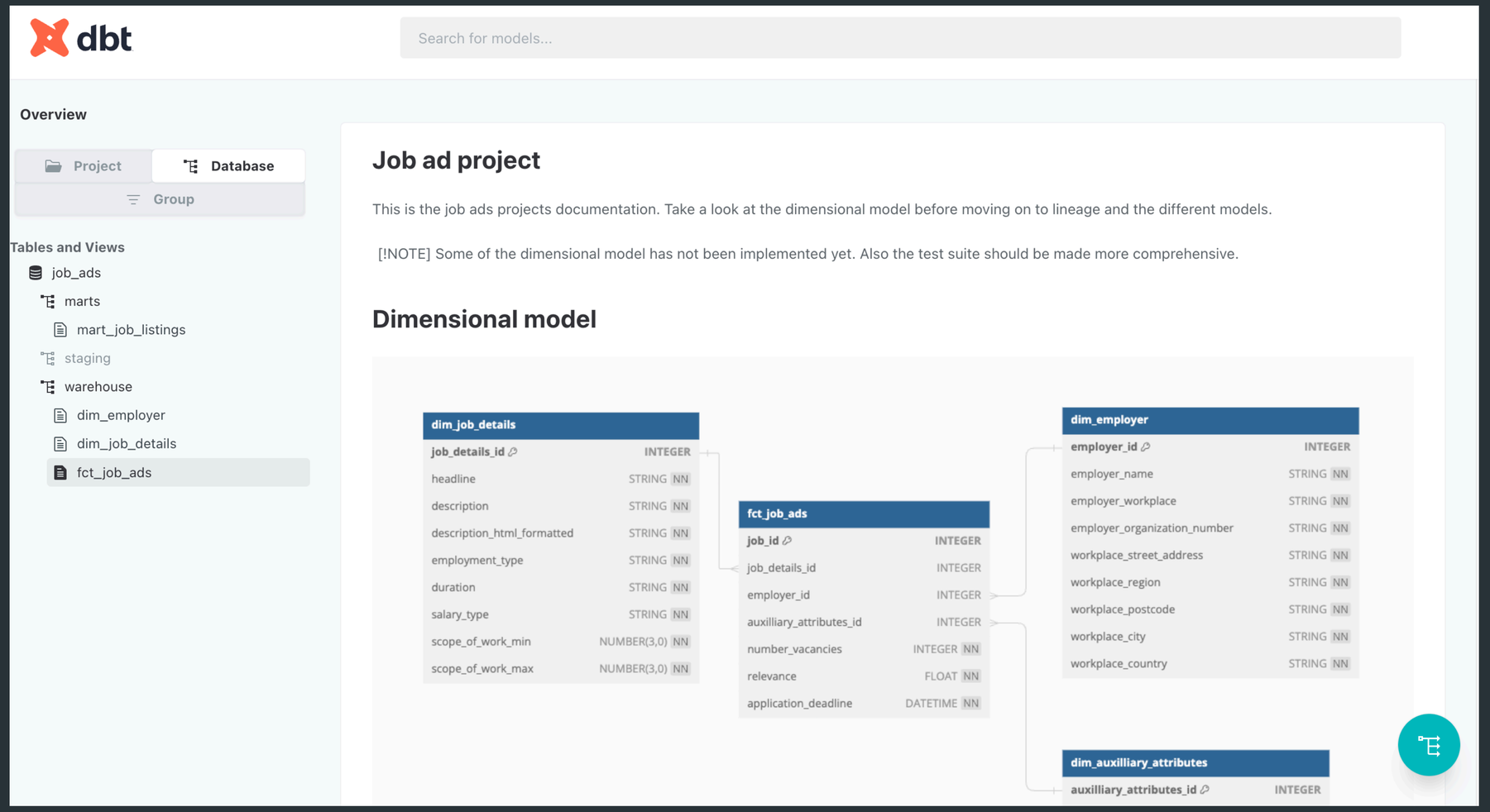
kokchun giang

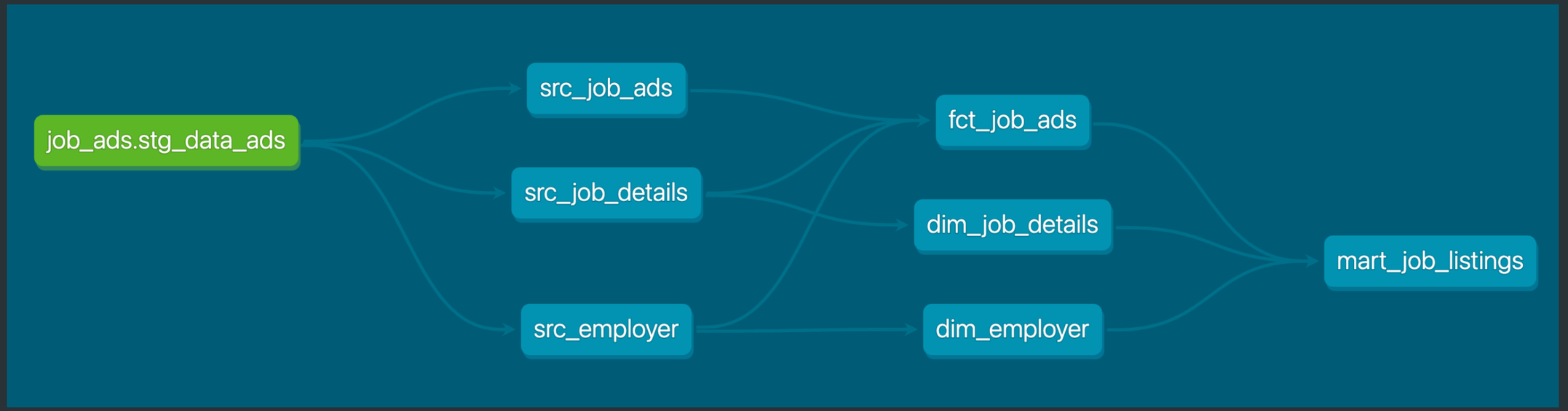
using **dbt** to transform and refine the data into


SQL queries to transform raw data
jinja macros for reusing code
dbt transformations can be orchestrated
what is data build tool (dbt)
consistent data
testing data
trust in data
integration with data warehouse
accurate data
documentation
compiles dbt code to SQL dialect
dbt core is open source
data lineage

web-based autogenerated **documentation** to understand the data models



data lineage to visualize how data flows through various transformations



source code is **compiled** to snowflake code

jinja macros

Source Compiled

```sql
WITH ja AS (SELECT * FROM {{ ref('src_job_ads') }}),

jd AS (SELECT * FROM {{ ref('src_job_details') }}),

e AS (SELECT * FROM {{ ref('src_employer') }})

SELECT
    {{dbt_utils.generate_surrogate_key(['jd.id', 'jd.headline'])}} AS job_details_key,
    {{dbt_utils.generate_surrogate_key(['e.id', 'e.employer_name'])}} AS employer_key,
    {# TODO: key to junk dimension #}
    relevance, -- if null, aggregation functions can handle it properly
    coalesce(vacancies, 1) as vacancies,
    application_deadline
    -- for verifying FK relationships work
    {# jd.id,
    e.workplace_city,
    e.employer_name,
    jd.description #}
FROM
    ja
LEFT JOIN
    jd ON ja.id = jd.id
LEFT JOIN
    e ON ja.id = e.id
```

dbt code

Source Compiled

```sql
WITH  __dbt__cte__src_job_ads as (



WITH stg_job_ads AS (SELECT * FROM job_ads.staging.data_field_job_ads
)

SELECT
    id,
    headline,
    number_of_vacancies AS vacancies,
    relevance,
    application_deadline
FROM stg_job_ads ORDER BY application_deadline ASC
),  __dbt__cte__src_job_details as (


WITH stg_job_ads AS (SELECT * FROM job_ads.staging.data_field_job_ads
)

SELECT
    id,
    headline,
    description__text AS "DESCRIPTION",
    description__text_formatted AS description_html_formatted
```

compiled snowflake SQL

file structure of a dbt project