**## Building Tests**

* Running test cases on the existing models
* To validate the models and working as expected
* From start, we can set TEST case from source, models, dim\_tables
* Continues on DAG ordering
* Using ‘yml’ file to run the test on specific models

# **Contents**:

* Build a TEST case with ‘yaml’ file
* Adding other Test cases:
* Applying TEST directly to source data
* Test foreign key relationships

# Two types of data tests:

* Singular
* Generic

# Generic Data Tests in-built: most common types

* Unique (mostly use on primary key to prevent fanning-out)
* Not\_null (mostly use on primary key to prevent fanning-out)
* Relationships
* Accepted\_values

# Build a TEST case with ‘yaml’ file

* Specify the model ‘NAME’ and ‘COLUMN’ name and test cases
* Generic test case if the column for `unique` and `not\_null`

version: 2

models:

- name: stg\_snow\_sales

columns:

- name: o\_custkey

tests:

- unique

- not\_null

* Run the command to test cases:

→

dbt test - - select stg\_snow\_sales

## Adding other Test cases:

* Check for null values and check if the values exists

version: 2

models:

- name: stg\_orders

columns:

- name: o\_custkey

tests:

- unique

- not\_null

# second test for accepted values on CUSTOMER's segments

- name: stg\_customers

columns:

- name: c\_mktsegment

tests:

- not\_null

- name: c\_mktsegment

tests:

- accepted\_values:

values:

[

"HOUSEHOLD",

"BUILDING",

"AUTOMOBILE",

"MACHINERY",

"FURNITURE",

]

# SINGULAR Test

* Adding singular test case by SQL queries to test conditions
* Query to generate results if ‘HAVING(total\_amount) <0 ‘
* Running the test case by specifying the SQL file
* EG:
  + Dbt test - - select <assert-positive-price.sql>

# Applying TEST directly to source data

* Using config ‘YML’ file to test the PRIMARY KEYs
* Check if the primary key is NOT\_NULL and UNIQUE
* EG: below we will check on table: ORDERS for column ‘o\_orderkey’. To check if the values are UNIQUE and NOT\_NULL

# Source configuration

version: 2

sources:

- name: snow\_sales

database: dbt\_trial

schema: dbo\_dbt

freshness: ## on schema level to check the freshness

warn\_after: {count: 6, period: hour}

error\_after: {count: 24, period: hour}

loaded\_at\_field: \_etl\_loaded\_date

tables:

- name: customers

freshness: null

- name: orders

columns:

- name: o\_orderkey

tests:

- unique

- not\_null

freshness: ## strict configurations

warn\_after: {count: 6, period: hour}

error\_after: {count: 24, period: hour}

loaded\_at\_field: \_etl\_loaded\_date

- name: lineitems

* Run the cmd:

→ `dbt test –select <source\_config yml file> `