**Write dry(don’t repeat yourself) sql code using ref()**

I have used the dry script in **cutomers.sql, orders.sql, customer\_order.sql** fles

**DBT seeds**

I have used a sample data in the csv **cities.csv** file and its configuration file is **seeds.yaml**

And we can use that data as **SELECT \* FROM {{ ref(cities) }** in the sql scripts.

**DBT Tests**

Schema tests are simple, pre-defined tests that you define in your YAML files. They check specific properties directly on your models, sources, and columns.

**Common Schema Tests**

* **unique**: Checks if values in a column are unique.
* **not\_null**: Ensures there are no NULL values in a column.
* **accepted\_values**: Checks if values in a column match an expected set of values.
* **relationships**: Validates referential integrity by checking if values in one column exist in another model’s column ( foreign key constraint).

We can refer to **schema.yaml** file.

**DBT Documentation**

dbt documentation provides an accessible way to document, visualize, and explore your data models, sources, tests, and transformations within a dbt project. This documentation can be easily generated and viewed through a web-based interface, helping teams understand the lineage, context, and structure of their data.

1: Add Descriptions to Your Models, Sources, and Columns in a .yaml file

2: Generate dbt Documentation  
 Command: dbt docs generate

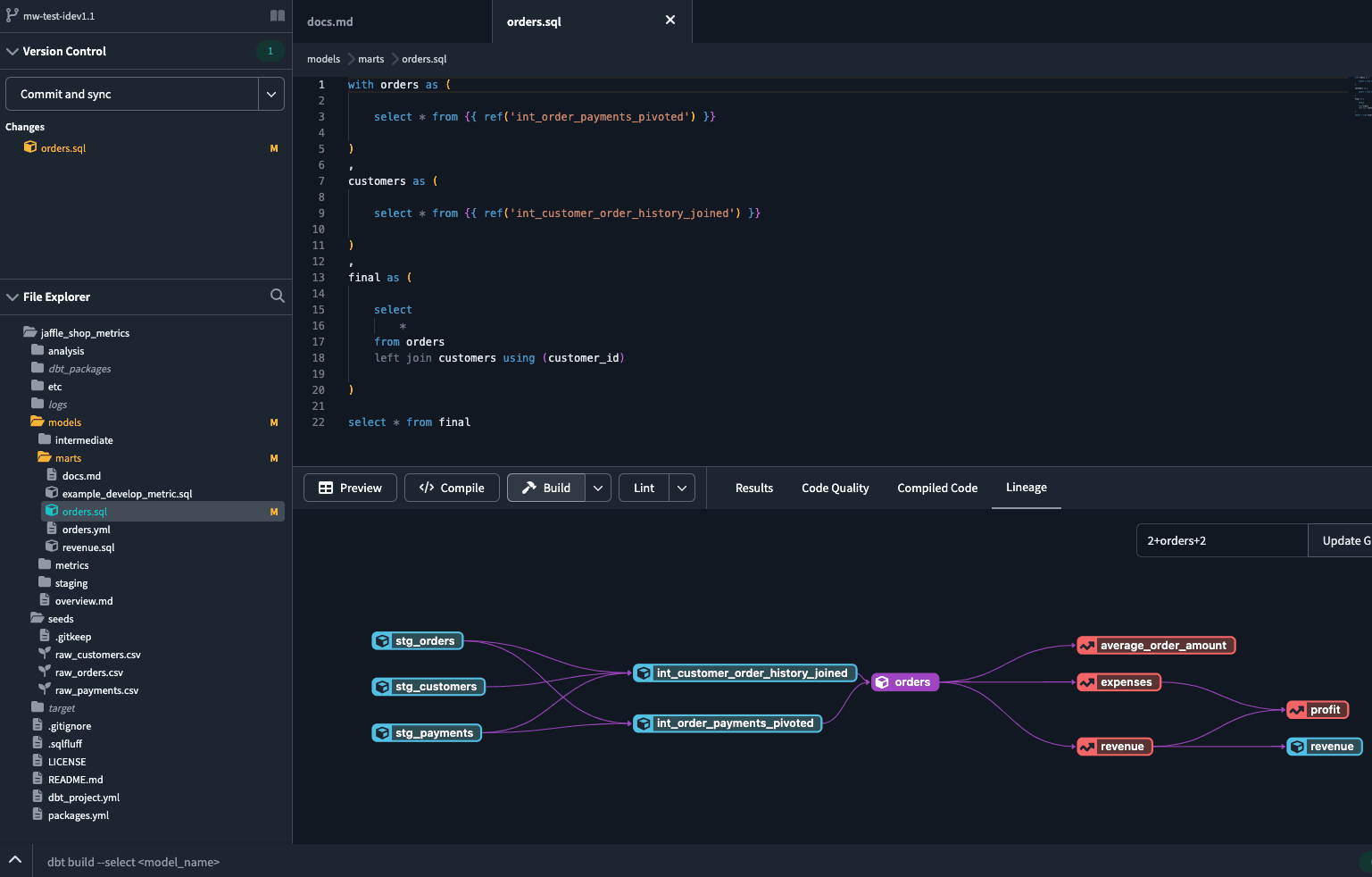
3: View the Documentation

Command: dbt docs serve

This command opens a web-based interface (usually at http://localhost:8000) where you can explore the documentation, including descriptions, tests, and a lineage graph.

**Data Lineage**

Data lineage provides a holistic view of how data moves through an organization, where it’s transformed and consumed. Overall, data lineage is a fundamental concept to understand in the practice of analytics engineering and modern data work



**Version Control**

Version control in dbt is essential for managing changes to your data models, transformations, and configurations over time, ensuring collaboration, traceability, and stability. Typically, dbt projects use Git for version control, allowing multiple team members to work on the same project, track changes, and roll back to previous versions if needed

**Git Workflow for a dbt Project**

Clone the Repository

Create a New Branch

Make Your Changes in dbt

Test Your Changes Locally

Commit Your Changes  
All these we can achieve using the git commands.