1. Create database

USE ROLE DBT\_EXECUTOR\_ROLE;

CREATE DATABASE PORTFOLIO\_TRACKING

COMMENT = 'DB for the portfolio tracking project';

1. Create Schema

USE ROLE DBT\_EXECUTOR\_ROLE;

CREATE SCHEMA PORTFOLIO\_TRACKING.SOURCE\_DATA;

1. Create Table

CREATE OR REPLACE TABLE

PORTFOLIO\_TRACKING.SOURCE\_DATA.ABC\_BANK\_POSITION (

account\_id TEXT,

symbol TEXT,

description TEXT,

exchange TEXT,

report\_date DATE,

quantity NUMBER(38,0),

cost\_base NUMBER(38,5),

position\_value NUMBER(38,5),

currency TEXT

);

A best practice to conjugate project order, ease of editing, and to avoid many merge conflicts is to

define a different YAML file for each source system that we get data from.

1. We can try our source by writing the following query, which is the simplest for testing a source:

In DBT:

SELECT \*

FROM {{ source('abc\_bank', 'ABC\_BANK\_POSITION') }}

In Snowflake:

SELECT \*

FROM PORTFOLIO\_TRACKING.SOURCE\_DATA.ABC\_BANK\_POSITION

LIMIT 500

1. To clean up the dev environment, I used following commands in Snowflake

DROP VIEW "PORTFOLIO\_TRACKING"."UA"."POSITION\_ABC\_BANK";

DROP SCHEMA "PORTFOLIO\_TRACKING"."UA";

DBT Commands

For running tests

dbt test

Run test on all sources

dbt test -s source:\*

Run the tests on all the tables in the abc\_bank source:

dbt test -s source:abc\_bank.\*

Run the tests on the ABC\_BANK\_POSITION table of the abc\_bank source:

dbt test -s source:abc\_bank.ABC\_BANK\_POSITION

For running models

To run a particular folder

dbt run -s refined

dbt run --select REF\_POSITION\_ABC\_BANK

Generate documentation with DBT

dbt docs generate

Serving Documentation Locally with

dbt docs serve

Used <https://sqlfmt.com/> for linting code.

pip install shandy-sqlfmt[jinjafmt]

sqlfmt .

sqlfmt /path/to/my/dir /path/to/a/file.sql

STG models: the STG model will be just concerned with receiving and adapting the data, without changing its meaning.

Refined model: will use the names from the STG model and will add the needed business rules to perform any required calculation. The goal of the refined layer is clear – take the “adapted” data served from the staging layer and by applying business rules, including combining multiple sources, produce “refined” data that can be used in the data marts to provide useful information for the platform users