DBT

Data Transformation Service

JERIN SAM

Table of Contents

[DBT 1](#_Toc182406586)

[Appendix 1](#_Toc182406587)

[Important Links 1](#_Toc182406588)

# DBT

DBT is an open-source data transformation tool designed for analytics engineering. It is widely used for ***transforming***, ***testing***, and ***documenting data*** in data warehouses. It is the “T” in ETL, it is used for transformation using SQL based syntax.

In DBT, Transformation is ***version controlled***.

A screenshot of a computer

Description automatically generated

DBT is not a processing engine. Instead, it is a data transformation tool that ***relies on the processing power of your data warehouse or query engine***.

## What DBT Is

* **SQL-Based Transformation Tool:** DBT uses SQL to define data transformations. You write SQL queries (models), and DBT compiles them into SQL scripts that your data warehouse executes.
* **Orchestration of SQL Jobs:** DBT organizes, manages, and orchestrates SQL transformations, but it does not directly handle the processing of data.
* **Workflow Management:** DBT handles dependencies between data models, allowing you to build complex workflows. It ensures that transformations run in the correct order based on these dependencies.
* **Code Management:** DBT provides a structured framework for managing SQL code, applying data tests, documenting transformations, and version-controlling data pipelines.

## What DBT Is Not

* **DBT is** **not a Database or Processing Engine**: DBT does not store data, nor does it have its own query execution engine. It depends on a data warehouse, such as:
  + BigQuery
  + Snowflake
  + Redshift
  + PostgreSQL
  + Databricks
  + Azure Synapse
* **Not an ETL Tool:** DBT focuses only on the Transformation (T) aspect of the ETL/ELT process. It does not handle Extraction (E) or Loading (L) of data from source systems into a data warehouse.

## How DBT Works

* **SQL Compilation:** DBT takes your SQL files (models) and Jinja templates, compiles them into raw SQL statements, and submits them to your data warehouse for execution.
* **Data Warehouse Execution:** The actual data processing happens in the data warehouse. DBT does not perform the heavy lifting; it delegates the query execution to the warehouse.
* **Results Storage:** Once the data warehouse completes the query, the results are stored in the tables or views specified by DBT.

## DBT Helps To

1. Modelling changes are easy to follow and revert by using version control feature of DBT.
2. View and create dependencies between models.
3. Data quality checks.
4. Error reporting.
5. Incremental load of fact tables.
6. Track history of dimension tables.
7. Easy-to-access documentation.

# Setup DBT and Configure DBT Project

## Setup DBT

***Refer:***

* ***Dockerfile -*** */dbt-learn-hands-on/install\_and\_config/2-dbt-install/Dockerfile*
* ***Python Package Requirement*** - */dbt-learn-hands-on/install\_and\_config/2-dbt-install/* *requirements.txt*
* ***Installation Details*** - */dbt-learn-hands-on/install\_and\_config/2-dbt-install/* *script-dockerfile-execute.md*
* ***SQL Server ODBC Installation -*** */dbt-learn-hands-on/install\_and\_config/3-install-sql-server-odbc-docker-container/install-odbc-driver-container.md*
* ***Host Machine VS Code Connection with Container’s Python Interpreter –***

*/dbt-learn-hands-on/install\_and\_config/4-connect-vscode-docker-container/config-vscode-connect-python.md*

In this tutorial, DBT is installed on docker container. Following are the steps –

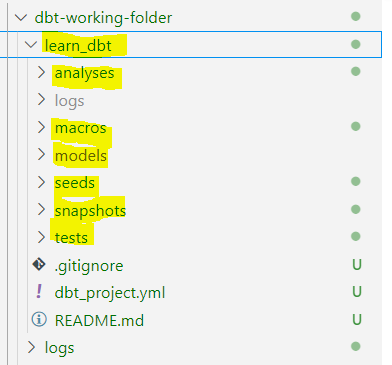
1. Create a *Dockerfile*, which contains Python 3.11.10 slim bookwork image.
   * Types of Docker Images
     + Bookworm - refers to the codename for the latest stable release of Debian.
     + Slim - indicates a minimal version of a Debian distribution with only the essential packages installed
     + Bullseye - is the codename for the previous stable Debian release
     + Alpine - is the “Dockerized” version of Alpine Linux
2. Install *dbt-core* and *dbt-sqlserver* *adapter* via *Dockerfile*.
3. Create working directory in host machine “***/dbt-learn-hands-on/main/dbt-working-folder***” and mount it with Docker container’s working directory using *Dockerfile*, so that any changes done through host machine can be synced in container’s working directory and vice versa.
4. Manually install SQL Server ODBC driver in the container.
5. After the ODBC setup, connect host machine’s VS Code to python interpreter present in Docker container.

## Configure and create DBT project

***Refer: /dbt-learn-hands-on/main/configure-dbt-project.md***

Following steps are taken to configure and create DBT folder structure in the container –

1. Create a Python Virtual Environment in the Docker Container and Activate It.
2. Create DBT Config/Profile Folder – Profile folder ***.dbt*** should be created at ***root*** directory. While initializing a DBT project using ***dbt init <project\_name>***a ***profiles.yml*** file will be created at this location to hold connection details of the database to be used by the dbt for the transformations. In this case its SQL Server.
3. Initialize DBT Project and Configure Database using ***dbt init <project\_name>*** - after executing the initialization script, a dbt folder structure will be created.



1. Test DBT Database Connection using ***dbt debug*** –
   * This command will be executed in the dbt project folder to test the connection with database used by DBT for the transformation, in this case its SQL Server.
   * This command uses ***profiles.yml*** file created in the above step.
   * ***Note -*** Generally, error occurs dung the testing, make sure that SQL Server connection details in the ***profiles.yml*** file should be inline with the reference profile file exist at ***“/main/dbt-working-folder/dbt-sql-server-profile.yml”,*** if required copy the same content in ***profiles.yml*** file.
2. Clean DBT Project Folder – This is required to delete examples exist in the dbt project folder.

# Appendix

## Important Links

* Dockerfile

<https://docs.docker.com/get-started/docker-concepts/building-images/writing-a-dockerfile/>

* Dockerfile CMD, RUN and ENTRYPOINT

<https://www.docker.com/blog/docker-best-practices-choosing-between-run-cmd-and-entrypoint/>

* Docker mount

<https://docs.docker.com/engine/storage/bind-mounts/>

* DBT SQL Server Installation

<https://docs.getdbt.com/docs/core/connect-data-platform/mssql-setup>