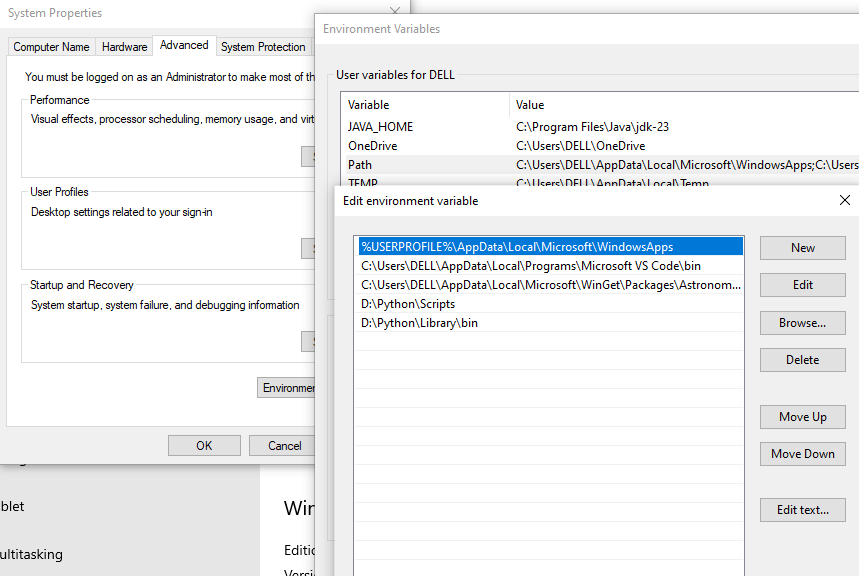
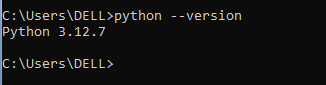
**STEP 1 : INSTALL PYTHON**

* https://www.python.org/downloads/release/python-3127/ Install windows installer 64 bit ( Make sure your python version is 3.12.7 )
* or install Anaconda that gives you jupyter notebook (Version 3.12.7)
* Prefer not to install in C:
* Find path of Python:
  + In command prompt: where python
* Optional: Add Python to PATH



**STEP 2: Check version**

Open CMD 🡪 Check python version if this is successfully installed



**STEP 3:** Create folder in your computer to manage Scripts & Code

* Folder name : Anything (dbt\_learn)
* CMD🡪 go to that folder using cd dbt\_learn;

**STEP 4:** Create Virtual Environment under that new folder.

* Why we need Virtual environment:

1. Different projects may need different versions of libraries like dbt-core, snowflake-connector, etc. Without a virtual environment, all these versions get installed globally and can clash with each other.

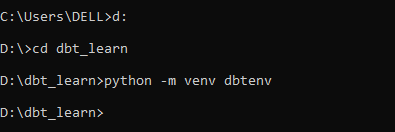
🔧 Example:

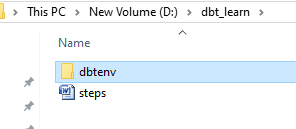
* Project A needs dbt-core==1.6
* Project B needs dbt-core==1.4
* Without isolation: **conflicts and errors**!

1. If something goes wrong or you want a fresh start, just delete the environment/ folder. That’s it. Much cleaner than uninstalling packages globally.

* Run below command :

**python -m venv <name> ( you can name anything like dbtenv)**





* **STEP 5 :** Activate your virtual environment

dbtenv\Scripts\activate -- Here dbtenv is the name of my virtual



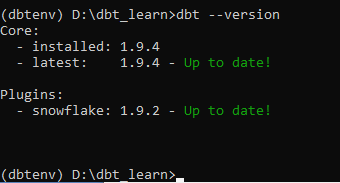
* **STEP6: Once you are in your Virtual environment, run below command to install dbt.**

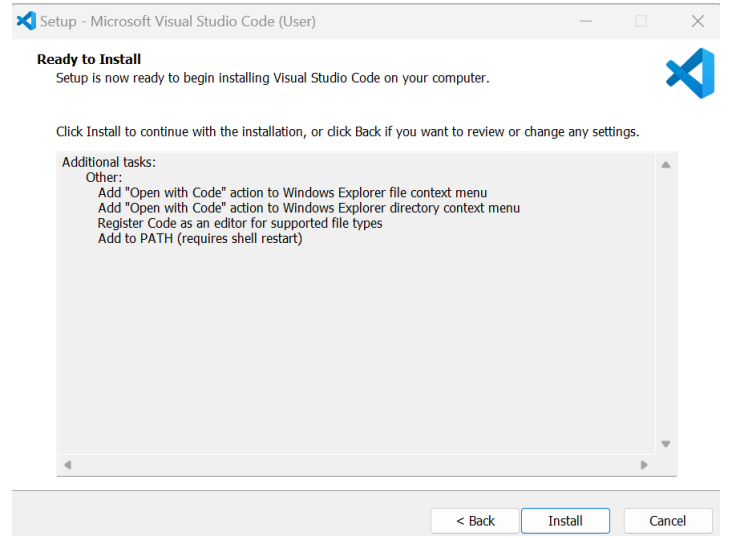
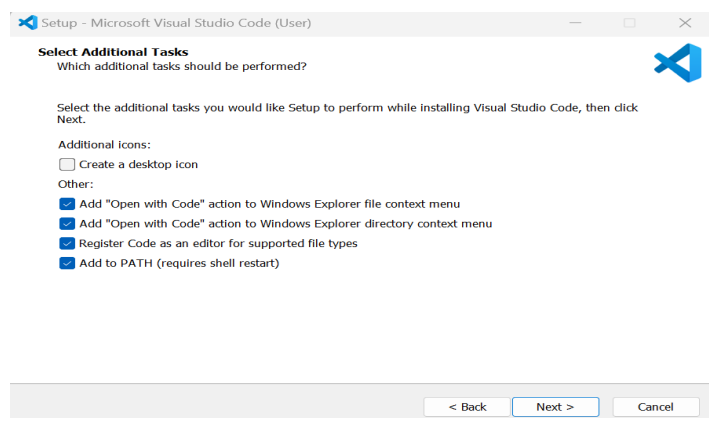
pip install dbt-snowflake

* **STEP7: CHECK dbt version.**

If you are seeing same as Screenshot,

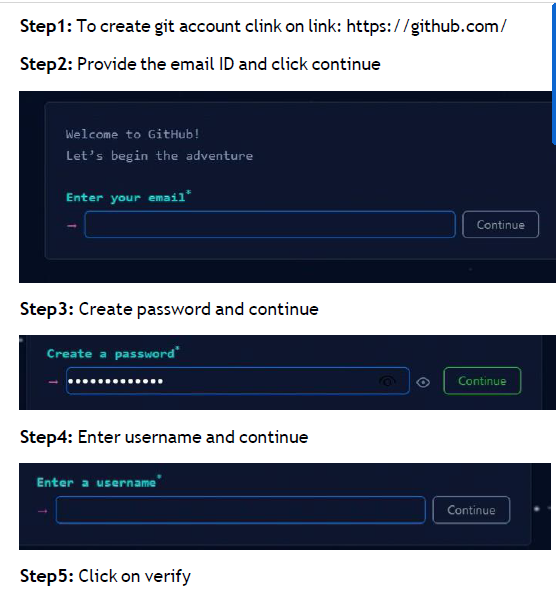
Congratulation, You are all set up.

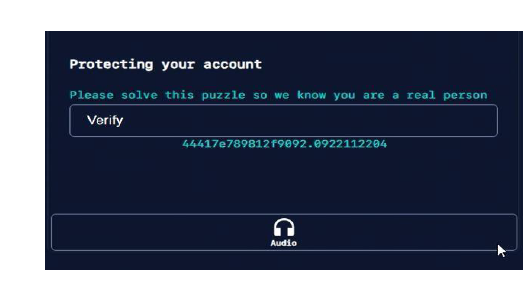


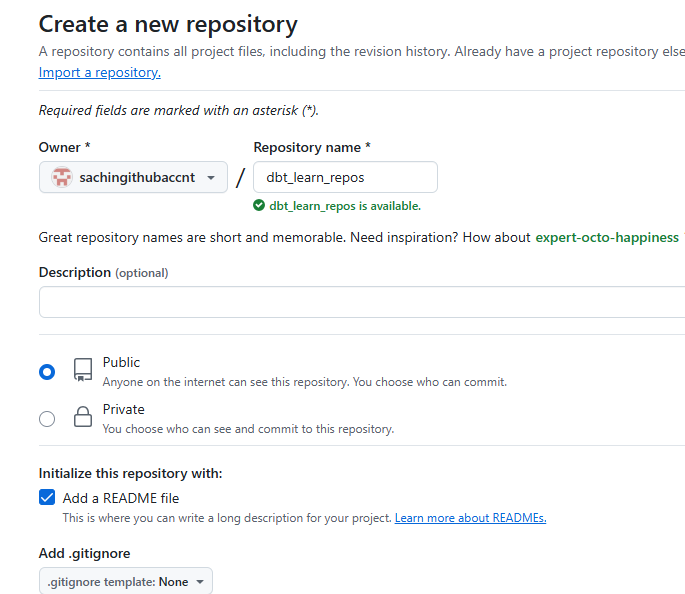
* **STEP8: Install VS Code**
  1. VS code setup: [Download Visual Studio Code - Mac, Linux, Windows](https://code.visualstudio.com/Download) for Visual Studio Code
  2. Step1: Open downloaded installer file
  3. Step2: Select I accept the agreement and check Next
  4. 
* **Why Git is important in a DBT project:Version control for your SQL models**
  1. DBT projects are made of .sql files, .yml configs, tests, etc. Git helps **track changes** over time.
     1. You can:
     2. See who changed what and why.
     3. Revert to earlier versions if something breaks.
     4. Compare different versions (diffs).
* **CI/CD Integration**

Teams often use Git + CI/CD pipelines to:

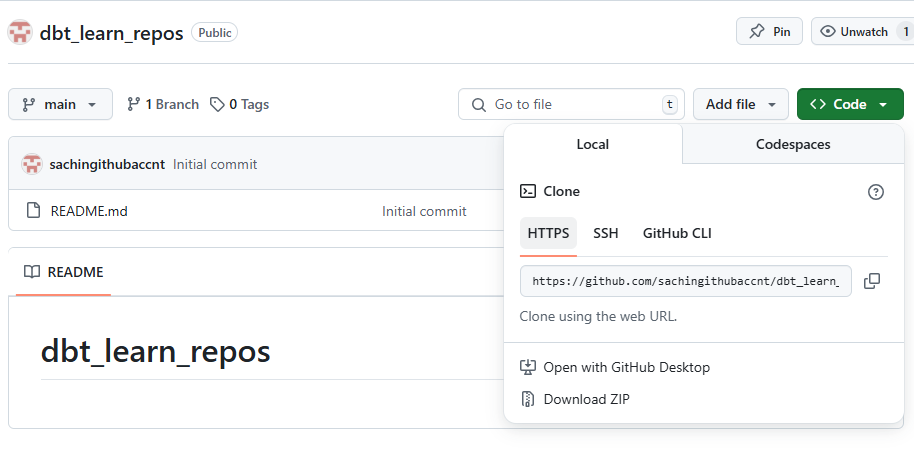
* Automatically run dbt build or dbt test on every push.
* Deploy models to production only after successful tests.







**Step7:** Copy HTTPS links for future reference

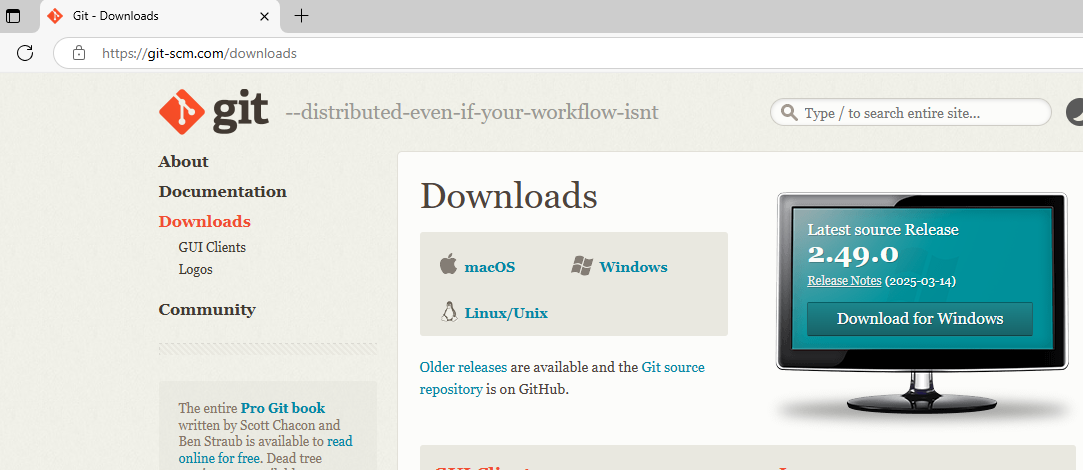


**/\* Optional\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**Step8:** Download git bash cmd from <https://git-scm.com/downloads>

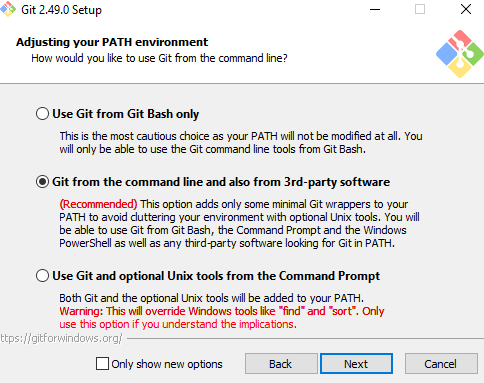
* **Why** Git Bash:

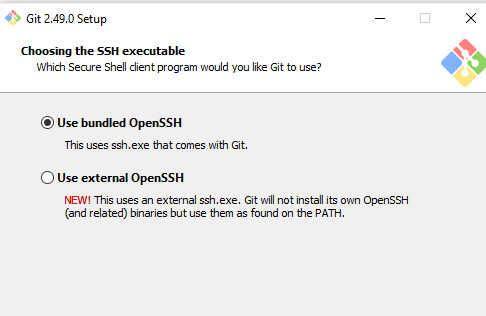
The Git Bash installer gives you the actual **Git engine** — the core tools you need for version control (git init, git clone, git commit

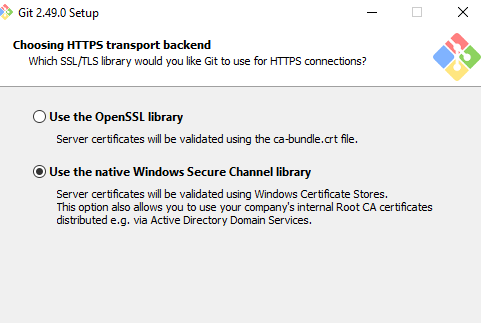


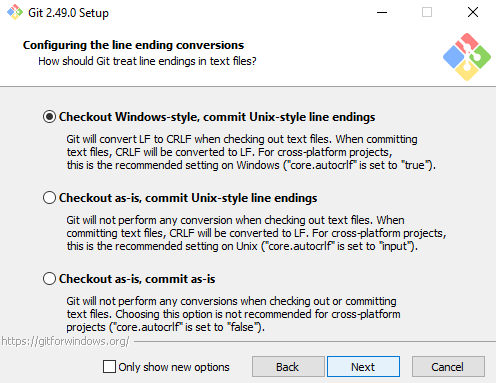
**So why not just use Command Prompt?**

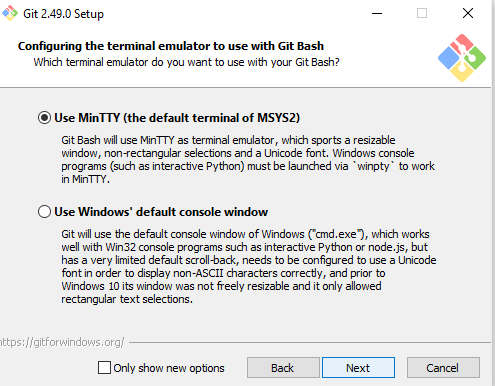
* Windows Command Prompt doesn't support Unix-style shell scripting.
* Commands like source, ls, export, etc. won't work.
* Many tutorials and tools (like **DBT**, **Python venv**, **Docker**, **Node**, etc.) assume you're using a Unix-like shell.
* **Install the Git bash**

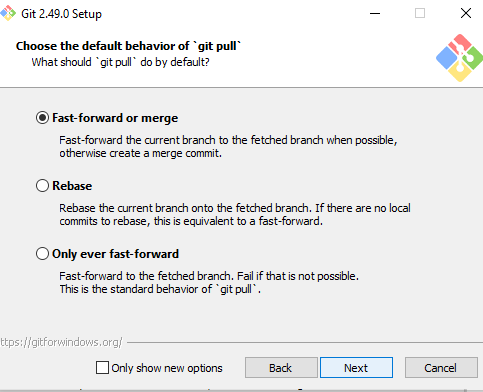
****

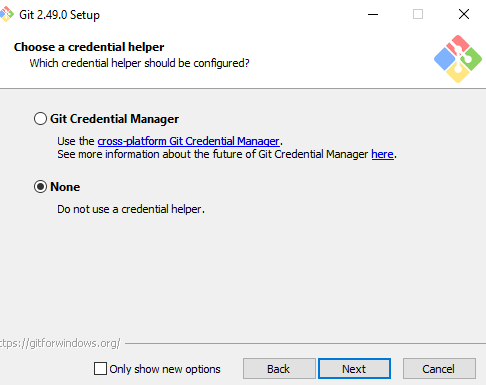
****

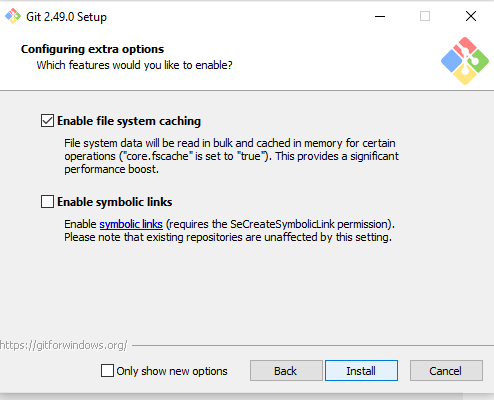
****

****

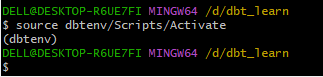
****

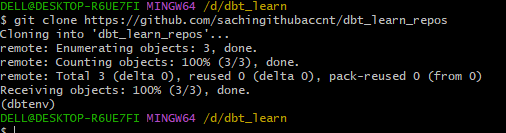
****

****

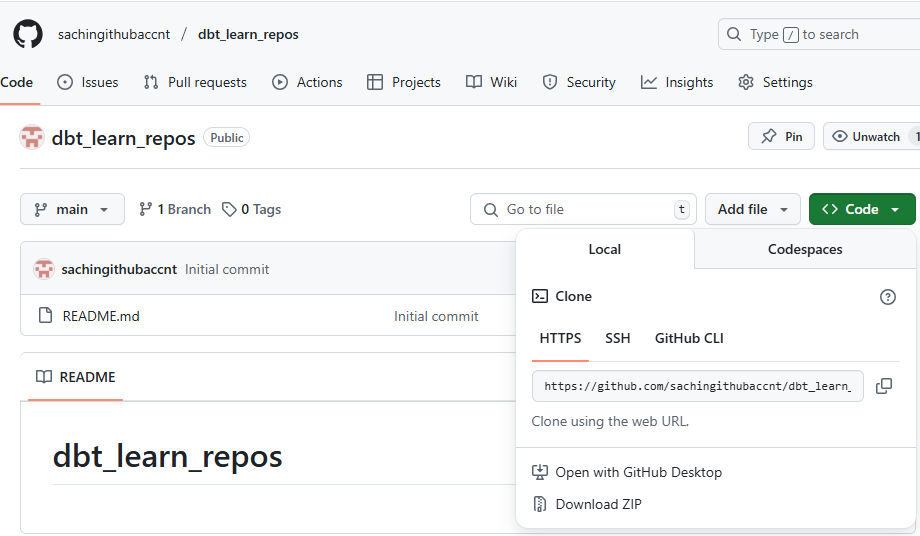
****

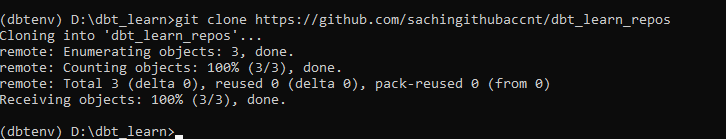
**Open the Git bash and activate the environment.**

****

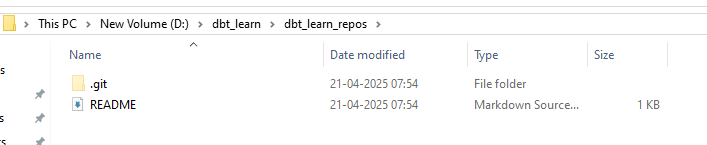
**Clone the repos  
  
**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

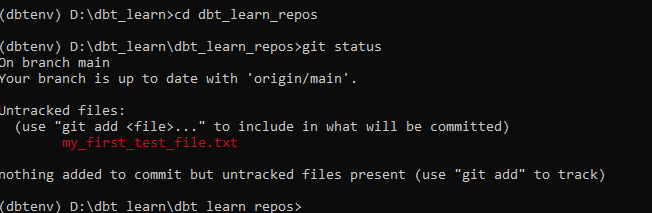
* **Step9: Clone the repository on your machine under the root folder i.e. dbt\_learn**
* Copy path from github  
    
  
* Go to CMD promt to your root folder and run below command  
    
  git clone <https://github.com/sachingithubaccnt/dbt_learn_repos>



* A folder would be created in your root folder



* **Step 10: Verify if we are able to PUSH and PULL from repos to env**  
    
   **PUSH:**
* Created a sample file in your dbt\_learn\_repos folder on your machine Push a file you created locally to GitHub (commit & push).



* Now Try to Add and commit the file. You can see some issues related to email and user. Please look below to resolve and commit first changes:

(dbtenv) D:\dbt\_learn\dbt\_learn\_repos>git add my\_first\_test\_file.txt

(dbtenv) D:\dbt\_learn\dbt\_learn\_repos>git commit -m "added test file from local"

Author identity unknown

\*\*\* Please tell me who you are.

Run

git config --global user.email "you@example.com"

git config --global user.name "Your Name"

to set your account's default identity.

Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'DELL@DESKTOP-R6UE7FI.(none)')

(dbtenv) D:\dbt\_learn\dbt\_learn\_repos>git config --global user.email "sachin.mittal04@gmail.com"

(dbtenv) D:\dbt\_learn\dbt\_learn\_repos>git config --global user.name "sachin"

(dbtenv) D:\dbt\_learn\dbt\_learn\_repos>git commit -m "added test file from local"

[main 67c1592] added test file from local

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 my\_first\_test\_file.txt

(dbtenv) D:\dbt\_learn\dbt\_learn\_repos>git push origin main

Username for 'https://github.com': sachin.mittal04@gmail.com

Password for 'https://sachin.mittal04%40gmail.com@github.com':

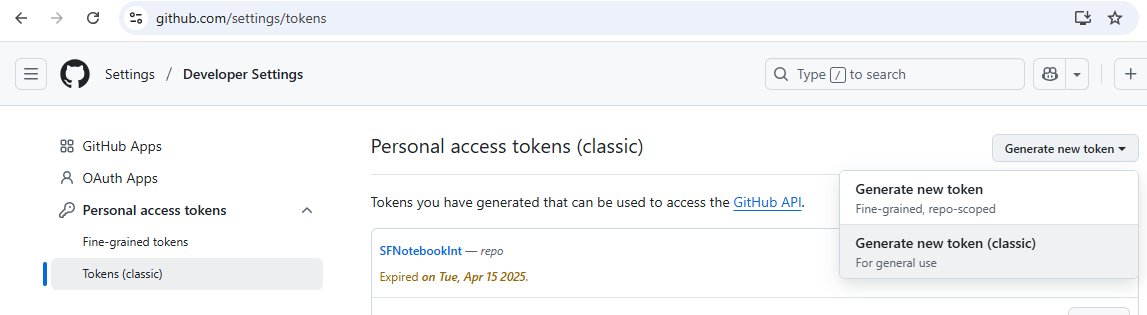
remote: Support for password authentication was removed on August 13, 2021.

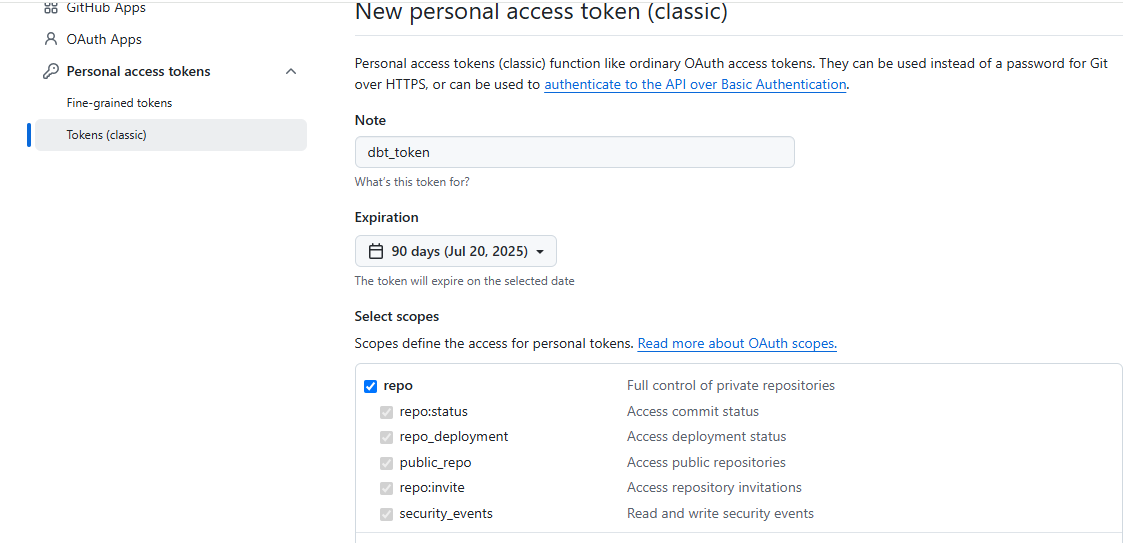
remote: Please see https://docs.github.com/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended modes of authentication.

fatal: Authentication failed for 'https://github.com/sachingithubaccnt/dbt\_learn\_repos/'

#### 1. ****Create a Personal Access Token (PAT)****

1. Go to: <https://github.com/settings/tokens>
2. Click **"Generate new token"** (Classic)
3. Give it a name like dbt setup token
4. Set expiration (choose 30 days, 90 days, or "no expiration")
5. Check scopes:
   * repo (for full control of private/public repos)
6. Click **Generate Token**
7. **Copy and save the token** somewhere secure — you won’t see it again.





**A personal Token is generated and save it in file as not available again:**

#### ****Use that token as password****

When Git asks:

Username for 'https://github.com': your\_github\_username

Password for 'https://github.com': <paste your token here>

(dbtenv) D:\dbt\_learn\dbt\_learn\_repos>git push origin main

Username for 'https://github.com': sachin.mittal04@gmail.com

Password for 'https://sachin.mittal04%40gmail.com@github.com':

Enumerating objects: 4, done.

Counting objects: 100% (4/4), done.

Delta compression using up to 4 threads

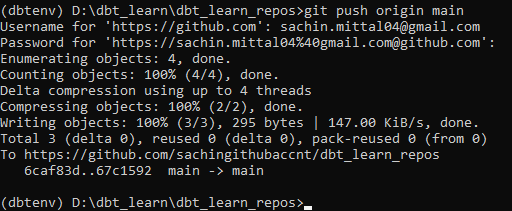
Compressing objects: 100% (2/2), done.

Writing objects: 100% (3/3), 295 bytes | 147.00 KiB/s, done.

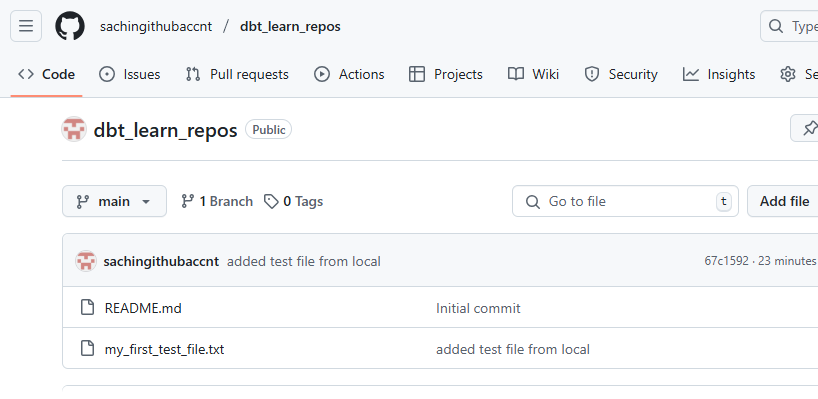
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

To https://github.com/sachingithubaccnt/dbt\_learn\_repos

6caf83d..67c1592 main -> main

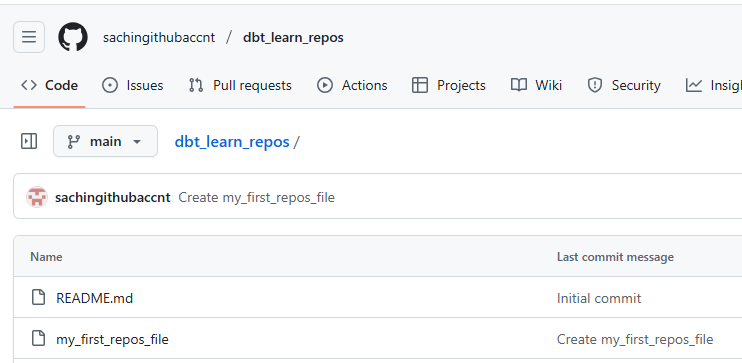


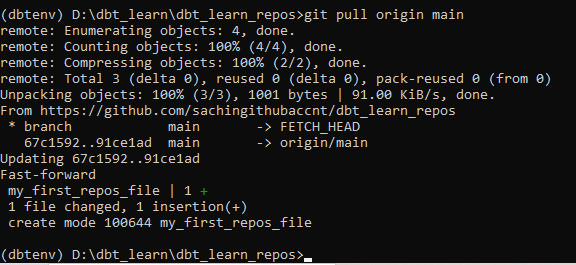
Verify in repository:



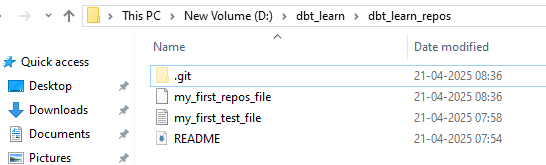
**PULL:**

* **Created a file in my repository which we will try to get on my local machine using PULL request:**

****

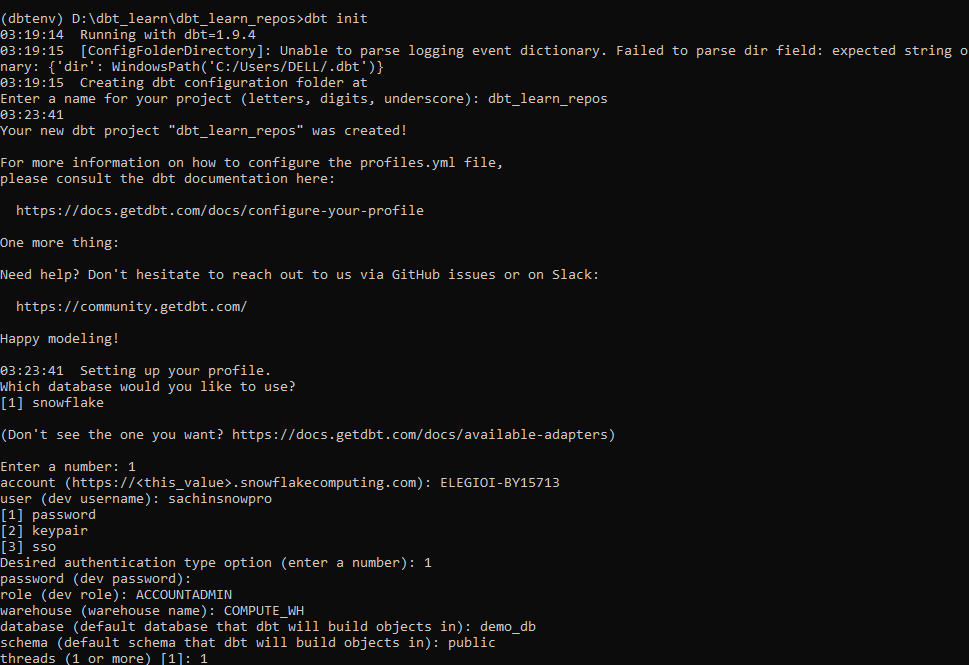
****

**Verify on your folder:**

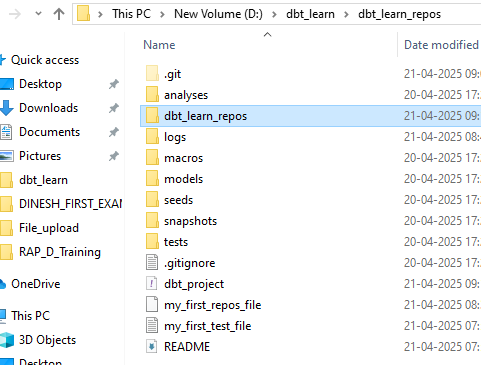
****

**Step 11: Now run the dbt init in your dbt\_learn\_repos folder**

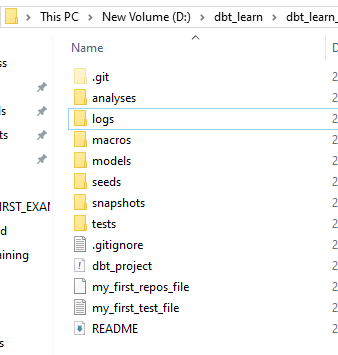
**Project name: same as your repos name i.e dbt\_learn\_repos**

****

Oncee you run dbt init this folder, it will create a **new directory**, and your Git setup won’t track it by default.



* Copy the Contents from dbt\_learn\_repos inner folder and paste it out side.
* Now delete the sub folder.



## Why we need to remove the inner folder (nested project folder)

When you run dbt init , it **creates a new folder named dbt\_learn\_repos** and puts all dbt files in there.

You'll end up with:

dbt\_learn\_repos/

└── dbt\_learn\_repos/

├── dbt\_project.yml

├── models/

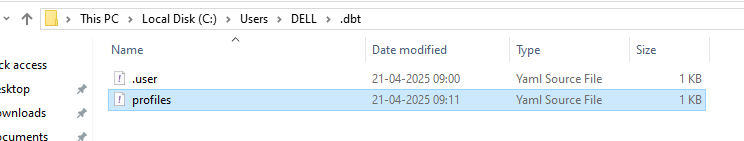
└── ...

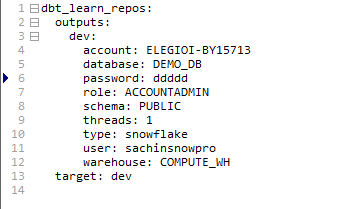
This structure is **redundant** and not ideal, especially when using Git and a virtual environment.

## Potential Issues if You Keep the Nested Folder

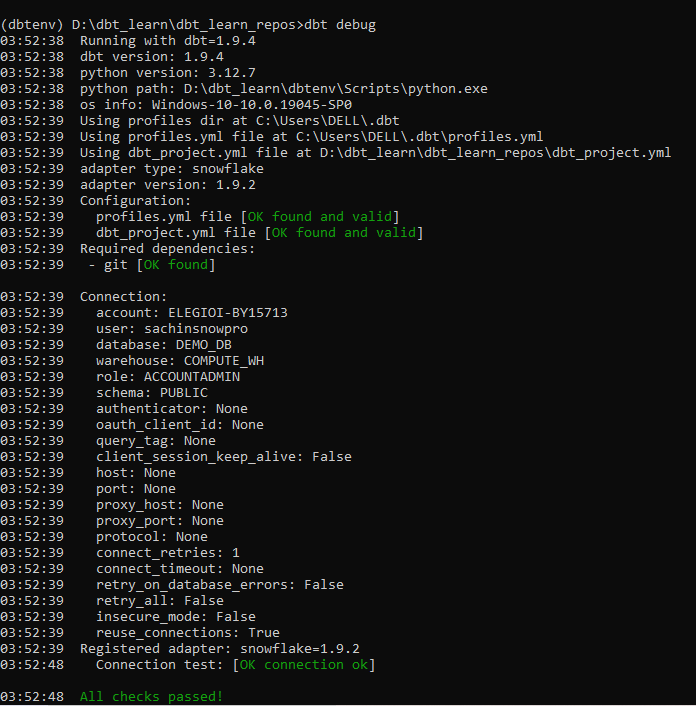
1. **Confusing File Paths**  
   You’ll always have to cd into the inner folder to run dbt commands.
2. **Git Confusion**  
   You cloned your Git repo to dbt\_learn\_repos — but dbt project is inside a subfolder, which defeats the purpose of the root Git folder.
3. **VS Code or IDE indexing problems**  
   Some editors won’t detect dbt\_project.yml properly unless it’s in the root.

**Step 12: Verify the profiles.yml file with all snowflake details**

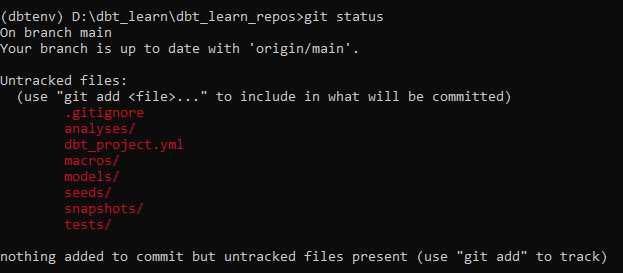


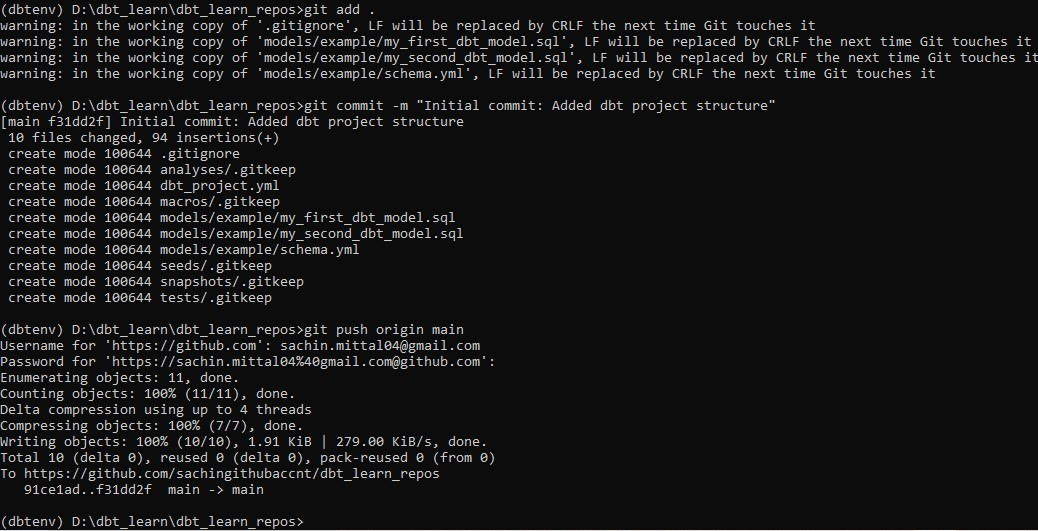
****

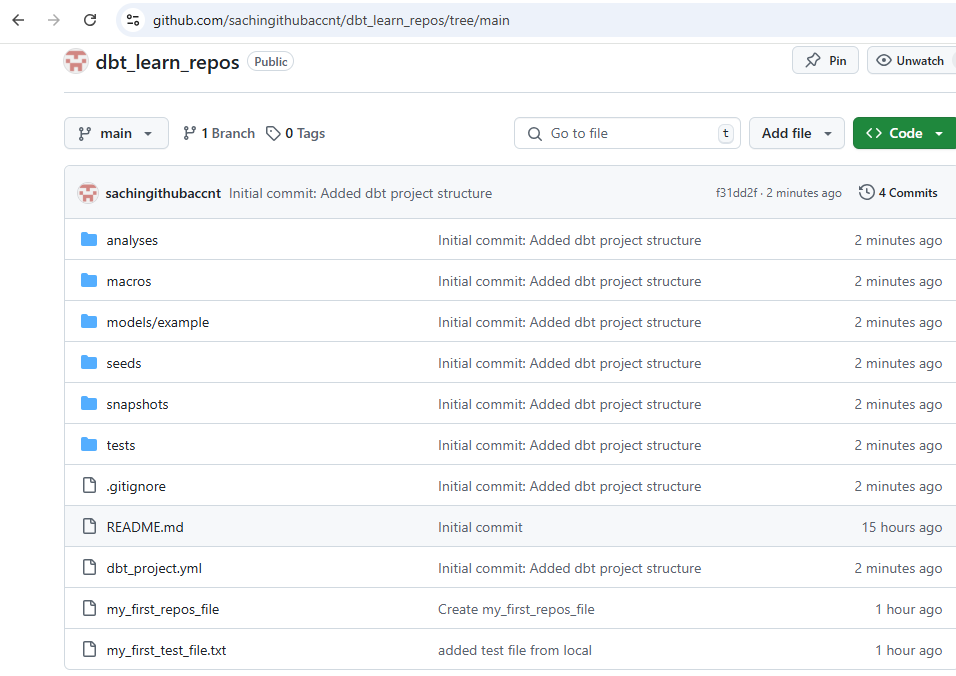
**Step13: Run dbt debug to verify if able to handshake with Snowflake**

****

**Step 14: perfect** time to commit your initialized dbt project to your Git repository!

****

****

****

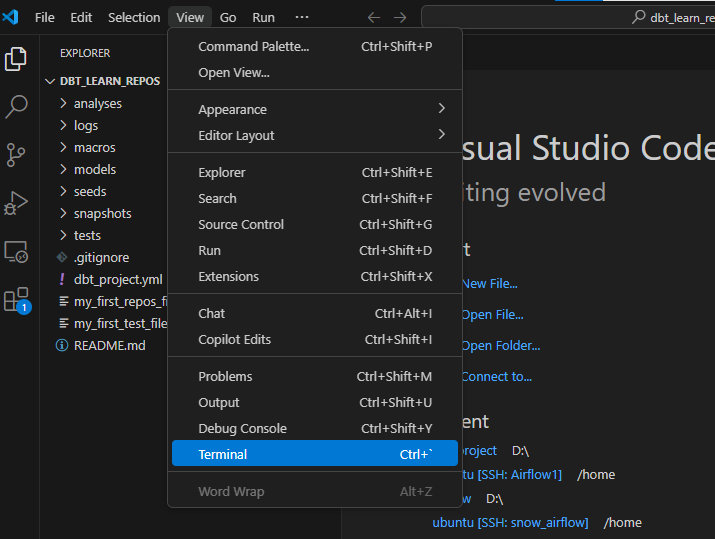
**Perfect at this moment we are good with CMD prompt and configure the DBT,GITHUB and Snowflake. But to edit the file or create a new file,models it is good we should have some IDE so we will work on Visual studio code:**

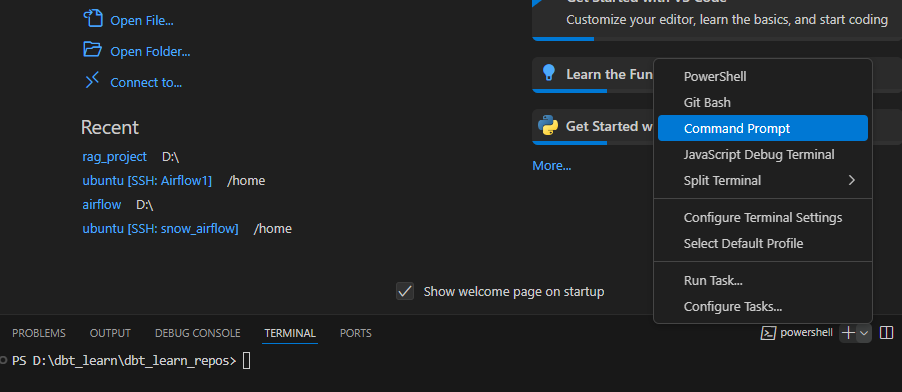
**Step 1: Open Your dbt Project in VS Code**

1. Open VS Code.
2. Click on **File > Open Folder**.
3. Select your project folder: D:\dbt\_learn\dbt\_learn\_repos.

**Step 2: Open Terminal in VS Code**

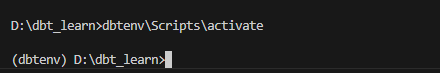
* Press Ctrl + ~ (tilde) or go to **Terminal > New Terminal**.
* Make sure the terminal path is pointing to D:\dbt\_learn\dbt\_learn\_repos.



**To avoid any system error, please switch to command prompt:**  
  


**Step 3: Activate the Virtual Environment**

**dbtenv\Scripts\activate**

****

Let’s walk through a **complete real-life example** to test Git push/pull workflow **in VS Code**, assuming your dbt project is already set up and committed once.

**You're starting from this point:**

* VS Code is open at: D:\dbt\_learn\dbt\_learn\_repos
* Terminal is open.
* Virtual environment is activated: (dbtenv) is showing.
* Your Git is initialized and connected to GitHub.

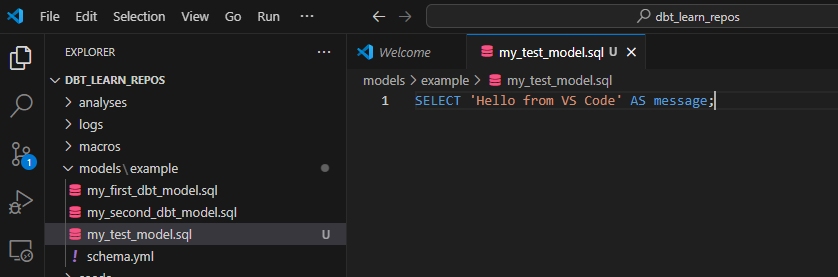
**GOAL**

1. Create a new file in models/ using VS Code.
2. Commit & push to GitHub via terminal in VS Code.
3. Simulate a pull from GitHub by creating a file directly on GitHub and pulling it locally.

**STEP 1: Create a New File in models/**

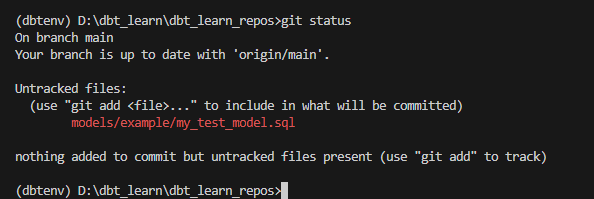
In VS Code Explorer:

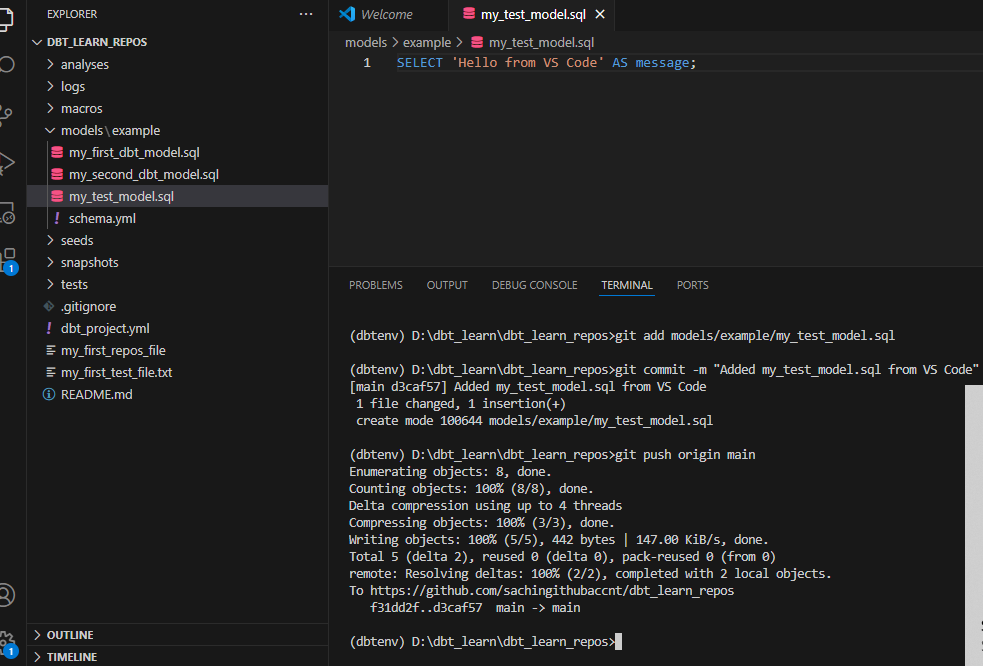
* Navigate to models/ folder.
* Right-click → **New File** → name it my\_test\_model.sql.
* Add this sample content:

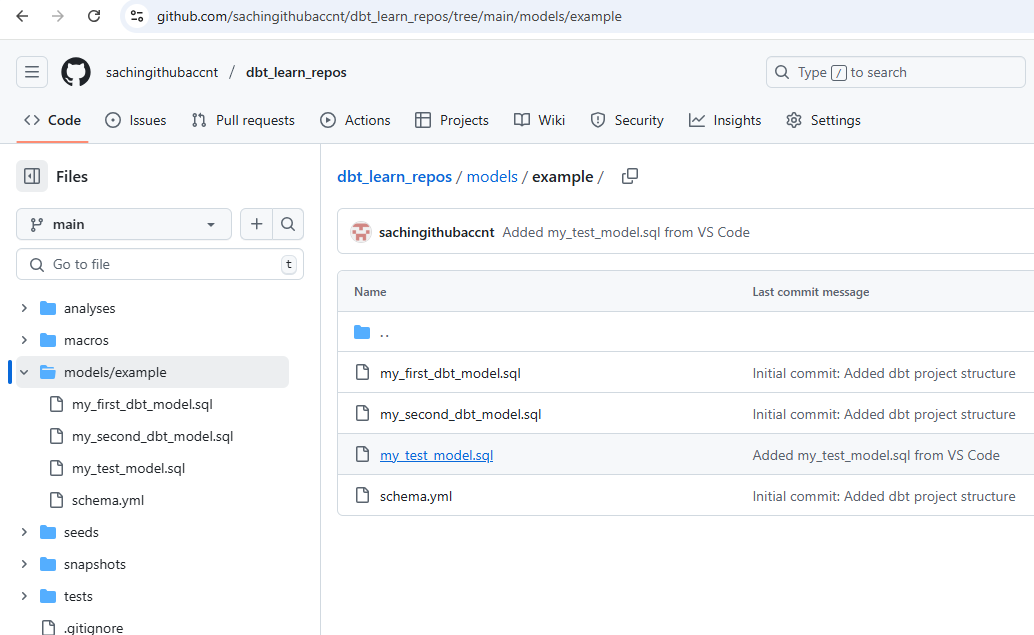
****

**STEP 2: Stage, Commit, and Push via Terminal in VS Code**

Run these in the terminal

****

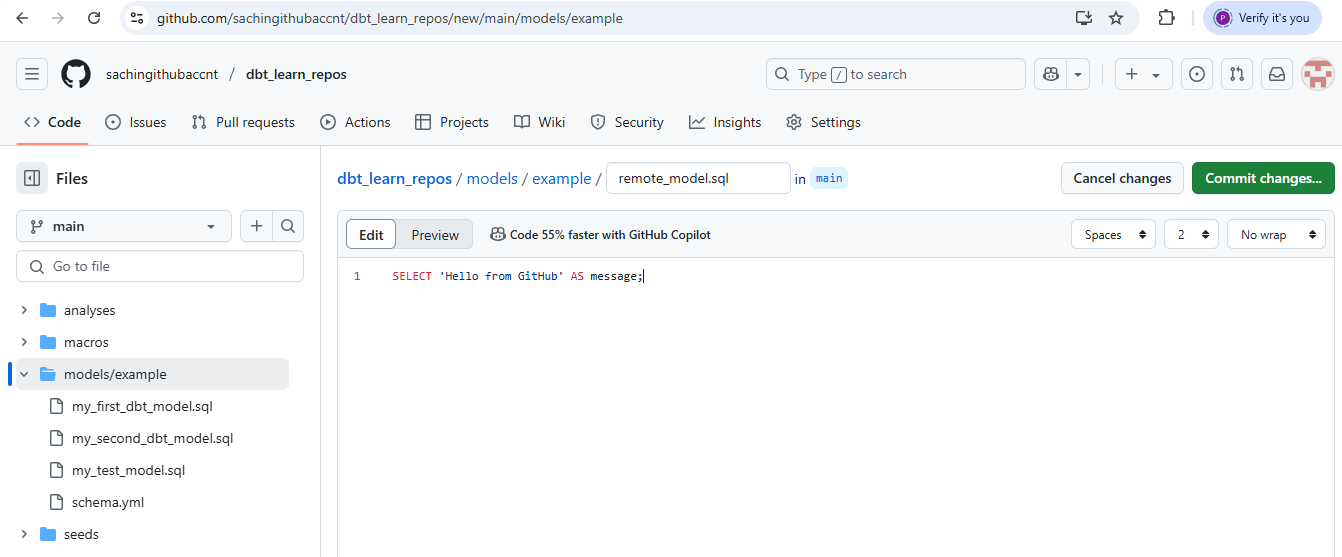
****

****

**STEP 3: Pull a Change from GitHub to Local**

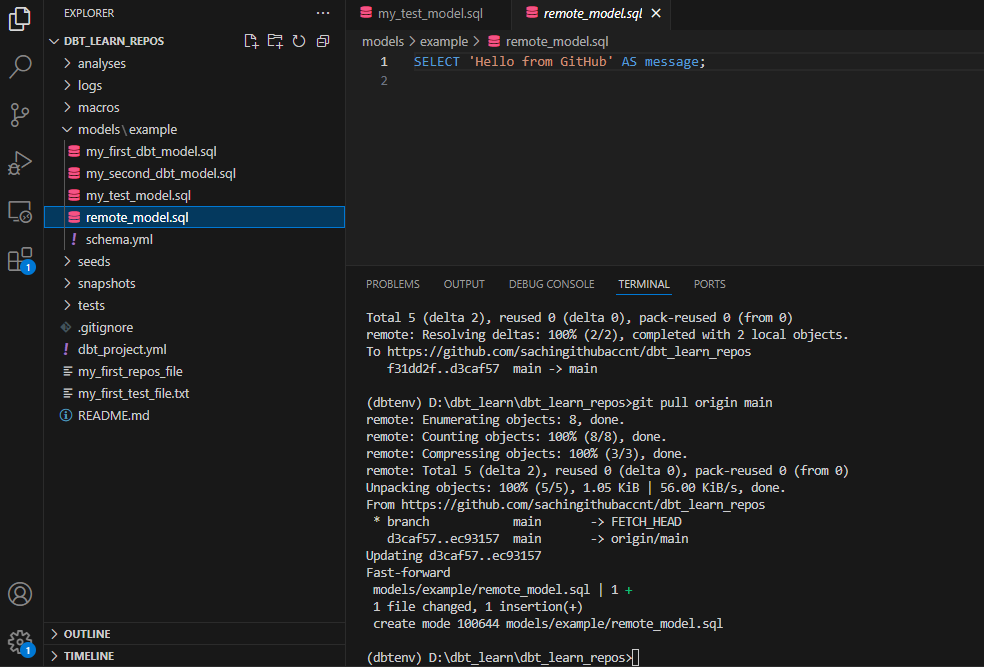
Now simulate a remote update:

1. Go to GitHub repo in your browser.
2. Click Add File > Create new file.
3. Name it models/remote\_model.sql
4. Add some content like:

****

**STEP 4: Pull the Remote File Locally**

Back in VS Code terminal:

****

**You’ve now tested:**

* Editing and pushing from VS Code.
* Pulling new changes made on GitHub.
* Seeing both synced in your local folder and on the cloud.

### ****Step-by-Step to Run DBT Model in VS Code****

#### ****1. Ensure DBT is Activated (Virtual Environment)****

Make sure your virtual environment is activated in your terminal inside VS Code. You should see (dbtenv) before the prompt:

(dbtenv) D:\dbt\_learn\dbt\_learn\_repos>

If not activated, activate it:

.\dbtenv\Scripts\activate

# Mac/Linux

source dbtenv/bin/activate

#### ****2. Navigate to the Correct Folder****

Ensure you are inside the DBT project directory:

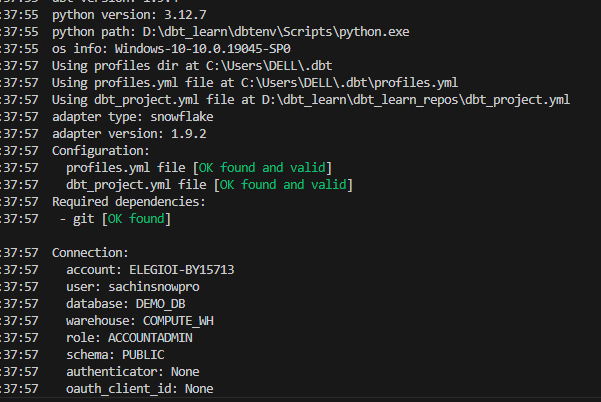
cd D:\dbt\_learn\dbt\_learn\_repos

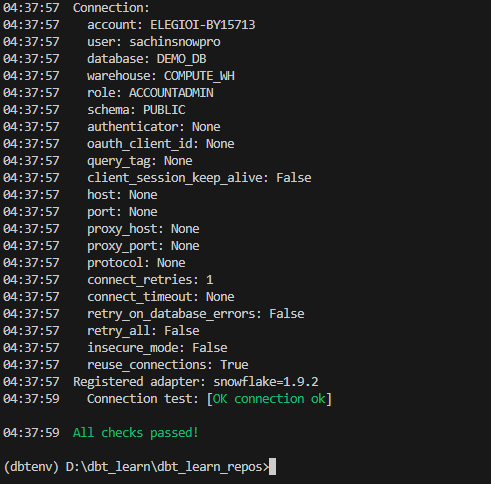
#### ****3. Test the Snowflake Connection****

Before running the model, let's test if your DBT can connect to Snowflake. Run:

dbt debug

This command checks your connection to Snowflake and other configurations. If you see a Connection successful message, that means DBT is configured correctly.



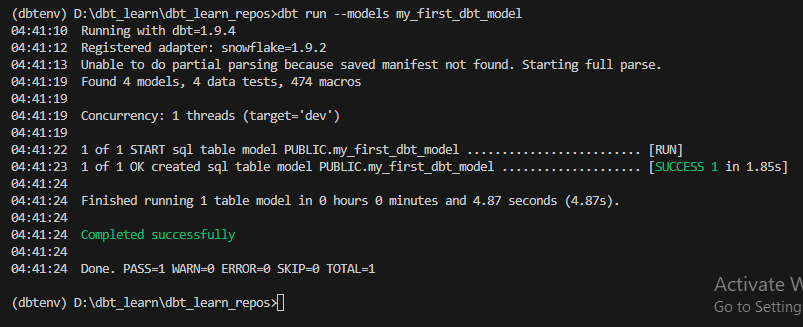


#### ****4. Run the Model****

To run your model and create the corresponding table/view in Snowflake, use the following command:

dbt run --models my\_first\_dbt\_model

This command will execute only the my\_first\_dbt\_model.sql model.



If you want to run all models, just use:

dbt run

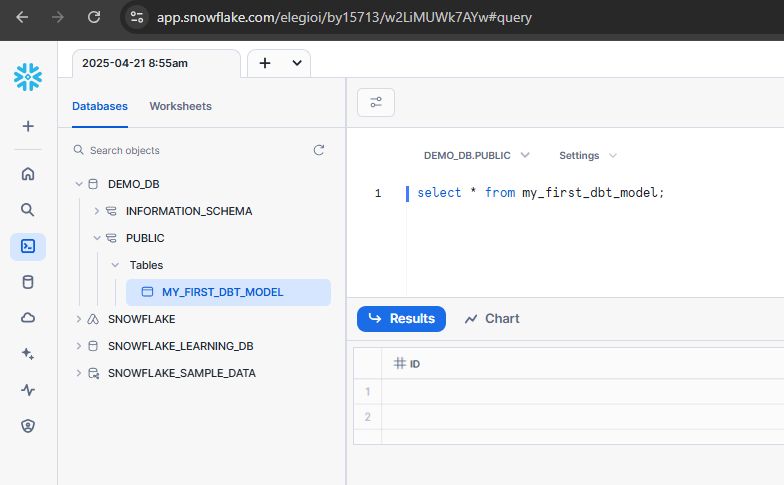
#### ****5. Check Snowflake for the Table/View****

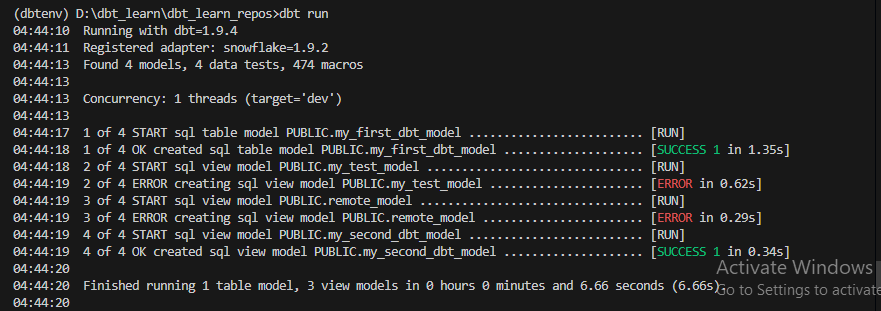
After the DBT run is complete, check your Snowflake database for the created table or view. You can log into Snowflake using either the UI or the SQL editor and run:

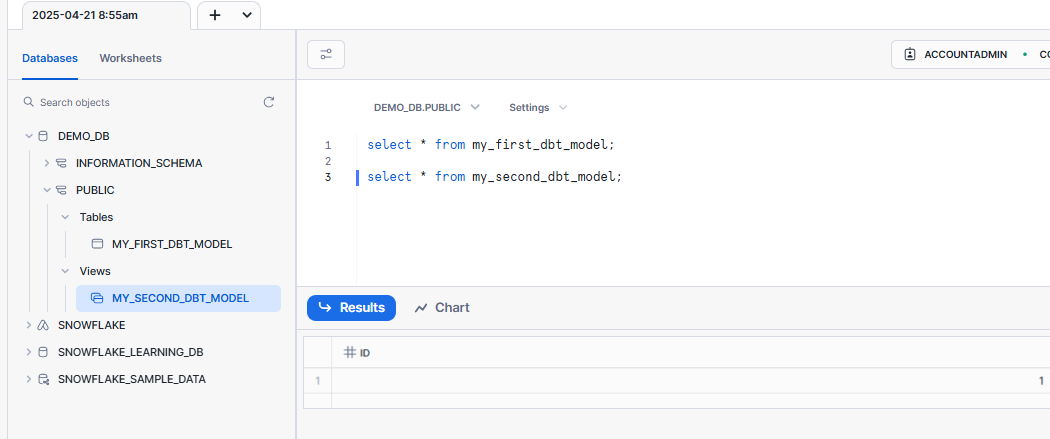
SHOW TABLES LIKE 'my\_first\_dbt\_model';

Or if you expect a view:

SHOW VIEWS LIKE 'my\_first\_dbt\_model';

****

****

****