

Lab 1: Preflight & Environment Readiness

Course: IT4065C – Data Technologies Administration

Repository: <https://github.com/ntious/IT4065C-Labs>

Time Required: ≤ 1 hour

Environment: Individual Ubuntu VM (VMware Cloud Services)

Goal of Lab 1

By the end of this lab, you will be able to:

1. Log into your Ubuntu virtual machine
2. Clone the course GitHub repository
3. Install PostgreSQL locally on your VM
4. Install dbt for PostgreSQL
5. Configure dbt to connect to a local database
6. Run dbt debug successfully
7. Run basic SQL queries against a local database

Note:

No transformations, no modeling, and no dbt run are required in this lab.

Step 1: Choose Your Ubuntu Environment (Required)

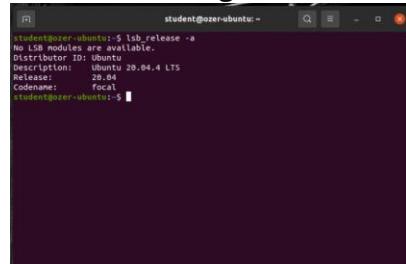
In **Step 1**, you must choose **one** of the following options:

- **Option A:** Use the UC-provided Ubuntu Virtual Machine (Sandbox)
- **Option B:** Install Ubuntu on your own Windows machine using the Microsoft Store

👉 You only need to complete **ONE option**. Do not do both.

Step 1.1 (OptionA): Log into Your Ubuntu VM (Using the UC Sandbox)

1. Log into your UC Sandbox Ubuntu VM at:
<https://range.ocri.io>
2. Use the following credentials (UC Sandbox only):
 - **Ubuntu VM Password:** Pa\$\$w0rd
3. Once logged in, open a Terminal window.
4. Confirm the operating system by running:
lsb_release -a
5. You should see Ubuntu 22.04 LTS (or higher)



```
student@ozer-ubuntu:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04 LTS
Release:      22.04
Codename:     focal
student@ozer-ubuntu:~$
```

! If the VM does not load correctly or you cannot log in, stop and contact your instructor or IT support before proceeding.

Step 1.2 (Option B): Creating Your Own Ubuntu Environment on Windows (Microsoft Store)

In this step, you will install Ubuntu Linux on Windows using the Microsoft Store. This creates a Linux environment using Windows Subsystem for Linux (WSL), no full virtual machine is required.

! This option is required only if you are not using the UC Sandbox VM.

Requirements

Before you begin, make sure you have:

- Windows 10 (version 21H2 or later) or Windows 11
- Administrator access on your computer
- Internet connection

Step 1.2.1: Open the Microsoft Store

1. Click the Start menu
2. Type Microsoft Store
3. Open the Microsoft Store app

Step 1.2.2: Download Ubuntu

1. In the Microsoft Store search bar, type Ubuntu
2. Select Ubuntu 22.04 LTS
 - **!** Do not select Ubuntu Server or preview versions
3. Click Get or Install
4. Wait for the download to complete

Step 1.2.3: Launch Ubuntu

1. Once installation finishes, click Open
(or open Ubuntu from the Start menu)

Ubuntu will initialize for the first time. This may take a few minutes.

Step 1.2.4: Create Your Linux User Account

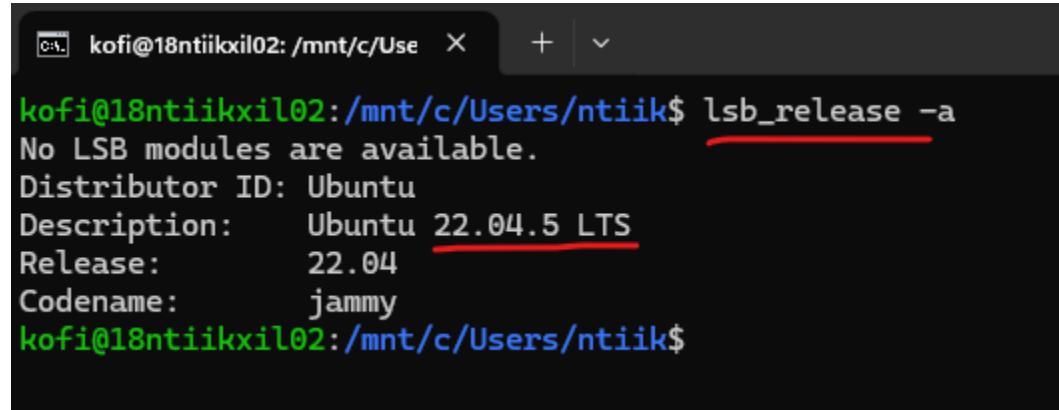
When prompted in the Ubuntu window:

1. Enter a Linux username
 - Lowercase letters only (e.g., student, ubuntu)
 2. Enter a password
 - You will not see characters as you type; this is normal
 3. Confirm the password
- !** Remember this password. You will need it throughout the course.

Step 1.2.5: Verify Ubuntu Installation

Once you see the Linux prompt, run:

```
lsb_release -a
```



```
kofi@18ntiikxil02:/mnt/c/Users/ntiik$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.5 LTS
Release:        22.04
Codename:       jammy
kofi@18ntiikxil02:/mnt/c/Users/ntiik$
```

Step 2: Update System Packages (Required for All Students)

Regardless of which option you chose, run the following commands **once**:

1. Run the following commands once:

```
sudo apt update
```

```
sudo apt upgrade -y
```

This helps prevent package conflicts later.

Step 3: Install Required System Tools

1. Install Git, Python, and PostgreSQL:

```
sudo apt install -y git python3 python3-pip python3-venv postgresql postgresql-contrib
```

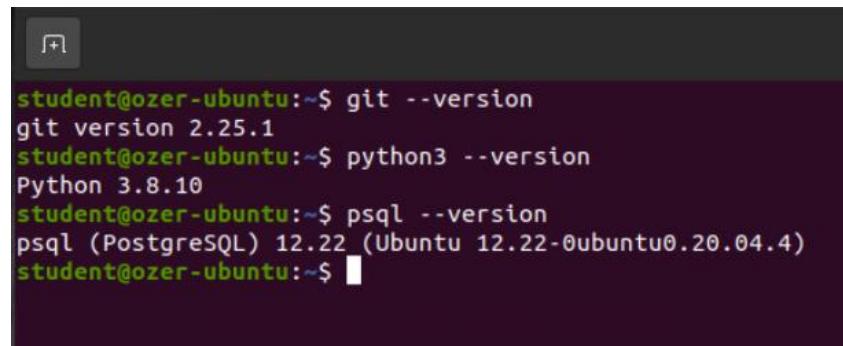
2. Verify each installation:

```
git --version
```

```
python3 --version
```

```
psql --version
```

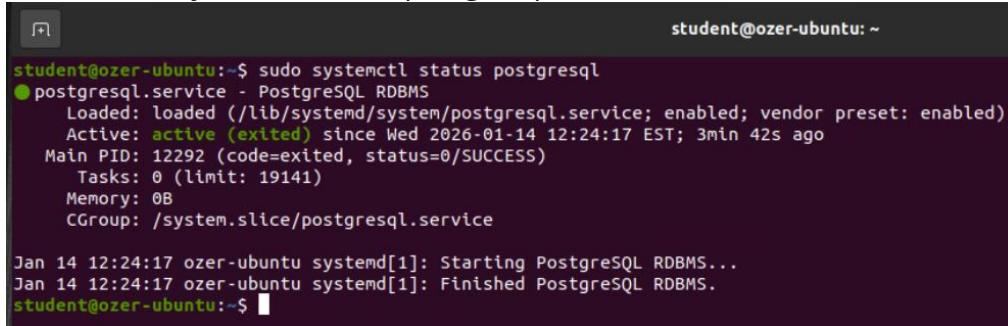
Each command should return a version number.



```
student@ozer-ubuntu:~$ git --version
git version 2.25.1
student@ozer-ubuntu:~$ python3 --version
Python 3.8.10
student@ozer-ubuntu:~$ psql --version
psql (PostgreSQL) 12.22 (Ubuntu 12.22-0ubuntu0.20.04.4)
student@ozer-ubuntu:~$
```

Step 4: Create Database and Schemas

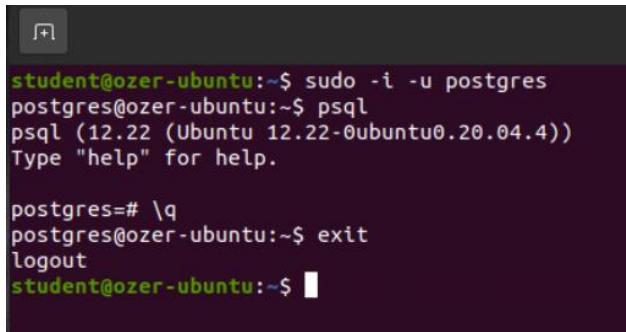
1. Check PostgreSQL service status:
sudo systemctl status postgresql



```
student@ozer-ubuntu:~$ sudo systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset: enabled)
   Active: active (exited) since Wed 2026-01-14 12:24:17 EST; 3min 42s ago
     Main PID: 12292 (code=exited, status=0/SUCCESS)
       Tasks: 0 (limit: 19141)
      Memory: 0B
        CGroup: /system.slice/postgresql.service

Jan 14 12:24:17 ozer-ubuntu systemd[1]: Starting PostgreSQL RDBMS...
Jan 14 12:24:17 ozer-ubuntu systemd[1]: Finished PostgreSQL RDBMS.
student@ozer-ubuntu:~$
```

2. Switch to the postgres user:
sudo -i -u postgres
3. Open the PostgreSQL shell:
Psql
4. Exit PostgreSQL:
\q
5. Exit the postgres user:
exit



```
student@ozer-ubuntu:~$ sudo -i -u postgres
postgres@ozer-ubuntu:~$ psql
psql (12.22 (Ubuntu 12.22-0ubuntu0.20.04.4))
Type "help" for help.

postgres=# \q
postgres@ozer-ubuntu:~$ exit
logout
student@ozer-ubuntu:~$
```

Step 5: Create Your Local Course Database and Schema

1. From your normal Ubuntu user account, run:

```
sudo -u postgres createdb it4065c || echo "Database already exists, continuing..."  
sudo -u postgres psql it4065c
```

If you see "database already exists," this is expected. Continue to the next step.

2. Inside PostgreSQL, run:

```
CREATE SCHEMA raw;  
CREATE SCHEMA student_kofi;
```

3. \q

This simulates schema-level governance isolation used later in the course.

```

student@ozer-ubuntu:~$ sudo -u postgres createdb it4065c || echo "Database already exists, continuing..."
student@ozer-ubuntu:~$ sudo -u postgres psql it4065c
psql (12.22 (Ubuntu 12.22-0ubuntu0.20.04.4))
Type "help" for help.

it4065c=# CREATE SCHEMA raw;
CREATE SCHEMA
it4065c=# CREATE SCHEMA student_kofi;
CREATE SCHEMA
it4065c=# \q
student@ozer-ubuntu:~$ 

```

4. Verify Schema Creation (Optional but Recommended)

```
sudo -u postgres psql it4065c -c "\dn+"
```

```

student@ozer-ubuntu:~$ sudo -u postgres psql it4065c -c "\dn+"
          List of schemas
   Name    | Owner     | Access privileges |      Description
---+-----+-----+-----+
public | postgres | postgres=UC/postgres+| standard public schema
      |           | =UC/postgres        |
raw   | postgres |                 | 
student_kofi | postgres |                 | 
(3 rows)

```

Step 6: Clone the Course GitHub Repository

- From your home directory run:

```
git clone https://github.com/ntious/IT4065C-Labs.git
cd IT4065C-Labs
```

- Confirm the repository structure:

```
ls
```

You should see the following folders:

- docs
- labs
- dbt
- scripts
- submissions

Screenshot 1: Take a screenshot showing the repository structure (docs, labs, dbt).

```

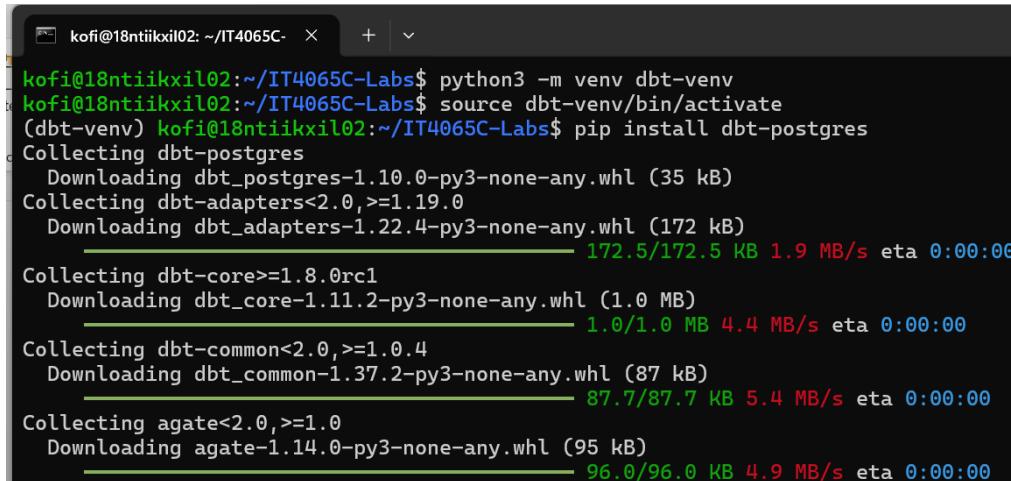
kofi@18ntiikxil02:~$ git clone https://github.com/ntious/IT4065C-Labs.git
Cloning into 'IT4065C-Labs'...
remote: Enumerating objects: 77, done.
remote: Counting objects: 100% (77/77), done.
remote: Compressing objects: 100% (67/67), done.
remote: Total 77 (delta 23), reused 9 (delta 2), pack-reused 0 (from 0)
Receiving objects: 100% (77/77), 23.24 KiB | 1.11 MiB/s, done.
Resolving deltas: 100% (23/23), done.
kofi@18ntiikxil02:~$ cd IT4065C-Labs
kofi@18ntiikxil02:~/IT4065C-Labs$ ls
LICENSE README.md dbt docs labs scripts submissions

```

You are now in the root directory of the course repository. All lab instructions will assume you are working from this location unless stated otherwise.

Step 7: Create and Activate dbt Virtual Environment

```
python3 -m venv dbt-venv  
source dbt-venv/bin/activate  
pip install dbt-postgres
```



```
kofi@18ntiikxil02:~/IT4065C-Labs$ python3 -m venv dbt-venv  
kofi@18ntiikxil02:~/IT4065C-Labs$ source dbt-venv/bin/activate  
(dbt-venv) kofi@18ntiikxil02:~/IT4065C-Labs$ pip install dbt-postgres  
Collecting dbt-postgres  
  Downloading dbt_postgres-1.10.0-py3-none-any.whl (35 kB)  
Collecting dbt-adapters<2.0,>=1.19.0  
  Downloading dbt_adapters-1.22.4-py3-none-any.whl (172 kB)  
          172.5/172.5 KB 1.9 MB/s eta 0:00:00  
Collecting dbt-core>=1.8.0rc1  
  Downloading dbt_core-1.11.2-py3-none-any.whl (1.0 MB)  
          1.0/1.0 MB 4.4 MB/s eta 0:00:00  
Collecting dbt-common<2.0,>=1.0.4  
  Downloading dbt_common-1.37.2-py3-none-any.whl (87 kB)  
          87.7/87.7 KB 5.4 MB/s eta 0:00:00  
Collecting agate<2.0,>=1.0  
  Downloading agate-1.14.0-py3-none-any.whl (95 kB)  
          96.0/96.0 KB 4.9 MB/s eta 0:00:00
```

1. Verify installation:

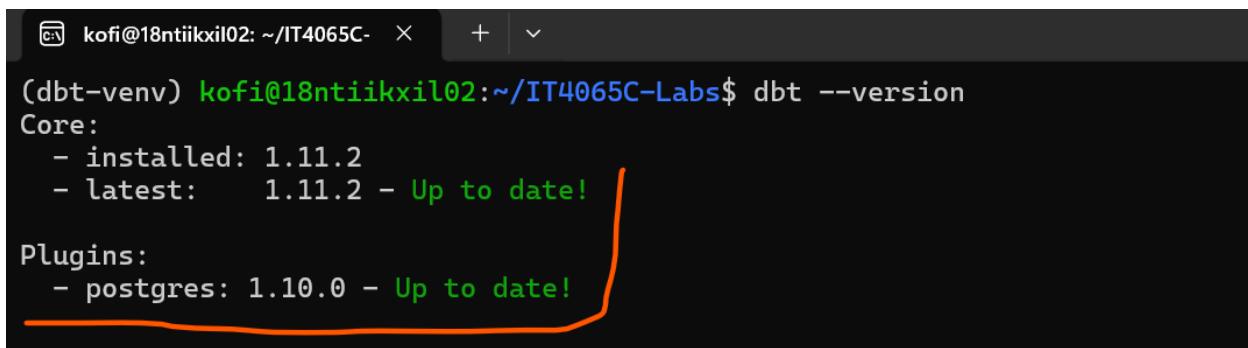
Run:

```
dbt --version
```

You must see the **Postgres adapter** listed.

Any time you work on dbt labs, you **must activate the virtual environment first**:

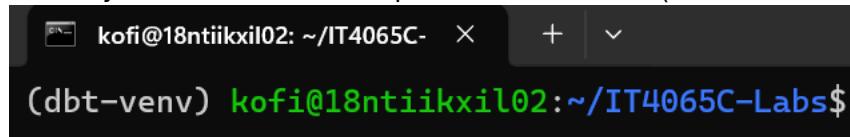
```
source ~/IT4065C-Labs/dbt-venv/bin/activate
```



```
(dbt-venv) kofi@18ntiikxil02:~/IT4065C-Labs$ dbt --version  
Core:  
  - installed: 1.11.2  
  - latest: 1.11.2 - Up to date!  
  
Plugins:  
  - postgres: 1.10.0 - Up to date!
```

Step 8: Configure dbt profiles.yml (Project Profile)

1. Ensure you are in the course repo and venv is active (as shown below)



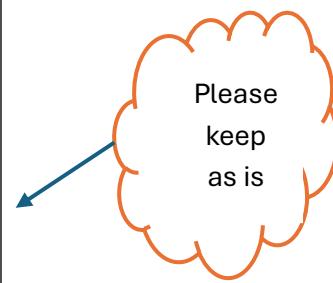
```
(dbt-venv) kofi@18ntiikxil02:~/IT4065C-Labs$
```

2. Navigate to the dbt project folder
cd dbt/it4065c_platform

```
kofi@18ntiikxil02:~/IT4065C- × + | v
(dbt-venv) kofi@18ntiikxil02:~/IT4065C-Labs$ cd dbt/it4065c_platform
(dbt-venv) kofi@18ntiikxil02:~/IT4065C-Labs/dbt/it4065c_platform$
```

3. Edit the project profile
nano profiles.yml
4. Set the profile to this exact configuration (copy and paste)

```
it4065c_platform:
  target: dev
  outputs:
    dev:
      type: postgres
      user: postgres
      password: "Pa$$w0rd123!"
      host: localhost
      port: 5432
      dbname: it4065c
      schema: student_kofi
      threads: 2
      sslmode: prefer
```



Note: Save changes and exit:

- **CTRL + O → Enter**
- **CTRL + X**

```
GNU nano 6.2      profiles.yml *
it4065c_platform:
  target: dev
  outputs:
    dev:
      type: postgres
      user: postgres
      password: "Pa$$w0rd123!"
      host: localhost
      port: 5432
      dbname: it4065c
      schema: student_kofi
      threads: 2
      sslmode: prefer
```

Note: This password was set earlier during database setup to allow dbt and psql to authenticate over localhost. Use it exactly as shown.

Step 9: dbt Debug

1. Run:
dbt debug

Success Criteria

You must see:

- Profile loaded successfully
- Connection test: OK

```
kofi@18ntiikxil02:~/IT4065C- 04:24:39 python path: /home/kofi/IT4065C-Labs/dbt-venv/bin/python3  
04:24:39 os info: Linux-5.15.167.4-microsoft-standard-WSL2-x86_64-with-glibc2.35  
04:24:39 Using profiles dir at /home/kofi/IT4065C-Labs/dbt/it4065c_platform  
04:24:39 Using dbt_project.yml file at /home/kofi/IT4065C-Labs/dbt/it4065c_platform/profiles.yml  
04:24:39 Using dbt_project.yml file at /home/kofi/IT4065C-Labs/dbt/it4065c_platform/dbt_project.yml  
04:24:39 adapter type: postgres  
04:24:39 adapter version: 1.10.0  
04:24:39 Configuration:  
04:24:39   profiles.yml file [OK found and valid]  
04:24:39   dbt_project.yml file [OK found and valid]  
04:24:39 Required dependencies:  
04:24:39   - git [OK found]  
04:24:39 Connection:  
04:24:39   host: localhost  
04:24:39   port: 5432  
04:24:39   user: postgres  
04:24:39   database: it4065c  
04:24:39   schema: student_kofi  
04:24:39   connect_timeout: 10  
04:24:39   role: None  
04:24:39   search_path: None  
04:24:39   keepalives_idle: 0  
04:24:39   sslmode: prefer  
04:24:39   sslcert: None  
04:24:39   sslkey: None  
04:24:39   sslrootcert: None  
04:24:39   application_name: dbt  
04:24:39   retries: 1  
04:24:39 Registered adapter: postgres=1.10.0  
04:24:39 Connection test: [OK connection ok]  
04:24:39 All checks passed!
```

If this fails, consult docs/troubleshooting.md.

💡 **Screenshot 2:** Take a screenshot showing successful dbt debug.

Step 10: Basic SQL Validation

1. Open PostgreSQL using the postgres user:

Run:

```
psql -h localhost -U postgres -d it4065c
```

Use password: Pa\$\$w0rd123!

2. Run:

```
CREATE TABLE student_kofi.test_table (id INT);
INSERT INTO student_kofi.test_table VALUES (1), (2);
SELECT COUNT(*) FROM student_kofi.test_table;
```

```
kofi@18ntiikxil02:~/IT4065C- 04:24:39 (dbt-venv) kofi@18ntiikxil02:~/IT4065C-Labs/dbt/it4065c_platform$ psql -h localhost -U postgres -d it4065c
Password for user postgres:
psql (14.20 (Ubuntu 14.20-Ubuntu0.22.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

it4065c=# CREATE TABLE student_kofi.test_table (id INT);
CREATE TABLE
it4065c=# INSERT INTO student_kofi.test_table VALUES (1), (2);
INSERT 0 2
it4065c=# SELECT COUNT(*) FROM student_kofi.test_table;
 count
-----
 2
(1 row)

it4065c=
```

💡 **Screenshot 3:** Take a screenshot showing the SELECT COUNT(*) result.

1. Exit PostgreSQL:

```
\q
```

Step 11: Lab 1 Deliverables

Submit **one MS Word or PDF file** on Canvas containing:

1. Screenshot 1: Repository structure (docs, labs, dbt)
2. Screenshot 2: Successful dbt debug output
3. Screenshot 3: SELECT COUNT(*) query output
4. Short reflection (3–5 sentences):

What schema am I responsible for, and why does schema isolation matter in governed data systems?

If you exit or reboot your VM, resume by:

1. Opening a terminal
cd ~/IT4065C-Labs
source dbt-venv/bin/activate