labassignment1

September 21, 2025

0.1 DS 6600: Data Engineering 1

0.1.1 Lab assignment 1

Instructions: use a Jupyter notebook to complete the following questions. Use a markdown cell for text and a code cell for Python code. Once your notebook is complete, follow these instructions to save the notebook as a PDF file and submit your assignment to Gradescope on Canvas.

0.2 Problem 1

We are going to use Git and GitHub for this lab assignment. Please complete all of the following steps:

- Create a new GitHub repository for this lab assignment called "DS6600_lab1". Make it public, add a README.md file, choose a .gitignore file that is specific to Python-based projects. Also, choose a licence (using choosealicense.com) that allows other people to use the code in your repository for both commercial and non-commercial use, including full rights to distribute and modify the code, but does not allow anyone to modify and distribute your code with a more restrictive license than the one included in your repository.

- In your terminal, choose a location on your computer for the local copy of this repository and use `git clone` to download it and connect this local directory to the GitHub repository.

- In your README.md file, explain (in your own words! Copy-and-pasted answers will receive no credit) why you chose the license you did and what would have happened had you not supplied a license. In addition, examine your .gitignore file to confirm that a file named ".env" will not be pushed to GitHub. Copy and paste the relevant section of the .gitignore file in your README.md file.

- As you work on this assignment, add, commit, and push your changes to GitHub. We will look at your repository's commit history to make sure there are at least 6 commits (one for each of the questions on this lab).

For this question on your assignment, all you need to supply in this notebook is a URL for your GitHub repository. Please type it here. [8 points]

Github repository URL:

https://github.com/stephanie-morrall/DS6600_lab1

0.3 Problem 2

Describe, in words, whether a virtual machine, a container, a virtual environment, or the global environment of a single computer is best suited for each of the following situations. Be clear about why the option you choose works best, and also about why the other options are insufficient or are overkill.

0.3.1 Part a

The local chapter of Meals on Wheels worked with some UVA computer science students and volunteers from WillowTree Apps, who built them a web portal for volunteers to sign up for shifts and a mobile app that shows volunteers their driving routes with GPS mapping. The portal and app have been extremely useful for the local Meals on Wheels chapter, but unfortunately, the portal and app were built only for the Charlottesville chapter and is not usable by any other Meals on Wheels chapter at present. You've been asked to help every other Meals on Wheels chapter to benefit from these products by generalizing the code, and then making it available for free to any chapter that wants to use the portal and app. Every chapter has their own database of volunteers and clients, necessitating a large amount of storage that must also be accessible through an internet connection. [6 points]

Container

Reason for container: It allows for reproducible and easy deployment of multiple software components without version conflicts, while also allowing each chapter to use its own database of volunteers and clients.

Reason against virtual machine: Overkill for the task because each VM duplicates a full computer remotely which increases cost and adds maintenance overhead compared to container deployments.

Reason against virtual environment: Too narrow because it only isolates language packages without bundling in system dependencies or providing OS-level isolation.

Reason against global environment: Fragile and unreproducible due to version conflicts, security risks, and poor scalability across multiple independent deployments.

0.3.2 Part b

The Legal Aid Justice Center, a Charlottesville-based legal aid nonprofit that advocates for the rights of immigrants and the working poor, has used a Freedom of Information Act request to get Virginia's Department of Labor and Industry (DOLI) to share their data on all official complaints of wage theft in Virginia over the last five years. The DOLI stored this data in an Excel file with horrible formatting: the column headers are given vague names, text has misspelling throughout the file (you see locations such as "Rochmond" and "Norflok"), data types are inconsistent within columns (sometimes a $ is included with the wage amounts, sometimes $ is excluded), and so on. They've asked you to clean the data and generate a report that lists the average claim of wage theft and a frequency table of the number of claims by locality. They don't need your code or data, just the report. [6 points]

Could be done in a Global Environment but isolating to at least the Virtual Environment level is best practice for most projects, including this one

Reason for virtual environment: It provides just enough isolation to install and manage the specific

Python libraries needed for cleaning the Excel file and generating the report, without interfering with other projects on the same computer. We don't need any more isolation than this because we don't need to share the code.

Reason against virtual machine: Overkill for simple data cleaning and reporting since a full remote operating system duplication is unnecessary and adds setup and maintenance overhead.

Reason against container: Adds more complexity than needed because there is no requirement to deploy the code as a reproducible service or run it across multiple machines.

Reason against global environment: More fragile because installing packages directly on the system can create conflicts with other projects and make the work harder to maintain.

0.3.3 Part c

An international NGO called Save the Children provides housing, food, medical care, and other assistance to individuals who have been displaced from their homes by conflict or natural disasters. But a major problem is that these displacement events can happen suddenly, without giving Save the Children any time to mobilize. Major migration events are more likely when the infrastructure and buildings in a city or town have been destroyed. Save the Children have asked you to build a predictive model that can detect from land-satellite images the extent of infrastructure damage in an area of interest and that can be deployed by several different individuals working for Save the Children. You build this model, but in addition to Python you require several open-source image processing software packages that only work on Ubuntu Linux. [6 points]

Virtual machine

Reason for virtual machine: The predictive model depends on Ubuntu Linux–specific image processing packages. A VM provides a full Ubuntu environment with guaranteed compatibility, making it possible to access and install and run all the necessary software regardless of the host operating system. Multiple staff members can remotely access and launch identical VMs, ensuring consistency and reproducibility. Additionally, since this model may be more robust, the additional processing power of a separate server could be useful.

Reason against container: While containers provide OS isolation, they rely on the host operating system's kernel. Since the required image processing packages only work on Ubuntu Linux, a full VM is safer to guarantee the software runs as expected and efficiently.

Reason against virtual environment: Too narrow because it only isolates Python packages. It cannot provide Ubuntu Linux or the specialized system-level dependencies needed for the image processing software.

Reason against global environment: Fragile and limiting because it, like the VE, assumes the host computer already runs Ubuntu Linux with the correct libraries. This would make setup inconsistent and not work on most staff members' machines (assuming that most people don't use Linux).

0.3.4 Part d

Sometime in the near future, Python announces the release of Python version 4. In this major update, all commands are now in Swedish, and any command written with English words such as "read" will now fail to run. You are willing to relearn Python in Swedish to use this much needed and long overdue upgrade, after all Python är ett programmeringsspråk som låter dig arbeta snabbt

och integrera system mer effektivt. But you don't want the code you are writing for your current project in Python 3 to stop working once Python 4 is released. [6 points]

Virtual environment

Reason for virtual environment: It allows you to isolate your Python 3 setup from the new Python 4 installation. This way, you can continue running and maintaining your current project in Python 3 while experimenting with Python 4 in a separate environment.

Reason against virtual machine: Overkill for managing two Python versions because duplicating an entire operating system adds unnecessary cost, complexity, and maintenance when all that's needed is version isolation.

Reason against container: Containers are also overkill because even though it would work they add more setup complexity than necessary for simply running different Python versions.

Reason against global environment: Upgrading to Python 4 globally would break all your Python 3 projects. You'd lose the ability to run old and new code without creating a VE for that old code.

0.4   Problem 3

One of the packages we will soon use is `ydata_profiling`, which creates an HTML dashboard from a dataframe for the purpose of exploratory data analysis. It remains one of the very best packages available for making an EDA dashboard, but the problem is this package does not work with Python 3.13:

Tough luck. If you are working in your global environment, now you have to reinstall an older version of Python on your machine and reinstall all of your packages again. There's a good chance that some of these reinstalls will generate more version incompatibility problems. And then you are stuck in a software version purgatory. When you get your system working again, if you can, it's fragile and may buckle and break with any additional new package. That's not a good place to be.

But you are working with virtual environments and dealing with obnoxious package version conflicts is one of the main reasons why virtual environments are so useful.

0.4.1   Part a

Create a new Conda environment that runs Python 3.12, *not* 3.13. Refer to the textbook's discussion of miniconda to help you with the code. Give this environment a name that is different from the one we've been using for class up to this point, and activate the environment on the terminal's command line. Use `conda` to install these packages: `neo4j python-dotenv pandas numpy scipy scikit-learn requests prince ipykernel conda-forge::wquantiles`. (We'll use `neo4j` and `python-dotenv` later in this lab.) Then use `pip` to install `ydata_profiling` (because the package is not available via `conda`). Copy and paste your command-line code into a markdown cell in your notebook.

[8 points]

conda create –name l1_ds6600 python=3.12 pip

conda activate l1_ds6600

conda install neo4j python-dotenv pandas numpy scipy scikit-learn requests prince ipykernel conda-forge::wquantiles

pip install ydata_profiling

0.4.2 Part b

Connect your notebook to this virtual environment as the kernel, and run the following code cell. If you are successful, it will run without error. (You may be prompted to "Upgrade to ydata-sdk". That's the paid version of the `ydata_profiling` package. My rule of thumb is not to pay for anything that can't be bothered to update itself to work on Python 3.13.)

[4 points]

```python
[1]: import numpy as np
     import pandas as pd
     import weighted # this is a module of wquantiles
     from scipy import stats
     import prince
     from ydata_profiling import ProfileReport
```

```
/opt/anaconda3/envs/l1_ds6600/lib/python3.12/site-packages/tqdm/auto.py:21:
TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

```
<IPython.core.display.HTML object>
```

Success!

0.5 Problem 4

The official Python images on Docker Hub use a version of Linux called Debian. However, sometimes you might need to install additional software in a container other than Python and Python packages, and a lot of open source software only works on another version of Linux called Ubuntu.

0.5.1 Part a

Write a Dockerfile that builds an image from the `ubuntu:latest` image on Docker Hub. Run the following Linux commands to install Python 3 onto Ubuntu: `apt-get update` and `apt-get install -y python3`. Once Python is installed, run the following command to launch Python from the command line: `python3`. Copy and paste the Dockerfile here. [8 points]

syntax=docker/dockerfile:1

FROM ubuntu:latest

RUN apt-get update && apt-get install -y python3

CMD ["python3"]

0.5.2 Part b

Prove that the Dockerfile is written correctly by building the image associated with this Dockerfile. Once you are able to build the image successfully, copy and paste the output of the build in your terminal here. [8 points]

I ran the following in the terminal: docker build . -t lab1docker

It had the following output:

[+] Building 11.9s (11/11) FINISHED docker:desktop-linux => [internal] load build definition from Dockerfile 0.0s => => transferring dockerfile: 189B 0.0s => resolve image config for docker-image://docker.io/docker/dockerfile:1 1.4s => [auth] docker/dockerfile:pull token for registry-1.docker.io 0.0s => docker-image://docker.io/docker/dockerfile:1@sha256:dabfc0969b935b2080555ace70ee69a5261af8a8f1b4df97b9e7fbcf 0.7s => => resolve docker.io/docker/dockerfile:1@sha256:dabfc0969b935b2080555ace70ee69a5261af8a8f1b4df97b9 0.0s => => sha256:552fe90f5a266a9c3fe2c5279cc00dc97f85e0ef40628ccfde9c09ddbf7fc1fc 13.25MB / 13.25MB 0.6s => => extracting sha256:552fe90f5a266a9c3fe2c5279cc00dc97f85e0ef40628ccfde9c09ddbf7fc1fc 0.1s => [internal] load metadata for docker.io/library/ubuntu:latest 0.7s => [auth] library/ubuntu:pull token for registry-1.docker.io 0.0s => [internal] load .dockerignore 0.0s => => transferring context: 2B 0.0s => [1/3] FROM docker.io/library/ubuntu:latest@sha256:353675e2a41babd526e2b837d7ec780c2a05bca0164f7ea5dbbd433d21 0.0s => => resolve docker.io/library/ubuntu:latest@sha256:353675e2a41babd526e2b837d7ec780c2a05bca0164f7ea 0.0s => [2/3] RUN apt-get update && apt-get install -y python3 5.9s => [3/3] WORKDIR /DS6600_lab1 0.0s => exporting to image 3.0s => => exporting layers 2.5s => => exporting manifest sha256:a8b2ac87d80ea0d50b66a8edf693c8762bffecb15483fdab86208a3039d3c10a 0.0s => => exporting config sha256:e3dd22162fbbba285e5c6f8ed3efb1a93c65eca3bfc02c205a8edb30211b1497 0.0s => => exporting attestation manifest sha256:836017b14134092bc2018ffe4aa27aa3304e21623b35a0a524f7ee923edc0 0.0s => => exporting manifest list sha256:dfb95ce27fe9c5381dd4bdd9c4829c9c92990527bc730ec9d8542080384c203 0.0s => => naming to docker.io/library/lab1docker:latest 0.0s => => unpacking to docker.io/library/lab1docker:latest 0.5s (l1_ds6600) stephanie@Daniels-MacBook-Pro-2 DS6600_lab1 %

0.5.3 Part c

We've mostly run Docker containers by using the `-p` tag to map the operations of a container to a port on our local machine. But another way to run a container is in "interactive mode" using the `-it` tag, which immediately replaces your command line with the command line that exists inside the container. If you've correctly specified your Dockerfile to both install and launch Python inside the container, running this container in interactive mode should result in showing you a Python command line. Type `docker run -it` followed by the name of the image you created in parts (a) and (b). Confirm that the Python prompt appears. Then copy-and-paste the output from your terminal from the `docker run` command here. [4 points]

I ran the command: docker run -it lab1docker

The Python prompt appeared. I got the following output in my terminal:

Python 3.12.3 (main, Aug 14 2025, 17:47:21) [GCC 13.3.0] on linux Type "help", "copyright", "credits" or "license" for more information. »>

0.6 Problem 5

Docker is for more than databases and Python. It can be used to deploy any kind of software on any operating system. For example, NetHack is a classic text-based adventure video game. It was first released in the 1980s.

Search on Docker Hub for NetHack. Find the Docker image for running NetHack that was posted by the user "matsuu". Then find a way to run a container on your computer from this image. If you are successful, then after answering the first few questions you will receive a passage from a sacred book, scroll, or text. Copy the passage and paste it here. [8 points]

I ran the following in my terminal:

docker run -it matsuu/nethack

After going through a few prompts, I was given this passage:

It is written in the Book of The Lady:

```
        After the Creation, the cruel god Moloch rebelled
        against the authority of Marduk the Creator.
        Moloch stole from Marduk the most powerful of all
        the artifacts of the gods, the Amulet of Yendor,
        and he hid it in the dark cavities of Gehennom, the
        Under World, where he now lurks, and bides his time.

    Your goddess The Lady seeks to possess the Amulet, and with it
    to gain deserved ascendance over the other gods.

    You, a newly trained Rambler, have been heralded
```

—+–.– from birth as the instrument of The Lady. You are destined |….d… to recover the Amulet for your deity, or die in the |…..@.| attempt. Your hour of destiny has come. For the sake |…….. of us all: Go bravely with The Lady! |……o| –More–

0.7 Problem 6

We've discussed connecting to MySQL, PostgreSQL, and Mongo using Docker, but there are many different kinds of database systems for different situations, and as a data scientist who can hang with the data engineers, you will need to be able to set up a database from a system you haven't used before. For this problem, use your Docker skills and your navigation of available documentation to create a local Python connection to a graph database running on Neo4j. See this [AWS blog[(https://aws.amazon.com/compare/the-difference-between-graph-and-relational-database/)] or the textbook for a deeper discussion of graph databases.

0.7.1 Part a

Make sure the conda environment you created in problem 3 is selected as the kernel for this notebook, then run the following import statements. [4 points]

```python
from neo4j import GraphDatabase
import dotenv
```

```
import os
```

0.7.2 Part b

Use the documentation listed on the [Docker Hub page for the official Neo4j Docker image](#), and the Neo4j documentation linked there, to determine:

- the default port the image runs on (there are two in this case, one called "bolt" is used for connecting to the database itself, and the other is used for an HTML dashboard with a user interface for working with the dashboard if you want to use a UI),

- the folder inside the container that stores the data (feel free to ignore examples of folders outside the container as we can use `volumes` to manage that),

- and the environmental variables required by the Neo4j image. Make sure you specify a password, and don't disable authentication with `--env=NEO4J_AUTH=none`. (You will need to use the Neo4j documentation outside of Docker Hub. Take a look at the "Getting started with Neo4j in Docker" page.)

List your answers, and how/where you found those answers. [8 points]

- the default port the image runs on: **7474 and 7687**
  - Found under "How to use this image" in the documentation for neo4j onf docker hub in this sentence: "This binds two ports (7474 and 7687) for HTTP and Bolt access to the Neo4j API."
- the folder inside the container that stores the data: **/data**
  - Found under "How to use this image" in the documentation for neo4j onf docker hub in this sentence: "TA volume is bound to /data to allow the database to be persisted outside the container."
- the environmental variables required by the Neo4j image: **NEO4J_AUTH=neo4j/your_password**(I am not including a password in this document for security reasons)
  - Found in the Neo4j documentation outside of docker hub which can be found at this link: https://neo4j.com/docs/operations-manual/current/docker/introduction/ . It was located on the "Getting started with Neo4j in Docker" page under "Using NEO4J_AUTH to set an initial password"

0.7.3 Part c

Create a compose.yaml file that launches a Neo4j container from the Neo4j official image on Docker Hub. In the compose.yaml file: attach the ports that are needed, load the necessary environmental variables via a .env file, and create a local volume named "neo4jdata" and map it to the data directory inside the container. Copy and paste your compose.yaml code into a Markdown cell in this notebook. [8 points]

services: neo4jdata: image: neo4j:latest env_file: .env ports: - "7474:7474" - "7687:7687" volumes: - neo4jdata:/data networks: - dbnetwork

volumes: neo4jdata:

networks: dbnetwork:

```
[3]: """So you can see the indentation properly I am also pasting
     the code in as a comment below

     services:
       neo4jdata:
         image: neo4j:latest
         env_file: .env
         ports:
           - "7474:7474"
           - "7687:7687"
         volumes:
           - neo4jdata:/data
         networks:
           - dbnetwork

     volumes:
       neo4jdata:

     networks:
       dbnetwork:


     """
```

```
[3]: 'So you can see the indentation properly I am also pasting\nthe code in as a
     comment below\n\nservices:\n  neo4jdata:\n    image: neo4j:latest\n    env_file:
     .env\n    ports:\n      - "7474:7474"\n      - "7687:7687"\n    volumes:\n
     - neo4jdata:/data\n    networks:\n      - dbnetwork\n\nvolumes:\n
     neo4jdata:\n\nnetworks:\n  dbnetwork: \n\n\n'
```

0.7.4 Part d

Type `docker compose up` on the command line to launch the Neo4j container. Then run the following Python code. If your container is launched, the following code will display the phrase "Connection to Neo4j established successfully", you've succeeded in getting Neo4j to run on your system, and you are a rockstar data engineer. [8 points]

```
[4]: dotenv.load_dotenv()
     NEO4J_AUTH = os.getenv('NEO4J_AUTH').split("/")

     URI = "bolt://localhost:7687"
     USERNAME = NEO4J_AUTH[0]
     PASSWORD = NEO4J_AUTH[1]

     try:
         # Create a Driver instance
         # This only provides connection information, it does not establish a␣
      ↪connection yet.
```

```python
    driver = GraphDatabase.driver(URI, auth=(USERNAME, PASSWORD))

    # Verify connectivity immediately
    # This forces the driver to create a connection and check credentials/
↪compatibility.
    driver.verify_connectivity()
    print("Connection to Neo4j established successfully.")

except Exception as e:
    print(f"Failed to connect to Neo4j: {e}")

finally:
    # Close the driver to release resources
    if 'driver' in locals() and driver:
        driver.close()
```

```
Connection to Neo4j established successfully.
```