Problem 1. https://github.com/Anny-tech/DS6600_lab1/

Problem 2.

a. '''Using containers. I think packaging a web portal + mobile backend so any chapter can run it the same way with its own DB would be a good idea in this case. Containers give repeatable deploys, easy scaling, and per-chapter isolation with volumes for persistent storage and standard network ports. A virtual environment only manages Python deps (not services/DBs), the global environment is brittle and hard to reproduce, and a full VM per chapter is heavier to build/maintain than a containerized stack.'''

b. '''Using a virtual environment. This would be a one-off data-cleaning/report task on the machine. A virtual environment (e.g., conda/venv) cleanly pins Python packages without touching system Python. A container or VM would work but is overkill, and the global environment risks version conflicts and dependency drift.'''

c. '''Using containers (Ubuntu base image). The type of model would depend on Ubuntu-only image libs and must run on multiple teammates' machines. Ship a Docker image FROM ubuntu with all system packages + Python so everyone runs the same stack anywhere. A virtual environment can't supply Ubuntu system libs, global is fragile/inconsistent across laptops, and a VM provides Ubuntu but with unnecessary overhead compared to a container.'''

d. '''Using virtual environments (pin Python 3). Keeping the current project in a venv that targets Python 3 while experimenting with Python 4 in a separate venv. This would prevent breakage when Python 4 lands. Containers or a VM would isolate too, but are heavier than needed; A global environment would risk upgrading to Python 4 and breaking the project.'''

Problem 3.

a. '''i. Create a fresh env on Python 3.12 and activate it conda create -n ds6600_lab1 python=3.12 -y conda activate ds6600_lab1

ii. Install the required packages with conda (wquantiles specifically from conda-forge as required) conda install -y neo4j python-dotenv pandas numpy scipy scikit-learn requests prince ipykernel conda-forge::wquantiles

iii. Install ydata-profiling via pip (not available via conda) pip install ydata-profiling '''

b. '''python -m ipykernel install --user --name ds6600_lab1 --display-name "Python_ds6600_lab1" '''

In [1]:

```python
import numpy as np
import pandas as pd
import weighted  # module from wquantiles
from scipy import stats
import prince
from ydata_profiling import ProfileReport
```

Upgrade to ydata-sdk

Improve your data and profiling with ydata-sdk, featuring data quality scoring, redundancy detection, outlier identification, text validation, and synthetic data generation.

Problem 4.

a.

```dockerfile
# syntax=docker/dockerfile:1
# Problem 4: Ubuntu-based image that installs Python 3 and launches it by default

FROM ubuntu:latest

# Avoid interactive prompts from apt
ENV DEBIAN_FRONTEND=noninteractive

# Install Python 3
RUN apt-get update && \
    apt-get install -y --no-install-recommends python3 && \
    rm -rf /var/lib/apt/lists/*

# Start a Python REPL when the container runs
CMD ["python3"]
```

b.

~~~~
docker build -t ds6600_lab1:ubuntu-python .
[+] Building 18.0s (8/8) FINISHED
docker:default
 => [internal] load build definition from Dockerfile
0.0s
 => => transferring dockerfile: 441B
0.0s
 => resolve image config for docker-image://docker.io/docker/dockerfile:1
0.3s
 => CACHED docker-image://docker.io/docker/
dockerfile:1@sha256:dabfc0969b935b2080555ace70ee69a5261af8a8f1b4df97b9e7fbcf6722eddf
0.0s
 => [internal] load metadata for docker.io/library/ubuntu:latest
0.4s
 => [internal] load .dockerignore
0.0s
 => => transferring context: 2B
0.0s
 => [1/2] FROM docker.io/library/
ubuntu:latest@sha256:353675e2a41babd526e2b837d7ec780c2a05bca0164f7ea5dbbd433d21d166fc
2.6s
 => => resolve docker.io/library/
ubuntu:latest@sha256:353675e2a41babd526e2b837d7ec780c2a05bca0164f7ea5dbbd433d21d166fc
0.0s
 => => sha256:353675e2a41babd526e2b837d7ec780c2a05bca0164f7ea5dbbd433d21d166fc
6.69kB / 6.69kB
0.0s
 => => sha256:985be7c735afdf6f18aaa122c23f87d989c30bba4e9aa24c8278912aac339a8d 424B
/ 424B
0.0s
 => => sha256:6d79abd4c96299aa91f5a4a46551042407568a3858b00ab460f4ba430984f62c
2.30kB / 2.30kB
0.0s
 => => sha256:953cdd4133718b72c5d0a78e754c1405c02510fdb5237265f7955863f1757f83
29.72MB / 29.72MB
1.1s
 => => extracting
sha256:953cdd4133718b72c5d0a78e754c1405c02510fdb5237265f7955863f1757f83
1.3s
 => [2/2] RUN apt-get update &&     apt-get install -y --no-install-recommends
python3 &&     rm -rf /var/lib/apt/lists/*
14.0s
 => exporting to image
0.5s
 => => exporting layers
0.5s
 => => writing image
sha256:a7fdd40383337dac6828e59240112f2a6689b21f2c57d6f772219b9db5aa2ac0
0.0s
 => => naming to docker.io/library/ds6600_lab1:ubuntu-python
``` <br>

c.
```bash
docker run -it --rm ds6600_lab1:ubuntu-python
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.
>>> import sys
>>> print(sys.version)
3.12.3 (main, Aug 14 2025, 17:47:21) [GCC 13.3.0]
~~~~

```
>>> exit()
```

Problem 5.

```
It is written in the Book of Amaterasu Omikami:

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 Amaterasu Omikami seeks to possess the Amulet, and with it
to gain deserved ascendance over the other gods.
                                                               - ---------
                                                               |..x......|
You, a newly trained Hatamoto, have been heralded
                                                               |.......@d|
from birth as the instrument of Amaterasu Omikami.  You are destined
                                                               |.........|
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 Amaterasu Omikami!
--More--



Ankita the Hatamoto           St:18 Dx:14 Co:18 In:10 Wi:8 Ch:9 Lawful
Dlvl:1 $:0 HP:15(15) Pw:2(2) AC:4 Xp:1
```

In [2]:

```python
#Problem 6. a.
from neo4j import GraphDatabase
import dotenv
import os
```

Problem 6 b.

Default ports: 7687 = Bolt (driver/database connection); 7474 = HTTP (browser/UI) Documented in the official image examples and docs. (Source: `html https://github.com/neo4j/docker-neo4j`)

Data directory in the container: /data Neo4j inside Docker stores database files in /data; mount a volume there to persist data. (Source: `html https://neo4j.com/docs/operations-manual/current/docker/mounting-volumes/`)

Required environment variable: NEO4J_AUTH=neo4j/Jayangdehi_2025 This sets the database password at container start. (Source: `html https://neo4j.com/docs/operations-manual/current/docker/introduction/`)

Problem 6 c. compose.yaml looks like:

```
  neo4j:
    image: neo4j:5
    container_name: lab1_neo4j
    ports:
      - "7474:7474"    # HTTP UI
      - "7687:7687"    # Bolt
    env_file:
      - .env
    volumes:
      - neo4jdata:/data
    restart: unless-stopped

volumes:
  neo4jdata:
```

Problem 6 d.

```
docker compose up -d
```

```
[+] Running 7/7
 ✔ neo4j Pulled
16.8s
   ✔ 456a3213e1b1 Pull complete
13.9s
   ✔ f4338095e996 Pull complete
14.8s
   ✔ 7622963ce7f6 Pull complete
14.8s
   ✔ 98a88966722a Pull complete
14.8s
   ✔ 529c897a79bb Pull complete
15.6s
   ✔ 4f4fb700ef54 Pull complete
15.6s
[+] Running 3/3
 ✔ Network ds6600_lab1_default     Created
0.0s
 ✔ Volume "ds6600_lab1_neo4jdata"  Created
0.0s
 ✔ Container lab1_neo4j            Started
```

In [3]:

```python
#Problem 6 part d
from neo4j import GraphDatabase
import dotenv, os

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 (no connection yet)
    driver = GraphDatabase.driver(URI, auth=(USERNAME, PASSWORD))
    # Force a connectivity check now
    driver.verify_connectivity()
    print("Connection to Neo4j established successfully.")
except Exception as e:
    print(f"Failed to connect to Neo4j: {e}")
finally:
    if 'driver' in locals() and driver:
        driver.close()
```

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

In [ ]: