Software Development Environment Lab Exercises

Development Environment Setup Planning

1. Choose one development scenario:

Building a data analysis project, Analyze sales data and create charts and reports

2. Environment Planning:
Which type of IDE would be most suitable: PyCharm; Python development: Python have many good libraries for data work. For example, NumPy help with numbers and math, Pandas help to clean data and work with tables like CSV or Excel, Matplotlib and Seaborn make graphs and pictures, and Scikit-learn for machine learning.

Additional tool:
1. Pylint:
Analyze code to detect stylistic and programming errors. They enforce coding standards and help identify potential bugs early.
2. Black: Automatically format code according to predefined style guidelines, improving readability and consistency.
3. PyTest: Facilitate automated testing to ensure code correctness and maintainability. Provide structures for writing and executing tests.
4. Github: Support collaborative code review processes, where team members can review, comment on, and approve code changes.
5. Docker: Ensures consistent environments across multiple developers or machines.

Hardware Requirement:
Minimum: Dual-core CPU, 8 GB RAM, 10 GB free disk, stable internet.
Recommended: Quad-core CPU, 16 GB RAM, SSD, second monitor (for charts + code).
Notes emphasize IDEs and containerized environments can be resource-intensive.

Team Collaboration:
Data Engineer A: Data cleaning, data processing
Data Engineer B: Data Visualization, data reporting

Set up time and Complexity:
Install IDE PyCharm ~20 minutes.
Install Python + packages (NumPy, Pandas, Matplotlib, Seaborn) ~15 minutes.
Configure GitHub repo ~10 minutes.
Containerized setup Docker ~30 minutes.
Total: 1–1.5 hours.
Complexity: Low–moderate, since Python environments are simpler than compiled languages.

Setup Guide Creation

Step 1 — Install prerequisites

1. Install Docker Desktop from docker.com.
2. Install Git from git-scm.com.
3. Install PyCharm to open the project. Download PyCharm:
https://www.jetbrains.com/pycharm/download

Step 2 — Get the project

1. Create a folder called **sales-data-analysis**.
2. Inside that folder, make a file called **requirements.txt** and put this in it:

```
numpy
pandas
matplotlib
seaborn
scikit-learn
jupyterlab
```

NumPy + Pandas = core data handling
Matplotlib + Seaborn = visualization
Scikit-learn = advanced analysis (optional, but good to have)
JupyterLab = the workspace to tie everything together

Step 2.5 — Initialize Git & push to GitHub

Create a repo on GitHub (e.g., sales-data-analysis).
In your project folder, run:

```
git init
git add .
git commit -m "Initial commit"
git branch -M main
git remote add origin
https://github.com/<your-username>/sales-data-analysis.git
git push -u origin main
```

Version control → every change is saved, you can roll back if needed.
Collaboration → your teammate can clone the repo and run the same Docker setup.
Sync with Docker → when you or your teammate update requirements.txt, rebuilding the image ensures everyone has the same environment.

Step 3 — Create the Dockerfile

In the same folder, create a file called **Dockerfile** (no extension) and copy this:

```
FROM python:3.11

WORKDIR /app

COPY requirements.txt .
RUN pip install -r requirements.txt

CMD ["jupyter", "lab", "--ip=0.0.0.0", "--allow-root", "--no-browser"]
```

Get Python → move to /app → bring package list → install packages → auto-run JupyterLab.

Step 4 — Build the image

Open a terminal (Command Prompt, PowerShell, or console), go to your project folder, and run:

```
docker build -t data-lab .
```

docker build → tells Docker to build an image.
-t data-lab → names the image data-lab (you can choose any name).
. (dot) → means "use the current folder" (where your Dockerfile is).

Step 5
— Run the container

```
docker run -it -p 8888:8888 -v %cd%:/app data-lab   # Windows
```

`-p 8888:8888` → makes Jupyter available in your browser.
`-v ...:/app` → shares your folder with the container so notebooks save to your computer.

Step 6 — Open JupyterLab

1. After the command runs, you'll see a link like:
http://127.0.0.1:8888/lab
2. Open it in your browser.
3. You can now create notebooks (.ipynb files) and start analyzing data.