📝 Git Exercise: Retrieve & Analyze Big Data with Git History

🔧 Scenario

Welcome to your new job! You've inherited a messy repository from a team that didn't believe in consistency. Your boss gave you a massive data file (`Lorem ipsum.txt`) that was added and then mysteriously deleted. Meanwhile, your colleague created a great analysis script in another branch — but never merged it.

Now it's your mission to:

1. Retrieve the missing data file from Git history.
2. Recover the analysis script from a different branch.
3. Make it all work together in `main.py`.
4. Add the large file to `.gitignore` so it doesn't clutter the repo again.
5. Commit and push the changes.

🚀 Your Mission

1. Create the repository from the script

```
chmod +x E4_repository.sh
./E4_repository.sh
```

---

2. Investigate the Git History

View the commit history to find where the boss's file was added:

```
git log
```

---

3. Recover the Big File

Find the commit where `Lorem ipsum.txt` was added. Use that commit hash to restore it:

```
git checkout <commit-hash> -- "Lorem ipsum.txt"
```

Confirm the file is back:

```
ls -lh "Lorem ipsum.txt"
```

---

4. Retrieve the Analysis Script from a Branch

Switch to the branch containing the analysis tool:

```
git checkout READ_LOREM_IPSUM
```

Copy the file into the current `main` branch:

```
git checkout main
git checkout READ_LOREM_IPSUM -- read_lorem.py
```

---

5. Create `main.py` to Run the Analysis

Now create a new file `main.py` and add this content:

```
from read_lorem import analyze_text

if __name__ == "__main__":
    analyze_text("Lorem ipsum.txt")
```

Save the file and stage the changes:

```
git add main.py
git commit -m "Add main.py to run lorem analysis"
```

---

6. Add the Large File to `.gitignore`

To avoid committing the big file again, add it to `.gitignore`:

```
echo "Lorem ipsum.txt" >> .gitignore
git add .gitignore
git commit -m "Ignore big data file from future commits"
```

---

7. Test Your Work

Run the script to make sure it works:

```
python main.py
```

✅ What You'll Learn

- 🔍 Navigating Git history and branches
- ♻️ Recovering deleted files from the past
- 🛠️ Merging useful work from isolated branches
- 🚫 Preventing large files from bloating the repo
- 🧠 Real-world Git workflows for data analysis

🎯 Bonus Challenge

Can you automate part of this recovery process with a shell or Python script?

You're now ready to handle messy repositories like a Git pro. Good luck! 🚀