Project Evaluation Checklist: HR Analytics Dashboard (Week 2)

Use this to evaluate and complete your work before the final deadline.

Project Foundation & Data Handling 🏗️

- [ ] **Project Structure**: The repository has a clean, standard folder structure (e.g., `/data` , `/notebooks` , `/src`).
- [ ] **Environment Management**:
    - [ ] An `environment.yml` (for Conda) and/or `requirements.txt` (for pip) file is included.
    - [ ] **Crucially**, the file must be portable and contain **no local file paths**.
- [ ] **Data Ingestion & Database**:
    - [ ] The correct Kaggle CSV file is loaded into a Pandas DataFrame.
    - [ ] A local SQLite database is created, and the data is programmatically inserted into an `employees` table.

Analysis & Insight Generation 📊

- [ ] **Jupyter Notebook**: The analysis is presented in a well-documented Jupyter Notebook with clear markdown explanations for each step, query, and conclusion.
- [ ] **Required Analysis**: All five specified business questions are answered accurately with supporting code and visualizations.
- [ ] **Custom Analysis**:
    - [ ] At least **three** additional, insightful business questions are defined and explored.
    - [ ] The analysis demonstrates a deeper understanding of the data, going beyond simple aggregations.

Application Development 💻

- [ ] **Web Framework**: A functional, single-page web app is built using a modern framework (e.g., **Streamlit, Plotly Dash, or an alternative**).
- [ ] **Core Features**:
    - [ ] **Visualizations**: At least **three** distinct, clearly-labeled, and relevant data visualizations are displayed.
    - [ ] **Interactivity**: An interactive filter works correctly, updating the view dynamically.
    - [ ] **CRUD Operations**: The features to **Create** a new employee and **Update** an existing employee's income are fully functional and robust.
- [ ] **User Experience (UX)**: The UI is intuitive, aesthetically pleasing, and includes professional branding elements (title, consistent theme).

.
.
.
.

Code Quality & Version Control ✅

- [ ] **Code Readability & Standards**: Python code is clean, and well-commented.
- [ ] **Git Best Practices**:
  - [ ] A comprehensive `.gitignore` file is present to exclude unnecessary files.
  - [ ] The project history shows **multiple, atomic commits** with clear, descriptive messages that tell a story of the project's development. (No single "Initial commit" with all files).

Final Submission & The Perfect Repository 🚀

- [ ] **Repository Contents**: All required files are present and correctly placed: `app.py`, analysis notebook, `.db` file, `README.md`, `LICENSE`, `.gitignore`, and environment file(s).
- [ ] **Repository Metadata**: The GitHub repository includes a concise description and relevant topics/tags (e.g., `python`, `data-analysis`, `streamlit`, `hr-analytics`).
- [ ] **License**: A `LICENSE` file is included, specifying an open-source license (e.g., MIT).

Sub-Checklist: The Perfect `README.md` File

- [ ] **Header**:
  - [ ] **Title**: A clear, prominent project title.
  - [ ] **Badges**: Include relevant badges for licenses or technologies.
  - [ ] **Pitch**: A compelling one-sentence summary of the project.
- [ ] **Table of Contents**: A navigable table of contents for easy reading.
- [ ] **Visual Demo**: A high-quality screenshot or an animated GIF showcasing the interactive dashboard.
- [ ] **Project Overview**: A detailed section explaining the project's purpose, the problem it solves, and the value it provides.
- [ ] **Data Source & Dictionary**:
  - [ ] Clearly state where the data was obtained, with a direct link to the source.
  - [ ] Provide a brief data dictionary explaining some of the most important columns.
- [ ] **Technology Stack**: A list of key technologies and libraries used.

.
.
.
.
.
.
.
.
.
.

- [ ] **Setup and Local Installation**:
    - [ ] Provide clear, step-by-step instructions for getting the project running locally.
    - [ ] All commands must be placed in **markdown code blocks** for easy copy-pasting.

    ```
    # Example: Clone the repository
    git clone [https://github.com/your-username/your-repo.git](https://github.com/your-username/your-repo.git)
    cd your-repo

    # Example: Create and activate the Conda environment
    conda env create -f environment.yml
    conda activate your_env_name
    ```

- [ ] **Usage**: A simple command in a **markdown code block** to launch the application (e.g., `streamlit run app.py`).
- [ ] **Author & Acknowledgments**: A section with your name and credit to the data provider.

Optional Enhancements & Self-Study ✨

(These are not required but demonstrate exceptional work)

- [ ] **(Self-Study) Advanced Visualizations**: Use **Seaborn** or **Plotly Express** for more sophisticated charts.
- [ ] **(Self-Study) Advanced SQL**: Refactor a complex query to use a **Common Table Expression (CTE)**.
- [ ] **(Self-Study) Function Annotations**: Use Python's **type hints** to improve code clarity.
- [ ] **(Self-Study) Code Organization**: For long scripts, use "super comments" to create logical, navigable sections.

```
# ──────────────────────────────────
# DATA LOADING AND PREPROCESSING
# ──────────────────────────────────
# ... code ...

# ──────────────────────────────────
# UI LAYOUT AND COMPONENTS
# ──────────────────────────────────
# ... code ...
```