Placement Eligibility Streamlit Application

Introduction

In today's EdTech space, evaluating student readiness for placements isn't just about test scores, it's about combining performance in programming, communication, teamwork, and mock interviews to make informed decisions.

As part of my GUVI + IIT mini-project, I built a **Placement Eligibility Web Application** using Python, SQL, and Streamlit. This project simulates how a real-world placement cell might function and makes it easier to filter students based on key placement criteria.

Placement Eligibility Streamlit Application

Live App: [Click to View the App]
(https://placement-eligibility-app-tykdjyd9aqni5nn5jihdsu.streamlit.app/)

Why I Built This App

I wanted to bring together everything I've been learning in data science — from data generation and SQL queries to creating user-friendly dashboards. The goal was to build an **end-to-end system** where:

- **Placement officers** can filter students based on customized cutoffs
- **Educators or mentors** can track student readiness using key metrics
- And even **students** can see where they stand and how they can improve

Core Features of the App

1. **Filter Eligible Students**
   Users can select:
   - Minimum number of programming problems solved
   - Average soft skills score (calculated from 6 soft skill traits)
   - Minimum mock interview score
2. Based on these filters, the app shows a clean, downloadable list of students who match all criteria.

3. **Table Viewer**
   The app includes a simple viewer where users can choose from:
   - `Students`, `Programming`, `SoftSkills`, or `Placements` tables
     and explore the data directly from the database.

4. **Insights Dashboard**
   This was one of my favorite parts to build! I loaded **pre-written SQL queries** from a file and displayed the results as both tables and charts.
   Some insights include:
   - Average mock scores by enrollment year
   - Soft skill performance by batch
   - Placement status breakdowns
   - Top performers in mock interviews
   - Package stats, certification comparisons, and more

What's Happening Behind the Scenes

To make everything run smoothly, I used:

- **SQLite** for the backend database
- **Python's OOP** to create a reusable `DatabaseManager` class
- **Pandas** for handling data and visualizations
- **Faker** to generate 100+ realistic fake student records

I also separated my code into clean modules:

- `data_generator.py` – For creating the data
- `db_manager.py` – For all database functions (connect, run queries, etc.)
- `streamlit_app.py` – The main file running the Streamlit interface
- `insights.sql` – A collection of 10+ SQL queries used in the dashboard

Technologies I Used

- **Python 3.8+**
- **Streamlit** – for interactive UI
- **SQLite** – as the relational database

- **Pandas** – for working with tabular data
- **Faker** – to simulate realistic student profiles
- **Matplotlib/Streamlit Charts** – for visualizations
- **VS Code + GitHub** – for coding and version control

Challenges I Faced

- Designing realistic filters that match real placement criteria
- Creating SQL queries that calculate averages across multiple columns
- Managing errors while dynamically displaying 10 different queries and charts
- Making sure everything loads fast and displays neatly, even with 100+ records

What I Learned

- Writing **clean, modular, and beginner-friendly Python code**
- Managing a relational database using SQL and SQLite
- Dynamically loading and displaying SQL queries in Streamlit
- Combining **multiple filters and user input** to shape queries
- Building a Streamlit app that's easy to navigate and intuitive for users

What's Next

If I continue this project, I'd love to add:

- Authentication for users (students vs placement officers)
- Editable forms to update student data
- More advanced visualizations using Plotly
- A resume ranking or feedback system based on placement readiness