**PLACEMENT ELIGIBILITY STREAMLIT APPLICATION**

**Introduction:**

Placement Eligibility Streamlit Application is a data-driven web application designed to assist institutions or training platforms in identifying students who are eligible for placement opportunities.

This project demonstrates:

* Integration of MySQL for backend data storage
* Use of Python's OOP concepts for generating and managing synthetic student data using Faker
* Customizable filters to check eligibility based on coding skills, soft skills, and mock interview performance
* A second tab for placement insights and analytics including charts and summaries
* Export and save options to download filtered student lists
* An interactive frontend built with Streamlit

The application simulates real-world student assessment data and provides actionable insights that can aid in streamlining the placement process.

**Problem Statement:**

Design and implement a Streamlit application where users can input eligibility criteria for placement. Based on these criteria, the application should query a dataset of student information to display eligible candidates' details.

**Approach:**

**Step 1**: **Dataset Creation**

* Generate four related tables using the [**Faker**](https://faker.readthedocs.io/en/master/) **library**:

**Students Table**

This table stores basic information about students enrolled in the course.

* **student\_id (Primary Key)**: Unique identifier for each student.
* **name**: Full name of the student.
* **age**: Age of the student.
* **gender**: Gender of the student (e.g., Male, Female, Other).
* **email**: Email address of the student.
* **phone**: Contact number of the student.
* **enrollment\_year**: Year when the student enrolled in the course.
* **course\_batch**: Name of the batch or cohort the student belongs to.
* **city**: City of residence for the student.
* **graduation\_year**: Expected or actual graduation year for the student.

**Link :** [**https://d.docs.live.net/4f1ffd180cf081a1/Desktop/placement\_eligibility\_app/students.csv**](https://d.docs.live.net/4f1ffd180cf081a1/Desktop/placement_eligibility_app/students.csv)

**Programming Table**

This table stores details of students' programming performance in the course.

* **programming\_id (Primary Key)**: Unique identifier for each programming record.
* **student\_id (Foreign Key)**: References the student\_id in the **Students Table**.
* **language**: Programming language being evaluated (e.g., Python, SQL).
* **problems\_solved**: Total number of coding problems solved by the student.
* **assessments\_completed**: Number of assessments completed by the student.
* **mini\_projects**: Number of mini projects submitted by the student.
* **certifications\_earned**: Number of programming certifications earned by the student.
* **latest\_project\_score**: Score received in the most recent programming project.

**Link** : <https://d.docs.live.net/4f1ffd180cf081a1/Desktop/placement_eligibility_app/programming.csv>

**Soft Skills Table**

This table stores data on students' performance in soft skills evaluations.

* **soft\_skill\_id (Primary Key)**: Unique identifier for each soft skill record.
* **student\_id (Foreign Key)**: References the student\_id in the **Students Table**.
* **communication**: Communication skills score (out of 100).
* **teamwork**: Teamwork skills score (out of 100).
* **presentation**: Presentation skills score (out of 100).
* **leadership**: Leadership skills score (out of 100).
* **critical\_thinking**: Critical thinking skills score (out of 100).
* **interpersonal\_skills**: Interpersonal skills score (out of 100).

**Link :** [**https://d.docs.live.net/4f1ffd180cf081a1/Desktop/placement\_eligibility\_app/soft\_skills.csv**](https://d.docs.live.net/4f1ffd180cf081a1/Desktop/placement_eligibility_app/soft_skills.csv)

**Placements Table**

This table stores details related to students’ placement readiness and outcomes.

* **placement\_id (Primary Key)**: Unique identifier for each placement record.
* **student\_id (Foreign Key)**: References the student\_id in the **Students Table**.
* **mock\_interview\_score**: Score in the mock interviews (out of 100).
* **internships\_completed**: Total number of internships completed by the student.
* **placement\_status**: Placement readiness status (e.g., Ready, Not Ready, Placed).
* **company\_name**: Name of the company where the student got placed (if applicable).
* **placement\_package**: Package offered during placement (in USD or local currency).
* **interview\_rounds\_cleared**: Number of interview rounds cleared by the student.
* **placement\_date**: Date when the placement offer was received.

**Link :** [**https://d.docs.live.net/4f1ffd180cf081a1/Desktop/placement\_eligibility\_app/placements.csv**](https://d.docs.live.net/4f1ffd180cf081a1/Desktop/placement_eligibility_app/placements.csv)

**Step 2: Data Storage**

* + Store the generated data in a relational database (e.g., SQLite, MySQL).
  + Implement Python OOP for database interactions**.**

In this project, all student-related data is systematically stored and managed using MySQL, a powerful relational database management system (RDBMS). The data is organized across multiple normalized tables to support efficient querying and analysis.

**Database Used:**

* Name: placement\_db
* Type: Relational (MySQL)
* Tool: MySQL Server (via mysql-connector-python in Python)

**Tables and Their Purpose:**

| Table Name | Description |
| --- | --- |
| students | Stores basic student information such as name, age, gender, batch, etc. |
| programming | Contains programming skill metrics like language known, problems solved, etc. |
| soft\_skills | Captures soft skill ratings like communication, teamwork, leadership, etc. |
| placements | Stores placement status, company name, package offered, and interview rounds. |

**Data Generation:**

* Data is generated using the Faker library and populated into MySQL using an Object-Oriented Programming (OOP) approach.
* Four separate classes handle generation for each table:
  + StudentGenerator
  + ProgrammingGenerator
  + SoftSkillsGenerator
  + PlacementsGenerator

**Storage Location:**

* Backend: All data is stored locally in MySQL (localhost).
* Frontend: Data is accessed dynamically in the Streamlit web interface using SQL queries and presented using Pandas DataFrames and visualizations.

**Link :** [**https://d.docs.live.net/4f1ffd180cf081a1/Desktop/placement\_eligibility\_app/data\_generator\_mysql.py**](https://d.docs.live.net/4f1ffd180cf081a1/Desktop/placement_eligibility_app/data_generator_mysql.py)

**Step 3: Streamlit Application :**

* + Create a user-friendly interface where users can:
    - Input eligibility criteria (e.g., problems solved > 50, soft skills score > 75).
    - View eligible students details dynamically.

The front-end interface of the Placement Eligibility System is built using Streamlit, an open-source Python library used for creating fast and interactive web applications for data projects.

**Purpose of the Streamlit App**

The goal of the Streamlit interface is to:

* Display and filter student eligibility data
* Visualize key insights using charts and tables
* Enable users to download eligible student records
* Provide placement statistics in an interactive format

**Main Features**

The application is structured into multiple tabs:

1. Student Profiles
   * Displays student information from the students table
   * Allows filtering by batch
2. Programming & Skills
   * Combines programming and soft\_skills tables
   * Shows technical and non-technical skill levels for each student
3. Placement Stats
   * Displays placement status, company name, and package from the placements table
   * Shows bar charts for placement insights
4. Eligible Student Export
   * Provides a downloadable CSV file of eligible students based on defined criteria (e.g., score, skills)

**Key Technologies Used**

* Streamlit – Web UI framework
* Pandas – Data manipulation and tabular rendering
* MySQL Connector – To connect Python with MySQL database

**Working Principle**

* On page load, Streamlit connects to the MySQL database
* SQL queries fetch data for each tab dynamically
* Users can interact with filters and visualizations
* DataFrames are displayed in a clean, interactive format

**Export Option**

* The app includes a button to download eligible student data in .csv format for reporting or external use.

**Step 4**: **SQL Queries and Insights :**

* + Write 10 SQL queries to extract meaningful insights, such as:
    - Average programming performance per batch.
    - Top 5 students ready for placement.
    - Distribution of soft skills scores.
  + Incorporate the insights into the Streamlit app.

To analyze and present meaningful information from the database, various **SQL queries** were written and executed within the application. These queries provided deep insights into student performance, skill levels, and placement statistics.

**Purpose of SQL Queries :**

The SQL queries are used to:

* Retrieve and filter student records
* Join multiple tables for cross-analysis
* Compute aggregate statistics
* Provide eligibility results and placement trends

**Link** **:** <https://d.docs.live.net/4f1ffd180cf081a1/Desktop/placement_eligibility_app/app.py>

**Technical Tags :**

Streamlit, Python, Faker, MySQL, Data Science, OOP, Dashboard Development.

**Results :**

A fully functional Streamlit application showcasing dynamic filtering of eligible students , 10 SQL queries providing actionable insights , Clean, modular code using OOP principles has been done and output verified successfully.