**University Database Design and Implementation Using SQLite**

**Name : Giridhar Reddy Goddilla**

**Student ID : 23067661**

**Course : Data Mining and Discovery**

**Degree : MSc Data Science**

**Table of Contents**

1. **Introduction**
2. **Data Generation Process**
3. **Database Schema**
4. **Justification for Tables and Constraints**
5. **Ethical and Privacy Discussion**
6. **Conclusion**
7. **References**

# **Introduction**

Managing and analysing data is important for any university to run efficiently. In this report, I describe how I created a University Database using SQLite. This database stores key information about students, professors, courses, enrolments, attendance, and departments. I designed it using relational database principles to make sure the data is accurate, avoid duplicate entries, and help users quickly find the information they need.

In this report, I will cover:

* The database structure I created to realistically show university activities, clearly connected through foreign and composite keys.
* How I used Python scripts with SQLite and the Faker library to generate realistic data, such as student GPAs and course enrolments.
* The creation of a large dataset with over 1000 student records to help with detailed analysis.
* Clear explanations of the relationships within the database, like students registering for courses, professors teaching, and tracking attendance.
* Ethical issues and privacy considerations taken into account when generating data and building the database to protect confidentiality.
* An evaluation showing how I met all the requirements of the assignment, focusing on realistic data, good database design, and following database design standards.

**A. Data Generation Process**

I made a university database using Python, SQLite, and the Faker library. My aim was to create data that looks real but doesn’t contain anyone's actual personal information. Let me explain how I did it step-by-step:

**1. Tools and Libraries**

**SQLite**: I used this software to build and manage my database.

**Faker**: This library allowed me to create believable student, professor, and department information.

**Random module**: This helped me add realistic variations to the data.

**Steps I Followed:**

1. **Departments:**

* First, I chose five common university departments: Computer Science, Electrical Engineering, Physics, Mathematics, and Chemistry.
* I randomly assigned each department a different city as its location.

1. **Professors:**
   * I created details for 10 professors, giving each one a random first and last name.
   * Then I randomly linked each professor to one of the five departments.

Each professor’s salary was randomly chosen, falling between £50,000 and £120,000 to be realistic.

1. **Courses:**
   * I defined five courses and randomly set their credit hours between 1 and 5.
   * I made sure each course was linked randomly to one of the departments.
2. **Students:**
   * I generated more than 1000 student profiles, making sure each one had unique details.
   * Every student was given random names, birthdates (making them between 18-22 years old), and a major selected from the available departments.
   * GPAs were randomly assigned between 2.5 and 4.0 to keep the data believable.
3. **Course Offerings:**
   * I randomly scheduled courses for Fall, Spring, and summer semesters between 2020 and 2025.
4. **Enrolments and Grades:**
   * Every student got enrolled randomly in three courses.
   * For each enrolment, I gave the student a realistic grade (A, B, C, D, or F).
5. **Attendance:**

I created attendance records by randomly marking each student as either 'Present' or 'Absent' on five different dates.

**Making the Data Realistic:**

* I included variety in GPAs, professor salaries, and course selections to make it feel like a real university.
* I set clear constraints, like GPA ranges from 0.0 to 4.0 and specific credit hours for courses, to maintain logical accuracy.
* By doing this, I ensured the database truly represents realistic scenarios that could happen in a university setting.
  + - This process helped me to understand the practical side of database management and data mining. It also provided a solid and realistic base for analysis, useful for making informed decisions in university management.

**Screenshots of My Database Generation Code**

I used Python together with the Faker library to fill my University Database with realistic and varied data. Below is a summary of my data generation approach for each table.

I uploaded all screenshots of the code used in my report below.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

**B. The schema**

I designed my university database to organize and store information about different parts of university life clearly and efficiently. To do this, I created multiple tables, each focusing on one specific area of university management.

In the Departments table, I stored important information like department IDs, names, and locations. Each department name is unique, so there's no confusion. Professors have their own table too, where I included their names, department details, and salaries. I made sure all salaries were realistic and accurate.

For the Courses table, I assigned each course an ID, name, and credit hours (between 1 and 5) and linked them to their departments. Similarly, the Students table holds student details such as names, birthdates, majors, and GPAs, making sure GPAs stayed between 0.0 and 4.0.

The Course Offerings table helps keep track of which courses are offered each semester—Fall, Spring, or Summer—between 2020 and 2025. Enrolments show which students signed up for each course and their grades, with grades carefully checked to be valid (from A to F).

Lastly, I created the Attendance table to clearly record whether students were Present or Absent on specific dates, helping track their class participation.

Throughout this database, I used primary and foreign keys to link tables together clearly, and I included different types of data (nominal, ordinal, interval, and ratio) as required for the assignment. This helped me avoid duplicate data, ensured accuracy, and made the database easy to manage.

Below are clear screenshots from DB Browser, showing the tables and columns of my university database.

A screenshot of a computer

AI-generated content may be incorrect.

## **Justification for Using Separate Tables and Attribute Constraints**

When designing my university database, I focused on keeping the information well-organized and clear by dividing it into multiple related tables. Each table helps me manage a specific part of the university’s activities, making the database easy to use and avoiding unnecessary repetition.

**Departments:**

* **Purpose:** Stores information about the different departments.
* **Constraints:**
  + **department\_id (Primary Key):** Makes sure every department is unique.
  + **department\_name:** Marked UNIQUE, so no two departments can have the same name.
  + **location:** Clearly identifies where each department is located.
* **Professors:**
* **Purpose:** Keeps track of professor details, including their department and salary.
* **Constraints:**
  + - **professor\_id (Primary Key):** Ensures each professor can be uniquely identified.
    - **department\_id (Foreign Key):** Links each professor to their department.
    - **salary:** CHECK constraint makes sure the salary is realistic and positive.
* **Courses:**
* **Purpose:** Contains details about the courses the university offers.
* **Constraints:**
  + - **course\_id (Primary Key):** Uniquely identifies each course.
    - **department\_id (Foreign Key):** Connects courses to the relevant department.
    - **credit\_hours:** CHECK constraint limits credit hours to between 1 and 5.
* **Students:**
* **Purpose:** Stores student personal and academic details.
* **Constraints:**
* **student\_id (Primary Key):** Each student has a unique identifier.
* **birthdate:** Clearly records each student's birthdate.
* **gpa:** CHECK constraint ensures GPA stays within realistic academic limits (0.0 to 4.0).

**Course Offerings:**

* **Purpose:** Tracks course availability each semester.
* **Constraints:**
* **offering\_id (Primary Key):** Identifies each course offering uniquely.
* **semester:** Limited to valid semesters (Fall, Spring, Summer).
* **year:** Restricted to the range of years from 2020 to 2025.

**Enrolments:**

* **Purpose:** Keeps track of student course enrollments and grades.
* **Constraints:**
  + **student\_id and course\_id (Compound Primary Key):** Ensures that each student-course combination is unique.
  + **grade:** Limited to standard grading options (A, B, C, D, F).

**Attendance:**

* **Purpose:** Records student attendance for specific dates.
* **Constraints:**
  + - **attendance\_id (Primary Key):** Uniquely identifies each attendance record.
    - **status:** Limited to 'Present' or 'Absent' to keep records consistent.

**Ethical and Data Privacy Discussion**

When creating this database, I was careful to handle ethical considerations and privacy issues responsibly:

* **Randomized Data Generation:** I used Python’s Faker library to generate all data randomly, which ensured that no real individuals' information was included.
* **Protection of Privacy:** By using completely fictional data, I made sure no real-world identities or personal information could be compromised.
* **Avoidance of Sensitive Information:** I intentionally avoided including sensitive personal details such as addresses, phone numbers, or financial data, making sure the database respects privacy guidelines and standards.
* **Realistic Yet Ethical Data:** Even though the data is fictional, I designed it to reflect realistic patterns and scenarios, such as varying student GPAs, realistic professor salaries, and typical enrollment distributions.
* **Transparency and Documentation:** I carefully documented every step of the data generation and database design process, clearly stating what kind of data was used and how it was handled. This approach ensures that my database remains transparent, ethically sound, and appropriate for academic and analytical purposes.
* **Compliance with Data Ethics:** By following best practices and ethical guidelines, my database fully supports academic integrity and responsible data use, making it suitable for learning and demonstration without risking the privacy or rights of any individual.

**Conclusions**

In this assignment, I successfully developed a detailed university database using SQLite. I used the Spyder IDE environment and DB Browser for SQLite to manage the database effectively. I carefully created multiple interconnected tables—including Departments, Professors, Courses, Students, Course Offerings, enrolments, and Attendance—to ensure organized and efficient data handling. Python, along with the Faker library, enabled me to generate realistic and randomized data that covered all required data types: nominal, ordinal, interval, and ratio. Throughout the project, I implemented clear constraints and utilized primary, foreign, and compound keys to maintain strong data integrity. This hands-on experience significantly improved my skills in database design, ethical data practices, and practical application of SQL, preparing me well for future academic projects and professional opportunities.

**References**

* SQLite Tutorial (n.d.). Retrieved from: https://www.w3schools.com/sql/sqlite\_intro.asp
* Python Faker Library Documentation. (2024). Retrieved from https://faker.readthedocs.io/en/master/
* Spyder IDE Documentation. (n.d.). Retrieved from https://docs.spyder-ide.org/current/
* DB Browser for SQLite Documentation. (n.d.). Retrieved from https://sqlitebrowser.org/dl/