# Event reporting system design

hey my mission is to Design and implement a basic event reporting system that tracks student registrations, attendance, and feedback. Stores this data in a database. Generates reports about student participation and event popularity, that uses sql to create database, python that has these functions : add registration, mark attendance, submit feedback and generates reports

Got it 👍 You want a **Basic Event Reporting System** that:

1. Uses **SQL** for database creation.
2. Uses **Python** to interact with the database.
3. Supports these functions:
    - Add Registration
    - Mark Attendance
    - Submit Feedback
    - Generate Reports (student participation & event popularity)

I'll give you a **complete working example** (SQLite + Python).

---

## 1. SQL – Database Setup (SQLite)

```sql
-- event_system.sql

-- Students table
CREATE TABLE IF NOT EXISTS students (
    student_id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    email TEXT UNIQUE NOT NULL
);

-- Events table
CREATE TABLE IF NOT EXISTS events (
    event_id INTEGER PRIMARY KEY AUTOINCREMENT,
    event_name TEXT NOT NULL,
    event_date TEXT NOT NULL
);

-- Registrations table
CREATE TABLE IF NOT EXISTS registrations (
    reg_id INTEGER PRIMARY KEY AUTOINCREMENT,
    student_id INTEGER,
    event_id INTEGER,
    FOREIGN KEY(student_id) REFERENCES students(student_id),
    FOREIGN KEY(event_id) REFERENCES events(event_id)
);

-- Attendance table
CREATE TABLE IF NOT EXISTS attendance (
    attendance_id INTEGER PRIMARY KEY AUTOINCREMENT,
    reg_id INTEGER,
    attended BOOLEAN,
    FOREIGN KEY(reg_id) REFERENCES registrations(reg_id)
);

-- Feedback table
CREATE TABLE IF NOT EXISTS feedback (
    feedback_id INTEGER PRIMARY KEY AUTOINCREMENT,
    reg_id INTEGER,
    feedback_text TEXT,
    rating INTEGER CHECK(rating >=1 AND rating <=5),
    FOREIGN KEY(reg_id) REFERENCES registrations(reg_id)
);
```

---

## 2. Python – Event System Functions

python

```python
import sqlite3

DB_NAME = "event_system.db"

# Initialize database
def init_db():
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    with open("event_system.sql", "r") as f:
        cursor.executescript(f.read())

    conn.commit()
    conn.close()

# Add student registration
def add_registration(student_name, email, event_id):
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()

    # Insert or ignore student
    cursor.execute("INSERT OR IGNORE INTO students (name, email) VALUES (?, ?)",
(student_name, email))

    cursor.execute("SELECT student_id FROM students WHERE email = ?", (email,))
    student_id = cursor.fetchone()[0]

    # Register student to event
    cursor.execute("INSERT INTO registrations (student_id, event_id) VALUES (?, ?)",
(student_id, event_id))

    conn.commit()
    conn.close()
    print(f"{student_name} registered for event {event_id}")

# Mark attendance
def mark_attendance(student_email, event_id, attended=True):
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()

    cursor.execute("""
        SELECT reg_id FROM registrations
        JOIN students USING(student_id)
        WHERE email = ? AND event_id = ?
    """, (student_email, event_id))
    reg = cursor.fetchone()
    if not reg:
        print("No registration found.")
        return
    reg_id = reg[0]
    cursor.execute("INSERT INTO attendance (reg_id, attended) VALUES (?, ?)", (reg_id, attended))
    conn.commit()
    conn.close()
    print(f"Attendance marked for {student_email} in event {event_id}")

# Submit feedback
def submit_feedback(student_email, event_id, feedback_text, rating):
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()

    cursor.execute("""
        SELECT reg_id FROM registrations
        JOIN students USING(student_id)
        WHERE email = ? AND event_id = ?
    """, (student_email, event_id))
    reg = cursor.fetchone()
    if not reg:
        print("No registration found.")
        return
    reg_id = reg[0]

    cursor.execute("INSERT INTO feedback (reg_id, feedback_text, rating) VALUES (?, ?, ?)",
                   (reg_id, feedback_text, rating))
    conn.commit()
    conn.close()
    print(f"Feedback submitted by {student_email}")

# Generate reports
def generate_reports():
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()

    print("\n--- Student Participation ---")
    cursor.execute("""
        SELECT e.event_name, COUNT(r.reg_id) as total_registrations,
               SUM(a.attended) as total_attended
        FROM events e
        LEFT JOIN registrations r ON e.event_id = r.event_id
        LEFT JOIN attendance a ON r.reg_id = a.reg_id
        GROUP BY e.event_id
    """)
    for row in cursor.fetchall():
        print(f"Event: {row[0]}, Registered: {row[1]}, Attended: {row[2]}")

    print("\n--- Event Popularity (Average Rating) ---")
    cursor.execute("""
        SELECT e.event_name, AVG(f.rating)
        FROM events e
        JOIN registrations r ON e.event_id = r.event_id
        JOIN feedback f ON r.reg_id = f.reg_id
        GROUP BY e.event_id
    """)
    for row in cursor.fetchall():
        print(f"Event: {row[0]}, Avg Rating: {round(row[1],2)}")

    conn.close()
```

---

## 3. Example Usage

```python
if __name__ == "__main__":
    init_db()

    # Add some events manually (only once)
    conn = sqlite3.connect(DB_NAME)
    conn.execute("INSERT OR IGNORE INTO events (event_name, event_date) VALUES (?, ?)", ("Tech Talk", "2025-09-15"))
    conn.execute("INSERT OR IGNORE INTO events (event_name, event_date) VALUES (?, ?)", ("AI Workshop", "2025-09-20"))
    conn.commit()
    conn.close()

    # Demo flow
    add_registration("Alice", "alice@example.com", 1)
    add_registration("Bob", "bob@example.com", 2)

    mark_attendance("alice@example.com", 1, True)
    mark_attendance("bob@example.com", 2, False)

    submit_feedback("alice@example.com", 1, "Great session!", 5)
    generate_reports()
```

---

✅ This will:

- Create the database.
- Register students for events.
- Mark attendance.
- Collect feedback.
- Generate reports on **student participation & event popularity**.

---

Do you want me to also make this into a **menu-driven CLI (command-line app)** so you can run it interactively (choose options 1–5)?

make this into a menu-driven CLI (command-line app)

Perfect 👍 A **menu-driven CLI app** will let you interact with the event reporting system easily (without editing code each time).

Here's the **full Python program**:

---

## 📌 Menu-Driven Event Reporting System (Python + SQLite)

python

```python
import sqlite3

DB_NAME = "event_system.db"

# ---------- DATABASE INITIALIZATION ----------
def init_db():
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()

    # Create tables
    cursor.executescript("""
    CREATE TABLE IF NOT EXISTS students (
        student_id INTEGER PRIMARY KEY AUTOINCREMENT,
        name TEXT NOT NULL,
        email TEXT UNIQUE NOT NULL
    );

    CREATE TABLE IF NOT EXISTS events (
        event_id INTEGER PRIMARY KEY AUTOINCREMENT,
        event_name TEXT NOT NULL,
        event_date TEXT NOT NULL
    );

    CREATE TABLE IF NOT EXISTS registrations (
        reg_id INTEGER PRIMARY KEY AUTOINCREMENT,
        student_id INTEGER,
        event_id INTEGER,
        FOREIGN KEY(student_id) REFERENCES students(student_id),
        FOREIGN KEY(event_id) REFERENCES events(event_id)
    );

    CREATE TABLE IF NOT EXISTS attendance (
        attendance_id INTEGER PRIMARY KEY AUTOINCREMENT,
        reg_id INTEGER,
        attended BOOLEAN,
        FOREIGN KEY(reg_id) REFERENCES registrations(reg_id)
    );

    CREATE TABLE IF NOT EXISTS feedback (
        feedback_id INTEGER PRIMARY KEY AUTOINCREMENT,
        reg_id INTEGER,
        feedback_text TEXT,
        rating INTEGER CHECK(rating >=1 AND rating <=5),
        FOREIGN KEY(reg_id) REFERENCES registrations(reg_id)
    );
    """)

    conn.commit()
    conn.close()

# ---------- FUNCTIONS ----------
def add_event(event_name, event_date):
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    cursor.execute("INSERT INTO events (event_name, event_date) VALUES (?, ?)", (event_name, event_date))
    conn.commit()
    conn.close()
    print(f"Event '{event_name}' added successfully!\n")

def add_registration(student_name, email, event_id):
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()

    # Insert or ignore student
    cursor.execute("INSERT OR IGNORE INTO students (name, email) VALUES (?, ?)", (student_name, email))

    cursor.execute("SELECT student_id FROM students WHERE email = ?", (email,))
    student_id = cursor.fetchone()[0]

    # Register student to event
    cursor.execute("INSERT INTO registrations (student_id, event_id) VALUES (?, ?)", (student_id, event_id))

    conn.commit()
    conn.close()
    print(f"{student_name} registered for event {event_id}\n")

def mark_attendance(student_email, event_id, attended=True):
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()

    cursor.execute("""
        SELECT reg_id FROM registrations
        JOIN students USING(student_id)
        WHERE email = ? AND event_id = ?
    """, (student_email, event_id))
    reg = cursor.fetchone()
    if not reg:
        print("No registration found.\n")
        return
    reg_id = reg[0]

    cursor.execute("INSERT INTO attendance (reg_id, attended) VALUES (?, ?)", (reg_id,
attended))
    conn.commit()
    conn.close()
    print(f"Attendance marked for {student_email} in event {event_id}\n")

def submit_feedback(student_email, event_id, feedback_text, rating):
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()

    cursor.execute("""
        SELECT reg_id FROM registrations
        JOIN students USING(student_id)
        WHERE email = ? AND event_id = ?
    """, (student_email, event_id))
    reg = cursor.fetchone()
    if not reg:
        print("No registration found.\n")
        return
    reg_id = reg[0]

    cursor.execute("INSERT INTO feedback (reg_id, feedback_text, rating) VALUES (?, ?, ?)",
                   (reg_id, feedback_text, rating))
    conn.commit()
    conn.close()
    print(f"Feedback submitted by {student_email}\n")

def generate_reports():
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()

    print("\n--- Student Participation ---")
    cursor.execute("""
        SELECT e.event_name, COUNT(r.reg_id) as total_registrations,
               COALESCE(SUM(a.attended), 0) as total_attended
        FROM events e
        LEFT JOIN registrations r ON e.event_id = r.event_id
        LEFT JOIN attendance a ON r.reg_id = a.reg_id
        GROUP BY e.event_id
    """)
    for row in cursor.fetchall():
        print(f"Event: {row[0]}, Registered: {row[1]}, Attended: {row[2]}")
    print("\n--- Event Popularity (Average Rating) ---")
    cursor.execute("""
        SELECT e.event_name, ROUND(AVG(f.rating),2)
        FROM events e
        JOIN registrations r ON e.event_id = r.event_id
        JOIN feedback f ON r.reg_id = f.reg_id
        GROUP BY e.event_id
    """)
    results = cursor.fetchall()
    if results:
        for row in results:
            print(f"Event: {row[0]}, Avg Rating: {row[1]}")
    else:
        print("No feedback submitted yet.")

    conn.close()
    print()

# ---------- MENU ----------
def menu():
    init_db()
    while True:
        print("===== Event Reporting System =====")
        print("1. Add Event")
        print("2. Register Student")
        print("3. Mark Attendance")
        print("4. Submit Feedback")
        print("5. Generate Reports")
        print("6. Exit")

        choice = input("Enter choice: ")

        if choice == "1":
            name = input("Enter event name: ")
            date = input("Enter event date (YYYY-MM-DD): ")
            add_event(name, date)

        elif choice == "2":
            name = input("Enter student name: ")
            email = input("Enter student email: ")
            event_id = int(input("Enter event ID: "))
            add_registration(name, email, event_id)
        elif choice == "3":
            email = input("Enter student email: ")
            event_id = int(input("Enter event ID: "))
            attended = input("Attended? (y/n): ").lower() == "y"
            mark_attendance(email, event_id, attended)

        elif choice == "4":
            email = input("Enter student email: ")
            event_id = int(input("Enter event ID: "))
            feedback = input("Enter feedback: ")
            rating = int(input("Enter rating (1-5): "))
            submit_feedback(email, event_id, feedback, rating)

        elif choice == "5":
            generate_reports()

        elif choice == "6":
            print("Exiting... Goodbye!")
            break

        else:
            print("Invalid choice, try again.\n")

# ---------- RUN ----------
if __name__ == "__main__":
    menu()
```

---

## ▶ How it works:

1. Run the program:

```bash
python event_system.py
```

2. You'll see a **menu**:

```markdown
===== Event Reporting System =====
1. Add Event
```

```
2. Register Student
3. Mark Attendance
4. Submit Feedback
5. Generate Reports
6. Exit
```

3. Enter a number to perform the action.

---

👉 Do you also want me to add an **"auto-list events and students" feature** in the menu so you don’t have to remember IDs/emails while testing?

give me any suggestions what i can work on to improve my project

Great question 👍 You already have a solid **basic event reporting system** with registrations, attendance, feedback, and reports. To **improve and make it stand out**, here are some suggestions (ranging from simple improvements to advanced features):

---

## 🔹 Database & Data Handling

- **Validation**: Prevent duplicate registrations for the same student & event.
- **More Data**: Add fields like event location, event capacity, student department/semester.
- **Constraints**: Ensure feedback is only submitted if attendance is marked (or allow feedback only after event date).
- **Better Reports**: Add event-wise attendance percentage, student-wise participation history.

---

## 🔹 Python Features

- **Auto Listing**: When registering students or marking attendance, show a list of available events and student emails instead of requiring IDs manually.
- **Search Feature**: Search events by name/date, or students by email.
- **Export Reports**: Save reports to CSV/Excel/PDF for sharing.

- **Graphical Reports**: Use `matplotlib` or `plotly` to show participation charts, event popularity graphs, feedback ratings.

---

### 🔹 User Experience

- **Menu Navigation**: Add a “Go Back” option inside submenus.
- **Input Validation**: Prevent invalid ratings (e.g., >5), wrong dates, or empty inputs.
- **Colored CLI Output**: Use libraries like `colorama` to make reports visually better.

---

### 🔹 Advanced Features (if you want to go beyond basics)

- **Authentication**: Add admin login (only admins can add events, generate reports).
- **Multiple Roles**: Student can log in to register and give feedback, admin manages events.
- **Web Version**: Build a Flask or Django app using the same database for a nicer interface.
- **Email/SMS Notifications**: Auto-send confirmation when a student registers or reminders before an event.
- **AI Insights**: Analyze feedback with NLP (like sentiment analysis to see if students liked/disliked events).

---

👉 If this project is for **college**, adding:

- CSV/PDF report export,
- Charts for event participation,
- and simple authentication (admin vs student),

will already take it from *basic* to *impressive*.

---

Do you want me to **add charts (matplotlib)** for participation and popularity in your existing project so reports look more visual?

Is this conversation helpful so far?