Practicum 1

A. Daccache, H. Sidhu, R. Zipp

Use Case

Contact Tracing DBMS for grades 9-12 boarding school

- Description: This use case is being implemented because it presents an opportunity to build a system within the specific / unique constraints of a residential school. The expected outcome of this use case is for boarding school administrators to be able to accurately and efficiently track the spread of COVID-19 through the campus and take appropriate measures such as the imposition of quarantine to mitigate the spread of the virus.
- Actors:
  - Database Architects: A. Daccache, H. Sidhu, R. Zipp for CS5200 F2020.
  - Secondary Actors: School administrators, student life professionals, and school nurse's offices.
- Precondition: All students, faculty, and staff should be assigned a unique School ID and accounted for in the DMBS. The school nurse's office must have the capacity to administer COVID-19 tests once every five days and the school should have the capacity to quarantine as many COVID-19 positive individuals as needed.
- Post-condition: DBMS successfully guides policy decisions regarding COVID-19 responses at the school without failing.
- Flow:
  - The structure of the school is outlined via data inserts to Building and SpaceRoom tables.
  - Faculty, staff, and student information inserted into DB.
  - Baseline medical records created for each entry.
  - Events such as classes and athletic events are scheduled and inserted into Events table. Attendees are recorded in EventAttendees table.
  - Entire school population is tested once every 5 days for COVID-19. Upon a positive diagnosis, the following protocols are activated:
    - Positive tests initiate creation of CaseID table and TreatmentPlan attribute associated with MedicalID. People.IsQuar attribute is updated to "True" and reverts to False after 15 days have passed.
    - DBMS runs contact tracing query. Attendees of events where positive patient was also present are also quarantined with IsQuar = true.
- Alternative Flows:
  - None to consider in this model. Possible future builds include protocols for moving patients off site to hospital.
- Exceptions:
  1. A quarantined person cannot attend events.

2. CaseID cannot be non-null without corresponding positive test result.
3. People cannot attend events on the same day that begin at the same time.
4. Group IDs (1 = student, 2 = faculty, 3 = staff) are final / immutable.

- Requirements: Assuming implementation in the United States, DBMS must be compliant with US privacy law, specifically the provisions of FERPA and HIPAA.

Conceptual Model

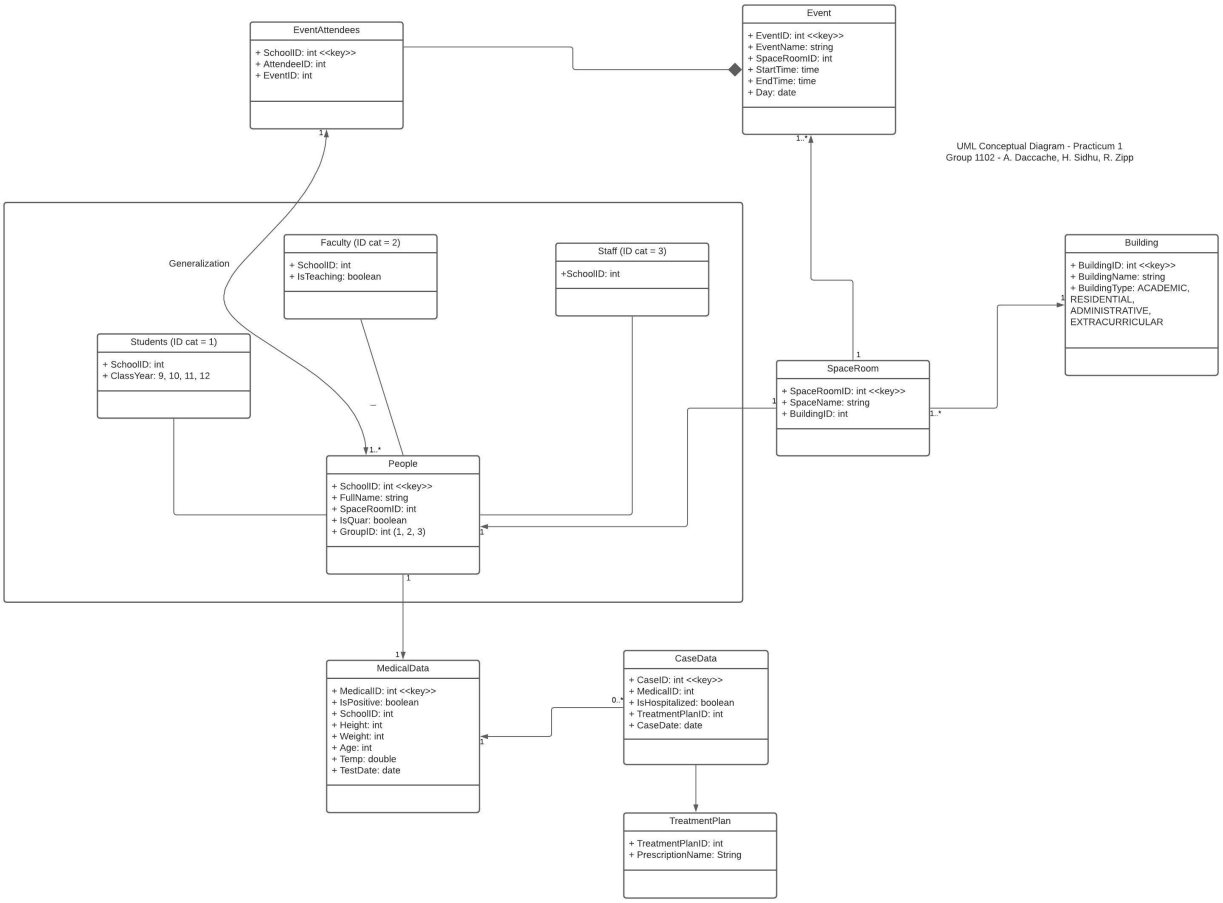


Figure 1: UML Class Diagram

Logical Data Model

https://app.lucidchart.com/invitations/accept/6fb653e2-047e-4435-8fa4-3648fb205f82

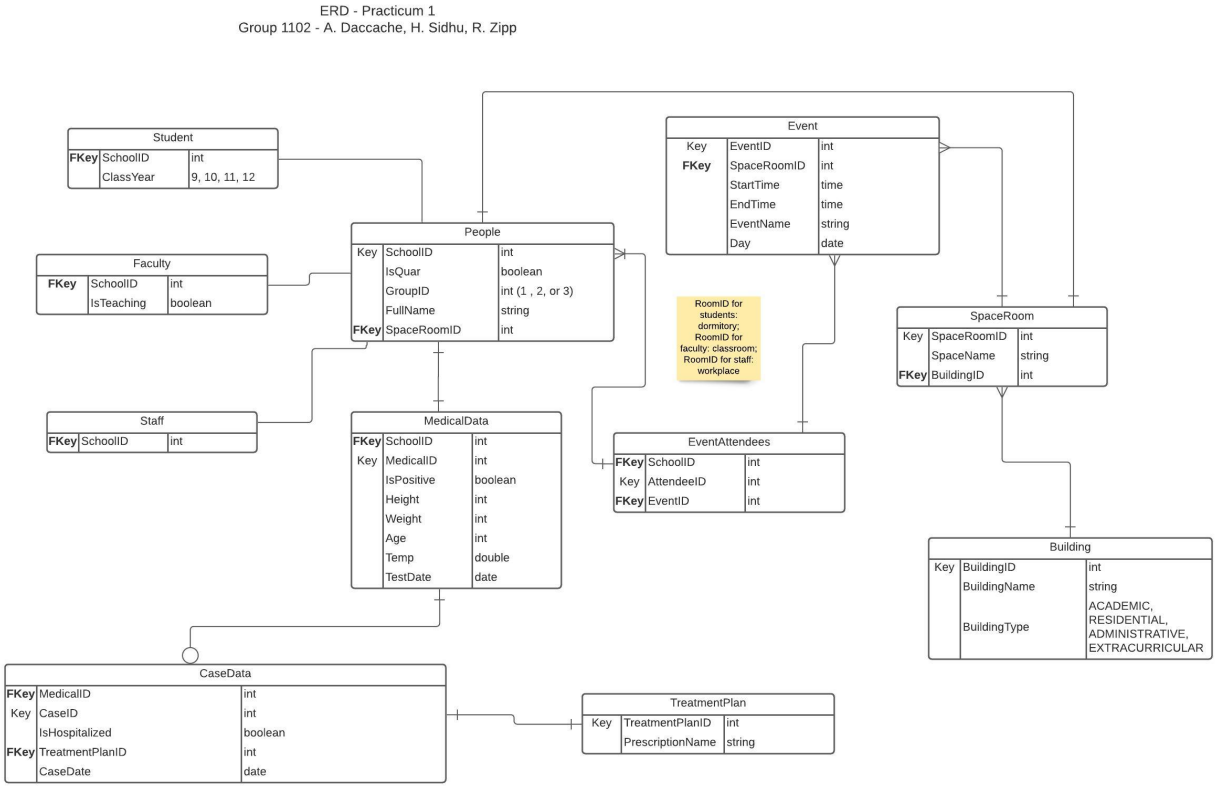


Figure 2: IE-Crow's Foot

SQL Data Definition Statements

```
1 • DESCRIBE Building;
```

100% 19:1

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
BuildingID	int	NO	PRI	NULL	auto_increment
BuildingName	text	YES		NULL	
BuildingType	set('ACADEMIC','RESIDENTIAL','ADMINISTRA...	YES		NULL	

```
CREATE TABLE Building (
    BuildingID INT PRIMARY KEY AUTO_INCREMENT,
    BuildingName TEXT,
    BuildingType SET('ACADEMIC', 'RESIDENTIAL', 'ADMINISTRATIVE', 'EXTRACURRICULAR')
);
```

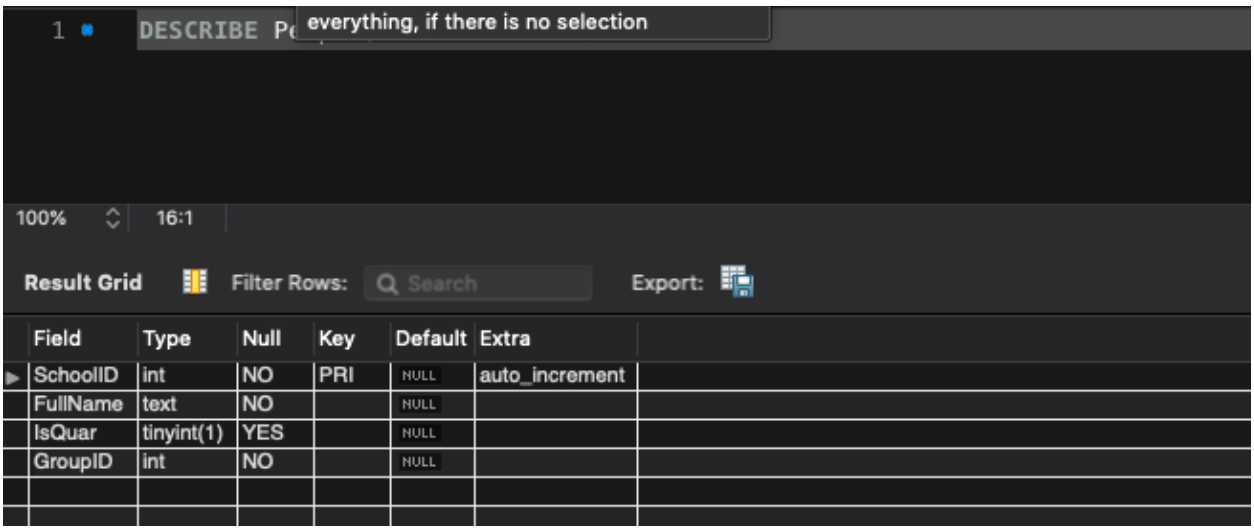
```
1 • DESCRIBE SpaceRoom;
```

100% 19:1

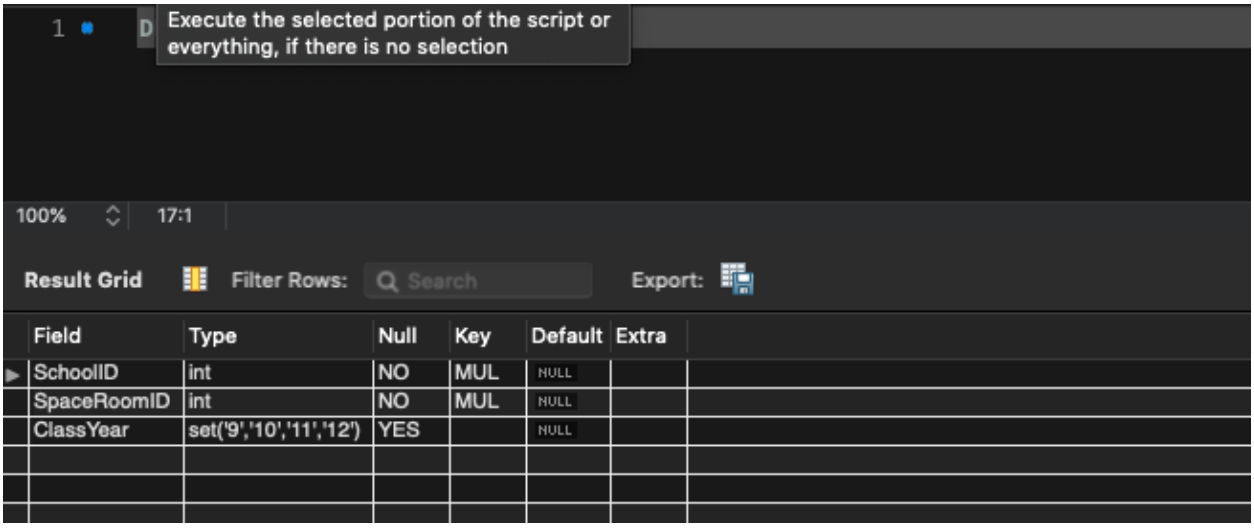
Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
SpaceRoomID	int	NO	PRI	NULL	auto_increment
SpaceName	text	YES		NULL	
BuildingID	int	NO	MUL	NULL	

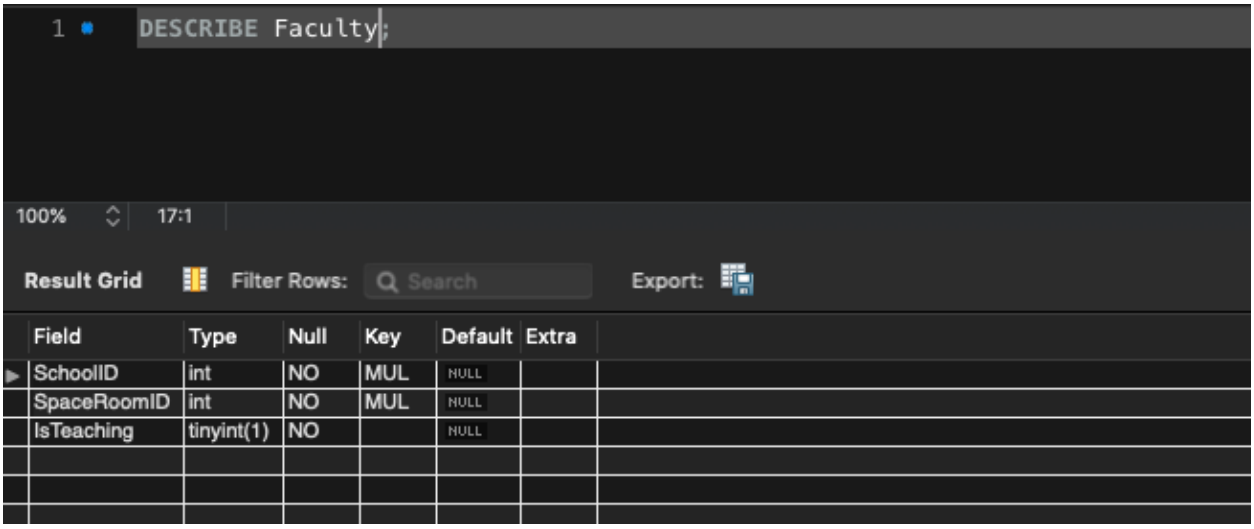
```
CREATE TABLE SpaceRoom (
    SpaceRoomID INT PRIMARY KEY AUTO_INCREMENT,
    SpaceName TEXT,
    BuildingID INT NOT NULL,
    CONSTRAINT BuildingID_fk FOREIGN KEY (BuildingID) REFERENCES Building(BuildingID)
);
```



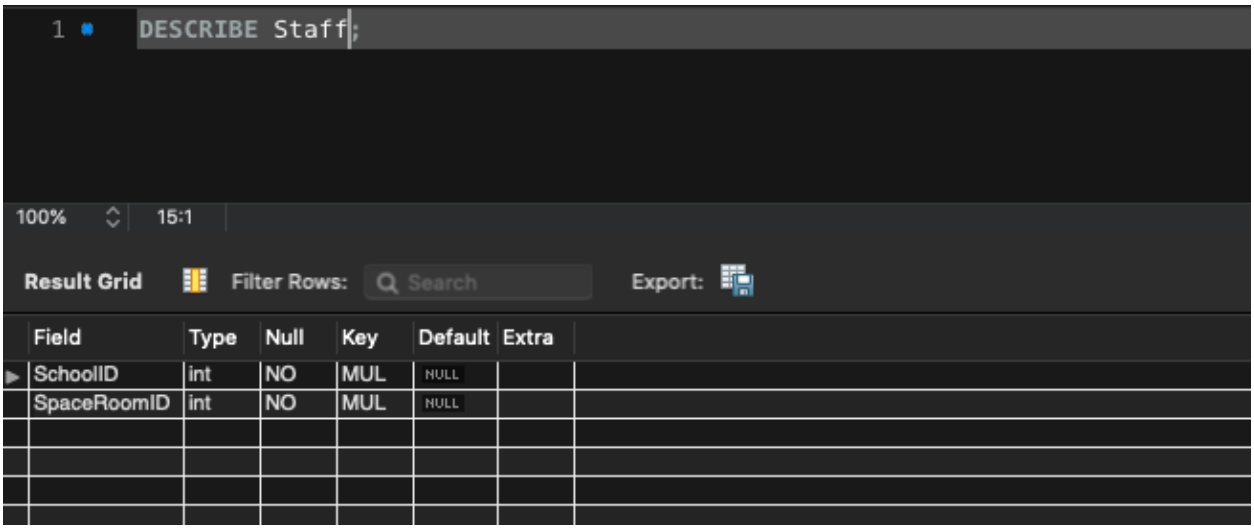

```
CREATE TABLE People (
    SchoolID INT PRIMARY KEY AUTO_INCREMENT,
    FullName TEXT NOT NULL,
    SpaceRoomID INT NOT NULL,
    IsQuar BOOLEAN,
    GroupID INT NOT NULL,
    CONSTRAINT SpaceRoomID_fk FOREIGN KEY (SpaceRoomID) REFERENCES SpaceRoom(SpaceRoomID)
);
```



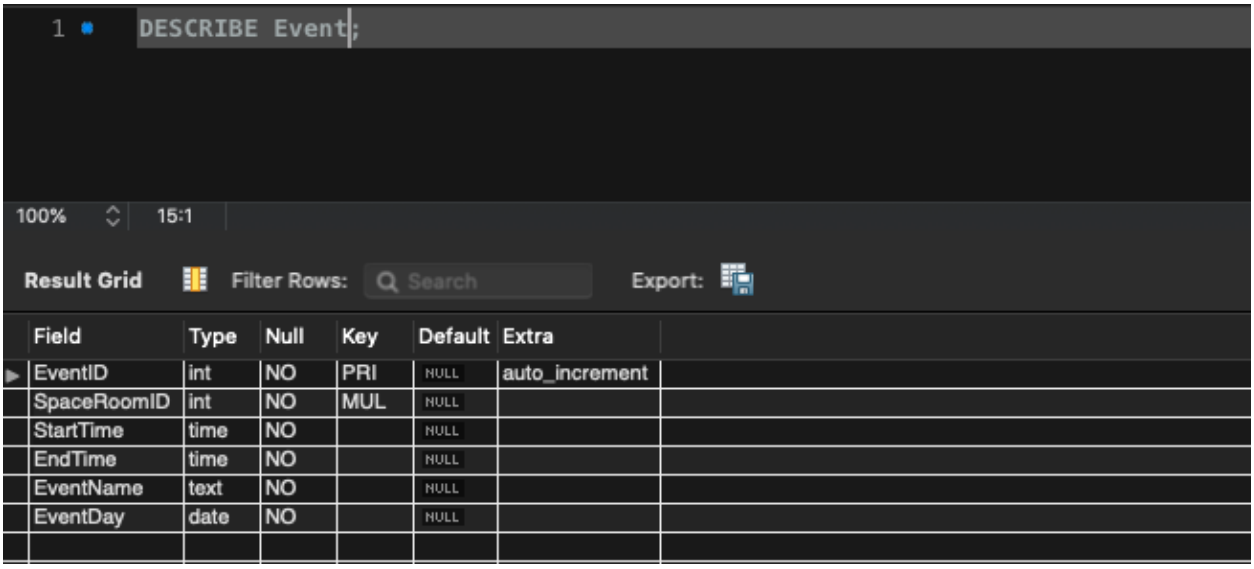

```
CREATE TABLE Student (
    SchoolID INT NOT NULL,
    ClassYear SET('9', '10', '11', '12'),
    CONSTRAINT StudentSchoolID FOREIGN KEY (SchoolID) REFERENCES People(SchoolID)
);
```



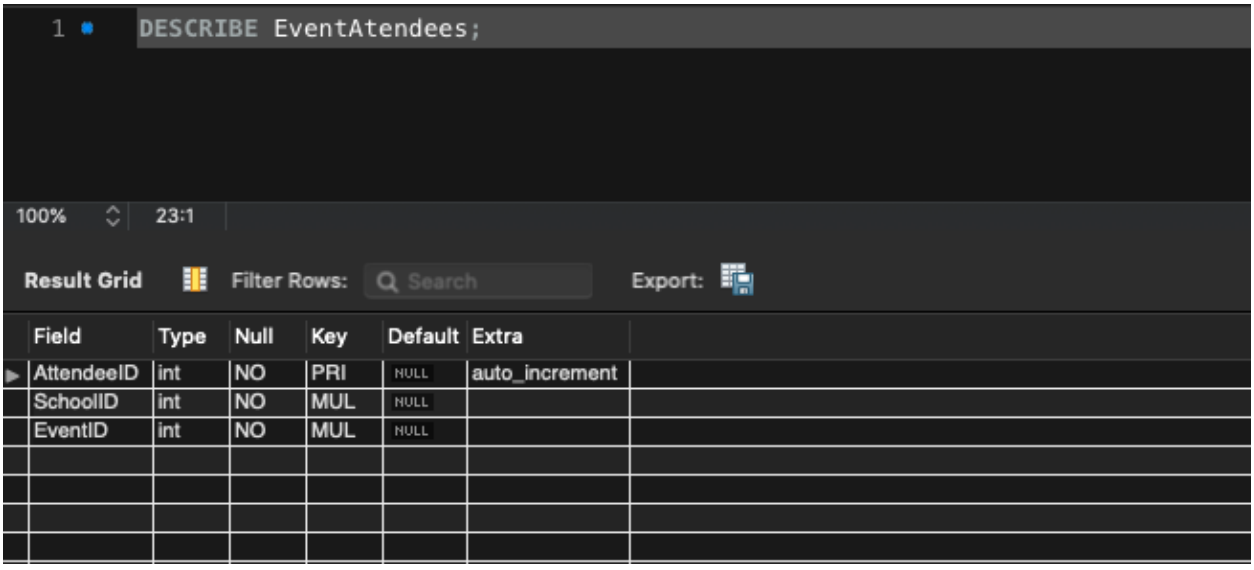

```sql
CREATE TABLE Faculty (
    SchoolID INT NOT NULL,
    IsTeaching BOOLEAN NOT NULL,
    CONSTRAINT FacultySchoolID FOREIGN KEY (SchoolID) REFERENCES People(SchoolID)
);
```



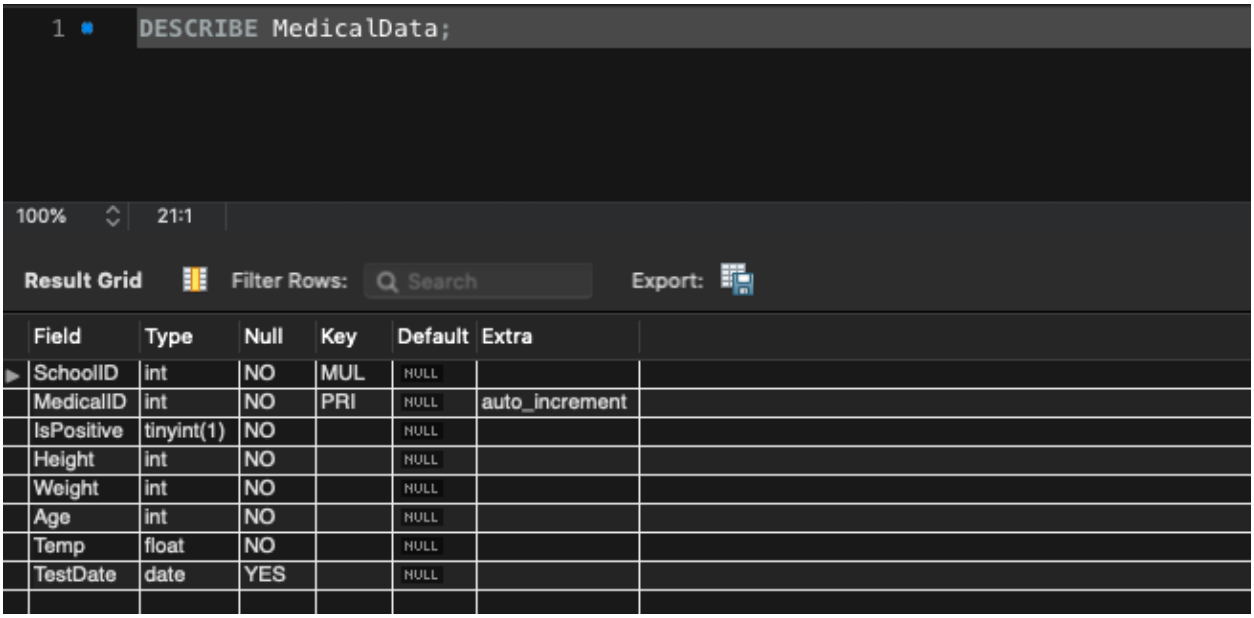

```sql
CREATE TABLE Staff (
    SchoolID INT NOT NULL,
    CONSTRAINT StaffSchoolID FOREIGN KEY (SchoolID) REFERENCES People(SchoolID)
);
```



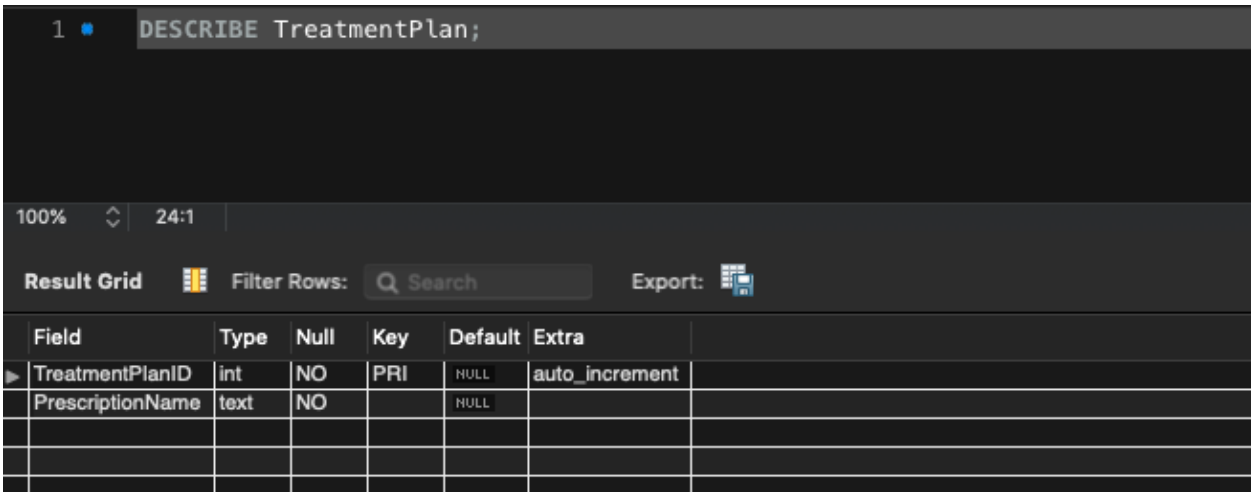

```sql
CREATE TABLE Event (
    EventID INT PRIMARY KEY AUTO_INCREMENT,
    SpaceRoomID INT NOT NULL,
    StartTime TIME NOT NULL,
    EndTime TIME NOT NULL,
    EventName TEXT NOT NULL,
    EventDay DATE NOT NULL,
    CONSTRAINT EventSpaceRoomID FOREIGN KEY (SpaceRoomID) REFERENCES SpaceRoom(SpaceRoomID)
);
```



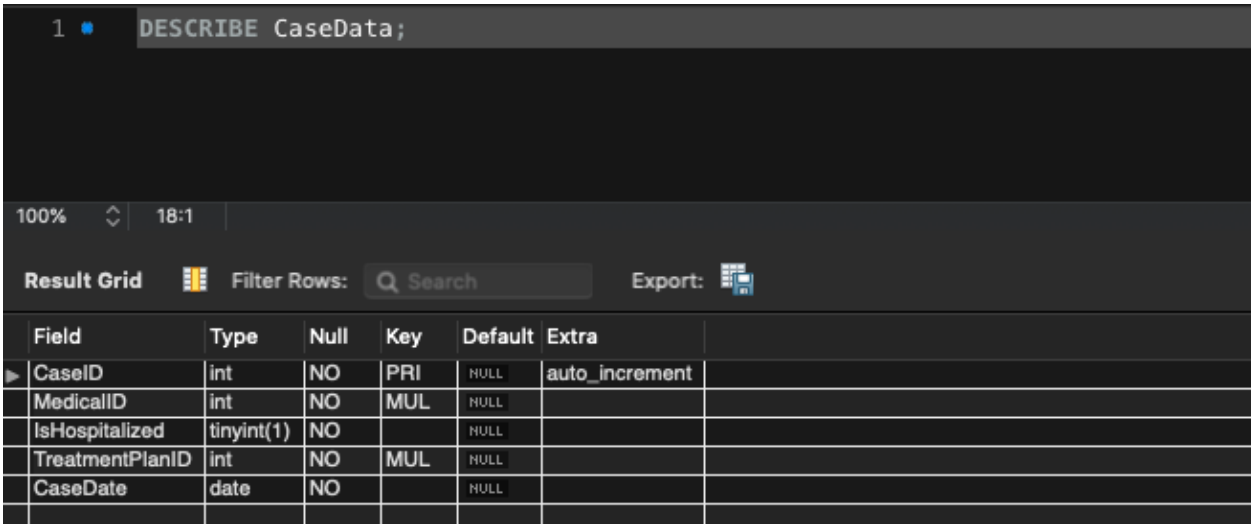

```sql
CREATE TABLE EventAtendees (
    AttendeeID INT PRIMARY KEY AUTO_INCREMENT,
    SchoolID INT NOT NULL,
    EventID INT NOT NULL,
    CONSTRAINT AttendeeSchoolID FOREIGN KEY (SchoolID) REFERENCES People(SchoolID),
    CONSTRAINT EventID_fk FOREIGN KEY (EventID) REFERENCES Event(EventID)
);
```




```sql
CREATE TABLE MedicalData (
    SchoolID INT NOT NULL,
    MedicalID INT PRIMARY KEY AUTO_INCREMENT,
    IsPositive BOOLEAN NOT NULL,
    Height INT NOT NULL,
    Weight INT NOT NULL,
    Age INT NOT NULL,
    Temp FLOAT NOT NULL,
    TestDate DATE,
    CONSTRAINT MedicalSchoolID FOREIGN KEY (SchoolID) REFERENCES People(SchoolID)
);
```




```sql
CREATE TABLE TreatmentPlan (
    TreatmentPlanID INT PRIMARY KEY AUTO_INCREMENT,
    PrescriptionName TEXT NOT NULL
);
```



```
1 • DESCRIBE CaseData;
```

100% 18:1

Result Grid | Filter Rows: Search | Export:

Field	Type	Null	Key	Default	Extra
CaseID	int	NO	PRI	NULL	auto_increment
MedicalID	int	NO	MUL	NULL	
IsHospitalized	tinyint(1)	NO		NULL	
TreatmentPlanID	int	NO	MUL	NULL	
CaseDate	date	NO		NULL	

```
CREATE TABLE CaseData (
    CaseID INT PRIMARY KEY AUTO_INCREMENT,
    MedicalID INT NOT NULL,
    IsHospitalized BOOLEAN NOT NULL,
    TreatmentPlanID INT NOT NULL,
    CaseDate DATE NOT NULL,
    CONSTRAINT MedicalID_fk FOREIGN KEY (MedicalID) REFERENCES MedicalData(MedicalID),
    CONSTRAINT treatmentPlanID_fk FOREIGN KEY (treatmentPlanID) REFERENCES
    TreatmentPlan(treatmentPlanID)
);
```

BCNF

1. First Normal Form
    i) None of the attributes hold multiple values
2. Second Normal Form
    i) No non-prime attribute is dependent on the proper subset of any candidate key
3. Third Normal Form
    i) Transitive functional dependency of non-prime attribute on any super key should be removed
    ii) For each functional dependency X->Y at least one of the following should hold: A. X i a super key of table B. Y is a prime attribute of table
4. BCNF (ie. 3.5NF)
    i) Example: created table 'EventAttendees' which acts as junction table to resolve functional dependencies

Queries

Query A: Join

Query B: Subquery

Query C: Group By

Query D: Complex Search

Query E: Advanced Query