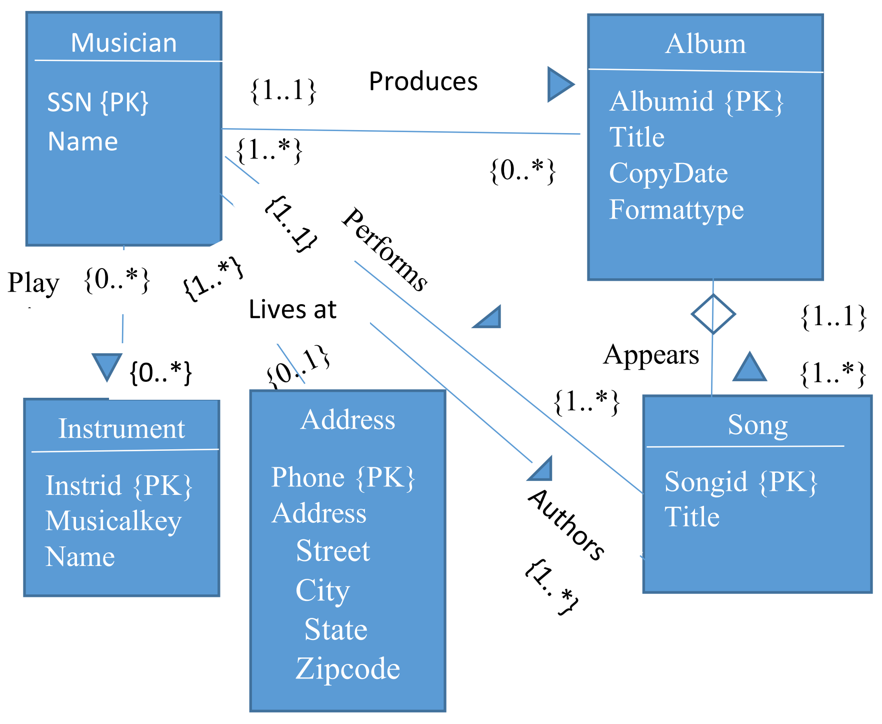
1) Notown Records has decided to store information about musicians who perform on its albums (as well as other company data) in a database. The company has wisely chosen to hire you as a database designer (at your usual consulting fee of $3600/day).

- Each musician that records at Notown has an SSN, a name, an address, and a phone number. Poorly paid musicians do not have cell phones, often share the same address, and no address has more than one landline phone. Cell phones are not tracked.
- Each instrument used in songs recorded at Notown has a unique identification number, a name (e.g., guitar, synthesizer, flute) and a musical key (e.g., C, B-flat, E-flat).
- Each album recorded on the Notown label has a unique identification number, a title, a copyright date, a format (e.g., CD or MC), and an album identifier.
- Each song recorded at Notown has a title and an author. The author of a song is a musician. There is 1 and only 1 author per song.
- Each musician may play several instruments, and a given instrument may be played by several musicians.
- Each album has a number of songs on it, but no song may appear on more than one album.
- Each song is performed by one or more musicians, and a musician may perform a number of songs.
- Each album has exactly one musician who acts as its producer. A musician may produce several albums, of course.

Design a conceptual schema for Notown and draw an UML diagram for your schema. Be sure to indicate all key and cardinality constraints and any assumptions you make. Identify any constraints you are unable to capture in the ER diagram and briefly explain why you could not express them. Once you have created the diagram, create the necessary SQL CREATE TABLE commands necessary to support it.

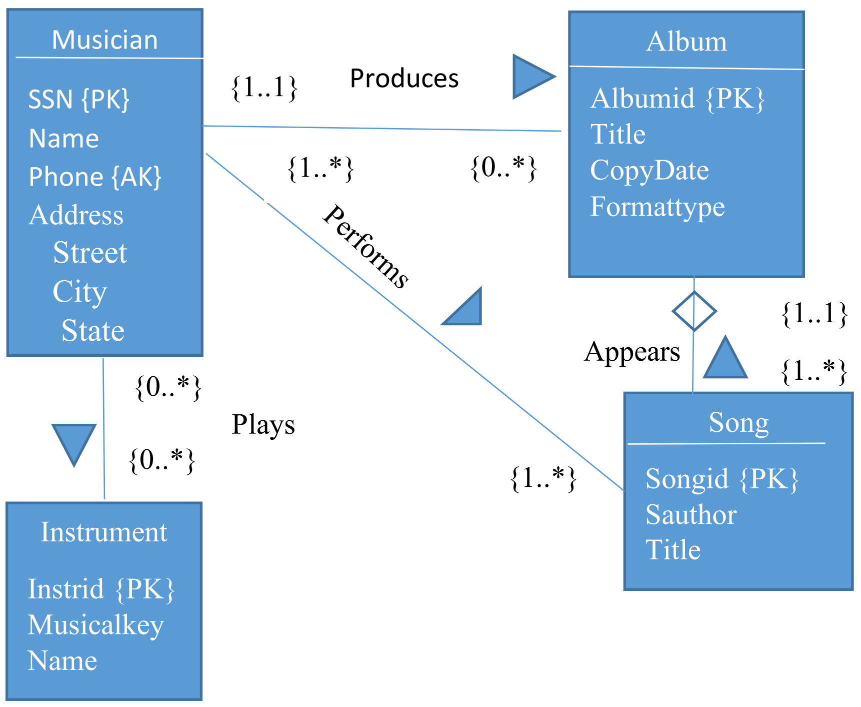


SQL :

```
CREATE TABLE address

                  ( phone CHAR(11) PRIMARY KEY,
                   Street VARCHAR(64) NOT NULL ,
                    City VARCHAR(64) NOT NULL,
                    State CHAR(2) NOT NULL,
                  );

CREATE TABLE musician
             ( ssn INT PRIMARY KEY,
               name VARCHAR(64) NOT NULL,
                phone CHAR(11) DEFAULT “NOT KNOWN”,
                CONSTRAINT musician_address_fk FOREIGN KEY  (phone) REFERENCES address(phone)
                          ON DELETE SET DEFAULT,
                          ON UPDATE SET DEFAULT
               );


 CREATE TABLE instrument
             ( instrumentid INT PRIMARY KEY,
               name VARCHAR(64) NOT NULL,
               musicalkey VARCHAR(64)
               );
```

```
CREATE TABLE album
          ( albumid INT AUTO_INCREMENT PRIMARY KEY,
            releasedate DATE NOT NULL,
            formattype char(8) NOT NULL,
            producer INT NOT NULL,
            FOREIGN KEY (producer) REFERENCES musician(ssn)
              ON UPDATE CASCADE ON DELETE CASCADE
            );


CREATE TABLE song
          ( songid INT AUTO_INCREMENT PRIMARY KEY,
            title VARCHAR(128) NOT NULL,
            author INT NOT NULL,
            albumid INT NOT NULL,
            FOREIGN KEY (albumid) REFERENCES album(albumid)
              ON UPDATE RESTRICT ON DELETE RESTRICT,
            FOREIGN KEY (author) REFERENCES musician(ssn)
              ON UPDATE RESTRICT ON DELETE RESTRICT
            );
-- mapping tables to support the multiple artists on a song

CREATE TABLE performs
            (artist INT,
             song INT,
             PRIMARY KEY (artist,song),
             FOREIGN KEY (artist) REFERENCES musician(ssn)
               ON UPDATE RESTRICT ON DELETE RESTRICT,
             FOREIGN KEY (song) REFERENCES song(songid)
               ON UPDATE RESTRICT ON DELETE RESTRICT
                    );
-- mapping table to support the multiple instruments a musician can play

CREATE TABLE musiciantoinstrument
          ( instrumentid INT,
           artist INT,
           PRIMARY KEY (instrumentid,artist),
           FOREIGN KEY (instrumentid) REFERENCES instrument(instrumentid)
               ON UPDATE RESTRICT ON DELETE RESTRICT,
          FOREIGN KEY (artist) REFERENCES musician(ssn)
               ON UPDATE RESTRICT ON DELETE RESTRICT
           );
```

b. Update the solution for the 21st century. Change the requirements such that addresses are not guaranteed to have a land line; however, all musicians have one cell phone.



SQL :

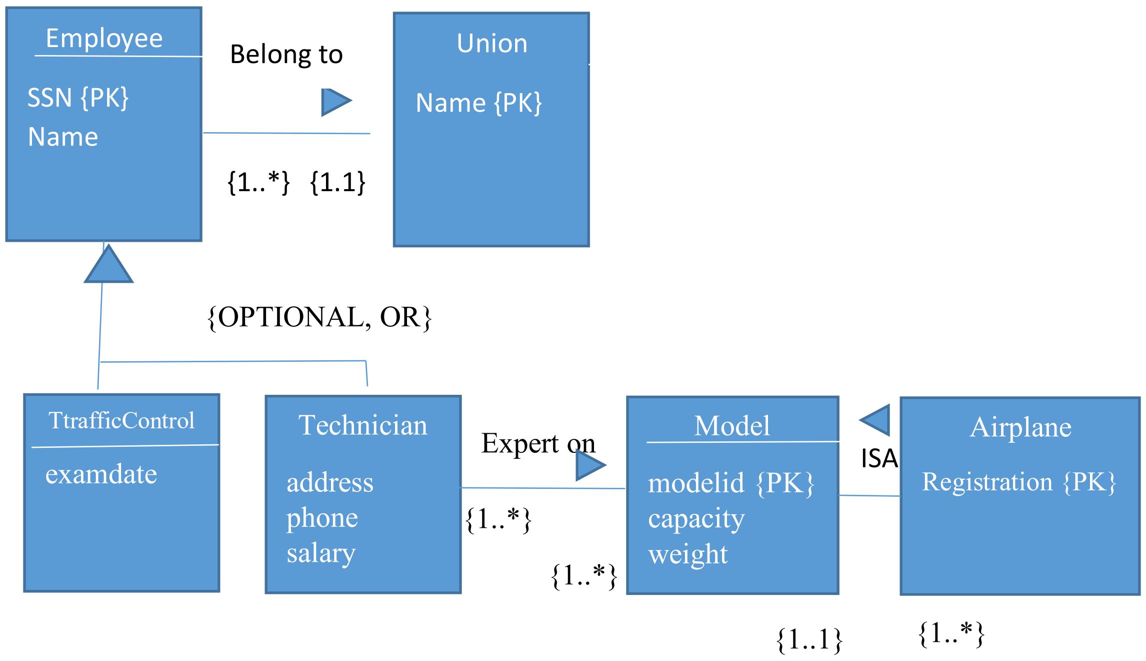
```
CREATE TABLE musician
          ( ssn INT PRIMARY KEY,
            name VARCHAR(64) NOT NULL,
             phone CHAR(11) NOT NULL,
              Street VARCHAR(64),
              City VARCHAR(64),
               State CHAR(2) );


 CREATE TABLE instrument
          ( instrumentid INT PRIMARY KEY,
            name VARCHAR(64) NOT NULL,
            musicalkey VARCHAR(64)
            );


 CREATE TABLE album
          ( albumid INT AUTO_INCREMENT PRIMARY KEY,
            releasedate DATE NOT NULL,
            formattype char(8) NOT NULL,
            producer INT NOT NULL,
            FOREIGN KEY (producer) REFERENCES musician(ssn)
              ON UPDATE CASCADE ON DELETE CASCADE );
  CREATE TABLE song

           ( songid INT AUTO_INCREMENT PRIMARY KEY,
            title VARCHAR(128) NOT NULL,
            author INT NOT NULL,
            albumid INT NOT NULL,
            FOREIGN KEY (albumid) REFERENCES album(albumid)
              ON UPDATE RESTRICT ON DELETE RESTRICT,
            FOREIGN KEY (author) REFERENCES musician(ssn)
              ON UPDATE RESTRICT ON DELETE RESTRICT
            );
 -- mapping tables to support the multiple artists on a song

 CREATE TABLE performs
            (artist INT,
             song INT,
             PRIMARY KEY (artist,song),
             FOREIGN KEY (artist) REFERENCES musician(ssn)
               ON UPDATE RESTRICT ON DELETE RESTRICT,
             FOREIGN KEY (song) REFERENCES song(songid)
               ON UPDATE RESTRICT ON DELETE RESTRICT
                    );
 -- mapping table to support the multiple instruments a musician can play

CREATE TABLE musiciantoinstrument
           ( instrumentid INT,
            artist INT,
            PRIMARY KEY (instrumentid,artist),
            FOREIGN KEY (instrumentid) REFERENCES instrument(instrumentid)
                ON UPDATE RESTRICT ON DELETE RESTRICT,
           FOREIGN KEY (artist) REFERENCES musician(ssn)
                ON UPDATE RESTRICT ON DELETE RESTRICT
            );
```

2)The Computer Science Department frequent fliers have been complaining to Dane County Airport officials about the poor organization at the airport. As a result, the officials decided that all information related to the airport should be organized using a DBMS, and you have been hired to design the database. Your first task is to organize the information about all the airplanes stationed and maintained at the airport. The relevant information is as follows:

- Every airplane has a registration number, and each airplane is of a specific model.

- The airport accommodates a number of airplane models, and each model is identified by a model number (e.g., DC-10) and has a capacity and a weight.
- A number of technicians work at the airport. You need to store the name, SSN, address, phone number, and salary of each technician.
- Each technician is an expert on one or more plane model(s), and his or her expertise may overlap with that of other technicians. This information about technicians must also be recorded.
- A traffic controller also has a SSN, a name, and must have an annual medical examination. For each traffic controller, you must store the date of the most recent exam.
- All airport employees (including technicians) belong to one and only one union, however the unions may vary depending on the job performed by the employee. You must track the unions your employees are a member of and associate the employee with the union. Assume that the union name is a unique value. You can also assume that each employee is uniquely identified by a social security number.

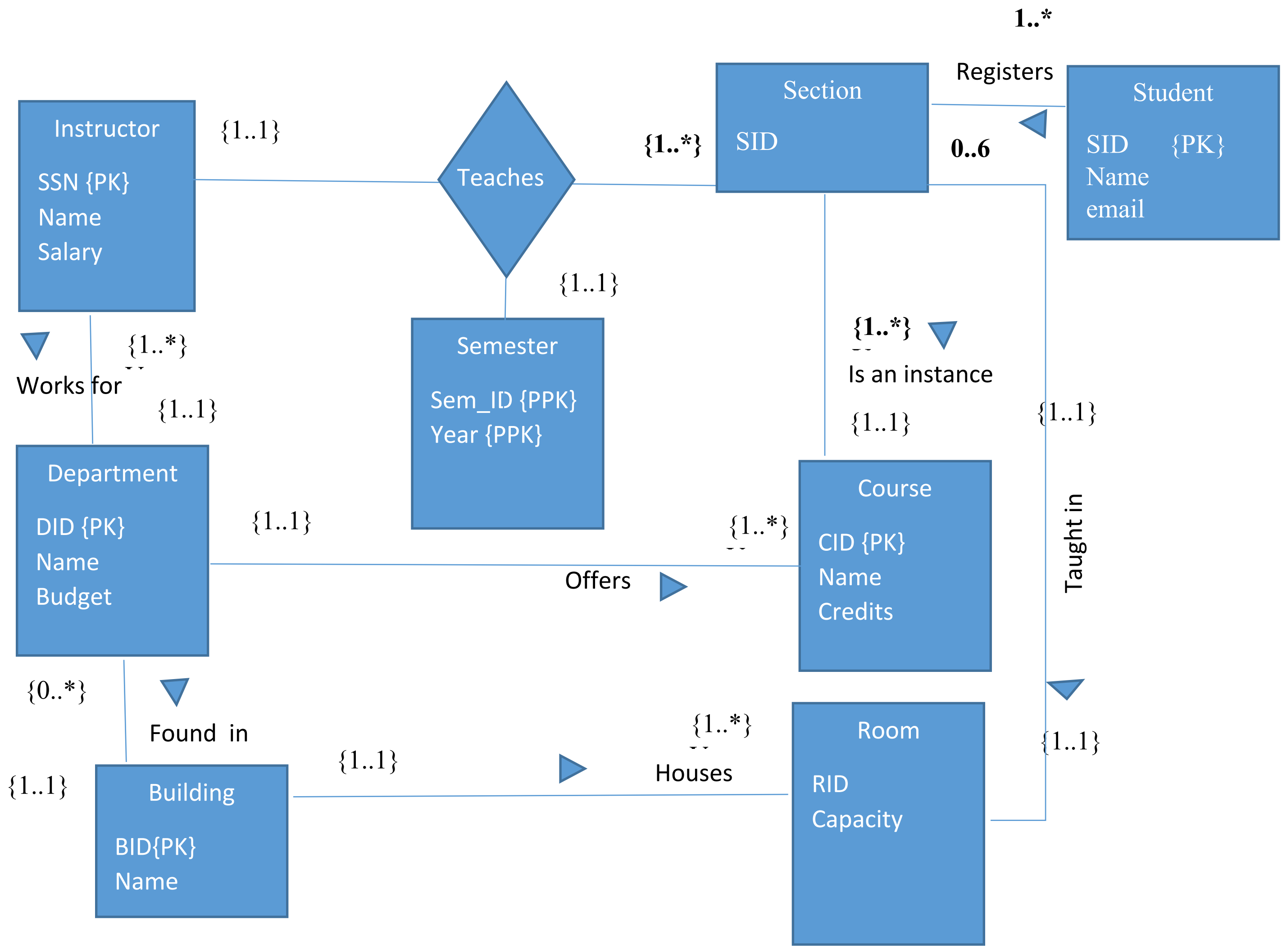
Design a conceptual schema for the airport and draw an UML diagram for your schema. Be sure to indicate all key and cardinality constraints and any assumptions you make. Once you have created the diagram, create the necessary SQL CREATE TABLE commands necessary to support it.



```
CREATE TABLE employee
        ( ssn INT PRIMARY KEY,
        Name VARCHAR(45),
         unionid INT,
        FOREIGN KEY (unionid) REFERENCES union (unionid)
           ON UPDATE RESTRICT ON DELETE RESTRICT

         );


CREATE TABLE union_details (

 Union_id INT PRIMARY KEY,

union_name VARCHAR(45)) ;

CREATE TABLE traffic_controller

        ( ssn INT PRIMARY KEY,
         physicalexam DATE NOT NULL,
         FOREIGN KEY (ssn) REFERENCES employee(ssn)
           ON UPDATE CASCADE ON DELETE CASCADE
         );


CREATE TABLE technician

        ( ssn INT PRIMARY KEY,
         address VARCHAR(128),
         salary DECIMAL(10,2) NOT NULL,
         phone CHAR(10),
         FOREIGN KEY (ssn) REFERENCES employee(ssn)
           ON UPDATE CASCADE ON DELETE CASCADE
         );


 CREATE TABLE model
          ( modelid INT PRIMARY KEY,
         capacity DECIMAL(10,2) NOT NULL,
         weight DECIMAL(10,2) NOT NULL
         );


 CREATE TABLE airplane
        ( registration INT PRIMARY KEY,
         modeltype INT NOT NULL,
         FOREIGN KEY (modeltype) REFERENCES model(modelid)
           ON UPDATE CASCADE ON DELETE CASCADE
         );


CREATE TABLE expertise
```

```
( ssn INT,
  modelid INT,
  PRIMARY KEY (ssn,modelid),
  FOREIGN KEY (ssn) REFERENCES technician(ssn)
     ON UPDATE CASCADE ON DELETE CASCADE,
  FOREIGN KEY (modelid) REFERENCES model(modelid)
     ON UPDATE CASCADE ON DELETE CASCADE
  );
```

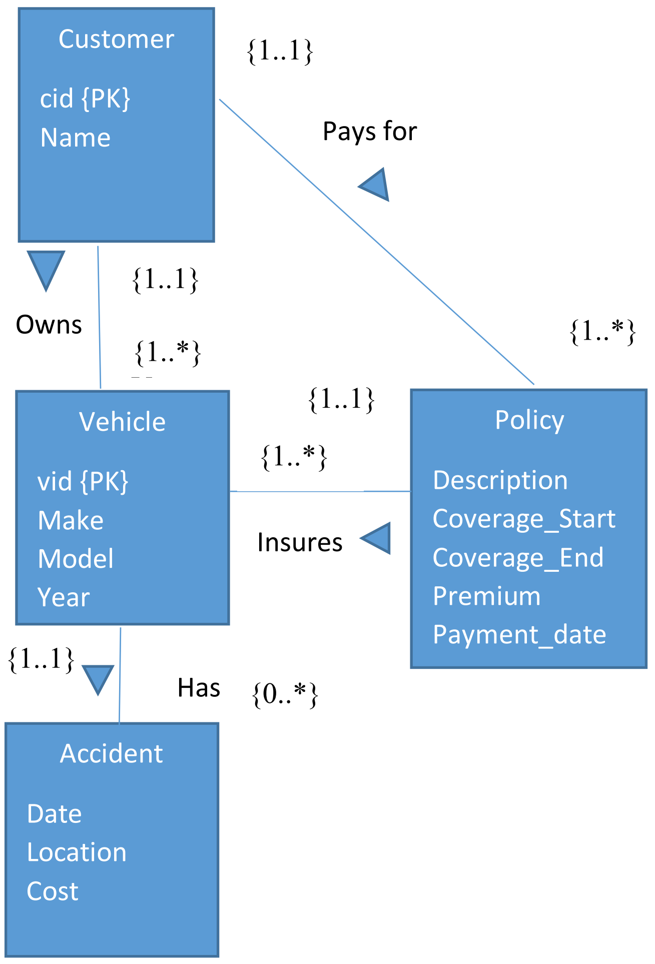
3. You are tracking courses taught by an instructor offered by a university. For the instructor you are tracking the instructor's ID, name, and salary. The ID is unique for each instructor. An instructor works in one department and many instructors may work in a department. A department contains a name, a budget and the building name where its offices are held. The department name is unique for each department at the university. Departments offer courses taught by an instructor. For each course the department tracks the course name, the course title and the number of credits a student earns in completing the course. A course section is a course offering in a specific time period, in particular a specific semester, and specific year. The values for semester are: 'fall', 'spring' and 'summer'. For each offering of a course there is a section id that is unique for the course. A course offering is assigned to a class room. A classroom is a room in one of the university's buildings, it has a room number that indicate the floor and the specific room in the building; it also has a person capacity. For each university building there is a building name. A Student is identified by his/her student Id. The database also tracks the student's name and email. Students register for specific course sections. A student may register for 0 to 6 different courses. A course section may have 1 to many students.


1..*
Registers
Section
SID
{1..*}
0..6
Student
SID {PK}
Name
email
Instructor
SSN {PK}
Name
Salary
{1..1}
Teaches
{1..1}
Semester
Sem_ID {PPK}
Year {PPK}
{1..*}
Works for
{1..1}
{1..*}
Is an instance
{1..1}
{1..1}
Department
DID {PK}
Name
Budget
{1..1}
{1..*}
Offers
Course
CID {PK}
Name
Credits
Taught in
{0..*}
Found in
{1..1}
Building
BID{PK}
Name
{1..1}
{1..*}
Houses
Room
RID
Capacity
{1..1}

```
CREATE TABLE building ( bid INT PRIMARY KEY,
                                          Bname VARCHAR(256) NOT NULL );
CREATE TABLE time ( semester_id ENUM(‘Fall’, ‘Spring’, ’Summer’) ,
                                     year Date,
                                     Constraint time_pk PRIMARY KEY (sid,year));
CREATE TABLE student(ssid INT PRIMARY KEY,
                                    Email VARCHAR(40) NOT NULL,
                                    Name VARCHAR(40) NOT NULL ) ;

CREATE TABLE room (rid INT,
                                 bid INT,
                                 Capacity INT NOT NULL,
                                 CONSTRAINT room_pk PRIMARY KEY (rid,bid),
                                 CONSTRAINT room_fk FOREIGN KEY bid REFERENCES building(bid)
                                     ON UPDATE CASCADE ON DELETE CASCADE);
CREATE TABLE department( did INT PRIMARY KEY,
                                           Bid INT NOT NULL,
                                           Name VARCHAR(256),
                                     CONSTRAINT dept_fk FOREIGN KEY (bid) REFERENCES building(bid)
                                     ON UPDATE RESTRICT ON DELETE RESTRICT);
CREATE TABLE instructor( iid INT PRIMARY KEY,
                                          Salary DECIMAL (9,2) NOT NULL,
                                           Name VARCHAR(256) NOT NULL,
                                           Did INT NOT NULL,
                                     CONSTRAINT instructor_fk FOREIGN KEY (did) REFERENCES department(did)
                                     ON UPDATE RESTRICT ON DELETE RESTRICT);
CREATE TABLE course( cid INT PRIMARY KEY,
                                     Credit INT NOT NULL,
                                      Name VARCHAR(256),
                                       Did INT NOT NULL,
                                     CONSTRAINT instructor_fk FOREIGN KEY (did) REFERENCES department(did)
                                     ON UPDATE RESTRICT ON DELETE RESTRICT);
CREATE TABLE course_section( section_id INT,
                                     cid INT NOT NULL,
                                     bid INT NOT NULL,
                                     rid INT NOT NULL,
                                     iid INT NOT NULL,
                                     semester_id INT NOT NULL,
                                     year_id INT NOT NULL,
                                      CONSTRAINT section_pk PRIMARY KEY (section_id, cid),
                                     CONSTRAINT section_fk FOREIGN KEY (cid) REFERENCES course(cid)
                                     ON UPDATE RESTRICT ON DELETE RESTRICT,
                                     CONSTRAINT section_fk2 FOREIGN KEY (bid,rid) REFERENCES room(bid,rid)
                                     ON UPDATE RESTRICT ON DELETE RESTRICT);
```

```
                          CONSTRAINT instructor_fk1 FOREIGN KEY (cid) REFERENCES instructor(iid)
                          ON UPDATE RESTRICT ON DELETE RESTRICT,
                          CONSTRAINT section_fk3 FOREIGN KEY (semester_id, yid) REFERENCES
                          time(semester_id, yid) ON UPDATE RESTRICT ON DELETE RESTRICT,
                          CONSTRAINT section_fk4 FOREIGN KEY (iid) REFERENCES instructor(iid)
                          ON UPDATE RESTRICT ON DELETE RESTRICT);
CREATE TABLE course_roster( section_id INT,
                             ssid INT,
                          CONSTRAINT roster_pk PRIMARY KEY (section_id, ssid),
                          CONSTRAINT roster_fk FOREIGN KEY (section_id) REFERENCES
                              section(section_id) ON UPDATE CASCADE ON DELETE CASCADE,
                          CONSTRAINT roster_fk2 FOREIGN KEY (ssid) REFERENCES student(ssid)
                          ON UPDATE CASCADE ON DELETE CASCADE);
```

4. You have been hired to create a database for a car insurance company. A customer of the insurance company may have multiple cars they insure with the company. Each car has had 0 to many accidents. The insurance company tracks the date, location and the cost of each accident. Each insurance policy covers one or more cars and has one premium payment associated with it. A policy is owned by 1 and only 1 customer. Each premium payment is for a particular period of time and has an associated due date, and the date when the payment was received.



```
CREATE TABLE customer( cid INT PRIMARY KEY,
                                    Name VARCHAR(256) NOT NULL ) ;
CREATE TABLE policy( pid INT PRIMARY KEY,
                               decription VARCHAR(256),
                               policy_start DATE NOT NULL,
                               policy_end DATE NOT NULL,
                               premium INT NOT NULL,
                               paid BOOL DEFAUL FALSE,
                               cid INT NOT NULL,
                               CONSTRAINT policy_fk1 FOREIGN KEY (cid) REFERENCES customer(cid)
                               ON UPDATE RESTRICT ON DELETE RESTRICT) ;

CREATE TABLE car( vid INT PRIMARY KEY,
                                    Make VARCHAR(256) NOT NULL,
                                    Model VARCHAR(256) NOT NULL,
                                    Year DATE NOT NULL,
                                    cid INT NOT NULL,
                               CONSTRAINT car_fk1 FOREIGN KEY (cid) REFERENCES customer(cid)
                               ON UPDATE RESTRICT ON DELETE RESTRICT,
                               CONSTRAINT car_fk2 FOREIGN KEY (pid) REFERENCES policy(pid)
                                                              ON UPDATE RESTRICT ON DELETE RESTRICT);
CREATE TABLE accident (accident_id INT,
                               vid INT NOT NULL,
                                date_occurred DATE NOT NULL,
                                cost INT,
                               location VARCHAR(256),
                               CONSTRAINT accident_fk1 FOREIGN KEY (vid) REFERENCES car(vid)
                               ON UPDATE CASCADE ON DELETE CASCADE);
```