

Q1

Q 1.1

Patients attend doctors. You want to store the attending information and the date of attending.

Patients identified by pid. Doctors is identified by did.

```
1 CREATE TABLE Doctors(  
2     did CHAR(10),  
3     PRIMARY KEY (did)  
4 )  
5 CREATE TABLE Patients(  
6     pid CHAR(10),  
7     PRIMARY KEY (pid)  
8 )  
9 CREATE TABLE Record(  
10    did CHAR(10),  
11    pid CHAR(10),  
12    info CHAR(10),  
13    data_of_attend DATE,  
14 )
```

Q 1.2

Continue 1, each patient attends doctors at most once.

```
1 CREATE TABLE Doctors(  
2     did CHAR(10),  
3     PRIMARY KEY (did)  
4 )  
5 CREATE TABLE Patients(  
6     pid CHAR(10),  
7     PRIMARY KEY (pid)  
8 )  
9 CREATE TABLE Record(  
10    did CHAR(10),  
11    pid CHAR(10),  
12    info CHAR(10),  
13    data_of_attend DATE,  
14    PRIMARY KEY (pid),  
15 )
```

Q 1.3

Continue 2, each patient attends doctors at least once.

```
1 CREATE TABLE Doctors(  
2     did CHAR(10),  
3     PRIMARY KEY (did)
```

```

2      did CHAR(10),
3      PRIMARY KEY (did)
4  )
5  CREATE TABLE Patients(
6      pid CHAR(10),
7      PRIMARY KEY (pid)
8  )
9  CREATE TABLE Record(
10     did CHAR(10) NOT NULL,
11     pid CHAR(10),
12     info CHAR(10),
13     data_of_attend DATE,
14     PRIMARY KEY (pid),
15 )

```

Q 1.4

Continue 1, only existing doctors can be attended by a patient.

```

1  CREATE TABLE Doctors(
2      did CHAR(10),
3      PRIMARY KEY (did)
4  )
5  CREATE TABLE Patients(
6      pid CHAR(10),
7      PRIMARY KEY (pid)
8  )
9  CREATE TABLE Record(
10     did CHAR(10),
11     pid CHAR(10),
12     info CHAR(10),
13     data_of_attend DATE,
14     PRIMARY KEY (did,pid),
15     FOREIGN KEY (did) REFERENCES Doctors
16 )

```

Q 1.5

Continue 1, every doctor must be attended by a patient

```

1 CREATE TABLE Record_Merge_Doctor(
2     did CHAR(10),
3     pid CHAR(10) Not NULL,
4     info CHAR(10),
5     data_of_attend DATE,
6     PRIMARY KEY (did),
7 )
8 CREATE TABLE Patients(
9     pid CHAR(10),
10    PRIMARY KEY (pid)
11 )

```

Q 1.6

Continue 1, each of (Name, Address) and (Name, Age) uniquely identifies a patient.

```

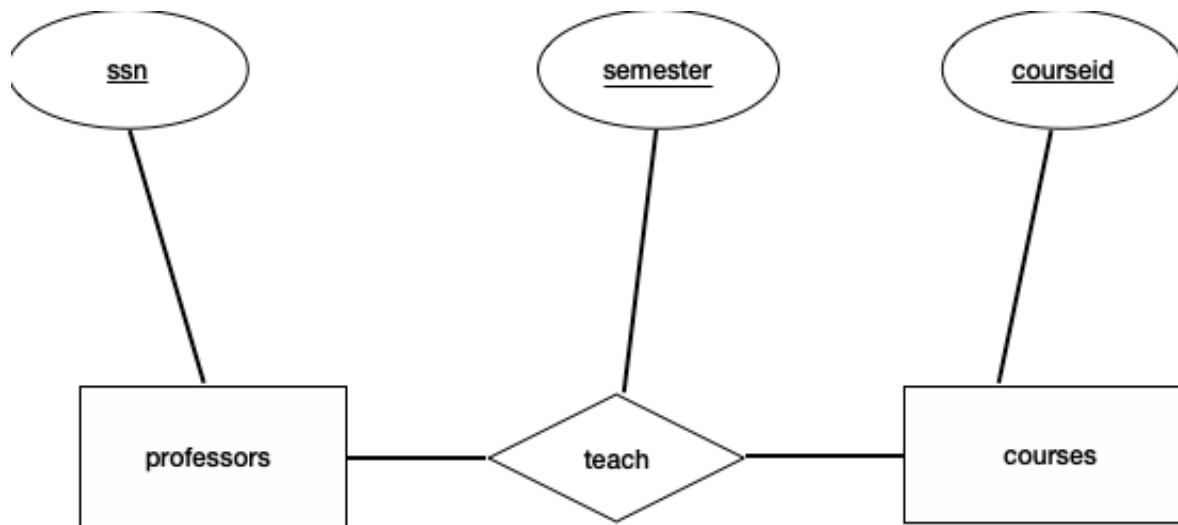
1 CREATE TABLE Doctors(
2     did CHAR(10),
3     PRIMARY KEY (did)
4 )
5 CREATE TABLE Patients(
6     pid CHAR(10),
7     Name CHAR(10),
8     Age CHAR(10),
9     Address CHAR(10),
10    PRIMARY KEY (pid),
11    UNIQUE (Name, Address),
12    UNIQUE (Name, Age)
13 )
14 CREATE TABLE Record(
15     did CHAR(10),
16     pid CHAR(10),
17     info CHAR(10),
18     data_of_attend DATE,
19 )

```

Q 2

Q 2.1

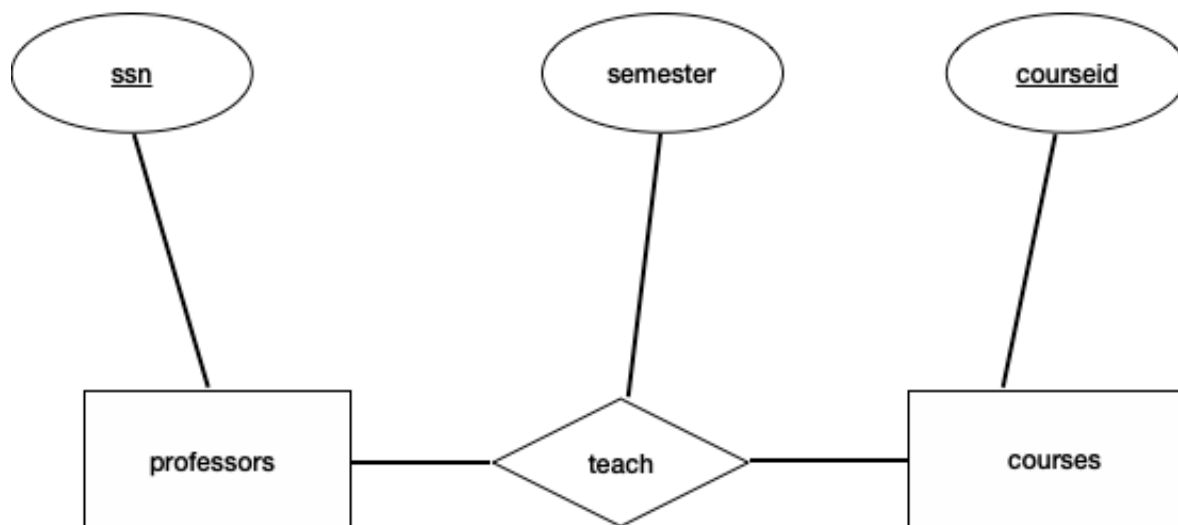
Professors can teach the same course in several semesters, and each offering must be recorded.



```
1 CREATE TABLE professors(  
2     ssn CHAR(10),  
3     PRIMARY KEY (ssn)  
4 )  
5 CREATE TABLE courses(  
6     courseid CHAR(10),  
7     PRIMARY KEY (courseid)  
8 )  
9 CREATE TABLE teach(  
10    ssn CHAR(10),  
11    courseid CHAR(10),  
12    semester CHAR(10),  
13    PRIMARY KEY (ssn, courseid, semester)  
14 )
```

Q 2.2

Professors can teach the same course in several semesters, and only the most recent such offering needs to be recorded. (Assume this condition applies in all subsequent questions.)



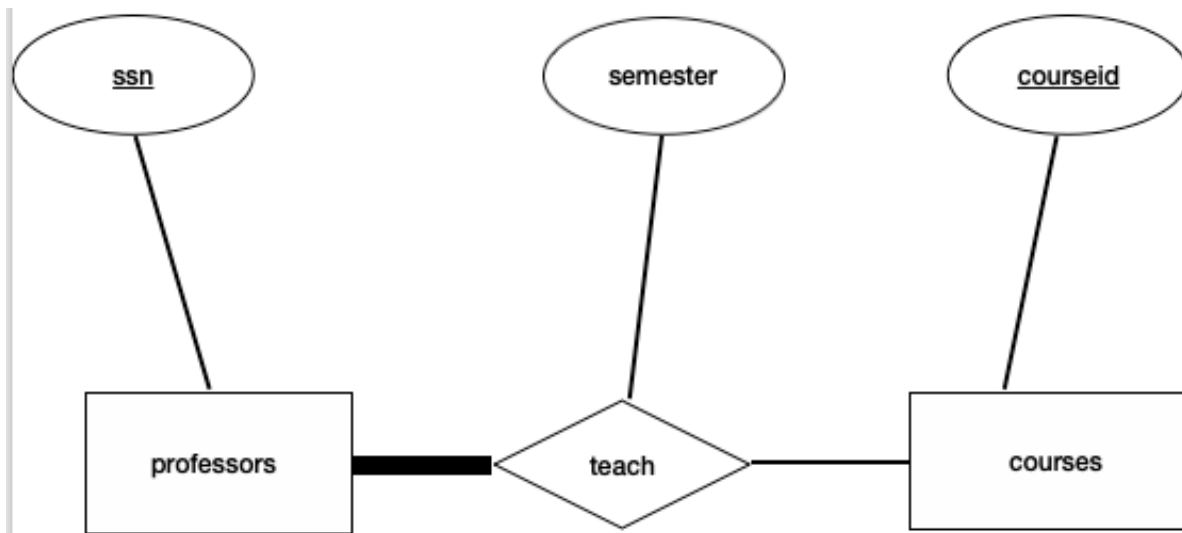
```

1 CREATE TABLE professors(
2     ssn CHAR(10),
3     PRIMARY KEY (ssn)
4 )
5 CREATE TABLE courses(
6     courseid CHAR(10),
7     PRIMARY KEY (courseid)
8 )
9 CREATE TABLE teach(
10    ssn CHAR(10),
11    courseid CHAR(10),
12    semester CHAR(10),
13    PRIMARY KEY (ssn, courseid)
14 )

```

Q 2.3

Every professor must teach some course.



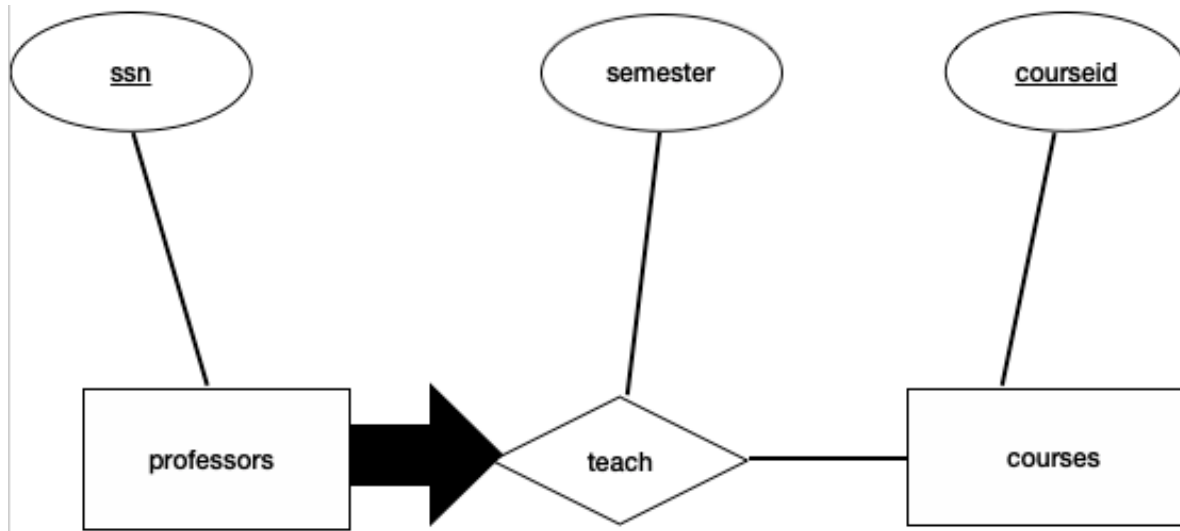
```

1 CREATE TABLE courses(
2     courseid CHAR(10),
3     PRIMARY KEY (courseid)
4 )
5 CREATE TABLE teach_professors(
6     ssn CHAR(10),
7     courseid CHAR(10) NOT NULL,
8     semester CHAR(10),
9     PRIMARY KEY (ssn, courseid)
10    FOREIGN KEY (courseid) REFERENCES courses
11 )

```

Q 2.4

Every professor teaches exactly one course (no more, no less).

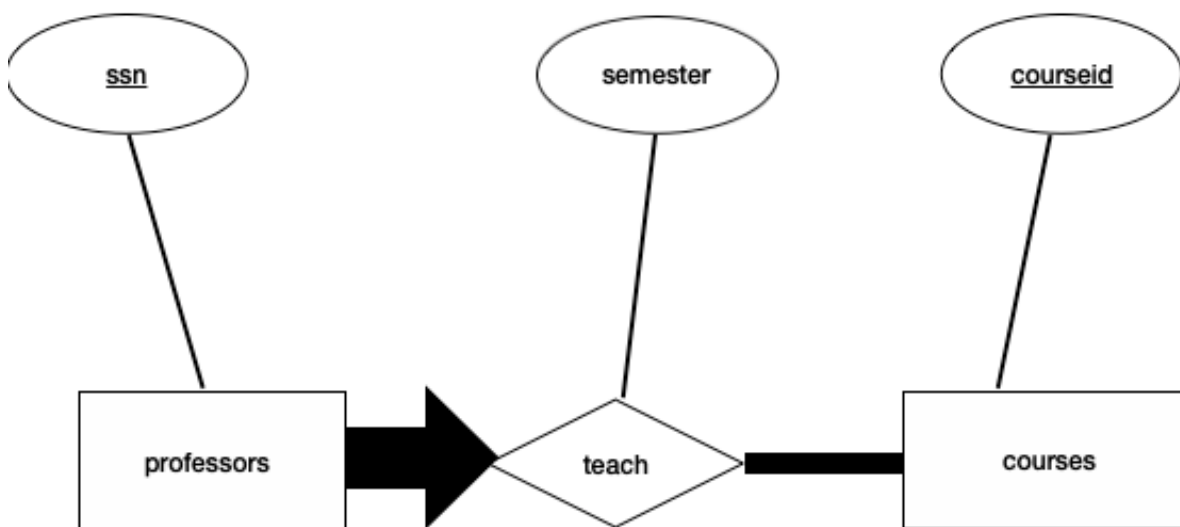


```

1 CREATE TABLE courses(
2     courseid CHAR(10),
3     PRIMARY KEY (courseid)
4 )
5 CREATE TABLE teach_professors(
6     ssn CHAR(10),
7     courseid CHAR(10) NOT NULL,
8     semester CHAR(10),
9     PRIMARY KEY (ssn)
10    FOREIGN KEY (courseid) REFERENCES courses
11 )
  
```

Q 2.5

Every professor teaches exactly one course (no more, no less), and every course must be taught by some professor.



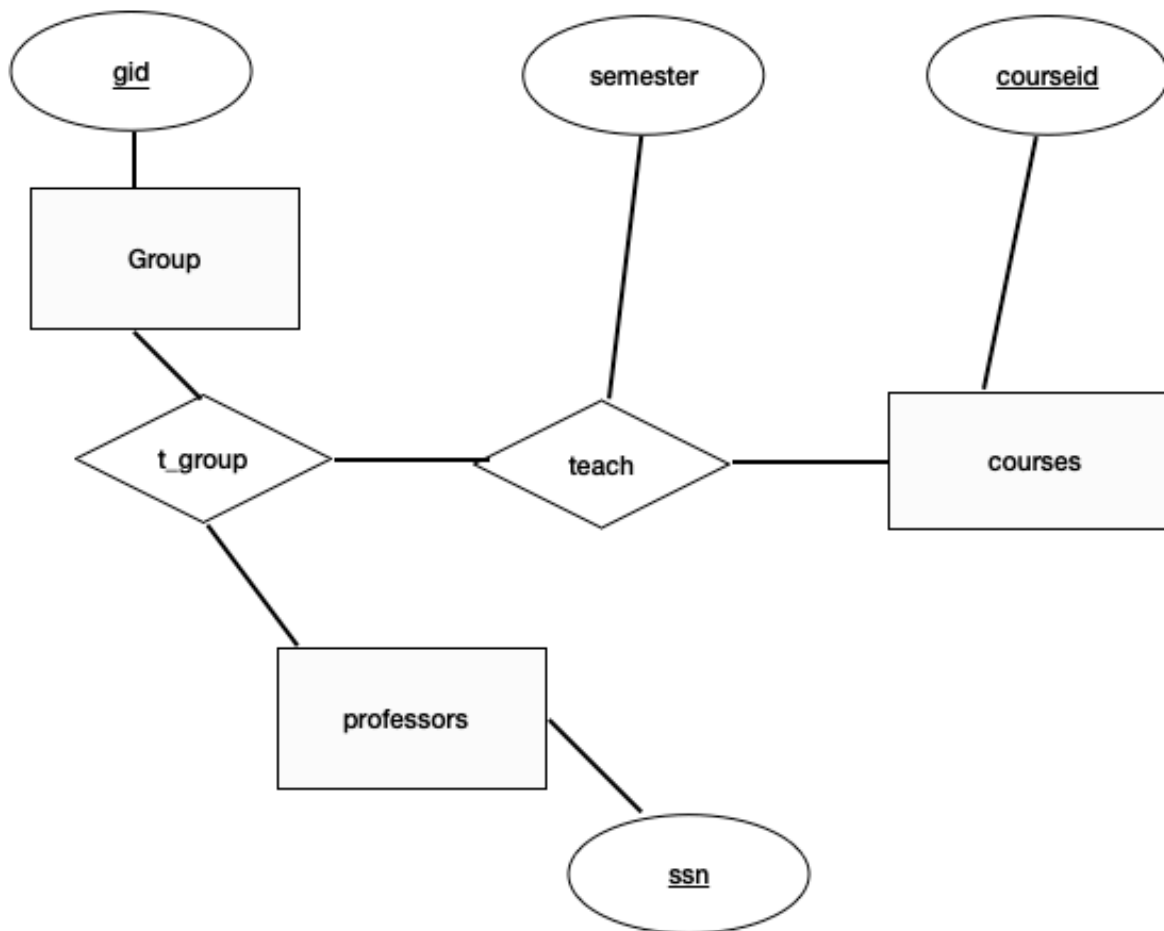
```

1 CREATE TABLE teach_courses(
2     courseid CHAR(10),
3     ssn CHAR(10) NOT NULL,
4     PRIMARY KEY (courseid, ssn)
5     FOREIGN KEY (ssn) REFERENCES teach_professors
6 )
7 CREATE TABLE teach_professors(
8     ssn CHAR(10),
9     courseid CHAR(10) NOT NULL,
10    semester CHAR(10),
11    PRIMARY KEY (ssn)
12    FOREIGN KEY (courseid) REFERENCES teach_courses
13 )

```

Q 2.6

Now suppose that certain courses can be taught by a team of professors jointly, but it is possible that no one professor in a team can teach the course. Model this situation, introducing additional entity sets and relationship sets if necessary.



```

1 CREATE TABLE professors(
2     ssn CHAR(10),
3     PRIMARY KEY (ssn)
4 )

```

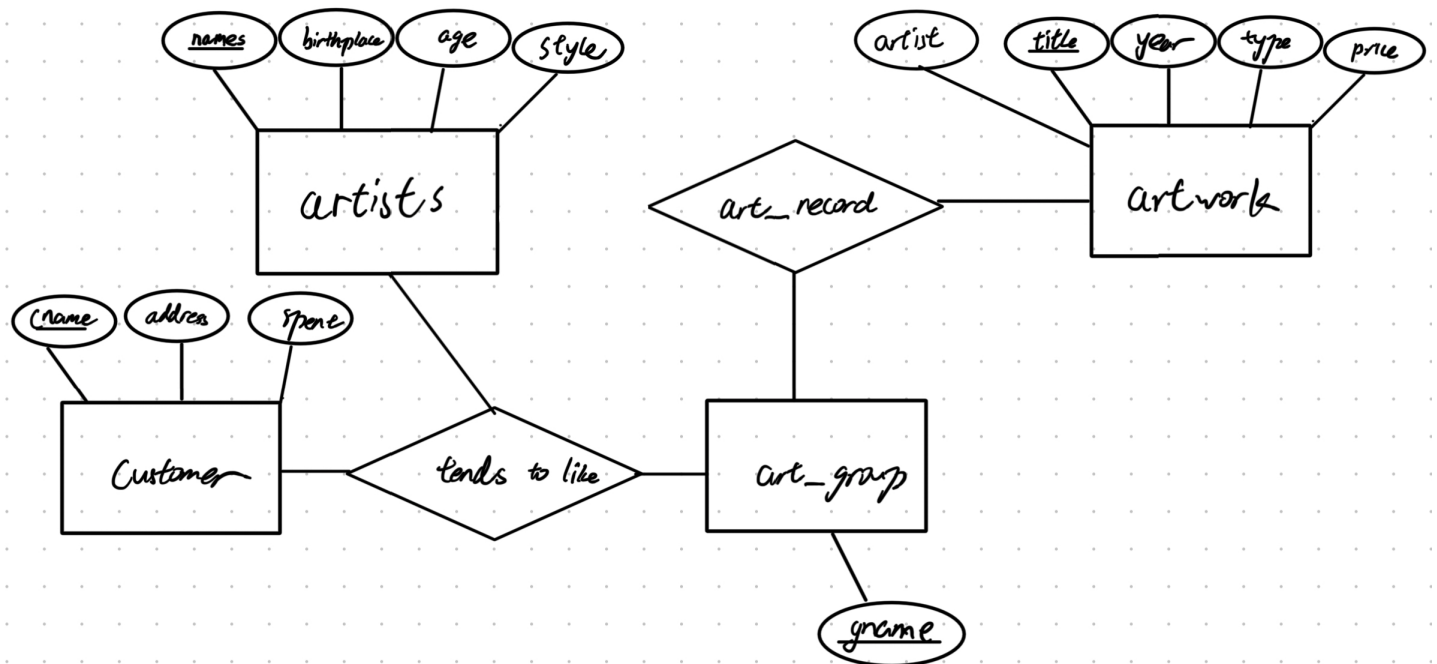
```

5 CREATE TABLE courses(
6     courseid CHAR(10),
7     PRIMARY KEY (courseid)
8 )
9 CREATE TABLE group(
10    gid CHAR(10),
11    PRIMARY KEY (gid)
12 )
13 CREATE TABLE t_group(
14    gid CHAR(10),
15    ssn CHAR(10),
16    PRIMARY KEY (gid,ssn)
17    FOREIGN KEY (gid) REFERENCES group
18    FOREIGN KEY (ssn) REFERENCES professors
19 )
20 CREATE TABLE teach(
21    gid CHAR(10),
22    courseid CHAR(10),
23    PRIMARY KEY (gid,courseid)
24    FOREIGN KEY (gid) REFERENCES t_group
25    FOREIGN KEY (courseid) REFERENCES courses
26 )

```

Q3

Q 3.1



Q 3.2

```

1 CREATE TABLE artists(

```

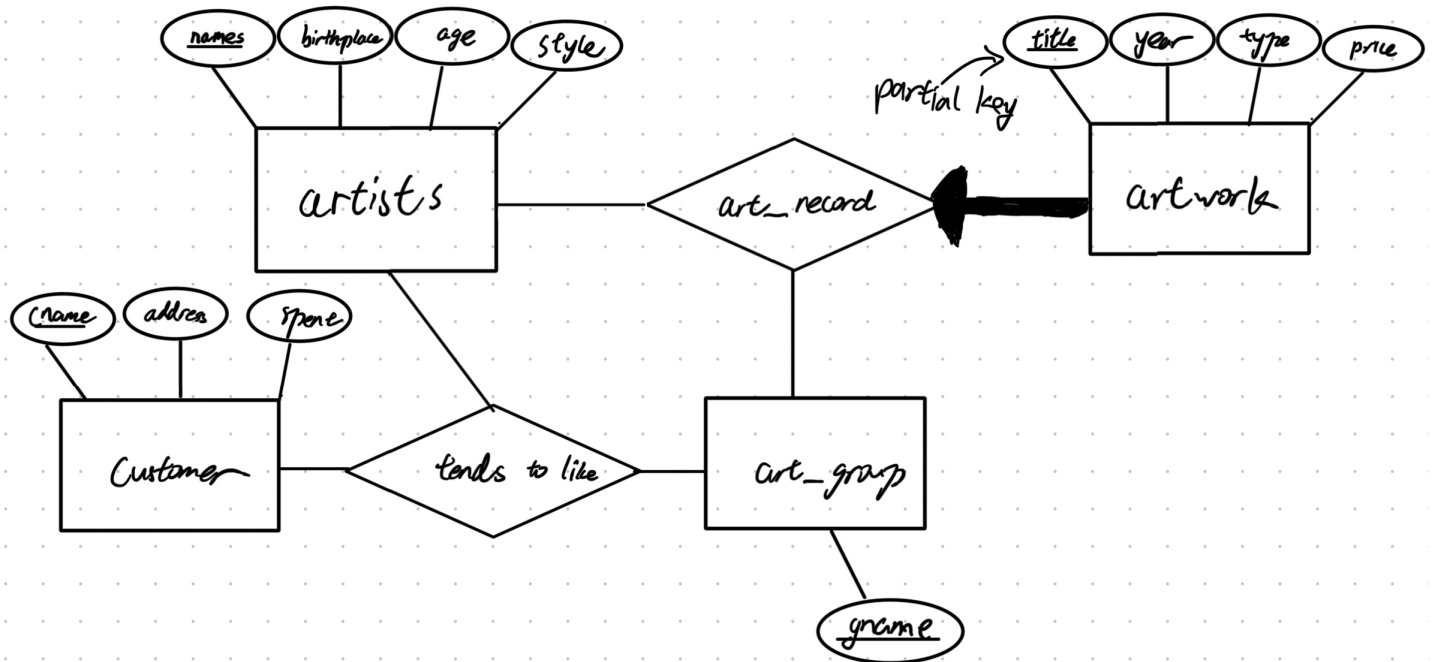


```

2      name CHAR(10),
3      birthplace CHAR(10),
4      age INTEGER,
5      sytle CHAR(10),
6      PRIMARY KEY (name)
7  )
8  CREATE TABLE artwork(
9      title CHAR(10),
10     year CHAR(10),
11     type CHAR(10),
12     artist CHAR(10),
13     price INTEGER,
14     PRIMARY KEY (title)
15 )
16 CREATE TABLE art_group(
17     gname CHAR(10),
18     PRIMARY KEY (gname)
19 )
20 CREATE TABLE customer(
21     cname CHAR(10),
22     address CHAR(10),
23     spent INTEGER,
24     PRIMARY KEY (cname)
25 )
26 CREATE TABLE art_record(
27     title CHAR(10),
28     gname CHAR(10),
29     PRIMARY KEY (title,gname)
30     FOREIGN KEY (title) REFERENCES artwork
31     FOREIGN KEY (gname) REFERENCES art_group
32 )
33 CREATE TABLE tends_to_like(
34     name CHAR(10),
35     cname CHAR(10),
36     gname CHAR(10),
37     PRIMARY KEY (naem,cname,gname)
38     FOREIGN KEY (name) REFERENCES artists
39     FOREIGN KEY (cname) REFERENCES customer
40     FOREIGN KEY (gname) REFERENCES art_group
41 )

```

Q 3.3



- Delete the artist attribute of artwork entity.
- Set the title as partial key.
- Link artists entity into art_record relationship.
- Create one to many relationship.
- Make names as foreign key
- Merge art_record and artwork as artwork_record:

```

1  CREATE TABLE artwork_record(
2      title CHAR(10),
3      year CHAR(10),
4      type CHAR(10),
5      price INTEGER,
6
7      name CHAR(10),
8      gname CHAR(10),
9      PRIMARY KEY (title,name,gname)
10     FOREIGN KEY (name) REFERENCES artists
11 )
  
```