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.

```
CREATE TABLE Doctors(
    did CHAR(10),
    PRIMARY KEY (did)
)
CREATE TABLE Patients(
    pid CHAR(10),
    PRIMARY KEY (pid)
)
CREATE TABLE Record(
    did CHAR(10),
    pid CHAR(10),
    info CHAR(10),
    data_of_attend DATE,
)
```

Q 1.2

Continue 1, each patient attends doctors at most once.

```
CREATE TABLE Doctors(
    did CHAR(10),
    PRIMARY KEY (did)
)
CREATE TABLE Patients(
    pid CHAR(10),
    PRIMARY KEY (pid)
)
CREATE TABLE Record(
    did CHAR(10),
    pid CHAR(10),
    info CHAR(10),
    data_of_attend DATE,
    PRIMARY KEY (pid),
)
```

Q 1.3

Continue 2, each patient attends doctors at least once.

```
CREATE TABLE Doctors(
```

```
    did CHAR(10),
    PRIMARY KEY (did)
)
CREATE TABLE Patients(
    pid CHAR(10),
    PRIMARY KEY (pid)
)
CREATE TABLE Record(
    did CHAR(10) NOT NULL,
    pid CHAR(10),
    info CHAR(10),
    data_of_attend DATE,
    PRIMARY KEY (pid),
)
```

Q 1.4

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

```
CREATE TABLE Doctors(
    did CHAR(10),
    PRIMARY KEY (did)
)
CREATE TABLE Patients(
    pid CHAR(10),
    PRIMARY KEY (pid)
)
CREATE TABLE Record(
    did CHAR(10),
    pid CHAR(10),
    info CHAR(10),
    data_of_attend DATE,
    PRIMARY KEY (did,pid),
    FOREIGN KEY (did) REFERENCES Doctors
)
```

Q 1.5

Continue 1, every doctor must be attended by a patient

```
CREATE TABLE Record_Merge_Doctor(
    did CHAR(10),
    pid CHAR(10) Not NULL,
    info CHAR(10),
    data_of_attend DATE,
    PRIMARY KEY (did),
)
CREATE TABLE Patients(
    pid CHAR(10),
    PRIMARY KEY (pid)
)
```

Q 1.6

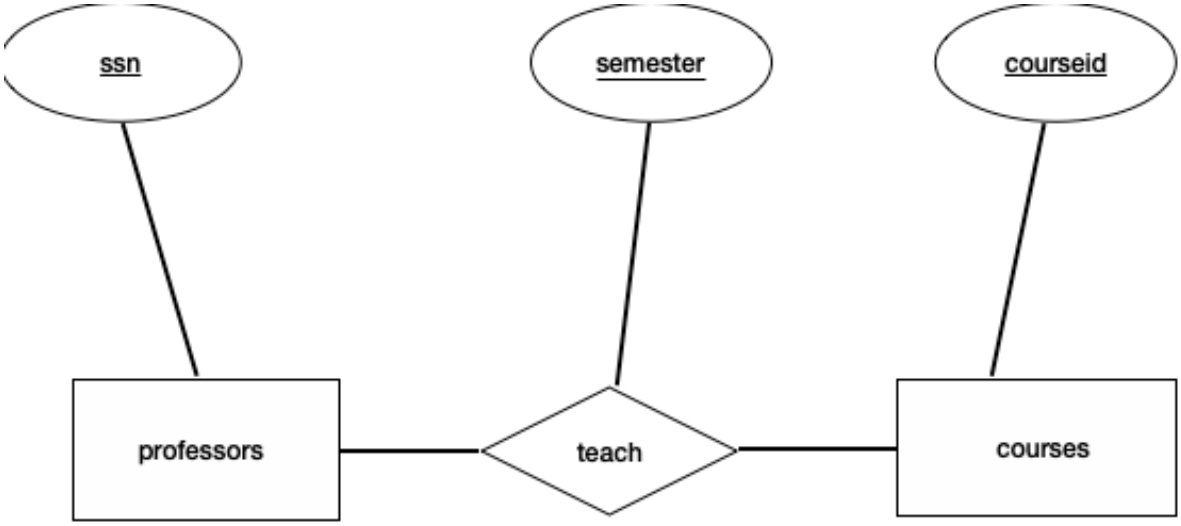
Continue 1, each of (Name, Address) and (Name,Age) uniquely identifies a patient.

```
CREATE TABLE Doctors(
    did CHAR(10),
    PRIMARY KEY (did)
)
CREATE TABLE Patients(
    pid CHAR(10),
    Name CHAR(10),
    Age CHAR(10),
    Address CHAR(10),
    PRIMARY KEY (pid),
    UNIQUE (Name,Address),
    UNIQUE (Name,Age)
)
CREATE TABLE Record(
    did CHAR(10),
    pid CHAR(10),
    info CHAR(10),
    data_of_attend DATE,
)
```

Q 2

Q 2.1

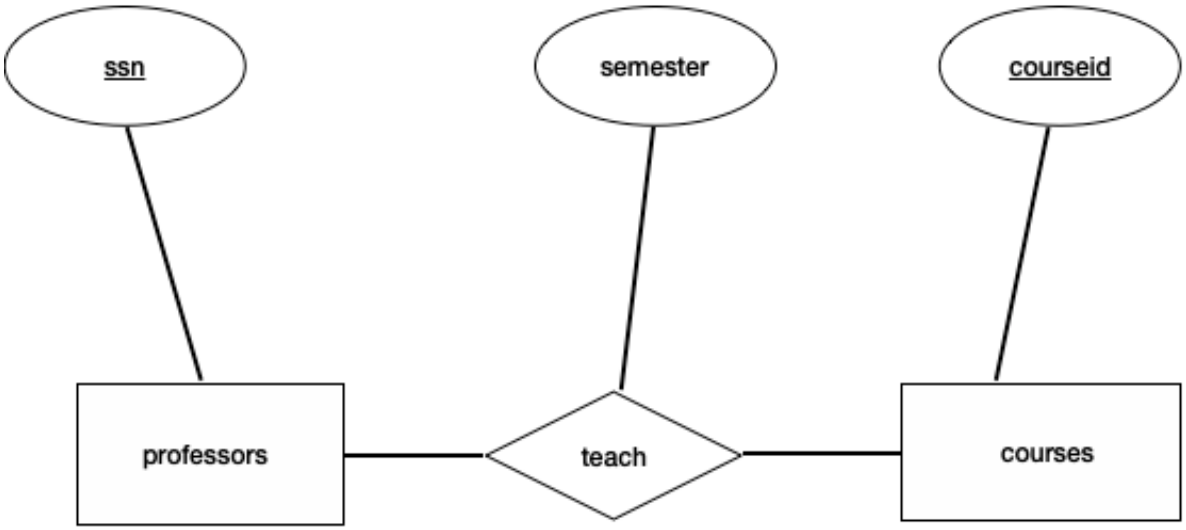
Professors can teach the same course in several semesters, and each offering must be recorded.




```sql
CREATE TABLE professors(
    ssn CHAR(10),
    PRIMARY KEY (ssn)
)
CREATE TABLE courses(
    courseid CHAR(10),
    PRIMARY KEY (courseid)
)
CREATE TABLE teach(
    ssn CHAR(10),
    courseid CHAR(10),
    semester CHAR(10),
    PRIMARY KEY (ssn, courseid, semester)
)
```

Q 2.2

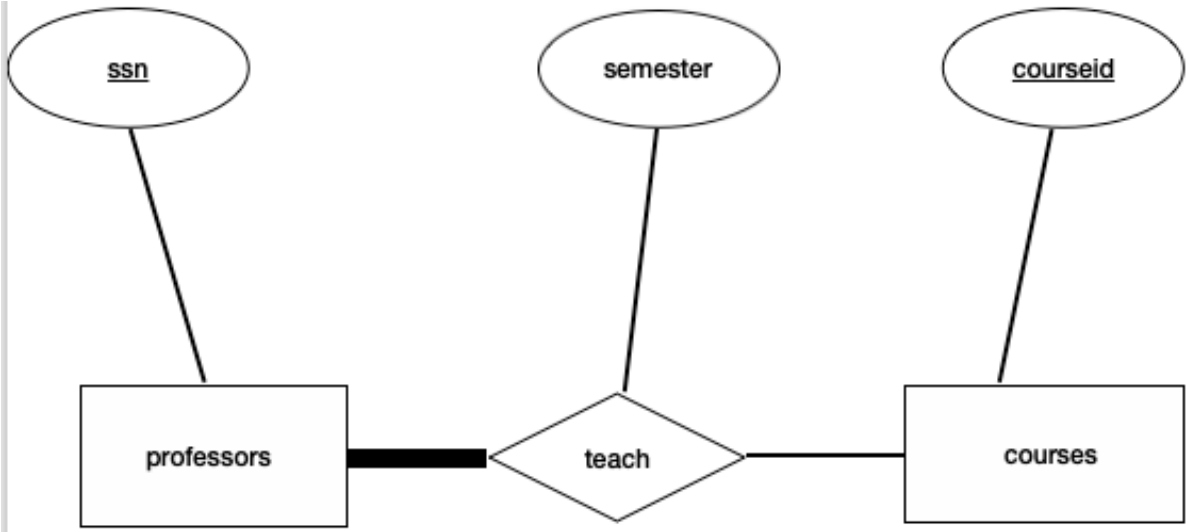
Professors can teach the same course in several semesters, and only the most recent such offering needs to be recorded. (Assume this condition applies in all subsequent questions.)



```
CREATE TABLE professors(
    ssn CHAR(10),
    PRIMARY KEY (ssn)
)
CREATE TABLE courses(
    courseid CHAR(10),
    PRIMARY KEY (courseid)
)
CREATE TABLE teach(
    ssn CHAR(10),
    courseid CHAR(10),
    semester CHAR(10),
    PRIMARY KEY (ssn, courseid)
)
```

Q 2.3

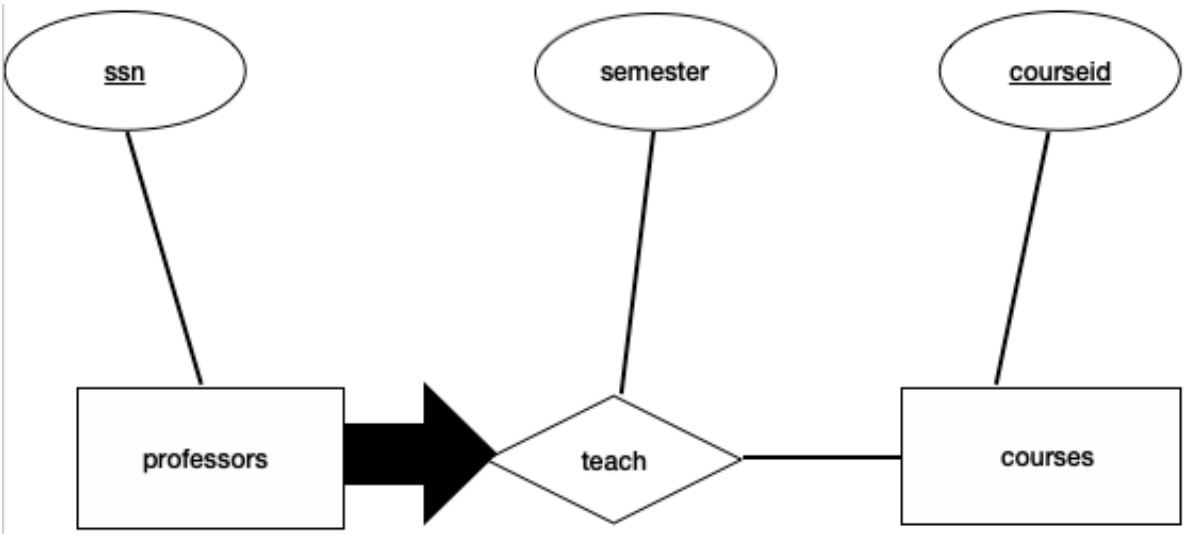
Every professor must teach some course.




```
CREATE TABLE courses(
    courseid CHAR(10),
    PRIMARY KEY (courseid)
)
CREATE TABLE teach_professors(
    ssn CHAR(10),
    courseid CHAR(10) NOT NULL,
    semester CHAR(10),
    PRIMARY KEY (ssn, courseid)
    FOREIGN KEY (courseid) REFERENCES courses
)
```

Q 2.4

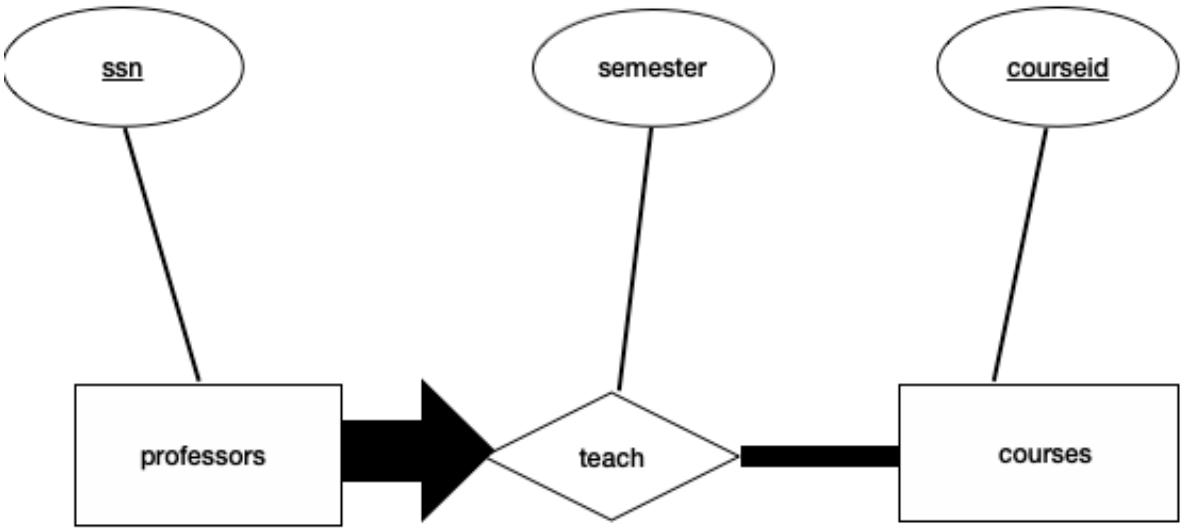
Every professor teaches exactly one course (no more, no less).




```sql
CREATE TABLE courses(
    courseid CHAR(10),
    PRIMARY KEY (courseid)
)
CREATE TABLE teach_professors(
    ssn CHAR(10),
    courseid CHAR(10) NOT NULL,
    semester CHAR(10),
    PRIMARY KEY (ssn)
    FOREIGN KEY (courseid) REFERENCES courses
)
```

Q 2.5

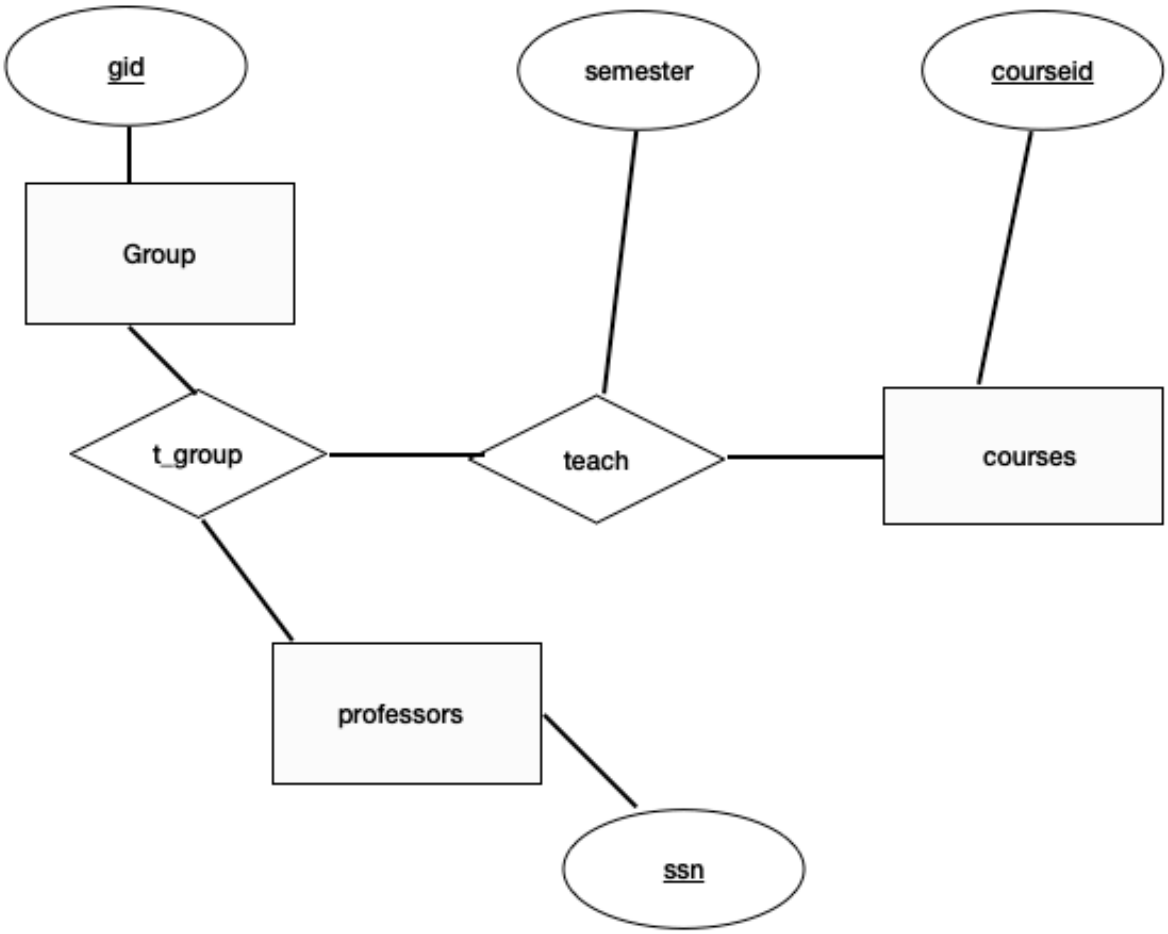
Every professor teaches exactly one course (no more, no less), and every course must be taught by some professor.



```
CREATE TABLE teach_courses(
    courseid CHAR(10),
    ssn CHAR(10) NOT NULL,
    PRIMARY KEY (courseid, ssn)
    FOREIGN KEY (ssn) REFERENCES teach_professors
)
CREATE TABLE teach_professors(
    ssn CHAR(10),
    courseid CHAR(10) NOT NULL,
    semester CHAR(10),
    PRIMARY KEY (ssn)
    FOREIGN KEY (courseid) REFERENCES teach_courses
)
```

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.

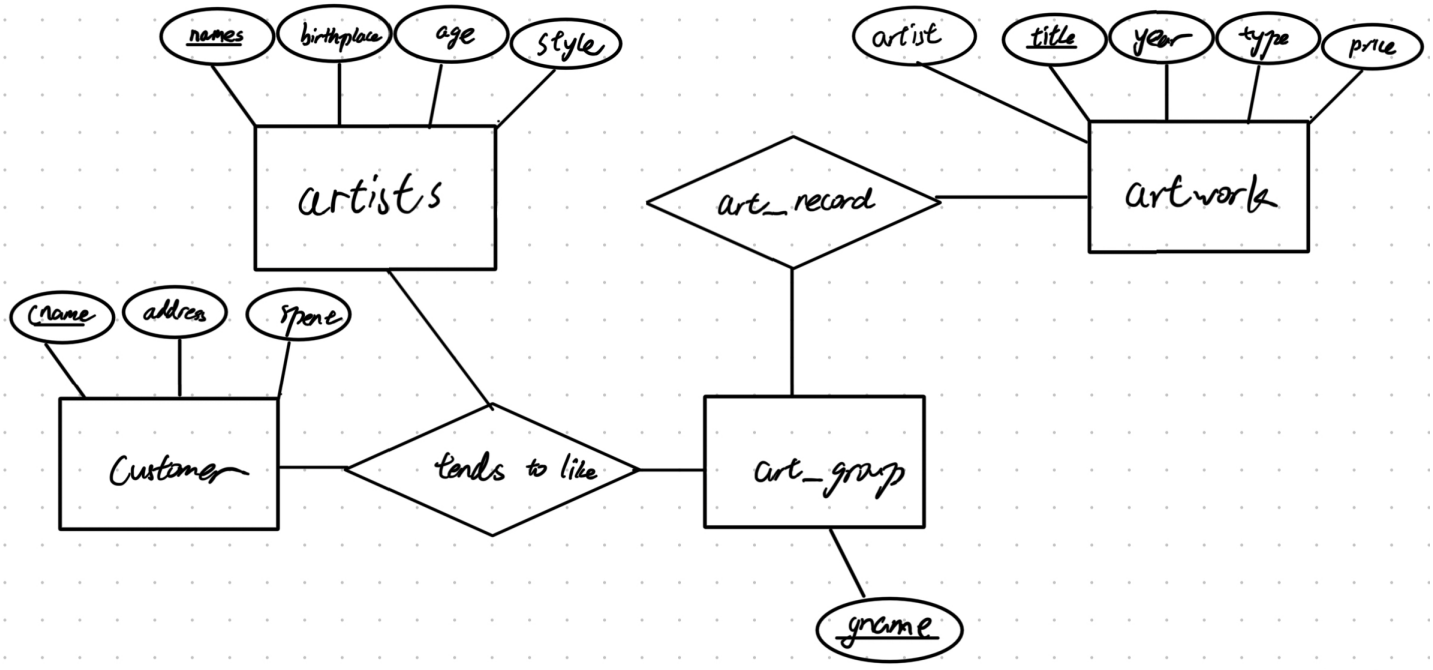



```
CREATE TABLE professors(
    ssn CHAR(10),
    PRIMARY KEY (ssn)
)
```

```
CREATE TABLE courses(
    courseid CHAR(10),
    PRIMARY KEY (courseid)
)
CREATE TABLE group(
    gid CHAR(10),
    PRIMARY KEY (gid)
)
CREATE TABLE t_group(
    gid CHAR(10),
    ssn CHAR(10),
    PRIMARY KEY (gid,ssn)
    FOREIGN KEY (gid) REFERENCES group
    FOREIGN KEY (ssn) REFERENCES professors
)
CREATE TABLE teach(
    gid CHAR(10),
    courseid CHAR(10),
    PRIMARY KEY (gid,courseid)
    FOREIGN KEY (gid) REFERENCES t_group
    FOREIGN KEY (courseid) REFERENCES courses
)
```

Q3

Q 3.1

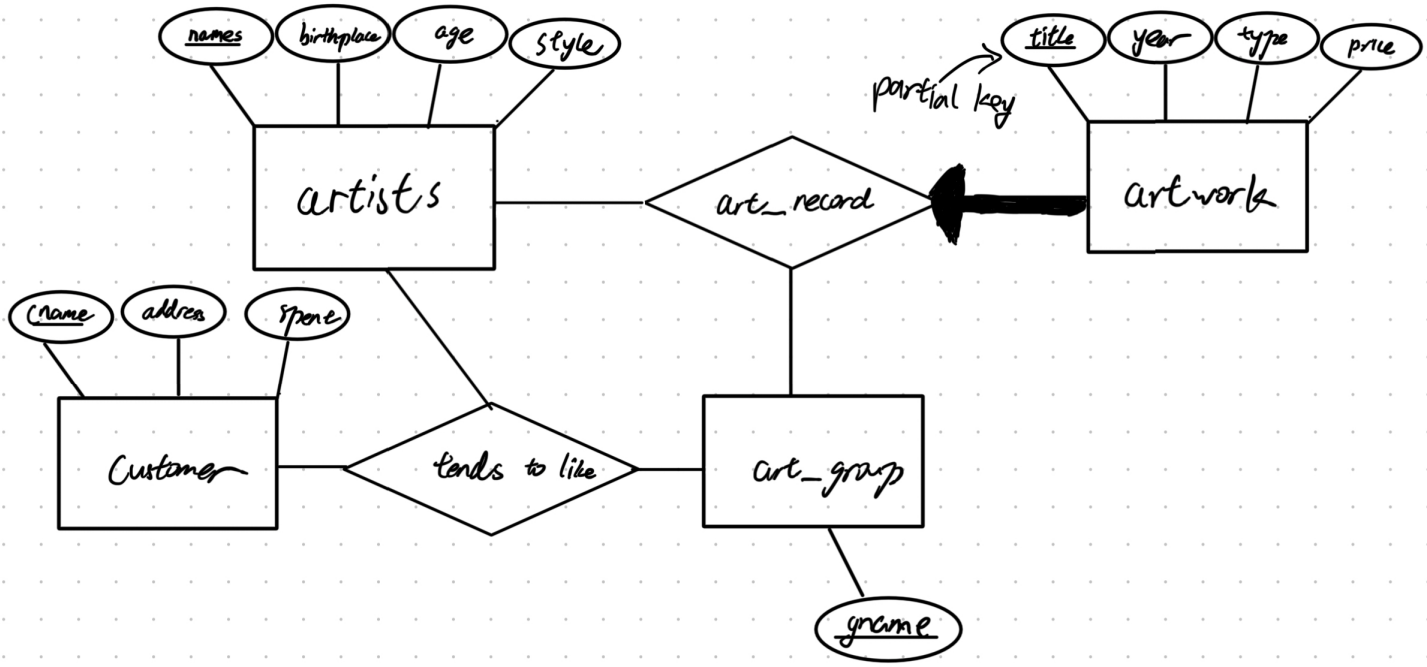


Q 3.2

```
CREATE TABLE artists(
```

```
    name CHAR(10),
    birthplace CHAR(10),
    age INTEGER,
    sytle CHAR(10),
    PRIMARY KEY (name)
)
CREATE TABLE artwork(
    title CHAR(10),
    year CHAR(10),
    type CHAR(10),
    artist CHAR(10),
    price INTEGER,
    PRIMARY KEY (title)
)
CREATE TABLE art_group(
    gname CHAR(10),
    PRIMARY KEY (gname)
)
CREATE TABLE customer(
    cname CHAR(10),
    address CHAR(10),
    spent INTEGER,
    PRIMARY KEY (cname)
)
CREATE TABLE art_record(
    title CHAR(10),
    gname CHAR(10),
    PRIMARY KEY (title,gname)
    FOREIGN KEY (title) REFERENCES artwork
    FOREIGN KEY (gname) REFERENCES art_group
)
CREATE TABLE tends_to_like(
    name CHAR(10),
    cname CHAR(10),
    gname CHAR(10),
    PRIMARY KEY (naem,cname,gname)
    FOREIGN KEY (name) REFERENCES artists
    FOREIGN KEY (cname) REFERENCES customer
    FOREIGN KEY (gname) REFERENCES art_group
)
```

Q 3.3



- Delet the artiist 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:

```
CREATE TABLE artwork_record(
    title CHAR(10),
    year CHAR(10),
    type CHAR(10),
    price INTEGER,

    name CHAR(10),
    gname CHAR(10),
    PRIMARY KEY (title,name,gname)
    FOREIGN KEY (name) REFERENCES artists
)
```