Hw2

~~3.2~~, ~~3.4~~, ~~3.5~~, ~~3.6~~, 3.7, 3.12, 3.15, 3.16 9/30/19

(3.2) # of tuples w/ cardinality 22?

22, because Cardinality = # of tuples

(3.4) A candidate is the minimum set of fields to uniquely identify each tuple, while the primary key is the candidate key that is enforced by DBMS. So, not all candidate keys are primary keys, but every primary key is a candidate key.

The set of nondescriptive attributes is a Super Key. If there are no constraints, this set of attributes is a candidate key.

(3.5)

1. From this set, name + age are Not Candidate keys (But, GPA should also not be as well as it can repeat)
2. ~~SSN~~ (student id) and email are candidate keys ~~in this instance of~~ the relation. However, they may not be for the entire relation but since we only have this snapshot of the database we cannot make assumptions for the entire database based on ~~this~~ one instance. If there are more fields to consider, the answer may change.

(3.6) A foreign key constraint states that every value of x in current table, must also appear in the table that x is key referenced from. It has the same attributes and type as the primary key it makes reference to.

These constraints are important because if they are not met, then ~~accessing~~ the ~~data~~ field that the foreign key is referencing can lead to errors and null values. By upholding these, we can have accurate, reliable + consistent data.

Referential Integrity: ensures the accuracy and consistency of data in a relationship. It requires that when a foreign key is used, it must ~~access~~ reference a valid, existing primary key.

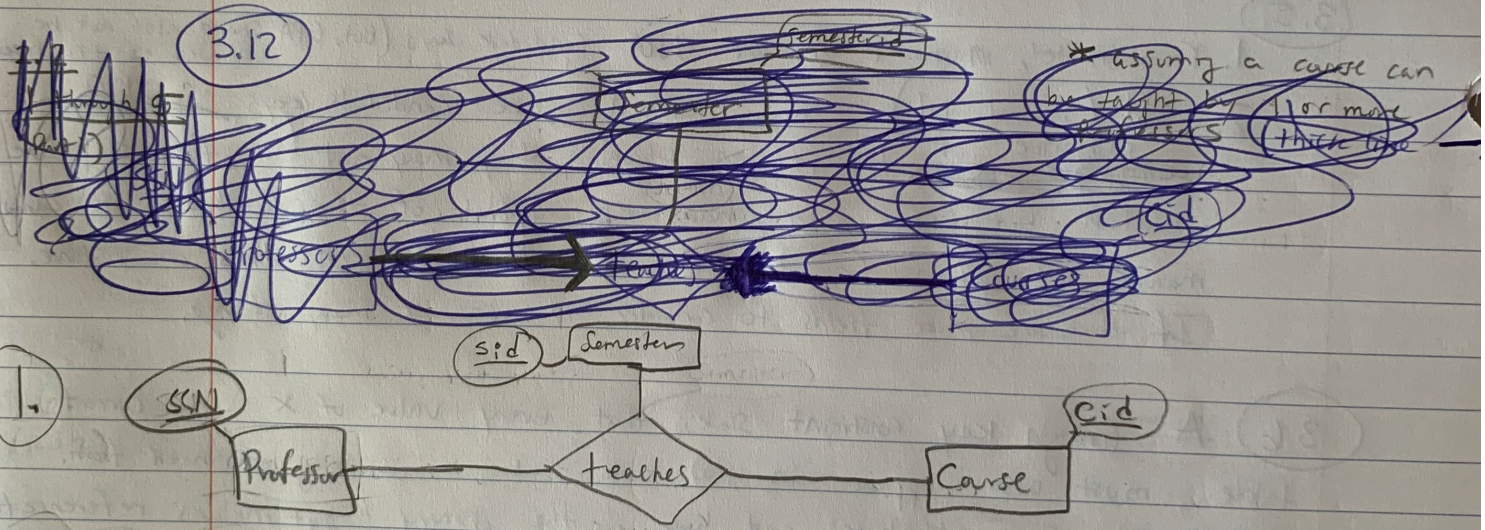
cont. ~~3.7~~, 3.12, 3.15, 3.16

3.7

1. Enrolled → foreign key (sid) references Students
   → foreign key (cid) references Courses
   Teaches → foreign key (fid references Faculty
   → foreign key (cid) references Courses
   Meets_In → foreign key (cid) references Courses
   → foreign key (rno) references Rooms

2. In the enrolled ~~relationship~~, the grade field should be an enumerated type that limits it from A – F.

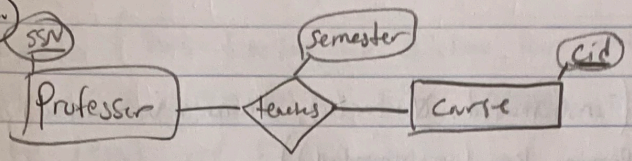
3.12



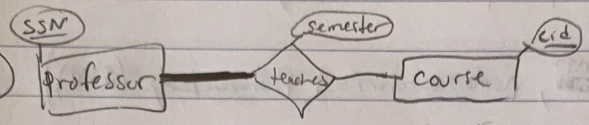
Create Table Professor (SSN: int, Primary key (SSN))
Semester (sid: int, Primary key (sid))
Course (cid: int, Primary key (cid))
teaches (SSN: int, Sid: int, Cid: int, Primary key (SSN, Sid, Cid),
Foreign key (SSN) references Professor,
Foreign key (Sid) references Semester,
Foreign key (Cid) References Course)

Professor (SSN: int, primary key (SSN))
Course (Cid: int, primary key (cid))
teaches (SSN: int, Cid: int, Primary key (SSN, Cid),
Semester: varchar (10),
Foreign key (SSN) references Professor,
Foreign key (cid) references Course)

2.

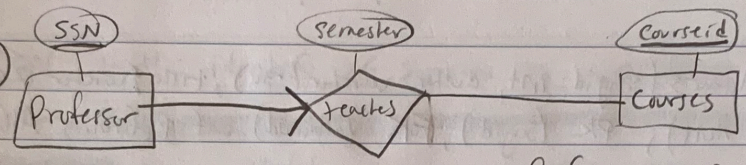


3.



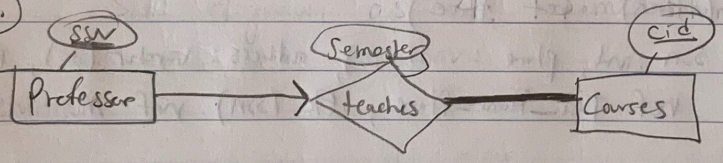
Professor + Course are same as in 2
teaches (SSN: int, cid: int, Primary key (SSN, cid),
Semester: varchar(10), Foreign key (SSN) references Professor,
Foreign key (cid) references Course)

4.



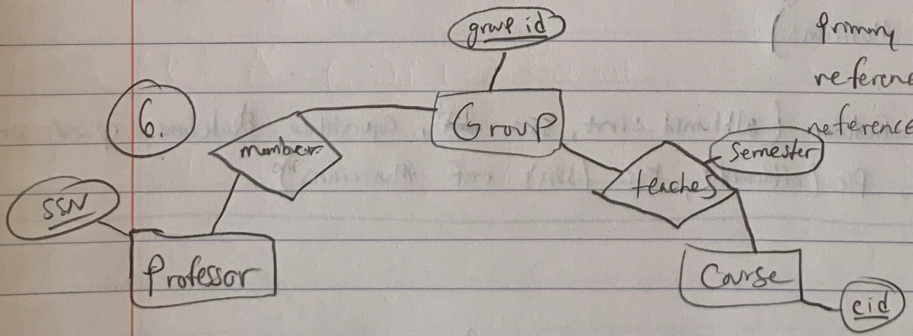
Professor + Courses are same as in 2.
teaches' primary key is now only (SSN) and no longer (SSN, Cid).

5.



Professor and courses are same as in 2.
teaches (SSN: int, cid: int, Semester: varchar(10),
primary key (SSN), Foreign key (SSN) references Professor, Foreign key (cid) references Courses)

6.



Professor (SSN: int, primary key (SSN))
Course (cid: ind, primary key (cid))
Group (grapid: int, primary key (grapid))
member (SSN: ind, grapid: int, primary key (SSN, grapid), Foreign key (SSN) references Professor,
Foreign key (grapid references Group)
teaches (Semester: varchar(10), Cid: int, grapid: int, primary key (cid, grapid), Foreign key (cid) references Courses,
foreign key (grapid) references Group)

Cont. 3.15, 3.16

3.15

Create Table Instruments(instrumentid: varchar(10), dname: varchar(30), key: varchar(5), primary key (instrumentid))

Create Table Musicians (SSN: int, name: varchar(30), Primary key (SSN))

Create Table Plays (SSN: int, instrumentid: varchar(10), Primary key (SSN, instrumentid), Foreign key (ssn) references Musicians, Foreign key (instrumentid) references Instruments)

Create Table Songs Apears (Songid: int, author: varchar(30), title varchar(25), albumid: int (albumid cannot be null), pk (songid), Fk (albumid) ref References Album_Producer)

Create Table [redacted] Place (address: varchar(50))

Create Table Tel_Home_Phone(phone: varchar(10), address: varchar(30), Pk (phone), Fk (address) ref Place)

Create Table Lives (ssn: int, phone: varchar(10), address: varchar(30), Pk (ssn, address), Fk (phone, address) ref Tel_Home_Phone, Fk (ssn) ref Musicians)

Create Table Perform (Songid: int, ssn: int, Pk (ssn, songid), Fk (ssn) ref Songs, Fk (ssn) ref Musicians)

Create Table Album_Producer (albumid: int, ssn: int, Copyrightdate: datetime, speed: int, title: varchar(30), Pk (albumid), Fk (ssn) ref Musicians)

3.16

Create Table Model (weight: int, model #: int, capacity: int, Pk (model#))

Create Table Test (FAA#: int, score: float, name: varchar(20), Pk (FAA#))

Create Table Plane (reg#: int, Pk (reg#))

Create Table Employees (union mem. #: int, SSN: int, name: varchar(20), Salary: int, Phone#: varchar(10), address: varchar(50), exam date: datetime, Pk (SSN))

Create Table testinfo (hrs: datetime, date: datetime, score: int, [redacted] SSN: int, reg#: int, FAA#: int, Pk (SSN, reg#, FAA#), Fk (FAA#) ref Test, Fk (reg#) ref Plane, Fk (SSN) ref Employees)

Create Table Type (model#: int, reg#: int, Pk(reg#), Fk (model#) ref Model, Fk (reg#) ref Plane)

Create Table isAnExpert(SSN: int, model#: int, Pk (SSN, model#), Fk (SSN) ref Employees, Fk (model#) ref Model)

SSN will identify if someone is a [redacted] technician as it is the primary key. [redacted] Therefore, only a technician is an expert on a Model as they are the expert.