Problem 1

On slide page 17 of Lec5 ER.pdf the first bullet point mentions that WorksIn4 does not allow an employee to work in a department for two or more periods. Show that is indeed the case.

Since a relationship set needs unique identification, it cannot allow employees to work for two or more periods. However, if it does, it will have duplicate ssn (primary key). Hence, WorksIn4 does not allow an employee to work in a department for two or more periods.

Problem 2

How many distinct tuples are in a relation instance with cardinality 35?

The relation instance should have 35 distinct tulles as cardinality is the number of tubules in a relation.

Problem 3

Answer each of the following questions briefly. The questions are based on the following relation schema:
Emp(eid: integer, ename: string, age: integer, salary: real) Works(eid: integer, did: integer, pct time: integer)
Dept(did: integer, dname: string, budget: real, managerid: integer)
1. Give an example of a foreign key constraint that involves the Dept relation. What are the options for enforcing this constraint when a user attempts to delete a Dept tuple?

In a Dept relation ‘did’ can be referred to in Works relation as a foreign key. Hence did can be an example of a foreign key constraint that involves department relation.

There are four options:
a)CASCADE : Delete tuples referring to it
b)Disallow the deletion of Dept viols if some works tuple refer to it.
c)For Works tuple referring to it, set the did field to did of some ‘default’ department
d)For every Works tuple that referring to it, set the did field to null

2. Write the SQL statements required to create the preceding relations, including appropriate versions of all primary and foreign key integrity constraints.

```
CREATE TABLE Emp (
                        eid INTEGER,
                        ename CHAR(10),
                        age INTEGER,
                        salary REAL,
                        PRIMARY KEY (eid)
);

CREATE TABLE Works (
                        eid INTEGER,
                        did INTEGER,
                        pcttine INTEGER,
                        PRIMARY KEY (eid,did),
                        FOREIGN KEY (did) REFERENCES Dept,
                        FOREIGN KEY (eid) REFERENCES Emp,ON DELETE CASCADE)

CREATE TABLE Dept (
                        did INTEGER,
                        budget REAL,
                        managerid INTEGER ,
                        PRIMARY KEY (did),
                        FOREIGN KEY (managerid) REFERENCES Emp, ON DELETE SET
                        NULL)
```

3. Define the Dept relation in SQL so that every department is guaranteed to have a manager.

```
CREATETABLE Dept (
                        did INTEGER,
                        budget REAL,
                        managerid INTEGER NOT NULL,
                        PRIMARY KEY (did),
                        FOREIGN KEY(managerid) REFERENCES Emp)
```

4. Write an SQL statement to add Amy Kennedy as an employee with eid = 11, age = 22 and salary = 87, 000.

```
INSERTINTO Emp (eid, ename, age, salary) VALUES (11, 'Amy Kennedy', 22, 87000)
```

5. Write an SQL statement to give every employee a 5% raise.

UPDATE Emp ESET E.salary = E.salary *1.05

6. Write an SQL statement to delete the Bakery department. Given the referential integrity constraints you chose for this schema, explain what happens when this statement is executed.

DELETE FROM Dept D WHERE D.dname = Bakery.

Since the relationship is set to DELETE ON CASCADE, all tuples in the Works relationship that have the same ‘did’ will be removed.

Problem 4

Explain why the addition of NOT NULL constraints to the SQL definition of the Manages relation (in Section 3.5.3) would not enforce the constraint that each department must have a manager. What, if anything, is achieved by requiring that the ssn field of Manages be non-null?

NOT NULL only shows that there should be a manager for every entry in Manages. But if it doesn’t guarantee that there will be an entry to every department in the relation.

Problem 5

Consider the scenario from Problem 1 that you did in the previous problem set, where you designed an ER diagram for a university database. Write SQL statements to create the corresponding relations and capture as many of the constraints as possible. If you cannot capture some constraints, explain why. Please use my sample solutions for Problem 1 as the starting point as you solve this problem. It will make grading much easier for us. I will make my sample solution set available late tomorrow, Sunday.

1,

```
CREATE TABLE Teaches (
                        ssn CHAR(10),
                        courseId INTEGER,
                        semester CHAR(10),
                        PRIMARY KEY(ssn, coursed, semid),
                        FOREIGN KEY (ssn) REFERENCES Professor,
                        FOREIGN KEY (courseid) REFERENCE course)
                        FOREIGN KEY(semester) REFERENCES Semester)

CREATE TABLE Course (
                    courseId INTEGER,
                    PRIMARY KEY (courseid))

CREATE TABLE Professor (
```

```
                ssn INTEGER,
                PRIMARY KEY(ssn))

CREATE TABLE Semester(
                semid INTEGER,
                PRIMARY KEY (semid))
```

2,

```
CREATE TABLE Teaches (
                ssn CHAR(10),
                courseId INTEGER,
                semester CHAR(10).
                PRIMARY KEY (ssn, coursed),
                FOREIGN KEY (ssn) REFERENCES Professor,
                FOREIGN KEY(coursed) REFERENCES Course)

CREATE TABLE Course ( courseId INTEGER, PRIMARY KEY (courseid))
CREATE TABLE Professor (ssn INTEGER, PRIMARY KEY(ssn))
```

3, Same table as the one created in 2. However,we can't capture participation constraints because we can't check whether there's an entry in Professor whenever there's an entry in teaches.

4, We can combine the teaches and professors table because there's a 1:N participation in the professor part.

```
CREATE TABLE ProfessorTeaches (
                ssn CHAR (10).
                courseId INTEGER,
                semester CHAR (10),
                PRIMARY KEY (ssn),
                FOREIGN KEY (courseId) REFERENCES Course)

CREATE TABLE Course (
                courseId INTEGER,
                PRIMARY KEY (courseId))
```

5, We can combine professor and teaches to represent that one professor teaches one course and since the course id is also included in this table we don't need a separate table for Course to ensure the constraint that a professor must teach a course.

```
CREATE TABLE ProfessorTeaches (
                    ssn CHAR (10),
                    cid INTEGER,
                    semester CHAR (10),
                    PRIMARY KEY (ssn),
                    FOREIGN KEY (cid) REFERENCES Course)
```

```
6, CREATE TABLE Teaches (
                    gid INTEGER.
                    cid INTEGER.
                    semester CHAR(10).
                    PRIMARY KEY (gid, cid),
                    FOREIGN KEY(gid) REFERENCES Group,
                    FOREIGN KEY(cid) REFERENCES Course
```

```
CREATE TABLE MemberOf (
                ssn CHAR(10),
                gid INTEGER,
                PRIMARY KEY (ssn, gid),
                FOREIGN KEY (ssn) REFERENCES Professor,
                FOREIGN KEY (gid) REFERENCES Group)

CREATE TABLE Course (
                cid INTEGER,
                PRIMARY KEY (cid) )

CREATE TABLE Group (
                gid INTEGER,
                PRIMARY KEY (gid))

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

Problem 6

Consider the scenario from Problem 2 that you did in the previous problem set, where you designed an ER diagram for a company database. Write SQL statements to create the corresponding relations and capture as many of the constraints as possible. If you cannot capture some constraints, explain why. Please use my sample solution for Problem 2 as the starting point as you solve this problem. It will make grading much easier for us.

```
CREATE TABLE Employees (
                        ssn CHAR(10),
                        salary INTEGER,
                        phone CHAR(20),
                        PRIMARY KEY (ssn))

CREATE TABLE Departments (
                        dno INTEGER,
                        budget INTEGER,
                        dname CHAR(20),
                        PRIMARY KEY (dno))

CREATE TABLE WorksIn(
                    dno INTEGER,
                    dname CHAR(20),
                    PRIMARY KEY(ssn, dno),
                    FOREIGN KEY(ssn) REFERENCES Employees,
                    FOREIGN KEY (dno) REFERENCES Departments)

CREATE TABLE Manages(
                    ssn CHAR(10),
                    dno INTEGER,
                    PRIMARY KEY (dno),
                    FOREIGN KEY(ssn) REFERENCES Employees,
                    FOREIGN KEY (dno) REFERENCES Departments)

CREATE TABLE Dependents (
                    ssn CHAR(10),
                    name CHAR(10),
                    age INTEGER,
                    PRIMARY KEY (ssn, name),
                    FOREIGN KEY(ssn) REFERENCES Employees,
                    ON DELETE CASCADE)
```