Part 1  Standard exercises

1. ER-diagram

Convert Employee entity set to relation

```sql
CREATE TABLE Employee ( ssn CHAR (10),
    name CHAR (20),
    lot INTEGER,
    PRIMARY KEY (ssn))
```

Convert Department entity set and Manage relationship to relation

```sql
CREATE TABLE Dept_Mgr ( did INTEGER,
    dname CHAR (20),
    budget INTEGER,
    ssn CHAR (10) NOT NULL,
    PRIMARY KEY (did),
    FOREIGN KEY (ssn) REFERENCES Employee )
```

Convert works-in relationship to relation

```sql
CREATE TABLE works_in ( did INTEGER,
    ssn CHAR (10),
    since INTEGER,
    PRIMARY KEY (did, ssn))
```
The participation constraints on \textit{works-in} cannot be captured in the definition of \textit{works-in} relation schema. It requires using \texttt{ASSERTIONS}.

2. We may introduce Manager entity set as a subclass of Employee.

\begin{center}
\begin{tikzpicture}
    \node [entity] (Manager) {Manager};
    \node [entity] (Employee) at (Manager.south) {Employee};
    \node [entity] (Department) at (Manager.east) {Department};
    \draw (Manager) -- (Employee); % ISA
    \draw (Manager) -- (Department) node [midway, above] {manages};
\end{tikzpicture}
\end{center}

Every manager entity must participate in manages relationship.
\implies total participation.
We can assume that each manages at most 1 department
\implies key constraint.

With the above E-R diagram, we will have the following changes

define Manager relation:

```
CREATE TABLE Manager ( ssn CHAR(10),
    PRIMARY KEY (ssn),
    FOREIGN KEY (ssn)
    REFERENCES Employee )
```

change in Dept-Mgr
CREATE TABLE Dept_Mgr ( did INTEGER, 
dname CHAR(20), 
budget INTEGER, 
SSn CHAR(10) NOT NULL, 
Sinc INTEGER, 
PRIMARY KEY (did), 
UNIQUE (SSn), 
FOREIGN KEY (SSn) REFERENCES Employee )

Total participation of manager need to be expressed by 
 ASSERTION.

There is no change to Employee relation and 
 works-in relation.

3. within the Employee entity-set relation
   SSn \rightarrow (name, lot)

within the Department entity-set
   did \rightarrow

within the Dept_Mgr relation
   did \rightarrow (dname, budget, Manages_Since, SSn)

within works-in relation
   (did, SSn) \rightarrow works-in. Since

4. 

---

Diagram:

- A
- B
- C
- D

A connects to B and D
B connects to C and D
C connects to A
D connects to A and C
4.1. List all nontrivial FDs

\[ AB \rightarrow C \]
\[ BC \rightarrow D \]
\[ CD \rightarrow A \]
\[ AD \rightarrow B \]

We have \( AB \rightarrow C \) and \( AB \rightarrow B \) then \( AB \rightarrow BC \)

But \( BC \rightarrow D \). Thus \( AB \rightarrow C, AB \rightarrow D \). Due to reflexivity,
\( AB \rightarrow A, AB \rightarrow B \). Therefore \( AB \rightarrow ABCD \) (\( AB \) is a superkey).

However, with the given FDs, we cannot imply a new FD that the left hand side is \( A \) or \( B \). In other words, \( A \) and \( B \) are not the key of the schema \( \rightarrow \) \( AB \) is a key.

Due to its symmetry, we can easily show that \( BC, CD, AD \) also are keys of the schema \( ABCD \).

The list of nontrivial FDs will be:

\[ AB \rightarrow C \quad AB \rightarrow D \]
\[ AB \rightarrow AC \quad AB \rightarrow AD \quad AB \rightarrow BC \quad AB \rightarrow BD \quad AB \rightarrow CD \]
\[ AB \rightarrow ABC \quad AB \rightarrow ABD \quad AB \rightarrow BCD \]
\[ AB \rightarrow ABCD \]

The list of nontrivial FDs whose left hand side is \( BC, CD, AD \) can be obtained in the same way:

\[ ABC \rightarrow A D \quad ABC \rightarrow B D \quad ABC \rightarrow C D \]
\[ ABC \rightarrow ABD \quad ABC \rightarrow ACD \quad ABC \rightarrow BCD \]
\[ ABC \rightarrow ABCD \]

For nontrivial FDs whose left hand side is \( AB, CD, ACD, BCD \) can be obtained in the same way.

4.2. All the keys of \( R \) are:

\[ AB, BC, CD, DA \]
4.3. All superkeys of $R$ that are not keys:
$ABC$, $ABD$, $ACD$, $BCD$, and $ABCD$.

5. Let denote
$A_1 A_2 \ldots A_m$ as $A$
$B_1 B_2 \ldots B_n$ as $B$
$C_1 C_2 \ldots C_j$ as $C$
$D_1 D_2 \ldots D_k$ as $D$

we have $A \rightarrow B$ and $C \rightarrow D$.

$A \rightarrow B$ then $AC \rightarrow BC$ (augmentation).

$C \rightarrow D$ then $BC \rightarrow BD$ (augmentation).

$\Rightarrow AC \rightarrow BC$ and $BC \rightarrow BD$

$\Rightarrow AC \rightarrow BD$ (transitivity). Q.E.O.

6. 

6.1. All the FDs are nontrivial
all the FDs have the left-hand side not a superkey
$\Rightarrow$ all the FDs violate BCNF.

6.2. 

This decomposition is lossless join because the common attributes between $ABDE$ and $ABC$ is $AB$ which is the key of $ABC$ (because $AB \rightarrow C$).

This decomposition is not dependency preserving.
6.3. Because $ABE$ is the key then $AB \rightarrow C$, $DE \rightarrow C$, and $B \rightarrow D$ all violates 3NF.

6.4. This decomposition is the same as the decomposition in 6.2, and it is lossless join but not dependency preserving.

Part 2 Design exercise.

In producing a relational schema supporting a database of course and student registration information, we will undergo through following steps:

- Conceptual schema design of the database to build ER diagram of the database
- Relational schema design to identify relations in the database as well as keys and other constraints on each relation:
- Normalization relational schema obtained from Step 2.

Step 1: Conceptual Schema design

1.1. We need to identify which entity entities will be included in the database and relationship between entities

1.2. In real-world scenario, we have to meet will customers to obtain information for this step

1.3. In our example, we need the following entity sets: Students, Course, Instructor, Textbook
Attributes for each entity set will be:

- **Students**
  - Sid
  - Sname

- **Instructors**
  - iid
  - iName

- **Courses**
  - CRN
  - Sem
  - title
  - location
  - time

  \[ CRN = \text{course number} \]
  \[ Sem = \text{semester} \]

Instructors teach courses.
A course must be taught by at least 1 instructor (total participation).
Furthermore, we assume a course is taught by at most 1 instructor (key constraint).

attributes of entity sets are not drawn to make the E-R diagram simple.

A course will use textbooks.

Students take courses. Every student takes at least 1 course (total participation).
Step 2: Relational Schema design

2.1. We obtain the information from the E-R diagram above.

2.2. In our database, we have the following relations (and their associated constraints):

Instructors (iid: string, iName: string)

Students (sid: string, sName: string)

Textbooks (thId: string, title: string, author: string, year: integer)

Courses (CRN: string, Sem: string, title: string, location: string, time: string)

Takes (CRN: string, Sem: string, iid: string)

Foreign Key: (CRN, Sem) referencing to Courses, iid referencing to Instructors

Uses (CRN: string, Sem: string, thId: string)

Foreign Key: (CRN, Sem) referencing to Courses, thId referencing to Textbooks
Takes (Stid: String, CRN: String, sem: String, grade: real)

Foreign keys: Stid referencing to Students (CRN, sem) referencing to Courses

Total participation constraint: Students.

Step 3: Normalization

3.1. The information needed for normalization is the relational schema from Step 2 and the functional dependencies.

3.2. We observe that in the relations described in Step 2, all primary key constraints are FDs that satisfy BCNF and 3NF. Besides primary key constraints, we may have the following functional dependencies which are also candidate key constraints:

In Textbooks: (title, year) can be a candidate key.

In Courses: (title, sem) can be a candidate key.

Therefore, the above relational schema design is already in BCNF and 3NF. No further normalization is needed.