Minh Ngo
SJSU ID 010980398
CS157A HW2

Q1

R(A,B,C,D) with FDs: AB-> C, C -> D, D-> A

a.
Step 1: we need to project the set of functional dependencies, which can be inferred from R

Consider:
{A}+ =A
{B}+ =B
{C}+ ={C,D,A} => C->A

{A,B}+ = {C,D,A,B} => AB->D
{A,C}+ ={D,A} => AC->D
{A,D}+ ={A,D}
{B,C}+ ={B,C,D,A} => BC->A, BC->D
{B,D}+ ={A,B,C,D} => BD->A, BD->C
{C,D}+ ={A,C,D} => CD->A

{A,B,C}+ ={A,B,C,D} => ABC->D
{B,C,D}+ ={A,B,C,D} => BCD->A
{C,D,A}+ ={A,C,D}
{A,B,D}+ ={A,B,C,D}=> ABD->C

Step 2, from the set of FDs, we determine the keys are AB BC and BD

Step 3, BCNF violations are
C->A
C->D
D->A
AC->D
CD->A

b. Now, we decompose the relation into BCNF compliances.
Pick C->A

We decompose the table into
R1= R(C, D, A)
R2= R(B, C)

As what we can see, BC is BCNF compliant. For CDA, we will evaluate the relation for BCNF.
We go through the same process of FD projection as above. We conclude that C is the key for
R(A, C, D). So, the FD: D-> A violates BCNF. So R(A, C, D) violates BCNF

Thus, we continue to break down R1 into AD and CD

In conclusion, we break R(A,B,C,D) into BC, AD, CD.

R(A, B, C, D) with B-> C and B-> D

a. We process through the same process as what we did above
First, try to project the set of FDs:
The resulting FDs:
B->D
B->C
AB->C
AB->D
BD->C
BC->D
BCA->D
BDA->C

Next, we determine the key: AB
BCNF violations:
B->D
B->C
BC->D
BD->C

b. Decompose the table
We choose B->C to begin the decomposition.
{B}+= {B, C, D}.
We break down the table R(A, B, C, D) into R(B, C, D) and R(A, B)

R(A, B) satisfies BCNF. For the table R(B, C, D), we see that B is the only key. The projected FDs all contain B on the left hand side→ no BCNF violations.

The final decompositions: R(A, B) and R(B, C, D)

R(A, B , C, D) with AB->C, CB-> D, CD-> A, DA-> B

a.
Step 1: we need to project the set of functional dependencies, which can be inferred from R

Consider:
{A}+ =A
{B}+ =B
{C}+ =C
{D}+ =D

{A,B}+ = {C,D,A,B} => AB->D
{A,C}+ ={A, C}
{A,D}+ ={A, D, B, C}=> AD->C
{B,C}+ ={B,C,D,A} => BC->A
{B,D}+ ={B, D}
{C,D}+ ={A,C,D, B} => CD->B

{A,B,C}+ ={A,B,C,D} => ABC->D
{B,C,D}+ ={A,B,C,D} => BCD->A
{A,B,D}+ ={A,B,C,D}=> ABD->C

Infer the key: AB, AD, BC, CD
b.
It is obvious that all FDs identified above contains the key on LHS→ no BCNF violations
No need to decompose

R(A, B, C, D) with A->B, B-> C, C-> D, D-> A
a.
We need to project the set of functional dependencies, which can be inferred from R
The set of new FDs inferred from the above:
A->B
B->C
C->D
D->A
A->D, A->C
B->A, B->D
C->A, C->B
D->C, D->B
AB->C, AB->D
CA->B, CA->D
DA->B, DA->C

BC->A, BC-> D
DB-> A, DB->C
DC->A, DC->B
ABC->D
ABD->C
BCD->A
ADC->B
We can infer that A, B, C, D are the keys.
It is clear that all of the above FDs contain keys on the left hand side. Thus, there is no BCNF violations.
b.
So, we do not need to decompose this.

R(A, B, C, D, E) with AB->C, DE->C, B->D.
a.
We go through the same process as above
We obtain the list of FDs:
AB->C
DE->C
B->D
BA->D
BC->D
EB->C, EB->D
ABC->D
ABD->C
AEB->C, AEB->D
AED->C
BEC->D
DBE->C
ABCE->D, ABDE->C

From the list of FDs above, we infer the key as: ABE
BCNF violations are:
AB->C
DE->C
B->D
BA->D
BC->D
EB->C, EB->D
ABC->D
ABD->C
AED->C
BEC->D

DBE->C

b.
We choose AB-> C to decompose the table.
{AB}+=ABCD
So, we initially decompose ABCDE into ABCD and ABE.
ABE is already in BCNF

We evaluate ABCD. The key is AB. We see that B->C violates BCNF (not contain the key).
So we decompose ABCD into ABC and BD.
BD is in BCBF. ABC is also in BCNF (AB is the only key and all FDs satisfies BCNF).

Thus, we conclude that the decompositions are:
ABE, ABC and BD.

R(A, B, C, D, E) with AB-> C, C->D, D->B, D->E
a.
Project the FDs:
C->B, C->D, C->E
D->B, D->E
BA->C, BA->D, BA->E
CA->B, CA->D, CA->E
AD->B, AD->C, AD->E
CB->D, CB->E
BD->E
CD->B, CD->E
CE->B, CE->D
DE->B
ACB->E
ABD->C, ABD->E
ABE->C, ABE->D
ACD->B, ACD->E
ACE->B, ACE->D
ADE->C, ADE->B
BCD->E, BCD->D
CED->B
ABCD->E
ACBE->D

We infer the keys as: AB, AC and DA
BCNF violations as the below:
C->B, C->D
D->E, D->B

BC->D, BC->E
DB->E
DC->B, DC->E
CE->B, CE->D
DE->B
BCD->E
BCE->D
CDE->B
b.
We choose D->B to decompose
{D}+= {D, B, E}
Thus, the newly decomposed tables: BDE and ABC

BDE is in BCNF (D, ED and DB are the keys; all FDs contain the key on the left hand side)
ABC is not in BCNF (AB & AC are the keys; however C->B does not contain the key on the left hand side)

Decompose ABC into BC and AC
Both are BCNF compliant

So, we have: BDE, AC and BC

Q2
a.

A	B	C	D	E
a	b	c	d1	e1
a1	b	c	d	e1
a	b1	c	d1	e

Applying the functional dependencies (B->E, CE->A), we now have the new table (green is what has changed)

A	B	C	D	E
a	b	c	d1	e1
a	b	c	d	e1
a	b1	c	d1	e

It is clear that this is a lossy join. It is not a lossless decomposition.

Example:

A	B	C
a	b	c
a	b_1	c

B	C	D
b	c	d_1
b	c	d
b_1	c	d_1

A	C	E
a	c	e_1
a	c	e

After we join them, this is the result.

A	B	C	D	E
a	b	c	d_1	e_1
a	b	c	d	e_1
a	b_1	c	d_1	e_1
a	b	c	d_1	e
a	b	c	d	e
a	b_1	c	d_1	e

There are 3 more tuples than the original table → It is a lossy join !

b. AC ->E and BC ->D

A	B	C	D	E
a	b	c	d1	e1
a1	b	c	d	e1
a	b1	c	d1	e

Applying the functional dependencies, we now have the new table (green is what has changed)

A	B	C	D	E
a	b	c	d	e
a1	b	c	D	e1
a	b1	c	d1	e

It is clear that this is a lossless join.

c.

A	B	C	D	E
a	b	c	d1	e1
a1	b	c	d	e1
a	b1	c	d1	e

Applying the functional dependencies, we now have the new table (green is what has changed)

A	B	C	D	E
a	b	c	d	e
a1	b	c	d	e
a	b1	c	d	e

The decomposition is lossless.

d.

A	B	C	D	E
a	b	c	d1	e1
a1	b	c	d	e1
a	b1	c	d1	e

Applying the functional dependencies, we now have the new table (green is what has changed)

A	B	C	D	E
a	b	c	d	e
a1	b	c	d	E
a	b1	c	d	e

This is lossless join.

Q3.

a.

```
SELECT  address
FROM    Studio
WHERE   name = 'MGM';
```

b.

```
SELECT  birthdate
FROM    MovieStar
WHERE   name = 'Sandra Bullock';
```

c.

```
SELECT  starName
FROM    StarsIn
```

```sql
WHERE   movieYear = 1980
    OR movieTitle LIKE '%Love%';
```

Q4

a.

```sql
SELECT  MovieStar.name
FROM    MovieStar ,StarsIn
WHERE   MovieStar.name = StarsIn.starName
   AND StarsIn.movieTitle = 'Titanic'
   AND MovieStar.gender     = 'M';
```

b.

```sql
SELECT  StarsIn.starName
FROM    Movies ,StarsIn, Studios
WHERE   Studios.name ='MGM'
   AND Movies.year = 1995
   AND Movies.title = StarsIn.movieTitle
   AND Movies.studioName = Studios.name;
```

c.

```sql
 SELECT  MovieExec.name
FROM    MovieExec, Studios
WHERE   MovieExec.cert# = Studios.presC#
   AND Studios.name  = 'MGM';
```

d.

```sql
SELECT  M1.title
FROM    Movies M1,
       Movies M2
WHERE   M1.length > M2.length
   AND M2.title  ='Gone With the Wind' ;
```