Jacob Snyder

CS121 Set 6

1.

a. No non-trivial functional dependencies. This is because it's a many to many mapping so nothing is strictly dependent on something else.
b. We have $b \rightarrow a$ because the relationship is one to many, so for every b there exists a single a.
c. We have $a \rightarrow b$ because the relationship is one to many, so for every a there exists a single b.
d. We have $b \rightarrow a$ and $a \rightarrow b$ because the relationship is one to one, so for every b there exists a single a and for every a there exists a single b.

2.

a. Union rule: $a \rightarrow b$ and $a \rightarrow y$ implies $a \rightarrow by$. Proof using Armstrong's axioms:
$ay \rightarrow by$ and $aa \rightarrow ay$ by the Augmentation rule
$aa \rightarrow by$ by the Transitivity rule
$aa = a$ because they are the same set of attributes

b. Decomposition rule: $a \rightarrow by$ implies $a \rightarrow b$ and $a \rightarrow y$. Proof using Armstrong's axioms:
$by \rightarrow b$ and $by \rightarrow y$ by the Reflexivity rule
$a \rightarrow by$ and $by \rightarrow b$ implies that $a \rightarrow b$ by the Transitivity rule
$a \rightarrow by$ and $by \rightarrow y$ implies that $a \rightarrow y$ by the Transitivity rule

c. Pseudotransitivity rule: $a \rightarrow b$ and $yb \rightarrow d$ implies $ay \rightarrow d$. Proof using Armstrong's axioms:
$ay \rightarrow by$ by the Augmentation rule
$ay \rightarrow d$ by the Transitivity rule

3.

a. Superkeys s of R will be of the form $s \rightarrow ABCDE$. From the given functional dependencies we can derive such superkeys:
$A \rightarrow BC$ implies $A \rightarrow B$ and $A \rightarrow C$ by Decomposition rule
$A \rightarrow B$ and $B \rightarrow E$ implies $A \rightarrow E$ by Transitivity rule
$A \rightarrow B$ and $B \rightarrow D$ implies $A \rightarrow D$ by Transitivity rule
$A \rightarrow A$ (reflexivity rule), $A \rightarrow BC$, $A \rightarrow D$, $A \rightarrow E$ implies $A \rightarrow ABCDE$ by Union rule
$E \rightarrow A$ and $A \rightarrow ABCDE$ implies $E \rightarrow ABCDE$ by Transitivity rule
$CD \rightarrow E$ and $E \rightarrow ABCDE$ implies $CD \rightarrow ABCDE$ by Transitivity rule
$B \rightarrow D$ implies $BC \rightarrow CD$ by Augmentation rule
$BC \rightarrow CD$ and $CD \rightarrow ABCDE$ implies $BC \rightarrow ABCDE$ by Transitivity rule
Therefore our candidate keys are A, E, CD, and BC.

b. First we include the trivial dependencies as follows:
$\alpha \rightarrow \beta$ where $\alpha = \{ABCDE\}$ and $\beta \subseteq \alpha$.
Now we can generate the rest of the functional dependencies by considering the candidate keys derived in part (a). Any functional dependency that has a candidate key on the left hand side and a subset of R on the right hand side is valid. We can also include other attributes in R in addition to the candidate key on the left hand side because the presence of the candidate key will ensure that the functional dependency is valid.
Lastly, we must consider the functional dependency $B \rightarrow D$ and its non-trivial derivatives, since B is not a candidate key and is therefore not covered by the above paragraph. It turns out the only non-trivial derivative is $B \rightarrow BD$ which can be derived by the Augmentation rule.
This covers all functional dependencies that appear in F^+.

4. No, $A \twoheadrightarrow BC$ does not imply $A \twoheadrightarrow B$ and $A \twoheadrightarrow C$. Counterexample:

	A	B	C	D
t_1	1	1	1	1
t_2	1	1	0	0
t_3	1	1	1	0
t_4	1	1	0	1

Note that $A \twoheadrightarrow BC$, by the definition given in the lecture slides. However, it is not true that $A \twoheadrightarrow B$ so this is not true.

5.

a. We start with $F = \{A \to E, BC \to D, C \to A, AB \to D, D \to G, BC \to E, D \to E, BC \to A\}$
We can first remove $BC \to E$ because it is extraneous ($BC \to A$ and $A \to E$ by Transitivity make $BC \to E$):
$$\{A \to E, BC \to D, C \to A, AB \to D, D \to G, D \to E, BC \to A\}$$
We can combine $D \to G$ and $D \to E$ with Union rule to $D \to EG$:
$$\{A \to E, BC \to D, C \to A, AB \to D, D \to EG, BC \to A\}$$
We can first remove $BC \to D$ because it is extraneous ($C \to A$ under Augmentation rule makes $BC \to AB$ and then using that with $AB \to D$ under union rule makes $BC \to D$):
$$\{A \to E, C \to A, AB \to D, D \to EG, BC \to A\}$$
We can remove the B from $BC \to A$ because it is extraneous: $C \to A$ already covers these dependencies.
Therefore we have $F_c = \{A \to E, C \to A, AB \to D, D \to EG\}$

b. We will compute BC^+ because it's commonly used in the definition of F:
$BC^+ = BC$
$BC^+ = ABCDE$ because $BC \to D$, $BC \to A$, $BC \to E$
$BC^+ = ABCDEG$ because $D \to G$
Therefore BC is a superkey.
Now we find B^+ and C^+:
$B^+ = B$
$C^+ = C$
$C^+ = ACE$ because $C \to A$ and $A \to C$
The attribute-set closure of B and C is not R so BC is a candidate key.

c. We start with $F = \{A \to E, BC \to D, C \to A, AB \to D, D \to G, BC \to E, D \to E, BC \to A\}$
$A \to E$ from F implies that (A, E) is in BCNF
We are left with $\{A, B, C, D, G\}$
We can use $D \to G$, to get that (D, G) is in BCNF
We are left with $\{A, B, C, D\}$
We can use $C \to A$, to get that (C, A) is in BCNF
We are left with $\{B, C, D\}$, which is in BCNF because BC is a candidate key.
Therefore our schema relations are as follows:
(A, E), (D, G), (C, A), (B, C, D)
Dependencies not preserved: $AB \to D$, $D \to E$

d. We start with $F = \{A \to E, BC \to D, C \to A, AB \to D, D \to G, BC \to E, D \to E, BC \to A\}$
$D \to EG$ from F using the Union rule implies that (D, E, G) is in BCNF
We are left with $\{A, B, C, D\}$
$C \to A$ implies that (C, A) is in BCNF
We are left with $\{B, C, D\}$
$BC \to D$ implies that (B, C, D) is in BCNF

Therefore our schema relations are as follows:
(D, EG), (C, A), (B, C, D)
Dependencies not preserved: $A \to E$, $AB \to D$

e. From $F_c = \{A \to E, C \to A, AB \to D, D \to EG\}$ we have:
$R_1(A, E)$ with A as a primary key because $A \to E$
$R_2(C, A)$ with C as a primary key because $C \to A$
$R_3(A, B, D)$ with AB as a primary key because $AB \to D$
$R_4(D, E, G)$ with D as a primary key because $D \to EG$
And then we add:
$R_5(B, C)$ because BC is a candidate key.

6.

a. For legibility we will call {course_id, section_id, dept, units, course_level, instructor_id, term, year, meet_time, room, num_students} as $\{A, B, C, D, E, G, H, I, J, K, L\}$
We have $HIJK \to HIJK$ (reflexivity) and $HIJK \to ABG$ therefore $HIJK \to ABGHIJK$ (union)
We also have $A \to CDE$ and $ABHI \to GJKL$ so by decomposition and union with $HIJK \to ABGHIJK$ we have $HIJK \to ABCDEGHIJKL$.
Therefore $KJHI$ is a superkey.
We have $ABHI \to ABHI$ (reflexivity) and $ABHI \to GJKL$ therefore $ABHI \to ABGHIJKL$ (Union)
We also have $A \to CDE$ so by decomposition and union with $ABHI \to ABGHIJKL$ we have $ABHI \to ABCDEGHIJKL$.
Therefore $ABHI$ is a superkey.
Therefore we have {course_id, section_id, term, year} and {room, meet_time, term, year} as superkeys.

b. We can see from { course_id, section_id, term, year } → { meet_time, room, num_students, instructor_id } and { room, meet_time, term, year } → { instructor_id, course_id, section_id } that instructor_id is extraneous. Therefore our two options for F_c are to either have instructor_id in the second or third relation:

F_{c1} = { course_id } → { dept, units, course_level }, { course_id, section_id, term, year } → { meet_time, room, num_students, instructor_id }, { room, meet_time, term, year } → { course_id, section_id }

F_{c2} = { course_id } → { dept, units, course_level }, { course_id, section_id, term, year } → { meet_time, room, num_students }, { room, meet_time, term, year } → { instructor_id, course_id, section_id }

I think it makes more sense to have instructor_id in the second dependency, because that's general info about the class. As opposed to the third dependency which is class ids.

c. From $F_c = \{A \to CDE, ABHI \to GJKL, HIJK \to AB\}$ we can perform BCNF decomposition and we can get a normal form. BCNF:
$A \to CDE$ gives us $R_1(ACDE)$ with A as a primary key
$ABHI \to GJKL$ gives us $R_2(A, B, G, H, I, J, K, L)$ with both $ABHI$ and $HIJK$ as candidate keys.
Normal form (using F_c and adding candidate keys):
We have $R_1(A, C, D, E)$ with A as primary key and $R_2(A, B, G, H, I, J, K, L)$ with $ABHI$ and $HIJK$ as candidate keys.
We notice that these yield equivalent schemas. Therefore we make our decision based on scale/practicality vs correct and complete representation. Because this is a relatively small

database (there isn't a ton of data in a school) we can focus on a correct and complete representation in the database. Therefore we should use the 3NF solution.

7. For legibility, we will call (email_id, send_date, from_addr, to_addr, subject, email_body, attachment_name, attachment_body) as (A, B, C, D, E, G, H, I)
   We start with $\{A, B, C, D, E, G, H, I\}$
   From $A \to BCEG$ we get (A, B, C, E, G) with A as a primary key
   We are left with $\{A, D, H, I\}$
   From $A \twoheadrightarrow D$ we get 2 relations:
   (A, D) with AD as the primary key
   (A, H, I) with AH as the primary key

   Therefore our relation schemas are as follows:

   (email_id, send_date, from_addr, subject, email_body) which is in 4NF because email_id is a superkey (according to the given $A \to BCEG$).

   (email_id, to_addr) which is in 4NF because both email_id and to_addr are primary keys, corresponding to a trivial dependency (according to the given $A \twoheadrightarrow D$)

   (email_id, attachment_name, attachment_body) which is in 4NF because email_id is a superkey (according to the given $AH \to I$).