1. Define the following concepts: functional dependency, closure of an attribute, canonical cover, prime attribute, dependency preservation, common lookup table, crosstab, 2NF, 3NF, BCNF

- Functional dependency: A functional dependency is a statement about allowable interactions in a database. A set of attributes (the antecedent) determines a set of other attributes (the consequent). A functional dependency is expressed as antecedent -> consequent

- Closure of an attribute: A set of attributes that can be reached from the given attribute (can use multiple attributes as input)

- Canonical cover: a minimal set of functional dependencies equivalent to the original relation with no redundant FDs or parts of FDs

- Prime attribute: An attribute that is part of any of the candidate keys of a relation

- Dependency preservation: FDs are preserved through decomposition. If the relations are combined through joining, the FDs are still there (lossless). 3NF preserves, BCNF+ does not

- Common lookup table: tables that contain everything. You can lookup any object in the database, which results in low security and a redundant/complicated mess

- Crosstab: values for one attribute become column names or tables (do not do this, make a column, not a new name/table)

- 2NF: a relation this has the properties of being in 1NF (no multiple values in cells), as well as every non-prime attribute being dependent on the whole of every primary key

- 3NF: a relation that has the properties of being in 2NF (1NF and non-prime attributes dependent on whole candidate keys) and every non-prime attribute is non-transitively dependent on every key. Conditions for an FD (X->Y) to be in 3NF if trivial (Y is a subset of X), or X is a superkey of R, or each attribute in the set difference {Y-X} is a prime attribute. 3NF is violated when a non-key attribute depends on another non-key attribute

BCNF: a relation that has all redundancy based on functional dependency removed. Has all the same conditions except this form removes the option for each attribute in the set difference {Y-X} is a prime attribute. If an FD has this property, it is not in BCNF.

2. Consider the following set of functional dependencies on the relation schema

R = (A, B, C, D, E, F, G)

AB → CD

C → EF

G → A

G → F

CE → F

a) Compute BG+

Given {B, G}, we have {B, G}. From here we take G->A and G->F to reach {B, G, A, F}. From here we take AB -> CD to reach {B, G, A, F, C, D}. From here we take C->EF to get {B, G, A, F, C, D, E}, which is {BG}+

{BG}+ = {B, G, A, F, C, D, E}

b) Find the candidate keys for R

By looking at the consequent of each FD, we do not see B or G, so every CK must contain B and G. Since the closure of BG was found above to include every attribute in R, {B, G} is the candidate key of minimal size (2 elements)

CK = {B, G}

c) Remove extraneous attributes and compute the canonical cover of the dependencies

1. G -> A
2. AB -> C
3. AB -> D
4. C -> E
5. C -> F
6. G -> F
7. CE -> F

- remove extra attr.

2: AB -> C. Does {A}+ include C? No. Does {B}+ include C? No. No extraneous attributes in 2

3: AB -> D. Does {A}+ include D? No. Does {B}+ include D? No. No extraneous attributes in 3

7: CE -> F. Does {E}+ include F? No. Does {C}+ include F? Yes, so E is extraneous. 7 is simplified to C -> F which is another rule so we actually can remove FD 7.

-remove unnecessary rules

1. Is G -> A implied elsewhere? No.
2. Is AB -> C implied elsewhere? No.
3. Is AB -> D implied elsewhere? No.
4. Is C -> E implied elsewhere? No.
5. Is C -> F implied elsewhere? No. If FD 7 was still available, then it would have been, but we removed this rule in the removal of extra attributes.

- combine FDs based on antecedent:

1. G -> AF
2. AB -> CD
4. C -> EF

F_c = {G -> AF
AB -> CD
C -> EF}

d) Decompose the relation into a collection of schemas in 3NF

start with a list of FDs:
R = (A, B, C, D, E, F, G)
1. G -> AF
2. AB -> CD
3. C -> EF

CK = {B, G}

Split each FD into its relation
R1(G, A, F) with G -> AF
R2(A, B, C, D) with AB -> CD
R3(C, E, F) with C -> EF
R4(B, G) with no FDs
{B, G} is not present in any of the relations, so we need a new relation
We now have the 3NF Decomposition: {R1(G, A, F) with G -> AF
R2(A, B, C, D)
with AB -> CD
R3(C, E, F)
with C -> EF
R4(B, G)
with no FDs}

3. Consider the following set of functional dependencies on the relation schema
R = (A, B, C, D, E, F):

$A \rightarrow BCD$

$BC \rightarrow DE$

$AB \rightarrow CD$

$D \rightarrow A$

B → D

a) Compute D+

Given {D}, we take D -> A to reach {D, A}. From here we can take A -> BCD to get {D, A, B, C}. We now take BC -> DE to get {D, A, B, C, E}.

{D}+ = {D, A, B, C, E}

b) Find the candidate keys for R

Observing the consequents of the FDs, we see that there is no F present, so it must be in every CK. Testing the closure of different minimal size supersets of {F}, we reach the following candidate keys of minimal size (2 elements):

CKs = {{F, A}, {F, B}, {F, D}}

c) Remove extraneous attributes and compute the canonical cover of the dependencies

-split FDs by their consequent:
FD1: D -> A
FD2: A -> B
FD3: A -> C
FD4: AB -> C
FD5: A -> D
FD6: BC -> D
FD7: AB -> D
FD8: B -> D
FD9: BC -> E

- remove extra attr

FD4: Does {B}+ include C? Yes: B -> D(FD8) -> A(FD1) -> C(FD3), so {A} is extra in FD4. Does {A}+ include C? Yes: A -> C (FD3). This makes FD4 unnecessary, so remove it.

FD6: Does {C}+ include D? No. Does {B}+ include D? Yes: B -> D(FD8), so C is extra in FD6

FD7: Does {B}+ include D? Yes: B -> D(FD8), so A is extra. Does {A}+ include D? Yes: A -> D(FD5), so B is extra. remove FD7

FD9: Does {C}+ include E? No. Does {B}+ include E? Yes: B -> D(FD8) -> A(FD1) -> C(FD3) -> E(FD9)

-remove unnecessary rules:
FD1: D -> A
FD2: A -> B
FD3: A -> C
FD4: A -> D

FD5: B -> D
FD6: B -> D
FD7: B -> E

1: is D -> A implied elsewhere? No
2: is A -> B implied elsewhere? No
3: is A -> C implied elsewhere? No
4: is A -> D implied elsewhere? Yes: A -> B(FD2) -> D(FD6), remove FD4
5: is B -> D implied elsewhere? Yes: B -> D(FD6), remove FD5
6: is B -> D implied elsewhere? No
7: Is B -> E implied elsewhere? No

resulting rules:
FD1: D -> A
FD2: A -> B
FD3: A -> C
FD4: B -> D
FD5: B -> E

-combine rules based on antecedent:
1: A -> BC
2: B -> DE
3: D -> A

F_c = {A -> BC
B -> DE
D -> A}

d) Decompose the relation into a collection of schemas in BCNF

R = (A, B, C, D, E, F)
CKs = {{F, A}, {F, B}, {F, D}}

F_c = {A -> BC	not SK, violation
B -> DE	not SK, violation
D -> A	not SK, violation}

create new relations:
R1(A, B, C) with A -> BC (not violation)
R2(A, D, E, F) with B -> DE (violation), D -> A (violation)
R3(B,D,E) with B -> DE (not violation)
R4(D,A) with D -> A (not violation)
R5(F,A) with no FDs

resulting BCNF:
R1(A, B, C) with A -> BC (not violation)
R3(B,D,E) with B -> DE (not violation)
R4(D,A) with D -> A (not violation)
R5(F,A) with no FDs

4. Consider the following set of functional dependencies on the relation schema
R = (A, B, C, D, E, F, G):

A → BCDF

BD → CF

B → D

E → A

DF →→ G

a) Compute A+

Given A, we start with A -> to get {A, B, C, D, F}. We cannot take any other FDs, so closure is done

{A}+ = {A, B, C, D, F}

b) Find all candidate keys for R

Since we cannot use the antecedent to get G (multivalued), then every CK must have G. A quick check of the other keys shows that we can get a minimal size CK (2 elements) with {E, G}

c) Remove extraneous attributes and compute the canonical cover of the dependencies

-split dependencies by consequent
FD1: A -> B
FD2: A -> C
FD3: A -> D
FD4: A -> F
FD5: BD -> C
FD6: BD -> F
FD7: B -> D
FD8: E -> A
(not using MVD in CC)

-remove extra attr

FD5: Does {D}+ include C? No. Does {B}+ include C? Yes: B -> D(FD7) -> C(FD5), so D is extra

FD6: Does {D}+ include F? No. Does {B}+ include F? Yes: B -> D(FD7) -> F(FD6), so D is extra

-remove unnecessary rules

FD1: A -> B
FD2: A -> C
FD3: A -> D
FD4: A -> F
FD5: B -> C
FD6: B -> F
FD7: B -> D
FD8: E -> A

1: Is A -> B implied anywhere? No
2: Is A -> C implied anywhere? Yes: A -> B(FD1) -> C(FD5), remove FD2
3: Is A -> D implied anywhere? Yes: A -> B(FD1) -> D(FD7), remove FD3
4: Is A -> F implied anywhere? Yes: A -> B(FD1) -> F(FD6), remove FD4
5: Is B -> C implied anywhere? No
6: Is B -> F implied anywhere? No
7: Is B -> D implied anywhere? No
8: Is E -> A implied anywhere? No

recombine rules to get CC:

F_c = {A -> B
B -> CDF
E -> A}

d) Decompose the relation into a collection of schemas in 4NF

R(A, B, C, D, E, F, G)
CK = {E, G}
F = {A -> B violation of BCNF (A not SK)
B -> CDF violation of BCNF (B not SK)
E -> A violation of BCNF (E not SK)
DF →→ G} violation of 4NF (DF not SK)

- R1(B,C,D,F) with B -> CDF
- R2(A,B,E,G) (violation)
  - R3(A,B) with A -> B
  - R4(A,E,G) (violation)
    - R5(E,A) with E -> A
    - R6(E,G) with no FDs
- R7(D,F,G) with DF →→ G

4NF Decomposition:

- R1(B,C,D,F) with B -> CDF
- R3(A,B) with A -> B
- R5(E,A) with E -> A
- R6(E,G) with no FDs
- R7(D,F,G) with DF $\rightarrow\rightarrow$ G