Consider the schema S(A, B, C, D, E) together with the functional dependencies:

- BD →A
- AB→C
- D→A
- B→C
- C→E

1) Is S in 3NF? Why or why not?

   The closures are below:

   $\{BD\}^+ = \{ABCDE\}$

   $\{AB\}^+ = \{ABCE\}$

   $\{D\}^+ = \{AD\}$

   $\{B\}^+ = \{BCE\}$

   $\{C\}^+ = \{CE\}$

   Aside from BD→A, all other FDs have a LHS that is **not** a superkey.

   The minimal key of S is:

Left	Middle	Right
BD	AC	E

   The closure of BD (given above) includes all of the attributes of S. Therefore, BD is a minimal key.

   In order for a relation to be in 3NF, every FD has to have a LHS that is a superkey and/or a RHS that is part of the minimal key. None of the FDs have a RHS that is part of a minimal key.

   This means that all FDs aside from BD→A violate 3NF. Therefore, S is not in 3NF.

2) If S is not in 3NF, decompose into 3NF using the **lossless-join decomposition** method.
First, you need to find the minimal cover. As a reminder, here are the steps for minimal cover.
   1. Put all FDs in standard form.
   2. Minimize the LHS of each FD.
   3. Delete redundant FDs.

   The minimal cover is:
   - D→A
   - B→C
   - C→E

   In this case, it is easy to see because each of the top two FDs is a copy of three of the bottom ones, but with an extra attribute on the LHS. This means that they are duplicates since the closures will be the same if you just remove the top two FDs (if you don't see why, try this with and without the top to FDs to see)

   Onto decomposition.

   (The answer may differ depending on which FD you decomposed on first.)

   First round of decomposition: D → A
   S1(A<u>D</u>), S2(BCDE)

   Second round of decomposition: B → C
   S1(A<u>D</u>), S3(<u>B</u>C), S4(BDE)

   Both S1 and S3 are in BCNF because they are two attribute relations, so they are guaranteed to be in BCNF. Next we check S4. If you work only from the FDs, then it will appear to be in BCNF. However, working from the closures, we see that B+ = {B,C,E}, so B → E holds in S4 but B is not a key for S4, because B does not determine D. Therefore, we must decompose again:

   S5(<u>B</u>,E), S6(<u>B</u>,<u>D</u>)

   All decomposed relations are two attribute relations, so we can stop decomposing.

   Before you are done, you need to check to see if you have lost any FDs (I.e., are there any FDs in the minimal cover for which both the LHS and the RHS do not appear together in a single relation). Since we have lost C → E, we need to add that back in as a relation to preserve all FDs.

You should also check to see whether there is one relation that is a strict subset of another. If there is, you should optimize your answer by removing the redundant relation.

Final answer:
S1(A<u>D</u>), S3(<u>B</u>C), S5(<u>B</u>,E), S6(<u>B,D</u>), S7(<u>C</u>E)

3) If S is not in 3NF, decompose into 3NF using the **synthesis** method.
In synthesis, create a relation for every FD in the minimal cover.
S-syn_1(AD), S-syn_2(BC), S-syn_3(CE)

[Note that we're using the prefix `S-syn_` to make it clear that this is a separate set of relations from the ones using the decomposition method. In reality, they'd just be named S1, S2, and S3, but we wanted to make it clear that they're separate from the ones above.]

Now, check to see if there is at least one relation containing a minimal key (this is to ensure that the decomposition is lossless). The minimal key of S is BD and there is no relation that contains BD. Add it.

You should also check to see whether there is one relation that is a strict subset of another. If there is, you should optimize your answer by removing the redundant relation.

Final answer:
S-syn_1(AD), S-syn_2(BC), S-syn_3(CE), S-syn_4(BD)

The two decomposition methods yield slightly different results. Decomposition can often vary depending on factors like the order in which you handle the FDs and which methods you use. This exercise is meant as a way for you to experience this yourself!