1. [25 pts]
(a) [5 pts] Start by looking at the existing Phone table. What non-trivial functional dependencies does the current table have?
There is no non-trivial FD in the current Phone table.

(b) [5 pts] What additional functional dependency is needed to express your boss' rule about phone type consistency?
number -> type

(c) [5 pts] Given this new functional dependency, what is the new primary key for the Phone table given its current schema?
(user_id, number)
Based on the new requirement, the same user cannot have two same numbers with different types.

(d) [5 pts] What normal form is the Phone table currently in? Briefly show your reasoning.
1NF. Now the table contains partial dependency from a non-prime attribute to a candidate key: candidate key:(user_id,number) , FD: number->type.

(e) [5 pts] Decompose Phone into multiple tables to produce a BCNF design if the current design isn't already there.
Phone(user_id, number, PRIMARY KEY (user_id, number) )
Phone_Type (number, type, PRIMARY KEY (number) )

2. [25 pts]
(a) [5pts] List all of the functional dependencies involving the attributes of the relation in question. (Just give the basic list, not the closure of that list. ☺)
(user_id, post_id, view_time) -> staying_time

(b) [5 pts] What are the candidate keys for this relation?
(user_id, post_id, view_time)

(c) [5 pts] Does the relation satisfy 1NF [Yes/No]? Show your reasoning.
Yes. all attributes are atomic.

(d) [5 pts] Does the relation satisfy 2NF [Yes/No]? Show your reasoning.
Yes. No partial dependency.

(e) [5 pts] Does the relation satisfy 3NF [Yes/No]? Show your reasoning.
Yes. No transitive dependency.
3. [25 pts]

R(A, B, C, D, E)

All attributes contain only atomic values.

FD1: A → BC
FD2: CD → E
FD3: B → D
FD4: A → E

(a) [5 pts] Compute A+, the attribute closure of the set of attributes {A}.

A+ = {ABCDE}

(b) [5 pts] List the candidate keys of R.

A

(c) [5 pts] What’s the highest normal form that R satisfies and why?

2NF. (It violates 3NF because of transitive dependency to a non-prime attribute, e.g., B→D)

(d) [5 pts] If R is not already at least in 3NF, then normalize R into 3NF and show the resulting relation(s) and their candidate keys. Your decomposition should be both join-lossless and dependency-preserving. If R is already in 3NF, just list the candidate keys of R.

R0(A, B, C), candidate key A
R1(B, D), candidate key B
R2(C, D, E), candidate key (C,D)

(e) [5 pts] Is your answer to (d) in BCNF? Why or why not?

Yes. All FDs in each relation are now key constraints.

4. [25 pts]

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

All attributes contain only atomic values.

FD1: AB → C
FD2: BC → E
FD3: B → D
FD4: F → G

(a) [5 pts] Compute A+, the attribute closure of the set of attributes {A}.
\( A^+ = \{ A \} \)

(b) [5 pts] List the candidate keys of \( R \).
\{ABF\}

(c) [5 pts] What’s the highest normal form that \( R \) satisfies and why?
1NF. It violates 2NF because of partial dependency (e.g., \( B \to D \))

(d) [5 pts] If \( R \) is not already at least in 3NF, then normalize \( R \) into 3NF and show the resulting relation(s) and their candidate keys. Your decomposition should be both join-lossless and dependency-preserving. If \( R \) is already in 3NF, just list the candidate keys of \( R \).

- \( R_0(B, D) \), candidate key \( B \)
- \( R_1(F, G) \) candidate key \( F \)
- \( R_2(A, B, C) \) candidate key \( A, B \)
- \( R_3(B, C, E) \) candidate key \( B, C \)
- \( R_4(A, B, F) \) candidate key \( A, B, F \)

(e) [5 pts] Is your answer to (d) in BCNF? Why or why not?
Yes. All FDs are now key constraints.