Database Normalization & Functional Dependencies — Detailed Solutions

Problem 1
Relation: R1(A, B, C, D, E, F)
Functional Dependencies: A → BC, D → E, BC → D, A → D

Goal: Find candidate key(s), prime & non-prime attributes, and highest normal form. Show step-by-step closures used to find keys.

Step-by-step closures
1) A+ = {A}. From A → BC add B, C → A+ = {A,B,C}. From A → D add D → A+ = {A,B,C,D}. From D → E add E → A+ = {A,B,C,D,E}. (F missing) => A is not a key.

2) AF+: start with {A,F}. Using A → BC gives {A,B,C,F}. A → D adds D => {A,B,C,D,F}. D → E adds E => {A,B,C,D,E,F}. All attributes obtained → **AF is a candidate key**.

3) BC+ = {B,C}. BC → D adds D => {B,C,D}. D → E adds E => {B,C,D,E}. No A or F → not a key.

Result:
Candidate Key(s): {AF}
Prime attributes: {A, F}
Non-prime attributes: {B, C, D, E}

Normal form check:
- 1NF: Yes (atomic values).
- 2NF: Candidate key AF is composite. Check partial dependencies: A → BC is a dependency where A (a part of key AF) determines non-prime attributes B and C → this is a partial dependency, so 2NF violated.
→ **Highest normal form = 1NF**.

Problem 2 — Multiple relations
For each subproblem we list FDs, compute closures to find candidate keys and list prime/non-prime attributes.

Q1. Relation R(ABCD) with FDs: AB → C, C → D, D → A

Closures:
- AB+ = {A,B}. From AB → C add C => {A,B,C}. From C → D add D => {A,B,C,D} → all attributes. So **AB** is a candidate key.
- Check smaller sets: A+ = {A} (no C/D without others), B+ = {B}, C+ = {C,D,A} (missing B) → not keys.

Result: Candidate key = {AB}. Prime attrs = {A,B}. Non-prime = {C,D}.

Q2. Relation R(ABCDE) with FDs: A → D, B → A, BC → D, AC → BE

Closures to find keys:
- AC+ = {A,C}. From AC → BE add B and E => {A,B,C,E}. From B → A already included. From A → D add D => {A,B,C,D,E} → all attributes. So **AC** is a candidate key.

- BC+ = {B,C}. From B → A add A => {A,B,C}. From AC → BE (A and C present) add B and E => E included => {A,B,C,E}. From A → D add D => {A,B,C,D,E} → all attributes. So **BC** is also a candidate key.

Check minimality: neither A nor C alone give all attributes, nor B alone. So keys are minimal.

Result: Candidate keys = {AC, BC}. Prime attributes = {A,B,C}. Non-prime = {D,E}.

Q3. Relation R(ABCDE) with FDs: B → A, A → C, BC → D, AC → BE

Closures:
- BC+ start: {B,C}. From B → A add A => {A,B,C}. From A → C nothing new. From AC → BE (A and C present) add B (already) and E => {A,B,C,E}. From BC → D add D => {A,B,C,D,E} → all attributes. So **BC** is a candidate key.

- AC+ = {A,C}. From AC → BE add B and E => {A,B,C,E}. From BC → D (B and C present) add D => all attributes. So **AC** is also a candidate key.

Result: Candidate keys = {BC, AC}. Prime attributes = {A,B,C}. Non-prime = {D,E}.

Q4. Relation R(ABCDEF) with FDs: A → BCD, BC → DE, B → D, D → A

We compute closures to find keys (note attribute F must be present in any key because no FD produces F):
- A+ = {A}. From A → BCD add B,C,D => {A,B,C,D}. From BC → DE (B and C present) add E => {A,B,C,D,E}. Missing F → A not key.

Test combinations with F:
- AF+ starts {A,F} → A → B,C,D -> {A,B,C,D,F}. BC → DE adds E -> all attributes → **AF** is a key.

- BF+ start {B,F}. From B → D add D => {B,D,F}. From D → A add A => {A,B,D,F}. From A → BCD add C => {A,B,C,D,F}. From BC → DE (B and C present) add E => all attributes → **BF** is a key.

- DF+ start {D,F}. From D → A add A => {A,D,F}. From A → BCD add B,C => {A,B,C,D,F}. From BC → DE add E => all attributes → **DF** is a key.

No key can omit F because F is not on RHS of any FD. So any key must include F; minimal left parts that work are A, B, or D with F.

Result: Candidate keys = {AF, BF, DF}. Prime attributes = {A,B,D,F}. Non-prime = {C,E}.

Problem 3 (Student database FD redundancy)
Given FDs: $X \to Y$, $WZ \to X$, $WZ \to Y$, $Y \to W$, $Y \to X$, $Y \to Z$

Goal: Remove redundant FDs and compute a minimal cover.

Step 1: Break RHS to single attributes (already mostly single). List:
1) $X \to Y$
2) $WZ \to X$
3) $WZ \to Y$
4) $Y \to W$
5) $Y \to X$
6) $Y \to Z$

Step 2: Remove extraneous attributes from LHS (none here: WZ are both needed when considered alone).

Step 3: Remove redundant dependencies by checking if each FD is implied by others.
- Is $WZ \to X$ redundant? Compute $(WZ)+$ using the set without $WZ \to X$:
Using $WZ \to Y$ (FD 3) we get Y. From $Y \to X$ (FD 5) we get X. So $WZ \to X$ is implied by $WZ \to Y$ and $Y \to X$ → **redundant**.

- Is $WZ \to Y$ redundant? Compute $(WZ)+$ without $WZ \to Y$ but with $WZ \to X$ and others: with $WZ \to X$ we get X. From $X \to Y$ (FD 1) we get Y. So $WZ \to Y$ is implied by $WZ \to X$ and $X \to Y$ → **redundant**.

- Is $X \to Y$ redundant? Using $Y \to X$ (FD 5) we can see $X \to Y$ is not implied by others unless $Y \leftrightarrow X$ present. But since $Y \to X$ exists, $X \to Y$ is implied only if $Y \to X$ and other allow X to derive Y; check X+ without $X \to Y$: X alone gives nothing else. However if $Y \to X$ exists, that does not give $X \to Y$. So $X \to Y$ is **not** implied by others; keep it.

- Note: There is mutual information: $X \to Y$ and $Y \to X$ together make X and Y equivalent. But one cannot be derived from the other without extra FDs.

However, because both $X \to Y$ and $Y \to X$ are present, and $Y \to W$ and $Y \to Z$ exist, we can simplify by keeping Y as the single determinant: from Y we get X, W, Z (and thus WZ and X). So $WZ \to X$ and $WZ \to Y$ are certainly redundant.

Minimal cover (concise):
One compact minimal cover is: { $Y \to X$, $Y \to W$, $Y \to Z$ }
This implies $X \to Y$ is redundant if you take $Y \to X$ and $X \to Y$ both? If you want to keep symmetry you could also keep $X \to Y$, but the minimal cover above uses Y as the single determinant and drops the WZ FDs.

Final answer: Minimal set of FDs = { $Y \to X$, $Y \to W$, $Y \to Z$ } (or combined: $Y \to XWZ$).