Q.1

a.

A --> B does not hold. a2 has two different values c1 and c2.

b.

B --> D holds. b3 has one value d3.

c.

BCD --> A does not hold. b3, c2, d3 have two different value a2, a3

d.

F={A-->D, B-->D, AB-->D, BC-->D, ABC-->D}

A+ = {A, D}

B+ = {B, D}

e.

AB+ = {A, B, D} c is missing in AB closure so, it is not a super key or candidate key.

Q.2

a.

A+ = {A, D, E}

b.

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

c.

1. B-->C
2. DB-->DC by augmentation of 1
3. DC-->C by reflexivity
4. DB--> C by transitivity of 2 and 3

Q.3

a.

A+ = {A, D, B, C} so, A is the candidate key

b.

BCNF:

In every FD x-->y, x has to be super key

A-->D A is a super key

B-->C B is not a super key

A-->BD A is super key

D-->B D is not super key

Does not satisfy BCNF

c.

3NF:

For every non-trivial FD x-->y:

X is a super key or

Y is a prime attribute

A-->D A is a super key and A is prime attribute

B-->C B is not a super key, C is not prime attribute so, 3NF fails.

Q.4

a.

A+ = {A, D}

A-->D A is not a super key so, BCNF fails.

b.

R1(A, D), R2(B, C, E), R3(A, B, C, F)

ABCDEF

AD ABCEF

(A-->D) BCE ABCF

(BC-->E) (AF-->BC)

c.

The relations include the original FD so, decomposition is dependency preserving.

d.

We cannot reach AF-->BC from R1 and R2 so, it is not lossless.

e.

AF-->BC is lost so, dependency preserving fails.

Q.5

a.

F = {A --> BC, B --> E, BD --> C, AD --> CE, E --> AD}

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

C in B closure so, D is extraneous in BD --> C

b.

F’ = {A --> B, B --> E, BD --> C, AD --> CE, E --> AD}

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

C in A closure so, C is extraneous in A --> BC

c.

Step1: split RHD

F={A-->B, A-->C, B --> E, BD --> C, AD --> C, AD --> E, E --> A, E -->D}

Step2: remove extraneous attributes (D is extraneous in AD-->E because A+ includes E)

F = {A-->B, A-->C, B --> E, B-->C, A-->E, E --> A, E -->D}

Step3: remove redundant functional dependencies

A-->B	A+ ={A, E, C, D} not include B so, not redundant
A-->C	A+ = {A, B, E, C, D} redundant
B --> E	B+ = {B, C} not redundant
B-->C	B+ = {B, E, A, D, C} redundant
A-->E	A+ = {A, B, E, D, C} redundant
E --> A	E+ = {E, D} not redundant
E -->D	E+ = {E, A, B, C} not redundant

Canonical form: F={A-->B, B-->E, E-->A, E-->D}