1)

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

AB -> C

BC -> AD

D ->E

CF -> B

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

2)

R (k1, k2, ..., kn)

Candidate Key = k1

There are k1^n super keys if the only candidate key is k1, because you can add zero or more attributes to a candidate key to produce a Super key.

Therefore one super key is K1, another super key could be k1 + k2, and so on you can keep adding, k1 + k2 + kn and so on are all super keys.

3)

Movies (title, year, length, genre, studioName, starName)

FD:

starName-> title, year, length, genre, studioName

title -> year, length, genre, studioName

year -> title, length, genre, studioName

length -> title, year, genre, studioName

genre -> title, year, length, studioName

studioName -> title, year, length, genre

4)

starName

5)

R (I, J, K, L, M)

IJ→K

KL→M

LM→J

IJ is not a candidate key for this relationship because its closure set does not give you all values in R.

IJ leaves out L and M.

IJL is a candidate key because you can get all values in R with the closure set of IJL.

6)

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

CE -> D

D -> B

C -> A

KEY: CE

prime: C, E

non-prime: D, B, A

Highest FORM: 1NF

This is only 1NF because C -> A is a partial dependency thus not 2nf. Therefore it can't be anything but 1NF

7)

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

AB -> C

BD -> EF

AD -> GH

A -> I

KEY: ABD

prime: A, B, D

non-prime: C, E, F, G, H, I

Highest FORM: 1NF

This is only 1 NF because AB -> c, BD -> EF, AD ->GH, A-> I are all partial dependency thus not 2nf.

8)

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

A -> BC

CD -> E

B -> D

E -> A

Minimal Cover: Already in minimal cover

a -> b

{a}+= a b c d e

w/o {a}+ = a c

Thus needed for minimal cover

a -> c

{a}+= a b c d e

w/o {a}+ = a b d

Thus needed for minimal cover

cd -> e

{cd}+ = c d e a b

w/o {cd}+ = c d

Thus needed for minimal cover

b -> d

{b}+ = b d

w/o {b}+ = b

Thus needed for minimal cover

e -> a

{e}+ = e a b c d

w/o {e}+ = e

Thus needed for minimal cover

9)

R (CSJDPQV)

C -> CSJDPQV

JP -> C

SD -> P

J -> S

R(r1,r2,r3,r4)

r1 = s d p

r2 = j s

r3 = j p c

r4 = c j q v d

Key: c

prime: c

non-prime: s j d p q v

Minimal Cover:

C -> J

C -> D

C -> Q

C -> V

JP -> C

SD -> P

J -> S

Highest FORM: BCNF

10)

Movies(starName, title, year, length, genre, studioName)

Movies(r1, r2)

r1 = starName, title

r2 = title, year, length, genre, studioName

starName-> title, year, length, genre, studioName 3nf

Below all break 3NF due to lack of Super key on the left side or Prime attribute on the right.

title -> year, length, genre, studioName

year -> title, length, genre, studioName

length -> title, year, genre, studioName

genre -> title, year, length, studioName

studioName -> title, year, length, genre

KEY: starName

prime: starName

non-prime: title, year, length, genre, studioName

Highest Form: 3nf *After decomposition

11)

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

R1(A, B, C)

R2(A, D, E)

A -> BC

CD -> E

B -> D

E -> A

If you join R1 and R2 on "A" you can bring the entire relation back together without any data addition or loss.

you can also with this decomposition fulfill the FD's as A -> BC, B -> D, CD ->E and E -> A.

12)

Yes the answer for Q.10 is dependency conserving because you are able to recreate the initial relation without any data loss and are able to keep the dependencies.