3.1

a) {a,c,e} {b,c,e} {d,c,e}
Note (not part of answer): Set notation ({,}) is not required as long as it is clear which attributes are in each key.

b) {b,c,d} {a,b,c,d} {a,b,d,e}

c) Example:

a	b	c	d	e
0	0	0	0	0
0	0	0	1	1

Note (not part of answer): b and c must be the same in both rows, and d differ to violate the functional dependency. Column a is essentially irrelevant and e must be different to respect the keys. Having additional rows is OK as long as the keys are respected.

Another correct solution where values are "as unique as possible":

a	b	c	d	e
0	1	2	3	4
5	1	2	6	7

d)

a → b
b c → d
d e → a

Additional derived dependencies (just a few examples, there are a lot of derived dependencies):

a c → d, b c e → a

R(a,b,c,d,e)
Decomposing on a → b, $\{a\}^+ = \{a, b\}$
R1(a,b) - BCNF
R2(a,c,d,e)
Decomposing on d e → a, $\{d, e\}^+ = \{d,e,a,(b)\}$
R21(d, e, a) - BCNF
R22(d, e, c) - BCNF

Final schema with keys:

R1(<u>a</u>,b)
R21(<u>d</u>, <u>e</u>, a)
R22(<u>d</u>, <u>e</u>, <u>c</u>)

3.2

We can note that the two sets A, B, E and C, D are not interconnected at all and can essentially be separate relations.

C, D can be email and name or any other pair of attributes where the first determines the second but not the other way around.

A, B, E
We have E -> A and A -> B. This could be something like A=model, B=manufacturer, E=licensePlate for cars or any other hierarchy where E is the most specific then A then B. Another example: E=planet, A=solarSystem, B=galaxy.

3.3

MVDs:

flightNo, departure ->> airport
flightNo, departure ->> passenger, movie
Note: The two above are equivalent and express “the airports a flight lands at are independent from the passengers and the movies they can watch”

passenger ->> movie
passenger ->> airport, flightNo, departure
Note: These two are also equivalent and express “The movies a passenger can watch are independent from the flights they are booked on and airports those flights land at”

R(flight, departure, airport, passenger, movie)
decomposing on flightNo, departure ->> airport
  R1(flightNo, departure, airport)
  R2(flightNo, departure, passenger, movie)
  decomposing on passenger ->> movie
    R21(passenger, movie)
    R22(passenger, flightNo, departure)

Final relations with keys (only trivial FDs):

R1(<u>flightNo</u>, <u>departure</u>, <u>airport</u>)

R21(<u>passenger</u>, <u>movie</u>)

R22(<u>passenger</u>, <u>flightNo</u>, <u>departure</u>)

Logically, the first ones lists where flights land, the second what movies passengers have and the third what passengers are on what flights.