1.

a. R1 or R2 = N/A

b. R1 x R2 = N/A

c. R1 join R2 = {
{ID, A, B, D, E},
{0, A0, B0, D0, E0}
}

d. R1 left join R2 on R1.ID == R2.ID = {
{ID, A, B, D, E},
{0, A0, B0, D0, E0},
{1, A1, B1, null, null},
{2, A2, B2, D2, E2},
}

e. R1 right join R2 = {
{ID, A, B, D, E},
{0, A0, B0, D0, E0},
{2, null, B99, D2, E2},
{3, null, B3, D3, E3}
}

f. R1 full join R2 on R1.ID == R2.ID = {
{ID, A, B, D, E},
{0, A0, B0, D0, E0},
{1, A1, B1, null, null},
{2, A2, B2, D2, E2},
{2, A2, B99, D2, E2},
{3, null, B3, D3, E3}
}

2.

q1. π pizza ((σ age > 20 ∧ gender = female (Person)) ⋉ Eats)

q2. π name ((σ gender = "female" (Person)) ⋉ (σ name = "Straw Hat" (Frequents))

q3. π pizzeria ((σ name = "Amy" ∨ name = "Fay" (Eats)) ⋈ (σ price < 10 (Serves)))

q4.π pizzeria ((σ name = "Amy" (Eats)) ∩ (σ name = "Fay" (Eats)) ⋈ (σ prize < 10 (Serves)))

q5. π name (Eats ∩ ((π pizza (σ pizzeria = "Dominos" (Serves))) ⋈ (π name (σ pizzeria ≠ Dominos (Frequents)))))

q6. π pizza (((π pizza (Eats)) - (π pizza (σ age > 24 (Eats ⋈ Person)))) ∪ ((π pizza (Serves)) - (π pizza (σ price > 10 (Serves)))))

q7. ρ a1 (π name, age (σ pizza = "Mushroom" (Eats ⋈ Person)))
ρ a2 (π name, age (σ pizza = "Mushroom" (Eats ⋈ Person)))
π age (a1 ⋈ a2 - ((a1) ⋈ a1.age < a2.age (a2)))

q8. ρ lessThan30 (σ age <= 30 (Person))
ρ pizzasEatenBy30Unders (lessThan30 ⋈ Eats)
π pizzeria (Serves) - (π pizzeria (pizzasEatenBy30Unders ⋈ Serves))

q9.

3.

a.

BG -> AF
BG -> AFB -> CD
BG -> AFBCD -> EF
{BG}+ = {A, B, C, D, E, F, G}

b.

CK = {BG}

c.

G -> A , G -> F => G -> AF : Same anticedent
C->E, E->F => CE->F can go

F = {
AB->CD
C->EF
G->FA
}

CK = {BG}

d.

The AB->CD functional dependency breaks down into the following table:
R1(A, B, C, D) : AB->CD

with everything else going in:
R2(A, C, E, F, G) : C->EF, G->FA

which breaks down further into:
R3(C, E, F) : C->EF

and
R4(G, F, A): G->FA

4.

a.

D -> A

A -> BCD, D -> A, B, C, D
BC -> DE, D -> A, B, C, D, E
{D}+ = {A, B, C, D, E}

b.

CK = {DF, BF, AF}

c.

BC->D is extraneous because B->D
A->D is extraneous because B->D
AB->CD is extraneous because A->BC and B->D

{
A->BC
BC->E
D->A
B->D
}

d.

A->BC, B->D and D->A can go into one relation:
R1(A, B, C, D) : A->BC, B->D, D->A

with whatever's left going in another:
R2(B, C, E, F) : BC->E

Because of the transitive dependency, R1 can be split
R3(A, B, C) : A->BC, and R4(B, D) : B->D

with R2 being split to have F in it's own relation
R5(B, F) and R6(B, C, E) : BC->E

5.

a.

CK={E}

b.

Because B->D, BD->CF can be simplified to B->CDF

{
A->BCDF
B->CDF
E->A
DF->->G
}

c.

The first functional dependency can be put into an initial relation:
R1(A, B, C, D, F) : A->BCDF

with everything else going in:
R2(A, D, E, F, G) : E->A and DF->->G

to fit 4NF, the multi-functional dependency is put in its own relation:
R3(D, F, G) : DF->->G

and everything else in: R4(E, A) : E->A