Datalog and Relational Algebra Sample test questions:

LIKES(drinker, beer) key: all

FREQUENTS(drinker, pub) key: all

SERVES(pub, beer, cost) key: pub, beer

**1. Pubs that serve drinks Joe likes**

ans(Pub) := πpub [ σdrinker = ‘Joe’ (LIKES ⋈ SERVES)]

ans(Pub) :- LIKES(‘Joe’, Beer), SERVES(Pub, Beer, \_).

**2. Drinkers who frequent pubs where drinks cost less than $3**

ans(Drinker) := πdrinker [ σcost < 3 (FREQUENTS ⋈ SERVES)]

ans(Drinker) :- FREQUENTS(Drinker, Pub), SERVES(Pub, \_, Cost), Cost < 3.

**4. Find drinkers who like some beers but do not frequent any pubs**

ans(Drinker) := πdrinker (LIKES) – πdrinker (FREQUENTS)

ans(Drinker) :- LIKES(Drinker, \_), /+ FREQUENTS(Drinker, \_).

**6. Find pubs that serve every beer that Joe likes**

**BEWARE: Does NOT mean the same thing as “Pubs that serve ONLY beer Joe likes”**

JoeLikes(Beer) := πbeer [ σdrinker = ‘Joe’ (LIKES)]

PubServe(Pub, Beer) := πpub, beer (SERVES)

//METHOD 1: DIVISION

ans(Pub) := PubServe(Pub, Beer) / JoeLikes(Beer)

//METHOD 2: MANUALLY DOING DIVISION

//Will have some (pub, beer) combos where the pub doesn’t actually serve the beer

AllPubsCrossBeerJoeLikes(pub, beer) := πpub[PubServe(Pub, Beer)] × JoeLikes(Beer)

//Difference between AllPubsCrossBeerJoeLikes and PubServe gives those (pub, beer) combos where the pub doesn’t actually serve the beer

PubsThatDontServeABeerJoeLikes(pub) := πpub[AllPubsAndBeerJoeLikes(pub, beer) – PubServe(Pub, Beer)]

//Difference between all pubs and PubsThatDontServeABeerJoeLikes is our answer

PubsThatServeEveryBeerJoeLikes(pub) := πpub(PubServe) – PubsThatDontAServeBeerJoeLikes(pub)

PubsThatDontServeABeerJoeLikes(Pub) :- LIKES(‘Joe’, Beer), /+ SERVES(Pub, Beer, \_).

ans(Pub) :- SERVES(Pub, Beer, \_), LIKES(‘Joe’, Beer), NOT PubsThatDontServeABeerJoeLikes(Pub).

**3. Find drinkers who like at least one expensive (over $8) beer that Joe likes**

ExpensiveBeerJoeLikes(Beer) := πbeer [ σdrinker = ‘Joe’ AND cost > 8 (LIKES ⋈ SERVES)]

OthersWhoAlsoLike(Drinker) := πdrinker (LIKES ⋈ ExpensiveBeerJoeLikes(Beer))

**5. Find drinkers who frequent pubs that serve either ’Stella Artois’ or ’Molsons’**

DrinkersWhoFrequentPubsThatServeStella(Drinker) = πdrinker [ σbeer =’Stella Artois’ (FREQUENTS ⋈ SERVES)]

DrinkersWhoFrequentPubsThatServeMolsons(Drinker) = πdrinker [ σbeer =’Molsons’ (FREQUENTS ⋈ SERVES)]

ans(Drinker) = DrinkersWhoFrequentPubsThatServeStella(Drinker) ∪ DrinkersWhoFrequentPubsThatServeMolsons(Drinker)

**7. Find all drinkers who frequent a pub that serves at least 2 beers they like, and one of them for at most $3**

Relational Algebra:

//Drinkers like the beer. Pub serves the beer. Drinker frequents the pub.

All1(Drinker, Beer, Pub, Cost) := LIKES ⋈ FREQUENTS ⋈ SERVES

All2(Drinker, Beer2, Pub, Cost2) := ρbeer -> beer2, cost -> cost2 (All1)

//The natural join finds same drinker, same pub, two different beers

ans(Drinker) := π­drinker­ [ σcost > 3 OR cost2 > 3 (All1 ⋈All1.beer != All2.beer2 All2)]