

The questions that will be discussed are 1, 2, 4a, 4b, 4d, and 4e.

1. **Solution:** In this question (and also Question 2), we are mainly inferring properties of the relation schema from an instance of the schema. Hence the superkeys/candidate keys identified here are generally only possibilities unless we have further information to derive more definite answers.

- (a) Possible superkeys of R are $\{A, C\}$, $\{A, D\}$, $\{A, B, C\}$, $\{A, B, D\}$, $\{A, C, D\}$, and $\{A, B, C, D\}$.
- (b) If $\{A, C\}$ is indeed a superkey of R , then $\{A, C\}$ must also be a candidate key of R since neither $\{A\}$ nor $\{C\}$ is a superkey of R (based on r). Moreover, any superkey which is a superset of $\{A, C\}$ can't be a candidate key of R . The two remaining possible superkeys, $\{A, D\}$ and $\{A, B, D\}$, are possible candidate keys of R . Note that at most one of $\{A, D\}$ and $\{A, B, D\}$ could be a candidate key.

2. **Solution:** The possible foreign keys are W , Y , and Z .

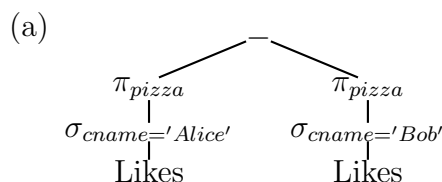
3. **Solution:**

- (a) Equivalent.
- (b) Not equivalent. Q_2 is an invalid relational algebra expression as the selection condition refers to a non-existent attribute.
- (c) Equivalent.
- (d) Equivalent.
- (e) Equivalent.

Note that binary operators in relational algebra are left associative. Hence, in the absence of parenthesis, the expression $R \times S \times T$ is evaluated as $(R \times S) \times T$. As the cross product operator is associative, $(R \times S) \times T$ is equivalent to $R \times (S \times T)$.

- (f) Equivalent.
- (g) Not equivalent. Consider the following database instance d : $r = \{(10, 10)\}$ and $s = \{(10, 20)\}$. The result of Q_1 on d is $\{(10)\}$ while the result of Q_2 on d is \emptyset .

4. **Solution:**

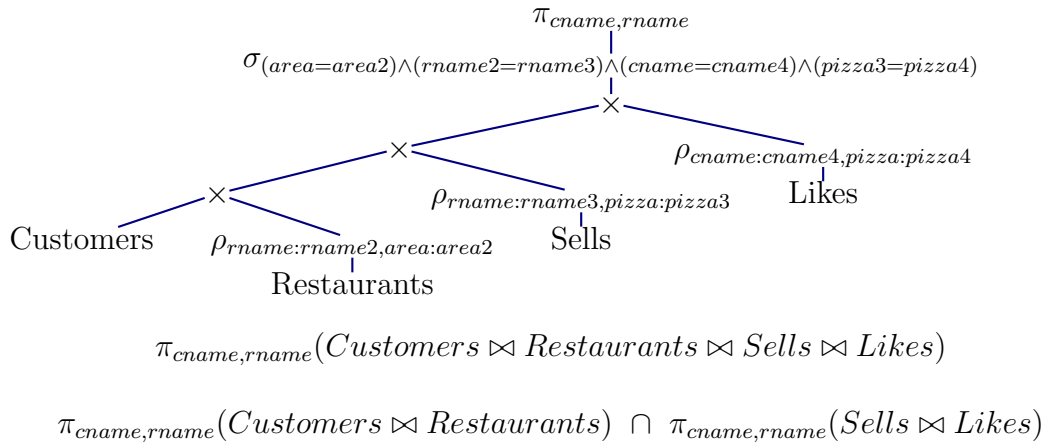


The following answer is incorrect:

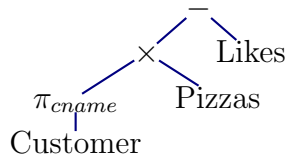
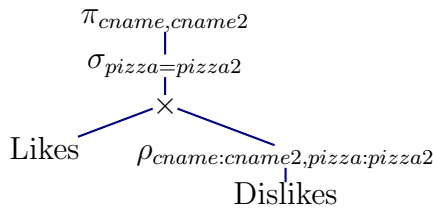
$$\pi_{pizza}(\sigma(\text{pizza=pizza2}) \text{ and } (cname='Alice') \text{ and } (cname2 <> 'Bob'))(Likes \times \rho_{cname:cname2, pizza:pizza2}(Likes)))$$

The above answer will compute exactly the set of pizzas that Alices likes (independent of whether Bob likes them) since for each pizza p that Alice likes, we have $(\text{'Alice'}, p, \text{'Alice'}, p) \in \text{Likes} \times \text{Likes}$.

(b)

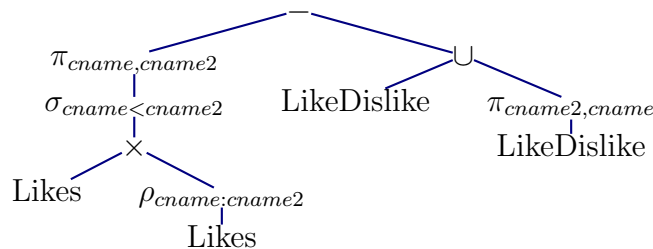


(c)

(d) Let **Dislike**(cname, pizza) denote the output computed by part (c).

(e) Two customers C and $C2$ likes exactly the same set of pizzas if there does not exist any pizza that (1) C likes but $C2$ does not like or (2) C does not like but $C2$ likes.

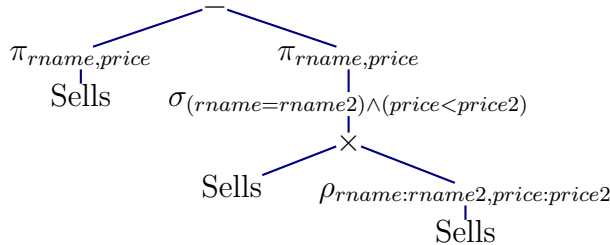
Let **LikeDislike**(cname, cname2) denote the output computed by part (d).



Note that the following answer is incorrect as it computes all customer-pairs $(C1, C2)$ where $C1$ and $C2$ like at least one common pizza but not necessarily exactly the same set of pizzas.

$$\pi_{cname, cname2}(\sigma_{(pizza=pizza2) \text{ and } (cname < cname2)}(\text{Likes} \times \rho_{cname:cname2, pizza:pizza2}(\text{Likes})))$$

- (f) The set $\pi_{rname,price}(Sells)$ can be partitioned into two disjoint subsets $\pi_{rname,price}(Sells) = S_1 \cup S_2$ where S_1 is the set of restaurant name and price pairs (r, pr) such that p is the price of some pizza sold by r that is not the most expensive pizza sold by r , and S_2 is the set of restaurant name and price pairs (r, pr) such that p is the price of the most expensive pizza sold by r . Thus, the required answer is given by $S_2 = \pi_{rname,price}(Sells) - S_1$. Note that $(r, pr) \in S_1$ if there exists some tuple $(r_2, p_2, pr_2) \in Sells$ where $r_2 = r$ and $pr_2 > pr$.



(g)

$$\pi_{cname,pizza}(Customers \rightarrow (Restaurants \bowtie Sells))$$

The following answer is incorrect:

$$\pi_{cname,pizza}(Customers \rightarrow Restaurants \bowtie Sells)$$

The above answer is equivalent to

$$\pi_{cname,pizza}((Customers \rightarrow Restaurants) \bowtie Sells)$$

However, outer joins are generally not associative with inner joins. In particular,

$$(Customers \rightarrow Restaurants) \bowtie Sells \neq Customers \rightarrow (Restaurants \bowtie Sells)$$

To see this, consider a customer tuple $c \in Customers$ who is not co-located with any restaurant. Therefore, c will appear exactly once in the result of $Customers \rightarrow Restaurants$ with a null value for attributes **rname**. Due to the null value for **rname**, this tuple with c will be a dangling tuple w.r.t. the natural join with $Sells$. Thus, c will not be preserved in the result of $(Customers \rightarrow Restaurants) \bowtie Sells$. In contrast, by performing the outer join after the natural join, c will be preserved in the result of $Customers \rightarrow (Restaurants \bowtie Sells)$.



You should pay attention to the join ordering when using outer joins!

The following is another incorrect answer:

$$\pi_{cname,pizza}(Customers \leftrightarrow (Restaurants \leftrightarrow Sells))$$

By using full outer joins for both joins, this answer will also preserve dangling tuples from **Sells** and **Restaurants** relations; thus, the query result could have tuples with a null value for **cname**, but such tuples are not required by the question.

To see this, consider a tuple $(r, p, pr) \in Sells$ where the restaurant r is not co-located with any customer. Thus, (r, p, pr) will join with exactly one tuple from **Restaurants** (say (r, a)), but the resultant tuple (r, a, p, pr) from the first join computation will not join with any tuple from **Customers**. This will produce $(null, p)$ in the query result, which is incorrect.

Moreover, if there is some tuple $r \in Restaurants$ where r is not co-located with any customer and r does not sell any pizza, then this will produce $(null, null)$ in the query result, which is also incorrect.



The type of joins matters: you should use appropriate outer joins to preserve the required dangling tuples only when necessary!

5. Solution:

- R_1 is the set of pizzas that Maggie likes.
- R_2 is the set of restaurant-pizza pairs (r, p) where r is a restaurant that sells some pizza and p is some pizza that Maggie likes.
- R_3 is the set of restaurants that don't sell some pizza that Maggie likes.
- R_4 is the set of restaurants that sell all the pizzas that Maggie likes.
- R_5 is the set of pizzas that Ralph likes.
- R_6 is the set of restaurants that sell some pizza that Ralph likes.
- R_7 is the set of restaurants that sell all the pizzas that Maggie likes and don't sell any pizza that Ralph likes.