CS2102 Tutorial 5: SQL 3 — Wk 7, Sem 2, 2021/22

Questions to be discussed: 1(a)-(e), 1(f)-(j), 2, 3(a)-(c)

1. Consider the following relational schema consisting of two tables.

```sql
CREATE TABLE R (
  a  INTEGER PRIMARY KEY,
  b  INTEGER,
  c  INTEGER
);

CREATE TABLE S (
  x  INTEGER PRIMARY KEY,
  y  INTEGER REFERENCES R(a)
);
```

For each of the following queries on the database, state whether the query is a valid SQL query.

(a) Query A

```sql
SELECT   a, b, SUM(c)
FROM     R JOIN S ON R.a = S.y
GROUP BY a, b;
```

(b) Query B

```sql
SELECT   b, SUM(c)
FROM     R JOIN S ON R.a = S.y
GROUP BY a, b;
```

(c) Query C

```sql
SELECT   a, b, SUM(c)
FROM     R JOIN S ON R.a = S.y
GROUP BY a;
```

(d) Query D

```sql
SELECT   a, b, x, SUM(c)
FROM     R JOIN S ON R.a = S.y
GROUP BY a, b;
```

(e) Query E

```sql
SELECT   a, b, y, SUM(c)
FROM     R JOIN S ON R.a = S.y
GROUP BY a, b;
```

(f) Query F

```sql
SELECT   a, b
FROM     R JOIN S ON R.a = S.y
GROUP BY a, b
HAVING   SUM(c) > 10;
```

(g) Query G

```sql
SELECT   a, b
FROM     R JOIN S ON R.a = S.y
GROUP BY a, b
HAVING   SUM(x) > 10;
```

(h) Query H

```
SELECT    a, b
FROM      R JOIN S ON R.a = S.y
WHERE     SUM(c) > 10;
```

(i) Query I

```
SELECT    a, b
FROM      R JOIN S ON R.a = S.y
GROUP BY a, b
WHERE     SUM(x) > SUM(c);
```

(j) Query I

```
SELECT    b, SUM(c)
FROM      R;
```

2. Consider the same relational schema from the previous question.

```sql
CREATE TABLE R (
  a  INTEGER PRIMARY KEY,
  b  INTEGER,
  c  INTEGER
);

CREATE TABLE S (
  x  INTEGER PRIMARY KEY,
  y  INTEGER REFERENCES R(a)
);
```

Are these two queries *equivalent*?

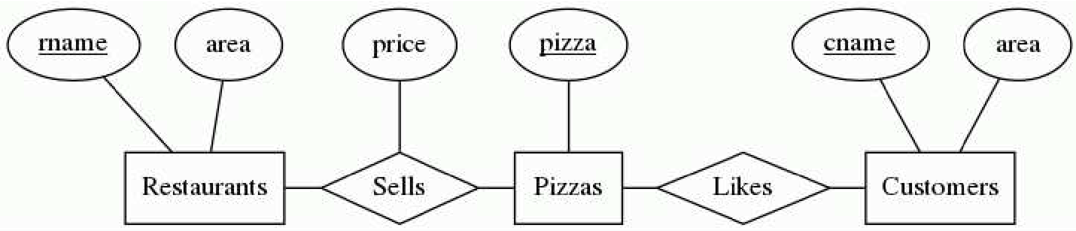
(a) Q_1

```sql
SELECT   COUNT(c)
FROM     R
WHERE    a = 10;
```

(b) Q_2

```sql
SELECT   COUNT(c)
FROM     R
WHERE    a = 10
GROUP BY a;
```

3. This question is based on the pizza database schema shown below.



Answer each of the following queries using SQL. For parts (a) to (e), remove duplicate records from all query results.

(a) Find the most expensive pizzas and the restaurants that sell them (*at the most expensive price*). Do **NOT** use any aggregate function in your answer.

(b) Find all restaurant pairs (R_1, R_2) such that the price of the most expensive pizza sold by R_1 is *higher* than than that of R_2. Exclude restaurant pairs where R_2 do not sell any pizza.

(c) For each restaurant that sels some pizza, find the restaurant name and the average price of its pizzas if its average price is higher than \$22. Do **NOT** use the `HAVING` clause in your answer.

(d) For each restaurant R that sells some pizza, let $totalPrice(R)$ denote the total price of all the pizzas sold by R. Find all pairs $(R, totalPrice(R))$ where $totalPrice(R)$ is *higher* than the average of $totalPrice()$ over all the restaurants.

(e) Find the customer pairs (C_1, C_2) such that $C_1 < C_2$ and they like *exactly* the same pizzas. Exclude customer pairs that do not like any pizza. Do **NOT** use the `EXCEPT` operator in your answer.

(f) Write an SQL statement to *increase* the selling prices of pizzas as follows:

- Increase by \$3 if the restaurant is located in `'Central'`.
- Increase by \$2 if the restaurant is located in `'East'`.
- Increase by \$1 *otherwise*.

4. In the lecture, we have discussed an algebraic approach to derive a SQL query involving universal quantification. In this question, we consider another approach to derive the same SQL query based on predicate logic.

   The following predicate logic expression specifies the set of names of students who have enrolled in every course offered by the CS department.

$$\{S.name \mid S \in Students \wedge \forall C(C \in Courses \wedge C.dept = \texttt{'CS'} \implies$$

$$\exists E(E \in Enrolls \wedge E.sid = S.studentId \wedge E.cid = C.courseId))\}$$

   Rewrite the above expression into an equivalent expression without using any universal quantifier and implication operators, and translate it into an equivalent SQL query.