CS2102 Tutorial 5: SQL 3 **Wk 8, Sem 2, 2020/21**

1. **Solution:**
   (a) Valid

   (b) Valid. It's fine for an attribute in the group-by clause to be absent from the select clause.

   (c) Valid. It's fine for attribute b to be in the select clause since the primary key of R is included in the group-by clause.

   (d) Invalid. Attribute x is missing from group-by clause.

   (e) Invalid. Attribute y and the primary key of S are missing from group-by clause.

   (f) Valid

   (g) Valid

   (h) Invalid. Aggregate function can't be used in the where clause.

   (i) Valid

   (j) Invalid. A query that has no group by clause can't have both aggregate function and non-aggregate attribute in the select clause.

2. **Solution:** Q1 and Q2 are not equivalent queries. Consider the case where is no record in R with a = 10. For Q1, the where clause will evaluate to an empty table and the aggregate function count(c) will evaluate to a single-row table with a single-column value of 0. For Q2, the where clause will evaluate to an empty table, the group-by clause will not produce any group and hence count(c) will not be evaluated at all. Consequently, the query result is an empty table.

3. **Solution:**
   (a) **Solution 1**:

   ```
   select pizza, rname
   from Sells
   where price >= all (select price from Sells);
   ```

   Solution 2:

   ```
   select pizza, rname
   from Sells S1
   where not exists (
       select  1
       from    Sells S2
       where   S2.price > S1.price
   );
   ```

   (b) **Solution 1**:

```
select R1.rname, R2.rname
from Restaurants R1, Restaurants R2
where (select max(price) from Sells where rname = R1.rname)
> (select max(price) from Sells where rname = R2.rname);
```

Solution 2:

```
with RestMaxPrice as (
    select rname, (
        select max(price)
        from Sells
        where rname = R.rname
        ) as maxPrice
    from Restaurants R
)
select R1.rname, R2.rname
from RestMaxPrice R1, RestMaxPrice R2
where R1.maxPrice > R2.maxPrice;
```

(c)
```
with RestaurantAvgPrice as
    (select rname, avg(price) as avgPrice
    from Sells
    group by rname)
select *
from RestaurantAvgPrice
where avgPrice > 22;
```

(d) **Solution 1**:

```
with RestaurantTotalPrices as
    (select rname, sum(price) as totalprice
    from Sells
    group by rname)
select rname, totalPrice
from RestaurantTotalPrices
where totalprice >
    (select avg(totalprice) from RestaurantTotalPrices);
```

Solution 2:

```
select rname, sum(price) as totalprice
from Sells S
group by rname
having sum(price)  >
    (select sum(price) / count(distinct rname) from Sells)
;
```

Invalid query:

```
select rname, sum(price) as totalprice
from Sells S
group by rname
having totalprice  >
    (select sum(price) / count(distinct rname) from Sells)
;
```

`totalprice` is undefined in the `HAVING` clause as the SELECT clause is evaluated after the HAVING clause.

Wrong answer:

```
select rname, sum(price)
from Sells
group by rname
having sum(price) > sum(price) / count(*);
```

The above query is incorrect because both *sum*(*price*) and *count*(∗) in the HAVING clause are computed w.r.t. a group.

(e) **Solution 1**: Customers C1 and C2 like exactly the same pizzas if and only if (a) for every pizza that C1 likes, C2 also likes that pizza, and (b) for every pizza that C2 likes, C1 also likes that pizza. Condition (a) holds if there does not exists any pizza that C1 likes that C2 does not like.

```
select C1.cname, C2.cname
from Customers C1, Customers C2
where C1.cname < C2.cname
and exists (select 1 from Likes L where L.cname = C1.cname)
and not exists (
    select 1
    from Likes L1
    where L1.cname = C1.cname
    and not exists (
        select 1
        from Likes L2
        where L2.cname = C2.cname
        and L2.pizza = L1.pizza
    )
)
and not exists (
    select 1
    from Likes L2
    where L2.cname = C2.cname
    and not exists (
        select 1
        from Likes L1
        where L1.cname = C1.cname
        and L1.pizza = L2.pizza
    )
);
```

Solution 2: This solution is based on the property that if customer C1 likes the set of pizzas S1 and customer C2 likes the set of pizzas S2, then S1 = S2 iff $|S1 \cap S2| = |S1| = |S2|$, where $|X|$ denote the cardinality of a set X.

```
with NumLike as  (
    select cname, count(*) as num from Likes L
    group by cname
),
NumBothLike as  (
    select L1.cname as cname1, L2.cname as cname2, count(*) as num
    from Likes L1, Likes L2
    where (L1.pizza = L2.pizza) and (L1.cname < L2.cname)
    group by L1.cname, L2.cname
)
select cname1, cname2
from NumBothLike B
where num =  (select num from NumLike N where N.cname = B.cname1)
and num =  (select num from NumLike N where N.cname = B.cname2);
```

Solution 3: This solution is based on the property that if customers C1 and C2 like exactly the same pizzas, then for the subset of all tuples $S \subseteq \textit{Likes}$ that are associated with C1 or C2, there must be exactly two tuples in S associated with each distinct pizza in S.

```
select C1.cname, C2.cname
from Customers C1, Customers C2
where C1.cname < C2.cname
and C1.cname in (select cname from Likes)
and not exists (
    select 1
    from Likes
    where cname in (C1.cname, C2.cname)
    group by pizza
    having count(*) <> 2
);
```

(f)
```
update Sells S
set price =
    case (select area from Restaurants where rname = S.rname)
    when 'Central' then price + 3
    when 'East' then price +2
    else price + 1
    end;
```

4. **Solution:** Using the rewriting rule $p \implies q \equiv \neg\, p \vee\, q$, we have

$$\{S.name \mid S \in Students \;\wedge\; \forall\, C(C \notin Courses \;\vee\; C.dept \neq' CS' \vee$$
$$\exists\, E(E \in Enrolls \;\wedge\; E.sid = S.studentId \;\wedge\; E.cid = C.courseId))\}$$

Using the rewriting rule $\forall\, C(p) \equiv \neg\,(\exists\, C(\neg\, p))$, we have

$$\{S.name \mid S \in Students \;\wedge\; \neg\,(\exists\, C(C \in Courses \;\wedge\; C.dept =' CS' \wedge$$
$$\neg\,(\exists\, E(E \in Enrolls \;\wedge\; E.sid = S.studentId \;\wedge\; E.cid = C.courseId))))\}$$

The above can be translated into the following SQL query:

```
select S.name
from Students S
where not exists (
    select 1
    from Courses C
    where C.dept = 'CS'
    and not exists (
        select 1
        from Enrolls E
        where E.sid = S.studentId
        and E.cid = C.courseId
    )
);
```