CS2102 Tutorial 5

1. If attribute A_i of table R appears in SELECT / HAVING clause

   ① A_i appears in GROUP BY clause

   ② A_i is input to aggregation function

   ③ primary key of R appear in GROUP BY clause

a) a, b appear in GROUP BY clause — valid ✓
   a also primary key
   c is input to SUM aggregation function

b) similar to a) — valid ✓

c) a is primary key — valid ✓
   c is input to SUM aggregation function

d) x does not appear in GROUP BY clause
   x is not input to aggregation function — invalid ✗
   b/c of s not in GROUP BY

e) since y is the foreign key of a, its value in each group is unique => could be valid?

f) a is primary key — valid ✓
   c is input to aggregation function

g) a is primary key — valid ✓
   x is input to aggregation function

h) valid but not meaningful ?

using aggregation in where clause

SUM(c) > 10 equivalent to c > 10 since just 1 tuple

i) x is input to aggregate function, valid ✓

j) valid but not meaningful ?

SUM(c) value same for all values of b

↳ sum of attribute c over all tuples in R

2. since a is a primary key,

GROUP BY a will only have groups with 1 tuple each

same as R

equivalent //

3 a)

```
FROM Sells S1
WHERE NOT EXISTS ( SELECT 1
                   FROM Sells S2
                   WHERE S2.pizza = S2.pizza AND S2.price > S1.price
```

or price >= ALL (SELECT S2.price
FROM Sells S2
WHERE S1.pizza = S2.pizza

b) Using common table expression

```
WITH maxPrice AS (
    SELECT pname, ( SELECT MAX(price) FROM Sells WHERE pname = R.pname ) AS maxP
    FROM Restaurant R
)
SELECT R1.name, R2.name
FROM maxPrice R1, maxPrice R2
WHERE R1.maxP > R2.maxP
```

c)
```
SELECT pname, AVG(price) AS avgPrice
FROM Sells
GROUP BY pname
```
} equivalent

d)
```
WITH Total1 AS (
    SELECT pname, SUM(price) AS sumP
    FROM Sells
    GROUP BY pname
)
T1.sumP > ( SELECT AVG (sumP) FROM Total1 T2);
```

e)
```
SELECT C1.name, C2.name
FROM Customer C1, Customer C2
WHERE C1.cname < C2.cname
AND EXISTS ( SELECT 1 FROM Likes WHERE cname = C2.cname)
```
C1 likes same pizza as C2 } AND NOT EXISTS (SELECT 1 FROM Likes L1
WHERE cname = C1.cname
AND NOT EXISTS (SELECT 1 FROM Likes L2 WHERE cname = C2.cname AND price = L1.price)

C2 likes same pizza as C1 } AND NOT EXISTS

(there does not exist a pizza liked by C1 not liked by C2 and vice versa → like at least 1 pizza) (universal quantification)

f) use case statement

```
UPDATE sells S
SET price = CASE ( SELECT area FROM Restaurant WHERE rnum = S.rnum )
            WHEN 'county' THEN price + 1
            WHEN 'east'   THEN price + 2
            ELSE price + 1
          END;
```

4.

students who have enrolled in every course offered by CS department

there does not exist a CS course the student has not enrolled

All CS courses student with sid = x has not enroll

```
SELECT C.cId
FROM C
WHERE NOT EXISTS ( SELECT 1
                   FROM E
                   WHERE E.cID = C.cID
                   AND E.sID = x )
```

C.dn = 'CS'

```
SELECT S.nam
FROM S
WHERE NOT EXISTS ( SELECT C.cId
                   FROM C
                   WHERE NOT EXISTS ( SELECT 1
                                      FROM E
                                      WHERE E.cID = C.cID
                                      AND E.sID = S.sID ));
```