CS2102 Tutorial 4: SQL 2

Wk 6, Sem 2, 2017/8

1. **Solution:**

(a) **select** ∗ **from** R **natural join** S;

A	B	X	Y	Z	C	D
8	5	30	0	1	60	100
4	3	60	1	3	30	100

(b) **select** ∗ **from** R **inner join** S **on** R.A = S.A;

X	A	Y	B	Z	A	B	C	D
30	8	0	5	1	8	5	60	500
60	4	1	3	3	4	2	40	200
60	4	1	3	3	4	3	30	100

(c) **select** ∗ **from** R **left outer join** S **on** R.A = S.A;

X	A	Y	B	Z	A	B	C	D
0	10	0	9	2	null	null	null	null
30	8	0	5	1	8	5	60	500
60	4	1	3	3	4	2	40	200
60	4	1	3	3	4	3	30	100
90	0	0	4	5	null	null	null	null

(d) **select** ∗ **from** R **right outer join** S **on** R.A = S.A;

X	A	Y	B	Z	A	B	C	D
30	8	0	5	1	8	5	60	500
60	4	1	3	3	4	2	40	200
60	4	1	3	3	4	3	30	100
null	null	null	null	null	17	1	20	100

(e) **select** ∗ **from** R **full outer join** S **on** R.A = S.A;

X	A	Y	B	Z	A	B	C	D
0	10	0	9	2	null	null	null	null
30	8	0	5	1	8	5	60	500
60	4	1	3	3	4	2	40	200
60	4	1	3	3	4	3	30	100
90	0	0	4	5	null	null	null	null
null	null	null	null	null	17	1	20	100

2. **Solution:**

(a)
```
select rname, max(price) as maxprice
from Sells
group by rname
```

(b)
```
select rname, avg(price)
from Sells
group by rname
having avg(price) > 22;
```

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

(d)
```
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);
```

Note that the following solution is incorrect (it happens to compute the correct output for the provided database instance).

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

The aggregate expressions in a HAVING clause are computed with respect to the records in some group.

Thus, the above HAVING clause condition is effectively comparing the sum of the prices of a group with the group's average price; this will evaluate to *true* if the group has more than one record; and *false*, otherwise (assuming for simplicity that all prices are non-null values).

The following is another possible solution (contributed by Edward Tu):

```
select rname, sum(price)
from Sells S
group by rname
having sum(price)  >
```

```
        (select sum(price) / count(distinct rname) from Sells)
    ;
```

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

Another approach is based on the property that $A \cap B \subseteq A$ for any two sets A and B; hence $A \cap B = A$ iff $A \cap B$ and A have the same cardinality.

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

The following is another solution (contributed by Zhu Chunqi):

```
with CustPair as
    (select distinct L1.cname as cname1, L2.cname as cname2
    from Likes L1, Likes L2
    where L1.cname < L2.cname
    and L1.pizza = L2.pizza)
select *
from CustPair CP
where not exists (
    select 1
    from Likes L
    where L.cname in (CP.cname1, CP.cname2)
    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 * 1.20
    when 'East' then price * 1.10
    else price * 1.05
    end;
```