CS2102 Tutorial 4: SQL 2

Wk 6, Sem 2, 2020/21

1. **Solution:**

a	b
2	20
3	30

(a)

Invalid query

(b)

a	b
2	20
3	30

(c)

a	b
1	10
2	20
3	30
4	40

(d)

a	b
3	30

(e)

Query (b) is invalid because the alias f for table foo declared in the inner query is not visible to the outer query.

In query (c), "a = b.a" is equivalent to "f.a = b.a". In query (d), "f > 100" is equivalent to "f.f > 100", and "a = a" is equivalent to "f.a = f.a". In query (e), "b > 20" is equivalent to "b.b > 20".

It is a good defensive programming practice to use distinct aliases and use explicitly qualified column names.

2. **Solution:**

(a)
```
select distinct cname
from Likes L, Sells S
where L.pizza = S.pizza
and S.rname = 'Corleone Corner';
```

(b)
```
select cname from Customers
except
select cname
from Likes L, Sells S
where L.pizza = S.pizza
and S.rname = 'Corleone Corner';
```

(c)
```
select distinct S.rname
from Sells S, Sells S2
where S.rname <> 'Corleone Corner'
and S2.rname = 'Corleone Corner'
and S.price > S2.price;
```

(d)
```
select rname, pizza, price
from Sells
where price is not null
except
select S.rname, S.pizza, S.price
from Sells S, Sells S2
where S.rname = S2.rname
and S.price < S2.price;
```

3. **Solution:**

(a) **Solution 1**:

```
select pizza
from Likes
where cname = 'Alice'
and pizza not in
    (select pizza
    from Likes
    where cname = 'Bob');
```

Solution 2:

```
select pizza
from Likes L1
where cname = 'Alice'
and not exists (
    select  1
    from    Likes L2
    where   L2.cname = 'Bob'
    and     L2.pizza = L1.pizza
    );
```

Solution 3:

```
select pizza
from Likes
where cname = 'Alice'
and not pizza = any (
    select pizza
    from Likes
    where cname = 'Bob'
);
```

Note that if Bob doesn't like any pizza, then the ANY subquery will evaluate to $false$, and "not $false$" will evaluate to *true*; thus, the query will return all the pizzas that Alice likes.

Solution 4:

```
select pizza from Likes where cname = 'Alice' except
select pizza from Likes where cname = 'Bob';
```

Wrong answer: The following answer is incorrect.

```
select pizza
from Likes
where cname = 'Alice'
and pizza <> any (
    select  pizza
    from    Likes
```

```
    where   cname = 'Bob'
    );
```

This answer looks similar to Solution 3 but it is incorrect. If Bob doesn't like any pizza, then the ANY subquery will evaluate to $false$ and the query will return an empty set, which is incorrect if Alice likes some pizza.

(b) A pizza is the output if there does not exist two distinct restaurants that are located in the same area selling that pizza.

```
select distinct pizza
from Sells S3
where not exists (
    select 1
    from  Sells S, Restaurants R, Sells S2, Restaurants R2
    where S.rname = R.rname
    and S2.rname = R2.rname
    and S.pizza = S2.pizza
    and R.area = R2.area
    and R.rname <> R2.rname
    and S.pizza = S3.pizza
);
```

Wrong answer: The following answer is incorrect.

```
select distinct pizza
from Sells S, Restaurants R
where S.rname = R.rname
and not exists (
    select 1
    from  Sells S2, Restaurants R2
    where S2.rname = R2.rname
    and S2.pizza = S.pizza
    and R.area = R2.area
    and R2.rname <> R.rname
);
```

This answer computes the pizzas that are sold by at most one restaurant in **some** area, which is a weaker condition than what is required by the question.

(c)
```
select distinct R.area, S.pizza, S.price
from Restaurants R, Sells S
where R.rname = S.rname
and S.price <= all (
    select S2.price
    from Restaurants R2, Sells S2
    where R2.rname = S2.rname
    and R2.area = R.area
    and S2.pizza = S.pizza
);
```

(d) You should recognize that this query is simply an extension of the previous query requiring an additional information (highest selling price) for each area-pizza pair. For a given area-pizza pair (A, P), the following query will compute the highest price of pizza P in area A:

```
select distinct S2.price
from Restaurants R2, Sells S2
where R2.rname = S2.rname
and R2.area = A
and S2.pizza = P
and R2.price >= all (
    select price
    from Restaurants R3, Sells S3
    where R3.rname = S3.rname
    and R3.area = A
    and S3.pizza = P
);
```

Since the above query will return a single one-column tuple, it can be used as a scalar subquery to extend the previous question's solution as follows.

```
select distinct R.area, S.pizza, S.price as minPrice, (
    select distinct S2.price
    from Restaurants R2, Sells S2
    where R2.rname = S2.rname
    and R2.area = R.area
    and S2.pizza = S.pizza
    and S2.price  >= all (
        select S3.price
        from Restaurants R3, Sells S3
        where R3.rname = S3.rname
        and R3.area = R.area
        and S3.pizza = S.pizza
        )
    ) as maxPrice
from Restaurants R, Sells S
where R.rname = S.rname
and S.price <= all (
    select S2.price
    from Restaurants R2, Sells S2
    where R2.rname = S2.rname
    and R2.area = R.area
    and S2.pizza = S.pizza
);
```

We will learn about other (simpler and more elegant) solutions for such queries later in class.

4. **Solution:** Queries $Q1$ and $Q2$ are equivalent. Whether the selection predicate "area = 'East'" is evaluated before or after the join/cross product operation does not change the semantics of the query.

5. **Solution:** Queries $Q1$ and $Q2$ are not equivalent. Observe that if the `Likes` relation is empty, then query $Q1$ simplifies to $\pi_{rname}(Sells \bowtie \sigma_{\text{area='East'}}(Restaurants))$, but the result of $Q2$ is always an empty set due to $Sells \times Restaurants \times \emptyset$.

6. **Solution:**

```
update Employees
set officeId =
    (select officeId from Offices
    where building = 'Tower1'
    and level = 5
    and roomNumber = 11)
where officeId = 123;
```

7. **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