1.
```
select distinct S.rname
from Likes L, Sells S
where L.cname = 'Alice'
and S.pizza = L.pizza
```

2.
```
select distinct S.pizza
from Customers C, Sells S, Restaurants R
where C.cname = 'Bob'
and C.area = R.area
and R.rname = S.rname
```

3.
```
select cname from Likes group by cname having count(*) >= 2
```

4.
```
select distinct S1.rname, S2.rname
from Sells S1, Sells S2
where  S1.pizza = S2.pizza
and not exists (
    select 1
    from Sells S3, Sells S4
    where  S3.rname = S1.rname
    and  S4.rname = S2.rname
    and S3.pizza = S4.pizza
    and S3.price <= S4.price
    )
```

5.
```
select cname
from Customers  C
where exists (select 1 from Restaurants R where area = C.area)
and not exists (
    select 1
    from Sells  S, Restaurants R
    where  S.rname = R.rname
    and R.area = C.area
    and S.pizza not in (
        select pizza
        from  Likes
        where cname = C.cname
    )
)
```

6.
```
select rname
from Restaurants R
where exists (
    select 1
    from Sells S1, Sells S2
    where S1.rname = S2.rname
    and S1.rname = R.rname
    and S1.pizza <> S2.pizza
    and S1.price + S2.price <= 40
)
```

7.
```
-- (Cpizza, Lpizza, Mpizza) in ThreePizzas if {Cpizza, Lpizza, Mpizz} are distinct
-- pizzas where Curly likes Cpizza, Larry likes Lpizza, and Moe likes Mpizza;
-- and each of them likes at least two of the three pizzas
with ThreePizzas as
   (select LC.pizza as Cpizza, LL.pizza as Lpizza, LM.pizza as Mpizza
   from Likes LC, Likes LL, Likes LM
   where LC.cname = 'Curly'
   and LL.cname = 'Larry'
   and LM.cname = 'Moe'
   and LC.pizza <> LL.pizza
   and LC.pizza <> LM.pizza
   and LL.pizza <> LM.pizza
   and ((LL.pizza in (select pizza from Likes where cname='Curly'))
      or (LM.pizza in (select pizza from Likes where cname='Curly')))
   and ((LC.pizza in (select pizza from Likes where cname='Larry'))
      or (LM.pizza in (select pizza from Likes where cname='Larry')))
   and ((LC.pizza in (select pizza from Likes where cname='Moe'))
      or (LL.pizza in (select pizza from Likes where cname='Moe')))
   )
select distinct S1.rname,
   case
   when (T.Cpizza < T.Lpizza) and (T.Cpizza < T.Mpizza) then T.Cpizza
   when (T.Lpizza < T.Cpizza) and (T.Lpizza < T.Mpizza) then T.Lpizza
   else T.Mpizza
   end as pizza1,
   case
   when (T.Lpizza < T.Cpizza) and (T.Cpizza < T.Mpizza) then T.Cpizza
   when (T.Mpizza < T.Cpizza) and (T.Cpizza < T.Lpizza) then T.Cpizza
   when (T.Cpizza < T.Lpizza) and (T.Lpizza < T.Mpizza) then T.Lpizza
   when (T.Mpizza < T.Lpizza) and (T.Lpizza < T.Cpizza) then T.Lpizza
   else T.Mpizza
   end as pizza2,
   case
   when (T.Lpizza < T.Cpizza) and (T.Mpizza < T.Cpizza) then T.Cpizza
   when (T.Cpizza < T.Lpizza) and (T.Mpizza < T.Lpizza) then T.Lpizza
   else T.Mpizza
   end as pizza3,
   S1.price + S2.price + S3.price as totalprice
from Sells S1, Sells S2, Sells S3, ThreePizzas T
where S1.rname = S2.rname
and S1.rname = S3.rname
and S1.price + S2.price + S3.price <= 80
and S1.pizza = T.Cpizza
and S2.pizza = T.Lpizza
and S3.pizza = T.Mpizza
```

8.
```
with RestaurantInfo as
   (select rname,
   (select count(*) from Sells S where rname = R.rname) as numPizza,
   (select
      case
      when max(price) - min(price)  is null then 0
      else max(price) - min(price)
      end
      from Sells S where rname = R.rname) as priceRange
   from Restaurants R)
select rname
from RestaurantInfo I
where  not exists (
   select 1
   from RestaurantInfo I2
   where ((I2.numpizza > I.numpizza) and (I2.priceRange >= I.priceRange))
   or ((I2.numpizza >= I.numpizza) and (I2.priceRange > I.priceRange))
)
```

The case expression could be replaced by **coalesce**(**max**(price)-**min**(price),0).

9.
```
select C.area, count(distinct C.cname), count(distinct R.rname),
case
   when max(S.price) is null then 0
   else max(S.price)
end
from Customers C natural left outer join
   (Restaurants R natural left outer join Sells S)
group by C.area
```

The two left outer joins are necessary to correctly compute the required counts. The first left outer join preserves a customer C even if there are no restaurants co-located with C. The second left outer join preserves a restaurant R even if R is not selling any pizzas so long as R is co-located with some customer. The case expression could be replaced by **coalesce**(**max**(S.price),0).

10.
```
select rname
from Restaurants R
where (select count(*) from Sells S where rname = R.rname) >= 3
and exists (select 1 from Sells S where rname = R.rname and price < 20)
and not exists (
   select 1
   from Customers C left outer join
   (Likes L inner join Sells S on S.pizza = L.pizza and S.rname = R.rname)
   on C.cname = L.cname
   group by C.area having count(distinct L.cname) < 2
)
```

The left outer join is necessary to ensure that condition (c) is satisfied. Specifically, the left outer join preserves a customer C even if C does not like any pizza or none of the pizzas that C like are sold by the restaurant R.