Solutions to the sample Revision Exercises

1.
(a)
You should design an ER diagram with the following components:
Entity **Employee** with cf as key
Entity **Manager** (but since its key is also the key for Employee, then it is **isA** to employee)
Apprentice has 2 attributes that are FR to two tables so most likely is a relationship. And since the key is only one of the two FK, it means that it is Many to one (arrow normal (not bold) from Employee towards Manager). (Another modeling it for Apprentice to be isA to Manager, and have a coach relationship to Manager that many to one.)
The **Project** has a key that consists of something that is FK of another table, which is the modeling of a weak entity to Manager.

(b) YES. It is independent of one another so it is fine.

2
(a) YES (it CAN be. We do not know if it is for sure, but it is possible. We noticed no violation. All the values on BC exist as DE)
(b) YES it CAN be. Not sure if it is, but we are sure that it CAN be, since there is no duplication.
(c) NO (because of the null values)

3.
(a)
(assume that the table T had an imaginary column (call it G) where the value of that imaginary column was the value of column C plus 10).
Then the table T would have been:

```
------------------------------------
B         C         D         G
------------------------------------
1         10        NULL   20
2         20        f         30
3         NULL   w         NULL
```

The join between S and T is executed first, so the result is

```
S                   T
A         B         B         C         D         G
=================================
2         20        1         10        NULL   20
3         30        1         10        NULL   20
3         30        2         20        f         30
```

(note that a any number added to a value NULL gives result NULL. And any comparison with NULL gives result NULL. However when we group, NULLS go together.)

Then we do the group by of the above relation based on T.D. Notice that for the GROUPING only, two values that are both NULL are considered the same, which means that the first two tuples will create one group and the third will create a group by itself.

As a result the final output is the table:

```
========
f       3
NULL    3
```

(b) Notice here that the = on which the join is based will give false if one of the values is NULL so the output will be

```
=========================
f            NULL         f
w            o            w
w            f            w
```

Question 4: You need to have:
An entity Customers with attributes CardID, FName and LName (CardID being the Key)
An entity RegularCustomers that is an isA to Customer
An entity PremiumCustomers that is an isA to Customer and has attributes Date and Points
An entity Products with attributes Name, VAT, Price and ProdID (ProdID is the key)
An entity Moment that has attributes Date and Time (the key is formed by both these attributes)
A relationship RegularPurchase between Moment, Products and RegularCustiomers, that has the attributes Quantity and Price.
A relationship PrtemiumPurchase between Moment, Products and PremiumCustiomers, that has the attributes Quantity, Price, Discount and Points.

Question 5:
(a)
$\rho(R1,Review) \bowtie_{R1.code=R2.code \wedge R1.supplier=R2.supplier \wedge (R1.user!=R2.user \vee R1.date!=R2.date)} \rho(R2,Review)$
-
$\rho(R1,Review) \bowtie_{R1.code=R2.code \wedge R1.supplier=R2.supplier \wedge (R1.user!=R2.user \vee R1.date!=R2.date)} \rho(R2,Review)$
$\bowtie_{R1.code=R3.code \wedge R1.supplier=R3.supplier \wedge (R1.user!=R3.user \vee R1.date!=R3.date) \wedge (R2.user!=R3.user \vee R2.date!=R3.date)}$
$\rho(R3,Review)$

The first line gives all those that have at least 2 reviews, while the second gives those that have at least 3 reviews.

(b) $\pi_{supplier}$ Product   -

$$\pi_{supplier} ((\rho(Suppliers, \pi_{supplier}Product) \times \rho(DiscProductCodes, \pi_{code,}\sigma_{discontinued=true}Product)) - \rho(SupDiscProductsNowSold, \pi_{code,\ supplier}\sigma_{discontinued=true}Product)))$$

all the possible <supplier-DiscProduct> pairs MINUS <supplier-DiscProduct> pairs that are now sold gives the pairs of a supplier with a disc product that are currently not sold.
Clearly these suppliers are NOT among those that sell EVERY discontinued product so they should NOT be in the answer set. So we take all the suppliers and we cut away those.

(c) $\pi_{code,supplier}Product - \pi_{code,supplier}(Product \bowtie Review)$

(d)
select supplier, avg(NoOfReviews)
from (select supplier, code, sum(*) as NoOfReviews from Reviews group by supplier, code) as T
group by supplier

(e)
select supplier
from Review
group by supplier
having AVG(rating) >= ALL (select avg(rating) from Reviews group by supplier)

6.
The FDs that hold are:
(a)
supplier→ brand,address (because a supplier sells products of a single brand and address)
suplier,code → name_prod,price (because a product has a single name and price, --- but this is covered anyway from the key property)
user→ email (Because a user owns only a single email)

(b)
So we create one table [user,email]
One table [supplier,brand,address]
And the other tables remain as is but without the attributes email address and brand.

7.
I. No because it violates A→ E II. Yes. It does not seem to violate anything

8.
(a) No (because the selection a=3 on S is not present in the Cartesian product)
(b) Yes (Why not? The keys cannot be null… the FK have no such problem)

(c) Yes (actually an intersection can be done with only a join.

9. True
10. False (you can always define 2 indexes on the same relation independently on whether they are clustered or not
11. False (Because then you cannot use indexes to make joins, if there is a join)
12. False (because that tuple maybe does not join with anything in S)
13. False (the order does not affect the number of tuples returned. Only the time)
14. False (you cannot create 2 clustered indexes on the same table. All the rest, you can do them without any problem)
15. True
16. False (pay attention to the possible nulls)
17. (i) D, (ii) A, (iii) B
18. Once we do the index lookup we have to read N tuples (in both cases). The better to do clustered is the one for which in a page you can fit more tuples. And this is the Writer because its tuples are smaller.

19. The cost of the Scan is #pages of the table writer which is 4K tuples / # Writer tuples per page = 4K / roundDown(1000 / 50) = 4K / 20 = 200 pages.
The cost of using an Unclustered B+tree would be: Lookup + Reading cost of the N qualifying tuples = 4 + N. The scan is better when 200 < 4 + N or when N > 196.

20. The index is useless for this case because it is hash on 2 attributes so nested loop join is the only option so M+M*N or N+N*M which makes C the correct answer

21. If by ordered it means on the attributes a and b respectively for R and S then the cost of the join is simply a scan for each relation, meaning 10000/roundDown(750/50) + 500/roundDown(750/350) = 10000 / 15 + 500 / 2 = 2000/3 + 250 ~ 666 + 250 = 916. If ordered on other attributes then we need a nested loops join which is as if they were not sorted.

22.
C1 = 500/RoundDown(750/350) + 500 * (4+1)
T1 = 500 (as much as the number of tuples of S because a is key and s.b is FK to that key)
C2 = 100K/RoundDown(750/1000) = 100K/ 0.5 = 200K
T2 = 100K * 25% = 25K
C3 = T1/RoundDown(750/(350+50)) + T2/0.5 +
T1/RoundDown(750/(350+50)) + T1/ RoundDown(750/(350+50) * T2/0.5
The first is the left table materialization which means you have to save the intermediate results on the disk (T1/RoundDown(750/(350+50))). The same you have to do with the right table materialization (T2/0.5). Now you do the block nested loop JOIN: You read the left relation ((T1/RoundDown(750/(350+50))), and for every page of it (T1/RoundDown(750/(350+50)) pages in total) you read the right relation (T2/0.5)
T3 = T1 * T2 (because they have no common attribute so they are a cartesian product)
C4 = 0

T4 = T3