**Section A**

1.

* 1. Superkey: is a unique key and reducible. Key: is a superkey but with the property that removing of its attribute will cause it to not be a superkey i.e. uniquely identifiable.
  2. They model the real world relationship and interaction inside the database design. They are also important in normalisation as they help reconstructing tables after they get decomposed.

1. 1. Select dep.dName, count(work.eID)  
      FROM Dept dep  
      JOIN WorksIn work  
      ON (dep.dId = work.dId)  
      WHERE work.percentageTime = 1  
      GROUP BY dep.dName
   2. Solution := π(eName, Salary) (σ(dId = 101, percentageTime >= 50) Emp ⋈ WorksIn))
   3. UPDATE WorksIn work  
      SET work.percentageTime = work.percentageTime - 0.02
   4. NULL: set all the instance of it to NULL.   
      Cascade: cascade the delete to the foreign relation.  
      Default: Set it to a default value that is specified in the schema   
      Rejection: Reject any such attempts using Triggers.
   5. by adding to the declaration NOT NULL constraints to Dept.managerID.   
      ALTER TABLE Dept  
      MODIFY managerID int NOT NULL

2.

1. 1. NO...one can not infer if the postulated FDs in F hold based on the information provided. The reason is simple: you do not have all the data...you can never say for sure that you won't ever have a tuple that will break your FDs...maybe they hold for the current instance, but you can't say that they will hold for every instance
   2. BC -> A doesn't hold because   
      BC -> A:   
      {2, 3} -> 1   
      {2, 3} -> 4  
      Since 2 identical instances are linking to different results then the FD doesn’t hold.
2. 1. AB, AC,BC. Holds in the strongest form 3NF because ignore all those fds look at what you need ABC AB -> C, BC -> A, AC -> C that means every single information is about the key, and nothing but the key. basically they are all keys to each other.  
      Not in BCNF since there are multiple keys
   2. {B, D} (composite of both B and D)
   3. A good ER Model would capture the business requirements with the least amount of redundancy. ER is another way to model data, it is a conceptual way, while normalization is a logical modelling of the data. Normalization can make sure that ER is efficient by making sure it doesn’t have redundancy, a good ER won’t have any redundancy unless needed for optimisation

------------------------

**Section C**

3.

1. [Picture](https://drive.google.com/open?id=0BxSfLz74ghGpYzgzOThSSzcxZnc) (there is a total relationship constraint between department and works\_in which I forgot to draw)
2. [Picture 1-3](https://drive.google.com/open?id=0BxSfLz74ghGpVl94TXhEaWdLaXM), [Picture 4-6](https://drive.google.com/open?id=0BxSfLz74ghGpRkt3a0twY1BJOXM), Step 7 isn’t applicable as we don’t have any n-ary relationship, step 8 and 9 aren’t applicable
3. 1. surrogate key. Create a relation S’ that include all attributes of S, set the surrogate key as PK, and include it as FK in T and U.
   2. Yes it would work, but it will generate many NULL values.
   3. It would work on anything and everything

**n.b.** I think this type of question is the easiest it relies on memorising and no need for brain power, I might do this only in the exam

---------------------

4.

1. 1. After any increase to any employee's salary, make sure that the manager salary is still the highest or at least equivalent to the new highest employee’s salary
   2. 1- When using after UPDATE, need to reference old. no need to reference new as it is the current. so remove new. from all statement  
      2- everywhere old is mentioned should be referenced as :OLD  
      3- The update in budget doesn’t reflect that an employer might work partially in one department which can be alluded from percentageTime in the worksIn table  
      4- Select sum(E.Salary) again suffer from above problem, this might sum all salaries of all employees so some employees will be summed more than once in different department despite they are working partially in those department
2. 1. [Picture](https://drive.google.com/open?id=0BxSfLz74ghGpWUtHMzZ6dHBrTlE)
   2. T3 -> T2 -> T1 -> T3. those it is not a conflict serializable
   3. 1- Timeout: if lock hasn’t been granted after X amount of time then the DBMS assume deadlock happened and abort the transaction and restart it  
      2- Deadlock prevention: looking ahead to see if transaction would cause deadlock, using timestamps or variant of 2PL  
      3- Deadlock detection and recovery: DBMS allow deadlock to occur but recognise it and break it.