1(a).
We assume that employees at the Happy Supplies Parts Warehouse are in charge of many different customer orders. A single order from the Warehouse can contain multiple parts and a customer can place many orders. We also assume that CageCode is the storage location for a Part

1(b),1(c) 1(d)
Unnormalized relation:
Parts(CustomerNumber,CustomerName,CustomerType,OrderDate,OrderTime,Employee,(PartNumber, PartName,Type,CageCode,QuantityOrdered,OrderPrice))

To make it to 1NF we have to make sure that all the attributes are atomic and there are no repeating groups, and since one order can list multiple parts, all the part attributes repeat inside a single order record which violates the rule of 1NF.

1NF:
Parts(**CustomerNumber**,CustomerName,CustomerType,**OrderDate**,**OrderTime**,**Employee**,**PartNumber**,PartName,Type,CageCode,QuantityOrdered,OrderPrice)

The composite PK are:CustomerNumber,OrderDate,OrderTime,Employee,PartNumber

2NF:It must be in 1NF and there is only one field that makes up the Primary Key or All the non-PK fields are dependent on All the PK fields.

Based on 1NF, we find the partial dependencies of
Customer Number -> CustomerName,CustomerType
Part Number->PartName,Type,CageCode

and the full dependency:(CustomerNumber,OrderDate,OrderTime,Employee,PartNumber)-> QuantityOrdered,OrderPrice

Thus, the 2NF relations
Customer(**CustomerNumber**,CustomerName,CustomerType)
Order(**CustomerNumber,OrderDate,OrderTime,Employee**)
Part(**PartNumber**,PartName,Type,CageCode)
OrderLine(**CustomerNumber,OrderDate,OrderTime,Employee,PartNumber**,QuantityOrdered,OrderPrice)

3NF: so that we can not have transitive dependencies

different parts of the same type can be stored in different cages and there is no valid dependency of Type -> CageCode

3NF relation
Customer(**<u>CustomerNumber</u>**,CustomerName,CustomerType)
Order(**<u>CustomerNumber,OrderDate,OrderTime,Employee</u>**)
Part(**<u>PartNumber</u>**,PartName,Type,CageCode)
OrderLine(***<u>CustomerNumber,OrderDate,OrderTime,Employee,PartNumber</u>***,QuantityOrdered,OrderPrice)

2.(a)
We assume that one patient can have multiple appointments per day and a Therapist can only work at only one branch on any given date, but they can work at many branches overall. We also assume that each appointment assigns one patient to one therapist at one branch at a specific time and date

2(b)(c)(d)
Unnormalized relation:
Therapy(staffNo,therapistName,(patNo,patName,appointmentDate,appointmentTime,branchNo))

we can find that the repeating group is (patNo,patName,appointmentDate,appointmentTime,branchNo) multiple appointments under one therapist

1NF, to make all attributes atomic
AppointmentLine(**<u>staffNo</u>**,therapistName,**<u>appointmentDate</u>**,**<u>appointmentTime</u>**,**<u>patNo</u>**,patName,branchNo)
so the composite PK are, staffNo,appointmentDate,appointmentTime,patNo

In order to reach 2NF, we need to remove partial dependencies and based on 1NF table, we notice that
staffNo ->therapistName
patNo -> patName
staffNo, appointmentDate -> branchNo therapist works one branch per date

so the 2NF relation is

Therapist(**<u>staffNo</u>**,therapistName)
Patient(**<u>patNo</u>**,patName)
TherapistDailyBranch**<u>(staffNo,appointmentMentDate</u>**,BranchNo)
Appointment(**<u>staffNo,appointmentDate,appointmentTime,patNo)</u>**

And in order to reach 3NF we need to remove transitive dependencies

and we have the transitive dependencies of staffNo,appointmentDate -> branchNo, so we want to only keep it in TherapistDailyBranch as opposed appointment

so 3NF relation is
Therapist(**<u>staffNo</u>**, therapistName)
Patient(**<u>patNo</u>**,patName)
TherapistDailyBranch(***<u>staffNo</u>***,**<u>appointmentDate</u>**,branchNo)
(FK: staffNo -> Therapist)
Appointment***<u>(staffNo,appointmentDate</u>***,**<u>appointmentTime,</u>*****<u>patNo</u>***)
(FK: staffNo, appointmentDate -> TherapistDailyBranch, patNo ->patient)

the final 3NF relation
Therapist(**<u>staffNo</u>**, therapistName)
Patient(**<u>patNo</u>**,patName)
TherapistDailyBranch(***<u>staffNo</u>***,**<u>appointmentDate</u>**,branchNo)
Appointment***<u>(staffNo,appointmentDate</u>***,**<u>appointmentTime,</u>*****<u>patNo</u>***)

3(a)
We assume that eNo uniquely identifies a single Employee and each contract applies to only one event. And an event can have multiple contacts since it might have different service needs.

Based on the data table given we find that the functional dependencies are
eNo -> eName (the employee number is unique for each employee)
contractNo -> eventNo (the event number depends on the contract No)
eventNo ->eventLoc (location depends on the event number)
eNo, contractNo ->hours (the hours worked of an employee depends on the employee number and contract No),
Thus, the PK should be eNo, ContractNo

And since there is no repeating groups, the given table is already in 1NF

to reach to 2NF, we need to remove partial dependencies

and the composite primary key is eNo,contractNo, so the partial dependencies are
eNo ->eName
contractNo ->eventNo

2NF relation:

Employee(**eNo**,eName)
EventContract(**contractNo**,eventNo,eventLoc)
HourWorked(**eNo,ContractNo**,hours)

to reach 3NF, we need to remove transitive dependency
and we have transitive dependency of eventNo ->eventLoc

3NF relation

Employee(**eNo**,eName)
EventContract(**contractNo**,*eventNo*) (eventNo serve as FK to reference Event location)
HourWorked***(eNo,ContractNo***,hours) (eNo reference to employee and contractNo reference eventContract)
EventLocation(**eventNo**,eventLoc)