Question 1 Happy Supplies Parts Warehouse - Database

a) Assumptions: Each order is placed by exactly one customer. Each order is also handled by exactly one employee. However, each employee can serve multiple customers. Each order placed may contain multiple parts.

In order to better see where we stand, we will model a table from all the handwritten features to spot dependencies and relations. The data table will form our first point, being in 1NF.

Table1NF

Customer Name	customerNumber	customerType	date	time	employee	type	partNumber	name	cageCode	quantityOrdered	unitPrice

At this first glance, the 1NF form of the data seems very congested and the table is big with fields that have no relations, so we will start normalizing by creating entities first. Some candidate PKs I can identify is customer, part and order. To uniquely identify a customer, customer number may be used as a PK. For parts, part numbers can used to uniquely identify a part, therefore treated as the PK. However, order is a more trickier attribute to represent, in order to uniquely identify a single order, a combination is needed. The customer placing the order (pulled as a FK), date and time of the order and the number of the ordered part is enough for a unique representation.

Customer2NF

customerNumber (PK)	customerName	customerType

Part2NF

partNumber (PK)	name	type	cageCode

Order2NF

customerNumber (PK, FK)	date (PK)	partNumber (PK, FK)	time (PK)	quantityOrdered	unitPrice	employee

At the 2 NF stage, all the table's non-key dependencies also look fine, just for employee, which seems tricky because it is not so clear. Employee depends on the order event (CustomerNumber, Date, Time), not on PartNumber. To resolve this partial dependecny, we will need to break Order into two tables. After isolating employee into its own table, 3NF normalization will be established. To make this isolation into more meaningful tables, we will use two different order table, a header and a line to eliminate partial dependencies. This seperation eliminates redundancy and updates anomalities.

Customer3NF

customerNumber (PK)	customerName	customerType

Part3NF

Part Number (PK)	name	type	cageCode

OrderHeader3NF

customerNumber (PK, FK)	date (PK)	time (PK)	employee

OrderLine3NF

customerNumber (PK, FK)	date (PK, FK)	time (PK, FK)	partNumber (PK, FK)	quantityOrdered	unitPrice

Question 2 Panacea Mental Health Corporation Therapist Table

a) Assumptions: Therapists may work at different branches. But they only see branches at one specific branch a day. One patient has one specific appointment at one branch with one therapist at a set time and date. Appointments do not overlap. Patients may have multiple appointments in a given day and with multiple different therapists.

Table1NF

staffNo	therapistName	patNo	patName	appointmentTime	appointmentDate	branchNo

As each column has atomic values and there are no repeating groups, the given table is already in 1NF (we just need to break appointmentDate and appointmentTime to form two non-repeating entries). We will now look at candidate keys and their dependencies to form a 2NF table. We can easily see that a staff and a patient can be uniquely illustrated with their IDs, but how to represent the appointment is not as intuitive. As an appointment depends on many keys, we will need to use the PKs from the first 2 tables and appointmentDate and appointmentTime.

Staff2v3NF

staffNo (PK)	therapistName

Patient2v3NF

patNo (PK)	patName

Appointment2v3NF

staffNo (PK, FK)	patNo (PK, FK)	appointmentDate (PK)	appointmentTime (PK)	branchNo

As staffNo can identify a therapist uniquely we will use it as a PK, and the same principle will apply to PatNo. However for appointments we will need a different approach to deciding on a primary key, because appointment requires a composite key (staffNo, patNo, appointmentDate, appointmentTime) to uniquely identify each appointment.

For the last transition to 3NF, we will check non-key to non-key dependencies known as transitive dependencies. In the first two tables the therapistName and patName directly depend on their respective PKs. Branch no also depends on the appointment, the full composite PK, not just staffNo or any single PK alone. Hence, it can be proved that we do not need to change to schema to have a 3NF table. So we leave the table as it is, no further action is needed.

Question 3 Maid Better Temp Agency Supplies - Database

a) Assumptions: Each employee has a unique eNo. An employee can have multiple contracts and events. Each contract is made based on one event. Employee hours are based on contract, not directly event level. The relationship between employee and contracts in many to many.

The given table is already at 1NF, values at every collumn are atomic. There are no repeating groups inside a row.

Table1NF

eNo	contractNo	hours	eName	eventNo	eventLoc

To change the table to 2NF, we will examine possible candidate PKs. For employees, eNo and for contracts contractNo seem straightforward. Also for events, eventNo uniquely identifies an event.

Employee2v3NF

eNo (PK)	eName

Event2v3NF

eventNo (PK)	eventLoc

Contract2v3NF

contractNo (PK)	eventNo (FK)

Hours2NF

eNo (PK, FK)	contractNo (PK, FK)	hours

In 2NF form, all non-key attributes depend on the whole key, and nothing but the key. contractNo determines eventNo, but since event has already been seperated, so no transitive dependency remains. hours also depend on the full composite key. Therefore, the schema will not need to be changed as it is in 3NF, no further action is needed.