Problem 1

Assumptions:

A customer can place multiple orders over time, but each order is associated with only one customer. Each order is handled by one employee at a given date and time. Customers are identified by a Customer Number, which is unique for each customer.

The original table is in Unnormalized Form because it contains redundant data and repeating groups. The table contains multiple parts per order, causing customer, order, and part details to be repeated across multiple rows. To address this issue, I will introduce Order ID as a unique identifier for each order. This ensures that each row represents a single atomic fact, eliminating repeating groups and storing atomic values.

Tables in UNF:

order
orderID
customerNumber
customerName
customerType
date
time
employee
partNumber
partName
type
cageCode
quantityOrdered
unitPrice

While the data is now atomic, the table still contains partial dependencies, meaning some columns depend only on a part of a composite primary key rather than the whole key. Customer Name and Customer Type depend only on Customer Number, not on the composite key (Order ID, Part Number). Part Name, Part Type, and Cage Code depend only on Part Number, not on the composite key. Order-level data (such as Order Date and Employee) is mixed with line-item data (such as Quantity Ordered). To remove these partial dependencies, I will create a Customers table to store customer-related information separately, create a Parts table to store

part-related information, create an Orders table that holds only order-level details and introduce an OrderDetails table to establish a many-to-many relationship between orders and parts.

Tables in 1NF:

order
orderID PK
customerNumber FK
orderDate
orderTime
employee

customer
customerNumber PK
customerName
customerType

order_detail
orderID PK
partNumber PK
quantityOrdered
unitPrice

part
partNumber PK
partName
partType
cageCode

The database is now in 1NF, but some attributes are still dependent on part of the primary key rather than the whole key: Employee is dependent on Order ID, meaning it should be stored separately to avoid duplication. Part Type and Cage Code are not directly dependent on Part Number, meaning they should be stored as separate entities. To eliminate these partial dependencies, I will create an Employees table and replace Employee with Employee ID in the Orders table, create a PartTypes table to store different part categories separately and create a StorageLocations table to store Cage Code separately.

Tables in 2NF:

order

orderID PK

customerNumber FK

orderDate

orderTime

employeeID FK

customer

customerNumber PK

customerName

customerType

order_detail

orderID PK

partNumber PK

quantityOrdered

unitPrice

part

partNumber PK

partName

partType FK

cageCode FK

employee

employeeID PK

employeeName

The database is in 2NF, but there are still transitive dependencies. Employee Name is dependent on Employee ID, not directly on Order ID. Part Type and Cage Code still exist in the Parts table but should reference separate entities. To resolve these transitive dependencies, I

will ensure Employee Name is fully managed in the Employees table and referenced using Employee ID and ensure Part Type and Cage Code reference their respective tables instead of being stored redundantly in the Parts table.

Tables in 3NF:

order (stores general order details)
orderID PK
customerNumber FK
orderDate
orderTime
employeeID FK

customer (stores customer information)
customerNumber PK
customerName
customerType

order_detail (stores details of each part in an order)
orderID PK
partNumber PK
quantityOrdered
unitPrice

part (stores part information)
partNumber PK
partName
partTypeID FK
cageCodeID FK

employee (stores employee details)
employeeID PK
employeeName

part_type (stores part category information)
partTypeID PK

partTypeName

storage_location (stores storage cage details)
cageCodeID PK
cageCodeName

Problem 2

Assumptions:

Therapists may work at multiple branches, but they only see patients at one branch per appointment. Patients can have multiple appointments per day, potentially with different therapists. Each appointment occurs at a specific date and time and is uniquely determined by the combination of staffNo, patNo, appointmentDate, and appointmentTime.

The table is in Unnormalized Form because therapist and patient names are repeated multiple times, creating redundancy, appointment contain multiple values and there is not primary key. To achieve 1NF, I will introduce a composite primary key to uniquely identify each appointment and ensure each row represents a single atomic fact.

Tables in UNF:

appointment
staffNo
therapistName
patNo
patName
appointmentDate
appointmentTime
branchNo

Therapist Name depends only on staffNo, not on the entire composite key. Patient Name depends only on patNo, not on the entire composite key. Branch Number (branchNo) depends on staffNo, not on the full composite key. This leads to partial dependencies, meaning we need to break the table into separate entities. To remove partial dependencies, I will move therapist details into a Therapists table with staffNo as the primary key, move patient details into a

Patients table with patNo as the primary key and create an Appointments table with staffNo, patNo, appointmentDate, and appointmentTime as the primary key.

Tables in 1NF:

appointment
staffNo PK
patNo PK
appointmentDate PK
appointmentTime PK
branchNo FK

therapist
staffNo PK
therapistName

patient
patNo PK
patName

branchNo is dependent on staffNo, not the entire primary key of Appointments.
This means branchNo should be stored in a separate table related to staffNo. To eliminate this partial dependency, I will create a TherapistBranches table where each therapist is assigned to a branch on a given date and modify Appointments so that branchNo is removed, as it can be derived from TherapistBranches.

Tables in 2NF:

appointment
staffNo PK
patNo PK
appointmentDate PK
appointmentTime PK

therapist
staffNo PK
therapistName

patient
patNo PK
patName

therapist_branch
staffNo PK
branchNo PK
assignedDate

Therapist Name (therapistName) is dependent on staffNo, but staffNo is not the primary key in Appointments. Patient Name (patName) is dependent on patNo, but patNo is not the primary key in Appointments. Transitive dependencies exist, meaning data is still not fully normalized. To remove transitive dependencies, I will ensure therapist details are only stored in Therapist and referenced by staffNo and ensure patient details are only stored in Patient and referenced by patNo.

Tables in 3NF:

appointment (stores all therapist-patient interactions at a specific date/time)
staffNo PK
patNo PK
appointmentDate PK
appointmentTime PK

therapist (stores therapist details)
staffNo PK
therapistName

patient (stores patient details)
patNo PK
patName

therapist_branch (tracks where each therapist is assigned)
staffNo PK
branchNo PK
assignedDate

branch (stores branch details)
branchNo PK
branchLocation

Problem 3

Assumptions:

Each employee is uniquely identified and may work on multiple contracts. Each contract is associated with a single event. Different contracts may exist for the same event, depending on different service needs.

Table is unnormalized because employee names and event locations are repeated and the table does not have a clear primary key that uniquely identifies each record. Therefore, I will introduce employeeContractID as a unique identifier for each row to achieve 1NF.

Tables in UNF:

employee_contract
eNo
eName
contractNo
hours
eventNo
eventLoc

There is partial dependency since eName depends only on eNo, not on contractNo and eventLoc depends only on eventNo, not on contractNo. To remove partial dependencies, I will create an Employees table to store employee details separately, create an Events table to store event details separately, create a Contracts table to track contracts linked to events and keep EmployeeContracts as an association table linking employees to contracts.

Tables in 1NF:

employee_contract
employeeContractID PK
eNo
eName
contractNo

hours
eventNo
eventLoc

eventNo is dependent on contractNo, not the full composite key of EmployeeContracts. This creates a transitive dependency that needs to be removed. To eliminate this transitive dependency, I will ensure eventNo is stored only in Contracts, not in EmployeeContracts and keep Events as a separate table that Contracts reference.

Tables in 2NF:

employee_contract
employeeContractID PK
eNo FK
contractNo FK
hours

employee
eNo PK
eName

contract
contractNo PK
eventNo FK

event
eventNo PK
eventLoc

Contract table shows a transitive dependency where eventNo leads to eventLoc, which should be dependent only on its primary key. To move to 3NF, I will ensure there is no transitive dependencies and remove any dependencies where non-key attributes depend on other non-key attributes.

Tables in 3NF:

employee_assignment (track assignments of employee to contracts and their hours)
employeeContractID PK

eNo FK
contractNo FK
hours

employee (stores employee details)
eNo PK
eName

Contract (tracks contracts and their associated events)
contractNo PK
eventNo FK

event (stores event details including locations)
eventNo PK
eventLoc