Assignment 2 Solution

SolutionQ1:

Let's create a similar design for a **PATIENT** entity type that tracks a patient's history of **medical treatments**. Each entry should track the hospital where the treatment was given, the start and end dates of the treatment, any procedures performed (if any), and a list of medications prescribed (if any). Each procedure includes the procedure name and the date it was performed. Each medication entry contains the medication name, dosage, and duration.

Scenario: Medical Treatment History for a PATIENT

We want to design an attribute for a **PATIENT** entity type that tracks their history of medical treatments. Each treatment will contain:

- The **hospital name** where the treatment was given.
- The **start and end dates** of the treatment.
- A list of **procedures** performed during the treatment (if any).
- A list of **medications** prescribed during the treatment (if any).

Each **procedure entry** will contain:

- **Procedure name**
- **Date** the procedure was performed.

Each **medication entry** will contain:

- **Medication name**
- **Dosage**
- **Duration** (number of days).

a)

- **Treatment History (Derived attribute)**:
    - **Hospital Name** (simple attribute)
    - **Start Date** (simple attribute)
    - **End Date** (simple attribute)
- **Procedures** (composite attribute):

    - **Procedure Name** (simple attribute)
    - **Procedure Date** (simple attribute)
- **Medications** (composite attribute):
    - **Medication Name** (simple attribute)
    - **Dosage** (simple attribute)
    - **Duration** (simple attribute)

This structure reflects the fact that a patient can have multiple treatments, and for each treatment, there can be multiple procedures and medications.

b) Alternative Design Using Entity and Relationship Types

An alternative design that uses **entity types** and **relationship types** (instead of composite/multivalued attributes) might look like this:

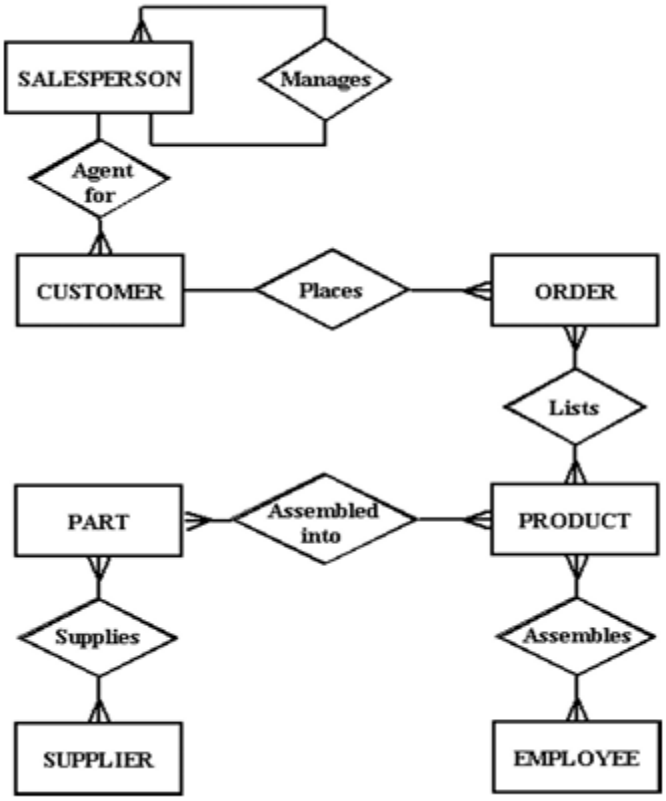
Entities:

1. **PATIENT** (Primary key: **Patient_ID**)
    - Attributes: Patient_Name, Date_Of_Birth, etc.
2. **TREATMENT** (Primary key: **Treatment_ID**, weak entity related to PATIENT)
    - Attributes: Start_Date, End_Date, Hospital_Name
    - Relationship with PATIENT: A patient has many treatments, a treatment belongs to one patient.
3. **PROCEDURE** (Primary key: **Procedure_ID**, weak entity related to TREATMENT)
    - Attributes: Procedure_Name, Procedure_Date
    - Relationship with TREATMENT: A treatment can have many procedures, and a procedure belongs to one treatment.
4. **MEDICATION** (Primary key: **Medication_ID**, weak entity related to TREATMENT)
    - Attributes: Medication_Name, Dosage, Duration
    - Relationship with TREATMENT: A treatment can have many medications, and a medication belongs to one treatment.

Relationships:

- **Undergoes** (relationship between **PATIENT** and **TREATMENT**)
    - A patient undergoes many treatments; a treatment is related to one patient.
- **Includes Procedure** (relationship between **TREATMENT** and **PROCEDURE**)
    - A treatment includes many procedures; a procedure is part of one treatment.
- **Prescribes Medication** (relationship between **TREATMENT** and **MEDICATION**)
    - A treatment prescribes many medications; a medication is prescribed in one treatment.

Solution Q2:



Solution Q3:

Entities

- **Customer:** Represents an individual or company placing an order.
- **Order:** Represents a single purchase request.
- **Product:** Represents an item available for purchase.

- **Payment:** Represents a method used to pay for an order.

Attributes

- **Customer:**
    - Name
    - Company Name
    - Address(composite Attribute)
    - Daytime Phone
    - PC owned
    - Printer owned
- **Order:**
    - Order Number (implied)
    - Order Date (implied)
    - Shipping Address (optional)
    - Payment Method
    - Card Number (if applicable)
    - Expiry Date (if applicable)
    - Signature
- **Product:**
    - SKU (Stock Keeping Unit)
    - Description
    - Price
    - Quantity
    - Total Price
    - Shipping
    - Total Shipping
- **Payment:**

- Payment Method
- Card Number (if applicable)
- Expiry Date (if applicable)
- Signature

Relationships

- **Places Order:** A Customer can place multiple Orders. (1:M)
- **Contains:** An Order contains multiple Products. (1:M)
- **Uses:** An Order uses a single Payment method. (1:1)

Solution Q4 (Part A):

a. How can we insert the supplier's information (e.g., "Budget Rentals") into the system without assigning any cars? In the current design, is it possible to insert supplier details without car data? What could be the issue with this?

Expected Answer
Since the current table structure requires each row to represent a car along with its supplier, the system cannot insert the supplier's information independently without assigning at least one car. This leads to an **insert anomaly**, as the system forces the insertion of a dummy car or incomplete car data just to register the supplier. This causes **data integrity** issues because the car may not yet exist.

b. How would the current table structure handle this update? What happens if you update the phone number for "Hertz" in only some rows but not all? How does this lead to an update anomaly, and what are the implications for **data consistency**?

Expected Answer
In the current table design, "Hertz" appears multiple times (e.g., in rows for C101, C103, and C105). To update the phone number for "Hertz," we would need to manually update each occurrence. If any row is missed during this update, there will be inconsistent phone numbers for the same supplier across different rows, resulting in an **update anomaly**. This inconsistency compromises **data integrity**, as queries may retrieve different contact information depending on which row is accessed.

c. If the last car rented from "Avis" is deleted from the system, what happens to the supplier information for "Avis"? How does this lead to a delete anomaly, and what are the potential issues with losing supplier data?

Expected Answer

When the last car rented from "Avis" (C104) is deleted from the table, the supplier information for "Avis" is also removed. This is a **delete anomaly** because although "Avis" might still be an active supplier, their contact details are lost just because there are no cars currently associated with them. This results in **data loss**, and when new cars are added from "Avis" in the future, the company will have to re-enter the supplier's details, leading to unnecessary redundancy and possible mistakes.

Solution Q4 Part B

The table is already in **1NF** and **2NF**:

- All values are atomic (1NF).
- No partial dependencies because CarID is the primary key (2NF).

Functional Dependencies

1. CarID → Model, PricePerDay, CategoryID, SupplierID
2. CategoryID → CategoryName
3. SupplierID → SupplierName, Email, Phone

Moving to 3NF:

To bring the table into 3NF, we eliminate transitive dependencies.

Car(CarID, Model, PricePerDay, CategoryID, SupplierID)

Category(CategoryID, CategoryName)

Supplier(SupplierID, SupplierName, Email, Phone)

Final Schema Design:

Car(<u>CarID</u>, Model, PricePerDay, *<u>CategoryID</u>*, *<u>SupplierID</u>*)

Category(<u>CategoryID</u>, CategoryName)

Supplier(<u>SupplierID</u>, SupplierName, Email, Phone)

underlined are primary key and italic are foreign keys

Justification:

- Insert Anomaly Resolved: Suppliers can be added independently in the Supplier table.
- Update Anomaly Resolved: Updating supplier information now happens in one place (Supplier table).
- Delete Anomaly Resolved: Deleting a car will not delete the supplier information.

Solution Q5

(1) Identify the functional dependencies represented by the data shown in the form

Patient No → Full Name

Ward No → Ward Name

Drug No → Name, Description, Dosage, Method of Admin

Patient No, Drug No, Start Date → Units per Day, Finish date

If bedNo was a unique number for the entire hospital, then could say that bedNo → wardNo. However, from further examination of the requirements specification, we can observe that bedNo is to do with the allocation of patients on the waiting list to beds..

(2) Describe and illustrate the process of normalizing the data shown in Figure to first (1NF), second (2NF), third (3NF),

First Normal Form

PatientMedication(PatientNo, DrugNo, StartDate, FullName, WardNo, WardName, BedNo, DrugName, DrugDescription, Dosage, MethodOfAdmin, UnitsPerDay, FinishDate)

Second Normal Form

PatientMedication(PatientNo, DrugNo, StartDate, WardNo, WardName, BedNo, UnitsPerDay, FinishDate)

Drug (DrugNo, DrugName, DrugDescription, Dosage, MethodOfAdmin)

Patient (PatientNo, FullName)

Third Normal Form

PatientMedication (PatientNo, DrugNo, StartDate, WardNo, BedNo, UnitsPerDay, FinishDate)

Drug (DrugNo, DrugName, DrugDescription, Dosage, MethodOfAdmin)

Patient (PatientNo, FullName)

Ward (WardNo, WardName)

(3) 1. **PatientMedication**

Primary Key: (PatientNo, DrugNo, StartDate)

Foreign Keys: PatientNo, DrugNo, WardNo

2. **Drug**

Primary Key: DrugNo

3. **Patient**

Primary Key: PatientNo

4. **Ward**

Primary Key: WardNo

There are no alternate keys in this design.

Solution Q6:

Given Relation:

R(Reservation_ID,Customer_ID,Room_Number,Stay_Date,Room_Type,Service_Code,Service_Charge)

Primary Key: **Reservation_ID** (as it uniquely identifies each reservation).

Identifying Dependencies:

Room_Type depends on Room_Number:

Room types are determined by room numbers, not by Reservation_ID.

Service_Charge depends on Service_Code:

Service charges are fixed per service, independent of Reservation_ID.

Already in 2NF

The original relation is already in 2NF We decomposed the relation into smaller relations to eliminate the transitive dependencies and achieved **3NF**. Therefore, no further normalization is necessary.

3NF

Reservation Detail (**Reservation_ID**, Customer_ID, *Room_Number* , *Service_Code*, stay date)

Room(**Room_Number** , room type)

Service(**Service_Code**,Service_Charge)

Note: bold underlined are primary keys and italic underlined are foreign keys