The questions and solutions are provided by Chatgpt. If you find any mistake, let everyone else know.

Help yourself by drawing the Table and Diagram from provided solution.

Question 1: Library Management System

A library management system is developed in your university that stores various types of information for efficient data processing through software. To be a member of the library, **students** must provide the following details: `Name`, `ContactNumber`, and `Email`. After registering, each student will be assigned a `MemberID`. The system keeps track of each **member**, including the `MembershipDate`.

Being a member, students can borrow multiple **books**. So, to keep track, the system maintains details about each **book** including `BookID` (unique identifier), `Title`, `ISBN`, `Genre`, and `PublicationYear`. Each book can have multiple **authors**. The system stores details about each **author** including `AuthorID`, `Name`, and `Nationality`.

The system also records when a **member borrows** a book, including `BorrowID`, `BorrowDate`, `ReturnDate`, and `DueDuration`. Each **book** can be borrowed by multiple members over time, and each member can borrow multiple books.

Tasks:

1. Design an ER diagram to represent the above scenario, capturing the entities, attributes, keys, relationships, and cardinality properties.
2. Convert the ERD into a relational database schema.

Solution

1. **Entities and Attributes:**
    - **Student**:
        - `MemberID` (Primary Key), `Name`, `ContactNumber`, `Email`, `MembershipDate`.
    - **Book**:
        - `BookID` (Primary Key), `Title`, `ISBN`, `Genre`, `PublicationYear`.
    - **Author**:
        - `AuthorID` (Primary Key), `Name`, `Nationality`.

    - **Borrow**:
        - `BorrowID` (Primary Key), `BorrowDate`, `ReturnDate`, `DueDuration`.
2. **Relationships and Cardinality:**
    - **BorrowedBy**: A **Student** can borrow multiple **Books** (Many-to-Many relationship through the `Borrow` table).
    - **WrittenBy**: A **Book** can have multiple **Authors** (Many-to-Many relationship).
3. **ER Diagram Overview:**
    - **Entities**: Student, Book, Author, Borrow.
    - **Relationships**: BorrowedBy, WrittenBy.
4. **Relational Schema:**
    - **Student Table**:
        - `MemberID` (Primary Key)
        - `Name`
        - `ContactNumber`
        - `Email`
        - `MembershipDate`
    - **Book Table**:
        - `BookID` (Primary Key)
        - `Title`
        - `ISBN`
        - `Genre`
        - `PublicationYear`
    - **Author Table**:
        - `AuthorID` (Primary Key)
        - `Name`
        - `Nationality`
    - **Borrow Table**:
        - `BorrowID` (Primary Key)
        - `MemberID` (Foreign Key references `Student.MemberID`)
        - `BookID` (Foreign Key references `Book.BookID`)
        - `BorrowDate`
        - `ReturnDate`
        - `DueDuration`
    - **WrittenBy Table**:
        - `AuthorID` (Foreign Key references `Author.AuthorID`)
        - `BookID` (Foreign Key references `Book.BookID`)

Question 2: Hospital Management System

A hospital wants to implement a management system to keep track of its **patients**, **doctors**, and **appointments**. For each **patient**, the hospital records details such as `PatientID` (unique identifier), `Name`, `Age`, `Gender`, `ContactNumber`, and `Address`. Each patient may have multiple medical conditions but only one assigned **doctor** for primary care.

For each **doctor**, the hospital maintains `DoctorID` (unique identifier), `Name`, `Specialization`, and `Experience`. A doctor can treat multiple **patients**.

Each **patient** can book multiple **appointments**. For each **appointment**, the hospital stores `AppointmentID`, `Date`, `Time`, `Reason`, and `DoctorID` (which doctor the patient booked the appointment with). A **doctor** can have multiple **appointments** scheduled with different patients.

Tasks:

1. Design an ER diagram to represent the above scenario, capturing the entities, attributes, keys, relationships, and cardinality properties.
2. Convert the ERD into a relational database schema.

Solution

1. **Entities and Attributes:**
    - **Patient**:
        - `PatientID` (Primary Key), `Name`, `Age`, `Gender`, `ContactNumber`, `Address`.
    - **Doctor**:
        - `DoctorID` (Primary Key), `Name`, `Specialization`, `Experience`.
    - **Appointment**:
        - `AppointmentID` (Primary Key), `Date`, `Time`, `Reason`, `DoctorID` (Foreign Key references `Doctor.DoctorID`).
2. **Relationships and Cardinality:**
    - **TreatedBy**: A **Patient** is treated by one primary **Doctor** (One-to-Many relationship).
    - **ScheduledFor**: A **Patient** can have multiple **Appointments** with different **Doctors** (One-to-Many).
3. **ER Diagram Overview:**
    - **Entities**: Patient, Doctor, Appointment.
    - **Relationships**: TreatedBy, ScheduledFor.
4. **Relational Schema:**
    - **Patient Table**:
        - `PatientID` (Primary Key)
        - `Name`
        - `Age`
        - `Gender`

        - `ContactNumber`
        - `Address`
        - `DoctorID` (Foreign Key references `Doctor.DoctorID` for primary care doctor)
    - **Doctor Table**:
        - `DoctorID` (Primary Key)
        - `Name`
        - `Specialization`
        - `Experience`
    - **Appointment Table**:
        - `AppointmentID` (Primary Key)
        - `PatientID` (Foreign Key references `Patient.PatientID`)
        - `DoctorID` (Foreign Key references `Doctor.DoctorID`)
        - `Date`
        - `Time`
        - `Reason`

Question 3: E-commerce System

An online shopping system is designed to manage **customers**, **products**, and **orders**. For each **customer**, the system records details such as `CustomerID` (unique identifier), `Name`, `Email`, `PhoneNumber`, and `ShippingAddress`. A customer can place multiple **orders**.

Each **order** is identified by `OrderID` and has associated details like `OrderDate`, `TotalAmount`, and `Status` (Pending, Shipped, Delivered, or Cancelled). Each order is placed by one **customer** but can contain multiple **products**.

For each **product**, the system tracks `ProductID` (unique identifier), `ProductName`, `Category`, `Price`, and `StockQuantity`. A product can appear in multiple **orders**. Additionally, a customer can leave **reviews** for products they have purchased, and for each review, the system stores `ReviewID`, `Rating`, `Comments`, and `ReviewDate`. Each review is linked to one customer and one product.

Tasks:

1. Design an ER diagram to represent the above scenario, capturing the entities, attributes, keys, relationships, and cardinality properties.
2. Convert the ERD into a relational database schema.

Solution

1. **Entities and Attributes:**

- **Customer**:
  - `CustomerID` (Primary Key), `Name`, `Email`, `PhoneNumber`, `ShippingAddress`.
- **Order**:
  - `OrderID` (Primary Key), `OrderDate`, `TotalAmount`, `Status`, `CustomerID` (Foreign Key references `Customer.CustomerID`).
- **Product**:
  - `ProductID` (Primary Key), `ProductName`, `Category`, `Price`, `StockQuantity`.
- **Review**:
  - `ReviewID` (Primary Key), `Rating`, `Comments`, `ReviewDate`, `CustomerID` (Foreign Key references `Customer.CustomerID`), `ProductID` (Foreign Key references `Product.ProductID`).

2. **Relationships and Cardinality:**
   - **Places**: A **Customer** can place multiple **Orders** (One-to-Many relationship).
   - **Contains**: An **Order** can contain multiple **Products** (Many-to-Many relationship through an intermediary table `OrderDetails`), with an additional attribute `Quantity` indicating the number of units.
   - **Reviews**: A **Customer** can review multiple **Products** (Many-to-Many relationship through the `Review` entity).
3. **ER Diagram Overview:**
   - **Entities**: Customer, Order, Product, Review.
   - **Relationships**: Places, Contains, Reviews.
4. **Relational Schema:**
   - **Customer Table**:
     - `CustomerID` (Primary Key)
     - `Name`
     - `Email`
     - `PhoneNumber`
     - `ShippingAddress`
   - **Order Table**:
     - `OrderID` (Primary Key)
     - `OrderDate`
     - `TotalAmount`
     - `Status`
     - `CustomerID` (Foreign Key references `Customer.CustomerID`)
   - **Product Table**:
     - `ProductID` (Primary Key)
     - `ProductName`
     - `Category`

        - `Price`
        - `StockQuantity`
    - **OrderDetails Table**:
        - `OrderID` (Foreign Key references `Order.OrderID`)
        - `ProductID` (Foreign Key references `Product.ProductID`)
        - `Quantity`
    - **Review Table**:
        - `ReviewID` (Primary Key)
        - `Rating`
        - `Comments`
        - `ReviewDate`
        - `CustomerID` (Foreign Key references `Customer.CustomerID`)
        - `ProductID` (Foreign Key references `Product.ProductID`)

Question 4: Employee Management System

A company wants to implement an employee management system to keep track of its **employees**, **departments**, and **projects**. For each **employee**, the system records details such as `EmployeeID` (unique identifier), `Name`, `DOB`, `Address`, `PhoneNumber`, and `Salary`. Each employee is assigned to a **department**. Each **department** has `DepartmentID`, `DepartmentName`, and `ManagerID` (the manager's EmployeeID).

Each employee can be assigned to multiple **projects**. For each **project**, the system stores `ProjectID`, `ProjectName`, `StartDate`, `EndDate`, and `Budget`. An employee can be involved in multiple projects, and a project can have multiple employees assigned.

The company also tracks **dependents** for each employee. For each **dependent**, the system maintains `DependentID`, `Name`, `Relationship` (e.g., spouse, child), and `DOB`. An employee can have multiple dependents.

Tasks:

1. Design an ER diagram to represent the above scenario, capturing the entities, attributes, keys, relationships, and cardinality properties.
2. Convert the ERD into a relational database schema.

Solution

1. **Entities and Attributes:**
    - **Employee**:

        - `EmployeeID` (Primary Key), `Name`, `DOB`, `Address`, `PhoneNumber`, `Salary`.
    - **Department**:
        - `DepartmentID` (Primary Key), `DepartmentName`, `ManagerID` (Foreign Key references `Employee.EmployeeID`).
    - **Project**:
        - `ProjectID` (Primary Key), `ProjectName`, `StartDate`, `EndDate`, `Budget`.
    - **Dependent**:
        - `DependentID` (Primary Key), `Name`, `Relationship`, `DOB`, `EmployeeID` (Foreign Key references `Employee.EmployeeID`).
2. **Relationships and Cardinality:**
    - **WorksIn**: An **Employee** is assigned to one **Department** (Many-to-One relationship).
    - **AssignedTo**: An **Employee** can be assigned to multiple **Projects** (Many-to-Many relationship).
    - **HasDependent**: An **Employee** can have multiple **Dependents** (One-to-Many relationship).
3. **ER Diagram Overview:**
    - **Entities**: Employee, Department, Project, Dependent.
    - **Relationships**: WorksIn, AssignedTo, HasDependent.
4. **Relational Schema:**
    - **Employee Table**:
        - `EmployeeID` (Primary Key)
        - `Name`
        - `DOB`
        - `Address`
        - `PhoneNumber`
        - `Salary`
        - `DepartmentID` (Foreign Key references `Department.DepartmentID`)
    - **Department Table**:
        - `DepartmentID` (Primary Key)
        - `DepartmentName`
        - `ManagerID` (Foreign Key references `Employee.EmployeeID`)
    - **Project Table**:
        - `ProjectID` (Primary Key)
        - `ProjectName`
        - `StartDate`
        - `EndDate`
        - `Budget`
    - **Dependent Table**:

        - `DependentID` (Primary Key)
        - `Name`
        - `Relationship`
        - `DOB`
        - `EmployeeID` (Foreign Key references `Employee.EmployeeID`)
    - **AssignedTo Table**:
        - `EmployeeID` (Foreign Key references `Employee.EmployeeID`)
        - `ProjectID` (Foreign Key references `Project.ProjectID`)

Question 5: University Event Management System

A university event management system is required to track details about different events organized on campus. For each **event**, the university stores information like `EventID` (unique identifier), `EventName`, `Location`, `Date`, and `Duration`. Each event is organized by one or more **departments**. The system needs to record the `DepartmentID` (unique identifier), `DepartmentName`, and the `HeadOfDepartment`. A department can organize multiple events over time.

Students can **participate** in multiple events. For each **student**, the system records details like `StudentID`, `Name`, `YearOfStudy`, and `Major`. The system also maintains the participation status of each student in different events (whether a student has registered, attended, or won an award). Each student can participate in multiple events, and each event can have multiple student participants.

In addition, each event is **sponsored** by one or more **companies**. For each **company**, the system tracks the `CompanyID`, `CompanyName`, and `SponsorshipAmount`. Each company can sponsor multiple events.

Tasks:

1. Design an ER diagram for the above scenario, capturing the entities, attributes, keys, relationships, and cardinality properties.
2. Convert the ER diagram into a relational schema.

Solution Overview:

1. **Entities and Attributes:**
    - **Event**: `EventID` (Primary Key), `EventName`, `Location`, `Date`, `Duration`.
    - **Department**: `DepartmentID` (Primary Key), `DepartmentName`, `HeadOfDepartment`.

    - **Student**: `StudentID` (Primary Key), `Name`, `YearOfStudy`, `Major`.
    - **Company**: `CompanyID` (Primary Key), `CompanyName`, `SponsorshipAmount`.
2. **Relationships:**
    - **Organizes**: One-to-Many relationship between `Department` and `Event`.
    - **Participates**: Many-to-Many relationship between `Student` and `Event` with a `Status` attribute indicating the participation status.
    - **Sponsors**: Many-to-Many relationship between `Company` and `Event` with the `SponsorshipAmount` attribute.
3. **Relational Schema:**
    - **Event Table**:
        - `EventID` (Primary Key)
        - `EventName`
        - `Location`
        - `Date`
        - `Duration`
    - **Department Table**:
        - `DepartmentID` (Primary Key)
        - `DepartmentName`
        - `HeadOfDepartment`
    - **Student Table**:
        - `StudentID` (Primary Key)
        - `Name`
        - `YearOfStudy`
        - `Major`
    - **Company Table**:
        - `CompanyID` (Primary Key)
        - `CompanyName`
        - `SponsorshipAmount`
    - **Organizes Table**:
        - `DepartmentID` (Foreign Key references `Department.DepartmentID`)
        - `EventID` (Foreign Key references `Event.EventID`)
    - **Participates Table**:
        - `StudentID` (Foreign Key references `Student.StudentID`)
        - `EventID` (Foreign Key references `Event.EventID`)
        - `Status` (Indicates if the student has registered, attended, or won an award)
    - **Sponsors Table**:
        - `CompanyID` (Foreign Key references `Company.CompanyID`)
        - `EventID` (Foreign Key references `Event.EventID`)
        - `SponsorshipAmount`

Question 6: Online Bookstore System

An online bookstore system is developed to manage book sales and inventory. For each **book**, the system keeps track of the `BookID` (unique identifier), `Title`, `Genre`, `PublicationYear`, and `Price`. Each book can have multiple **authors**, and the system records the `AuthorID`, `AuthorName`, and `Country` for each author. A book can have multiple authors, and each author can write multiple books.

Customers can **order** multiple books. The system stores customer details such as `CustomerID`, `Name`, `Email`, and `Phone`. Each customer can place multiple **orders**, and each order can include multiple books. For each **order**, the system keeps the `OrderID`, `OrderDate`, `TotalAmount`, and a reference to the `CustomerID`. The system also stores details about which books are in each order.

The system further tracks **reviews** for books. A **customer** can write a review for a **book**. For each review, the system stores a `ReviewID`, `Rating`, `Comments`, and the date when the review was submitted. Each customer can write multiple reviews for different books, but each review is specific to one book.

Tasks:

1. Design an ER diagram to represent the above scenario, capturing entities, attributes, keys, relationships, and cardinality properties.
2. Convert the ER diagram into a relational database schema.

Solution Overview:

1. **Entities and Attributes:**
    - **Book**: `BookID` (Primary Key), `Title`, `Genre`, `PublicationYear`, `Price`.
    - **Author**: `AuthorID` (Primary Key), `AuthorName`, `Country`.
    - **Customer**: `CustomerID` (Primary Key), `Name`, `Email`, `Phone`.
    - **Order**: `OrderID` (Primary Key), `OrderDate`, `TotalAmount`, `CustomerID`.
    - **Review**: `ReviewID` (Primary Key), `Rating`, `Comments`, `Date`, `CustomerID`, `BookID`.
2. **Relationships:**
    - **Writes**: Many-to-Many relationship between `Author` and `Book`.
    - **Includes**: Many-to-Many relationship between `Order` and `Book` with additional `Quantity` attribute to indicate the number of copies ordered.
    - **Orders**: One-to-Many relationship between `Customer` and `Order`.

- **Reviews**: Many-to-Many relationship between `Customer` and `Book` through `Review` entity.

3. **Relational Schema:**
   - **Book Table**:
     - `BookID` (Primary Key)
     - `Title`
     - `Genre`
     - `PublicationYear`
     - `Price`
   - **Author Table**:
     - `AuthorID` (Primary Key)
     - `AuthorName`
     - `Country`
   - **Customer Table**:
     - `CustomerID` (Primary Key)
     - `Name`
     - `Email`
     - `Phone`
   - **Order Table**:
     - `OrderID` (Primary Key)
     - `OrderDate`
     - `TotalAmount`
     - `CustomerID` (Foreign Key references `Customer.CustomerID`)
   - **Writes Table**:
     - `AuthorID` (Foreign Key references `Author.AuthorID`)
     - `BookID` (Foreign Key references `Book.BookID`)
   - **Includes Table**:
     - `OrderID` (Foreign Key references `Order.OrderID`)
     - `BookID` (Foreign Key references `Book.BookID`)
     - `Quantity` (Number of copies ordered)
   - **Review Table**:
     - `ReviewID` (Primary Key)
     - `Rating`
     - `Comments`
     - `Date`
     - `CustomerID` (Foreign Key references `Customer.CustomerID`)
     - `BookID` (Foreign Key references `Book.BookID`)

Question 7: Travel Agency Management System

A travel agency manages different travel **packages** offered to its customers. For each **package**, the agency stores the `PackageID` (unique identifier), `PackageName`, `Destination`, `StartDate`, and `EndDate`. Each package is handled by one or more **agents**. For each **agent**, the agency keeps track of the `AgentID`, `Name`, `Experience`, and `ContactNumber`. An agent can handle multiple packages.

Customers can book multiple **packages** through the agency. For each **customer**, the system records the `CustomerID`, `Name`, `Email`, and `PhoneNumber`. Each **customer** can book multiple **packages**, and each package can have multiple **customers** associated with it (many-to-many relationship). The agency also tracks the date on which a package was booked and the `TotalAmount` paid by the customer.

The agency provides **guides** for some packages. For each **guide**, the agency keeps the `GuideID`, `Name`, `LanguagesSpoken`, and `GuideType` (local/international). Each guide can be assigned to multiple packages, and each package can have multiple guides.

Tasks:

1. Design an ER diagram to represent the above scenario, capturing the entities, attributes, keys, relationships, and cardinality properties.
2. Convert the ER diagram into a relational database schema.

Solution Overview:

1. **Entities and Attributes:**
    - **Package**: `PackageID` (Primary Key), `PackageName`, `Destination`, `StartDate`, `EndDate`.
    - **Agent**: `AgentID` (Primary Key), `Name`, `Experience`, `ContactNumber`.
    - **Customer**: `CustomerID` (Primary Key), `Name`, `Email`, `PhoneNumber`.
    - **Guide**: `GuideID` (Primary Key), `Name`, `LanguagesSpoken`, `GuideType`.
2. **Relationships:**
    - **Handles**: Many-to-Many relationship between `Agent` and `Package`.
    - **Books**: Many-to-Many relationship between `Customer` and `Package` with attributes `BookingDate` and `TotalAmount`.
    - **AssignedTo**: Many-to-Many relationship between `Guide` and `Package`.
3. **Relational Schema:**
    - **Package Table**:
        - `PackageID` (Primary Key)

        - `PackageName`
        - `Destination`
        - `StartDate`
        - `EndDate`
    - **Agent Table**:
        - `AgentID` (Primary Key)
        - `Name`
        - `Experience`
        - `ContactNumber`
    - **Customer Table**:
        - `CustomerID` (Primary Key)
        - `Name`
        - `Email`
        - `PhoneNumber`
    - **Guide Table**:
        - `GuideID` (Primary Key)
        - `Name`
        - `LanguagesSpoken`
        - `GuideType`
    - **Handles Table**:
        - `AgentID` (Foreign Key referencing `Agent.AgentID`)
        - `PackageID` (Foreign Key referencing `Package.PackageID`)
    - **Books Table**:
        - `CustomerID` (Foreign Key referencing `Customer.CustomerID`)
        - `PackageID` (Foreign Key referencing `Package.PackageID`)
        - `BookingDate`
        - `TotalAmount`
    - **AssignedTo Table**:
        - `GuideID` (Foreign Key referencing `Guide.GuideID`)
        - `PackageID` (Foreign Key referencing `Package.PackageID`)

Question 8: Gym Management System

A gym wants to build a management system to keep track of its **members**, **trainers**, and **classes** offered. For each **member**, the gym stores the `MemberID` (unique identifier), `Name`, `Age`, `MembershipStartDate`, and `MembershipType` (monthly/quarterly/yearly).

Each **trainer** is responsible for conducting multiple **classes**. For each **trainer**, the gym keeps track of the `TrainerID`, `Name`, `Specialization`, and `YearsOfExperience`. Each **trainer** can conduct multiple classes, but each class is conducted by only one trainer.

For each **class**, the system stores the `ClassID`, `ClassName`, `Schedule`, and `Duration`. Each **class** can have multiple members registered for it, and each member can register for multiple classes.

The gym also tracks the **progress** of its members in different classes. For each **member-class pair**, the system records the `ProgressID`, `Attendance` (percentage), and `Grade` (A/B/C/etc.) to evaluate performance.

Tasks:

1. Design an ER diagram to represent the above scenario, capturing the entities, attributes, keys, relationships, and cardinality properties.
2. Convert the ER diagram into a relational database schema.

Solution Overview:

1. **Entities and Attributes:**
    - **Member**: `MemberID` (Primary Key), `Name`, `Age`, `MembershipStartDate`, `MembershipType`.
    - **Trainer**: `TrainerID` (Primary Key), `Name`, `Specialization`, `YearsOfExperience`.
    - **Class**: `ClassID` (Primary Key), `ClassName`, `Schedule`, `Duration`.
    - **Progress**: `ProgressID` (Primary Key), `Attendance`, `Grade`.
2. **Relationships:**
    - **Conducts**: One-to-Many relationship between `Trainer` and `Class`.
    - **RegistersFor**: Many-to-Many relationship between `Member` and `Class`.
    - **Performance**: One-to-One relationship between `Progress` and `Member-Class`.
3. **Relational Schema:**
    - **Member Table**:
        - `MemberID` (Primary Key)
        - `Name`
        - `Age`
        - `MembershipStartDate`
        - `MembershipType`
    - **Trainer Table**:
        - `TrainerID` (Primary Key)
        - `Name`

        - `Specialization`
        - `YearsOfExperience`
    - **Class Table**:
        - `ClassID` (Primary Key)
        - `ClassName`
        - `Schedule`
        - `Duration`
        - `TrainerID` (Foreign Key referencing `Trainer.TrainerID`)
    - **RegistersFor Table**:
        - `MemberID` (Foreign Key referencing `Member.MemberID`)
        - `ClassID` (Foreign Key referencing `Class.ClassID`)
    - **Progress Table**:
        - `ProgressID` (Primary Key)
        - `Attendance`
        - `Grade`
        - `MemberID` (Foreign Key referencing `Member.MemberID`)
        - `ClassID` (Foreign Key referencing `Class.ClassID`)
