a. **ER-to-Relational Mapping Algorithm**
Step 1: Mapping of Regular Entity Types.

- For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E.
- Choose one of the key attributes of E as the primary key for R.
- If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.

For every regular entity in the ERD, here's the relation:

- **USERS(UserID PK, Username, Email)**
- **ITEM(ItemID PK, ItemName, Price)**
- **CATEGORY(CategoryID PK, CategoryName, Description)**
- **CONDITION(ConditionID PK, Type, Detail)**
- **AUCTION(AuctionID PK, StartDate, EndDate, Bid)**
- **PAYMENT(PaymentID PK, Method, TotalAmount)**
- **SHIPPING(ShippingID PK, Address, Cost)**

Step 2: Mapping of Weak Entity Types

- For each weak entity type W in the ER schema with owner entity type E, create a relation R & include all simple attributes (or simple components of composite attributes) of W as attributes of R.
- Also, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).
- The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any.

There are no weak entities in my ERD, so this step does not apply.

Step 3: Mapping of Binary 1:1 Relation Types

- For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.

  There are three possible approaches:

1. Foreign Key (2 relations) approach: Choose one of the relations-say S-and include a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S.
    a. Example: 1:1 relation MANAGES is mapped by Choosing the participating entity type DEPENDENT to serve in the role of S, because its participation in the MANAGES relationship type is total.
2. Merged relation (1 relation) option: An alternate mapping of a 1:1 relationship type is possible by merging the two entity types and the relationship into a single relation. This may be appropriate when both participations are total.

3. Cross-reference or relationship relation (3 relations) option: The third alternative is to set up a third relation R for the purpose of cross referencing the primary keys of the two relations S and T representing the entity types.

This is a 1:1 relationship: AUCTION — PAYMENT (each auction at most has one payment). I place the foreign key on the side with total participation. Like PAYMENT has total participation in RESULTS_IN.

Step 4: Mapping of Binary 1:N Relationship Types.

- For each regular binary 1:N relationship type R, identify the relation S that represents the participating entity type at the N-side of the relationship type.
- Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R.
- Include any simple attributes of the 1:N relation type as attributes of S.

1. **USERS LISTS ITEMS (1:N)**
2. **CATEGORY BELONGS_TO ITEMS (1:N)**
3. **CONDITION HAS_CONDITION ITEMS (1:N)**
4. **USERS CREATES AUCTION (1:N)**
5. **AUCTION SHIPS_BY SHIPPING**

Step 5: Mapping of Binary M:N Relationship Types.

- For each regular binary M:N relationship type R, create a new relation S to represent R. This is a relationship relation.
- Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S.
- Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.

There are no M:N relationships in my ERD. No new tables.

Step 6: Mapping of Multivalued attributes.

- For each multivalued attribute A, create a new relation R.
- This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute.
- The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

There are no multivalued attributes in my ERD. This step is skipped.

Step 7: Mapping of N-ary Relationship Types.

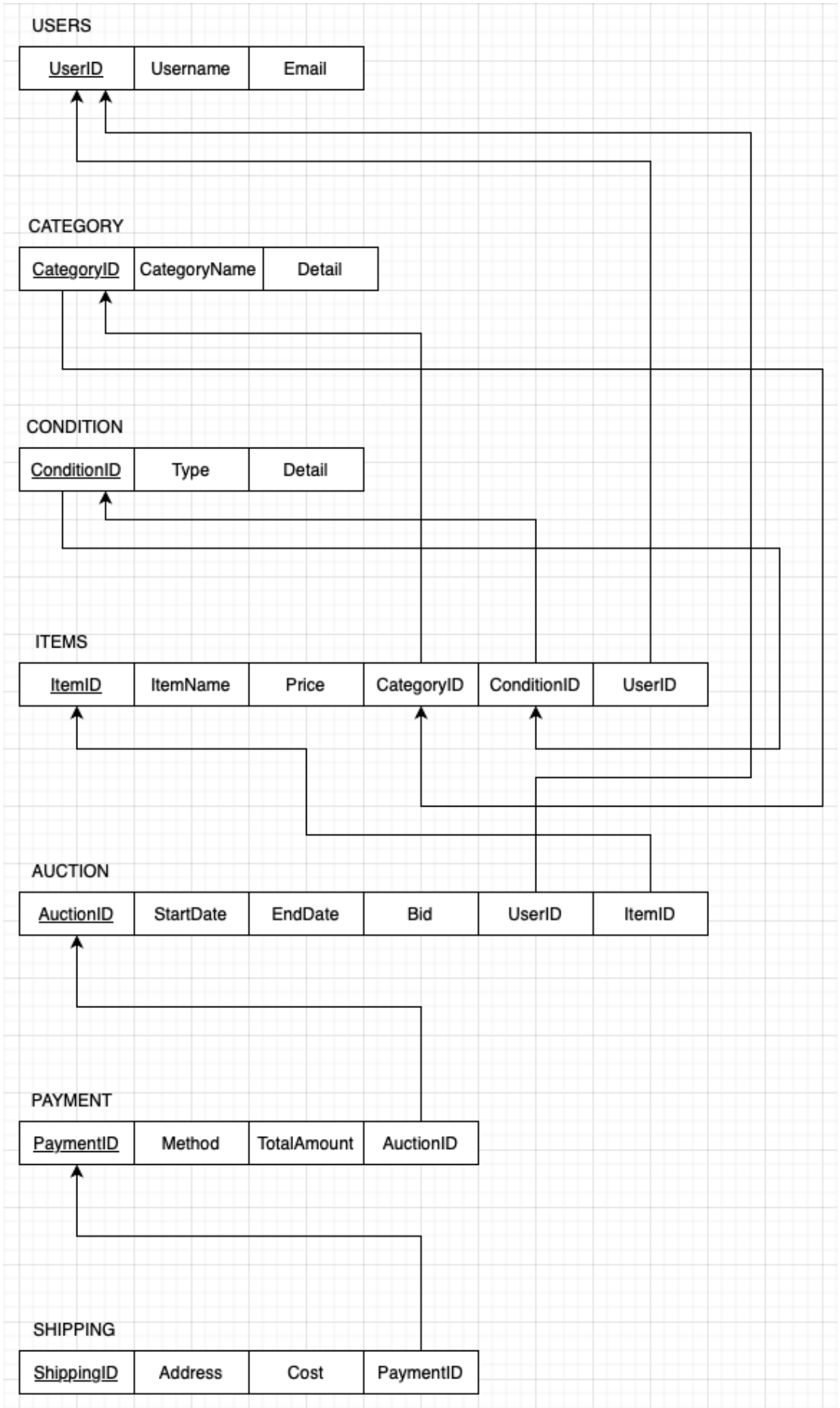
- For each n-ary relationship type R, where n>2, create a new relationship S to represent R.

- Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.
- Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S.

All relationships in my ERD are binary. Skipped this step.

Relational Schema:

- **USERS(UserID PK, Username, Email)**
- **ITEM(ItemID PK, ItemName, Price, UserID FK, CategoryID FK, ConditionID FK)**
- **CATEGORY(CategoryID PK, CategoryName, Description)**
- **CONDITION(ConditionID PK, Type, Detail)**
- **AUCTION(AuctionID PK, StartDate, EndDate, Bid, UserID FK, ItemID FK)**
- **PAYMENT(PaymentID PK, Method, TotalAmount, AuctionID FK)**
- **SHIPPING(ShippingID PK, Address, Cost, AuctionID FK)**


USERS
UserID
Username
Email
CATEGORY
CategoryID
CategoryName
Detail
CONDITION
ConditionID
Type
Detail
ITEMS
ItemID
ItemName
Price
CategoryID
ConditionID
UserID
AUCTION
AuctionID
StartDate
EndDate
Bid
UserID
ItemID
PAYMENT
PaymentID
Method
TotalAmount
AuctionID
SHIPPING
ShippingID
Address
Cost
PaymentID

b. Normalization

Normalization is used to eliminate redundancy, avoid anomalies, and ensure each relation is logically structured.

1NF: No repeating groups, all attributes are atomic.

2NF: No partial dependency, applies only to composite keys.

3NF: No transitive dependency; non-key attributes must depend only on the key.

1. **Users:** UserID, UserName, Email — **Users is in 3NF**
    a. 1NF: All attributes are atomic.
    b. 2NF: UserID is a single attribute, no partial dependencies exist.
    c. 3NF: UserName and Email depend only on UserID. No transitive dependencies, non-key attributes.
2. **Items:** ItemID, ItemName, Price, Description, CategoryID, ConditionID, UserID — **Items is in 3NF**
    a. 1NF: All attributes are atomic.
    b. 2NF: ItemID is a single attribute, no partial dependencies exist.
    c. 3NF: ItemName, Price, and Description depend on ItemID. CategoryID, ConditionID, and UserID are foreign keys and do not determine any other attributes. No non-key attributes depend on other non-key attributes.
3. **Category:** CategoryID, CategoryName, Detail — **Category is in 3NF**
    a. 1NF: All attributes are atomic.
    b. 2NF: Primary key is a single attribute, no partial dependencies exist.
    c. 3NF: CategoryName and Detail depend on CategoryID. No transitive dependencies.
4. **Condition:** ConditionID, Type — **Condition is in 3NF**
    a. 1NF: All attributes are atomic.
    b. 2NF: Primary key is a single attribute, no partial dependencies exist.
    c. 3NF: Type depends directly on ConditionID, and there are no additional attributes.
5. **Auction:** AuctionID, StartDate, EndDate, Bid, UserID, ItemID — **Auction is in 3NF**
    a. 1NF: All attributes are atomic.
    b. 2NF: AuctionID is a single attribute, no partial dependencies exist.
    c. 3NF: StartDate, EndDate, Bid, UserID, ItemID depend on AuctionID. UserID and ItemID are not determined on other attributes. No transitive dependencies.
6. **Payment:** PaymentID, Method, TotalAmount, AuctionID — **Payment is in 3NF**
    a. 1NF: All attributes are atomic.
    b. 2NF: Primary key is a single attribute, no partial dependencies exist.
    c. 3NF: Method and TotalAmount depend on PaymentID. AuctionID doesn't determine any non-key attributes. No transitive dependencies.
7. **Shipping:** ShippingID, Address, Cost, PaymentID — **Shipping is in 3NF**
    a. 1NF: All attributes are atomic.
    b. 2NF: Primary key is a single attribute, no partial dependencies exist.
    c. 3NF: Address, Cost, and PaymentID depend on ShippingID. PaymentID does not determine Address or Cost.

All relations satisfy 1NF, 2NF, and 3NF. The entire database schema is already in 3NF.

c. List all the constraints clearly as described in the step above.

USERS

- **PK:** UserID
- **UNIQUE:** Email
- **NOT NULL:** UserName, Email

ITEMS

- **PK:** ItemID
- **FK:** UserID - USERS(UserID), CategoryID - CATEGORY(CategoryID), ConditionID - CONDITION(ConditionID)
- **NOT NULL:** ItemName, Price
- **CHECK:** Price >= 0

CATEGORY

- **PK:** CategoryID
- **NOT NULL:** CategoryName

CONDITION

- **PK:** ConditionID
- **NOT NULL:** Type

AUCTION

- **PK:** AuctionID
- **FK:** UserID - USERS(UserID), ItemID - ITEMS(ItemID)
- **NOT NULL:** StartDate, EndDate, Bid
- **CHECK:** EndDate >= StartDate, Bid >= 0

PAYMENT

- **PK:** PaymentID
- **FK:** AuctionID - AUCTION(AuctionID)
- **UNIQUE:** AuctionID (1:1)
- **NOT NULL:** Method, TotalAmount
- **CHECK:** TotalAmount >= 0

SHIPPING

- **PK:** ShippingID
- **FK:** PaymentID - PAYMENT(PaymentID)
- **NOT NULL:** Address, Cost
- **CHECK:** Cost >= 0

In this project, I wrote a database based on the ERD related to Step 2. And I chose the entities **Users, Items, Category, Condition, Auction, Payment, and Shipping**. They all represent the core components of an online auction system.
How are they connected? Users list items for sale, Items belong to a category associated with a condition, Items show in auctions that users can place Bid, Each completed Auction results in exactly one payment and each payment linked to one shipping information.
I transformed the ERD into a consistent relational schema. Therefore, this structure makes the database easier to manage, more organized, and better suited for an online auction platform.