**Instructions:**

For each **table** in your ERD, you should have a paragraph explaining:

* What data is in this table?
* What attributes are included?
* Are there any foreign keys?
* What is the primary key?
* What table(s) does this table have a relationship with?
* What is the relationship between the tables, and why is it that?

**Entity List:**

* USERS
* ADDRESSES
* USER\_ADDRESSES (Bridge table. Might remove/simplify later)
* USER\_ROLE
* PRODUCTS
* CATEGORY
* SUB-CATEGORY
* MANUFACTURER
* WAREHOUSE
* ORDERS
* ORDER\_DEATILS (bridge table)

**Entity Explanation Paragraphs:**

* **USERS**
  + At the current moment, the data that will be held in the users table includes basic user information including a username, first name, last name, date of birth, email address, registration date, user status, an address, and a role. In the context of an ecommerce application, a user is any person who wants to register to view, create, or purchase products. Within this users table, there is a series of basic user identifiers that will be captured upon registration on the ecommerce site. Upon registration each user will be assigned a unique UserID to ensure we don’t have any duplicate users even if two users have some of the same information. In this table there are 2 foreign keys. One foreign key relates to an AddressID in the ADDRESS table and the other refers to a RoleID in the USER\_ROLE table. The USERS table relates to the USER\_ADDRESSES table in a 1:M relationship because a user could have multiple address. The USERS table also related to the USER\_ROLE table so that each user is assigned 1 role, but role could belong to multiple users.
* **ADDRESSES**
  + The ADDRESS table hold exactly what it sounds like, address information. In this implementation of an ecommerce site, a user could potentially have multiple saved addressed within their account. In this instance, an address consists of an address name, address line 1, address line 2 (apartment or building if needed), city, state, zip code, and country. At the moment this table does not have any foreign keys, but if simplified it will have 1 foreign key relating to a UserID.
* **USER\_ADDRESSES** *(might remove later and simplify relationship)*
  + This table is a bridge table. This table is necessary to solve a M:M relationship between the USERS and ADDRESSES table. The only additional information that this table contains is whether or not a user address is their primary address or not. In this table there are two foreign keys. One foreign key relates to the UserID from the USERS table while the other foreign key relates to the AddressID from the ADDRESSES table. This table is used to convert a M:M relationship to two different 1:M relationships with the ADDRESSES and USERS table.
* **USER\_ROLE**
  + Upon registration, each user will automatically be assigned a default role. Each users needs a role because there will be different kinds of users within the ecommerce website. For example, there might be a default role, an admin role, a moderator, and potentially many more. The USER\_ROLE table will store a role name, role description, role creation date, and a flag as to whether or not the role is active. Each role upon creation will be assigned a unique primary key defined as the RoleID. This table has a 1:M relationship to the USERS table. More specifically a role can belong to many users, but a user can only have 1 role at a time.
* **PRODUCTS**
  + The products table is the most complex and important in regards to this ecommerce website. The products table contains all the essential information about products including the product code, product name, product description 1, product description 2, the product cost, the product sale price, isActive flag, isDeleted flag, the product category, product sub-category, product manufacturer, and product location. Each product is assigned a unique primary key upon creation. The products table has 4 foreign keys. One foreign key points to the item’s category. An item can only belong to 1 category at a time. Another foreign key points to a sub-category. An item can also only belong to 1 sub-category at a time. Another foreign key is the manufacturerID. This field points to whoever the manufacturer of the product is. This field is optional as some products may not have a manufacturer or they may be made in house. The final foreign key is the warehouse ID key. This key tells what warehouse a product is currently stored at.
* **CATEGORY**
  + The category table contains the list of different categories a product can belong to. In the instance of sporting goods, the main category may be something like MLB, NBA, NCAA, or even MISC. The category table currently stores the categoryID, the category name, and the category description. In the future this table can be expanded to stored additional category information if needed. There are currently no foreign key relationships within this table. A category can belong to multiple products, but a product can have only 1 category at a time.
* **SUB-CATEGORY**
  + The sub-category table contains the different sub-categories a product can belong to. Again, in relation to a sporting goods business, the sub category would be the specific team’s name that an item belongs to. For example, the main category may be MLB and the sub-category would then be Chicago Cubs. Like the category table, the sub-category table currently stores the sub-category id, the sub-category name, and the sub-category description. There are currently no foreign key relationships within this table. A sub-category can belong to multiple products, but a product can have only 1 sub-category at a time.
* **MANUFACTURER**
  + The manufactures table served to uniquely identify each manufacturer that is sold and distributed by the company we are building this database for. The manufactures table is used to make sure we don’t repeat each manufacturer each time it is used in the products table. Because we may carry/sell multiple products from the same manufacturer we don’t want to repeat that value everywhere in the products table. This table stores basic information like the id, name, description, address 1, address 2, city, state, zip, and country of each manufacturer. This table currently doesn’t have any foreign keys within it. A product can have only 1 manufacturer at a time, but a manufacturer can belong to many products.
* **WAREHOUSE**
  + The warehouse table is used to identify where each product is located within the business. As All Sports has two different warehouse locations, we need to be able to identify where each product is currently located. The WAREHOUSES table contains the warehouse id, identifying name, address 1, address 2, city, state, zip code, and country. These identifiers can be used to identify the geographic location of each item that All Sports sells. The warehouse table currently does not contain any foreign keys. The relationship to the products table is 1:M. More specifically a warehouse can belong to many products, but a product can only belong to one warehouse at a time.
* **ORDERS**
  + As this is an ecommerce website, we want users to be able to order products from the website. The ORDERS table is used to stored the various orders that a user may or may not place. The orders table stores basic order information including the orderID, the userID, the order total, the order date, and the addressID. In this table, there are currently two foreign keys. One foreign key relates to the UserID who placed the order and the AddressID of the user who placed the ID. In this business, a user may have 0 or many orders. Each order consists of 1 or many products, but since we cannot have a M:M relationship in a relational database, we must create a bridge table to deal with this situation. This new table ORDER\_DETAILS can then have a M:1 relationship with the ORDERS table and M:1 relationship with the products table.
* **ORDER\_DEATILS**
  + The ORDER\_DETAILS table acts as a bridge table between the orders table and the products table. This table is necessary because without it there would be a many to many relationship between PRODUCTS and ORDERS. As an order many contain many products and a product may exist on many orders, we need a bridge table to solve this relationship. This is where the ORDER\_DETAILS table come in to convert a M:M relationship into two separate 1:M relationships. The order details table contains the order details unique id, the order id, the product id, and the quantity ordered. This table contains 2 foreign keys. The first foreign key is the OrderID which links back to the ORDERS table to identify what order was placed. The second foreign key is the ProductID. This foreign key relates back to the PRODUCTS table and whatever product is a part of the order.