

TECHFAM

CMPSC 431W

Phase Two: Shoe Retailers' Database

Ben Myers

Chao Guo

Hiren Patel

Guojing Wang

Jeff Li

Dishti Gurnani

Table of Contents

Table of Contents	2
2. ANALYSIS AND CONCEPTUAL DESIGN.....	6
2.1. Sales Item.....	6
Figure 1: Sales_Item, Footwear, Address, and Image ER Diagram	6
2.2. Categories	8
Figure 2: Category ER Diagram	9
2.3. Suppliers	10
Figure 3: Suppliers, Company, Phone ER Diagram.....	10
2.4. Company.....	11
2.5. Registered Users	12
Figure 4: Registered_Users, Suppliers, Company, Credit_Card ER Diagram	12
2.6. Rating	13
Figure 5: Rating ER Diagram.....	14
Figure 6: Items_Review ER Diagram	15
2.7. Browsing	15
2.8. Searching	16
Figure 7: Searching Process Diagram.....	17
2.9. Sale.....	17
Figure 8: Sale ER Diagram	18
2.10. Bidding.....	18
Figure 9: Auction and Bid ER Diagram	20
2.11. Order and Sale Reports	21
Figure 10: Order Sale Report Process Diagram	22
2.12. Delivery	22
Figure 11: UPS ER Diagram.....	23
2.13. Custom Shoes	23
Figure 12: Custom_Shoe ER Diagram.....	24
2.14. Customer Service	25
Figure 13: Customer_Service ER Diagram.....	25
3. MASTER ER DIAGRAM	26

4. TECHNOLOGY SURVEY	27
4.1. MySQL	27
Figure 14: Pros and Cons of MySQL.....	27
4.2. Frontend	28
Figure 15: Frontend Programming Languages	28
4.3. Backend	28
4.4. Other Technologies	29
5. CATEGORY TREE AND NORMALIZATION.....	30
SCHEMAS IN THIRD NORMAL FORM	31
6. SQL STATEMENTS	34
6.1. Sales Item.....	34
Figure 18: SQL Table for Sales_Item	34
Figure 19: SQL Table for Footwear.....	34
Figure 20: SQL Table for Address.....	35
Figure 21: SQL Table for has_visual	35
Figure 22: SQL Table for Image.....	35
6.2. Categories	35
Figure 23: SQL Table for Category	35
6.3. Suppliers	35
Figure 24: SQL Table for Supplier	36
Figure 25: SQL Table for Phone.....	36
Figure 26: SQL Table for contact_by	36
6.4. Company.....	36
Figure 27: SQL Table for Company	36
6.5. Registered Users	36
Figure 28: SQL Table for Register_Users.....	37
Figure 29: SQL Table for Credit_Card	37
6.6. Ratings	37
Figure 30: SQL Table for Ratings.....	37
Figure 31: SQL Table for Item Review.....	37
6.7. Sale.....	38

Figure 32: SQL Table for Sale	38
6.8. Bidding.....	38
Figure 33: SQL Table for Auction	38
Figure 34: SQL Table for Bid	39
6.9. Delivery.....	39
Figure 35: SQL Table for UPS.....	39
6.10. Custom Shoes	39
Figure 36: SQL Table for Custom_Shoe.....	39
6.11. Customer Service	40
Figure 37: SQL Table for Customer_Service	40
7. CONCLUSION.....	41
8. FINAL SYSTEM FRAMEWORK	42
Figure 38: Framework	42
9. REFLECTION	44
10. APPENDIX A – GITHUB GRAPHS	45
11. APPENDIX B – ASANA SUMMARY	46
12. APPENDIX F – DATABASE STORAGE	47
13. WORKS CITED	62

1. INTRODUCTION

When forming our startup, we considered several different items to sell. We wanted to distribute a popular product that was easy to categorize for simple website navigation. We brainstormed and decided to join the \$52-billion-dollar shoe industry.^[5]

After choosing to create an online shoe store, we looked at competing companies to see how they displayed their products. We noticed Amazon provides a name, description, price, image, company name, delivery information, and availability (sometimes including number in stock).^[1] While looking at eBay, we found additional important details, like current bid amounts and condition of the product.^[3] This gave us a general idea of what information we needed to store for each sales item.

Next we looked directly at the competition, and began browsing shoe sites. We noticed that stores like Designer Shoe Warehouse allowed you to search for shoes based on category, brand, size, and color.^[8] We also discovered that shoe stores often sell other products like socks, backpacks, and shoe care items.^[2]

This report covers a detailed description of the conceptual design and analysis of our online shoe store. We cover the idea and working that goes into each feature provided through this online shoe store. We also discuss the database and web structure along with the all the technology that is used to build this online shoe store.

Using our knowledge on Database Management Systems we will develop an envisioned model that helps customers shop for shoes in the most intriguing way possible. We will meet the following requirements described further in this analysis.

2. ANALYSIS AND CONCEPTUAL DESIGN

2.1. Sales Item

With our initial research complete, we designed the **Sales_Item** entity to represent each product (See Figure 1). It is identified by its unique key, *item_id*. Every sales item has a *name* and *description* so sellers can properly market their product. We added a *brand* attribute so users can search for their favorite brands (See Section 2.7). Each sales item has two prices recorded, the *list_price* and the *reserved_price*, which are explained in more detail in Section 2.8 and 2.9. The *state* attribute describes the quality of used products. Finally, the *count* attribute tracks how many items are in stock.

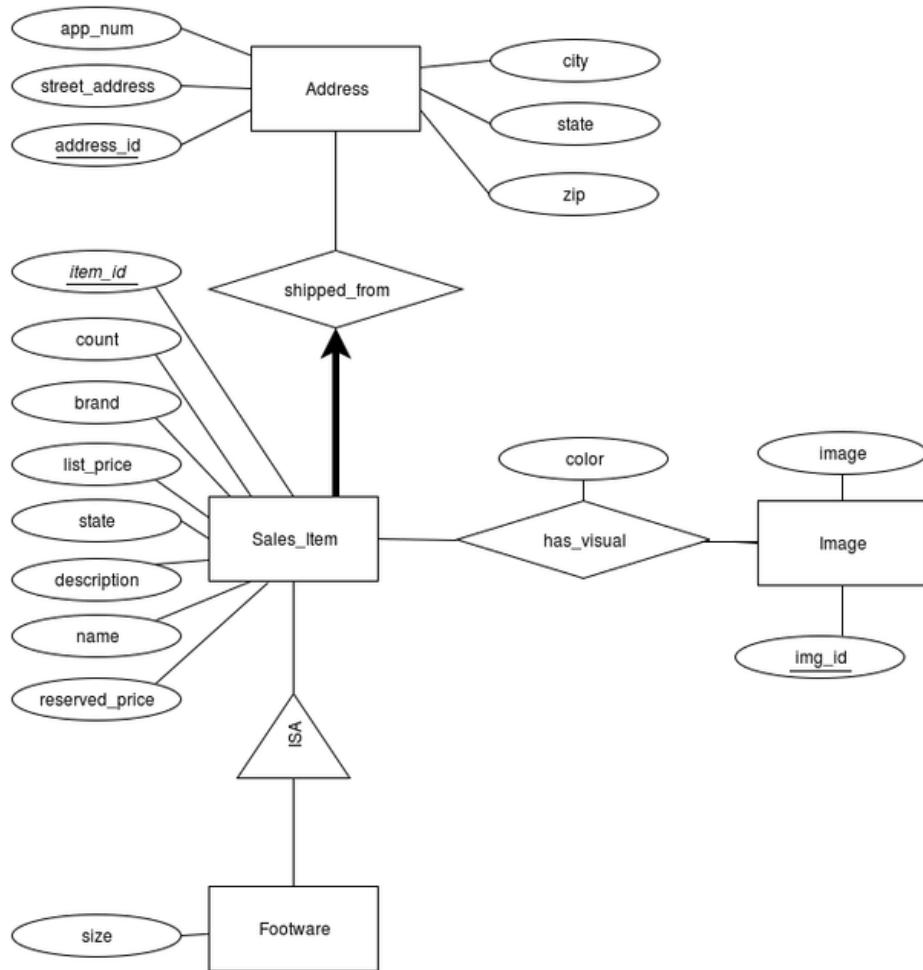


Figure 1: Sales_Item, Footware, Address, and Image ER Diagram

The ER diagram displayed in Figure 1 was later converted into a table in our database with **Sales_Item** set as the name of the table. *Item_id* was set as the primary key and is stored as an integer value. The rest of the attributes were added as regular columns, where *count* was set as an integer, *brand*, *state*, *description* was set as string values. Also, the *list_price* and *reserved_price* was added as decimal values, only allowing 2 digits after the decimal, thus using the notation for currency.

Like other shoe websites, we wanted to sell non-shoe items. We provide bags, sunglasses, shoe care items, and socks. Bags, sunglasses, and care items can be fully described by the **Sales_Item** entity, but socks and shoes require additional information. As a result, we created the **Footwear** entity (See Figure 1) to account for these missing fields, which includes the *size* attribute (for socks and shoe size).

In order to show this ISA relationship in the database, we made a new table called **Footwear**, where a foreign key (*item_id*) from the **Sales_Item** table is used as the primary key. That way, every **Footwear** field is required to have a corresponding **Sales_Item** field, which holds the rest of the data for the **Footwear** field. We also added ON DELETE CASCADE line to the foreign key, so that if the **Sales_Item** field is deleted, the **Footwear** item field that corresponds to the given **Sales_Item** will also be deleted. Lastly, the **Footwear** table has a column called *size* which holds the size of the shoe (recorded as a string).

Next, **Sales_Items** require information that is too complex to represent as attributes. The **Address** was created as a separate entity because attributes may be reused by other entities, such as users and shipping. Note that every **Sales_Item** is required to have exactly one address. Attributes of the **Address** entity include the *street_address*, *app_num* (apartment number), *city*, *state*, and *zip_code*. Each **Sales_Item** has its own visual, where color is an attribute of the relationship between the **Sales_Item** and the visual. This attribute allows the user to filter out products when searching or browsing, based on the color of bag, shoe, or sock that they desire (See sections 2.7 and 2.8). Visual is

represented by the **Image** entity (See Figure 1). Every **Sales_Item** can have one or more image, which will be displayed on each product's webpage. Each image has a unique *img_id*, as well as an *image* attribute, which holds the actual photo data.

As shown in the Figure 1, the **Address** entity is too complex to be record as a single attribute, so it was turned into its own table called **Address** (See Figure 1). Again, the primary key is the *address_id* (set as an integer) and the rest of the attributes are recorded as strings (or as an integer for zip) in the table. Since a **Sales_Item** is required to have exactly one **Address**, we added an *address_id* foreign key to the **Sales_Item** table, and force all entries in that column to be not null. That way, every field in **Sales_item** will have an address. Note that an address does not necessarily have to have a relationship with a **Sales_Item**.

The **Image** table on the other hand, needs a primary key *img_id* (integer) and an image name stored in *image*(char). As shown in the ER diagram, there is a many-to-many relationship between **Sales_Item** and **Image**, so the **has_visual** table was created to record this relationship. In the **has_visual** table, the primary keys of a **Sales_item** and **Image** are stored, and the combination of these two foreign keys act as the primary key of the table. In addition, the relationship in the ER diagram contains the attribute *color*. This was added to the table and is represented as a column of strings.

2.2. Categories

There is a variety of footwear to categorize. This categorization will dictate the databases needed for this online business. TechFam must distribute enough styles of footwear to become a central hub for all individuals looking to purchase shoes. However, it is critical to be efficient when creating the databases, avoiding any data redundancies.

TechFam's shoe selection targets people of all genders and ages. As a result, the root node, "All," must have three children, "Men", "Women", and "Kids". The children nodes of these three

nodes are categories of the different types of shoe, such as “Boots,” “Sneakers,” etc. Next, the different types of shoes are broken down even further by subtypes. In addition to the first three children nodes, there is a fourth node, “Accessories,” contains items that are not shoes. See Appendix C for an in-depth visualization of the categories.

We researched a few different shoe websites to obtain the categories. [Zappos](#)^[7] is a general shoe distributor that contains the vastest selection, contributing to the majority of our category ideas. [JustFab](#)^[11] delved deep into the various types of women’s shoes.

The ER diagram in Figure 2 explains the basic outline for how each category will be stored in the database. **Categories** have primary key, *categoryID*, to uniquely identify each other. The *description* attribute briefly details the kind of product that is contained within the category itself. Note that there are relationships between the category and its parent node and children node(s) which follow the classification tree in Appendix C. Each child node points to its parent, and can have a max of one parent. The parent, on the other hand, can have multiple children. This is shown by the **Parent_of** relationship.

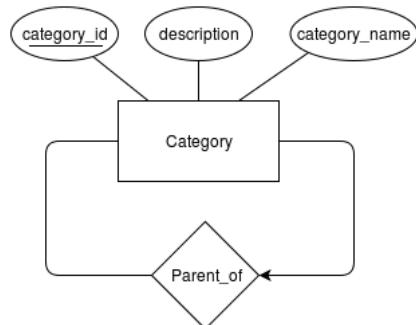


Figure 2: Category ER Diagram

When the ER Diagram was converted into a table, the **Category** entity and the **Parent_of** relationship were combined into one table called **Category**. This table uses the *category_id* (integer) as the primary

key, and contains *description* and *category_name* columns. These components make up the **Category** entity section of the table. Next, the *parent_id* was added so that each child could identify its parent **Category** field. Since *parent_id* is an attribute, we guarantee that each child can only have a max of one parent (or no parent at all if it's the root node), while parents can still have multiple children.

There is one more important relationship in the ER diagram called **member_of**. This shows that every **Sales_Item** is required to have an identifying category. This relationship was merged with the **Sales_Item** table with the *category_id* column. This is a foreign key that points to the **Category** the **Sales_Item** is a member of. Note also that *category_id* is set to NOT NULL, so that every **Sales_Item** is required to have a **Category**.

2.3. Suppliers

Suppliers is a parent entity of both companies and registered users. A **Suppliers** job is to sell shoes that they manufacture on our website. The attributes for the **Suppliers** entity include the *supplier_id*, *email* and *name*. The supplier is assigned a *name* and is used to identify the supplier.

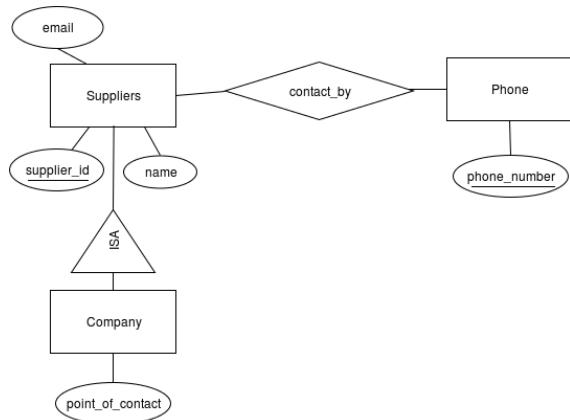


Figure 3: Suppliers, Company, Phone ER Diagram

Multiple phone numbers of the suppliers are also recorded, as represented by the **contact_by** relationship. The point of contact helps the customer contact the seller in case there are any questions

or concerns. This relationship requires two additional tables (besides the **Supplier** table). The first table is the **Phone** table, which simply has a primary key of phone number. The second table is the **contact_by** table. This table takes two foreign keys, the primary key of **Suppliers**, and the primary key of **Phone**. Then it will match a given phone number with a supplier. The primary keys of this table is the combination of both foreign keys.

The supplier also has an **Address**, shown by the *location_address* relationship. A **Supplier** can have many addresses, but an **Address** can only point to a single **Supplier**. Therefore, there is a one-to-many relationship between **Suppliers** and **Address**. The database records this information by adding the *supplier_id* attribute to the **Address** table, which points to the primary key of a **Supplier**.

2.4. Company

Company is a child entity of **Supplier**. It contains all the information of supplier, with the addition of the *point_of_contact* attribute, which is the name of the person to contact for the given company. **Company** can use the name attribute of **Suppliers** to record the company name, and also use the email attribute as well as the phone and address relationships to record where the company is located, and how to contact them. An important thing to note is that companies cannot buy products; they can only sell.

The **Company** table is fairly simplistic. It uses the **Supplier's** *supplier_id* as its primary key. If the **Supplier's** field is deleted, the corresponding **Company's** field will also be deleted. The **Company** table also has the *point_of_contact* column, which stores strings.

2.5. Registered Users

Registered_Users is the second child entity of **Suppliers**. Along with selling products, **Registered_Users** can bid or directly buy shoes. This feature asks the user to fill out the following information while they register:

- A user ID or username, which helps users stay anonymous and allows us to identify them.
- A password to secure their account.
- Gender.
- An email and multiple phone numbers as a mode of contact to the registered user. Although this information are located in the in the **Supplier** entity.
- Multiple credit cards and addresses of the registered user, which helps process payments, billing information, and return information.
- Age and income is used to calculate a statistic for each user. The income here stands for the income the user has made on our website.

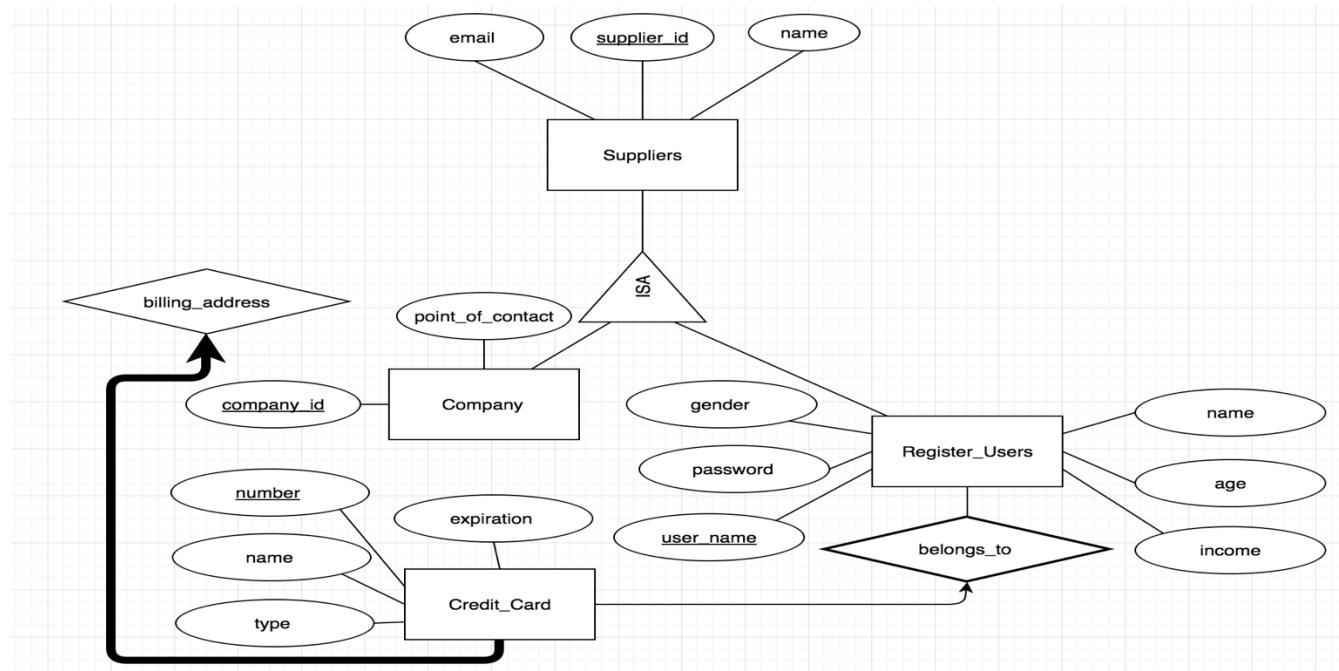


Figure 4: Registered_Users, Suppliers, Company, Credit_Card ER Diagram

Similar to the **Company** table, the **Register_Users** table has a foreign key that links to a unique *supplier_id*. The *supplier_id* cannot be null, that way every **Register_Users** has a corresponding **Supplier** entity. Like the **Company** table, the **Register_Users** field will be deleted if the **Supplier** field is deleted. Unlike the **Company** table, the **Register_Users** use their own *username* attribute as the primary key. Since every *username* must be unique and is the main way other users will identify one another, we thought *username* would suite as a better primary key.

The **Credit_Card** is a separate entity. This allows each **Register_User** to have multiple credit cards and multiple billing addresses. It records the *username*, credit card *number*, and *billing_address_id* of each member. It also uses these two foreign keys as the table's primary key.

2.6. Rating

Rating is an important measure for a seller's reputation. A user might want to know what rating a seller might have. Many websites now have rating system for sellers. For example, sellers on eBay can earn stars based on the number of ratings they get^[4]. The rating system on this website can be modeled by a **Rating** entity, which has a relationship with a single **Register_User**, who create the rating, and a single **Supplier**, who the register user rates. The application quality entity **Rating** is created with primary key *rating_id*, a scale based non-key attribute *value*, and a textual non-key attribute *explanation*.

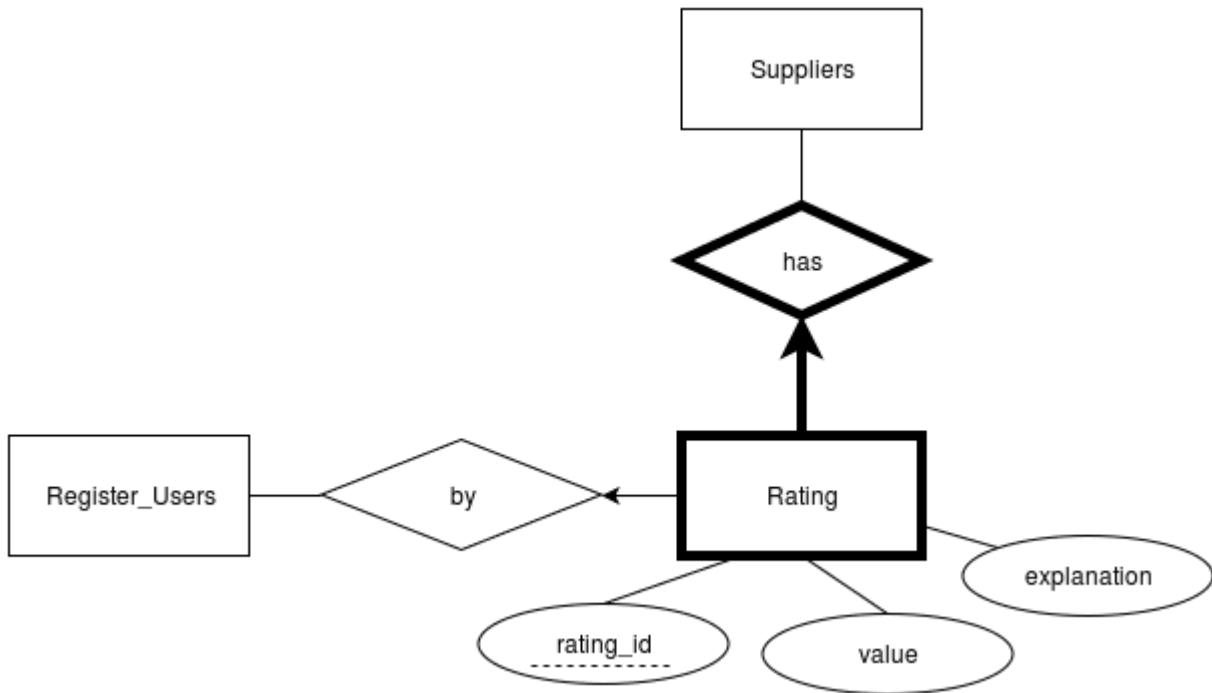


Figure 5: Rating ER Diagram

The *value* is a scale attribute that can be selected from 1 star to 5 stars. The *explanation* is a textual attribute that allows a user to explain why one gives such scaled-rating or any additional comments. A seller will also have an average rating showing on his/her profile. The average rating calculates the mean of each rating value then rounds to the nearest star scale.

For the table for **Rating**, there is a partial key *rating_id*, in addition to the *explanation* and *value* attributes. There is also a foreign key pointing to the *username* of the reviewer and the *supplier_id* of the company or user being reviewed. These two foreign keys help combine **Rating** with the **by** and **has** relationships. Also, in the ER diagram, **Rating** is a weak entity to **Supplier**. So if the **Supplier** deletes their account, then all the ratings for them will also be deleted. This is done with the ON DELETE CASCADE line where the *supplier_id* is declared a foreign key. **Rating** is given a primary key which combines the *supplier_id* with the partial key *rating_id*.

We also have a review for items called Item_Review. It takes the and *item_id* instead of *supplier_id*.

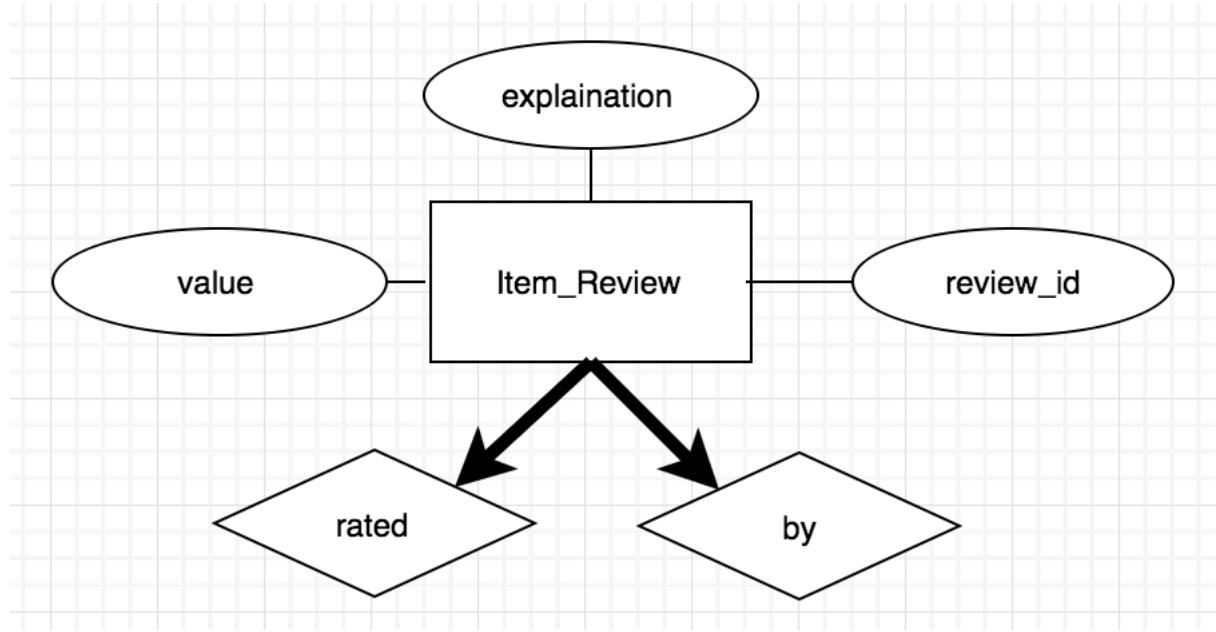


Figure 6: Items_Review ER Diagram

2.7. Browsing

Users are able to browse the items by traversing the category tree (Appendix C). When a user chooses a category, the user will want to know all of the categories that are directly linked to that current category. The tree traversal algorithm to be implemented will follow a derivative of the breadth-first search algorithm. At each point, they are given a summary of all the items that appear in that category. For example, if the user is in the “Women’s” category, the provided categories are “Boots”, “Flats”, “Pump”, “Sandals”, “Sneakers”, and “Wedges”. However, in traditional breadth-first search algorithms, the other nodes in the same level as the previously mentioned categories, will be shown. So our form of the algorithm will limit the traversing within the current node and its children.

Another important browsing feature is to allow the user the ability to traverse both ways through the tree, including moving downwards and upwards through the tree. The user will be able to

return to the parent node of their current category. In addition, there will be a home button to traverse the user back to the root node.

In the browser bar, user can access their account information. This includes their profile, feedback, and settings. Profile contains their personal information, items they are selling, current bids, and history. Feedback has ratings and reviews. Settings include login information, credit card information, and shipping/billing address.

There will be a homepage which consists of the current deals and other features usually found in homepages. This homepage is not located anywhere in the classification tree, yet contains various items from different categories.

During our research of other shoe/clothing sites, we were attracted to a certain type of user interface that made browsing very easy and fast. Particularly, Macy's online store featured a row of tabs near the top of the page. These tabs were the major categories of clothing, "Men", "Women", etc. When the user selects one of the tabs, a drop down menu appears, listing all of the categories within that major category. Furthermore, under those categories were subcategories that were indented to easily show its lower levelness. We found this interface in other sources as well and we find this way of browsing to be the most user-friendly because it gives the user a broad overview of the classification tree. TechFam will follow this structure of browsing.

2.8. Searching

On shopping websites, like Amazon, a user might want to search for a series of specific items. This can be modeled by two kinds querying, "User search Items", and "User choose Categories then search Items", as shown in Figure 5.

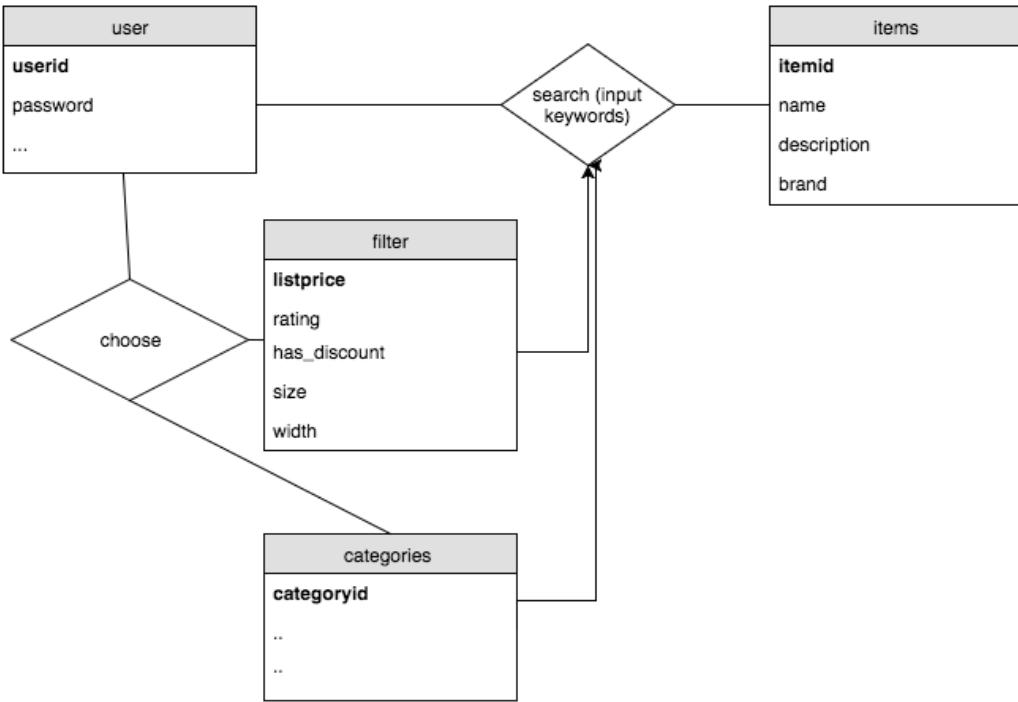


Figure 7: Searching Process Diagram

If a user chooses to input keywords, the database will look for items that contains these keywords. A user can also choose certain categories or other filters first then input keywords to narrow down the results. Attributes in Sales_Item like *name*, *description*, *brand*, etc. shall be searchable.

2.9. Sale

Items on sale will be the ones not auctioned off. We will create an option for buying the item now. When the user enters a specific item description, that item information will be pulled from our database. The user interface will display all the items matching that description. The user will be allowed to add items to the cart, use their credit card information and check out. After the transaction has completed the item count will decrement. The transaction information will remain in the system for six months and will be stored using the ER diagram in Figure 6.

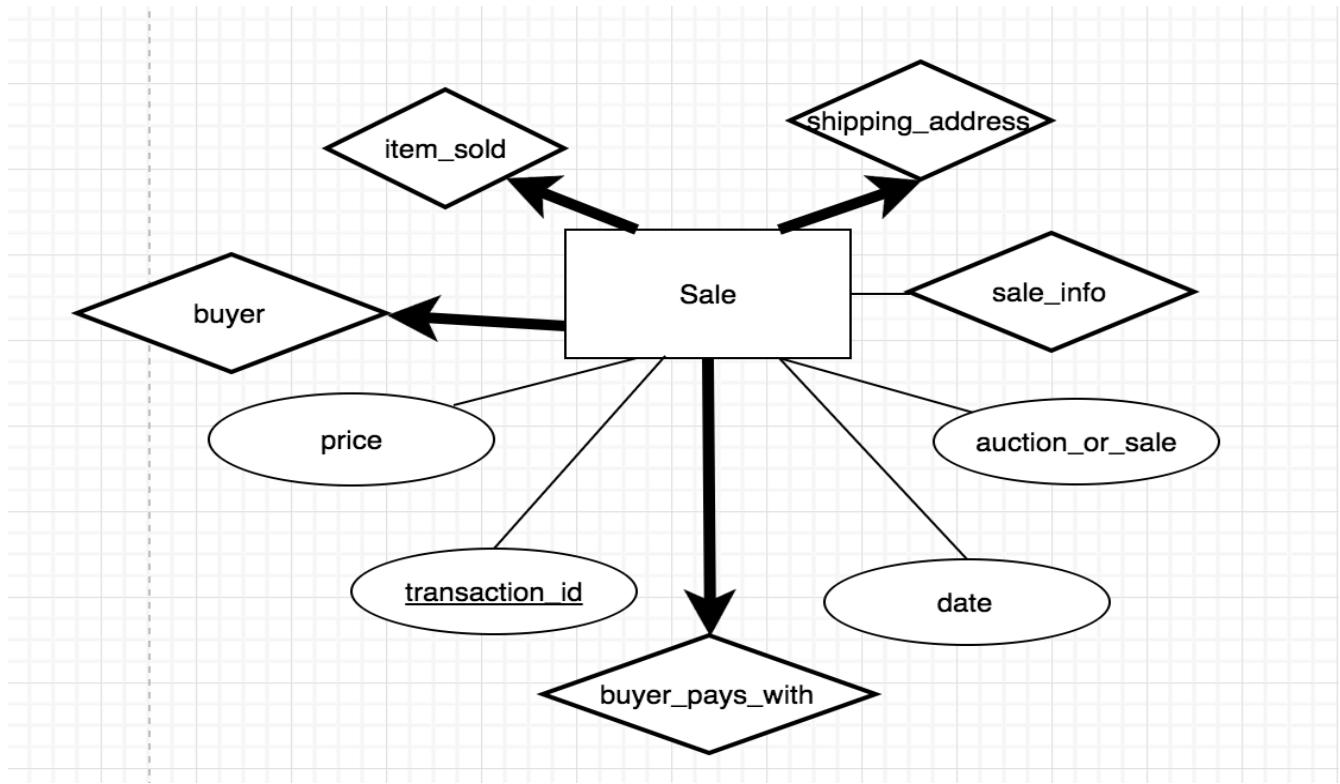


Figure 8: Sale ER Diagram

We created a table call **Sale**, where we had *transaction_id* as the primary key. The table checks if it is an auction or a regular sale, using a Boolean. The foreign key that was referenced from Sales_Item was *item_id*, where it was linked on the ER diagram by relationship *item_sold*. It also includes *credit_card_number* which is referenced from the Credit_Card entity. This credit card information belongs to the user that bought the item.

2.10. Bidding

Unlike many shoe websites, our database will have the option of auctioning items. A user can access these items by searching through the pull-down menu and clicking on an item through the categories included, as well as entering a specific item name through the search bar. The database will

retrieve information and after clicking the auctioned items option the user will be able to view the item. The user has the option to sort items based on price from highest to lowest or vice versa.

Bidding Rules:

- Users must remember each bid is a contract.
- Each item will have a specific time stamp listed under description.
- Bids can only be canceled 24 hours prior to time stamp deadline.
- Bids will be in 2 dollar increments.

Selling an item:

- Under options, select selling button and Users can only sell one item at a time.
- Select the item for sale and give a brief description.
- List the timestamp value for the starting and ending.
- Select the reserve price for the item.
- Review everything and select the confirm button.
- You will be notified when the auction has ended and who had the winning bid.

We will feature manual bidding and automatic bidding. First let's talk about the manual way, where the user enters a bid value that is at least \$2 more than the current highest bid. The user will be notified when someone overtakes them as the lead bidder. On the other hand, we have automatic bidding, where the user will be prompted to enter the maximum value they are willing to pay for the item being auctioned off. The system will bid \$2 more than the current highest bid until your bid reaches the maximum value you are willing to play or the auction ends. The user with the highest bid wins, and pays the amount they bid. For both bidding options at the end of the auction, the user with the winning bid will be notified, that he or she has won. The system will charge the user's credit card with winning bid amount and after receiving the payment, the seller will be required to ship the item.

The transaction will stay in the system for 6 months. The user can view their current bids under the account options.

The **Auction** will have a *time_stamp_start* for the start of the auction and a *time_stamp_end* for the end of the auction. The auction will record all the bids that users place and these will have a relationship (**item_auctioned**) to the sale items.

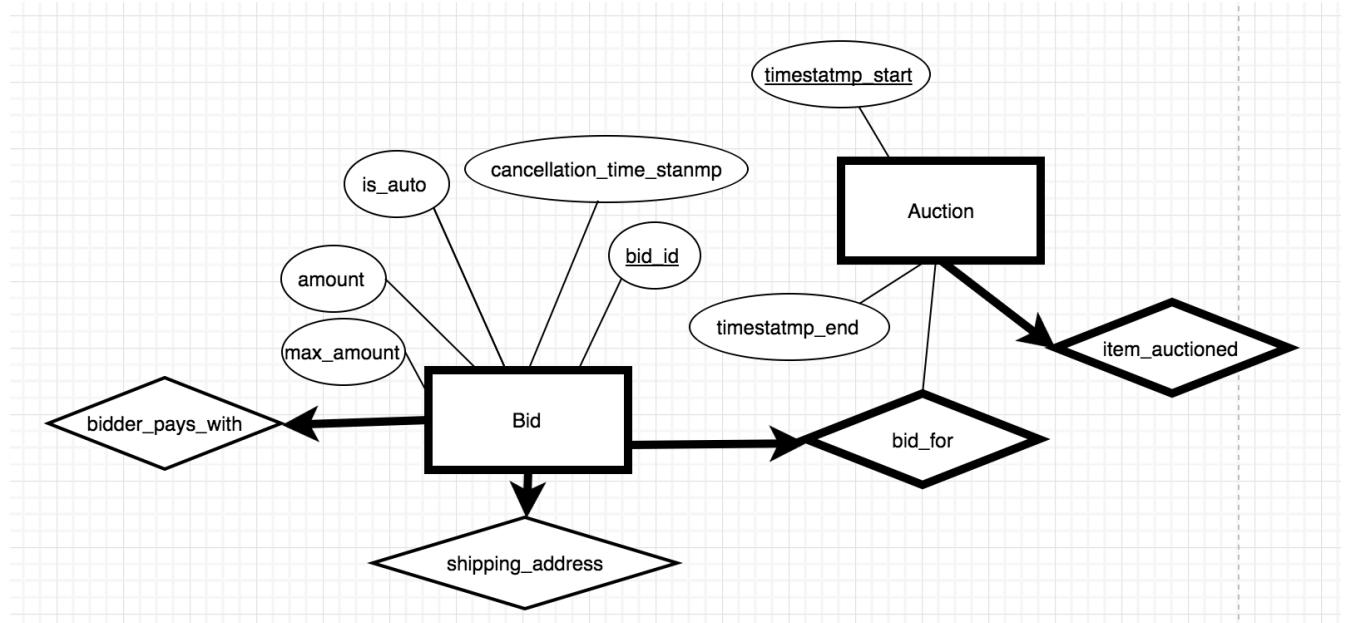


Figure 9: Auction and Bid ER Diagram

The table on the left was created for Auctions taking place, where it included timestamps for start and end. We referenced the *item_id* from the Sales_Items entity through the relationship of *item_auctioned*. Our partial key is *timestamp_start*, which is sourced from *item_id*. This could only be done using on delete cascade. *timestamp_end* refers to the time when the auction ends. Timestamps are recorded as integers based on the Unix machine time. This method is simpler than using traditional timestamps and yields the same performance.

For the Bid entity, we created a table called Bid, where we had *bid_id* as the primary key, *amount* as a decimal because it can not exceed more than 7-digit value max. *cancellation_timestamp*

holds the time when the auction is over. *username* lets us know who is the bidder. *is_auto* is a Boolean value that determines if automatic bidding feature is turned on. *max_amount* determines the highest bid before automatic bidding ceases. Our foreign key was *item_id* and *auction_timestamp_start*, which are referenced through Auction. Again we had to use an on delete cascade if the they are removed.

2.11. Order and Sale Reports

With everything set up, it is important to consider how our system keeps track of all the orders that go through the system. We want to be able to generate a weekly report for each category that we are selling. This will not only inform us on how we are doing in terms of revenue but also help us monitor consumer behavior. By recording and analyzing reports of user activity, companies can improve their sites functionality.^[9] Having this functionality in our application and design, we believe our marketing team will be able to take a full advantage of this report function and produce the most effective marketing strategy that will help to maximize company's profit.

Moving into the implementation of our report, each report will carry a specific report id that is relevant to the seller account detail and the time period for which the report is on. Our goal is to make our report as informative and dynamic as possible. We want to create an interface that allows our users to easily assess their data with a few clicks of a button.

Our design will include a bunch of useful keywords that relate to type of data that will be displayed such as total revenue, total cost, number of purchases, etc. In the backend of this functionality, we will use these keywords and put together a query that will access our database and pull up the request data and present them in a neat view.

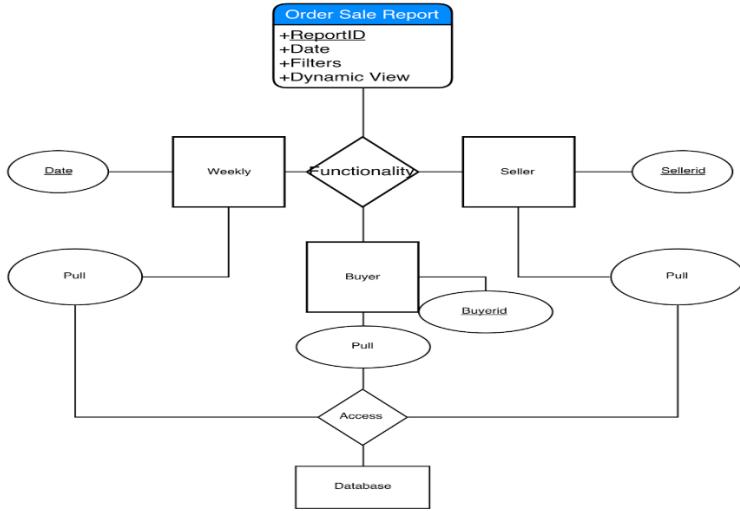


Figure 10: Order Sale Report Process Diagram

2.12. Delivery

The other important thing that we are focusing on is our delivery system. We understand that it is our duty to fulfill our customers' orders with the best service possible. This is why we are taking extra steps to secure every single transaction amongst our users.

With our delivery system, we will generate a unique id (*shipping_id*) for every delivery that is only available for the buyer, seller, and our engineers. Nevertheless, this id will only be created after a confirmation of payment has been posted. This means there will be lower chance of error for shipping confirmation. Why compete in the shipping market when companies like UPS, FedEx, and DHL have proven themselves to be among the best in the industry. While companies like UPS help to provide their customers with useful developer API, we aim to utilize their API to create the best interface for our users without traversing to another website.^[10] We want to provide the tracking number for the shipment to allow users to see the status of their shipment from our website. We hope that our efforts in providing a friendly user interface will keep our customers satisfied with our products and services.

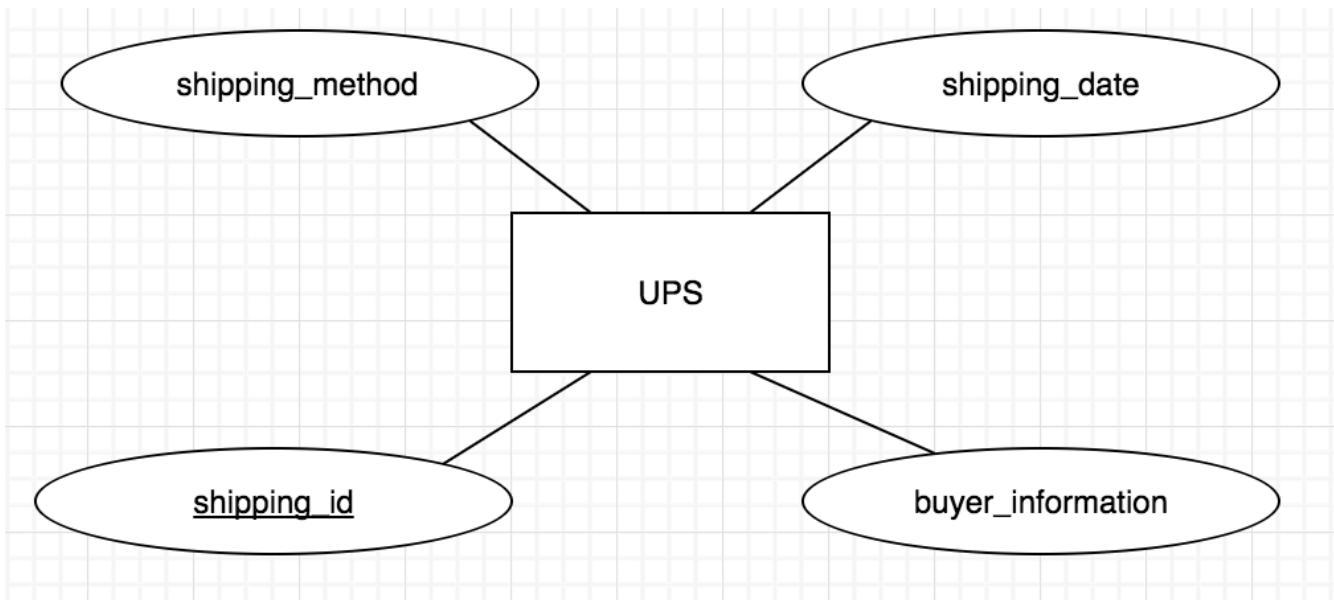


Figure 11: UPS ER Diagram

For the entity in the ER diagram of UPS, created a table called UPS, created tuples of its attributes. Used *shipping_id* as the primary key. We have an attribute that stores the method of shipping (*shipping_method*). Another attribute is used to record the date of shipping (*shipping_date*). For the foreign key, we used *transaction_id* because once a sale has been made *sale_info* relationship is accessed, where it gives the transaction id to the UPS service. *address_id* keeps track on where the package is being shipped to.

2.13. Custom Shoes

Unlike most sites, our team decided to create a custom shoe feature. We researched Nike's custom shoe system for guidance.^[6] First, the user will click on the custom shoe feature in the browser bar. Then, they can pick between a men's, women's, boys', or girls' shoe. After that, they can pick a particular category of shoe, such as athletic, casual, etc. and then pick a subcategory, such as running shoe or cleat. From there, the user has several features they can choose from a multiple choice list. These features include the size of the shoe, the width of the shoe, the side, back, and front image, tongue style, the shoe sole style, and the coloring of the shoe.

The way the custom shoe information is stored is in the **Custom_Shoe** entity. This entity is a sub-entity of the sales items but it contains additional information for the customization process such as a side, back, and front image labels. In our database, we will store every possible combination of custom shoes in the **Custom_Shoe** entity. Then, during the creation process, as the user selects new attributes, different images will be displayed on screen, based on the default and user selections.

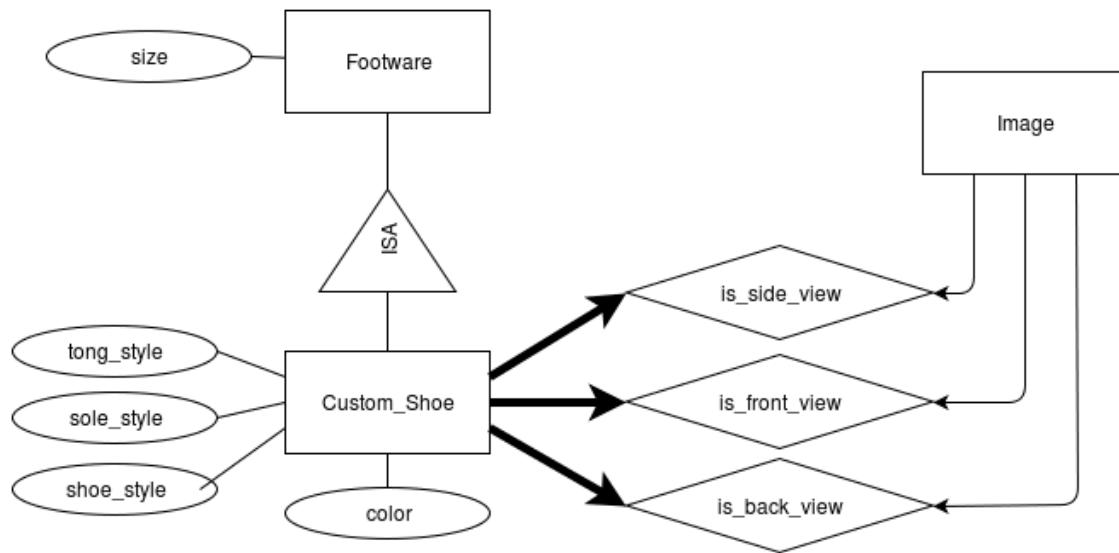


Figure 12: Custom_Shoe ER Diagram

We created a table based on the entity **Custom_Shoe**, each custom shoe you design will allow you to view the front, side, and back image. It will allow you to customize the tong, sole, shoe and color style, as shown in the ER diagram. Again the primary key is *item_id* because selecting a specific item allows you to customize it, where on delete cascade is used. The foreign keys include the relationships of *side_img_id*, *front_img_id*, and *back_img_id*, which reference the **Image** entity. We also have a foreign key of *item_id* which references **Footwear**.

2.14. Customer Service

We are providing users a way to contact us. This bonus feature allows us to connect us with the users. Users can email or call us. We also provide a FAQ page that helps users navigate better. This page also helps users get answers to common questions. Customer service options will be located at the bottom of every page. This feature also helps us get feedback about our services, which provides a scope for improvement.

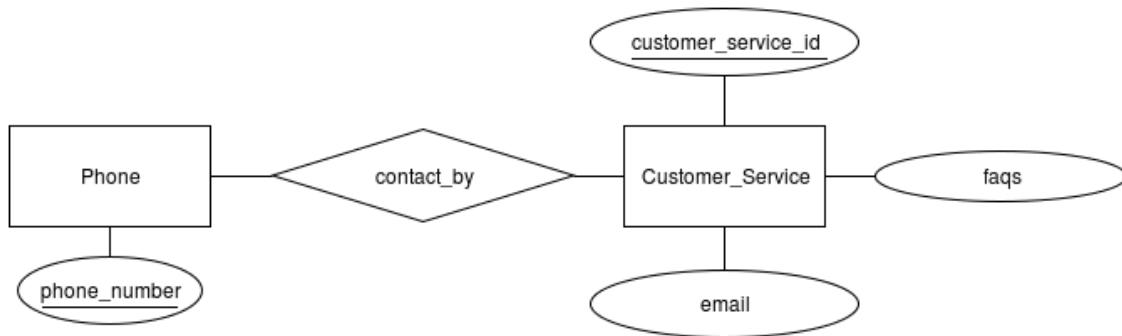
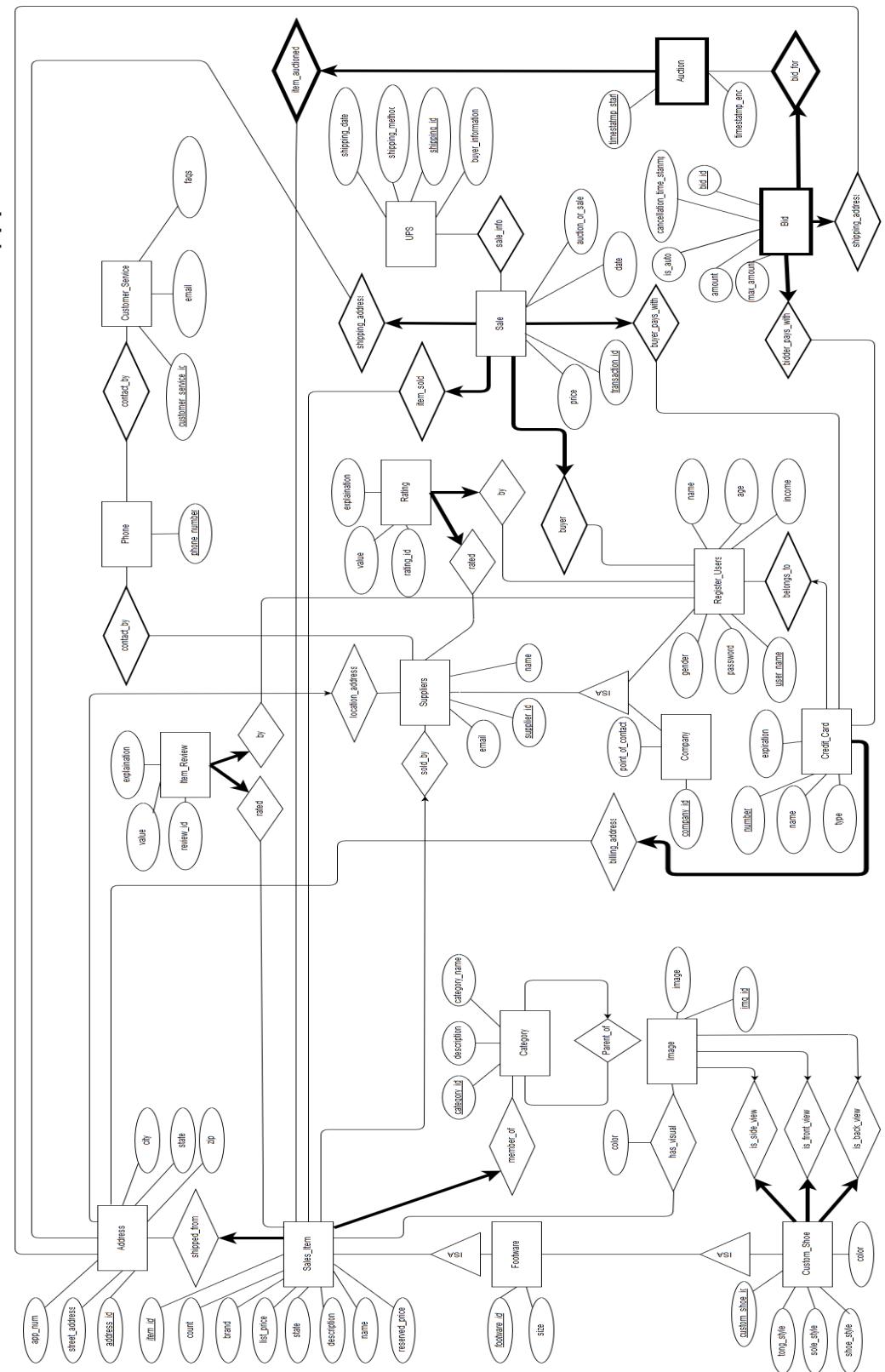


Figure 13: Customer_Service ER Diagram

We created the table Customer_Service as an extra feature. The primary key for it is *customer_service_id*. We created a tuple of *faqs* for customers to access frequent questions, we also included an *email*, and a *phone_number*. The foreign key for it is *phone_number*, where it is referenced from the entity Phone.

3. MASTER ER DIAGRAM



4. TECHNOLOGY SURVEY

4.1. MySQL

We selected MySQL as our database because it is a relational database, meaning that it is suited for storage and retrieval. The relational part of the database allows for an easier and wider range of access to data that will allow us to manipulate the data in our system to its full potential.

MySQL is easier for us to use, given our limited experiences in database language. Moreover, MySQL is also an open source software that is owned by Oracle, which means that our group will not have to spend any money to use it. Needless to say, MySQL is also very popular amongst programmers. There is a great online community that we can fall back on when we are in trouble.

We did take several limitations of MySQL into account including poor performance scaling, limited functionality, and lesser overall quality compared to other database languages.^[11] We decided that poor performance scaling would not be a big issue due to the fact that we are not operating with a ton of data. Next, we are also not anticipating a great number of functionality to implement inside our database, as we will have backend codes that will do the job. Our intended use of the database should be straightforward – push and pull requests. Lastly, quality and stability is not something we are currently targeting for in our prototype phase. Our goal to set up the environment first and make everything work before we optimize to our environment.

Pro	Cons
Easy to use	Poor performance scaling
Open source	Limited functionality
Great online community	Inferior quality to top-tier databases
Suitable performance	Stability issues

Figure 14: Pros and Cons of MySQL

4.2. Frontend

Since our team has the most experiences in JavaScript, we decided to design our front end in JavaScript, HTML 5, and CSS. We chose JavaScript because it remains one of today's most used languages in websites. It is easier to code than server-side languages, and it offers a great range of variety when it comes to dynamic views and displays for our users. Moreover, JavaScript is entirely a client side programming, meaning the computation is done on our users' computer rather than having data sent back to the server and back out to the user. We rely on this mechanism to make it easier for our servers, so we can reach up to more users.

HTML 5 and CSS will be used to make our website more visibly attractive to our users. We hope that our front-end code will be able to take a full advantage of our design to create the best user experiences possible. Nevertheless, these two languages are easy to learn and implement since they are ubiquitous in today's websites.

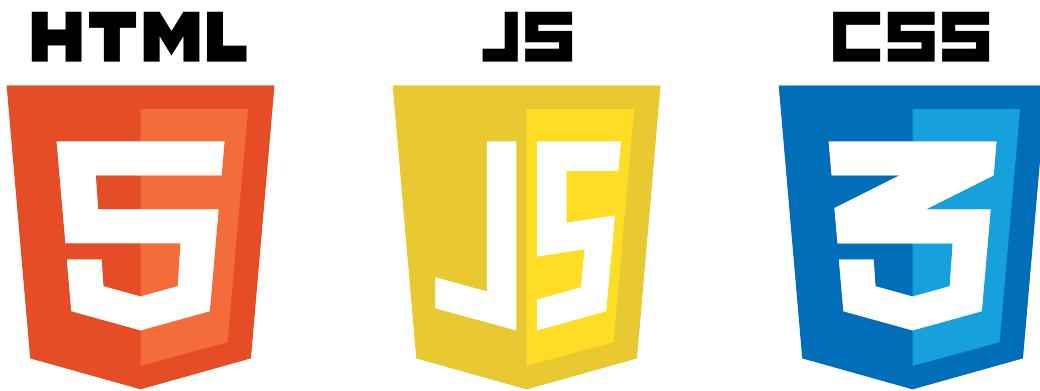


Figure 15: Frontend Programming Languages

4.3. Backend

We want to link our frontend code to our database through a backend written in Java. Mainly because Java is still one of the most popular languages in the industry and many of us have experience

with it. Java offers a wide range of functionality and online communities to help us achieve what we want.

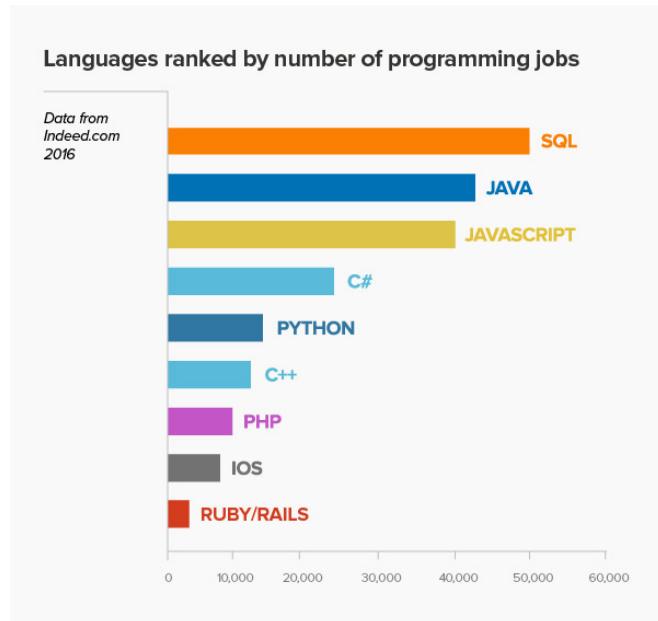


Figure 16: Most Used Programming Languages

4.4. Other Technologies

Draw.io – We used Draw.io to do most of our Entity-Relationship diagrams. It is an easy to use online program that allows us to organize our diagrams neatly.

Asana – We use Asana to plan most of our daily activities and group goals. It offers a clear view of what our goals are and how to get there, if we put in the work to create it.

GitHub – GitHub is where our repository is located. We have access to our peers' work and to the progression of our big project. Moreover, it is very convenient for each of us to edit and make commentary about others work.

GroupMe – We use GroupMe to communicate with one another.

5. CATEGORY TREE AND NORMALIZATION

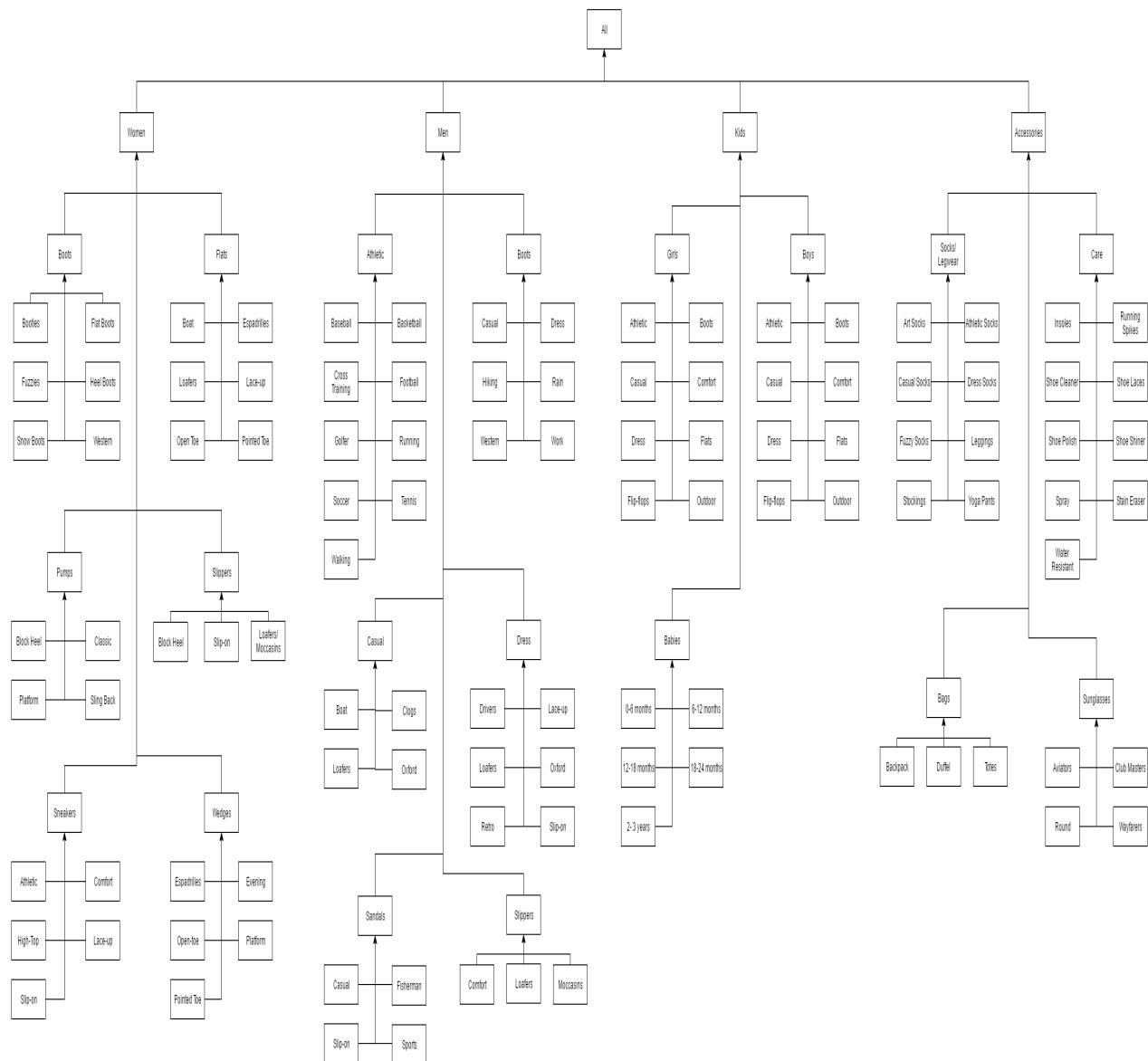


Figure 17: Classification Diagram

SCHEMAS IN THIRD NORMAL FORM

address

<u>address_id:</u> integer	app_num: string	street_address: string	city: string	state: string	zip: integer	supplier_id: integer
-------------------------------	--------------------	---------------------------	-----------------	------------------	-----------------	-------------------------

auction

timestamp_start: timestamp	timestamp_end: timestamp	<u>item_id:</u> integer
-------------------------------	-----------------------------	----------------------------

bid

<u>bid_id:</u> integer	amount: integer	cancellation_timestamp: timestamp	<u>item_id:</u> integer	<u>auction_timestamp_start:</u> timestamp	credit_card_number: integer	shipping_address_id: integer	is_auto: Boolean	Max_amount: real
---------------------------	--------------------	--------------------------------------	----------------------------	--	--------------------------------	---------------------------------	---------------------	---------------------

category

<u>category_id:</u> integer	description: string	category_name: string	parent_id: integer
--------------------------------	------------------------	--------------------------	-----------------------

company

point_of_contact: string	<u>supplier_id:</u> integer
-----------------------------	--------------------------------

contact_by

<u>phone_number:</u> integer	<u>supplier_id:</u> integer
---------------------------------	--------------------------------

credit_card

<u>number:</u> integer	name: string	type: string	expiration: timestamp	username: string	<u>billing_address_id:</u> integer
---------------------------	-----------------	-----------------	--------------------------	---------------------	---------------------------------------

<u>custom_shoe</u>	tong_style: string	sole_style: string	shoe_style: string	color: string	side_img_id: integer	front_img_id: integer	back_img_id: integer	item_id: integer
--------------------	--------------------	--------------------	--------------------	---------------	----------------------	-----------------------	----------------------	------------------

<u>customer_service</u>	customer_service_id: integer	email: string	faqs: string	phone_number: integer
-------------------------	------------------------------	---------------	--------------	-----------------------

<u>footwear</u>	size: string	item_id: integer
-----------------	--------------	------------------

<u>has_visual</u>	item_id: integer	img_id: integer	color: string
-------------------	------------------	-----------------	---------------

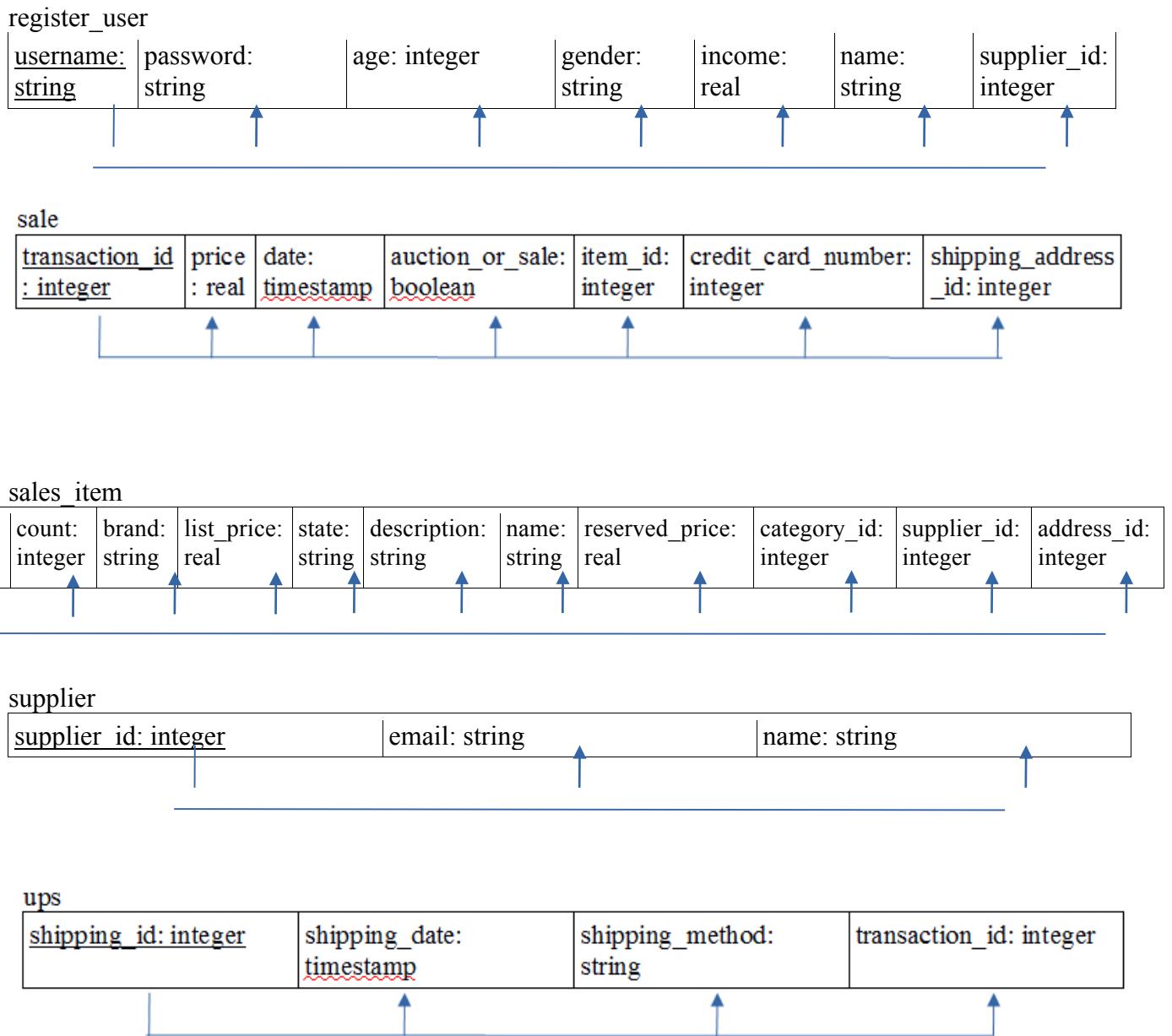
<u>image</u>	img_id: integer	image: blob
--------------	-----------------	-------------

<u>pays_with</u>	username: string	number: integer	address_id: integer
------------------	------------------	-----------------	---------------------

<u>phone</u>	phone_number: integer
--------------	-----------------------

<u>rating</u>	rating_id: integer	explanation: string	value: real	username: string	supplier_id: integer
---------------	--------------------	---------------------	-------------	------------------	----------------------

<u>item_review</u>	review_id: integer	explanation: string	value: real	username: string	item_id: integer
--------------------	--------------------	---------------------	-------------	------------------	------------------



6. SQL STATEMENTS

6.1. Sales Item

```
CREATE TABLE Sales_Item
( item_id      INTEGER,
  count        INTEGER,
  brand        CHAR(40),
  list_price   DECIMAL(7,2),
  state        CHAR(20),
  description  VARCHAR(999),
  name         CHAR(80),
  reserved_price DECIMAL(7,2),
  category_id  INTEGER NOT NULL,
  supplier_id  INTEGER,
  address_id   INTEGER NOT NULL,
  PRIMARY KEY (item_id),
  FOREIGN KEY (category_id) REFERENCES Category (category_id),
  FOREIGN KEY (supplier_id) REFERENCES Suppliers (supplier_id),
  FOREIGN KEY (address_id) REFERENCES Address (address_id)
);
```

Figure 18: SQL Table for Sales_Item

```
CREATE TABLE Footwear
( size CHAR(40),
  item_id INTEGER,
  PRIMARY KEY (item_id),
  FOREIGN KEY (item_id) REFERENCES Sales_Item (item_id)
  ON DELETE CASCADE
);
```

Figure 19: SQL Table for Footwear

```
CREATE TABLE Address
( address_id    INTEGER,
  app_num       CHAR(20),
  street_address CHAR(40),
  city          CHAR(20),
  state         CHAR(20),
  zip           INTEGER,
  supplier_id   INTEGER,
  PRIMARY KEY    (address_id),
  FOREIGN KEY    (supplier_id) REFERENCES Suppliers (supplier_id)
);
```

Figure 20: SQL Table for Address

```
CREATE TABLE has_visual
( item_id INTEGER,
  img_id INTEGER,
  color CHAR(80),
  PRIMARY KEY (item_id, img_id),
  FOREIGN KEY (item_id) REFERENCES Sales_Item (item_id),
  FOREIGN KEY (img_id) REFERENCES Image (img_id)
);
```

Figure 21: SQL Table for has_visual

```
CREATE TABLE Image
( img_id INTEGER,
  image CHAR(128),
  PRIMARY KEY (img_id)
);
```

Figure 22: SQL Table for Image

6.2. Categories

```
CREATE TABLE Category
( category_id INTEGER,
  description CHAR(128),
  category_name CHAR(40),
  parent_id INTEGER,
  PRIMARY KEY (category_id),
  FOREIGN KEY (parent_id) REFERENCES Category (category_id)
);
```

Figure 23: SQL Table for Category

6.3. Suppliers

```
CREATE TABLE Suppliers
( supplier_id INTEGER,
  email      CHAR(40),
  name       CHAR(40),
  PRIMARY KEY (supplier_id)
);
```

Figure 24: SQL Table for Supplier

```
CREATE TABLE Phone
( phone_number BIGINT,
  PRIMARY KEY (phone_number)
);
```

Figure 25: SQL Table for Phone

```
CREATE TABLE contact_by
( phone_number BIGINT,
  supplier_id INTEGER,
  PRIMARY KEY (phone_number, supplier_id),
  FOREIGN KEY (phone_number) REFERENCES Phone (phone_number),
  FOREIGN KEY (supplier_id) REFERENCES Suppliers (supplier_id)
);
```

Figure 26: SQL Table for contact_by

6.4. Company

```
CREATE TABLE Company
( point_of_contact CHAR(40),
  supplier_id INTEGER,
  PRIMARY KEY (supplier_id),
  FOREIGN KEY (supplier_id) REFERENCES Suppliers (supplier_id)
  ON DELETE CASCADE
);
```

Figure 27: SQL Table for Company

6.5. Registered Users

```
CREATE TABLE Register_Users
( username    CHAR(40),
  password    VARCHAR(40),
  age         INTEGER,
  gender      CHAR(10),
  income      REAL,
  supplier_id INTEGER NOT NULL UNIQUE,
  PRIMARY KEY (username),
  FOREIGN KEY (supplier_id) REFERENCES Suppliers (supplier_id)
  ON DELETE CASCADE
);
```

Figure 28: SQL Table for Register_Users

```
CREATE TABLE Credit_Card
( number BIGINT,
  name CHAR(40),
  type CHAR(20),
  expiration DATE,
  username CHAR(40),
  billing_address_id INTEGER NOT NULL,
  PRIMARY KEY (number),
  FOREIGN KEY (username) REFERENCES Register_Users (username),
  FOREIGN KEY (billing_address_id) REFERENCES Address (address_id)
);
```

Figure 29: SQL Table for Credit_Card

6.6. Ratings

```
CREATE TABLE Rating
( rating_id INTEGER,
  explanation CHAR(128),
  value REAL,
  username CHAR(40),
  supplier_id INTEGER,
  PRIMARY KEY (supplier_id, rating_id),
  FOREIGN KEY (USERNAME) REFERENCES Register_Users (username),
  FOREIGN KEY (supplier_id) REFERENCES Suppliers (supplier_id)
  ON DELETE CASCADE
);
```

Figure 30: SQL Table for Ratings

```
CREATE TABLE Item_Review
( review_id INTEGER,
  explanation CHAR(128),
  value REAL,
  username CHAR(40),
  item_id INTEGER,
  PRIMARY KEY (item_id, review_id),
  FOREIGN KEY (USERNAME) REFERENCES Register_Users (username),
  FOREIGN KEY (item_id) REFERENCES Sales_Item (item_id)
  ON DELETE CASCADE
);
```

Figure 31: SQL Table for Item Review

6.7. Sale

```
CREATE TABLE Sale
( transaction_id INTEGER,
  price DECIMAL(7,2),
  date BIGINT,
  auction_or_sale BOOL,
  item_id INTEGER NOT NULL,
  credit_card_number BIGINT NOT NULL,
  shipping_address_id INTEGER NOT NULL,
  PRIMARY KEY (transaction_id),
  FOREIGN KEY (item_id) REFERENCES Sales_Item (item_id),
  FOREIGN KEY (credit_card_number) REFERENCES Credit_Card (number),
  FOREIGN KEY (shipping_address_id) REFERENCES Address (address_id)
);
```

Figure 32: SQL Table for Sale

6.8. Bidding

```
CREATE TABLE Auction
( timestamp_start BIGINT,
  timestamp_end   BIGINT,
  item_id         INTEGER,
  PRIMARY KEY (timestamp_start, item_id),
  FOREIGN KEY (item_id) REFERENCES Sales_Item (item_id)
  ON DELETE CASCADE
);
```

Figure 33: SQL Table for Auction

```

CREATE TABLE Bid
( bid_id          INTEGER,
  amount          DECIMAL(7,2),
  cancellation_timestamp BIGINT,
  item_id         INTEGER NOT NULL,
  auction_timestamp_start BIGINT,
  credit_card_number   BIGINT NOT NULL,
  shipping_address_id  INTEGER NOT NULL,
  is_auto          BOOLEAN NOT NULL,
  max_amount        DECIMAL(7,2),
  PRIMARY KEY (bid_id, item_id, auction_timestamp_start),
  FOREIGN KEY (item_id, auction_timestamp_start) REFERENCES Auction (item_id,
  timestamp_start)
  ON DELETE CASCADE,
  FOREIGN KEY (credit_card_number) REFERENCES Credit_Card (number),
  FOREIGN KEY (shipping_address_id) REFERENCES Address (address_id)
);

```

Figure 34: SQL Table for Bid

6.9. Delivery

```

CREATE TABLE UPS
( shipping_id INTEGER,
  shipping_date BIGINT,
  shipping_method CHAR(20),
  transaction_id INTEGER,
  PRIMARY KEY (shipping_id),
  FOREIGN KEY (transaction_id) REFERENCES Sale (transaction_id)
);

```

Figure 35: SQL Table for UPS

6.10. Custom Shoes

```

CREATE TABLE Custom_Shoe
( tong_style    CHAR(20),
  sole_style    CHAR(20),
  shoe_style    CHAR(20),
  color          CHAR(80),
  side_img_id   INTEGER NOT NULL,
  front_img_id  INTEGER NOT NULL,
  back_img_id   INTEGER NOT NULL,
  item_id        INTEGER,
  PRIMARY KEY (item_id),
  FOREIGN KEY (side_img_id) REFERENCES Image (img_id),
  FOREIGN KEY (front_img_id) REFERENCES Image (img_id),
  FOREIGN KEY (back_img_id) REFERENCES Image (img_id),
  FOREIGN KEY (item_id) REFERENCES Footwear (item_id)
  ON DELETE CASCADE
);

```

Figure 36: SQL Table for Custom_Shoe

6.11. Customer Service

```
CREATE TABLE Customer_Service
( customer_service_id INTEGER,
  email CHAR(40),
  faqs CHAR(128),
  phone_number BIGINT,
  PRIMARY KEY (customer_service_id),
  FOREIGN KEY (phone_number) REFERENCES Phone (phone_number)
);
```

Figure 37: SQL Table for Customer_Service

7. CONCLUSION

There were many options to choose from when selecting a database. After our first meeting we gathered ideas from everyone and ended with a shoe retailers' database. The shoe industry is enormous and there were many features that we wanted to talk about besides the given ones. Everyone did their research on ideas and how we would incorporate the given features. The bonus features we selected were custom shoes and customer service. We assigned each person separate features, where each would do research on their specific topic and write a rough draft for analysis, we discussed how each person should implement certain ideas pertaining to their feature.

Next, we broke up into 2 teams, we had 3 on editing, while the rest on presentation. The biggest challenge we faced was making sure the information lined up perfectly between each topic because some features are closely connected to each other. Our main goal was making sure we developed a database that was user-friendly, yet offered complex features. Overall, phase one was a success where we implemented the design specifications according to the criteria.

As we move forward with our design we will implement a website that represents our database in a consumer friendly manner. Our website will allow the user to quickly locate desired items and purchase safely within our database on a protected web server. To do this we will apply browsing and searching techniques using the Breadth-First Search algorithm. Another addition will be Order and Sales Report, where we will store the transaction information in our database for business purposes. Most importantly, the database will be ready for real-world use.

8. FINAL SYSTEM FRAMEWORK

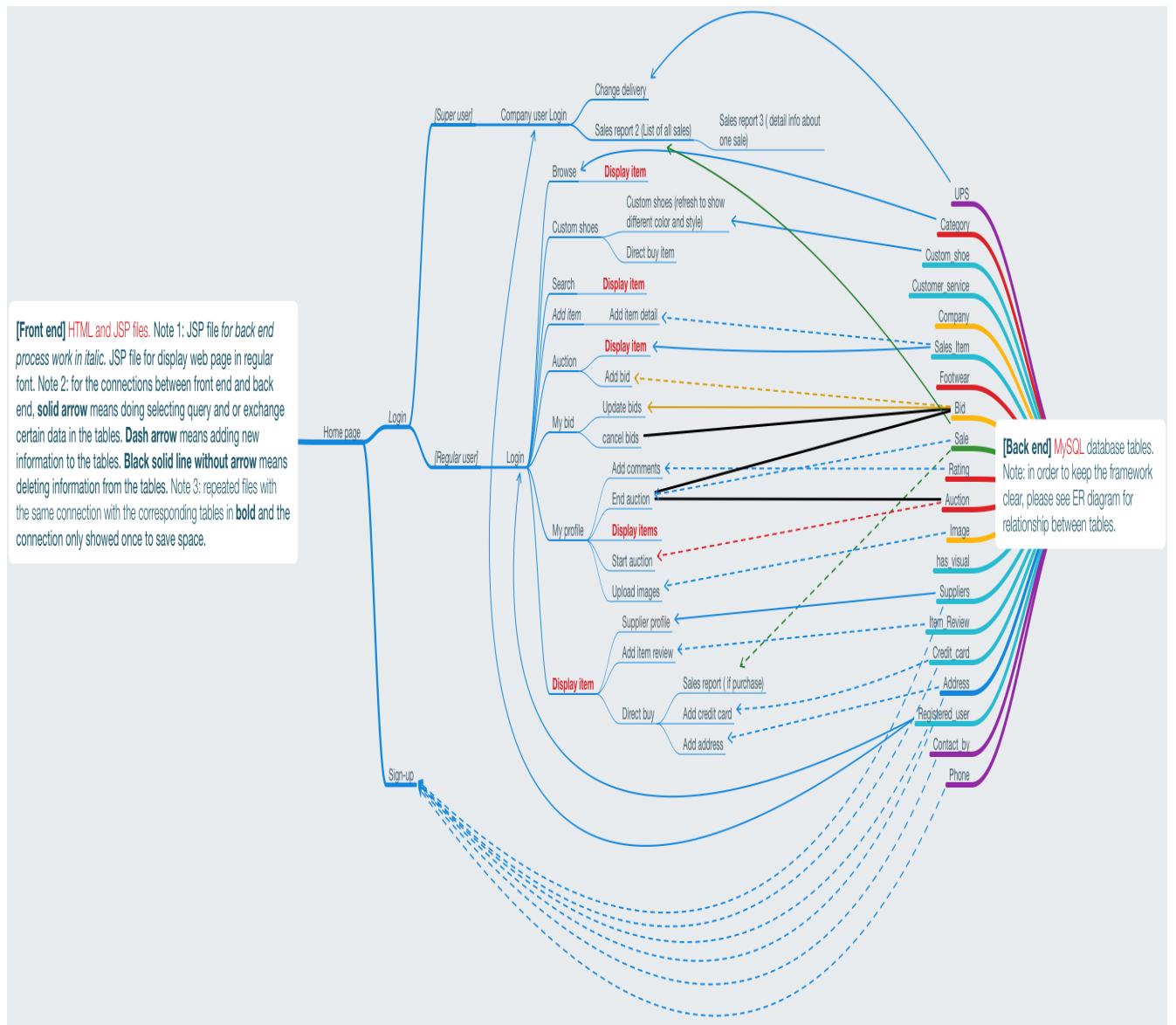


Figure 38: Framework

The framework above shows main relationship between the frontend webpages and backend database.

Besides the notes showed in the graph, a few essential explanations are as follows:

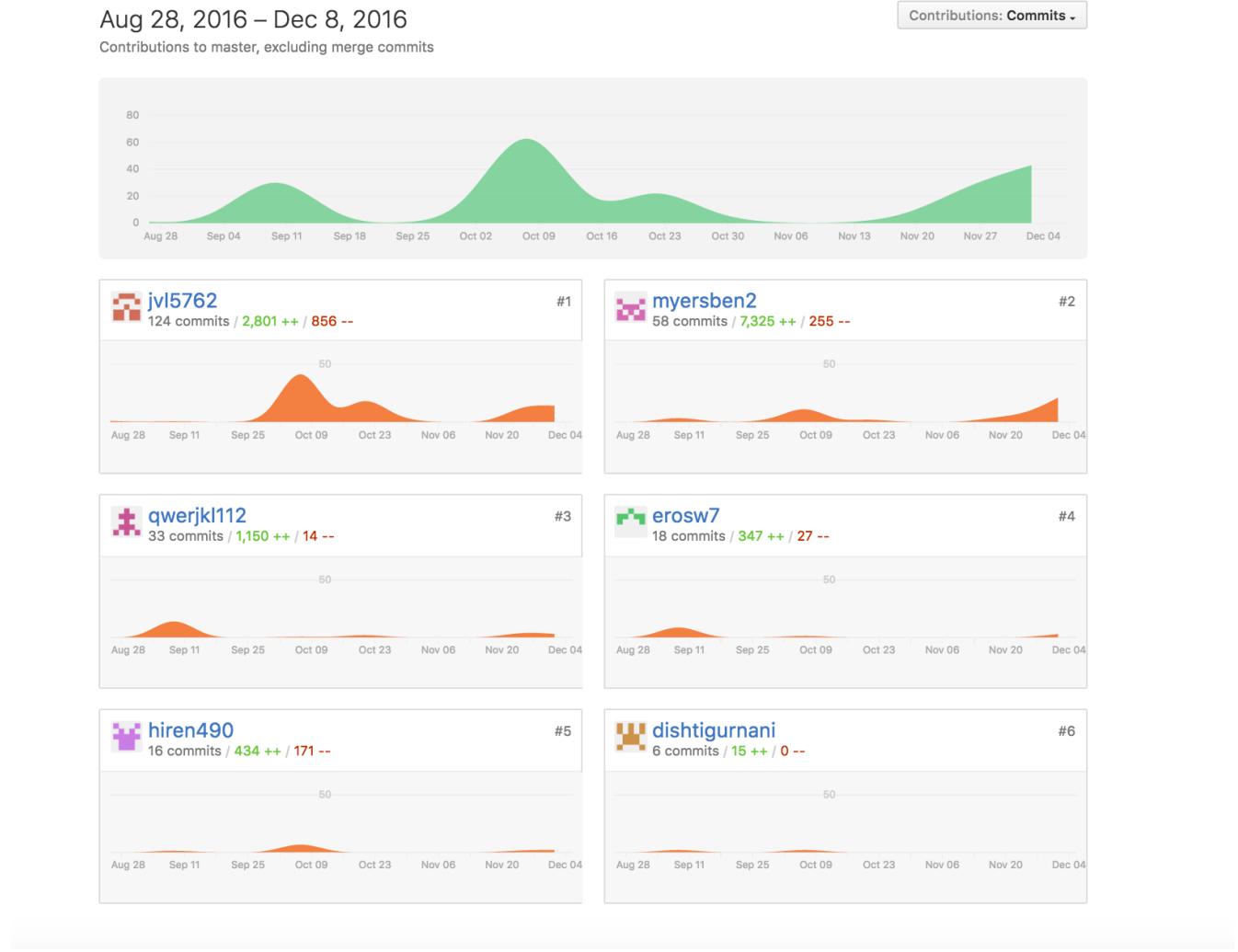
1. Login is divided into super users and regular users' login, where super users can change delivery and view sales reports. Regular users are the customers. After login, they can browse through different categories, add items, place bids for auctions and directly buy items, etc.

2. For auction, users can find a list of items under Auction page. Once they bid on one item, they can trace their bid in My bid. In My bid page, users can either update their bid value or delete existing bid.
3. Under My profile, users can start an auction, and they can choose to end an auction as well. End auction will also delete all the bids tied to the auction item and the auction will be deleted from Auction page. This transaction will be added to Sale table. Thus End Auction has two black solid lines and one dash line as its connection.

9. REFLECTION

Looking back, we set really high expectations for ourselves and wanted to create a shoe enterprise that reflected big corporations like Macys and eBay. Our main goal was to bring the features offered by them as well as our own, but at the same time creating a consumer-friendly environment. Our main features included: Sales Items, Categories, Suppliers, Company, Registered Users, Rating, Browsing, Searching, Sale, Bidding, Sales Reports, Delivery, Custom Shoes, and Customer Service. We ended up achieving all of them except Customer Service and had an issue with Register Users. We didn't implement Customer Service, the main reason was due to the lack of time and conflicts with other features. Another feature that we implemented was Registered Users, but during the presentation it failed, possible reasons for the bug due to duplication of data or the email confirmation was not sent properly. A sub-feature that we added was rating items, initially, we set out to just rate users, but ended up being a nice feature. Another feature that we didn't have in mind when we were setting goals was images. It ended up being a really nice feature for all the items, especially custom shoes. For the most, we were successful in hitting the goals we set, but some fell out due to the reasons listed above. Even though we hit a majority of the goals, all we can say is setting goals is easy, but achieving them not so much.

10. APPENDIX A – GITHUB GRAPHS



11. APPENDIX B – ASANA SUMMARY

Task ID	Created At	Completed At	Last Modified	Name	Assignee	Due Date	Tags	Notes	Projects	Parent Task
2.EE+14	12/5/16	12/8/16	12/8/16	Ben, Jeff, Frank Tasks				UPS - Jeff	Cmpsc 431 Project	
2.EE+14	11/29/16	12/8/16	12/8/16	All Backend tasks		11/26/16		Add User	Cmpsc 431 Project	
2.EE+14	11/29/16		11/29/16	Ben's tasks	Benjamin Myers			List items		All Backend tasks
2.EE+14	11/29/16		12/5/16	Jeff's tasks	Jeff Li			Add User		All Backend tasks
2.EE+14	11/29/16		11/29/16	Hiren's task	Hiren Patel			search		All Backend tasks
2.IE+14	11/16/16	12/8/16	12/8/16	Project Assignment				Direct buy	Cmpsc 431 Project	
2.EE+14	12/8/16	12/8/16	12/8/16	Phase 3 task Jeff Li				Ben -	Cmpsc 431 Project	
2.EE+14	12/8/16	12/8/16	12/8/16	Phase 3 task Dishi Gurnani				Ben -	Cmpsc 431 Project	
2.EE+14	12/8/16	12/8/16	12/8/16	Phase 3 task Eros Wang				Ben -	Cmpsc 431 Project	
2.EE+14	12/8/16	12/8/16	12/8/16	Phase 3 task Frank Guo				Ben -	Cmpsc 431 Project	
2.EE+14	12/8/16	12/8/16	12/8/16	Phase 3 task Hiren Patel				Ben -	Cmpsc 431 Project	
2.EE+14	12/8/16	12/8/16	12/8/16	Phase 3 task Benjamin Myers				Ben -	Cmpsc 431 Project	
2.EE+14	11/29/16		11/29/16							Cmpsc 431 Project
2.IE+14	11/16/16	11/29/16	11/29/16	Project Assi Eros Wang				Direct buy	Cmpsc 431 Project	
2.IE+14	11/16/16	11/29/16	11/29/16	Project Assi Hiren Patel				Direct buy	Cmpsc 431 Project	
2.IE+14	11/16/16	11/29/16	11/29/16	Project Assi Jeff Li				Direct buy	Cmpsc 431 Project	
2.IE+14	11/16/16	11/29/16	11/29/16	Project Assi Frank Guo				Direct buy	Cmpsc 431 Project	
2.IE+14	11/16/16	11/29/16	11/29/16	Project Assi Benjamin Myers				Direct buy	Cmpsc 431 Project	
2E+14	10/25/16	11/29/16	11/29/16	Frontend People				Download	Cmpsc 431 Project	
2E+14	10/19/16	10/24/16	10/24/16	Add line to Benjamin M	Benjamin Myers	10/20/16			Cmpsc 431 Project	
2E+14	10/19/16	10/24/16	10/24/16	Making mas Benjamin M	Benjamin Myers	10/22/16			Cmpsc 431 Project	
2E+14	10/19/16	11/29/16	11/29/16	Creating ini Eros Wang	Eros Wang	10/25/16			Cmpsc 431 Project	
2E+14	10/19/16	11/29/16	11/29/16	Creating ini Dishi Gurnani	Dishi Gurnani	10/22/16			Cmpsc 431 Project	
2E+14	10/19/16	11/29/16	11/29/16	Creating sql Jeff Li	Jeff Li	10/22/16			Cmpsc 431 Project	
2E+14	10/19/16	11/29/16	11/29/16	Creating sql Hiren Patel	Hiren Patel	10/22/16			Cmpsc 431 Project	
2E+14	10/19/16	11/29/16	11/29/16	Creating sql Benjamin Myers	Benjamin Myers	10/22/16			Cmpsc 431 Project	
2E+14	10/19/16	11/29/16	11/29/16	DATAbase				package	Cmpsc 431 Project	
1.9E+14	10/6/16	10/12/16	10/12/16	Second Meeting of Phase Two				Perfecting	Cmpsc 431 Project	
1.9E+14	10/3/16	10/6/16	10/6/16	First Meeting Phase 2		10/3/16			Cmpsc 431 Project	
1.9E+14	10/3/16	10/12/16	10/12/16	Technology Frank Guo	Frank Guo	10/10/16			Cmpsc 431 Project	
1.9E+14	10/3/16	10/11/16	10/11/16	Make the R	Benjamin Myers	10/6/16			Cmpsc 431 Project	
1.9E+14	10/3/16	10/3/16	10/3/16	Jeff Li						Make the Relation Schema
1.9E+14	10/3/16	10/3/16	10/3/16	Hiren Patel						Make the Relation Schema
1.9E+14	10/3/16	10/3/16	10/3/16	Frank Guo						Make the Relation Schema
1.9E+14	10/13/16	10/13/16	10/13/16	Create Phas Frank Guo	Frank Guo	10/13/16			Cmpsc 431 Project	
1.9E+14	10/12/16	10/12/16	10/12/16	Normalize/I Benjamin M	Benjamin Myers	10/10/16			Cmpsc 431 Project	
1.9E+14	10/3/16	10/3/16	10/3/16	Jeff Li						Normalize/Fix ER Diagram
1.9E+14	10/3/16	10/3/16	10/3/16	Benjamin Myers						Normalize/Fix ER Diagram
1.9E+14	10/3/16	10/3/16	10/3/16	Hiren Patel						Normalize/Fix ER Diagram
1.9E+14	10/13/16	10/13/16	10/13/16	Fix Up Writ Jeff Li	Jeff Li	10/12/16			Cmpsc 431 Project	
1.9E+14	10/3/16	10/3/16	10/3/16	Frank Guo	Frank Guo					Fix Up Writing in Phase 1
1.9E+14	10/3/16	10/3/16	10/3/16	Hiren Patel	Hiren Patel					Fix Up Writing in Phase 1
1.9E+14	10/3/16	10/3/16	10/3/16	Benjamin Myers	Benjamin Myers					Fix Up Writing in Phase 1
1.9E+14	10/12/16	10/12/16	10/12/16	Create Data Benjamin M	Benjamin Myers	10/11/16			Cmpsc 431 Project	
1.9E+14	10/3/16	10/12/16	10/12/16	Find items t Dishi Gurnani	Dishi Gurnani	10/12/16			Cmpsc 431 Project	
1.9E+14	10/3/16	10/12/16	10/12/16	Find items t Eros Wang	Eros Wang	10/12/16			Cmpsc 431 Project	
1.9E+14	10/3/16	10/12/16	10/12/16	Add items s Benjamin M	Benjamin Myers	10/12/16			Cmpsc 431 Project	
2E+14	10/12/16	10/13/16	10/13/16	Fix Up Writ Hiren Patel	Hiren Patel	10/12/16			Cmpsc 431 Project	
2E+14	10/12/16	10/13/16	10/13/16	Create Imag Benjamin M	Benjamin Myers	10/13/16			Cmpsc 431 Project	
2E+14	10/12/16	10/12/16	10/12/16	Normalize/I Hiren Patel	Hiren Patel	10/10/16			Cmpsc 431 Project	
2E+14	10/12/16	10/12/16	10/12/16	Normalize/I Jeff Li	Jeff Li	10/10/16			Cmpsc 431 Project	
1.8E+14	9/15/16	9/15/16	9/15/16	ER Diagram Dishi Gurnani						Cmpsc 431 Project
1.8E+14	9/15/16	9/15/16	9/15/16	Citations Eros Wang	Eros Wang					Cmpsc 431 Project
1.8E+14	9/15/16	9/15/16	9/15/16	Formatting Benjamin Myers	Benjamin Myers					Cmpsc 431 Project
1.8E+14	9/15/16	9/15/16	9/15/16	Final Revie Hiren Patel	Hiren Patel					Cmpsc 431 Project
1.8E+14	9/15/16	9/20/16	9/20/16	Final Revie Frank Guo	Frank Guo					Cmpsc 431 Project
1.8E+14	9/15/16	9/15/16	9/15/16	Editing Tea Hiren Patel	Hiren Patel	9/14/16				Cmpsc 431 Project
1.8E+14	9/15/16	9/15/16	9/15/16	Editing Tea Benjamin M	Benjamin Myers	9/14/16				Cmpsc 431 Project
1.8E+14	9/15/16	9/15/16	9/15/16	Editing Tea Dishi Gurnani	Dishi Gurnani	9/14/16				Cmpsc 431 Project
1.8E+14	9/15/16	9/15/16	9/15/16	Google slide Frank Guo	Frank Guo	9/15/16				https://docs.Cmpsc 431 Project
1.8E+14	9/13/16	9/15/16	9/15/16	Fourth Team Meeting				Presenting	Cmpsc 431 Project	
1.8E+14	9/20/16	9/20/16	9/20/16	Google Doc Frank Guo	Frank Guo	9/15/16				https://docs.Cmpsc 431 Project
1.8E+14	9/10/16	9/14/16	9/14/16	Individual A Benjamin M	Benjamin Myers	9/13/16				Sale Items, Cmpsc 431 Project
1.8E+14	9/10/16	9/15/16	9/15/16	Individual A Dishi Gurnani	Dishi Gurnani	9/13/16				Suppliers, R Cmpsc 431 Project
1.8E+14	9/10/16	9/15/16	9/15/16	Individual A Eros Wang	Eros Wang	9/13/16				Rating, Sea Cmpsc 431 Project
1.8E+14	9/10/16	9/15/16	9/15/16	Individual A Hiren Patel	Hiren Patel	9/13/16				Sale Bidding, Cmpsc 431 Project
1.8E+14	9/10/16	9/15/16	9/15/16	Individual A Frank Guo	Frank Guo	9/13/16				OSR, Deliv Cmpsc 431 Project
1.8E+14	9/10/16	9/15/16	9/15/16	Requirements on Individual Analysis						Hey guys, Cmpsc 431 Project
1.8E+14	9/10/16	9/15/16	9/15/16	Individual A Jeff Li	Jeff Li					Categories, Cmpsc 431 Project
1.8E+14	9/10/16	9/15/16	9/15/16	Third Team Meeting		9/10/16		Assign		Cmpsc 431 Project
1.8E+14	9/8/16	9/15/16	9/15/16	Classification Tree						Cmpsc 431 Project
1.8E+14	9/6/16	9/15/16	9/15/16	Second Tea Benjamin M	Benjamin Myers	9/8/16				6:00 pm Cmpsc 431 Project
1.8E+14	9/6/16	9/15/16	9/15/16	Conceptual Ideas Phase		9/6/16				Starts at 6:0 Cmpsc 431 Project
1.8E+14	9/6/16	9/15/16	9/15/16	First Team ! Benjamin M	Benjamin Myers	9/6/16				Cmpsc 431 Project
1.7E+14	8/29/16	10/3/16	10/3/16	Submit Written Proposal		9/16/16				Cmpsc 431 Project
1.7E+14	8/29/16	10/3/16	10/3/16	Present Proposal		9/16/16				Cmpsc 431 Project

12. APPENDIX F – DATABASE STORAGE

After completing our ER and schema designs for the important information we needed to store, we created our database and began filling it with sample data. Our goal was to fill out enough information to make the site feel like it was already up and running with real users. Below lists all the data stored in each table of our database, in alphabetical order. Note that some fields will be missing, due to difficulty of fitting all of the information on the page.

Table 1: Address Table

address_id	app_num	street_address	city	state	zip	supplier_id	
1		54 Franklin Ave.	Manahawkin	NJ	8050	1	
2		14 Acacia Dr.	Glendora	CA	91740	2	
3		8077 University St.	Lakeville	MN	55044	3	
4		7976 West St.	Grovetown	GA	30813	4	
5		1 SW. Buttonwood Avenue	Kalispell	MT	59901	5	
6		9714 Miles Lane	Johnston	RI	2919	6	
7		255 Longbranch Lane	Ellicott City	MD	21042	7	
8		7444 W. County Lane	Brentwood	NY	11717	8	
9		368 Garfield Lane	Mount Airy	MD	21771	9	
10		8252 Country Drive	Freeport	NY	11520	10	
11		112 Wild Rose Lane	Barberton	OH	44203	11	
12		9954 East Thatcher Ave.	Muskegon	MI	49441	12	
13		38 Lake Street	Marshfield	WI	54449	13	
14		371 W. Spring Street	Pittsburgh	PA	15286	14	
15		8961 Oxford Lane	Merrick	NY	11566	15	
16		215 E Madison St	Hillsboro	TX	76645	16	
17		216 E Madison St	Hillsboro	TX	76645	17	
18		217 E Madison St	Hillsboro	TX	76645	18	
19		218 E Madison St	Hillsboro	TX	76645	19	
20		219 E Madison St	Hillsboro	TX	76645	20	
21		220 E Madison St	Hillsboro	TX	76645	21	
22		221 E Madison St	Hillsboro	TX	76645	22	
23		222 E Madison St	Hillsboro	TX	76645	23	
24		223 E Madison St	Hillsboro	TX	76645	24	
25		224 E Madison St	Hillsboro	TX	76645	25	
26		225 E Madison St	Hillsboro	TX	76645	26	
27		226 E Madison St	Hillsboro	TX	76645	27	
28		227 E Madison St	Hillsboro	TX	76645	28	
29		228 E Madison St	Hillsboro	TX	76645	29	
30		229 E Madison St	Hillsboro	TX	76645	30	
31		230 E Madison St	Hillsboro	TX	76645	31	
32		231 E Madison St	Hillsboro	TX	76645	32	
33		232 E Madison St	Hillsboro	TX	76645	33	
34		233 E Madison St	Hillsboro	TX	76645	34	
35		234 E Madison St	Hillsboro	TX	76645	35	
36		235 E Madison St	Hillsboro	TX	76645	36	
37		236 E Madison St	Hillsboro	TX	76645	37	
38		237 E Madison St	Hillsboro	TX	76645	38	
39		238 E Madison St	Hillsboro	TX	76645	39	
40		239 E Madison St	Hillsboro	TX	76645	40	
41		240 E Madison St	Hillsboro	TX	76645	41	
42		241 E Madison St	Hillsboro	TX	76645	42	
43		242 E Madison St	Hillsboro	TX	76645	43	
44		243 E Madison St	Hillsboro	TX	76645	44	
45		244 E Madison St	Hillsboro	TX	76645	45	

Table 2: Auction Table

```
mysql> SELECT * FROM Auction;
+-----+-----+-----+
| timestamp_start | timestamp_end | item_id |
+-----+-----+-----+
| 1475197317 | 1472580117 | 14 |
| 1475262117 | 1472666517 | 7 |
| 1475370117 | 1472752917 | 5 |
| 1475456517 | 1472839317 | 4 |
| 1475542917 | 1472925717 | 11 |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

Table 3: Bid Table

```
mysql> SELECT * FROM Bid;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| bid_id | amount | cancellation_timestamp | item_id | auction_timestamp_start | credit_card_number | shipping_address_id | is_auto | max_amount |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | 500.00 | 1472580118 | 14 | 1475197317 | 6011003807832773 | 4 | 1 | 700.00 |
| 2 | 800.00 | 1472580119 | 14 | 1475197317 | 6011003807839287 | 2 | 1 | 1000.00 |
| 3 | 950.00 | 1472580120 | 14 | 1475197317 | 379580062576998 | 1 | 0 | NULL |
| 4 | 20.00 | 1472666517 | 7 | 1475262117 | 6011003807832773 | 4 | 1 | 220.00 |
| 5 | 60.00 | 1472666518 | 7 | 1475262117 | 6011003807839287 | 2 | 0 | NULL |
| 6 | 100.00 | 1472752918 | 5 | 1475370117 | 4430400082342452 | 6 | 1 | 300.00 |
| 7 | 150.00 | 1472752919 | 5 | 1475370117 | 6011003807838232 | 8 | 1 | 350.00 |
| 9 | 175.00 | 1472752920 | 5 | 1475370117 | 4430400082342452 | 6 | 0 | NULL |
| 10 | 178.00 | 1472752921 | 5 | 1475370117 | 1234567890123456 | 12 | 1 | 378.00 |
| 11 | 200.00 | 1472752922 | 5 | 1475370117 | 4430400082732378 | 3 | 0 | NULL |
| 12 | 100.00 | 1472839318 | 4 | 1475456517 | 6011003807838232 | 8 | 1 | 300.00 |
| 13 | 200.00 | 1472839319 | 4 | 1475456517 | 6011003807832773 | 4 | 0 | NULL |
| 14 | 2000.00 | 1472925717 | 11 | 1475542917 | 379580062565888 | 5 | 1 | 2200.00 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
13 rows in set (0.01 sec)
```

Table 4: Category Table

category_id	description	category_name	parent_id
1	All	all items	NULL
100	women	women's shoes	100
101	boots	women's boots	100
111	sbooties	women's sbooties	110
112	flat_boots	women's flat boots	110
113	fuzzies	women's fuzzies boots	110
114	heel_boots	women's heel boots	110
115	snow_boots	women's snow boots	110
116	western	women's western boots	110
120	flats	women's flats	100
121	boats	women's boat flats	120
122	espadrilles	women's espadrilles flats	120
123	loafers	women's loafers	120
124	lace_up	women's lace up flats	120
125	open_toe	women's open toe flats	120
126	pointed_toe	women's pointed toe flats	120
130	pumps	women's pumps	100
131	block_heel	women's block heel pumps	130
132	classic	women's classic pumps	130
133	platform	women's platform pumps	130
134	sling_back	women's sling back pumps	130
140	slippers	women's slippers	100
141	block_heel	women's block heel slippers	140
142	slip_on	women's slip-on slippers	140
143	loafers/moccasins	women's loafers/moccasins	140
145	athletic	women's athletic	150
151	athletic	women's athletic sneakers	150
152	comfort	women's comfort sneakers	150
153	high_top	women's high-top sneakers	150
154	lace_up	women's lace-up sneakers	150
155	slip_on	women's slip-on sneakers	150
156	wedges	women's wedges	160
161	espadrilles	women's espadrilles wedges	160
162	evening	women's eveningwedges	160
163	open_toe	women's open-toe wedges	160
164	platform	women's platform wedges	160
165	pointed_toe	women's pointed-toe wedges	160
200	men	all men's	1
210	athletic	men's athletic shoes	200
211	baseball	men's baseball shoes	210
212	basketball	men's basketball shoes	210
213	cross_training	men's cross training shoes	210
214	football	men's football shoes	210
215	golf	men's golf shoes	210
216	running	men's running shoes	210
217	soccer	men's soccer shoes	210
218	tennis	men's tennis shoes	210
219	walking	men's walking shoes	210
220	boots	men's boots	260
221	casual	men's casual boots	220
222	dress	men's dress boots	220
223	hiking	men's hiking boots	220
224	rain	men's rain boots	220
225	western	men's western boots	220
226	work	men's work boots	220
230	casual	men's casual shoes	200
231	boats	men's boats	230
232	clogs	men's clogs	230
233	loafers	men's loafers	230
234	oxford	men's oxford	230
240	dress	men's dress shoes	200
241	drivers	men's drivers	240
242	lace_up	men's lace-up dress shoes	240
243	loafers	men's loafer dress shoes	240
244	oxford	men's oxford dress shoes	240
245	retro	men's retro dress shoes	240
246	slip_on	men's slip-on dress shoes	240
250	sandals	men's sandals	200
251	casual	men's casual sandals	250
252	fisherman	men's fisherman sandals	250
253	slip_on	men's slip-on sandals	250
254	sports	men's sports sandals	250
260	slippers	men's slippers	200
261	comfort	men's comfort slippers	260
262	loafers	men's loafer slippers	260
263	moccasins	men's moccasins slippers	260
300	kids	all kids' shoes	1
310	girls	all girls' shoes	300
311	athletic	girls' athletic shoes	310
312	boots	girls' boots	310
313	casual	girls' casual shoes	310
314	comfort	girls' comfort shoes	310
315	dress	girls' dress shoes	310
316	flats	girls' flats	310
317	flip_flop	girls' flip-flops	310
318	outdoor	girls' outdoor shoes	310
320	boys	all boys' shoes	300
321	athletic	boys' athletic shoes	320
322	boots	boys' boots	320
323	casual	boys' casual shoes	320
324	comfort	boys' comfort shoes	320
325	dress	boys' dress shoes	320
326	flats	boys' flats	320
327	flip_flop	boys' flip-flops	320
328	outdoor	boys' outdoor shoes	320
330	babies	all babies' shoes	300
331	0_6_months	babies' shoes for 0-6 months babies	330
332	6_12_months	babies' shoes for 6-12 months babies	330
333	12_18_months	babies' shoes for 12-18 months babies	330
334	18_24_months	babies' shoes for 18-24 months babies	330
335	2_3_years	babies' shoes for 2-3 years old kids	330
400	accessories	all shoe related accessories	1
410	socks/legwear	all socks and legwear	400
411	art	art socks	410

412	athletic	athletic socks	410
413	casual	casual socks	410
414	dress	dress socks	410
415	fuzzy	fuzzy socks	410
416	leggings	leggings	410
417	stickings	stickings	410
418	yoga_pants	yoga pants	410
420	care	all shoe care products	400
421	insoles	insoles	420
422	running_spikes	running spikes	420
423	shoe_cleaner	shoe cleaner	420
424	shoe_laces	shoe laces	420
425	shoe_polish	shoe polish	420
426	shoe_shiner	shoe shiner	420
427	spray	spray	420
428	stain_eraser	stain eraser	420
429	water_resistant	water resistant products	420
430	bags	all bags	400
431	backpack	backpack	430
432	duffel	duffel	430
433	totes	totes	430
440	sunglasses	all sunglasses	400
441	aviators	aviators sunglasses	440
442	club_masters	clubmasters sunglasses	440
443	round	round sunglasses	440
444	wayfarers	wayfarers sunglasses	440

130 rows in set (0.01 sec)

Table 5: Company_Data Table

mysql> SELECT * FROM Company;	
+	-+-----+
point_of_contact	supplier_id
+	-+-----+
Craig B. McDonough	16
Emmaline Sidhu	17
Yelena Mckibben	18
Margeret Dejean	19
Byron Loftin	20
Twanna Brin	21
Lyn Laroque	22
Angelika Polinsky	23
Madaline Bolanos	24
Melonie Krebbs	25
Queen Treacy	26
Damien Aguon	27
Sanjuana Strackbein	28
Hanh Tai	29
Jacki Stinger	30
Jeannine Ricken	31
Candra Mire	32
Dotty Perrigo	33
Adriene Kirsh	34
Agnus Shiflett	35
Davis Rybak	36
Margherita Rodriguez	37
Milford Neilsen	38
Carey Rone	39
Lionel Haapala	40
Charissa Winkleman	41
Trina Styles	42
Virgie Beale	43
Roosevelt Marcantel	44
Cordell Vanhooser	45
+	-+-----+
30 rows in set (0.00 sec)	

Table 6: Contact_By Table

```
mysql> SELECT * FROM contact_by;
+-----+-----+
| phone_number | supplier_id |
+-----+-----+
| 8144417999 | 1 |
| 2135732879 | 2 |
| 9178893204 | 3 |
| 8142738930 | 4 |
| 2138940007 | 5 |
+-----+-----+
5 rows in set (0.00 sec)
```

Table 7: Credit_Card Table

```
mysql> SELECT * FROM Credit_Card;
+-----+-----+-----+-----+-----+-----+
| number | name | type | expiration | username | billing_address_id |
+-----+-----+-----+-----+-----+-----+
| 379580062526777 | Gabe Moran | credit | 2021-09-01 | gabe7 | 7 |
| 379580062565888 | Eros Castillo | credit | 2020-03-01 | eros5 | 5 |
| 379580062576538 | Oliver Gibbs | credit | 2021-02-01 | olly14 | 14 |
| 379580062576998 | Abby Jackson | credit | 2020-09-01 | abby1 | 1 |
| 379580062590111 | Neal Dean | credit | 2019-08-01 | mammoth12 | 12 |
| 379580062598383 | Jeff Casey | credit | 2020-05-01 | jeff9 | 9 |
| 1234567890123456 | Mammy Mammoth | credit | 2020-02-01 | mammoth12 | 12 |
| 4430400082342452 | Frank Figueroa | debit | 2021-08-01 | frank6 | 6 |
| 4430400082724253 | Kelly Quinn | debit | 2021-04-01 | kelly10 | 10 |
| 4430400082732378 | Cathy Ho | debit | 2019-03-01 | cathy3 | 3 |
| 4430400082735352 | Noel Ramirez | debit | 2020-03-01 | noho13 | 13 |
| 6011003807822317 | Rodolfo Richardson | credit | 2020-01-01 | polarbear15 | 15 |
| 6011003807832773 | Dishti Foster | credit | 2021-05-01 | dishti4 | 4 |
| 6011003807838232 | Hiren Norris | credit | 2019-05-01 | hiren8 | 8 |
| 6011003807839287 | Benjamin Smith | credit | 2020-02-01 | ben2 | 2 |
| 6011003807839972 | Linda Rivera | credit | 2020-12-01 | linda11 | 11 |
+-----+-----+-----+-----+-----+-----+
```

Table 8: Custom_Shoe Table

```
mysql> SELECT * FROM Custom_shoe;
+-----+-----+-----+-----+-----+-----+-----+-----+
| tong_style | sole_style | shoe_style | color | side_img_id | front_img_id | back_img_id | item_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Graphic Mesh | Rubber | Athletic | Green | 101 | 102 | 103 | 101 |
| Mesh | Rubber | Athletic | Blue | 104 | 105 | 106 | 102 |
| Suede | Cork Nitrile | Casual | Grey | 107 | 108 | 109 | 103 |
| Suede | Leather | Boots | Black | 110 | 111 | 112 | 104 |
| Leather | Cork Nitrile | Work | Brown | 113 | 114 | 115 | 105 |
+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```

Table 9: Customer_Service Table

```
mysql> SELECT * FROM Customer_Service;
+-----+-----+-----+
| customer_service_id | email | faqs | phone_number |
+-----+-----+-----+
| 1 | abby1@techfam.com | Q: Can I chooses different shipping methods? A:Yes,we offer UPS ground shipping and priority shipping. | 8144417999 |
| 2 | ben2@techfan.com | Q: Can I return? A: 30 days for direct sale items and no returns for auction items. | 2135732879 |
| 3 | cathy3@techfam.com | Q: Can I bid twice for one item. A:Yes. | 9178893204 |
| 4 | dishti4@techfam.com | Q: Can I work at your company? A: No we aren't hiring. | 8142738930 |
| 5 | eros5@techfam.com | Q: What if I really want to work at your company? A: Thanks for you interest, but I said we weren't hiring. | 2138940007 |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

Table 10: Footwear Table

size	item_id
5-9.5	1
5-9.5	2
5-10	3
5-10	4
5.5-7.5	5
5-10	6
5-11	7
5-9.5	8
5-10	9
5-9.5	10
5-9.5	11
5-9.5	12
7-10	13
8-12	14
8-11	15
7-12	16
8-12	17
8-12	18
10-12	19
7-12	20
7-12	21
7-12	22
7-12	23
7-12	24
7-12	25
7-12	26
7-12	27
7-12	28
1-13 kids	29
1-13 kids	30
1-13 kids	31
1-13 kids	32
1-13 kids	33
18-24 months	34
1-13 kids	35
1-13 kids	36
1-13 kids	37
12-18 months	38
6-12 months	39
6-12 months	40
7-10	41
7-9	42
7-9	43
7-9	44
7-9	45
7-9	46
7-9	47
7-9	48
7-9	49
7-9	50

5-11	51
5-11	52
5-11	53
5-11	54
5-11	55
5-10	56
5-10	57
5-11	58
5.5-10	59
6-8	60
5-11	61
5-11	62
S-XL	63
5-11	64
5-11	65
5-11	66
5-11	67
5-11	68
5-11	69
5-11	70
7-10	71
5-6	72
5-10	73
5-10	74
8-13	75
8-13	76
7.5-14	77
8-13	78
8-13	79
8-13	80
7-13	81
7-13	82
7-13	83
7-16	84
7-13	85
7-13	86
S-XXL	87
8-13	88
7-13	89
1-4	90
3-7	91
3-7	92
3-7	93
3.5-7	94
3-7	95
"0-6 months, 6-12 months"	96
"0-6 months, 6-12 months"	97
18-24 months	98
"0-6, 6-12, 12-18, 18-24 Months"	99
"6-12, 12-18, 18-24 Months"	100
7	101
7	102
8	103
9	104

7	105
-----+	-----+

105 rows in set (0.01 sec)

Table 11: Has_Visual Table

```
mysql> SELECT * FROM has_visual;
+-----+-----+
| item_id | img_id | color
+-----+-----+
| 1       | 1       | black
| 2       | 2       | black
| 3       | 3       | Sughero
| 4       | 4       | Nude
| 5       | 5       | Black/Ouro
| 6       | 6       | Ivory
| 7       | 7       | Black/Ivory
| 8       | 8       | Silver/Silver
| 9       | 9       | Burgundy
| 10      | 10      | Black
| 11      | 11      | Black
| 12      | 12      | Silver
| 13      | 13      | Carnival
| 14      | 14      | Black
| 15      | 15      | Black
| 16      | 16      | Black
| 17      | 17      | Camel/Camel
| 18      | 18      | Blue
| 19      | 19      | Black
| 20      | 20      | Black
| 21      | 21      | Blue Marine
| 22      | 22      | Black
| 23      | 23      | White
| 24      | 24      | Blue
| 25      | 25      | Blue
| 26      | 26      | Blue
| 27      | 27      | Black
| 28      | 28      | Black
| 29      | 29      | quail grey
| 30      | 30      | Blue Tint
| 31      | 31      | Suede Blush
| 32      | 32      | Platinum
| 33      | 33      | Silver
| 34      | 34      | Chestnut
| 35      | 35      | Black Denim
| 36      | 36      | Blue
| 37      | 37      | Black
| 38      | 38      | Suede
| 39      | 39      | Red
| 40      | 40      | Blue
| 41      | 41      | Black
| 42      | 42      | Bordeaux/Dark Brown
| 43      | 43      | Fantasy Silver
| 44      | 44      | Nero/Oro
| 45      | 45      | Gold
| 46      | 46      | Ivory
| 47      | 47      | Black
| 48      | 48      | White
| 49      | 49      | Cognac
| 50      | 50      | Black
+-----+-----+
```

51	51	"Black Saffiano
52	52	"Dark Stone
53	53	"Black Leather
54	54	"Brown
55	55	"Brown
56	56	"Black
57	57	"Natural
58	58	"Black
59	59	"Glazed Ginger
60	60	"Black
61	61	"Raspberry
62	62	"Black
63	63	"Cream
64	64	"Black
65	65	"Garnet
66	66	"Black
67	67	Black
68	68	"Black
69	69	"Black
70	70	"Platino
71	71	"Silver
72	72	Electric Blue
73	73	White/Black
74	74	"Pearl Grey
75	75	"Black
76	76	Orange
77	77	Black
78	78	Dark Brown
79	79	Black/Brown
80	80	Brown
81	81	"Canteen
82	82	"Black
83	83	"Black
84	84	"Black
85	85	Black
86	86	Cognac
87	87	"Stone
88	88	Brown
89	89	Brown
90	90	Brown
91	91	Black
92	92	Navy
93	93	Black
94	94	Magenta/Navy
95	95	Onix/Sun Glow/Grey
96	96	Grey
97	97	Pink
98	98	White
99	99	Grey
100	100	Grey

100 rows in set (0.00 sec)

Table 12: Image Table

```
mysql> SELECT img_id FROM Image;
```

img_id
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

Note that this table also has a column of images, but this could not be printed out due to the size of the data. Also note that the real table goes all the way up to Image_ID 115.

Table 13: Phone Table

```
mysql> SELECT * FROM Phone;
+-----+
| phone_number |
+-----+
| 2134567389 |
| 2135732879 |
| 2137849380 |
| 2137892635 |
| 2138940007 |
| 8142738930 |
| 8144413490 |
| 8144417880 |
| 8144417930 |
| 8144417991 |
| 8144417999 |
| 9176352937 |
| 9176538290 |
| 9178269309 |
| 9178893204 |
+-----+
15 rows in set (0.01 sec)
```

Table 14: Rating Table

```
mysql> SELECT * FROM Rating;
+-----+-----+-----+-----+
| rating_id | explanation | value | username | supplier_id |
+-----+-----+-----+-----+
| 4 | very good! | 5 | dishti4 | 1 |
| 1 | excellent! | 4 | abby1 | 3 |
| 5 | good! | 5 | eros5 | 4 |
| 2 | satisfied purchase! | 5 | ben2 | 7 |
| 3 | good stuff but I don't like the color | 4 | cathy3 | 13 |
+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```

Table 15: Register Users Table

username	password	age	gender	income	supplier_id
abby1	7c4a8d09ca3762af61e59520943dc26494f8941b	20	Female	100	1
ben2	3d4f2bf07dc1be38b20cd6e46949a1071f9d0e3d	22	Male	2000	2
cathy3	5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8	35	Female	200	3
dishti4	0c3310ae5f5ac0e5c164912108328111ce443619	20	Female	2300	4
eros5	7c4a8d09ca3762af61e59520943dc26494f8941b	22	Male	2500	5
frank6	7c4a8d09ca3762af61e59520943dc26494f8941b	21	Male	3000	6
gabe7	0c3310ae5f5ac0e5c164912108328111ce443619	28	Female	300	7
hiren8	5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8	20	Male	5000	8
jeff9	3d4f2bf07dc1be38b20cd6e46949a1071f9d0e3d	22	Male	10000	9
kelly10	7c4a8d09ca3762af61e59520943dc26494f8941b	29	Female	400	10
linda11	0c3310ae5f5ac0e5c164912108328111ce443619	32	Female	1200	11
mammoth12	5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8	60	Female	3000	12
noho13	0c3310ae5f5ac0e5c164912108328111ce443619	33	Male	500	13
olly14	3d4f2bf07dc1be38b20cd6e46949a1071f9d0e3d	17	Male	400	14
polarbear15	5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8	34	Male	800	15
tech	40bd001563085fc35165329ea1ff5c5ecbdbbeef	1	Female	100	-1

16 rows in set (0.01 sec)

Table 16: Sale Table

transaction_id	price	date	auction_or_sale	item_id	credit_card_number	shipping_address_id
1	950.00	1472580117		14	379580062576998	1
2	50.00	147266517		7	6011003807839287	2
3	200.00	1472752917		5	4430400082732378	3
4	200.00	1472839317		4	6011003807832773	4
5	2250.00	1472925717		11	379580062565888	5
6	195.00	1478385345		70	379580062598383	9

6 rows in set (0.00 sec)

Table 17: Sales Item Table

item_id	count	brand	list_price	state	name	reserved_price	category_id	supplier_id	address_id
1	44	Hilfiger Collection	680.00	new	Rock-n-Roll Demi	650.00	132	8	8
2	46	Giuseppe Zanotti	695.00	new	Moon Boots	650.00	112	10	10
3	51	KENDALL + KYLIE	725.00	new	Adult High Heel Boots	2000.00	114	14	14
4	58	Dolce & Gabbana	225.00	new	Dieci Ankle Strap Pumps	200.00	133	1	1
5	78	Schutz	220.00	new	Myrna Sandals	200.00	132	13	13
6	33	Soludos	75.00	new	Tulip Lace Smoking Slipper Espadrilles	70.00	122	12	12
7	79	Kate Spade New York	68.00	new	Berry Slippers	50.00	142	7	7
8	61	Giuseppe Zanotti	1225.00	new	Leather Wing Sneakers	1200.00	154	13	13
9	63	Tommy Morrison	70.00	new	Romy Wedge Pumps	55.00	145	1	1
10	72	Sarah Flint	1195.00	new	Cortina Boots	1150.00	115	9	9
11	81	Marc Jacobs	2295.00	new	Dede Platform Button Boots	2250.00	114	4	4
12	42	Sergio Rossi	1895.00	new	Mermaid Cage Booties	1850.00	111	11	11
13	59	Stuart Weitzman	785.00	new	Gor Fur It Booties	750.00	113	4	4
14	12	Calvin Klein Collection	995.00	new	Bob Zip Boots	950.00	221	3	3
15	34	E Boot New York	395.00	new	Berserkorme Lace Up Oxford	350.00	220	4	4
16	30	Salvatore Ferragamo	300.00	new	Apple Plain Toe Lace Up Shoes	700.00	245	8	8
17	91	Calvin Klein Collection	895.00	new	Urban High Top Sneakers	850.00	219	10	10
18	13	Marni	720.00	new	Sandals	700.00	251	11	11
19	43	Eytys	380.00	new	Jojo Sandals	350.00	253	3	3
20	81	Salvatore Ferragamo	780.00	new	Destin Apron Toe Bit Loafers	750.00	243	9	9
21	72	Salvatore Ferragamo	640.00	new	Massie Bit Suede Loafers	600.00	250	13	13
22	63	Nike	150.00	new	Mid Flyknit Flyknit	100.00	216	6	6
23	45	Nike	65.00	new	Nike Court Lite	50.00	218	8	8
24	55	Nike	140.00	new	Nike Golf FI Impact 2	100.00	215	12	12
25	23	Nike	100.00	new	Nike Zoom Train Action	90.00	213	7	7
26	21	Under Armour	75.00	new	Under Armour UA Jet Mid	70.00	212	3	3
27	49	Mizuno	36.00	new	Mizuno 9-Spike Advanced Franchise 8 Low	35.00	211	15	15
28	15	Levi's	50.00	new	Levi's 9-Spike Advanced Franchise 8 Mid Football	51.00	214	4	4
29	34	The North Face	65.00	new	The North Face Kids Amore (Little Kid/Big Kid)	60.00	312	14	14
30	31	Vans	42.00	new	Vans Kids Classic Slip-On (Little Kid/Big Kid)	40.00	313	4	4
31	97	Elephantito	73.00	new	Elephantito Mary Jane (Toddler/Little Kid)	70.00	316	14	14
32	53	Jack Rogers	69.00	new	Jack Rogers Miss Hamptons II (Toddler/Little Kid/Big Kid)	60.00	317	13	13
33	37	Freshly Picked	60.00	new	Freshly Picked Soft Sole Moccasins (Infant/Toddler)	50.00	314	4	4
34	38	Ugg	50.00	new	UGG Australia Ugg Boot	45.00	324	3	3
35	42	TOMS	42.00	new	TOMS Kids Rimini Espadrille (Little Kid/Big Kid)	40.00	323	5	5
36	28	Keen	50.00	new	Keen Kids Newport H2 (Little Kid/Big Kid)	45.00	323	12	12
37	45	Kenneth Cole	56.00	new	Kenneth Cole Reaction Kids T-Flex Sr (Little Kid/Big Kid)	50.00	325	4	4
38	18	Elephantito	73.00	new	Elephantito Sabrina (Toddler/Little Kid)	70.00	333	13	13
39	57	Converse	35.00	new	Converse Kids Chuck Taylor All Star Street Mid (Infant/Toddler)	30.00	331	14	14
40	100	Ugg	48.00	new	UGG Australia Ugg Boot	45.00	327	1	1
41	57	Giuseppe Zanotti	1395.00	new	Fur Boots	1360.00	113	12	12
42	77	Maison Margiela	1295.00	new	Loafer Wedges	1250.00	164	10	10
43	58	Charlotte Olympia	1225.00	new	Leandra Sandals	1220.00	164	14	14
44	3	Versace	1040.00	new	Wedge Sandals	1000.00	164	4	4
45	84	Giuseppe Zanotti	995.00	new	Leather Sneakers	950.00	152	11	11
46	83	Sarah Flint	995.00	new	Lily Flat	920.00	136	34	34
47	36	R13	995.00	new	Platform Combat Boots	950.00	112	14	14
48	16	Giuseppe Zanotti	995.00	new	Studded Slip On Sneakers	950.00	155	9	9
49	54	Aquazzura	950.00	new	Tiger Lily Flat Booties	900.00	111	2	2
50	54	Aquazzura	950.00	new	Eva Booties	900.00	111	4	4
51	62	Michael Kors	99.00	new	HIGHMICHAEL Michael Kors Fulton Moc Flats	NULL	120	16	16

51	62	Michael Kors	99.00	new	MICHAEL Michael Kors Fulton Moc Flats	NULL	120	16	16
52	37	Lucky Brand	139.00	new	women's Perforated Basel Booties	NULL	111	17	17
53	51	Michael Kors	99.00	new	MICHAEL Michael Kors MK Flex Mid Pointed-Toe Pumps	NULL	130	16	16
54	65	GUESS	89.00	new	6 by GUESS Halleen Riding Boots	NULL	112	18	18
55	56	Style& Co.	79.50	new	Style& Co. Jamila Zip Booties	NULL	111	19	19
56	93	Converse	54.99	new	Converse Women's Chuck Taylor Ox Casual Sneakers From Finish Line	NULL	152	20	20
57	47	American Rag	85.50	new	Dawn Western Boots	NULL	116	29	29
58	24	Carlos by Carlos Santana	179.00	new	Acce Western Boots	NULL	116	21	21
59	67	Timberland	140.00	new	Women's Kensington Cold-Weather Boots	NULL	115	22	22
60	88	Bandolino	45.00	new	Women's Limerick Boat Shoes	NULL	121	23	23
61	27	Michael Kors	89.00	new	MICHAEL Michael Kors Dana Espadrille Flats	NULL	122	16	16
62	66	Steve Madden	89.00	new	Men's Sunshine Lace-Up Pointed Flats	NULL	124	38	38
63	59	Isotoner Signature	24.00	new	Isotoner Satin Ballerina Slipper	NULL	140	24	24
64	29	BCGeneration	69.00	new	Parade Platform Pump	NULL	133	25	25
65	79	Calvin Klein	129.00	new	Cassila Mary Jane Pumps	NULL	131	26	26
66	88	Calvin Klein	99.00	new	Women's Gayle Pointed-Toe Pumps	NULL	132	26	26
67	59	Anne Klein	100.00	new	Ideona Slingback Pumps	NULL	134	27	27
68	87	Bandolino	69.00	new	Yara Pointed Toe Demi Wedges	NULL	165	28	28
69	69	American Rag	59.50	new	Audria Two-Piece Platform Wedges	NULL	164	29	29
70	81	Badgley Mischka	195.00	new	Awake Evening Wedge Pumps	NULL	162	30	30
71	56	Mac Fisher	89.00	new	Haisey Espadrille Wedge Sandals	NULL	161	31	31
72	50	Michael Kors	195.00	new	MICHAEL Michael Kors Nikko High-Top Sneakers	NULL	153	16	16
73	86	Adidas	88.00	new	Women's Superstar Casual Sneakers From Finish Line	NULL	154	32	32
74	76	Michael Kors	130.00	new	MICHAEL Michael Kors Kyle Perforated Slip-On Sneakers	NULL	155	16	16
75	73	Reebok	55.00	new	Men's Trainfusion 5.0 L MT Training Sneakers From Finish Line	NULL	218	33	33
76	72	Nike	100.00	new	Men's Free Run Running Sneakers From Finish Line	NULL	216	34	34
77	46	Nike	75.00	new	Men's Air Versitile Basketball Sneakers From Finish Line	NULL	212	34	34
78	61	Polo Ralph Lauren	350.00	new	Polo Ralph Lauren Men's Mersey Pull-On Boots	NULL	226	35	35
79	3	Timberland	170.00	new	"Men's Earthkeeper Original 6"" Boot"	NULL	226	22	22
80	59	North Face	110.00	new	"Men's Chilkat II Waterproof Lace-Up Boots	NULL	224	36	36
81	98	Timberland	198.00	new	"Men's 6"" Premium Waterproof Boots"	NULL	224	22	22
82	58	Polo Ralph Lauren	195.00	new	Polo Ralph Lauren Men's Nickson Wingtip Boots	NULL	222	35	35
83	95	Clarks	80.00	new	Men's Cottrell Step Bike Toe Casual Slip-Ons	NULL	238	37	37
84	6	Calvin Klein	110.00	new	Calvin Klein Men's Brodie Leather Oxford	NULL	244	26	26
85	76	Steve Madden	60.00	new	Men's Trace Loafers	NULL	243	38	38
86	60	Steve Madden	60.00	new	Men's Need Drivers	NULL	241	38	38
87	51	Club Room	44.00	new	Men's Slippers, Paul Suede Slip-On Loafers	NULL	262	39	39
88	51	Timberland	80.00	new	Men's Altamont 2.0 Fisherman Sandals	NULL	252	22	22
89	51	Alfani	60.00	new	Men's Surf Slingback Sandals	NULL	253	48	48
90	39	Tommy Hilfiger	30.00	new	Boys' 'Lil' Aiden Booties	NULL	322	41	41
91	56	Tommy Hilfiger	39.00	new	Boys' or Little Boys' Robbie Lace-Up Dress Shoes	NULL	325	41	41
92	65	Stride Rite	44.00	new	Little Girls' or Toddler Girls' Simone Mary-Janes	NULL	313	42	42
93	31	Hush Puppies	50.00	new	Girls' or Little Girls' Kenzie Flats	NULL	316	43	43
94	93	Crocs	15.00	new	Girls' or Little Girls' Hawaii Stripe Flip-Flops	NULL	317	44	44
95	14	Adidas	65.00	new	Little Girls' Hyperhiker Outdoor Sneakers From Finish Line	NULL	318	32	32
96	35	Robeez	30.00	new	Baby Boys' Morgan Booties	NULL	331	45	45
97	60	Robeez	38.00	new	Baby Girls' Classic Booties	NULL	331	45	45
98	72	Robeez	25.00	new	Baby Girls' Special Occasion Shoes	NULL	334	45	45
99	23	Robeez	26.00	new	Stylish Steve Shoes, Baby Boys	NULL	333	45	45
100	65	Robeez	26.00	new	Baby Girls' Cool Water Mary-Jane Layette Shoes	NULL	334	45	45
101	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
102	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
103	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
104	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
105	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
106	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
107	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
108	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
109	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
110	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
111	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
112	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
113	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
114	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
115	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
116	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
117	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
118	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
119	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
120	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
121	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
122	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
123	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
124	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
125	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
126	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
127	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
128	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
129	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
130	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
131	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45
132	200	TechFam	200.00	new	Custom Shoes	200.00	150	0	45

132 rows in set (0.00 sec)

Note that this table also has a description column, but due to the size of the data, it could not be fit

Table 18: Suppliers Table

```

mysql> SELECT * FROM Suppliers;
+-----+-----+-----+
| supplier_id | email           | name          |
+-----+-----+-----+
|      0 | us@techfam.com   | TechFam       | |
|      1 | abby1@techfam.com | Abby Jackson  |
|      2 | ben2@techfam.com  | Benjamin Smith|
|      3 | cathy3@techfam.com | Cathy Ho      |
|      4 | dishti4@techfam.com | Dishti Foster |
|      5 | eros5@techfam.com  | Eros Castillo |
|      6 | frank6@techfam.com | Frank Figueroa |
|      7 | gabe7@techfam.com  | Gabe Moran    |
|      8 | hiren8@techfam.com | Hiren Norris  |
|      9 | jeff9@techfam.com  | Jeff Casey    |
|     10 | kelly10@techfam.com | Kelly Quinn   |
|     11 | linda11@techfam.com | Linda Rivera  |
|     12 | mammoth12@techfam.com | Neal Dean    |
|     13 | noho13@techfam.com | Noel Ramirez  |
|     14 | olly14@techfam.com | Oliver Gibbs  |
|     15 | polarbear15@techfam.com | Rodolfo      | Richardson |
|     16 | a16@techfam.com    | Michael Kors  |
|     17 | a17@techfam.com    | Lucky Brand   |
|     18 | a18@techfam.com    | GUESS         |
|     19 | a19@techfam.com    | Style& Co.   |
|     20 | a20@techfam.com    | Converse      |
|     21 | a21@techfam.com    | Carlos by Carlos Santana |
|     22 | a22@techfam.com    | Timberland   |
|     23 | a23@techfam.com    | Nautica      |
|     24 | a24@techfam.com    | Isotoner Signature |
|     25 | a25@techfam.com    | BCBGeneration |
|     26 | a26@techfam.com    | Calvin Klein  |
|     27 | a27@techfam.com    | Anne Klein    |
|     28 | a28@techfam.com    | Bandolino     |
|     29 | a29@techfam.com    | American Rag  |
|     30 | a30@techfam.com    | Badgley Mischka |
|     31 | a31@techfam.com    | Marc Fisher   |
|     32 | a32@techfam.com    | Adidas        |
|     33 | a33@techfam.com    | Reebok        |
|     34 | a34@techfam.com    | Nike          |
|     35 | a35@techfam.com    | Polo Ralph Lauren |
|     36 | a36@techfam.com    | North Face   |
|     37 | a37@techfam.com    | Clarks        |
|     38 | a38@techfam.com    | Steve Madden  |
|     39 | a39@techfam.com    | Club Room    |
|     40 | a40@techfam.com    | Alfani        |
|     41 | a41@techfam.com    | Tommy Hilfiger |
|     42 | a42@techfam.com    | Stride Rite   |
|     43 | a43@techfam.com    | Hush Puppies  |
|     44 | a44@techfam.com    | Crocs         |
|     45 | a45@techfam.com    | Robeez        |
+-----+-----+-----+
46 rows in set (0.00 sec)

```

Table 19: UPS Table

```

mysql> SELECT * FROM UPS;
+-----+-----+-----+-----+
| shipping_id | shipping_date | shipping_method | transaction_id |
+-----+-----+-----+-----+
|      1 | 1472666517 | Shipped        |      1 |
|      2 | 1472752917 | Not Yet Shipped |      2 |
|      3 | 1472839317 | Delivered      |      3 |
|      4 | 1472925717 | Shipped        |      4 |
|      5 | 1473012117 | Shipped        |      5 |
|      6 | 1478471745 | Not Yet Shipped |      6 |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

```

Table 18: Item_Review

mysql> SELECT * FROM Item_Review;				
review_id	explanation	value	username	item_id
4	very good!	5	dishti4	1
1	excellent!	4	abby1	3
5	good!	5	eros5	4
2	satisfied purchase!	5	ben2	7
3	good stuff but I don't like the color	4	cathy3	13

5 rows in set (0.00 sec)

13. WORKS CITED

1. "Amazon." Amazon. N.p., n.d. Web. 13 Sept. 2016.
2. Caleres. *Shoes, boots, sandals - famous footwear online*. 2016. Web. 15 Sept. 2016.
3. "EBay." *Electronics, Cars, Fashion, Collectibles, Coupons and More*. N.p., n.d. Web. 15 Sept. 2016.
4. "Feedback Scores, Stars, and Your Reputation." Feedback Scores, Stars, and Your Reputation. N.p., n.d. Web. 13 Sept. 2016.
5. Harden, Seth. "Footwear Industry Statistics." Statistic Brain. N.p., 03 Sept. 2016. Web. 15 Sept. 2016.
6. "Official Store. Nike.com." N.p., n.d. Web. 15 Sept. 2016.
7. "Online Shoes, Clothing, Free Shipping and Returns| Zappos.com." Online Shoes, Clothing, Free Shipping and Returns| Zappos.com. N.p., n.d. Web. 15 Sept. 2016.
8. "Shoes, Boots, Sandals, Handbags, Free Shipping! | DSW.com." *Shoes, Boots, Sandals, Handbags, Free Shipping! | DSW.com*. N.p., n.d. Web. 15 Sept. 2016.
9. "The Purpose of Weekly Status Reports." Reportingadvice. N.p., 2013. Web. 14 Sept. 2016.
10. "United States." UPS Developer Kit. N.p., n.d. Web. 14 Sept. 2016.
11. Women's shoes, boots, Handbags & clothing online. 2010. Web. 15 Sept. 2016.
12. "Five Advantages & Disadvantages Of MySQL." *Datarealm*. N.p., 2016. Web. 14 Oct. 2016.