Database Design

Project 4 Final Report

Dr. Yicheng Tu

Clothes Shop: 3SumClosest

Executive Board:

- **Huy Duong**
- **James Casanova Williams**
- **Si Dang**

I. Overview

Group consists of three members: Huy Duong, Si Dang, and James Casanova Williams. We proposed a new online clothes store name “3SumClosest” where users can purchase clothes of their choice. The main idea of our website is to allow users from all around the USA to buy all types of clothes on our website with convenient checkout options.

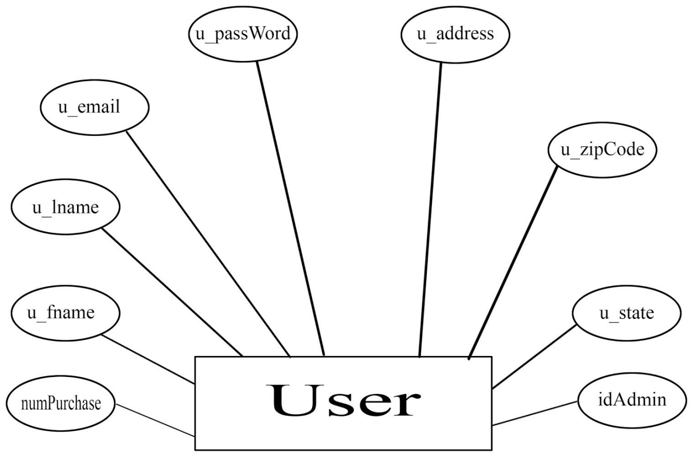
The website will be similar to Hollister, a well-known in-store and online clothing purchasing site, but we strive to offer clothes at a better price and an easy checkout option. We constructed the website with multiple views, where the admin view gets the opportunity to add and or remove clothes from the database. Additionally the administrator can alter the inventory count of the clothes on the database. Both of these functionalities will be conveniently available through the website and thus our website gives the user a seemingly user-friendly environment.

II. Database Design

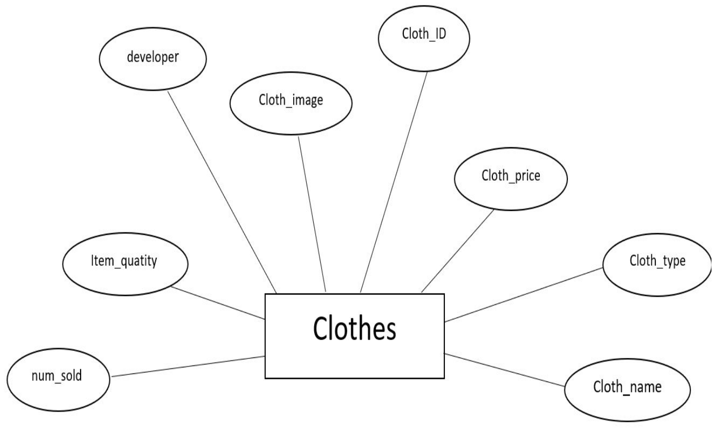
The following ER diagram provides a high-level introduction to our database. This will be helpful to understand the relation between the tables. (Only Ships to U.S.)

Entity-Relationship Diagram:

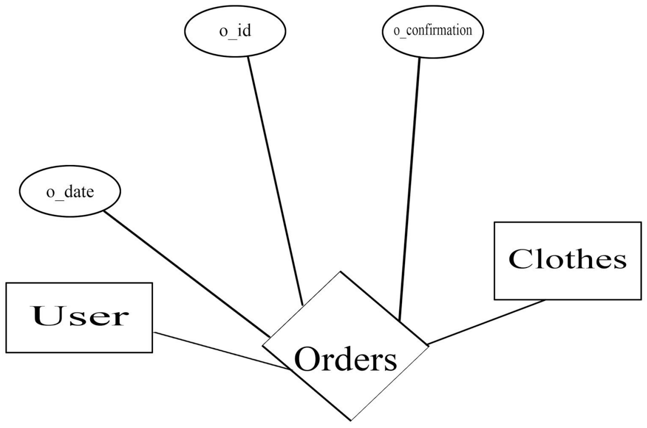
Users



Clothes



Orders



III. <u>Client Functionality</u>

Every page will have a menu bar at the top of the website. This will help all types of users to move around the website with aese. There is also a section, at the bottom of each page, where the contact information, both phone number and email address is provided. Also, credit is given to w3.css for providing the template for the website.

1. **Guest User**

If the user is not logged in, the menu bar will consist of four different options of "Home Page", "Shopping Cart", "Create Account" and "Login".

a. **Home Page**

The home page is the first page users can see when they connect to the website. The home page will have the most trending collection's name and it's pictures. Also, the most best selling clothes will be displayed on the home page with the prices. The top clothes keep changing as the number of purchases for each garment changes.

The home page will also consist of the shop by Category section. This section will have six different types of options. Each option will take the user to the different page, where they will be able to view more clothes under that specific category. Clicking on the home page button, the user will be brought back to the home page view.

Guest users can see the option to purchase on the home page, and they will still be able to purchase clothes. The log out page is simply for the logged in user to log out the website once they finish visiting the website.

The application layer will check if the user is logged in or not. If so, the section will get closed and the user will be logged out and directly to the home page of the guest user.

The application layer is implemented by HTML, CSS and PHP.

b. **Shopping Cart**

The shopping cart will contain all of the items that the user has selected from the website.

They will be able to adjust the quantity of items in their cart as well as remove items from it.

They will then be able to proceed to the checkout and purchase their items.

c. Create Account

The Create Account page will ask the user to enter their personal information as well as the password they would like to use for security purposes.

It will ask them their first name, last name, address, email address, password, address, zip code, and state.

Once all the information is entered the user will click the submit button and their account will be created.

d. Login

The login page will be for the user with an existing account. Here the user will ask to input their user name in the form of email and password attached to that account.

Similar to the New user page, if there is a missing information detected then the error message will pop up.

The application layer will query the database with the information the user provides. If the information is valid for a user in the database then it will log that user in.

Additionally, if a user tries to login with a username not present in the user table, the application layer will query the database and look for that username if not present in the table, the application layer will return the error message on the client side.

2. Account User

Once the user is logged in, the menu bar will consist of the four options: “Home Page”, “Shopping Cart”, “Create Account”, and “Logout”

a. Home Page

The home page function is the same as mentioned above in the guest user section above.

When the user is logged in, they can purchase the clothes they like. They can also input the number of clothes they would like to purchase and add them in their shopping cart.

The same function for adding the clothes to the shopping cart is applied to all the clothes which users see in the shop by category section.

The application layer will invoke the shopping cart php file, which will query the database and get the cloth id which was selected by the client. Meanwhile the shopping cart will be updated and displayed on the client sides.

b. Shopping Cart

The shopping cart is a temporary state page that shows the user which items they have added to their order during their current visit to the website.

The users can change their shopping cart by both modifying the quantities of clothes in their cart when by add the clothes to shopping cart, as well as deleting clothes from their cart.

The users will be able to view the total price of their order.

When the user click on purchase button, they will be transfer to order page.

The user is supposed to enter their details, such as email, first name, last name, address and credit card number. Once hit the submit they will be provided a confirmation number which will be unique for each purchase order.

c. Login

The user will click the button on the menu bar on the home page and be brought to this page where they will be prompted to enter the email address and password associated with their account.

There will also be a button below offering them the chance to create an account.

Once the user has entered their information into the email and password boxes they will click submit and be brought to the home page.

d. Logout

This will be a clickable button on the menu bar that will logout the user when clicked and take them back to the home page.

3. Admin

a. Login

The user will click the button on the menu bar on the home page and be brought to this page where they will be prompted to enter the email address and password associated with their account.

There will also be a button below offering them the chance to create an account.

Once the user has entered their information into the email and password boxes they will click submit and be brought to the home page.

b. Home Page

In the admin page, the admin can see top collections and look around the website mostly the same as a user except the admin will have several options a normal user does not have.

The admin will have access to add clothes, change prices, change stock, delete clothes and update clothing quantity.

The application layer will query the database with the manager’s email and password. If successful, the user will be detected to admin view.

Here the user will not be able to buy any clothes.

c. Add Clothes

The page will require the admin to insert the clothing name, clothing type, clothing year released, the name of the clothing designer, the clothing’s price and the stock quantity.

When the admin clicks on add cloth the application layer will use the information inserted and update the clothing table with it.

When the update is successful the admin will be notified by a message.

d. Change Price

This page will require the admin to input the cloth ID, cloth name and new price.

The application layer will query the database with the inserted cloth ID and cloth name. If matched with any existing clothes in the cloth table then the price for that cloth will be changed in the database.

e. Change Stock

This page will require the admin to input the cloth ID, cloth name and new stock for that cloth.

The application layer will query the database with the inserted clothing ID and clothing name. If matched with any existing clothes in the cloth table then the stock number entered will be added to the existing stock number in the database.

f. Delete Clothes

Only the admin can view this page.

This page will display the item's picture and information with a red button at the bottom of the page that says "Delete Item".

Once the button is clicked, the application layer will clear the item's information from the database.

When the update is successful the admin will be notified by message.

g. Log Out

This will be a clickable button on the menu bar that will logout the user when clicked and take them back to the home page.

IV. Database Tables

The Database Tables section explains in detail the tables structure in the database. This section breaks down the tables relationships, keys, table input types, and other details. Please refer to the E-R diagrams at the beginning of the document for a higher-level graphical description.

Users

```
Users (user_ID: INT(11), user_email: VARCHAR(40), user_fname:
VARCHAR(20), user_lname: VARCHAR(20), user_pass: VARCHAR(20),
num_purchased_clothes: INT(20), id_admin: TINYINT(1))
```

Foreign keys: `none`
Candidate key: `user_fname, user_lname`
Primary key: `user_ID`
Not null: `user_ID, user_email, user_fname, user_lname, user_pass`

This table stores all the information about the user on their id, email, first and last name, password, number of clothes purchased by the user and whether they are just a user or they have admin access. Using these, it helps the person to login and also keep track of the purchases made by them and helps with what view they are eligible to access, such as if they are an admin, they have a different webpage view. The primary key is the user_ID which is unique to every person.

Clothes

`Clothes(cloth_ID: INT(11) , cloth_name: VARCHAR(50), cloth_type: VARCHAR(20), year_released: DATE(YYYY-MM-DD), designer: VARCHAR(20), price: DOUBLE(5,2), stock_qty: INT(11), image: VARCHAR(40), num_sold: INT(5))`

Foreign keys: `none`
Candidate key: `cloth_name`
Primary key: `cloth_ID`
Not null: `cloth_ID, cloth_name, cloth_type, year_released, designer, price`

This table stores the information on all the clothes that are on the database. It includes a unique cloth_ID which is the primary key. It has information on the cloth’s id, name, type, the year it got released, the designer of the cloth, its price, quantity and the number of clothes that got sold so far. When a user is trying to purchase a cloth, information on its availability and the cloth type they belong to and other information such as its price are obtained from this table.

Orders

`Orders(order_ID: INT(11), user_email: VARCHAR(50), cloth_ID: VARCHAR(20), order_date: DATE(YYYY-MM-DD), cloth_name: VARCHAR(20), order_confirmation: VARCHAR(40), user_ID: INT(11)`

Foreign keys: `cloth_ID(Clothes.cloth_ID), user_ID (Users.user_ID)`
Candidate key: `order_confirmation`
Primary key: `cloth_ID`
Not null: `cloth_ID, cloth_name, cloth_type, year_released, designer, price`

This table stores the information on the orders made by the user. It consists of the user’s email and their ID to identify the user and has information on the cloth’s name and ID that they are purchasing. It further expands to the date of purchase, order id and an order confirmation number which acts as the candidate key as it generates unique combinations so that it does not repeat the same order confirmation number again. When a user makes a purchase, this table stores the information on the order that was placed by the user.

V. SQL Queries

Home Page Queries

Either when a signed in user or a guest user who is not signed in when visiting the home page they will see our website store cloth according to our database sales.

```
$product_array = $db_handle -> runQuery("SELECT * FROM
clothes ORDER BY num_sold DESC");
```

Which calls the function in our **dbController.php**:

```
function runQuery($query)
{
        $result = mysqli_query($this -> conn, $query);
        while($row = mysqli_fetch_assoc($result))
        {
                $resultset[] = $row;
        }
        if(!empty($resultset))
        {
                return $resultset;
        }
}
```

This collects the list of clothes that have sold the most and then sorts them in descending order. Therefore the front end developers can decide how many they want to display in what way.

If a user has anything in their cart signed in or not, the home page keeps the shopping cart while the user session is active. The user's cart is saved to the session not to the database, but keeps a track of the cloth_id's and queries to keep the shopping cart updated for every page the user visits.

```
$productByCode = $db_handle -> runQuery("SELECT * FROM clothes
WHERE cloth_ID = '" . $_GET["cloth_ID"] . "'");
```

Which again, calls the above function.

Jean Page Queries

Similar to the home page, jean page queries clothes by jean type or similar type clothes for the user to browse. Also will collect all clothes and can be displayed however the front end developer wishes.

```
$product_array = $db_handle -> runQuery("SELECT * FROM clothes
WHERE cloth_type = 'my-jean' OR cloth_type = 'FPS' OR
cloth_type = 'jean' ORDER BY cloth_name");
```

Which calls the function in our **dbController.php**:

```
function runQuery($query)
{
        $result = mysqli_query($this -> conn, $query);
        while($row = mysqli_fetch_assoc($result))
        {
                $resultset[] = $row;
        }
        if(!empty($resultset))
        {
                return $resultset;
        }
}
```

The returned results are sent back to the action **page.php**.

Shirt Page Queries

Similar to the home page, shirt page queries clothes by shirt type clothes for the user to browse. Also will collect all clothes and can be displayed however the front end developer wishes:

```
$product_array = $db_handle -> runQuery("SELECT * FROM clothes
WHERE cloth_type = 'shirt'");
```

Which calls the function in our **dbController.php**:

```
function runQuery($query)
{
```

```
            $result = mysqli_query($this -> conn, $query);
            while($row = mysqli_fetch_assoc($result))
            {
                    $resultset[] = $row;
            }
            if(!empty($resultset))
            {
                    return $resultset;
            }
    }
```

The returned results are sent back to the fighter **page.php**.

Shoe Page Queries

Similar to the home page, shoe page queries clothes by shoe type or similar type clothes for the user to browse. Also will collect all clothes and can be displayed however the front end developer wishes.

```
    $product_array = $db_handle -> runQuery("SELECT * FROM clothes
    WHERE cloth_type = 'shoes' OR cloth_type = 'FPS' OR cloth_type
    = 'shoees' ORDER BY cloth_name");
```

Which calls the function in our **dbController.php**:

```
    function runQuery($query)
    {
            $result = mysqli_query($this -> conn, $query);

            while($row = mysqli_fetch_assoc($result))
            {
                     $resultset[] = $row;
            }
            if(!empty($resultset))
            {
                    return $resultset;
            }
    }
```

The returned results are sent back to the action **page.php**.

New User Queries

The new user page is used to create a new user and needs to check the database to see if there exists a user already with the email the user provides so as to not have duplicate email addresses. If not then the newly created user can be inserted into the database. The newly created user is required to have a first name, last name, email, and password. Otherwise the information will not be inserted into the database.

Our SQL prepared statement:

```
$sql = "SELECT user_ID FROM user WHERE user_email = ?";
```

Combining to to our database connection:

```
$stmt = mysqli_prepare($link, $sql);
```

Our SQL execution statement:

```
mysqli_stmt_execute($stmt);
```

Then results are checked if the email exists, if not then we can proceed with the following insert.

```
$sql = "INSERT INTO user (user_email, user_fname, user_lname,
user_pass, num_purchased_clothes, is_admin)  VALUES (?, ?, ?, ?,
0, 0)";
```

Combining to to our database connection:

```
$stmt = mysqli_prepare($link, $sql);
```

Executing the query:

```
$result = mysqli_query($conn, $stmt);
```

Once all statements are executed successfully, there will be a new user in the database and the user will be able to sign into their account.

Log In Queries

The login page is used to log users into the website, which needs to check if the user trying to sign in is actually a user. The query will check if what the user types in matches what is in the database, otherwise the user cannot sign into their account. The user will enter their email and password.

Our SQL prepared statement:

```
$query = "SELECT user_email, user_fname, user_pass,
is_admin FROM user WHERE user_pass = '".$_POST["user_pass"]."'
AND user_email = '".$_POST["user_email"]."'";
```

Combining to to our database connection:

```
$stmt = mysqli_prepare($link, $sql);
```

Our SQL execution statement:

```
$result = mysqli_query($conn, $query);
```

Then results are checked if the email exists, if not then we can proceed with the following insert:

```
$query = "SELECT user_email, user_fname, user_pass, is_admin
FROM user WHERE user_pass = v '".$_POST["user_pass"]."' AND
user_emailv = '".$_POST["user_email"]."'";
```

Combining to to our database connection:

```
$stmt = mysqli_prepare($link, $sql);
```

Executing the query:

```
$result = mysqli_query($conn, $stmt);
```

Once all statements are executed successfully, there will be a new user in the database and the user will be able to sign into their account.

Manager Add Cloth Queries

The manager page has an add cloth selection for managers to add new clothes products they want to sell on the website easily. It accepts a clothing name, price, designer, year released, initial stock quantity, clothing type, and clothing image. Just like new users we need to see if this clothing is already in the database, so first we need to check if the clothing already exists before putting a duplicate cloth into the database.

Our SQL prepared statement:

```
$sql = "SELECT cloth _name FROM clothes WHERE cloth_name = ?";
```

Combining to to our database connection:

```
$stmt = mysqli_prepare($link, $sql)
```

Executing the query:

```
$result = mysqli_query($conn, $stmt)
```

If the cloth is not in the database we can go ahead and insert:

```
$sql = "INSERT INTO clothes (cloth_name, cloth_type,
year_released, designer, price, stock_qty, image,
num_sold) VALUES (?, ?, ?, ?, ?, ?, ?, 0)";
```

Combining to to our database connection:

```
$stmt = mysqli_prepare($link, $sql)
```

Executing the query:

```
$result = mysqli_query($conn, $stmt)
```

Once all statements are executed successfully, there will be a new cloth in the database and users will be able to see a new cloth.

Manager Change Cloth Price Queries

The manager page has a change cloth price selection for managers to change current cloth product prices if they need to do a sale or just mark a cloth product down or up. It accepts a cloth_name, cloth_id, and new price. If the cloth does not exist the manager will get an error that the cloth does not exist. Otherwise, the manager can execute the change price query.

Our SQL prepared statement:

```
$sql = "SELECT cloth_name FROM clothes WHERE cloth_name =
? && cloth_ID = ?";
```

Combining to to our database connection:

```
$stmt = mysqli_prepare($link, $sql)
```

Executing the query:

```
$result = mysqli_query($conn, $stmt)
```

If the cloth exists, then it can continue to change the price of the cloth :

```
$sql = "UPDATE clothes SET price = ? WHERE (cloth_ID = ?
&& cloth_name = ?)";
```

Combining to to our database connection:

```
$stmt = mysqli_prepare($link, $sql)
```

Executing the query:

```
$result = mysqli_query($conn, $stmt)
```

Once all statements are executed successfully, the cloth that the manager wanted to update its price to will be updated.

Manager Update Cloth Quantities Queries

The manager page has a change cloth quantity for managers to change current cloth product quantities if they receive a shipment. Negative numbers can even be input to clear out current quantity if an error was made counting. It accepts a cloth_name, cloth _id, and new quantity. If the cloth does not exist the manager will get an error that the cloth does not exist. Otherwise, the manager can execute the change price query.

Our SQL prepared statement:

```
$sql = "SELECT cloth_name, stock_qty FROM clothes WHERE
cloth_name = '".$_POST["cloth _name"]."' && cloth_ID =
'".$_POST["cloth_id"]."'";
```

Combining to to our database connection:

```
$stmt = mysqli_prepare($link, $sql)
```

Executing the query:

```
$result = mysqli_query($conn, $stmt)
```

If the clothing exists, then it can continue to change the price of the cloth

:

```
$sql = "UPDATE clothes SET stock_qty = ? WHERE (cloth_ID =
? && cloth_name = ?)";
```

Combining to to our database connection:

```
$stmt = mysqli_prepare($link, $sql)
```

Executing the query:

```
$result = mysqli_query($conn, $stmt)
```

Once all statements are executed successfully, the cloth that the manager wanted to update the cloth stock quantity.

Manager Delete Queries

The manager page has to delete a selected cloth from the database if it exists. It accepts a cloth name, and cloth_id. If the cloth does not exist the manager will get an error that the clothing does not exist. Otherwise, the manager can execute the query and delete the cloth.

Our SQL prepared statement:

```
$sql = "SELECT cloth_name FROM clothes WHERE cloth_name =
? && cloth_ID = ?";
```

Combining to to our database connection:

```
$stmt = mysqli_prepare($link, $sql)
```

Executing the query:

```
$result = mysqli_query($conn, $stmt)
```

If the clothing exists, then it can continue to delete the cloth:

```
$sql = "DELETE FROM clothes WHERE (cloth_ID = ? &&
cloth_name = ?)";
```

Combining to to our database connection:

```
$stmt = mysqli_prepare($link, $sql)
```

Executing the query:

```
$result = mysqli_query($conn, $stmt)
```

Once all statements are executed successfully, the cloth that the manager wanted to delete will be deleted from the database and no longer part of the cloth store available clothes.

Purchase Queries

Once the user signed in or not wants to complete a purchase on the website it takes the current shopping cart cloth IDs to use to query the database to collect those cloth to finalize the sale. There are several different queries in the purchase page. During a purchase we need to figure out if a purchase is being completed by a user with the given email address. Because we want to know what users purchase what clothes and or type later. Also, we want to later reward users who purchase clothes with discounts determined by how many clothes they purchase. So the purchase page is executed in two different directions, one records if the purchase is made by a user or if it's by a non account user by email.

Additionally, the purchase page has to make sure the cloth quantity is greater than 0, meaning it's in stock. If not the user is told it will be in stock soon and cancels the order.

Our SQL prepared statement from our shopping cart:

```
$sql = "SELECT cloth_name, stock_qty, num_sold FROM
clothes WHERE cloth_name = '$cloth_name' && cloth_ID =
'$cloth_id'";
```

Used to see if the user purchasing is a account holder:

```
$sql5 = "SELECT * FROM user WHERE user_email =
'$user_email'";
```

Updating cloth stock after purchase:

```
$sql = "UPDATE clothes SET stock_qty = ?, num_sold = ?
WHERE (cloth_ID = ? && cloth_name = ?)";
```

Inserting into the orders table:

```
$sql2 = "INSERT INTO orders (user_email, cloth_ID,
order_date, cloth_name) VALUES (?, ?, ?, ?)";
```

Used to update the user account if the user purchasing user is an account holder:

```
$sql4 = "UPDATE user SET num_purchased_clothes = ?
WHERE(user_email = ?)";
```

Used to update the final purchase confirmation code for the user:

```
$sql3 = "UPDATE orders SET order_confirmation = ? WHERE
(user_email = ? && order_date = ?)";
```

Combining to to our database connection:

```
$stmt = mysqli_prepare($link, $sql)
```

Executing the query:

```
$result = mysqli_query($conn, $stmt)
```

Once all statements are executed successfully, the cloth that the manager wanted to delete will be deleted from the database and no longer part of the cloth store available clothes.

VI. <u>Database Initialization Queries</u>

The following four SQL queries are used to initialize the database tables:

```
CREATE TABLE IF NOT EXISTS `clothes` (
	`cloth_name` varchar(50) NOT NULL,
	`cloth_ID` int(11) NOT NULL AUTO_INCREMENT,
	`cloth_type` varchar(20) NOT NULL,
	`year_released` date NOT NULL,
	`designer` varchar(20) NOT NULL,
	`price` double(5, 2) NOT NULL,
	`stock_qty` int(11) NOT NULL,
	`image` varchar(40) NOT NULL,
	`num_sold` int(5) NOT NULL,
PRIMARY KEY (`cloth_ID`) )
ENGINE = InnoDB AUTO_INCREMENT = 41 DEFAULT CHARSET = latin1;


CREATE TABLE IF NOT EXISTS `orders` (
	`order_id` int(11) NOT NULL AUTO_INCREMENT,
	`user_email` varchar(40) NOT NULL,
	`cloth_ID` varchar(20) NOT NULL,
	`order_date` date NOT NULL,
	`cloth_name` varchar(20) NOT NULL,
	`order_confirmation` varchar(40) DEFAULT NULL,
PRIMARY KEY (`order_id`) )
ENGINE = InnoDB AUTO_INCREMENT = 37 DEFAULT CHARSET = latin1;


 CREATE TABLE IF NOT EXISTS `user`(
       `user_ID` int(11) NOT NULL AUTO_INCREMENT,
       `user_email` varchar(40) NOT NULL,
       `user_fname` varchar(20) NOT NULL,
       `user_lname` varchar(20) NOT NULL,
       `user_pass` varchar(20) NOT NULL,
       `num_purchased_clothes` int(20) NOT NULL,
       `is_admin` tinyint(1) NOT NULL,
PRIMARY KEY (`user_ID`) )
ENGINE = InnoDB AUTO_INCREMENT = 19 DEFAULT CHARSET = latin1;
```