**Amazon:**

Search for the item you want, this will query the database using a SELECT query and print out all the matching results on the page.

By default these results are ORDERed by relevance and have 24 items per page, most likely using the LIMIT 24 keyword.

An example of this query would be:

*SELECT \* FROM items*

*WHERE (item\_name LIKE ‘%Christmas\*Decorations%’*

*OR item\_description LIKE ‘%Christmas\*Decorations%’)*

*LIMIT 24;*

The \* (Asterisk) is a wildcard meaning ‘anything’. I’ve used it here as spaces don’t show up in LIKE queries, and without it, would only search for ‘ChristmasDecorations’. An Asterisk will SELECT anything such as ‘Christmas Decorations’ to ‘Christmas\_Decorations’ which is good enough for an example.

This would return the first 24 items that had either Christmas Decorations in their name, or in their description.

Once an item has been selected, you’re then presented with all the item information, ranging from description to price, and reviews. You have to select a quantity from a combo box and then add it to your basket.

An example query for returning all the information would be similar to:

*SELECT \* FROM items*

*WHERE item\_id=‘1234567’*

The page is mostly written in javascript, so it would just pull the values from the rows this query had selected, and place them accordingly on the screen.

An example query for adding items to the basket would be:

*INSERT INTO basket (‘item\_id’, ‘item\_quantity’)*

*VALUES (‘1234567’. ‘2’);*

Or if the item\_id already exists in basket:

*UPDATE basket*

*SET item\_quantity=item\_quantity + 1*

*WHERE item\_id=‘1234567’;*

Once added to your basket, you’re presented with the option of editing your basket.

Here you can remove items completely, or just change the quantity you would like to purchase.

An example query of removing the item completely would be:

*DELETE FROM basket*

*WHERE (item\_id=‘1234567’);*

Or simply changing the quantity:

*UPDATE basket*

*SET item\_quantity=item\_quantity - 1*

*WHERE item\_id=‘1234567’;*

Once you’re happy with your basket, it will ask you to sign up or sign in. If you haven’t got an account with Amazon you’re going to have to fill out a form containing all your personal information.

**[Your name]**

**[Email]**

**[Password]**

**[Password Confirmation]**

These would all then be (simply) INSERTed into the users table for you to be able to login in.

Pseudo code for the login would be similar to:

*query = (“SELECT \* FROM users WHERE email=@email AND password=@password”);*

*query.parameter(“@email”) = emailbox.text;*

*query.parameter(“@password”) = passwordbox.text;*

*query.run();*

*if (query.rows == 1)*

*{*

*Login();*

*}*

*else*

*{*

*InvalidLogin();*

*}*

Hopefully, they won’t actually save your password on the database, and will instead, encrypt the password and save the encryption on the database.

An example would be to encrypt a password in SHA-256:

*password = ‘password123’*

*hash = ef92b778bafe771e89245b89ecbc08a44a4e166c06659911881f383d4473e94f*

Despite how secure you think your database is, there is always a chance that you could have missed an exception and to have all your plain text passwords out in the open for hackers is a lawsuit waiting to happen.

The same goes for card details. Ideally, you don’t want these being saved onto the database and instead use a 3rd party payment provider, such as PayPal.

After logging in, you’ll be at the checkout screen where you enter all your payment details and your address for delivery.

As you can have multiple delivery addresses, I'm assuming these will be saved on a separate table linked to your user\_id.

List of Entities with attributes:

**item** = { *item\_id [PK], seller\_id, item\_name, item\_description, item\_price, item\_stock, isPrime, item\_seller, item\_dimensions, item\_weight, item\_brand, item\_rating* }

**customer** = { *customer\_id [PK], customer\_email, password\_hash, customer\_name* }

**seller** = { *seller\_id [PK], seller\_name, seller\_location, seller\_rating* }

**delivery address** = { *customer\_id [FK], address\_full\_name, address\_line1, address\_line2, city, country, postcode, phone\_number* }

**order** = { *order\_id [PK], customer\_id [FK], total\_cost* }

**orderitems** = { *order\_id [FK], item\_id [FK], item\_quantity* }