AC32006 / AC52001 - Database Systems

Coursework 1 - Cover Sheet

TEAM NUMBER 12 COMPANY NAME Master Race Parts

TEAM MEMBERS Alek Alexiev 170011640
Elisaveta Peeva 180012881
Heidi Bruce 150014288
Laura Hutchison 180075929
Preslav Chonev 180007405
Taha Kashaf 180020889

☐ Company description / Specification report	
Word count (900-1000) 943	**Number of pages 2**
☐ E-R diagram	**Number of pages 1**
☐ User interface designs	**Number of pages 5**
☐ SQL CREATE statements	**Number of pages 5**

	Total pages 13

Specification Report

Master Race Parts is a multi-branch computer hardware store that caters to enthusiasts and newcomers alike. While the store has multiple retail branches, it also allows for customers to purchase products online via the store's website. Online purchases require users to have an account, and users with an account are able to purchase in-store using the same account to streamline the in-store experience.

Customers who do not have an account are required to make one to complete an online checkout but are not required to make one to complete an in-store checkout. Nonetheless, the option to register at checkout is given to them.

The way we handle all this on the backend via our database is through four tables. Our database has more than four tables, but the four that handle the aforementioned retail experience are as follows, with the structure and logic also defined.

The tables are *Customer*, *Address*, *Payment* and *Order*. At the time of purchasing online, the customer is required to make an account (if they are not already logged in). The details they are required to give are fairly standard, first and last name, email, phone and address. These are stored in a *Customer* table, with each customer being given a unique ID at the time of creation that acts as the tables primary key. This is straightforward enough for online purchases.

However, if the customer is purchasing in store, they might opt not to register an account, and just make a single order. If this is the case, the order would still be stored in the *Order* table as normal, but the customerID foreign key that links the *Order* table to the *Customer* table would be null.

The reason we have opted to track customers and orders in this way is because it is important to store what orders are made, and who is making them - for customer service in the case of potential problems. As well as analytics and marketing, to see what products are selling well, and to what demographic. A secondary purpose to this system is to retain a consistent experience at checkout for registered users, be it in store or online. While also allowing one-time customers to smoothly shop at our store whilst keeping a record of the orders they make as well. Another final aspect of tracking the orders is stock, a crucial point for any retail chain.

Stock tracking is done via two main tables, and two linking tables. These are *Branch*, *Product*, *Availability_in_branch* and *Product_Order*, respectively. First, the previously mentioned table that tracks all orders made by customers, has a one-to-many link to the linking table *Product_Order*. This linking table is then linked to *Product* (another many-to-one connection). This is because a single order may contain multiple different types of products, as well as a multiple orders containing the same type of product. Hence, necessitating the use of a linking table

The *Product* table just stores information about the product. SKU, type, brand, supplier, etc. Whilst the *Product_Order* linking table stores orderID (from the *Order* table), the SKU (from the *Product* table) as well the quantity of the product purchased. This is vital for real-time stock tracking.

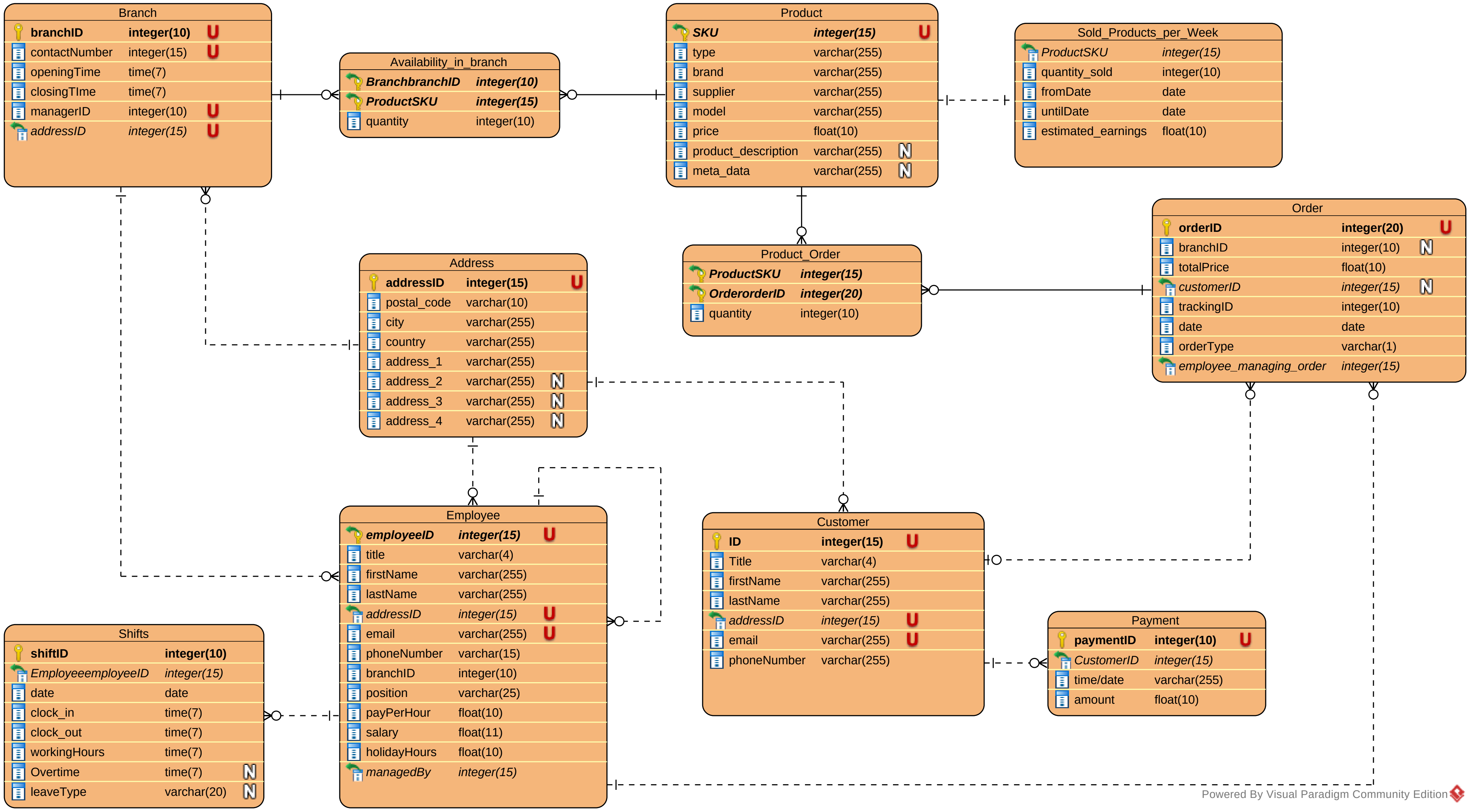
We mentioned at the start that we have multiple branches. As a result of this, stock is spread out amongst these branches. Which leads to another potential problem - we need to be able to track not only what is in stock, but where. Therefore, we have the *Availability_in_branch* linking table that links *Product* and *Branch*. This works in a similar way to the *Product_Order* linking table described in the previous paragraph as it is solving the same type of problem (multiple products would be in stock independently at multiple different branches). The linking table stores branchID, ProductSKU, as well as quantity. To easily track availability of products at any specific branch.

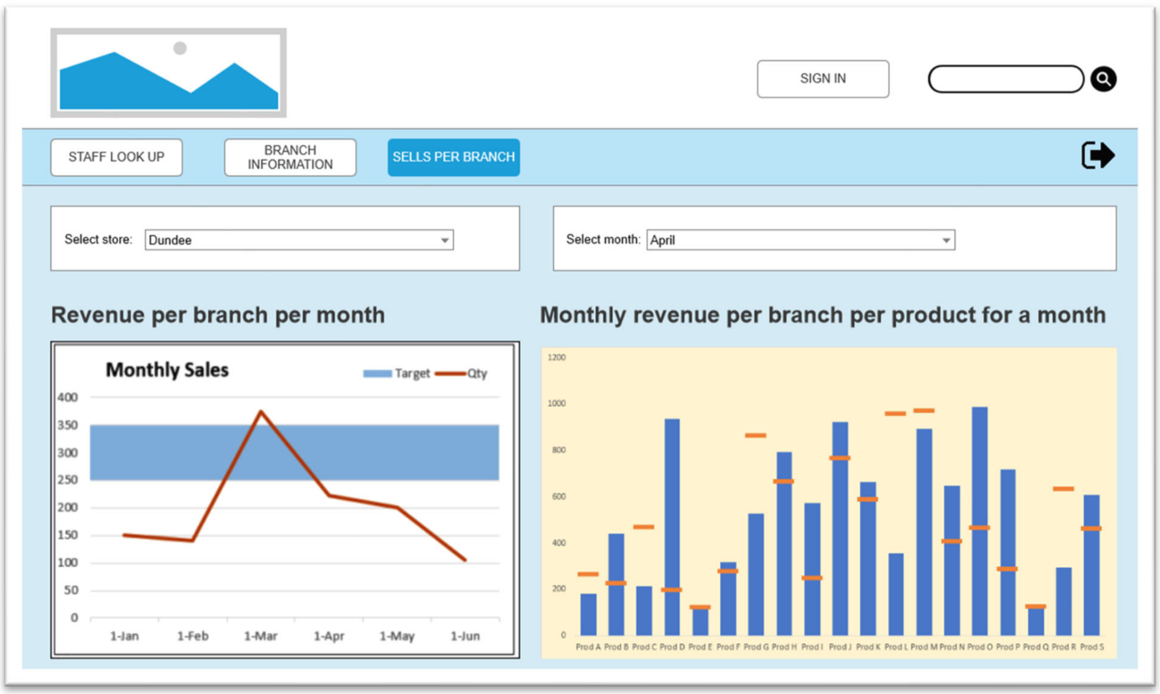
So far we've mentioned customers, checkout experience, stock tracking, but there is also the case of the staff at our stores. We have an *Employee* table that tracks the relevant information about all our staff. This includes title, first and last name, address, email, position, pay, etc.

This *Employee* table is the final major piece of our database design. It links to several of the other tables for several different purposes. First, there is the obvious task of tracking what staff members work where, what their roles are, and other general shift information. This task is made possible by the *Employee* table linking both to *Branch* as well as a *Shift* table. The link to *Branch* is a many to one link, allowing us to track what staff members work at what branch. In a similar way, the *Shift* table links back to *Employee* to track what employees worked what shifts.

The *Employee* table is also using a recursive relationship to refer to itself, as the manager position is stored in the table like any other employee would be, but the managers have the ability to access the table (via a Management System, of course) to make changes to employee details.

Finally, the *Employee* table is linked to the *Order* table mentioned at the start of the report with a simple one-to-many link. This is because the store has a policy that every order made, must be managed/associated with an employee. Either the person that made the sale, or the staff member working checkout at the time. This both helps managers monitor employee output as well as aiding in customer service problems should they arise. For example, if a customer has a complaint about an order, we would be able to retroactively track the employee who made the sale.


Branch
branchID integer(10) U
contactNumber integer(15) U
openingTime time(7)
closingTIme time(7)
managerID integer(10) U
addressID integer(15) U
Availability_in_branch
BranchbranchID integer(10)
ProductSKU integer(15)
quantity integer(10)
Product
SKU integer(15) U
type varchar(255)
brand varchar(255)
supplier varchar(255)
model varchar(255)
price float(10)
product_description varchar(255) N
meta_data varchar(255) N
Sold_Products_per_Week
ProductSKU integer(15)
quantity_sold integer(10)
fromDate date
untilDate date
estimated_earnings float(10)
Address
addressID integer(15) U
postal_code varchar(10)
city varchar(255)
country varchar(255)
address_1 varchar(255)
address_2 varchar(255) N
address_3 varchar(255) N
address_4 varchar(255) N
Product_Order
ProductSKU integer(15)
OrderorderID integer(20)
quantity integer(10)
Order
orderID integer(20) U
branchID integer(10) N
totalPrice float(10)
customerID integer(15) N
trackingID integer(10)
date date
orderType varchar(1)
employee_managing_order integer(15)
Employee
employeeID integer(15) U
title varchar(4)
firstName varchar(255)
lastName varchar(255)
addressID integer(15) U
email varchar(255) U
phoneNumber varchar(15)
branchID integer(10)
position varchar(25)
payPerHour float(10)
salary float(11)
holidayHours float(10)
managedBy integer(15)
Customer
ID integer(15) U
Title varchar(4)
firstName varchar(255)
lastName varchar(255)
addressID integer(15) U
email varchar(255) U
phoneNumber varchar(255)
Shifts
shiftID integer(10)
EmployeeemployeeID integer(15)
date date
clock_in time(7)
clock_out time(7)
workingHours time(7)
Overtime time(7) N
leaveType varchar(20) N
Payment
paymentID integer(10) U
CustomerID integer(15)
time/date varchar(255)
amount float(10)
Powered By Visual Paradigm Community Edition

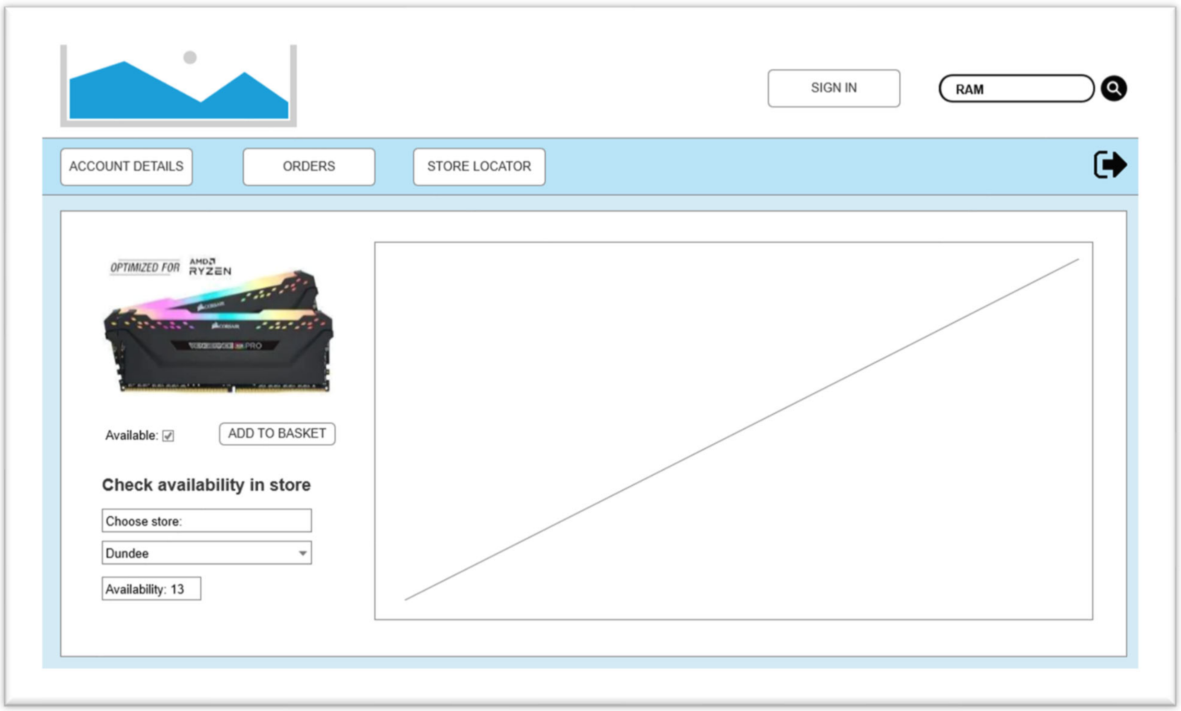

SIGN IN
STAFF LOOK UP
BRANCH INFORMATION
SELLS PER BRANCH
Revenue per branch per month
Monthly Sales
Target
Qty
1-Jan
1-Feb
1-Mar
1-Apr
1-May
1-Jun
Monthly revenue per branch per product for a month
Prod A Prod B Prod C Prod D Prod E Prod F Prod G Prod H Prod I Prod J Prod K Prod L Prod M Prod N Prod O Prod P Prod Q Prod R Prod S

Select store: Dundee

Select month: April

Top: Screen showing a mock up of aggregated Sold_Products_per_Week tables.

Bottom: A screen with data from a Product table, in a web page with data on a real-life product available for sale in store or online.



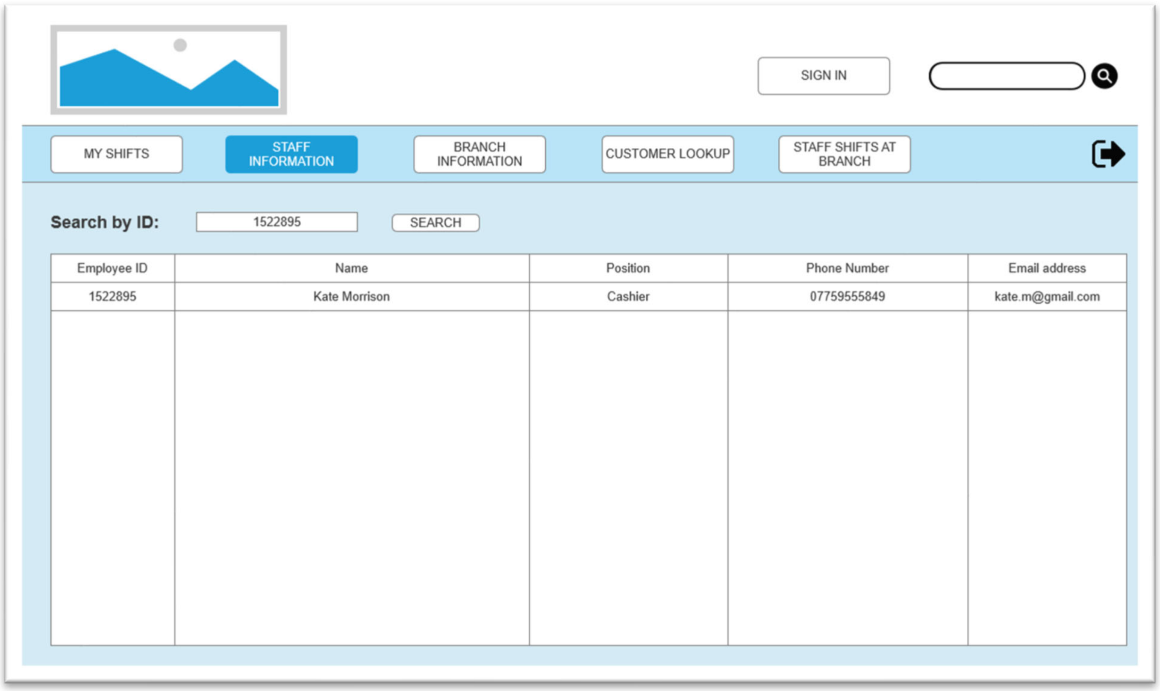
RAM

Available: ✓

Choose store:

Dundee

Availability: 13

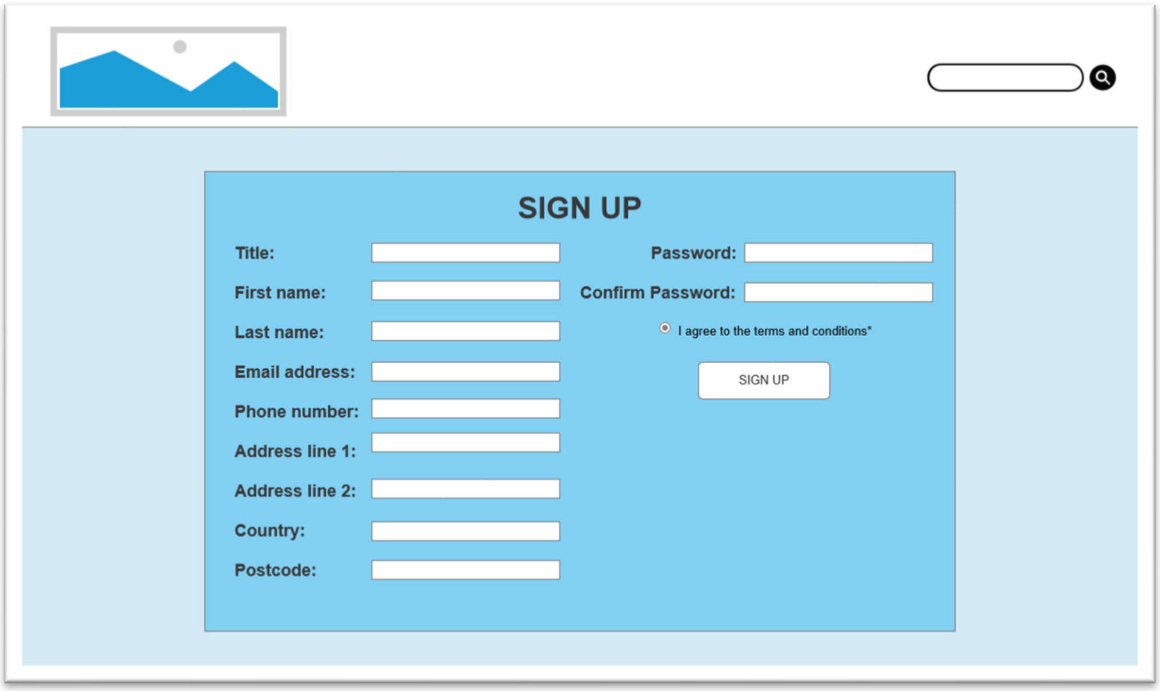


Search by ID: 1522895

Employee ID	Name	Position	Phone Number	Email address
1522895	Kate Morrison	Cashier	07759555849	kate.m@gmail.com

Top: A screen showing a search by ID through Employee tables, allowing managers or other appropriate staff to search through employees.

Bottom: A screen allowing a customer to create an account in the database, which is filed into the Customer table.



SIGN UP

Title:

First name:

Last name:

Email address:

Phone number:

Address line 1:

Address line 2:

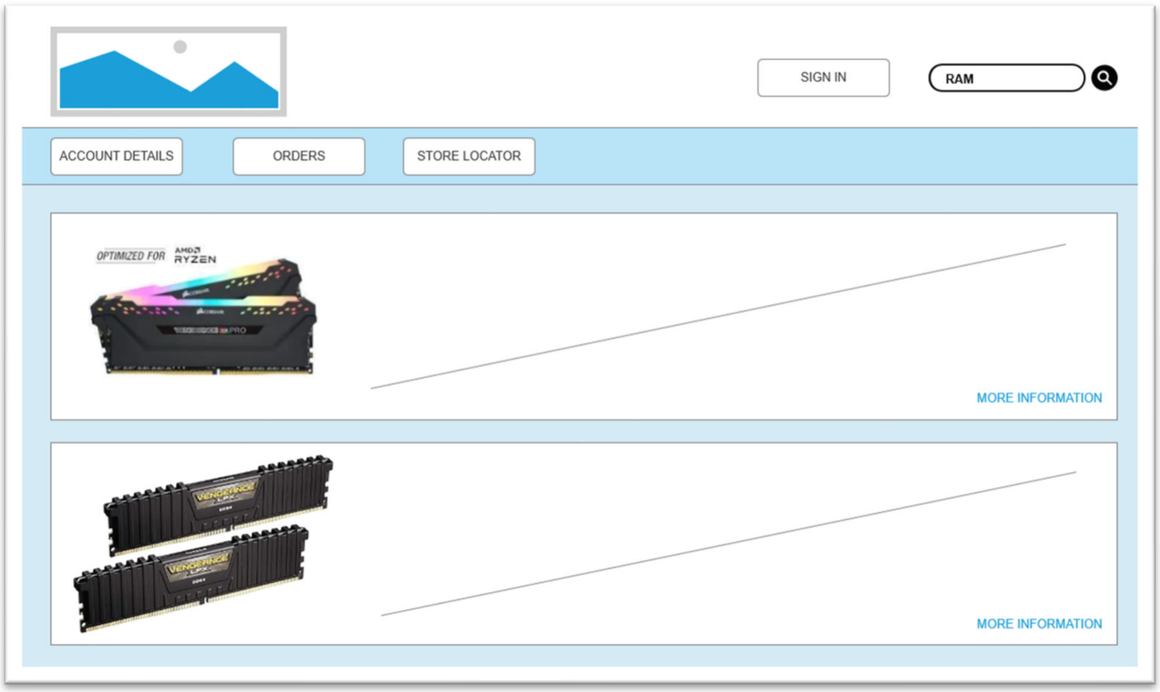
Country:

Postcode:

Password:

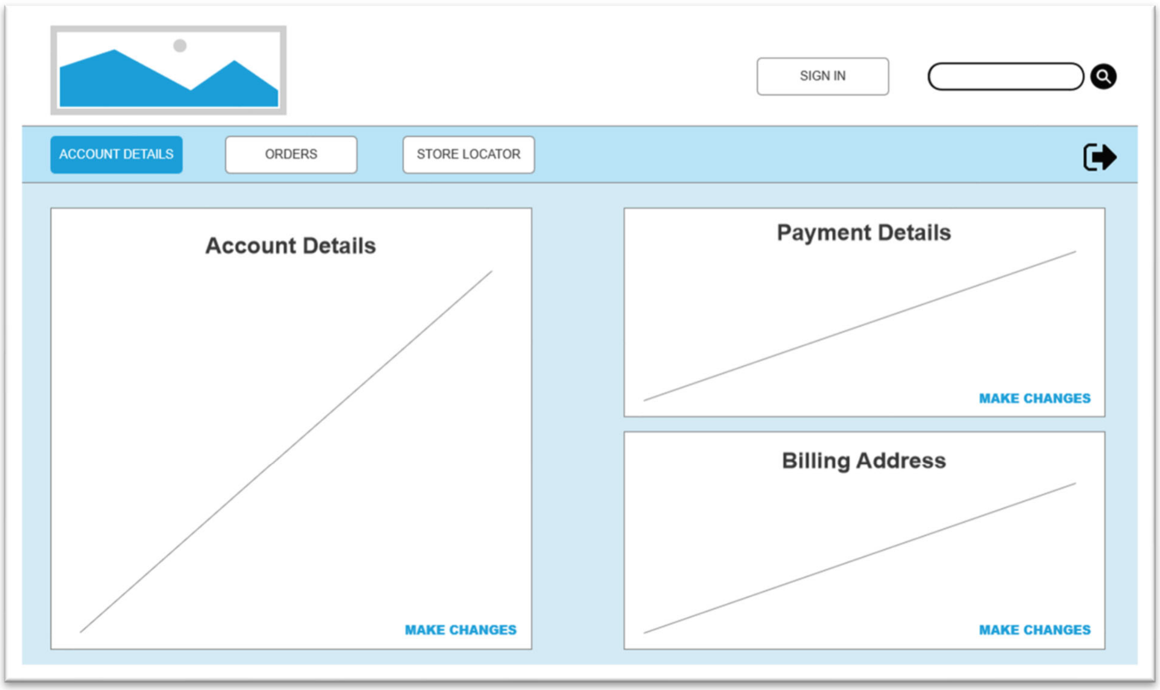
Confirm Password:

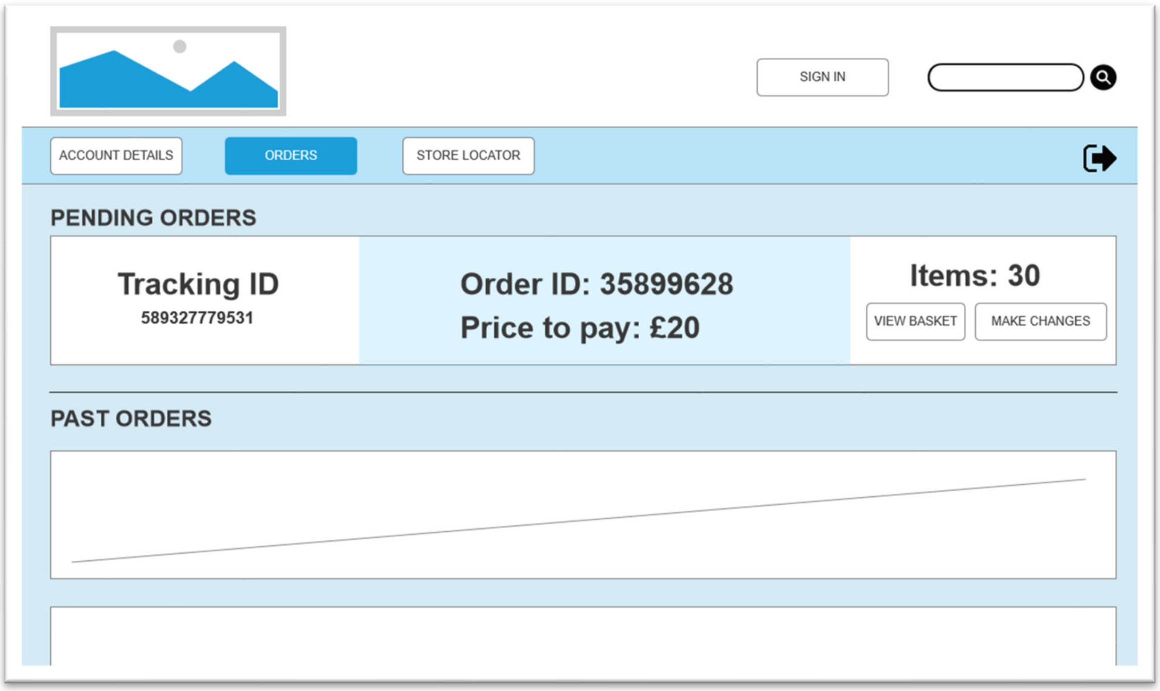
I agree to the terms and conditions*



Top: Screen showing a customer's order. The data is retrieved for the customer from the Order and Product_Order tables.

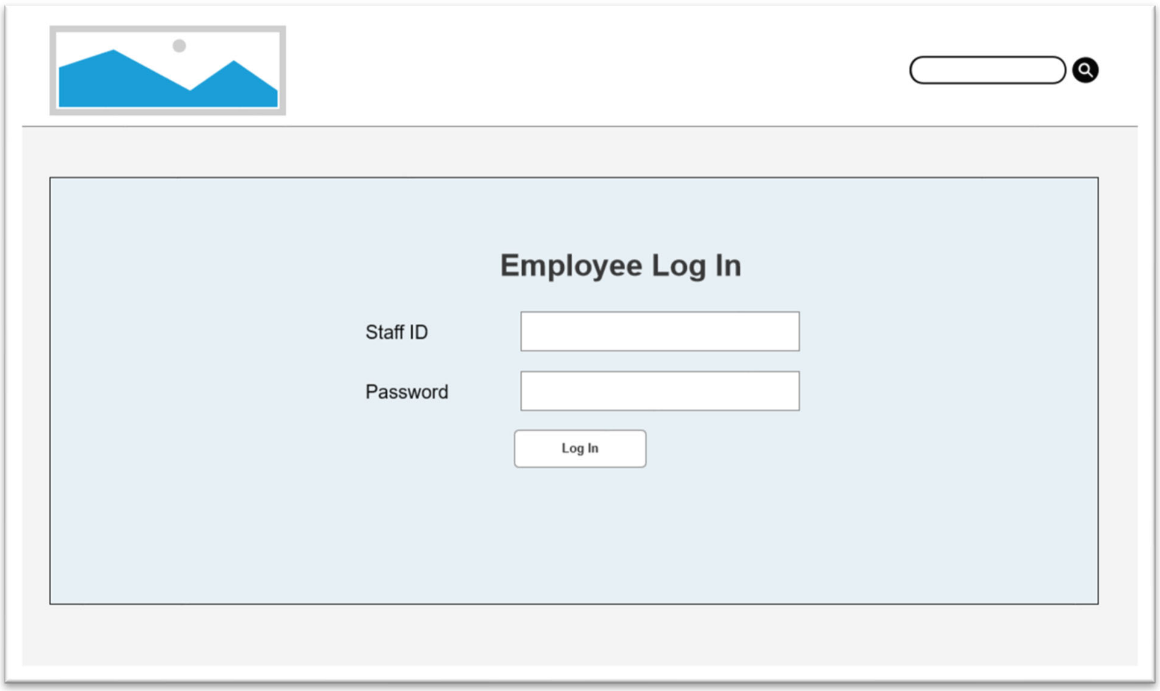
Bottom: A screen showing a customer their account details, allowing them to update or amend their details.

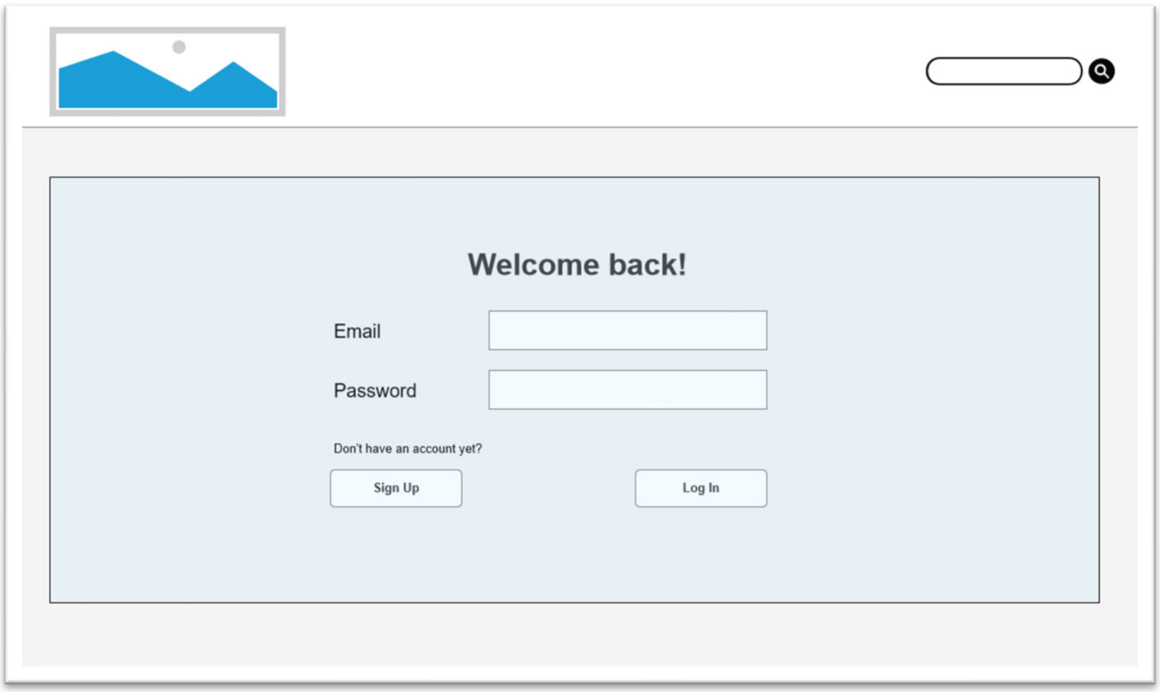




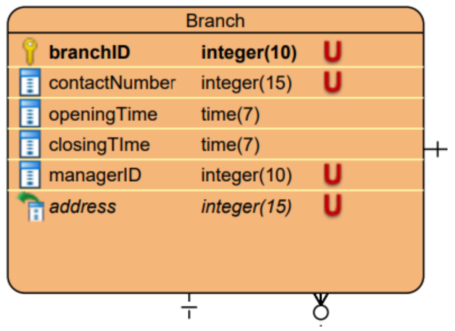
Top: A customer's order history screen. They have the option here to view past orders or make changes to their pending orders or basket.

Bottom: An employee login screen, wherein they will be able to access their data and higher-ranked employees such as managers will be able to access the data of other employees for work-related purposes.

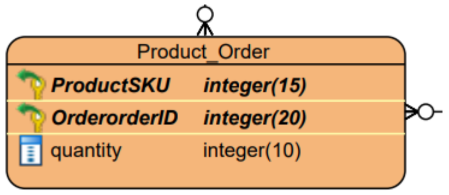




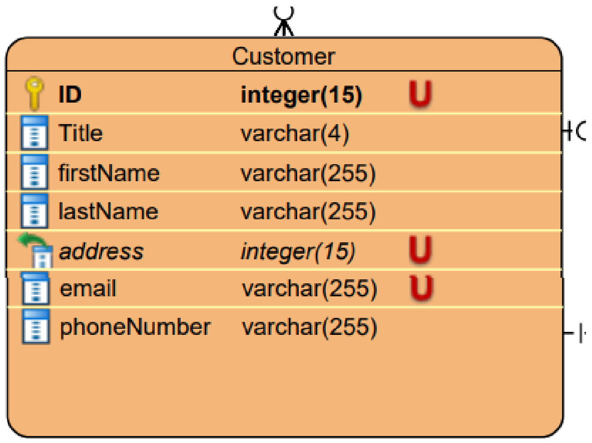
Top: A login screen for customers, where they will be able to make an account if they do not already have one.



CREATE TABLE Branch (
branchID int(10) NOT NULL,
contactNumber int(15) UNIQUE,
managerID int FOREIGN KEY REFERENCES Employee(employeeID),
openingTime time(7),
closingTime time(7),
address int FOREIGN KEY REFERENCES Address(addressID),
PRIMARY KEY (branchID),
CONSTRAINT NULL_CONDS CHECK (contactNumber IS NOT NULL)
);

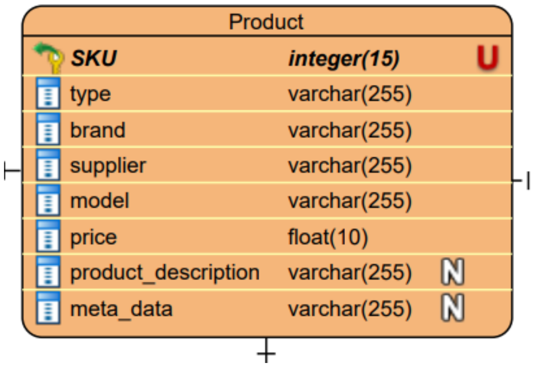


CREATE TABLE Product_Order(
ProductSKU int FOREIGN KEY REFERENCES Product(SKU),
OrderorderID int FOREIGN KEY REFERENCES Order(orderID),
quantity int(10) NOT NULL,
PRIMARY KEY (ProductSKU, OrderorderID)
);



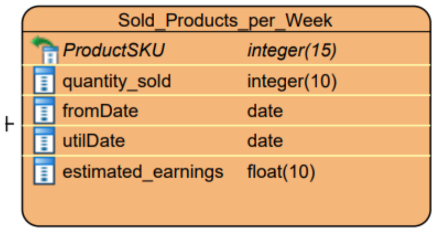
Customer		
ID	**integer(15)**	U
Title	varchar(4)	
firstName	varchar(255)	
lastName	varchar(255)	
address	*integer(15)*	U
email	varchar(255)	U
phoneNumber	varchar(255)	

```
CREATE TABLE Customer(
        customerID int(15) NOT NULL,
        Title varchar(4),
        firstName varchar(255) NOT NULL,
        lastName varchar(255) NOT NULL,
        addressID int FOREIGN KEY (addressID) REFERENCES Address(addressID),
        email varchar(255) UNIQUE,
        phoneNumber varchar(255),
        PRIMARY KEY (customerID)
);
```



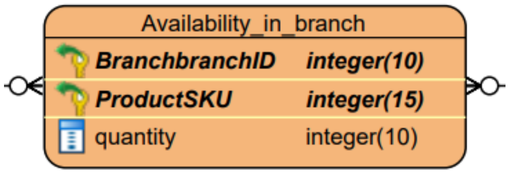
Product		
SKU	***integer(15)***	U
type	varchar(255)	
brand	varchar(255)	
supplier	varchar(255)	
model	varchar(255)	
price	float(10)	
product_description	varchar(255)	N
meta_data	varchar(255)	N

```
CREATE TABLE Product (
        SKU int(15) NOT NULL,
        type varchar(255) NOT NULL,
        brand varchar(255) NOT NULL,
        supplier varchar(255) NOT NULL,
        model varchar(255) NOT NULL,
        price float(10) NOT NULL,
        atCost float(10) NOT NULL,
        product_description varchar(255),
        meta_data varchar(255),
        PRIMARY KEY (SKU)
);
```



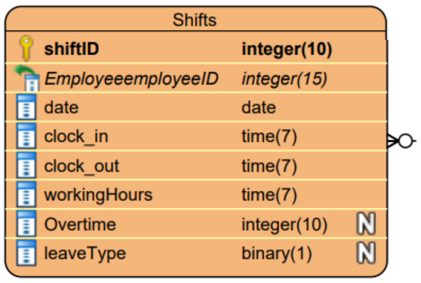
Sold_Products_per_Week	
ProductSKU	*integer(15)*
quantity_sold	integer(10)
fromDate	date
utilDate	date
estimated_earnings	float(10)

```
CREATE TABLE Sold_Products_per_Week(
            ProductSKU int FOREIGN KEY REFERENCES Product(SKU),
            quantity_sold int(10) NOT NULL,
            fromDate date NOT NULL,
            untilDate date NOT NULL,
            estimated_earnings float(10) NOT NULL,
            PRIMARY KEY (ProductSKU, fromDate, untilDate)
);
```



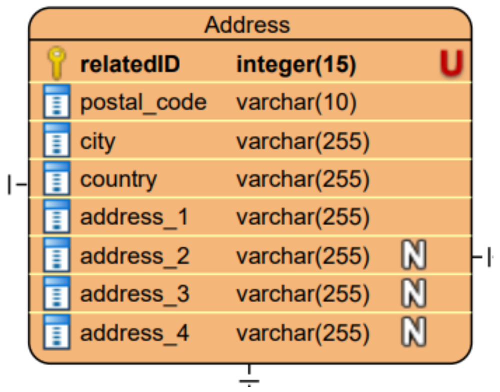
Availability_in_branch	
BranchbranchID	***integer(10)***
ProductSKU	***integer(15)***
quantity	integer(10)

```
CREATE TABLE Availability_in_branch (
            BranchbranchID int FOREIGN KEY REFERENCES Branch(branchID),
            ProductSKU int FOREIGN KEY REFERENCES Product(SKU),
            quantity int NOT NULL,
            PRIMARY KEY(BranchbranchID, ProductSKU)
```



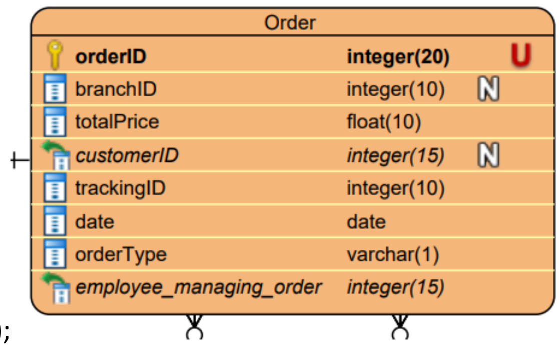
Shifts		
shiftID	**integer(10)**	
EmployeeemployeeID	*integer(15)*	
date	date	
clock_in	time(7)	
clock_out	time(7)	
workingHours	time(7)	
Overtime	integer(10)	N
leaveType	binary(1)	N

```
);
CREATE TABLE Shifts(
            shiftID int(10) NOT NULL,
            EmployeeemployeeID int(15) NOT NULL,
            date date NOT NULL,
            clock_in time(7) NOT NULL,
            clock_out time(7) NOT NULL,
            workingHours time(7) NOT NULL,
            Overtime time(7),
            leaveType binary(1),
            PRIMARY KEY (shiftID)
);
```



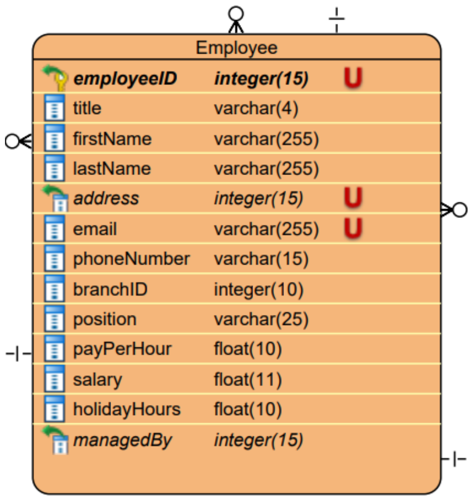
Address		
relatedID	**integer(15)**	U
postal_code	varchar(10)	
city	varchar(255)	
country	varchar(255)	
address_1	varchar(255)	
address_2	varchar(255)	N
address_3	varchar(255)	N
address_4	varchar(255)	N

```
CREATE TABLE Address (
        addressID int(15) NOT NULL,
        postal_code varchar(10) NOT NULL,
        city varchar(255) NOT NULL,
        country varchar(255) NOT NULL,
        address_1 varchar(255) NOT NULL,
        address_2 varchar(255),
        address_3 varchar(255),
        address_4 varchar(255),
        PRIMARY KEY (addressID)
```



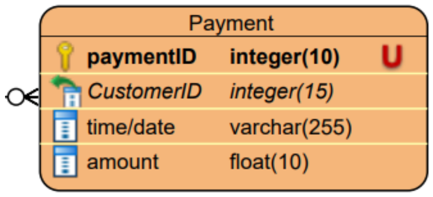
Order		
orderID	**integer(20)**	U
branchID	integer(10)	N
totalPrice	float(10)	
customerID	*integer(15)*	N
trackingID	integer(10)	
date	date	
orderType	varchar(1)	
employee_managing_order	*integer(15)*	

```
);
CREATE TABLE Order(
        orderID int(20) NOT NULL,
        branchID int FOREIGN KEY REFERENCES Branch(branchID),
        totalPrice float(10) NOT NULL,
        customerID FOREIGN KEY REFERENCES Customer(customerID),
        trackingID int(10) NOT NULL,
        date date NOT NULL,
        orderType varchar(1) NOT NULL,
        employee_managing_order int FOREIGN KEY REFERENCES Employee(employeeID),
        PRIMARY KEY (orderID)
);
```



Employee		
employeeID	***integer(15)***	U
title	varchar(4)	
firstName	varchar(255)	
lastName	varchar(255)	
address	*integer(15)*	U
email	varchar(255)	U
phoneNumber	varchar(15)	
branchID	integer(10)	
position	varchar(25)	
payPerHour	float(10)	
salary	float(11)	
holidayHours	float(10)	
managedBy	*integer(15)*	

```
CREATE TABLE Employee (
        employeeID int(15) NOT NULL,
        title varchar(4),
        firstName varchar(255) NOT NULL,
        lastName varchar(255) NOT NULL,
        address int FOREIGN KEY REFERENCES Address(addressID),
        email varchar(255) UNIQUE,
        phoneNumber varchar(15) NOT NULL,
        branchID int FOREIGN KEY REFERENCES Branch(branchID),
        position varchar(25) NOT NULL,
        payPerHour float(10) NOT NULL,
        salary float(11) NOT NULL,
        holidayHours float(10) NOT NULL,
        managedBy int FOREIGN KEY REFERENCES Employee(employeeID),
        PRIMARY KEY (employeeID),
        CONSTRAINT NULL_CONDS CHECK(email IS NOT NULL)
);
```



Payment		
paymentID	**integer(10)**	U
CustomerID	*integer(15)*	
time/date	varchar(255)	
amount	float(10)	

```
CREATE TABLE Payment (
        paymentID int(10) NOT NULL,
        CustomerID int FOREIGN KEY REFERENCES Customer(ID),
        timeAndDate varchar(255) NOT NULL,
        amount float(10) NOT NULL,
        PRIMARY KEY (paymentID)
);
```