DB Project Part 1 Part 3

The PostgreSQL account name for the database: ag4478

URL of the web application: We will submit this during the meeting.

Description of the front-end:

We made a couple of changes to the ER diagram that we had submitted originally. We removed the supplier reviews entity from the database as we felt that it did not add much value considering we already have product reviews for a product which is more relevant. Also, we linked each product to exactly one supplier which removed the step of choosing a particular supplier for a product and simplified the process of placing an order by a customer.

In our web application, there is a login/sign up page at the start. A user can login if they already have an account or can sign up. While signing up, the user enters details like name, phone number and password along with the address which consists of street, apartment number, city, state and zip.

Two of the web pages that require the most interesting database operations are the product catalog page and the past orders page. These also form the main components of the web front-end.

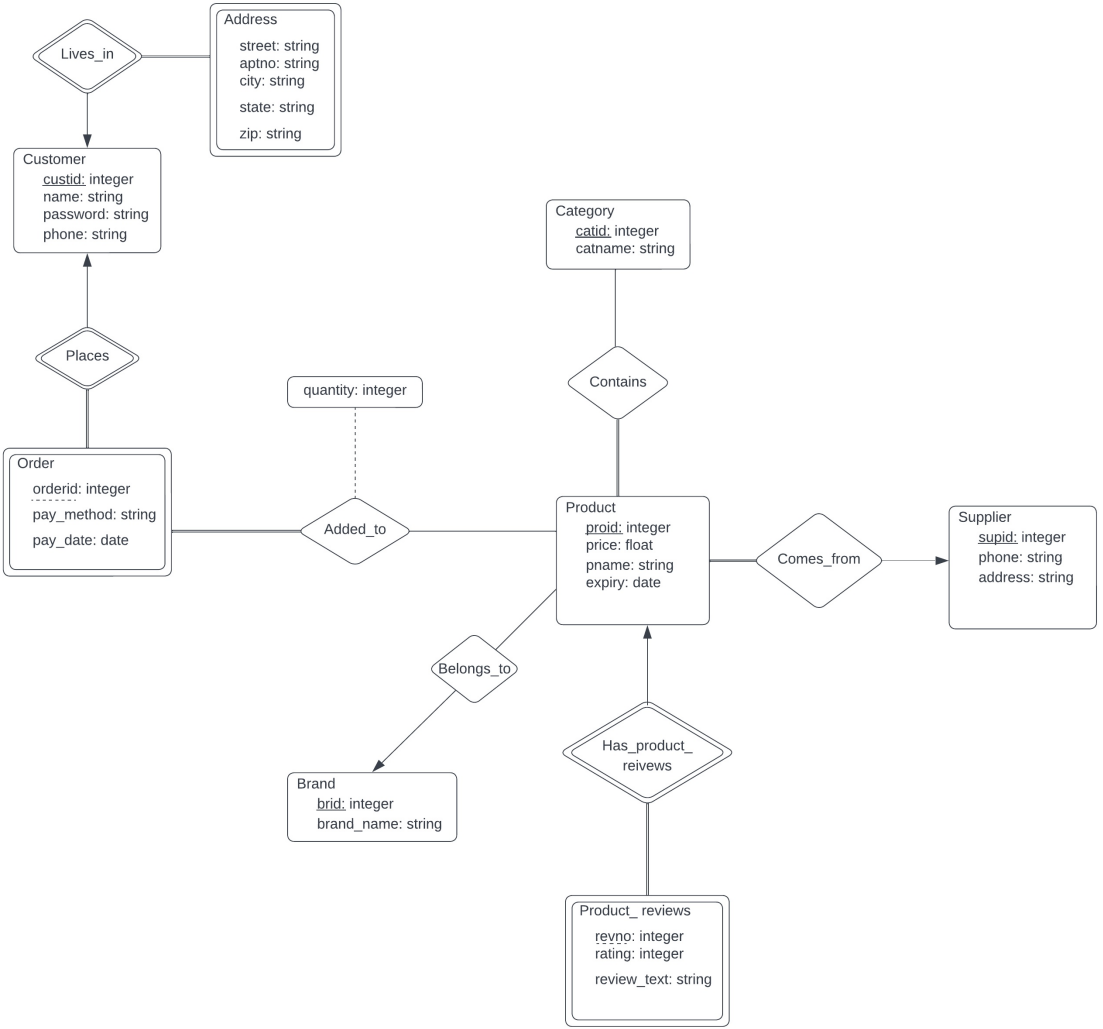
The product catalog page shows all the products in the database with details such as price, expiry date, category. A user can click on the product id of a product to view additional information about that particular product such as brand, supplier id, supplier's phone number, supplier's address, ratings and reviews. There is a quantity column for each product where the user can select the quantity of each product they want to buy. There is a

dropdown menu where the user selects the payment method and then places the order by clicking submit.

To get all information about a product, we had to do joins across various tables in the database such as product, contains, comes_from, category, brand, productreview.

The orders page shows the order id, total number of items, amount, payment method and date of all past orders by a customer. The customer can click on any order id to see the exact items in that order and their details. The vieworders page has a join over added_to, product and placesorder tables while the vieworderdetails page has a join over added_to, customer, product, brand, belongs_to, contains and category tables.

ER Diagram


Address
street: string
aptno: string
city: string
state: string
zip: string
Lives_in
Customer
custid: integer
name: string
password: string
phone: string
Places
Order
orderid: integer
pay_method: string
pay_date: date
quantity: integer
Added_to
Category
catid: integer
catname: string
Contains
Product
proid: integer
price: float
pname: string
expiry: date
Comes_from
Supplier
supid: integer
phone: string
address: string
Belongs_to
Brand
brid: integer
brand_name: string
Has_product_
reivews
Product_ reviews
revno: integer
rating: integer
review_text: string

Project Schema

```
CREATE TABLE Product(
proid INTEGER,
price DECIMAL CHECK (price > 0) NOT NULL ,
pname VARCHAR(50) NOT NULL,
expiry DATE CHECK (expiry > TO_DATE('2022-01-01', 'yyyy-mm-dd')),
supid INTEGER NOT NULL,
PRIMARY KEY(proid),
FOREIGN KEY (supid) REFERENCES Supplier(supid)
);


CREATE TABLE Category(
catid INTEGER,
catname VARCHAR(50) NOT NULL,
PRIMARY KEY(catid)
);


CREATE TABLE Contains(
catid INTEGER,
proid INTEGER,
PRIMARY KEY(catid, proid),
FOREIGN KEY(catid) REFERENCES Category(catid),
FOREIGN KEY(proid) REFERENCES Product(proid)
);
-- Product has a total participation constraint in the Contains relation which
we can't model yet
```

```
CREATE TABLE Brand(
brid INTEGER,
brand_name VARCHAR(50) NOT NULL,
PRIMARY KEY(brid)
);


CREATE TABLE Belongs_to(
brid INTEGER NOT NULL,
proid INTEGER,
PRIMARY KEY(proid),
FOREIGN KEY(brid) REFERENCES Brand(brid),
FOREIGN KEY(proid) REFERENCES Product(proid)
);


CREATE TABLE Supplier(
supid INTEGER,
phone CHAR(14) NOT NULL,
address VARCHAR(50) NOT NULL,
PRIMARY KEY(supid)
);


CREATE TABLE ProductReview(
revno INTEGER,
rating DECIMAL CHECK (rating > 0 AND rating < 5),
review_text VARCHAR(500),
proid INTEGER,
PRIMARY KEY(revno, proid),
FOREIGN KEY (proid) REFERENCES Product(proid)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

```
CREATE TABLE Customer(
custid INTEGER,
name VARCHAR(50) NOT NULL,
password VARCHAR(50) NOT NULL CHECK (char_length(password) >=
5),
phone CHAR(14) NOT NULL,
PRIMARY KEY(custid)
);


CREATE TABLE Address(
street VARCHAR(50),
aptno VARCHAR(5),
city VARCHAR(50),
state CHAR(2) CHECK (state IN('AL', 'AK', 'AZ', 'AR', 'CA', 'CO', 'CT', 'DE',
'DC', 'FL', 'GA', 'HI', 'ID', 'IL', 'IN', 'IA', 'KS', 'KY', 'LA', 'ME', 'MD', 'MA', 'MI',
'MN', 'MS', 'MO', 'MT', 'NE', 'NV', 'NH', 'NJ', 'NM', 'NY', 'NC', 'ND', 'OH', 'OK',
'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VT', 'VA', 'WA', 'WV', 'WI', 'WY')),
zip CHAR(5),
custid INTEGER,
PRIMARY KEY(street, aptno, city, state, zip, custid),
FOREIGN KEY(custid) REFERENCES Customer(custid)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);



CREATE TABLE PlacesOrder(
orderid INTEGER,
custid INTEGER,
pay_method VARCHAR(50) NOT NULL,
```

```
pay_date DATE NOT NULL CHECK (pay_date > TO_DATE('2022-01-01',
'yyyy-mm-dd')),
PRIMARY KEY (custid, orderid),
FOREIGN KEY (custid) REFERENCES Customer(custid)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);



CREATE TABLE Added_to(
orderid INTEGER,
proid INTEGER,
quantity INTEGER NOT NULL CHECK (quantity > 0),
custid INTEGER,
PRIMARY KEY(orderid, proid),
FOREIGN KEY (custid, orderid) REFERENCES PlacesOrder(custid,
orderid),
FOREIGN KEY (proid) REFERENCES Product(proid)
);
-- Order has a total participation constraint in the Added_to relation which
we can't model yet
```