Medicine Dealers

<u>CMPS 3420-60 Spring 2021</u>

Skyler Ercoli

Tawfic Jobah

Jose Figueroa

Andrew Mccuan

Table of Contents

Phase 1: Conceptual Design

1.1 - Fact-Finding Techniques and Information Gathering

1.1.1 - Introduction to the Enterprise/Organization.

Medicine Dealers is a web-based pharmacy giving customers access to a large variety of over-the-counter drugs and prescription drugs. We provide a service that allows customers to purchase pharmacy drugs from the comfort of their home and not having the hassle of having to go pick up your prescriptions. Our services allow us to have competitive pricing to other pharmacies and fast delivery times to make sure customers get their medicine. Our website will be able to keep info on customers to make recurring prescriptions/purchases easier. One such thing is that we can keep track of how many refills a customer has left on prescription and be able to ship them more when needed. Also, another feature is that we will be able to store customers' insurance to provide discounts on prescription drugs. Our website's database will also be able to keep track of stock of all products that we will be able to sell to customers.

1.1.2 - Description of Fact-Finding Techniques.

To gain insight into the business of pharmaceuticals, we will be researching countless online examples, as well as gathering information from those who have experience picking up prescriptions. We will also be researching legal documentation on the requirements to run a pharmacy, to ensure that our pharmacy is run to standards. This research should provide us with what information is necessary to run a functioning database containing all the data necessary to provide people with medicine.

1.1.3 - The Miniverse of Interest.

The database will cover the transactions of medicines from Medicine Dealers to our customers. These transactions will be customers ordering prescription or over-the-counter medicine and the database will handle orders and product sales. The database will also keep track of the number of products that will be available to sell. The major entities in our store database will be customers, employees, products, orders, insurance, prescription (weak entity), and supplier. Customers will order products and their insurance can give them a discount on prescription medicine. Employees will pack orders with products and will order from suppliers when quantities are running low.

1.1.4 - Itemized Description of Entity sets and Relationship sets.

Entity Sets:

Customer: c_id, c_fname, c_lname, c_insurance, c_phonenum, c_email, c_password, c_doctor, c_dob, c_address
The customer of the website will be purchasing medicine, that being prescriptions or over-the-counter drugs.

Employee: e_id, e_fname, e_lname, e_salary, e_position, e_ssn, e_address, e_email, e_password
The employee that works for our business, they will be responsible for filling orders and ordering more products from suppliers.

Product: p_id, p_price, p_name, p_supplier, p_quanity,p_PrescriptionNeeded
These are the products that we sell to customers and they are made up of two classes, prescription, and over-the-counter drugs.

Prescription: pre_doctorName, pre_lastFilled, pre_refill
These are any forms of medicine that are prescribed by the doctor.

Orders: o_id, o_product, o_shipDate, (o_amount)
This is for keeping track of items purchased by customers and contains shipping info and order number.

Insurance: i_name, i_discount
The insurance provider a customer might have and can give different discounts for prescriptions.

Supplier: s_id, s_name
The supplier is the outside organization that will provide the product to us.

Relationship Sets:

Purchases: Between Customer and Order; Many to Many
Customer purchases an order.

Contains: Between Order and Product; Many to Many
The product is in stock.

Ships: Between Order and Employee; Many to Many
The employee ships the order.

Restock: Between Employee and Product; Many to Many
The employee restocks the product if needed.

Covers: Between Insurance and Customer; Many to 1
The insurance will cover the customer's prescription.

Discounts: Between Insurance and Prescription; Many to Many
The insurance will pride a discount on the customer's prescription.
- Attribute: d_discount

Requires: Between Customer and Prescription; Many to 1
Customers obtain the product at full price.

Orders From: Between Employee and Supplier; Many to Many
Employee orders the medicine from the supplier.

Provides: Between Supplier and Product; Many to Many
The supplier provides the product.

1.1.5 - User Groups, Data Views, and Operations.

In the database, there are two user groups: customers and employees. The first user group is the customer, this user will be able to create an account, log in to an account, and edit their account info (ex. Insurance provider, email, name). Also, they will be able to view products available to purchase, add products to a cart and purchase them, and fulfill prescriptions. The other user group employee will be able to login into their employee account, can see orders so they can fulfill them, and order more stock on products that are getting low.

1.2 - Conceptual Database Design.

1.2.1 - Entity Type Descriptions

Customer

Customer - This entity holds information about a customer for the customer account and this will be used to purchase products from the website.
Candidate Keys: c_id, c_phonenum, c_email
Primary Key: c_id
Strong/Weak: Strong

Attribute name	c_id	c_fname	c_lname	c_insurance	c_phonenum	c_email	c_password	c_doctor	c_dob	c_address
description	Unique ID	Customer name	Customer last name	Customers insurance	Customer phone num	Customer email	Customers password	Customers doctor	Customer date of birth	Customer home address

					ber					
domain/type	int	char	char	char	int	char	char	char	int	car
value-range	0 - max customer	a-z	a-z	a-z	000000000 - 999999999	A-z & 0-9 & special char	A-z & 0-9 & special char	a-z	1/1/1900 - 1/1/2021	A-z & 000000 - 999999
Default value	+1	null	null	null	null	null	null	null	null	null
null?	no	no	no	yes	yes	yes	no	yes	no	no
unique?	yes	no	no	no	yes	yes	no	no	no	no
Single or multiple-value	single	single	single	single	single	single	single	single	single	single
Simple or composite	simple	simple	simple	simple	simple	simple	simple	simple	simple	composite

Employee

Employee - The employee entity contains the employee's information such as position, salary, social security number, etc.
Candidate Keys: e_id, e_snn, e_email
Primary Key: e_id
Strong/Weak: Strong

Attribute name	e_id	e_fname	e_lname	e_salary	e_position	e_ssn	e_address	e_email	e_password
description	Employee id	Employees first name	Employees last name	Employee salary	Employee position	Employee social security	Employee address	Employee email	Employee password

						number			
domain/type	int	char	char	int	char	int	char	char	int
value-range	0 - max employee	a-z	a-z	000000 - 999999	a-z	000000000 - 999999999	a-z * 000000 - 999999	A-z & 0-9 & special char	A-z & 0-9 & special char
Default value	+1	null	null	null	null	null	null	null	null
null?	no	yes	yes	yes	yes	yes	yes	yes	yes
unique?	yes	no	no	no	no	yes	no	yes	no
Single or multiple-value	single	single	single	single	multiple	single	single	single	single
Simple or composite	simple	simple	simple	simple	simple	simple	composite	simple	simple

Product

Product - This entity holds information about products that are sold to customers. The information that is stored is an id, price. Also, this entity is the primary class to the Prescription entity.
Candidate Keys: p_id
Primary Key: p_id
Strong/Weak: Strong

Attribute name	p_id	p_price	p_name	p_supplier	p_quanitity	p_PrescriptionNeeded
description	Product unique ID	Product sell price	Product name	Supplier of product	Quantity of product	If prescription is needed
domain/type	int	int	char	char	int	bool

value-range	0 to max amount of products	000000-999999	a-z	a-z	000-999	0-1
Default value	+1	0	null	null	0	0
null?	no	yes	yes	yes	yes	yes
unique?	yes	no	no	no	no	no
Single or multiple-value	single	single	single	single	single	single
Simple or composite	simple	simple	simple	simple	simple	simple

Prescription

Prescription - The prescription entity holds the information of the doctor prescribing the medication and the customers' filled and refilled info. This entity is a subclass to the primary class product.
Foreign Key: p_id
Strong/Weak: Weak

Attribute name	pre_doctorName	pre_lastFilled	pre_refill
description	Name of doctor who gave prescription	Last day prescription was filled	How many refills there are
domain/type	char	date/time	int
value-range	a-z	1/1/1990 - 1/1/2022	00-99

Default value	null	0	0
null?	yes	yes	yes
unique?	no	no	no
Single or multiple-value	single	single	single
Simple or composite	simple	simple	simple

Orders

Order - The entity will keep track of the customers' order id when purchasing medication. Order entity will also contain the product(s), product cost(s), and shipping information.
Candidate Keys: o_id
Primary Key: o_id
Strong/Weak: Weak

Attribute name	o_id	o_shipDate	o_amount
description	Unique id of order	Date of the order	Amount of products being purchased
domain/type	int	char	int

value-range	0 to max order	a-z	00-99
Default value	+1	null	0
null?	no	yes	yes
unique?	yes	no	no
Single or multiple-value	single	multiple-value	multiple-value
Simple or composite	simple	simple	simple

Insurance

Insurance - The insurance entity will contain the name of the insurance and the discount rate that the customer will receive.
Candidate Keys: i_name
Primary Key: i_name
Strong/Weak: Strong

Attribute name	i_name	i_discount
description	Name of insurance company	Discount rate from insurance
domain/type	char	var

value-range	a-z	00-99
Default value	null	null
null?	yes	yes
unique?	yes	no
Single or multiple-value	single	single
Simple or composite	simple	simple

Supplier

Supplier - The supplier entity will hold information about the supplier, this information is a supplier id and the supplier name.
Candidate Keys: s_id
Primary Key: s_id
Strong/Weak: Strong

Attribute name	s_id	s_name
description	Supplier unique id	Supplier name
domain/type	int	var
value-range	0- max supplier	a-z
Default value	+1	null
null?	no	yes
unique?	yes	no

Single or multiple-valu e	single	single
Simple or composite	simple	simple

1.2.2 - Relationship Type Description

Relationship: Purchases

Description: A customer purchases items in the order.
Entities Involved: Customer / Order
Cardinality: Many to Many both ways
Participation constraint:
Customer: Partial/Optional
Order: Total/Mandatory
Attributes: None

Relationship: Contains

Description: An order contains certain products and amounts.
Entities Involved: Order / Product
Cardinality: Many to Many both ways
Participation constraint:
Order: Total/Mandatory
Product: Partial/Optional
Attributes: None

Relationship: Ships

Description: An employee packages and ships an order.
Entities Involved: Order / Employee
Cardinality: Many to Many
Participation constraint:
Order: Total/Mandatory
Employee: Partial/Optional
Attributes: None

Relationship: Restocks

Description: An employee will make sure there is enough product in stock and restock if needed.

Entities Involved: Employee / Product
Cardinality: Many to Many
Participation constraint: Total/Mandatory for both sides
Attributes: None

Relationship: Covers
Description: The insurance covers a customer.
Entities Involved: Customer / Insurance
Cardinality: Many to 1
Participation constraint: Total/Mandatory for both sides
Attributes: None

Relationship: Discounts
Description: Insurance provides discounts on prescriptions.
Entities Involved: Insurance / Prescription
Cardinality: Many to Many
Participation constraint:
Insurance: Partial/Optional
Prescription: Total/Mandatory
Attributes: d_discount

Relationship: Requires
Description: The customer requires a prescription to be healthy.
Entities Involved: Customer / Prescription
Cardinality: Many to 1
Participation constraint:
Customer: Partial/Optional
Prescription: Total/Mandatory
Attributes: None

Relationship: Orders From
Description: An employee orders more products from a supplier.
Entities Involved: Employee / Supplier
Cardinality: Many to Many
Participation constraint: Total/Mandatory for both sides
Attributes: None

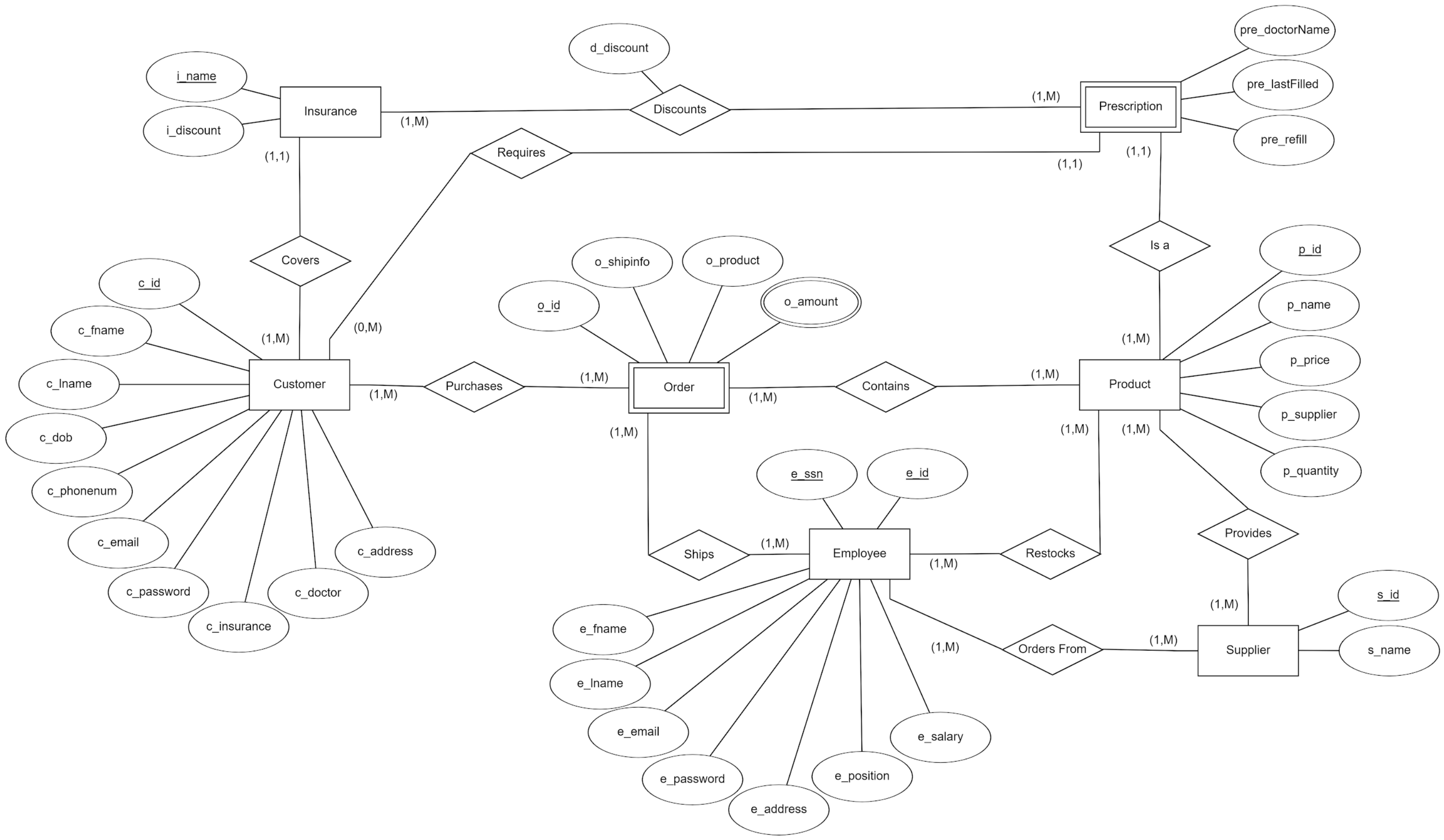
Relationship: Provides
Description: A supplier provides a product to the organization.
Entities Involved: Product / Supplier
Cardinality: Many to Many

Participation constraint: Total/Mandatory for both sides
Attributes: None

1.2.3 - Related Entity Types

In our database, we have one section that uses a feature from the Enhanced ER Model, and this feature is the sub-class. We have the entity Product which is the primary class and Prescription which is the sub-class.

1.2.4 - ER Diagram



Phase 2: Conversion from Conceptual to Relational Database

2.1 - The ER and Relational models:

This section will go into detail about the differences and similarities of an entity-relationship model and relation model. It will go in-depth on how to convert an ER(entity-relationship) model to a relation model. We will be going into the history and which model is more compatible for what situation.

2.1.1 - Descriptions of ER and Relational Models

ER:

An ER model shows the relationship of entities in a database. An ER model is used to show relationships between entities in an easy-to-read way for normal people. It is usually drawn in boxes, ovals, and diamonds that are connected by lines. The boxes are to symbolize entities, diamonds symbolize relationships between entities, and ovals symbolize attributes of the entities. ER modeling was developed in 1976 by Peter Chen for databases and design. In Chen's paper, he explains ER modeling by saying that, "*The entity-relationship model adopts the more natural view that the real world consists of entities and relationships. It incorporates some of the important semantic information about the real world.*" The best part of the ER diagram is how easy it is to distinguish between entities, attributes, and relationships because of the different shapes used.

Relational model:

The relation model was developed in 1969 by scientist Edgar Codd. This type of model was created to show all models of a database asset of tuples that can refer to relations. This type of model is made to show database programmers exactly what attributes will be associated with each element.

2.1.2 - Model Comparisons

Both the Er and Relational models are useful because you can use them to brainstorm ideas and fix mistakes before coding and wasting time. An ER model is nice to use because everyone can understand the correlation between entities, attributes, and relationships. The downside of the ease of reading an ER model though is its simplicity, not all the information can be explained on a visual diagram and that's when a relation model comes in handy. Relation models show the database as a set of tables. It can go more in-depth and explain every connection and relationship to all the data and attributes but because it is in a table and not visual it becomes harder to read.

2.2 - Conceptual to Logical Conversion Process

This section will be used to explain the conversion of an ER model to a relation model. We will explain how to convert entity types, relationships, and attributes into relations.

2.2.1 - Converting Entity Types to Relations

Strong Entities:

To convert a strong entity, you will need to figure out which attribute can become your primary key. If the entity contains any composite or multivalued attributes then see below.

Weak Entities:

Converting a weak entity is slightly different. Since a weak entity does not have its primary key, you need to give it a foreign key from the associated strong entity's primary keys.

Composite Attributes:

Composite attributes would be separated into individual attributes.

Multivalued Attributes:

Multivalued attributes will need a new relation, with a foreign key connecting the newly created related to the main relation.

2.2.2 - Converting Relationship Types to Relations

There are three ways to convert binary relationships to relations.

Foreign key approach:

When two entities are linked by a relationship, you choose the entity that has full participation will have the primary key and the other entity will take the foreign key. Now both entities can show that there is a relation between both of them and which entity is more important.

Merged Relation:

When two entities both have full participation then you would need to merge both entities. Both entities will be merged and turn into one relation. Both will have the same number of tuples. This method can only be used with relationships and entities that have one-to-one total participation.

Cross-Reference Method:

This method takes two entities and makes a new relation with only the primary keys from both entities. The upside of this method is that everything is covered when doing the conversions.

When converting many to many relationships then you can only use the cross-reference method. Converting one too many then you can use all methods except for merged relation methods. MultiValue attributes will need a new relation for every attribute with a foreign key connecting them all. Converting n-ary relationships where a group of entities is linked by one relationship, you start by creating a new relationship that has the primary keys from every entity and a new relation is used to reference the relationships.

2.2.3 - Converting Extended Types to Relations

When converting extended types into the relational model, there are four ways to do it.

Option 1:

The first way is to create a superclass with many subclasses. The superclass will be the main relation and its primary key will be inherited by the subclass relations. The subclasses will have the superclasses' primary key as well as their attributes.

Option 2:

The second option is to create multiple relations, only from subclasses. The primary keys will be shared and no superclass will be involved.

Option 3:

The third option is to create one relation with multiple attribute types. The relation would have all the attributes from both the superclass and the sub-class, but the type attribute is needed to prevent confusion.

Option 4:

The fourth option is to create a relation with multiple types. Similar to option three but can hold multiple type attributes.

Union Types:

Each relation in the union will need to share a key attribute. The key attribute will connect all of the union types to the main entity. The key attribute can be taken from the ER model.

2.2.4 - Database Constraints

Constraints are a set of rules that are used to keep the integrity of the data in a database. Constraints are put on columns or tables to limit the type of data that gets entered.

Entity Constraints:

The entity constraint is a constraint where an entity in the database cant be null.

Primary key and unique key constraints:

A primary key constraint is used to distinguish a tuple in a relation. Usually limits the chances of duplicate data in a database. A candidate key is either one or more attributes than is unique for each tuple.

Referential Constraints:

A referential constraint is used between two tables and it is also known as a foreign key constraint. In this constraint if an attribute in one relation references a value in a different relation then that value has to exist and either be null or available in the other table.

Check constraints and business rules:

Check constraints are used to make sure that inaccurate data isn't used in the database, it does this by specifying a certain range of acceptable values. For example, someone's social security number should not have letters or their data of birth cant be the next day.

2.3 - Results of ER to Relational Conversion

With our database conversion from ER to the relational model it went pretty smoothly for most of the entities. The weak entities conversion was another straightforward conversion. The relationship discounts were somewhat difficult to convert but after some discussion we decided it has two primary keys, one from the insurance entity and the other from the product entity. Then finally the special case entity was prescription is a subclass of product so it inherits the foreign key from the product.

2.3.1 Relation Schema

Customer(c_id, c_fname, c_lname, c_phonenum, c_email, c_password, c_doctor, c_dob, c_address **)**

Attributes:	Domain:	Constraints:	Candidate Keys:
c_id	int	Not null	yes(primary)
c_fname	char	Not null	no
c_lname	char	Not null	no
c_phoneNum	int	null	no
c_email	char	null	no
c_password	char	null	no
c_doctor	char	null	no
c_dob	int	null	no
c_address	char	null	no

Employee(e_id, e_fname, e_lname, e_salary, e_position, e_ssn, e_address, e_email, e_password **)**

Attributes:	Domain:	Constraints:	Candidate Keys:
e_id	int	Not null	yes(primary)
e_fname	char	Not null	no
e_lname	char	Not null	no
e_salary	int	null	no
e_position	char	null	no
e_ssn	int	null	no
e_address	char	null	no
e_email	char	null	no
e_password	int	null	no

Product(p_id, p_price, p_name, p_supplier, p_quantity, p_PrescriptionNeeded **)**

Attributes:	Domain:	Constraints:	Candidate Keys:
p_id	int	Not null	yes(primary)
p_price	int	null	no
p_name	char	null	no
p_supplier	char	null	no
p_quantity	int	null	no
p_PrescriptionNeeded	bool	null	no

Insurance(i_id, i_name, i_discount **)**

Attributes:	Domain:	Constraints:	Candidate Keys:
i_id	int	Not null	yes(primary)
i_name	char	Not null	no
i_discount	int	null	no

Supplier(s_id, s_name **)**

Attributes:	Domain:	Constraints:	Candidate Keys:
s_id	int	Not null	yes(primary)
s_name	char	null	no

Orders(c_id, o_id, o_shipDate**)**

Attributes:	Domain:	Constraints:	Candidate Keys:
o_id	int	Not null	yes
c_id	int	Foreign key	no
o_shipDate	date-time	Not null	no

Discounts(i_id, p_id, d_dicsount **)**

Attributes:	Domain:	Constraints:	Candidate Keys:
i_id	int	Foreign key	yes
p_id	int	Foreign key	yes
d_discount	int	null	no

Prescription(p_id, c_id, pre_doctorName, pre_lastFilled, pre_refill **)**

Attributes:	Domain:	Constraints:	Candidate Keys:
p_id	int	Foreign key	yes
c_id	int	Foreign key	yes
pre_doctorName	char	null	no
pre_lastFilled	date-time	null	no
pre_refill	int	null	no

Contains(o_id, p_id, p_quantity **)**

Attributes:	Domain:	Constraints:	Candidate Keys:
o_id	int	Foreign key	yes
p_id	int	Foreign key	yes
p_quantity	int	null	no

Restocks(e_id, p_id, restock_date **)**

Attributes:	Domain:	Constraints:	Candidate Keys:
e_id	int	Foreign key	yes
p_id	int	Foreign key	yes
restock_date	date-time	null	no

OrdersFrom(e_id, s_id, order_date **)**

Attributes:	Domain:	Constraints:	Candidate Keys:
e_id	int	Foreign key	yes
s_id	int	Foreign key	yes
order_date	date-time	null	no

Covers(c_id, i_id **)**

Attributes:	Domain:	Constraints:	Candidate Keys:
c_id	int	Foreign key	yes

i_id	int	Foreign key	yes

Ships(o_id, e_id, ship_date **)**

Attributes:	Domain:	Constraints:	Candidate Keys:
e_id	int	Foreign key	yes
p_id	int	Foreign key	yes
ship_date	date-time	null	no

Provides(p_id, s_id, _date **)**

Attributes:	Domain:	Constraints:	Candidate Keys:
s_id	int	Foreign key	yes
p_id	int	Foreign key	yes
provide_date	date-time	null	no

2.3.2 Sample Data

Customer:

c_id	c_fname	c_lname	c_phonenum	c_email	c_password	c_doctor	c_dob	c_address
1	Yehudit	Joscelyne	197-843-8111	yjoscelyne0@macromedia.com	DwC2uWz	Phillip McGraw	3/6/1976	9772 Columbus Pass, Bakersfield, CA, 93314
2	Bradly	Cheal	455-593-9976	bcheal1@prnewswire.com	SxCEx9rffpoq	Libby Caldwell	6/18/1989	3 Clarendon Way, Bakersfield, CA, 93312
3	Claresta	Garrod	441-246-3580	cgarrod2@seattletimes.com	f0LGTb3lQiH0	Horatio Gauche	4/10/1945	67 Dottie Junction, Bakersfield, CA, 93307
4	Chanda	Hackney	541-389-4971	chackney3@unc.edu	axqUcIES	Derek Shepherd	7/29/1949	0 Mcbride Trail, Bakersfield, CA, 93311
5	Wendeline	Bawme	854-488-2858	wbawme4@myspace.com	unCGTf1CZz	Abby Normal	2/29/1960	11658 Mifflin Hill, Bakersfield, CA, 93311

6	Irvine	Kochel	688-123-3253	ikochel5@weebly.com	U72R9penD	Phillip McGraw	7/3/1986	9932 Columbus Court, Bakersfield, CA, 93311
7	Robbin	Gon	133-552-3384	rgon6@wisc.edu	ZZ1iWEkk14	John Dorian	3/2/1973	13106 Anniversary Parkway, Bakersfield, CA, 93314
8	Nelie	Gerritsma	736-884-3120	ngerritsma7@princeton.edu	uR3CY9Ed	Libby Caldwell	7/29/1984	94 Bashford Center, Bakersfield, CA, 93311
9	Paulo	Haisell	305-941-3559	phaisell8@shareasale.com	No3QQsPHjq	Abby Normal	12/5/1969	6 Mallory Place, Bakersfield, CA, 93307
10	Ulrick	O'Shirine	818-757-1733	uoshirine9@cisco.com	c49jqp	Derek Shepherd	9/17/1956	799 Doe Crossing Court, Bakersfield, CA, 93311
11	Pegeen	Fone	742-717-1788	pfonea@live.com	jhiYmbTPR5pi	Horatio Gauche	9/19/1989	50 Buell Pass, Bakersfield, CA, 93311
12	Reeva	Aylmer	527-388-8202	raylmerb@usa.gov	kXKx7T2O	Elloit Reed	3/20/1950	14485 Lotheville Avenue
13	April	Mouth	600-255-6495	amouthc@vimeo.com	Sb5lgeRb	Elloit Reed	11/20/1990	7 Debra Trail, Bakersfield, CA, 93314
14	Jess	Devereu	293-312-9936	jdevereud@bandcamp.com	IE2x1FG53	Libby Caldwell	7/20/1970	88522 Lukken Plaza, Bakersfield,CA, 93307

15	Jorgan	Lyddiatt	399-250-4108	jlyddiatte@bizjournals.com	s2yOR400LCz	John Dorian	8/21/1979	3837 Ohio Poin, Bakersfield, CA, 93307
16	Alene	Dieton	952-892-5982	adietonf@wisc.edu	tZGMvjENn	Phillip McGraw	11/17/1999	238 West Parkway, Bakersfield, CA, 93311
17	Ferne	Drache	351-606-7433	fdracheg@geocities.jp	vTsDrJuz	Derek Shepherd	1/26/1977	03 Eagle Crest Way, Bakersfield, CA, 93311
18	Perle	Gyford	864-367-1364	pgyfordh@intel.com	sKnt6Q8	Abby Normal	1/5/1966	3225 Porter Circle, Bakersfield, CA, 93311
19	Gae	Jiroutka	449-335-0484	gjiroutkai@free.fr	VmVvwABn4	Elloit Reed	11/9/1957	311 Union Junction, Bakersfield,CA,93307, San Luis Obispo, CA 93407
20	Gaylor	St. Clair	429-433-7562	gstclairj@t.co	VrJKuVGDX	Sandra Lee	9/1/1948	4429 Atwood Plaza, Bakersfield, CA, 93311
21	Hill	Luke	919-724-1995	hlukek@jalbum.net	vOVJfRlGY	Abby Normal	12/11/1956	68 Judy Road, San Luis Obispo, CA 93407
22	Verile	Realy	804-917-5825	vrealyl@moonfruit.com	aDN8pr	Elloit Reed	7/18/1969	05 Merchant Parkway, Bakersfield, CA, 93312
23	Doyle	Keddey	508-476-9095	dkeddeym@printfriendly.com	Zr3AFOmN	John Dorian	5/19/1983	23 Michigan Plaza, Bakersfield, CA, 93311

24	Jackson	Petranek	472-687-4577	jpetranekn@acquirethisname.com	riUN5UJzdWd	Abby Normal	7/17/1998	58 Oxford Way, Bakersfield, CA, 93314
25	Babette	Pund	316-320-8515	bpundo@google.com	fydYd8Zn	Derek Shepherd	5/18/1980	1884 Village Green Hill, Bakersfield, CA, 93307
26	Ingram	Jakobssen	802-508-6044	ijakobssenp@wix.com	HcZh3gI3	Sandra Lee	10/20/1983	5947 Logan Terrace, San Luis Obispo, CA 93407
27	Ruthie	Slainey	213-380-4842	rslaineyq@newyorker.com	eio1my	John Cook	9/22/1944	6 High Crossing Drive, Bakersfield, CA, 93312
28	Ulrikaumeko	Augustus	431-234-5329	uaugustusr@issuu.com	8FbQrjF1Bi	Abby Normal	10/10/1971	16 Porter Way, Bakersfield, CA, 93311
29	Jessika	Olensby	578-591-0252	jolensbys@bandcamp.com	ziIHS9B0jBlY	Elloit Reed	8/21/1984	67 Roth Lane, Bakersfield, CA, 93307
30	Shina	Antonchik	237-169-5590	santonchikt@godaddy.com	iaev9c	John Dorian	11/25/1965	6404 Maple Wood Road, Bakersfield, CA, 93314
31	Ogdon	Blackborough	448-263-5129	oblackboroughu@ft.com	sChwZVMDdz	Phillip McGraw	8/9/1975	25837 Burrows Point, San Luis Obispo, CA 93407

32	Louisette	Tuckwell	478-194-3384	ltuckwellv@washingtonpost.com	kdoC1msoX3K	Derek Shepherd	4/2/1956	9 Anzinger Alley, Bakersfield, CA, 93311
33	Lizette	Ipplett	975-857-2932	lipplettw@google.com.br	4W8hq8	John Cook	2/19/1986	3835 Lindbergh Park, Bakersfield, CA, 93314
34	Cullie	Carmen	461-910-9239	ccarmenx@howstuffworks.com	XXibsvq7MSS	John Sins	3/8/1941	63643 Maple Wood Junction, Bakersfield, CA, 93312
35	Corella	Willock	569-859-3786	cwillocky@cpanel.net	Qgpe5n	Jan Pol	2/7/1950	8016 Little Fleur Pass, Bakersfield, CA, 93311
36	Gladys	Kindon	521-925-9384	gkindonz@gnu.org	6c2JM6S	Jan Pol	12/29/1973	5 Anthes Junction, San Luis Obispo, CA 93407
37	Cherilynn	Hoston	962-931-8077	choston10@naver.com	5Ya9VDHT62R9	John Cook	10/23/1950	67 Red Cloud Crossing, Bakersfield, CA, 93307
38	Carl	Nel	611-611-9879	cnel11@imgur.com	qKA8nH	John Dorian	4/15/1954	44395 Eggendart Avenue, San Luis Obispo, CA 93407
39	Durant	Balloch	405-849-6602	dballoch12@nasa.gov	iYL7hCAg9tDL	John Cook	2/28/1995	42222 Dottie Circle, Bakersfield, CA, 93312

40	Kylie	Breadmore	235-204-0062	kbreadmore13@thetimes.co.uk	pg5KWTtsq	Derek Shepherd	10/26/1992	7425 Mccormick Parkway, Bakersfield, CA, 93311

Employee:

e_id	e_fname	e_lname	e_salary	e_position	e_snn	e_address	e_email	e_password
1	Ruggiero	Blint	50000	Packer	494-27-6559	00 Laurel Avenue,Bakersfield, CA, 93312	rblint0@house.gov	2mb8po0k
2	Thayne	Ollett	80000	Pharmacist Technician	561-25-4086	04153 Blackbird Plaza, Bakersfield, CA, 93311	tollett1@cdc.gov	uxbxANwe1
3	Bald	Connochie	60000	Packer	892-26-7957	1090 Sycamore Plaza,Bakersfield, CA, 93312	bconnochie2@opensource.org	dq204X

4	Cecil	Flancinbaum	50000	Packer	831-53-5856	407 Fuller Road,Bakersfield, CA, 93311	cflancinbaum3@nature.com	2GohZFxGf
5	Livvyy	Roget	60000	Supervisor	301-25-0091	80 Loeprich Crossing, Bakersfield, CA, 93311	lroget4@unblog.fr	vnZsyj7h
6	Hope	Skullet	75000	Manager	598-49-4296	4652 Lakewood Gardens Way, Bakersfield, CA, 93312	hskullet5@nyu.edu	QxzxKYYSof
7	Ambrose	Bemand	80000	Pharmacist Technician	462-31-1186	60231 Birchwood Hill, Bakersfield, CA, 93307	abemand6@guardian.co.uk	zfVVj8g2F
8	Amye	Andriveau	50000	Packer	410-86-5794	1066 East Lane, Bakersfield, CA, 93311	aandriveau7@example.com	Eamv86
9	Dominik	Bellenger	65000	Packer	618-57-8993	34092 Morrow Alley, Bakersfield,	dbellenger8@storify.com	iZxifoxu

						CA, 93311		
10	Ulberto	Durrand	55000	Packer	355-99-7363	26731 Basil Avenue, Bakersfield, CA, 93311	udurrand9@diigo.com	zHqz7V

Product:

p_id	p_price	p_name	p_supplier	p_quantity	p_PrescriptionNeeded
1	108	Risperidone	Pfizer Consumer Healthcare	191	false
2	6	Gelato Homecare	Cardinal Health	165	false
3	147	Alpan 40 (Number 110)	BioActive Nutritional Inc	159	false
4	60	Quinapril	Top Care	17	false
5	132	Propranolol Hydrochloride	Medline Industries	163	false

6	57	Dental Plak	Cardinal Health	226	false
7	31	GABAPENTIN	Antigen Laboratories Inc	104	false
8	32	EDEMA HP	Infirst Healthcare	131	false
9	133	California Mugwort	Pfizer Consumer Healthcare	201	true
10	12	Dilaudid	Uriel Pharmacy Inc	244	false
11	35	Gabitidine	Cardinal Health	73	false
12	131	BLADE HOWL	Infirst Healthcare	34	false
13	40	Deb Gold	Cardinal Health	149	false
14	30	Tramadol Hydrochloride	Cardinal Health	167	true
15	99	Anti-Bacterial Hand Sanitizer	Top Care	198	true
16	28	CHOLINE MAGNESIUM TRISALICYLATE	Infirst Healthcare	160	false

17	96	TEMODAR	Proficient Rx LP	89	false
18	134	Bumetanide	Infirst Healthcare	241	false
19	91	Ibuprofen	Cardinal Health	193	false
20	24	Molluscum Control	Infirst Healthcare	43	false
21	119	Diclofenac Sodium	Cardinal Health	14	false
22	118	Atorvastatin Calcium	Top Care	191	false
23	143	Solbar Shield SPF40	Clinical Solutions Wholesale	193	true
24	91	Amitriptyline Hydrochloride	Cardinal Health	117	false
25	67	Femara	Pfizer Consumer Healthcare	127	false
26	95	Clara	Proficient Rx LP	136	false
27	8	Pollens - Trees, Tree Mix 5	BioActive Nutritional Inc	162	false

28	88	Hydrochlorothiazide	Medline Industries	224	true
29	130	Citalopram	Top Care	197	false
30	144	Fast Freeze	Pfizer Consumer Healthcare	168	true
31	16	Neutrogena Oil-Free Acne Wash	Clinical Solutions Wholesale	118	false
32	65	Vimpat	Clinical Solutions Wholesale	137	false
33	84	Amlodipine Besylate	Infirst Healthcare	168	false
34	109	SPRYCEL	Medline Industries	41	true
35	89	Scott-Vincent Borba	Cardinal Health	47	true
36	41	Trazodone Hydrochloride	Antigen Laboratories Inc	208	false
37	70	Sulwhasoo	BioActive Nutritional Inc	27	false

38	43	citalopram hydrobromide	Medline Industries	82	false
39	77	Dicyclomine	Cardinal Health	15	false
40	93	Metoclopramide	Cardinal Health	114	true

Insurance:

i_id	**i_name**	**i_discount**
1	Aetna	5
2	Blue Cross	7

3	Cigna	6
4	Harvard Pilgrim	5
5	Humana	8
6	Kaiser Permanente	10

Supplier:

s_id	s_name
1	Pfizer Consumer Healthcare

2	Cardinal Health
3	Medline Industries
4	Top Care
5	Infirst Healthcare
6	Antigen Laboratories Inc
7	Proficient Rx LP
8	Uriel Pharmacy Inc
9	BioActive Nutritional Inc
10	Clinical Solutions Wholesale

Orders:

o_id	c_id	o_shipDate

1	30	4/23/2019 20:50
2	39	1/22/2021 15:27
3	40	1/22/2019 13:57
4	38	1/28/2019 11:11
5	18	4/14/2021 12:08
6	25	10/3/2019 2:50
7	2	2/21/2021 18:19
8	20	11/4/2020 9:31
9	31	1/25/2020 0:04
10	5	7/5/2020 14:57
11	17	12/24/2020 15:06
12	14	4/10/2021 21:40
13	30	1/8/2019 6:44

14	13	2/25/2019 10:13
15	19	10/5/2020 7:54
16	5	4/9/2021 23:02
17	19	2/14/2019 19:46
18	38	11/6/2019 2:22
19	29	5/30/2019 12:11
20	4	10/19/2019 22:26
21	7	8/26/2020 8:36
22	17	9/10/2019 18:59
23	17	1/30/2020 8:44
24	28	11/27/2020 20:02
25	2	5/28/2020 7:10

26	19	1/11/2021 2:54
27	3	3/18/2019 13:14
28	10	9/5/2019 17:16
29	3	1/28/2019 1:44
30	35	1/15/2021 14:43
31	22	6/25/2020 1:55
32	21	6/27/2020 19:09
33	22	1/11/2021 21:38
34	27	10/7/2019 3:25
35	23	9/4/2019 7:16
36	34	10/18/2019 7:08
37	31	7/26/2019 2:57
38	20	8/31/2019 1:09

39	37	12/8/2019 9:29
40	24	7/31/2019 14:26

Discounts:

i_id	p_id	d_discount
3	34	15
1	30	16
3	4	12
2	26	17
6	35	9
1	24	12
6	23	8
3	5	10
5	7	11

4	32	7
4	40	15
5	5	18
3	25	8
4	21	7
1	4	3
2	21	7
4	20	12
5	17	2
6	33	11
4	37	2
6	15	16
6	2	17
5	8	6

4	12	18
1	39	17
4	16	12
5	5	3
2	13	20
4	32	5
1	25	2
6	6	14
5	26	6
3	34	19
5	6	17
2	13	11
2	5	20
1	28	20

1	22	8
2	31	12
5	4	18

Prescription:

p_id	c_id	pre_doctorName	pre_lastFilled	pre_refill
26	22	Elloit Reed	7/22/2020 3:17	5
23	37	John Cook	6/12/2020 0:51	2
1	39	John Cook	10/22/2020 22:36	2
11	8	Libby Caldwell	11/4/2020 12:47	4
11	1	Phillip McGraw	12/2/2020 4:36	3
5	32	Derek Shepherd	2/1/2021 21:26	4
35	4	Derek Shepherd	3/2/2021 4:31	3
40	31	Phillip McGraw	12/1/2020 0:08	4

23	5	Abby Normal	8/21/2020 10:35	5
30	35	Jan Pol	3/6/2021 19:42	1
13	2	Libby Caldwell	8/24/2020 11:32	3
33	29	Elloit Reed	9/25/2020 2:28	3
40	3	Horatio Gauche	2/11/2020 13:38	4
40	2	Libby Caldwell	6/30/2020 2:31	1
13	11	Horatio Gauche	2/5/2021 10:19	1
23	35	Jan Pol	2/8/2021 1:03	4
23	9	Abby Normal	1/26/2021 8:54	4
31	25	Derek Shepherd	9/16/2020 9:40	2
25	5	Abby Normal	3/25/2021 23:16	1
29	24	Abby Normal	6/1/2020 17:14	2
9	32	Derek Shepherd	5/3/2020 12:35	1

28	32	Derek Shepherd	3/9/2020 9:54	2
11	39	John Cook	11/17/2020 3:30	3
8	8	Libby Caldwell	12/21/2020 14:04	2
24	12	Elloit Reed	8/9/2020 4:29	4
14	9	Abby Normal	6/15/2020 4:30	1
31	9	Abby Normal	11/12/2020 15:20	5
4	23	John Dorian	1/27/2020 18:30	4
38	16	Phillip McGraw	1/13/2020 4:23	1
35	3	Horatio Gauche	2/2/2020 10:46	5
17	3	Horatio Gauche	4/8/2020 8:28	1
5	22	Elloit Reed	5/10/2020 21:47	1
6	25	Derek Shepherd	5/18/2020 19:25	2
36	29	Elloit Reed	1/1/2021 1:04	1

29	6	Phillip McGraw	5/17/2020 6:31	3
13	32	Derek Shepherd	3/18/2020 10:36	1
9	13	Elloit Reed	1/17/2020 11:11	4
28	3	Horatio Gauche	7/8/2020 7:29	4
25	23	John Dorian	9/7/2020 1:21	5
5	17	Derek Shepherd	11/17/2020 10:48	1

Contains:

o_id	p_id	p_quantity
75	3	2
70	18	4
72	25	1
12	23	4

44	40	2
94	20	3
31	15	4
60	33	1
30	12	1
15	15	4
4	17	5
55	8	2
35	20	1
9	3	3
17	27	2
24	31	4
56	29	2

81	6	2
68	7	1
36	36	1
58	25	3
89	4	5
2	30	1
36	40	5
51	22	2
8	39	1
65	2	3
52	13	5
81	40	1
47	19	5

65	26	3
65	39	5
100	3	5
26	13	1
48	18	2
14	33	5
14	11	1
75	20	2
43	20	2
91	21	2
44	10	3
24	30	3
87	13	5

47	32	2
6	38	1
97	1	2
55	19	1
98	1	1
64	9	1
52	35	3
47	10	5
36	26	2
9	1	3
5	20	3
3	26	1
15	27	3

77	6	2
92	35	3
13	7	4
95	23	1
80	21	2
15	11	5
5	17	5
69	12	1
43	6	2
70	33	4
85	23	1
55	17	2
68	1	4

34	31	4
77	11	3
34	7	2
18	27	1
71	28	2
77	7	5
97	27	5
49	31	3
66	23	1
92	34	4
56	17	3
29	34	2
1	4	4

78	16	4
62	11	4
93	34	1
70	34	3
70	21	1
10	30	5
3	19	5
15	22	2
76	10	5
55	27	2
25	28	4
50	38	1
78	22	1

9	2	2
97	22	2
35	20	2
16	37	5
1	40	5

Restocks:

e_id	p_id	restock_date
6	26	8/9/2020 7:57
3	22	7/17/2020 10:47
8	7	1/14/2021 7:18
8	40	9/6/2020 23:20
3	35	10/6/2020 0:20

4	31	9/30/2020 20:25
4	38	3/5/2021 20:49
6	30	8/21/2020 17:10
2	20	9/13/2020 12:51
2	17	4/4/2021 8:11
9	5	12/8/2020 23:01
7	30	10/21/2020 2:46
9	26	1/27/2021 13:23
9	33	4/10/2021 20:02
5	3	2/25/2021 22:38
9	24	4/9/2021 23:51
7	40	12/26/2020 10:57
4	33	11/10/2020 5:32

7	36	7/7/2020 21:42
1	9	9/18/2020 12:21
10	35	12/3/2020 11:54
8	7	2/1/2021 5:05
4	6	6/3/2020 23:17
10	31	5/16/2020 0:59
9	37	7/29/2020 12:32
9	24	5/20/2020 11:37
10	1	1/21/2021 18:26
2	20	11/10/2020 4:40
4	21	9/15/2020 13:24
2	20	4/28/2020 13:16
7	30	11/29/2020 2:06

7	7	6/5/2020 5:20
4	1	6/1/2020 18:16
6	5	1/9/2021 6:40
2	29	2/15/2021 5:47
1	20	4/18/2021 8:30
8	10	7/1/2020 17:10
9	19	5/31/2020 10:38
1	15	6/25/2020 2:04
1	9	1/13/2021 18:53

OrdersFrom:

e_id	d_id	order_date
9	5	1/11/2021 12:39
7	3	9/30/2020 17:20
10	6	10/25/2020 17:20
2	3	5/2/2020 2:37
4	2	4/27/2020 4:03
8	6	8/17/2020 0:11
2	6	1/28/2021 17:58
7	9	12/25/2020 10:36
8	8	8/14/2020 15:31
2	8	8/9/2020 18:44
7	1	11/20/2020 19:03
8	8	8/10/2020 16:03

2	3	7/22/2020 18:58
9	7	2/27/2021 13:57
4	6	9/29/2020 22:57
4	10	5/26/2020 22:41
1	4	4/15/2021 10:11
3	2	12/10/2020 0:32
5	3	11/11/2020 9:53
5	10	11/6/2020 14:44
9	9	6/23/2020 19:41
8	8	7/13/2020 0:51
9	8	8/11/2020 10:18
10	9	11/6/2020 12:41
4	6	9/19/2020 8:07

4	4	11/25/2020 16:00
8	10	5/29/2020 3:50
8	2	12/31/2020 5:45
10	8	4/18/2021 20:22
9	9	12/4/2020 18:06
7	7	1/28/2021 21:14
8	7	12/21/2020 14:23
3	6	7/27/2020 3:39
10	6	5/18/2020 18:16
10	10	2/11/2021 13:05
6	2	12/23/2020 8:45
8	2	6/5/2020 19:53
2	6	11/15/2020 19:34

2	7	9/9/2020 17:55
5	9	12/10/2020 7:05

Covers:

c_id	i_id
1	2
2	6
3	2
4	5
5	6
6	1
7	4
8	4
9	1

10	2
11	5
12	2
13	3
14	5
15	5
16	2
17	3
18	4
19	4
20	4
21	6
22	6

23	4
24	5
25	2
26	4
27	6
28	4
29	4
30	4
31	5
32	3
33	3
34	2
35	2

36	3
37	5
38	2
39	5
40	3

Ships:

e_id	p_id	ship_date
7	31	7/15/2020 2:22
2	24	12/3/2020 12:55
1	35	8/20/2020 22:22
10	17	7/19/2020 18:38
2	8	12/7/2020 3:47

8	12	6/28/2020 14:11
1	7	11/21/2020 20:30
1	8	11/17/2020 18:37
6	14	9/14/2020 17:33
8	20	4/17/2021 17:13
2	25	3/27/2021 15:46
9	25	11/15/2020 6:24
8	32	5/9/2020 0:02
10	33	11/4/2020 12:55
1	28	8/25/2020 11:00
1	19	12/14/2020 3:29
3	34	10/16/2020 2:36
6	15	1/28/2021 7:02

1	34	10/4/2020 13:51
7	6	9/13/2020 4:37
4	37	7/4/2020 7:56
1	9	4/2/2021 1:04
9	11	11/3/2020 11:10
7	24	5/25/2020 11:08
7	10	6/29/2020 15:14
9	1	4/12/2021 14:39
8	27	8/25/2020 0:21
7	21	11/6/2020 0:13
6	31	11/24/2020 1:30
2	27	7/5/2020 3:26
2	16	10/27/2020 5:54

4	32	1/5/2021 16:06
2	10	5/24/2020 7:50
7	34	3/16/2021 4:09
1	19	11/17/2020 6:54
7	26	7/10/2020 10:00
10	23	12/12/2020 17:17
6	21	7/29/2020 18:00
1	25	9/12/2020 12:00
6	24	6/26/2020 9:47

Provides:

s_id	**p_id**	**provide_date**
10	20	10/28/2020 19:55

7	12	2/7/2021 0:39
8	23	7/6/2020 5:50
2	7	6/27/2020 12:33
10	28	2/2/2021 2:10
7	36	4/15/2021 5:08
3	16	7/26/2020 1:19
7	22	8/16/2020 13:28
2	24	11/17/2020 14:42
8	4	3/22/2021 3:55
8	1	5/17/2020 17:34
8	28	12/25/2020 3:46
7	6	11/10/2020 14:38
4	40	8/11/2020 6:09

9	3	12/27/2020 14:23
5	26	10/19/2020 16:08
1	1	12/22/2020 7:02
3	15	11/20/2020 7:00
3	7	12/17/2020 8:55
1	15	2/19/2021 7:09
2	10	3/30/2021 4:08
10	40	11/13/2020 17:05
8	19	10/27/2020 12:46
7	22	7/26/2020 4:07
6	26	12/9/2020 21:51
6	8	10/24/2020 2:38
1	17	2/13/2021 4:49

7	18	4/15/2021 11:38
5	28	4/14/2021 21:04
5	23	10/27/2020 12:56
10	33	4/13/2021 15:52
9	40	10/22/2020 6:21
10	2	8/18/2020 23:16
1	33	11/6/2020 13:24
3	33	7/12/2020 3:53
3	5	6/26/2020 20:21
8	20	7/21/2020 18:04
10	13	10/12/2020 8:40
7	16	6/11/2020 23:48
4	19	1/5/2021 1:41

2.4 - Sample Queries

Our database for Medicine Dealers gives the ability to query data from several key components in the transactions of medicines from our store to the customer. The key components of our database are customer data, product data, and employee data. The components make transactions and querying data about products and customers easy. Also, we have data about orders so we can see how many orders went out between certain dates and can track how much product we have left in stock of certain items. This database provides us with a rich amount of options to query data about the transactions of medicines from the seller to the buyer.

2.4.1 Design Of Queries

1. Select the cheapest drug (before discount).
2. Select all customers who ordered Ibuprofen.
3. Select all customers who live in Bakersfield.
4. Select all employees making over $60,000.
5. Select all drugs that are below 50 in quantity.

6. Select drugs that cost over $90.
7. Select all customers whose doctor is John Dorian.
8. Select all customers who have a prescription of Lidocaine Hydrochloride .
9. Select all customers with the first name Carl.
10. Select all customers who made an order between 1-1-2019 and 1-1-2020.
11. Select suppliers that every employee has ordered from.
12. Select customers who have purchased all products.

2.4.2 Relational Algebra Expressions

Relational algebra is the component of the relational data model that describes the data behavior of the retrieval requests from the database application. It is highly used in procedural query languages and can assist its users to query the database instances.

SYMBOLS: π σ ⋈ ←

1. Select the cheapest drug (before discount).

   $D \leftarrow \pi_{p\_id,\ p\_price}((\sigma_{min(cost)}(\sigma_{p\_id}\ Product)) \bowtie Product)$

2. Select all customers who ordered Ibuprofen.

   $O \leftarrow \pi_{c\_id}((\sigma_{o\_product\ =\ \text{'IIbuprofen'}}\ Order) \bowtie_{o\_product\ =\ o\_product} Purchases)$

3. Select all customers who live in Bakersfield.

   $CB \leftarrow \pi_{c\_id}(\sigma_{c\_address\ =\ \text{"Bakersfield"}}\ Customer)$

4. Select all employees making over $60,000.

   $ES \leftarrow \pi_{e\_fname,\ e\_lname}(\sigma_{e\_salary\ >\ 60{,}000}\ Employee)$

5. Select all drugs that are below 50 in quantity.

   $Q \leftarrow \pi_{p\_id}(\sigma_{p\_quantity\ <\ 50}\ Product)$

6. Select drugs that cost over $90.

   $D \leftarrow \pi_{p\_id}(\sigma_{p\_price\ >\ 90}\ Product)$

7. Select all customers whose doctor is John Dorian.

   $Oz \leftarrow \pi_{e\_fname,\ e\_lname}(\sigma_{c\_doctor\ =\ \text{'John Dorian'}}\ Customer)$

8. Select all customers who have a prescription of Lidocaine Hydrochloride.

   $C \leftarrow \pi_{c\_fname,\ c\_lname,\ c\_id}((\sigma_{p\_name\ =\text{'Lidocaine Hydrochloride'}}\ Product) \bowtie Prescription) \bowtie Customer)$

9. Select all customers with the first name Carl.

   $C \leftarrow \pi_{c\_fname}(\sigma_{c\_fname = \text{'Carl'}} \text{ Customer})$

10. Select all customers who made an order between 1-1-2019 and 1-1-2020.

    $A \leftarrow \pi_{c\_id}((\sigma_{o\_shipDate > \text{' 2019 - 01 - 01'} \wedge \text{'}o\_shipDate < \text{' 2020 - 01 - 01'}} \text{ Order}) \bowtie \text{Customer})$

11. Select suppliers that every employee has ordered from.

    $E \leftarrow \pi_{e\_id}(\text{Employee})$

    $S1 \leftarrow \text{OrdersFrom} \div E$

    $S2 \leftarrow \pi_{s\_id}(S1) * \text{Supplier}$

12. Select customers who have purchased all products.

    $P \leftarrow \pi_{p\_id}(\text{Product})$

    $C \leftarrow \text{Contains} \div P$

    $C1 \leftarrow \pi_{c\_id}(C) * \text{Customer}$

2.4.3 Relational Calculus Expressions

Relational calculus is a non procedural query language used for relation queries. Relational calculus has two forms, tuple relational calculus (TRC) and domain relational calculus (DRC), they are very similar in formatting but TRC uses tuples when selecting the values and DRC only selects a few attributes to be found.

1. Select the cheapest drug (before discount).

   {p | Product(p) ^ ~∃d(Product(d) ^ d.p_cost < p.p_cost ^ d.p_id != p.p_id)}

2. Select all customers who ordered Ibuprofen.

   {c.c_fname, c.c_lname |(Customer(c) ^ ∃o(Order(o) ^ o.c_id = c.c_id ^ ∃o(Contains(i) ^ i.o_id = o.o_id ^ ∃p(Product(p) ^ p.p_id = i.p_id ^ p.name = 'Ibuprofen'))))}

3. Select all customers who live in Bakersfield.

   {c.c_fname, c.c_lname | Customer(c) ^ c.c_address = "Bakersfield"}

4. Select all employees making over $60,000.

{e.e_Fname, e.e_Lname | Employee(e) ^ e.salary > 60000}

5. Select all drugs that are below 50 in quantity.

   {p.pname | Product(p) ^ p.p_quantity < 50}

6. Select drugs that cost over $90.

   {p.p_name | Product(p) ^ p.p_price > 90}

7. Select all customers whose doctor is John Dorian.

   {c.c_fname, c.c_lname | Customer(c) ^ c.c_doctor = “John Dorian”}

8. Select all customers who have insurance.

   {c.c_fname, c.c_lname | Customer(c) ^ ∃p(Covers(p) ^ p.c_id = c.c_id ^ ∃i(Insurance(i) ^ i.i_id = p.i_id))}

9. Select all customers with the first name Carl.

   {c.c_fname, c.c_lname | Customer(c) ^ c.fname = “Carl” }

10. Select all customers who made an order on or between 1-1-2019 and 1-1-2020.

    {c.c_fname, c.c_lname | ∃c(Customer(c) ^ ∃o(Order(o) ^ c.c_id = o.c_id ^ o_shipDate <= “1-1-2020” ^ o_shipDate >= “1-1-2019”)}

Phase 3: Physical Implementation and Relational Normalization

3.1 - Relational Normalization

3.1.1 Normalization Process

Normalization is a process taken when designing a database to help reduce redundant data. By reducing large tables into more detailed smaller tables, you can minimize redundancy in the database. This also helps keep the tables easier to read when implementing and makes it easier to do different procedures.

First normal form:

This form is used to normalize relations that do not have any composite or multi-valued attributes. This can be done by creating a new relation for the multi-valued attribute containing all the attributes as well as the primary key of the entities it has a relationship with.

Second normal form:

This form's criteria is that all non-primary attributes should be fully dependent on all of the primary keys. One way to do this is to split up the relation into other relations that contain non-primary attributes. The new relations that were created will also maintain a relationship with the original primary key and all of its functional dependent attributes.

Third normal form:

This form is used to identify relations that have no nonprime attributes that are dependent on its primary key. The relation should be broken up so that each relation does not have any transitive dependencies and every non-prime attribute has full functional dependencies.

Boyce-Codd normal form:

This form is similar to the third normal form. The difference is when there are two or more overlapping candidate keys. This form needs to break up the relation into two separate relations, similar to the second normal form.

Anomalies:

Multiple problems can happen in a database if the relational model is not normalized. These types of problems or anomalies can happen during insertion, deletion, and/or updating of the data. Insertion anomalies can happen when trying to insert a new tuple into a relation that contains the same data. Deletion anomalies happen when deleting one tuple ends up deleting another tuple that was later needed. Update anomalies happen when a change in a value makes the need to update previous data. If data from one place of the databases are being updated and is used elsewhere then all similar data needs to be updated.

Relationship between normalization and update anomalies

Normalization of structures in the relation model can help reduce the redundancies and increase the safety of data. Depending on how well this is done, it can help reduce your risk of anomalies in the future and save you headaches. Normalization is the key to properly checking your relation model.

3.1.2 Application to Relational Model

Customer(c_id, c_fname, c_lname, c_phonenum, c_email, c_password, c_doctor, c_dob, c_address **)**

1. This is in the third normal form.
2. This relation has no anomalies

Employee(e_id, e_fname, e_lname, e_salary, e_position, e_ssn, e_address, e_email, e_password **)**

1. This is in the third normal form.
2. This relation has no anomalies

Product(p_id, p_price, p_name, p_supplier, p_quantity, p_PrescriptionNeeded **)**

1. This is in the third normal form.
2. This relation has no anomalies

Insurance(i_id, i_name, i_discount **)**

1. This is in the third normal form.
2. This relation has no anomalies

Transformation:

Insurance(i_name, i_discount)

Supplier(s_id, s_name **)**

1. This is in the third normal form.
2. This relation has no anomalies

Orders(c_id, o_id, o_shipDate**)**

1. This is in the third normal form.
2. This relation has no anomalies

Discounts(i_id, p_id, d_dicsount **)**

1. This is in the third normal form.
2. This relation has no anomalies

Prescription(p_id, c_id, pre_doctorName, pre_lastFilled, pre_refill **)**

1. This is in the third normal form.
2. This relation has no anomalies

Contains(o_id, p_id, p_quantity **)**

1. This is in the third normal form.
2. This relation has no anomalies

Restocks(e_id, p_id, restock_date **)**

1. This is in the third normal form.
2. This relation has no anomalies

OrdersFrom(e_id, s_id, order_date **)**

1. This is in the third normal form.
2. This relation has no anomalies

Covers(c_id, i_id **)**

1. This is in the third normal form.
2. This relation has no anomalies

Ships(o_id, e_id, ship_date **)**

1. This is in the third normal form.
2. This relation has no anomalies

Provides(p_id, s_id, _date **)**

1. This is in the third normal form.
2. This relation has no anomalies

3.2 - Database Implementation

3.2.1 Background Information

The main purpose of a relational database management system (RDBMS) is to create a relational database that you can create easily and modify to your liking. RDBMS is a more improved version of DBMS, the biggest difference is that DBMS saves data into files, while RDBMS saves data into tables. A few advantages of RDBMS are higher security and better data integrity.

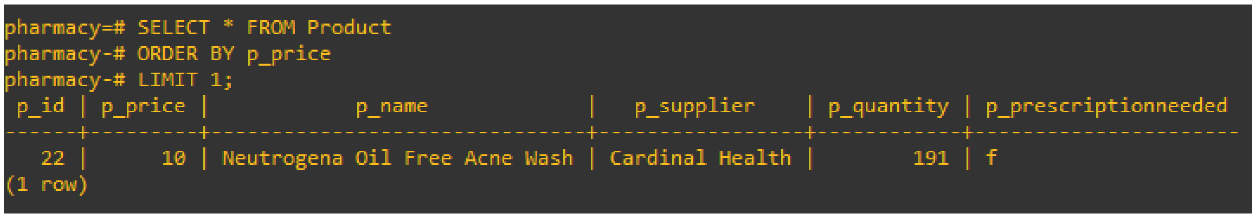
3.2.2 Schema and Hosting

We will be using our group member's server to host PostgreSQL for our RDBMS. The self server is being hosted through Microsoft Azure and running Debian 10. A few reasons we chose this is because of experience from a group member and ease of use. PostgreSQL is also very commonly used and can be very helpful when errors come up.

3.3 - Query Implementation

1.Select the cheapest drug (before discount).

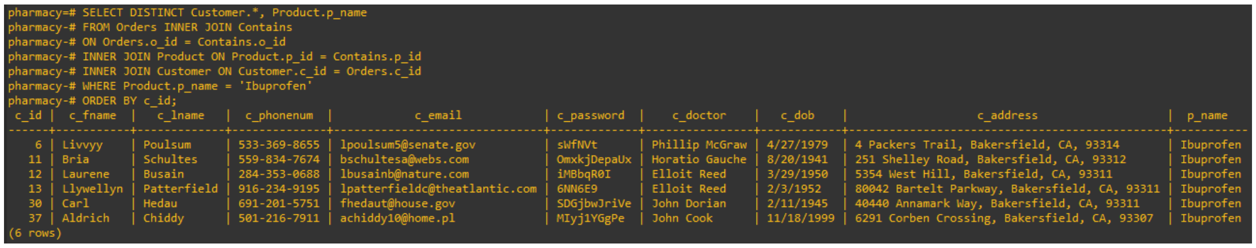
```
        SELECT * FROM Product
        ORDER BY p_price
        LIMIT 1;
```



```
pharmacy=# SELECT * FROM Product
pharmacy-# ORDER BY p_price
pharmacy-# LIMIT 1;
 p_id | p_price |            p_name             |   p_supplier    | p_quantity | p_prescriptionneeded
------+---------+-------------------------------+-----------------+------------+----------------------
   22 |      10 | Neutrogena Oil Free Acne Wash | Cardinal Health |        191 | f
(1 row)
```

2.Select all customers who ordered Ibuprofen.

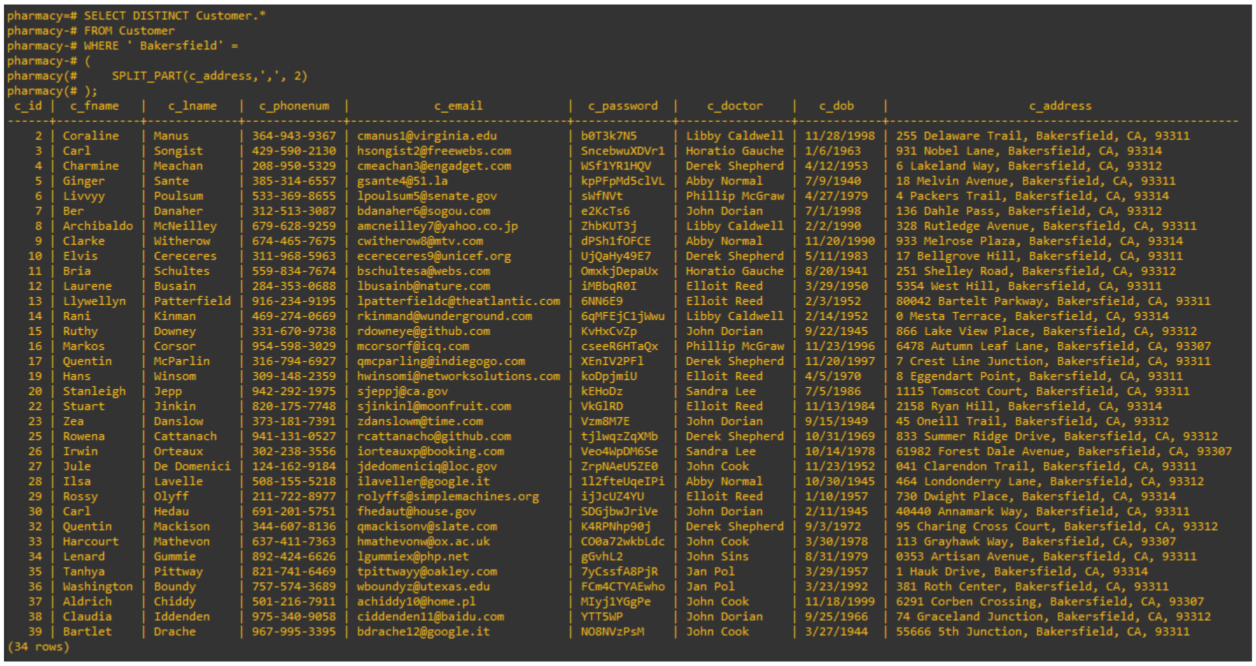
```
        SELECT DISTINCT Customer.*, Product.p_name
        FROM Orders INNER JOIN Contains
                ON Orders.o_id = Contains.o_id
                INNER JOIN Product ON Product.p_id = Contains.p_id
                INNER JOIN Customer ON Customer.c_id = Orders.c_id
        WHERE Product.p_name = 'Ibuprofen'
        ORDER BY c_id;
```



```
pharmacy=# SELECT DISTINCT Customer.*, Product.p_name
pharmacy-# FROM Orders INNER JOIN Contains
pharmacy-# ON Orders.o_id = Contains.o_id
pharmacy-# INNER JOIN Product ON Product.p_id = Contains.p_id
pharmacy-# INNER JOIN Customer ON Customer.c_id = Orders.c_id
pharmacy-# WHERE Product.p_name = 'Ibuprofen'
pharmacy-# ORDER BY c_id;
 c_id |  c_fname  |   c_lname   |  c_phonenum  |            c_email            | c_password  |    c_doctor    |   c_dob    |                   c_address                   |  p_name
------+-----------+-------------+--------------+-------------------------------+-------------+----------------+------------+-----------------------------------------------+-----------
    6 | Livvyy    | Poulsum     | 533-369-8655 | lpoulsum5@senate.gov          | sWfNVt      | Phillip McGraw | 4/27/1979  | 4 Packers Trail, Bakersfield, CA, 93314       | Ibuprofen
   11 | Bria      | Schultes    | 559-834-7674 | bschultesa@webs.com           | OmxkjDepaUx | Horatio Gauche | 8/20/1941  | 251 Shelley Road, Bakersfield, CA, 93312      | Ibuprofen
   12 | Laurene   | Busain      | 284-353-0688 | lbusainb@nature.com           | iMBbqR0I    | Elloit Reed    | 3/29/1950  | 5354 West Hill, Bakersfield, CA, 93311        | Ibuprofen
   13 | Llywellyn | Patterfield | 916-234-9195 | lpatterfieldc@theatlantic.com | 6NN6E9      | Elloit Reed    | 2/3/1952   | 80042 Bartelt Parkway, Bakersfield, CA, 93311 | Ibuprofen
   30 | Carl      | Hedau       | 691-201-5751 | fhedaut@house.gov             | SDGjbwJriVe | John Dorian    | 2/11/1945  | 40440 Annamark Way, Bakersfield, CA, 93311    | Ibuprofen
   37 | Aldrich   | Chiddy      | 501-216-7911 | achiddy10@home.pl             | MIyj1YGgPe  | John Cook      | 11/18/1999 | 6291 Corben Crossing, Bakersfield, CA, 93307  | Ibuprofen
(6 rows)
```

3.Select all customers who live in Bakersfield.

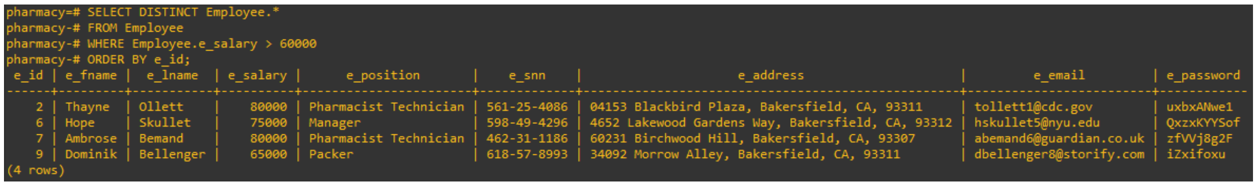
```
SELECT DISTINCT Customer.*
FROM Customer
WHERE ' Bakersfield' =
(
        SPLIT_PART(c_address, ',', 2)
);
```



```
pharmacy=# SELECT DISTINCT Customer.*
pharmacy-# FROM Customer
pharmacy-# WHERE ' Bakersfield' =
pharmacy-# (
pharmacy(#     SPLIT_PART(c_address,',', 2)
pharmacy(# );
 c_id |  c_fname   |   c_lname   |  c_phonenum  |            c_email            |  c_password  |    c_doctor    |   c_dob    |                    c_address
------+------------+-------------+--------------+-------------------------------+--------------+----------------+------------+--------------------------------------------------
    2 | Coraline   | Manus       | 364-943-9367 | cmanus1@virginia.edu          | b0T3k7N5     | Libby Caldwell | 11/28/1998 | 255 Delaware Trail, Bakersfield, CA, 93311
    3 | Carl       | Songist     | 429-590-2130 | hsongist2@freewebs.com        | SncebwuXDVr1 | Horatio Gauche | 1/6/1963   | 931 Nobel Lane, Bakersfield, CA, 93314
    4 | Charmine   | Meachan     | 208-950-5329 | cmeachan3@engadget.com        | WSf1YR1HQV   | Derek Shepherd | 4/12/1953  | 6 Lakeland Way, Bakersfield, CA, 93312
    5 | Ginger     | Sante       | 385-314-6557 | gsante4@51.la                 | kpPFpMd5clVL | Abby Normal    | 7/9/1940   | 18 Melvin Avenue, Bakersfield, CA, 93311
    6 | Livvyy     | Poulsum     | 533-369-8655 | lpoulsum5@senate.gov          | sWfNVt       | Phillip McGraw | 4/27/1979  | 4 Packers Trail, Bakersfield, CA, 93314
    7 | Ber        | Danaher     | 312-513-3087 | bdanaher6@sogou.com           | e2KcTs6      | John Dorian    | 7/1/1998   | 136 Dahle Pass, Bakersfield, CA, 93312
    8 | Archibaldo | McNeilley   | 679-628-9259 | amcneilley7@yahoo.co.jp       | ZhbKUT3j     | Libby Caldwell | 2/2/1990   | 328 Rutledge Avenue, Bakersfield, CA, 93311
    9 | Clarke     | Witherow    | 674-465-7675 | cwitherow8@mtv.com            | dPSh1fOFCE   | Abby Normal    | 11/20/1990 | 933 Melrose Plaza, Bakersfield, CA, 93314
   10 | Elvis      | Cereceres   | 311-968-5963 | ecereceres9@unicef.org        | UjQaHy49E7   | Derek Shepherd | 5/11/1983  | 17 Bellgrove Hill, Bakersfield, CA, 93311
   11 | Bria       | Schultes    | 559-834-7674 | bschultesa@webs.com           | OmxkjDepaUx  | Horatio Gauche | 8/20/1941  | 251 Shelley Road, Bakersfield, CA, 93312
   12 | Laurene    | Busain      | 284-353-0688 | lbusainb@nature.com           | iMBbqR0I     | Elloit Reed    | 3/29/1950  | 5354 West Hill, Bakersfield, CA, 93311
   13 | Llywellyn  | Patterfield | 916-234-9195 | lpatterfieldc@theatlantic.com | 6NN6E9       | Elloit Reed    | 2/3/1952   | 80042 Bartelt Parkway, Bakersfield, CA, 93311
   14 | Rani       | Kinman      | 469-274-0669 | rkinmand@wunderground.com     | 6qMFEjC1jWwu | Libby Caldwell | 2/14/1952  | 0 Mesta Terrace, Bakersfield, CA, 93314
   15 | Ruthy      | Downey      | 331-670-9738 | rdowneye@github.com           | KvHxCvZp     | John Dorian    | 9/22/1945  | 866 Lake View Place, Bakersfield, CA, 93312
   16 | Markos     | Corsor      | 954-598-3029 | mcorsorf@icq.com              | cseeR6HTaQx  | Phillip McGraw | 11/23/1996 | 6478 Autumn Leaf Lane, Bakersfield, CA, 93307
   17 | Quentin    | McParlin    | 316-794-6927 | qmcparling@indiegogo.com      | XEnIV2PFl    | Derek Shepherd | 11/20/1997 | 7 Crest Line Junction, Bakersfield, CA, 93311
   19 | Hans       | Winsom      | 309-148-2359 | hwinsomi@networksolutions.com | koDpjmiU     | Elloit Reed    | 4/5/1970   | 8 Eggendart Point, Bakersfield, CA, 93311
   20 | Stanleigh  | Jepp        | 942-292-1975 | sjeppj@ca.gov                 | kEHoDz       | Sandra Lee     | 7/5/1986   | 1115 Tomscot Court, Bakersfield, CA, 93311
   22 | Stuart     | Jinkin      | 820-175-7748 | sjinkinl@moonfruit.com        | VkGlRD       | Elloit Reed    | 11/13/1984 | 2158 Ryan Hill, Bakersfield, CA, 93314
   23 | Zea        | Danslow     | 373-181-7391 | zdanslowm@time.com            | Vzm8M7E      | John Dorian    | 9/15/1949  | 45 Oneill Trail, Bakersfield, CA, 93312
   25 | Rowena     | Cattanach   | 941-131-0527 | rcattanacho@github.com        | tjlwqzZqXMb  | Derek Shepherd | 10/31/1969 | 833 Summer Ridge Drive, Bakersfield, CA, 93312
   26 | Irwin      | Orteaux     | 302-238-3556 | iorteauxp@booking.com         | Veo4WpDM6Se  | Sandra Lee     | 10/14/1978 | 61982 Forest Dale Avenue, Bakersfield, CA, 93307
   27 | Jule       | De Domenici | 124-162-9184 | jdedomeniciq@loc.gov          | ZrpNAeU5ZE0  | John Cook      | 11/23/1952 | 041 Clarendon Trail, Bakersfield, CA, 93311
   28 | Ilsa       | Lavelle     | 508-155-5218 | ilaveller@google.it           | 1l2fteUqeIPi | Abby Normal    | 10/30/1945 | 464 Londonderry Lane, Bakersfield, CA, 93312
   29 | Rossy      | Olyff       | 211-722-8977 | rolyffs@simplemachines.org    | ijJcUZ4YU    | Elloit Reed    | 1/10/1957  | 730 Dwight Place, Bakersfield, CA, 93314
   30 | Carl       | Hedau       | 691-201-5751 | fhedaut@house.gov             | SDGjbwJriVe  | John Dorian    | 2/11/1945  | 40440 Annamark Way, Bakersfield, CA, 93311
   32 | Quentin    | Mackison    | 344-607-8136 | qmackisonv@slate.com          | K4RPNhp90j   | Derek Shepherd | 9/3/1972   | 95 Charing Cross Court, Bakersfield, CA, 93312
   33 | Harcourt   | Mathevon    | 637-411-7363 | hmathevonw@ox.ac.uk           | CO0a72wkbLdc | John Cook      | 3/30/1978  | 113 Grayhawk Way, Bakersfield, CA, 93307
   34 | Lenard     | Gummie      | 892-424-6626 | lgummiex@php.net              | gGvhL2       | John Sins      | 8/31/1979  | 0353 Artisan Avenue, Bakersfield, CA, 93311
   35 | Tanhya     | Pittway     | 821-741-6469 | tpittwayy@oakley.com          | 7yCssfA8PjR  | Jan Pol        | 3/29/1957  | 1 Hauk Drive, Bakersfield, CA, 93314
   36 | Washington | Boundy      | 757-574-3689 | wboundyz@utexas.edu           | FCm4CTYAEwho | Jan Pol        | 3/23/1992  | 381 Roth Center, Bakersfield, CA, 93311
   37 | Aldrich    | Chiddy      | 501-216-7911 | achiddy10@home.pl             | MIyj1YGgPe   | John Cook      | 11/18/1999 | 6291 Corben Crossing, Bakersfield, CA, 93307
   38 | Claudia    | Iddenden    | 975-340-9058 | ciddenden11@baidu.com         | YTT5WP       | John Dorian    | 9/25/1966  | 74 Graceland Junction, Bakersfield, CA, 93312
   39 | Bartlet    | Drache      | 967-995-3395 | bdrache12@google.it           | NO8NVzPsM    | John Cook      | 3/27/1944  | 55666 5th Junction, Bakersfield, CA, 93311
(34 rows)
```

4.Select all employees making over $60,000

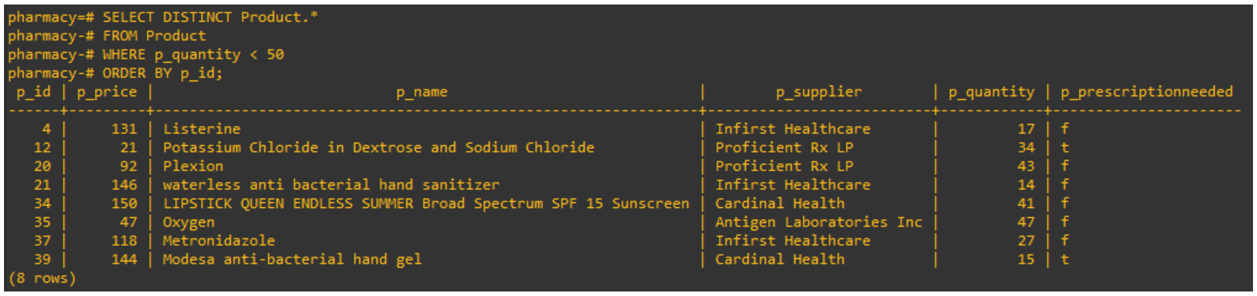
```
SELECT DISTINCT Employee.*
FROM Employee
WHERE Employee.e_salary > 60000
ORDER BY e_id;
```



```
pharmacy=# SELECT DISTINCT Employee.*
pharmacy-# FROM Employee
pharmacy-# WHERE Employee.e_salary > 60000
pharmacy-# ORDER BY e_id;
 e_id | e_fname |  e_lname  | e_salary |      e_position       |    e_snn    |                     e_address                     |          e_email           | e_password
------+---------+-----------+----------+-----------------------+-------------+---------------------------------------------------+----------------------------+------------
    2 | Thayne  | Ollett    |    80000 | Pharmacist Technician | 561-25-4086 | 04153 Blackbird Plaza, Bakersfield, CA, 93311     | tollett1@cdc.gov           | uxbxANwe1
    6 | Hope    | Skullet   |    75000 | Manager               | 598-49-4296 | 4652 Lakewood Gardens Way, Bakersfield, CA, 93312 | hskullet5@nyu.edu          | QxzxKYYSof
    7 | Ambrose | Bemand    |    80000 | Pharmacist Technician | 462-31-1186 | 60231 Birchwood Hill, Bakersfield, CA, 93307      | abemand6@guardian.co.uk    | zfVVj8g2F
    9 | Dominik | Bellenger |    65000 | Packer                | 618-57-8993 | 34092 Morrow Alley, Bakersfield, CA, 93311        | dbellenger8@storify.com    | iZxifoxu
(4 rows)
```

5.Select all drugs that are below 50 in quantity.

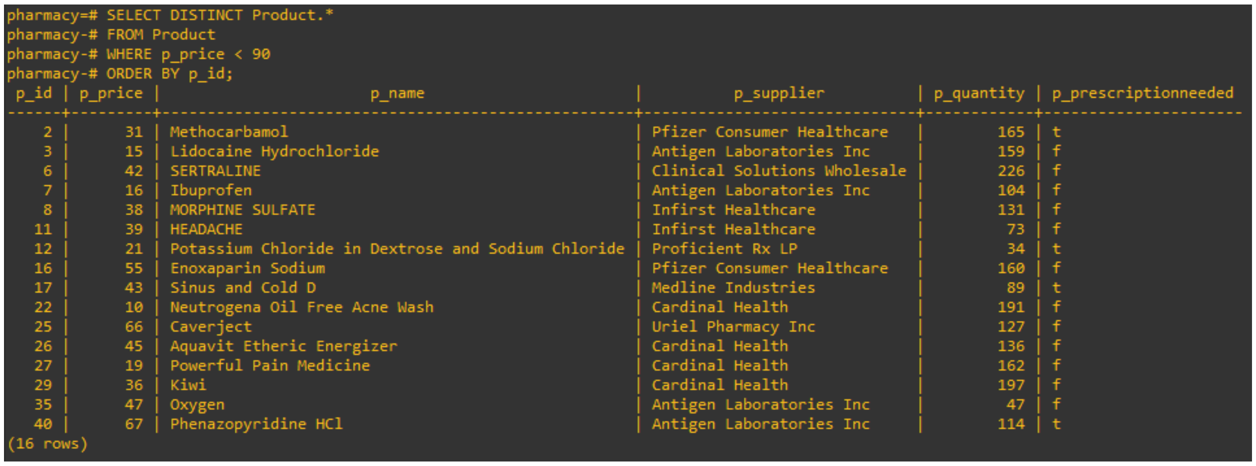
SELECT DISTINCT Product.*
FROM Product
WHERE p_quantity < 50
ORDER BY p_id;



```
pharmacy=# SELECT DISTINCT Product.*
pharmacy-# FROM Product
pharmacy-# WHERE p_quantity < 50
pharmacy-# ORDER BY p_id;
 p_id | p_price |                          p_name                           |         p_supplier         | p_quantity | p_prescriptionneeded
------+---------+-----------------------------------------------------------+----------------------------+------------+----------------------
    4 |     131 | Listerine                                                 | Infirst Healthcare         |         17 | f
   12 |      21 | Potassium Chloride in Dextrose and Sodium Chloride        | Proficient Rx LP           |         34 | t
   20 |      92 | Plexion                                                   | Proficient Rx LP           |         43 | f
   21 |     146 | waterless anti bacterial hand sanitizer                   | Infirst Healthcare         |         14 | f
   34 |     150 | LIPSTICK QUEEN ENDLESS SUMMER Broad Spectrum SPF 15 Sunscreen | Cardinal Health        |         41 | f
   35 |      47 | Oxygen                                                    | Antigen Laboratories Inc   |         47 | f
   37 |     118 | Metronidazole                                             | Infirst Healthcare         |         27 | f
   39 |     144 | Modesa anti-bacterial hand gel                            | Cardinal Health            |         15 | t
(8 rows)
```

6.Select drugs that cost over $90.

SELECT DISTINCT Product.*
FROM Product
WHERE p_price < 90
ORDER BY p_id;

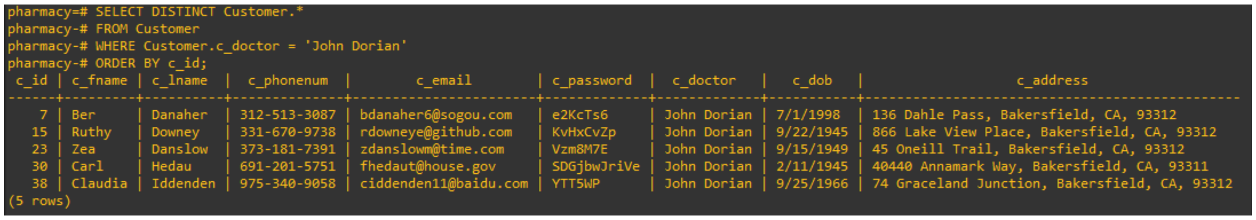


```
pharmacy=# SELECT DISTINCT Product.*
pharmacy-# FROM Product
pharmacy-# WHERE p_price < 90
pharmacy-# ORDER BY p_id;
 p_id | p_price |                       p_name                       |         p_supplier          | p_quantity | p_prescriptionneeded
------+---------+----------------------------------------------------+-----------------------------+------------+----------------------
    2 |      31 | Methocarbamol                                      | Pfizer Consumer Healthcare  |        165 | t
    3 |      15 | Lidocaine Hydrochloride                            | Antigen Laboratories Inc    |        159 | f
    6 |      42 | SERTRALINE                                         | Clinical Solutions Wholesale |       226 | f
    7 |      16 | Ibuprofen                                          | Antigen Laboratories Inc    |        104 | f
    8 |      38 | MORPHINE SULFATE                                   | Infirst Healthcare          |        131 | f
   11 |      39 | HEADACHE                                           | Infirst Healthcare          |         73 | f
   12 |      21 | Potassium Chloride in Dextrose and Sodium Chloride | Proficient Rx LP            |         34 | t
   16 |      55 | Enoxaparin Sodium                                  | Pfizer Consumer Healthcare  |        160 | f
   17 |      43 | Sinus and Cold D                                   | Medline Industries          |         89 | t
   22 |      10 | Neutrogena Oil Free Acne Wash                      | Cardinal Health             |        191 | f
   25 |      66 | Caverject                                          | Uriel Pharmacy Inc          |        127 | f
   26 |      45 | Aquavit Etheric Energizer                          | Cardinal Health             |        136 | f
   27 |      19 | Powerful Pain Medicine                             | Cardinal Health             |        162 | f
   29 |      36 | Kiwi                                               | Cardinal Health             |        197 | f
   35 |      47 | Oxygen                                             | Antigen Laboratories Inc    |         47 | f
   40 |      67 | Phenazopyridine HCl                                | Antigen Laboratories Inc    |        114 | t
(16 rows)
```

7. Select all customers whose doctor is John Dorian.

SELECT DISTINCT Customer.*
FROM Customer
WHERE Customer.c_doctor = 'John Dorian'

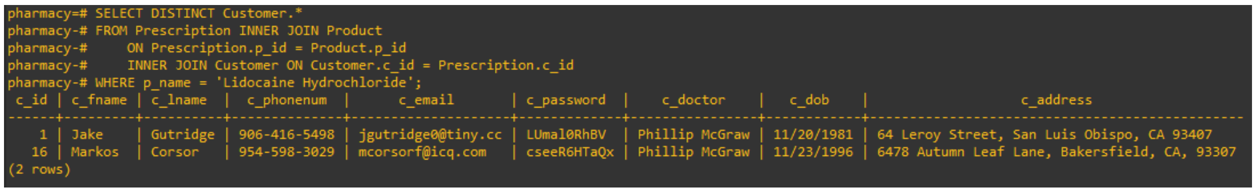
```
        ORDER BY c_id;
```



```
pharmacy=# SELECT DISTINCT Customer.*
pharmacy-# FROM Customer
pharmacy-# WHERE Customer.c_doctor = 'John Dorian'
pharmacy-# ORDER BY c_id;
 c_id | c_fname | c_lname  |  c_phonenum  |        c_email        | c_password |  c_doctor   |   c_dob   |                  c_address
------+---------+----------+--------------+-----------------------+------------+-------------+-----------+---------------------------------------------
    7 | Ber     | Danaher  | 312-513-3087 | bdanaher6@sogou.com   | e2KcTs6    | John Dorian | 7/1/1998  | 136 Dahle Pass, Bakersfield, CA, 93312
   15 | Ruthy   | Downey   | 331-670-9738 | rdowneye@github.com   | KvHxCvZp   | John Dorian | 9/22/1945 | 866 Lake View Place, Bakersfield, CA, 93312
   23 | Zea     | Danslow  | 373-181-7391 | zdanslowm@time.com    | Vzm8M7E    | John Dorian | 9/15/1949 | 45 Oneill Trail, Bakersfield, CA, 93312
   30 | Carl    | Hedau    | 691-201-5751 | fhedaut@house.gov     | SDGjbwJriVe | John Dorian | 2/11/1945 | 40440 Annamark Way, Bakersfield, CA, 93311
   38 | Claudia | Iddenden | 975-340-9058 | ciddenden11@baidu.com | YTT5WP     | John Dorian | 9/25/1966 | 74 Graceland Junction, Bakersfield, CA, 93312
(5 rows)
```

8. Select all customers who have a prescription of Lidocaine Hydrochloride

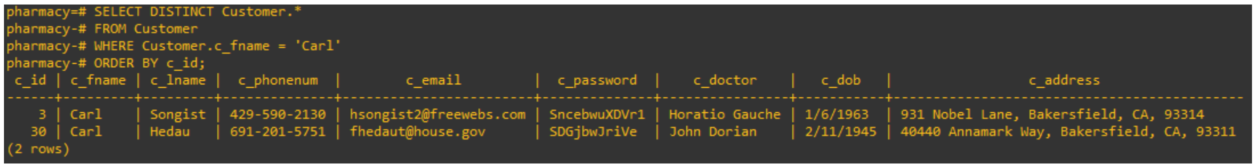
```
        SELECT DISTINCT Customer.*
        FROM Prescription INNER JOIN Product
                ON Prescription.p_id = Product.p_id
                INNER JOIN Customer ON Customer.c_id = Prescription.c_id
        WHERE p_name = 'Lidocaine Hydrochloride';
```



```
pharmacy=# SELECT DISTINCT Customer.*
pharmacy-# FROM Prescription INNER JOIN Product
pharmacy-#     ON Prescription.p_id = Product.p_id
pharmacy-#     INNER JOIN Customer ON Customer.c_id = Prescription.c_id
pharmacy-# WHERE p_name = 'Lidocaine Hydrochloride';
 c_id | c_fname | c_lname  |  c_phonenum  |      c_email       | c_password |   c_doctor    |   c_dob    |                  c_address
------+---------+----------+--------------+--------------------+------------+---------------+------------+---------------------------------------------
    1 | Jake    | Gutridge | 906-416-5498 | jgutridge0@tiny.cc | LUmal0RhBV | Phillip McGraw | 11/20/1981 | 64 Leroy Street, San Luis Obispo, CA 93407
   16 | Markos  | Corsor   | 954-598-3029 | mcorsorf@icq.com   | cseeR6HTaQx | Phillip McGraw | 11/23/1996 | 6478 Autumn Leaf Lane, Bakersfield, CA, 93307
(2 rows)
```

9. Select all customers with the first name Carl--

```
        SELECT DISTINCT Customer.*
        FROM Customer
        WHERE Customer.c_fname = 'Carl'
        ORDER BY c_id;
```

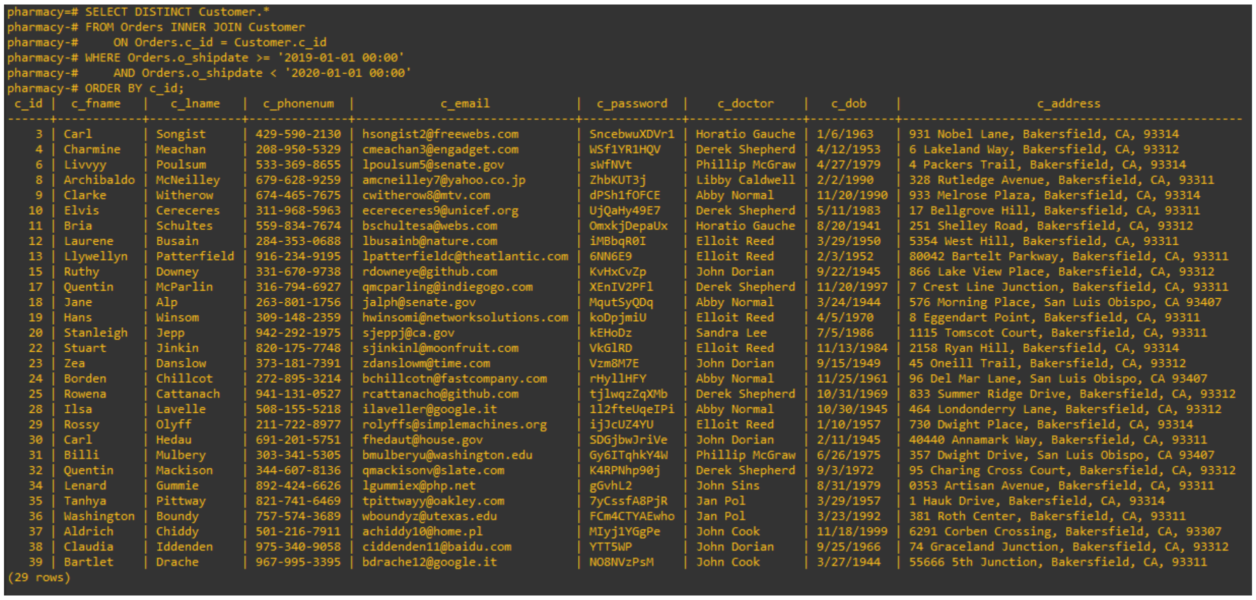


```
pharmacy=# SELECT DISTINCT Customer.*
pharmacy-# FROM Customer
pharmacy-# WHERE Customer.c_fname = 'Carl'
pharmacy-# ORDER BY c_id;
 c_id | c_fname | c_lname |  c_phonenum  |        c_email         |  c_password  |    c_doctor    |   c_dob   |                 c_address
------+---------+---------+--------------+------------------------+--------------+----------------+-----------+--------------------------------------------
    3 | Carl    | Songist | 429-590-2130 | hsongist2@freewebs.com | SncebwuXDVr1 | Horatio Gauche | 1/6/1963  | 931 Nobel Lane, Bakersfield, CA, 93314
   30 | Carl    | Hedau   | 691-201-5751 | fhedaut@house.gov      | SDGjbwJriVe  | John Dorian    | 2/11/1945 | 40440 Annamark Way, Bakersfield, CA, 93311
(2 rows)
```

10. Select all customers who made an order between 1-1-2019 and 1-1-2020--

```
        SELECT DISTINCT Customer.*
        FROM Orders INNER JOIN Customer
           ON Orders.c_id = Customer.c_id
```

```
WHERE Orders.o_shipdate >= '2019-01-01 00:00'
    AND Orders.o_shipdate < '2020-01-01 00:00'
ORDER BY c_id;
```



```
pharmacy=# SELECT DISTINCT Customer.*
pharmacy-# FROM Orders INNER JOIN Customer
pharmacy-#     ON Orders.c_id = Customer.c_id
pharmacy-# WHERE Orders.o_shipdate >= '2019-01-01 00:00'
pharmacy-#     AND Orders.o_shipdate < '2020-01-01 00:00'
pharmacy-# ORDER BY c_id;
 c_id |  c_fname   |   c_lname   |  c_phonenum  |            c_email            |  c_password  |    c_doctor    |   c_dob    |                   c_address
------+------------+-------------+--------------+-------------------------------+--------------+----------------+------------+------------------------------------------------
    3 | Carl       | Songist     | 429-590-2130 | hsongist2@freewebs.com        | SncebwuXDVr1 | Horatio Gauche | 1/6/1963   | 931 Nobel Lane, Bakersfield, CA, 93314
    4 | Charmine   | Meachan     | 208-950-5329 | cmeachan3@engadget.com        | WSf1YR1HQV   | Derek Shepherd | 4/12/1953  | 6 Lakeland Way, Bakersfield, CA, 93312
    6 | Livvyy     | Poulsum     | 533-369-8655 | lpoulsum5@senate.gov          | sWfNVt       | Phillip McGraw | 4/27/1979  | 4 Packers Trail, Bakersfield, CA, 93314
    8 | Archibaldo | McNeilley   | 679-628-9259 | amcneilley7@yahoo.co.jp       | ZhbKUT3j     | Libby Caldwell | 2/2/1990   | 328 Rutledge Avenue, Bakersfield, CA, 93311
    9 | Clarke     | Witherow    | 674-465-7675 | cwitherow8@mtv.com            | dPSh1fOFCE   | Abby Normal    | 11/20/1990 | 933 Melrose Plaza, Bakersfield, CA, 93314
   10 | Elvis      | Cereceres   | 311-968-5963 | ecereceres9@unicef.org        | UjQaHy49E7   | Derek Shepherd | 5/11/1983  | 17 Bellgrove Hill, Bakersfield, CA, 93311
   11 | Bria       | Schultes    | 559-834-7674 | bschultesa@webs.com           | OmxkjDepaUx  | Horatio Gauche | 8/20/1941  | 251 Shelley Road, Bakersfield, CA, 93312
   12 | Laurene    | Busain      | 284-353-0688 | lbusainb@nature.com           | iMBbqR0I     | Elloit Reed    | 3/29/1950  | 5354 West Hill, Bakersfield, CA, 93311
   13 | Llywellyn  | Patterfield | 916-234-9195 | lpatterfieldc@theatlantic.com | 6NN6E9       | Elloit Reed    | 2/3/1952   | 80042 Bartelt Parkway, Bakersfield, CA, 93311
   15 | Ruthy      | Downey      | 331-670-9738 | rdowneye@github.com           | KvHxCvZp     | John Dorian    | 9/22/1945  | 866 Lake View Place, Bakersfield, CA, 93312
   17 | Quentin    | McParlin    | 316-794-6927 | qmcparling@indiegogo.com      | XEnIV2PFl    | Derek Shepherd | 11/20/1997 | 7 Crest Line Junction, Bakersfield, CA, 93311
   18 | Jane       | Alp         | 263-801-1756 | jalph@senate.gov              | MqutSyQDq    | Abby Normal    | 3/24/1944  | 576 Morning Place, San Luis Obispo, CA 93407
   19 | Hans       | Winsom      | 309-148-2359 | hwinsomi@networksolutions.com | koDpjmiU     | Elloit Reed    | 4/5/1970   | 8 Eggendart Point, Bakersfield, CA, 93311
   20 | Stanleigh  | Jepp        | 942-292-1975 | sjeppj@ca.gov                 | kEHoDz       | Sandra Lee     | 7/5/1986   | 1115 Tomscot Court, Bakersfield, CA, 93311
   22 | Stuart     | Jinkin      | 820-175-7748 | sjinkinl@moonfruit.com        | VkGlRD       | Elloit Reed    | 11/13/1984 | 2158 Ryan Hill, Bakersfield, CA, 93314
   23 | Zea        | Danslow     | 373-181-7391 | zdanslowm@time.com            | Vzm8M7E      | John Dorian    | 9/15/1949  | 45 Oneill Trail, Bakersfield, CA, 93312
   24 | Borden     | Chillcot    | 272-895-3214 | bchillcotn@fastcompany.com    | rHyllHFY     | Abby Normal    | 11/25/1961 | 96 Del Mar Lane, San Luis Obispo, CA 93407
   25 | Rowena     | Cattanach   | 941-131-0527 | rcattanacho@github.com        | tjlwqzZqXMb  | Derek Shepherd | 10/31/1969 | 833 Summer Ridge Drive, Bakersfield, CA, 93312
   28 | Ilsa       | Lavelle     | 508-155-5218 | ilaveller@google.it           | 1l2fteUqeIPi | Abby Normal    | 10/30/1945 | 464 Londonderry Lane, Bakersfield, CA, 93312
   29 | Rossy      | Olyff       | 211-722-8977 | rolyffs@simplemachines.org    | ijJcUZ4YU    | Elloit Reed    | 1/10/1957  | 730 Dwight Place, Bakersfield, CA, 93314
   30 | Carl       | Hedau       | 691-201-5751 | fhedaut@house.gov             | SDGjbwJriVe  | John Dorian    | 2/11/1945  | 40440 Annamark Way, Bakersfield, CA, 93311
   31 | Billi      | Mulbery     | 303-341-5305 | bmulberyu@washington.edu      | Gy6ITqhkY4W  | Phillip McGraw | 6/26/1975  | 357 Dwight Drive, San Luis Obispo, CA 93407
   32 | Quentin    | Mackison    | 344-607-8136 | qmackisonv@slate.com          | K4RPNhp90j   | Derek Shepherd | 9/3/1972   | 95 Charing Cross Court, Bakersfield, CA, 93312
   34 | Lenard     | Gummie      | 892-424-6626 | lgummiex@php.net              | gGvhL2       | John Sins      | 8/31/1979  | 0353 Artisan Avenue, Bakersfield, CA, 93311
   35 | Tanhya     | Pittway     | 821-741-6469 | tpittwayy@oakley.com          | 7yCssfA8PjR  | Jan Pol        | 3/29/1957  | 1 Hauk Drive, Bakersfield, CA, 93314
   36 | Washington | Boundy      | 757-574-3689 | wboundyz@utexas.edu           | FCm4CTYAEwho | Jan Pol        | 3/23/1992  | 381 Roth Center, Bakersfield, CA, 93311
   37 | Aldrich    | Chiddy      | 501-216-7911 | achiddy10@home.pl             | MIyj1YGgPe   | John Cook      | 11/18/1999 | 6291 Corben Crossing, Bakersfield, CA, 93307
   38 | Claudia    | Iddenden    | 975-340-9058 | ciddenden11@baidu.com         | YTT5WP       | John Dorian    | 9/25/1966  | 74 Graceland Junction, Bakersfield, CA, 93312
   39 | Bartlet    | Drache      | 967-995-3395 | bdrache12@google.it           | NO8NVzPsM    | John Cook      | 3/27/1944  | 55666 5th Junction, Bakersfield, CA, 93311
(29 rows)
```

Phase 4: Programming Logic for SQL

4.1 - Introduction

Structured query language (SQL) was developed at IBM by Donald D. Chamberlin and Raymond F. Boyce in the early 1970s. It was originally called sequel but changed to SQL due to some trademark conflicts. In 1986, companies such as the American National standard institute (ANSI) and International Standards Organization (ISO) adopted the SQL language in relational database communication. SQL or sequel is a programming language that is used to manage the data in a database. Structured query language can access and manage the database with SQL statements. With the statements, you can add columns, retrieve information, modify data, update or dilute rows of data, and many more actions. The advantage that SQL has compared to other programs such as C++ and Java; SQL can create multiple views, it has well-defined standards, easy to learn, interactive language, and SQL queries are portable.

- A *view* is a logical subset of data contained in a table. The object is created with CREATE VIEW and acts as a virtual table. Views can simplify multiple tables into a single virtual table while hiding the complexity of data. The command is helpful when trying to secure sensitive information. A scenario where a view can be useful is when you are trying to give an engineer access to a table, but do not want to disclose sensitive information. A view will then be created to allow the engineer to access the data but not all.

- There are different types of functions: *scalar, inline table-valued, and multi-statement table-valued*. Scalar functions return a value of a particular data type and the function can have multiple statements. The inline table-valued function is somewhat similar to view, but the function can have an input into it. Unlike a scalar function, and Inline table-valued function can only return one single statement. Multi-statement table-valued functions allow multiple inputs to be passed similar to the scalar function, but multi-statement table-valued functions allow you to define the output table format.

- *Stored procedures* are similar to a function, however, to execute a stored procedure, an exec statement must be used. It forms a logical unit and is used to encapsulate a set of operations to execute on a database. A scenario for stored procedures is when you are trying to add a new customer or employee to the Customer or Employee table. By simply implementing an insert, the customer or employee will be easily added to the table without having to change any code.

- *Trigger* creation syntax is CREATE TRIGGER. There are three types of triggers: insert, update, delete, and they are all meant to respond automatically when something occurs on a table in the database. Triggers are used to maintain the integrity of the data. The user provides the trigger once and it is used when a specific action occurs. A scenario where a trigger can be used is when wanting to delete an employee from the Employee table. By creating the trigger, it will delete the employee and everything in its row since they are no longer associated with the company.

4.2 - Syntax of Programming Logic

Views:

CREATE VIEW view_name AS
SELECT task.name , task.description
FROM table_name
WHERE condition ;

Stored Functions:

CREATE FUNCTION function.name (arguments) RETURNS data.type
AS
BEGIN
DECLARE variable.name data.type ;
SELECT task.name , task.description

```
FROM table.name
WHERE condition ;
RETURN variable.name ;
END;
```

Stored Procedures:

```
DELIMITER  //
CREATE [OR REPLACE] PROCEDURE procedure_name
[(variable_name         IN|OUT          variable_type)]
AS
(DECLARE variables)
BEGIN
SQL statements
END //
DELIMITER;
```

Triggers:

```
DELIMITER  //
CREATE TRIGGER trigger.name
[before | after] {insert | update | delete} on [table_name] [for each row] [trigger_body]
END //
```

```
DELIMITER;
```

Stored Procedures and Functions:

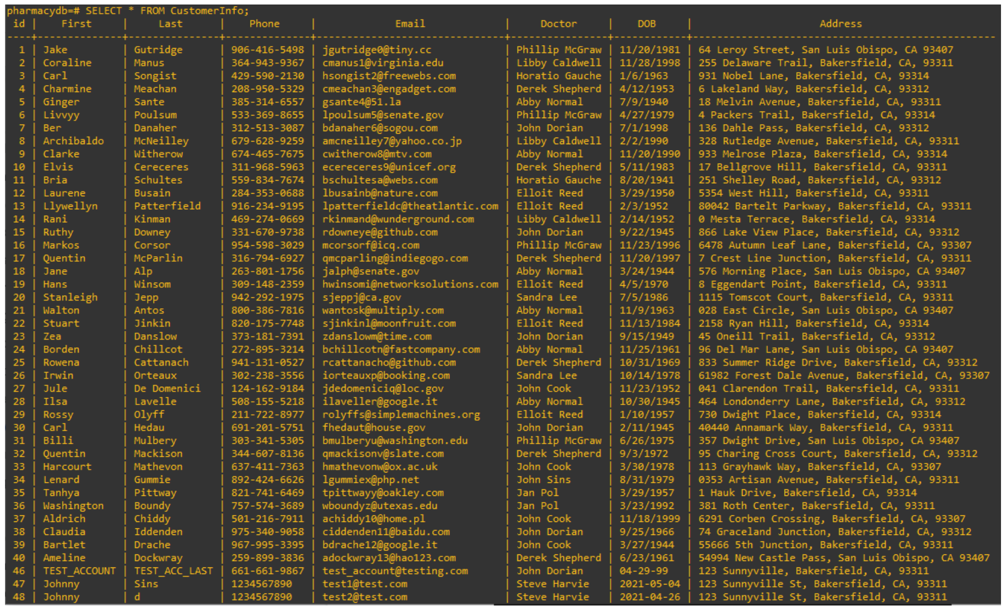
Similarities	Differences
● They both take arguments and use them to perform tasks. ● They both have the same programming style and structure.	● Stored Procedures may or may not return a value, while Functions must return a value. ● Functions can be called from Procedures, while Stored Procedures cannot be called from Functions. ● Stored Procedures can have both input and out parameters while function can only have an input. ● Functions can be called by using the “select command only, while Stored Procedures uses the “Exec / Execute” command. ● Stored Procedures can be created with and without parameters, while Functions can be created with parameters only. ● Functions allows only SELECT statements in it, while Stored Procedures allows SELECT and DML statements.

4.3 - Implementation

4.3.1 - Views

1. This view is to get data from the Customer table but not to grab passwords.

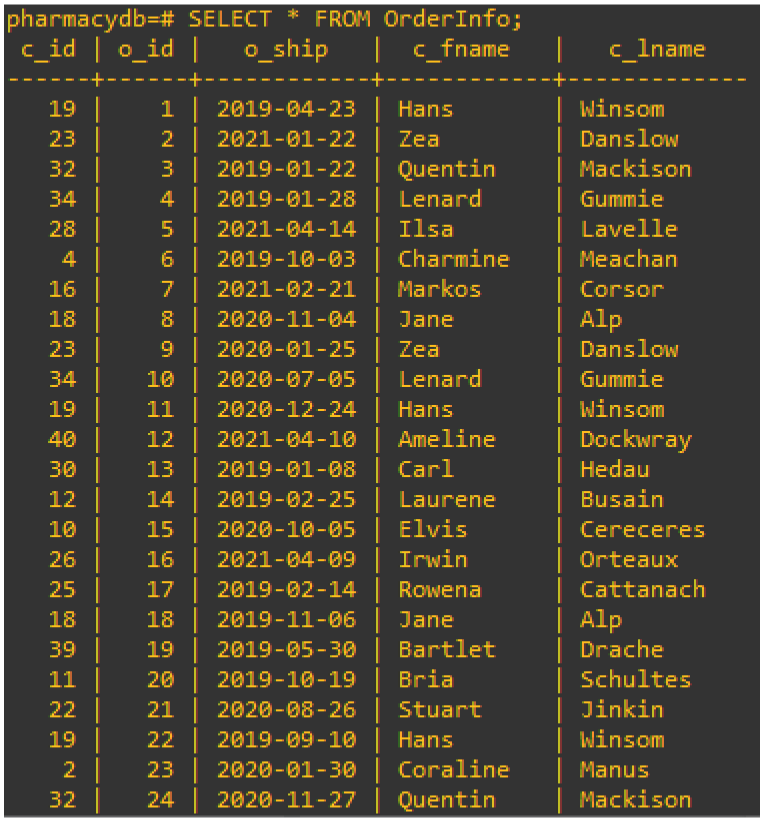
```
CREATE OR REPLACE VIEW CustomerInfo
AS SELECT c_id AS ID, c_fname AS "First", c_lname AS "Last",
c_phonenum AS "Phone", c_email AS "Email", c_doctor AS "Doctor",
c_dob AS "DOB", c_address AS "Address"
FROM Customer;
```



```
pharmacydb=# SELECT * FROM CustomerInfo;
 id |    First     |     Last      |    Phone     |             Email             |     Doctor     |    DOB     |                     Address
----+--------------+---------------+--------------+-------------------------------+----------------+------------+--------------------------------------------------
  1 | Jake         | Gutridge      | 906-416-5498 | jgutridge0@tiny.cc            | Phillip McGraw | 11/20/1981 | 64 Leroy Street, San Luis Obispo, CA 93407
  2 | Coraline     | Manus         | 364-943-9367 | cmanus1@virginia.edu          | Libby Caldwell | 11/28/1998 | 255 Delaware Trail, Bakersfield, CA, 93311
  3 | Carl         | Songist       | 429-590-2130 | hsongist2@freewebs.com        | Horatio Gauche | 1/6/1963   | 931 Nobel Lane, Bakersfield, CA, 93314
  4 | Charmine     | Meachan       | 208-950-5329 | cmeachan3@engadget.com        | Derek Shepherd | 4/12/1953  | 6 Lakeland Way, Bakersfield, CA, 93312
  5 | Ginger       | Sante         | 385-314-6557 | gsante4@51.la                 | Abby Normal    | 7/9/1940   | 18 Melvin Avenue, Bakersfield, CA, 93311
  6 | Livvyy       | Poulsum       | 533-369-8655 | lpoulsum5@senate.gov          | Phillip McGraw | 4/27/1979  | 4 Packers Trail, Bakersfield, CA, 93314
  7 | Ber          | Danaher       | 312-513-3087 | bdanaher6@sogou.com           | John Dorian    | 7/1/1998   | 136 Dahle Pass, Bakersfield, CA, 93312
  8 | Archibaldo   | McNeilley     | 679-628-9259 | amcneilley7@yahoo.co.jp       | Libby Caldwell | 2/2/1990   | 328 Rutledge Avenue, Bakersfield, CA, 93311
  9 | Clarke       | Witherow      | 674-465-7675 | cwitherow8@mtv.com            | Abby Normal    | 11/20/1990 | 933 Melrose Plaza, Bakersfield, CA, 93314
 10 | Elvis        | Cereceres     | 311-968-5963 | ecereceres9@unicef.org        | Derek Shepherd | 5/11/1983  | 17 Bellgrove Hill, Bakersfield, CA, 93311
 11 | Bria         | Schultes      | 559-834-7674 | bschultesa@webs.com           | Horatio Gauche | 8/20/1941  | 251 Shelley Road, Bakersfield, CA, 93312
 12 | Laurene      | Busain        | 284-353-0688 | lbusainb@nature.com           | Elloit Reed    | 3/29/1950  | 5354 West Hill, Bakersfield, CA, 93311
 13 | Llywellyn    | Patterfield   | 916-234-9195 | lpatterfieldc@theatlantic.com | Elloit Reed    | 2/3/1952   | 80042 Bartelt Parkway, Bakersfield, CA, 93311
 14 | Rani         | Kinman        | 469-274-0669 | rkinmand@wunderground.com     | Libby Caldwell | 2/14/1952  | 0 Mesta Terrace, Bakersfield, CA, 93314
 15 | Ruthy        | Downey        | 331-670-9738 | rdowneye@github.com           | John Dorian    | 9/22/1945  | 866 Lake View Place, Bakersfield, CA, 93312
 16 | Markos       | Corsor        | 954-598-3029 | mcorsorf@icq.com              | Phillip McGraw | 11/23/1996 | 6478 Autumn Leaf Lane, Bakersfield, CA, 93307
 17 | Quentin      | McParlin      | 316-794-6927 | qmcparling@indiegogo.com      | Derek Shepherd | 11/20/1997 | 7 Crest Line Junction, Bakersfield, CA, 93311
 18 | Jane         | Alp           | 263-801-1756 | jalph@senate.gov              | Abby Normal    | 3/24/1944  | 576 Morning Place, San Luis Obispo, CA 93407
 19 | Hans         | Winsom        | 309-148-2359 | hwinsomi@networksolutions.com | Elloit Reed    | 4/5/1970   | 8 Eggendart Point, Bakersfield, CA, 93311
 20 | Stanleigh    | Jepp          | 942-292-1975 | sjeppj@ca.gov                 | Sandra Lee     | 7/5/1986   | 1115 Tomscot Court, Bakersfield, CA, 93311
 21 | Walton       | Antos         | 800-386-7816 | wantosk@multiply.com          | Abby Normal    | 11/9/1963  | 028 East Circle, San Luis Obispo, CA 93407
 22 | Stuart       | Jinkin        | 820-175-7748 | sjinkinl@moonfruit.com        | Elloit Reed    | 11/13/1984 | 2158 Ryan Hill, Bakersfield, CA, 93314
 23 | Zea          | Danslow       | 373-181-7391 | zdanslowm@time.com            | John Dorian    | 9/15/1949  | 45 Oneill Trail, Bakersfield, CA, 93312
 24 | Borden       | Chillcot      | 272-895-3214 | bchillcotn@fastcompany.com    | Abby Normal    | 11/25/1961 | 96 Del Mar Lane, San Luis Obispo, CA 93407
 25 | Rowena       | Cattanach     | 941-131-0527 | rcattanacho@github.com        | Derek Shepherd | 10/31/1969 | 833 Summer Ridge Drive, Bakersfield, CA, 93312
 26 | Irwin        | Orteaux       | 302-238-3556 | iorteauxp@booking.com         | Sandra Lee     | 10/14/1978 | 61982 Forest Dale Avenue, Bakersfield, CA, 93307
 27 | Jule         | De Domenici   | 124-162-9184 | jdedomeniciq@loc.gov          | John Cook      | 11/23/1952 | 041 Clarendon Trail, Bakersfield, CA, 93311
 28 | Ilsa         | Lavelle       | 508-155-5218 | ilaveller@google.it           | Abby Normal    | 10/30/1945 | 464 Londonderry Lane, Bakersfield, CA, 93312
 29 | Rossy        | Olyff         | 211-722-8977 | rolyffs@simplemachines.org    | Elloit Reed    | 1/10/1957  | 730 Dwight Place, Bakersfield, CA, 93314
 30 | Carl         | Hedau         | 691-201-5751 | fhedaut@house.gov             | John Dorian    | 2/11/1945  | 40440 Annamark Way, Bakersfield, CA, 93311
 31 | Billi        | Mulbery       | 303-341-5305 | bmulberyu@washington.edu      | Phillip McGraw | 6/26/1975  | 357 Dwight Drive, San Luis Obispo, CA 93407
 32 | Quentin      | Mackison      | 344-607-8136 | qmackisonv@slate.com          | Derek Shepherd | 9/3/1972   | 95 Charing Cross Court, Bakersfield, CA, 93312
 33 | Harcourt     | Mathevon      | 637-411-7363 | hmathevonw@ox.ac.uk           | John Cook      | 3/30/1978  | 113 Grayhawk Way, Bakersfield, CA, 93307
 34 | Lenard       | Gummie        | 892-424-6626 | lgummiex@php.net              | John Sins      | 8/31/1979  | 0353 Artisan Avenue, Bakersfield, CA, 93311
 35 | Tanhya       | Pittway       | 821-741-6469 | tpittwayy@oakley.com          | Jan Pol        | 3/29/1957  | 1 Hauk Drive, Bakersfield, CA, 93314
 36 | Washington   | Boundy        | 757-574-3689 | wboundyz@utexas.edu           | Jan Pol        | 3/23/1992  | 381 Roth Center, Bakersfield, CA, 93311
 37 | Aldrich      | Chiddy        | 501-216-7911 | achiddy10@home.pl             | John Cook      | 11/18/1999 | 6291 Corben Crossing, Bakersfield, CA, 93307
 38 | Claudia      | Iddenden      | 975-340-9058 | ciddenden11@baidu.com         | John Dorian    | 9/25/1966  | 74 Graceland Junction, Bakersfield, CA, 93312
 39 | Bartlet      | Drache        | 967-995-3395 | bdrache12@google.it           | John Cook      | 3/27/1944  | 55666 5th Junction, Bakersfield, CA, 93311
 40 | Ameline      | Dockwray      | 259-899-3836 | adockwray13@hao123.com        | Derek Shepherd | 6/23/1961  | 54994 New Castle Pass, San Luis Obispo, CA 93407
 46 | TEST_ACCOUNT | TEST_ACC_LAST | 661-661-9867 | test_account@testing.com      | John Dorian    | 04-29-99   | 123 Sunnyville, Bakersfield, CA, 93311
 47 | Johnny       | Sins          | 1234567890   | test1@test.com                | Steve Harvie   | 2021-05-04 | 123 Sunnyville St, Bakersfield, CA, 93311
 48 | Johnny       | d             | 1234567890   | test2@test.com                | Steve Harvie   | 2021-04-26 | 123 Sunnyville St, Bakersfield, CA, 93311
```

2. This view combines the Orders table and the Customers table. Here this gives a view with the customers name and the order information.

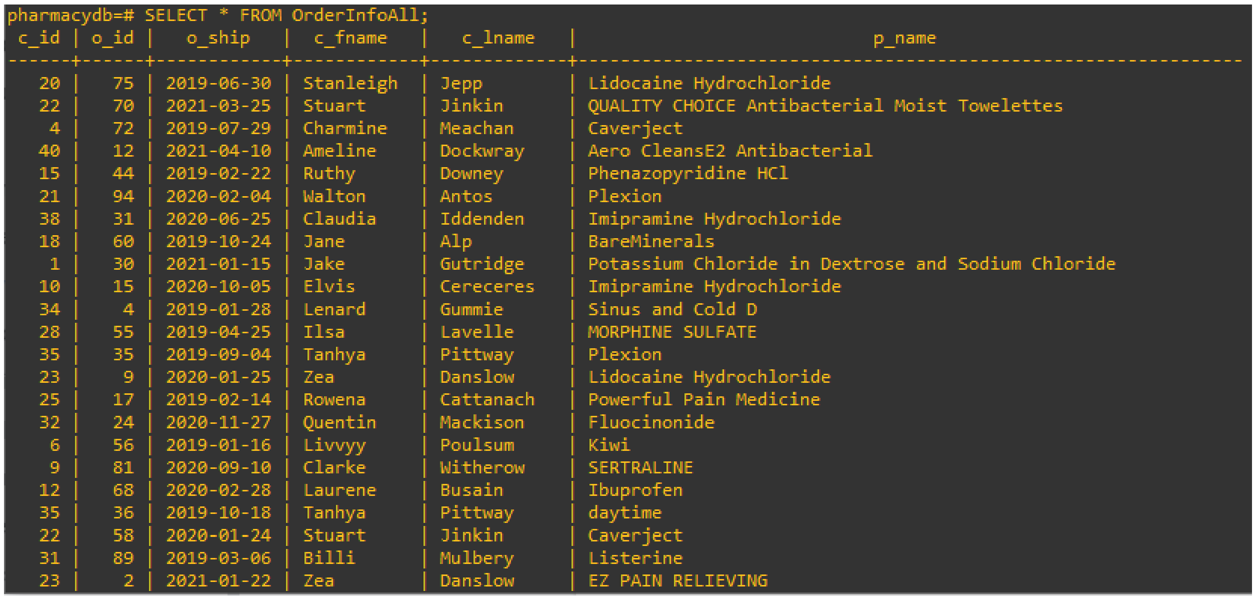
```
CREATE OR REPLACE VIEW OrderInfo
AS SELECT Orders.c_id AS C_ID, Orders.o_id AS O_ID, Orders.o_shipdate AS O_SHIP,
Customer.c_fname AS C_FNAME, Customer.c_lname As C_LNAME
FROM Orders, Customer
WHERE Orders.c_id = Customer.c_id;
```



```
pharmacydb=# SELECT * FROM OrderInfo;
 c_id | o_id |   o_ship   |  c_fname   |   c_lname
------+------+------------+------------+-------------
   19 |    1 | 2019-04-23 | Hans       | Winsom
   23 |    2 | 2021-01-22 | Zea        | Danslow
   32 |    3 | 2019-01-22 | Quentin    | Mackison
   34 |    4 | 2019-01-28 | Lenard     | Gummie
   28 |    5 | 2021-04-14 | Ilsa       | Lavelle
    4 |    6 | 2019-10-03 | Charmine   | Meachan
   16 |    7 | 2021-02-21 | Markos     | Corsor
   18 |    8 | 2020-11-04 | Jane       | Alp
   23 |    9 | 2020-01-25 | Zea        | Danslow
   34 |   10 | 2020-07-05 | Lenard     | Gummie
   19 |   11 | 2020-12-24 | Hans       | Winsom
   40 |   12 | 2021-04-10 | Ameline    | Dockwray
   30 |   13 | 2019-01-08 | Carl       | Hedau
   12 |   14 | 2019-02-25 | Laurene    | Busain
   10 |   15 | 2020-10-05 | Elvis      | Cereceres
   26 |   16 | 2021-04-09 | Irwin      | Orteaux
   25 |   17 | 2019-02-14 | Rowena     | Cattanach
   18 |   18 | 2019-11-06 | Jane       | Alp
   39 |   19 | 2019-05-30 | Bartlet    | Drache
   11 |   20 | 2019-10-19 | Bria       | Schultes
   22 |   21 | 2020-08-26 | Stuart     | Jinkin
   19 |   22 | 2019-09-10 | Hans       | Winsom
    2 |   23 | 2020-01-30 | Coraline   | Manus
   32 |   24 | 2020-11-27 | Quentin    | Mackison
```

3. This view combines the Customer, Orders, and Product table to output order info that has a customer's name, the customer's ID, the order ID, the order ship date, and the product name.

```
CREATE OR REPLACE VIEW OrderInfoAll
AS SELECT Orders.c_id AS C_ID, Orders.o_id AS O_ID, Orders.o_shipdate AS O_SHIP,
Customer.c_fname AS C_FNAME, Customer.c_lname As C_LNAME, Product.p_name
FROM Orders, Customer, Product, Contains
WHERE Orders.c_id = Customer.c_id
   AND Contains.o_id = Orders.o_id
   AND Contains.p_id = Product.p_id;
```



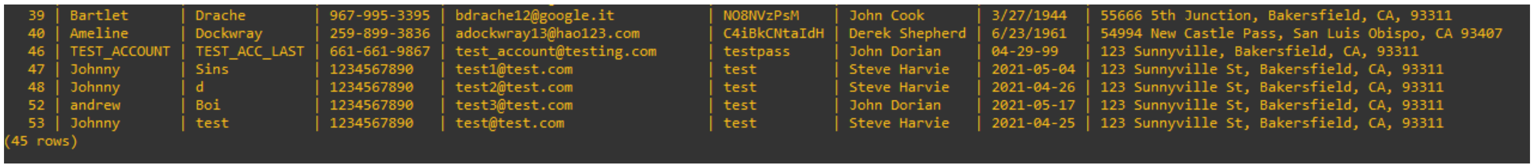
```
pharmacydb=# SELECT * FROM OrderInfoAll;
 c_id | o_id |   o_ship   |  c_fname   |   c_lname   |                         p_name
------+------+------------+------------+-------------+-----------------------------------------------------------------
   20 |   75 | 2019-06-30 | Stanleigh  | Jepp        | Lidocaine Hydrochloride
   22 |   70 | 2021-03-25 | Stuart     | Jinkin      | QUALITY CHOICE Antibacterial Moist Towelettes
    4 |   72 | 2019-07-29 | Charmine   | Meachan     | Caverject
   40 |   12 | 2021-04-10 | Ameline    | Dockwray    | Aero CleansE2 Antibacterial
   15 |   44 | 2019-02-22 | Ruthy      | Downey      | Phenazopyridine HCl
   21 |   94 | 2020-02-04 | Walton     | Antos       | Plexion
   38 |   31 | 2020-06-25 | Claudia    | Iddenden    | Imipramine Hydrochloride
   18 |   60 | 2019-10-24 | Jane       | Alp         | BareMinerals
    1 |   30 | 2021-01-15 | Jake       | Gutridge    | Potassium Chloride in Dextrose and Sodium Chloride
   10 |   15 | 2020-10-05 | Elvis      | Cereceres   | Imipramine Hydrochloride
   34 |    4 | 2019-01-28 | Lenard     | Gummie      | Sinus and Cold D
   28 |   55 | 2019-04-25 | Ilsa       | Lavelle     | MORPHINE SULFATE
   35 |   35 | 2019-09-04 | Tanhya     | Pittway     | Plexion
   23 |    9 | 2020-01-25 | Zea        | Danslow     | Lidocaine Hydrochloride
   25 |   17 | 2019-02-14 | Rowena     | Cattanach   | Powerful Pain Medicine
   32 |   24 | 2020-11-27 | Quentin    | Mackison    | Fluocinonide
    6 |   56 | 2019-01-16 | Livvyy     | Poulsum     | Kiwi
    9 |   81 | 2020-09-10 | Clarke     | Witherow    | SERTRALINE
   12 |   68 | 2020-02-28 | Laurene    | Busain      | Ibuprofen
   35 |   36 | 2019-10-18 | Tanhya     | Pittway     | daytime
   22 |   58 | 2020-01-24 | Stuart     | Jinkin      | Caverject
   31 |   89 | 2019-03-06 | Billi      | Mulbery     | Listerine
   23 |    2 | 2021-01-22 | Zea        | Danslow     | EZ PAIN RELIEVING
```

4.3.2 - Stored procedures/functions

1. This function will add a new row to the Customer table and this creates a new customer to the database.

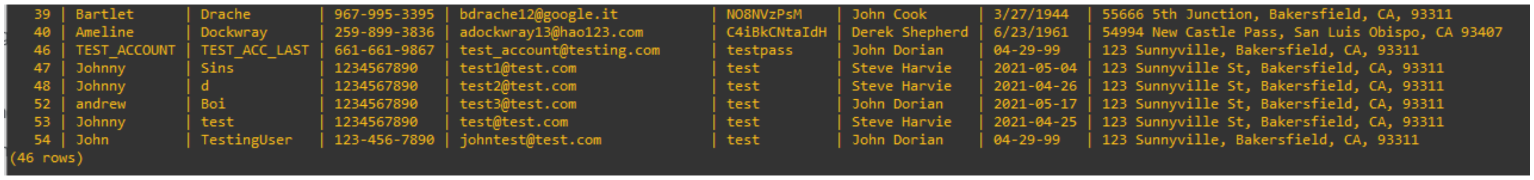
```
CREATE OR REPLACE FUNCTION RegisterCustomer(
    fname character varying(20),
    lname character varying(20),
    phoneNum character varying(15),
    email character varying(50),
    cpassword character varying(32),
    doctor character varying(50),
    dob character varying(10),
    caddress character varying(50)
)

RETURNS VOID AS
    $BODY$
        DECLARE
            useremailCount integer;
        BEGIN
            SELECT COUNT(*) INTO useremailCount
            From Customer
            WHERE c_email = email;
            IF useremailCount = 0 THEN
                INSERT INTO Customer (c_id, c_fname, c_lname, c_phonenum, c_email, c_password,
c_doctor, c_dob, c_address) VALUES (DEFAULT, fname, lname, phoneNum, email, cpassword,
doctor, dob, caddress);
            END IF;
        END;
    $BODY$ LANGUAGE plpgsql;
```

Before Addition:



```
 39 | Bartlet      | Drache        | 967-995-3395 | bdrache12@google.it          | NO8NVzPsM   | John Cook      | 3/27/1944  | 55666 5th Junction, Bakersfield, CA, 93311
 40 | Ameline      | Dockwray      | 259-899-3836 | adockwray13@hao123.com       | C4iBkCNtaIdH| Derek Shepherd | 6/23/1961  | 54994 New Castle Pass, San Luis Obispo, CA 93407
 46 | TEST_ACCOUNT | TEST_ACC_LAST | 661-661-9867 | test_account@testing.com     | testpass    | John Dorian    | 04-29-99   | 123 Sunnyville, Bakersfield, CA, 93311
 47 | Johnny       | Sins          | 1234567890   | test1@test.com               | test        | Steve Harvie   | 2021-05-04 | 123 Sunnyville St, Bakersfield, CA, 93311
 48 | Johnny       | d             | 1234567890   | test2@test.com               | test        | Steve Harvie   | 2021-04-26 | 123 Sunnyville St, Bakersfield, CA, 93311
 52 | andrew       | Boi           | 1234567890   | test3@test.com               | test        | John Dorian    | 2021-05-17 | 123 Sunnyville St, Bakersfield, CA, 93311
 53 | Johnny       | test          | 1234567890   | test@test.com                | test        | Steve Harvie   | 2021-04-25 | 123 Sunnyville St, Bakersfield, CA, 93311
(45 rows)
```

After Addition:

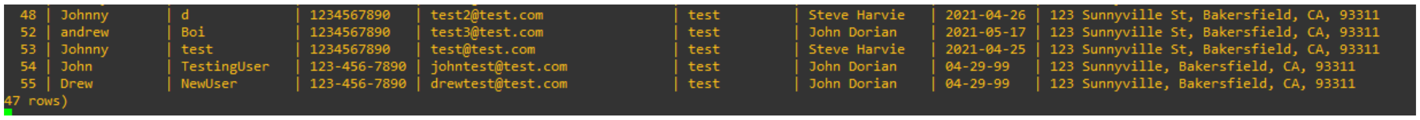


```
 39 | Bartlet      | Drache        | 967-995-3395 | bdrache12@google.it          | NO8NVzPsM   | John Cook      | 3/27/1944  | 55666 5th Junction, Bakersfield, CA, 93311
 40 | Ameline      | Dockwray      | 259-899-3836 | adockwray13@hao123.com       | C4iBkCNtaIdH| Derek Shepherd | 6/23/1961  | 54994 New Castle Pass, San Luis Obispo, CA 93407
 46 | TEST_ACCOUNT | TEST_ACC_LAST | 661-661-9867 | test_account@testing.com     | testpass    | John Dorian    | 04-29-99   | 123 Sunnyville, Bakersfield, CA, 93311
 47 | Johnny       | Sins          | 1234567890   | test1@test.com               | test        | Steve Harvie   | 2021-05-04 | 123 Sunnyville St, Bakersfield, CA, 93311
 48 | Johnny       | d             | 1234567890   | test2@test.com               | test        | Steve Harvie   | 2021-04-26 | 123 Sunnyville St, Bakersfield, CA, 93311
 52 | andrew       | Boi           | 1234567890   | test3@test.com               | test        | John Dorian    | 2021-05-17 | 123 Sunnyville St, Bakersfield, CA, 93311
 53 | Johnny       | test          | 1234567890   | test@test.com                | test        | Steve Harvie   | 2021-04-25 | 123 Sunnyville St, Bakersfield, CA, 93311
 54 | John         | TestingUser   | 123-456-7890 | johntest@test.com            | test        | John Dorian    | 04-29-99   | 123 Sunnyville, Bakersfield, CA, 93311
(46 rows)
```

2. This function will delete a customer from the customer table when called.

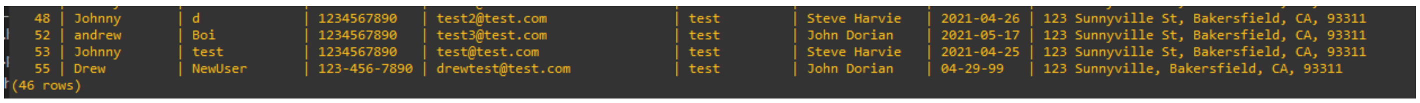
```
CREATE OR REPLACE FUNCTION deleteCustomerTest(
    email character varying(256)
)
RETURNS VOID AS
    $BODY$
        BEGIN
            DELETE FROM Customer
            WHERE c_email = email;
        END;
    $BODY$ LANGUAGE plpgsql;
```

Before Deletion:



```
  48 | Johnny       | d            | 1234567890   | test2@test.com          | test      | Steve Harvie  | 2021-04-26 | 123 Sunnyville St, Bakersfield, CA, 93311
  52 | andrew       | Boi          | 1234567890   | test3@test.com          | test      | John Dorian   | 2021-05-17 | 123 Sunnyville St, Bakersfield, CA, 93311
  53 | Johnny       | test         | 1234567890   | test@test.com           | test      | Steve Harvie  | 2021-04-25 | 123 Sunnyville St, Bakersfield, CA, 93311
  54 | John         | TestingUser  | 123-456-7890 | johntest@test.com       | test      | John Dorian   | 04-29-99   | 123 Sunnyville, Bakersfield, CA, 93311
  55 | Drew         | NewUser      | 123-456-7890 | drewtest@test.com       | test      | John Dorian   | 04-29-99   | 123 Sunnyville, Bakersfield, CA, 93311
47 rows)
```

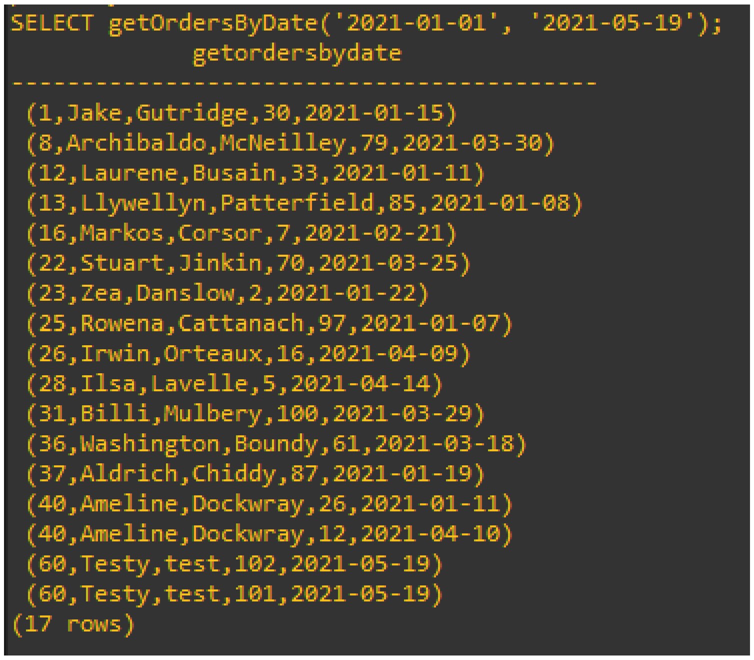
After Deletion:



```
  48 | Johnny       | d            | 1234567890   | test2@test.com          | test      | Steve Harvie  | 2021-04-26 | 123 Sunnyville St, Bakersfield, CA, 93311
  52 | andrew       | Boi          | 1234567890   | test3@test.com          | test      | John Dorian   | 2021-05-17 | 123 Sunnyville St, Bakersfield, CA, 93311
  53 | Johnny       | test         | 1234567890   | test@test.com           | test      | Steve Harvie  | 2021-04-25 | 123 Sunnyville St, Bakersfield, CA, 93311
  55 | Drew         | NewUser      | 123-456-7890 | drewtest@test.com       | test      | John Dorian   | 04-29-99   | 123 Sunnyville, Bakersfield, CA, 93311
(46 rows)
```

3. This function returns orders between two dates. The function takes in two variables, the first being the start date and the second being the end and will return all orders between the two dates.

```
CREATE OR REPLACE FUNCTION getOrdersByDate(
    datefrom date,
    dateto date
)

RETURNS TABLE (
    c_id integer,
    c_fname character varying(256),
    c_lname character varying(256),
    o_id integer,
    o_shipdate date
)
AS  $BODY$
            SELECT Customer.c_id, Customer.c_fname, Customer.c_lname, Orders.o_id,
Orders.o_shipdate
            FROM Orders INNER JOIN Customer
                ON Orders.c_id = Customer.c_id
            WHERE Orders.o_shipdate >= datefrom
                AND Orders.o_shipdate <= dateto;
    $BODY$ LANGUAGE sql;
```



```
SELECT getOrdersByDate('2021-01-01', '2021-05-19');
             getordersbydate
------------------------------------------
 (1,Jake,Gutridge,30,2021-01-15)
 (8,Archibaldo,McNeilley,79,2021-03-30)
 (12,Laurene,Busain,33,2021-01-11)
 (13,Llywellyn,Patterfield,85,2021-01-08)
 (16,Markos,Corsor,7,2021-02-21)
 (22,Stuart,Jinkin,70,2021-03-25)
 (23,Zea,Danslow,2,2021-01-22)
 (25,Rowena,Cattanach,97,2021-01-07)
 (26,Irwin,Orteaux,16,2021-04-09)
 (28,Ilsa,Lavelle,5,2021-04-14)
 (31,Billi,Mulbery,100,2021-03-29)
 (36,Washington,Boundy,61,2021-03-18)
 (37,Aldrich,Chiddy,87,2021-01-19)
 (40,Ameline,Dockwray,26,2021-01-11)
 (40,Ameline,Dockwray,12,2021-04-10)
 (60,Testy,test,102,2021-05-19)
 (60,Testy,test,101,2021-05-19)
(17 rows)
```

4.3.3 - Triggers

Deleting a row from a table

```
CREATE FUNCTION deleteOrders() RETURNS TRIGGER AS $_$
BEGIN
   DELETE FROM Orders
   WHERE NOT EXISTS (
      SELECT *
      FROM Customer
      WHERE Orders.c_id = Customer.c_id
   );
   RETURN NULL;
END;
$_$ LANGUAGE plpgsql;
CREATE TRIGGER customer_delete
AFTER DELETE ON customer
FOR EACH ROW
EXECUTE PROCEDURE deleteOrders();
```

Before:

After:

Updating a row in a table

```
DELEMITTER //
CREATE TRIGGER update_productquantitiy
        BEFORE UPDATE
        ON Product FROM ROW
BEGIN
SET new.product_info  = CURDATE();
END //
DELIMITER;
```

Before:

After:

Inserting a row into a table.

```
DELIMITER //
CREATE TRIGGER ----
AFTER INSERT
ON --- FOR EACH ROW
        Begin
                DECLARE lid INT;
                Select Max(---)+1 INTO lid FROM ---;
                INSERT INTO -- (---)
VALUES (---);
END //
DELIMITER ;
```

Before:

After:

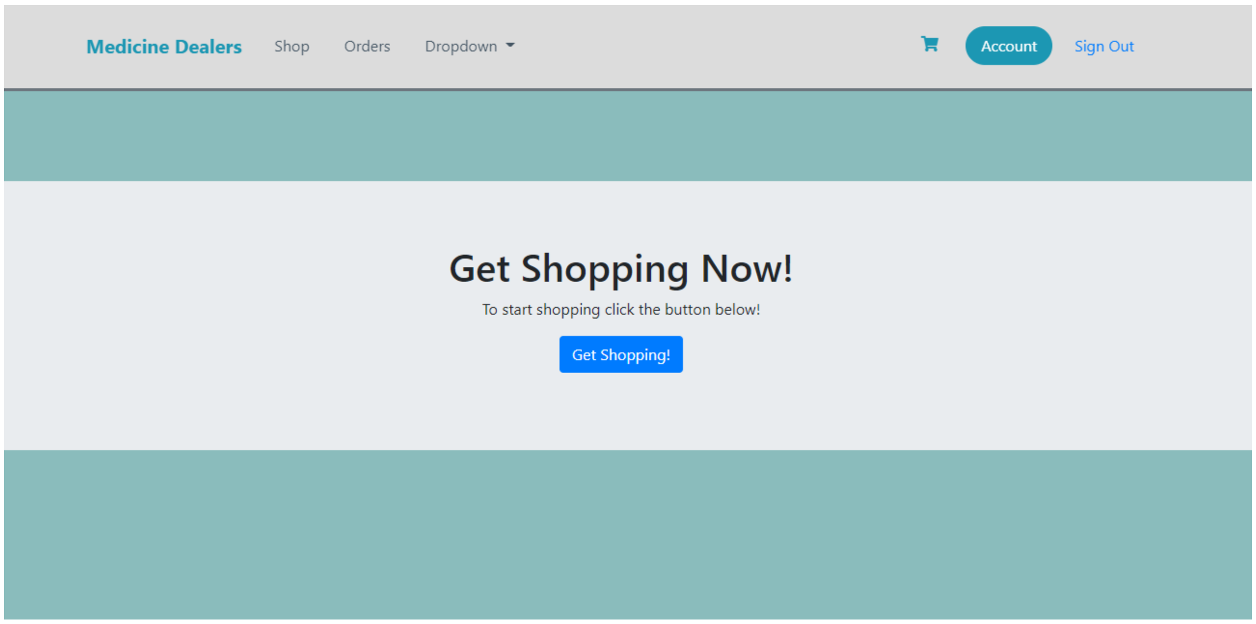
Phase 5: GUI Development

5.1 - GUI Functionalities and User Groups

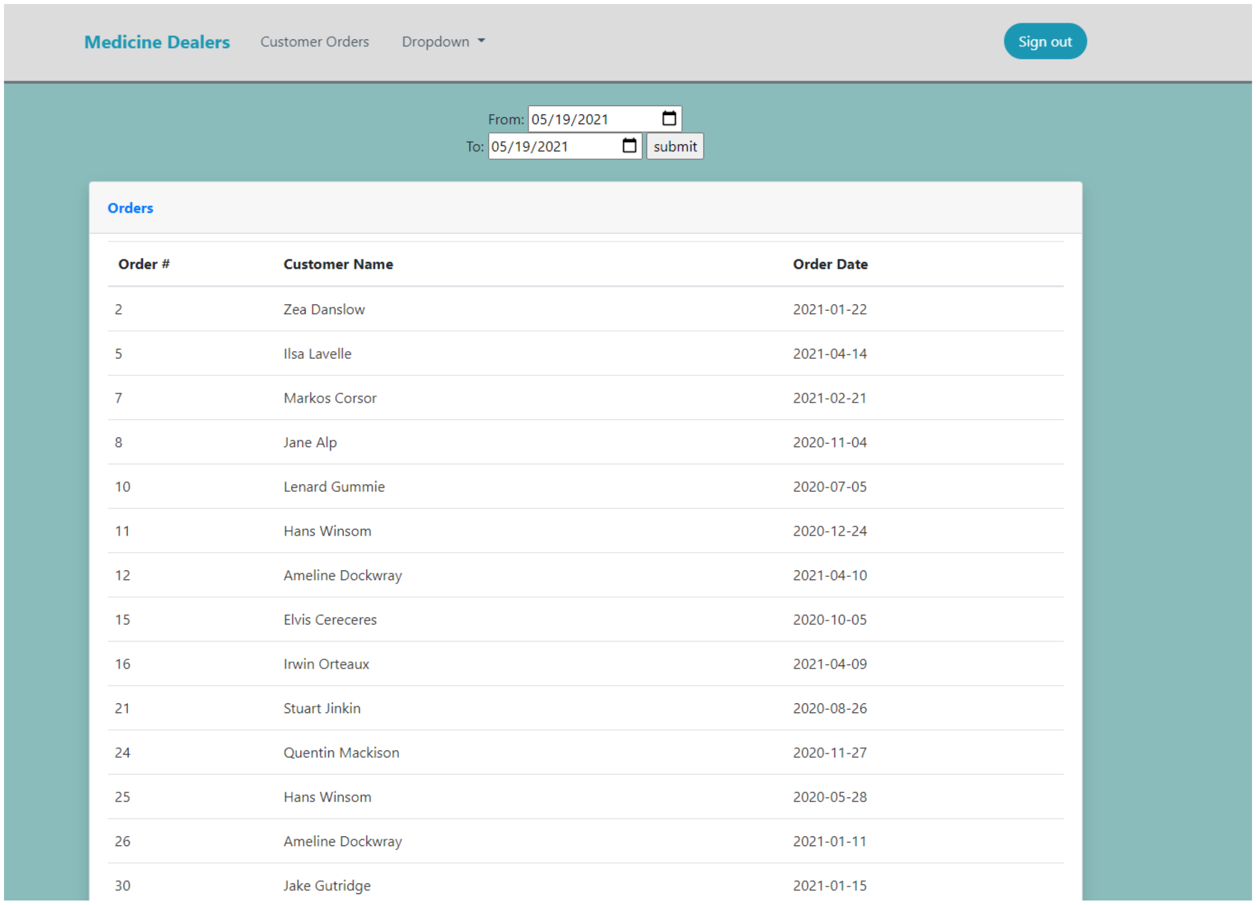
There are two user groups. A customer group which can purchase products and an employee group that can check orders and sales. The customer will be able to view the products, create an account, delete an account, and purchase medicine. The employees can check orders that were made by customers, check the customers information and cancel orders.

You can find the website here: https://andrewmccuan.tech/frontend.

5.1.1 - Itemized Descriptions of the GUI



The GUI from the customer side looks like this. It will prompt them to shop but first make them make an account and/or log in. Once logged in they can view the products and also update their own account.



From: 05/19/2021
To: 05/19/2021 submit

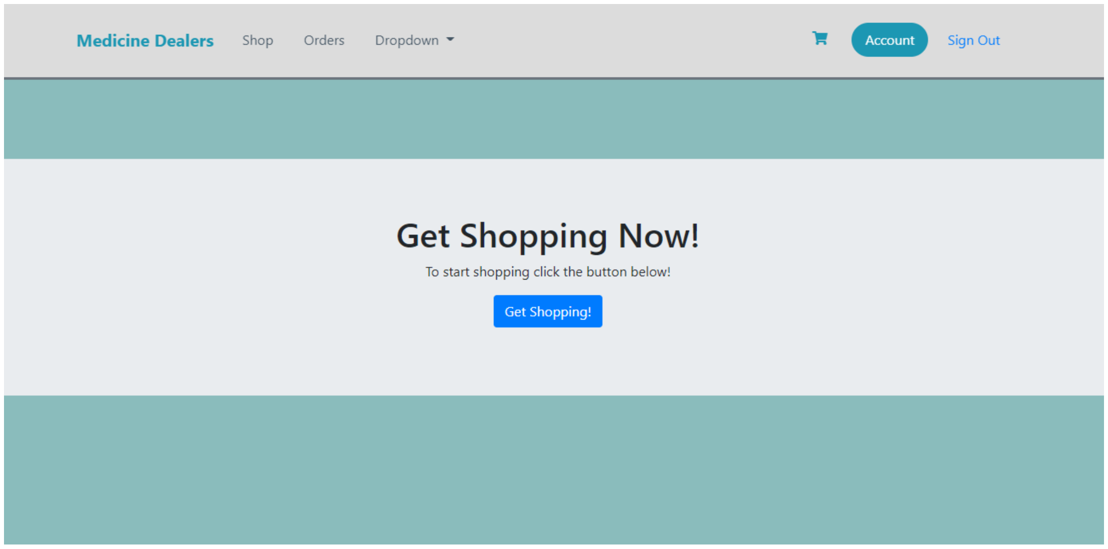
Orders

Order #	Customer Name	Order Date
2	Zea Danslow	2021-01-22
5	Ilsa Lavelle	2021-04-14
7	Markos Corsor	2021-02-21
8	Jane Alp	2020-11-04
10	Lenard Gummie	2020-07-05
11	Hans Winsom	2020-12-24
12	Ameline Dockwray	2021-04-10
15	Elvis Cereceres	2020-10-05
16	Irwin Orteaux	2021-04-09
21	Stuart Jinkin	2020-08-26
24	Quentin Mackison	2020-11-27
25	Hans Winsom	2020-05-28
26	Ameline Dockwray	2021-01-11
30	Jake Gutridge	2021-01-15

The GUI for the employee side will allow them to search orders from a certain time period. It will show the order number, order date, and customer name.

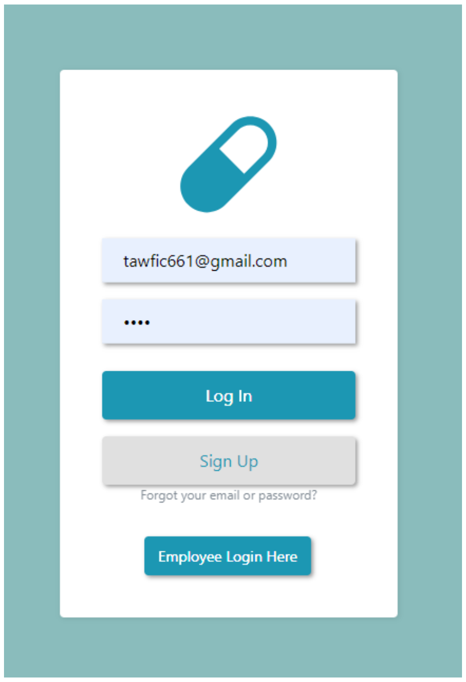
5.1.2 - Screenshots and Walkthrough

The customer will be greeted by the frontpage where it will tell them to sign in or register their account. After that they can shop through the products and add items to the cart. From the cart you can submit the purchase and it will show on the customers order page, all the orders they have done.



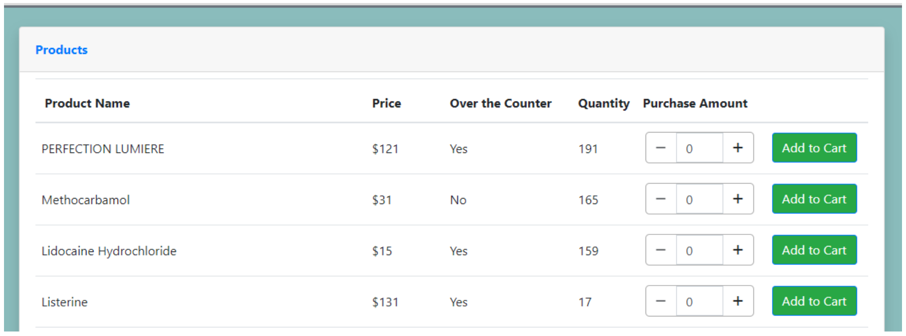
Get Shopping Now!

To start shopping click the button below!



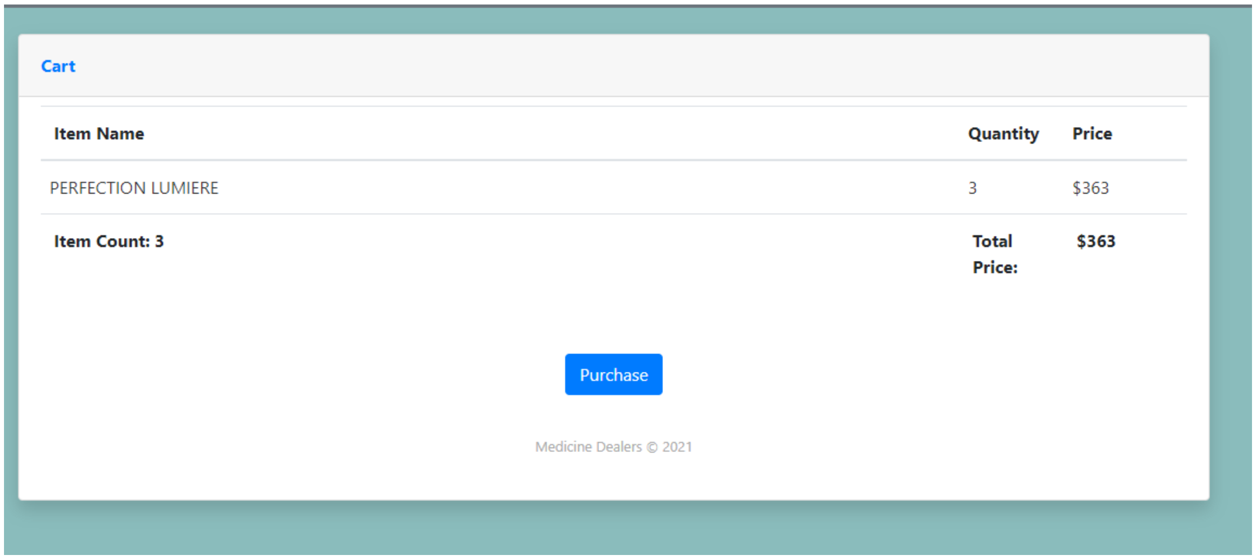
tawfic661@gmail.com

••••



Products

Product Name	Price	Over the Counter	Quantity	Purchase Amount	
PERFECTION LUMIERE	$121	Yes	191	− 0 +	Add to Cart
Methocarbamol	$31	No	165	− 0 +	Add to Cart
Lidocaine Hydrochloride	$15	Yes	159	− 0 +	Add to Cart
Listerine	$131	Yes	17	− 0 +	Add to Cart



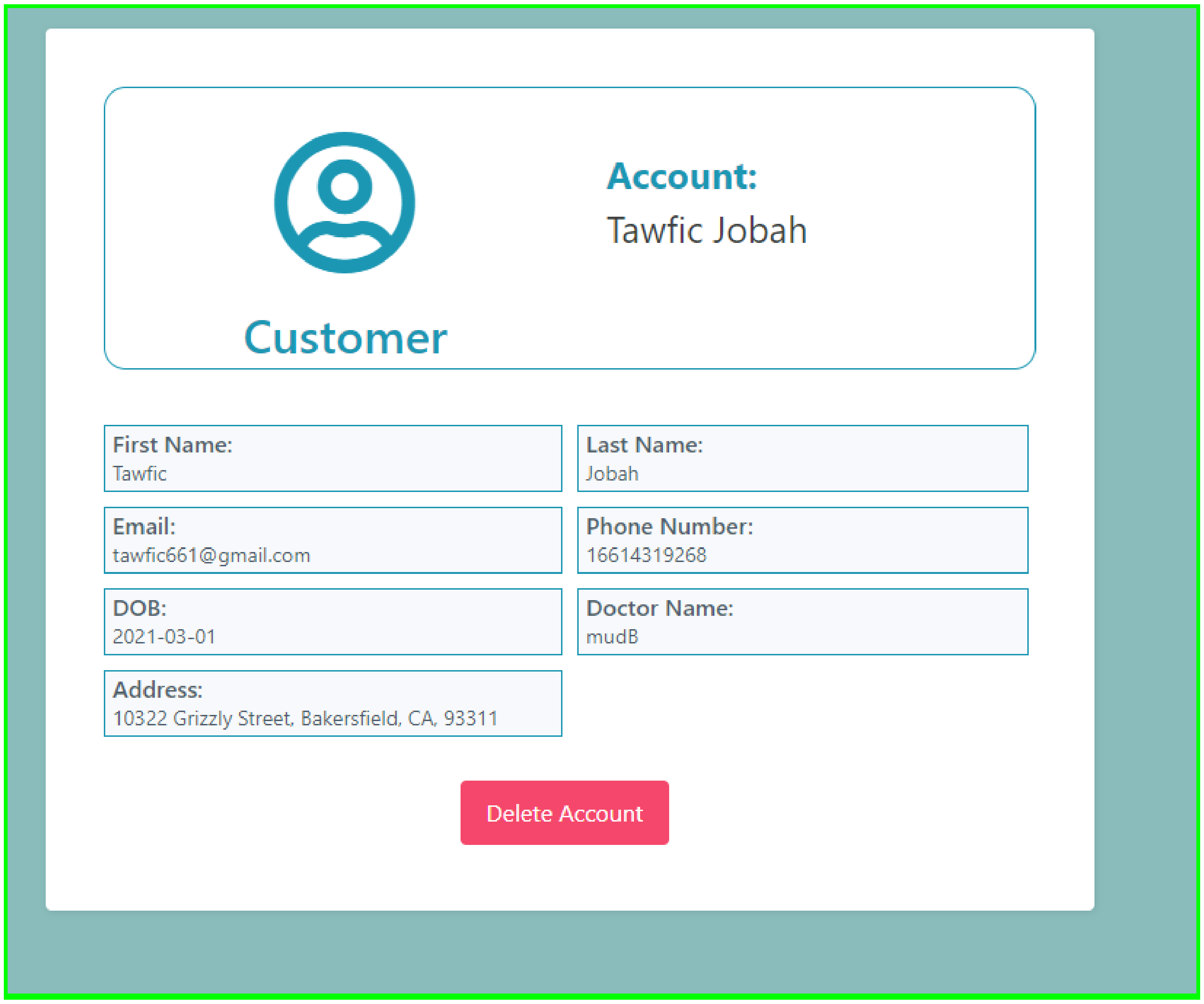
Cart

Item Name	Quantity	Price
PERFECTION LUMIERE	3	$363
Item Count: 3	**Total Price:**	**$363**

Medicine Dealers © 2021

5.1.3 - Demonstration of Programming Logic

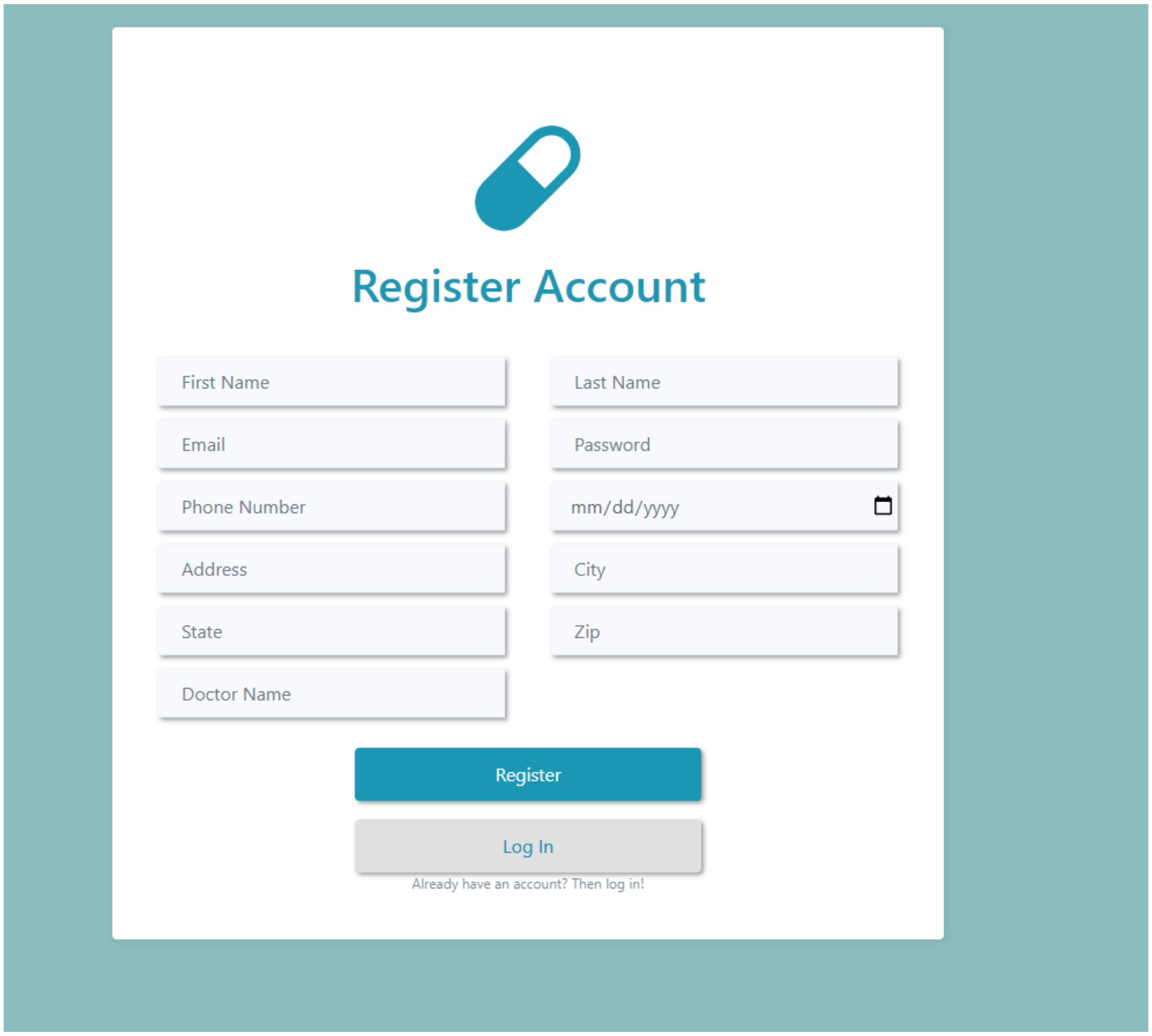
When a customer logs in they can delete their own account. We used a function that runs on the button click.



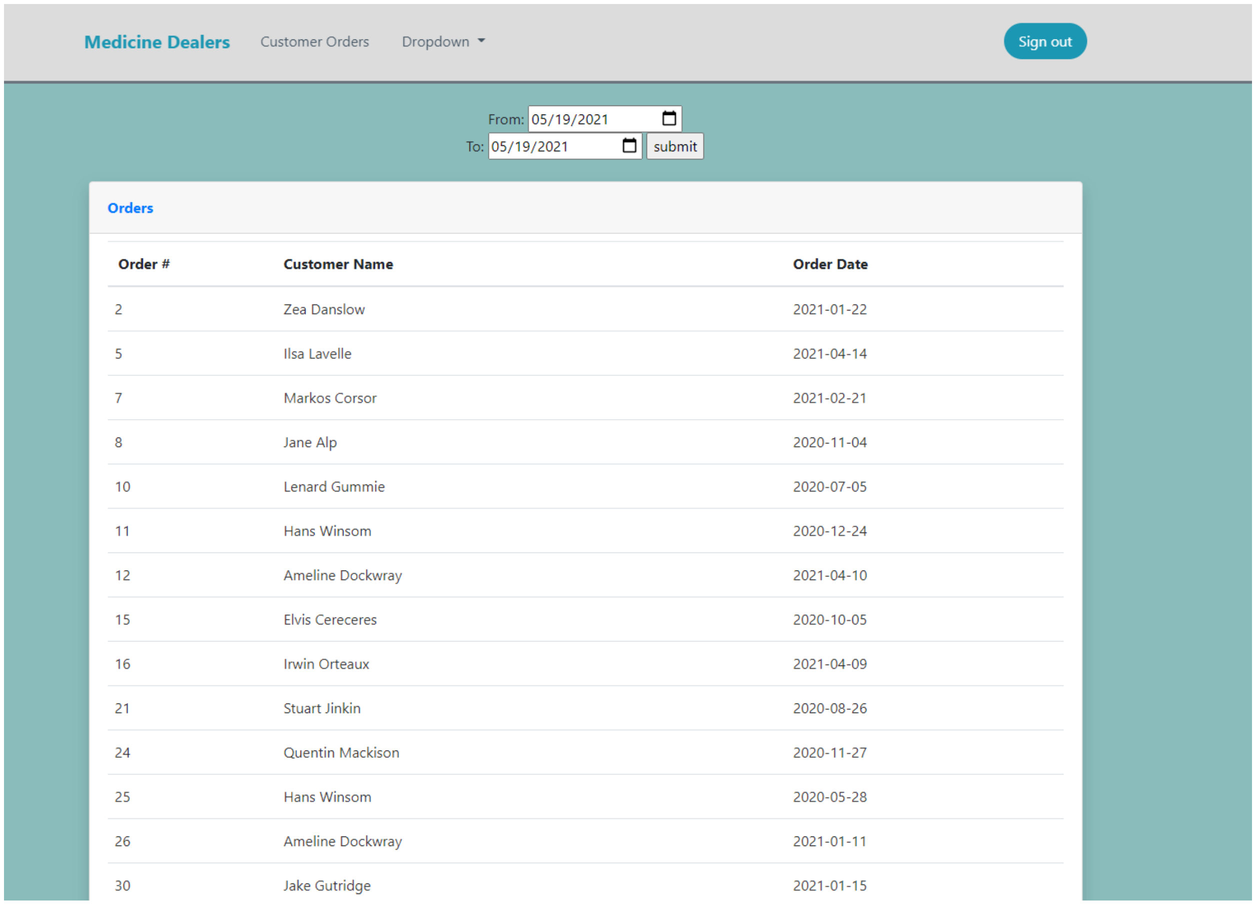
Account:
Tawfic Jobah

First Name: Tawfic	Last Name: Jobah
Email: tawfic661@gmail.com	Phone Number: 16614319268
DOB: 2021-03-01	Doctor Name: mudB
Address: 10322 Grizzly Street, Bakersfield, CA, 93311	

This is similar to the delete, as it is a function but it is an insert function. Checks for duplicate entries and then adds a new customer to the database.



This table uses an inner join to check for orders made by customers from one time frame to another. This view combines two tables which are the customer and order to do the check.



From: 05/19/2021
To: 05/19/2021 submit

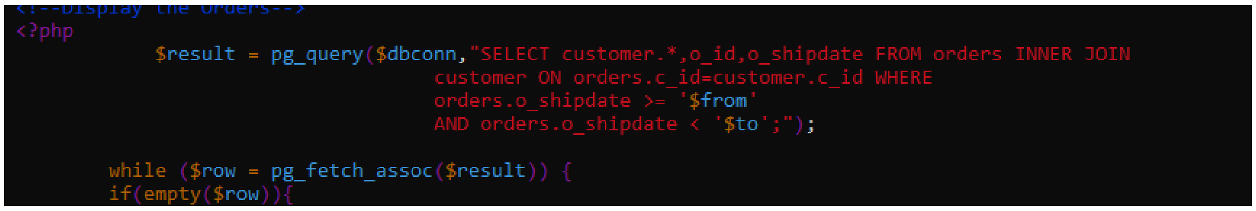
Orders

Order #	Customer Name	Order Date
2	Zea Danslow	2021-01-22
5	Ilsa Lavelle	2021-04-14
7	Markos Corsor	2021-02-21
8	Jane Alp	2020-11-04
10	Lenard Gummie	2020-07-05
11	Hans Winsom	2020-12-24
12	Ameline Dockwray	2021-04-10
15	Elvis Cereceres	2020-10-05
16	Irwin Orteaux	2021-04-09
21	Stuart Jinkin	2020-08-26
24	Quentin Mackison	2020-11-27
25	Hans Winsom	2020-05-28
26	Ameline Dockwray	2021-01-11
30	Jake Gutridge	2021-01-15

5.2 - GUI Programming

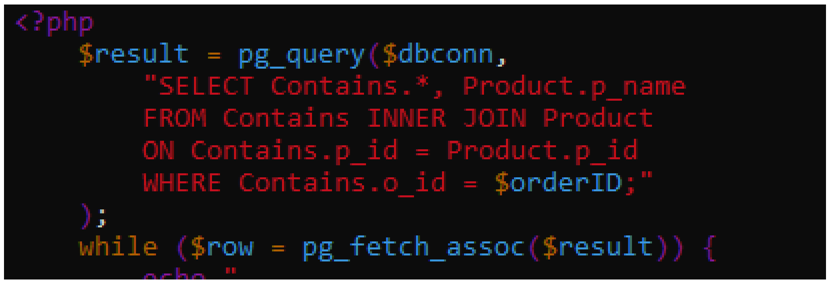
5.2.1 - Server-side Programming

In this section of code, we used an inner join to select all customers who made an order from one time period to another. After the information is queried from the database, it is printed on the screen.



```
<!--Display the Orders-->
<?php
            $result = pg_query($dbconn,"SELECT customer.*,o_id,o_shipdate FROM orders INNER JOIN
                                        customer ON orders.c_id=customer.c_id WHERE
                                        orders.o_shipdate >= '$from'
                                        AND orders.o_shipdate < '$to';");

        while ($row = pg_fetch_assoc($result)) {
        if(empty($row)){
```

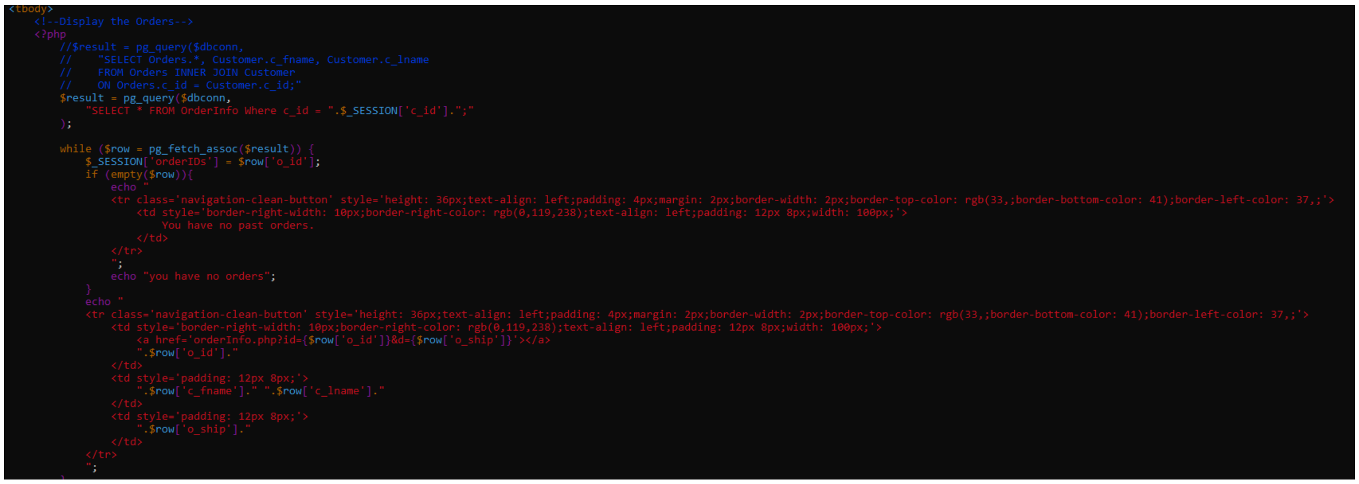
This code also does an inner join but uses it to get product information and displays it on the orders page.



```
<?php
    $result = pg_query($dbconn,
        "SELECT Contains.*, Product.p_name
        FROM Contains INNER JOIN Product
        ON Contains.p_id = Product.p_id
        WHERE Contains.o_id = $orderID;"
    );
    while ($row = pg_fetch_assoc($result)) {
        echo "
```

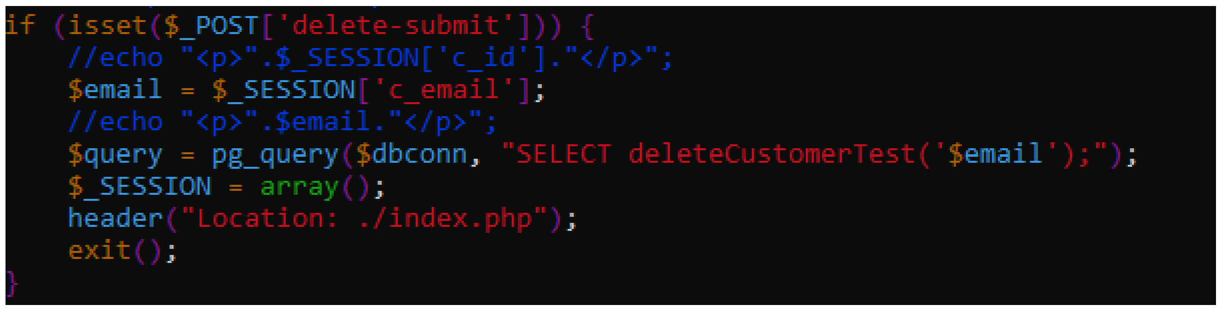
5.2.2 - Middle-tier Programming

Code that loops through and grabs the products from the database and prints to the screen.



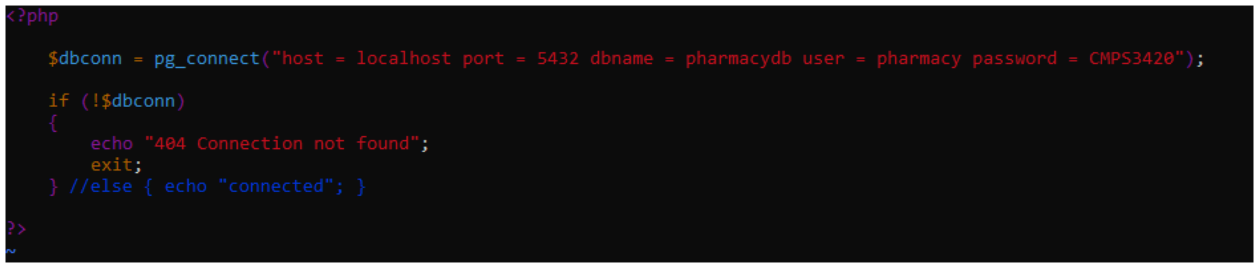
```
<tbody>
    <!--Display the Orders-->
    <?php
        //$result = pg_query($dbconn,
        //    "SELECT Orders.*, Customer.c_fname, Customer.c_lname
        //    FROM Orders INNER JOIN Customer
        //    ON Orders.c_id = Customer.c_id;"
        $result = pg_query($dbconn,
            "SELECT * FROM OrderInfo Where c_id = ".$_SESSION['c_id'].";"
        );

        while ($row = pg_fetch_assoc($result)) {
            $_SESSION['orderIDs'] = $row['o_id'];
            if (empty($row)){
                echo "
                <tr class='navigation-clean-button' style='height: 36px;text-align: left;padding: 4px;margin: 2px;border-width: 2px;border-top-color: rgb(33,;border-bottom-color: 41);border-left-color: 37,;'>
                    <td style='border-right-width: 10px;border-right-color: rgb(0,119,238);text-align: left;padding: 12px 8px;width: 100px;'>
                        You have no past orders.
                    </td>
                </tr>
                ";
                echo "you have no orders";
            }
            echo "
            <tr class='navigation-clean-button' style='height: 36px;text-align: left;padding: 4px;margin: 2px;border-width: 2px;border-top-color: rgb(33,;border-bottom-color: 41);border-left-color: 37,;'>
                <td style='border-right-width: 10px;border-right-color: rgb(0,119,238);text-align: left;padding: 12px 8px;width: 100px;'>
                    <a href='orderInfo.php?id={$row['o_id']}&d={$row['o_ship']}'></a>
                    ".$row['o_id']."
                </td>
                <td style='padding: 12px 8px;'>
                    ".$row['c_fname']." ".$row['c_lname']."
                </td>
                <td style='padding: 12px 8px;'>
                    ".$row['o_ship']."
                </td>
            </tr>
            ";
```

Code that calls a stored function and deletes a customers account.



```
if (isset($_POST['delete-submit'])) {
    //echo "<p>".$_SESSION['c_id']."</p>";
    $email = $_SESSION['c_email'];
    //echo "<p>".$email."</p>";
    $query = pg_query($dbconn, "SELECT deleteCustomerTest('$email');");
    $_SESSION = array();
    header("Location: ./index.php");
    exit();
}
```

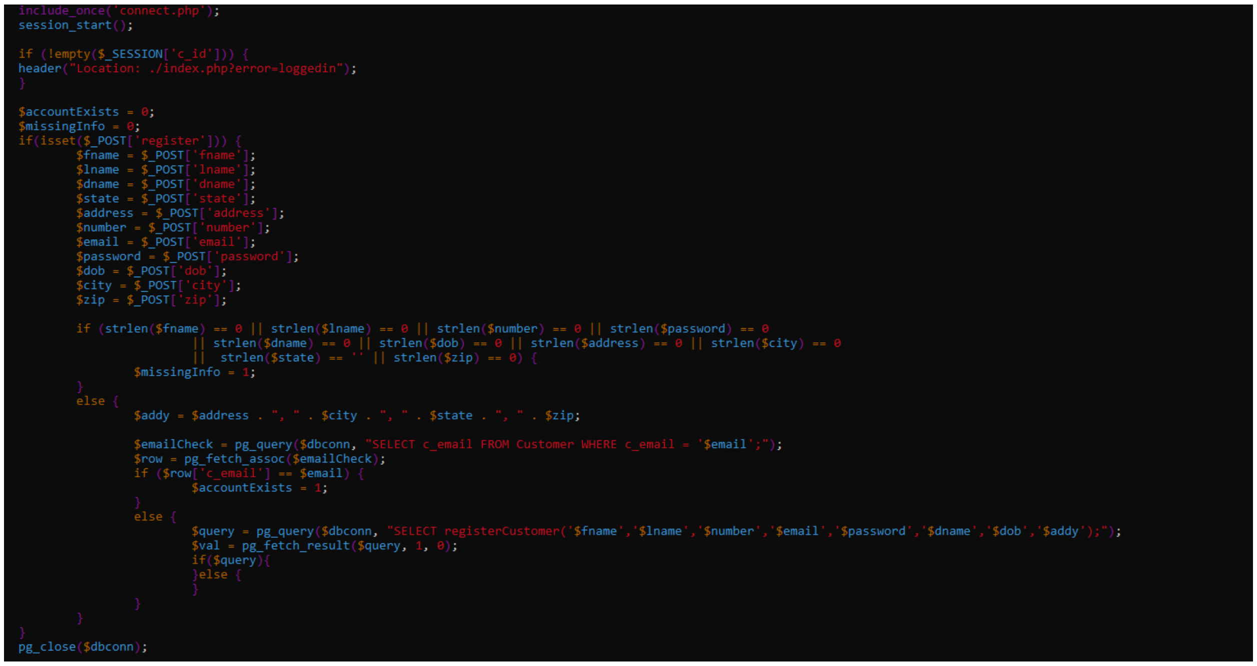
Code that is in a separate php file that is being included in most other php files, that connects to the database.



```
<?php

    $dbconn = pg_connect("host = localhost port = 5432 dbname = pharmacydb user = pharmacy password = CMPS3420");

    if (!$dbconn)
    {
        echo "404 Connection not found";
        exit;
    } //else { echo "connected"; }

?>
~
```

5.2.3 - Client-side Programming

This code is for the register page and it checks every value to make sure that it isn't empty. After that it checks to see if the account exists, then it inserts it into the database. It is also using a function that registers the user.



```php
include_once('connect.php');
session_start();

if (!empty($_SESSION['c_id'])) {
header("Location: ./index.php?error=loggedin");
}

$accountExists = 0;
$missingInfo = 0;
if(isset($_POST['register'])) {
        $fname = $_POST['fname'];
        $lname = $_POST['lname'];
        $dname = $_POST['dname'];
        $state = $_POST['state'];
        $address = $_POST['address'];
        $number = $_POST['number'];
        $email = $_POST['email'];
        $password = $_POST['password'];
        $dob = $_POST['dob'];
        $city = $_POST['city'];
        $zip = $_POST['zip'];

        if (strlen($fname) == 0 || strlen($lname) == 0 || strlen($number) == 0 || strlen($password) == 0
                        || strlen($dname) == 0 || strlen($dob) == 0 || strlen($address) == 0 || strlen($city) == 0
                        ||  strlen($state) == '' || strlen($zip) == 0) {
                $missingInfo = 1;
        }
        else {
                $addy = $address . ", " . $city . ", " . $state . ", " . $zip;

                $emailCheck = pg_query($dbconn, "SELECT c_email FROM Customer WHERE c_email = '$email';");
                $row = pg_fetch_assoc($emailCheck);
                if ($row['c_email'] == $email) {
                        $accountExists = 1;
                }
                else {
                        $query = pg_query($dbconn, "SELECT registerCustomer('$fname','$lname','$number','$email','$password','$dname','$dob','$addy');");
                        $val = pg_fetch_result($query, 1, 0);
                        if($query){
                        }else {
                        }
                }
        }
}
pg_close($dbconn);
```