Table of Contents

DataBase Design

1. Enterprise Description

Intro

This pharmacy business was founded in September 20, 2014 in Toronto, Ontario. We started this pharmacy business because people need medicine. We see the pharmaceutical industry as a vibrant sector of our economy which will ensure future growth for our business.

Mission Statement

“Our mission is to make money selling the best prescription and non-prescription pharmaceutical drugs as well as plant based medicines.”

Business & Financial Objectives

Create a pharmaceutical retail company by offering the most convenient customer experience. Financially we want to be responsible for 0.001% ($328,000) of all pharmaceutical sales in Canada. We are aiming for 0.00001% ($3,280) in the first year, 0.0001% ($32,280) in the second year and 0.001% % ($328,000) in the third year as customers get used to filling their prescriptions online.

Industry Overview & Trends

We operate in the Canadian pharmaceutical industry. Sales of Pharmacy stores in Canada are at $32 billion per year; $20 billion of which is from prescription and non-prescription drugs. Sales growth has been steady increasing at about 5-10% each year for the last 10 years. Prescription drugs sales account for over 50% of this $32 billion.

Major competitors

1.) Shoppers Drug Mart – Canadian company
Sales: (31.8%) $10,445,000,000 Employees: 52,714
2.) Katz Group – Canadian company
Sales: (16.7%) $5,485,000,000 Employees: 12,834
3.) Jean Coutu – Canadian company
Sales: (12.2%) $4,007,000,000 Employees: 19,811
4.) McKesson -American company (7.9%)
Sales: (7.9%) $2,594,000,000 Employees: 6,877

Rest of industry
Sales: (31.4%) $10,314,000,000

Our Place within the Industry

Our plan is to be in the top 25 grossing pharmacy retailers in Canada. Intelligently managing our database tables will allow us to optimize which people and products are performing the best as well as the conditions that facilitated this. Computer procedures can then be done which can give metrics on what the business potentially needs next. Our herbal medicine products come mostly from dried-out plants that the farmers don't value much and this will give us an extra edge in profitability.

Primary Target Market

We have several target market demographics. These include the disproportionately large and growing population of people in their older years and the growing demand for alternative medicines as well as online shoppers who don't want the annoyance of going to stores.

Organizational Structure

Current Employees:

Our group has extensive programming experience which will be vital in developing the database as well as the online store. The team leader Kevin, is very knowledgeable and experienced in this field and is able to distill and distribute tasks to Zachary, Oskar and Dante. All members of the team will contribute to the technical challenges of the database and database management system.

Voting Shares	Profit Shares	Owner
25%	25%	Zachary
25%	25%	Dante
25%	25%	Kevin
25%	25%	Oskar

Financial Structure

A business investment of $1,000 will be sufficient for the team to operate during the first year. Kevin and Oskar have garages which will be used as the initial shop and inventory warehouse for the online and garage shops. In this initial stage only non-prescription pharmaceuticals are to be sold. The $1,000 will primarily be used for web server costs and marketing. Additionally, the database management system will have to be upgraded in order to provide strong concurrency support. The board members will be learning these technical skills in the follow up course they are taking and can handle this task with no cost out of pocket. By the third year a retail store will be opened which will have a yearly lease at $50,000 and a licensed pharmacist for $80,000 per year. At the end of this year the company can peak in sales and be in a prime position to be acquired by one of the top four players in the industry. Each of these four companies will see extra value in our company in order to keep our online technology and product out of the hands of the other top three companies.

Operational Plan

We will make our products available through retail stores as well as our online shop. We are starting with a small shop in Oskar's garage which will not sell prescription pharmaceuticals. Prescription sales will begin on the 3rd year when there is enough money to hire a licensed pharmacist for the new retail store. Kevin's garage will be the warehouse/inventory where the products will be stored to be shipped. His living room will be where products are packaged up and counted.1

We are creating a database for internal management and also for our products. Users will see different views of the database based on their role and security permissions.

Employees who are not high level security will only view their store locations, inventory, products and accounting records associated with their store. This will enable them to do their day to day jobs without focusing on the whole business.

Employees who are high level security above 50 in the database will oversee each store and warehouse's inventory as well as the accounting for each store location and the customers and employees information. Depending on the security level there may be less privilege to view other employee database information.

Customers will have a view of the products list including name, price and which store has stock of each product. They won't be able to see any sales quantity information on product orders.

Paying health magazines to do reviews on our products, social media promotion, forum posting and flyer campaigns especially at industry events are the main things that we will be doing to promote our products. This will be the plan for the first year.

The year two marketing plan will be based on the mathematical metrics run on our database which will direct us to the most effective marketing methods. This will include how our current customers found out about us. The entire first year marketing budget is $250. The second year amount depends on the success of the first year campaign, but is likely to be around $2,500. We won't know until then if it will be the cheap methods or expensive ones that lead to future customers.

References:

http://www.ic.gc.ca/eic/site/oca-bc.nsf/eng/ca02856.html

http://www.ic.gc.ca/eic/site/lsg-pdsv.nsf/eng/h_hn01703.html

2. Functional Requirements

Overview:

The functional requirements of our system describe several different types of users, each with a distinct set of actions available to them. As each different user will have different uses for the system, they will be presented with a different set of options. Having set up functionality specific to each type of user also allows us to keep data hidden from users who don't have full permission. This allows our single database to be seen through multiple different "views", each meeting the requirements of the user in question.

Notes on the Menu System:

The menu system will first prompt the user to choose which type of user they are. Once their type has been chosen the user will be prompted with a list of different types of queries they can make. They choose by typing in a number from 1 to n, where n is the number of query choices. If the user types 0, they will be taken to the previous menu. If the chosen query needs any additional information such as the value of a variable, then the user will be prompted to enter this. The relevant SQL statement will then be executed by the system. In this way, our system exposes queries to the user through entered numbers, along with relevant required information inputted by the user.

Users:

We have three different types of employees: cashiers, pharmacists and administrators. We also have one set of requirements representing the options for all customers. The following is a description of each user's requirements and the expected behaviour of the system:

Cashiers:

Cashiers deal with a small window of our company. They spend all of their time at the register and because of this they only need to be able to do a few things on a day to day basis. Their most common tasks are to get the price of a product and record the sale of a product. When the cashier checks the price, they will need to check if the product is a prescription. Assuming no prior database knowledge, we have made it easy for the cashier to select which query they want based on a simple menu system of options. The system will execute the appropriate prewritten SQL statement based on which option was chosen by the cashier. In the case of the cashier getting the price based on product ID, they will enter the ID after choosing this menu option. This ID will be read as a variable and will be passed to the prewritten SQL statement.

Pharmacists:

Pharmacists need to view a queue of prescriptions to be filled. They are able to cancel prescriptions and add prescriptions. Also, when a prescription is picked up they will select the menu option for removing from it the prescription queue. The pharmacist will then enter the ID of which prescription to remove and the prewritten SQL query will take this variable and execute the statement.

Administrators:

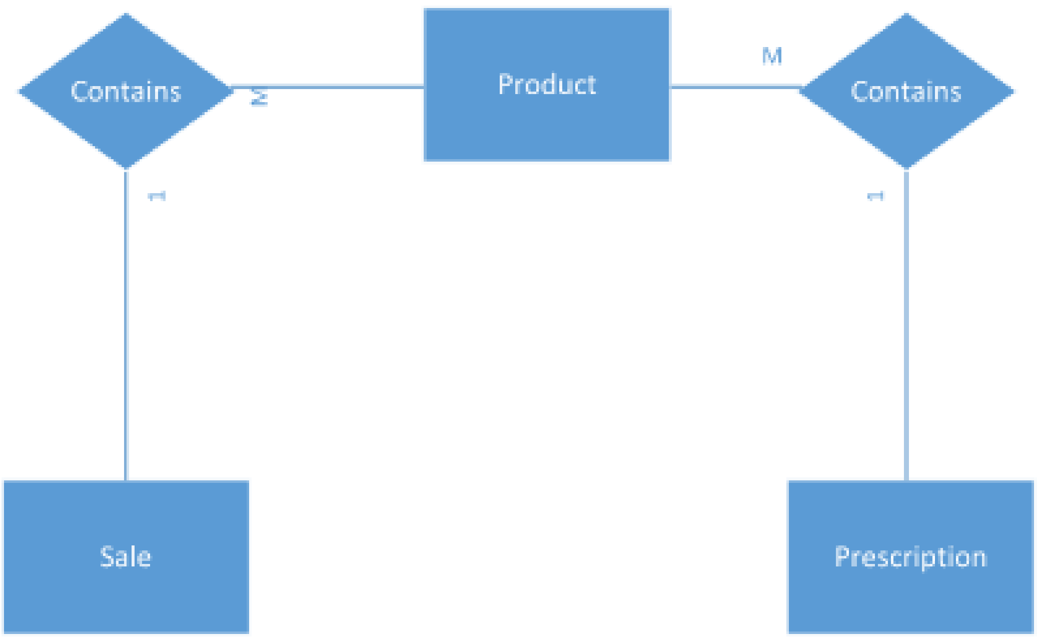
Administrators need to manage and view the pharmacy's sales data in several different ways. The administrator will be able to see all of the products that have ever been sold, as well as all products that share a price with any other product to allow them to compare product values. The administrator will also be able to see the products with the greatest price. To display products of specific stock values, the admin can enter these stock values in that will execute a query to output these products. Another type of query would be to view all products in a certain department that have a stock level in a certain range (which the admin will input). To keep track of employee sales, the admin can also see all products that have ever been sold by a specified employee. While testing out his complex query skills, the administrator also supports a query that shows all customers who have prescriptions for every prescription product in the database. Finally, the administrator can manually edit stock levels of products in case a discrepancy is detected after a physical inventory check.

Customers:

Customers can see a wide variety of product information as well as information pertaining to their specific prescription orders. For products they can see a view ordered by a variety of attributes. These include by price, categories, tags and stock level. After they choose this menu option they have a secondary option to choose which attribute to sort by. The customer can then see the output of this query. A customer wanting to see the lowest prices will find this functionality invaluable when making shopping decisions.

3. Conceptual Schema

3.1. E-R Diagram



The E-R Diagram above outlines the relationship between the tables and represents 5 tables within the database management system. The relationships are represented as such in the diagram above where **sales** and **Prescription** contain a **Product**. The tables are shown in the next section with the appropriate attribute names and sample values.

3.2. Tables

3.2.1. Product Table

SKU	Name	Department	IsPrescription	Price	Stock
037000141303	Head & Shoulders Men Refresh Dandruff Shampoo 700 ml	Personal Care	0	12.99	7
064541319809	Tylenol Childrens Acetaminophen Suspension Usp Cherry 100 ml	Everyday Medicines and First Aid	0	6.96	3
056100074816	Vicks Nyquil Base Cough/Cold Cherry	Everyday Medicines and First Aid	0	9.99	6
7501098605229	SEROQUEL XR (QUETEAPINA)30TAB 300MG	Drugs	1	126.65	2
628791051713	Plan B Emergency Contraception 2 Tablets	Sexual Wellbeing	1	44.49	4
067981100907	Durex Maximum 12 Condoms	Sexual Wellbeing	0	19.49	12
683702100072	Iron Kids Gummies Omega-3s for Smart Kids 60 Gummies	Vitamins and Supplements	0	9.99	0
20080403191042	Lithium Carbonate - 250mg (100 Capsules)	Drugs	1	12.30	7
050200560002	Sunny D Citrus Drink	Groceries	0	3.99	22

The **Product** table describes the products within the pharmacy which includes prescription and non-prescription items. The *Stock Keeping Unit* (*SKU*) uniquely identifies each item which is followed by the *Name* of the product and the *Department* where it is found. The value *IsPrescription* determines whether the item requires a prescription to purchase, where the value 1 means that that product requires a prescription and 0 means it does not. Finally each product will have its *Price* and *Stock* level.

3.2.2. Sales Table

ID	Timestamp	Name
1	2014-11-14-12.12.23.00	Debra
2	2014-11-14-13.05.23.00	Craig
3	2014-11-14-15.22.23.00	Faheed

The **Sales** table describes sales; specifically who performed a sales transaction and records the time it was performed and who did it. The first attribute *ID* uniquely identifies each sale performed, the second attribute records the time (*Timestamp*) that that sale was performed and the third attribute records which employee (*Name*) performed the sale.

3.2.3. Prescription Table

ID	Customer	Doctor	DateOrdered	DateDelivered	isFulfilled
1	John	House	2014-11-13-12.00.23.00	2014-11-14-15.22.23.00	1
2	Mary	Chase	2014-11-13-12.00.23.00	Null	0
3	Toop	Pepper	2014-11-13-12.00.23.00	Null	0

The **Prescription** table describes prescriptions; specifically the *Customer* who was prescribed the product, the *Doctor* who prescribed it and the *DateOrdered* and *DateDelivered*. The value *Null* identifies that the prescription has not been ordered or has not been delivered. The *isFulfilled* attribute determines whether the prescription has been filled, where 1 means that the prescription has been filled, and 0 means it has not been filled.

3.2.4. SaleProduct Table

ID	SaleID	ProductSKU	Quantity
1	1	037000141303	2
2	1	067981100907	3
3	2	064541319809	1
4	2	056100074816	1
5	2	683702100072	1
6	2	050200560002	7
7	3	7501098605229	1
8	3	20080403191042	1

The **SaleProduct** table describes the relationship between the products and the sales; specifically who sold what product and the amount that was sold. This table has one primary key unique identifier *ID* and two foreign keys *Sale* and *Product* which correspond to other tables within the database. The first foreign key *Sale* references the *ID* in the **Sales** table. The second foreign key *Product* references the *SKU* in the **Product** table. And finally the *Quantity* is the amount of that product sold by that employee.

3.2.5. PrescriptionProduct Table

ID	Prescription	Product	Quantity
1	1	7501098605229	1
2	1	20080403191042	2
3	2	628791051713	1
4	3	7501098605229	1
5	3	20080403191042	1
6	3	628791051713	1

The **PrescriptionProduct** table outlines the relationship between the prescription and the associated prescription product being prescribed. It has a unique identifier *ID* and two foreign keys Prescription and Product. The first foreign key *Prescription* references the *ID* in the **Prescription** table. The second foreign key Product references the *SKU* in the **Product** table. The last attribute is the *Quantity* of the prescribed product.

4. Data Dictionary

Column Name	**Data Type**	**Reasoning**
	Product Table:	
ID	integer NOT NULL	Internal identifiers for products
Name	varchar(500) NOT NULL	Name by which clients identify the product
SKU	varchar(25) NOT NULL	Industry Identifier for product
Department	varchar(100)	Labels to categorize products by use
IsPrescription	integer	1 or 0; Determines whether product can be sold over the counter
Price	double NOT NULL	Monetary value
Stock	integer NOT NULL	Quantity of the product contained in the store
	Sale Table:	
ID	integer NOT NULL	Internal identifiers for each transaction processed at a Point of Sale
Timestamp	timestamp NOT NULL	Date at which sale happened
Employee	varchar(25) NOT NULL	Employee who processed the sale
	Prescription Table:	
ID	integer NOT NULL	Internal identifiers for prescriptions
Customer	varchar(100) NOT NULL	Name of Customer who will pick it up
Doctor	varchar(100) NOT NULL	Name of Doctor who prescribed it
DateOrdered	timestamp NOT NULL	Date when the Prescription was ordered.
DateDelivered	timestamp	Date when the Prescription was delivered, if it is null, the prescription has not been picked up.
isFulfilled	integer NOT NULL	1 or 0; Determines whether the pharmacist has gathered the contents of the prescription.
	SaleProduct Relation:	
ID	integer NOT NULL	Database-enforced primary key
SaleID	integer NOT NULL	Sale identifier
ProductID	integer NOT NULL	Product Identifier
Quantity	integer NOT NULL	Amount of this product sold in this sale.
	PrescriptionProduct Relation:	
ID	integer NOT NULL	Database-enforced primary key
PrescriptionID	integer NOT NULL	Prescription identifier
ProductID	integer NOT NULL	Product Identifier
Quantity	integer NOT NULL	Amount of this product in this prescription.

<u>User Manual</u>

5. User Manual

5.1. Installation

<u>Overview</u>:

You must be in a Linux environment to run the script required to install the database. You must also have IBM DB2 Installed. To run the Pharmacy Database you must have the following files and directories in the current directory which you are in:

Required files in the current working directory:

- TuringAssistant *(Shell script)*
- Pharmacy *(Shell script)*
- PharmacyDatabase*(zip file)*

Within the zip file you will have the following shell scripts which are required to run the database:

The scripts that are present within the PharmacyDatabase.zip are:

- DB2Admin
- DB2Employee
- DB2Create
- DB2Customer
- DB2Drop
- DB2MainMenu
- DB2Pharmacist
- DB2Populate

<u>Step 1</u>:

Open a terminal and cd into the directory containing the files TuringAssistant (Shell script), Pharmacy *(Shell script),PharmacyDatabase(zip file)*. Run the following command:

```
chmod 755 Pharmacy
```

<u>Step 2</u>:

Now that the appropriate files have been given the proper rights it is now time to transfer it to the turing directory or any other directory you wish to install the database. To install it within turing in the terminal run this command using your SCS username:

```
scp  PharmacyDatabase.zip Pharmacy yourusername@turing,acs.ryerson.ca
```

<u>Step 3</u>:

Secure shell (SSH) with your SCS username into turing (or any other directory):

```
ssh yourusername@turing,acs.ryerson.ca
```

Step 4:

Now that you are within turing (or any other directory) run the following command to initiate the install options.

```
./Pharmacy
```

Step 5:

Follow to onscreen prompts to install or start the database. Please refer to the ***How to Use*** section for further instructions.

Note:

Provided is the *DB2Run* script which will automatically transfer all necessary files to turing and start the database. In order to use it please run the command in the same directory as the pharmacy (*shell script*) and PharmacyDatabase.zip:

```
chmod 775 DB2Pharmacy
```

Once the files have been given the appropriate permissions run the command and simply follow the onscreen prompts:

```
./DB2Run
```

This file will automatically transfer and do all relevant actions to enter the database management system.

5.2. How to Use

To use the pharmacy database simply follow the onscreen prompts for relevant input to interact with the database management system.

System

6. Source Code

6.1. Create Tables

```
#!/bin/ksh

db2 connect to db2data > /dev/null

#Create Tables
echo "[DB2] Creating Tables..."

#Product Table
db2 "CREATE TABLE Product(
        SKU varchar(25) NOT NULL,
        Name varchar(70) NOT NULL,
        Department varchar(40),
        IsPrescription integer,
        Price decimal (20,2) NOT NULL,
        Stock integer NOT NULL DEFAULT 0,
        PRIMARY KEY(SKU)
)"

#Sales Table
db2 "CREATE TABLE Sale(
        ID integer NOT NULL GENERATED BY DEFAULT AS IDENTITY (START WITH 1),
        Timestamp timestamp NOT NULL,
        Name varchar(30) NOT NULL DEFAULT,
        PRIMARY KEY(ID)
)"

#Prescription Table
db2 "CREATE TABLE Prescription(
        ID integer NOT NULL GENERATED BY DEFAULT AS IDENTITY (START WITH 1),
        Customer varchar(30) NOT NULL,
        Doctor varchar(30) NOT NULL,
        DateOrdered timestamp NOT NULL,
        DateDelivered timestamp,
        isFulfilled integer NOT NULL DEFAULT 0,
        PRIMARY KEY(ID)
)"

#SalesProduct Table
db2 "CREATE TABLE SaleProduct(
        ID integer NOT NULL GENERATED BY DEFAULT AS IDENTITY (START WITH 1),
        SaleID integer NOT NULL,
        ProductSKU varchar(70) NOT NULL,
        Quantity integer NOT NULL DEFAULT 0,
        PRIMARY KEY(ID),
        FOREIGN KEY(SaleID) REFERENCES Sale(ID),
        FOREIGN KEY(ProductSKU) REFERENCES Product(SKU)
)"

#PrescriptionProduct Table
```

```
db2 "CREATE TABLE PrescriptionProduct(
        ID integer NOT NULL GENERATED BY DEFAULT AS IDENTITY (START WITH 1),
        PrescriptionID integer NOT NULL,
        ProductSKU varchar(70) NOT NULL,
        Quantity integer NOT NULL DEFAULT 0,
        PRIMARY KEY(ID),
        FOREIGN KEY(ProductSKU) REFERENCES Product(SKU),
        FOREIGN KEY(PrescriptionID) REFERENCES Prescription(ID)
)"
echo "[DB2] Creating Tables: Done"
echo "[DB2] Creating Views:..."

db2 "CREATE VIEW ProductView AS SELECT
        Name, Price, Stock, Department FROM Product"

echo "[DB2] Creating Views: Done"

echo "[DB2] Creating triggers:..."
db2 "CREATE TRIGGER Stockabove0
        AFTER
        INSERT ON SaleProduct
        referencing NEW AS NewProd
        FOR EACH ROW
        BEGIN ATOMIC
        UPDATE Product SET Stock = Stock - NewProd.quantity WHERE SKU = NewProd.ProductSKU;
        END"

echo "[DB2] Creating triggers: DONE"
echo "[DB2] Creating assertion constraints..."
db2 "ALTER TABLE Product ADD CONSTRAINT price_constraint CHECK (Price >= 0)"
echo "[DB2] Creating assertion constraints: DONE"
```

6.2. Populate Tables

```
#!/bin/ksh

db2 connect to db2data > /dev/null

echo "[DB2] Populating Tables..."

#Product Table
echo "Populating Tables... Product"
db2 "INSERT INTO PRODUCT VALUES ('037000141303', 'Head & Shoulders Men Refresh Dandruff Shampoo 700 ml', 'Personal Care', 0, 12.99, 7)"
db2 "INSERT INTO PRODUCT VALUES ('064541319809', 'Tylenol Childrens Acetaminophen Suspension Usp Cherry 100 ml', 'Everyday Medicines and First Aid', 0, 6.96, 3)"
db2 "INSERT INTO PRODUCT VALUES ('056100074816', 'Vicks Nyquil Base Cough/Cold Cherry', 'Everyday Medicines and First Aid', 0, 9.99, 6)"
db2 "INSERT INTO PRODUCT VALUES ('7501098605229', 'SEROQUEL XR (QUETEAPINA)30TAB 300MG', 'Drugs', 1, 126.65, 2)"
db2 "INSERT INTO PRODUCT VALUES ('628791051713', 'Plan B Emergency Contraception 2 Tablets', 'Sexual Wellbeing', 1, 44.49, 4)"
db2 "INSERT INTO PRODUCT VALUES ('067981100907', 'Durex Maximum 12 Condoms', 'Sexual Wellbeing', 0, 19.49, 12)"
db2 "INSERT INTO PRODUCT VALUES ('683702100072', 'Iron Kids Gummies Omega-3s for Smart Kids 60 Gummies', 'Vitamins and Supplements', 0, 9.99, 0)"
db2 "INSERT INTO PRODUCT VALUES ('20080403191042', 'Lithium Carbonate - 250mg (100 Capsules)' , 'Drugs', 1, 12.30, 7)"
```

```
db2 "INSERT INTO PRODUCT VALUES ('050200560002', 'Sunny D Citrus Drink', 'Groceries', 0, 3.99, 22)"

#Sales Table
echo "Populating Tables... Sales"
db2 "INSERT INTO SALE (Timestamp, Name) VALUES ('2014-11-14-12.12.23.00', 'Debra')"
db2 "INSERT INTO SALE (Timestamp, Name) VALUES ('2014-11-14-13.05.23.00', 'Craig')"
db2 "INSERT INTO SALE (Timestamp, Name) VALUES ('2014-11-14-15.22.23.00', 'Faheed')"

#Prescription Table
echo "Populating Tables... Prescription"
db2 "INSERT INTO PRESCRIPTION (Customer, Doctor, DateOrdered, DateDelivered, IsFulfilled) VALUES ('John', 'House', '2014-
11-13-12.00.23.00', '2014-11-14-15.22.23.00', 1)"
db2 "INSERT INTO PRESCRIPTION (Customer, Doctor, DateOrdered, DateDelivered, IsFulfilled) VALUES ('Mary', 'Chase', '2014-
11-13-12.00.23.00', Null, 0)"
db2 "INSERT INTO PRESCRIPTION (Customer, Doctor, DateOrdered, DateDelivered, IsFulfilled) VALUES ('Toop', 'Pepper', '2014-
11-13-12.00.23.00', Null, 0)"

#SalesProduct Table
echo "Populating Tables... SalesProduct"
db2 "INSERT INTO SALEPRODUCT (SaleID, ProductSKU, Quantity) VALUES (1,'628791051713',2)"
db2 "INSERT INTO SALEPRODUCT (SaleID, ProductSKU, Quantity) VALUES (1,'067981100907',3)"
db2 "INSERT INTO SALEPRODUCT (SaleID, ProductSKU, Quantity) VALUES (2,'064541319809',1)"
db2 "INSERT INTO SALEPRODUCT (SaleID, ProductSKU, Quantity) VALUES (2,'056100074816',1)"
db2 "INSERT INTO SALEPRODUCT (SaleID, ProductSKU, Quantity) VALUES (2,'683702100072',1)"
db2 "INSERT INTO SALEPRODUCT (SaleID, ProductSKU, Quantity) VALUES (2,'050200560002',7)"
db2 "INSERT INTO SALEPRODUCT (SaleID, ProductSKU, Quantity) VALUES (3,'7501098605229',1)"
db2 "INSERT INTO SALEPRODUCT (SaleID, ProductSKU, Quantity) VALUES (3,'20080403191042',1)"

#PerscriptionProduct Table
echo "Populating Tables... PreProduct"
db2 "INSERT INTO PRESCRIPTIONPRODUCT (PrescriptionID, ProductSKU, Quantity) VALUES (1,'7501098605229',1)"
db2 "INSERT INTO PRESCRIPTIONPRODUCT (PrescriptionID, ProductSKU, Quantity) VALUES (1,'20080403191042', 2)"
db2 "INSERT INTO PRESCRIPTIONPRODUCT (PrescriptionID, ProductSKU, Quantity) VALUES (2,'628791051713', 1)"
db2 "INSERT INTO PRESCRIPTIONPRODUCT (PrescriptionID, ProductSKU, Quantity) VALUES (3,'7501098605229', 1)"
db2 "INSERT INTO PRESCRIPTIONPRODUCT (PrescriptionID, ProductSKU, Quantity) VALUES (3,'628791051713', 1)"
db2 "INSERT INTO PRESCRIPTIONPRODUCT (PrescriptionID, ProductSKU, Quantity) VALUES (3,'20080403191042', 1)"

echo "[DB2] Populating Tables: DONE"
```

6.3. Drop Tables

```
#!/bin/ksh

db2 connect to db2data > /dev/null

echo "[DB2] Drop triggers:..."
db2 "DROP TRIGGER Stockabove0"
echo "[DB2] Drop triggers: DONE"

echo "[DB2] Drop Tables:..."
db2 "DROP TABLE Product"
db2 "DROP TABLE Sale"
db2 "DROP TABLE Prescription"
db2 "DROP TABLE SaleProduct"
db2 "DROP TABLE PrescriptionProduct"
```

```
echo "[DB2] Drop Tables: DONE"

echo "[DB2] Drop Views:..."
db2 "DROP VIEW ProductView"
echo "[DB2] Drop Views: DONE"
```

6.4. Main Menu

```
#!/bin/ksh
#DB2 Pharmacy - Main Menu

db2 connect to db2data > /dev/null

SELECTION=-1
# Will print infinitely until the user enters 0 to exit the program.
while [ $SELECTION -ne 0 ]
do
	SELECTION=-1
	#Print Main Menu
	echo "----<USER SELECTION MENU>----"
	echo "1 | Administrators"
	echo "2 | Pharmacists"
	echo "3 | Employees"
	echo "4 | Customers"
	echo "0 | EXIT DATABASE"
	echo "----------------------------"
	echo -n "Please enter the number corresponding to the selection above: " ; read SELECTION

	#Select and run next script to run that user queries
	case $SELECTION in
		1)
			#Start Admin script with associated queries
			./DB2Admin
			;;
		2)
			#Start Pharmacists script with associated queries
			./DB2Pharmacist
			;;
		3)
			#Start Cashier script with associated queries
			./DB2Employee
			;;
		4)
			#Start Customer script with associated queries
			./DB2Customer
			;;
		0)
			#Check if user wants to exit
			echo "Are you sure you want to exit?"
			echo -n "Enter 'Yes' to exit (Enter anything to return to MAIN MENU): " ; read exit_message
			if [ "$exit_message" = "Yes" ]
			then
				SELECTION=0
			else
				SELECTION=-1
				fi
				;;
			*)
				#If user enters soemthing that is NOT part of the selection
				echo "ERROR: Please ONLY enter the selection numbers."
				echo " "
				SELECTION=-1
				;;
		esac
done
```

6.5. Submenus

6.5.1. Administrator

```
#!/bin/ksh
#DB2 Pharmacy - Adiministrators Script

db2 connect to db2data > /dev/null

SELECTION=-1
# Will print infinitely until the user enters 0 to exit the previous Script.
while [ $SELECTION -ne 0 ]
do
		SELECTION=-1
		#Print Adiministrator Menu
		echo "----<ADMINISTRATOR MENU>----"
		echo "1 | Show All Products sold"
		echo "2 | Show All Products that have a shared price"
		echo "3 | Show Products with the greatest price"
		echo "4 | Show Products of certain stock values"
		echo "5 | Show Products in a certain Department in a certain stock range"
		echo "6 | Show Products sold by a certain employee"
		echo "7 | Show Customers who have prescriptions for every prescription product"
		echo "8 | Update stock levels of a product"
		echo "0 | RETURN TO MAIN MENU"
		echo "----------------------------"
		echo -n "Please enter the number corresponding to the selection above: " ; read SELECTION

		#Select and run user queries
		case $SELECTION in
			1)
				#Show All Products sold
					echo -n "All products sold: " ;
					db2 "SELECT * FROM Product WHERE EXISTS (SELECT * FROM SaleProduct WHERE
Product.SKU = SaleProduct.ProductSKU)"
				;;
			2)
				#Show All Products that have a shared price
					echo -n "All products that share a price with at least one other product: " ;
					db2 "SELECT * FROM Product x WHERE Price = ANY (SELECT Price FROM Product y
WHERE x.Name <> y.Name)";
				;;
			3)
				#Show Products with the greatest price
```

```
                        echo -n "All products that have the greatest price: " ;
                        db2 "SELECT * FROM Product x WHERE Price >= ALL (SELECT Price FROM Product y)"
                ;;
        4)
                #Show Products of certain stock values
                        echo -n "Enter 3 stock values: " ; read s1; read s2; read s3;
                        db2 "SELECT * FROM Product WHERE Stock IN ($s1, $s2, $s3)"
                ;;
        5)
                #Show Products in a certain Department in a certain stock range
                        echo -n "Enter the Department: " ; read dep;
                        echo -n "Enter the minimum and maximum stock to find: " ; read min; read max;
                        db2 "SELECT * FROM Product WHERE Department = '$dep' AND Stock BETWEEN $min
AND $max"
                ;;
        6)
                #Show Products sold by a certain employee
                        echo -n "Enter the Employee name: " ; read employee;
                        db2 "SELECT * FROM Product WHERE EXISTS
                        (SELECT * FROM Sale
                        INNER JOIN SaleProduct
                        ON Sale.ID = SaleProduct.SaleID
                        AND Sale.Name = '$employee'
                        AND SaleProduct.ProductSKU = Product.SKU)"
                ;;
        7)
                #Show Customers who have prescriptions for every prescription product
                        echo -n "Customers with prescriptions for every prescription product: " ;
                        db2 "SELECT Customer FROM Prescription WHERE NOT EXISTS (
                        SELECT * FROM Product WHERE IsPrescription = 1 AND NOT EXISTS (
                        SELECT * FROM PrescriptionProduct WHERE Product.IsPrescription = 1
                        AND Product.SKU = PrescriptionProduct.ProductSKU AND
                        PrescriptionProduct.PrescriptionID = Prescription.ID))"
                ;;
        8)
                #Update stock levels of a product
                        echo -n "Enter SKU of product to update: " ; read sku;
                        echo -n "Enter new amount of stock: " ; read stock;
                        db2 "UPDATE Product SET stock = $stock WHERE SKU = $sku"

                ;;
        0)
                #Return to Main Menu
                SELECTION=0
                ;;
        *)
                #If user enters soemthing that is NOT part of the selection
                echo "ERROR: Please ONLY enter the selection numbers."
                echo " "
                SELECTION=-1
                ;;
    esac
done
```

6.5.2. Pharmacist

```
#!/bin/ksh
#DB2 Pharmacy - Pharmacist Script

db2 connect to db2data > /dev/null

SELECTION=-1
# Will print infinitely until the user enters 0 to exit the previous Script.
while [ $SELECTION -ne 0 ]
do
	SELECTION=-1
	#Print Pharmacist Menu
	echo "----<PHARMACIST MENU>----"
	echo "1 | View all Prescriptions"
	echo "2 | Queue of pending Prescriptions"
	echo "3 | Fill a Prescription"
	echo "0 | RETURN TO MAIN MENU"
	echo "----------------------------"
	echo -n "Please enter the number corresponding to the selection above: " ; read SELECTION

	#Select and run user queries
	case $SELECTION in
		1)
			#View All Prescriptions
			db2 "SELECT * FROM Prescription"
			;;
		2)
			#Queue of pending Prescriptions
			db2 "SELECT * FROM Prescription WHERE isFulfilled = 0 ORDER BY DateOrdered FETCH FIRST 5
ROWS ONLY"
			;;
		3)
			#Place an order for a pending prescription
			echo -n "Enter the Prescription ID: " ; read sale_id
			echo -n "The Products in this prescription are as follows: " ;
			db2 "SELECT Name, Stock, Department, Price FROM Product
			INNER JOIN PrescriptionProduct
			ON PrescriptionProduct.PrescriptionID = $sale_id
			AND Product.SKU = PrescriptionProduct.ProductSKU"

			echo -n "Once you have ordered the products above please enter 'Yes to confirm': " ; read
message
			if [ "$message" = "Yes" ]
				then
					db2 "UPDATE Prescription SET DateDelivered=current timestamp, isFulfilled=
1 WHERE ID = $sale_id" > /dev/null
					echo "The prescription with the id $sale_id has been filled.";
			fi
			;;
		0)
			#Return to Main Menu
			SELECTION=0
			;;
```

```
                *)
                        #If user enters soemthing that is NOT part of the selection
                        echo "ERROR: Please ONLY enter the selection numbers."
                        echo " "
                        SELECTION=-1
                        ;;
        esac
done
```

6.5.3. Employee

```
#!/bin/ksh
#DB2 Pharmacy - Employee Script

db2 connect to db2data > /dev/null

#read the employee's Name
echo -n "Enter your name: " ; read emp_name

SELECTION=-1
# Will print infinitely until the user enters 0 to exit the previous Script.
while [ $SELECTION -ne 0 ]
do
        SELECTION=-1
        #Print Employee Menu
        echo "----<EMPLOYEE MENU>----"
        echo "1 | Lookup SKU"
        echo "2 | Price of a Product"
        echo "3 | Product Information"
        echo "4 | Record a Sale"
        echo "0 | RETURN TO MAIN MENU"
        echo "----------------------------"
        echo -n "Please enter the number corresponding to the selection above: " ; read SELECTION

        #Select and run user queries
        case $SELECTION in
                1)
                        #Sort by Name
                        echo -n "Enter the name or a partial name: " ; read SEARCH
                        db2 "SELECT Name, SKU FROM Product WHERE name LIKE '%$SEARCH%'"
                        ;;
                2)
                        #Price of product
                        echo -n "Enter the Product SKU: " ; read SKU
                        db2 "SELECT price FROM product WHERE sku = '$SKU'"
                        ;;
                3)
                        #Price of product
                        echo -n "Enter the Product SKU: " ; read SKU
                        db2 "SELECT Name, Department, IsPrescription, Price, Stock FROM product WHERE sku = '$SKU'"
                        ;;
                4)
                        #Record Sale
                        echo -n "Enter the Product SKU: " ; read SKU
                        echo -n "Enter the Product Quantity: " ; read prod_quan
                        EMP=$(db2 -x -n "SELECT 1 FROM Product where SKU = '$SKU' AND Stock - $prod_quan >= 0 AND
IsPrescription = 0")
                        EMP="${EMP#"${EMP%%[![:space:]]*}"}"
                        if [ "$EMP" -eq 1 ]
                            then

                                db2 "INSERT INTO Sale (Timestamp, Name) VALUES (current timestamp,
'$emp_name')"  > /dev/null
                                    EMP=$(db2 -x -n "SELECT MAX(ID) FROM SALE")
                                    EMP="${EMP#"${EMP%%[![:space:]]*}"}"
                                db2 "INSERT INTO SaleProduct (SaleID, ProductSKU, Quantity) VALUES ($EMP,
'$SKU', $prod_quan)"  > /dev/null
                                db2 "Select * FROM SaleProduct WHERE SaleID = $EMP."
                            else
                                echo "[ERROR] Product $SKU is either out-of-stock or a Prescription product or
not a Product at all"
                                db2 "SELECT * FROM product WHERE sku = '$SKU'"
                        fi
                        ;;
                0)
                        #Return to Main Menu
                        SELECTION=0
                        ;;
                *)
                        #If user enters soemthing that is NOT part of the selection
                        echo "ERROR: Please ONLY enter the selection numbers."
                        echo " "
                        SELECTION=-1
                        ;;
        esac
done
```

6.5.4. Customer

```
#!/bin/ksh
#DB2 Pharmacy - Customer Script

db2 connect to db2data > /dev/null

SELECTION=-1
# Will print infinitely until the user enters 0 to exit the previous Script.
while [ $SELECTION -ne 0 ]
do
        SELECTION=-1
        #Print Customer Menu
        echo "----<CUSTOMER MENU>----"
        echo "1 | List Products"
        echo "2 | Submit a Prescription"
        echo "3 | Prescription History"
        echo "0 | RETURN TO MAIN MENU"
        echo "----------------------------"
        echo -n "Please enter the number corresponding to the selection above: " ; read SELECTION

        #Select and run user queries
        case $SELECTION in
```

```
		1)
				#Organize the Available products by user selected categories
				echo "----<SORT PRODUCTS BY:>----"
				echo "1 | Name"
				echo "2 | Price"
				echo "3 | Department"
				echo "4 | Stock Level"
				echo "----------------------------"
				echo -n "Please enter the number corresponding to the selection above: " ; read sort_opt
				case $sort_opt in
					1)
						#Sort by Name
						db2 "SELECT * FROM ProductView ORDER BY Name"
						;;
					2)
						#Sort by Price
						db2 "SELECT * FROM ProductView ORDER BY Price"
						;;
					3)
						#Sort by Department
						db2 "SELECT * FROM ProductView ORDER BY Department"
						;;
					4)
						#Sort by Stock Level
						db2 "SELECT * FROM ProductView ORDER BY Stock"
						;;
					*)
						#If user enters soemthing that is NOT part of the selection
						echo "ERROR: Please ONLY enter the selection numbers."
						echo " "
						;;
				esac
				;;
		2)
				#Submit a Prescription
				echo -n "Enter your Name: " ; read cus_name
				echo -n "Enter your Doctor's Name: " ; read doc_name
				db2 "INSERT INTO Prescription (Customer, Doctor, DateOrdered) VALUES ('$cus_name',
'$doc_name', current timestamp)"

				id_exists=$(db2 -x -n "SELECT MAX(ID) FROM Prescription WHERE Customer = '$cus_name' AND
Doctor = '$doc_name'")
				id_exists="${id_exists#"$id_exists%%[![:space:]]*}"}"
				while [ 1 ]
				do
					echo -n "Enter Product SKU (Enter 'exit' to finish): " ; read SKU
					if [ "$SKU" -eq 'exit' ]
					then
						break

					fi
					echo -n "Enter Product Quanity: " ; read quantity
					EMP=$(db2 -x -n "SELECT 1 FROM Product where SKU = '$SKU' AND Stock - $quantity >=
0 AND IsPrescription = 1")
```

```
                EMP="${EMP#"${EMP%%[![:space:]]*}"}"
                if [ "$EMP" -eq 1 ]
                    then
                        db2 "INSERT INTO PrescriptionProduct (PrescriptionID, ProductSKU,
Quantity) VALUES ($id_exists, '$SKU', $quantity)"
                        db2 "SELECT Name FROM PRoduct WHERE SKU = '$SKU'"
                    else
                        echo "[ERROR] Product '$SKU' is either out-of-stock or not a
Prescription product or not a Product at all!"
                fi
            done
            prod_exists=$(db2 -x -n "SELECT 1 FROM PrescriptionProduct WHERE PrescriptionID = $id_exists")
            prod_exists="${prod_exists#"${prod_exists%%[![:space:]]*}"}"
            if [ "$prod_exists" -eq 1 ]
                then
                    db2 "SELECT Doctor, DateOrdered, DateDelivered, isFulfilled FROM
prescription WHERE id = $id_exists"
                    db2 "SELECT Product.Name, PrescriptionProduct.quantity FROM Product
INNER JOIN PrescriptionProduct ON Product.SKU = PrescriptionProduct.ProductSKU AND PrescriptionProduct.PrescriptionID =
$id_exists"
            fi
            ;;
        3)
            #Prescription History
            echo -n "Enter your Name: " ; read cus_name
            max_id=0
            while [ 1 ]
            do
                sale_id=$(db2 -x -n "SELECT MIN(ID) FROM Prescription WHERE customer = '$cus_name'
AND ID > $max_id")
                sale_id="${sale_id#"${sale_id%%[![:space:]]*}"}"
                if [ "$sale_id" == "-" ]
                    then
                        break
                fi

                max_id=$sale_id
                db2 "SELECT Doctor, DateOrdered, DateDelivered, isFulfilled FROM prescription WHERE
id = $sale_id"
                db2 "SELECT Product.Name, PrescriptionProduct.quantity FROM Product INNER JOIN
PrescriptionProduct ON Product.SKU = PrescriptionProduct.ProductSKU AND PrescriptionProduct.PrescriptionID = $sale_id"
            done
            ;;
        0)
            #Return to Main Menu
            SELECTION=0
            ;;
        *)

            SELECTION=-1
            ;;
    esac
done
```

6.6. Pharmacy (Executable)

```
#!/bin/bash
#DB2 Pharmacy - Pharmacy

#Initialize Database
. db2init > /dev/null
wait

SELECTION=-1
# Will print infinitely until the user enters 0 to exit the previous Script.
while [ $SELECTION -ne 0 ]
do
	echo "----<DB2 OPTIONS>----"
	if [ ! -f DB2MainMenu ];then
		echo "1 | Automatically Install Database"
	else
		echo "1 | Automatically Start Database"
	fi
	echo "2 | ADVANCED OPTIONS"
	echo "0 | EXIT DATABASE"
	echo "----------------------------"
	echo -n "Please enter the number corresponding to the selection above: " ; read SELECTION

	#Select and run user queries
	case $SELECTION in
		1)
			if [ ! -f DB2MainMenu ];then
				#unzip all nessesary files and delete the zip file
				echo "[INITIALIZING] Unzipping Nesseary Files and removing."
				unzip -j -q -o PharmacyDatabase.zip 2> /dev/null

				#Give permmissions needed to the files
				echo "[INITIALIZING] Giving files Permissions..."
				chmod -R 775 ./

				echo "[DB2] Dropping Previous Tables..."
				./DB2Drop > /dev/null
				echo "[DB2] Dropping Previous Tables: Done"
				#Create tables
				echo "[DB2] Creating Tables..."
				./DB2Create > /dev/null
				echo "[DB2] Creating Tables: DONE"
				#Populating tables
				echo "[DB2] Populating Tables..."
				./DB2Populate > /dev/null
				echo "[DB2] Polulating Tables: DONE"
			fi
				./DB2MainMenu
			;;
		2)
			echo "----<ADVANCED OPTIONS>----"
			echo "1 | Create Tables"
			echo "2 | Populate Tables"
			echo "3 | Drop Tables"
			echo "4 | Start MainMenu"
			echo "5 | Install Scripts"
			echo "6 | Uninstall Scripts"
			echo -n "Please enter the number corresponding to the selection above or any other input to
return: " ; read option

			#Select and run user queries
			case $option in
				1)
					# Create Tables
					echo "[DB2] Creating Tables..."
					./DB2Create
					echo "[DB2] Creating Tables: DONE"
					;;
				2)
					# Create Tables
					echo "[DB2] Populating Tables..."
					./DB2Populate
					echo "[DB2] Polulating Tables: DONE"
					;;
				3)
					echo "[DB2] Dropping Tables..."
					./DB2Drop
					echo "[DB2] Dropping Tables: Done"
					;;
				4)
					./DB2MainMenu
					;;
				6)
					#Check if user wants to exit
					echo "Are you sure you want to Remove all files?"
					echo -n "Enter 'Yes' to Remove all files (Enter anything to return to DB2
OPTIONS): " ; read exit_message
					if [ "$exit_message" = "Yes" ]
					then
						SELECTION=0
					else
						SELECTION=-1
					fi
					;;
				5)
					#unzip all nessesary files and delete the zip file
					echo "[INITIALIZING] Unzipping Nesseary Files and removing."
					unzip -j -q -o PharmacyDatabase.zip 2> /dev/null

					#Give permmissions needed to the files
					echo "[INITIALIZING] Giving files Permissions..."
					chmod -R 775 ./
					;;
				*)
					#If user enters soemthing that is NOT part of the selection
					echo "ERROR: Please ONLY enter the selection numbers."
					echo " "
					SELECTION=-1
					;;
				esac
				;;
			0)
				echo "Are you sure you want to exit?"
				echo -n "Enter 'Yes' to exit (Enter anything to return to DB2 OPTIONS): " ; read exit_message
				if [ "$exit_message" = "Yes" ]
				then
					echo "[DB2] Goodye =D."
					sleep 1
					exit
				else
					SELECTION=-1
				fi
				;;
			*)
				#If user enters soemthing that is NOT part of the selection
				echo "ERROR: Please ONLY enter the selection numbers."
				echo " "
				SELECTION=-1
				;;
	esac
done

echo "[DB2] Removing and exiting."
echo "[DB2] Goobye =D."
sleep 1
#Remove all scripts from turing
rm -f ./\DB2*
```

6.7. Miscellaneous Scripts (DB2Run)

NOTE: The scripts within this section are used to start up the database and to perform all the actions necessary to initialize and run the database. These scripts do not contain any relevant SQL and is only meant to transfer and run the necessary files required by the pharmacy database.

```
#!/bin/bash

#echo "[INITIALIZING] Zipping files..."
# Compress all the nessary files to be sent make sure all files accounted for
#zip -r -q PharmacyDatabase.zip DB2Files/ > /dev/null

#Send to turing then connect to turing
echo "[INITIALIZING] Loading files into turning directory."
echo -n "[INITIALIZING] Please enter your SCS User name: " ; read user
scp PharmacyDatabase.zip Pharmacy $user@turing.acs.ryerson.ca:~/ > /dev/null
echo "[INITIALIZING] Files loaded."
echo "[INITIALIZING] Connecting to turing."
ssh $user@turing.acs.ryerson.ca './Pharmacy'
```