Pharmacy Database Design Project

By: Jaime Engl

Table of Contents

Table of Contents……………………………………….2
Executive Summary…………………………………3
Entity-Relationship Diagram……………………….4
Create Table Statements
People Table……………………………………5-6
Employees Table…………………………..………7
Technicians Table…………………………………..8
Pharmacists Table…………………………………..9
Customers Table…………………………………..10
Drugs Table……….....................…..…………11-12
Method_of_Delivery Table……………………..13
Prescriptions Table………………………….14-15
Accepted_Precript Table……………………..16-17
PreValidate_Prescript Table………………18-19
Fill_Prescript Table……………………...…20-21
PostValidate_Prescript Table……….........22-23
Sales Table……………………………………...24-25
Views……………………………………………………26
Stored Procedure……………………………………27
Reports……………………………………………28-29
Trigger……………………………………………........30
Security……………………………………………31-32
Implementation Notes…………………………..…33
Known Problems…………………………………….34
Future Enhancements……………………………..35

Executive Summary

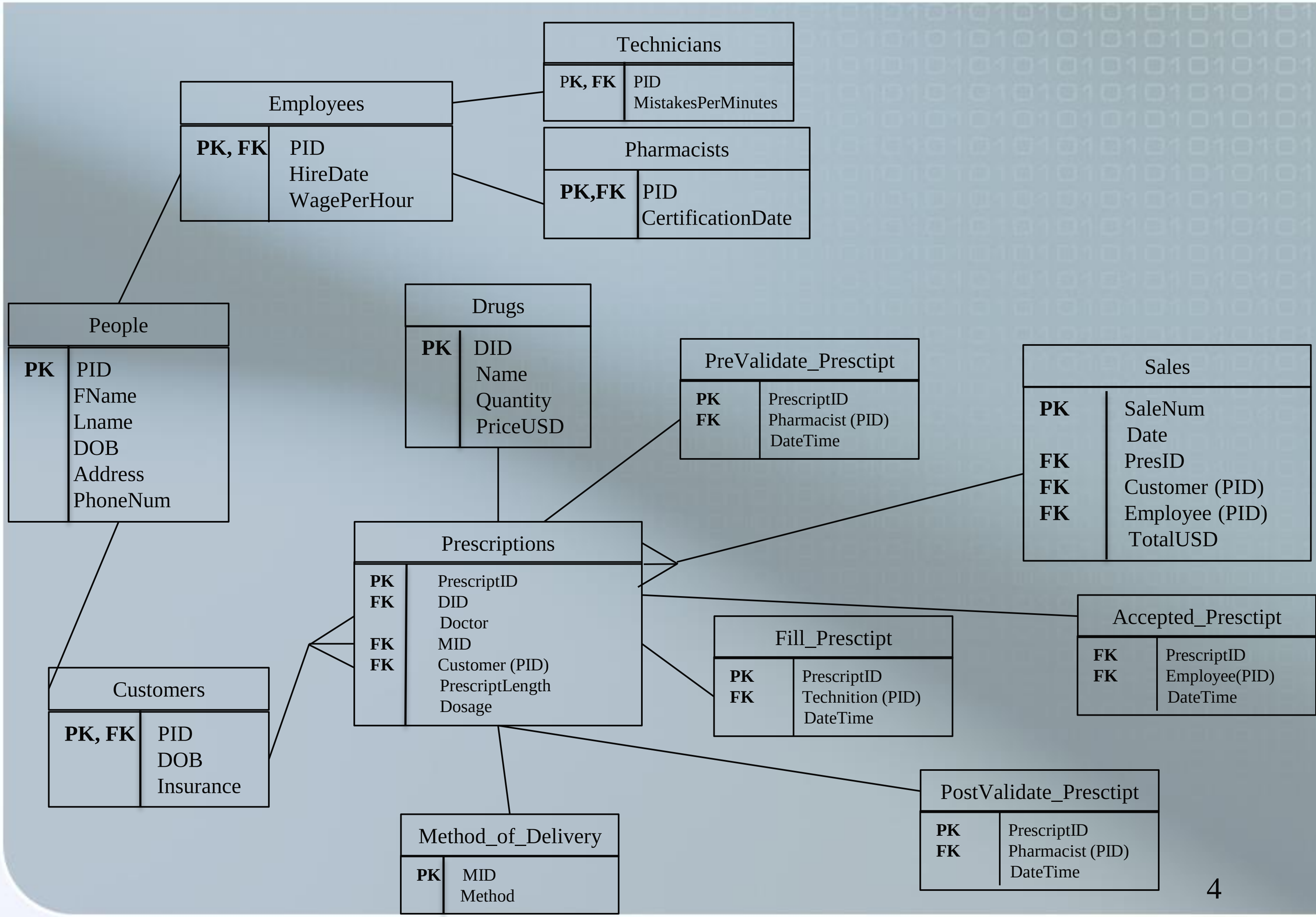
A pharmacy is a business that needs to handle a large amount of private information about numerous drugs, customers, and employees. In order to keep track of all this information, the pharmacy needs an effective and efficient database.

This database design organizes data such as people, employees, customers, drugs, prescriptions, and sales. In particular, it emphasizes the process in which a prescription is filled which is broken down into four steps. Each step has detailed data that can be recalled if there is an issue.

The presentation of the database will show an ER diagram, create statements for each table and snippets of test data, views, reports, stored procedures, triggers, and security. Known problems with the database will be established at the end along with possible future enhancements to the database.

This database was created for PostgreSQL 9.2.4.

Entity-Relationship Diagram



Technicians

PK, FK	PID MistakesPerMinutes

Employees

PK, FK	PID HireDate WagePerHour

Pharmacists

PK,FK	PID CertificationDate

People

PK	PID
	FName
	Lname
	DOB
	Address
	PhoneNum

Drugs

PK	DID
	Name
	Quantity
	PriceUSD

PreValidate_Presctipt

PK	PrescriptID
FK	Pharmacist (PID)
	DateTime

Sales

PK	SaleNum
	Date
FK	PresID
FK	Customer (PID)
FK	Employee (PID)
	TotalUSD

Prescriptions

PK	PrescriptID
FK	DID
	Doctor
FK	MID
FK	Customer (PID)
	PrescriptLength
	Dosage

Accepted_Presctipt

FK	PrescriptID
FK	Employee(PID)
	DateTime

Fill_Presctipt

PK	PrescriptID
FK	Technition (PID)
	DateTime

Customers

PK, FK	PID DOB Insurance

Method_of_Delivery

PK	MID
	Method

PostValidate_Presctipt

PK	PrescriptID
FK	Pharmacist (PID)
	DateTime

Create Table Statements

People table

- A people table makes sense here because an technician or pharmacist can also be a customer and to ensure that data is not duplicated.

```
CREATE TABLE IF NOT EXISTS people (
       PID         serial not null,
       Fname       text   not null,
       Lname       text   not null,
       DOB         date   not null,
       Address     text   not null,
       PhoneNum    text,
primary key(PID)
);
```

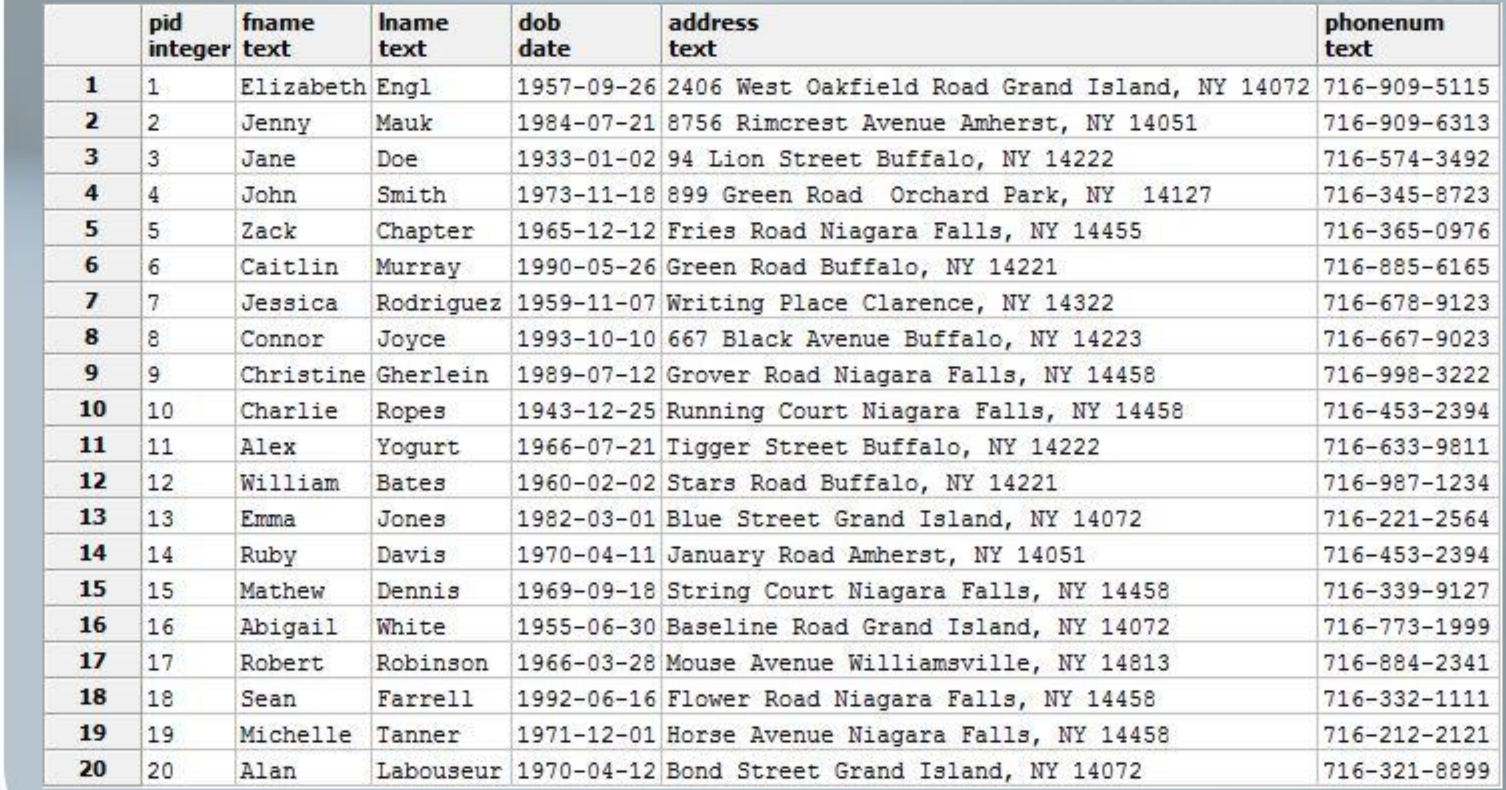
Functional Dependencies

PID → Fname, Lname, Address, PhoneNum

Create Table Statements

`People` table continued…

Test Data



	pid	fname	lname	dob	address	phonenum
	integer	text	text	date	text	text
1	1	Elizabeth	Engl	1957-09-26	2406 West Oakfield Road Grand Island, NY 14072	716-909-5115
2	2	Jenny	Mauk	1984-07-21	8756 Rimcrest Avenue Amherst, NY 14051	716-909-6313
3	3	Jane	Doe	1933-01-02	94 Lion Street Buffalo, NY 14222	716-574-3492
4	4	John	Smith	1973-11-18	899 Green Road Orchard Park, NY 14127	716-345-8723
5	5	Zack	Chapter	1965-12-12	Fries Road Niagara Falls, NY 14455	716-365-0976
6	6	Caitlin	Murray	1990-05-26	Green Road Buffalo, NY 14221	716-885-6165
7	7	Jessica	Rodriguez	1959-11-07	Writing Place Clarence, NY 14322	716-678-9123
8	8	Connor	Joyce	1993-10-10	667 Black Avenue Buffalo, NY 14223	716-667-9023
9	9	Christine	Gherlein	1989-07-12	Grover Road Niagara Falls, NY 14458	716-998-3222
10	10	Charlie	Ropes	1943-12-25	Running Court Niagara Falls, NY 14458	716-453-2394
11	11	Alex	Yogurt	1966-07-21	Tigger Street Buffalo, NY 14222	716-633-9811
12	12	William	Bates	1960-02-02	Stars Road Buffalo, NY 14221	716-987-1234
13	13	Emma	Jones	1982-03-01	Blue Street Grand Island, NY 14072	716-221-2564
14	14	Ruby	Davis	1970-04-11	January Road Amherst, NY 14051	716-453-2394
15	15	Mathew	Dennis	1969-09-18	String Court Niagara Falls, NY 14458	716-339-9127
16	16	Abigail	White	1955-06-30	Baseline Road Grand Island, NY 14072	716-773-1999
17	17	Robert	Robinson	1966-03-28	Mouse Avenue Williamsville, NY 14813	716-884-2341
18	18	Sean	Farrell	1992-06-16	Flower Road Niagara Falls, NY 14458	716-332-1111
19	19	Michelle	Tanner	1971-12-01	Horse Avenue Niagara Falls, NY 14458	716-212-2121
20	20	Alan	Labouseur	1970-04-12	Bond Street Grand Island, NY 14072	716-321-8899

Create Table Statements

Employees table

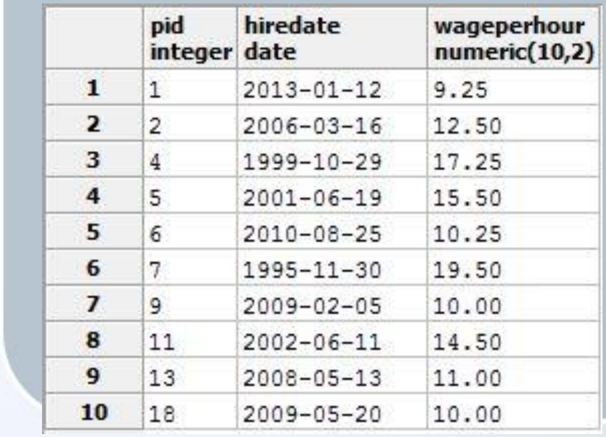
- The employees table keeps track of all employees both technicians and pharmacists and common information among people in that group including hire date and wage.

```
CREATE TABLE IF NOT EXISTS employees (
       PID          serial          not null references people(PID),
       HireDate     date            not null ,
       WagePerHour  numeric (10,2) not null,
primary key (PID)
);
```

Functional Dependencies

PID → HireDate, WagePerHour

Test Data



	pid integer	hiredate date	wageperhour numeric(10,2)
1	1	2013-01-12	9.25
2	2	2006-03-16	12.50
3	4	1999-10-29	17.25
4	5	2001-06-19	15.50
5	6	2010-08-25	10.25
6	7	1995-11-30	19.50
7	9	2009-02-05	10.00
8	11	2002-06-11	14.50
9	13	2008-05-13	11.00
10	18	2009-05-20	10.00

Create Table Statements

Technicians table

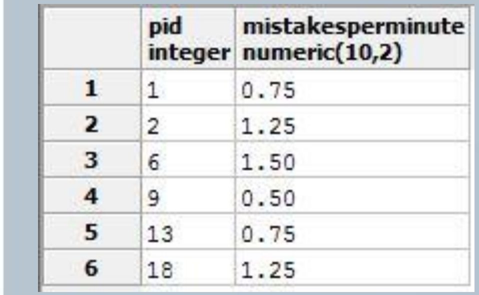
- The technicians table is for employees who are technicians only. The company keeps track of their mistakes per minute to see hour accurate they are.

```
CREATE TABLE IF NOT EXISTS technicians (
       PID               serial          not null references people(PID),
       MistakesPerMinute numeric (10,2)  not null,
primary key (PID)
);
```

Functional Dependencies

PID → SpeedPerMinute

Test Data



	pid integer	mistakesperminute numeric(10,2)
1	1	0.75
2	2	1.25
3	6	1.50
4	9	0.50
5	13	0.75
6	18	1.25

Create Table Statements

Pharmacists table

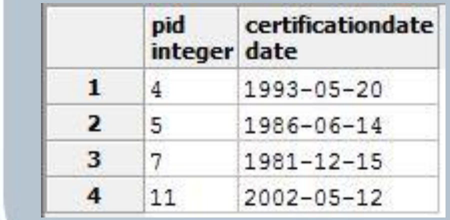
- The pharmacists table is for employees who are a certified pharmacist. It keeps tracks of when they were certified to see information like who has the most experience.

```
CREATE TABLE IF NOT EXISTS pharmacists (
       PID                 serial not null references people(PID),
       CertificationDate   date   not null,
primary key (PID)
);
```

Functional Dependencies

PID → CertificationDate

Test Data



	pid integer	certificationdate date
1	4	1993-05-20
2	5	1986-06-14
3	7	1981-12-15
4	11	2002-05-12

Create Table Statements

Customers table

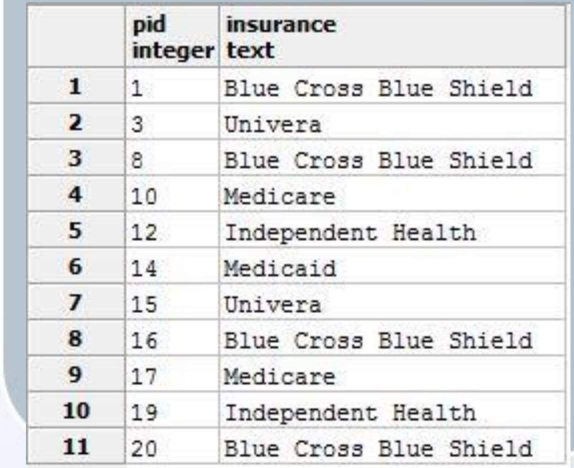
- The customers table keeps track of what insurance each customer has. Keep in mind that an employee can also be a customer.

```
CREATE TABLE IF NOT EXISTS customers (
       PID        serial not null references people(PID),
       Insurance  text   not null,
primary key (PID)
);
```

Functional Dependencies

PID → Insurance

Test Data



	pid integer	insurance text
1	1	Blue Cross Blue Shield
2	3	Univera
3	8	Blue Cross Blue Shield
4	10	Medicare
5	12	Independent Health
6	14	Medicaid
7	15	Univera
8	16	Blue Cross Blue Shield
9	17	Medicare
10	19	Independent Health
11	20	Blue Cross Blue Shield

Create Table Statements

Drugs table

- The drugs table organizes all of the different kinds of drugs and all their relevant information such as name, how much inventory (in units) the pharmacy has, and how much it costs.

```
CREATE TABLE IF NOT EXISTS drugs (
       DID       serial        not null,
       Name      text          not null,
       Quantity  integer       not null,
       PriceUSD  numeric(10,2) not null,
primary key (DID)
);
```

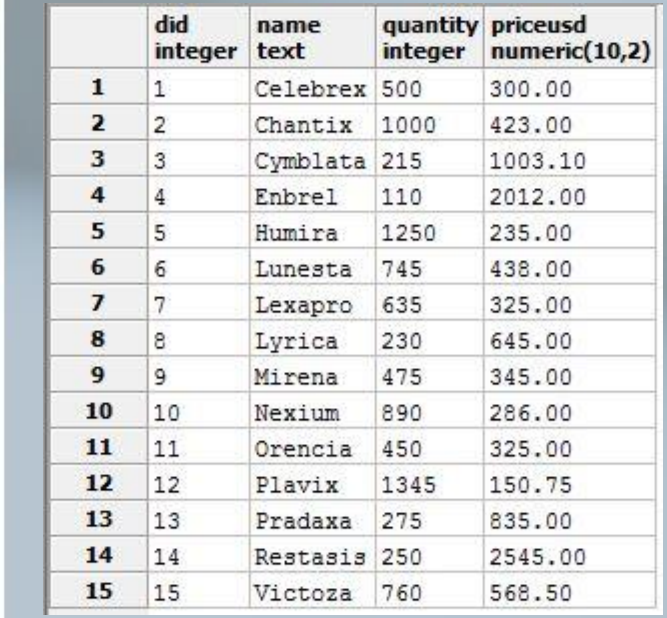
Functional Dependencies

DID → Name, Quantity, PriceUSD

Create Table Statements

`Drugs` table continued…

Test Data



	did integer	name text	quantity integer	priceusd numeric(10,2)
1	1	Celebrex	500	300.00
2	2	Chantix	1000	423.00
3	3	Cymblata	215	1003.10
4	4	Enbrel	110	2012.00
5	5	Humira	1250	235.00
6	6	Lunesta	745	438.00
7	7	Lexapro	635	325.00
8	8	Lyrica	230	645.00
9	9	Mirena	475	345.00
10	10	Nexium	890	286.00
11	11	Orencia	450	325.00
12	12	Plavix	1345	150.75
13	13	Pradaxa	275	835.00
14	14	Restasis	250	2545.00
15	15	Victoza	760	568.50

Create Table Statements

Method_of_Delivery table

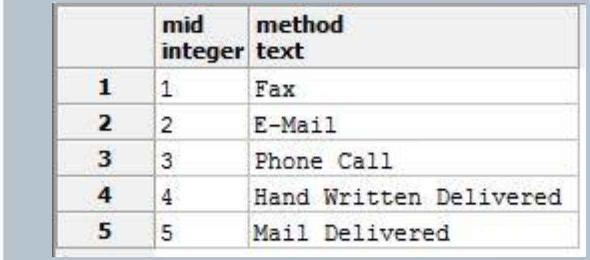
- The method of delivery table lists the five possible methods that the prescription was delivered.

```
CREATE TABLE IF NOT EXISTS method_of_delivery (
       MID        serial    not null,
       Method     text      not null,
primary key (MID)
);
```

Functional Dependencies

MID → Method

Test Data



	mid integer	method text
1	1	Fax
2	2	E-Mail
3	3	Phone Call
4	4	Hand Written Delivered
5	5	Mail Delivered

Create Table Statements

Prescriptions table

- The prescriptions table keeps track of all of the prescriptions that this pharmacy fills. It records information such as which drug, who prescribed it, who it's for, how long the prescription last, and the dosage.

```
CREATE TABLE IF NOT EXISTS prescriptions (
       PrescriptID      serial   not null,
       MID              serial   not null references method_of_delivery(MID),
       DID              serial   not null references drugs(DID),
       Doctor           text     not null,
       Customer         serial   not null references people(PID),
       PrescriptLength text     not null,
       Dosage           text     not null,
primary key (PrescriptID)
);
```

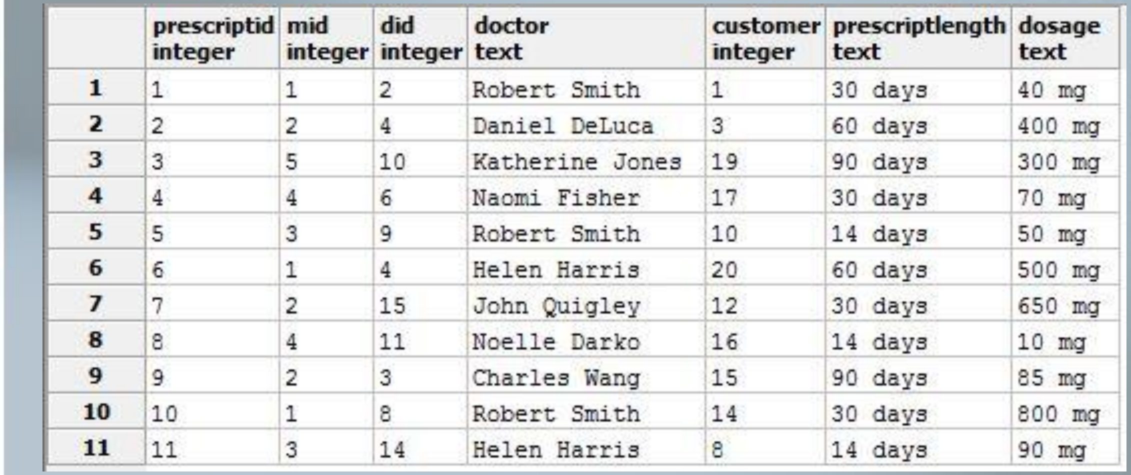
Functional Dependencies

PresID → DID, Customer, Employee, Pharmacist, PresLength

Create Table Statements

`Prescriptions` table continued…

Test Data



	prescriptid integer	mid integer	did integer	doctor text	customer integer	prescriptlength text	dosage text
1	1	1	2	Robert Smith	1	30 days	40 mg
2	2	2	4	Daniel DeLuca	3	60 days	400 mg
3	3	5	10	Katherine Jones	19	90 days	300 mg
4	4	4	6	Naomi Fisher	17	30 days	70 mg
5	5	3	9	Robert Smith	10	14 days	50 mg
6	6	1	4	Helen Harris	20	60 days	500 mg
7	7	2	15	John Quigley	12	30 days	650 mg
8	8	4	11	Noelle Darko	16	14 days	10 mg
9	9	2	3	Charles Wang	15	90 days	85 mg
10	10	1	8	Robert Smith	14	30 days	800 mg
11	11	3	14	Helen Harris	8	14 days	90 mg

Create Table Statements

Accepted_Prescript table

- Accepted_Prescript is the first of four tables related to the prescription process. Before a prescription can be Pre-Validated by a pharmacist it must be accepted by either a technician or pharmacist. This table holds information like which prescription was accepted, by whom it was accepted and the date and time it was accepted.

```
CREATE TABLE IF NOT EXISTS accepted_prescript (
      PrescriptId serial    not null references prescriptions(PrescriptID),
      Employee    serial    not null references people(PID),
      DateTime    timestamp not null,
primary key (PrescriptID)
);
```

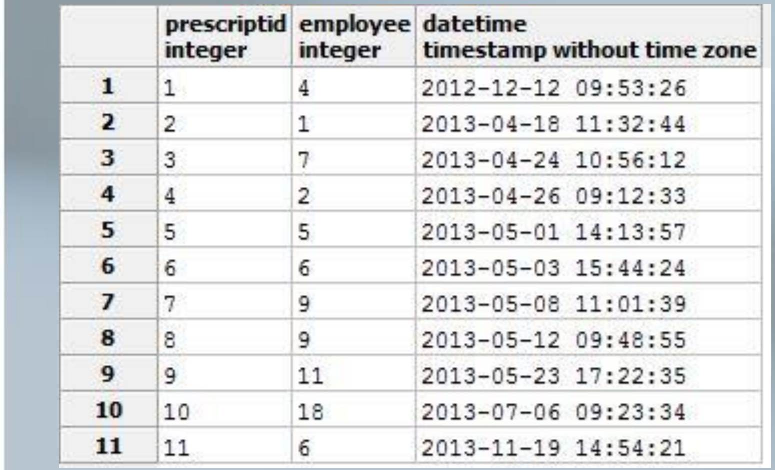
Functional Dependencies

PrescriptID → Employee, DateTime

Create Table Statements

`Accepted_Prescript` table continued…

Test Data



	prescriptid integer	employee integer	datetime timestamp without time zone
1	1	4	2012-12-12 09:53:26
2	2	1	2013-04-18 11:32:44
3	3	7	2013-04-24 10:56:12
4	4	2	2013-04-26 09:12:33
5	5	5	2013-05-01 14:13:57
6	6	6	2013-05-03 15:44:24
7	7	9	2013-05-08 11:01:39
8	8	9	2013-05-12 09:48:55
9	9	11	2013-05-23 17:22:35
10	10	18	2013-07-06 09:23:34
11	11	6	2013-11-19 14:54:21

Create Table Statements

PreValidate_Prescript table

- PreValidate_Prescript is the second part of the process. It can only be done by the pharmacist. It hold the same kind of information as in Accepted_Prescript.

```
CREATE TABLE IF NOT EXISTS prevalidate_prescript (
       PrescriptId serial    not null references prescriptions(PrescriptID),
       Pharmacist  serial    not null references people(PID),
       DateTime    timestamp not null,
primary key (PrescriptID)
);
```

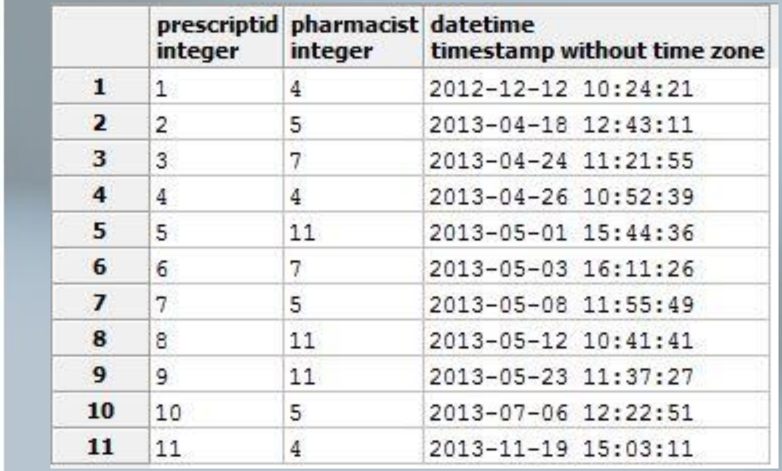
Functional Dependencies

PrescriptID → Pharmacist, DateTime

Create Table Statements

`PreValidate_Prescript` table continued…

Test Data



	prescriptid integer	pharmacist integer	datetime timestamp without time zone
1	1	4	2012-12-12 10:24:21
2	2	5	2013-04-18 12:43:11
3	3	7	2013-04-24 11:21:55
4	4	4	2013-04-26 10:52:39
5	5	11	2013-05-01 15:44:36
6	6	7	2013-05-03 16:11:26
7	7	5	2013-05-08 11:55:49
8	8	11	2013-05-12 10:41:41
9	9	11	2013-05-23 11:37:27
10	10	5	2013-07-06 12:22:51
11	11	4	2013-11-19 15:03:11

Create Table Statements

Fill_Prescript table

- Fill_Prescript is the third part of the process. The prescription is filled by a technician. This table holds the same kind of data as in the last two tables.

```
CREATE TABLE IF NOT EXISTS fill_prescript (
       PrescriptId   serial    not null references prescriptions(PrescriptID),
       Technician    serial    not null references people(PID),
       DateTime      timestamp not null,
primary key (PrescriptID)
);
```

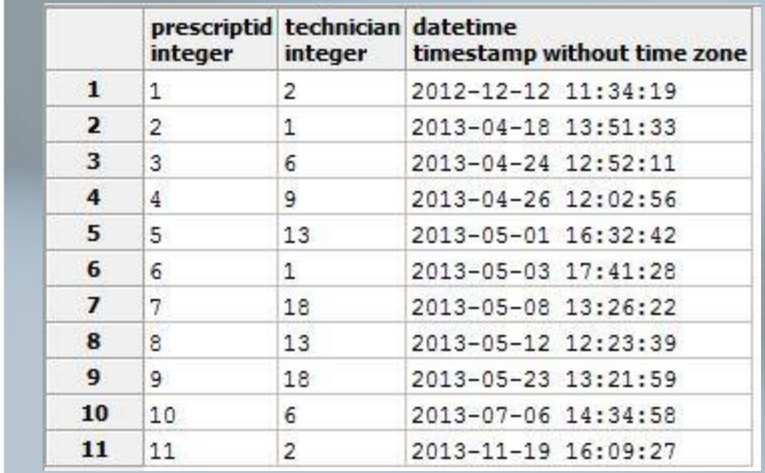
Functional Dependencies

PrescriptID → Technician, DateTime

Create Table Statements

`Fill_Prescript` table continued…

Test Data



	prescriptid integer	technician integer	datetime timestamp without time zone
1	1	2	2012-12-12 11:34:19
2	2	1	2013-04-18 13:51:33
3	3	6	2013-04-24 12:52:11
4	4	9	2013-04-26 12:02:56
5	5	13	2013-05-01 16:32:42
6	6	1	2013-05-03 17:41:28
7	7	18	2013-05-08 13:26:22
8	8	13	2013-05-12 12:23:39
9	9	18	2013-05-23 13:21:59
10	10	6	2013-07-06 14:34:58
11	11	2	2013-11-19 16:09:27

Create Table Statements

PostValidate_Prescript table

- PostValidate_Prescript is final step in the prescription process. It can only be done by a pharmacist. It holds the same kind of data as in the previous three tables.

```
CREATE TABLE IF NOT EXISTS postvalidate_prescript (
       PrescriptId   serial    not null references prescriptions(PrescriptID),
       Pharmacist    serial    not null references people(PID),
       DateTime      timestamp not null,
primary key (PrescriptID)
);
```

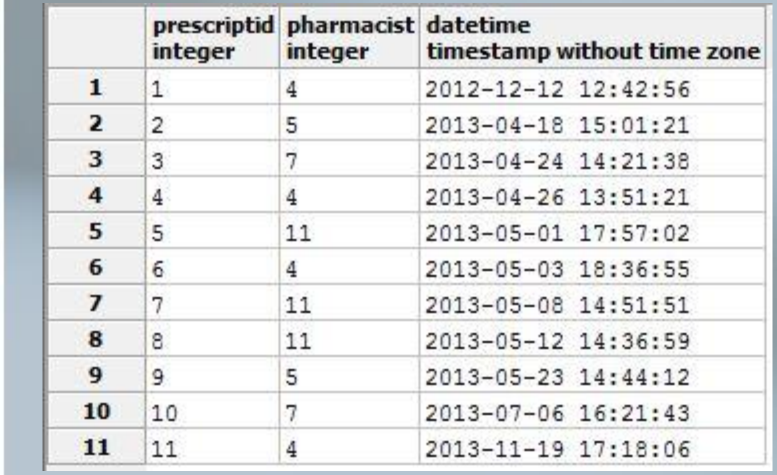
Functional Dependencies

PrescriptID → Pharmacist, DateTime

Create Table Statements

`PostValidate_Prescript` table continued…

Test Data



	prescriptid integer	pharmacist integer	datetime timestamp without time zone
1	1	4	2012-12-12 12:42:56
2	2	5	2013-04-18 15:01:21
3	3	7	2013-04-24 14:21:38
4	4	4	2013-04-26 13:51:21
5	5	11	2013-05-01 17:57:02
6	6	4	2013-05-03 18:36:55
7	7	11	2013-05-08 14:51:51
8	8	11	2013-05-12 14:36:59
9	9	5	2013-05-23 14:44:12
10	10	7	2013-07-06 16:21:43
11	11	4	2013-11-19 17:18:06

Create Table Statements

Sales table

- The Sales table keeps track of all the sales of prescriptions that occur at the pharmacy. The data that can be found in this table is the Sale Number, the date, the prescriptions ID, the customer, the employee who sold it, and the total.

```
CREATE TABLE IF NOT EXISTS sales (
      SaleNum     serial        not null,
      Date        timestamp     not null,
      PrescriptID serial        not null references prescriptions(PresID),
      Customer    serial        not null references people(PID),
      Employee    serial        not null references people(PID),
      TotalUSD    numeric(10,2) not null,
primary key (SaleNum)
);
```

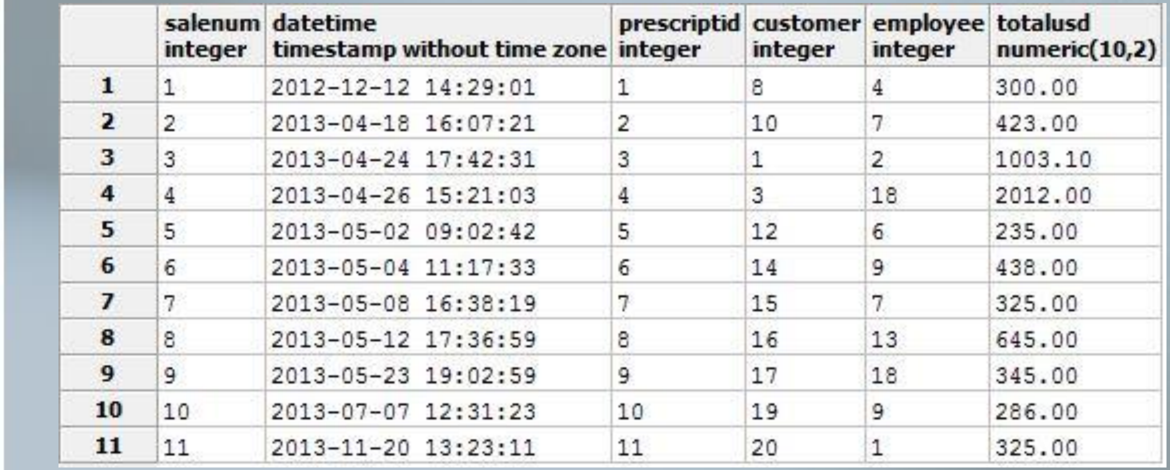
Functional Dependencies

SaleNum → Date, PresID, Customer, Employee, TotalUSD

Create Table Statements

`Sales` table continued…

Test Data



	salenum	datetime	prescriptid	customer	employee	totalusd
	integer	timestamp without time zone	integer	integer	integer	numeric(10,2)
1	1	2012-12-12 14:29:01	1	8	4	300.00
2	2	2013-04-18 16:07:21	2	10	7	423.00
3	3	2013-04-24 17:42:31	3	1	2	1003.10
4	4	2013-04-26 15:21:03	4	3	18	2012.00
5	5	2013-05-02 09:02:42	5	12	6	235.00
6	6	2013-05-04 11:17:33	6	14	9	438.00
7	7	2013-05-08 16:38:19	7	15	7	325.00
8	8	2013-05-12 17:36:59	8	16	13	645.00
9	9	2013-05-23 19:02:59	9	17	18	345.00
10	10	2013-07-07 12:31:23	10	19	9	286.00
11	11	2013-11-20 13:23:11	11	20	1	325.00

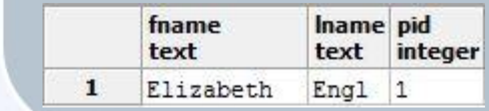
Views

employees_who_are_customers view

- This view allows you to see which of the pharmacy’s employees are also customers. It selects the first and last names as well as the PID. This is could be helpful to the company to see if their employees really use their services or not.

```
create view employees_who_are_customers
as
select p.Fname, p.Lname, p.pid
from people p
where p.pid = (select e.pid
               from employees e,
               customers c
               where e.pid = c.pid)
Order by pid;
```

Test Data Example



	fname text	lname text	pid integer
1	Elizabeth	Engl	1

Stored Procedures

Add_People Stored Procedure

- This stored procedure makes it easy to add a new person to the people table. You can add their first and last name as well as their Date of Birth, Address, and Phone Number.

```
CREATE OR REPLACE FUNCTION add_people("Fname" text, "Lname" text, "DOB" date,
"Address" text, "PhoneNum" text)
   RETURNS void AS
$BODY$BEGIN
   INSERT INTO people VALUES (Fname, Lname, DOB, Address, PhoneNum);
END$BODY$
   LANGUAGE plpgsql;
```

Reports

All Prescriptions from a Certain Time Period

- This report will generate all of the prescriptions (and the information that you choose to select) that occurred within a certain period. It could be a year, month, or day. It can be done on the `accepted_prescript` table, the `prevalidate_prescript` table, the `fill_prescript` table, the `postvalidate_prescript` table, and the `sales` table.

```
select *
from --------------
where extract(----- from DateTime) = ‘----';
```

Use Example:

```
select prescriptid, pharmacist, datetime
from postvalidate_prescript
where extract(year from DateTime) = '2013';
```

Reports

How Many Prescriptions Has Each Employee Sold

- This report shows the PID of the employee and the number of prescriptions that they have dealt with in a given time period. It can be used on any of the `*****_prescript` tables as well as `sales`.

```
select count(*) as NumberofPrescriptions, employee
from -----
where extract(---- from DateTime) = ‘----'
  and employee = --
group by employee;
```

Use Example:

```
select count(*) as NumberofPrescriptions, employee
from sales
where extract(year from DateTime) = '2013'
  and employee = 18
group by employee;
```

Trigger

Update_Quantity_Trigger

- This trigger updates the quantity in the drugs table when a prescription is filled.

```
CREATE OR REPLACE FUNCTION update_quantity_function(“Quantity” integer,
“DID” integer)
  RETURNS trigger AS
$BODY$
BEGIN
  UPDATE drugs SET Quantity = Quantity – 1 WHERE DID = DID;
END
$BODY$
LANGUAGE plpgsql

CREATE TRIGGER update_quantity_trigger
AFTER INSERT
ON fill_prescript
FOR EACH STATEMENT
EXECUTE PROCEDURE update_quanity_function
```

Security

Tenchnician role

- A technician has some privileges but they are limited unlike those of the pharmacists. This will ensure that technicians can only insert, update, and delete on the correct tables.

```
REVOKE ALL PRIVILEGES ON people FROM technician;
REVOKE ALL PRIVILEGES ON employees FROM technician;
REVOKE ALL PRIVILEGES ON customers FROM technician;
REVOKE ALL PRIVILEGES ON technicians FROM technician;
REVOKE ALL PRIVILEGES ON pharmacists FROM technician;
REVOKE ALL PRIVILEGES ON drugs FROM technician;
REVOKE ALL PRIVILEGES ON prescriptions FROM technician;
REVOKE ALL PRIVILEGES ON method_of_delivery FROM technician;
REVOKE ALL PRIVILEGES ON accepted_prescript FROM technician;
REVOKE ALL PRIVILEGES ON prevalidate_prescript FROM technician;
REVOKE ALL PRIVILEGES ON fill_prescript FROM technician;
REVOKE ALL PRIVILEGES ON postvalidate_prescript FROM technician;
REVOKE ALL PRIVILEGES ON sales FROM technician;

GRANT INSERT, SELECT ON people FROM technician;
GRANT SELECT ON employees FROM technician;
GRANT INSERT, UPDATE, SELECT ON customers FROM technician;
GRANT SELECT ON technicians FROM technician;
GRANT SELECT ON pharmacists FROM technician;
GRANT INSERT, SELECT ON drugs FROM technician;
GRANT INSERT, SELECT ON prescriptions FROM technician;
GRANT INSERT, UPDATE, SELECT ON method_of_delivery FROM technician;
GRANT INSERT, SELECT ON accepted_prescript FROM technician;
GRANT SELECT ON prevalidate_prescipt FROM technician;
GRANT INSERT, SELECT ON fill_prescript FROM technician;
GRANT SELECT ON postvalidate_prescript FROM technician;
GRANT INSERT, SELECT ON sales FROM technician;
```

Security

Pharmacist role

- Pharmacists should have more control over the database. For example: anything that has to do with the PreValidate_prescript and PostValidate_prescipt tables there has to be tight security. The only people who should be able to insert into those tables are pharmacists because they are the only ones who can prevalidate or postvalidate a prescription.

```
REVOKE ALL PRIVILEGES ON people FROM pharmacist;
REVOKE ALL PRIVILEGES ON employees FROM pharmacist;
REVOKE ALL PRIVILEGES ON customers FROM pharmacist;
REVOKE ALL PRIVILEGES ON technicians FROM pharmacist;
REVOKE ALL PRIVILEGES ON pharmacists FROM pharmacist;
REVOKE ALL PRIVILEGES ON drugs FROM pharmacist;
REVOKE ALL PRIVILEGES ON prescriptions FROM pharmacist;
REVOKE ALL PRIVILEGES ON method_of_delivery FROM pharmacist;
REVOKE ALL PRIVILEGES ON accepted_prescript FROM pharmacist;
REVOKE ALL PRIVILEGES ON prevalidate_prescript FROM pharmacist;
REVOKE ALL PRIVILEGES ON fill_prescript FROM pharmacist;
REVOKE ALL PRIVILEGES ON postvalidate_prescript FROM pharmacist;
REVOKE ALL PRIVILEGES ON sales FROM pharmacist;

GRANT INSERT, UPDATE, SELECT ON people FROM pharmacist;
GRANT INSERT, SELECT ON employees FROM pharmacist;
GRANT INSERT, UPDATE, SELECT ON customers FROM pharmacist;
GRANT INSERT, UPDATE, SELECT ON technicians FROM pharmacist;
GRANT INSERT, SELECT ON pharmacists FROM pharmacist;
GRANT INSERT, UPTATE, SELECT ON drugs FROM pharmacist;
GRANT INSERT, UPDATE, SELECT ON prescriptions FROM pharmacist;
GRANT INSERT, UPDATE, SELECT ON method_of_delivery FROM pharmacist;
GRANT INSERT, UPDATE, SELECT ON accepted_prescript FROM pharmacist;
GRANT INSERT, UPDATE, SELECT ON prevalidate_prescipt FROM pharmacist;
GRANT INSERT, UPDATE, SELECT ON fill_prescript FROM pharmacist;
GRANT INSERT, UPDATE, SELECT ON postvalidate_prescript FROM pharmacist;
GRANT INSERT, UPDATE, SELECT ON sales FROM pharmacist;
```

Implementation Notes

- The implementation of this database would take a lot of time. In order to input all of the data of all customers, employees, and drugs. There would have to be a time period in which time would spent just adding all of the previous data from the company. Teaching the employees how to use the database will take a few days but will be pretty simple.

Known Problems

- It’s difficult to have a quantity column under Drugs because pharmacies dispense many different kinds of drugs, creams, etc. It is almost impossible to find a common unit of measure. I chose to use “units” meaning whatever package they come in. However, this will lead to inventory problems in the future, especially when the pharmacy needs to order more of product.

Future Enhancements

- In the future, I would like to make a more complete inventory system that makes it easy to keep all the drugs together and manage the quantities.
- Also, in the future it would nice to have a customer loyalty program where people who have been buying from the pharmacy for a long time can receive some rewards.
- In addition, it would helpful to have either tables in this database or another database that works along side this one, to check which drugs cannot interact with other drugs, so that the pharmacists don’t have to do this by hand all the time.