

---

FUNDAMENTOS DE BASES DE DATOS

# Reporte de la solución

PROYECTO FINAL

420003117 Jose Luis Onofre Franco

417024956 Oscar Andres Rosas Hernandez

Enero 2020

# Índice general

<b>I</b>	<b>¿Cómo llegamos a la solución?</b>	<b>4</b>
<b>1.</b>	<b>Hablemos del problema</b>	<b>5</b>
<b>2.</b>	<b>Modelo Entidad Relación y los requerimientos</b>	<b>6</b>
2.1.	Entidades creadas . . . . .	6
2.1.1.	Sobre personas . . . . .	6
2.1.2.	Sobre establecimientos . . . . .	6
2.1.3.	Sobre empleados y los diferentes tipos . . . . .	7
2.1.4.	Sobre los medicamentos y consultas . . . . .	7
2.2.	Relaciones creadas . . . . .	8
2.2.1.	Sobre personas . . . . .	8
2.2.2.	Sobre establecimientos . . . . .	8
2.2.3.	Sobre empleados y los diferentes tipos . . . . .	8
2.2.4.	Sobre los medicamentos y consultas . . . . .	8
2.3.	Diagrama final . . . . .	9
<b>3.</b>	<b>Modelo Relacional y los requerimientos</b>	<b>10</b>
3.1.	Relaciones creadas . . . . .	10
3.1.1.	Sobre personas y empleados . . . . .	10
3.1.2.	Sobre establecimientos, los medicamentos y consultas . . . . .	11
3.2.	Diagrama final . . . . .	11
<b>4.</b>	<b>Normalización</b>	<b>12</b>
4.1.	¿Que es la normalización? . . . . .	12

4.1.1. Definición . . . . .	12
4.2. Primera Forma Normal . . . . .	12
4.3. Segunda Forma Normal . . . . .	13
4.3.1. Dependencia Funcional y Determinantes . . . . .	13
4.3.2. Definición . . . . .	14
4.3.3. Nuestro caso . . . . .	14
4.4. Tercera Forma Normal . . . . .	15
4.4.1. Nuestro caso . . . . .	15
4.4.2. Nuestro caso . . . . .	15
<b>5. Implementación es SQL</b>	<b>17</b>
5.1. DDL . . . . .	17
5.2. Población de datos para pruebas . . . . .	17
5.3. Triggers . . . . .	17
5.4. Funciones . . . . .	17
 <b>II Demostración de nuestra solución</b>	 <b>18</b>
 <b>6. Queries</b>	 <b>19</b>
6.1. Comparemos ventas por sucursal . . . . .	19
6.2. Número de consultas por doctor . . . . .	20
6.3. Número de consultas por mes del año . . . . .	21
6.4. Productos más vendidos . . . . .	22
6.5. Consultas por especialidad . . . . .	23
6.6. Mayor ingreso por especialidad . . . . .	24
6.7. Balance promedio entre clientes vs pacientes . . . . .	25
6.8. Cajeros mas productivos . . . . .	26
6.9. Ventas por hora . . . . .	27
6.10. Cuentas de doctores que generan una proxima cita . . . . .	28
6.11. Total de medicamentos en receta por especialidad cuyo precio es mayor a 500 . . . . .	29

6.12. Posible ingreso por farmacos que vengan por receta . . . . .	30
6.13. Demografica de nuestros pacientes . . . . .	31
6.14. La ubicación por estado que genera más ingreso a través de compras por clientes . . . . .	32
6.15. Ventas de fármacos que necesitan receta . . . . .	33

## Parte I

¿Cómo llegamos a la solución?

# Capítulo 1

## Hablemos del problema

La cadena mexicana de farmacias “Doctor Ahorro” ha comenzado un plan de expansión dentro de la República Mexicana y Centroamérica, gracias al éxito obtenido en Ciudad de México, Guadalajara y Monterrey.

La cadena está orientada principalmente al mercado que representa la población de bajos recursos.

Actualmente toda la información que se procesa en la cadena se hace a través de un sistema que ha presentado múltiples fallas en los últimos 3 años, llegando incluso a perder información de los clientes, productos y ventas.

Debido a lo anterior, el CEO de la cadena decidió recurrir a la empresa Computólogos A.C. para obtener una herramienta confiable y eficaz.

## Capítulo 2

# Modelo Entidad Relación y los requerimientos

### 2.1. Entidades creadas

#### 2.1.1. Sobre personas

En nuestra base de datos trabajaremos con personas, y muchos roles que estas mismas juegan, personas como clientes, doctores, personal (y sus subdivisiones como repartidor, cajero y gerente).

Primeramente decidimos crear una entidad base **person**, en ella colocamos los datos que consideramos obvios guardar y aquellos incluidos en los requerimientos. Citamos de manera textual de dichos requerimientos: nombre completo, género, dirección, teléfono, correo electrónico.

A estos agregamos un numero de telefono asociado a esa persona, el curp como atributo identificador y finalmente usamos un entidad **zip** y **state** para evitar posteriormente la redundancia de información al momento de guardar la dirección.

Ahora hablaremos de los empleados de la empresa, un empleado (**employee**) es una especialización de **person** donde guardaremos también dos datos adicionales y citando a los requerimientos: los mismos datos que un cliente, además de su rfc, número de seguro social, sucursal en la que trabajan.

#### 2.1.2. Sobre establecimientos

Justamente para poder cumplir el último requerimiento presentado creamos una entidad **establishment** que estará asociada al empleado.

Siguiendo los requerimientos el siguiente punto son los establecimientos, lo cual nos lleva perfectamente a hablar un poco más de **establishment** en ella guardaremos los telefonos asociados a ese lugar (notar que puede haber más de uno), la dirección (que será guardará de la misma) manera que hacíamos antes con **person** y citando a los requerimientos: Establecimientos o Sucursales: dirección, teléfonos, responsable, estado y ciudad.

Notar que para lograr cumplir ese último requerimiento, donde tenemos que ser capaces de poder saber quien es el responsable o gerente del lugar tenemos que hablar de las especializaciones que hicimos de los empleados.

### 2.1.3. Sobre empleados y los diferentes tipos

Decidimos crear varias entidades que especializan de **employee** (de hecho es una especialización total), estas son **cleaner** (no hay mucho que agregar), **general** (para todos los empleados generales de la tienda), **delivery** (con los requerimientos propuestos de guardar su licencia e información del vehículo que maneja), **cashier** (que no tiene ninguna atributo especial pero que si esta involucrada en una relación importante de la que hablaremos en la siguiente sección), **doctor** (que tiene su licencia asociada) y **manager** que es el encargado de manejar un **establishment**.

También hablaremos de que un doctor esta relacionado con otra entidad muy importante: **specialty** esta entidad contiene el nombre de la misma así como el costo actual de una consulta en dicha especialidad.

### 2.1.4. Sobre los medicamentos y consultas

Ahora hablemos de las medicinas, para ellas creamos una entidad **drug** que contiene basado en los requerimientos: marca, nombre, precio, ingrediente(s) activo(s), si su venta requiere receta o es de venta al público y presentación (p.e. cápsula, jarabe, comprimido, etcétera).

Es importante hablar de que no guardamos de manera directa el ingrediente activo, lo que hicimos ese crear otra entidad llamada **substance**, también agregamos un atributo a **drug** que nos cuenta si es que cierto medicamento necesita una prescripción.

Relacionada a esto están las recetas, están representadas por la entidad **consultation** en ellas tenemos el turno de la consulta que la generó (bueno, para ser precisos esta dentro de una relación con la entidad **consultation**), tenemos una relación que nos permite acceder a los datos del paciente, al médico que la emitió, unido a estos esta la entidad **prescription** que contiene una firma digital, indicaciones adicionales, la fecha de la siguiente cita y el identificador de la misma llamada invoice.

Finalmente hablaremos de la compra de productos, para esto creamos la entidad



**purchase** donde guardaremos el método de compra, el tiempo y la hora y una forma de acceder (mediante las relaciones) a el establecimiento, los productos el vendedor y las recetas asociadas.

## 2.2. Relaciones creadas

Las relaciones dentro de esta solución son en general muy sencillas así que esta sección será mucho más corta que la anterior.

### 2.2.1. Sobre personas

La relación **lives** que une a una **person** y a un **zip**, de manera similar entre **zip** y **state** mediante la relación **belongs**.

### 2.2.2. Sobre establecimientos

De manera similar tenemos la relación **is\_ubicated** que conecta con **zip**.

También tenemos la relación **works** que nos dice en que **establishment** trabaja algún empleado. Tenemos la relación **manages** que nos dice para un **establishment** cual es el gerente.

### 2.2.3. Sobre empleados y los diferentes tipos

La gran razón por la cual dividimos los empleados fue justamente para poder tener relaciones que no afectarán que todo tipo de empleados sino solo a un tipo.

Por ejemplo tenemos la relación **makes** que une a un cajero con una compra, otro ejemplo es la relación **offer** que nos dice de que especialidad se está dando una consulta.

### 2.2.4. Sobre los medicamentos y consultas

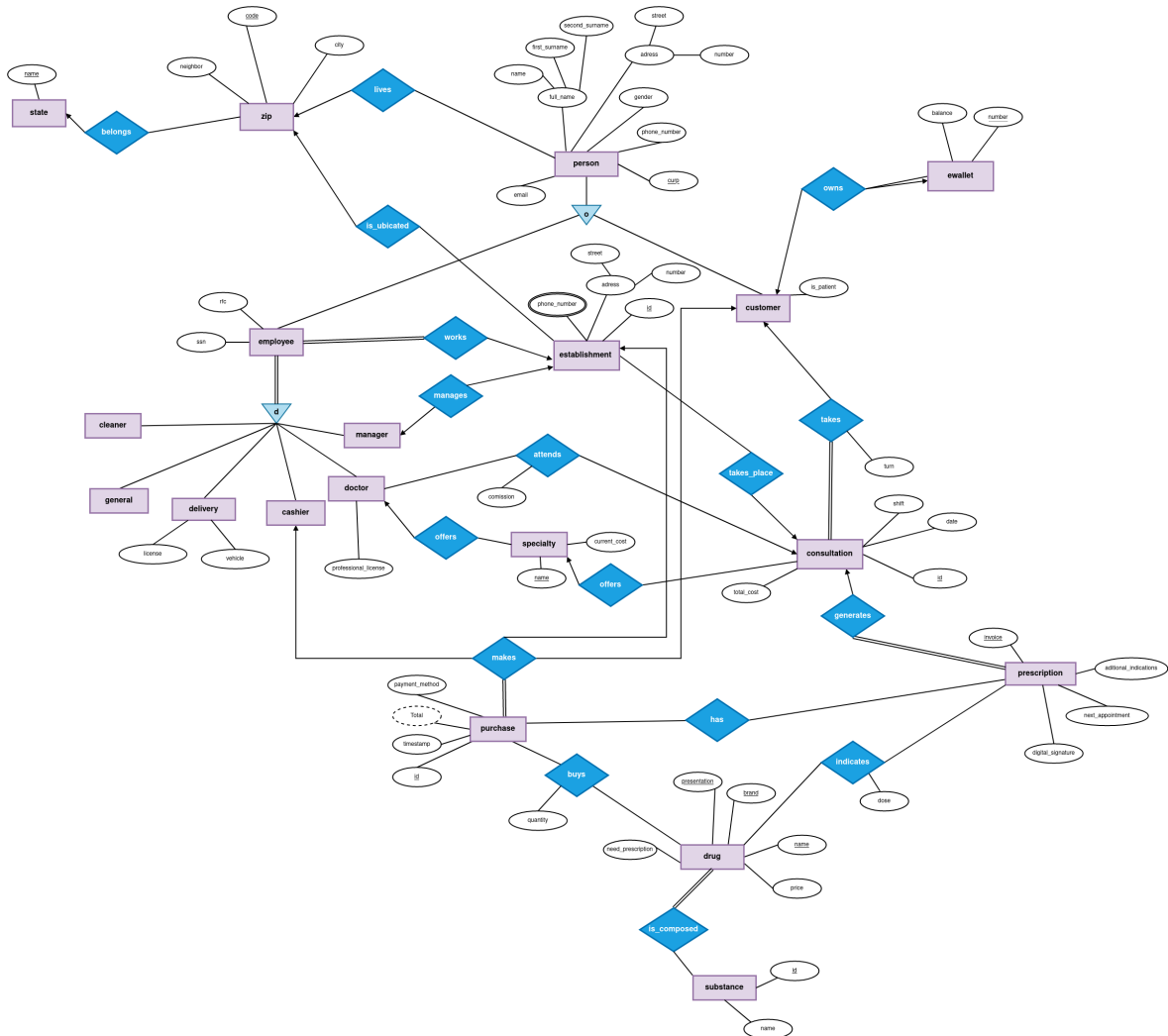
Tenemos relaciones como **buys** que nos dice para una compra cuáles medicamentos compro, **has** que une una compra con una prescripción.

También tenemos la relación **is\_composed** que nos une un medicamento con los ingredientes activos. y la relación **indicates** que nos dice la dosis de un medicamento prescrito en una consulta.

Relacionado a esto tenemos **generates** que nos da las prescripciones que genera una consulta, **attends** que nos dice que doctor está tomando cierta conducta y la relación

takes\_place que nos dice el establecimiento en el que ocurre una consulta.

## 2.3. Diagrama final



# Capítulo 3

## Modelo Relacional y los requerimientos

### 3.1. Relaciones creadas

Algo importante a destacar es que en esta sección solo hablaremos de manera general sobre el proceso de transformación entre el modelo entidad relación extendido y el modelo relacional, no daremos una visión detallada y explicación de cada una de las relaciones resultantes, si se necesita consultar dicha información le indicamos que el mejor lugar es el diccionario de datos donde podrá saber más sobre los mismos.

#### 3.1.1. Sobre personas y empleados

De manera análoga a lo que hicimos, para el modelo entidad relación empezamos con la entidad **person**, pasarla al modelo relacional es trivial siguiendo las pautas que conocemos del curso eliminamos atributos compuestos (name) y ponemos los “atributos hoja”, y como llave foránea colocamos la llave primaria de la relación **zip**, este también fue sencillo de pasar al modelo relacional se eliminaron las relaciones intermedias en vez de eso como vimos en clase se agregó una llave foránea que sea la llave primaria de la relación **state**.

**person** no desaparece pues su especialización no es total, es decir en esta solución que estamos proponiendo existe la posibilidad de tener personas que sean doctores y clientes al mismo tiempo o empleados generales que también sean repartidores.

La entidad que si tenía una especialización total fue **employee** que desaparece y los atributos son absorbidos por sus entidades hijas. En este caso estas son **general** (que tiene el identificador del establecimiento como llave foránea, el rfc y el numero de seguridad social), de manera idéntica con **cleaner**, **manager**, **cashier**, y de manera

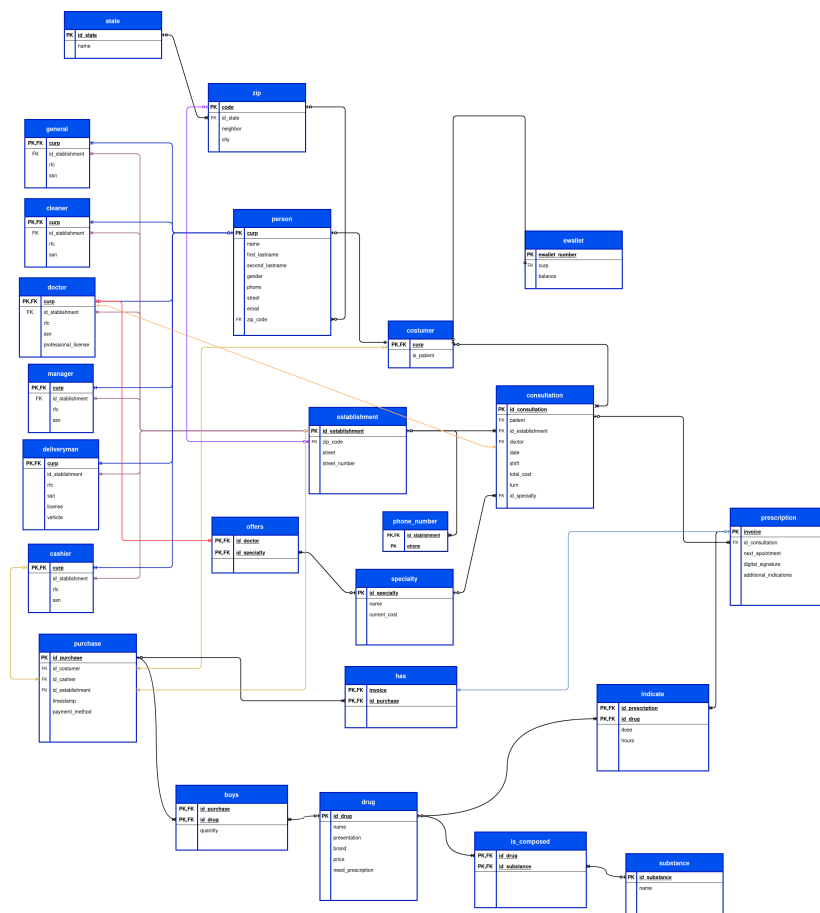
parecida con `deliveryman` (agregando sus atributos únicos), `doctor` (agregando sus atributos únicos).

### 3.1.2. Sobre establecimientos, los medicamentos y consultas

Creamos la relación **establishment**, de manera parecida a **person** contiene un atributo que es una llave foránea a la relación **zip**. Otro interesante es la creación de **phone\_number** una relación que ahora nos permite tener varios números de teléfono asociados a un establecimiento.

Ahora hablemos de las medicinas, la entidad **drug** es igualmente fácil de traducir usando las reglas que vimos en la clase, ahora si conservamos la relación **buys** que básicamente contiene las llaves primarias de **purchase** y **drug** y la cantidad, de manera análoga con la anterior relación **is\_composed** y con **indicates**.

### 3.2. Diagrama final



# Capítulo 4

## Normalización

### 4.1. ¿Que es la normalización?

Una relación bien estructurada.

Es aquella que permite libremente escribir, actualizar o eliminar un registro sin que ocurran anomalías.

En otras palabras una relación bien estructurada, es aquella relación que contiene el mínimo de redundancia y permite a los usuarios insertar, modificar y borrar registros en una tabla sin errores o inconsistencias.

#### 4.1.1. Definición

Decimos que normalizar es el proceso por el cual convertimos todas nuestras relaciones en relaciones bien estructuradas y pequeñas. Es decir, a efectos prácticos lo que hacemos es dividir las relaciones que tenemos.

Podemos decir alternante que Normalización es el proceso de eliminación de redundancias en una tabla para que sea más fácil de modificar.

Podemos decir que la normalización es de descomposición de las tablas para eliminar información redundante y anomalías al momento de insertar, actualizar o eliminar la información.

### 4.2. Primera Forma Normal

Llegar a primera forma es muy sencillo, sobretodo considerando el proceso que ya llevamos hasta llegar aquí.

Basta con asegurarnos que cada atributo sea atómico, es decir eliminar los atributos multivalor ... y podemos decir que no tenemos atributo multivalor, en nuestro diagrama solo tenemos un solo atributo multivaluado que fue correctamente traducido a una relación extra lo cual nos permite que el atributo teléfono siga siendo atómico.

De los demás atributos podemos decir que todos son claramente atómicos con unas contadas casi excepciones:

- **name** no es atómico si consideramos a personas con dos nombres, pero dado que solo vamos a ocupar el nombre para mostrarlo y en ese caso siempre acabaremos concatenando ambos nombres podemos considerar al conjunto de todos los nombres que tenga una persona como tóxicos para nuestro caso.
- **additional\_indications** este será un campo de texto y es bastante posible que el texto sea una serie de indicaciones por ejemplo: “reposo, no comer comida picante y evitar la actividad física intensa por 3 días”, y si desde un punto de vista semántico podemos discutir que estos son en realidad 3 indicaciones, pero es que de igual manera no tiene sentido mucho separarlas sobretodo considerando que al ser personas diferentes podemos acabar con indicaciones muy parecidas pero que no son iguales como evitar salir a correr por 3 días y no hacer actividad física en 3 días, lo cual nos causaría una enorme cantidad de problemas y de la misma manera este campo solo será usado para ser mostrado al doctor que haga la próxima consulta y seguimiento, por lo que para esta solución de igual manera consideramos que las indicaciones son atómicas.

## 4.3. Segunda Forma Normal

### 4.3.1. Dependencia Funcional y Determinantes

Al hablar de la normalización tenemos que hablar casi obligatoriamente de dependencias funcionales y de los determinantes.

Decimos que una dependencia funcional es una restricción entre dos conjuntos de atributos en la base de datos.

Es decir, en una relación  $R(\text{Atributo}_1, \text{Atributo}_2, \dots)$  sean  $X$  y  $Y$  subconjuntos de  $R$ , de la forma:  $X = \{\text{Atributo}_a, \text{Atributo}_b, \dots\}$  y  $Y = \{\text{Atributo}_\alpha, \text{Atributo}_\beta, \dots\}$  entonces decimos que:

**Hay una dependencia funcional de  $X$  a  $Y$  denotada por  $X \longrightarrow Y$  si y sólo si es que para cualquiera dos tupla  $t_1, t_2$  de nuestra base de datos, si los valores de  $t_1$  son iguales a los de  $t_2$  en  $X$ , entonces también su valores serán iguales en los atributos de  $Y$ .**

Decimos por lo mismo que  $X$  determina a  $Y$ , por eso es común ver que se llama a  $X$  como el determinante de una dependencia funcional.

### 4.3.2. Definición

Antes, antes, antes, para hablar de la segunda forma normal tenemos que hablar de lo que es una dependencia funcional parcial y total:

- **Dependencia Funcional Parcial:**

Decimos que tenemos una dependencia funcional parcial  $X \longrightarrow Y$  si es que podemos encontrar un subconjunto de  $X$  que también determina a  $Y$ .

- **Dependencia Funcional Total:**

Decimos que tenemos una dependencia funcional total  $X \longrightarrow Y$  si es que NO podemos encontrar un subconjunto de  $X$  que también determina a  $Y$ .

Decimos que una Relación  $R$  esta en su segunda forma normal si y solo si todos los atributos que no son llaves primarias, secundarias o candidatas solo dependen de la llave primaria.

O de otro modo decimos que un esquema de relación  $R$  está en 2FN si todo atributo no primo depende funcionalmente de manera total de la clave primaria de  $R$ .

### 4.3.3. Nuestro caso

Ahora vayamos relación a relación para ver que no hay una sola que no cumpla la segunda forma normal, recordemos que la segunda forma normal se puede resumir como “the hole key”.

- para **state** solo tenemos un atributo por lo que es trivial que se cumpla 2FN
- para **zip** es igualmente sencillo porque solo tenemos una llave, es decir no poder hacer conjuntos propios no vacíos a partir de ella.
- para **specialty** igual, 1 llave, es decir no poder hacer conjuntos propios no vacíos.
- para **drug** igual, 1 llave, es decir no poder hacer conjuntos propios no vacíos.
- $\dots$ , para evitarnos repetir las 24 relaciones solo hablaremos de aquellas que tengan más de un atributo como llave primaria.

- nota que para casos como `phone_number` es cierto que tenemos una llave compuesta, pero es que no tenemos nada mas que llaves, no hay atributos no primos, por lo que es imposible incumplir la 2NF.

y bueno, hemos acabado todas las relaciones que tenemos, por lo que es trivial hablar de que nuestra solución ya esta en 2NF.

## 4.4. Tercera Forma Normal

### 4.4.1. Nuestro caso

Este se base en el concepto de dependencia transitiva. Una dependencia funcional  $X \rightarrow Y$  en un esquema de relación  $R$  es una dependencia transitiva si existe un conjunto de atributos  $Z$  que no sea subconjunto de las llaves claves de  $R$ , y se cumplen que si  $X \rightarrow Y$  como  $Y \rightarrow Z$ , entonces  $X \rightarrow Z$ .

En la 3FN se elimina las dependencias transitivas; es decir atributos no clave no dependen de otros atributos no clave.

Decimos que una Relación  $R$  esta en su segunda forma normal si y solo si todos los atributos solo dependen de la llave primaria.

Es decir, tenemos que asegurarnos que todo atributo no dependa de nada mas que de la llave:

### 4.4.2. Nuestro caso

- Hay tablas triviales como `is_composed` en las que como no tenemos atributos que sean llave es trivial que se cumple 3FN
- Hay tablas triviales como `state`, `substance`, `offer`, `phone_number`, `customer`, `has` en las que como solo tenemos un atributo es trivial que se cumple 3FN
- para `zip` todo depende solo de la llave, se cumple 3FN
- para `specialty` igual, del id depende el nombre y con el nombre no podemos saber el costo ni inversamente.
- para `drug` podriamos argumentar que dado el nombre podriamos deducir la marca pero eso no se puede aplicar para todos los casos (solo en medicamentos de patente, que por el giro del negocio seran minimos).

Ademas otro dependencia interesante es que si usamos el nombre, la marca y la presentacion entonces podemos acotar el precio, y si, tecnicamente eso incumple



la 3FN lo que nos dictaría usar la tupla nombre, marca y presentación como llave primaria compuesta, pero esto traería dos problemas, primero que nada en el rendimiento pues usar esos 3 valores como llave primaria haría innecesariamente lento usar la tabla y que ahora usar estas como llaves foraneas complicaría bastante su uso, por esto decidimos ignorar esta dependencia funcional y elegir un id como llave para esta tabla.

- para **doctor** podríamos argumentar dado el turno podríamos adivinar y la fecha podríamos adivinar el doctor pero en nuestra solución el responsable de un turno es solo eso, el responsable, es decir no hay impedimento para que algún otro doctor haga una consulta sin ser el responsable, esta es la única posible dependencia transitiva que incumpliría la 3FN,
- para **purchase** podríamos argumentar que el total debería ser un atributo calculado y no estar presente, pero es que debido a los programas de cliente frecuente de la sucursal el total de una compra puede ser menor que la suma de cada una de los elementos comprados y como las compras una vez creadas son inmutables para nuestra solución entonces es necesario tener ese atributo.
- para **establishment, person, ewallet, doctor, deliveryman, prescription, indicates, general, buys, cleaner, cashier** consideramos que no hay ambigüedad y es obvio que no existe dependencia funcional transitiva que incumpla la 3NF

# Capítulo 5

## Implementación es SQL

### 5.1. DDL

Nosotros usa

### 5.2. Población de datos para pruebas

### 5.3. Triggers

### 5.4. Funciones

## Parte II

### Demostración de nuestra solución

# Capítulo 6

## Queries

### 6.1. Comparemos ventas por sucursal

Query Editor		Query History	
1	SELECT	cashier.id_establishment, SUM(total) AS total_sales	
2	FROM	purchase, cashier	
3	WHERE	cashier.curp = purchase.cashier	
4	GROUP BY	cashier.id_establishment	
5	ORDER BY	id_establishment ASC;	
6			

Data Output				Explain	Messages	Notifications
	id_establishment integer		total_sales numeric			
1		1	21188.40			
2		2	31439.86			
3		3	25251.63			
4		4	19928.73			
5		5	26354.65			
6		6	28406.50			
7		7	31646.72			
8		8	28444.63			
9		9	24997.46			
10		10	25334.36			
11		11	19646.78			
12		12	30805.87			

✓ Successfully run. Total query runtime: 52 msec. 20 rows affected.

Figura 6.1: Con nuestra solución podemos hacer consultas para comparar con información a tiempo real las ventas de cada sucursal y ordenarlas por sucursal

## 6.2. Número de consultas por doctor

farmacias/postgres@Localhost

Query Editor Query History

```

1 SELECT
2     CONCAT(p.first_lastname, ' ', p.second_lastname, ' ', p.name) AS fullname,
3     d.curp,
4     COUNT(d.curp) AS total_consultations
5 FROM doctor as d, consultation as c, person as p
6 WHERE
7     d.curp = c.doctor AND
8     p.curp = d.curp
9 GROUP BY
10    d.curp, fullname
11 ORDER BY
12    total_consultations DESC;

```

Data Output Explain Messages Notifications

	fullname text	curp [PK] character (18)	total_consultations bigint
1	Simak Killingbeck Jenny	UTEA769140VWWMP914	45
2	Minall Warriar Roman	FLWV678212SBGPV018	42
3	Oultram Lias Jaymie	NJPX477125JTDBL030	40
4	Ghidetti Iori Cull	TTDP798536TBIAW919	40
5	De Blase Mackley Baxy	FYMD787703WAMEH778	40
6	Starrs D'Alessio Bartholomeus	ZGOM793916SALAG489	40
7	Marple Iacoboni Alica	WQVH961260NTIXJ879	39
8	Cristoferi Cassin Alexandra	RSVD601747IMRDU031	39
9	Cuchey Boolis Norton	PFKF271336XPCYA822	38
10	Brennon Franzotto Tonnie	RKWT706451XAKSH115	38
11	Baudi Currier Jeri	NEPT613248TSSJT590	
12	Barns Aldiss Arvin	AJBR420040UGFWC279	

✓ Successfully run. Total query runtime: 58 msec. 60 rows affected.

Figura 6.2: Con nuestra solución podemos medir la productividad de cada doctor mostrando las consultas que han dado y ordenarlos por de mayor a menor usando dicha metrica

## 6.3. Número de consultas por mes del año

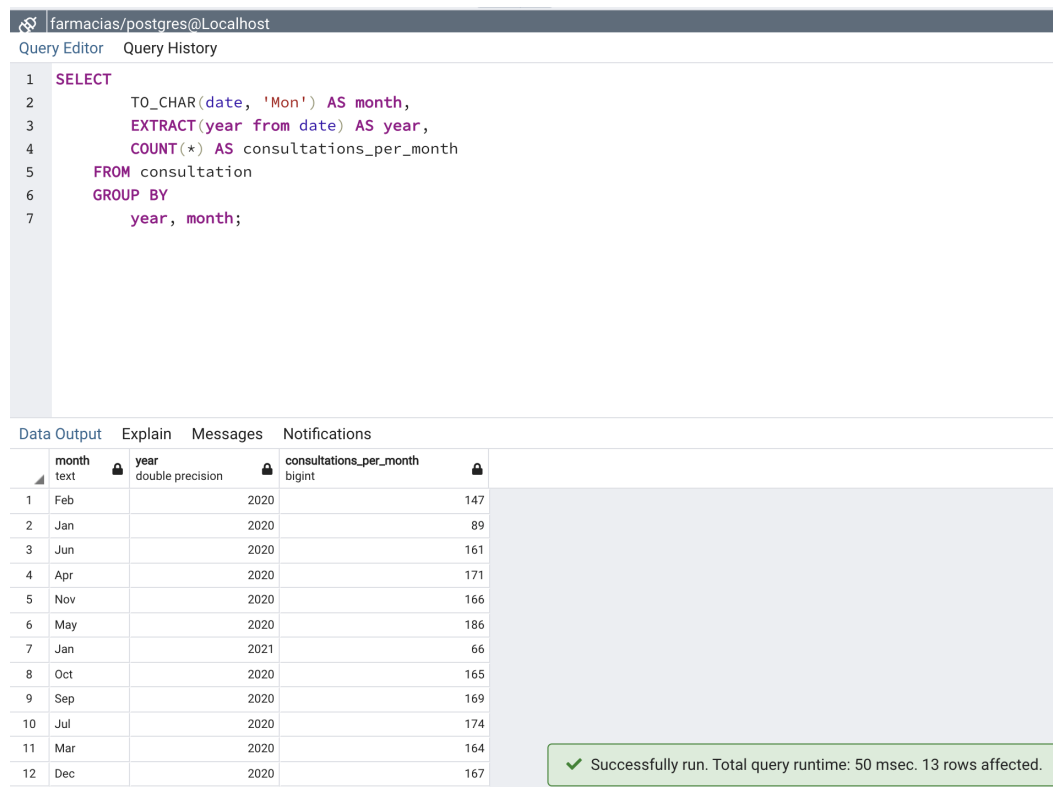


Figura 6.3: Con nuestra solución podemos tambien ver las fluctuaciones a lo largo del año en el numero de consultas creadas, permitiendo alocar de mejor manera recursos con esta información

6.4. Productos más vendidos

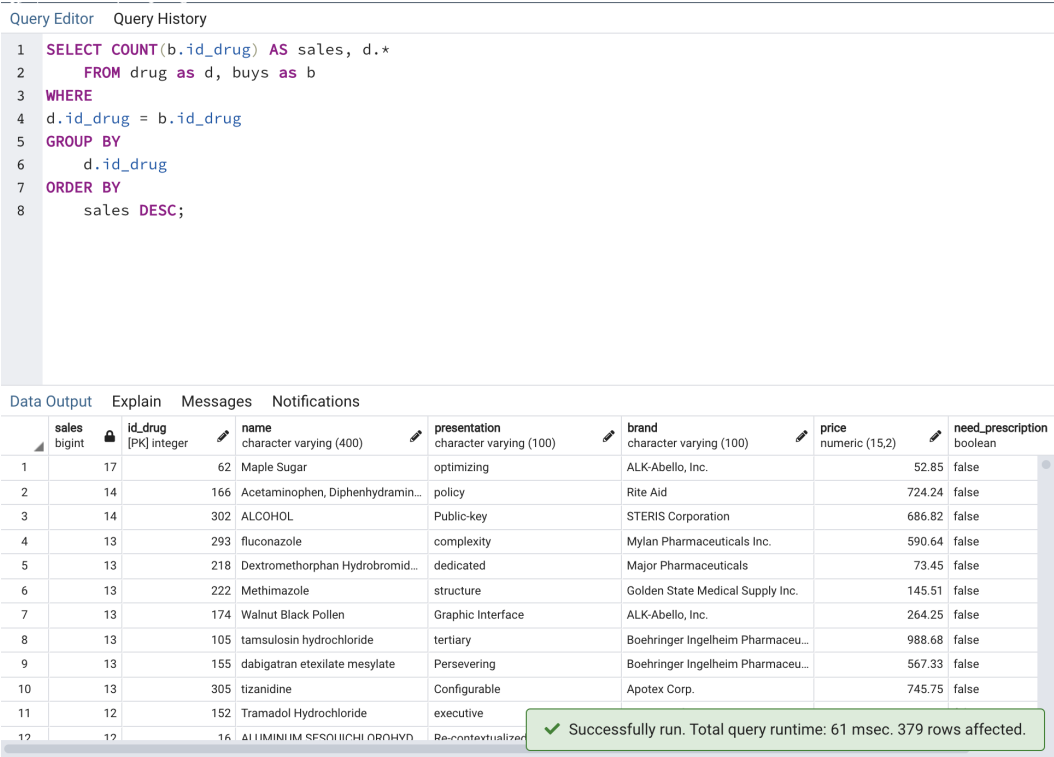


Figura 6.4: Con nuestra solución podemos tambien saber con información de tiempo real cuales son los productos que mas se han vendido (dando detalles que la presentación mas popular por ejemplo)

# 6.5. Consultas por especialidad

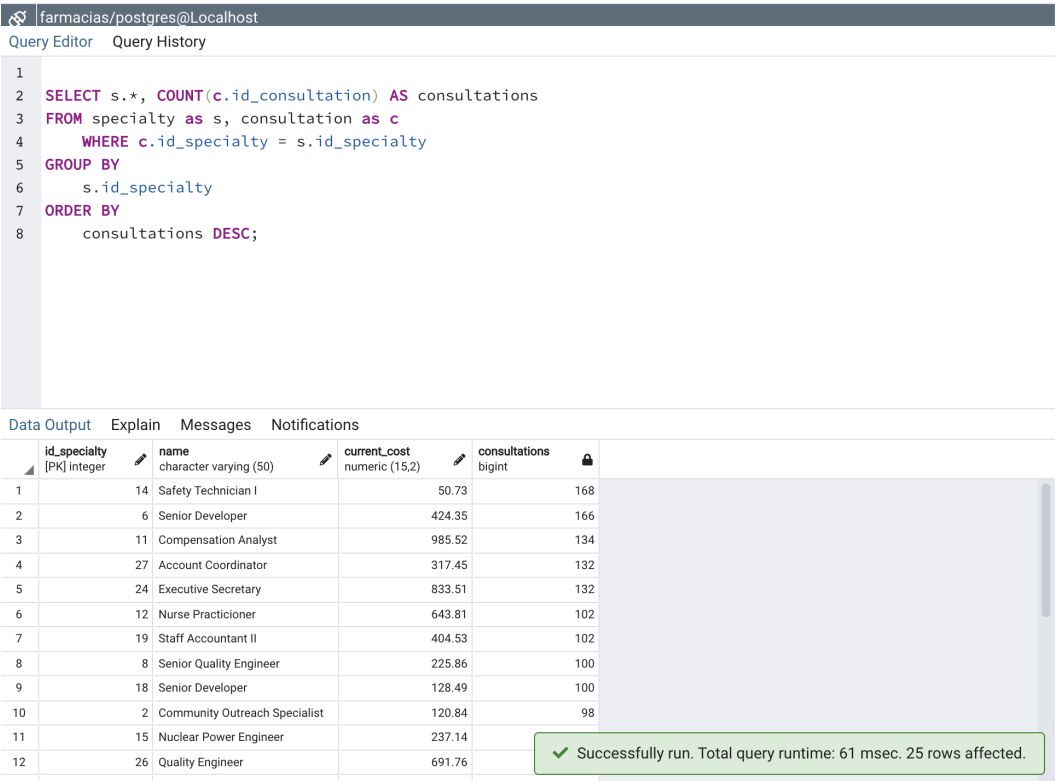


Figura 6.5: Con nuestra solución podemos tambien cual es el tipo de consulta mas popular y cuando es la comision en este momento por dicha especialidad



## 6.6. Mayor ingreso por especialidad

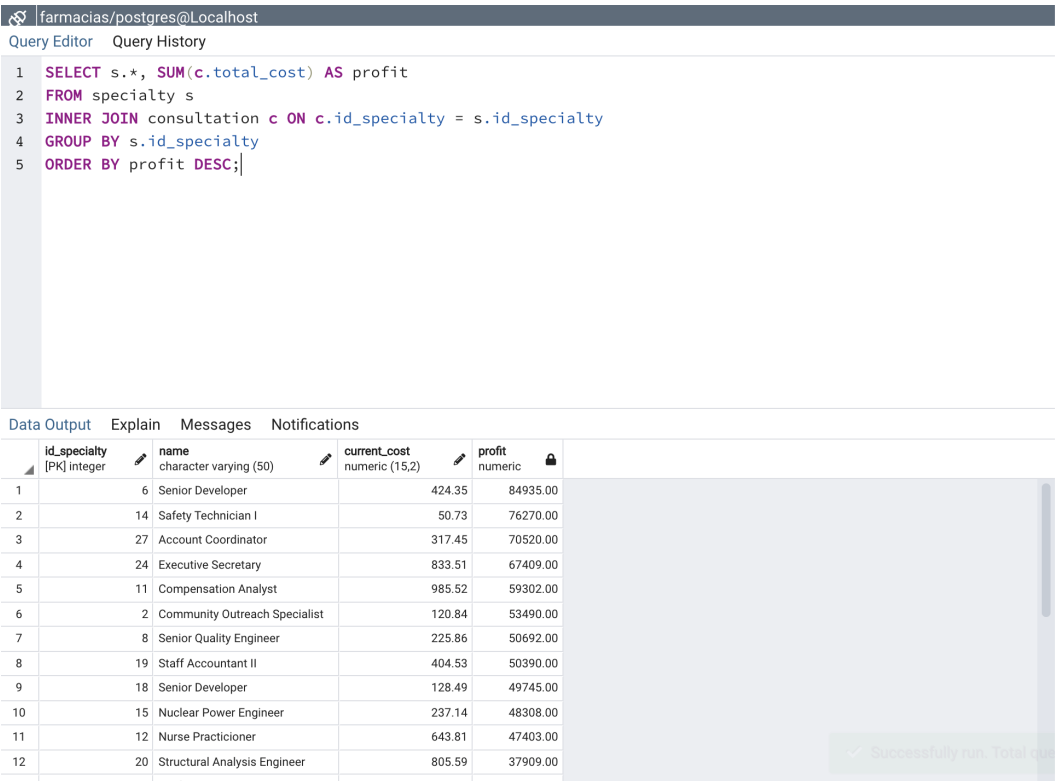
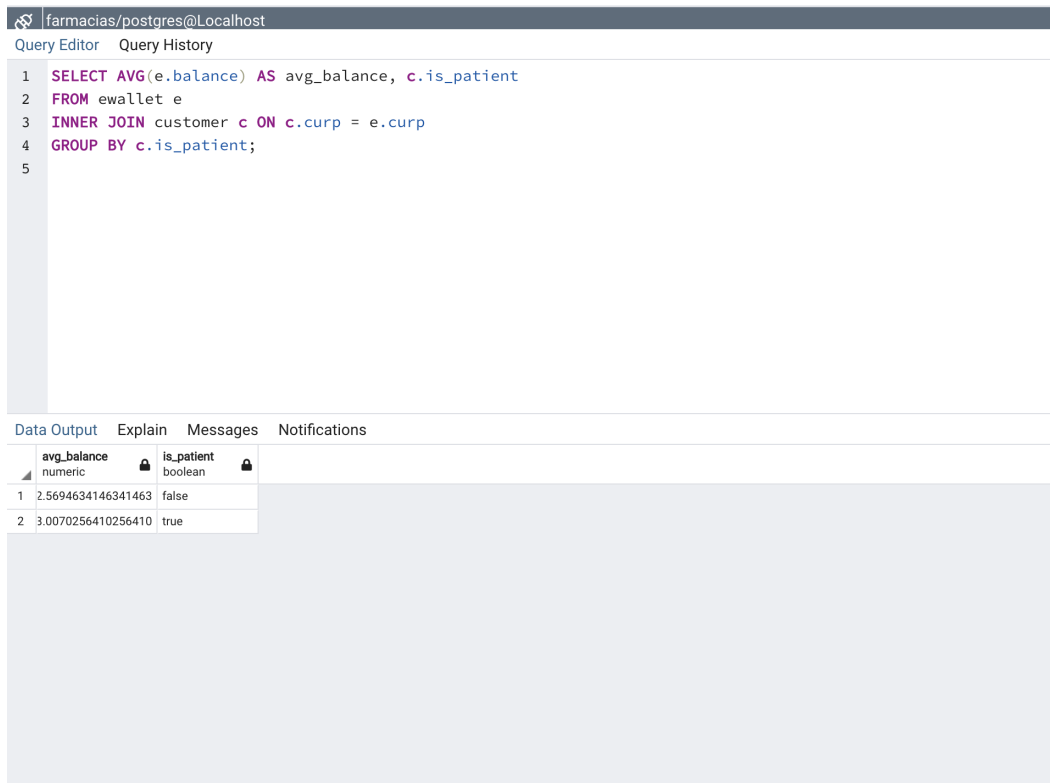


Figura 6.6: Con nuestra solución podemos ordenar las especialidades basados en la cantidad de ingresos que hemos recibido por ellas (es decir sumar la comision de cada consulta hecha de dicha especialidad)

## 6.7. Balance promedio entre clientes vs pacientes



The screenshot shows a PostgreSQL query editor interface. The top bar indicates the connection is to 'farmacias/postgres@Localhost'. Below the bar, there are tabs for 'Query Editor' and 'Query History'. The query editor contains the following SQL query:

```
1 SELECT AVG(e.balance) AS avg_balance, c.is_patient
2 FROM ewallet e
3 INNER JOIN customer c ON c.curp = e.curp
4 GROUP BY c.is_patient;
5
```

Below the query editor, there are tabs for 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with two columns: 'avg\_balance' (numeric) and 'is\_patient' (boolean). The table contains two rows of data:

	avg_balance numeric	is_patient boolean
1	2.5694634146341463	false
2	3.0070256410256410	true

Figura 6.7: Con nuestra solución podemos tambien usando información en tiempo real saber cual es el balance promedio de alguien que solo sea un cliente y alguien que sea un paciente

# 6.8. Cajeros mas productivos

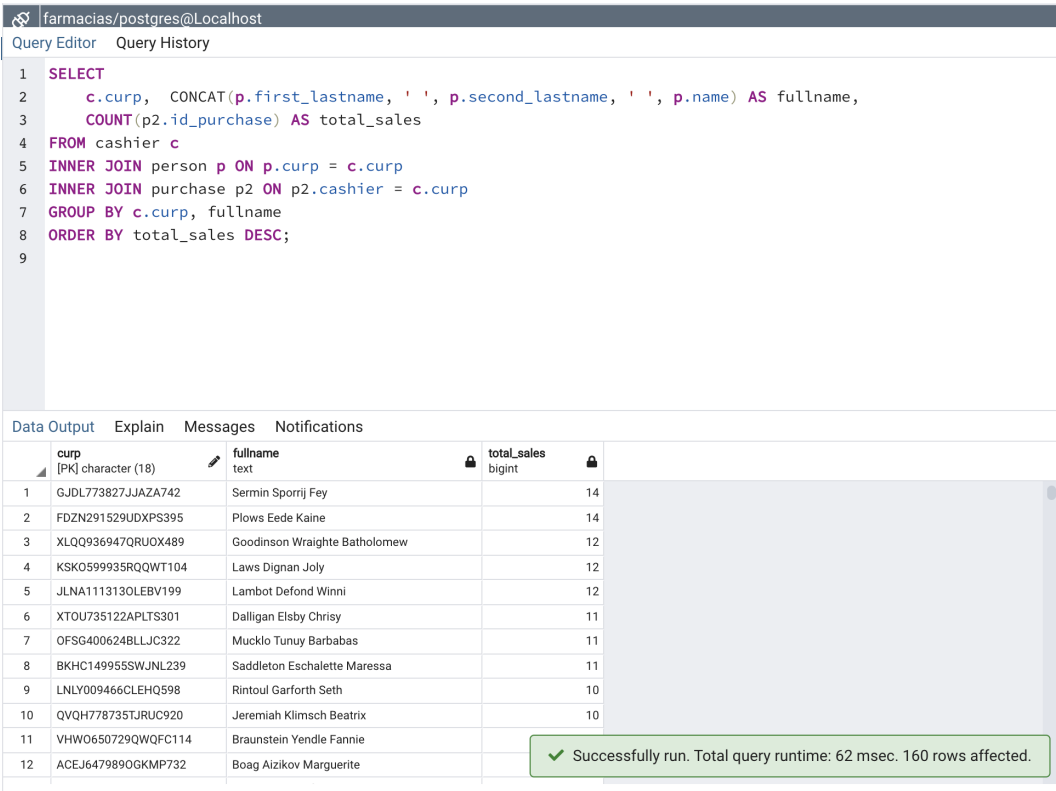


Figura 6.8: Con nuestra solución podemos saber para cada cajero cuantas ventas ha realizado

# 6.9. Ventas por hora

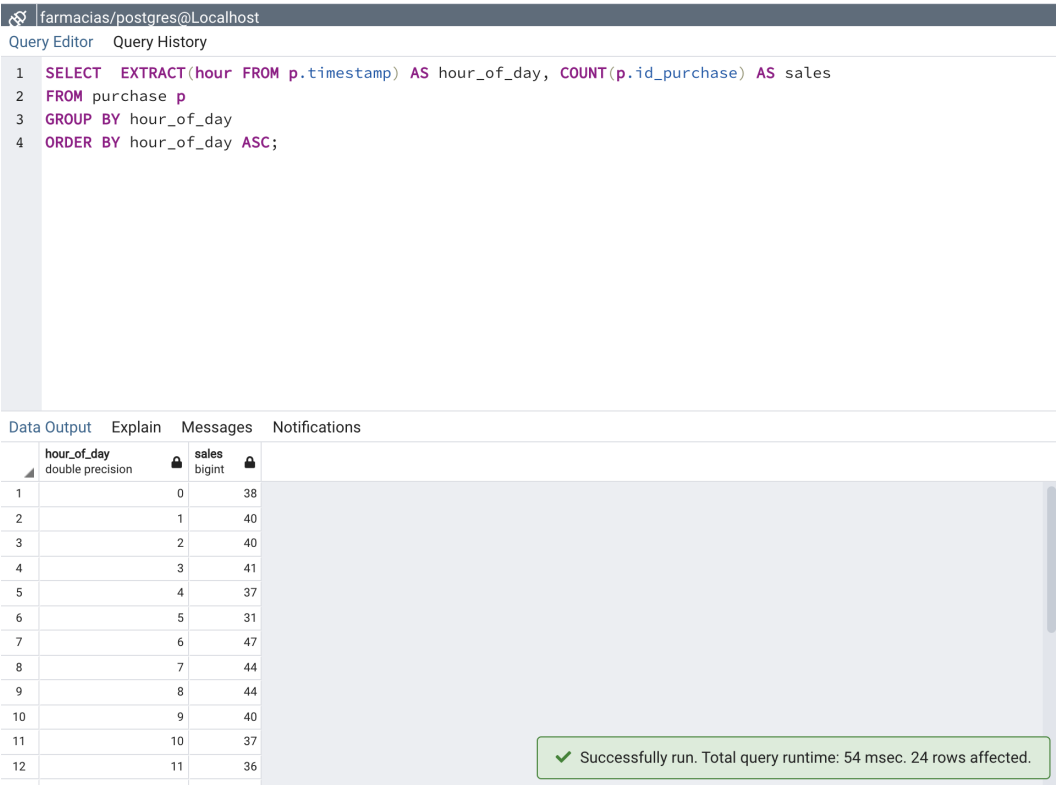


Figura 6.9: Con nuestra solución de manera similar a como hicimos con los meses del año podemos ver como fluctuan las venta a lo largo de la hora del día en tiempo real

# 6.10. Cuentas de doctores que generan una proxima cita

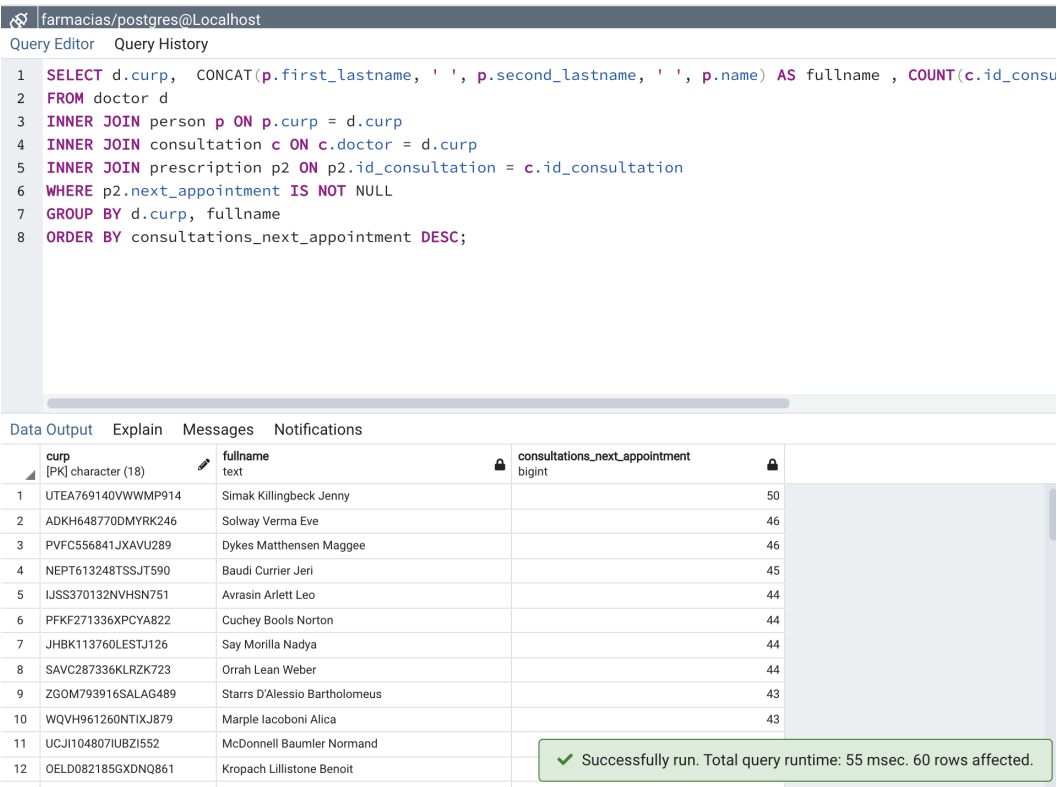


Figura 6.10: Con nuestra solución podemos saber rapidamente quienes son los doctores que tienen en este momento una cita pendiente

## 6.11. Total de medicamentos en receta por especialidad cuyo precio es mayor a 500

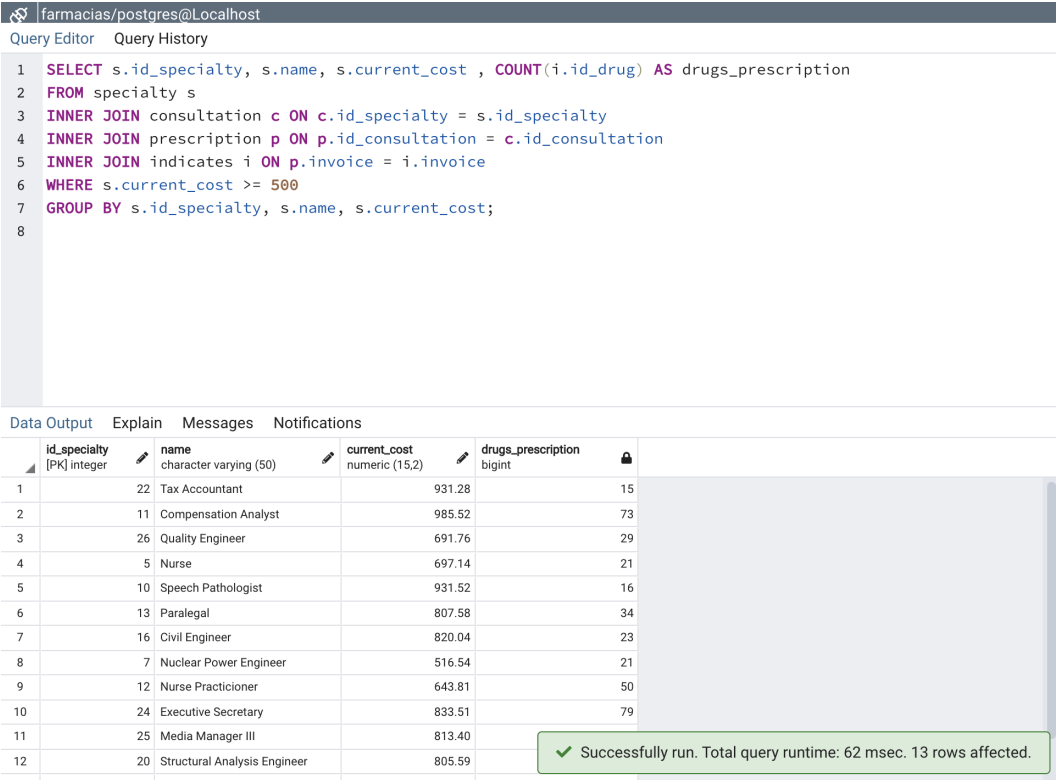


Figura 6.11: aqui por ejemplo podemos ver los medicamentos que requieren receta que provienen de una cita por especialidad que costo mas de 500, ademas mostramos cuantas veces dicho medicamento ha sido comprado

## 6.12. Posible ingreso por farmacos que vengan por receta

farmacias/postgres@Localhost																																																							
Query Editor Query History																																																							
<pre> 1 SELECT d.id_drug, d.name, COUNT(i.id_drug) * d.price AS possible_revenue 2 FROM drug d 3 INNER JOIN indicates i ON i.id_drug = d.id_drug 4 GROUP BY d.id_drug, d.name; 5 </pre>																																																							
Data Output	Explain	Messages	Notifications																																																				
<table> <tr> <th></th><th>id_drug [PK] integer</th><th>name character varying (400)</th><th>possible_revenue numeric</th></tr> <tr><td>1</td><td>384</td><td>Atenolol</td><td>374.58</td></tr> <tr><td>2</td><td>351</td><td>Neomycin and Polymyxin B Sulf...</td><td>324.14</td></tr> <tr><td>3</td><td>184</td><td>SODIUM FLUORIDE and POTASSI...</td><td>89.42</td></tr> <tr><td>4</td><td>116</td><td>entacapone</td><td>1006.12</td></tr> <tr><td>5</td><td>87</td><td>Diphenhydramine Hydrochloride</td><td>3425.64</td></tr> <tr><td>6</td><td>273</td><td>Rabeprazole Sodium</td><td>3731.40</td></tr> <tr><td>7</td><td>51</td><td>Triclosan</td><td>242.22</td></tr> <tr><td>8</td><td>272</td><td>Zinc Oxide</td><td>1569.36</td></tr> <tr><td>9</td><td>70</td><td>Clonazepam</td><td>550.69</td></tr> <tr><td>10</td><td>190</td><td>ropinirole</td><td>492.00</td></tr> <tr><td>11</td><td>350</td><td>Hydrocodone Bitartrate And Acet...</td><td>614.22</td></tr> <tr><td>12</td><td>314</td><td>Fluphenazine Hydrochloride</td><td>797.13</td></tr> </table>		id_drug [PK] integer	name character varying (400)	possible_revenue numeric	1	384	Atenolol	374.58	2	351	Neomycin and Polymyxin B Sulf...	324.14	3	184	SODIUM FLUORIDE and POTASSI...	89.42	4	116	entacapone	1006.12	5	87	Diphenhydramine Hydrochloride	3425.64	6	273	Rabeprazole Sodium	3731.40	7	51	Triclosan	242.22	8	272	Zinc Oxide	1569.36	9	70	Clonazepam	550.69	10	190	ropinirole	492.00	11	350	Hydrocodone Bitartrate And Acet...	614.22	12	314	Fluphenazine Hydrochloride	797.13			<div> <span>✓</span> Successfully run. Total query runtime: 62 msec. 364 rows affected. </div>
	id_drug [PK] integer	name character varying (400)	possible_revenue numeric																																																				
1	384	Atenolol	374.58																																																				
2	351	Neomycin and Polymyxin B Sulf...	324.14																																																				
3	184	SODIUM FLUORIDE and POTASSI...	89.42																																																				
4	116	entacapone	1006.12																																																				
5	87	Diphenhydramine Hydrochloride	3425.64																																																				
6	273	Rabeprazole Sodium	3731.40																																																				
7	51	Triclosan	242.22																																																				
8	272	Zinc Oxide	1569.36																																																				
9	70	Clonazepam	550.69																																																				
10	190	ropinirole	492.00																																																				
11	350	Hydrocodone Bitartrate And Acet...	614.22																																																				
12	314	Fluphenazine Hydrochloride	797.13																																																				

Figura 6.12: aqui podemos ver los ingresos que hemos tenido debido a medicamentos que provienen de una prescripción

# 6.13. Demografica de nuestros pacientes

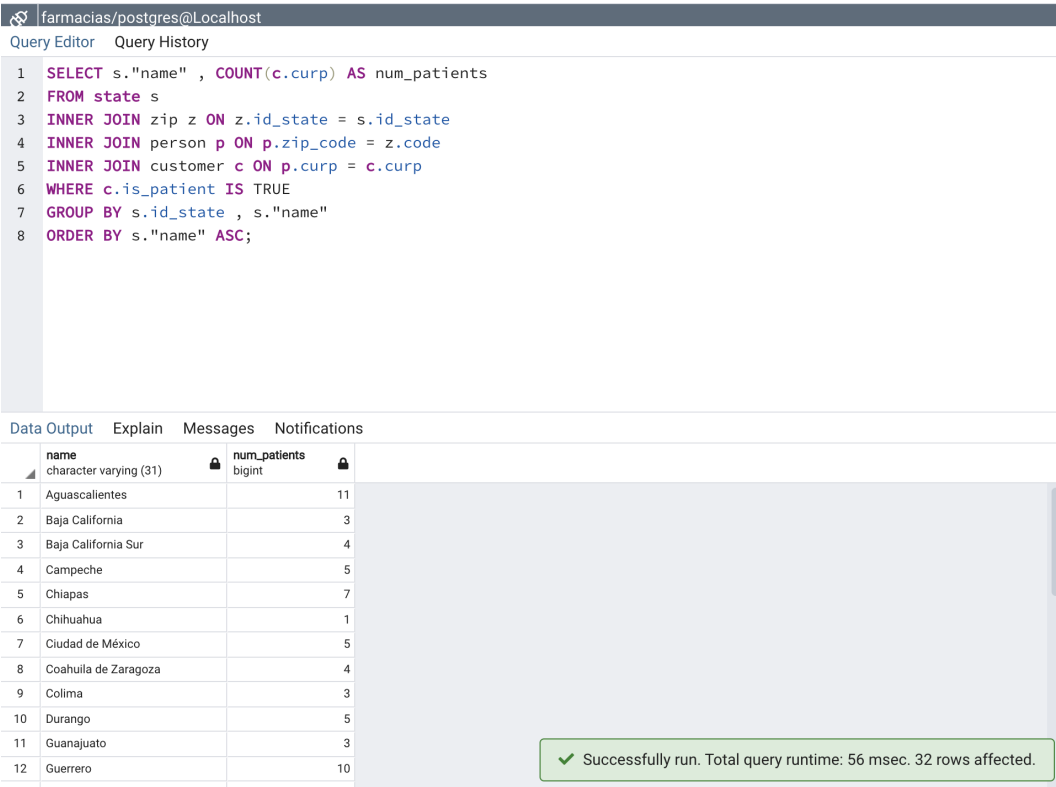


Figura 6.13: aqui podemos la cantidad de pacientes que atendemos por estado de origen



## 6.14. La ubicación por estado que genera más ingreso a través de compras por clientes

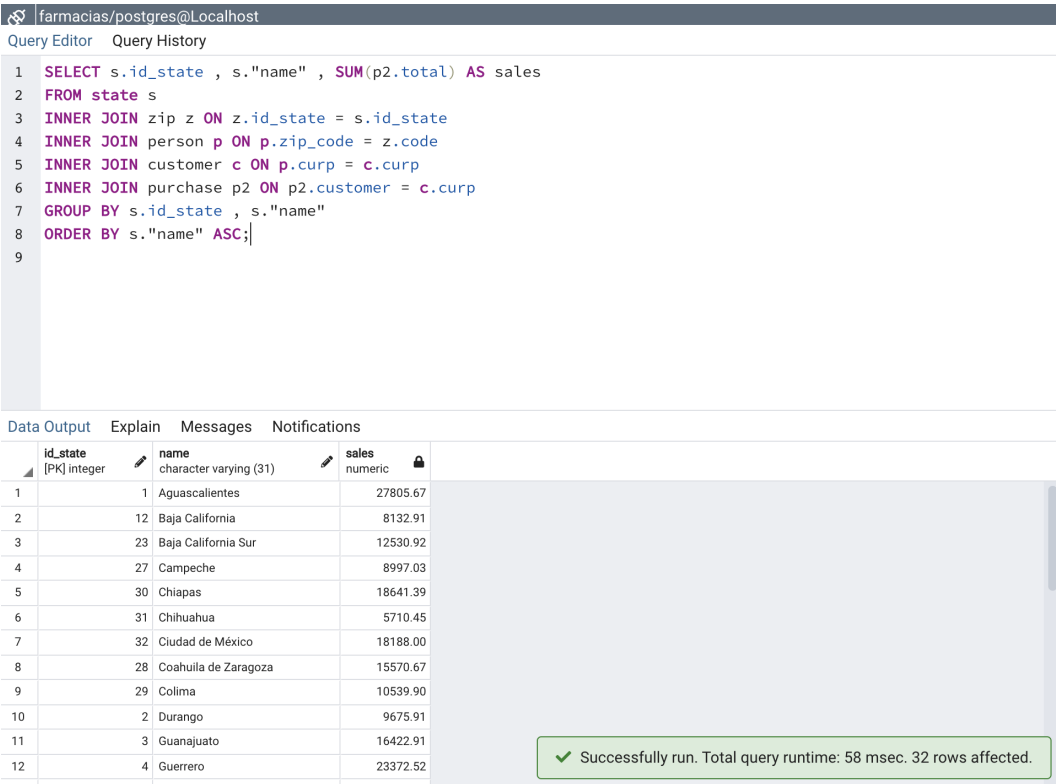


Figura 6.14: La ubicación por estado que genera más ingreso a través de compras por clientes

# 6.15. Ventas de fármacos que necesitan receta

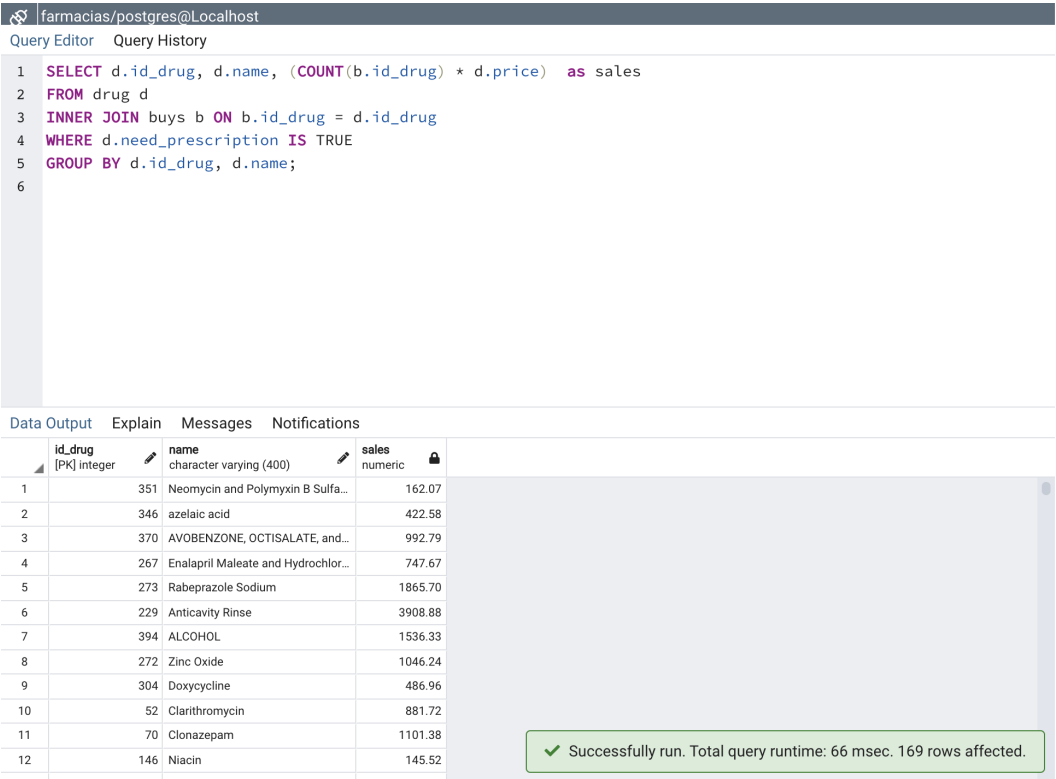


Figura 6.15: Aqui podemos ver las ventas de los medicamentos en los que es necesaria una receta para su venta