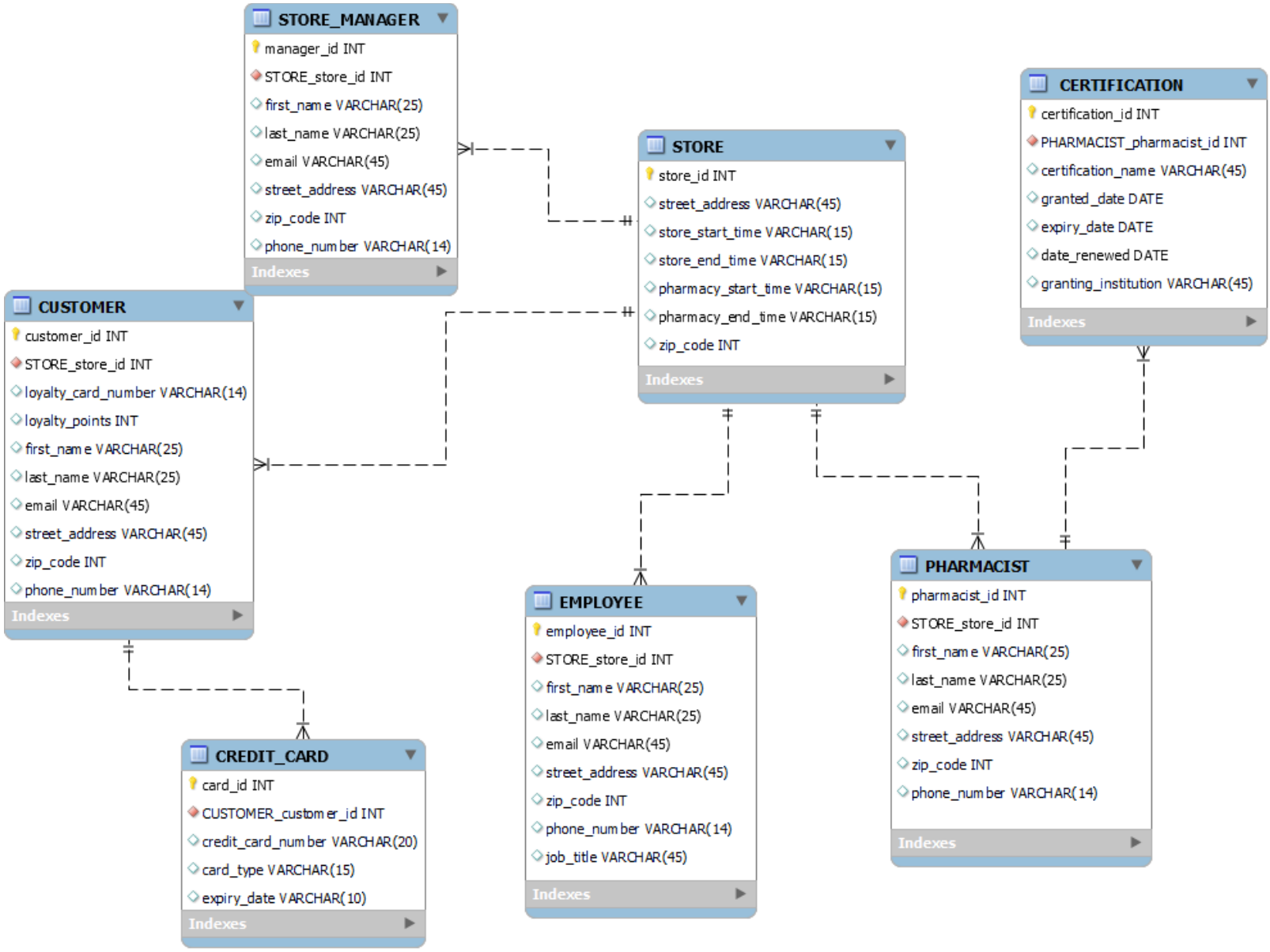
CVS Database

ENTITY CREATIONS

John A, Peter D, Carter D

Data Model:


STORE_MANAGER
manager_id INT
STORE_store_id INT
first_name VARCHAR(25)
last_name VARCHAR(25)
email VARCHAR(45)
street_address VARCHAR(45)
zip_code INT
phone_number VARCHAR(14)
Indexes
STORE
store_id INT
street_address VARCHAR(45)
store_start_time VARCHAR(15)
store_end_time VARCHAR(15)
pharmacy_start_time VARCHAR(15)
pharmacy_end_time VARCHAR(15)
zip_code INT
Indexes
CERTIFICATION
certification_id INT
PHARMACIST_pharmacist_id INT
certification_name VARCHAR(45)
granted_date DATE
expiry_date DATE
date_renewed DATE
granting_institution VARCHAR(45)
Indexes
CUSTOMER
customer_id INT
STORE_store_id INT
loyalty_card_number VARCHAR(14)
loyalty_points INT
first_name VARCHAR(25)
last_name VARCHAR(25)
email VARCHAR(45)
street_address VARCHAR(45)
zip_code INT
phone_number VARCHAR(14)
Indexes
EMPLOYEE
employee_id INT
STORE_store_id INT
first_name VARCHAR(25)
last_name VARCHAR(25)
email VARCHAR(45)
street_address VARCHAR(45)
zip_code INT
phone_number VARCHAR(14)
job_title VARCHAR(45)
Indexes
PHARMACIST
pharmacist_id INT
STORE_store_id INT
first_name VARCHAR(25)
last_name VARCHAR(25)
email VARCHAR(45)
street_address VARCHAR(45)
zip_code INT
phone_number VARCHAR(14)
Indexes
CREDIT_CARD
card_id INT
CUSTOMER_customer_id INT
credit_card_number VARCHAR(20)
card_type VARCHAR(15)
expiry_date VARCHAR(10)
Indexes

Data Model Commentary:

Given that we were tasked with modeling information for CVS from the chain level down to individual pharmacists and customers the store was a good table to start with. There is multiple 1:m relationships from the store table with customer pharmacists and store manager. While they all have a relationship with the store their information is unique and related only to them. For example, we have store manager as its own 1:m relationship with the store as one store can have many managers and one manger can only manage one store. It could be argued that one store manager has many pharmacists as well working "under them" but we felt 1, with prior knowledge knowing the pharmacy and store aspect of CVS's are relatively separate and it could provide issues with the pharmacist data down the lone should the manager data change. By tying it to the store that is a much simpler change for all three, customer, store manager and pharmacist. Pharmacists also have the certification table which is not at all related to the store manager. We had a similar reasoning for the location and 1:m relationship between customer and the CREDIT_CARD table.

Another area of note is the store table where we are recording the hours of the store and the pharmacy. Hours of stores are recorded in a variety of formats ad that is why we gave it a VARCHAR datatype, so the data can be recorded as necessary.

Data Dictionary:

Attribute	Data Type	Length	Constraint	Description
customer_id	INT	10	Primary Key	Arbitrary number assined to a customer for identification, stored as an INT. This is auto-generated.
loyalty_card_number	INT	10	NULL	Arbitrary number assined to a customer loyalty card for identification, stored as an INT. This is auto-generated.
loyalty_points	INT	10	NULL	Sum of loyalty points a customer has accrued. A function of purchases and points used.
first_name	VARCHAR	25	NULL	String of characters containing the first name of a customer.
last_name	VARCHAR	25	NULL	String of characters containing the last name of a customer.
email	VARCHAR	45	NULL	String of characters containing the email of a customer.
street_address	VARCHAR	45	NULL	String of characters containing the street address of a customer.
zip_code	INT	10	NULL	Number stored as INT containg the zip code of a customer. Information that will be used with street address to get a percise location of a customers home.
phone_number	VARCHAR	14	NULL	Phone number of the customer. Stored as a string of characters because phone number will never have mathematical functions called on it.
STORE_store_id	INT	10	Foreign Key	An INT pulled from the store table which assigns customers to a store. This data type is an arbitrary number generated in the store table to identify diferrent store locations.

Queries:

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1. Subquery	X								X	X
2. BETWEEN			X							
3. GROUP BY		X				X	X	X	X	
4. HAVING		X				X			X	
5. ORDER BY				X	X		X	X		X
6. IN/NOT IN	X									
7. Aggregate function		X				X	X	X	X	X
8. REGEXP	X		X							
9. Date function				X	X	X				
10. Calculation						X			X	

#1 Customers that have either a visa or MasterCard card and first_name starts with a j or c.

Mastercard and visa historically have the lowest credit card fees of companies so it's good to know how much of your customer base uses a visa or MasterCard credit cards. We can then filter by name to improve readability.

```
> SELECT first_name, last_name

FROM customer

WHERE customer_id  IN (SELECT CUSTOMER_customer_id FROM credit_card WHERE card_type IN ('MasterCard','Visa')) AND first_name REGEXP '^J|^C|^c|^j'

LIMIT 5;

+ --------------- + -------------- +

| first_name      | last_name      |

+ --------------- + -------------- +

| Jennifer        | Young          |

| Joseph          | King           |

+ --------------- + -------------- +

2 rows
```

#2 Stores that have more than 5 pharmacists

It can allow the business to know which stores are overstaffed as they can make the query show them which stores have more pharmacists than they believe is needed.

```
> SELECT store_id, COUNT(pharmacist_id) AS average_employees

FROM pharmacist JOIN store ON pharmacist.STORE_store_id = store.store_id

GROUP BY store_id

HAVING average_employees > 5

LIMIT 5;
```

```
+ ------------- + ---------------------- +

| store_id      | average_employees      |

+ ------------- + ---------------------- +

| 6             | 8                      |

| 8             | 7                      |

| 12            | 7                      |

| 15            | 7                      |

| 36            | 7                      |

+ ------------- + ---------------------- +

5 rows
```

#3 Customers with loyalty points between 750 and 1000 and last name ends with h or b

It can allow the business to know the customers that have points on their high scale so they can possibly reward them for being loyal. We can then filter by name to improve readability.

```
> SELECT first_name, last_name, loyalty_points

FROM customer

WHERE loyalty_points BETWEEN 750 AND 1000 AND last_name REGEXP '^H|^B|^h|^b'

LIMIT 5;
```

```
+ --------------- + -------------- + -------------------- +
| first_name      | last_name      | loyalty_points      |
+ --------------- + -------------- + -------------------- +
| Michael         | Bennett        | 950                 |
| Patricia        | Hall           | 900                 |
+ --------------- + -------------- + -------------------- +
2 rows
```

#4 Managers that are in stores that have open before 9am and close before 9pm

Allows the business to know the managers of stores that open before 9am and close before 9pm and be able to communicate with them about how they feel about the hours.

```
> SELECT store_id, first_name, last_name
FROM store JOIN store_manager ON store.store_id=store_manager.STORE_store_id
WHERE store_start_time < '9:00 AM'  AND store_end_time < '9:00 PM'
ORDER BY first_name
LIMIT 5;
+ ------------- + --------------- + -------------- +
| store_id      | first_name      | last_name      |
+ ------------- + --------------- + -------------- +
| 6             | Amy             | Miller         |
| 6             | Amy             | Hall           |
| 36            | Anna            | Green          |
| 15            | Carol           | White          |
| 37            | Carol           | Watson         |
+ ------------- + --------------- + -------------- +
5 rows
```

#5 Pharmacists that are in stores that have open after 11am and close before 5pm

Allows the business to know the pharmacists of pharmacies that open after 11am and close before 5pm and be able to communicate with them about how they feel about the hours.

```
> SELECT store_id, first_name, last_name

FROM store JOIN pharmacist ON store.store_id=pharmacist.STORE_store_id

WHERE pharmacy_start_time > '11:00 AM'  AND pharmacy_end_time < '5:00 PM'

ORDER BY first_name

LIMIT 5;
```

store_id	first_name	last_name
15	Amelia	James
15	Ava	Cooper
12	Daniel	Morris
36	David	Brown
36	Ella	Gonzalez

5 rows

#6 Average years of certification of all pharmacists of stores with pharmacists

Allows the business to know the average years of experience of each pharmacy of their pharmacists. This allows them to know the average level of experience each of their pharmacies has.

```
> SELECT store_id, ROUND(AVG(DATEDIFF(CURRENT_DATE,granted_date))/365,0) AS
average_years_since_certification

FROM store

LEFT JOIN pharmacist ON store.store_id=pharmacist.STORE_store_id

LEFT JOIN certification ON pharmacist.pharmacist_id =
certification.PHARMACIST_pharmacist_id

GROUP BY store_id

HAVING average_years_since_certification  IS NOT NULL

LIMIT 5
```

```
+ ------------ + ------------------------------------ +
| store_id     | average_years_since_certification    |
+ ------------ + ------------------------------------ +
| 6            | 6                                    |
| 8            | 6                                    |
| 36           | 3                                    |
| 37           | 4                                    |
| 49           | 6                                    |
+ ------------ + ------------------------------------ +
5 rows
```

#7 List the top 5 stores with the highest customer count.

Allows the business to know the top 5 stores with the most customers.

```
> SELECT store_id, COUNT(customer_id) AS customer_count
FROM store JOIN customer ON customer.STORE_store_id = store.store_id
GROUP BY store_id
ORDER BY customer_count DESC
LIMIT 5;
+ ------------ + ------------------ +
| store_id     | customer_count     |
+ ------------ + ------------------ +
| 6            | 10                 |
| 15           | 8                  |
| 8            | 7                  |
| 12           | 7                  |
| 49           | 7                  |
+ ------------ + ------------------ +
5 rows
```

#8 Calculate the average loyalty points of customers for each store.

Allows the business to know which stores have the most loyal customers so that they can see if there is a replicable reason and if so, replicate it in other stores.

```
> SELECT store_id, AVG(loyalty_points) AS average_loyalty_points

FROM store JOIN customer ON customer.STORE_store_id = store.store_id

GROUP BY store_id

ORDER BY average_loyalty_points DESC

LIMIT 5;
```

store_id	average_loyalty_points
49	1100.0000
37	1014.2857
36	975.0000
8	857.1429
15	681.2500

5 rows

#9 Calculate the average number of certifications per pharmacist in each store that has pharmacists.

Allows the business to know the average number of certifications per pharmacist at each pharmacy. Allows you to know pharmacies have more certified pharmacists.

```
> SELECT store_id, FORMAT(AVG(number_of_certifications),0) AS
average_of_certifications_per_pharmacist

FROM store LEFT JOIN (

    SELECT STORE_store_id , pharmacist_id, COUNT(*) AS
number_of_certifications

    FROM certification LEFT JOIN pharmacist ON pharmacist.pharmacist_id =
certification.PHARMACIST_pharmacist_id

    GROUP BY STORE_store_id, pharmacist_id

) AS certification_count ON store.store_id =
certification_count.STORE_store_id

GROUP BY store_id

HAVING average_of_certifications_per_pharmacist  IS NOT NULL

LIMIT 5;
```

store_id	average_of_certifications_per_pharmacist
6	7
8	7
36	7
37	8
49	7

5 rows

#10 Find the customers who have more loyalty points than the average loyalty points of all customers.

Allows the business to know all the customers that have higher than average loyalty points. They can use this to possibly reward this group of people or try to see if it’s store specific using query 8.

```
> SELECT first_name, last_name, loyalty_points

FROM customer

WHERE loyalty_points > (SELECT AVG(loyalty_points) FROM customer)

ORDER BY loyalty_points

LIMIT 5;
```

first_name	last_name	loyalty_points
Christopher	Turner	800
Charles	Stewart	800
Mary	Morgan	800
Steven	Wilson	800
Michael	Anderson	850

5 rows