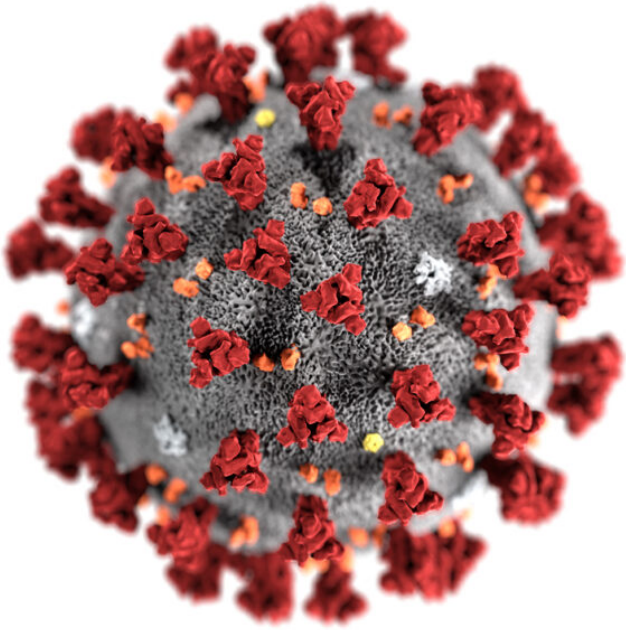
State-Based COVID-19 Vaccination Allocation Database Report



Members: Justin Nichols, Andrew Kimball, Michael Gee, Cameron Gamble
CSC 4402 Final Project
Instructor: Mingxuan Sun
22 April 2021

Table of Contents

1 Preface

The ultimate goal of this database application is to gather the current vaccination allocation, case count, and population of each U.S. state/territory to develop observations and make conclusions about the COVID-19 vaccine distribution around the country. Note, we have included both U.S. states as well as territories, but we will refer to both as just 'states'. The database gathers the number of allocated vaccines from the approved companies (Pfizer, Moderna, and Janssen) given to each state and compares it to the state's population and total COVID-19 case count. The database shows trends surrounding how the number of cases as well as the population of the state influences the state-based vaccine allocation, distribution, and vaccinate rate. This dependence can be viewed within the database design.

Due to the COVID-19 vaccine campaign being the largest of its kind in world history, the data, regulations, information, and logistics are fast changing. With this, we would like to bring attention to a few key changes that occurred in the last few weeks. From the start of the development of this database application in early April 2021, many things have changed. The primary one being that the Johnson and Johnson Janssen vaccine has been indefinitely suspended due to health concerns and abnormal side effects. Although these new developments are significant, information is currently still scarce surrounding how this will change the overall vaccine distribution. Therefore, we have made the decision to use data obtained from the start of April (when we started on this project), which does not include these recent developments.

The data used within this database application comes from as recent as the first week of April 2021. The vaccine data acquired comes from https://www.data.gov/, the population statistics come from https://www.census.gov/, and the case count numbers come from https://www.cdc.gov/. We are using the relational database application MYSQL on the MariaDB server to make our schemas.

Our group wanted to pick a topic that was not only relevant but also had a multitude of data to gather and obtain results from. COVID-19 was an obvious choice for us, considering how much it has impacted the country in the last year. This database gathers COVID-19 data unbiasedly and makes connections within the data in a factual manner.

2 Domain Application

The database is based on the COVID-19 outbreak that has spread across the United States and the world. There are many conditions that have affected the federal government's vaccine distribution plans. The primary two are the population and case count of a state. In this section, we identify the entities and corresponding attributes of the database along with the constraints and assumptions imposed on the data. We translated this knowledge into an Entity Relationship model.

2.1 Database Entities and Corresponding Attributes

We identified six primary entities that are of importance within our database: total_cases_per_state, total_population_per_state, total_vaccines_per_state, moderna, pfizer, and janssen. Among the total_cases_per_state, total_population_per_state, and total_vaccines_per_state entities, one-to-one relationships exist considering each state has unique data and trends associated with it. However, there is a many-to-one relationship that exists between the moderna, pfizer, and janssen entities toward the total_vaccines_per_state entity. A one-to-many exists in the opposite direction, that being total_vaccines_per_state entity to the moderna, pfizer, and janssen entities.

Here is a complete list of the entities and attributes:

- total_cases_per_state(state, total_cases)
- total_population_per_state(state, total_population)
- total_vaccines_per_state(state, total_vaccines)
- moderna(state, week, first_dose, second_dose)
- pfizer(state, week, first_dose, second_dose)
- jansen(state, week, only_dose)

2.2 Constraints

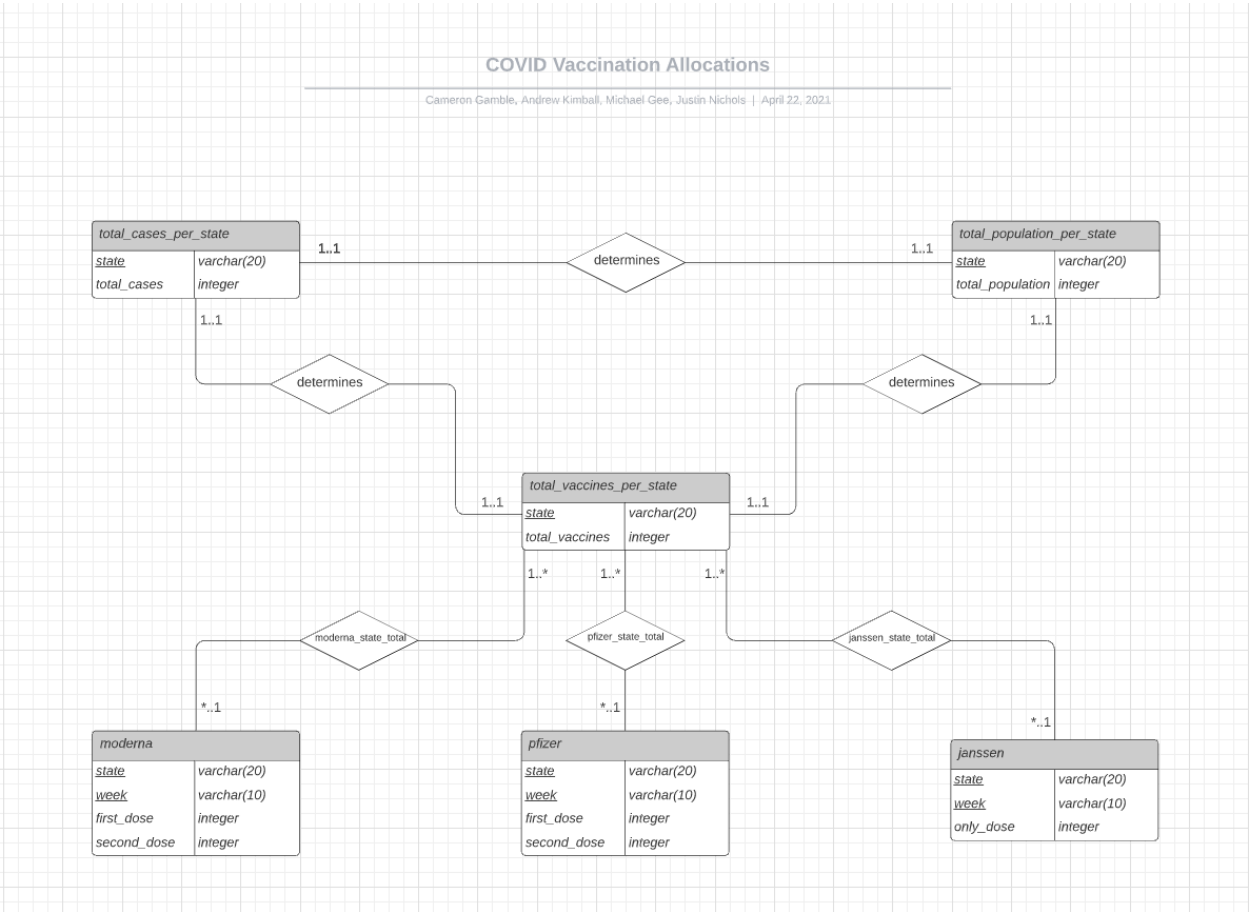
- Within the total_cases_per_state entity, the attribute *state* must be a varchar(20), not null, unique. The attribute *total_cases* must be an integer and not null. If no cases exist for a particular state, 0 will be the value.
- Within the total_population_per_state entity, the attribute *state* must be a varchar(20), not null, unique. The attribute *total_population* must be an integer and not null. There will always be a population value higher than 0.
- Within the total_vaccines_per_state entity, the attribute *state* must be a varchar(20), not null, unique. The attribute *total_vaccines* must be an integer and not null. If no vaccines exist for a particular state, 0 will be the value.
- Within the moderna and pfizer entities, the attribute *state* must be a varchar(20) and not null. The attribute *week* must be a varchar(10) and not null. The attributes *first_dose* and *second_dose* must be integers. If no doses exist for a particular state during a week, 0 will be the value.
- Within the janssen entity, the attribute *state* must be a varchar(20) and not null. The attribute *week* must be a varchar(10) and not null. The attribute *only_dose* must be an integer. If no doses exist for a particular state during a week, 0 will be the value.

2.3 Assumptions

- Assume that every state has a population, case, and vaccine count. These values could range from 0 to any reasonable positive number.

- Assume that the vaccine data can drastically change depending on the week due to the fast-changing environment the world is in currently.
- Assume that the Moderna and Pfizer vaccines require two doses.
- Assume that the Janssen vaccine requires only one dose.
- Assume that Moderna, Pfizer, and Janssen all had different authorization dates. Whenever a particular vaccine was made available to the public is when the data starts for it.
- Assume for each week for a particular vaccine includes all states. There is no week where a state is missing a value.

2.4 Entity-Relationship Model



3 Database Design

We constructed six primary tables that are of importance within our database: total_cases_per_state, total_population_per_state, total_vaccines_per_state, moderna, pfizer, and janssen. Below is a detailed representation of each, including the attributes, constraint types, keys, functional dependencies, and normal forms. All tables are in BCNF due to no non-primary attributes having dependencies with other non-primary attributes. There also is a description on what specific data each table holds.

3.1 Tables

3.11 total_cases_per_state

This table holds the total number of COVID-19 cases per state. Each entry is unique and can be identified using the state. There are 59 total entries in this table.

total_cases_per_state (state varchar(20) not null unique, total_cases integer not null, primary key(state))
Primary Key: state
Functional Dependencies: state → total_cases
Normal Form: The table schema is in BCNF.

```
MariaDB [cs440238]> describe total_cases_per_state;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| state       | varchar(20) | NO   | PRI | NULL    |       |
| total_cases | int(11)     | NO   |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
2 rows in set (0.002 sec)
```

3.12 total_population_per_state

This table holds the total population per state. Each entry is unique and can be identified using the state. There are 59 total entries in this table.

total_population_per_state (state varchar(20) not null unique, total_population integer not null, primary key(state))
Primary Key: state
Functional Dependencies: state → total_population
Normal Form: The table schema is in BCNF.

```
MariaDB [cs440238]> describe total_population_per_state;
+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra |
+------------------+-------------+------+-----+---------+-------+
| state            | varchar(20) | NO   | PRI | NULL    |       |
| total_population | int(11)     | NO   |     | NULL    |       |
+------------------+-------------+------+-----+---------+-------+
2 rows in set (0.004 sec)
```

3.13 total_vaccines_per_state

This table holds the total vaccines per state. The way we obtained the value for each state was taking just the second_dose count from both Moderna and Pfizer and the only_dose count from

Janssen. The idea behind this was to show the number of people fully vaccinated with each vaccine. Since both the Moderna and Pfizer vaccines require two doses and the Janssen vaccines only required one dose, taking only the dose that dictated a fully vaccinated person was necessary. Each entry is unique and can be identified using the state. There are 63 total entries in this table.

total_vaccines_per_state (state varchar(20) not null unique, total_vaccines integer not null, primary key(state))
Primary Key: state
Functional Dependencies: state → total_vaccines
Normal Form: The table schema is in BCNF.

```
MariaDB [cs440238]> describe total_vaccines_per_state;
+---------------+------------+------+-----+---------+-------+
| Field         | Type       | Null | Key | Default | Extra |
+---------------+------------+------+-----+---------+-------+
| state         | varchar(20) | NO   | PRI | NULL    |       |
| total_vaccines | int(11)    | NO   |     | NULL    |       |
+---------------+------------+------+-----+---------+-------+
2 rows in set (0.011 sec)
```

3.14 moderna

This table holds the total first dose and second dose allocations by state per week. The date of data obtained starts from the week of 21 December 2020 and goes until the week of 5 April 2021. Each entry is unique and can be identified using the state and the week. There are 1008 total entries in this table.

moderna (state varchar(20) not null, week varchar(10) not null, first_dose integer, second_dose integer, primary key(state, week))
Primary Key: state, week
Functional Dependencies: state, week → first_dose; state, week → second_dose
Normal Form: The table schema is in BCNF.

```
MariaDB [cs440238]> describe moderna;
+-------------+------------+------+-----+---------+-------+
| Field       | Type       | Null | Key | Default | Extra |
+-------------+------------+------+-----+---------+-------+
| state       | varchar(20) | NO   | PRI | NULL    |       |
| week        | varchar(10) | NO   | PRI | NULL    |       |
| first_dose  | int(11)    | YES  |     | NULL    |       |
| second_dose | int(11)    | YES  |     | NULL    |       |
+-------------+------------+------+-----+---------+-------+
4 rows in set (0.002 sec)
```

3.14 pfizer

This table holds the total first dose and second dose allocations by state per week. The date of data obtained starts from the week of 14 December 2020 and goes until the week of 5 April 2021. Each entry is unique and can be identified using the state and the week. There are 1071 total entries in this table.

pfizer (state varchar(20) not null, week varchar(10) not null, first_dose integer, second_dose integer, primary key(state, week)
Primary Key: state, week
Functional Dependencies: state, week → first_dose; state, week → second_dose
Normal Form: The table schema is in BCNF.

```
MariaDB [cs440238]> describe pfizer;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| state       | varchar(20) | NO   | PRI | NULL    |       |
| week        | varchar(10) | NO   | PRI | NULL    |       |
| first_dose  | int(11)     | YES  |     | NULL    |       |
| second_dose | int(11)     | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.003 sec)
```

3.16 janssen

This table holds the total dose allocations by state per week. The date of data obtained starts from the week of 1 March 2020 and goes until the week of 5 April 2021. Each entry is unique and can be identified using the state and the week. There are 315 total entries in this table.

janssen (state varchar(20) not null, week varchar(10) not null, only_dose integer, primary key(state, week))
Primary Key: state, week
Functional Dependencies: state, week → only_dose
Normal Form: The table schema is in BCNF.

```
MariaDB [cs440238]> describe janssen;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| state     | varchar(20) | NO   | PRI | NULL    |       |
| week      | varchar(10) | NO   | PRI | NULL    |       |
| only_dose | int(11)     | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.002 sec)
```

3.2 Create Table Statements

```
#Create Table total_cases_per_state
drop table if exists total_cases_per_state;
create table if not exists total_cases_per_state (
        state varchar(20) not null unique,
        total_cases integer not null,
        primary key(state)
);
load data local infile 'Cases.csv' into table total_cases_per_state fields terminated by ',';
```

```
#Create Table total_population_per_state
drop table if exists total_population_per_state;
create table if not exists total_population_per_state (
        state varchar(20) not null unique,
        total_population integer not null,
        primary key(state)
);
load data local infile 'Population.csv' into table total_population_per_state fields terminated by ',';
```

```
#Create Table moderna
drop table if exists moderna;
create table if not exists moderna (
        state varchar(20) not null,
        week varchar(10) not null,
        first_dose integer,
        second_dose integer,
        primary key(state, week)
);
load data local infile 'Moderna.csv' into table moderna fields terminated by ',';
```

```
#Create Table pfizer
drop table if exists pfizer;
create table if not exists pfizer (
        state varchar(20) not null,
        week varchar(10) not null,
        first_dose integer,
        second_dose integer,
        primary key(state, week)
);
load data local infile 'Pfizer.csv' into table pfizer fields terminated by ',';
```

```
#Create Table janssen
drop table if exists janssen;
create table if not exists janssen (
        state varchar(20) not null,
        week varchar(10) not null,
        only_dose integer,
        primary key(state, week)
);
load data local infile 'Janssen.csv' into table janssen fields terminated by ',';
```

```
#Create Table total_vaccines_per_state
drop table if exists total_vaccines_per_state;
create table if not exists total_vaccines_per_state (
        state varchar(20) not null unique,
        total_vaccines integer not null,
        primary key(state)
)
as (select t1.s1 as state, (t1.total_vaccines1 + t2.total_vaccines2 + t3.total_vaccines3) as total_vaccines from (
(select state as s1, sum(second_dose) as total_vaccines1 from moderna group by state) as t1,
(select state as s2, sum(second_dose) as total_vaccines2 from pfizer group by state) as t2,
(select state as s3, sum(only_dose) as total_vaccines3 from janssen group by state) as t3)
where t1.s1 = t2.s2 and t2.s2 = t3.s3 group by t1.s1);
```

3.3 Obtaining the Raw Data

All data was gathered in a csv format during the first week of April 2021. The case count data for each state was obtained from https://www.cdc.gov/. The population data for each state was obtained from https://www.census.gov/. Considering the official 2020 census population data is currently unavailable during the construction of this project, we used the estimated 2019 population data. The vaccine data corresponding to Moderna, Pfizer, and Janssen was obtained from https://www.data.gov/. The final raw data sets included, Moderna.csv, Pfizer.csv, Janssen.csv, Population.csv, and Cases.csv.

The Population.csv and Cases.csv files were made by scratch using the above resources mentioned.
However, here are the exact links for the Moderna.csv, Pfizer.csv, and Janssen.csv files.
Moderna.csv: https://catalog.data.gov/dataset/covid-19-vaccine-distribution-allocations-by-jurisdiction-moderna
Pfizer.csv:
https://catalog.data.gov/dataset/covid-19-vaccine-initial-allocations-pfizer
Janssen.csv:
https://catalog.data.gov/dataset/covid-19-vaccine-distribution-allocations-by-jurisdiction-janssen

4 SQL Queries

Query Matrix:										
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1.Join	X	X	X						X	X
2. Subquery				X	X		X	X	X	X
3.GROUP BY						X			X	X
4.GROUP BY with HAVING										
5. ORDER BY	X	X	X			X			X	X
6. Aggregate				X	X		X	X	X	X
7. String operations (like)						X				
8. Set operations										
9. IN/NOT										
10. IS NULL									X	

4.1 Query 1

This query finds the percentage of vaccines given to a state compared to its population and outputs the top ten states with the highest percentage.

```
select v.state, (total_vaccines / total_population) as vaccine_to_population_ratio
from total_vaccines_per_state v inner join total_population_per_state p
on v.state = p.state order by vaccine_to_population_ratio desc limit 10;
```

Result:

```
+----------------------+----------------------------+
| state                | vaccine_to_population_ratio |
+----------------------+----------------------------+
| Alaska               |                     0.6058 |
| U.S. Virgin Islands  |                     0.4188 |
| Vermont              |                     0.3668 |
| Puerto Rico          |                     0.3646 |
| District of Columbia |                     0.3564 |
| West Virginia        |                     0.3506 |
| Maine                |                     0.3484 |
| Rhode Island         |                     0.3483 |
| New Hampshire        |                     0.3445 |
| Hawaii               |                     0.3443 |
+----------------------+----------------------------+
```

4.2 Query 2

This query finds the percent of each state's cases compared to its population and outputs the top ten states with the highest percentage.

```
select c.state, (total_cases / total_population) as case_to_population_ratio
from total_cases_per_state c inner join total_population_per_state p
on c.state = p.state order by case_to_population_ratio desc limit 10;
```

Result:

```
+--------------+--------------------------+
| state        | case_to_population_ratio |
+--------------+--------------------------+
| North Dakota |                   0.1350 |
| South Dakota |                   0.1328 |
| Rhode Island |                   0.1291 |
| Utah         |                   0.1201 |
| Tennessee    |                   0.1187 |
| Arizona      |                   0.1156 |
| Iowa         |                   0.1112 |
| Oklahoma     |                   0.1107 |
| Arkansas     |                   0.1094 |
| Wisconsin    |                   0.1091 |
+--------------+--------------------------+
```

4.3 Query 3

This query finds the percentage of vaccines given to a state compared to its cases and outputs the top ten states with the highest ratio.

```
select c.state, (total_vaccines / total_cases) as vaccine_to_cases_ratio
from total_cases_per_state c inner join total_vaccines_per_state v
on c.state = v.state order by vaccine_to_cases_ratio desc limit 10;
```

Result:

```
+---------------------+-----------------------+
| state               | vaccine_to_cases_ratio |
+---------------------+-----------------------+
| Marshall Islands    |             2700.0000 |
| Hawaii              |               15.9033 |
| U.S. Virgin Islands |               15.3629 |
| Vermont             |               11.9789 |
| Puerto Rico         |               10.8707 |
| Maine               |                9.3182 |
| Oregon              |                8.4289 |
| Alaska              |                7.0573 |
| Washington          |                6.7883 |
| District of Columbia |               5.6638 |
+---------------------+-----------------------+
```

4.4 Query 4

This query finds the total number of Moderna, Pfizer, and Janssen vaccines distributed to fully vaccinate a person.

```
select * from (
        (select sum(second_dose) as moderna from moderna) as t1,
        (select sum(second_dose) as pfizer from pfizer) as t2,
        (select sum(only_dose) as janssen from janssen) as t3
);
```

Result:

```
+----------+----------+----------+
| moderna  | pfizer   | janssen  |
+----------+----------+----------+
| 49711600 | 53733030 | 10575700 |
+----------+----------+----------+
```

4.5 Query 5

This query finds the percent of vaccines given to each state compared to its population and takes the average. The output represents how much of the U.S. is covered by the number of current vaccines available.

```
select avg(vaccine_to_population_ratio) as avg_ratio from
        (select (total_vaccines / total_population) as vaccine_to_population_ratio from
        total_vaccines_per_state v, total_population_per_state p where v.state = p.state)
as state_vaccinations;
```

Result:

```
+-------------+
| avg_ratio   |
+-------------+
| 0.31679572 |
+-------------+
```

4.6 Query 6

This query finds the population, cases, and vaccines of states that start with ‘M’ or ‘N’ that have a population, case count, and vaccine number over 500,000.

```
select p.state, total_population, total_cases, total_vaccines
from total_population_per_state p, total_cases_per_state c, total_vaccines_per_state v
where p.state = c.state and c.state = v.state and p.state regexp '^M|^N' and
total_population >= 5000000 and total_cases >= 500000 and total_vaccines >= 500000
group by p.state order by p.state;
```

Result:

```
+---------------+-----------------+-------------+----------------+
| state         | total_population | total_cases | total_vaccines |
+---------------+-----------------+-------------+----------------+
| Massachusetts |         6892503 |      633081 |        2334840 |
| Michigan      |         9986857 |      739244 |        3317920 |
| Minnesota     |         5639632 |      517881 |        1820165 |
| Missouri      |         6137428 |      580980 |        2017930 |
| New Jersey    |         8882190 |      905144 |        2955860 |
| New York      |        19453561 |     1865349 |        3759985 |
| North Carolina |        10488084 |      912203 |        3365580 |
+---------------+-----------------+-------------+----------------+
```

4.7 Query 7

This query finds the total numbers of vaccines distributed. This includes both first and second doses.

```
select (m1.total + m2.total + p1.total + p2.total + j1.total) as total_vaccines from (
        (select sum(m.first_dose) as total from moderna m) as m1,
        (select sum(m.second_dose) as total from moderna m) as m2,
        (select sum(p.first_dose) as total from pfizer p) as p1,
        (select sum(p.second_dose) as total from pfizer p) as p2,
        (select sum(j.only_dose) as total from janssen j) as j1
);
```

Result:

```
+----------------+
| total_vaccines |
+----------------+
|      217975090 |
+----------------+
```

4.8 Query 8

This query uses each vaccine's effectiveness to approximately find the number of people that still contracted COVID-19 after getting the vaccine.

```
select
        ((select sum(m.second_dose) from moderna m) * (1 - 0.941)) as moderna_infected,
        ((select sum(p.second_dose) from pfizer p) * (1 - 0.95)) as pfizer_infected,
        ((select sum(j.only_dose) from janssen j) * (1 - 0.72)) as janssen_infected;
```

Result:

```
+------------------+-----------------+------------------+
| moderna_infected | pfizer_infected | janssen_infected |
+------------------+-----------------+------------------+
|      2932984.400 |      2686651.50 |       2961196.00 |
+------------------+-----------------+------------------+
```

4.9 Query 9

This query finds the total number of vaccines distributed each week across the country.

```
select p_total.week, (ifnull(p_total.weekly_doses,0) + ifnull(m_total.weekly_doses,0)
+ ifnull(j_total.weekly_doses,0)) as weekly_vaccine_total from (
        (select week, sum(first_dose + second_dose) as weekly_doses from pfizer p group by week) as p_total
        left join
        (select week, sum(first_dose + second_dose) as weekly_doses from moderna m group by week) as m_total
        on p_total.week = m_total.week
        left join
        (select week, sum(only_dose) as weekly_doses from janssen j group by week) as j_total
        on m_total.week = j_total.week
) order by weekly_vaccine_total desc;
```

Result:

week	weekly_vaccine_total
4/5/21	21223620
3/29/21	20207120
3/1/21	17673560
3/22/21	16284880
3/15/21	16217860
12/21/20	16082350
3/8/21	15298380
2/22/21	13500040
2/8/21	11159750
2/15/21	10999750
2/1/21	10166530
12/28/20	9381050
1/25/21	8600350
1/18/21	8599750
1/4/21	8392500
1/11/21	8263500
12/14/20	5924100

4.10 Query 10

This query finds the state that obtained the highest percentage of vaccines compared to its population for each of the three vaccines.

```
select * from (
        (select vac1.state as moderna_state, max(vac1.moderna_percentage) as max_moderna_percentage from
                (select m.state, (sum(m.second_dose) / p.total_population) as moderna_percentage from
                moderna m join total_population_per_state p on m.state = p.state group by m.state
                ) as vac1
        group by vac1.moderna_percentage order by max_moderna_percentage desc limit 1) as max1,
        (select vac2.state as pfizer_state, max(vac2.pfizer_percentage) as max_pfizer_percentage from
                (select pf.state, (sum(pf.second_dose) / p.total_population) as pfizer_percentage from
                pfizer pf join total_population_per_state p on pf.state = p.state group by pf.state
                ) as vac2
        group by vac2.pfizer_percentage order by max_pfizer_percentage desc limit 1) as max2,
        (select vac3.state as janssen_state, max(vac3.janssen_percentage) as max_janssen_percentage from
                (select j.state, (sum(j.only_dose) / p.total_population) as janssen_percentage from
                janssen j join total_population_per_state p on j.state = p.state group by j.state
                ) as vac3
        group by vac3.janssen_percentage order by max_janssen_percentage desc limit 1) as max3
);
```

Result:

moderna_state	max_moderna_percentage	pfizer_state	max_pfizer_percentage	janssen_state	max_janssen_percentage
Alaska	0.2549	Alaska	0.3049	Palau	0.2110

5 How to Use the Database Application

1. Download the Team13_Project.zip file locally to your computer
2. Extract the Team13_Project.zip file
3. Open a terminal and go into the CSC class server using *ssh* *cs4402xx@classes.csc.lsu.edu*
4. In the class server, create a new directory using *mkdir project*
5. Open a second terminal and use *cd* to get to where the Team13_Project folder is located
6. Once inside the Team13_Project folder in the second terminal, use *scp * cs4402XX@classes.csc.lsu.edu:~/project/* to transfer the files over to the class server
7. You can now close the second terminal. In the terminal that is connected to the class server, *cd project* to get into the project directory
8. To create the tables and run the queries, enter into MariaDB using *mysql -ucs4402XX -pxxxxxx*
9. Switch to your databases using *use cs4402XX;*
10. Run the queries and output the tables using *source Team13_SQL.sql;*

6 Conclusion

Throughout the development and implementation of this project, our team has learned the benefits of utilizing the database model and the theory of logical design to construct a complex database application. By understanding data creation, fetching, organization, and manipulation, we were able to use SQL to find valuable connections in the data and form conclusions about COVID-19. The data acquired allows for COVID-19 statistics to be more clearly seen on how the virus has affected each state differently and how that has ultimately determined the current allocation rate each state has currently. Through the use of ER modeling, we were able to abstractly represent our COVID-19 database in the form of entities and relationships. Although this was not a simple task, the use of techniques we have learned throughout this semester aided us in producing a finished product.

Finding data on our subject matter led to some difficulties due to it being so recently available. We find that it would probably be beneficial to work with more complete data sets made after events have taken place. This could allow us to draw better insights from more fleshed-out data. Ultimately, we have learned the importance that database design takes part in creating a successful and useful application.