Text

Description automatically generated

BC2402 DESIGNING & DEVELOPING DATABASES

Group Project on COVID-19

AY21/22 SEM 1 | SEMINAR 6, GROUP 2  
Instructor: Prof Teoh Teik Toe

| Matric No. | Group Member Name |
| --- | --- |
| U1920048B | Tan Jun Hong |
| U2021421C | Khoo Teng Khing, Joshua |
| U2021729K | Leow Ken Hing Bryan |
| U2021955A | Ng Zheng Jie |
| U2022992A | Loh Yi Ze |

**Table of Contents**

[**1. Introduction**](#_heading=h.gjdgxs) **[1](#_heading=h.gjdgxs)**

[1.1 Purpose](#_heading=h.30j0zll) [1](#_heading=h.30j0zll)

[1.2 Summary of Recommendations](#_heading=h.womzj4sfhio7) [1](#_heading=h.womzj4sfhio7)

[**2. Relational Database Design**](#_heading=h.1fob9te) **[2](#_heading=h.1fob9te)**

[2.1 Introduction to Relational Databases / mySQL](#_heading=h.3znysh7) [2](#_heading=h.3znysh7)

[2.2 Principles in Relational Database Design](#_heading=h.2et92p0) [2](#_heading=h.2et92p0)

[2.2.1 Dependencies](#_heading=h.a0gvt5bvk4n1) [2](#_heading=h.a0gvt5bvk4n1)

[2.2.2 Steps in Normalisation](#_heading=h.1nqjk7ko7hme) [2](#_heading=h.1nqjk7ko7hme)

[2.3 Challenges Presented by the Data and our Solutions](#_heading=h.cdsm201kwp3j) [2](#_heading=h.cdsm201kwp3j)

[2.3.1 Data Cleaning](#_heading=h.f85tmba925b4) [2](#_heading=h.f85tmba925b4)

[2.3.2 Data Inconsistency between Sources](#_heading=h.59wghenjwobj) [4](#_heading=h.59wghenjwobj)

[2.3.2.1 Number Discrepancy in Overlapping Data](#_heading=h.uuu9ukrvb64z) [4](#_heading=h.uuu9ukrvb64z)

[2.3.2.2 Missing Data](#_heading=h.3otq65hnwvry) [5](#_heading=h.3otq65hnwvry)

[2.3.2 Data Normalisation](#_heading=h.oxb8dr1gt9au) [5](#_heading=h.oxb8dr1gt9au)

[2.3.2.1 Getting to First Normal Form](#_heading=h.1i8mgnlzj3i2) [5](#_heading=h.1i8mgnlzj3i2)

[2.3.2.1 Getting to Second Normal Form](#_heading=h.s2a51ang1wj1) [7](#_heading=h.s2a51ang1wj1)

[2.3.2.2 Getting to Third Normal Form](#_heading=h.d4jcagrsvx4j) [7](#_heading=h.d4jcagrsvx4j)

[2.3.2.3 Optimising the Database](#_heading=h.hcrqsqiuwrvm) [8](#_heading=h.hcrqsqiuwrvm)

[2.3.2.4 Cleaning up the Database](#_heading=h.d7y6wdspqyk7) [9](#_heading=h.d7y6wdspqyk7)

[2.4 Our Proposed Entity Relationship Diagram (“ERD”)](#_heading=h.qnqxphgyfkdf) [10](#_heading=h.qnqxphgyfkdf)

[2.5 Deploying our mySQL Database Implementation](#_heading=h.3yum1d61wfs0) [11](#_heading=h.3yum1d61wfs0)

[**3. Nonrelational Database Design**](#_heading=h.mxw2tfteggx) **13**

[3.1 Introduction to Non-Relational Databases / mongoDB](#_heading=h.l2jmzdmcz4ii) [13](#_heading=h.l2jmzdmcz4ii)

[3.2 Principles in Non-Relational Database Design](#_heading=h.8wi7r5el1bov) [13](#_heading=h.8wi7r5el1bov)

[3.3 Challenges Presented by the Data and our Solutions](#_heading=h.z1zalenstw49) [13](#_heading=h.z1zalenstw49)

[3.3.1 Choosing the COVID-19 Dataset](#_heading=h.vfqd87pmjbve) [13](#_heading=h.vfqd87pmjbve)

[3.3.2 Data Cleaning](#_heading=h.v2e1auumh6wy) [14](#_heading=h.v2e1auumh6wy)

[3.3.3 Importing](#_heading=h.pfqs2mggg1cp) [15](#_heading=h.pfqs2mggg1cp)

[3.3.4 Merging](#_heading=h.lbe3ps768hin) [15](#_heading=h.lbe3ps768hin)

[3.3.5 Exporting](#_heading=h.dioxj86fnes0) [15](#_heading=h.dioxj86fnes0)

[3.3.6 Advantages of merging collections now press refresh on the content table on the top left](#_heading=h.b3nbvxkjoox8) [16](#_heading=h.b3nbvxkjoox8)

[3.4 Deploying our mongoDB Database Implementation](#_heading=h.qgiafrzi9hv6) [17](#_heading=h.qgiafrzi9hv6)

[**4. Our Recommendations to WHO**](#_heading=h.jo7t817ot5g) **[19](#_heading=h.jo7t817ot5g)**

[4.1 Comparisons/Inconsistencies between our Relational and Non-Relational Database Implementations](#_heading=h.tofaf9zhx7uc) [19](#_heading=h.tofaf9zhx7uc)

[4.2 Justification for our Recommendation](#_heading=h.o621oz7qu6me) [19](#_heading=h.o621oz7qu6me)

[**Appendix B – noSQL queries**](#_heading=h.eago06ao6yfk) **[27](#_heading=h.eago06ao6yfk)**

[**Appendix C – Distribution of Work**](#_heading=h.vd10qwr20sdr) **38**

# 1. Introduction

## 1.1 Purpose

COVID-19 has been a problem for governments around the world since 2020. Due to the widespread nature of the virus, there are tons of data on COVID-19, relating to deaths, vaccinations and many more. There is an increasing need to model this data, as it can provide useful insights to tackling the virus and perhaps give an idea if the country is out of the COVID-19 crisis. As such, it is important to store the data properly, so as to facilitate queries and make the retrieval of data more efficient and simple.

## 1.2 Summary of Recommendations

Due to the unstructured nature of the data, there are many fields that are empty or null, which suggests that a relational database may not be the best way to represent the data.

The ever-changing and dynamic nature of the COVID-19 situation also warrants a need for greater flexibility

As such, NoSQL seems to be the better choice for storing the data, and it is more flexible if there is a need to include new fields/data in the future, since we do not need to define a schema beforehand. We will compare both relational and non relational databases in detail below, and give our detailed explanation as to why NoSQL is chosen.

# 2. Relational Database Design

## 2.1 Introduction to Relational Databases / mySQL

In the 1970s, Edgar F.Codd introduced the term “relational database” in his research paper “A Relational Model of Data for Large Shared Data Banks”. MySQL is an open-source implementation of relational databases.

## 2.2 Principles in Relational Database Design

A relational database is a collection of tables, and each table has a collection of rows which serve as the records of the table. In each row, there are attributes: pieces of data that are related to the record. Before we discuss our database implementation, we will first introduce the relevant terminology and principles:

### 2.2.1 Dependencies

Before discussing normalisation, we define functional dependency as the following:

| 2.2.1.1 | Full dependency: The primary key fully determines the value of another non-key attribute |
| --- | --- |
| 2.2.1.2 | Partial dependency: The primary key partially determines the value of another non-key attribute. |
| 2.2.1.3 | Transitive dependency: A non-key field determines the value of another non-key attribute. |

### 2.2.2 Steps in Normalisation

*Normalisation* is an essential concept in relational database design that aims to avoid unnecessary duplication of data through smaller, well-structured relations. We list the requirements for the different forms in normalisation. Each Normal Form is a prerequisite of the next.

| 2.2.2.1 | First Normal Form: no multi-valued attributes, every attribute value is atomic |
| --- | --- |
| 2.2.2.2 | Second Normal Form: every non-key attribute is fully and functionally dependent on the ENTIRE primary key, i.e. no partial dependencies. |
| 2.2.2.3 | Third Normal Form: No transitive dependencies. |

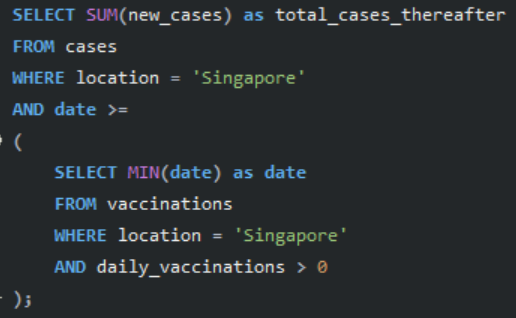
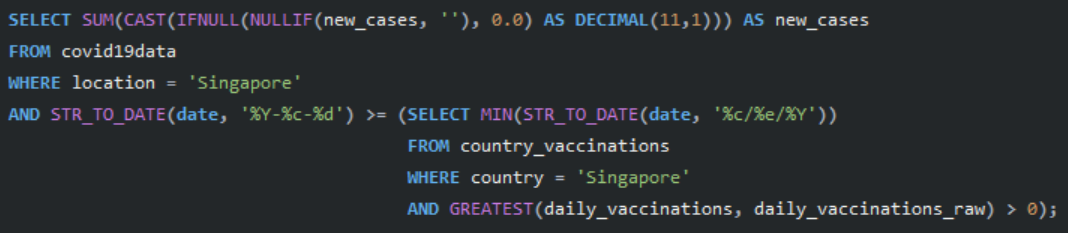
## 2.3 Challenges Presented by the Data and our Solutions

### 2.3.1 Data Cleaning

Missing values in the data are formatted as an empty string. These values not provided by the data should be formatted as NULL instead.

In addition, some columns in the data are in the incorrect data type, having a generic type of ‘text’ instead of, for example, ‘int’ for numbers and ‘date’ for date. In most circumstances, this will not a pose an immediate challenge, since mySQL automatically does type conversion when needed[[1]](#footnote-0).

However, for more complex cases e.g. involving dates, it is mandatory for the attribute to be of date data type to utilise date-related functions such as DATEDIFF(). This forces users to perform the data type conversion in the SQL query itself, which reduces readability and query development time.



*Figure 1: Two SQL queries written for Question 6, before and after data cleaning. The difference in readability demonstrates the importance of having cleaned, properly-formatted databases.*

Enforcing correct data types for the attributes will also help in identifying potential data type mismatches in the data attributes, e.g. caused by errors in data entry.

For our implementation, we converted all empty strings to null and converted each attribute into the most appropriate data type, out of the below four:

| DATE | Used for attributes containing date. |
| --- | --- |
| INT | Used for numbers that will not exceed 2 billion.[[2]](#footnote-1) |
| BIGINT | Used for numbers that might exceed 2 billion.  (for reference, the world population stands at approximately 8 billion as of 2021) |
| DECIMAL | Used for numbers that have a decimal component. |

A snippet of the mySQL script used to perform data cleaning is shown:

| # 1. Create a new table called "country\_vaccinations\_cleaned"  CREATE TABLE country\_vaccinations\_cleaned AS SELECT \* FROM country\_vaccinations;  # 2. Change the formatting of the DATE column to proper mySQL format.  # We need to do this, else mySQL cannot modify the column to be DATE datatype in the next step  # Note: need to set SAFE\_UPDATES flag to off because we are not specifying a primary key in the WHERE clause. We don't need to, because every row has to be updated.  # We set the flag back on afterward  SET SQL\_SAFE\_UPDATES = 0;  UPDATE country\_vaccinations\_cleaned SET date = STR\_TO\_DATE(date, '%c/%e/%Y');  SET SQL\_SAFE\_UPDATES = 1;  # 3. Change the data types  ALTER TABLE country\_vaccinations\_cleaned  #country --> TEXT  #iso\_code --> TEXT  MODIFY COLUMN date DATE,  MODIFY COLUMN total\_vaccinations BIGINT,  MODIFY COLUMN people\_vaccinated BIGINT,  MODIFY COLUMN people\_fully\_vaccinated BIGINT,  MODIFY COLUMN daily\_vaccinations\_raw INT,  MODIFY COLUMN daily\_vaccinations INT,  MODIFY COLUMN total\_vaccinations\_per\_hundred DECIMAL(12,3),  MODIFY COLUMN people\_vaccinated\_per\_hundred DECIMAL(12,3),  MODIFY COLUMN people\_fully\_vaccinated\_per\_hundred DECIMAL(12,3),  MODIFY COLUMN daily\_vaccinations\_per\_million INT,  MODIFY COLUMN vaccines TEXT,  MODIFY COLUMN source\_name TEXT,  MODIFY COLUMN source\_website TEXT; |
| --- |

The full script can be found in database\_normalisation\_script\_sem6\_grp2.sql.

### 2.3.2 Data Inconsistency between Sources

#### 2.3.2.1 Number Discrepancy in Overlapping Data

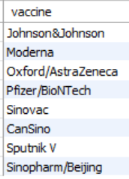
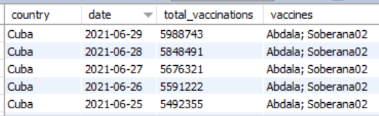
Inspecting total vaccination numbers in the 3 data sources, we find discrepancies in the numbers reported:

| Data Source | Screenshot | total\_vaccinations for Austria, 2021-06-29 |
| --- | --- | --- |
| country\_vaccinations\_by\_manufacturer |  | 7631324 (106817+753994+1300224+5470289) |
| country\_vaccinations |  | 7639269 |
| covid19data |  | 7648954 |

While this difference is around 0.2% and well within expectations for real-world data, it still poses a concern when normalising our database and writing our queries. For this purpose, we standardised by using the number reported from *country\_vaccinations* for daily aggregates, and continued using *country\_vaccinations\_by\_manufacturer* for queries that require manufacturer-specific information.

#### 2.3.2.2 Missing Data

In addition, when listing the distinct vaccines found in *country\_vaccinations\_by\_manufacturer*, we obtain the list pictured on the left. This list excludes vaccines such as ‘Abdala’ and ‘Soberana02’, vaccines used by Cuba according to the *country\_vaccinations* source.

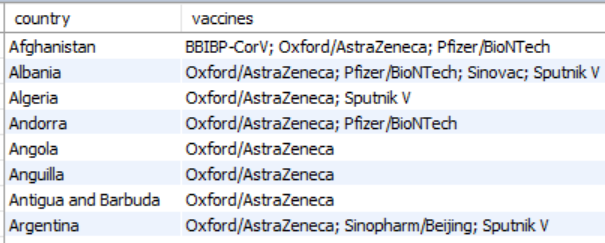
*Figure 2: List of vaccines found in country\_vaccinations\_by\_manufacturer, vs entries in country\_vaccinations for Cuba. This illustrates the issue of missing data in real-world datasets with multiple sources.*

We discuss the implication of this in the next section on data normalisation.

### 2.3.2 Data Normalisation

#### 2.3.2.1 Getting to First Normal Form

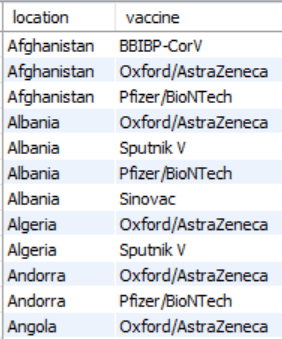
With reference to 2.2.2.1, there is a need to handle the multi-valued vaccines attribute shown in Figure 3.



*Figure 3: multi-valued attributes in the vaccines attribute of the country\_vaccinations data*

However, we are unable to simply drop this attribute following the discussion in 2.3.2.2, since doing so would cause us to lose valuable data. We must hence decompose this attribute.

A country can have many vaccines, and a vaccine can belong to many countries. This is hence a many-to-many relationship, and can be decomposed via an intersection table, as seen in Figure 4, to form a weak entity. *location* and *vaccine* are foreign keys from the location and vaccine table, forming the composite primary key of this intersection table.



*Figure 4: the intersection table between location (country) and vaccine*

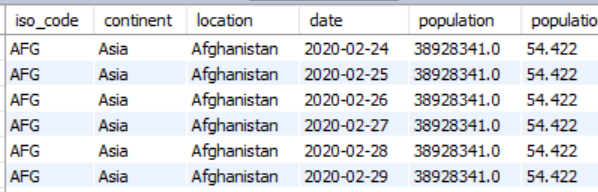
A snippet of the mySQL script used to perform this decomposition is shown:

| ...  # 2. Splice the vaccines column into 6 different columns and save it as a view.  CREATE VIEW spliced AS SELECT  country,  SUBSTRING\_INDEX(vaccines, ';', 1) AS vaccine1,  SUBSTRING\_INDEX(SUBSTRING\_INDEX(vaccines, '; ', 2), '; ', -1) AS vaccine2,  SUBSTRING\_INDEX(SUBSTRING\_INDEX(vaccines, '; ', 3), '; ', -1) AS vaccine3,  SUBSTRING\_INDEX(SUBSTRING\_INDEX(vaccines, '; ', 4), '; ', -1) AS vaccine4,  SUBSTRING\_INDEX(SUBSTRING\_INDEX(vaccines, '; ', 5), '; ', -1) AS vaccine5,  SUBSTRING\_INDEX(SUBSTRING\_INDEX(vaccines, '; ', 6), '; ', -1) AS vaccine6  FROM country\_vaccinations  GROUP BY country;  # 3. Union all the splices together into a new intersection table.  CREATE TABLE locations\_vaccines AS  SELECT country as location, vaccine1 as vaccine FROM spliced  UNION SELECT country as location, vaccine2 as vaccine FROM spliced  UNION SELECT country as location, vaccine3 as vaccine FROM spliced  UNION SELECT country as location, vaccine4 as vaccine FROM spliced  UNION SELECT country as location, vaccine5 as vaccine FROM spliced  UNION SELECT country as location, vaccine6 as vaccine FROM spliced  GROUP BY location, vaccine1  ORDER BY location ASC;  SELECT \* FROM locations\_vaccines;  # 4. Drop the vaccines column from the original  ALTER TABLE country\_vaccinations\_cleaned  DROP vaccines;  # 5. Create the vaccine table  CREATE TABLE vaccine AS  SELECT vaccine FROM locations\_vaccines  GROUP BY vaccine  ORDER BY vaccine ASC; |
| --- |

The full script can be found in database\_normalisation\_script\_sem6\_grp2.sql.

#### 2.3.2.1 Getting to Second Normal Form

With reference to 2.2.2.2, we also need to remove attributes related to a location in *covid19\_data.*

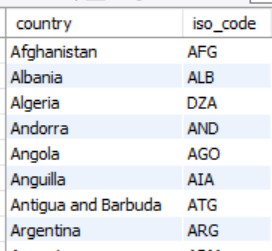


*Figure 5: location information e.g. population causing partial dependencies.*

In *covid19\_data*, the primary key is a composite key of location and date. Location attributes like population, however, are only dependent on location and not date, resulting in a partial dependency. It also additionally results in anomalies. For example, there is insertion anomaly since adding a new row would force users to duplicate location attributes. There is also modification anomaly, since changing Afghanistan’s population would force changes to all other rows containing Afghanistan, for instance. Hence, we put these columns in a separate *location* table with *location* as the primary key. This *location* then becomes a foreign key in the original data source.

#### 2.3.2.2 Getting to Third Normal Form

With reference to 2.2.2.3, we also needed to separate *location* and *iso\_code*. This is because *location* and *iso\_code* are both candidate keys - location attributes can be derived from *iso\_code*, and *iso\_code* can be derived from *country*. This creates a transitive dependency where a functional dependency exists on the non-key field *iso\_code*. We hence place *iso\_code* in a separate table with *location (country)* as the foreign key from the *location* table.



*Figure 6: location and iso\_code in the same table causing transitive dependencies*

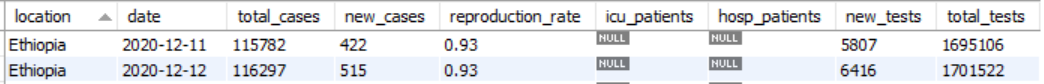
A snippet of the mySQL script used to get to second and third normal form is shown:

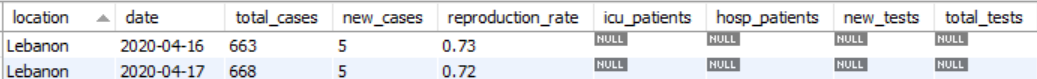
| # 1. Create locations table  CREATE TABLE locations AS SELECT  location, iso\_code, continent, population, population\_density, median\_age, aged\_65\_older, aged\_70\_older, gdp\_per\_capita, extreme\_poverty, cardiovasc\_death\_rate, diabetes\_prevalence, female\_smokers, male\_smokers, handwashing\_facilities, hospital\_beds\_per\_thousand, life\_expectancy, human\_development\_index  FROM covid19data\_cleaned  GROUP BY location;  # 2. Drop related data from covid19 table  ALTER TABLE covid19data\_cleaned  DROP iso\_code,  ...;  # 3.1. Rename country column to location  ALTER TABLE country\_vaccinations\_cleaned  RENAME COLUMN country TO location;  # 3.2. Join sources with location to form new table  CREATE TABLE location2 AS SELECT  locations.location, iso\_code, continent, population, population\_density  , median\_age, aged\_65\_older, aged\_70\_older  , gdp\_per\_capita, extreme\_poverty, cardiovasc\_death\_rate  , diabetes\_prevalence, female\_smokers, male\_smokers  , handwashing\_facilities, hospital\_beds\_per\_thousand, life\_expectancy  , human\_development\_index, source\_name, source\_website  FROM locations  LEFT JOIN  (  SELECT location, source\_name, source\_website FROM country\_vaccinations\_cleaned  GROUP BY location  ) t ON locations.location = t.location;  # 3.3. Override current location table with new table  DROP TABLE locations;  ALTER TABLE location2  RENAME TO location;  # 4. Create iso\_code table  CREATE TABLE iso\_code AS SELECT  location, iso\_code FROM location;  # 5. Drop iso\_code  ALTER TABLE location  DROP iso\_code;  ALTER TABLE country\_vaccinations\_cleaned  DROP iso\_code,  DROP source\_name,  DROP source\_website; |
| --- |

The full script can be found in database\_normalisation\_script\_sem6\_grp2.sql.

#### 2.3.2.3 Optimising the Database

Lastly, in order to optimise the database, we split *covid19\_data* and *country\_vaccinations* into various tables grouped by their attribute type, where we have separate tables for vaccines, tests, hospitals etc. This is not for normalisation, but simply to increase database and query efficiency by reducing the number of nulls in the rows. This is because data collection is typically done in batches, resulting in groups of attributes absent/present.





*Figure 7: the null distribution in covid19\_data, showing that the attributes can be grouped and split into separate tables for greater database efficiency.*

A snippet of the mySQL script used to perform the splitting is shown.

| CREATE TABLE tests AS SELECT  location, date,  # TESTS  tests\_units,  total\_tests,  new\_tests,  total\_tests\_per\_thousand,  new\_tests\_per\_thousand,  new\_tests\_smoothed,  new\_tests\_smoothed\_per\_thousand,  positive\_rate,  tests\_per\_case  FROM covid19data\_cleaned;  CREATE TABLE cases AS SELECT  ... |
| --- |

The full script can be found in database\_normalisation\_script\_sem6\_grp2.sql.

#### 2.3.2.4 Cleaning up the Database

With these split tables, we delete rows where all non-key attributes are missing, and set the relevant primary/foreign keys in each table.

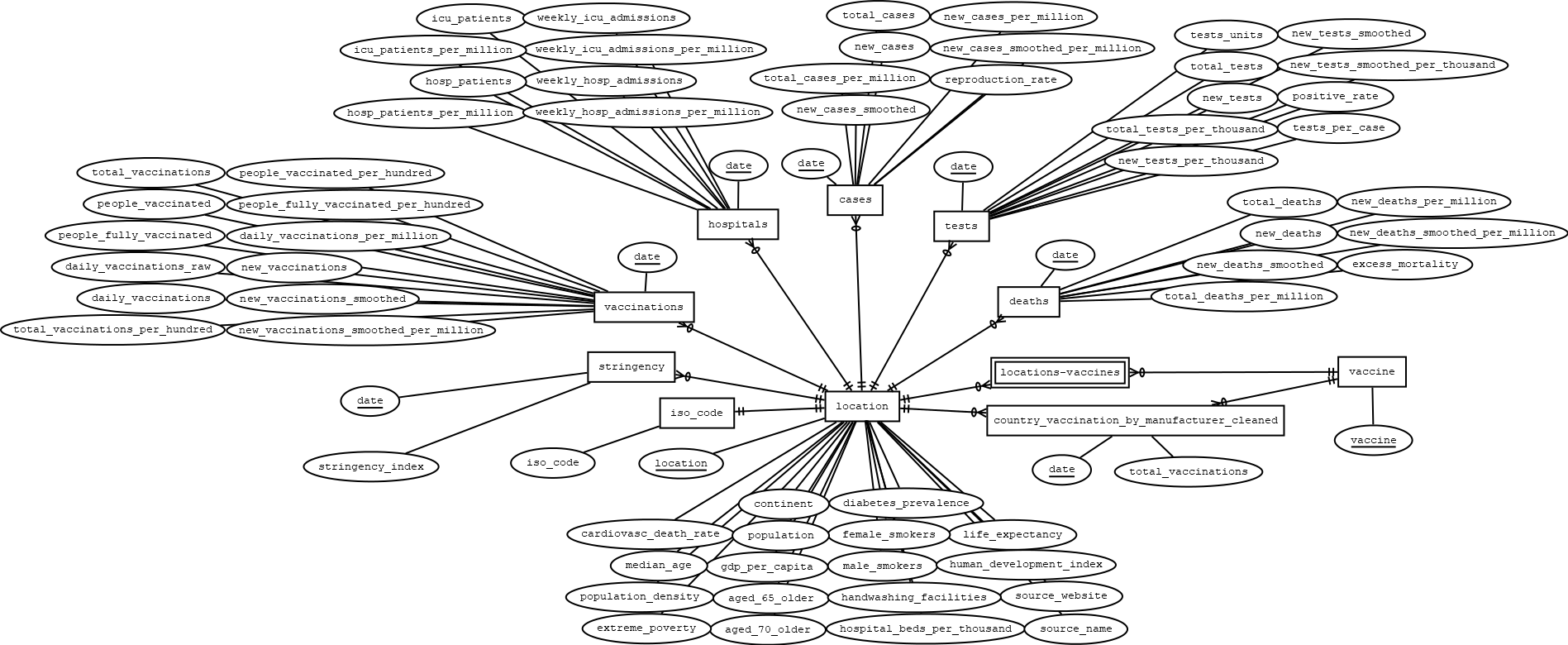
A snippet of the mySQL script used to perform the splitting is shown.

| # 1. Delete rows where all the non-primary keys are are null  DELETE FROM tests  WHERE total\_tests is null or 0  AND new\_tests is null or 0  AND total\_tests\_per\_thousand is null or 0  AND new\_tests\_per\_thousand is null or 0  AND new\_tests\_smoothed is null or 0  AND new\_tests\_smoothed\_per\_thousand is null or 0  AND positive\_rate is null or 0  AND tests\_per\_case is null or 0;  DELETE FROM cases  ...  DROP TABLE covid19data;  DROP TABLE covid19data\_cleaned;  DROP TABLE country\_vaccinations;  DROP TABLE country\_vaccinations\_cleaned;  DROP TABLE country\_vaccinations\_by\_manufacturer;  ALTER TABLE location  MODIFY COLUMN location VARCHAR(255),  ADD CONSTRAINT location\_pk PRIMARY KEY (location);  ...  ALTER TABLE country\_vaccinations\_by\_manufacturer\_cleaned  MODIFY COLUMN location VARCHAR(255),  MODIFY COLUMN vaccine VARCHAR(255),  ADD CONSTRAINT country\_vaccinations\_by\_manufacturer\_cleaned\_pk PRIMARY KEY (location, date, vaccine),  ADD CONSTRAINT country\_vaccinations\_by\_manufacturer\_fk1 FOREIGN KEY(location) REFERENCES location(location) ON DELETE RESTRICT ON UPDATE CASCADE,  ADD CONSTRAINT country\_vaccinations\_by\_manufacturer\_fk2 FOREIGN KEY(vaccine) REFERENCES vaccine(vaccine) ON DELETE RESTRICT ON UPDATE CASCADE; |
| --- |

The full script can be found in database\_normalisation\_script\_sem6\_grp2.sql.

## 2.4 Our Proposed Entity Relationship Diagram (“ERD”)

With the transformations applied in Section 2.3, here is our final ERD:



* Since an ER model has no foreign keys, they are not indicated in the ERD.
  + Hence, note that *location* in *location* is a foreign key in all other entities, and *vaccine* in *vaccine* is a foreign key in *locations-vaccines* and *country\_vaccinations\_by\_manufacturer\_cleaned*.
* We chose not to drop any attribute from the database entirely (only shifting them around the entities) since each attribute represents real data collected from real sources.
  + In addition, countries may use their own internal methods to calculate certain attributes differently from how we would have calculated them.
  + An example of this is as between *total\_cases* and *total\_cases\_per\_million*, where countries may have a more accurate *population* number than the one in our database when reporting for the latter. We should only perform our own calculations when the relevant data is missing entirely from the database, i.e. the *total\_cases\_per\_million* entry for that entity instance (record) is NULL.

## 2.5 Deploying our mySQL Database Implementation

| 1. | Download the file covid\_database\_sem6\_grp2.sql |  |
| --- | --- | --- |
| 2. | Open MySQLWorkbench |  |
| 3. | Connect to a MySQL database.  If no live database has been set-up set, connect to the local instance. This is a “database” that exists on your local computer. |  |
| 4. | On the toolbar at the top, select Server > Data Import |  |
| 5. | In the Data Import window, select ‘Import from Self-Contained File’ and navigate to the downloaded database file using the three dot  icon located at the end of the address bar. |  |
| 6. | Select the ‘Start Import’ button on the bottom right. |  |
| 7. | Once Status shows ‘1 of 1 imported’, press the small refresh icon on the right of the ‘SCHEMAS’ pane. The database should appear in this ‘SCHEMAS’ pane as covid\_database\_sem6\_grp2. |  |

# 3. Nonrelational Database Design

## 3.1 Introduction to Non-Relational Databases / mongoDB

Non-relational database, or NoSQL, is another type of database that has much more flexibility as compared to traditional relational databases. Advantages of NoSQL include high scalability and high performance in most types of operation. NoSQL databases use sharding[[3]](#footnote-2) to partition data on multiple machines. Thus, we can have the right data in the right place at the right time. More importantly, NoSQL has flexibility - as data grows, the database can expand to another device and shrink if things slow down, even an outage of one machine, the entire network can still function. The data in NoSQL database is simply a single file hence offering high availability.

MongoDB is an implementation of the NoSQL database. It uses JSON-like documents.

## 3.2 Principles in Non-Relational Database Design

NoSQL is fundamentally different from traditional Relational Database Management Systems (RDBMS). While RDMS imposes rigid schema based structures to their data model, NoSQL are based on non-relational data models, meaning they are flexible and developers are free to change the data model to suit their use case accordingly. New fields can be easily added, and these modifications can be easily applied to existing data sets, which makes it highly maintainable. Non-relational database design give up some features of the traditional databases for speed, but more importantly for horizontal scalability.[[4]](#footnote-3)

## 3.3 Challenges Presented by the Data and our Solutions

### 3.3.1 Choosing the COVID-19 Dataset

There were two covid19data sources provided, each having different structures.

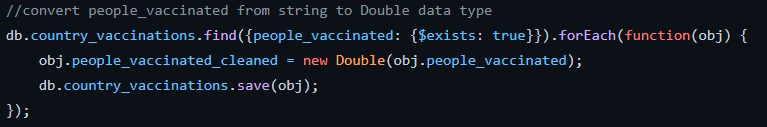
The first, covid19data.json, provided had nested arrays, which made it hard to import using MongoDB Compass and query, but the data are of correct data type. As for covid19data\_2.json, it is of a much simpler structure and does not require complex querying to use. However, the data types are all in String format, which means we have to do data cleaning before we can effectively use the data.

We chose the second covid19data\_2.json, since data cleaning is a one-time job and the simpler structure will make querying easier in the future. It was also easier to import in MongoDB Compass.

### 3.3.2 Data Cleaning

Following our decision in Section 3.3.1, we had to choose the most appropriate data type and modified them accordingly.

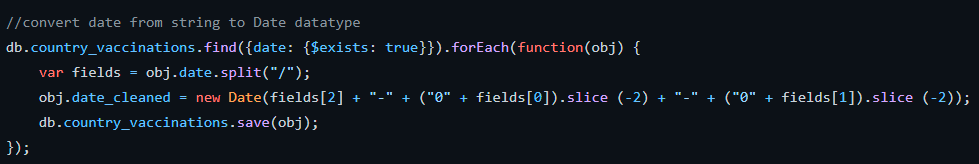
A snippet of the noSQL script used to perform the data cleaning is shown:



The full script can be found in noSQL\_database\_cleaning.js.

Furthermore, different databases report the data in different formats. For the country\_vaccinations database, date was reported in M/D/YYYY format as a string. Thus, when converting to Date we have to first convert the string YYYY-MM-DD format as YYYY-MM-DD is an ISO format so it will parse to UTC time but M/D/YYYY is in Non-ISO format, which will parse to time in local time zone. Since data entry occurs at different times, parsing in M/D/YYYY may result in the date being parsed to the incorrect day. [[5]](#footnote-4)

A snippet of the noSQL script used to perform this date parsing is shown:



The full script can be found in noSQL\_database\_cleaning.js.

Similar to the issue described in 2.3.2.1, we faced data inconsistencies.

For example, the timing cutoff for different dataset may be different. One dataset may report their numbers at noon everyday, while another dataset reports their numbers at midnight. This difference in timing may have resulted in different numbers being reported, even though both numbers are correct. We decide to standardise by using the number reported by country\_vaccinations.

### 3.3.3 Importing

One of the challenges we faced was that the data provided was large. The import with Mongo Compass itself will take longer, especially for our merged collections. We solved it by downloading MongoDB Command Line Database Tools, which includes a bunch of useful tools for MongoDB. One of them is mongoimport, which is a command line tool to import databases. Using mongoimport, we are able to specify the database and collection to import to, as well as the batch size. By changing the batch size to a lower value, we were able to import large JSON files without much issues.

### 3.3.4 Merging

We realised that most of our queries revolve around two collections, namely covid19data and country\_vaccinations\_by\_manufacturer. As such, we proposed to merge both collections into one consolidated collection for easier querying in the future. That posed a few problems.

Firstly, we had to find similar columns so that we have some basis to merge the collections. We carefully studied the data, and realised that both collections share 2 fields, namely dates and location. (This is similar to the SQL database, where the primary key was a composite of date and location). From there, we used aggregation operations to merge them together, and exported them into a new collection.

For the country\_vaccinations dataset, we chose not to merge this collection with the other two collections because the data in country\_vaccinations are aggregates while country\_vaccinations\_by\_manufacturers is to show data based on manufacturers specifically. Furthermore, if we were to merge this collection, the main collection size would be too huge and would require more advanced concepts like sharding in order for us to implement the database. Hence, we chose not to merge country\_vaccinations collection since it will not be required to perform queries with other collections.

### 3.3.5 Exporting

Another challenge we face is exporting the cleaned data. After cleaning and outputting the data using aggregate, we tried to export the data using NoSQLBooster. However, since it is the free version, we are unable to export due to its limitation. (exporting is a pay-only option) As such, we switched over to DataGrip[[6]](#footnote-5), which is a program developed by JetBrains. Since there is an educational license that is tied to an educational email address, we are able to create a JetBrains Account to make full use of DataGrip. From there, we are able to export the data to JSON after aggregate, and use the data for other queries.

### 3.3.6 Advantages of merging collections now press refresh on the content table on the top left

The main advantage visible to us after merging collections is the efficiency and speed of query. Although noSQL’s “Lookup” function is similar to SQL’s “JOIN” function, the Lookup function has a significantly worse performance time compared to JOIN. This means that if we were to perform multiple lookups, the query time would increase significantly. Furthermore, minor changes in aggregation requirements (such as changing what we want our query to derive based on the questions) would require major changes to the code. However, by merging collections together, the use of “Lookup” is kept minimal, allowing for quicker queries. In addition, the code is easier to change and can be used more flexibly to adjust our queries according to different business needs.

The second advantage of merging collections is the concept of extensibility. Data regarding COVID-19 changes almost everyday, therefore, we want our methodology of storing data to cater to such needs. With a single collection, even when data size increases or changes, it is easier to partition the data. We can then leverage on MongoDB’s auto-sharding feature to distribute data. Such data optimisation is key to many organisations.

Thirdly, merging collections saves data storage space. Having multiple collections in 1 database takes up much lesser storage compared to multiple databases having a single collection each. This is because storage is typically allocated at the database level.[[7]](#footnote-6) Thus we should aim to keep our data to a single database. Hence, there is a need to merge the collections within a single database to capitalise on the functionalities of MongoDB. As such, by merging collections, our database implementation can help organisations save storage space for other more important needs.[[8]](#footnote-7)

## 3.4 Deploying our mongoDB Database Implementation

| 1. | Create database with the following details:  Database Name: merged\_country\_vac\_manu\_covid19data  Collection Name: country\_vac\_with\_covid19data  in MongoDB Compass. |  |
| --- | --- | --- |
| 2. | Add a new collection country\_vaccinations\_cleaned |  |
| 3. | Import json file “country\_vac\_with\_covid19data.json”  to the collection: country\_vac\_withcovid19data |  |
| 4. | Import json file “country\_vaccinations\_cleaned.json”  to the collection country\_vaccinations\_cleaned |  |
| 5. | Verify that files are imported via noSQLbooster. |  |

# 4. Our Recommendations to WHO

We recommend WHO to utilise the NoSQL database model. Justification for this recommendation is detailed in Section 4.2; we will first discuss the difference between these two approaches in the next two Sections (4.1).

## 4.1 Comparisons/Inconsistencies between our Relational and Non-Relational Database Implementations

There are some main differences between relational and nonrelational database implementations. The table below explains some of the differences.

| Relational Database | Non-Relational Database |
| --- | --- |
| Rigid schemas   * Each column must be defined before data can be inserted. | Dynamic in nature   * New information can easily be added any time. * Some data can have more fields than others. |
| Stored in rows and columns | Stored in JSON format |
| Difficult to scale | Easy to scale (especially horizontal stability) |
| Suited for interconnected data | Suited for unstructured data |
| Requires normalisation to reduce duplication of rows | Collections are usually combined into one for easier query (no normalisation needed) |

## 4.2 Justification for our Recommendation

NoSQL is our database choice as a document oriented solution fits well to store COVID 19 data which is dynamic. COVID19 data is updated periodically to maintain the integrity of the data, hence the dynamic requirements make noSQL our recommendation. NoSQL fits the use case of COVID 19 data.

Firstly, the data provided contains a lot of null and empty values for the relational database, which may increase storage use and cause issues when querying the dataset. Non-relational databases do not have this issue, as the fields that are empty can just be omitted. Moreover, the data provided is very unstructured, and provides a lot of columns, some of which may seem redundant. Non-relational databases generally handle this sort of data much better as compared to relational databases.

Next, relational databases are rigid in nature, and each column must be defined before data can be inserted. This means that whenever we have new rows that have new columns that have to be added, we have to modify the entire database to accommodate that. On the other hand, non-relational databases are dynamic in nature, which allows for more flexibility. For COVID-19 data, because it is new, it is constantly evolving, and there is likely more data and fields to be added in the future. Therefore, it makes more sense to use a dynamic database, allowing for easier modifications in the future.

Thirdly, the main advantage of NoSQL is its ability to deal with enormous amounts of data. COVID 19 data is growing in size and fields, hence it would be better for the WHO to adopt a non relational database design which supports horizontal scalability. This way, information can be easily updated periodically to maintain data integrity. Even if data size becomes enormous, storage can still be optimised through sharding.

Fourthly, noSQL is designed to be document oriented. Therefore, by merging the collections based on our implementation described earlier, the needs for joins and lookups are reduced, making queries faster. This addresses one of the main concerns of noSQL which is the slow runtime of lookups. All the data is easily accessible within a single collection. By using a non-relational database design, the WHO can benefit through faster access to the data where they need it and when they need it. In addition, noSQL is also more user-friendly for the staff, as it is easier to alter queries in NoSQL to derive information needed for different business problems.

Lastly, since there are only two collections in the non-relational database as compared to multiple tables in the relational database, it is much easier for users to query. There is no need to JOIN between several tables and figure out the columns to join on. This means that it will be faster to retrieve all of the data together as they are stored in the same place.

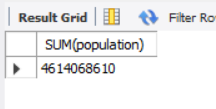
In conclusion, we recommend NoSQL as the database model to be used for storing COVID-19 data.

~ ⦁ ~

**Appendix A: SQL queries**

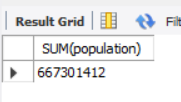
1. What is the total population in Asia?

| # The normalised "locations" table contains the population count for each country. # We group the table by continents, then filter the table to only rows where the continent is 'Asia', # This returns a table of all the Asian countries. # Then we apply an aggregate SUM() function on the population column to obtain a single value response.  **SELECT** **SUM(**population**)**  **FROM** location  **GROUP** **BY** continent  **HAVING** continent **=** 'Asia'**;** |
| --- |

(1 row returned)

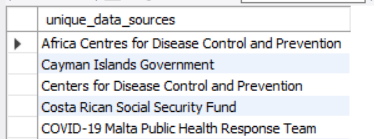
2. What is the total population among the ten ASEAN countries?

| # The normalised "locations" table contains the population count for each country.  # We then sum the populations of rows that are in ASEAN.  **SELECT** **SUM(**population**)**  **FROM** location  **WHERE** location **IN** **(**'Brunei'**,** 'Cambodia'**,** 'Indonesia'**,** 'Laos'**,** 'Malaysia'**,** 'Myanmar'**,** 'Philippines'**,** 'Singapore'**,** 'Thailand'**,** 'Vietnam'**);** |
| --- |

(1 row returned)

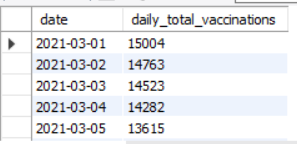
3. Generate a list of unique data sources (source\_name).

| # Using the source\_name column as instructed in the question, we obtain 91 rows.  # However, this may not be correct method if we want to generate the list of UNIQUE data sources,  # because multiple countries list "Ministry of Health" as their source\_name,  # but these Ministry of Health(s) are for their respective countries and should hence be considered different data sources.  # Examples of countries with MOH: Argentina, Austria, Uganda, Zimbabwe, ...  # By selecting source\_website as opposed to source\_name, i.e.  ## SELECT DISTINCT(source\_name) as unique\_data\_sources  ## FROM sources  ## ORDER BY unique\_data\_sources;  # We get 147 rows instead of 91.  **SELECT** **DISTINCT(**source\_name**)** **as** unique\_data\_sources  **FROM** locations  **WHERE** source\_name **IS** **NOT** **NULL**  #**select** **is** **not** **null** since sources **is** **left-**joined **with** locations  **ORDER** **BY** unique\_data\_sources**;** |
| --- |

(92 rows returned)

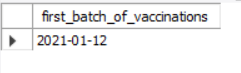
4. Specific to Singapore, display the daily total\_vaccinations starting (inclusive) March-1 2021 through (inclusive) May-31 2021.

| # We used daily vaccinations instead of raw daily vaccinations as it is cleaned and validated, so it is likely to be more accurate.  # From now onwards, we will use daily\_vaccinations.  **SELECT** **date,** daily\_vaccinations **AS** daily\_total\_vaccinations  **FROM** vaccinations  **WHERE** location **=** 'Singapore' **AND** **date** **BETWEEN** '2021-03-01' **AND** '2021-05-31'  **ORDER** **BY** **date;** |
| --- |

 (92 rows returned)

5. When is the first batch of vaccinations recorded in Singapore?

| # total\_vaccinations: the total number of COVID-19 vaccination doses administered  # daily\_vaccinations: for a certain data entry, the number of vaccination for that date/country  # 1/11/2021 has 3400 total\_vaccinations, but 0 under daily\_vaccinations  # 1/12/2021 has 6200 total\_vaccinations, but 2800 under daily\_vaccinations  # It is reasonable to assume that total\_vaccinations for today is calculated by taking the  # total\_vaccinations of yesterday plus the daily\_vaccinations of today, i.e.  # total\_vaccinations\_today = total\_vaccinations\_yesterday + daily\_vaccinations\_today.  # Therefore, the numbers for 1/12/2021 makes sense, since  # 6200 = 3400 + 2800  # We cannot do the same confirmation for 1/11/2021, since there is missing data for 1/10/2021 and earlier  # Hence, data on 1/12/2021 is more reliable.  **SELECT** **MIN(date)** **as** first\_batch\_of\_vaccinations  **FROM** vaccinations  **WHERE** location **=** 'Singapore'  **AND** daily\_vaccinations **>** 0**;** |
| --- |

(1 row returned)

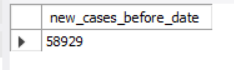
6. Based on the date identified in (5), specific to Singapore, compute the total number of new cases thereafter. For instance, if the date identified in (5) is Jan-1 2021, the total number of new cases will be the sum of new cases starting from (inclusive) Jan-1 to the last date in the dataset.

| # new\_cases: new confirmed cases of COVID-19  # Since we want the latest date in the dataset, we only need to set the lower bound of date and not upper bound  # Note: this lower bound is inclusive.  **SELECT** **SUM(**new\_cases**)** **as** total\_cases\_thereafter  **FROM** cases  **WHERE** location **=** 'Singapore'  **AND** **date** **>=**  **(**  **SELECT** **MIN(date)** **as** **date**  **FROM** vaccinations  **WHERE** location **=** 'Singapore'  **AND** daily\_vaccinations **>** 0  **);** |
| --- |

(1 row returned)

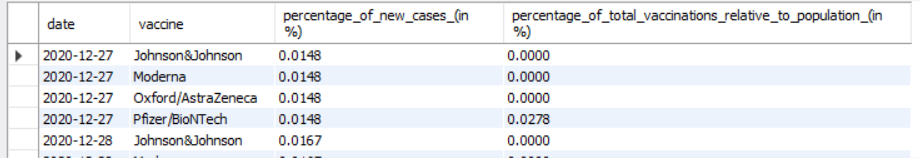
7. Compute the total number of new cases in Singapore before the date identified in (5). For instance, if the date identified in (5) is Jan-1 2021 and the first date recorded (in Singapore) in the dataset is Feb-1 2020, the total number of new cases will be the sum of new cases starting from (inclusive) Feb-1 2020 through (inclusive) Dec-31 2020.

| # Since we want the earliest date in the dataset, we only need to set the upper bound of date and not upper bound.  # Note: this upper bound is NOT inclusive.  **SELECT** **SUM(**new\_cases**)** **AS** new\_cases\_before\_date  **FROM** cases  **WHERE** location **=** 'Singapore'  **AND** **DATE** **<**  **(**  **SELECT** **MIN(date)** **as** **date**  **FROM** vaccinations  **WHERE** location **=** 'Singapore'  **AND** daily\_vaccinations **>** 0  **);** |
| --- |

(1 row returned)

8. Herd immunity estimation. On a daily basis, specific to Germany, calculate the percentage of new cases (i.e., percentage of new cases = new cases / populations) and total vaccinations on each available vaccine in relation to its population.

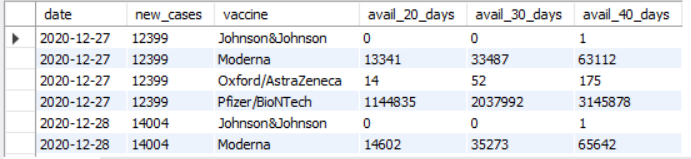
| # Join our cases and country\_vaccinations\_by\_manufacturer table with the location table on the composite primary key of country & time.  # From here, we calculate the percentage of new cases and vaccinations over total population via the formula below,  # and group by date and vaccine to show "daily basis" and "each available vaccine" respectively.  #we calculate total vaccinations on each available vaccine in relation to population  as a % as well.  **SELECT** **date,** vaccine**,** new\_cases**/**population **\*** 100 **as** 'percentage\_of\_new\_cases\_(in %)'**,** total\_vaccinations**/**population **\*** 100 **as** 'percentage\_of\_total\_vaccinations\_relative\_to\_population\_(in %)'  **FROM**  cases **NATURAL** **JOIN** country\_vaccinations\_by\_manufacturer\_cleaned **NATURAL** **JOIN** location  **WHERE** location **=** 'Germany'  **GROUP** **BY** **date,** vaccine**;** |
| --- |



(744 rows returned)

9. Vaccination Drivers. Specific to Germany, based on each daily new case, display the total vaccinations of each available vaccines after 20 days, 30 days, and 40 days.

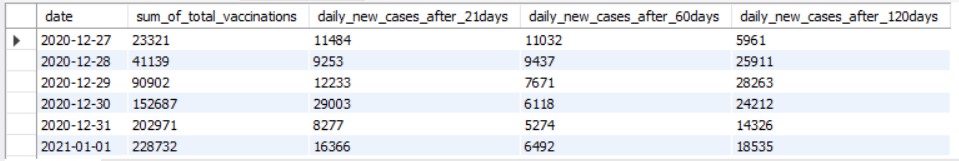
| **SELECT** **DISTINCT** c**.date,** c**.**new\_cases**,** c**.**vaccine**,**  d20**.**D20\_avail\_vaccine **AS** avail\_20\_days**,**  d30**.**D30\_avail\_vaccine **AS** avail\_30\_days**,**  d40**.**D40\_avail\_vaccine **AS** avail\_40\_days  **FROM**  **(** # Create three columns each containing the same data of 1 date (20, 30 and 40 days later respectively)  **SELECT** **DISTINCT** cd**.date,** cd**.**new\_cases**,** cm**.**vaccine**,** cm**.**total\_vaccinations**,**  **date\_add(**cd**.date,** **interval** 20 **DAY)** **AS** DAY20**,**  **date\_add(**cd**.date,** **interval** 30 **DAY)** **AS** DAY30**,**  **date\_add(**cd**.date,** **interval** 40 **DAY)** **AS** DAY40  **FROM** cases cd  **JOIN** country\_vaccinations\_by\_manufacturer\_cleaned cm **on** cm**.date** **=** cd**.date** **AND** cm**.**location **=** cd**.**location  **WHERE** cd**.**location **=** 'Germany'  **)** c  # Each of these left joins "duplicates" and "shifts" the vaccinations\_by\_manufacturer table down.  # We do this three times in total to get the total\_vaccinations for 20, 30  # and 40 days later respectively.  **LEFT** **JOIN(SELECT** **date,** vaccine**,** total\_vaccinations **AS** D20\_avail\_vaccine  **FROM** country\_vaccinations\_by\_manufacturer\_cleaned  **WHERE** location **=** 'Germany'**)** d20 **ON** d20**.date** **=** c**.**DAY20 **AND** d20**.**vaccine **=** c**.**vaccine  **LEFT** **JOIN(SELECT** **date,** vaccine**,** total\_vaccinations **AS** D30\_avail\_vaccine  **FROM** country\_vaccinations\_by\_manufacturer\_cleaned  **WHERE** location **=** 'Germany'**)** d30 **ON** d30**.date** **=** c**.**DAY30 **AND** d30**.**vaccine **=** c**.**vaccine  **LEFT** **JOIN(SELECT** **date,** vaccine**,** total\_vaccinations **AS** D40\_avail\_vaccine  **FROM** country\_vaccinations\_by\_manufacturer\_cleaned  **WHERE** location **=** 'Germany'**)** d40 **ON** d40**.date** **=** c**.**DAY40 **AND** d40**.**vaccine **=** c**.**vaccine**;** |
| --- |



(744 rows returned)

10. Vaccination Effects. Specific to Germany, on a daily basis, based on the total number of accumulated vaccinations (sum of total\_vaccinations of each vaccine in a day), generate the daily new cases after 21 days, 60 days, and 120 days

| # We adopt the same approach as Q9, but implement it using views instead.  # Views are an easier way to visualise the querying process. In mySQL's default settings and  # without an ALGORITHM specified, the MERGE setting is automatically used where the statement  # retrieves parts of the view definition to replace corresponding parts of the statement in a view resolution.[[9]](#footnote-8)[[10]](#footnote-9)  # This means that no extra space is required to store the view.  # View for sum of total\_vaccinations across dates for Germany  #Here we accumulated the sum of vaccine instead of displaying them individually  **CREATE** **VIEW** total **AS**  **SELECT**  **date** **,** **SUM(**total\_vaccinations**)** **AS** total  **FROM** country\_vaccinations\_by\_manufacturer\_cleaned cm  **WHERE** location **=** "Germany"  **GROUP** **BY** **date**  **ORDER** **BY** **date;**  # Append three columns each containing the same data of 1 date (20, 30 and 40 days later respectively)  **CREATE** **VIEW** total\_with\_lagged\_dates **AS**  **SELECT** **\*,**  **date\_add(**total**.date,** **INTERVAL** 21 **day)** **AS** d21**,**  **date\_add(**total**.date,** **INTERVAL** 60 **day)** **AS** d60**,**  **date\_add(**total**.date,** **INTERVAL** 120 **day)** **AS** d120  **FROM** total**;**  # Create three different views for new\_cases across dates  **CREATE** **VIEW** day21\_cases **AS**  **SELECT** **date,** new\_cases **AS** day21  **FROM** cases  **WHERE** location **=** "Germany"**;**  **CREATE** **VIEW** day60\_cases **as**  **SELECT** **date,** new\_cases **as** day60  **FROM** cases  **WHERE** location **=** "Germany"**;**  **CREATE** **VIEW** day120\_cases **as**  **SELECT** **date,** new\_cases **AS** day120  **FROM** cases  **WHERE** location **=** "Germany"**;**  # Join total\_with\_lagged\_dates with the 3 different views on the lagged dates.  **SELECT**  total\_with\_lagged\_dates**.date,** total **AS** sum\_of\_total\_vaccinations**,**  day21 **AS** daily\_new\_cases\_after\_21days**,** day60 **AS** daily\_new\_cases\_after\_60days**,** day120 **AS** daily\_new\_cases\_after\_120days  **FROM** total\_with\_lagged\_dates  **LEFT** **JOIN** day21\_cases **ON** total\_with\_lagged\_dates**.**d21 **=** day21\_cases**.date**  **LEFT** **JOIN** day60\_cases **ON** total\_with\_lagged\_dates**.**d60 **=** day60\_cases**.date**  **LEFT** **JOIN** day120\_cases **ON** total\_with\_lagged\_dates**.**d120 **=** day120\_cases**.date;**  **DROP** **VIEW** day120\_cases**,** day21\_cases**,** day60\_cases**,** total**,** total\_with\_lagged\_dates |
| --- |

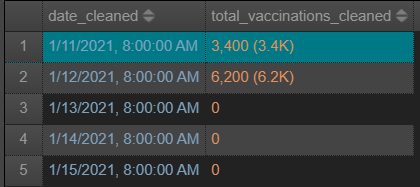


(186 rows returned)

# Appendix B – noSQL queries

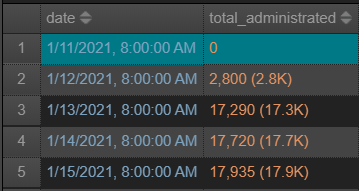
1. Display a list of total vaccinations per day in Singapore. [source table: country\_vaccinations]

| db**.**country\_vaccinations\_cleaned**.**find**(**  **{**country**:** "Singapore"**},**  **{**date\_cleaned**:** 1**,** \_id**:** 0**,** total\_vaccinations\_cleaned**:** 1**}**  **).**sort**({**date\_cleaned**:** 1**})** |
| --- |

(169 documents)

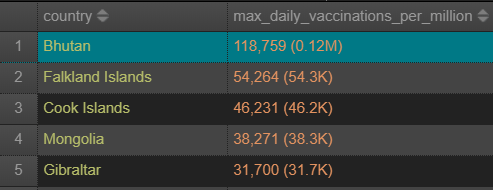
2. Display the sum of daily vaccinations among ASEAN countries. [source table: country\_vaccinations]

| db**.**country\_vaccinations\_cleaned**.**aggregate**(**  **{**$match**:** **{**country**:** **{**$in**:** **[**"Brunei"**,** "Myanmar"**,** "Cambodia"**,** "Indonesia"**,** "Laos"**,** "Malaysia"**,** "Philippines"**,** "Singapore"**,** "Thailand"**,** "Vietnam"**]}}},**  **{**$group**:** **{**\_id**:** "$date\_cleaned"**,** total\_administrated**:** **{**$sum**:** **{**$round**:** **[**"$daily\_vaccinations\_cleaned"**,** 0**]}}}},**  **{**$sort**:** **{**"\_id"**:** 1**}},**  **{**$project**:** **{**\_id**:** 0**,** "date"**:** "$\_id"**,** "total\_administrated"**:** "$total\_administrated"**}}**  **)** |
| --- |

(171 documents)

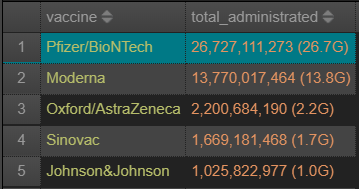
3. Identify the maximum daily vaccinations per million of each country. Sort the list based on daily vaccinations per million in a descending order. [source table: country\_vaccinations]

| db**.**country\_vaccinations\_cleaned**.**aggregate**(**  **{**$group**:** **{**\_id**:** "$country"**,** max\_daily\_vaccinations\_per\_million**:** **{**$max**:** "$daily\_vaccinations\_per\_million\_cleaned"**}}},**  **{**$sort**:** **{**"max\_daily\_vaccinations\_per\_million"**:** **-**1**}},**  **{**$project**:** **{**"country"**:** 1**,** "max\_daily\_vaccinations\_per\_million"**:** 1**}},**  **{**$project**:** **{**\_id**:** 0**,** "country"**:** "$\_id"**,** "max\_daily\_vaccinations\_per\_million"**:** "$max\_daily\_vaccinations\_per\_million"**}}**  **)** |
| --- |

(217 documents)

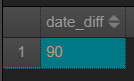
4. Which is the most administrated vaccine? Display a list of total administration (i.e., sum of total vaccinations) per vaccine. [source table: country\_vaccinations\_by\_manufacturer]

| db**.**country\_vac\_with\_covid19data**.**aggregate**(**  **{**$match**:** **{**vaccinations\_by\_manufacturer\_data**:** **{**$exists**:** **true,** $ne**:[]}}},**  **{**$unwind**:** "$vaccinations\_by\_manufacturer\_data"**},**  **{**$group**:** **{**\_id**:** "$vaccinations\_by\_manufacturer\_data.vaccine"**,** total\_administrated**:** **{**$sum**:** "$vaccinations\_by\_manufacturer\_data.total\_vaccinations\_cleaned"**}}},**  **{**$sort**:** **{**"total\_administrated"**:** **-**1**}},**  **{**$project**:** **{**\_id**:** 0**,**"vaccine"**:** "$\_id"**,**"total\_administrated"**:** "$total\_administrated"**}}**  **)** |
| --- |

(8 documents)

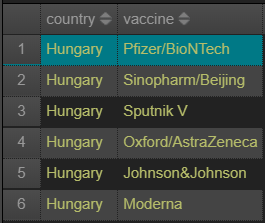
5. Italy has commenced administrating various vaccines to its populations as a vaccine becomes available. Identify the first dates of each vaccine being administrated, then compute the difference in days between the earliest date and the 4th date. [source table: country\_vaccinations\_by\_manufacturer]

| // First dates of each vaccine being administrated  db**.**country\_vac\_with\_covid19data**.**aggregate**(**  **{**$match**:** **{location:** "Italy"**}},**  **{**$match**:** **{**vaccinations\_by\_manufacturer\_data**:** **{**$exists**:** **true,** $ne**:** **[]}}},**  **{**$unwind**:** "$vaccinations\_by\_manufacturer\_data"**},**  **{**$group**:** **{**\_id**:** "$vaccinations\_by\_manufacturer\_data.vaccine"**,** date**:** **{**$min**:** "$date\_cleaned"**}}},**  **{**$sort**:** **{**"date"**:** 1**}},**  **{**$project**:** **{**\_id**:** 0**,** "vaccine"**:** "$\_id"**,** "date"**:** "$date"**}}**  **)**  // Difference in days between earliest date and 4th date  db**.**country\_vac\_with\_covid19data**.**aggregate**(**  **{**$match**:** **{location:** "Italy"**}},**  **{**$match**:** **{**vaccinations\_by\_manufacturer\_data**:** **{**$exists**:** **true,** $ne**:** **[]}}},**  **{**$unwind**:** "$vaccinations\_by\_manufacturer\_data"**},**  **{**$group**:** **{**\_id**:** "$vaccinations\_by\_manufacturer\_data.vaccine"**,** date**:** **{**$min**:** "$date\_cleaned"**}}},**  **{**$group**:** **{**\_id**:** **null,** date**:** **{**$addToSet**:** "$date"**}}},**  **{**$project**:** **{**date\_diff**:** **{**$dateDiff**:** **{**startDate**:** **{**$min**:** "$date"**},** endDate**:** **{**$max**:** "$date"**},** unit**:** "day"**}}}},**  **{**$project**:** **{**\_id**:** 0**,** "date\_diff"**:** "$date\_diff"**}}**  **)** |
| --- |

(4 documents)

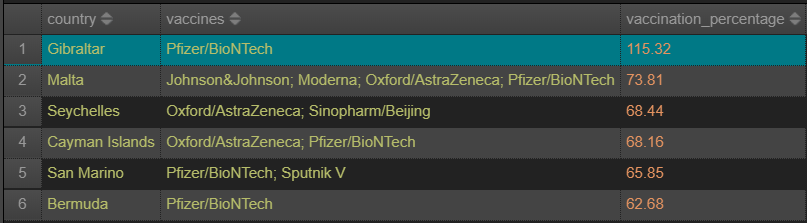
6. What is the country with the most types of administrated vaccine? [source table: country\_vaccinations\_by\_manufacturer]

| db**.**country\_vac\_with\_covid19data**.**aggregate**([**  **{**$group**:** **{**\_id**:** **{**location**:** "$location"**,** vaccine**:** "$vaccinations\_by\_manufacturer\_data.vaccine"**}}},**  **{**$project**:** **{**\_id**:** 0**,** location**:** "$\_id.location"**,** all\_vaccine**:**"$\_id.vaccine"**,** count**:** **{**$size**:** "$\_id.vaccine"**}}},**  **{**$sort**:** **{**count**:** **-**1**}},**  **{**$limit**:** 1**},**  **{**$unwind**:** "$all\_vaccine"**},**  **{**$project**:** **{**\_id**:** 0**,**"country"**:** "$location"**,** "vaccine"**:** "$all\_vaccine"**}}**  **])** |
| --- |

(6 documents)

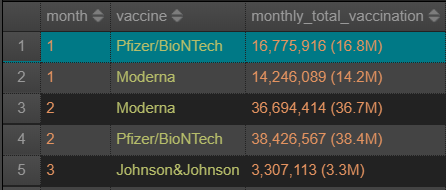
7. What are the countries that have fully vaccinated more than 60% of its people? For each country, display the vaccines administrated. [source table: country\_vaccinations]

| db**.**country\_vaccinations\_cleaned**.**aggregate**([**  **{**$group**:** **{**\_id**:** **{**country**:** "$country"**},** vaccines**:** **{**$max**:** "$vaccines"**},** vaccination\_percentage**:** **{**$max**:** "$people\_fully\_vaccinated\_per\_hundred\_cleaned"**}}},**  **{**$match**:** **{**"vaccination\_percentage"**:** **{**$gt**:** 60**}}},**  **{**$project**:** **{**\_id**:** 0**,** "country"**:**"$\_id.country"**,** "vaccines"**:** "$vaccines"**,** "vaccination\_percentage"**:** "$vaccination\_percentage"**}},**  **{**$sort**:** **{**vaccination\_percentage**:** **-**1**}}**  **])** |
| --- |

(6 documents)

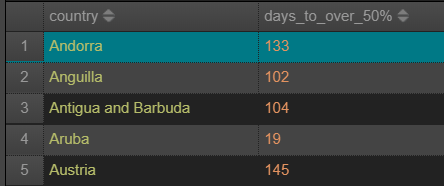
8. Monthly vaccination insight – display the monthly total vaccination amount of each vaccine per month in the United States. [source table: country\_vaccinations\_by\_manufacturer]

| db**.**country\_vac\_with\_covid19data**.**aggregate**([**  **{**$match**:** **{location:** "United States"**}},**  **{**$match**:** **{**"vaccinations\_by\_manufacturer\_data"**:** **{**$exists**:** **true,** $ne**:[]}}},**  **{**$unwind**:** "$vaccinations\_by\_manufacturer\_data"**},**  **{**$project**:** **{**\_id**:** 0**,** month**:** **{**$month**:** "$date\_cleaned"**},** "vaccine"**:** "$vaccinations\_by\_manufacturer\_data.vaccine"**,** "total\_vaccinations\_cleaned"**:** "$vaccinations\_by\_manufacturer\_data.total\_vaccinations\_cleaned"**}},**  **{**$group**:** **{**\_id**:** **{**month**:** "$month"**,** vaccine**:** "$vaccine"**},** monthly\_total\_vaccination**:** **{**$max**:** "$total\_vaccinations\_cleaned"**}}},**  **{**$project**:** **{**\_id**:** 0**,**"month"**:** "$\_id.month"**,** "vaccine"**:** "$\_id.vaccine"**,** "monthly\_total\_vaccination"**:** "$monthly\_total\_vaccination"**}},**  **{**$sort**:** **{**month**:** 1**}}**  **])** |
| --- |

(16 documents)

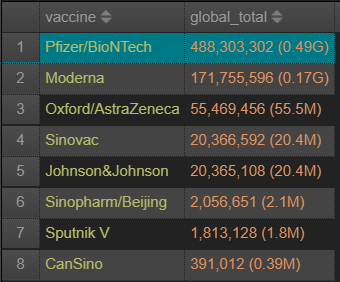
9. Days to 50 percent. Compute the number of days (i.e., using the first available date on records of a country) that each country takes to go above the 50% threshold of vaccination administration (i.e., total\_vaccinations\_per\_hundred > 50) [source table: country\_vaccinations]

| db**.**country\_vaccinations\_cleaned**.**aggregate**([**  **{**$project**:** **{**country**:** 1**,** date\_cleaned**:** 1**,** total\_vaccinations\_per\_hundred\_cleaned**:** 1**}},**  **{**$group**:** **{**\_id**:** "$country"**,** minDate**:** **{**$min**:** "$date\_cleaned"**},** date50**:** **{**$min**:** **{**$cond**:** **[{**$gt**:** **[**"$total\_vaccinations\_per\_hundred\_cleaned"**,** 50**]},** "$date\_cleaned"**,** **null]}}}},**  **{**$match**:** **{**$and**:** **[{**date50**:** **{**$ne**:** **null}},** **{**date50**:** **{**$exists**:** **true}}]}},**  **{**$project**:** **{**date\_diff**:** **{**  $dateDiff**:** **{**  startDate**:** "$minDate"**,**  endDate**:** "$date50"**,**  unit**:** "day"  **}**  **}}},**  **{**$project**:** **{**\_id**:** 0**,** "country"**:** "$\_id"**,** "days\_to\_over\_50%"**:** "$date\_diff"**}},**  **{**$sort**:** **{**"country"**:** 1**}}**  **])** |
| --- |

(85 documents)

10. Compute the global total of vaccinations per vaccine. [source table: country\_vaccinations\_by\_manufacturer]

| db**.**country\_vac\_with\_covid19data**.**aggregate**([**  **{**$match**:** **{**"vaccinations\_by\_manufacturer\_data"**:** **{**$exists**:** **true,** $ne**:** **[]}}},**  **{**$unwind**:** "$vaccinations\_by\_manufacturer\_data"**},**  **{**$group**:** **{**\_id**:** **{**location**:** "$location"**,** vaccine**:** "$vaccinations\_by\_manufacturer\_data.vaccine"**},** total\_vaccination**:** **{**$max**:** "$vaccinations\_by\_manufacturer\_data.total\_vaccinations\_cleaned"**}}},**  **{**$group**:** **{**\_id**:** **{**vaccine**:** "$\_id.vaccine"**},** global\_total**:** **{**$sum**:** "$total\_vaccination"**}}},**  **{**$project**:** **{**\_id**:** 0**,** vaccine**:** "$\_id.vaccine"**,** global\_total**:** "$global\_total"**}},**  **{**$sort**:** **{**global\_total**:** **-**1**}}**  **])** |
| --- |

(8 documents)

11. What is the total population in Asia?

| db**.**country\_vac\_with\_covid19data**.**aggregate**([**  **{**$match**:** **{**continent**:** "Asia"**}},**  **{**$group**:** **{**\_id**:** **{**country**:** "$location"**},** population**:** **{**$avg**:** "$population\_cleaned"**}}},**  **{**$group**:** **{**\_id**:** **null,** total\_population**:** **{**$sum**:** "$population"**}}},**  **{**$project**:** **{**\_id**:** 0**,** "total\_population"**:** "$total\_population"**}}**  **])** |
| --- |

(1 document)

12. What is the total population among the ten ASEAN countries?

| db**.**country\_vac\_with\_covid19data**.**aggregate**([**  **{**$match**:** **{**location**:** **{**$in**:** **[**"Brunei"**,** "Cambodia"**,** "Indonesia"**,** "Laos"**,** "Malaysia"**,** "Myanmar"**,** "Philippines"**,** "Singapore"**,** "Thailand"**,** "Vietnam"**]}}},**  **{**$group**:** **{**\_id**:** **{**location**:** "$location"**},** population**:** **{**$max**:** "$population\_cleaned"**}}},**  **{**$group**:** **{**\_id**:** **null,** total\_population**:** **{**$sum**:** "$population"**}}},**  **{**$project**:** **{**\_id**:** 0**,** "total\_population"**:** "$total\_population"**}}**  **])** |
| --- |

(1 document)

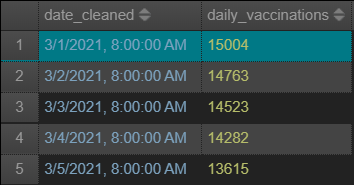
13. Generate a list of unique data sources (source\_name).

| db**.**country\_vaccinations\_cleaned**.**aggregate**([**  **{**$group**:** **{**\_id**:null,** unique\_data\_sources**:** **{**$addToSet**:** "$source\_name"**}}},**  **{**$project**:** **{**\_id**:** 0**,** "unique\_data\_sources"**:** "$unique\_data\_sources"**}}**  **])** |
| --- |

# 

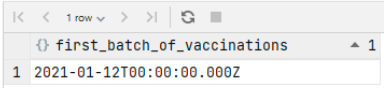
14. Specific to Singapore, display the daily total\_vaccinations starting (inclusive) March-1 2021 through (inclusive) May-31 2021. come to that section

| db**.**country\_vaccinations\_cleaned**.**aggregate**([**  **{**$match**:** **{** $and**:** **[** **{**country**:** "Singapore"**},** **{**date\_cleaned**:** **{**$gte**:** ISODate**(**"2021-03-01"**),** $lt**:** ISODate**(**"2021-04-01"**)}}]}},**  **{**$project**:** **{**\_id**:**0**,** date\_cleaned**:** 1**,** "daily\_vaccinations"**:** "$daily\_vaccinations"**}},**  **{**$sort**:** **{**"date\_cleaned"**:** 1**}}**  **])** |
| --- |

(92 documents)

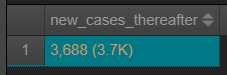
15. When is the first batch of vaccinations recorded in Singapore?

| db**.**country\_vaccinations\_cleaned**.**aggregate**([**  **{**$match**:** **{**$and**:** **[{**country**:** "Singapore"**},** **{**daily\_vaccinations\_cleaned**:** **{**$gt**:** 0**}}]}},**  **{**$project**:** **{**date\_cleaned**:** 1**}},**  **{**$limit**:** 1**},**  **{**$project**:** **{**\_id**:** 0**,** "first\_batch\_of\_vaccinations"**:**"$date\_cleaned"**}}**  **])** |
| --- |

(1 document)

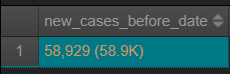
16. Based on the date identified in (5), specific to Singapore, compute the total number of new cases thereafter. For instance, if the date identified in (5) is Jan-1 2021, the total number of new cases will be the sum of new cases starting from (inclusive) Jan-1 to the last date in the dataset.

| db**.**country\_vac\_with\_covid19data**.**aggregate**([**  **{**$match**:** **{**location**:** "Singapore"**}},**  **{**$match**:** **{**"date\_cleaned"**:** **{**$gte**:** ISODate**(**"2021-01-12"**)}}},**  **{**$group**:** **{**\_id**:** **null,** new\_cases\_thereafter**:** **{**$sum**:** "$new\_cases\_cleaned"**}}},**  **{**$project**:** **{**\_id**:** 0**,** "new\_cases\_thereafter"**:** "$new\_cases\_thereafter"**}}**  **])** |
| --- |

(1 document)

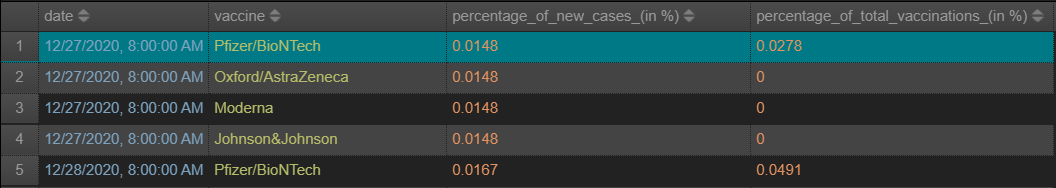
17. Compute the total number of new cases in Singapore before the date identified in (5). For instance, if the date identified in (5) is Jan-1 2021 and the first date recorded (in Singapore) in the dataset is Feb-1 2020, the total number of new cases will be the sum of new cases starting from (inclusive) Feb-1 2020 through (inclusive) Dec-31 2020.

| db**.**country\_vac\_with\_covid19data**.**aggregate**([**  **{**$match**:** **{**location**:** "Singapore"**}},**  **{**$match**:** **{**"date\_cleaned"**:** **{**$lt**:** ISODate**(**"2021-01-12"**)}}},**  **{**$group**:** **{**\_id**:** **null,** new\_cases\_before\_date**:** **{**$sum**:** "$new\_cases\_cleaned"**}}},**  **{**$project**:** **{**\_id**:** 0**,** "new\_cases\_before\_date"**:** "$new\_cases\_before\_date"**}}**  **])** |
| --- |

(1 document)

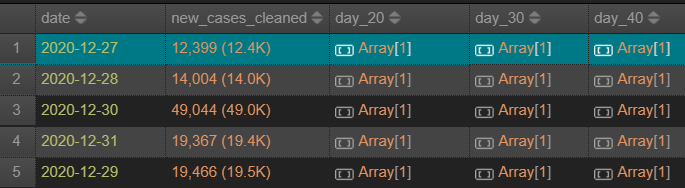
18. Herd immunity estimation. On a daily basis, specific to Germany, calculate the percentage of new cases and total vaccinations on each available vaccine in relation to its population.

| db**.**country\_vac\_with\_covid19data**.**aggregate**([**  **{**$match**:** **{**location**:** "Germany"**}},**  **{**$match**:** **{**vaccinations\_by\_manufacturer\_data**:** **{**$exists**:** **true,** $ne**:** **[]}}},**  **{**$unwind**:** "$vaccinations\_by\_manufacturer\_data"**},**  **{**$project**:** **{**date**:** "$date\_cleaned"**,** "percentage\_of\_new\_cases\_(in %)"**:** **{**$multiply**:** **[{**$divide**:** **[**"$new\_cases\_cleaned"**,** "$population\_cleaned"**]},** 100**]},** vaccine**:** "$vaccinations\_by\_manufacturer\_data.vaccine"**,** "percentage\_of\_total\_vaccinations\_(in %)"**:** **{**$multiply**:** **[{**$divide**:** **[**"$vaccinations\_by\_manufacturer\_data.total\_vaccinations\_cleaned"**,** "$population\_cleaned"**]},** 100**]}}},**  **{**$project**:** **{**\_id**:** 0**,** date**:** 1**,** vaccine**:** "$vaccine"**,** "percentage\_of\_new\_cases\_(in %)"**:** "$percentage\_of\_new\_cases\_(in %)"**,** "percentage\_of\_total\_vaccinations\_(in %)"**:** "$percentage\_of\_total\_vaccinations\_(in %)"**}},**  **{**$sort**:** **{**date**:** 1**,** vaccine**:** **-**1**}}**  **])**  **//note percentage of total vaccinations relative to population is calculated as a % as well** |
| --- |

(744 documents)

19. Vaccination Drivers. Specific to Germany, based on each daily new case, display the total vaccinations of each available vaccines after 20 days, 30 days, and 40 days.

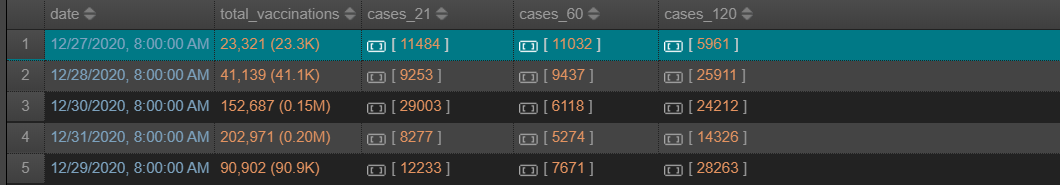
| db**.**country\_vac\_with\_covid19data**.**aggregate**([**  **{**$match**:** **{**location**:** "Germany"**}},**  **{**$match**:** **{**vaccinations\_by\_manufacturer\_data**:** **{**$exists**:** **true,** $ne**:** **[]}}},**  **{**$lookup**:** **{**  from**:** "country\_vac\_with\_covid19data"**,**  let**:** **{**date20**:** **{**$dateAdd**:** **{**startDate**:** "$date\_cleaned"**,** unit**:** "day"**,** amount**:** 20**}},** location**:** "$location"**},**  pipeline**:** **[{**$match**:** **{**$expr**:** **{**$and**:** **[{**$eq**:** **[**"$date\_cleaned"**,** "$$date20"**]},** **{**$eq**:** **[**"$location"**,** "$$location"**]}]}}}],**  as**:** "day\_20"  **}},**  **{**$lookup**:** **{**  from**:** "country\_vac\_with\_covid19data"**,**  let**:** **{**date30**:** **{**$dateAdd**:** **{**startDate**:** "$date\_cleaned"**,** unit**:** "day"**,** amount**:**30**}},** location**:** "$location"**},**  pipeline**:** **[{**$match**:** **{**$expr**:** **{**$and**:** **[{**$eq**:[**"$date\_cleaned"**,** "$$date30"**]},** **{**$eq**:[**"$location"**,** "$$location"**]}]}}}],**  as**:**"day\_30"  **}},**  **{**$lookup**:{**  from**:** "country\_vac\_with\_covid19data"**,**  let**:** **{**date40**:** **{**$dateAdd**:** **{**startDate**:** "$date\_cleaned"**,** unit**:** "day"**,** amount**:**40**}},** location**:** "$location"**},**  pipeline**:** **[{**$match**:** **{**$expr**:** **{**$and**:** **[{**$eq**:[**"$date\_cleaned"**,** "$$date40"**]},** **{**$eq**:** **[**"$location"**,** "$$location"**]}]}}}],**  as**:**"day\_40"  **}},**  **{**$project**:** **{**\_id**:** 0**,** date**:** 1**,** new\_cases\_cleaned**:** 1**,** day\_20**:** "$day\_20.vaccinations\_by\_manufacturer\_data"**,** day\_30**:** "$day\_30.vaccinations\_by\_manufacturer\_data"**,** day\_40**:** "$day\_40.vaccinations\_by\_manufacturer\_data"**}},**  **{**$project**:** **{**date**:** 1**,** new\_cases\_cleaned**:** 1**,** "day\_20.vaccine"**:** 1**,** "day\_20.total\_vaccinations\_cleaned"**:** 1**,** "day\_30.vaccine"**:** 1**,** "day\_30.total\_vaccinations\_cleaned"**:** 1**,** "day\_40.vaccine"**:** 1**,** "day\_40.total\_vaccinations\_cleaned"**:** 1**}}**  **])** |
| --- |

Within each array for example, we display the total vaccination of EACH available vaccine:



20. Vaccination Effects. Specific to Germany, on a daily basis, based on the total number of accumulated vaccinations (sum of total\_vaccinations of each vaccine in a day), generate the daily new cases after 21 days, 60 days, and 120 days.

| db**.**country\_vac\_with\_covid19data**.**aggregate**([**  **{**$match**:** **{**location**:**"Germany"**}},**  **{**$lookup**:** **{**  from**:**"country\_vac\_with\_covid19data"**,**  let**:{**date21**:** **{**$dateAdd**:** **{**startDate**:** "$date\_cleaned"**,** unit**:** "day"**,** amount**:** 21**}},** location**:** "$location"**},**  pipeline**:[{**$match**:** **{**$expr**:** **{**$and**:** **[{**$eq**:** **[**"$date\_cleaned"**,** "$$date21"**]},** **{**$eq**:** **[**"$location"**,** "$$location"**]}]}}}],**  as**:**"day21"**}},**  **{**$lookup**:** **{**  from**:** "country\_vac\_with\_covid19data"**,**  let**:** **{**date60**:** **{**$dateAdd**:** **{**startDate**:** "$date\_cleaned"**,** unit**:** "day"**,** amount**:** 60**}},** location**:** "$location"**},**  pipeline**:** **[{**$match**:** **{**$expr**:** **{**$and**:** **[{**$eq**:** **[**"$date\_cleaned"**,** "$$date60"**]},** **{**$eq**:** **[**"$location"**,** "$$location"**]}]}}}],**  as**:** "day60"**}},**  **{**$lookup**:** **{**from**:** "country\_vac\_with\_covid19data"**,** let**:** **{**date120**:** **{**$dateAdd**:** **{**startDate**:** "$date\_cleaned"**,** unit**:** "day"**,** amount**:** 120**}},** location**:** "$location"**},**  pipeline**:** **[{**$match**:** **{**$expr**:** **{**$and**:** **[{**$eq**:** **[**"$date\_cleaned"**,** "$$date120"**]},** **{**$eq**:** **[**"$location"**,** "$$location"**]}]}}}],**  as**:** "day120"**}},**  **{**$match**:** **{**vaccinations\_by\_manufacturer\_data**:** **{**$exists**:** **true,** $ne**:** **[]}}},**  **{**$project**:** **{**\_id**:** 0**,** date**:** "$date\_cleaned"**,** total\_vaccinations**:** **{**$sum**:** "$vaccinations\_by\_manufacturer\_data.total\_vaccinations\_cleaned"**},** cases\_21**:** "$day21.new\_cases\_cleaned"**,** cases\_60**:** "$day60.new\_cases\_cleaned"**,** cases\_120**:** "$day120.new\_cases\_cleaned"**}}**  **])** |
| --- |

(186 documents) Note: Here we accumulate the vaccines and calculate the total instead of displaying each individual vaccine.

# Task Allocation

| S/N | Tasks | Member In-charge |
| --- | --- | --- |
| 1 | *[relational database] ERD Development* | *All members* |
| 2 | *[relational database] Normalization to 3NF* | *All members* |
| 3 | *[relational database] SQL Queries 1-10* | *All members* |
| 4 | *[nonrelational database] noSQL development - Data Cleaning and Merging* | *All members* |
| 5 | *[nonrelational database] noSQL development Queries 1-10* | *All members* |
| 6 | *[nonrelational database] noSQL development Queries 11-20* | *All members* |

For the ERD Development, all members have equal contribution to this process. We each came up with our own ERD Diagrams and exchanged ideas, and eventually came up with the final ERD Diagram together by combining our ideas and deriving the best solution. Therefore, no specific member was in charge, but this was a team effort.

For the Normalization to 3NF, it was done on a single computer but everyone was present throughout the process, each researching the ways we can convert the dataset to a 3NF, everyone contributed by researching syntax and helping with the typing of code for repetitive portions. Hence, all members are in charge.

For the noSQL data cleaning and merging, all members came together to discuss the appropriate data types and thereafter, we typed the code for repetitive portions and copy pasted the code onto 1 final cleaning script. We then discussed the best way to merge the collections and helped each other to correct the syntax, until we got the desired collection. Therefore, all members contributed equally in the process.

For the SQL Queries and noSQL queries, we want to ensure accuracy and efficiency in query(as much as possible). Thus, we asked every team member to do all 30 queries on their own, then we came together to discuss and to check for differences in answers. As most answers that all members came up with are similar, it is difficult to distinguish who exactly contributed to which query as all members did their part in this. Hence, all members' contributions are equal in this portion as well.

1. <https://dev.mysql.com/doc/refman/8.0/en/type-conversion.html> [↑](#footnote-ref-0)
2. <https://dev.mysql.com/doc/refman/8.0/en/integer-types.html> [↑](#footnote-ref-1)
3. <https://docs.mongodb.com/manual/sharding/> [↑](#footnote-ref-2)
4. <https://www.scylladb.com/glossary/nosql-design-principles/> [↑](#footnote-ref-3)
5. <https://stackoverflow.com/questions/41510633/mongodb-date-and-isodate-parsing> [↑](#footnote-ref-4)
6. <https://www.jetbrains.com/datagrip/> [↑](#footnote-ref-5)
7. <https://www.mongodb.com/community/forums/t/creating-a-new-database-vs-a-new-collection-vs-a-new-cluster/99187/3> [↑](#footnote-ref-6)
8. <https://stackoverflow.com/questions/20087895/why-does-mongodb-takes-up-so-much-space> [↑](#footnote-ref-7)
9. <https://dev.mysql.com/doc/refman/8.0/en/view-algorithms.html> [↑](#footnote-ref-8)
10. <https://dev.mysql.com/doc/refman/8.0/en/switchable-optimizations.html#optflag_derived-merge> [↑](#footnote-ref-9)