

NANYANG
TECHNOLOGICAL
UNIVERSITY

BC2402 Design v Dev databases

Team 6

AU YEW RONG ROYDON (OU YAORONG) (U2021424J)

CAMERON LEE XIN YI (U2010300A)

LIM CHIEN HUI (U2021884G)

LOW LEE HANG (U2021604G)

MICHELLE LAM SU-ANN (U2021548K)

Contents

1. Introduction.....	3
1.1 Business Problem.....	3
2. Exploring data.....	3
3. Relational data model	4
3.1 ERD diagrams.....	4
3.2 Design considerations of ERD diagram	5
4. Differences.....	6
4.1 SQL vs NOSQL General Differences.....	6
4.2 SQL vs NOSQL Differences for Queries	8
5. Recommendations.....	8
5.2 Recommendations for WHO	8
6. Possible Improvements for WHO.....	9
6.1 Improvement in their data collected	9
7. Conclusion	10
8. References.....	11
9. Appendix.....	12
9.1 Expected output for SQL Queries.....	12
9.2 Expected output for noSQL Queries.....	18
9.3 List of Attributes for ERD	37

1. Introduction

1.1 Business Problem

In recent times, the emergence of a global pandemic – COVID-19 – has revealed fatal flaws in data collection and storage. These flaws have limited the efficacy of policies and community responses which could have prevented the spread of the disease. This problem arose due to poor implementation of the database and information overloading. Fundamentally, optimising the way data is stored is crucial in minimising the spread of this disease. Our team believes that leveraging on a suitable database to store this flood of information will be especially useful in assisting countries to respond quickly during this pandemic.

An appropriate database is capable of storing all information while still ensuring ease of access to every piece of useful information. This means that data that are not essential will also be stored in a manner that does not interfere with or disrupt the access to essential data. This way, relevant insights can be easily obtained thus aiding the decision-making process.

This report aims to explore the differences between relational data model and non-relational data models and analyse their individual strengths and weaknesses. Ultimately, this report aims to determine a model that will be more suitable for WHO in solving health-related problems, especially those brought about by COVID-19.

2. Exploring data

3 datasets were provided – ‘country_vaccinations’, ‘country_vaccinations_by_manufacturer’ data and ‘covid19data’. We further explored these 3 datasets and discovered a few interesting findings that would affect our querying approach.

Inconsistency in ‘covid19data’ Dataset

The covid19data dataset consists of 2 similar column attributes - location and continent. These 2 columns both provide similar information, which is the geographical location. We investigated the differences in these columns. Upon examination, we found that the number of countries under the continent ‘Asia’ was inaccurate. The table only contained 50 countries under Asia. However, there should be 51 countries. Further exploration revealed that North Korea was not included as a country classified as Asia. Therefore, the integrity and accuracy of the data is at risk. Any queries based on the column attribute ‘continent’ may not produce the most reliable results.

Inconsistency in Total Number of Vaccinations

The ‘total_vaccinations’ column attribute is repeated across 2 tables – ‘covid19data’ and ‘country_vaccinations_by_manufacturer’. However, the values under the total_vaccinations’ column attribute were not consistent and differed across the 2 tables. We investigated this discrepancy to decide on the most appropriate column attribute to be used. We discovered that there were many missing and null values under this column in the ‘covid19data’ table. Consequently, basing any of our queries on that column attribute containing many missing values, would not be appropriate or reliable.

3. Relational data model

3.1 ERD diagrams

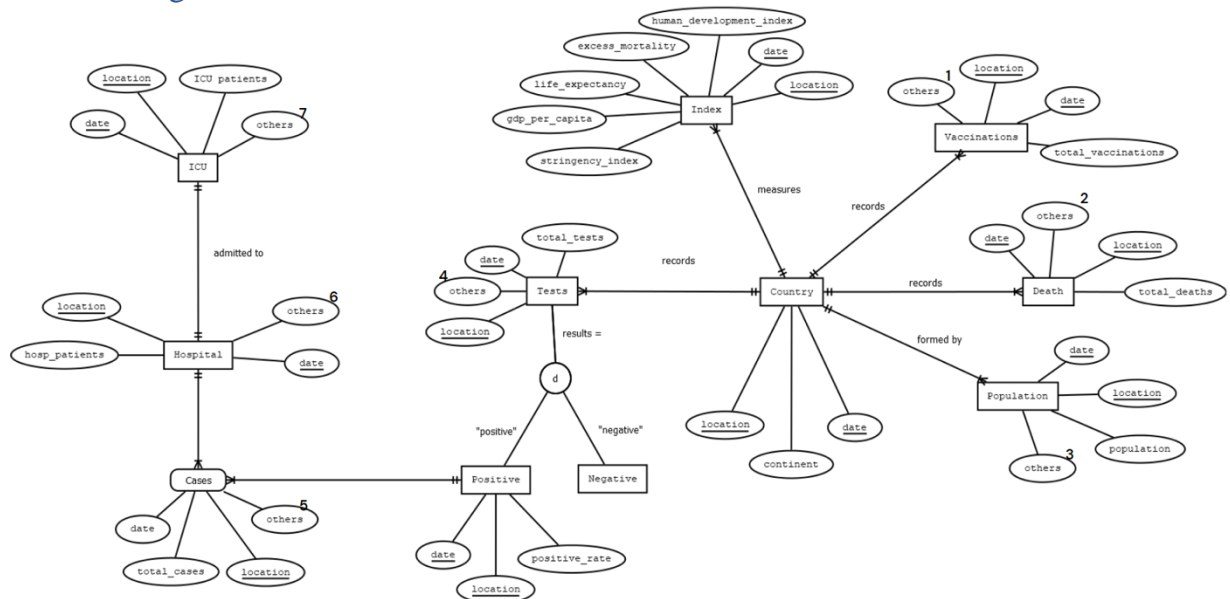


Figure 1: Covid19 Data

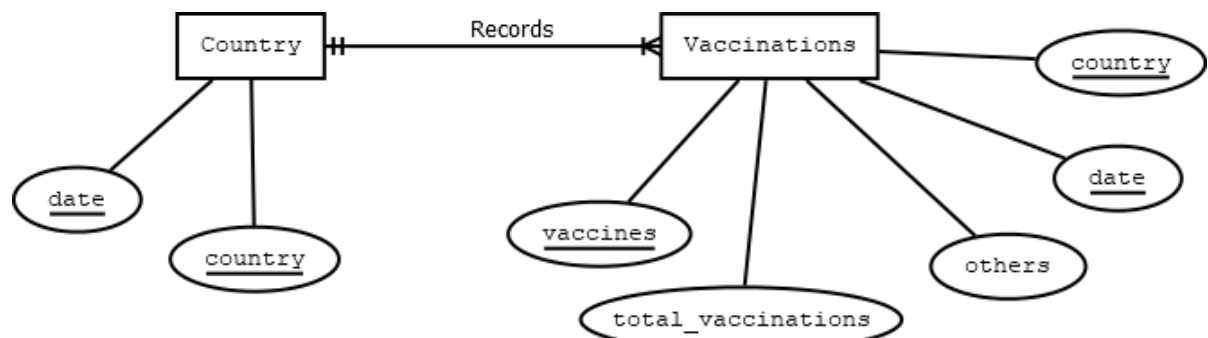


Figure 2: Country_Vaccinations

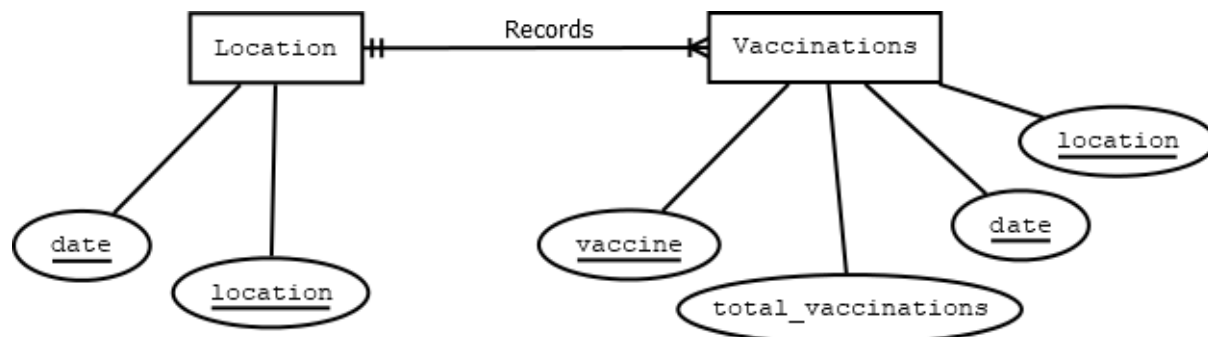


Figure 3: Country_Vaccinations_By_Manufacturer

3.2 Design considerations of ERD diagram

Our team scrutinized the 3 databases, namely, 'covid19data', 'country_vaccinations_by_manufacturer' and 'country_vaccinations' and created our ERD diagram based on the original databases in order to prevent loss of crucial information.

For 'country_vaccinations' and 'country_vaccinations_by_manufacturer', our team decided to reorganize the data into second normal form (2NF), as shown in Figure 1 and Figure 2. On the other hand, our team decided to apply normalization on 'covid19data' to the third normal form (3NF) as we believe that 3NF will improve the performance of the queries.

covid19data

For the covid19data, our team has created an ERD with a total of 10 main entities, as shown in figure 1.

Firstly, we created a Country entity, as it serves as the main connection to the other entities. The Country entity has the attributes – location, date, and continent. Location and date form the composite primary key for the Country entity. This ensures that there will be distinct locations and dates that do not overlap. Likewise, for the remaining 9 entities, we have identified location and date as the foreign keys to these entities as these foreign keys take reference from the primary key of the Country entity.

Secondly, our team created an Index entity, as Country measures indexes. The Index entity includes the attributes gdp_per_capita, stringency_index, etc. The Country entity and Index entity have a 1:M relationship between them as 1 country measures 1 to many Indexes, while 1 index is measured by only 1 country.

Thirdly, the Population entity is created, as Country is formed by a population. The Population entity has the attributes population, population_density, etc. Like the Index entity, the Population entity has a 1:M relationship with Country as well.

Fourthly, the Vaccination entity is created, as the Country entity records vaccinations. The Vaccination entity has the attributes total_vaccinations, people_fully_vaccinated, etc. Likewise, there is a 1:M relationship between Vaccination and Country, as each vaccination is recorded by the respective country, while each country can record multiple vaccinations.

From the above, we can identify that each country is able to record their death rate that is caused by covid19. Hence, we created an entity for Death, which has the attributes total deaths, new_deaths, etc. Similarly, there is a 1:M relationship between Death and Country.

We then created the Test entity to represent total_tests that were recorded for each Country. These tests could either result in a "positive" or "negative" result. If positive, the test will be confirmed as a covid case and will be subsequently admitted to the hospital. The Cases entity created is an associative entity which is represented by the rounded rectangle. 1 positive test results in 1 to many covid cases while 1 covid case is confirmed only when 1 test results in a positive result.

Lastly, the Hospital admits patients to the ICU. The Hospital has a 1-1 relationship w ICU as 1 Hospital admission results in at most 1 ICU admission.

Country Vaccinations

Based on Figure 2, our team created the Country entity as it was the main part that connects with vaccinations. Country and date are the primary keys as they uniquely identify each country and will not be null. Afterwards, our team created Vaccinations entity which consist of the number of total vaccinations. For this table, our team identified the composite primary keys to be country, date, and

vaccines. Country and date from the Vaccinations entity is also a foreign key, as it can reference back to the location entity.

Country Vaccinations By Manufacturer

As mentioned in the “country_vaccinations” table above, our team applied the same design consideration for this table, as shown in Figure 3. However, the only difference is that the total_vaccinations of this entity is segregated by their manufacturer and does not consider the individual’s doses taken.

4. Differences

4.1 SQL vs NOSQL General Differences

Each type of database – relational and non-relational database – has its own individual strengths and weaknesses. There is no one single type of database that can suit every organisation’s needs and circumstances. Rather, depending on the organization’s objectives, one database will be better catered towards achieving their goals.

Properties

Fundamentally, all the key differences between SQL and noSQL databases are attributed towards the difference in properties.

Relational SQL databases follow the ACID Consistency Model (Atomicity, Consistency, Isolation and Durability). This model ensures that data is only inserted if all the operations within the transaction are successful. In comparison, non-relational noSQL databases follow the BASE Consistency Model (Basically Available, Soft State and Eventual Consistency). This model prioritises scalability over consistency, unlike the ACID Consistency Model.

Consistency

One widely known difference between relational and non-relational databases is database properties, which contributes to the difference in database structure and its suitability towards handling different types of data.

Organizations with more unstructured data will find it more convenient to rely on non-relational databases than to rely on relational databases. This is due to the difference in the way data is organised. Data in relational databases are organised in tables, where each column represents an attribute. This creates a predefined structure that is more suited for well-structured data. In contrast, non-relational databases are based on more flexible data models. This allows for easier data insertion into non-relational databases as data inserted need not be confined to a specific structure. Unlike in relational databases, where data can only be inserted, provided they follow the predefined structure.

Scalability

Relational databases are vertically scalable while non-relational databases are horizontally scalable. Vertical scaling involves a trade-off between performance and cost. To improve the performance even as the database expands, upgrading is required. This involves increasing CPU, RAM and storage, which is rather costly. Whereas, maintaining and expanding a non-relational database with horizontal scaling is more cost-effective. This process is known as sharding, which enables load to be distributed across multiple servers. To do this, more systems with smaller CPU, RAM and storage configurations are used. This means that non-relational databases can handle large volumes of data at a lower cost, unlike relational databases.

Speed

Non-relational databases are generally well-known to be faster than relational databases in terms of reading and writing speed. This is mainly due to non-relational databases being specifically designed for unstructured data. This means that it can store a wide variety of data types, regardless of whether it is document-oriented, column-oriented, or graph-based etc. Each data entity inserted is stored together and not segmented or partitioned. In the case of performing read or write operations on a single data entity, a non-relational database will be relatively faster.

However, it is not always the case that non-relational databases will always be faster. In fact, as the data collection becomes larger, relational databases gradually become faster. This is because relational databases are normalized databases, in which data is broken down into multiple tables. The issue of data redundancy and data duplication is minimised. This is especially helpful for queries that require joins or updates.

Geospatial Feature

One main feature of non-relational databases is its ability to support spatial databases. MongoDB, in particular, is capable of doing that. A spatial database is built to capture and store points, lines, and areas of cartographic information.

Importing Method

Non-relational databases, specific to MongoDB, has an additional feature allowing for greater flexibility. This feature is the importing of data. This flexibility may create discrepancies, despite importing the same dataset. Whereas the predefined structure required for relational databases will be helpful in ensuring data consistency and maintaining data integrity.

The data can be imported into MongoDB via 2 different methods – (1) Importing the data through MongoDB Compass, (2) Importing the data directly into the noSQL booster platform. With the former method, MongoDB Compass encounters an issue while attempting to import the dataset that contains an array, if any. Consequently, the dataset shrinks as the entire array is removed from the dataset. In contrast, arrays remain in the data when the latter method is used. As such, manual unwinding of the array is required, through the use of the ‘\$unwind’ operator. However, this operator removes the entire array as long as there is a missing or null field. This could lead to important information possibly being removed, unknowingly. Despite the same dataset being used, the amount of information available may be less than the original dataset that was initially imported. Consequently, retrieving information from this dataset may be inaccurate or unreliable. We discovered these issues while attempting to use the ‘covid19data’ dataset. Hence, do take note that we imported the ‘covid19data_2’ dataset through MongoDB Compass to use for our Appendix C noSQL queries instead.

The table below provides a brief summary of the key differences between relational and non-relational databases.

Overall Summary of the Differences in Databases		
Feature	Relational Databases	Non-Relational Databases
Flexibility	Low	High
Scalability	Costly	Cost-Effective
Supporting Geospatial Data	Poor	Good
Consistency	Good	Poor

4.2 SQL vs NOSQL Differences for Queries

Queries

When constructing our queries, we observed that there is minimal difference in the amount of time taken for each query in both SQL and noSQL. This indicates that efficiency of the code between SQL and noSQL is comparable. However, this could differ as the dataset increases.

One main advantage of noSQL queries is its ability to leverage on the aggregation pipeline. This is advantageous as changes made to the documents are passed through the pipeline. This feature provides a systematic approach to solving most of our questions. Unlike SQL, additional collections were not required. Minimising the number of collections makes querying more efficient. The larger number of tables in SQL, each storing different types of information, may create confusion among users.

However, we observed that codes for noSQL queries may be lengthy and thus more complex. Whereas SQL queries are more intuitive. Consequently, SQL queries are relatively simpler to construct and debug.

5. Recommendations

5.2 Recommendations for WHO

One major health-related aspect WHO is interested in is COVID-19 data. Generally, health-related data, such as COVID-19 data, are often largely unstructured, constantly being updated and revised in real-time in large amounts and spatial in nature. Keeping in mind these characteristics of health-related data like COVID-19, we further examine the differences in relational and non-relational databases to decide on a more suitable option for WHO.

In the context of COVID-19, the flexible nature of non-relational databases will prove to be valuable. The process of inserting and updating data is especially efficient provided that it does not follow a predefined structure as mentioned. COVID-19 data will be obtained across many different countries, but not all countries will have the same amount of information. In other words, certain countries may not have information for certain attributes. Data received will be largely inconsistent, with no fixed structure to it. The flexibility of a non-relational database allows for ease of updating and insertion of data. Time is saved as existing data received does not need to be constantly refined before it can be inserted into the database. This way, WHO can continue being a reliable source of information as they are able to keep up with the fast pace of information being received.

Besides that, if such sparse data were to be inserted into a relational database, this would result in the creation of unnecessary cells that have missing or null values. This is because in a relational database, every attribute is required to be filled in due to its predefined structure, as mentioned above. This could lead to inefficient implementation of databases. Furthermore, this could mislead users into assuming these missing values are information that are available but have yet to be updated. Users might inaccurately account for these missing or null values and expect results higher than expected. In reality, those missing or null values for that particular country are unavailable and no such attribute exists at all.

One example of such a misleading situation happening can be seen from Question 8 of the SQL queries. There was an inconsistency in terms of the dates recorded for Germany, where the 'covid19data' table has earlier records of dates than the 'country_vaccinations_by_manufacturer' table. However, majority of the attributes' information related to the 'covid19data' table is missing or null. This could lead to users misinterpreting those dates to still be important. Users assume that these attributes exist and information regarding those attributes are available. Consequently, this led to us inaccurately accounting for these dates in our final query, with the assumption that the attribute data is available but has yet to be updated.

Our team would also like to advise WHO to make use of non-relational databases when recording COVID-19-related data to avoid misleading users, or possibly conveying inaccurate information.

Given that WHO is a well-established organisation known for providing a large range of global data, their database is constantly expanding. Moreover, as the pandemic has yet to stabilise, COVID-19 related datasets are likely to continue growing over the years. Thus, it would be financially beneficial for them to leverage on non-relational databases, that is capable of handling large volumes of data while still ensuring optimal performance.

One main health-related aspect WHO is focusing on in recent times, is COVID-19 data. The geospatial feature of non-relational databases will be especially relevant in this aspect for WHO. COVID-19 related data is inherently spatial in nature. To minimise the spread of this disease globally, it is critical to identify the exact location of individuals tested positive for COVID-19 and to track their movements. By relying on non-relational databases, WHO can offer additional valuable geospatial insights during this pandemic. Tracking geospatial data will be helpful in assisting countries to respond quickly in the midst of this pandemic.

However, one critical aspect WHO should be aware of when leveraging on non-relational databases is its importing method and the resulting differences. A non-relational database with the use of MongoDB Compass and noSQL booster platform, gives WHO 2 different importing options. Each option will result in a different dataset.

Based on the aforementioned reasons, it is clear that the effectiveness of relational and non-relational databases will vary depending on the circumstances. In the case of WHO, non-relational databases will be a more suitable option for them. This is mainly due to their need to update their database of information quickly and with large amounts at any one time. Ultimately, in the context of COVID-19 related data, the use of non-relational databases is also highly recommended. This is due to its additional ability to handle and track geospatial data. Overall, our team would recommend the use of non-relational databases to WHO, provided they are focused on recording data related to the recent global pandemic – COVID-19.

6. Possible Improvements for WHO

6.1 Improvement in their data collected

Currently, SQL queries in Appendix B do not require any indexing because the data columns are separated into third normal form (3NF). This is a normalising principle that reduces the duplication of data, avoids data anomalies, ensures referential integrity, and simplifies data management.

One possible improvement would be to make use of indexing. Indexing practices can reduce memory usage and speed up the process of retrieving data from a database. Overall, this improves the performance of queries. It will be especially beneficial when retrieving records that is frequently searched for by users.

We also found out that columns like death and hospital beds are rarely used for this particular set of queries. Hence, we should place these columns into another table to improve the efficiency of our queries by reducing information overloading. This table contains all columns that are rarely used for queries. When the queries are needed, we can then extract the relevant columns and place them into a table to be queried.

7. Conclusion

This report reveals that both SQL and noSQL has its own pros and cons. While noSQL has its own limitations, it can offer greater benefits to WHO. This is mainly due to the flexibility of the non-relational database and its ability to store large volumes of information efficiently. These characteristics will be especially helpful in assisting a well-established organization like WHO, in providing reliable and accurate health-related information. At the same time, implementing such a database allows essential and valuable data to be easily found.

Taking into consideration WHO's objectives, our team's overall recommendation to them would be to rely on non-relational databases to achieve their goals.

8. References

1. Ruihan, W., & Zongyan, Y. (2018). *SQL vs NoSQL: A Performance Comparison*. University of Rochester.
2. *What's the Difference? Relational vs Non-Relational Databases*. (2021, June 1). Logi Analytics. <https://www.logianalytics.com/relational-vs-non-relational-databases/>
3. Pawlan, D. (2021, April 29). *Relational vs. Non-Relational Database: Pros & Cons*. The Aloa Blog. <https://aloha.co/blog/relational-vs-non-relational-database-pros-cons>
4. *The BASE Model Offers an Alternative to Database Engineering*. (2020, July 25). Lifewire. <https://www.lifewire.com/abandoning-acid-in-favor-of-base-1019674>

9. Appendix

9.1 Expected output for SQL Queries

Question	Expected Output																		
/* Importing instructions: Head to Server -> Data Import -> Import from Self Contained File -> Start Import */																			
1	<p>// Filtering based on continent. //Chosen answer SELECT SUM(DISTINCT(population)) FROM covid19data WHERE continent = "Asia";</p> <table border="1"> <tr> <td></td><td>SUM(DISTINCT(population))</td></tr> <tr> <td>▶</td><td>4614068610</td></tr> </table> <p>//Filtering based on location SELECT DISTINCT(population) FROM covid19data WHERE location = "Asia";</p> <table border="1"> <tr> <td></td><td>population</td></tr> <tr> <td>▶</td><td>4639847425.0</td></tr> </table> <p>//This is because, in this table, there are only 50 continents. However, Asia should contain 51 locations. We discovered that North Korea is not included as a location.</p>		SUM(DISTINCT(population))	▶	4614068610		population	▶	4639847425.0										
	SUM(DISTINCT(population))																		
▶	4614068610																		
	population																		
▶	4639847425.0																		
2	<p>// Where ASEAN countries are Brunei, Myanmar, Cambodia, Indonesia, Laos, Malaysia, Philippines, Singapore, Thailand and Vietnam SELECT SUM(DISTINCT(population)) AS total_population FROM covid19data WHERE location IN ("Brunei", "Myanmar", "Cambodia", "Indonesia", "Laos", "Malaysia", "Philippines", "Singapore", "Thailand", "Vietnam");</p> <table border="1"> <tr> <td></td><td>total_population</td></tr> <tr> <td>▶</td><td>667301412</td></tr> </table>		total_population	▶	667301412														
	total_population																		
▶	667301412																		
3	<p>SELECT DISTINCT(source_name) FROM country_vaccinations ORDER BY source_name;</p> <table border="1"> <tr> <td></td><td>source_name</td></tr> <tr> <td>▶</td><td>Africa Centres for Disease Control and Prevention</td></tr> <tr> <td></td><td>Cayman Islands Government</td></tr> <tr> <td></td><td>Centers for Disease Control and Prevention</td></tr> <tr> <td></td><td>Costa Rican Social Security Fund</td></tr> <tr> <td></td><td>COVID-19 Malta Public Health Response Team</td></tr> <tr> <td></td><td>COVID-19 Vaccine Information Platform</td></tr> <tr> <td></td><td>Directorate General for Health via Data Science ...</td></tr> <tr> <td></td><td>Directorate General of Health Services</td></tr> </table>		source_name	▶	Africa Centres for Disease Control and Prevention		Cayman Islands Government		Centers for Disease Control and Prevention		Costa Rican Social Security Fund		COVID-19 Malta Public Health Response Team		COVID-19 Vaccine Information Platform		Directorate General for Health via Data Science ...		Directorate General of Health Services
	source_name																		
▶	Africa Centres for Disease Control and Prevention																		
	Cayman Islands Government																		
	Centers for Disease Control and Prevention																		
	Costa Rican Social Security Fund																		
	COVID-19 Malta Public Health Response Team																		
	COVID-19 Vaccine Information Platform																		
	Directorate General for Health via Data Science ...																		
	Directorate General of Health Services																		
4	<p>SELECT date, daily_vaccinations FROM country_vaccinations WHERE country = "Singapore" AND date BETWEEN "3/1/2021" AND "5/31/2021";</p>																		

	<table><tr><th></th><th>date</th><th>daily_vaccinations</th></tr><tr><td>▶</td><td>3/1/2021</td><td>15004.0000</td></tr><tr><td></td><td>3/2/2021</td><td>14763.0000</td></tr><tr><td></td><td>3/3/2021</td><td>14523.0000</td></tr><tr><td></td><td>3/4/2021</td><td>14282.0000</td></tr><tr><td></td><td>3/5/2021</td><td>13615.0000</td></tr><tr><td></td><td>3/6/2021</td><td>12948.0000</td></tr><tr><td></td><td>3/7/2021</td><td>12281.0000</td></tr><tr><td></td><td>3/8/2021</td><td>12325.0000</td></tr></table> <p>//We chose to use 'daily_vaccinations' attribute instead of 'total_vaccinations' attribute. We do not advise using the 'total_vaccinations' attribute because there are many values of '0'.</p> <p>This implies that the total_vaccinations attribute may not be updated on a daily basis. Hence, using the 'total_vaccinations' attribute may result in inaccurate and unreliable results. Using 'daily_vaccinations' attribute, we will be able to find total vaccinations for each day in Singapore.</p>		date	daily_vaccinations	▶	3/1/2021	15004.0000		3/2/2021	14763.0000		3/3/2021	14523.0000		3/4/2021	14282.0000		3/5/2021	13615.0000		3/6/2021	12948.0000		3/7/2021	12281.0000		3/8/2021	12325.0000
	date	daily_vaccinations																										
▶	3/1/2021	15004.0000																										
	3/2/2021	14763.0000																										
	3/3/2021	14523.0000																										
	3/4/2021	14282.0000																										
	3/5/2021	13615.0000																										
	3/6/2021	12948.0000																										
	3/7/2021	12281.0000																										
	3/8/2021	12325.0000																										
5	<p>SELECT date, total_vaccinations FROM covid19data WHERE total_vaccinations > 0 AND location = "Singapore" ORDER BY date ASC LIMIT 1;</p> <table><tr><th></th><th>date</th><th>total_vaccinations</th></tr><tr><td>▶</td><td>2021-01-11</td><td>3400.0</td></tr></table> <p>//To verify that the above information regarding total_vaccinations and its date is accurate, we compare it with total_vaccinations data in the country_vaccinations table as seen in the query below.</p> <p>SELECT total_vaccinations, date FROM country_vaccinations WHERE total_vaccinations > 0 AND country = "Singapore" ORDER BY date ASC LIMIT 1;</p> <table><tr><th></th><th>total_vaccinations</th><th>date</th></tr><tr><td>▶</td><td>3400.0000</td><td>1/11/2021</td></tr></table> <p>//As seen, total_vaccinations amount of 3400 and date in the country_vaccinations table corresponds to the total_vaccinations amount and date in the covid19data table. The consistency in data across 2 tables further support that the information obtained from the covid19data table is accurate.</p>		date	total_vaccinations	▶	2021-01-11	3400.0		total_vaccinations	date	▶	3400.0000	1/11/2021															
	date	total_vaccinations																										
▶	2021-01-11	3400.0																										
	total_vaccinations	date																										
▶	3400.0000	1/11/2021																										
6	<p>SELECT SUM(new_cases) FROM covid19data WHERE date >= "2021-01-11" AND location = "Singapore";</p> <table><tr><th></th><th>SUM(new_cases)</th></tr><tr><td>▶</td><td>3710</td></tr></table>		SUM(new_cases)	▶	3710																							
	SUM(new_cases)																											
▶	3710																											
7	<p>SELECT SUM(new_cases) FROM covid19data WHERE date < "2021-01-11" AND location = "Singapore";</p> <table><tr><th></th><th>SUM(new_cases)</th></tr><tr><td>▶</td><td>58907</td></tr></table>		SUM(new_cases)	▶	58907																							
	SUM(new_cases)																											
▶	58907																											
8	//Method 1: Using VIEWS																											

```
//Check that VIEWS have not yet been created
```

```
drop VIEW if EXISTS percentageNew;  
drop VIEW if EXISTS vaccinations;
```

//2 views are created to filter out only the relevant information need that is related to Germany.

//This view created contains information on the daily percentage of new cases in Germany.

```
CREATE VIEW percentageNew
```

```
AS
```

```
SELECT (new_cases/population)*100 AS percentages, date FROM covid19data  
WHERE location = "Germany"
```

```
GROUP BY date
```

```
ORDER BY date;
```

//This view created contains information on the total vaccination of each available vaccine in Germany.

```
CREATE VIEW vaccinations
```

```
AS
```

```
SELECT          total_vaccinations,          vaccine,          date          FROM  
country_vaccinations_by_manufacturer
```

```
WHERE location = "Germany"
```

```
GROUP BY date, vaccine
```

```
ORDER BY date;
```

	total_vaccinations	vaccine	date
▶	0	Johnson&Johnson	2020-12-27
	2	Moderna	2020-12-27
	0	Oxford/AstraZe...	2020-12-27
	23319	Pfizer/BioNTech	2020-12-27
	0	Johnson&Johnson	2020-12-28
	2	Moderna	2020-12-28
	0	Oxford/AstraZe...	2020-12-28
	41137	Pfizer/BioNTech	2020-12-28

//Take note, there is an inconsistency in dates for covid19data table and country_vaccinations_by_manufacturer table.

The covid19data table records a larger range of dates than the country_vaccinations_by_manufacturer table. This is because the first date recorded in covid19data table for Germany is earlier than the first date recorded in country_vaccinations_by_manufacturer table for Germany.

// Display the herd immunity estimation for Germany

```
SELECT percentages, percentageNew.date, total_vaccinations, vaccine
```

```
FROM percentageNew LEFT JOIN vaccinations ON percentageNew.date =  
vaccinations.date
```

```
GROUP BY percentageNew.date, vaccine
```

```
ORDER BY percentageNew.date;
```

	<table><tr><th>percentages</th><th>date</th><th>total_vaccinations</th><th>vaccine</th></tr><tr><td>0.0014441907694845355</td><td>2020-03-15</td><td>NULL</td><td>NULL</td></tr><tr><td>0.0017628675756435198</td><td>2020-03-16</td><td>NULL</td><td>NULL</td></tr><tr><td>0.002369188989608928</td><td>2020-03-17</td><td>NULL</td><td>NULL</td></tr><tr><td>0.003664186497783078</td><td>2020-03-18</td><td>NULL</td><td>NULL</td></tr><tr><td>0.0035722834488158797</td><td>2020-03-19</td><td>NULL</td><td>NULL</td></tr><tr><td>0.0054043766977074185</td><td>2020-03-20</td><td>NULL</td><td>NULL</td></tr><tr><td>0.002822736503992501</td><td>2020-03-21</td><td>NULL</td><td>NULL</td></tr><tr><td>0.003174832600685012</td><td>2020-03-22</td><td>NULL</td><td>NULL</td></tr></table> <p>//While the covid19data table has a larger range of dates for Germany, majority of the information seems to be missing until 2020-12-27. This includes data such as 'total_vaccinations' data. This implies that Germany may actually have more total_vaccinations data before 2020-12-27, which has not been updated or recorded yet. Hence, 'total_vaccinations' data will be recorded as null in Germany for dates before 2020-12-27 when finding the herd immunity estimation. It is important to still take note of and record these dates to be mindful that Germany may have more data contributing to Germany's total_vaccinations data that has not yet been updated.</p>	percentages	date	total_vaccinations	vaccine	0.0014441907694845355	2020-03-15	NULL	NULL	0.0017628675756435198	2020-03-16	NULL	NULL	0.002369188989608928	2020-03-17	NULL	NULL	0.003664186497783078	2020-03-18	NULL	NULL	0.0035722834488158797	2020-03-19	NULL	NULL	0.0054043766977074185	2020-03-20	NULL	NULL	0.002822736503992501	2020-03-21	NULL	NULL	0.003174832600685012	2020-03-22	NULL	NULL
percentages	date	total_vaccinations	vaccine																																		
0.0014441907694845355	2020-03-15	NULL	NULL																																		
0.0017628675756435198	2020-03-16	NULL	NULL																																		
0.002369188989608928	2020-03-17	NULL	NULL																																		
0.003664186497783078	2020-03-18	NULL	NULL																																		
0.0035722834488158797	2020-03-19	NULL	NULL																																		
0.0054043766977074185	2020-03-20	NULL	NULL																																		
0.002822736503992501	2020-03-21	NULL	NULL																																		
0.003174832600685012	2020-03-22	NULL	NULL																																		
9	<p>//Method 2: Without using VIEWS</p> <p>//The difference between this method (method 2) and method 1 above, is that //method 2 does not display NULL values.</p> <p>//This method is less accurate than above. Further explanation below.</p> <pre>SELECT new_cases/population*100, covid19data.date, country_vaccinations_by_manufacturer.total_vaccinations, vaccine FROM covid19data JOIN country_vaccinations_by_manufacturer ON covid19data.date = country_vaccinations_by_manufacturer.date WHERE covid19data.location = "Germany" AND country_vaccinations_by_manufacturer.location = "Germany" GROUP BY covid19data.date, vaccine ORDER BY covid19data.date;</pre> <table><tr><th>percentages</th><th>date</th><th>total_vaccinations</th><th>vaccine</th></tr><tr><td>0</td><td>2020-01-06</td><td>NULL</td><td>NULL</td></tr><tr><td>0</td><td>2020-01-18</td><td>NULL</td><td>NULL</td></tr><tr><td>0</td><td>2020-01-24</td><td>NULL</td><td>NULL</td></tr><tr><td>0.0000011935460904830872</td><td>2020-01-27</td><td>NULL</td><td>NULL</td></tr><tr><td>0.0000035806382714492613</td><td>2020-01-28</td><td>NULL</td><td>NULL</td></tr><tr><td>0</td><td>2020-01-29</td><td>NULL</td><td>NULL</td></tr><tr><td>0</td><td>2020-01-30</td><td>NULL</td><td>NULL</td></tr><tr><td>0.0000011935460904830872</td><td>2020-01-31</td><td>NULL</td><td>NULL</td></tr></table> <p>//When VIEWS are not used, this resulting query implies that the earliest data being recorded in Germany only begins from 2020-12-27. However, this is not the case. This is misleading as it could inaccurately imply that the dates before 2020-12-27 are irrelevant and do not contribute total_vaccination date in Germany at all. As mentioned above, this is not an accurate assumption.</p>	percentages	date	total_vaccinations	vaccine	0	2020-01-06	NULL	NULL	0	2020-01-18	NULL	NULL	0	2020-01-24	NULL	NULL	0.0000011935460904830872	2020-01-27	NULL	NULL	0.0000035806382714492613	2020-01-28	NULL	NULL	0	2020-01-29	NULL	NULL	0	2020-01-30	NULL	NULL	0.0000011935460904830872	2020-01-31	NULL	NULL
percentages	date	total_vaccinations	vaccine																																		
0	2020-01-06	NULL	NULL																																		
0	2020-01-18	NULL	NULL																																		
0	2020-01-24	NULL	NULL																																		
0.0000011935460904830872	2020-01-27	NULL	NULL																																		
0.0000035806382714492613	2020-01-28	NULL	NULL																																		
0	2020-01-29	NULL	NULL																																		
0	2020-01-30	NULL	NULL																																		
0.0000011935460904830872	2020-01-31	NULL	NULL																																		
10	<p>//Check that VIEWS have not yet been created</p> <pre>drop VIEW if EXISTS new_cases_dates; drop VIEW if EXISTS vac_type;</pre> <p>//2 views are created to filter out only the relevant information need that is related to Germany.</p> <p>//This view created contains information on the dates of each daily new_case in Germany and the corresponding dates after 20 days, 30 days and 40 days.</p>																																				

```
CREATE VIEW new_cases_dates (date, new_cases, 20days, 30days, `40days`)
AS
SELECT date, new_cases, date + INTERVAL +20 DAY AS 20days, date + INTERVAL
+30 DAY AS 30days,
date + INTERVAL +40 DAY AS 40days
FROM covid19data
WHERE new_cases > 0 AND location = 'Germany';
```

//This view created contains total_vaccinations (accumulated) of each available vaccine for each day in Germany.

```
CREATE VIEW vac_type (date, total_vaccinations, vaccine)
AS
SELECT date, total_vaccinations, vaccine
FROM country_vaccinations_by_manufacturer
WHERE location = 'Germany';
```

//Displays the total_vaccinations of each available vaccine in Germany after 20 days.

```
SELECT t1.date, 20days, total_vaccinations, vaccine
FROM new_cases_dates t1
LEFT JOIN vac_type t2 ON t1.`20days` = t2.date
ORDER BY t1.date;
```

	date	20days	total_vaccinations	vaccine
▶	2020-01-27	2020-02-16	NULL	NULL
	2020-01-28	2020-02-17	NULL	NULL
	2020-01-31	2020-02-20	NULL	NULL
	2020-02-01	2020-02-21	NULL	NULL
	2020-02-02	2020-02-22	NULL	NULL
	2020-02-03	2020-02-23	NULL	NULL
	2020-02-07	2020-02-27	NULL	NULL
	2020-02-09	2020-02-29	NULL	NULL

//Displays the total_vaccinations of each available vaccine in Germany after 30 days.

```
SELECT t1.date, 30days, total_vaccinations, t2.vaccine
FROM new_cases_dates t1
LEFT JOIN vac_type t2 ON t1.`30days` = t2.date
ORDER BY t1.date;
```

	date	30days	total_vaccinations	vaccine
▶	2020-01-27	2020-02-26	NULL	NULL
	2020-01-28	2020-02-27	NULL	NULL
	2020-01-31	2020-03-01	NULL	NULL
	2020-02-01	2020-03-02	NULL	NULL
	2020-02-02	2020-03-03	NULL	NULL
	2020-02-03	2020-03-04	NULL	NULL
	2020-02-07	2020-03-08	NULL	NULL
	2020-02-09	2020-03-10	NULL	NULL

//Displays the total_vaccinations of each available vaccine in Germany after 40 days.

```
SELECT t1.date, 40days, total_vaccinations, t2.vaccine
```



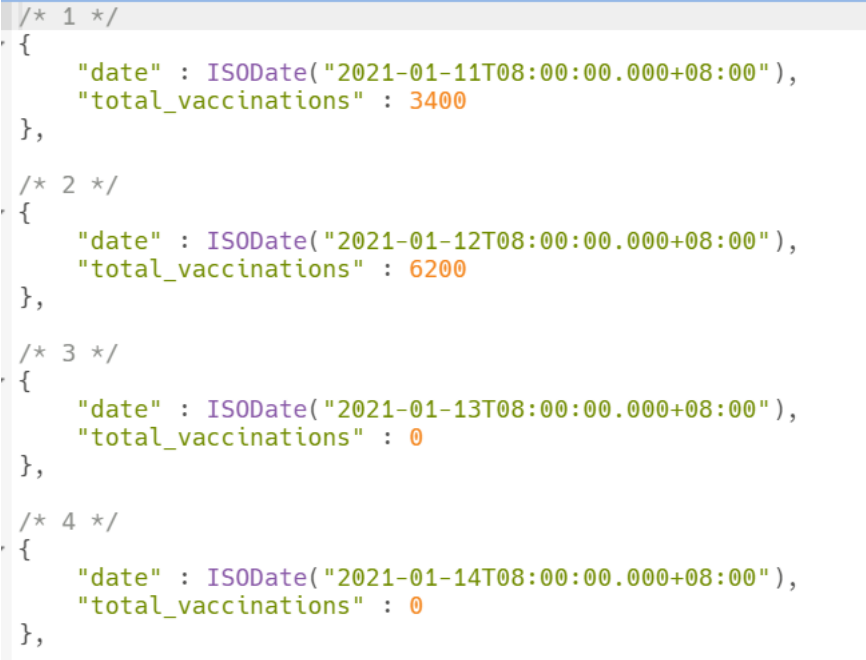
```

FROM new_cases_dates t1
LEFT JOIN vac_type t2 ON t1.`40days` = t2.date
ORDER BY t1.date;

```

	date	40days	total_vaccinations	vaccine
►	2020-01-27	2020-03-07	NULL	NULL
	2020-01-28	2020-03-08	NULL	NULL
	2020-01-31	2020-03-11	NULL	NULL
	2020-02-01	2020-03-12	NULL	NULL
	2020-02-02	2020-03-13	NULL	NULL
	2020-02-03	2020-03-14	NULL	NULL
	2020-02-07	2020-03-18	NULL	NULL
	2020-02-09	2020-03-20	NULL	NULL

9.2 Expected output for noSQL Queries

Question	Expected Output
	<p>/* Importing instructions:</p> <ol style="list-style-type: none"> 1) Create database 2) Create 3 collections 3) Import each JSON file to the respective collection, through MongoDB Compass <p>**TAKE NOTE: For covid19data collection, we imported the covid19data_2 JSON file from ntulearn as we faced issues with the other covid19data file.</p> <p>Thus, for all queries here based on covid19data collection, it is based off the covid19data_2 JSON file. */</p>
1	<pre>db.country_vaccinations.aggregate([{\$match: {"country": "Singapore"}}, {\$project: {_id:0, "date": {\$convert :{input :"\$date", to : "date"}}, "total_vaccinations":{\$convert :{input :"\$total_vaccinations", to : "int"}}}}, {\$sort: {date: 1}}])</pre>  <pre>{ "date" : ISODate("2021-01-11T08:00:00.000+08:00"), "total_vaccinations" : 3400 }, { "date" : ISODate("2021-01-12T08:00:00.000+08:00"), "total_vaccinations" : 6200 }, { "date" : ISODate("2021-01-13T08:00:00.000+08:00"), "total_vaccinations" : 0 }, { "date" : ISODate("2021-01-14T08:00:00.000+08:00"), "total_vaccinations" : 0 },</pre> <p>//This query returns the total number of vaccinations for each day in Singapore. Essentially, we observed that the values in total_vaccinations is accumulated over the days. This means that all subsequent records after the first date (2021-01-11) total_vaccinations is recorded, should be recording values that are either the same as the previous date, or larger. However, we observed that majority of the total_vaccinations data is recorded as '0'. This could indicate that the total_vaccinations data on these days were not being updated accordingly. Hence, this implies that this country_vaccinations collection may be missing information. As such, the resulting query based on this collection may not be accurate or reliable</p>
2	<pre>db.country_vaccinations.aggregate([{ \$match: { country: {</pre>

	<pre> \$in: ["Brunei", "Myanmar", "Cambodia", "Indonesia", "Laos", "Malaysia", "Philippines", "Singapore", "Thailand", "Vietnam"] } }, { \$project : { daily_vaccinations : {\$convert : {input : "\$daily_vaccinations", to : "int"}}, date: {\$convert : {input : "\$date", to : "date"}} } }, { \$group : {_id : {groupByDate : "\$date"}, sum : {\$sum: "\$daily_vaccinations"}} }, { \$sort: {"_id.groupByDate": 1} } }) </pre>  <pre> /* 1 */ { "_id" : { "groupByDate" : ISODate("2021-01-11T08:00:00.000+08:00") }, "sum" : 0 }, /* 2 */ { "_id" : { "groupByDate" : ISODate("2021-01-12T08:00:00.000+08:00") }, "sum" : 2800 }, /* 3 */ { "_id" : { "groupByDate" : ISODate("2021-01-13T08:00:00.000+08:00") }, "sum" : 17290 }, </pre>
3	<pre> db.country_vaccinations.aggregate([{ \$project : { daily_vaccinations_per_million : {\$convert : {input : "\$daily_vaccinations_per_million", to : "int"}}, country : "\$country" } }, { \$group : {_id : {groupByCountry : "\$country"}, max : {\$max : "\$daily_vaccinations_per_million"}} }, { \$sort : {max : -1} }] </pre>

	<pre> D) /* 1 */ { "_id" : { "groupByCountry" : "Bhutan" }, "max" : 118759 }, /* 2 */ { "_id" : { "groupByCountry" : "Falkland Islands" }, "max" : 54264 }, /* 3 */ { "_id" : { "groupByCountry" : "Cook Islands" }, "max" : 46231 }, </pre>
4	<pre> db.country_vaccinations_by_manufacturer.aggregate([{ \$project : { total_vaccinations : {\$convert : {input : "\$total_vaccinations", to : "int"}}, vaccine : "\$vaccine" } }, { \$group : {_id : {groupByVaccine : "\$vaccine"}, sum : {\$sum : "\$total_vaccinations"}} }, { \$sort : {sum : -1} }]) </pre>

	<pre> /* 1 */ { "_id" : { "groupByVaccine" : "Pfizer/BioNTech" }, "sum" : 26727111273 }, /* 2 */ { "_id" : { "groupByVaccine" : "Moderna" }, "sum" : 13770017464 }, /* 3 */ { "_id" : { "groupByVaccine" : "Oxford/AstraZeneca" }, "sum" : 2200684190 }, </pre> <p>//However, summing up total_vaccinations data from the country_vaccinations_by_manufacturer collection, as required by the question, will not produce the most accurate query. We observed that values in total_vaccinations for each vaccine type is accumulated over the days. This means that the value of total_vaccinations for a particular vaccine on a particular day will include the total_vaccinations from the previous day for the corresponding vaccine type, as well. As such, summing up the total_vaccinations do not accurately display the total administration per vaccine.</p> <p>To get a more accurate representation of the exact amount of the total administration per vaccine, the maximum value of the total_vaccinations per vaccine should be considered instead.</p>
5	<pre> db.country_vaccinations_by_manufacturer.aggregate([{ \$match : {location : "Italy"} }, { \$project : { date : {\$convert : {input : "\$date", to : "date"}}, vaccine : "\$vaccine" } }, { \$group : {_id : {groupByVaccine : "\$vaccine"}, date : {\$push : "\$date"}} }, { \$project : { vaccine : 1, minDate : {\$min : "\$date"}, } }, { \$group : { _id : 0, minDate : {\$min : "\$minDate"}, maxDate : {\$max : "\$minDate"} } }]) </pre>

	<pre> } } { \$project : { _id : 0, dayDiff : { \$dateDiff : { startDate : "\$minDate", endDate : "\$maxDate", unit : "day" } } } } D) /* 1 */ { "_id" : { "groupByVaccine" : "Pfizer/BioNTech" }, "sum" : 26727111273 }, /* 2 */ { "_id" : { "groupByVaccine" : "Moderna" }, "sum" : 13770017464 }, /* 3 */ { "_id" : { "groupByVaccine" : "Oxford/AstraZeneca" }, "sum" : 2200684190 }, </pre>
6	<pre> db.country_vaccinations_by_manufacturer.aggregate([{ \$project : { "location" : 1 , "vaccine" : 1 } }, { \$group : { _id : {groupByCountry : "\$location"}, vaccine : { \$addToSet : "\$vaccine" } } }, { \$unwind : "\$vaccine" }, { \$group : { _id : "\$_id", vaccineCount : { \$sum : 1 } } }, { \$sort : { vaccineCount :-1 } }] D) </pre>

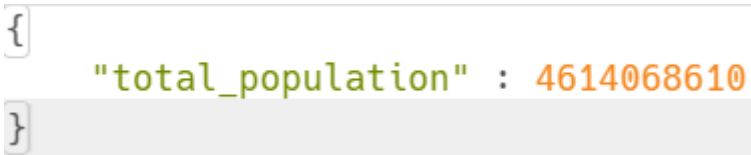
	<pre> /* 1 */ { "_id" : { "groupByCountry" : "Hungary" }, "vaccineCount" : 6 }, /* 2 */ { "_id" : { "groupByCountry" : "Slovakia" }, "vaccineCount" : 5 }, /* 3 */ { "_id" : { "groupByCountry" : "Bulgaria" }, "vaccineCount" : 4 }, </pre>
7	<pre> db.country_vaccinations_by_manufacturer.aggregate([{ \$match : {location : "United States"} }, { \$project : { date : { \$convert : {input : "\$date", to : "date"} }, total_vaccinations : { \$convert : {input : "\$total_vaccinations", to : "int"} }, vaccine: 1 } }, { \$project : { dateMonth : { \$month: "\$date" }, total_vaccinations : 1, vaccine: 1 } }, { \$group : { _id : {month : "\$dateMonth", vaccine: "\$vaccine"}, total_vaccinations: { \$max: "\$total_vaccinations" } } }, { \$sort : { _id : 1 } }]) </pre>

	<pre> /* 1 */ { "_id" : { "month" : 1, "vaccine" : "Moderna" }, "total_vaccinations" : 14246089 }, /* 2 */ { "_id" : { "month" : 1, "vaccine" : "Pfizer/BioNTech" }, "total_vaccinations" : 16775916 }, </pre>
8	<pre> db.country_vaccinations_by_manufacturer.aggregate([{ \$match : {location : "United States"} }, { \$project : { date : {\$convert : {input : "\$date", to : "date"}}, total_vaccinations : {\$convert : {input : "\$total_vaccinations", to : "int"}}, vaccine: 1 } }, { \$project : { dateMonth : {\$month: "\$date"}, total_vaccinations : 1, vaccine: 1 } }, { \$group : { _id : {month : "\$dateMonth", vaccine: "\$vaccine"}, total_vaccinations: {\$max: "\$total_vaccinations"} } }, { \$sort : {_id : 1} }]) </pre> <pre> /* 1 */ { "_id" : { "month" : 1, "vaccine" : "Moderna" }, "total_vaccinations" : 14246089 }, /* 2 */ { "_id" : { "month" : 1, "vaccine" : "Pfizer/BioNTech" }, "total_vaccinations" : 16775916 }, </pre>

9	<pre> db.firstAvailDate.drop() // Create a new collection containing first available dates for each country db.country_vaccinations.aggregate([{\$project: { total_vaccinations_per_hundred: {\$convert: {input: "\$total_vaccinations_per_hundred", to: "double"}}, country: 1, date: {\$convert: {input: "\$date", to: "date"}} }}, {\$group: { _id: {groupByCountry: "\$country"}, first_available_date: {\$min: "\$date"} }}, {\$sort: {"_id.groupByCountry": 1}}, {\$out: "firstAvailDate"}]) db.firstAvailDate.find() // Joining 2 collections to compute the number of days that each country takes to reach the 50 percent threshold db.country_vaccinations.aggregate([{\$project: { total_vaccinations_per_hundred: {\$convert: {input: "\$total_vaccinations_per_hundred", to: "double"}}, country: 1, date: {\$convert: {input: "\$date", to: "date"}} }}, {\$match: {"total_vaccinations_per_hundred": {\$gt: 50}}}, {\$group: { _id: {groupByCountry: "\$country"}, firstDateAbv50: {\$min: "\$date"} }}, {\$lookup: { from: "firstAvailDate", localField: "_id.groupByCountry", foreignField: "_id.groupByCountry", as: "firstAvailableDate" }}, { \$unwind: "\$firstAvailableDate" }, { \$project: { daysTo50Percent : {\$dateDiff : { startDate : "\$firstAvailableDate.first_available_date", endDate : "\$firstDateAbv50", unit : "day" } } } },]) </pre>
---	---

	<pre> { \$sort: {_id: 1} } D) /* 1 */ { "_id" : { "groupByCountry" : "Andorra" }, "daysTo50Percent" : NumberLong(133) }, /* 2 */ { "_id" : { "groupByCountry" : "Anguilla" }, "daysTo50Percent" : NumberLong(102) }, /* 3 */ { "_id" : { "groupByCountry" : "Antigua and Barbuda" }, "daysTo50Percent" : NumberLong(104) }, </pre>
10	<pre> db.country_vaccinations_by_manufacturer.aggregate([{\$project: { total_vaccinations: {\$convert: {input: "\$total_vaccinations", to: "int"}}, location: 1 vaccine: 1, }}, {\$group: { _id: {groupByLocation: "\$location", groupByVaccine: "\$vaccine"}, total_vaccinations: {\$max: "\$total_vaccinations"} }}, {\$group: { _id: {groupByVaccine: "\$_id.groupByVaccine"}, total_vaccinations: {\$sum: "\$total_vaccinations"} }}, {\$sort: {"total_vaccinations":1}}]) </pre>

	<pre> /* 1 */ { "_id" : { "groupByVaccine" : "CanSino" }, "total_vaccinations" : 391012 }, /* 2 */ { "_id" : { "groupByVaccine" : "Sputnik V" }, "total_vaccinations" : 1813128 }, /* 3 */ { "_id" : { "groupByVaccine" : "Sinopharm/Beijing" }, "total_vaccinations" : 2056651 }, </pre> <p>/* As mentioned in qn 4, the total_vaccinations data for each vaccine type from the country_vaccinations_by_manufacturer collection is accumulated over the days. Additionally, for this question, it is important to take note that total_vaccinations data for each vaccine is accumulated over the days, within each individual country. This means the total_vaccinations values are not accumulated across countries. Hence, finding the maximum total vaccinations for each vaccine type is equivalent to finding the total number of vaccinations for each vaccine type. Using those value, we can find the global total number of vaccinations across all countries, for each vaccine type. This is done by summing up the total_vaccinations value for each vaccine type for all countries. */</p>
11	<p>// Take note, location is used here instead of continent. Further explanation can be found below.</p> <pre> db.covid19data.aggregate([{\$match: {location: "Asia"}}, {\$project: {continent: 1, population: {\$convert: {input: "\$population", to: "double"}}}}, {\$group: { _id: {groupByLocation: "\$location"}, total_population: {\$last: "\$population"} }}]) </pre> <pre> { "_id" : { "groupByLocation" : null }, "total_population" : 4639847425 } </pre> <p>// ANS: The total population in Asia is 4,639,847,425.</p>

	<p>/* In the query below, continent was considered since continent also consists of countries in Asia. Resulting total population of Asia is smaller than above at 4,614,068,610, when filtering by continent instead. */</p> <pre>db.covid19data.aggregate([{\$match: {continent: "Asia"}}, {\$project: {continent: 1, population: {\$convert: {input: "\$population", to: "double"}}}}, {\$group: { _id: {groupByContinent: "\$continent"}, population: {\$addToSet: "\$population"} }}, {\$project: {_id: 0, total_population: {\$sum: "\$population"}}}])</pre>  <p>/* This is because, in this table, there are only 50 locations under the Asia continent. However, Asia should contain 51 locations. We discovered that North Korea is not included as a location under Asia. Data accuracy and integrity is compromised. Hence, it would be inaccurate to filter based on continents. */</p>
12	<pre>db.covid19data.aggregate([{\$match: {\$or: [{location: "Brunei"}, {location: "Myanmar"}, {location: "Cambodia"}, {location: "Indonesia"}, {location: "Laos"}, {location: "Malaysia"}, {location: "Philippines"}, {location: "Singapore"}, {location: "Thailand"}, {location: "Vietnam"}]}}, {\$project: { location: 1, population: {\$convert: {input: "\$population", to: "double"}} }}, {\$group: { _id: {groupByLocation: "\$location"}, population: {\$last: "\$population"} }}, {\$group: { _id: null, total_population: {\$sum: "\$population"} }}])</pre>

	<pre>{ "_id" : null, "total_population" : 667301412 }</pre>
13	<pre>db.country_vaccinations.distinct("source_name") /* 1 */ Africa Centres for Disease Control and Prevention, /* 2 */ COVID-19 Malta Public Health Response Team, /* 3 */ COVID-19 Vaccine Information Platform, /* 4 */ Cayman Islands Government, /* 5 */ Centers for Disease Control and Prevention, /* 6 */ Costa Rican Social Security Fund, /* 7 */ Directorate General for Health via Data Science for Social Good, /* 8 */ Directorate General of Health Services,</pre>
14	<pre>db.country_vaccinations.aggregate([{\$project: { daily_vaccinations: {\$convert: {input: "\$daily_vaccinations", to: "double"}}, daily_vaccinations_raw: {\$convert: {input: "\$daily_vaccinations_raw", to: "double"}}, country: 1, date: {\$convert: {input: "\$date", to: "date"}} }}, {\$match: {\$and:[{date: {\$gte: ISODate("2021-03-01")}}, {date: {\$lte: ISODate("2021-05-31")}}, {country: "Singapore"}] }}, {\$sort: {date: 1}}])</pre>

	<pre> /* 1 createdAt:7/4/2021, 9:48:43 AM*/ { "_id" : ObjectId("60e1137b1e59c1b4f5f2c3aa"), "country" : "Singapore", "daily_vaccinations" : 15004, "daily_vaccinations_raw" : 0, "date" : ISODate("2021-03-01T08:00:00.000+08:00") }, /* 2 createdAt:7/4/2021, 9:48:43 AM*/ { "_id" : ObjectId("60e1137b1e59c1b4f5f2c3ab"), "country" : "Singapore", "daily_vaccinations" : 14763, "daily_vaccinations_raw" : 0, "date" : ISODate("2021-03-02T08:00:00.000+08:00") }, /* 3 createdAt:7/4/2021, 9:48:43 AM*/ { "_id" : ObjectId("60e1137b1e59c1b4f5f2c3ac"), "country" : "Singapore", "daily_vaccinations" : 14523, "daily_vaccinations_raw" : 0, "date" : ISODate("2021-03-03T08:00:00.000+08:00") }, </pre>
15	<pre> db.country_vaccinations.aggregate([{\$project: {_id:0,country:1, total_vaccinations: {\$convert: {input: "\$total_vaccinations", to: "int"}}}, date: {\$convert: {input: "\$date", to: "date"}}}} {\$match: {country: "Singapore"}}, {\$match: {"total_vaccinations": {\$gt:0}}}, {\$group: {_id: {groupByCountry: "\$country"}, first_batch: {\$min: "\$date"}}}]) [{ "_id" : { "groupByCountry" : "Singapore" }, "first_batch" : ISODate("2021-01-11T08:00:00.000+08:00") }] </pre>
16	<pre> db.covid19data.aggregate([{\$match: {location: "Singapore"}}, {\$project: {location: 1, new_cases: {\$convert: {input: "\$new_cases", to: "double"}}}, date: {\$convert: {input: "\$date", to: "date"}}}}, {\$match: {date: {\$gte: ISODate('2021-01-11')}}}, {\$group: {_id: {groupByLocation: "\$location"}, totalNewCases: {\$sum: "\$new_cases"}}}]) </pre>

	<pre>{ "_id" : { "groupByLocation" : "Singapore" }, "totalNewCases" : 3710 }</pre>
17	<pre>db.covid19data.aggregate([{\$match: {location: "Singapore"}}, {\$project: {location: 1, new_cases: {\$convert: {input: "\$new_cases", to: "double"}}}, date: {\$convert: {input: "\$date", to: "date"}}}}, {\$match: {date: {\$lt: ISODate('2021-01-11')}}}, {\$group: {_id: {groupByLocation: "\$location"}, totalNewCases: {\$sum: "\$new_cases"}}}}])</pre> <pre>{ "_id" : { "groupByLocation" : "Singapore" }, "totalNewCases" : 58907 }</pre>
18	<pre>db.country_vaccinations_by_manufacturer.updateMany({}, [{ \$set: { "date": { \$toDate: "\$date" } } }]) // Joining covid19data collection with country_vaccinations_by_manufacturer collection db.covid19data.aggregate([{ \$match : {location : "Germany"} }, { \$project : { date : {\$convert : {input : "\$date", to : "date"}}, new_cases : {\$convert : {input : "\$new_cases", to : "decimal"}}, population : {\$convert : {input : "\$population", to : "decimal"}} } }, { \$sort : { "date" : 1 } }, { \$lookup:{ from:"country_vaccinations_by_manufacturer", localField : "date", foreignField : "date", as:"totalVac" } }, {</pre>

	<pre> \$unwind: "\$totalVac" }, { \$match: {"totalVac.location": "Germany"} }, { \$project : { _id: 0, "date" : 1, "percentageNewCases" : {\$divide : ["\$new_cases", "\$population"]}, totalVac: 1 } }, { \$project: { "totalVac.vaccine": 1, "totalVac.total_vaccinations": 1, date: 1, percentageNewCases:1 } }, { \$sort: {"date": 1} })) /* 1 */ { "date" : ISODate("2020-12-27T08:00:00.000+08:00"), "totalVac" : { "vaccine" : "Johnson&Johnson", "total_vaccinations" : "0" }, "percentageNewCases" : NumberDecimal("0.0001479877797589979798635645528507878") }, /* 2 */ { "date" : ISODate("2020-12-27T08:00:00.000+08:00"), "totalVac" : { "vaccine" : "Moderna", "total_vaccinations" : "2" }, "percentageNewCases" : NumberDecimal("0.0001479877797589979798635645528507878") }, </pre>
19	<pre> db.covid19data.aggregate([{ \$match : {"location" : "Germany"} }, { \$project : { date : {\$convert : {input : "\$date" , to : "date"}}, new_cases : {\$convert : {input : "\$new_cases", to : "double"}}, location : "\$location" } }, {\$project: {_id: 0, date: 1, "20DaysLater": {\$dateAdd: { startDate: "\$date", unit: "day", </pre>

	<pre> amount: 20 } }, "30DaysLater": { \$dateAdd: { startDate: "\$date", unit: "day", amount: 30 } }, "40DaysLater": { \$dateAdd: { startDate: "\$date", unit: "day", amount: 40 } } } }, { \$sort : { date : 1 } }, { \$lookup : { from : "country_vaccinations_by_manufacturer", localField: "20DaysLater", foreignField : "date" as : "20DaysLaterVacc" } }, { \$lookup : { from : "country_vaccinations_by_manufacturer", localField: "30DaysLater", foreignField : "date" as : "30DaysLaterVacc" } }, { \$lookup : { from : "country_vaccinations_by_manufacturer", localField: "40DaysLater", foreignField : "date" as : "40DaysLaterVacc" } }, { \$unwind: "\$20DaysLaterVacc" }, { \$unwind: "\$30DaysLaterVacc" }, { \$unwind: "\$40DaysLaterVacc" }, { \$match:{ </pre>
--	---

	<pre> db.country_vaccinations_by_manufacturer.aggregate([{ \$match : {location : "Germany"} }, { \$project : { date: 1, total_vaccinations : {\$convert : {input : "\$total_vaccinations", to : "double"}}, vaccine: 1 } }, { \$group :{ _id : {groupByDate : "\$date"} total_vaccinations : {\$sum : "\$total_vaccinations"} } }, { \$sort: {_id: 1} }, { \$project : { date : 1, total_vaccinations : 1, "21DaysLater" : {\$dateAdd : {startDate : "\$_id.groupByDate", unit : "day", amount : 21}}, "60DaysLater" : {\$dateAdd : {startDate : "\$_id.groupByDate", unit : "day", amount : 60}}, "120DaysLater" : {\$dateAdd : {startDate : "\$_id.groupByDate", unit : "day", amount : 120}} } }, { \$sort : {"_id.groupByDate" : 1} }, { \$lookup : { from : "covid19data", localField : "21DaysLater", foreignField : "date", as : "dailyCasesAfter21Days" } }, { \$lookup : { from : "covid19data", localField : "60DaysLater", foreignField : "date", as : "dailyCasesAfter60Days" } }, { \$lookup : { from : "covid19data", localField : "120DaysLater", </pre>
--	--

```

        foreignField : "date",
        as : "dailyCasesAfter120Days"
    }
},
{
    $unwind: "$dailyCasesAfter21Days"
},
{
    $unwind: "$dailyCasesAfter60Days"
},
{
    $unwind: "$dailyCasesAfter120Days"
},
{
    $match: {
        $and: [
            { "dailyCasesAfter21Days.location": "Germany" },
            { "dailyCasesAfter60Days.location": "Germany" },
            { "dailyCasesAfter120Days.location": "Germany" }
        ]
    }
},
{
    $project: {
        "_id.groupByDate": 1,
        total_vaccinations: 1,
        "dailyCasesAfter21Days.date": 1,
        "dailyCasesAfter21Days.new_cases": 1,
        "dailyCasesAfter60Days.date": 1,
        "dailyCasesAfter60Days.new_cases": 1,
        "dailyCasesAfter120Days.date": 1,
        "dailyCasesAfter120Days.new_cases": 1
    }
}
}
)

/* 1 */
{
    "_id" : {
        "groupByDate" : ISODate( "2020-12-27T08:00:00.000+08:00" )
    },
    "total_vaccinations" : 23321,
    "dailyCasesAfter21Days" : {
        "date" : ISODate( "2021-01-17T08:00:00.000+08:00" ),
        "new_cases" : "11484.0"
    },
    "dailyCasesAfter60Days" : {
        "date" : ISODate( "2021-02-25T08:00:00.000+08:00" ),
        "new_cases" : "11032.0"
    },
    "dailyCasesAfter120Days" : {
        "date" : ISODate( "2021-04-26T08:00:00.000+08:00" ),
        "new_cases" : "5961.0"
    }
},

```

9.3 List of Attributes for ERD

Country Vaccinations By Manufacturer

Others:

- people_vaccinated
- people_fully_vaccinated
- daily_vaccinations_raw
- total_vaccinations_per_hundred
- people_vaccinated_per_hundred
- people_fully_vaccinated_per_hundred
- daily_vaccinations_per_million
- source_name
- source_website

covid19data

Others:

1. people_vaccinated, people_fully_vaccinated, new_vaccinations, new_vaccinations_smoothed, total_vaccinations_per_hundred, people_vaccinated_per_hundred, people_fully_vaccinated_per_hundred, new_vaccinations_smoothed_per_million
2. new_deaths, new_deaths_smoothed, total_death_per_million, new_death_per_million, new_death_smoothed_per_million
3. population_density, median_age, aged_65_older, aged_70_older, cardiovasc_death_rate, diabetes_prevalence, female_smokers, male_smokers, handwashing_facilities, hospital_beds_per_thousand
4. new_tests, total_tests_per_thousand, new_tests_per_thousand, new_tests_smoothed, new_tests_smoothed_per_thousand, tests_units
5. new_cases, new_cases_smoothed, total_cases_per_million, new_cases_per_million, new_cases_smoothed_per_million, reproduction_rate
6. hosp_patients_per_million, weekly_hosp_admissions, weekly_hosp_admissions_per_million
7. icu_patients_per_million, weekly_icu_admissions, weekly_icu_admissions_per_million