



**NANYANG  
TECHNOLOGICAL  
UNIVERSITY**  
SINGAPORE

Nanyang Business School

**Nanyang Technological University  
Nanyang Business School**

**BC2402 – Designing and Developing Databases**  
Semester 1, 2021

**Group Project**

**COVID'19**

**The New Normal – Vaccinations and Re-opening**

**Querying between Relational and Non-Relational Databases**

**Members:**

**Alvin Lim Swee Hung (U2010149F)**

**Derick Siah Cheong Yong (U2010064F)**

**Ishita Goel (U1940458G)**

**Lim Qing Rui (U2010816G)**

**Melvin Ng Wee Kiat (U2010779D)**

## **Table of Contents**

<b>1. Introduction.....</b>	<b>3</b>
<b>2. Normalisation and Importing of Relational Database (SQL).....</b>	<b>4</b>
2.1. Tables Created .....	4
2.2. Keys (Primary Key and Foreign Key) .....	4
2.3. Creation and Insertion of Tables .....	5
2.4. Retaining and Dropping of Columns .....	6
<b>3. Data conversion and Importing of Non-Relational Database (NoSQL) .....</b>	<b>7</b>
3.1. Data import and Collection Creation .....	7
3.2. Data conversion in NoSQL .....	7
3.3. Column dropping and ERD .....	8
<b>4. MySQL Queries (Questions 1 to 10).....</b>	<b>9</b>
<b>5. MongoDB &amp; NoSQL Queries (Questions 1 to 20).....</b>	<b>20</b>
<b>6. Relational &amp; Non-Relational Data Model.....</b>	<b>40</b>
6.1. Function Difference Between Both Models .....	40
6.2. Data Storage and Data Model Requirements .....	40
<b>7. Recommendation to WHO.....</b>	<b>41</b>
7.1. Relational & Non-relational Database Implementation .....	41
7.2. ACID Compliance Between SQL and NoSQL Databases:.....	41
7.3. Skill Set Available: .....	42
7.4. Final Recommendation:.....	43
<b>8. References .....</b>	<b>44</b>
<b>9. Appendix .....</b>	<b>46</b>
Appendix A: Steps to import and normalize Relational Database.....	46
Appendix B: Entity Relationship Diagrams (ERD) for Relational Database...	48
Appendix C: Steps to import and convert data in Non-Relational Database.	49
Appendix D: Data conversion codes in Non-Relational Database .....	52

## **1. Introduction**

As the world continues its fight against the COVID-19 pandemic, information becomes a critical source of advantage that can help countries strategize and develop better plans to curb the spread of the virus. In order to fully utilize information, it is important to first have a suitable database system where information can be stored. To that end, this report aims to explore the database system that would be most suited for the World Health Organisation (WHO) by showcasing two databases, SQL and MongoDB, through a series of queries.

Based on the non-fictional dataset provided, containing information on covering COVID vaccinations to country information itself, the team analysed and conducted extensive investigations on the dataset and analysis from secondary information and research to achieve a deeper insight on the queries involved, detailing sources and reasons explaining possible results derived from the data provided. The dataset was first normalized and prepared using data conversion before querying was conducted. Columns that proved insightful and useful to the analysis were selected, while other columns were omitted where possible.

Recommendations provided will be kept aligned with WHO's needs and data requirements as well as presence of data scientists, availability of users and data structure of both databases used to evaluate the feasibility of databases in assisting WHO in creating pandemic reports, both current and in the future. The report will also include some insights on COVID-19 based on observations and trends that are identified through the queries.

## **2. Normalisation and Importing of Relational Database (SQL)**

There are a total of 3 datasets provided for the team to query: country\_vaccinations, country\_vaccinations\_by\_manufacturer, and covid19\_data. Each dataset contains different number of columns and information regarding COVID-19. However, the covid19\_data dataset contains many columns that are not closely related. Therefore, the team will be normalising the covid19\_data dataset. After normalisation, the team expects the data to be more enhanced and organised. Normalisation will also allow the team to group similar data together, linking each of them with a primary and foreign key. This overall makes each table easier to view and queries easier to compute while minimising redundancy.

For the remaining 2 datasets, the team has decided not to normalise the tables as those datasets do not contain enough columns for normalisation to have any significant impact.

### **2.1. Tables Created**

The covid19 dataset is massive, providing a variety of information about countries regarding covid-19 (like the number of cases, when the cases occur, vaccinations etc.) Therefore, to view the data in an easier manner and analyze it more effectively the team split the covid19data table into 8 different tables (namely: cases, deaths, hospital\_patients, hospital\_admissions, covid19\_test, vaccination, reproduction\_rate, country and country\_info).

The team split the tables based on information provided by the columns, grouping the columns providing information on the same topic to form a new table.

For example, the cases table contains (iso\_code, date, total\_cases, new\_cases, new\_cases\_smoothed, total\_cases\_per\_million, new\_cases\_per\_million, new\_cases\_smoothed\_per\_million) all of which provide information on the covid19 cases.

In total, the schema 'country\_vaccinations has a total of 11 tables (including country\_vaccinations and country\_vaccinations\_by\_manufacturer).

### **2.2. Keys (Primary Key and Foreign Key)**

For the country table, the team set iso\_code as a primary key because it can uniquely identify each row in the country table. However, for the cases table, iso\_code as a primary key on its own cannot uniquely identify the number of cases per day. Hence, on top of iso\_code, the date also must be declared as a primary key to identify the number of cases for each individual country per day. Similarly, this is applicable to other tables, namely,

1. deaths
2. hospital\_patients
3. hospital\_admissions
4. covid19\_test
5. vaccination
6. country\_info
7. reproduction\_rate

In addition, for all the tables except the country table, the team has set the iso\_code as the foreign key as its' values correspond to the values of the primary key in another table which is an existing row in the related table with the same primary key value.

### 2.3. Creation and Insertion of Tables

Based on the explanation provided in the sections above, the team then proceeded with the creation of the tables mentioned. Using the table named “cases” as an example, a step-by-step guide for table creation and value insertion is shown below.

```
create table `cases` (  
  `iso_code` varchar(15) not null,   
  `date` date not null,   
  `total_cases` text,   
  `new_cases` text,   
  `new_cases_smoothed` text,   
  `total_cases_per_million` text,   
  `new_cases_per_million` text,   
  `new_cases_smoothed_per_million` text,   
  primary key (`iso_code`, `date`),   
  foreign key (`iso_code`) references `country` (`iso_code`)  
);
```

Not null is specified for primary & foreign keys

Original data type is retained

Firstly, the team created an empty table with the column names derived from the main table in “covid19data”. The condition not null is also specified for the primary and foreign keys. For the data type in each column, the team has decided to retain the data type used in the original implementation that was provided, which is “text” given that the mathematical functions apply to the data type in this case.

```
insert into cases  
select iso_code, date, total_cases, new_cases, new_cases_smoothed,  
total_cases_per_million, new_cases_per_million, new_cases_smoothed_per_million  
from covid19data;
```

Next, data was inserted into the empty table with an insert statement which was created by selecting the corresponding columns from “covid19data” and copying the data over to the new table. With the “cases” table completed, the steps mentioned above was repeated for the other tables until the normalisation process was complete. Refer to **Appendix B** for the Team’s Entity Relationship Diagram after Normalisation.

## 2.4. Retaining and Dropping of Columns

After normalisation of the given data, team will assess which columns are deemed useful for the analysis. After careful assessment, the team will be dropping the below columns from table “country\_info”:

Table	Column
`country_info`	`stringency_index` `median_age` text, `aged_65_older` text, `aged_70_older` text, `gdp_per_capita` text, `extreme_poverty` text, `cardiovasc_death_rate` text, `diabetes_prevalence` text, `female_smokers` text, `male_smokers` text, `handwashing_facilities` text, `hospital_beds_per_thousand` text, `life_expectancy` text, `human_development_index` text, `excess_mortality` text,

The reason for dropping these columns is due to the type of lack of depth it provides in the analysis. The above columns have been identified to contain no direct link to the team’s COVID-19 analysis, the healthcare context in place, nor to the SQL questions provided. For example, the column `extreme\_poverty` has little impact on the COVID-19 analysis the team is conducting.

The columns that the team kept contain two types of information: columns that the team requires for answering the questions provided, and the columns that can potentially provide a more in-depth analysis into COVID-19. For example, the column `reproduction\_rate` is not used by the team to answer the questions provided. However, since the reproduction rates provided is related to COVID-19, the team felt that it could provide insightful information for future analysis.

**Note:** Refer to **Appendix A** for steps in importing data and execution of the team’s normalisation codes.

### 3. Data conversion and Importing of Non-Relational Database (NoSQL)

In NoSQL, normalisation was not required due to the presence of series of collections which stores the data, where each collection can have a different set of fields. Each field would contain certain values stored as records, where each record in the given table would comprise of a value for all its fields. Due to normalization not being required, Primary keys and Foreign keys were not defined in NoSQL. The process of denormalized data provides flexibility and performance due to reduced impedance mismatch, ease of variation for schema across various records and improved performance by inserting all related data simultaneously while removing joins across tables for data retrieval. (Krug P.,2019) The data stored within this project was stored in 3 collections: (i) 'country\_vaccinations', (ii) 'country\_vaccinations\_by\_manufacturer' and (iii) 'covid19data' in the database known as 'country\_vaccinations' (similar to SQL).

#### 3.1. Data import and Collection Creation

The data for the project was imported from 3 JSON files provided: (i) 'country\_vaccinations.json', (ii) 'country\_vaccinations\_by\_manufacturer.json' and (iii) 'covid19data\_2.json'. MongoDB was utilised to import the data and create the overall database 'country\_vaccinations', then NoSQL Booster was used to check that the data were imported for the variables using the following codes for each collection:

```
db.country_vaccinations.find()  
  
db.country_vaccinations_by_manufacturer.find()  
  
db.covid19data.find()
```

It is also essential to include the "Stop on errors" options when importing the datafile to prevent the importing of erratic data into the database. This allows the dropping of the collection if errors are found in the data given and the team can request for a new set of data or investigate the datafile given to determine erratic code structure. It is also essential to note that if the presence of a document is more than 16 megabytes, it will cause errors in the data implementation. Once the data was imported, the team ensured that the proper database was set as the default database using the code: "use country\_vaccinations". Refer to **Appendix C** for steps in importing the data depicted by images.

#### 3.2. Data conversion in NoSQL

The presence of the original data values as a string decreased efficiency of data retrieval and manipulation in the queries. To overcome the hurdle, the team converted the data using the aggregate function to data type 'double'. This allowed the team to perform data manipulation and query investigation without the need to convert the data within the code using conversion formats such as \$toDouble.



Based on the above image, the team converted the `total_vaccinations` column in the collection `'country_vaccinations_by_manufacturer'` to a double data type, which is an 8 byte (64-bit IEEE 754 floating point) numerical data. Location and Vaccine columns will remain as a string format, while any form of date type will be converted to a date format MM/DD/YYYY with a recorded time of 8:00:00 AM. The aggregate function selects all columns listed within to convert it into numerical and date data for us to query on. The output is then parsed back into its respective collection, in this case it is sent back to `'country_vaccinations_by_manufacturer'` collection.

Refer to **Appendix D** for the conversion of key columns in collections `'country_vaccinations'` and `'covid19data'` that the team will be using in the query.

An example of data converted before and after can be seen below:

#### Before

Field	Value	Data Type
<code>_id</code>	60e113961e59c1b4f5f2d757	ObjectId
<code>location</code>	Austria	String
<code>date</code>	2021-01-08	String
<code>vaccine</code>	Johnson&Johnson	String
<code>total_vaccinations</code>	0	String

Annotation: "Data type before conversion" points to the type column.

#### After

Field	Value	Data Type
<code>_id</code>	60e113961e59c1b4f5f2d757	ObjectId
<code>location</code>	Austria	String
<code>vaccine</code>	Johnson&Johnson	String
<code>date</code>	1/8/2021, 8:00:00 AM	Date
<code>total_vaccinations</code>	0	Double

Annotation: "Data type after conversion" points to the type column.

### 3.3. Column dropping and ERD

Due to the nature of NoSQL, there was no ERD performed for the non-relational database. Unused columns were not dropped to preserve data integrity and minimise complication in the database as each document fit the maximum requirement of 16 megabytes. Other noticeable traits in the NoSQL database were that `'covid19data'` collection had a huge, sizeable amount of data, amounting to 0.1M, while `'country_vaccinations_by_manufacturer'` and `'country_vaccinations'` were 8.3K and 28.2K each. This can be attributed to the number of fields/columns in each collection, as well as the type of data present in each record.



#### 4. MySQL Queries (Questions 1 to 10)

Q1) What is the total population in Asia?

	continent	Total_population_in_Asia
▶	Asia	4614068610

```
select continent, sum(p) as 'Total_population_in_Asia' from
(select country, continent, max(population) as p from country_info, country
where country_info.iso_code = country.iso_code and
continent = "Asia"
group by country) as a;
```

As the question requested for the total population in Asia, the team summed up the population for each country in Asia from the sub query. Within the sub query, the team must first set 2 conditions. For the first condition, the team identified that the population is an accumulated figure. Thus, the team used the MAX function to retrieve the MAX value which will return the highest value of the total population for the individual country alongside with the country and continent referencing the “country” and “country\_info” table. The second condition is to set that it will only display records if the iso\_code for both tables match, filtering out records for which its continent belongs to Asia and grouping them by its country.

Q2) What is the total population among the ten ASEAN countries?

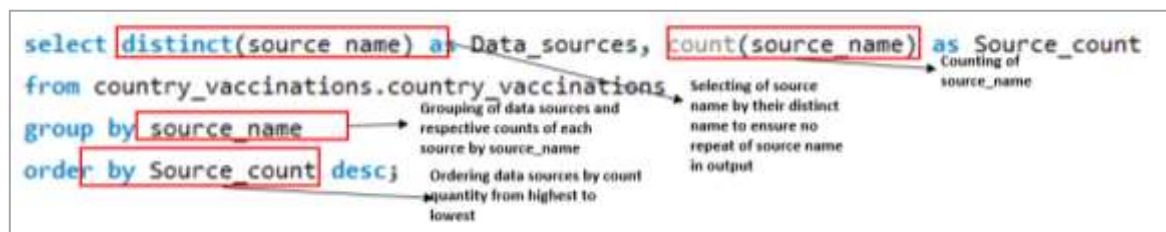
	continent	Total_population_in_ASEAN
▶	Asia	667301412

```
select continent, sum(p) as Total_population_in_ASEAN from
(select country, continent, max(population) as p from country_info, country
where country_info.iso_code = country.iso_code and
country in ("Singapore", "Malaysia", "Brunei", "Cambodia", "Indonesia",
"Laos", "Myanmar", "Philippines", "Thailand", "Vietnam")
group by country) as a;
```

Now the question is asking for the total population in ASEAN countries. Hence, the approach is similar to question 1. Instead of filtering out records for which its continent belongs to Asia, the team now filter the records for which its country is in the list of ASEAN countries.

### Q3) Generate a list of unique data sources (source\_name).

**Approach:** The question requires the team to obtain a list of unique data sources from which the vaccinations/country/iso\_code data were taken from. The first step to extract the data would be to obtain the source\_name without any repeated source names. Hence, the team used the DISTINCT() function in the SELECT query statement to select only unique source names. As the column 'source\_name' was not included in any Normalized Tables, the team extracts the data from the original table 'country\_vaccinations' as only the country\_vaccination table has information on the data sources. The data would then be presented in alphabetical order from A to Z for visual purposes, hence the team used ORDER BY source\_name ASC to order it in ascending manner to show the results of Data\_sources in alphabetical order.



**Output:** The output generated displays the Data Sources starting with Ministry of Health at the top of the list with the highest source count quantity of 7074 and lowest count quantity of 67 from Government of the Falkland Islands.

Data_sources	Source_count
Ministry of Health	7074
World Health Organization	4715
Government of the United Kingdom	1010
SPC Public Health Division	897
Pan American Health Organization	650
Africa Centres for Disease Control and Prevention	459
Federal Office of Public Health	343
Ministry of Public Health	327
Norwegian Institute of Public Health	211
Official data from provinces via covid19tracker.ca	199
National Health Commission	198
Official data from local governments via gogov.ru	198

First Source\_name displayed by highest count with respective source count of 7074

Data_sources	Source_count
Government of Jersey	102
Taiwan Centers for Disease Control	102
Ministry of Health and Social Services	101
Government of Aruba	94
Government of Curacao	94
Government of Saint Helena	92
National Health Security Agency	92
Government of Uzbekistan	91
Government of Eswatini	89
Ministry of Health and Wellness	89
Government of Zambia	77
Government of the Falkland Islands	67

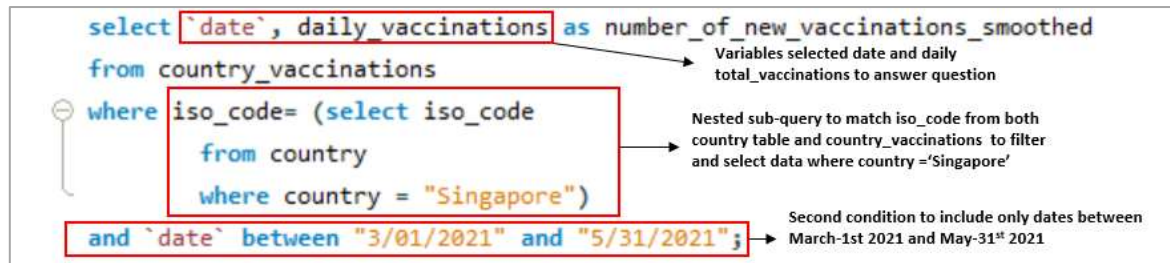
Last Source\_name displayed by lowest count with respective source count of 67

### Q4) Specific to Singapore, display the daily total\_vaccinations starting (inclusive) March-1 2021 through (inclusive) May-31 2021.

**Approach:** This question requires us to find the daily total vaccinations for the Country 'Singapore', displaying the input dates starting from March-1 2021 to May-31 2021 inclusive of both dates as the start and ending dates. For this question, the team used the `date` and the daily\_vaccinations as the number of new vaccinations as it is a smoothed data and can display the trend over the period chosen (3 months, March to May).

The team matched the iso\_code in the nested sub-query "SELECT iso\_code" statement to match the country name of 'Singapore' with the normalized 'country' table. This allowed greater matching accuracy between the country and country\_vaccination table to extract Singapore's iso\_code. This was done in the WHERE statement to specify the condition that the country must be from Singapore. Lastly, the BETWEEN

function was used with the AND function to include the specified input dates between March-01 2021 to May-31 2021 inclusive using the America date format of MM/DD/YYYY. The output returned both the daily dates and an increasing number of vaccinations smoothed data per day from the start to end date, showing a positive uptrend vaccinations in Singapore.



**Output:** The output generated shows the start date of March 1<sup>st</sup> 2021 and end date of May 31<sup>st</sup> 2021 in Singapore with the smoothed daily vaccination data that can be used for further analyses to plot trendlines/linear regressions/exponentials if necessary. Interpreting the results, it is observed that daily vaccination rates have tripled to 45,540 on 31 May 2021 when compared to the vaccination rate on 3<sup>rd</sup> January 2021. The reason for the increase may be due to the stabilisation of vaccine production worldwide, making vaccine supplies more accessible. The increase of vaccines recognised (e.g Sinovac & Moderna) by the Singapore government coupled with their vaccination campaign may have also contributed to the increase in vaccination numbers.

date	number_of_new_vaccinations_smoothed
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
3/9/2021	14118.0000
3/10/2021	15911.0000
3/11/2021	17705.0000
3/12/2021	19925.0000
3/13/2021	22144.0000
3/14/2021	24364.0000
3/15/2021	25873.0000

Start date of March 1 2021

date	number_of_new_vaccinations_smoothed
5/22/2021	47627.0000
5/23/2021	46799.0000
5/24/2021	45972.0000
5/25/2021	45910.0000
5/26/2021	45848.0000
5/27/2021	45787.0000
5/28/2021	45725.0000
5/29/2021	45664.0000
5/30/2021	45602.0000
5/31/2021	45540.0000

End date of May 31 2021

### Q5) When is the first batch of vaccinations recorded in Singapore?

**Approach:** The question required the team to find the date when first batch of vaccination was recorded in Singapore. The team used the table 'country\_vaccinations' to find the first vaccination date ('First\_Vaccination\_Date'). The team selected the columns 'country', 'date' as 'First\_Vaccination\_Date' and the total\_vaccinations on that day. The team used min(str\_to\_date(date, "%m/%d/%Y")), where the inner function of str\_to\_date() converts the text format in which the date is stored to the date format (in the YYYY-MM-DD format). The min() function finds the first (or minimum) date for the given conditions, that is, where the country is Singapore, and the total vaccinations is greater than 0.

```
select country,min(str_to_date(date, "%m/%d/%Y")) as date,total_vaccinations
from country_vaccinations
where total_vaccinations > 0 and country = 'Singapore';
```

**Output:** Running the query, the team found the first vaccination date to be 11 January 2021.

	country	First_Vaccination_Date	total_vaccinations
▶	Singapore	2021-01-11	3400.0000

### Q6) Based on the date identified in (5), specific to Singapore, compute the total number of new cases thereafter.

**Approach:** In this question the team made use of the date of the first vaccination obtained in **Question 5**, that is '2021-01-11' (11 January 2021) to find the number of cases after this date (inclusive).

For this question the team made use of a subquery, to find the **iso\_code** for Singapore. For the subquery the team made use of the 'country' table and selected the **iso\_code** where the country is Singapore.

For the main query the team used the 'cases' table. For the number of cases, the team made use of the 'new\_cases' column. The raw data for the number of new cases was used instead of the smoothed data since no trend analysis of the data was required. In this case the raw data may be more accurate when it comes to reflecting the total sum of new cases. Finally, the team used the sum() function to find the total number of cases for Singapore where the date is greater than and equal to '2021-01-11'.

```
select sum(new_cases) as total_number_of_cases
from cases
where iso_code in
(select iso_code
from country
where country = 'Singapore') and date >= '2021-01-11';
```

**Output:** Comparing the result of 3710 new cases, the team observed a dip in the number of new cases when compared to the total number of new cases before the vaccination drive started (see Q7 below). However, this may be due to the tightened covid-19 measures and the “circuit breaker” that was implemented. Impacts on the vaccination drive in relation to new cases may need to be further analyzed to determine whether causation exists.

	total_number_of_cases
▶	3710

**Q7) Compute the total number of new cases in Singapore before the date identified in (5).**

**Approach:** To successfully calculate the total number of new cases in Singapore before the first vaccination date, the team made use of the date identified in question 5, 11 January 2021 (2021-01-11) to resolve this query.

<pre> select sum(new_cases) as Total_number_of_cases from cases where iso_code in   (select iso_code    from country    where country = 'Singapore') and 'date' &lt;= '2021-01-10'; </pre>	
	Sub-query for identifying Singapore's iso code from "country" table
	Setting date condition to include only rows before "2021-01-11" in calculation

As the question requested for the total number of new cases before 11 January 2021, the team summed up the daily number of new cases from the very first record that can be found in the “cases” table for Singapore up till the date of 10 January 2021. In this instance, raw data for daily new cases was used instead of smoothed data as no trend analysis was required in this question. Also, in resolving the query, two conditions were set. Firstly, a sub-query was used to identify the iso code for Singapore by referencing the “country” table to ensure that only records with the iso code for Singapore was used in the calculation. Secondly, the calculation was set to include only data before 11 January 2021 by setting the date to be less than or equal to “2021-01-10”.

**Output:** Running the query, the team obtained a result of 58,907 new cases before the start of the vaccination drive. Based on secondary research, the high number of cases was due to the covid-19 outbreak that occurred in the foreign workers dormitory as well as the gradual rise of local transmission over the course of 2020.

	Total_number_of_cases
▶	58907



**Q8) 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.**

**Approach:** Based on the team's interpretation of the question, there were 2 parts that needed to be resolved in order to estimate Germany's herd immunity progress. The first part consists of calculating the percentage of new cases while the second part consists of calculating the percentage of the total vaccinations based on the various vaccines available.

	iso_code	date	(new_cases_smoothed/population*100)	population
	DEU	2020-02-02	0.0000017055773633003316	83783945.0
	DEU	2020-02-03	0.00000187506090814893	83783945.0
	DEU	2020-02-04	0.0000013642231814221688	83783945.0
	DEU	2020-02-05	0.0000013642231814221688	83783945.0
	DEU	2020-02-06	0.0000013642231814221688	83783945.0
	DEU	2020-02-07	0.0000013642231814221688	83783945.0
	DEU	2020-02-08	0.0000008521919086049242	83783945.0
	DEU	2020-02-09	0.0000006815148176658428	83783945.0
	DEU	2020-02-10	0.00000034135418187816294	83783945.0

To resolve the first part of the query, the team first created a view named "new\_cases\_percentage", as shown above, that reflected the date, Germany's percentage of new cases and the country's population count on a daily basis. Information from the view was obtained by joining the tables named "country.info" and "cases" based on the date and iso\_code of the data. Upon filtering the data to obtain information based on the condition that country\_info.iso\_code = "DEU", the team proceeded to calculate the percentage of new cases by taking the column "new\_cases\_smoothed" and dividing it by the population number found in "country.info". Smoothed data was used in this case as the team believes that it would be a better representation for trend analysis when looking at the percentage of new cases by date. Raw data may not be as suitable due to spikes or dips that can be reflected during trend analysis if data is not collected at regular intervals.

<pre> select new_cases_percentage.`date`, new_cases_percentage.`(new_cases_smoothed/population*100)` as Percentage_of_new_cases, Vaccine, (total_vaccinations/population*100) as Percentage_of_total_vaccination from new_cases_percentage, country_vaccinations_by_manufacturer where country_vaccinations_by_manufacturer.`date` = new_cases_percentage.`date` and country_vaccinations_by_manufacturer.location = "Germany" group by new_cases_percentage.`date`, vaccine order by new_cases_percentage.`date`, vaccine; </pre>		Condition for joining the view with the table
	country_vaccinations_by_manufacturer.location = "Germany"	Filtering the data and selecting only those where location = "Germany"
	group by new_cases_percentage.`date`, vaccine order by new_cases_percentage.`date`, vaccine;	Grouping and ordering the data by date and vaccine for trend analysis on herd immunity

Next, referencing the view created, the team proceeded to join the view with the table named "country\_vaccinations\_by\_manufacturer" based on the condition that the rows

have the same date. The data is also filtered by specifying the location in the aforementioned table to be Germany. Finally, using the information found in both tables, the team calculated the total vaccinations on each available vaccine in relation to its population by taking the total\_vaccinations divided by population. The information is then grouped and ordered by the date and the type of vaccine available in Germany. The final output displayed contains the date, percentage of new cases, vaccine and total vaccination percentage that were selected from the combined table.

**Output:** Based on the output generated, the team observed an increase in the percentage of new cases relative to population in the early stages of the pandemic. However, as the percentage of total vaccination relative to the population started to rise, the team observe a declining trend in percentage of new cases relative to population. Germany is also relatively well on track to achieving herd immunity given that the total vaccination administered has risen considerably since the start of 2021 and is now above 87% (for all vaccine types combined), relative to their population. However, for better herd immunity estimation, data relating to the number of fully vaccinated individuals should also be used to supplement the analysis given that the high vaccination rate may be caused by individuals taking multiple doses of vaccines (2 shots or more).

	date	Percentage_of_new_cases	Vaccine	Percentage_of_total_vaccination
	2021-06-29	0.000698224462932606	Johnson&Johnson	2.280321128349829
	2021-06-29	0.000698224462932606	Moderna	7.616813698614931
	2021-06-29	0.000698224462932606	Oxford/AstraZeneca	13.649917057498307
	2021-06-29	0.000698224462932606	Pfizer/BioNTech	64.65000663313239
	2021-06-30	0.0006421277966799009	Johnson&Johnson	2.305519273412108
	2021-06-30	0.0006421277966799009	Moderna	7.7234988159127616
	2021-06-30	0.0006421277966799009	Oxford/AstraZeneca	13.809513266533344
	2021-06-30	0.0006421277966799009	Pfizer/BioNTech	65.52405714483844

↓

Last date of expected output

↓

Total vaccination percentage in relation to population grouped by vaccine type.

Finally, it is also observed that information relating to new cases and total vaccinations ends on 6 June 2021. This is because information relevant to total vaccinations based on the vaccine type ends on 6 June 2021. As such, the team determined that there was little need to display the output after 6 June 2021 given that the information on percentage of new cases would unlikely be helpful on its own for the purpose of herd immunity estimation, given the missing information on total vaccination.

**Q9) 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.**

**Approach:** There are two parts to this question. Firstly, the question is requesting the team to retrieve daily the number of new cases in Germany. Then corresponding to this date, find the total vaccinations for each vaccine available in Germany after 20, 30, and 40 days respectively.

date1	new_cases_smoothed	vaccine	date2	total_vaccinations1	date3	total_vaccinations2	date4	total_vaccinations3
2020-12-07	18584.857	Johnson&Johnson	2020-12-27	0	2021-01-06	0	2021-01-16	0
2020-12-07	18584.857	Moderna	2020-12-27	2	2021-01-06	16	2021-01-16	13341
2020-12-07	18584.857	Oxford/AstraZeneca	2020-12-27	0	2021-01-06	5	2021-01-16	14
2020-12-07	18584.857	Pfizer/BioNTech	2020-12-27	23319	2021-01-06	459272	2021-01-16	1144835
2020-12-08	19227.286	Johnson&Johnson	2020-12-28	0	2021-01-07	0	2021-01-17	0
2020-12-08	19227.286	Moderna	2020-12-28	2	2021-01-07	19	2021-01-17	14602
2020-12-08	19227.286	Oxford/AstraZeneca	2020-12-28	0	2021-01-07	6	2021-01-17	15
2020-12-08	19227.286	Pfizer/BioNTech	2020-12-28	41137	2021-01-07	513416	2021-01-17	1191952
2020-12-09	19486.429	Johnson&Johnson	2020-12-29	0	2021-01-08	0	2021-01-18	0
2020-12-09	19486.429	Moderna	2020-12-29	2	2021-01-08	68	2021-01-18	18756

The solution for this query requires creating dummy tables to extract the columns from. As the question is requesting for three dates that derives from the same table, the query would require a total of three separate tables to retrieve these dates. Firstly, the team selected the column names that they would like to retrieve, inclusive of the three date columns. Then, the team referenced each date to the *country\_vaccinations\_by\_manufacturer* table while renaming each of the table (C1, C2, C3) to create two more “dummy” tables to retrieve the second and third date. The team then selected the *total\_vaccinations* from each table which will correspond to the dates retrieved accordingly.

```
SELECT C1.location, cases.`date` date1, new_cases_smoothed, C1.vaccine,
C1.`date` date2, C1.total_vaccinations total_vaccinations1,
C2.`date` date3, C2.total_vaccinations total_vaccinations2,
C3.`date` date4, C3.total_vaccinations total_vaccinations3
FROM cases,
country_vaccinations_by_manufacturer C1,
country_vaccinations_by_manufacturer C2,
country_vaccinations_by_manufacturer C3
```

Next, the team specified the conditions for the other dates using the *WHERE* function along with *date\_add*: *C1.date = cases.date + 20 days*, *C2.date = C1.date + 10 days*, and *C3.date = C1.date + 20 days*. This ensures that each date retrieved is correctly matched.

```
where C2.`date` = date_add(C1.`date`, interval 10 DAY)
and C3.`date` = date_add(C1.`date`, interval 20 DAY)
and C1.`date` = date_add(cases.`date`, interval 20 DAY)
```

Finally, the team specified the remaining conditions to ensure that the results correspond to each other. The team did this by specifying that the vaccines from all three tables must match, and the location selected must be “Germany”.



```

AND C1.vaccine = C2.vaccine
AND C1.vaccine = C3.vaccine
AND C1.location = "Germany"
AND C2.location = "Germany"
AND C3.location = "Germany"
AND cases.iso_code = (SELECT iso_code
                        FROM country
                        WHERE country = "Germany")

```

**Output:** From the results generated, it can be observed that the daily new cases in Germany is a vaccination driver for Germans to get vaccinated. As the daily new cases increases, the total number of vaccinations has also increased. While this can easily be inferred that the Germans would like to mitigate their risk of contracting the virus and facing serious symptoms, it cannot be assumed that there are no underlying factors affecting the total vaccination rates. Thus, the number of daily new cases might not be the only vaccination driver in driving the total vaccination rates in Germany. Based on the results, it can also be seen that Germany's vaccination rate is on a good track with more than half of its population being vaccinated as of 30<sup>th</sup> June 2021. Information regarding *new\_cases\_smoothed* were also observed to stop after 21<sup>st</sup> May 2021. This is because there were no more *total\_vaccinations* data after 30<sup>th</sup> June 2021, 40 days after 21<sup>st</sup> May 2021. As a result, the team decided that information relating to *new\_cases\_smoothed* after 21<sup>st</sup> May 2021 was not required as there is insufficient data to determine, analyse and conclude the vaccination drivers after the previously mentioned date.

date1	new_cases_smoothed	vaccine	date2	total_vaccinations1	date3	total_vaccinations2	date4	total_vaccinations3
2021-05-19	9040.429	Moderna	2021-06-08	4567113	2021-06-18	5472927	2021-06-28	6291280
2021-05-19	9040.429	Oxford/AstraZeneca	2021-06-08	9774463	2021-06-18	10461183	2021-06-28	11304970
2021-05-19	9040.429	Pfizer/BioNTech	2021-06-08	41374400	2021-06-18	48527634	2021-06-28	53504675
2021-05-20	8625.571	Johnson&Johnson	2021-06-09	1076126	2021-06-19	1657745	2021-06-29	1910543
2021-05-20	8625.571	Moderna	2021-06-09	4653601	2021-06-19	5554857	2021-06-29	6381667
2021-05-20	8625.571	Oxford/AstraZeneca	2021-06-09	9838082	2021-06-19	10511836	2021-06-29	11436439
2021-05-20	8625.571	Pfizer/BioNTech	2021-06-09	42526844	2021-06-19	48794591	2021-06-29	54166326
2021-05-21	8299.714	Johnson&Johnson	2021-06-10	1164493	2021-06-20	1670525	2021-06-30	1931655
2021-05-21	8299.714	Moderna	2021-06-10	4742516	2021-06-20	5626102	2021-06-30	6471052
2021-05-21	8299.714	Oxford/AstraZeneca	2021-06-10	9893582	2021-06-20	10544321	2021-06-30	11570155
2021-05-21	8299.714	Pfizer/BioNTech	2021-06-10	43458493	2021-06-20	48980073	2021-06-30	54898640

**Q10) 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.**

**Approach:** Similar to Q9, there are two parts to this question. Firstly, the question is requesting the team to retrieve daily total number of accumulated vaccinations per vaccine in Germany. Then corresponding to the dates, find the number of new cases after 21, 60, and 120 days respectively.

date1	vaccine	sum_vaccinations	date2	new_cases_smoothed	date3	new_cases_smoothed	date4	new_cases_smoothed
2020-12-27	Johnson&Johnson	0	2021-01-17	17245.571	2021-02-25	7837.143	2021-04-26	20788.0
2020-12-27	Moderna	2	2021-01-17	17245.571	2021-02-25	7837.143	2021-04-26	20788.0
2020-12-27	Oxford/AstraZeneca	0	2021-01-17	17245.571	2021-02-25	7837.143	2021-04-26	20788.0
2020-12-27	Pfizer/BioNTech	23319	2021-01-17	17245.571	2021-02-25	7837.143	2021-04-26	20788.0
2020-12-28	Johnson&Johnson	0	2021-01-18	16895.143	2021-02-26	7892.429	2021-04-27	20004.286
2020-12-28	Moderna	2	2021-01-18	16895.143	2021-02-26	7892.429	2021-04-27	20004.286
2020-12-28	Oxford/AstraZeneca	0	2021-01-18	16895.143	2021-02-26	7892.429	2021-04-27	20004.286
2020-12-28	Pfizer/BioNTech	41137	2021-01-18	16895.143	2021-02-26	7892.429	2021-04-27	20004.286

The solution for this query is similar to Q9. Since the question is requesting for three dates from the same table, specifically 21 days, 60 days, and 120 days after each daily total accumulated vaccinations, the solution would also require the team to create three separate tables to retrieve the dates from. Firstly, the team retrieved the daily total number of accumulated vaccinations based on vaccines. The total number of accumulated vaccinations has been renamed to *sum\_vaccinations*. Then, the team specified the column names to retrieve, specifically the three dates, and the new cases that corresponds to each of the date. Next, the creation of the dummy tables (*C1*, *C2*, *C3*) is required to extract three different dates from the same table. Thereafter, the number of new cases that corresponds to each of the dates will also be retrieved from the same tables as the date.

```
SELECT cm.location, cm.`date` date1, vaccine, sum(total_vaccinations) sum_vaccinations,
C1.`date` date2, C1.new_cases_smoothed,
C2.`date` date3, C2.new_cases_smoothed,
C3.`date` date4, C3.new_cases_smoothed
FROM country_vaccinations_by_manufacturer cm,
cases C1,
cases C2,
cases C3
```

Thirdly, the team specified the conditions for each of the dates to match using the *WHERE* and *date\_add* functions: *C1.date = cm.date + 21 days*, *C2.date = cm.date + 60 days*, and *C3.date = cm.date + 120 days*.

```
WHERE C1.`date` = date_add(cm.`date`,INTERVAL 21 DAY)
AND C2.`date` = date_add(cm.`date`,INTERVAL 60 DAY)
AND C3.`date` = date_add(cm.`date`,INTERVAL 120 DAY)
```

Fourthly, the team specified the condition such that every column and rows retrieved must only contain information from "Germany". Finally, a groupby using the original date and vaccine was used to retrieve all related information after summing the total vaccinations.

```

AND C1.iso_code = (SELECT iso_code
                    FROM country
                    WHERE country = "Germany")
AND C2.iso_code = (SELECT iso_code
                    FROM country
                    WHERE country = "Germany")
AND C3.iso_code = (SELECT iso_code
                    FROM country
                    WHERE country = "Germany")
AND location = "Germany"
GROUP by date1, vaccine

```

**Output:** Based on the results, it can be observed that to a certain extent, the vaccination does play a part in reducing the number of new cases. From *date2* (+21 days) and *date3* (+60 days), the number of new cases detected has not decreased. Instead, it followed an upward trend from thousands of cases to ten thousand cases and more. However, the number of new cases detected from *date4* (+120 days) showed that the vaccination has helped reduced the number of cases as the new cases followed a downward trend from thousands to hundreds. Thus, this could be inferred that the vaccination plays a part in lowering the number of new cases in the long run, but it is not as effective in doing so in the short run.

date1	vaccine	sum_vaccinations	date2	new_cases_smoothed	date3	new_cases_smoothed	date4	new_cases_smoothed
2020-12-27	Johnson&Johnson	0	2021-01-17	17245.571	2021-02-25	7837.143	2021-04-26	20788.0
2020-12-27	Moderna	2	2021-01-17	17245.571	2021-02-25	7837.143	2021-04-26	20788.0
2020-12-27	Oxford/AstraZeneca	0	2021-01-17	17245.571	2021-02-25	7837.143	2021-04-26	20788.0
2020-12-27	Pfizer/BioNTech	23319	2021-01-17	17245.571	2021-02-25	7837.143	2021-04-26	20788.0
2020-12-28	Johnson&Johnson	0	2021-01-18	16895.143	2021-02-26	7892.429	2021-04-27	20004.286
2020-12-28	Moderna	2	2021-01-18	16895.143	2021-02-26	7892.429	2021-04-27	20004.286
2020-12-28	Oxford/AstraZeneca	0	2021-01-18	16895.143	2021-02-26	7892.429	2021-04-27	20004.286
2020-12-28	Pfizer/BioNTech	41137	2021-01-18	16895.143	2021-02-26	7892.429	2021-04-27	20004.286
2020-12-29	Johnson&Johnson	0	2021-01-19	14755.571	2021-02-27	7965.143	2021-04-28	20562.714
2020-12-29	Moderna	2	2021-01-19	14755.571	2021-02-27	7965.143	2021-04-28	20562.714
2020-12-29	Oxford/AstraZeneca	0	2021-01-19	14755.571	2021-02-27	7965.143	2021-04-28	20562.714
2020-12-29	Pfizer/BioNTech	90900	2021-01-19	14755.571	2021-02-27	7965.143	2021-04-28	20562.714
2020-12-30	Johnson&Johnson	0	2021-01-20	15246.571	2021-02-28	7968.571	2021-04-29	19490.0
2020-12-30	Moderna	2	2021-01-20	15246.571	2021-02-28	7968.571	2021-04-29	19490.0
2020-12-30	Oxford/AstraZeneca	0	2021-01-20	15246.571	2021-02-28	7968.571	2021-04-29	19490.0
2020-12-30	Pfizer/BioNTech	152685	2021-01-20	15246.571	2021-02-28	7968.571	2021-04-29	19490.0
2020-12-31	Johnson&Johnson	0	2021-01-21	13380.0	2021-03-01	8010.0	2021-04-30	18356.286
2020-12-31	Moderna	2	2021-01-21	13380.0	2021-03-01	8010.0	2021-04-30	18356.286

date1	vaccine	sum_vaccinations	date2	new_cases_smoothed	date3	new_cases_smoothed	date4	new_cases_smoothed
2021-03-03	Pfizer/BioNTech	6017401	2021-03-24	16031.286	2021-05-02	17997.714	2021-07-01	577.857
2021-03-04	Johnson&Johnson	19	2021-03-25	15050.0	2021-05-03	17933.286	2021-07-02	566.429
2021-03-04	Moderna	242620	2021-03-25	15050.0	2021-05-03	17933.286	2021-07-02	566.429
2021-03-04	Oxford/AstraZeneca	780600	2021-03-25	15050.0	2021-05-03	17933.286	2021-07-02	566.429
2021-03-04	Pfizer/BioNTech	6158278	2021-03-25	15050.0	2021-05-03	17933.286	2021-07-02	566.429
2021-03-05	Johnson&Johnson	20	2021-03-26	15794.714	2021-05-04	17676.143	2021-07-03	558.0
2021-03-05	Moderna	255783	2021-03-26	15794.714	2021-05-04	17676.143	2021-07-03	558.0
2021-03-05	Oxford/AstraZeneca	890692	2021-03-26	15794.714	2021-05-04	17676.143	2021-07-03	558.0
2021-03-05	Pfizer/BioNTech	6305870	2021-03-26	15794.714	2021-05-04	17676.143	2021-07-03	558.0
2021-03-06	Johnson&Johnson	20	2021-03-27	16241.714	2021-05-05	16846.857	2021-07-04	568.714
2021-03-06	Moderna	264256	2021-03-27	16241.714	2021-05-05	16846.857	2021-07-04	568.714
2021-03-06	Oxford/AstraZeneca	974330	2021-03-27	16241.714	2021-05-05	16846.857	2021-07-04	568.714
2021-03-06	Pfizer/BioNTech	6415460	2021-03-27	16241.714	2021-05-05	16846.857	2021-07-04	568.714

Like Q9, the results generated has stopped at 6<sup>th</sup> March 2021 for *date1* as there were no more *new\_cases\_smoothed* data after 4<sup>th</sup> July 2021. As a result, the team has also decided that information relating to sum of total\_vaccinations after 6<sup>th</sup> March 2021 was not required due to insufficient data to determine, analyse, and conclude the vaccination effects after 6<sup>th</sup> March 2021.



## 5. MongoDB & NoSQL Queries (Questions 1 to 20)

### Q1) Display a list of total vaccinations per day in Singapore.

**Approach:** To resolve this query, the team made use of the match statement to filter the documents where the location is equal to “Singapore” out. Following that, the team used the sort statement to arrange the date in an ascending order. Finally, a project statement is used to display the fields that wanted in the output.

```
db.country_vaccinations.aggregate([
  {$match:{country:"Singapore"}},
  {$sort:{date:1}},
  {$project:{_id:0, country:1, total_vaccinations:1, date:1}}
])
```

**Output:** Projecting the results of the query, the team obtained the list of figures as shown below. The team analyzed that there is an increase in the number of total vaccinations in Singapore. This is because Singapore started rolling out its vaccination to people working in key sectors such as healthcare and followed by the elderly. Afterwards, Singapore started to expand its vaccination programme to the rest of the population, such as the youth. (Co, 2021)

Key	Value	Type
▲ (1)	{ country : "Singapore", date : ISODate("2021-01-11T08:00:00.000+08:00"), total_vaccinations : 3400 }	Object
country	Singapore	String
date	1/11/2021, 8:00:00 AM	Date
total_vaccinations	3,400 (3.4K)	int32
▶ (2)	{ country : "Singapore", date : ISODate("2021-01-12T08:00:00.000+08:00"), total_vaccinations : 6200 }	Object
▶ (3)	{ country : "Singapore", date : ISODate("2021-01-13T08:00:00.000+08:00"), total_vaccinations : 0 }	Object
▶ (4)	{ country : "Singapore", date : ISODate("2021-01-14T08:00:00.000+08:00"), total_vaccinations : 0 }	Object
▶ (5)	{ country : "Singapore", date : ISODate("2021-01-15T08:00:00.000+08:00"), total_vaccinations : 0 }	Object
▶ (6)	{ country : "Singapore", date : ISODate("2021-01-16T08:00:00.000+08:00"), total_vaccinations : 0 }	Object
▶ (7)	{ country : "Singapore", date : ISODate("2021-01-17T08:00:00.000+08:00"), total_vaccinations : 0 }	Object
▶ (8)	{ country : "Singapore", date : ISODate("2021-01-18T08:00:00.000+08:00"), total_vaccinations : 0 }	Object
▶ (9)	{ country : "Singapore", date : ISODate("2021-01-19T08:00:00.000+08:00"), total_vaccinations : 0 }	Object
▶ (10)	{ country : "Singapore", date : ISODate("2021-01-20T08:00:00.000+08:00"), total_vaccinations : 0 }	Object

### Q2) Display the sum of daily vaccinations among ASEAN countries.

**Approach:** To derive the sum of daily vaccination among ASEAN countries, the team first made use of the project statement to display those field that is needed to solve for this query. Following that, the match function was used to obtain documents where countries belong to ASEAN by using the \$in function. Next, the team grouped them by country and an accumulator called “total\_administered” was created by summing up the respective ASEAN countries’ daily vaccinations. Finally, the team used the sort statement to arrange the total\_administered in a descending order.

```
db.country_vaccinations.aggregate([
  {$project: {_id:0, country:1, daily_vaccinations:1}},
  {$match:{country:{$in: ["Singapore", "Malaysia", "Brunei", "Cambodia", "Indonesia", "Laos", "Myanmar", "Philippines", "Thailand", "Vietnam"]}}},
  {$group: { _id:{country: "$country"}, total_administered:{$sum:"$daily_vaccinations"}},
  {$sort: {total_administered:-1}}
])
```

**Output:** Upon running the query, the team obtained the list of figures as shown below. Evaluating the output of the query, Indonesia has the highest total amount of administered vaccine amongst the other ASEAN countries, and the team analysed that its outcome is related to the population size of the country. Based on secondary research, the team found out that among all ASEAN countries, Singapore has the highest vaccination rate but even if Singapore has a 100% vaccinate rate, it will never be the top in ASEAN countries in terms of total amount of administered vaccine due to its population size. In addition, even though Lao has the lowest total amount of administered vaccine, it has a higher vaccination rate compared to Philippines. (COVID-19 Vaccination Rate in ASEAN, 2021)

Key	Value	Type
(1) { country : "Indonesia" }	{ total_administered : 40160688 }	Document
↳ { _id	{ country : "Indonesia" }	Object
↳ total_administered	40,160,688 (40.2M)	int32
(2) { country : "Philippines" }	{ total_administered : 9940897 }	Document
(3) { country : "Thailand" }	{ total_administered : 9051334 }	Document
(4) { country : "Malaysia" }	{ total_administered : 7437027 }	Document
(5) { country : "Cambodia" }	{ total_administered : 6853738 }	Document
(6) { country : "Singapore" }	{ total_administered : 5230097 }	Document
(7) { country : "Myanmar" }	{ total_administered : 3384231 }	Document
(8) { country : "Vietnam" }	{ total_administered : 3367360 }	Document
(9) { country : "Laos" }	{ total_administered : 1332960 }	Document
(10) { country : "Brunei" }	{ total_administered : 76971 }	Document

**Q3) Identify the maximum daily vaccinations per million of each country. Sort the list based on daily vaccinations per million in a descending order**

**Approach:** The data for this question was retrieved from the 'country\_vaccinations' collection using the aggregate and \$project function, which groups the subqueries and displays the results respectively. Data retrieved from the collection were the country names and the maximum daily\_vaccinations\_per\_million. The data was grouped using the \$group function to match the country and vaccinations per million, with a nested \$max function to derive the maximum vaccination output. The \$sort function was used to display the values from highest to lowest maximum vaccination per million count quantity as displayed in the output.

<pre> db.country_vaccinations.aggregate([   {\$project: { _id:0, country:1, daily_vaccinations_per_million:1}},   {\$group: { _id:\$country, Max_daily_vaccinations_per_million:{\$max: "\$daily_vaccinations_per_million"}},   {\$sort: {Max_daily_vaccinations_per_million:-1}} ]) </pre>	<p>Filtering data according to desired output of country and maximum daily vaccinations per million of each country</p> <p>Finding the maximum daily vaccination per million of each country using \$max function and returning the single value for each country in output</p> <p>Grouping data by their country which the daily vaccinations per million were obtained from</p> <p>Sorting maximum daily vaccination per million according to descending count using '-1'</p>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Output:** The output generated shows Bhutan as the country with highest vaccination count, as attributed to the quick vaccination rollouts and ability to secure vaccination supplies. (Schiffing S., et al., 2021) Falkland Islands was the second country to be displayed, and this shows possible intra-country efforts to secure vaccine supplies and pandemic containment efforts with an effective taskforce and vaccination delivery plan throughout the country. Despite this, it is surprising to see that the country was the second highest in terms of maximum vaccinations despite being the country with lowest data source count in *Question 13*.

Key	Value	Type
1 { country : "Bhutan" }	{ Max_daily_vaccinations_per_million : 118759 }	Document
2 { country : "Falkland Islands" }	{ Max_daily_vaccinations_per_million : 54264 }	Document
3 { country : "Cook Islands" }	{ Max_daily_vaccinations_per_million : 46231 }	Document
4 { country : "Mongolia" }	{ Max_daily_vaccinations_per_million : 38271 }	Document
5 { country : "Gibraltar" }	{ Max_daily_vaccinations_per_million : 31700 }	Document
6 { country : "Wallis and Futuna" }	{ Max_daily_vaccinations_per_million : 30500 }	Document
7 { country : "Saint Helena" }	{ Max_daily_vaccinations_per_million : 27178 }	Document
8 { country : "Aruba" }	{ Max_daily_vaccinations_per_million : 25092 }	Document
9 { country : "Seychelles" }	{ Max_daily_vaccinations_per_million : 24415 }	Document
10 { country : "Andorra" }	{ Max_daily_vaccinations_per_million : 22481 }	Document
11 { country : "Iceland" }	{ Max_daily_vaccinations_per_million : 22113 }	Document

**Q4) Which is the most administered vaccine? Display a list of total administration (i.e., sum of total vaccinations) per vaccine.**

**Approach:** The total administration was obtained by finding the sum of the maximum of each vaccine administered by each country. Firstly, the \$project statement was used to display the vaccines and total\_vaccinations for each vaccine administered. Location was used to group each vaccine by the country which ordered to respective vaccine administered. 2 different \$group statements were used, the first grouping each vaccine type by the location and obtaining the vaccine maximum administered quantity. The second \$group statement was used to obtain the sum of each vaccine by the maximum administered vaccine type by their quantity under the field total\_administered. Lastly, the team sorted the vaccine administration count to obtain the vaccine name and administered quantity in the descending order using the value '-1'. This allows us to analyse the vaccine popularity and administration count in the output.

```

db.getCollection("country_vaccinations_by_manufacturer").aggregate([
  {$project: {id:0, vaccine:1, location:1, total_vaccinations:1}},
  {$group: {_id:{location:"$location", vaccine:"$vaccine"},
    Max_total_vaccination:{$max:"$total_vaccinations"}}},
  {$group: {_id: {vaccine:"$id.vaccine"},
    total_administered:{$sum:"$Max_total_vaccination"}}},
  {$sort: {total_administered: -1}}
])

```

Filtering results to obtain vaccine type and quantity of each vaccine administered globally data found in Singapore in desired collection. Location is subsequently displayed to ensure that quantity is retrieved from each unique country

Vaccine grouped with each country, maximum of total\_vaccination administered retrieved per country and vaccine

Sum used as a mathematical operator to sum up all the quantity of vaccine retrieved for each vaccine according to the country they were obtained from

Sorting the vaccination quantity of each vaccine according to descending order using '-1' value

**Output:** The output displayed Pfizer as the most popular vaccine, probably attributed to it being the first COVID mRNA vaccine to receive the USA Food and Drug Administration approval in early December 2020. The actual output differs from the expected output of more than 40 billion vaccines administered as the max of total vaccinations was used. This led to the acceleration of its popularity and the quick rollout, manufacturing resulting in ample supply and global delivery in the race to reach the endemic. (Hopkins J.S., et al, 2021) Moderna was the next vaccine to receive the FDA approval and subsequently rolled out, receiving huge popularity amongst countries who were not able to obtain the doses of Pfizer's vaccines. The other vaccinations could have lower administration rates due to the timeline in which they were rolled out and lower efficacy rates in compared to the top 2 vaccines with efficacy rates of over 90%. (McDonald J., et al., 2021)

Key	Value	Type
(1) { vaccine : "PfizerBioNTech" }	{ total_administered : 488303302 }	Document
> _id	{ vaccine : "PfizerBioNTech" }	Object
total_administered	488303302 (0.488)	Int32
(2) { vaccine : "Moderna" }	{ total_administered : 171755566 }	Document
(3) { vaccine : "OxfordAstraZeneca" }	{ total_administered : 95489456 }	Document
(4) { vaccine : "Sinovac" }	{ total_administered : 20386592 }	Document
(5) { vaccine : "Johnson&Johnson" }	{ total_administered : 20385108 }	Document
(6) { vaccine : "SinopharmBeijing" }	{ total_administered : 2056851 }	Document
(7) { vaccine : "Sputnik V" }	{ total_administered : 1813128 }	Document
(8) { vaccine : "CanSino" }	{ total_administered : 301012 }	Document

**Q5) Italy has commenced administering various vaccines to its populations as a vaccine becomes available. Identify the first dates of each vaccine being administered, then compute the difference in days between the earliest date and the 4<sup>th</sup> date.**

**Approach:** To find the difference in days between the first date of earliest vaccine administered and the first date of fourth administered vaccine for the country Italy, the team first use the match statement to focus on the country/location, "Italy".

The team then implement the query in two parts. The inner query uses the group statement to group by the vaccine types and finds the earliest dates of each vaccine administered using "\$min" function. The team then projected the dates obtained as "date". Then, the team use the projected results in the outer query to find the first ("\$min:date") and the fourth date ("\$max:date"). Finally, the team use the project statement to find the difference in days between the maxDate and minDate by using the \$subtract function.

```
db.country_vaccinations_by_manufacturer.aggregate([
  // inner query //
  { $match: { "location": "Italy" } },
  { $group: { _id: { vaccine: "$vaccine" },
    date: { $min: "$date" } } },
  { $project: { _id: 0, vaccine: "$_id.vaccine", date: "$date" } },

  // outer query //
  { $group: { _id: { },
    "minDate": { "$min": "$date" }, "maxDate": { "$max": "$date" } } },
  { $project: { _id: 0, Difference_In_Days: { $divide: [ { $subtract: [ "$maxDate", "$minDate" ] }, 1000*60*60*24 ] } } }
])
```



**Output:** After running the query the team got the difference in days between the earliest date and the fourth date to be 90.

Key	Value	Type
(1)	{ Difference_In_Days : 90 }	Object
Difference_In_Days	90	Int32

#### Q6) What is the country with the most types of administered vaccine?

**Approach:** To find the country with the most types of administered vaccine the team first make use of project statement to select the required columns (that is, location and vaccine) from the collection (Country\_vaccinations\_by\_manufacturer). The team then make use of the group statement to group by the location and then the team use the “\$addToSet:\$vaccine” to get an array of all *unique* values of the vaccines. Then the team \$sort the results in descending order and use “\$limit:1” to get the country with most types of vaccine as well as the vaccine types (that is, the topmost value in after arranging the results in descending order).

```
db.getCollection("country_vaccinations_by_manufacturer").aggregate([
  {$project:{_id:0, location:1, vaccine:1}},
  {$group:{_id:{location:"$location"}, vaccine_types: {$addToSet: "$vaccine"}}},
  {$project: {_id:1, vaccine:1, vaccine_types: {$size: "$vaccine_types"}}},
  {$sort: {vaccine_types: -1}},
  {$limit: 1}
])
```

**Output:** According to the query the team got Hungary to be the country with most types of vaccines administered (6 vaccine\_types).

Key	Value	Type
(1){ location : "Hungary" }	{ vaccine_types : 6 }	Document
_id	{ location : "Hungary" }	Object
vaccine_types	6	Int32

#### Q7) What are the countries that have fully vaccinated more than 60% of its people? For each country, display the vaccines administrated.

**Approach:** In order to resolve this query, information relating to country, vaccines and vaccination percentage (fully vaccinated) must first be obtained. As such, the team made use of the project statement to select the information mentioned above. Following that, the match function was used to obtain records where “people\_fully\_vaccinated\_per\_hundred” is greater than 60. Finally, the filtered records were grouped by country and vaccine and a new value called “vaccinated percentage” was created by taking the max of a country’s fully vaccinated percentage.

```
db.getCollection("country_vaccinations").aggregate([
  {$project: {_id:0, country:1, vaccines:1, people_fully_vaccinated_per_hundred:1}},
  {$match:{people_fully_vaccinated_per_hundred:{$gt:60}}},
  {$group: {_id:{country: "$country", vaccines: "$vaccines"}, vaccinated_percentage: {$max: "$people_fully_vaccinated_per_hundred"}}},
  {$sort: {vaccinated_percentage: -1}}
])
```

Selecting only the columns required to resolve the query by projecting country, vaccines and "people\_fully\_vaccinated\_per\_hundred"  
 Setting condition to obtain records where fully vaccinated percentage is greater than 60  
 Max function to obtain highest vaccinated percentage for a country



**Output:** Upon running the query, the output generated is shown below. Looking at the results, Gibraltar has the highest vaccinated percentage at 115.32% among all the other countries listed. Although the figure of 115.32% may seem erroneous initially, the team's research revealed that the reason why Gibraltar has a vaccination percentage that high is because the country has been administering vaccines to thousands of workers who live in Spain but commute across the border to Gibraltar everyday (Farzan, 2021). This was made possible as Gibraltar had already finished administering vaccines to all adults in the country, mainly due to its small population size of 30,000 and the steady stream of vaccine doses imported from Britain.

Key	Value	Type
(1) { country : "Gibraltar", vaccines : "PfizerBioNTech" }	{ vaccinated_percentage : 115.32 }	Document
↳ _id	{ country : "Gibraltar", vaccines : "PfizerBioNTech" }	Object
↳ vaccinated_percentage	115.32	float
(2) { country : "Malta", vaccines : "Johnson&Johnson; Moderna; Oxford/AstraZeneca; PfizerBioNTech" }	{ vaccinated_percentage : 73.81 }	Document
(3) { country : "Seychelles", vaccines : "Oxford/AstraZeneca; Sinopharm/Beijing" }	{ vaccinated_percentage : 88.44 }	Document
(4) { country : "Cayman Islands", vaccines : "Oxford/AstraZeneca; PfizerBioNTech" }	{ vaccinated_percentage : 88.66 }	Document
(5) { country : "San Marino", vaccines : "PfizerBioNTech; Sputnik V" }	{ vaccinated_percentage : 85.88 }	Document
(6) { country : "Bermuda", vaccines : "PfizerBioNTech" }	{ vaccinated_percentage : 82.68 }	Document

## Q8) Monthly vaccination insight – display the monthly total vaccination amount of each vaccine per month in the United States.

**Approach:** To derive the monthly vaccination insights for the United States each month, the team first utilised the match function to derive documents where location = "United States". Following that, the relevant information is then selected using the project statement. In this case, information on vaccine, total vaccinations and the date was required. For the date information, the function \$month was used to extract the month of a date. Next, the group statement was used to combine documents according to the months and vaccine type. A new value called "monthly\_total\_vaccination" was also created by taking the max of total vaccinations for each month.

```

db.getCollection("country_vaccinations_by_manufacturer").aggregate([
  { $match: { location: "United States" } },
  { $project: { _id: 0, date: { $month: "$date" }, vaccine: 1, total_vaccinations: 1 } },
  { $group: { _id: { month: "$date", vaccine: "$vaccine" }, monthly_total_vaccination: { $max: "$total_vaccinations" } } },
  { $sort: { _id: 1 } }
])

```

→ Selecting documents where location = "United States"

→ Extracting month from date and projecting relevant data for query

→ Extracting highest total vaccination number for each month

**Output:** Evaluating the output of the query, it can be seen that monthly vaccination rates have gone up sharply between January 2021 and June 2021 with an approximately 10-fold increase. Some reasons for the increase may be due to the availability of the Johnson&Johnson vaccine, the intensification of efforts to implement vaccine mandates by the Biden administration and the increase in pro-vaccination campaigns encouraging more people to be vaccinated. Information from the table also revealed Pfizer to be the most administer vaccine in the United States. This can be attributed to the fact that it was the first vaccine approved by authorities for use.

1	{ "month": 1, "vaccine": "Moderna" }	{ "monthly_total_vaccination": 1400000 }	Document
2	{ "month": 1, "vaccine": "PfizerBioTech" }	{ "monthly_total_vaccination": 1077000 }	Document
3	{ "month": 1, "vaccine": "Johnson&Johnson" }	{ "monthly_total_vaccination": 1000000 }	Document
4	{ "month": 2, "vaccine": "Moderna" }	{ "monthly_total_vaccination": 1520000 }	Document
5	{ "month": 2, "vaccine": "PfizerBioTech" }	{ "monthly_total_vaccination": 1000000 }	Document
6	{ "month": 2, "vaccine": "Johnson&Johnson" }	{ "monthly_total_vaccination": 1000000 }	Document
7	{ "month": 3, "vaccine": "Moderna" }	{ "monthly_total_vaccination": 1500000 }	Document
8	{ "month": 3, "vaccine": "PfizerBioTech" }	{ "monthly_total_vaccination": 1000000 }	Document
9	{ "month": 3, "vaccine": "Johnson&Johnson" }	{ "monthly_total_vaccination": 1000000 }	Document
10	{ "month": 4, "vaccine": "Moderna" }	{ "monthly_total_vaccination": 1500000 }	Document
11	{ "month": 4, "vaccine": "PfizerBioTech" }	{ "monthly_total_vaccination": 1000000 }	Document
12	{ "month": 4, "vaccine": "Johnson&Johnson" }	{ "monthly_total_vaccination": 1000000 }	Document
13	{ "month": 5, "vaccine": "Moderna" }	{ "monthly_total_vaccination": 1500000 }	Document
14	{ "month": 5, "vaccine": "PfizerBioTech" }	{ "monthly_total_vaccination": 1000000 }	Document
15	{ "month": 5, "vaccine": "Johnson&Johnson" }	{ "monthly_total_vaccination": 1000000 }	Document
16	{ "month": 6, "vaccine": "Moderna" }	{ "monthly_total_vaccination": 1500000 }	Document
17	{ "month": 6, "vaccine": "PfizerBioTech" }	{ "monthly_total_vaccination": 1000000 }	Document
18	{ "month": 6, "vaccine": "Johnson&Johnson" }	{ "monthly_total_vaccination": 1000000 }	Document

**Q9) 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)**

**Approach:** To retrieve the number of days each country takes to go above the 50% threshold, the team first used the group function to group all countries together while extracting the earliest date using the minimum function. Within the group function, the push function was used to collect all dates and vaccination numbers per hundred in an array based on the country grouped earlier. Next, the team projected the variables while adding a filter function to specifically extract the countries vaccination rates that are above 50%. The filter function will extract all values that are above 50% and store them into an array. Then, using the projected results of the earlier code, the team utilised the dateDiff function to create a new projection. The dateDiff was used to find the difference in days between the earliest date specified in the group function, and the earliest date within the array generated by the earlier filter function. Finally, to ensure that the output generated is clean for analysis, the team used the match function to ensure that only non-null values are shown while sorting by country in an ascending alphabetical order.

```

db.getCollection("country_vaccinations").aggregate([
  { $group: {
    _id: { Country: "$country" },
    "First_date": { $min: "$date" },
    "DellyPercentages": { $push: { "date": "$date",
    "Vaccinations_per_hundred": "$total_vaccinations_per_hundred" } } },
  { $project: {
    _id: 1,
    First_date: 1,
    Country: "$_id.Country",
    DellyPercentages2: { $filter: {
      inputs: "$DellyPercentages",
      as: "x",
      cond: { $gt: [ "$x.Vaccinations_per_hundred", 50 ] }
    } } },
    { $project: { _id: 0, Country: 1, DaysOverFifty: { $dateDiff: { startDate: "$First_date", endDate: { $min: "$DellyPercentages2.date" }, unit: "day" } } },
    { $match: { "DaysOverFifty": { $notNull: true } } },
    { $sort: { Country: 1 } }
  ] }

```

**Output:** From the output, the team has observed an incredibly low number of days from some of the results. One example is Bhutan, despite being less developed, it has been hailed as “arguably the fastest vaccination campaign to be executed during a pandemic” (*Aljazeera, 2021*) as it sees its vaccination rates hit 90% within a span of 7 days. This is in line with the team’s output as it shows Bhutan only took 5 days to hit 50%. This is possible due to the low population (<800,000) that it has. Furthermore, the government of Bhutan was also active in rolling out the vaccination system, implementing a digital platform that allows Bhutan citizens to pre-register online before receiving jabs to lower the processing time needed at the vaccine centre. Most importantly, the citizens also play a huge role as they trust their government and thus are willing to come forward to be vaccinated. From the output, the team observed Norway taking the longest to vaccinate 50% of its population. This is due to the country being larger and coupled with the delays experienced. From the team’s research, it was found that 90% Norwegians wanted to get vaccinated, and this results in a larger effort to vaccine everyone together. Thus, the Norway Institute of Public Health has added 4 to 5-week delay in their updated vaccine scenario (*Norwegian News Agency, 2021*). This explains the large number of days it took despite it being smaller than some countries such as China and America. The team also noticed some abnormality within the output as countries like Sint Maarten and Pitcairn were observed to hit 50% in 0 days. Upon further research, the team found that this is due to their low population (<50,000) and the proactive efforts by the government to encourage citizens to come forward to get vaccinated.

Key	Value	Type
IC3 (1)	{ Country : "Andorra", DaysOverFifty : 130 }	Object
IC3 (2)	{ Country : "Anguilla", DaysOverFifty : 400 }	Object
IC3 (3)	{ Country : "Antigua and Barbuda", DaysOverFifty : 104 }	Object
IC3 (4)	{ Country : "Aruba", DaysOverFifty : 13 }	Object
IC3 (5)	{ Country : "Austria", DaysOverFifty : 145 }	Object
IC3 (6)	{ Country : "Bahamas", DaysOverFifty : 104 }	Object
IC3 (7)	{ Country : "Barbados", DaysOverFifty : 111 }	Object
IC3 (8)	{ Country : "Belgium", DaysOverFifty : 143 }	Object
IC3 (9)	{ Country : "Bermuda", DaysOverFifty : 71 }	Object
IC3 (10)	{ Country : "Bhutan", DaysOverFifty : 5 }	Object
IC3 (11)	{ Country : "British Virgin Islands", DaysOverFifty : 1 }	Object
IC3 (12)	{ Country : "Canada", DaysOverFifty : 155 }	Object

Key	Value	Type
IC3 (55)	{ Country : "Morocco", DaysOverFifty : 146 }	Object
IC3 (56)	{ Country : "Maurit", DaysOverFifty : 40 }	Object
IC3 (57)	{ Country : "Netherlands", DaysOverFifty : 133 }	Object
IC3 (58)	{ Country : "Northern Ireland", DaysOverFifty : 113 }	Object
IC3 (59)	{ Country : "Norway", DaysOverFifty : 181 }	Object
IC3 (60)	{ Country : "Poland", DaysOverFifty : 0 }	Object
IC3 (61)	{ Country : "Puerto Rico", DaysOverFifty : 150 }	Object
IC3 (62)	{ Country : "Portugal", DaysOverFifty : 147 }	Object
IC3 (63)	{ Country : "Qatar", DaysOverFifty : 123 }	Object
IC3 (64)	{ Country : "Saint Helena", DaysOverFifty : 40 }	Object
IC3 (65)	{ Country : "Saint Kitts and Nevis", DaysOverFifty : 99 }	Object
IC3 (66)	{ Country : "San Marino", DaysOverFifty : 45 }	Object
IC3 (69)	{ Country : "Russia", DaysOverFifty : 0 }	Object
IC3 (72)	{ Country : "Sint Maarten (Dutch part)", DaysOverFifty : 0 }	Object

#### Q10) Compute the global total of vaccinations per vaccine.

**Approach:** For this question, the team first utilised the project function to output the needed variables: vaccine, location, and total\_vaccinations. Then, the team utilised the group function to group the total\_vaccination according to location and vaccine, while extracting the highest total\_vaccination rate using the max function to prevent double counting. Next, using the numbers extracted from the group function, the team performed another group to extract the global total by summing up all the highest total\_vaccination rate extracted earlier while grouping them by vaccines.

```
db.getCollection("country_vaccinations_by_manufacturer").aggregate([
  {$project: {_id:0, vaccine:1, location:1, total_vaccinations:1}},
  {$group: {_id:{location: "$location", vaccine:"$vaccine"},
    Max_total_vaccination:{$max:"$total_vaccinations"}}},
  {$group: {_id: {vaccine:"$_id.vaccine"},
    total_administered:{$sum:"$Max_total_vaccination"}}},
  {$sort: {total_administered: -1}}
])
```

**Output:** Based on the output generated, Pfizer/BioNTech is the most-used vaccine followed by Moderna and Oxford/AstraZeneca respectively. Pfizer being the widely used vaccine is likely due to its high effectiveness, especially against the highly contagious Delta Variant (*Wall Street Journal*, 2021). Pfizer has also been vouched to contain lesser side effects as compared to Moderna (*The Straits Times*, 2021), the next in-line for the most-used vaccine. Furthermore, health concerns such as blood-clotting was observed in other vaccines such as Johnson&Johnson and AstraZeneca, but not in Pfizer. Thus, making Pfizer the more popular vaccine due to its effectiveness and safety. Sinovac, a vaccine that is also approved in Singapore is seen to ranked 4<sup>th</sup>. The reason for this is due to the longer time it took to be approved and accepted by other countries for use. For Singapore, the team has found that there was a lack of data for the country to approve Sinovac as a vaccine to be recognised. For other countries across the world, it was likely due to a lack of confidence in the Chinese vaccine. As for CanSino, the lowest-used vaccine, it was found that there was a low efficacy rate, resulting in low effectiveness against the virus. Thus, it was not widely-used and recommended.

Key	Value {}	Type
(1) { vaccine: "Pfizer/BioNTech" }	{ Global_total : 488303362 }	Document
(2) { vaccine: "Moderna" }	{ Global_total : 171555596 }	Document
(3) { vaccine: "Oxford/AstraZeneca" }	{ Global_total : 154614456 }	Document
(4) { vaccine: "Sinovac" }	{ Global_total : 20984582 }	Document
(5) { vaccine: "Johnson&Johnson" }	{ Global_total : 20365106 }	Document
(6) { vaccine: "Sinopharm/Beijing" }	{ Global_total : 2056681 }	Document
(7) { vaccine: "Sputnik V" }	{ Global_total : 1813428 }	Document
(8) { vaccine: "CanSino" }	{ Global_total : 891013 }	Document

### Q11) What is the total population in Asia?

**Approach:** The team first made use of the match statement to filter the documents where the continent is equal to “Asia” out. Following that, the team use the project statement to display those field that is needed to solve for this query such as location, continent as well as population. Next, the team grouped them by location, continent and an accumulator called “max\_population” was created by taking the max of the respective Asia countries’ population. Finally, the team used another group statement to create an accumulator called “total\_population” by summing up the “max\_population”. As mentioned in question 1 under the section of relational database queries, the team identified that the population is an accumulated figure. Thus, the team needed to create two accumulators: one to get the maximum population of Asian countries, and the other to get Asia's overall population based on the first accumulators.

```
db.getCollection("covid19data").aggregate([
  {$match: {continent: "Asia"}},
  {$project: {location:1, continent:1, population:1}},
  {$group: {_id: {location: "$location", continent: "$continent"}, max_population: {$max: "$population"}}},
  {$group: {_id: "$_id.continent", total_population: {$sum: "$max_population"}}}
])
```

**Output:** Projecting the results of the query, the team obtained the same figure that the team obtained in question 1 under the section of relational database queries.

Key	Value	Type
(1) Asia	{ total_population : 4614068610 }	Document
_id	Asia	String
total_population	4,614,068,610 (4.6G)	Int32

### Q12) What is the total population among the ten ASEAN countries?

**Approach:** Using the same approach as question 11. Instead of filtering out documents whereby the continent is not in Asia, the team filtered out documents that the location does not belong to ASEAN by using the \$in function.

```
db.getCollection("covid19data").aggregate([
  {$match: {location: {$in: ["Singapore", "Malaysia", "Brunei", "Cambodia", "Indonesia", "Laos", "Myanmar", "Philippines", "Thailand", "Vietnam"]}}},
  {$project: {location:1, continent:1, population:1}},
  {$group: {_id: {location: "$location", continent: "$continent"}, country_population: {$max: "$population"}}},
  {$group: {_id: "$_id.continent", total_population_in_ASEAN: {$sum: "$country_population"}}}
])
```

**Output:** Upon running the query, the output generated is shown below. Which the team obtained the same list of figures that the team obtained in question 2 under the section of relational database queries.

Key	Value	Type
(1) Asia	{ total_population_in_ASEAN : 667301412 }	Document
_id	Asia	String
total_population_in_ASEAN	667,301,412 (0.67G)	Int32



### Q13) Generate a list of unique data sources (source\_name).

**Approach:** The question was approached using the aggregate function obtaining data from the 'country\_vaccinations' collection to nest all the \$group, \$sort and \$project statements. Firstly, the data was grouped by their unique source names and counted using the \$sum function, with each count being stored in the totalSourceCount field and matched to their respective source names. The data was then parsed through a \$sort statement to obtain an output sorting the data according to a descending order, with the highest quantity displaying at the top of the output using the value of '-1'. The \$project statement was then used to display the list of unique data sources with their respective source count.

```
db.country_vaccinations.aggregate([
  { $group: { _id: { source_name: "$source_name" }, totalSourceCount: { $sum: 1 } } },
  { $sort: { totalSourceCount: -1 } },
  { $project: { source_name: 1, totalSourceCount: 1 } }
])
```

Grouping data source by their unique source name

Grouped data is counted using the \$sum function to obtain respective data source quantity count

Sorting data count according to descending count using "-1"

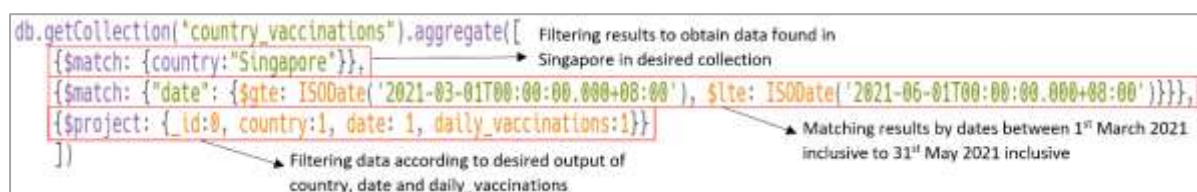
Filtering data according to desired output of unique source name and the quantity of each source

**Output:** The output returned data sorted according to the quantity of sources and the source name that the data was derived from, with Ministry of Health being the top source, World Health Organization being second, and the Government of the United Kingdom being the third. This could be due to the advancements in technology in Singapore used to capture data about the vaccinations and covid data locally, resulting in higher data source count in Singapore, as well as an active COVID taskforce in obtaining the country's vaccinations/pandemic data.

Key	Value	Type
(1) { source_name : "Ministry of Health" }	{ totalSourceCount : 7074 }	Document
{ _id :	{ source_name : "Ministry of Health" }	Object
totalSourceCount :	7,074 (7.1K)	Int32
(2) { source_name : "World Health Organization" }	{ totalSourceCount : 4715 }	Document
(3) { source_name : "Government of the United Kingdom" }	{ totalSourceCount : 1010 }	Document
(4) { source_name : "SPC Public Health Division" }	{ totalSourceCount : 897 }	Document
(5) { source_name : "Pan American Health Organization" }	{ totalSourceCount : 690 }	Document

**Q14) Specific to Singapore, display the daily total\_vaccinations starting (inclusive) March-1 2021 through (inclusive) May-31 2021.**

**Approach:** To obtain the daily total vaccination rates similar to the Relational Database Queries, the team used the \$match function similar to SQL's where statement to ensure that the country that the data is obtained from is Singapore, and the collection obtained is from 'country\_vaccinations'. The data obtained was then further matched against the dates between 1<sup>st</sup> March 2021 inclusive to 31<sup>st</sup> May 2021 inclusive using a second \$match statement with the function \$gte nesting inside the date field to ensure the starting and ending dates are included in the queries. Finally, in the \$project statement, \_id was suppressed to prevent it from showing as the team only want to display the country, date and daily vaccination rates that were matched to the date itself. NoSQL will then sort and return the output according to the dates in ascending order, displaying the oldest dates at the top of the output.



**Output:** The output returned all dates recorded in the NoSQL database with the respective daily vaccination rates, from 1<sup>st</sup> March 2021 inclusive to 31<sup>st</sup> May 2021 inclusive. This was cross-referenced with the country that the data was obtained from, which in this scenario was Singapore, to ensure accuracy of data. For further analysis on the output, refer to question 4 under the SQL Relational database queries.

Key	Value	Type
1 (1)	{ country: "Singapore", date: ISODate("2021-03-01T00:00:00.000+08:00") }	Object
country	Singapore	String
date	3/1/2021, 8:00:00 AM	Date
daily_vaccinations	15,004 (15.0K)	int32
2 (2)	{ country: "Singapore", date: ISODate("2021-03-02T00:00:00.000+08:00") }	Object
3 (3)	{ country: "Singapore", date: ISODate("2021-03-03T00:00:00.000+08:00") }	Object
4 (4)	{ country: "Singapore", date: ISODate("2021-03-04T00:00:00.000+08:00") }	Object
5 (5)	{ country: "Singapore", date: ISODate("2021-03-05T00:00:00.000+08:00") }	Object
6 (6)	{ country: "Singapore", date: ISODate("2021-03-06T00:00:00.000+08:00") }	Object
7 (7)	{ country: "Singapore", date: ISODate("2021-03-07T00:00:00.000+08:00") }	Object
8 (8)	{ country: "Singapore", date: ISODate("2021-03-08T00:00:00.000+08:00") }	Object
9 (9)	{ country: "Singapore", date: ISODate("2021-03-09T00:00:00.000+08:00") }	Object

### Q15) When is the first batch of vaccinations recorded in Singapore?

**Approach:** To get the date when the first batch of vaccinations was recorded in Singapore, the team first use the match statement to focus on document where the country is "Singapore". Then, the team made use of the group statement to find the first vaccination date as First\_vaccination using the function \$min. Finally, the team project the result obtained using the project statement.

```
db.getCollection("country_vaccinations").aggregate([
  {$match: {country:"Singapore"}},
  {$group: { _id: "$country", First_vaccination: {$min: "$date"} }},
  {$project: { _id:0, First_vaccination:1}}
])
```

**Output:** Implementing the query the team obtained the first vaccination date to be 11-01-2021 (11 January 2021).

Key	Value	Type
1	{ First_vaccination: ISODate("2021-01-11T08:00:00.000+08:00") }	Object
First_vaccination	1/11/2021, 8:00:00 AM	Date

### Q16) Based on the date identified in (15), specific to Singapore, compute the total number of new cases thereafter.

**Approach:** To get the total number of new cases after the date identified in the Q15 the team apply a similar approach as used in the SQL query 5. The team first use the project statement to get select the necessary information, which is the location, date and the new\_cases (The raw data for the number of new cases was used instead of the smoothed data since no trend analysis of the data was required). The team then use one match statement to select the documents where the location is "Singapore" and the other match statement to select the documents with date greater than and equal to (\$gte) the date obtained in question 15 (that is, 11-01-2021). Then, the team use the group statement to get the sum of the new cases as total\_new\_cases using the \$sum function. Finally, the team projected the result obtained using the project statement.

```
db.getCollection("covid19data").aggregate([
  {$project: {location:1, date: {$convert: { input: "$date", to: "date"}},
    new_cases: {$convert: { input: "$new_cases", to: "double"} }},
  {$match: {location:"Singapore"}},
  {$match: {"date": {$gte: ISODate('2021-01-11T00:00:00.000+08:00')}}},
  {$group: { _id:{}, total_new_cases: {$sum: "$new_cases"} }},
  {$project: { _id:0, total_new_cases:1}}
])
```

**Output:** Implementing the queries the team got the total\_new\_cases after 11/01/2021 to be 3,710.

Key	Value	Type
1	{ total_new_cases : 3710 }	Object
total_new_cases	3,710 (3.7K)	Int32



**Q17) Compute the total number of new cases in Singapore before the date identified in (15).**

**Approach:** Similar to the approach in SQL, the team first selected the information relevant to the resolution of the query which mainly included the location, date and “new\_cases”. Next, a match statement is used to filter out documents where the location is specified to be “Singapore”. Following that, a second match statement is then included to select documents dated earlier than 11 January 2021. In the case of NoSQL, ISODate format was used rather than. This is slightly different from the date format in SQL where it is specified in terms of only years, months and days. Finally, a group statement is formulated to add up all the new cases before 11 January 2021.

```
db.getCollection("covid19data").aggregate([
  {$project: {location:1, date:1, new_cases:1}},
  {$match: {location:"Singapore"}},
  {$match: {"date": {$lt: ISODate('2021-01-11T00:00:00.000+08:00')}}},
  {$group: { _id:{}, total_new_cases: {$sum: "$new_cases"} }},
  {$project: { _id:0, total_new_cases:1}}
])
```

→ Selecting documents where location = "Singapore"

→ Adding up all new cases before 11 January 2021

→ Selecting documents where date is earlier than 11 January 2021 using ISODate format.

**Output:** Projecting the results of the query, the team obtained the same figure of 58,907 that the team obtained in SQL. For the analysis of the output, refer to question 7 under the section of relational database queries.

Key	Value	Type
0(1)	{total_new_cases:58907}	Object

**Q18) 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.**

**Approach:** Due to the length of the codes, the approach for this question will be broken down into 2 parts for ease of understanding. In part 1, a match statement is first created to filter for documents in “Covid19data” where location = “Germany”. Next, a lookup (known as left outer join in SQL) was performed on the table titled “Country\_vaccinations\_by\_manufacturer” based on matching dates. The resulting information is then funnelled through a pipeline where it is filtered to identify documents with location = “Germany”. Relevant information is then projected and stored as “a”. Finally in order to use the data stored within “a”, \$unwind was used to deconstruct the array field to output a document for each element within.

Part 1

```
db.getCollection("covid19data").aggregate([
  {$match: {location:"Germany"}},
  {$lookup: {
    from: "country_vaccinations_by_manufacturer",
    localField: "date",
    foreignField: "date",
    pipeline: [
      {$match: {location:"Germany"}},
      {$project: {location:1, date:1, vaccine:1, total_vaccinations:1}}
    ],
    as: "a"
  }},
  {$unwind: "$a"},
  {$project: {location:1, date:1, vaccine:1, total_vaccinations:1}}
])
```

→ Filtering for documents where location = "Germany"

→ Performing a left outer join based on matching date

→ Filtering results for documents where location = "Germany" and projecting the relevant information for query resolution. Final outcome is stored as "a".

→ Deconstructing the array field "a" to output a document for each element

With the necessary information obtained, the team moved on to project the information required from the array field “a” and the Covid19data collection. Two new fields are also added by dividing “new\_cases\_smoothed” and “total\_vaccinations” by the population and multiplying the outcome by 100 to convert them into percentages. Finally, a group by statement is used to consolidate all the data obtained.

Part 2

```
{
  $project: {
    _id: 0,
    location: 1,
    date: 1,
    vaccine: "$a.vaccine",
    total_vaccinations: "$a.total_vaccinations",
    new_cases_smoothed: 1,
    population: 1
  },
  $addFields: {
    percentage_of_new_cases: {
      $multiply: [
        {
          $divide: [
            "$new_cases_smoothed",
            "$population"
          ]
        },
        100
      ]
    },
    percentage_of_total_vaccinations: {
      $multiply: [
        {
          $divide: [
            "$total_vaccinations",
            "$population"
          ]
        },
        100
      ]
    }
  },
  $group: {
    _id: {
      location: "$location",
      date: "$date",
      vaccine: "$vaccine",
      percentage_of_new_cases: "$percentage_of_new_cases",
      percentage_of_total_vaccinations: "$percentage_of_total_vaccinations"
    }
  },
  $sort: {
    _id: 1
  }
}
```

Projecting relevant information from array “a” and “Covid19data”

Adding new fields by dividing “new\_cases\_smoothed” & “total\_vaccinations” by the population and multiplying it by 100 to find the percentage

Grouping data by location, date, vaccine, “percentage\_of\_new\_cases” & “percentage\_of\_total\_vaccinations”

**Output:** Upon running the query, the outcome obtained matches the outcome that was previously derived when using SQL. For further analysis on the output, refer to question 8 under the section of relational database queries.

```
{
  "_id" : {
    "location" : "Germany",
    "date" : ISODate("2021-06-30T08:00:00.000+08:00"),
    "vaccine" : "Moderna",
    "percentage_of_new_cases" : 0.0006421277966799009,
    "percentage_of_total_vaccinations" : 7.7234988159127616
  },
},
/* 743 */
{
  "_id" : {
    "location" : "Germany",
    "date" : ISODate("2021-06-30T08:00:00.000+08:00"),
    "vaccine" : "Oxford/AstraZeneca",
    "percentage_of_new_cases" : 0.0006421277966799009,
    "percentage_of_total_vaccinations" : 13.809513266533344
  },
},
/* 744 */
{
  "_id" : {
    "location" : "Germany",
    "date" : ISODate("2021-06-30T08:00:00.000+08:00"),
    "vaccine" : "Pfizer/BioNTech",
    "percentage_of_new_cases" : 0.0006421277966799009,
    "percentage_of_total_vaccinations" : 65.52405714483844
  },
}
```

**Q19) 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.**

**Approach:** For this question, the team first used the match function to ensure that all results generated are for Germany only. Then, the team projected the variables required while adding in the "20Days", "30Days", and "40Days" variables which derives from adding 20, 30, and 40 days respectively to the date retrieved.

```
db.getCollection("covid19data").aggregate([
  {$match:{location:"Germany"}},
  {$project:{
    _id:0,
    location:1,
    date:1,
    new_cases_smoothed:1,
    "20Days":{$add:["$date",1000*60*60*24*20]},
    "30Days":{$add:["$date",1000*60*60*24*30]},
    "40Days":{$add:["$date",1000*60*60*24*40]}
  }},

```

Then, using the results obtained, the team utilised the lookup function as variables from another collection (country\_vaccinations\_by\_manufacturer) was needed to answer the question. The variables retrieved were date, vaccine, and total\_vaccinations, with a match function to ensure that each variable retrieved only contained information regarding Germany. There will be a total of three lookup functions as each lookup is used to match the earlier created date variables (20Days, 30Days, 40Days) and retrieve the relevant information corresponding to those dates.

```
{
  $lookup: {
    from: "country_vaccinations_by_manufacturer",
    localField: "20Days",
    foreignField: "date",
    pipeline: [
      {
        $match: {location: "Germany"}
      },
      {
        $project: {
          _id: 0,
          date: 1,
          vaccine: 1,
          total_vaccinations: 1
        }
      }
    ],
    as: "20DaysVax"
  },
  $lookup: {
    from: "country_vaccinations_by_manufacturer",
    localField: "30Days",
    foreignField: "date",
    pipeline: [
      {
        $match: {location: "Germany"}
      },
      {
        $project: {
          _id: 0,
          date: 1,
          vaccine: 1,
          total_vaccinations: 1
        }
      }
    ],
    as: "30DaysVax"
  },
  $lookup: {
    from: "country_vaccinations_by_manufacturer",
    localField: "40Days",
    foreignField: "date",
    pipeline: [
      {
        $match: {location: "Germany"}
      },
      {
        $project: {
          _id: 0,
          date: 1,
          vaccine: 1,
          total_vaccinations: 1
        }
      }
    ],
    as: "40DaysVax"
  }
},

```

Finally, the match function is used to ensure that no null or blank values are returned. After which, the team projected the variables accordingly, country, date, new\_cases\_smoothed, 20DaysVax, 30DaysVax, 40DaysVax with each of the “DaysVax” variable containing the date, vaccine, and total vaccinations.

```
{
  $match: { "20DaysVax": { $ne: [] } }, /* before here if no need to display nicely */
  $project: { _id: 0, country: 1, date: 1, new_cases_smoothed: 1, "20DaysVax": "$20DaysVax", "30DaysVax": "$30DaysVax", "40DaysVax": "$40DaysVax" },
  $match: { "30DaysVax": { $ne: [] } },
  $match: { "40DaysVax": { $ne: [] } }
}
```

**Output:** Upon running the query, the results obtained matches the outcome that was previously derived when using SQL. For further analysis on the output, refer to question 9 under the section of relational database queries.

**Screenshots for (2020-12-07):**

```

"date" : ISODate( "2020-12-07T08:00:00.000+08:00" ),
"new_cases_smoothed" : 18584.857,
"20DaysVax" : [
  {
    "vaccine" : "Johnson&Johnson",
    "total_vaccinations" : 0,
    "date" : ISODate( "2020-12-27T08:00:00.000+08:00" )
  },
  {
    "vaccine" : "Moderna",
    "total_vaccinations" : 2,
    "date" : ISODate( "2020-12-27T08:00:00.000+08:00" )
  },
  {
    "vaccine" : "Oxford/AstraZeneca",
    "total_vaccinations" : 0,
    "date" : ISODate( "2020-12-27T08:00:00.000+08:00" )
  },
],
"30DaysVax" : [
  {
    "vaccine" : "Johnson&Johnson",
    "total_vaccinations" : 0,
    "date" : ISODate( "2021-01-06T08:00:00.000+08:00" )
  },
  {
    "vaccine" : "Moderna",
    "total_vaccinations" : 16,
    "date" : ISODate( "2021-01-06T08:00:00.000+08:00" )
  },
  {
    "vaccine" : "Oxford/AstraZeneca",
    "total_vaccinations" : 5,
    "date" : ISODate( "2021-01-06T08:00:00.000+08:00" )
  },
  {
    "vaccine" : "Pfizer/BioNTech",
    "total_vaccinations" : 459272,
    "date" : ISODate( "2021-01-06T08:00:00.000+08:00" )
  },
],
"40DaysVax" : [
  {
    "vaccine" : "Johnson&Johnson",
    "total_vaccinations" : 0,
    "date" : ISODate( "2021-01-16T08:00:00.000+08:00" )
  },
  {
    "vaccine" : "Moderna",
    "total_vaccinations" : 13341,
    "date" : ISODate( "2021-01-16T08:00:00.000+08:00" )
  },
  {
    "vaccine" : "Oxford/AstraZeneca",
    "total_vaccinations" : 14,
    "date" : ISODate( "2021-01-16T08:00:00.000+08:00" )
  },
  {
    "vaccine" : "Pfizer/BioNTech",
    "total_vaccinations" : 1144835,
    "date" : ISODate( "2021-01-16T08:00:00.000+08:00" )
  },
]

```

**Q20) 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.

**Approach:** The code for this question is similar to Q19. The team first used the match function to ensure that all results generated are for Germany only. Then, the team projected the variables required while adding in the “21Days”, “60Days”, and “120Days” variables which derives from adding 21, 60, and 120 days respectively to the date retrieved.



```

db.getCollection("covid19data").aggregate([
  {$match:{location:"Germany"}},
  {$project:{
    _id:0,
    location:1,
    date:1,
    "21Days":{$add:["$date",1000*60*60*24*21]},
    "60Days":{$add:["$date",1000*60*60*24*60]},
    "120Days":{$add:["$date",1000*60*60*24*120]}
  }},

```

Then, using the results obtained, the team utilised the lookup function as variables from another collection ("country\_vaccinations\_by\_manufacturer") was needed to answer the question. The variables retrieved were date and new\_cases\_smoothed, with a match function to ensure that each variable retrieved only contained information regarding Germany. There will be a total of three lookup functions as each lookup is used to match the earlier created date variables (21Days, 60Days, 120Days) and retrieve the relevant information corresponding to those dates.

```

{$lookup:{
  from:"country_vaccinations_by_manufacturer",
  localField:"date",
  foreignField:"date",
  pipeline:[
    {$match:{location:"Germany"}},
    {$project:{_id:0,date:1,vaccine:1, total_vaccinations:1}}
  ],
  as: "Vaccine_Count"
}},
{$lookup:{
  from:"covid19data",
  localField:"21Days",
  foreignField:"date",
  pipeline:[
    {$match:{location:"Germany"}},
    {$project:{_id:0, new_cases_smoothed:1, date:1}}
  ],
  as: "21DaysCases"
}},
{$lookup:{
  from:"covid19data",
  localField:"60Days",
  foreignField:"date",
  pipeline:[
    {$match:{location:"Germany"}},
    {$project:{_id:0,new_cases_smoothed:1, date:1}}
  ],
  as: "60DaysCases"
}},
{$lookup:{
  from:"covid19data",
  localField:"120Days",
  foreignField:"date",
  pipeline:[
    {$match:{location:"Germany"}},
    {$project:{_id:0,new_cases_smoothed:1, date:1}}
  ],
  as: "120DaysCases"
}},

```

Finally, the match function is used to ensure that no null or blank values are returned. After which, the team projected the variables accordingly, country, date, vaccination count as Vaccine\_Count, 21DaysCases, 60DaysCases, 120DaysCases with each of the "DaysCases" variable containing the date, new\_cases extracted previously.

```

{ $match: { "21DaysCases": { $ne: [] } } }, /* before here if no need to display nicely */
{ $match: { "60DaysCases": { $ne: [] } } },
{ $match: { "120DaysCases": { $ne: [] } } },
{ $match: { "Vaccine_Count": { $ne: [] } } },
{ $match: { "location": "Germany" } },
{ $project: { _id: 0, country: 1, date: 1,
  "Vaccine_Count": "$Vaccine_Count",
  "21DaysCases": "$21DaysCases",
  "60DaysCases": "$60DaysCases",
  "120DaysCases": "$120DaysCases" } }
}
1)

```

**Output:** Upon running the query, the outcome obtained matches the outcome that was previously derived when using SQL. For further analysis on the output, refer to question 10 under the section of relational database queries.

```

/* 1 */
{
  "date" : ISODate("2020-12-27T08:00:00.000+08:00"),
  "Vaccine_Count" : [
    {
      "vaccine" : "Johnson&Johnson",
      "date" : ISODate("2020-12-27T08:00:00.000+08:00"),
      "total_vaccinations" : 0
    },
    {
      "vaccine" : "Moderna",
      "date" : ISODate("2020-12-27T08:00:00.000+08:00"),
      "total_vaccinations" : 2
    },
    {
      "vaccine" : "Oxford/AstraZeneca",
      "date" : ISODate("2020-12-27T08:00:00.000+08:00"),
      "total_vaccinations" : 0
    },
    {
      "vaccine" : "Pfizer/BioNTech",
      "date" : ISODate("2020-12-27T08:00:00.000+08:00"),
      "total_vaccinations" : 23319
    }
  ]
},

```

```

],
"21DaysCases" : [
  {
    "date" : ISODate("2021-01-17T08:00:00.000+08:00"),
    "new_cases_smoothed" : 17245.571
  }
],
"60DaysCases" : [
  {
    "date" : ISODate("2021-02-25T08:00:00.000+08:00"),
    "new_cases_smoothed" : 7837.143
  }
],
"120DaysCases" : [
  {
    "date" : ISODate("2021-04-26T08:00:00.000+08:00"),
    "new_cases_smoothed" : 20788
  }
]
]

```

## **6. Relational & Non-Relational Data Model**

### **6.1. Function Difference Between Both Models**

The team noticed a difference between the functions used for the relational data model and the non-relational data model. The functions for the relational data model are simpler in a way that they are generally only used once, while non-relational data model allows multiple usages of similar functions.

For example, in the relational data model, the team only uses the “group by” function at the end of the code to group similar fields together. However, for the non-relational data model, the data can be grouped multiple times. This is similar to the SELECT and \$project function, where relational data model only allows the usage of the SELECT function once (unless it is a sub-query), whereas non-relational data model allows the usage of \$project multiple times within a single query. This may result in some confusion during the learning phase due to the difference in both models.

### **6.2. Data Storage and Data Model Requirements**

To ensure successful data import, the team also had to run through the documents in NoSQL to ensure that the size did not exceed 16 megabytes for each document. The “stop on errors” function was used to ensure successful data implementation into MongoDB. Unlike SQL, where documents could be classified as NULL where no value was included, NoSQL required users to ensure that each record stores a value, regardless of whether they were to be 0 value or otherwise. As the team was given 2 different JSON files for the project for the ‘covid19data’ collection, it was seen that unsuccessful import was derived from the first file. This could mean that the records were not input properly into the first file version (*GeeksforGeeks, 2020*).

NoSQL, however, allows for more flexible data storage, as long as the data can be stored in their respective collections and columns. On the other hand, SQL does not impose a limit for each data storage for the respective document. It can be derived from the project that SQL allows a more structured query due to the presence of normalization and re-categorizing of data into various tables. This creates a relational schema where entity relations were needed to justify the one-to-many or many-to-many relationship and data interrelation using primary keys and foreign keys. Hence, NoSQL requires a more stringent method of data storage and less structured model, while SQL does not require a strict method of data storage while providing a more structured model. (*MongoDB, n.d*)



## 7. Recommendation to WHO

### 7.1. Relational & Non-relational Database Implementation:

- **Language:** NoSQL uses JSON which provides a dynamic schema for unstructured data giving queries and data the ability to be categorized in different manners, SQL used the structured query language which is widely versatile with predefined schemas hence better used for more complex queries such as Q9 and 10 of SQL query. Queries will lag on NoSQL databases due to the complexity of data retrieval. This is because the data is broken down in SQL through the use of joins, queries and updates allowing the backend database to perform querying at a faster speed in compared to NoSQL. *(GeeksforGeeks, 2020)*
- **Scalability:** SQL vertical scalable option for data that is structured, allowing for tables to be created, however, data retrieval needs to be done in reference to other tables which can be complicated if there is an increased number of tables involved. NoSQL provides a horizontal scaling option, allowing for traffic to be shared across various servers, increasing stability of the database for large or changing datasets. *(GeeksforGeeks, 2020)*
- **Schema design:** NoSQL provides a more flexible database design, with the lack of tables allowing data to be stored in collections where collections do not have to be interrelated unlike SQL. NoSQL is better suited to run management systems, big data applications and real-time analytics due to its ability to incorporate a huge amount of unstructured data in the database. SQL is limited by the design and tables involved. SQL is better suited for structured data or applications requiring multi-row transactions such as accounting systems or legacy(old) systems built for a relational structure. *(GeeksforGeeks, 2020)*

### 7.2. ACID Compliance Between SQL and NoSQL Databases:

ACID compliance is a set of properties of database transactions that reduces anomalies and protects the integrity of the database. Transaction is a set of database operations that satisfy ACID properties. ACID stands for:

- **Atomicity:** It means that the transaction either happens completely or doesn't happen at all.
- **Consistency:** A consistent transaction will not violate integrity constraints placed on the data by the database rules. Enforcing consistency ensures that if a violation of data integrity constraints occurs the process will be aborted, and changes rolled back to their previous, legal state.
- **Isolation:** Any reads or writes performed on the database will not be impacted by other reads and writes of separate transactions occurring on the same database. It means that multiple transactions can happen concurrently without reading the wrong data.

- **Durability:** It ensures that changes made to the database (transactions) that are successfully committed will survive permanently, even in the case of system failures.

The SQL databases are ACID compliant. They make use of a system called “locking” to keep the database on hold while the transaction takes place. When a transaction is taking place the data that the transaction is working with is locked until the transaction is finished. In this way, concurrent transactions cannot work on the data already being used. Once the transaction begins it can either run completely or it can fail (due to an error) (*Watts Stephen, 2020*).

On the other hand, NoSQL databases generally sacrifice ACID compliance for scalability and processing speed. (*Geek Culture, 2021*)

### 7.3. Skill Set Available:

#### 1. Quantitative Measure

- As database requirements change overtime, the use of NoSQL has risen considerably. On popular database that uses NoSQL is MongoDB. Currently, MongoDB has **more than 26,800 customers** in more than 100 countries. The MongoDB database platform has been downloaded over 175 million times and there have been more than 1.5 million MongoDB University registrations (*MongoDB, n.d*). As such, with these figures in mind, it can be said that there is a skilled pool of individuals in the market who can utilise MongoDB.
- Based on a survey conducted by ScaleGrid, hundreds of developers, engineers, software architects, developer teams, and IT leaders were interviewed to discover the current NoSQL v. SQL usage. Based on the results it was found that MySQL dominated the report with a 38.9% usage rate, followed by MongoDB at 24.6% (*ScaleGrid, 2019*). From this survey the team can see that SQL is still the most common database platform that is being used, as such, WHO may find it easier to scout for SQL professionals compared to NoSQL professionals.

#### 2. Qualitative Measure

- SQL is a mature technology with many experienced vendors and developers, who can assist with large scale deployments. Independent consultants are available due to the history and experience of SQL in establishing their user and consumer database, with the historical usage of SQL in databases. The availability across vendors is higher for SQL, with availability across most major platforms from operating systems to architecture and programming language. The SQL language is also proprietary, leaving a gap in the community benefitting from open systems. (*Talend., n.d.*)
- NoSQL is newer and more reliant on community support with limited experts in the field, as it is considered relatively new with communities being more fractured. However, the NoSQL community is more focused on onboarding users due to the open system format, increasing prospective user uptake. Due to the lack of understanding of NoSQL, limited help is available in deployment of the database for large scale schema-less operations. (*GeeksforGeeks, 2020*) In addition, compatibility varies widely for NoSQL, requiring dependencies to be investigated more carefully.

#### **7.4. Final Recommendation:**

WHO (World Health Organization) monitors health situation, trends, progress and performance of health systems. In the wake of COVID-19, data will be invaluable to assess the impact on health workforce capacity and essential services to improve preparedness going forward. (*World Health Organization, n.d.*)

The databases that the team were provided in this project were very structured. As the team mentioned earlier SQL is better suited for structured databases. In addition, on the basis of data implementations carried out (specifically Q9 and Q10 for SQL implementation and Q19 and Q20 for NoSQL implementation) the team can observe that multi-row operations are required for observing trends. SQL is better suited for implementing multi-row operations.

Though NoSQL is better suited on the basis of real-time data analysis, COVID-19 data is updated once daily hence there is no need to handle real-time data.

SQL is ACID compliant which provides additional security to the databases, ensuring that while carrying out data analysis the database will not be compromised (that is, the integrity of the database is protected while carrying out database operations).

Moreover, on the basis of skill-set the team observed that though there is an increasing pool of individuals with the ability to utilize NoSQL, SQL is still one of the most commonly implemented languages and hence individuals with SQL proficiency may be found more easily than NoSQL. Taking a more qualitative measure, SQL is well developed and has a more established platform (in terms of experts and professionals) compared to NoSQL.

Hence, based on the analysis, the team has chosen to recommend SQL (Relational model) to WHO.

## 8. References

Farzan A.N. (18 March 2019). Tiny Gibraltar has vaccinated its adult population, British health official says. Accessed 26 October 2021, Retrieved from: <https://www.washingtonpost.com/world/2021/03/18/gibraltar-vaccine-coronavirus/>

Liptak K., Collins K. (10 September 2021). Biden announces new vaccine mandates that could cover 100 million Americans. Accessed 27 October 2021, Retrieved from: <https://edition.cnn.com/2021/09/09/politics/joe-biden-covid-speech/index.html>

Schiffing S., Phelan C. (28 September 2021). What the world can learn from Bhutan's rapid COVID vaccine rollout. Accessed 27 October 2021, Retrieved from: <https://theconversation.com/what-the-world-can-learn-from-bhutans-rapid-covid-vaccine-rollout-168341>

Norwegian News Agency. (24 June 2021). Sky-high estimates were not high enough: Almost everybody says yes to the Covid vaccine in Norway. Accessed 27 October 2021, Retrieved from: <https://sciencenorway.no/covid19-vaccines-virus/sky-high-estimates-were-not-high-enough-almost-everybody-says-yes-to-the-covid-vaccine-in-norway/1880146>

Krug P. (2 Oct 2019). The Normalization Spectrum. Accessed 26 October 2021, Retrieved from: <https://dzone.com/articles/the-normalization-spectrum>

Hopkins J.S., Uribe A. (11 October 2021). Pfizer-BioNTech Covid-19 Vaccine Is World's Preferred Shot. Accessed 28 October 2021, Retrieved from: <https://www.wsj.com/articles/pfizer-biontech-covid-19-vaccine-is-worlds-preferred-shot-11633950181#:~:text=Driving%20the%20popularity%2C%20government%20officials,AstraZeneca%20and%20Johnson%20%26%20Johnson%20shots>

Li J., Tan N., Fang Y. (11 June 2021). More in S'pore opting for Pfizer Covid-19 vaccine despite longer wait than for Moderna. Accessed 27 October 2021, Retrieved from: <https://www.straitstimes.com/singapore/covid-19-more-opting-for-pfizer-vaccine-despite-longer-wait-than-for-moderna>

McDonald J., Wu HZ. (11 April 2021). Top Chinese official admits vaccines have low effectiveness. Accessed 28 October 2021, Retrieved from: <https://apnews.com/article/china-gao-fu-vaccines-offer-low-protection-coronavirus-675bcb6b5710c7329823148ffbff6ef9>

Reuters. (1 April 2021). CanSinoBIO says COVID-19 shot may be less effective over time, booster shot promising. Accessed 28 October 2021, Retrieved from: <https://www.reuters.com/article/us-health-coronavirus-vaccine-cansinobio-idUSKBN2BO4CG>

Dutta P.K. (9 June 2021). Why WHO approval is not enough for Chinese Covid-19 vaccines. Accessed 28 October 2021, Retrieved from:

<https://www.indiatoday.in/coronavirus-outbreak/story/who-approval-chinese-covid-19-vaccines-sinovac-sinopharm-1812733-2021-06-09>

MongoDB. (n.d.). MongoDB Limits and Thresholds. Accessed 28 October 2021, retrieved from: <https://docs.mongodb.com/manual/reference/limits/>

GeeksforGeeks. (15 September 2020) SQL vs NoSQL: Which one is better to use?. Accessed 28 October 2021, Retrieved from: <https://www.geeksforgeeks.org/sql-vs-nosql-which-one-is-better-to-use/>

Co C. (24 March 2021). COVID-19 vaccination now open to Singapore residents aged 45 to 59. Accessed 28 October 2021. Retrieved from: <https://www.channelnewsasia.com/singapore/covid-19-vaccination-younger-age-groups-277536>

ASEAN Information Center. (2 July 2021). Accessed 28 October 2021. Retrieved from: [http://www.aseanhai.net/english/ewt\\_news.php?nid=4077&filename=index](http://www.aseanhai.net/english/ewt_news.php?nid=4077&filename=index)

G. Justin. (4 February 2020). What's an ACID compliant database? Accessed 28 October 2021, retrieved from: <https://retool.com/blog/whats-an-acid-compliant-database/>

Watts Stephen. (24 April 2020). ACID Explained: Atomic, Consistent, Isolated & Durable. Accessed 28 October 2021, retrieved from: <https://www.bmc.com/blogs/acid-atomic-consistent-isolated-durable/>

Geek Culture. (23 June 2021). SQL VS NoSQL. Accessed 28 October 2021. Retrieved from: <https://medium.com/geekculture/sql-vs-nosql-92fa8ec1758b>

World Health Organization. (n.d.) Health Service Data. Accessed 28 October 2021. Retrieved from: <https://www.who.int/data/data-collection-tools/health-service-data>

Talend. (n.d.). SQL vs NoSQL: Differences, Databases and Decisions. Accessed 29 October 2021. Retrieved from: <https://www.talend.com/resources/sql-vs-nosql/>

Scalegrid. (4 March 2019). 2019 Database trends-SQL vs. NoSQL, Top Databases, Single vs. Multiple Database Use. Accessed 29 October 2021. Retrieved from: <https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use/>

MongoDB. (n.d.). Company Mission and Vision. Accessed 29 October 2021.

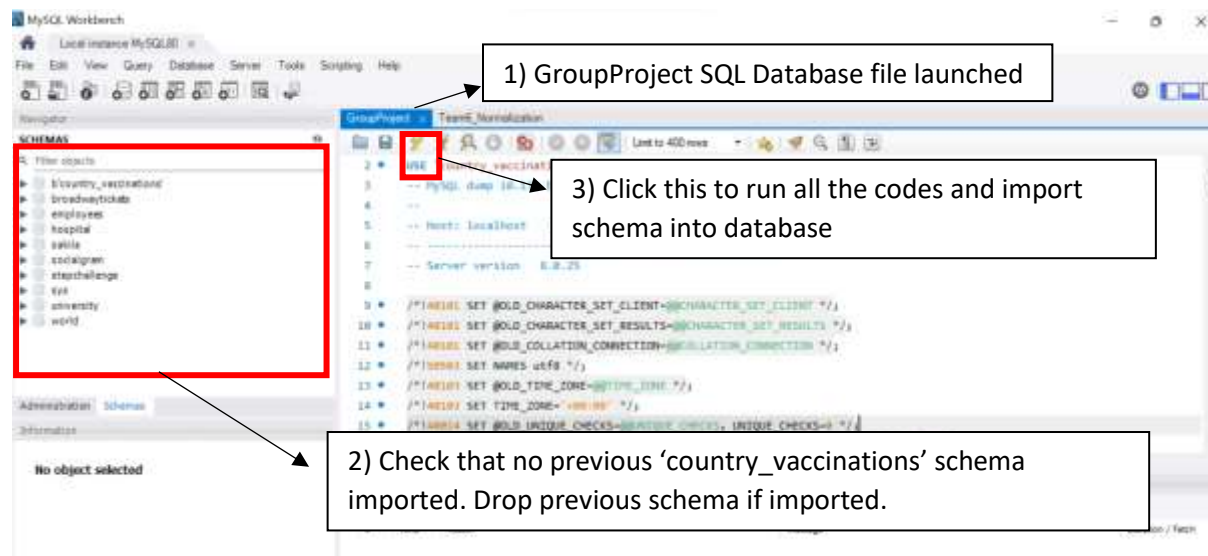
Retrieved from: <https://www.mongodb.com/company#:~:text=MongoDB%20has%20more%20than%2026%2C800,1.5%20million%20MongoDB%20University%20registrations.>



## 9. Appendix

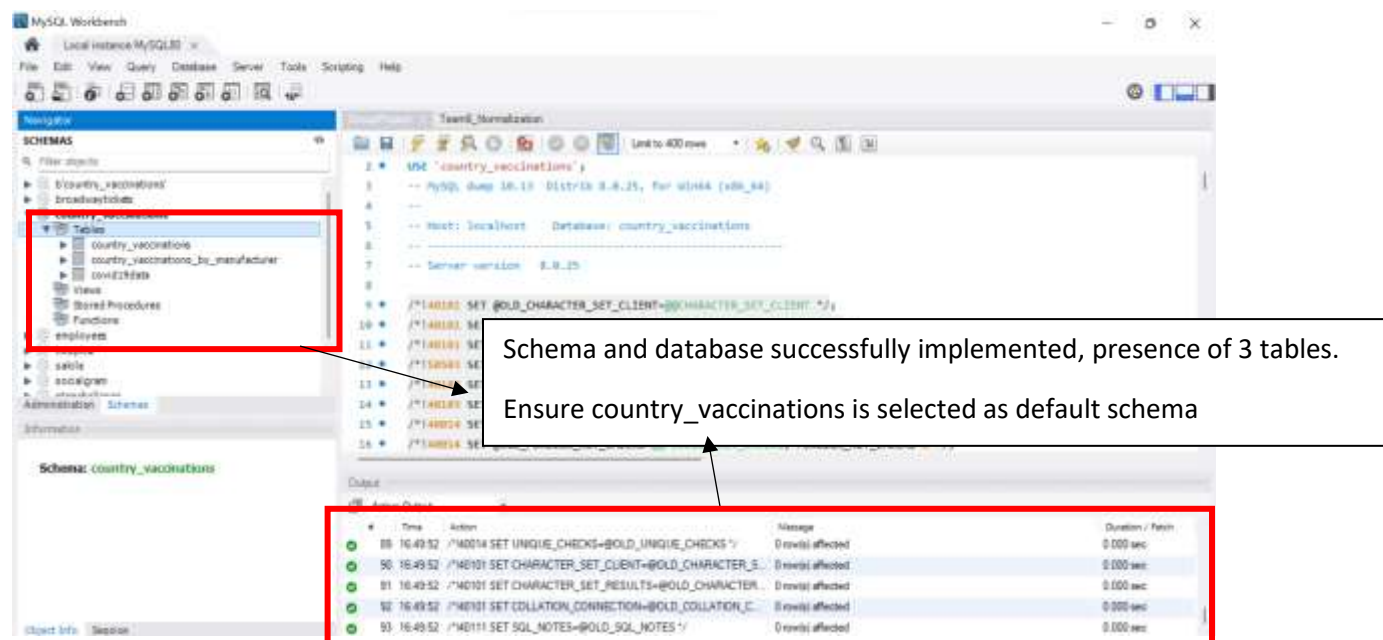
### Appendix A: Steps to import and normalize Relational Database

#### Step1: Launch GroupProject SQL Database File

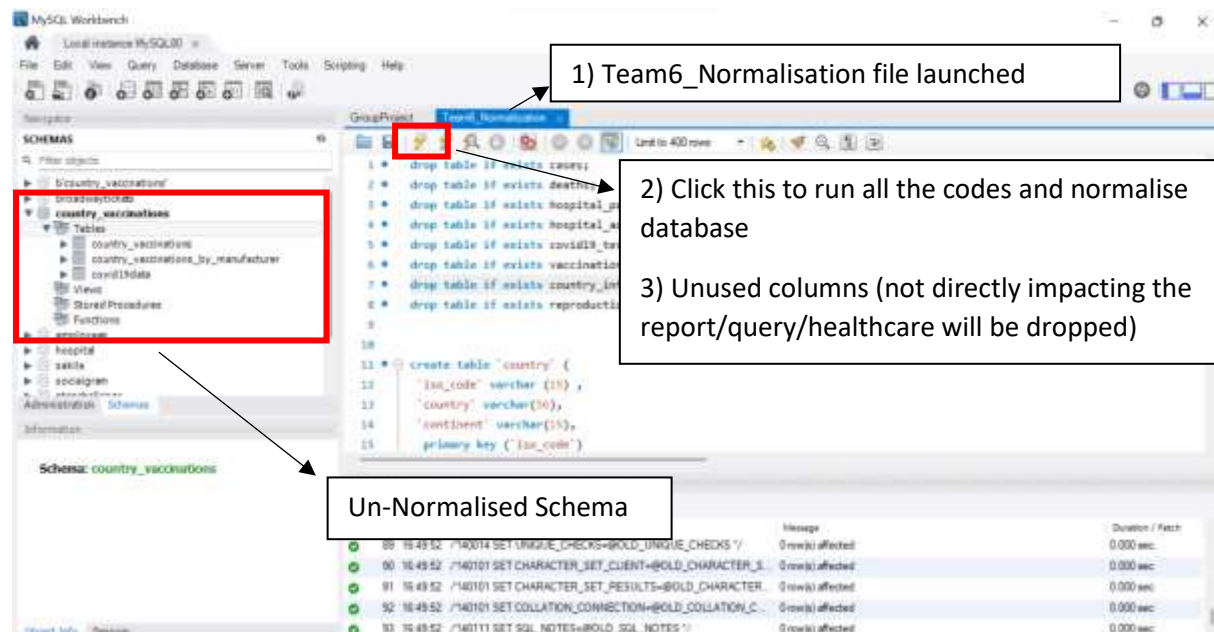


#### Step 2: Ensure database has been imported

- Check for 3 tables:
  - o country\_vaccinations
  - o country\_vaccinations\_by\_manufacturer
  - o covid19data



**Step 3:** Normalise codes and drop columns which are not required in the covid healthcare analysis. Launch SQL Script 'Team6\_Normalisation'



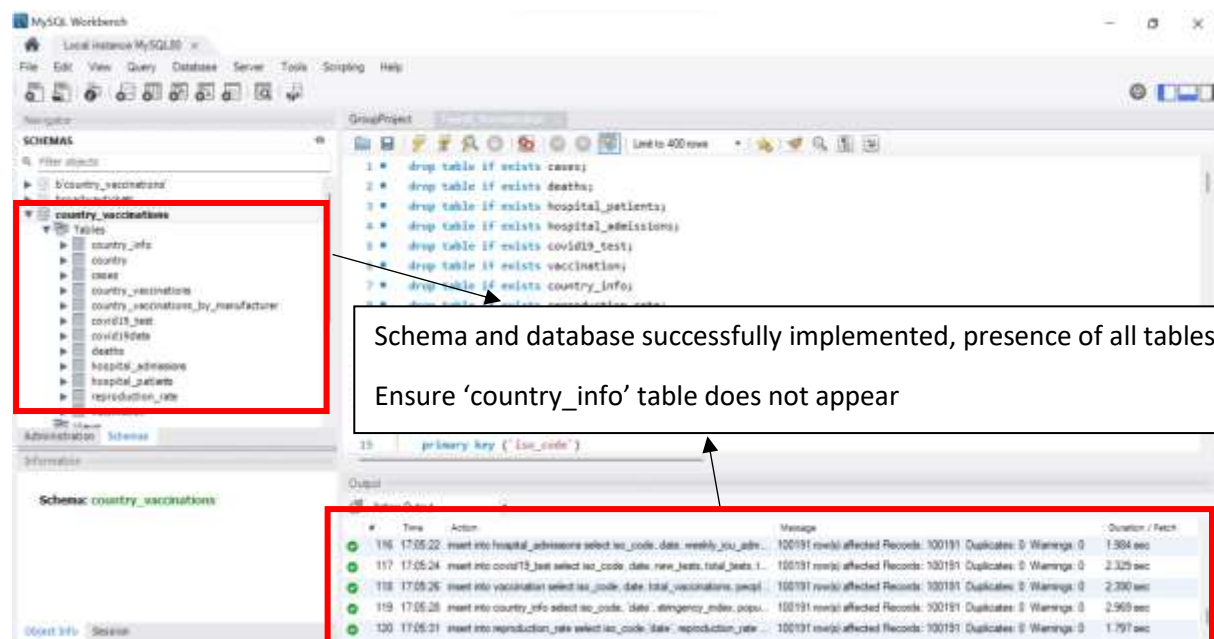
1) Team6\_Normalisation file launched

2) Click this to run all the codes and normalise database

3) Unused columns (not directly impacting the report/query/healthcare will be dropped)

Un-Normalised Schema

**Step 4:** Ensure normalized tables appear and unwanted table 'country\_info' has been dropped



Schema and database successfully implemented, presence of all tables. Ensure 'country\_info' table does not appear

## Appendix B: Entity Relationship Diagrams (ERD) for Relational Database

Figure B.1: ERD for covid19data after Normalisation

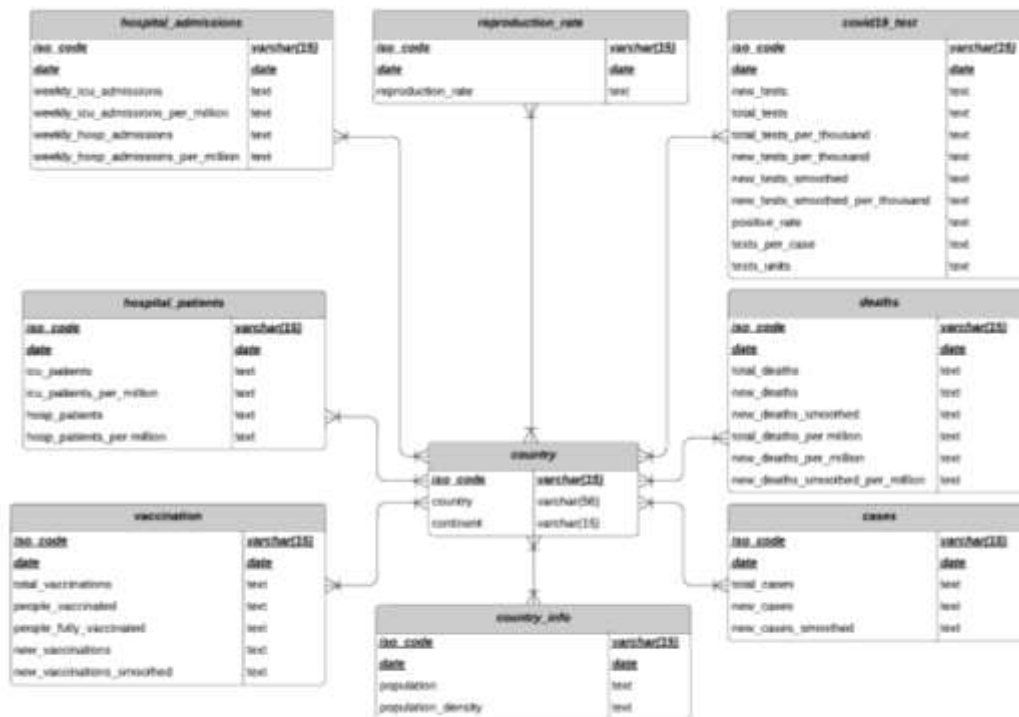


Figure B.2: ERD for country vaccinations by manufacturer (No Normalisation)

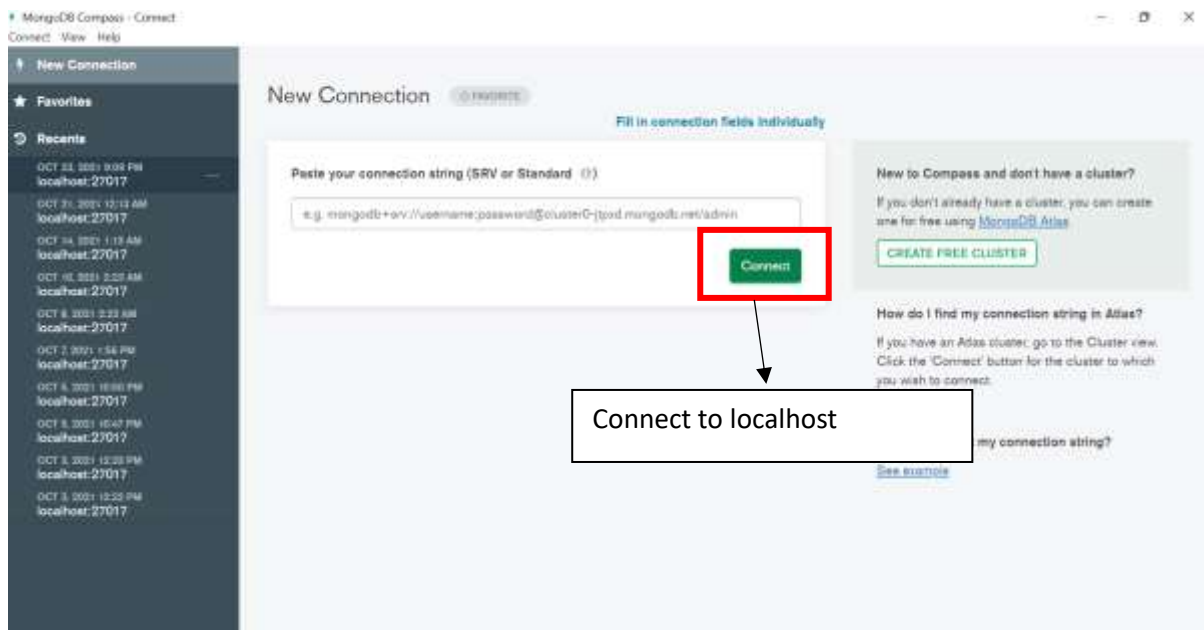


Figure B.3: ERD for country vaccinations (No Normalisation)

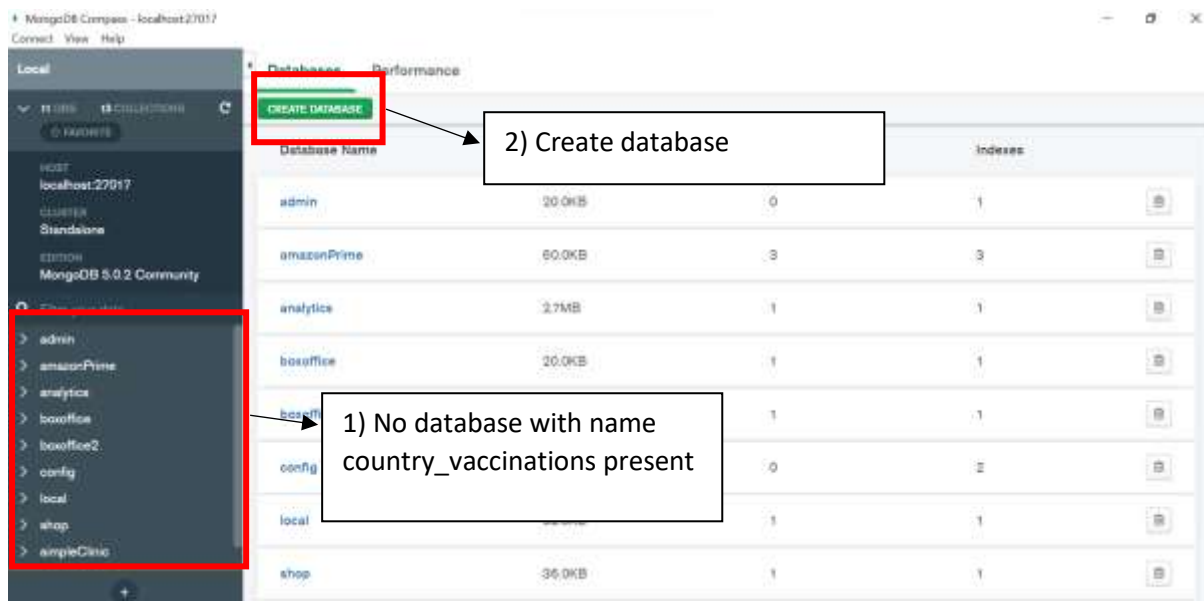


## Appendix C: Steps to import and convert data in Non-Relational Database

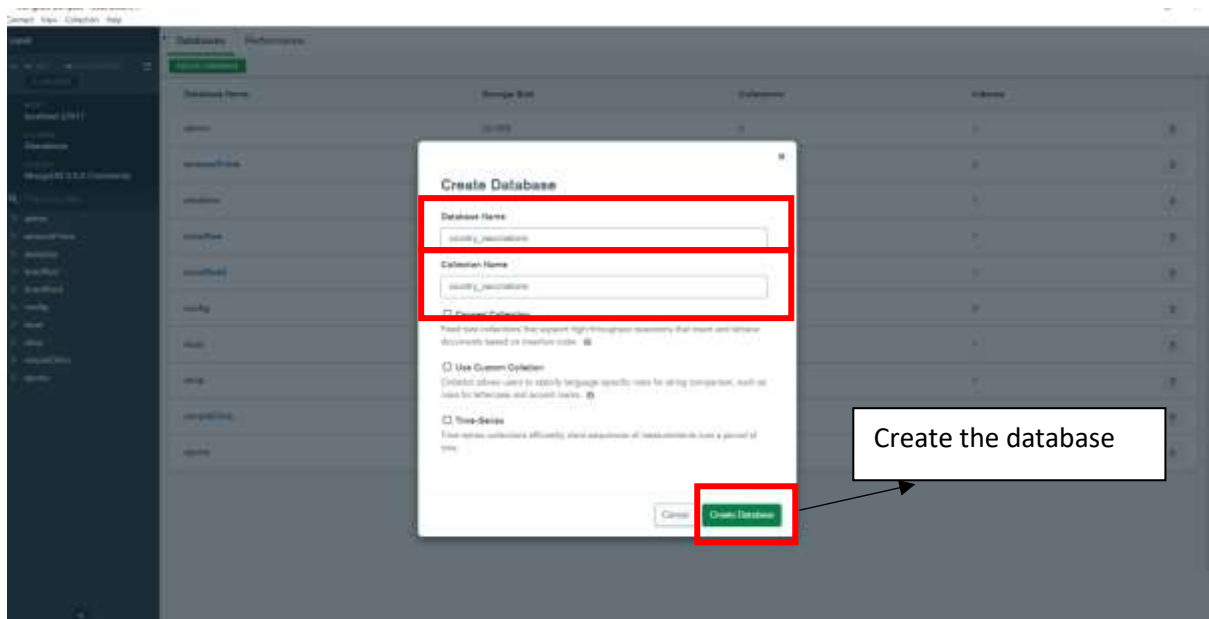
### Step 1: Launch MongoDB and connect to local host



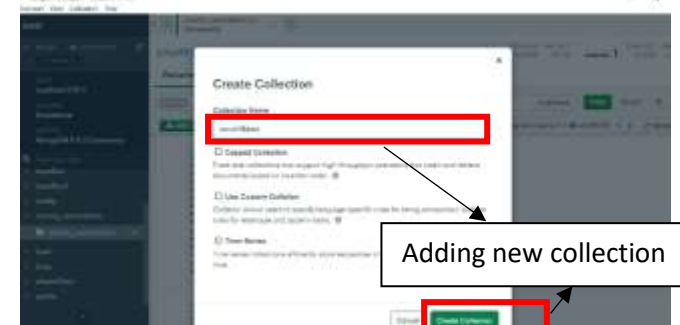
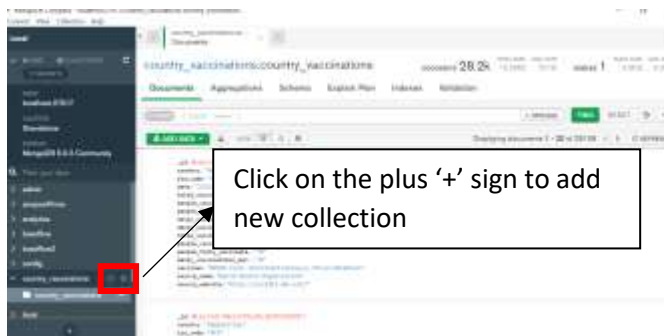
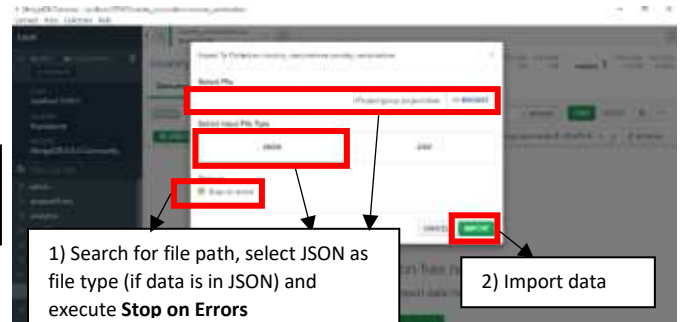
### Step 2: Check that no similar database name “country\_vaccinations” have been implemented (drop if present) and click create database



**Step 3: Create database and import data into MongoDB (Set country\_vaccinations as both database name and first collection name)**



Find the database created and begin importing data by clicking on the first collection named country\_vaccinations.



Repeat importing steps for collections 'country\_vaccinations\_by\_manufacturer' and 'covid19data'.



Step 4: Launch NoSQL Booster, open SQL Script 'Team 6\_NoSQL Data Conversion.js' and run data conversion codes



## Appendix D: Data conversion codes in Non-Relational Database

Figure D.1: Data conversion for 'country\_vaccinations' collection

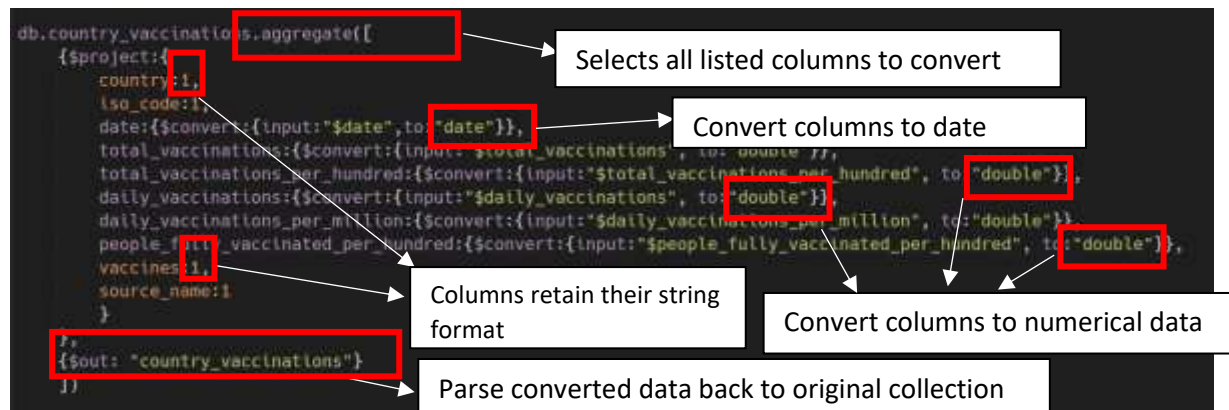


Figure D.2: Data conversion for 'covid19data' collection

