

NANYANG TECHNOLOGICAL UNIVERSITY

NANYANG BUSINESS SCHOOL

BC2402 Designing and Developing Databases



Project Title:

COVID'19

The New Normal - Vaccinations and Re-opening

Project Submitted by Seminar 5 Group 6:

Yap Wen Tong Genevieve (U2010909A)

Sng Yi Xuan (U2021009B)

Sydney Teo Wen Xuen (U2021555B)

Shanais Yuen Ziqing (U2011433A)

Yam Hui Jing (U2021656A)

Contents

Contents	2
1. Introduction	3
2. Relational Data Model	3
2.1 Introduction to Relational Data Model (RDBMS)	3
2.2 Entity Relationship Diagram (ERD)	4
2.2 Table Creation	6
2.3 Justification for new approach	13
2.3.1 Advantages of Normalisation	15
2.4 SQL Queries	16
3. Non-relational Data Model	25
3.1 Introduction to non-relational data model (Non-RDBMS)	25
3.2 Collection Creation	26
2.4 NoSQL Queries	29
4. Difference between Relational and Non-relational Models	49
4.1 Characteristics	49
4.2 Brewer's CAP Theorem	51
5. Recommendations to WHO	53
References	55
Appendix	55
MySQL Queries (non-normalised)	56

1. Introduction

In this project, we designed and implemented relational and non-relational databases to analyse three different Covid-19 datasets, namely “country_vaccinations”, “country_vaccinations_by_manufacturer” and “covid19data” respectively. These databases are organized collections of information, and data pertaining to Covid-19, where we can perform structured and non-structured queries to obtain these data and conduct analyses to better understand the data. This allowed us to better measure the vaccination driver and its effectiveness by tracking the vaccination and the number of covid cases in the respective countries.

The aim of this project report is to illustrate the considerations we have while designing both relational and non-relational data models that incorporate the three datasets used in the project.

After which, we will critically evaluate the advantages and disadvantages of the relational and non-relational database implementations in terms of process, people, and platform. After which, we will then make a recommendation to the World Health Organisation (WHO) as to which database model would be the most suitable and optimal for them. The aim is to recommend a model that will help WHO better manage and analyse data from these datasets, enabling them to better come up with effective solutions to combat the Covid-19 situation.

2. Relational Data Model

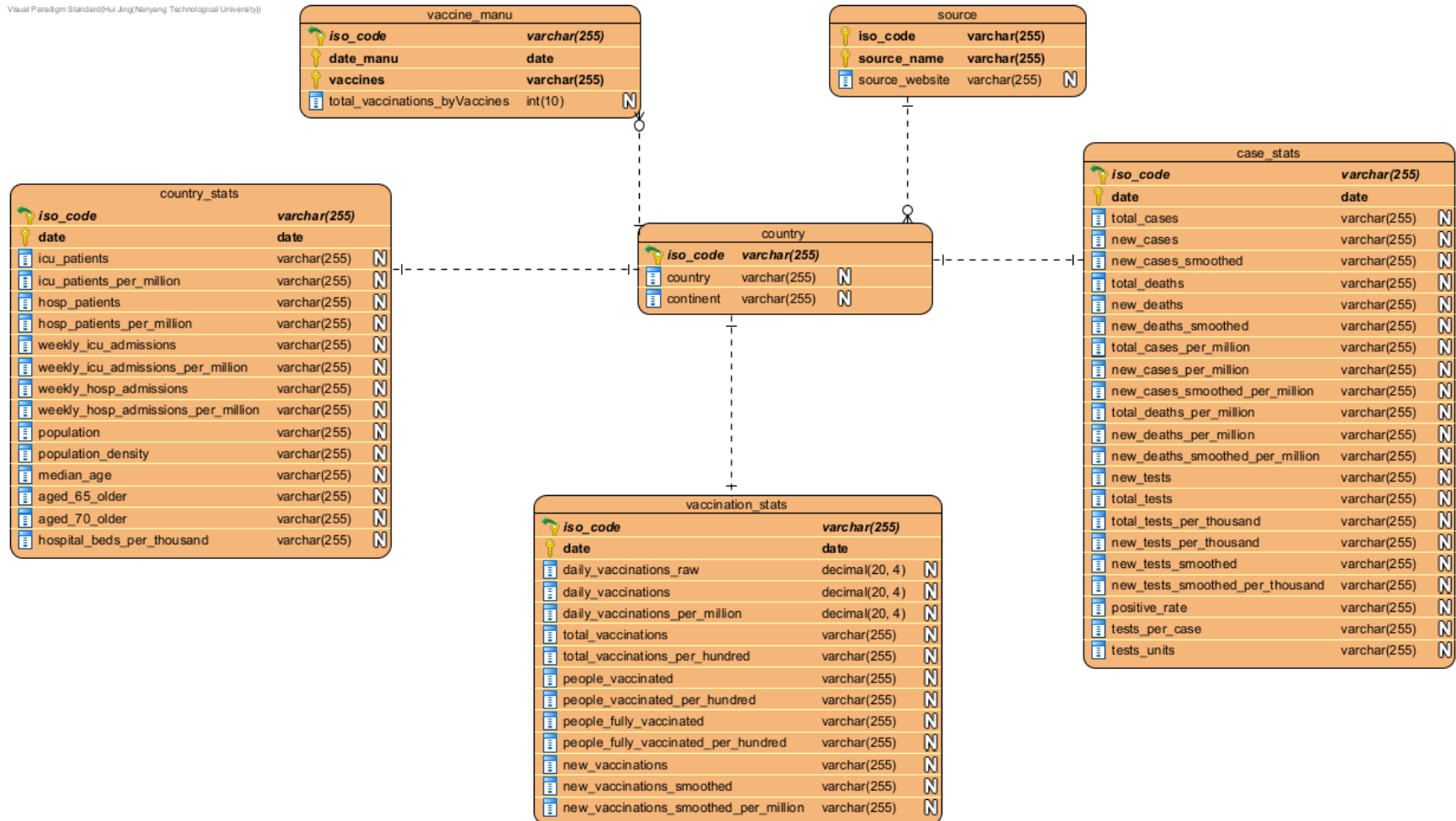
2.1 Introduction to Relational Data Model (RDBMS)

A relational database is structured, meaning the data is organized in tables. It is also called SQL databases. SQL stands for Structured Query Language and it's the language relational databases are written in. SQL is used to execute queries, retrieve data, and edit data by updating, deleting, or creating new records. (Power BI Report, n.d.)

Many times, the data within these tables have relationships with one another, or dependencies. The relationship between each data point is clear and searching through those relationships is relatively easy. The relationship between tables and field types is called a schema. The table itself would be made up really of one variable or object that we would be looking through. The column would represent the data point itself that needs to be stored and the row is a record of the data points per column. In this structure, all queries would be related to this table and the structure of the table would allow for easy sorting, filtering, computations, etc. Key is created to allow for connections to be made between two or more tables to solidify associations between the two.

2.2 Entity Relationship Diagram (ERD)

Visual Paradigm Standard@hui.jing(Nanyang Technological University)



- Normalisation has been done on the three original tables, namely *country_vaccinations*, *country_vaccinations_by_manufacturer*, and *covid19data*, constructing the ER Diagram above consisting of six tables which are all inter-linked. As the database deals with data from three different data sources, it is assumed that DB design is expected to connect all three databases.
- Since this is more of a reporting/analytics database, no real time data transactions are expected. Country table, *vaccine_manu* table and *source* table are constructed in 3NF while the rest of the tables are in 2NF as there are still transitive dependencies. Should there be data transactions in the future, all the insert, update, and delete anomalies will need to be resolved.
- Tables constructed (*country*, *vaccine_manu*, *source*, *case_stats*, *vaccination_stats*, *country_stats*) were implemented to resolve the insert/update/delete anomalies by mapping each item to iso_code.
- A dashed line is a "Non-Identifying Relationship" where a foreign key in a table is not part of the primary key such as date in the *vaccination_stats* entity.
- There are one-to-one relationships from *country* entity to *country_stats*, *vaccination_stats* and *case_stats* entities as each country has only one statistic on country, vaccination and case.
- There is a one-to-many relationship from *country* entity to *vaccine_manu* entity as each country administers more than 1 vaccine type.
- There is a many-to-one relationship from *country* entity to *source* entity as there are sources that provide multiple countries' information.

2.2 Table Creation

Country Table

Country Table contains each country's iso_code and the continent it belongs to. It uses a primary key, which is iso_code.

To resolve insert/delete/update anomalies, the *Country* Table was created to map each country to a unique iso_code.

Table: country

<i>Column Name</i>	<i>Data Type</i>	<i>Description</i>
<u>iso_code</u>	text - PK	ISO 3166-1 alpha-3 – three-letter country codes
country	text	Geographical location
continent	text	Continent of the geographical location

Vaccine_Manu Table

Vaccine_Manu Table contains vaccines administered by each country. It uses a composite primary key made up of iso_code, date_manu and vaccines.

Total_vaccinations in *country_vaccinations_by_manufacturer* has been renamed as total_vaccinations_byVaccines in *vaccine_manu* table to avoid confusion.

Table: vaccine_manu

<i>Column Name</i>	<i>Data Type</i>	<i>Description</i>
<u>iso_code</u>	text - PK-FK	ISO 3166-1 alpha-3 – three-letter country codes
<u>date_manu</u>	text - PK	Date of observation
<u>vaccines</u>	text - PK	Vaccine type
total_vaccinations_by Vaccine	int(10)	Total number of COVID-19 vaccination doses administered

Vaccination_Stats Table

Vaccination_Stats Table contains vaccination statistics for each country. It uses a composite primary key made up of iso_code and date.

The team observed that there are repetitive attributes such as people_vaccinated and total_vaccinations in both *country_vaccinations* and *covid19data* table. Hence, we decide to merge them and present them under *vaccination_stats* table by using the values in covid19data as it contains more updated information.

The rationale of including total_vaccinations under *vaccination_stats* table despite total_vaccinations_byVaccines is because total_vaccinations information has more values and will be frequently retrieved by MOH for analysis purpose

Table: vaccination_stats

<i>Column Name</i>	<i>Data Type</i>	<i>Description</i>
<u>iso_code</u>	text - PK-FK	ISO 3166-1 alpha-3 – three-letter country codes
<u>date</u>	text - PK	Date of observation
daily_vaccinations_raw	decimal(20,4)	For a certain data entry, the number of vaccination for that date/country
daily_vaccinations	decimal(20,4)	For a certain data entry, the number of vaccination for that date/country
daily_vaccinations_per_million	decimal(20,4)	Ratio (in ppm) between vaccination number and total population for the current date in the country
total_vaccinations	text	Total number of COVID-19 vaccination doses administered
total_vaccinations_per_hundred	text	Total number of COVID-19 vaccination doses administered per 100 people in the total population

people_vaccinated	text	Total number of people who received at least one vaccine dose
people_vaccinated_per_hundred	text	Total number of people who received at least one vaccine dose per 100 people in the total population
people_fully_vaccinated	text	Total number of people who received all doses prescribed by the vaccination protocol
people_fully_vaccinated_per_hundred	text	Total number of people who received all doses prescribed by the vaccination protocol per 100 people in the total population
new_vaccinations	text	New COVID-19 vaccination doses administered (only calculated for consecutive days)
new_vaccinations_smoothed	text	New COVID-19 vaccination doses administered (7-day smoothed). For countries that don't report vaccination data on a daily basis, we assume that vaccination changed equally on a daily basis over any periods in which no data was reported. This produces a complete series of daily figures, which is then averaged over a rolling 7-day window
new_vaccinations_smoothed_per_million	text	New COVID-19 vaccination doses administered (7-day smoothed) per 1,000,000 people in the total population

Case Stats Table

Case_Stats Table contains the daily case statistics for each country. It uses a composite primary key made up of iso_code and date.

Table: case_stats

<i>Column Name</i>	<i>Data Type</i>	<i>Description</i>
<u>iso_code</u>	text - PK	ISO 3166-1 alpha-3 – three-letter country codes
<u>date</u>	text - PK	Date of observation

total_cases	text	Total confirmed cases of COVID-19
new_cases	text	New confirmed cases of COVID-19
new_cases_smoothed	text	New confirmed cases of COVID-19 (7-day smoothed)
total_deaths	text	New confirmed cases of COVID-19 (7-day smoothed)
new_deaths	text	New deaths attributed to COVID-19
new_deaths_smoothed	text	New deaths attributed to COVID-19 (7-day smoothed)
total_cases_per_million	text	Total confirmed cases of COVID-19 per 1,000,000 people
new_cases_per_million	text	New confirmed cases of COVID-19 per 1,000,000 people
new_cases_smoothed_per_million	text	New confirmed cases of COVID-19 per 1,000,000 people
total_deaths_per_million	text	Total deaths attributed to COVID-19 per 1,000,000 people
new_deaths_per_million	text	New deaths attributed to COVID-19 per 1,000,000 people
new_deaths_smoothed_per_million	text	New deaths attributed to COVID-19 (7-day smoothed) per 1,000,000 people
new_tests	text	New tests for COVID-19 (only calculated for consecutive days)
total_tests	text	Total tests for COVID-19
total_tests_per_thousa	text	Total tests for COVID-19 per 1,000 people

nd		
new_tests_per_thousand	text	New tests for COVID-19 per 1,000 people
new_tests_smoothed	text	New tests for COVID-19 (7-day smoothed). For countries that don't report testing data on a daily basis, we assume that testing changed equally on a daily basis over any periods in which no data was reported. This produces a complete series of daily figures, which is then averaged over a rolling 7- day window
new_tests_smoothed_per_thousand	text	New tests for COVID-19 (7-day smoothed) per 1,000 people
positive_rate	text	The share of COVID-19 tests that are positive, given as a rolling 7-day average (this is the inverse of tests_per_case)
tests_per_case	text	Tests conducted per new confirmed case of COVID-19, given as a rolling 7-day average (this is the inverse of positive_rate)
tests_units	text	Units used by the location to report its testing data

Country_Stats Table

Country_Stats Table contains countries' statistics that are relevant to Covid'19 situation. It uses a composite primary key made up of iso_code and date.

The attributes which the team deem not fulfilling the purpose of MOH to analyse the Covid'19 situation are removed. They are gdp_per_capita, extreme_poverty, cardiovasc_death_rate, diabetes_prevalence, etc.

Table: country_stats

<i>Column Name</i>	<i>Data Type</i>	<i>Description</i>
<u>iso_code</u>	text - PK	ISO 3166-1 alpha-3 – three-letter country codes

<u>date</u>	text	Date of observation
icu_patients	text	Number of COVID-19 patients in intensive care units (ICUs) on a given day
icu_patients_per_million	text	Number of COVID-19 patients in intensive care units (ICUs) on a given day per 1,000,000 people
hosp_patients	text	Number of COVID-19 patients in hospital on a given day
hosp_patients_per_million	text	Number of COVID-19 patients in hospital on a given day per 1,000,000 people
weekly_icu_admissions	text	Number of COVID-19 patients newly admitted to intensive care units (ICUs) in a given week
weekly_icu_admissions_per_million	text	Number of COVID-19 patients newly admitted to intensive care units (ICUs) in a given week per 1,000,000 people
weekly_hosp_admissions	text	Number of COVID-19 patients newly admitted to hospitals in a given week
weekly_hosp_admissions_per_million	text	Number of COVID-19 patients newly admitted to hospitals in a given week per 1,000,000 people
population	text	Population in 2020
population_density	text	Number of people divided by land area, measured in square kilometers, most recent year available
median_age	text	Median age of the population, UN projection for 2020
aged_65_older	text	Median age of the population, UN projection for 2020
aged_70_older	text	Share of the population that is 70 years and older in 2015

hospital_beds_per_tho usand	text	Hospital beds per 1,000 people, most recent year available since 2010
--------------------------------	------	--

Source Table

Source Table contains the sources that provide information for each country. It uses a primary key made up of source_name and a foreign key which is iso_code.

Table: source

<i>Column Name</i>	<i>Data Type</i>	<i>Description</i>
<u>source_name</u>	text - PK	Source of the information (national authority, international organization, local organization etc.)
source_website	text	Website of the source of information
<u>iso_code</u>	text - PK-FK	ISO 3166-1 alpha-3 – three-letter country codes

2.3 Justification for new approach

The objective of normalization is to isolate data so that when changes are made in one table, such as deletion and insertion. The effects will be multiplied through the other information base (or tables) by methods for the described associations.

We aim to achieve 2NF/ 3NF which uses normalizing principles to reduce the duplication of data, avoid data anomalies, ensure referential integrity, and simplify data management.

In the creation of our normalized tables, we ensured referential integrity between our tables by establishing foreign and primary keys relations between our tables. Below is an example where we created a constraint `country_stats_fk` where it references to the iso_code primary key in the country's table.

```
CREATE TABLE country_stats (  
    iso_code VARCHAR(255),  
    date DATE,  
    primary key (iso_code, date),  
    `icu_patients` text,  
    `icu_patients_per_million` text,  
    `hosp_patients` text,  
    `hosp_patients_per_million` text,  
    `weekly_icu_admissions` text,  
    `weekly_icu_admissions_per_million` text,  
    `weekly_hosp_admissions` text,  
    `weekly_hosp_admissions_per_million` text,  
    `population` text,  
    `population_density` text,  
    `median_age` text,  
    `aged_65_older` text,  
    `aged_70_older` text,  
    `hospital_beds_per_thousand` text,  
    CONSTRAINT `country_stats_fk`  
        FOREIGN KEY (`iso_code`)  
        REFERENCES `country_vaccinations`.`country`(`iso_code`)  
        ON DELETE CASCADE  
        ON UPDATE CASCADE  
);
```

An example of how referential integrity is established is as follows:

We first perform a SELECT statement on case_stats where iso_code = "DEU"

```
205 • SELECT * FROM country WHERE iso_code = "DEU";
206 • SELECT * FROM case_stats WHERE iso_code = "DEU";
207
```

iso_code	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	total_cases_per_million	new_cases_per_million
DEU	2020-01-06								
DEU	2020-01-18								
DEU	2020-01-24								
DEU	2020-01-27	1.0	1.0					0.012	0.012
DEU	2020-01-28	4.0	3.0					0.048	0.036
DEU	2020-01-29	4.0	0.0					0.048	0.0
DEU	2020-01-30	4.0	0.0					0.048	0.0
DEU	2020-01-31	5.0	1.0					0.06	0.012
DEU	2020-02-01	8.0	3.0	1.143		0.0		0.095	0.036
DEU	2020-02-02	10.0	2.0	1.429		0.0		0.119	0.024
DEU	2020-02-03	12.0	2.0	1.571		0.0		0.143	0.024
DEU	2020-02-04	12.0	0.0	1.143		0.0		0.143	0.0
DEU	2020-02-05	12.0	0.0	1.143		0.0		0.143	0.0
DEU	2020-02-06	12.0	0.0	1.143		0.0		0.143	0.0

Next, we performed an UPDATE on the table country, changing the iso_code from "DEU" to "Germany". As you can observe from the case_stats table, the iso_code in that table has also changed to "Germany".

```
205 • UPDATE country SET iso_code = "GERMANY" WHERE iso_code = "DEU";
206 • SELECT * FROM case_stats WHERE iso_code = "GERMANY";
```

iso_code	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	total_cases_per_million	new_cases_per_million
GERMANY	2020-01-06								
GERMANY	2020-01-18								
GERMANY	2020-01-24								
GERMANY	2020-01-27	1.0	1.0					0.012	0.012
GERMANY	2020-01-28	4.0	3.0					0.048	0.036
GERMANY	2020-01-29	4.0	0.0					0.048	0.0
GERMANY	2020-01-30	4.0	0.0					0.048	0.0
GERMANY	2020-01-31	5.0	1.0					0.06	0.012
GERMANY	2020-02-01	8.0	3.0	1.143		0.0		0.095	0.036
GERMANY	2020-02-02	10.0	2.0	1.429		0.0		0.119	0.024
GERMANY	2020-02-03	12.0	2.0	1.571		0.0		0.143	0.024
GERMANY	2020-02-04	12.0	0.0	1.143		0.0		0.143	0.0
GERMANY	2020-02-05	12.0	0.0	1.143		0.0		0.143	0.0
GERMANY	2020-02-06	12.0	0.0	1.143		0.0		0.143	0.0

Here we show an additional table vaccine_manu where the iso_code has also been changed to "Germany"

```
205 • UPDATE country SET iso_code = "GERMANY" WHERE iso_code = "DEU";
206 • SELECT * FROM case_stats WHERE iso_code = "GERMANY";
207 • SELECT * FROM vaccine_manu WHERE iso_code = "GERMANY";
```

iso_code	date_manu	vaccines	total_vaccinations_byVaccines
GERMANY	2020-12-27	Johnson&Johnson	0
GERMANY	2020-12-27	Moderna	2
GERMANY	2020-12-27	Oxford/AstraZeneca	0
GERMANY	2020-12-27	Pfizer/BioNTech	23319
GERMANY	2020-12-28	Johnson&Johnson	0
GERMANY	2020-12-28	Moderna	2
GERMANY	2020-12-28	Oxford/AstraZeneca	0
GERMANY	2020-12-28	Pfizer/BioNTech	41137
GERMANY	2020-12-29	Johnson&Johnson	0
GERMANY	2020-12-29	Moderna	2
GERMANY	2020-12-29	Oxford/AstraZeneca	0
GERMANY	2020-12-29	Pfizer/BioNTech	90900
GERMANY	2020-12-30	Johnson&Johnson	0
GERMANY	2020-12-30	Moderna	2
GERMANY	2020-12-30	Oxford/AstraZeneca	0

This reduced redundancy as information only needs to be inserted or updated once. The Covid dataset is highly dynamic and WHO is likely to have many entries every day. If WHO were to be storing the same information several times in different tables, it would lead to wastage of storage space and an increase in the total size of the data stored. Additionally, more time and effort would be required to update these inconsistencies as changes have to be done on each table separately. Therefore, relations should be normalized so that when relations in a database are to be altered during the lifetime of the database, we do not lose information or introduce inconsistencies, whilst at the same time, increasing speed and efficiency.

2.3.1 Advantages of Normalisation

There are many advantages to normalization. Here, we have categorized it into three categories.

Speed and Efficiency

A smaller database can be maintained as normalization eliminates the duplicate data. Therefore, this results in a reduction in overall size of the database. Normalization also helps maintain better performance. As the database becomes smaller in size, the passes through the data becomes faster and shorter thereby improving response time and speed.

Data Consistency

The primary goal of database normalization is to reduce data redundancy and improve data integrity. An additional advantage of normalization is an increase in data consistency. The basic normalization technique is the decomposition of database schema: divide larger tables into smaller ones and link them using relationships. Besides that, users would only need to join the tables that are needed for their queries. This ensures that there are no redundant rows of data when querying.

Connections to other systems

A database normalization process is essential for enabling the implementation of any data management software system, such as a product information management (PIM) tool. With good basic organization, installing this system is quicker and easier, and it can easily be linked to the company's data sources without delays or the need to correct synchronization problems.

Practically, WHO will find it much easier to maintain the databases it already has and to make any new additions. It's also faster to connect data sources to any internal or external system, as no revisions will be necessary to ensure that the data sent is correct. On top of that, security is increased, since normalization ensures data can be more accurately located.

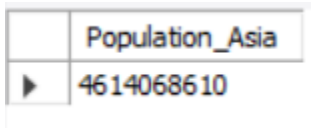
2.4 SQL Queries

In this section, we aim to highlight the approaches we took to tackle some of the questions or queries that WHO could have when handling the data pertaining to Covid19. Here, the goal is to accentuate the ease of usage of SQL and to showcase the benefits it could bring to WHO.

As there are many questions, with some being relatively simple, we hence aim to explain only those questions that are generally harder to obtain or questions that are interesting in the context of Covid19. We will explain the steps we took and the thought process we had when approaching the question.

The following are the questions that we will be explaining:

- [Question 1](#)
- [Question 5, 6, 7](#)
- [Question 8](#)
- [Question 9](#)
- [Question 10](#)

Question 1	What is the total population in Asia?
Query	<pre># Question 1 • SELECT sum(pop) AS Population_Asia FROM (SELECT MAX(population) as pop FROM country_stats WHERE iso_code IN (SELECT DISTINCT(iso_code) FROM country WHERE continent = "Asia") GROUP BY iso_code) as t1;</pre>
Output	
Explanation	Subquery t1 provides a temporary list of countries in the continent Asia and their most updated population figures for MySQL to perform the outer query on. From this list, we can calculate the total population in Asia using <i>SUM(population)</i> as the outer query. We need to do this because our tables are now normalized and the iso_code and population are in different tables of country and country_stats respectively.

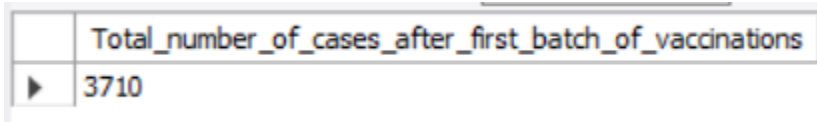
Question 2	What is the total population among the ten ASEAN countries?				
Query	<pre># Question 2 • SELECT SUM(DISTINCT(population)) AS total_population_ASEAN FROM country_stats WHERE iso_code IN ('BRN', 'IDN', 'KHM', 'LAO', 'MMR', 'MYS', 'PHL', 'SGP', 'THA', 'VNM');</pre>				
Output	<table border="1"> <thead> <tr> <th></th><th>total_population_ASEAN</th></tr> </thead> <tbody> <tr> <td>▶</td><td>667301412</td></tr> </tbody> </table>		total_population_ASEAN	▶	667301412
	total_population_ASEAN				
▶	667301412				

Question 3	Generate a list of unique data sources (source_name).																
Query	<pre># Question 3 • select distinct(source_name) from source order by source_name;</pre>																
Output	<table><thead><tr><th>source_name</th></tr></thead><tbody><tr><td>Africa Centres for Disease...</td></tr><tr><td>Cayman Islands Government</td></tr><tr><td>Centers for Disease Contro...</td></tr><tr><td>Costa Rican Social Securit...</td></tr><tr><td>COVID-19 Malta Public He...</td></tr><tr><td>COVID-19 Vaccine Informa...</td></tr><tr><td>Directorate General for He...</td></tr><tr><td>Directorate General of Heal...</td></tr><tr><td>Directorate of Health</td></tr><tr><td>Extended Programme for I...</td></tr><tr><td>Extraordinary commissione...</td></tr><tr><td>Federal Office of Public He...</td></tr><tr><td>Finnish Institute for Health...</td></tr><tr><td>Government of Andorra</td></tr><tr><td>Government of Aruba</td></tr></tbody></table> <p>[92 data sources returned]</p>	source_name	Africa Centres for Disease...	Cayman Islands Government	Centers for Disease Contro...	Costa Rican Social Securit...	COVID-19 Malta Public He...	COVID-19 Vaccine Informa...	Directorate General for He...	Directorate General of Heal...	Directorate of Health	Extended Programme for I...	Extraordinary commissione...	Federal Office of Public He...	Finnish Institute for Health...	Government of Andorra	Government of Aruba
source_name																	
Africa Centres for Disease...																	
Cayman Islands Government																	
Centers for Disease Contro...																	
Costa Rican Social Securit...																	
COVID-19 Malta Public He...																	
COVID-19 Vaccine Informa...																	
Directorate General for He...																	
Directorate General of Heal...																	
Directorate of Health																	
Extended Programme for I...																	
Extraordinary commissione...																	
Federal Office of Public He...																	
Finnish Institute for Health...																	
Government of Andorra																	
Government of Aruba																	

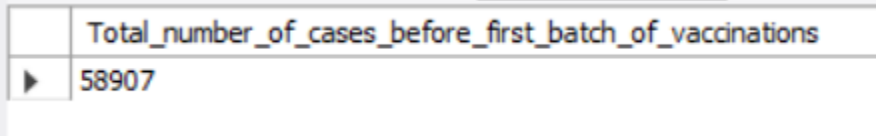
Question 4	Specific to Singapore, display the daily total_vaccinations starting (inclusive) March-1 2021 through (inclusive) May-31 2021.																																													
Query	<pre># Question 4 • SELECT date, total_vaccinations FROM vaccination_stats WHERE iso_code = "SGP" AND date between '2021-03-01' AND '2021-05-31';</pre>																																													
Output	<table><thead><tr><th></th><th>date</th><th>total_vaccinations</th></tr></thead><tbody><tr><td>▶</td><td>2021-03-01</td><td>525039.0</td></tr><tr><td></td><td>2021-03-02</td><td></td></tr><tr><td></td><td>2021-03-03</td><td></td></tr><tr><td></td><td>2021-03-04</td><td>565000.0</td></tr><tr><td></td><td>2021-03-05</td><td></td></tr><tr><td></td><td>2021-03-06</td><td></td></tr><tr><td></td><td>2021-03-07</td><td>596000.0</td></tr><tr><td></td><td>2021-03-08</td><td>611314.0</td></tr><tr><td></td><td>2021-03-09</td><td></td></tr><tr><td></td><td>2021-03-10</td><td></td></tr><tr><td></td><td>2021-03-11</td><td></td></tr><tr><td></td><td>2021-03-12</td><td></td></tr><tr><td></td><td>2021-03-13</td><td></td></tr><tr><td></td><td>2021-03-14</td><td></td></tr></tbody></table> <p>[92 rows of data returned]</p>		date	total_vaccinations	▶	2021-03-01	525039.0		2021-03-02			2021-03-03			2021-03-04	565000.0		2021-03-05			2021-03-06			2021-03-07	596000.0		2021-03-08	611314.0		2021-03-09			2021-03-10			2021-03-11			2021-03-12			2021-03-13			2021-03-14	
	date	total_vaccinations																																												
▶	2021-03-01	525039.0																																												
	2021-03-02																																													
	2021-03-03																																													
	2021-03-04	565000.0																																												
	2021-03-05																																													
	2021-03-06																																													
	2021-03-07	596000.0																																												
	2021-03-08	611314.0																																												
	2021-03-09																																													
	2021-03-10																																													
	2021-03-11																																													
	2021-03-12																																													
	2021-03-13																																													
	2021-03-14																																													

Question 5	5. When is the first batch of vaccinations recorded in Singapore?				
Query	<pre># Question 5 • select min(vs.date) as First_batch_of_vaccinations from vaccination_stats vs where iso_code = "SGP" and vs.total_vaccinations > 0;</pre>				
Output	<table border="1"> <thead> <tr> <th></th><th>First_batch_of_vaccinations</th></tr> </thead> <tbody> <tr> <td>▶</td><td>2021-01-11</td></tr> </tbody> </table>		First_batch_of_vaccinations	▶	2021-01-11
	First_batch_of_vaccinations				
▶	2021-01-11				

Explanation	Use of min() function to find the first date
--------------------	--

Question 6	Based on the date identified in (5), specific to Singapore, compute the total number of new cases thereafter. For instance, if the date identified in (5) is Jan-1 2021, the total number of new cases will be the sum of new cases starting from (inclusive) Jan-1 to the last date in the dataset.
Query	<pre>-- # Question 6 • select sum(cs.new_cases) as Total_number_of_cases_after_first_batch_of_vaccinations from case_stats cs where cs.date >= (select min(vs.date) as First_batch_of_vaccinations from vaccination_stats vs where iso_code = "SGP" and vs.total_vaccinations>0) and iso_code = "SGP";</pre>
Output	
Explanation	We first placed the query which returns the date when the first batch of vaccinations were recorded in Singapore in a subquery. The outer query allows us to find the total number of new cases. We aliased this column as Total_number_of_cases_before_first_batch_of_vaccinations to clearly represent the values returned. This query was performed on the condition that the date of the rows returned came after the date captured in the subquery.

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

Query	<pre># Question 7 • select sum(cs.new_cases) as Total_number_of_cases_before_first_batch_of_vaccinations from case_stats cs where cs.date < (select min(vs.date) as First_batch_of_vaccinations from vaccination_stats vs where iso_code = "SGP" and vs.total_vaccinations>0) and iso_code = "SGP";</pre>
Output	
Explanation	<p>Similar to question 6, however notice that we use < instead as we are finding the number of cases before the first batch of vaccinations.</p>

Question 8	<p>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.</p>
Query	<pre># Question 8 • SELECT t1.date_manu, t1.iso_code, t3.new_cases/t2.population*100 as perc_new_cases, t1.vaccines, t1.total_vaccinations_byVaccines/t2.population*100 as perc_total_vac FROM (SELECT * FROM vaccine_manu WHERE iso_code = "DEU") as t1 LEFT JOIN (SELECT population, date FROM country_stats WHERE iso_code = "DEU") as t2 ON t1.date_manu = t2.date LEFT JOIN (SELECT date, new_cases FROM case_stats WHERE iso_code = "DEU") as t3 on t1.date_manu = t3.date ORDER BY t1.date_manu;</pre>

Output	<table><tr><th>date_manu</th><th>iso_code</th><th>perc_new_cases</th><th>vaccines</th><th>perc_total_vac</th></tr><tr><td>2021-06-18</td><td>DEU</td><td>0.0011135785024207203</td><td>Johnson&Johnson</td><td>1.953764530901475</td></tr><tr><td>2021-06-18</td><td>DEU</td><td>0.0011135785024207203</td><td>Moderna</td><td>6.53219062434933</td></tr><tr><td>2021-06-18</td><td>DEU</td><td>0.0011135785024207203</td><td>Oxford/AstraZeneca</td><td>12.485904071478133</td></tr><tr><td>2021-06-18</td><td>DEU</td><td>0.0011135785024207203</td><td>Pfizer/BioNTech</td><td>57.919967841094135</td></tr><tr><td>2021-06-19</td><td>DEU</td><td>0.0011887719061211548</td><td>Johnson&Johnson</td><td>1.9785950637678855</td></tr><tr><td>2021-06-19</td><td>DEU</td><td>0.0011887719061211548</td><td>Moderna</td><td>6.629977855542609</td></tr><tr><td>2021-06-19</td><td>DEU</td><td>0.0011887719061211548</td><td>Oxford/AstraZeneca</td><td>12.546360761599374</td></tr><tr><td>2021-06-19</td><td>DEU</td><td>0.0011887719061211548</td><td>Pfizer/BioNTech</td><td>58.23859332477122</td></tr><tr><td>2021-06-20</td><td>DEU</td><td>0.0006313858818655531</td><td>Johnson&Johnson</td><td>1.9938485828042591</td></tr><tr><td>2021-06-20</td><td>DEU</td><td>0.0006313858818655531</td><td>Moderna</td><td>6.715012046759078</td></tr><tr><td>2021-06-20</td><td>DEU</td><td>0.0006313858818655531</td><td>Oxford/AstraZeneca</td><td>12.585133106348717</td></tr><tr><td>2021-06-20</td><td>DEU</td><td>0.0006313858818655531</td><td>Pfizer/BioNTech</td><td>58.45997464072622</td></tr><tr><td>2021-06-21</td><td>DEU</td><td>0.000588418222608162</td><td>Johnson&Johnson</td><td>2.036941564401151</td></tr><tr><td>2021-06-21</td><td>DEU</td><td>0.000588418222608162</td><td>Moderna</td><td>6.829455213645049</td></tr><tr><td>2021-06-21</td><td>DEU</td><td>0.000588418222608162</td><td>Oxford/AstraZeneca</td><td>12.685783654613065</td></tr><tr><td>2021-06-21</td><td>DEU</td><td>0.000588418222608162</td><td>Pfizer/BioNTech</td><td>58.934511856657025</td></tr></table> <p>[744 rows returned]</p>	date_manu	iso_code	perc_new_cases	vaccines	perc_total_vac	2021-06-18	DEU	0.0011135785024207203	Johnson&Johnson	1.953764530901475	2021-06-18	DEU	0.0011135785024207203	Moderna	6.53219062434933	2021-06-18	DEU	0.0011135785024207203	Oxford/AstraZeneca	12.485904071478133	2021-06-18	DEU	0.0011135785024207203	Pfizer/BioNTech	57.919967841094135	2021-06-19	DEU	0.0011887719061211548	Johnson&Johnson	1.9785950637678855	2021-06-19	DEU	0.0011887719061211548	Moderna	6.629977855542609	2021-06-19	DEU	0.0011887719061211548	Oxford/AstraZeneca	12.546360761599374	2021-06-19	DEU	0.0011887719061211548	Pfizer/BioNTech	58.23859332477122	2021-06-20	DEU	0.0006313858818655531	Johnson&Johnson	1.9938485828042591	2021-06-20	DEU	0.0006313858818655531	Moderna	6.715012046759078	2021-06-20	DEU	0.0006313858818655531	Oxford/AstraZeneca	12.585133106348717	2021-06-20	DEU	0.0006313858818655531	Pfizer/BioNTech	58.45997464072622	2021-06-21	DEU	0.000588418222608162	Johnson&Johnson	2.036941564401151	2021-06-21	DEU	0.000588418222608162	Moderna	6.829455213645049	2021-06-21	DEU	0.000588418222608162	Oxford/AstraZeneca	12.685783654613065	2021-06-21	DEU	0.000588418222608162	Pfizer/BioNTech	58.934511856657025
date_manu	iso_code	perc_new_cases	vaccines	perc_total_vac																																																																																		
2021-06-18	DEU	0.0011135785024207203	Johnson&Johnson	1.953764530901475																																																																																		
2021-06-18	DEU	0.0011135785024207203	Moderna	6.53219062434933																																																																																		
2021-06-18	DEU	0.0011135785024207203	Oxford/AstraZeneca	12.485904071478133																																																																																		
2021-06-18	DEU	0.0011135785024207203	Pfizer/BioNTech	57.919967841094135																																																																																		
2021-06-19	DEU	0.0011887719061211548	Johnson&Johnson	1.9785950637678855																																																																																		
2021-06-19	DEU	0.0011887719061211548	Moderna	6.629977855542609																																																																																		
2021-06-19	DEU	0.0011887719061211548	Oxford/AstraZeneca	12.546360761599374																																																																																		
2021-06-19	DEU	0.0011887719061211548	Pfizer/BioNTech	58.23859332477122																																																																																		
2021-06-20	DEU	0.0006313858818655531	Johnson&Johnson	1.9938485828042591																																																																																		
2021-06-20	DEU	0.0006313858818655531	Moderna	6.715012046759078																																																																																		
2021-06-20	DEU	0.0006313858818655531	Oxford/AstraZeneca	12.585133106348717																																																																																		
2021-06-20	DEU	0.0006313858818655531	Pfizer/BioNTech	58.45997464072622																																																																																		
2021-06-21	DEU	0.000588418222608162	Johnson&Johnson	2.036941564401151																																																																																		
2021-06-21	DEU	0.000588418222608162	Moderna	6.829455213645049																																																																																		
2021-06-21	DEU	0.000588418222608162	Oxford/AstraZeneca	12.685783654613065																																																																																		
2021-06-21	DEU	0.000588418222608162	Pfizer/BioNTech	58.934511856657025																																																																																		
Explanation	<p>By taking subquery t2 and t3 to be <i>LEFT JOIN</i> to subquery t1, we ensure that we only present non-null rows from the <i>vaccine_manu</i> table, from which we derived the <i>date</i> and <i>iso_code</i> columns for the output. Subquery t2 returns the <i>population</i> figures from <i>country_stats</i>, while subquery t3 returns the <i>new_cases</i> from the <i>case_stats</i> table. These 2 outputs allow us to calculate the percentage of new cases as a proportion of the German population. We joined these 3 subqueries based on a condition which selects all records which have matches in the dates of all 3 tables.</p> <p>→ From the output, we can see the percentage of new cases dropping, as the percentage of total vaccination made in proportion to the population increases. This allows us to see if Germany is on track to achieving Herd Immunity</p>																																																																																					

Question 9	Vaccination Drivers. Specific to Germany, based on each daily new case, display the total vaccinations of each available vaccines after 20 days, 30 days, and 40 days.																																																																																																																																																																																																																	
Query	<pre># Question 9 • SELECT t1.date, t1.new_cases, t2. 20_days, t2.vaccines, t2.max_vac_20days, t3. 30_days, t3.vaccines, t3.max_vac_30days, t4. 40_days, t4.vaccines, t4.max_vac_40days FROM (SELECT date, new_cases FROM case_stats WHERE iso_code = "DEU" ORDER BY date) t1 LEFT JOIN (SELECT iso_code, date_manu as 20_days, vaccines, total_vaccinations_byVaccines AS max_vac_20days FROM vaccine_manu WHERE iso_code = "DEU" order by date_manu asc) t2 on datediff(t2. 20_days, t1.date)=20 LEFT JOIN (SELECT iso_code, date_manu as 30_days, vaccines, total_vaccinations_byVaccines AS max_vac_30days FROM vaccine_manu WHERE iso_code = "DEU" order by date_manu asc) t3 ON (datediff(t3. 30_days, t1.date)=30 AND t3.vaccines = t2.vaccines) LEFT JOIN (SELECT iso_code, date_manu as 40_days, vaccines, total_vaccinations_byVaccines AS max_vac_40days FROM vaccine_manu WHERE iso_code = "DEU" order by date_manu asc) t4 ON (datediff(t4. 40_days, t1.date)=40 AND t4.vaccines = t2.vaccines);</pre>																																																																																																																																																																																																																	
Output	<table><tr><th>date</th><th>new_cases</th><th>20_days</th><th>vaccines</th><th>max_vac_20days</th><th>30_days</th><th>vaccines</th><th>max_vac_30days</th><th>40_days</th><th>vaccines</th><th>max_vac_40days</th></tr><tr><td>2020-12-01</td><td>24766.0</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td></tr><tr><td>2020-12-02</td><td>23275.0</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td></tr><tr><td>2020-12-03</td><td>23591.0</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td></tr><tr><td>2020-12-04</td><td>15970.0</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td></tr><tr><td>2020-12-05</td><td>26126.0</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td></tr><tr><td>2020-12-06</td><td>10910.0</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td></tr><tr><td>2020-12-07</td><td>5456.0</td><td>2020-12-27</td><td>Johnson&Johnson</td><td>0</td><td>2021-01-06</td><td>Johnson&Johnson</td><td>0</td><td>2021-01-16</td><td>Johnson&Johnson</td><td>0</td></tr><tr><td>2020-12-07</td><td>5456.0</td><td>2020-12-27</td><td>Moderna</td><td>2</td><td>2021-01-06</td><td>Moderna</td><td>16</td><td>2021-01-16</td><td>Moderna</td><td>13341</td></tr><tr><td>2020-12-07</td><td>5456.0</td><td>2020-12-27</td><td>Oxford/AstraZeneca</td><td>0</td><td>2021-01-06</td><td>Oxford/AstraZeneca</td><td>5</td><td>2021-01-16</td><td>Oxford/AstraZeneca</td><td>14</td></tr><tr><td>2020-12-07</td><td>5456.0</td><td>2020-12-27</td><td>Pfizer/BioNTech</td><td>23319</td><td>2021-01-06</td><td>Pfizer/BioNTech</td><td>459272</td><td>2021-01-16</td><td>Pfizer/BioNTech</td><td>1144835</td></tr><tr><td>2020-12-08</td><td>29263.0</td><td>2020-12-28</td><td>Johnson&Johnson</td><td>0</td><td>2021-01-07</td><td>Johnson&Johnson</td><td>0</td><td>2021-01-17</td><td>Johnson&Johnson</td><td>0</td></tr><tr><td>2020-12-08</td><td>29263.0</td><td>2020-12-28</td><td>Moderna</td><td>2</td><td>2021-01-07</td><td>Moderna</td><td>19</td><td>2021-01-17</td><td>Moderna</td><td>14602</td></tr><tr><td>2020-12-08</td><td>29263.0</td><td>2020-12-28</td><td>Oxford/AstraZeneca</td><td>0</td><td>2021-01-07</td><td>Oxford/AstraZeneca</td><td>6</td><td>2021-01-17</td><td>Oxford/AstraZeneca</td><td>15</td></tr><tr><td>2020-12-08</td><td>29263.0</td><td>2020-12-28</td><td>Pfizer/BioNTech</td><td>41137</td><td>2021-01-07</td><td>Pfizer/BioNTech</td><td>513416</td><td>2021-01-17</td><td>Pfizer/BioNTech</td><td>1191952</td></tr><tr><td>2020-12-09</td><td>25089.0</td><td>2020-12-29</td><td>Johnson&Johnson</td><td>0</td><td>2021-01-08</td><td>Johnson&Johnson</td><td>0</td><td>2021-01-18</td><td>Johnson&Johnson</td><td>0</td></tr><tr><td>2020-12-09</td><td>25089.0</td><td>2020-12-29</td><td>Moderna</td><td>2</td><td>2021-01-08</td><td>Moderna</td><td>68</td><td>2021-01-18</td><td>Moderna</td><td>18756</td></tr><tr><td>2020-12-09</td><td>25089.0</td><td>2020-12-29</td><td>Oxford/AstraZeneca</td><td>0</td><td>2021-01-08</td><td>Oxford/AstraZeneca</td><td>7</td><td>2021-01-18</td><td>Oxford/AstraZeneca</td><td>16</td></tr><tr><td>2020-12-09</td><td>25089.0</td><td>2020-12-29</td><td>Pfizer/BioNTech</td><td>90900</td><td>2021-01-08</td><td>Pfizer/BioNTech</td><td>575264</td><td>2021-01-18</td><td>Pfizer/BioNTech</td><td>1269903</td></tr></table> <p>[1086 rows returned]</p>	date	new_cases	20_days	vaccines	max_vac_20days	30_days	vaccines	max_vac_30days	40_days	vaccines	max_vac_40days	2020-12-01	24766.0	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	2020-12-02	23275.0	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	2020-12-03	23591.0	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	2020-12-04	15970.0	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	2020-12-05	26126.0	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	2020-12-06	10910.0	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	2020-12-07	5456.0	2020-12-27	Johnson&Johnson	0	2021-01-06	Johnson&Johnson	0	2021-01-16	Johnson&Johnson	0	2020-12-07	5456.0	2020-12-27	Moderna	2	2021-01-06	Moderna	16	2021-01-16	Moderna	13341	2020-12-07	5456.0	2020-12-27	Oxford/AstraZeneca	0	2021-01-06	Oxford/AstraZeneca	5	2021-01-16	Oxford/AstraZeneca	14	2020-12-07	5456.0	2020-12-27	Pfizer/BioNTech	23319	2021-01-06	Pfizer/BioNTech	459272	2021-01-16	Pfizer/BioNTech	1144835	2020-12-08	29263.0	2020-12-28	Johnson&Johnson	0	2021-01-07	Johnson&Johnson	0	2021-01-17	Johnson&Johnson	0	2020-12-08	29263.0	2020-12-28	Moderna	2	2021-01-07	Moderna	19	2021-01-17	Moderna	14602	2020-12-08	29263.0	2020-12-28	Oxford/AstraZeneca	0	2021-01-07	Oxford/AstraZeneca	6	2021-01-17	Oxford/AstraZeneca	15	2020-12-08	29263.0	2020-12-28	Pfizer/BioNTech	41137	2021-01-07	Pfizer/BioNTech	513416	2021-01-17	Pfizer/BioNTech	1191952	2020-12-09	25089.0	2020-12-29	Johnson&Johnson	0	2021-01-08	Johnson&Johnson	0	2021-01-18	Johnson&Johnson	0	2020-12-09	25089.0	2020-12-29	Moderna	2	2021-01-08	Moderna	68	2021-01-18	Moderna	18756	2020-12-09	25089.0	2020-12-29	Oxford/AstraZeneca	0	2021-01-08	Oxford/AstraZeneca	7	2021-01-18	Oxford/AstraZeneca	16	2020-12-09	25089.0	2020-12-29	Pfizer/BioNTech	90900	2021-01-08	Pfizer/BioNTech	575264	2021-01-18	Pfizer/BioNTech	1269903
date	new_cases	20_days	vaccines	max_vac_20days	30_days	vaccines	max_vac_30days	40_days	vaccines	max_vac_40days																																																																																																																																																																																																								
2020-12-01	24766.0	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL																																																																																																																																																																																																								
2020-12-02	23275.0	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL																																																																																																																																																																																																								
2020-12-03	23591.0	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL																																																																																																																																																																																																								
2020-12-04	15970.0	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL																																																																																																																																																																																																								
2020-12-05	26126.0	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL																																																																																																																																																																																																								
2020-12-06	10910.0	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL																																																																																																																																																																																																								
2020-12-07	5456.0	2020-12-27	Johnson&Johnson	0	2021-01-06	Johnson&Johnson	0	2021-01-16	Johnson&Johnson	0																																																																																																																																																																																																								
2020-12-07	5456.0	2020-12-27	Moderna	2	2021-01-06	Moderna	16	2021-01-16	Moderna	13341																																																																																																																																																																																																								
2020-12-07	5456.0	2020-12-27	Oxford/AstraZeneca	0	2021-01-06	Oxford/AstraZeneca	5	2021-01-16	Oxford/AstraZeneca	14																																																																																																																																																																																																								
2020-12-07	5456.0	2020-12-27	Pfizer/BioNTech	23319	2021-01-06	Pfizer/BioNTech	459272	2021-01-16	Pfizer/BioNTech	1144835																																																																																																																																																																																																								
2020-12-08	29263.0	2020-12-28	Johnson&Johnson	0	2021-01-07	Johnson&Johnson	0	2021-01-17	Johnson&Johnson	0																																																																																																																																																																																																								
2020-12-08	29263.0	2020-12-28	Moderna	2	2021-01-07	Moderna	19	2021-01-17	Moderna	14602																																																																																																																																																																																																								
2020-12-08	29263.0	2020-12-28	Oxford/AstraZeneca	0	2021-01-07	Oxford/AstraZeneca	6	2021-01-17	Oxford/AstraZeneca	15																																																																																																																																																																																																								
2020-12-08	29263.0	2020-12-28	Pfizer/BioNTech	41137	2021-01-07	Pfizer/BioNTech	513416	2021-01-17	Pfizer/BioNTech	1191952																																																																																																																																																																																																								
2020-12-09	25089.0	2020-12-29	Johnson&Johnson	0	2021-01-08	Johnson&Johnson	0	2021-01-18	Johnson&Johnson	0																																																																																																																																																																																																								
2020-12-09	25089.0	2020-12-29	Moderna	2	2021-01-08	Moderna	68	2021-01-18	Moderna	18756																																																																																																																																																																																																								
2020-12-09	25089.0	2020-12-29	Oxford/AstraZeneca	0	2021-01-08	Oxford/AstraZeneca	7	2021-01-18	Oxford/AstraZeneca	16																																																																																																																																																																																																								
2020-12-09	25089.0	2020-12-29	Pfizer/BioNTech	90900	2021-01-08	Pfizer/BioNTech	575264	2021-01-18	Pfizer/BioNTech	1269903																																																																																																																																																																																																								
Explanation	<p>Our team noticed the potential of MySQL to create dynamic tables that moved with time, with the <i>LEFT JOIN</i> keyword and a conditional statement that joins the subqueries based on the difference in days across dates captured in various subqueries. To prevent unnecessary duplication of rows, the second condition for the joins specified that only rows where the vaccines are equal to each other should be returned.</p>																																																																																																																																																																																																																	

→ The output allows us to see which vaccine is the main driver out of the 4 vaccines in Germany's vaccination plan, when it comes to reducing the number of new cases.

Question 10

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

Query

```
# Question 10
• SELECT * FROM
  (SELECT date_manu, sum(total_vaccinations_byVaccines) as total_number_of_accumulated_vaccinations
   FROM vaccine_manu
   WHERE iso_code = "DEU"
   GROUP BY date_manu) t1
 LEFT JOIN
  (SELECT date as 21_days, new_cases as daily_new_cases_21days
   FROM case_stats
   WHERE iso_code = "DEU") t2
   on datediff(t2. 21_days, t1.date_manu) = 21
 LEFT JOIN
  (SELECT date as 60_days, new_cases as daily_new_cases_60days
   FROM case_stats
   WHERE iso_code = "DEU") t3
   on datediff(t3. 60_days, t1.date_manu) = 60
 LEFT JOIN
  (SELECT date as 120_days, new_cases as daily_new_cases_120days
   FROM case_stats
   WHERE iso_code = "DEU") t4
   on datediff(t4. 120_days, t1.date_manu) = 120;
```

Output

date	total_number_of_accumulated_vaccinat...	21_days	daily_new_cases_21days	60_days	daily_new_cases_60days	120_days	daily_new_cases_120days
2020-12-27	23321	2021-01-17	11484.0	2021-02-25	11032.0	2021-04-26	5961.0
2020-12-28	41139	2021-01-18	9253.0	2021-02-26	9437.0	2021-04-27	25911.0
2020-12-29	90902	2021-01-19	12233.0	2021-02-27	7671.0	2021-04-28	28263.0
2020-12-30	152687	2021-01-20	29003.0	2021-02-28	6118.0	2021-04-29	24212.0
2020-12-31	202971	2021-01-21	8277.0	2021-03-01	5274.0	2021-04-30	14326.0
2021-01-01	228732	2021-01-22	16366.0	2021-03-02	6492.0	2021-05-01	18535.0
2021-01-02	276370	2021-01-23	12430.0	2021-03-03	10852.0	2021-05-02	8776.0
2021-01-03	298551	2021-01-24	10078.0	2021-03-04	11393.0	2021-05-03	5510.0
2021-01-04	345130	2021-01-25	6887.0	2021-03-05	9581.0	2021-05-04	24111.0
2021-01-05	397029	2021-01-26	9387.0	2021-03-06	8264.0	2021-05-05	22458.0
2021-01-06	459293	2021-01-27	15636.0	2021-03-07	6504.0	2021-05-06	17917.0
2021-01-07	513441	2021-01-28	14883.0	2021-03-08	5129.0	2021-05-07	15090.0
2021-01-08	575339	2021-01-29	12831.0	2021-03-09	6834.0	2021-05-08	13125.0
2021-01-09	634837	2021-01-30	17518.0	2021-03-10	21163.0	2021-05-09	0.0
2021-01-10	669489	2021-01-31	748.0	2021-03-11	4745.0	2021-05-10	7321.0
2021-01-11	736621	2021-02-01	6668.0	2021-03-12	12770.0	2021-05-11	19696.0
2021-01-12	817360	2021-02-02	7690.0	2021-03-13	10568.0	2021-05-12	6590.0
2021-01-13	928980	2021-02-03	12487.0	2021-03-14	8978.0	2021-05-13	13631.0

[186 rows returned]

Explanation

Queries for question 9 and 10 are similar in that we left joined several subqueries based on a conditional statement that specifies the joins to be based on the

difference in days across dates captured in various subqueries using the *DATEDIFF* function. We performed a *SUM* function in the select statement of subquery t1, which calculates the total number of vaccinations made, across the 4 different vaccinations types, on a daily basis.

→ The output produced allows us to see the effect of the number of vaccinations made and how it affects the number of new cases

3. Non-relational Data Model

3.1 Introduction to non-relational data model (Non-RDBMS)

A non-relational database is a database that does not use the tabular schemas of rows and columns found in relational databases. It does not need to have a fixed schema. Also called NoSQL (Non-Structured Query Language) databases, non-relational databases contain various data types such as structured, unstructured, semi-structured and polymorphic data. This ability to store different types of data makes non-relational databases much more flexible than relational databases. In addition, it is a storage model that is optimized for the type of data it's storing. (E., 2021)

There are four different types of NoSQL database types: (Meysman, 2021)

1. Document-oriented databases

- Manages a set of named string fields and object data values in an entity that is referred to as a document
- Designed for storing, retrieving and managing document-oriented information
- Documents are rows, collections are data tables
- All documents stored need not have the same structure (XML, YAML, JSON, BSON or Plain Text)
- A document could contain the entire data for 1 entity.
- Data structures can be shaped based on application requirements
- Use: content management and monitoring web and mobile applications
- Examples: Couchbase Server, CouchDB, MarkLogic and MongoDB

2. Key-Value stores

- Stores data as a unique key-value pair by using an appropriate hashing function, which provides an even distribution of hashed keys across the data storage
- The key is something which you do lookup on to obtain its corresponding value(s).
- The values can be in any format (JSON, BLOB, String, etc.).
- The values can encapsulate a new set of key-value pairs as an object
- Use: session management, caching in web applications (managing shopping cart details for buyers)
- Examples: Aerospike, DynamoDB, Redis and Riak.

3. Columns-Oriented Stores

- Stores data in the form of sets of tables with columns and rows
- Columns can be divided into groups known as column families, which holds a set of columns that are logically related and typically retrieved together, as a unit
- These column families usually contain columns that tend to be queried together
- Use: recommendation engines, catalogs, fraud detection

- Examples: Accumulo, Amazon SimpleDB, Cassandra, HBase and Hypertable

4. Graph stores

- Database that is multi-relational in nature and stores entities and their relationships with each other in a graph like structure
- Supports the analysis of the relationships between entities
- The entity is stored as a node, while the relationships are stored as edges
- Both nodes and edges can have properties that provide information about that node or edge, similar to columns in a table.
- Edges can also have a direction indicating the nature of the relationship.
- Use: social media platforms, reservation systems or customer relationship management.
- Examples: AllegroGraph, IBM Graph and Neo4j. (MongoDB, 2021)

3.2 Collection Creation

In the original NoSQL collections that were provided to the team, we realised that most of the field's data types are not in the correct format. For example, date was stored as a string thus any date manipulation would require us to first convert the string date into a proper date-time format. Secondly, numeric values such as total vaccinations and new cases are also stored as a string datatype. Thus, we will be unable to do arithmetic operations on these fields if we do not first convert them into a numeric datatype. Therefore, to aid in our querying, we first created our own collections with the fields having the correct data type.

Country Vac Collection

This collection is used to store vaccination statistics for each country. The fields like `_id`, `daily_vaccinations` and `source_website` are dropped from the original collection as they are attributes that are not required for the queries. The data types are converted to the appropriate ones for easier analysis. For example, date's data type has been converted to date while `total_vaccinations`' are converted to double.

Collection: `country_vac`

<i>Field Name</i>	<i>Data Type</i>
country	String
iso_code	String
date	date

total_vaccinations	double
people_vaccinated	double
people_fully_vaccinated	double
daily_vaccinations	double
total_vaccinations_per_hundred	double
people_vaccinated_per_hundred	double
people_fully_vaccinated_per_hundred	double
daily_vaccinations_per_million	double
vaccines	String
source_name	String

Country_Vac_Manu Collection

This collection is used to store the vaccines administered by each country for specific date entry. The field like _id is dropped from the original collection as it is an irrelevant attribute. Similarly, date's data type has been converted to date while total_vaccinations' are converted to double. The team also decided to standardise the field name for countries' names as country instead of location.

Collection: country_vac_manu

<i>Field Name</i>	<i>Data Type</i>
country	String
date	date
vaccine	String

total_vaccinations	double
--------------------	--------

Covid19data2 Collection

This collection is used to store the daily case statistics for each country. The fields like _id, stringency_index, population_density, median_age, aged_65_older, etc are dropped from the original collection as they are not required for the execution of queries.

Collection: covid19data2

<i>Field Name</i>	<i>Data Type</i>
country	String
date	date
total_cases	double
new_cases	double
new_cases_smoothed	double
total_cases_per_million	double
new_cases_per_million	double
population	double
iso_code	String
continent	String

2.4 NoSQL Queries

Similar to Section 2.4: SQL Queries, in this section we aim to highlight the approaches we took to tackle some of the questions or queries that WHO could have when handling the data pertaining to Covid19. The goal is to accentuate the ease of usage of NoSQL and to showcase the benefits it could bring to WHO as well as to highlight any potential difference in terms of output as compared to its SQL counterpart.

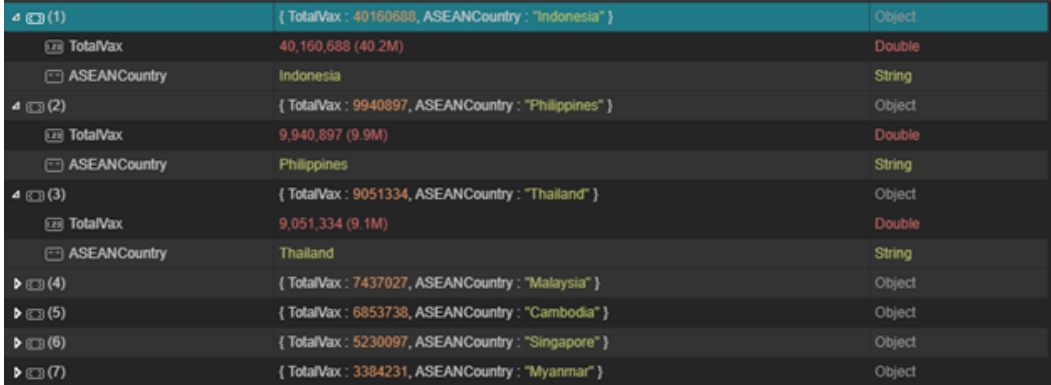
Similar to above, we will only be explaining certain questions that are harder to obtain or are interesting in the context of Covid19. We will explain the steps we took and the thought process we had when approaching the question.

The following are the questions that we will be explaining:

- [Question 5](#)
- [Question 7](#)
- [Question 8](#)
- [Question 9](#)
- [Question 15, 16, 17](#)
- [Question 18](#)
- [Question 19](#)
- [Question 20](#)

Question 1	Display a list of total vaccinations per day in Singapore.
Query	<pre>// Question 1: Display a list of total vaccinations per day in Singapore db.country_vac.aggregate([{\$match:{country:"Singapore"}}, {\$project:{_id:0, country:1,date:1,total_vaccinations:1}}, {\$sort: {date:1}},])</pre>
Output	<div><div><div>▲ (1)</div><div><div>{ country : "Singapore", date : ISODate("2021-01-11T08:00:00.000+08:00"), total_vaccinations</div><div>countrySingaporeString</div><div>date1/11/2021, 8:00:00 AMDate</div><div>total_vaccinations3,400 (3.4K)Double</div></div></div><div>▲ (2)</div><div><div>{ country : "Singapore", date : ISODate("2021-01-12T08:00:00.000+08:00"), total_vaccinations</div><div>countrySingaporeString</div><div>date1/12/2021, 8:00:00 AMDate</div><div>total_vaccinations6,200 (6.2K)Double</div></div></div> <div>▲ (3)</div> <div><div>{ country : "Singapore", date : ISODate("2021-01-13T08:00:00.000+08:00"), total_vaccinations</div><div>countrySingaporeString</div><div>date1/13/2021, 8:00:00 AMDate</div><div>total_vaccinations0Double</div></div>

[169 documents returned]

Question 2	Display the sum of daily vaccinations among ASEAN countries.
Query	<pre>// Question 2: Display the sum of daily vaccinations among ASEAN countries. db.country_vac.aggregate([{\$match:{country:{\$in:['Brunei', 'Cambodia', 'Indonesia', 'Laos', 'Malaysia', 'Myanmar', 'Philippines', 'Singapore', 'Thailand', 'Vietnam']}}}, {\$project: {_id: 0, country: 1, TotalVax: {\$convert: {input: "\$daily_vaccinations" , to : "double"}}}}, {\$group: {_id:{\$groupByCountry:"\$country"},TotalVax:{\$sum:"\$TotalVax"}}}, {\$project: {_id:0,ASEANCountry:"\$_id.groupByCountry",TotalVax:1}}, {\$sort:{TotalVax:-1}}])</pre>
Output	 <p>[10 documents returned]</p>

Question 3	Identify the maximum daily vaccinations per million of each country. Sort the list based on daily vaccinations per million in a descending order.
Query	<pre>// Question 3: Identify the maximum daily vaccinations per million of each country. // Sort the list based on daily vaccinations per million in a descending order. db.country_vac.aggregate([{\$group: {_id:{\$groupByCountry:"\$country"}, maxDailyVac:{\$max:"\$daily_vaccinations_per_million"}}}, {\$project: {_id:0, country:"\$_id.groupByCountry", maxDailyVac:1}}, {\$sort:{maxDailyVac:-1}}])</pre>

Output	<div> <div> (1) </div> <div> { maxDailyVac : 118759, country : "Bhutan" } </div> <div>Object</div> </div> <div> <div>maxDailyVac</div> <div>118,759 (0.12M)</div> <div>Double</div> </div> <div> <div>country</div> <div>Bhutan</div> <div>String</div> </div>
	<div> <div> (2) </div> <div> { maxDailyVac : 54264, country : "Falkland Islands" } </div> <div>Object</div> </div> <div> <div>maxDailyVac</div> <div>54,264 (54.3K)</div> <div>Double</div> </div> <div> <div>country</div> <div>Falkland Islands</div> <div>String</div> </div>
	<div> <div> (3) </div> <div> { maxDailyVac : 46231, country : "Cook Islands" } </div> <div>Object</div> </div> <div> <div>maxDailyVac</div> <div>46,231 (46.2K)</div> <div>Double</div> </div> <div> <div>country</div> <div>Cook Islands</div> <div>String</div> </div>
	[217 documents returned]

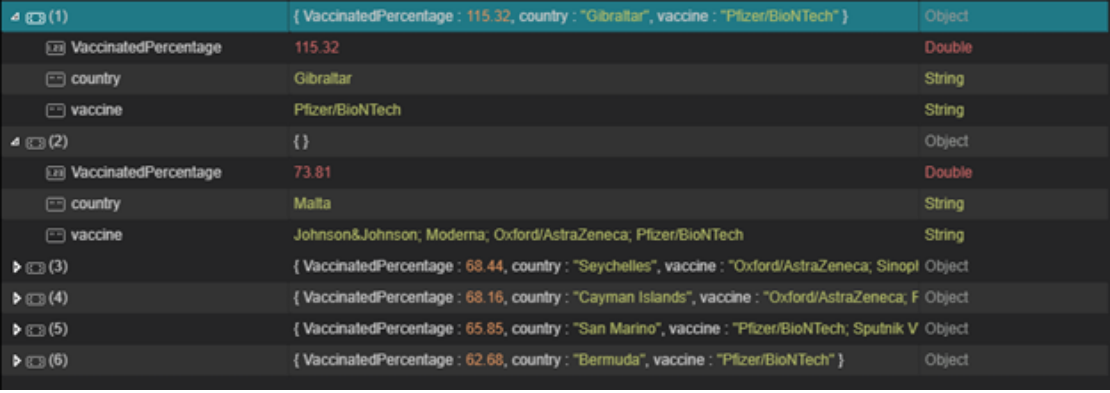
Question 4	Which is the most administrated vaccine? Display a list of total administration (i.e., sum of total vaccinations) per vaccine.
Query	<pre>// Question 4: Which is the most administrated vaccine? Display a list of total administration (i.e., sum of total vaccinations) per vaccine. db.country_vac_manu.aggregate([{ \$group: { _id: { groupByCountry: "\$country", groupByVac: "\$vaccine" }, TotalVac: { \$sum: "\$total_vaccinations" } } }, { \$project: { _id: 0, Country: "\$_id.groupByCountry", Vaccine: "\$_id.groupByVac", TotalVac: 1 } }, { \$group: { _id: { groupByVac2: "\$Vaccine" }, TotalAdministered: { \$sum: "\$TotalVac" } } }, { \$project: { _id: 0, Vaccine: "\$_id.groupByVac2", TotalAdministered: 1 } }, { \$sort: { TotalAdministered: -1 } }])</pre>
Output	<div> <div> (1) </div> <div> { TotalAdministered : 26727111273, Vaccine : "Pfizer/BioNTech" } </div> <div>Object</div> </div> <div> <div>TotalAdministered</div> <div>26,727,111,273 (26.7G)</div> <div>Double</div> </div> <div> <div>Vaccine</div> <div>Pfizer/BioNTech</div> <div>String</div> </div> <div> <div> (2) </div> <div> { TotalAdministered : 13770017464, Vaccine : "Moderna" } </div> <div>Object</div> </div> <div> <div>TotalAdministered</div> <div>13,770,017,464 (13.8G)</div> <div>Double</div> </div> <div> <div>Vaccine</div> <div>Moderna</div> <div>String</div> </div> <div> <div> (3) </div> <div> { TotalAdministered : 2200684190, Vaccine : "Oxford/AstraZeneca" } </div> <div>Object</div> </div> <div> <div>TotalAdministered</div> <div>2,200,684,190 (2.2G)</div> <div>Double</div> </div> <div> <div>Vaccine</div> <div>Oxford/AstraZeneca</div> <div>String</div> </div>
	[6 documents returned]

Question 5	Italy has commenced administrating various vaccines to its populations as a vaccine becomes available. Identify the first dates of each vaccine being administrated, then compute the difference in days between the earliest date and the 4 th date.
------------	--

Query	<pre>db.country_vac_manu.aggregate({\$match: {country:"Italy"}}, {\$group: {_id:{groupByVaccine: "\$vaccine"}, firstAdminstratedDate:{\$min: "\$date"}}}, {\$sort: {firstAdminstratedDate:1}}, {\$project: {_id:0,Vaccine: "\$_id.groupByVaccine",firstAdminstratedDate:1}},)</pre> <pre>db.country_vac_manu.aggregate({\$match: {country:"Italy"}}, {\$group: {_id:{groupByVaccine: "\$vaccine"}, date:{\$min: "\$date"}}}, {\$sort: {firstAdminstratedDate:1}}, {\$project: {_id:0,Vaccine: "\$_id.groupByVaccine",date:1 }}, {\$limit: 4}, {\$group: { _id: null, minDate:{\$max: "\$date"}, maxDate:{\$min: "\$date"}}}, {\$project: {_id:0, Difference_In_Days: {\$dateDiff: { startDate: "\$maxDate", endDate: "\$minDate", unit: "day", }}}}})</pre>									
Output	<div><div><div><div><div>1</div><div>{ firstAdminstratedDate : ISODate("2020-12-27T08:00:00.000+08:00"), Vaccine : "Pfizer/BioNTech" }</div><div>Object</div></div><div><div>firstAdminstratedDate</div><div>12/27/2020, 8:00:00 AM</div><div>Date</div></div><div><div>Vaccine</div><div>Pfizer/BioNTech</div><div>String</div></div></div><div><div>2</div><div>{ firstAdminstratedDate : ISODate("2021-01-06T08:00:00.000+08:00"), Vaccine : "Moderna" }</div><div>Object</div></div><div><div>firstAdminstratedDate</div><div>1/6/2021, 8:00:00 AM</div><div>Date</div></div><div><div>Vaccine</div><div>Moderna</div><div>String</div></div></div><div><div>3</div><div>{ firstAdminstratedDate : ISODate("2021-01-15T08:00:00.000+08:00"), Vaccine : "Oxford/AstraZeneca" }</div><div>Object</div></div><div><div>firstAdminstratedDate</div><div>1/15/2021, 8:00:00 AM</div><div>Date</div></div><div><div>Vaccine</div><div>Oxford/AstraZeneca</div><div>String</div></div></div> <div><div>4</div><div>{ firstAdminstratedDate : ISODate("2021-03-27T08:00:00.000+08:00"), Vaccine : "Johnson&Johnson" }</div><div>Object</div></div> <table><tr><th>Key</th><th>Value</th><th>Type</th></tr><tr><td>1</td><td>{ Difference_In_Days : 90 }</td><td>Object</td></tr><tr><td>Difference_In_Days</td><td>90</td><td>Int64</td></tr></table>	Key	Value	Type	1	{ Difference_In_Days : 90 }	Object	Difference_In_Days	90	Int64
Key	Value	Type								
1	{ Difference_In_Days : 90 }	Object								
Difference_In_Days	90	Int64								
Explanation	<p>Here, the questions requires us to first identify the first date administered for each vaccine. We first filtered the document to match the location to Italy, followed by a group by vaccine type and at the same time, finding the minimum date in each group which will correspond to the “firstAdminstratedDate” of each vaccine. Next, we sort the “firstAdminstratedDate” in ascending order and did a project to only view the vaccine types and its corresponding “firstAdminstratedDate” ,as seen in the first picture.</p> <p>Next, the question requires us to compute the difference between the 4th date and the earliest date. Thus, using the output previously, we next do a limit:4 to only show the 4 earliest dates. Next, using group aggregate of id = null, it allows us to perform aggregate functions like min, max, sum, thus we are able to obtain the min and max date from the list of 4 dates. With the 2 specific dates obtained, using the “dateDiff” feature in nosql, we calculated the difference between these 2 days to obtained the answer of 90 as shown in the second picture.</p>									

Question 6	What is the country with the most types of administered vaccine?																																	
Query	<pre>// Question 6: What is the country with the most types of administered vaccine? db.country_vac_manu.aggregate([{\$group: {_id: {groupByCountry: "\$country", groupByVaccine: "\$vaccine"}}}, {\$group: {_id: {groupByCountry2: "\$_id.groupByCountry"}, vaccine: {\$push: "\$_id.groupByVaccine"}}}, {\$project: {_id: 0, "Country": "\$_id.groupByCountry2", "NumberOfVaccineTypes": {\$size: "\$vaccine"}, "Vaccines": "\$vaccine"}}, {\$sort: {NumberOfVaccineTypes: -1}}, {\$limit: 1}])</pre>																																	
Output	<table><tr><th>Key</th><th>Value</th><th>Type</th></tr><tr><td>4 (1)</td><td>{ Country : "Hungary", NumberOfVaccineTypes : 6 } (3 fields)</td><td>Object</td></tr><tr><td>Country</td><td>Hungary</td><td>String</td></tr><tr><td>NumberOfVaccineTypes</td><td>6</td><td>Int32</td></tr><tr><td>Vaccines</td><td>Array[6]</td><td>Array</td></tr><tr><td>0</td><td>Oxford/AstraZeneca</td><td>String</td></tr><tr><td>1</td><td>Sinopharm/Beijing</td><td>String</td></tr><tr><td>2</td><td>Johnson&Johnson</td><td>String</td></tr><tr><td>3</td><td>Sputnik V</td><td>String</td></tr><tr><td>4</td><td>Moderna</td><td>String</td></tr><tr><td>5</td><td>Pfizer/BioNTech</td><td>String</td></tr></table>	Key	Value	Type	4 (1)	{ Country : "Hungary", NumberOfVaccineTypes : 6 } (3 fields)	Object	Country	Hungary	String	NumberOfVaccineTypes	6	Int32	Vaccines	Array[6]	Array	0	Oxford/AstraZeneca	String	1	Sinopharm/Beijing	String	2	Johnson&Johnson	String	3	Sputnik V	String	4	Moderna	String	5	Pfizer/BioNTech	String
Key	Value	Type																																
4 (1)	{ Country : "Hungary", NumberOfVaccineTypes : 6 } (3 fields)	Object																																
Country	Hungary	String																																
NumberOfVaccineTypes	6	Int32																																
Vaccines	Array[6]	Array																																
0	Oxford/AstraZeneca	String																																
1	Sinopharm/Beijing	String																																
2	Johnson&Johnson	String																																
3	Sputnik V	String																																
4	Moderna	String																																
5	Pfizer/BioNTech	String																																

Question 7	What are the countries that have fully vaccinated more than 60% of its people? For each country, display the vaccines administered.
Query	<pre>db.country_vac.aggregate({\$group: { _id: {groupByCountry: "\$country", groupByVaccine: "\$vaccines"} , VaccinatedPercentage: {\$max: "\$people_fully_vaccinated_per_hundred"} }}, {\$match: {"VaccinatedPercentage": {\$gt: 60}}}, {\$sort: {VaccinatedPercentage: -1}}, {\$project: { _id: 0, country: "\$_id.groupByCountry", vaccine: "\$_id.groupByVaccine", VaccinatedPercentage: 1}},)</pre>










Output	
Explanation	<p>The first step we took when approaching this question is to perform a group aggregate on country and vaccine, followed by a max function on the “people_fully_vaccinated_per_hundred”. This effectively, allows us to find out what is the highest “VaccinatedPercentage” pertaining to each group (country, vaccine). Next, we filter the document to only find those groups with “VaccinatedPercentage” greater than 60% as per the question requirement. Following which, we sort the documents in descending order, showing the country and its vaccines administered, with the highest percentage first. Lastly, we do a project to only view the country, vaccine administered and vaccinated percentage.</p>

Question 8	<p>Monthly vaccination insight – display the monthly total vaccination amount of each vaccine per month in the United States.</p>
Query	<pre> db.country_vac_manu.aggregate([{\$match:{country:"United States"}}, {\$group:{_id:{monthNum:{month:"\$date"},groupbyVaccine:"\$vaccine"},"maxTotalVac":{"\$max":{"total_vaccinations"}}}, {\$group:{_id:{monthByGroup:"\$_id.monthNum"}, Insights:{\$push:{"Vaccine": "\$_id.groupbyVaccine", "monthly_total_vaccine": "\$maxTotalVac"}}}}, {\$project:{_id:0, "Month": "\$_id.monthByGroup", "Insights":1}}, {\$sort:{Month:1}}]) </pre>

Output	<table><thead><tr><th>Key</th><th>Value</th><th>Type</th></tr></thead><tbody><tr><td>(1)</td><td>{ Month : 1 } (2 fields)</td><td>Object</td></tr><tr><td>Month</td><td>1</td><td>Int32</td></tr><tr><td>Insights</td><td>Array[2]</td><td>Array</td></tr><tr><td>0</td><td>{ Vaccine : "Moderna", monthly_total_vacc</td><td>Object</td></tr><tr><td>Vaccine</td><td>Moderna</td><td>String</td></tr><tr><td>monthly_total_vaccine</td><td>14,246,089 (14.2M)</td><td>Double</td></tr><tr><td>1</td><td>{ Vaccine : "Pfizer/BioNTech", monthly_tots</td><td>Object</td></tr><tr><td>Vaccine</td><td>Pfizer/BioNTech</td><td>String</td></tr><tr><td>monthly_total_vaccine</td><td>16,775,916 (16.8M)</td><td>Double</td></tr><tr><td>(2)</td><td>{ Month : 2 } (2 fields)</td><td>Object</td></tr><tr><td>(3)</td><td>{ Month : 3 } (2 fields)</td><td>Object</td></tr><tr><td>(4)</td><td>{ Month : 4 } (2 fields)</td><td>Object</td></tr><tr><td>(5)</td><td>{ Month : 5 } (2 fields)</td><td>Object</td></tr><tr><td>(6)</td><td>{ Month : 6 } (2 fields)</td><td>Object</td></tr></tbody></table>	Key	Value	Type	(1)	{ Month : 1 } (2 fields)	Object	Month	1	Int32	Insights	Array[2]	Array	0	{ Vaccine : "Moderna", monthly_total_vacc	Object	Vaccine	Moderna	String	monthly_total_vaccine	14,246,089 (14.2M)	Double	1	{ Vaccine : "Pfizer/BioNTech", monthly_tots	Object	Vaccine	Pfizer/BioNTech	String	monthly_total_vaccine	16,775,916 (16.8M)	Double	(2)	{ Month : 2 } (2 fields)	Object	(3)	{ Month : 3 } (2 fields)	Object	(4)	{ Month : 4 } (2 fields)	Object	(5)	{ Month : 5 } (2 fields)	Object	(6)	{ Month : 6 } (2 fields)	Object
Key	Value	Type																																												
(1)	{ Month : 1 } (2 fields)	Object																																												
Month	1	Int32																																												
Insights	Array[2]	Array																																												
0	{ Vaccine : "Moderna", monthly_total_vacc	Object																																												
Vaccine	Moderna	String																																												
monthly_total_vaccine	14,246,089 (14.2M)	Double																																												
1	{ Vaccine : "Pfizer/BioNTech", monthly_tots	Object																																												
Vaccine	Pfizer/BioNTech	String																																												
monthly_total_vaccine	16,775,916 (16.8M)	Double																																												
(2)	{ Month : 2 } (2 fields)	Object																																												
(3)	{ Month : 3 } (2 fields)	Object																																												
(4)	{ Month : 4 } (2 fields)	Object																																												
(5)	{ Month : 5 } (2 fields)	Object																																												
(6)	{ Month : 6 } (2 fields)	Object																																												
Explanation	<p>This question is helpful to government and health officials should they want to understand the vaccination coverage as well as the popularity of a particular vaccine in a country across the months. Here, we are interested to know the monthly total vaccinations for each vaccine per month for the United States. As such, we first perform a match on the country United States.</p> <p>Next, we do a group on “monthNum” and vaccines. However, to be able to group by “monthNum” we have to use a nosql inbuild function “\$month” which Returns the month for a date as a number between 1 (January) and 12 . This allows us to effectively be able to group the documents by month as well as the vaccines. In the group by clause, we also perform a max to find the highest total vaccination pertaining to each month for a particular vaccine.</p> <p>However, this output is not ideal as these groups are separated, instead, we want to be able to view the vaccines and its total vaccinations in a single month document. Hence, we perform another group by on the “monthNum”, at the same time, we create array “Insights” and using the \$push function, we push the vaccine and its corresponding monthly total vaccination into the array.</p> <p>Lastly, we project only the month number and the “Insights” and sorted the document in ascending order of the months.</p>																																													
Question 9	<p>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)</p>																																													

Query	<pre>db.country_vac.aggregate([{\$group:{ _id:{groupbyCountry:"\$country"}, "FirstAvailableDate":{\$min:"\$date"}, "DailyPercentages":{\$push:{date:"\$date","VaccinationsPercentage":'\$total_vaccinations_per_hundred'}}}}, {\$project:{ _id:0, FirstAvailableDate:1, "Country": "\$_id.groupbyCountry", DailyPercentages:{\$filter:{ input:"\$DailyPercentages", as: "perc", cond: {\$gt:["\$perc.VaccinationsPercentage",50]} }}}}, {\$project:{\$_id:0,Country:1,"Days_to_over_50%":{\$dateDiff:{ startDate:"\$FirstAvailableDate", endDate:{\$min:"\$DailyPercentages.date"}, unit:"day" }}}}, {\$match:{"Days_to_over_50%":{\$ne:null}}}, {\$sort:{Country:1}}])</pre>
Output	<div><div><div>(1)</div><div>{ Country : "Andorra", "Days_to_over_50%" : 133 }</div><div>Object</div></div><div><div>Country</div><div>Andorra</div><div>String</div></div><div><div>Days_to_over_50%</div><div>133</div><div>Int64</div></div><div><div>(2)</div><div>{ Country : "Anguilla", "Days_to_over_50%" : 102 }</div><div>Object</div></div><div><div>Country</div><div>Anguilla</div><div>String</div></div><div><div>Days_to_over_50%</div><div>102</div><div>Int64</div></div><div><div>(3)</div><div>{ Country : "Antigua and Barbuda", "Days_to_over_50%" : 104 }</div><div>Object</div></div><div><div>Country</div><div>Antigua and Barbuda</div><div>String</div></div><div><div>Days_to_over_50%</div><div>104</div><div>Int64</div></div><div><div>(4)</div><div>{ Country : "Aruba", "Days_to_over_50%" : 19 }</div><div>Object</div></div><div><div>(5)</div><div>{ Country : "Austria", "Days_to_over_50%" : 145 }</div><div>Object</div></div><div><div>(6)</div><div>{ Country : "Bahrain", "Days_to_over_50%" : 104 }</div><div>Object</div></div><div><div>(7)</div><div>{ Country : "Barbados", "Days_to_over_50%" : 111 }</div><div>Object</div></div><div><div>(8)</div><div>{ Country : "Belgium", "Days_to_over_50%" : 143 }</div><div>Object</div></div><div><div>(9)</div><div>{ Country : "Bermuda", "Days_to_over_50%" : 71 }</div><div>Object</div></div><div><div>(10)</div><div>{ Country : "Bhutan", "Days_to_over_50%" : 5 }</div><div>Object</div></div></div> <div>[85 docs returned]</div>
Explanation	<p>Firstly, we perform a group by on countries and using min on date to obtain the first available date for each country. We also created an array called "DailyPercentages" and push each date and its vaccination percentage into it. Next, we perform a project to view only the first available date for each country, the country's name, and the daily vaccination percentages where we filter and present only those where the vaccination percentage is greater than 50%.</p> <p>Next, we perform yet another project, showing the country's name again, but this time showing the "Days_to_over_50%". To do that, we use the in-build nosql function \$dateDiff and find the difference in terms of the number of days between the "firstAvailableDate" and the first date where its corresponding "DailyPercentages" is above 50%.</p> <p>Lastly, to only show those countries that have total vaccinations above the 50% threshold, we do a match on "Days_to_over_50%" not equal to null and sorted the countries in ascending alphabetical order.</p>

Question 10	Compute the global total of vaccinations per vaccine.																																							
Query	<pre>// Question 10: Compute the global total of vaccinations per vaccine. db.country_vac_manu.aggregate([{\$group: {_id: {groupByCountry: "\$country", groupByVax: "\$vaccine"}, TotalVax: {\$max: "\$total_vaccinations"}}}, {\$project: {_id: 0, Country: "\$_id.groupByCountry", Vaccine: "\$_id.groupByVax", TotalVax: 1}}, {\$group: {_id: {groupByVax2: "\$Vaccine"}, TotalAdministered: {\$sum: "\$TotalVax"}}}, {\$project: {_id: 0, Vaccine: "\$_id.groupByVax2", "Global_Total": "\$TotalAdministered"}}, {\$sort: {TotalAdministered: -1}}])</pre>																																							
Output	<table><tr><th>Key</th><th>Value</th><th>Type</th></tr><tr><td>1</td><td>{ Vaccine : "OxfordAstraZeneca", Global_Total : 55469456 }</td><td>Object</td></tr><tr><td> Vaccine</td><td>OxfordAstraZeneca</td><td>String</td></tr><tr><td> Global_Total</td><td>55,469,456 (55.5M)</td><td>Double</td></tr><tr><td>2</td><td>{ Vaccine : "Johnson&Johnson", Global_Total : 20365108 }</td><td>Object</td></tr><tr><td> Vaccine</td><td>Johnson&Johnson</td><td>String</td></tr><tr><td> Global_Total</td><td>20,365,108 (20.4M)</td><td>Double</td></tr><tr><td>3</td><td>{ Vaccine : "SinopharmBeijing", Global_Total : 2056651 }</td><td>Object</td></tr><tr><td>4</td><td>{ Vaccine : "Moderna", Global_Total : 171755596 }</td><td>Object</td></tr><tr><td>5</td><td>{ Vaccine : "Sputnik V", Global_Total : 1813128 }</td><td>Object</td></tr><tr><td>6</td><td>{ Vaccine : "Sinovac", Global_Total : 20366592 }</td><td>Object</td></tr><tr><td>7</td><td>{ Vaccine : "PfizerBioNTech", Global_Total : 488303302 }</td><td>Object</td></tr><tr><td>8</td><td>{ Vaccine : "CanSino", Global_Total : 391012 }</td><td>Object</td></tr></table>	Key	Value	Type	1	{ Vaccine : "OxfordAstraZeneca", Global_Total : 55469456 }	Object	Vaccine	OxfordAstraZeneca	String	Global_Total	55,469,456 (55.5M)	Double	2	{ Vaccine : "Johnson&Johnson", Global_Total : 20365108 }	Object	Vaccine	Johnson&Johnson	String	Global_Total	20,365,108 (20.4M)	Double	3	{ Vaccine : "SinopharmBeijing", Global_Total : 2056651 }	Object	4	{ Vaccine : "Moderna", Global_Total : 171755596 }	Object	5	{ Vaccine : "Sputnik V", Global_Total : 1813128 }	Object	6	{ Vaccine : "Sinovac", Global_Total : 20366592 }	Object	7	{ Vaccine : "PfizerBioNTech", Global_Total : 488303302 }	Object	8	{ Vaccine : "CanSino", Global_Total : 391012 }	Object
Key	Value	Type																																						
1	{ Vaccine : "OxfordAstraZeneca", Global_Total : 55469456 }	Object																																						
Vaccine	OxfordAstraZeneca	String																																						
Global_Total	55,469,456 (55.5M)	Double																																						
2	{ Vaccine : "Johnson&Johnson", Global_Total : 20365108 }	Object																																						
Vaccine	Johnson&Johnson	String																																						
Global_Total	20,365,108 (20.4M)	Double																																						
3	{ Vaccine : "SinopharmBeijing", Global_Total : 2056651 }	Object																																						
4	{ Vaccine : "Moderna", Global_Total : 171755596 }	Object																																						
5	{ Vaccine : "Sputnik V", Global_Total : 1813128 }	Object																																						
6	{ Vaccine : "Sinovac", Global_Total : 20366592 }	Object																																						
7	{ Vaccine : "PfizerBioNTech", Global_Total : 488303302 }	Object																																						
8	{ Vaccine : "CanSino", Global_Total : 391012 }	Object																																						

Question 11	What is the total population in Asia?									
Query	<pre>// Question 11: What is the total population in Asia? db.covid19data2.aggregate([{\$match: {continent: "Asia"}}, {\$group: {_id:{groupByCountry:"\$country"}, population_asia:{\$max: "\$population"}}}, {\$group: {_id:{continent:"Asia"},population:{\$sum:"\$population_asia"}}}])</pre>									
Output	<table><tr><td> (1) { continent : "Asia" }</td><td>{ population : 4614068610 }</td><td>Document</td></tr><tr><td> _id</td><td>{ continent : "Asia" }</td><td>Object</td></tr><tr><td> population</td><td>4,614,068,610 (4.6G)</td><td>Double</td></tr></table>	 (1) { continent : "Asia" }	{ population : 4614068610 }	Document	 _id	{ continent : "Asia" }	Object	 population	4,614,068,610 (4.6G)	Double
 (1) { continent : "Asia" }	{ population : 4614068610 }	Document								
 _id	{ continent : "Asia" }	Object								
 population	4,614,068,610 (4.6G)	Double								

Question 12	What is the total population among the ten ASEAN countries?									
Query	<pre>// Question 12: What is the total population among the ten ASEAN countries? db.covid19data2.aggregate([{\$match:{country:{\$in:["Singapore", "Brunei", "Indonesia", "Myanmar", "Malaysia", "Laos", "Philippines", "Thailand", "Vietnam", "Cambodia"]}}}, {\$group:{_id:{groupByCountry:"\$country"},"TotalPopulation":{\$max:"\$population"}}}, {\$group:{_id:{}, "Total_ASEAN_Population":{\$sum:"\$TotalPopulation"}}}, {\$project:{_id:0, Total_ASEAN_Population:1}},])</pre>									
Output	<table><tr><th>Key</th><th>Value</th><th>Type</th></tr><tr><td>(1)</td><td>{ Total_ASEAN_Population : 667301412 }</td><td>Object</td></tr><tr><td>Total_ASEAN_Population</td><td>667,301,412 (0.67G)</td><td>Double</td></tr></table>	Key	Value	Type	(1)	{ Total_ASEAN_Population : 667301412 }	Object	Total_ASEAN_Population	667,301,412 (0.67G)	Double
Key	Value	Type								
(1)	{ Total_ASEAN_Population : 667301412 }	Object								
Total_ASEAN_Population	667,301,412 (0.67G)	Double								

Question 13	Generate a list of unique data sources (source_name).
Query	<pre>// Question 13: Generate a list of unique data sources (source_name) db.country_vac.distinct("source_name")</pre>

Output	(1)	Africa Centres for Disease Control and Prevention	String
	(2)	COVID-19 Malta Public Health Response Team	String
	(3)	COVID-19 Vaccine Information Platform	String
	(4)	Cayman Islands Government	String
	(5)	Centers for Disease Control and Prevention	String
	(6)	Costa Rican Social Security Fund	String
	(7)	Directorate General for Health via Data Science for Social Good	String
	(8)	Directorate General of Health Services	String
	(9)	Directorate of Health	String
	(10)	Extended Programme for Immunisation	String
	(11)	Extraordinary commissioner for the Covid-19 emergency	String
	(12)	Federal Office of Public Health	String
	[92 docs returned]		

Question 14	Specific to Singapore, display the daily total_vaccinations starting (inclusive) March-1 2021 through (inclusive) May-31 2021.																																	
Query	<pre>db.country_vac.aggregate([{ \$match: { \$and: [{ country: "Singapore", date: { \$gte: ISODate("2021-03-01"), \$lte: ISODate("2021-05-31") } }] } }, { \$project: { _id: 0, date: 1, country: 1, total_vaccinations: 1 } }, { \$sort: {date:1}}])</pre>																																	
Output	<table><tr><th>Key</th><th>Value</th><th>Type</th></tr><tr><td>1</td><td>{ country: "Singapore", date: ISODate("2021-03-01T08:00:00.000+08:00"), total_vaccinations: 525039 }</td><td>Object</td></tr><tr><td>country</td><td>Singapore</td><td>String</td></tr><tr><td>date</td><td>3/1/2021, 8:00:00 AM</td><td>Date</td></tr><tr><td>total_vaccinations</td><td>525,039 (0.53M)</td><td>Double</td></tr><tr><td>2</td><td>{ country: "Singapore", date: ISODate("2021-03-02T08:00:00.000+08:00"), total_vaccinations: 0 }</td><td>Object</td></tr><tr><td>country</td><td>Singapore</td><td>String</td></tr><tr><td>date</td><td>3/2/2021, 8:00:00 AM</td><td>Date</td></tr><tr><td>total_vaccinations</td><td>0</td><td>Double</td></tr><tr><td>3</td><td>{ country: "Singapore", date: ISODate("2021-03-03T08:00:00.000+08:00"), total_vaccinations: 0 }</td><td>Object</td></tr><tr><td>4</td><td>{ country: "Singapore", date: ISODate("2021-03-04T08:00:00.000+08:00"), total_vaccinations: 565000 }</td><td>Object</td></tr></table> <p>[92 docs returned]</p>	Key	Value	Type	1	{ country: "Singapore", date: ISODate("2021-03-01T08:00:00.000+08:00"), total_vaccinations: 525039 }	Object	country	Singapore	String	date	3/1/2021, 8:00:00 AM	Date	total_vaccinations	525,039 (0.53M)	Double	2	{ country: "Singapore", date: ISODate("2021-03-02T08:00:00.000+08:00"), total_vaccinations: 0 }	Object	country	Singapore	String	date	3/2/2021, 8:00:00 AM	Date	total_vaccinations	0	Double	3	{ country: "Singapore", date: ISODate("2021-03-03T08:00:00.000+08:00"), total_vaccinations: 0 }	Object	4	{ country: "Singapore", date: ISODate("2021-03-04T08:00:00.000+08:00"), total_vaccinations: 565000 }	Object
Key	Value	Type																																
1	{ country: "Singapore", date: ISODate("2021-03-01T08:00:00.000+08:00"), total_vaccinations: 525039 }	Object																																
country	Singapore	String																																
date	3/1/2021, 8:00:00 AM	Date																																
total_vaccinations	525,039 (0.53M)	Double																																
2	{ country: "Singapore", date: ISODate("2021-03-02T08:00:00.000+08:00"), total_vaccinations: 0 }	Object																																
country	Singapore	String																																
date	3/2/2021, 8:00:00 AM	Date																																
total_vaccinations	0	Double																																
3	{ country: "Singapore", date: ISODate("2021-03-03T08:00:00.000+08:00"), total_vaccinations: 0 }	Object																																
4	{ country: "Singapore", date: ISODate("2021-03-04T08:00:00.000+08:00"), total_vaccinations: 565000 }	Object																																

Question 15	When is the first batch of vaccinations recorded in Singapore?									
Query	<pre>// Question 15: When is the first batch of vaccinations recorded in Singapore? db.country_vac.aggregate({\$match: {"country": "Singapore"}}, {\$match: {"total_vaccinations":{"\$gt:0}}}, {\$sort: {"total_vaccinations":1}}, {\$project: {_id:0, "First_Vaccination_Date":"\$date"}}, {\$limit: 1})</pre>									
Output	<table><tr><th>Key</th><th>Value</th><th>Type</th></tr><tr><td>(1)</td><td>{ First_Vaccination_Date : ISODate("2021-01-11T08:00:00.000+08:00") }</td><td>Object</td></tr><tr><td>First_Vaccination_Date</td><td>1/11/2021, 8:00:00 AM</td><td>Date</td></tr></table>	Key	Value	Type	(1)	{ First_Vaccination_Date : ISODate("2021-01-11T08:00:00.000+08:00") }	Object	First_Vaccination_Date	1/11/2021, 8:00:00 AM	Date
Key	Value	Type								
(1)	{ First_Vaccination_Date : ISODate("2021-01-11T08:00:00.000+08:00") }	Object								
First_Vaccination_Date	1/11/2021, 8:00:00 AM	Date								
Explanation	First, we filter the documents to match the country “Singapore” as well as to only show those documents where the total_vaccinations is greater than 0. With this, we can now find when is the first batch of vaccinations. This can be done by doing an ascending sort on total vaccinations, and the first date in the sorted list will be the first batch of vaccinations recorded in Singapore.									

Question 16	Based on the date identified in (15), specific to Singapore, compute the total number of new cases thereafter. For instance, if the date identified in (15) is Jan-1 2021, the total number of new cases will be the sum of new cases starting from (inclusive) Jan-1 to the last date in the dataset.
Query	<pre>db.covid19data2.aggregate([{\$lookup: { from: "country_vac", pipeline: [{\$match: {"country": "Singapore"}}, {\$match: {"total_vaccinations":{"\$gt:0}}}, {\$sort: {"total_vaccinations":1}}, {\$project: {_id:0, "date":"\$date"}}, {\$limit: 1}], as: "First_vaccination_date" } }, {\$unwind: "\$First_vaccination_date"}, {\$match: {\$and:[{country:"Singapore"}, {\$expr:{\$gte: ["\$date", "\$First_vaccination_date.date"]}}]} }, {\$group: {_id:{groupbycountry:"\$country"}, sum_newcases:{\$sum:"\$new_cases"}}}, {\$project: {_id:0, "Total_number_of_cases_after_first_batch_of_vaccinations":"\$sum_newcases"}}])</pre>

Output	<table><tr><th>Key ↕</th><th>Value ↗</th><th>Type</th></tr><tr><td>4 (1)</td><td>{ Total_number_of_cases_after_first_batch</td><td>Object</td></tr><tr><td>Total_number_of_cases_after_first_batch_of_vaccinations</td><td>3,710 (3.7K)</td><td>Double</td></tr></table>	Key ↕	Value ↗	Type	4 (1)	{ Total_number_of_cases_after_first_batch	Object	Total_number_of_cases_after_first_batch_of_vaccinations	3,710 (3.7K)	Double
Key ↕	Value ↗	Type								
4 (1)	{ Total_number_of_cases_after_first_batch	Object								
Total_number_of_cases_after_first_batch_of_vaccinations	3,710 (3.7K)	Double								
Explanation	<p>Now similar to SQL, where we can perform an outer SELECT query using the inner query's output. In NoSQL, we can perform a similar task but by using the lookup function which aims to join 2 collections based on certain fields.</p> <p>However, here, we do not wish to join the output from (15) with the current covid19data2 collection based on any fields. Hence in the lookup function, the pipeline is carried out to obtain the output from (15) and this output will then be joined into all the existing documents in the covid19data2's collection. We used the \$unwind function to deconstruct the array field showing just the date in the array "First_vaccination_date".</p> <p>With the first vaccination date available in all the documents, we can now perform a match based on country "Singapore" and using \$expr -> \$gte, to show only those documents where the date of the document is greater than or equal to the first vaccination date. With that, we can now find the total number of cases after the first vaccination date. Hence, the next step is to conduct a group by on country (to show only 1 document) and we sum the number of new_cases in the group. Lastly, we perform a project to show only the total number of cases after the first batch of vaccinations which amounts to 3710.</p>									

Question 17	<p>Compute the total number of new cases in Singapore before the date identified in (15). For instance, if the date identified in (15) is Jan-1 2021 and the first date recorded (in Singapore) in the dataset is Feb-1 2020, the total number of new cases will be the sum of new cases starting from (inclusive) Feb-1 2020 through (inclusive) Dec-31 2020.</p>									
Query	<pre>db.covid19data2.aggregate([{\$lookup: { from: "country_vac", pipeline: [{\$match: {"country": "Singapore"}}, {\$match: {"total_vaccinations":{\$gt:0}}}, {\$sort: {"total_vaccinations:1}}, {\$project: {_id:0, "date":"\$date"}}, {\$limit: 1}], as: "First_vaccination_date" } }, {\$unwind: "\$First_vaccination_date"}, {\$match: {\$and:[{country:"Singapore"}, {\$expr:{\$lt: ["\$date", "\$First_vaccination_date.date"]}}]} }, {\$group: {_id:{groupbycountry:"\$country"},sum_newcases:{\$sum:"\$new_cases"}}}, {\$project: {_id:0, "Total_number_of_cases_before_first_batch_of_vaccinations": "\$sum_newcases"}}])</pre>									
Output	<table><tr><th>Key</th><th>Value</th><th>Type</th></tr><tr><td>(1)</td><td>{ Total_number_of_cases_before_first_batch_of_vaccination: 58,907 (58.9K)</td><td>Object</td></tr><tr><td>Total_number_of_cases_before_first_batch_of_vaccination: 58,907 (58.9K)</td><td></td><td>Double</td></tr></table>	Key	Value	Type	(1)	{ Total_number_of_cases_before_first_batch_of_vaccination: 58,907 (58.9K)	Object	Total_number_of_cases_before_first_batch_of_vaccination: 58,907 (58.9K)		Double
Key	Value	Type								
(1)	{ Total_number_of_cases_before_first_batch_of_vaccination: 58,907 (58.9K)	Object								
Total_number_of_cases_before_first_batch_of_vaccination: 58,907 (58.9K)		Double								
Explanation	<p>Question 17 is very similar to question 16 where instead of finding the total number of new cases after the first batch of vaccination, we are instead interested in finding the number of cases before the first batch. Hence, the only difference here in the query is instead of finding the dates in the document which is greater than the date of the first batch of vaccination, we find the dates in the document which is LESSER (hence we used \$lt : ["\$date": \$First_vaccination_date.date"])</p>									

Question 18	<p>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.</p>
-------------	---

Query

```
// Question 18: Herd immunity estimation. On a daily basis, specific to Germany, calculate the percentage of new cases and
// total vaccinations on each available vaccine in relation to its population.
db.covid19data2.aggregate([
  {$match:{country:"Germany"}},
  {$lookup:{
    from:"country_vac_manu",
    localField:"date",
    foreignField:"date",
    pipeline:[
      {$match:{country:"Germany"}},
      {$project:{country:1,date:1,vaccine:1,total_vaccinations:1}}
    ],
    as: "vac_data"
  }},
  {$unwind:"$vac_data"},
  {$project:{
    _id:0,
    country:1,
    date:1,
    "perc_new_cases":{$multiply:[{$divide:["$new_cases","$population"]},100]},
    "perc_total_vac":{$multiply:[{$divide:["$vac_data.total_vaccinations","$population"]},100]},
    "vac":"$vac_data.vaccine"
  }},
  {$group:{_id:{groupbycountry:"$country",groupbydate:"$date",groupbycases:"$perc_new_cases"},
    "VacPer2":{$push:{"Vaccine":"$vac","Percentage_total_cases":"$perc_total_vac"}}}},
  {$project:{
    _id:0,
    "Country":"$_id.groupbycountry",
    "Date":"$_id.groupbydate",
    "Percentage_new_cases":"$_id.groupbycases",
    "Vaccines_percentage":"$VacPer2"
  }},
  {$sort:{Date:1}},
])
```

Output

Key	Value	Type
1	{ Country : "Germany", Percentage_new_cases : 0.0147987779758998 } (4 fields)	Object
Country	Germany	String
Date	12/27/2020, 8:00:00 AM	Date
Percentage_new_cases	0.0148	Double
Vaccines_percentage	Array[4]	Array
0	{ Vaccine : "Johnson&Johnson", Percentage_total_cases : 0 }	Object
1	{ Vaccine : "Moderna", Percentage_total_cases : 0.0000023870921809661745 }	Object
2	{ Vaccine : "Oxford/AstraZeneca", Percentage_total_cases : 0 }	Object
3	{ Vaccine : "Pfizer/BioNTech", Percentage_total_cases : 0.02783230128397511 }	Object
Vaccine	Pfizer/BioNTech	String
Percentage_total_cases	0.0278	Double
2	{ Country : "Germany", Percentage_new_cases : 0.016714419451125152 } (4 fields)	Object
3	{ Country : "Germany", Percentage_new_cases : 0.023233568197343776 } (4 fields)	Object
4	{ Country : "Germany", Percentage_new_cases : 0.05853627446165253 } (4 fields)	Object
5	{ Country : "Germany", Percentage_new_cases : 0.023115407134385948 } (4 fields)	Object
6	{ Country : "Germany", Percentage_new_cases : 0.0025267370735526954 } (4 fields)	Object

[186 documents returned]

Explanation

Here, the first step we took is to perform a lookup to join the covid19data2 collection as well as the country_vac_manu collection based on the date field. In the lookup pipeline, we are only interested in finding those documents that match the country "Germany" and we only want to show the country, date, vaccine, and the total vaccinations in the documents obtained from the country_vac_manu collection. This information is saved into "vac_data" which we \$unwind to obtain an array "vac_data" with its elements being those obtained from country_vac_manu.

Now with the attributes vaccines, total vaccination, new cases, and population

	<p>available in each document, we are able to now calculate the percentage of new cases and percentage of total vaccination in relation to the population. We perform a project to show the country, date, "perc_new_cases" obtained by dividing "new_cases" by the population *100 as well as the "perc_total_vac" by dividing total vaccinations with the population *100.</p> <p>However, as the question wants to see the "perc_total_vac" on each available vaccine for each available date. Therefore, we have to perform a group by on country, date, "perc_new_cases" (to show all 3 attributes in a single document) and create an array called "VacPer2" where we pushed the vaccine type and its corresponding "perc_total_vac" into it.</p> <p>Lastly, to present the information nicely, we conduct a project where we display only the country's name, date, percentage new cases (<i>note that _id is used in front of these attributes as in a group by document</i>), and an array "VaccinesPercentage" (to replace "VacPer2") and sort the documents in ascending order of date.</p>
--	--

Question 19	Vaccination Drivers. Specific to Germany, based on each daily new case, display the total vaccinations of each available vaccine after 20 days, 30 days, and 40 days.
Query	<pre> db.covid19data2.aggregate([{\$match:{country:"Germany"}}, {\$project:{ _id:0, country:1, date:1, new_cases:1, "twentydayslater":{\$dateAdd: { startDate:"\$date" , unit: "day", amount: 20, }}, "thirtydayslater":{\$dateAdd: { startDate:"\$date" , unit: "day", amount: 30, }}, "fortydayslater":{\$dateAdd: { startDate:"\$date" , unit: "day", amount: 40, }}, }},]) </pre>

```

    {$lookup:{
      from:"country_vac_manu",
      localField:"twentydayslater",
      foreignField:"date",
      pipeline:[
        {$match:{country:"Germany"}},
        {$project:{_id:0,date:1,vaccine:1,total_vaccinations:1}}
      ],
      as: "twentyVac"
    }},
    {$lookup:{
      from:"country_vac_manu",
      localField:"thirtydayslater",
      foreignField:"date",
      pipeline:[
        {$match:{country:"Germany"}},
        {$project:{_id:0,date:1,vaccine:1,total_vaccinations:1}}
      ],
      as: "thirtyVac"
    }},
    {$lookup:{
      from:"country_vac_manu",
      localField:"fortydayslater",
      foreignField:"date",
      pipeline:[
        {$match:{country:"Germany"}},
        {$project:{_id:0,date:1,vaccine:1,total_vaccinations:1}}
      ],
      as: "fortyVac"
    }},
    //{$match:{twentyVac:{$ne:[]}}, /* comment out to remove the null values before 12-07-2020 */
    {$project:{_id:0,country:1,date:1,new_cases:1,"TwentyDaysLaterVacData":"$twentyVac",
      "ThirtyDaysLaterVacData":"$thirtyVac","FortyDaysLaterVacData":"$fortyVac"}}
  })

```

Output

▶ (318)	{6 fields}	Object
▶ (319)	{6 fields}	Object
country	Germany	String
date	12/7/2020, 8:00:00 AM	Date
new_cases	5,456 (5.5K)	Double
▶ TwentyDaysLaterVacData	Array[4]	Array
▶ 0	{ vaccine : "Johnson&Johnson", date : ISODate("2020-12-27T08:00:00.000+08:00"), total_vaccinations : 0 }	Object
vaccine	Johnson&Johnson	String
date	12/27/2020, 8:00:00 AM	Date
total_vaccinations	0	Double
▶ 1	{ vaccine : "Moderna", date : ISODate("2020-12-27T08:00:00.000+08:00"), total_vaccinations : 2 }	Object
▶ 2	{ vaccine : "Oxford/AstraZeneca", date : ISODate("2020-12-27T08:00:00.000+08:00"), total_vaccinations : 0 }	Object
▶ 3	{ vaccine : "Pfizer/BioNTech", date : ISODate("2020-12-27T08:00:00.000+08:00"), total_vaccinations : 23319 }	Object
▶ ThirtyDaysLaterVacData	Array[4]	Array
▶ FortyDaysLaterVacData	Array[4]	Array
▶ (320)	{6 fields}	Object
▶ (321)	{6 fields}	Object
▶ (322)	{6 fields}	Object
▶ (323)	{6 fields}	Object
▶ (324)	{6 fields}	Object

[528 documents returned]

Explanation

The key to tackling this question is to first imagine how we would like our table to be at the end of the query. As the question requires us to display the daily new cases and the total vaccinations of each vaccine after 20, 30, and 40 days. How we would imagine the final table to be like would be to show the country's name, the current

	<p>date, the current date's new cases, and 3 arrays. Each array is to show the date corresponding to the 20th/30th/40th, the vaccine type, and for each corresponding vaccine, its total vaccinations.</p> <p>Now we have that picture in mind, we aim to first project the documents to be similar to that of the end picture. Hence, we first performed a match on country "Germany" and performed a project to view the country's name, the current date, and the current date's new cases. Additionally, we created 3 additional fields, which correspond to the 20th, 30th, and 40th date from the current date, which was obtained by using \$dateAdd.</p> <p>Now, with the 20th, 30th, and 40th dates available in each document, we can perform 3 separate lookups joining on the 20th, 30th, and 40th date fields respectively. In each lookup, we join with the country_vac_manu collection to obtain the vaccine and its total vaccinations corresponding to each date.</p> <p>Now, to present the information nicely, we do a project, showing only country, date, new cases, as well as the 3 arrays previously obtained from the lookup joined collection with country_vac_manu.</p>
--	--

Question 20	<p>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.</p>
Query	<pre> db.country_vac_manu.aggregate([{\$match:{country:"Germany"}}, {\$group: {_id:{groupbydate:"\$date",groupbycountry:"\$country"}, "vacdata":{\$push:{"Vaccine":"\$vaccine", "VaccinationsAdministered":"\$total_vaccinations"}}}}, {\$project:{ _id:0, "date":"\$_id.groupbydate", "country":"\$_id.groupbycountry", "total_vac": {\$sum: "\$vacdata.VaccinationsAdministered"}, "threeweekslater":{\$dateAdd: { startDate:"\$_id.groupbydate", unit: "day", amount: 21, }}, "twomonthslater":{\$dateAdd: { startDate:"\$_id.groupbydate" , unit: "day", amount: 60, }}, "fourmonthslater":{\$dateAdd: { startDate:"\$_id.groupbydate" , unit: "day", amount: 120, }}, }}]), </pre>

```

    {$lookup:{
      from:"covid19data2",
      localField:"threeweekslater",
      foreignField:"date",
      pipeline:[
        {$match:{country:"Germany"}},
        {$project:{_id:0,date:1,new_cases:1}}
      ],
      as: "threeweeks"
    }},
    {$lookup:{
      from:"covid19data2",
      localField:"twomonthslater",
      foreignField:"date",
      pipeline:[
        {$match:{country:"Germany"}},
        {$project:{_id:0,date:1,new_cases:1}}
      ],
      as: "twomonths"
    }},
    {$lookup:{
      from:"covid19data2",
      localField:"fourmonthslater",
      foreignField:"date",
      pipeline:[
        {$match:{country:"Germany"}},
        {$project:{_id:0,date:1,new_cases:1}}
      ],
      as: "twomonths"
    }},
    {$lookup:{
      from:"covid19data2",
      localField:"fourmonthslater",
      foreignField:"date",
      pipeline:[
        {$match:{country:"Germany"}},
        {$project:{_id:0,date:1,new_cases:1}}
      ],
      as: "fourmonths"
    }},
    {$project: {"Date":"$date","Country":"$country","Total_number_of_accumulated_vaccinations":"$total_vac",
    "CasesIn21Days":"$threeweeks","CasesIn60Days":"$twomonths","CasesIn120Days":"$fourmonths"}},
    {$sort:{Date:1}}
  ])

```

Output

Key	Value	Type
1	{6 fields}	Object
Date	12/27/2020, 8:00:00 AM	Date
Country	Germany	String
Total_number_of_accumulated_v	23,321 (23.3K)	Double
CasesIn21Days	[{ date : ISODate("2021-01-17T08:00:00.000+08:00"), new_cases : 11484 }]	Array
CasesIn60Days	[{ date : ISODate("2021-02-25T08:00:00.000+08:00"), new_cases : 11032 }]	Array
CasesIn120Days	[{ date : ISODate("2021-04-26T08:00:00.000+08:00"), new_cases : 5961 }]	Array
2	{6 fields}	Object
Date	12/28/2020, 8:00:00 AM	Date
Country	Germany	String
Total_number_of_accumulated_v	41,139 (41.1K)	Double
CasesIn21Days	[{ date : ISODate("2021-01-18T08:00:00.000+08:00"), new_cases : 9253 }]	Array
CasesIn60Days	[{ date : ISODate("2021-02-26T08:00:00.000+08:00"), new_cases : 9437 }]	Array
CasesIn120Days	[{ date : ISODate("2021-04-27T08:00:00.000+08:00"), new_cases : 25911 }]	Array
3	{6 fields}	Object

[186 documents returned]

Explanation	<p>To approach question 20, we first perform a match on country, germany followed by a group by date. Next, we also created an array called "vacdata", where we push the vaccine type and its corresponding total vaccinations into the array. Effectively, we will now have documents that are grouped according to dates, and in each document, there will be an array vacdata which contains 4 documents, each pertaining the a type of vaccine along with its total vaccination count.</p> <p>The next step is to join the collection with the covid19data2 collection in order to obtain the daily new cases. However, we would like the obtain the daily new cases for the 20th, 60th and 120th day as such, similar to the approach taken in question 19, we first have to create local fields corresponding to the 21th, 60th and 120th day in all the documents and this can be done using \$dateAdd.</p> <p>Now, with the 21th, 60th, and 120th dates available in each document, we can perform 3 separate lookups joining on the 21th, 60th, and 120th date fields respectively. In each lookup, we join with the country_vac_manu collection to obtain the date and its corresponding daily new cases.</p> <p>Now, to present the information nicely, we do a project, showing date, country, total vaccinations, as well as the 3 arrays obtained from the lookup. In each array, we can view the date correspond to how many days after the current date as well as that corresponding date's daily new cases.</p>
--------------------	---

4. Difference between Relational and Non-relational Models

This section aims to highlight the differences in characteristics between relational and non-relational models. (Kandaurova, N. ,2020)

4.1 Characteristics

	RDBMS	Non-RDBMS
Data size	Medium size to large.	Optimized for huge data
Structure	Structured data (in tables) Faster in performing queries that require the joining of tables.	Unstructured data (in documents) Takes longer to perform queries that have multiple lookups.
Language	All about the SQL language.	Not locked into one language, depending on the type of NoSQL database implemented. For example, MongoDB stores all documents in a JSON format, & queries based on the JavaScript programming language.
Schemas	Requires a predefined schema that determines how tables are configured and data is stored	Uses a dynamic schema that requires no predefined data structure
ACID compliance	Yes, it follow strict ACID properties Atomicity, Consistency, Isolation, Durability. Provides strong data security.	No, it offers flexible data structure based on the BASE model. Basically Available, Soft state, Eventual consistency Does not provide strong database security.
Data integrity	Deliver a high degree of data integrity adhering to the principles of ACID, which is essential when supporting workloads such as	Difficult to deliver the same level of data integrity not as RDBMS, most adhering to BASE principles, meaning data in a distributed environment might be temporarily

	<p>financial transactions.</p> <p>Tables have relationships with other tables</p> <p>Use of primary and foreign keys</p> <p>Cascade delete and update</p>	<p>inconsistent.</p> <p>No cascade function: manually CRUD on one variable that may exist in multiple collections</p>
Query Complexity	<p>Easier to query data.</p> <p>Efficiently executes queries, retrieves and edits data quickly.</p> <p>Queries can be run by less technical staff. More well known query language.</p>	<p>Harder to query data.</p> <p>Lack consistency and typically require more work to query data, as query complexity increases.</p> <p>Requires more professional engineers, developers or data scientists.</p>
Scalability and Flexibility	<p>Designed to run on a single server in order to maintain data integrity.</p>	<p>Not constrained by the limits of single-server architectures</p> <p>Easily store and combine data of any structure without the need to modify a schema.</p>
Scaling	<p>Scale vertically</p> <p>Need to increase the capacity of a single server to scale the database.</p> <p>Not efficient for large distributed data sets.</p>	<p>Scale horizontally</p> <p>Objects can be easily stored on multiple servers without having to be linked.</p> <p>Possible to accommodate large stores of distributed data, while supporting increased levels of traffic.</p>
Handling of ever changing and highly dynamic Covid-19 related data	<p>Ensures accuracy in updated records, protects against data corruption.</p>	<p>When updating the data and more columns are added, the data input type does not need to be modified to meet ACID requirements</p>
Do you need real-time data?	<p>Works well for analysis of past data, as it is not built to handle data inputs in a flexible manner, with strict ACID properties</p>	<p>Easily update data</p>

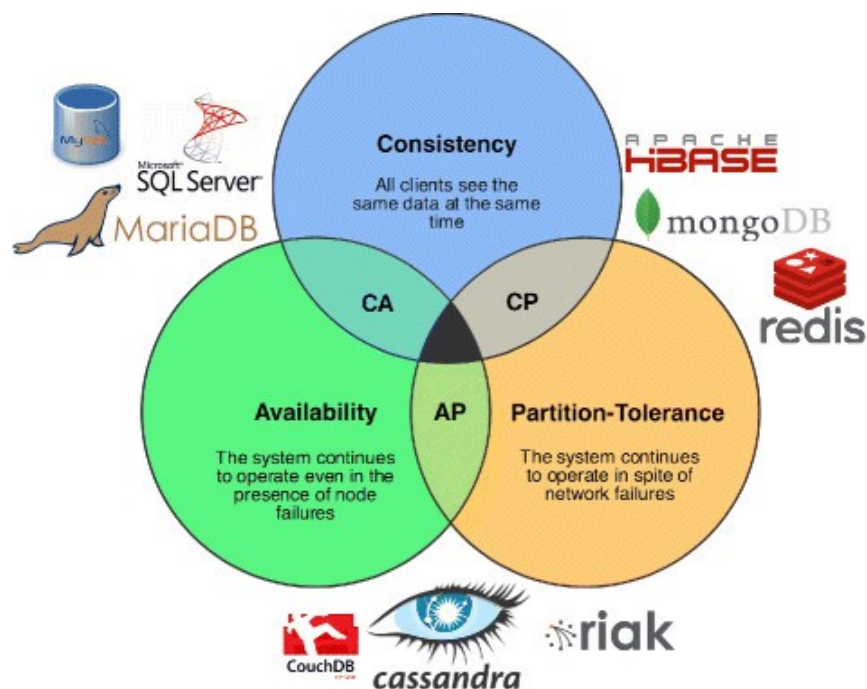
	imposed.	
Maturity	Built on mature technologies that are well known and supported by large developer communities.	NoSQL products are not as mature and the technologies not as well supported as SQL products.

To summarize in short, SQL databases offer great benefits for transactional data whose structure does not change frequently and where data integrity is paramount. (Pawlan, 2021) On the other hand. (What's the Difference? Relational vs Non-Relational Databases, 2021) NoSQL provides more flexibility and scalability, which lends itself to rapid development and iteration. (Chan, M. , 2021)

4.2 Brewer's CAP Theorem

Brewer's CAP Theorem

The CAP theorem states that, in a distributed system, it can have only two out of the three attributes.



CAP stands for Consistency, Availability and Partition tolerance. (CAP Theorem, 2021)

Consistency represents the “C” in ACID properties in non-distributed systems, as applied to the ideal properties of database transactions and means that data will never be persisted that breaks certain pre-set constraints.

Availability allows the user to retrieve the data they stored in a distributed system, no matter what happens inside the cluster. If a request is made, a response must be given from the system, even if a node (or many nodes) in the cluster goes down.

Partition tolerance means that the cluster continues to function even if there is communication break (partition) between two nodes.

CP database delivers consistency and partition tolerance at the expense of availability. When a partition occurs between any two nodes, the system has to shut down the non-consistent node until the partition is resolved. An example of a CP database includes NoSQL.

AP database delivers availability and partition tolerance at the expense of consistency. When a partition occurs, all nodes remain available but those at the wrong end of a partition might return an older version of data than others.

CA database delivers consistency and availability across all nodes. It cannot do this if there is a partition between any two nodes in the system, however, and therefore cannot deliver fault tolerance. An example of a CA database includes SQL.

5. Recommendations to WHO

WHO works worldwide to promote health, keep the world safe, and serve the vulnerable. (What We Do, 2021) Their goal is to ensure that a billion more people have universal health coverage, to protect a billion more people from health emergencies, and provide a further billion people with better health and well-being. Their guiding principle states that all people should enjoy the highest standard of health, regardless of race, religion, political belief, economic or social condition. This has guided WHO's work for the past 70 years. In light of the pandemic, the WHO Technology for Covid-19 Team has developed the Covid-19 information dashboard that allows access to current and reliable data on COVID-19 cases submitted directly to WHO by countries. The aim of this is to help its users become more aware of the evolving situation, and make better decisions during a health emergency such as the Covid-19 pandemic.



Fig 5.1, Features of the Covid-10 Detailed Surveillance Data Dashboard

As we can see from Fig 5.1, the features of the Covid-19 data dashboard mostly present data involving only 2 columns. In addition, as a world-renowned institute for updated health reports, the data presented by WHO has to maintain high standards of accuracy and timeliness.

Based on our analysis, our team recommends WHO to **implement a Relational database to store their data, and for its engineering team to conduct queries on, to generate insights from the stored data.**

Justification:

Considerations:	Why Relational Database Model (RDBMS) is recommended:
People: The engineering team at WHO	RDBMS takes less time to manage, and it is a more well-known query language. This makes it suitable for a team of varying levels of expertise to implement.
Accuracy: Ensures accuracy of data input	RDBMS ensures that data is standardised across different tables, and that data recorded is consistent for all countries. Stringent protocols put in place such as ACID reduces data inconsistency and likelihood of errors.
Analysis: Data analysts would like to observe trends across various entities (e.g. total vaccination and confirmed cases)	The data analysed to observe these trends might be found from multiple schemas. This would thus require join queries which RDBMS supports due to its specificity in storing data; data types are standardised across columns, which is crucial in order for the joining of tables. (SQL vs NoSQL Exact Difference (Know When To Use NoSQL and SQL), 2021)

Despite the advantages to using RDBMS for WHO, there are several limitations:

- RDBMS lacks the flexibility that Non-RDBMS provides. Changes made to the data will incur downtime
- When doing more complex queries across large tables, it may be more taxing on the RDBMS
- Due to the strict specificity of data input in RDBMS, WHO may reject data that are not standardised, resulting in the loss of valuable information (Sheldon, 2021)

We believe WHO prioritises the accuracy of results presented over how fast they present results. This might be the case since there are numerous analyses conducted using renowned data sets and dashboards such as the ones provided by WHO. As such, accuracy of data is of highest importance. **With this in mind, we believe that it is in WHO's best interest to implement a Relational database to store their data, and for its engineering team to conduct queries on, to generate insights from the stored data.**

References

¹ *CAP Theorem*. (2021, August 3). IBM. <https://www.ibm.com/in-en/cloud/learn/cap-theorem>

² Chan, M. (2021, September 15). *SQL vs. NoSQL – what's the best option for your database needs?* Thorn Technologies. <https://www.thorntech.com/sql-vs-nosql/>

³ E. (2021, November 8). *Non-relational data and NoSQL - Azure Architecture Center*. Microsoft Docs. <https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/non-relational-data#columnar-data-stores>

⁴ Kandaurova, N. (2020, December 21). *Differences Between Relational and Non-Relational Database*. Jelvix. <https://jelvix.com/blog/relational-vs-non-relational-database>

⁵ Meysman, A. (2021, August 11). *NoSQL Database Types*. Dzone.Com. <https://dzone.com/articles/nosql-database-types-1>

⁶ MongoDB. (2021). *Types of NoSQL Databases*. <https://www.mongodb.com/scale/types-of-nosql-databases>

⁷ Pawlan, D. (2021, April 29). *Relational vs. Non-Relational Database: Pros & Cons*. The Aloa Blog. <https://aloe.co/blog/relational-vs-non-relational-database-pros-cons>

⁸ *Power BI Report*. (n.d.). APP. <https://app.powerbi.com/view?r=eyJrljoiYWVhZGVkNWUtNmM0Ni00MDAwLTljYWMTN2EwNTM3YjQzYmRmliwidCI6ImY2MTBjMGI3LWJkMjQtNGIzOS04MTBiLTNkYzI4MGFmYjU5MCIslmMiOjh9>

⁹ Sheldon, R. (2021, May 3). *How to choose between SQL and NoSQL databases*. Simple Talk. <https://www.red-gate.com/simple-talk/databases/nosql/how-to-choose-between-sql-and-nosql-databases/>

¹⁰ *SQL vs NoSQL Exact Difference (Know When To Use NoSQL and SQL)*. (2021, November 1). Software Testing Help. <https://www.softwaretestinghelp.com/sql-vs-nosql/>

¹¹ *What we do*. (2021). WHO. <https://www.who.int/about/what-we-do>

¹² *What's the Difference? Relational vs Non-Relational Databases*. (2021, June 1). Logi Analytics. <https://www.logianalytics.com/relational-vs-non-relational-databases/>

Appendix

MySQL Queries (non-normalised)

1. What is the total population in Asia?

MySQL query:

```
# Question 1
• SELECT SUM(population) as Population_Asia FROM
  (SELECT DISTINCT(iso_code), max(date), population FROM covid19data
   WHERE continent = "Asia"
   GROUP BY iso_code, population) as t1;
```

Output:

	Population_Asia
▶	4614068610

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

MySQL query:

Question 2

- ```
SELECT SUM(DISTINCT(population)) AS total_population_ASEAN
FROM covid19data
WHERE location IN ('Brunei', 'Cambodia', 'Indonesia', 'Laos', 'Malaysia',
'Myanmar', 'Philippines', 'Singapore', 'Thailand', 'Vietnam');
```

Output:

|   |                        |
|---|------------------------|
|   | total_population_ASEAN |
| ▶ | 667301412              |

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

MySQL query:

# Question 3

- ```
select distinct(source_name)
from country_vaccinations
order by source_name;
```

Output: 92 data sources returned.

source_name		source_name
Presidency of the Maldives		Africa Centres for Disease...
Prime Minister's Office		Cayman Islands Government
Public Health Agency of Sweden		Centers for Disease Contro...
Public Health France		Costa Rican Social Securit...
Public Health Institute		COVID-19 Malta Public He...
RIVM via European CDC		COVID-19 Vaccine Informa...
Robert Koch Institut		Directorate General for He...
Saudi Health Council		Directorate General of Heal...
Sciensano		Directorate of Health
Secretary of Health		Extended Programme for I...
Social Security Institute		Extraordinary commissione...
SPC Public Health Division		Federal Office of Public He...
Statens Serum Institute		Finnish Institute for Health...
Taiwan Centers for Disease Control		Government of Andorra
World Health Organization		Government of Aruba

- Specific to Singapore, display the daily total_vaccinations starting (inclusive) March-1 2021 through (inclusive) May-31 2021.

MySQL query:

```
# Question 4
• SELECT date, total_vaccinations
FROM covid19data
WHERE iso_code = "SGP" AND date between '2021-03-01' AND '2021-05-31';
```

Output: 92 rows of data returned.

	date	total_vaccinations
▶	2021-03-01	525039.0
	2021-03-02	
	2021-03-03	
	2021-03-04	565000.0
	2021-03-05	
	2021-03-06	
	2021-03-07	596000.0
	2021-03-08	611314.0
	2021-03-09	
	2021-03-10	
	2021-03-11	
	2021-03-12	
	2021-03-13	
	2021-03-14	

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

MySQL query:

Question 5

- ```
select min(cvd.date) as First_batch_of_vaccinations
from covid19data cvd
where iso_code = "SGP"
and cvd.total_vaccinations > 0;
```

**Output:** The first batch of vaccinations was recorded on 1 November 2021.

|   | First_batch_of_vaccinations |
|---|-----------------------------|
| ▶ | 2021-01-11                  |

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

**MySQL query:**

#### # Question 6

- ```
select sum(cvd.new_cases) as Total_number_of_cases_after_first_batch_of_vaccinations
from covid19data cvd
where cvd.date >=
  (select min(cvd.date) as First_batch_of_vaccinations
   from covid19data cvd
   where iso_code = "SGP"
   and cvd.total_vaccinations>0)
and iso_code = "SGP";
```

Output:

	Total_number_of_cases_after_first_batch_of_vaccinations
▶	3710

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

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

MySQL query:

Question 7

- ```
select sum(cvd.new_cases) as Total_number_of_cases_before_first_batch_of_vaccinations
from covid19data cvd
where cvd.date <
 (select min(cvd.date) as First_batch_of_vaccinations
 from covid19data cvd
 where iso_code = "SGP"
 and cvd.total_vaccinations>0)
and iso_code = "SGP";
```

Output:

|   | Total_number_of_cases_before_first_batch_of_vaccinations |
|---|----------------------------------------------------------|
| ▶ | 58907                                                    |

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

MySQL query:

```
Question 8
• SELECT t1.date, t1.location, t2.perc_new_cases, t1.vaccine, t1.total_vaccinations/t2.population*100 as perc_total_vac FROM
 (SELECT * FROM country_vaccinations_by_manufacturer
 WHERE location = "Germany") as t1
 LEFT JOIN
 (SELECT date, population, new_cases/population*100 as perc_new_cases
 FROM covid19data
 WHERE location = "Germany"
 ORDER BY date) as t2
 ON t1.date = t2.date
 ORDER BY date;
```

Output: 744 rows returned.

| date       | location | perc_new_cases       | vaccine            | perc_total_vac           |
|------------|----------|----------------------|--------------------|--------------------------|
| 2020-12-27 | Germany  | 0.0147987779758998   | Johnson&Johnson    | 0                        |
| 2020-12-27 | Germany  | 0.0147987779758998   | Moderna            | 0.0000023870921809661745 |
| 2020-12-27 | Germany  | 0.0147987779758998   | Oxford/AstraZeneca | 0                        |
| 2020-12-27 | Germany  | 0.0147987779758998   | Pfizer/BioNTech    | 0.02783230128397511      |
| 2020-12-28 | Germany  | 0.016714419451125152 | Johnson&Johnson    | 0                        |
| 2020-12-28 | Germany  | 0.016714419451125152 | Moderna            | 0.0000023870921809661745 |
| 2020-12-28 | Germany  | 0.016714419451125152 | Oxford/AstraZeneca | 0                        |
| 2020-12-28 | Germany  | 0.016714419451125152 | Pfizer/BioNTech    | 0.049098905524202756     |
| 2020-12-29 | Germany  | 0.023233568197343776 | Johnson&Johnson    | 0                        |
| 2020-12-29 | Germany  | 0.023233568197343776 | Moderna            | 0.0000023870921809661745 |
| 2020-12-29 | Germany  | 0.023233568197343776 | Oxford/AstraZeneca | 0                        |
| 2020-12-29 | Germany  | 0.023233568197343776 | Pfizer/BioNTech    | 0.10849333962491263      |

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

## MySQL query:

```
Question 9
• select t1.date, t1.new_cases, t2. 20_days, t2.vaccine , t2.max_vac_20days, t3. 30_days,
 t3.vaccine, t3.max_vac_30days, t4. 40_days, t4.vaccine, t4.max_vac_40days from
 (SELECT date, new_cases
 FROM covid19data
 WHERE location = "Germany"
 ORDER BY date) t1
 left join
 (SELECT location, date as 20_days, vaccine, total_vaccinations AS max_vac_20days
 FROM country_vaccinations_by_manufacturer
 WHERE location = "Germany"
 order by date asc) t2
 on datediff(t2. 20_days, t1.date)=20
 left join
 (SELECT location, date as 30_days, vaccine, total_vaccinations AS max_vac_30days
 FROM country_vaccinations_by_manufacturer
 WHERE location = "Germany"
 order by date asc) t3
 on (datediff(t3. 30_days, t1.date)=30
 and
 t3.vaccine = t2.vaccine)
 left join
 (SELECT location, date as 40_days, vaccine, total_vaccinations AS max_vac_40days
 FROM country_vaccinations_by_manufacturer
 WHERE location = "Germany"
 order by date asc) t4
 on (datediff(t4. 40_days, t1.date)=40
 and
 t4.vaccine = t2.vaccine);
```

Output: 1086 rows returned.

| date       | new_cases | 20_days    | vaccine            | max_vac_20days | 30_days    | vaccine            | max_vac_30days | 40_days    | vaccine            | max_vac_40days |
|------------|-----------|------------|--------------------|----------------|------------|--------------------|----------------|------------|--------------------|----------------|
| 2020-12-01 | 24766.0   | NULL       | NULL               | NULL           | NULL       | NULL               | NULL           | NULL       | NULL               | NULL           |
| 2020-12-02 | 23275.0   | NULL       | NULL               | NULL           | NULL       | NULL               | NULL           | NULL       | NULL               | NULL           |
| 2020-12-03 | 23591.0   | NULL       | NULL               | NULL           | NULL       | NULL               | NULL           | NULL       | NULL               | NULL           |
| 2020-12-04 | 15970.0   | NULL       | NULL               | NULL           | NULL       | NULL               | NULL           | NULL       | NULL               | NULL           |
| 2020-12-05 | 26126.0   | NULL       | NULL               | NULL           | NULL       | NULL               | NULL           | NULL       | NULL               | NULL           |
| 2020-12-06 | 10910.0   | NULL       | NULL               | NULL           | NULL       | NULL               | NULL           | NULL       | NULL               | NULL           |
| 2020-12-07 | 5456.0    | 2020-12-27 | Pfizer/BioNTech    | 23319          | 2021-01-06 | Pfizer/BioNTech    | 459272         | 2021-01-16 | Pfizer/BioNTech    | 1144835        |
| 2020-12-07 | 5456.0    | 2020-12-27 | Oxford/AstraZeneca | 0              | 2021-01-06 | Oxford/AstraZeneca | 5              | 2021-01-16 | Oxford/AstraZeneca | 14             |
| 2020-12-07 | 5456.0    | 2020-12-27 | Moderna            | 2              | 2021-01-06 | Moderna            | 16             | 2021-01-16 | Moderna            | 13341          |
| 2020-12-07 | 5456.0    | 2020-12-27 | Johnson&Johnson    | 0              | 2021-01-06 | Johnson&Johnson    | 0              | 2021-01-16 | Johnson&Johnson    | 0              |
| 2020-12-08 | 29263.0   | 2020-12-28 | Pfizer/BioNTech    | 41137          | 2021-01-07 | Pfizer/BioNTech    | 513416         | 2021-01-17 | Pfizer/BioNTech    | 1191952        |
| 2020-12-08 | 29263.0   | 2020-12-28 | Oxford/AstraZeneca | 0              | 2021-01-07 | Oxford/AstraZeneca | 6              | 2021-01-17 | Oxford/AstraZeneca | 15             |
| 2020-12-08 | 29263.0   | 2020-12-28 | Moderna            | 2              | 2021-01-07 | Moderna            | 19             | 2021-01-17 | Moderna            | 14602          |
| 2020-12-08 | 29263.0   | 2020-12-28 | Johnson&Johnson    | 0              | 2021-01-07 | Johnson&Johnson    | 0              | 2021-01-17 | Johnson&Johnson    | 0              |
| 2020-12-09 | 25089.0   | 2020-12-29 | Pfizer/BioNTech    | 90900          | 2021-01-08 | Pfizer/BioNTech    | 575264         | 2021-01-18 | Pfizer/BioNTech    | 1269903        |
| 2020-12-09 | 25089.0   | 2020-12-29 | Oxford/AstraZeneca | 0              | 2021-01-08 | Oxford/AstraZeneca | 7              | 2021-01-18 | Oxford/AstraZeneca | 16             |
| 2020-12-09 | 25089.0   | 2020-12-29 | Moderna            | 2              | 2021-01-08 | Moderna            | 68             | 2021-01-18 | Moderna            | 18756          |
| 2020-12-09 | 25089.0   | 2020-12-29 | Johnson&Johnson    | 0              | 2021-01-08 | Johnson&Johnson    | 0              | 2021-01-18 | Johnson&Johnson    | 0              |

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

MySQL query:

```
Question 10
• select * from
 (select date, sum(total_vaccinations) as total_number_of_accumulated_vaccinations
 FROM country_vaccinations_by_manufacturer
 WHERE location = "Germany"
 group by date) t1
 left join
 (select date as 21_days, new_cases as daily_new_cases_21days
 from covid19data
 where location = "Germany"
) t2
 on datediff(t2. 21_days, t1.date)=21

 left join
 (select date as 60_days, new_cases as daily_new_cases_60days
 from covid19data
 where location = "Germany"
) t3
 on datediff(t3. 60_days, t1.date)=60

 left join
 (select date as 120_days, new_cases as daily_new_cases_120days
 from covid19data
 where location = "Germany"
) t4
 on datediff(t4. 120_days, t1.date)=120;
```

**Output:**186 rows returned.

| date       | total_number_of_accumulated_vaccinat... | 21_days    | daily_new_cases_21days | 60_days    | daily_new_cases_60days | 120_days   | daily_new_cases_120days |
|------------|-----------------------------------------|------------|------------------------|------------|------------------------|------------|-------------------------|
| 2020-12-27 | 23321                                   | 2021-01-17 | 11484.0                | 2021-02-25 | 11032.0                | 2021-04-26 | 5961.0                  |
| 2020-12-28 | 41139                                   | 2021-01-18 | 9253.0                 | 2021-02-26 | 9437.0                 | 2021-04-27 | 25911.0                 |
| 2020-12-29 | 90902                                   | 2021-01-19 | 12233.0                | 2021-02-27 | 7671.0                 | 2021-04-28 | 28263.0                 |
| 2020-12-30 | 152687                                  | 2021-01-20 | 29003.0                | 2021-02-28 | 6118.0                 | 2021-04-29 | 24212.0                 |
| 2020-12-31 | 202971                                  | 2021-01-21 | 8277.0                 | 2021-03-01 | 5274.0                 | 2021-04-30 | 14326.0                 |
| 2021-01-01 | 228732                                  | 2021-01-22 | 16366.0                | 2021-03-02 | 6492.0                 | 2021-05-01 | 18535.0                 |
| 2021-01-02 | 276370                                  | 2021-01-23 | 12430.0                | 2021-03-03 | 10852.0                | 2021-05-02 | 8776.0                  |
| 2021-01-03 | 298551                                  | 2021-01-24 | 10078.0                | 2021-03-04 | 11393.0                | 2021-05-03 | 5510.0                  |
| 2021-01-04 | 345130                                  | 2021-01-25 | 6887.0                 | 2021-03-05 | 9581.0                 | 2021-05-04 | 24111.0                 |
| 2021-01-05 | 397029                                  | 2021-01-26 | 9387.0                 | 2021-03-06 | 8264.0                 | 2021-05-05 | 22458.0                 |
| 2021-01-06 | 459293                                  | 2021-01-27 | 15636.0                | 2021-03-07 | 6504.0                 | 2021-05-06 | 17917.0                 |
| 2021-01-07 | 513441                                  | 2021-01-28 | 14883.0                | 2021-03-08 | 5129.0                 | 2021-05-07 | 15090.0                 |
| 2021-01-08 | 575339                                  | 2021-01-29 | 12831.0                | 2021-03-09 | 6834.0                 | 2021-05-08 | 13125.0                 |
| 2021-01-09 | 634837                                  | 2021-01-30 | 17518.0                | 2021-03-10 | 21163.0                | 2021-05-09 | 0.0                     |
| 2021-01-10 | 669489                                  | 2021-01-31 | 748.0                  | 2021-03-11 | 4745.0                 | 2021-05-10 | 7321.0                  |
| 2021-01-11 | 736621                                  | 2021-02-01 | 6668.0                 | 2021-03-12 | 12770.0                | 2021-05-11 | 19696.0                 |
| 2021-01-12 | 817360                                  | 2021-02-02 | 7690.0                 | 2021-03-13 | 10568.0                | 2021-05-12 | 6590.0                  |
| 2021-01-13 | 928980                                  | 2021-02-03 | 12487.0                | 2021-03-14 | 8978.0                 | 2021-05-13 | 13631.0                 |