BC2402 Designing & Developing Databases

Seminar Group: 7
Group: 5

Team Members:
Adrian Goh Jun Wei (U1721134D)
Deionna Chee Rui Ping (U2021449D)
Faith Lee Kai Ling (U2022307C)
Muhammad Ihsan Bin Mohammad Azmi (U2021160B)

Professor:
Chan Wai Xin

Table of Contents

Executive Summary

The rise of COVID-19 in recent times has made data crucial for planning powerful responses and preventing the misinformation of the general public. Proper management of data builds the foundation for its effective utilisation. In conjunction with WHO's efforts to improve data access, our team has implemented relational and non-relational databases, and demonstrated how both have capabilities to derive insights and provide huge value to a user.

Implementation of the relational database was carried out via mySQL, where raw data variables were first normalised before being translated into the program. For the non-relational database, it was implemented via noSQL. No normalisation was required, hence the raw data could be directly uploaded into collections as documents, after creating a database.

After implementations, both types of databases were compared, and it was concluded that the relational database was a better fit given the current context and the needs of WHO, due to its higher accuracy and ease of use.

1. Introduction

Information has been a pertinent issue throughout the COVID-19 pandemic. Viruses are complicated, with experts in the field requiring years of dedication to properly understand them. This inherent complexity combined with the lack of in-depth knowledge of the general public on the matter has resulted in a climate ripe for the constant propagation of misinformation and fake news. How we utilise available data is key to navigating this global pandemic, whether it is to keep people informed of the facts, or to make the appropriate decisions. For this to be possible, it is crucial that an efficient way to manage and access the data we have is implemented. This aligns with WHO's efforts to find a solution to infodemic management on the current situation.

1.1. Problem Statement

How can we better manage the large amounts of data that we have to develop valuable insights that impactful decisions can be based on?

1.2. Approach

To address this problem, making use of currently available data, our team implemented both relational and non-relational models. These two models were then evaluated and compared to each other to determine the most appropriate one to tackle the issue at hand.

Our relational database made use of mySQL. The raw datasets used were cleaned to ensure that empty or irrelevant entries were removed. This cleaning was also a necessary step to achieve a third normal form (3NF) during the normalisation of our relational data model. Through the normalisation process, the different columns were organised into a total of 11 tables from the original 3 datasets. The normalised tables had unnecessary data and functional dependencies removed from them. To show the relationships between decomposed created tables, we constructed an Entity Relation Diagram (ERD). After creating the tables in mySQL from the raw data, queries were performed on the created database showing its capability to generate insights.

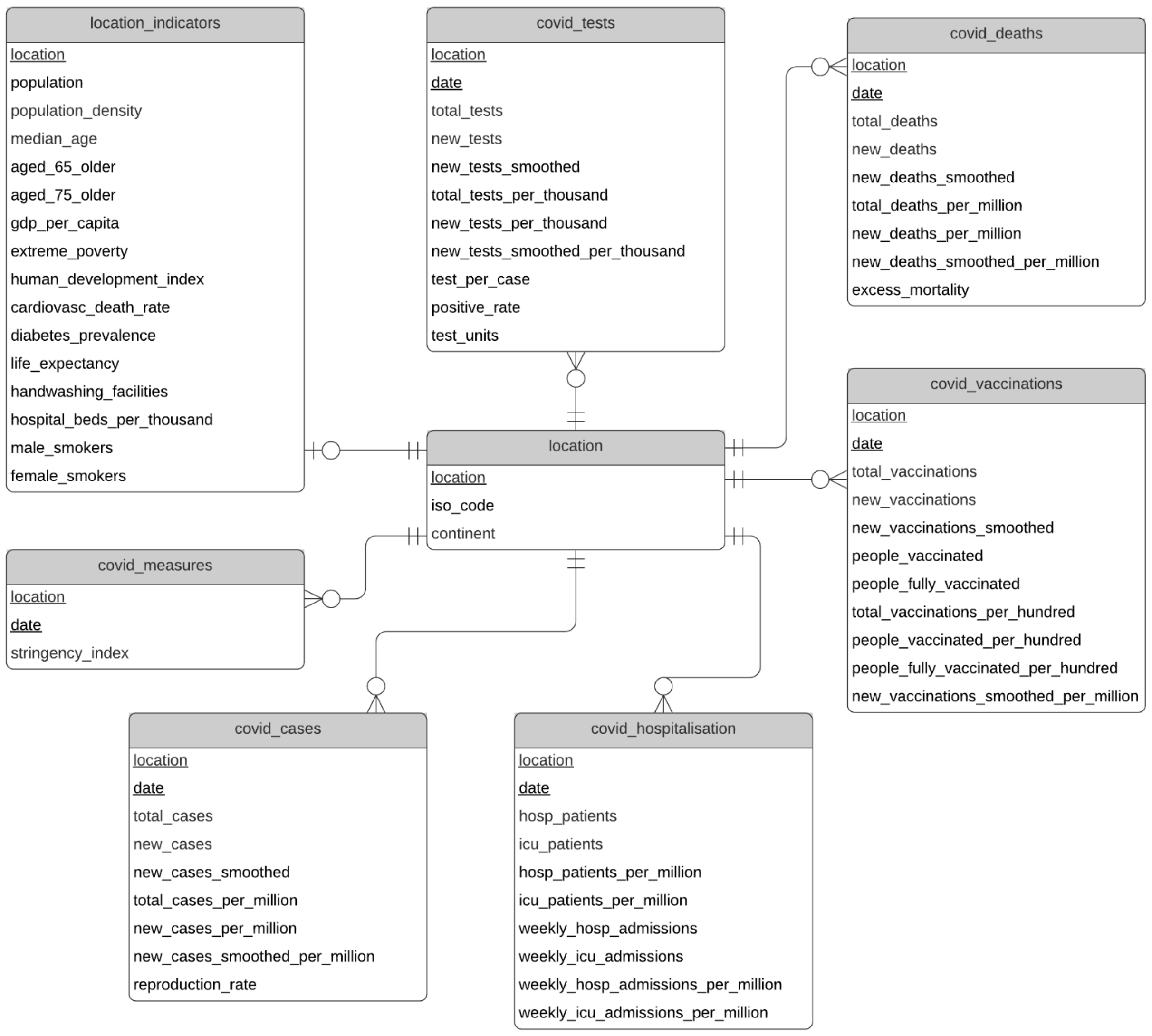
For our non-relational database on mongoDB, the nature of the original data did not require any normalisation or further decomposition. Hence, the implementation of our non-relational model was relatively simpler, as we created a database called "covidData" in MongoDB and uploaded the 3 given JSON files into 3 new collections. All the queries were successfully done directly on these collections.

After implementation, both models were compared to determine the most appropriate one to use to suit our goal of better data management in the context of the COVID-19. When making the comparisons, special attention had to be placed on certain characteristics, such as flexibility, scalability, and data integrity. With the comparisons made, our team was able to make a solid recommendation to WHO as to how to tackle the ongoing infodemic

2. Relational Data Model

2.1 Entity-Relation Model

We created an Entity-Relationship Diagram (ERD) to better understand the different entities and their relationships.



We identified 8 entities: ‘location’, ‘location_indicators’, ‘covid_cases’, ‘covid_measures’, ‘covid_tests’, ‘covid_deaths’, ‘covid_vaccinations’ and ‘covid_hospitalsiation’. The attributes of these entities are extracted from the ‘covid19data’ table.

The ‘location’ entity contains the name, ISO code and continent of the various locations. The attribute ‘location’ is chosen as the primary key as it represents the name, which is unique to each location. As the ‘location’ entity has a 1:1 relationship with 1 entity (‘location_indicators’) and 1:M relationship with the remaining 6 entities, the other entities will have ‘location’ as their foreign key to enforce referential integrity.

The ‘location_indicators’ entity contains the economic and health indicators as well as demographics of each location. ‘location’ is chosen as the primary key as it is required to uniquely identify each record in the entity since each location only has one record.

The remaining entities consist of daily records. Hence, all these entities have ‘location’ and ‘date’ as their composite primary key as both of these attributes are required to uniquely identify each record.

The ‘covid_cases’ entity contains the data relating to the number of Covid-19 cases in each location.

The ‘covid_measures’ entity contains the data relating to the Covid-19 measures in place in each location. It consists of the stringency index, which measures the strictness of the measures in place.

The ‘covid_tests’ entity contains the data relating to the number of Covid-19 tests in each location.

The ‘covid_deaths’ entity contains the data relating to the number of Covid-19 related deaths in each location.

The ‘covid_vaccinations’ entity contains the data relating to the number of Covid-19 vaccinations in each location.

The ‘covid_hospitalisation’ entity contains the data relating to the number of Covid-19 patients and hospital admission rate in each location.

Lastly, some locations do not have any records in certain entities due to the location being a continent (e.g. the location ‘Asia’ does not have any record in ‘covid_measures’) or simply because of lack of information (e.g. the location ‘Niger’ does not have any record in ‘covid_hospitalisation’). Therefore, we established 1(mandatory):M(optional) relationship between ‘location’ and the other entities.

2.2. Functional Dependencies and 3NF

In the ‘covid19data’, the attributes ‘location’ and ‘date’ form the candidate key as they are required to uniquely identify each record in the table.

2.2.1. Partial Dependency

Partial dependency arises when a non-key attribute can be determined using a portion of the candidate key. In the ‘covid19data’ table, the attributes ‘iso_code’ and ‘continent’ can be determined using ‘location’, which is part of the candidate key (‘location’, ‘date’). Similarly, the attributes ‘population’, ‘population_density’, median_age’, ‘aged_65_older’, ‘aged_70_older’, ‘gdp_per_capita’, ‘extreme_poverty’, ‘cardiovasc_death_rate’, ‘diabetes_prevalence’, ‘female_smokers’, ‘male_smokers’, ‘handwashing_facilities’, ‘hospital_beds_per_thousand’, ‘life_expectancy’ and ‘human_development_index’ can be determined with just the ‘location’ key.

Hence, to remove these partial dependencies, we created the 2 tables: ‘location’ with ‘location’, ‘iso_code’ and ‘continents’ as the attributes and ‘location_indicators’ with ‘location’ and the 15 other attributes which are partially dependent on the candidate key in the covid19data table.

2.2.2. Full Dependency

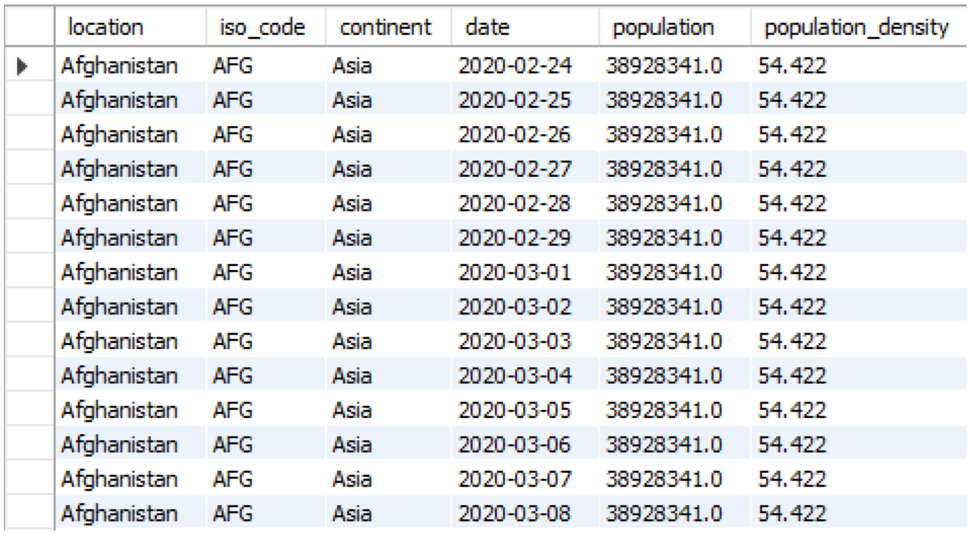
We further split up the ‘covid19data’ table to obtain 6 other tables for the other entities identified. In these 7 tables, the non-key attributes are fully dependent on the candidate key (‘location, ‘date’).

In the ‘location’ and ‘location_indicators’ tables, the non-key attributes are fully dependent on the primary key ‘location’.

2.2.3. Third Normal Form Normalisation

As there are no partial and transitive dependencies in the 8 tables created, we have achieved the third normal form.

The reason for normalising the dataset is because we observed that the ‘covid19data’ table contains redundant repetitive data, which we aimed to eliminate.



location	iso_code	continent	date	population	population_density
Afghanistan	AFG	Asia	2020-02-24	38928341.0	54.422
Afghanistan	AFG	Asia	2020-02-25	38928341.0	54.422
Afghanistan	AFG	Asia	2020-02-26	38928341.0	54.422
Afghanistan	AFG	Asia	2020-02-27	38928341.0	54.422
Afghanistan	AFG	Asia	2020-02-28	38928341.0	54.422
Afghanistan	AFG	Asia	2020-02-29	38928341.0	54.422
Afghanistan	AFG	Asia	2020-03-01	38928341.0	54.422
Afghanistan	AFG	Asia	2020-03-02	38928341.0	54.422
Afghanistan	AFG	Asia	2020-03-03	38928341.0	54.422
Afghanistan	AFG	Asia	2020-03-04	38928341.0	54.422
Afghanistan	AFG	Asia	2020-03-05	38928341.0	54.422
Afghanistan	AFG	Asia	2020-03-06	38928341.0	54.422
Afghanistan	AFG	Asia	2020-03-07	38928341.0	54.422
Afghanistan	AFG	Asia	2020-03-08	38928341.0	54.422

As shown in the table above, there is redundant repetition of data in various attributes of the locations which can be reduced by dividing the ‘covid19data’ table into smaller tables.

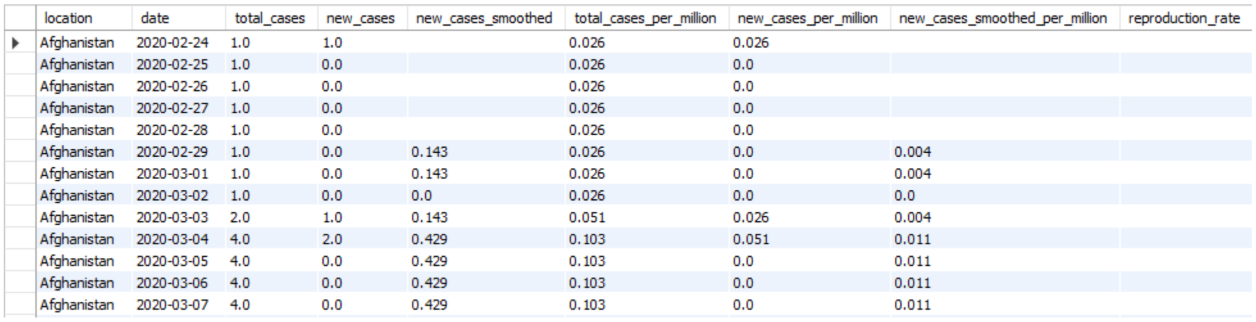
2.2.4. Modifications to datasets

We created the 8 tables by extracting their attributes from the ‘covid19data’ table using ‘CREATE TABLE AS SELECT’ statements. While doing so, we ensured that records without non-key attributes are not selected into the new tables.

```
-- CREATE covid_cases TABLE
CREATE TABLE covid_cases
AS (SELECT
             location,
             date,
             total_cases,
             new_cases,
             new_cases_smoothed,
             total_cases_per_million,
             new_cases_per_million,
             new_cases_smoothed_per_million,
             reproduction_rate
      FROM covid19data
    WHERE
             (total_cases,
             new_cases,
             new_cases_smoothed,
             total_cases_per_million,
             new_cases_per_million,
             new_cases_smoothed_per_million,
             reproduction_rate) <> ('','','','','','','')
      GROUP BY
             location, date);
```

The figure above shows the creation of the ‘covid_cases’ table by selecting the required attributes from the ‘covid19data’ table. The WHERE clause ensures that records with empty cells for all the non-key attributes (‘total_cases’, ‘new_cases’, ‘new_cases_smoothed’, ‘total_cases_per_million’, ‘new_cases_per_million’, ‘new_cases_smoothed_per_million, ‘reproduction_rate’) are not selected into the new table.

Here is a snippet of the ‘covid_cases’ table:



location	date	total_cases	new_cases	new_cases_smoothed	total_cases_per_million	new_cases_per_million	new_cases_smoothed_per_million	reproduction_rate
Afghanistan	2020-02-24	1.0	1.0		0.026	0.026		
Afghanistan	2020-02-25	1.0	0.0		0.026	0.0		
Afghanistan	2020-02-26	1.0	0.0		0.026	0.0		
Afghanistan	2020-02-27	1.0	0.0		0.026	0.0		
Afghanistan	2020-02-28	1.0	0.0		0.026	0.0		
Afghanistan	2020-02-29	1.0	0.0	0.143	0.026	0.0	0.004	
Afghanistan	2020-03-01	1.0	0.0	0.143	0.026	0.0	0.004	
Afghanistan	2020-03-02	1.0	0.0	0.0	0.026	0.0	0.0	
Afghanistan	2020-03-03	2.0	1.0	0.143	0.051	0.026	0.004	
Afghanistan	2020-03-04	4.0	2.0	0.429	0.103	0.051	0.011	
Afghanistan	2020-03-05	4.0	0.0	0.429	0.103	0.0	0.011	
Afghanistan	2020-03-06	4.0	0.0	0.429	0.103	0.0	0.011	
Afghanistan	2020-03-07	4.0	0.0	0.429	0.103	0.0	0.011	

The ‘covid_cases’ table contains 96590 rows whereas the ‘covid19data’ contains 100191 rows, which means that 3601 records in the ‘covid19data’ do not contain any data relating to the number of covid cases. We observed similar results for the remaining 7 tables created.

2.3. Instructions on Deployment

Import ‘GroupProject.sql’ into MySQL Workbench.

Open and run the ‘normalisation_of_relational_database.sql’ script (also shown in Appendix A) to create the 8 new tables.

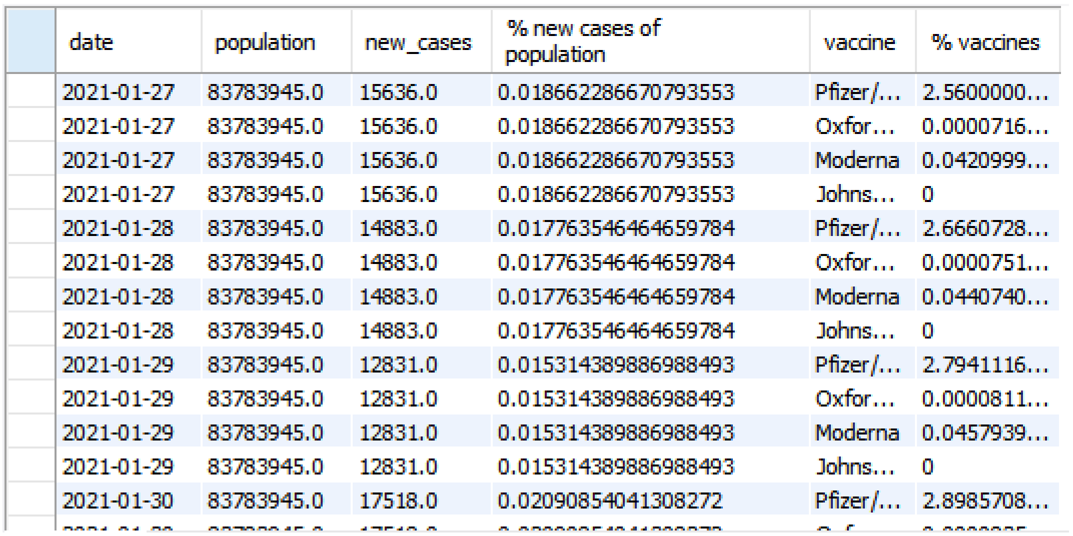
2.4. Relational Database Implementation

To demonstrate the possible insights that can be generated from this database from the implementation, we will explain our team’s queries and provide the output obtained for the more challenging questions, namely question 8, 9, and 10. The script for the relational database implementation in MySQL can be found in Appendix B.

Question 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.

```
SELECT
	`date`,`population`,`new_cases`, (`new_cases`/`population` * 100) as `% new
cases of population`,`vaccine`,(`total_vaccinations`/`population` * 100) as `%
vaccines`
FROM
	(SELECT

`y`.`date`,`y`.`new_cases`,`z`.`population`,`x`.`vaccine`,`x`.`total_vaccinations`
	FROM
		(SELECT * FROM `covid_cases` WHERE `location` = "Germany") y
	LEFT JOIN
		(SELECT * FROM country_vaccinations_by_manufacturer WHERE location =
"Germany") x
	  ON `x`.`date` = `y`.`date`
	LEFT JOIN
		(SELECT `location`, `population` FROM `location_indicators` WHERE
`location` = "Germany") z
	  ON `y`.`location` = `z`.`location`
	) a;
```

For this question, the columns had to be obtained from three different tables; covid_cases, country_vaccinations_by_manufacturer, and location_indicators. The three tables were first joined in a subquery, matching the rows by date and location, to obtain a subtable with the following columns; date, new_cases, population, vaccine, and total_vaccinations. From here, the main query performed calculative operations on resulting columns to calculate the percentages of the new cases and the vaccines of the total population. Columns were renamed for clarity.



date	population	new_cases	% new cases of population	vaccine	% vaccines
2021-01-27	83783945.0	15636.0	0.018662286670793553	Pfizer/...	2.5600000...
2021-01-27	83783945.0	15636.0	0.018662286670793553	Oxfor...	0.0000716...
2021-01-27	83783945.0	15636.0	0.018662286670793553	Moderna	0.0420999...
2021-01-27	83783945.0	15636.0	0.018662286670793553	Johns...	0
2021-01-28	83783945.0	14883.0	0.017763546464659784	Pfizer/...	2.6660728...
2021-01-28	83783945.0	14883.0	0.017763546464659784	Oxfor...	0.0000751...
2021-01-28	83783945.0	14883.0	0.017763546464659784	Moderna	0.0440740...
2021-01-28	83783945.0	14883.0	0.017763546464659784	Johns...	0
2021-01-29	83783945.0	12831.0	0.015314389886988493	Pfizer/...	2.7941116...
2021-01-29	83783945.0	12831.0	0.015314389886988493	Oxfor...	0.0000811...
2021-01-29	83783945.0	12831.0	0.015314389886988493	Moderna	0.0457939...
2021-01-29	83783945.0	12831.0	0.015314389886988493	Johns...	0
2021-01-30	83783945.0	17518.0	0.02090854041308272	Pfizer/...	2.8985708...

(1083 rows returned)

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.

```
WITH daily_new_cases AS (
	SELECT
		location, date, new_cases
	FROM
		covid_cases
	WHERE
		location = 'Germany'
), vaccinations_by_manufacturer AS (
	SELECT
		date, vaccine, total_vaccinations
	FROM
		country_vaccinations_by_manufacturer
	WHERE
		location = 'Germany'
), distinct_vaccines AS (
	SELECT DISTINCT
		vaccine
	FROM
		vaccinations_by_manufacturer
)

SELECT
    daily_new_cases_with_vaccines.*,
    VBM_20.date,
    VBM_20.total_vaccinations,
    VBM_30.date,
    VBM_30.total_vaccinations,
    VBM_40.date,
    VBM_40.total_vaccinations
FROM
    (SELECT
        *
    FROM
        daily_new_cases, distinct_vaccines) daily_new_cases_with_vaccines
        LEFT JOIN
    vaccinations_by_manufacturer VBM_20 ON
DATE_ADD(daily_new_cases_with_vaccines.date,
        INTERVAL 20 DAY) = VBM_20.date
        AND daily_new_cases_with_vaccines.vaccine = VBM_20.vaccine
        LEFT JOIN
    vaccinations_by_manufacturer VBM_30 ON
DATE_ADD(daily_new_cases_with_vaccines.date,
        INTERVAL 30 DAY) = VBM_30.date
        AND daily_new_cases_with_vaccines.vaccine = VBM_30.vaccine
        LEFT JOIN
    vaccinations_by_manufacturer VBM_40 ON
DATE_ADD(daily_new_cases_with_vaccines.date,
        INTERVAL 40 DAY) = VBM_40.date
        AND daily_new_cases_with_vaccines.vaccine = VBM_40.vaccine
ORDER BY daily_new_cases_with_vaccines.date ASC ,
daily_new_cases_with_vaccines.vaccine ASC;
```

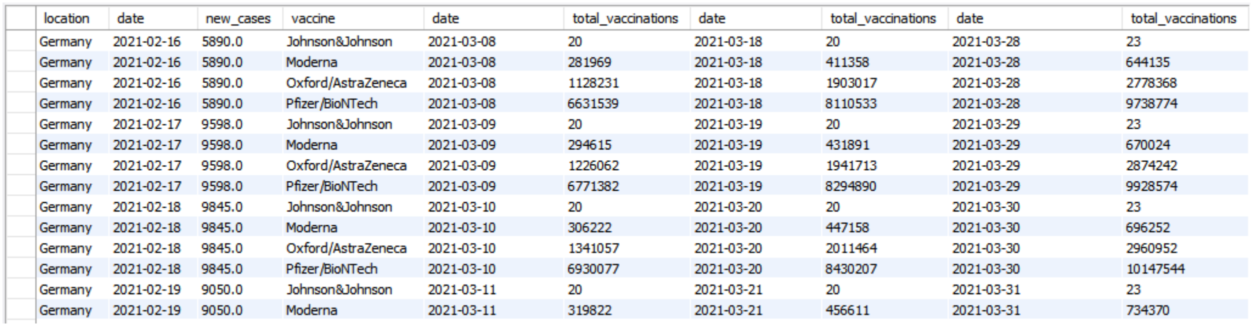
Before we are able to query the desired result, we will first generate 3 helper tables using the WITH statement: *daily_new_cases, vaccinations_by_manufacturer, distinct_vaccines.* Doing this improves the readability as well as allow the same table to be reused and referenced multiple times easily.

Daily_new_cases contains the daily covid cases from Germany. We will retrieve the *columns location, date*, and *new_cases* from it.

Vaccinations_by_manufacturer contains the total_vaccinations count for Germany by date. We will use this table to join the main table, and compare it to get the vaccination count for the 20th, 30th and 40th date.

Distinct_vaccines contains the different types of vaccines in totality that we will be observing.

We first generate *daily_new_cases_with_vaccines* using both our *daily_new_cases* and *distinct_vaccines* tables. The table will then be joined to *Vaccinations_by_manufacturer* thrice, with all the JOIN matching the type of vaccine, and each JOIN matching the data that's 20th, 30th, 40th days ahead using DATE_ADD and INTERVAL XX DAY (where XX is the number of days ahead).



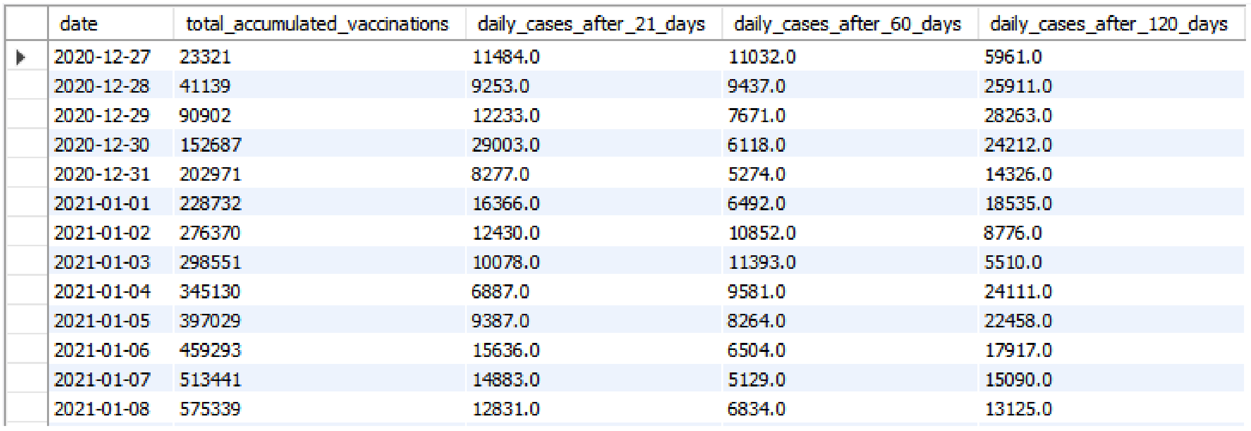
location	date	new_cases	vaccine	date	total_vaccinations	date	total_vaccinations	date	total_vaccinations
Germany	2021-02-16	5890.0	Johnson&Johnson	2021-03-08	20	2021-03-18	20	2021-03-28	23
Germany	2021-02-16	5890.0	Moderna	2021-03-08	281969	2021-03-18	411358	2021-03-28	644135
Germany	2021-02-16	5890.0	Oxford/AstraZeneca	2021-03-08	1128231	2021-03-18	1903017	2021-03-28	2778368
Germany	2021-02-16	5890.0	Pfizer/BioNTech	2021-03-08	6631539	2021-03-18	8110533	2021-03-28	9738774
Germany	2021-02-17	9598.0	Johnson&Johnson	2021-03-09	20	2021-03-19	20	2021-03-29	23
Germany	2021-02-17	9598.0	Moderna	2021-03-09	294615	2021-03-19	431891	2021-03-29	670024
Germany	2021-02-17	9598.0	Oxford/AstraZeneca	2021-03-09	1226062	2021-03-19	1941713	2021-03-29	2874242
Germany	2021-02-17	9598.0	Pfizer/BioNTech	2021-03-09	6771382	2021-03-19	8294890	2021-03-29	9928574
Germany	2021-02-18	9845.0	Johnson&Johnson	2021-03-10	20	2021-03-20	20	2021-03-30	23
Germany	2021-02-18	9845.0	Moderna	2021-03-10	306222	2021-03-20	447158	2021-03-30	696252
Germany	2021-02-18	9845.0	Oxford/AstraZeneca	2021-03-10	1341057	2021-03-20	2011464	2021-03-30	2960952
Germany	2021-02-18	9845.0	Pfizer/BioNTech	2021-03-10	6930077	2021-03-20	8430207	2021-03-30	10147544
Germany	2021-02-19	9050.0	Johnson&Johnson	2021-03-11	20	2021-03-21	20	2021-03-31	23
Germany	2021-02-19	9050.0	Moderna	2021-03-11	319822	2021-03-21	456611	2021-03-31	734370

(2100 rows returned)

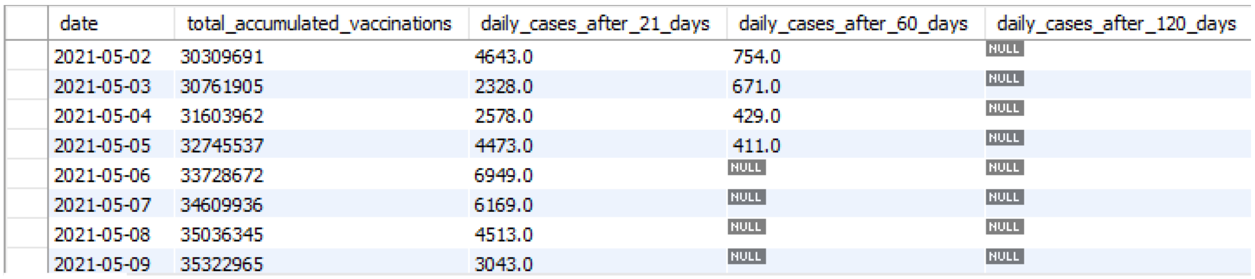
Question 10: Vaccination Effects. Specific to Germany, on a daily basis, based on the total number of accumulated vaccinations (sum of total_vaccinations of each vaccine in a day), generate the daily new cases after 21 days, 60 days, and 120 days.

```
SELECT
    T1.date,
      total_accumulated_vaccinations,
    T2.new_cases AS daily_cases_after_21_days,
    T3.new_cases AS daily_cases_after_60_days,
    T4.new_cases AS daily_cases_after_120_days
FROM
    (SELECT
        location, date,  SUM(total_vaccinations) AS total_accumulated_vaccinations
    FROM
        country_vaccinations_by_manufacturer
    WHERE
        location = 'Germany'
    GROUP BY
        location, date) T1
INNER JOIN
    (SELECT
        date, new_cases FROM covid_cases
    WHERE
        location = 'Germany') T2
ON T2.date = date_add(T1.date, interval 21 day)
LEFT JOIN
    (SELECT
        date, new_cases FROM covid_cases
    WHERE
        location = 'Germany') T3
ON T3.date = date_add(T1.date, interval 60 day)
LEFT JOIN
    (SELECT
        date, new_cases FROM covid_cases
    WHERE
        location = 'Germany') T4
ON T4.date = date_add(T1.date, interval 120 day);
```

To obtain the daily cases after 21, 60 and 120 days respectively, we joined the ‘country_vaccinations_by_manufacturer’ table with the ‘covid_cases’ table four times on the date field, where the ’covid_cases’ tables’ dates are 21, 60 and 120 days after the date matched on the ‘country_vaccinations_by_manufactuer’ table. For each of ‘covid_cases’ tables joined, we selected the new_cases fields whereas for the ‘country_vaccinations_by_manufacturer’ table, we selected the sum of the total_vaccinations field for each day as the accumulated_vaccinations as well as the date. We used INNER_JOIN for the ‘covid_cases’ where its date is 21 days after the date in the ‘country_vaccinations_by_manufacturer’ table so that only the rows with the record of new_cases after 21 days are shown. Afterwards, we used LEFT JOIN for the remaining 2 ‘covid_cases’ tables’ where their dates are 60 and 120 days after the date in the ‘country_vaccinations_by_manufacturer’ table so that all the rows from the first inner join are shown even if there is no record for new_cases after 60 or/and 120 days.



date	total_accumulated_vaccinations	daily_cases_after_21_days	daily_cases_after_60_days	daily_cases_after_120_days
2020-12-27	23321	11484.0	11032.0	5961.0
2020-12-28	41139	9253.0	9437.0	25911.0
2020-12-29	90902	12233.0	7671.0	28263.0
2020-12-30	152687	29003.0	6118.0	24212.0
2020-12-31	202971	8277.0	5274.0	14326.0
2021-01-01	228732	16366.0	6492.0	18535.0
2021-01-02	276370	12430.0	10852.0	8776.0
2021-01-03	298551	10078.0	11393.0	5510.0
2021-01-04	345130	6887.0	9581.0	24111.0
2021-01-05	397029	9387.0	8264.0	22458.0
2021-01-06	459293	15636.0	6504.0	17917.0
2021-01-07	513441	14883.0	5129.0	15090.0
2021-01-08	575339	12831.0	6834.0	13125.0



date	total_accumulated_vaccinations	daily_cases_after_21_days	daily_cases_after_60_days	daily_cases_after_120_days
2021-05-02	30309691	4643.0	754.0	NULL
2021-05-03	30761905	2328.0	671.0	NULL
2021-05-04	31603962	2578.0	429.0	NULL
2021-05-05	32745537	4473.0	411.0	NULL
2021-05-06	33728672	6949.0	NULL	NULL
2021-05-07	34609936	6169.0	NULL	NULL
2021-05-08	35036345	4513.0	NULL	NULL
2021-05-09	35322965	3043.0	NULL	NULL

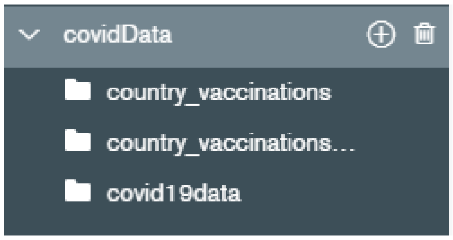
(169 rows returned)

3. Non Relational Data Model

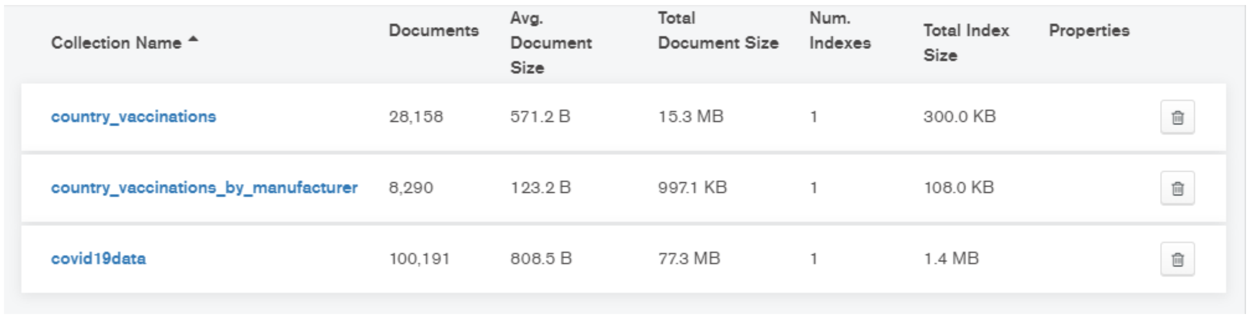
After analysing the given data, we chose not to rework the three given collections. We experimented with different combinations of aggregations, but did not find that the collections we created would be as encompassing as the collections we already had.

3.1. Instructions on Deployment

Import the json files as collections into a database named “covidData” in MongoDB.



covidData
- country_vaccinations
- country_vaccinations...
- covid19data



Collection Name ⏶	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
country_vaccinations	28,158	571.2 B	15.3 MB	1	300.0 KB	
country_vaccinations_by_manufacturer	8,290	123.2 B	997.1 KB	1	108.0 KB	
covid19data	100,191	808.5 B	77.3 MB	1	1.4 MB	

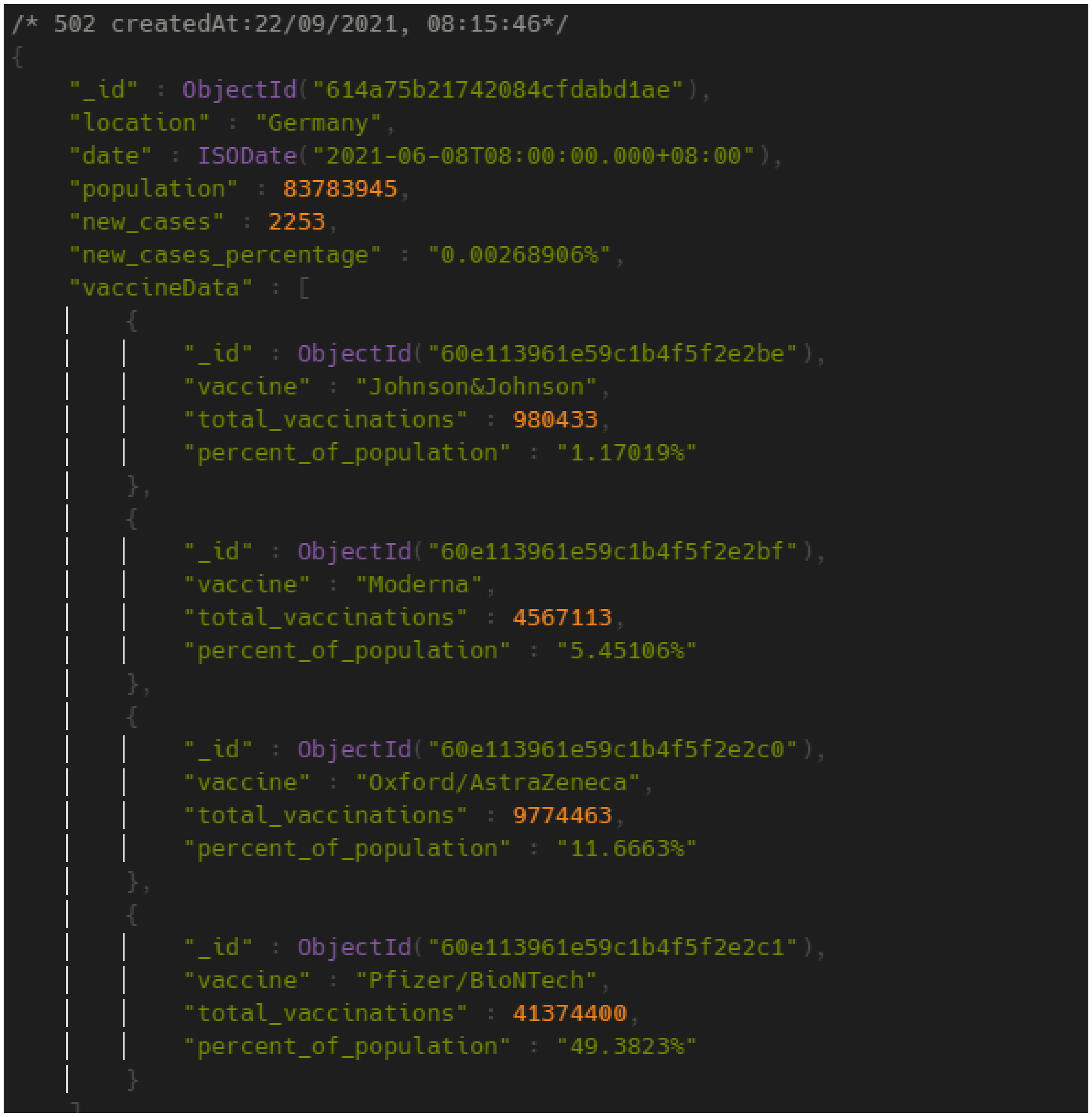
3.2. Non Relational Database Implementation

Similar to our relational implementation section, we will only be discussing the more challenging queries, namely question 18, 19, and 20. The rest of the script for non relational database implementation can be found in Appendix C.

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.covid19data.aggregate(
    [
        {$match:{"location":"Germany"}},
        {$project:{"location":1,
                    date:{$convert:{input: "$date",to:"date"}},
                    "population":{$convert:{input: "$population",to:"double"}},
                    "new_cases":{$ifNull: [ {$convert:{input:
"$new_cases",to:"double"}}, 0 ]}
        }},
        {$project:{"location":1,
                    date:1,
                    "population":1,
                    "new_cases":1,
                    "new_cases_percentage":

{$concat:[{$convert:{input:{$multiply:[{$divide:["$new_cases","$population"]},100]}
,to:"string"}},"%"]}
        }},
        {$lookup:
            {
                from:"country_vaccinations_by_manufacturer",
                let: {date:"$date",population:"$population"},
                pipeline:[
                    {$match:{"location":"Germany"}},
                    {$project:{"location":1,
                                date:{$convert:{input: "$date",to:"date"}},
                                "vaccine":1,

"total_vaccinations":{$convert:{input:"$total_vaccinations",to:"double"}},
                                "percent_of_population":

{$concat:[{$convert:{input:{$multiply:[{$divide:[{$convert:{input:"$total_vaccinati
ons",to:"double"}},"$$population"]},100]},to:"string"}},"%"]}
                    }},
                    {$match:
                        {$expr:
                            {$eq:["$date","$$date"]}
                        }
                    },

{$project:{"vaccine":1,"total_vaccinations":1,"percent_of_population":1}}
                ],
                as: "vaccineData"
            }
        }
    ]
);
```

First, we obtained the necessary fields, and made calculations (such as the daily percentage of new cases over population) from “covid19data”. Then, we performed a lookup to our foreign collection “country_vaccinations_by_manufacturer”, which contains information about total vaccinations for each vaccine daily. We extracted the necessary fields and calculations from that collection, and finally “joined” those fields to our local collection on date. The result is a new “vaccineData” array field joined to the local collection, containing information regarding total vaccinations as a percentage of the population for each vaccine. This table thus gives us insight into herd immunity estimation, since we expect the percentage of new cases to fall as the total vaccinations percentage increases.



```
/* 502 createdAt:22/09/2021, 08:15:46*/
{
    "_id" : ObjectId("614a75b21742084cfdabd1ae"),
    "location" : "Germany",
    "date" : ISODate("2021-06-08T08:00:00.000+08:00"),
    "population" : 83783945,
    "new_cases" : 2253,
    "new_cases_percentage" : "0.00268906%",
    "vaccineData" : [
        {
            "_id" : ObjectId("60e113961e59c1b4f5f2e2be"),
            "vaccine" : "Johnson&Johnson",
            "total_vaccinations" : 980433,
            "percent_of_population" : "1.17019%"
        },
        {
            "_id" : ObjectId("60e113961e59c1b4f5f2e2bf"),
            "vaccine" : "Moderna",
            "total_vaccinations" : 4567113,
            "percent_of_population" : "5.45106%"
        },
        {
            "_id" : ObjectId("60e113961e59c1b4f5f2e2c0"),
            "vaccine" : "Oxford/AstraZeneca",
            "total_vaccinations" : 9774463,
            "percent_of_population" : "11.6663%"
        },
        {
            "_id" : ObjectId("60e113961e59c1b4f5f2e2c1"),
            "vaccine" : "Pfizer/BioNTech",
            "total_vaccinations" : 41374400,
            "percent_of_population" : "49.3823%"
        }
```

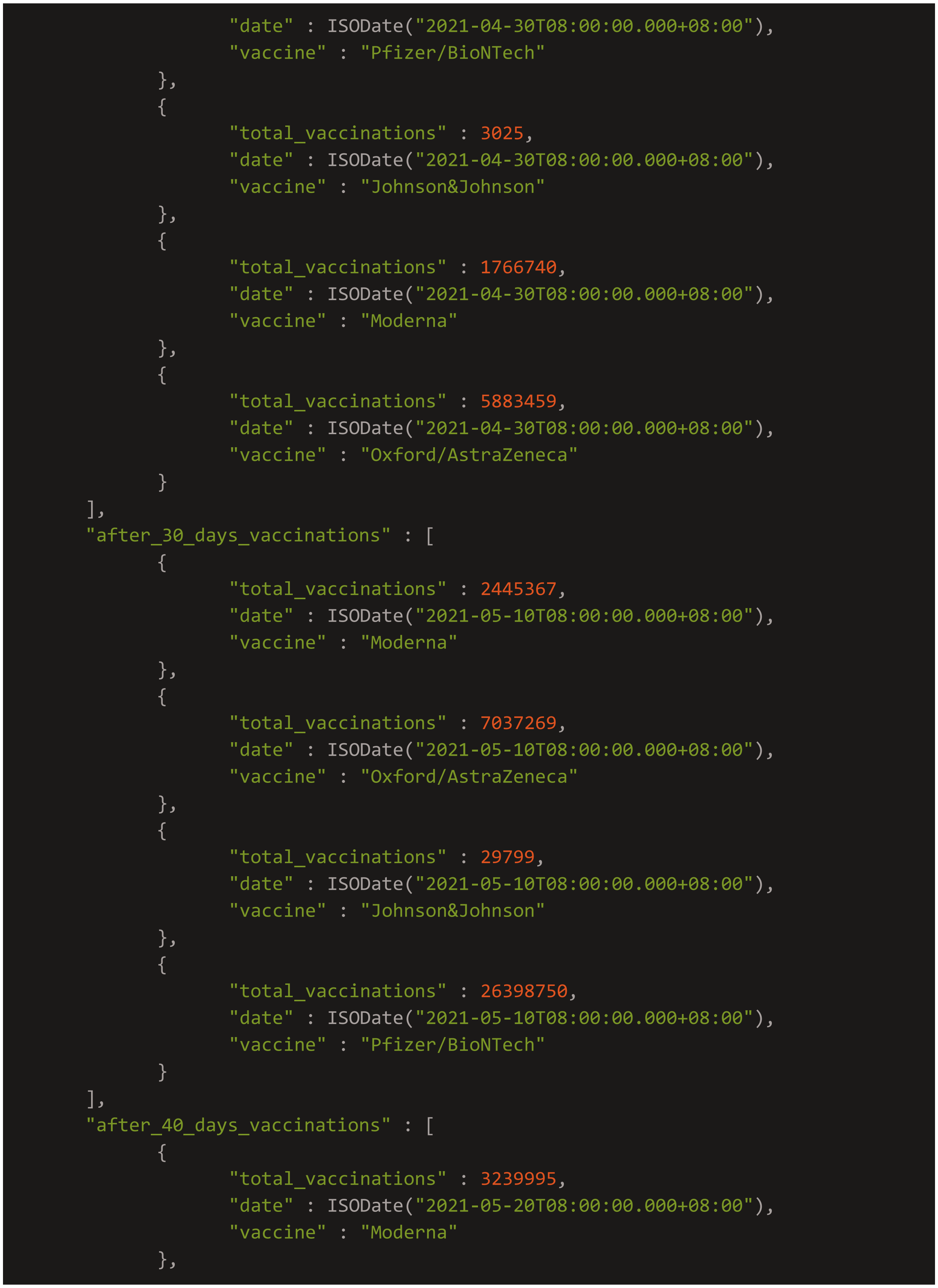
(528 docs fetched)

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

```
db.covid19data.aggregate([
    { $match: { location: "Germany" } },
    { $group: { _id: [{ location: "$location" }, { new_cases_date: { $convert: {
input: "$date", to: "date" } } }, { new_cases: "$new_cases" }] } },
    {
        $project: {
            _id: 0, location: { $arrayElemAt: ["$_id.location", 0] },
new_cases_date: { $arrayElemAt: ["$_id.new_cases_date", 0] },
            new_cases: { $arrayElemAt: ["$_id.new_cases", 0] }
        }
    },
    {
        $lookup: {
            from: "country_vaccinations_by_manufacturer",
            let: { new_cases_date: "$new_cases_date" },
            pipeline: [
                { $match: { location: "Germany" } },
                {
                    $group: {
                        _id: { date: { $convert: { input: "$date", to: "date" } },
vaccine: "$vaccine" },
                        total_vaccinations: { $max: { $convert: { input:
"$total_vaccinations", to: "int" } } }
                    }
                },
                { $project: { date: "$_id.date", vaccine: "$_id.vaccine",
total_vaccinations: 1 } },
                { $match: { $expr: { $eq: ["$date", { $dateAdd: { startDate:
"$$new_cases_date", unit: "day", amount: 20 } }] } } },
                { $project: { _id: 0, date: 1, vaccine: 1, total_vaccinations: 1 }
},
                { $sort: { date: 1 } }], as: "after_20_days_vaccinations"
        }
    },
    {
        $lookup: {
            from: "country_vaccinations_by_manufacturer",
            let: { new_cases_date: "$new_cases_date" },
            pipeline: [
                { $match: { location: "Germany" } },
                {
                    $group: {
                        _id: { date: { $convert: { input: "$date", to: "date" } },
vaccine: "$vaccine" },
                        total_vaccinations: { $max: { $convert: { input:
"$total_vaccinations", to: "int" } } }
                    }
                },
                { $project: { date: "$_id.date", vaccine: "$_id.vaccine",
total_vaccinations: 1 } },
                { $match: { $expr: { $eq: ["$date", { $dateAdd: { startDate:
"$$new_cases_date", unit: "day", amount: 30 } }] } } },
                { $project: { _id: 0, date: 1, vaccine: 1, total_vaccinations: 1 }
},
                { $sort: { date: 1 } }], as: "after_30_days_vaccinations"
        }
    },
    {
        $lookup: {
            from: "country_vaccinations_by_manufacturer",
            let: { new_cases_date: "$new_cases_date" },
            pipeline: [
                { $match: { location: "Germany" } },
                {
                    $group: {
                        _id: { date: { $convert: { input: "$date", to: "date" } },
vaccine: "$vaccine" },
                        total_vaccinations: { $max: { $convert: { input:
"$total_vaccinations", to: "int" } } }
                    }
                },
                { $project: { date: "$_id.date", vaccine: "$_id.vaccine",
total_vaccinations: 1 } },
                { $match: { $expr: { $eq: ["$date", { $dateAdd: { startDate:
"$$new_cases_date", unit: "day", amount: 40 } }] } } },
                { $project: { _id: 0, date: 1, vaccine: 1, total_vaccinations: 1 }
},
                { $sort: { date: 1 } }], as: "after_40_days_vaccinations"
        }
    },
    { $project: { location: 1, new_cases_date: 1, new_cases: 1,
after_20_days_vaccinations: 1, after_30_days_vaccinations: 1,
after_40_days_vaccinations: 1 } }
])
```

Firstly, we projected the necessary fields and data type conversions from "covid19data", our local collection. Then, we perform lookups to our foreign collection "country_vaccinations_by_manufacturer", with the "join" condition being date plus 20 days, 30 days and 40 days respectively for each of the three lookups performed. The result is 3 arrays for 20 days, 30 days, and 40 days, concatenated to the local fields, and each array contains information of the total vaccination numbers of each vaccine for the respective number of days plus the "new_cases_date" local field.

```
{
        "location" : "Germany",
        "new_cases_date" : ISODate("2021-04-10T08:00:00.000+08:00"),
        "new_cases" : "18728.0",
        "after_20_days_vaccinations" : [
                {
                        "total_vaccinations" : 22057392,
```



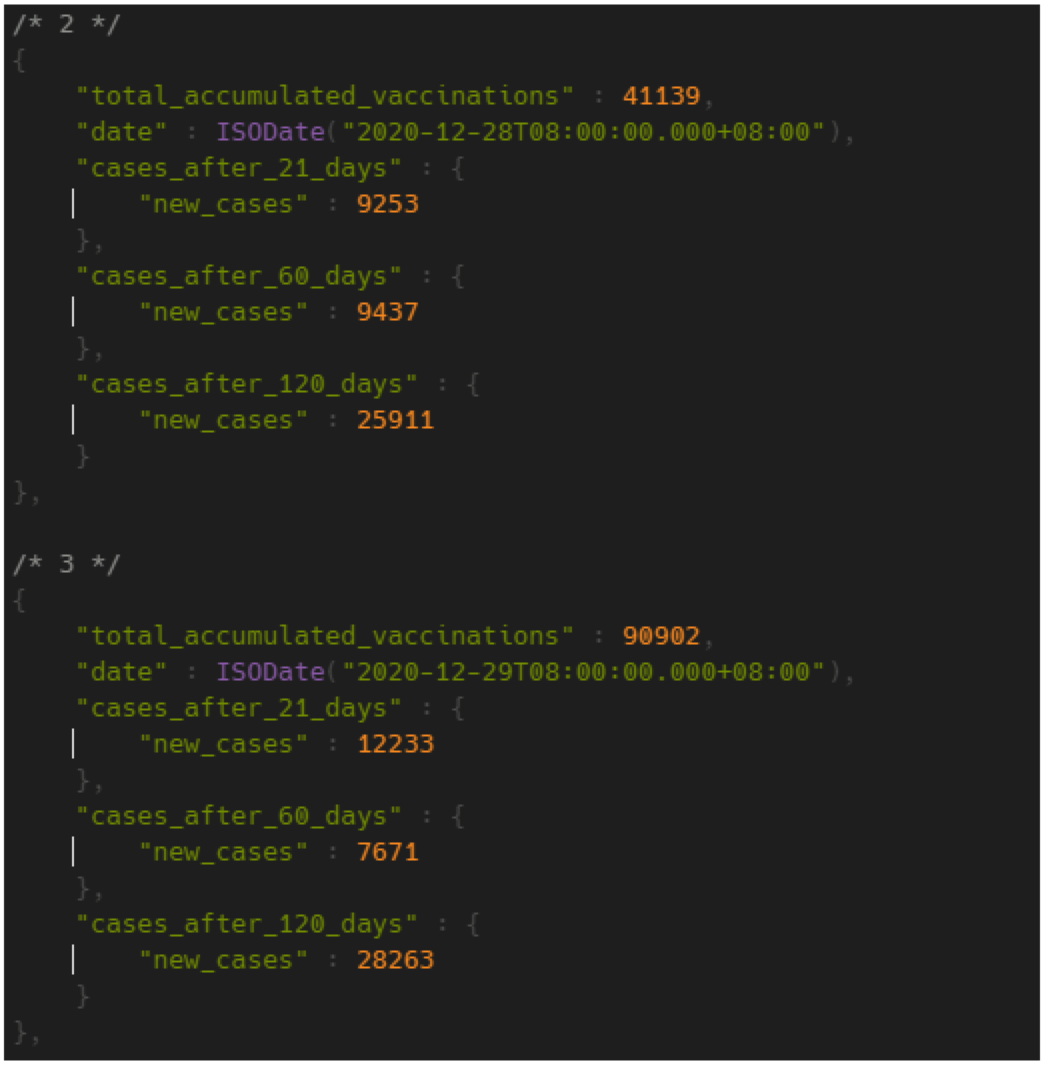
```
                        "date" : ISODate("2021-04-30T08:00:00.000+08:00"),
                        "vaccine" : "Pfizer/BioNTech"
                },
                {
                        "total_vaccinations" : 3025,
                        "date" : ISODate("2021-04-30T08:00:00.000+08:00"),
                        "vaccine" : "Johnson&Johnson"
                },
                {
                        "total_vaccinations" : 1766740,
                        "date" : ISODate("2021-04-30T08:00:00.000+08:00"),
                        "vaccine" : "Moderna"
                },
                {
                        "total_vaccinations" : 5883459,
                        "date" : ISODate("2021-04-30T08:00:00.000+08:00"),
                        "vaccine" : "Oxford/AstraZeneca"
                }
        ],
        "after_30_days_vaccinations" : [
                {
                        "total_vaccinations" : 2445367,
                        "date" : ISODate("2021-05-10T08:00:00.000+08:00"),
                        "vaccine" : "Moderna"
                },
                {
                        "total_vaccinations" : 7037269,
                        "date" : ISODate("2021-05-10T08:00:00.000+08:00"),
                        "vaccine" : "Oxford/AstraZeneca"
                },
                {
                        "total_vaccinations" : 29799,
                        "date" : ISODate("2021-05-10T08:00:00.000+08:00"),
                        "vaccine" : "Johnson&Johnson"
                },
                {
                        "total_vaccinations" : 26398750,
                        "date" : ISODate("2021-05-10T08:00:00.000+08:00"),
                        "vaccine" : "Pfizer/BioNTech"
                }
        ],
        "after_40_days_vaccinations" : [
                {
                        "total_vaccinations" : 3239995,
                        "date" : ISODate("2021-05-20T08:00:00.000+08:00"),
                        "vaccine" : "Moderna"
                },
                {
                        "total_vaccinations" : 32075707,
                        "date" : ISODate("2021-05-20T08:00:00.000+08:00"),
                        "vaccine" : "Pfizer/BioNTech"
                },
                {
                        "total_vaccinations" : 98647,
                        "date" : ISODate("2021-05-20T08:00:00.000+08:00"),
                        "vaccine" : "Johnson&Johnson"
                },
                {
                        "total_vaccinations" : 8333361,
                        "date" : ISODate("2021-05-20T08:00:00.000+08:00"),
                        "vaccine" : "Oxford/AstraZeneca"
                }
        ]
    }
```

(528 docs fetched)

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

```
db.country_vaccinations_by_manufacturer.aggregate([
    {$match: {location: "Germany"}},
    {$group: {_id:{date: {$convert: {input: "$date", to: "date"}}},
    total_accumulated_vaccinations: {$sum: {$convert:{input: "$total_vaccinations",
to: "double"}}}}},
    {$project: {_id:0,country: "$_id.country" , date:"$_id.date",
total_accumulated_vaccinations: 1}},
    {
        $lookup: {
            from: "covid19data",
            let: {
                covid_date: "$date",
            },
            pipeline: [
                {$match: {location: "Germany"}},
                {

                    $project: {
                    location:1, date: {$convert: {input:"$date", to: "date"}},
new_cases:{$convert: {input:"$new_cases", to: "double"}}
                    }
                },
                {
                        $match:{$or:[
                            {$expr: {$eq: ["$date",{$dateAdd: {startDate: 
"$$covid_date", unit: "day", amount: 21}}]}},
                            {$expr: {$eq: ["$date",{$dateAdd: {startDate: 
"$$covid_date", unit: "day", amount: 60}}]}},
                            {$expr: {$eq: ["$date",{$dateAdd: {startDate: 
"$$covid_date", unit: "day", amount: 120}}]}}
                            ]
                        }
                },
                {
                    $project: {_id:0, new_cases:1}
                }
            ],
            as: "new_cases_list"
        }
    },
    {$project: 
{total_accumulated_vaccinations:1,date:1,cases_after_21_days:{$arrayElemAt:["$new_c
ases_list",0]}, cases_after_60_days:{$arrayElemAt:["$new_cases_list",1]}, 
cases_after_120_days:{$arrayElemAt:["$new_cases_list",2]}}},
    {$sort: {date:1}}
])
```

After subsetting our documents to those from Germany, and converting and projecting the needed variables such as total_vaccinations from the country_by_manufacturers collection, we used the lookup function to append the number of new cases for the requested periods for each day, using $expr, $eq, and $dateAdd to specify the intervals of 21, 60, and 120 days for each document.



```
/* 2 */
{
    "total_accumulated_vaccinations" : 41139,
    "date" : ISODate("2020-12-28T08:00:00.000+08:00"),
    "cases_after_21_days" : {
    |   "new_cases" : 9253
    },
    "cases_after_60_days" : {
    |   "new_cases" : 9437
    },
    "cases_after_120_days" : {
    |   "new_cases" : 25911
    }
},

/* 3 */
{
    "total_accumulated_vaccinations" : 90902,
    "date" : ISODate("2020-12-29T08:00:00.000+08:00"),
    "cases_after_21_days" : {
    |   "new_cases" : 12233
    },
    "cases_after_60_days" : {
    |   "new_cases" : 7671
    },
    "cases_after_120_days" : {
    |   "new_cases" : 28263
    }
},
```

(186 docs fetched)

4. Inconsistency between Relational Data Model and Non Relational Data Model

Database	
Relational Database (MySQL)	Non-Relational Database (MongoDB)
country_vaccinations	country_vaccinations
country_vaccinations_by_manufacturer	country_vaccinations_by_manufacturer
covid_cases	covid19data
covid_deaths	
covid_hospitalisation	
covid_measures	
covid_tests	
covid_vaccinations	

To meet 3NF standards, a total of 8 tables were used for relational database implementation. 3 collections were used for non-relational database implementation since the defining of relations was not needed.

One difference between the two structures is the use of normalisation for the relational database. This was done to remove redundancies. Tables in relational databases have to be properly linked to prevent data inconsistencies when making changes.

Further comparisons between the two types of databases will be made in the next section.

5. Recommendation to WHO

We will propose a recommendation to WHO based on a few factors, such as the current implementation of the relational and non-relational dataset above, our understanding of the two types of data modelling approaches, and the assumed needs and requirements of WHO, as well as the nature of the data.

5.1. Comparisons between relational DB and non-relational DB

5.1.1. Flexibility

Relational databases have strict schemas. This means that we need to know the structure of the data beforehand. Data follows the same data type. For example, a "weight" attribute can only be an integer or float, but not both. As such, if there is a change in requirements, such as changing the data type of an attribute, or adding/removing a column, we have to perform some form of migration to ensure the data adheres to the schema after the change.

In contrast, non-relational databases do not follow any schemas, which is also why they are described as “schemaless”. This means we do not need to know the structure of the data beforehand. Data does not have to follow the same data type. For example, an attribute “weight” can have values such as “5kg”, “500mg”, “50.0”. As such, no changes need to be made to the database if there are any changes in requirements with regards to the data - we can update/add/remove keys without breaking anything.

5.1.2. Scalability

Relational databases are vertically scalable. This means that they can only be scaled by using more powerful hardware such as adding more RAM or using a faster-computing chip. This can be quite a costly deal for processing large batches of data.

NoSQL databases are designed to support seamless, online horizontal scalability, which means it is done by adding more servers. This is generally cheaper when compared to vertical scaling.

5.1.3. Data storage footprint

As relational databases support joins, a well-designed schema will have no duplicated information. This allows the memory required to be smaller and thus cheaper. However, because the structure of the database is pre-defined, a record with NULL values will cost the same amount of memory as one without NULL values, making data with many blank values less cost-efficient.

Non-relational databases (MongoDB in our case) has high memory usage, it stores key names for each value pair. Due to no functionality of joins, there is data repetition. This will result in unnecessary usage of memory, which increases exponentially especially with the amount of data.

5.1.4. Data Integrity

Data Integrity is a key feature of relational databases. This is made possible because relational databases guaranteed ACID (Atomic, Consistency, Isolation and Durability) transactions. Thus, we can better ensure the data is validated across all the tables and there's no duplicate, unrelated or unauthorized data inserted in the system e.g. in the case of a repeated query being executed twice due to issues like network fault.

In contrast, most non-relational databases support BASE transactions (Basically Available, Soft-state, Eventually consistent). This means data is replicated on many storage systems (to ensure high availability) and some values may only be partially correct.

5.1.5. Suitability for Data Analysis

Relational databases are more suitable and easier for well-structured data such as text and numbers that can fit on an excel sheet easily.

In contrast, non-relational databases are more suitable for semi-structured data, such as social media or geographical data which requires large amounts of text mining or image processing.

5.2. Assumption about the needs and requirements of WHO

As COVID-19 is a global pandemic that greatly impacts every single country, WHO needs to be able to make the most correct decisions based on the data they have, as a wrong decision might result in the further loss of lives of millions. As such, **access to accurate and consistent data will be of utmost importance**.

5.3. Recommendation and justification

Given the above assumption that we value data accuracy the most, we would recommend the usage of relational databases for WHO as relational databases triumph over non-relational databases over data integrity.

In addition, as mentioned by WHO, COVID-19 was first reported about 2 years ago on 31 December 2019. This means that the data structure and requirements of the type of data needed to be stored are very unlikely to change. As such, the flexibility of non-relational databases is negligible.

Thirdly, while it may be true that scaling non-relational databases is indeed more cost-effective, we believe that the cost savings will be less of a priority for WHO when compared to the lives of millions in the world. Also, as the access to the dataset does not need to be available down to the seconds and minutes, the need to scale might not even be a concern altogether. In addition, with the current data set presented, it is unlikely that the cost-saving will even be significant to be a factor of consideration. Furthermore, due to the smaller data footprint of relational databases, one could even argue that relational databases might even be cheaper at this scale.

Lastly, given the nature of the data (mostly text and integers), it will be easier to write and perform queries on relational databases.

5.4. Limitations

At the moment, all the data in the "covid19data" table are all "text" data types (as shown below), regardless of what they should be.

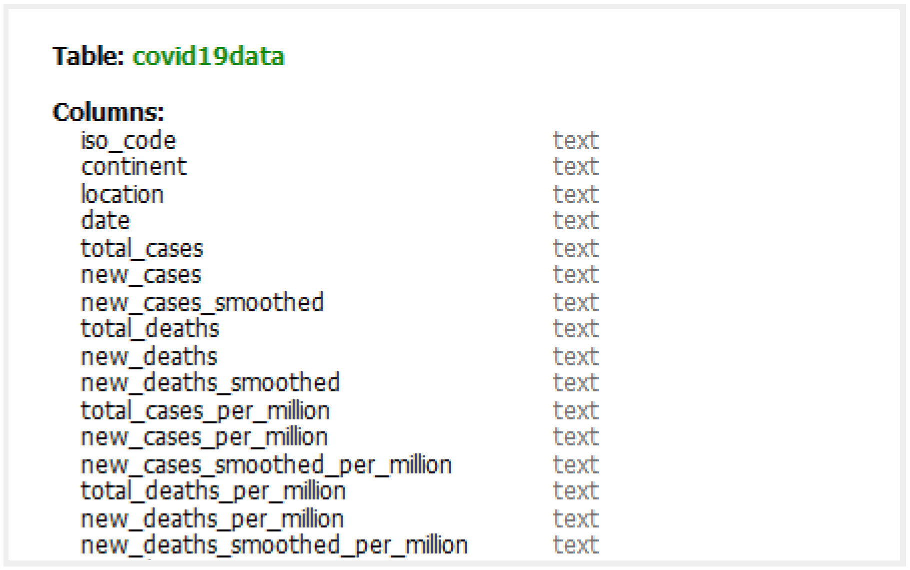


Table: covid19data

Columns:

iso_code	text
continent	text
location	text
date	text
total_cases	text
new_cases	text
new_cases_smoothed	text
total_deaths	text
new_deaths	text
new_deaths_smoothed	text
total_cases_per_million	text
new_cases_per_million	text
new_cases_smoothed_per_million	text
total_deaths_per_million	text
new_deaths_per_million	text
new_deaths_smoothed_per_million	text

As such, to perform query and analysis such as aggregated total value, or minimum/maximum value of certain attributes, we have to do datacasting (converting the data type of an attribute to the specified data type). While Database Management System (DBMS) will implicitly cast these data, we can't assume that it will do them correctly due to the ambiguity of the data.

6. Conclusion

To summarise, we began our project by looking through the data for any functional dependencies or irrelevant entries. To eliminate these, we normalised the datasets to 3NF, giving us a total of 11 tables. We then implemented both our relational and non-relational databases, and performed various queries showing the capabilities of both types. This process allowed us to identify some inconsistencies between the two data models.

After much comparison and evaluation, we were able to conclude that while it did have its limitations, the relational data model was the better one for WHO’s use, due its higher accuracy and ease of use, that outweigh other factors such as reducing costs and having more flexibility.

Overall, our team believes that our approach to the abundant amounts of information related to COVID-19 is effective, and will allow WHO or whoever implements our method to efficiently manage the data, enabling them to gain insights that could positively impact the world.

References

Eessaar, E. (2016, February). The Database Normalization Theory and the Theory of Normalized Systems: Finding a Common Ground. ResearchGate. Retrieved November 11, 2021, from https://www.researchgate.net/publication/297731569_The_Database_Normalization_Theory_and_the_Theory_of_Normalized_Systems_Finding_a_Common_Ground.

Gyorodi, C. (2015). (PDF) A comparative study of relational and Non-Relational database models in a web- based application. ResearchGate. Retrieved November 11, 2021, from https://www.researchgate.net/publication/285752873_A_Comparative_Study_of_Relational_and_Non-Relational_Database_Models_in_a_Web-_Based_Application.

IBM. (n.d.). ACID Properties of transactions. Acid properties of transactions. Retrieved November 11, 2021, from https://www.ibm.com/docs/en/cics-ts/5.4?topic=processing-acid-properties-transactions.

Stankova, R. (2018). (PDF) Design and Analysis of a relational database for behavioral experiments Data Processing. ResearchGate. Retrieved November 11, 2021, from https://www.researchgate.net/publication/323466947_Design_and_Analysis_of_a_Relational_Database_for_Behavioral_Experiments_Data_Processing.

Talend. (n.d.). What is data integrity and why is it important? Talend. Retrieved November 11, 2021, from https://www.talend.com/resources/what-is-data-integrity/.

Appendix A: MySQL Script for Normalisation

```
-- CREATE location TABLE
CREATE TABLE location
AS (SELECT
        location,
        iso_code,
        continent
    FROM
        covid19data
    GROUP BY
        location);

-- Create TABLE location_indicators
CREATE TABLE location_indicators
AS (SELECT
        location,
        population,
        population_density,
        median_age, aged_65_older,
        aged_70_older,
        gdp_per_capita,
        extreme_poverty,
        cardiovasc_death_rate,
        diabetes_prevalence,
        female_smokers,
        male_smokers,
        handwashing_facilities,
        hospital_beds_per_thousand,
        life_expectancy,
        human_development_index
    FROM covid19data
    WHERE
        (population,
        population_density,
        median_age,
        aged_65_older,
        aged_70_older,
        gdp_per_capita,
        extreme_poverty,
        cardiovasc_death_rate,
        diabetes_prevalence,
        female_smokers,
        male_smokers,
        handwashing_facilities,
        hospital_beds_per_thousand,
        life_expectancy,
        human_development_index) <> ('','','','','','','','','','','','','','','')
    GROUP BY
        location);

-- CREATE covid_cases TABLE
CREATE TABLE covid_cases
AS (SELECT
        location,
        date,
        total_cases,
        new_cases,
        new_cases_smoothed,
        total_cases_per_million,
        new_cases_per_million,
        new_cases_smoothed_per_million,
        reproduction_rate
    FROM covid19data
    WHERE
        (total_cases,
        new_cases,
        new_cases_smoothed,
        total_cases_per_million,
        new_cases_per_million,
        new_cases_smoothed_per_million,
        reproduction_rate) <> ('','','','','','','')
    GROUP BY
        location, date);


-- CREATE covid_deaths TABLE
CREATE TABLE covid_deaths
AS (SELECT
        location,
        date,
        total_deaths,
        new_deaths,
        new_deaths_smoothed,
        total_deaths_per_million,
        new_deaths_per_million,
        new_deaths_smoothed_per_million,
        excess_mortality
    FROM covid19data
    WHERE
        (total_deaths,
        new_deaths,
        new_deaths_smoothed,
        total_deaths_per_million,
```

```sql
        new_deaths_per_million,
        new_deaths_smoothed_per_million,
        excess_mortality) <> ('','','','','','','')
    GROUP BY
        location,
        date);

-- CREATE covid_hospitalisation TABLE
CREATE TABLE covid_hospitalisation
AS (SELECT
        location,
        date,
        icu_patients,
        icu_patients_per_million,
        hosp_patients,
        hosp_patients_per_million,
        weekly_icu_admissions,
        weekly_icu_admissions_per_million,
        weekly_hosp_admissions,
        weekly_hosp_admissions_per_million
    FROM covid19data
    WHERE
        (icu_patients,
        icu_patients_per_million,
        hosp_patients,
        hosp_patients_per_million,
        weekly_icu_admissions,
        weekly_icu_admissions_per_million,
        weekly_hosp_admissions,
        weekly_hosp_admissions_per_million) <> ('','','','','','','','')
    GROUP BY
        location,
        date);

-- CREATE covid_tests TABLE
CREATE TABLE covid_tests
AS (SELECT
        location,
        date,
        new_tests,
        total_tests,
        total_tests_per_thousand,
        new_tests_per_thousand,
        new_tests_smoothed,
        new_tests_smoothed_per_thousand,
        positive_rate,
        tests_per_case,
        tests_units
    FROM covid19data
    WHERE
        (new_tests,
        total_tests,
        total_tests_per_thousand,
        new_tests_per_thousand,
        new_tests_smoothed,
        new_tests_smoothed_per_thousand,
        positive_rate,
        tests_per_case,
        tests_units) <> ('','','','','','','','','')
    GROUP BY
        location,
        date);

-- CREATE covid_vaccinations TABLE
CREATE TABLE covid_vaccinations
AS (SELECT
    location,
    date,
    total_vaccinations,
    people_vaccinated,
    people_fully_vaccinated,
    new_vaccinations,
    new_vaccinations_smoothed,
    total_vaccinations_per_hundred,
    people_vaccinated_per_hundred,
    people_fully_vaccinated_per_hundred,
    new_vaccinations_smoothed_per_million FROM
    covid19data
    WHERE
        (total_vaccinations ,
        people_vaccinated,
        people_fully_vaccinated,
        new_vaccinations,
        new_vaccinations_smoothed,
        total_vaccinations_per_hundred,
        people_vaccinated_per_hundred,
        people_fully_vaccinated_per_hundred,
        new_vaccinations_smoothed_per_million) <> ('' , '', '', '', '', '', '', '',
'')
    GROUP BY
        location ,
        date);

-- CREATE covid_measures TABLE
```

```sql
CREATE TABLE covid_measures AS (SELECT location, date, stringency_index FROM
    covid19data
WHERE
    stringency_index <> ''
GROUP BY location , date);
```

Appendix B: MySQL Script for Queries 1-7

```
-----------------
-- 1) What is the total population in Asia?
-----------------

SELECT
    SUM(population) AS total_population
FROM
    location_indicators
        LEFT JOIN
    location ON location_indicators.location = location.location
WHERE
    location.continent = 'Asia';


-----------------
-- 2) What is the total population among the ten ASEAN countries?
-----------------

 SELECT
        SUM(population)
    FROM
        location_indicators
    WHERE
        location IN ('Brunei', 'Myanmar' , 'Cambodia', 'Indonesia', 'Laos',
'Malaysia', 'Philippines', 'Singapore', 'Thailand' , 'Vietnam');


-----------------
-- 3) Generate a list of unique data sources (source_name)
-----------------

SELECT
    DISTINCT `source_name`
FROM
    `country_vaccinations`;


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

select
      `date`,`total_vaccinations`
from
```

```
      (select * from `covid_vaccinations` where `location` = "Singapore") as x
where
      `date` between '2021-03-01' and '2021-05-31';


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

SELECT
    MIN(date)
FROM
    covid_vaccinations
WHERE
    location = 'Singapore'
        AND total_vaccinations <> '';


-----------------
-- 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.
-----------------

SELECT
    SUM(new_cases)
FROM
    covid_cases
WHERE
    location = 'Singapore'
    AND date >= (SELECT MIN(date) FROM covid_vaccinations
                WHERE
                    location = 'Singapore'
                    AND total_vaccinations IS NOT NULL
                    AND total_vaccinations <> '');


-----------------
-- 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.
-----------------

SELECT
    SUM(`new_cases`) AS `total_cases_before_vaccine`
FROM
    `covid_cases`
WHERE
    `location` = "Singapore"
    AND `date` < (SELECT MIN(`date`)
                            FROM `covid_vaccinations`
                  WHERE `location` = 'Singapore'
                            AND `total_vaccinations` > 0);
```

Appendix C: NoSQL Script for Queries 1-17

```
// q1. Display a list of total vaccinations per day in Singapore.
// [source table: country_vaccinations]

db.country_vaccinations.aggregate([{
    $match: {
        country: "Singapore"
    }
}, {
    $project: {
        date: 1,
        daily_vaccinations: 1
    }
}])

// q2. Display the sum of daily vaccinations among ASEAN countries.
// [source table: country_vaccinations]
db.country_vaccinations.aggregate(
   [
        {$match :
               {
                          country : {$in : ["Brunei", "Burma", "Cambodia", "Timor",
"Indonesia", "Laos", "Malaysia", "Philippines", "Singapore", "Thailand", "Vietnam",
"Myanmar"]}
               }
        },
        {$group:
             {
               _id: { "country" : "$country" },
               "sum": {$sum: {$convert:{input: "$daily_vaccinations", to: "int"}}}
             }
        }
   ]
);

// q3. Identify the maximum daily vaccinations per million of each country.
// Sort the list based on daily vaccinations per million in a descending order.
// [source table: country_vaccinations]

db.country_vaccinations.aggregate([
     { $group: { _id: "$country", "max_daily_vaccinations_per_million": { $max: {
$convert: { input: "$daily_vaccinations_per_million", to: "int" } } } } },
    { $sort: { max_daily_vaccinations_per_million: -1 } }
])


//  q4.  Which  is  the  most  administrated  vaccine?  Display  a  list  of  total
administration
// (i.e., sum of total vaccinations) per vaccine.
// [source table: country_vaccinations_by_manufacturer]

db.country_vaccinations_by_manufacturer.aggregate([
          {$group:  {_id:{groupByVaccine:  "$vaccine"},  total_administrated:{$max:
{$convert:{input: "$total_vaccinations", to: "double"}}}}},
    {$project: {_id:1 ,total_administrated:1}},
    {$sort: {total_administrated : -1}}
])


// q5: Italy has commenced administrating various vaccines to its populations as a
vaccine becomes available.
// Identify the first dates of each vaccine being administrated,
// then compute the difference in days between the earliest date and the 4th date.
// [source table: country_vaccinations_by_manufacturer]

db.country_vaccinations_by_manufacturer.aggregate([{
    $match: {
        location: "Italy",
        total_vaccinations: {
            $ne: 0
        }
    }
}, {
    $group: {
        _id: {
            vaccine: "$vaccine"
        },
        first_date: {
            $min: "$date"
        }
    }
}, {
    $group: {
        _id: null,
        min_date: {
            $min: "$first_date"
        },
        max_date: {
            $max: "$first_date"
        }
    }
}, {
    $project: {
        daysDifference: {
            $dateDiff: {
                startDate: {
                    $dateFromString: {
                        dateString: "$min_date"
                    }
                },
                endDate: {
                    $dateFromString: {
                        dateString: "$max_date"
                    }
                },
                unit: "day"
            }
        }
    }
}])


// q6. What is the country with the most types of administrated vaccine?
// [source table: country_vaccinations_by_manufacturer]
db.country_vaccinations_by_manufacturer.aggregate(
        [
            {$group:
                {
                _id: {"country":"$location"},
                "Vaccines": {$addToSet:"$vaccine"},
                }
            },
            {$project:
                    {"_id":1,"Vaccines":1,count : {$size:"$Vaccines"}}
            },
            {$sort:{count:-1}}
        ]
    );
```

```
// q7. What are the countries that have fully vaccinated more than 60% of its people?
// For each country, display the vaccines administrated.
// [source table: country_vaccinations]

db.country_vaccinations.aggregate([
    { $project: { country: 1, vaccines: 1, people_vaccinated_per_hundred: { $convert:
{ input: "$people_vaccinated_per_hundred", to: "double" } } } },
    { $match: { people_vaccinated_per_hundred: { $gt: 60 } } },
       {  $group:  {_id:  [  {  country:  "$country"  },  {  vaccines:  "$vaccines"  }  ],
max_people_vaccinated_per_hundred: { $max:"$people_vaccinated_per_hundred" } } },
    { $project: { _id:0, country: { $arrayElemAt: ["$_id.country", 0] }, vaccines: {
$arrayElemAt: ["$_id.vaccines",0] }, max_people_vaccinated_per_hundred: 1 } }
])


// q8. Monthly vaccination insight – display the monthly total vaccination amount of
each
// vaccine per month in the United States.
// [source table: country_vaccinations_by_manufacturer

db.country_vaccinations_by_manufacturer.aggregate([
    {$match: {location: "United States"}},
     {$group:  {_id:[{location :"$location"},{ vaccine :"$vaccine"},{month: {$month:
{$convert: {input:"$date", to: "date"}}}}] ,
     monthly_total_vaccination:  {$max:  {$convert:{input:  "$total_vaccinations",  to:
"double"}}}}},
    {$sort: {"_id.2":1, "_id.1":1}}
])


//  q9:  Days  to  50  percent.  Compute  the  number  of  days  (i.e.,  using  the  first
available date on records of a country)
//   that   each   country   takes   to   go   above   the   50%   threshold   of   vaccination
administration (i.e., total_vaccinations_per_hundred > 50)
// [source table: country_vaccinations]

db.country_vaccinations.aggregate([{
    $group : {
        _id: {
            country: "$country"
        },
        first_date: {
            $min: {
                $convert: {
                    input: "$date",
                    to: "date"
                }
            }
        }
    }
}, {
    $project: {
        country: "$_id.country",
        first_date: "$first_date"
    }
}, {
    $lookup: {
       from: "country_vaccinations",
       localField: "country",
       foreignField: "country",
       pipeline: [
            {
               $match: {
                    $expr: {
                        $gt: [{$toDouble: "$total_vaccinations_per_hundred"}, 50]
                    }
                }
            },
            {
                $sort: {
                    date: 1
                }
            },
       ],
       as: "country_vaccinations"
    }
}, {
    $project: {
        country: "_id.country",
        days_to_50th_percent_of_population_vaccinated: {
            $dateDiff: {
                startDate: "$first_date",
                endDate: {
                    $convert: {
                        input: {
                            $arrayElemAt: ["$country_vaccinations.date", 0]
                        },
                        to: "date"
                    }
                },
                unit: "day"
            }
        }
    }
}])
```

```
// q10. Compute the global total of vaccinations per vaccine.
// [source table: country_vaccinations_by_manufacturer]
db.country_vaccinations_by_manufacturer.aggregate(
    [
        {$group:
        { _id:{"Vaccine":"$vaccine"},
            count:{$max: {$convert:{input: "$total_vaccinations", to: "double"}}}
        }
        }
        ]
    )

// q11. What is the total population in Asia?

db.covid19data.aggregate([
      {  $project:  {  continent:  1,  location:  1,  population:  {  $convert:  {  input:
"$population", to:"double" } } } },
    { $match: { continent: { $eq: "Asia" } } },
    { $group: { _id: "$location", max_pop: { $max: "$population" } } },
    { $group: { _id: "asiaPopulation", asiaPopulation: { $sum: "$max_pop" } } },
    { $project: {_id: 0, asiaPopulation: 1 } }
])


// q12. What is the total population among the ten ASEAN countries?

db.covid19data.aggregate([
    {$match: {location: {$in: ['Brunei', 'Myanmar' , 'Cambodia', 'Indonesia', 'Laos',
'Malaysia', 'Philippines', 'Singapore', 'Thailand' , 'Vietnam']}}},
      {$group:  {_id:  {groupByLocation:  "$location"},  population:  {$avg:  {$convert:
{input: "$population", to: "double"}}}}},
    {$group: {_id: '', total_population: {$sum: "$population"}}},
    {$project: {_id:1, total_population: 1}}
])


// q13: Generate a list of unique data sources (source_name)

db.country_vaccinations.aggregate([
    {
        $group: {
            _id: {
                source_name: "$source_name",
            }
        }
    },
])
```

```
//  q14:  Specific  to  Singapore,  display  the  daily  total_vaccinations  starting
(inclusive) March-1 2021 through (inclusive) May-31 2021
db.country_vaccinations.aggregate(
    [
                 {$project:{"country":1,"total_vaccinations":1,date:{$convert:{input:
"$date",to:"date"}}}},
        {$match:
        {$and:[{"country":"Singapore"},
                {"date":{$gte:ISODate("2021-03-01T08:00:00.000+08:00")   }},
                {"date":{$lte:ISODate("2021-05-31T08:00:00.000+08:00")}}]}

        }
        ]
    )

// q15. When is the first batch of vaccinations recorded in Singapore?

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


// q16. Based on the date identified in (5), specific to Singapore,
// compute the total number of new cases thereafter.
// For instance, if the date identified in (5) is Jan-1 2021, the total number of new
cases will
// be the sum of new cases starting from (inclusive) Jan-1 to the last date in the
dataset.

db.covid19data.aggregate([
    {$match: {location: "Singapore"}},
        {$project:  {location:1,  date:  {$convert:  {input:  "$date",  to:  "date"}},
new_cases: {$convert: {input: "$new_cases", to: "double"}}}},
    {
        $lookup: {
            from: "country_vaccinations",
            let: {
                covid_country: "$location",
                covid_date: "$date"
            },
            pipeline: [
                {$match: {country: "Singapore"}},
                  {$group: {_id:null, min_date: {$min: {$convert: {input:"$date", to:
"date"}}}}},
                {
                    $project: {
                    country:1, min_date:1
                    }
                },
                {
                    $match:{
                        $expr: {$lte: ["$min_date", "$$covid_date"]}}
                },
        ],
        as: "after_vaccination"
        }
    },
    {$match: {$expr: {$ne:["$after_vaccination",[]]}}},
    {$group: {_id: '', sum_new_cases: {$sum: "$new_cases"}}},
    {$project: {_id:0, sum_new_cases:1}}
])


// q17: 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

db.covid19data.aggregate([{
    $match: {
        location: "Singapore"
    }
}, {
    $lookup: {
        from: "country_vaccinations",
        pipeline: [
            { $match: { country: "Singapore" } },
            { $group: { _id: "first_date", first_date: { $min: "$date" } } },
            { $project: { _id: 0, first_date: 1 } }
        ],
        as: "sgFirstVaccineBatch"
    }
}, {
    $unwind: "$sgFirstVaccineBatch"
}, {
    $match: {
        $expr: {
            $lt: [
                {
                    $dateFromString: {
                        dateString: "$date"
                    }
                },
                {
                    $convert: {
                        input: "$sgFirstVaccineBatch.first_date",
                        to: "date"
                    }
                }
            ]
        }
    }
}, {
    $group: {
        _id: "$location",
        totalNewCases: {
            $sum: {
                $convert: {
                    input: "$new_cases",
                    to: "double"
                }
            }
        }
    }
}])
```