# Report

Sandor Kovacs
sk10g20
31773478

May 2021

## 1 Introduction

The aim of this report is to create a SQlite database to represent current Coronavirus data. The report involves creating and normalising a database to hold the data from a dataset and constructing queries against the data once it is in a suitable form.

## 2 The Relational Model

### 2.1 EX1

| Dataset | |
|---|---|
| dateRep | NUMERIC |
| day | INTEGER |
| month | INTEGER |
| year | INTEGER |
| cases | INTEGER |
| deaths | INTEGER |
| countriesAndTerritories | TEXT |
| geoId | TEXT |
| countryterritoryCode | TEXT |
| popData2019 | INTEGER |
| continentExp | TEXT |

### 2.2 EX2

dateRep → day
dateRep → month
dateRep → year
countriesAndTerritories → geoId
countriesAndTerritories → countryterritoryCode
countriesAndTerritories → popData2019

countriesAndTerritories → continentExp
day, month, year → dateRep
dateRep, countriesAndTerritories → cases
geoId → countriesAndTerritories
dateRep, countriesAndTerritories → deaths

There are null values in the dataset for data both in countryterritoryCode as well as popData2019 for Wallis-and-Futuna and Cases-on-an-international-conveyance-Japan. This is why they can not be part of primary key as they can not determine anything else due to their null values.

### 2.3 EX3

dateRep, countriesAndTerritories → Predicts Everything
day, month, year, countriesAndTerritories → Predicts Everything
dateRep, geoId → Predicts Everything
day, month, year, geoId → Predicts Everything

### 2.4 EX4

There are several primary keys that could be used. The primary key that I will be using is dateRep, countriesAndTerritories, as this is the easiest and most convenient one to use.

## 3 Normalisation

### 3.1 EX5

Partial Key Dependencies:
day, month, year → dateRep
dateRep → day, month, year
countriesAndTerritories → geoId, countryterritoryCode, continentExp, popData2019
geoId → countryterritoryCode, continentExp, popData2019, countriesAndTerritories

### 3.2 EX6

| DateTable | |
|---|---|
| dateRep(key) | NUMERIC |
| day | INTEGER |
| month | INTEGER |
| year | INTEGER |

| CountryDataTable | |
|---|---|
| countriesAndTerritories(key) | TEXT |
| geoId | TEXT |
| countryterritoryCode | TEXT |
| popData2019 | INTEGER |
| continentExp | TEXT |

| CaseAndDeathTable | |
|---|---|
| dateRep(key) | NUMERIC |
| cases | INTEGER |
| deaths | INTEGER |
| countriesAndTerritories(key) | TEXT |

In order to be in second normal form no partial key dependencies are allowed so I searched all partial keys, and then decomposed the original table so that no partial keys could be found in any of the new tables. Than we have our tables in second normal form.

### 3.3 EX7

I could not find any transitive dependencies in my new tables.

### 3.4 EX8

To be in third normal form no transitive dependencies are allowed. In each table, the attributes are only dependent on the key. There is one exception "CountryDataTable", where the attributes are dependent not only on "countriesAndTerritories" but also "geoId". However, while "geoId" is dependent on "countriesAndTerritories", the same is true the other way around which means this is not a transitive dependency.

### 3.5 EX9

"DateTable" and "CaseAndDeathTable" are in Boyce-Codd Normal Form. "CountryDataTable" is not in Boyce-Codd Normal Form, as "geoId" can determine "countriesAndTerritories" even though it is a non prime attribute.

## 4 Modelling

### 4.1 EX10

Firstly, I downloaded the "dataset.csv" file, than I ran the following lines in the terminal:

The first line is to create a new database named "coronavirus.db". The second line is to change the default mode to csv. The third line imports the

```
sqlite3 coronavirus.db
.mode csv
.import dataset.csv dataset
.once dataset.sql
```

dataset into a table named "dataset". Finally, the last line exports the data into a file named "dataset.sql".

## 4.2 EX11

In this exercise, I created five tables according to the specifications:

```
CREATE TABLE NameOfTable
(
attribute1 TYPE,
attribute2 TYPE,
attribute3 TYPE
);
```

Than I executed the following commands in the terminal to store the new tables in the "dataset2.sql" file:

```
sqlite3 coronavirus.db ¡ ex11.sql
sqlite3 coronavirus.db
.output dataset2.sql
.dump datesTable, deathCasesTable, geoIdTable, countriesTable, countries-
DataTable
```

The first line is to execute ex11.sql against the coronavirus.db database than we go into the database and output the results to the dataset2.sql than I dumped the tables into it.

## 4.3 EX12

In this exercise I inserted the appropriate data into each of the normalised tables.

Afterwards I executed the following commands in the terminal to store the new tables in the "dataset3.sql" file:

The first line is to execute ex12.sql against the coronavirus.db database than we go into the database and output the results to the dataset3.sql than I dumped the tables into it.

## 4.4 EX13

To test my database and SQL code, I created a new database and tested my code on it, and I got the expected output.

```
INSERT INTO NameOfTable(firstAttribute, secondAttribute, thirdAttribute)
SELECT firstAttribute, secondAttribute, thirdAttribute
FROM dataset GROUP BY firstAttribute;
```

```
sqlite3 coronavirus.db ¡ ex12.sql
sqlite3 coronavirus.db
.output dataset3.sql
.dump datesTable, deathCasesTable, geoIdTable, countriesTable, countries-
DataTable
```

# 5 Querying

## 5.1 EX14

In this exercise I just selected the sum of the cases and deaths columns:

```
SELECT SUM(cases), SUM(deaths)
FROM CaseAndDeathTable;
```

## 5.2 EX15

In this exercise, I selected the "cases" and "dateRep" columns. Then as I only wanted to check data in the UK I only permitted the query to sum where countriesAndTerritories='United-Kingdom' . Finally, I used a command called "strftime", which allows sorting a column, based on a date and than sorted it in ascending order with the ASC command.

```
SELECT dateRep, cases
FROM dataset
WHERE countriesAndTerritories='United-Kingdom'
ORDER BY strftime("%d/%m/%Y",dateRep) ASC;
```

## 5.3 EX16

In this exercise, I selected the "dateRep", "continentExp" columns as well as the sum of "cases" and "deaths". Firstly I grouped the output by continent and date to allow me to order it based on the same criteria. I learned that GROUP BY is usually followed by ORDER BY to get the expected output. The substr command allowed me to take only a part of a string as dateRep was the year month and day and I needed them separately so I separated them.

```
SELECT continentExp, dateRep, sum(cases), sum(deaths)
FROM dataset
GROUP BY continentExp, substr(dateRep,7,4)||substr(dateRep,4,2)||substr(dateRep,1,2);
ORDER BY continentExp, substr(dateRep,7,4)||substr(dateRep,4,2)||substr(dateRep,1,2);
```

## 5.4 EX17

In this exercise, I selected the "countriesAndTerritories" and calculated the number of cases and deaths as a percentage of the population. The printf command returns a formatted string with the calculated values. I calculated the percentages with a 2 decimal accuracy(using the .2f command) as this is the way it is mostly used. Than I grouped it bu countriesAndTerritories and ordered it by the same criteria.

```
SELECT countriesAndTerritories, printf("%.2f",100.0*(sum(cases)*1.0/popData2019*1.0)),
printf("%.2f", 100.0*(sum(deaths)*1.0/popData2019*1.0))
FROM dataset
GROUP BY countriesAndTerritories
ORDER BY countriesAndTerritories
```

## 5.5 EX18

In this exercise, I selected the "countriesAndTerritories" and calculated the percentage of deaths per cases selecting it from the dataset. I worked with a 2 decimal accuracy, and in order for the output to be just the first 10 I limited the output after I grouped it and ordered it.

```
SELECT countriesAndTerritories, printf("%.2f",100.0*(sum(deaths)*1.0/sum(cases)*1.0))
FROM dataset
GROUP BY countriesAndTerritories
ORDER BY 2 DESC
LIMIT 10
```

## 5.6 EX19

In this exercise I selected the dateRep and the sum of cases and deaths but I used UNBOUNDED PRECEDING so that the frame would start at the first data of the partition and ends at the current row allowing me to calculate the cumulative deaths and cases easily. I selected this from the dataset but before selecting it I did a modification where I firstly ordered the data by the date and put it in ascending order. Then as I only needed data in the United Kingdom I only allowed the query to select data where countriesAndTerritories='United-Kingdom'. Finally, I ordered them by date using the substr command again.

```
SELECT dateRep, SUM(cases) OVER (ROWS UNBOUNDED PRECEDING),
SUM(deaths) OVER (ROWS UNBOUNDED PRECEDING)
FROM (SELECT * FROM dataset ORDER BY
substr(dateRep,7,4)||substr(dateRep,4,2)||substr(dateRep,1,2) ASC)
WHERE countriesAndTerritories='United-Kingdom'
ORDER BY substr(dateRep,7,4)||substr(dateRep,4,2)||substr(dateRep,1,2);;
```