Report

Patrik-Tibor Csanyi/ptc1g20/31574602

May 2021

1 Introduction

In this report we attempt to take a csv file, introduce the data from it into a database, normalise the previously mentioned database, and then perform queries on it.

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

Functional dependencies:
dateRep → day
dateRep → month
dateRep → year
countriesAndTerritories → geoId
countriesAndTerritories → countryterritoryCode
countriesAndTerritories → popData2019
countriesAndTerritories → continentExp
geoId → countriesAndTerritories

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

Two "countriesAndTerritories" have null for data both in countryterritoryCode as well as popData2019: Wallis-and-Futuna and Cases-on-an-international-conveyance-Japan. This is the only reason they do not predict other data

The combination on deaths and cases also can not predict any data.

2.3 EX3

Candidate keys:
dateRep, countriesAndTerritories → Predicts All
dateRep, geoId → Predicts All
day, month, year, countriesAndTerritories → Predicts All
day, month, year, geoId → Predicts All

2.4 EX4

The primary key I will be using is (dateRep, countriesAndTerritories), as this is the easiest to use

3 Normalisation

3.1 EX5

Partial Key Dependencies:
dateRep → day, month, year
day, month, year → dateRep
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

I found all partial keys, and then broke down the original table so that no partial keys remained in any of the new tables

3.3 EX7

There are no transitive dependencies in my new tables

3.4 EX8

In each table, the attributes are solely dependent on the key/keys.

The only exception is the "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.

3.5 EX9

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

4 Modelling

4.1 EX10

After downloading the "dataset.csv" file, I ran the following lines in the terminal:

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

The first line creates the new database named "coronavirus.db". The second line tells SQLite that the dataset will be in a csv file format. The third line imports the 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 three tables according to the specifications in EX6. The format I followed in the "ex11.sql" file for each table was:

```
CREATE TABLE TableName
(
attribute1 TYPE,
attribute2 TYPE,
attribute3 TYPE,
attribute4 TYPE,
);
```

Afterwards 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 DateTable CountryDataTable CaseAndDeathTable
```

4.3 EX12

In this exercise I inserted the appropriate data into each of the normalised tables. To insert the data, I followed the format below for each table:

```
INSERT INTO TableName(attribute-1, attribute-2, attribute-3, attribute-4)
SELECT attribute-1, attribute-2, attribute-3, attribute-4
FROM dataset GROUP BY attribute-1;
```

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

```
sqlite3 coronavirus.db < ex12.sql
sqlite3 coronavirus.db
.output dataset3.sql
.dump DateTable CountryDataTable CaseAndDeathTable
```

4.4 EX13

To test my database and SQL code, I deleted every file except for "ex11.sql" "ex12.sql" and "dataset.csv". Then, I created a new database, and ran my code again on it, to get the appropriate output.

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 "dateRep" and "cases" columns. Then to make sure I only get UK data, I made sure to get countriesAndTerritories='United-Kingdom' with the where command. Finally, I used a command called "strftime", which allows sorting a column, based on a date.

```
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". First I grouped the output by continent and date, after which I order them based on the same criteria.

```
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" as well as calculating the number of cases and deaths as a percentage of the population, by working with the "cases" and "deaths" columns in the way described below. The printf command merely returns a formatted string with the calculated values. I chose to calculate the percentages with a 2 decimal accuracy.

```
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" as well as calculating the percentage of deaths per cases, by working with the "cases" and "deaths" columns in the way described below. I once again worked with a 2 decimal accuracy, and limited the output to the first 10 countries, ordered by death per case percentage.

```
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 "countriesAndTerritories" as well as calculating the cumulative deaths and cases, by working with the "cases" and "deaths" columns in the way described below. To do this I needed to order the data by date, before I did any operations, so that the cumulative sum would come out correctly. After all was calculated, I just needed to select the data regarding the UK, and order it once more.

```
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);;
```

6 Extension

6.1 EX20