1. The Relational Model

Please DO NOT USE LETTERS in your report to represent columns or attributes, please make sure to use their full names as given in the column headings in the dataset.csv. Failure to name them correctly may result in a significant loss of marks!

For this first part, you should answer the questions by looking directly at the data in the dataset and answering them in your LaTeX report - you do not need to use SQLite or Datagrip or any tools yet!

EX1: Express the relation directly represented in the dataset file. Assign relevant SQLite data types to each column.

- We would suggest representing this in written Relation form and then using a LaTeX table to represent the columns and types. Do not use an image.
- Look at the data that is in the dataset and express it as a Relation, as seen earlier in the course.
- Types: To ease the modelling process, we will be using SQLite data types (INTEGER, TEXT, BLOB, REAL)
- Note: You must use the attribute names exactly as they appear in the dataset

EX2: List the **minimal set** of Functional Dependencies (FDs)

- Every FD must have **only one** attribute on the RHS (right hand side)
  - Where A -> B,C,D this is represented as A -> B, A -> C, A -> D
- Every FD must be **minimal** on it' s LHS (left hand side)
  - There can be more than one attribute on the LHS, but there should be no attributes on the LHS that add no further information (e.g. If A -> C, then A,B -> C would not be minimal as the B is not adding anything further)
- There should be no trivial FDs (a -> a adds nothing of value)
- There must be no redundant FDs (those which are already implied by other FDs)
  - Tip: If A -> B and C -> B and C -> A then C -> B is redundant (as this is implied by C -> A -> B) and thus does not feature in the minimal set.
- Tip: Explain any assumptions you make applying what you know of the domain to the data and consider future data and the impact it may have as well.
- Tip: You will need to think and determine whether values are 'blank' (a known value of blank) or null (an as yet unknown value) as this may have an impact on your dependencies. Explain any assumptions and decisions you make in the report.

EX3: From your minimal set of functional dependencies, list the potential candidate keys

- Tip: Remember, a candidate key must contain no redundant attributes

EX4: Identify a suitable primary key, and justify your decision

- Tip: You may find it useful to use your scripting and processing skills to test whether the data and your FDs/keys hold

2. Normalisation

For this part, you should be able to answer all these questions by looking at the dataset directly and writing the answers in the report. You do not need to use SQlite, Datagrip or any tools yet!

- Keys: Where possible, you should only introduce new Surrogate Keys where they are necessary. If and where they are necessary, to avoid anomalies, you should explain this in your report with a justification.
- Attributes: While you are able to introduce new attributes if you wish, you must not rename or remove or change the values of any of the attributes in the original relation (such as the dateRep (date reported), even though it is otherwise broken up). All must appear as originally named in your broken down relations.
- NULL values: NULL values are not values in themselves, but represent unknown values in the dataset (you cannot treat all NULL values as the same 'null' value). NULL values can be present throughout the normalisation process, you do not need to remove them. However, you may find you need to introduce surrogate keys in the case where a NULL could or is present in something you would want to be a key or split into a relation, for example.

EX5: List any partial-key dependencies in the relation as it stands and any resulting additional relations you should create as part of the decomposition.

EX6: Use decomposition and your answer to the above to achieve 2nd Normal Form, introducing appropriate new relations. List the new relations and their fields, types and keys. Explain the process you took.

- Note: When decomposing, you may have options as to what attributes to use - you should use the appropriate attributes identified in your primary key in EX4.

EX7: List transitive dependencies, if any, in your new relations

EX8: Convert your relations into 3rd Normal Form using your answers to the above. List the new relations and their fields, types and keys. Explain the process you took.

- Note: Depending on assumptions you have made and earlier processes, you may already be in 3NF - in which case, demonstrate why your table is in 3NF.

EX9: Finally, convert your relations into Boyce-Codd Normal Form. Justify and explain how your relations are in BCNF.

- - Note: Originally, this was just explaining whether the dataset was in BCNF or not and why - if you have explained this rather than converting into BCNF, that will be acceptable too. However, to make things easier, we would encourage forming your relation into BCNF where possible if it is not already in BCNF.

3. Modelling

In this part, we are going to start modelling our database physically, so you will want to make sure you have SQLite set up, and a tool such as Datagrip to make it easier to work with.

Where possible, you should only introduce new Surrogate Keys where they are necessary. If and where they are necessary, to avoid anomalies, you should explain this in your report with a justification.

- Attributes: While you may add new attributes if you wish, all original attribute names and their values must remain as they were in the original dataset, you **may not rename them or change their values** - this ensures consistency with the original dataset and easy importing of new data.

Each SQL statement should be written in the report, as well as saved as ex<number>.sql (e.g. ex11.sql for the first). You should explain the steps for **EX10** and provide the steps and SQL for EX11-13. Briefly write down the process you went through to go with it - enough that a person with just your report could reproduce what you have done.

EX10: Using the CSV import function (.import), import the raw dataset into *SQLite* into a single table called 'dataset' in an SQLite database called *coronavirus.db*. Dump (using .dump) this table as dataset.sql (which will include the SQL CREATE and INSERT statements), such that running it will import the full dataset into a fresh SQLite database

- Importing CSV: You can import a CSV in SQlite using .mode csv then .import *file table*
  - Note: You may not change the CSV file - it must be the original provided dataset file. The attribute names must be as in the original file.
- Dumping to SQL:
  - When you dump a table (Using the .dump command in the sqlite3 CLI), it will output the SQL needed to create and populate the database (CREATE TABLE and INSERT statements)
  - To dump the entire database to a file (e.g. dataset.sql), use *.output dataset.sql* then *.dump*
- The entire database at this point should be dumped as *dataset.sql*
- You should be able to run *sqlite3 coronavirus.db < dataset.sql* on an empty coronavirus.db to create and then populate the dataset table.

EX11: Write the SQL to create the full normalised representation, including all additional tables (with correct types) with no data and excluding the dataset table. Write the set of SQL statements to *ex11.sql*, run them on your database and as above, dump the full database at this point to *dataset2.sql*

You should not modify/create/include the dataset table. The SQL should contain CREATE statements to create any new tables. You should include indexes and foreign keys where appropriate, and list and justify these in your answer.

If you have introduced any surrogate keys, please list and justify them as part of this answer.

- The SQL statements to create the tables should be saved as *ex11.sql*
- The entire database at this point should be dumped as *dataset2.sql*

EX12: Write INSERT statements using SELECT to populate the new tables **from** the 'dataset' table. Save these to *ex12.sql*, run them on your database and as above, dump the database as it stands now to *dataset3.sql*

- The SQL statements to populate the tables from the dataset table should be saved as *ex12.sql*
- The entire database at this point should be dumped as *dataset3.sql*
- Tip: Consider using INSERT OR IGNORE INTO (which won't error on duplicates on your keys) or SELECT DISTINCT to avoid selecting duplicate rows when you pull from the dataset
- Warning: If your INSERT statements are containing data values directly, you're doing it wrong! (You should be populating FROM the dataset table)

EX13: Test and ensure that on a clean SQLite database, you can execute dataset.sql followed by ex11.sql followed by ex12.sql to successfully populate your database.

You **must** be able to run the following commands and end up with a fully populated database.

```
sqlite3 coronavirus.db < dataset.sql
sqlite3 coronavirus.db < ex11.sql
sqlite3 coronavirus.db < ex12.sql
```

In the report, you just need to confirm you have run these exact commands and they have worked.

4. Querying

For each exercise in this question, you will need to write an SQL query (against your newly created normalised tables in your database - you *must not create further tables* specifically for these queries!).

Each SQL statement should be written in the report, as well as saved as *ex<number>.sql* (e.g. ex14.sql for the first) which can be run against your database, as it stands at the end of **EX12**.

As well as including the SQL, you should also briefly describe your approach for each in the report.

- IMPORTANT: When run, the query **MUST return the results** (such that sqlite3 coronavirus.db < ex14.sql will return the results). It should *NOT* return headings as part of the data!
- Ordering: Remember, you can order by things not in the SELECT part of the query (e.g. ORDER BY year,month,day)

Write an SQL statement for each of the following:

EX14: The worldwide total number of cases and deaths (with total cases and total deaths as columns)

EX15: The number of cases by date, in increasing date order, for the United Kingdom (with the date reported and number of cases as columns)

EX16: The number of cases and deaths by date, in increasing date order, for each country (with country, date, number of cases and number of deaths as columns)

- Tip: You may choose whether to organise by date first or by country first
- Note: This was originally by continent, but the dataset no longer has a continent - if you have done so by continent, this will also be accepted

EX17: The total number of cases and deaths as a percentage (to 2DP) of the population, for each country (with country name, % cases of population, % deaths of population as columns)

- Tip: A percentage is out of 100, not out of 1! (e.g. with a population of 50,000 and a total number of deaths at 3700, it'd be ((3700/50000)*100) = 7.4%. This should be to 2 DP.

EX18: A descending list of the the top 10 countries, by percentage (to 2DP) total deaths out of total cases in that country (with country name and % deaths of country cases as columns)

EX19: The date against a cumulative running total of the number of deaths by day and cases by day for the united kingdom (with date, cumulative UK deaths and cumulative UK cases as columns)

- Tip: You will want to use the Window Functions in SQLite to achieve this. Please ensure your version of SQLite supports this.