**Project name - Converting asset results from Json to a sqlite3 database and enabling querying through a user-friendly browser interface**

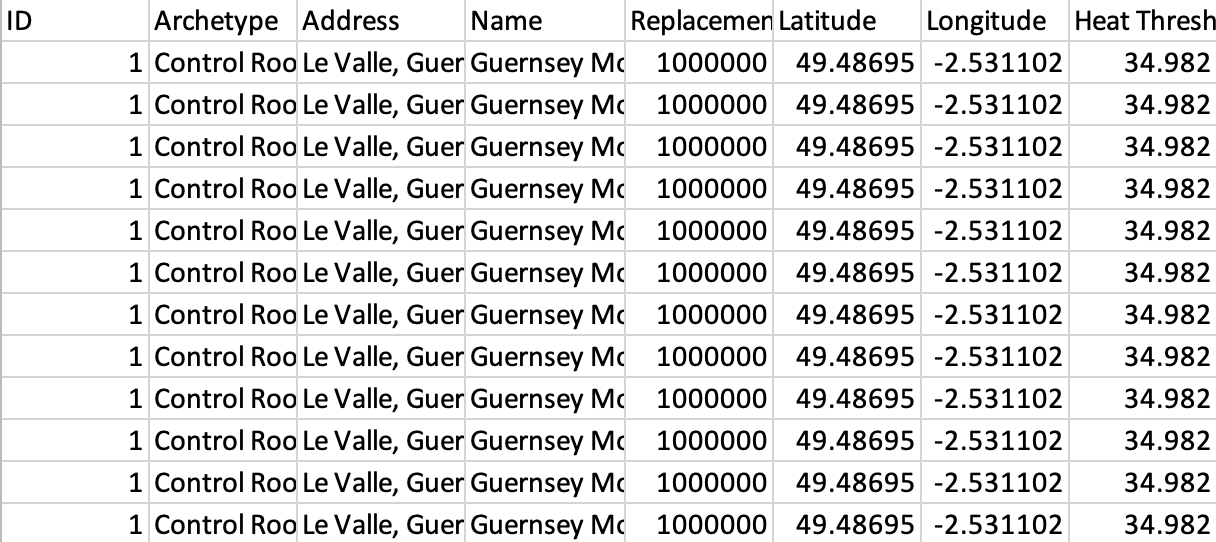
1 asset

1 or 2 rcps

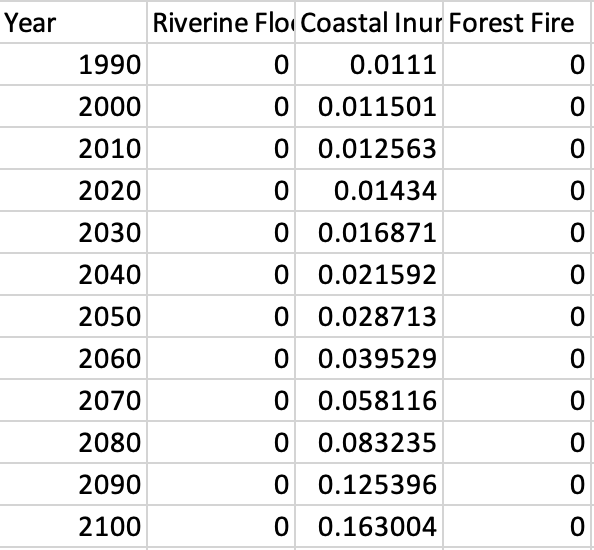
The two DBs are as follows:

Step 1 – suck the Json data into a SQLITE database, called **MyDataBase.db**

1. A table for non changing fields, as shown below



2. Another table for changing fields



**Prep steps**

Step 0.1 (for the front end) – set up a server through flask to implement the python + SQL code written in the back end

~~DONE Step 0.2 – set up SQLite 3 on my machine~~

~~Step 0.3 – through Python code, pull out the relevant data fields from the Json~~

For year in range(1990, 2101)

row = {}

row["Year"] = year

row["flood\_riverine\_Failure\_Probability"] = Haz\_FP\_by\_Year[year]["flood\_riverine"]

row[“flood\_surfacewater\_Failure\_Probability"] = Haz\_FP\_by\_Year[year]["flood\_surfacewater"]

row[“soil\_movement\_Failure\_Probability"] = Haz\_FP\_by\_Year[year]["soil\_movement"]

row[“wind\_Failure\_Probability"] = Haz\_FP\_by\_Year[year]["wind"]

row[“heat\_Failure\_Probabillity"] = Haz\_FP\_by\_Year[year]["heat"]

row[“forest\_fire\_Failure\_Probability"] = Haz\_FP\_by\_Year[year]["forest\_fire"]

row[“inundation\_Failure\_Probability"] = Haz\_FP\_by\_Year[year]["inundation"]

row[“Asset\_Failure\_probability”] = Asset\_FP\_by\_Year[year]

row["flood\_riverine\_VAR"] = Hazard\_VAR\_by\_Year[year]["flood\_riverine"]

row[“flood\_surfacewater\_VAR"] = Hazard\_VAR\_by\_Year [year]["flood\_surfacewater"]

row[“soil\_movement\_VAR"] = Hazard\_VAR\_by\_Year [year]["soil\_movement"]

row[“wind\_VAR"] = Hazard\_VAR\_by\_Year [year]["wind"]

row[“heat\_VAR"] = Hazard\_VAR\_by\_Year [year] ["heat"]

row[“forest\_fire\_VAR"] = Hazard\_VAR\_by\_Year [year]["forest\_fire"]

row[“inundation\_VAR"] = Hazard\_VAR\_by\_Year [year]["inundation"]

row[“Asset\_VAR”] = Asset\_VAR\_by\_Year[year]

row["TIP"] = TIP\_by\_Year [year]

**db.execute("INSERT INTO full\_table VALUES (:year, :**Extreme\_Heat\_FP**, :** Extreme\_Wind\_FP**, :** Forest\_Fire\_FP**)", row)**

~~Step 2 – create the html template for the user interface~~

~~Step 3 - create SQLITE queries to facilitate the user querying that DB, as follows:~~

 - 1. what VAR did the asset have in the year 2020/2050 (answer: 2%)

SELECT Asset\_VAR

FROM full\_table

WHERE Year = "2050";

 - 2. which hazard has the highest FP in the year 2020/2050 (answer: riverine flooding) – done through a function with string formatting in it, taking VAR/Failure Probability and user selected year as the inputs

 - 3. at which year was the max asset VAR reached? (answer: 1900)

SELECT Year

FROM full\_table

WHERE Asset\_VAR = (

SELECT

MAX(Asset\_VAR)

FROM

full\_table);

Step 4 – set up Flask to facilitate the web application working through a browser

To get into the virtual environment from which to run Flask, from the project directory type the following - **source venv/bin/activate**

**Then type: flask run**

**To get out of venv type ‘deactivate’**

**To activate debug mode (so that you don’t have to kill the server and start again after making changes, just refresh the browser):**

flask --app app --debug run

Step 5 - write the python code to produce the query outputs

Step 6 - post the result to the browser for the user to see

Step 7 - refresh the browser page so that it’s ready to process the user’s next query

\

**Project description to submit**

**Querying asset climate risk results – converting Json data into a sqlite3 database to be queried through a user-friendly web browser interface**

In this project I built a Flask-based web application to enable a user to gain insights into their asset’s modelled climate risks results (originally stored in JSON format) by submitting a few basic queries through a (drop down box-based) browser interface.

The answer to each query (submitted by a user by clicking a ‘submit’ button) is displayed at the bottom of the webpage (e.g. “The asset's VAR in your chosen year of 2090 is: 0.00022”).

To enable this browser-based querying process I had to complete the following key steps:

* Using Flask, set up a server connection through which to display information in a web browser
* Using python dictionaries, pull out the relevant data from the JSON file and use it to populate two sqlite3 database tables.
* Set up a HTML file to enable the browser display of the following three basic query types, using drop down boxes (note VAR, TIP and FP are three commonly used climate risk metrics):
  + What is the asset’s value at risk (VAR) / technical insurance premium (TIP) / failure probability (FP) in a given year (user chooses a year from a drop-down box)?
  + What is the hazard with the highest VAR / FP in a given year (user chooses a year from a drop-down box)?
  + In what year was this asset's maximum (user chooses a metric – VAR / TIP / FP) reached?

Following is a description of all files used in the project and the functions they each perform:

| **File name** | **File type** | **Key functions performed** |
| --- | --- | --- |
| temp.json | JSON | * Holding the original dataset of modelled asset risk results in a lightweight format |
| Queries.py | Python | * Enabling the querying (through the use of cursor.execute commands) of the sqlite3 database so as to respond to any of the three types of questions the user can submit in the browser |
| App.py | Python | * Calling the appropriate function for each application route, to handle both the GET and POST cases * Rendering the html templates to be displayed in the browser |
| Create\_changing\_table.py | Python | * Generates the schema of the full SQL table user to answer user queries |
| Json\_to\_SQL\_Code.py | Python | * Converting all metric data read from the JSON file into Python dictionaries (e.g. Haz\_FP\_by\_Year and Hazard\_VAR\_by\_Year) * Creating the sqlite3 database holding all the relevant asset risk information the user will be able to query through the browser interface |
| Project\_webpage.html | HTML | * Setting up the querying text the drop down boxes and the ‘submit’ buttons to be displayed in the browser |
| MyDataBase.db | sqlite3 database | This sqlite3 database contains the following 2 tables, which between them hold all the information originally held in the JSON file:   * full\_table – containing all modelled risk information to be used in answering the user’s climate risk queries, i.e. all modelled FP and VAR results for each year between 1990 and 2100, for 8 climate hazards and for the asset as a whole, as well as TIP results for the asset as a whole * non\_changing – containing general reference information about the asset, such as its geo- location (in latitude and longitude) and its heat threshold |