|  |
| --- |
| Technical Documentation  *Authors*:   * *Kirran Kayani* * *Kantai Melau* * *Chadi Ghosn* |



***Table of Content***

Contents

[Sources 3](#_Toc115370662)

[Web Scraping 4](#_Toc115370663)

[ETL process 5](#_Toc115370664)

[PostgreSQL DB 6](#_Toc115370665)

[Flask 7](#_Toc115370666)

[HTML 8](#_Toc115370667)

[CSS 8](#_Toc115370668)

[JavaScript 9](#_Toc115370669)

[I. Plotly.js 9](#_Toc115370670)

[II. Charts.js 9](#_Toc115370671)

[III. Leaflet.js 9](#_Toc115370672)

[Tableau 10](#_Toc115370673)

# Sources

There were two data sources used in creating a database for this project.

The first data source was a CSV file taken from Kaggle (linked below). The file contains Guardian League table data for UK universities from 2013 to 2021. <https://www.kaggle.com/datasets/leonler/the-guardian-league-table-main-table-20132021>

The second data source was from the Guardian website (linked below), which we used to obtain the latest year’s data via web scraping.

<https://www.theguardian.com/education/ng-interactive/2021/sep/11/the-best-uk-universities-2022-rankings>

# Web Scraping

Scraping was done on: <https://www.theguardian.com/education/ng-interactive/2021/sep/11/the-best-uk-universities-2022-rankings> to fill in the missing 2022 data. The following was undertaken:

1. The libraries to be used were imported. They included ‘Browser from splinter’, ‘BeautifulSoup’, ‘ChromeDriverManager’ and ‘pandas.
2. The chrome browser was then initiated and opened to access the soup.
3. From the soup results, iterations were done using loops within the table in order to obtain the required data.
4. The data was stripped to remain with the text only and appended to empty lists that had been created.
5. The lists were then merged to form a table.
6. The table was converted into a Data Frame.
7. From the Data Frame, the table was exported as a CSV which is accessible in the resources folder.

# ETL process

The “Extract, Transform and Load” process started by importing all needed python modules from the environment that we created. We used “Pandas”, “bs4 | Beautiful Soup”, and “SQL alchemy | create-engine”. Afterwards, we imported our two sources of data, by reading the csv file using the pandas. read\_csv method.

After analyzing all available fields in the tables, we identified the main and most crucial columns in the table, and we had to make sure that the data integrity is positive. So, we used a function to be able to identify null values in the tables. The function is “.isnull().any()”. The output was a list of all table columns with a Boolean of true or false. True if the column had null values, and false if not.

As mentioned above, we need the columns “Year and Institution” to be “False” i.e., no null values, which it was in both tables. Therefore, we were able to continue the ETL process.

Moreover, we looked at the column names from the main source website, and we discovered that the naming convention used in the excel file provided by Kaggle was incorrect, which led us to rename the column names to be able to achieve a perfect match as we know ahead that we will need to append both tables before pushing the data to the SQL server.

We prepared the dimension that we will need for our entity relational diagram by using the .unique() function to create a column with a high cardinality that would be used for filtering and slicing the fact table.

Moreover, we used the .append() function to append both tables from both sources into one fact table. We exported it to excel to be used in Tableau for reporting, as well as exported it as a json file to be used with JavaScript in our website.

Finally, using SQL Alchemy, we connected out Jupyter notebook to PostgreSQL DB using a local connection, and we pushed both tables to an existing created database and tables in Postgres. We used if\_exists=’append’ so if the table is not empty, it will append the data instead of hitting an error.

# PostgreSQL DB

We decided on using PostgreSQL Database, as it is a relational database. This decision was based on the fact that this type of database, provides an instinctive way to represent data which allow easy modification of the relationship between tables (DIMs and FACTs). When we talk relational DBs, it automatically refers to structured data. We used structured data as it provides faster indexing, easy updating and deleting records, complies with a data model, and has a standardized format.

In our case, we created a database for this project called “Uni\_Ranking”. Afterwards, using the “CREATE TABLE” function using T-SQL, we created our Fact table and Dimension table, by using the same column names and data types from the fact table created in the Jupyter notebook.

After running the notebook, we used a “SELECT \*” statement to make sure that the data was populated in the database and ready for usage.

And finally, we did a UT – unit testing to make sure that the data matches between source and the data warehouse.

Graphical user interface

Description automatically generatedOur entity relational diagram is as follows:

# Flask

The Flask API was used to develop the web application.

The following process was undertaken:

1. Libraries were imported. The libraries used were ‘Flask’, ‘render template’ and ‘psycopg2’
2. An instance of Flask was created.
3. A connection to the PostgreSQL database was created to obtain the data. This was done using a function which returned the connection.
4. An app route to the index.html was created. This was then coded to fetch the top 10 Universities from the list using a loop.
5. The universities were then stripped to retain just the text contained.
6. This was then returned to the index.html.
7. A route to the DF route (“/DF”) was created to link to the “DF” page through the “DF.html”.
8. A route to the tableau (“/tableau”) was created to link to the “tableau” page through the “tableau.html”.
9. The instance was then closed.

# HTML

We rendered our Flask app using three HTML templates: index, DF, and tableau. The navigation bar and carousel elements were shaped utilizing Bootstrap’s CSS and JavaScript.

The index page references the top 10 universities for the latest year in a list, returned within the Flask app. This page also contains a map, pinpointing these universities and their rank. The map is created using the Leaflet.js library, which is discussed below.

The DF page contains two dropdown buttons in which the user can select the Factor they are interested in, and the year they want to look at. The dropdown data is generated using D3.js. The options the user selects will update three visualizations on the page, all created using Plotly.js.

The first visualization is a bar chart illustrating the scores (in descending order) for each university for the chosen factor. The user can zoom into this chart to take a closer look into the data. The second visualization shows a scatter plot of the rank of each university within the chosen year. The final visualization displays detail of all universities within the chosen year. The default graphs are for the factor “Guardian score / 100” for the year 2022.

Below this, there are a second set of dropdown buttons which allow the user to compare the main factors for any two universities they are interested in. This updates a radar chart, which is created using Chart.js.

The final tableau html page embeds the Tableau dashboards within the page.

# CSS

We used a CSS stylesheet to enable the HTML pages to be responsive, in particular when the window size is reduced the navigation bar collapses. This was done using a CSS media query.

The CSS was also used to format the font, color and position for the different elements of the HTML, in addition the sizing of the map and tableau element on the page.

# JavaScript

## Plotly.js

Plotly is the main JS library that we used for data visualization. We used division ids from the html file to reference the html code that we will be modifying and using in JavaScript.

For example, we have an id called “selDataset” which represents the options of the drop-down menu in our website. We used it to be able to connect to the button and populate it with the options from the json file.

We used a loop to be able to extract the needed information to use it in our logic. Moreover, we used the chosen option “year” and “institution” to filter the data from the json file and append it to a list.

Then, to be able to plot the data, we create a trace where is specify the chart type, title and other information and then we plot the charts.

We created an if statement where it will set the start of the loop and the end of it based on the year selected. We were able to use the index of the data from the json file to set it up.

We used the layout to be able to fix the layout of the bar chart, scatter chart and the table.

## Charts.js

The charts library was used in our JavaScript to be able to plot a radar for university benchmarking.

We then used 6 different columns in the radar for comparison. And based on the selected university, the json file will be filtered to retrieve the values for the 6 chosen columns for the most recent year (2022) and displays the data accordingly.

We finally used some layout functions to set the background color, hover function, border color and the points over the radar chart.

## Leaflet.js

To produce a map highlighting the top 10 universities on the index page, we utilized Leaflet.js. We created an array of objects for the top 10 universities, containing the university location (latitude and longitude coordinates), the name and the rank. We then looped through this array to create markers on the map which, when clicked on, show the name and rank of the university.

# Tableau

Tableau was used to visualize the data. For this, we thought it useful for students looking for an institution to enrol in, to get key information. This was with how the cost varies by rank. Are top ranked institutions more expensive than the rest? How satisfied are the current students with the institution? How does the institution fair across different years? How big is the variation in terms of the rank relative to other years? How did the University rank according to the Guardian Score? How do the entry tariffs vary according to institutions? This would determine how easy it would be to get into various institutions.

From this, especially displayed graphically, it would be easier for a student to make an informed choice as to which University would be best suited for them.

The following steps were undertaken:

1. The excel sheet that contained the merged data was imported to form the Data Source.
2. Various sheets were developed to visualize the data. They include:
   1. Rank vs Cost. This plotted the institution rank vs the spend per student. The Rank was used to colour the plots and the details of the institution were added. This was then arranged in pages depending on the year.
   2. Average Satisfaction. The Average Satisfaction with the course was plotted against the Institution. This was then set to display in descending order with the rank as the detail.
   3. University Rank Difference. A table was created to plot the difference in rank between the current year and the previous year. This was possible through the use of a calculation to determine the rank difference.
   4. University Guardian Score. A bar graph was created to show the Average Guardian Score versus the Institution. The institution charts were then coloured by the rank division. This was through a calculation to divide the institutions according to the rank for the specific year.
   5. University Entry Tariff. This was displayed per institution with the rank being used to detail the colours for identification of the best performing institutions in that specific year.
3. From this, 2 dashboards were created to display the graphs.