PROJECT: MONITORING FLIGHT INFORMATION

Introduction

This project is concerned with a large dataset that consists of flight information in 2023. The end product will be a dashboard in which various information on flights can be monitored.

Part 1: Getting acquainted with the data

In the first part, you will use only one of the tables in the data, namely `airports.csv`. Use `pandas` to read the `.csv` into Python and save it as a `DataFrame`. This dataset is part of a larger database on all flights departing New York City in 2023 and contains information on various airports. Use this dataset to perform the following tasks. It is advised to use the functions such as `scatter_geo` from the library `plotly.express`.

- Set up a map of the world with points on it indicating the airports in the set.
- Identify the airports outside of the US and in addition, make a new map of only the US.
  Extra: Color code the airports by their altitude
- Make a function that takes the FAA-abbreviation of the name of an airport as input and then plots a map of the world and add a line from NYC to the airport on that map.
  Extra: Specify the function to make a map of only the US if the airport is located in the US
- Extend the previous function to accept a list of FAA-abbreviations and plot a line for each of the multiple lines for each of the airports.
- Look op the position of John F. Kennedy airport in New York City and compute the Euclidean distance $\sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2}$ for each airport and visualize the distribution of the distances in a suitable figure.
- Since the earth is not flat, it makes much mor sense to compute the geodesic distance between two airports. i.e. the length of a circular arc connecting them. For two locations with difference $\Delta\lambda$ in longitude and $\Delta\phi$ in latitude, this distance is given by

$$R\sqrt{\left(2\sin\left(\frac{\Delta\phi}{2}\right)\cos\left(\frac{\Delta\lambda}{2}\right)\right)^2 + \left(2\cos(\phi_m)\sin\left(\frac{\Delta\lambda}{2}\right)\right)^2},$$

  where $\phi_m = \frac{\phi_1+\phi_2}{2}$ is the midpoint of the two latitudes and R denotes the radius of the earth. Look up this radius and repeat the previous task for this distance.
- Analyse the different time zones; Make a graphical representation of the time zones of the airports that represents the relative amount of flights to them.

Use your own creativity to discover any other features about this dataset you find interesting or noteworthy! Look for relations between variables of create attractive visualizations of the data. Points will be awarded for creative insights or features of the final product!

Create one `.py`-file which, upon execution produces all required figures

Part 2: Setting up a GitHub repository

The final product needs proper version control. Therefore everyone in the group will need an account on GitHub. Set up a private repository with your group and add a `.py`-file to it containing the code used for Part 1. Make sure to add a `README.md` file to it.

The `README.md`-file is very important as it contains information for users of the software you create. It should therefore serve as a guide for a use to use your library. `.md`-files can be opened and edited in any text editor, including Visual Studio Code.

Initialize `git` Take turns within your group and add small sections to the `README.md`-file each time. Between these turns `commit` and `push` the changes until you are satisfied with the file.Make sure to keep updating the `README` throughout the project, as this file will also be judged at the end.

Part 3: Interacting with the database

The file `flights_database.db` is now available on Canvas. This database contains the following tabular data:

- `airlines`: contains information of the airlines operating from NYC.
- `airports`: The table you have been working with in part 1.
- `flights`: A very large table containing all (425,352) flights departing NYC in 2023 including flight information.
- `planes`: Information on the planes used.
- `weather`: hour-by-hour information on the weather in 2023.

Use the library `sqlite3` to connect to the database. For each task of the remaining parts, it is most prudent to write `SQL`-queries to acces the parts of the database needed for that task, rather then loading it all into one or more `pd.DataFrame`s.

For this part, you have to investigate the entire database and link the different tabular data together. There is a lot to investigate. Below are some interesting tasks to complete with this database, but as with part 1, feel free to investigate further!

- Verify that the distances you computed in part 1 are roughly equal to the distances in the variable `distance` in the table `flights`. If they are much off, recall that latitude and longitude represent angles expressed in degrees, while the functions `sin` and `cos` expects entries in radial angles.

- For each flight, the origin from which it leaves can be fount in the variable `origin` in the table `flights`. Identify all different airports in NYC from which flights depart and save a `DataFrame` contain the information about those airports from `airports`
- Write a function that takes a month and day and an airport in NYC as input, and produces a figure similar to the one from part 1 containing all destinations of flights on that day.
- Also write a function that returns statistics for that day, i.e. how many flights, how many unique destinations, which destination is visited most often, etc.
- Write a function that, given a departing airport and an arriving airport, returns a `dict` describing how many times each plane type was used for that flight trajectory. For this task you will need to match the columns `tailnum` to `type` in the table `planes` and match this to the `tailnum`s in the table `flights`.
- Compute the average departure delay per flight for each of the airlines. Visualize the results in a barplot with the full (rotated) names of the airlines on the x-axis.
- Write a function that takes as input a range of months and a destination and returns the amount of delayed flights to that destination.
- Write a function that takes a destination airport as input and returns the top 5 airplane manufacturers with planes departing to this destination. For this task, you have to combine data from `flights` and `planes`.
- Investigate whether there is a relationship between the distance of a flight and the arrival delay time.
- Group the table `flights` by plane model using the `tailnum`. For each model, compute the average speed by taking the average of the distance divided by flight time over all flights of that model. Use this information to fill the column `speed` in the table `planes`.
- The wind direction is given in `weather` in degrees. Compute for each airport the direction the plane follows when flying there from New York.
- Write a function that computes the inner product between the flight direction and the wind speed of a given flight.
- Is there a relation between the sign of this inner product and the air time?

Part 4: Data wrangling

Now it is time to do some wrangling. Again, there are many interesting things to be done with the data. Below, we list some tasks that can be done using the data wrangling techniques in the lecture.

- Check the table `flights` for missing values think of ways to resolve them.
- Look for duplicates in the flights table. Take into account that here a flight number can occur multiple times, only count it as duplicate when the same flight appears multiple times.
- Convert the (schedueled and actual) arrival departure and departure moments to `datetime` objects.

- Write a function that checks whether the data in `flights` is in order. That is, verify that the `air_time`, `dep_time`, `sched_dep_time` etc. match for each flight. If not, think of ways to resolve it if this is not the case.
- Create a column that contains the local arrival time, incorporating the time difference between arrival and departure airport.
- In addition, information on the different types of planes and airlines will be important. Consider studying what the effect of the wind or precipitation is on different plane types.
- Next week, the specifics of the dashboard will be released. Count on requirements asking for the statistics for different departure and arrival airports. Consider making functions that take these airports as input and generate numerical or graphical representations of the data grouped by them. Many of the tasks from the previous parts can be grouped!
- The dashboard will also contain a general results tab that can show anything you find remarkable in the data. Think of things you would like to display here. Are there interesting relations to be seen in the data? Are there airports that stand out when it comes to delay? Are there plane times that fly considerable faster? Which plane routes are taken most often from NYC? Is there a significant relation between (some) weather variables and the delay time? Come up with convincing graphical or numerical summaries that can be displayed in this part of the dashboard to display the results of the study.

Part 5: Creating a dashboard

Install `streamlit` and set up a script used to create the dashboard. You are free to choose the layout of the dashboard as well as the different options for navigation through it. Below, the minimal requirements are listed. The users of this dashboard will be business analysts working for the airports on NYC. You may use the graphs created in part 1 to make the dashboard more visually pleasing.

Minimal dashboard requirements: The dashboard should at least contain the following

- An opening page with general statistics of the flights departing from NYC, these statistics can be any of the results found in earlier tasks, this page should contain at least one numerical summary and one graphical summary of the data.
- The dashboard should have a sidebar in which an arrival and departure airport can be selected showing flight statistics for this combination.
- The dashboard should include an analysis of the delays and possible causes. Think about time of day, weather or other factors. You can choose to split this analysis per departing airport or apply it for the general data.
- The dashboard should contain statistics as a function of time. That is, it is possible to enter a date somewhere in the dashboard and see relevant statistics of the airport(s) such as delay times on that day.

See the Assignment page for a detailed grading rubric of the assignment. A point will be deducted if the dashboard does not meet the minimal requirements. However, most points can be earned by including extra features using the research done in earlier parts of the project.

Part 6: Deployment and submission

Publish the repository. Go to your GitHub repository and go to settings. In the "Danger zone", you find the option to `Change visibility`, which can be used to set it to public. To check that it worked, try finding the repository in incognito mode, while not logged into GitHub.

Deploy the dashboard. Sign up in the Steamlit community cloud and connect your GitHub account following the steps in the slides of Lecture 7. Again, to see whether it worked, you, or a group member, can look for the dashboard incognito.

Submit the assignment. To submit the assignment, one of your group members must hand in two links:

- A link to the location of the dashboard.
- A link to the GitHub repository.

After submitting, the repository should not be changed anymore.

The group projects will be graded per group and not individually, unless the GitHub repository suggests the work is not distributed equally enough. In this case, your group will be contacted.