Technical Analysis

Using data from the Bureau of Transportation Statistics, we have chosen to use flight data to create an interactive dashboard involving airline flight delays. Using statistics acquired from a government resource, we have taken data from multiple months for the year 2020-2021 for the first quarter. Specifically, December of 2020, and January through March of 2021. Data can be found at <https://www.transtats.bts.gov/DL_SelectFields.asp?gnoyr_VQ=FGK&QO_fu146_anzr=b0-gvzr> Each month includes approximately 360,000 data points, making our total data haul over a million piece of data. Not all data was used. We took a sample from each month, using the pandas random function, and then extracted 11% of the original data into a new dataframe. The main thing being analyzed are airline delays by timeframe and by airline. We started with multiple files of a total for more than 500MB. We then took those files and merged them together. Once the files were merged, we then dropped any unwanted or unnecessary data. Such as empty columns or redundant data. After the data was cleaned, we still had too large of files.  So, we randomized 11 percent of the total data and stored that into a DB. So from the original .csv, to a dataframe, to the 11% .csv, and then into a SQLite database. The user should be able to use the dashboard to determine which airline has the most delays, which day of the week has the most arrival and departure delay, and the ability to compare airlines and the reasons for delay, by airline. All parts of the dashboard are interactive, using d3 code in JavaScript.

After extracting the dataframe of 11% from pandas, it was converted into a .csv. In order to read the .csv, a SQLite database was created. In order to read the SQLite database, we had to create our own API using flask. A connection was created in an app.py file to the previous flight delay data using conn = sqlite3.connect. A query was created using basic SQL to left join carrier data with delay data using the actual market carrier name (the name for the airline). After creating a home @app route, more routes were created for each element of the dashboard. Each route jsonified a different part of the original data, and then used that JSON as a data source for the target JavaScript on the actual deployment of the dashboard. Some python was used in each route to accomplish different things for each route, such as grouping the data by a different series, or finding the average of a set of data.

For the JavaScript files, d3 and plotly were used. In two of the bar graphs on the dashboard, plotly was used using multiples traces and advanced layouts. In the scatter plot, no plotly was used, opting for a pure d3 scatter plot. An svg was created, height and width were defined, margins set, and then x axis and y axis data inserted. The scatter plot visualization was an edited version of a class assignment. The sunburst visualization will be described later.

To display everything, an index.html file was created as the final step between our visualizations and our display for the viewer. D3, plotly, and Bootstrap were inserted into the HTML file. Bootstrap was used to format the visualizations using <div> classes, then the actual visualization using <div IDs>. Since we used <div ID> in the HTML, hashtags (#) were used to insert the JavaScript into the index.html file. A small dictionary was inserted using bootstrap and <p> tags to define the different reasons for delay displayed in one of the interactive bar graphs.

The sunburst is populated by a hierarchical json flare file. The json contains lists of dictionaries following the levels State =>  Airport => Airline => Delay Reason. The chart lets the user drill down through the levels to show the percentage of delay reasons for each airline at a specific airport. For readability, the dataset is only the top 50 airports in the US. The json was created by first taking the state, origin airport, market carrier, and delay reason columns from the full 11% dataframe. The rows were aggregated to combine the minutes delayed for each reason by each airline at each airport. With the clean dataframe a function was used to iterate through the rows and organize them into the necessary hierarchy. The Javascript code was downloaded from observable, it includes an app.js file with all the d3 for the chart, a runtime.js file, an inspector.css file, and an index.js file. All these files are called from the html and work together to display the chart. Some JavaScript code has been included below.

Text

Description automatically generated

Figure - Scatter Plot

Text

Description automatically generated

Figure - Dropdown Menu for Bar Graphs

Text

Description automatically generated

Figure - JavaScript for Starburst Dimensions