Flight Analytics: Airport Weather Cancellations ETL Pipeline

By Azucena Faus, Jacqueline Urenda and Saba Alemayehu

GitHub Link: https://github.com/fausa/ADS507_Team1_Final_Project

Objective:

The primary goal and function of the Flight Analytics ETL pipeline is to extract monthly weather and airport flight cancellations data from a daily airport weather from a comma-separated (CSV) file (Kaggle, 2021) and daily flight information for those same dates from an airlines database (Relational Dataset Repository). Once the data is normalized; that is, cleaned, processed, merged, and made useful to the data science team, the data is loaded into a data warehouse that is readily available for data scientists to work with. The data scientists can then use the data to create time series forecasts to predict flight cancellations for that same month the following year. The data for this pipeline contains only the month of January and therefore serves as a snapshot of how the data will be inputted as more monthly data becomes available.

For the output and visualization, we plan to load the data into a schema implemented by the data warehouse called domestic_flight_weather_database so the data science team can develop time series forecasting models or other predictive models to understand how date, weather type, weather severity, airline carrier and airport locations affect flight cancellations. Finally, an interactive heatmap of flight cancellations and top weather conditions per airport will be provided as output to the pipeline.

Source Data:

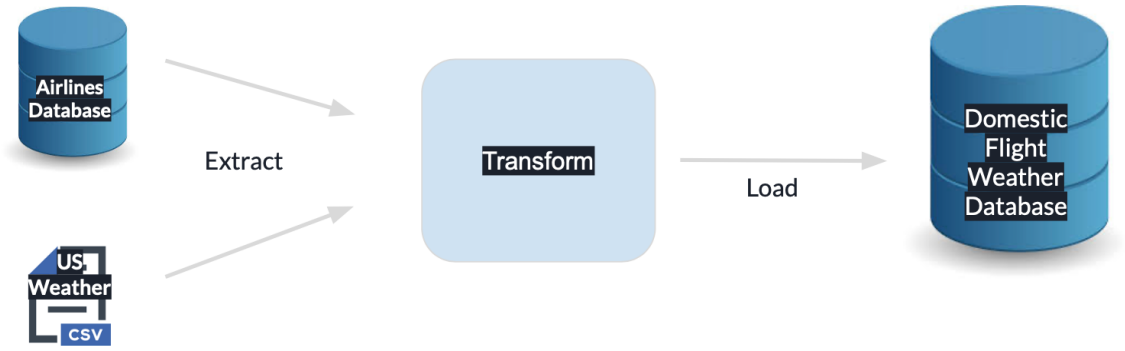
Airlines Database: This database was obtained from the Relational Dataset Repository. This database contains flight data within the United States for the month of January, 2016. Tables include information regarding flight time performance (information whether flights went on as scheduled, delayed, or canceled, as well as reasons for cancellations), airport code and location

data, cancellation codes, airline information, etc. This database was chosen because it contains useful information that can be used to help predict flight cancellations.

US Weather Events: This dataset is a csv file obtained from Kaggle. The dataset contains weather details reported by airports within the United States for the years 2016-2021. Columns include weather type, weather severity, weather start time, weather end time, location in latitude and longitude, as well as the airport code.. This dataset was chosen because the weather details centered on U.S. airports can be used in conjunction with the flight data in the airlines database to coordinate flight cancellations due to weather, with the weather that causes the canceled flights.

ETL Pipeline Process:

The Airports Weather Cancellations ETL pipeline in its current form, is manually deployed and monitored. As more data is collected, future renditions will be run in batches on a monthly basis.



Data Extraction:

- Tables were extracted from the Airlines DB using embedded SQL in a Visual Studio Code Python notebook, and converted to dataframes:
    1. l_airport → airport_codes_table

2. l_airline_id → airlineID_carrier_table
3. l_cancellation → CancellationCode_table
4. on_time_performance_2016 → airline_flights_table

- US weather Data CSV file was loaded into visual studio as a dataframe using python
    1. Weather US → Weather_df

Data Transformation:

Weather data contained nulls for the City column, however, all airport codes were available so it was decided during the merge, weather and fight data would be merged on matching airport codes. Flight dates in the airline data were not compatible with the datetime format used in the weather data, so in order to match FlightDates to lie between the start and end weather dates in the weather data, this field had to be transformed.

All the nulls in the airline data were in columns that were deemed unnecessary for our purposes.

A new feature was generated in case data scientists are interested in finding relationships to canceled flights due to weather and the combination pattern of weather type and weather severity. A cross product of the types and severities of weather was found and turned into a dataframe to have indices that codify weather type and severity in a variable called weather_code. This variable was then added to the Weather_df dataframe.

- Transformed date and time columns in airline_flights_table for compatibility with the Weather_df dataframe.
- Modified and cleaned columns in the weather_df dataframe
- Split up the column called “Description” in the airport_codes_table into separate columns that contained each airport codes’ related airport name, City, and State.

- Created a Weather_comb dataframe from the possible combinations between ‘type’ and ‘severity’ columns in the Weather_df dataframe. This unique code is called weather_code.
- Added this weather_code feature from the Weather_comb dataframe into the Weather_df dataframe

Loading/New Database Creation:

Creation of the database was done as soon as a connection was established with the SQL server at the beginning of the code. The loading step involved converting many of the dataframes that had been processed during transformation, into tables. Two of these dataframes were also merged using the JOIN statement. Those dataframes were airline_flights_table and Weather_df.

- Created the domestic_flight_weather_team_1 database using SQL
- Created new tables and inserted modified/transformed data frames into new SQL database
    1. Weather_df → airport_weather
    2. airport_codes_table → airportID_table
    3. airlineID_carrier_table → airline_id_table
    4. CancellationCode_table → cancellation_code_table
    5. Weather_comb → Weather_TypeSeverity
    6. airline_flights_table → airline_copy (select columns)
- Join airline_copy and airport_weather tables based on airport code and flight time

Output/ Visualization:

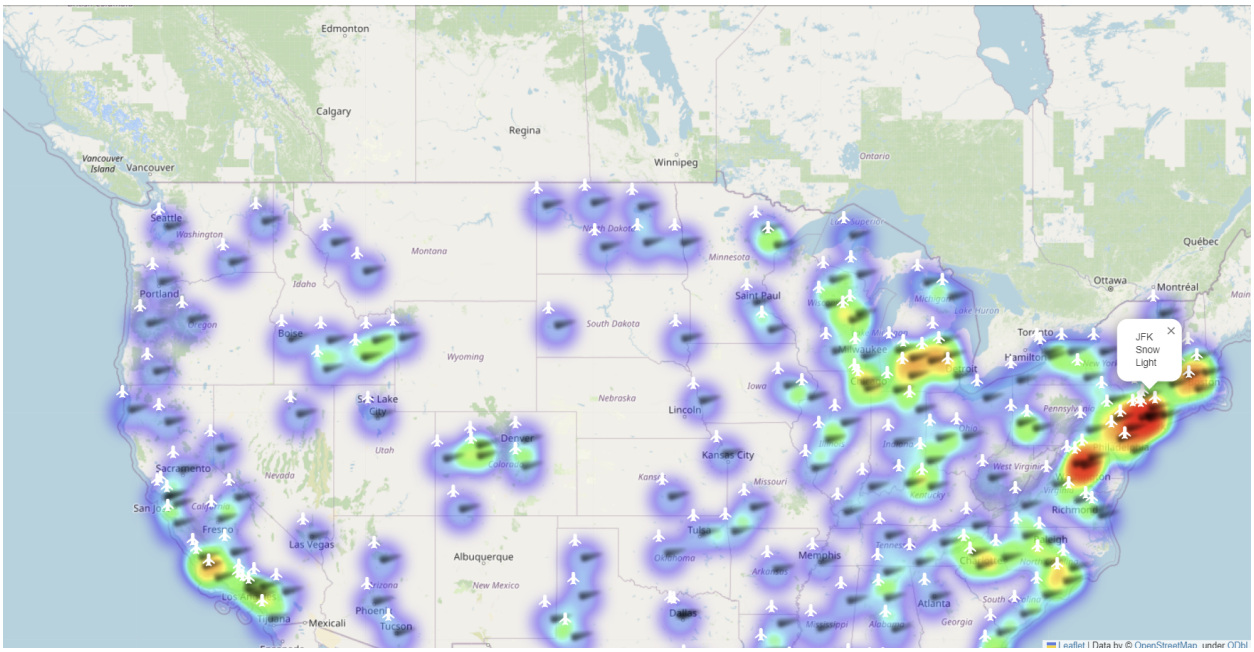
The main table that was used for visualization and will be used by the data science team for forecasting is the flight_origin_weather_table, which was created by joining the airline_copy

and airport_weather tables. This table contains important columns such as flight date, airline id, flight origin, location latitude & longitude, flight cancellation, cancellation code, weather code, origin weather type, origin weather severity.

The goal was to generate a heatmap of canceled flights throughout the continental U.S. with airport name markers. First, it was important to obtain the base map and so EarthPy was used (Earth Lab, 2021). Python packages geopandas and folium were also used for this aspect of the project.

The main flight_origin_weather_table was then used in conjunction with the Cancellation_codes_table to develop a heatmap of canceled flights in the continental U.S. for the month of January, 2016, along with a clickable marker to indicate the name of the airport, the top weather condition for most canceled flights, and the top cancellation reason provided by the airline flight data. The top number of canceled flights that month were in Washington DC and Baltimore, due mostly to Snow weather conditions. In second place were La Guardia, JFK and Newark, also due to snow. The output map is saved onto the local drive of the data engineer who manually runs the script, as an HTML file for ease of view.

Weather Flight Cancellations Heatmap:



System Information, Improvements & Next Steps:

The system will likely scale as the dataset size grows since the plan is to process data in month-sized data batches. As mentioned, this ETL process for the month of January only provides a snapshot of how the data will be extracted and processed on a monthly basis. If weather data is collected at a higher granularity, that is, if weather data were to scale up significantly, or if there is interest in more airport data outside the current continental U.S. list obtained here, there is the option of partitioning data tables to help performance of queries in the pipeline.

It is also important to note that this ETL pipeline may one day work with streaming data; where data science teams might want to use current weather conditions and past data for the same time frame, to predict next day or future cancellations among other things. In that case, the

pipeline could be improved to be ready to handle streaming data, or shorter intervals of batch processing.

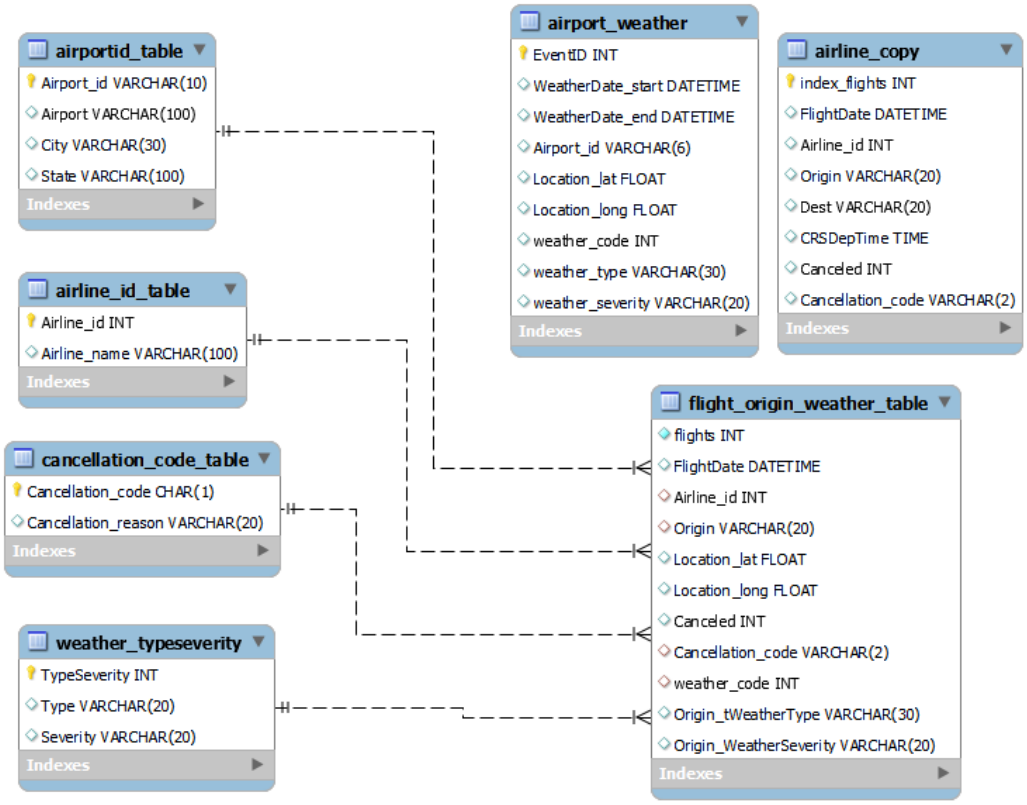
Additionally, a step to simplify this process would be to create a more automated process in which the data can be easily extracted as more data becomes available. There are several hard-coded aspects to the current deployment that can be modified to accept user-input. These methods have yet to be explored and implemented, however possible automation considerations include code-based tools that can provide customs scripts and blueprints of how the automated process should work. Examples of this include Airflow, Dagster, and Prefect (Floris).

Another opportunity for improvement is to add tiles to the final output dashboard that depict more detailed insights, statistics, and aggregating views, such as analysis of the differences between airlines or if destination weather plays an equally important role in flight cancellations. This can help our business analytics teams with their own analysis.

A pivotal step to ensure the success of this data pipeline is to conduct routine maintenance. This will be an ongoing process to make sure there are no data errors and that the ETL process is running smoothly (Database Maintenance Explained). Routine maintenance and database management can be automated through Amazon Web Services (AWS). Conducting database maintenance will also assist in making the system more secure.

While on the subject of security, the current approach is that the code runs on an internal server that requires user credentials to work with, but the database currently does not require a password for direct access. A next step towards increasing the database security would be assigning database user access based on the security authorization level of each member on the data science team - a perfect opportunity to apply the Principle of least Privilege (Reis & Housley, 2022).

Architecture Diagram:



References:

Beaulieu, A. (2020). *Learning SQL*. O'Reilly.

Earth Lab. (2021). *Download data with EarthPy*. https://earthpy.readthedocs.io/en/latest/gallery_vignettes/get_data.html

Floris, S. (2022, March 7). *The Complete Guide to Data Pipeline Automation*. Medium. Retrieved from https://medium.com/codex/the-complete-guide-to-data-pipeline-automation-6e25b5bcfd9e

Holtz, Y. (2018). *The Python Graph Gallery*. Python-Graph-Gallery. https://www.python-graph-gallery.com/map/

Kaggle. (2020). [US Weather Events (2016 - 2021)] [WeatherEvents_Jan2016-Dec2021.csv]. https://relational.fit.cvut.cz/dataset/Airline

Lewis. R. (2021, March 14). Plotting Maps with GeoPandas Beginners guide to geospatial data plotting. Towards Data Science. https://towardsdatascience.com/plotting-maps-with-geopandas-428c97295a73

Reiss, J., & Housley, M. (2022). *Fundamentals of Data Engineering*. O'Reilly.

Relational Dataset Repository. [airline][U.S. Department of Transportation (DOT), Bureau of Transportation Statistics (BTS)]. https://www.kaggle.com/datasets/sobhanmoosavi/us-weather-events

Database maintenance explained. OfficeTools. (2015). retrieved from https://www.officetools.com/knowledgebase/database-maintenance-explained/