Weather-Based Outfit Recommendation Tool : The Report

Github Repository Link: https://github.com/sarafina-chea/Final_Project_206.git

1. **The goals for your project including what APIs/websites you planned to work with and what data you planned to gather?**

   The project aimed to revolutionize the concept of dressing for the weather. Initially, we opted to work with Rapid API Asos, a British online fashion and cosmetic retailer. Still, we found that it needed more information regarding clothing products as it only included stock numbers and item names. Our goal was to create the most powerful and valuable Weather-Based Outfit Recommendation Tool with relevant information. We found that the necessary information is to be added to the product details for Asos. As a result, we found a Forever21 API to work better as it included a patent category, share link, product image, list price, discount price, etc. Rapid API, Forever21 API documentation as it helped to query all information about categories and products, as it is on the official Forever21 website. We also utilized OpenWeatherMap API to gather weather data based on input location, such as temperature in Fahrenheit, humidity, wind speed, and precipitation. Additionally, we used the Vogue Website to collect the latest fashion trend data, including popular colors and patterns.

2. **The goals that were achieved including what APIs/websites you actually worked with and what data you did gather**

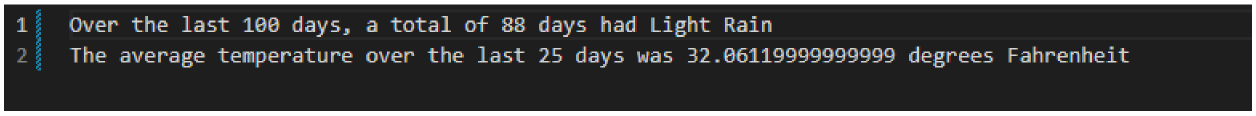
   We were able to successfully scrape the Vogue website for current trends using Beautiful Soup, the Forever 21 API for current seasonal clothing items that match Vogue trends, and gather weather data from the OpenWeather API for the current day and the past 100 days of a user's location. In addition, we successfully synthesized this data into a program that generates an outfit based on the current weather of a given location, in addition to product links to visualize the outfit and series of graphs to show trends in temperature, humidity, windspeed, and precipitation.

3. **The problems that you faced**

   While we were able to individually work on functions calling to separate APIs, we struggled with combining the functions into one file. The output from one function was at times unexpected, and led to edge cases that we hadn't considered, such as having an empty dictionary be the input for another function. We ended up spending a few days debugging (retesting API keys, accounting for bugs that had a domino effect across functions, comparing different file versions on our local computers), before we were able to fully integrate our functions into the final file.
   We also had difficulty with key errors, as at times the Vogue website (scraped with Beautiful Soup) would return a dictionary with the "product" key in it, sometimes we

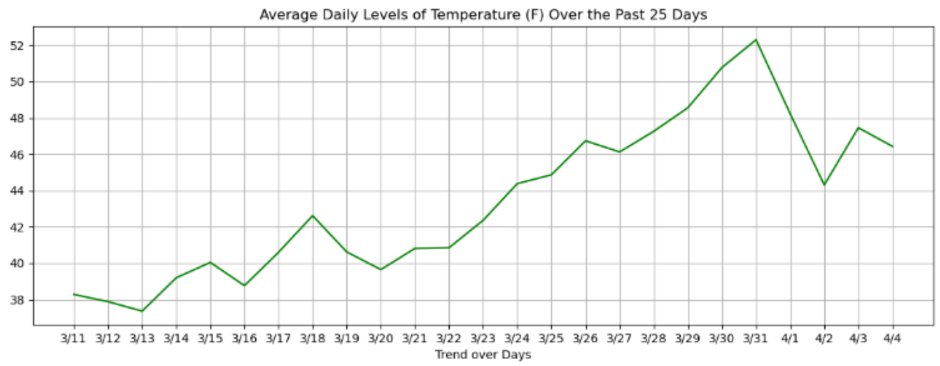
would instead get a KeyError seemingly randomly. In this case we put in try/except blocks.

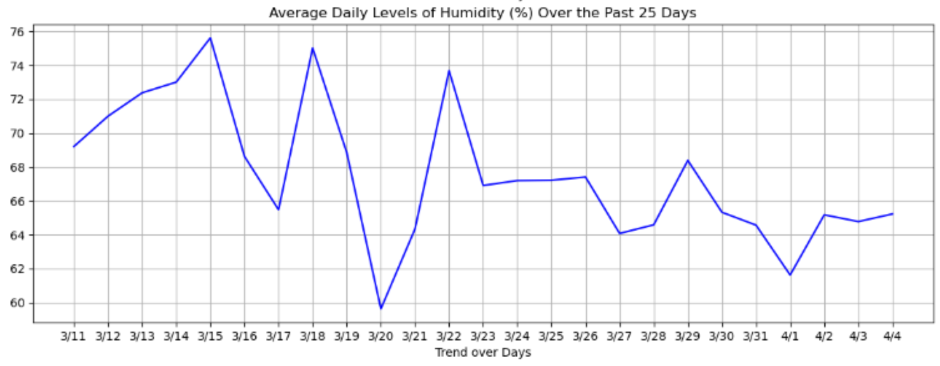
4. **The calculations from the data in the database (i.e. a screenshot)**

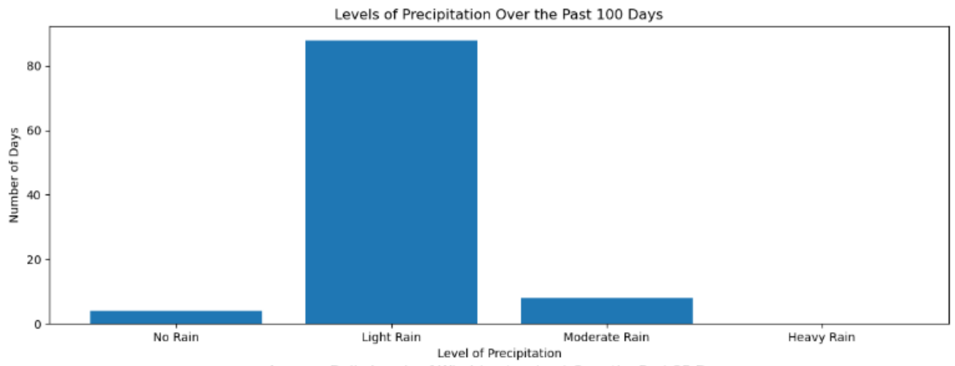


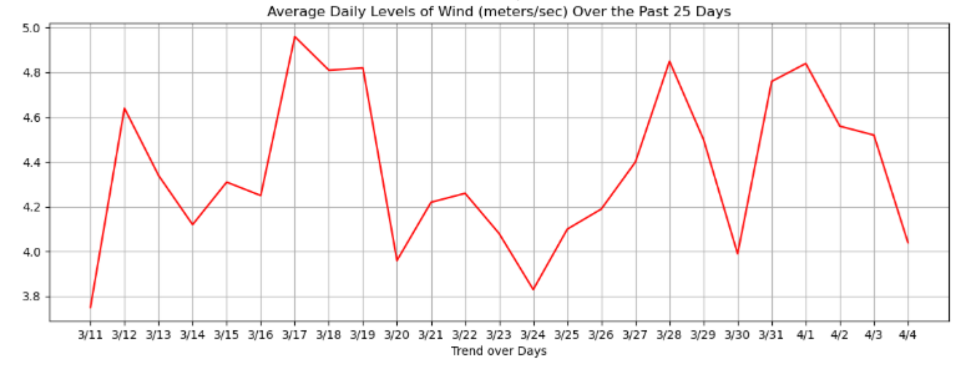
```
Over the last 100 days, a total of 88 days had Light Rain
The average temperature over the last 25 days was 32.06119999999999 degrees Fahrenheit
```

5. **The visualization that you created (i.e. screenshot or image file)**









6. Instructions for running your code (10 points)

a. Run retrieve_historical_weather_data() to gather 7 rows of weather data. Vogue trends and Forever21 product information are put into the database at the same time as they create an outfit recommendation. (Run this function 5 times to gather more than 100 rows of data total).

i. If there is a KeyError because of the ‘product’ key on line 286, please rerun the program a few times! This is a frequent occurrence, but the key does exist!

7. **Documentation for each function that you wrote. This includes describing the input and output for each function (20 points)**

get_user_location() - [Sarafina]

Purpose: Retrieves user’s location based on a given city
Arguments: None
Returns: User’s Location As a String

get_season() - [Sarafina]

Purpose: Retrieves user’s season based on input
Arguments: None
Returns: User’s Season As a String

retrieve_weather_data(user_location) - [Hannah]

Purpose: Uses OpenWeatherMap API to retrieve weather data based on user’s location
Arguments: user_location (User’s Location as a String)
Returns: JSON response containing weather data

parse_weather_data(weather_json) - [Hannah]

Purpose: Parses JSON response from OpenWeatherMap API to retrieve the relevant weather data (temperature, humidity, wind speed, precipitation)
Arguments: weather_json (JSON response from OWM API)
Returns: Extracted Data as a Dictionary

scrape_fashion_data() - [Sarafina]

Purpose: Scrapes the Vogue website to collect information about the latest fashion trends and styles
Arguments: None
Returns: HTML response from Vogue

parse_fashion_data(fashion_html) - [Sarafina]

Purpose: Parses the HTML response from Vogue website using BeautifulSoup to extract relevant fashion data (Popular colors, styles, patterns, items) from “Trends” section and “Fashion” tab
Arguments: fashion_html (HTML Response from Vogue)

Returns: Extracted Data as a Dictionary (Keys are Seasons, Clothing items and the adjective that are behind them are Values)

{Fall: [“loose jeans”, “brown flowy dress”], Spring: [“pink skirt”, …}

query_forever21_api(fashion_data_dict) - [Amy]

Explanation: Use dict, item - adjective - creating search query or link - could use beautiful soup - use beautiful soup to parse website and recommend random shirt - and pants– -separate and one at a time- don't worry about season, go based on seasons fall-or autumn or spring -winter- summer

Purpose: Queries the ASOS API for clothing recommendations based on the user’s preferred style and the current weather

Arguments: fashion_data_dict, not current weather, use Sarafinas {Fall: [“loose jeans”, “brown flowy dress”], Spring: [“pink skirt”, …} for the function, you’ll take a search term and then pick an adjective from each of the items, pick a random item for that API

Returns: Link to product list

parse_forever21_data(asos_json, clothing_article) - [Amy]

Get product ID, get more info from API - save it and for product ID use other API to return more information which includes price, etc —-- maybe use regex? Or just split on the forward slash!!!

Purpose: Parses JSON response from ASOS API to extract relevant product data (<u>Product Name, Description, Category, Image URL, and Price)</u>

Arguments: asos_json (JSON Response from ASOS API), clothing_article (top, bottom, or shoe)

Returns: Extracted Data as Dictionary

get_outfit(weather_dict, season, fashion_dict) - [Hannah + Sarafina]

Purpose: Combines the weather data, fashion trend data, and ASOS product information to make a personalized outfit recommendation for the user using a simple algorithm

Arguments: weather_dict (Weather Data Dictionary), fashion_dict (Fashion Data Dictionary), product_dict (Product Data Dictionary)

Returns: A Dict of 2 Recommended Outfits (Each Outfit Description is a List)

{fit 1: {top: green blouse [link], bottom: blue jeans [link], shoe: black heel [link]} fit 2: {top: black tank-top, …}}

main() - [Whole Group]

Putting it all together!

8. **You must also clearly document all resources you used. The documentation should be of the following form**

Date	**Issue Description**	**Location of Resource**	**Result (Did it solve the issue?)**
4/11/23	The OpenWeather API requires a date to pull historical data from, and I wasn't sure if there was a method to pull the current date without having the user input a day manually.	https://www.geeksforgeeks.org/get-current-date-using-python/	This resource allowed me to pull the current date automatically within the program without having to hard code a starting date (or have the user input the date). (It did resolve the issue).
4/14/23	Forever21 API helps to query for all information about categories, products, etc. as on official websites. One issue I had was being able to access and send an unlimited amount of requests. After a while I had to pay to send more requests.	https://rapidapi.com/apidojo/api/forever21	This resource allowed me to access information and data about the product information. Specifically it allowed me to sort based on product id to find Product Name, Description, Category, Image URL, and Price.
4/10/23	The Vogue Website was used to collect the latest fashion trend data. When getting links for articles sometimes in the article hrefs it would have https it would double up in my query string and it made my life kind of difficult.	https://www.vogue.com/fashion	This resource allowed me to access articles that would then allow me to collect fashion trends, clothing, and popular trends such as colors and patterns.
4/11/23	We were unsure what API would allow us to gather data from a specific location, as well as what categories of precipitation we could use in our code to create outfit recommendations for future weather conditions.	https://openweathermap.org/api https://openweathermap.org/api	This resource allowed us to gather weather data from the current day and over the past months given a location, and it has a list of possible

			returned descriptions of precipitation.