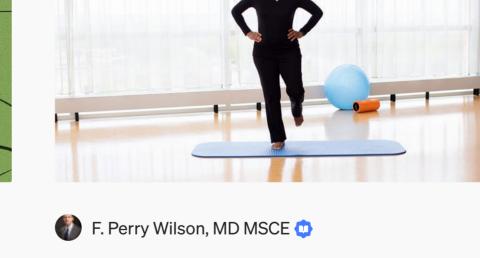
Medium + Get unlimited access to the best of Medium for less than \$1/week. Become a member My Journey into Building Scalable **Data Pipeline** Haikel Zanina 4 min read · Just now **Data Sources** Cities mySQL cloud scraping database **DenWeather** Weather **Flights** Scheduled automation ntroduction It all started with a fictional company called Gans. My instructor asked me to help them collect data from external sources to predict e-scooter movements. Sounds simple, right? But little did I know, this task would help me build my first end-to-end data pipeline using web scraping, APIs, and databases, and eventually automate the whole process. In this article, I'll share how data flows from the internet to my database every day, using Python and various libraries. WEB SCRAPING N DATA TRANSFORMATION USING PYTHON WEB STRAPPING USING CLOUD DATABASE Web Scraping: Extracting Data from Wikipedia The first step was to collect demographic data for the largest German cities. Since Wikipedia doesn't provide an API for this, I turned to BeautifulSoup, a Python library for pulling data out of HTML and XML files. **Scraping Wikipedia:** To get the list of German cities by population, I used the following code snippet: import requests from bs4 import BeautifulSoup url = ("https://en.wikipedia.org/wiki/List\_of\_German\_cities\_by\_population\_within response = requests.get(url) soup = BeautifulSoup(response.content, "html.parser") This pulls the Wikipedia page and parses the HTML using BeautifulSoup. The next step was to extract the relevant information from the table: parent = soup.find(class\_ = 'wikitable sortable') children = parent.contents[1] dictionary = {"City": [], "Country": [], "Population": []} for i, child in enumerate(children): if((i != 0) & (i % 2 == 0)): dictionary['City'].append(child.contents[3].get\_text("|", strip=True).re dictionary['Population'].append(child.contents[7].get\_text("|", strip=Tr dictionary['Country'].append(child.contents[5].get\_text("|", strip=True) if(i == 40): break Here, I manually parsed the table content, cleaning the data as I went. In practice, scraping can be messy, especially with complex HTML structures like Wikipedia's. . . . **Collecting Data Using APIs** Next, Gans asked for weather and flight information for major cities. I used OpenWeatherMap and AeroDataBox APIs for this task. The weather data required getting a forecast for the next day, and flight data was collected based on airport arrivals. Weather Data from OpenWeatherMap: After signing up on OpenWeatherMap, I used the following API call to fetch the weather forecast: url = f"http://api.openweathermap.org/data/2.5/forecast?q={city\_name},{country\_c response = requests.get(url) json\_data = response.json() I then iterated through the response and extracted the necessary information, while handling possible missing data: try: dictionary['Snow'].append(json\_data['list'][0]['snow']['3h']) except KeyError: dictionary['Snow'].append('0') Flight Data from AeroDataBox: Similarly, to get flight arrivals, I used the AeroDataBox API. This API provided live flight data, which I then processed into the format needed: url = f"https://aerodatabox.p.rapidapi.com/airports/{icao\_code}/flights/arrivals response = requests.get(url, headers=headers) json\_data = response.json() As with the weather data, I had to handle missing or incomplete information and extract only the relevant fields. . . . **Storing Data in MySQL** Once I had the data, it was time to store it in a MySQL database for further analysis. I used SQLAlchemy, a Python library that simplifies database interaction. **Database Setup and Connection:** First, I set up a MySQL database with the following schema: CREATE DATABASE gans; Next, I connected to the database using **SQLAlchemy**: from sqlalchemy import create\_engine # Database connection string engine = create\_engine(f'mysql+pymysql://{user}:{password}@{host}:{port}/{databa df.to\_sql('weather', con=engine, if\_exists='append', index=False) This code uses SQLAlchemy to directly insert the cleaned data into the database. The df.to\_sql() function stores the data in the weather table. **ERR Diagram** Once you know what kind of data you want to store you need to think about how your tables are going to look like and how are you going to connect them. weather\_data v city TEXT city\_id BIGINT forecast\_time TEXT outlook TEXT temperature DOUBLE city VARCHAR(100) feels\_like DOUBLE ocuntry VARCHAR(100) wind\_speed DOUBLE rain\_prob DOUBLE cities\_city\_id INT population city\_id INT Population INT airport\_data \_\_ flights\_data city TEXT city\_id BIGINT flight\_num TEXT status TEXT icao TEXT departure\_time TEXT iata TEXT arrival\_time TEXT name TEXT timeZone TEXT shortName TEXT arrival\_icao TEXT countryCode TEXT arrival\_airport\_name TEXT timeZone TEXT airline.name TEXT location.lat DOUBLE location.lon DOUBLE . . . **Automating the Data Pipeline** To ensure that data is collected and stored daily, I set up a cron job to run the Python script every day. Since I didn't use Apache Airflow for orchestration, I leveraged AWS Lambda to schedule the job. **Challenges and Solutions** Throughout the process, I faced several challenges: • Handling messy data: Scraping HTML data often involved dealing with inconsistent structures, especially from Wikipedia. • Missing API data: Some data wasn't available, so I used try-except blocks to prevent errors during processing. • Scheduling: Automating the pipeline involved using AWS Lambda and cron jobs to ensure the script runs daily without manual intervention. **Conclusion** Building this data pipeline was an exciting journey. From scraping data off the web and making API calls to transforming and storing it in a MySQL database, I learned a lot about the practical applications of data engineering. Now, every day, my code pulls fresh data from the internet, processes it, and loads it into a database automatically, without any manual help. Feel free to leave comments, share your experiences, or check out my GitHub repository for more detailed code examples! Photo by Roman Synkevych on Unsplash Written by Haikel Zanina Edit profile O Followers · 1 Following No responses yet What are your thoughts? **Recommended from Medium Always Free** 24 GB RAM + 4 CPU + 200 GB Marendraverma2 (a) @harendra21 (b) @harendra21 Harendra in Stackademic by Abdur Rahman **Python is No More The King of Data** How I Am Using a Lifetime 100% **Free Server** Science Get a server with 24 GB RAM + 4 CPU + 200 5 Reasons Why Python is Losing Its Crown GB Storage + Always Free → Oct 23 ※ 9.4K ■ 35 → Oct 26 **3** 6.9K **1** 103 Lists Stories to Help You Level-Up Staff picks at Work 780 stories · 1493 saves 19 stories · 891 saves Self-Improvement 101 **Productivity 101** 20 stories · 3127 saves 20 stories · 2642 saves on Medium

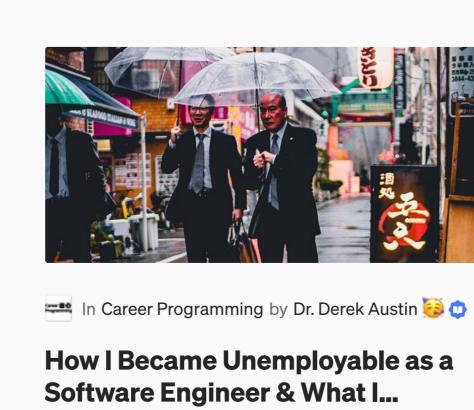


**How Old Is Your Body? Stand On** 

According to new research, the time you can

stand on one leg is the best marker of...

One Leg and Find Out



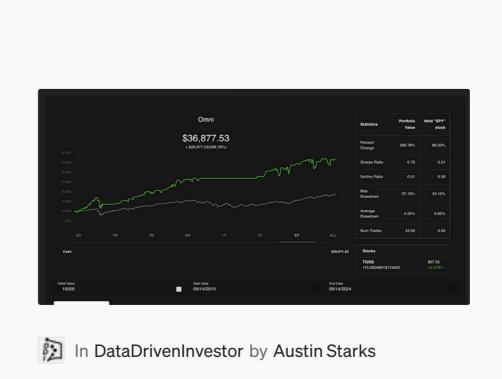
Me In The Medium Blog by Brittany Jezouit

An introduction to academic

How academics use Medium to share

research and ideas easily, connect with non-...

writing on Medium



develop a trading strategy. It is... It literally took one try. I was shocked.

I used OpenAl's o1 model to

Help Status About Careers Press Blog Privacy Terms Text to speech Teams

Being a developer is a creative, highly-paid

profession that can even be fun, but it's also...