ISSN: 2583-9888
DOI: doi.org/10.51219/JAIMLD/Pankaj-dureja/113

URF PUBLISHERS
connect with research world

Journal of Artificial Intelligence, Machine Learning and Data Science

https://urfpublishers.com/journal/artificial-intelligence

Vol: 1 & Iss: 3

Research Article

Web Scraping: Extracting and Analyzing Pre- and Post-COVID-19 Passenger Volumes

Pankaj Dureja*

Pankaj Dureja, USA

Citation: Dureja P. Web Scraping: Extracting and Analyzing Pre- and Post-COVID-19 Passenger Volumes. *J Artif Intell Mach Learn & Data Sci 2023*, 1(3), 408-410. DOI: doi.org/10.51219/JAIMLD/Pankaj-dureja/113

Received: 01 July, 2023; **Accepted:** 18 July, 2023; **Published:** 20 July, 2023

***Corresponding author:** Pankaj Dureja, USA, E-mail: Pankaj.Dureja@gmail.com

A B S T R A C T

This paper presents a solution developed for an oil and gas company to track post-COVID air travel trends and their impact on jet fuel demand. With air travel being significantly impacted during the COVID-19 pandemic, it became crucial to understand the resurgence of air travel post-pandemic to forecast jet fuel demand accurately. The Transportation Security Administration (TSA) website provides daily passenger volume data, which serves as a valuable basis for forecasting demand. However, accessing and integrating this data into the company's database posed a challenge due to the unavailability of downloadable files or API endpoints. To address this challenge, we developed an automated web scraping solution using Python, BeautifulSoup, requests, and MemSQL to retrieve, process, and load TSA passenger volume data into the company's database for analysis.

Keywords: Web Scraping, Data Extraction, BeautifulSoup, requests, HTML tables, Python, MySQL Connector

1. Introduction

In the era of big data, the internet serves as an expansive repository of valuable information across various domains. However, accessing and utilizing this data in a structured format poses a significant challenge. Web scraping emerges as a solution to this challenge, enabling automated extraction of data from websites for analysis, research, and business intelligence purposes.

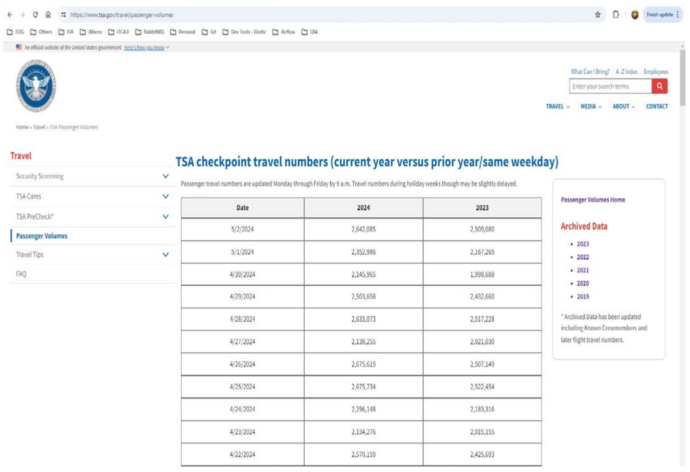
Web scraping entails the meticulous parsing of HTML and CSS elements within web pages to pinpoint and extract the desired information. However, this process encounters various hurdles. Websites often implement measures to impede or block automated scraping efforts, employing techniques such as CAPTCHAs, rate limiting, IP blocking, and other anti-scraping mechanisms. Moreover, the ethical and legal ramifications of scraping without proper authorization or in violation of terms of service loom large.

In this paper, we explore the methodologies and best practices for web scraping, drawing insights from established literature in the field. The discussion encompasses techniques for identifying and extracting relevant data using Python libraries like BeautifulSoup and requests. Additionally, we delve into advanced concepts such as HTML tables for precise data extraction and mysql connector for loading data into the database.

2. Problem Statement

Working within an oil and gas company, the impact of COVID-19 on air travel necessitated a detailed examination of post-pandemic flight trends to assess the corresponding demand for jet fuel. Upon analysis, it was discerned that the Transportation Security Administration (TSA) website offers valuable insights through its daily passenger volume data. However, accessing this data posed a challenge as it is neither downloadable nor accessible via an API endpoint, complicating the process of integrating it as structured data into the company's database. The task at hand involves retrieving and loading all available years of both current and archived TSA passenger volume data for comprehensive analysis. The TSA website's format for publishing passenger volumes can be viewed in the provided screenshot:

https://www.tsa.gov/travel/passenger-volumes



TSA checkpoint travel numbers (current year versus prior year/same weekday)

Passenger travel numbers are updated Monday through Friday by 9 a.m. Travel numbers during holiday weeks though may be slightly delayed.

Date	2024	2023
5/2/2024	2,642,085	2,509,080
5/1/2024	2,352,986	2,167,265
4/30/2024	2,145,965	1,998,688
4/29/2024	2,503,658	2,432,660
4/28/2024	2,633,073	2,517,228
4/27/2024	2,138,255	2,021,030
4/26/2024	2,675,619	2,507,149
4/25/2024	2,675,734	2,522,454
4/24/2024	2,296,148	2,183,316
4/23/2024	2,134,276	2,015,155
4/22/2024	2,570,159	2,425,693

* Archived Data has been updated including Known Crewmembers and later flight travel numbers.

3. Solution Implemented

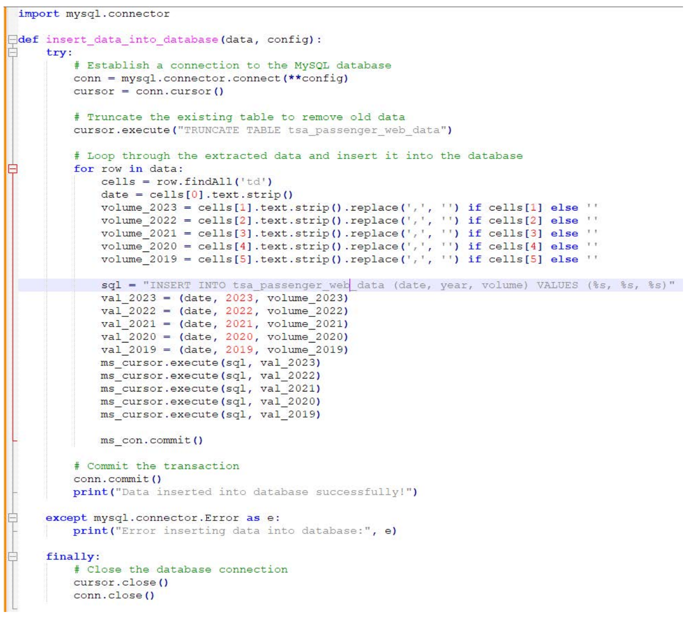
Our solution utilizes Python programming along with the BeautifulSoup and requests library to scrape the TSA website and extract passenger volume data. We employ MemSQL as the database management system to store and manage the extracted data efficiently. The solution is designed to retrieve data for all available years, including both current and archived data, ensuring comprehensive analysis.

The solution is divided into three components:

1. Web scraping module: We use Python's requests library to make HTTP requests to the TSA website and retrieve HTML content. BeautifulSoup is then employed to parse the HTML and extract relevant data from the website's tables.

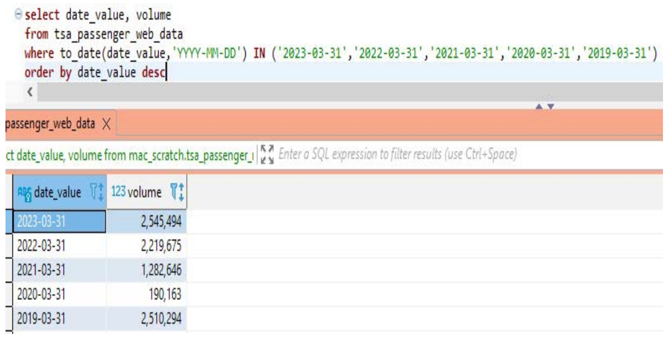
```
 containing the data
d('table', {'class': 'views-table'})

empty list to store the extracted data


able variable is not None before proceeding
None:
ble rows
.findAll('tr')

gh the rows and extract the data
ows[1:]:
row.findAll('td')
ells[0].text.strip()
023 = cells[1].text.strip().replace(',', '')
022 = cells[2].text.strip().replace(',', '')
021 = cells[3].text.strip().replace(',', '')
020 = cells[4].text.strip().replace(',', '')
019 = cells[5].text.strip().replace(',', '')

the extracted data in a dictionary
```

2. Database integration: The extracted data is stored in a MemSQL database to facilitate efficient data management and analysis. We utilize the mysql.connector library to establish a connection with the database and execute SQL queries to insert the scraped data into appropriate tables.



```
import mysql.connector

def insert_data_into_database(data, config):
    try:
        # Establish a connection to the MySQL database
        conn = mysql.connector.connect(**config)
        cursor = conn.cursor()

        # Truncate the existing table to remove old data
        cursor.execute("TRUNCATE TABLE tsa_passenger_web_data")

        # Loop through the extracted data and insert it into the database
        for row in data:
            cells = row.findAll('td')
            date = cells[0].text.strip()
            volume_2023 = cells[1].text.strip().replace(',', '') if cells[1] else ''
            volume_2022 = cells[2].text.strip().replace(',', '') if cells[2] else ''
            volume_2021 = cells[3].text.strip().replace(',', '') if cells[3] else ''
            volume_2020 = cells[4].text.strip().replace(',', '') if cells[4] else ''
            volume_2019 = cells[5].text.strip().replace(',', '') if cells[5] else ''

            sql = "INSERT INTO tsa_passenger_web_data (date, year, volume) VALUES (%s, %s, %s)"
            val_2023 = (date, 2023, volume_2023)
            val_2022 = (date, 2022, volume_2022)
            val_2021 = (date, 2021, volume_2021)
            val_2020 = (date, 2020, volume_2020)
            val_2019 = (date, 2019, volume_2019)
            ms_cursor.execute(sql, val_2023)
            ms_cursor.execute(sql, val_2022)
            ms_cursor.execute(sql, val_2021)
            ms_cursor.execute(sql, val_2020)
            ms_cursor.execute(sql, val_2019)

            ms_con.commit()

        # Commit the transaction
        conn.commit()
        print("Data inserted into database successfully!")

    except mysql.connector.Error as e:
        print("Error inserting data into database:", e)

    finally:
        # Close the database connection
        cursor.close()
        conn.close()
```

During the onset of the COVID-19 pandemic, there was a notable decline in air travel, particularly evident in March 2020. Comparatively, March 2019 saw approximately 2.6 million travelers, whereas March 2020 witnessed a stark decline of 90%, with only 190,163 individuals traveling. It took nearly two years for air travel to gradually return to pre-pandemic levels.



```
select date_value, volume
from tsa_passenger_web_data
where to_date(date_value,'YYYY-MM-DD') IN ('2023-03-31','2022-03-31','2021-03-31','2020-03-31','2019-03-31')
order by date_value desc
```

passenger_web_data
ct date_value, volume from mac_scratch.tsa_passenger_
Enter a SQL expression to filter results (use Ctrl+Space)

date_value	volume
2023-03-31	2,545,494
2022-03-31	2,219,675
2021-03-31	1,282,646
2020-03-31	190,163
2019-03-31	2,510,294

Figure 1: Database Table.

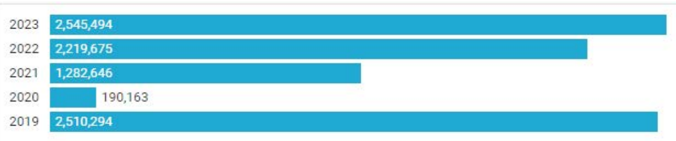


Figure 2: Bar Chart for March 31 for 5 Consecutive years from 2019 to 2023.

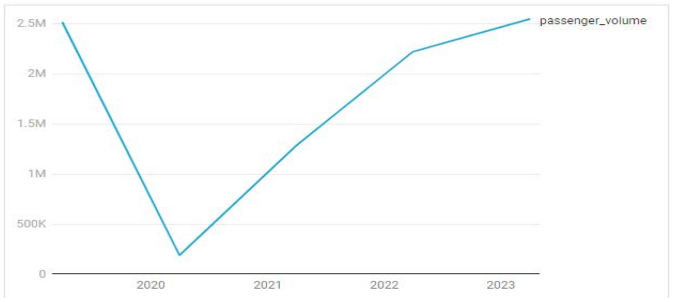
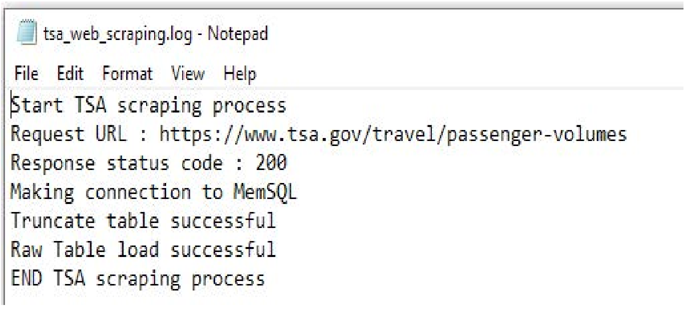


Figure 3: Line Chart for March 31 for 3 Consecutive years from 2019 to 2023.

3. Error Handling and Logging: The solution incorporates error handling mechanisms to address potential issues during data retrieval and database operations. Detailed logging is implemented to track the execution flow and capture relevant information for debugging and analysis purposes.



```
Start TSA scraping process
Request URL : https://www.tsa.gov/travel/passenger-volumes
Response status code : 200
Making connection to MemSQL
Truncate table successful
Raw Table load successful
END TSA scraping process
```

3.1. Potential Extended use cases

1. **Enhanced Market Research:** Web scraping can be leveraged to gather data on consumer trends, competitor strategies, and market dynamics, enabling businesses to make informed decisions and stay ahead of the curve.
2. **Streamlined Price Monitoring:** By scraping e-commerce websites, businesses can monitor competitor pricing, identify pricing trends, and optimize their own pricing strategies to remain competitive in the market.
3. **Efficient Content Aggregation:** Media outlets and content platforms can use web scraping to aggregate news articles, blog posts, and other relevant content from across the web, providing users with comprehensive and up-to-date information on various topics.
4. **Enhanced Academic Research:** Researchers and academics can utilize web scraping to collect data for their studies, analyze trends, and gain insights into various phenomena, contributing to advancements in knowledge and understanding across different fields.

3.2. Impact

1. **Increased efficiency:** Web scraping automates the data collection process, saving time and resources compared to manual methods.
2. **Competitive advantage:** Access to timely and accurate data enables businesses to stay competitive and adapt to market changes effectively.
3. **Better decision making:** Data-driven insights obtained through web scraping empower businesses and individuals to make informed decisions and strategies.

4. Scope

The scope of web scraping extends across industries and domains, including e-commerce, finance, healthcare, marketing, research, and more. With advancements in technology and the availability of powerful tools and libraries, the potential applications of web scraping continue to grow, offering new opportunities for innovation and growth.

5. Conclusion

The presented implementation demonstrates a robust solution for automating the extraction and storage of TSA passenger volume data for analysis. By utilizing web scraping techniques and database integration, the solution provides the oil and gas company with valuable insights into post-COVID air travel trends, facilitating informed decision-making regarding jet fuel demand forecasting and resource allocation.

6. References

1. Mitchell R. Web scraping with python. O'Reilly Media 2018; 6-10.
2. Mitchell R. Web Scraping with Python, O'Reilly Media 2018: 15-31.
3. Sweigart A. Automate the Boring Stuff with Python.
4. Mitchell R. Web Scraping with Python O'Reilly Media 2018; 88-102.
5. https://realpython.com/python-web-scraping-practical-introduction/
6. https://www.tsa.gov/travel/passenger-volumes