TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

Report

IT 360

Information Assurance and Security

Indeed Scraper

Prepared by:
Ibtihel Idoudi

Submitted to:
Dr. Manel Abdelkader

Clear explanation of the main concepts:

- **What is Web Scraping:**

Web scraping is the automated process of extracting large amounts of data from websites.
It involves using software tools or scripts to access web pages and retrieve specific unstructured data in an HTML format which is then saved into a structured format for further analysis or use.
Web scraping allows users to gather data from multiple sources on the internet quickly and efficiently.

- **Why do we use Web Scraping:**

Web Scraping has multiple applications across various industries:

-Market Research: Web scraping enables businesses to gather information about competitors, market trends, pricing strategies, and consumer sentiment by extracting data from websites, forums, and social media platforms.

-Price Monitoring: E-commerce businesses often use web scraping to monitor competitor prices, track price fluctuations, and adjust their own pricing strategies accordingly.

-Content Aggregation: Websites and news portals utilize web scraping to aggregate content from various sources across the web, providing users with comprehensive and up-to-date information on specific topics.

-Lead Generation: Sales and marketing professionals use web scraping to extract contact information such as email addresses and phone numbers from websites, helping them generate leads for their products or services.

-Social Media Analysis: Web scraping allows businesses to track brand mentions, sentiment analysis, and customer feedback on social media platforms, helping them understand their online reputation and customer perceptions.

- ## How can we do Web Scraping:

  -Online Services: Some online platforms and services provide web scraping functionalities. Users can input the URL of the website they want to scrape, and specify the data they need, and the service will scrape the website and provide the desired data in a structured format.
  -APIs (Application Programming Interfaces): Many large websites like Google, Twitter, Facebook, and Stack Overflow offer APIs that allow developers to access their data in a structured and controlled manner. APIs provide predefined methods and endpoints for developers to request specific data from the website's servers.
  -Custom Code: For more complex scraping tasks or when specific requirements cannot be met using online services or APIs, developers may choose to write custom code for web scraping. This involves writing scripts or programs using programming languages like Python, JavaScript, or Ruby.

- ## What are the components of Web Scraping:

The two main components of web scraping are:

-Web Crawler: is an artificial intelligence algorithm that is responsible for navigating the web and accessing various web pages. It systematically browses through websites, following links from one page to another to discover and content. The crawler starts from a seed URL and recursively visits linked pages, building a map of the website's structure. It retrieves HTML content from each visited page and stores it for further processing by the web scraper.

-Web Scraper: The web scraper is the component that extracts specific data from the HTML content retrieved by the crawler. It parses the HTML documents, identifies relevant data elements based on predefined rules or patterns, and extracts the desired information. This can include text, images, links, or structured data. Once the data is extracted, it may be processed, cleaned, and stored for analysis, reporting, or other purposes.

- **Functional Flow of Website Scraping:**

The functional flow of web scraping typically involves several steps:

-Input Specification: Define the target website(s) and the specific data you want to extract. This may include identifying the URLs of the web pages to scrape and determining the structure of the desired data.

-Tools Selection: Select appropriate tools and libraries for web scraping based on your programming language and requirements. Such as BeautifulSoup, Scrapy, and Selenium

-Web Crawling: The process starts with a web crawler or spider visiting the specified URLs and retrieving the HTML content of the web pages.

-HTML Parsing: Once the HTML content is retrieved, the web scraper parses the HTML documents to understand their structure and content. This involves tokenizing the HTML into elements such as tags, attributes, and text content, and building a hierarchical representation known as the Document Object Model (DOM).

-Data Extraction: The scraper identifies and extracts the specific data elements of interest from the parsed HTML documents. This can include text, images, links, or structured data embedded within the HTML structure. Techniques such as XPath, CSS selectors, or regular expressions may be used to locate and extract the data accurately.

-Data Processing: The extracted data may undergo further processing, cleaning, or transformation to prepare it for analysis or storage. This could involve tasks such as removing HTML tags, filtering out irrelevant information, or converting data into a standardized format.

-Output Generation: Finally, the scraped data is typically stored or exported in a usable format such as a database, CSV file, or spreadsheet.

It may also be integrated with other systems or applications for further analysis, reporting, or visualization.

- **Is Web Scraping ethical:**

Web scraping can be considered ethical when performed responsibly, adhering to legal guidelines and ethical standards. Ethical web scraping involves obtaining data with proper consent, respecting website policies and terms of service, and ensuring that the scraping process does not cause harm or infringe upon privacy rights. Conversely, unethical web scraping occurs when it violates these principles, such as through unauthorized access, excessive requests that strain server resources, or misuse of extracted data for malicious purposes. Therefore, the ethicality of web scraping depends on factors such as compliance with regulations, respect for website owners' rights, and consideration for the privacy and well-being of individuals.

Overview of the main existing solutions

- **Do Not Require Coding:**

1. **Cloud-Based Scraping Platforms:**
    - Characteristics:

- Cloud web scrapers are hosted solutions that run in cloud environments, offering scalability, reliability, and accessibility.
- These scrapers allow users to deploy and manage scraping tasks remotely, without the need for local infrastructure or resources.
- They often provide features such as distributed scraping, proxy management, and data storage in the cloud.
- Examples include Apify, ScrapingBee, and DataHen.

    - Advantages:

- Scalability to handle large-scale scraping tasks.
- Accessibility from anywhere with an internet connection.
- Built-in features for proxy management, scheduling, and data storage.

    - Limitations:

- Dependency on the service provider's infrastructure and reliability.
- Potential cost implications for large-scale or long-term usage.

2. **Browser Extensions:**
   - Characteristics:

- Browser extensions are small software programs that extend the functionality of web browsers.
- Web scraping browser extensions are specifically designed to extract data from web pages directly within the browser.
- They typically provide features for selecting and extracting data elements, saving extracted data, and exporting it to various formats.
- Examples include Web Scraper, Data Miner, and Scraper.
  - Advantages:
- Seamless integration with web browsers for convenient data extraction.
- There is no need to install additional software or set up separate environments.
- Instant access to scraped data directly within the browser.
  - Limitations:
- Limited functionality compared to standalone scraping tools or platforms.
- It may have performance limitations for scraping large amounts of data or complex websites.

3. **API-Based Solutions:**
   - Characteristics:

- APIs and data services provide structured access to data from web sources.
- They offer endpoints for retrieving specific data sets or performing actions on data.
- Can return data in various structured formats including JSON, XML, or CSV.
  - Advantages:
- Provides sanctioned access to data without the need for scraping.
- Data is typically well-organized and easier to work with.
- APIs often have rate limits but can be more lenient compared to direct scraping
- Allow developers to focus on data consumption rather than data extraction and parsing.
  - Limitations:
- Not all websites offer APIs, and those that do may restrict access to certain endpoints or data.
- Some APIs may have usage-based pricing models, which can become expensive for large-scale data extraction.
- APIs may not provide access to all the data available on a website.

4. **Visual Scraping Tools:**
   - Characteristics:

- Offer an intuitive point-and-click interface that allows users to interact with web pages and select the data elements they want to extract.
- Supports drag-and-drop functionality for easy rearranging and resizing of extraction fields.
- Offer template creation features, allowing users to define reusable extraction templates for similar types of web pages or websites.
  - <u>Advantages:</u>
- User-friendly interface, suitable for non-programmers.
- Quick setup and deployment.
- Can handle basic scraping tasks efficiently without coding knowledge.
- Allow developers to focus on data consumption rather than data extraction and parsing.
  - <u>Limitations</u>:
- Limited customization compared to code-based solutions.
- May struggle with complex scraping scenarios.
- Scalability can be an issue for large-scale tasks.

- **<u>Require Coding:</u>**

1. **Code-Based Libraries/Frameworks:**
   - <u>Characteristics</u>:
- They are typically written in programming languages such as Python, JavaScript, or Ruby, and they rely on web scraping libraries such as BeautifulSoup or Scrapy.
- Allow developers to parse HTML and XML documents, navigate the DOM (Document Object Model), and extract data.
  - <u>Advantages:</u>
- Developers have full control over the scraping process.
- Scraping logic can be tailored to specific website structures and data requirements.
- These libraries have large communities, offering support and resources for large-scale data extraction.
  - <u>Limitations</u>:
- Requires knowledge of programming and web technologies.
- Websites frequently change their structure, requiring regular updates to scraping scripts.
- Scraping too aggressively may lead to being blocked by websites.

2. **Browser Automation Tools:**
   - <u>Characteristics</u>:

- The process of automating interactions with web browsers programmatically
- Capable of handling JavaScript-rendered content and interacting with dynamic elements.
- It involves controlling a web browser, such as Google Chrome or Mozilla Firefox, through scripting or code to perform various tasks, such as navigating to web pages, filling out forms, clicking buttons, and extracting data.
  - Advantages:
- Can be used for a wide range of scraping tasks, from simple data extraction to complex interaction-based scraping scenarios.
- Enable interactive scraping tasks where users can simulate human interactions
- Users do not need to manually parse JavaScript code to extract data
  - Limitations:
- Browser automation scripts may face compatibility issues with different browser versions or web page layouts, requiring updates or adjustments to maintain functionality.
- Slower performance compared to direct HTTP requests.

3. **Command line tools:**
  - Characteristics:
- Operated via a command-line interface.
- Designed for simple scraping tasks and
- Often offers options for recursive downloading, allowing users to scrape multiple pages or entire websites.
  - Advantages:
- Lightweight and consume fewer system resources compared to browser-based solutions
- Easy to integrate into scripts or workflows.
- Suitable for downloading files or static content, such as images, documents, or other media.
  - Limitations:
- Limited parsing capabilities.
- Not suitable for extracting structured data from HTML.
- Lack of features for handling dynamic content or JavaScript-rendered elements

High-Level Design of the Proposed Solution

In our project, we have opted to concentrate on extracting data from the ***Indeed*** website. This choice stems from Indeed's extensive array of job listings and its widespread recognition among job seekers and employers alike. Indeed serves as an invaluable source of real-time job market data, facilitating the extraction of insights into prevailing job trends, salary differentials, and skill demands. By focusing our efforts on Indeed, we aim to provide valuable insights to support job seekers from the fields of consultancy, engineering and analysis.

- ## ***Design:***

The solution involves a combination of code-based web scraping techniques and data processing methods to extract and analyze job listing data from the Indeed website.

- ## ***Components:***

  Data Extraction Component: Responsible for retrieving raw data, such as job listings and company information, from Indeed's website using web scraping techniques.

  Data Processing Component: Cleans, filters, and transforms the raw data extracted from Indeed to ensure its quality and usability for further analysis.

  Data Storage Component: Stores the processed data in a structured format, such as a database or file system, for easy retrieval and future use.

  Data Analysis Component: Analyzes the stored data to derive valuable insights such as; job market trends, salary variations, and skill demands, facilitating informed decision-making for users.

- ## ***Users:***

1. website Administrator: an individual or entity responsible for overseeing a website's operation, maintenance, and security.

2. User/Scraper: an individual designed to extract data from websites, typically for analysis or archival purposes.

- ***Exchanged Messages/Data:***

  Input Data:

- Search criteria (job title, location, keywords)
- Configuration parameters (scraping settings, data processing rules)

  Output Data:

- Extracted job data (title, company, location, salary, etc.)
- Processed job data (cleaned, standardized, and structured)

  Messages:

- Progress updates (e.g., scraping page X of Y) for long-running tasks.

- ***Functions:***

  1. website Administrator:

- **Monitoring Scraping Activity**: The administrator monitors the scraping activity on the website to ensure it complies with the website's terms of service and legal requirements.
- **Managing Access Permissions:** They may manage access permissions for users who perform scraping activities on the website, ensuring that only authorized individuals can access certain features or data.
- **Handling Scraping Requests:** They handle requests from users or automated bots that perform scraping activities on the website, providing guidance or assistance as needed.

  2. User/Scraper:

- **Specifying Scraping Parameters:** Define the parameters for the scraping process, such as scraping criteria (job titles, locations, salaries...), target URL, target data
- **Initiating Scraping Process**: Initiate the web scraping process to extract job listings and relevant information from the Indeed website.
- **Extracting Job Data**: Use scraping techniques to extract job data from Indeed's website, including job titles, company names, locations, and salaries.

- **Processing Extracted Data:** Perform basic data processing tasks, such as cleaning and formatting the extracted data to ensure its quality and usability.
- **Storing Scraped Data:** Save the extracted job data in a structured format, such as a database or file system, for further analysis and use.
- **Analyze Scraped Data:** Analyzes the stored data to derive valuable insights such as; job market trends, salary variations, and skill demands

- ***Tools:***

1. Programming language: Python is a popular choice for web scraping due to its simplicity and the wide range of libraries available

2. Web scraping libraries and frameworks:

- **Requests:** A Python library for making HTTP requests to web servers. It's used in the project to fetch the HTML content of Indeed's web pages.

- **BeautifulSoup:** A Python library for parsing HTML and extracting data from it. It helps isolate job listings, salaries, and other information from Indeed's HTML pages.

- **Selenium:** An automation tool for web browsers. It's handy for scraping dynamic content on Indeed.

- **Time:** provides various functions to work with time-related tasks such as pausing the execution of a program

- **Matplotlib:** is a comprehensive Python library for creating static, animated, and interactive visualizations.

- **Seaborn:** is a powerful and user-friendly data visualization library built on top of matplotlib.

- **Pandas:** For storing extracted data in a structured format like data frames and data manipulation and analysis.

- ***<u>Working Flow:</u>***

1. Planning and Scope Definition

- **Goal**: The primary goal is to extract job listings from Indeed, specifically for the "consultant" “engineer” “analyst” job titles.
- **Scope**: The scope includes extracting details such as job titles, company names, locations, and dates posted across multiple pages on the Indeed website.

2. Research and Preparation

- **Understand the HTML structure**: Before writing the script, one would need to inspect Indeed's HTML structure to identify the tags and classes containing the necessary data.
- **Choose tools**: This script uses Selenium for browser automation and BeautifulSoup for parsing HTML.

3. Access and Parsing

Set up Chrome options to customize browser behavior

```
options = webdriver.ChromeOptions()
options.add_argument('user-agent=Mozilla/)
driver = webdriver.Chrome(options=options)
```

Purpose: Customizes the browser to mimic a real user by setting the user-agent, which can help avoid detection and blocking by the website.

Open the initial page

```
url = ”
driver.get(url)
```

Purpose: Opens the initial page containing job listings for "consultant" roles.

Wait for the page to load

```
sleep(5)
```

Purpose: Ensures the page fully loads before attempting to scrape data, which is crucial when dealing with dynamic content.

Parse the initial HTML

```
html = driver.page_source
soup = BeautifulSoup(html, 'html.parser')
```

Purpose: Parses the HTML of the loaded page to make it accessible for data extraction.

4. Setting up the scraping process

Define the get_data function

```
def get_data(job_listing):
  title = job_listing.find("a").find("span").text.strip()
  try:
    company = job_listing.find('span', class_="css-92r8pb eu4oa1w0").text.strip()
  except AttributeError:
    company = ''
  return ('consultant', title, company, location, date_posted)
```

Purpose: Extracts specific details from each job listing, handling missing values gracefully with try-except blocks.

5. Data Extraction

Loop to scrape data from multiple pages

```
records = []
while True:
  try:
    url = 'https://ng.indeed.com/' + soup.find('a', {'aria-label':'Next Page'}).get('href')
  except AttributeError:
    break
  driver.get(url)
  html = driver.page_source
  soup = BeautifulSoup(html, 'html.parser')
  job_listings = soup.find_all('div', class_='job_seen_beacon')
  for job_listing in job_listings:
    record = get_data(job_listing)
    records.append(record)
```

Purpose: Continuously navigate through the pages by finding the "Next Page" link, scraping data from each page until no more pages are found.

Close the browser

```
driver.quit()
```

Purpose: Closes the WebDriver instance to free up resources once the scraping is complete.

6. Data Storage

Convert list of records into a DataFrame

```
df = pd.DataFrame(records, columns=['ID', 'Title', 'Company', 'Location', 'Date Posted'])
```

Purpose: Converts the scraped data into a pandas DataFrame for easier manipulation and analysis.

Save DataFrame to a CSV file

```
df.to_csv('consulting_job.csv', index=False)
print("Data saved to consulting_job.csv")
```

Purpose: Saves the DataFrame to a CSV file for persistent storage and further analysis.

7. Data Processing and Storing

```
filtered_engineer=engineer[engineer['Title'].str.contains('engineer', case=False)]
```

Purpose: Filtering data based on the term “engineer” and saving the filtered data into the csv file

```
missing_values=filtered_engineer.isnull().sum()
```

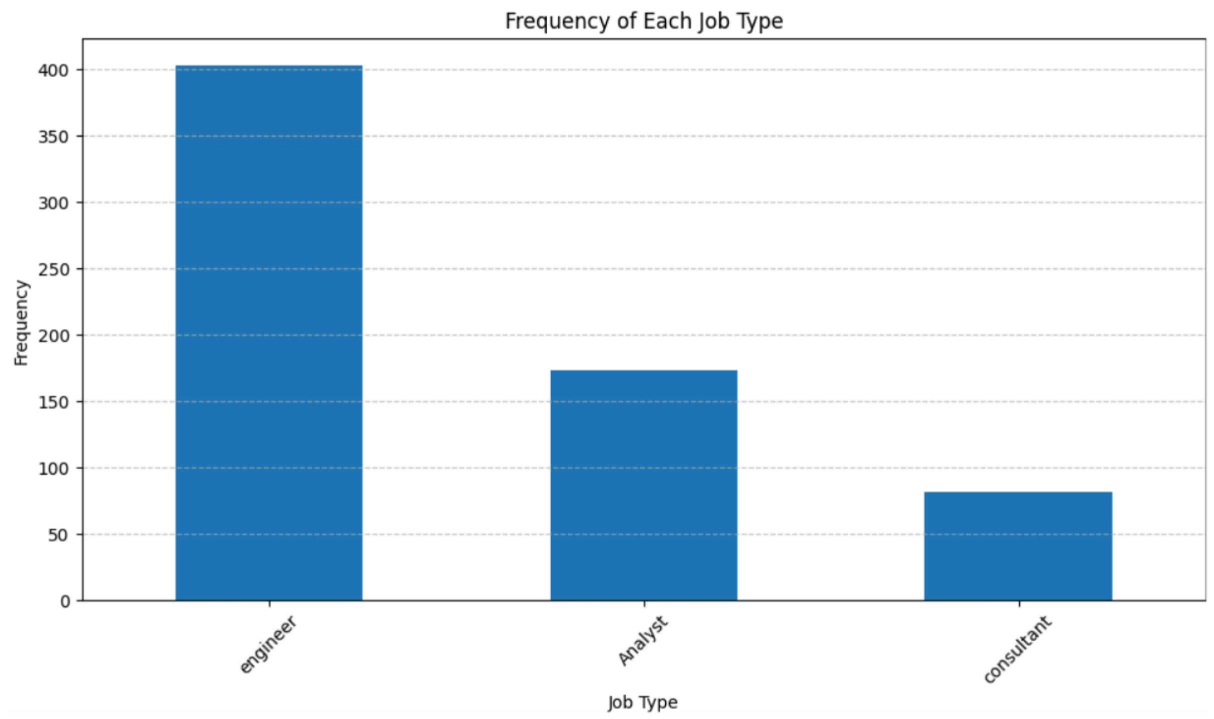
```
filtered_analyst=filtered_analyst.dropna()
```

Purpose: Identify and remove the missing values in the data frame ‘filtered_engineer”

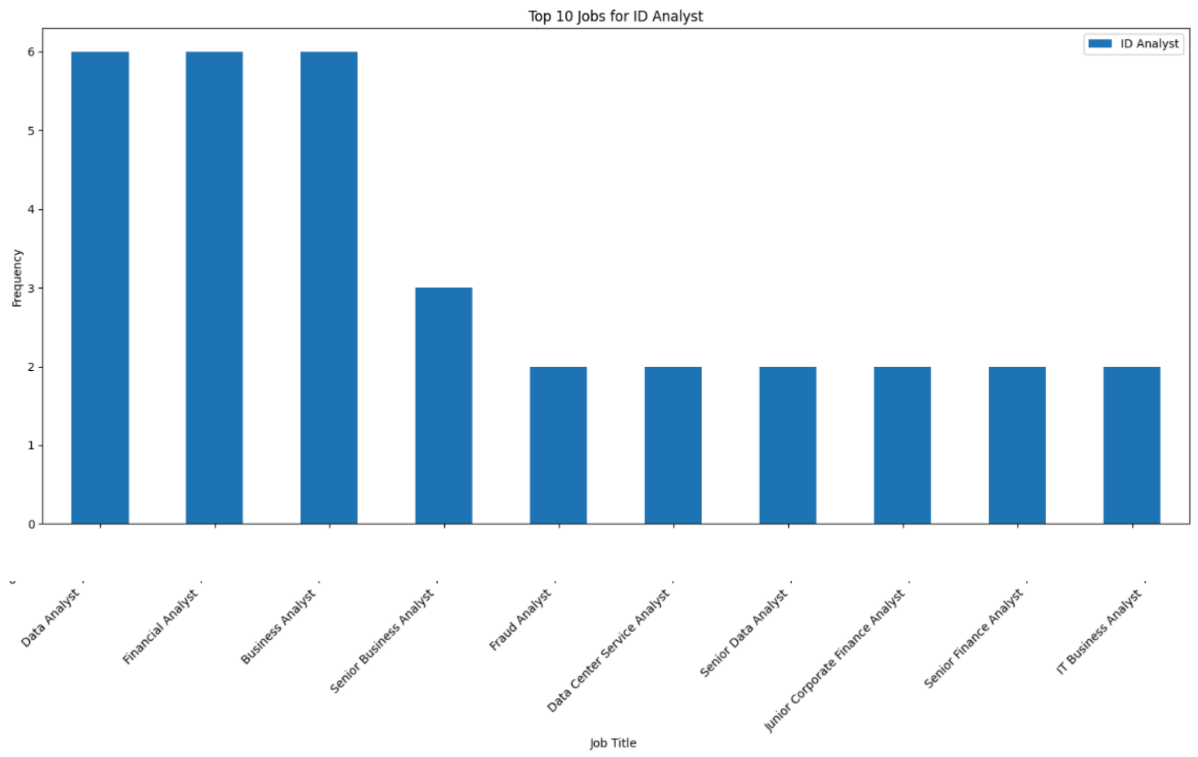
```
filtered_engineer.to_csv("engineer_job.csv" , index=False)
```

Purpose: Store the filtered data int the CSV file

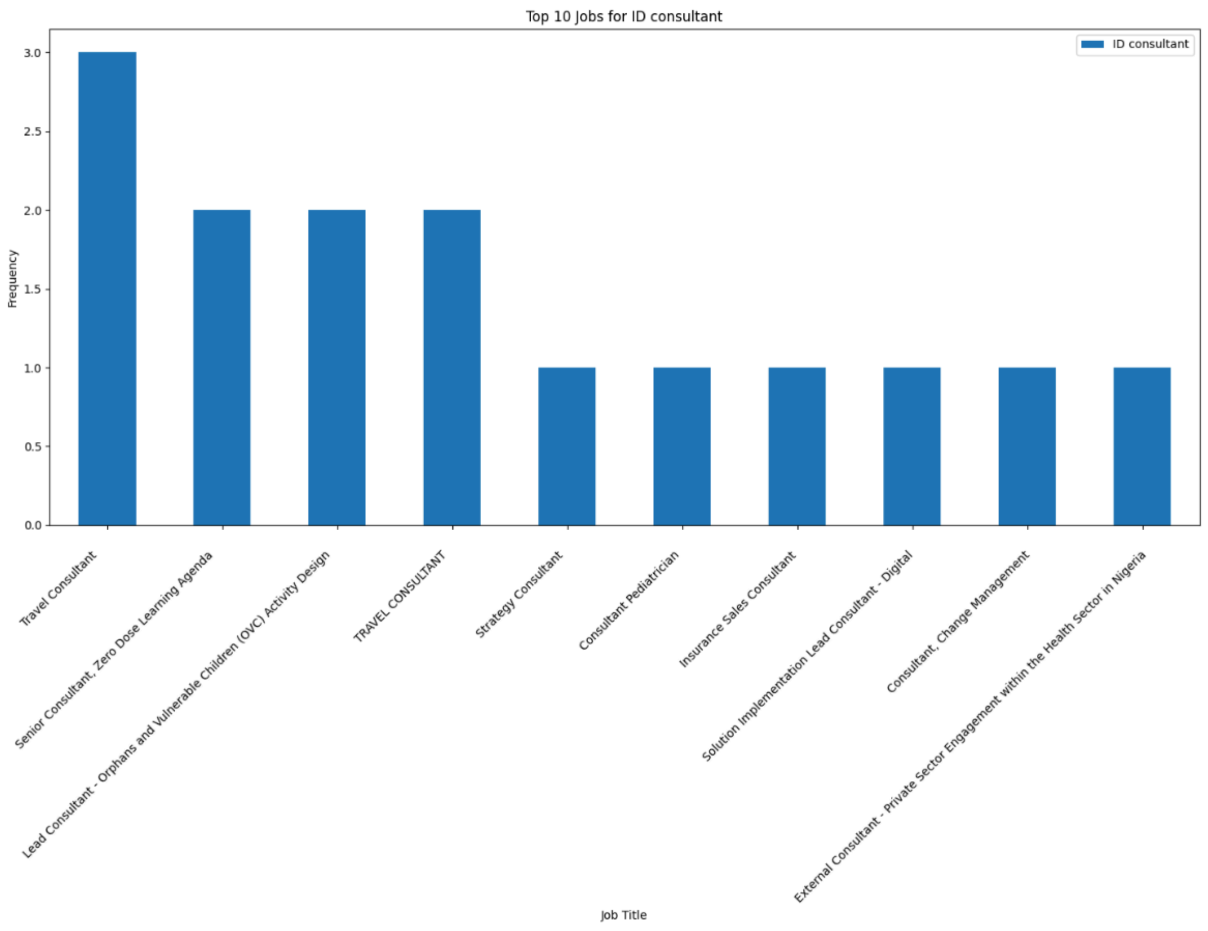
8. Data Analysis



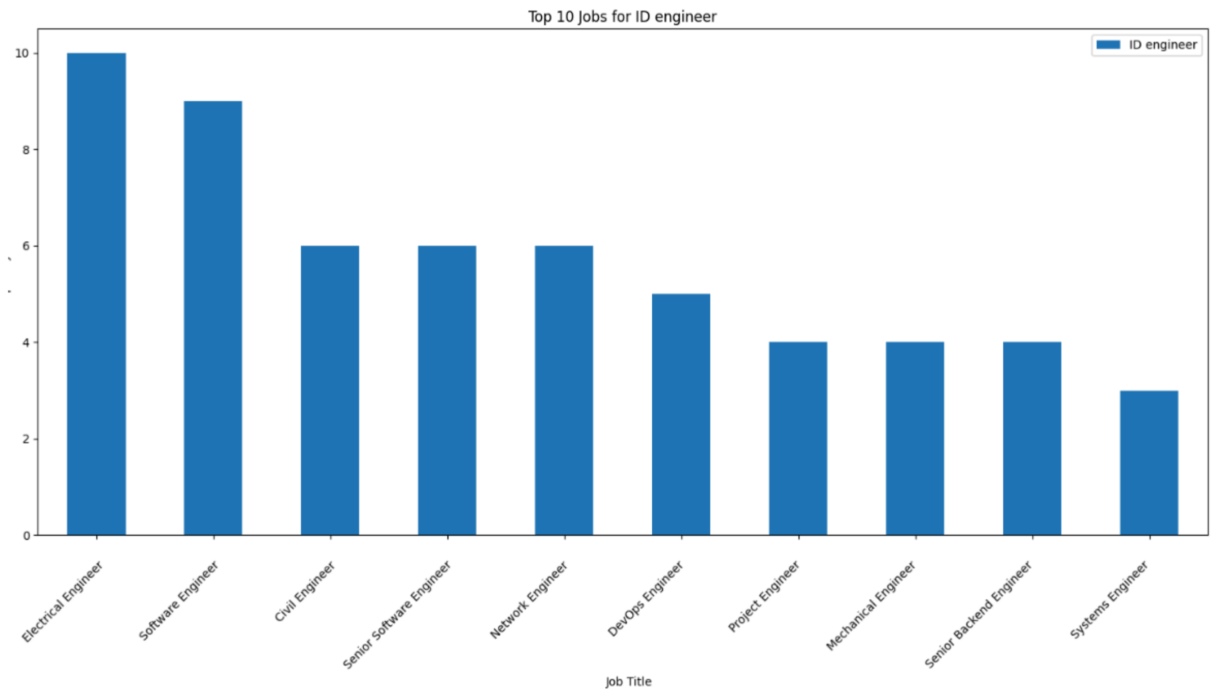
The graph indicates a high demand for engineers. Analysts follow as the second most frequent job type Consultants are the third most sought-after.



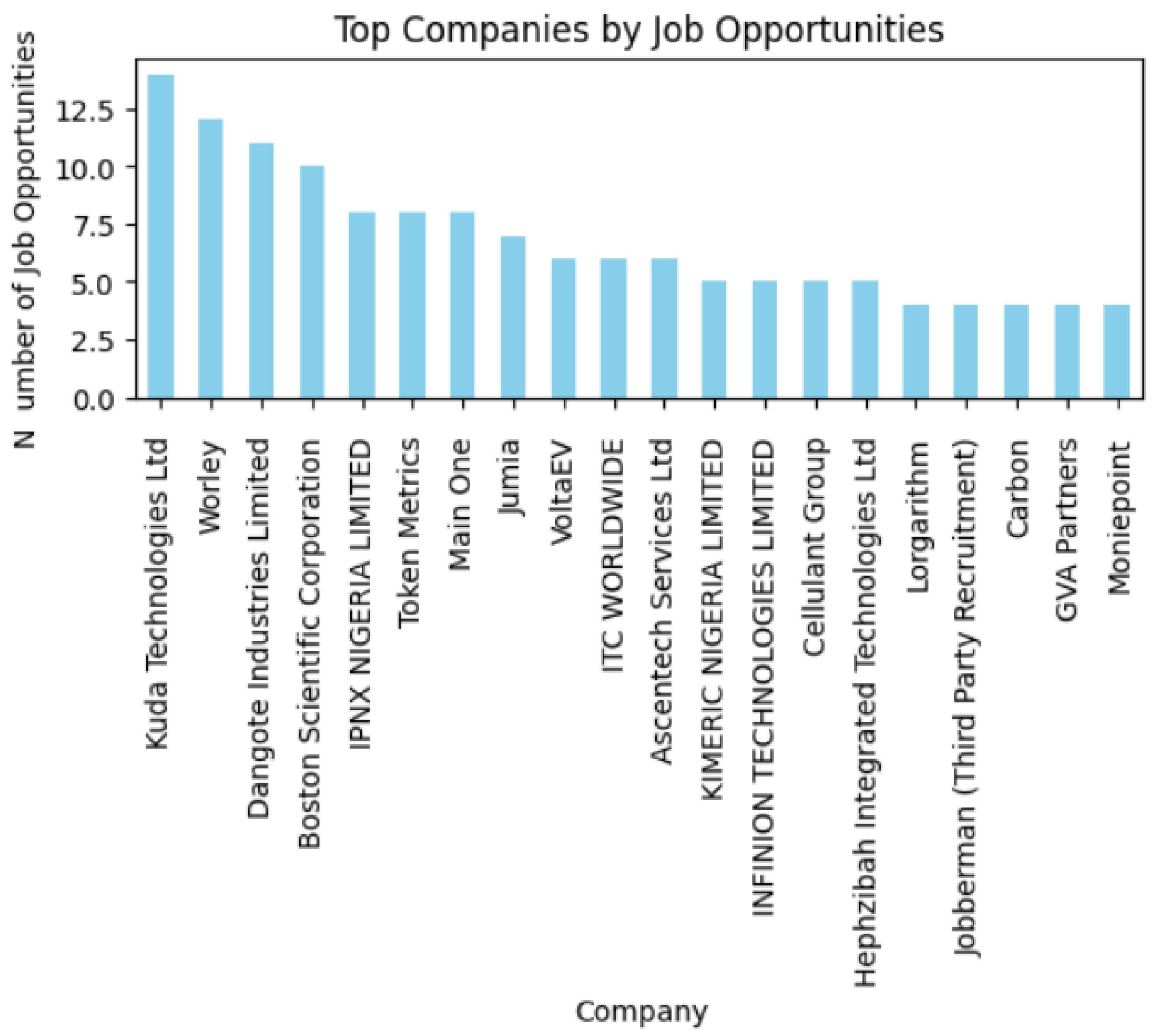
The graph shows that in the analysts' category, most of the recruiters are looking for business analysts, data analysts and financial analysts.



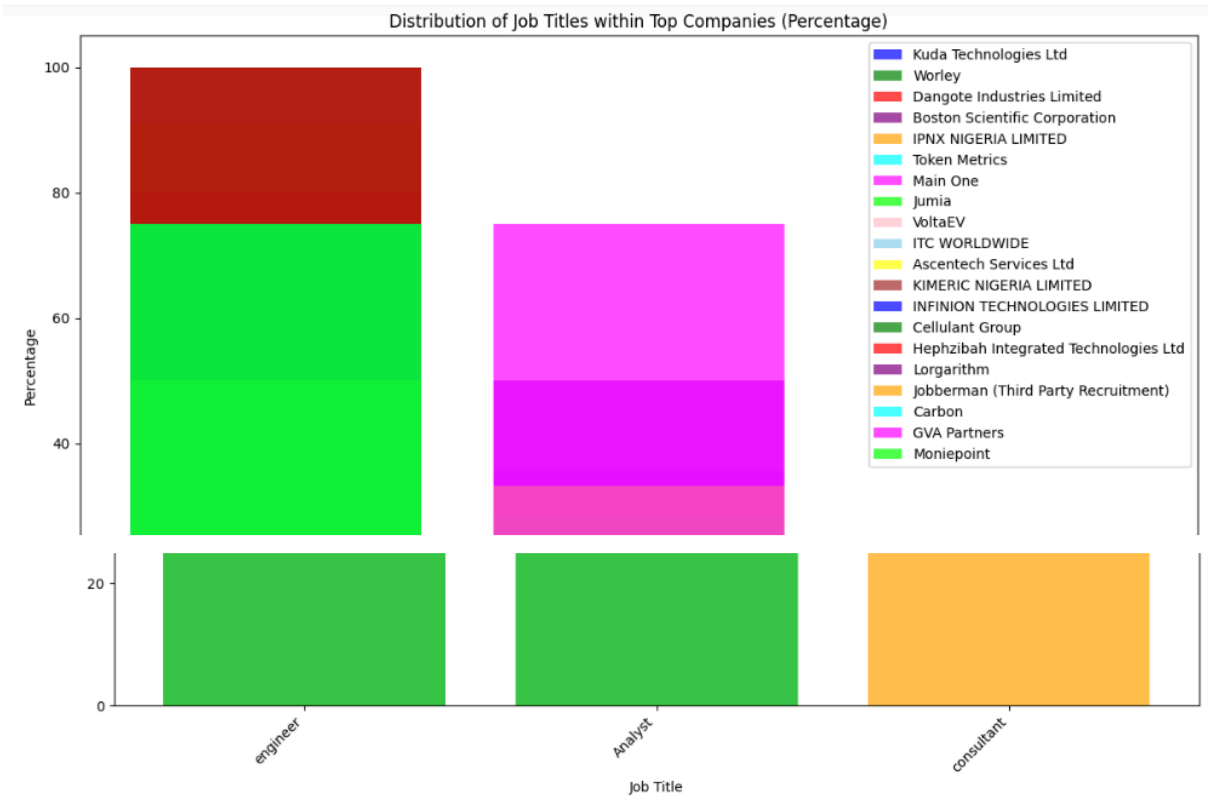
The graph demonstrates that the most demanded job in the consultancy field is the travel consultant.



The graph shows that in the engineers' category, most of the recruiters are looking for electrical and software engineers.

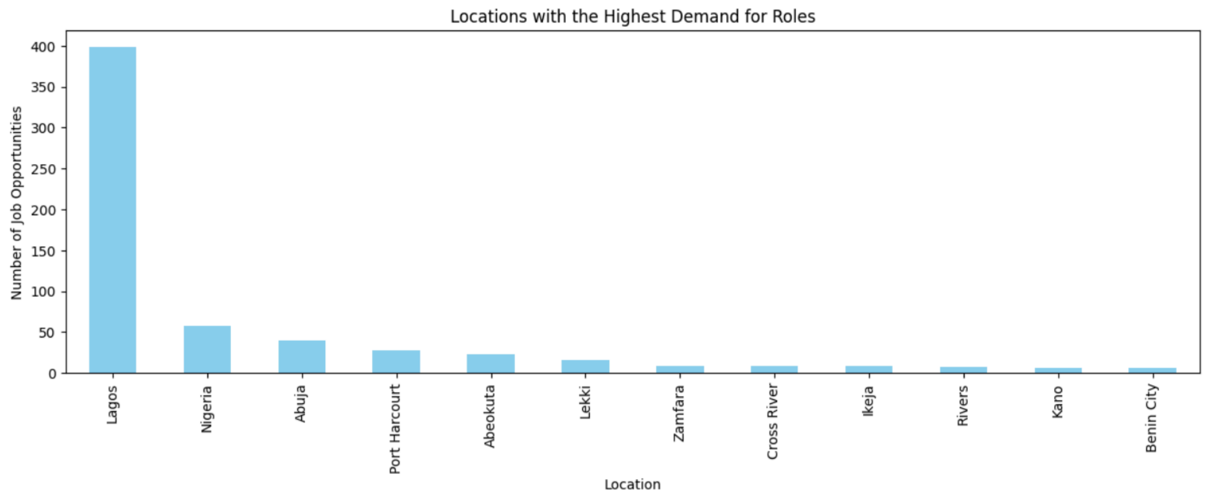


The graph illustrates that Worley and Kuda Technologies Ltd are the companies with the highest number of job opportunities



The graph shows that:

- The majority of engineers' job offers come from cellulant group, jumia, Hephizibah technologies..
- The distribution of the job offers for the analysts is evenly spread across multiple companies such as cellulant group, main one, etc
- All of the consultants' job offers come from Jobberman company.



The graph reveals that Lagos is the city with the highest demand for these jobs