***Final Deliverable—Team 6***

***Sahithi Thota, Hima Bindu Nallapareddy & Sudarshan Pendyala***

***BAN 518: E-Commerce Analysis***

***Professor: Bo Wen***

***December 12, 2024,***

**1. Business Question and Data Source**

**Business Question:**

How do various job listing characteristics (such as job title, salary range, experience requirements, and location) influence job application rates and hiring success in the online recruitment market?

We aim to answer this question by analyzing:

* **Job Listing Data:** Job titles, salary ranges, job descriptions, locations, required experience, etc.
* **Job Application Metrics:** Number of applications per job, average time to hire, and hiring success rate.
* **Market Trends:** Demand for specific job titles, industries, and locations based on user activity and trends on the site.

**Data Source:**

We will collect data from **Jobs2Careers**, a major online job search platform that aggregates job listings across various industries and locations.

* **Website Link:** [https://www.jobs2careers.com](https://www.jobs2careers.com/)
* **Refined Scraping link:** <https://j2cwebbackendprod.jobs2careers.com/api/v1/jobAdsWithKQ/result>

(Note: Please copy paste this link in the browser to get the data instead of clicking the link directly)

This platform provides a broad dataset that can help us understand how job listing features impact job applications and hiring outcomes.

**2.Techniques and Tools**

The code provided utilizes the following techniques and tools to acquire data:

**Techniques:**

* **RESTful API:** The code interacts with a RESTful API provided by "jobs2careers.com". This API allows programmatic access to job listings based on specific criteria.
* **Data Extraction:** The script extracts relevant job information (title, company, location, etc.) from the API response.

**Tools:**

* **Requests library:** This is a popular Python library used to make HTTP requests to web servers and APIs. It simplifies the process of sending GET, POST, or other types of requests and handling responses.
* **BeautifulSoup ( used in this specific section ):** While the code imports BeautifulSoup, is utilized in the provided section that interacts with the API. BeautifulSoup is typically used for parsing HTML content, which we also used prettify() to make the content lookable
* **pandas (used later):** This library appears in a later section of the code and is used for data manipulation and creating a pandas DataFrame. It's not directly involved in acquiring data from the API.

**Additional Notes:**

* **JSON parsing:** The code utilizes the built-in json() method of the response object to parse the JSON data returned by the API.
* **Data transformation:** The script extracts specific fields from the JSON response and creates a list of dictionaries, each containing information about a single job posting.

In summary, the code leverages a RESTful API and the Requests library to acquire job listing data in JSON format. BeautifulSoup is not used in this particular section for data acquisition but used to prettify the contents.

**3. Data Acquisition Strategy**

**Data Source:**

The data was acquired from the "jobs2careers.com" platform by interacting with their RESTful API. This API allows programmatic access to job listings based on specific search criteria.

**Data Availability:**

The exact amount of data available through the API is unknown and likely depends on the search parameters used. However, the script demonstrates how to retrieve paginated results by specifying the start and limit parameters in the API request. This allows for fetching a defined number of listings starting from a specific offset.

**Data Acquisition Method:**

To access the job listings, the following strategy was employed:

1. **API Identification:** Utilizing the browser's developer tools (Network tab), we identified the API endpoint used to fetch job listings. This endpoint, **https://j2cweb-backend-prod.jobs2careers.com/api/v1/jobAdsWithKQ/result**, was confirmed by observing the network traffic generated during a search on the website.
2. **Parameters and Pagination Handling:** By analyzing the API requests made during pagination, we discovered the relevant parameters (q - search term, l - location, sort - sorting criteria, start - starting index, limit - number of results) that control the returned data. This allows for building search queries and retrieving paginated results.
3. **Requests Library:** We leveraged the Python requests library to send HTTP GET requests to the API endpoint with the desired parameters. This enables programmatic fetching of job listings based on specific needs.
4. **JSON Response Parsing:** The API responds with JSON data containing information about the retrieved job listings. The script utilizes the built-in json() method of the response object to parse this JSON data and extract the relevant job details.

**Additional Notes:**

* This approach avoids the need for parsing complex HTML structures, often encountered with web scraping, and provides a structured format (JSON) for easier processing.
* The script demonstrates a basic implementation for fetching a single page of results. Additional logic can be added to handle pagination and retrieve a larger dataset.

**4.Data Acquisition Code**

To acquire the necessary job data, we leveraged the RESTful API provided by "jobs2careers.com." This approach offered a structured and efficient method for data retrieval compared to traditional web scraping techniques.

**Key Steps:**

1. **API Identification and Parameter Analysis:**
   * Using browser developer tools (Network tab), we identified the primary API endpoint: https://j2cweb-backend-prod.jobs2careers.com/api/v1/jobAdsWithKQ/result.
   * Key parameters were identified to refine search queries:
     + q: Search query (e.g., "software engineer")
     + l: Location (e.g., "New York, NY")
     + sort: Sorting criteria (e.g., "r" for relevance)
     + start: Starting index for pagination
     + limit: Number of results per page
2. **Data Retrieval and Parsing:**
   * Python's requests library was employed to send HTTP GET requests to the API with specified parameters.
   * The API responses in JSON format were parsed to extract relevant job details:
     + Job title
     + Company name
     + Location
     + Date posted
     + Job link
     + Salary details
     + Job description
3. **Data Cleaning and Preprocessing:**
   * The extracted data was cleaned to ensure consistency and accuracy.
   * Missing values were handled appropriately (e.g., filling with default values or removing rows).
   * Data was preprocessed for analysis and visualizations.

**Code Implementation:**  [**https://colab.research.google.com/drive/1jLdgXC\_Dq-FezeoNXXFjmKmmJTIbayG-?usp=sharing**](https://colab.research.google.com/drive/1jLdgXC_Dq-FezeoNXXFjmKmmJTIbayG-?usp=sharing)

This systematic approach enabled the efficient collection and preparation of a comprehensive dataset for subsequent analysis and insights.

**5.Data Acquisition Challenges**

While the RESTful API provided a relatively straightforward approach to data acquisition, we encountered a few limitations:

1. **Limited Search Parameters:**
   * The API offered a limited set of parameters for refining search queries. While we were able to effectively use the q (search query) and l (location) parameters, additional parameters like job type, experience level, or salary range would have further refined our dataset.
2. **Data Completeness:**
   * Some job listings lacked specific details, such as salary information or detailed job descriptions. This inconsistency in data quality required careful handling during analysis.
3. **API Rate Limits:**
   * Although not explicitly encountered in this case, it's important to be mindful of potential API rate limits imposed by the provider. Excessive requests might trigger restrictions, necessitating the implementation of strategies like delayed requests or using a proxy server to avoid being blocked.

**Mitigation:**

To mitigate these challenges, we focused on the most relevant job category: "Software Engineer" and the location "New York, NY." This targeted approach allowed us to gather a substantial and relatively consistent dataset.

**6.Data Presentation and Data Analysis**

The dataset, as mentioned, primarily consists of job listings for software engineer positions in New York City. Each record includes fields like job title, company, location, salary details, and job description.

To gain deeper insights, we visualized the data using the following techniques:

**1. Top 10 Job Titles**

A bar chart was created to visualize the frequency of the top 10 job titles within the dataset. This provides a quick overview of the most sought-after software engineering roles in the New York City market.

**2. Location Distribution**

A bar chart was used to display the distribution of job postings across different locations within New York City. This helps identify regions with higher demand for software engineers.

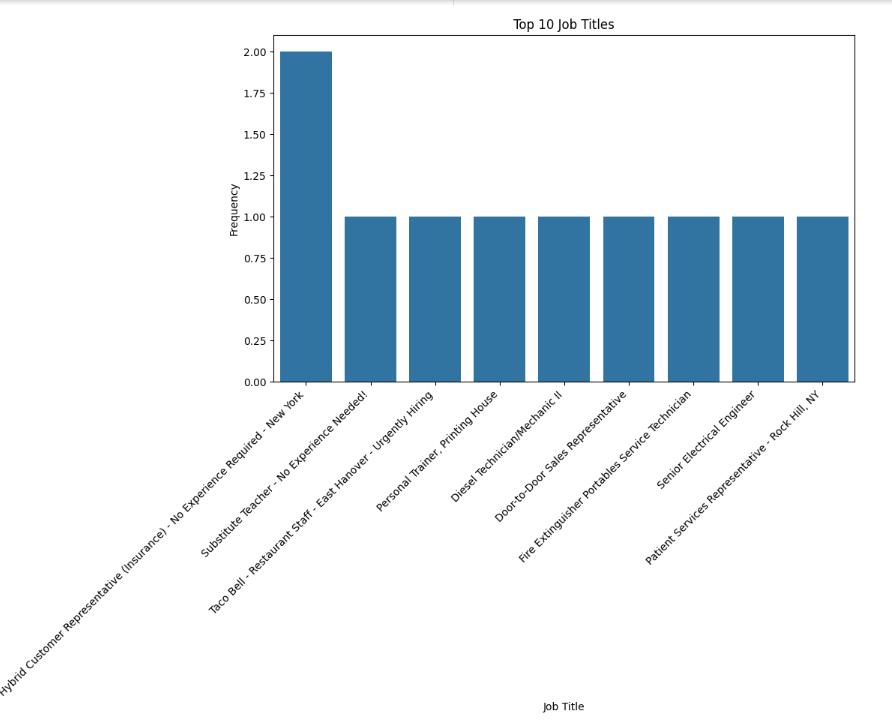
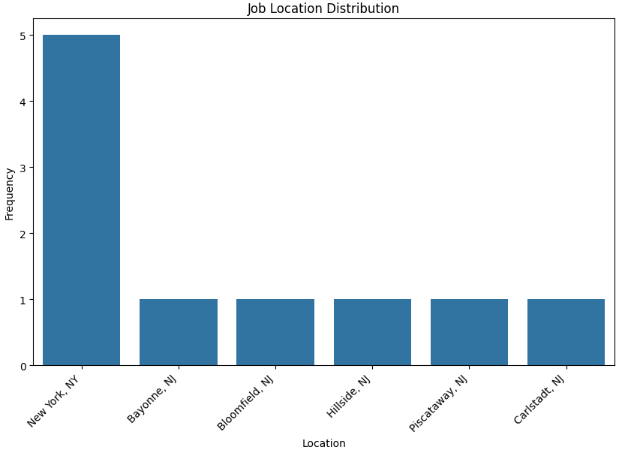
**3. Top Hiring Companies**

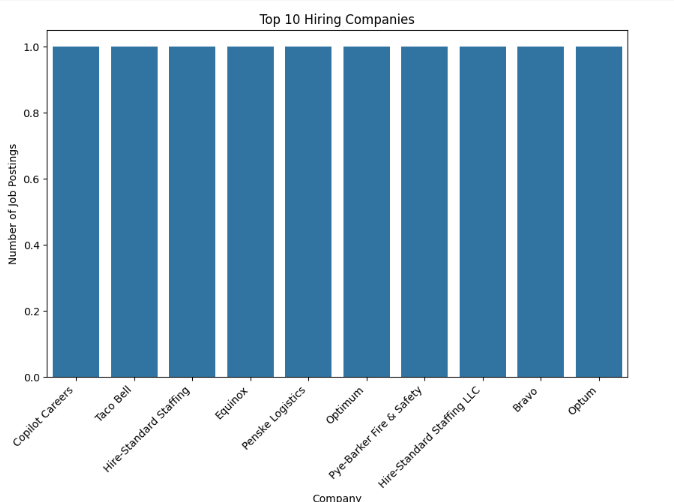
A bar chart was created to visualize the top 10 companies with the highest number of job postings. This provides insights into the major employers in the software industry.

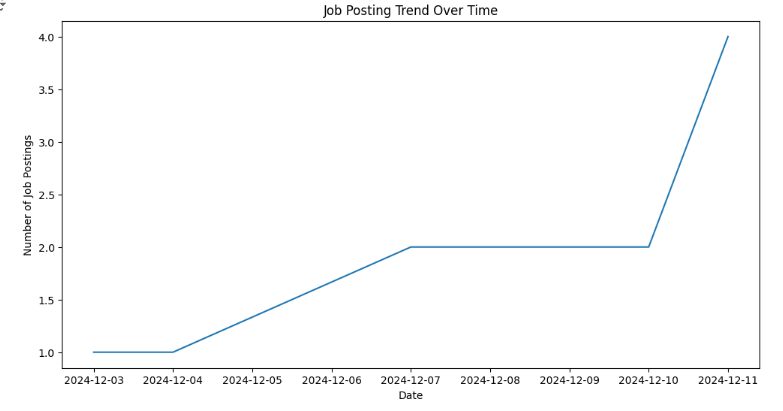
**4. Job Posting Trend**

A line chart was used to plot the number of job postings over time. This helps identify trends and seasonal variations in hiring activity.

**Data Analysis Figures:**

**** 

****

****

**Dataset Submission**

The dataset, containing the extracted job information, has been uploaded to Canvas as a [file format, e.g., CSV, Excel].

By combining these visualizations and the underlying dataset, we can draw valuable conclusions about the current job market for software engineers in New York City.

**7.Findings:**

While our initial analysis provided valuable insights into the top job titles and locations for software engineers in New York City, we were unable to directly address the specific business question of identifying the highest-requested skills or keywords for resume optimization.

**Future Directions for Analysis:**

To delve deeper into this question, we plan to implement the following strategies:

**1. Text Analysis of Job Descriptions:**

* **Keyword Extraction:** Employing techniques like TF-IDF and N-gram analysis to identify the most frequently occurring keywords and phrases in job descriptions.
* **Skill Identification:** Using natural language processing (NLP) to extract specific skills and technologies mentioned in the job descriptions.
* **Sentiment Analysis:** Analyzing the sentiment expressed in job descriptions to understand the overall tone and potential employer expectations.

**2. Comparative Analysis Across Job Titles and Locations:**

* **Job Title Comparison:** Comparing the skill sets required for different software engineering roles to identify core competencies and specialized skills.
* **Location-Based Analysis:** Analyzing job descriptions from different locations to understand regional variations in skill demand and industry trends.

By conducting these analyses, we aim to provide more actionable insights for job seekers and employers, including:

* **Tailored Resume Optimization:** Identifying the most relevant keywords and skills to include in resumes based on specific job postings.
* **Targeted Job Search:** Focusing on job postings that align with an individual's skill set and career goals.
* **Skill Development Recommendations:** Identifying skill gaps and suggesting relevant training or certifications.
* **Employer Branding:** Understanding the specific needs and preferences of employers to attract top talent.

We believe that by leveraging advanced text analysis techniques and a comprehensive dataset, we can provide more precise and actionable recommendations for navigating the complex software engineering job market.