JOB ANALYTICS

TEAM MEMBERS:

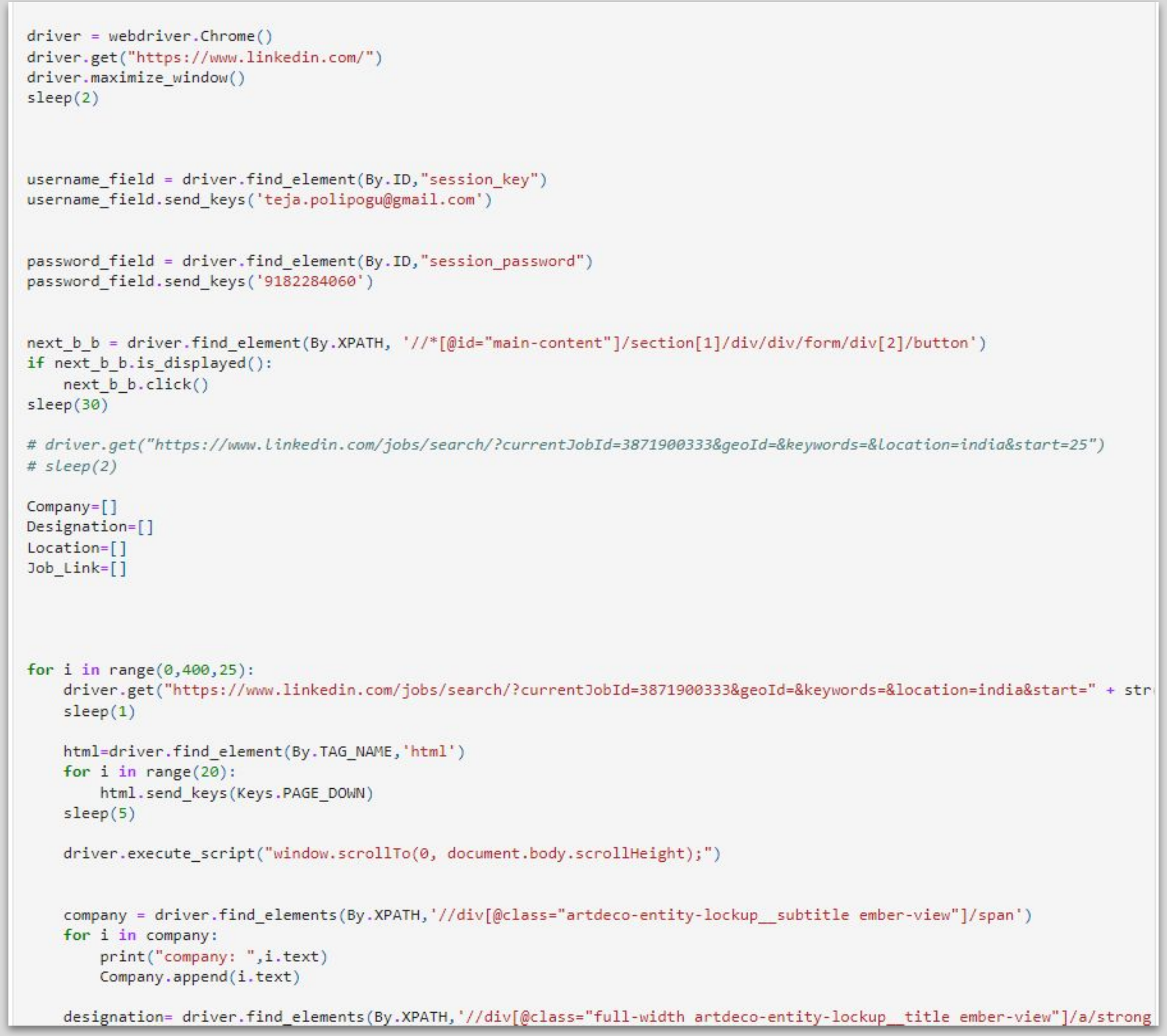
1. P. Penchala Teja
2. Allagadapa Harish
3. Harshit Gupta
4. Raghav Sharma
5. Bibhakar Paul

OBJECTIVE

The purpose of this job analysis is to provide a comprehensive understanding of the various job positions in India which is posted on Linkedin.This job analysis will encompass all key job positions in different companies, spanning across departments and divisions. We will examine the duties, responsibilities, qualifications, skills, and competencies required for each position.

EXTRACTION

Data extraction serves a pivotal role, particularly when faced with the intricate task of obtaining data page wise and scrolling the page.To overcome this challenge, we strategically used loop on link of each pages for a seamless and efficient extraction process.



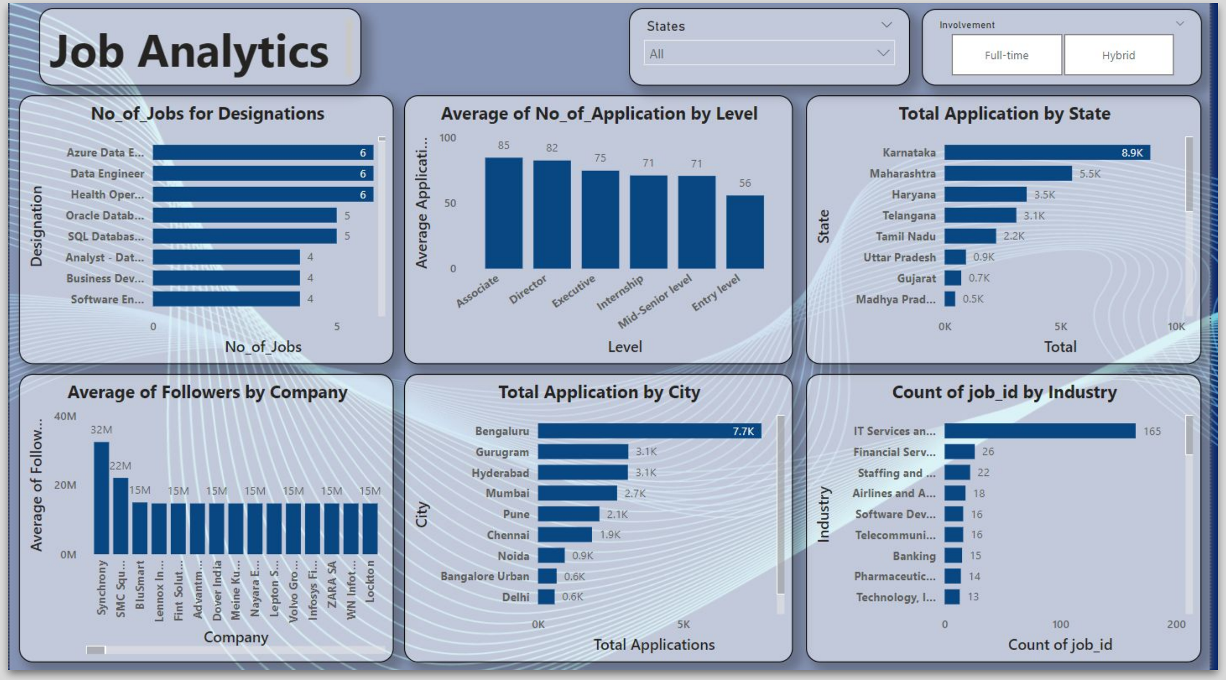
```python
driver = webdriver.Chrome()
driver.get("https://www.linkedin.com/")
driver.maximize_window()
sleep(2)



username_field = driver.find_element(By.ID,"session_key")
username_field.send_keys('teja.polipogu@gmail.com')


password_field = driver.find_element(By.ID,"session_password")
password_field.send_keys('9182284060')


next_b_b = driver.find_element(By.XPATH, '//*[@id="main-content"]/section[1]/div/div/form/div[2]/button')
if next_b_b.is_displayed():
    next_b_b.click()
sleep(30)

# driver.get("https://www.linkedin.com/jobs/search/?currentJobId=3871900333&geoId=&keywords=&location=india&start=25")
# sleep(2)

Company=[]
Designation=[]
Location=[]
Job_Link=[]




for i in range(0,400,25):
    driver.get("https://www.linkedin.com/jobs/search/?currentJobId=3871900333&geoId=&keywords=&location=india&start=" + str
    sleep(1)

    html=driver.find_element(By.TAG_NAME,'html')
    for i in range(20):
        html.send_keys(Keys.PAGE_DOWN)
    sleep(5)

    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")


    company = driver.find_elements(By.XPATH,'//div[@class="artdeco-entity-lockup__subtitle ember-view"]/span')
    for i in company:
        print("company: ",i.text)
        Company.append(i.text)

    designation= driver.find_elements(By.XPATH,'//div[@class="full-width artdeco-entity-lockup__title ember-view"]/a/strong
```

DASHBOARD

Integrating data from multiple sources with different formats and structures, and cleansing it to ensure accuracy and consistency.

Compare job postings and recruitment metrics with competitors in the industry.

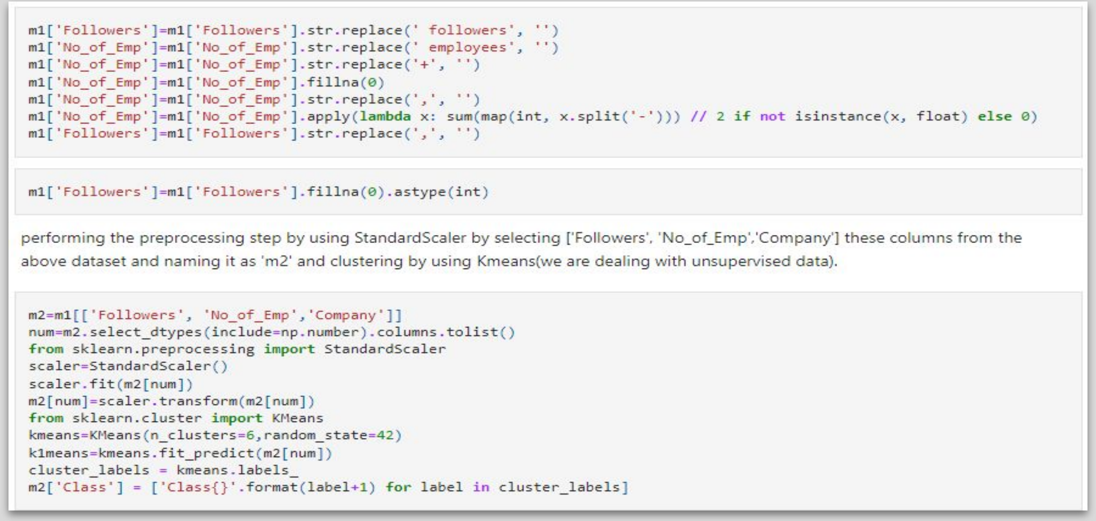
Visualize the trend of job postings over time to identify fluctuations and patterns in job demand.



CLUSTERING AND NLTK

Use clustering algorithms to categorize companies based on attributes such as employee count and LinkedIn followers. This categorization can help in understanding market segments and identifying patterns among different types of companies.

Apply clustering algorithms to analyze and visualize patterns in the data. Use NLTK for additional analysis tasks, such as sentiment analysis or skill extraction, to gain deeper insights into the clustered data.

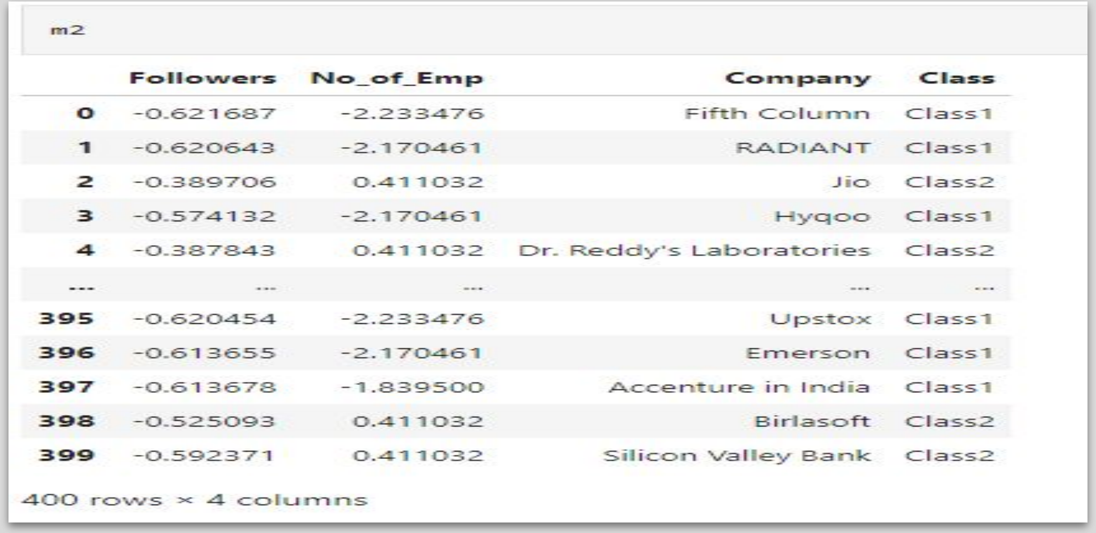


```
m1['Followers']=m1['Followers'].str.replace(' followers', '')
m1['No_of_Emp']=m1['No_of_Emp'].str.replace(' employees', '')
m1['No_of_Emp']=m1['No_of_Emp'].str.replace('+', '')
m1['No_of_Emp']=m1['No_of_Emp'].fillna(0)
m1['No_of_Emp']=m1['No_of_Emp'].str.replace(',', '')
m1['No_of_Emp']=m1['No_of_Emp'].apply(lambda x: sum(map(int, x.split('-'))) // 2 if not isinstance(x, float) else 0)
m1['Followers']=m1['Followers'].str.replace(',', '')
```

```
m1['Followers']=m1['Followers'].fillna(0).astype(int)
```

performing the preprocessing step by using StandardScaler by selecting ['Followers', 'No_of_Emp','Company'] these columns from the above dataset and naming it as 'm2' and clustering by using Kmeans(we are dealing with unsupervised data).

```
m2=m1[['Followers', 'No_of_Emp','Company']]
num=m2.select_dtypes(include=np.number).columns.tolist()
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(m2[num])
m2[num]=scaler.transform(m2[num])
from sklearn.cluster import KMeans
kmeans=KMeans(n_clusters=6,random_state=42)
k1means=kmeans.fit_predict(m2[num])
cluster_labels = kmeans.labels_
m2['Class'] = ['Class{}'.format(label+1) for label in cluster_labels]
```



```
m2
```

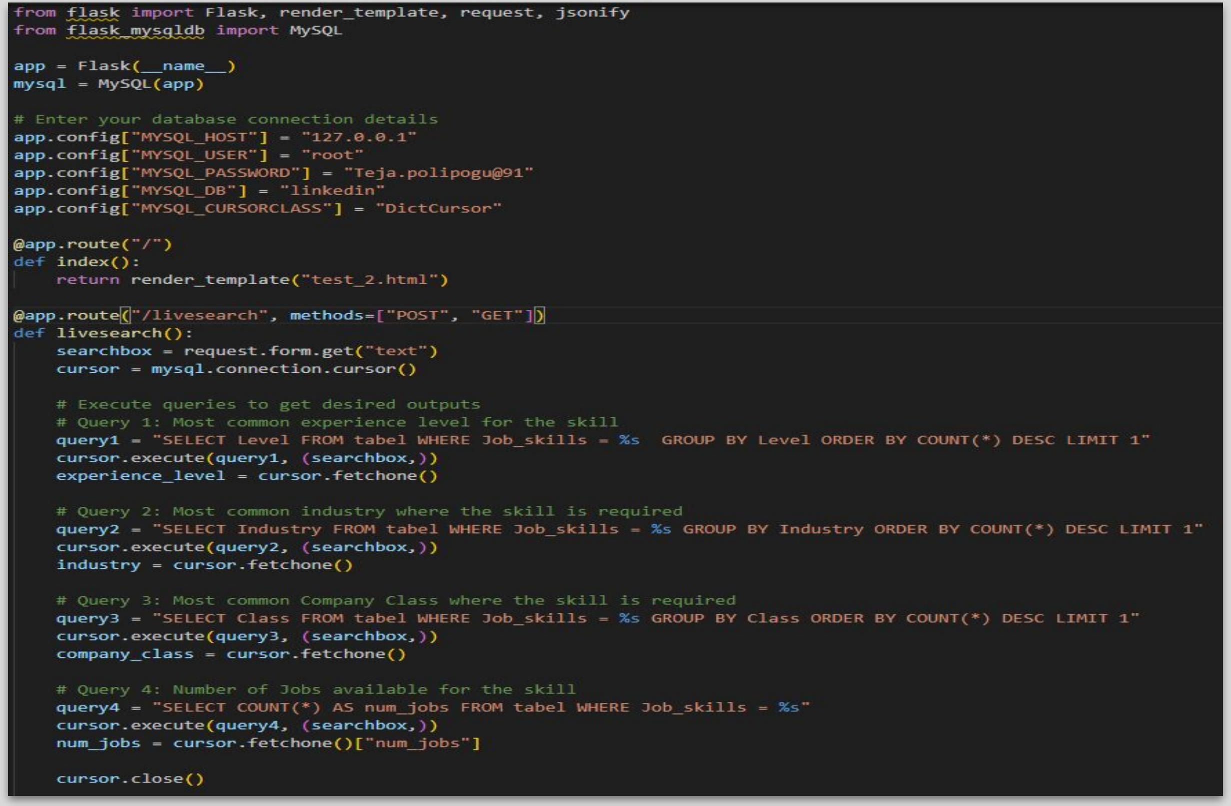
	Followers	No_of_Emp	Company	Class
0	-0.621687	-2.233476	Fifth Column	Class1
1	-0.620643	-2.170461	RADIANT	Class1
2	-0.389706	0.411032	Jio	Class2
3	-0.574132	-2.170461	Hyqoo	Class1
4	-0.387843	0.411032	Dr. Reddy's Laboratories	Class2
...	...	...	...	...
395	-0.620454	-2.233476	Upstox	Class1
396	-0.613655	-2.170461	Emerson	Class1
397	-0.613678	-1.839500	Accenture in India	Class1
398	-0.525093	0.411032	Birlasoft	Class2
399	-0.592371	0.411032	Silicon Valley Bank	Class2

400 rows × 4 columns

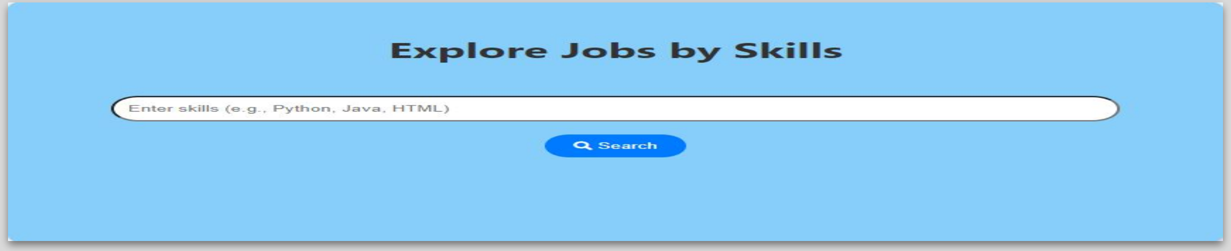
SEARCH BAR

One of the primary challenges encountered was loading data into the search bar efficiently. This involved integrating the search functionality with the backend database and ensuring smooth retrieval and display of search results.

To address this challenge, extensive research was conducted into various methods for data retrieval and integration with frontend components. Ultimately, a combination of MySQL connection requests and server-side APIs was implemented to fetch and display search results dynamically. Additionally, caching mechanisms were employed to optimize data retrieval and minimize latency.



```python
from flask import Flask, render_template, request, jsonify
from flask_mysqldb import MySQL

app = Flask(__name__)
mysql = MySQL(app)

# Enter your database connection details
app.config["MYSQL_HOST"] = "127.0.0.1"
app.config["MYSQL_USER"] = "root"
app.config["MYSQL_PASSWORD"] = "Teja.polipogu@91"
app.config["MYSQL_DB"] = "linkedin"
app.config["MYSQL_CURSORCLASS"] = "DictCursor"

@app.route("/")
def index():
    return render_template("test_2.html")

@app.route("/livesearch", methods=["POST", "GET"])
def livesearch():
    searchbox = request.form.get("text")
    cursor = mysql.connection.cursor()

    # Execute queries to get desired outputs
    # Query 1: Most common experience level for the skill
    query1 = "SELECT Level FROM tabel WHERE Job_skills = %s  GROUP BY Level ORDER BY COUNT(*) DESC LIMIT 1"
    cursor.execute(query1, (searchbox,))
    experience_level = cursor.fetchone()

    # Query 2: Most common industry where the skill is required
    query2 = "SELECT Industry FROM tabel WHERE Job_skills = %s GROUP BY Industry ORDER BY COUNT(*) DESC LIMIT 1"
    cursor.execute(query2, (searchbox,))
    industry = cursor.fetchone()

    # Query 3: Most common Company Class where the skill is required
    query3 = "SELECT Class FROM tabel WHERE Job_skills = %s GROUP BY Class ORDER BY COUNT(*) DESC LIMIT 1"
    cursor.execute(query3, (searchbox,))
    company_class = cursor.fetchone()

    # Query 4: Number of Jobs available for the skill
    query4 = "SELECT COUNT(*) AS num_jobs FROM tabel WHERE Job_skills = %s"
    cursor.execute(query4, (searchbox,))
    num_jobs = cursor.fetchone()["num_jobs"]

    cursor.close()
```



CHALLENGES AND LEARNINGS OUTCOMES

Challenges:

- Extracting relevant skills and qualifications from unstructured job descriptions.
- Dealing with special characters, punctuation, and typos in search queries.
- Providing real-time updates to search results as the user types.
- Supporting complex search queries with multiple filters, operators, and conditions.

Learning Outcomes:

- Enhance programming skills in languages such as Python, PowerBI, or SQL for data manipulation, analysis, and visualization.
- Learn to leverage libraries and frameworks such as pandas, scikit-learn, and matplotlib for data analytics tasks.
- Learn techniques for ensuring cross-platform compatibility and responsiveness of the search bar across different devices and browsers.
- Gain proficiency in version control systems such as Git for managing code changes, branching, and collaboration with team members.
- Develop skills in error handling and debugging to identify and resolve issues that arise during search implementation.

THANK YOU