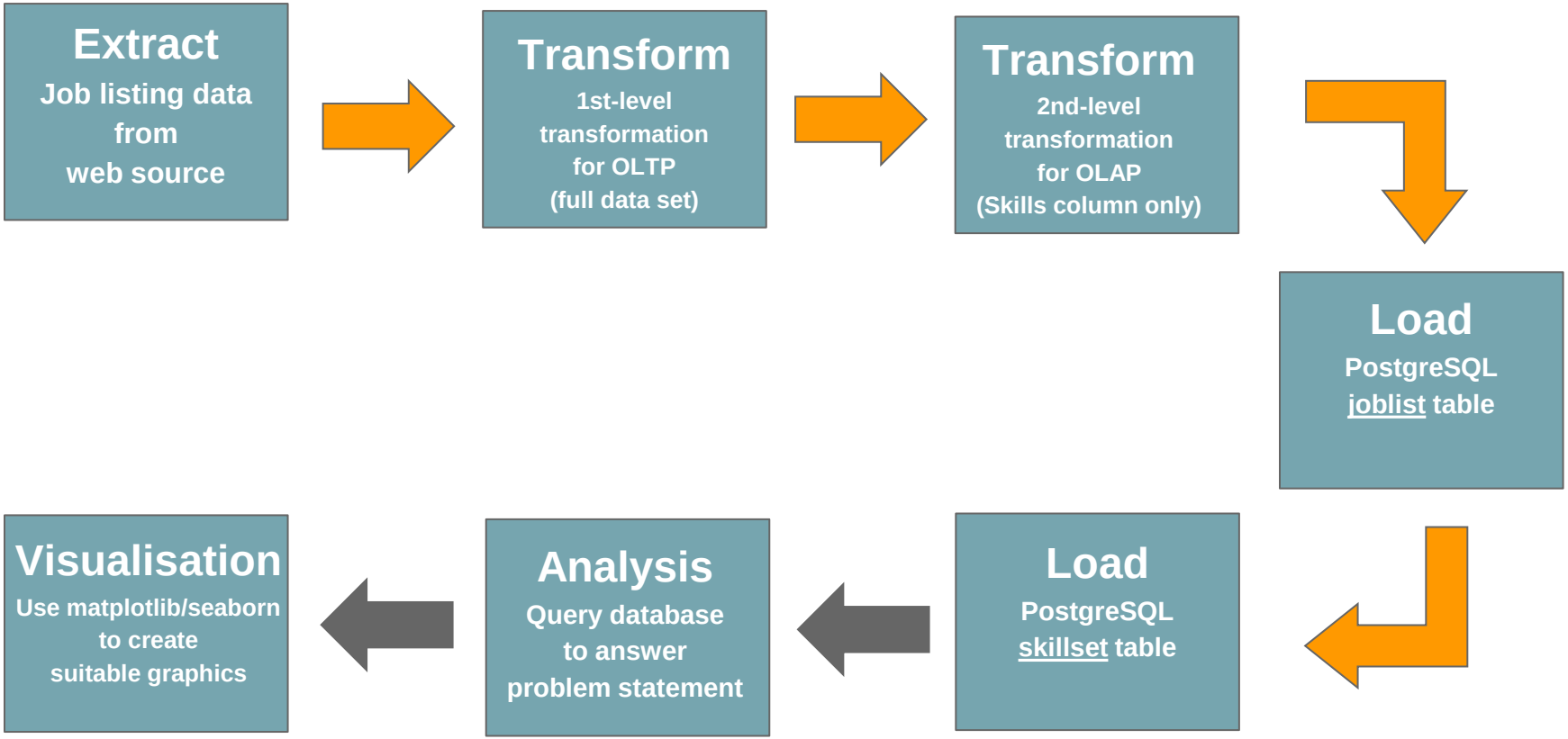
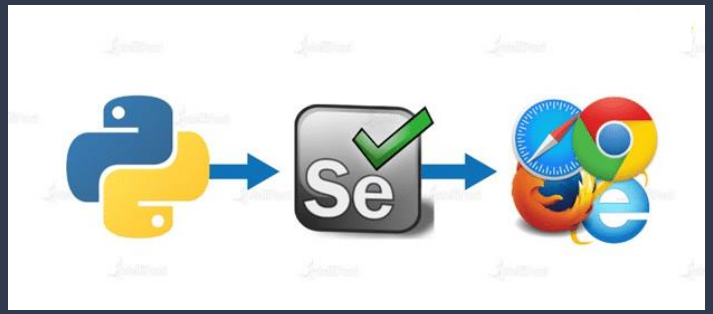
# An Analysis of Job Postings

# Objective

At a given point of time,
what are the most in-demand skills
requested by employers for Data Engineers in
Singapore, based on an analysis of
job postings found on a job portal.




# Pipeline Conceptualisation




# EXTRACTION

# Scraping and its issues

Decided to scrape data from a job portal

- Websites that block any kind of machine-based work (Indeed)
- Websites that can ban your account/IP address for heavy machine use (Linkedin Jobs)
- Do not overload servers with too many requests, check a website's terms of service on scraping.
- Recommended to not do very frequent extractions, and to use a temp account.

Determining a suitable website and Python library

- Website code had to be inspected to see if the data required is written in a suitable format. Best for required data to be in their separate element on the web code
- Websites that do not provide specific job details on the results search page (e.g., Jobstreet)
- Python extraction libraries can scrape data by using the xPath, element ID, Class Name, CSS Selectors, or Tag Name of the web element
- Understand your libraries' capabilities and limitations - for example, Beautifulsoup cannot parse JavaScript-based web elements.

```python
# Web extraction using selenium - import selenium package tools

from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.support.select import Select
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import TimeoutException
```

1

```python
def extract(page):
    url = f'https://www.foundit.sg/srp/results?start={page}&sort=1&limit=15&query=%22data+engineer%22&locations=Singapore'
    driver.get(url)
    driver.implicitly_wait(2)
    return driver.page_source  # Return the page source
```

2

```python
job_title = card.find_element(By.XPATH, ".//div[@class='jobTitle']").text.strip()

company_name = card.find_element(By.XPATH, ".//div[contains(@class, 'companyName')]

job_details = card.find_element(By.XPATH,'.//div[@class="details"]').text
```

3

```python
# Store job data in a dictionary
job = {
    "Job Title": job_title,
    "Company Name": company_name,
    "Job Details": job_details,
    "Skills": skills,
    "Time" : time,
    "Years of Exp" : years,
    "Salary" : salary
            }
joblist.append(job)
```

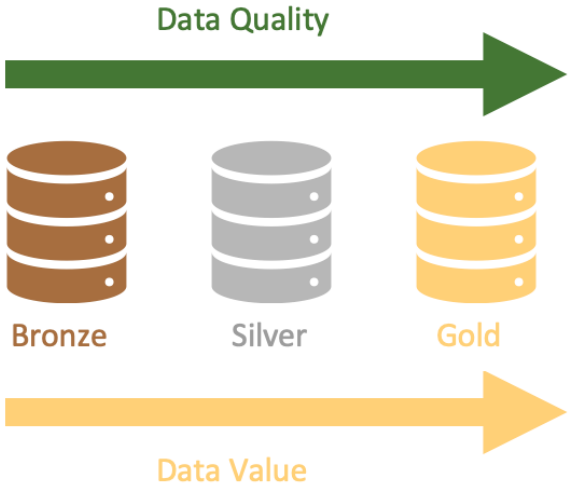
4

```python
for i in range(0,210,15):
    print(f'Getting list,{i}')
    page_source = extract(i)
    transform(page_source)

df = pd.DataFrame(joblist)
df.to_csv('raw_datalist_22March.csv')

# Use %store magic command to store the DataFrame
# to be retrievable in other notebooks
%store df

# Close the driver
driver.quit()
```

5

# Extraction codes

# TRANSFORMATION



# Desired Outcomes

1. **Raw (but clean) dataset**
   - **Pilot study**

| | job_id | job_title | company_name | employment_type | skills | date_posted | experience | salary |
|---|---|---|---|---|---|---|---|---|
| 0 | JD001 | Trainee to Junior Data Engineer | Luxoft India Llp | Full time | Data Scientist | a day ago | 0-3 Years | Not specified |
| 1 | JD002 | Big Data Engineer - Hortonworks Hadoop Cloudera | Kerry Consulting Pte Ltd | Contract Job, Part time | big data hadoop cloudera hortonworks data engi... | 10 days ago | 5-10 Years | 417-625 SGD monthly |
| 2 | JD003 | Data Engineer | Elliott Moss Consulting Pte Ltd | Contract Job, Part time | a/b testing, Econometrics, Pricing, Python, Ma... | 9 days ago | 3-10 Years | Not specified |

2. **Job listing and individual skill**
   - **Main objective**

| | job_id | skills |
|---|---|---|
| 0 | JD001 | Data Scientist |
| 1 | JD002 | Big Data Hadoop Cloudera Hortonworks Data Engi... |
| 2 | JD003 | A/B Testing |
| 3 | JD003 | Econometrics |
| 4 | JD003 | Pricing |

# Data Exploration & Cleaning

1

```
# rename fields
df1.columns = ["job_title",
               "company_name",
               "employment_type",
               "skills",
               "date_posted",
               "experience",
               "salary"]
```

2

```
#check for null values
df1.isnull().sum()

# Replace null values in 'Skills' column with "Not specified"
df1['skills'] = df1['skills'].fillna("Not Specified")
```

3

```
# Remove duplicates
df1 = df1.drop_duplicates()
```

```
# Changing the index to 'JD001', 'JD002', etc.
df1.index= ['JD{:03d}'.format(i+1) for i in range(len(df1))]

# Assigning column header 'Job ID' to the index
df1.index.name ='job_id'
```

5

4

```
# Remove unnecessary commas at the end of each string & ' " ' in the 'skills' column
df1['skills'] = df1['skills'].str.rstrip(', ').str.replace('"', '')
```

1. **Raw (but clean) dataset**

| | job_id | job_title | company_name | employment_type | skills | date_posted | experience | salary |
|---|---|---|---|---|---|---|---|---|
| 0 | JD001 | Trainee to Junior Data Engineer | Luxoft India Llp | Full time | Data Scientist | a day ago | 0-3 Years | Not specified |
| 1 | JD002 | Big Data Engineer - Hortonworks Hadoop Cloudera | Kerry Consulting Pte Ltd | Contract Job, Part time | big data hadoop cloudera hortonworks data engi... | 10 days ago | 5-10 Years | 417-625 SGD monthly |
| 2 | JD003 | Data Engineer | Elliott Moss Consulting Pte Ltd | Contract Job, Part time | a/b testing, Econometrics, Pricing, Python, Ma... | 9 days ago | 3-10 Years | Not specified |

2. **Job listing and individual skill**

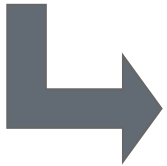
| | job_id | skills |
|---|---|---|
| 0 | JD001 | Data Scientist |
| 1 | JD002 | Big Data Hadoop Cloudera Hortonworks Data Engi... |
| 2 | JD003 | A/B Testing |
| 3 | JD003 | Econometrics |
| 4 | JD003 | Pricing |

# Further Transformation

1

```
# Split the 'skills' column by comma and explode it to create multiple rows
df4 = df1.assign(skills=df1['skills'].str.split(',')).explode('skills')
```

| | job_id | job_title | company_name | employment_type | skills | date_posted | experience | salary |
|---|---|---|---|---|---|---|---|---|
| **149** | JD150 | Data Engineering | Luxoft India Llp | Full time | Java, Python, Sql | 4 days ago | 0-3 Years | 3000-4600 SGD monthly |



| | job_id | skills |
|---|---|---|
| **669** | JD150 | Java |
| **670** | JD150 | Python |
| **671** | JD150 | Sql |

# Standardisation of Skill Names

2

```
# Check frequency of each skill to determine what can be combined and standardised
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()

# Create a dictionary to hold grouped words
```

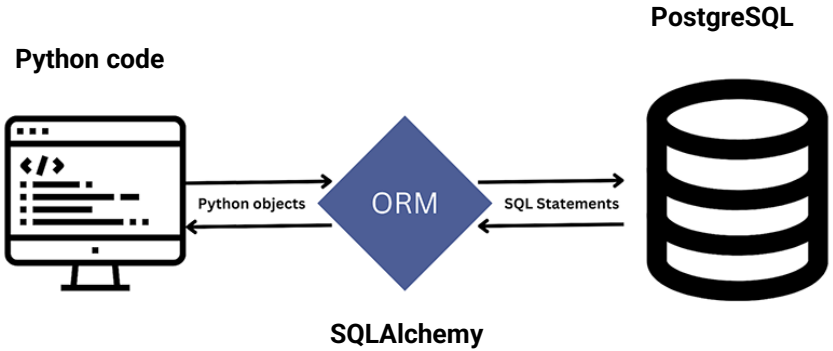
```
Stem 10: pyspark
Words: ['Pyspark', 'Pyspark', 'Pyspark', 'Pyspark', 'Pyspark', 'Pyspark', 'Py
spark', 'Pyspark', 'Pyspark', 'Pyspark', 'Pyspark', 'Pyspark', 'Pyspark']

Stem 22: spark
Words: ['Spark', 'Spark', 'Spark', 'Spark', 'Spark']

Stem 44: apache spark
Words: ['Apache Spark', 'Apache Spark', 'Apache Spark', 'Apache Spark', 'Apache Spark']
```

```
skills_df1['skills'] = skills_df1['skills'].replace(['aws','amazon web services', 'amazon web service'], 'AWS')
skills_df1['skills'] = skills_df1['skills'].replace(['apache','spark','apache spark','Apache Pyspark','Spark',
                                                     'Apache Spark','pyspark'], 'Apache Spark/Pyspark')
```

python + PostgreSQL

SQLAlchemy

# LOADING...



# Loading Data using SQLAlchemy

**SQLAlchemy** is a powerful tool kit and Object-Relational Mapping(**ORM**) library for Python.

In this project, we are using ‘Engine’ -

- maintains a pool of connections to the database, allowing efficient reuse of connections across multiple requests.
- It handles the execution of SQL commands, whether they are DDL , DML or DQL
- It integrates seamlessly with other components of SQLAlchemy, such as the ORM and SQL expression language.
- It supports various database management systems, including SQLite, MySQL, PostgreSQL, and more.

Install SQLAlchemy

1

```
!pip install sqlalchemy
```

Import all the libraries

2

```
#Import libraries for sqlalchemy

import sqlalchemy as db
from sqlalchemy import create_engine
from sqlalchemy.exc import SQLAlchemyError
from sqlalchemy_utils import create_database
```

Establish connection to PostgreSQL database '**joblisting_dataeng**'

```
engine = db.create_engine('postgresql://postgres:admin@localhost:5432/joblisting_dataeng')

create_database(engine.url)

conn = engine.raw_connection()
```

3

Configuration details:

- Host: localhost
- Port: 5432
- Username: postgres
- Password: admin

4

# Install, import & create a database connection

**Joblist Table**

```
try:
    # Create a cursor object
    cur = conn.cursor()

    # Create table joblist if it doesn't exist in PostgreSQL
    create_joblist = '''
                CREATE TABLE IF NOT EXISTS joblist(
                job_id VARCHAR(10) PRIMARY KEY,
                job_title VARCHAR(150),
                company_name VARCHAR(150),
                employment_type VARCHAR(50),
                skills VARCHAR(250),
                date_posted VARCHAR(50),
                experience VARCHAR(20),
                salary VARCHAR(120)
                );
                '''

    # Execute SQL command
    cur.execute(create_joblist)
```

**Skillset Table**

```
    # Create the table skillset if it doesn't exist
    create_skillset = '''
                CREATE TABLE IF NOT EXISTS skillset(
                id SERIAL PRIMARY KEY,
                job_id VARCHAR(10),
                skills VARCHAR(100),
                FOREIGN KEY (job_id) REFERENCES joblist(job_id)
                );
                '''

    cur.execute(create_skillset)

    # Commit changes
    conn.commit()

except Exception as e:
    print("Error:", e)

finally:
    # Close communication with server
    cur.close()
    conn.close()
```

# Create tables  and load to PostgreSQL

```
joblist = pd.read_csv('joblist_clean.csv',
                      sep = ',',index_col=None)
joblist.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 179 entries, 0 to 178
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   job_id           179 non-null    object
 1   job_title        179 non-null    object
 2   company_name     179 non-null    object
 3   employment_type  179 non-null    object
 4   skills           177 non-null    object
 5   date_posted      179 non-null    object
 6   experience       179 non-null    object
 7   salary           179 non-null    object
dtypes: object(8)
memory usage: 11.3+ KB
```

1

```
skillsets = pd.read_csv('skillset_standardized.csv',
                        sep = ',')
skillsets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 788 entries, 0 to 787
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   job_id  788 non-null    object
 1   skills  786 non-null    object
dtypes: object(2)
memory usage: 12.4+ KB
```

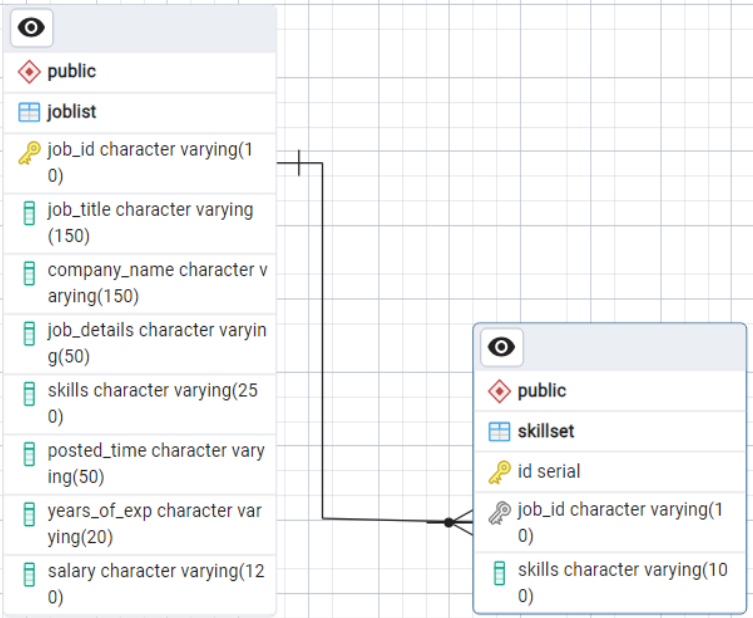
2

```
# Load dataframes into the database

joblist.to_sql(name='joblist', con=engine,
               if_exists='append', index=False)

skillsets.to_sql(name='skillset', con=engine,
                 if_exists='append', index=False)
```

3

# Read & Load the csv files to the tables in PostgreSQL

ERD of **joblisting_dataeng** database




View table for the number of occurrences of the **Top 15 Skills**

```
CREATE VIEW skillset_view AS
SELECT DISTINCT skills, COUNT(*) AS skill_count
FROM skillset
GROUP BY skills
ORDER BY skill_count DESC;

SELECT * FROM skillset_view;
```

View table

Data Output | Messages | Notifications

| | skills character varying (100) | skill_count bigint |
|---|---|---|
| 1 | Etl/Elt Pipeline | 89 |
| 2 | Azure | 40 |
| 3 | Data Visualization/Power Bi | 29 |
| 4 | Sql | 29 |
| 5 | Data Analysis | 28 |
| 6 | Big Data | 26 |
| 7 | Machine Learning | 25 |
| 8 | Hadoop | 24 |
| 9 | Apache Spark/Pyspark | 23 |
| 10 | Business Intelligence | 22 |
| 11 | Azure Data Factory | 17 |
| 12 | Data Engineering | 16 |
| 13 | Tableau | 15 |
| 14 | Scala | 12 |
| 15 | Data Modelling | 12 |

Total rows: 176 of 176 Query complete 00:00:00.192

# Entity Relationship Diagram of Joblist Foundit


# DATA ANALYSIS & VISUALIZATION

# Extract from the database & view the top 15 Skills

```
# Connect to PostgreSQL database (assuming 'joblist' is your existing database)
engine = create_engine('postgresql://postgres:admin@localhost:5432/joblisting_dataeng')

# Write SQL query to select data from skillsets_view
skills_query = "SELECT * FROM skillset_view;"

# Execute SQL query and fetch data into a DataFrame
df = pd.read_sql_query(skills_query, engine)

print(df.head(15))

# Get the top 15 skills by count
top_skills = df.nlargest(15, 'skill_count')
```

| | skills | skill_count |
|---|---|---|
| 0 | Etl/Elt Pipeline | 89 |
| 1 | Azure | 40 |
| 2 | Sql | 29 |
| 3 | Data Analysis | 28 |
| 4 | Data Visualization/Power Bi | 28 |
| 5 | Big Data | 26 |
| 6 | Machine Learning | 25 |
| 7 | Hadoop | 24 |
| 8 | Apache Spark/Pyspark | 23 |
| 9 | Business Intelligence | 22 |
| 10 | Azure Data Factory | 17 |
| 11 | Data Engineering | 16 |
| 12 | Tableau | 15 |
| 13 | Data Modelling | 12 |
| 14 | Data Reliability | 12 |

## Import all the libraries

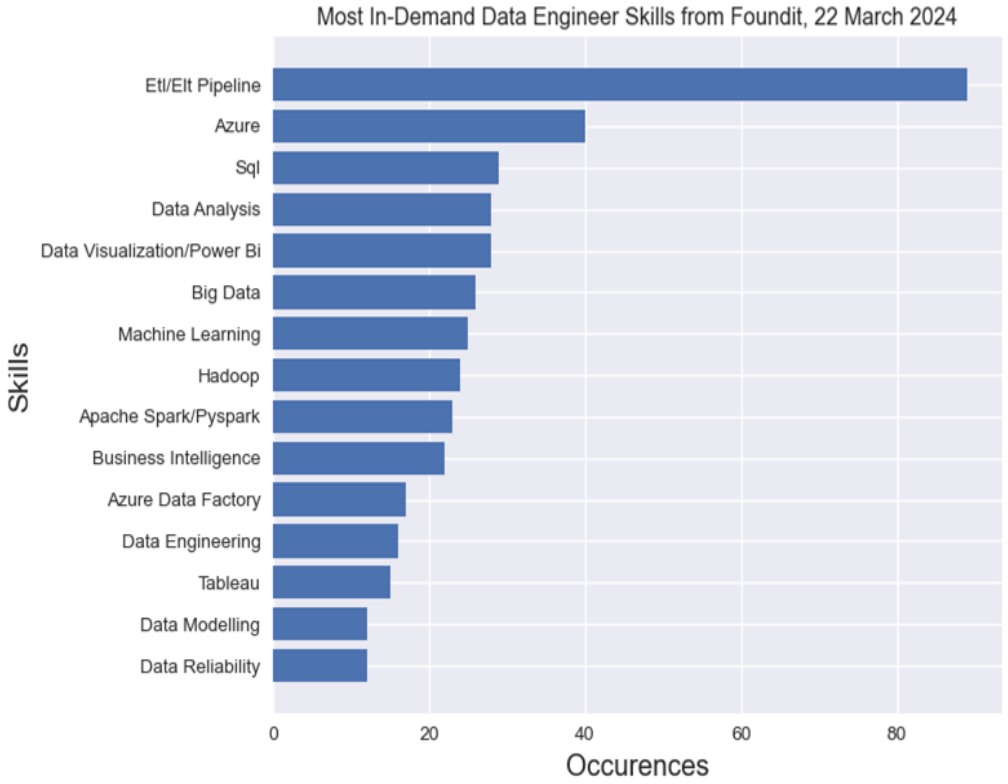
1

```
# Import libraries or visualisation

import matplotlib.pyplot as plt
from wordcloud import WordCloud
```

## Horizontal Bar Chart

2

```
plt.style.use('seaborn-v0_8')

plt.barh(top_skills['skills'], top_skills['skill_count'])
plt.ylabel('Skills', fontsize=15)
plt.xlabel('Occurences', fontsize=15)
plt.gca().invert_yaxis()  # Invert the y-axis

plt.title('Most In-Demand Data Engineer Skills from Foundit, 22 March 2024'

plt.tight_layout()
plt.show()
```




# Horizontal Bar Chart Analysis

```
# Get the top 150 skills by count
top_150_skills = df.nlargest(150, 'skill_count')

# Convert the DataFrame to a dictionary of skill counts
skills_dict = dict(zip(top_150_skills['skills'],
                       top_150_skills['skill_count']))

# Generate the word cloud
wordcloud = WordCloud(background_color="white",
                      width=1500,
                      height=800,
                      scale=5,
                      relative_scaling=0,
                      color_func=lambda *args, **kwargs: (0,0,128)
                     ).generate_from_frequencies(skills_dict)

plt.figure(figsize=(20,10))
plt.axis("off")
plt.imshow(wordcloud)
```


# Word Cloud Analysis

# Conclusion

1. Challenges
2. Summary
3. Implications
4. Future Directions