

# Hash Agile Task

## Question:

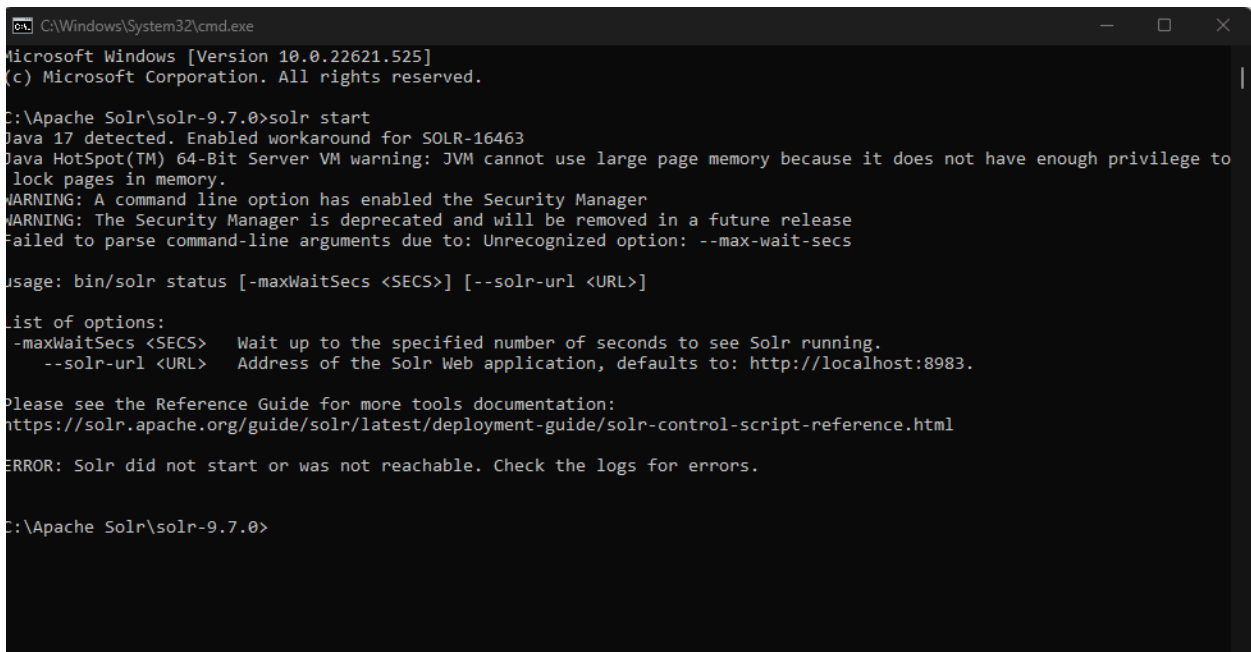
Indexing Using Apache Solr

Dataset: [Link](#)

## Steps:

Installed Solr From Official Website

Then using cmd start The solr using solr start command



```
CA\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22621.525]
(c) Microsoft Corporation. All rights reserved.

C:\Apache Solr\solr-9.7.0>solr start
Java 17 detected. Enabled workaround for SOLR-16463
Java HotSpot(TM) 64-Bit Server VM warning: JVM cannot use large page memory because it does not have enough privilege to
lock pages in memory.
WARNING: A command line option has enabled the Security Manager
WARNING: The Security Manager is deprecated and will be removed in a future release
Failed to parse command-line arguments due to: Unrecognized option: --max-wait-secs

Usage: bin/solr status [-maxWaitSecs <SECS>] [--solr-url <URL>]

List of options:
--maxWaitSecs <SECS>  Wait up to the specified number of seconds to see Solr running.
--solr-url <URL>      Address of the Solr Web application, defaults to: http://localhost:8983.

Please see the Reference Guide for more tools documentation:
https://solr.apache.org/guide/solr/latest/deployment-guide/solr-control-script-reference.html

ERROR: Solr did not start or was not reachable. Check the logs for errors.

C:\Apache Solr\solr-9.7.0>
```

Then Go to the mentioned port <http://localhost:8983>

Then go to the documents tab to create the index for the downloaded CSV dataset file.

Copy the contents from the CSV file and manually paste it in the Document field and submit it.

Second Challenge for Hash Agile Int | HAT Interview - Second Round Prog | Hash Agile Task - Google Docs | Solr Admin

localhost:8983/solr/#/employees/documents

Inbox | Gmail | ChatGPT | GitHub | LinkedIn | LeetCode | Top Interview Quest... | Unstop | New folder

**Solr**

- Dashboard
- Logging
- Security
- Core Admin
- Java Properties
- Thread Dump
- employees
  - Overview
  - Analysis
  - Documents
  - Paramsets
  - Files
  - Ping
  - Plugins / Stats
  - Query
  - Replication

Request-Handler (qt)  
/update

Document Type  
JSON

Document(s)

```
E02251,Genesis Herrera,Manager,IT,Research &
Development,Female,Latino,34,10/3/2015,"$126,898 ",10%,Brazil,Manaus,
E02252,Olivia Vazquez,Network Engineer,IT,Specialty
Products,Female,Latino,53,4/13/2020,"$93,053 ",0%,Brazil,Sao Paulo,
E02253,Leilani Ng,Systems Analyst,IT,Corporate,Female,Asian,48,9/19/2011,"$50,513
",0%,United States,Seattle,10/30/2019
E02254,Olivia Mendoza,Sr. Account
Representative,Sales,Corporate,Female,Latino,43,5/7/2017,"$86,533 ",0%,United
States,Columbus,
```

Commit Within  
1000

Overwrite  
true

Submit Document

Documentation

Then it will create indexes for each field and verify them by going to schema tab and by selecting a field.

Inbox | Gmail | ChatGPT | GitHub | LinkedIn | LeetCode | Top Interview Quest... | Unstop | New folder

**Solr**

- Dashboard
- Logging
- Security
- Core Admin
- Java Properties
- Thread Dump
- employees
  - Overview
  - Analysis
  - Documents
  - Paramsets
  - Files
  - Ping
  - Plugins / Stats
  - Query
  - Replication

Add Field | Add Dynamic Field | Add Copy Field | Manipulate Field Type

Full\_Name

Field: Full\_Name

Field-Type: org.apache.solr.schema.TextField

PI Gap: 100

Docs: 1,222

Flags:	Indexed	Tokenized	Stored	Multivalued
Properties	✓	✓	✓	✓
Schema	✓	✓	✓	✓
Index	✓	✓	✓	

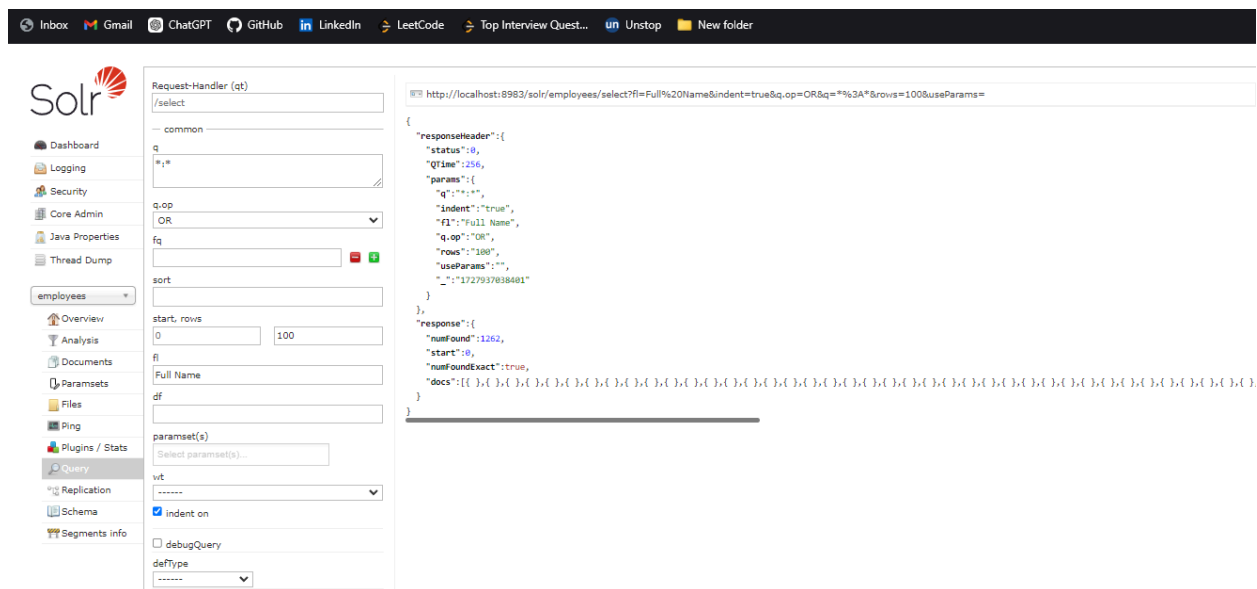
Index Analyzer: org.apache.solr.analysis.TokenizerChain

Query Analyzer: org.apache.solr.analysis.TokenizerChain

Load Term Info

Documentation | Solr Query S

Then go to the Query tab and give appropriate fields and check the indexing working correctly.



Then Additionally I have done an interface using Python Tkinter and retrieved the particular person data by their name field indexing the code for Python Tkinter is below.

(i have used Tkinter but At first i try to use HTML and jS but its giving CORS error, Cross-Origin Requests: When a web application tries to make requests to a server that is on a different origin (domain, protocol, or port) than its own, the browser blocks this request for security reasons unless the server explicitly allows it.) That's why I used tkinter since it is run in local desktop. So I thought it would work and it worked.

## Solr\_interface.py

```
import tkinter as tk
from tkinter import messagebox
import requests

# Solr server details
SOLR_URL = 'http://localhost:8983/solr/employees/select'

# Create main window
root = tk.Tk()
root.title("Employee Search")

# Create input field for search query
tk.Label(root, text="Enter Full Name:").pack(pady=10)
```

```

entry = tk.Entry(root, width=50)
entry.pack(pady=5)

# Create a text area to display results
results_text = tk.Text(root, height=20, width=80)
results_text.pack(pady=10)

# Function to search employees in Solr
def search_employees():
    query = entry.get()
    if not query:
        messagebox.showwarning("Input Error", "Please enter a name to search.")
        return

    params = {
        'q': f'Full_Name:"{query}"', # Use quotes to handle names with spaces
        'wt': 'json',
        'indent': 'true'
    }

    try:
        response = requests.get(SOLR_URL, params=params)
        data = response.json()

        results_text.delete(1.0, tk.END) # Clear previous results

        if data['response']['docs']:
            results = ""
            for doc in data['response']['docs']:
                # Extract fields
                full_name = doc.get('Full_Name', 'N/A')
                job_title = doc.get('Job_Title', 'N/A')
                department = doc.get('Department', 'N/A')
                business_unit = doc.get('Business_Unit', 'N/A')
                gender = doc.get('Gender', 'N/A')
                ethnicity = doc.get('Ethnicity', 'N/A')
                age = doc.get('Age', 'N/A')
                hire_date = doc.get('Hire_Date', 'N/A')
                annual_salary = doc.get('Annual_Salary', 'N/A')
                bonus = doc.get('Bonus %', 'N/A')
                country = doc.get('Country', 'N/A')
                city = doc.get('City', 'N/A')
                exit_date = doc.get('Exit_Date', 'N/A')

```

```

# Format the result string
results += (
    f"Full Name: {full_name}, Job Title: {job_title}, Department: {department}, "
    f"Business Unit: {business_unit}, Gender: {gender}, Ethnicity: {ethnicity}, "
    f"Age: {age}, Hire Date: {hire_date}, Annual Salary: {annual_salary}, "
    f"Bonus %: {bonus}, Country: {country}, City: {city}, Exit Date: {exit_date}\n"
)
results_text.insert(tk.END, results)
else:
    results_text.insert(tk.END, "No results found.")
except Exception as e:
    messagebox.showerror("Error", f"An error occurred: {e}")

# Create search button
search_button = tk.Button(root, text="Search", command=search_employees)
search_button.pack(pady=10)

# Run the application
root.mainloop()

```

## Explanation(Main Functionality)

### User Interface:

Input Field: A text box to enter the employee's full name.

Search Button: A button to initiate the search.

### Functionality:

When the user clicks the Search button:

The app sends a request to a Solr server to find the employee by name.

It uses the entered name to search the employee database.

### Display Results:

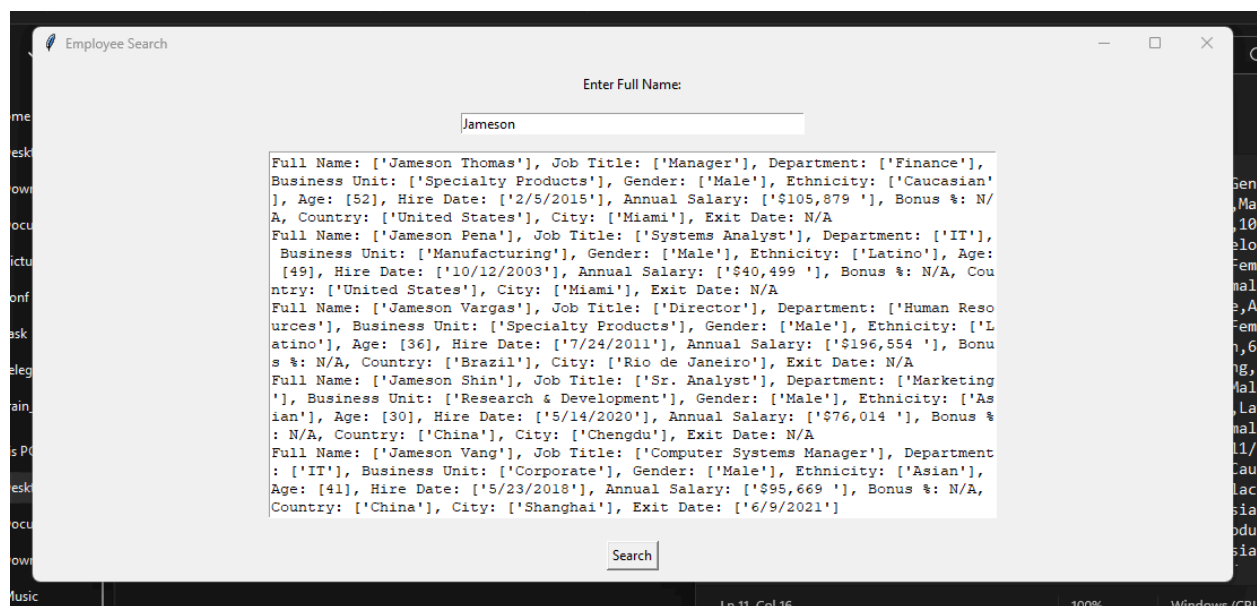
If an employee is found:

Their details (like job title, department, salary, etc.) are displayed.

If no employee is found:

A message indicates that no results were found.

## Output:



I understand Apache Solr is an open-source search platform that enables fast and powerful full-text search and indexing of large datasets. It is designed for scalability and can handle complex search queries efficiently.

Whenever there is a query of data, Apache Solr says Hi :)