



Git link : <https://github.com/TheDanHisway/hashagile-assignment>

- First to run solr as standalone,
Installed zookeeper and statered
- Started solr
- Exposed port using ngrok as i am using colab for further tasks and have to expose it globally

```
zkServer.cmd
solr start
ngrok http 8983
```

[illegible]

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

C:\Users\inika\Downloads\solr-9.7.0\bin>solr start -p 8983
Stopping Solr process 18590 running on port 8983
Waiting up to 180 seconds for process 18590 to exit

C:\Users\inika\Downloads\solr-9.7.0\bin>
C:\Users\inika\Downloads\solr-9.7.0\bin>solr start -z localhost:2181
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.
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:\Users\inika\Downloads\solr-9.7.0\bin>solr start

ERROR: Process 8396 is already listening on port 8983. If this is Solr, please stop it first before starting (or use restart). If this is not Solr, then please choose a different port using -p PORT

C:\Users\inika\Downloads\solr-9.7.0\bin>
```

```
Command Prompt - ngrok ht x + v
ngrok (Ctrl+C to quit)
Share what you're building with ngrok https://ngrok.com/share-your-ngrok-story

Session Status      online
Account             Mohamed Danish M A (Plan: Free)
Update              update available (version 3.16.1, Ctrl-U to update)
Version             3.16.0
Region              India (in)
Latency             1958ms
Web Interface       http://127.0.0.1:4040
Forwarding           https://f10f-223-181-208-107.ngrok-free.app -> http://localhost:8983

Connections
ttl    opn    rt1    rt5    p50    p90
154    0        0.00   0.00   1.63   90.39

HTTP Requests
-----
14:15:39.895 IST GET /solr/admin/collections 200 OK
14:15:39.893 IST GET /solr/admin/cores 200 OK
14:15:38.283 IST GET /solr/admin/info/system 200 OK
14:15:34.890 IST GET /solr/ 200 OK
13:46:44.338 IST GET /solr/Hash_Danish/facets 404 Not Found
13:46:43.715 IST GET /solr/Hash_Danish/select 404 Not Found
13:46:43.115 IST GET /solr/Hash_Danish/select 404 Not Found
13:46:42.355 IST GET /solr/Hash_Danish/select 404 Not Found
13:46:41.728 IST POST /solr/Hash_Danish/update 404 Not Found
13:46:40.256 IST POST /solr/Hash_Danish/update 404 Not Found
```

localhost | assign | Check | Download | Solr A | localhost | Solr A | S x | NEXT | New | hash | GitHub | IPYNI | +

localhost:8983/solr/#/~cloud

- Dashboard
- Logging
- Security
- Cloud
- Nodes
- Tree
- ZK Status
- Graph
- Schema Designer
- Collections
- Java Properties
- Thread Dump

Refresh Show all details

Hosts 1 - 1 of 1. Filter by: Host/node name Show 10 hosts per page.

Host	Node	CPU	Heap	Disk usage	Requests	Collections	Replicas
localhost Windows 11 7.8Gb Java 11 Load: ~1 show details...	8983_solr Uptime: 1h 12m Java 11.0.24 Solr 9.7.0 hide details...	1%	29% Max: 512.0Mb Used: 151.9Mb	528.4Kb Total #docs: 1.3k Avg size/doc: 428.8 Hash_MohamedDanish_s1r1: 528.4Kb	RPM: 0.01	Hash_MohamedDanish	Hash_MohamedDanish_s1r1 (1.3k docs) deleted: 0 warmupTime: 0 avg size/doc: 428.8b

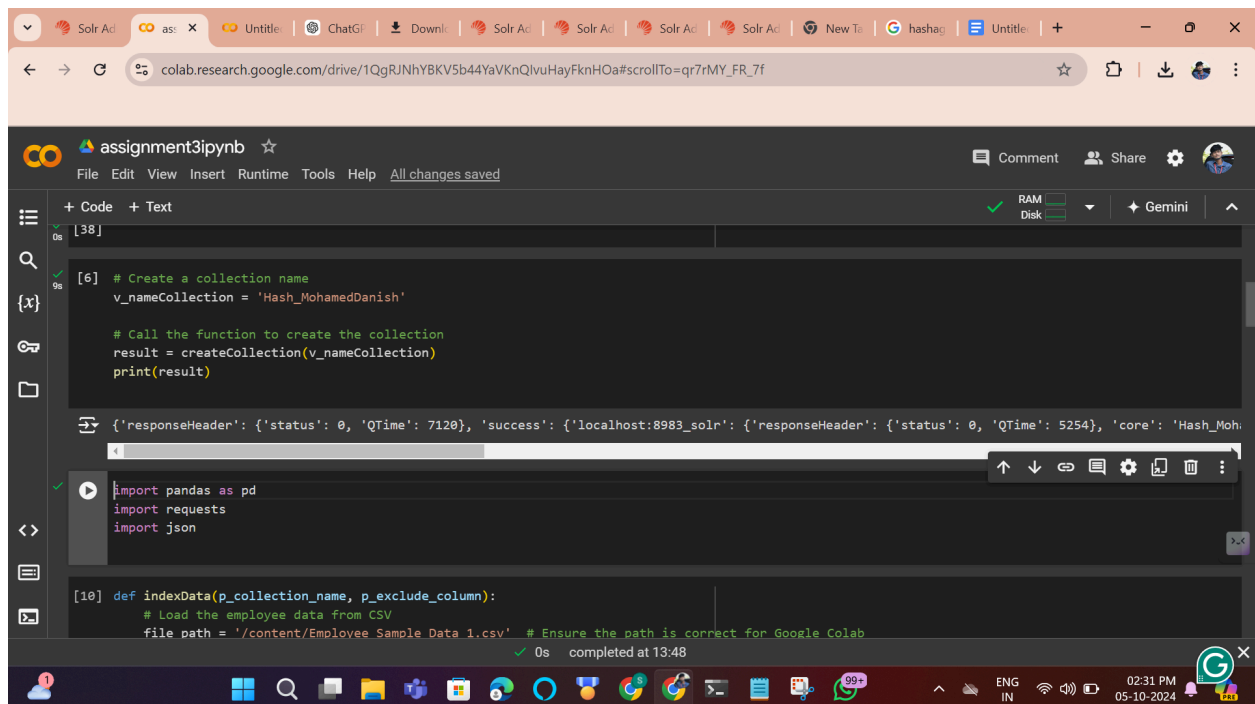
Documentation | Solr Query Syntax | Community | Issue Tracker | Slack | IRC

1] CREATING COLLECTIONS

```
import requests

def createCollection(p_collection_name):
    ngrok_url =
    "https://7c94-223-181-208-107.ngrok-free.app/solr/admin/collections"
    params = {
        "action": "CREATE",
        "name": p_collection_name,
        "numShards": 1,
        "replicationFactor": 1
    }
    response = requests.get(ngrok_url, params=params)
    return response.json()
```

Output



The screenshot shows a Google Colab notebook titled "assignment3ipynb". The notebook contains the following code:

```
[38]

[6] # Create a collection name
v_nameCollection = 'Hash_MohamedDanish'

# Call the function to create the collection
result = createCollection(v_nameCollection)
print(result)
```

The output of the code is a JSON object:

```
{'responseHeader': {'status': 0, 'QTime': 7120}, 'success': {'localhost:8983_solr': {'responseHeader': {'status': 0, 'QTime': 5254}, 'core': 'Hash_Mohi
```

The notebook also shows the following code:

```
import pandas as pd
import requests
import json

[10] def indexData(p_collection_name, p_exclude_column):
    # Load the employee data from CSV
    file_path = '/content/Employee Sample Data 1.csv' # Ensure the path is correct for Google Colab
```

The notebook interface includes a sidebar with icons for file management, search, and other tools. The bottom status bar shows the time as 02:31 PM on 05-10-2024.

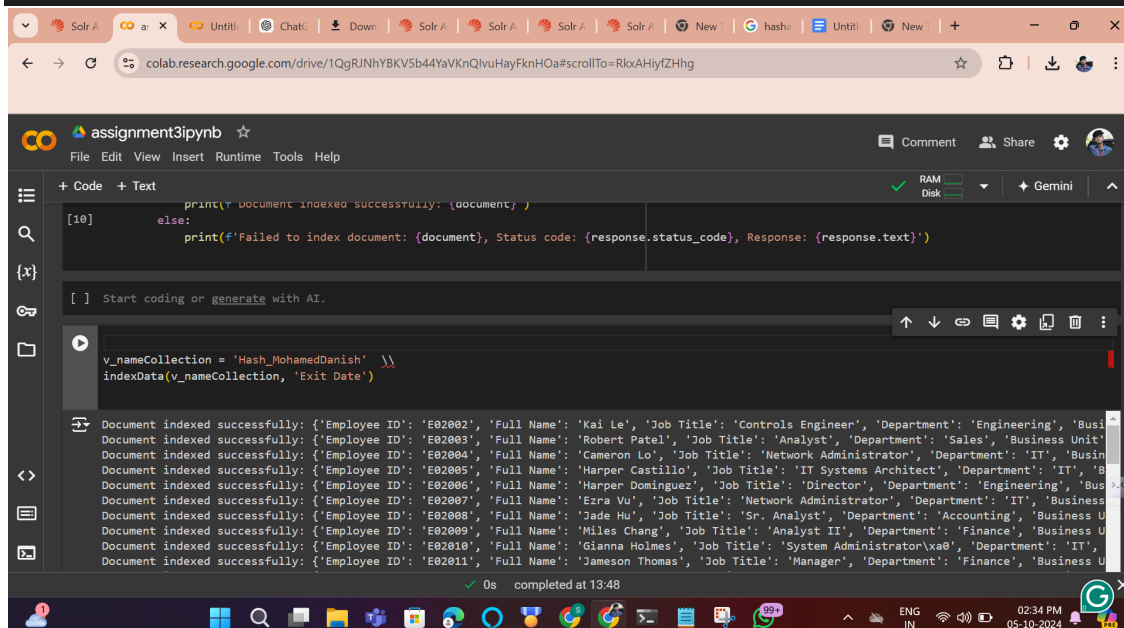
2] INDEXING

```
def indexData(p_collection_name, p_exclude_column):

    file_path = '/content/Employee Sample Data 1.csv'
    try:
        employee_data = pd.read_csv(file_path, encoding='utf-8')
    except UnicodeDecodeError:
        employee_data = pd.read_csv(file_path, encoding='ISO-8859-1') #\
    if p_exclude_column in employee_data.columns:
        employee_data = employee_data.drop(columns=[p_exclude_column])

    for index, row in employee_data.iterrows():
        document = row.to_dict()
        response = requests.post(f'https://7c94-223-181-208-107.ngrok-free.app/solr/{p_collection_name}/update/json/docs',
                                data=json.dumps(document),
                                headers={'Content-Type': 'application/json'})

        if response.status_code == 200:
            print(f'Document indexed successfully: {document}')
        else:
            print(f'Failed to index document: {document}, Status code: {response.status_code}, Response: {response.text}')
```



The screenshot shows a Google Colab notebook titled "assignment3ipynb". The notebook is open to a cell containing the following code:

```
v_nameCollection = 'Hash_MohamedDanish'
indexData(v_nameCollection, 'Exit Date')
```

The output of the code is displayed in the cell, showing 11 documents indexed successfully. Each document is a dictionary containing the following fields: Employee ID, Full Name, Job Title, and Department. The documents are as follows:

- Document indexed successfully: {'Employee ID': 'E02002', 'Full Name': 'Kai Le', 'Job Title': 'Controls Engineer', 'Department': 'Engineering', 'Business Unit': 'Sales'}
- Document indexed successfully: {'Employee ID': 'E02003', 'Full Name': 'Robert Patel', 'Job Title': 'Analyst', 'Department': 'Sales', 'Business Unit': 'Sales'}
- Document indexed successfully: {'Employee ID': 'E02004', 'Full Name': 'Cameron Lo', 'Job Title': 'Network Administrator', 'Department': 'IT', 'Business Unit': 'IT'}
- Document indexed successfully: {'Employee ID': 'E02005', 'Full Name': 'Harper Castillo', 'Job Title': 'IT Systems Architect', 'Department': 'IT', 'Business Unit': 'IT'}
- Document indexed successfully: {'Employee ID': 'E02006', 'Full Name': 'Harper Dominguez', 'Job Title': 'Director', 'Department': 'Engineering', 'Business Unit': 'Engineering'}
- Document indexed successfully: {'Employee ID': 'E02007', 'Full Name': 'Ezra Vu', 'Job Title': 'Network Administrator', 'Department': 'IT', 'Business Unit': 'IT'}
- Document indexed successfully: {'Employee ID': 'E02008', 'Full Name': 'Jade Hu', 'Job Title': 'Sr. Analyst', 'Department': 'Accounting', 'Business Unit': 'Accounting'}
- Document indexed successfully: {'Employee ID': 'E02009', 'Full Name': 'Miles Chang', 'Job Title': 'Analyst II', 'Department': 'Finance', 'Business Unit': 'Finance'}
- Document indexed successfully: {'Employee ID': 'E02010', 'Full Name': 'Gianna Holmes', 'Job Title': 'System Administrator', 'Department': 'IT', 'Business Unit': 'IT'}
- Document indexed successfully: {'Employee ID': 'E02011', 'Full Name': 'Jameson Thomas', 'Job Title': 'Manager', 'Department': 'Finance', 'Business Unit': 'Finance'}

The notebook interface includes a file explorer on the left, a code editor in the center, and a terminal at the bottom showing the execution progress. The terminal output shows "0s completed at 13:48".

3] SEARCH BY COLUMN

```
import requests

def searchByColumn(p_collection_name, p_column_name, p_column_value):
    url =
f'https://f10f-223-181-208-107.ngrok-free.app/solr/{p_collection_name}/select'

    params = {
        'q': f'{p_column_name}:{p_column_value}',
        'wt': 'json'
    }
    response = requests.get(url, params=params)
    return response.json()

search_results = searchByColumn('Hash_MohamedDanish', 'Department', 'IT')
print(search_results)
```

The screenshot shows a Google Colab notebook interface. The top bar includes the Colab logo, the notebook name 'assignment3ipynb', and various icons for file management, runtime, and help. The main area displays a Python script that uses the 'requests' library to send a GET request to a Solr endpoint. The script defines a function 'searchByColumn' that takes a collection name, column name, and column value as arguments. It constructs a URL and a JSON query parameter, then returns the response as JSON. The script is executed, and the output is displayed as a JSON object containing employee information.

```
import requests

def searchByColumn(p_collection_name, p_column_name, p_column_value):
    url = f'https://f10f-223-181-208-107.ngrok-free.app/solr/{p_collection_name}/select'
    params = {
        'q': f'{p_column_name}:{p_column_value}',
        'wt': 'json'
    }
    response = requests.get(url, params=params)
    return response.json()

search_results = searchByColumn('Hash_MohamedDanish', 'Department', 'IT')
print(search_results)
```

Output:

```
{
  "response": {
    "numFound": 1,
    "start": 0,
    "docs": [
      {
        "Employee_ID": "E02077",
        "Full_Name": "Eva Chavez",
        "Job_Title": "IT Systems Architect",
        "Business_Unit": "Specialty Products",
        "Hire_Date": "2017-01-01"
      }
    ]
  }
}
```

The bottom status bar indicates the notebook is completed at 13:48.

4] GET EMP COUNT

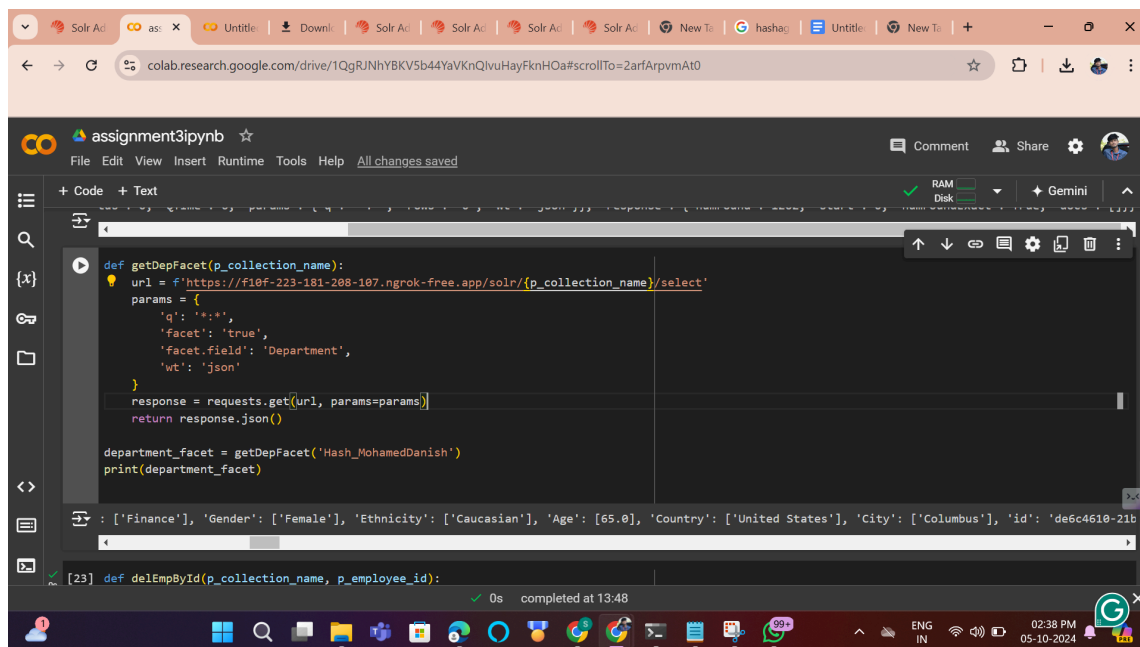
```
def getEmpCount(p_collection_name):  
    url =  
f'https://f10f-223-181-208-107.ngrok-free.app/solr/{p_collection_name}/select'  
    params = {  
        'q': '*:~',  
        'rows': 0,  
        'wt': 'json'  
    }  
    response = requests.get(url, params=params)  
    return response.json()  
  
employee_count = getEmpCount('Hash_MohamedDanish')  
print(employee_count)
```

The screenshot shows a Google Colab notebook titled 'assignment3ipynb'. The code cell contains the same Python code as shown in the previous block. The output cell displays the result of the function call: a JSON object with fields like 'numFound', 'start', 'numFoundExact', and 'docs'. The status bar at the bottom indicates the code was completed at 13:48.

5] GET DEPT FACET

```
def getDepFacet(p_collection_name):  
    url =  
f'https://f10f-223-181-208-107.ngrok-free.app/solr/{p_collection_name}/select'  
  
    params = {  
        'q': '*:*',  
        'facet': 'true',  
        'facet.field': 'Department',  
        'wt': 'json'  
    }  
  
    response = requests.get(url, params=params)  
    return response.json()  
  
department_facet = getDepFacet('Hash_MohamedDanish')  
print(department_facet)
```

Output



The screenshot displays a Google Colab notebook interface. The top toolbar includes options for File, Edit, View, Insert, Runtime, Tools, and Help, along with status indicators for RAM and Disk usage, and a Gemini icon. The code editor shows the same Python script as in the previous block. Below the code, the output of the script is visible, showing a JSON response with a list of departments: ['Finance'], ['Female'], ['Caucasian'], [65.0], ['United States'], ['Columbus'], and a unique identifier 'de6c4618-21b'. The bottom status bar indicates that the code was executed successfully at 13:48 on 05-10-2024.

```
def getDepFacet(p_collection_name):  
    url =  
f'https://f10f-223-181-208-107.ngrok-free.app/solr/{p_collection_name}/select'  
  
    params = {  
        'q': '*:*',  
        'facet': 'true',  
        'facet.field': 'Department',  
        'wt': 'json'  
    }  
  
    response = requests.get(url, params=params)  
    return response.json()  
  
department_facet = getDepFacet('Hash_MohamedDanish')  
print(department_facet)
```

```
{  
  "facet_counts": {  
    "facet_fields": {  
      "Department": [  
        "Finance",  
        "Female",  
        "Caucasian",  
        "65.0",  
        "United States",  
        "Columbus",  
        "de6c4618-21b"  
      ]  
    }  
  },  
  "response": {  
    "status": "OK",  
    "max_score": 1.0,  
    "docs": []  
  }  
}
```

[23] def delEmpById(p_collection_name, p_employee_id):
0s completed at 13:48

6] DELETE EMPLOYEE

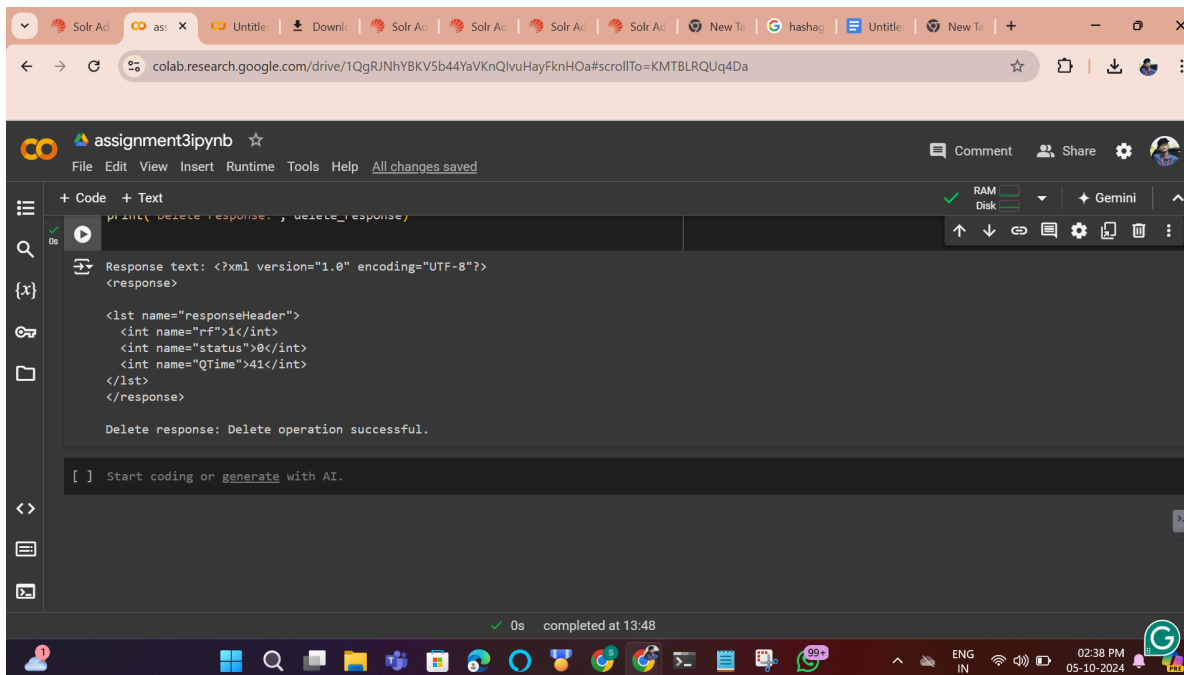
```
def delEmpById(p_collection_name, p_employee_id):
    url =
f'https://f10f-223-181-208-107.ngrok-free.app/solr/{p_collection_name}/update?commit=true'

    data = f'<delete><id>{p_employee_id}</id></delete>'
    headers = {'Content-Type': 'text/xml'}
    response = requests.post(url, data=data, headers=headers)

    print("Response text:", response.text)
    if response.status_code == 200:
        if '<int name="status">0</int>' in response.text:
            return "Delete operation successful."
        else:
            return "Delete operation failed."
    else:
        return f"Error: {response.status_code} - {response.text}"

delete_response = delEmpById('Hash_MohamedDanish', 'E02002')
print("Delete response:", delete_response)
```

Output



The screenshot shows a Google Colab notebook interface. The top bar displays the notebook name 'assignment3ipynb' and various icons for file management, runtime, and sharing. The left sidebar contains icons for code, text, and other notebook elements. The main area shows a code cell with the following output:

```
print(delete_response, delete_response)

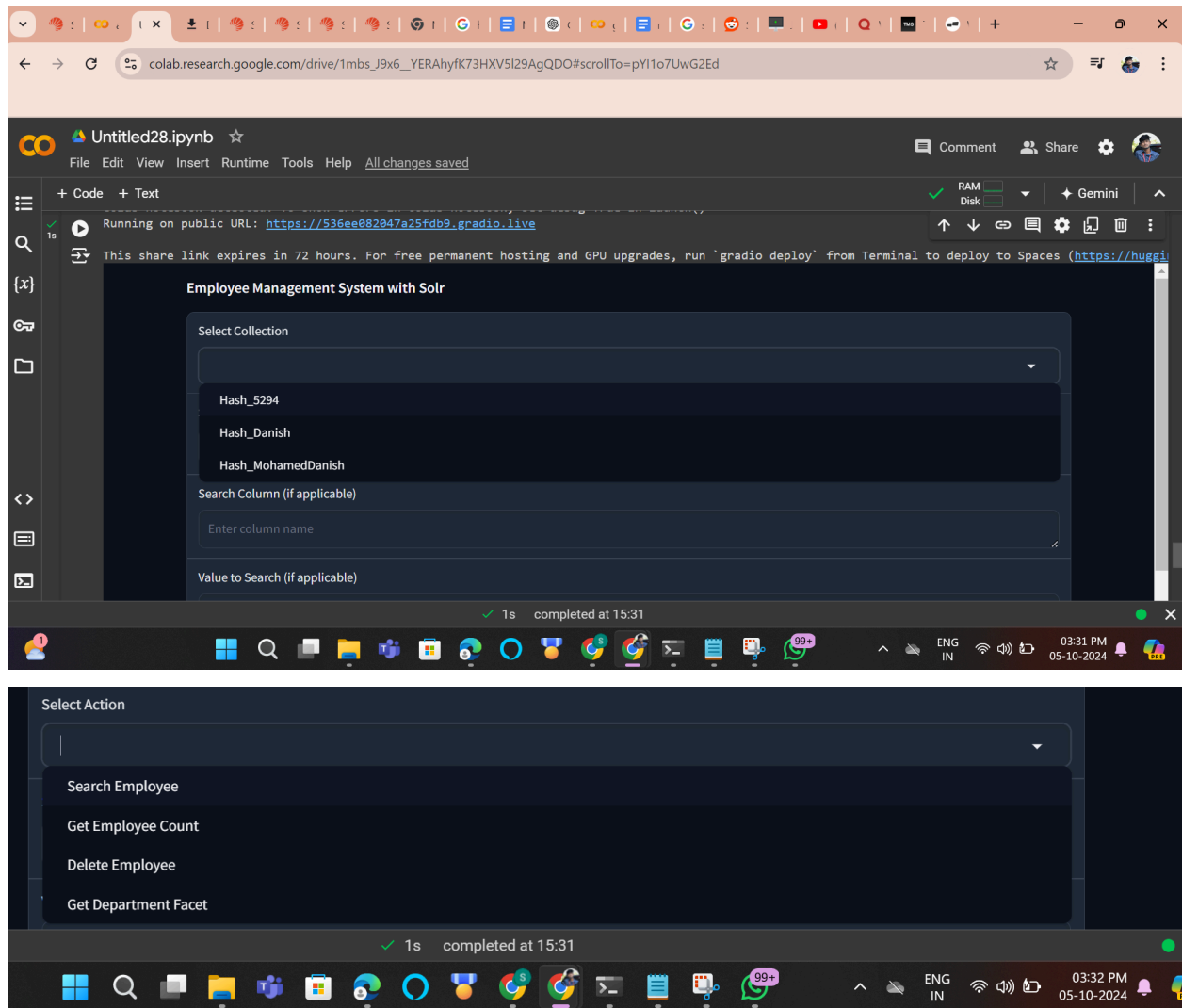
Response text: <?xml version="1.0" encoding="UTF-8"?>
<response>

<lst name="responseHeader">
  <int name="nf">1</int>
  <int name="status">0</int>
  <int name="QTime">41</int>
</lst>
</response>

Delete response: Delete operation successful.
```

At the bottom of the notebook, there is a status bar indicating '0s completed at 13:48'. The Windows taskbar is visible at the very bottom, showing the time as 02:38 PM on 05-10-2024.

8] CREATED CUSTOM UI using gradio



Previously created gradio and a dashboard with aggregations like average salary, average age, max country, no. employees each dept.

Data fetched using requests from solr

Code

```
import gradio as gr
import requests
import numpy as np
```

```

import matplotlib.pyplot as plt
import pandas as pd

def get_departments():
    ngrok_url = 'https://ac9d-106-222-122-100.ngrok-free.app'
    query_url =
f"{ngrok_url}/solr/employeess/terms?terms.fl=Department&terms.limit=-1&wt=
json"

    try:
        response = requests.get(query_url)
        response.raise_for_status()
        data = response.json()
        departments = data["terms"]["Department"]
        return departments if departments else []
    except Exception as e:
        print(f"Error fetching departments: {str(e)}")
        return []

def search_employees(department):
    ngrok_url = 'https://ac9d-106-222-122-100.ngrok-free.app'
    query_url =
f"{ngrok_url}/solr/employeess/select?q=Department:\"{department}\"&wt=json
&rows=100"

    try:
        response = requests.get(query_url)
        response.raise_for_status()

        data = response.json()

        if data["response"]["numFound"] == 0:
            return "No employees found for this department.", None

        employee_data = []
        for doc in data["response"]["docs"]:
            employee_data.append({
                "Job Title": doc['Job_Title'][0],
                "Salary": float(doc['Annual_Salary'][0].replace('$',
''').replace(',', ' ').strip()),

```

```

        "Age": doc['Age'][0],
        "Country": doc['Country'][0]
    })

df = pd.DataFrame(employee_data)

avg_salary = df["Salary"].mean()
avg_age = df["Age"].mean()
total_employees = len(df)
max_country = df["Country"].value_counts().idxmax()

plt.figure(figsize=(12, 6))

plt.subplot(2, 2, 1)
plt.bar(['Average Salary'], [avg_salary], color='skyblue')
plt.ylabel('Salary ($)')
plt.title('Average Salary')

plt.subplot(2, 2, 2)
plt.bar(['Average Age'], [avg_age], color='lightgreen')
plt.ylabel('Age (years)')
plt.title('Average Age')

plt.subplot(2, 2, 3)
plt.bar(['Total Employees'], [total_employees], color='salmon')
plt.ylabel('Count')
plt.title('Total Employees')

plt.subplot(2, 2, 4)
plt.bar([max_country],
[ df["Country"].value_counts()[max_country]], color='orange')
plt.ylabel('Count')
plt.title('Max Employees Country')

plt.tight_layout()
plt.savefig("employee_statistics.png")
plt.close()

return (f"Average Salary: ${avg_salary:,.2f}\n"
        f"Average Age: {avg_age:.2f} years\n")

```

```

        f"Total Employees: {total_employees}\n"
        f"Max Employees Country: {max_country}"),
"employee_statistics.png"

except Exception as e:
    return f"An error occurred: {str(e)}", None

with gr.Blocks() as interface:
    gr.Markdown("# Employee Analytics Dashboard")
    gr.Markdown("### Select a department to see analytics.")

    department_dropdown = gr.Dropdown(
        label="Select Department",
        choices=get_departments(),
        value=None
    )

    submit_btn = gr.Button("Get Analytics")

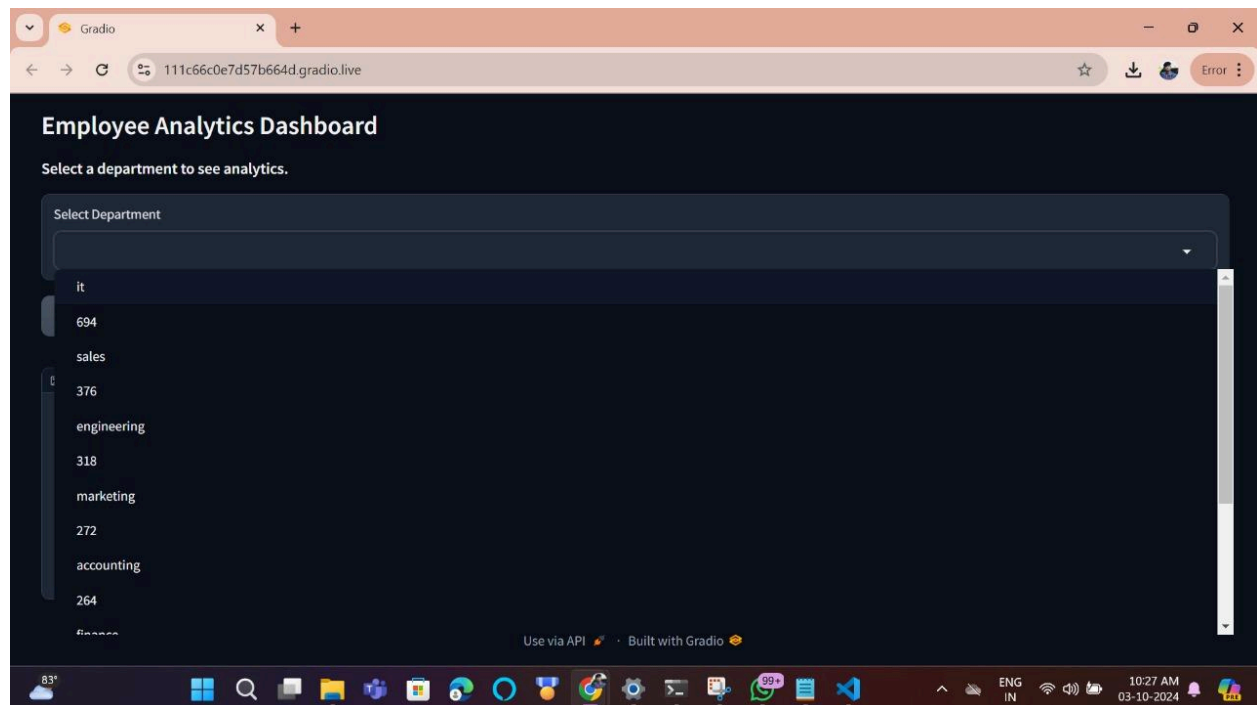
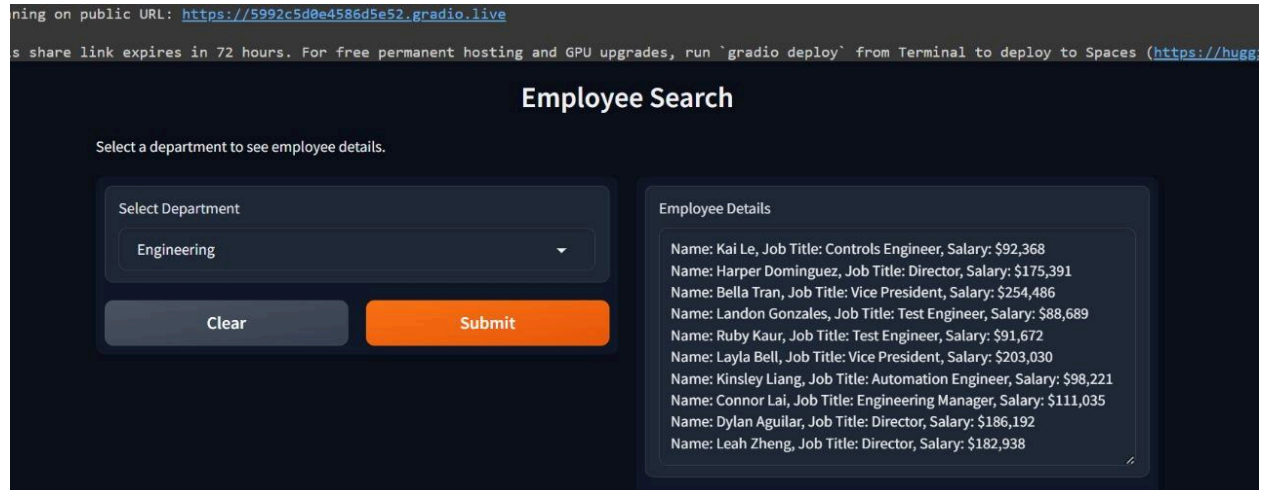
    output_text = gr.Markdown(label="Analytics Results",
elem_id="analytics-output")
    stats_chart = gr.Image(label="Employee Statistics Chart")

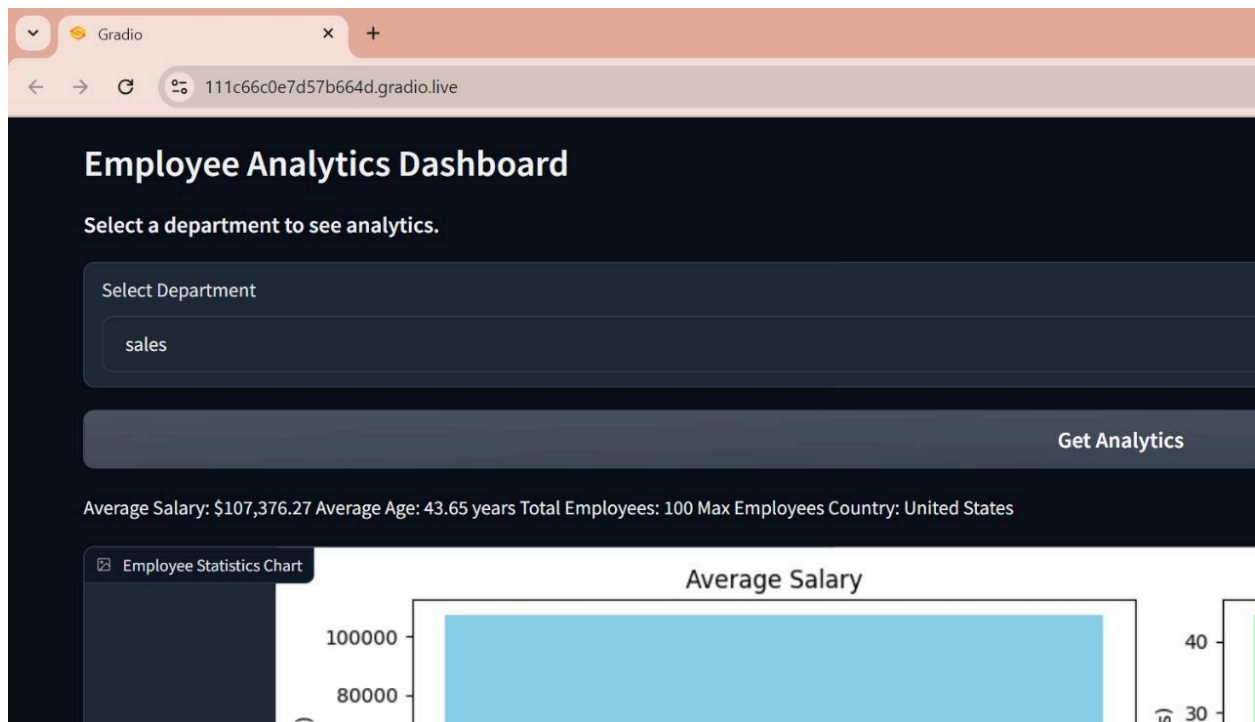
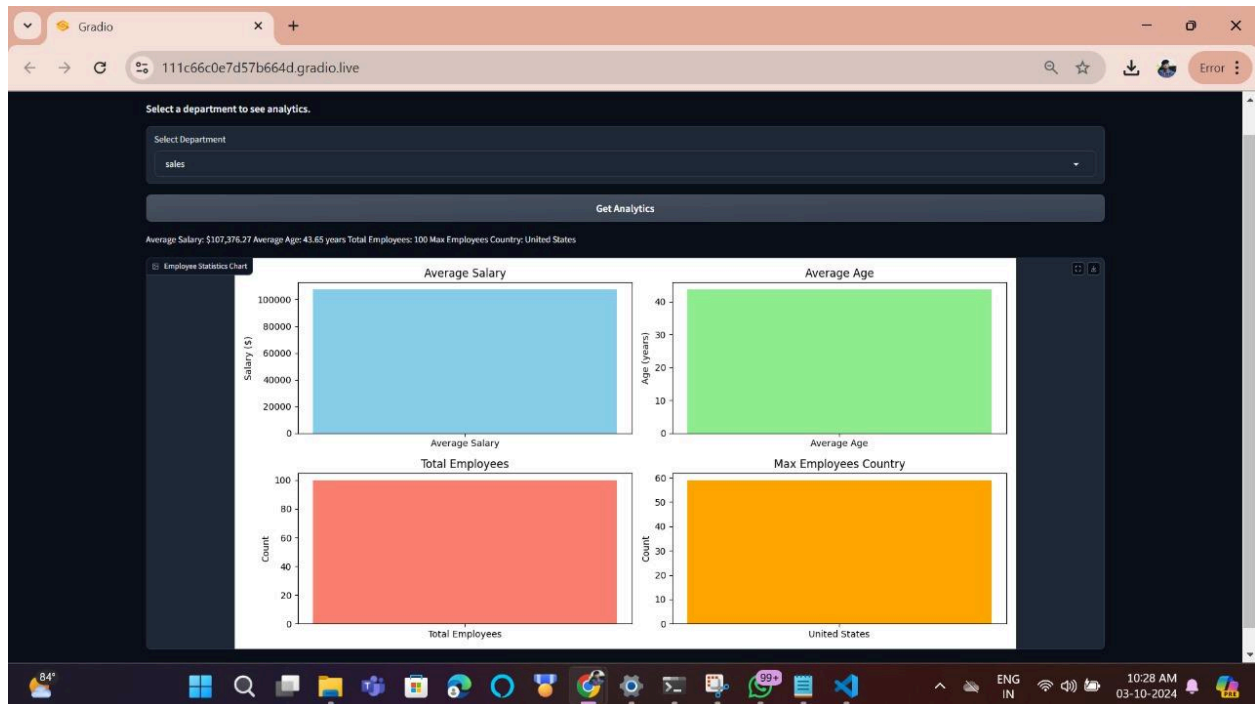
    submit_btn.click(fn=search_employees, inputs=department_dropdown,
outputs=[output_text, stats_chart])

interface.launch(share=True)

```

OUTPUT





Dockerizing My Gradio Project

To dockerize my Gradio application, I created a Dockerfile and built the Docker image

Step 1: Created a Dockerfile

- I created a new file named `Dockerfile`

Added the Following Code to the Dockerfile:

```
FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY employee_dashboard.py .
EXPOSE 7860
CMD ["python", "employee_dashboard.py"]
```

Step 2: Created a requirements.txt File

- I created a `requirements.txt` file in the same directory.

Added the Required Libraries:

```
gradio
requests
numpy
matplotlib
pandas
```

Step 3: Built the Docker Image

- I opened a terminal in the directory where my Dockerfile and Python script were located.

Ran the Following Command to Build the Docker Image:

```
docker build -t employee_dashboard .
```

Step 4: Ran the Docker Container

Once the image was built, I ran the container with this command:

```
docker run -p 7860:7860 employee_dashboard
```