Practical :- 11

Aim: Write a program to illustrate the working of jaql.

Description:

This project demonstrates the simulation of JAQL (JavaScript Object Notation Query Language) functionality using a Python script that processes a JSON file. JAQL is typically used for querying semi-structured data in Big Data environments. However, due to the deprecation of native JAQL engines like IBM BigInsights, this simulation provides a modern and practical way to understand JAQL-like transformations using familiar tools such as Python.

A JSON file named employees.json is created, containing an array of employee records with fields like name, age, and dept. The Python script reads this file, filters out employees from the "IT" department, and transforms the output to only include their name and age. The result is then saved in another file named output.json.

This program mimics a JAQL pipeline consisting of read, filter, and transform operations, making it a useful exercise for understanding data flow in a functional style.

Step 1: Create JSON file

"C:\Users\DELL\Desktop\Smita\employees.json"

employees.json:

```
[
  { "name": "Alice", "age": 30, "dept": "IT" },
  { "name": "Bob", "age": 45, "dept": "HR" },
  { "name": "Charlie", "age": 25, "dept": "IT" },
  { "name": "David", "age": 35, "dept": "Finance" }
]
```

Step 2: Write the Python (JAQL-style) Script

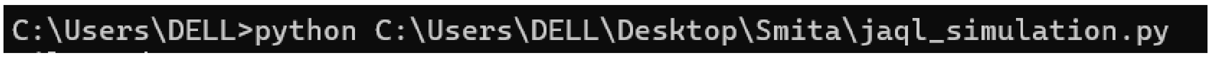
"C:\Users\DELL\Desktop\Smita\jaql_simulation.py"

Python code: jaql_simulation.py

```
import json
# Load JSON data
with open("C:\\Users\\DELL\\Desktop\\Smita\\employees.json", "r") as f:
   data = json.load(f)
# JAQL-like filter and transform
result = [
   {"name": emp["name"], "age": emp["age"]}
```

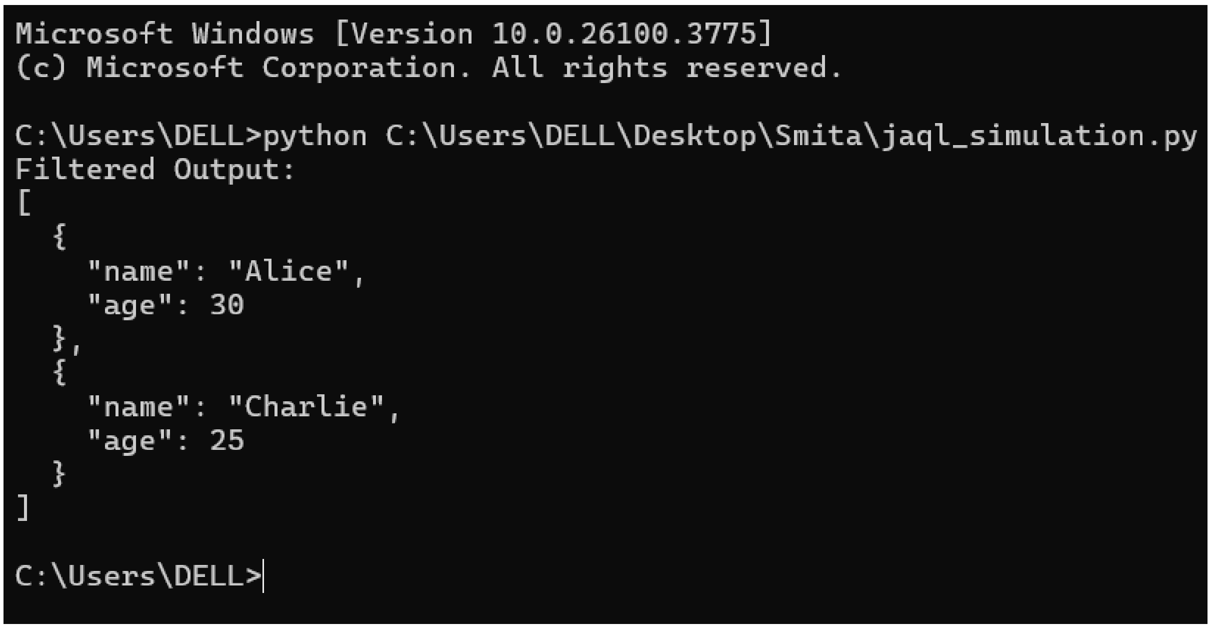
```
    for emp in data if emp["dept"] == "IT"
]
# Save result
with open("C:\\Users\\DELL\\Desktop\\Smita\\output.json", "w") as f:
    json.dump(result, f, indent=2)
# Print the result
print("Filtered Output:")
print(json.dumps(result, indent=2))
```

Step 3: Run the Script in Command Prompt



```
C:\Users\DELL>python C:\Users\DELL\Desktop\Smita\jaql_simulation.py
```

Output:



```
Microsoft Windows [Version 10.0.26100.3775]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL>python C:\Users\DELL\Desktop\Smita\jaql_simulation.py
Filtered Output:
[
  {
    "name": "Alice",
    "age": 30
  },
  {
    "name": "Charlie",
    "age": 25
  }
]

C:\Users\DELL>
```

Learnings:

This project helped understand how JAQL-like queries can be simulated using Python. It demonstrated reading, filtering, and transforming JSON data effectively. Even without a JAQL engine, Python provides a simple way to practice the core concepts of querying semi-structured data, making it a good starting point for learning Big Data tools.