**Exp No: 6**

# AIM:

## Handling JSON data using HDFS and Python

To handle JSON data using HDFS and python.

# PROCEDURE:

## Step 1: Create json file on bash & save as emp.json

nano emp.json ; Paste the below content on it [

{"name": "John Doe", "age": 30, "department": "HR", "salary": 50000},

{"name": "Jane Smith", "age": 25, "department": "IT", "salary": 60000},

{"name": "Alice Johnson", "age": 35, "department": "Finance", "salary": 70000},

{"name": "Bob Brown", "age": 28, "department": "Marketing", "salary": 55000},

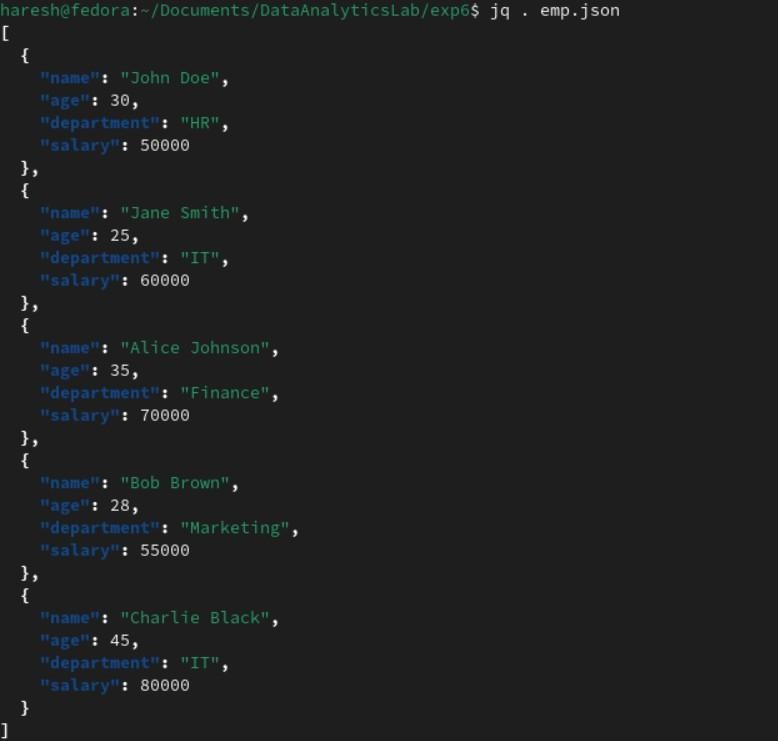
{"name": "Charlie Black", "age": 45, "department": "IT", "salary": 80000}

]



## Step 2 : Check json is readable or any error by giving

install jq by sudo apt-get install jq hadoop@Ubuntu:~$ jq . emp.json



## Step 3: Install pandas and hdfs dependencies for python.

**Step 4: Create process\_data.py file** from hdfs import InsecureClient import pandas as pd

import json

# Connect to HDFS

hdfs\_client = InsecureClient('http://localhost:9870', user='hdfs')

# Read JSON data from HDFS try:

with hdfs\_client.read('/home/hadoop/emp.json', encoding='utf-8') as reader:

json\_data = reader.read() # Read the raw data as a string if not json\_data.strip(): # Check if data is empty

raise ValueError("The JSON file is empty.")

print(f"Raw JSON Data: {json\_data[:1000]}") # Print first 1000 characters for debugging data = json.loads(json\_data) # Load the JSON data

except json.JSONDecodeError as e: print(f"JSON Decode Error: {e}") exit(1)

except Exception as e:

print(f"Error reading or parsing JSON data: {e}") exit(1)

# Convert JSON data to DataFrame try:

df = pd.DataFrame(data) except ValueError as e:

print(f"Error converting JSON data to DataFrame: {e}") exit(1)

# Projection: Select only 'name' and 'salary' columns projected\_df = df[['name', 'salary']]

# Aggregation: Calculate total salary total\_salary = df['salary'].sum()

# Count: Number of employees earning more than 50000 high\_earners\_count = df[df['salary'] > 50000].shape[0]

# Limit: Get the top 5 highest earners top\_5\_earners = df.nlargest(5, 'salary')

# Skip: Skip the first 2 employees skipped\_df = df.iloc[2:]

# Remove: Remove employees from a specific department filtered\_df = df[df['department'] != 'IT']

# Save the filtered result back to HDFS filtered\_json = filtered\_df.to\_json(orient='records') try:

with hdfs\_client.write('/home/hadoop/filtered\_employees.json', encoding='utf-8', overwrite=True) as writer:

writer.write(filtered\_json)

print("Filtered JSON file saved successfully.") except Exception as e:

print(f"Error saving filtered JSON data: {e}") exit(1)

# Print results

print(f"Projection: Select only name and salary columns") print(f"{projected\_df}")

print(f"Aggregation: Calculate total salary")

print(f"Total Salary: {total\_salary}") print(f"\n")

print(f"# Count: Number of employees earning more than 50000")

print(f"Number of High Earners (>50000): {high\_earners\_count}")

print(f"\n")

print(f"limit Top 5 highest salary")

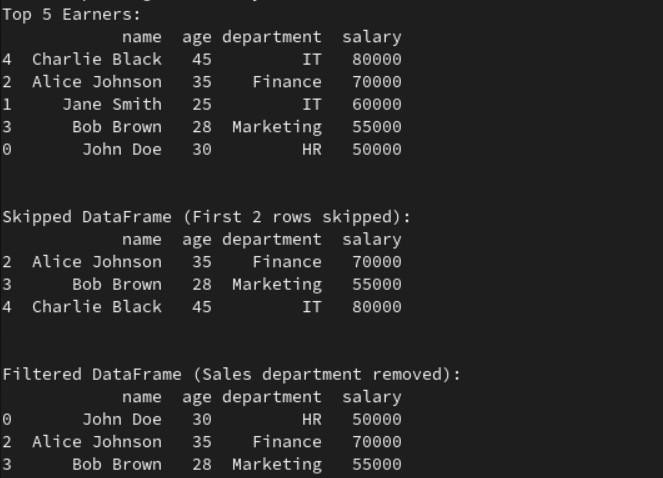
print(f"Top 5 Earners: \n{top\_5\_earners}") print(f"\n")

print(f"Skipped DataFrame (First 2 rows skipped): \n{skipped\_df}") print(f"\n")

print(f"Filtered DataFrame (Sales department removed): \n{filtered\_df}")

## Step 5: run the file by

bash: python3 process\_data.py



# RESULT:

Experiment has been successfully executed and output has been verified.