Reg. No: 210701303

Ex No: 6

Import a JASON file from the command line. Apply the following actions with the data present in the JASON file where, projection, aggregation, remove, count, limit, skip and sort

AIM:

To import a JASON file from the command line and apply the following actions with the data present in the JASON file where, projection, aggregation, remove, count, limit, skip and sort.

PROCEDURE:

PROCEDURE:

Step 1: Install Required Packages

Install the necessary packages using pip: \$ pip install pandas --break-system-packages

Step 2: Verify Package Installation

Verify that the required packages are installed:

\$ python

- >>> import pandas as pd
- >>> from hdfs import InsecureClient
- >>> print("Pandas version:", pd._version_)
- >>> client = InsecureClient('http://localhost:9870', user='hadoop')
- >>> print("HDFS status:", client.status('/'))
- >>> exit()

Step 3: Create process data.py File

Create the Python script for processing data:

\$ nano process_data.py

Paste the following code into the 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

trv

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

json_data = reader.read()

if not json_data.strip():

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

Reg. No: 210701303

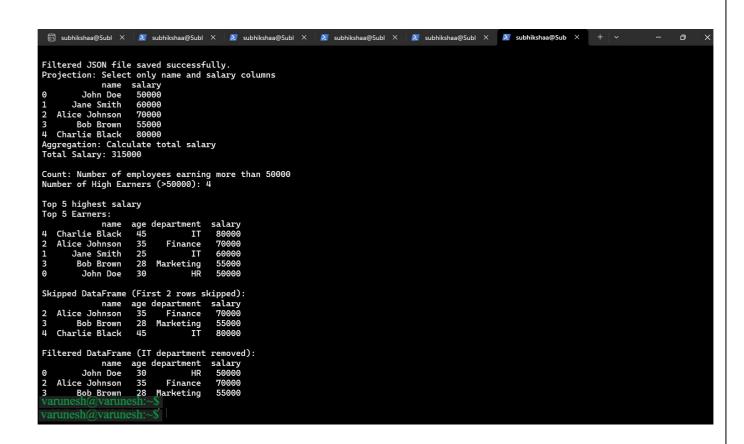
```
data = json.loads(json\_data)
except Exception as e:
  print(f"Error reading or parsing JSON data: {e}")
  exit(1)
# Convert JSON data to DataFrame
df = pd.DataFrame(data)
# Projection: Select 'name' and 'salary'
projected_df = df[['name', 'salary']]
# Aggregation: Calculate total salary
total_salary = df['salary'].sum()
# Count: Employees earning more than 50000
high_earners_count = df[df['salary'] > 50000].shape[0]
# Limit: Top 5 highest earners
top_5_earners = df.nlargest(5, 'salary')
# Skip: Skip the first 2 employees
skipped_df = df.iloc[2:]
# Remove: Filter out employees from IT department
filtered_df = df[df['department'] != 'IT']
# Save the filtered data 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)
except Exception as e:
  print(f"Error saving filtered JSON data: {e}")
Step 4: Run the Script
       Execute the script to process the data:
       $ python3 process_data.py
```

Reg. No: 210701303

Step 5: To view the output

hadoop@Ubuntu:~/Documents\$ hdfs dfs -cat /home/hadoop/emp.json

```
🗟 subhikshaa@Subl 🗴 🙎 subhikshaa@Subl 🗴 🔰 subhikshaa@Subl 🗴 🔊 subhikshaa@Subl 🗴 🔻 subhikshaa@Subl 🗴 🛣 subhikshaa@Subl 🗡
 varunesh@varunesh:~$ python3 processdata.py
1
Filtered JSON file saved successfully.
Projection: Select only name and salary columns
name salary
0 John Doe 50000
1 Jane Smith 60000
    John Doe
Jane Smith
Alice Johnson
                         70000
55000
   Bob Brown
Charlie Black
                         80000
Aggregation: Calculate total salary
Total Salary: 315000
Count: Number of employees earning more than 50000 Number of High Earners (>50000): 4
Top 5 highest salary
Top 5 Earners:
                       age department salary
45 IT 80000
35 Finance 70000
                name
4
2
1
    Charlie Black
    Alice Johnson
Jane Smith
Bob Brown
John Doe
                         35
25
                                  Finance
IT
                               Marketing
HR
3
                          28
                                                55000
                         30
Skipped DataFrame (First 2 rows skipped):
   name age department salary
Alice Johnson 35 Finance 70000
```



	Reg. No: 210701303
RESULT:	
Thus to import a JASON file from the command line and apply the following data present in the JASON file where, projection, aggregation, remove, count, limit, been executed and verified successfully.	