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:

1. **Required Packages Installation**
   - **Install Pandas**
     Pandas is required for manipulating and analyzing data.
     Installation:
     ✦ pip install pandas
   - **Install HDFS**
     HDFS provides a Python interface to interact with Hadoop Distributed File System (HDFS).
     Installation:
     ✦ pip install hdfs
   - **Optional Packages**
     These packages may help when working with large datasets or different formats: ○ **PyArrow** (for Apache Arrow support):
     ✦ pip install pyarrow ○ **HDFS3** (alternative to HDFS):
     ✦ pip install hdfs3

2. **Create a json file (for example: emp.json) with the following content:**

```
[
  {"name": "Alice", "salary": 60000, "department": "HR"},
  {"name": "Bob", "salary": 55000, "department": "Finance"},
  {"name": "Charlie", "salary": 70000, "department": "IT"},
  {"name": "David", "salary": 45000, "department": "Sales"},
  {"name": "Eve", "salary": 80000, "department": "IT"}
]
```

3. **Copy the json file to the hdfs directory using the command:**

```
$ hdfs dfs copyFromLocal  /path/to/emp.json /home/hadoop
```

Also give the necessary permissions if not already given using the command:

```
$ hdfs dfs -chmod 777 /home/hadoop
```

4. **Python Script: process_data.py**

The following script reads a JSON file from HDFS, processes it using Pandas, and performs several operations such as projection, aggregation, counting, limiting, skipping, and filtering.

#process_data.py

```
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 (e.g., 'Sales') 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)
```

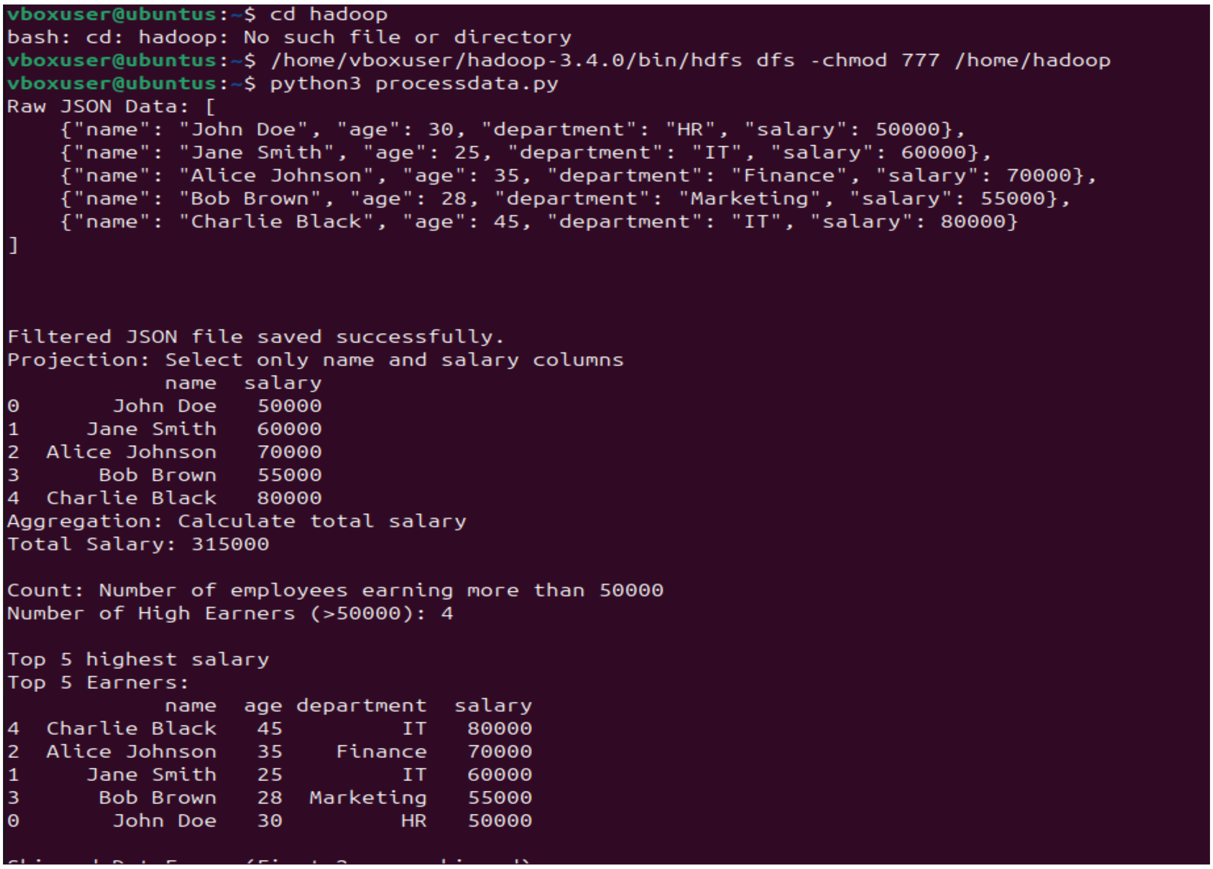
```
   print("Filtered JSON file saved successfully.")
except Exception as e:     print(f"Error saving
filtered JSON data: {e}")     exit(1)
# Print results
print(f"Projection: Select only name and salary columns\n{projected_df}") print(f"Aggregation:
Total Salary: {total_salary}")
print(f"Number of High Earners (>50000): {high_earners_count}")
print(f"Top 5 Earners: \n{top_5_earners}") print(f"Skipped
DataFrame (First 2 rows skipped): \n{skipped_df}") print(f"Filtered
DataFrame (IT department removed): \n{filtered_df}")
```

5. Run the Script

Execute the Python script by running the following command in your terminal:

```
python3 process_data.py
```

Output:



```
vboxuser@ubuntus:~$ cd hadoop
bash: cd: hadoop: No such file or directory
vboxuser@ubuntus:~$ /home/vboxuser/hadoop-3.4.0/bin/hdfs dfs -chmod 777 /home/hadoop
vboxuser@ubuntus:~$ python3 processdata.py
Raw JSON Data: [
    {"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}
]



Filtered JSON file saved successfully.
Projection: Select only name and salary columns
            name  salary
0       John Doe   50000
1     Jane Smith   60000
2  Alice Johnson   70000
3      Bob Brown   55000
4  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:
            name  age department  salary
4  Charlie Black   45         IT   80000
2  Alice Johnson   35    Finance   70000
1     Jane Smith   25         IT   60000
3      Bob Brown   28  Marketing   55000
0       John Doe   30         HR   50000

Skipped DataFrame (First 2 rows skipped):
```

Result:

Thus 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 have been executed and verified successfully.