

## **Import Json file and do projection, aggregation, limit,count ,skip and remove using python and hdfs.**

### **Aim:**

To import Json file and do projection, aggregation, limit,count ,skip and remove using python and hdfs.

### **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: put the employees.json local directory to home/hadoop directory**

#### **Step 3: Install Required Packages**

Open your terminal or command prompt and run the following commands to install the required Python packages.

**pip install pandas**

**pip install hdfs**

#### **Step 4: Verify Installation**

Test the package installations by running the following Python commands in a Python shell or a script:

```
import pandas as pd  
  
from hdfs import InsecureClient  
  
# Check pandas version  
print("Pandas version:", pd.__version__)
```

```
# Test HDFS client connection
```

```
client = InsecureClient('http://localhost:9870', user='hadoop')
```

```
print("HDFS status:", client.status('/'))
```

This will print the version of Pandas installed and confirm whether the HDFS connection is successful.

#### **Step 5: Create the process\_data.py File**

Create a new Python file named process\_data.py and add the following code to it:

```
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"# Count: Number of employees earning more than 50000")

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}")
```

#### **Step 6: Run the process\_data.py Script**

Run the script in your terminal or command prompt by typing the following command:

```
python3 process_data.py
```

Make sure your HDFS is up and running, and the /home/hadoop/emp.json file exists on your HDFS.

The script will read the JSON file from HDFS, process the data, and save the filtered results back to HDFS.

#### **Step 7: Check the output.**

## OUTPUT:

```
Activities Terminal Sep 17 19:30 user@Ubuntu: ~
user@ubuntu:~$ python3 process_data.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
Projection:
  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

Limit: 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):
  name age department salary
2 Alice Johnson 35 Finance 70000
3 Bob Brown 28 Marketing 55000
4 Charlie Black 45 IT 80000

Filtered DataFrame (IT department removed):
  name age department salary
0 John Doe 30 HR 50000
2 Alice Johnson 35 Finance 70000
3 Bob Brown 28 Marketing 55000
user@ubuntu:~$
```

## Result:

Thus the program to import Json file and to do projection, aggregation, limit, count, skip and remove using python and hdf5 is executed successfully.