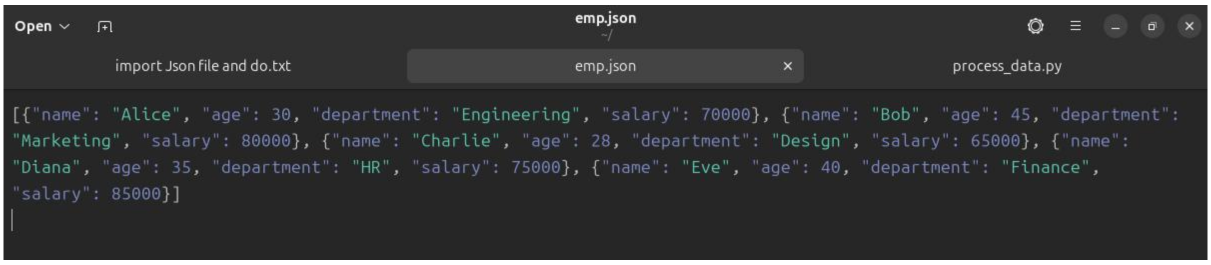
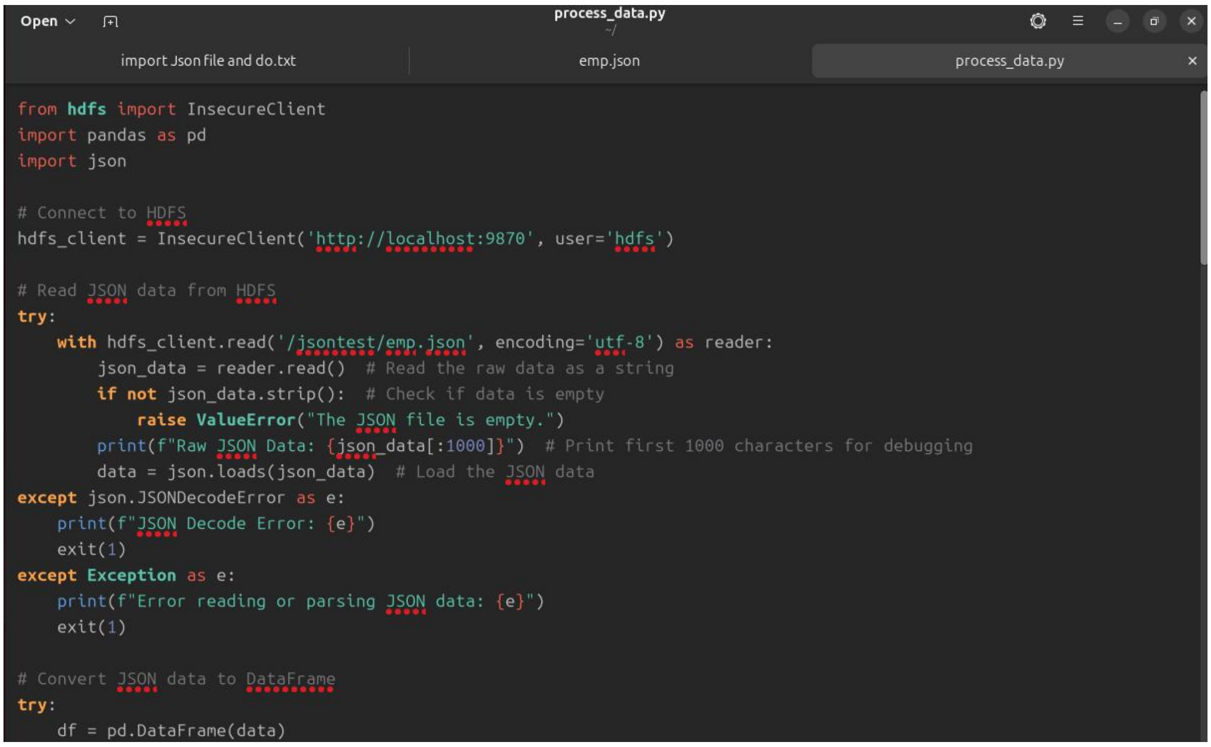
Importing JSON File and perform various Operations using HDFS and Python

emp.json:

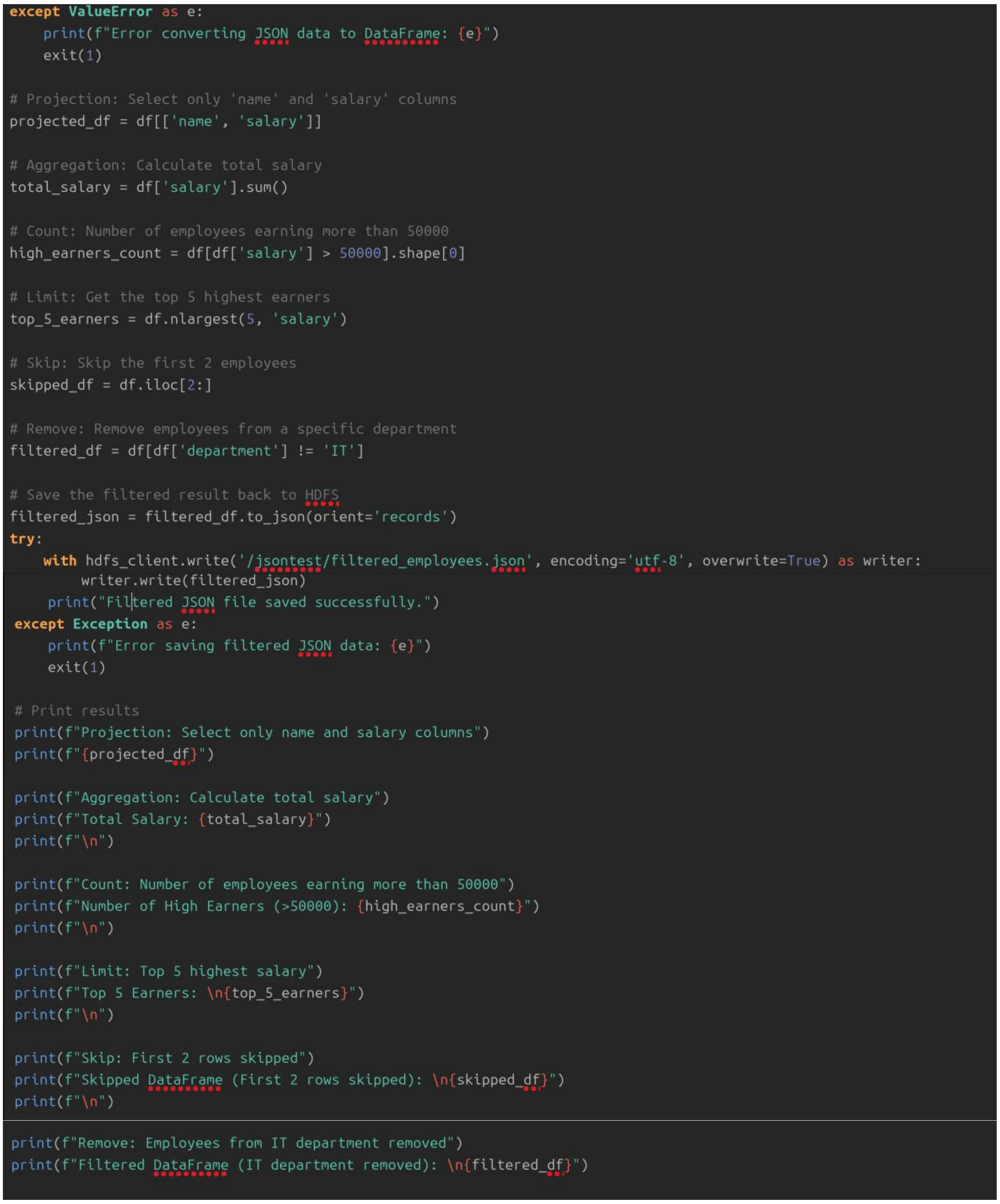


```
[{"name": "Alice", "age": 30, "department": "Engineering", "salary": 70000}, {"name": "Bob", "age": 45, "department":
"Marketing", "salary": 80000}, {"name": "Charlie", "age": 28, "department": "Design", "salary": 65000}, {"name":
"Diana", "age": 35, "department": "HR", "salary": 75000}, {"name": "Eve", "age": 40, "department": "Finance",
"salary": 85000}]
```

process_data.py:



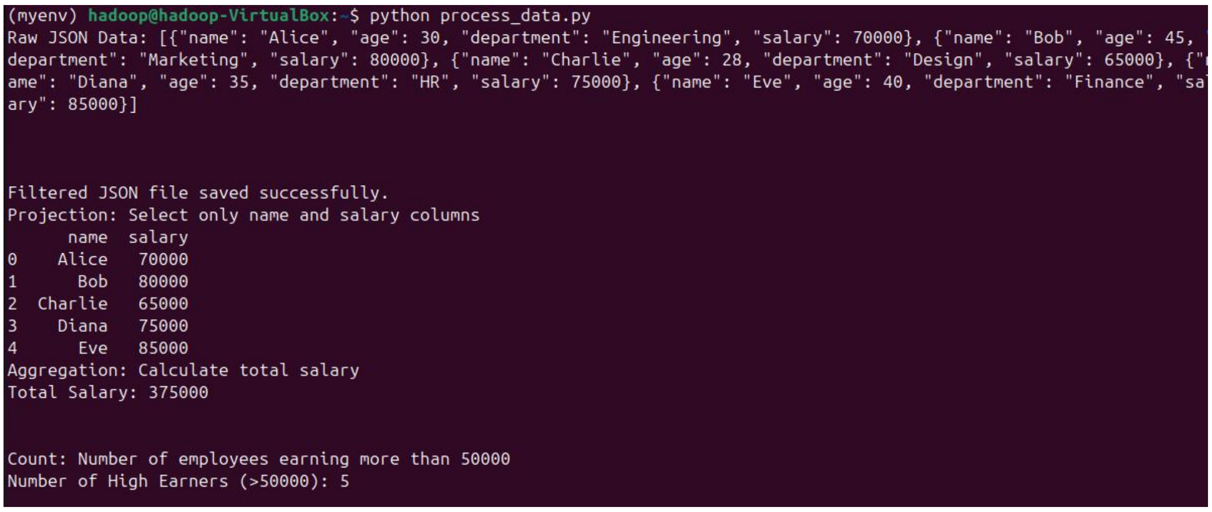
```python
from hdfs import InsecureClient
import pandas as pd
import json

# Connect to HDFS
hdfs_client = InsecureClient('http://localhost:9870', user='hdfs')

# Read JSON data from HDFS
try:
    with hdfs_client.read('/jsontest/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)
```



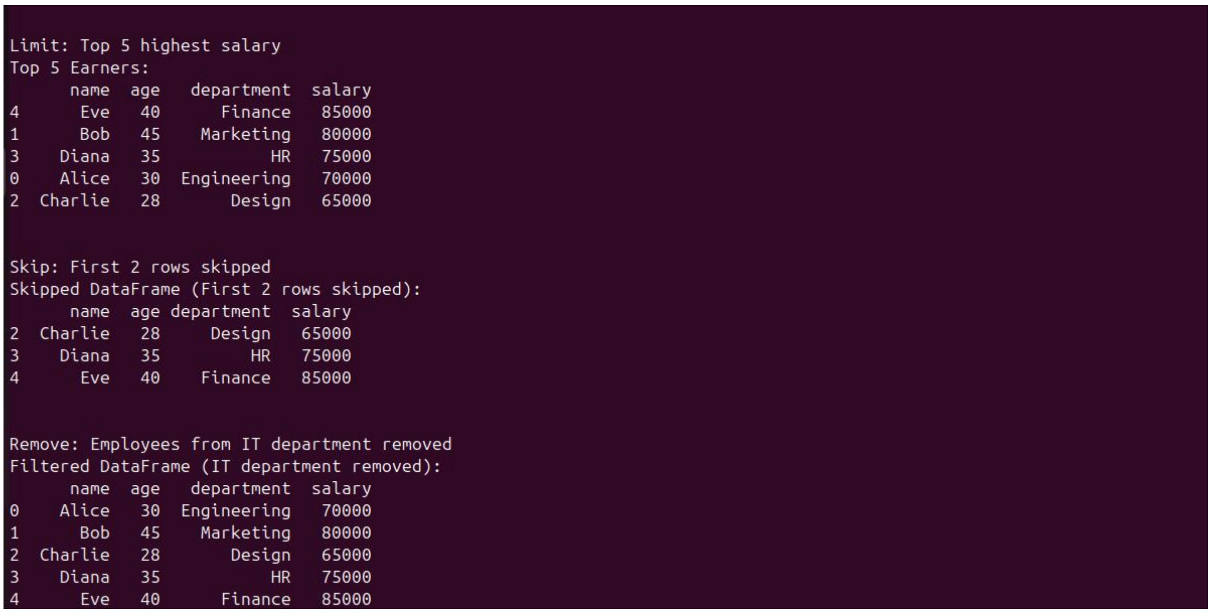
```python
except ValueError as e:
    print(f"Error converting JSON data to DataFrame: {e}")
    exit(1)

# 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('/jsontest/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"Skip: First 2 rows skipped")
print(f"Skipped DataFrame (First 2 rows skipped): \n{skipped_df}")
print(f"\n")

print(f"Remove: Employees from IT department removed")
print(f"Filtered DataFrame (IT department removed): \n{filtered_df}")
```

210701120

Output:



```
(myenv) hadoop@hadoop-VirtualBox:~$ python process_data.py
Raw JSON Data: [{"name": "Alice", "age": 30, "department": "Engineering", "salary": 70000}, {"name": "Bob", "age": 45, 
department": "Marketing", "salary": 80000}, {"name": "Charlie", "age": 28, "department": "Design", "salary": 65000}, {"
ame": "Diana", "age": 35, "department": "HR", "salary": 75000}, {"name": "Eve", "age": 40, "department": "Finance", "sa
ary": 85000}]



Filtered JSON file saved successfully.
Projection: Select only name and salary columns
      name  salary
0    Alice   70000
1      Bob   80000
2  Charlie   65000
3    Diana   75000
4      Eve   85000
Aggregation: Calculate total salary
Total Salary: 375000


Count: Number of employees earning more than 50000
Number of High Earners (>50000): 5
```



```

Limit: Top 5 highest salary
Top 5 Earners:
      name  age   department  salary
4      Eve   40      Finance   85000
1      Bob   45    Marketing   80000
3    Diana   35           HR   75000
0    Alice   30  Engineering   70000
2  Charlie   28       Design   65000


Skip: First 2 rows skipped
Skipped DataFrame (First 2 rows skipped):
      name  age department  salary
2  Charlie   28     Design   65000
3    Diana   35         HR   75000
4      Eve   40    Finance   85000


Remove: Employees from IT department removed
Filtered DataFrame (IT department removed):
      name  age   department  salary
0    Alice   30  Engineering   70000
1      Bob   45    Marketing   80000
2  Charlie   28       Design   65000
3    Diana   35           HR   75000
4      Eve   40      Finance   85000
```