

Ex No 6

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

AIM:

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

PROCEDURE:

- Create a json file 'employees.json' and provide data in it.
- Open the command prompt.
- Navigate to the folder where employees.json is stored.
- Load and view the JSON data with jq.
- Use the jq commands for projection, aggregation, removal, counting, limiting, and sorting operations.

employees.json:

```
[
  {
    "id": 1,
    "name": "Alice Johnson",
    "department": "Engineering",
    "age": 29,
    "salary": 70000
  },
  {
    "id": 2,
    "name": "Bob Smith",
    "department": "Marketing",
    "age": 35,
```

```
    "salary": 55000
  },
  {
    "id": 3,
    "name": "Charlie Davis",
    "department": "Engineering",
    "age": 25,
    "salary": 60000
  },
  {
    "id": 4,
    "name": "Dana Lee",
    "department": "Human Resources",
    "age": 40,
    "salary": 65000
  },
  {
    "id": 5,
    "name": "Eve Martinez",
    "department": "Finance",
    "age": 45,
    "salary": 75000
  }
]
```

OUTPUT:

```

varusha@Ubuntu: ~
2024-09-25 14:03:20.755 warn: util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[{"name": "John Doe", "age": 30, "department": "HR", "salary": 50000}, {"name": "Alice Johnson", "age": 35, "department": "Finance", "salary": 70000}, {"name": "Bob Brown", "age": 28, "department": "Marketing", "salary": 55000}]varusha@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}
]
Sorted by name
  name  salary
2  Alice Johnson  70000
3    Bob Brown  55000
4  Charlie Black  80000
1  Jane Smith   60000
0    John Doe   50000
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

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
varusha@Ubuntu: ~$

```

RESULT:

Thus to import a JSON file from the command line and apply the following actions with the data present in the JSON file where, projection, aggregation, remove, count, limit, skip and sort using jq tool is completed successfully

