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:

Installation of jq packages:

Running jq queries:

I. **Projection:**

```
jq ".[] | {name: .name, salary: .salary}" Desktop/employees.json
```

I. **Aggregation:**

```
jq "[.[] | .salary] | add" Desktop/employees.json
```

I. **Remove:**

```
jq "del(.[] | .age)" Desktop/employees.json
```

I. **Count:**

```
jq ". | length" Desktop/employees.json
```

I. **Limit:**

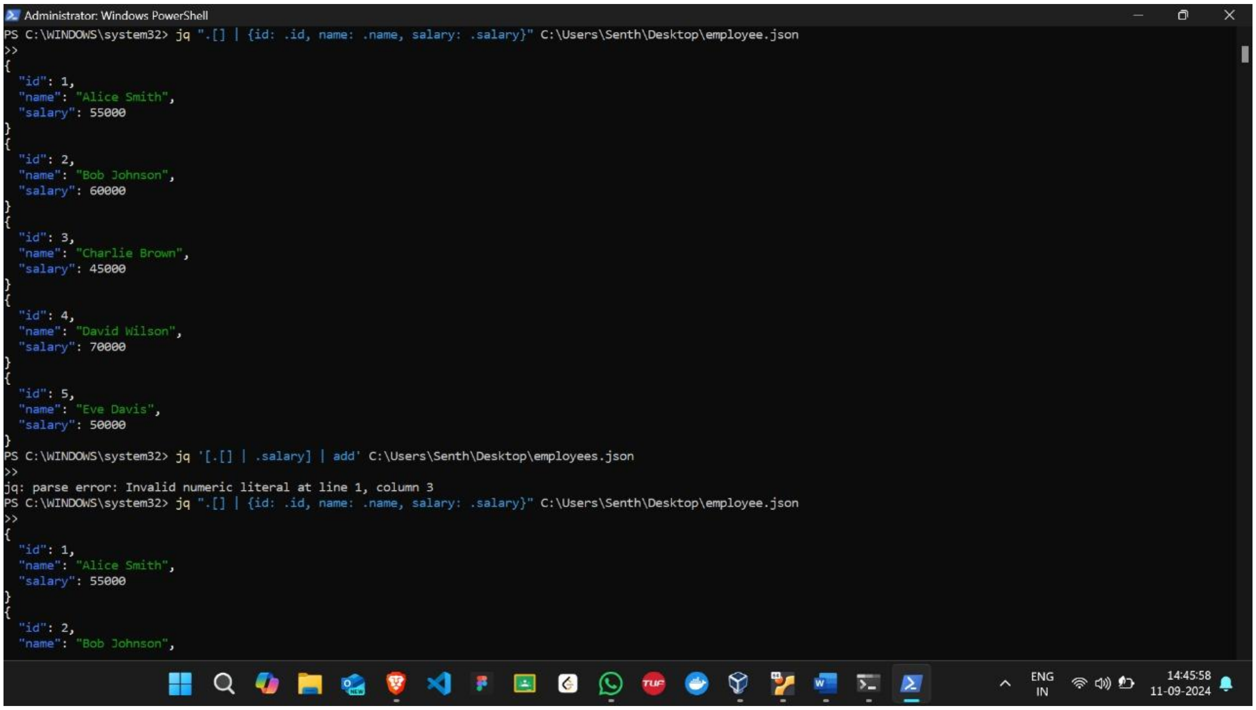
```
jq ".[0:3]" Desktop/employees.json
```

I. **Skip:**

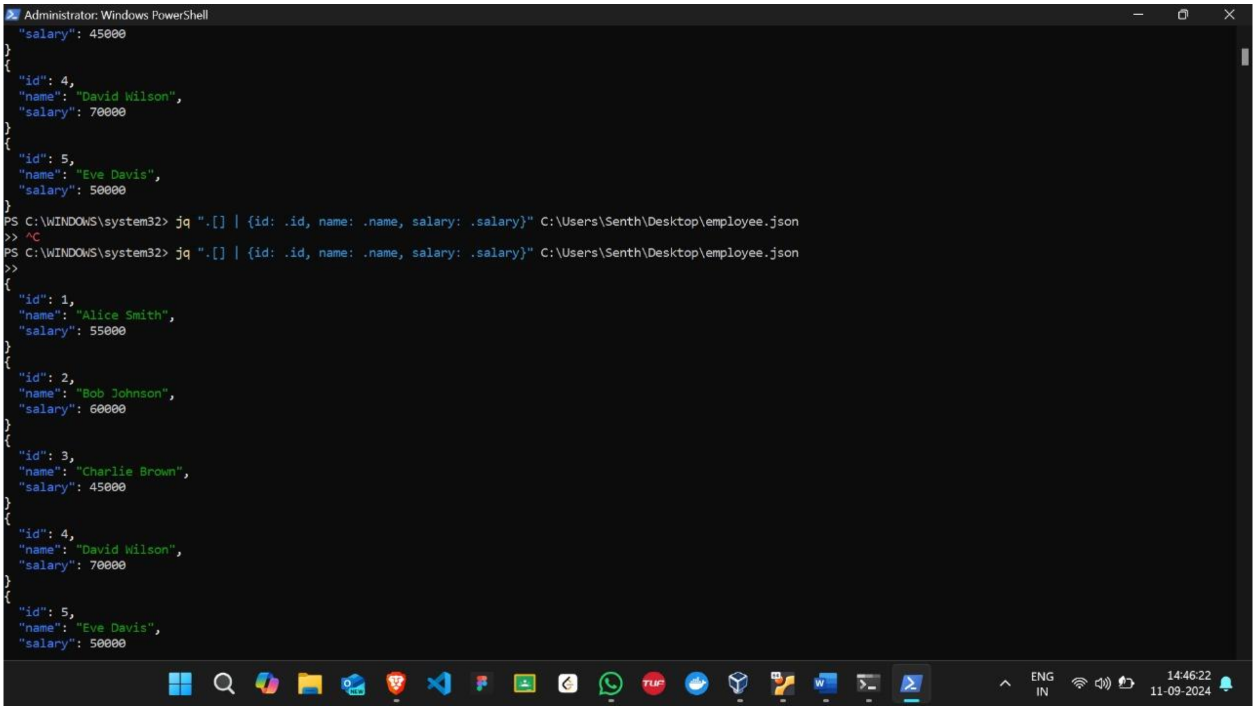
```
jq ".[2:]" Desktop/employees.json
```

I. **Sort:**

```
jq "sort_by(.age)" Desktop/employees.json
```

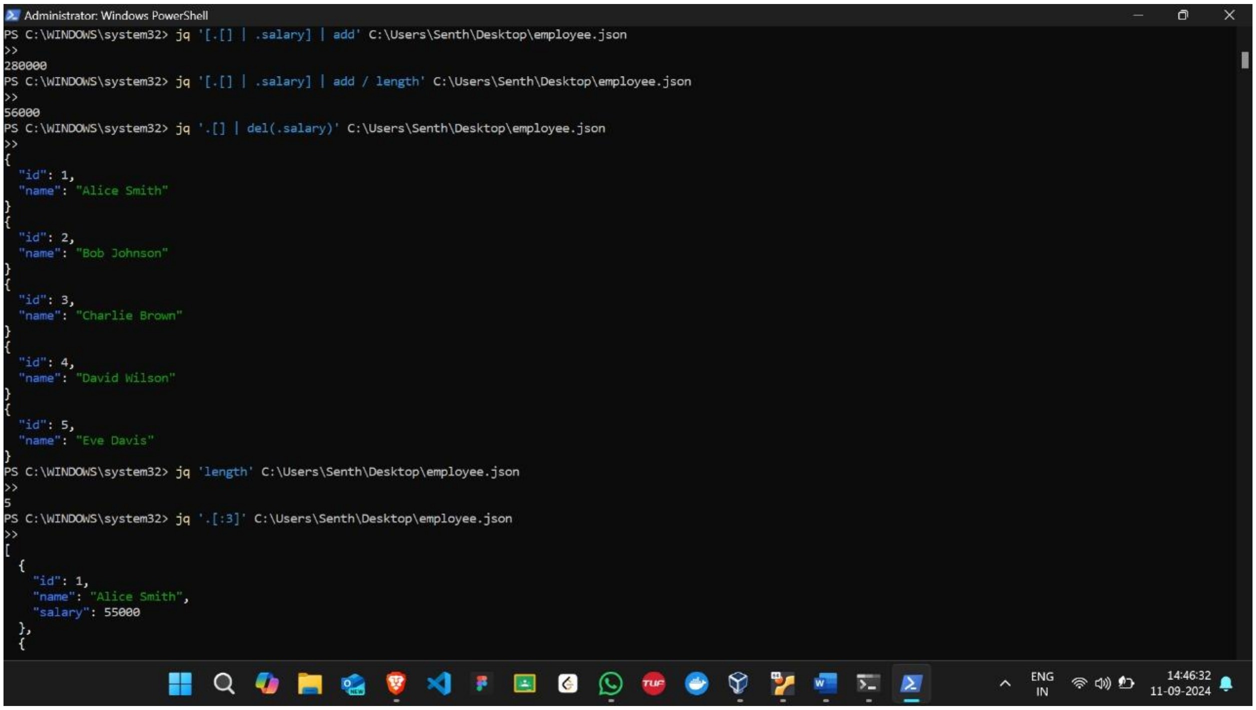


```
PS C:\WINDOWS\system32> jq ".[] | {id: .id, name: .name, salary: .salary}" C:\Users\Senth\Desktop\employee.json
>>
{
  "id": 1,
  "name": "Alice Smith",
  "salary": 55000
}
{
  "id": 2,
  "name": "Bob Johnson",
  "salary": 60000
}
{
  "id": 3,
  "name": "Charlie Brown",
  "salary": 45000
}
{
  "id": 4,
  "name": "David Wilson",
  "salary": 70000
}
{
  "id": 5,
  "name": "Eve Davis",
  "salary": 50000
}
PS C:\WINDOWS\system32> jq '[.[] | .salary] | add' C:\Users\Senth\Desktop\employees.json
>>
jq: parse error: Invalid numeric literal at line 1, column 3
PS C:\WINDOWS\system32> jq ".[] | {id: .id, name: .name, salary: .salary}" C:\Users\Senth\Desktop\employee.json
>>
{
  "id": 1,
  "name": "Alice Smith",
  "salary": 55000
}
{
  "id": 2,
  "name": "Bob Johnson",
```

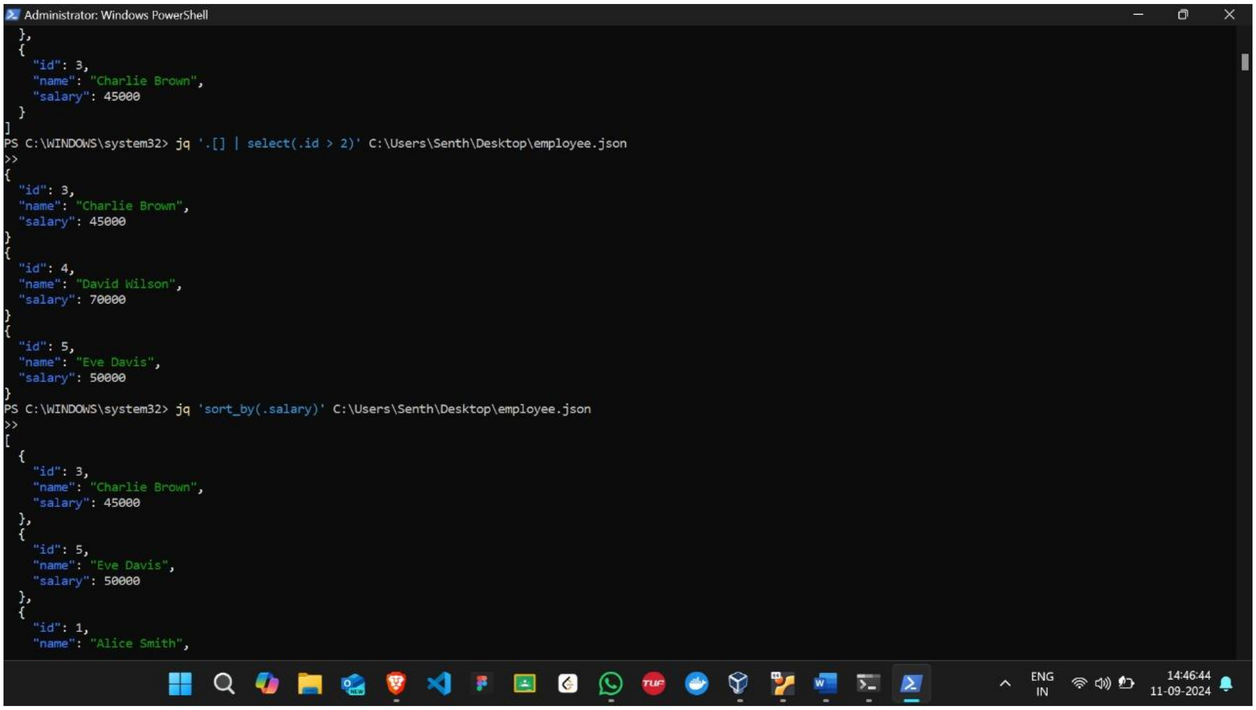


```
  "salary": 45000
}
{
  "id": 4,
  "name": "David Wilson",
  "salary": 70000
}
{
  "id": 5,
  "name": "Eve Davis",
  "salary": 50000
}
PS C:\WINDOWS\system32> jq ".[] | {id: .id, name: .name, salary: .salary}" C:\Users\Senth\Desktop\employee.json
>> ^C
PS C:\WINDOWS\system32> jq ".[] | {id: .id, name: .name, salary: .salary}" C:\Users\Senth\Desktop\employee.json
>>
{
  "id": 1,
  "name": "Alice Smith",
  "salary": 55000
}
{
  "id": 2,
  "name": "Bob Johnson",
  "salary": 60000
}
{
  "id": 3,
  "name": "Charlie Brown",
  "salary": 45000
}
{
  "id": 4,
  "name": "David Wilson",
  "salary": 70000
}
{
  "id": 5,
  "name": "Eve Davis",
  "salary": 50000
```

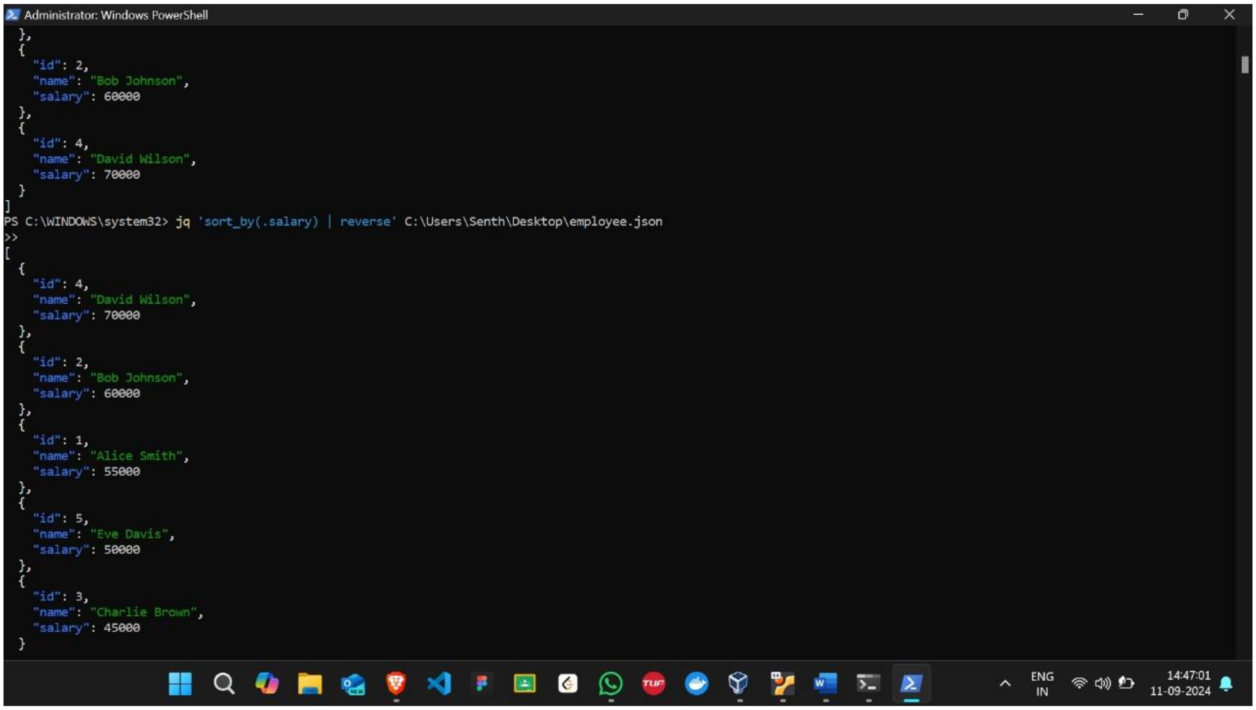
210701238



```
PS C:\WINDOWS\system32> jq '[.[] | .salary] | add' C:\Users\Senth\Desktop\employee.json
>>
280000
PS C:\WINDOWS\system32> jq '[.[] | .salary] | add / length' C:\Users\Senth\Desktop\employee.json
>>
56000
PS C:\WINDOWS\system32> jq '.[] | del(.salary)' C:\Users\Senth\Desktop\employee.json
>>
{
  "id": 1,
  "name": "Alice Smith"
}
{
  "id": 2,
  "name": "Bob Johnson"
}
{
  "id": 3,
  "name": "Charlie Brown"
}
{
  "id": 4,
  "name": "David Wilson"
}
{
  "id": 5,
  "name": "Eve Davis"
}
PS C:\WINDOWS\system32> jq 'length' C:\Users\Senth\Desktop\employee.json
>>
5
PS C:\WINDOWS\system32> jq '.[:3]' C:\Users\Senth\Desktop\employee.json
>>
[
  {
    "id": 1,
    "name": "Alice Smith",
    "salary": 55000
  },
  {
```



```
  },
  {
    "id": 3,
    "name": "Charlie Brown",
    "salary": 45000
  }
]
PS C:\WINDOWS\system32> jq '.[] | select(.id > 2)' C:\Users\Senth\Desktop\employee.json
>>
{
  "id": 3,
  "name": "Charlie Brown",
  "salary": 45000
}
{
  "id": 4,
  "name": "David Wilson",
  "salary": 70000
}
{
  "id": 5,
  "name": "Eve Davis",
  "salary": 50000
}
PS C:\WINDOWS\system32> jq 'sort_by(.salary)' C:\Users\Senth\Desktop\employee.json
>>
[
  {
    "id": 3,
    "name": "Charlie Brown",
    "salary": 45000
  },
  {
    "id": 5,
    "name": "Eve Davis",
    "salary": 50000
  },
  {
    "id": 1,
    "name": "Alice Smith",
```



```
  },
  {
    "id": 2,
    "name": "Bob Johnson",
    "salary": 60000
  },
  {
    "id": 4,
    "name": "David Wilson",
    "salary": 70000
  }
]
PS C:\WINDOWS\system32> jq 'sort_by(.salary) | reverse' C:\Users\Senth\Desktop\employee.json
>>
[
  {
    "id": 4,
    "name": "David Wilson",
    "salary": 70000
  },
  {
    "id": 2,
    "name": "Bob Johnson",
    "salary": 60000
  },
  {
    "id": 1,
    "name": "Alice Smith",
    "salary": 55000
  },
  {
    "id": 5,
    "name": "Eve Davis",
    "salary": 50000
  },
  {
    "id": 3,
    "name": "Charlie Brown",
    "salary": 45000
  }
```

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.