Employee Program using PySpark

Find the total, minimum and maximum salary of all employees.

Employee.txt file

1 Ajay Programmer 20000
2 Poonam Data_Scientest 40000
3 Komal Programmer 30000
4 Kinjal Bussiness_Analyst 70000
5 Chintan Team_Lead 60000

Program using Pyspark Shell

```
>>> sum_salary = 0
>>> max_salary = 0
>>> min_salary = 0

>>> final_result = sc.emptyRDD()
>>> result_list = []

>>> hdfsfile = sc.textFile("hdfs://localhost:9000/user/employee.txt")
>>> hdfsfile.collect()
['1 Ajay Programmer 20000', '2 Poonam Data_Scientest 40000', '3 Komal Programmer 30000', '4 Kinjal Bussiness_Analyst 70000', '5 Chintan Team_Lead 60000 ']

>>> data = hdfsfile.map(lambda line: line.split(" "))
>>> data.collect()
[['1', 'Ajay', 'Programmer', '20000'], ['2', 'Poonam', 'Data_Scientest', '40000'], ['3', 'Komal', 'Programmer', '30000'], ['4', 'Kinjal', 'Bussiness_Analyst', '70000'], ['5', 'Chintan', 'Team_Lead', '60000', '']]

>>> salary = data.map(lambda sal:[sal[i] for i in [3]])
>>> salary.collect()
[['20000'], ['40000'], ['30000'], ['70000'], ['60000']]

>>> total_salary = salary.flatMap(lambda tot_sal: tot_sal)
>>> total_salary.collect()
['20000', '40000', '30000', '70000', '60000']

>>> total_salary = total_salary.map(lambda tot_sal:int(tot_sal))
>>> total_salary.collect()
[20000, 40000, 30000, 70000, 60000]

>>> sum_salary = total_salary.sum()
>>> print(sum_salary)
220000

>>> max_salary = total_salary.max()
>>> print(max_salary)
70000

>>> min_salary = total_salary.min()
>>> print(min_salary)
20000
```

```
>>> result_list.append(sum_salary)
>>> result_list.append(max_salary)
>>> result_list.append(min_salary)
>>> print(result_list)
[220000, 70000, 20000]

>>> final_result = sc.parallelize(result_list)
>>> final_result.collect()
[220000, 70000, 20000]
```

Program using Python (employee_spark.py)

```
from pyspark import SparkContext

def Emp_Analysis():
        sc = SparkContext(appName='EmployeeAnalysis')
        sum_salary = 0
        max_salary = 0
        min_salary = 0
        final_result = sc.emptyRDD()
        result_list = []
        hdfsfile = sc.textFile("hdfs://localhost:9000/user/employee.txt")
        data = hdfsfile.map(lambda line: line.split(" "))
        salary = data.map(lambda sal:[sal[i] for i in [3]])
        total_salary = salary.flatMap(lambda tot_sal: tot_sal)
        total_salary = total_salary.map(lambda tot_sal:int(tot_sal))

        sum_salary = total_salary.sum()
        max_salary = total_salary.max()
        min_salary = total_salary.min()

        result_list.append(sum_salary)
        result_list.append(max_salary)
        result_list.append(min_salary)

        final_result = sc.parallelize(result_list)
        final_result.saveAsTextFile("hdfs://localhost:9000/sparkoutput_EmployeeAnalysis3")
        sc.stop()

if __name__ == '__main__':
        Emp_Analysis()
```

Executing file in hadoop

faculty@glsica:~/hadoop-3.1.3$ spark-submit --master local employee_spark.py

Viewing output on hdfs

faculty@glsica:~/hadoop-3.1.3$ hdfs dfs -cat /sparkoutput_EmployeeAnalysis3/part-00000

```
2020-08-28 15:14:08,703 INFO sasl.SaslDataTransferClient: SASL encryption trust check:
localHostTrusted = false, remoteHostTrusted = false
220000
70000
20000
```