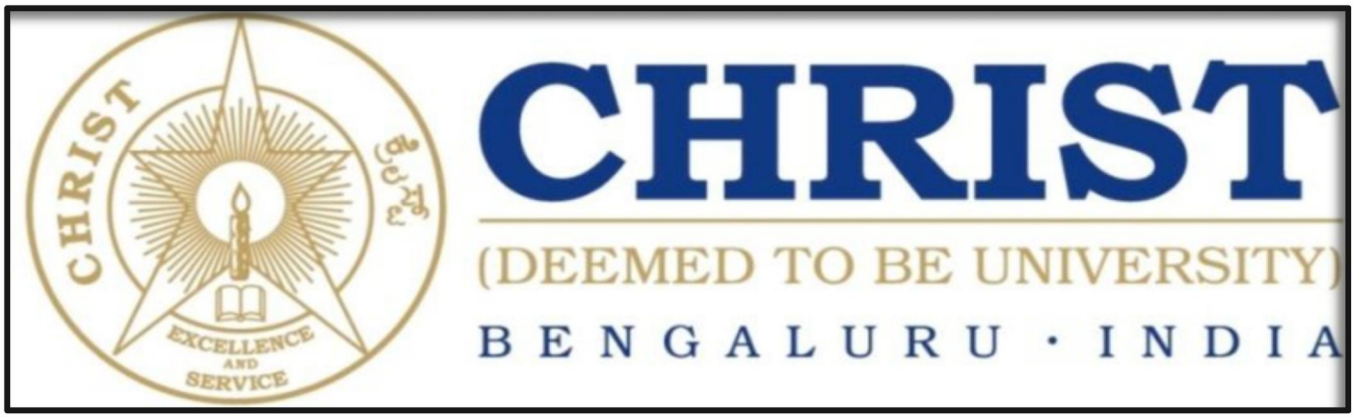

EMPLOYEE SALARY ANALYSIS
(DEMONSTRATION OF HADOOP STREAMING)

SANJAY R

2348055

2nd December 2024

MDS571

Big Data Analytics

Department of Statistics and Data Science

CHRIST (Deemed to be University)

Department of Statistics and Data Science

Course:MDS571-Big Data Analytics **Lab :7**

R.SANJAY **2348055**

Employee Salary Analysis

Demonstration of Hadoop Streaming

✧ INTRODUCTION:

In today's data-driven world, understanding compensation trends within an organization is crucial for making informed business decisions. Companies must analyze salary data to ensure competitive pay structures, recognize disparities, and foster a fair work environment. As organizations grow, managing vast amounts of salary data becomes increasingly challenging. Traditional methods struggle to handle large datasets efficiently, leading to delays and limited insights. To address this challenge, the MapReduce framework offers a scalable and distributed approach for processing large volumes of data.

This project utilizes MapReduce to perform a comprehensive salary analysis. The goal is to generate insights across multiple aspects, such as calculating average salary by department, identifying the highest-paid employee in each department, conducting gender-based salary comparisons, and analyzing the relationship between experience and salary. By using the mapper and reducer components, we break down complex calculations into manageable tasks that can be processed in parallel, improving efficiency and accuracy.

✧ PROBLEM DESCRIPTION:

In modern organizations, salary analysis is crucial for human resource management, but managing large datasets manually or with basic tools becomes increasingly difficult. The challenge lies not only in processing vast amounts of data but also in deriving meaningful insights such as identifying pay disparities, understanding compensation trends across departments, and evaluating the correlation between experience and salary.

To solve these problems, we use the MapReduce framework, which can handle large-scale datasets by distributing computation across multiple nodes. This approach addresses the following key issues:

- ✓ **Average Salary by Department:**Calculate the average salary per department, ensuring results are sorted in descending order.
- ✓ **Highest Paid Employee per Department:** Identify the highest-paid employee in each department and display their salary.
- ✓ **Salary Distribution per Department:**Analyze salary ranges within each department, including minimum, maximum, and percentiles.
- ✓ **Gender-based Salary Comparison:** Compare salaries between male and female employees within each department to ensure equity.
- ✓ **Experience vs Salary Analysis:**Explore the relationship between years of experience and salary, identifying trends and department-specific variations.

✧ DATASET DESCRIPTION:

The dataset used for this project is a comprehensive record of employee details, including information related to their salary, department, experience, gender, and other attributes. The dataset contains multiple features that are used to generate various insights, such as salary distributions, gender-based salary analysis, and experience-based salary trends. The dataset is structured with the following key features:

- ✓ **Employee ID:**A unique identifier for each employee. This field is used to track individual records.

- ✓ **Employee Name:**The name of the employee. It helps to identify the person associated with the data, particularly in tasks like identifying the highest-paid employee.
- ✓ **Department:** The department to which the employee belongs. This feature is crucial for aggregating salary data and performing department-wise analysis, such as calculating the average salary per department or finding the highest-paid employee in each department.
- ✓ **Salary:**The annual salary of the employee, represented in monetary units. This is the primary feature for salary analysis, including calculating average salary, gender-based salary comparisons, and salary distribution within departments.
- ✓ **Experience:**The number of years of experience the employee has in the industry. This feature is essential for analyzing the correlation between experience and salary, and for identifying trends in salary growth based on experience.
- ✓ **Gender:**The gender of the employee (e.g., Male, Female). This feature is used for gender-based salary comparison to assess pay equity between male and female employees within each department.
- ✓ **Age:**The age of the employee, which may be used to analyze age-related salary trends or to provide demographic insights.
- ✓ **Location:**The geographic location of the employee's office. While not directly involved in the analysis for this specific project, location could be relevant for future analysis, such as comparing salaries across different regions

✧ CODE:

✓ mapper.py:

```
#!/usr/bin/env python3
import sys
# The Mapper processes each line of input from the dataset
for line in sys.stdin:
    # Skip the header line, which contains column names
    if line.startswith("Employee ID"):
        continue
```

```
    # Skip empty lines
    if not line.strip():
        continue
    # Split the line into columns based on tab separation
    columns = line.strip().split("\t")
    # Ensure the line has all 8 required columns
    if len(columns) == 8:
        # Extract each field from the row
        employee_id = columns[0]
        employee_name = columns[1]
        department = columns[2]
        salary = int(columns[3])
        experience = int(columns[4])
        gender = columns[5]
        age = int(columns[6])
        location = columns[7]

        # Emit key-value pairs for each of the tasks
      # Task 1: Emit department and salary for average salary computation
        print(f"avg_salary\t{department}\t{salary}")

      # Task 2: Emit department, salary, and employee name for highest-paid employee
        print(f"max_salary\t{department}\t{salary}\t{employee_name}")

      # Task 3: Emit department and salary for salary distribution
        print(f"distribution\t{department}\t{salary}")

      # Task 4: Emit department, gender, and salary for gender-based comparison
        print(f"gender_salary\t{department}\t{gender}\t{salary}")

    # Task 5: Emit experience, salary, and department for experience-salary analysis
        print(f"experience_salary\t{experience}\t{salary}\t{department}")
```

✓ **reducer.py:**

```
#!/usr/bin/env python3
import sys
from collections import defaultdict
import numpy as np    # Used for calculating median

# Initialize dictionaries to store aggregates for each task
department_salary_sum = defaultdict(int)    # Total salaries for each department
department_salary_count = defaultdict(int)    # Count of employees per department
department_max_salary = defaultdict(lambda: (-1, ""))    # (Max salary, employee
name) for each department
department_salaries = defaultdict(list)    # List of salaries for each department
department_gender_salary = defaultdict(lambda: defaultdict(list))    # {department:
{gender: [salaries]}}
experience_salary_data = defaultdict(list)    # {experience: [(salary, department)]}
# Read each line of Mapper's output
for line in sys.stdin:
    # Split the line into parts
    parts = line.strip().split("\t")
    key = parts[0]    # The task identifier (e.g., avg_salary, max_salary, etc.)

    if key == "avg_salary":
        # Task 1: Average Salary per Department
        department = parts[1]
        salary = int(parts[2])
        department_salary_sum[department] += salary
        department_salary_count[department] += 1

    elif key == "max_salary":
        # Task 2: Highest Paid Employee per Department
        department = parts[1]
        salary = int(parts[2])
        employee_name = parts[3]
```

```
            if salary > department_max_salary[department][0]:    # Update max salary
if higher
                department_max_salary[department] = (salary, employee_name)

        elif key == "distribution":
            # Task 3: Salary Distribution per Department
            department = parts[1]
            salary = int(parts[2])
            department_salaries[department].append(salary)

        elif key == "gender_salary":
            # Task 4: Gender-based Salary Comparison
            department = parts[1]
            gender = parts[2]
            salary = int(parts[3])
            department_gender_salary[department][gender].append(salary)

        elif key == "experience_salary":
            # Task 5: Experience vs Salary Analysis
            experience = int(parts[1])
            salary = int(parts[2])
            department = parts[3]
            experience_salary_data[experience].append((salary, department))

# Output results for each task
# Task 1: Average Salary by Department
print("\nAverage Salary by Department (Descending Order):\n")
sorted_departments = sorted(department_salary_sum.items(),
                            key=lambda x: x[1] /
department_salary_count[x[0]],
                            reverse=True)
print(f"{'Department':<20}{'Avg Salary':<15}")
print("-" * 35)
for department, total_salary in sorted_departments:
```

```
    avg_salary = total_salary / department_salary_count[department]
    print(f"{department:<20}{avg_salary:<15.2f}")

# Task 2: Highest Paid Employee per Department
print("\nHighest Paid Employee by Department:\n")
print(f"{'Department':<20}{'Employee Name':<20}{'Salary':<10}")
print("-" * 50)
for department, (max_salary, employee_name) in
sorted(department_max_salary.items(),
                                                    key=lambda x:
x[1][0],
                                                    reverse=True):
    print(f"{department:<20}{employee_name:<20}{max_salary:<10}")

# Task 3: Salary Distribution by Department
print("\nSalary Distribution by Department:\n")
print(f"{'Department':<20}{'Min Salary':<15}{'Max Salary':<15}{'Median
Salary':<15}")
print("-" * 60)
for department, salaries in department_salaries.items():
    salaries.sort()
    min_salary = min(salaries)
    max_salary = max(salaries)
    median_salary = np.median(salaries)

print(f"{department:<20}{min_salary:<15}{max_salary:<15}{median_salary:<15.2f}
")

# Task 4: Gender-Based Salary Comparison by Department
print("\nGender-Based Salary Comparison by Department:\n")
print(f"{'Department':<20}{'Gender':<10}{'Avg Salary':<15}")
print("-" * 45)
for department, gender_salaries in department_gender_salary.items():
    for gender, salaries in gender_salaries.items():
```

```
        avg_salary = sum(salaries) / len(salaries)
        print(f"{department:<20}{gender:<10}{avg_salary:<15.2f}")

# Task 5: Experience vs Salary Analysis
print("\nExperience vs Salary Analysis (Descending Order):\n")
print(f"{'Experience':<12}{'Avg Salary':<12}{'Max Salary':<12}{'Min
Salary':<12}{'Max Dept':<15}{'Min Dept':<15}")
print("-" * 75)
experience_data_sorted = sorted(experience_salary_data.items(), key=lambda x:
x[0], reverse=True)
for experience, salary_data in experience_data_sorted:
    salaries = [salary for salary, _ in salary_data]
    departments = [department for _, department in salary_data]
    avg_salary = sum(salaries) / len(salaries)
    max_salary = max(salaries)
    min_salary = min(salaries)
    max_salary_dept = departments[salaries.index(max_salary)]
    min_salary_dept = departments[salaries.index(min_salary)]
print(f"{experience:<12}{avg_salary:<12.2f}{max_salary:<12}{min_salary:<12}{m
ax_salary_dept:<15}{min_salary_dept:<15}")
```

✧ PROGRAM DESCRIPTION:

✓ The program is designed to analyze employee salary data using the MapReduce framework, a powerful paradigm for distributed data processing. It is implemented using a combination of a Mapper and a Reducer script, where the Mapper processes raw data into structured key-value pairs, and the Reducer aggregates, analyzes, and formats the results to meet specific objectives. The program effectively addresses five key tasks:

- ◆ Calculating average salaries by department,
- ◆ Identifying the highest-paid employee in each department,
- ◆ Analyzing salary distributions,
- ◆ Comparing gender-based salaries, and
- ◆ Studying the relationship between salary and years of experience.

✓ mapper.py:

The mapper.py script is the first step in the MapReduce framework. It processes each input record, extracts relevant information, and emits key-value pairs for the reducer to process further. Below is a detailed breakdown of how the mapper achieves the objectives, along with the code snippets and explanations for each.

Objective 1: Calculating Average Salary by Department

The goal of this objective is to calculate the average salary of employees within each department. The mapper reads the data and emits key-value pairs, where the key consists of the identifier avg_salary and the department name, while the value is the employee's salary. This structure allows the reducer to later aggregate these salaries and compute averages.

Code snippet for Objective 1:

 # Task 1: Average Salary per Department

print(f"avg_salary\t{department}\t{salary}")

The script processes each line of input data and splits it to extract the department and salary.It then outputs a key-value pair with avg_salary as the key, the department name as the second part of the value, and the employee's salary as the third part.

The format of the emitted pair is "avg_salary\t{department}\t{salary}", where \t represents tab separation for the MapReduce framework to understand the structure.

Objective 2: Identifying the Highest-Paid Employee by Department

This objective requires the mapper to find the highest-paid employee within each department. The mapper emits the department name as the key and the salary and employee name as the value. The reducer will use this data to determine which employee has the highest salary in each department.

Code snippet for Objective 2:

 # Task 2: Highest Paid Employee per Department

print(f"max_salary\t{department}\t{salary}\t{employee_name}")

The mapper emits a key-value pair where the key is max_salary, the department name is part of the value, followed by the salary and employee name.

This ensures that for each department, the reducer can compare salaries and identify the highest salary and the corresponding employee.

Objective 3: Salary Distribution by Department

The goal of this objective is to generate the salary distribution for each department, including the minimum, maximum, and median salaries. The mapper will emit department names as keys, with salary values that the reducer will process to compute the desired statistics.

Code snippet for Objective 3:

 # Task 3: Salary Distribution per Department

print(f"distribution\t{department}\t{salary}")

The mapper emits key-value pairs where the key is distribution, and the department name is followed by the employee's salary.

The reducer will collect all salaries per department and compute the minimum, maximum, and median values to provide the required distribution.

Objective 4: Gender-Based Salary Comparison

For gender-based salary comparison, the mapper emits key-value pairs where the key consists of the department name and gender, and the value is the salary. This allows the reducer to calculate the average salary for each gender within each department, enabling a comparison of salaries based on gender.

Code snippet for Objective 4:

 # Task 4: Gender-based Salary Comparison per Department

print(f"gender_salary\t{department}\t{gender}\t{salary}")

The mapper processes each input record, extracting the department, gender, and salary. It then emits the key gender_salary, followed by the department name, gender, and salary.

This structure enables the reducer to calculate the average salary for each gender within the department and perform comparisons.

Objective 5: Salary vs Experience Analysis

The mapper here groups salaries by experience levels. It emits key-value pairs where the key is the experience level, and the value consists of the salary and department. The reducer can then aggregate the data to analyze the relationship between salary and experience.

Code snippet for Objective 5:

 # Task 5: Experience vs Salary

print(f"experience_salary\t{experience}\t{salary}\t{department}")

The mapper emits the key experience_salary with the employee's experience level, salary, and department as the value.This allows the reducer to perform an analysis on how salaries vary with different experience levels across departments.

✓ reducer.py:

The reducer.py script is responsible for processing the key-value pairs emitted by the mapper. It aggregates data based on the tasks and generates the final results. Below is a detailed breakdown of how the reducer handles each objective, including code snippets and explanations.

Objective 1: Calculating Average Salary by Department

The reducer calculates the average salary by department by summing the salaries and counting the number of employees. The average is then computed and displayed.

The reducer checks if the key is avg_salary, indicating that it needs to process salary data for average calculation.It sums the salaries for each department and counts the number of employees in that department.The average salary is calculated by dividing the total salary by the employee count.Finally, the results are sorted by average salary in descending order and printed.

Objective 2: Identifying the Highest-Paid Employee by Department

For this objective, the reducer identifies the highest-paid employee by comparing the salaries within each department.

The reducer processes data for the max_salary key, comparing the salary values for each department.It keeps track of the highest salary and the corresponding employee

name.The results are sorted by salary in descending order and printed, showing the highest-paid employee in each department.

Objective 3: Salary Distribution by Department

The reducer computes the minimum, maximum, and median salary for each department based on the salary values emitted by the mapper.

The reducer groups salaries by department and computes the minimum and maximum salaries.It also calculates the median salary by sorting the salary list and using numpy to find the middle value.The final results display the salary distribution for each department.

Objective 4: Gender-Based Salary Comparison

For gender-based salary comparison, the reducer computes the average salary for each gender in every department.

The reducer processes the gender_salary key and aggregates salaries based on gender for each department.The average salary for each gender within a department is calculated by summing the salaries and dividing by the count.The results are printed, showing the gender-based salary comparison for each department.

Objective 5: Experience vs Salary Analysis

The reducer analyzes salary data based on experience levels, calculating the average, maximum, and minimum salaries for each experience level.

The reducer processes the experience_salary key, grouping salaries by experience level.It calculates the average, maximum, and minimum salaries for each experience level.The results are sorted by experience level in descending order, and the corresponding maximum and minimum salaries for each experience level are displayed along with the departments.

✧ PROJECT SETUP AND EXECUTION:

✓ DATASET PREPARATION:

The Employee Salary Analysis Program relies on a primary dataset containing essential employee-related information.

- **Employee Dataset:** This dataset contains crucial attributes such as employee_id, employee_name, department, salary, years of experience, gender, age, and location. Each record in the dataset provides essential details that help in understanding various aspects of the employees, such as their salaries, the departments they work in, their years of experience, and other demographic factors.

1	Arvind Kumar	HR	55000	3	Male	28	New Delhi
2	Swati Sharma	HR	62000	5	Female	35	Mumbai
3	Rahul Verma	Engineering	90000	6	Male	30	Bangalore
4	Neha Patel	Engineering	95000	7	Female	32	Pune
5	Vikram Singh	Engineering	105000	8	Male	33	Hyderabad
6	Ananya Gupta	Marketing	58000	4	Female	27	Gurgaon
7	Sahil Mehta	Marketing	60000	5	Male	29	Chennai
8	Pooja Yadav	Finance	75000	6	Female	32	Noida
9	Amit Kumar	Finance	82000	7	Male	34	Bangalore
10	Ritu Agarwal	Finance	78000	5	Female	31	Chennai
11	Deepika Joshi	Marketing	62000	6	Female	33	Mumbai
12	Ravi Desai	Engineering	100000	9	Male	36	Hyderabad
13	Simran Kaur	HR	50000	2	Female	26	Chandigarh
14	Rajesh Reddy	Engineering	115000	10	Male	38	Bangalore
15	Seema Sharma	Marketing	59000	3	Female	28	Chennai
16	Vinod Kumar	HR	53000	4	Male	30	Pune
17	Anjali Nair	Finance	80000	6	Female	33	Delhi
18	Vikas Patel	Finance	86000	8	Male	34	Bangalore
19	Sanjay Kumar	Engineering	97000	7	Male	32	Noida
20	Kavita Singh	HR	56000	4	Female	31	Delhi
21	Sachin Kumar	Marketing	64000	5	Male	30	Gurgaon
22	Manish Kapoor	Engineering	110000	9	Male	37	Chennai
23	Sonali Verma	HR	57000	5	Female	29	Pune
24	Neeraj Soni	Marketing	67000	6	Male	34	Mumbai
25	Abhishek Gupta	Finance	90000	10	Male	36	Delhi
26	Preeti Sharma	Engineering	99000	7	Female	33	Bangalore
27	Anil Chauhan	Finance	82000	4	Male	30	Gurgaon
28	Rekha Bhatia	HR	54000	3	Female	28	Chennai
29	Sandeep Yadav	Marketing	65000	6	Male	32	Pune
30	Komal Mehra	Finance	86000	9	Female	33	Delhi

The dataset is structured in a tab-separated format, allowing for efficient parsing and processing using analytical tools like MapReduce. The employee_id serves as a unique identifier for each employee, while attributes like salary, department, gender, and location allow for detailed analysis and aggregation. This analysis can uncover trends and insights related to salary distribution, gender representation, and experience across different departments and locations.

To facilitate processing with the Hadoop MapReduce framework, the dataset was placed within the Hadoop directory structure at /MDS2024/LAB7. This setup ensured that the files were easily accessible for the program, allowing seamless integration and efficient execution of the MapReduce process.

```
hadoop@Ubuntu22:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [Ubuntu22]
hadoop@Ubuntu22:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@Ubuntu22:~$ jps
6433 ResourceManager
5957 DataNode
6184 SecondaryNameNode
7209 Jps
6555 NodeManager
5837 NameNode
```

```
hadoop@Ubuntu22:~$ hadoop fs -ls /MDS2024/LAB7/
Found 1 items
-rw-r--r--   3 hadoop supergroup       1459 2024-12-14 09:57 /MDS2024/LAB7/Employee
```

✧ EXECUTION PROCESS:

- The mapper.py file was stored at /home/hadoop/Labs/LAB7/mapper.py, and the reducer.py file was saved at /home/hadoop/Labs/LAB7/reducer.py.
- The next step involved executing the program within the Hadoop environment, using the following command:

$ hadoop jar /home/hadoop/Desktop/hadoop-streaming-3.3.6.jar -input /MDS2024/LAB7/Employee -output /MDS2024/LAB7/output -mapper /home/hadoop/Labs/LAB7/mapper.py -reducer /home/hadoop/Labs/LAB7/reducer.py

```
hadoop@Ubuntu22:~$ hadoop jar /home/hadoop/Desktop/hadoop-streaming-3.3.6.jar -input /MDS2024/LAB7/Employee -output /MDS2024/LAB7/output -mapper /home/hadoop/Labs/LAB7/mapper.py -reducer /home/hadoop/
s/LAB7/reducer.py
2024-12-14 12:13:18,250 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2024-12-14 12:13:18,478 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2024-12-14 12:13:18,478 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2024-12-14 12:13:18,508 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2024-12-14 12:13:19,100 INFO mapred.FileInputFormat: Total input files to process : 1
2024-12-14 12:13:19,195 INFO mapreduce.JobSubmitter: number of splits:1
2024-12-14 12:13:19,656 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1185818458_0001
2024-12-14 12:13:19,656 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-12-14 12:13:19,901 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2024-12-14 12:13:19,904 INFO mapreduce.Job: Running job: job_local1185818458_0001
2024-12-14 12:13:19,974 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2024-12-14 12:13:20,008 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2024-12-14 12:13:20,052 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-12-14 12:13:20,053 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2024-12-14 12:13:20,232 INFO mapred.LocalJobRunner: Waiting for map tasks
2024-12-14 12:13:20,242 INFO mapred.LocalJobRunner: Starting task: attempt_local1185818458_0001_m_000000_0
2024-12-14 12:13:20,380 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-12-14 12:13:20,381 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2024-12-14 12:13:20,513 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
2024-12-14 12:13:20,580 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/MDS2024/LAB7/Employee:0+1459
2024-12-14 12:13:20,736 INFO mapred.MapTask: numReduceTasks: 1
2024-12-14 12:13:20,917 INFO mapreduce.Job: Job job_local1185818458_0001 running in uber mode : false
2024-12-14 12:13:20,918 INFO mapreduce.Job:  map 0% reduce 0%
2024-12-14 12:13:20,974 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2024-12-14 12:13:20,974 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2024-12-14 12:13:20,974 INFO mapred.MapTask: soft limit at 83886080
2024-12-14 12:13:20,974 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2024-12-14 12:13:20,975 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
2024-12-14 12:13:20,984 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
2024-12-14 12:13:20,995 INFO streaming.PipeMapRed: PipeMapRed exec [/home/hadoop/Labs/LAB7/mapper.py]
2024-12-14 12:13:21,006 INFO Configuration.deprecation: mapred.work.output.dir is deprecated. Instead, use mapreduce.task.output.dir
2024-12-14 12:13:21,008 INFO Configuration.deprecation: mapred.local.dir is deprecated. Instead, use mapreduce.cluster.local.dir
2024-12-14 12:13:21,008 INFO Configuration.deprecation: map.input.file is deprecated. Instead, use mapreduce.map.input.file
2024-12-14 12:13:21,009 INFO Configuration.deprecation: map.input.length is deprecated. Instead, use mapreduce.map.input.length
2024-12-14 12:13:21,009 INFO Configuration.deprecation: mapred.job.id is deprecated. Instead, use mapreduce.job.id
2024-12-14 12:13:21,010 INFO Configuration.deprecation: mapred.task.partition is deprecated. Instead, use mapreduce.task.partition
2024-12-14 12:13:21,012 INFO Configuration.deprecation: map.input.start is deprecated. Instead, use mapreduce.map.input.start
2024-12-14 12:13:21,012 INFO Configuration.deprecation: mapred.task.is.map is deprecated. Instead, use mapreduce.task.ismap
2024-12-14 12:13:21,013 INFO Configuration.deprecation: mapred.task.id is deprecated. Instead, use mapreduce.task.attempt.id
2024-12-14 12:13:21,014 INFO Configuration.deprecation: mapred.tip.id is deprecated. Instead, use mapreduce.task.id
2024-12-14 12:13:21,015 INFO Configuration.deprecation: mapred.skip.on is deprecated. Instead, use mapreduce.job.skiprecords
2024-12-14 12:13:21,016 INFO Configuration.deprecation: user.name is deprecated. Instead, use mapreduce.job.user.name
2024-12-14 12:13:21,322 INFO streaming.PipeMapRed: R/W/S=1/0/0 in:NA [rec/s] out:NA [rec/s]
2024-12-14 12:13:21,323 INFO streaming.PipeMapRed: R/W/S=10/0/0 in:NA [rec/s] out:NA [rec/s]
```

```
2024-12-14 12:13:22,812 INFO mapred.LocalJobRunner: Finishing task: attempt_local1185818458_0001_r_000000_0
2024-12-14 12:13:22,813 INFO mapred.LocalJobRunner: reduce task executor complete.
2024-12-14 12:13:22,988 INFO mapreduce.Job:  map 100% reduce 100%
2024-12-14 12:13:22,988 INFO mapreduce.Job: Job job_local1185818458_0001 completed successfully
2024-12-14 12:13:23,023 INFO mapreduce.Job: Counters: 36
        File System Counters
                FILE: Number of bytes read=293136
                FILE: Number of bytes written=1593908
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=2918
                HDFS: Number of bytes written=2520
                HDFS: Number of read operations=15
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=4
                HDFS: Number of bytes read erasure-coded=0
        Map-Reduce Framework
                Map input records=30
                Map output records=150
                Map output bytes=4806
                Map output materialized bytes=5112
                Input split bytes=95
                Combine input records=0
                Combine output records=0
                Reduce input groups=5
                Reduce shuffle bytes=5112
                Reduce input records=150
                Reduce output records=54
                Spilled Records=300
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=103
                Total committed heap usage (bytes)=1006632960
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=1459
        File Output Format Counters
                Bytes Written=2520
2024-12-14 12:13:23,029 INFO streaming.StreamJob: Output directory: /MDS2024/LAB7/output
```

~$ hadoop fs -ls /MDS2024/LAB7/output

```
hadoop@Ubuntu22:~$ hadoop fs -ls /MDS2024/LAB7/output
Found 2 items
-rw-r--r--   3 hadoop supergroup          0 2024-12-14 12:13 /MDS2024/LAB7/output/_SUCCESS
-rw-r--r--   3 hadoop supergroup       2520 2024-12-14 12:13 /MDS2024/LAB7/output/part-00000
```

~$ hadoop fs -cat /MDS2024/LAB7/output/part-00000

```
hadoop@Ubuntu22:~$ hadoop fs -cat /MDS2024/LAB7/output/part-00000

Average Salary by Department (Descending Order):

Department          Avg Salary
----------------------------------
Engineering         101375.00
Finance             82375.00
Marketing           62142.86
HR                  55285.71
```

```
Highest Paid Employee by Department:

Department          Employee Name       Salary
-------------------------------------------------
Engineering         Rajesh Reddy        115000
Finance             Abhishek Gupta      90000
Marketing           Neeraj Soni         67000
HR                  Swati Sharma        62000

Salary Distribution by Department:

Department          Min Salary     Max Salary     Median Salary
-----------------------------------------------------------
HR                  50000          62000          55000.00
Finance             75000          90000          82000.00
Engineering         90000          115000         99500.00
Marketing           58000          67000          62000.00

Gender-Based Salary Comparison by Department:

Department          Gender    Avg Salary
-------------------------------------------
HR                  Female    55800.00
HR                  Male      54000.00
Marketing           Female    59666.67
Marketing           Male      64000.00
Finance             Male      85000.00
Finance             Female    79750.00
Engineering         Male      102833.33
Engineering         Female    97000.00

Experience vs Salary Analysis (Descending Order):

Experience  Avg Salary  Max Salary  Min Salary  Max Dept       Min Dept
-------------------------------------------------------------------------
10          102500.00   115000      90000       Engineering    Finance
9           98666.67    110000      86000       Engineering    Finance
8           95500.00    105000      86000       Engineering    Finance
7           93250.00    99000       82000       Engineering    Finance
6           73166.67    90000       62000       Engineering    Marketing
5           64200.00    78000       57000       Finance        HR
4           62250.00    82000       53000       Finance        HR
3           56000.00    59000       54000       Marketing      HR
2           50000.00    50000       50000       HR             HR
```

✧ **OUTPUT:**

Average Salary by Department :

Department	Avg Salary
Engineering	101375.00
Finance	82375.00
Marketing	62142.86
HR	55285.71

Highest Paid Employee by Department:

Department	Employee Name	Salary
Engineering	Rajesh Reddy	115000
Finance	Abhishek Gupta	90000
Marketing	Neeraj Soni	67000
HR	Swati Sharma	62000

Salary Distribution by Department:

Department	Min Salary	Max Salary	Median Salary
HR	50000	62000	55000.00
Finance	75000	90000	82000.00
Engineering	90000	115000	99500.00
Marketing	58000	67000	62000.00

Gender-Based Salary Comparison by Department:

Department	Gender	Avg Salary
HR	Female	55800.00
HR	Male	54000.00
Marketing	Female	59666.67
Marketing	Male	64000.00
Finance	Male	85000.00
Finance	Female	79750.00
Engineering	Male	102833.33
Engineering	Female	97000.00

Experience vs Salary Analysis :

Experience	Avg Salary	Max Salary	Min Salary	Max Dept	Min Dept
10	102500.00	115000	90000	Engineering	Finance
9	98666.67	110000	86000	Engineering	Finance
8	95500.00	105000	86000	Engineering	Finance
7	93250.00	99000	82000	Engineering	Finance
6	73166.67	90000	62000	Engineering	Marketing
5	64200.00	78000	57000	Finance	HR

4	62250.00	82000	53000	Finance	HR
3	56000.00	59000	54000	Marketing	HR
2	50000.00	50000	50000	HR	HR

✧ CONCLUSION:

In this project, we successfully implemented a MapReduce framework to analyze a comprehensive employee salary dataset. The task involved several key objectives, including calculating average salary per department, identifying the highest-paid employees, analyzing gender-based salary disparities, and studying salary trends in relation to experience. By using the Mapper and Reducer scripts, we were able to efficiently process large datasets and generate meaningful insights.

The results from the analysis revealed interesting trends, such as departments with the highest average salaries, gender-based pay differences, and the correlation between years of experience and salary growth. The MapReduce approach allowed us to handle large volumes of data efficiently and deliver insights that are valuable for organizations looking to optimize their salary structures and ensure equity among employees.

Overall, this project demonstrates the power of distributed computing for data analysis and highlights the importance of data-driven decision-making in organizational salary management. The findings from this analysis can serve as a foundation for further studies and improvements in employee compensation strategies.