

# Big Data – Hadoop - Hive

Date: 02/13/2025

## Scenario Question:

You are a data analyst working for a company that manages a large workforce. Your manager has asked you to analyze the employee data to gather insights on employee performance, salary distribution, and department-wise headcount. You are given an employee dataset with the following columns: Employee\_ID, Gender, Marital\_Status, Education\_Level, Annual\_Salary, Department, Job\_Level, Years\_Experience, Bonus, Performance\_Rating, and City that needs to be uploaded to **Hive** (Hadoop) for further analysis.

## Task 1: Select Basic Data

- Your first task is to retrieve a list of all employees along with their key details. This includes their ID, gender, marital status, education level, annual salary, bonus, and performance rating.

The screenshot shows the Ambari Query Editor interface. On the left is the Database Explorer showing a tree view of databases and tables. The main area is the Query Editor, which contains a worksheet with the following SQL query:

```
1 SELECT employee_id, gender, marital_status, education_level, annual_salary, department, job_level, years_experience, bonus, performance_rating, city
2 FROM employee_dataset;
```

Below the query editor are buttons for 'Execute', 'Explain', 'Upload', and 'Save as...'. The 'Execute' button has been clicked, and the 'Query Process Results (Status: SUCCEEDED)' section is visible. It shows a table with 11 columns: employee\_id, gender, marital\_status, education\_level, annual\_salary, department, job\_level, years\_experience, bonus, performance\_rating, and city. The table contains 4 rows of data.

employee_id	gender	marital_status	education_level	annual_salary	department	job_level	years_experience	bonus	performance_rating	city
E1000	Male	Married	Bachelor's	108613	Sales	Mid-level	7	11610	2	Seattle
E1001	Female	Single	Bachelor's	105591	Marketing	Junior	13	15958	3	Chicago
E1002	Female	Married	Master's	113180	Marketing	Senior	25	12363	1	Seattle
E1003	Male	Single	High School	107622	Finance	Junior	15	15990	4	San Francisco

## Task 2: Filtering Data

- Next, you need to filter the data to find employees who have a **performance rating of 4 or higher**. This information will help you identify top performers in the organization.

The screenshot shows the Ambari web interface. On the left, the 'Database Explorer' shows a tree view of databases and tables. The 'employee\_dataset' table is selected, showing its schema: employee\_id (STRING), gender (STRING), marital\_status (STRING), education\_level (STRING), annual\_salary (INT), department (STRING), job\_level (STRING), years\_experience (INT), bonus (INT), performance\_rating (INT), and city (STRING). The 'Query Editor' on the right contains the following SQL query:

```
1 SELECT employee_id, performance_rating
2 FROM employee_dataset
3 WHERE performance_rating > 3;
```

Below the query editor, the 'Query Process Results' section shows the status 'SUCCEEDED'. The 'Results' tab is active, displaying a table with the following data:

employee_id	performance_rating
E1003	4
E1004	5
E1006	5
E1008	4

## Task 3: Sorting Data

- You are then tasked with identifying the **top 5 highest-paid employees** in the company.
- Sorting the employees by their salary in descending order will give you this information.

The screenshot shows the Ambari web interface. The 'Query Editor' contains the following SQL query:

```
1 SELECT employee_id, annual_salary
2 FROM employee_dataset
3 ORDER BY annual_salary DESC
4 LIMIT 5;
```

Below the query editor, the 'Query Process Results' section shows the status 'SUCCEEDED'. The 'Results' tab is active, displaying a table with the following data:

employee_id	annual_salary
E1268	119947
E1528	119925
E1335	119825
E1099	119670
E1795	119403

## Task 4: Salary Statistics

- Your manager is interested in understanding the distribution of employee salaries. You are required to retrieve the **minimum**, **maximum**, and **average salary** from the dataset to provide a general overview of the company's salary range.

The screenshot shows the Ambari Query Editor interface. On the left, the Database Explorer shows the 'default' database with a table named 'employee\_dataset'. The table schema includes columns like employee\_id, gender, marital\_status, education\_level, annual\_salary, department, job\_level, years\_experience, bonus, performance\_rating, and city. The main Query Editor window contains the following SQL query:

```
1 SELECT MIN(annual_salary) AS Min_annual_salary,
2        MAX(annual_salary) AS Max_annual_salary,
3        AVG(annual_salary) AS Avg_annual_salary
4 FROM employee_dataset;
```

The query has been executed successfully, as indicated by the 'Query Process Results (Status: SUCCEEDED)' message. The results are displayed in a table with the following data:

min_annual_salary	max_annual_salary	avg_annual_salary
40043	119947	80095.267

## Task 5: Salary Category

- To make the data more insightful, you are asked to categorize employees into "Low," "Medium," and "High" salary groups based on their annual salary. This categorization will be used in further analysis.

The screenshot shows the Ambari Query Editor interface. On the left, the Database Explorer shows the 'default' database with a table named 'employee\_dataset'. The main Query Editor window contains the following SQL query:

```
1 SELECT employee_id, annual_salary,
2        CASE
3          WHEN annual_salary < 100000 THEN 'Low'
4          WHEN annual_salary BETWEEN 100000 AND 105000 THEN 'Medium'
5          ELSE 'High'
6        END AS Salary_Category
7 FROM employee_dataset;
```

The query has been executed successfully, as indicated by the 'Query Process Results (Status: SUCCEEDED)' message. The results are displayed in a table with the following data:

employee_id	annual_salary	salary_category
E1000	108613	High
E1001	105591	High
E1002	113180	High
E1003	107622	High
E1004	118385	High
E1005	107848	High
E1006	78210	Low

## Task 6: Grouping by Department

- Finally, your manager wants to see how employees are distributed across different **departments**. You will need to group the data by department and count the number of employees in each department.

The screenshot displays a web-based database interface with the following components:

- Database Explorer:** Shows a tree view of databases. The 'default' database is selected, showing tables like 'employee\_dataset', 'employee\_id', 'gender', 'marital\_status', 'education\_level', 'annual\_salary', 'department', 'job\_level', 'years\_experience', 'bonus', 'performance\_rating', and 'city'.
- Query Editor:** Contains a SQL query:

```
1 SELECT department, COUNT(employee_id) AS NumberOfEmployees
2 FROM employee_dataset
3 GROUP BY department;
```
- Query Process Results (Status: SUCCEEDED):** Displays the results of the query in a table format.

department	numberofemployees
Finance	199
HR	213
IT	214
Marketing	201
Sales	173