**DOCUMENTATION**

**Objective:**

Calculate the occupancy rate of a selected node within a specified date range using job data stored in a DataFrame.

**Steps:**

1. **Data Preparation:**
   * Imported required libraries: **numpy** and **pandas**.
   * Loaded job data stored in a SAS7BDAT file using the **sas.read\_sas** function and processed it in chunks.
   * Concatenated the processed chunks into a single DataFrame named **slurm**.
   * Extracted relevant columns from **slurm** DataFrame to create the **job\_state** DataFrame.
2. **Filtering Data:**
   * Filtered the **job\_state** DataFrame to include data only within the specified date range (September 1, 2020 to February 28, 2023).
   * Reset the index of the filtered **job\_state** DataFrame.
3. **Occupancy Rate Calculation Functions:**
   * Created the **generate\_matrix** function to generate a matrix representing the availability of cores for a selected node on a selected date.
   * The **generate\_matrix** function takes inputs of **job\_state**, **selected\_node**, and **selected\_date** to filter and process data.
   * The function calculates available cores based on job data and generates a matrix where occupied cores are marked with 1.
   * Created the **calculate\_occupancy\_rate** function to calculate the occupancy rate from a given matrix.
   * The **calculate\_occupancy\_rate** function calculates the sum of occupied cores and divides it by the total possible cores and time.
4. **User Interaction and Occupancy Rate Calculation:**
   * Used user inputs to specify the selected node, start date, and end date for the analysis.
   * Filtered the **job\_state** DataFrame further based on the selected node.
   * Printed the available partitions for the selected node.
   * Iterated over the date range and calculated matrices for each day using the **generate\_matrix** function.
   * Concatenated matrices to create a combined matrix for the entire date range.
   * Calculated the occupancy rate using the **calculate\_occupancy\_rate** function.
5. **Output:**
   * Printed the available partitions for the selected node.
   * Printed the calculated occupancy rate as a percentage for the specified date range.

**Key Takeaways:**

* We imported necessary libraries for data manipulation and analysis.
* The job data was loaded from a SAS7BDAT file and processed in chunks to create the **job\_state** DataFrame.
* The data was filtered based on the specified date range and then used for occupancy rate calculations.
* Custom functions were created for generating matrices and calculating occupancy rates.
* User inputs were utilized for specifying the selected node and date range.
* The occupancy rate was calculated and presented as the output.

**Note:**

* You can replace the file names in the code with the actual paths to your data files.
* The provided code is a simplified example. Depending on the complexity and specifics of your data, you might need to make further adjustments.

**Matrix Creation:**

The matrix is a fundamental part of the occupancy rate calculation process. It's a representation of the availability of computational resources (cores) for a given node and a specific date. Each row in the matrix represents a core, and each column represents a second within a 24-hour period.

* A matrix cell with a value of 0 represents an available core at a particular second.
* A matrix cell with a value of 1 represents an occupied core at a particular second.

The idea is to populate this matrix based on the jobs that ran on the selected node for the chosen date. As jobs are executed on cores, the matrix will reflect which cores were occupied during specific time intervals.

**Occupancy Rate Calculation:**

The occupancy rate is a measure of how busy the selected node's cores were during a given time period. It provides insights into resource utilization and system efficiency. A higher occupancy rate implies that a larger portion of the available computing resources (CPU cores) were utilized during the selected time interval. Conversely, a lower occupancy rate indicates that a significant portion of CPU cores remained idle and were not utilized by any jobs. The occupancy rate is a valuable metric for system administrators and resource managers to understand how effectively their computing resources are being utilized. It can help in optimizing job scheduling, resource allocation, and system performance.

To calculate the occupancy rate:

1. **Matrix Processing:**

* The matrix is processed column-wise. Each column represents a second, and we count the number of non-zero elements (occupied cores) in each column.
* This gives us the count of occupied cores at each second during the time period.

1. **Summation and Total Cores:**

* We sum up the counts of occupied cores across all columns (seconds) to get the total number of occupied core-seconds during the specified time range.

1. **Calculation:**

* The occupancy rate is calculated by dividing the total number of occupied core-seconds by the total number of core-seconds available in the system (max\_cores \* time\_of\_interest).
* **max\_cores** represents the total number of cores in the node, and **time\_of\_interest** is the length of the time period being analyzed.

1. **Result:**

* The occupancy rate is a value between 0 and 1, typically multiplied by 100 to express it as a percentage.
* A higher occupancy rate indicates that the cores were heavily utilized during the analyzed time period, while a lower occupancy rate suggests that the cores were relatively less busy.

**Intuition:**

* The matrix provides a visual representation of how core utilization changes over time.
* The occupancy rate is a quantitative measure that summarizes the utilization trend and helps identify peak periods of activity.
* It's important for resource planning, capacity management, and understanding how efficiently computational resources are being used.

**Explanation:**

1. **Date Range Calculation:**

* The function starts by calculating a 30-day date range before and after the selected date. This range is used to filter jobs that fall within this time frame.

1. **Data Filtering:**

* The **job\_state\_window** dataframe is created by filtering the main **job\_state** dataframe to select rows where the **NodeList** matches the selected node and the job start and end dates fall within the calculated date range.

1. **Further Filtering:**

* From the **job\_state\_window**, a **job\_state\_subset** is created by applying additional filtering based on the selected date. This dataframe includes jobs that were active during the selected day.

1. **Matrix Initialization:**

* The **matrix** is initialized as a numpy array of zeros, with dimensions representing the maximum number of cores and the number of seconds in a day.

1. **Processing Jobs:**

* The **job\_state\_subset** is sorted based on the job start time.
* For each job, the start and end times are truncated to align with the selected day. Then, the core utilization period within the selected day is determined.

1. **Core Availability Check:**

* The function checks if the required cores are available during the specified time period for the job. If they are, the matrix cells representing these cores and time intervals are marked as occupied (1).

1. **Matrix Return:**

* The final matrix represents core availability and occupation during the selected day. It will have 0s for available cores and 1s for occupied cores at each second.

This function essentially populates the matrix to show how cores were utilized during the selected day for the chosen node. The matrix will have 1s wherever jobs occupied cores and 0s where cores were available but not used. This matrix is then used to calculate the occupancy rate as discussed earlier. Top of Form

Top of Form