2879. Display the First Three Rows

Easy

Problem Description

In this problem, we're working with a `DataFrame` named `employees` that has four columns with specific data types:

- `employee_id` which is of type `int`,
- `name` which is of type `object`, indicating it could be a string or a mix of different types,
- `department` which is also an `object`,
- and `salary` which is an `int`.

The task is to create a solution that will display the first `3` rows of this DataFrame. This means we need to write a function that takes this DataFrame as an input and returns a new DataFrame composed only of the first three entries from the original.

Intuition

When dealing with DataFrames in Python, the Pandas library is the go-to tool as it provides extensive functionalities for data manipulation and analysis. One of the basic methods available in Pandas for DataFrame objects is the `.head()` method.

The `.head()` method is used to retrieve the top `n` rows from a DataFrame, where `n` defaults to `5` when no argument is provided. Since in this case, we are interested in getting just the first three rows, we can simply call `employees.head(3)`. This line of code will return a new DataFrame with only the first three rows of the `employees` DataFrame.

Hence, the solution approach is straightforward:

- Import the Pandas library to be able to work with DataFrames.
- Define a function `selectFirstRows()` that accepts the `employees` DataFrame as a parameter.
- Inside the function, use the `.head()` method on the `employees` DataFrame with `3` as an argument to extract the first three rows.
- Return this subset of the DataFrame.

Solution Approach

The implementation of the solution for this particular problem is very straightforward because it leverages the built-in functionality provided by the Pandas library, rather than requiring a complex algorithm or data structure. Here's a step-by-step explanation:

- First, we import the Pandas library, which is a powerful tool for data manipulation in Python. It's standard to import Pandas and give it the alias `pd`, which is what we see in the solution `import pandas as pd`.
- Next, we define a function `selectFirstRows()`, which is our solution function. It expects one argument, `employees`, which is a DataFrame that we want to process.
- Inside the function, we use the Pandas DataFrame `.head()` method. The method `.head(n)` returns the first `n` rows of a DataFrame. By calling `employees.head(3)`, we are asking for the first three rows of the `employees` DataFrame.

This is the complete function:

```
def selectFirstRows(employees: pd.DataFrame) -> pd.DataFrame:
    return employees.head(3)
```

The algorithm and pattern used here is direct and makes use of the high-level abstractions provided by Pandas for common data operations. Since the task does not require any conditional logic or iteration that would need to be explicitly programmed, we do not need to delve into more complex data structures or algorithms.

The function will output a new DataFrame object that contains only the first three rows of the `employees` DataFrame, maintaining the same column structure: `employee_id`, `name`, `department`, and `salary`.

In terms of computational complexity, this operation is usually O(1), constant time, as it simply returns a view of the first few rows and does not involve any re-computation of data.

Example Walkthrough

Let's consider a scenario where we have a `DataFrame` called `employees` that looks like this:

employee_id	name	department	salary
1	Alice	Engineering	70000
2	Bob	Marketing	60000
3	Charlie	Sales	50000
4	Dana	HR	80000
5	Eve	Engineering	90000

Our goal is to create a function that, when given this DataFrame, will return a new DataFrame consisting only of the first three rows. We'll use Python along with the Pandas library to achieve this.

Here is a step-by-step explanation using the given `employees` DataFrame:

1. First, we import the Pandas library using `import pandas as pd`.
2. Then, we define our function `selectFirstRows()` which will accept one argument, the `employees` DataFrame. The function signature will look like this: `def selectFirstRows(employees: pd.DataFrame) -> pd.DataFrame`.
3. Inside this function, we will call the `.head()` method on the `employees` DataFrame. Since we need the first three rows, we pass the integer `3` as an argument to `.head()`, which will be `employees.head(3)`.
4. Finally, the function will return the result of `employees.head(3)`, which is the new DataFrame containing the first three rows of `employees`.

Applying the function `selectFirstRows()` to our `employees` DataFrame will look like this:

```
import pandas as pd

# The DataFrame 'employees' is defined as shown in the example above
employees = pd.DataFrame({
    'employee_id': [1, 2, 3, 4, 5],
    'name': ['Alice', 'Bob', 'Charlie', 'Dana', 'Eve'],
    'department': ['Engineering', 'Marketing', 'Sales', 'HR', 'Engineering'],
    'salary': [70000, 60000, 50000, 80000, 90000]
})

# Definition of our function
def selectFirstRows(employees: pd.DataFrame) -> pd.DataFrame:
    return employees.head(3)

# Calling the function with our DataFrame
first_three_employees = selectFirstRows(employees)

# The 'first three employees' DataFrame now holds:
# | employee_id | name     | department  | salary |
# |-------------|----------|-------------|--------|
# | 1           | Alice    | Engineering | 70000  |
# | 2           | Bob      | Marketing   | 60000  |
# | 3           | Charlie  | Sales       | 50000  |
```

In this case, the output we get from `first_three_employees` is exactly as we expect: the top three entries from our original `employees` DataFrame, maintaining the integrity of the data's structure.

Solution Implementation

Python

```
import pandas as pd

def selectFirstRows(employees_df: pd.DataFrame) -> pd.DataFrame:
    # Select and return the first three rows of the DataFrame
    return employees_df.head(3)
```

Java

```
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

public class EmployeeSelector {

    /**
     * Selects and returns the first three rows of the employee data.
     * This method assumes that there is a List of Maps where each Map represents
     * a row in a DataFrame, with the key being the column name and the value being the cell data.
     *
     * @param employeesData List of Maps representing employee data.
     * @return A List containing the first three Maps (rows) of the employee data.
     */
    public List<Map<String, Object>> selectFirstRows(List<Map<String, Object>> employeesData) {
        // Check if employeesData is large enough; if not, return the original list
        if (employeesData.size() <= 3) {
            return employeesData;
        }

        // Return the first three elements of the List using stream
        return employeesData.stream()
                            .limit(3)
                            .collect(Collectors.toList());
    }
}
```

C++

```
#include <iostream>
// Assume a DataFrame class that stores employee records and provides a head() function similar to pandas
class DataFrame {
public:
    // Constructor, destructor, and other necessary methods would go here

    // Method to get first N rows of the DataFrame
    DataFrame head(int n) {
        // Implementation would go here
        // For now, let's assume it returns a new DataFrame with the first n rows
        return DataFrame(); // Placeholder
    }
};

// Function that selects and returns the first three rows of a DataFrame
DataFrame selectFirstRows(const DataFrame& employeesDf) {
    // Select and return the first three rows of the DataFrame
    return employeesDf.head(3);
}

// The rest of your C++ code would go here...
```

TypeScript

```
// Assuming the use of a library similar to pandas in TypeScript for DataFrame operations,
// like danfo.js, because TypeScript/JavaScript does not have a native DataFrame type

import { DataFrame } from 'danfojs-node'; // Replace with the appropriate import based on the DataFrame library used

// Function to select the first three rows of a DataFrame
function selectFirstRows(employeesDf: DataFrame): DataFrame {
  // Select and return the first three rows of the employees DataFrame
  const firstThreeRows: DataFrame = employeesDf.head(3);

  return firstThreeRows;
}

// Usage of the function assumes that DataFrame is populated
// For example:
// let employeesDf = new DataFrame({ // Data populated here });
// let firstRows = selectFirstRows(employeesDf);
// firstRows.print(); // This would be the equivalent of viewing the DataFrame in a Python context
```

```
import pandas as pd

def selectFirstRows(employees_df: pd.DataFrame) -> pd.DataFrame:
    # Select and return the first three rows of the DataFrame
    return employees_df.head(3)
```

Time and Space Complexity

The time complexity of the `selectFirstRows` function is `O(1)` because retrieving the first few rows of a dataframe is a constant time operation. It does not depend on the size of the dataframe, as the number of rows to retrieve is always fixed at 3.