Exercise: Working with Employee Data in PySpark

This exercise demonstrates how to work with a sample employee dataset using PySpark.
It includes tasks such as filtering data, calculating averages, sorting, and adding new columns.
Below is the PySpark script along with explanations and solutions for each task.

PySpark Code

```
from pyspark.sql import SparkSession

# Initialize a Spark session
spark = SparkSession.builder.appName("Employee Data Analysis" .getOrCreate()

# Sample employee data
data = [
   (1, 'Arjun', 'IT', 75000),
   (2, 'Vijay', 'Finance', 85000),
   (3, 'Shalini', 'IT', 90000),
   (4, 'Sneha', 'HR', 50000),
   (5, 'Rahul', 'Finance', 60000),
   (6, 'Amit', 'IT', 55000)
]

# Define schema (columns)
columns = ['EmployeeID', 'EmployeeName', 'Department', 'Salary']

# Create DataFrame
employee_df = spark.createDataFrame(data, columns)

# Show the DataFrame
employee_df.show()
```

Tasks and Explanations

Task 1: Filter Employees by Salary

Objective: Filter employees with a salary greater than 60,000.

Code:

```
high_salary_df = employee_df.filter(employee_df['Salary'] > 60000)
high_salary_df.show()
```

Task 2: Calculate the Average Salary by Department

Objective: Group the employees by department and calculate the average salary for each department.

Code:

```
avg_salary_by_dept_df = employee_df.groupBy('Department').avg('Salary')
avg_salary_by_dept_df.show()
```

Task 3: Sort Employees by Salary

Objective: Sort the employees in descending order of their salary.

Code:

```
sorted_df = employee_df.orderBy(employee_df['Salary'].desc())
sorted_df.show()
```

Task 4: Add a Bonus Column

Objective: Add a new column called `Bonus`, which should be 10% of the employee's salary.

Code:

```
bonus_df = employee_df.withColumn('Bonus', employee_df['Salary'] * 0.1)
bonus_df.show()
```