Teste de Conhecimentos em PySpark

Parte 1: Manipulação de Dados

Criação de DataFrame

Crie um DataFrame a partir do seguinte conjunto de dados:

```
data = [
   ("Alice", 34, "Data Scientist"),
   ("Bob", 45, "Data Engineer"),
   ("Cathy", 29, "Data Analyst"),
   ("David", 35, "Data Scientist")
]
columns = ["Name", "Age", "Occupation"]
```

Filtragem e Seleção

Selecione apenas as colunas "Name" e "Age" do DataFrame criado.

Filtre as linhas onde a "Age" é maior que 30.

Agrupamento e Agregação

Agrupe os dados pelo campo "Occupation" e calcule a média de "Age" para cada grupo.

Ordenação

Ordene o DataFrame resultante da questão anterior pela média de "Age" em ordem decrescente.

Parte 2: Funções Avançadas

Uso de UDFs (User Defined Functions)

Crie uma função em Python que converte idades para categorias:

Menor que 30: "Jovem"

Entre 30 e 40: "Adulto"

Maior que 40: "Senior"

Aplique essa função ao DataFrame usando uma UDF.

Funções de Janela

Use funções de janela para adicionar uma coluna que mostre a diferença de idade entre cada indivíduo e a média de idade do seu "Occupation".

Parte 3: Performance e Otimização

Particionamento

Explique como o particionamento pode ser usado para melhorar a performance em operações de leitura e escrita de dados em PySpark. Dê um exemplo de código que particiona um DataFrame por uma coluna específica.

Broadcast Join

Descreva o conceito de Broadcast Join em PySpark e como ele pode ser usado para otimizar operações de join. Implemente um exemplo de Broadcast Join entre dois DataFrames.

Parte 4: Integração com Outras Tecnologias

Leitura e Escrita de Dados

Demonstre como ler dados de um arquivo CSV e escrever o resultado em um formato Parquet.

Integração com Hadoop

Explique como PySpark se integra com o Hadoop HDFS para leitura e escrita de dados. Dê um exemplo de código que leia um arquivo do HDFS e salve o resultado de volta no HDFS.

Parte 5: Problema de Caso

Processamento de Logs

Considere que você tem um grande arquivo de log com as seguintes colunas: "timestamp", "user_id", "action". Cada linha representa uma ação realizada por um usuário em um determinado momento.

Carregue o arquivo de log em um DataFrame.

Conte o número de ações realizadas por cada usuário.

Encontre os 10 usuários mais ativos.

Salve o resultado em um arquivo CSV.