Task: Optimize the query plan

Suppose we want to compose a query in which we get for each question the number of answers to this question for each month. See the query below which does that in a suboptimal way and try to rewrite it to achieve a more optimal plan.

```
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, count, month

import os

spark = SparkSession.builder.appName('Optimize I').getOrCreate()

base_path = os.getcwd()

project_path = ('/').join(base_path.split('/')[0:-3])

answers_input_path = os.path.join(project_path, 'data/answers')

questions_input_path = os.path.join(project_path, 'output/questions-transformed')

answersDF = spark.read.option('path', answers_input_path).load()

questionsDF = spark.read.option('path', questions_input_path).load()

'''
Answers aggregation

Here we : get number of answers per question per month
'''

answers_month = answersDF.withColumn('month',
month('creation_date')).groupBy('question_id', 'month').agg(count('*').alias('cnt'))

resultDF = questionsDF.join(answers_month, 'question_id').select('question_id',
'creation_date', 'title', 'month', 'cnt')

resultDF.orderBy('question_id', 'month').show()

'''
Task:

see the query plan of the previous result and rewrite the query to optimize it
'''
```

Tips

As you might remember from our spark subunit, there are several ways one can improve performance of a Spark job:

1. By picking the right operators
2. Reduce the number of shuffles and the amount of data shuffled
3. Tuning Resource Allocation
4. Tuning the Number of Partitions
5. Reducing the Size of Data Structures
6. Choosing Data Formats