Seekho Bigdata
Data is the New Oil

PySpark Scenario-Based Interview Questions

DAY 9 — Performance Optimization Basics

Karthik Kondpak
9989454737

PySpark Scenario-Based Interview Questions (Complete Notes Series)

DAY 9 — Performance Optimization Basics (Partitions, Cache, Broadcast)

Concepts Covered Today

- What is partitioning and why it matters
- Repartition vs Coalesce
- Caching & Persistence
- Broadcast joins
- Real-world optimization mindset

Sample Scenario: Large Orders Dataset

Assume orders_df has **500 million records** and is used multiple times in the pipeline.

✅ Question 1: Check Number of Partitions

🔹 Scenario

Before optimizing, you want to understand how data is distributed.

🧪 PySpark Solution

```
orders_df.rdd.getNumPartitions()
```

📝 Explanation

- Too few partitions → under-utilized cluster
- Too many partitions → task scheduling overhead
- Default comes from `spark.sql.shuffle.partitions`

✅ Question 2: Repartition Data for Parallel Processing

🔹 Scenario

Orders are heavily skewed. Repartition data to improve parallelism.

🧪 PySpark Solution

```
optimized_df = orders_df.repartition(200,
"customer_id")
```

📝 Explanation

- repartition() causes a **full shuffle**
- Used when increasing partitions or redistributing data

✅ Question 3: Reduce Partitions Using Coalesce

🔹 Scenario

After heavy filtering, data size reduces drastically. Reduce partitions efficiently.

🧪 PySpark Solution

```
final_df = optimized_df.coalesce(50)
```

📝 Explanation

- coalesce() avoids shuffle
- Best when **reducing** number of partitions

✅ Question 4: Cache Data Used Multiple Times

🔹 Scenario

The same dataset is used for **multiple aggregations and joins**.

🧪 PySpark Solution

```
orders_df.cache()
orders_ df.cou nt()#triggers cache
```

📝 Explanation

- Cache stores data in memory
- Avoids recomputation
- Use only when dataset is reused

✅ Question 5: Persist with Storage Level

🔹 Scenario

Dataset is too large to fit fully in memory.

🧪 PySpark Solution

```
from pyspark import StorageLevel

orders_df.persist(StorageLevel.MEMORY_AND_DISK)
```

📝 Explanation

- Prevents OOM errors

- Frequently asked conceptual question

Question 6: Optimize Join Using Broadcast

🔹 Scenario

`custome rs_df`is small (50K rows) and joined with massive `orders_df.`

🧪 PySpark Solution

```
from pyspark.sql.functions import broadcast

optimized_join_df = orders_df.join(
    broadcast(customers_df),
    "customer_id",
    "inner"
)
```

📝 **Explanation**

- Broadcast avoids shuffle
- One of the **most important interview answers**

Seekho Bigdata

Data is the New Oil

Let's build your Data Engineering journey together!

📩 Call us directly at: 9989454737

🌐 https://seekhobigdata.com/