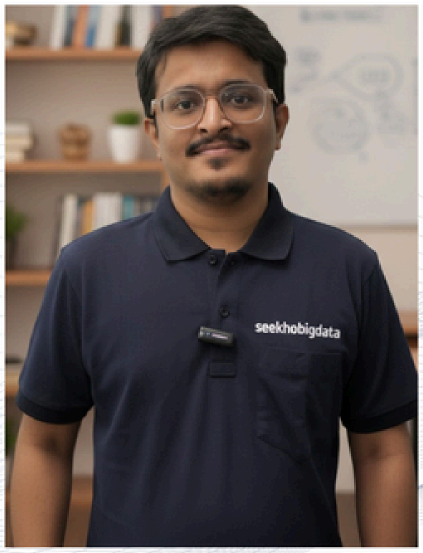
Seekho Bigdata
Data is the New Oil

PySpark Scenario-Based Interview Questions (Complete Notes Series)

DAY 10 — Spark SQL & Catalyst Optimizer (Execution Plans)



Karthik Kondpak
9989454737

PySpark Scenario-Based Interview Questions (Complete Notes Series)

DAY 10 — Spark SQL & Catalyst Optimizer (Execution Plans)

Concepts Covered Today

- Spark SQL vs DataFrame API
- Catalyst Optimizer (Logical → Optimized → Physical Plan)
- explain() (Simple, Extended, Formatted)
- Predicate Pushdown & Column Pruning
- Join strategies (Broadcast / Sort-Merge / Shuffle Hash)

Scenario

You are working on an **Indian e-commerce analytics platform** (Flipkart-like).

Tables:

- orders (1 billion rows)
- customers (5 million rows)
- products (50K rows)

Goal: Build **monthly revenue by city** efficiently.

✅ Question 1: DataFrame API vs Spark SQL — Which is Faster?

🔹 Interview Question

Is Spark SQL faster than DataFrame API?

Correct Answer

Both use **Catalyst Optimizer** → performance is the same **if logic is identical**.

Performance difference comes from **bad logic**, not API choice.

✅ Question 2: View Logical & Physical Plans

🔹 Scenario

Job is slow. You want to inspect execution plan.

🧪 PySpark Solution

```
monthly_revenue_df.explain("formatted")
```

📝 What to Look For

- Filters pushed before joins
- Join type selected
- Number of shuffles

Catalyst Optimizer Flow (Interview Must-Explain)

Unresolved Logical Plan – SQL/DataFrame parsed

Analyzed Logical Plan – Columns & tables resolved

Optimized Logical Plan – Predicate pushdown, column pruning

Physical Plan – Actual execution strategy

Interview tip: Always mention **4 phases**.

✅ Question 3: Predicate Pushdown Example

🔹 Scenario

Fetch only **2024 orders** from Parquet.

🧪 Bad Code (No Pushdown)

```
df.filter(year(col("order_date")) == 2024)
```

🧪 Optimized Code

```
df.filter(col("order_date") >= "2024-01-01")
```

📝 Why Faster?

- Filter applied at **storage layer**
- Fewer rows read from disk

✅ Question 4: Column Pruning

🔹 Scenario

Orders table has 50 columns, but only 3 are needed.

🧪 Optimized Code

```
df.select("order_id", "city", "amount")
```

📝 Explanation

- Spark reads only required columns
- Massive IO reduction

Question 5: Understanding Join Strategies

🔹 Scenario

Join `orders` with `products`.

Execution Plan Insight

```
BroadcastHashJoin  ← Best case
SortMergeJoin     ← Default for large tables
ShuffleHashJoin   ← Rare
```

📝 Interview Explanation

- Broadcast if one table is small
- Sort-Merge for large-large joins

Seekho Bigdata

Data is the New Oil

Let's build your Data Engineering journey together!

📩 Call us directly at: 9989454737

🌐 https://seekhobigdata.com/