09 Advanced PySpark

Advanced PySpark: Spark SQL, DataFrame APIs, Aggregations, Window Functions & Joins

Basic Level Questions (1-10)

1. **What is the difference between a DataFrame and an RDD in PySpark?** *Focus on: Structured vs unstructured data, optimization benefits, ease of use, Catalyst optimizer*
2. **How do you create a DataFrame from a Python list or dictionary in PySpark?** *Focus on: spark.createDataFrame(), schema definition, data type inference*
3. **What is the Catalyst optimizer and how does it benefit DataFrame operations?** *Focus on: Query planning, predicate pushdown, column pruning, cost-based optimization*
4. **Explain the difference between transformation and action operations in PySpark.** *Focus on: Lazy evaluation, examples of each type, execution triggers*
5. **How do you perform basic filtering operations on a DataFrame?** *Focus on: filter() vs where(), column expressions, multiple conditions*
6. **What are the different ways to select columns from a DataFrame?** *Focus on: select(), col(), column expressions, renaming columns*
7. **How do you handle null values in PySpark DataFrames?** *Focus on: dropna(), fillna(), isNull(), isNotNull(), coalesce()*
8. **What is the difference between collect() and take() actions?** *Focus on: Memory implications, use cases, performance considerations*
9. **How do you perform basic aggregations like sum, count, and average?** *Focus on: agg(), groupBy(), built-in functions*
10. **What is a window function and when would you use it?** *Focus on: Analytical functions, partitioning, ordering, frame specifications*

Intermediate Level Questions (11-20)

11. **How do you optimize join operations in PySpark for better performance?** *Focus on: Broadcast joins, bucketing, salting, join strategies, data skewness*
12. **Explain the difference between map() and mapPartitions() transformations.** *Focus on: Per-record vs per-partition processing, performance implications, use cases*
13. **How would you implement a custom aggregation function using UDAF?** *Focus on: User-defined aggregate functions, state management, performance considerations*
14. **What are the different types of window functions and how do you use them?** *Focus on: Ranking functions, analytical functions, frame specifications (ROWS, RANGE)*
15. **How do you handle data skewness in joins and aggregations?** *Focus on: Salting techniques, broadcast variables, repartitioning strategies*
16. **Explain the concept of partitioning and bucketing in Spark.** *Focus on: Data locality, shuffle reduction, performance optimization*
17. **How do you implement complex nested aggregations using DataFrame API?** *Focus on: Multiple groupBy levels, struct columns, array aggregations*
18. **What is the difference between inner, outer, left, and right joins in Spark?** *Focus on: Data matching behavior, null handling, performance characteristics*
19. **How do you use window functions to calculate running totals and moving averages?** *Focus on: Frame specifications, ordering, partitioning strategies*
20. **How do you convert between DataFrame and SQL representations?** *Focus on: createOrReplaceTempView(), spark.sql(), mixing APIs*

Advanced/Difficult Level Questions (21-30)

21. How would you implement a complex ETL pipeline that handles incremental updates using advanced DataFrame operations? *Focus on: Change data capture patterns, merge operations, state management, checkpointing*

22. Design a solution for handling deeply nested JSON data with dynamic schemas using PySpark. *Focus on: Schema inference, explode functions, struct operations, performance optimization*

23. How would you implement a custom partitioner for optimal data distribution in a highly skewed dataset? *Focus on: Custom partitioning logic, hash functions, load balancing, performance tuning*

24. Explain how you would optimize a complex multi-way join involving 5+ large tables. *Focus on: Join order optimization, broadcast strategies, bucketing, query planning*

25. How do you implement complex analytical functions like percentiles, NTILE, and LAG/LEAD across different partitions? *Focus on: Window specifications, frame boundaries, null handling, performance optimization*

26. Design a real-time streaming solution using PySpark that handles late-arriving data with watermarking. *Focus on: Structured streaming, watermarks, event time processing, state management*

27. How would you implement a custom data source connector for reading from a proprietary database? *Focus on: DataSource API, partition strategies, predicate pushdown, schema inference*

28. Explain how you would troubleshoot and optimize a PySpark job that's experiencing frequent OOM errors. *Focus on: Memory management, garbage collection, partition sizing, broadcast variables*

29. How do you implement complex data validation and quality checks using advanced DataFrame operations? *Focus on: Statistical functions, anomaly detection, constraint validation, reporting*

30. Design a solution for processing time-series data with complex temporal aggregations and gap filling. *Focus on: Time-based windows, interpolation techniques, temporal joins, performance optimization*

Practical Coding Scenarios

Scenario-Based Questions

Scenario A: Given a DataFrame with user transaction data, write PySpark code to find the top 3 products purchased by each user in the last 30 days.

Scenario B: You have two large datasets that need to be joined, but one has significant data skew. Show how you would handle this.

Scenario C: Implement a sliding window calculation to compute 7-day moving averages for multiple metrics simultaneously.

Scenario D: Write a complex aggregation that calculates both cumulative and period-over-period metrics in a single operation.