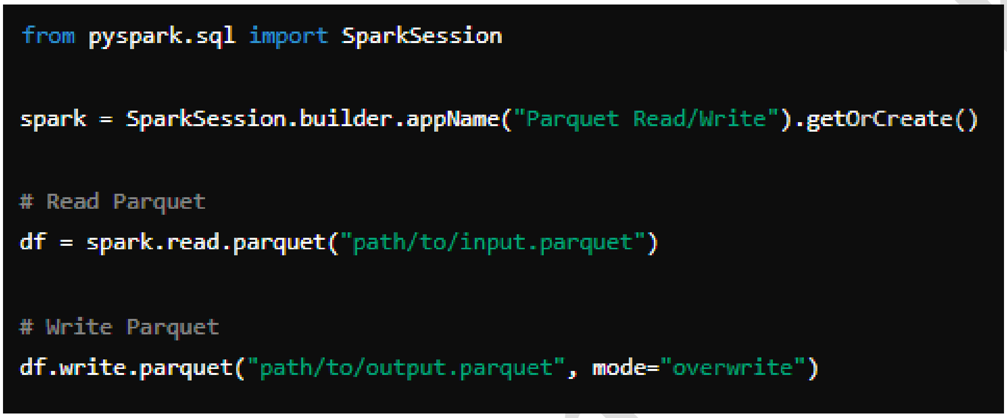
Persistent Systems AWS Data Engineer Interview Guide – Experienced 3+

Round 1: Technical Coding and Conceptual Questions

Duration: 60 minutes
This round focused on PySpark coding, SQL queries, Spark internals, and performance optimization techniques.

Coding Tasks and Solutions

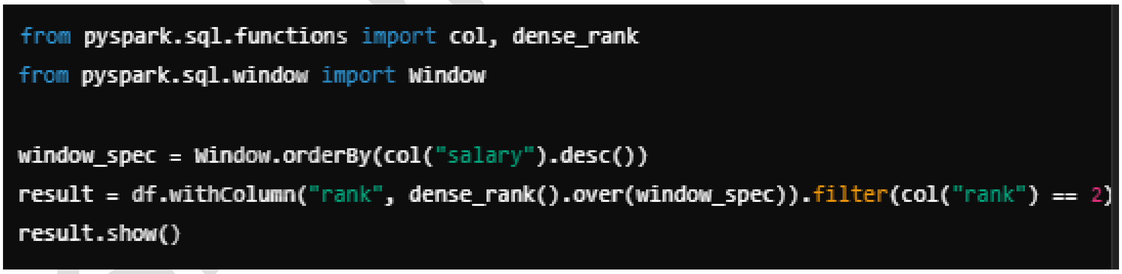
1. **Read and Write Data in Parquet Format (PySpark)**



```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("Parquet Read/Write").getOrCreate()

# Read Parquet
df = spark.read.parquet("path/to/input.parquet")

# Write Parquet
df.write.parquet("path/to/output.parquet", mode="overwrite")
```

This task tested my knowledge of **I/O operations** with Parquet, a popular storage format.

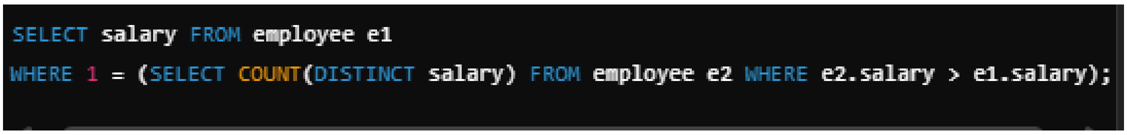
2. **Find 2nd Highest Salary in PySpark**



```
from pyspark.sql.functions import col, dense_rank
from pyspark.sql.window import Window

window_spec = Window.orderBy(col("salary").desc())
result = df.withColumn("rank", dense_rank().over(window_spec)).filter(col("rank") == 2)
result.show()
```

Using **dense_rank()** avoids gaps in ranking caused by duplicate values, ensuring accurate results.

3. **SQL Query for 2nd Highest Salary without Using MAX()**



```
SELECT salary FROM employee e1
WHERE 1 = (SELECT COUNT(DISTINCT salary) FROM employee e2 WHERE e2.salary > e1.salary);
```

This query uses a **correlated subquery** to count distinct salaries greater than the current one.

Conceptual Questions and Explanations

1. **Partitions in Spark**
   - Default Partition Size: 128 MB
   - Default Number of Partitions: Depends on the cluster and input data; typically based on the size of data and block size.
   - Partition Creation: Partitions are created based on input splits, usually determined by the data source (like HDFS).

2. **DAG Creation and Execution in Spark**
   - Stages and Tasks: After submitting a Spark application, the driver program creates a DAG (Directed Acyclic Graph), breaking it into stages based on shuffle boundaries. Each stage consists of multiple tasks.
   - DAG Scheduler submits stages to the Task Scheduler, which assigns tasks to executors.

3. **Monitoring Spark Applications**
   - Use the Spark UI to view stages, tasks, shuffle read/write sizes, and job progress.
   - Tools like Ganglia, Grafana, or Cloud-specific monitoring services (like AWS CloudWatch for EMR) are commonly used.

4. **Optimization Techniques**
   - Hive: Partitioning, bucketing, indexing, and query caching.
   - SQL: Use CTEs, avoid Cartesian products, use appropriate indexes, and optimize joins.
   - Spark: Broadcast joins, repartitioning vs. coalescing, cache/persist, and predicate pushdown.

5. **Adaptive Query Execution (AQE)**
   - AQE dynamically adjusts query plans during execution based on runtime metrics to optimize performance. It can change join strategies, repartition data, or skewed partitions handling.

6. **Catalyst Optimizer**
   - Spark's Catalyst Optimizer transforms logical query plans into optimized physical plans using techniques like predicate pushdown, column pruning, and reorder joins.

Round 2: Project Explanation (60 Minutes)

This round was project-centric, focusing on my ability to communicate and justify design decisions.

Key Discussion Points:

- **End-to-End Data Pipeline**
    - Data Ingestion: Explain how data was collected from multiple sources (Kafka, S3, or relational databases).
    - Processing Framework: Describe the use of Spark on EMR or Databricks for real-time or batch processing.
    - Data Storage: Explain why you chose specific storage (e.g., S3 for raw data, Redshift or Snowflake for analytics).
    - Orchestration: How Airflow or Step Functions managed task dependencies and retries.
    - Monitoring and Alerting: Using CloudWatch for metrics and alerts on failed jobs or resource spikes.

AWS Intermediate-Level Questions

- Explain the difference between S3 One Zone-IA and S3 Standard-IA.
- How would you implement cross-region replication for S3?
- What are partitioning strategies in Redshift?
- What's the role of Glue Data Catalog in Spark jobs?
- Explain AWS Lake Formation and its benefits.
- How does IAM role chaining work?
- What's the difference between RDS Read Replicas and Multi-AZ deployments?
- Explain Kinesis Data Firehose vs. Kinesis Data Streams.
- How do you handle cost optimization in AWS EMR clusters?
- Describe Amazon Athena and how it interacts with S3.
- What are provisioned throughput and auto-scaling in DynamoDB?
- How would you implement VPC peering between two AWS accounts?
- Describe step scaling policies vs. target tracking policies in AWS Auto Scaling.
- What are transient clusters in EMR, and when would you use them?
- How do you secure data at rest and in transit for AWS RDS?

Glassdoor Persistent System Review –

https://www.glassdoor.co.in/Reviews/Persistent-Systems-Reviews-E150639.htm

Persistent System Careers –

https://careers.persistent.com/

Subscribe to my YouTube Channel for Free Data Engineering Content –

https://www.youtube.com/@shubhamwadekar27

Connect with me here –

https://bento.me/shubhamwadekar

Checkout more Interview Preparation Material on –

https://topmate.io/shubham_wadekar