perplexity

Generate "spark 2 to Spark3 migration" demo PPT content ; make it concise and meaningful - Add why upgrade / mongo changes / pyspark changes / legacy settings etc

Migrating from Spark 2.x to Spark 3.x involves significant performance improvements, API changes, and configuration updates. Below is a structured outline for a concise PPT presentation:

Why Upgrade to Spark 3.x?

- **Performance Gains**:
  - Adaptive Query Execution (AQE) dynamically optimizes queries during runtime (e.g., skew join handling, shuffle partition coalescing)[1] [2].
  - Up to 2x faster SQL performance due to Catalyst optimizer enhancements[1].
- **Cost Savings**: Reduced infrastructure costs via efficient resource utilization[2].
- **Modern Features**:
  - ANSI SQL compliance for standardized queries[1].
  - Structured Streaming improvements (event-time watermarking, late data handling)[1].
- **Security**: Updated libraries (e.g., log4j) mitigate vulnerabilities[2].

MongoDB Spark Connector Changes

- **New Namespace**:
  - Use `com.mongodb.spark.sql.connector.MongoTableProvider` instead of `com.mongodb.spark.DefaultSource`[3] [4].
- **Configuration Updates**:

```
# Old (Spark 2.x)
.config("spark.mongodb.input.uri", "mongodb://host/db.collection")

# New (Spark 3.x)
.config("spark.mongodb.read.connection.uri", "mongodb://host/db.collection") [^5]
```

- **Dependency Management**:
  - Scala 2.12/2.13 artifacts for Spark 3.1+/3.2+ compatibility[5].
  - Requires Spark 3.2+ for full functionality[4].

PySpark Migration Changes

- **Dependency Requirements**:
  - Pandas ≥0.23.2 and PyArrow ≥0.12.1[6].
- **Timestamp Handling**:
  - Respects session timezone by default (disable via `spark.sql.execution.pandas.respectSessionTimeZone=False`)[6].
- **Fallback Behavior**:
  - Enable `spark.sql.execution.arrow.fallback.enabled=True` to revert to non-optimized paths[6].

Legacy Configuration Settings

Spark 2 Behavior	Spark 3 Config to Enable Legacy Mode
Case-insensitive parsing	`spark.sql.legacy.caseInsensitiveParser=true`[7]
Old timestamp parsing (SimpleDateFormat)	`spark.sql.legacy.timeParserPolicy=legacy`[7]
Allow non-existent paths in `INSERT OVERWRITE`	`spark.sql.legacy.allowNonEmptyLocationInCTAS=true`[7]

Recommended Migration Steps

1. **Test Compatibility**:
   - Use `spark.sql.legacy.*` settings to identify breaking changes[7].
2. **Update Dependencies**:
   - MongoDB Connector v10.x, Scala/PyArrow versions[3] [5] [6].
3. **Benchmark**:
   - Validate performance gains from AQE and S3-optimized committer[2].
4. **Refactor Code**:
   - Migrate UDFs to Pandas UDFs/Vectorized UDFs for performance[1] [6].

This structure balances technical depth with readability, emphasizing actionable insights and critical changes. Use visuals like performance graphs, code snippets, and comparison tables to enhance engagement.

⁂

1. https://blog.devgenius.io/spark-3-0-vs-spark-2-x-e632bf121adc
2. https://slack.engineering/unlocking-efficiency-and-performance-navigating-the-spark-3-and-emr-6-upgrade-journey-at-slack/

3. https://www.mongodb.com/docs/spark-connector/current/
4. https://www.mongodb.com/blog/post/new-mongodb-spark-connector
5. https://www.mongodb.com/docs/spark-connector/v10.1/java/api/
6. https://downloads.apache.org/spark/docs/3.0.1/pyspark-migration-guide.html
7. https://community.cloudera.com/t5/Community-Articles/Spark-3-legacy-configurations-list-Spark-2-behaviour/ta-p/398673