Section II

A. Would you store data in databricks as delta or parquet format? Based on your choice, please explain why.

I would recommend using Delta format for storing data in Databricks, especially if your use case involves complex data workflows or requires robust data management features.

1. ACID Transactions: Ensures strong data consistency and integrity, especially important in concurrent read/write scenarios.
2. Schema Enforcement & Evolution: Maintains data quality and adapts to changing data structures.
3. Time Travel: Allows accessing and restoring previous versions of data, useful for audits and rollbacks.
4. Performance Optimizations: Includes features like file compaction and Z-ordering for faster query performance.
5. Unified Batch and Streaming: Seamlessly handles both real-time and batch data processing.

B. Please explain the process of how you would optimize PySpark or SQL code to effectively use databricks spark cluster.

Optimizing PySpark or SQL code to effectively use a Databricks Spark cluster involves several strategies aimed at improving performance and resource utilization. Here are key steps in the optimization process:

1. Data Understanding: Address data skewness and partition data efficiently for even distribution.
2. Data Storage Optimization:
    a. Use efficient formats like Parquet or Delta.
    b. Implement data caching with `.cache()` or `.persist()` for frequent data access.
3. Resource Allocation:
    a. Choose the right cluster size.
    b. Enable dynamic resource allocation for optimal use of resources.
4. Spark Configurations:
    a. Adjust `spark.sql.shuffle.partitions` for shuffle optimization.

    b. Configure `spark.executor.memory` and `spark.driver.memory` for better memory management.
5. Code Efficiency:
    a. Minimize shuffles in transformations.
    b. Utilize broadcast joins when joining a large DataFrame with a smaller one.
    c. Apply filters early in the data processing pipeline using `.filter()`.
6. SQL Optimization:
    a. Implement predicate pushdown.
    b. Write efficient SQL queries, selecting only necessary columns and rows.
7. Leverage Databricks Features:
    a. Take advantage of Delta Lake's Z-ordering and data skipping.
    b. Use the optimized Databricks runtime.
8. Monitoring and Tuning:
    a. Use Spark UI and Databricks dashboards for performance monitoring.
    b. Continuously refine and tune configurations based on observed performance.