Optimizations in Processing

Here are **10 key optimizations** you made to enhance processing efficiency:

1. **Delta Lake for Storage & Query Optimization**
   - Used Delta Lake for storing Go vulnerability incidents, ensuring ACID transactions and efficient time travel.
   - Delta caching improves read performance significantly over raw Parquet.

2. **Partitioning on Year, Month, and Day**
   - Partitioning by year, month, and day allows pruning of unnecessary data, reducing query scan time for time-based queries.

3. **Use of Databricks Autoloader**
   - Used **Databricks Autoloader** (`cloudFiles`) to efficiently ingest files from **ADLS**, reducing overhead compared to manual batch ingestion.

4. **Efficient Data Transformation with Spark SQL & PySpark**
   - Applied **distributed transformations** using **Spark SQL** instead of traditional row-wise operations, improving efficiency.
   - Used **broadcast joins** for small tables to optimize join performance.

5. **Delta Lake Merge for SCD Type 2 Implementation**
   - Implemented **MERGE INTO** for SCD Type 2 in `dim_packages`, preventing unnecessary overwrites and reducing data duplication.

6. **Compression & File Format Optimization**
   - Used **Parquet format with Snappy compression**, reducing storage size and improving read speeds compared to CSV or JSON.

7. **Z-Ordering for Faster Queries**
   - Used **Z-ORDER BY (modified_dt, published_dt)** to co-locate data for faster query execution when filtering on timestamps.

8. **Adaptive Query Execution (AQE) in Databricks**
    - Leveraged **AQE** to dynamically optimize joins and shuffle partitions at runtime, reducing unnecessary memory consumption.

9. **Auto-Scaling Compute for Cost Efficiency**
    - Used **Databricks Auto-Scaling clusters** with **spot instances**, optimizing costs while handling varying workloads.

10. **Incremental Processing with Change Data Capture (CDC)**

- Instead of full reprocessing, only **modified and new records** are processed using **timestamp-based CDC**, reducing unnecessary computations.