**Option B: Store Raw Data First and Transform It Afterwards (Batch Processing)**

**Reason for Choosing Option B:**

1. **Flexibility & Maintenance:**
   * If you have raw data, it adds flexibility to any transformation changes you might have. Also makes it easy to maintain & debug where you would only need to make changes to the batch transformation pipeline in case of any logic changes.
2. **Scalability:**
   * Since transformations are isolated, makes it easier to scale either process independently.
3. **Fault Tolerance:**
   * In case of errors or failures, don’t need to load raw data again. Can simply fix the error in transformation pipeline and rerun.
4. **Performance**:
   * Faster ingestion since no transformation logic is involved during API calls.

**Advantages and Disadvantages of Options:**

* **Advantages of Option B outlined above.**
* **Disadvantages would be:**
  + Additional storage for raw data (could be temporary, if we decide to delete raw data if transformed correctly)
  + As data is being ingested, transformation logic would be on hold so slight delay in data availability.
* **Advantages of Option A :**
  + Easy architecture for small scale data/systems
  + Immediate availability of transformed data once pipeline is run
* **Disadvantages of Option A:**
  + Pipeline can become complex
  + Performance would be slow for high ingestion
  + Can become hard to debug as ingestion and transformation happening at same time.

**Risk Mitigation Strategies:**

1. **Error Handling:**
   * Implement retry mechanisms and error logging during batch transformation.
2. **Versioning:**
   * Maintain versioned transformation logic to ensure reproducibility and traceability of results.
3. **Monitoring:**
   * Use monitoring tools to track batch jobs and identify bottlenecks.
4. **Validation:**
   * Validate source and target data. Like check for counts of id’s or rows loaded.