NCP-ADS Quick Revision Sheet

Domain: Data Manipulation and Software Literacy (20%)

Generated: 2026-02-17 | Use this as a last-mile exam revision sheet for high-yield concepts, decision rules, and pitfalls.

1. Core Decision Rules

- Start with **cudf.pandas** for fastest migration of existing pandas code, then optimize hotspots in direct cuDF.
- Use **cuDF** directly when you need stronger control, GPU-only pathways, or cuDF-native APIs.
- Use **Dask-cuDF** for multi-GPU or out-of-core workloads, deployed with Dask-CUDA.
- Prefer **dask.dataframe** API with backend set to cudf for portability; convert with to_backend only when needed.
- For distributed ETL, default to **Parquet** for projection and predicate pushdown.

2. Memory and Scale Playbook

Concern	Recommended Action	Why it helps
Worker allocations	Initialize RMM pool (for example 0.8-0.9)	Faster, more stable device allocations
Spill behavior	Enable cuDF spilling for ETL workloads	Reduces OOM risk under pressure
Partition sizing	Start around 1/32-1/16 GPU memory for shuffle-heavy jobs	Balances utilization and stability
Execution calls	Avoid compute on large collections; prefer persist+wait	Keeps data distributed across workers
Backend switching	Minimize repeated to_backend conversions	Avoids expensive CPU-GPU movement

3. I/O and Data Type Hotspots

- **Parquet strings:** ZSTD often beats Snappy for file-size reduction; encoding choice depends on cardinality and string-length profile.
- **JSON Lines:** Use lines=True and byte-range reads for very large sources; use multi-source reads for many small files.
- **cuDF dtypes:** know nullable semantics, nested list/struct support, and decimal handling for precision-sensitive workloads.
- **GPUDirect Storage:** when supported, direct storage-to-GPU path can improve ingest throughput and reduce CPU bounce-buffer overhead.

4. Exam Pitfalls to Avoid

- Assuming cudf.pandas means all steps run on GPU; check fallback behavior.
- Calling compute too early and exhausting single-device memory.
- Sorting globally without business need, causing unnecessary all-to-all shuffle cost.
- Ignoring data skew when diagnosing join and groupby performance.
- Running with unpinned environments and then failing to reproduce performance.

5. 12-Minute Pre-Exam Checklist

- Can you explain when to use cudf.pandas vs direct cuDF?
- Can you set up a LocalCUDACluster with RMM and spilling defaults?
- Do you know safe partition-size starting ranges for shuffle-heavy pipelines?
- Can you explain why persist differs from compute in distributed execution?
- Can you describe one JSON and one Parquet optimization for ingest?
- Can you state two common causes of avoidable OOM in Dask-cuDF jobs?

Primary References (official and high-signal)

- https://docs.rapids.ai/api/cudf/stable/cudf_pandas/faq/
- https://docs.rapids.ai/api/dask-cudf/stable/best_practices/
- https://docs.rapids.ai/api/cudf/stable/user_guide/io/read-json/
- https://developer.nvidia.com/blog/encoding-and-compression-guide-for-parquet-string-data-using-rapids/
- https://developer.nvidia.com/blog/boosting-data-ingest-throughput-with-gpudirect-storage-and-rapids-cudf/
- https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html
- https://docs.docker.com/get-started/docker-overview/

This sheet is aligned to NCP-ADS Data Manipulation and Software Literacy exam preparation. Use with hands-on drills for retention.