# Google Cloud PDE Cheat Sheet

One-page cram sheet for Professional Data Engineer design, pipelines, governance, quality, and analytics patterns.

| Best for | Focus | Use with |
| --- | --- | --- |
| Last review before drills or exam day | High-yield product mapping and data-engineering decisions | Use with the quick summary for rapid refresh |

## 1. Design data processing systems

| | |
| --- | --- |
| **Ingestion choice** | Match batch, streaming, CDC, and event-driven patterns to latency and reliability needs. |
| **Storage selection** | BigQuery, Cloud Storage, Bigtable, Spanner, and Cloud SQL solve different access and scale problems. |
| **Transformation layer** | Pick Dataflow, Dataproc, or SQL-based transformation based on scale, control, and ops overhead. |
| **Schema strategy** | Partitioning, clustering, and schema evolution shape both performance and long-term cost. |

## 2. Build and operationalize pipelines

| | |
| --- | --- |
| **Batch vs streaming** | Use batch for predictable periodic jobs; use streaming when freshness and event-time logic matter. |
| **Reliability** | Idempotency, retries, dead-letter handling, and checkpointing reduce pipeline fragility. |
| **Orchestration** | Use Composer, Workflows, or event triggers when steps need coordination and observability. |
| **Testing** | Validate data quality, business rules, and edge cases before production scale. |

## 3. Ensure solution quality

| | |
| --- | --- |
| **Observability** | Logs, metrics, lineage, and alerting are part of the data system, not extras. |
| **Data quality** | Freshness, completeness, accuracy, duplicates, and drift should all be measurable. |
| **Cost-performance balance** | The best design is not always the cheapest or the fastest in isolation. |
| **SLA thinking** | Choose architecture based on availability, latency, and recovery expectations. |

## 4. Govern and secure data

| | |
| --- | --- |
| **IAM and least privilege** | Data access should be scoped by role, dataset, table, service account, and purpose. |
| **Encryption and keys** | Default encryption helps, but key control, access design, and auditability still matter. |

| | |
|---|---|
| **Sensitive data controls** | Masking, tokenization, DLP, and policy controls matter for regulated workloads. |
| **Lineage and audit** | Good governance needs traceability for origin, change, and access. |

## 5. Support machine learning and analytics

| | |
|---|---|
| **Feature-ready data** | Good data engineering makes downstream analytics and ML practical, repeatable, and trusted. |
| **BigQuery-first patterns** | Many Google Cloud solutions simplify when analytics workloads can stay in BigQuery. |
| **Serving patterns** | Operational systems, dashboards, and ML consumers often need different storage or latency profiles. |
| **Practical mindset** | The best answer usually balances scalability, governance, and operational simplicity. |