**SEO Data Pipeline: Architecture & Scaling Document**

**1. Quick Data Discovery Tests**

Before building the pipeline, lightweight checks on each CSV (gsc\_data.csv, analytics\_data.csv, rank\_data.csv) should be performed to validate assumptions:

* **Schema validation**
  + Verify expected columns are present
  + Check column types
* **Row-level validation**
  + Ensure Primary Key columns are not NULL in source.
  + Check for duplicates at the expected primary key grain:
* **Range and sanity checks**
  + Perform sanity checks on each column depending its data type and metric type.
* **Sample profiling**
  + Row counts per month/day
  + Distinct number of Key Columns values
  + Identification of joining keys and relation between datasets
  + Distribution of sessions, clicks, CTR to spot outliers.

**2. Data Challenges**

During data discovery, I found multiple following issues:

* **Duplicate rows**: Need aggregation at the right grain before joins.
* **Pre-computed averages**: CTR and conversion\_rate columns must be recomputed from raw counts when rolling up.
* **URL normalization**: Pages may appear with/without trailing slashes, UTM tags, case differences.

**3. Scaling Considerations**

**a) 10M rows/day per source**

* **Storage**: CSVs in GCS with respective data coming into their own data source folders
* **Ingestion:** PySpark based framework can be used to ingest data if there are any limitations coming up with BQ native ingestion jobs and data can be stored partition on the incoming dates
* **Staging &Transformations**: Use BigQuery (serverless, scalable) for SQL-based staging, joins, deduplication. 10M rows/day is trivial for BQ.
* **ETL engine**: Orchestrate with Cloud Composer (Airflow); transformations triggered after file arrival.

**b) 50 clients (multi-tenancy)**

* **Shared dataset with client\_id column**
  + Tables partitioned by client\_id, data\_date.

**4. GCP Implementation**

**Services chosen:**

* **Cloud Storage (GCS)** → landing zone for monthly/daily CSVs. Auto-ingestion triggers downstream jobs.
* **DataProc Spark Jobs**→ Spark based ingestion framework for scalability and large datasets
* **BigQuery** → main warehouse; staging, history, fact and dimension tables. Handles joins, deduplication, analytics queries at scale.
* **Cloud Composer (Airflow)** → orchestrates the daily ETL DAG (load → validate → transform → build facts).
* **Looker Studio / Looker** → BI layer for reporting keyword and page performance.

**Why these services:**

* All are **serverless or managed** → minimal infra overhead.
* **BigQuery** can handle 10M+ rows/day effortlessly.
* **Composer** provides retry logic, alerting, and dependency management.
* **dbt** enforces software engineering best practices for SQL models.

**5. Monitoring & Alerting**

* **Detect missing data**
  + After file arrival, validate row counts vs. historical averages.
  + If row\_count = 0 or data\_date missing → trigger alert (Slack/Email).
* **DQ Framework:**
  + Customized data quality checks on each layer and send email in case of failures
* **Detect failed runs**
  + Airflow DAG task failure triggers Cloud Monitoring alert.
  + Log all steps with run\_id and data\_date for traceability.
* **Track data freshness**
  + Metadata table: (source, last\_successful\_date, row\_count, run\_date).
  + Dashboards in Looker showing freshness status (e.g., “Last GSC load = 2 days ago”).

**6. BASIC ARCHITECTURE DIAGRAM**

