**Appendix -A**

**Purpose:** This Appendix-A documents the implementation of the Medallion pipeline with concise screenshots and actions. It follows the lifecycle Pre-merge → Bronze (ADF) → Silver (Databricks) → Gold (Synapse) → Consumption (Power BI).

A screenshot of a computer

AI-generated content may be incorrect.

Figure A-1 — Resource Group

What it shows is One resource group hosting all Medallion pipeline services.

* amazon2023-data-factory — Ingestion & scheduling.
* amazondatastorageaccount — ADLS Gen2 lake (bronze/silver/gold).
* amazon2023-workspace — Databricks for cleansing/curation.
* amazon23-synapse-workspace — Serverless SQL for querying/reporting.
* amazon23synapsestorage — Synapse workspace storage.

Why it matters: Confirms end to end readiness in a single resource group, simplifying RBAC and cost tagging. *(Deleting the RG removes all contained resources.)*

A screenshot of a computer

AI-generated content may be incorrect.

Figure A-2 — Storage Account amazondatastorageaccount (ADLS Gen2)

Storage account settings for the data lake.

* Region & replication: Canada Central; copies to Canada East (RA-GRS).
* Data lake: Hiearchical namespace is enabled.
* Tier & access: Hot tier; anonymous access disabled.
* Protection: Soft delete enabled (7 days); versioning disabled.
* Security: Secure transfer required; TLS ≥ 1.2; key access enabled.

Why it matters: Confirms the account is lake-ready and protected for bronze/silver/gold used by Databricks, Synapse, and Power BI

A screenshot of a computer

AI-generated content may be incorrect.

Figure A-3 — Containers in amazondatastorageaccount

What it shows: Two containers and their roles in the data lake.

* amazon2023data — Primary container with /bronze (raw), /silver (cleaned, year-partitioned Parquet), /gold (curated Parquet).
* $logs — Azure-managed logging/diagnostics.
* Access: Private (no anonymous access). Status: Available.
* Use: Ingest via Data Factory → transform in Databricks → query in Synapse.
* Governance: Apply RBAC to ADF/Databricks/Synapse as needed.  
  Why it matters: Confirms a secure, ready container that anchors the Medallion layout.

A screenshot of a computer

AI-generated content may be incorrect.

Figure A-4 — Container amazon2023data

What it shows : Root of the lake container with the standard Medallion structure.

* bronze — Raw landing zone (immutable source files).
* silver — Cleaned Parquet, partitioned by review\_year for pruning.
* gold — Curated Parquet ready for Synapse/Power BI (e.g., year\_summary, product\_year).
* Status & access — Container is Private and Available (secure, ready for use).
* Path stability — Consistent folder names used by Databricks jobs and Synapse views.

Why it matters: Confirms the expected layout that ingestion, transformation, and reporting rely on enables reliable automation and query performance.

A screenshot of a computer

AI-generated content may be incorrect.

Figure A-5 — Bronze landing

What it shows (brief): The raw dataset staged in amazon2023data/bronze/All\_Beauty.

* Size/type/tier: 1.6 GiB, Block blob, Hot tier.
* Source: landed via Data Factory Copy from GitHub/HTTP.
* Role: immutable raw copy no edits; serves as the single source of truth.
* Access: private container; ADF writes, Databricks reads to create Silver.
* Quick checks: item present, size reasonable vs. source, no failed transfers.

Why it matters: Confirms ingestion succeeded and provides the trusted input that Databricks uses to produce the Silver layer.

A computer screen shot of a computer

AI-generated content may be incorrect.

Figure A-6 — Silver layer (year-partitioned Parquet)

What it shows (brief): The amazon2023data/silver/ folder with correctly written Parquet outputs.

* Partitions: review\_year=2015 … 2023
* Format: Parquet (columnar, compressed)
* Job marker: \_SUCCESS present
* Purpose: cleaned, standardized data for analytics

Why it matters: Confirms Silver is correctly produced year partitions enable pruning and faster Synapse/Power BI queries, and the data is ready to build Gold datasets.

A computer screen shot of a computer

AI-generated content may be incorrect.

Figure A-7 — Gold layer (curated Parquet datasets

What it shows (brief): The amazon2023data/gold/ folder with curated, analysis-ready outputs.

* Datasets: brand\_year, category\_year, month\_trend, price\_rating, product\_year, top\_reviews, year\_summary.
* Format: Parquet with stable, documented columns.
* Purpose: Aggregated/denormalized tables for BI; filter/join by review\_year, brand, category, parent\_asin, month.
* Consumption: Read by Synapse Serverless (OPENROWSET/views) and Power BI.

Why it matters: Confirms curation is complete and ready for governed SQL access and fast reporting.

A screenshot of a computer

AI-generated content may be incorrect.

Figure A-8 — Data Factory: Copy activity (GitHub → ADLS Bronze)

What it shows (brief): ADF pipeline that copies the raw *All\_Beauty* file from GitHub/HTTP into ADLS Bronze.

* Activity: Copy data (datafromgithub).
* Source: dataset githubDataCsv (HTTP GET to GitHub raw URL).
* Sink: amazon2023data/bronze/All\_Beauty/ in amazondatastorageaccount.
* Run controls: Debug, Publish, optional Trigger.

Why it matters: Confirms ingestion is configured end to end and ready to land raw data into Bronze.

A computer screen shot of a computer

AI-generated content may be incorrect.

Figure A-9 — Data Factory Dataset

What it shows (brief): Source dataset configuration for pulling the raw CSV from GitHub.

* Linked service: githublink (HTTP).
* Base/relative URL: points to All\_Beauty.csv (raw).
* Method & format: GET, DelimitedText.
* Settings: Comma delimiter, UTF-8 encoding, No compression.

Why it matters: Ensures the Copy data activity has a valid, repeatable source definition to land files into bronze.

A screenshot of a computer

AI-generated content may be incorrect.

Figure A-10 — Databricks Cluster Configuration

User manual — set this up (steps):

* Create compute: Databricks → Compute → Create → “All-purpose cluster”.
* Runtime: pick 14.3 LTS (or current LTS). Turn Photon ON.
* Access mode: Single user → select your account (ensures isolation).
* Size & policy: Node Standard\_D4ds\_v5 (or similar 4 vCPU/16 GB). Set min/max = 1.
* Cost controls: Enable Auto-terminate = 20 min; keep spot/preemptible OFF for stability.
* Permissions: Grant only required users Can Attach To (readers = Can View).
* Secrets & storage: Ensure your SP/Managed Identity has Storage roles; reference secrets via Key Vault/Databricks Secrets (no hard-coded creds).
* Attach & run: Open the ETL notebook → Attach to this cluster → Run all.

A screenshot of a computer

AI-generated content may be incorrect.

Figure A-11 — Databricks Notebook: ADLS auth & ETL run

What it shows (brief): A Databricks notebook initializing Spark, configuring ADLS Gen2 access via Service Principal (OAuth Client Credentials), then running ETL cells.

* Auth cell: app (client) ID, secret, and tenant configured for ADLS access.
* Source: reads bronze/All\_Beauty.
* Outputs: writes silver (year-partitioned Parquet)
* Execution: cells run top-to-bottom on the selected cluster.

Why it matters: Confirms a secure, reproducible ETL path from bronze → silver

A screenshot of a computer

AI-generated content may be incorrect.

Figure A-12 — Synapse Studio Home (“amazon23-synapse-workspace”)

* Open Develop → SQL script, paste/open All\_Beauty\_Synapse.sql .
* Set Connect to = Built-in (Serverless); database will be AmazonAwDatabase (the script creates/uses it).
* Run to create the Managed Identity credential, external data source/file format, and the vw\_silver view over /silver.

**Appendix – B**

**Purpose**

This appendix documents a practical, reproducible pipeline for the Electronics category using the Medallion pattern: Bronze (ingest) → Silver (clean) → Gold (analytics) → Synapse views → Power BI. The pipeline runs on Databricks with data in Azure Blob Storage; Synapse Serverless provides read-only views for BI.

**B1. Files and where they run**

* *amazon\_data\_pipeline.py* — Ingestion to Bronze  
  *What it does:* downloads Electronics reviews and product metadata and uploads the raw files to Bronze.  
  *Run where:* local machine or a small VM.  
  *Result:* immutable raw data in the lake (source of truth).
* *create\_ingestion\_job.sh* — Register the Databricks ingest job (optional)  
  *What it does:* creates a Databricks job via REST API so ingestion is repeatable.  
  *Run where:* any terminal with Databricks CLI and curl.  
  *Result:* a job ID you can schedule or trigger.
* *01\_bronze\_to\_silver.py* — Bronze → Silver (Databricks)  
  *What it does:* standardizes schema, validates, de-duplicates, enriches with product metadata; writes optimized Delta/Parquet to Silver (often partitioned by review\_year).  
  *Run where:* Databricks job/cluster.
* *sentiment\_analysis.py* — Silver → Gold (Databricks)  
  *What it does:* runs an ensemble (VADER, TextBlob, rule-based) to add sentiment, confidence, and quality metrics; produces curated Gold tables for BI/ML (typically partitioned by review\_year).  
  *Run where:* Databricks job/cluster (after Silver).
* *run\_jobs.sh* — Orchestration  
  *What it does:* triggers Silver then Gold jobs in order via REST API.  
  *Run where:* terminal/CI (cron, GitHub Actions, Azure DevOps).

**B2. Synapse views for BI (over Gold)**

Create lightweight Serverless SQL views that hide storage paths and present a clean, consistent model to Power BI:

1. v\_executive\_dashboard — KPIs and summary (e.g., total\_reviews, positive\_percentage, NPS proxy, verified\_purchase\_rate).
2. v\_product\_performance\_dashboard — product-level performance and rankings (brand/category, price band, avg rating, volume).
3. v\_sentiment\_analysis\_dashboard — sentiment outcomes and alignment (final\_sentiment, intensity, per-model averages, dispersion).
4. v\_time\_trends\_dashboard — seasonality and trends (year/month, season; volume, rating, positivity).
5. v\_sentiment\_intensity\_dashboard — strength buckets (Strong/Moderate/Mild) by brand/category/time.
6. v\_historical\_recent\_dashboard — compares recent performance to historical baselines.
7. v\_covid\_impact\_dashboard — impact during pandemic periods (volume, rating, sentiment).
8. v\_gold\_reviews\_complete — foundation view over the final Gold table; other views depend on it.

**B3. Permissions and security (checklist)**

* Storage account roles
  + Databricks MI/SP → Storage Blob Data Contributor (read/write Silver & Gold).
  + Synapse Managed Identity → Storage Blob Data Reader (read Gold for views).
* Secrets
  + Use Key Vault or Databricks Secrets; avoid hard-coding credentials.
* Workspaces
  + Limit “Can Attach To” on Databricks clusters; restrict Synapse access to the minimum set of users.

**B4. Quality and validation (five quick checks)**

1. Bronze — File counts and sizes match the source; no partial uploads.
2. Silver — Partition folders exist (e.g., review\_year=YYYY); row counts and null rates look reasonable.
3. Gold — Sentiment fields populated; confidence and quality scores within expected ranges.
4. Synapse — Each view returns rows; data types match the intended model.
5. Power BI — KPIs look plausible; filters and drill-downs behave as expected.

**B5. Operational notes**

* Save a small hand-labeled validation set to monitor sentiment accuracy over time.
* Tag releases of transformation logic (e.g., electronics-v1.2) to make dashboards reproducible.
* Add simple run notifications and retries in run\_jobs.sh for smoother operations.
* Keep Bronze indefinitely if storage allows; it protects reproducibility when logic changes.

Outcome: With these files and steps, Electronics data flows from raw ingestion to sentiment-enhanced, analytics-ready Gold, exposed via Synapse views and ready for dashboards in Power BI.