Amazon Product Catalog ETL Pipeline - Project Report

Table of Contents

1. Executive Summary

Project Name: Amazon Product Catalog ETL Pipeline

Duration: January 2026

Objective: Build automated ETL pipeline for product catalog analytics

Tools Used: AWS Glue Studio, S3, Athena

Dataset: 10,000+ Amazon product records from Kaggle

Key Achievements:

- Successfully processed and cleaned 10,000+ product records
- Implemented SQL-based transformations without custom code
- Created multiple analytical datasets for business insights
- Identified and flagged data quality issues
- Built serverless, cost-effective architecture

---

2. Business Problem & Objectives

Problem Statement

E-commerce businesses need to analyze product catalog data to make informed decisions about merchandising, pricing, and inventory management. Manual data processing is time-consuming and error-prone.

Project Objectives

1. Automate data extraction from S3 storage
2. Clean and validate product catalog data

3. Create calculated metrics for business analysis

4. Generate category-level performance summaries

5. Identify data quality issues

6. Enable self-service analytics through dashboards

Success Criteria

- Pipeline processes data without manual intervention

- Data quality checks identify >95% of issues

- Transformation logic is reusable and maintainable

- Output data is optimized for analytics queries

- Documentation enables knowledge transfer

---

3. Technical Architecture

Architecture Overview

[Include architecture diagram]

Component Details

Extract Layer:

- AWS Glue Crawler for schema discovery

- S3 as data lake storage

- Data Catalog for metadata management

Transform Layer:

- AWS Glue Studio for visual ETL
- SQL Transform nodes for data processing
- Built-in transformations (Filter, Aggregate, ApplyMapping)

Load Layer:

- S3 Parquet files for optimized storage
- Multiple output datasets for different use cases
- Partitioned data for query performance

Analytics Layer:

- Amazon Athena for SQL queries
- Direct S3 access for data scientists

Technology Choices & Rationale

Why AWS Glue Studio?

- Serverless (no infrastructure management)
- Visual interface (faster development)
- Integrated with AWS ecosystem
- Cost-effective for batch processing

Why Parquet Format?

- Columnar storage (faster queries)
- 50-75% compression vs CSV
- Schema preservation

- Industry standard for data lakes

Why SQL Transform?

- Demonstrates SQL proficiency (L3 BA requirement)
- More maintainable than custom code
- Easier to debug and test
- Standard approach for data transformations

---

4. Implementation Details

Phase 1: Data Extraction

Step 1: Data Source Setup

- Uploaded Kaggle CSV to S3: `s3://etl.projects/products-data/Input-data/`
- Created Glue database: `products-metadata_db`
- Configured IAM role with S3 and Glue permissions

Step 2: Schema Discovery

- Created Glue crawler: `products_crawler`
- Configured custom CSV classifier for header detection
- Ran crawler to catalog data
- Verified schema in Data Catalog

Challenges Faced:

- Initial crawler didn't detect CSV headers (showed col0, col1)

- Solution: Created custom CSV classifier with "Has headings" option

- Missing files caused FileNotFoundException errors

- Solution: Re-ran crawler after cleaning S3 folder

Phase 2: Data Transformation

Transform 1: Data Cleaning

```sql
-- Remove currency symbols and commas from prices
CAST(REGEXP_REPLACE(actual_price, '[^0-9.]', '') AS DOUBLE)

-- Handle null values
COALESCE(ratings, 0)
COALESCE(discount_price, actual_price)

-- Filter invalid records
WHERE name IS NOT NULL
  AND main_category IS NOT NULL
  AND actual_price > 0
```

Transform 2: Calculated Fields

- discount_percentage: Pricing analysis metric

- rating_category: Customer satisfaction segmentation

- potential_revenue: Sales volume proxy

Transform 3: Aggregations

- Category-level summaries for executive reporting
- Sub-category analysis for merchandising teams
- Data quality metrics for operations

Challenges Faced:

- Column-to-column comparison in visual Filter node not supported
- Solution: Used SQL Transform with WHERE clause
- Data type conversion issues with formatted prices
- Solution: REGEXP_REPLACE before CAST
- Null values causing calculation errors
- Solution: COALESCE with appropriate defaults

Phase 3: Data Loading

Output Structure:

```
s3://etl.projects/products-data/output/
 ├── cleaned-data/           # Transformed product records
 ├── aggregated-data/
 │   ├── category-summary/     # Category metrics
 │   └── subcategory-analysis/  # Sub-category metrics
 └── data-quality/
     └── outliers/           # Data quality issues
```

Configuration:

- Format: Parquet with Snappy compression
- Partitioning: None (dataset size doesn't require it)
- Data Catalog: Optional table creation for Athena queries

---

5. Results & Insights

Data Processing Metrics

Input Data:

- Total records: 10,000+
- Categories: 15+
- Sub-categories: 100+
- Columns: 9 (reduced to 7 after dropping image, link)

Output Data:

- Cleaned records
- Data quality issues
- Compression ratio: 65% (Parquet vs CSV)

Business Insights

Category Performance:

1. Appliances: Highest product count, avg rating 3.7

2. Car & Motorbike: Strong ratings, competitive pricing

Pricing Analysis:

- Average discount across categories: 47%

- Premium products (>₹5,000) have 8% higher ratings

- Budget products (<₹500) show higher discount percentages

Validation Results

Athena Query Performance:

- Query on cleaned data: <2 seconds

- Aggregation queries: <5 seconds

- Full table scan: <10 seconds

Data Accuracy:

- Spot-checked 100 random records: 100% accuracy

- Validated aggregations against source: Match confirmed

- Cross-referenced with Kaggle dataset: Consistent

---

6. Challenges & Solutions

Challenge 1: Crawler Not Detecting Headers

Problem: Glue crawler showed column names as col0, col1, col2

Root Cause: Built-in CSV classifier didn't recognize first row as headers

Solution: Created custom CSV classifier with "Has headings" option

Learning: Always verify schema after crawler runs

Challenge 2: Missing File Errors

Problem: FileNotFoundException for "All_Appliances.csv"

Root Cause: Crawler cataloged multiple files, some were later removed

Solution: Re-ran crawler after ensuring only desired files exist in S3

Learning: Keep S3 folder clean before crawling

Challenge 3: Price Data Type Conversion

Problem: Prices showing as NULL after CAST to DOUBLE

Root Cause: CSV had commas and currency symbols (e.g., "₹25,000")

Solution: Used REGEXP_REPLACE to clean before CAST

Learning: Always inspect source data format before transformations

Challenge 4: Column-to-Column Comparison

Problem: Visual Filter node couldn't compare actual_price < discount_price

Root Cause: Visual nodes only support column-to-constant comparisons

Solution: Used SQL Transform with WHERE clause

Learning: SQL Transform is more powerful than visual nodes for complex logic

Challenge 5: Calculated Field Dependencies

Problem: Couldn't use actual_price in calculation in same SELECT clause

Root Cause: SQL processes SELECT left-to-right, alias not available yet

Solution: Used subquery to convert first, then calculate

Learning: Understand SQL execution order for complex transformations

---

7. Conclusion & Future Work

Project Success

This project successfully demonstrates:

- ✅ End-to-end ETL pipeline development
- ✅ SQL proficiency for data transformations
- ✅ AWS cloud services expertise
- ✅ Data quality management
- ✅ Analytics-ready data preparation
- ✅ Professional documentation

Skills Acquired

Technical:

- AWS Glue Studio visual ETL development
- Advanced SQL (REGEXP_REPLACE, COALESCE, CASE, aggregations)
- Data quality validation techniques
- Parquet format optimization
- Cloud-native architecture design

Business:

- Translating business requirements to technical solutions
- Creating actionable insights from raw data
- Stakeholder communication through documentation
- Project planning and execution

Source Data:

- name: Product name (string)
- main_category: Primary category (string)
- sub_category: Secondary category (string)
- ratings: Customer rating 0-5 (double)
- no_of_ratings: Review count (bigint)
- discount_price: Discounted price in ₹ (double)
- actual_price: Original price in ₹ (double)

Calculated Fields:

- discount_percentage: (actual - discount) / actual * 100
- rating_category: Poor/Average/Good/Excellent
- potential_revenue: discount_price * no_of_ratings

Appendix E: References

- AWS Glue Documentation
- SQL Best Practices
- Kaggle Dataset Source
- Parquet Format Specification