Azure Lab 3: Data Preprocessing on Azure

Student Name: Shaima Mohammed

Student ID: 60104699

Course: DSAI3202 Parallel & Distributed Comp

Instructor: Dr. Oussama Djedidi

Institution: University of Doha for Science & Technology

Date: 11November 2025

1️⃣ Objective

The purpose of this lab was to design and implement a **data preprocessing pipeline** using **Azure Databricks** connected to **Azure Data Lake Storage Gen2 (ADLS Gen2)**.
The lab focused on cleaning, transforming, and organizing the **Goodreads dataset** across three data layers – **Silver**, **Processed**, and **Gold** – to prepare it for analytics and machine-learning use.

The tasks included:

- Connecting Databricks to Azure storage securely
- Loading and validating Silver layer data
- Cleaning and filtering the reviews dataset
- Saving a consistent Processed layer
- Building a Curated (Gold) table through joins and enrichment
- Creating new features and validation queries
- Documenting the full ETL process

2️⃣ Tools and Environment

- **Azure Databricks Workspace** (Notebook-based PySpark)
- **Azure Data Lake Storage Gen2 (ADLS Gen2)** – Storage account: `goodreads60104699`
- **PySpark** for data processing
- **Delta Lake** for final storage format
- **Databricks SQL** for validation queries

3️⃣ Data Sources

All source files were already available in the Silver layer:

- `/silver/books/`
- `/silver/authors/`
- `/silver/reviews/`

Each dataset contained structured Parquet files extracted from the Goodreads API.

4️⃣ Implementation Steps

Step 1 – Connect to ADLS Gen2

I started by setting up secure access to my storage account `goodreads60104699` using the Shared Key method:

```
spark.conf.set(
    "fs.azure.account.key.goodreads60104699.dfs.core.windows.net",
    "********"
)
```

To verify the connection, I listed the folders with:

```
display(dbutils.fs.ls("abfss://lakehouse@goodreads60104699.dfs.core.windows.net/"))
```

This confirmed successful connectivity between Databricks and my Data Lake.

Step 2 – Load Silver Layer Data

Using PySpark's `read.parquet()`, I loaded the three datasets from the Silver layer:

```
books   = spark.read.parquet(".../silver/books/")
authors = spark.read.parquet(".../silver/authors/")
reviews = spark.read.parquet(".../silver/reviews/")
```

I checked schemas and previewed sample rows to ensure the data was read correctly.

All datasets loaded successfully.

Step 3 – Clean and Prepare the Reviews Data

This was the main cleaning section of the lab (`load_silver_data_60104699.ipynb`).

Operations performed:

1. Removed rows with missing `review_id`, `book_id`, or `user_id`.
2. Casted the `rating` column to integer and kept only values from 1 to 5.
3. Trimmed white spaces in `review_text` and kept reviews with ≥ 10 characters.
4. Removed duplicate `review_id` records.
5. Standardized and validated `date_added` formats to timestamp.
6. Filtered reviews to keep only records dated after **2014-01-01**.

This step completed the requirements for **"V.2 – Clean the Data"** in the lab instructions.

Step 4 – Save to Processed Layer

After cleaning, I saved the final dataset to:

`/processed/reviews_clean_final/`

using the overwrite mode:

```
reviews_clean.write.mode("overwrite").parquet(output_path)
```

Verified with `dbutils.fs.ls()` and record count.

Step 5 – Build Curated (Gold) Table

In the second notebook (`Goodreads_silver_to_gold.ipynb`), I loaded the cleaned reviews and joined them with the books and authors datasets.

Bridge Table Creation

```
authors_bridge = (
    books
    .select(col("book_id"), explode(split(col("authors"),
","")).alias("name"))
    .withColumn("name", trim(regexp_replace(col("name"), r"[\[\]']", "")))
    .withColumn("author_id", monotonically_increasing_id().cast("string"))
)
```

Join and select final columns

```
curated_reviews = (
    reviews_clean
    .join(books_authors, on="book_id", how="inner")
    .select("review_id","book_id","title","author_id","name",
            "user_id","rating","review_text","date_added","average_rating")
)
```

This step satisfied **"V.3 – Build Curated Table (Gold Layer)"**.

The curated table was saved in Delta format under:

`/gold/curated_reviews/`

Step 6 – Feature Engineering

To add analytical value, I created new features using Spark aggregations:

- `review_length` – word count of each review.
- `avg_rating_book` – average rating per book.
- `review_count_book` – number of reviews per book.
- `avg_rating_author` – average rating per author.

Example:

```
avg_rating_book =
curated_reviews.groupBy("book_id").agg(avg("rating").alias("avg_rating_book")
)
```

Covers the **"V.3 – Aggregations and Enrichment"** requirements.

Step 7 – Save Final Gold Layer and Register as Table

The final dataset `curated_final` was saved and registered as a Delta table in Databricks:

```
curated_final.write.format("delta").mode("overwrite").save(".../gold/curated_
reviews/")
curated_final.write.format("delta").mode("overwrite").saveAsTable("curated_re
views")
```

The table was verified with `SHOW TABLES` and `SELECT COUNT(*)`.

Step 8 – SQL Validation

I executed several SQL queries to validate the Gold table:

1. Count total records and distinct IDs.
2. Compute average, min, and max ratings.
3. Find top 5 authors by review count.
4. Sample records with features for quality check.

Results confirmed clean and consistent data with no missing or corrupted records.

5️⃣ Results and Verification

Layer	Path	Format	Purpose
Silver	`/silver/books`, `authors`, `reviews`	Parquet	Source data
Processed	`/processed/reviews_clean_final/`	Parquet	Cleaned reviews
Gold	`/gold/curated_reviews/`	Delta	Curated + enriched dataset
SQL Table	`curated_reviews`	Delta Table	Final queryable table

6️⃣ Key Learnings

- Established secure connections between Databricks and ADLS Gen2.
- Performed complete data cleaning and type casting with PySpark.
- Designed a multi-layer data pipeline (Silver → Processed → Gold).
- Created aggregations and new features to add value to the dataset.
- Learned to validate data quality using Spark SQL queries.

7️⃣ Conclusion

The Goodreads dataset was successfully cleaned, transformed, and enriched following the lab instructions.
The final Gold layer is clean, consistent, and ready for further analysis or machine-learning tasks.

Each step was tested and validated inside Databricks, and the outputs were saved to Azure Data Lake Storage under my account `goodreads60104699`.

Appendix – Notebooks Used

- `load_silver_data_60104699.ipynb` – Data cleaning and Processed layer generation
- `Goodreads_silver_to_gold.ipynb` – Building Gold layer, feature engineering, SQL validation