Documentation Summary: E-commerce Sales ETL

1. Input Data Sources

The ETL pipeline processes three source datasets, which must be downloaded from the provided Google Drive link and placed in a location accessible by the Databricks environment (e.g., a DBFS path or Unity Catalog Volume).

Dataset	Data Role	File Format	Key Columns
Customers	Customer Information	Excel (.xlsx)	Customer ID, Customer Name, Email, Country
Products	Product Catalog	CSV (.csv)	Product ID, Product Category, Product Sub Category
Orders	Sales Transactions	JSON (.json)	Order ID, Customer ID, Product ID, Profit, Order Date

2. ETL Pipeline Overview (etl.py)

The pipeline follows a modular **Extract, Clean, Transform, Aggregate, Load (ECTAL)** flow using **PySpark** in a Databricks environment.

Phase	Action	Key Functions
Extract	Read raw files (Excel, CSV, JSON).	read_customers, read_products, read_orders
Clean	Data validation, cleansing, and issue reporting.	clean_customers, clean_products, clean_orders

Transform	Build Dimension tables and enrich the Orders Fact table.	build_customer_dim, build_product_dim, build_orders_enriched
Aggregate	Calculate final profit metrics.	aggregate_profit_by
Load	Write all outputs (7 tables) to Parquet files.	write_parquet

3. Core Transformation & Data Quality

The pipeline ensures high data quality by applying specific validation rules and separating invalid records into dedicated **issue reports**.

Dataset	Core Cleansing & Imputation	Key Validations & Issues Tracked
Customers	Drop null/duplicate IDs. **Impute missing names** from the email prefix.	Checks for valid formats: Email, Phone (4 patterns), Postal Code (ZIP/ZIP+4), Region (North/South/East/West/Central).
Products	Drop null/duplicate IDs. Rename price column to UnitPrice.	Filters out rows where UnitPrice < 0 (negative_unit_price).
Orders	Drop null/duplicate IDs. Convert dates.	**Referential Integrity**: Checks if Customer ID and Product ID exist in the *cleaned* dimension tables (invalid_customer_id, invalid_product_id).

4. Output Structure and Analytics Requirements

The ETL creates a star schema optimized for reporting.

Output Table Role	Purpose	Key Content
Dimensions	customers, products	Cleaned keys and attributes (e.g., CustomerName, Category).
Fact	orders_enriched	Fact table with joined attributes, **ProfitRounded to 2 decimal places**, and Year extraction.
Aggregate	aggregates	**Total Profit** grouped by Year, Category, SubCategory, and CustomerName.
Issues	*data_issues	Separate tables containing records that failed cleaning validations.

The final aggregates table is used to perform four required SQL queries: Profit by Year, Profit by Year + Category, Profit by Customer, and Profit by Customer + Year.

4. Testing Approach (test_etl_pipeline.py)

Testing follows a **Test-Driven Development (TDD)** approach using **Pytest** and local PySpark fixtures.

Test Category	Focus and Coverage
Unit Tests	Tests *each function in isolation* to confirm logic correctness (e.g., duplicate removal, imputation, RegEx validation).

Data Quality Tests	Strict verification that invalid data (e.g., negative price, invalid phone, non-existent foreign keys) is correctly routed to the issue reports.
Schema & Join Tests	Confirm dimension tables are built correctly and the `orders_enriched` fact table contains all required joined columns (`CustomerName`, `Category`, etc.) and calculated fields (`ProfitRounded`, `Year`).
Value & Aggregation Tests	Strict assertion on final numerical values, ensuring correct **profit rounding** and accurate **Total Profit** sums in the aggregate table.
SQL Validation Tests	Confirms the aggregate table supports the expected output of the required downstream SQL queries (e.g., checking `Profit by Year` amounts).