Enterprise-Grade Data Engineering Project Plan using PySpark & Databricks

---

Project Overview

Project Title: Retail Customer & Sales Insights Platform
Source Dataset: Brazilian E-Commerce Public Dataset by Olist
Processing Type: Batch
Environment: Databricks Community Edition
Tech Stack: PySpark, Delta Lake, SQL, Python, Databricks Jobs/Workflows

---

Dataset Components

- Customers (master data with changes over time)
- Orders (each row is a single order)
- Order Items (line items per order)
- Products (catalog with changing prices)
- Returns (inferred from canceled/refunded orders)
- Store Locations (approximated from customer zip codes)

Project Structure

- /project_root/
- ├── /raw_data/
- ├── /notebooks/
- │   ├── bronze/
- │   ├── silver/
- │   ├── gold/

- │   └── utils/
- ├── /configs/
- ├── /scripts/
- └── /output/

PHASE 1: BRONZE LAYER (Raw Ingestion)

Simple Tasks

- **Read CSV/Parquet files into DataFrames:** Load raw datasets using PySpark.
- **Add ingestion timestamp column:** Track when data was ingested.
- **Write raw data to Delta format:** Persist raw data using Delta Lake format for versioning.
- **Create tables if not exist:** Initialize Delta tables programmatically.

Medium Tasks

- **Handle corrupt records using `badRecordsPath`:** Save bad records during ingestion to review later.
- **Add metadata (file name, load time):** Store additional context about each file ingested.
- **Track schema versions:** Save schema details per file to detect and manage drift.
- **Parameterize file paths and formats:** Use widgets/config files to make paths dynamic.

Complex Tasks

- **Ingest dynamically varying schemas:** Support flexible file schemas using inferred logic.

- **File tracker and audit log table:** Maintain a tracking table with file-level metadata and job status.
- **Partial failure support (continue on error):** Ensure robust ingestion by skipping only failed files.
- **Archive processed files:** Move files to an archive folder after successful processing.

Advanced Tasks

- **Parallel ingestion logic:** Speed up ingestion using parallel file processing.
- **Build schema registry table:** Keep historical record of schema versions per dataset.
- **Generate profiling summary per file:** Automatically generate statistics like min, max, null count.
- **Replay mechanism for past ingestions:** Re-run ingestion logic for previously processed data on demand.
- **Redact/mask sensitive fields (e.g., emails):** Apply masking to confidential information.

PHASE 2: SILVER LAYER (Cleansing + Transformation)

Simple Tasks

- **Rename columns to snake_case:** Standardize column naming convention.
- **Type casting and null checks:** Ensure correct data types and handle nulls.
- **Join orders with items and products:** Denormalize data for downstream use.
- **Apply UDFs for parsing:** Use user-defined functions for custom logic.

Medium Tasks

- **Deduplication using ROW_NUMBER():** Remove duplicates based on business logic.
- **Apply validation rules and remove bad rows:** Filter out invalid rows using rule sets.
- **Use broadcast joins for small dimensions:** Improve join performance.
- **SCD Type 1 for dimension tables:** Overwrite older data with latest updates.

Complex Tasks

- **SCD Type 2 implementation with Delta Merge:** Track historical changes with versioned rows.
- **Late-arriving data support:** Update tables when out-of-order files are received.
- **Versioned/historical tables:** Keep complete change history for business audit.
- **Data quality report table:** Summarize row-level validation results.

Advanced Tasks

- **Rule-driven validation engine (JSON config):** Configure rules dynamically from external configs.
- **DQ scoring framework:** Assign data quality scores to batches or records.
- **Transformation step registration framework:** Modularize transformations with step-wise chaining.
- **Generate column-level lineage metadata:** Record the source and logic of each transformed column.

PHASE 3: GOLD LAYER (Analytics & Business KPIs)

Simple Tasks

- **Daily sales by region:** Aggregate sales grouped by region and date.
- **Revenue by product category:** Calculate revenue by grouping on product categories.
- **Top 5 customers by order volume:** Identify high-value customers by total purchases.

Medium Tasks

- **Customer Lifetime Value (CLTV):** Sum total spend over customer lifespan.
- **Rolling sales (7/30-day):** Use window functions to track trends.
- **Star schema with fact/dim tables:** Organize data warehouse structure for BI tools.
- **Workflow scheduling:** Automate daily/weekly refresh using Databricks Workflows.

Complex Tasks

- **Parameterized KPI builder framework:** Generate KPIs dynamically using parameters.
- **Trend comparison tables (YoY, MoM):** Compare current metrics with previous periods.
- **Delta rollback support (version restore):** Restore tables to previous versions using Delta Time Travel.
- **Dimension snapshot tables:** Maintain daily snapshots for slowly changing dimensions.

Advanced Tasks

- **Multi-level aggregates (day/week/month):** Build tables with different granularity.
- **Churn model-ready dataset:** Engineer features for ML churn prediction.
- **Seasonality pattern detection:** Analyze ordering patterns using calendar/time dimensions.

- **Co-occurrence matrix for recommendation:** Track product pairs frequently bought together.
- **Push data to external system (e.g., JDBC/API):** Export data for consumption by external apps.

Bonus Features

- **Config-driven pipelines using JSON/YAML:** Centralize control over file paths, rules, and logic.
- **Retry logic and exception handling utilities:** Ensure pipelines recover gracefully on failure.
- **Custom logging with Delta sink:** Persist log records in structured, queryable form.
- **Job execution tracker table:** Record job metadata like start time, status, and row count.
- **Full notebook parameterization (widgets):** Use widgets to dynamically pass parameters.

Next Steps

1. Download and load the Olist dataset into `/raw_data/`
2. Begin with Bronze ingestion for "orders" dataset
3. Use modular notebooks and build utilities for reuse
4. Proceed to Silver, then Gold with increasing complexity
