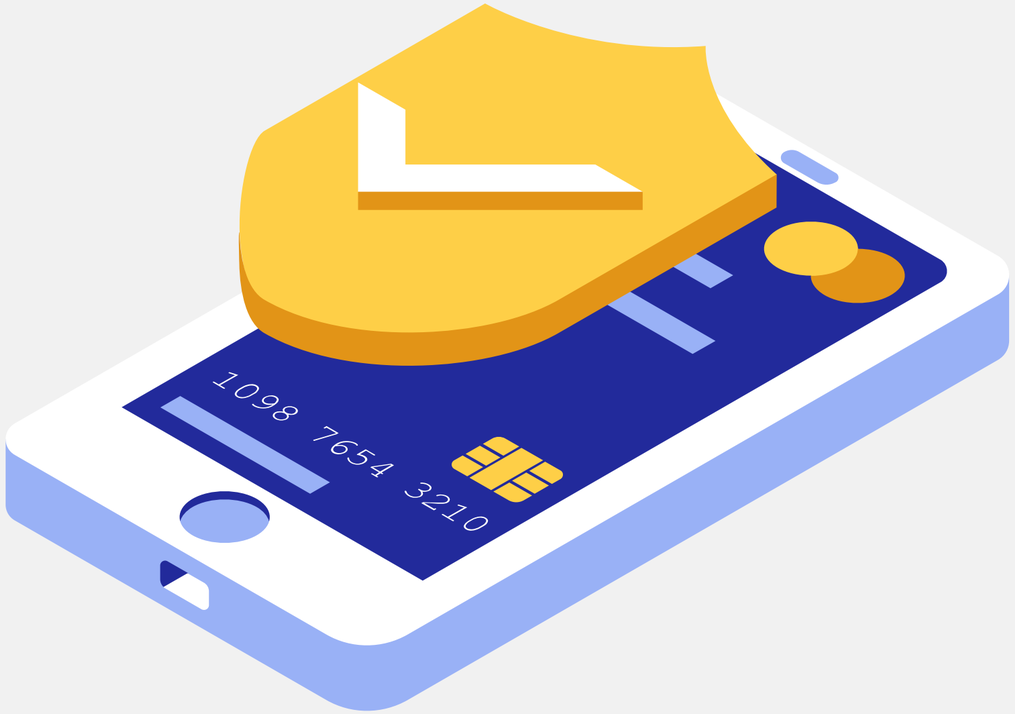
Data Processing Pipelines

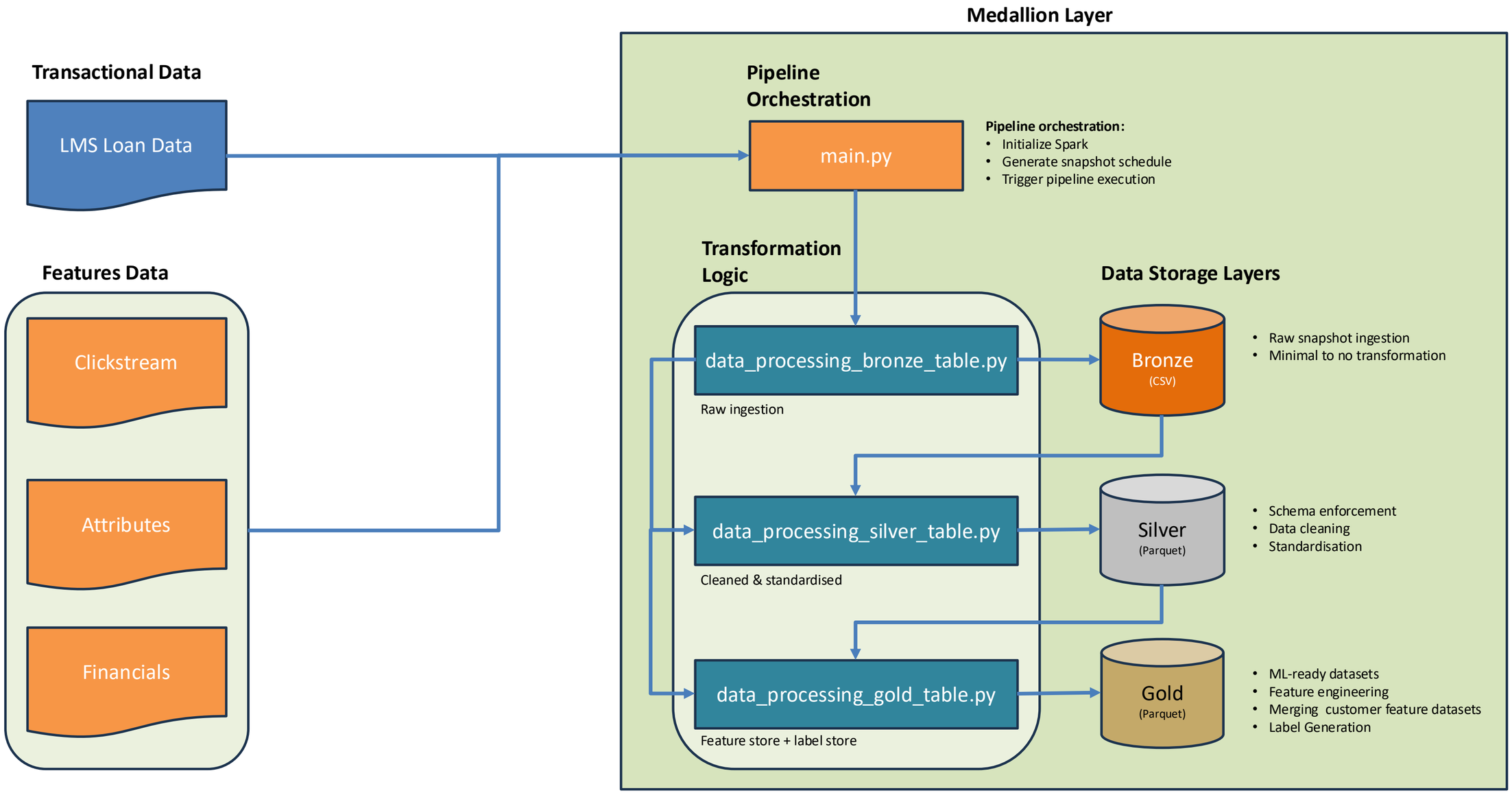
Assignment Objective

To build an end to end data processing pipeline that transforms raw loan and customer data into ML-ready feature and label stores using the Medallion Architecture.

The pipeline applies data ingestion, cleaning, schema enforcement, feature engineering and modular PySpark processing across **bronze, silver and gold** layers. It also uses Docker and GitHub to support a more reproducible machine learning engineering workflow.



Medallion Architecture Design



Bronze Layer Implementation: Raw Data Ingestion

Raw LMS and customer feature datasets are ingested using PySpark filtered by snapshot_date and stored as monthly CSV partitions in the bronze datamart. Minimal transformation is applied to preserve source data integrity and maintain traceability for downstream silver and gold processing.

Input Sources

- lms_loan_daily.csv
- feature_clickstream.csv
- features_attributes.csv
- features_financials.csv

Implementation Steps

1. Read raw CSV files using PySpark.
2. Filter records by monthly snapshot_date.
3. Save each monthly snapshot as a separate bronze file.
4. Store outputs under the bronze datamart folders.

Output Tables

- bronze_loan_daily_YYYY_MM_DD.csv
- bronze_clickstream_YYYY_MM_DD.csv
- bronze_attributes_YYYY_MM_DD.csv
- bronze_financials_YYYY_MM_DD.csv

Design Rationale

- Preserves raw source data for traceability.
- Keeps bronze layer close to the original source.
- Enables repeatable monthly backfill processing.
- Foundation for downstream silver processing.

Silver Layer Implementation: Cleaning and Standardisation

The silver layer reads bronze monthly snapshots, enforces schema, removes duplicates, cleans invalid numeric values, converts date fields and derives repayment indicators such as mob and dpd.

The cleaned outputs are stored as Parquet files and used as the foundation for gold-level feature and label generation.

Input Sources

- bronze_loan_daily_YYYY_MM_DD.csv
- bronze_clickstream_YYYY_MM_DD.csv
- bronze_attributes_YYYY_MM_DD.csv
- bronze_financials_YYYY_MM_DD.csv

Key Processing Steps

1. Load monthly bronze loan and feature files.
2. Enforce column data types.
3. Remove duplicate feature records.
4. Clean invalid characters such as _ in numeric fields.
5. Extreme or unrealistic values were replaced with nulls.
6. Convert snapshot_date into date format.
7. Create loan repayment features* such as: dpd, mob installments_missed, first_missed_date

Output Tables

- silver_loan_daily_YYYY_MM_DD.parquet
- silver_clickstream_YYYY_MM_DD.parquet
- silver_attributes_YYYY_MM_DD.parquet
- silver_financials_YYYY_MM_DD.parquet

Design Rationale

- Improves data quality before feature engineering.
- Standardises schema for downstream joins.
- Converts raw CSV data into Parquet for scalable storage and analytics.
- Creates reliable intermediate tables for gold feature and label stores.

features are further elaborated in slide 7*

Gold Layer Implementation

The gold layer converts cleaned silver tables into final ML-ready outputs. It creates a default label based on 30dpd_6mob, joins customer feature datasets, engineers financial risk indicators, and stores the final feature and label stores as Parquet files for future model training.

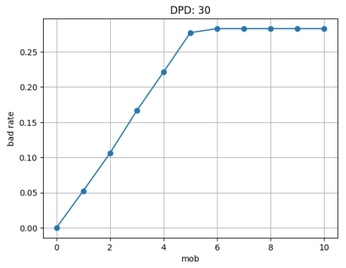
Label store and feature store are stored separately to support reusable training pipelines.

Input Sources

- silver_loan_daily_YYYY_MM_DD.parquet
- silver_clickstream_YYYY_MM_DD.parquet
- silver_attributes_YYYY_MM_DD.parquet
- silver_financials_YYYY_MM_DD.parquet

Output Tables

- gold_label_store_YYYY_MM_DD.parquet
- gold_feature_store_YYYY_MM_DD.parquet



Key Processing Steps

1. Load cleaned silver loan and feature tables.
2. Generate default labels from loan repayment behaviour.
3. Filter loans at mob = 6.
4. Assign default label using dpd >= 30.
5. Join attributes, financials, and clickstream features by Customer_ID and snapshot_date.
6. Create engineered features*: debt_to_income_ratio, emi_to_salary_ratio, high_credit_utilization_flag

Design Rationale

- The gold layer separates labels and features into reusable ML-ready stores.
- This supports future model training while keeping feature engineering and label generation traceable within the Medallion Architecture.

features are further elaborated in slide 7*

Feature Engineering & Label Design

Layer	Feature/ Label	Formula/ Logic	Business Meaning
Silver	dpd	Days between snapshot date and first missed payment	Measures delinquency severity
	mob	Instalment number	Loan repayment progression
	installments_missed	Ceiling of Overdue_amt/ due_amnt	Rounded up number of missed installment
	first_missed_date	snapshot_date – installments_missed months	Capture the 1st time period where the payment was due
Gold	label	dpd >= 30 at mob = 6	Binary loan default prediction target
	debt_to_income_ratio	Outstanding Debt / Annual Income	Debt burden risk
	emi_to_salary_ratio	EMI / Monthly Salary	Repayment affordability
	high_credit_utilization_flag	Utilization > 50%	Indicator of high credit usage risk

Data Quality & Validation Checks ✓

Duplicate Checks	Duplicate records were removed from feature datasets using dropDuplicates() before downstream joins and feature engineering. Validation Logic • Checked duplicate Customer_ID and snapshot_date combinations. • Prevented duplicate customer feature records in the gold feature store.
Schema enforcement	Column data types were explicitly enforced using PySpark casting to standardise loan and feature datasets.
Invalid Value Cleaning (Silver Layer)	• Removed _ characters from numeric financial columns. • Converted cleaned values into numeric datatypes. Extreme or unrealistic values were replaced with nulls based on cutoff thresholds derived from binning the latest cumulative dataset. • Age values < 0 or > 100 → NULL. • Num_Bank_Accounts, Num_Credit_Card values <0 or > 15 → NULL. • Interest_Rate values < 0 or > 100 → NULL. • Delay_from_due_date, Num_Credit_Inquiries values <0 or > 15 → NULL.
Derived Repayment Validation	Repayment behaviour features were engineered and validated using loan overdue information. • mob • installments_missed • first_missed_date • dpd

ML Compataility and Leakage Control

ML Compatibility

- Gold feature and label stores are structured for downstream model training.
- Feature and label stores are kept separate to prevent leakage, improve reusability and support different model training use cases (can be joined later during model training using Customer_ID & snapshot_date).
- Parquet format improves scalability and analytical efficiency.

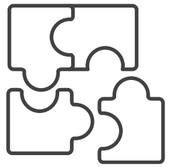
Leakage Control

- Labels are generated using repayment behaviour at **mob = 6**.
- Feature joins use historical customer information available at **snapshot time**.
- Future repayment outcomes are used only for label generation, not feature creation.

Limitations and Improvements

Current Limitations

- Pipeline currently runs locally using local[*] Spark mode.
- Batch processing only; no real-time streaming ingestion. Train, validation, test, and OOT data split are not yet implemented.
- Feature store metadata cataloging is not implemented.
- No workflow orchestration tool such as Airflow is used yet.
- No automated model monitoring or retraining pipeline.



Future Improvements

- Deploy pipeline on distributed cloud infrastructure.
- Integrate Airflow for automated scheduling and orchestration.
- Add train, validation, test, and OOT dataset generation.
- Introduce online feature store for real-time inference.
- Add model monitoring and data drift detection.
- Implement CI/CD pipeline and container registry deployment.