SMU
SINGAPORE MANAGEMENT UNIVERSITY

School of
Computing and
Information Systems

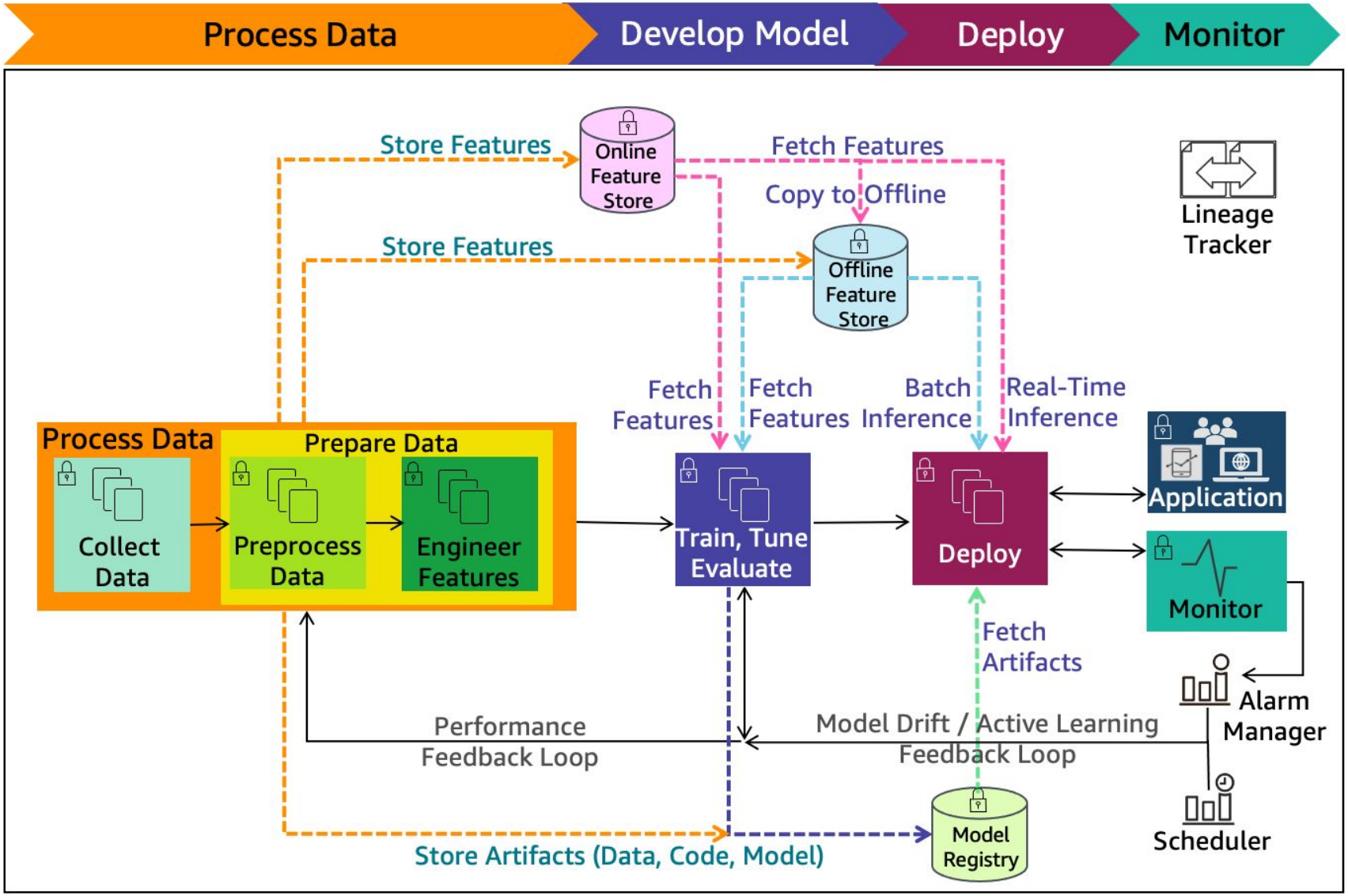
CS611 Machine Learning Engineering

Lecture 3: Data Preparation

Ulysses CHONG Min Zhen

ulysseschong@smu.edu.sg

Machine Learning Lifecycle



Outline

Data Preparation

- Medallion Architecture
- Data cleaning and preprocessing
- Splitting data
- Feature engineering
- Feature store

Big Data Processing with PySpark

Lab

- Data preprocessing with PySpark

Group Project + Assignment 1

Class Case Study Discussion

Create 1 label (30 mins)

Create 1 feature (20 mins)

Case Study

You are the Data Scientist at a bank.
You are tasked to build a credit scoring model to predict if a customer will default on a loan.
At the moment, you have access to the loan management system which shows daily repayments of loans by customers.

Activity 1: Label engineering
Align modeling approach (regression? classification? smth else?)
Align label definition
Write pseudo-code to engineer label (medallion architecture compliant)

Activity 2: Feature engineering
Identify and design 1 feature
Write pseudo-code to engineer feature (medallion architecture compliant)

Case Study (data available)

Loan Management System (LMS)

SELECT * FROM lms.loan_daily LIMIT 100

loan_id	customer_id	loan_amt	due_amt	paid_amt	overdue_amt	balance_amt	snapshot_date
1	123	1000	100	100	0	700	2024-08-01
1	123	1000	100	100	0	800	2024-07-01
1	123	1000	100	100	0	900	2024-06-01
2	234	1000	100	0	300	900	2024-08-01
2	234	1000	100	0	200	900	2024-07-01
2	234	1000	100	0	100	900	2024-06-01
2	234	1000	100	100	0	900	2024-05-01

Case Study (data available)

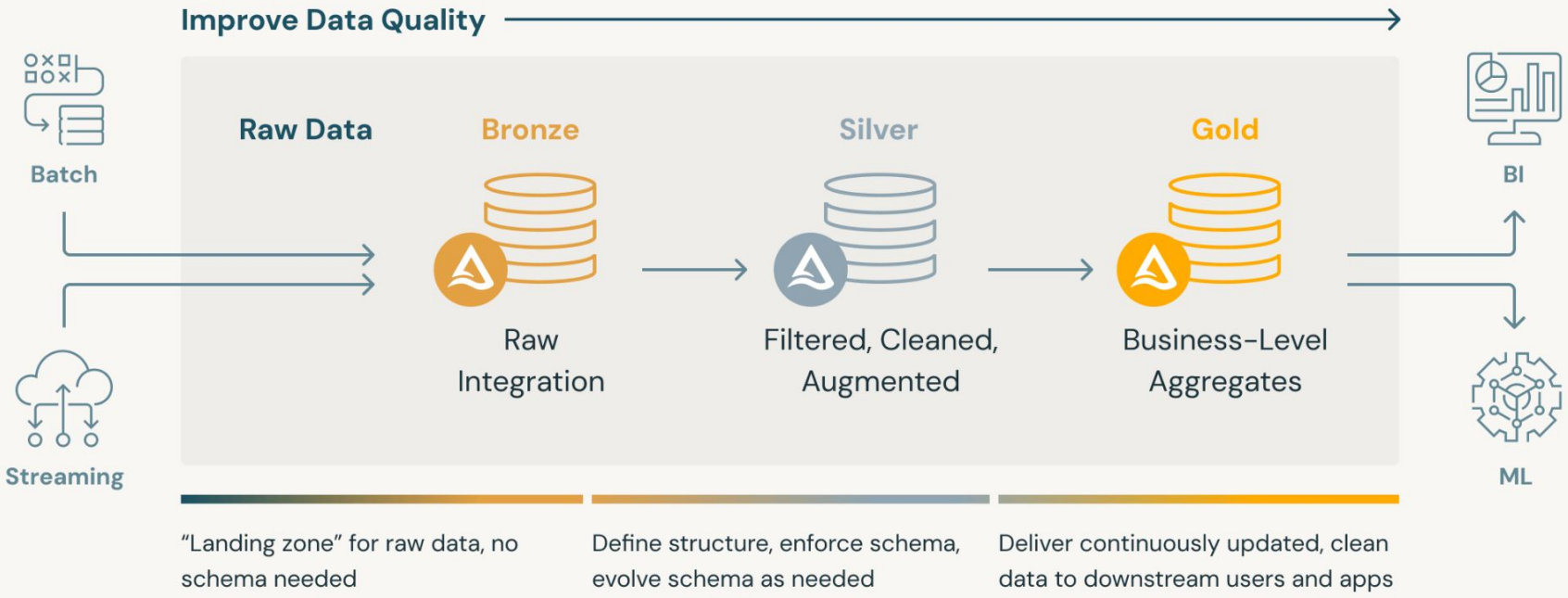
Loan Management System (LMS)

SELECT * FROM lms.loan_daily LIMIT 100

loan_id	customer_id	loan_amt	due_amt	paid_amt	overdue_amt	balance_amt	snapshot_date
3	345	1000	100	100	0	300	2024-08-01
3	345	1000	100	350	0	400	2024-07-01
3	345	1000	100	0	250	750	2024-06-01
3	345	1000	100	100	150	750	2024-05-01
3	345	1000	100	50	150	850	2024-04-01
3	345	1000	100	0	100	900	2024-03-01
3	345	1000	100	100	0	900	2024-02-01

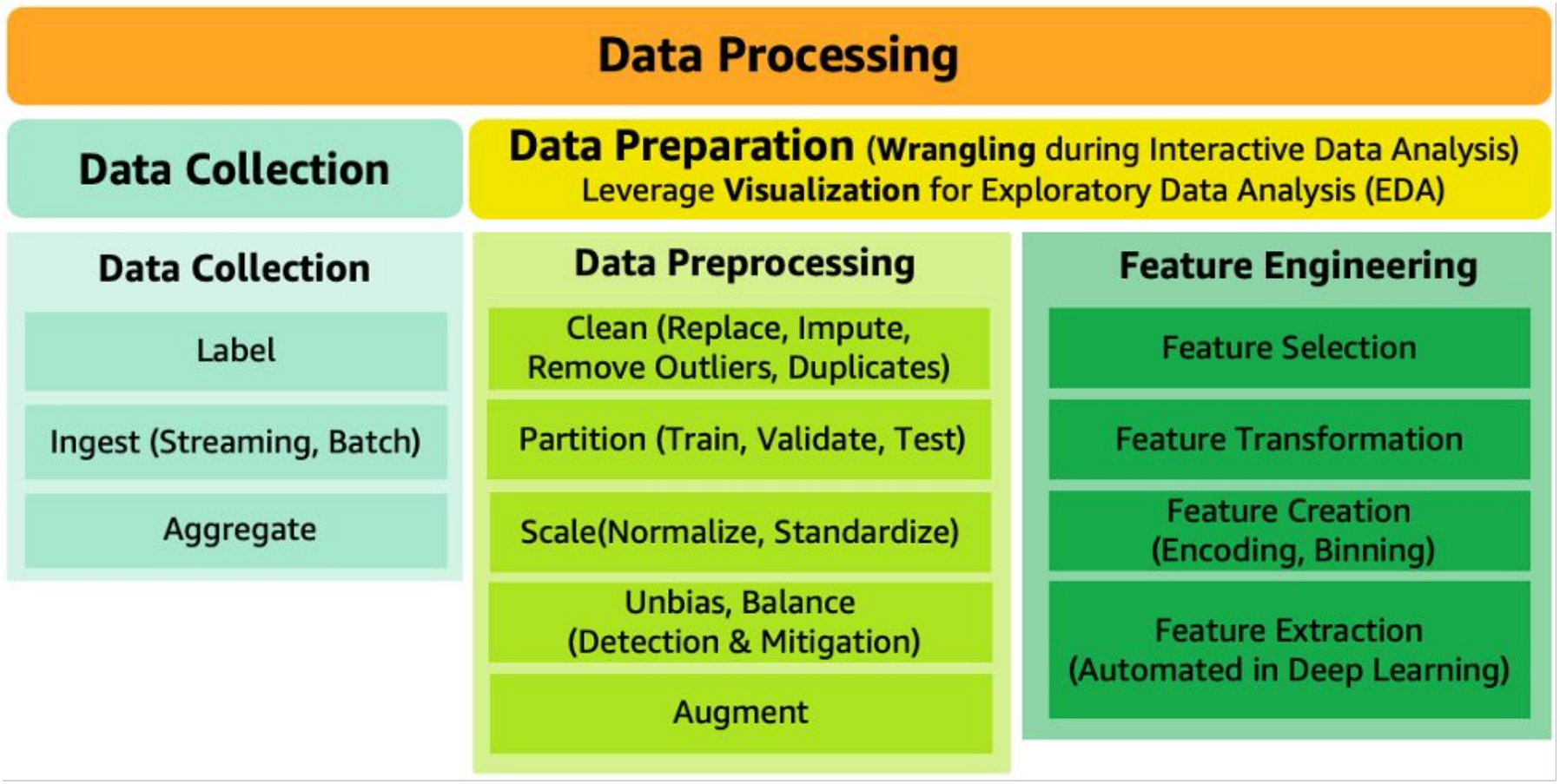
Data Preparation

Organizing data with Medallion Architecture



Read here for bronze / silver / gold guidance: https://www.databricks.com/glossary/medallion-architecture

Data Processing



Data Preparation

- Raw data are often not compatible with algorithm input format
- Even worse, real world dataset are generally very noisy
  - For example: inconsistent unit of measures, inconsistent values representing same meaning, wrong values, missing values, etc.
- Expect your input to be the same as the one in your deployment environment

Data Preparation

- We generally employ several exploratory data analysis techniques such as descriptive statistics, correlation analysis, data visualization
- Popular open source python libraries for data preparation:
  - pandas (https://pandas.pydata.org/docs/user_guide/index.html)
  - numpy (https://numpy.org/doc/stable/user/index.html)
  - scikit-learn (https://scikit-learn.org/stable/tutorial/index.html)

Data Cleaning

- Several data cleaning tasks:
  - Simple rules to handle inconsistency in data
  - Duplicates
  - Missing values: remove or impute
  - Outlier: remove or impute

- Take note of the risk of removing data instances / features

Splitting Data

- We do not use all data in our dataset for machine learning training
- Split dataset into training, validation, and testing set
  - Common splitting: 80% training, 10% validation, and 10% testing. Ensure you have sufficient testing set.
  - For imbalance datasets, maintain the same class distribution on all sets. Use stratified splitting.
- For very small dataset, people may use cross validation
- Be careful of data leakage
  - It happens when your model gets data during inference that it should not have access to.

Detecting and Mitigating Bias

- Biases are imbalances in the accuracy of prediction across different groups such as age or income bracket.
  - https://aws.amazon.com/blogs/machine-learning/learn-how-amazon-sagemaker-clarify-helps-detect-bias/

- Measuring data bias: https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-measure-data-bias.html

Feature Engineering

- Feature creation
  - Example: one-hot encoding, binning, calculated features
- Feature selection
  - Example: selecting subset of relevant features
- Feature transformation
  - Example: log transform, creating domain specific features
- Feature extraction
  - Example: PCA, LDA, bag of words, n-gram

Feature Store

- Centralized store for features and associated metadata so that features can be easily discovered and reused.
  - Online feature store: for low latency real-time inference use cases
  - Offline feature store: for training and batch inference
- Features are stored in a collection called a feature group.
- Metadata may include short description, storage configuration, event time, and tags to store additional information such as the author, data source, version, etc.
- Feature group names and associated metadata should not contain any personal identifiable information (PII) or confidential information.

15 mins break

.. and then you'll change the way you use pandas forever!!

Big Data Processing with PySpark

PySpark; “Pandas’s buff sibling on steroids” - anon

PySpark

source: https://www.youtube.com/watch?v=IELMSD2kdmk

PySpark

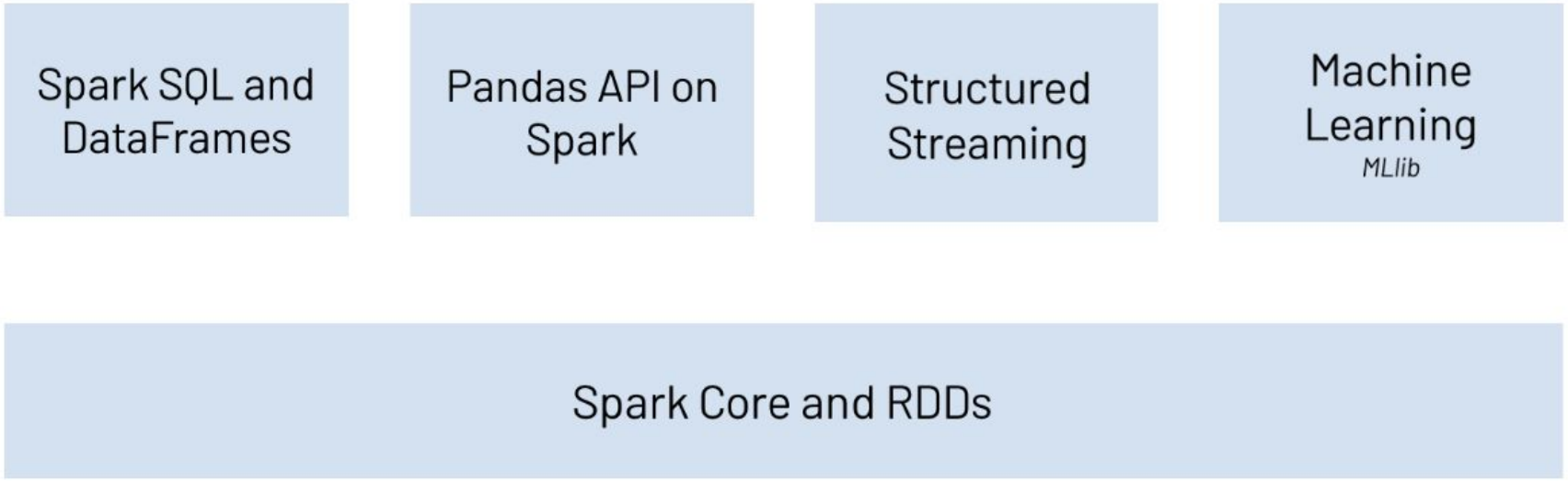
"PySpark is the Python API for Apache Spark. It enables you to perform real-time, large-scale data processing in a distributed environment using Python. It also provides a PySpark shell for interactively analyzing your data.

PySpark combines Python's learnability and ease of use with the power of Apache Spark to enable processing and analysis of data at any size for everyone familiar with Python.

PySpark supports all of Spark's features such as Spark SQL, DataFrames, Structured Streaming, Machine Learning (MLlib) and Spark Core."

source: https://spark.apache.org/docs/latest/api/python/index.html

PySpark



source: https://spark.apache.org/docs/latest/api/python/index.html

PySpark

Pandas ↔ Polars ↔ SQL ↔ PySpark

blog.DailyDoseofDS.com

Operation	Pandas	Polars	SQL	PySpark
Import	import pandas as pd	import polars as pl	-	from pyspark.sql import SparkSession spark = SparkSession.builder.appName("ABCD")
Read CSV	df = pd.read_csv(file)	df = pl.read_csv(file)	LOAD DATA INFILE 'data.csv' INTO TABLE table FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' IGNORE 1 ROWS;	df = spark.read .csv("data.csv")
Print first 10 (or k) rows	df.head(10)	df.head(10)	SELECT * FROM table LIMIT 10;	df.show(10)
Dimensions	df.shape	df.shape	SELECT count(*) FROM table;	df.count()
			SELECT count(*) FROM INFORMATION_SCHEMA.COLUMNS where TABLE_NAME = 'table';	len(df.columns)
Datatype	df.dtypes	df.dtypes	DESCRIBE table;	df.printSchema()
Select column(s)	df[["col1", "col2"]]	df[["col1", "col2"]]	SELECT column FROM table;	df.select("col1", "col2")
Filter Data	df[df.column > 10]	df[df.column > 10]	SELECT * FROM table where column>10;	df.filter(df["column"]>10)
		df.filter(pl.col("column") > 10)		
Sort	df.sort_values("column")	df.sort("column")	SELECT * FROM table ORDER BY column;	df.orderBy("column")
Fill NaN	df.column.fillna(0)	df.column.fill_nan(0)	UPDATE table SET column=0 WHERE column IS NULL;	df.na.fill(0)
Join	pd.merge(df1, df2, on ="col", how="inner")	df1.join(df2, on="col", how="inner")	SELECT * FROM table1 JOIN table2 ON (table1.col = table2.col);	df1.join(df2, on="col", how="inner")
Concatenate	pd.concat((df1, df2))	pl.concat((df1, df2))	SELECT * FROM table1 UNION ALL table2;	df1.union(df2)
Group	df.groupby("column"). agg_col.mean()	df.groupby("column"). agg(pl.mean("agg_col"))	SELECT column, avg(agg_col) FROM table GROUP BY column;	df.groupBy("column"). agg(avg("agg_col"))
Unique values	df.column.unique()	df.column.unique()	SELECT DISTINCT column FROM table;	df.select("column"). distinct()
Rename column	df.rename(columns = {"old_name": "new_name"})	df.rename(mapping = {"old_name": "new_name"})	ALTER TABLE table RENAME COLUMN old_name TO new_name;	df.withColumnsRenamed({"old_name": "new_name"})
Delete column	df.drop(columns = ["column"])	df.drop(name = ["column"])	ALTER TABLE table DROP COLUMN column;	df.drop("col1", "col2")

source:

https://blog.dailydoseofds.com/p/15-pandas-polars-sql-pyspark-translations

Lab

Lab Activities

1. Set up lab 2 git folder
2. Load Bronze table with PySpark and process to Silver table
3. Save Silver table as parquet file
4. Build Gold table with PySpark for engineered labels
5. Save Gold table as parquet file
6. (Bonus) convert to py executable scripts

Dataset to be provided by instructor

original source datasets for instructors: https://www.kaggle.com/datasets/parisrohan/credit-score-classification

Homework

Assignment 1 released!

Deadline: by 21 May 2025

Group Project

Form groups of 6-7 people

Start working towards your proposal presentation Week 6

Questions?

Thank You