# Programação Funcional

Project 1 - Extract Transfer Load

Letícia Coêlho Barbosa

Mar 2025

## Contents

# 1 Overview

## 1.1 Description

As data input, we consume two tables. One stores order information and the other stores product information in those orders.

| id | client_id | order_date | status | origin |
|---|---|---|---|---|
| 1 | 112 | 2024-10-02T03:05:39 | Pending | P |
| 2 | 117 | 2024-08-17T03:05:39 | Complete | O |
| 3 | 120 | 2024-09-10T03:05:39 | Cancelled | O |

Table 1: Order Information

| order_id | product_id | quantity | price | tax |
|---|---|---|---|---|
| 12 | 224 | 8 | 139.42 | 0.12 |
| 13 | 213 | 1 | 160.6 | 0.16 |
| 2 | 203 | 7 | 110.37 | 0.15 |

Table 2: Product Information in Orders

The goal of the project is to aggregate information related to the total amount paid and the total taxes for each order.

The return should be filtered according to the order status (*Pending* — *Cancelled* — *Complete*) and the desired store type (*O - Online* — *P - Physical*).

| Order ID | Total Amount | Total Taxes | Date | Status | Origin |
|---|---|---|---|---|---|
| 4 | 3086.71 | 422.5537 | 2024-03-11T03:05:39 | Pending | O |
| 6 | 1757.79 | 249.7358 | 2024-04-18T03:05:39 | Pending | O |
| 9 | 785.2 | 109.928 | 2025-01-08T03:05:39 | Pending | O |
| 12 | 2671.06 | 351.6412 | 2024-08-28T03:05:39 | Pending | O |
| 16 | 1654.23 | 206.3961 | 2024-08-07T03:05:39 | Pending | O |
| 20 | 750.14 | 88.2166 | 2024-04-12T03:05:39 | Pending | O |

Table 3: Aggregated Order Data

## 1.2 Requirements

1. Ocaml
   - Install Ocaml
     https://ocaml.org/docs/installing-ocaml
   - Install necessary packages
     Following the project instructions : https://github.com/leticiacb1/ETL-FunctionalProgramming/
2. Dune
   Install Dune : https://dune.build/
3. SQLite3
   Install SQLite on Ubuntu: https://www.geeksforgeeks.org/how-to-install-sqlite-3-in-ubuntu/

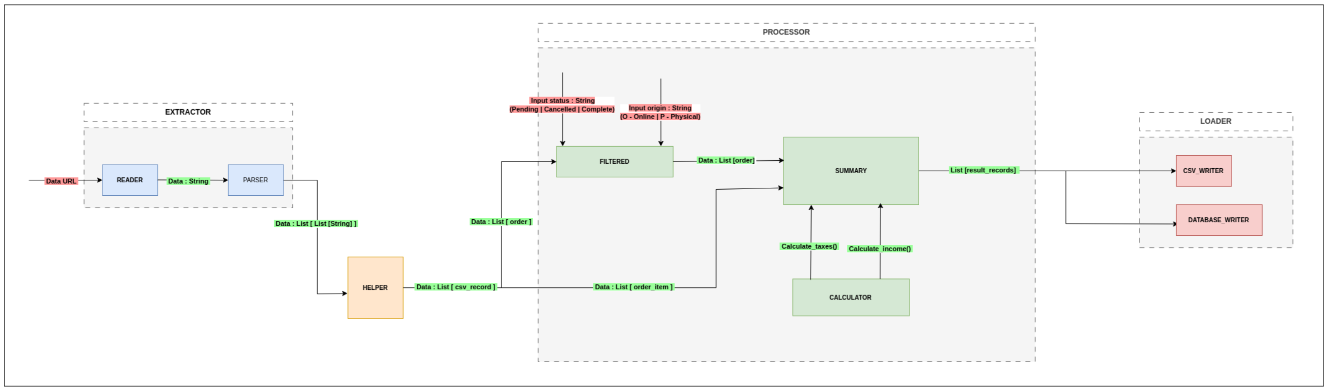
## 1.3 Code Division

- `src/` – Code files.
  - `extractor/` – Extracts data from a given URL.

    - `extractor_utils.ml` – Utility functions for extraction.
    - `reader.ml` – Functions for reading data sources. Extracts content from a URL as a single large string.
    - `parser.ml` – Functions for parsing data. Splits the large string received from `reader.ml` into lines and removes empty lines from the content.
  - `helper/` – Helper functions and modules.
    - `helper_utils.ml` – General utility functions.
    - `order.ml` – Functions used by `mapper.ml` to transform a list of strings into the `Order` type.
    - `order_item.ml` – Functions used by `mapper.ml` to transform a list of strings into the `OrderItem` type.
    - `mapper.ml` – Functions for mapping data. Converts the list of strings received from `parser.ml` into a more readable data type, a `csv_record` (`Order` — `OrderItem`).
  - `processor/` – Modules for processing data.
    - `filter.ml` – Functions for filtering order data using user-input status and origin.
    - `calculator.ml` – Functions used by `summary.ml` to calculate income and taxes from orders.
    - `summary.ml` – Aggregates `Order` and `OrderItem` data to return a summary, grouped by `order_id`, with total income and taxes for each order.
  - `loader/` – Modules for loading data.
    - `csv_writer.ml` – Functions for writing CSV files.
    - `database_writer.ml` – Functions for writing to the database.
  - `shared/` – Shared constants and types.
    - `constants.ml` – Constant values used throughout the project.
    - `types.ml` – Type definitions used in the project (`Order`, `OrderItem`, `csv_record`, `result_record`).
  - `main.ml` – The main entry point of the project. Defines the pipeline of processes.
  - `dune` – Dune configuration file.

- `test/` – Unit test files.
  - `dune` – Dune configuration file.
  - `extractor_test.ml` – Tests for the extractor module.
  - `helper_test.ml` – Tests for the helper module.
  - `loader_test.ml` – Tests for the loader module.
  - `processor_test.ml` – Tests for the processor module.

## 1.4 High Architeture Diagram



## 2 Project Requirements

### 2.1 Mandatory

1. ✓ The project must be implemented in OCaml.
2. ✓ To calculate the output, you need to use map, reduce, and filter functions.
3. ✓ The code must include functions for reading and writing CSV files. This will result in impure functions.
4. ✓ Separate impure functions from pure functions in the project's files.
5. ✓ The input must be loaded into a list structure of Records.
6. ✓ The use of Helper Functions to load fields into a Record is mandatory.
7. ✓ A project report must be written, indicating how the steps were built. This is similar to a guide for someone who would redo the project in the future. You should declare whether or not you used Generative AI in this report.

### 2.2 Optional

1. ✓ Read the input data from a static file on the internet (exposed via HTTP)
2. ✓ Save the output data in an SQLite Database
3. ✓ Join the tables before starting the Transform step
4. ✓ Organize the ETL project using dune
5. ✓ Document all generated functions
6. ✗ Generate an additional output that contains the average revenue and taxes paid, grouped by month and year.
7. ✓ Create complete test files for pure functions.

## 3 References

1. https://ocaml.org/manual/5.3/index.html/
2. https://dune.readthedocs.io/en/stable/
3. IA Generative Usage

   Generative AI was used for the following purposes in the development of the project :

   - Functions Documentation
   - Code Refactoring and Optimization Suggestions
   - Dune usage (general questions)
   - Bug fixes
   - Unit test examples
   - Improve the writing of this document