Bond Valuation Assignment

Description

Every Monday morning the portfolio management desk receives a fresh file with bond positions—some already held, some under review as potential investments. Before the weekly investment meeting, these positions must be valued consistently so managers can compare them fairly. The same valuations also flow into risk and accounting processes downstream, where they are automatically consumed.

The engineering team is responsible for turning the raw file into usable valuations. The task is straightforward in principle: take the input file, apply the valuation rules, and produce output that portfolio managers can read and other systems can reuse. There is no fixed requirement about how the process should be delivered; the technical implementation details are up to you. It could be a script, an API, a batch process, or something else, as long as it produces reliable and consistent results and serves the business goal.

The input file arrives in a tabular format (see attached csv).

Calculation Rules

You do not need to build full bond pricing models. A **DiscountFactor (DF)** is provided for each row — use it directly.

- **Coupon Bonds** (Type = Bond)

  Coupon per period (C_{pp})

  $$C_{pp} = \frac{Rate}{Payments\ per\ year}$$

  Number of periods (n)

  $$n = Years\ to\ maturity \times Payments\ per\ year$$

  Present Value (PV):

  $$PV = \left(\left(1 + C_{pp}\right)^{n} \times Face\ value\right) \times DF$$

- **Zero-Coupon Bonds** (Type = Zero-Coupon)

  Present Value (PV):

  $$PV = \left((1 + Rate)^{Years\ to\ Maturity} \times Face\ value\right) \times DF$$

**DF is the discount factor to maturity, for training purposes it's calculated in the input dataset assuming 4.5% annual yield.*

Expected Output

For each bond, the output must at least contain:

- BondID, Type, PresentValue (rounded to 2 decimals) and any additional columns you think are needed.

The output will be reviewed in daily portfolio meetings to give managers a clear overview of present values across positions. If the process fails, produces incomplete results, or outputs values without explanation, it directly affects their ability to make investment decisions in those meetings. At the same time, the results flow into risk and accounting systems downstream.

The process must therefore be **robust, reliable, and transparent**: portfolio managers should be able to trust what they see and the team should be able to trace and understand issues if they occur.

Constraints

- Keep external dependencies to a minimum; if you use any, explain why.
- Focus on clarity, correctness and reusability.
- Include clear instructions on how to run the solution (e.g. README, usage guide). The expectation is that another team member could take what you deliver and run it successfully without needing extra explanations.