Test Plan

This section describes the core tests that will validate each VaR/ES module. Tests can be automated with `pytest` or run interactively.

---

1. Flat-Price Unit Test

Purpose:
Verify that when prices never change, VaR and ES outputs are zero for all modules (except ES for the EWM module, which is not implemented).

Data Setup:

- A `pandas.Series` of constant values (e.g. 100.0) over at least one window's length (≥ 1,260 trading days).

Test Steps:

1. Call `compute_var` and `compute_es` on the flat series for:
   - `parametric5yr` (both VaR & ES)
   - `historical` (both VaR & ES)
   - `montecarlo` (both VaR & ES)
2. Call `compute_var` (only) on the EWM module.
3. Assert that all returned VaR and ES series are identically zero.

Pass Criteria:

- VaR and ES series for parametric5yr, historical, montecarlo are zero.
- EWM VaR series is zero.

---

2. Parametric Closed-Form Consistency

Purpose:
Check that `parametric5yr`'s VaR and ES match the analytical GBM formulas when σ=0 (deterministic drift).

Data Setup:

- Simulate a deterministic GBM path of length ≥ 6 × 252 days with known drift μ and σ=0, so that log-returns = (μ–½σ²)/252 each day.

Test Steps:

1. Compute the theoretical 5-day VaR and ES in closed form, using the same daily-drift scaling your code uses.
2. Call `parametric5yr.compute_var(prices, var_level)` and `compute_es(prices, es_level)`.
3. Compare the last value of each series to the theoretical value.

Pass Criteria:
Exact match (within floating-point tolerance) between code output and theory.

---

3. Backtest Exception Frequency

Purpose:
Validate the nominal exception rate for `parametric5yr` VaR on a stochastic GBM path.

Data Setup:

- Simulate 10 years (≈ 2,520 trading days) of GBM daily prices with moderate volatility.

Test Steps:

1. Compute the 5-day VaR series at 99% with `parametric5yr.compute_var`.
2. Compute realized 5-day P&L:

```
pnl5 = prices.shift(-5) - prices
```

3. Count exceptions where `pnl5 < -VaR`.
4. Compute frequency = exceptions / number_of_tests.

Pass Criteria: Exception frequency within ± 0.005 of 0.01.

---

4. Monte Carlo vs. Parametric Deterministic Agreement

Purpose: On a deterministic GBM (σ=0), ensure `montecarlo.compute_var` matches `parametric5yr.compute_var` exactly.

Data Setup:

- Use the same deterministic GBM path from Test 2 (σ=0, length ≥ 6 × 252).

Test Steps:

1. Compute `v_param = parametric5yr.compute_var(prices, var_level)`.
2. Compute `v_mc = montecarlo.compute_var(prices, var_level, window_days=5*252, n_sims=1_000)`.
3. Assert the two series are identical.

Pass Criteria: `pd.testing.assert_series_equal(v_param, v_mc)` passes without error.

5. Portfolio Consistency Visualization Test

Purpose:
Ensure that for the actual multi-stock portfolio, the VaR and ES time-series from all four methods evolve similarly, with no method showing a drastic divergence.

Data Setup:

1. Load the provided CSV (`software/data/portfolio.csv`) with dates as index and stock price columns.
2. Compute the portfolio series, e.g.:

```
portfolio = df.sum(axis=1)
```

Test Steps:

1. Compute the full VaR and ES series at the chosen levels (e.g. 99% VaR, 97.5% ES) for each method:
    - `parametric5yr`
    - `parametric_ewm`
    - `historical`
    - `montecarlo`
2. Plot **one** overlaid time-series graph of VaR and **one** of ES, with all four methods labeled.

```
plt.figure()
for series, label in [(var1, 'Parametric5yr'), ...]:
    plt.plot(series.index, series.values, label=label)
plt.legend(); plt.title('5-day VaR @ 99%');
plt.savefig('test_var_plot.png')
```

   and similarly for ES.
3. Visually inspect (or programmatically check) that no curve deviates sharply from the others—e.g., the pointwise ratios between any two methods remain within a moderate band (e.g. ±20%) over time.

Pass Criteria:

- Two plot files (`test_var_plot.png`, `test_es_plot.png`) are generated.
- All four method curves remain roughly aligned (no method shows a sustained, drastic deviation beyond ±20% of the group median at any date).