# Dev A Phase 3 Completion Guide

## Summary

Phase 3 implemented the full quant engine atop the Phase 2 data layer. The objective was to establish a functional Backtrader-based backtesting system using real market and macro data sourced from the Postgres data layer.

## Data Layer Integration

OHLCV price data is loaded through loader.py from the market_data table. Macro data is accessed via econ_loader.py from the econ_data table. All strategies now rely exclusively on this Phase 2 infrastructure.

## Quant Engine

The custom Cerebro wrapper (build_cerebro and run_cerebro) sets up data feeds, binds strategies, configures analyzers, and returns normalized results. Backtester stability and correctness were validated with multiple strategies.

## StrategyBase Abstraction

StrategyBase(bt.Strategy) unifies symbol-to-data mapping, abstract target-weight computation, order routing, and trade logging. Strategies implement compute_target_weights() and return a weight dict each bar.

## Implemented Strategies

Three macro strategies were implemented: GoldRealYields, USDDivergence, and CurveSteepener. Each uses macro series (e.g., DGS10, DGS2) and rolling MA logic to construct signals. All compile and run with real data.

## Interface Layer

run_backtest(strategy_id, params) composes feeds, loads price data, calls the Cerebro wrapper, and returns a JSON-serializable dict containing metrics, orders, trades, returns, period, tickers, and

parameters.

## Testing

Scripts under scripts/ demonstrate functional end-to-end execution, including test_run_backtest_full.py, test_run_backtest_gold_real_yields.py, test_run_backtest_usd_divergence.py, and test_run_backtest_curve_steepener.py.

## Outcome

The quant engine is fully functional, stable, and integrated with the Phase 2 data layer. Phase 3 deliverables are fully satisfied.