

Quant Project 1 Details:

Project 1 — Portfolio Optimisation & Risk Visualiser

1) Short description — what you're trying to do: Build a Python project that: - Downloads historical asset prices and computes returns and risk measures. - Implements classical portfolio optimisation (mean-variance / Markowitz) using a solver (e.g., cvxpy). - Adds realistic features like constraints (no shorting, weight bounds), transaction costs, and risk measures like VaR/CVaR. - Produces visual outputs: efficient frontier, portfolio weights, portfolio value over time, and interactive visualisations (Plotly/Dash or Streamlit). - Packages work into notebooks and scripts, and hosts on GitHub as a polished portfolio piece.

2) Tools & accounts: - Python 3.10+ - PyCharm Community Edition - Git installed locally - GitHub account - Optional: Conda or PyCharm-managed virtualenv

3) Libraries to install: - numpy, pandas, scipy, matplotlib, plotly, cvxpy, yfinance, scikit-learn, statsmodels, jupyter - Optional: numba, dash or streamlit, pytest, black/flake8

4) Outline of goals & deliverables: - Reproducible Jupyter notebook demonstrating workflow - Python package with modules for data, stats, optimisation, backtest, and viz - Interactive dashboard (optional) - README.md with summary, results, and instructions - Stretch: transaction-cost-aware rebalancer, rolling-window optimisation, saved figures/reports, unit tests

5) High-level workflow: 1. Data ingestion: fetch historical prices 2. Preprocessing: clean data, compute returns 3. Estimate statistics: mean returns, covariance, shrinkage covariance 4. Implement optimisers: mean-variance optimisation with constraints 5. Compute efficient frontier 6. Risk metrics: VaR/CVaR 7. Backtest / simulate portfolios 8. Visualise results 9. Package & document notebooks, scripts, README 10. Polish code, add tests, prepare for GitHub

6) PyCharm + GitHub setup: - Create project in PyCharm with virtualenv - Use VCS -> Git to initialize repo - Create GitHub repo and push - Optional: use branches for features

7) Suggested file/folder structure: portfolio-optimiser/ | — .gitignore | — README.md | — requirements.txt | — notebooks/ | — portfolio_opt_explained.ipynb | — src/portfolio_opt/ | | — **init.py** | | — data.py | | — stats.py | | — optim.py | | — backtest.py | | — viz.py | — reports/figures/ | — dash_app.py | — tests/test_stats.py

8) Step-by-step implementation plan: 1. Project repo & environment 2. Data ingestion module 3. Exploratory notebook 4. Statistics module 5. Optimisation module 6. Visualisation module 7. Backtest & VaR 8. Integration notebook 9. Add tests 10. README & docs 11. Push & polish 12. Prepare interview-friendly summary

9) Example git & pip commands provided for setting up virtualenv, installing requirements, and initial commit/push.

10) Checklist for employer impression: - Clear README, reproducible steps, images - Modular code structure
- Unit tests - Demo notebook running end-to-end - Thoughtful commit history