

Sprint 2

REQUIREMENTS

Story ID	Story description
US1	As an Analyst, I want to query portfolio returns by date range so that I can measure performance versus benchmarks. (<i>DB design completed in Sprint 1</i>)
US4	As an Analyst, I want to run a performance attribution report (contribution to return by asset/strategy) so that I can identify drivers of performance. (<i>DB design completed in Sprint 1</i>)
US3	As an Analyst, I want to compute rolling 12-month or custom rolling Sharpe for the portfolio so that I can assess risk-adjusted performance. (<i>DB design completed in Sprint 1</i>)
US6	As a PM, I want to simulate portfolio rebalancing based on target strategy allocations so that I can test different allocation scenarios. (<i>New for Sprint 2</i>)
US7	As an Analyst, I want to calculate portfolio Value at Risk (VaR) for a selected confidence interval so that I can evaluate downside risk. (<i>New for Sprint 2</i>)
US8	As an Admin, I want to automatically flag trades with inconsistent FX rates or missing data so that data quality is maintained. (<i>New for Sprint 2</i>)
US2	As a PM, I want to see current positions and exposures by sector/asset so that I can decide rebalances. (<i>DB design completed in Sprint 1</i>)
US5	As an Admin, I want data integrity checks on trades so that dirty data is flagged. (<i>DB design completed in Sprint 1</i>)

CONCEPTUAL DESIGN

Entity:Fund

Attributes:

- fund_id [PK]
- name
- currency
- inception_date
- benchmark_index

Entity:Portfolio

Attributes:

- portfolio_id [PK]
- fund_id [FK - Fund.fund_id]
- name
- start_date
- strategy

Entity:Investor

Attributes:

- investor_id [PK]
- name
- investor_type (institutional / retail)
- contact_info

Entity:Asset

Attributes:

- asset_id [PK]
- ticker
- name
- asset_type (Equity/Bond/ETF/FX/Derivative)
- currency
- sector

Entity:Price

Attributes:

- price_id [PK]
- asset_id [FK - Asset.asset_id]
- price_date
- close_price

- source

Entity:Trade

Attributes:

- trade_id [PK]
- portfolio_id [FK - Portfolio.portfolio_id]
- asset_id [FK - Asset.asset_id]
- trade_date
- side (BUY/SELL)
- quantity
- price
- currency
- trade_fx_rate
- fees

Entity:Position(derived/aggregated)

Attributes:

- position_id [PK]
- portfolio_id [FK]
- asset_id [FK]
- as_of_date
- quantity
- market_value (derived using Price)

Entity:DailyReturn

Attributes:

- return_id [PK]
- portfolio_id [FK]
- return_date
- daily_return_pct
- nav

Entity:StrategyAllocation

Attributes:

- allocation_id [PK]
- portfolio_id [FK]
- strategy_name
- target_pct

Relationships:

- Fund 1 - Portfolio (Fund has many Portfolios)
- Portfolio 1 - Trade (Portfolio records many trades)

- Asset 1 - Price (Asset has many daily prices)
- Portfolio 1 - DailyReturn (Portfolio has many daily return records)
- Portfolio 1 - Position (Portfolio has many positions)
- Portfolio 1 - StrategyAllocation (Portfolio has multiple allocation targets)

Participation:

- Portfolio has total participation with Fund.
- Trade has partial participation with Portfolio.
- StrategyAllocation has total participation with Portfolio.

LOGICAL DESIGN WITH HIGHEST NORMAL FORM IDENTIFICATION

Table: strategy_allocation(New)

Columns:

- allocation_id SERIAL PRIMARY KEY
- portfolio_id INT REFERENCES portfolio(portfolio_id) ON DELETE CASCADE
- strategy_name VARCHAR(100)
- target_pct NUMERIC(5,2)

Normalization: 3NF

Indexes:

- Non-clustered index on (portfolio_id) for fast lookup of allocations per portfolio.

Table: trade (*updated for data integrity*)

- Add CHECK constraint for valid FX rates: CHECK(trade_fx_rate > 0)
- Add trigger for missing trade price/quantity

Other tables remain as in Sprint 1, with derived calculations for VaR and rolling Sharpe now handled through SQL or stored functions.

SQL QUERIES

```
1. SELECT p.name AS portfolio_name, d.return_date,  
    ROUND((AVG(d.daily_return_pct) OVER (PARTITION BY d.portfolio_id  
ORDER BY d.return_date ROWS BETWEEN 252 PRECEDING AND CURRENT  
ROW))/NULLIF(STDDEV(d.daily_return_pct) OVER (PARTITION BY  
d.portfolio_id ORDER BY d.return_date ROWS BETWEEN 252 PRECEDING AND  
CURRENT ROW),0)),2) AS rolling_12m_sharpe  
FROM daily_return d  
JOIN portfolio p ON d.portfolio_id = p.portfolio_id  
ORDER BY p.portfolio_id, d.return_date;
```

The screenshot displays the MySQL Workbench interface. The SQL Editor window contains the following query:

```
107  
108 SELECT p.name AS portfolio_name, d.return_date,  
109 ROUND((AVG(d.daily_return_pct) OVER (PARTITION BY d.portfolio_id ORDER BY d.return_date  
110 FROM daily_return d
```

The Results Grid shows the output of the query, displaying columns: portfolio_name, return_date, and rolling_12m_sharpe. The data is as follows:

portfolio_name	return_date	rolling_12m_sharpe
Growth Portfolio	2025-10-27	19.6
Growth Portfolio	2025-10-28	0.43
Growth Portfolio	2025-10-29	0.25
Growth Portfolio	2025-10-30	0.32
Growth Portfolio	2025-10-31	0.5
Growth Portfolio	2025-11-01	0.32
Growth Portfolio	2025-11-02	0.26
Growth Portfolio	2025-11-03	0.05
Growth Portfolio	2025-11-04	-0.11

The Output window shows the execution log with the following messages:

- 1 12:41:04 SELECT p.name AS portfolio_name, d.return_date, ROUND((AVG(d.daily_return_pct) OVER (PARTITION BY d...
- 2 12:41:18 SELECT p.name AS portfolio_name, d.return_date, ROUND((AVG(d.daily_return_pct) OVER (PARTITION BY d...
- 3 12:41:33 SELECT p.name AS portfolio_name, d.return_date, ROUND((AVG(d.daily_return_pct) OVER (PARTITION BY d...
- 4 12:41:41 USE HF
- 5 12:41:43 CREATE TABLE fund (fund_id INT PRIMARY KEY, name VARCHAR(200) UNIQUE NOT NULL, current...
- 6 12:41:54 SELECT p.name AS portfolio_name, d.return_date, d.daily_return_pct, d.nav FROM daily_return d JOIN portfolio ...
- 7 12:42:00 SELECT p.name AS portfolio_name, a.ticker, a.name AS asset_name, pos.quantity, pos.market_value FROM po...
- 8 12:42:05 SELECT p.name AS portfolio_name, d.return_date, ROUND((AVG(d.daily_return_pct) OVER (PARTITION BY d...
- 9 12:42:10 SELECT p.name AS portfolio_name, d.return_date, ROUND((AVG(d.daily_return_pct) OVER (PARTITION BY d...

2. WITH ranked_returns AS (

SELECT

d.portfolio_id,

d.daily_return_pct,

ROW_NUMBER() OVER (PARTITION BY d.portfolio_id ORDER BY
d.daily_return_pct) AS rn,

COUNT(*) OVER (PARTITION BY d.portfolio_id) AS total_rows

FROM daily_return d

)

SELECT

p.name AS portfolio_name,

r.daily_return_pct AS var_95

FROM portfolio p

JOIN ranked_returns r

ON p.portfolio_id = r.portfolio_id

WHERE r.rn = FLOOR(0.05 * r.total_rows);

The screenshot displays the MySQL Workbench interface. The SQL Editor window contains the following query:

```
156 p.name AS portfolio_name,
157 r.daily_return_pct AS var_95
158 FROM portfolio p
159 JOIN ranked_returns r
160 ON p.portfolio_id = r.portfolio_id
161 WHERE r.rn = FLOOR(0.05 * r.total_rows);
162
```

The Results window shows the execution output. The first two rows are successful, but subsequent rows contain error messages. The errors are summarized in the table below:

Line	Time	Action	Message	Duration / Fetch
6	12:41:54	SELECT p.name AS portfolio_name, d.return_date, d.daily_return_pct, d.nav FROM daily_return d JOIN portfolio...	30 row(s) returned	0.031 sec / 0.000 sec
7	12:42:00	SELECT p.name AS portfolio_name, a.ticker, a.name AS asset_name, pos.quantity, pos.market_value FROM ...	0 row(s) returned	0.031 sec / 0.000 sec
8	12:42:05	SELECT p.name AS portfolio_name, d.return_date, ROUND(AVG(d.daily_return_pct)) OVER (PARTITION BY d.portfolio...	30 row(s) returned	0.000 sec / 0.000 sec
9	12:42:10	SELECT p.name AS portfolio_name, d.return_date, ROUND(AVG(d.daily_return_pct)) OVER (PARTITION BY d.portfolio...	30 row(s) returned	0.031 sec / 0.000 sec
10	12:48:30	SELECT p.name AS portfolio_name, PERCENTILE_CONT(0.05) WITHIN GROUP (ORDER BY d.daily_return_pct) AS p...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se...	0.000 sec
11	12:48:46	SELECT p.name AS portfolio_name, PERCENTILE_CONT(0.05) WITHIN GROUP (ORDER BY d.daily_return_pct) AS p...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se...	0.000 sec
12	12:49:08	SELECT p.name AS portfolio_name, sa.strategy_name, sa.target_pct, ROUND(SUM(pos.market_value) / ...	Error Code: 1055. Expression #1 of PARTITION BY or ORDER BY clause of window 'unnamed window' is n...	0.016 sec
13	12:51:06	SELECT p.name AS portfolio_name, PERCENTILE_CONT(0.05) WITHIN GROUP (ORDER BY d.daily_return_pct) AS p...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se...	0.000 sec
14	12:51:53	SELECT p.name AS portfolio_name, (SELECT d2.daily_return_pct FROM daily_return d2 WHERE d2.portfolio_id = d.portfolio_id) AS d...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se...	0.000 sec
15	12:52:36	WITH ranked_returns AS (SELECT d.portfolio_id, d.daily_return_pct, ROW_NUMBER() OVER (PARTITION BY d.portfolio_id ORDER BY d.daily_return_pct) AS rn, COUNT(*) OVER (PARTITION BY d.portfolio_id) AS total_rows FROM daily_return d) SELECT p.name AS portfolio_name, r.daily_return_pct AS var_95 FROM portfolio p JOIN ranked_returns r ON p.portfolio_id = r.portfolio_id WHERE r.rn = FLOOR(0.05 * r.total_rows);	0 row(s) returned	0.032 sec / 0.000 sec

3. SELECT

```
p.name AS portfolio_name,  
sa.strategy_name,  
sa.target_pct,  
ROUND(SUM(pos.market_value) / total_portfolio_value * 100, 2) AS  
current_pct  
FROM position pos  
JOIN portfolio p ON pos.portfolio_id = p.portfolio_id  
JOIN strategy_allocation sa ON p.portfolio_id = sa.portfolio_id  
JOIN (  
    -- Calculate total market value per portfolio  
    SELECT portfolio_id, SUM(market_value) AS total_portfolio_value  
    FROM position  
    GROUP BY portfolio_id  
) AS totals ON pos.portfolio_id = totals.portfolio_id  
GROUP BY p.name, sa.strategy_name, sa.target_pct, total_portfolio_value;
```

The screenshot displays the MySQL Workbench interface. The main window shows a SQL query in the 'SQL File 2' tab. The query is a complex SELECT statement that calculates the current percentage of market value for each portfolio, based on the total market value of the portfolio. The query is as follows:

```
-- Calculate total market value per portfolio  
SELECT portfolio_id, SUM(market_value) AS total_portfolio_value  
FROM position  
GROUP BY portfolio_id  
) AS totals ON pos.portfolio_id = totals.portfolio_id  
GROUP BY p.name, sa.strategy_name, sa.target_pct, total_portfolio_value;
```

The 'Result Grid' shows the results of the query, with columns: portfolio_name, strategy_name, target_pct, current_pct. The results are as follows:

portfolio_name	strategy_name	target_pct	current_pct
Growth Portfolio	Growth	0.60	100.00
Dividend Portfolio	Dividend	0.50	100.00
Bond Portfolio	Bond	0.80	100.00

The 'Output' tab shows the execution log, including the query text and the results of the query. The log indicates that the query was executed successfully, with 3 rows returned. The 'Message' column shows the error messages for each row, which are related to the 'PARTITION BY' clause in the query.

VIEWS AND STORED PROGRAMS

View: vw_portfolio_exposure

- Goal: Show current positions and exposures by asset type/sector for PM/Analyst review.

Stored Procedure: sp_rebalance_simulation

- Parameters: IN portfolio_id INT, IN target_date DATE
- Goal: Simulate portfolio rebalancing to match target allocations, adjusting positions and producing a summary report.

Stored Function: fn_portfolio_var

- Parameters: IN portfolio_id INT, IN confidence_level DECIMAL(4,2)
- Goal: Calculate historical VaR for a portfolio at the specified confidence level; returns numeric value.

Trigger: trg_trade_integrity (*AFTER INSERT/UPDATE on trade*)

- Goal: Automatically validate trade data (price, quantity, FX rate) and flag anomalies in a log table.