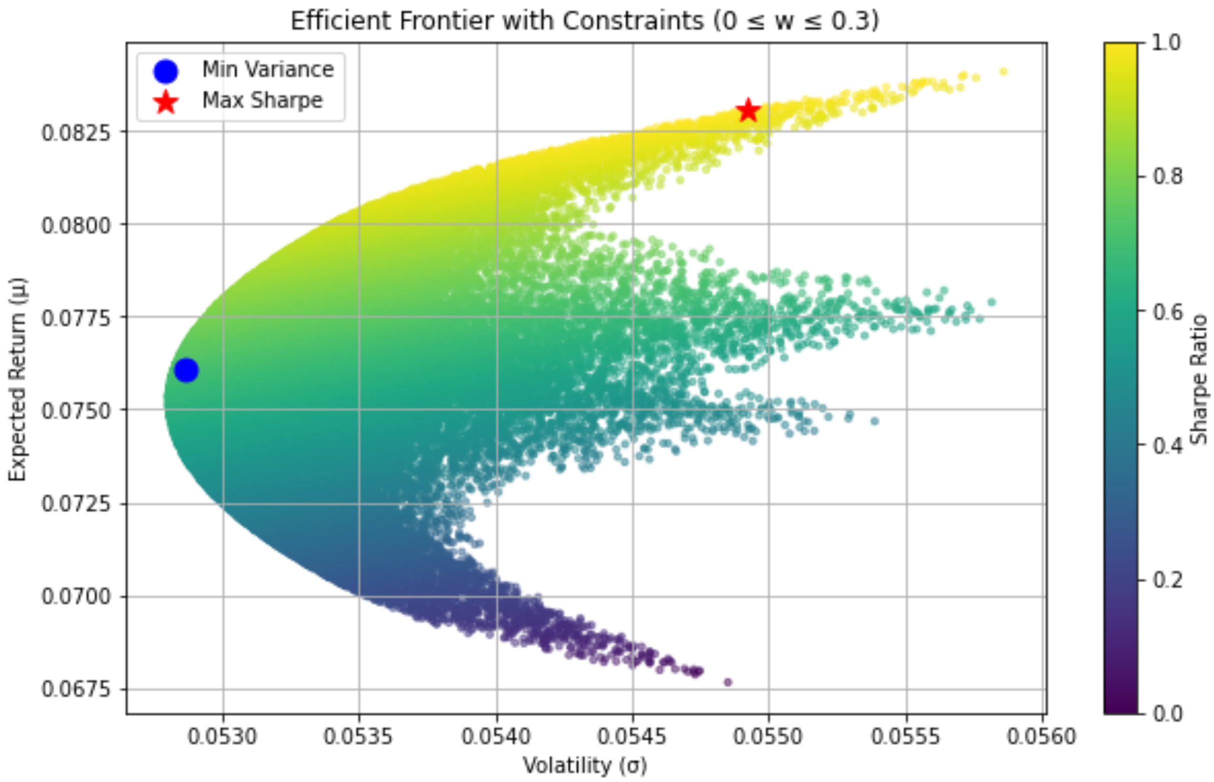
In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize

# === 1. Load data ===
df = pd.read_csv("ETF_data_2010_2024.csv", index_col=0, parse_dates=True)
returns = np.log(df / df.shift(1)).dropna()

# Annualized returns and covariance
mu = returns.mean() * 12
cov = returns.cov() * 12
rf = 0.01  # risk-free rate (1%)

# === 2. Optimization functions ===
def portfolio_performance(weights, mu, cov):
    ret = np.dot(weights, mu)
    vol = np.sqrt(np.dot(weights.T, np.dot(cov, weights)))
    return ret, vol

def min_variance(mu, cov, bounds):
    n = len(mu)
    init = np.ones(n) / n
    constraints = ({'type': 'eq', 'fun': lambda w: np.sum(w) - 1})
    result = minimize(lambda w: portfolio_performance(w, mu, cov)[1]**2,
                      method='SLSQP', bounds=bounds, constraints=constrai
    return result.x

def max_sharpe(mu, cov, rf, bounds):
    n = len(mu)
    init = np.ones(n) / n
    constraints = ({'type': 'eq', 'fun': lambda w: np.sum(w) - 1})
    def neg_sharpe(w):
        ret, vol = portfolio_performance(w, mu, cov)
        return -(ret - rf) / vol
    result = minimize(neg_sharpe, init, method='SLSQP', bounds=bounds, co
    return result.x

# === 3. Weight constraints ===
bounds = tuple((0, 0.3) for _ in range(len(mu)))

# === 4. Optimal portfolios ===
w_minvar = min_variance(mu, cov, bounds)
w_sharpe = max_sharpe(mu, cov, rf, bounds)

# === 5. Efficient frontier simulation ===
n_portfolios = 50000
results = np.zeros((3, n_portfolios))
for i in range(n_portfolios):
    w = np.random.dirichlet(np.ones(len(mu)), size=1).flatten()
    # Ensure weights respect the constraints (0.3 max per asset)
    while any(w > 0.3):
        w = np.random.dirichlet(np.ones(len(mu)), size=1).flatten()
    ret, vol = portfolio_performance(w, mu, cov)
    sharpe = (ret - rf) / vol
```

```
    results[0, i] = vol
    results[1, i] = ret
    results[2, i] = sharpe

# === 6. Plot the efficient frontier ===
plt.figure(figsize=(10,6))
plt.scatter(results[0,:], results[1,:], c=results[2,:], cmap='viridis', s
minvar_ret, minvar_vol = portfolio_performance(w_minvar, mu, cov)
sharpe_ret, sharpe_vol = portfolio_performance(w_sharpe, mu, cov)
plt.scatter(minvar_vol, minvar_ret, c='blue', marker='o', s=120, label='M
plt.scatter(sharpe_vol, sharpe_ret, c='red', marker='*', s=150, label='Ma
plt.xlabel('Volatility (σ)')
plt.ylabel('Expected Return (μ)')
plt.title('Efficient Frontier with Constraints (0 ≤ w ≤ 0.3)')
plt.colorbar(label='Sharpe Ratio')
plt.legend()
plt.grid(True)
plt.show()

# === 7. Print optimal weights ===
print("Min Variance Portfolio Weights:\n", pd.Series(w_minvar, index=mu.i
print("\nMax Sharpe Portfolio Weights:\n", pd.Series(w_sharpe, index=mu.i
```



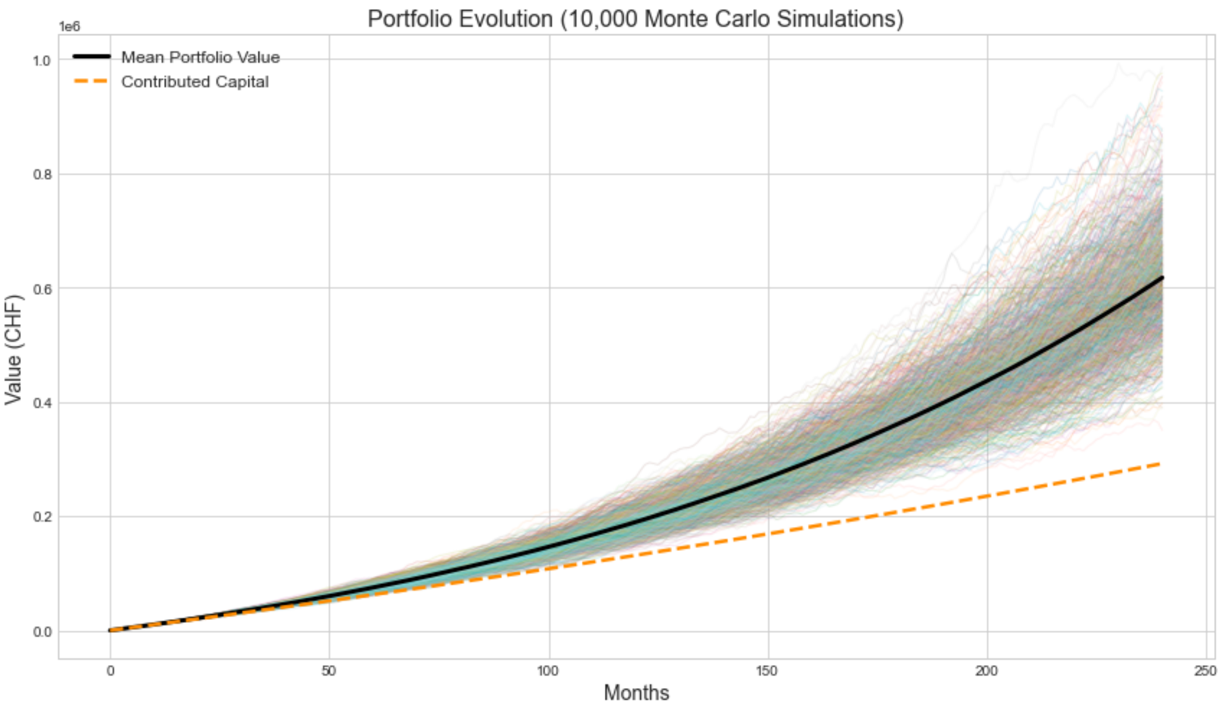
```
Min Variance Portfolio Weights:
 VWRA    0.25
AGGG    0.25
GLD     0.25
REET    0.25
dtype: float64

Max Sharpe Portfolio Weights:
 VWRA    0.135111
AGGG    0.271724
GLD     0.300000
REET    0.293165
dtype: float64
```

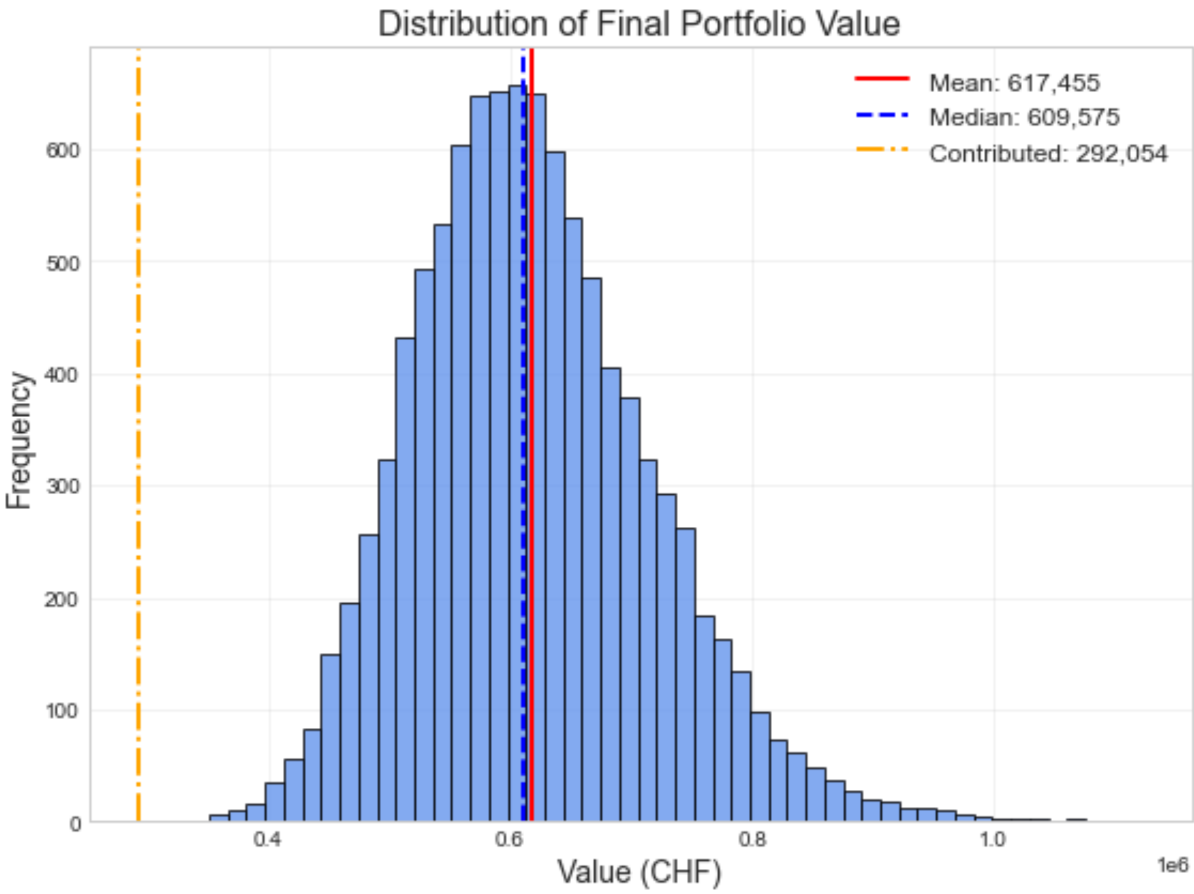
In [3]:

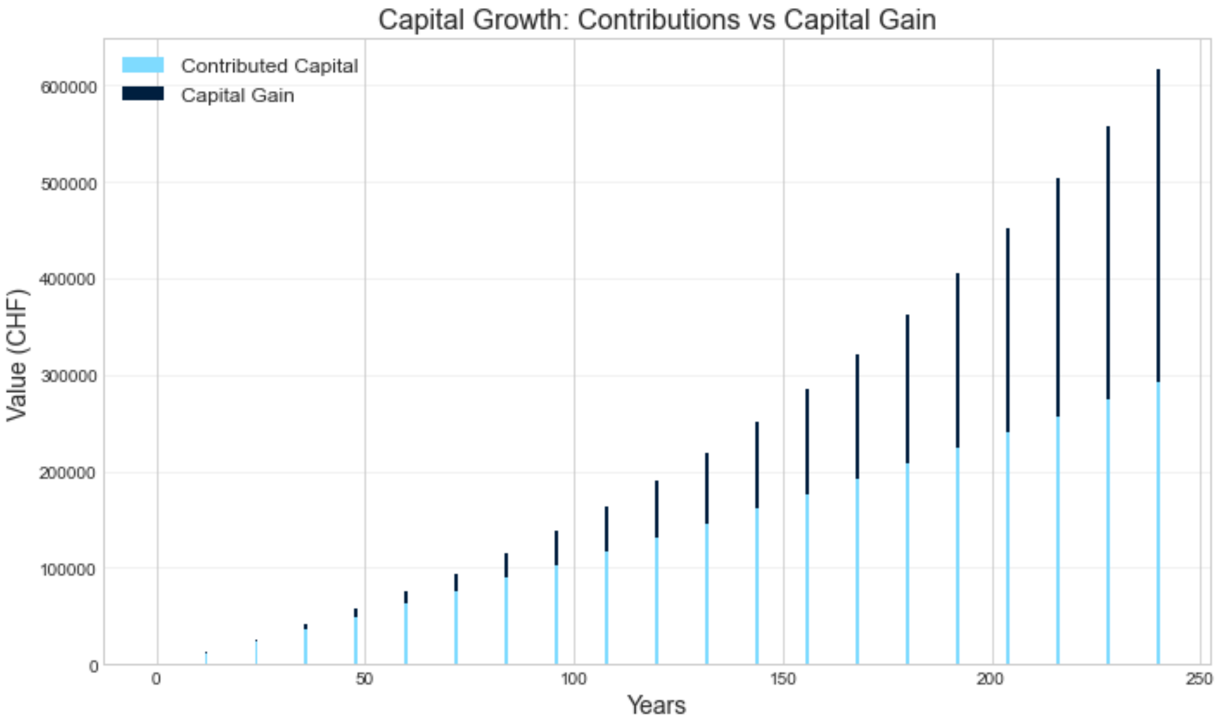
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# === SIMULATION PARAMETERS ===
np.random.seed(42)
years = 20
months = years * 12
n_scenarios = 10000
C0 = 1000           # initial monthly contribution
delta = 0.02        # annual growth rate of contributions
cost_rate = 0.001   # transaction costs

# === HISTORICAL DATA ===
df = pd.read_csv("ETF_data_2010_2024.csv", index_col=0, parse_dates=True)
returns = np.log(df / df.shift(1)).dropna()
mu = returns.mean().values * 12
cov = returns.cov().values * 12
weights = np.array([0.135111, 0.271724, 0.3, 0.293165])  # optimized weig
n_assets = len(mu)

# === SIMULATION FUNCTION ===
def simulate_portfolio(mu, cov, weights, C0, delta, months, n_scenarios,
    chol = np.linalg.cholesky(cov / 12)
    portfolio_values = np.zeros((months + 1, n_scenarios))
    contributions = np.zeros(months + 1)
    for t in range(1, months + 1):
        Ct = C0 * (1 + delta) ** (t // 12)
        contributions[t] = contributions[t-1] + Ct
        rand = chol @ np.random.randn(n_assets, n_scenarios)
        asset_returns = mu.reshape(-1,1)/12 + rand
        portfolio_monthly = np.dot(weights, asset_returns)
        portfolio_values[t] = (portfolio_values[t-1] + Ct) * (1 + portfol
    return portfolio_values, contributions

# === RUN SIMULATION ===
portfolio_values, contributions = simulate_portfolio(mu, cov, weights, C0
final_values = portfolio_values[-1]
median = np.percentile(final_values, 50)
mean = np.mean(final_values)
p5, p25, p75, p95 = np.percentile(final_values, [5, 25, 75, 95])
invested_capital = contributions[-1]
success_rate = np.mean(final_values > invested_capital)
```

```
print(f"Mean: {mean:,.0f} CHF | Median: {median:,.0f} CHF")
print(f"5th–95th percentile: {p5:,.0f} – {p95:,.0f} CHF")
print(f"Total contributed capital: {invested_capital:,.0f} CHF")
print(f"Success rate: {success_rate*100:.2f}%")

# === GRAPH STYLE ===
plt.style.use('seaborn-whitegrid')

# === 1. PORTFOLIO EVOLUTION (SPAGHETTI PLOT) ===
plt.figure(figsize=(12,7))
colors = plt.cm.tab20(np.linspace(0, 1, 1000))  # vibrant multi-color lin
for i in range(1000):
    plt.plot(portfolio_values[:,i], color=colors[i], alpha=0.15, linewidt
# Highlight the mean trajectory
plt.plot(np.mean(portfolio_values, axis=1), color='black', linewidth=3, l
plt.plot(contributions, color='darkorange', linestyle='--', linewidth=2.5
plt.title('Portfolio Evolution (10,000 Monte Carlo Simulations)', fontsiz
plt.xlabel('Months', fontsize=14)
plt.ylabel('Value (CHF)', fontsize=14)
plt.legend(fontsize=12)
plt.tight_layout()
plt.show()

# === 2. HISTOGRAM OF FINAL CAPITAL DISTRIBUTION ===
plt.figure(figsize=(8,6))
plt.hist(final_values, bins=50, color='cornflowerblue', edgecolor='black'
plt.axvline(mean, color='red', linestyle='-', linewidth=2, label=f"Mean:
plt.axvline(median, color='blue', linestyle='--', linewidth=2, label=f"Me
plt.axvline(invested_capital, color='orange', linestyle='-.', linewidth=2
plt.title('Distribution of Final Portfolio Value', fontsize=16)
plt.xlabel('Value (CHF)', fontsize=14)
plt.ylabel('Frequency', fontsize=14)
plt.legend(fontsize=12)
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()

# === 3. STACKED BAR: CONTRIBUTED CAPITAL VS CAPITAL GAINS ===
plt.figure(figsize=(10,6))
mean_portfolio = np.mean(portfolio_values, axis=1)
gain = mean_portfolio - contributions
years_axis = np.arange(0, months+1, 12)
plt.bar(years_axis, contributions[years_axis], color='#7FDBFF', label='Co
plt.bar(years_axis, gain[years_axis], bottom=contributions[years_axis], c
plt.title('Capital Growth: Contributions vs Capital Gain', fontsize=16)
plt.xlabel('Years', fontsize=14)
plt.ylabel('Value (CHF)', fontsize=14)
plt.legend(fontsize=12)
plt.grid(axis='y', alpha=0.3)
plt.tight_layout()
plt.show()
```

```
Mean: 617,455 CHF | Median: 609,575 CHF
5th–95th percentile: 470,583 – 788,808 CHF
Total contributed capital: 292,054 CHF
Success rate: 100.00%
```


Portfolio Evolution (10,000 Monte Carlo Simulations)
1e6
Mean Portfolio Value
Contributed Capital
Value (CHF)
Months
0.0
0.2
0.4
0.6
0.8
1.0
0
50
100
150
200
250


Distribution of Final Portfolio Value
Mean: 617,455
Median: 609,575
Contributed: 292,054
Frequency
Value (CHF)
1e6
0
100
200
300
400
500
600
0.4
0.6
0.8
1.0

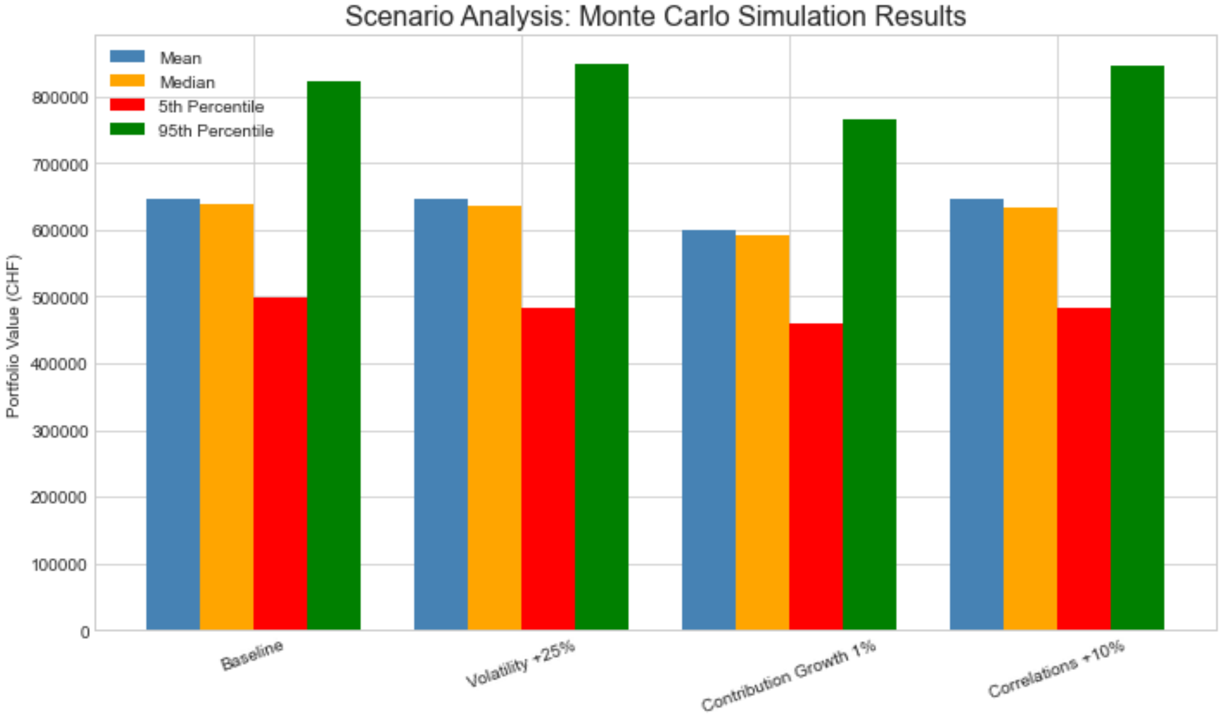


In [4]:
```
import numpy as np
import pandas as pd
from scipy.optimize import minimize
import matplotlib.pyplot as plt

# === GENERAL PARAMETERS ===
np.random.seed(42)
years = 20
months = years * 12
n_sim = 10000
C0 = 1000
growth_rate_base = 0.02
cost_rate = 0.001

# === LOAD HISTORICAL DATA ===
df = pd.read_csv("ETF_data_2010_2024.csv", index_col=0, parse_dates=True)
returns = np.log(df / df.shift(1)).dropna()
mean_returns = returns.mean() * 12
cov_matrix = returns.cov() * 12

# === PORTFOLIO OPTIMIZATION (Minimum Variance) ===
n = returns.shape[1]
bounds = tuple((0, 0.3) for _ in range(n))
constraints = ({'type': 'eq', 'fun': lambda w: np.sum(w) - 1})
w0 = np.ones(n) / n
```

```
def port_var(w): return w.T @ cov_matrix @ w
res = minimize(port_var, w0, method='SLSQP', bounds=bounds, constraints=c
weights = res.x

# === MONTE CARLO SIMULATION FUNCTION ===
def monte_carlo_sim(mu, cov, weights, C0, growth_rate, months, n_sim, cos
    simulated = np.random.multivariate_normal(mu/12, cov/12, (months, n_s
    port_rets = simulated @ weights
    V = np.zeros((months, n_sim))
    for sim in range(n_sim):
        value = 0
        for t in range(months):
            Ct = C0 * ((1 + growth_rate) ** (t // 12))
            value = (value + Ct) * (1 + port_rets[t, sim])
            if t % 12 == 0 and t > 0:
                value *= (1 - cost_rate)  # annual rebalancing costs
            V[t, sim] = value
    return V[-1]

# === SCENARIOS ===
# 1. Baseline
final_base = monte_carlo_sim(mean_returns, cov_matrix, weights, C0, growt

# 2. Volatility +25%
cov_vol_up = cov_matrix * 1.25
final_vol_up = monte_carlo_sim(mean_returns, cov_vol_up, weights, C0, gro

# 3. Reduced contribution growth (1%)
final_contrib_low = monte_carlo_sim(mean_returns, cov_matrix, weights, C0

# 4. Higher correlations (+10%)
corr_matrix = cov_matrix / np.outer(np.sqrt(np.diag(cov_matrix)), np.sqrt
corr_worse = corr_matrix + 0.1 * (1 - corr_matrix)  # increase correlatio
cov_corr_up = np.outer(np.sqrt(np.diag(cov_matrix)), np.sqrt(np.diag(cov_
final_corr_up = monte_carlo_sim(mean_returns, cov_corr_up, weights, C0, g

# === FUNCTION FOR SCENARIO STATISTICS ===
def scenario_stats(values):
    return {
        "Mean": np.mean(values),
        "Median": np.percentile(values, 50),
        "P5": np.percentile(values, 5),
        "P95": np.percentile(values, 95)
    }

# === CALCULATE RESULTS FOR ALL SCENARIOS ===
results = {
    "Baseline": scenario_stats(final_base),
    "Volatility +25%": scenario_stats(final_vol_up),
    "Contribution Growth 1%": scenario_stats(final_contrib_low),
    "Correlations +10%": scenario_stats(final_corr_up)
}

df_results = pd.DataFrame(results).T
df_results = df_results.round(2)

print("\n=== Scenario Analysis (Final Portfolio Values in CHF) ===")
```

```
print(df_results)

# === PLOT COMPARATIVE BAR CHART ===
plt.figure(figsize=(10,6))
x = np.arange(len(df_results))
bar_width = 0.2

plt.bar(x - bar_width*1.5, df_results['Mean'], width=bar_width, label='Me
plt.bar(x - bar_width*0.5, df_results['Median'], width=bar_width, label='
plt.bar(x + bar_width*0.5, df_results['P5'], width=bar_width, label='5th 
plt.bar(x + bar_width*1.5, df_results['P95'], width=bar_width, label='95t

plt.xticks(x, df_results.index, rotation=20)
plt.ylabel('Portfolio Value (CHF)')
plt.title('Scenario Analysis: Monte Carlo Simulation Results', fontsize=1
plt.legend()
plt.tight_layout()
plt.show()
```

```
=== Scenario Analysis (Final Portfolio Values in CHF) ===
                             Mean     Median         P5        P95
Baseline                646788.69  639502.48  497402.54  823171.40
Volatility +25%         646688.12  635810.31  482740.01  848686.38
Contribution Growth 1%  599180.77  591329.78  459581.35  766412.65
Correlations +10%       645367.79  633853.08  481609.37  846530.35
```



In [ ]: