Financial Data Analysis

```
"""
Financial Data Analysis Script
Analyzing cryptocurrency market data with technical indicators

This script demonstrates data ingestion from CSV, data wrangling,
custom analysis functions, and visualization capabilities.
"""

# Standard library imports
import warnings
from datetime import datetime
from typing import Dict

# Third-party libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Suppress warnings for cleaner output
warnings.filterwarnings('ignore')

# Try to use seaborn for better plots, but it's optional
try:
    import seaborn as sns
    sns.set_style("whitegrid")
    USE_SEABORN = True
except ImportError:
    USE_SEABORN = False
    print("Note: seaborn not available, using default matplotlib style")

# Configure plot settings
plt.rcParams['figure.figsize'] = (12, 6)
plt.style.use('default')


def ingest_data_from_csv(filepath: str) -> pd.DataFrame:
    """
    Load and process CSV data file.
    Handles different data types and cleans the dataset.
    """
    # Read CSV file - using Date column as index for time series
    df = pd.read_csv(filepath, index_col="Date", parse_dates=True)

    # Need to handle different data types properly
    # Price and volume columns should be numeric (float)
    numeric_cols = []
    for col in df.columns:
        if 'Close' in col or 'Volume' in col or 'High' in col or 'Low' in col:
            numeric_cols.append(col)

    # Convert to float, handling any non-numeric values
    for col in numeric_cols:
        df[col] = pd.to_numeric(df[col], errors='coerce').astype(float)

    # Fix timezone issues if present (some APIs include timezone info)
    if hasattr(df.index, 'tz') and df.index.tz is not None:
        df.index = df.index.tz_localize(None)

    # Clean up: remove duplicate dates and fill missing values
    df = df[~df.index.duplicated(keep='last')]
    df = df.sort_index()
    df = df.ffill()  # Forward fill for missing prices

    # Print summary
    print(f"✓ Loaded {len(df)} rows from {filepath}")
```

```
    }

    # Calculate return statistics if available
    if 'Returns' in df.columns:
        returns = df['Returns'].dropna()
        if len(returns) > 0:
            analysis['returns_stats'] = {
                'mean_daily_return': float(returns.mean() * 100),
                'volatility': float(returns.std() * 100),
                'total_return': float(df['Cumulative_Returns'].iloc[-1] * 100) if 'Cumulative_Returns' in
df.columns else None,
            }

    # RSI analysis - check for overbought/oversold conditions
    if 'RSI' in df.columns:
        rsi = df['RSI'].dropna()
        if len(rsi) > 0:
            current_rsi = rsi.iloc[-1]
            analysis['indicators']['RSI'] = {
                'current': float(current_rsi),
                'mean': float(rsi.mean()),
                'oversold_signal': current_rsi < 30,  # RSI < 30 suggests oversold
                'overbought_signal': current_rsi > 70,  # RSI > 70 suggests overbought
            }

    # MACD analysis - look for bullish/bearish crossovers
    if 'MACD' in df.columns and 'MACD_Signal' in df.columns:
        if len(df) > 0:
            macd = df['MACD'].iloc[-1]
            signal = df['MACD_Signal'].iloc[-1]
            analysis['indicators']['MACD'] = {
                'current': float(macd) if pd.notna(macd) else None,
                'signal': float(signal) if pd.notna(signal) else None,
                'bullish_cross': macd > signal if (pd.notna(macd) and pd.notna(signal)) else False,
            }

    return analysis


def visualize_data(df: pd.DataFrame, price_col: str = "BTCUSDT_Close", save_path: str = "analysis_plot.png"):
    """
    Create visualization plots showing price, RSI, and MACD indicators.
    Saves the plot to a file.
    """
    # Create 3 subplots stacked vertically
    fig, axes = plt.subplots(3, 1, figsize=(14, 10))
    fig.suptitle('Financial Data Analysis Dashboard', fontsize=16, fontweight='bold')

    # Top plot: Price with moving averages
    ax1 = axes[0]
    ax1.plot(df.index, df[price_col], label='Price', linewidth=2, color='#2E86AB')

    # Add moving averages if they exist
    if 'MA_20' in df.columns:
        ax1.plot(df.index, df['MA_20'], label='MA 20', alpha=0.7, color='orange', linestyle='--')
    if 'MA_50' in df.columns:
        ax1.plot(df.index, df['MA_50'], label='MA 50', alpha=0.7, color='red', linestyle='--')

    # Add Bollinger Bands as shaded area
    if 'BB_Upper' in df.columns and 'BB_Lower' in df.columns:
        ax1.fill_between(df.index, df['BB_Upper'], df['BB_Lower'],
                         alpha=0.2, color='gray', label='Bollinger Bands')

    ax1.set_title('Price Chart with Technical Indicators', fontweight='bold')
    ax1.set_ylabel('Price (USDT)', fontweight='bold')
```

PROGRAM OUTPUT

```
Note: seaborn not available, using default matplotlib style
============================================================
Financial Data Analysis Script
============================================================

[Step 1] Creating sample CSV data...
  Created sample data: sample_financial_data.csv
✓ Loaded 366 rows from sample_financial_data.csv
  Date range: 2024-01-01 to 2024-12-31
  Columns: BTCUSDT_Close, BTCUSDT_High, BTCUSDT_Low...

[Step 2] Wrangling data...
  ✓ Data cleaned: 366 rows, 6 columns

[Step 3] Calculating technical indicators...
  ✓ Indicators calculated: RSI, MACD, Moving Averages, Bollinger Bands, ATR

[Step 4] Analyzing market trends...
  Current Price: $49,961.38
  Price Range: 37,411.73 — 54,615.33
  Mean Daily Return: 0.01%
  Volatility: 1.90%
  Total Return: -1.06%

  RSI Analysis:
    Current RSI: 60.19
    Oversold Signal: False
    Overbought Signal: False

  MACD Analysis:
    MACD: -128.3115
    Signal: -421.4639
    Bullish Cross: True

[Step 5] Creating visualizations...
✓ Visualization saved to financial_analysis.png

============================================================
Analysis Complete!
============================================================
Final dataset: 366 rows × 18 columns
Generated files: sample_financial_data.csv, financial_analysis.png
```

Financial Data Analysis Dashboard