New Features

1. Public Holiday Detection

The system now automatically detects public holidays for different regions:

- US holidays for USD-related assets (SPY, US100, etc.)
- UK holidays for GBP pairs
- ECB holidays for EUR pairs
- Japanese holidays for JPY pairs
- Swiss holidays for CHF pairs
- Australian holidays for AUD pairs

Holiday gaps are no longer flagged as data issues in the validator.

2. News Event Downtime Tracking

The collector fetches high-impact economic events daily from TradingEconomics API and logs downtime windows (1 hour before to 1 hour after) in:

```
ohlc_data/{symbol}/trading_downtime.csv
```

3. Email Alerts for Stale Data

The `alert_check.py` script monitors for data that hasn't been updated in 7+ days and sends email alerts.

Email Alert Setup

1. **Configure email settings** in `alert_check.py`:

   ```python
   EMAIL_CONFIG = {
       "EMAIL_FROM": "your_email@gmail.com",
       "EMAIL_TO": "your_email@gmail.com",
       "SMTP_SERVER": "smtp.gmail.com",
       "SMTP_PORT": 587,
       "EMAIL_PASSWORD": "your_app_password"
   }
   ```

2. **For Gmail users**:
   - Enable 2-factor authentication
   - Generate an app-specific password: https://myaccount.google.com/apppasswords

- Use the app password instead of your regular password

3. **Schedule the alert script** to run daily: Using cron (Linux/Mac):

```bash
# Edit crontab
crontab -e

# Add this line to run daily at 9 AM
0 9 * * * /path/to/venv/bin/python /path/to/alert_check.py
```

Using systemd timer (Linux): Create `/etc/systemd/system/ohlcv-alert.service`:

```ini
[Unit]
Description=OHLCV Stale Data Alert Check

[Service]
Type=oneshot
User=your-username
WorkingDirectory=/path/to/script/directory
ExecStart=/path/to/venv/bin/python /path/to/alert_check.py
```

Create `/etc/systemd/system/ohlcv-alert.timer`:

```ini
[Unit]
Description=Run OHLCV Alert Check daily

[Timer]
OnCalendar=daily
Persistent=true

[Install]
WantedBy=timers.target
```

Enable and start:

```bash
sudo systemctl daemon-reload
sudo systemctl enable ohlcv-alert.timer
sudo systemctl start ohlcv-alert.timer
```# OHLCV Data Collector Setup Instructions

Requirements

Create a `requirements.txt` file with the following content:

```
yfinance>=0.2.28
pandas>=2.0.0
python-dateutil>=2.8.2
pytz>=2023.3
tabulate>=0.9.0
holidays>=0.35
aiohttp>=3.9.0
```

Installation Steps

1. **Create a virtual environment** (recommended):

```bash
python3 -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
```

2. **Install dependencies**:

```bash
pip install -r requirements.txt
```

3. **Configure settings** (optional):

```bash
# Copy config template and edit with your settings
cp config_template.py config.py
# Edit config.py with your email settings
```

4. **Make scripts executable** (Linux/Mac):

```bash
chmod +x ohlcv_collector.py
chmod +x monitor.py
chmod +x validator.py
chmod +x alert_check.py
chmod +x test_holidays.py
chmod +x check_symbols.py
chmod +x migrate_data.py
chmod +x reset_collector.py
```

Running the Script

First Time Setup:

If you have existing data from the old version:

```bash
# Check which symbols are valid
python check_symbols.py

# Clean up duplicate/invalid symbols
python migrate_data.py

# Or for a fresh start
python reset_collector.py
```

Development/Testing:

```bash
python ohlcv_collector.py
```

Production Deployment:

Option 1: Using systemd (Linux)

Create `/etc/systemd/system/ohlcv-collector.service`:

```ini
[Unit]
Description=OHLCV Data Collector
After=network.target

[Service]
Type=simple
User=your-username
WorkingDirectory=/path/to/script/directory
Environment="PATH=/path/to/venv/bin"
ExecStart=/path/to/venv/bin/python /path/to/ohlcv_collector.py
Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
```

Then:

bash

```
sudo systemctl daemon-reload
sudo systemctl enable ohlcv-collector
sudo systemctl start ohlcv-collector
```

Option 2: Using screen/tmux

bash

```
screen -S ohlcv-collector
python ohlcv_collector.py
# Press Ctrl+A, D to detach
```

Option 3: Using Docker

Create a `Dockerfile`:

dockerfile

```
FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY ohlcv_collector.py .

CMD ["python", "-u", "ohlcv_collector.py"]
```

Build and run:

bash

```
docker build -t ohlcv-collector .
docker run -d --name ohlcv-collector -v $(pwd)/ohlc_data:/app/ohlc_data ohlcv-collector
```

Directory Structure

After running, the script creates:

```
.
├── ohlcv_collector.py     # Main collector (enhanced)
├── monitor.py             # Monitor script (enhanced)
├── validator.py           # Data validator (enhanced)
├── alert_check.py         # Email alert script
├── test_holidays.py       # Holiday detection tester
├── check_symbols.py       # Symbol validity checker
├── migrate_data.py        # Data migration tool
├── reset_collector.py     # Reset/cleanup tool
├── config_template.py     # Configuration template
├── config.py             # Your configuration (create from template)
├── ohlc_data/
│   ├── eurusd/
│   │   ├── 1m.csv
│   │   ├── 5m.csv
│   │   ├── ...
│   │   └── trading_downtime.csv  # News event downtimes
│   ├── btcusd/
│   │   └── ...
│   └── ...
├── status.json            # Tracks last update times
├── collector.log          # Main log file
├── errors.log            # Error log file
└── alert_check.log       # Alert script log
```

Features

1. **Automatic Symbol Validation**: The script validates which Yahoo Finance symbols are available and only tracks valid ones.
2. **Duplicate Prevention**: Checks existing data before appending to avoid duplicates.
3. **Timezone Handling**: All timestamps are converted to GMT for consistency.
4. **4-Hour Timeframe**: Since Yahoo Finance doesn't directly support 4h intervals, the script fetches 1h data and resamples it.
5. **Error Recovery**: The script logs errors and continues running. Failed updates are retried on the next cycle.
6. **Status Tracking**: The `status.json` file tracks the last successful update for each symbol/timeframe combination.

Monitoring

- Check `collector.log` for general operation logs
- Check `errors.log` for any errors

- Check `alert_check.log` for email alert history
- Monitor `status.json` to see last update times
- Use `tail -f collector.log` to watch real-time logs

Using the Monitor Script

Run the included monitor script to see detailed statistics:

```bash
python monitor.py
```

This displays:

- Data collection status for each symbol/timeframe
- Row counts and date ranges
- Last update times
- Latest news downtime windows
- Recent errors (if any)

Data Validation

Run the validator script to check data integrity:

```bash
python validator.py
```

This checks for:

- Missing or duplicate data
- Invalid OHLC values (High < Low, etc.)
- Time gaps in the data (holiday-aware)
- Data freshness
- File corruption

Testing Holiday Detection

Test that holiday detection is working:

```bash
python test_holidays.py
```

This shows:

- Upcoming holidays for each currency
- Example holiday checks for symbols

Stopping the Script

- **If running directly**: Press Ctrl+C
- **If using systemd**: `sudo systemctl stop ohlcv-collector`
- **If using screen**: Reattach with `screen -r ohlcv-collector` and press Ctrl+C
- **If using Docker**: `docker stop ohlcv-collector`

Notes

1. Yahoo Finance has rate limits. The script spaces out requests appropriately.
2. Some forex pairs might not be available on Yahoo Finance. The script automatically detects and skips unavailable symbols.
3. The script uses async operations to handle multiple timeframes efficiently without blocking.
4. Data is saved incrementally, so you can stop and restart the script without losing data.

Troubleshooting

Common Issues

1. **Many symbols showing 0 rows:**
   - Run `python check_symbols.py` to verify which Yahoo Finance symbols are valid
   - Run `python migrate_data.py` to clean up duplicate/invalid symbol directories
   - The collector now uses correct Yahoo Finance symbols for all assets
2. **"Never" shown in Updated column:**
   - This happens when `status.json` doesn't exist or wasn't updated
   - Delete `status.json` and restart the collector for a fresh start
   - The status file is only created after successful data collection
3. **Email alerts not sending:**
   - Verify email settings in `config.py`
   - For Gmail: Ensure you're using an app-specific password, not your regular password
   - Check `alert_check.log` for error messages
   - Test SMTP connection manually
4. **TradingEconomics API errors:**
   - The provided API key has rate limits

- Consider getting your own key at https://tradingeconomics.com/api
- Check `errors.log` for API response codes

5. **Holiday detection not working:**
   - Run `python test_holidays.py` to verify
   - Ensure `holidays` package is installed correctly
   - Some markets may trade on certain holidays
6. **Missing trading_downtime.csv files:**
   - Files are only created when high-impact events are detected
   - Check if the API is returning events properly
   - Not all symbols may have relevant news events
7. **Duplicate forex pairs (e.g., GBPUSD and USDGBP):**
   - Yahoo Finance typically only has one direction for each pair
   - Run the migration script to clean up duplicates
   - The updated collector only creates valid pairs

Summary of Enhancements

The OHLCV collection system has been enhanced with:

1. **Holiday-Aware Gap Detection**
   - Automatically detects public holidays for each currency
   - Validator no longer flags holiday gaps as issues
   - Reduces false positives in data integrity checks
2. **News Event Tracking**
   - Fetches high-impact economic events daily from TradingEconomics
   - Creates downtime windows (±1 hour around events)
   - Stores in `trading_downtime.csv` for each symbol
   - Useful for backtesting strategies that avoid news periods
3. **Automated Monitoring**
   - Email alerts when data hasn't updated for 7+ days
   - Daily health check via `alert_check.py`
   - Proactive notification of collection issues
4. **Enhanced Monitoring**
   - Monitor script shows latest news downtime per symbol
   - Better visibility into data collection status

All enhancements maintain the existing async architecture and GMT timestamp format, ensuring backward compatibility with your backtesting systems.