Commodity Price Data Pipeline

A robust, modular data pipeline for collecting, storing, and verifying USDA AMS commodity price data.

Round Lakes Commodities

Overview

This pipeline automates the collection of commodity price data from the USDA My Market News (MMN) API, stores it in a database (SQLite for development, MySQL/PostgreSQL for production), and includes verification to ensure data integrity.

Key Features

- **Modular Architecture**: Separate agents for collection, storage, and verification
- **Async Data Collection**: Concurrent API requests for fast data fetching
- **Flexible Database Support**: SQLite, MySQL, and PostgreSQL
- **Data Verification**: Automated integrity checks after each load
- **Excel-Based Configuration**: Easy report management via spreadsheet
- **Historical Backfill**: Bulk load years of historical data
- **Comprehensive Logging**: Detailed logs for monitoring and debugging

Quick Start

1. Installation

```bash
# Clone or copy the pipeline files
cd commodity_pipeline

# Install dependencies
pip install -r requirements.txt
```

2. Configuration

```bash
# Copy the example environment file
cp .env.example .env

# Edit .env with your settings
# Most importantly: add your USDA API key
```

Required configuration:

- `USDA_AMS_API_KEY`: Your API key from USDA My Market News

3. Place the Collector Script

Copy your `usda_ams_collector_asynch.py` file to the pipeline directory:

```bash
cp /path/to/usda_ams_collector_asynch.py ./
```

4. Run the Pipeline

```bash
# Test connections
python main.py test

# View database status
python main.py status

# Collect today's data
python main.py daily

# Collect data for a specific date
python main.py daily --date 11/25/2025

# Run historical backfill
python main.py backfill --start-date 01/01/2020 --end-date 12/31/2024

# Verify data integrity
python main.py verify
```

Architecture

```
┌─────────────────────────────────────────────────────────────┐
│                    Pipeline Orchestrator                    │
│                   (Coordinates all agents)                  │
└─────────────────────────────┬───────────────────────────────┘
                              │
          ┌───────────────────┼───────────────────┐
          ▼                   ▼                   ▼
┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐
│ USDA Collector  │ │ Database Agent  │ │Verification Agent│
│ (Data Fetching) │ │ (Data Storage)  │ │ (Data Integrity) │
```

```
 └──────────────────┘  └────────────────┘  └──────────────────┘
     │           │           │
     ▼           ▼           ▼
 USDA AMS API    SQLite/MySQL    Quality Reports
              PostgreSQL
```

Components

1. **USDA Collector** (`usda_ams_collector_asynch.py`)
   - Fetches data from USDA AMS Market News API
   - Handles authentication and retries
   - Parses various report formats
2. **Database Agent** (`agents/database_agent.py`)
   - Manages database connections
   - Creates and maintains schema
   - Handles bulk inserts with duplicate prevention
3. **Verification Agent** (`agents/verification_agent.py`)
   - Validates record counts
   - Checks data completeness
   - Samples records for value verification
4. **Pipeline Orchestrator** (`core/pipeline_orchestrator.py`)
   - Coordinates the ETL workflow
   - Manages run statistics
   - Handles errors and logging

Database Schema

price_data Table

Column	Type	Description
id	INTEGER	Auto-increment primary key
report_date	DATE	Date of the price observation
commodity	VARCHAR(100)	Commodity name (corn, ethanol, etc.)
location	VARCHAR(150)	Market location
price	DECIMAL(12,4)	Price value
price_low	DECIMAL(12,4)	Low price (if range)

Column	Type	Description
price_high	DECIMAL(12,4)	High price (if range)
basis	DECIMAL(12,4)	Basis value (if applicable)
unit	VARCHAR(50)	Price unit ($/bu, $/gal, etc.)
source_report	VARCHAR(150)	Source report name
report_type	VARCHAR(50)	Report type (grain, ethanol, etc.)
fetch_timestamp	DATETIME	When data was fetched

Unique Constraint: (report_date, commodity, location, source_report)

Configuration Files

.env

Environment variables for API keys, database credentials, and settings.

report_config.xlsx

Excel file defining which USDA reports to fetch:

Column	Description
id	USDA report ID
name	Descriptive name
type	Parser type (grain, ethanol, generic)
frequency	daily or weekly
enabled	true/false

Commands Reference

Command	Description
`python main.py daily`	Collect today's data
`python main.py daily --date MM/DD/YYYY`	Collect specific date
`python main.py backfill`	Historical backfill (uses config dates)
`python main.py backfill --start-date --end-date`	Custom date range
`python main.py status`	Show database statistics
`python main.py verify`	Run verification checks
`python main.py test`	Test API and DB connections
`python main.py reports`	Show configured reports

Options

Option	Description
`--config PATH`	Path to report config file
`--collector PATH`	Path to USDA collector script
`--log-level LEVEL`	DEBUG, INFO, WARNING, ERROR
`--output FILE`	Save results to JSON file

Extending the Pipeline

Adding New Data Sources

1. Create a new collector class following the `USDACollector` interface
2. Register it with the orchestrator
3. Add appropriate parsing logic

Custom Verification Rules

Extend `VerificationAgent` with additional check methods:

```python
def _verify_custom_rule(self, records, source):
    # Your verification logic
    return VerificationResult(...)
```

Database Migration

To switch from SQLite to MySQL/PostgreSQL:

1. Update `.env` with `DB_TYPE=mysql` or `DB_TYPE=postgresql`
2. Add connection credentials
3. Re-run `python main.py status` to create tables

Troubleshooting

Common Issues

"No API key found"

- Ensure `USDA_AMS_API_KEY` is set in your `.env` file

"Could not find usda_ams_collector_asynch.py"

- Place the collector script in the pipeline directory
- Or specify path with `--collector /path/to/script.py`

"Resource not found" for some reports

- Some report IDs may be outdated
- Check current IDs at https://mymarketnews.ams.usda.gov/

Empty results for today

- Markets may be closed (weekends, holidays)
- Try a recent weekday with `--date`

Logs

Check the `./logs` directory for detailed logs:

```bash
tail -f logs/pipeline_20251201.log
```

Future Enhancements

- Scheduled execution (cron/Task Scheduler integration)
- Notion integration for documentation and insights
- Additional data sources (EIA, Census Bureau)
- Fundamental data tables (WASDE, supply/demand)
- Analytics and reporting agents
- Cloud deployment (AWS RDS, S3)

License

Proprietary - Round Lakes Commodities

Support

For issues or questions, review the logs and verify your configuration matches the examples provided.