Step 5: Saving Market Data - Core Data Storage System

🎯 Overview

This directory contains the core data storage system for the Alpaca trading project, implementing the essential data storage requirements outlined in Step 5. The system focuses on efficient data storage, retrieval, and backup capabilities to support backtesting and analysis.

🏗️ System Architecture

The Step 5 system consists of three core components focused on data storage:

1. **Data Management** (`data_management.py`) - Core database operations and data validation
2. **Data Export** (`data_export.py`) - Multi-format data export for analysis and backtesting
3. **Database Migration** (`database_migration.py`) - Database schema management and migration

📊 Current Data Status

- **Total Records**: 285,231
- **Total Symbols**: 200
- **Date Range**: 2018-01-02 to 2025-08-14
- **Database**: SQLite with enhanced schema
- **Data Quality**: Good (418 date gaps expected due to market closures)

🚀 Quick Start

1. Basic Data Overview

```
# Show overview of available data
python data_management.py --overview

# Check data quality
python data_management.py --quality-check
```

2. Export Data

```
# Export to CSV (separate files per symbol)
python data_export.py --format csv --symbols AAPL ABBV --start-date 2025-01-01

# Export to JSON
python data_export.py --format json --symbols AAPL ABBV

# Export for backtesting
python data_export.py --format backtest --symbols AAPL ABBV

# Export summary report
python data_export.py --format summary
```

📁 File Structure

```
Step 5: Saving Market Data/
├── data_management.py      # Core data management system
├── data_export.py          # Multi-format export utilities
├── database_migration.py   # Database schema migration
├── market_data.db          # SQLite database (98MB)
├── data_backups/           # Automated backup directory
├── exports/                # Export output directory
└── README.md               # This documentation
```

🔧 Core Components

Data Management (`data_management.py`)
Key Features: - SQLite database with enhanced schema for OHLCV data - Data validation and cleaning - Automated backup creation - Flexible data retrieval with filtering - Timestamp and timezone handling

Main Classes: - `MarketDataManager` - Core data management operations

Key Methods:

```python
# Initialize manager
manager = MarketDataManager()

# Save data to database
manager.save_data_to_database(data_df, timeframe='Day')

# Retrieve data with filters
data = manager.get_data_from_database(
    symbols='AAPL',
    start_date='2025-01-01',
    end_date='2025-08-14'
)

# Get data summary
summary = manager.get_data_summary()

# Create backup
manager.create_backup()
```

Data Export (`data_export.py`)
Key Features: - Multiple export formats (CSV, JSON, backtesting) - Flexible symbol and date filtering - Separate file export per symbol - Backtesting-ready data format

Main Classes: - `DataExporter` - Export functionality

Key Methods:

```python
# Initialize exporter
exporter = DataExporter()

# Export to CSV
exporter.export_to_csv(
    symbols=['AAPL', 'ABBV'],
    start_date='2025-01-01',
    separate_files=True
)

# Export to JSON
exporter.export_to_json(
    symbols=['AAPL', 'ABBV'],
    start_date='2025-01-01'
)

# Export for backtesting
exporter.export_for_backtesting(
    symbols=['AAPL', 'ABBV'],
    start_date='2025-01-01'
)
```

Database Migration (`database_migration.py`)

Key Features: - Database schema management - Migration utilities for schema updates - Data integrity checks

💾 Storage Implementation

Database Schema

The system uses SQLite with the following table structure:

```sql
CREATE TABLE market_data (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    symbol TEXT NOT NULL,
    timestamp TEXT NOT NULL,
    open REAL NOT NULL,
    high REAL NOT NULL,
    low REAL NOT NULL,
    close REAL NOT NULL,
    volume INTEGER,
    trade_count INTEGER,
    vwap REAL,
    timeframe TEXT DEFAULT 'Day',
    data_source TEXT DEFAULT 'Alpaca',
    created_at TEXT DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(symbol, timestamp, timeframe)
)
```

File Storage

- **CSV Export**: Structured format with OHLCV columns
- **JSON Export**: Hierarchical format with metadata
- **Backtesting Export**: Optimized format for strategy backtesting

Backup Strategy

- Automated daily backups
- Historical backup retention
- Metadata tracking for each backup

🔄 Data Workflow

1. **Data Collection**: Market data retrieved from Alpaca API
2. **Data Storage**: Data saved to SQLite database with validation
3. **Data Export**: Data exported in various formats for analysis
4. **Data Backup**: Regular automated backups for data safety

📈 Usage Examples

Basic Data Retrieval

```python
from data_management import MarketDataManager

# Initialize manager
manager = MarketDataManager()

# Get recent data for AAPL
data = manager.get_data_from_database(
    symbols='AAPL',
    start_date='2025-08-01'
)

print(f"Retrieved {len(data)} records for AAPL")
```

Data Export

```python
from data_export import DataExporter

# Initialize exporter
exporter = DataExporter()

# Export recent data for multiple symbols
success = exporter.export_to_csv(
    symbols=['AAPL', 'ABBV', 'ABT'],
    start_date='2025-07-01',
    separate_files=True
)

if success:
    print("Export completed successfully")
```

🎯 Step 5 Requirements Implementation

This implementation covers all the core requirements from Step 5:

1. ✅ **Choose Data Storage Method**: SQLite database with file export options
2. ✅ **Database Storage**: Enhanced schema for OHLCV data with proper indexing
3. ✅ **File Storage**: Structured CSV/JSON export with organized directory structure
4. ✅ **Timestamps and Timezones**: Proper timestamp handling and timezone management
5. ✅ **Regular Updates**: Automated backup and data management processes
6. ✅ **Automate Data Retrieval and Storage**: Integrated data management pipeline
7. ✅ **Backtesting Integration**: Optimized export formats for backtesting
8. ✅ **Data Cleaning and Validation**: Comprehensive data validation and cleaning

📝 Notes

- **Analysis and Visualization**: Advanced analysis tools have been moved to Step 7 (Trading Strategy)
- **Workflow Automation**: Complex workflow pipelines are available in Step 7
- **Focus**: This step focuses purely on data storage, retrieval, and export capabilities
- **Integration**: Designed to work seamlessly with Step 4 (Data Collection) and Step 7 (Trading Strategy)