CLAUDE.md

This file provides guidance to Claude Code (claude.ai/code) when working with code in this repository.

Project Overview

This is a KrakenPro cryptocurrency trading data pipeline that collects, processes, and stores multi-timeframe OHLC data with 29 technical indicators. The system is deployed on Google Cloud Platform (GCP) project `cryptobot-462709` and consists of three automated data collection pipelines (daily, hourly, and 5-minute intervals) that feed into BigQuery for AI-powered trading analysis.

Architecture

Data Collection Pipeline

Three Cloud Functions run on different schedules: - **Daily Function** (`cloud_function_daily/`): Fetches daily OHLC for all ~675 USD trading pairs at midnight ET - **Hourly Function** (`cloud_function_hourly/`): Fetches hourly OHLC for all USD pairs every hour - **5-Minute Function** (`cloud_function_5min/`): Fetches 5-minute OHLC for top 10 hourly gainers every 5 minutes

Each function: 1. Queries Kraken Pro API for OHLC data 2. Calculates 29 technical indicators (RSI, MACD, Bollinger Bands, ADX, CCI, etc.) 3. Uploads to BigQuery with duplicate detection 4. Implements rate limiting (1.5s between API calls)

Technical Indicators Calculation

All three pipelines share identical `calculate_technical_indicators(df)` function that computes: - Momentum: RSI, MACD, ROC, Momentum, Stochastic, Williams %R - Trend: SMA (20/50/200), EMA (12/26/50), ADX, KAMA, TRIX - Volatility: Bollinger Bands, ATR - Volume: OBV, PVO - Oscillators: CCI, PPO, Ultimate Oscillator, Awesome Oscillator

The function requires minimum data points (50-200 candles depending on indicator) to calculate properly.

BigQuery Schema

Dataset: `cryptobot-462709.crypto_trading_data`

Tables: - `crypto_analysis` (45 fields): Daily data - `crypto_hourly_data` (46 fields): Hourly data - `crypto_5min_top10_gainers` (43 fields): 5-minute data for top gainers

All tables share same schema except 5-minute lacks `altname`, `base`, `quote` fields. The 5-minute function depends on hourly data to identify top gainers.

Cloud Schedulers

Three schedulers trigger functions via HTTP: - `daily-crypto-fetch-job`: 0 0 * * * (midnight ET) - `hourly-crypto-fetch-job`: 0 * * * * (hourly) - `fivemin-top10-fetch-job`: */5 * * * * (every 5 minutes)

Timezone: America/New_York

Stock Price App

React + Vite application in `stock-price-app/` for visualizing trading data: - Uses Recharts and Lightweight Charts for visualization - Deploys to Cloud Run - Dockerfile + nginx for production serving

Common Commands

Check BigQuery Data

```
python check_bigquery_counts.py
```

Displays record counts, recent data, latest timestamps, and unique pairs for all three tables.

Manually Trigger Cloud Functions

```
 # Make functions publicly accessible first (required
gcloud functions add-invoker-policy-binding daily-cry
gcloud functions add-invoker-policy-binding hourly-cr
gcloud functions add-invoker-policy-binding fivemin-t

# Trigger functions via HTTP
curl https://daily-crypto-fetcher-cnyn5l4u2a-uc.a.run
curl https://hourly-crypto-fetcher-cnyn5l4u2a-uc.a.ru
curl https://fivemin-top10-fetcher-cnyn5l4u2a-uc.a.ru
```

View Cloud Function Logs

```
 gcloud functions logs read daily-crypto-fetcher --pro
gcloud functions logs read hourly-crypto-fetcher --pr
gcloud functions logs read fivemin-top10-fetcher --pr
```

Check Cloud Schedulers

```
 gcloud scheduler jobs list --project=cryptobot-462709

# Manually trigger a scheduler
gcloud scheduler jobs run daily-crypto-fetch-job --lo
```

Deploy Cloud Functions

```
 # From root directory
cd cloud_function_daily && python deploy_via_api.py
cd cloud_function_hourly && python deploy.py
cd cloud_function_5min && python deploy_all.py
```

Deploy Trading App to Cloud Run

```
 cd stock-price-app

gcloud run deploy crypto-trading-app \
  --source . \
  --platform managed \
  --region us-central1 \
  --allow-unauthenticated \
  --port 8080 \
  --project cryptobot-462709
```

Local Development - Trading App

```
 cd stock-price-app
npm install
npm run dev      # Development server
npm run build    # Production build
npm run preview  # Preview production build
npm run lint     # Run ESLint
```

Key Implementation Details

Duplicate Detection

All functions implement duplicate detection before uploading to BigQuery:

```
 # Query existing records by timestamp range
# Create composite key: pair + timestamp
# Filter out existing records
# Upload only new records
```

Rate Limiting

Kraken API requires rate limiting between calls:

```
time.sleep(1.5)  # 1.5 seconds between pair requests
```

Error Handling

- Failed pairs are logged but don't stop the entire fetch
- Functions return 200 even if some pairs fail
- Minimum data checks prevent indicator calculation on insufficient data

Project Migration Pattern

When migrating between GCP projects, use `deploy_all_to_cryptobot.py` as reference: 1. Update PROJECT*ID constants in all main.py files 2. Update DATASET*ID references 3. Update deploy scripts 4. Maintain same function names and entry points

Deployment Status

The system is 95% deployed. All infrastructure exists but requires: 1. Making functions publicly accessible (3 gcloud commands) 2. Initial function triggers to populate tables 3. Optional trading app deployment

See `QUICK_START_GUIDE.md` for step-by-step setup instructions.

Cost Structure

Monthly estimated costs (~$135/month): - Cloud Functions: $126 (daily: $4, hourly: $72, 5-min: $50) - BigQuery Storage: $2 - Cloud Scheduler: $0.30 - Cloud Run: $5 (if deployed)

Important Files

- `cloud_function_*/main.py` : Core data fetching and indicator calculation logic
- `check_bigquery_counts.py` : Verify data collection is working
- `setup_schedulers_cryptobot.py` : Configure Cloud Schedulers
- `deploy_all_to_cryptobot.py` : Reference for project migration
- `QUICK_START_GUIDE.md` : Step-by-step deployment instructions
- `FINAL_DEPLOYMENT_STATUS.md` : Current deployment status

Technical Constraints

1. **Windows Environment**: Several scripts include Windows-specific encoding fixes:

```
 import sys
import io
if sys.platform == 'win32':
    sys.stdout = io.TextIOWrapper(sys.stdout.buffer,
```

2. **Data Dependencies**: 5-minute function requires hourly table to be populated first to identify top gainers

3. **Indicator Calculation**: Requires minimum historical data (50-200 candles) for accurate technical indicators

4. **API Limits**: Kraken API enforces rate limits; functions implement 1.5s delays between requests

Query Examples

Find oversold cryptos with strong trends:

```
 SELECT pair, close, rsi, adx, sma_200, datetime
FROM `cryptobot-462709.crypto_trading_data.crypto_ana
WHERE DATE(datetime) = CURRENT_DATE() - 1
  AND rsi < 30        -- Oversold
  AND adx > 25        -- Strong trend
  AND close > sma_200 -- Above 200-day MA
ORDER BY rsi ASC
LIMIT 20;
```

Get top 10 gainers from hourly data:

```
 WITH latest_hour AS (
    SELECT pair, close, timestamp,
           ROW_NUMBER() OVER (PARTITION BY pair ORDER
    FROM `cryptobot-462709.crypto_trading_data.crypto_
    WHERE datetime >= TIMESTAMP_SUB(CURRENT_TIMESTAMP
)
SELECT pair, close
FROM latest_hour
WHERE rn = 1
ORDER BY close DESC
LIMIT 10;
```