CLAUDE CODE MASTER OPTIMIZATION PROMPT

AIAlgoTradeHits.com - Trading Intelligence Platform

Copy this entire prompt and paste it at the start of any Claude Code session.

🎯 PROJECT CONTEXT

You are working on **AIAlgoTradeHits.com**, a cross-asset AI-driven trading analytics platform. Always optimize for speed, cost-efficiency, and leverage all available paid subscriptions.

📦 ACTIVE SUBSCRIPTIONS & API KEYS

1. TwelveData API ($229/month - PREMIUM TIER)

```
API_KEY: 16ee060fd4d34a628a14bcb6f0167565
BASE_URL: https://api.twelvedata.com
RATE_LIMIT: 1597 API credits/minute (PAID TIER - USE AGGRESSIVELY)
```

ALWAYS USE TWELVEDATA AS PRIMARY DATA SOURCE for:

- Stocks, ETFs, Forex, Crypto, Indices, Commodities
- 112 technical indicators via `/technical_indicators` endpoint
- Historical data with up to 5000 data points per request
- **Batch requests**: Combine up to 8 symbols per request to maximize throughput

2. Kraken API (FREE - WebSocket for Real-Time)

```
WEBSOCKET_URL: wss://ws.kraken.com/
REST_URL: https://api.kraken.com
RATE_LIMIT: 15-20 API counter (decays 0.33-1/sec depending on tier)
```

USE FOR: Real-time crypto streaming, order book data, ticker updates **OPTIMIZATION**: Use WebSocket instead of polling for live data

3. CoinMarketCap API

```
BASE_URL: https://pro-api.coinmarketcap.com
RATE_LIMIT: 30 calls/minute (Basic), scales with tier
```

USE FOR: Market cap rankings, global metrics, crypto metadata

4. Finnhub API

```
BASE_URL: https://finnhub.io/api/v1
```

USE FOR: News sentiment, company financials, alternative data

5. GCP Vertex AI + Gemini

```
PROJECT_ID: cryptobot-462709
REGION: us-central1
MODEL: gemini-2.0-pro (preferred), gemini-1.5-pro (fallback)
```

PRICING:

- Input: $1.25/1M tokens (under 128K context)
- Output: $5.00/1M tokens
- **USE CACHING** for repeated context (75% discount)

🚀 PARALLEL PROCESSING OPTIMIZATION

Cloud Functions Architecture

Always deploy data fetchers with **PARALLEL EXECUTION**:

```
python
```

```
# OPTIMAL: Split workload across 10 parallel functions
PARALLEL_WORKERS = 10
SYMBOLS_PER_WORKER = total_symbols // PARALLEL_WORKERS

# Deploy pattern:
for i in range(PARALLEL_WORKERS):
    deploy_function(f"fetcher-{i}", symbols[i*batch:(i+1)*batch])
```

Concurrent API Calls (Python)

python

```
import asyncio
import aiohttp

async def fetch_batch(symbols, session):
    tasks = [fetch_symbol(s, session) for s in symbols]
    return await asyncio.gather(*tasks, return_exceptions=True)

# TwelveData batch optimization
async def fetch_twelvedata_batch(symbols):
    # Batch up to 8 symbols per request
    batches = [symbols[i:i+8] for i in range(0, len(symbols), 8)]
    async with aiohttp.ClientSession() as session:
        results = await asyncio.gather(*[
            session.get(
                "https://api.twelvedata.com/time_series",
                params={"symbol": ",".join(batch), "apikey": API_KEY, ...}
            ) for batch in batches
        ])
    return results
```

Cloud Tasks for Distributed Workload

bash

```
# Create task queue for high-volume processing
gcloud tasks queues create data-fetch-queue \
    --max-concurrent-dispatches=100 \
    --max-dispatches-per-second=500
```

GCP INFRASTRUCTURE

BigQuery Configuration

```
PROJECT: cryptobot-462709
DATASET: crypto_trading_data
REGION: us-central1
```

Tables (24 total):

- `{asset}_weekly`, `{asset}_daily`, `{asset}_hourly`, `{asset}_5min`
- Assets: crypto, stocks, etfs, forex, indices, commodities

OPTIMIZATION RULES:

1. Always use `PARTITION BY date` and `CLUSTER BY symbol`
2. Use `INSERT ... SELECT` for bulk operations (not row-by-row)
3. Use streaming inserts for real-time data
4. Query only needed columns (avoid `SELECT *`)

Cloud Functions Best Practices

```yaml
Runtime: python311
Memory: 512MB (default), 2GB (AI functions)
Timeout: 540s (max for 2nd gen)
Concurrency: 80 (default), 1000 (high-throughput)
Min instances: 0 (cost savings), 1 (low latency)
```

AI INTEGRATION PRIORITY

Model Selection (Cost vs Performance)

Use Case	Model	Cost
Simple queries	Gemini Flash	$
Daily analysis	Gemini Pro	$$

Use Case	Model	Cost
Complex patterns	Claude Sonnet	$$$
Critical decisions	Claude Opus	$$$$

AI Cloud Function Endpoints

```
/api/ai/chat       - Conversational queries
/api/ai/predict    - Price predictions
/api/ai/patterns   - Pattern detection
/api/ai/sentiment  - Sentiment analysis
/api/ai/signals    - Trading signals
```

⚡ SPEED OPTIMIZATION CHECKLIST

When processing data, ALWAYS:

1. **Use TwelveData's batch endpoint** - 8 symbols per request
2. **Deploy parallel Cloud Functions** - Split workload 10x
3. **Use Kraken WebSocket** for real-time crypto (not polling)
4. **Enable BigQuery result caching** for repeated queries
5. **Use async/await** for all API calls
6. **Implement exponential backoff** for rate limits
7. **Cache AI responses** with Redis or in-memory for identical prompts

Data Collection Schedule (Optimized)

```
Daily:    Once at midnight ET (all assets)
Hourly:   Every 4 hours (balance cost/freshness)
5-Min:    Top 20 assets only (during market hours)
Real-time: WebSocket for top 10 crypto pairs
```

🔧 ENVIRONMENT VARIABLES TEMPLATE

bash

```
# Data APIs
TWELVEDATA_API_KEY=16ee060fd4d34a628a14bcb6f0167565
TWELVEDATA_BASE_URL=https://api.twelvedata.com
KRAKEN_WS_URL=wss://ws.kraken.com/
CMC_API_KEY=your_coinmarketcap_key
FINNHUB_API_KEY=your_finnhub_key

# GCP
GCP_PROJECT_ID=cryptobot-462709
GCP_REGION=us-central1
BIGQUERY_DATASET=crypto_trading_data

# AI
VERTEX_AI_MODEL=gemini-2.0-pro
ANTHROPIC_API_KEY=your_anthropic_key

# Optimization
PARALLEL_WORKERS=10
BATCH_SIZE=8
ENABLE_CACHING=true
```

📋 QUICK REFERENCE COMMANDS

Deploy Parallel Fetchers

bash

```
for i in {0..9}; do
  gcloud functions deploy fetcher-$i \
    --gen2 --runtime=python311 --region=us-central1 \
    --memory=512MB --timeout=540s \
    --set-env-vars="WORKER_ID=$i,TOTAL_WORKERS=10"
done
```

Trigger All Fetchers

bash

```
# Parallel execution with GNU parallel
parallel -j10 curl {} ::: $(gcloud functions list --format="value(url)")
```

BigQuery Bulk Insert

```
sql

INSERT INTO `cryptobot-462709.crypto_trading_data.stocks_daily`
SELECT * FROM EXTERNAL_QUERY(
  "projects/cryptobot-462709/locations/us/connections/twelve_data",
  "SELECT * FROM staging_table"
);
```

⚠️ CRITICAL REMINDERS

1. **TwelveData is PAID ($229/mo)** - Maximize usage, don't throttle unnecessarily
2. **Always use parallel processing** - Never sequential for bulk operations
3. **Prefer WebSocket over REST** for real-time data
4. **Cache everything** - AI responses, API results, BigQuery queries
5. **Batch API calls** - TwelveData supports 8 symbols/request
6. **Use Gemini first** - 66% cheaper than Claude for routine analysis
7. **Monitor costs** - Set GCP budget alerts at $200, $500, $1000

🎯 DEFAULT BEHAVIOR

When I ask you to:

- **Fetch data**: Use TwelveData with batching + parallel functions
- **Stream prices**: Use Kraken WebSocket
- **Analyze patterns**: Use Gemini Pro, escalate to Claude for complex
- **Build pipelines**: Deploy 10 parallel Cloud Functions
- **Query data**: Use BigQuery with partitioning/clustering
- **Process bulk**: Use Cloud Tasks queue with 100+ concurrency

Always optimize for speed first, then cost.

Last Updated: December 2025 Platform: AIAlgoTradeHits.com