Crypto Intelligence System Expansion: Technical Design Document

Date: 2025-06-14

Executive Summary

This document outlines a comprehensive technical design plan for expanding the existing MindsDB-based Crypto Intelligence System. The current system, which successfully integrates CoinMarketCap for market data, TimeGPT for price forecasting, and AI agents for analysis, will be enhanced by incorporating five additional powerful APIs: DefiLlama, Binance, Blockchain.com, Whale Alert, and the Dune Analytics Sim API.

This expansion aims to significantly broaden the system's data coverage and analytical capabilities, providing deeper insights into DeFi markets, real-time trading dynamics, blockchain network health, large-scale transaction movements, and custom on-chain analytics. The design plan details the integration architecture, data flow, new agent designs, database schema updates, implementation phases, and critical considerations for performance, rate limiting, cost, and risk. The ultimate goal is to create an even more robust, versatile, and insightful cryptocurrency intelligence platform.

1. Current System Analysis

The existing Crypto Intelligence System, as detailed in the "MindsDB Crypto Intelligence System" documentation and the "Developer Technical Guide," is a sophisticated platform built on MindsDB. Its core strengths lie in real-time data ingestion, AI-powered forecasting, and intelligent agent-based analysis, all while maintaining significant cost optimization.

1.1. Core Components & Architecture

- **Data Source**: Primarily uses a CoinMarketCap API Handler (`coinmarketcap_datasource`) for market data on 1000+ cryptocurrencies, updating every 15-30 minutes in development.
- **AI Engines**:
  - **TimeGPT (`timegpt_engine`)**: Utilized for specialized time-series price forecasting (e.g., `crypto_price_forecaster` model for 24-hour Bitcoin price predictions).
  - **Anthropic Claude Haiku (`anthropic_engine`)**: Powers cost-effective intelligent analysis agents.
  - **OpenAI GPT-3.5 (`openai_engine`)**: Used for general market analysis and sentiment analysis agents.
  - **Lightwood**: A built-in MindsDB engine, noted to have current dependency issues (`ModuleNotFoundError: No module named 'lightwood'`).
- **Database**: PostgreSQL (`crypto_postgres`) is used for persistent data storage, although the primary live data interaction seems to be directly with the MindsDB data source handlers. The `DEVELOPER_TECHNICAL_GUIDE.md` shows an example `crypto_postgres.price_history` table.
- **Intelligent Agents**: Four agents are currently implemented:
  1. `price_alert_agent` (Claude Haiku): Generates alerts based on TimeGPT forecasts and market data.
  2. `analytics_agent` (Claude Haiku): Provides market condition insights and risk assessment.
  3. `crypto_analytics_agent` (GPT-3.5-turbo): Offers general market analysis.
  4. `sentiment_agent` (GPT-3.5-turbo): Classifies market sentiment.

- **Dashboarding**: SQL-based views (`crypto_dashboard` , `market_alerts`) are created within MindsDB for real-time market views and alerts.
- **MindsDB Project**: The system is organized within a MindsDB project named `crypto_analytics` .

1.2. Data Flow

1. CoinMarketCap API Handler fetches data into `coinmarketcap_datasource.listings` and other related tables.
2. TimeGPT models (e.g., `crypto_price_forecaster`) use data from `coinmarketcap_datasource` to generate predictions.
3. AI Agents query `coinmarketcap_datasource` tables and TimeGPT model predictions to provide analyses and alerts.
4. PostgreSQL can be used for historical data storage via jobs (e.g., `crypto_data_sync_dev`).
5. Views provide summarized data for dashboards.

1.3. Strengths

- **Cost-Effective AI**: Significant savings by using specialized models like TimeGPT and cost-optimized LLMs like Claude Haiku.
- **Real-Time Capabilities**: Designed for real-time data ingestion and analysis.
- **SQL-Based Interface**: Leverages MindsDB's SQL syntax for data querying, model creation, and agent interaction.
- **Scalable Architecture**: Built with components that can scale.

1.4. Areas for Expansion (as per objective)

The current system relies heavily on CoinMarketCap for market data. The planned expansion will introduce diverse data dimensions:
* DeFi specific metrics (TVL, yields).
* Granular real-time exchange data (order books, trades).
* Fundamental blockchain network statistics (hash rate, block data).
* On-chain large transaction monitoring.
* Customizable, deep on-chain analytics via SQL queries on decoded blockchain data.

This expansion will transform the system from a market-focused platform to a comprehensive, multi-faceted crypto intelligence hub.

2. New APIs Technical Analysis

This section details the capabilities, integration methods, and other relevant aspects of the five new APIs to be integrated.

2.1. DefiLlama API

- **Capabilities**: Provides comprehensive data on DeFi protocols, including Total Value Locked (TVL) across various chains and protocols, yield farming opportunities (APIs, lending/borrowing rates), protocol revenue, fees, and cross-chain bridge activity. It aims to be an open and transparent source for DeFi analytics.
- **Data Types**:
    - TVL (historical and real-time, excluding liquid staking and double-counted TVL).
    - Yields (APYs, pools, stablecoins, ETH).
    - Protocol metadata (name, category, chains).
    - Chain-specific data.

    - Bridge data (volumes, top tokens).
- **Integration Methods**:
    - Public REST API (e.g., `/protocols`, `/v2/historicalChainTvl/{chain}`, `/yields`, `/chains`).
    - Data is often sourced via "adapters" (scripts that fetch and standardize data from various protocols), managed in their `DefiLlama-Adapters` GitHub repository. This implies that the API's comprehensiveness relies on these community-contributed or team-maintained adapters.
- **Authentication**: The public API generally does not require authentication for basic data access, but rate limits may apply. Specific premium features or higher access tiers, if any, are not detailed in the provided research but should be investigated.
- **Pricing**: The core API is presented as open and free. The research does not specify costs for standard API usage.
- **Rate Limiting**: Not explicitly detailed in the provided research for the public API, but standard practice suggests undocumented or generous limits for public use, with potential for stricter limits or dedicated endpoints for heavy users.
- **Key Endpoints (Examples from research)**:
    - `/protocols`: List all protocols with current TVL.
    - `/charts/{chain}`: Historical TVL for a specific chain.
    - `/yields`: Aggregated yield farming data.

2.2. Binance API

- **Capabilities**: Offers extensive access to real-time and historical trading data from the Binance exchange. This includes order book data, recent trades, candlestick (Kline) data, 24hr ticker statistics, and current average prices.
- **Data Types**:
    - Order Book: Price levels, quantities (bids/asks).
    - Trades: Price, quantity, timestamp, buyer/seller maker status.
    - Kline/Candlestick: OHLCV data for various intervals (1s to 1M).
    - Ticker Statistics: Price change, volume, high/low, weighted average price.
    - Account Information & Trading: (Not the focus of this data ingestion plan, but available).
- **Integration Methods**:
    - **REST API**: For fetching snapshot data, historical data, and placing/managing orders. Endpoints like `/api/v3/depth`, `/api/v3/trades`, `/api/v3/klines`, `/api/v3/ticker/24hr`.
    - **WebSocket Streams**: For real-time data feeds, including individual trades (`<symbol>@trade`), order book updates (`<symbol>@depth`), and Kline updates (`<symbol>@kline_<interval>`). Essential for low-latency applications.
- **Authentication**:
    - Public data endpoints (like market data) generally do not require an API key.
    - Authenticated endpoints (for trading, account data) require API key and secret, often included in request headers or query parameters with a signature.
- **Pricing**: Access to market data APIs is generally free. Trading fees apply for executed orders.
- **Rate Limiting**: Binance employs IP-based and UID-based rate limits. For REST API, limits are typically per minute (e.g., 1200 requests per minute per IP). WebSocket connections also have limits (e.g., 5 messages per second). Weights are assigned to different REST endpoints.
- **Key Endpoints (Examples from research)**:
    - `GET /api/v3/depth`: Order book.
    - `GET /api/v3/trades`: Recent trades.

- `GET /api/v3/klines` : Candlestick data.
- `GET /api/v3/ticker/24hr` : 24hr price change statistics.
- WebSocket: `wss://stream.binance.com:9443/ws/<symbol>@<streamName>`

2.3. Blockchain.com API

- **Capabilities**: Provides access to blockchain network statistics, block data, transaction details, and market data. Focuses on fundamental on-chain metrics.
- **Data Types**:
  - Block Data: Height, hash, timestamp, transactions, size.
  - Transaction Data: Hash, inputs, outputs, fees.
  - Network Statistics: Hash rate, difficulty, block height, mempool size.
  - Address Data: Balance, transaction history.
  - Market Data: Price, market cap (though CoinMarketCap is already used for this).
- **Integration Methods**:
  - **REST API**: For querying blocks, transactions, addresses, and network stats (e.g., `blockchain.info/rawblock/{block_hash}` , `blockchain.info/q/totalbc`).
  - **WebSocket API**: For real-time notifications of new blocks and transactions.
  - **Charts API**: JSON feed for data used in their charts and stats pages.
- **Authentication**: Most public data endpoints are accessible without an API key. For higher usage or specific services, an API key might be required. The documentation mentions “Build bitcoin apps on top of our APIs for free.”
- **Pricing**: Generally free for public data access. Rate limits apply.
- **Rate Limiting**: The public API has rate limits to prevent abuse (e.g., one request every 10 seconds for some endpoints). Users needing higher limits can request an API key.
- **Key Endpoints (Examples from research)**:
  - `/q/getblockcount` : Current block height.
  - `/q/hashrate` : Estimated network hash rate.
  - `/rawblock/{block_hash_or_index}` : Detailed block data.
  - `/rawtx/{tx_hash_or_index}` : Detailed transaction data.
  - WebSocket: `wss://ws.blockchain.info/inv`

2.4. Whale Alert API

- **Capabilities**: Specializes in monitoring and reporting large cryptocurrency transactions (“whale” activity) across multiple blockchains. Provides real-time alerts and historical data for significant transfers, mints, burns, and other events.
- **Data Types**:
  - Transaction Details: Hash, timestamp, blockchain, from/to addresses (and their attributed owners if known), amount, symbol, USD value.
  - Event Types: Transfer, mint, burn, freeze, unfreeze, lock, unlock.
- **Integration Methods**:
  - **REST API**: To retrieve live and historical transaction data. Endpoints for fetching specific transactions or recent large transactions with filters.
  - **WebSocket API**: For real-time, push-based alerts. Users can subscribe to specific event types, blockchains, symbols, and minimum USD values.
- **Authentication**: Requires an API key for both REST and WebSocket access. API keys are tied to subscription plans.

- **Pricing**:
  - **Personal Plan**: $29.95/month (up to 100 alerts/hour, WebSocket access).
  - **Enterprise REST API**: $699/month (live feed, 30 days historical, 100 CPM rate limit).
  - **Priority Alerts API**: $1299/month (lowest latency WebSocket, 1000 alerts/hour).
  - Free trial available for Personal Plan.
- **Rate Limiting**: Varies by plan. Personal plan has limits like 100 alerts/hour. Enterprise REST API has 100 calls per minute (CPM).
- **Key Endpoints/Streams (Examples from research)**:
  - REST: `/v1/transaction/{blockchain}/{hash}`, `/v1/transactions`
  - WebSocket: `wss://leviathan.whale-alert.io/ws?api_key=YOUR_API_KEY`
    - Subscription message: `{"type": "subscribe_alerts", "min_value_usd": 100000, ...}`

2.5. Dune Analytics Sim API

- **Capabilities**: The “Sim APIs” by Dune provide access to real-time blockchain activity and ownership data across 60+ chains with a single request. This is distinct from querying Dune’s user-generated SQL dashboards. It focuses on normalized, real-time data with enriched metadata and pricing. The research highlights an EVM Balances API as an example. Dune itself is known for its powerful DuneSQL engine (a Trino fork) for custom on-chain data analysis via its platform, but the Sim API seems to be a more direct, structured data access layer.
- **Data Types (based on Balances API example)**:
  - Token Balances: Address, amount, chain, decimals, price_usd, symbol, value_usd.
  - Likely other endpoints for transactions, token metadata, etc. (as suggested by “Explore EVM Endpoints” in research).
- **Integration Methods**:
  - **REST API**: Requests are made over HTTPS. The example shows a GET request.
  - URL structure: `https://api.sim.dune.com/v1/...`
- **Authentication**: Requires an API key included in the `X-Sim-Api-Key` header.
- **Pricing**: Not detailed in the provided research data. Typically, such APIs have tiered pricing based on usage, features, and data access levels. This needs to be investigated from Dune’s official Sim API documentation/ pricing page.
- **Rate Limiting**: Not detailed in the provided research, but standard for such APIs. Limits would likely depend on the subscription tier.
- **Key Endpoints (Example from research)**:
  - `GET https://api.sim.dune.com/v1/evm/balances/{wallet_address}`

3. API Integration Architecture

The expanded system will integrate these new APIs as data sources within the MindsDB ecosystem, similar to the existing CoinMarketCap handler.

3.1. MindsDB Handler Integration

For each new API, a suitable MindsDB integration strategy will be employed:

- **DefiLlama, Blockchain.com, Dune Analytics Sim API, Whale Alert (REST parts)**:
  - These primarily offer REST APIs. If MindsDB has a generic HTTP/REST API handler, it could be configured.
  - If not, or if more complex logic (pagination, specific authentication flows, data transformation) is needed, **custom MindsDB handlers** might need to be developed. This involves writing Python code that conforms to MindsDB’s data handler interface.

    - The `DefiLlama-Adapters` repository suggests that for DefiLlama, the "API" might be an aggregation layer over scripts that fetch data. A custom handler would be more likely here to encapsulate this logic or interact with their structured API endpoints.
- **Binance API, Whale Alert (WebSocket parts)**:
    - MindsDB's primary interaction model is pull-based (querying a data source). Integrating real-time WebSocket streams directly into a MindsDB `DATABASE` handler is less common.
    - **Strategy 1 (Preferred for real-time agents)**: A separate Python service (or services) could subscribe to Binance and Whale Alert WebSockets. This service would then write the incoming data into a message queue (e.g., Kafka, Redis Streams) or directly into staging tables in the PostgreSQL database. MindsDB could then query these staging tables.
    - **Strategy 2 (Micro-batching)**: If near real-time is sufficient, the REST API counterparts could be polled frequently (respecting rate limits). For Binance, this is less ideal for capturing every tick or order book change.
    - **Strategy 3 (MindsDB Stream Jobs - if applicable)**: Explore if MindsDB's job or streaming capabilities can directly ingest from a message queue populated by the WebSocket clients.

3.2. System Diagram (Conceptual)

```
External APIs:
  - CoinMarketCap API --> [CMC Handler]
  - DefiLlama API ------> [DefiLlama Handler/Custom] ----> MindsDB Core
  - Binance API --------> [Binance Handler/Custom/WebSocket Service] -> |
  - Blockchain.com API -> [Blockchain.com Handler/Custom] ------------> | (AI Engines:
TimeGPT, Claude, OpenAI)
  - Whale Alert API ----> [Whale Alert Handler/Custom/WebSocket Service] > | (Agents)
  - Dune Sim API -------> [Dune Sim API Handler/Custom] ---------------> |
                                                                         |
                                                                         V
                                                                PostgreSQL Database
                                                                  (Persistent Stor-
age,
                                                                   Staging for Web-
Socket Data)
                                                                         ^
                                                                         |
                                                                MindsDB Queries & Vie
ws
                                                                         |
                                                                         V
                                                                Dashboards /
External Apps
```

- **Handlers**: Each API will have a corresponding "handler" or integration point within or feeding into MindsDB.
- **WebSocket Services**: Dedicated services for Binance and Whale Alert WebSockets will push data into PostgreSQL staging tables or a message queue.
- **MindsDB Core**: Continues to host AI engines, models, and agents.
- **PostgreSQL**: Expands its role to store data from all new sources, including real-time streams.

4. Data Flow Design

1. **DefiLlama Data**:
    - A MindsDB handler (custom or configured generic) will periodically poll DefiLlama API endpoints (e.g., for TVL, yields).
    - Data will be ingested into new tables in `crypto_postgres` (e.g., `defillama_protocols`, `defillama_tvl_history`, `defillama_yields`).
    - MindsDB will access this data via its connection to `crypto_postgres` or directly if the handler materializes tables within MindsDB's virtual database layer.
2. **Binance Data**:
    - **Real-time (WebSocket)**: A dedicated Python service will connect to Binance WebSocket streams (trades, order book depth for selected pairs). This service will parse messages and insert them into PostgreSQL staging tables (e.g., `binance_live_trades`, `binance_order_books`) with high frequency.
    - **Historical/Snapshot (REST)**: A MindsDB handler will poll REST API endpoints for Kline data, 24hr stats, etc., storing them in tables like `binance_klines`, `binance_ticker_stats`.
    - Agents can query both the live staging tables (for immediate reactions) and historical tables.
3. **Blockchain.com Data**:
    - A MindsDB handler will poll relevant REST API endpoints for network statistics (hash rate, difficulty) and block summaries.
    - Data will be stored in tables like `blockchain_network_stats`, `blockchain_blocks`.
    - For real-time block/transaction notifications, a WebSocket client (similar to Binance's) could feed a staging table, or agents could use the REST API for specific lookups.
4. **Whale Alert Data**:
    - **Real-time (WebSocket)**: A dedicated Python service will subscribe to Whale Alert WebSocket API for transactions meeting defined criteria (e.g., >$1M USD). Alerts will be written to a PostgreSQL table (e.g., `whale_transactions_live`).
    - **Historical/Lookup (REST)**: The REST API can be used by a MindsDB handler for ad-hoc lookups or fetching recent transactions if the WebSocket feed is missed, storing in `whale_transactions_archive`.
5. **Dune Analytics Sim API Data**:
    - A MindsDB handler will poll the Dune Sim API endpoints (e.g., balances, and potentially others like transactions or token metadata once explored).
    - Data will be stored in tables like `dune_wallet_balances`, `dune_token_metadata`.
6. **Data Aggregation & Agent Access**:
    - Agents within MindsDB will now have access to a much richer dataset by querying tables from `coinmarketcap_datasource` and the new tables in `crypto_postgres` (or virtual tables created by new handlers).
    - Views can be created in MindsDB/PostgreSQL to join and pre-process data from multiple sources for easier agent consumption or dashboarding.

5. New Agent Designs

These agents will leverage data from multiple new APIs in conjunction with existing data sources.

Agent 5: DeFi Opportunity & Risk Analyzer

- **Purpose**: To identify promising DeFi yield opportunities and assess associated risks by correlating TVL changes, protocol performance, and market sentiment.
- **Data Sources**:
  - `defillama_protocols`, `defillama_tvl_history`, `defillama_yields` (DefiLlama)
  - `coinmarketcap_datasource.listings` (for token prices, market cap)
  - `sentiment_agent` (existing, for overall market sentiment)
  - `binance_klines` (for price volatility of underlying assets)
- **Prompt Template Example**:
  ```
  You are a DeFi Opportunity and Risk Analyzer. Based on the latest DefiLlama TVL trends for protocol {protocol_name}, its current yield offerings for {asset_type} pools, the price volatility of its native token {token_symbol} from Binance, and overall market sentiment, provide:
  1. An assessment of the opportunity (potential APY, stability).
  2. Key risks (e.g., TVL decline, high token volatility, smart contract risk indicators if available, impermanent loss potential).
  3. A confidence score (Low, Medium, High) for engaging with this opportunity.
  Consider recent changes in TVL, yield consistency, and market cap of the protocol.
  ```
- **Example Query**:
  ```sql
  SELECT * FROM defi_opportunity_agent
  WHERE question = 'Analyze opportunities for stablecoin farming on Curve Finance, considering its TVL on Ethereum and recent CRV token performance.';
  ```

Agent 6: On-Chain Activity & Market Impact Correlator

- **Purpose**: To correlate significant on-chain events (large transactions, network congestion) with market movements and exchange activity.
- **Data Sources**:
  - `whale_transactions_live` (Whale Alert)
  - `blockchain_network_stats` (Blockchain.com - e.g., mempool size, transaction fees)
  - `binance_live_trades`, `binance_order_books` (Binance - for immediate market reaction)
  - `coinmarketcap_datasource.listings` (for price changes)
- **Prompt Template Example**:
  ```
  You are an On-Chain Activity and Market Impact Correlator. A large transaction of {amount} {symbol} was just detected by Whale Alert involving {from_owner} and {to_owner} (details: {transaction_hash}). Simultaneously, Bitcoin network fees are {current_fee_level} and Binance order book for {symbol}/USDT shows {order_book_state}.
  Analyze:
  1. The potential market impact of this whale transaction.
  2. Correlation with current network conditions.
  3. Any immediate observable effects on Binance's {symbol}/USDT price and order book depth.
  Provide a concise summary and indicate if this warrants a market alert.
  ```
- **Example Query**:
  ```sql
  ```

```
    SELECT * FROM onchain_market_correlator_agent
    WHERE question = 'A $50M BTC transfer from an unknown wallet to Binance occurred. Net-
work fees are high. What's the likely impact on BTC price and Binance order book?';
```

Agent 7: Cross-Asset Arbitrage & Anomaly Detector

- **Purpose**: To identify potential arbitrage opportunities or significant price/metric anomalies by comparing data across CoinMarketCap, Binance, and DeFi platforms.
- **Data Sources**:
  - `coinmarketcap_datasource.listings` (aggregated market prices)
  - `binance_ticker_stats`, `binance_live_trades` (exchange-specific prices)
  - `defillama_yields` (for assets in DeFi pools)
  - `dune_wallet_balances` (potentially to track large holder movements if relevant to anomalies)
- **Prompt Template Example**:
  ```
  You are a Cross-Asset Arbitrage and Anomaly Detector. Compare the price of {symbol} on CoinMarketCap ({cmc_price}) with its current trading price on Binance ({binance_price}). Also, check its TVL trend on DefiLlama if it's a major DeFi asset.
  Identify:
  1. Any significant price discrepancies that might indicate arbitrage opportunities (consider potential fees).
  2. Unusual volume spikes on Binance not reflected in overall market data.
  3. Sudden, unexplained TVL drops in related DeFi protocols.
     Provide a summary of findings and a confidence level for any identified anomaly or opportunity.
     ```
- **Example Query**:
  ```sql
    SELECT * FROM arbitrage_anomaly_agent
    WHERE question = 'Is there a significant price difference for LINK between CoinMarketCap and Binance right now? Any unusual activity?';
  ```

6. Database Schema Updates (PostgreSQL)

New tables will be created in the `crypto_postgres` database. Timestamps should be `TIMESTAMPTZ` for time zone awareness.

6.1. DefiLlama Tables

- **`defillama_protocols`**
  - `id` VARCHAR(255) PRIMARY KEY (DefiLlama slug or ID)
  - `name` VARCHAR(255)
  - `symbol` VARCHAR(50)
  - `category` VARCHAR(100)
  - `chains` JSONB (Array of chains it's on)
  - `tvl` DECIMAL(30, 2)
  - `logo_url` TEXT
  - `url` TEXT
  - `last_updated` TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP

- **`defillama_tvl_history`**
  - `protocol_id` VARCHAR(255) REFERENCES `defillama_protocols` (`id`)
  - `chain` VARCHAR(100) NULL (for chain-specific TVL, NULL for total)
  - `date` DATE
  - `tvl` DECIMAL(30, 2)
  - `last_fetched` TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP
  - PRIMARY KEY (`protocol_id` , `date` , `chain`)
- **`defillama_yields`**
  - `pool_id` VARCHAR(255) PRIMARY KEY (DefiLlama pool ID)
  - `protocol_id` VARCHAR(255) REFERENCES `defillama_protocols` (`id`)
  - `chain` VARCHAR(100)
  - `project` VARCHAR(255)
  - `symbol` VARCHAR(100)
  - `tvl_usd` DECIMAL(30, 2)
  - `apy` DECIMAL(10, 4)
  - `apy_base` DECIMAL(10, 4) NULL
  - `apy_reward` DECIMAL(10, 4) NULL
  - `pool_metadata` JSONB (e.g., tokens in pool)
  - `last_updated` TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP

6.2. Binance Tables

- **`binance_klines`**
  - `symbol` VARCHAR(20)
  - `interval` VARCHAR(5)
  - `open_time` TIMESTAMPTZ
  - `open_price` DECIMAL(20, 8)
  - `high_price` DECIMAL(20, 8)
  - `low_price` DECIMAL(20, 8)
  - `close_price` DECIMAL(20, 8)
  - `volume` DECIMAL(30, 8)
  - `close_time` TIMESTAMPTZ
  - `quote_asset_volume` DECIMAL(30, 8)
  - `number_of_trades` INTEGER
  - `taker_buy_base_asset_volume` DECIMAL(30, 8)
  - `taker_buy_quote_asset_volume` DECIMAL(30, 8)
  - `last_fetched` TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP
  - PRIMARY KEY (`symbol` , `interval` , `open_time`)
- **`binance_live_trades`** (For WebSocket feed)
  - `id` BIGSERIAL PRIMARY KEY
  - `symbol` VARCHAR(20)
  - `trade_id` BIGINT
  - `price` DECIMAL(20, 8)
  - `quantity` DECIMAL(30, 8)

- `quote_quantity` DECIMAL(30, 8)
- `trade_time` TIMESTAMPTZ
- `is_buyer_maker` BOOLEAN
- `is_best_match` BOOLEAN
- `ingested_at` TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP

- **`binance_order_books`** (Snapshot from WebSocket or frequent REST polls)
  - `id` BIGSERIAL PRIMARY KEY
  - `symbol` VARCHAR(20)
  - `last_update_id` BIGINT
  - `bids` JSONB – Array of [price, quantity]
  - `asks` JSONB – Array of [price, quantity]
  - `snapshot_time` TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP
- **`binance_ticker_stats`**
  - `symbol` VARCHAR(20) PRIMARY KEY
  - `price_change` DECIMAL(20, 8)
  - `price_change_percent` DECIMAL(10, 3)
  - `weighted_avg_price` DECIMAL(20, 8)
  - `prev_close_price` DECIMAL(20, 8)
  - `last_price` DECIMAL(20, 8)
  - `last_qty` DECIMAL(30, 8)
  - `bid_price` DECIMAL(20, 8)
  - `ask_price` DECIMAL(20, 8)
  - `open_price` DECIMAL(20, 8)
  - `high_price` DECIMAL(20, 8)
  - `low_price` DECIMAL(20, 8)
  - `volume` DECIMAL(30, 8)
  - `quote_volume` DECIMAL(30, 8)
  - `open_time` TIMESTAMPTZ
  - `close_time` TIMESTAMPTZ
  - `first_id` BIGINT
  - `last_id` BIGINT
  - `count` BIGINT
  - `last_updated` TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP

6.3. Blockchain.com Tables

- **`blockchain_network_stats`**
  - `id` SERIAL PRIMARY KEY
  - `timestamp` TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP
  - `metric_name` VARCHAR(100) – e.g., ‘hash_rate’, ‘difficulty’, ‘mempool_size’
  - `value_numeric` DECIMAL(30, 8) NULL
  - `value_text` TEXT NULL
  - `source_chain` VARCHAR(50) DEFAULT ‘Bitcoin’ – If expanding to other chains via this API
  - UNIQUE (`timestamp` , `metric_name` , `source_chain`)

- **`blockchain_blocks`**
  - `hash` VARCHAR(64) PRIMARY KEY
  - `height` BIGINT UNIQUE
  - `block_time` TIMESTAMPTZ
  - `received_time` TIMESTAMPTZ
  - `size_bytes` INTEGER
  - `num_transactions` INTEGER
  - `difficulty` DECIMAL(30,10)
  - `merkle_root` VARCHAR(64)
  - `previous_block_hash` VARCHAR(64)
  - `nonce` BIGINT
  - `last_fetched` TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP

6.4. Whale Alert Tables

- **`whale_transactions`** (Can serve both live and archive)
  - `id` BIGSERIAL PRIMARY KEY
  - `transaction_hash` VARCHAR(255) UNIQUE
  - `blockchain` VARCHAR(50)
  - `symbol` VARCHAR(50)
  - `timestamp` TIMESTAMPTZ
  - `amount` DECIMAL(38, 18) – Sufficient precision for various cryptos
  - `amount_usd` DECIMAL(30, 2)
  - `from_address` VARCHAR(255)
  - `from_owner_type` VARCHAR(100) NULL – e.g., ‘exchange’, ‘wallet’, ‘unknown’
  - `from_owner_name` VARCHAR(255) NULL – e.g., ‘Binance’, ‘vitalik.eth’
  - `to_address` VARCHAR(255)
  - `to_owner_type` VARCHAR(100) NULL
  - `to_owner_name` VARCHAR(255) NULL
  - `transaction_type` VARCHAR(50) – ‘transfer’, ‘mint’, ‘burn’
  - `details_json` JSONB NULL – For any extra data from Whale Alert
  - `ingested_at` TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP

6.5. Dune Analytics Sim API Tables

- **`dune_wallet_balances`**
  - `id` BIGSERIAL PRIMARY KEY
  - `wallet_address` VARCHAR(255)
  - `chain` VARCHAR(50)
  - `token_address` VARCHAR(255) – ‘native’ for native currency
  - `symbol` VARCHAR(50)
  - `amount` DECIMAL(38, 18) – Raw amount with decimals
  - `decimals` INTEGER
  - `price_usd` DECIMAL(20, 8) NULL
  - `value_usd` DECIMAL(30, 2) NULL
  - `last_fetched` TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP

    - UNIQUE (`wallet_address` , `chain` , `token_address`)
- **`dune_token_metadata`** (If Sim API provides such an endpoint)
    - `chain` VARCHAR(50)
    - `token_address` VARCHAR(255)
    - `name` VARCHAR(255)
    - `symbol` VARCHAR(50)
    - `decimals` INTEGER
    - `logo_url` TEXT NULL
    - `last_fetched` TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP
    - PRIMARY KEY (`chain` , `token_address`)

7. Implementation Phases and Priorities

A phased approach is recommended to manage complexity and deliver value incrementally.

Phase 1: Foundational Data Layers & Core APIs (Estimate: 4-6 Weeks)

- **Priority 1: DefiLlama Integration**
    - Develop/configure MindsDB handler for DefiLlama.
    - Set up PostgreSQL tables: `defillama_protocols` , `defillama_tvl_history` , `defillama_yields` .
    - Implement data ingestion jobs.
    - Dependencies: DefiLlama API access.
- **Priority 2: Blockchain.com Integration**
    - Develop/configure MindsDB handler for Blockchain.com (network stats, block summaries).
    - Set up PostgreSQL tables: `blockchain_network_stats` , `blockchain_blocks` .
    - Implement data ingestion jobs.
    - Dependencies: Blockchain.com API access.
- **Initial Agent Testing**: Test existing agents with new foundational data if relevant, or simple new queries.

Phase 2: Real-Time Exchange & Transaction Monitoring (Estimate: 6-8 Weeks)

- **Priority 1: Binance API Integration**
    - Develop WebSocket client service for live trades and order books (for key pairs like BTC/USDT, ETH/USDT).
    - Set up PostgreSQL staging tables: `binance_live_trades` , `binance_order_books` .
    - Develop/configure MindsDB handler for REST API (Klines, ticker stats).
    - Set up PostgreSQL tables: `binance_klines` , `binance_ticker_stats` .
    - Dependencies: Binance API access.
- **Priority 2: Whale Alert API Integration**
    - Select Whale Alert plan (e.g., Personal or Enterprise REST for initial phase).
    - Develop WebSocket client service for live alerts.
    - Set up PostgreSQL table: `whale_transactions` .
    - Develop/configure MindsDB handler for REST API (if needed for historical lookups).
    - Dependencies: Whale Alert API key and subscription.
- **Agent Development**: Begin development of `On-Chain Activity & Market Impact Correlator` agent.

Phase 3: Advanced On-Chain Analytics & Agent Refinement (Estimate: 4-6 Weeks)

- **Priority 1: Dune Analytics Sim API Integration**
  - Investigate full capabilities and pricing of Dune Sim API.
  - Develop/configure MindsDB handler for Dune Sim API (initially for balances, expand as other relevant endpoints are identified).
  - Set up PostgreSQL tables: `dune_wallet_balances`, (and `dune_token_metadata` if applicable).
  - Dependencies: Dune Sim API key and subscription.
- **Agent Development & Refinement**:
  - Develop `DeFi Opportunity & Risk Analyzer` and `Cross-Asset Arbitrage & Anomaly Detector` agents.
  - Refine all new and existing agents to leverage the full spectrum of available data.
  - Develop new views for complex cross-API queries.

Phase 4: Optimization, Testing, and Deployment (Estimate: 3-4 Weeks)

- **Activities**:
  - End-to-end system testing.
  - Performance optimization (query tuning, indexing, handler efficiency).
  - Rate limit handling refinement.
  - Security review.
  - Documentation updates.
  - Full deployment to production environment.
- **Dependencies**: Completion of all prior phases.

Timeline Summary:
* Total Estimated Time: 17-24 Weeks.
* This timeline is aggressive and assumes dedicated resources. Actual time may vary based on team size, expertise, and unforeseen challenges.

8. Performance and Rate Limiting Considerations

8.1. Performance

- **Data Ingestion**:
  - WebSocket services (Binance, Whale Alert) must be highly performant to handle real-time data streams and write to PostgreSQL efficiently. Consider batch inserts.
  - MindsDB handlers polling REST APIs should be optimized to fetch only necessary data and handle pagination efficiently.
  - Schedule polling jobs carefully to balance data freshness with API load.
- **Database Performance**:
  - Proper indexing on PostgreSQL tables is crucial (as suggested in schema definitions with PRIMARY KEYs and UNIQUE constraints; additional indexes on frequently queried columns like timestamps, foreign keys will be needed).
  - Regular database maintenance (VACUUM, ANALYZE).
  - Partition large tables (e.g., `binance_live_trades`, `whale_transactions`) by time if they grow excessively.

- **Agent Query Performance**:
  - Agents querying multiple large tables can be slow. Pre-aggregate data using MindsDB/PostgreSQL views where possible.
  - Optimize agent prompts to request concise information.
  - Cache agent responses for common queries if applicable.
- **MindsDB Performance**: Monitor MindsDB instance resources (CPU, memory, disk I/O). Scale as needed.

8.2. Rate Limiting Strategy

- **Centralized API Key Management**: Store API keys securely (e.g., Kubernetes secrets, HashiCorp Vault).
- **Understand Limits**: Thoroughly document the rate limits for each API and each plan/key.
- **Distributed Polling**: Stagger polling intervals for different data points/APIs to avoid bursting limits.
- **Exponential Backoff & Retry**: Implement exponential backoff and retry mechanisms in handlers and WebSocket clients for transient errors or rate limit responses (e.g., HTTP 429).
- **Prioritization**: If limits are tight, prioritize fetching more critical data more frequently.
- **Caching**: Cache API responses at the handler level where data doesn't change frequently (e.g., protocol lists, token metadata). MindsDB may offer some caching.
- **Monitoring**: Implement monitoring for API call success/failure rates and HTTP status codes (especially 429 Too Many Requests, 401 Unauthorized, 403 Forbidden). Alert on high error rates.
- **Dedicated Keys/Proxies**: For very high volume APIs, consider using multiple API keys or proxy services if permitted by API T&Cs.

9. Cost Analysis

Estimating costs requires confirming pricing for APIs where it's not explicitly provided in the research.

- **CoinMarketCap**: Existing cost (assume professional tier).
- **TimeGPT, Anthropic, OpenAI**: Existing costs.
- **DefiLlama API**: Assumed free for public data. Costs may arise if a premium/enterprise tier is needed for higher reliability or specific data not in public API.
- **Binance API**: Market data is generally free. No direct API subscription cost for data.
- **Blockchain.com API**: Public data is free. API key for higher limits might have associated costs or be free upon request – needs verification.
- **Whale Alert API**:
  - Personal Plan: $29.95/month.
  - Enterprise REST API: $699/month.
  - Priority Alerts API: $1299/month.
  - **Recommendation**: Start with Personal Plan for WebSocket access during development/initial rollout, or Enterprise REST if historical data and higher REST limits are immediately critical. Evaluate upgrade based on needs.
- **Dune Analytics Sim API**: Pricing not available in research. Needs investigation. Likely a tiered subscription model. This is a significant unknown.
- **Infrastructure Costs**:
  - Increased PostgreSQL storage and I/O.
  - Potentially higher CPU/memory for MindsDB if more models/agents are run concurrently.
  - Resources for new WebSocket client services (e.g., small containerized applications).
  - Estimate: Additional $50-$200/month for cloud resources, depending on scale and efficiency.

- **Development Costs**: Internal team time or contractor costs for developing custom handlers and WebSocket services.

Estimated New Monthly API Costs (excluding Dune Sim API and potential DefiLlama/Blockchain.com premium tiers):

* Whale Alert (starting with Personal Plan): ~$30/month.
* If Whale Alert Enterprise REST is chosen: ~$700/month.

Action Item: Obtain pricing for Dune Sim API and clarify any costs for higher-tier access to DefiLlama and Blockchain.com APIs.

10. Risk Assessment and Mitigation

Risk	Likelihood	Impact	Mitigation Strategy
API Unavailability/ Changes	Medium	High	Implement robust error handling, retries. Monitor API status pages. Have alternative sources for critical data if possible (though difficult for specialized APIs). Design handlers for easy updates.
Rate Limit Exceeded	Medium	Medium	Implement strategies in Sec 8.2. Request higher limits if necessary. Monitor usage closely.
Data Quality/Accuracy Issues	Medium	Medium	Cross-validate data with other sources where possible. Implement data validation checks during ingestion. Report issues to API providers. Be aware of data source methodologies (e.g., DefiLlama adapters).
API Key Compromise	Low	High	Store API keys securely (secrets management). Restrict key permissions if possible. Implement monitoring for unusual API key activity. Rotate keys periodically.
Cost Overruns	Medium	Medium	Monitor API usage and associated costs closely. Set budget alerts. Optimize API calls to stay within cheaper tiers if feasible. Clearly understand pricing models before committing.
Custom Handler Development Complexity	Medium	Medium	Allocate sufficient development time. Use experienced developers. Start

Risk	Likelihood	Impact	Mitigation Strategy
			with simpler handlers and iterate. Leverage MindsDB community/ support.
WebSocket Service Reliability	Medium	Medium	Implement robust connection management, auto-reconnect, and error logging for WebSocket clients. Consider running redundant instances. Use a message queue for decoupling.
Scalability Issues	Medium	Medium	Design for scalability from the start (database, services). Monitor performance metrics and scale resources proactively. Optimize queries and data processing.
Dune Sim API Limitations/Cost	Unknown	High	Prioritize research into Dune Sim API capabilities, limitations, and pricing early in Phase 3. Have contingency plans if it's unsuitable or too expensive (e.g., rely on other on-chain sources or limit scope).

11. Deployment Plan

This plan assumes a containerized deployment environment (e.g., Docker, Kubernetes) as hinted in the `DEVELOPER_TECHNICAL_GUIDE.md`.

1. **Environment Preparation**:
   - Ensure production environment (MindsDB, PostgreSQL, application hosts for WebSocket services) has sufficient resources.
   - Set up secure API key management for production keys.
2. **Database Migration**:
   - Apply new PostgreSQL table schemas to the production database.
   - Migrate any necessary seed data or initial historical data.

3. **Deploy WebSocket Services**:
    - Containerize and deploy the Python services for Binance and Whale Alert WebSockets.
    - Configure them with production API keys and database connections.
    - Monitor their logs and connectivity.
4. **Deploy/Update MindsDB Handlers**:
    - If custom handlers were developed, package and deploy them to the MindsDB environment.
    - Update MindsDB configurations to create/use the new data sources (`CREATE DATABASE ... WITH ENGINE ...`).
5. **Deploy New AI Models & Agents**:
    - Run SQL scripts in MindsDB to create new models (if any specific to new data) and the new agents.
    - Update existing agents if their prompts or data sources change.
6. **Data Ingestion Kick-off**:
    - Enable/start data ingestion jobs for REST-based APIs.
    - Verify data is flowing correctly into PostgreSQL tables.
7. **Dashboard & View Updates**:
    - Update existing dashboards/views or create new ones to incorporate data from the new sources.
8. **Monitoring & Alerting Setup**:
    - Configure monitoring for new services, API call success rates, data freshness, and system resource usage.
    - Set up alerts for critical issues (e.g., API failures, WebSocket disconnects, high error rates).
9. **Phased Rollout (Optional)**:
    - Consider enabling agents or features powered by new APIs incrementally to monitor impact and performance.
10. **Post-Deployment Review**:
    - After a period of stable operation (e.g., 1 week), review performance metrics, costs, and system health. Make adjustments as needed.

This comprehensive design plan provides a roadmap for significantly enhancing the Crypto Intelligence System. Diligent execution of each phase, coupled with proactive risk management and performance monitoring, will be key to a successful expansion.

12. References

- Dune Analytics Sim API Documentation (https://docs.sim.dune.com/)
- Dune Analytics Sim API Quickstart (https://sim.dune.com/)
- Dune Query Engine Documentation (https://docs.dune.com/query-engine/overview)
- Dune API Overview (General Platform) (https://docs.dune.com/api-reference/overview/introduction)
- TokenizedHQ Dune Analytics API Review (https://tokenizedhq.com/dune-analytics-api/)
- DeFiLlama API Documentation (https://defillama.com/docs/api)
- DeFiLlama Adapters GitHub (https://github.com/DefiLlama/DefiLlama-Adapters)
- Binance API Documentation (Market Data) (https://developers.binance.com/docs/binance-spot-api-docs/rest-api/market-data-endpoints)
- Binance Developer Vision - Real-time Data (https://dev.binance.vision/t/accessing-real-time-data-with-binance-api/21669)
- Blockchain.com API Main Page (https://www.blockchain.com/api)
- Blockchain.com Charts - Hash Rate (https://www.blockchain.com/charts/hash-rate)
- Whale Alert API Documentation (https://developer.whale-alert.io/documentation/)

- Whale Alert Main Website (https://whale-alert.io/)
- Whale Alert Pricing/Plans (Inferred from developer.whale-alert.io snippets) (https://developer.whale-alert.io/)