# Project Plan: Financial Data Harmonization & Agentic MCP Layer (Hybrid Strategy)

**Objective:** Harmonize Trading, Risk, and Market Data from legacy (Sybase) and modern (Postgres) sources into Snowflake using a **Daily Batch (T-1)** foundation for deep analytics, augmented by **Direct-to-API** tools for real-time intraday visibility.

Timeline: 12 Weeks

Owner: Senior Tech Lead

## Phase 1: Ingestion & Architecture (Weeks 1-3)

*Goal: Establish reliable "Nightly Dump" pipes from source to Snowflake "Bronze" layer.*

### 1.1 Ingestion Strategy (Batch ELT)

We will use Snowflake's efficient COPY INTO command for bulk loading. This is 5-10x cheaper than streaming.

* **The "Landing Zone":**
  + Data Engineering to provision an S3/Azure Blob container.
  + Folder Structure: /raw/{source}/{entity}/YYYY-MM-DD/file.parquet
* **Sybase ASE (Legacy):**
  + *Method:* Scheduled Cron job triggers bcp (Bulk Copy Program) or SQL export to the Landing Zone every night at 01:00 UTC.
  + *Content:* Full snapshot of Trades (incremental since last run) and Positions (full dump).
* **PostgreSQL (Modern):**
  + *Method:* Scheduled export script to Landing Zone.
  + *Content:* Daily dump of Market\_Data\_History and Client\_Static.

### 1.2 Snowflake Layering (The Medallion Architecture)

* **Bronze Layer (Raw):**
  + Tables partitioned by load\_date.
  + Data is "Appended" daily. We do not overwrite; we keep a full history of dumps for auditability.
* **Silver Layer (Cleaned):**
  + Deduplicated via QUALIFY window functions.
  + *Normalization:* Map Sybase 1/0 booleans to Postgres True/False.
* **Gold Layer (The Agent's Truth):**
  + Star Schema optimized for read-heavy Agent queries.

## Phase 2: The Unified Data Model & History (Weeks 4-7)

*Goal: Handle the "Time Travel" aspect of financial data using dbt and ensure Logic Parity.*

Since we are T-1, the ability to look back at "Risk as of Last Thursday" is critical. We will use two distinct strategies for history.

### 2.1 Strategy A: Slowly Changing Dimensions (Client & Product)

For data that changes rarely (SCD Type 2).

* **Mechanism:** dbt snapshots.
* **Table Structure:** dim\_client\_history will have valid\_from and valid\_to columns.

### 2.2 Strategy B: Periodic Snapshots (Positions & Risk)

For data that changes every day.

* **Mechanism:** Daily Partitioning.
* **Table Structure:** fact\_positions\_daily will have a snapshot\_date column.

### 2.3 The Serving Layer (Critical for Agents)

To prevent the Agent from querying expired data by mistake, we create strictly filtered Views.

* **v\_dim\_client\_current:** SELECT \* FROM dim\_client\_history WHERE valid\_to IS NULL
* **v\_fact\_risk\_latest:** SELECT \* FROM fact\_risk\_daily WHERE snapshot\_date = (SELECT MAX(snapshot\_date) FROM fact\_risk\_daily)

### 2.4 Logic Parity & Validation (The "Trust" Phase)

*Addressed CTO Probe: "How do we know the math matches Sybase?"*

* **Parallel Run:** For Weeks 6-7, we run the legacy Sybase Risk Report and the new Snowflake dbt model side-by-side.
* **Automated Reconciliation:**
  + Build a Python script (check\_parity.py) that compares the Sybase output CSV vs. Snowflake Gold tables.
  + **Tolerance:** Flag any PnL discrepancy > $0.01 or Risk > 0.1%.
  + **Sign-off:** The Risk Desk Head must sign off on the parity report before Phase 3 begins.

## Phase 3: The Hybrid MCP Server Build (Weeks 8-10)

*Goal: Build an Agent that is smart about History (Snowflake) and aware of the Now (API).*

### 3.1 Architecture: The "Hybrid" Bridge

Addressed CTO Probe: "Traders need real-time data."

The MCP Server will route queries based on "Temporal Intent":

1. **Deep Analysis/History:** Routes to Snowflake (T-1 Data).
2. **Live Checks:** Routes to a direct API (OMS/Bloomberg) for *current* price/position.

### 3.2 Tool Definitions

The Agent will have access to these specific Python functions:

#### Tool A: get\_client\_profile (Snowflake)

* **Target:** v\_dim\_client\_current
* **Logic:** Fast lookup of static data.

#### Tool B: analyze\_historical\_risk (Snowflake)

* **Target:** fact\_risk\_daily
* **Logic:** "How did exposure change last week?"

#### Tool C: get\_live\_market\_data (Direct API)

* **Source:** Direct HTTP call to Market Data Provider (e.g., Refinitiv/Bloomberg API) or internal OMS Cache.
* **Usage:** *Only* for "What is the price of AAPL right now?"
* **Note:** This bypasses Snowflake entirely to solve the T-1 latency issue.

### 3.3 Governance & Guardrails

*Addressed CTO Probe: "Preventing Hallucinations."*

* **Code-Level Enforcement:** The Agent does *not* write SQL to choose tables. It calls a Python function.
  + The Python function get\_current\_risk() is **hardcoded** to query v\_fact\_risk\_latest.
  + The Agent physically *cannot* query the history table unless it calls the specific get\_historical\_risk(date) tool.
* **System Prompt:** "You are a Risk Assistant. For current prices, ALWAYS use the get\_live\_market\_data tool. For trend analysis, use Snowflake tools."

## Phase 4: Deployment (Weeks 11-12)

* **Week 11:** UAT with Risk Managers using the Hybrid Agent.
* **Week 12:** Production Deploy (Read-Only).

## Phase 5: Strategic Roadmap (The Future)

*Addressed CTO Probe: "How does this retire Sybase?"*

### 5.1 The "Strangler Fig" Pattern

By moving the **Consumption Layer** (Agents/Reports) to Snowflake/MCP, we decouple the traders from the underlying Sybase database.

1. **Today:** Sybase feeds Snowflake. Agents read Snowflake.
2. **Tomorrow:** We replace Sybase with a new OMS. The new OMS feeds Snowflake.
3. **Result:** We update the **Silver Layer** transformation logic to map the New OMS to the existing Gold Schema. The Agent (and the traders) never notice the switch. Sybase can then be decommissioned safely.

### 5.2 Path to Real-Time Snowflake

Once the batch process is stable, we can upgrade **Phase 1.1** from "Daily Batch" to "Micro-batch (15 mins)" or "Snowpipe Streaming" without changing the downstream Agent tools, simply by increasing the ingestion frequency.