## Project Documentation: Cross-Sell Prediction Model - Data Preparation

### 1. Project Objective

The primary goal of this project is to build a machine learning model that can predict the **next product a new client is most likely to purchase**. The model will be used to provide targeted cross-sell recommendations, improving marketing efficiency and deepening client relationships.

The first and most critical phase of this project was to perform the necessary data engineering to transform a complex, transactional database into a single, clean, and logically sound dataset suitable for training a predictive model.

### 2. Overall Methodology

The core strategy was to create a **unified, client-centric view**, where each row represents a single client. This involved:

- Understanding the schema and relationships of ten distinct tables.
- Using SQL with Common Table Expressions (CTEs) to aggregate and organize data.
- Joining these disparate sources into a single feature matrix.
- Systematically identifying and resolving data quality issues, logical fallacies, and sources of machine learning bias like target leakage.

---

### 3. Iteration History and Query Development

The final query was developed through a series of iterative refinements, with each version addressing a critical flaw discovered during validation.

- **Goal:** Combine all known tables to create a comprehensive 360-degree profile of each client.
- **Logic:** The initial query used LEFT JOINs to merge tables like pcpg_retention, client_metrics, rpt_agents, activities, and opportunities. It used CTEs to pre-aggregate multi-row data (like agent snapshots and client activities) into single values.
- **Problems Discovered:**
    1. **Typographical Errors:** Initial runs failed due to simple typos in column names (e.g., we_ vs. wc_, _o vs. _c).
    2. **SQL Dialect Issues:** The query failed due to syntax differences in the data lake environment (e.g., DATE_SUB was not supported and was replaced with ADD_MONTHS).

    —-----------------------------------------------

- **Goal:** Introduce a target variable to predict.
- **Logic:** A new CTE, NextProductTarget, was created to identify the second policy (policy_rank = 2) a client ever purchased.
- **Problem Discovered (Critical Insight): Feature columns and the target column used different product categories**.
    - The features (e.g., has_life_insurance) came from the retention table's simplified categories.
    - The target (e.g., 'Annuities') came from the client_metrics table's detailed categories.
    - This mismatch would have made it impossible for the model to learn correctly.

—---------------------------------------------

- **Goal:** Fix the feature/target mismatch.
- **Logic:** We abandoned the has_ flags from the retention table. A new CTE, ClientProductPortfolio, was created to count every client's policies, using the exact same product categories from wti_lob_txt that were used for the target variable.
- **Problem Discovered (Critical Insight):** You found a logical inconsistency where a client could have the same product for their initial_product_purchased and next_product_purchased. This was traced back to policies having the **exact same register_date**, causing the ranking function to be inconsistent.
- **Solution:** A **tie-breaker** (policy_no) was added to the ORDER BY clause of the ranking function to ensure a consistent, deterministic order every time.

—-----------------------------------------------

- **Goal:** Address why a single client was still appearing in dozens of rows.
- **Problem Discovered (Critical Insight):** Through validation, we discovered that the source tables (client_metrics and pcpg_retention) were not one-row-per-client or even one-row-per-policy. They were **snapshot tables** containing multiple records over time. This caused joins to "fan out," creating massive duplication.
- **Solution:**
    1. A DistinctPolicies CTE was created at the beginning to create a clean, deduplicated source of truth for all policy-related calculations.
    2. A LatestRetentionSnapshot CTE was created to select only the single, most recent record for each client from the retention table, preventing it from causing duplicates.

—-------------------------------------------------

- **Goal:** Eliminate the final, most subtle form of **target leakage**.
- **Problem Discovered (Critical Insight):** Identified that even with a correct target, our features (like AUM, age, policy counts) were being calculated based on the client's state **today**, not at the time the prediction should have been made.
- **Solution:** The entire query was re-architected to create a true **point-in-time** dataset.

1. A FeaturesAtFirstPurchase CTE was built to calculate all features (age, AUM, portfolio) as they were at the moment of the client's very first purchase.
2. The target remained the product category of the second purchase.
3. You requested adding the purchase dates and the time between them as features, which was a valuable addition.
4. The final JOIN structure was made more robust to prevent NULL primary keys.

—----------------------------------------

Data Visualizations to get a deeper understanding of the data and evaluate its suitability and competence for our ML use case were periodically done.

—----------------------------------------

But, we discovered that the aum_band, etc, in the client_metrics tables are for recent business years - no data is present (as per observation in client_metrics, pcpg_retention, transactions tables) for the register_date of the second policy purchased.

---