- users: id, email, password_hash, created_at (PK: id) (UNIQUE: email) (FD: id → email, password_hash, created_at; email → id, password_hash, created_at)
- accounts: id, account_type, name, starting_cash, created_at (PK: id) (FD: id → account_type, name, starting_cash, created_at)
- account_memberships: account_id, user_id, role (PK: account_id, user_id) (FK: account_id → accounts.id; user_id → users.id) (FD: (account_id, user_id) → role)
- tickers: symbol, name, asset_type (PK: symbol) (FD: symbol → name, asset_type)
- price_bars: ticker, time, open, high, low, close, volume, source (REAL from Yahoo Finance; SIM from generator) (PK: ticker, time) (FK: ticker → tickers.symbol) (FD: (ticker, time) → open, high, low, close, volume, source)
- groups: id, name, created_at, created_by (FK → users.id) (PK: id) (FK: created_by → users.id) (FD: id → name, created_at, created_by)
- group_memberships: group_id, user_id, role(owner|manager|member|viewer), added_at (PK: group_id, user_id) (FK: group_id → groups.id; user_id → users.id) (FD: (group_id, user_id) → role, added_at)
- news_articles: id, published_at, source, title, url, sentiment, impact_tags[] (PK: id) (FD: id → published_at, source, title, url, sentiment, impact_tags[])
- news_ticker_map: article_id, ticker (PK: article_id, ticker) (FK: article_id → news_articles.id; ticker → tickers.symbol) (FD: (article_id, ticker) → ∅) // key-only bridge
- user_news: user_id, ticker (PK: user_id, ticker) (FK: user_id → users.id; ticker → tickers.symbol) (FD: (user_id, ticker) → ∅) // key-only link (treat id as user_id)
- Transactions: id, account_id, time, ticker, side(BUY/SELL), qty, price, kind(ORDER|FILL), status, requested_by, approved_by NULLABLE (PK: id) (FK: account_id → accounts.id; ticker → tickers.symbol; requested_by → users.id; approved_by → users.id (nullable)) (FD: id → account_id, time, ticker, side, qty, price, kind, status, requested_by, approved_by)

Explanations for each entity:

- Users
  - **Assumptions:** One row per human user. email is unique; password_hash or external SSO token is stored; created_at is system-generated.
  - **Why entity (not attribute elsewhere):** Users are the principal actors and are referenced by multiple tables (account_memberships, group_memberships, transactions.requested_by/approved_by, user_news). They have their own lifecycle and constraints.
- Accounts:

    - **Assumptions:** Represents a paper-trading account/portfolio (e.g., “Personal”, “Group A”). Holds its own starting_cash, account_type, name, created_at.
    - **Why entity:** Accounts own transactions and cash/position state. They are not attributes of a user because multiple users can belong to the same account with different roles.
- Account_memberships:
    - **Assumptions:** M:N (many to none) between users and accounts. role governs permissions (owner/trader/risk/viewer). Composite PK (account_id, user_id) prevents duplicates.
    - **Why entity:** Membership depends on **both** user and account and carries its own attribute (role). Modeling as an attribute of either parent would cause redundancy and update anomalies.
- Groups:
    - **Assumptions:** Social/organizational construct (study group, club, class team). created_by references to the user who created the group.
    - **Why entity:** Groups have their own lifecycle and membership separate from trading accounts (you can be in a group without sharing an account).
- Group_memberships:
    - **Assumptions:** M:N between users and groups. Stores role (owner/manager/member/viewer) and added_at. Composite PK (group_id, user_id).
    - **Why entity:** Same reason as account_memberships: independent attributes and M:N cardinality.
- Tickers
    - **Assumptions:** Security master record per tradable symbol. symbol is the natural PK. asset_type can be stock|crypto|…; names are not unique across all markets, but symbol is unique in our universe.
    - **Why entity:** Referenced by price_bars, transactions, user_news, and news_ticker_map. Keeping a single source of truth avoids repeating symbol metadata.
- Price_bars:
    - **Assumptions:** Time-series market data; one row per (ticker, time) with OHLCV. source indicates REAL vs SIM. The composite PK (ticker, time) is unique.
    - **Why entity:** Very high cardinality, independent ingestion cadence, and different lifecycle/retention than tickers or transactions. Not an attribute of tickers because it’s a repeating time-series.
- News_articles:

    - **Assumptions:** Curated news with published_at, source, title, url. sentiment is optional; impact_tags[] is a small array/string of tags (kept denormalized for simplicity/perf at this stage).
    - **Why entity:** Articles link to many tickers and are consumed by many users. They carry their own metadata and lifecycle separate from symbols.
- News_ticker_map:
    - **Assumptions:** M:N between news and tickers (one article can affect many tickers; one ticker can be in many articles). Composite PK (article_id, ticker).
    - **Why entity:** Pure relationship table, prevents duplication of article rows per ticker and supports efficient joins.
- User_news:
    - **Assumptions:** M:N between users and tickers for “followed” symbols. Composite PK (user_id, ticker).
    - **Why entity:** It captures a user-specific preference over tickers; not an attribute of users or tickers because it depends on both and is many-valued.
- Transactions:
    - **Assumptions:** Immutable execution log: one row per trade event. Required fields: account_id, ticker, time, side (BUY/SELL), qty, price, kind (ORDER|FILL), status. requested_by is required (who placed it); approved_by is nullable (some accounts may require no approval).
    - **Why entity:** Central event stream with auditability and foreign keys to users/accounts/tickers. Not an attribute on account because there are many per account and they have their own lifecycle.

Explanations for each relationship:

- **Users ↔ Accounts** via **Account_Memberships**
    - **Cardinality:** M:N realized as a bridge.
    - **Why:** A user can participate in many accounts (personal plus team funds); an account can have many users with roles. Modeling directly as attributes would break normalization and permission flexibility.
- **Users ↔ Groups** via **Group_Memberships**
    - **Cardinality:** M:N.
    - **Why:** A user can join many groups; groups have many members; membership carries role and added_at.
- **Users → Groups (created_by)**
    - **Cardinality:** 1:M (a user can create many groups; a group is created by exactly one user).
    - **Why:** Ownership/audit trail requirement.

- **Accounts → Transactions**
    - **Cardinality:** 1:M (each transaction belongs to exactly one account; an account has many transactions).
    - **Why:** All orders/fills execute within an account context (cash/positions).
- **Users → Transactions (requested_by / approved_by)**
    - **Cardinality:** 1:M from Users to Transactions for each role. approved_by is optional (0 or 1 approving user).
    - **Why:** Auditability: who placed and who approved (if policy requires approval). Keeps separation of duties.
- **Tickers → Transactions**
    - **Cardinality:** 1:M (a transaction references exactly one ticker; a ticker appears in many transactions).
    - **Why:** Each trade is for one instrument.
- **Tickers → Price_Bars**
    - **Cardinality:** 1:M with composite uniqueness (ticker, time).
    - **Why:** One symbol has many time-stamped bars; a bar belongs to exactly one symbol.
- **News_Articles ↔ Tickers** via **News_Ticker_Map**
    - **Cardinality:** M:N.
    - **Why:** One article may impact multiple tickers; each ticker can be referenced by many articles.
- **Users ↔ Tickers** via **User_News** (watchlist)
    - **Cardinality:** M:N.
    - **Why:** Each user can follow many tickers; each ticker can be followed by many users.