Conventions

- Current_date = 30 September 2025.
- Snapshot_start_date = 30 September 2024.

household (SCD2)

Grain: one row per Household_key.

Columns

- Household_key (surrogate SCD2 key)
- Household_id (natural/business id starts at 1 and increments by 1). Target count of household_id: 45,000.
- household_tenure (integer). Years of tenure. Sample uniformly at random from 1 to 40.
- household_registration_type (text: 'Individual','Joint','Trust','Institutional'). Target distribution: Individual 64%, Joint 22%, Trust 9%, Institutional 5%.
- household_registration_date (date). Current_date − household_tenure ± U(−180, +180) days.
- household_segment (text: 'Self-Directed','Advice-Seeking','Discretionary Managed','Retirement Income','Business/Institutional','Active Trader'). Target distribution: Self‑Directed 15%, Advice‑Seeking 25%, Discretionary Managed 30%, Retirement Income 15%, Business/Institutional 10%, Active Trader 5%.
- household_status (text: 'Active','Terminated'). Target distribution: Active 90%, Terminated 10%.
- household_advisor_id (fk → advisors.advisor_id). Sample uniformly at random one advisor_id.
- from_date (date). Defines SCD type 2 window (inclusive).
- to_date (date). Defines SCD type 2 window (exclusive). ~12% switch to household_status='Terminated' within last 3 months from Current_date. Emit prior row with to_date = change_date; new row with from_date = change_date, to_date = '9999‑12‑31'.

Indexes / guards

```
CREATE UNIQUE INDEX ux_household_current ON household(household_id)

WHERE to_date = DATE '9999-12-31';

CREATE INDEX ix_household_id_window ON household(household_id, from_date, to_date);
```

advisors (SCD2)

Grain: one row per advisor advisor_key.

Columns

- advisor_key (surrogate SCD2 key)
- advisor_id (natural/business id starts at 1 and increments by 1). Target count of advisor_id: 500.
- advisor_tenure (integer). Years of tenure. Sampled uniformly at random from 1 to 40.
- firm_name (text). generate fake names ending with LLC; patterns like: [Summit|Harbor|Granite|Cedar|Crescent|Atlas|Pioneer|Ridge|Oak|River] [Capital|Advisors|Partners|Wealth|Financial] LLC.
- firm_affiliation_model (text: 'RIA','Hybrid RIA','Broker-Dealer W-2','Independent BD','Bank/Trust','Insurance BD','Wirehouse'). Target distribution: RIA 35%, Hybrid RIA 20%, Independent BD 18%, Broker‑Dealer W‑2 12%, Wirehouse 7%, Bank/Trust 5%, Insurance BD 3%.
- advisor_role (text: 'Lead Advisor','Associate Advisor','Relationship Manager','Portfolio Manager','Client Service Associate'). Target distribution: Lead Advisor 45%, Associate Advisor 20%, Relationship Manager 15%, Portfolio Manager 10%, Client Service Associate 10%.
- advisor_status (text: 'Active','Terminated'). Target distribution: Active 92%, Terminated 8%.
- practice_segment (text: 'Solo Practice','Small Team','Ensemble','Enterprise'). Target distribution: Solo Practice 22%, Small Team 36%, Ensemble 28%, Enterprise 14%.
- from_date (date). Defines SCD type 2 window (inclusive).
- to_date (date default '9999-12-31'). Defines SCD type 2 window (exclusive). ~10% receive advisor_status change = ‘Terminated’ in last 24 months). Emit prior row with to_date = change_date; new row with from_date = change_date, to_date = '9999‑12‑31'.

Indexes / guards

```
CREATE UNIQUE INDEX ux_advisors_current ON advisors(advisor_id)

WHERE to_date = DATE '9999-12-31';

CREATE INDEX ix_advisor_id_window ON advisors(advisor_id, from_date, to_date);
```

business_line

Grain: one row per business_line_key.

Columns

- business_line_key (surrogate key)
- business_line_name (text:'Managed Portfolio',Separately Managed Account,'Mutual Fund Wrap','Annuity','Cash')

account (SCD2)

Grain: one row per account account_key.

Columns

- account_key (surrogate SCD2 key)
- account_id (natural/business id starts at 1 and increments by 1). Target count of account_id: 72,000.
- advisor_key (fk → advisors.advisor_key). Accounts per advisor: min 15; median 145; max 400. Sampled uniformly at random.
- household_key. Lookup household_id from table households based on household_advisor_id.
- business_line_key. % of accounts by business line: Managed Portfolio 45%, Separately Managed Account 18%, Mutual Fund Wrap 25%, Annuity 6%, Cash 6%.
- account_type (text: 'Taxable','IRA','401k','Trust','Custody'). Target distribution: Taxable 55%, IRA 22%, 401k 10%, Trust 10%, Custody 3%.
- account_custodian (text: 'Schwab','Fidelity','Pershing','In-House','BankTrust'). Target distribution: Schwab 45%, Fidelity 30%, Pershing 15%, BankTrust 6%, In-House 4%.
- opened_date (date). Between household_registration_date and current_date.
- account_status (text: 'Open','Closed'). Target distribution: Open 88%, Closed 12%.
- closed_date (date, nullable). If account_status is Closed, set closed_date > opened_date.If account_status is Open, closed_date is null.
- account_risk_profile (text: 'Conservative','Moderate','Aggressive'). Sample uniformly at random.
- from_date (date)
- to_date (date).Defines SCD type 2 window (exclusive). When account_status changes, emit prior row with to_date = change_date; new row with from_date = change_date, to_date = '9999-12-31'.

Indexes / guards

```
CREATE UNIQUE INDEX ux_accounts_current ON accounts(account_id)

WHERE to_date = DATE '9999-12-31';

CREATE INDEX ix_account_id_window ON accounts(account_id, from_date, to_date);

CREATE INDEX ix_accounts_household ON accounts(household_id);

CREATE INDEX ix_accounts_advisor   ON accounts(advisor_id);
```

product

Grain: one row per product_id.

Columns

- product_id (natural/business id starts at 1 and increments by 1). Target count of product_id: 350.
- asset_category (text: 'Equity','Fixed Income','Multi-Asset','Cash'). % of product_id by asset_category: Equity 50%, Fixed Income 35%, Multi-Asset 12%, Cash 3%.
- asset_subcategory (text: ‘Common stock’, ‘Preferred Stock’, ‘Equity Mutual Fund’, ‘Balanced Fund (60/40)’ , ‘Target-Date Fund’, ‘U.S. Treasury Bill’, ‘U.S. Treasury Note’, ’Investment-Grade Corporate Bond ’, ‘Municipal Bond’,’Money Market Fund’). % of product_id by asset_subcategory: Common Stock 20%, Preferred Stock 5%, Equity Mutual Fund 25%, Balanced Fund (60/40) 7%, Target-Date Fund 5%, U.S. Treasury Bill 4%, U.S. Treasury Note 7%, Investment-Grade Corporate Bond 12%, Municipal Bond 12%, Money Market Fund 3%.
- product_line (text: Mutual Fund, ETF, Separately Managed Account Strategy, Annuity Contract, Money Market). % of product_id by product_line: 40% Mutual Fund, 25% ETF, 20% SMA Strategy, 10% Annuity Contract, 5% Money Market.
- product_name (text).

tier_fee

Columns

- Business_line_key.
- tier_min_aum.
- tier_max_aum.
- tier_fee [%].

Values by business line:

Managed Portfolio

- $0 – $1M → 90 bps
- $1M – $5M → 75 bps
- $5M+ → 55 bps

Separately Managed Account (SMA)

- $0 – $1M → 110 bps
- $1M – $5M → 90 bps
- $5M+ → 70 bps.

Mutual Fund Wrap

- $0 – $1M → 75 bps
- $1M – $5M → 60 bps
- $5M+ → 45 bps.

Annuity

- $0+ → 25 bps

Cash (sweep programs are low and often flat/near-flat)

- $0 – $1M → 10 bps
- $1M+ → 5 bps

advisor_payout_rate

- firm_affiliation_model. List all distinct firm_affiliation_models.
- advisor_payout_rate. Values by firm_affiliation_model: RIA: 78%, Hybrid RIA 70%, Independent BD 85%, Broker-Dealer W-2 45%, Wirehouse 42%, Bank/Trust 35%, Insurance BD 75%.

fact_account_initial_assets

- account_key. (fk → account.account_key). All accounts opened at snapshot_date.
- account_initial_assets:
    - If account_type is Taxable, normal sampling with median $120k, minimum $10k, standard deviation $60k.
    - If account_type is IRA, normal sampling with median $150k, minimum $5k, standard deviation $75k.
    - If account_type is 401k, normal sampling with median $80k, minimum $2k, standard deviation $40k.
    - If account_type is Trust, normal sampling with median $400k, minimum $100k, standard deviation $250k.
    - If account_type is Custody, normal sampling with median $600k, minimum $250k, standard deviation $400k.

fact_account_monthly

This table should be generated by looping through dates first.

- snapshot_date (date). EOM dates between snapshot_start_date and current_date.
- account_key (fk → account.account_key). All accounts opened at snapshot_date.
- account_monthly_return (%). See below calculation method.
- account_net_flow. See below calculation method.
- account_assets_previous_month. If snapshot_date is snapshot_start_date, equals account_initial_assets (retrieved from table fact_account_initial_assets by joining on account_key). If snapshot_date is greater than snapshot_start_date, it is equal to account_assets at previous snapshot_date for the same account_id.
- account_assets. Equals to account_assets_previous_month x (1+ account_monthly_return) + account_net_flow.
- advisor_key. Retrieved via table account -> advisor_key.
- household_key. Retrieved via table account -> household_key.
- business_line_key. Retrieved via table account -> business_line_key.

How to calculate account_monthly_return

- If snapshot_date is snapshot_start_date, account_monthly_return is 0%.
- If snapshot_date is greater than snapshot_start_date:

account_monthly_return [%] = base_return[%] + noise [%]. Limited at +/- 12%.

Base_return [%] =

  - If account_risk_profile is Conservative, normal sampling with median 0.30% and standard deviation 1.0%.
  - If account_risk_profile is Moderate, normal sampling with median 0.55% and standard deviation 2.0%.
  - If account_risk_profile is Aggressive, normal sampling with median 0.80% and standard deviation 3.5%.

Noise [%] = Normal sampling with median 0.25% and standard deviation 0.2%. minimum -0.5%, maximum 0.5%.

How to calculate account_net_flow

account_net_flow = percentage[%] * account_assets_previous_month.

Percentage[%] is:

- If account_type is 401k, normal sampling with median 0.7% and standard deviation 0.3%.
- If account_type is IRA, normal sampling with median 0.2% and standard deviation 0.4%.
- If account_type is Taxable, normal sampling with median 0.05% and standard deviation 0.8%.
- If account_type is Trust, normal sampling with median -0.25% and standard deviation 0.35%.
- If account_type is Custody, normal sampling with median 0% and standard deviation 6%.

fact_account_product_monthly

(distinct pairs of snapshot_date and account_key taken from fact_account_monthly).

- snapshot_date (date). EOM dates between snapshot_start_date and current_date.
- account_key (fk → account.account_key). All accounts opened at snapshot_date.
- Product_id. See below calculation method.
- product_allocation_pct. See below calculation method.

How to calculate product_id and product_allocation_pct:

Step 1: For every distinct account_key, determine k as the number of products per account. To do so, sample uniformly at random k between 2 and 5.
Step 2: Determine business_line for the account (retrieve business_line_name from business_line_table via business_line_key).

Step 3: For every account_key, loop from 1 to k and based on the business line, draw an asset_category from a categorical distribution with probabilities given by these business-line weights:

- For business line “Managed Portfolio”: Equity 45%, Fixed Income 35%, Multi-Asset 18%, Cash 2%
- For business line “Separately Managed Account Strategy”: Equity 65%, Fixed Income 20%, Multi-Asset 13%, Cash 2%
- For business line “Mutual Fund Wrap”: Equity 55%, Fixed Income 30%, Multi-Asset 13%, Cash 2%
- For business line “Annuity Program”: Multi-Asset 70%, Fixed Income 30%, Equity 0%, Cash 0%
- For the business line “Cash Program”: Cash 100%.

Step 4: Once the business line is sampled, sample uniformly at random one product_id from this category, using the product table.
Step 5: Sample a random weight (product_allocation_pct) to this product_id from 20 to 100. If the product_id is the last one, calculate the product_allocation_pct as being 100 minus the sum of product_allocation_pct for the rest of products in the same account_key.

fact_household_monthly

(fact_account_monthly grouped by snapshot_date, household_key)

- snapshot_date. All values from fact_account_monthly.
- household_key. All values from fact_account_monthly.
- household_assets. Sum of account_assets.
- Asset_range_bucket (text). Depending on household_assets:
  - $0 – $100k
  - $100k – $250k
  - $250k – $500k
  - $500k – $1M
  - $1 – $5M
  - $5M – $10M
  - $10M+
- high_net_worth_flag. If household_assets >= $1M, set true otherwise false.
- household_net_flow. Sum of account_net_flow.

fact_revenue_monthly

Based on fact_account_monthly.

- snapshot_date (date). 1:1 from fact_account_monthly.

- account_key. 1:1 from fact_account_monthly.
- advisor_key. 1:1 from fact_account_monthly.
- household_key. 1:1 from fact_account_monthly.
- business_line_key. 1:1 from fact_account_monthly.
- account_assets.1:1 from fact_account_monthly.
- fee_percentage. Retrieve tier_fee column from table tier_fee by joining on business_line_key and tier_min_aum <= account_assets <= tier_max_aum.
- gross_fee_amount = account_assets x fee_percentage.
- third_party_fee. See below calculation method.
- advisor_payout_rate. Retrieve advisor_payout_rate column from table advisor_payout_rate by joining on firm_affiliation_model using advisor table.
- advisor_payout_amount = (gross_fee_amount - third_party_fee) x advisor_payout_rate.
- net_revenue = gross_fee_amount - third_party_fee - advisor_payout_amount.

<u>How to calculate third_party_fee</u>

third_part_fee = percentage[%] * gross_fee_amount.

Percentage[%] is sampled normally with a median of 10% and a standard deviation of 5%.

transactions

many millions of rows (needs to prove query optimization , filtering capacities).

- transaction_id,
- advisor_key,
- account_key,
- household_key,
- business_line_key,
- product_id,
- transaction_date,
- gross_revenue,
- revenue_fee,
- third_party_fee,
- transaction_type (deposit/withdrawal/fee)

fact_customer_feedback

Grain: one feedback per (household, advisor, date).

Columns

- feedback_date (date). Date range is maximum the current_date and minimum Jauary 1st in the year before current_date. In this date interval range, sample one date randomly.
- feedback_id (natural/business id starts at 1 and increments by 1). Target count: 1k feedback_id per month.
- household_key (fk → household.household_key). Randomly sampled household_key where household_status = ‘Active’ and household.from_date >= feedback_date.
- advisor_key (fk → advisors.advisor_key). 1) Retrieve primary_advisor_id from household table; 2) retrieve advisor_key from advisor table joining on primary_advisor_id <-> advisor_id where advisor.from_date <= feedback_date < advisor.to_date.
- feedback_text (text; ≤ 200 chars / ≤ 2 sentences). Make them up.
- satisfaction_score (integer 0–100). Sampled normally with a median of 90.

Indexes:

```
CREATE INDEX ix_fb_house_date ON customer_feedback(household_key, feedback_date);

CREATE INDEX ix_fb_adv_date   ON customer_feedback(advisor_key, feedback_date);
```

date

Grain: one row per calendar day (≥ last 10y + next 2y).

Columns

- calendar_day (date, pk)
- month_name (text). Derived from calendar_day.
- month (int). Derived from calendar_day.
- day_of_month (int). Derived from calendar_day.
- month_start_date (date). Derived from calendar_day.
- month_end_date (date). Derived from calendar_day.
- quarter (int). Derived from calendar_day.
- quarter_name (text). Derived from calendar_day.
- quarter_start_date (date). Derived from calendar_day.
- quarter_end_date (date). Derived from calendar_day.
- year (int). Derived from calendar_day.
- is_weekend (boolean). Derived from calendar_day.

Data Integrity checks

Table: household

Primary Key & Uniqueness

- **Check 1.1**: No duplicated household_key values
- **Check 1.2**: Exactly one record per household_id where to_date = '9999-12-31'
- **Check 1.3**: Target count: Exactly 45,000 distinct household_id values

Date Integrity

- **Check 1.4**: No conflicting dates: from_date < to_date for all records
- **Check 1.5**: No gaps in SCD2 history: For each household_id, next record's from_date = previous record's to_date
- **Check 1.6**: household_registration_date should align with tenure calculation: household_registration_date ≈ current_date - (household_tenure * 365) ± 180 days

Referential Integrity

- **Check 1.7**: All household_advisor_id exist in advisors.advisor_id
- **Check 1.8**: Referenced advisor must be active at time of household registration

Business Rules

- **Check 1.9**: household_tenure is between 1 and 40 years
- **Check 1.10**: Terminated households should have to_date != '9999-12-31'

Table: advisors

Primary Key & Uniqueness

- **Check 2.1**: No duplicated advisor_key values
- **Check 2.2**: Exactly one record per advisor_id where to_date = '9999-12-31'
- **Check 2.3**: Target count: Exactly 500 distinct advisor_id values

Date Integrity

- **Check 2.4**: No conflicting dates: from_date < to_date for all records
- **Check 2.5**: No gaps in SCD2 history: For each advisor_id, next record's from_date = previous record's to_date

Business Rules

- **Check 2.6**: advisor_tenure is between 1 and 40 years
- **Check 2.7**: Terminated advisors should have to_date != '9999-12-31'

Table: account

Primary Key & Uniqueness

- **Check 3.1**: No duplicated account_key values
- **Check 3.2**: Exactly one record per account_id where to_date = '9999-12-31'

- **Check 3.3**: Target count: Exactly 72,000 distinct account_id values

Date Integrity

- **Check 3.4**: No conflicting dates: from_date < to_date for all records
- **Check 3.5**: No gaps in SCD2 history: For each account_id, next record's from_date = previous record's to_date
- **Check 3.6**: opened_date is between corresponding household_registration_date and current_date
- **Check 3.7**: All closed accounts have closed_date > opened_date
- **Check 3.8**: Open accounts have closed_date IS NULL
- **Check 3.9**: Closed accounts have closed_date IS NOT NULL

Referential Integrity

- **Check 3.10**: All advisor_key exist in advisors.advisor_key
- **Check 3.11**: All household_key exist in household.household_key
- **Check 3.12**: All business_line_key exist in business_line.business_line_key
- **Check 3.13**: Referenced advisor must be active during account lifetime
- **Check 3.14**: Referenced household must be active during account lifetime

Business Rules

- **Check 3.15**: Accounts per advisor: min 15, max 400

Table: product

Primary Key & Uniqueness

- **Check 4.1**: No duplicated product_id values
- **Check 4.2**: Target count: Exactly 350 distinct product_id values

Business Rules

- **Check 4.6**: product_name is not null and not empty

Table: advisor_payout_rate

Primary Key & Completeness

- **Check 6.1**: All firm_affiliation_model values from advisors table exist

Table: fact_account_initial_assets

Referential Integrity

- **Check 7.1**: All account_key exist in account.account_key
- **Check 7.2**: Only accounts opened at snapshot_start_date should be included

Business Rules

- **Check 7.3**: account_initial_assets are positive (> 0)
- **Check 7.4**: No account should exceed $20M in initial assets

Table: fact_account_monthly

Date Integrity

- **Check 8.1**: Exactly 12 distinct snapshot_date values (end-of-month dates)
- **Check 8.2**: First snapshot_date = snapshot_start_date, last snapshot_date = current_date

Referential Integrity

- **Check 8.4**: All account_key exist in account.account_key
- **Check 8.5**: All advisor_key exist in advisors.advisor_key
- **Check 8.6**: All household_key exist in household.household_key
- **Check 8.7**: All business_line_key exist in business_line.business_line_key

Account Coverage

- **Check 8.8**: All accounts with opened_date <= snapshot_date and (closed_date > snapshot_date OR closed_date IS NULL) are included
- **Check 8.9**: No accounts with closed_date <= snapshot_date are included

Business Rules

- **Check 8.10**: account_monthly_return is between -12% and +12%
- **Check 8.11**: For snapshot_date = snapshot_start_date, account_monthly_return = 0%
- **Check 8.12**: account_net_flow is no more than 30% of account_assets_previous_month
- **Check 8.13**: account_assets is positive and ≤ $20M
- **Check 8.14**: Asset calculation validation: account_assets = account_assets_previous_month * (1 + account_monthly_return) + account_net_flow
- **Check 8.15**: For first month, account_assets_previous_month matches fact_account_initial_assets.account_initial_assets
- **Check 8.16**: For subsequent months, account_assets_previous_month matches previous month's account_assets

Table: fact_account_product_monthly

Referential Integrity

- **Check 9.1**: All (snapshot_date, account_key) combinations exist in fact_account_monthly
- **Check 9.2**: All product_id exist in product.product_id

Business Rules

- **Check 9.3**: Sum of product_allocation_pct per (snapshot_date, account_key) equals 100%
- **Check 9.4**: Each account has between 2 and 5 products
- **Check 9.5**: product_allocation_pct is between 0 and 100

Table: fact_household_monthly

Referential Integrity

- **Check 10.1**: All (snapshot_date, household_key) combinations derived from fact_account_monthly
- **Check 10.2**: All household_key exist in household.household_key

Business Rules

- **Check 10.3**: household_assets = sum of account_assets for all household accounts
- **Check 10.4**: household_net_flow = sum of account_net_flow for all household accounts
- **Check 10.5**: high_net_worth_flag is TRUE if and only if household_assets >= $1M
- **Check 10.7**: No households with high_net_worth_flag = FALSE have household_assets >= $1M

Table: fact_revenue_monthly

Referential Integrity

- **Check 11.1**: All records match fact_account_monthly 1:1
- **Check 11.2**: All foreign keys match corresponding values in fact_account_monthly

Business Rules

- **Check 11.3**: gross_fee_amount = account_assets * fee_percentage
- **Check 11.4**: advisor_payout_amount = (gross_fee_amount - third_party_fee) * advisor_payout_rate
- **Check 11.5**: net_revenue = gross_fee_amount - third_party_fee - advisor_payout_amount
- **Check 11.6**: All monetary amounts are non-negative
- **Check 11.7**: net_revenue should be positive

Table: fact_customer_feedback

Referential Integrity

- **Check 13.1**: All household_key exist in household.household_key
- **Check 13.2**: All advisor_key exist in advisors.advisor_key
- **Check 13.3**: Referenced household must have household_status = 'Active'
- **Check 13.4**: Referenced advisor must be active during feedback period

Business Rules

- **Check 13.5**: feedback_date is between January 1st of previous year and current_date
- **Check 13.6**: satisfaction_score is between 0 and 100

List of questions

- What was our net inflow of client assets this month, and how does it compare to the same month last year?
- Which top 20 clients had the largest withdrawals in the past 7 days?
- What’s our YTD fee revenue by product line, and which products are growing fastest vs. last quarter?
- How much month-to-date fee revenue have we generated, and how does it compare to the same period last month?
- Which advisors brought in the most net new AUM in the past 30 days?
- How many high-net-worth clients do we have right now, and what’s their total AUM?
- Which product lines contributed the most to net new AUM this quarter, and where are we seeing declines?
- Which client segments have the highest churn rate over the past 6 months?
- Exposure check (policy-light): Which clients have >70% of AUM in a single product or in cash?
- Which client segments have the highest churn rate over the past 6 months?
- Which advisors show warning signals: ≥2 low-satisfaction feedbacks (≤60) in the last 90 days AND net outflows over the last 30 days?

- Which business segment is responsible for most revenue?
- Which segments have the most room to grow?
- From a production or asset level perspective, which segments are the most critical? (if we wanted to grow the business as quickly as possible, which segments we should prioritize and why?) Because if we look in the AFP report, we see that Enterprise are a small % of firms as a number, but they make up a quarter of revenue so if we can have more.

- For this firm or advisor, was was their EOM asset value and affiliation credit? So they see these things together. Broken by business line. How much of that was mutual fund data / cash equivalent?
- What helps an advisor makes more revenue? Is it product? Is it business lines? Types of accounts? Are they IRAs, 401ks?

- What is that we can tell advisors: if you do this, you generate more revenue?