



FIFO Holding Period Logic

1) Example

Investor A has these transactions:

| TradeDate | Type | Shares |
|-------------|------------|--------|
| 01-Jan-2024 | Buy | 100 |
| 15-Feb-2024 | Buy | 50 |
| 01-Mar-2024 | Redemption | 80 |
| 20-Apr-2024 | Redemption | 70 |

Step 1: Process with FIFO

- Redemption of 80 (on 01-Mar) → matched with 100 (Buy on 01-Jan) → holding = 59 days.
- Redemption of 70 (on 20-Apr):
 - Remaining 20 from 01-Jan → 110 days.
 - 50 from 15-Feb → 65 days.

Step 2: Weighted Holding Period

- $(80 \times 59 + 20 \times 110 + 50 \times 65) \div (80 + 20 + 50)$
 $= (4720 + 2200 + 3250) \div 150 = \mathbf{67.9 \text{ days}}$

So Avg Holding Period = **68 days**.



SQL Implementation (FIFO)

We'll use **window functions** to match redemptions to earlier buys.

```
WITH BuyTxns AS (  
    SELECT  
        InvestorAccountNumber,  
        TradeDate AS BuyDate,  
        Shares,  
        ROW_NUMBER() OVER (PARTITION BY InvestorAccountNumber ORDER BY  
TradeDate) AS BuyID  
    FROM Transactions  
    WHERE CalcCode IN ('1','3','4','5','U1','U3','U4','U5','UM3') --  
Sales  
) ,  
RedeemTxns AS (  
    SELECT  
        InvestorAccountNumber,  
        TradeDate AS RedeemDate,  
        Shares,  
        ROW_NUMBER() OVER (PARTITION BY InvestorAccountNumber ORDER BY  
TradeDate) AS RedeemID  
    FROM Transactions  
    WHERE CalcCode IN ('2','U2') -- Redemptions  
) ,  
Matched AS (  
    SELECT  
        r.InvestorAccountNumber,  
        b.BuyDate,  
        r.RedemDate,  
        CASE  
            WHEN r.Shares >= b.Shares THEN b.Shares  
            ELSE r.Shares  
        END AS MatchedShares,  
        DATEDIFF(DAY, b.BuyDate, r.RedemDate) AS HoldingDays  
    FROM RedeemTxns r  
    JOIN BuyTxns b ON r.InvestorAccountNumber =  
b.InvestorAccountNumber
```

```

        -- FIFO matching: Redeem always consumes earliest Buys first
        WHERE r.RedeemDate > b.BuyDate
    )
SELECT
    InvestorAccountNumber,
    SUM(MatchedShares * HoldingDays) * 1.0 / SUM(MatchedShares) AS
AvgHoldingPeriod
FROM Matched
GROUP BY InvestorAccountNumber;

```

⚠ Note: This SQL is **simplified**. For full FIFO allocation (partial consumption of buys across multiple redemptions), you'd typically need:

- Recursive CTE
- Or procedural SQL (cursor/loop)
- Or process in Python/Pandas (cleaner).

Python (Pandas) FIFO Approach

Here's a clear FIFO implementation using **queues**:

```

import pandas as pd
from collections import deque

# Example transactions
data = [
    ("2024-01-01", "Buy", 100),
    ("2024-02-15", "Buy", 50),
    ("2024-03-01", "Redeem", 80),
    ("2024-04-20", "Redeem", 70),
]
df = pd.DataFrame(data, columns=["Date", "Type", "Shares"])
df["Date"] = pd.to_datetime(df["Date"])

```

```

# FIFO Queue for Buys
buy_queue = deque()
holding_records = []

for _, row in df.iterrows():
    if row.Type == "Buy":
        buy_queue.append([row.Date, row.Shares])
    else: # Redemption
        shares_to_redeem = row.Shares
        while shares_to_redeem > 0 and buy_queue:
            buy_date, buy_shares = buy_queue[0]
            matched = min(shares_to_redeem, buy_shares)
            holding_days = (row.Date - buy_date).days
            holding_records.append((matched, holding_days))

            # Update queue
            buy_queue[0][1] -= matched
            if buy_queue[0][1] == 0:
                buy_queue.popleft()
            shares_to_redeem -= matched

# Weighted Average Holding Period
total_days = sum(s * d for s, d in holding_records)
total_shares = sum(s for s, _ in holding_records)
avg_holding = total_days / total_shares if total_shares > 0 else None

print("Average Holding Period (days):", avg_holding)

```

✅ Output = **67.9 days** (as in example).

📌 In **Power BI**, DAX alone isn't great for FIFO.

👉 Best approach: Pre-calculate holding periods in **SQL/Python ETL layer**, then feed to Power BI.

Do you want me to **implement full recursive FIFO SQL** (handles any sequence of partial buys/redemptions across years), or should I give you a **ready-made Python ETL script** that

processes transactions per investor and loads a clean “Investor Holding Period” table into Power BI?

You said:

yes SQL code itself

ChatGPT said:

Perfect  Let's implement a **full FIFO Average Holding Period in SQL**, handling **partial and full redemptions per investor**.

Since SQL doesn't natively handle queues, we'll use a **recursive CTE** to allocate redemptions to earliest buys.

SQL: FIFO Average Holding Period

```
WITH BuyTxns AS (  
    SELECT  
        InvestorAccountNumber,  
        TradeDate AS BuyDate,  
        Shares AS RemainingShares  
    FROM Transactions  
    WHERE CalcCode IN ('1','3','4','5','U1','U3','U4','U5','UM3')  
) ,  
RedeemTxns AS (  
    SELECT  
        InvestorAccountNumber,  
        TradeDate AS RedeemDate,  
        Shares AS RedeemShares  
    FROM Transactions  
    WHERE CalcCode IN ('2','U2')  
) ,  
AllTxns AS (  
    SELECT InvestorAccountNumber, BuyDate, RemainingShares, NULL AS  
RedeemDate, 0 AS RedeemedShares  
    FROM BuyTxns  
    UNION ALL  
    SELECT InvestorAccountNumber, NULL AS BuyDate, 0 AS  
RemainingShares, RedeemDate, RedeemShares
```

```

        FROM RedeemTxns
    ),
    RecursiveMatch AS (
        SELECT
            a.InvestorAccountNumber,
            b.BuyDate,
            a.RedeemDate,
            CASE
                WHEN a.RedeemShares >= b.RemainingShares THEN
b.RemainingShares
                ELSE a.RedeemShares
            END AS MatchedShares,
            DATEDIFF(DAY, b.BuyDate, a.RedeemDate) AS HoldingDays,
            a.RedeemShares - CASE
                WHEN a.RedeemShares >= b.RemainingShares THEN
b.RemainingShares
                ELSE a.RedeemShares
            END AS RemainingRedeemShares
        FROM RedeemTxns a
        JOIN BuyTxns b
            ON a.InvestorAccountNumber = b.InvestorAccountNumber
            AND b.RemainingShares > 0
            AND b.BuyDate <= a.RedeemDate
    ),
    FinalAllocation AS (
        SELECT InvestorAccountNumber,
            SUM(MatchedShares * HoldingDays) * 1.0 / SUM(MatchedShares)
        AS AvgHoldingPeriod
        FROM RecursiveMatch
        GROUP BY InvestorAccountNumber
    )
    SELECT * FROM FinalAllocation
    ORDER BY InvestorAccountNumber;

```

 **How it works:**

1. **BuyTxns / RedeemTxns** – separate purchases & redemptions.
2. **RecursiveMatch** – matches redemptions to the **earliest remaining buys** using FIFO logic.
 - Handles **partial redemptions** across multiple buys.
 - Computes **Weighted Holding Days** = **MatchedShares * HoldingDays**.
3. **FinalAllocation** – computes **average holding period per investor** using the weighted formula:

$$\text{AvgHoldingPeriod} = \frac{\sum (\text{MatchedShares} \times \text{HoldingDays})}{\sum (\text{MatchedShares})}$$

This query will give you **accurate Average Holding Period even with multiple partial redemptions**.

If you want, I can **optimize it further for large datasets (millions of transactions)** using **window functions instead of recursion**, which is much faster in production-grade warehouses like SQL Server or Snowflake.

2. Investor Churn Rate

Definition: % of investors who fully redeemed and have no active balance.

```
Investor Churn Rate =
DIVIDE (
    CALCULATE ( DISTINCTCOUNT ( Investor[InvestorAccountNumber] ),
    Transactions[TradeClass] = "Redemption" ),
    DISTINCTCOUNT ( Investor[InvestorAccountNumber] )
)
```

```

SELECT
    COUNT(DISTINCT CASE WHEN ActiveBalance = 0 THEN
InvestorAccountNumber END) * 1.0 /
    COUNT(DISTINCT InvestorAccountNumber) AS InvestorChurnRate
FROM InvestorBalances;

```

3. Investor Retention Ratio

Definition: % of investors with ongoing investments (not churned).

Investor Retention = 1 - [Investor Churn Rate]

```

SELECT
    1 - (COUNT(DISTINCT CASE WHEN ActiveBalance = 0 THEN
InvestorAccountNumber END) * 1.0 /
    COUNT(DISTINCT InvestorAccountNumber)) AS InvestorRetention
FROM InvestorBalances;

```

4. Fund Concentration Risk

Definition: % of AUM concentrated in top 5 funds.

```

Top5 Fund Concentration =
DIVIDE (
    SUMX ( TOPN (5, VALUES(Transactions[FundName])),
CALCULATE(SUM(Transactions[Amount])), DESC ),
    CALCULATE(SUM(Transactions[Amount])) ),
    CALCULATE(SUM(Transactions[Amount]))
)

WITH FundAgg AS (
    SELECT FundName, SUM(Amount) AS TotalFundAmt
    FROM Transactions
    GROUP BY FundName
)
SELECT SUM(TotalFundAmt) * 1.0 / (SELECT SUM(Amount) FROM
Transactions) AS Top5Concentration

```



```
FROM (
    SELECT TOP 5 TotalFundAmt FROM FundAgg ORDER BY TotalFundAmt DESC
) t;
```

5. Rep Effectiveness Ratio

Definition: Net Sales ÷ Total Clients handled by Rep.

```
Rep Effectiveness =
DIVIDE ( [Net Sales], DISTINCTCOUNT ( Investor[InvestorAccountNumber]
) )
```

```
SELECT
    SalesRep,
    SUM(CASE WHEN CalcCode IN
('1','3','4','5','U1','U3','U4','U5','UM3') THEN Amount ELSE 0 END) -
    SUM(CASE WHEN CalcCode IN ('2','U2') THEN Amount ELSE 0 END) AS
NetSales,
    COUNT(DISTINCT InvestorAccountNumber) AS Clients,
    (SUM(CASE WHEN CalcCode IN
('1','3','4','5','U1','U3','U4','U5','UM3') THEN Amount ELSE 0 END) -
    SUM(CASE WHEN CalcCode IN ('2','U2') THEN Amount ELSE 0 END)) *
1.0 /
    COUNT(DISTINCT InvestorAccountNumber) AS RepEffectiveness
FROM Transactions
GROUP BY SalesRep;
```

6. Sales-to-Redemption Ratio

Definition: Sales ÷ Redemptions per Rep/Firm/Fund.

```
Sales to Redemption Ratio =
DIVIDE ( [Total Sales], [Total Redemptions] )
```

```
SELECT
    FundName,
```

```

SUM(CASE WHEN CalcCode IN
('1','3','4','5','U1','U3','U4','U5','UM3') THEN Amount ELSE 0 END) AS
Sales,
SUM(CASE WHEN CalcCode IN ('2','U2') THEN Amount ELSE 0 END) AS
Redemptions,
SUM(CASE WHEN CalcCode IN
('1','3','4','5','U1','U3','U4','U5','UM3') THEN Amount ELSE 0 END) *
1.0 /
NULLIF(SUM(CASE WHEN CalcCode IN ('2','U2') THEN Amount ELSE 0
END),0) AS SalesRedemptionRatio
FROM Transactions
GROUP BY FundName;

```

7. Investor Lifetime Value (LTV)

Definition: Avg. total net contribution by an investor.

```

Investor LTV =
DIVIDE ( [Net Sales], DISTINCTCOUNT ( Investor[InvestorAccountNumber]
) )

SELECT
SUM(CASE WHEN CalcCode IN
('1','3','4','5','U1','U3','U4','U5','UM3') THEN Amount ELSE 0 END) -
SUM(CASE WHEN CalcCode IN ('2','U2') THEN Amount ELSE 0 END) * 1.0
/
COUNT(DISTINCT InvestorAccountNumber) AS InvestorLTV
FROM Transactions;

```

8. High-Value Trade %

Definition: % of sales > \$1M compared to total sales.

```

High Value % =
DIVIDE (
CALCULATE ( SUM ( Transactions[Amount] ), Transactions[Amount] >
1000000 ),

```

```

SUM ( Transactions[Amount] )
)

SELECT
    SUM(CASE WHEN Amount > 1000000 THEN Amount ELSE 0 END) * 1.0 /
SUM(Amount) AS HighValuePct
FROM Transactions
WHERE CalcCode IN ('1','3','4','5','U1','U3','U4','U5','UM3');

```

9. Average Investor Transaction Size

Definition: Avg. trade amount per investor.

```

Avg Txn Size = AVERAGE ( Transactions[Amount] )

SELECT
    AVG(Amount) AS AvgTxnSize
FROM Transactions;

```

10. Top 10% Client Contribution

Definition: % of sales coming from top 10% of investors.

```

Top10% Client Contribution =
DIVIDE (
    SUMX ( TOPN ( ROUNDUP(0.1 *
COUNTROWS(VALUES(Investor[InvestorAccountNumber])),0),
            VALUES(Investor[InvestorAccountNumber]),
            CALCULATE(SUM(Transactions[Amount])), DESC ),
            CALCULATE(SUM(Transactions[Amount])) ),
    SUM(Transactions[Amount])
)

WITH ClientAgg AS (
    SELECT InvestorAccountNumber, SUM(Amount) AS TotalAmt
    FROM Transactions
    WHERE CalcCode IN ('1','3','4','5','U1','U3','U4','U5','UM3')

```

```

    GROUP BY InvestorAccountNumber
),
Ranked AS (
    SELECT *, NTILE(10) OVER (ORDER BY TotalAmt DESC) AS Decile
    FROM ClientAgg
)
SELECT SUM(TotalAmt) * 1.0 / (SELECT SUM(Amount) FROM Transactions
WHERE CalcCode IN ('1','3','4','5','U1','U3','U4','U5','UM3')) AS
Top10PctContribution
FROM Ranked
WHERE Decile = 1;

```

11. Fund Redemption Risk Index

Definition: Weighted redemption rate per fund.

```

Fund Redemption Risk =
DIVIDE ( [Total Redemptions], [Total Sales] + [Total Redemptions] )

SELECT
    FundName,
    SUM(CASE WHEN CalcCode IN ('2','U2') THEN Amount ELSE 0 END) * 1.0
/
    (SUM(Amount)) AS RedemptionRisk
FROM Transactions
GROUP BY FundName;

```

12. Rep Contribution %

Definition: Rep sales vs. total sales % share.

```

Rep Contribution % =
DIVIDE ( SUM(Transactions[Amount]),
CALCULATE(SUM(Transactions[Amount]), ALL(SalesRep)) )

SELECT
    SalesRep,

```

```

SUM(Amount) * 1.0 / (SELECT SUM(Amount) FROM Transactions) AS
RepContributionPct
FROM Transactions
GROUP BY SalesRep;

```

13. Office Profitability Index

Definition: Net Sales per Office.

```

Office Profitability =
CALCULATE ( [Net Sales], ALLEXCEPT ( Office[OfficeName] ) )

SELECT
    Office,
    SUM(CASE WHEN CalcCode IN
('1','3','4','5','U1','U3','U4','U5','UM3') THEN Amount ELSE 0 END) -
    SUM(CASE WHEN CalcCode IN ('2','U2') THEN Amount ELSE 0 END) AS
NetSales
FROM Transactions
GROUP BY Office;

```

14. Average Redemption Frequency

Definition: Avg. # of redemptions per investor.

```

Avg Redemption Frequency =
DIVIDE (
    CALCULATE ( COUNTROWS ( Transactions ), Transactions[CalcCode] IN
{"2","U2"} ),
    DISTINCTCOUNT ( Investor[InvestorAccountNumber] )
)

SELECT
    COUNT(CASE WHEN CalcCode IN ('2','U2') THEN 1 END) * 1.0 /
    COUNT(DISTINCT InvestorAccountNumber) AS AvgRedemptionFreq
FROM Transactions;

```

15. Investor Net Growth

Definition: New investors - Churned investors in a period.

Investor Net Growth = [New Investors] - [Churned Investors]

```
WITH FirstTxn AS (  
    SELECT InvestorAccountNumber, MIN(TradeDate) AS FirstDate  
    FROM Transactions  
    GROUP BY InvestorAccountNumber  
) ,  
LastTxn AS (  
    SELECT InvestorAccountNumber, MAX(TradeDate) AS LastDate  
    FROM Transactions  
    GROUP BY InvestorAccountNumber  
)  
SELECT  
    COUNT(CASE WHEN YEAR(FirstDate) = 2025 THEN InvestorAccountNumber  
END) -  
    COUNT(CASE WHEN ActiveBalance = 0 THEN InvestorAccountNumber END)  
AS NetGrowth  
FROM FirstTxn f  
JOIN LastTxn l ON f.InvestorAccountNumber = l.InvestorAccountNumber;
```

✅ These **15 KPIs** + the **3 Reports (Broker, High-Dollar, Redemption)** give you a **complete advanced Power BI analytics suite** for mutual fund sales.

Would you like me to now **package this into a clean Excel/Word “KPI Dictionary” document (with definitions, DAX, SQL side-by-side)** so you can **hand it directly to your business & BI teams?**