

## Laneway Transactions

The SQL view `silver.laneway_transactions_rmit` was created to extract and structure the necessary information required for engagement measurement and visualisations. It consolidates transaction records, applies data cleaning and classification rules, and introduces derived fields (such as `member_key` and `transaction_type`) to support accurate reporting.

### Member Information

Member key was formulated to be a unique identifier as the `member_number` was not unique across different brands.

Column Name	Description	Source
<code>member_number</code>	The member's identification number.	production.bronze.laneway_monthly_transactions. <code>member_number</code>
<code>member_key</code>	A derived key uniquely identifying the member across different account types (Childcare or Guild).	Concatenation of a prefix (“CH” or “GU”) and the <code>member_number</code> from the <code>childcare_accounts</code> or <code>guild_accounts</code> tables.

### Transaction Information

Column Name	Description	Source
<code>unique_id</code>	A unique identifier for the transaction record	production.bronze.laneway_monthly_transactions. <code>unique_id</code>
<code>account_id</code>	The unique identifier for the account associated with the transaction.	production.bronze.laneway_monthly_transactions. <code>account_key</code>
<code>effective_date</code>	The date the transaction became effective.	production.bronze.laneway_monthly_transactions. <code>effective_date</code> Converted from a timestamp/string to a DATE format from the bronze table.
<code>description</code>	A combined description of the transaction.	Concatenation of <code>transaction_type_group</code> and <code>transaction_description</code> from the production.bronze.laneway_monthly_transactions
<code>debit_amount</code>	The amount debited from the account for this transaction	production.bronze.laneway_monthly_transactions
<code>credit_amount</code>	The amount credited to the account for this transaction.	production.bronze.laneway_monthly_transactions

The *transaction\_type* field is a derived column designed to simplify the classification of transactions for reporting and KPI measurement. Specifically, it supports the calculation of the Engagement KPI, which segments members into *Committed*, *Comfortable*, or *Disengaged* groups.

Classification logic:

1. SuperSuper – Transactions are classified as *SuperSuper* if the *transaction\_type\_group* matches this string.
2. Employer – Transactions are classified as *Employer* if they relate to account code P03 and contain the word “employer”.
3. Member – Transactions are classified as *Member* if they meet a specific combination of *transaction\_type\_group*, *account\_code*, and *transaction\_type\_code* values.
4. Rollover – Transactions are classified as *Rollover* if they are an external transfer, contain the phrase “rollover in” in the description, and have a *debit\_amount* of 0.

Column Name	Description	Source
<b>transaction_type</b>	A unique identifier for the transaction record	production.bronze.laneway_monthly_transactions

## SQL Query:

```

CREATE VIEW silver.laneway_transactions_rmit (
    unique_id,
    account_id,
    member_number,
    member_key,
    effective_date,
    description,
    debit_amount,
    credit_amount,
    transaction_type
)
WITH SCHEMA COMPENSATION
AS (
SELECT
    laneway.unique_id,
    laneway.account_id,
    laneway.member_number,
    ga.member_key,
    laneway.effective_date,
    laneway.description,
    laneway.debit_amount,
    laneway.credit_amount,
    laneway.transaction_type
FROM
    (SELECT
        unique_id unique_id,
        TRIM(member_number) member_number,
        TRIM(account_key) account_id,
        TO_DATE(effective_date) effective_date,
        CONCAT(transaction_type_group, ' ', transaction_description)
description,

```

```

debit_amount,
credit_amount,
(CASE
    WHEN LOWER(transaction_type_group) LIKE 'supersuper' THEN
"SuperSuper"
    WHEN TRIM(account_code) IN ('P03') AND
LOWER(transaction_type_group) LIKE '%employer%' THEN "Employer"
    WHEN LOWER(transaction_type_group) LIKE 'member' AND
TRIM(account_code) IN ('P03', 'P04', 'P05') AND TRIM(transaction_type_code)
IN ('01', '08', '9V', '9W', '8H', '8K') THEN "Member"
    WHEN LOWER(transaction_type_group) LIKE 'external%transfer%'
AND LOWER(transaction_description) LIKE '%rollover%in%' AND debit_amount =
0.0 THEN "Rollover"
    ELSE NULL
END) transaction_type
FROM production.bronze.laneway_monthly_transactions
WHERE (LOWER(transaction_type_group) LIKE 'supersuper' OR
(LOWER(transaction_type_group) LIKE 'member') OR
(LOWER(transaction_type_group) LIKE '%employer%') OR
(LOWER(transaction_type_group) LIKE 'external%transfer%' AND
LOWER(transaction_description) LIKE '%rollover%in%'))
    AND YEAR(effective_date) >= 2024
) laneway

LEFT JOIN
(SELECT DISTINCT
CONCAT("CH", member_number) member_key,
TRIM(member_number) member_number,
TRIM(account_number) account_id
FROM production.silver.childcare_accounts

UNION

SELECT DISTINCT
CONCAT("GU", member_number) member_key,
TRIM(member_number) member_number,
TRIM(account_number) account_id
FROM production.silver.guild_accounts
) ga
ON (ga.member_number = laneway.member_number AND ga.account_id =
laneway.account_id)
);

```

## Member Data

The SQL view `silver.memberdata` was created to consolidate member information across multiple brands, Childcare, Future Super, and Guild into a unified structure.

It selects key demographic and membership attributes such as gender, date of birth, join and exit dates, and location information. Derived columns, including `member_key` and standardized gender values, ensure consistency across brands.

### Member Information

Column Name	Description	Source
<code>member_number</code>	The original unique identifier for the member within their source system.	Directly from the respective member tables ( <code>childcare_members</code> , <code>future_members</code> ).
<code>member_key</code>	Globally Unique Key for the member across all brands.	Concatenation of a brand prefix ('CH', 'FS' or 'GU') and the <code>member_number</code> .

Column Name	Description	Source
<code>sex</code>	The standardized gender identity of the member.	Categorized as ' <b>Female</b> ' (from 'female', 'woman'), ' <b>Male</b> ' (from 'male', 'man'), or ' <b>Other</b> ' (for all else, including unmapped or NULL values) from the brand contact's table.
<code>birth_date</code>	The member's date of birth.	From the contacts tables: <code>guild_contacts</code> , <code>childcare_contacts</code> and <code>future_contacts</code>
<code>join_date</code>	The date the member joined the fund.	From the contacts tables: <code>guild_members</code> , <code>childcare_members</code> and <code>future_members</code>
<code>exit_date</code>	The date the member exited, withdrew, or their membership was updated to inactive.	From the contacts tables: <code>guild_members</code> , <code>childcare_members</code> and <code>future_members</code>

### Location

Column Name	Description	Source
<code>Postal Code</code>	The postal code of the member's residential address.	Directly from the respective contact table.
<code>State</code>	The state or region of the member's residential address.	Directly from the respective contact table.

<b>Country</b>	The country of the member's residential address.	Directly from the respective contact table.
<b>Brand</b>	The originating product brand for this member record.	Directly from the respective contact table.

### SQL Query:

```

CREATE VIEW silver.memberdata (
    member_number,
    member_key,
    sex,
    birth_date,
    join_date,
    exit_date,
    PostalCode,
    State,
    Country,
    Brand
)
WITH SCHEMA COMPENSATION
AS (
    SELECT DISTINCT
        member_number,
        member_key,
        (CASE
            WHEN LOWER(TRIM(sex)) IN ('female','woman') THEN 'Female'
            WHEN LOWER(TRIM(sex)) IN ('male','man') THEN 'Male'
            ELSE "Other"
        END) sex,
        birth_date,
        join_date,
        exit_date,
        PostalCode,
        State,
        Country,
        Brand
    FROM (
        SELECT
            CONCAT('CH', member_number) member_key,
            member_number,
            sex,
            production.silver.standardize_date(birth_date) AS
            birth_date,
            production.silver.standardize_date(join_date) AS join_date,
            production.silver.standardize_date(exit_date) AS exit_date,
            PostalCode,
            State,

```

```

        Country,
        "Childcare" AS Brand
FROM (
    SELECT
        id,
        sex,
        dateOfBirth AS birth_date,
        PostalCode,
        State,
        Country,
        TFN Supplied AS Tfnflag
    FROM production.silver.childcare_contacts
) cc
JOIN (
    SELECT
        TRIM(Member Number) AS member_number,
        Latest_Login_Type,
        Latest_Interaction_Date,
        Latest_Interaction_Types,
        memberType,
        Join Date AS join_date,
        Exit Date AS exit_date,
        Latest Employer Contribution Date AS
latest_employer_contri_date,
        Latest Voluntary Contribution Date AS
latest_voluntary_contri_date,
        Most Recent Contribution Date AS recent_contri_date,
        3rd Most Recent Contribution Type AS recent_contri_type,
        Total Contributions for Month AS total_contributions,
        contact_id
    FROM production.silver.childcare_members
) cm ON cc.id = cm.contact_id

UNION

SELECT
    (CASE WHEN SUBSTR(member_number, 1, 2) = 'FS' THEN
member_number ELSE CONCAT('FS', member_number) END) member_key,
    member_number,
    sex,
    production.silver.standardize_date(birth_date) AS
birth_date,
    production.silver.standardize_date(join_date) AS join_date,
    (CASE WHEN memberType = 'withdrawn' THEN
production.silver.standardize_date(updated_at) ELSE NULL END)
exit_date,
    PostalCode,
    State,
    Country,
    "Future Super" AS Brand
FROM (

```

```

SELECT
    id,
    gender AS sex,
    birth_date,
    residential_address_postcode AS PostalCode,
    residential_address_state AS State,
    residential_address_country AS Country,
    tax_file_number_valid AS Tfnflag
    FROM production.silver.future_contacts
) fc
JOIN (
    SELECT
        TRIM(member_number) member_number,
        contact_id,
        status AS memberType,
        join_date,
        updated_at
    FROM production.silver.future_members
) fm ON fc.id = fm.contact_id

UNION

SELECT
    CONCAT('GU', member_number) member_key,
    member_number,
    sex,
    production.silver.standardize_date(birth_date) AS
birth_date,
    production.silver.standardize_date(join_date) AS join_date,
    production.silver.standardize_date(exit_date) AS exit_date,
    PostalCode,
    State,
    Country,
    "Guild" AS Brand
FROM (
    SELECT
        id,
        sex,
        dateOfBirth AS birth_date,
        PostalCode,
        State,
        Country,
        TFN Supplied AS Tfnflag
        FROM production.silver.guild_contacts
) gc
JOIN (
    SELECT
        TRIM(Member Number) AS member_number,
        Latest_Login_Type,
        Latest_Interaction_Date,
        Latest_Interaction_Types,

```

```
memberType,  
Join Date AS join_date,  
Exit Date AS exit_date,  
Latest Employer Contribution Date AS  
latest_employer_contri_date,  
Latest Voluntary Contribution Date AS  
latest_voluntary_contri_date,  
Most Recent Contribution Date AS recent_contri_date,  
3rd Most Recent Contribution Type AS recent_contri_type,  
Total Contributions for Month AS total_contributions,  
contact_id  
FROM production.silver.guild_members  
) gm ON gc.id = gm.contact_id  
) AS members  

```

## Future Transactions

The silver.future\_transactions\_rmit view was developed to extract and categorize transactional data related to Future Super members. Its primary purpose is to capture basic transaction information that could be utilised in Power BI.

This view consolidates information from future\_registry\_transactions and future\_member\_accounts, ensuring a standardized structure for reporting. All transaction dates are harmonized and filtered to include only activity from 2024 onwards, with known data anomalies between May 20, 2025, and June 3, 2025, excluded to maintain data accuracy.

## Member Information

Member key was formulated to be a unique identifier as the *member\_number* was not unique across different brands.

Column Name	Description	Source
<b>member_number</b>	The member's identification number within the Future Super system	production.silver.future_registry_transactions
<b>member_key</b>	A derived global key uniquely identifying the member.	Generated by prefixing 'FS' to the member_number if not already present. Joined from future_member_accounts

## Transactions Information:

Column Name	Description	Source
<b>unique_id</b>	A unique identifier for the transaction record.	production.silver.future_registry_transactions
<b>account_id</b>	The unique identifier for the account associated with the transaction.	production.silver.future_registry_transactions
<b>effective_date</b>	The date the transaction became effective.	Converted to a date using to_date(effective_date) and filtered to include only transactions from 2024 onward, excluding the period 2025-05-20 to 2025-06-03. From production.silver.future_registry_transactions
<b>description</b>	A combined description of the transaction.	production.silver.future_registry_transactions
<b>debit_amount</b>	The amount debited from the account for this transaction.	Converted from cents to dollars using (credit_amount_cents / 100). From production.silver.future_registry_transactions.
<b>credit_amount</b>	The amount credited to the account for this transaction.	Converted from cents to dollars using (credit_amount_cents / 100). From production.silver.future_registry_transactions

Classification logic:

1. Rollover: account\_code in ('P06', 'P07'), transaction\_type\_code = '79', and account\_sub\_account does not contain 'inttfr'.
2. Employer: account\_code = 'P03' and description contains 'employer'.
3. Member: account\_code in ('P03', 'P04', 'P05'), transaction\_type\_code in ('01', '08', '9V', '9W', '8H'), and description includes 'member concessional contribution' or 'member non-concessional contribution'.
4. NULL: If none of the above conditions are met.

Column Name	Description	Source
<b>transaction_type</b>	A unique identifier for the transaction record	production.bronze.laneway_monthly_transactions

**SQL Query:**

```

CREATE VIEW silver.future_transactions_rmit (
    unique_id,
    account_id,
    member_number,
    member_key,
    effective_date,
    description,
    debit_amount,
    credit_amount,
    transaction_type
)
WITH SCHEMA COMPENSATION AS (
    SELECT
        fr.unique_id,
        fr.account_id,
        fr.member_number,
        fa.member_key,
        fr.effective_date,
        fr.description,
        fr.debit_amount,
        fr.credit_amount,
        fr.transaction_type
    FROM (
        SELECT
            unique_identity AS unique_id,
            TRIM(member_number) AS member_number,
            TRIM(account_id) AS account_id,
            TO_DATE(effective_date) AS effective_date,
            description,
            (debit_amount_cents / 100) AS debit_amount,
            (credit_amount_cents / 100) AS credit_amount,
            CASE
                WHEN account_code IN ('P06', 'P07')
                    AND TRIM(transaction_type_code) = '79'

```

```

        AND LOWER(account_sub_account) NOT LIKE
'%inttfr%'
        THEN 'Rollover'
WHEN TRIM(account_code) IN ('P03')
        AND LOWER(description) LIKE '%employer%'
        THEN 'Employer'
WHEN TRIM(account_code) IN ('P03', 'P04', 'P05')
        AND TRIM(transaction_type_code) IN ('01', '08',
'9V', '9W', '8H')
        AND (
            LOWER(description) LIKE '%member
concessional contribution%'
            OR LOWER(description) LIKE '%member non-
concessional contribution%'
        )
        THEN 'Member'
        ELSE NULL
    END AS transaction_type
FROM production.silver.future_registry_transactions
WHERE TRIM(account_code) IN ('P06', 'P07', 'P03', 'P04',
'P05')
        AND transaction_type_code IN ('01', '08', '9V', '9W',
'8H', '79')
        AND effective_date NOT BETWEEN TO_DATE('2025-05-20',
'yyyy-MM-dd')
        AND TO_DATE('2025-06-03', 'yyyy-MM-dd')
        AND YEAR(effective_date) >= 2024
) fr
LEFT JOIN (
    SELECT DISTINCT
        CASE
            WHEN SUBSTR(member_number, 1, 2) = 'FS'
                THEN member_number
            ELSE CONCAT('FS', member_number)
        END AS member_key,
        member_number,
        account_id
    FROM production.silver.future_member_accounts
) fa
ON (fa.member_number = fr.member_number AND fa.account_id =
fr.account_id)
);

```

## Power BI Data Dictionary Table: Member\_Last\_Contributions

This calculated table aggregates transaction-level data from transactiondata, which is itself an appended dataset combining laneway\_transactions and future\_registry\_transactions, into a member-level summary.

The table captures key contribution metrics including the most recent transaction dates and total transaction amounts, segmented by transaction type — *Employer, Member, and Rollover*.

Column Name	Data Type	Description	Calculation Logic
<b>member_key</b>	Text	Unique identifier for each member.	Retrieved directly from 'MemberData'[member_key].
<b>member_number</b>	Text	Member's account number.	Retrieved directly from transactiondata[member_number].
<b>Quarter_End</b>	Date	The last date of the current quarter.	Calculated as MAX('DateTable'[Date]) within each quarter context.
<b>Last Transaction Date</b>	Date	The most recent transaction date for the member.	Uses CALCULATE(MAX('TransactionData'[effective_date]), TREATAS(...), DATESINPERIOD(..., -1, QUARTER))
<b>Total Transactions</b>	Whole Number	Total number of transactions in the last quarter.	Counts 'TransactionData'[unique_id] over the last quarter.
<b>Total T Amount</b>	Decimal	Net total transaction amount (credits minus debits).	Uses SUMX ('TransactionData', COALESCE(credit_amount, 0) - COALESCE(ABS(debit_amount), 0)) over the last quarter.
<b>Last Rollover Date</b>	Date	The most recent rollover transaction date.	Filters where 'TransactionData'[transaction_type] = "Rollover" and takes the max effective date
<b>Last 2 Quarter Rollover</b>	Whole Number	Number of rollover transactions in the last two quarters.	Counts 'TransactionData'[unique_id] for "Rollover" type within DATESINPERIOD(..., -2, QUARTER).
<b>Total Rollovers</b>	Whole Number	Number of rollover transactions in the last quarter.	Counts 'TransactionData'[unique_id] for "Rollover" type within DATESINPERIOD(..., -1, QUARTER).
<b>Total Rollover T Amount</b>	Decimal	Total amount from rollover transactions in the last quarter.	Sum of (credit_amount - debit_amount) for "Rollover" transactions.
<b>Last Employer Contribution Date</b>	Date	Most recent employer contribution date.	Filters where 'TransactionData'[transaction_type] = "Employer" and takes the max effective date
<b>Last 2 Quarter Employer Contributions</b>	Whole Number	Most recent employer contribution date.	Filters where 'TransactionData'[transaction_type] = "Employer" and takes the max effective date.
<b>Last 2 Quarter Employer Contributions</b>	Whole Number	Number of employer contributions in the last two quarters.	Counts 'TransactionData'[unique_id] where transaction_type = "Employer" within DATESINPERIOD(..., -2, QUARTER).
<b>Total Employer Contributions</b>	Whole Number	Number of employer contributions in the last quarter.	Counts "Employer" transactions within DATESINPERIOD(..., -1, QUARTER).

<b>Total Employer T Amount</b>	Decimal	Total amount contributed by employers in the last quarter.	Sum of (credit_amount - debit_amount) where transaction_type = "Employer".
<b>Last Member Contribution Date</b>	Date	The most recent member contribution date.	Filters where 'TransactionData'[transaction_type] = "Member" and takes the max effective date.
<b>Last Member Contribution Date</b>	Date	Most recent member contribution date.	Filters where 'TransactionData'[transaction_type] = "Member" and takes the max effective date.
<b>Last 2 Quarter Member Contributions</b>	Whole Number	Number of member contributions in the last two quarters.	Counts "Member" transactions within DATESINPERIOD(..., -2, QUARTER).
<b>Total Member Contributions</b>	Whole Number	Number of member contributions in the last quarter.	Counts "Member" transactions within DATESINPERIOD(..., -1, QUARTER)
<b>Total Member T Amount</b>	Decimal	Total amount contributed by members in the last quarter.	Sum of (credit_amount - debit_amount) where transaction_type = "Member".

```

Member_Last_Contributions_ByQuarter =
VAR QuarterTable =
    ADDCOLUMNS (
        SUMMARIZE ( 'DateTable', 'DateTable'[Year],
        'DateTable'[Quarter], 'DateTable'[YearQuarter],
        DateTable[YearQuarterKey] ),
        "QuarterEnd", CALCULATE ( MAX ( 'DateTable'[Date] ) )
    )
RETURN
GENERATE (
    QuarterTable,
    SUMMARIZECOLUMNS (
        'MemberData'[member_key],
        'MemberData'[member_number],
        "Quarter_End", [QuarterEnd],
        // ----- All transactions -----
        "Last Transaction Date",
        CALCULATE (
            MAX ( 'TransactionData'[effective_date] ),
            TREATAS ( VALUES ( 'MemberData'[member_key] ),
            'TransactionData'[member_key] ),
            TREATAS (
                DATESINPERIOD ( 'DateTable'[Date], [QuarterEnd],
                -1, QUARTER ),
                'TransactionData'[effective_date]
            )
        ),

```

```

    "Total Transactions",
    CALCULATE (
        COUNT ( 'TransactionData'[unique_id] ),
        TREATAS ( VALUES ( 'MemberData'[member_key] ),
        'TransactionData'[member_key] ),
        TREATAS (
            DATESINPERIOD ( 'DateTable'[Date], [QuarterEnd],
-1, QUARTER ),
            'TransactionData'[effective_date]
        )
    ),
    "Total T Amount",
    CALCULATE (
        SUMX (
            'TransactionData',
            COALESCE ( 'TransactionData'[credit_amount], 0 )
            - COALESCE (
                abs ('TransactionData'[debit_amount]), 0 )
        ),
        TREATAS ( VALUES ( 'MemberData'[member_key] ),
        'TransactionData'[member_key] ),
        TREATAS (
            DATESINPERIOD ( 'DateTable'[Date], [QuarterEnd],
-1, QUARTER ),
            'TransactionData'[effective_date]
        )
    ),
    // ----- Rollover -----
    "Last Rollover Date",
    CALCULATE (
        MAX ( 'TransactionData'[effective_date] ),
        'TransactionData'[transaction_type] = "Rollover",
        TREATAS ( VALUES ( 'MemberData'[member_key] ),
        'TransactionData'[member_key] ),
        TREATAS (
            DATESINPERIOD ( 'DateTable'[Date], [QuarterEnd],
-1, QUARTER ),
            'TransactionData'[effective_date]
        )
    ),
    "Last 2 Quarter Rollovers",
    CALCULATE (
        COUNT ( 'TransactionData'[unique_id] ),
        KEEPFILTERS ( 'TransactionData'[transaction_type] =
"Rollover" ),
        TREATAS ( VALUES ( 'MemberData'[member_key] ),
```

```

'TransactionData'[member_key] ),
    TREATAS (
        DATESINPERIOD ( 'DateTable'[Date], [QuarterEnd],
-2, QUARTER ),
            'TransactionData'[effective_date]
        )
),
),

"Total Rollovers",
CALCULATE (
    COUNT ( 'TransactionData'[unique_id] ),
    'TransactionData'[transaction_type] = "Rollover",
    TREATAS ( VALUES ( 'MemberData'[member_key] ),
'TransactionData'[member_key] ),
    TREATAS (
        DATESINPERIOD ( 'DateTable'[Date], [QuarterEnd],
-1, QUARTER ),
            'TransactionData'[effective_date]
        )
),
),

"Total Rollover T Amount",
CALCULATE (
    SUMX (
        'TransactionData',
        COALESCE ( 'TransactionData'[credit_amount], 0 )
        - COALESCE (
abs('TransactionData'[debit_amount]), 0 )
        ),
        'TransactionData'[transaction_type] = "Rollover",
        TREATAS ( VALUES ( 'MemberData'[member_key] ),
'TransactionData'[member_key] ),
    TREATAS (
        DATESINPERIOD ( 'DateTable'[Date], [QuarterEnd],
-1, QUARTER ),
            'TransactionData'[effective_date]
        )
),
),

// ----- Employer -----
"Last Employer Contribution Date",
CALCULATE (
    MAX ( 'TransactionData'[effective_date] ),
    'TransactionData'[transaction_type] = "Employer",
    TREATAS ( VALUES ( 'MemberData'[member_key] ),
'TransactionData'[member_key] )
),

"Last 2 Quarter Employer Contributions",
CALCULATE (
    COUNT ( 'TransactionData'[unique_id] ),

```

```

        KEEPFILTERS ( 'TransactionData'[transaction_type] =
"Employer" ),
            TREATAS ( VALUES ( 'MemberData'[member_key] ),
'[member_key] ),
            TREATAS (
                DATESINPERIOD ( 'DateTable'[Date], [QuarterEnd],
-2, QUARTER ),
                    'TransactionData'[effective_date]
                )
            ),
        )

    "Total Employer Contributions",
    CALCULATE (
        COUNT ( 'TransactionData'[unique_id] ),
        'TransactionData'[transaction_type] = "Employer",
        TREATAS ( VALUES ( 'MemberData'[member_key] ),
'TransactionData'[member_key] ),
        TREATAS (
            DATESINPERIOD ( 'DateTable'[Date], [QuarterEnd],
-1, QUARTER ),
                'TransactionData'[effective_date]
            )
        ),
    )

    "Total Employer T Amount",
    CALCULATE (
        SUMX (
            'TransactionData',
            COALESCE ( 'TransactionData'[credit_amount], 0 )
            - COALESCE (
abs('TransactionData'[debit_amount]), 0 )
        ),
        'TransactionData'[transaction_type] = "Employer",
        TREATAS ( VALUES ( 'MemberData'[member_key] ),
'TransactionData'[member_key] ),
        TREATAS (
            DATESINPERIOD ( 'DateTable'[Date], [QuarterEnd],
-1, QUARTER ),
                'TransactionData'[effective_date]
            )
        ),
    )

    // ----- Member -----
    "Last Member Contribution Date",
    CALCULATE (
        MAX ( 'TransactionData'[effective_date] ),
        'TransactionData'[transaction_type] = "Member",
        TREATAS ( VALUES ( 'MemberData'[member_key] ),
'TransactionData'[member_key] ),
        TREATAS (
            DATESINPERIOD ( 'DateTable'[Date], [QuarterEnd],

```

```

-1, QUARTER ),
        'TransactionData'[effective_date]
    )
),
),

"Last 2 Quarter Member Contributions",
CALCULATE (
    COUNT ( 'TransactionData'[unique_id] ),
    KEEPFILTERS ( 'TransactionData'[transaction_type] =
"Member" ),
        TREATAS ( VALUES ( 'MemberData'[member_key] ),
'[member_key] ),
        TREATAS (
            DATESINPERIOD ( 'DateTable'[Date], [QuarterEnd],
-2, QUARTER ),
        'TransactionData'[effective_date]
    )
),
),

"Total Member Contributions",
CALCULATE (
    COUNT ( 'TransactionData'[unique_id] ),
    'TransactionData'[transaction_type] = "Member",
        TREATAS ( VALUES ( 'MemberData'[member_key] ),
'[member_key] ),
        TREATAS (
            DATESINPERIOD ( 'DateTable'[Date], [QuarterEnd],
-1, QUARTER ),
        'TransactionData'[effective_date]
    )
),
),

"Total Member T Amount",
CALCULATE (
    SUMX (
        'TransactionData',
        COALESCE ( 'TransactionData'[credit_amount], 0 )
        - COALESCE (
abs ('TransactionData'[debit_amount]), 0 )
    ),
        'TransactionData'[transaction_type] = "Member",
        TREATAS ( VALUES ( 'MemberData'[member_key] ),
'[member_key] ),
        TREATAS (
            DATESINPERIOD ( 'DateTable'[Date], [QuarterEnd],
-1, QUARTER ),
        'TransactionData'[effective_date]
    )
)
)
)
)

```

## Power BI Measure Data Dictionary

- Quarter-on-Quarter Dashboard & Quarter-View Dashboard

Measure Name	Formula (DAX)	Description	Output Format	Calculation Logic
<b>Committed E Transaction Amount</b>	CALCULATE (CALCULATE (SUM(Member_Last_Contributions_ByQuarter[Total Employer T Amount]), Member_Last_Contributions_ByQuarter[Financial Status] = "Committed") / SUM(Member_Last_Contributions_ByQuarter[Total Employer T Amount]) * 100	Calculates the percentage of employer contribution transactions made by <i>Committed</i> members	Percentage	1. Sum employer transactions for <i>Committed</i> members. 2. Divide by total employer transactions. 3. Multiply by 100.
<b>Committed M Transaction Amount</b>	CALCULATE (SUM(Member_Last_Contributions_ByQuarter[Total Member T Amount]), Member_Last_Contributions_ByQuarter[Financial Status] = "Committed") / SUM(Member_Last_Contributions_ByQuarter[Total Member T Amount]) * 100	Calculates the percentage of member contribution transactions made by <i>Committed</i> members	Percentage	1. Sum member transactions for <i>Committed</i> members. 2. Divide by total member transactions. 3. Multiply by 100.
<b>Committed R Transaction Amount</b>	CALCULATE (SUM(Member_Last_Contributions_ByQuarter[Total Rollover T Amount]), Member_Last_Contributions_ByQuarter[Financial Status] = "Committed") / SUM(Member_Last_Contributions_ByQuarter[Total Rollover T Amount]) * 100	Calculates the percentage of rollover contribution transactions made by <i>Committed</i> members.	Percentage	1. Sum rollover transactions for <i>Committed</i> members. 2. Divide by total rollover transactions. 3. Multiply by 100.
<b>Committed Transaction Amount</b>	CALCULATE (SUM(Member_Last_Contributions_ByQuarter[Total T Amount]), Member_Last_Contributions_ByQuarter[Financial Status] = "Committed") / SUM(Member_Last_Contributions_ByQuarter[Total T Amount]) * 100	Calculates the percentage of total contribution transactions made by members with a " <i>Committed</i> " financial status.	Percentage	1. Sum all total transactions for <i>Committed</i> members. 2. Divide by total transactions from all members. 3. Multiply by 100.

Measure Name	Formula (DAX)	Description	Output Format	Calculation Logic
<b>Disengaged E Transaction Amount</b>	CALCULATE (SUM(Member_Last_Contributions_ByQuarter[Total_Employer_T_Amount]), Member_Last_Contributions_ByQuarter[Financial_Status] = "Disengaged") / SUM(Member_Last_Contributions_ByQuarter[Total_Employer_T_Amount]) * 100	Calculates the percentage of employer contributions made by <i>Disengaged</i> members.	Percentage	1. Sum employer transactions for <i>Disengaged</i> members. 2. Divide by total employer transactions. 3. Multiply by 100
<b>Disengaged M Transaction Amount</b>	CALCULATE (SUM(Member_Last_Contributions_ByQuarter[Total_Member_T_Amount]), Member_Last_Contributions_ByQuarter[Financial_Status] = "Disengaged") / SUM(Member_Last_Contributions_ByQuarter[Total_Member_T_Amount]) * 100	Calculates the percentage of member contributions made by <i>Disengaged</i> members.	Percentage	1. Sum member transactions for <i>Disengaged</i> members. 2. Divide by total member transactions. 3. Multiply by 100
<b>Disengaged R Transaction Amount</b>	CALCULATE (SUM(Member_Last_Contributions_ByQuarter[Total_Rollover_T_Amount]), Member_Last_Contributions_ByQuarter[Financial_Status] = "Disengaged") / SUM(Member_Last_Contributions_ByQuarter[Total_Rollover_T_Amount]) * 100	Calculates the percentage of rollover contributions made by <i>Disengaged</i> members.	Percentage	1. Sum rollover transactions for <i>Disengaged</i> members. 2. Divide by total rollover transactions. 3. Multiply by 100
<b>Disengaged Transaction Amount</b>	CALCULATE (SUM(Member_Last_Contributions_ByQuarter[Total_T_Amount]), Member_Last_Contributions_ByQuarter[Financial_Status] = "Disengaged") / SUM(Member_Last_Contributions_ByQuarter[Total_T_Amount]) * 100	Calculates the percentage of total contribution transactions made by members classified as <i>Disengaged</i> .	Percentage	1. Sum all total transactions for <i>Disengaged</i> members. 2. Divide by total transactions from all members. 3. Multiply by 100.

Measure Name	Formula (DAX)	Description	Output Format	Calculation Logic
<b>Disengaged Count Last Q</b>	CALCULATE ( Member_Last_Contributions_ByQuarter[ActiveMembers], FILTER( Member_Last_Contributions_ByQuarter, Member_Last_Contributions_ByQuarter[Last Transaction Date] < STARTOFCQUARTER(DataTable[Date]) && Member_Last_Contributions_ByQuarter[Financial Status] = "Disengaged" ) )	Counts the number of previously active members whose last transaction occurred before the start of the current quarter and whose financial status is Disengaged	Percentage	1. Identify the start of the current quarter. 2. Filter members whose last transaction occurred before that date. 3. Restrict to those with <i>Disengaged</i> status. 4. Count using the ActiveMembers measure logic
<b>Exited Members</b>	VAR StartDate = STARTOFCQUARTER('DataTable'[Date]) VAR EndDate = ENDOFCQUARTER('DataTable'[Date]) RETURN CALCULATE ( DISTINCTCOUNT('memberdata'[member_key]), FILTER('memberdata', 'memberdata'[exit_date] >= StartDate && 'memberdata'[exit_date] <= EndDate ) )	Counts the number of members who exited during the current quarter based on their recorded exit dates.	Whole number	1. Identify the start and end dates of the current quarter. 2. Filter members whose exit_date falls within this range. 3. Count the distinct member keys meeting the condition.
<b>Employer Contribution Rate</b>	SUM(Member_Last_Contributions_ByQuarter[Total Employer Contributions]) / SUM(Member_Last_Contributions_ByQuarter[Total Transactions])	Calculates the proportion of total contributions made by employers during the period.	Percentage	1. Sum all employer contributions. 2. Divide by the total of all transaction types (employer, member, rollover). 3. Optionally multiply by 100 for percentage representation.
<b>Member Contribution Rate</b>	SUM(Member_Last_Contributions_ByQuarter[Total Member Contributions]) / SUM(Member_Last_Contributions_ByQuarter[Total Transactions])	Calculates the proportion of total contributions made by members during the period.	Percentage	1. Sum all member contributions. 2. Divide by total transactions. 3. Optionally multiply by 100 for percentage representation.

Measure Name	Formula (DAX)	Description	Output Format	Calculation Logic
<b>Rollover Rate</b>	CALCULATE (SUM (Member_Last_Contributions_ByQuarter[Total Rollovers]) / SUM (Member_Last_Contributions_ByQuarter[Total Transactions]))	Calculates the proportion of total contributions that are rollovers (funds transferred from other accounts).	Percentage	1. Sum all rollover transactions. 2. Divide by total transactions. 3. Optionally multiply by 100 for percentage representation.
<b>Total_TA_by_Comfortable</b>	CALCULATE (SUM (CALCULATE (SUM (Member_Last_Contributions_ByQuarter[Total T Amount]), Member_Last_Contributions_ByQuarter[Financial Status] = "Comfortable"))))	Calculates the total transaction amount contributed by members whose financial status is "Comfortable."	Dollars	1. Filter the dataset for members with <i>Comfortable</i> financial status. 2. Sum the total transaction amount (Total T Amount) for those members.
<b>Total_TA_by_Committed</b>	CALCULATE (SUM (Member_Last_Contributions_ByQuarter[Total T Amount]), Member_Last_Contributions_ByQuarter[Financial Status] = "Committed"))	Calculates the total transaction amount contributed by members whose financial status is "Committed"	Dollars	1. Filter the dataset for members with <i>Committed</i> financial status. 2. Sum the total transaction amount (Total T Amount) for those members.
<b>Total_TA_by_Disengaged</b>	CALCULATE (SUM (Member_Last_Contributions_ByQuarter[Total T Amount]), Member_Last_Contributions_ByQuarter[Financial Status] = "Disengaged"))	Calculates the total transaction amount contributed by members whose financial status is "Disengaged."	Dollars	1. Filter the dataset for members with <i>Disengaged</i> financial status. 2. Sum the total transaction amount (Total T Amount) for those members.
<b>Active Members</b>	VAR SelectedDate = ENDOFQUARTER (DateTable[Date]) RETURN CALCULATE ( DISTINCTCOUNT (Member_Last_Contributions_ByQuarter[member_key]), FILTER ('memberdata', 'memberdata'[join_date] <= SelectedDate && (ISBLANK('memberdata'[exit_date])    ISBLANK('memberdata'[join_date])))	Calculates the total number of active members as of the end of each quarter	Whole Number	Calculates the number of members who joined on or before the quarter-end date and have not exited (exit date blank or after the quarter-end), returning the distinct count of active members for that quarter

Measure Name	Formula (DAX)	Description	Output Format	Calculation Logic
<b>Financial Status</b>	<pre> Financial Status =  SWITCH(TRUE(), AND(Member_Last_Contributions_ByQuarter[Last 2 Quarter Rollovers] &gt;= 1, OR(Member_Last_Contributions_ByQuarter[Last 2 Quarter Employer Contributions] &gt;= 1, Member_Last_Contributions_ByQuarter[Last 2 Quarter Member Contributions] &gt;= 1)), "Committed", OR(Member_Last_Contributions_ByQuarter[Last 2 Quarter Rollovers] &gt;= 1, OR(Member_Last_Contributions_ByQuarter[Last 2 Quarter Employer Contributions] &gt;= 1, Member_Last_Contributions_ByQuarter[Last 2 Quarter Member Contributions] &gt;= 1)), "Comfortable", "Disengaged") </pre>	Classifies members into Committed, Comfortable, or Disengaged based on their transaction and contribution activity over the last two quarters.	Whole number	Evaluates contribution and rollover activity over the last two quarters. If both rollover and contribution activity are present → <i>Committed</i> ; if some activity is present → <i>Comfortable</i> ; if no activity → <i>Disengaged</i> .

- Member Demographics Dashboard

Measure Name	Formula (DAX)	Description	Output Format	Calculation Logic
<b>Current Members</b>	<pre>Current Members = VAR EndDate = ENDOFQUARTER('DateTable'[Date]) RETURN CALCULATE (      DISTINCTCOUNT('MemberLast_Contributions_ByQuarter'[member_key]),     FILTER (         'memberdata',          'memberdata'[join_date] &lt;= EndDate &amp;&amp;         ( ISBLANK ('memberdata'[exit_date])            'memberdata'[exit_date] &gt; EndDate )     ) )</pre>	Calculates the total number of active members as of the end of each quarter	Whole Number	Calculates the number of members who joined on or before the quarter-end date and have not exited (exit date blank or after the quarter-end), returning the distinct count of active members for that quarter
<b>Exited Members</b>	<pre>Exited Members = VAR StartDate = STARTOFQUARTER('DateTable'[Date]) VAR EndDate = ENDOFQUARTER('DateTable'[Date]) RETURN CALCULATE (     DISTINCTCOUNT(         'memberdata'[member_key]),     FILTER (         'MemberData',          'memberdata'[exit_date] &gt;= StartDate &amp;&amp;         'memberdata'[exit_date] &lt;= EndDate     ) )</pre>	Counts the number of members who exited during the current quarter based on their recorded exit dates.	Whole number	Calculates the number of members who exit during the quarter (exit date after the quarter-start and before the quarter-end), returning the distinct count of active members for that quarter.
<b>Joined Members</b>	<pre>Joined Members = VAR StartDate = STARTOFQUARTER('DateTable'[Date]) VAR EndDate = ENDOFQUARTER('DateTable'[Date]) RETURN CALCULATE (</pre>	Calculates number of members who newly join within the selected quarter.	Whole number	Calculates the number of members who joined within the quarter-start date and the quarter-end date and have not exited (exit date blank or after the quarter-

	<pre> DISTINCTCOUNT( 'memberdata'[member_key] ),     FILTER ( 'MemberData', 'memberdata'[join_date] &gt;= StartDate &amp;&amp; 'memberdata'[join_date] &lt;= EndDate &amp;&amp; ( ISBLANK ( 'memberdata'[exit_date] )    'memberdata'[exit_date] &gt; EndDate ) ) ) </pre>			end), returning the distinct count of newly joined members for that quarter .
<b>Committed Current Members</b>	<pre> Committed Current Members = VAR SelectedDate = ENDOFQUARTER ( DateTable[Date] ) VAR CurrentIDsQ = CALCULATETABLE ( DISTINCT ( memberdata[member_key] ), FILTER ( ALL ( memberdata ), memberdata[join_date] &lt;= SelectedDate &amp;&amp; ( ISBLANK ( memberdata[exit_date] )    memberdata[exit_date] &gt; SelectedDate ) ) ) VAR CommittedIDsQ = CALCULATETABLE ( DISTINCT ( Member_Last_Contributions_ByQuarter[member_key] ), Member_Last_Contributions_ByQuarter[Financial_Status] = "Committed" ) RETURN COUNTRROWS ( INTERSECT ( CurrentIDsQ, CommittedIDsQ ) ) </pre>	Calculates number of Committed current members	Whole Number	Calculate the number of members whose financial status are “Committed” among those current/active members by evaluating the intersection between two temporary calculated tables.
<b>Committed Exited Members</b>	Committed Exited Members =	Calculates number of	Whole Number	Calculate the number of members

	<pre> VAR StartDate = STARTOFQUARTER('Date Table'[Date] ) VAR EndDate = ENDOFQUARTER('DateTa ble'[Date] ) VAR ExitedIDsQ = CALCULATETABLE ( DISTINCT( 'memberdata'[member_ key] ), FILTER ( ALL(memberdata), 'memberdata'[exit_da te] &gt;= StartDate &amp;&amp; 'memberdata'[exit_da te] &lt;= EndDate ) ) VAR CommittedIDsQ = CALCULATETABLE ( DISTINCT ( Member_Last_Contribu tions_ByQuarter[memb er_key] ), Member_Last_Contribu tions_ByQuarter[Finan cial_Status] = "Committed" ) RETURN COUNTRROWS ( INTERSECT ( ExitedIDsQ, CommittedIDsQ) ) </pre>	Committed exited members		whose financial status are “Committed” among those exited members by evaluating the intersection between two temporary calculated tables.
<b>Committed Joined Members</b>	<pre> Committed Joined Members = VAR StartDate = STARTOFQUARTER('Date Table'[Date] ) VAR EndDate = ENDOFQUARTER('DateTa ble'[Date] ) VAR JoinedIDsQ = CALCULATETABLE ( DISTINCT( 'memberdata'[member_ key] ), FILTER ( ALL(memberdata), 'memberdata'[join_da te] &gt;= StartDate &amp;&amp; </pre>	Calculates number of Committed newly joined members	Whole Number	Calculate the number of members whose financial status are “Committed” among those joined members by evaluating the intersection between two temporary calculated tables.

	<pre>'memberdata'[join_date] &lt;= EndDate &amp;&amp;                                 (                                 ISBLANK (                                 'memberdata'[exit_date] )                                    'memberdata'[exit_date] &gt; EndDate )                                 )                 ) VAR CommittedIDsQ =     CALCULATETABLE (         DISTINCT         (         Member_Last_Contributions_ByQuarter[member_key] ),         Member_Last_Contributions_ByQuarter[Financial Status] =         "Committed"         ) RETURN COUNTRows (     INTERSECT (         JoinedIDsQ,         CommittedIDsQ) )</pre>			
<b>Committed Active at Start Quarter</b>	<pre>Committed Active at Start Quarter = VAR StartDate = STARTOFLQUARTER('Date Table'[Date]) VAR IDs =     CALCULATETABLE (         VALUES(         Member_Last_Contributions_ByQuarter[member_key] ),         KEEPFILTERS(         Member_Last_Contributions_ByQuarter[Financial Status] =         "Committed" )         ) RETURN CALCULATE (     DISTINCTCOUNT(     memberdata[member_key] ),     TREATAS( IDs,     memberdata[member_key] ),     FILTER(         'memberdata',         'memberdata'[join_date] &lt;= StartDate &amp;&amp;</pre>	Calculates number of Committed active members at start of each quarter	Whole Number	Calculate the number of members whose financial status are “Committed” among those active members as of the start-date of the selected quarter and have not exited before that date (exit date blank or after the quarter-start).

	<pre> ( ISBLANK(memberdata[exit_date])    memberdata[exit_date] &gt; StartDate ) ) </pre>			
<b>Committed Churn Rate</b>	Committed Churn Rate = DIVIDE ( [Committed Exited Members], [Committed Active at Start Quarter], 0 ) * 100	Calculates churn rate of Committed members for each quarter	Percentage	Divides the number of Committed exited members by the number of Committed active members at the start of each quarter, then multiplies by 100 to return the ratio.
<b>Committed Joined Rate</b>	Committed Join Rate = DIVIDE ( [Committed Joined Members], [Committed Active at Start Quarter], 0 ) * 100	Calculate joined rate of Committed Members for each quarter	Percentage	Divides the number of Committed newly joined members by the number of Committed active members at the start of each quarter, then multiplies by 100 to return the ratio.
<b>Committed % Current Members</b>	Committed % Current Members = DIVIDE ( [Committed Current Members], [Current Members] ) * 100	Calculate shares of Committed members among active members as of the end of each quarter	Percentage	Divides the number of Committed current members by the total number of current members as of the end of each quarter, then multiplies by 100 to return the ratio.
<b>Committed % Exited Members</b>	Committed % Exited Members = DIVIDE ( [Committed Exited Members], [Exited Members] ) * 100	Calculate shares of Committed members among active members as of the end of each quarter	Percentage	Divides the number of Committed exited members by the total number of current members as of the end of each quarter, then multiplies by 100 to return the ratio.
<b>Committed % Joined Members</b>	Committed % Joined Members = DIVIDE ( [Committed Joined Members], [Joined Members] ) * 100	Calculate shares of Committed members among active members as of the end of each quarter	Percentage	Divides the number of Committed newly joined members by the total number of current members as of the end of each quarter, then multiplies by 100 to return the ratio.
<b>Comfortable Current Members</b>	Comfortable Current Members =	Calculates number of Comfortable	Whole Number	Calculate the number of members whose financial

	<pre> VAR SelectedDate = ENDOFQUARTER ( DateTable[Date] ) VAR CurrentIDsQ = CALCULATETABLE (     DISTINCT ( memberdata[member_key] ),         FILTER (             ALL ( memberdata ), memberdata[join_date] ] &lt;= SelectedDate &amp;&amp; ( ISBLANK ( memberdata[exit_date] ] )    memberdata[exit_date] ] &gt; SelectedDate ) ) ) VAR CommittedIDsQ = CALCULATETABLE (     DISTINCT ( Member_Last_Contributions_ByQuarter[memb er_key] ),  Member_Last_Contributions_ByQuarter[Finan cial_Status] = "Committed" )  VAR Comfortable_Raw = CALCULATETABLE (     DISTINCT ( Member_Last_Contributions_ByQuarter[memb er_key] ),  Member_Last_Contributions_ByQuarter[Finan cial_Status] = "Comfortable" )  VAR ComfortableIDsQ = EXCEPT ( Comfortable_Raw, CommittedIDsQ )  RETURN COUNTRROWS ( INTERSECT ( CurrentIDsQ, ComfortableIDsQ ) ) </pre>	current members	status are "Comfortable" among those current/active members (ensuring members that qualify as "Committed" are not double counted as "Comfortable") by evaluating the intersection between temporary calculated tables.
--	--	-----------------	--

<b>Comfortable Exited Members</b>	<pre> Comfortable_Exited_Members = VAR StartDate = STARTOFCALCULATE('Date Table'[Date]) VAR EndDate = ENDOFCALCULATE('Date Table'[Date]) VAR ExitedIDsQ = CALCULATETABLE (     DISTINCT(         'memberdata'[member_key] ),         FILTER (             ALL(memberdata),             'memberdata'[exit_date] &gt;= StartDate &amp;&amp;             'memberdata'[exit_date] &lt;= EndDate         ) ) VAR CommittedIDsQ = CALCULATETABLE (     DISTINCT(         Member_Last_Contributions_ByQuarter[member_key] ),         Member_Last_Contributions_ByQuarter[Financial_Status] =         "Committed" ) VAR Comfortable_Raw = CALCULATETABLE (     DISTINCT(         Member_Last_Contributions_ByQuarter[member_key] ),         Member_Last_Contributions_ByQuarter[Financial_Status] =         "Comfortable" ) VAR ComfortableIDsQ = EXCEPT (     Comfortable_Raw,     CommittedIDsQ ) RETURN COUNTROWS (     INTERSECT ( </pre>	Calculates number of Comfortable exited members	Whole Number	Calculate the number of members whose financial status are "Comfortable" among those exited members (ensuring members that qualify as "Committed" are not double counted as "Comfortable") by evaluating the intersection between temporary calculated tables.
-----------------------------------	---	---	--------------	--

	ExitedIDsQ, ComfortableIDsQ ) )			
<b>Comfortable Joined Members</b>	<pre> Comfortable Joined Members = VAR StartDate = STARTOFCALCULATE('Date Table'[Date] ) VAR EndDate = ENDOFCALCULATE('DateTa ble'[Date] ) VAR JoinedIDsQ = CALCULATETABLE (     DISTINCT(         'memberdata'[member_ key] ),         FILTER (             ALL(memberdata),             'memberdata'[join_da te] &gt;= StartDate &amp;&amp;             'memberdata'[join_da te] &lt;= EndDate &amp;&amp;             (                 ISBLANK (                     'memberdata'[exit_da te] )                        'memberdata'[exit_da te] &gt; EndDate )             )         ) VAR CommittedIDsQ = CALCULATETABLE (     DISTINCT     (         Member_Last_Contribu tions_ByQuarter[memb er_key] ),         Member_Last_Contribu tions_ByQuarter[Fin ancial Status] =         "Committed"     )  VAR Comfortable_Raw = CALCULATETABLE (     DISTINCT (         Member_Last_Contribu tions_ByQuarter[memb er_key] ),         Member_Last_Contribu tions_ByQuarter[Fin ancial Status] =         "Comfortable"     ) </pre>	Calculates number of Comfortable newly joined members	Whole Number	Calculate the number of members whose financial status are "Comfortable" among those newly joined members (ensuring members that qualify as "Committed" are not double counted as "Comfortable") by evaluating the intersection between temporary calculated tables.

	<pre> VAR ComfortableIDsQ = EXCEPT (     Comfortable_Raw,     CommittedIDsQ )  RETURN COUNTRows (     INTERSECT (         JoinedIDsQ,         ComfortableIDsQ ) ) </pre>			
<b>Comfortable Active at Start Quarter</b>	<pre> Comfortable Active at Start Quarter = VAR StartDate = STARTOFLONGEST('Date Table'[Date]) VAR IDs =     CALCULATETABLE(         VALUES(             Member_Last_Contributions_ByQuarter[member_key]),         KEEPFILTERS(             Member_Last_Contributions_ByQuarter[Financial Status] =                 "Comfortable")) ) RETURN CALCULATE(     DISTINCTCOUNT(         memberdata[member_key]),     TREATAS( IDs,         memberdata[member_key]),     FILTER(         'memberdata',         'memberdata'[join_date] &lt;= StartDate &amp;&amp;             (                 ISBLANK(memberdata[exit_date])                    memberdata[exit_date] &gt; StartDate)     ) ) </pre>	Calculates number of Comfortable active members at start of each quarter	Whole Number	Calculate the number of members whose financial status are “Comfortable” among those active members as of the start-date of the selected quarter and have not exited before that date (exit date blank or after the quarter-start)
<b>Comfortable Churn Rate</b>	<pre> Comfortable Churn Rate = DIVIDE (     [Comfortable Exited Members],     [Comfortable Active at Start Quarter], 0 ) * 100 </pre>	Calculates churn rate of Comfortable members for each quarter	Percentage	Divides the number of Comfortable exited members by the number of Comfortable active members at the start of each quarter, then multiplies by 100 to return the ratio.
<b>Comfortable Joined Rate</b>	Comfortable Join Rate =	Calculate joined rate of	Percentage	Divides the number of Comfortable

	DIVIDE ( [Comfortable Joined Members], [Comfortable Active at Start Quarter], 0 ) * 100	Comfortable Members for each quarter		newly joined Comfortable by the number of Comfortable active members at the start of each quarter, then multiplies by 100 to return the ratio.
<b>Comfortable % Current Members</b>	Comfortable % Current Members = DIVIDE( [Comfortable Current Members], [Current Members] ) * 100	Calculate shares of Comfortable members among active members as of the end of each quarter	Percentage	Divides the number of Comfortable current members by the total number of current members as of the end of each quarter, then multiplies by 100 to return the ratio.
<b>Comfortable % Exited Members</b>	Comfortable % Exited Members = DIVIDE( [Comfortable Exited Members], [Exited Members] ) * 100	Calculate shares of Comfortable members among active members as of the end of each quarter	Percentage	Divides the number of Comfortable exited members by the total number of current members as of the end of each quarter, then multiplies by 100 to return the ratio.
<b>Comfortable % Joined Members</b>	Comfortable % Joined Members = DIVIDE( [Comfortable Joined Members], [Joined Members] ) * 100	Calculate shares of Comfortable members among active members as of the end of each quarter	Percentage	Divides the number of Comfortable newly joined members by the total number of current members as of the end of each quarter, then multiplies by 100 to return the ratio.
<b>Disengaged Current Members</b>	Disengaged Current Members = VAR EndDate = ENDOFQUARTER ( DateTable[Date] ) VAR CurrentIDsQ = CALCULATETABLE ( VALUES ( 'memberdata'[member_key] ), REMOVEFILTERS ( 'memberdata' ), 'memberdata'[join_date] <= EndDate, ISBLANK ( 'memberdata'[exit_date] )    'memberdata'[exit_date] > EndDate	Calculates number of Disengaged current members	Whole Number	Calculate the number of members whose financial status are “Disengaged” among those current/active members (ensuring members that qualify as “Committed” and “Comfortable” are not double counted as “Comfortable”) by evaluating the intersection between temporary calculated tables.

	<pre>         ) VAR CommittedIDsQ =     CALCULATETABLE (         DISTINCT ( Member_Last_Contribu tions_ByQuarter[memb er_key] ),  Member_Last_Contribu tions_ByQuarter[Finan cial Status] = "Committed" ) VAR Comfortable_Raw =     CALCULATETABLE (         DISTINCT ( Member_Last_Contribu tions_ByQuarter[memb er_key] ),  Member_Last_Contribu tions_ByQuarter[Finan cial Status] = "Comfortable" ) VAR Disengaged_Raw =     CALCULATETABLE (         DISTINCT ( Member_Last_Contribu tions_ByQuarter[memb er_key] ),  Member_Last_Contribu tions_ByQuarter[Finan cial Status] = "Disengaged" ) VAR ComfortableIDsQ = EXCEPT ( Comfortable_Raw, CommittedIDsQ ) VAR DisengagedIDsQ = EXCEPT ( Disengaged_Raw, UNION ( CommittedIDsQ, ComfortableIDsQ ) ) RETURN COUNTRows ( INTERSECT ( CurrentIDsQ, DisengagedIDsQ ) ) </pre>			
<b>Disengaged Exited Members</b>	Disengaged Exited Members = VAR StartDate = STARTOFLQUARTER('Date Table'[Date] ) VAR EndDate = ENDOFLQUARTER('DateTa ble'[Date] )	Calculates number of Disengaged exited members	Whole Number	Calculate the number of members whose financial status are “Disengaged” among those exited members (ensuring

```

VAR ExitedIDsQ =
    CALCULATETABLE (
        DISTINCT(
            'memberdata'[member_key] ),
            FILTER (
                ALL(memberdata),
                'memberdata'[exit_date] >= StartDate &&
                'memberdata'[exit_date] <= EndDate
            )
        )
VAR CommittedIDsQ =
    CALCULATETABLE (
        DISTINCT(
            Member_Last_Contributions_ByQuarter[member_key] ),
            Member_Last_Contributions_ByQuarter[Financial Status] =
            "Committed"
        )
VAR Comfortable_Raw =
    CALCULATETABLE (
        DISTINCT (
            Member_Last_Contributions_ByQuarter[member_key] ),
            Member_Last_Contributions_ByQuarter[Financial Status] =
            "Comfortable"
        )
VAR Disengaged_Raw =
    CALCULATETABLE (
        DISTINCT (
            Member_Last_Contributions_ByQuarter[member_key] ),
            Member_Last_Contributions_ByQuarter[Financial Status] =
            "Disengaged"
        )
VAR ComfortableIDsQ =
    EXCEPT (
        Comfortable_Raw,
        CommittedIDsQ )
VAR DisengagedIDsQ =
    EXCEPT (

```

members that qualify as "Committed" and "Comfortable" are not double counted as "Comfortable") by evaluating the intersection between temporary calculated tables.

	<pre> Disengaged_Raw, UNION ( CommittedIDsQ, ComfortableIDsQ ) ) RETURN COUNTRROWS ( INTERSECT ( ExitedIDsQ, DisengagedIDsQ ) ) </pre>			
<b>Disengaged Joined Members</b>	<pre> Disengaged Joined Members = VAR StartDate = STARTOFQUARTER('Date Table'[Date] ) VAR EndDate = ENDOFQUARTER('DateTa ble'[Date] ) VAR JoinedIDsQ = CALCULATETABLE ( DISTINCT( 'memberdata'[member_ key] ), FILTER ( ALL(memberdata), 'memberdata'[join_da te] &gt;= StartDate &amp;&amp; 'memberdata'[join_da te] &lt;= EndDate &amp;&amp; ( ISBLANK ( 'memberdata'[exit_da te] )    'memberdata'[exit_da te] &gt; EndDate ) ) ) VAR CommittedIDsQ = CALCULATETABLE ( DISTINCT ( Member_Last_Contribu tions_ByQuarter[memb er_key] ), Member_Last_Contribu tions_ByQuarter[Finan cial_Status] = "Committed" ) VAR Comfortable_Raw = CALCULATETABLE ( DISTINCT ( Member_Last_Contribu tions_ByQuarter[memb er_key] ), </pre>	Calculates number of Disengaged newly joined members	Whole Number	Calculate the number of members whose financial status are "Disengaged" among those newly joined members (ensuring members that qualify as "Committed" and "Comfortable" are not double counted as "Comfortable") by evaluating the intersection between temporary calculated tables.

	<pre> Member_Last_Contributions_ByQuarter[Financial Status] = "Comfortable" ) VAR Disengaged_Raw = CALCULATETABLE ( DISTINCT ( Member_Last_Contributions_ByQuarter[member_key] ),  Member_Last_Contributions_ByQuarter[Financial Status] = "Disengaged" ) VAR ComfortableIDsQ = EXCEPT ( Comfortable_Raw, CommittedIDsQ ) VAR DisengagedIDsQ = EXCEPT ( Disengaged_Raw, UNION ( CommittedIDsQ, ComfortableIDsQ ) ) RETURN COUNTRows ( INTERSECT ( JoinedIDsQ, DisengagedIDsQ ) ) </pre>			
<b>Disengaged Active at Start Quarter</b>	<pre> Disengaged Active at Start Quarter = VAR StartDate = STARTOFAQTER('Date Table'[Date]) VAR IDs = CALCULATETABLE( VALUES( Member_Last_Contributions_ByQuarter[member_key] ), KEEPFILTERS( Member_Last_Contributions_ByQuarter[Financial Status] = "Disengaged" ) ) RETURN CALCULATE( DISTINCTCOUNT( memberdata[member_key] ), TREATAS( IDs, memberdata[member_key] ), FILTER( 'memberdata', </pre>	Calculates number of Disengaged active members at start of each quarter	Whole Number	Calculate the number of members whose financial status are "Disengaged" among those active members as of the start-date of the selected quarter and have not exited before that date (exit date blank or after the quarter-start)

	'memberdata'[join_date] <= StartDate && ( ISBLANK(memberdata[exit_date])    memberdata[exit_date] > StartDate ) ) )			
<b>Disengaged Churn Rate</b>	Disengaged Churn Rate = DIVIDE ( [Disengaged Exited Members], [Disengaged Active at Start Quarter], 0 ) * 100	Calculates churn rate of Disengaged members for each quarter	Percentage	Divides the number of Disengaged exited members by the number of Disengaged active members at the start of each quarter, then multiplies by 100 to return the ratio.
<b>Disengaged Joined Rate</b>	Disengaged Join Rate = DIVIDE ( [Disengaged Joined Members], [Disengaged Active at Start Quarter], 0 ) * 100	Calculate joined rate of Disengaged Members for each quarter	Percentage	Divides the number of Disengaged newly joined members by the number of Disengaged active members at the start of each quarter, then multiplies by 100 to return the ratio.
<b>Disengaged % Current Members</b>	Disengaged % Current Members = DIVIDE( [Disengaged Current Members], [Current Members] ) * 100	Calculate shares of Disengaged members among active members as of the end of each quarter	Percentage	Divides the number of Disengaged current members by the total number of current members as of the end of each quarter, then multiplies by 100 to return the ratio.
<b>Disengaged % Exited Members</b>	Disengaged % Exited Members = DIVIDE( [Disengaged Exited Members], [Exited Members] ) * 100	Calculate shares of Disengaged members among active members as of the end of each quarter	Percentage	Divides the number of Disengaged exited members by the total number of current members as of the end of each quarter, then multiplies by 100 to return the ratio.
<b>Disengaged % Joined Members</b>	Disengaged % Joined Members = DIVIDE( [Disengaged Joined Members], [Joined Members] ) * 100	Calculate shares of Disengaged members among active members as of the end of each quarter	Percentage	Divides the number of Disengaged newly joined members by the total number of current members as of the end of each quarter, then multiplies by 100 to return the ratio.

<b>Measure Name</b>	<b>Formula (DAX)</b>	<b>Description</b>	<b>Output Format</b>	<b>Calculation Logic</b>
<b>Current Members</b>	<pre>Current Members = VAR EndDate = ENDOFQUARTER(DateTable[Date]) RETURN CALCULATE (      DISTINCTCOUNT(Member         _Last_Contributions_         ByQuarter[member_key]     ),     FILTER (          'memberdata',          'memberdata'[join_da te] &lt;= EndDate &amp;&amp;         ( ISBLANK (             'memberdata'[exit_da te] )                'memberdata'[exit_da te] &gt; EndDate )     ) )</pre>	Calculates the total number of active members as of the end of each quarter	Whole Number	Calculates the number of members who joined on or before the quarter-end date and have not exited (exit date blank or after the quarter-end), returning the distinct count of active members for that quarter
<b>Exited Members</b>	<pre>Exited Members = VAR StartDate = STARTOFQUARTER('Date Table'[Date] ) VAR EndDate = ENDOFQUARTER('DateTa ble'[Date] ) RETURN CALCULATE (     DISTINCTCOUNT(         'memberdata'[member_ key] ),     FILTER (          'MemberData',          'memberdata'[exit_da te] &gt;= StartDate &amp;&amp;          'memberdata'[exit_da te] &lt;= EndDate     ) )</pre>	Counts the number of members who exited during the current quarter based on their recorded exit dates.	Whole number	Calculates the number of members who exit during the quarter (exit date after the quarter-start and before the quarter-end), returning the distinct count of active members for that quarter.
<b>Joined Members</b>	<pre>Joined Members = VAR StartDate = STARTOFQUARTER('Date Table'[Date] ) VAR EndDate = ENDOFQUARTER('DateTa ble'[Date] ) RETURN CALCULATE (</pre>	Calculates number of members who newly join within the selected quarter.	Whole number	Calculates the number of members who joined within the quarter-start date and the quarter-end date and have not exited (exit date blank or after the quarter-

	<pre> DISTINCTCOUNT( 'memberdata'[member_key] ),     FILTER ( 'MemberData', 'memberdata'[join_date] &gt;= StartDate &amp;&amp; 'memberdata'[join_date] &lt;= EndDate &amp;&amp; ( ISBLANK ( 'memberdata'[exit_date] )    'memberdata'[exit_date] &gt; EndDate ) ) ) </pre>			end), returning the distinct count of newly joined members for that quarter .
<b>Committed Current Members</b>	<pre> Committed Current Members = VAR SelectedDate = ENDOFQUARTER ( DateTable[Date] ) VAR CurrentIDsQ = CALCULATETABLE ( DISTINCT ( memberdata[member_key] ), FILTER ( ALL ( memberdata ), memberdata[join_date] &lt;= SelectedDate &amp;&amp; ( ISBLANK ( memberdata[exit_date] )    memberdata[exit_date] &gt; SelectedDate ) ) ) VAR CommittedIDsQ = CALCULATETABLE ( DISTINCT ( Member_Last_Contributions_ByQuarter[member_key] ), Member_Last_Contributions_ByQuarter[Financial_Status] = "Committed" ) RETURN COUNTRROWS ( INTERSECT ( CurrentIDsQ, CommittedIDsQ ) ) </pre>	Calculates number of Committed current members	Whole Number	Calculate the number of members whose financial status are “Committed” among those current/active members by evaluating the intersection between two temporary calculated tables.
<b>Committed Exited Members</b>	Committed Exited Members =	Calculates number of	Whole Number	Calculate the number of members

	<pre> VAR StartDate = STARTOFQUARTER('Date Table'[Date] ) VAR EndDate = ENDOFQUARTER('DateTa ble'[Date] ) VAR ExitedIDsQ = CALCULATETABLE ( DISTINCT( 'memberdata'[member_ key] ), FILTER ( ALL(memberdata), 'memberdata'[exit_da te] &gt;= StartDate &amp;&amp; 'memberdata'[exit_da te] &lt;= EndDate ) ) VAR CommittedIDsQ = CALCULATETABLE ( DISTINCT ( Member_Last_Contribu tions_ByQuarter[memb er_key] ), Member_Last_Contribu tions_ByQuarter[Finan cial_Status] = "Committed" ) RETURN COUNTRROWS ( INTERSECT ( ExitedIDsQ, CommittedIDsQ) ) </pre>	Committed exited members		whose financial status are “Committed” among those exited members by evaluating the intersection between two temporary calculated tables.
<b>Committed Joined Members</b>	<pre> Committed Joined Members = VAR StartDate = STARTOFQUARTER('Date Table'[Date] ) VAR EndDate = ENDOFQUARTER('DateTa ble'[Date] ) VAR JoinedIDsQ = CALCULATETABLE ( DISTINCT( 'memberdata'[member_ key] ), FILTER ( ALL(memberdata), 'memberdata'[join_da te] &gt;= StartDate &amp;&amp; </pre>	Calculates number of Committed newly joined members	Whole Number	Calculate the number of members whose financial status are “Committed” among those joined members by evaluating the intersection between two temporary calculated tables.

	<pre>'memberdata'[join_date] &lt;= EndDate &amp;&amp;                                 (                                 ISBLANK (                                 'memberdata'[exit_date] )                                    'memberdata'[exit_date] &gt; EndDate )                                 )                 ) VAR CommittedIDsQ =     CALCULATETABLE (         DISTINCT         (         Member_Last_Contributions_ByQuarter[member_key] ),         Member_Last_Contributions_ByQuarter[Financial Status] =         "Committed"         ) RETURN COUNTRows ( INTERSECT ( JoinedIDsQ, CommittedIDsQ) )</pre>			
<b>Committed Active at Start Quarter</b>	<pre>Committed Active at Start Quarter = VAR StartDate = STARTOFQUARTER('Date Table'[Date]) VAR IDs =     CALCULATETABLE (         VALUES(         Member_Last_Contributions_ByQuarter[member_key] ),         KEEPFILTERS(         Member_Last_Contributions_ByQuarter[Financial Status] =         "Committed" )         ) RETURN CALCULATE (     DISTINCTCOUNT(     memberdata[member_key] ),     TREATAS( IDs,     memberdata[member_key] ),     FILTER(         'memberdata',         'memberdata'[join_date] &lt;= StartDate &amp;&amp;</pre>	Calculates number of Committed active members at start of each quarter	Whole Number	Calculate the number of members whose financial status are “Committed” among those active members as of the start-date of the selected quarter and have not exited before that date (exit date blank or after the quarter-start).

	<pre> ( ISBLANK(memberdata[exit_date])    memberdata[exit_date] &gt; StartDate ) ) </pre>			
<b>Committed Churn Rate</b>	Committed Churn Rate = DIVIDE ( [Committed Exited Members], [Committed Active at Start Quarter], 0 ) * 100	Calculates churn rate of Committed members for each quarter	Percentage	Divides the number of Committed exited members by the number of Committed active members at the start of each quarter, then multiplies by 100 to return the ratio.
<b>Committed Joined Rate</b>	Committed Join Rate = DIVIDE ( [Committed Joined Members], [Committed Active at Start Quarter], 0 ) * 100	Calculate joined rate of Committed Members for each quarter	Percentage	Divides the number of Committed newly joined members by the number of Committed active members at the start of each quarter, then multiplies by 100 to return the ratio.
<b>Committed % Current Members</b>	Committed % Current Members = DIVIDE ( [Committed Current Members], [Current Members] ) * 100	Calculate shares of Committed members among active members as of the end of each quarter	Percentage	Divides the number of Committed current members by the total number of current members as of the end of each quarter, then multiplies by 100 to return the ratio.
<b>Committed % Exited Members</b>	Committed % Exited Members = DIVIDE ( [Committed Exited Members], [Exited Members] ) * 100	Calculate shares of Committed members among active members as of the end of each quarter	Percentage	Divides the number of Committed exited members by the total number of current members as of the end of each quarter, then multiplies by 100 to return the ratio.
<b>Committed % Joined Members</b>	Committed % Joined Members = DIVIDE ( [Committed Joined Members], [Joined Members] ) * 100	Calculate shares of Committed members among active members as of the end of each quarter	Percentage	Divides the number of Committed newly joined members by the total number of current members as of the end of each quarter, then multiplies by 100 to return the ratio.
<b>Comfortable Current Members</b>	Comfortable Current Members =	Calculates number of Comfortable	Whole Number	Calculate the number of members whose financial

	<pre> VAR SelectedDate = ENDOFQUARTER ( DateTable[Date] ) VAR CurrentIDsQ = CALCULATETABLE (     DISTINCT ( memberdata[member_key] ),         FILTER (             ALL ( memberdata ), memberdata[join_date] ] &lt;= SelectedDate &amp;&amp; ( ISBLANK ( memberdata[exit_date] ] )    memberdata[exit_date] ] &gt; SelectedDate ) ) ) VAR CommittedIDsQ = CALCULATETABLE (     DISTINCT ( Member_Last_Contributions_ByQuarter[memb er_key] ),  Member_Last_Contributions_ByQuarter[Finan cial_Status] = "Committed" )  VAR Comfortable_Raw = CALCULATETABLE (     DISTINCT ( Member_Last_Contributions_ByQuarter[memb er_key] ),  Member_Last_Contributions_ByQuarter[Finan cial_Status] = "Comfortable" )  VAR ComfortableIDsQ = EXCEPT ( Comfortable_Raw, CommittedIDsQ )  RETURN COUNTRROWS ( INTERSECT ( CurrentIDsQ, ComfortableIDsQ ) ) </pre>	current members	status are "Comfortable" among those current/active members (ensuring members that qualify as "Committed" are not double counted as "Comfortable") by evaluating the intersection between temporary calculated tables.
--	--	-----------------	--

<b>Comfortable Exited Members</b>	<pre> Comfortable_Exited_Members = VAR StartDate = STARTOFCALCULATE('Date Table'[Date]) VAR EndDate = ENDOFCALCULATE('Date Table'[Date]) VAR ExitedIDsQ = CALCULATETABLE (     DISTINCT(         'memberdata'[member_key] ),         FILTER (             ALL(memberdata),             'memberdata'[exit_date] &gt;= StartDate &amp;&amp;             'memberdata'[exit_date] &lt;= EndDate         ) ) VAR CommittedIDsQ = CALCULATETABLE (     DISTINCT(         Member_Last_Contributions_ByQuarter[member_key] ),         Member_Last_Contributions_ByQuarter[Financial_Status] =         "Committed" ) VAR Comfortable_Raw = CALCULATETABLE (     DISTINCT(         Member_Last_Contributions_ByQuarter[member_key] ),         Member_Last_Contributions_ByQuarter[Financial_Status] =         "Comfortable" ) VAR ComfortableIDsQ = EXCEPT (     Comfortable_Raw,     CommittedIDsQ ) RETURN COUNTROWS (     INTERSECT ( </pre>	Calculates number of Comfortable exited members	Whole Number	Calculate the number of members whose financial status are "Comfortable" among those exited members (ensuring members that qualify as "Committed" are not double counted as "Comfortable") by evaluating the intersection between temporary calculated tables.
-----------------------------------	---	---	--------------	--

	ExitedIDsQ, ComfortableIDsQ ) )			
<b>Comfortable Joined Members</b>	<pre> Comfortable Joined Members = VAR StartDate = STARTOFQUARTER('Date Table'[Date] ) VAR EndDate = ENDOFQUARTER('DateTa ble'[Date] ) VAR JoinedIDsQ = CALCULATETABLE ( DISTINCT( 'memberdata'[member_ key] ), FILTER ( ALL(memberdata), 'memberdata'[join_da te] &gt;= StartDate &amp;&amp; 'memberdata'[join_da te] &lt;= EndDate &amp;&amp; ( ISBLANK ( 'memberdata'[exit_da te] )    'memberdata'[exit_da te] &gt; EndDate ) ) ) VAR CommittedIDsQ = CALCULATETABLE ( DISTINCT ( Member_Last_Contribu tions_ByQuarter[memb er_key] ),  Member_Last_Contribu tions_ByQuarter[Fin ancial Status] = "Committed" )  VAR Comfortable_Raw = CALCULATETABLE ( DISTINCT ( Member_Last_Contribu tions_ByQuarter[memb er_key] ),  Member_Last_Contribu tions_ByQuarter[Fin ancial Status] = "Comfortable" )</pre>	Calculates number of Comfortable newly joined members	Whole Number	Calculate the number of members whose financial status are “Comfortable” among those newly joined members (ensuring members that qualify as “Committed” are not double counted as “Comfortable”) by evaluating the intersection between temporary calculated tables.

	<pre> VAR ComfortableIDsQ = EXCEPT (     Comfortable_Raw,     CommittedIDsQ )  RETURN COUNTRows (     INTERSECT (         JoinedIDsQ,         ComfortableIDsQ ) ) </pre>			
<b>Comfortable Active at Start Quarter</b>	<pre> Comfortable Active at Start Quarter = VAR StartDate = STARTOFLONGEST('Date Table'[Date]) VAR IDs =     CALCULATETABLE(         VALUES(             Member_Last_Contributions_ByQuarter[member_key]),         KEEPFILTERS(             Member_Last_Contributions_ByQuarter[Financial Status] =                 "Comfortable")) ) RETURN CALCULATE(     DISTINCTCOUNT(         memberdata[member_key]),     TREATAS( IDs,         memberdata[member_key]),     FILTER(         'memberdata',         'memberdata'[join_date] &lt;= StartDate &amp;&amp;             (                 ISBLANK(memberdata[exit_date])                    memberdata[exit_date] &gt; StartDate)     ) ) </pre>	Calculates number of Comfortable active members at start of each quarter	Whole Number	Calculate the number of members whose financial status are “Comfortable” among those active members as of the start-date of the selected quarter and have not exited before that date (exit date blank or after the quarter-start)
<b>Comfortable Churn Rate</b>	<pre> Comfortable Churn Rate = DIVIDE (     [Comfortable Exited Members],     [Comfortable Active at Start Quarter], 0 ) * 100 </pre>	Calculates churn rate of Comfortable members for each quarter	Percentage	Divides the number of Comfortable exited members by the number of Comfortable active members at the start of each quarter, then multiplies by 100 to return the ratio.
<b>Comfortable Joined Rate</b>	Comfortable Join Rate =	Calculate joined rate of	Percentage	Divides the number of Comfortable

	DIVIDE ( [Comfortable Joined Members], [Comfortable Active at Start Quarter], 0 ) * 100	Comfortable Members for each quarter		newly joined Comfortable by the number of Comfortable active members at the start of each quarter, then multiplies by 100 to return the ratio.
<b>Comfortable % Current Members</b>	Comfortable % Current Members = DIVIDE( [Comfortable Current Members], [Current Members] ) * 100	Calculate shares of Comfortable members among active members as of the end of each quarter	Percentage	Divides the number of Comfortable current members by the total number of current members as of the end of each quarter, then multiplies by 100 to return the ratio.
<b>Comfortable % Exited Members</b>	Comfortable % Exited Members = DIVIDE( [Comfortable Exited Members], [Exited Members] ) * 100	Calculate shares of Comfortable members among active members as of the end of each quarter	Percentage	Divides the number of Comfortable exited members by the total number of current members as of the end of each quarter, then multiplies by 100 to return the ratio.
<b>Comfortable % Joined Members</b>	Comfortable % Joined Members = DIVIDE( [Comfortable Joined Members], [Joined Members] ) * 100	Calculate shares of Comfortable members among active members as of the end of each quarter	Percentage	Divides the number of Comfortable newly joined members by the total number of current members as of the end of each quarter, then multiplies by 100 to return the ratio.
<b>Disengaged Current Members</b>	Disengaged Current Members = VAR EndDate = ENDOFQUARTER ( DateTable[Date] ) VAR CurrentIDsQ = CALCULATETABLE ( VALUES ( 'memberdata'[member_key] ), REMOVEFILTERS ( 'memberdata' ), 'memberdata'[join_date] <= EndDate, ISBLANK ( 'memberdata'[exit_date] )    'memberdata'[exit_date] > EndDate	Calculates number of Disengaged current members	Whole Number	Calculate the number of members whose financial status are “Disengaged” among those current/active members (ensuring members that qualify as “Committed” and “Comfortable” are not double counted as “Comfortable”) by evaluating the intersection between temporary calculated tables.

	<pre>         ) VAR CommittedIDsQ =     CALCULATETABLE (         DISTINCT ( Member_Last_Contribu tions_ByQuarter[memb er_key] ),  Member_Last_Contribu tions_ByQuarter[Fin ancial Status] = "Committed" ) VAR Comfortable_Raw =     CALCULATETABLE (         DISTINCT ( Member_Last_Contribu tions_ByQuarter[memb er_key] ),  Member_Last_Contribu tions_ByQuarter[Fin ancial Status] = "Comfortable" ) VAR Disengaged_Raw =     CALCULATETABLE (         DISTINCT ( Member_Last_Contribu tions_ByQuarter[memb er_key] ),  Member_Last_Contribu tions_ByQuarter[Fin ancial Status] = "Disengaged" ) VAR ComfortableIDsQ = EXCEPT ( Comfortable_Raw, CommittedIDsQ ) VAR DisengagedIDsQ = EXCEPT ( Disengaged_Raw, UNION ( CommittedIDsQ, ComfortableIDsQ ) ) RETURN COUNTRows ( INTERSECT ( CurrentIDsQ, DisengagedIDsQ ) ) </pre>			
<b>Disengaged Exited Members</b>	Disengaged Exited Members = VAR StartDate = STARTOFLQUARTER('Date Table'[Date] ) VAR EndDate = ENDOFLQUARTER('DateTa ble'[Date] )	Calculates number of Disengaged exited members	Whole Number	Calculate the number of members whose financial status are “Disengaged” among those exited members (ensuring

```

VAR ExitedIDsQ =
    CALCULATETABLE (
        DISTINCT(
            'memberdata'[member_key] ),
            FILTER (
                ALL(memberdata),
                'memberdata'[exit_date] >= StartDate &&
                'memberdata'[exit_date] <= EndDate
            )
        )
VAR CommittedIDsQ =
    CALCULATETABLE (
        DISTINCT(
            Member_Last_Contributions_ByQuarter[member_key] ),
            Member_Last_Contributions_ByQuarter[Financial Status] =
            "Committed"
        )
VAR Comfortable_Raw =
    CALCULATETABLE (
        DISTINCT (
            Member_Last_Contributions_ByQuarter[member_key] ),
            Member_Last_Contributions_ByQuarter[Financial Status] =
            "Comfortable"
        )
VAR Disengaged_Raw =
    CALCULATETABLE (
        DISTINCT (
            Member_Last_Contributions_ByQuarter[member_key] ),
            Member_Last_Contributions_ByQuarter[Financial Status] =
            "Disengaged"
        )
VAR ComfortableIDsQ =
    EXCEPT (
        Comfortable_Raw,
        CommittedIDsQ )
VAR DisengagedIDsQ =
    EXCEPT (

```

members that qualify as "Committed" and "Comfortable" are not double counted as "Comfortable") by evaluating the intersection between temporary calculated tables.

	<pre> Disengaged_Raw, UNION ( CommittedIDsQ, ComfortableIDsQ ) ) RETURN COUNTRROWS ( INTERSECT ( ExitedIDsQ, DisengagedIDsQ ) ) </pre>			
<b>Disengaged Joined Members</b>	<pre> Disengaged Joined Members = VAR StartDate = STARTOFQUARTER('Date Table'[Date] ) VAR EndDate = ENDOFQUARTER('DateTa ble'[Date] ) VAR JoinedIDsQ = CALCULATETABLE ( DISTINCT( 'memberdata'[member_ key] ), FILTER ( ALL(memberdata), 'memberdata'[join_da te] &gt;= StartDate &amp;&amp; 'memberdata'[join_da te] &lt;= EndDate &amp;&amp; ( ISBLANK ( 'memberdata'[exit_da te] )    'memberdata'[exit_da te] &gt; EndDate ) ) ) VAR CommittedIDsQ = CALCULATETABLE ( DISTINCT ( Member_Last_Contribu tions_ByQuarter[memb er_key] ), Member_Last_Contribu tions_ByQuarter[Finan cial_Status] = "Committed" ) VAR Comfortable_Raw = CALCULATETABLE ( DISTINCT ( Member_Last_Contribu tions_ByQuarter[memb er_key] ), </pre>	Calculates number of Disengaged newly joined members	Whole Number	Calculate the number of members whose financial status are "Disengaged" among those newly joined members (ensuring members that qualify as "Committed" and "Comfortable" are not double counted as "Comfortable") by evaluating the intersection between temporary calculated tables.

	<pre> Member_Last_Contributions_ByQuarter[Financial Status] = "Comfortable" ) VAR Disengaged_Raw = CALCULATETABLE ( DISTINCT ( Member_Last_Contributions_ByQuarter[member_key] ),  Member_Last_Contributions_ByQuarter[Financial Status] = "Disengaged" ) VAR ComfortableIDsQ = EXCEPT ( Comfortable_Raw, CommittedIDsQ ) VAR DisengagedIDsQ = EXCEPT ( Disengaged_Raw, UNION ( CommittedIDsQ, ComfortableIDsQ ) ) RETURN COUNTRows ( INTERSECT ( JoinedIDsQ, DisengagedIDsQ ) ) </pre>			
<b>Disengaged Active at Start Quarter</b>	<pre> Disengaged Active at Start Quarter = VAR StartDate = STARTOFAQTER('Date Table'[Date]) VAR IDs = CALCULATETABLE( VALUES( Member_Last_Contributions_ByQuarter[member_key] ), KEEPFILTERS( Member_Last_Contributions_ByQuarter[Financial Status] = "Disengaged" ) ) RETURN CALCULATE( DISTINCTCOUNT( memberdata[member_key] ), TREATAS( IDs, memberdata[member_key] ), FILTER( 'memberdata', </pre>	Calculates number of Disengaged active members at start of each quarter	Whole Number	Calculate the number of members whose financial status are "Disengaged" among those active members as of the start-date of the selected quarter and have not exited before that date (exit date blank or after the quarter-start)

	'memberdata'[join_date] <= StartDate && ( ISBLANK(memberdata[exit_date])    memberdata[exit_date] > StartDate ) )			
<b>Disengaged Churn Rate</b>	Disengaged Churn Rate = DIVIDE ( [Disengaged Exited Members], [Disengaged Active at Start Quarter], 0 ) * 100	Calculates churn rate of Disengaged members for each quarter	Percentage	Divides the number of Disengaged exited members by the number of Disengaged active members at the start of each quarter, then multiplies by 100 to return the ratio.
<b>Disengaged Joined Rate</b>	Disengaged Join Rate = DIVIDE ( [Disengaged Joined Members], [Disengaged Active at Start Quarter], 0 ) * 100	Calculate joined rate of Disengaged Members for each quarter	Percentage	Divides the number of Disengaged newly joined members by the number of Disengaged active members at the start of each quarter, then multiplies by 100 to return the ratio.
<b>Disengaged % Current Members</b>	Disengaged % Current Members = DIVIDE( [Disengaged Current Members], [Current Members] ) * 100	Calculate shares of Disengaged members among active members as of the end of each quarter	Percentage	Divides the number of Disengaged current members by the total number of current members as of the end of each quarter, then multiplies by 100 to return the ratio.
<b>Disengaged % Exited Members</b>	Disengaged % Exited Members = DIVIDE( [Disengaged Exited Members], [Exited Members] ) * 100	Calculate shares of Disengaged members among active members as of the end of each quarter	Percentage	Divides the number of Disengaged exited members by the total number of current members as of the end of each quarter, then multiplies by 100 to return the ratio.
<b>Disengaged % Joined Members</b>	Disengaged % Joined Members = DIVIDE( [Disengaged Joined Members], [Joined Members] ) * 100	Calculate shares of Disengaged members among active members as of the end of each quarter	Percentage	Divides the number of Disengaged newly joined members by the total number of current members as of the end of each quarter, then multiplies by 100.