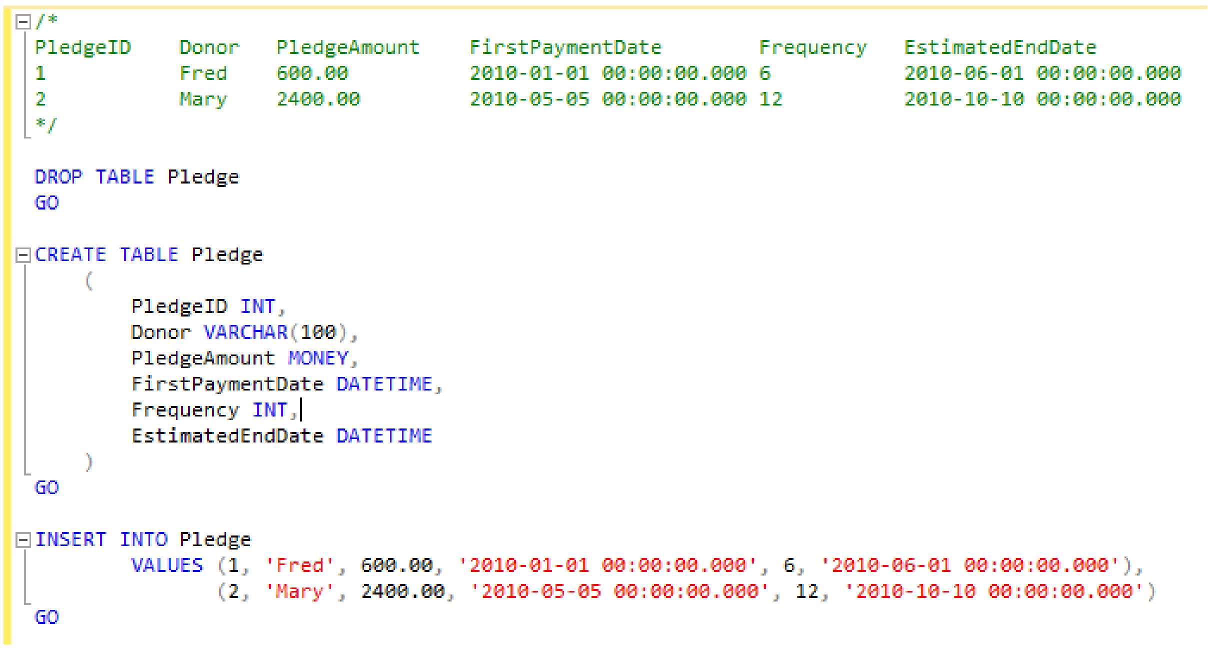
SQL Server – Create Detail From Summary

A while back I was asked what the best approach would be to produce a report in SQL Server Reporting Services (SSRS) that showed the predicted total donation value per month for a charity. The reason this had presented a challenge was that the data was captured via a Microsoft Dynamics solution and the records in the database were stored in a format similar to those shown below:



```
/*
PledgeID    Donor   PledgeAmount    FirstPaymentDate        Frequency   EstimatedEndDate
1           Fred    600.00          2010-01-01 00:00:00.000 6           2010-06-01 00:00:00.000
2           Mary    2400.00         2010-05-05 00:00:00.000 12          2010-10-10 00:00:00.000
*/

DROP TABLE Pledge
GO

CREATE TABLE Pledge
    (
        PledgeID INT,
        Donor VARCHAR(100),
        PledgeAmount MONEY,
        FirstPaymentDate DATETIME,
        Frequency INT,
        EstimatedEndDate DATETIME
    )
GO

INSERT INTO Pledge
        VALUES (1, 'Fred', 600.00, '2010-01-01 00:00:00.000', 6, '2010-06-01 00:00:00.000'),
               (2, 'Mary', 2400.00, '2010-05-05 00:00:00.000', 12, '2010-10-10 00:00:00.000')
GO
```

As you can see, each pledge made is stored as a single pledge amount, along with how many separate monthly payments they wish to make, and when they wish to start their payments.

My solution to this challenge was to convert the data into a set of monthly payments using a Common Table Expression (CTE) and then feed that translated data into an SSRS report.

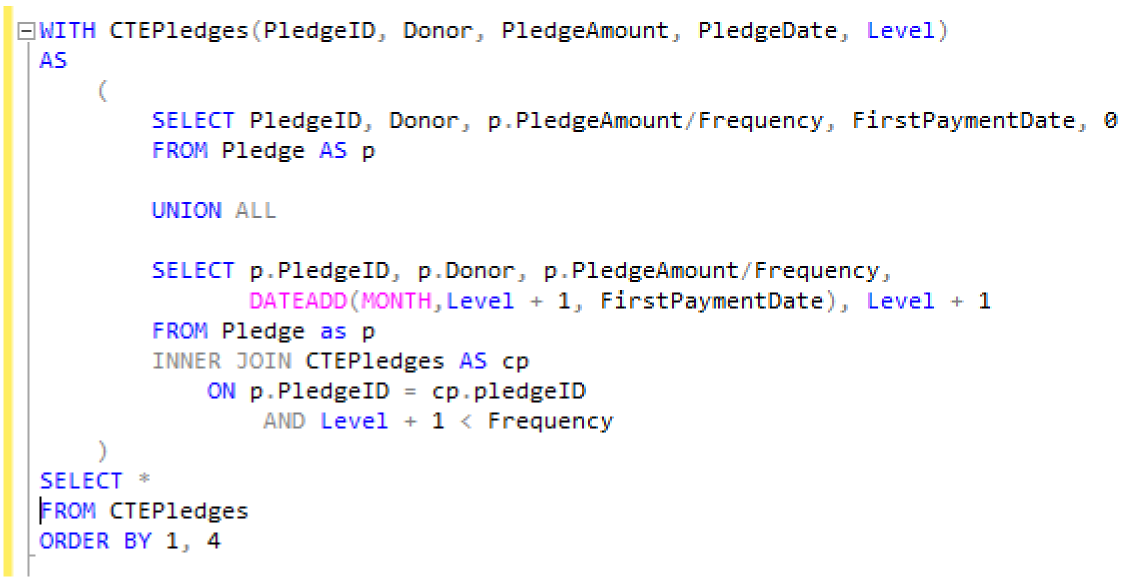
You will find a script for this whole solution here.

So, this is how I would like to see the data that feeds the SSRS report dataset:

PledgeID	Donor	Monthly Payment	Payment Date	Payment Index
1	Fred	100.00	2010-01-01 00:00:00.000	0
1	Fred	100.00	2010-02-01 00:00:00.000	1
1	Fred	100.00	2010-03-01 00:00:00.000	2
1	Fred	100.00	2010-04-01 00:00:00.000	3
1	Fred	100.00	2010-05-01 00:00:00.000	4
1	Fred	100.00	2010-06-01 00:00:00.000	5
2	Mary	200.00	2010-05-05 00:00:00.000	0
2	Mary	200.00	2010-06-05 00:00:00.000	1
2	Mary	200.00	2010-07-05 00:00:00.000	2
2	Mary	200.00	2010-08-05 00:00:00.000	3
2	Mary	200.00	2010-09-05 00:00:00.000	4
2	Mary	200.00	2010-10-05 00:00:00.000	5
2	Mary	200.00	2010-11-05 00:00:00.000	6
2	Mary	200.00	2010-12-05 00:00:00.000	7
2	Mary	200.00	2011-01-05 00:00:00.000	8
2	Mary	200.00	2011-02-05 00:00:00.000	9
2	Mary	200.00	2011-03-05 00:00:00.000	10
2	Mary	200.00	2011-04-05 00:00:00.000	11

CTEs are perfect for any kind of recursive querying.

Here is the query I used:



```
WITH CTEPledges(PledgeID, Donor, PledgeAmount, PledgeDate, Level)
AS
    (
        SELECT PledgeID, Donor, p.PledgeAmount/Frequency, FirstPaymentDate, 0
        FROM Pledge AS p

        UNION ALL

        SELECT p.PledgeID, p.Donor, p.PledgeAmount/Frequency,
               DATEADD(MONTH,Level + 1, FirstPaymentDate), Level + 1
        FROM Pledge as p
        INNER JOIN CTEPledges AS cp
            ON p.PledgeID = cp.pledgeID
                AND Level + 1 < Frequency
    )
SELECT *
FROM CTEPledges
ORDER BY 1, 4
```

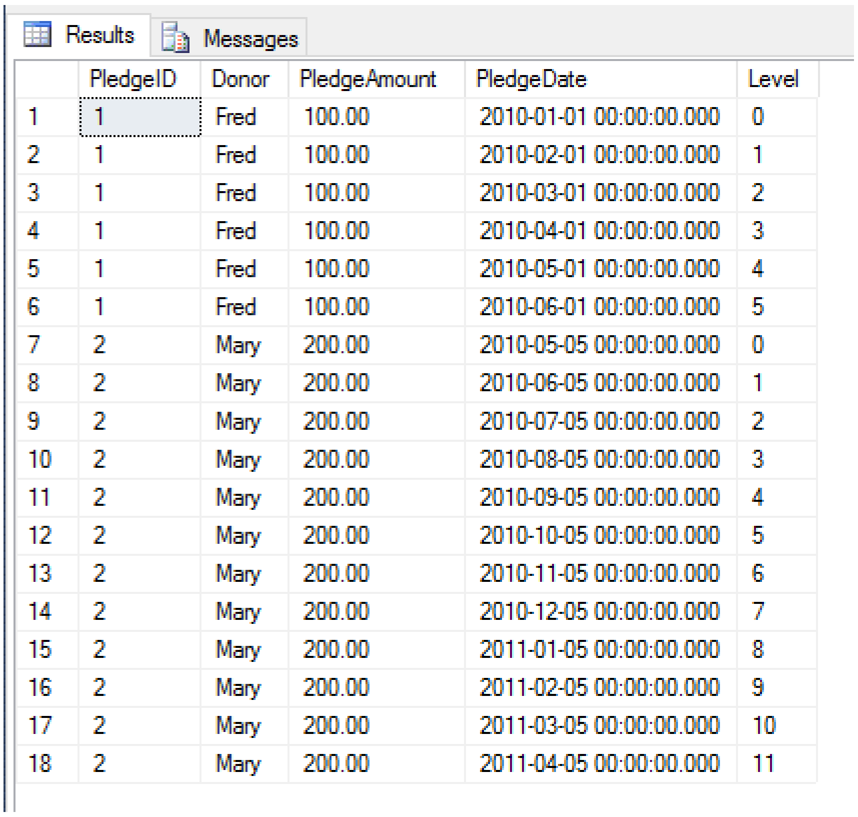
The first part of the CTE query (before the UNION ALL) loads the parent records (the two pledge records) along with an initial payment amount and date.

The second part of the CTE query (after the UNION ALL) uses the parent records that have been loaded into the CTE to add an increment to the month of the start date to generate the series of donation payment dates.

The increment to add to the month is provided through the derived column called "**Level**".

- In the first part of the CTE this is set to the literal value of 0.
- In the second part of the CTE this is set to the parent record's Level value plus 1.

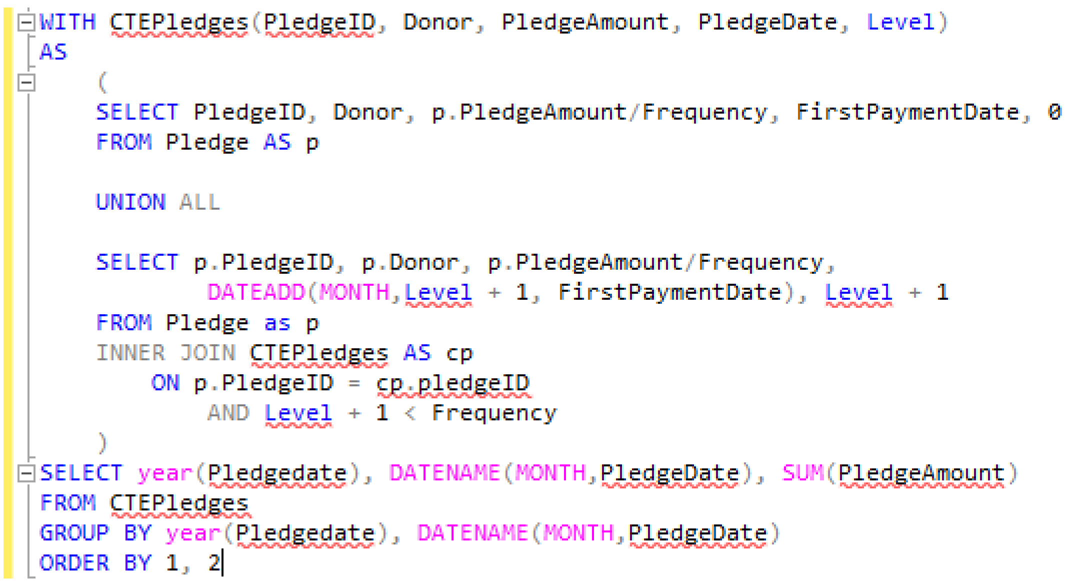
Here is the resulting data set:



	PledgeID	Donor	PledgeAmount	PledgeDate	Level
1	1	Fred	100.00	2010-01-01 00:00:00.000	0
2	1	Fred	100.00	2010-02-01 00:00:00.000	1
3	1	Fred	100.00	2010-03-01 00:00:00.000	2
4	1	Fred	100.00	2010-04-01 00:00:00.000	3
5	1	Fred	100.00	2010-05-01 00:00:00.000	4
6	1	Fred	100.00	2010-06-01 00:00:00.000	5
7	2	Mary	200.00	2010-05-05 00:00:00.000	0
8	2	Mary	200.00	2010-06-05 00:00:00.000	1
9	2	Mary	200.00	2010-07-05 00:00:00.000	2
10	2	Mary	200.00	2010-08-05 00:00:00.000	3
11	2	Mary	200.00	2010-09-05 00:00:00.000	4
12	2	Mary	200.00	2010-10-05 00:00:00.000	5
13	2	Mary	200.00	2010-11-05 00:00:00.000	6
14	2	Mary	200.00	2010-12-05 00:00:00.000	7
15	2	Mary	200.00	2011-01-05 00:00:00.000	8
16	2	Mary	200.00	2011-02-05 00:00:00.000	9
17	2	Mary	200.00	2011-03-05 00:00:00.000	10
18	2	Mary	200.00	2011-04-05 00:00:00.000	11

This query can now be used behind a dataset in an SSRS report to generate the monthly summary report.

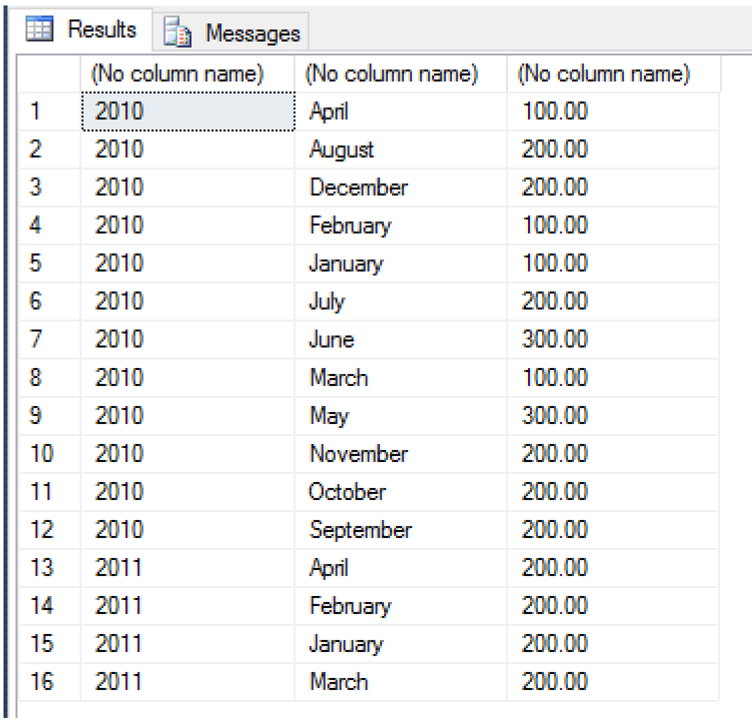
Alternatively, you could simply use a GROUP BY clause on the final SELECT of the CTE to generate a quick monthly summary as shown below:



```
WITH CTEPledges(PledgeID, Donor, PledgeAmount, PledgeDate, Level)
AS
    (
    SELECT PledgeID, Donor, p.PledgeAmount/Frequency, FirstPaymentDate, 0
    FROM Pledge AS p

    UNION ALL

    SELECT p.PledgeID, p.Donor, p.PledgeAmount/Frequency,
            DATEADD(MONTH,Level + 1, FirstPaymentDate), Level + 1
    FROM Pledge as p
    INNER JOIN CTEPledges AS cp
        ON p.PledgeID = cp.pledgeID
            AND Level + 1 < Frequency
    )
SELECT year(Pledgedate), DATENAME(MONTH,PledgeDate), SUM(PledgeAmount)
FROM CTEPledges
GROUP BY year(Pledgedate), DATENAME(MONTH,PledgeDate)
ORDER BY 1, 2
```



	(No column name)	(No column name)	(No column name)
1	2010	April	100.00
2	2010	August	200.00
3	2010	December	200.00
4	2010	February	100.00
5	2010	January	100.00
6	2010	July	200.00
7	2010	June	300.00
8	2010	March	100.00
9	2010	May	300.00
10	2010	November	200.00
11	2010	October	200.00
12	2010	September	200.00
13	2011	April	200.00
14	2011	February	200.00
15	2011	January	200.00
16	2011	March	200.00

If you are itching to learn more why not book on to our SQL Server Database Querying training courses? *This link will take you to the course outlines:*

http://ptr.co.uk/databases-business-intelligence-courses.