Tarea4 - Fundamentos y Diseño de Bases de Datos

Alegria Sallo Daniel Rodrigo (215270)

15 de junio de 2024

El codigo fuente de los ejercicios puede encontrarse en el repositorio https

1. Investigacion

Investigar las funciones mas importantes de SQL SERVER, tales como:

1. **SubString:** La función SUBSTRING se utiliza para extraer una parte de una cadena de caracteres.
   Syntaxis: `SUBSTRING ( expression, start, length )`
   - **expression**: La cadena de caracteres de la que se desea extraer una parte.
   - **start**: La posición inicial (basada en 1).
   - **length**: La longitud de la parte que se desea extraer.

   Ejemplo: `SELECT SUBSTRING('Hello World', 7, 5); -- Devuelve 'World'`

2. **DatePart:** La función DATEPART devuelve una parte especifica de una fecha
   Syntaxis: `DATEPART ( part, date )`
   - **part**: La parte de la fecha que se desea extraer (por ejemplo, year, month, day, hour, etc).
   - **date**: La fecha de la que se desea extraer la parte.

   Ejemplo: `DATEPART(year, "2003/08/27"); -- Devuelve 2024`

3. **DateDiff:** La función DATEDIFF devuelve la diferencia entre dos fechas en terminos de una parte especifica
   Syntaxis: `DATEDIFF( part, startdate, enddate )`
   - **part**: La parte de la fecha para la diferencia (por ejemplo, day, month, year, etc).
   - **startdate**: La fecha de inicio.
   - **enddate**: La fecha final.

   Ejemplo: `DATEDIFF(day, "2003/08/27", 2003/08/3"); -- Devuelve 25`

4. **IsNull:** La función ISNULL reemplaza los valores NULL con un valor especificado
   Syntaxis: `ISNULL( expression, replacement )`
   - **expression**: La expresion que puede ser NULL.
   - **replacement**: El valor que se utilizara en lugar de NULL.

   Ejemplo: `ISNULL(NULL, "No hay valor"); -- Devuelve "No hay valor"`

5. **Cast:** La función CAST convierte una expresion de un tipo de datos a otro
   Syntaxis: `CAST( expression as data_type )`
   - **expression**: La expresion que se va a convertir.

- **data_type**: El tipo de datos al que se va convertir la expresion.

Ejemplo: `CAST(123 as VARCHAR(10)); -- Devuelve "123"`

6. **GetDate:** La función GETDATE devuelve la fecha y hora actuales del sistema

   Syntaxis: `GETDATE()` Ejemplo: `GETDATE(); -- Devuelve la fecha y hora actual`

7. **Str:** La función STR convierte un numero en una cadena de caracteres.

   Syntaxis: `STR( number, length, decimal )`

   - **number**: El numero que se va a convertir.
   - **length**: La longitud de la caden resultante, incluyendo el punto decimal y los decimales.
   - **decimal**: El numero de lugares decimales.

   Ejemplo: `STR(123.45, 6, VARCHAR(10)); -- Devuelve "123"`

2. Ejercicios

Antes de desarrollar los ejercicios, decidi hacer un script que cree procedimientos que completen la informacion de una tabla. El script no funciona, pero lo intente. Cada vez que haga referencia a este procedimiento en los ejercicios, simmplemente estoy tratando de completar la informacion.

```
use DBCreditoRural;
go
set dateformat dMy
go

--[[[ LIMPIAR ]]]
DROP PROCEDURE IF EXISTS cr_CompletarComunidades;
DROP PROCEDURE IF EXISTS cr_CompletarPrestatarios;
DROP PROCEDURE IF EXISTS cr_CompletarPrestamos;
DROP PROCEDURE IF EXISTS cr_CompletarOficiales;
DROP TYPE IF EXISTS TTablaComunidades;
DROP TYPE IF EXISTS TTablaPrestatarios;
DROP TYPE IF EXISTS TTablaPrestamos;
DROP TYPE IF EXISTS TTablaOficiales;
go

--[[[ TYPOS ]]]
CREATE TYPE TTablaComunidades AS TABLE (CodComunidad TCodComunidad)
CREATE TYPE TTablaPrestatarios AS TABLE (CodPrestatario TCodPrestatario)
CREATE TYPE TTablaPrestamos AS TABLE (DocPrestamo TDocPrestamo)
CREATE TYPE TTablaOficiales AS TABLE (CodOficial TCodOficial)
go

--[[[ PORCEDIMIENTOS ]]]
CREATE PROCEDURE cr_CompletarComunidades @TablaComunidades TTablaComunidades READONLY
AS
BEGIN
    select T.*, C.Nombre as NombreComunidad
    from Comunidad C inner join @TablaComunidades T
    on P.CodComunidad = T.CodComunidad
END;

CREATE PROCEDURE cr_CompletarPrestatarios @TablaPrestatarios TTablaPrestatarios READONLY
AS
```

```
BEGIN
    select T.*, P.Nombres as NombresPrestatario, P.DocIdentidad as DocIdentidadPrestatario,
↪   P.CodComunidad
    from Prestatario P inner join @TablaPrestatarios T
    on P.CodPrestatario = T.CodPrestatario
END;

CREATE PROCEDURE cr_CompletarPrestamos @TablaPrestamos TTablaPrestamos READONLY
AS
BEGIN
    select T.*, P.CodPrestatario, P.Importe as ImportePrestamo, P.FechaVencimiento,
↪   P.CodComunidad
    from Prestamo P inner join @TablaPrestamos T
    on P.DocPrestamo = T.DocPrestamo
END;

CREATE PROCEDURE cr_CompletarOficiales @TablaOficiales TTablaOficiales READONLY
AS
BEGIN
    select T.*, O.Nombres as NombreOficial, O.Email as EmailOficial
    from Oficial O inner join @TablaOficiales T
    on O.CodOficial = T.CodOficial
END;

-- exec proc_Procedimiento table
```

Escribir las sentencias SELECT para obtener la siguiente informacion

1. Relación de préstamos cancelados de un determinado prestatario.

```
WITH R(CodPrestatario, DocPrestamo) as (
    SELECT P.CodPrestatario, P.DocPrestamo
    FROM Prestamo P left outer join Amortizacion A on P.DocPrestamo = A.DocPrestamo
    GROUP BY P.CodPrestatario, P.DocPrestamo
    HAVING SUM(P.Importe) = SUM(IsNull(A.Importe, 0))
)
exec cr_CompletarPrestamos R;
go
```

2. Relación de préstamos efectuados por los prestatarios de una determinada comunidad.

```
WITH
R (CodComunidad, CodPrestatario) as (
    SELECT Pio.CodComunidad, Pio.CodPrestatario
    FROM Prestatario Pio inner join Prestamo Pmo on Pio.CodPrestatario = Pmo.CodPrestatario
)
exec cr_CompletarPrestamos R;
go
```

3. Relación de prestatarios que hasta la fecha hayan efectuado más de 5 préstamos.

```
WITH
R (CodPrestatario) as (
    SELECT Pio.CodPrestatario
```

```
    FROM Prestatario Pio inner join Prestamo Pmo on Pio.CodPrestatario = Pmo.CodPrestatario
    GROUP BY Pio.CodPrestatario
    HAVING COUNT(Pmo.DocPrestamo) > 5
)
exec cr_CompletarPrestatarios R;
go
```

4. Relación de prestatarios morosos, es decir, aquellos que aún no han cancelado alguna de sus deudas y ya pasó la fecha de vencimiento.

```
WITH
R (CodPrestatario) as (
    SELECT P.CodPrestatario
    FROM Prestamo P left outer join Amortizacion A on P.DocPrestamo = A.DocPrestamo
    GROUP BY P.DocPrestamo, P.CodPrestatario, P.FechaVencimiento, P.Importe
    HAVING SUM(IsNull(A.Importe, 0)) < P.Importe and GetDate() > P.FechaVencimiento
)
exec cr_CompletarPrestatarios R;
go
```

5. Relación de las 5 comunidades que tienen el mayor número de prestatarios.

```
WITH
R (CodComunidad, NroPrestatarios) as (
    SELECT TOP(5) C.CodComunidad, COUNT(P.CodPrestatario) as NroPrestatarios
    FROM Comunidad C inner join Prestatario P on C.CodComunidad = P.CodComunidad
    GROUP BY C.CodComunidad
    ORDER BY NroPrestatarios DESC
)
exec cr_CompletarComunidades R
go
```

6. Relación de comunidades cuyos prestatarios que aún tienen saldos, no hayan efectuado ninguna amortización en lo que va del año 2004.

```
WITH
R1 (CodPrestatario) as (
    SELECT P.CodPrestatario, P.DocPrestamo
    FROM Prestamo P left outer join Amortizacion A on P.DocPrestamo = A.DocPrestamo
    WHERE DATEPART(year, A.FechaCancelacion) = 2004
    GROUP BY P.DocPrestamo
    HAVING (SUM(P.Importe) > SUM(IsNull(A.Importe, 0)))
),
R2 (CodComunidad) as (
    SELECT P.CodComunidad
    FROM Prestatario P left outer join R1 on P.CodPrestatario = R1.DocPrestamo
)
exec cr_CompletarComunidades R2;
go
```

7. Relación de comunidades que no tengan prestatarios morosos

```
WITH
R1 (CodPrestatario) as (
```

```
    SELECT P.CodPrestatario
    FROM Prestamo P left outer join Amortizacion A on P.DocPrestamo = A.DocPrestamo
    GROUP BY P.DocPrestamo, P.CodPrestatario, P.FechaVencimiento
    HAVING SUM(IsNull(A.Importe, 0)) < SUM(P.Importe) and GetDate() > P.FechaVencimiento
),
R2 (CodComunidad) as (
    SELECT P.CodComunidad
    FROM Prestatario P, R1
    WHERE P.CodPrestatario != R1.CodPrestatario
)
exec cr_CompletarComunidades R2;
go
```

8. Relación de comunidades que tengan prestatarios morosos

```
WITH
R1 (CodPrestatario) as (
    SELECT P.CodPrestatario
    FROM Prestamo P left outer join Amortizacion A on P.DocPrestamo = A.DocPrestamo
    GROUP BY P.DocPrestamo, P.CodPrestatario, P.FechaVencimiento
    HAVING SUM(IsNull(A.Importe, 0)) < SUM(P.Importe) and GetDate() > P.FechaVencimiento

),
R2 (CodComunidad) as (
    SELECT P.CodComunidad
    FROM Prestatario P, R1
    WHERE P.CodPrestatario = R1.CodPrestatario
)
exec cr_CompletarComunidades R2;
go
```

9. Relación de comunidades con 3 de sus prestatarios más importantes (los prestatarios más importantes son los que han obtenido mayor número de préstamos).

```
WITH
R1 (CodComunidad, CodPrestatario, NroPrestamos) as (
    -- Contar el Nro de prestamos por prestatario
    SELECT Pio.CodComunidad, Pio.CodPrestatario, COUNT(Pmo.DocPrestamo) as NroPrestamos
    FROM Prestatario Pio left outer join Prestamo Pmo on Pio.CodPrestatario =
↪   Pmo.CodPrestatario
    GROUP BY Pio.CodPrestatario
    ORDER BY NroPrestamos DESC
),
R2 (CodComunidad, NombreComunidad, CodPrestatario) as (
    -- Seleccionar los los top 3 prestatarios por comunidad
    SELECT CodComunidad, C.Nombre as NombreComunidad, P.CodPrestatario
    FROM Prestatario P inner join Comunidad C on P.CodComunidad = C.CodComunidad
    WHERE CodPrestatario IN (
        select top(3) CodPrestatario from R1
        where R1.CodComunidad = P.CodComunidad
        order by NroPrestamos desc
    )
    GROUP BY CodComunidad DESC
)
exec cr_CompletarPrestatarios R2;
go
```

10. Relación de prestatarios que en ninguno de sus préstamos hayan incurrido en mora.

```
WITH
R1 as (
    SELECT P.CodPrestatario
    FROM Prestamo P left outer join Amortizacion A on P.DocPrestamo = A.DocPrestamo
    GROUP BY P.DocPrestamo, P.CodPrestatario, P.FechaVencimiento
    HAVING SUM(IsNull(A.Importe, 0)) < SUM(P.Importe) and GetDate() > P.FechaVencimiento
),
R2 as (
    SELECT P.CodPrestatario
    FROM Prestatario P, R1
    WHERE P.CodPrestatario = R1.CodPrestatario

)
exec cr_CompletarPrestatarios R2;
go
```

11. Relación de prestatarios que en todas las veces que solicitó un préstamo, sólo una vez incurrió en mora.

```
WITH
R1 as (
    SELECT P.CodPrestatario, P.DocPrestamo
    FROM Prestamo P left outer join Amortizacion A on P.DocPrestamo = A.DocPrestamo
    GROUP BY P.DocPrestamo, P.CodPrestatario, P.FechaVencimiento
    HAVING SUM(IsNull(A.Importe, 0)) < SUM(P.Importe) and GetDate() > P.FechaVencimiento
),
R2 as (
    SELECT CodPrestatario
    FROM R1
    GROUP BY DocPrestamo
    HAVING COUNT(DocPrestamo) = 1
)
exec cr_CompletarPrestatarios R2;
go
```

12. Relación de prestatarios que hayan cancelado sus préstamos sin pagos parciales.

```
WITH
R as (
    SELECT P.CodPrestatario
    FROM Prestamo P left outer join Amortizacion A on P.DocPrestamo = A.DocPrestamo
    GROUP BY P.DocPrestamo
    HAVING SUM(P.Importe) = SUM(IsNull(A.Importe, 0)) and
           COUNT(A.DocCancelacion) = 1
)
exec cr_CompletarPrestatarios R;
go
```

13. Relación de los oficiales de crédito estrella de cada mes del año 2003. (Se considera oficial de crédito “estrella” del mes, al oficial de crédito que haya colocado el mayor número de préstamos en el mes)

```
WITH
MEJORES as (
    SELECT DATEPART(month, P.FechaPrestamo) as Mes, O.CodOficial
```

```
    FROM Oficial O inner join Prestamo P on O.CodOficial = P.CodOficial
    WHERE DATEPART(year, P.FechaPrestamo) = 2003
),
R as (
    SELECT M.Mes, M.CodOficial
    FROM MEJORES M
    WHERE M.CodOficial in (
        select top(1) m.CodOficial
        from MEJORES m
        where m.Mes = M.Mes and m.CodOficial = M.CodOficial
        order by m.CodOficial desc
    )
    GROUP BY DATEPART(month, P.FechaPrestamo), O.CodOficial
    HAVING DATEPART(year, P.FechaPrestamo) = 2003
)

exec cr_CompletarOficiales R;
go
```

14. Relación de oficiales de crédito que no hayan colocado por lo menos un préstamo en algún mes del año 2003.

```
WITH
R as (
    SELECT O.CodOficial
    FROM Oficial O left outer join Prestamo P on O.CodOficial = P.CodOficial
    WHERE P.FechaPrestamo BETWEEN "01/01/2003" AND "12/31/2003"
    GROUP BY O.CodOficial
    HAVING COUNT(P.DocPrestamo) < 1
)
exec cr_CompletarOficiales R;
go
```

15. Relación de préstamos en riesgo. Se considera un préstamo en riesgo cuando tiene saldo y ha trascurrido más de 6 meses de su fecha de vencimiento.

```
WITH
R as (
    SELECT P.DocPrestamo
    FROM Prestamo P left outer join Amortizacion A on P.DocPrestamo = A.DocPrestamo
    GROUP BY P.DocPrestamo, P.FechaVencimiento
    HAVING ( SUM(P.Importe) - SUM(IsNull(A.Importe, 0)) ) > 0 and
             DateDiff(month, GetDate(), P.FechaVencimiento ) > -6
)
exec cr_CompletarPrestamos R;
go
```

16. Relación de comunidades con los saldos totales de los préstamos que están en riesgo.

```
WITH
R1 (CodPrestatario, DocPrestamo, SaldoEnRiesgo) as (
    -- Prestatarios en riesgo
    SELECT P.CodPrestatario, P.DocPrestamo, (SUM(P.Importe) - SUM(IsNull(A.Importe, 0))) as
↪   SaldoEnRiesgo
    FROM Prestamo P left outer join Amortizacion A on P.DocPrestamo = A.DocPrestamo
    GROUP BY P.DocPrestamo
```

```
    HAVING ( SUM(P.Importe) - SUM(IsNull(A.Importe, 0)) ) > 0 and
             DateDiff(month, GetDate(), P.FechaVencimiento ) > -6
),
R2 (CodComunidad, SaldoTotalEnRiesgo) as (
    -- Comunidades con Saldos Totales
    SELECT C.CodComunidad, SUM(IsNull(SaldoEnRiesgo,0)) as SaldoTotalEnRiesgo
    FROM Comunidad C inner join R1 on C.CodPrestatario = R1.CodPrestatario
    GROUP BY C.CodComunidad
)
exec cr_CompletarComunidades R2;
go
```

17. Relación de oficiales de crédito con los saldos totales de los préstamos efectuados por ellos que están en riesgo.

```
WITH
R1 as (
    SELECT P.CodOficial, P.DocPrestamo, (SUM(P.Importe) - SUM(IsNull(A.Importe, 0))) as
↪   SaldoEnRiesgo
    FROM Prestamo P left outer join Amortizacion A on P.DocPrestamo = A.DocPrestamo
    GROUP BY P.DocPrestamo
    HAVING ( SUM(P.Importe) - SUM(IsNull(A.Importe, 0)) ) > 0 and
             DateDiff(month, GetDate(), P.FechaVencimiento ) > -6
),
R2 as (
    SELECT O.CodOficial, SUM(IsNull(SaldoEnRiesgo,0)) as SaldoTotalEnRiesgo
    FROM Oficial O inner join R1 on O.CodOficial = R1.CodOficial
    GROUP BY C.CodOficial
)
exec cr_CompletarOficiales R2;
go
```

18. Relación de oficiales de crédito que hayan efectuado préstamos en todas las comunidades.

```
WITH
R as (
    SELECT Pmo.CodOficial, COUNT(Pio.CodComunidad) as NroComunidades
    FROM Prestamo Pmo inner join Prestatario Pio on Pmo.CodPrestatario = Pio.CodPrestatario
    GROUP BY Pmo.CodOficial
    HAVING COUNT(Pio.CodComunidad) = (
        select COUNT(CodComunidad) from Comunidad
    )
)
exec cr_CompletarOficiales R;
go
```

19. Relación de comunidades cuyos montos totales de préstamo hayan disminuido en los dos últimos años, es decir, el monto total del 2003 sea menor al del 2002 y el monto total del 2002 sea menor al del 2001.

```
WITH
P1 as (
    SELECT P.CodPrestatario, P.DocPrestamo, SUM(P.Importe) as MontoPrestamo
    FROM Prestamo P
    GROUP BY P.DocPrestamo, P.FechaPrestamo
    HAVING P.FechaPrestamo BETWEEN "01/01/2001" AND "31/12/2001"
),
```

```
P2 as (
    SELECT P.CodPrestatario, P.DocPrestamo, SUM(P.Importe) as MontoPrestamo
    FROM Prestamo P
    GROUP BY P.DocPrestamo, P.FechaPrestamo
    HAVING P.FechaPrestamo BETWEEN "01/01/2002" AND "31/12/2002"
),
P3 as (
    SELECT P.CodPrestatario, P.DocPrestamo, SUM(P.Importe) as MontoPrestamo
    FROM Prestamo P
    GROUP BY P.DocPrestamo, P.FechaPrestamo
    HAVING P.FechaPrestamo BETWEEN "01/01/2003" AND "31/12/2003"
),

Q1 as (
    SELECT C.CodComunidad, SUM(IsNull(MontoPrestamo,0)) as MontoTotalPrestamo
    FROM Comunidad C inner join T1 on C.CodPrestatario = T1.CodPrestatario
    GROUP BY C.CodComunidad
),
Q2 as (
    SELECT C.CodComunidad, SUM(IsNull(MontoPrestamo,0)) as MontoTotalPrestamo
    FROM Comunidad C inner join T2 on C.CodPrestatario = T2.CodPrestatario
    GROUP BY C.CodComunidad
),
Q3 as (
    SELECT C.CodComunidad, SUM(IsNull(MontoPrestamo,0)) as MontoTotalPrestamo
    FROM Comunidad C inner join T3 on C.CodPrestatario = T3.CodPrestatario
    GROUP BY C.CodComunidad
),

R1 as (
    SELECT C.CodComunidad, Q3.MontoTotalPrestamo
    FROM Q3 inner join Q2 on Q3.CodComunidad = Q2.CodComunidad
    WHERE Q3.MontoTotalPrestamo < Q2.MontoTotalPrestamo
),
R2 as (
    SELECT C.CodComunidad, Q2.MontoTotalPrestamo
    FROM Q2 inner join Q1 on Q2.CodComunidad = Q1.CodComunidad
    WHERE Q2.MontoTotalPrestamo < Q1.MontoTotalPrestamo
),
R3 as (
    SELECT C.CodComunidad
    FROM R1 inner join R2 on R1.CodComunidad = R2.CodComunidad
    WHERE R1.MontoTotalPrestamo < R2.MontoTotalPrestamo
)

exec cr_CompletarComunidades R3;
go
```

20. Calcular el índice de morosidad por comunidad. El índice de morosidad es igual al porcentaje del saldo del préstamo que esta en riesgo sobre el total del préstamo.

```
WITH
Q1 as (
    SELECT P.CodPrestatario, (SUM(P.Importe) - SUM(IsNull(A.Importe, 0))) as SaldoMoroso
    FROM Prestamo P left outer join Amortizacion A on P.DocPrestamo = A.DocPrestamo
    GROUP BY P.DocPrestamo, P.CodPrestatario, P.FechaVencimiento
    HAVING SUM(IsNull(A.Importe, 0)) < SUM(P.Importe) and GetDate() > P.FechaVencimiento
```

```
),
Q2 as (
    SELECT P.CodPrestatario, (SUM(P.Importe) - SUM(IsNull(A.Importe, 0))) as Saldo
    FROM Prestamo P left outer join Amortizacion A on P.DocPrestamo = A.DocPrestamo
    GROUP BY P.DocPrestamo, P.CodPrestatario, P.FechaVencimiento
),
Q3 as (
    SELECT Q1.CodPrestatario, SaldoMoroso, Saldo
    FROM Q1 left outer join Q2 on Q1.CodPrestatario = Q2.CodPrestatario
),

R as (
    SELECT P.CodComunidad, SUM(SaldoMoroso)/SUM(Saldo) as IndiceMorosidad
    FROM Prestatario P left outer join Q3 on P.CodPrestatario = Q3.CodPrestatario
    GROUP BY P.CodComunidad
)

exec cr_CompletarComunidades R;
go
```

21. Relación de préstamos colocados por comunidad, para los años 2000, 2001, 2002 y 2003, con la siguiente información: R(CodComunidad,NombreComunidad,Total2000,Total2001,Total2002,Total2003)

```
WITH
T as (
    SELECT Pio.CodComunidad, Pmo.CodPrestatario, Pmo.DocPrestamo, Pmo.FechaPrestamo
    FROM Prestamo Pmo left outer join Prestatario Pio on Pmo.CodPrestatario =
↪   Pio.CodPrestatario
),

Q0 as (
    SELECT C.CodComunidad, COUNT(T.DocPrestamo) as Total2000
    FROM Comunidad C left outer join T on C.CodComunidad = T.CodComunidad
    GROUP BY T.CodComunidad
    HAVING DATEPART(year, T.FechaPrestamo) = 2000
),
Q1 as (
    SELECT C.CodComunidad, COUNT(T.DocPrestamo) as Total2001
    FROM Comunidad C left outer join T on C.CodComunidad = T.CodComunidad
    GROUP BY T.CodComunidad
    HAVING DATEPART(year, T.FechaPrestamo) = 2001
),
Q2 as (
    SELECT C.CodComunidad, COUNT(T.DocPrestamo) as Total2002
    FROM Comunidad C left outer join T on C.CodComunidad = T.CodComunidad
    GROUP BY T.CodComunidad
    HAVING DATEPART(year, T.FechaPrestamo) = 2002
),
Q3 as (
    SELECT C.CodComunidad, COUNT(T.DocPrestamo) as Total2003
    FROM Comunidad C left outer join T on C.CodComunidad = T.CodComunidad
    GROUP BY T.CodComunidad
    HAVING DATEPART(year, T.FechaPrestamo) = 2003
),

R as (
    SELECT Q0.CodComunidad, Q0.Total2000, Q1.Total2001, Q2.Total2002, Q3.Total2003
```

```
    FROM Q0, Q1, Q2, Q3
    WHERE Q0.CodComunidad = Q1.CodComunidad = Q2.CodComunidad = Q3.CodComunidad
    ORDER BY Q0.CodComunidad
)

SELECT C.NombreComunidad, R.*
FROM Comunidad C inner join R on C.CodComunidad = R.CodComunidad;
go
```

22. Relación de préstamos colocados por comunidad, para los meses del año 2003, con la siguiente información: R(CodComunidad,NombreComunidad,TotalEnero,TotalFebrero, ..., TotalDiciembre)

```
WITH
T as (
    SELECT Pio.CodComunidad, Pmo.CodPrestatario, Pmo.DocPrestamo, Pmo.FechaPrestamo
    FROM Prestamo Pmo left outer join Prestatario Pio on Pmo.CodPrestatario =
↪   Pio.CodPrestatario
),

Q1 as (
    SELECT C.CodComunidad, COUNT(T.DocPrestamo) as TotalMes1
    FROM Comunidad C left outer join T on C.CodComunidad = T.CodComunidad
    GROUP BY T.CodComunidad
    HAVING DATEPART(year, T.FechaPrestamo) = 2003 and DATEPART(month, T.FechaPrestamo) = 1
),
Q2 as (
    SELECT C.CodComunidad, COUNT(T.DocPrestamo) as TotalMes2
    FROM Comunidad C left outer join T on C.CodComunidad = T.CodComunidad
    GROUP BY T.CodComunidad
    HAVING DATEPART(year, T.FechaPrestamo) = 2003 and DATEPART(month, T.FechaPrestamo) = 2
),
Q3 as (
    SELECT C.CodComunidad, COUNT(T.DocPrestamo) as TotalMes3
    FROM Comunidad C left outer join T on C.CodComunidad = T.CodComunidad
    GROUP BY T.CodComunidad
    HAVING DATEPART(year, T.FechaPrestamo) = 2003 and DATEPART(month, T.FechaPrestamo) = 3
),
Q4 as (
    SELECT C.CodComunidad, COUNT(T.DocPrestamo) as TotalMes4
    FROM Comunidad C left outer join T on C.CodComunidad = T.CodComunidad
    GROUP BY T.CodComunidad
    HAVING DATEPART(year, T.FechaPrestamo) = 2003 and DATEPART(month, T.FechaPrestamo) = 4
),
Q5 as (
    SELECT C.CodComunidad, COUNT(T.DocPrestamo) as TotalMes5
    FROM Comunidad C left outer join T on C.CodComunidad = T.CodComunidad
    GROUP BY T.CodComunidad
    HAVING DATEPART(year, T.FechaPrestamo) = 2003 and DATEPART(month, T.FechaPrestamo) = 5
),
Q6 as (
    SELECT C.CodComunidad, COUNT(T.DocPrestamo) as TotalMes6
    FROM Comunidad C left outer join T on C.CodComunidad = T.CodComunidad
    GROUP BY T.CodComunidad
    HAVING DATEPART(year, T.FechaPrestamo) = 2003 and DATEPART(month, T.FechaPrestamo) = 6
),
Q7 as (
    SELECT C.CodComunidad, COUNT(T.DocPrestamo) as TotalMes7
```

```
    FROM Comunidad C left outer join T on C.CodComunidad = T.CodComunidad
    GROUP BY T.CodComunidad
    HAVING DATEPART(year, T.FechaPrestamo) = 2003 and DATEPART(month, T.FechaPrestamo) = 7
),
Q8 as (
    SELECT C.CodComunidad, COUNT(T.DocPrestamo) as TotalMes8
    FROM Comunidad C left outer join T on C.CodComunidad = T.CodComunidad
    GROUP BY T.CodComunidad
    HAVING DATEPART(year, T.FechaPrestamo) = 2003 and DATEPART(month, T.FechaPrestamo) = 8
),
Q9 as (
    SELECT C.CodComunidad, COUNT(T.DocPrestamo) as TotalMes9
    FROM Comunidad C left outer join T on C.CodComunidad = T.CodComunidad
    GROUP BY T.CodComunidad
    HAVING DATEPART(year, T.FechaPrestamo) = 2003 and DATEPART(month, T.FechaPrestamo) = 9
),
Q10 as (
    SELECT C.CodComunidad, COUNT(T.DocPrestamo) as TotalMes10
    FROM Comunidad C left outer join T on C.CodComunidad = T.CodComunidad
    GROUP BY T.CodComunidad
    HAVING DATEPART(year, T.FechaPrestamo) = 2003 and DATEPART(month, T.FechaPrestamo) = 10
),
Q11 as (
    SELECT C.CodComunidad, COUNT(T.DocPrestamo) as TotalMes11
    FROM Comunidad C left outer join T on C.CodComunidad = T.CodComunidad
    GROUP BY T.CodComunidad
    HAVING DATEPART(year, T.FechaPrestamo) = 2003 and DATEPART(month, T.FechaPrestamo) = 11
),
Q12 as (
    SELECT C.CodComunidad, COUNT(T.DocPrestamo) as TotalMes12
    FROM Comunidad C left outer join T on C.CodComunidad = T.CodComunidad
    GROUP BY T.CodComunidad
    HAVING DATEPART(year, T.FechaPrestamo) = 2003 and DATEPART(month, T.FechaPrestamo) = 12
)

R as (
    SELECT Q0.CodComunidad, Q1.TotalMes1, Q2.TotalMes2, Q3.TotalMes3, Q4.TotalMes4,
                            Q5.TotalMes5, Q6.TotalMes6, Q7.TotalMes7, Q8.TotalMes8,
                            Q9.TotalMes9, Q10.TotalMes10, Q11.TotalMes11, Q12.TotalMes12
    FROM Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, Q10, Q11, Q12
    WHERE Q1.CodComunidad = Q2.CodComunidad = Q3.CodComunidad = Q4.CodComunidad =
          Q5.CodComunidad = Q6.CodComunidad = Q7.CodComunidad = Q8.CodComunidad =
          Q9.CodComunidad = Q10.CodComunidad = Q11.CodComunidad = Q12.CodComunidad
    ORDER BY Q0.CodComunidad
)

SELECT C.NombreComunidad, R.*
FROM Comunidad C inner join R on C.CodComunidad = R.CodComunidad
go
```

– Tal vez la pregunta no debio haber especificado las entradas "TotalX¿ – ALTERNATIVAMENTE:

```
WITH
T as (
    SELECT Pio.CodComunidad, Pmo.CodPrestatario, Pmo.DocPrestamo, Pmo.FechaPrestamo
    FROM Prestamo Pmo left outer join Prestatario Pio on Pmo.CodPrestatario =
↪   Pio.CodPrestatario
```

```
),
R as (
    SELECT C.CodComunidad, DATEPART(month, T.FechaPrestamo) as Mes, COUNT(T.DocPrestamo) as
↪   NroPrestamos
    FROM Comunidad C left outer join T on C.CodComunidad = T.CodComunidad
    GROUP BY T.CodComunidad, DATEPART(month, T.FechaPrestamo)
    HAVING DATEPART(year, T.FechaPrestamo) = 2003
),

SELECT C.NombreComunidad, R.*
FROM Comunidad C inner join R on C.CodComunidad = R.CodComunidad
go
```