

Datawarehouse

Filipe Fidalgo

ffidalgo@ipcb.pt

Sumário

- CTE - Common Table Expressions
 - SQL - Recursivo
- PIVOT (e UNPIVOT)
- GROUP BY – Variantes...
- Exercícios

CTE

- Common Table Expressions (CTE): Trata-se de uma funcionalidade do SQL Server (disponível depois da versão 2005) que permite definir uma tabela derivada temporária. As principais vantagens de CTE são:
 - Permitem implementar consultas recursivas;
 - Podem substituir vistas não requeridas para uso geral, i.e. quando não é necessário guardar a definição dos dados;
 - Permite referenciar a mesma tabela várias vezes na mesma instrução.
- As CTEs podem ser definidas em rotinas pelo utilizador, tais como funções, procedimentos armazenados, triggers, ou views.

CTE

- Uma CTE é composta de:
 - um nome de expressão representando a CTE,
 - uma lista de colunas opcionais,
 - e uma consulta definindo a CTE
- A sintaxe para criação de CTE é a seguinte.

WITH

nome_CTE_1 AS (consulta de definição da CTE 1)

[,nome_CTE_2 AS (consulta de definição da CTE 2)

[...]

Consulta_que_usa_CTEs

CTE

- Vejamos agora uma tabela com os empregados e os seus supervisores:

```
CREATE TABLE MyEmployees
(
EmployeeID smallint NOT NULL,
FirstName varchar(30) NOT NULL,
LastName  varchar(40) NOT NULL,
Title  varchar(50) NOT NULL,
DeptID smallint NOT NULL,
ManagerID int NULL,
CONSTRAINT PK_EmployeeID PRIMARY KEY (EmployeeID)
);
```

CTE

- Insira-se agora alguns registos:

```
INSERT INTO MyEmployees VALUES
(1, 'Ke', 'Sánchez', 'Chief Executive Officer', 16, NULL)
,(273, 'Bria', 'Welcker', 'Vice President of Sales', 3, 1)
,(274, 'Stephe', 'Jiang', 'North American Sales Manager', 3, 273)
,(275, 'Michael', 'Blythe', 'Sales Representative', 3, 274)
,(276, 'Linda', 'Mitchell', 'Sales Representative', 3, 274)
,(285, 'Syed', 'Abbas', 'Pacific Sales Manager', 3, 273)
,(286, 'Lyn', 'Tsoflias', 'Sales Representative', 3, 285)
,(16, 'David', 'Bradley', 'Marketing Manager', 4, 273)
,(23, 'Mary', 'Gibso', 'Marketing Specialist', 4, 16);
```

CTE

- Um exemplo simples de CTE, poderia ser a contagem de subordinados por cada chefe:

100 %

	EmployeeID	ManagerID
1	1	NULL
2	16	273
3	23	16
4	273	1
5	274	273
6	275	274
7	276	274
8	285	273
9	286	285



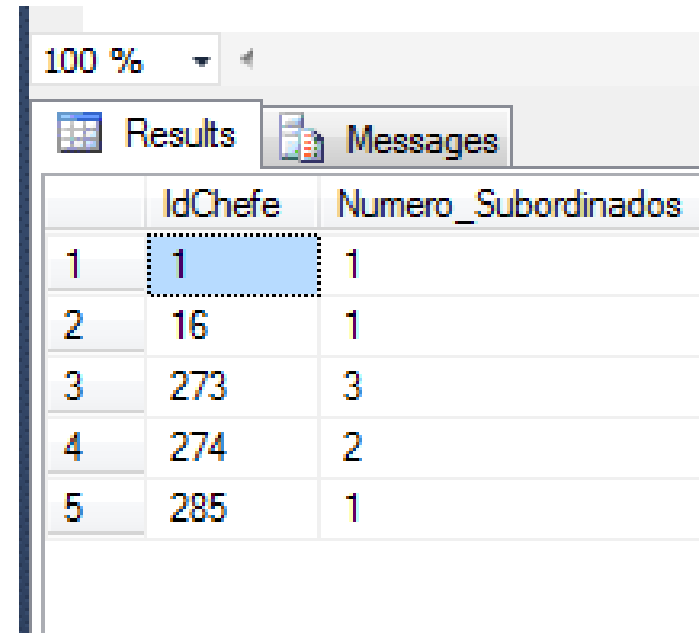
100 %

	ManagerID	Numero_Subordinados
1	1	1
2	16	1
3	273	3
4	274	2
5	285	1

CTE

- Um exemplo simples de CTE, poderia ser a contagem de subordinados por cada chefe:

```
WITH Conta_Subordinados (IdEmpregado, IdChefe)
AS
(
    SELECT EmployeeID, ManagerID
    FROM MyEmployees
    WHERE ManagerID IS NOT NULL
)
SELECT IdChefe, count(IdEmpregado)
FROM Conta_Subordinados
GROUP BY IDChefe
```



100 %

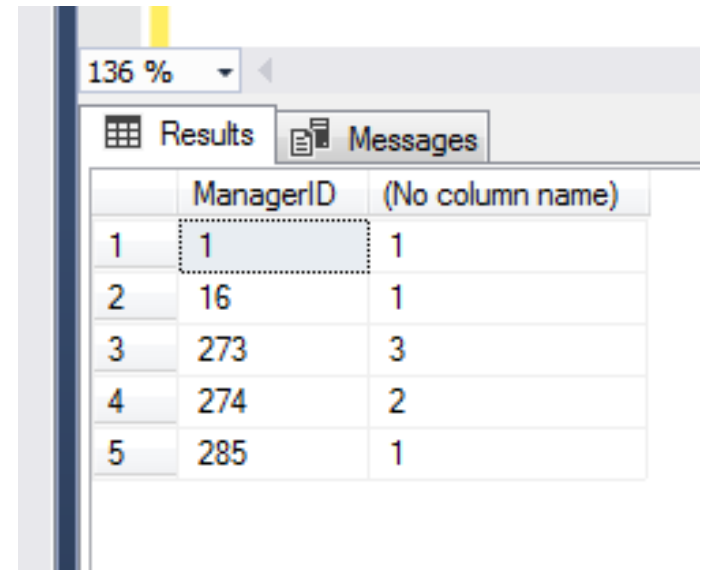
Results Messages

	IdChefe	Numero_Subordinados
1	1	1
2	16	1
3	273	3
4	274	2
5	285	1

CTE

- Caso não fosse usada a lista de colunas:

```
WITH Conta_Subordinados
AS
(
    SELECT EmployeeID, ManagerID
    FROM MyEmployees
    WHERE ManagerID IS NOT NULL
)
SELECT ManagerID, count(EmployeeID)
FROM Conta_Subordinados
GROUP BY ManagerID
```



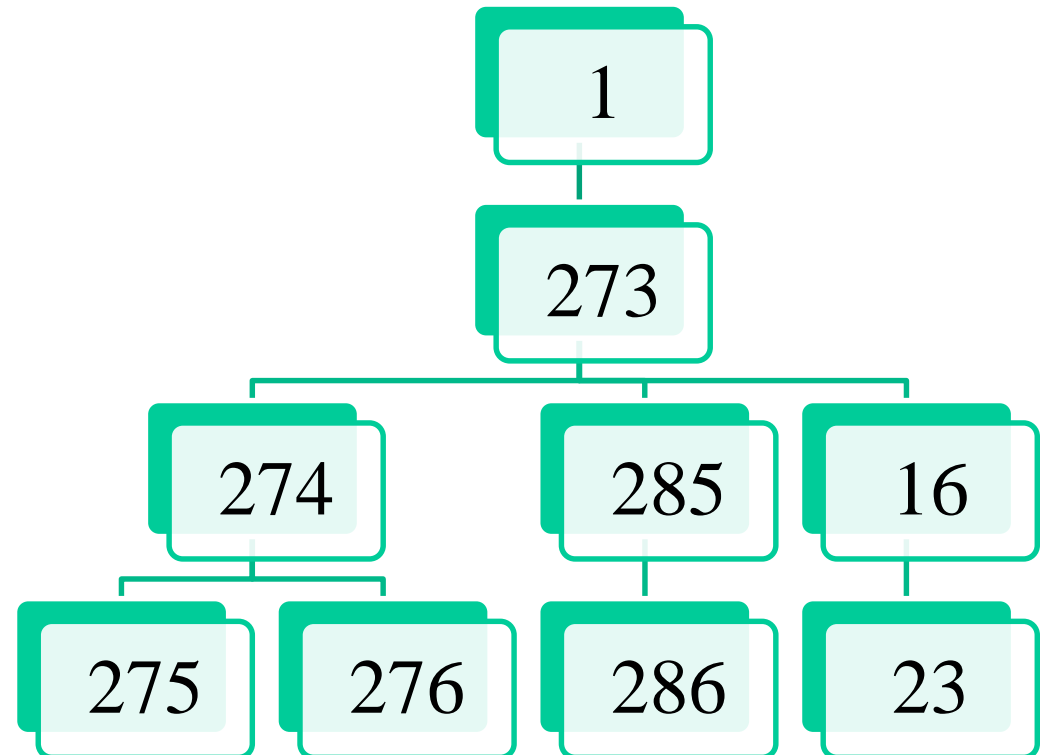
	ManagerID	(No column name)
1	1	1
2	16	1
3	273	3
4	274	2
5	285	1

CTE

- Vejamos agora a hierarquia dos supervisores:

100 %

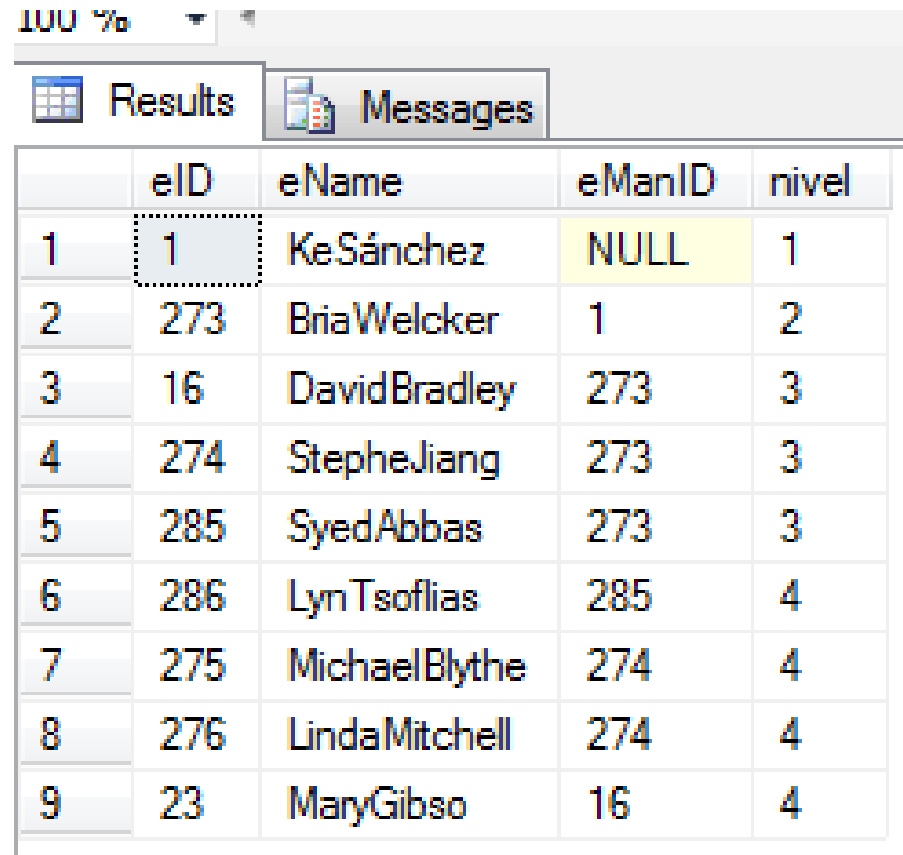
	EmployeeID	ManagerID
1	1	NULL
2	16	273
3	23	16
4	273	1
5	274	273
6	275	274
7	276	274
8	285	273
9	286	285



- Assim, “1” é de nível 1; “273” é de nível 2; “274; 285; 16” são de nível 3; e os restantes de nível 4.

CTE

- Para obtermos os níveis hierárquicos para cada empregado, seria necessária a implementação de consultas recursivas....



100 %

Results Messages

	eID	eName	eManID	nivel
1	1	KeSánchez	NULL	1
2	273	BriaWelcker	1	2
3	16	DavidBradley	273	3
4	274	StepheJiang	273	3
5	285	SyedAbbas	273	3
6	286	LynTsoflias	285	4
7	275	MichaelBlythe	274	4
8	276	LindaMitchell	274	4
9	23	MaryGibso	16	4

CTE

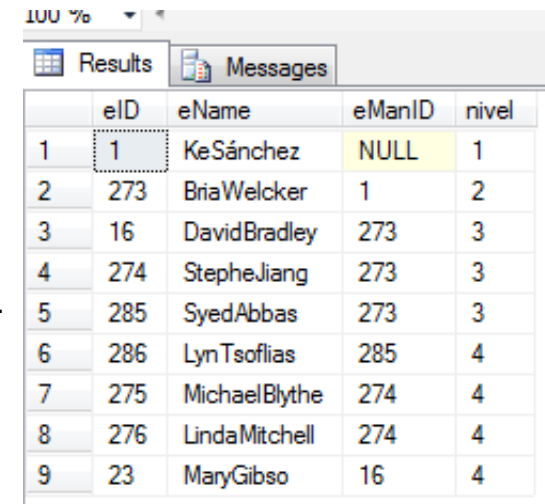
- Para obtermos os níveis hierárquicos para cada empregado, seria necessária a implementação de consultas recursivas....

```
WITH cte (eID, eName, eManID, nivel)
as
(
    SELECT EmployeeID, FirstName + LastName, ManagerID, 1
    FROM MyEmployees
    WHERE ManagerID IS NULL

    UNION ALL

    SELECT EmployeeID, FirstName + LastName, ManagerID, nivel+1
    FROM MyEmployees JOIN cte ON ManagerID=eID
)

select * from cte
ORDER BY Nivel
```



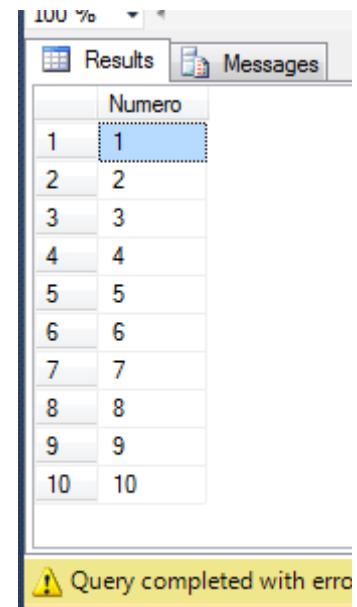
100 %

	eID	eName	eManID	nivel
1	1	Ke Sánchez	NULL	1
2	273	Bria Welcker	1	2
3	16	David Bradley	273	3
4	274	Stephe Jiang	273	3
5	285	Syed Abbas	273	3
6	286	Lyn Tsofilias	285	4
7	275	Michael Blythe	274	4
8	276	Linda Mitchell	274	4
9	23	Mary Gibso	16	4

CTE

- A recursividade pode não ser linear no início....
- Um exemplo que pode ajudar: “Gerar números até um máximo” (CTE + Função: MAXRECURSION)

```
WITH CTE_Numerico AS  
(  
    SELECT 1 AS Numero  
    UNION ALL  
    SELECT Numero + 1 FROM CTE_Numerico  
)  
SELECT * FROM CTE_Numerico  
OPTION (MAXRECURSION 9)
```

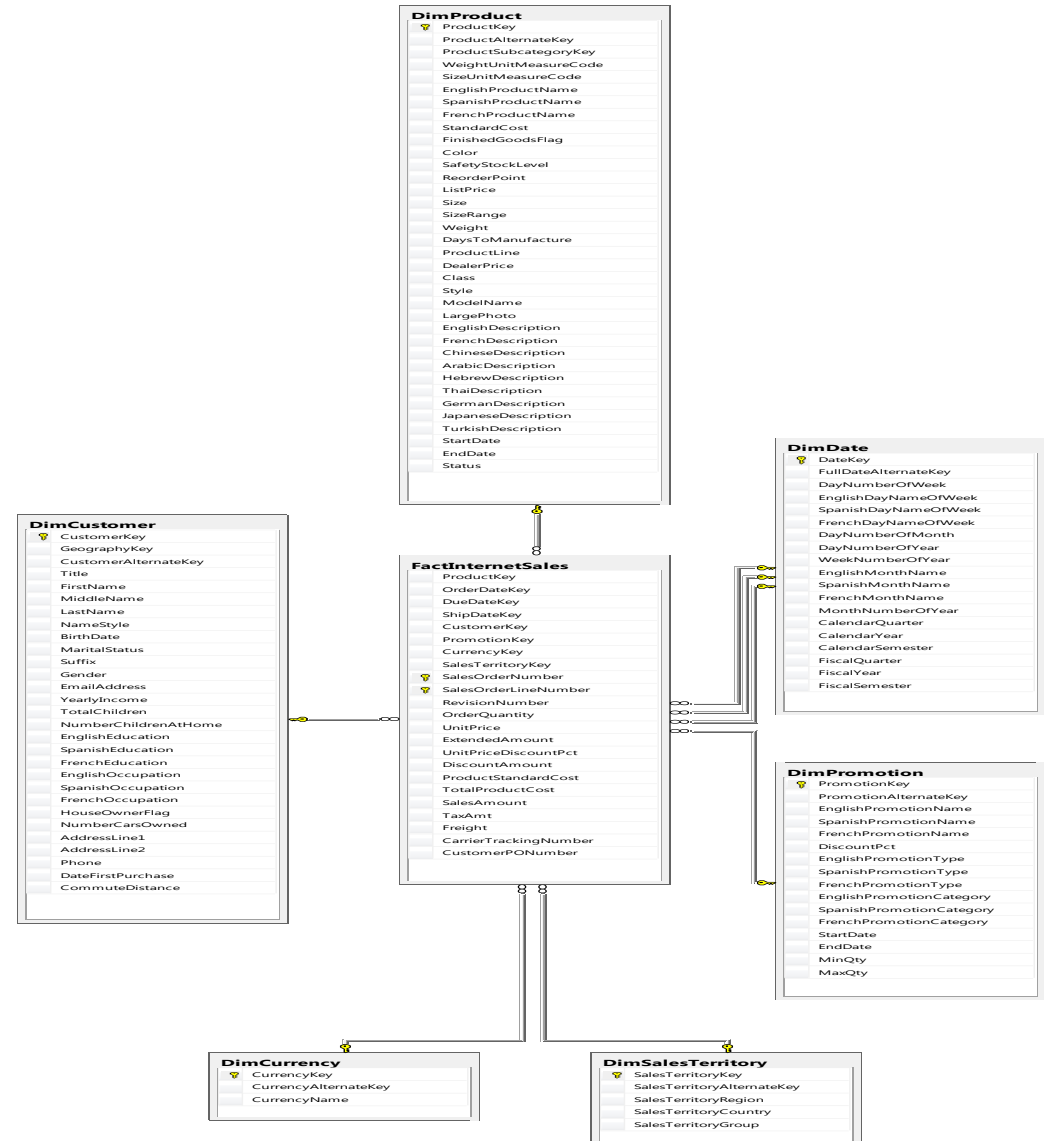


	Numero
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

Query completed with error

mais... SQL

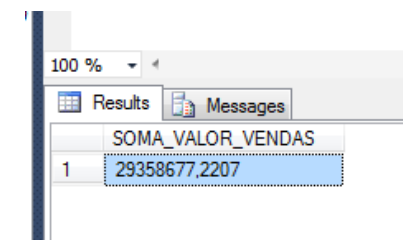
- Consideremos a bd “AdventureWorks DW”, e o DataMart “FactInternetSales”.
(ver pdf)



mais... SQL

- Consideremos a bd “AdventureWorksDW”, e o DataMart “FactInternetSales”.
- Se precisarmos de saber o valor total de vendas, podemos obtê-lo consultando a tabela de factos, somando todos os valores do atributo (medida) “SalesAmount”:

```
SELECT SUM(F.SalesAmount) SOMA_VALOR_VENDAS  
FROM FactInternetSales F
```



The screenshot shows a SQL Server query results window. The window has a title bar with '100 %' and a dropdown arrow. Below the title bar, there are two tabs: 'Results' and 'Messages'. The 'Results' tab is active, showing a table with one column, 'SOMA_VALOR_VENDAS', and one row with the value '29358677.2207'.

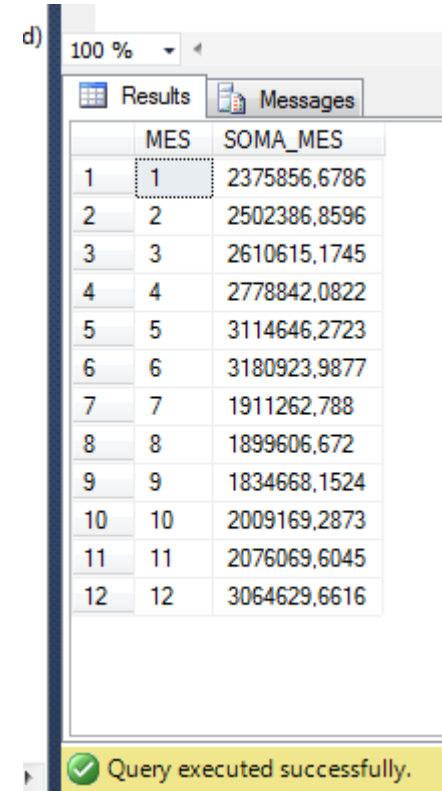
	SOMA_VALOR_VENDAS
1	29358677.2207

mais... SQL

- Embora importante, esse valor torna-se mais relevante se associado a outros elementos/atributos.
- Por exemplo, sabermos quais os valores de vendas em cada mês....

```
SELECT D.MonthNumberOfYear MES,  
       sum(F.SalesAmount) SOMA_MES  
FROM FactInternetSales F JOIN DimDate D  
     ON F.OrderDateKey = D.DateKey  
GROUP BY D.MonthNumberOfYear  
ORDER BY 1
```

d)

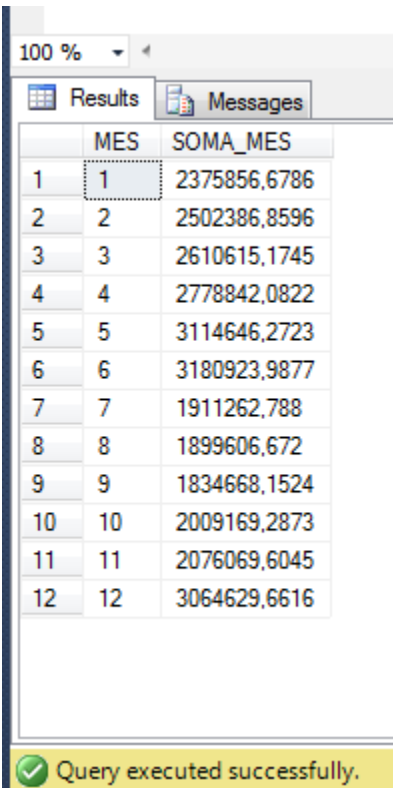


	MES	SOMA_MES
1	1	2375856,6786
2	2	2502386,8596
3	3	2610615,1745
4	4	2778842,0822
5	5	3114646,2723
6	6	3180923,9877
7	7	1911262,788
8	8	1899606,672
9	9	1834668,1524
10	10	2009169,2873
11	11	2076069,6045
12	12	3064629,6616

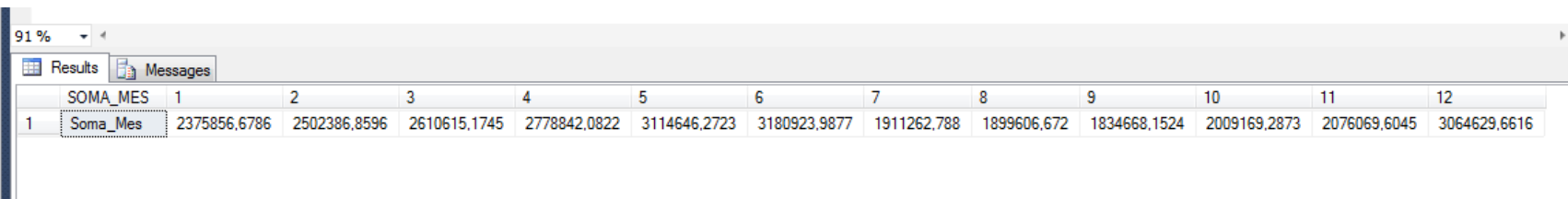
Query executed successfully.

PIVOT

- Imaginemos que é necessária uma situação em que precisa de transformar as linhas do SELECT em COLUNAS
- Aqui entra o operador PIVOT – transforma os dados que ficam em formato horizontal (linhas) e coloca-os em formato vertical (Colunas).



	MES	SOMA_MES
1	1	2375856,6786
2	2	2502386,8596
3	3	2610615,1745
4	4	2778842,0822
5	5	3114646,2723
6	6	3180923,9877
7	7	1911262,788
8	8	1899606,672
9	9	1834668,1524
10	10	2009169,2873
11	11	2076069,6045
12	12	3064629,6616



	SOMA_MES	1	2	3	4	5	6	7	8	9	10	11	12
1	Soma_Mes	2375856,6786	2502386,8596	2610615,1745	2778842,0822	3114646,2723	3180923,9877	1911262,788	1899606,672	1834668,1524	2009169,2873	2076069,6045	3064629,6616

PIVOT

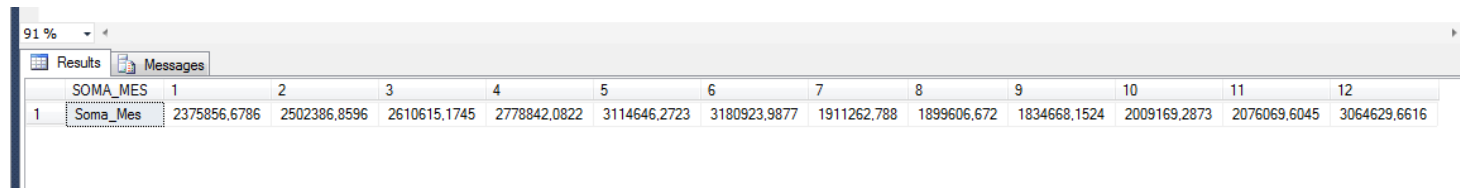
- Vejamos a sintaxe:

```
SELECT <non-pivoted column>,  
    [first pivoted column] AS <column name>,  
    [second pivoted column] AS <column name>,  
    ...  
    [last pivoted column] AS <column name>  
FROM  
    (<SELECT query that produces the data>  
    AS <alias for the source query>  
PIVOT  
(  
    <aggregation function>(<column being aggregated>  
FOR  
[<column that contains the values that will become column headers>]  
    IN ( [first pivoted column], [second pivoted column],  
    ... [last pivoted column])  
) AS <alias for the pivot table>  
<optional ORDER BY clause>;
```

PIVOT

- Aplicando:

```
SELECT 'Soma_Mes' SOMA_MES
      , [1]
      , [2]
      , [3]
      , [4]
      , [5]
      , [6]
      , [7]
      , [8]
      , [9]
      , [10]
      , [11]
      , [12]
```



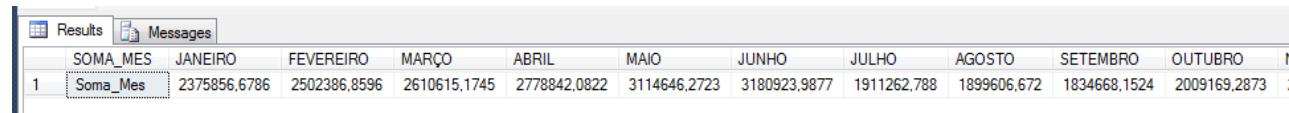
SOMA_MES	1	2	3	4	5	6	7	8	9	10	11	12
1 Soma_Mes	2375856,6786	2502386,8596	2610615,1745	2778842,0822	3114646,2723	3180923,9877	1911262,788	1899606,672	1834668,1524	2009169,2873	2076069,6045	3064629,6616

```
FROM (
SELECT D.MonthNumberOfYear MES, F.SalesAmount SOMA_MES
FROM FactInternetSales F JOIN DimDate D
     ON F.OrderDateKey = D.DateKey
) aa
PIVOT (SUM(Soma_Mes)
FOR aa.mes IN ([1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12]))P
```

PIVOT

- Melhorando o output:

```
SELECT 'Soma_Mes' SOMA_MES
      , [1] AS JANEIRO
      , [2] AS FEVEREIRO
      , [3] AS MARÇO
      , [4] AS ABRIL
      , [5] AS MAIO
      , [6] AS JUNHO
      , [7] AS JULHO
      , [8] AS AGOSTO
      , [9] AS SETEMBRO
      , [10] AS OUTUBRO
      , [11] AS NOVEMBRO
      , [12] AS DEZEMBRO
```



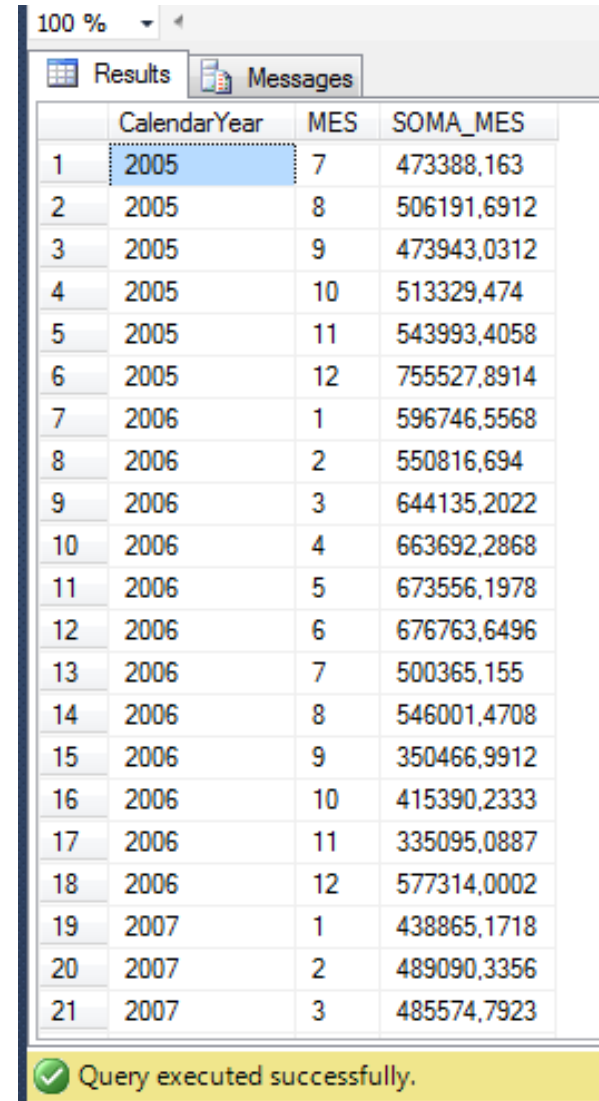
	SOMA_MES	JANEIRO	FEVEREIRO	MARÇO	ABRIL	MAIO	JUNHO	JULHO	AGOSTO	SETEMBRO	OUTUBRO	NOVEMBRO	DEZEMBRO
1	Soma_Mes	2375856,6786	2502386,8596	2610615,1745	2778842,0822	3114646,2723	3180923,9877	1911262,788	1899606,672	1834668,1524	2009169,2873	2009169,2873	2009169,2873

```
FROM (
SELECT MonthNumberOfYear, SalesAmount
FROM FactInternetSales F JOIN DimDate D
    ON F.OrderDateKey = D.DateKey
) aa
PIVOT (SUM(SalesAmount)
FOR MonthNumberOfYear IN ([1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12]))P
```

Mais informação...

- Ou ainda, sabermos quais os valores de vendas em cada mês, em cada ano....

```
SELECT D.CalendarYear, D.MonthNumberOfYear  
MES, sum(F.SalesAmount) SOMA_MES  
FROM FactInternetSales F JOIN DimDate D  
ON F.OrderDateKey = D.DateKey  
GROUP BY D.CalendarYear,  
D.MonthNumberOfYear  
ORDER BY 1,2
```



100 %

Results Messages

	CalendarYear	MES	SOMA_MES
1	2005	7	473388,163
2	2005	8	506191,6912
3	2005	9	473943,0312
4	2005	10	513329,474
5	2005	11	543993,4058
6	2005	12	755527,8914
7	2006	1	596746,5568
8	2006	2	550816,694
9	2006	3	644135,2022
10	2006	4	663692,2868
11	2006	5	673556,1978
12	2006	6	676763,6496
13	2006	7	500365,155
14	2006	8	546001,4708
15	2006	9	350466,9912
16	2006	10	415390,2333
17	2006	11	335095,0887
18	2006	12	577314,0002
19	2007	1	438865,1718
20	2007	2	489090,3356
21	2007	3	485574,7923

✓ Query executed successfully.

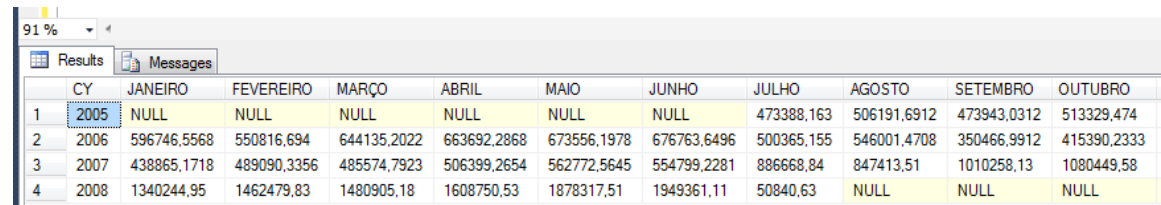
PIVOT

- Usando PIVOT....

```

SELECT      CY
            , [1] AS JANEIRO
            , [2] AS FEVEREIRO
            , [3] AS MARÇO
            , [4] AS ABRIL
            , [5] AS MAIO
            , [6] AS JUNHO
            , [7] AS JULHO
            , [8] AS AGOSTO
            , [9] AS SETEMBRO
            , [10] AS OUTUBRO
            , [11] AS NOVEMBRO
            , [12] AS DEZEMBRO

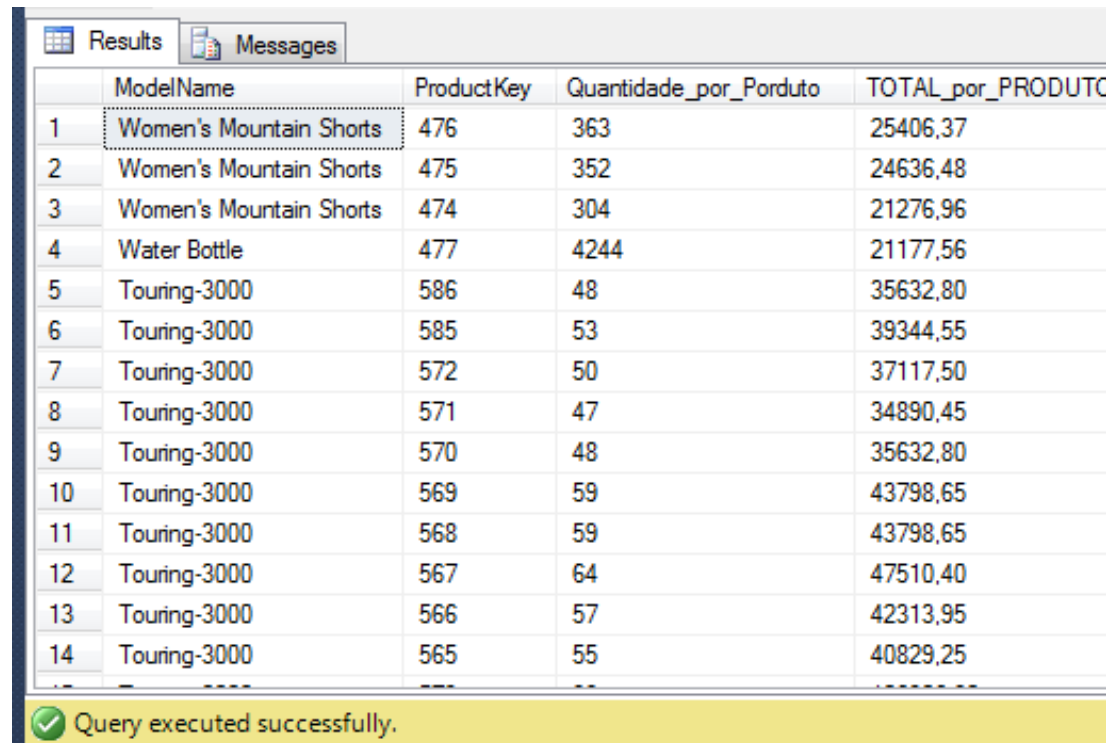
FROM (
SELECT MonthNumberOfYear, SalesAmount, CalendarYear as CY
FROM FactInternetSales F JOIN DimDate D
    ON F.OrderDateKey = D.DateKey
) aa
PIVOT (SUM(SalesAmount)
FOR MonthNumberOfYear IN ([1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12]))P
ORDER BY 1
    
```



	CY	JANEIRO	FEVEREIRO	MARÇO	ABRIL	MAIO	JUNHO	JULHO	AGOSTO	SETEMBRO	OUTUBRO
1	2005	NULL	NULL	NULL	NULL	NULL	NULL	473388,163	506191,6912	473943,0312	513329,474
2	2006	596746,5568	550816,694	644135,2022	663692,2868	673556,1978	676763,6496	500365,155	546001,4708	350466,9912	415390,2333
3	2007	438865,1718	489090,3356	485574,7923	506399,2654	562772,5645	554799,2281	886668,84	847413,51	1010258,13	1080449,58
4	2008	1340244,95	1462479,83	1480905,18	1608750,53	1878317,51	1949361,11	50840,63	NULL	NULL	NULL


ROLLUP

- Suponha-se agora que pretendíamos ter uma listagem que por cada modelo de produto nos indicasse o número de produtos vendidos, bem como o respetivo valor.



The screenshot shows a database query results window with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with the following data:

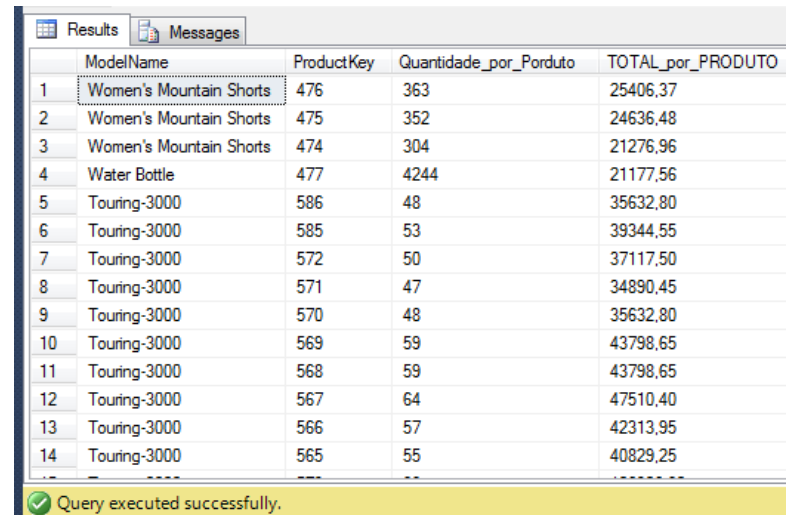
	ModelName	ProductKey	Quantidade_por_Produto	TOTAL_por_PRODUTO
1	Women's Mountain Shorts	476	363	25406,37
2	Women's Mountain Shorts	475	352	24636,48
3	Women's Mountain Shorts	474	304	21276,96
4	Water Bottle	477	4244	21177,56
5	Touring-3000	586	48	35632,80
6	Touring-3000	585	53	39344,55
7	Touring-3000	572	50	37117,50
8	Touring-3000	571	47	34890,45
9	Touring-3000	570	48	35632,80
10	Touring-3000	569	59	43798,65
11	Touring-3000	568	59	43798,65
12	Touring-3000	567	64	47510,40
13	Touring-3000	566	57	42313,95
14	Touring-3000	565	55	40829,25

At the bottom of the window, a yellow status bar indicates:  Query executed successfully.

ROLLUP

- Suponha-se agora que pretendíamos ter uma listagem que por cada modelo de produto nos indicasse o número de produtos vendidos, bem como o respetivo valor.

```
SELECT DP.ModelName, DP.ProductKey, COUNT(FIS.OrderQuantity)  
      Quantidade_por_Porduto, SUM(FIS.SalesAmount) TOTAL_por_PRODUTO  
FROM DIMProduct DP JOIN FactInternetSales FIS  
ON DP.ProductKey = FIS.ProductKey  
GROUP BY DP.ModelName, DP.ProductKey  
ORDER BY 1 DESC, 2 DESC
```

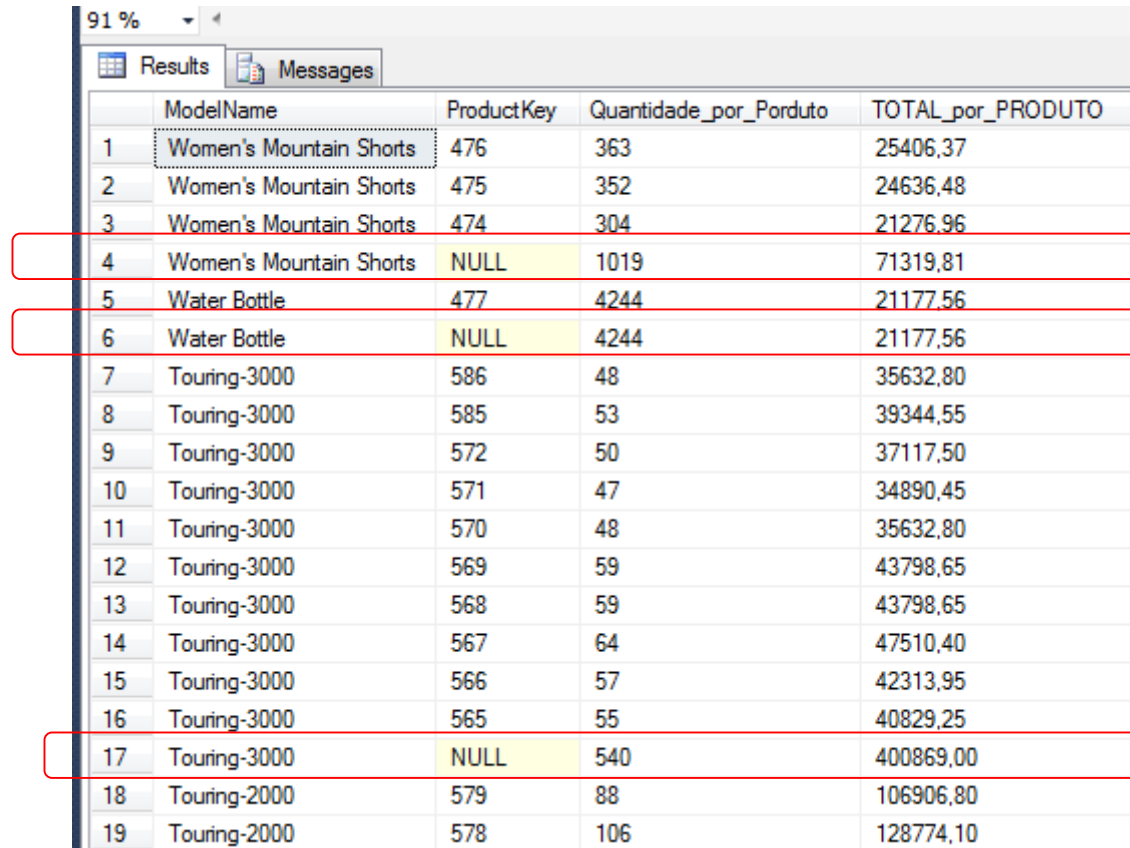


	ModelName	ProductKey	Quantidade_por_Porduto	TOTAL_por_PRODUTO
1	Women's Mountain Shorts	476	363	25406,37
2	Women's Mountain Shorts	475	352	24636,48
3	Women's Mountain Shorts	474	304	21276,96
4	Water Bottle	477	4244	21177,56
5	Touring-3000	586	48	35632,80
6	Touring-3000	585	53	39344,55
7	Touring-3000	572	50	37117,50
8	Touring-3000	571	47	34890,45
9	Touring-3000	570	48	35632,80
10	Touring-3000	569	59	43798,65
11	Touring-3000	568	59	43798,65
12	Touring-3000	567	64	47510,40
13	Touring-3000	566	57	42313,95
14	Touring-3000	565	55	40829,25

Query executed successfully.

ROLLUP

- Adicionalmente, seria interessante ter um valor agrupado para cada modelo, quer da quantidade quer do respetivo valor que total desse modelo.




	ModelName	ProductKey	Quantidade_por_Porduto	TOTAL_por_PRODUTO
1	Women's Mountain Shorts	476	363	25406,37
2	Women's Mountain Shorts	475	352	24636,48
3	Women's Mountain Shorts	474	304	21276,96
4	Women's Mountain Shorts	NULL	1019	71319,81
5	Water Bottle	477	4244	21177,56
6	Water Bottle	NULL	4244	21177,56
7	Touring-3000	586	48	35632,80
8	Touring-3000	585	53	39344,55
9	Touring-3000	572	50	37117,50
10	Touring-3000	571	47	34890,45
11	Touring-3000	570	48	35632,80
12	Touring-3000	569	59	43798,65
13	Touring-3000	568	59	43798,65
14	Touring-3000	567	64	47510,40
15	Touring-3000	566	57	42313,95
16	Touring-3000	565	55	40829,25
17	Touring-3000	NULL	540	400869,00
18	Touring-2000	579	88	106906,80
19	Touring-2000	578	106	128774,10

ROLLUP

- ROLLUP é uma das variantes do GROUP BY, em que cria agrupamentos e sumarizações de acordo com as colunas agrupadas.
- No final de cada agrupamento realizado pelo ROLLUP podem observar-se o totalizador geral na última linha do resultset.

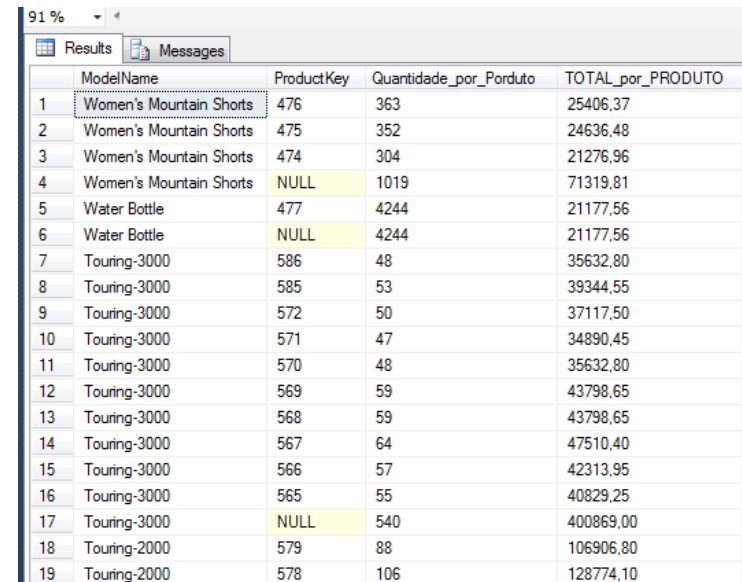
193	Classic Vest	471	168	10668,00
194	Classic Vest	NULL	562	35687,00
195	Bike Wash	484	908	7218,60
196	Bike Wash	NULL	908	7218,60
197	All-Purpose Bike Stand	486	249	39591,00
198	All-Purpose Bike Stand	NULL	249	39591,00
199	NULL	NULL	60398	29358677,2207

 Query executed successfully.

ROLLUP

- Aplicando ROLLUP:

```
SELECT DP.ModelName, DP.ProductKey,  
       COUNT(FIS.OrderQuantity) Quantidade_por_Porduto,  
       SUM(FIS.SalesAmount) TOTAL_por_PRODUTO  
FROM DIMProduct DP JOIN FactInternetSales FIS  
ON DP.ProductKey = FIS.ProductKey  
GROUP BY ROLLUP (DP.ModelName, DP.ProductKey)  
ORDER BY 1 DESC, 2 DESC
```



	ModelName	ProductKey	Quantidade_por_Porduto	TOTAL_por_PRODUTO
1	Women's Mountain Shorts	476	363	25406,37
2	Women's Mountain Shorts	475	352	24636,48
3	Women's Mountain Shorts	474	304	21276,96
4	Women's Mountain Shorts	NULL	1019	71319,81
5	Water Bottle	477	4244	21177,56
6	Water Bottle	NULL	4244	21177,56
7	Touring-3000	586	48	35632,80
8	Touring-3000	585	53	39344,55
9	Touring-3000	572	50	37117,50
10	Touring-3000	571	47	34890,45
11	Touring-3000	570	48	35632,80
12	Touring-3000	569	59	43798,65
13	Touring-3000	568	59	43798,65
14	Touring-3000	567	64	47510,40
15	Touring-3000	566	57	42313,95
16	Touring-3000	565	55	40829,25
17	Touring-3000	NULL	540	400869,00
18	Touring-2000	579	88	106906,80
19	Touring-2000	578	106	128774,10

CUBE

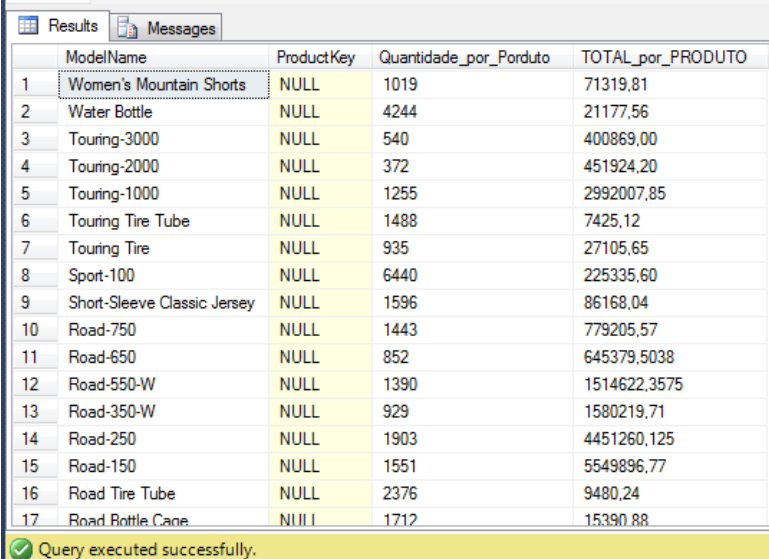
- A sintaxe do CUBE é idêntica à do ROLLUP, a diferença é que o CUBE apresenta totais para cada agrupamento (neste caso fá-lo para cada ProductKey, dentro de cada ModelName)
- Para o exemplo em análise o número de linhas no caso do ROLLUP é: 199 e o no caso do CUBE: 357....

	ModelName	ProductKey	Quantidade_por_Porduto	TOTAL_por_PRODUTO
1	Sport-100	214	2230	78027,70
2	NULL	214	2230	78027,70
3	Sport-100	217	2085	72954,15
4	NULL	217	2085	72954,15
5	Sport-100	222	2125	74353,75
6	NULL	222	2125	74353,75
7	Cycling Cap	225	2190	19688,10
8	NULL	225	2190	19688,10
9	Long-Sleeve Logo Jersey	228	429	21445,71
10	NULL	228	429	21445,71
11	Long-Sleeve Logo Jersey	231	442	22095,58
12	NULL	231	442	22095,58
13	Long-Sleeve Logo Jersey	234	452	22595,48
14	NULL	234	452	22595,48
15	Long-Sleeve Logo Jersey	237	413	20645,87
16	NULL	237	413	20645,87
17	Road-150	310	336	1202298,72
18	NULL	310	336	1202298,72

```
SELECT DP.ModelName, DP.ProductKey,  
COUNT(FIS.OrderQuantity) Quantidade_por_Porduto,  
SUM(FIS.SalesAmount) TOTAL_por_PRODUTO  
FROM DIMProduct DP JOIN FactInternetSales FIS  
ON DP.ProductKey = FIS.ProductKey  
GROUP BY CUBE (DP.ModelName, DP.ProductKey)
```

GROUPING SETS

- A função GROUPING SETS no GROUP BY possibilita a geração de totais dos dados utilizando as colunas inseridas na função, numa única consulta.
- Diferente das funções ROLLUP e CUBE, a função GROUPING SETS não retorna um total geral.



	ModelName	ProductKey	Quantidade_por_Porduto	TOTAL_por_PRODUTO
1	Women's Mountain Shorts	NULL	1019	71319,81
2	Water Bottle	NULL	4244	21177,56
3	Touring-3000	NULL	540	400869,00
4	Touring-2000	NULL	372	451924,20
5	Touring-1000	NULL	1255	2992007,85
6	Touring Tire Tube	NULL	1488	7425,12
7	Touring Tire	NULL	935	27105,65
8	Sport-100	NULL	6440	225335,60
9	Short-Sleeve Classic Jersey	NULL	1596	86168,04
10	Road-750	NULL	1443	779205,57
11	Road-650	NULL	852	645379,5038
12	Road-550-W	NULL	1390	1514622,3575
13	Road-350-W	NULL	929	1580219,71
14	Road-250	NULL	1903	4451260,125
15	Road-150	NULL	1551	5549896,77
16	Road Tire Tube	NULL	2376	9480,24
17	Road Bottle Cage	NULL	1712	15390,88

```
SELECT DP.ModelName, DP.ProductKey,  
COUNT(FIS.OrderQuantity) Quantidade_por_Porduto,  
SUM(FIS.SalesAmount) TOTAL_por_PRODUTO  
FROM DIMProduct DP JOIN FactInternetSales FIS  
ON DP.ProductKey = FIS.ProductKey  
GROUP BY GROUPING SETS (DP.ModelName, DP.ProductKey)  
ORDER BY 1 DESC
```

E ANTES...

- Algumas destas funções estão disponíveis apenas para versões de SQLServer posteriores a 2008, e antes?
- O mesmo resultado poderia ser obtido com recurso ao UNION ALL
- Mas bem mais complexo

```
SELECT
    *
FROM (

    SELECT DP.ModelName, DP.ProductKey, COUNT(FIS.OrderQuantity)
    Quantidade_por_Porduto, SUM(FIS.SalesAmount)
    TOTAL_por_PRODUTO
        FROM DIMProduct DP JOIN FactInternetSales FIS
            ON DP.ProductKey = FIS.ProductKey
        GROUP BY DP.ModelName, DP.ProductKey

    UNION ALL

        SELECT DP.ModelName, 1, COUNT(FIS.OrderQuantity)
    Quantidade_por_Porduto, SUM(FIS.SalesAmount)
    TOTAL_por_PRODUTO
        FROM DIMProduct DP JOIN FactInternetSales FIS
            ON DP.ProductKey = FIS.ProductKey
        GROUP BY DP.ModelName

    UNION ALL

        SELECT 'Total', 1, COUNT(FIS.OrderQuantity)
    Quantidade_por_Porduto, SUM(FIS.SalesAmount)
    TOTAL_por_PRODUTO
        FROM DIMProduct DP JOIN FactInternetSales FIS
            ON DP.ProductKey = FIS.ProductKey

) A
ORDER BY
    (CASE WHEN A.ModelName = 'TOTAL' THEN 1 ELSE 0 END),
    A.ModelName DESC,
    (CASE WHEN A.ProductKey = 1 THEN 1 ELSE 0 END),
    A.ProductKey DESC
```

E ANTES...

```

SELECT
    *
FROM (

SELECT DP.ModelName, DP.ProductKey, COUNT(FIS.OrderQuantity)
Quantidade_por_Porduto, SUM(FIS.SalesAmount)
TOTAL_por_PRODUTO
    FROM DIMProduct DP JOIN FactInternetSales FIS
        ON DP.ProductKey = FIS.ProductKey
    GROUP BY DP.ModelName, DP.ProductKey

UNION ALL

    SELECT DP.ModelName, 1, COUNT(FIS.OrderQuantity)
Quantidade_por_Porduto, SUM(FIS.SalesAmount)
TOTAL_por_PRODUTO
    FROM DIMProduct DP JOIN FactInternetSales FIS
        ON DP.ProductKey = FIS.ProductKey
    GROUP BY DP.ModelName

UNION ALL

    SELECT 'Total', 1, COUNT(FIS.OrderQuantity)
Quantidade_por_Porduto, SUM(FIS.SalesAmount)
TOTAL_por_PRODUTO
    FROM DIMProduct DP JOIN FactInternetSales FIS
        ON DP.ProductKey = FIS.ProductKey

) A
ORDER BY
    (CASE WHEN A.ModelName = 'TOTAL' THEN 1 ELSE 0 END),
    A.ModelName DESC,
    (CASE WHEN A.ProductKey = 1 THEN 1 ELSE 0 END),
    A.ProductKey DESC
    
```

100 %

Results Messages

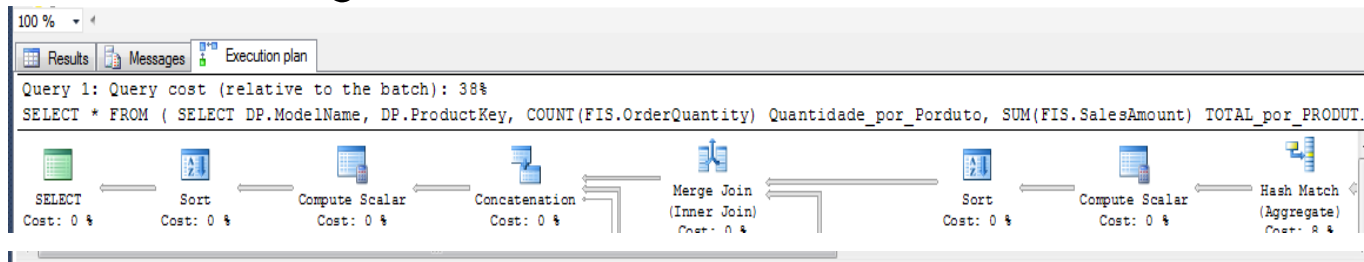
	ModelName	ProductKey	Quantidade_por_Porduto	TOTAL_por_PRODUTO
1	Women's Mountain Shorts	476	363	25406,37
2	Women's Mountain Shorts	475	352	24636,48
3	Women's Mountain Shorts	474	304	21276,96
4	Women's Mountain Shorts	1	1019	71319,81
5	Water Bottle	477	4244	21177,56
6	Water Bottle	1	4244	21177,56
7	Touring-3000	586	48	35632,80
8	Touring-3000	585	53	39344,55
9	Touring-3000	572	50	37117,50
10	Touring-3000	571	47	34890,45
11	Touring-3000	570	48	35632,80
12	Touring-3000	569	59	43798,65
13	Touring-3000	568	59	43798,65
14	Touring-3000	567	64	47510,40
15	Touring-3000	566	57	42313,95
16	Touring-3000	565	55	40829,25

Query executed successfully.

Performance

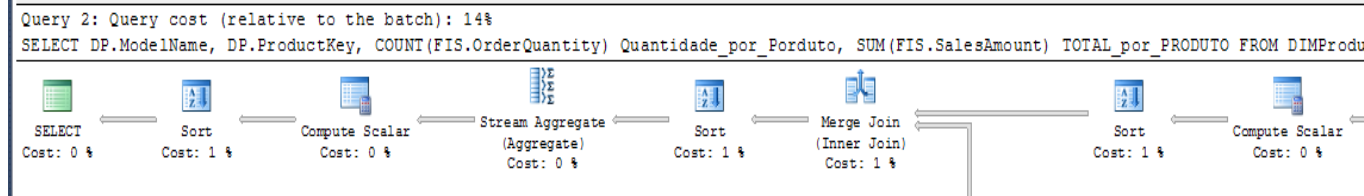
- Para além de uma escrita, mais simples, mesmo em termos de execução, as funções são mais eficientes.

(! Atenção tendo em conta o volume e dados envolvidos !)



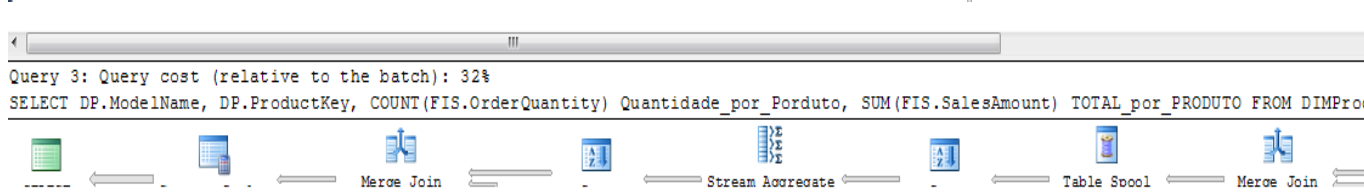
Query1 (38%) (Rs:199)

UNION ALL



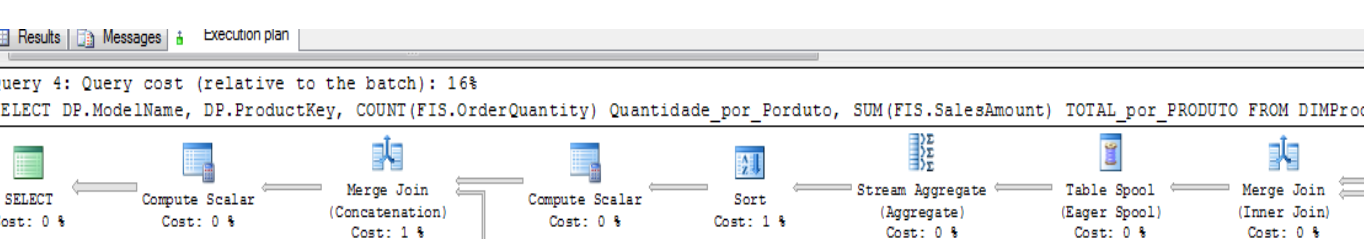
Query2 (14%) (Rs:199)

ROLLUP



Query2 (32%) (Rs:357)

CUBE



Query4 (16%) (Rs:198)

GroupingSets

Revisão

- Vimos:

CTE - Common Table Expressions

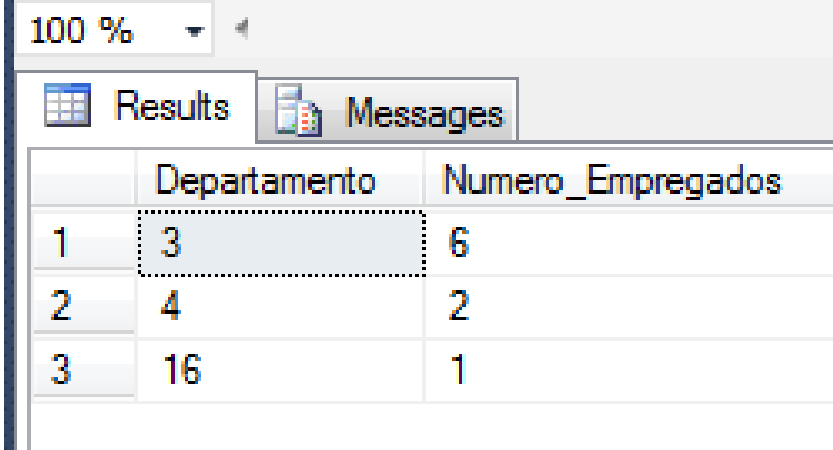
SQL - Recursivo

PIVOT (e UNPIVOT)

GROUP BY – Variantes...

Exercícios

- Considere ainda a nossa tabela “MyEmployees”. Recorrendo a CTE, crie uma listagem que mostre o número de empregados por departamento, ordenados do maior para o menor.



100 %

Results Messages

	Departamento	Numero_Empregados
1	3	6
2	4	2
3	16	1

Exercícios

- Suponha que numa estrutura de dados tem uma entidade funcionário, onde existem relações de ascendência (avô, pai e filho). Crie uma listagem que a demonstre.

```
CREATE TABLE Funcionarios(  
    Id_funcionario INT NOT NULL,  
    Name VARCHAR(100) NOT NULL,  
    ParentId INT NULL  
);
```

```
INSERT INTO Funcionarios VALUES (1, 'Miguel', NULL); -- Avo  
INSERT INTO Funcionarios VALUES (2, 'Filipe', 1); -- Filho do avo Miguel  
INSERT INTO Funcionarios VALUES (3, 'Rui', 2); -- Filho do pai Filipe e Neto Avo Miguel
```

```
INSERT INTO Funcionarios VALUES (5, 'Joao', NULL); -- Avo  
INSERT INTO Funcionarios VALUES (6, 'Ana', 5); -- Filha do avo Joao  
INSERT INTO Funcionarios VALUES (7, 'Nuno', 5); -- Filho do avo Joao  
INSERT INTO Funcionarios VALUES (4, 'Rita', 6); -- Filha da mae Ana e Neta do avo Joao
```

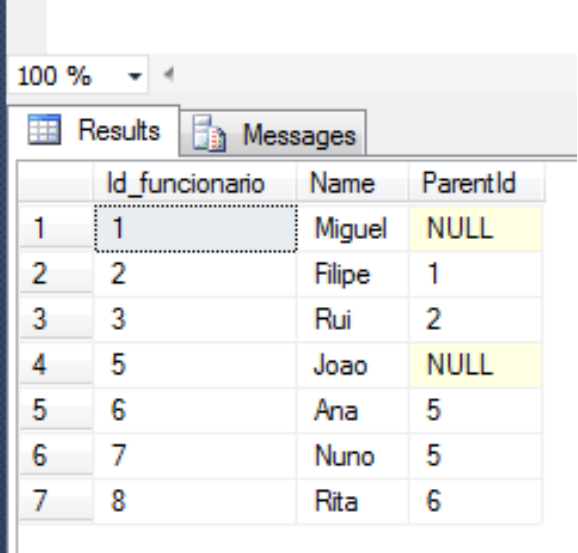
Exercícios

- Suponha que numa estrutura de dados tem uma entidade funcionário, onde existem relações de paternidade (avô, pai e filho). Crie uma listagem que a demonstre.

```
CREATE TABLE Funcionarios(  
    Id_funcionario INT NOT NULL,  
    Name VARCHAR(100) NOT NULL,  
    ParentId INT NULL  
);
```

```
INSERT INTO Funcionarios VALUES (1, 'Miguel', NULL); -- Avo  
INSERT INTO Funcionarios VALUES (2, 'Filipe', 1); -- Filho do avo Miguel  
INSERT INTO Funcionarios VALUES (3, 'Rui', 2); -- Filho do pai Filipe e Neto Avo Miguel
```

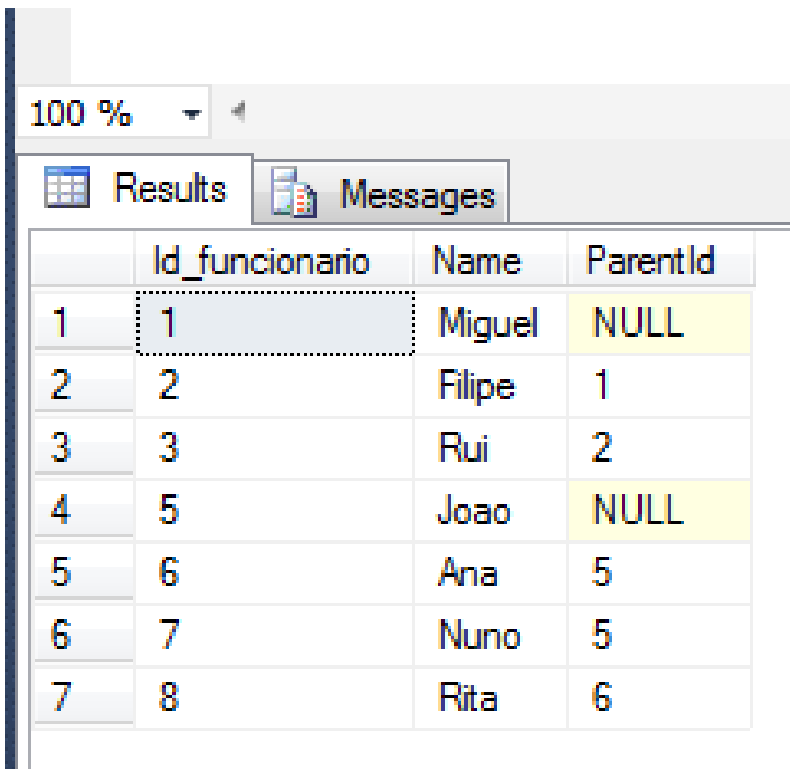
```
INSERT INTO Funcionarios VALUES (5, 'Joao', NULL); -- Avo  
INSERT INTO Funcionarios VALUES (6, 'Ana', 5); -- Filha do avo Joao  
INSERT INTO Funcionarios VALUES (7, 'Nuno', 5); -- Filho do avo Joao  
INSERT INTO Funcionarios VALUES (8, 'Rita', 6); -- Filha da mae Ana e Neta do avo Joao
```



	Id_funcionario	Name	ParentId
1	1	Miguel	NULL
2	2	Filipe	1
3	3	Rui	2
4	5	Joao	NULL
5	6	Ana	5
6	7	Nuno	5
7	8	Rita	6

Exercícios

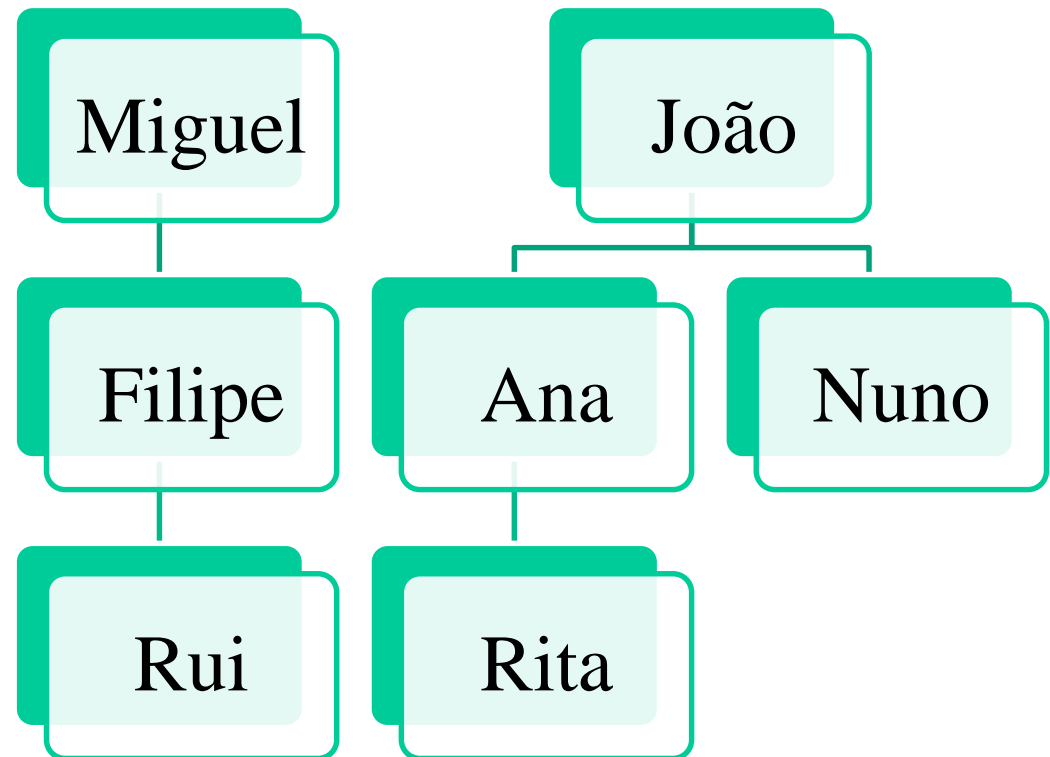
- Suponha que numa estrutura de dados tem uma entidade funcionário, onde existem relações de paternidade (avô, pai e filho). Crie uma listagem que a demonstre.



100 %

Results Messages

	Id_funcionario	Name	ParentId
1	1	Miguel	NULL
2	2	Filipe	1
3	3	Rui	2
4	5	Joao	NULL
5	6	Ana	5
6	7	Nuno	5
7	8	Rita	6



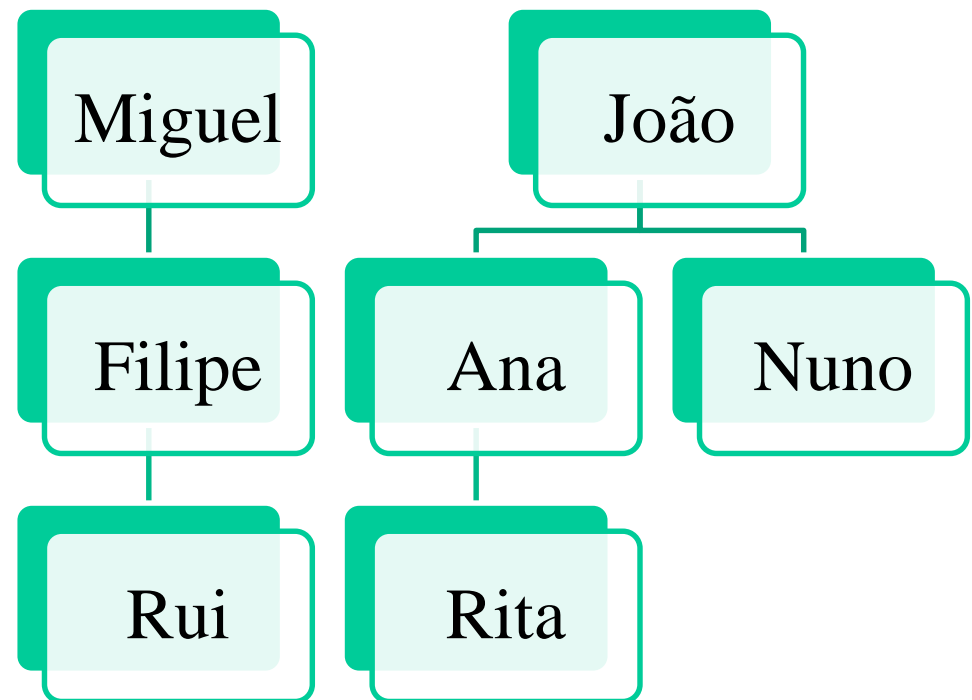
Exercícios

- Suponha que numa estrutura de dados tem uma entidade funcionário, onde existem relações de paternidade (avô, pai e filho). Crie uma listagem que a demonstre.

100 %

Results Messages

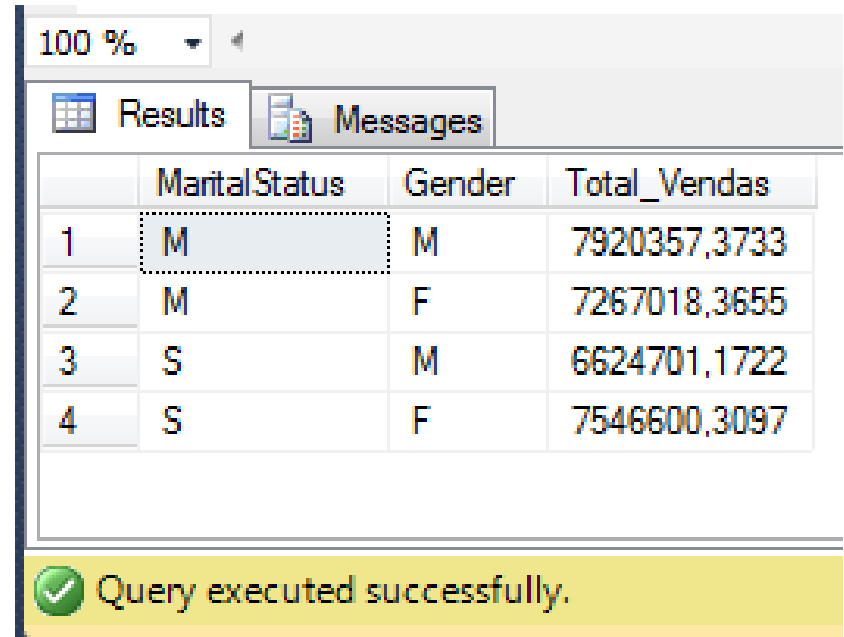
	Id	Nome	Geracao	Acendente
1	1	Miguel	0	NULL
2	5	Joao	0	NULL
3	6	Ana	1	5
4	7	Nuno	1	5
5	4	Rita	2	6
6	2	Filipe	1	1
7	3	Rui	2	2



Exercícios

- Consideremos agora, relativamente à “AdventureWorksDW” o valor total de vendas por género e por estado civil.

```
SELECT MaritalStatus, Gender, SUM(SalesAmount) Total_Vendas
FROM FactInternetSales F JOIN DimCustomer D
    ON F.CustomerKey = D.CustomerKey
GROUP BY MaritalStatus, Gender
ORDER BY 1
```



	MaritalStatus	Gender	Total_Vendas
1	M	M	7920357,3733
2	M	F	7267018,3655
3	S	M	6624701,1722
4	S	F	7546600,3097

✓ Query executed successfully.

Crie uma listagem, com a mesma informação, mas com:

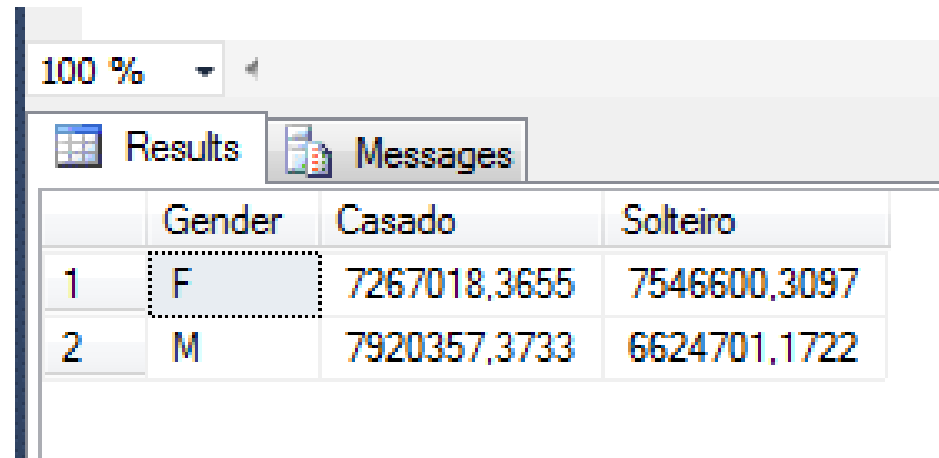
- O estado civil como coluna;
- Como género como coluna.

Exercícios

- Consideremos agora, relativamente à “AdventureWorksDW” o valor total de vendas por género e por estado civil.

Crie uma listagem, com a mesma informação, mas com:

- a) O estado civil como coluna;



100 %

Results Messages

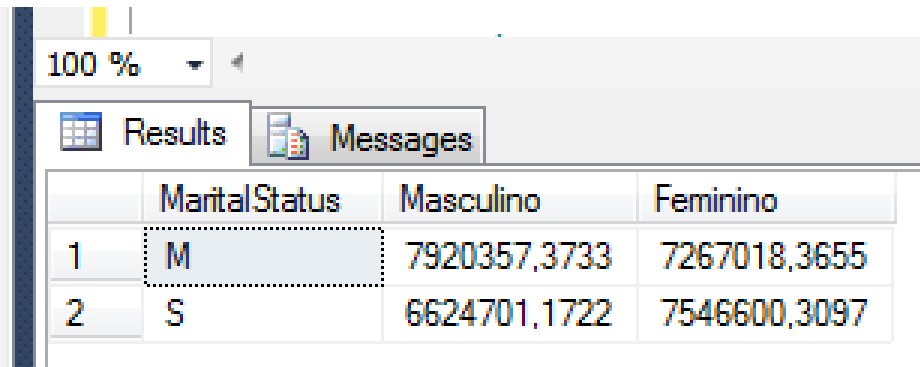
	Gender	Casado	Solteiro
1	F	7267018,3655	7546600,3097
2	M	7920357,3733	6624701,1722

Exercícios

- Consideremos agora, relativamente à “AdventureWorksDW” o valor total de vendas por género e por estado civil.

Crie uma listagem, com a mesma informação, mas com:

b) Com o género como coluna.



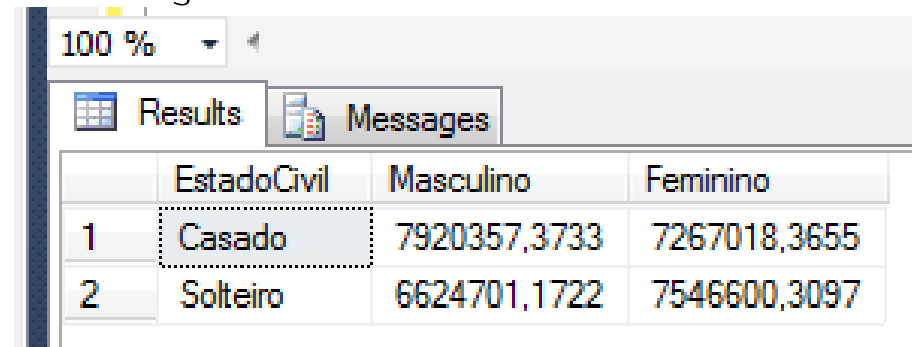
	MaritalStatus	Masculino	Feminino
1	M	7920357,3733	7267018,3655
2	S	6624701,1722	7546600,3097

Exercícios (melhorando o output)

- Consideremos agora, relativamente à “AdventureWorksDW” o valor total de vendas por género e por estado civil.

Crie uma listagem, com a mesma informação, mas com:

b) Com o género como coluna.



The screenshot shows a SQL Server Enterprise Manager interface. At the top, there is a '100 %' zoom level indicator. Below it, there are two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with three columns: 'EstadoCivil', 'Masculino', and 'Feminino'. The table contains two rows of data. The first row shows 'Casado' for 'EstadoCivil', '7920357,3733' for 'Masculino', and '7267018,3655' for 'Feminino'. The second row shows 'Solteiro' for 'EstadoCivil', '6624701,1722' for 'Masculino', and '7546600,3097' for 'Feminino'.

	EstadoCivil	Masculino	Feminino
1	Casado	7920357,3733	7267018,3655
2	Solteiro	6624701,1722	7546600,3097

Exercícios

- Ainda, relativamente à “AdventureWorksDW” o valor total de vendas por género e por estado civil.

Crie uma listagem, mas apresentado os totais gerais e por item.

	MaritalStatus	Gender	Total_Vendas
1	M	F	7267018,3655
2	M	M	7920357,3733
3	M	NULL	15187375,7388
4	S	F	7546600,3097
5	S	M	6624701,1722
6	S	NULL	14171301,4819
7	NULL	NULL	29358677,2207

1) Total: Casado/Feminino
2) Total: Casado/Masculino
3) Total: Casado
4) Total: Solteiro/Feminino
5) Total: Solteiro/Masculino
6) Total: Solteiro
7) Total Geral

Exercícios

- Ainda, relativamente à “AdventureWorksDW” o valor total de vendas por género e por estado civil.

Crie uma listagem, mas apresentado os totais gerais e por item.

Results		Messages	
	MaritalStatus	Gender	Total_Vendas
1	M	F	7267018,3655
2	M	M	7920357,3733
3	M	NULL	15187375,7388
4	S	F	7546600,3097
5	S	M	6624701,1722
6	S	NULL	14171301,4819
7	NULL	NULL	29358677,2207

Exercícios (Melhorando o Output)

- Ainda, relativamente à “AdventureWorksDW” o valor total de vendas por género e por estado civil.

Crie uma listagem, mas apresentado os totais gerais e por item.

100 %

Results Messages

	EstadoCivil	EstadoCivil	Total_Vendas
1	Casado	Feminino	7267018,3655
2	Casado	Masculino	7920357,3733
3	Casado	TOTAL_ITEM	15187375,7388
4	Solteiro	Feminino	7546600,3097
5	Solteiro	Masculino	6624701,1722
6	Solteiro	TOTAL_ITEM	14171301,4819
7	TOTAL_GERAL	TOTAL_ITEM	29358677,2207

1) Total: Casado/Feminino
2) Total: Casado/Masculino
3) Total: Casado
4) Total: Solteiro/Feminino
5) Total: Solteiro/Masculino
6) Total: Solteiro
7) Total Geral

Exercícios

- Ainda, relativamente à “AdventureWorksDW” o valor total de vendas por género e por estado civil.

Podemos ainda criar uma listagem com mais detalhe apresentando também todos os totais.

100 %

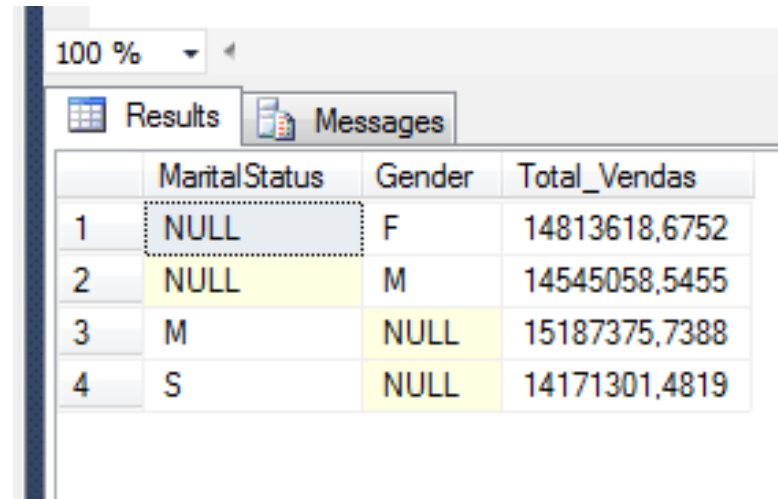
	MaritalStatus	Gender	Total_Vendas
1	NULL	NULL	29358677,2207
2	NULL	F	14813618,6752
3	NULL	M	14545058,5455
4	M	NULL	15187375,7388
5	M	F	7267018,3655
6	M	M	7920357,3733
7	S	NULL	14171301,4819
8	S	F	7546600,3097
9	S	M	6624701,1722

1) Total Geral
2) Total Feminino
3) Total Masculino
4) Total: Casado
5) Total: Casado/Feminino
6) Total: Casado/Masculino
7) Total; Solteiro
8) Total: Solteiro/Feminino
9) Total: Solteiro/Masculino

Exercícios

- Ainda, relativamente à “AdventureWorksDW” o valor total de vendas por género e por estado civil.

Crie uma consulta para listar os valores totais das colunas género e estado civil.



100 %

Results Messages

	MaritalStatus	Gender	Total_Vendas
1	NULL	F	14813618,6752
2	NULL	M	14545058,5455
3	M	NULL	15187375,7388
4	S	NULL	14171301,4819