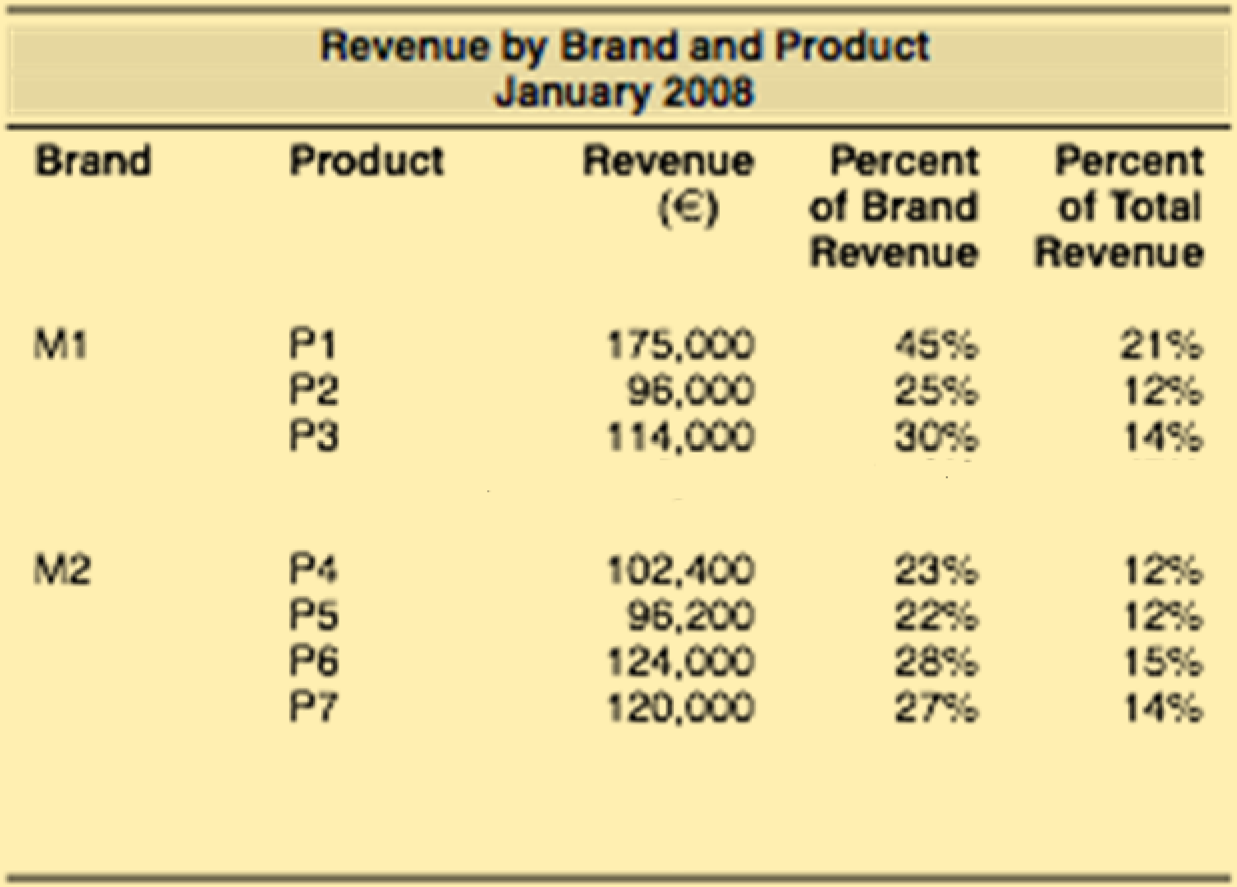
MODERATELY DIFFICULT REPORTS WITH COMPARISON ACROSS AGGREGATION LEVELS

Sales(Customer, Product, Brand, Date, City, Region, Area, Quantity, Revenue, Margin)



Revenue by Brand and Product
January 2008

Brand	Product	Revenue (€)	Percent of Brand Revenue	Percent of Total Revenue
M1	P1	175,000	45%	21%
	P2	96,000	25%	12%
	P3	114,000	30%	14%
M2	P4	102,400	23%	12%
	P5	96,200	22%	12%
	P6	124,000	28%	15%
	P7	120,000	27%	14%

Intuition: OVER clause with PARTITION BY

R

P	...
P1	...
P1	...
P2	...
P2	...
P2	...
P2	...
P2	...

```
SELECT      P, COUNT(*) AS No
FROM        R
GROUP BY    P;
```

$\Rightarrow$

P	No
P1	2
P2	5

```
SELECT      P,
            COUNT(*) OVER (PARTITION BY  P) AS No
FROM        R
ORDER BY    P;
```

$\Rightarrow$

P	No
P1	2
P1	2
P2	5
P2	5
P2	5
P2	5
P2	5

Intuition: OVER clause without PARTITION BY

R

P	...
P1	...
P1	...
P2	...
P2	...
P2	...
P2	...
P2	...

```
SELECT      COUNT(*) AS No
FROM        R
```

$\Rightarrow$

No
7

```
SELECT      P, COUNT(*) OVER( ) AS No
FROM        R
ORDER BY    P;
```

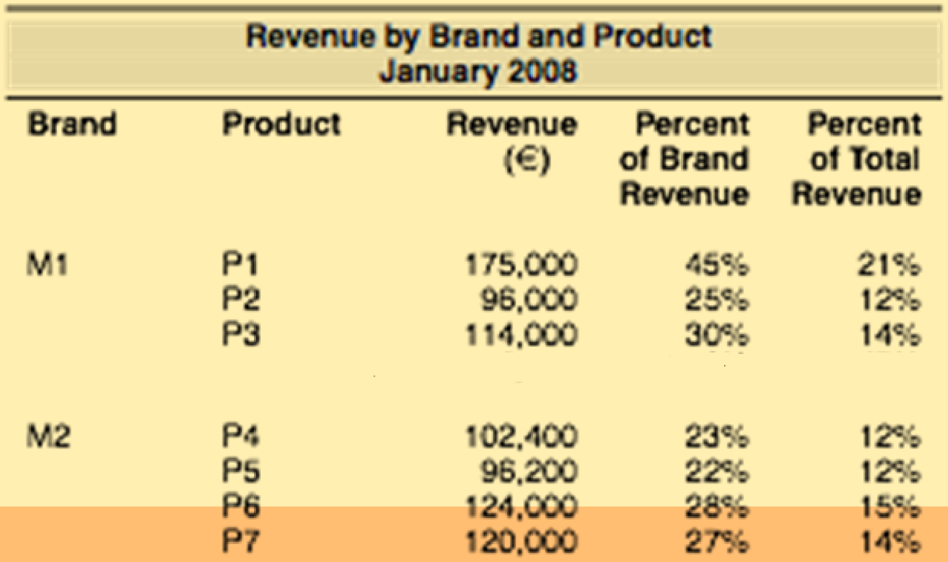
$\Rightarrow$

P	No
P1	7
P1	7
P2	7
P2	7
P2	7
P2	7
P2	7

MODERATELY DIFFICULT REPORTS WITH COMPARISON ACROSS AGGREGATION LEVELS

Sales(Customer, Product, Brand, Date, City, Region, Area, Quantity, Revenue, Margin)

```
WITH temp AS
  SELECT Brand, Product, SUM(Revenue) AS TotRevenue,
  FROM Sales WHERE Year(Date)=2008 and Month(Date)=1
  GROUP BY Brand, Product
SELECT Brand, Product, TotRevenue,
          SUM(TotRevenue) OVER( PARTITION BY Brand) As TotBrandRevenue,
          SUM(TotRevenue) OVER( ) As TotRevenue
FROM temp
ORDER BY Brand, Product
```



Revenue by Brand and Product
January 2008

Brand	Product	Revenue (€)	Percent of Brand Revenue	Percent of Total Revenue
M1	P1	175,000	45%	21%
	P2	96,000	25%	12%
	P3	114,000	30%	14%
M2	P4	102,400	23%	12%
	P5	96,200	22%	12%
	P6	124,000	28%	15%
	P7	120,000	27%	14%

Syntax

SELECT	Select Attributes (S_A), Select Aggregation Functions (S_{AF}),
FROM	Fact table (F) and a dimension table (D1)
WHERE	Where condition (W_C)
GROUP BY	Grouping Attributes (G_A)
HAVING	Having condition (H_C) with aggregation functions (H_{AF})
ORDER BY	Sorting attributes (O_A);

Syntax

SELECT Select Attributes (S_A), Select Aggregation Functions (S_{AF}),
Analytic Function (A_F) **OVER(**
[**PARTITION BY** <attribute list>]
[**ORDER BY** <sort attribute list>
[<window clause>]])
FROM Fact table (F) and a dimension table (D1)
WHERE Where condition (W_C)
GROUP BY Grouping Attributes (G_A)
HAVING Having condition (H_C) with aggregation functions (H_{AF})
ORDER BY Sorting attributes (O_A);

ANALYTIC SQL

SELECT Select Attributes (S_A), Select Aggregation Functions (S_{AF}),
Analytic Function (A_F) **OVER(**
  [**PARTITION BY** <attribute list>]
  [**ORDER BY** <sort attribute list>
   [<window clause>]])
FROM Fact table (F) and a dimension table (D1)
WHERE Where condition (W_C)
GROUP BY Grouping Attributes (G_A)
HAVING Having condition (H_C) with aggregation functions (H_{AF})
ORDER BY Sorting attributes (O_A);

Syntax

ORDER BY O_A

SELECT S_A, S_{AF},

A_F **OVER (...)**

HAVING H_C

GROUP BY G_A

WHERE W_C

FROM F, D1

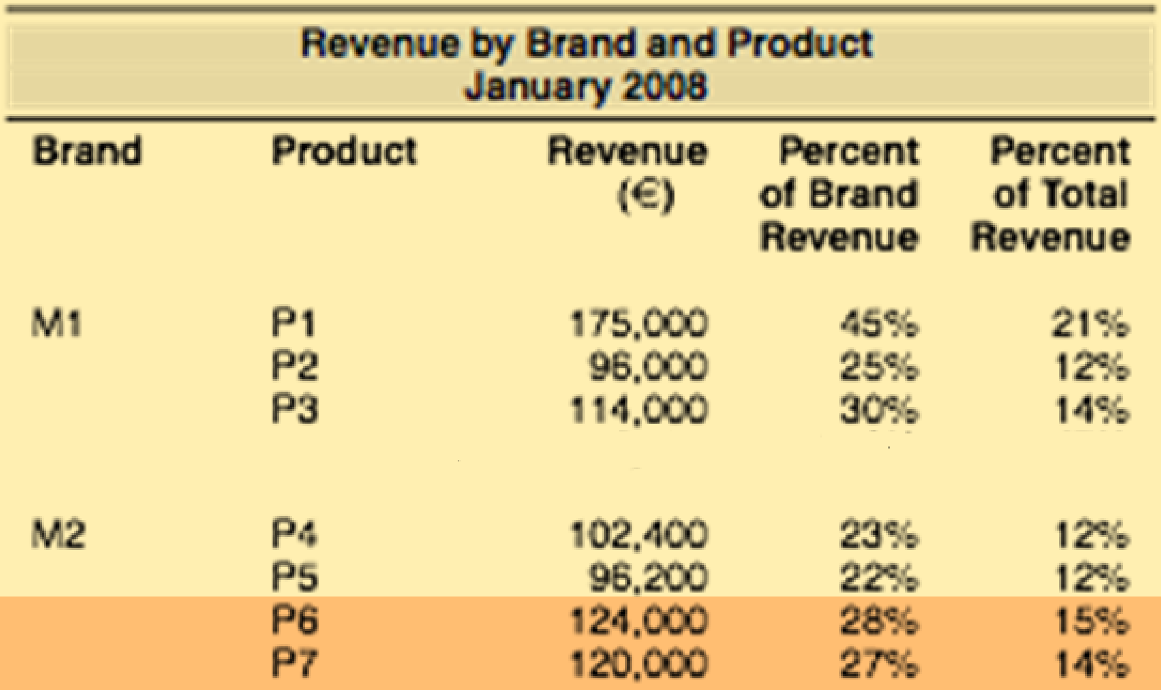
Semantics

$$\begin{tikzcd}[column sep=small]
& \tau_{O_A} \arrow[d, dash] & \\
& \pi^b_{S_A \cup S_{AF} \cup A_F} \arrow[d, dash] & \\
& {}_{G_A}\Omega_{S_{AF} \cup A_F} \arrow[d, dash] & \\
& \sigma_{H_C} \arrow[d, dash] & \\
& {}_{G_A}\gamma_{S_{AF} \cup H_{AF}} \arrow[d, dash] & \\
& \sigma_{W_C} \arrow[d, dash] & \\
& \times \arrow[dl, dash] \arrow[dr, dash] & \\
\text{F} & & \text{D1}
\end{tikzcd}$$

MODERATELY DIFFICULT REPORTS WITH COMPARISON ACROSS AGGREGATION LEVELS

Sales(Customer, Product, Brand, Date, City, Region, Area, Quantity, Revenue, Margin)

```
SELECT Brand, Product, SUM(Revenue) AS TotRevenue,
        SUM(SUM(Revenue)) OVER( PARTITION BY Brand) As TotBrandRevenue,
        SUM(SUM(Revenue)) OVER( ) As TotRevenue
FROM Sales
WHERE Year(Date)=2008 and Month(Date)=1
GROUP BY Brand, Product
ORDER BY Brand, Product
```

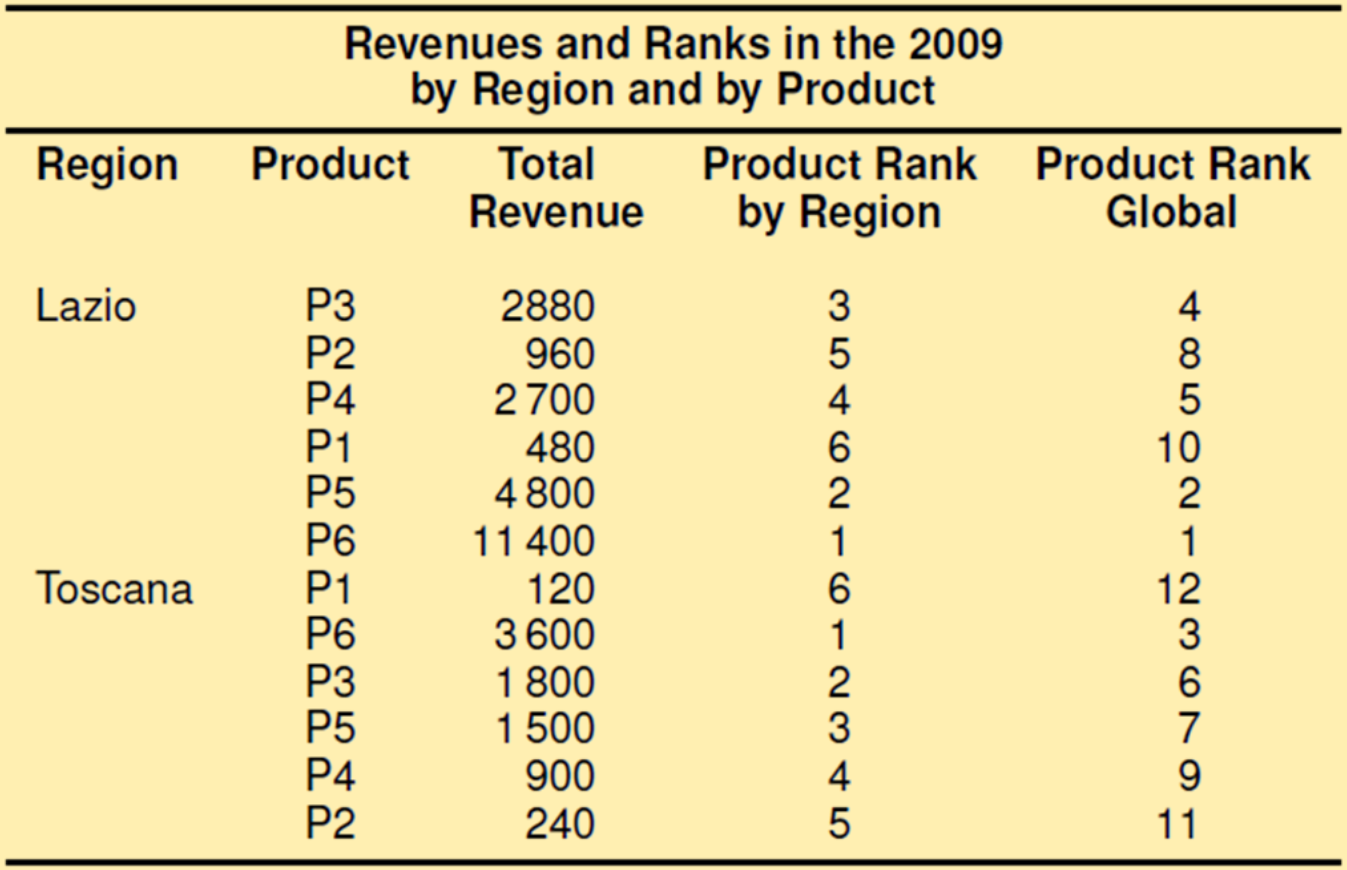


Revenue by Brand and Product
January 2008

Brand	Product	Revenue (€)	Percent of Brand Revenue	Percent of Total Revenue
M1	P1	175,000	45%	21%
	P2	96,000	25%	12%
	P3	114,000	30%	14%
M2	P4	102,400	23%	12%
	P5	96,200	22%	12%
	P6	124,000	28%	15%
	P7	120,000	27%	14%

VERY DIFFICULT REPORTS WITHOUT ANALYTIC SQL: RANK

Sales(Customer, Product, Brand, Date, City, Region, Area, Quantity, Revenue, Margin)



Revenues and Ranks in the 2009 by Region and by Product

Region	Product	Total Revenue	Product Rank by Region	Product Rank Global
Lazio	P3	2880	3	4
	P2	960	5	8
	P4	2 700	4	5
	P1	480	6	10
	P5	4 800	2	2
	P6	11 400	1	1
Toscana	P1	120	6	12
	P6	3 600	1	3
	P3	1 800	2	6
	P5	1 500	3	7
	P4	900	4	9
	P2	240	5	11

Which are the **best 5** products sold in Toscana?

RANK

```
SELECT Customer, Product, SUM(Revenue) AS TotalRev,
        RANK ( ) OVER (ORDER BY SUM(Revenue) ) AS Rank
FROM Sales WHERE Customer IN ('C1', 'C2')
GROUP BY Customer, Product   ORDER BY TotalRev DESC;
```

Customer	Product	TotalRev	Rank
C1	P1	1100	7
C1	P3	1050	6
C2	P1	1000	5
C2	P2	900	4
C2	P4	800	3
C1	P2	250	2
C2	P3	200	1

RANK WITH PARTITIONS

```
SELECT Customer, Product, SUM(Revenue) AS TotalRevenue,
         RANK ( ) OVER (PARTITION BY Customer
           ORDER BY SUM(Revenue) DESC) AS Rank
FROM Sales WHERE Customer IN ('C1', 'C2')
GROUP BY Customer, Product;
```

Customer	Product	TotalRev	Rank
C1	P1	1100	1
C1	P3	1050	2
C1	P2	250	3
C2	P1	1000	1
C2	P2	900	2
C2	P4	800	3
C2	P3	200	4

RANK vs DENSE_RANK vs ROW_NUMBER

```
<RankFunction>()
OVER(
        [PARTITION BY <attribute list>]
         ORDER BY <sort attribute list>
      ) [ AS Ide ]
```

- Consider the values in the ascending order
  - (10; 20; 20; 30; 30; 40)
- RANK() of a value is 1 + the number of values that strictly precedes it
  - ranks (1; 2; 2; 4; 4; 6)
- DENSE_RANK() of a value is 1 + the number of distinct values that precedes it
  - dense ranks (1; 2; 2; 3; 3; 4)
- PERCENT_RANK() is (RANK() – 1) / (TotalRows – 1)
  - percent ranks (0; 0.2; 0.2; 0.6; 0.6; 1)
- ROW_NUMBER() is the row number
  - row numbers (1; 2; 3; 4; 5; 6)
- CUME_DIST() of a value is the number of values lower or equal than it / TotalRows
  - cumulative distribution (0.16; 0.5; 0.5; 0.83; 0.83; 1)
- NTILE(3) is the tertile of the value (3 is a parameter, can be any integer)
  - tertiles (1; 1; 2; 2; 3; 3)

VERY DIFFICULT REPORTS WITHOUT ANALYTIC SQL: EXERCISE AT HOME!

Sales(Customer, Product, Brand, Date, City, Region, Area, Quantity, Revenue, Margin)

We want to partition the customers into four groups:

- **Top5%**, with 5% of customers with the highest amount of revenues.
- **Next15%**, with 15% of other customers with the highest amount of revenues.
- **Middle30%**, with 30% of other customers with the highest amount of revenues.
- **Bottom50%**, with 50 % of the customers with the lowest amount of revenues.

For each customer group we want to know their number, and the percentage of the sum of their revenues compared to total revenue of all sales.

Group	**Number of customers**	**Percent of total revenue**
Top5%	1	20
Next15%	3	50
Middle30%	6	20
Bottom50%	10	10

OTHER ANALYTIC FUNCTIONS

- COUNT(), SUM(), AVG(), MIN(), MAX() … and all standard aggregates

Sales(Brand, Product, Revenue)

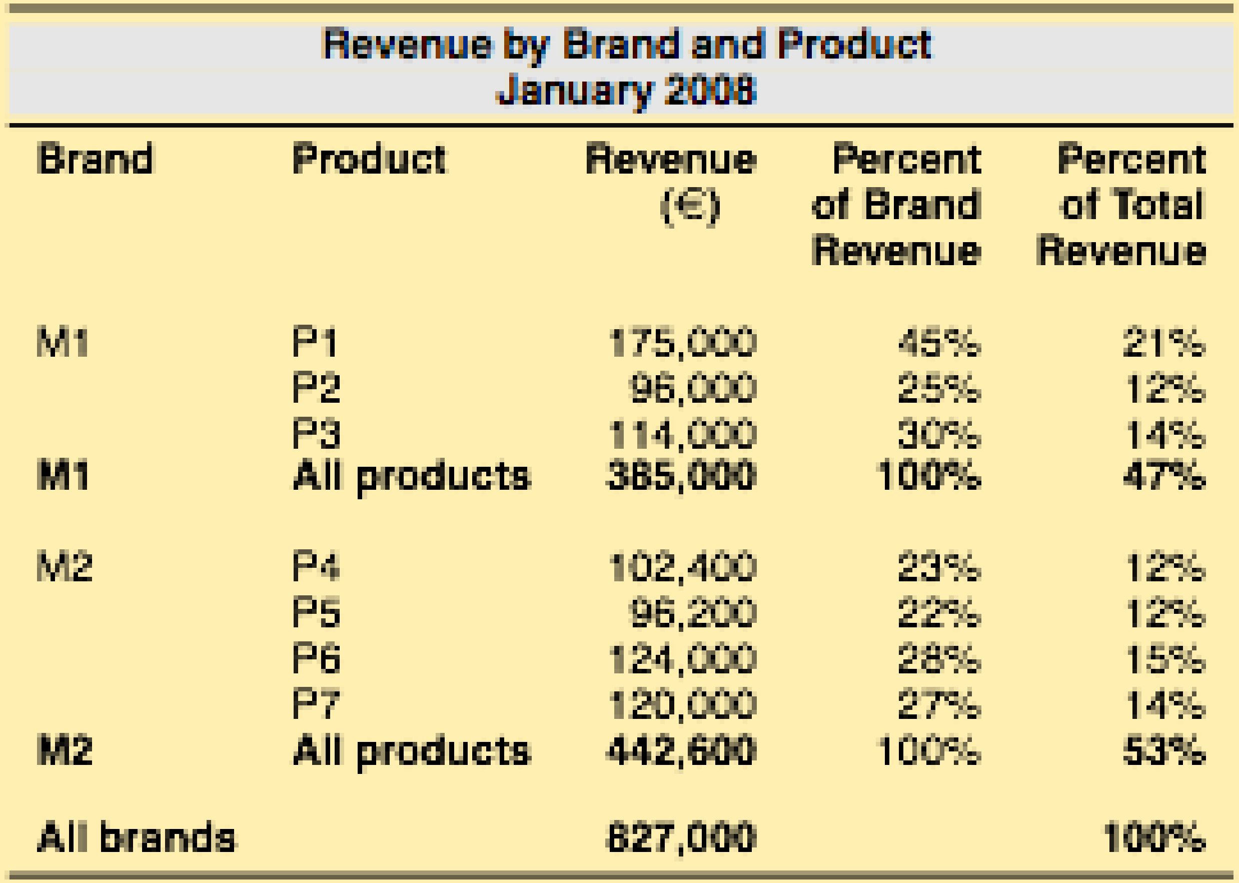
Brand	Product	prodRevenue	PctOverBrand	PctOverTot
B1	P1	40	40	20
B1	P2	60	60	30
B2	P3	20	20	10
B2	P4	80	80	40

```
SELECT Brand, Product,SUM(Revenue) AS prodRevenue,
  100 * SUM(Revenue) / SUM(SUM(Revenue)) OVER(PARTITION BY Brand) AS PctOverBrand,
  100 * SUM(Revenue) / SUM(SUM(Revenue)) OVER() AS PctOverTot
FROM sales
GROUP BY Brand, Product

SELECT Brand, Product,SUM(Revenue) AS prodRevenue,
  100 * RATIO_TO_REPORT(SUM(Revenue)) OVER(PARTITION BY Brand) AS PctOverBrand,
  100 * RATIO_TO_REPORT(SUM(Revenue)) OVER() AS PctOverTot
FROM sales
GROUP BY Brand, Product
```

EXERCISE AT HOME: MODERATELY DIFFICULT REPORTS WITH COMPARISON ACROSS AGGREGATION LEVELS

Sales(Customer, Product, Brand, Date, City, Region, Area, Quantity, Revenue, Margin)



Revenue by Brand and Product
January 2008

Brand	Product	Revenue (€)	Percent of Brand Revenue	Percent of Total Revenue
M1	P1	175,000	45%	21%
	P2	96,000	25%	12%
	P3	114,000	30%	14%
M1	**All products**	**385,000**	**100%**	**47%**
M2	P4	102,400	23%	12%
	P5	96,200	22%	12%
	P6	124,000	28%	15%
	P7	120,000	27%	14%
M2	**All products**	**442,600**	100%	**53%**
All brands		**827,000**		**100%**

OTHER ANALYTIC FUNCTIONS

- LAG(attribute, offset=1, default=NULL) and LEAD(attribute, offset=1, default =NULL)
  - The value of attribute in offset rows before (LAG) or after (LEAD)

```
WITH temp AS (
  SELECT Store, Year, SUM(Sales) as TotalRev
  FROM Sales
  GROUP BY Store, Year )
SELECT Store, Year, TotalRev,
  LEAD(TotalRev, 1, 0)
   OVER(PARTITION BY Store
           ORDER BY Year DESC) AS PrevRev
FROM temp
ORDER BY Store, Year
```

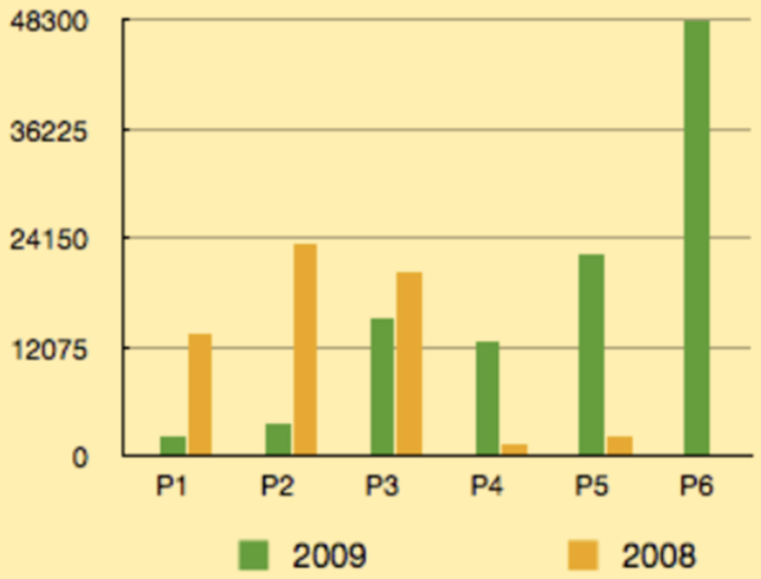
Store	Year	TotalRev	PrevRev
S1	2015	1100	1000
S1	2014	1000	200
S1	2013	200	0
S2	2015	1000	900
S2	2014	900	800
S2	2013	800	200
S2	2012	200	0

MODERATELY DIFFICULT REPORTS WITH COMPARISON BETWEEN COLUMNS (VARIANCE REPORT)

Sales(Customer, Product, Brand, Date, City, Region, Area, Quantity, Revenue, Margin)

Comparison between Revenue by Brand and by Product 2009 – 2008

Brand	Product	Revenue (€) 2009	Revenue (€) 2008	Delta (%)
B1	P1	2 100	13 560	−546
	P2	3 720	23 640	−535
	P3	15 300	20 340	−33
B2	P4	12 600	1 440	89
	P5	22 500	2 100	91
	P6	48 300		100



Delta = 100 x (Revenue2009 - Revenue2008)/Revenue2009

A product may have been sold in one year, but not in the other !

FULL [OUTER] JOIN

```
SELECT  *
FROM    R  NATURAL FULL JOIN S
```

R

A	B
1	a
2	b
3	c

S

A	C
1	x
3	y
5	z

```
SELECT  *
FROM    R  FULL JOIN S ON R.A = S.A
```

A	B	C
1	a	x
2	b	
3	c	y
5		z

```
SELECT  *
FROM    R  FULL JOIN S  USING (A)
-- syntax not available in SQL Server
```

SOLUTION WITH FULL [OUTER] JOIN

```
WITH        Revenue09 AS
            ( SELECT      Brand, Product, SUM(Revenue) AS Revenue2009
              FROM        Sales
              WHERE       YEAR(Date) = 2009
              GROUP BY    Brand, Product
            )
            , Revenue08 AS
            ( SELECT      Brand, Product, SUM(Revenue) AS Revenue2008
              FROM        Sales
              WHERE       YEAR(Date) = 2008
              GROUP BY    Brand, Product
            )

SELECT      Revenue09.Brand AS Brand, Revenue09.Product AS Product
            , Revenue2009
            , Revenue2008
            , CASE
              WHEN Revenue2009 IS NULL THEN −100
              WHEN Revenue2008 IS NULL THEN 100
              ELSE ROUND(100∗(Revenue2009 − Revenue2008) / Revenue2009)
            END AS Delta
FROM        Revenue09 FULL JOIN Revenue08 USING (Brand, Product)
ORDER BY    Brand, Product;
```

SOLUTION USING LAG-LEAD (and NO JOIN)

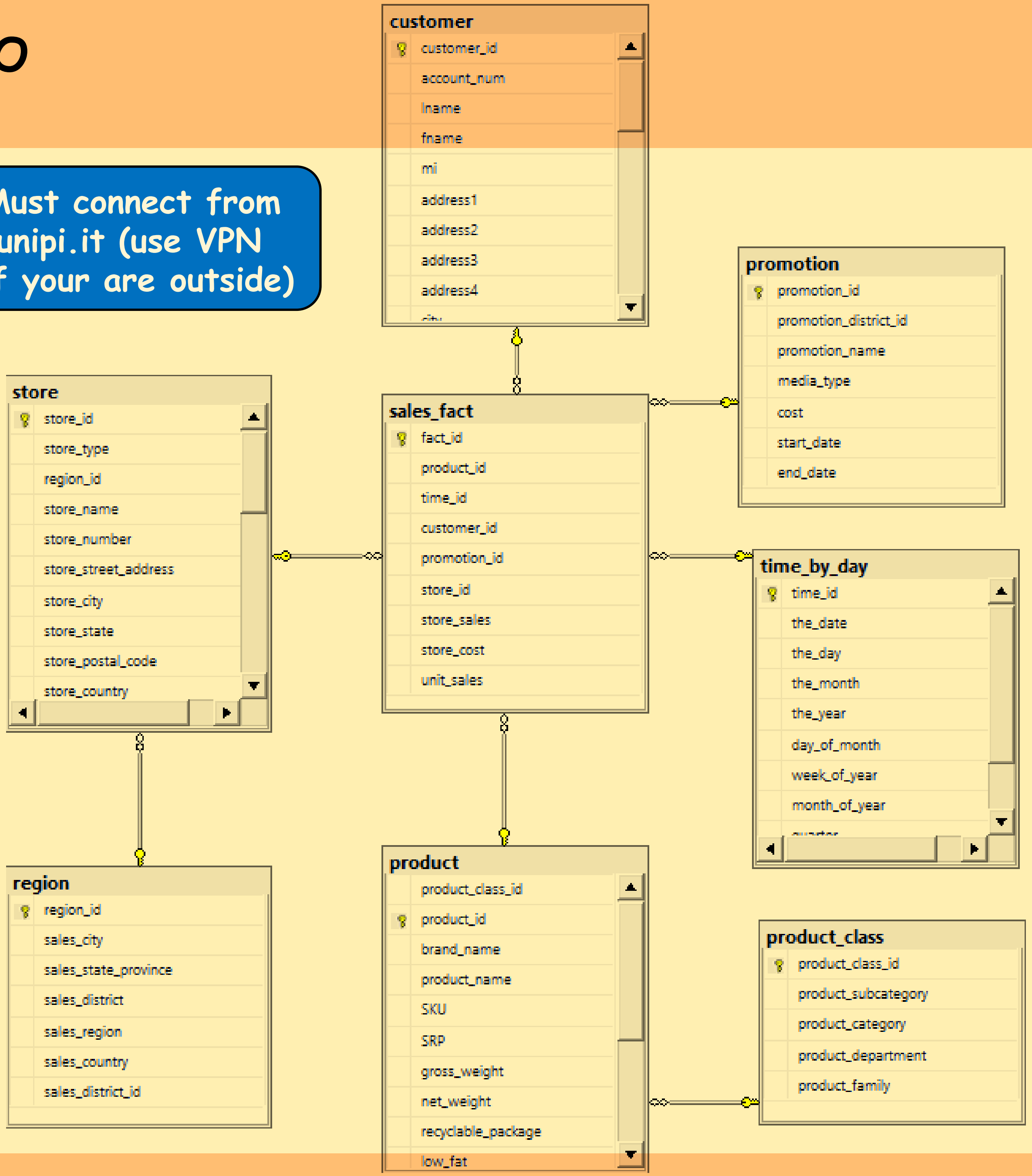
Exercise at Home!

Foodmart datawarehouse DEMO

- RDBMS: Microsoft SQL Server
- **SQL Server: lds.di.unipi.it**
- **Login: dsd   Pwd:**

Must connect from .unipi.it (use VPN if your are outside)

- GUI:
  - SQL Server Management Studio
    - Win only
  - Azure Data Studio
    - Win, Linux, Mac OS


customer
customer_id
account_num
lname
fname
mi
address1
address2
address3
address4
promotion
promotion_id
promotion_district_id
promotion_name
media_type
cost
start_date
end_date
store
store_id
store_type
region_id
store_name
store_number
store_street_address
store_city
store_state
store_postal_code
store_country
sales_fact
fact_id
product_id
time_id
customer_id
promotion_id
store_id
store_sales
store_cost
unit_sales
time_by_day
time_id
the_date
the_day
the_month
the_year
day_of_month
week_of_year
month_of_year
region
region_id
sales_city
sales_state_province
sales_district
sales_region
sales_country
sales_district_id
product
product_class_id
product_id
brand_name
product_name
SKU
SRP
gross_weight
net_weight
recyclable_package
low_fat
product_class
product_class_id
product_subcategory
product_category
product_department
product_family