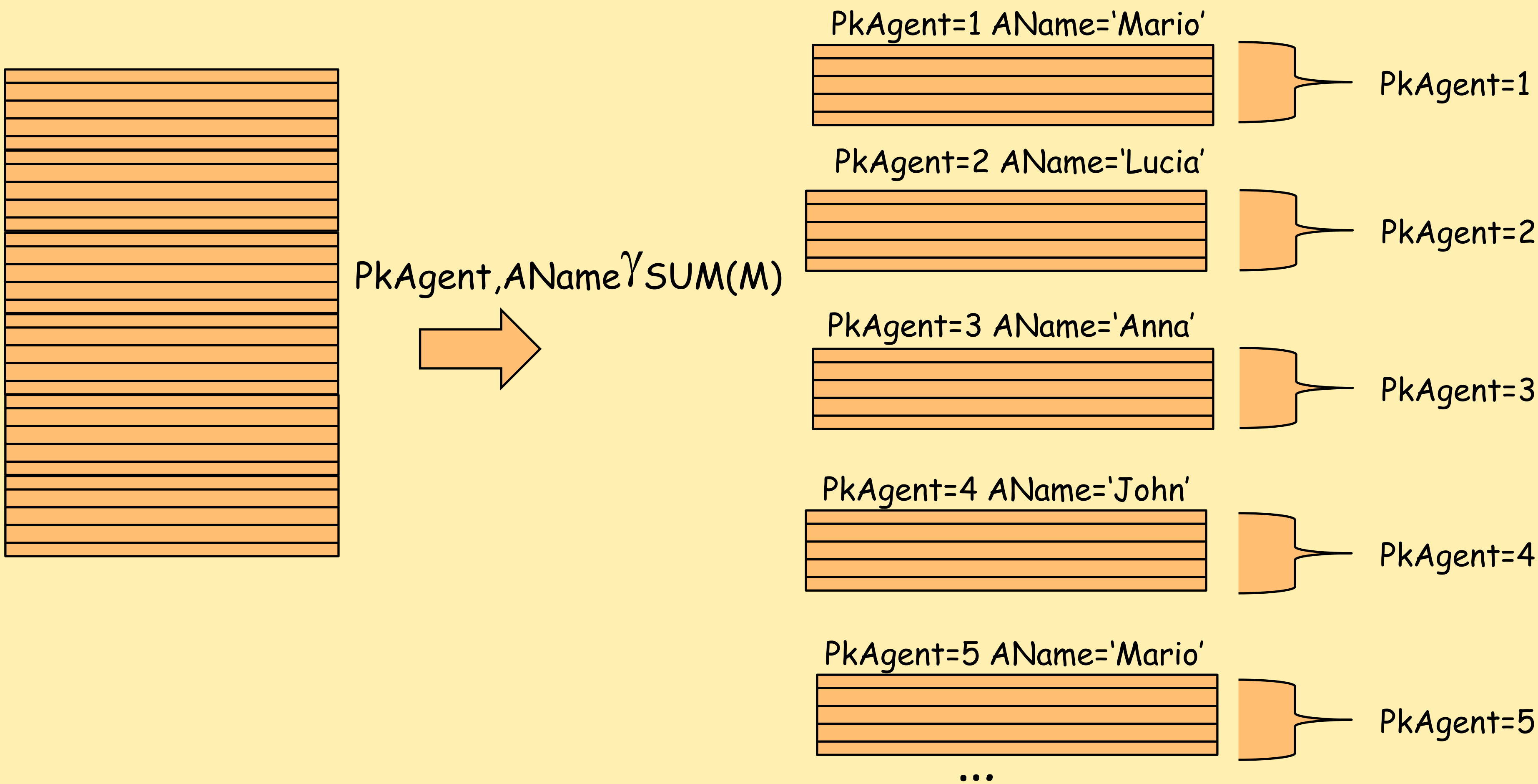
TODAY: RELATIONAL DBMS EXTENSIONS FOR DW

- SQL extensions
- Index and storage structures
- Star query physical plans
- Materialized views
- Optimization techniques for star queries with grouping and aggregations ⇦

FD AND GROUPINGS

PkAgent → AName
implies
groups by **PkAgent, AName** are the same groups as **PkAgent**



$_{\text{PkAgent}}\gamma_{\text{SUM(M)}}$? $\Pi_{\text{PkAgent,SM}}$ (**$_{\text{PkAgent,AName}}\gamma_{\text{SUM(M) As SM}}$**)

SIMPLE QUERY REWRITE OPT.: GROUPING AND PROJECTING

Let $B \notin X$, and $X \to B$

$$_{X}\gamma_{F}(E) \equiv \quad \ldots \quad ({}_{X \cup \{B\}}\gamma_{F}(E))$$

This will be used later on this lesson

QUERY

```
SELECT    PKAgent, SUM(Qty) AS TQ
FROM      Order, Agent
WHERE     FKAgent = PKAgent
GROUP BY  PKAgent
```

QUERY REWRITING

```
SELECT    PKAgent, SUM(Qty) AS TQ
FROM      Order, Agent
WHERE     FKAgent = PKAgent
GROUP BY  PKAgent, AName
```

MATERIALIZED VIEW V

```
SELECT    PKAgent, AName, SUM(Qty) AS TQ
FROM      Order, Agent
WHERE     FKAgent = PKAgent
GROUP BY  PKAgent, AName
```

QUERY REWRITING

```
SELECT    PKAgent, TQ
FROM      V
```

More in detail in the next lesson

SIMPLE QUERY REWRITE OPT.: ANTICIPATING HAVING WRT GROUP BY

$$\sigma_{\phi}({}_{X}\gamma_{F}(E)) \stackrel{?}{\equiv} {}_{X}\gamma_{F}(\sigma_{\phi}(E))$$

Two cases to consider:

1) if ϕ depends only on X, i.e., $\phi = \phi_X$:

$$\sigma_{\phi_X}({}_{X}\gamma_{F}(E)) \equiv {}_{X}\gamma_{F}(\sigma_{\phi_X}(E))$$

QUERY

SELECT	PKAgent, SUM(Qty) AS TQ
FROM	Order, Agent
WHERE	FKAgent = PKAgent
GROUP BY	PKAgent, AName
HAVING	AName **LIKE** 'R%'

QUERY REWRITING

SELECT	PKAgent, SUM(Qty) AS SQ
FROM	Order, Agent
WHERE	FKAgent = PKAgent
	AND AName **LIKE** 'R%'
GROUP BY	PKAgent, AName

SIMPLE QUERY REWRITE OPT.: ANTICIPATING HAVING WRT GROUP BY

$$\sigma_{\phi}(_X\gamma_F(E)) \stackrel{?}{\equiv} {}_X\gamma_F(\sigma_{\phi}(E))$$

Two cases to consider:

2) if ϕ depends on agg. F, i.e., $\phi = \phi_F$, rewriting is possible only in two cases

$$\sigma_{\mathrm{Mb} \geq \mathrm{v}}(_X\gamma_{\mathrm{MAX(b)\ AS\ Mb}}(E)) \equiv {}_X\gamma_{\mathrm{MAX(b)\ AS\ Mb}}(\sigma_{\mathrm{b} \geq \mathrm{v}}(E))$$

$$\sigma_{\mathrm{mb} \leq \mathrm{v}}(_X\gamma_{\mathrm{MIN(b)\ AS\ mb}}(E)) \equiv {}_X\gamma_{\mathrm{MIN(b)\ AS\ mb}}(\sigma_{\mathrm{b} \leq \mathrm{v}}(E))$$

QUERY

```
SELECT    PKAgent, MAX(Qty) AS MQ
FROM      Order, Agent
WHERE     FKAgent = PKAgent
GROUP BY  PKAgent, AName
HAVING    MAX(Qty) >= 10
```

QUERY REWRITING

```
SELECT    PKAgent, MAX(Qty) AS MQ
FROM      Order, Agent
WHERE     FKAgent = PKAgent
          AND Qty >= 10
GROUP BY  PKAgent, AName
```

THE PRE-GROUPING PROBLEM

$$_X\gamma_F(R \underset{f_k=p_k}{\bowtie} S)$$

- The standard way to evaluate queries with group-by is to perform the joins first and then the group-by.

- To produce cheaper physical plans the optimizer should consider doing the group-by before the join.

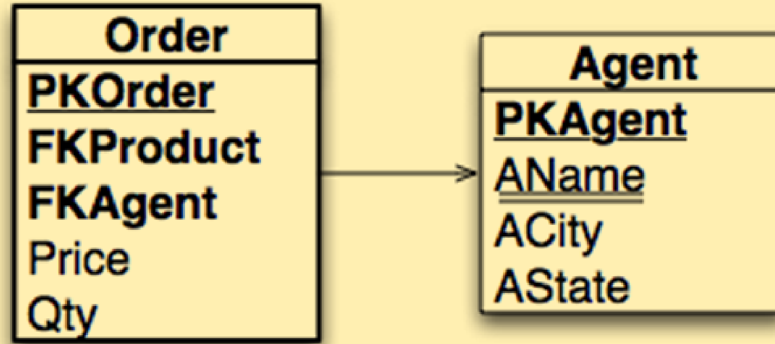
When the group-by can be pushed below the join on R ?

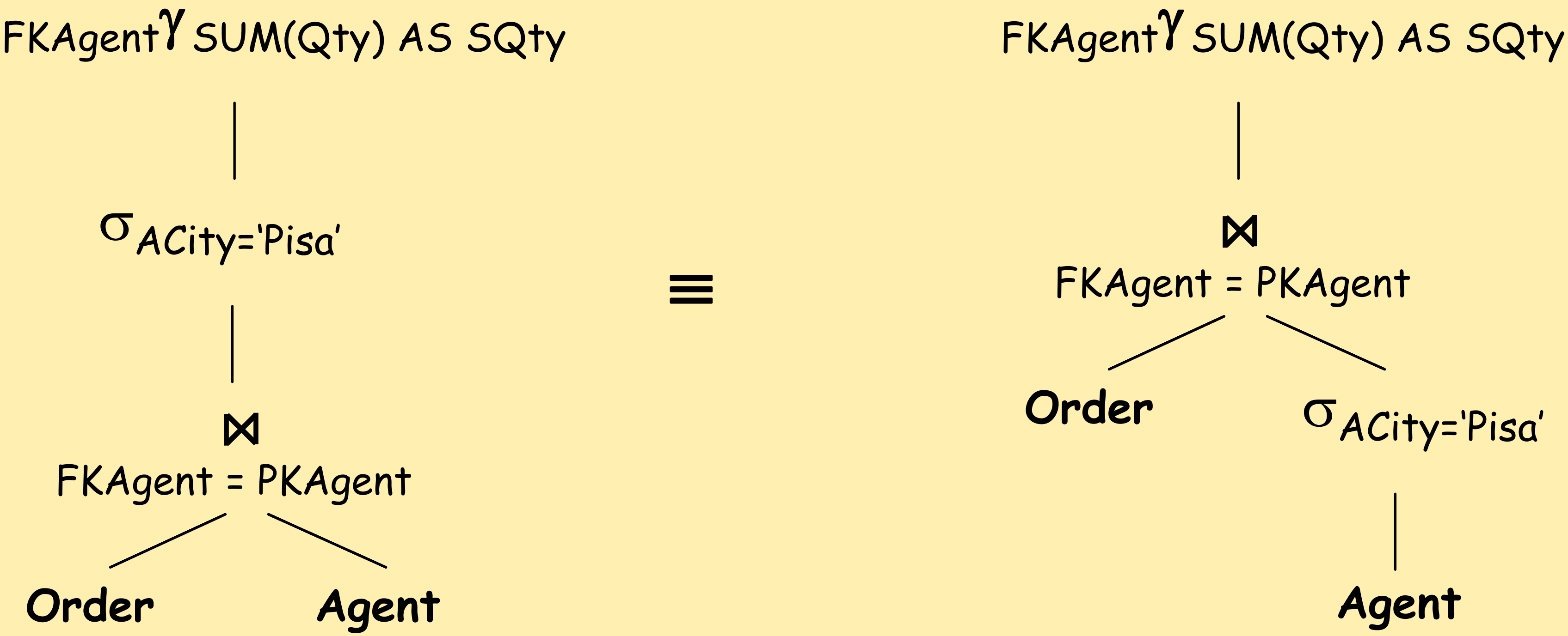
$$_X\gamma_F(R \underset{f_k=p_k}{\bowtie} S) \stackrel{?}{\equiv} \ldots((_{X'}\gamma_{F'}(R)) \underset{f_k=p_k}{\bowtie} S)$$

It is possible in 3 cases ...

FIRST CASE: EXAMPLE

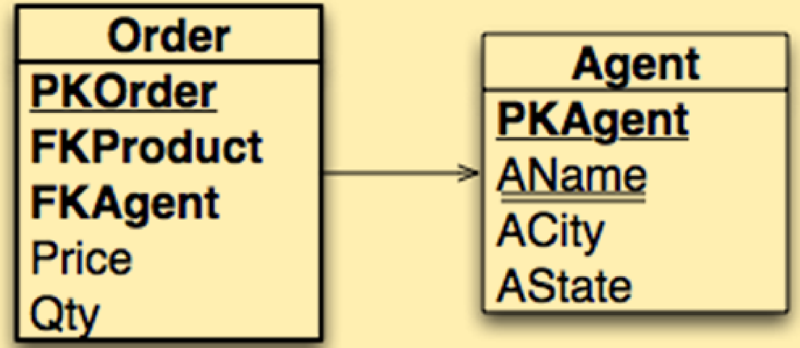
```
SELECT     FKAgent, SUM(Qty) AS SQ
FROM       Order, Agent
WHERE      FKAgent = PKAgent AND ACity = 'Pisa'
GROUP BY   FKAgent;
```

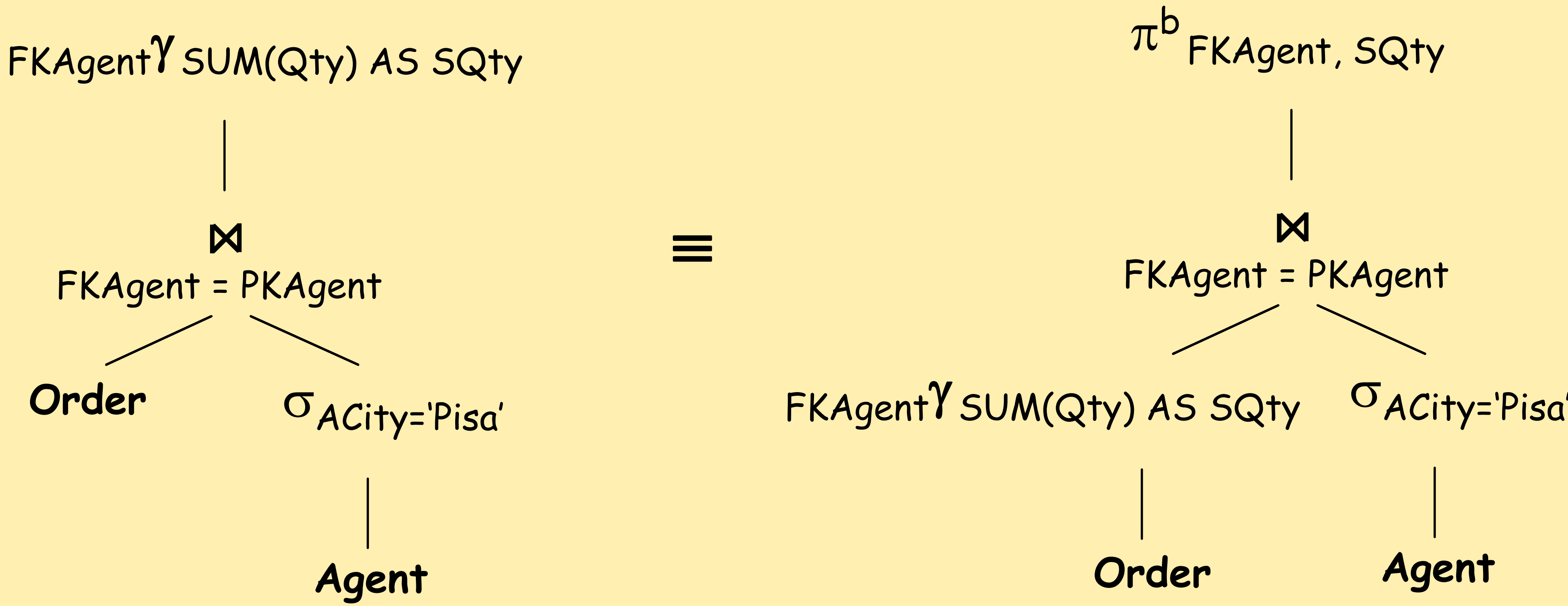




FIRST CASE: EXAMPLE

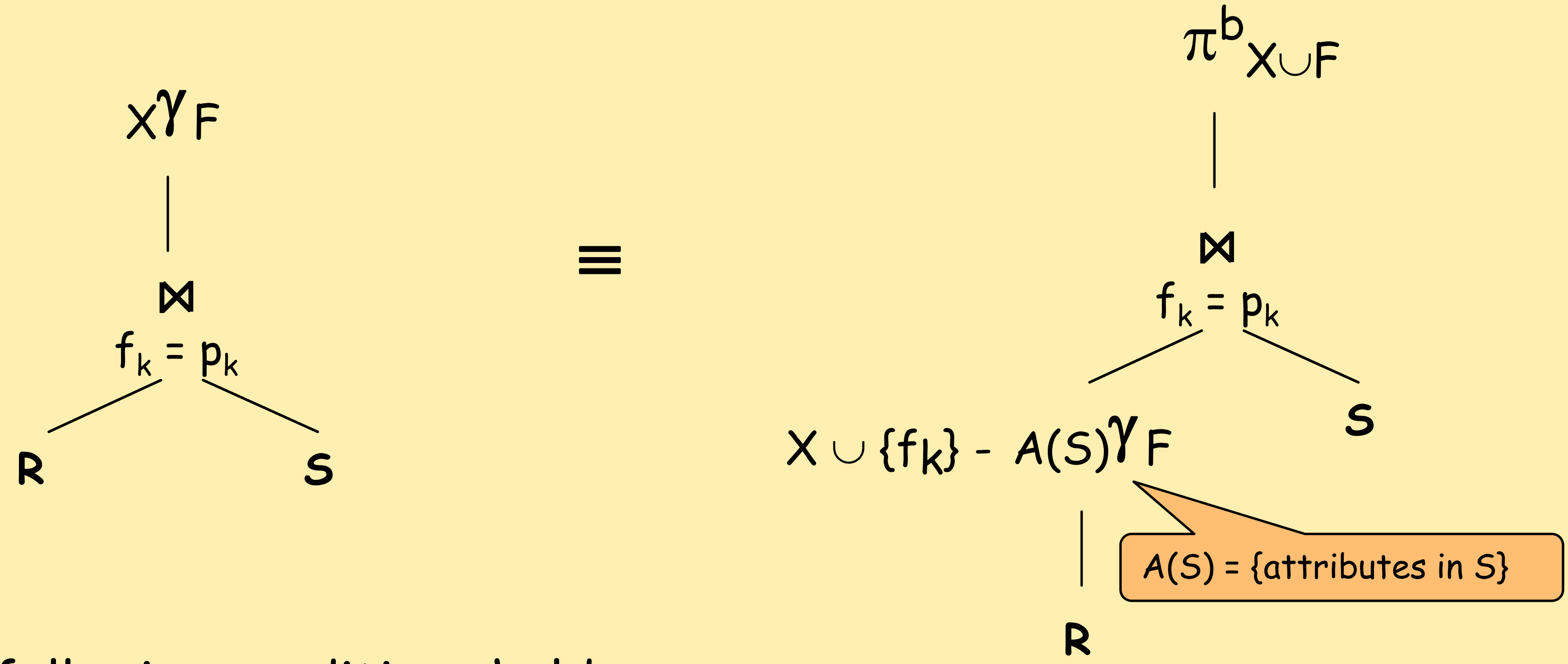
```
SELECT    FKAgent, SUM(Qty) AS SQ
FROM      Order, Agent
WHERE     FKAgent = PKAgent AND ACity = 'Pisa'
GROUP BY  FKAgent;
```





FIRST CASE: THE INVARIANT GROUPING RULE

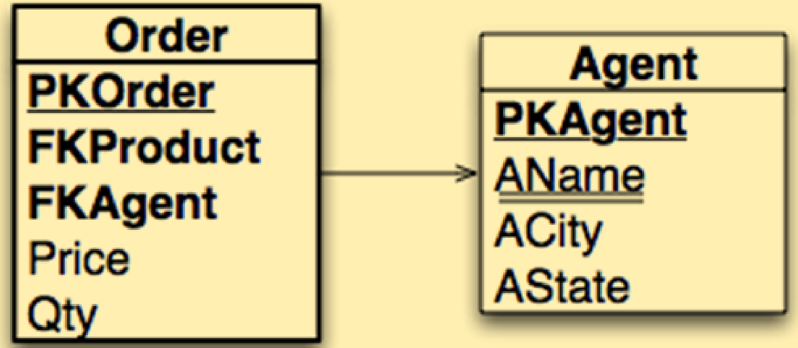
Proposition 1. R has the **invariant grouping** property



if the following conditions hold:

1. $X \rightarrow f_k$ the foreign key of R is determined by X in $R \bowtie_{f_k = p_k} S$
2. Each aggregate function in F uses only attributes from R.

EXAMPLES

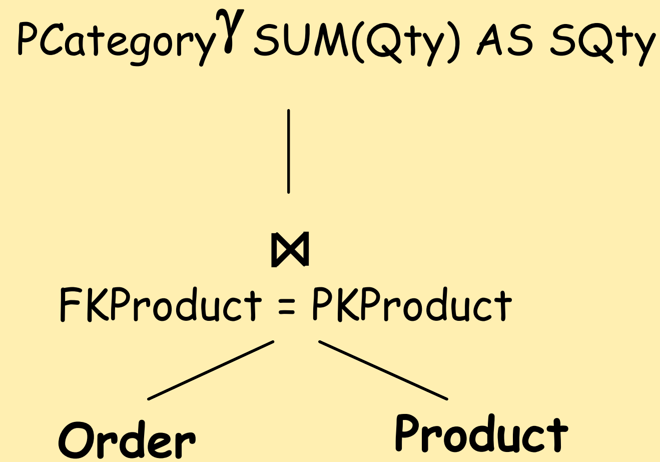



```
SELECT    PKAgent, ACity, SUM(Qty) AS SQ
FROM      Order, Agent
WHERE     FKAgent = PKAgent
GROUP BY  PKAgent, ACity;
```

```
SELECT    AName, SUM(Qty) AS SQ
FROM      Order, Agent
WHERE     FKAgent = PKAgent AND ACity = 'Pisa'
GROUP BY  AName;
```

EXAMPLE NOT WORKING


```
SELECT      PCategory, SUM(Qty) AS SQ
FROM        Order, Product
WHERE       FKProduct = PKProduct
GROUP BY    PCategory;
```

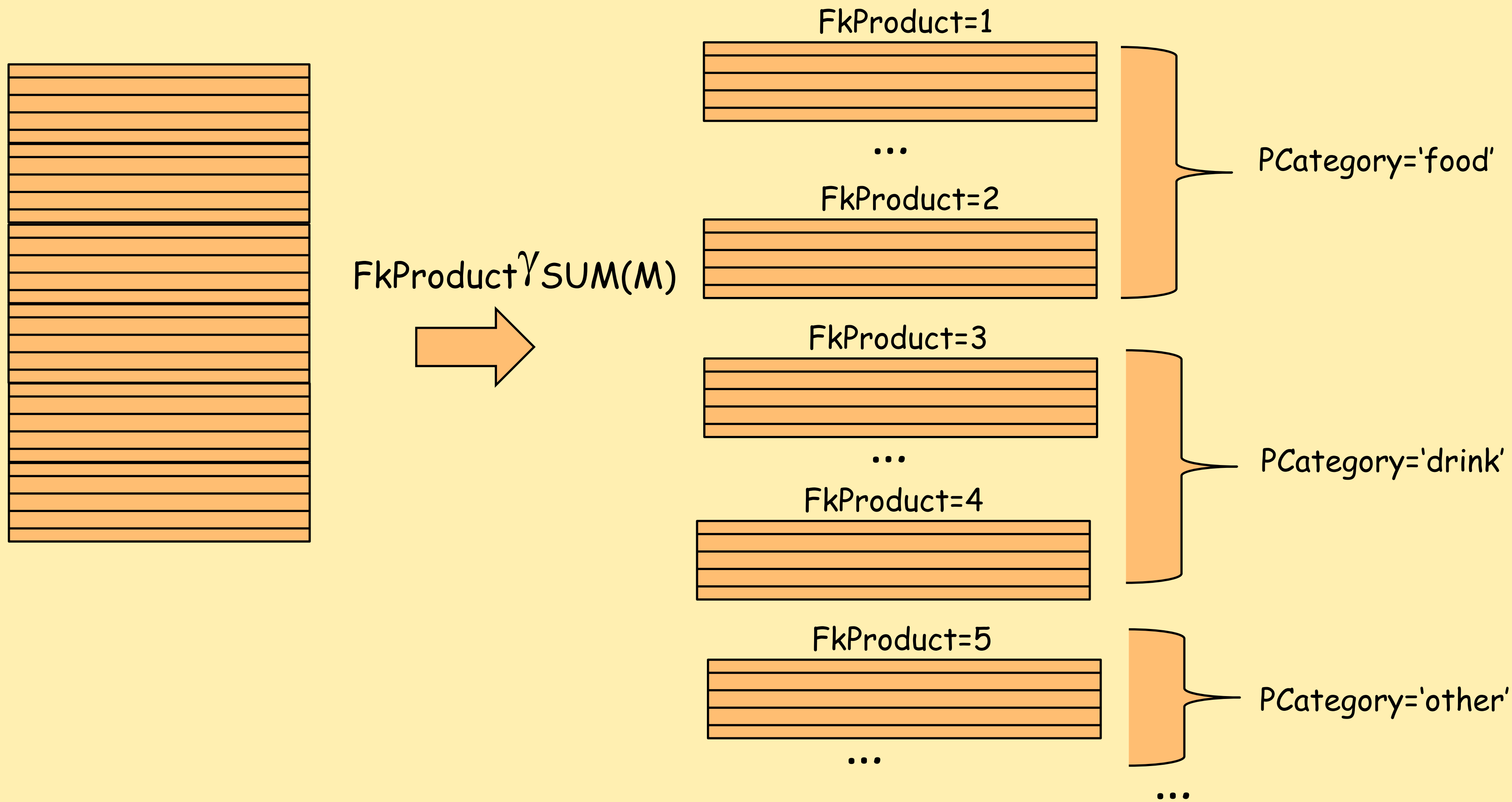


NO PRE-GROUPING WITH THE INVARIANT GROUPING RULE

Condition 1 is false, Condition 2 is true

FD AND GROUPINGS

FkProduct → PCategory
implies
groups by **FkProduct** are included in the groups by **PCategory**



PCategory γ SUM(M) ?

PCategory γ SUM(SM) (**FkProduct γ SUM(M) As SM**)

QUERY REWRITE OPT.: DECOMPOSABLE AGGREGATE FUNCTIONS

An aggregate function f is called **decomposable** if there is a local aggregate function $\mathbf{f_l}$ and a global aggregate function $\mathbf{f_g}$, such that for each multiset V and for any partition of it $\{V_1,V_2\}$ we have

$$f(V_1 \cup^{all} V_2) = f_g(\{f_l(V_1), f_l(V_2)\})$$

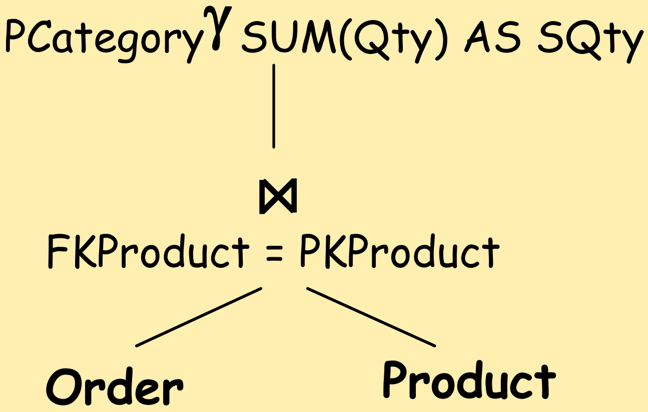
For example **MIN**, **MAX**, **SUM** and **COUNT** are **decomposable**.

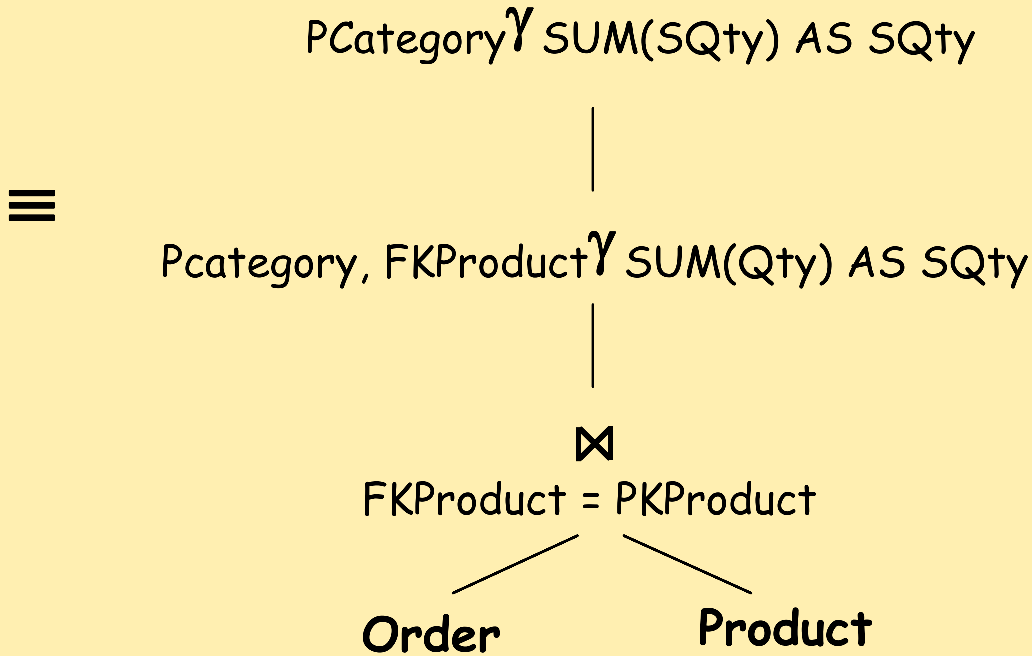
- MIN(V1 U V2) = MIN({MIN(V1), MIN(V2)})
- MAX(V1 U V2) = MAX({MAX(V1), MAX(V2)})
- SUM(V1 U V2) = SUM({SUM(V1), SUM(V2)})
- COUNT(V1 U V2) = SUM({COUNT(V1), COUNT(V2)})

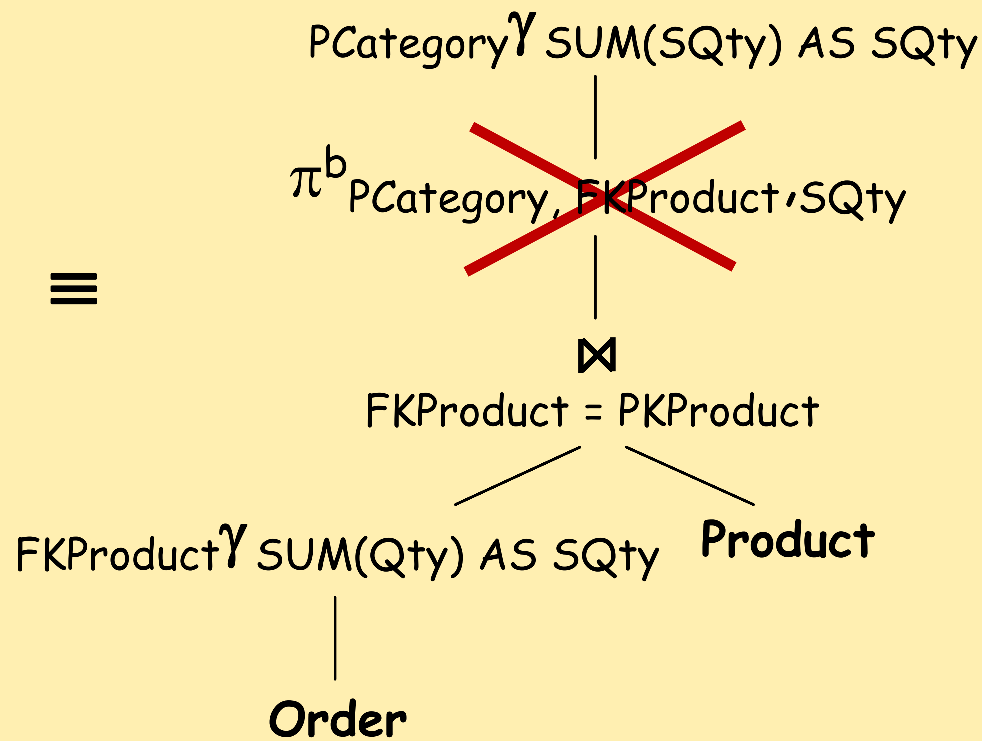
And ***AVG*** ?

AVG(V1 U V2) = SUM({SUM(V1), SUM(V2)}) / SUM({COUNT(V1), COUNT(V2)})

SECOND CASE: EXAMPLE







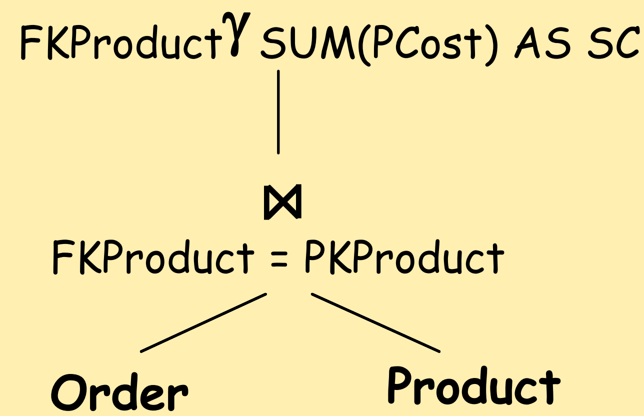
SECOND CASE: DOUBLE GROUPING

Definition. In $_X\gamma_F(R \underset{C_j}{\bowtie} S)$ R has the **early partial aggregation** property if all the aggregate functions are **decomposable** and they use attributes of R.

> **Proposition 1.** If R does not have the **invariant grouping** property because **Condition 1** does not hold, but it has the **early partial aggregation** property, then:
>
> $$_X\gamma_F(R \underset{f_k=p_k}{\bowtie} S) \equiv {}_X\gamma_{F_g}((_{X \cup \{f_k\} - A(S)}\gamma_{F_l}(R)) \underset{f_k=p_k}{\bowtie} S)$$

EXAMPLE NOT WORKING


```
SELECT    FKProduct, SUM(PCost) AS SC
FROM      Order, Product
WHERE     FKProduct = PKProduct
GROUP BY  FKProduct;
```

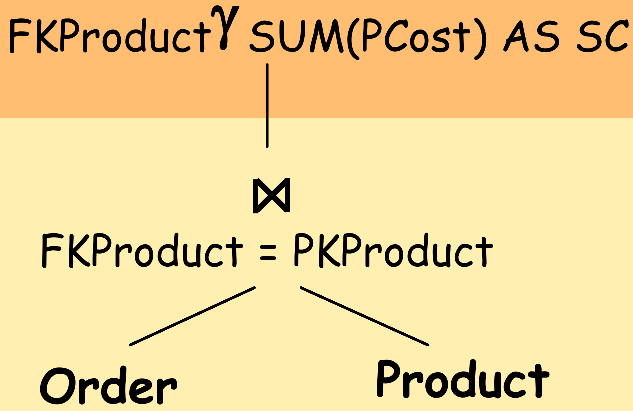


NO PRE-GROUPING WITH THE RULES

INVARIANT GROUPING (Condition 1 is true, Condition 2 is false)

AND EARLY PARTIAL AGG. (Condition 1 is true, Condition 2 is false)

ATTENTION



FKProduct ⟶ PCost

...	FKProduct	...	PKProduct	PName	PCost	...
...	1	...	1	P1	100	...
...	1	...	1	P1	100	...
...	2	...	2	P2	200	...
...	2	...	2	P2	200	...

Grouping on FKProduct: all the records of a group have the same value of PCost

AGGREGATION FUNCTIONS OF REPEATED VALUES

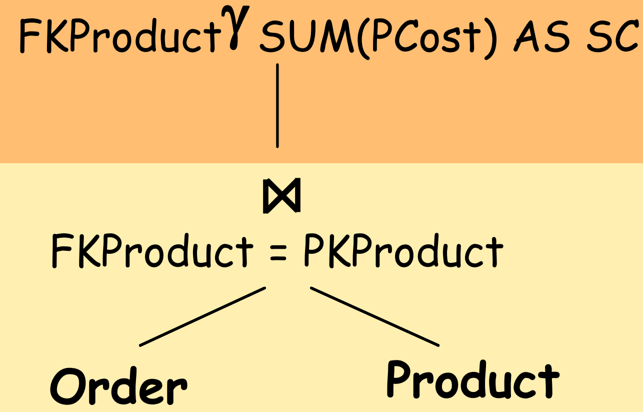
SUM(T) applied to a bag of **repeated values** (**T** = {v, v, ..., v}) with **Tcount** elements have the following property:

$$SUM(T) = v \times \mathbf{Tcount}$$

$$\left.\begin{array}{l} MIN(T) \\ MAX(T) \\ AVG(T) \end{array}\right\} = v$$

$$COUNT(T) = \mathbf{Tcount}$$

ANOTHER EQUIVALENCE RULE

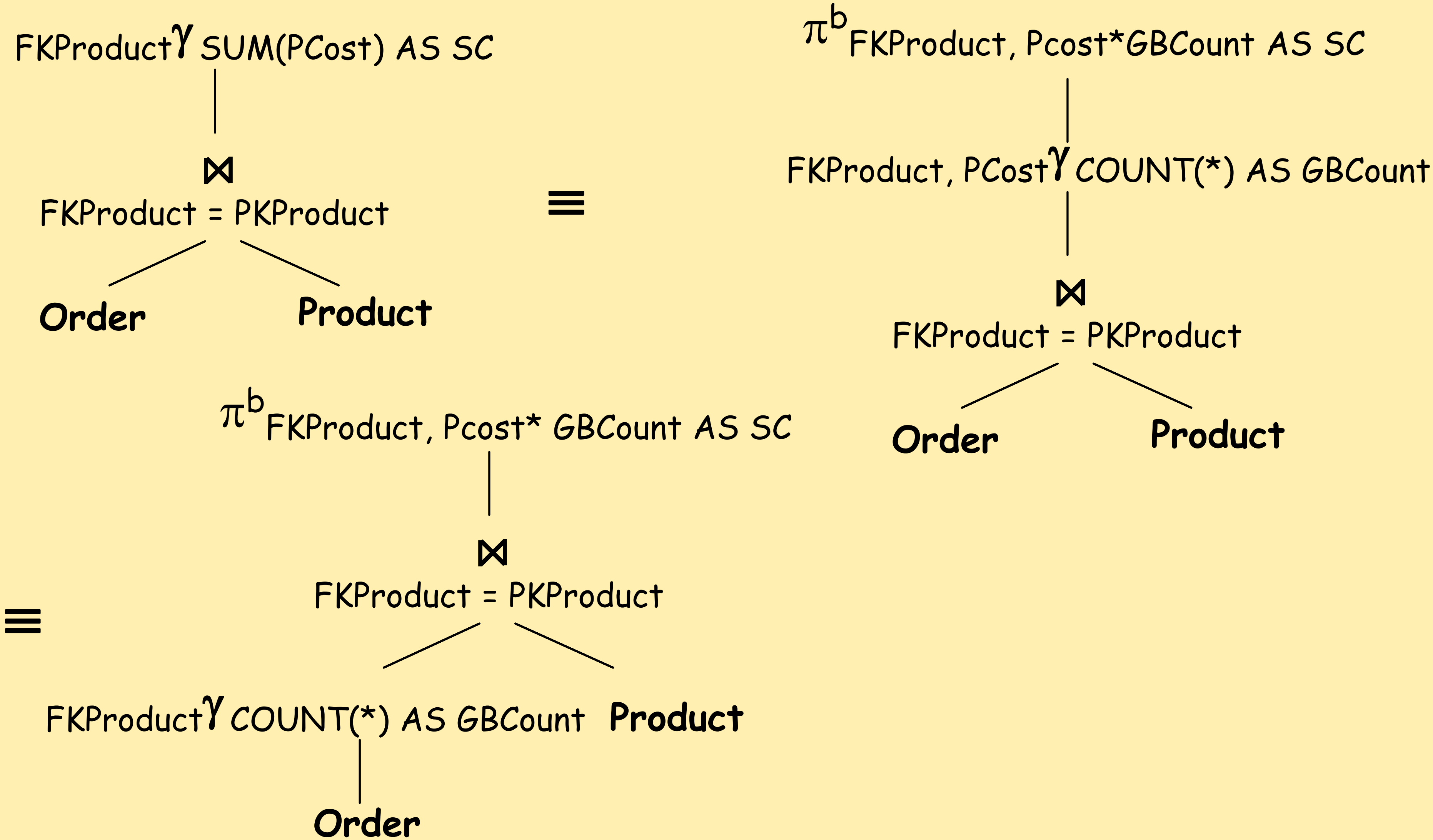


Since grouping on FKProduct the records of a group have the same value for PCost, we can compute the aggregation function SUM in another way

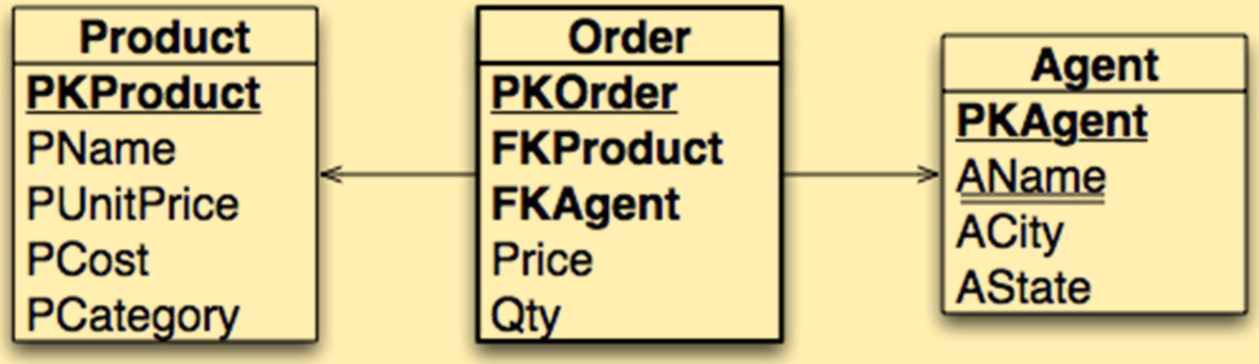
Let $B \notin X$, and $X \rightarrow B$, and $F = \text{SUM}(B)$

$$
{}_{X}\gamma_{\text{SUM}(B)\text{ AS SB}}(E) \equiv \pi^{b}_{X\cup\{B\times\text{GBcount AS SB}\}}\left({}_{X\cup\{B\}}\gamma_{\text{COUNT(*) AS GBcount}}(E)\right)
$$

THIRD CASE: THE GROUPING AND COUNTING RULE



EXERCISE




```
SELECT     PKProduct, (SUM(Price) - SUM(PCost)) AS M
FROM       Order, Product
WHERE      FKProduct = PKProduct
GROUP BY   PKProduct;
```