Master Universitario en Inteligencia Artificial Aplicada

Bases de Datos e Infraestructuras

uc3m | Universidad Carlos III de Madrid

Estructura del Curso

0.- Introducción

1.- Diseño de BB.DD. Relacionales (estática)

2.- Operación de BB.DD. Relacionales (dinámica básica)

3.- SQL para consultas analíticas

4.- Otros elementos de las BBDD Relacionales

Tema 3: Consultas Analíticas en SQL

Estructura del Tema

1.- La Operativa Analítica

- agregación y agrupación

- tipos de agrupación

2.- Analítica avanzada en **SQL**: ámbito de fila

- partición

- encuadre

Tema 3: Operativa Analítica

¿Cuántos años tienes? Sí, tú.

```
SELECT age
    FROM people
    WHERE id='you';
```

¿Cuántos años tenéis? Al grupo.

```
SELECT AVG(age)
    FROM people;
```

AGGREGATE

Tema 3: Consultas Selectivas vs. Analíticas

¡ Te voy a ascender !

```
UPDATE people
   SET category=category+1
   WHERE id='you';
```

¡ Os voy a ascender !

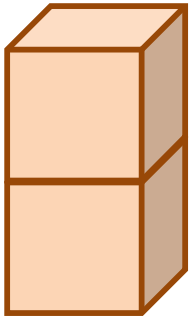
? ? ?

Tema 3: Consultas Selectivas vs. Analíticas

¿ Cuál es vuestra edad?

Er, ¡espera! ¿Cuántos sois?

Agrupaos por color...

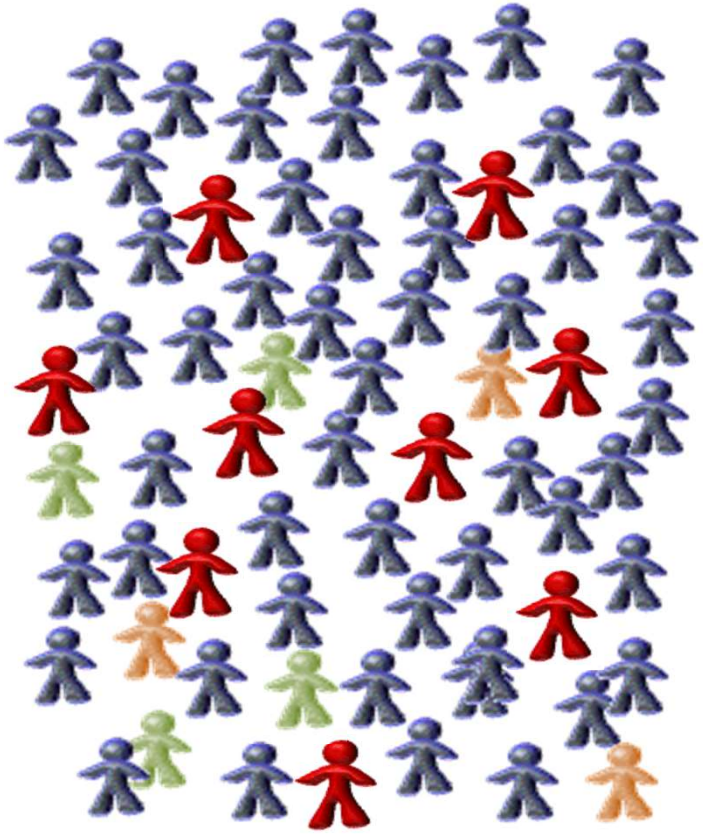


COUNT	AVG(age)
9	47.33
61	27.54

```
SELECT count('X'), AVG(age)
   FROM people
   GROUP BY color;
```

Nota: tras agrupar, los datos originales (indiv.) ***no están*** *en el área de trabajo…*

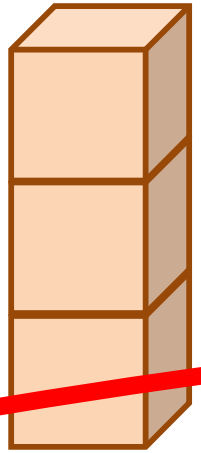
Tema 3.1: Consultas Analíticas I – *Agrupar*



¿ Que edad tenéis los "no naranjas" ?

Agrupaos por color...

Absteneos los grupos con menos de 5…



COUNT	AVG(age)
9	47.33
54	26.85
4	32.25

WHERE

!=

HAVING

```
SELECT count('X'), AVG(age)
   FROM people
   WHERE color!='orange'
   GROUP BY color
   HAVING count('X')>=5;
```

Tema 3.1: Agrupación

Crea una relación nueva según un criterio (subconjunto del esquema). Posibilita la proyección de atributos **agregados**.

Notación: $\pi_{\texttt{proyección}}\ \sigma_{\texttt{selección}}\ \mathcal{G}_{\texttt{criterio}}(\texttt{relación})$

Ejemplo: *¿Cuántos por naciolalidad?* $\pi_{\text{count ('x')}}\ (\mathcal{G}_{\text{Nationality}}\ \textbf{(Person)}\ )$

Person

Name	*Age*	*Nationality*
Fulano	29	Spanish
Mengano	49	Spanish
John Doe	73	English
Smith	14	English
Zutano	3	Spanish
Pelancejo	25	Spanish

→

Nationality	*Tally*
Spanish	4
English	2

```
SELECT Nationality, count('x') tally
   FROM Person
   GROUP BY Nationality;
```

Tras agrupar existirá una fila por cada valor diferente (del criterio) → en esa fila, en cada columna solo puede haber un valor…

Tema 3.1: Agregación

- `MIN(col)`
- `MAX(col)`
- `SUM(col)`
- `AVG(col)`

- `VAR_POP(col)`
- `VAR_SAMP(col)`
- `STDDEV_POP(col)`
- `STDDEV_SAMP(col)`
- `COVAR_POP(col1,col2)`
- `COVAR_SAMP(col1,col2)`

- `MEDIAN(col)`
- `PERCENTILE_CONT([0..1]) WITHIN GROUP (ORDER BY col)`
- `PERCENTILE_DISC([0..1]) WITHIN GROUP (ORDER BY col)`

- `LISTAGG(column [,separator]) WITHIN GROUP (ORDER BY col)`

Tema 3.1: Agrupación en SQL

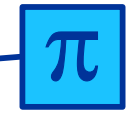
- La cláusula **GROUP BY** define el *criterio de agrupación*
- El área de trabajo agrupada sólo tiene, a priori, las columnas incluidas en el criterio de agrupación.
- Se le pueden añadir columnas aplicando funciones de agregación sobre las columnas excluidas del criterio de agrupación.
- Existen diversas funciones de agregación, como:
  COUNT, AVG, SUM, MIN, MAX, CONCAT (listagg), …

  MEDIAN, VARIANCE, STDDEV, CORR, COVAR, etc.

- **Condición Individual**: la clausula WHERE se ejecuta antes de agrupar;
- **Condición Colectiva**: si se desea realizar una selección del área de trabajo ya agrupada, se usa la clausula HAVING

Tema 3.1: La instrucción SELECT

```
[WITH
      <símbolo> AS <subquery>
      [, <símbolo> AS <subquery> … ] ]

SELECT [ALL|DISTINCT] <lista de selección>

      FROM <cláusula de origen>

      [WHERE <condición> ]

      [GROUP BY <expresión> [HAVING <condcn>]]

      [{UNION|UNION ALL|MINUS|INTERSECT} <query>]

      [ORDER BY <expresión> [ASC|DESC]] ;
```



Tema 3.1: Agrupación y Orden: Ejemplos

Curso	Nombre	Apellido	ptos	Edad
2º	Fulano	Pérez	113	32
1º	Mengana	Gómez	47	19
1º	Zutano	Smith	89	44
2º	Peranceja	García	45	10
1º	John	Doe	67	78

```
SELECT curso, count('x'),
       avg(edad), sum(ptos)
       FROM Gente
       GROUP BY curso;
```

Curso	count	AVG(edad)	SUM(ptos)
2º	2	21	158
1º	3	47	203

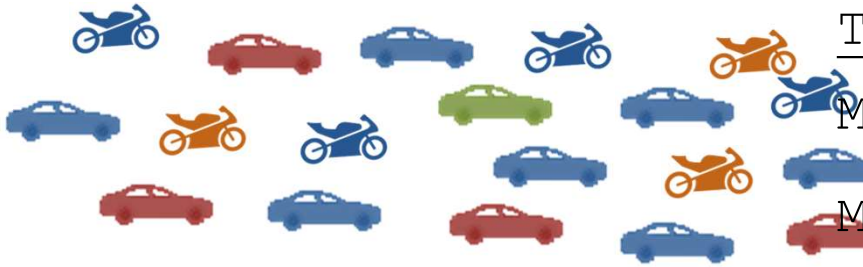
```
SELECT * FROM Gente ORDER BY apellido;
```

Curso	Nombre	Apellido	DNI	Edad
1º	John	Doe	6789	78
2º	Peranceja	García	2345	10
1º	Mengana	Gómez	4567	19
2º	Fulano	Pérez	0123	32
1º	Zutano	Smith	8901	45

Tema 3.2: Agrupación Extendida: ROLL UP

ROLL-UP: unión de agrupaciones por todos los **prefijos** del criterio

Ejemplo: ¿Cuántos vehículos? ¿De cada tipo? ¿Y según el tipo y color?



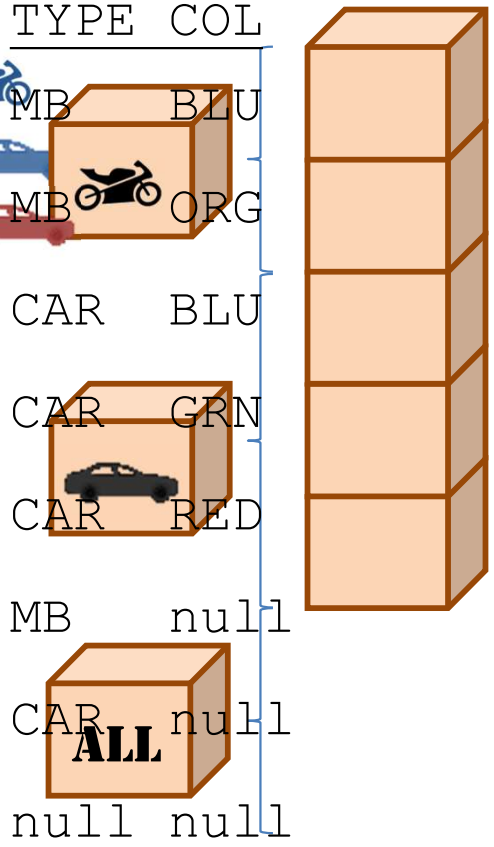
```
SELECT count('X'), SUM(km)
    FROM vehicles
    GROUP BY ROLLUP(type,color);
```

Ģ ROLLUP (A,B) ≡

Ģ(A,B)

∪ Ģ(A)

∪ Ģ(∅)



COUNT	AVG(KM)
4	57171
3	67413
7	92828
1	154944
4	99718
7	61560
12	100301
19	86028

Tema 3.2: Agrupación Extendida: CUBE & G.S.

Agrupación CUBE es la unión de agrupaciones según todas las **combinaciones**

Ģ CUBE(A,B) ≡ Ģ(P(A,B)) ≡ Ģ(A,B) ∪ Ģ(A) ∪ Ģ(B) ∪ Ģ(∅)

```
SELECT count('X'), SUM(km)
   FROM vehicles
   GROUP BY CUBE (type,color);
```

GROUPING SETS permite elegir qué combinaciones escoger.

Ģ GROUPING SETS ((A,B),(B)) ≡ Ģ(A,B) ∪ Ģ(B)

```
SELECT count('X'), SUM(km)
   FROM vehicles
   GROUP BY GROUPING SETS ( (type,color), (type), (color) );
```

- En las columnas del criterio **Ģ**, los atributos ausentes adoptan el valor nulo. La función `GROUPING(col)` devuelve **1** si *col* está ausente (**0** en otro caso)

```
SELECT DECODE(grouping(type),1,'all types', type),
       DECODE(grouping(color),1,'all colors', color),
       count('X'), SUM(km)
   FROM vehicles GROUP BY CUBE (type,color);
```

Tema 3.2: Agrupación Extendida: ejemplo

Resuelve la siguiente consulta en la BD de vinilos:

- Películas visualizadas mensualmente, tanto en global como detalladas por género.

```
With sales as
      (select isvn, to_char(order_s,'YYYY/MM') month, qtty
          from sale_line )
select month, decode(grouping(format),1,'GLOBAL', format) F,
       sum(qtty) TOTALS
   from sales join discs using(isvn)
   group by rollup (month,format)
   order by month;
```

- ¿Quieres las visualizaciones anuales o prefieres omitir ese dato?
- ¿Cuál sería la diferencia ejecutándolo de otro modo? (uniendo consultas)

Tema 3.3: Más Agregación

- `COUNT ('X')`
  - `COUNT (DISTINCT col [, col…])`
  - `APPROX_COUNT_DISTINCT(col)`

- `WM_CONCAT(col)` `/*almost deprecated */`
- `LIST_AGG(col [, sep]) WITHIN GROUP (ORDER BY col[,col]…)`

Limita la agregación a un valor concreto (ya sea FIRST o LAST según cierto orden)

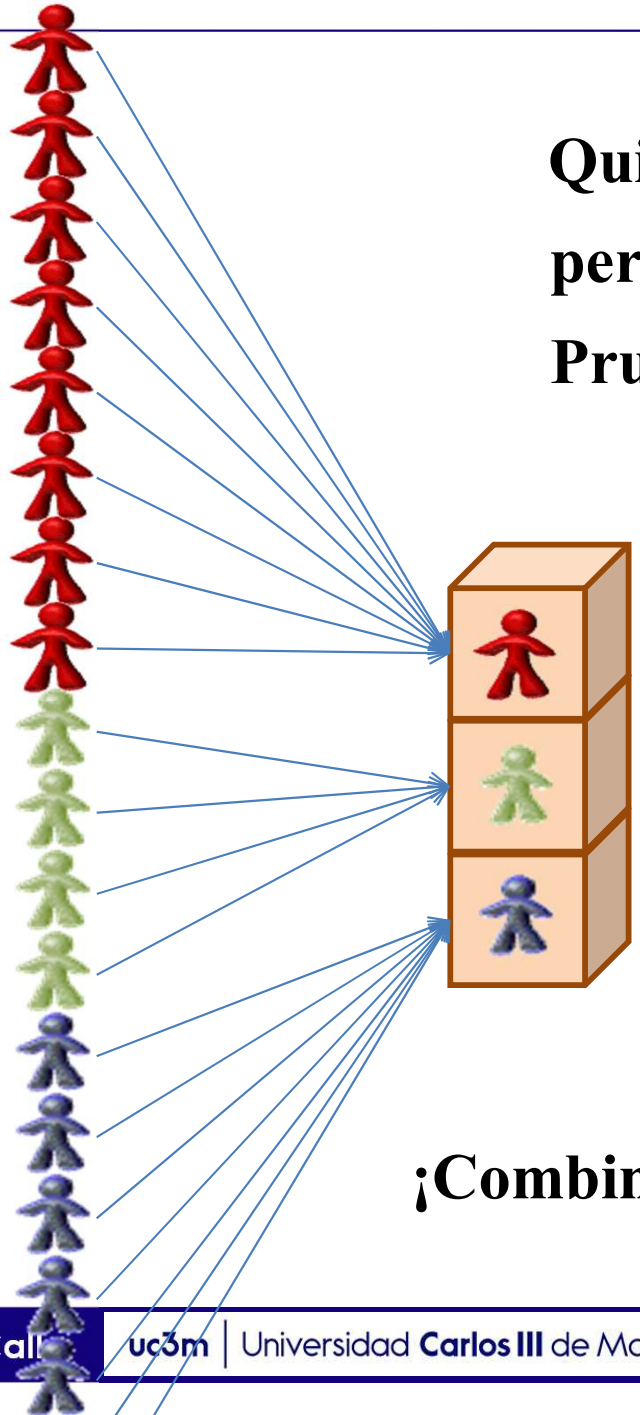
- *`function`*`(col1) KEEP (DENSE_RANK FIRST ORDER BY col2)`

- `RANK(value) WITHIN GROUP (ORDER BY col [,col…])`
- `DENSE_RANK(value) WITHIN GROUP (ORDER BY col [,col…])`
- `PERCENT_RANK(value) WITHIN GROUP (ORDER BY col [,col…])`

  `* 'value' es un valor constante tomado como hipotéticamente insertado`

…y hay más (incluso def por el usuario); puedes consultar la documentación en:
https://docs.oracle.com/database/121/SQLRF/functions003.htm#SQLRF20035

Tema 3.4: Funciones Analíticas



Quieres recuperar CADA FILA

pero incluyendo agregación de columnas?

Prueba las posibilidades analíticas…

COUNT	AVG(age)
8	46.25
4	32.25
54	26.85

¡Combina características individuales y de grupo!

Tema 3.4: Func. Analíticas: la Partición

```
function ()                    /* cualquier agregación */
  OVER ( [PARTITION BY criteria1]
         [ORDER BY criteria2 [windowing_clause] ] )
```

- Partición no transforma el área de trabajo (tampoco el 'order by')

  → es una '***consulta individual***' que añade agregación (no es una *consulta colectiva*)

- Por otro lado, utiliza áreas de trabajo auxiliares:

  OVER define área de trabajo auxiliar para la fila;

  PARTITION by *key*, restringe su contenido al valor de key en la fila

  - puede limitarse a una Ventana, si se establece un orden

Tema 3.4: Partición vs. Agrupación

```
SELECT department, AVG(salary) "mean"
    FROM employees
    GROUP BY department;
```

DEPARTMENT	MEAN
Marketing	1497.67
Sales	3464.98
H.R.	1675.50
…	…

```
SELECT employee_id, salary, department,
    AVG(salary) OVER (PARTITION BY department) "Mean"
    FROM employees;
```

EMPLOYEE_ID	SALARY	DEPARTMENT	MEAN
324/OT5-22	1268.00	Marketing	1497.67
052/OT5-19	1795.75	Marketing	1497.67
566/TR7-13	4115.00	Sales	3464.98
…	…	…	…

Tema 3.4: Partición vs. Agrupación

- Si no se definen ni agrupación ni partición, se agrega toda la tabla

```
SELECT count('x') staff, AVG(salary) mean
    FROM employees;
```

STAFF	MEAN
1452	1683.13

- Para combinar cada fila con la tabla completa, se puede añadir una 'partición nula' (con una clausula vacía o con *partition by null*).

```
SELECT employee_id, salary, AVG(salary) OVER () mean
   FROM employees;
```

- O se puede hacer un producto cartesiano…

```
SELECT employee_id, salary, mean
   FROM employees CROSS JOIN
    (SELECT AVG(salary) mean
        FROM employees);
```

employee_ID	salary	MEAN
324/OT5-22	1268.00	1683.13
052/OT5-19	1795.75	1683.13
566/TR7-13	4115.00	1683.13
…	…	…

¡Ambas dan el mismo resultado!

Tema 3.4: Partición vs. Agrupación

- Algunas agregaciones dependen del criterio de ordenación

```
SELECT department, listagg(name,', ')
       within group (ORDER BY salary) staff_names
  FROM employees
  GROUP BY department;
```

DEPARTMENT	STAFF_NAMES
Marketing	John, Erika, Peter, Jack,…
Sales,	Mary, Paul, Mary, Cathy, …
H.R.	Bertha, Angus, Gregory, …
…	…

- *Order* es obliatorio, pero puedes usar *"order by null"* (*order by rownum*) para recuperar las filas en su *orden natural* (según están almacenadas)

```
SELECT department, listagg(name,', ')
       within group (ORDER BY null) staff_names
  FROM employees
  GROUP BY department;
```

DEPARTMENT	STAFF_NAMES
Marketing	Peter, Erika, John, Jack, …
…	…

Tema 3.4: Funciones Analíticas - ejemplos

Ejemplo: de una tabla de películas, extrae títulos, país, presupuesto, ingresos, y media de su país en presupuesto, para aquellas películas cuyo presupuesto es superior a la media y cuyos ingresos son inferiores al presupuesto.

```
select *
   from (select movie_title, country, budget, gross,
          avg(budget) over (partition by country) mean_budget
         from movies)
   where budget is not null and gross is not null
         and budget>mean_budget
         and budget>gross;
```

Tema 3.5: Partición y Encuadre (windowing)

- Algunas funciones analíticas permiten definir un subconjunto de filas de la partición sobre los que operar (encuadre o ventana).
- Para ello, es imprescindible ordenar la partición
  (incluyendo una clausula 'order by')
- El encuadre puede ser: un rango de **filas** (`ROWS`) o de **valores** (`RANGE`)
- Por defecto (con cláusula *order by* y sin cláusula de encuadre), la ventana va desde la primera fila a la actual.

```
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
```

- Si se omite la ordenación (sin cláusula *order by*), se toman todas las filas, lo que sería equivalente a aplicar un orden y un encuadre total:

```
ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
```

Tema 3.5: Partición y Encuadre (windowing)

- **Ejemplos:**
  - desde la primera a la actual
    ```
    ROWS UNBOUNDED PRECEDING
    ```
  - desde el primer valor al actual
    ```
    RANGE UNBOUNDED PRECEDING
    ```
  - solo el valor actual
    ```
    RANGE CURRENT ROW
    ```
  - fila actual y dos anteriores
    ```
    ROWS 2 PRECEDING
    ```
  - del valor anterior al posterior
    ```
    RANGE BETWEEN 1 PRECEDING AND 1 FOLLOWING
    ```
  - “el mismo día de la semana pasada"
    ```
    RANGE BETWEEN 7 PRECEDING AND 7 PRECEDING
    ```
  - desde la fila actual hasta el final
    ```
    ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING
    ```

Tema 3.5: Partición y Encuadre

TABLE recent_sales:

P_TYPE	S_DATE	TOTAL
…	…	…
bakery	20/5/17	436
dairy	15/5/17	192
dairy	15/5/17	97
dairy	16/5/17	136
dairy	17/5/17	128
dairy	17/5/17	183
dairy	18/5/17	158
dairy	19/5/17	163
dairy	20/5/17	214
fish	15/5/17	105
…	…	…

PARTICIÓN

Fila actual

```
SELECT prod_type, SUM(TOTAL) OVER
(PARTITION BY prod_type
 ORDER BY s_date
 <windowing clause> ) mysum
FROM recent_sales;
```

Para definir esta ventana, utiliza…

Solo el primer valor de la partición:
Todo hasta la fila actual
Desde el día antes al día después (si existen)
Todo = omitir claúsula, o:
De ahora en adelante…

Sólo el día anterior (si existe)
Fila actual y las dos anteriores (si existen)
Hoy

```
SELECT prod_type, SUM(TOTAL)
KEEP (DENSE_RANK FIRST
RANGE UNBOUNDED PRECEDING
ROWS 2 PRECEDING
RANGE BETWEEN
ROWS BETWEEN
    1 PRECEDING AND
    UNBOUNDED PRECEDING AND
      1 FOLLOWING
    UNBOUNDED FOLLOWING
ROWS BETWEEN
    CURRENT ROW AND
    UNBOUNDED FOLLOWING
```

Tema 3.6: Más Funciones Analíticas

Existen varias funciones de agregación que cuentan con una función analítica análoga (y homónima), pero también existen funciones analíticas específicas:

- **`NTILE(n) OVER ([partition] order by col)`**
  divide el conjunto ordenado de valores de col en *n* subconjuntos de 'igual' cardinalidad (si la division no es exacta, los primeros tienen un elemento más) y proporciona el ordinal del subconjunto al que pertenece la fila actual.
  ¡Cuidado! Pueden existir filas 'empatadas' en tiles distintos (refina ordenación).

  EJEMPLO:

```
select movie_title, ntile(10) OVER (order by gross) decile
   from movies;
```

- **`RATIO_TO_REPORT(col) OVER ([partition])`**
  proporciona el ratio (porcentaje) de la fila actual respecto a la suma de valores (puedes añadir partición). EJEMPLO:

```
select audience, ratio_to_report(audience) OVER () share
   from tv_programs;
```

Tema 3.6: Más Funciones Analíticas

La pseudocolumna estática **ROWNUM** es muy útil para **limit**ar y/o elegir resultados. Tenemos algo similar (y más estable) entre las funciones analíticas:

- **ROW_NUMBER () OVER (analytic_clause)**
  proporciona un número natural distinto a cada fila dentro de su Ventana

  EJEMPLO:

```
WITH grp AS
     (SELECT movie_title, title_year, gross,
             row_number() OVER (partition by title_year
                                order by gross desc) n
        FROM movies
        WHERE gross is not null)

SELECT title_year, n podium, gross, movie_title
   FROM grp
   WHERE n<=3
   ORDER BY title_year;
```

Tema 3.6: Más Funciones Analíticas

Las funciones analíticas aportan información colectiva, pero también pueden traer información de otra fila (caso particular, donde esa fila representa al colectivo):

- **`LAG(col [,offset [,default]]) OVER ([partition] order by…)`** permite proyectar información de una fila previa (por defecto con offset=1, la fila inmediatamente anterior). Si no existe, adopta *default* o *null*. Requiere establecer ordenación. La coletilla **`IGNORE NULLS`** justo antes de OVER saltará valores nulos.
- **`LEAD:`** análoga a la anterior, pero observando las filas siguientes.
- **`FIRST_VALUE(col) OVER ([partition] order by…)`**
  Como las anteriores, pero toma la primera fila de la ventana (puede saltar nulos)
- **`LAST_VALUE:`** … análogo a la anterior, pero toma la última fila.
- **`NTH_VALUE(col,n) [FROM LAST] OVER ([partition] order by…)`**
  similar a las anteriores, toma la fila enésima de la ventana.

```
select director_name, gross, gross-lag(gross,1,0) OVER
                                (partition by director_name
                                 order by title_year) diff
    from movies
    order by director_name,title_year;
```