

# Basi di dati

Corso di laurea in Ingegneria Informatica  
Scuola di Ingegneria - Università di Pisa

---

**Oracle MySQL**  
A.A. 2016-2017

Ing. Francesco Pistolesi

*Postdoctoral Researcher*  
Dipartimento di Ingegneria dell'Informazione  
[francesco.pistolesi@iet.unipi.it](mailto:francesco.pistolesi@iet.unipi.it)

# Analytics

---



# Analytic functions

---

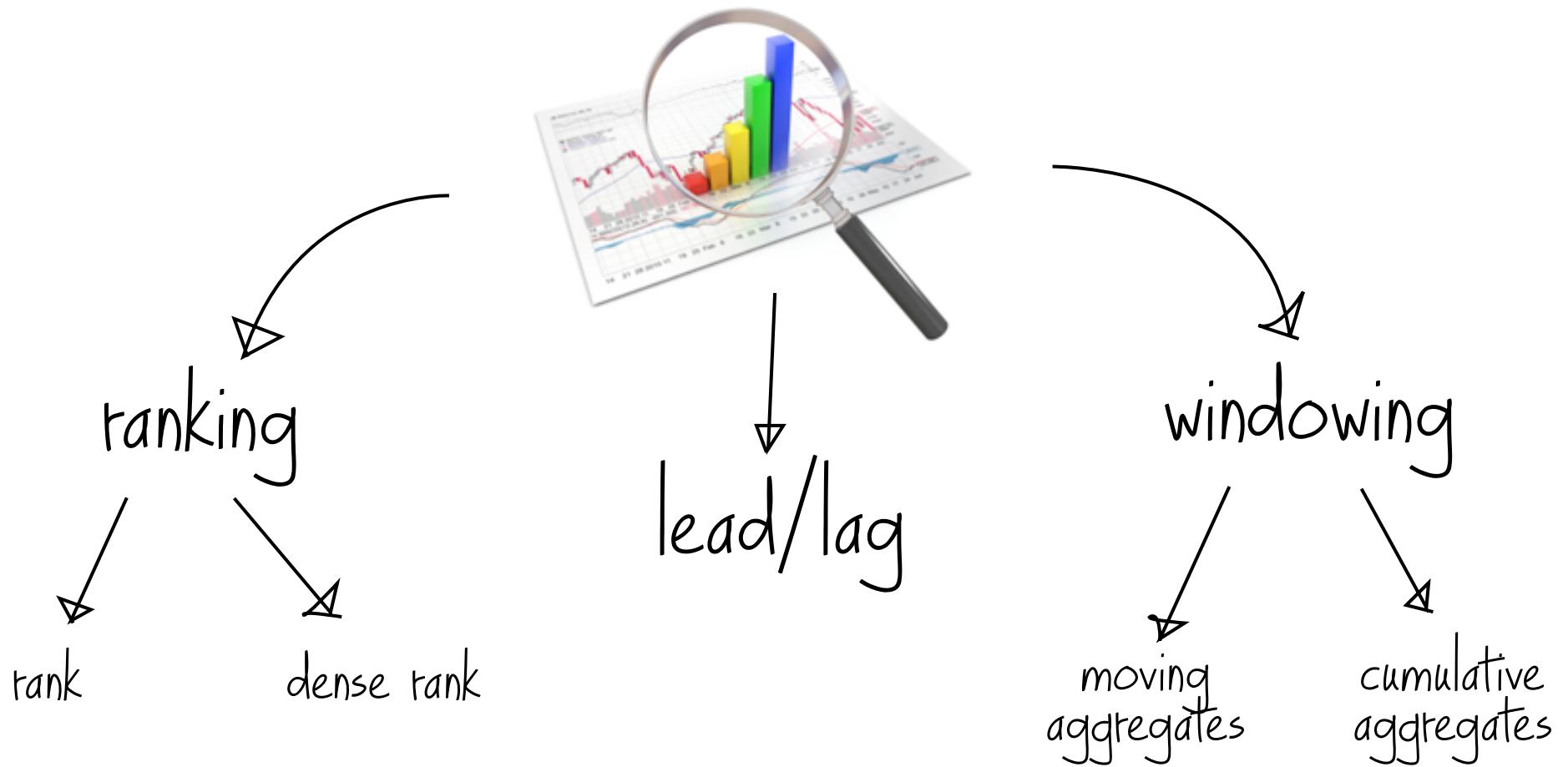
record non aggregato

Affiancano il **dettaglio** a un risultato aggregato ottenuto da operazioni eseguite su un insieme di **record connessi al dettaglio**

operatori di aggregazione, rank, windowing...

# Tipi di analytics

---



# A cosa servono

---

## Ranking

prevalentemente per stilare classifiche

## Lead/Lag

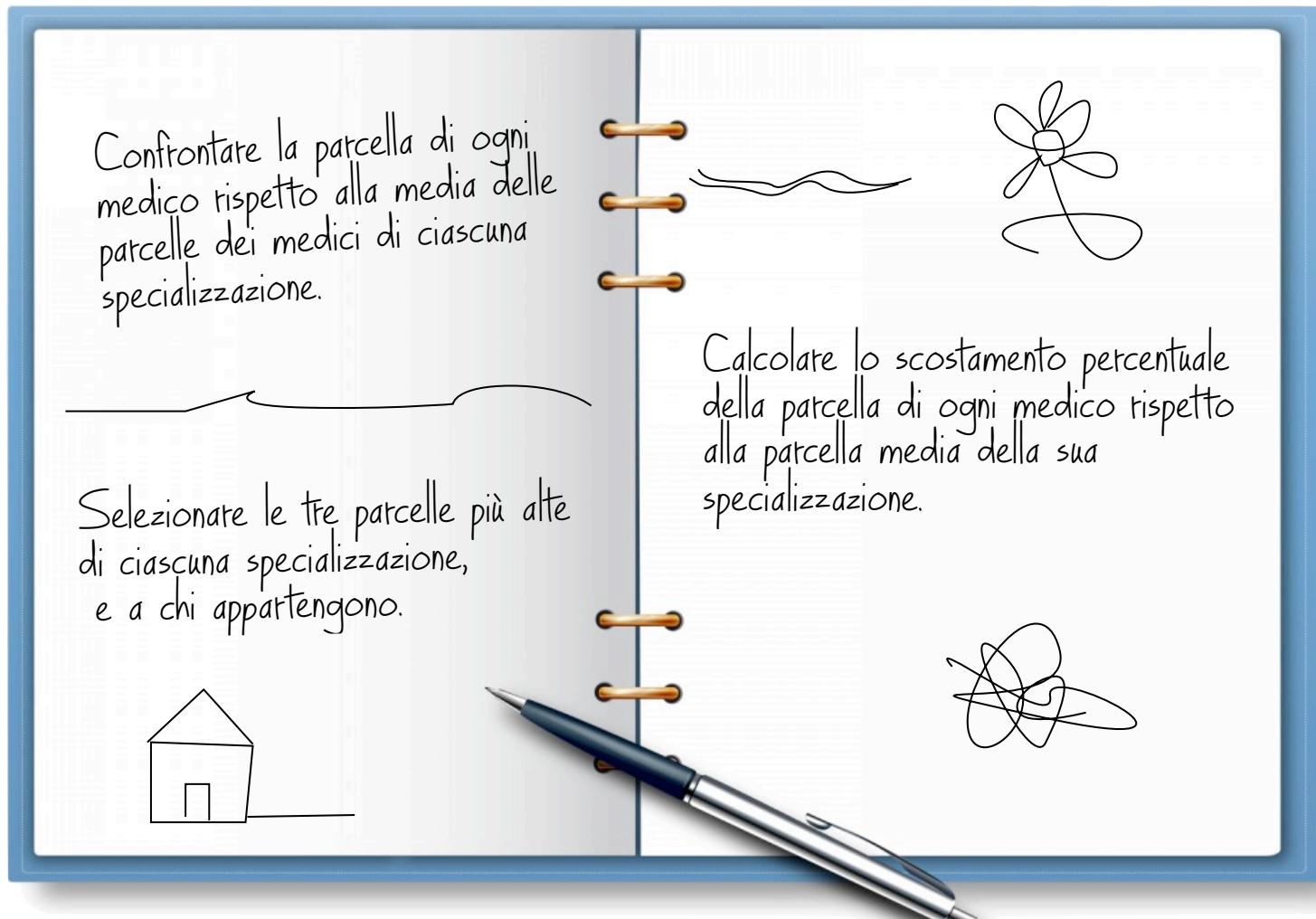
trovate un valore in una row posizionata X row  
prima (lag) o dopo (lead) la current row

## Windowing

calcolate valori aggregati cumulativi o mobili

# Esempi di query analytics

---



# Un primo esempio: aggregazioni spinte

---

Considerate le visite del 2016, indicare il numero di visite effettuate da  
ogni medico e il totale complessivo di tali visite

# Modificatore WITH ROLLUP

Considerate le visite del 2016, indicare il numero di visite effettuate da ogni medico e il totale complessivo di tali visite

```
1 | SELECT V.Medico,  
2 |     COUNT(*)  
3 | FROM Visita V  
4 | WHERE YEAR(V.Data) = 2016  
5 | GROUP BY V.Medico WITH ROLLUP;
```

Medico	COUNT(*)
001	4
002	4
004	5
013	5
014	3
[Null]	21



super-aggregate record

# WITH ROLLUP: attributi multipli

---

Considerate le visite del 2016, indicare il numero di visite effettuate da ogni paziente con ogni medico, il totale delle visite di ogni medico, e il totale complessivo delle visite

# WITH ROLLUP: attributi multipli

```
1 SELECT V.Medico,  
2      V.Paziente,  
3      COUNT(*)  
4 FROM Visita V  
5 WHERE YEAR(V.Data) = 2016  
6 GROUP BY V.Medico,  
7      V.Paziente  
8 WITH ROLLUP;
```

Medico	Paziente	COUNT(*)
001	g ox9	1
001	ttw2	3
001	[Null]	4
002	bbc4	1
002	g ox9	1
002	hho1	2
002	[Null]	4
004	psq9	1
004	slq6	4
004	[Null]	5
013	ffq8	2
013	kkw1	3
013	[Null]	5
014	aaa1	1
014	ddd6	2
014	[Null]	3
[Null]	[Null]	21

# Attenzione

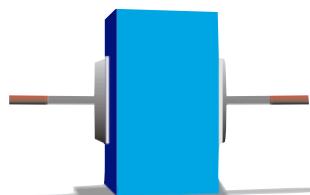
---



# Aggregazione con partition

Per ogni specializzazione, indicarne il nome, la matricola il cognome e la parcella di ogni suo medico, e la parcella media dei medici di quella specializzazione

```
1 SELECT M.Specializzazione, M.Matricola, M.Cognome, M.Parcella, D.Media  
2 FROM Medico M INNER JOIN  
3 ( partition  
4   SELECT M2.Specializzazione, AVG(M2.Parcella) AS Media  
5   FROM Medico M2  
6   GROUP BY M2.Specializzazione  
7 ) AS D  
8   ON M.Specializzazione = D.Specializzazione  
9 ORDER BY M.Specializzazione, M.Parcella;
```



la partition è l'insieme sul quale si aggrega

# Rank su partition (con gap)

---

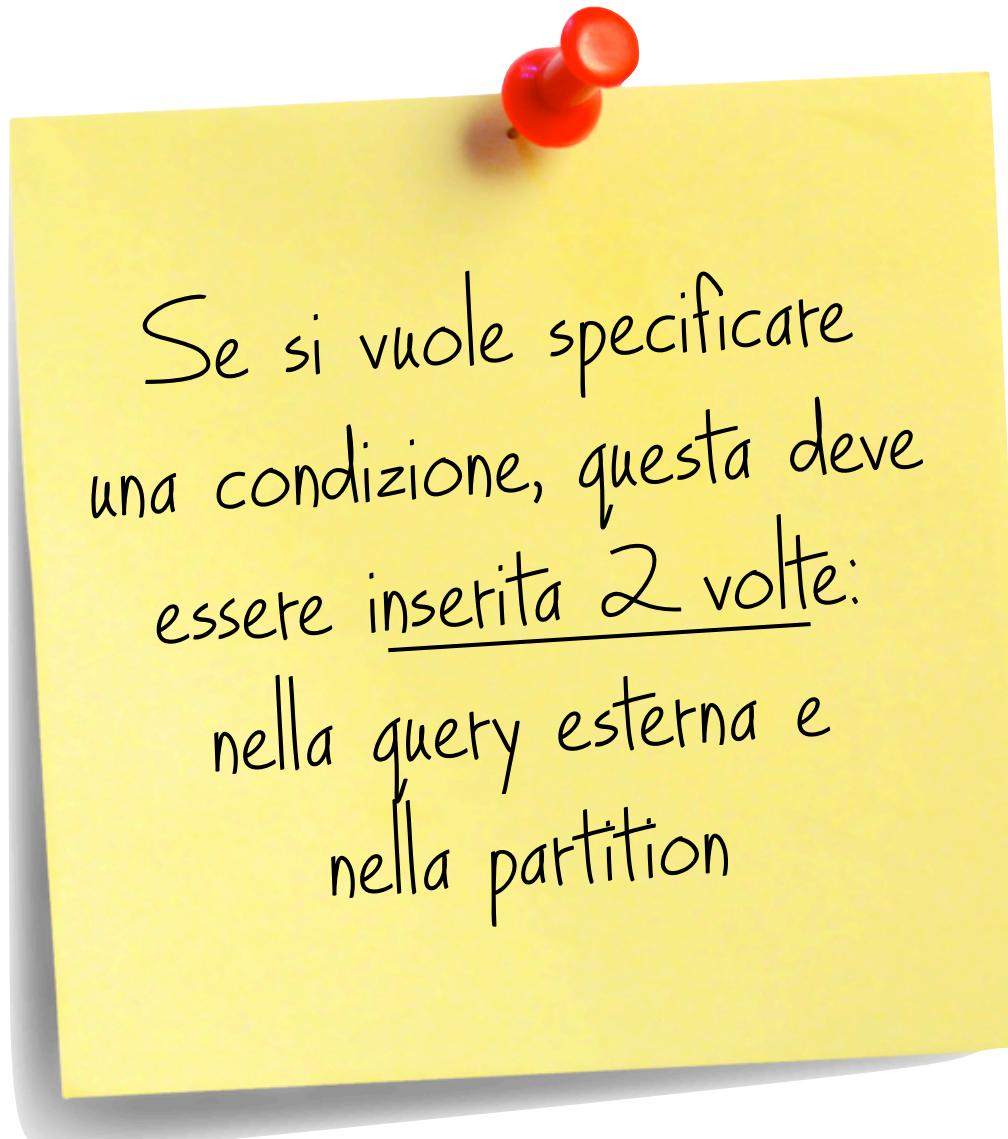
Per ogni specializzazione, restituirne il nome, assieme alla matricola il cognome e la posizione in classifica dei medici aventi le tre parcellle più alte

```
1 SELECT *
2 FROM(
3     SELECT M.Specializzazione, M.Matricola, M.Cognome, M.Parcella,
4         (SELECT 1 + COUNT(*)
5             FROM Medico M0
6             WHERE M0.Specializzazione = M.Specializzazione
7                 AND M0.Parcella > M.Parcella) AS Rank
8     FROM Medico M
9 ) AS D
10 WHERE D.Rank <= 3
11 ORDER BY D.Specializzazione, D.Rank;
```

inefficiente!

# Con condizione aggiuntiva

---



# Esempio: rank con condizione aggiuntiva

---

Selezionare le tre parcelli più alte dei medici di Pisa di ciascuna specializzazione, e la matricola dei medici a cui appartengono



✂  
**KEEP  
CALM  
AND  
CTRL-C &  
CTRL-V**

questa è una condizione sui record, e deve essere imposta sia nella query che nella partition

# Esempio: rank con condizione aggiuntiva

---

Selezionare le tre parcelli più alte dei medici di Pisa di ciascuna specializzazione, e la matricola dei medici a cui appartengono

```
1  SELECT *
2  ▼ FROM(
3      SELECT M.Specializzazione, M.Matricola, M.Parcella,
4      (
5          SELECT 1 + COUNT(*)
6          FROM Medico M0
7          WHERE M0.Specializzazione = M.Specializzazione
8              AND M0.Parcella > M.Parcella
9              AND M0.Citta = 'Pisa'
10     ) AS Rank
11     FROM Medico M
12     WHERE M.Citta = 'Pisa'
13     ) AS D
14 WHERE D.Rank <= 3
15 ORDER BY D.Specializzazione, D.Rank;
```

# Su grandi moli di dati...

---



# Migliorare

---

```
1 ALTER TABLE Medico  
2 ADD INDEX (Specializzazione, Parcella);
```



Indice per migliorare le performance della correlated subquery nel select,  
ma l'indice porta con sé svantaggi relativi all'aggiornamento...alternative?

vediamole nelle prossime slide...

# Funzione GROUP\_CONCAT

---

In query con raggruppamento, restituisce **una stringa per gruppo** contenente i valori assunti da un attributo nei vari record del gruppo

*diversi da NULL*



per default, il separatore è la virgola

# Esempio

---

Per ciascuna specializzazione, indicarne il nome, la parcella media e tutte le parcelle dei medici che ne fanno parte come stringa concatenata

```
1 | SELECT M.Specializzazione,  
2 |     AVG(M.Parcella) AS ParcellaMedia,  
3 |     GROUP_CONCAT(M.Parcella) AS ListaParcelle  
4 | FROM Medico M  
5 | GROUP BY M.Specializzazione;
```

1 record per specializzazione,  
ma l'attributo ListaParcelle contiene un  
result set concatenato

Specializzazione	ParcellaMedia	ListaParcelle
Cardiologia	228.0000	260,180,230,220,250
Gastroenterologia	120.0000	130,110
Medicina generale	50.0000	50
Nefrologia	145.0000	120,170
Neurologia	200.0000	200
Ortopedia	170.0000	180,160
Otorinolaringoiatria	175.0000	250,200,150,100
Psichiatria	150.0000	150

# Modifica della capacità della lista

---

- group\_concat\_max\_len

Command-Line Format	--group_concat_max_len=#
<b>System Variable</b>	<b>Name</b> <u>group_concat_max_len</u> <b>Variable Scope</b> Global, Session <b>Dynamic Variable</b> Yes
<b>Permitted Values</b> (32-bit platforms)	<b>Type</b> integer <b>Default</b> 1024 <b>Min Value</b> 4 <b>Max Value</b> 4294967295
<b>Permitted Values</b> (64-bit platforms)	<b>Type</b> integer <b>Default</b> 1024 <b>Min Value</b> 4 <b>Max Value</b> 18446744073709547520



il risultato di `group_concat()` è inserito nella variabile di sistema `group_concat_max_len` che può essere opportunamente dimensionata

# Due variabili di sistema

---

```
1 SET GLOBAL group_concat_max_len = 5000;
```

```
1 SET SESSION group_concat_max_len = 2000;
```

```
1 SHOW SESSION VARIABLES  
2 LIKE 'group_%';
```

Variable_name	Value
group_concat_max_len	2000

```
1 SHOW GLOBAL VARIABLES  
2 LIKE 'group_%';
```

Variable_name	Value
group_concat_max_len	5000

# Funzione FIND\_IN\_SET

---

per default

Ricerca una sottostringa in una lista concatenata (con virgole come separatori)  
restituendo **la posizione della prima occorrenza**



se non trova la sottostringa, restituisce 0

# Esempio

---

Per ogni specializzazione, restituirne il nome, la parcella media e la posizione del medico con parcella 200 euro, in una classifica crescente delle parcelle

```
1 SELECT D.Specializzazione, il primo parametro è la sottostringa cercata
2     D.ParcellaMedia, ↗
3     FIND_IN_SET('200', D.ListaParcelle) AS Pos
4 FROM(
5     SELECT M.Specializzazione,
6             AVG(M.Parcella) AS ParcellaMedia,
7             GROUP_CONCAT(M.Parcella
8                         ORDER BY M.Parcella ← ordinamento crescente
9                         ) AS ListaParcelle
10    FROM Medico M
11    GROUP BY M.Specializzazione
12    ) AS D;
```

Nota: l'annotazione "il primo parametro è la sottostringa cercata" si riferisce alla funzione FIND\_IN\_SET, mentre l'annotazione "ordinamento crescente della partizione (per parcella)" si riferisce all'ordine di concatenazione dei valori nella colonna ListaParcelle.

## Esempio (cont.)

---

Per ogni specializzazione, restituirne il nome, la parcella media e la posizione del medico con parcella 200 euro, in una classifica crescente delle parcelle

Specializzazione	ParcellaMedia	Pos
Cardiologia	228.0000	0
Gastroenterologia	120.0000	0
Medicina generale	50.0000	0
Nefrologia	145.0000	0
Neurologia	200.0000	1
Ortopedia	170.0000	0
Otorinolaringoiatria	175.0000	3
Psichiatria	150.0000	0



Potrebbe essere restituito zero anche se group\_concat\_max\_len è troppo poco capiente per contenere tutta la concatenazione  
*(bisogna sempre stare attenti al dimensionamento)*

# Rank su partition con GROUP\_CONCAT

Per ogni specializzazione, effettuare il rank per parcella e restituire i tre medici con la parcella più elevata

```
1  SELECT DD.*  
2  FROM  
3  (SELECT M.Specializzazione, M.Matricola, M.Cognome, M.Parcella,  
4    FIND_IN_SET(M.Parcella, D.ListaParcelle) AS Rank  
5  FROM Medico M,  
6  (SELECT M2.Specializzazione,  
7    GROUP_CONCAT(M2.Parcella  
8      ORDER BY M2.Parcella DESC  
9    ) AS ListaParcelle  
10   FROM Medico M2  
11   GROUP BY M2.Specializzazione  
12   ) AS D  
13 WHERE M.Specializzazione = D.Specializzazione  
14   ) AS DD  
15 WHERE DD.Rank <= 3  
16 AND DD.Rank > 0  
17 ORDER BY DD.Specializzazione, DD.Rank;
```

ordinamento decrescente

group\_concat concatena fino a 1024 byte  
(di default) e poi cessa la processazione

# Dense rank su partition con GROUP\_CONCAT

Per ogni specializzazione, effettuare il dense rank per parcella e restituire i tre medici con la parcella più elevata

```
1  SELECT DD.*  
2  FROM  
3  (SELECT M.Specializzazione, M.Matricola, M.Cognome, M.Parcella,  
4   FIND_IN_SET(M.Parcella, D.ListaParcelle) AS Rank  
5  FROM Medico M,  
6  (SELECT M2.Specializzazione,  
7   GROUP_CONCAT(DISTINCT M2.Parcella  
8   ORDER BY M2.Parcella DESC  
9  ) AS ListaParcelle  senza duplicati nella partition,  
10  FROM Medico M2  
11  GROUP BY M2.Specializzazione  
12  ) AS D  ho una sola occorrenza  
13  WHERE M.Specializzazione = D.Specializzazione  
14  ) AS DD  per ogni parcella  
15  WHERE DD.Rank <= 3  
16  AND DD.Rank > 0  
17  ORDER BY DD.Specializzazione, DD.Rank;
```

dense = senza gap

# Analytics efficiente

---



# Analytics efficiente

---

Utilizzo avanzato di variabili user-defined e dell'ordine di processazione  
per scrivere **query efficienti**



uso spinto delle session variable @  
(appuccio più efficiente della definizione degli indici)

# Funzione row number

Assegnare un **numero di riga** a ogni record in Medico

```
1 | SELECT @row_number := @row_number + 1 AS RowNumber, M.*  
2 | FROM (SELECT @row_number := 0) AS N,  
3 | Medico M;
```

creazione della variabile  
e assegnamento

all'arrivo di un nuovo record da proiettare,  
prima di proiettarlo, si incrementa la variabile

la virgola effettua  
un prodotto cartesiano  
(equivale a cross join)

si usa := perché non è una SET

# Row number

---

RowNumber	Matricola	Cognome	Nome	Specializzazione	Parcella	Citta
1	001	Rossi	Mario	Otorinolaringoiatria	100	Pisa
2	002	Verdi	Luigi	Otorinolaringoiatria	150	Pisa
3	003	Gialli	Franco	Otorinolaringoiatria	200	Pisa
4	004	Turchesi	Viola	Otorinolaringoiatria	250	Pisa
5	005	Neri	Marta	Gastroenterologia	130	Milano
6	006	Bianchi	Ada	Ortopedia	160	Milano
7	007	Rosi	Alvaro	Ortopedia	180	Roma
8	008	Gialli	Margherita	Nefrologia	120	Roma
9	009	Arancioni	Ugo	Nefrologia	170	Firenze
10	010	Celesti	Clelia	Neurologia	200	Firenze
11	011	Marroni	Osvaldo	Psichiatria	150	Pisa
12	012	Grigi	Ettore	Medicina generale	50	Pisa
13	013	Blu	Fabiola	Gastroenterologia	110	Siena
14	014	Indachi	Loredana	Cardiologia	180	Pisa
15	015	Ciani	Gualtiero	Cardiologia	230	Pisa
16	016	Verdolini	Maria Paola	Cardiologia	220	Pisa
17	017	Amaranti	Adevane	Cardiologia	250	Pisa
18	018	Terra di Siena	Bruciata	Cardiologia	260	Siena

# Row number su partition: query errata

Per ogni specializzazione, associare un row number a ogni medico in ordine decrescente di parcella

```
1 ▼SELECT IF(@spec = M.Specializzazione, ← 1
2           @rownumber := @rownumber + 1, ← 2
3           @rownumber := 1
4   ) AS RowNumber,
5   (@spec := M.Specializzazione) AS Specializzazione,
6   M.Matricola,
7   M.Parcella
8 FROM Medico M,
9   (SELECT (@spec := '')) AS N
10 ORDER BY M.Specializzazione, M.Parcella DESC;
```

qui si presuppone che @spec sia valutata prima di essere assegnata, ma l'istruzione alla riga 5 potrebbe essere eseguita prima di quella alla riga 1



MySQL processa gli attributi da proiettare nell'ordine che vuole (lazy evaluation), non si può mai fare affidamento sull'ordine in cui è scritta la proiezione

# Funzione LEAST()

---

possono essere anche espressioni o assegnamenti

Restituisce **il più piccolo** dei due operandi



può essere usata, in modo malizioso, per forzare un assegnamento  
esattamente in un punto della proiezione

# Row number con LEAST()

Per ogni specializzazione, associare un row number a ogni medico in ordine decrescente di parcella

la specializzazione della current row è uguale a quella del record precedente?

```
1 SELECT IF(@spec = M.Specializzazione,  
2           @rownumber := @rownumber + 1,  
3           @rownumber := 1 + LEAST(0, @spec := M.Specializzazione))  
4 AS RowNumber,  
5 M.Specializzazione,  
6 M.Matricola,  
7 M.Parcella  
8 FROM Medico M,  
9      (SELECT (@spec := '')) AS N  
10 ORDER BY M.Specializzazione, M.Parcella DESC;
```

→ sì → incremento

→ no → restart numbering  
e riassegno @spec

Nota bene: least() restituisce sempre 0 perché il valore più piccolo fra 0 e una stringa è sempre 0, quindi la riga 3 è sempre @rownumber := 1+0, ma in aggiunta forza l'assegnamento di @spec proprio dopo aver valutato il valore di @spec nella condizione dell'if (riga 1)

# Risultato row number su partition

---

	RowNumber	Specializzazione	Matricola	Parcella
Cardiologia	1	Cardiologia	018	260
	2	Cardiologia	017	260
	3	Cardiologia	015	230
	4	Cardiologia	016	220
	5	Cardiologia	014	180
Gastroenterologia	1	Gastroenterologia	005	130
	2	Gastroenterologia	013	110
Medicina generale	1	Medicina generale	012	50
	1	Nefrologia	009	170
Nefrologia	2	Nefrologia	008	120
	1	Neurologia	010	200
Ortopedia	1	Ortopedia	007	180
	2	Ortopedia	006	160
Otorinolaringoiatr	1	Otorinolaringoiatr	004	250
	2	Otorinolaringoiatr	003	200
	3	Otorinolaringoiatr	002	150
	4	Otorinolaringoiatr	001	100
Psichiatria	1	Psichiatria	011	150

Nei vostri pensieri...

---



# Dense rank con LEAST()

---

Per ogni specializzazione, effettuare il dense rank per parcella e restituire i tre medici con la parcella più elevata

# Dense rank con LEAST()

```
1  SELECT D.*  
2  FROM(          ← la specializzazione è quella del record precedente?  
3      SELECT IF(@spec = M.Specializzazione  
4  
5      ,  
6          IF(@parc = M.Parcella, ← la parcella è quella del  
7              @rank := @rank,  
8              @rank := @rank + 1  
9              + LEAST(0, @parc := M.Parcella)  
10             )  
11            ,  
12                @rank := 1 + LEAST(0, @spec := M.Specializzazione)  
13                + LEAST(0, @parc := M.Parcella)  
14            ) AS DenseRank,  
15            M.Specializzazione,  
16            M.Matricola,  
17            M.Parcella  
18    FROM Medico M,  
19        (SELECT (@spec := '')) AS N  
20    ORDER BY M.Specializzazione, M.Parcella DESC  
21  ) AS D  
22 WHERE D.DenseRank <= 3;
```

# Risultato dense rank

---

DenseRank	Specializzazione	Matricola	Parcella
1	Cardiologia	018	260
1	Cardiologia	017	260
2	Cardiologia	015	230
3	Cardiologia	016	220
1	Gastroenterologia	005	130
2	Gastroenterologia	013	110
1	Medicina generale	012	50
1	Nefrologia	009	170
2	Nefrologia	008	120
1	Neurologia	010	200
1	Ortopedia	007	180
2	Ortopedia	006	160
1	Otorinolaringoiatr	004	250
2	Otorinolaringoiatr	003	200
3	Otorinolaringoiatr	002	150
1	Psichiatria	011	150

senza gap



stesso rank

# Funzione GREATEST()

---

Restituisce **il più grande** fra i due operandi



come prevedibile, anche greatest() è utilizzata per forzare assegnamenti di variabili posti come operando

# Rank con LEAST() e GREATEST()

---

```
1  SELECT IF(@spec = M.Specializzazione
2
3    ,
4      IF(@parc = M.Parcella,
5          @rank := @rank + LEAST(0, @gap := @gap + 1),
6          @rank := @rank + GREATEST(@gap, @gap := 1)
7              + LEAST(0, @parc := M.Parcella)
8      )
9
10     ,
11         @rank := 1 + LEAST(0, @spec := M.Specializzazione)
12             + LEAST(0, @parc := M.Parcella)
13             + LEAST(0, @gap := 1)
14     ) AS Rank,
15     M.Specializzazione,
16     M.Matricola,
17     M.Parcella
18   FROM Medico M,
19       (SELECT (@spec := '')) AS N
20 ORDER BY M.Specializzazione, M.Parcella DESC;
```

# Vieni Olga...

---



# Grazie Olga...



```
IF(@spec = M.Specializzazione  
    ,  
        IF(@parc = M.Parcella,  
            @rank := @rank + LEAST(0, @gap := @gap + 1),  
            @rank := @rank + GREATEST(@gap, @gap := 1)  
                + LEAST(0, @parc := M.Parcella)  
        )  
    ,  
        @rank := 1 + LEAST(0, @spec := M.Specializzazione)  
            + LEAST(0, @parc := M.Parcella)  
            + LEAST(0, @gap := 1)  
    ) AS Rank,
```

**se la specializzazione è la stessa:**

**se la parcella è la stessa:**

si assegna al rank il rank del record precedente (least( ) torna 0 e incrementa gap);

**altrimenti:**

si assegna al rank il rank del record precedente più il gap (greatest( ) torna il valore di @gap e lo reimposta a 1, e least( ) torna 0 aggiornando la parcella corrente);

**altrimenti:**

si assegna al rank 1 (le tre chiamate a least( ) tornano 0 aggiornando, rispettivamente, la specializzazione corrente (@spec), la parcella corrente (@parc) e il gap (@gap))

# Risultato rank

---

gap

Rank	Specializzazione	Matricola	Parcella
1	Cardiologia	018	260
1	Cardiologia	017	260
3	Cardiologia	015	230
1	Gastroenterologia	005	130
2	Gastroenterologia	013	110
1	Medicina generale	012	50
1	Nefrologia	009	170
2	Nefrologia	008	120
1	Neurologia	010	200
1	Ortopedia	007	180
2	Ortopedia	006	160
1	Otorinolaringoiatr	004	250
2	Otorinolaringoiatr	003	200
3	Otorinolaringoiatr	002	150
1	Psichiatria	011	150

stesso rank

# Top-K queries

---

Individuare, per ciascuna specializzazione, i medici nelle  
**prime K posizioni** della classifica per parcella

è un problema risolvibile mediante funzionalità analytics



# Top-K query (K = 3)

---

```
1  SELECT D.*  
2  FROM(  
3      SELECT IF(@spec = M.Specializzazione  
4          ,  
5              IF(@parc = M.Parcella,  
6                  @rank := @rank + LEAST(0, @gap := @gap + 1),  
7                  @rank := @rank + GREATEST(@gap, @gap := 1)  
8                  + LEAST(0, @parc := M.Parcella)  
9              )  
10             ,  
11                 @rank := 1 + LEAST(0, @spec := M.Specializzazione)  
12                     + LEAST(0, @parc := M.Parcella)  
13                     + LEAST(0, @gap := 1)  
14             ) AS Rank,  
15             M.Specializzazione,  
16             M.Matricola,  
17             M.Parcella  
18     FROM Medico M,  
19         (SELECT (@spec := '')) AS N  
20     ORDER BY M.Specializzazione, M.Parcella DESC  
21 ) AS D  
22 WHERE D.Rank <= 3;
```

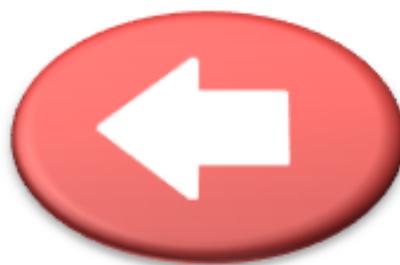
# Top-K query (K = 3)

---

Rank	Specializzazione	Matricola	Parcella
1	Cardiologia	018	260
1	Cardiologia	017	260
3	Cardiologia	015	230
1	Gastroenterologia	005	130
2	Gastroenterologia	013	110
1	Medicina generale	012	50
1	Nefrologia	009	170
2	Nefrologia	008	120
1	Neurologia	010	200
1	Ortopedia	007	180
2	Ortopedia	006	160
1	Otorinolaringoiatr	004	250
2	Otorinolaringoiatr	003	200
3	Otorinolaringoiatr	002	150
1	Psichiatria	011	150

# Lead/Lag

---



**Back**



**Forward**

# Lead/Lag

---

Ricavano il valore  $\checkmark$  di un attributo relativo a una row **posta X posizioni prima (lag) o dopo (lead)** la current row

*anche usando function*



ovviamente si presuppone un ordinamento nella partition  
affinché il "balzo" avanti o indietro abbia un senso

# Lag

---

Considerati gli esordi di ciascuna patologia del paziente ‘slq6’, restituire patologia, data dell’esordio e data dell’esordio precedente della stessa patologia

# Lag: esempio

---

Considerati gli esordi di ciascuna patologia del paziente ‘slq6’, restituire patologia, data dell’esordio e data dell’esordio precedente della stessa patologia

```
1  SELECT E.Patologia,
2      E.DataEsordio,
3  IF(@pat = E.Patologia
4      ,
5      se la patologia della row precedente (@pat) è quella della current row
6      proietto la data dell'esordio precedente (@data) e aggiorno il valore di @data
7      alla data dell'esordio della current row (E.DataEsordio) mediante LEAST()
8      ,
9      altrimenti aggiorno la patologia e la data esordio a quelle della current row
10
11
12  )
13 FROM Esordio E,
14     (SELECT (@pat := '')) AS N
15 WHERE E.Paziente = 'slq6'
16 ORDER BY E.Patologia, E.DataEsordio;
```

# Nota importante sulle variabili user-defined

---

Una variabile user-defined **non può contenere una data**, una data può esservi memorizzata come numero di giorni trascorsi dall'anno 0

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				



si ottiene con `to_days()`  
(l'operazione inversa è `from_days()`)

# Lag: esempio (cont.)

---

Considerati gli esordi di ciascuna patologia del paziente ‘slq6’, restituire patologia, data dell’esordio e data dell’esordio precedente della stessa patologia

```
1  SELECT E.Patologia,
2      E.DataEsordio,
3  -      IF(@pat = E.Patologia
4      ,
5  -          FROM_DAYS(@esordio_prec
6              + LEAST(0, @esordio_prec := TO_DAYS(E.DataEsordio))
7              + LEAST(0, @pat := E.Patologia)
8      )
9      ,
10         NULL + LEAST(0, @pat := E.Patologia)
11         + LEAST(0, @esordio_prec := TO_DAYS(E.DataEsordio))
12    )
13  FROM Esordio E,
14      (SELECT (@pat := '')) AS N
15 WHERE E.Paziente = 'slq6'
16 ORDER BY E.Patologia, E.DataEsordio;
```

# Lag con aggregazione

---

Considerato ciascun paziente, indicarne il codice fiscale e il numero medio di giorni trascorsi dall'inizio di ogni suo esordio e l'inizio dell'esordio precedente

```
1  SELECT D.Paziente, AVG(GGdaPrec)
2  FROM(
3      SELECT E.Paziente,
4      IF(@paz = E.Paziente
5          ,
6              DATEDIFF(E.DataEsordio, @esordio_prec)
7                  + LEAST(0, @esordio_prec := E.DataEsordio)
8          ,
9              NULL + LEAST(0, @paz := E.Paziente)
10                 + (@esordio_prec = NULL)
11      ) AS GGdaPrec
12  FROM Esordio E,
13      (SELECT (@paz := '')) AS N
14  ORDER BY E.Paziente, E.DataEsordio
15 ) AS D
16 GROUP BY D.Paziente;
```

# Lag con LEAST() e stringhe

---

su MySQL sommate 0 a una stringa restituisce 0

Se l'attributo a cui si “somma” lo zero (per effettuare l'assegnamento con LEAST( )) è una stringa, **occorre usare CONCAT()**



in questo modo lo 0 restituito da least() viene concatenato alla stringa ed essa mantiene il suo valore (con lo zero in coda). Prima di proiettare, occorre togliere lo zero: lo si fa con replace()

# Esempio

---

Date le visite del paziente 'slq6' in ciascuna specializzazione, restituire cognome del medico, specializzazione, data, e cognome del medico della stessa specializzazione che ha visitato il paziente la volta precedente

# Soluzione

```
1  SELECT D.Cognome,  
2      D.Specializzazione,  
3      D.`Data`,  
4  IF(@spec = D.Specializzazione  
     ,  
     REPLACE(CONCAT(@med,LEAST(0, @med := D.Cognome)), '0', '')  
     ,  
     NULL + LEAST(0, @spec := D.Specializzazione)  
     + LEAST(0, @med := D.Cognome))  
10    )  
11  FROM (  
12      SELECT *  
13      FROM Visita V  
14      INNER JOIN  
15          Medico M ON V.Medico = M.Matricola  
16      ,  
17      (SELECT (@spec := '')) AS N  
18    ) AS D  
19  WHERE D.Paziente = 'slq6'  
20  ORDER BY D.Specializzazione, D.`Data`;
```

REPLACED

se usassimo solo `least(0,@med:=D.Cognome)`, quando l'if vale true proietteremmo '0' e non @med. Per proiettare l'attuale valore di @med, lo si assegna con `least()` si concatena uno '0' in coda, dopodiché `replace("stringa", '0', '')` toglie lo '0' in coda e proietta @med (che vale D.Cognome)

derived table obbligatoria

# Lead

---

Lead può essere implementata **invertendo il sort nella partition**



invertendo l'ordinamento della partition, i record più recenti  
vengono prima della current row

# Windowing functions

---

Medico	Paziente	Data	Mutuata
001	slq6	2003-06-03	0
001	slq6	2004-04-05	0
002	slq6	2005-01-16	0
002	slq6	2007-11-30	0
003	slq6	2009-12-03	0
003	slq6	2012-07-21	0
004	slq6	2007-02-23	0
005	slq6	2005-02-22	0
005	slq6	2009-11-21	0
006	slq6	2006-01-24	0
006	slq6	2009-07-18	0
007	slq6	2004-11-30	0
007	slq6	2009-08-01	0
001	slq6	2003-08-01	0
001	slq6	2004-11-30	0
000	slq6	2005-01-01	0
000	slq6	2005-10-01	0
000	slq6	2006-05-01	0

Medico	Paziente	Data	Mutuata
001	slq6	2003-06-03	0
001	slq6	2004-04-05	0
002	slq6	2005-01-16	0
002	slq6	2007-11-30	0
003	slq6	2009-12-03	0
003	slq6	2012-07-21	0
004	slq6	2007-02-23	0
005	slq6	2005-02-22	0
005	slq6	2009-11-21	0
006	slq6	2006-01-24	0
006	slq6	2009-07-18	0
007	slq6	2004-11-30	0
007	slq6	2009-08-01	0
001	slq6	2003-08-01	0
001	slq6	2004-11-30	0
000	slq6	2005-01-01	0
000	slq6	2005-10-01	0
000	slq6	2006-05-01	0

Medico	Paziente	Data	Mutuata
001	slq6	2003-06-03	0
001	slq6	2004-04-05	0
002	slq6	2005-01-16	0
002	slq6	2007-11-30	0
003	slq6	2009-12-03	0
003	slq6	2012-07-21	0
004	slq6	2007-02-23	0
005	slq6	2005-02-22	0
005	slq6	2009-11-21	0
006	slq6	2006-01-24	0
006	slq6	2009-07-18	0
007	slq6	2004-11-30	0
007	slq6	2009-08-01	0
001	slq6	2003-08-01	0
001	slq6	2004-11-30	0
000	slq6	2005-01-01	0
000	slq6	2005-10-01	0
000	slq6	2006-05-01	0

# Windowing functions

esempi sono moving sum, moving average, somme cumulative...

Processano un result set e calcolano valori aggregati basati su record  
“adiacenti” alla current row



l'adiacenza è definita da un sort sul result set

current  
row →

A diagram illustrating the concept of adjacency. A wavy arrow points from the 'A' icon to the 'current row' label. Another arrow points from the 'current row' label to the table.

Medico	Paziente	Data	Mutata
001	slq6	2003-06-03	0
001	slq6	2004-04-05	0
002	slq6	2005-01-16	0
002	slq6	2007-11-30	0
003	slq6	2009-12-03	0
003	slq6	2012-07-21	0
004	slq6	2007-02-23	0
005	slq6	2005-02-22	0
005	slq6	2009-11-21	0
006	slq6	2006-01-24	0
006	slq6	2009-07-18	0
006	slq6	2009-07-18	0
006	slq6	2009-07-18	0
006	slq6	2009-07-18	0

The table shows a sequence of records. The 4th record (Medico 002, Paziente slq6, Data 2005-01-16) is highlighted with a red background, representing the 'current row'. A large red bracket on the right side of the table is labeled 'window', indicating the context of adjacent rows for windowing operations.

# Esempio: moving average

---



Calcolare ✓ la durata media di ogni terapia del paziente 'ttw2', relativamente alla terapia precedente e a quella successiva, caratterizzate dallo stesso farmaco.  
Proiettare per ogni terapia il farmaco assunto, la sua durata e la durata media rispetto alla terapia precedente e successiva con lo stesso farmaco

# Vediamo il problema...

```
1 SELECT T.Patologia,  
2     T.Farmaco,  
3     T.DataInizioTerapia,  
4     DATEDIFF(T.DataFineTerapia,  
5                 T.DataInizioTerapia) AS Durata  
6 FROM Terapia T  
7 WHERE T.Paziente = 'ttw2'  
8     AND T.Farmaco='Gaviscon'  
9 ORDER BY T.DataEsordio;
```



In questa slide si considera solo il farmaco "Gaviscon" per dare un'idea del problema

Patologia	Farmaco	DataInizioTerapia	Durata
✗ Gastrite	Gaviscon	1994-12-03	23
✓ Reflusso gastroesofageo	Gaviscon	2008-01-15	26
✓ Reflusso gastroesofageo	Gaviscon	2009-10-20	21
✗ Gastrite	Gaviscon	2015-03-15	7

manca la terapia precedente

$$(23+26+21)/3 = 23.3$$

$$(26+21+7)/3 = 18$$

✓ record che contribuirà al risultato

✗ record scartato: la window è incompleta

manca la terapia successiva

# Come si fa?

---

Il problema risiede nel generare una partition che permetta aggregazione di un record con il **precedente** e il **successivo**



in questo caso l'ordine è stabilito dalla data  
e dal farmaco, che deve essere lo stesso

# Ogni record riveste più ruoli...

Patologia	Farmaco	DataInizioTerapia	Durata
Gastrite	Gaviscon	1994-12-03	23
Reflusso gastroesofageo	Gaviscon	2008-01-15	26
Reflusso gastroesofageo	Gaviscon	2009-10-20	21
Gastrite	Gaviscon	2015-03-15	7

Patologia	Farmaco	DataInizioTerapia	Durata
Gastrite	Gaviscon	1994-12-03	23
Reflusso gastroesofageo	Gaviscon	2008-01-15	26
Reflusso gastroesofageo	Gaviscon	2009-10-20	21
Gastrite	Gaviscon	2015-03-15	7

■ current row

■ attributo d'interesse

○ window attorno alla current row



se usassimo i join, dovremmo considerare  
3 istanze di Terapia per affiancare a ogni  
terapia target, la precedente e la successiva!

il record con il piolino a  
fianco fa da current row  
nella partition blu  
e da record precedente  
nella partition fuxia. Non  
è mai record successivo  
perché qui non esiste una  
terapia antecedente al  
3/12/1994

# Step 1: row numbering delle terapie

---

```
1 ▼SELECT IF(@farmaco = X.Farmaco,
2                 @rownumber := @rownumber + 1,
3                 @rownumber := 1 + LEAST(0, @farmaco := X.Farmaco)
4 ▲             ) AS RowNumber,
5                 X.Farmaco,
6 ▼                 DATEDIFF(IFNULL(X.DataFineTerapia, CURRENT_DATE),
7                               X.DataInizioTerapia
8 ▲                         ) AS Durata
9 FROM
10 ▼     (SELECT *
11          FROM Terapia T
12          WHERE T.Paziente = 'ttw2'
13 ▲     ) AS X
14     ,
15 ▼     (SELECT (@farmaco := ''))
16 ▲     ) AS Y
17 ORDER BY
18                 X.Farmaco,
19                 X.DataInizioTerapia;
```

# Cosa si ottiene

---

RowNumber	Farmaco	Durata
1	Fluibron	5504
1	Gaviscon	23
2	Gaviscon	26
3	Gaviscon	21
4	Gaviscon	7
1	Inderal	53

dato un farmaco, le terapie con lo stesso farmaco sono numerate con interi crescenti: terapie più recenti hanno RowNumber più alto

## Step 2: replicazione della terapia corrente

```
1  SELECT A.RowNumber + B.Num AS GapNumber,
2      B.Num,
3      A.Farmaco,
4      A.Durata
5  FROM(
6    SELECT IF(@farmaco = X.Farmaco,
7              @rownumber := @rownumber + 1,
8              @rownumber := 1 + LEAST(0, @farmaco := X.Farmaco)
9            ) AS RowNumber,
10     X.Farmaco,
11     DATEDIFF(...) AS Durata
12   FROM
13     (...) AS X
14   ,
15     (...) AS Y
16   ORDER BY
17           X.Farmaco, X.DataInizioTerapia
18 ) AS A
19 ,
20 (
21   SELECT 0 AS Num
22   UNION
23   SELECT 1 AS Num
24   UNION
25   SELECT 2 AS Num
26 ) AS B;
```

triplico ogni record  
con un prodotto  
cartesiano ad arte

Num
0
1
2

# Prodotto cartesiano triplicante...

Patologia	Farmaco	DataInizioTerapia	Durata
Gastrite	Gaviscon	1994-12-03	23
Reflusso gastroesofageo	Gaviscon	2008-01-15	26
Refluxo gastroesofageo	Gaviscon	2009-10-20	21
Gastrite	Gaviscon	2015-03-15	7



GapNumber	Num	Farmaco	Durata
1	0	Gaviscon	21
2	1	Gaviscon	21
3	2	Gaviscon	21
2	0	Gaviscon	7
3	1	Gaviscon	7
4	2	Gaviscon	7
3	0	Gaviscon	26
4	1	Gaviscon	26
5	2	Gaviscon	26
4	0	Gaviscon	23
5	1	Gaviscon	23
6	2	Gaviscon	23

Num
0
1
2



Ogni terapia target è ora tripla.  
Num = 1 indica il record originale,  
Num={0, 2} sono le copie

# Cosa succede

---

Una window per ogni terapia target che ha un record precedente (successivo) che rispetta le condizioni: stesso farmaco, stesso paziente e data precedente (successiva)

GapNumber	Num	Farmaco	Durata
1	0	Gaviscon	21
2	1	Gaviscon	21
3	2	Gaviscon	21
2	0	Gaviscon	7
3	1	Gaviscon	7
4	2	Gaviscon	7
3	0	Gaviscon	26
4	1	Gaviscon	26
5	2	Gaviscon	26
4	0	Gaviscon	23
5	1	Gaviscon	23
6	2	Gaviscon	23

} window 1  
} window 2  
} window 3

■ current row ( $\text{Num} = 1$ )

**DON'T FORGET!**  
nella figura si considera un solo farmaco

# Squash

---

```
1  SELECT C.Farmaco,
2      MAX(CASE C.Num
3          WHEN 1 THEN C.Durata
4          ELSE NULL
5          END
6          ) AS Durata,
7      AVG(C.Durata) AS WindowAVG
8  ▼ FROM (
9      SELECT A.RowNumber + B.Num AS GapNumber,
10         B.Num,
11         A.Farmaco,
11         A.Durata
12  ▲     ) AS C
13  GROUP BY C.Farmaco,
14          C.GapNumber
15  HAVING COUNT(*) > 2
16  ORDER BY C.Farmaco;
```

# Risultato

---

Farmaco	Durata	WindowAVG
Aprovel	50	50.0000
Aspirina	5249	5249.0000
Efferalgan	27	27.0000
EN	5	5.0000
Fluibron	5306	5306.0000
Gaviscon	26	23.3333
Inderal	53	53.0000
Lobivon	56	56.0000
Lyrica	1242	1242.0000
Peptazol	7439	7439.0000
Protargolo	26	26.0000
Quait	53	53.0000
Seglor	23	23.0000
Trittico	991	991.0000

