SQL

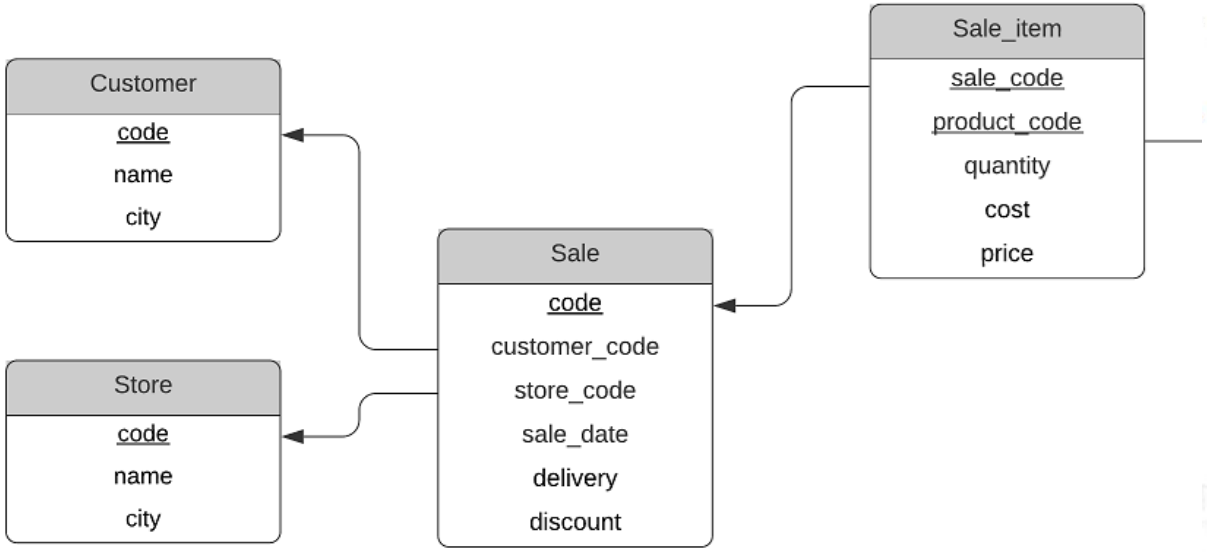
Funciones analíticas

UNIVERSIDAD AUSTRAL

INGENIERÍA

Analytic Functions

Sales by Store and Month



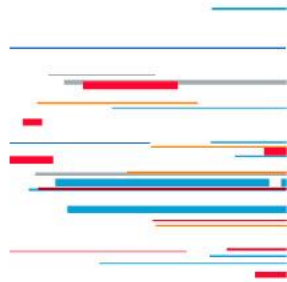

```
SELECT month(sale_date) Mes, st.name Store, SUM(monto-discount+delivery) AS Monto
FROM commerce.customer c INNER JOIN commerce.sale s ON c.code = s.customer_code
   INNER JOIN (SELECT s.code AS code, sum(si.quantity*si.price) AS monto
                  FROM commerce.sale s INNER JOIN commerce.sale_item si ON s.code=si.sale_code
                  GROUP BY s.code) sm ON s.code = sm.code
   INNER JOIN commerce.store st ON s.store_code = st.code
GROUP BY month(sale_date), st.name
```

Analytic Functions

Group By vs Pivot Table

MES	NAME	MONTO
1	Pergamino	244
1	Libres	1010
2	Baires I	3845
2	Baires II	10031
2	Pergamino	1289

Sucursal / Mes	1	2	Total
Baires I		3845	**3845**
Baires II		10031	**10031**
Libres	1010		**1010**
Pergamino	244	1289	**1533**
Total	**1254**	**15165**	**16419**



UNIVERSIDAD AUSTRAL
INGENIERÍA

CUBE Operator

Sintaxis: GROUP BY CUBE (ATT1, ATT2…)

Includes Totals and Subtotals

MES	NAME	MONTO
1	Pergamino	244
1	Libres	1010
1	-	1254
2	Baires I	3845
2	Baires II	10031
2	Pergamino	1289
2	-	15165
-	Libres	1010
-	Baires I	3845
-	Baires II	10031
-	Pergamino	1533
-	-	16419

```
SELECT st.name, month(sale_date) AS mes,
        SUM(monto-discount+delivery) AS monto
FROM (…)
GROUP BY CUBE (month(sale_date), st.name)
```

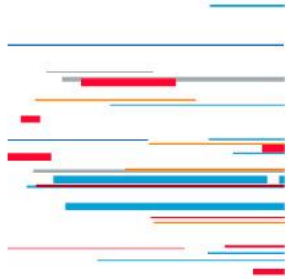
UNIVERSIDAD AUSTRAL
INGENIERÍA

Query Clause Evaluation Order

UNIVERSIDAD AUSTRAL
INGENIERÍA

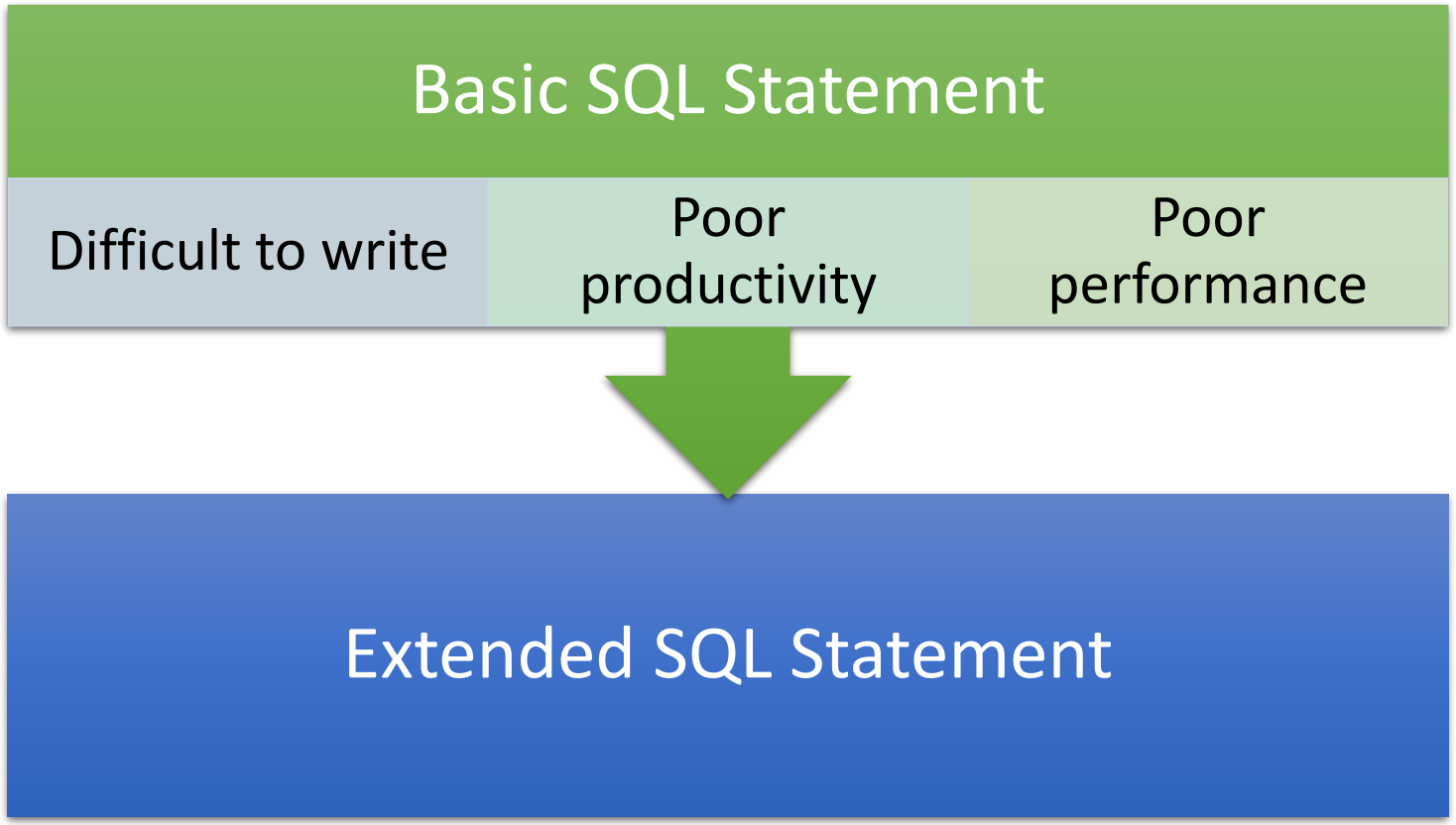
Analytic Functions

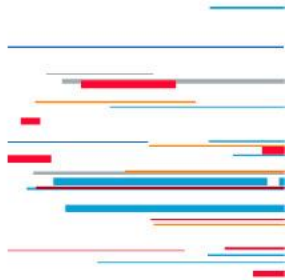
- Top and worst performers
- Trends
- Quantitative Contributions



UNIVERSIDAD AUSTRAL
INGENIERÍA

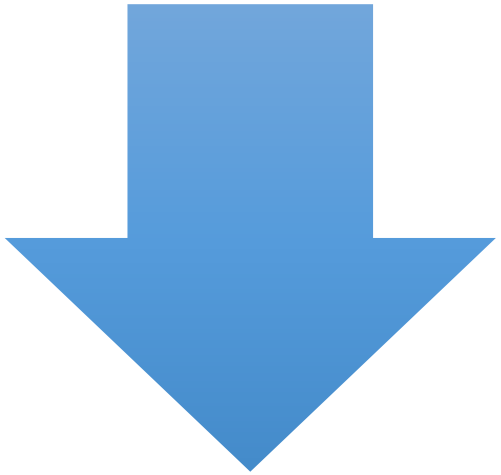
Analytic Functions





UNIVERSIDAD AUSTRAL
INGENIERÍA

Analytic Functions vs Aggregate Functions

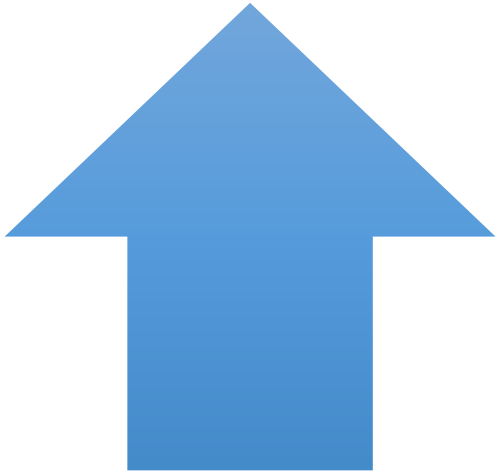


Aggregate Function

- Computes one value
- Reduces group to a single row
- Calculated before analytic functions

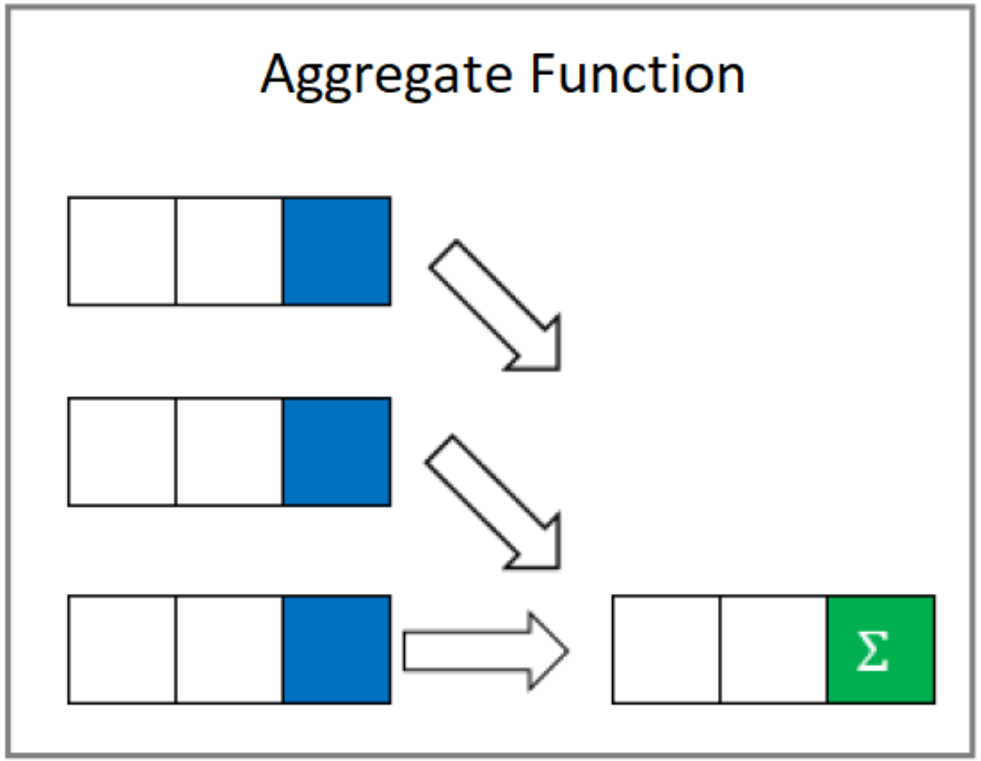
Analytic Function

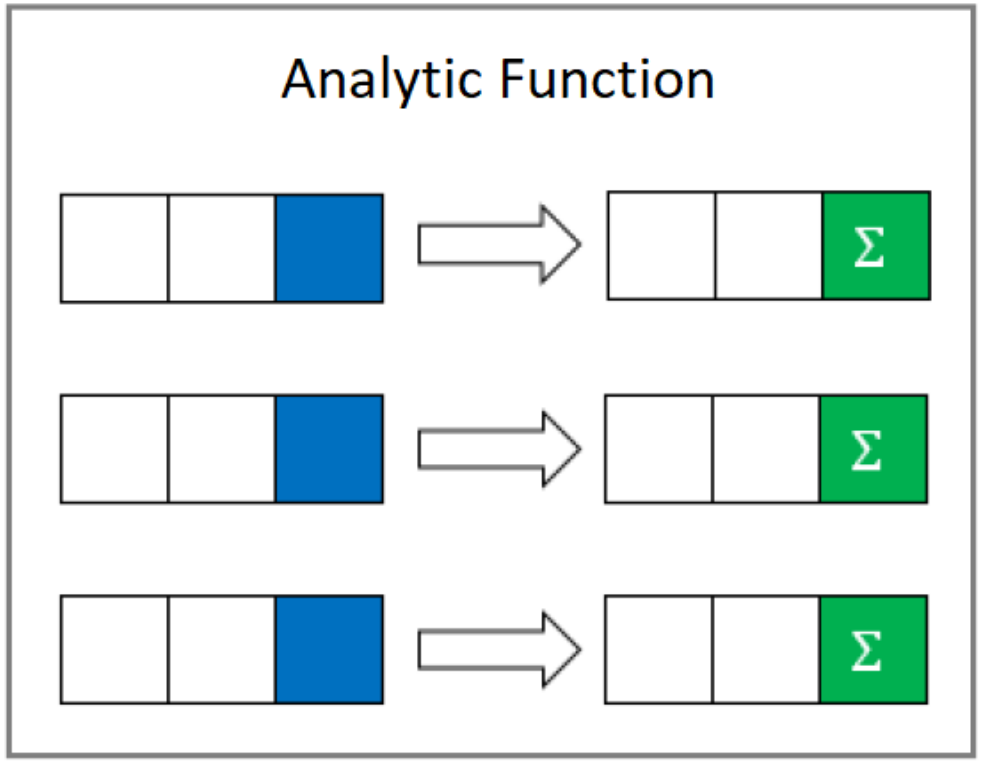
- Computes multiple values
- Preserves number of rows in a group
- Calculated after aggregate functions



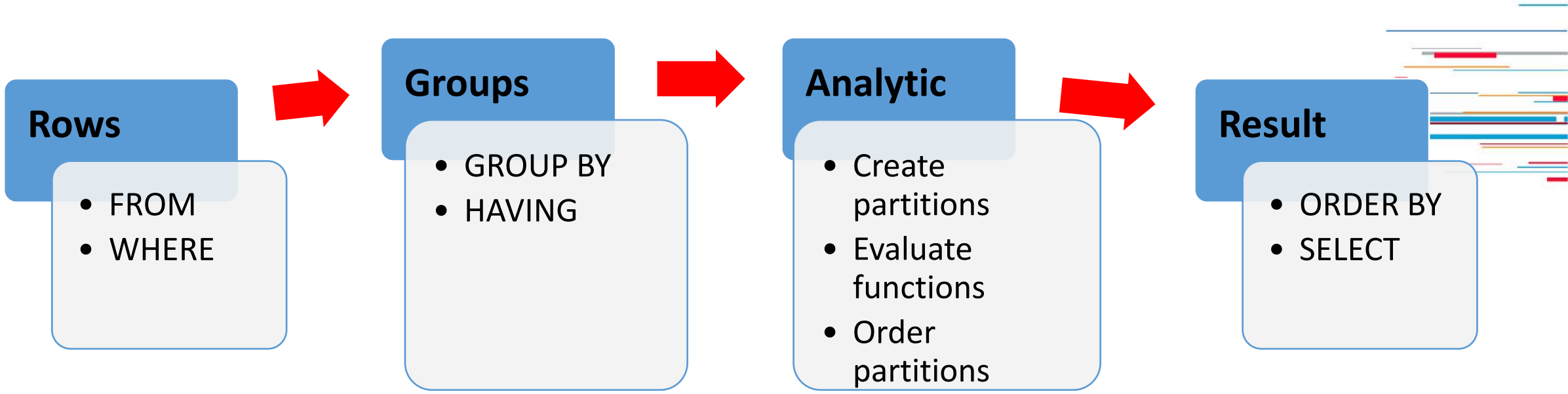
UNIVERSIDAD AUSTRAL
INGENIERÍA

Analytic Functions vs Aggregate Functions





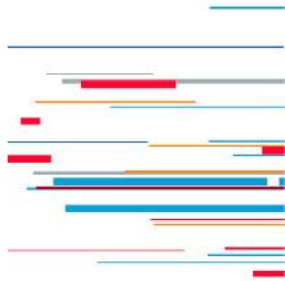
Analytic Fc Evaluation Order



UNIVERSIDAD AUSTRAL
INGENIERÍA

RANK: Basic Syntax

- `<AnalyticFunction> ([<column-list>]) OVER ( [ORDER BY <ordering>] )`
  - Place in SELECT clause list
  - OVER clause identifies window (set of rows)
  - Ordering criteria for function evaluation



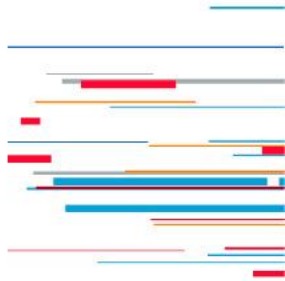
Ranking de alumnos:

```
SELECT a.nombre, AVG(c.nota) AS Promedio,
RANK() OVER (ORDER BY AVG(c.nota) DESC) AS Ranking
FROM escuela.cursa c NATURAL JOIN escuela.alumno a
GROUP BY a.nombre
ORDER BY AVG(c.nota) DESC
```

nombre	Promedio	Ranking
Bartolome Mitre	10	1
Arturo Frondizi	9	2
Juan Domingo Peron	8	3
Arturo Illia	8	3

RANK: Charactertistics

- Sets a *ranking* for each row, according to the criteria
- When more than one row with the same ranking
  - Number in the rows are repeated
  - Next rows are skipped
  - ie: 1,2,3,3,3,6,7



Ranking de clientes:

```
SELECT c.name AS Cliente, SUM(t.total + s.delivery - s.discount) AS Consumo,
        RANK() OVER (ORDER BY SUM(t.total + s.delivery - s.discount) DESC) AS Ranking
FROM commerce.customer c JOIN commerce.sale s ON c.code = s.customer_code
        JOIN (SELECT si.sale_code, SUM(si.quantity * si.price) AS Total
        FROM commerce.sale_item si
        GROUP BY si.sale_code) t ON s.code = t.sale_code
GROUP BY c.name
ORDER BY Ranking
```

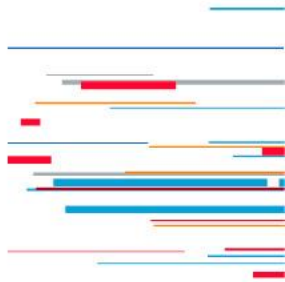
Cliente	Consumo	Ranking
Arturo Frondizi	7080	1
Hipolito Yrigoyen	4024	2
Arturo Illia	2719	3
Bartolome Mitre	2596	4

RANK VARIATIONS

Golf Leaderboard				
Score	RANK()	DENSE_RANK()	NTILE (2)	ROW_ NUMBER()
-10	1	1	1	1
-9	2	2	1	2
-9	2	2	2	3
-8	4	3	2	4

RANK VARIATIONS

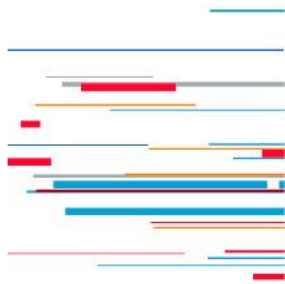
```
SELECT m.nombre, COUNT(*) AS Cursantes,
        RANK() OVER (ORDER BY COUNT(*) DESC) AS Rank_puro,
        DENSE_RANK() OVER (ORDER BY COUNT(*) DESC) AS Rank_DS,
        NTILE(2) OVER (ORDER BY COUNT(*) DESC) AS Rank_Ntile,
        ROW_NUMBER() OVER (ORDER BY COUNT(*) DESC) AS Rank_RN
FROM escuela.materia m NATURAL JOIN escuela.cursa c
GROUP BY m.nombre
ORDER BY COUNT(*) DESC
```



nombre	Cursantes	Rank_puro	Rank_DS	Rank_Ntile	Rank_RN
Arte	3	1	1	1	1
Matematica	3	1	1	1	2
Educacion fisica	2	3	2	1	3
Geografia	2	3	2	2	4
Historia	2	3	2	2	5
Literatura	2	3	2	2	6

PARTITION: Basic Syntax

- <AnalyticFunction> ([<column-list>]) OVER ([PARTITION BY <partitioning>] [[ORDER BY <ordering>])
  - PARTITION BY keywords
  - Divides result into partitions
  - Analytic function evaluated for each partition



Ranking de alumnos por materia:

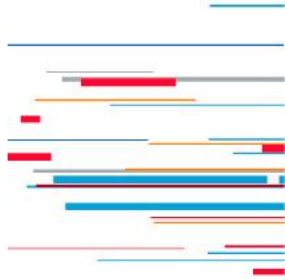
```
SELECT m.nombre Materia, a.nombre Alumno, c.nota nota,
RANK() OVER
(PARTITION BY m.nombre ORDER BY c.nota DESC) Ranking
FROM escuela.alumno a NATURAL JOIN escuela.cursa c
NATURAL JOIN escuela.materia m
ORDER BY Materia, Ranking
```

Materia	Alumno	nota	Ranking
Arte	Bartolome Mitre	10	1
Arte	Arturo Illia	9	2
Arte	Arturo Frondizi	9	2
Educacion fisica	Arturo Illia	10	1
Educacion fisica	Juan Domingo Peron	9	2
Geografia	Arturo Frondizi	9	1
Geografia	Arturo Illia	8	2
Historia	Arturo Illia	7	1
Historia	Juan Domingo Peron	7	1
Literatura	Juan Domingo Peron	9	1
Literatura	Arturo Illia	7	2
Matematica	Arturo Frondizi	10	1
Matematica	Bartolome Mitre	10	1
Matematica	Arturo Illia	7	3

PARTITION

Ranking de clientes por sucursal:

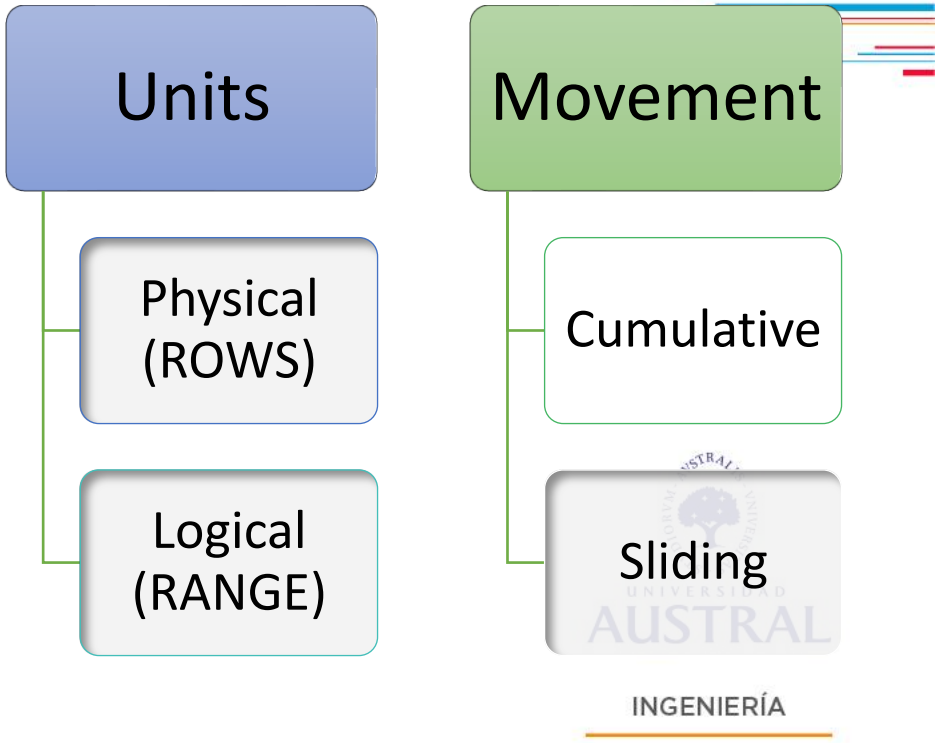
```
SELECT st.name Sucursal, c.name Cliente,
        SUM(t.total + s.delivery - s.discount) AS Consumo,
        RANK() OVER (PARTITION BY st.name ORDER BY
        SUM(t.total + s.delivery - s.discount) DESC) Ranking
FROM customer c JOIN sale s ON c.code = s.customer_code JOIN
        (SELECT si.sale_code, SUM(si.quantity * si.price) AS total
        FROM sale_item si
        GROUP BY si.sale_code) t ON s.code = t.sale_code
        JOIN store st ON s.store_code = st.code
GROUP BY st.name, c.name
ORDER BY st.name, consumo DESC
```



Sucursal	Cliente	Consumo	Ranking
Baires I	Hipolito Yrigoyen	2735	1
Baires I	Arturo Illia	1110	2
Baires II	Arturo Frondizi	7080	1
Baires II	Bartolome Mitre	2596	2
Baires II	Arturo Illia	355	3
Libres	Arturo Illia	1010	1
Pergamino	Hipolito Yrigoyen	1289	1
Pergamino	Arturo Illia	244	2

WINDOW: Concepts Review

- Comparison between “windows” according to the change of numerical values.
- For example:
  - Price average in a 90 day window
  - Average annual growth of sales
  - Marketing campaigns performance in recent months
  - Cumulative sales performance this year

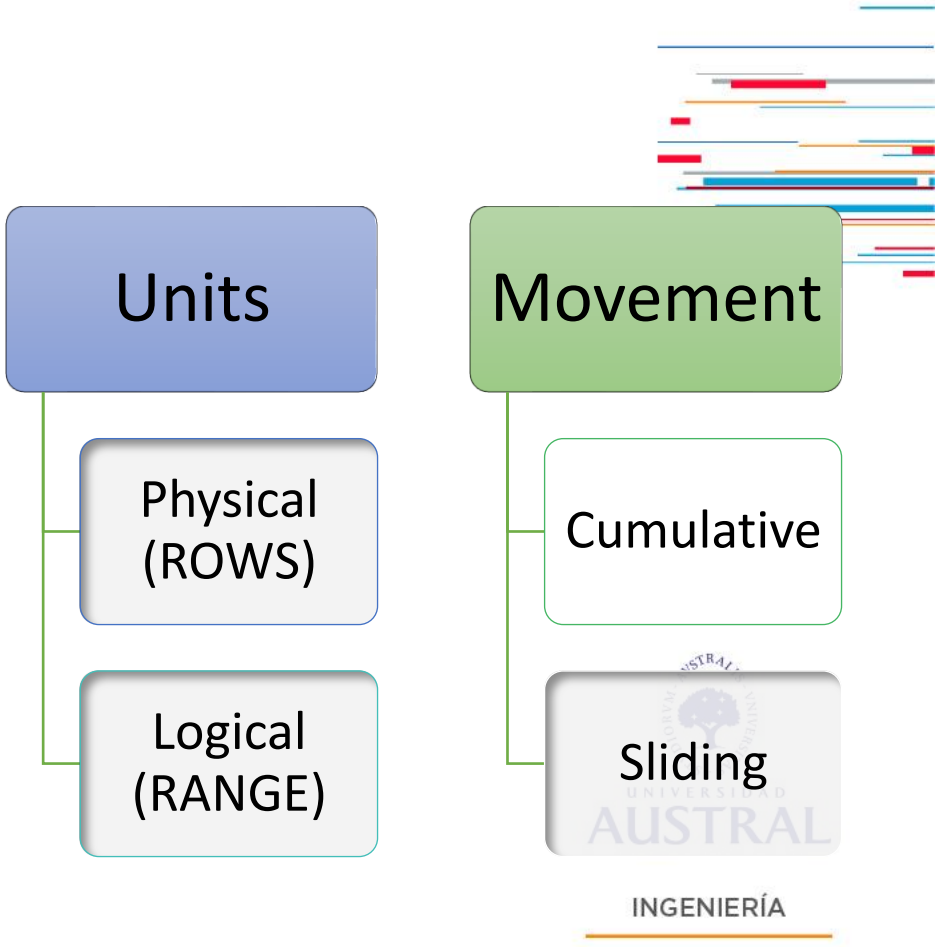


WINDOW: Basic Syntax

- <AnalyticFunction> ([<column-list>]) OVER
  ([<partitioning>]
    <ordering>
    [<window-specification>])
  - Applies to Aggregate Functions

- Physical Rows Example:

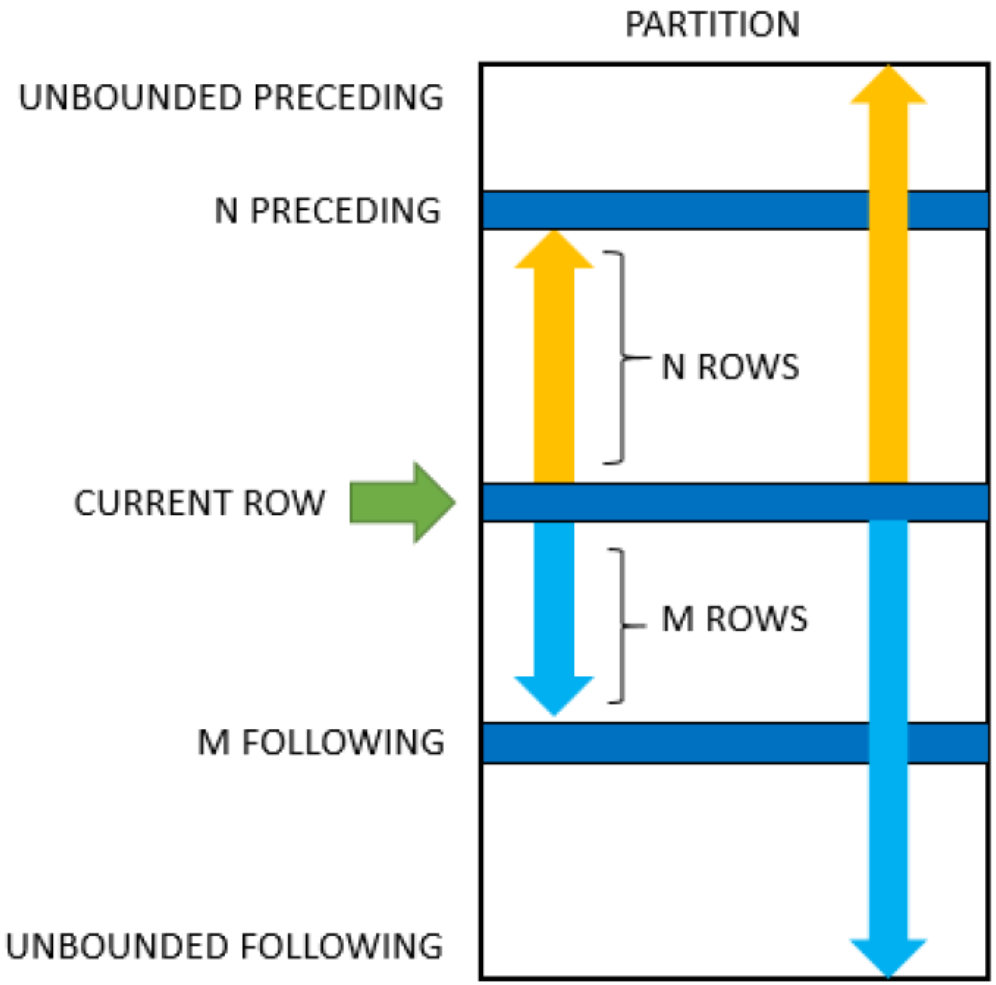
```
ROWS UNBOUNDED PRECEDING
ROWS 2 PRECEDING
ROWS 3 FOLLOWING
```



Window Syntax

```
{ RANGE | ROWS } frame_start
{ RANGE | ROWS } BETWEEN frame_start AND frame_end
```

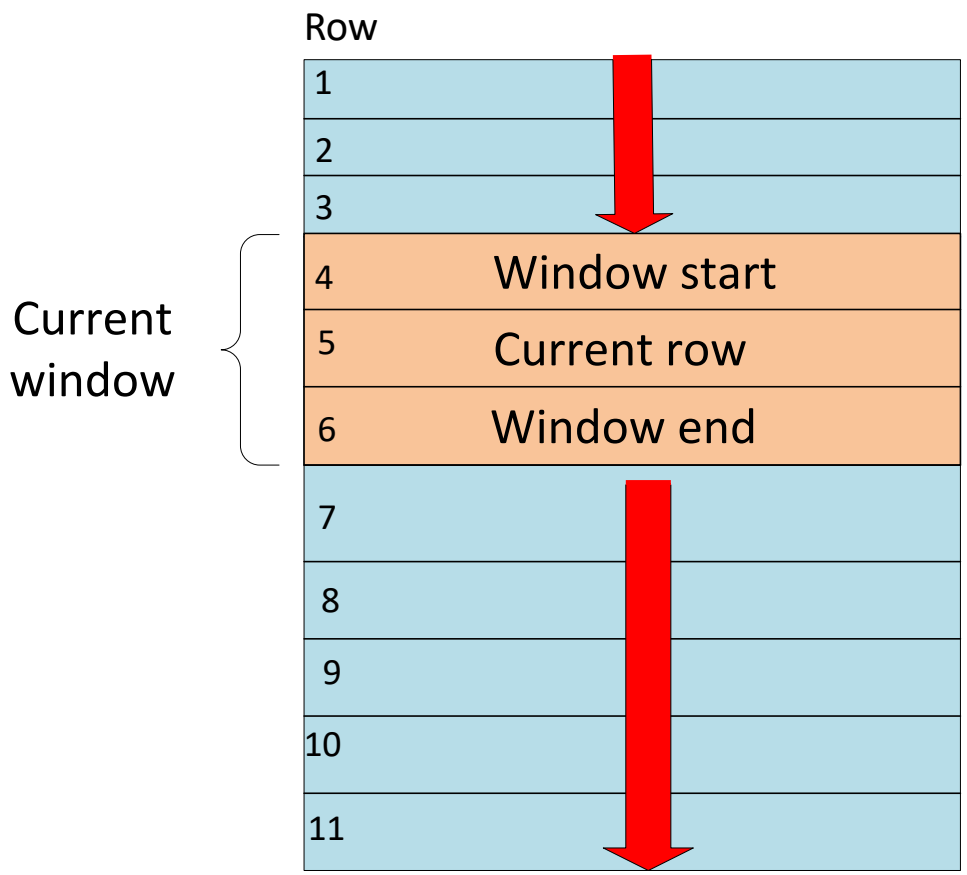
```
frame_start:
    N PRECEDING
    UNBOUNDED PRECEDING
    CURRENT ROW

frame_end:
    CURRENT ROW
    UNBOUNDED FOLLOWING
    N FOLLOWING
```



UNIVERSIDAD AUSTRAL
INGENIERÍA

Sliding, Centered Physical Window

```
ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING
```



Window Boundaries

Current Row	Window Start	Window End
1	1	2
2	1	3
3	2	4
4	3	5
5	4	6
6	5	7
7	6	8
8	7	9
9	8	10
10	9	11
11	10	11

Sliding Physical Window Example

- Moving average of sum of sales by code and date
- Centered physical window of 3 rows
- No partitioning

```
SELECT s.sale_date, s.code, SUM(t.total + s.delivery - s.discount) AS Consumo,
AVG(SUM(t.total + s.delivery - s.discount)) OVER
        (ORDER BY s.code, sale_date ROWS BETWEEN 1 PRECEDING
        AND 1 FOLLOWING) AS ConsumoPromedioMovil
FROM commerce.sale s JOIN
        (SELECT si.sale_code, SUM(si.quantity * si.price) AS total
        FROM commerce.sale_item si
        GROUP BY si.sale_code) t ON s.code = t.sale_code
GROUP BY s.code, s.sale_date
ORDER BY s.code, sale_date
```

sale_date	code	Consumo	ConsumoPro-medioMovil
1/1/2016	10	244	438
1/2/2016	11	632	366,6666667
1/2/2016	12	224	622
2/1/2016	20	1010	943,3333333
2/2/2016	21	1596	1408,333333
2/2/2016	22	1619	1441,666667
3/2/2016	30	1110	1541,666667
3/2/2016	31	1896	1357
3/2/2016	32	1065	1105,333333
4/2/2016	40	355	848,6666667
4/2/2016	41	1126	865,6666667
4/2/2016	42	1116	924,6666667
5/2/2016	50	532	982
5/2/2016	51	1298	1475,333333
5/2/2016	52	2596	1947

Cummulative Physical Window Example

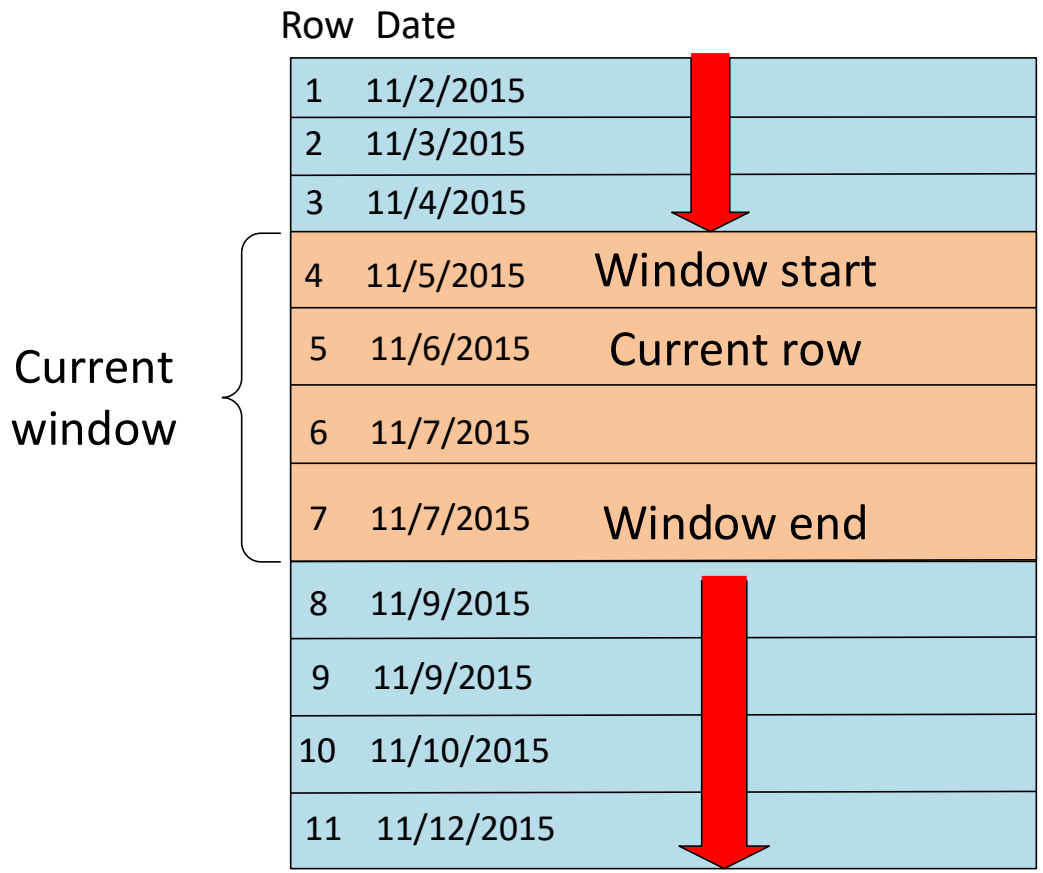
- Cummulative sales amount
- Partitioning by month

```
SELECT s.sale_date, (t.total + s.delivery - s.discount) AS Consumo,
(SUM(t.total + s.delivery - s.discount) OVER
        (PARTITION BY MONTH(sale_date) ORDER BY sale_date
        ROWS UNBOUNDED PRECEDING)) AS ConsumoAcumulado
FROM commerce.sale s JOIN
        (SELECT si.sale_code, SUM(si.quantity * si.price) AS total
        FROM commerce.sale_item si
        GROUP BY si.sale_code) t ON s.code = t.sale_code
ORDER BY sale_date
```

sale_date	code	Consumo	Consumo-Acumulado
1/1/2016	10	244	244
2/1/2016	20	1010	1254
1/2/2016	11	632	632
1/2/2016	12	224	856
2/2/2016	21	1596	2452
2/2/2016	22	1619	4071
3/2/2016	30	1110	5181
3/2/2016	31	1896	7077
3/2/2016	32	1065	8142
4/2/2016	40	355	8497
4/2/2016	41	1126	9623
4/2/2016	42	1116	10739
5/2/2016	50	532	11271
5/2/2016	51	1298	12569
5/2/2016	52	2596	15165

Sliding, Centered Logical Window

```
RANGE BETWEEN 1 PRECEDING AND 1 FOLLOWING
```



Window Boundaries

Current Row	Window Start	Window End
1	1	2
2	1	3
3	2	4
4	3	5
5	4	7
6	5	7
7	5	7
8	8	10
9	8	10
10	8	10
11	11	11

UNIVERSIDAD AUSTRAL
INGENIERÍA

Sliding Logical Window Example

- Moving average of sum of sales by day
- Centered logical window of 3 days
- No partitioning

```
SELECT s.sale_date, s.code, SUM(t.total + s.delivery - s.discount) AS Consumo,
AVG(SUM(t.total + s.delivery - s.discount)) OVER
        (ORDER BY sale_date RANGE BETWEEN 3 PRECEDING
        AND 3 FOLLOWING) AS ConsumoPromedioMovil
FROM commerce.sale s JOIN
        (SELECT si.sale_code, SUM(si.quantity * si.price) AS total
        FROM commerce.sale_item si
        GROUP BY si.sale_code) t ON s.code = t.sale_code
GROUP BY s.code, s.sale_date
ORDER BY s.code, sale_date
```

sale_date	code	Consumo	ConsumoPro-medioMovil
1/1/2016	10	244	627
1/2/2016	11	632	1073,9
1/2/2016	12	224	1073,9
2/1/2016	20	1010	627
2/2/2016	21	1596	1166,53846
2/2/2016	22	1619	1166,53846
3/2/2016	30	1110	1166,53846
3/2/2016	31	1896	1166,53846
3/2/2016	32	1065	1166,53846
4/2/2016	40	355	1166,53846
4/2/2016	41	1126	1166,53846
4/2/2016	42	1116	1166,53846
5/2/2016	50	532	1300,81818
5/2/2016	51	1298	1300,81818
5/2/2016	52	2596	1300,81818