



# ANÁLISIS Y DISEÑO LÓGICO DE SISTEMAS

Noviembre 2023

Luis E.Canales C.  
lcanales@utalca.cl

# Valores Lógicos

---

OR AND NOT  
ANY LIKE  
SOME  
EXISTS BETWEEN  
IN

# Valores Lógicos

---

- Los valores NULL no son ni Verdaderos ni Falsos

# SQL – Valores Lógicos

---

- Los datos reales a menudo tienen información faltante
- Los DBMS a menudo modelan información faltante con NULL

# SQL – Valores Lógicos

---

- FALSO = 0
- VERDADERO = 1
- DESCONOCIDO = 0.5
- precio < 25 es DESCONOCIDO cuando el precio = NULL

# SQL – Valores Lógicos

---

- Definiciones formales:

<b>C1 AND C2</b>	<b>MIN (C1, C2)</b>
<b>C1 OR C2</b>	<b>MAX (C1, C2)</b>
<b>NOT C</b>	<b>1 - C</b>

- La regla para SELECT ... FROM ... WHERE <**C**> ... Es la siguiente:  
if **C** = **TRUE**, entonces **incluye** la fila en el output  
if **C** = **FALSE** or **UNKNOWN**, entonces no las incluye.

# SQL – Valores Lógicos

---

Cuál es el output?

```
SELECT P.nombre  
      FROM Persona AS P  
      WHERE P.edad >= 21
```

Nombre	edad
Boris	19
Ema	32
Juan	NULL
NULL	24

# SQL – Valores Lógicos

---

Cuál es el output?

```
SELECT P.nombre  
  FROM Persona AS P  
 WHERE P.edad >= 21
```

Nombre	edad
Boris	19
Ema	32
Juan	NULL
NULL	24



# SQL – Valores Lógicos

---

- ¿Por qué la lógica NULL puede fallarnos?

```
SELECT P.nombre  
FROM Persona AS P  
WHERE P.edad >= 21  
      Or P.edad < 21
```

Siempre verdadero?

Nombre	edad
Boris	19
Ema	32
Juan	NULL
NULL	24

# SQL – Valores Lógicos

---

- ¿Por qué la lógica NULL puede fallarnos?

```
SELECT P.nombre  
FROM Persona AS P  
WHERE P.edad >= 21  
      Or P.edad < 21
```

NO!

Nombre	edad
Boris	19
Ema	32
Juan	NULL
NULL	24

# SQL – Valores Lógicos

---

- ¿Por qué la lógica NULL y de 3 valores puede fallarnos?

Otro caso raro

```
SELECT P.nombre  
  FROM Persona AS P  
  WHERE P.edad = P.edad
```

Nombre	edad
Boris	19
Ema	32
Juan	NULL
NULL	24

# Agregación

---

- No utilices el SQL, sólo haz el cálculo mental:

- ¿Cuál es el salario mínimo global?
- ¿Cuál es el salario mínimo de un contador?
- ¿Cuál es el salario mínimo para cada puesto de trabajo?

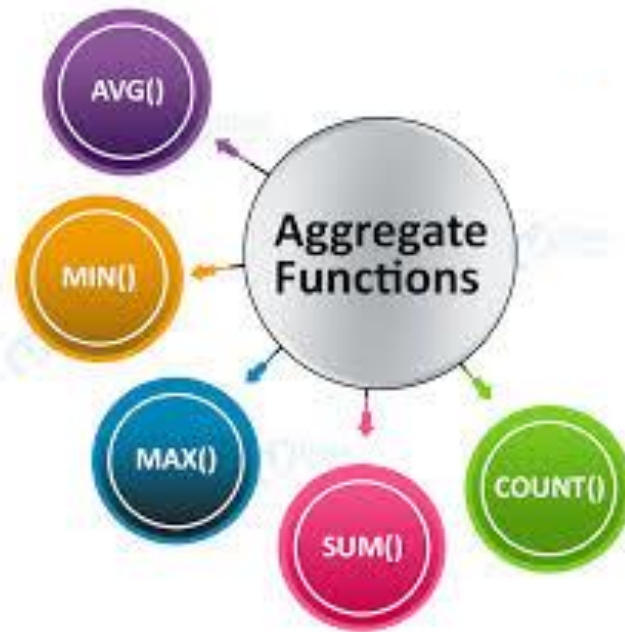
UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000
012	Javiera	Soporte	250000
234	Nora	Investigador	2000000

- ¿Qué hace este fragmento de código?

```
sum = 0
For each row in Sueldos:
    If row.Trabajo == 'Contador':
        sum = sum + row.Salario
output(sum)
```

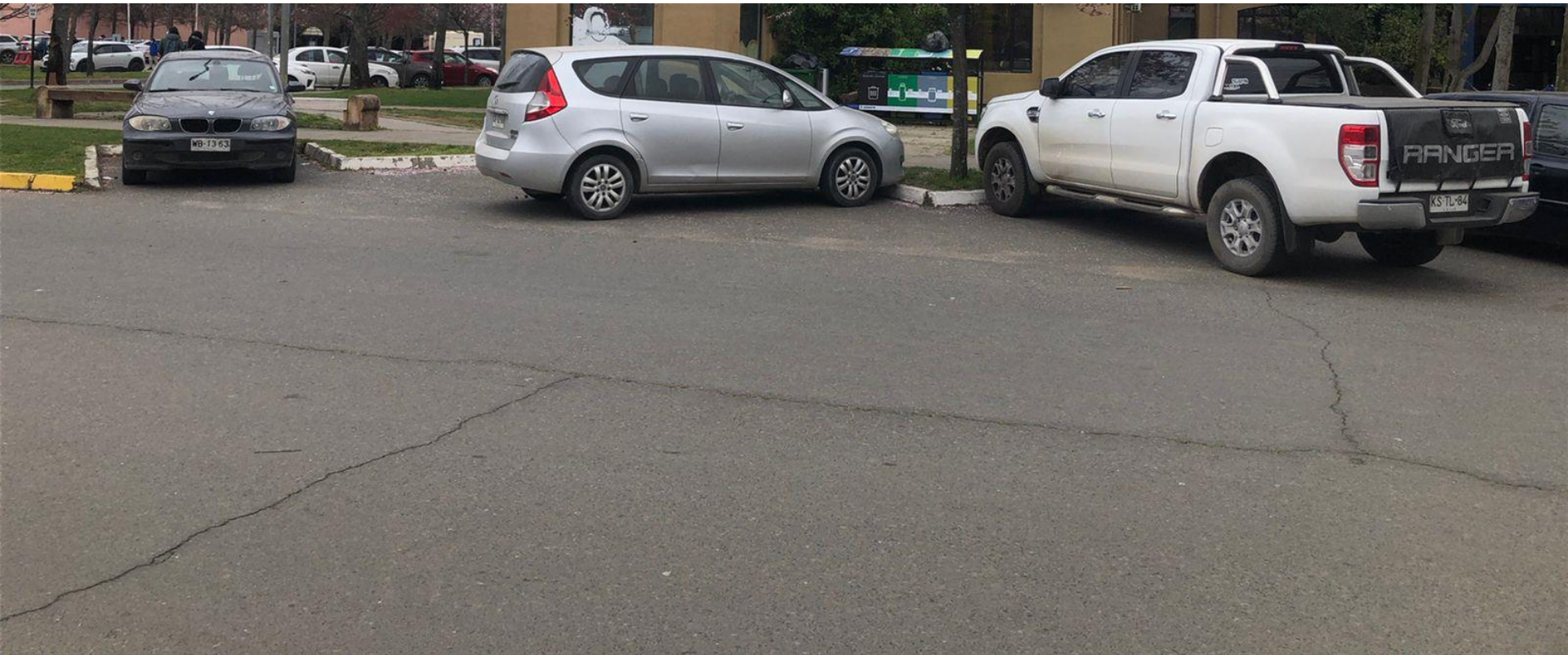
# Funciones de agregación

---









# Resumen - Join

---

- Agregados
  - **Funciones de agregación**
  - GROUP BY
  - Cláusulas WHERE vs HAVING en SQL





# Y ahora algo completamente diferente...

---

- Hasta ahora, hemos hablado de consultas simples sobre tablas unidas (Join)
- Ahora vamos a ver las consultas complejas que resumen en una sola tabla

# Resultados prácticos

---

- Las agregaciones permiten una clase diferente de consultas y resultados SQL-
  - A menudo necesitamos resúmenes de datos porque necesitamos tomar decisiones y transmitir sucintamente información.

• "¿Qué tan popular es esta serie de Netflix?"	COUNT
• "¿Gasto demasiado en café mensualmente?"	SUM
• "¿Estoy siendo estafado por este distribuidor?"	AVG
• "¿Quién obtuvo la calificación más alta en la clase?"	MAX
• "¿Cuál es la comida más barata en el casino?"	MIN

Agregados muy comunes  
"que se encuentran en los SGBD"

# Resultados prácticos

---

- Estas son las 5 funciones básicas de agregación
- Algunas bases de datos soportan mucho más
  - stddev , var, checksum.

• "¿Qué tan popular es esta serie de Netflix?"	COUNT
• "¿Gasto demasiado en café mensualmente?"	SUM
• "¿Estoy siendo estafado por este distribuidor?"	AVG
• "¿Quién obtuvo la calificación más alta en la clase?"	MAX
• "¿Cuál es la comida más barata en el casino?"	MIN

# Resultados prácticos

---

- AGG( attr ) opera sobre todos los valores no NULL
  - AGG(DISTINCT attr) también es posible
- ¡Excepción!
  - COUNT(\*) cuenta todas las filas, independientemente de los campos NULL

SQL Query	SQL Op
SELECT <b>COUNT</b> (*) FROM PeliculasNetflix...	COUNT
SELECT <b>SUM</b> (cost) FROM ConsumoCafe...	SUM
SELECT <b>AVG</b> (cost) FROM ConsumoCafe...	AVG
SELECT <b>MAX</b> (score) FROM NotasAsignatura...	MAX
SELECT <b>MIN</b> (price) FROM PrecioAlmuerzo...	MIN

# Ejemplos

---

- ¿Cuál es el resultado de cada una de estas consultas?

UserID	N o m b r e	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000
012	Javiera	Soporte	250000
234	Nora	Investigador	2000000

```
SELECT SUM(salario)  
FROM Sueldos;
```

```
SELECT MIN(salario)  
FROM Sueldos;
```

```
SELECT MIN(salario)  
FROM Sueldos  
WHERE trabajo = 'Contador';
```

# Ejemplos

- ¿Cuál es el resultado de cada una de estas consultas?

UserID	N o m b r e	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000
012	Javiera	Soporte	250000
234	Nora	Investigador	2000000

```
SELECT COUNT(*)  
FROM Sueldos;
```

```
SELECT COUNT(trabajo)  
FROM Sueldos;
```

```
SELECT COUNT(DISTINCT trabajo)  
FROM Sueldos  
WHERE trabajo = 'Contador';
```

```
SELECT COUNT(salario)  
FROM Sueldos  
WHERE trabajo = 'Contador';
```

# Agregados

---

- Agregados
  - Funciones de agregación
  - **GROUP BY**
  - Cláusulas WHERE vs HAVING en SQL



# Utilización de los grupos en la agregación

---

- Calcule el salario medio de cada empleo

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000
012	Javiera	Soporte	250000
234	Nora	Investigador	2000000

```
SELECT AVG(Salario)
FROM Sueldos
WHERE trabajo = 'Contador';

SELECT AVG(Salario)
FROM Sueldos
WHERE trabajo = 'Profesora';

SELECT AVG(Salario)
FROM Sueldos
WHERE ...

...
```



# Utilización de los grupos en la agregación

- **GROUP BY** nos permite escribir una sola consulta!

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000
012	Javiera	Soporte	250000
234	Nora	Investigador	2000000

```
SELECT trabajo, AVG(Salario)
FROM Sueldos
GROUP BY trabajo;
```

```
SELECT AVG(Salario)
FROM Sueldos

WHERE trabajo = 'Contador';
SELECT AVG(Salario)
FROM Sueldos
WHERE trabajo = 'Profesora';

SELECT AVG(Salario)
FROM Sueldos
WHERE ...

...
```

# Utilización de los grupos en la agregación

---

- **GROUP BY** nos permite escribir una sola consulta!

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000
012	Javiera	Soporte	250000
234	Nora	Investigador	2000000

```
SELECT trabajo, AVG(Salario)
FROM Sueldos
GROUP BY trabajo;
```

# Utilización de los grupos en la agregación

- Los grupos particionan los datos en función de los valores de las columnas que coinciden antes de aplicar la agregación

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000
012	Javiera	Soporte	250000
234	Nora	Investigador	2000000

```
SELECT trabajo, AVG(Salario)
FROM Sueldos
GROUP BY trabajo;
```

Trabajo	AVG(Salario)
Contador	550000
Profesora	950000
Soporte	250000
Investigador	2000000

# GROUP BY: Orden de operaciones

---

1. Agrupar tuplas por valor
2. Aplicar la agregación a cada grupo

UserID	N o m b r e	Trabajo	Salar io
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000
012	Javiera	Soporte	250000
234	Nora	Investigador	2000000

# GROUP BY: Orden de operaciones

1. Agrupar tuplas por valor
2. Aplicar la agregación a cada grupo

UserID	N o m b r e	Trabajo	Salar io
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000
012	Javiera	Soporte	250000
234	Nora	Investigador	2000000



Trabajo	Salar io
Contador	500000
Contador	600000

Trabajo	Salar io
Profesora	900000
Profesora	1000000

Trabajo	Salar io
Soporte	250000

Trabajo	Salar io
Investigador	2000000

# GROUP BY: Orden de operaciones

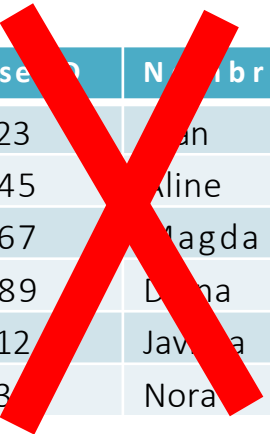
1. Agrupar tuplas por valor
2. Aplicar la agregación a cada grupo



Cada grupo se convirtió en una fila en la respuesta

# GROUP BY: Campos retenidos

- Los campos utilizados en la agregación se conservan
  - Todos los demás campos se descartan



Usuario	Nombre	Trabajo	Salario
123	Alan	Contador	500000
345	Weline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000
012	Javiera	Soporte	250000
234	Nora	Investigador	2000000

Trabajo	Salario
Contador	500000
Contador	600000

Trabajo	Salario
Profesora	900000
Profesora	1000000

Trabajo	Salario
Soporte	250000

Trabajo	Salario
Investigador	2000000

```
SELECT trabajo, AVG(Salario)
FROM Sueldos
GROUP BY trabajo;
```

# GROUP BY: Campos retenidos

- Los campos utilizados en la agregación se conservan
  - Todos los demás campos se descartan y no pueden aparecer en la cláusula SELECT

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000



Trabajo	AVG(Salario)
Contador	550000
Profesora	9500000

¿Qué nombre elegir para este grupo?



# GROUP BY: Campos retenidos

---

- ¿Cuál es el resultado de esta consulta?

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000
012	Aline	Contador	700000
234	Aline	Contadoa	800000

```
SELECT Nombre, MAX(Salario)
FROM Sueldos
GROUP BY Nombre;
```

Nombre	MAX(Salario)
Juan	500000
Aline	800000
Magda	900000
Diana	1000000

# GROUP BY: Campos retenidos

---

- ¿Cuál es el resultado de esta consulta?

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000
012	Aline	Contador	700000
234	Aline	Contador	800000

```
SELECT Nombre, MAX(Salario)
FROM Sueldos
GROUP BY Nombre;
```

Nombre	MAX(Salario)
Juan	500000
Aline	800000
Magda	900000
Diana	1000000

# GROUP BY: Múltiples columnas

- Los grupos dividen los datos en función de los valores de las columnas que coinciden antes de aplicar la agregación

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000
012	Aline	Contador	700000
234	Aline	Contador	800000

```
SELECT Trabajo, Nombre, MAX(Salario)
FROM Sueldos
GROUP BY Trabajo, Nombre;
```

Trabajo	Nombre	MAX(Salario)
Contador	Juan	500000
Contador	Aline	800000
Profesora	Magda	900000
Profesora	Diana	1000000

# Agregados

---

- Agregados
  - Funciones de agregación
  - GROUP BY
  - Cláusulas WHERE vs HAVING en SQL



# Modelo mental de una Query (Consulta)

## FWGHOS

**SELECT...**

**FROM...**

**WHERE...**

**GROUP BY...**

**HAVING...**

**ORDER BY**

```
SELECT Trabajo, MAX(Salario)
FROM Sueldos
GROUP BY Salario
HAVING MIN(Salario) > 55000
ORDER BY Trabajo;
```

**SELECT...**



**ORDER BY**



**HAVING...**



**GROUP BY...**



**WHERE...**



**FROM...**

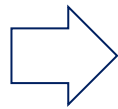
Tablas/Relaciones

# Utilización de los grupos en la agregación



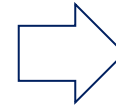
# Utilización de los grupos en la agregación

Nombre	Valor
A	10
A	20
B	40
C	20
C	50



$\Sigma$

```
SELECT nombre, SUM(Valor)
FROM Tabla
GROUP BY Nombre
```



Nombre	SUM(Valor)
A	30
B	40
C	70



# WHERE: Filtrar tuplas

---

- Algunos grupos se vacían si hay una cláusula cláusula **WHERE**
  - Los grupos vacíos desaparecen por completo
  - Nunca tendrán  $\text{count}(*)=0$



# WHERE: Filtrar tuplas

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000
012	Aline	Contadora	700000
234	Aline	Contadora	800000

```
SELECT Trabajo, AVG(Salario)
FROM Sueldos
WHERE Salario < 600000
GROUP BY Trabajo;
```

**FW**GHOS