

# Curso SQL

---

Nelson López Centeno

# Curso SQL

---

- ☐ Consultas - SELECT
  - ☐ Combinar - JOIN
  - ☐ Anexar - UNION
- ☐ Subconsultas y CTE
- ☐ Funciones de ventana
- ☐ Modificar Datos - INSERT, UPDATE, DELETE
- ☐ Transacciones
- ☐ Modificar Estructura - CREATE, ALTER, DROP
  - ☐ Procedimientos almacenados
  - ☐ Triggers

# Configuración del entorno de trabajo



MySQL

<https://dev.mysql.com/downloads/mysql/>



DBBeaver

<https://dbeaver.io/download/>

# SELECT

---

**SELECT DISTINCT  
FROM  
WHERE  
GROUP BY  
HAVING  
ORDER BY  
LIMIT**

**Orden de ejecución**

**FROM  
WHERE  
GROUP BY  
HAVING  
SELECT  
DISTINCT  
ORDER BY  
LIMIT**

# TIPOS DATOS

---

## Numéricos

**INT, SMALLINT, TINYINT, BIGINT**

**DECIMAL, NUMERIC**

**FLOAT, REAL**

## Cadenas Caracteres

**CHAR**

**VARCHAR**

**TEXT**

## Fecha y Hora

**DATE**

**TIME**

**DATETIME**

## Binario

**BINARY**

**VARBINARY**

**BLOB**

# FUNCIONES DE TEXTO

---

**UPPER, LOWER**

**LEFT, RIGHT**

**LENGTH**

**TRIM, LTRIM, RTRIM, BTRIM**

**SUBSTRING**

**CONCAT**

# **FUNCIONES DE FECHA Y HORA**

---

**CURRENT\_DATE**

**CURRENT\_TIME**

**CURRENT\_TIMESTAMP**

**DATEADD**

**DATEDIFF**

**DATEPART**

# FILTRADO - WHERE

---

**=, <>, <, >, <=, >=**

**AND, OR, NOT**

**BETWEEN ... AND**

**IN (...)**

**LIKE**



# CASE

---

**CASE exp**

**WHEN valor1 THEN res1**

**WHEN valor2 THEN res2**

**WHEN valor3 THEN res3**

**ELSE res4**

**END**

**CASE**

**WHEN cond1 THEN res1**

**WHEN cond2 THEN res2**

**WHEN cond3 THEN res3**

**ELSE res4**

**END**

# FUNCIONES DE AGREGACIÓN

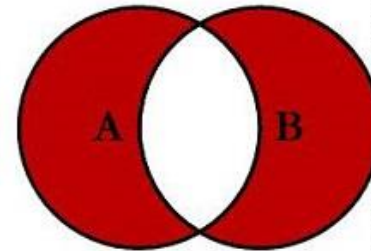
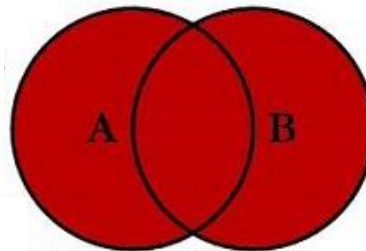
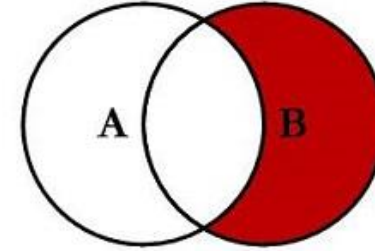
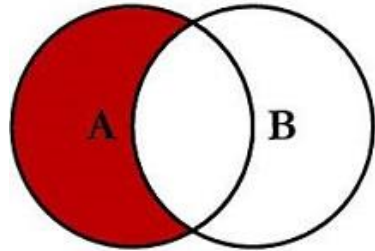
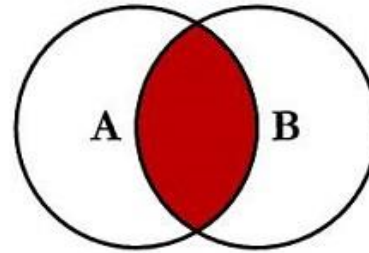
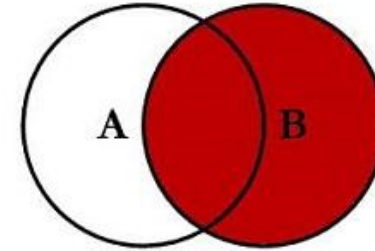
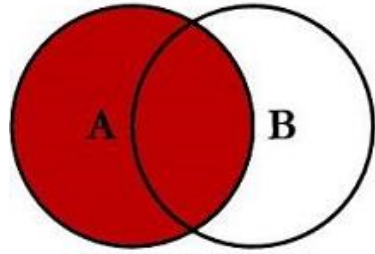
---

**COUNT**  
**SUM**  
**MIN, MAX**  
**AVG**

**Estas funciones se utilizan  
frecuentemente en  
combinación con  
GROUP BY y con  
HAVING**

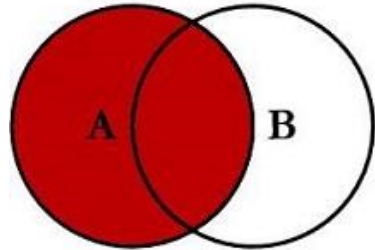
# JOIN

## SQL JOINS

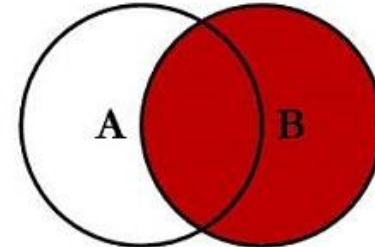


# JOIN

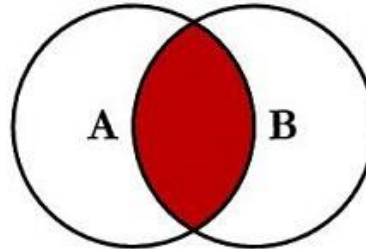
## SQL JOINS



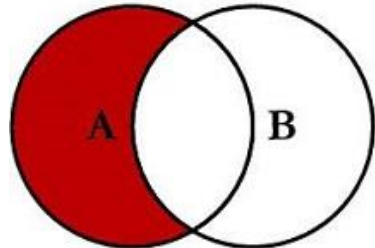
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key
```



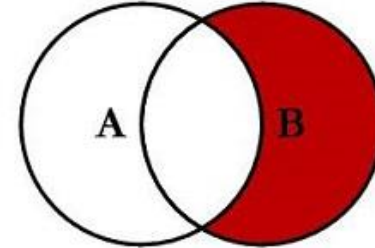
```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key
```



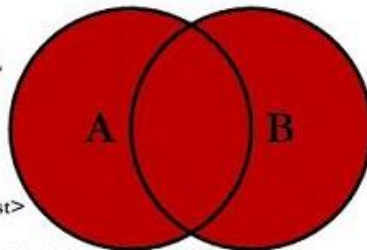
```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```



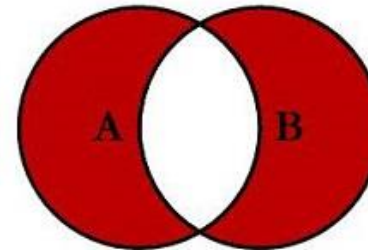
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key  
WHERE B.Key IS NULL
```



```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key
```

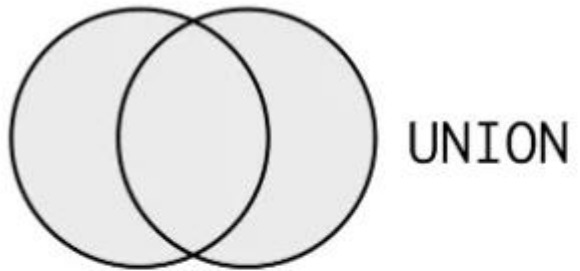


```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL  
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

# UNION

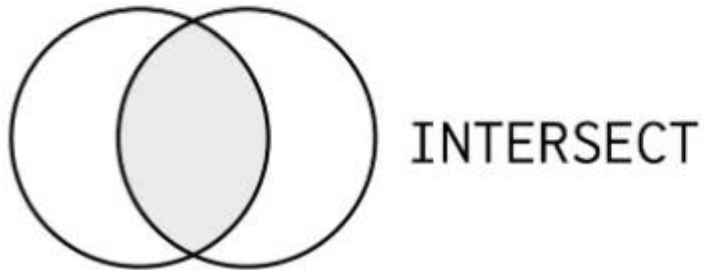
---



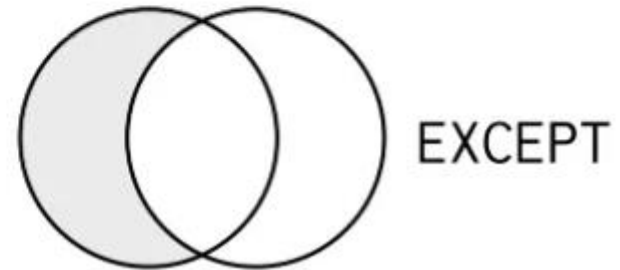
UNION



UNION ALL



INTERSECT



EXCEPT

# SUBCONSULTA

---

## Filtrar una consulta basándose en otra

```
SELECT c1
FROM t1
WHERE
    c2 > (
        SELECT AVG(c3)
        FROM t2
    )
```

```
SELECT c1
FROM t1
WHERE
    c2 IN (
        SELECT c3
        FROM t2
        WHERE c4 > 200
    )
```

# SUBCONSULTA

---

## Crear nuevas columnas a partir de agregaciones

```
SELECT  
    c1,  
    (  
        SELECT AVG(c3)  
        FROM t2  
    ) AS c2  
FROM t1
```

# SUBCONSULTA

---

## Consultas correlacionadas

```
SELECT
    c1,
    c2
FROM t1
WHERE c2 = (
    SELECT MAX(c3)
    FROM t2
    WHERE t1.c1 = t2.c4
)
```



# CTE: Common Table Expression

---

## Organizar consultas complejas

```
WITH cte1 AS (  
    SELECT  
        c1,  
        SUM(c2)  
    FROM t1  
    GROUP BY c1  
)
```

```
SELECT  
    ...  
FROM cte1  
...  
  
SELECT  
    ...  
FROM cte1
```

# CTE: Common Table Expression

---

## Consulta recursiva

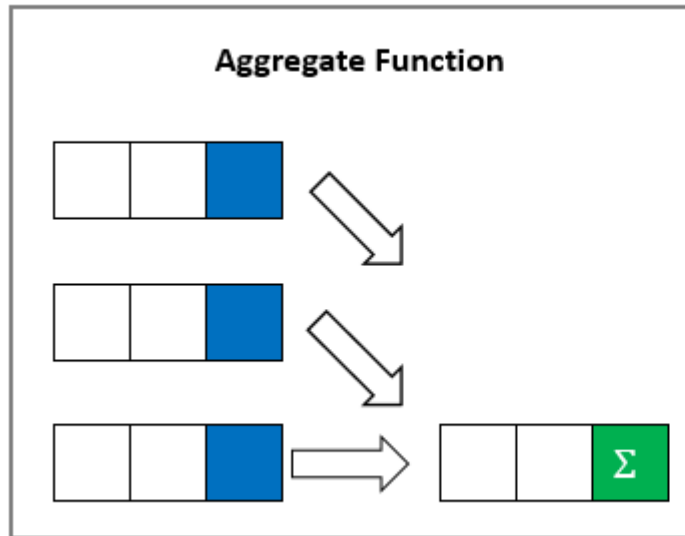
```
WITH cte_recursiva AS (  
    SELECT  
        c1  
    FROM t1  
    INNER JOIN cte_recursiva  
    ...  
)
```

```
SELECT  
    ...  
FROM cte_recursiva  
...
```

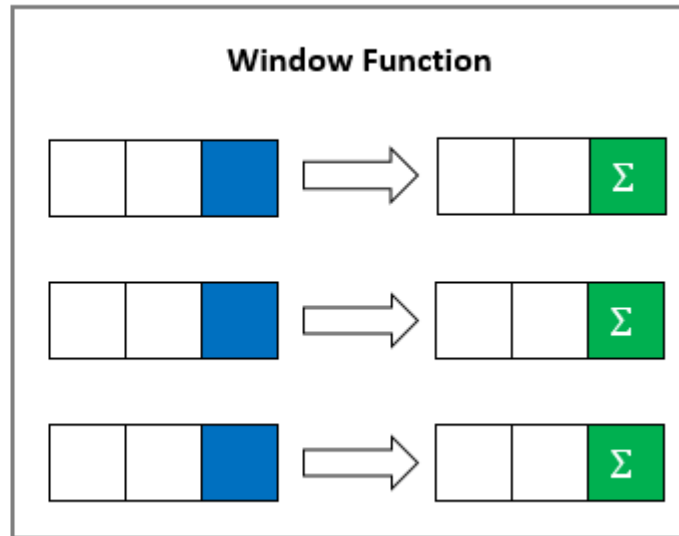
# Funciones Ventana

---

```
SELECT  
  c1,  
  c2,  
  SUM(c3)  
FROM T1  
GROUP BY c1,c2
```



```
SELECT  
  c1,  
  c2,  
  SUM(c3) OVER ()  
FROM T1
```



# Funciones Ventana

```
SELECT
    spend,
    SUM(spend) OVER (PARTITION BY product ORDER BY transaction_date) AS running_total
FROM product_spend;
```

Output

<b>ORDER BY</b> transaction_date	<b>PARTITION BY</b> product	<b>SUM(spend)</b> spend	running_total
10/07/2022 16:00:00	3.5mm headphone jack	7.99	7.99
07/01/2022 11:00:00	computer mouse	45.00	45.00
08/28/2022 16:00:00	microwave	49.99	49.99
03/01/2023 17:00:00	microwave	34.49	84.48
07/08/2023 16:00:00	microwave	64.95	149.43
12/26/2021 12:00:00	refrigerator	246.00	246.00
03/02/2022 11:00:00	refrigerator	299.99	545.99

# Funciones Ventana

---

**OVER** se usa con las funciones de agregación:

**COUNT(), SUM(), AVG(), MIN(), MAX()**

y además con:

**FIRST\_VALUE()**

**LAST\_VALUE()**

**ROW\_NUMBER()**

**RANK()**

**DENSE\_RANK()**

**LEAD()**

**LAG()**

# Modificar Datos

---

Al modificar los datos se tienen que **cumplir** las **restricciones** definidas en la base de datos.

- Claves primarias
- Claves foráneas
- Columnas identidad
- Columnas con valores únicos
- Columnas que no admiten NULL
- ...

# Modificar Datos - **INSERT**

---

**INSERT INTO T1 (c1, c2)**

**VALUES**

**(v1, v2),**

**(v3, v4),**

**...**

**INSERT INTO T1 (c1, c2)**

**SELECT**

**c3, c4**

**FROM T2**

**WHERE**

**c5 > 10**

# Modificar Datos - UPDATE

---

```
UPDATE T1
SET
    c1 = v1,
    c2 = v2
WHERE
    c3 > 10
```

```
UPDATE T1
SET c1 = (
    SELECT
        c2
    FROM T2
    WHERE
        c3 = T1.c4
)
```



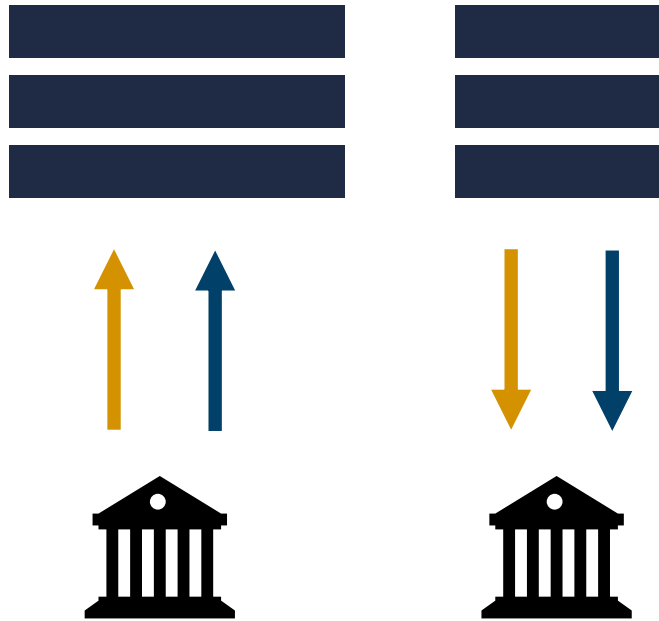
# Modificar Datos - **DELETE**

---

```
DELETE FROM T1  
WHERE  
    c3 > 10
```

# Transacciones - **ACID**

---



**Atomicidad**

**Consistencia**

**Aislamiento**

**Durabilidad**

# Transacciones - ACID

---

**Atomicidad:** Garantiza que cada transacción sea tratada como una sola unidad que tiene éxito por completo, o bien fracasa por completo.

**Consistencia:** (Integridad) Garantiza que una transacción solo pueda llevar los datos de la base de datos de un estado válido a otro.

**Aislamiento:** Asegura que la ejecución simultánea de transacciones deja la base de datos en el mismo estado que se habría obtenido si las transacciones se hubieran ejecutado secuencialmente.

**Durabilidad:** Garantiza que una vez que una transacción se confirma, seguirá confirmada incluso si hay un error del sistema, como puede ser un corte de energía o un bloqueo.

# Transacciones explícitas

---

**START TRANSACTION;**

**COMMIT;**

**ROLLBACK;**

# Modificar Estructura

---

**CREATE DATABASE BD1**

**ALTER DATABASE BD1 MODIFY NAME BD1**

**DROP DATABASE BD2**

# Modificar Estructura

---

```
CREATE TABLE T1 (  
    ID int NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    ID2 int NOT NULL,  
    C1 varchar(50) NOT NULL,  
    C2 varchar(50) NULL,  
    C3 int UNIQUE,  
    FOREIGN KEY (ID2) REFERENCES T2(ID)  
);
```

```
ALTER TABLE T1 ADD C4 int
```

```
DROP TABLE T1
```