MySQL - Cheatsheet Completo

📋 ÍNDICE RÁPIDO

SELECT BÁSICO

sql

```sql
-- Seleccionar columnas específicas
SELECT columna1, columna2 FROM tabla;

-- Seleccionar todas las columnas
SELECT * FROM tabla;

-- Ejemplo
SELECT id, nombre FROM empleados;
```

WHERE Y FILTRADO

```sql
-- Sintaxis básica
SELECT columnas FROM tabla WHERE condición;

-- Ejemplo
SELECT * FROM empleados WHERE salario > 3000;
```

OPERADORES DE COMPARACIÓN

Operador	Descripción	Ejemplo
`=`	Igual a	`WHERE edad = 30`
`!=` o `<>`	Diferente de	`WHERE ciudad != 'Madrid'`
`>`	Mayor que	`WHERE salario > 4000`
`<`	Menor que	`WHERE edad < 40`
`>=`	Mayor o igual	`WHERE salario >= 3000`
`<=`	Menor o igual	`WHERE edad <= 50`

```sql
-- Ejemplos
SELECT * FROM empleados WHERE salario > 4000;
SELECT * FROM clientes WHERE ciudad = 'Madrid';
SELECT * FROM productos WHERE precio <= 100;
```

OPERADORES LÓGICOS

AND - Ambas condiciones deben cumplirse

```sql
SELECT * FROM empleados
WHERE salario > 3000 AND edad < 40;
```

OR - Al menos una condición debe cumplirse

```sql
SELECT * FROM empleados
WHERE departamento = 'Ventas' OR departamento = 'Marketing';
```

NOT - Niega la condición

```sql
SELECT * FROM empleados
WHERE NOT departamento = 'Recursos Humanos';
```

Combinar con paréntesis

```sql
SELECT * FROM empleados
WHERE departamento = 'Ventas' AND (edad > 30 OR salario > 5000);
```

LIKE - PATRONES

Comodín	Descripción
%	Cualquier número de caracteres (incluso cero)
_	Un solo carácter

sql

```sql
-- Nombres que comienzan con 'M'
SELECT * FROM empleados WHERE nombre LIKE 'M%';

-- Nombres que terminan con 'ez'
SELECT * FROM clientes WHERE apellido LIKE '%ez';

-- Nombres que contienen 'an' en cualquier posición
SELECT * FROM empleados WHERE nombre LIKE '%an%';

-- Nombres de 4 letras que terminan en 'o'
SELECT * FROM empleados WHERE nombre LIKE '___o';
```

IN - LISTA DE VALORES

sql

```sql
-- Filtrar por lista de IDs
SELECT * FROM empleados WHERE id IN (1, 3, 5, 7);

-- Filtrar por múltiples departamentos
SELECT * FROM empleados
WHERE departamento IN ('Ventas', 'Finanzas', 'IT');
```

BETWEEN - RANGOS

sql

```sql
-- Rango de salarios (incluye los límites)
SELECT * FROM empleados
WHERE salario BETWEEN 2500 AND 5000;

-- Rango de fechas
SELECT * FROM pedidos
WHERE fecha_pedido BETWEEN '2023-01-01' AND '2023-12-31';
```

DISTINCT - VALORES ÚNICOS

```sql
-- Eliminar duplicados
SELECT DISTINCT departamento FROM empleados;

-- Con WHERE
SELECT DISTINCT departamento FROM empleados
WHERE salario > 3000;

-- Con múltiples columnas (combinación única)
SELECT DISTINCT nombre, departamento FROM empleados
WHERE salario > 4000;
```

IS NULL / IS NOT NULL

```sql
-- Buscar valores nulos
SELECT * FROM empleados WHERE telefono IS NULL;

-- Buscar valores no nulos
SELECT * FROM empleados WHERE telefono IS NOT NULL;
```

ORDER BY - ORDENACIÓN

```sql
-- Orden ascendente (predeterminado)
SELECT * FROM empleados ORDER BY salario ASC;

-- Orden descendente
SELECT * FROM empleados ORDER BY salario DESC;

-- Múltiples columnas
SELECT * FROM empleados
ORDER BY departamento ASC, salario DESC;

-- Por campo calculado
SELECT nombre, CHAR_LENGTH(nombre) AS longitud
FROM empleados
ORDER BY longitud DESC;

-- Manejar valores NULL (al final)
SELECT * FROM clientes
ORDER BY telefono IS NULL, telefono;
```

LIMIT - LIMITAR RESULTADOS

```sql
-- Sintaxis
SELECT columnas FROM tabla
ORDER BY columna
LIMIT cantidad_filas [OFFSET inicio];

-- Los 5 salarios más altos
SELECT * FROM empleados
ORDER BY salario DESC
LIMIT 5;

-- Paginación: registros del 6 al 10
SELECT * FROM empleados
ORDER BY fecha_ingreso DESC
LIMIT 5 OFFSET 5;
```

FUNCIONES DE AGREGACIÓN

Función	Descripción	Ejemplo
`COUNT()`	Cuenta registros	`SELECT COUNT(*) FROM empleados;`
`SUM()`	Suma valores	`SELECT SUM(salario) FROM empleados;`
`AVG()`	Promedio	`SELECT AVG(salario) FROM empleados;`
`MAX()`	Valor máximo	`SELECT MAX(salario) FROM empleados;`
`MIN()`	Valor mínimo	`SELECT MIN(salario) FROM empleados;`

sql

```sql
-- Contar empleados
SELECT COUNT(*) AS total_empleados FROM empleados;

-- Salario promedio con filtro
SELECT AVG(salario) AS salario_promedio
FROM empleados
WHERE edad > 30;

-- Salario máximo
SELECT MAX(salario) AS salario_maximo FROM empleados;

-- Suma total
SELECT SUM(salario) AS total_salarios FROM empleados;
```

GROUP BY - AGRUPACIÓN

```sql
-- Agrupar y contar
SELECT departamento, COUNT(*) AS cantidad_empleados
FROM empleados
GROUP BY departamento;

-- Múltiples agregaciones
SELECT departamento,
    COUNT(*) AS cantidad,
    AVG(salario) AS salario_promedio,
    MAX(salario) AS salario_max
FROM empleados
GROUP BY departamento;

-- Agrupar por múltiples columnas
SELECT departamento, ciudad, COUNT(*) AS total
FROM empleados
GROUP BY departamento, ciudad;
```

HAVING - FILTRAR GRUPOS

```sql
-- Filtrar grupos después de agregación
SELECT departamento, COUNT(*) AS cantidad_empleados
FROM empleados
GROUP BY departamento
HAVING COUNT(*) > 5;

-- Con promedio
SELECT edad, AVG(salario) AS salario_promedio
FROM empleados
GROUP BY edad
HAVING AVG(salario) > 3000;

-- Suma de ventas por cliente
SELECT cliente_id, SUM(importe) AS total_ventas
FROM ventas
GROUP BY cliente_id
HAVING SUM(importe) > 10000;
```

FUNCIONES DE TEXTO

CONCAT() - Concatenar cadenas

```sql
SELECT CONCAT(nombre, ' ', apellido) AS nombre_completo
FROM empleados;

-- Manejar NULL
SELECT CONCAT(COALESCE(nombre, 'Desconocido'), ' ', COALESCE(apellido, ''))
AS nombre_completo
FROM empleados;
```

SUBSTRING() - Extraer subcadena

```sql
-- Primeros 3 caracteres
SELECT SUBSTRING(nombre, 1, 3) FROM empleados;

-- Últimos 4 caracteres
SELECT SUBSTRING(nombre, CHAR_LENGTH(nombre)-3) FROM empleados;
```

CHAR_LENGTH() - Longitud de cadena

```sql
SELECT nombre, CHAR_LENGTH(nombre) AS longitud
FROM empleados;
```

REPLACE() - Reemplazar texto

```sql
SELECT REPLACE(nombre, 'Juan', 'Carlos') FROM empleados;
```

TRIM(), LTRIM(), RTRIM() - Eliminar espacios

```sql
-- Ambos lados
SELECT TRIM('   Hola Mundo   '); -- 'Hola Mundo'

-- Izquierda
SELECT LTRIM('   Hola'); -- 'Hola'

-- Derecha
SELECT RTRIM('Hola   '); -- 'Hola'

-- Caracteres específicos
SELECT TRIM(LEADING '0' FROM '000123'); -- '123'
```

FUNCIONES NUMÉRICAS

ROUND() - Redondear

```sql
-- A entero más cercano
SELECT ROUND(15.7); -- 16

-- A 2 decimales
SELECT ROUND(salario, 2) FROM empleados;
```

FLOOR() - Redondear hacia abajo

```sql
SELECT FLOOR(15.7); -- 15
```

CEIL() - Redondear hacia arriba

```sql
SELECT CEIL(15.3); -- 16
```

RAND() - Número aleatorio

```sql
-- Entre 0 y 1
SELECT RAND();

-- Entre 1 y 100
SELECT FLOOR(1 + (RAND() * 100));
```

FUNCIONES DE FECHA

NOW() - Fecha y hora actual

```sql
SELECT NOW();  -- '2025-01-20 14:30:45'
```

DATE_FORMAT() - Formatear fecha

```sql
-- Formato personalizado
SELECT DATE_FORMAT(fecha_ingreso, '%d/%m/%Y') FROM empleados;

-- Formato completo
SELECT DATE_FORMAT(NOW(), '%d/%m/%Y %H:%i:%s');
```

Símbolos de formato:

- `%d` - Día (2 dígitos)
- `%m` - Mes (2 dígitos)
- `%Y` - Año completo
- `%H` - Hora (24 horas)
- `%i` - Minutos
- `%s` - Segundos

DATEDIFF() - Diferencia en días

```sql
SELECT DATEDIFF(NOW(), fecha_ingreso) AS antiguedad
FROM empleados;
```

EXTRACT() - Extraer parte de fecha

```sql
-- Extraer año
SELECT EXTRACT(YEAR FROM fecha_ingreso) FROM empleados;

-- Extraer mes
SELECT EXTRACT(MONTH FROM fecha_ingreso) FROM empleados;

-- Extraer día
SELECT EXTRACT(DAY FROM fecha_ingreso) FROM empleados;
```

SUBCONSULTAS

Subconsulta Escalar (un valor)

```sql
-- Empleados con salario mayor al promedio
SELECT nombre
FROM empleados
WHERE salario > (SELECT AVG(salario) FROM empleados);

-- Empleado con salario máximo
SELECT nombre
FROM empleados
WHERE salario = (SELECT MAX(salario) FROM empleados);
```

Subconsulta de Columna (múltiples valores)

```sql
-- Con IN
SELECT nombre
FROM empleados
WHERE departamento IN (SELECT departamento FROM departamentos);
```

Subconsulta de Tabla

```sql
-- En FROM
SELECT *
FROM (SELECT nombre, salario FROM empleados WHERE salario > 4000) AS sub;

-- Salario promedio por departamento
SELECT sub.departamento, sub.salario_promedio
FROM (
    SELECT departamento, AVG(salario) AS salario_promedio
    FROM empleados
    GROUP BY departamento
) AS sub;
```

Subconsulta en HAVING

```sql
SELECT departamento, SUM(salario) AS total_salarios
FROM empleados
GROUP BY departamento
HAVING SUM(salario) > (SELECT AVG(salario) FROM empleados);
```

ORDEN DE EVALUACIÓN SQL

```
1. FROM       - Selecciona la tabla
2. WHERE      - Filtra registros individuales
3. GROUP BY   - Agrupa registros
4. HAVING     - Filtra grupos
5. SELECT     - Selecciona columnas (incluye alias)
6. DISTINCT   - Elimina duplicados
7. ORDER BY   - Ordena resultados
8. LIMIT      - Limita número de resultados
```

⚠️ **IMPORTANTE:** No puedes usar alias de columna en WHERE, GROUP BY o HAVING porque SELECT se evalúa después. Sí puedes usarlos en ORDER BY.

```sql
-- ❌ ERROR - Alias no disponible en WHERE
SELECT salario * 1.1 AS nuevo_salario
FROM empleados
WHERE nuevo_salario > 5000;

-- ✅ CORRECTO
SELECT salario * 1.1 AS nuevo_salario
FROM empleados
WHERE salario * 1.1 > 5000;

-- ✅ CORRECTO - Alias disponible en ORDER BY
SELECT salario * 1.1 AS nuevo_salario
FROM empleados
ORDER BY nuevo_salario DESC;
```

EJEMPLOS COMBINADOS

Consulta completa con todas las cláusulas

```sql
SELECT departamento,
    COUNT(*) AS num_empleados,
    AVG(salario) AS salario_promedio
FROM empleados
WHERE edad > 25
GROUP BY departamento
HAVING COUNT(*) > 3
ORDER BY salario_promedio DESC
LIMIT 5;
```

Con subconsulta y funciones

```sql
SELECT nombre,
    salario,
    ROUND((salario - (SELECT AVG(salario) FROM empleados)), 2) AS diferencia
FROM empleados
WHERE salario > (SELECT AVG(salario) FROM empleados)
ORDER BY diferencia DESC;
```

Búsqueda avanzada con LIKE y OR

```sql
SELECT * FROM empleados
WHERE (nombre LIKE 'A%' OR nombre LIKE 'M%')
  AND departamento IN ('Ventas', 'Marketing')
  AND salario BETWEEN 3000 AND 6000
ORDER BY salario DESC;
```

TIPS Y BUENAS PRÁCTICAS

✅ **Usa DISTINCT con cuidado** - Puede afectar el rendimiento en tablas grandes

✅ **Siempre usa ORDER BY con LIMIT** - Sin ORDER BY, el orden no está garantizado

✅ **WHERE antes de HAVING** - WHERE filtra antes de agrupar (más eficiente)

✅ **Alias claros** - Usa nombres descriptivos: `AS total_ventas` en lugar de `AS t`

✅ **Paréntesis en condiciones complejas** - Mejora legibilidad y evita errores

✅ **COUNT(*) vs COUNT(columna)** - `COUNT(*)` cuenta todos, `COUNT(columna)` ignora NULL

✅ **IS NULL vs = NULL** - Siempre usa `IS NULL`, no `= NULL`

Fin del Cheatsheet 🎯