

Ejercicios para Consultas en SQL

SINTAXIS BÁSICA: SELECT...FROM + USO DE WHERE, AND, OR, BETWEEN

Base de datos: escuela

- 1) Seleccionar todos los datos de la tabla alumnos:

```
SELECT * FROM escuelas.alumnos;
```

- 2) Seleccionar solamente el legajo y el nombre de los alumnos:

```
SELECT legajo, nombre FROM escuelas.alumnos;
```

- 3) Mostrar todos los datos de aquellos alumnos aprobados (con notas mayores o iguales a 7)

```
SELECT *  
FROM escuelas.alumnos  
WHERE nota >= 7;
```

- 4) Mostrar el id y el nombre de aquellas escuelas cuya capacidad sea inferior a 200 (no mostrar la columna capacidad).

```
SELECT id, nombre  
FROM escuelas.escuelas  
WHERE capacidad < 200;
```

- 5) Mostrar el nombre y la nota de aquellos alumnos cuya nota se encuentre entre 8 y 10

```
SELECT nombre, nota  
FROM escuelas.alumnos  
WHERE nota >= 8 AND nota <= 10;
```

- 6) Repetir el ejercicio anterior, utilizando BETWEEN

```
SELECT nombre, nota  
FROM escuelas.alumnos  
WHERE nota BETWEEN 8 AND 10;
```

- 7) Mostrar el nombre, la localidad y la provincia de aquellas escuelas situadas en Buenos Aires o Jujuy

```
SELECT nombre, localidad, provincia  
FROM escuelas.escuelas  
WHERE provincia = "Buenos Aires" OR provincia = "Jujuy";
```

SINTAXIS BÁSICA: SELECT...FROM + USO DE LIKE, NOT LIKE Y %

- 8) Mostrar todos los datos de los alumnos llamados Pedro González

```
SELECT *  
FROM escuelas.alumnos  
WHERE nombre LIKE 'Pedro González';
```

- 9) Repetir el ejercicio anterior, pero con aquellos que no se llamen Pedro González

```
SELECT *  
FROM escuelas.alumnos  
WHERE nombre NOT LIKE 'Pedro González';
```

- 10) Mostrar todos los datos de los alumnos cuyo nombre comience con R

```
SELECT *  
FROM escuelas.alumnos  
WHERE nombre LIKE 'R%';
```

- 11) Mostrar todos los datos de los alumnos cuyo nombre termine con A

```
SELECT *  
FROM escuelas.alumnos  
WHERE nombre LIKE '%A';
```

- 12) Mostrar todos los datos de los alumnos cuyo nombre contenga una M

```
SELECT *  
FROM escuelas.alumnos  
WHERE nombre LIKE '%M%';
```

USOS DE JOIN Y ALIAS PARA TABLAS Y CAMPOS

- 13) Mostrar el legajo, el nombre del alumno y el nombre de la escuela de todos los alumnos

```
SELECT alu.legajo, alu.nombre, esc.nombre  
FROM alumnos alu  
INNER JOIN escuelas esc ON alu.id_escuela = esc.id;
```

- 14) Modificar el ejercicio anterior utilizando alias de columnas de modo tal que los datos se muestren de esta manera:

	Nro legajo	Nombre y Apellido	Escuela
▶	101	Juan Perez	Normal 1
	105	Pedro González	Normal 1
	190	Roberto Luis Sánchez	Normal 1

```
SELECT alu.legajo AS 'Nro legajo', alu.nombre AS 'Nombre y Apellido',  
esc.nombre AS 'Escuela'  
FROM alumnos alu  
INNER JOIN escuelas esc ON alu.id_escuela = esc.id;
```

- 15) Mostrar todos los alumnos, tengan o no escuela asignada.

```
SELECT alu.legajo, alu.nombre, esc.nombre  
FROM alumnos alu  
LEFT JOIN escuelas esc ON alu.id_escuela = esc.id;
```

- 16) Mostrar todas las escuelas con el nombre de cada alumno (aunque no tengan escuela asignada).

```
SELECT esc.*, alu.nombre
FROM escuelas esc
RIGHT JOIN alumnos alu ON esc.id = alu.id_escuela;
```

USO DE IS NULL / IS NOT NULL

17) Mostrar todos los datos de los alumnos que tengan notas.

```
SELECT *
FROM escuelas.alumnos
WHERE nota IS NOT NULL;
```

18) Mostrar todos los datos de los alumnos que no tengan notas.

```
SELECT *
FROM escuelas.alumnos
WHERE nota IS NULL;
```

ALTER TABLE

19) Realizar lo siguiente:

- Agregar a través de Alter Table una en la tabla **escuelas** columna llamada "Partido", a la derecha de Localidad con una cadena vacía como valor por defecto (armar la sentencia a través de Alter Table y ejecutar desde la consulta).

```
ALTER TABLE `escuelas`.`escuelas`
ADD COLUMN `partido` VARCHAR(45) NULL DEFAULT '' AFTER `localidad`;
```

- Ejecutar una consulta donde se vean todos los campos para confirmar que se ha agregado el campo "partido".
- Eliminar esa columna utilizando Alter Table, no es necesario ejecutarlo desde la consulta.

INSERT STATEMENT (operación a nivel de registro)

20) Agregar una nueva escuela utilizando Insert Statement (no agregar el ID). Ejecutar desde la

	id	nombre	localidad	provincia	capacidad
	5	Esc. Nº 2 Tomás Santa coloma	Capital Federal	Buenos Aires	250

```
INSERT INTO `escuelas`.`escuelas`
(`nombre`, `localidad`, `provincia`, `capacidad`)
VALUES
('Esc. Nº 2 Tomás Santa coloma', 'Capital Federal', 'Buenos Aires', 250);
```

consulta:

CREATE STATEMENT

- 21) Utilizando Create Statement duplicar la tabla Alumnos. Analizar el código que se genera y responder: ¿Qué función cumple el Auto_increment de la última línea?

```
CREATE TABLE `alumnos_dos` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `id_escuela` int(11) DEFAULT NULL,  
  `legajo` int(11) DEFAULT NULL,  
  `nombre` varchar(45) DEFAULT NULL,  
  `nota` decimal(10,0) DEFAULT NULL,  
  `grado` int(11) DEFAULT NULL,  
  `email` varchar(45) NOT NULL,  
  PRIMARY KEY (`id`),  
  KEY `fk_escuela_id_idx` (`id_escuela`)  
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8;
```

LIMIT Y ORDER BY

- 22) Obtener un ranking de las primeras 3 escuelas de mayor capacidad.

```
SELECT *  
FROM escuelas  
ORDER BY capacidad DESC  
LIMIT 3;
```

FUNCIONES DE AGREGACIÓN Y AGRUPAMIENTO / USO DE IN

- 23) Contar la cantidad de alumnos de la tabla homónima. Llamar a la columna “Cantidad de alumnos”.

```
SELECT COUNT(*) AS 'Cantidad de alumnos'  
FROM alumnos;
```

- 24) Repetir la consulta anterior consultando solamente cuya nota sea menor a 7.

```
SELECT COUNT(*) AS 'Cantidad de alumnos'  
FROM alumnos  
WHERE nota < 7;
```

- 25) Obtener la capacidad total de las escuelas de la provincia de Buenos Aires

```
SELECT SUM(capacidad) AS 'Capacidad total'  
FROM escuelas  
WHERE provincia LIKE 'Buenos Aires';
```

- 26) Repetir el ejercicio anterior pero solamente con las escuelas de Córdoba y Jujuy

```
SELECT SUM(capacidad) AS 'Capacidad total'  
FROM escuelas  
WHERE provincia IN ('Córdoba', 'Jujuy');
```

27) Obtener el promedio de notas de los alumnos aprobados con más de 7

```
SELECT AVG(nota) AS 'Promedio de notas'
FROM alumnos
WHERE nota > 7;
```

28) Obtener la capacidad máxima y la capacidad mínima de alumnos

```
SELECT MAX(capacidad) AS 'Capacidad máxima',
MIN(capacidad) AS 'Capacidad mínima'
FROM escuelas;
```

29) Obtener el total de capacidad de las escuelas por provincia

```
SELECT provincia, SUM(capacidad) AS 'Suma de capacidad'
FROM escuelas
GROUP BY provincia;
```

30) Obtener la cantidad de alumnos por grado

```
SELECT grado, COUNT(grado) AS 'Cantidad de alumnos'
FROM alumnos
GROUP BY grado;
```

DIFERENCIAS ENTRE HAVING Y WHERE

31) Comparar las diferencias entre el uso de WHERE y HAVING: Mostrar las escuelas y la nota máxima para cada una siempre y cuando sean mayores o iguales a 7.

```
115 • SELECT e.nombre AS Escuela,
116         MAX(nota) AS 'Mayor nota'
117 FROM escuelas e
118 INNER JOIN alumnos a ON e.id = a.id_escuela
119 WHERE nota >= 7
120 -- WHERE nota = 10
121 GROUP BY e.nombre
122 ;
123 |
```

Con muchos registros esta forma de filtrar es poco eficiente... acá aparece HAVING. El HAVING pregunta sobre los resultados del operador COUNT, AVG, MAX, ETC. Primero hace la consulta y luego filtra.

```
124 • SELECT e.nombre AS Escuela,
125         MAX(nota) AS 'Mayor nota'
126 FROM escuelas e
127 INNER JOIN alumnos a ON e.id = a.id_escuela
128 GROUP BY e.nombre
129 HAVING MAX(nota) >= 7
130 ;
```

Se puede decir que el having es el “where” del agrupamiento.

SUBCONSULTAS

32) Mostrar la información de las escuelas cuyos alumnos tengan una nota igual a 10, utilizando una subconsulta.

```
1 • SELECT *
2 FROM escuelas
3 -- WHERE escuelas.id IN (1,2)
4 WHERE escuelas.id IN
5 (SELECT id_escuela FROM alumnos WHERE nota = 10);
```