Unit 7 - Intro SQL

GUILTY PLEASURES

Eating baby food

Stalking other people's social media

Dressing as "Alice in Wonderland" while watching the movie

Intro to SQL

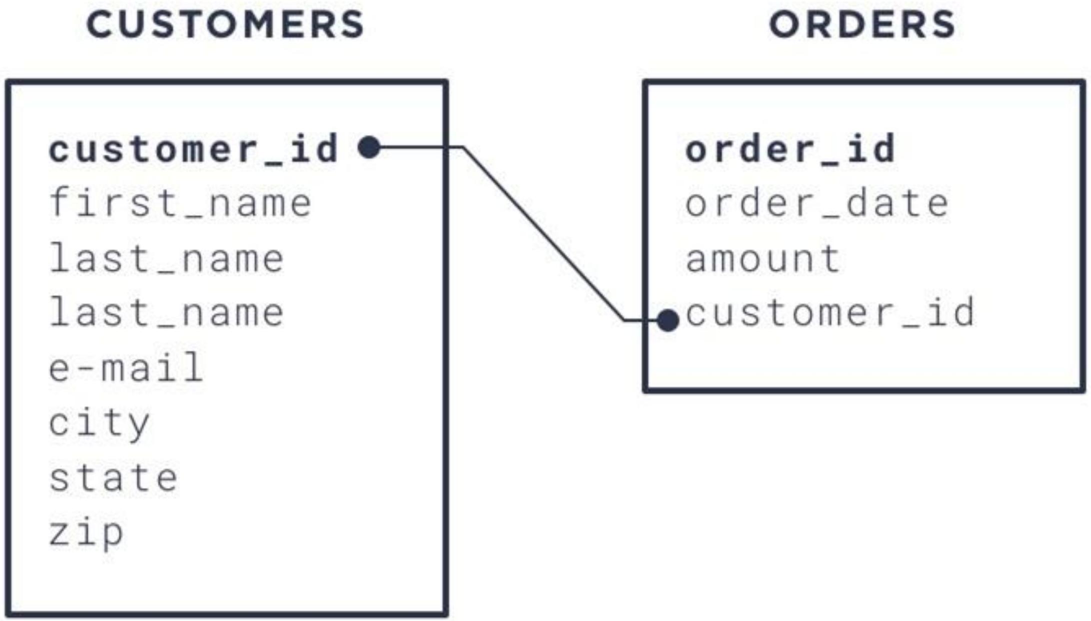
DATA ANALYTICS | IRONHACK

What is SQL?

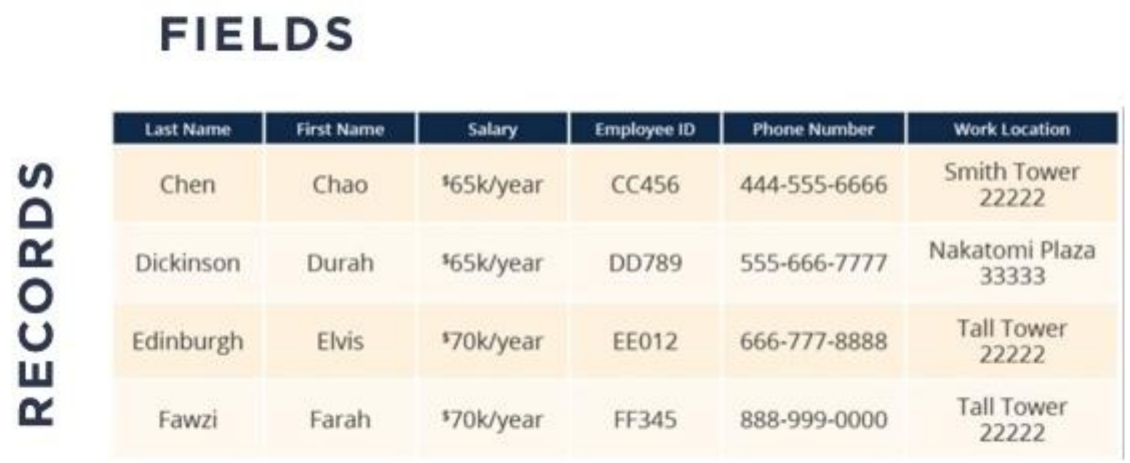
Structured Query Language

- Developed in the 1970s by IBM researchers
- Was deemed as a standard language for Relational databases by ANSI and ISO
- Most prominent SQL standards used in the industry include standard SQL, T-SQL, P-SQL
- Major players in relational databases are Oracle, Microsoft, Amazon, Google

What are Tables? Relations



What are fields and records?



FIELDS

RECORDS

Last Name	First Name	Salary	Employee ID	Phone Number	Work Location
Chen	Chao	$65k/year	CC456	444-555-6666	Smith Tower 22222
Dickinson	Durah	$65k/year	DD789	555-666-7777	Nakatomi Plaza 33333
Edinburgh	Elvis	$70k/year	EE012	666-777-8888	Tall Tower 22222
Fawzi	Farah	$70k/year	FF345	888-999-0000	Tall Tower 22222

Relational databases save space*

*minimize data redundancy

Employees_stores Non-relational model

emp_ID	First_name	Last_name	Store_id	Store_city	Store_type	Store_address
1	Vladimir	Popov	A	London	Showcase	Frank St. 26
2	Cinar	Horton	B	Norwich	Regular	Newton Av. 3

Relational model

Employees_stores

emp_id	**store_id**
1	**A**
2	**B**

If an employee changes store, we just update this.

Employees

emp_id	First_name	Last_name
1	Vladimir	Popov
2	Cinar	Horton

We can have old employees.

Stores

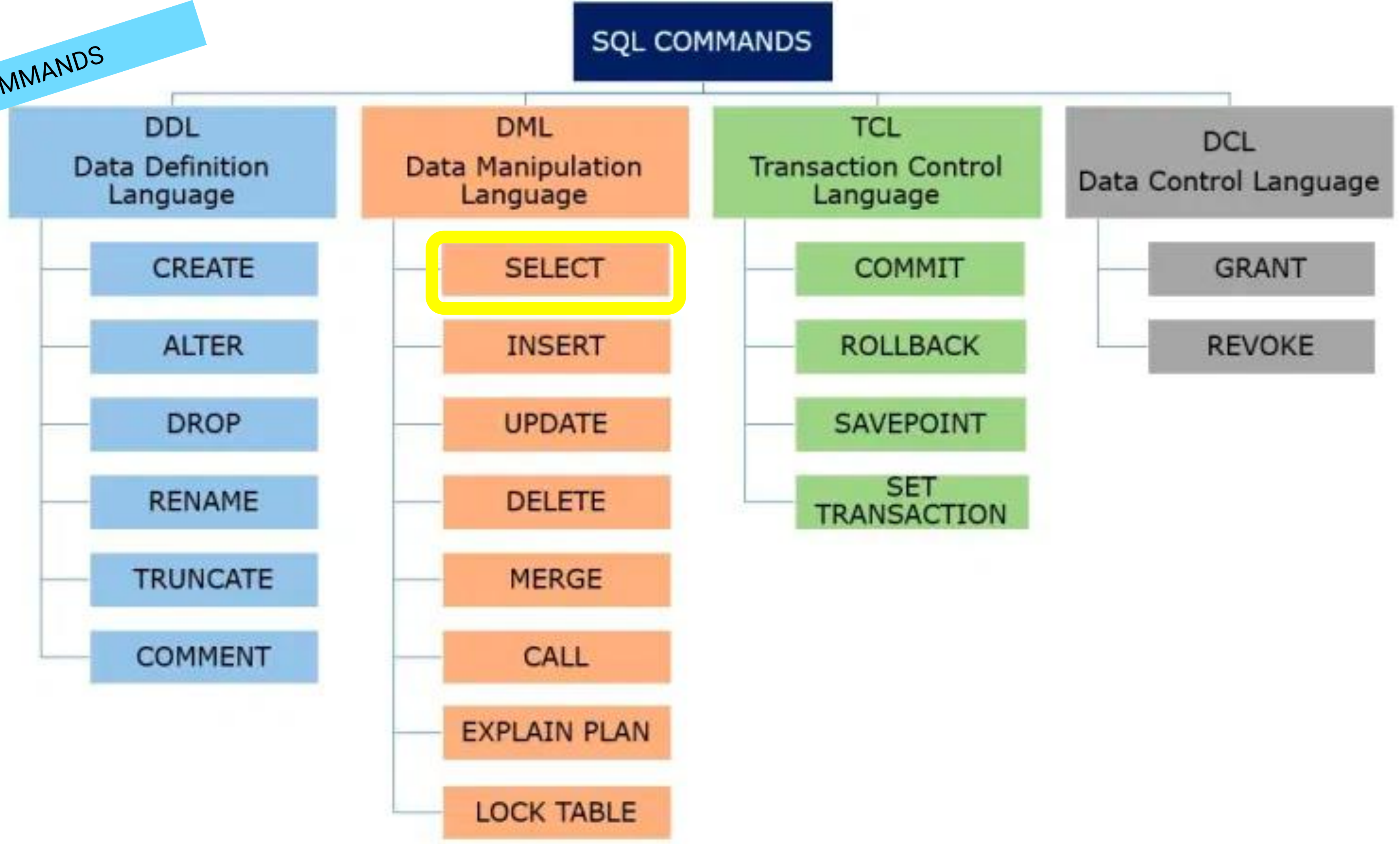
Store_id	city	type	address
A	London	Showcase	Frank St. 26
B	Norwich	Regular	Newton Av. 3

Only 1 row per store. Easy to maintain and update. We can have stores without employees.

SQL INSTRUCTION

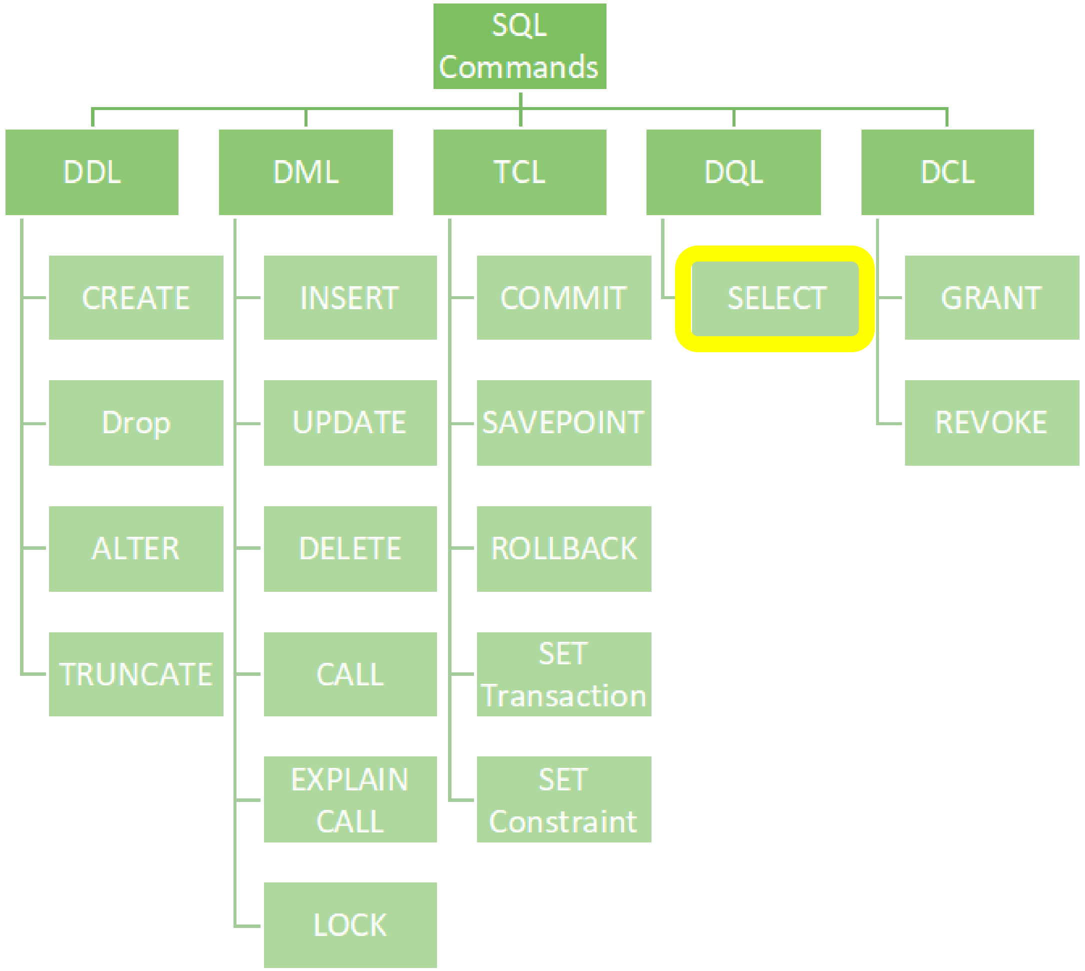
Command + Clause + Operators + Functions

COMMANDS



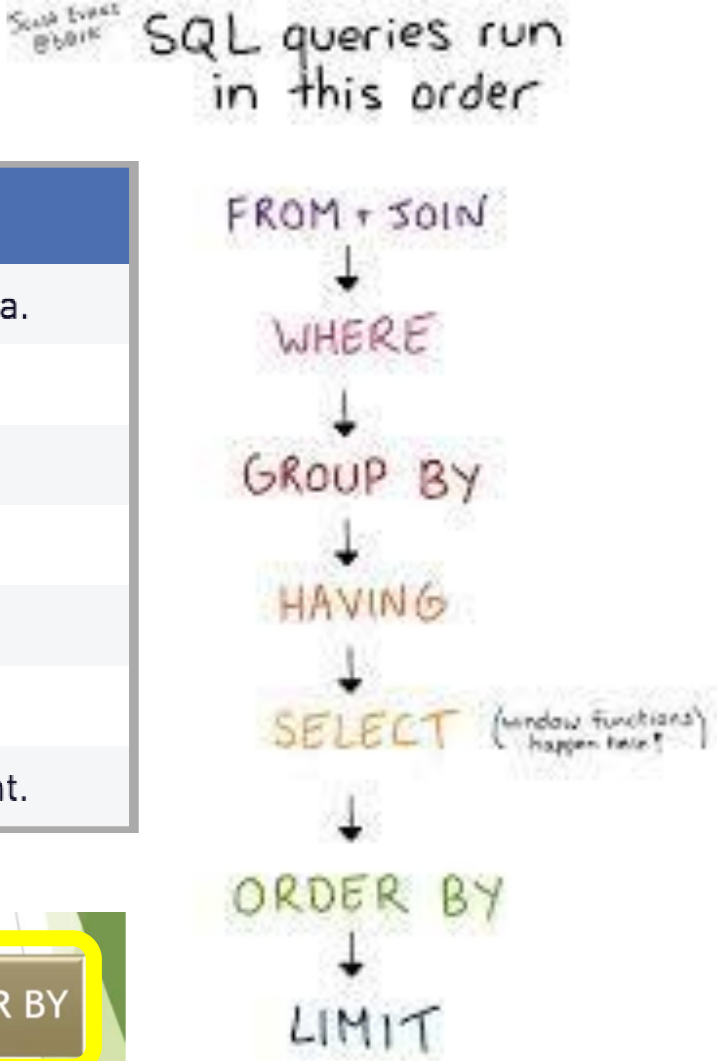
SQL COMMANDS
DDL
Data Definition Language
DML
Data Manipulation Language
TCL
Transaction Control Language
DCL
Data Control Language
CREATE
ALTER
DROP
RENAME
TRUNCATE
COMMENT
SELECT
INSERT
UPDATE
DELETE
MERGE
CALL
EXPLAIN PLAN
LOCK TABLE
COMMIT
ROLLBACK
SAVEPOINT
SET TRANSACTION
GRANT
REVOKE

COMMANDS


SQL Commands
DDL
DML
TCL
DQL
DCL
CREATE
Drop
ALTER
TRUNCATE
INSERT
UPDATE
DELETE
CALL
EXPLAIN CALL
LOCK
COMMIT
SAVEPOINT
ROLLBACK
SET Transaction
SET Constraint
SELECT
GRANT
REVOKE

CLAUSES

ORDER	CLAUSE	FUNCTION
1	from	Choose and join tables to get base data.
2	where	Filters the base data.
3	group by	Aggregates the base data.
4	having	Filters the aggregated data.
5	select	Returns the final data.
6	order by	Sorts the final data.
7	limit	Limits the returned data to a row count.


Julia Evans @b0rk
SQL queries run in this order
FROM + JOIN
WHERE
GROUP BY
HAVING
SELECT (window functions happen here!)
ORDER BY
LIMIT

FROM
WHERE
GROUP BY
HAVING
ORDER BY

Operators and built in functions SQL

DATA ANALYTICS | IRONHACK

MULTIPLE CONDITIONS IN WHERE | AND, OR

- You can use multiple conditions in a WHERE clausule:

```
SELECT * FROM bank.loans
WHERE status NOT IN (‘B’,’b’) AND amount > 100000;


SELECT * FROM bank.loans
WHERE amount > 100000 AND amount < 200000;


SELECT distinct A2 FROM bank.district
WHERE  A2 IN (‘Benesov’,’Beroun’) OR A4 < 75000
LIMIT 10;
```

LOGICAL OPERATORS | AND, OR

- You can use logical operators in a WHERE clausule:

```
select * from bank.loan
where status = 'B' and amount > 100000;

select * from bank.loan
where status = 'B' and amount > 100000 and duration <= 24;

select * from bank.loan
where status = 'B' or status = 'D';
```

NUMERIC FUNCTIONS | ROUND, COUNT, MIN, MAX, FLOOR, CEILING

- You can those functions in your queries:

```
select order_id, round(amount/1000,2)
from bank.order;

select count(order_id) from bank.order;

select max(amount) from bank.order;
select min(amount) from bank.order;

select floor(avg(amount)) from bank.order;
select ceiling(avg(amount)) from bank.order;
```

Datetime functions and processing order

DATA ANALYTICS | IRONHACK

CONVERTING OBJECTS TO DATE, DATETIME | CONVERT()

- Now, let’s see how can change how some objects are being displayed:

```
SELECT account_id, district_id, CONVERT(date, date), frequency FROM
bank.account;
```

FORMATTING DATES | DATE_FORMAT()

- We can change the output format of a date with function DATE_FORMAT(string, format):

```
SELECT DATE_FORMAT(CONVERT(date, date), ‘%Y-%M-%D’)) FROM bank.loan;

SELECT DATE_FORMAT(CONVERT(date, date), ‘%Y’)) AS ‘Year’  FROM bank.loan;
```

- %Y -> XXXX, %y -> XX
- %M -> ‘November’, %m -> ‘11’
- %D -> ‘X th’, %d -> X

Date time formats

Nulls and CASE statement

DATA ANALYTICS | IRONHACK

DEALING WITH NULL VALUES

- In order to know if you have a NULL, you can use the function ISNULL(argument), which returns:
    - 0 if the argument not NULL
    - 1 if the argument is NULL

```
SELECT ISNULL(‘Hello’);
SELECT SUM(ISNULL(card_id)) FROM bank.card;
SELECT distinct k_symbol FROM bank.order;
SELECT * FROM bank.order
WHERE k_symbol = NULL; # Blank spaces are not NULL!!!
SELECT * FROM bank.order WHERE k_symbol IS NOT NULL AND k_symbol =  ‘ ‘;
```

UTILITY OF CASE STATEMENTS

- Case statements are a way to replace the column values by new values (ONLY WHEN DISPLAYED, not in the table)

USING CASE STATEMENTS

- An example:

```
SELECT loan_id, account_id,
CASE
        WHEN status = “A” THEN “Good - Contract finished”
        WHEN status = “B” THEN “Defaulter - Contract finished”
        WHEN status = “C” THEN “Good - Contract running”
        ELSE “In debt - Contract running”
END AS “Status_Description”
FROM bank.loan;
```

Distinct in between like REGEXP

DATA ANALYTICS | IRONHACK

DISTINCT

- As we have seen previously, this command allow us to get the unique values from a columns as “unique()” function from Pandas.

```
SELECT DISTINCT A3 FROM bank.district;
```

IN

- This command allow us to check for multiple values in a “list”

```
SELECT * FROM bank.account
WHERE district_id IN (1,2,3,4,5);
```

BETWEEN

- This command allows us to check ranges, although the same can be accomplished with logical operators.

```
SELECT * FROM bank.loan
WHERE (amount - payments) BETWEEN 1000 AND 10000;

SELECT * FROM bank.loan
WHERE (amount - payments) > 1000 AND (amount - payments) < 10000;
```

LIKE

- This command allows to search for column values which have a “pattern” called “mask”.

- It has two “wildcards”:
  - % -> zero, one or multiples characters
  - _ -> single character

```
SELECT * FROM bank.district
WHERE A3 LIKE ‘north%’;

SELECT * FROM bank.district
WHERE A3 LIKE ‘north_M%’;
```

REGEXP (REGularEXPressions)

- This term encapsulates a whole set of wildcards in order to look for patterns in strings.

- The way to look for regular expressions is typing “regexp” and pattern you are looking for. Some “regex” popular patterns are:
  - ‘^’ -> beginning of the string
  - ‘$’ -> end of the string
  - ‘|’ -> OR

```
SELECT * FROM bank.district WHERE A2 regexp '^B';
SELECT * FROM bank.district WHERE A2 regexp 'ov$';
SELECT distinct k_symbol FROM bank.order WHERE k_symbol regexp ‘ip|is’
```

Revisiting ORDER BY

DATA ANALYTICS | IRONHACK

ORDER BY WITH MULTIPLE COLUMNS

- ORDER BY accepts several columns at the same time separated by comma.

- When several columns are used, the output is first sorted according to the first column in the list, then according to the second and so on.

- Precedence of values:
    - Null values
    - Special characters
    - Numbers
    - Letters

ORDER BY WITH MULTIPLE COLUMNS

- ORDER BY accepts several columns at the same time separated by comma.
- When several columns are used, the output is first sorted according to the first column in the list, then according to the second and so on.

EXAMPLES

```
SELECT * FROM bank.orders ORDER BY account_id, bank_to, k_symbol;
SELECT * FROM bank.orders ORDER BY account_id asc,  k_symbol desc;
SELECT nombre, usuario_id
FROM TiktokDB.Usuarios;
```

```
SELECT * FROM Usuarios WHERE pais = 'Espana';
```

```
INSERT INTO Usuarios (nombre_usuario,
email, fecha_registro) VALUES
('luis_artist', 'luis@example.com');
```

```
SELECT usuario_id, nombre_usuario
FROM TikTokDB.Usuarios WHERE
pais LIKE "Méx%";
```

```
SELECT * FROM Videos WHERE duracion_segundos > 'largo';
```

```
DELETE FROM Likes WHERE like_id = 3 AND video_id =
2 OR usuario_id = 4;
```

```
INSERT INTO Likes (video_id, usuario_id) VALUES (3);
```

```
SELECT * FROM Comentarios ORDERY BY
fecha_comentario DESC;
```

```
UPDATE Videos SET titulo = "Nuevo título" WHERE id = 5;
```

```
INSERT INTO Comentarios (video_id, usuario_id,
texto_comentario) VALUES (2, 3, NULL);
```

```sql
DELETE FROM Videos WHERE usuario_id IN ('1', '2', '3');
```

```
UPDATE Usuarios SET pais = México WHERE usuario_id = 2;
```

```
INSERT INTO Videos (titulo, descripcion,
fecha_publicacion, duracion_segundos) VALUES
('Video', 'Descripción', '2021-05-01', 'cinco minutos');
```

```
INSERT INTO Usuarios (nombre, email, fecha_registro, pais, likes)
VALUES
('pepe_gamer', 'pepe@example.com', '2021-05-10', 'España',1),
('maria_dancer', 'maria@example.com', '2020-08-22', 'México',3);
```